aws

Entwicklerhandbuch

Amazon Data Firehose

Amazon Data Firehose: Entwicklerhandbuch

Table of Contents

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Was ist Amazon Data Firehose?

Amazon Data Firehose ist ein vollständig verwalteter Service für die Bereitstellung von Echtzeit-Streaming-Daten an Ziele wie Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service, Amazon OpenSearch Serverless, Splunk, Apache Iceberg Tables und alle benutzerdefinierten HTTP-Endpunkte oder HTTP-Endpunkte unterstützter Drittanbieter wie Datadog, Dynatrace, MongoDB, New Relic, Coralogix und Elastic. LogicMonitor Mit Amazon Data Firehose müssen Sie keine Anwendungen schreiben oder Ressourcen verwalten. Sie konfigurieren Ihre Datenproduzenten so, dass sie Daten an Amazon Data Firehose senden, und Amazon Data Firehose sendet die Daten automatisch an das von Ihnen angegebene Ziel. Sie können Amazon Data Firehose auch so konfigurieren, dass Ihre Daten vor der Auslieferung transformiert werden.

Weitere Informationen zu AWS Big-Data-Lösungen finden Sie unter Big Data auf AWS. Weitere Informationen zu AWS -Streaming-Datenlösungen finden Sie unter Was sind Streaming-Daten?

Lernen Sie die wichtigsten Konzepte kennen

Wenn Sie mit Amazon Data Firehose beginnen, können Sie davon profitieren, die folgenden Konzepte zu verstehen.

Firehose-Stream

Die zugrunde liegende Einheit von Amazon Data Firehose. Sie verwenden Amazon Data Firehose, indem Sie einen Firehose-Stream erstellen und dann Daten an ihn senden. Weitere Informationen erhalten Sie unter Tutorial: Einen Firehose-Stream von der Konsole aus erstellen und Daten an einen Firehose-Stream senden.

Aufzeichnen

Die interessanten Daten, die Ihr Datenproduzent an einen Firehose-Stream sendet. Ein Datensatz kann bis zu 1000 KB groß sein.

Datenproduzent

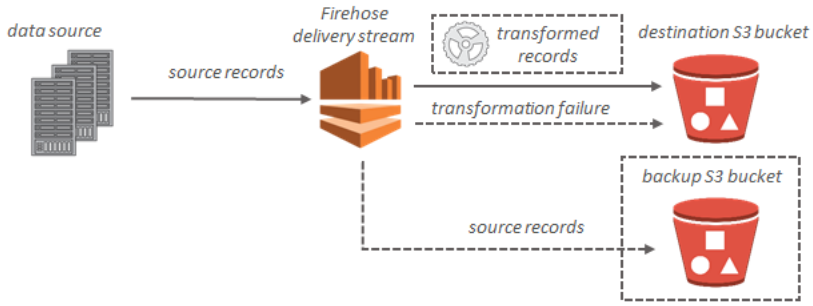
Produzenten senden Platten an Firehose-Streams. Beispielsweise ist ein Webserver, der Protokolldaten an einen Firehose-Stream sendet, ein Datenproduzent. Sie können Ihren Firehose-Stream auch so konfigurieren, dass er automatisch Daten aus einem vorhandenen Kinesis-Datenstream liest und in Ziele lädt. Weitere Informationen finden Sie unter Daten an einen Firehose-Stream senden.

Puffergröße und Pufferintervall

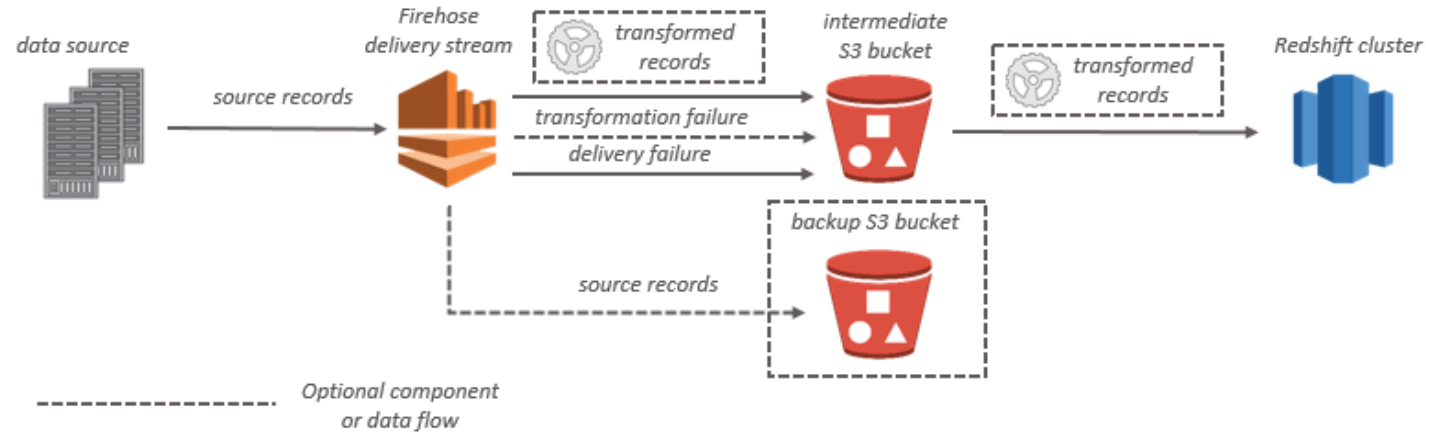
Amazon Data Firehose puffert eingehende Streaming-Daten auf eine bestimmte Größe oder für einen bestimmten Zeitraum, bevor sie an Ziele gesendet werden. Buffer Sizeist drin MBs und Buffer Interval ist in Sekunden.

Den Datenfluss in Amazon Data Firehose verstehen

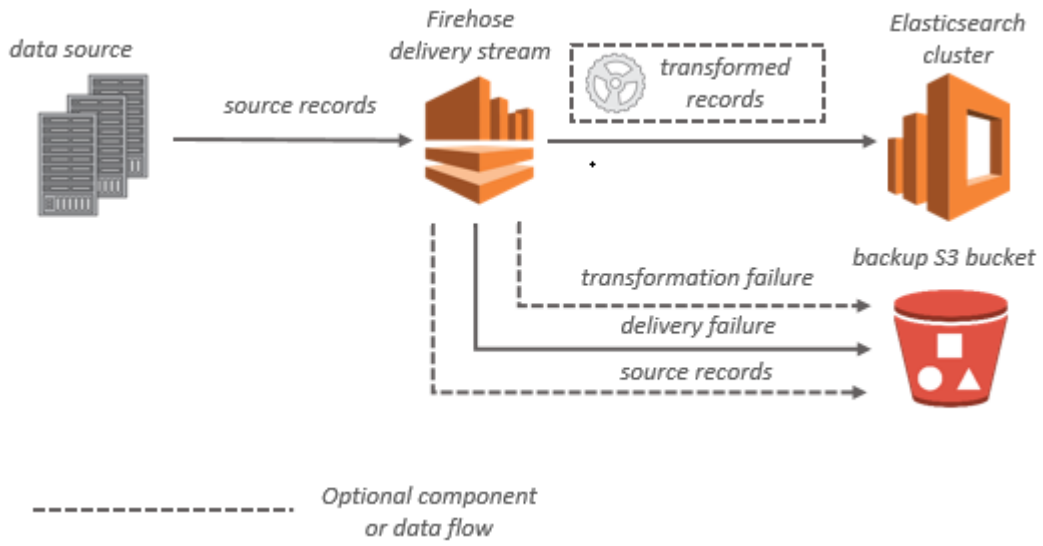
Für Amazon-S3-Ziele werden die Streaming-Daten in Ihren S3-Bucket geleitet. Wenn die Datentransformation aktiviert ist, können Sie optional Quelldaten in einem anderen Amazon-S3-Bucket sichern.



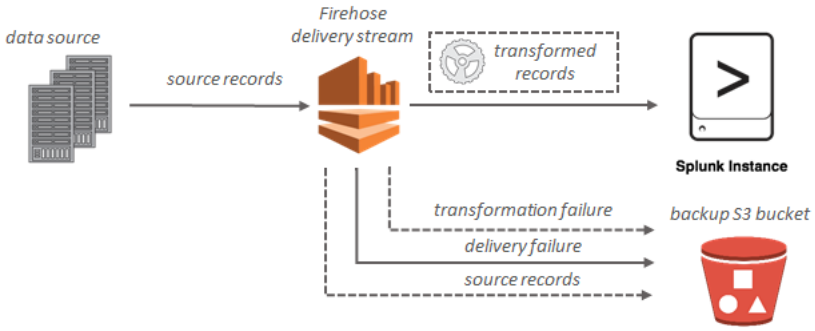
Für Amazon-Redshift-Ziele werden die Streaming-Daten zuerst in Ihren S3-Bucket geleitet. Amazon Data Firehose gibt dann einen Amazon Redshift COPY Redshift-Befehl aus, um Daten aus Ihrem S3-Bucket in Ihren Amazon Redshift Redshift-Cluster zu laden. Wenn die Datentransformation aktiviert ist, können Sie optional Quelldaten in einem anderen Amazon-S3-Bucket sichern.



Bei OpenSearch Servicezielen werden Streaming-Daten an Ihren OpenSearch Service-Cluster übermittelt und können optional gleichzeitig in Ihrem S3-Bucket gesichert werden.



Für Splunk-Ziele werden die Streaming-Daten an Splunk gesendet und können gleichzeitig optional in Ihrem S3-Bucket gesichert werden.



Firehose mit einem AWS SDK verwenden

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Codebeispiele
AWS SDK für Go	AWS SDK für Go Codebeispiele

SDK-Dokumentation	Codebeispiele
AWS SDK für Java	AWS SDK für Java Codebeispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Codebeispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Codebeispiele
AWS SDK für .NET	AWS SDK für .NET Codebeispiele
AWS SDK für PHP	AWS SDK für PHP Codebeispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Codebeispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Codebeispiele
AWS SDK für Ruby	AWS SDK für Ruby Codebeispiele
AWS SDK für Rust	AWS SDK für Rust Codebeispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Codebeispiele
AWS SDK für Swift	AWS SDK für Swift Codebeispiele

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Provide feedback (Feedback geben) auswählen.

Vollständige Voraussetzungen für die Einrichtung von Amazon Data Firehose

Bevor Sie Amazon Data Firehose zum ersten Mal verwenden, führen Sie die folgenden Aufgaben aus.

Aufgaben

- Melden Sie sich an für AWS
- (Optional) Laden Sie Bibliotheken und Tools herunter

Melden Sie sich an für AWS

Wenn Sie sich für Amazon Web Services (AWS) registrieren, wird Ihr AWS Konto automatisch für alle Dienste angemeldet AWS, einschließlich Amazon Data Firehose. Berechnet werden Ihnen aber nur die Services, die Sie nutzen.

Wenn Sie bereits ein AWS Konto haben, fahren Sie mit der nächsten Aufgabe fort. Wenn Sie kein AWS -Konto haben, führen Sie die folgenden Schritte zum Erstellen eines Kontos aus.

Um sich für ein AWS Konto zu registrieren

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Ein Teil des Anmeldevorgangs umfasst den Empfang eines Telefonanrufs oder einer Textnachricht und die Eingabe eines Bestätigungscodes auf der Telefontastatur.

   Wenn Sie sich für eine anmelden AWS-Konto, wird eine Root-Benutzer des AWS-Kontoserstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

(Optional) Laden Sie Bibliotheken und Tools herunter

Die folgenden Bibliotheken und Tools helfen Ihnen dabei, programmgesteuert und von der Befehlszeile aus mit Amazon Data Firehose zu arbeiten:

- Die Firehose-API-Operationen sind die grundlegenden Operationen, die Amazon Data Firehose unterstützt.
- AWS SDKs Für Go, Java, .NET, Node.js, Python und Ruby bieten Amazon Data Firehose-Unterstützung und Beispiele.

  Wenn Ihre Version von AWS SDK für Java keine Beispiele für Amazon Data Firehose enthält, können Sie das neueste AWS SDK auch von GitHubherunterladen.
- Das AWS Command Line Interfaceunterstützt Amazon Data Firehose. Das AWS CLI ermöglicht es Ihnen, mehrere AWS Dienste von der Befehlszeile aus zu steuern und sie mithilfe von Skripten zu automatisieren.

Tutorial: Einen Firehose-Stream von der Konsole aus erstellen

Sie können das AWS-Managementkonsole oder ein AWS SDK verwenden, um einen Firehose-Stream zu Ihrem ausgewählten Ziel zu erstellen.

Sie können die Konfiguration Ihres Firehose-Streams jederzeit nach seiner Erstellung aktualisieren, indem Sie die Amazon Data Firehose-Konsole verwenden oder. UpdateDestination Ihr Firehose-Stream bleibt so`Active`, wie Ihre Konfiguration aktualisiert wird, und Sie können weiterhin Daten senden. Die aktualisierte Konfiguration wird in der Regel innerhalb weniger Minuten wirksam. Die Versionsnummer eines Firehose-Streams wird 1 nach dem Update der Konfiguration um den Wert von erhöht. Sie wird im Namen des gelieferten Amazon-S3-Objektnamen wiedergegeben. Weitere Informationen finden Sie unter Amazon S3 S3-Objektnamenformat konfigurieren.

Führen Sie die Schritte in den folgenden Themen aus, um einen Firehose-Stream zu erstellen.

Themen

Wählen Sie Quelle und Ziel für Ihren Firehose-Stream

1. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie Create Firehose stream (Firehose-Stream erstellen) aus.
3. Wählen Sie auf der Firehose Firehose-Stream erstellen aus einer der folgenden Optionen eine Quelle für Ihren Firehose-Stream aus.

    - Direct PUT — Wählen Sie diese Option, um einen Firehose zu erstellen, in den Producer-Anwendungen direkt schreiben. Hier finden Sie eine Liste von AWS Diensten, Agenten und Open-Source-Diensten, die in Direct PUT in Amazon Data Firehose integriert sind. Diese Liste

erhebt keinen Anspruch auf Vollständigkeit, und es kann zusätzliche Dienste geben, mit denen Daten direkt an Firehose gesendet werden können.

- AWS SDK
- AWS Lambda
- AWS CloudWatch Logs
- AWS CloudWatch Events
- AWS Metrische Cloud-Streams
- AWS IoT
- AWS Eventbridge
- Amazon Simple Email Service
- Amazon SNS
- AWS WAF-Web-ACL-Protokolle
- Amazon API Gateway – Zugriffsprotokolle
- Amazon Pinpoint
- Amazon-MSK-Broker-Protokolle
- Abfrageprotokolle von Amazon Route 53 Resolver
- AWS Protokolle der Netzwerk-Firewall-Warnmeldungen
- AWS Netzwerk-Firewall-Flussprotokolle
- Amazon ElastiCache Redis SLOWLOG
- Kinesis Agent (linux)
- Kinesis Tap (windows)
- Fluentbit
- Fluentd
- Apache Nifi
- Snowflake

- Amazon Kinesis Data Streams — Wählen Sie diese Option, um einen Firehose-Stream zu konfigurieren, der einen Kinesis-Datenstream als Datenquelle verwendet. Anschließend können Sie Firehose verwenden, um Daten einfach aus einem vorhandenen Kinesis-Datenstrom zu lesen und in Ziele zu laden. Weitere Informationen zur Verwendung von Kinesis Data Streams als Datenquelle finden Sie unter Senden von Daten an einen Firehose-Stream mit Kinesis Data Streams.

- Amazon MSK — Wählen Sie diese Option, um einen Firehose-Stream zu konfigurieren, der Amazon MSK als Datenquelle verwendet. Anschließend können Sie Firehose verwenden, um Daten einfach aus einem vorhandenen Amazon MSK-Cluster zu lesen und in bestimmte S3-Buckets zu laden. Weitere Informationen finden Sie unter Senden von Daten an einen Firehose-Stream mit Amazon MSK.

4. Wählen Sie ein Ziel für Ihren Firehose-Stream aus einem der folgenden Ziele, die Firehose unterstützt.

   - OpenSearch Amazon-Dienst
   - Amazon OpenSearch Serverlos
   - Amazon Redshift
   - Amazon S3
   - Apache Iceberg-Tabellen
   - Coralogix
   - Datadog
   - Dynatrace
   - Elastic
   - HTTP-Endpunkt
   - Honeycomb
   - Logic.Monitor
   - Logz.io
   - MongoDB Cloud
   - New Relic
   - Splunk
   - Splunk Observability Cloud
   - Sumo Logic
   - Snowflake

5. Für den Firehose-Streamnamen können Sie entweder den Namen verwenden, den die Konsole für Sie generiert, oder einen Firehose-Stream Ihrer Wahl hinzufügen.

Quelleinstellungen konfigurieren

Sie können die Quelleinstellungen auf der Grundlage der Quelle konfigurieren, die Sie auswählen, um Informationen von der Konsole an einen Firehose-Stream zu senden. Sie können Quelleinstellungen für Amazon MSK und Amazon Kinesis Data Streams als Quelle konfigurieren. Für Direct PUT als Quelle sind keine Quelleinstellungen verfügbar.

Quelleinstellungen für Amazon MSK konfigurieren

Wenn Sie Amazon MSK zum Senden von Informationen an einen Firehose-Stream wählen, können Sie zwischen MSK bereitgestellten Clustern und MSK-Serverless-Clustern wählen. Anschließend können Sie Firehose verwenden, um Daten einfach aus einem bestimmten Amazon MSK-Cluster und Thema zu lesen und sie in das angegebene S3-Ziel zu laden.

Geben Sie im Abschnitt Quelleinstellungen der Seite Werte für die folgenden Felder ein.

Amazon-MSK-Cluster-Konnektivität

Wählen Sie je nach Ihrer Cluster-Konfiguration entweder die Option Private Bootstrap-Broker (empfohlen) oder Öffentliche Bootstrap-Broker. Der Apache-Kafka-Client verwendet Bootstrap-Broker als Ausgangspunkt für die Verbindung mit dem Cluster. Öffentliche Bootstrap-Broker sind für den öffentlichen Zugriff von außen vorgesehen AWS, während private Bootstrap-Broker für den Zugriff von innen vorgesehen sind. AWS Weitere Informationen über Amazon MSK finden Sie unter Amazon Managed Streaming for Apache Kafka.

Um eine Verbindung zu einem bereitgestellten oder serverless Amazon-MSK-Cluster über einen privaten Bootstrap Broker herzustellen, muss der Cluster alle der folgenden Anforderungen erfüllen.

- Der Cluster muss aktiv sein.
- Der Cluster muss IAM als eine seiner Zugriffskontrollmethoden verwenden.
- Private Multi-VPC-Konnektivität muss für die IAM-Zugriffskontrollmethode aktiviert sein.
- Sie müssen diesem Cluster eine ressourcenbasierte Richtlinie hinzufügen, die dem Firehose-Service Principal die Erlaubnis erteilt, den Amazon MSK-API-Vorgang aufzurufen. `CreateVpcConnection`

Um über einen öffentlichen Bootstrap-Broker eine Verbindung zu einem bereitgestellten Amazon-MSK-Cluster herzustellen, muss der Cluster alle der folgenden Anforderungen erfüllen.

- Der Cluster muss aktiv sein.

- Der Cluster muss IAM als eine seiner Zugriffskontrollmethoden verwenden.
- Der Cluster muss öffentlich zugänglich sein.

MSK-Cluster-Konto

Sie können das Konto auswählen, in dem sich der Amazon MSK-Cluster befindet. Dies kann eines der folgenden sein.

- Girokonto — Ermöglicht es Ihnen, Daten aus einem MSK-Cluster in das AWS Girokonto aufzunehmen. Dazu müssen Sie den ARN des Amazon MSK-Clusters angeben, aus dem Ihr Firehose-Stream Daten liest.
- Kontoübergreifend — Ermöglicht es Ihnen, Daten aus einem MSK-Cluster in ein anderes Konto aufzunehmen. AWS Weitere Informationen finden Sie unter Kontenübergreifender Versand von Amazon MSK.

Topic

Geben Sie das Apache Kafka-Thema an, aus dem Firehose Firehose-Stream Daten aufnehmen soll. Sie können dieses Thema nicht aktualisieren, nachdem die Firehose-Stream-Erstellung abgeschlossen ist.

> **Note**
>
> Firehose dekomprimiert automatisch Apache Kafka-Nachrichten.

Quelleinstellungen für Amazon Kinesis Data Streams konfigurieren

Konfigurieren Sie die Quelleinstellungen für Amazon Kinesis Data Streams, um Informationen an einen Firehose-Stream zu senden, wie folgt.

> **Important**
>
> Wenn Sie die Kinesis Producer Library (KPL) verwenden, um Daten an einen Kinesis Data Stream zu schreiben, können Sie die Aggregation dazu verwenden, die an diesen Kinesis Data Stream geschriebenen Datensätze zu kombinieren. Wenn Sie diesen Datenstream dann als Quelle für Ihren Firehose-Stream verwenden, deaggregiert Amazon Data Firehose die Datensätze, bevor es sie an das Ziel übermittelt. Wenn Sie Ihren Firehose-Stream so konfigurieren, dass er die Daten transformiert, deaggregiert Amazon Data Firehose die Datensätze, bevor es sie an übermittelt. AWS Lambda Weitere Informationen finden Sie unter

Entwickeln von Amazon-Kinesis-Data-Streams-Produzenten mit der Kinesis Producer Library und Aggregation.

Wählen Sie unter den Quelleinstellungen einen vorhandenen Stream in der Kinesis-Datenstream-Liste aus, oder geben Sie einen Datenstream-ARN im Format `arn:aws:kinesis:[Region]:[AccountId]:stream/[StreamName]` ein.

Wenn Sie noch keinen bestehenden Datenstream haben, wählen Sie Create, um einen neuen über die Amazon Kinesis Kinesis-Konsole zu erstellen. Möglicherweise benötigen Sie eine IAM-Rolle, die über die erforderlichen Berechtigungen für den Kinesis-Stream verfügt. Weitere Informationen finden Sie unter ???. Nachdem Sie einen neuen Stream erstellt haben, wählen Sie das Aktualisierungssymbol, um die Kinesis-Stream-Liste zu aktualisieren. Wenn Sie eine große Anzahl an Streams haben, können Sie die Liste mit Filter by name (Nach Namen filtern) filtern.

> **Note**
>
> Wenn Sie einen Kinesis-Datenstream als Quelle für einen Firehose-Stream konfigurieren, sind Amazon Data Firehose `PutRecord` und der `PutRecordBatch` Betrieb deaktiviert. Verwenden Sie in diesem Fall die Kinesis Data Streams `PutRecord` and `PutRecords` Operations, um Ihrem Firehose-Stream Daten hinzuzufügen.

Amazon Data Firehose beginnt mit dem Lesen von Daten von der `LATEST` Position Ihres Kinesis-Streams. Weitere Informationen zu den Positionen von Kinesis Data Streams finden Sie unter GetShardIterator.

Amazon Data Firehose ruft den Kinesis Data Streams GetRecordsStreams-Vorgang einmal pro Sekunde für jeden Shard auf. Wenn jedoch die vollständige Sicherung aktiviert ist, ruft Firehose den Kinesis Data Streams `GetRecords` Streams-Vorgang zweimal pro Sekunde für jeden Shard auf, einen für das primäre Lieferziel und einen weiteren für ein vollständiges Backup.

Mehr als ein Firehose-Stream kann aus demselben Kinesis-Stream lesen. Andere Kinesis-Anwendungen (Konsumenten) können ebenfalls Daten aus demselben Stream lesen. Jeder Anruf von einem Firehose-Stream oder einer anderen Verbraucheranwendung wird auf das allgemeine Drosselungslimit für den Shard angerechnet. Planen Sie Ihre Anwendungen mit Bedacht, um eine Drosselung zu vermeiden. Weitere Informationen zu den Volume-Limits bei Kinesis Data Streams finden Sie unter Amazon Kinesis Streams Limits.

Fahren Sie mit dem nächsten Schritt fort, um die Datensatztransformation und die Formatkonvertierung zu konfigurieren.

(Optional) Konfiguration der Datensatztransformation und Formatkonvertierung

Konfigurieren Sie Amazon Data Firehose, um Ihre Datensatzdaten zu transformieren und zu konvertieren.

Wenn Sie Amazon MSK als Quelle für Ihren Firehose-Stream wählen.

Geben Sie im Abschnitt Quelldatensätze mit AWS Lambda transformieren Werte für das folgende Feld an.

1. Datentransformation

   Um einen Firehose-Stream zu erstellen, der eingehende Daten nicht transformiert, aktivieren Sie nicht das Kontrollkästchen Datentransformation aktivieren.

   Um eine Lambda-Funktion anzugeben, die Firehose aufrufen und verwenden soll, um eingehende Daten vor der Übertragung zu transformieren, aktivieren Sie das Kontrollkästchen Datentransformation aktivieren. Sie können eine neue Lambda-Funktion mit einem der Lambda-Vorlage konfigurieren oder eine vorhandene Lambda-Funktion auswählen. Ihre Lambda-Funktion muss das von Firehose benötigte Statusmodell enthalten. Weitere Informationen finden Sie unter Transformieren Sie Quelldaten in Amazon Data Firehose.
2. Machen Sie im Bereich Convert record format (Datensatzformat konvertieren) Angaben im folgenden Feld:

   Konvertierung des Datensatzformats

   Um einen Firehose-Stream zu erstellen, der das Format der eingehenden Datensätze nicht konvertiert, wählen Sie Disabled.

   Um das Format der eingehenden Datensätze zu konvertieren, wählen Sie Enabled (Aktiviert). Geben Sie dann das gewünschte Ausgabeformat an. Sie müssen eine AWS Glue Tabelle angeben, die das Schema enthält, das Firehose für die Konvertierung Ihres Datensatzformats verwenden soll. Weitere Informationen finden Sie unter Konvertiert das Eingabedatenformat.

Ein Beispiel dafür, wie Sie die Konvertierung von Datensatzformaten einrichten CloudFormation, finden Sie unter AWS::KinesisFirehose: DeliveryStream.

Wenn Sie Amazon Kinesis Data Streams oder Direct PUT als Quelle für Ihren Firehose-Stream wählen

Geben Sie im Abschnitt Quelleinstellungen die folgenden Felder ein.

1. Wählen Sie unter Datensätze transformieren eine der folgenden Optionen aus:

   a. Wenn Ihr Ziel Amazon S3 oder Splunk ist, wählen Sie im Abschnitt Amazon CloudWatch Logs für Quelldatensätze dekomprimieren die Option Dekomprimierung aktivieren aus.

   b. Geben Sie im Abschnitt Quelldatensätze mit AWS Lambda transformieren Werte für das folgende Feld an:

      Datentransformation

      Um einen Firehose-Stream zu erstellen, der eingehende Daten nicht transformiert, aktivieren Sie nicht das Kontrollkästchen Datentransformation aktivieren.

      Um eine Lambda-Funktion anzugeben, die Amazon Data Firehose aufrufen und verwenden soll, um eingehende Daten vor der Übermittlung zu transformieren, aktivieren Sie das Kontrollkästchen Datentransformation aktivieren. Sie können eine neue Lambda-Funktion mit einem der Lambda-Vorlage konfigurieren oder eine vorhandene Lambda-Funktion auswählen. Ihre Lambda-Funktion muss das von Amazon Data Firehose geforderte Statusmodell enthalten. Weitere Informationen finden Sie unter Transformieren Sie Quelldaten in Amazon Data Firehose.

2. Machen Sie im Bereich Convert record format (Datensatzformat konvertieren) Angaben im folgenden Feld:

   Konvertierung des Datensatzformats

   Um einen Firehose-Stream zu erstellen, der das Format der eingehenden Datensätze nicht konvertiert, wählen Sie Disabled.

   Um das Format der eingehenden Datensätze zu konvertieren, wählen Sie Enabled (Aktiviert). Geben Sie dann das gewünschte Ausgabeformat an. Sie müssen eine AWS Glue Tabelle

angeben, die das Schema enthält, das Amazon Data Firehose für die Konvertierung Ihres Datensatzformats verwenden soll. Weitere Informationen finden Sie unter Konvertiert das Eingabedatenformat.

Ein Beispiel dafür, wie Sie die Konvertierung von Datensatzformaten einrichten CloudFormation, finden Sie unter AWS::KinesisFirehose: DeliveryStream.

Zieleinstellungen konfigurieren

In diesem Abschnitt werden die Einstellungen beschrieben, die Sie für Ihren Firehose-Stream basierend auf dem von Ihnen ausgewählten Ziel konfigurieren müssen.

Themen

Zieleinstellungen für Amazon S3 konfigurieren

Sie müssen die folgenden Einstellungen angeben, um Amazon S3 als Ziel für Ihren Firehose-Stream zu verwenden.

- Geben Sie Werte für folgende Felder ein.

  S3 bucket

  Wählen Sie einen S3-Bucket, den Sie besitzen und an den die Streaming-Daten geliefert werden sollen. Sie können einen neuen S3-Bucket erstellen oder einen vorhandenen auswählen.

  Neues Zeilentrennzeichen

  Sie können Ihren Firehose-Stream so konfigurieren, dass ein neues Zeilentrennzeichen zwischen Datensätzen in Objekten hinzugefügt wird, die an Amazon S3 geliefert werden. Wählen Sie dazu Aktiviert. Um kein neues Zeilentrennzeichen zwischen Datensätzen in Objekten hinzuzufügen, die an Amazon S3 geliefert werden, wählen Sie Deaktiviert. Wenn Sie Athena verwenden möchten, um S3-Objekte mit aggregierten Datensätzen abzufragen, aktivieren Sie diese Option.

  Dynamische Partitionierung

  Wählen Sie Aktiviert, um die dynamische Partitionierung zu aktivieren und zu konfigurieren.

  Deaggregation mehrerer Datensätze

  Dabei werden die Datensätze im Firehose-Stream analysiert und entweder anhand eines gültigen JSON-Codes oder anhand des angegebenen neuen Zeilentrennzeichens getrennt.

  Wenn Sie mehrere Ereignisse, Protokolle oder Datensätze zu einem einzigen PutRecord PutRecordBatch API-Aufruf zusammenfassen, können Sie dennoch die dynamische Partitionierung aktivieren und konfigurieren. Wenn Sie bei aggregierten Daten die dynamische Partitionierung aktivieren, analysiert Amazon Data Firehose die Datensätze und sucht innerhalb jedes API-Aufrufs nach mehreren gültigen JSON-Objekten. Wenn der Firehose-Stream mit Kinesis Data Stream als Quelle konfiguriert ist, können Sie auch die integrierte Aggregation in der Kinesis Producer Library (KPL) verwenden. Die Datenpartitionsfunktion wird ausgeführt, nachdem die Daten deaggregiert wurden. Daher kann jeder Datensatz in jedem API-Aufruf an unterschiedliche Amazon-S3-Präfixe übermittelt werden. Sie können die Lambda-Funktionsintegration auch nutzen, um jede andere

Deaggregation oder jede andere Transformation vor der Datenpartitionierungsfunktion durchzuführen.

> **Important**
>
> Wenn Ihre Daten aggregiert sind, kann die dynamische Partitionierung erst nach der Deaggregation der Daten angewendet werden. Wenn Sie also die dynamische Partitionierung für Ihre aggregierten Daten aktivieren, müssen Sie Aktiviert auswählen, um die Deaggregation mehrerer Datensätze zu aktivieren.

Firehose Stream führt die folgenden Verarbeitungsschritte in der folgenden Reihenfolge durch: KPL-Deaggregation (Protobuf), JSON- oder Delimiter-Deaggregation, Lambda-Verarbeitung, Datenpartitionierung, Datenformatkonvertierung und Amazon S3 S3-Lieferung.

Deaggregationstyp für mehrere Datensätze

Wenn Sie die Deaggregation mehrerer Datensätze aktiviert haben, müssen Sie die Methode angeben, mit der Firehose Ihre Daten deaggregieren soll. Verwenden Sie das Dropdown-Menü, um entweder JSON oder Delimited auszuwählen.

Inline-Parsing

Dies ist einer der unterstützten Mechanismen zur dynamischen Partitionierung Ihrer Daten, die für Amazon S3 bestimmt sind. Um Inline-Parsing als dynamische Partitionierungsmethode für Ihre Streaming-Daten zu verwenden, müssen Sie Datensatzparameter auswählen, die als Partitionierungsschlüssel verwendet werden sollen, und für jeden angegebenen Partitionierungsschlüssel einen Wert angeben. Wählen Sie Aktiviert, um die Inline-Parsing zu aktivieren und zu konfigurieren.

> **Important**
>
> Wenn Sie in den obigen Schritten eine AWS Lambda-Funktion für die Transformation Ihrer Quelldatensätze angegeben haben, können Sie diese Funktion verwenden, um Ihre an S3 gebundenen Daten dynamisch zu partitionieren, und Sie können Ihre Partitionierungsschlüssel trotzdem mit Inline-Parsing erstellen. Bei der dynamischen Partitionierung können Sie entweder Inline-Parsing oder Ihre AWS Lambda-Funktion verwenden, um Ihre Partitionierungsschlüssel zu erstellen. Oder Sie können sowohl

Inline-Parsing als auch Ihre AWS Lambda-Funktion gleichzeitig verwenden, um Ihre Partitionierungsschlüssel zu erstellen.

Dynamische Partitionierung-Schlüssel

Sie können die Felder Schlüssel und Wert verwenden, um die Datensatzparameter anzugeben, die als dynamische Partitionierungsschlüssel verwendet werden sollen, und JQ-Abfragen, um dynamische Partitionierungsschlüsselwerte zu generieren. Firehose unterstützt nur jq 1.6. Sie können bis zu 50 dynamische Partitionierungsschlüssel angeben. Sie müssen gültige JQ-Ausdrücke für Ihre dynamischen Partitionierungsschlüsselwerte eingeben, um die dynamische Partitionierung für Ihren Firehose-Stream erfolgreich zu konfigurieren.

S3-Bucket-Präfix

Wenn Sie die dynamische Partitionierung aktivieren und konfigurieren, müssen Sie die S3-Bucket-Präfixe angeben, an die Amazon Data Firehose partitionierte Daten liefern soll.

Damit die dynamische Partitionierung korrekt konfiguriert werden kann, muss die Anzahl der S3-Bucket-Präfixe mit der Anzahl der angegebenen Partitionierungsschlüssel identisch sein.

Sie können Ihre Quelldaten mit Inline-Parsing oder mit Ihrer angegebenen AWS Lambda-Funktion partitionieren. Wenn Sie eine AWS Lambda-Funktion zum Erstellen von Partitionierungsschlüsseln für Ihre Quelldaten angegeben haben, müssen Sie die S3-Bucket-Präfixwerte manuell im folgenden Format eingeben: "partitionKeyFromLambda:KeyID“. Wenn Sie Inline-Parsing verwenden, um die Partitionierungsschlüssel für Ihre Quelldaten anzugeben, können Sie die S3-Bucket-Vorschauwerte entweder manuell im folgenden Format eingeben: "partitionKeyFromquery:keyID“ oder Sie können die Schaltfläche Dynamische Partitionierungsschlüssel anwenden wählen, um Ihre dynamischen Partitionierungspaare zur automatischen Generierung Ihrer S3-Bucket-Präfixe zu verwenden. key/value Bei der Partitionierung Ihrer Daten mit Inline-Parsing oder AWS Lambda können Sie auch die folgenden Ausdrucksformen in Ihrem S3-Bucket-Präfix verwenden:! {namespace:value}, wobei der Namespace entweder Query oder Lambda sein kann. partitionKeyFrom partitionKeyFrom

Zeitzone für das Ausgabepräfix für S3-Bucket und S3-Fehler

Wählen Sie eine Zeitzone, die Sie für Datum und Uhrzeit in benutzerdefinierten Präfixen für Amazon S3 S3-Objekte verwenden möchten. Standardmäßig fügt Firehose ein Zeitpräfix

in UTC hinzu. Sie können die in S3-Präfixen verwendete Zeitzone ändern, wenn Sie eine andere Zeitzone verwenden möchten.

Hinweise zum Puffern

Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

S3-Komprimierung

Wählen Sie GZIP-, Snappy-, Zip- oder Hadoop-kompatible Snappy-Datenkomprimierung oder keine Datenkomprimierung. Snappy-, Zip- und Hadoop-kompatible Snappy-Komprimierung ist für Firehose-Streams mit Amazon Redshift als Ziel nicht verfügbar.

S3-Dateierweiterungsformat (optional)

Geben Sie ein Dateierweiterungsformat für Objekte an, die an den Amazon S3 S3-Ziel-Bucket geliefert werden. Wenn Sie diese Funktion aktivieren, überschreibt die angegebene Dateierweiterung die Standarddateierweiterungen, die von Datenformatkonvertierungs- oder S3-Komprimierungsfunktionen wie .parquet oder .gz hinzugefügt wurden. Vergewissern Sie sich, dass Sie die richtige Dateierweiterung konfiguriert haben, wenn Sie diese Funktion mit Datenformatkonvertierung oder S3-Komprimierung verwenden. Die Dateierweiterung muss mit einem Punkt (.) beginnen und kann die zulässigen Zeichen enthalten: 0-9a-z! -_.*' (). Die Dateierweiterung darf 128 Zeichen nicht überschreiten.

S3-Verschlüsselung

Firehose unterstützt die serverseitige Amazon S3-Verschlüsselung mit AWS Key Management Service (SSE-KMS) zur Verschlüsselung von gelieferten Daten in Amazon S3. Sie können wählen, ob Sie den im Ziel-S3-Bucket angegebenen Standardverschlüsselungstyp verwenden oder mit einem Schlüssel aus der Liste der Schlüssel verschlüsseln möchten, die Sie besitzen. AWS KMS Wenn Sie die Daten mit AWS KMS Schlüsseln verschlüsseln, können Sie entweder den AWS verwalteten Standardschlüssel (aws/s3) oder einen vom Kunden verwalteten Schlüssel verwenden. Weitere Informationen finden Sie unter Schutz von Daten mithilfe serverseitiger Verschlüsselung mit AWS KMS-verwalteten Schlüsseln (SSE-KMS).

Konfigurieren Sie die Zieleinstellungen für Apache Iceberg-Tabellen

Firehose unterstützt Apache Iceberg Tables als Ziel in allen Regionen AWS-Regionenaußer China und im asiatisch-pazifischen Raum (Malaysia). AWS GovCloud (US) Regions

Weitere Informationen zu Apache Iceberg Tables als Ziel finden Sie unter. Liefern Sie Daten mit Amazon Data Firehose an Apache Iceberg Tables

Zieleinstellungen für Amazon Redshift konfigurieren

In diesem Abschnitt werden Einstellungen für die Verwendung von Amazon Redshift als Firehose-Stream-Ziel beschrieben.

Wählen Sie eines der folgenden Verfahren, je nachdem, ob Sie über einen von Amazon Redshift bereitgestellten Cluster oder eine Arbeitsgruppe von Amazon Redshift Serverless verfügen.

- Von Amazon Redshift bereitgestellte Cluster
- Zieleinstellungen für Amazon Redshift Serverless Workgroup konfigurieren

> **Note**
>
> Firehose kann nicht in Amazon Redshift Redshift-Cluster schreiben, die erweitertes VPC-Routing verwenden.

Von Amazon Redshift bereitgestellte Cluster

In diesem Abschnitt werden die Einstellungen für die Verwendung des von Amazon Redshift bereitgestellten Clusters als Firehose-Stream-Ziel beschrieben.

- Geben Sie Werte für folgende Felder ein:

  Cluster

  Das Amazon-Redshift-Cluster, in das S3-Bucket-Daten kopiert werden. Konfigurieren Sie den Amazon Redshift Redshift-Cluster so, dass er öffentlich zugänglich ist, und entsperren Sie die IP-Adressen von Amazon Data Firehose. Weitere Informationen finden Sie unter Firehose Zugriff auf ein Amazon Redshift Redshift-Ziel gewähren .

  Authentifizierung

  Sie können wählen, ob Sie das username/password direkt eingeben oder das Geheimnis abrufen möchten, AWS Secrets Manager um auf den Amazon Redshift Redshift-Cluster zuzugreifen.

  - Benutzername

Geben Sie einen Amazon Redshift Redshift-Benutzer mit Zugriffsberechtigungen für den Amazon Redshift Redshift-Cluster an. Dieser Benutzer muss über die `INSERT`-Berechtigung von Amazon Redshift für das Kopieren von Daten aus dem S3-Bucket in den Amazon-Redshift-Cluster verfügen.

- Passwort

  Geben Sie das Passwort für den Benutzer an, der über Zugriffsberechtigungen für den Cluster verfügt.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das die Anmeldeinformationen für den Amazon Redshift Redshift-Cluster enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines AWS Secrets Manager für Ihre Amazon Redshift Redshift-Anmeldeinformationen. Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Datenbank

Die Amazon-Redshift-Datenbank, in die die Daten kopiert werden.

Tabelle

Die Amazon-Redshift-Tabelle, in die die Daten kopiert werden.

Spalten

(Optional) Die spezifischen Spalten der Tabelle, zu der die Daten kopiert werden. Verwenden Sie diese Option, wenn die Anzahl der in Ihren Amazon-S3-Objekten definierten Spalten kleiner als die Anzahl der Spalten in der Amazon-Redshift-Tabelle ist.

Intermediäres S3-Zwischenziel

Firehose liefert Ihre Daten zuerst an Ihren S3-Bucket und gibt dann einen Amazon Redshift COPY Redshift-Befehl aus, um die Daten in Ihren Amazon Redshift Redshift-Cluster zu laden. Geben Sie einen S3-Bucket an, den Sie besitzen und an den die Streaming-Daten geliefert werden sollen. Erstellen Sie einen neuen S3-Bucket oder wählen Sie einen vorhandenen Bucket aus, den Sie besitzen.

Firehose löscht die Daten nicht aus Ihrem S3-Bucket, nachdem sie in Ihren Amazon Redshift Redshift-Cluster geladen wurden. Sie können die Daten in Ihrem S3-Bucket mithilfe einer

Lebenszykluskonfiguration verwalten. Weitere Informationen finden Sie unter Objekt-Lebenszyklusmanagement im Benutzerhandbuch zum Amazon Simple Storage Service.

Intermediate S3-Präfix

(Optional) Lassen Sie die Option leer, wenn Sie das Standardpräfix für Amazon-S3-Objekte verwenden möchten. Firehose verwendet automatisch ein Präfix im `YYYY/MM/dd/HH` UTC-Zeitformat für gelieferte Amazon S3 S3-Objekte. Sie können am Anfang dieses Präfix Elemente hinzufügen. Weitere Informationen finden Sie unter Amazon S3 S3-Objektnamenformat konfigurieren.

COPY-Optionen

Parameter, die Sie im COPY-Befehl von Amazon Redshift angeben können. Diese sind möglicherweise für Ihre Konfiguration erforderlich. Beispielsweise ist "`GZIP`" erforderlich, wenn die Amazon S3 S3-Datenkomprimierung aktiviert ist. „`REGION`" ist erforderlich, wenn sich Ihr S3-Bucket nicht in derselben AWS Region wie Ihr Amazon Redshift Redshift-Cluster befindet. Weitere Informationen finden Sie unter COPY im Entwicklerhandbuch für Amazon Redshift Database.

Befehl COPY

Der COPY-Befehl von Amazon Redshift. Weitere Informationen finden Sie unter COPY im Entwicklerhandbuch für Amazon Redshift Database.

Retry duration

Zeitdauer (0—7200 Sekunden), bis Firehose es erneut versucht, falls Daten in Ihrem Amazon Redshift COPY Redshift-Cluster ausfallen. Firehose versucht es alle 5 Minuten, bis die Wiederholungsdauer abgelaufen ist. Wenn Sie die Wiederholungsdauer auf 0 (Null) Sekunden setzen, versucht Firehose bei einem COPY fehlgeschlagenen Befehl nicht erneut.

Hinweise zum Puffern

Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

S3-Komprimierung

Wählen Sie GZIP-, Snappy-, Zip- oder Hadoop-kompatible Snappy-Datenkomprimierung oder keine Datenkomprimierung. Snappy-, Zip- und Hadoop-kompatible Snappy-Komprimierung ist für Firehose-Streams mit Amazon Redshift als Ziel nicht verfügbar.

S3-Dateierweiterungsformat (optional)

S3-Dateierweiterungsformat (optional) — Geben Sie ein Dateierweiterungsformat für Objekte an, die an den Amazon S3 S3-Ziel-Bucket geliefert werden. Wenn Sie diese Funktion aktivieren, überschreibt die angegebene Dateierweiterung die Standarddateierweiterungen, die durch Datenformatkonvertierungs- oder S3-Komprimierungsfunktionen wie .parquet oder .gz hinzugefügt wurden. Vergewissern Sie sich, dass Sie die richtige Dateierweiterung konfiguriert haben, wenn Sie diese Funktion mit Datenformatkonvertierung oder S3-Komprimierung verwenden. Die Dateierweiterung muss mit einem Punkt (.) beginnen und kann die zulässigen Zeichen enthalten: 0-9a-z! -_.*' (). Die Dateierweiterung darf 128 Zeichen nicht überschreiten.

S3-Verschlüsselung

Firehose unterstützt die serverseitige Amazon S3-Verschlüsselung mit AWS Key Management Service (SSE-KMS) zur Verschlüsselung von gelieferten Daten in Amazon S3. Sie können wählen, ob Sie den im Ziel-S3-Bucket angegebenen Standardverschlüsselungstyp verwenden oder mit einem Schlüssel aus der Liste der Schlüssel verschlüsseln möchten, die Sie besitzen. AWS KMS Wenn Sie die Daten mit AWS KMS Schlüsseln verschlüsseln, können Sie entweder den AWS verwalteten Standardschlüssel (aws/s3) oder einen vom Kunden verwalteten Schlüssel verwenden. Weitere Informationen finden Sie unter Schutz von Daten mithilfe serverseitiger Verschlüsselung mit AWS KMS-verwalteten Schlüsseln (SSE-KMS).

Zieleinstellungen für Amazon Redshift Serverless Workgroup konfigurieren

In diesem Abschnitt werden Einstellungen für die Verwendung der Amazon Redshift Serverless Workgroup als Firehose-Stream-Ziel beschrieben.

- Geben Sie Werte für folgende Felder ein:

  Workgroup name (Name der Arbeitsgruppe)

  Die Arbeitsgruppe von Amazon Redshift Serverless, in die S3-Bucket-Daten kopiert werden. Konfigurieren Sie die Amazon Redshift Serverless-Arbeitsgruppe so, dass sie öffentlich zugänglich ist, und entsperren Sie die Firehose-IP-Adressen. Weitere Informationen finden Sie im Abschnitt Herstellen einer öffentlich zugänglichen Instance von Amazon Redshift

Serverless unter Verbindung mit Amazon Redshift Serverless herstellen und auch Firehose Zugriff auf ein Amazon Redshift Redshift-Ziel gewähren .

Authentifizierung

Sie können wählen, ob Sie das username/password direkt eingeben oder das Geheimnis von abrufen möchten AWS Secrets Manager , um auf die Amazon Redshift Serverless Workgroup zuzugreifen.

- Benutzername

  Geben Sie einen Amazon Redshift-Benutzer mit Zugriffsberechtigungen für die Amazon Redshift Serverless-Arbeitsgruppe an. Dieser Benutzer muss über die `INSERT`-Berechtigung von Amazon Redshift für das Kopieren von Daten aus dem S3-Bucket in die Arbeitsgruppe von Amazon Redshift Serverless verfügen.

- Passwort

  Geben Sie das Passwort für den Benutzer an, der über Zugriffsberechtigungen für die Amazon Redshift Serverless-Arbeitsgruppe verfügt.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das die Anmeldeinformationen für die Amazon Redshift Serverless Workgroup enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines AWS Secrets Manager für Ihre Amazon Redshift Redshift-Anmeldeinformationen. Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Datenbank

Die Amazon-Redshift-Datenbank, in die die Daten kopiert werden.

Tabelle

Die Amazon-Redshift-Tabelle, in die die Daten kopiert werden.

Spalten

(Optional) Die spezifischen Spalten der Tabelle, zu der die Daten kopiert werden. Verwenden Sie diese Option, wenn die Anzahl der in Ihren Amazon-S3-Objekten definierten Spalten kleiner als die Anzahl der Spalten in der Amazon-Redshift-Tabelle ist.

Intermediäres S3-Zwischenziel

Amazon Data Firehose übermittelt Ihre Daten zuerst an Ihren S3-Bucket und gibt dann einen Amazon COPY Redshift-Befehl aus, um die Daten in Ihre Amazon Redshift Serverless-Arbeitsgruppe zu laden. Geben Sie einen S3-Bucket an, den Sie besitzen und an den die Streaming-Daten geliefert werden sollen. Erstellen Sie einen neuen S3-Bucket oder wählen Sie einen vorhandenen Bucket aus, den Sie besitzen.

Firehose löscht die Daten nicht aus Ihrem S3-Bucket, nachdem sie in Ihre Amazon Redshift Serverless-Arbeitsgruppe geladen wurden. Sie können die Daten in Ihrem S3-Bucket mithilfe einer Lebenszykluskonfiguration verwalten. Weitere Informationen finden Sie unter Objekt-Lebenszyklusmanagement im Benutzerhandbuch zum Amazon Simple Storage Service.

Intermediate S3-Präfix

(Optional) Lassen Sie die Option leer, wenn Sie das Standardpräfix für Amazon-S3-Objekte verwenden möchten. Firehose verwendet automatisch ein Präfix im `YYYY/MM/dd/HH` UTC-Zeitformat für gelieferte Amazon S3 S3-Objekte. Sie können am Anfang dieses Präfix Elemente hinzufügen. Weitere Informationen finden Sie unter Amazon S3 S3-Objektnamenformat konfigurieren.

COPY-Optionen

Parameter, die Sie im COPY-Befehl von Amazon Redshift angeben können. Diese sind möglicherweise für Ihre Konfiguration erforderlich. Beispielsweise ist "`GZIP`" erforderlich, wenn die Amazon S3 S3-Datenkomprimierung aktiviert ist. „`REGION`" ist erforderlich, wenn sich Ihr S3-Bucket nicht in derselben AWS Region wie Ihre Amazon Redshift Serverless-Arbeitsgruppe befindet. Weitere Informationen finden Sie unter COPY im Entwicklerhandbuch für Amazon Redshift Database.

Befehl COPY

Der COPY-Befehl von Amazon Redshift. Weitere Informationen finden Sie unter COPY im Entwicklerhandbuch für Amazon Redshift Database.

Retry duration

Zeitdauer (0—7200 Sekunden), bis Firehose es erneut versucht, falls Daten COPY an Ihre Amazon Redshift Serverless-Arbeitsgruppe ausfallen. Firehose versucht es alle 5 Minuten, bis die Wiederholungsdauer abgelaufen ist. Wenn Sie die Wiederholungsdauer auf 0 (Null) Sekunden setzen, versucht Firehose bei einem COPY fehlgeschlagenen Befehl nicht erneut.

Hinweise zum Puffern

Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

S3-Komprimierung

Wählen Sie GZIP-, Snappy-, Zip- oder Hadoop-kompatible Snappy-Datenkomprimierung oder keine Datenkomprimierung. Snappy-, Zip- und Hadoop-kompatible Snappy-Komprimierung ist für Firehose-Streams mit Amazon Redshift als Ziel nicht verfügbar.

S3-Dateierweiterungsformat (optional)

S3-Dateierweiterungsformat (optional) — Geben Sie ein Dateierweiterungsformat für Objekte an, die an den Amazon S3 S3-Ziel-Bucket geliefert werden. Wenn Sie diese Funktion aktivieren, überschreibt die angegebene Dateierweiterung die Standarddateierweiterungen, die durch Datenformatkonvertierungs- oder S3-Komprimierungsfunktionen wie .parquet oder .gz hinzugefügt wurden. Vergewissern Sie sich, dass Sie die richtige Dateierweiterung konfiguriert haben, wenn Sie diese Funktion mit Datenformatkonvertierung oder S3-Komprimierung verwenden. Die Dateierweiterung muss mit einem Punkt (.) beginnen und kann die zulässigen Zeichen enthalten: 0-9a-z! -_.*' (). Die Dateierweiterung darf 128 Zeichen nicht überschreiten.

S3-Verschlüsselung

Firehose unterstützt die serverseitige Amazon S3-Verschlüsselung mit AWS Key Management Service (SSE-KMS) zur Verschlüsselung von gelieferten Daten in Amazon S3. Sie können wählen, ob Sie den im Ziel-S3-Bucket angegebenen Standardverschlüsselungstyp verwenden oder mit einem Schlüssel aus der Liste der Schlüssel verschlüsseln möchten, die Sie besitzen. AWS KMS Wenn Sie die Daten mit AWS KMS Schlüsseln verschlüsseln, können Sie entweder den AWS verwalteten Standardschlüssel (aws/s3) oder einen vom Kunden verwalteten Schlüssel verwenden. Weitere Informationen finden Sie unter Schutz von Daten mithilfe serverseitiger Verschlüsselung mit AWS KMS-verwalteten Schlüsseln (SSE-KMS).

Konfigurieren Sie die Zieleinstellungen für den Dienst OpenSearch

Firehose unterstützt Elasticsearch-Versionen — 1.5, 2.3, 5.1, 5.3, 5.5, 5.6 sowie alle 6.*-, 7.*- und 8.*-Versionen. Firehose unterstützt Amazon OpenSearch Service 2.x bis 2.11.

In diesem Abschnitt werden Optionen für die Nutzung von OpenSearch Service für Ihr Ziel beschrieben.

- Geben Sie Werte für folgende Felder ein:

  OpenSearch Dienstdomäne

  Die OpenSearch Dienstdomäne, an die Ihre Daten übermittelt werden.

  Index

  Der OpenSearch Service-Indexname, der bei der Indizierung von Daten in Ihrem OpenSearch Service-Cluster verwendet werden soll.

  Index rotation

  Wählen Sie aus, ob und wie oft der OpenSearch Serviceindex rotiert werden soll. Wenn die Indexrotation aktiviert ist, hängt Amazon Data Firehose den entsprechenden Zeitstempel an den angegebenen Indexnamen an und rotiert. Weitere Informationen finden Sie unter Konfigurieren Sie die Indexrotation für Service OpenSearch .

  Typ

  Der Name des OpenSearch Servicetyps, der bei der Indizierung von Daten für Ihren Service-Cluster verwendet werden soll. OpenSearch Für Elasticsearch 7.x und OpenSearch 1.x kann es nur einen Typ pro Index geben. Wenn Sie versuchen, einen neuen Typ für einen vorhandenen Index anzugeben, der bereits einen anderen Typ hat, gibt Firehose während der Laufzeit einen Fehler zurück.

  Lassen Sie dieses Feld für Elasticsearch 7.x leer.

  Retry duration

  Zeitdauer, die Firehose benötigt, um es erneut zu versuchen, falls eine Indexanforderung fehlschlägt. OpenSearch Für die Dauer des Wiederholungsversuchs können Sie einen beliebigen Wert zwischen 0 und 7200 Sekunden festlegen. Die Standarddauer für den Wiederholungsversuch beträgt 300 Sekunden. Firehose versucht es mehrmals mit exponentiellem Back Off, bis die Wiederholungsdauer abgelaufen ist.

  Nach Ablauf der Wiederholungsdauer übermittelt Firehose die Daten an die Dead Letter Queue (DLQ), einen konfigurierten S3-Fehler-Bucket. Für Daten, die an DLQ geliefert

werden, müssen Sie die Daten erneut vom konfigurierten S3-Fehler-Bucket zum Ziel zurückleiten. OpenSearch

Wenn Sie verhindern möchten, dass der Firehose-Stream aufgrund von Ausfallzeiten oder Wartungsarbeiten an OpenSearch Clustern Daten an DLQ übermittelt, können Sie die Wiederholungsdauer auf einen höheren Wert in Sekunden konfigurieren. Sie können den Wert für die Wiederholungsdauer auf einen Wert von über 7200 Sekunden erhöhen, indem Sie sich an den Support wenden.AWS

DocumentID-Typ

Gibt die Methode zum Einrichten der Dokument-ID an. Die unterstützten Methoden sind Firehose-generierte Dokument-ID und OpenSearch Service-generierte Dokument-ID. Die von Firehose generierte Dokument-ID ist die Standardoption, wenn der Dokument-ID-Wert nicht festgelegt ist. OpenSearch Die vom Dienst generierte Dokument-ID ist die empfohlene Option, da sie schreibintensive Operationen wie Protokollanalysen und Beobachtbarkeit unterstützt, wodurch weniger CPU-Ressourcen in der OpenSearch Dienstdomäne verbraucht werden und somit die Leistung verbessert wird.

Destination VPC connectivity (Ziel-VPC-Konnektivität

Wenn sich Ihre OpenSearch Service-Domain in einer privaten VPC befindet, verwenden Sie diesen Abschnitt, um diese VPC anzugeben. Geben Sie auch die Subnetze und Untergruppen an, die Amazon Data Firehose verwenden soll, wenn es Daten an Ihre OpenSearch Service-Domain sendet. Sie können dieselben Sicherheitsgruppen verwenden, die die OpenSearch Service-Domain verwendet. Wenn Sie verschiedene Sicherheitsgruppen angeben, stellen Sie sicher, dass diese ausgehenden HTTPS-Verkehr zur Sicherheitsgruppe der OpenSearch Dienstdomäne zulassen. Stellen Sie außerdem sicher, dass die Sicherheitsgruppe der OpenSearch Service-Domain HTTPS-Verkehr von den Sicherheitsgruppen zulässt, die Sie bei der Konfiguration Ihres Firehose-Streams angegeben haben. Wenn Sie dieselbe Sicherheitsgruppe sowohl für Ihren Firehose-Stream als auch für die OpenSearch Service-Domain verwenden, stellen Sie sicher, dass die eingehende Regel der Sicherheitsgruppe HTTPS-Verkehr zulässt. Weitere Informationen zu den Regeln der Sicherheitsgruppe finden Sie unter Sicherheitsgruppenregeln im Amazon-VPC-Benutzerhandbuch.

> **Important**
>
> Wenn Sie Subnetze für die Übertragung von Daten an das Ziel in einer privaten VPC angeben, stellen Sie sicher, dass Sie über genügend freie IP-Adressen in den ausgewählten Subnetzen verfügen. Wenn in einem bestimmten Subnetz keine kostenlose IP-Adresse verfügbar ist, kann Firehose die Datenlieferung in der privaten VPC nicht erstellen oder hinzufügen ENIs , und die Lieferung wird beeinträchtigt oder schlägt fehl.

Puffer-Hinweise

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für Serverless OpenSearch

In diesem Abschnitt werden Optionen für die Verwendung von OpenSearch Serverless für Ihr Ziel beschrieben.

- Geben Sie Werte für folgende Felder ein:

  OpenSearch Serverlose Erfassung

  Der Endpunkt für eine Gruppe von OpenSearch serverlosen Indizes, an die Ihre Daten übermittelt werden.

  Index

  Der OpenSearch Serverless-Indexname, der bei der Indizierung von Daten für Ihre Serverless-Sammlung verwendet werden soll. OpenSearch

  Destination VPC connectivity (Ziel-VPC-Konnektivität

  Wenn sich Ihre OpenSearch Serverless-Sammlung in einer privaten VPC befindet, verwenden Sie diesen Abschnitt, um diese VPC anzugeben. Geben Sie auch die Subnetze und Untergruppen an, die Amazon Data Firehose verwenden soll, wenn es Daten an Ihre OpenSearch Serverless-Sammlung sendet.

> **Important**
>
> Wenn Sie Subnetze für die Übertragung von Daten an das Ziel in einer privaten VPC angeben, stellen Sie sicher, dass Sie über genügend freie IP-Adressen in den ausgewählten Subnetzen verfügen. Wenn in einem bestimmten Subnetz keine kostenlose IP-Adresse verfügbar ist, kann Firehose die Datenlieferung in der privaten VPC nicht erstellen oder hinzufügen ENIs , und die Lieferung wird beeinträchtigt oder schlägt fehl.

Retry duration

Zeitdauer, bis Firehose es erneut versucht, falls eine Indexanforderung an OpenSearch Serverless fehlschlägt. Für die Dauer des Wiederholungsversuchs können Sie einen beliebigen Wert zwischen 0 und 7200 Sekunden festlegen. Die Standarddauer für den Wiederholungsversuch beträgt 300 Sekunden. Firehose versucht es mehrmals mit exponentiellem Back Off, bis die Wiederholungsdauer abgelaufen ist.

Nach Ablauf der Wiederholungsdauer übermittelt Firehose die Daten an die Dead Letter Queue (DLQ), einen konfigurierten S3-Fehler-Bucket. Für Daten, die an DLQ geliefert werden, müssen Sie die Daten erneut vom konfigurierten S3-Fehler-Bucket zum serverlosen Ziel zurückleiten. OpenSearch

Wenn Sie verhindern möchten, dass der Firehose-Stream aufgrund von Ausfallzeiten oder Wartungsarbeiten an OpenSearch serverlosen Clustern Daten an DLQ übermittelt, können Sie die Wiederholungsdauer auf einen höheren Wert in Sekunden konfigurieren. Sie können den Wert für die Dauer der Wiederholungsversuche auf über 7200 Sekunden erhöhen, indem Sie sich an den Support wenden.AWS

Puffer-Hinweise

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für den HTTP-Endpunkt

In diesem Abschnitt werden Optionen für die Verwendung von HTTP Endpunkt als Ihr Ziel beschrieben.

> **Important**
>
> Wenn Sie einen HTTP-Endpunkt als Ziel wählen, lesen und befolgen Sie die Anweisungen unterVerstehen Sie die Anforderungen- und Antwortspezifikationen für die HTTP-Endpunktzustellung.

- Geben Sie Werte für folgende Felder an:

  Name des HTTP-Endpunkts – optional

  Geben Sie einen benutzerfreundlichen Namen für den HTTP-Endpunkt an. Beispiel, `My HTTP Endpoint Destination`.

  URL des HTTP-Endpunkts

  Geben Sie die URL für den HTTP-Endpunkt im folgenden Format an: `https://xyz.httpendpoint.com`. Die URL muss eine HTTPS-URL sein.

  Authentifizierung

  Sie können entweder den Zugriffsschlüssel direkt eingeben oder das Geheimnis abrufen, AWS Secrets Manager um auf den HTTP-Endpunkt zuzugreifen.

  - (Optional) Zugriffsschlüssel

    Wenden Sie sich an den Endpunktbesitzer, wenn Sie den Zugriffsschlüssel benötigen, um die Datenlieferung an seinen Endpunkt von Firehose zu ermöglichen.

  - Secret

    Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den Zugriffsschlüssel für den HTTP-Endpunkt enthält. Wenn Sie Ihr Geheimnis nicht in der Dropdownliste sehen, erstellen Sie eines AWS Secrets Manager für den Zugriffsschlüssel. Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP oder Deaktiviert, um den enable/disable Inhalt Ihrer Anfrage zu kodieren.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den ausgewählten HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

> **Important**
>
> Wenn Sie für die HTTP-Endpunktziele 413 Antwortcodes vom Zielendpunkt in CloudWatch Logs sehen, verringern Sie die Größe der Pufferhinweise in Ihrem Firehose-Stream und versuchen Sie es erneut.

Konfigurieren Sie die Zieleinstellungen für Datadog

In diesem Abschnitt werden Optionen für die Verwendung von Datadog als Ziel beschrieben. Weitere Informationen zu Datadog finden Sie unter amazon_web_services/. https://docs.datadoghq.com/integrations/

- Geben Sie Werte für die folgenden Felder an.

  URL des HTTP-Endpunkts

  Wählen Sie aus einer der folgenden Optionen im Dropdownmenü aus, wohin Sie Daten senden möchten.

  - Datadog protokolliert - US1
  - Datadog-Protokolle - US3
  - Datadog-Protokolle - US5
  - Datadog-Protokolle - AP1
  - Datadog-Protokolle – US
  - Datadog-Protokolle – GOV
  - Datadog-Metriken – USA
  - Datadog-Metriken - US5
  - Datadog-Metriken - AP1
  - Datadog-Metriken – EU

- Datadog-Konfigurationen - US1
- Datadog-Konfigurationen - US3
- Datadog-Konfigurationen - US5
- Datadog-Konfigurationen - AP1
- Datadog-Konfigurationen - EU
- Datadog-Konfigurationen — US-Regierung

Authentifizierung

Sie können entweder den API-Schlüssel direkt eingeben oder den geheimen Schlüssel für den Zugriff auf Datadog abrufen. AWS Secrets Manager

- API-Schlüssel

  Wenden Sie sich an Datadog, um den API-Schlüssel zu erhalten, den Sie benötigen, um die Datenlieferung an diesen Endpunkt von Firehose zu aktivieren.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den API-Schlüssel für Datadog enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP oder Deaktiviert, um den enable/disable Inhalt Ihrer Anfrage zu kodieren.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den ausgewählten HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für Honeycomb

In diesem Abschnitt werden Optionen für die Verwendung von Honeycomb als Ziel beschrieben. Weitere Informationen zu Honeycomb finden Sie unter https://docs.honeycomb. -cloudwatch-metrics/. io/getting-data-in/metrics/aws

- Geben Sie Werte für folgende Felder an:

  Honeycomb-Kinesis-Endpunkt

  Geben Sie die URL für den HTTP-Endpunkt im folgenden Format an: b.io/1/kinesis_events/ {{dataset}} https://api.honeycom

Authentifizierung

Sie können entweder den API-Schlüssel direkt eingeben oder den geheimen Schlüssel für den Zugriff auf Honeycomb abrufen. AWS Secrets Manager

- API-Schlüssel

  Wenden Sie sich an Honeycomb, um den API-Schlüssel zu erhalten, den Sie benötigen, um die Datenlieferung an diesen Endpunkt von Firehose zu aktivieren.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den API-Schlüssel für Honeycomb enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in AWS Secrets Manager. Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP, um die Inhaltskodierung Ihrer Anfrage zu aktivieren. Dies ist die empfohlene Option für das Ziel Honeycomb.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den ausgewählten HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist.

Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für Coralogix

In diesem Abschnitt werden Optionen für die Verwendung von Coralogix als Ziel beschrieben. Weitere Informationen zu Coralogix finden Sie unter Erste Schritte mit Coralogix.

- Geben Sie Werte für folgende Felder an:

  URL des HTTP-Endpunkts

  Wählen Sie die HTTP-Endpunkt-URL aus den folgenden Optionen im Dropdown-Menü aus:

  - Coralogix – USA
  - Coralogix – SINGAPUR
  - Coralogix – IRLAND
  - Coralogix - INDIEN
  - Coralogix - STOCKHOLM

  Authentifizierung

  Sie können entweder den privaten Schlüssel direkt eingeben oder den geheimen Schlüssel für den Zugriff auf Coralogix abrufen. AWS Secrets Manager

- Privater Aktivierungsschlüssel

  Wenden Sie sich an Coralogix, um den privaten Schlüssel, den Sie für die Datenlieferung an diesen Endpunkt benötigen, von Firehose zu erhalten.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den privaten Schlüssel für Coralogix enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP, um die Inhaltskodierung Ihrer Anfrage zu aktivieren. Dies ist die empfohlene Option für das Coralogix-Ziel.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den ausgewählten HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

- applicationName: Die Umgebung, in der Sie Data Firehose ausführen
- subsystemName: Der Name der Data-Firehose-Integration
- ComputerName: Der Name des verwendeten Firehose-Streams

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel variiert je nach Dienstanbieter.

Konfigurieren Sie die Zieleinstellungen für Dynatrace

In diesem Abschnitt werden Optionen für die Verwendung von Dynatrace als Ziel beschrieben. Weitere Informationen finden Sie unter -metric-streams/ https://www.dynatrace.com/support/help/technology-support/cloud-platforms/amazon-web-services/integrations/cloudwatch.

- Wählen Sie Optionen, um Dynatrace als Ziel für Ihren Firehose-Stream zu verwenden.

  Art der Einnahme

  Wählen Sie aus, ob Sie Metriken oder Protokolle (Standard) zur weiteren Analyse und Verarbeitung in Dynatrace bereitstellen möchten.

  URL des HTTP-Endpunkts

  Wählen Sie die HTTP-Endpunkt-URL (Dynatrace US, Dynatrace EU oder Dynatrace Global) aus dem Drop-down-Menü aus.

  Authentifizierung

  Sie können entweder das API-Token direkt eingeben oder das Geheimnis für den Zugriff auf Dynatrace abrufen. AWS Secrets Manager

  - API-Token

Generieren Sie das Dynatrace-API-Token, das Sie benötigen, um die Datenlieferung von Firehose an diesen Endpunkt zu aktivieren. Weitere Informationen finden Sie unter Dynatrace API — Tokens und Authentifizierung.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das das API-Token für Dynatrace enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

API-URL

Geben Sie die API-URL Ihrer Dynatrace-Umgebung an.

Inhaltskodierung

Wählen Sie aus, ob Sie die Inhaltskodierung aktivieren möchten, um den Hauptteil der Anfrage zu komprimieren. Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wenn diese Option aktiviert ist, wird der Inhalt im GZIP-Format komprimiert.

Retry duration

Geben Sie an, wie lange Firehose erneut versucht, Daten an den ausgewählten HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eintrifft, startet Firehose den Zähler für die Dauer der Wiederholungsversuche. Es versucht es so lange, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Firehose es als Fehler bei der Datenzustellung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Firehose Daten an den HTTP-Endpunkt sendet, entweder beim ersten Versuch oder nach einem erneuten Versuch, startet es den Timeout-Zähler für die Bestätigung neu und wartet auf eine Bestätigung vom HTTP-Endpunkt.

Selbst wenn die Wiederholungsdauer abläuft, wartet Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Firehose, ob noch Zeit im

Wiederholungszähler übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die Puffer-Hinweise beinhalten die Puffergröße und das Intervall für Ihre Streams. Die empfohlene Puffergröße für das Ziel variiert je nach Dienstanbieter.

Konfigurieren Sie die Zieleinstellungen für LogicMonitor

In diesem Abschnitt werden Optionen für die Verwendung von LogicMonitor als Ziel beschrieben. Weitere Informationen finden Sie unter https://www.logicmonitor.com.

- Geben Sie Werte für folgende Felder an:

  URL des HTTP-Endpunkts

  Geben Sie die URL für den HTTP-Endpunkt im folgenden Format an.

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

  Authentifizierung

  Sie können entweder den API-Schlüssel direkt eingeben oder das Geheimnis abrufen, von dem aus Sie AWS Secrets Manager darauf zugreifen können LogicMonitor.

  - API-Schlüssel

    Wenden Sie sich LogicMonitor an Firehose, um den API-Schlüssel zu erhalten, den Sie benötigen, um die Datenlieferung an diesen Endpunkt zu aktivieren.

  - Secret

Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den API-Schlüssel für LogicMonitor enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in AWS Secrets Manager. Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP oder Deaktiviert, um den enable/disable Inhalt Ihrer Anfrage zu kodieren.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den ausgewählten HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für Logz.io

In diesem Abschnitt werden Optionen für die Verwendung von Logz.io als Ziel beschrieben. Weitere Informationen finden Sie unter https://logz.io/.

> **Note**
>
> In der Region Europa (Mailand) wird Logz.io nicht als Amazon Data Firehose-Ziel unterstützt.

- Geben Sie Werte für folgende Felder an:

  URL des HTTP-Endpunkts

  Geben Sie die URL für den HTTP-Endpunkt im folgenden Format an. Die URL muss eine HTTPS URL sein.

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

  Beispiel

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

  Authentifizierung

  Sie können entweder das Versand-Token direkt eingeben oder das Secret von abrufen, AWS Secrets Manager um auf Logz.io zuzugreifen.

  - Versand-Token

    Wenden Sie sich an Logz.io, um das Versand-Token zu erhalten, das Sie benötigen, um die Datenlieferung an diesen Endpunkt von Firehose zu aktivieren.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das das Versand-Token für Logz.io enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an Logz.io zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Zieleinstellungen für MongoDB Atlas konfigurieren

In diesem Abschnitt werden Optionen für die Verwendung von MongoDB Atlas für Ihr Ziel beschrieben. Weitere Informationen finden Sie unter MongoDB Atlas auf Amazon Web Services.

- Geben Sie Werte für folgende Felder an:

  URL des API Gateway

  Geben Sie die URL für den HTTP-Endpunkt im folgenden Format an.

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

  Die URL muss eine HTTPS URL sein.

  Authentifizierung

  Sie können entweder den API-Schlüssel direkt eingeben oder das Geheimnis abrufen, AWS Secrets Manager um auf MongoDB Atlas zuzugreifen.

  - API-Schlüssel

    Folgen Sie den Anweisungen in MongoDB Atlas on Amazon Web Services, um die Informationen zu erhaltenAPIKeyValue, die Sie benötigen, um die Datenlieferung an diesen Endpunkt von Firehose zu aktivieren.

  - Secret

    Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den API-Schlüssel für API Gateway enthält, der von Lambda unterstützt wird, das mit MongoDB Atlas interagiert. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP oder Deaktiviert, um den enable/disable Inhalt Ihrer Anfrage zu kodieren.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den ausgewählten Drittanbieter zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Konfigurieren Sie die Zieleinstellungen für New Relic

In diesem Abschnitt werden Optionen für die Verwendung von New Relic als Ziel beschrieben. Weitere Informationen finden Sie unter https://newrelic.com.

- Geben Sie Werte für folgende Felder an:

  URL des HTTP-Endpunkts

  Wählen Sie die HTTP-Endpunkt-URL aus den folgenden Optionen in der Drop-down-Liste aus.

  - New-Relic-Protokolle – USA
  - New-Relic-Metriken – USA
  - New-Relic-Metriken – EU

  Authentifizierung

  Sie können entweder den API-Schlüssel direkt eingeben oder das Geheimnis von abrufen, AWS Secrets Manager um auf New Relic zuzugreifen.

  - API-Schlüssel

    Geben Sie Ihren Lizenzschlüssel, eine 40-stellige hexadezimale Zeichenfolge, in den Einstellungen Ihres New Relic One Accounts ein. Sie benötigen diesen API-Schlüssel, um die Datenlieferung von Firehose an diesen Endpunkt zu ermöglichen.

  - Secret

    Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den API-Schlüssel für New Relic enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in AWS Secrets Manager. Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP oder Deaktiviert, um den enable/disable Inhalt Ihrer Anfrage zu kodieren.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den New Relic HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für Snowflake

In diesem Abschnitt werden Optionen für die Verwendung von Snowflake für Ihr Ziel beschrieben.

> **Note**
>
> Die Firehose-Integration mit Snowflake ist in den folgenden Ländern verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Europa (Irland), USA Ost (Ohio), Asien-Pazifik (Tokio), Europa (Frankfurt), Asien-Pazifik (Singapur), Asien-Pazifik (Seoul) und Asien-Pazifik (Sydney), Asien-Pazifik (Mumbai), Europa (London), Südamerika (Sao Paulo), Kanada (Zentral), Europa (Paris), Asien-Pazifik, Asien-Pazifik (Osaka), Europa (Stockholm), Asien-Pazifik (Jakarta). AWS-Regionen

Verbindungseinstellungen

- Geben Sie Werte für folgende Felder an:

  URL des Snowflake-Kontos

  Geben Sie eine Snowflake-Konto-URL an. Beispiel: `xy12345.us-east-1.aws.snowflakecomputing.com`. Informationen zum Ermitteln Ihrer Konto-URL finden Sie in der Snowflake-Dokumentation. Beachten Sie, dass Sie die Portnummer nicht angeben dürfen, wohingegen das Protokoll (https://) optional ist.

  Authentifizierung

  Sie können entweder den Benutzernamen, den privaten Schlüssel und die Passphrase manuell eingeben oder das Geheimnis für den Zugriff auf Snowflake abrufen. AWS Secrets Manager

  - Anmeldung des Benutzers

Geben Sie den Snowflake-Benutzer an, der zum Laden von Daten verwendet werden soll. Stellen Sie sicher, dass der Benutzer Zugriff auf das Einfügen von Daten in die Snowflake-Tabelle hat.

- Privater Aktivierungsschlüssel

  Geben Sie den privaten Schlüssel für die Authentifizierung mit Snowflake im Format an. PKCS8 Nehmen Sie außerdem keine PEM-Header und -Fußzeile als Teil des privaten Schlüssels auf. Wenn der Schlüssel auf mehrere Zeilen aufgeteilt ist, entfernen Sie die Zeilenumbrüche. Im Folgenden finden Sie ein Beispiel dafür, wie Ihr privater Schlüssel aussehen muss.

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

  Entfernen Sie das Leerzeichen darin KEY_CONTENT und stellen Sie es Firehose zur Verfügung. Es sind keine Zeichen header/footer oder Zeilenumbrüche erforderlich.

- Passphrase

  Geben Sie die Passphrase zum Entschlüsseln des verschlüsselten privaten Schlüssels an. Sie können dieses Feld leer lassen, wenn der private Schlüssel nicht verschlüsselt ist. Weitere Informationen finden Sie unter Verwenden von Schlüsselpaar-Authentifizierung und Schlüsselrotation.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das die Anmeldeinformationen für Snowflake enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Konfiguration der Rollen

Standard-Snowflake-Rolle verwenden — Wenn diese Option ausgewählt ist, gibt Firehose keine Rolle an Snowflake weiter. Es wird davon ausgegangen, dass die Standardrolle Daten lädt. Bitte stellen Sie sicher, dass die Standardrolle berechtigt ist, Daten in die Snowflake-Tabelle einzufügen.

Benutzerdefinierte Snowflake-Rolle verwenden — Geben Sie eine nicht standardmäßige Snowflake-Rolle ein, die Firehose beim Laden von Daten in die Snowflake-Tabelle übernehmen soll.

Snowflake-Konnektivität

Die Optionen sind „Privat“ oder „Öffentlich“.

Private VPCE-ID (optional)

Die VPCE-ID für Firehose, um sich privat mit Snowflake zu verbinden. Das ID-Format ist com.amazonaws.vpce. [Region] .vpce-svc-. *[id]* Weitere Informationen finden Sie unter & Snowflake.AWS PrivateLink

> **Note**
>
> Wenn Ihr Snowflake-Cluster für private Links aktiviert ist, verwenden Sie eine `AwsVpceIds` basierte Netzwerkrichtlinie, um Amazon Data Firehose-Daten zuzulassen. Firehose verlangt nicht, dass Sie eine IP-basierte Netzwerkrichtlinie in Ihrem Snowflake-Konto konfigurieren. Die Aktivierung einer IP-basierten Netzwerkrichtlinie könnte die Firehose-Konnektivität beeinträchtigen. Wenn Sie einen Sonderfall haben, für den IP-basierte Richtlinien erforderlich sind, wenden Sie sich an das Firehose-Team, indem Sie ein Support-Ticket einreichen. Eine Liste der VPCE IDs , die Sie verwenden können, finden Sie im. Zugreifen auf Snowflake in VPC

Datenbankkonfiguration

- Sie müssen die folgenden Einstellungen angeben, um Snowflake als Ziel für Ihren Firehose-Stream zu verwenden.
  - Snowflake-Datenbank — Alle Daten in Snowflake werden in Datenbanken verwaltet.
  - Snowflake-Schema — Jede Datenbank besteht aus einem oder mehreren Schemas, bei denen es sich um logische Gruppierungen von Datenbankobjekten wie Tabellen und Ansichten handelt
  - Snowflake-Tabelle — Alle Daten in Snowflake werden in Datenbanktabellen gespeichert, die logisch als Sammlungen von Spalten und Zeilen strukturiert sind.

Optionen zum Laden von Daten für Ihre Snowflake-Tabelle

- Verwenden Sie JSON-Schlüssel als Spaltennamen
- Verwenden Sie VARIANT-Spalten
  - Name der Inhaltsspalte — Geben Sie einen Spaltennamen in der Tabelle an, in die die Rohdaten geladen werden müssen.
  - Name der Metadatenspalte (optional) — Geben Sie einen Spaltennamen in der Tabelle an, in die die Metadateninformationen geladen werden müssen. Wenn Sie dieses Feld aktivieren, wird in der Snowflake-Tabelle je nach Quelltyp die folgende Spalte angezeigt.

    Für Direct PUT als Quelle

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

    Für Kinesis Data Stream als Quelle

    ```
    {
    "kinesisStreamName" : "streamname",
    "kinesisShardId" : "Id",
    "kinesisPartitionKey" : "key",
    "kinesisSequenceNumber" : "1234",
    "subsequenceNumber" : "2334",
    "IngestionTime" : "timestamp"
    }
    ```

Retry duration

Zeitdauer (0—7200 Sekunden), bis Firehose es erneut versucht, falls das Öffnen des Kanals oder die Zustellung an Snowflake aufgrund von Problemen mit dem Snowflake-Dienst fehlschlägt. Firehose wiederholt es mit exponentiellem Backoff, bis die Wiederholungsdauer endet. Wenn Sie die Wiederholungsdauer auf 0 (Null) Sekunden festlegen, versucht Firehose bei Snowflake-Fehlern nicht erneut und leitet Daten an den Amazon S3 S3-Fehler-Bucket weiter.

Puffer-Hinweise

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich. Weitere Informationen finden Sie unter Pufferhinweise konfigurieren.

Konfigurieren Sie die Zieleinstellungen für Splunk

In diesem Abschnitt werden Optionen für die Verwendung von Splunk als Ziel beschrieben.

> **Note**
>
> Firehose liefert Daten an Splunk-Cluster, die mit Classic Load Balancer oder einem Application Load Balancer konfiguriert sind.

- Geben Sie Werte für folgende Felder an:

  Splunk cluster-Endpunkt

  Informationen zur Bestimmung des Endpunkts finden Sie in der Splunk-Dokumentation unter Amazon Data Firehose zum Senden von Daten an die Splunk-Plattform konfigurieren.

  Splunk-Endpunkttypen

  Wählen Sie in den meisten Fällen `Raw endpoint`. Wählen Sie aus`Event endpoint`, ob Sie Ihre Daten vorverarbeitet haben, um Daten je AWS Lambda nach Ereignistyp an verschiedene Indizes zu senden. Informationen darüber, welcher Endpunkt verwendet werden soll, finden Sie in der Splunk-Dokumentation unter Amazon Data Firehose für das Senden von Daten an die Splunk-Plattform konfigurieren.

  Authentifizierung

  Sie können entweder das Authentifizierungstoken direkt eingeben oder das Geheimnis für den Zugriff auf Splunk abrufen. AWS Secrets Manager

  - Authentifizierungstoken

    Informationen zum Einrichten eines Splunk-Endpunkts, der Daten von Amazon Data Firehose empfangen kann, finden Sie in der Splunk-Dokumentation unter Installations- und Konfigurationsübersicht für das Splunk-Add-on für Amazon Data Firehose. Speichern Sie das Token, das Sie von Splunk erhalten, wenn Sie den Endpunkt für diesen Firehose-Stream einrichten, und fügen Sie es hier hinzu.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das das Authentifizierungstoken für Splunk enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

HEC Bestätigungs-Timeout

Geben Sie an, wie lange Amazon Data Firehose auf die Indexbestätigung von Splunk wartet. Wenn Splunk die Bestätigung nicht sendet, bevor das Timeout erreicht ist, betrachtet Amazon Data Firehose dies als Fehler bei der Datenzustellung. Amazon Data Firehose versucht dann entweder erneut, oder erstellt eine Sicherungskopie der Daten in Ihrem Amazon S3 S3-Bucket, je nachdem, welchen Wert Sie für die Wiederholungsdauer festgelegt haben.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an Splunk zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung von Splunk. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an Splunk sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung von Splunk gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel variiert je nach Dienstanbieter.

Konfigurieren Sie die Zieleinstellungen für Splunk Observability Cloud

In diesem Abschnitt werden Optionen für die Verwendung von Splunk Observability Cloud als Ziel beschrieben. Weitere Informationen finden Sie unter -apiconfig.html# https://docs.splunk.com/observability/en/gdi/get-data-in/connect/aws/aws- -api connect-to-aws-using. the-splunk-observability-cloud

- Geben Sie Werte für folgende Felder an:

  URL des Cloud-Ingest-Endpunkts

  Sie finden die URL für die Echtzeit-Datenaufnahme Ihrer Splunk Observability Cloud in der Splunk-Observability-Konsole unter Profil > Organisationen > Endpunkt zur Echtzeit-Datenerfassung.

  Authentifizierung

  Sie können entweder das Zugriffstoken direkt eingeben oder das Geheimnis für den Zugriff auf Splunk Observability Cloud abrufen. AWS Secrets Manager

  - Zugriffstoken

    Kopieren Sie Ihr Splunk Observability-Zugriffstoken mit dem INGEST-Autorisierungsbereich von Access Tokens unter Einstellungen in der Splunk Observability-Konsole.

  - Secret

    Wählen Sie ein Geheimnis aus, das das Zugriffstoken für AWS Secrets Manager Splunk Observability Cloud enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in. AWS Secrets Manager Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP oder Deaktiviert, um den enable/disable Inhalt Ihrer Anfrage zu kodieren.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an den ausgewählten HTTP-Endpunkt zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für Sumo Logic

In diesem Abschnitt werden Optionen für die Verwendung von Sumo Logic als Ziel beschrieben. Weitere Informationen finden Sie unter https://www.sumologic.com.

- Geben Sie Werte für folgende Felder an:

  URL des HTTP-Endpunkts

  Geben Sie die URL für den HTTP-Endpunkt im folgenden Format an: `https://deployment name.sumologic.net/receiver/v1/kinesis/dataType/access token`. Die URL muss eine HTTPS-URL sein.

  Inhaltskodierung

  Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP oder Deaktiviert, um den enable/disable Inhalt Ihrer Anfrage zu kodieren.

  Retry duration

  Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an Sumo Logic zu senden.

  Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

  Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Elastic-Ziel ist von Dienstanbieter zu Dienstanbieter unterschiedlich.

Konfigurieren Sie die Zieleinstellungen für Elastic

In diesem Abschnitt werden Optionen für die Verwendung von Elastic als Ziel beschrieben.

- Geben Sie Werte für folgende Felder an:

  Elastic Endpunkt-URL

  Geben Sie die URL für den HTTP-Endpunkt im folgenden Format an: `https://<cluster-id>.es.<region>.aws.elastic-cloud.com`. Die URL muss eine HTTPS-URL sein.

  Authentifizierung

  Sie können entweder den API-Schlüssel direkt eingeben oder den geheimen Schlüssel von abrufen, AWS Secrets Manager um auf Elastic zuzugreifen.

  - API-Schlüssel

    Wenden Sie sich an Elastic, um von Firehose den API-Schlüssel zu erhalten, den Sie benötigen, um die Datenlieferung an ihren Service zu aktivieren.

- Secret

  Wählen Sie ein Geheimnis aus AWS Secrets Manager , das den API-Schlüssel für Elastic enthält. Wenn Sie Ihr Geheimnis nicht in der Drop-down-Liste sehen, erstellen Sie eines in AWS Secrets Manager. Weitere Informationen finden Sie unter Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose.

Inhaltskodierung

Amazon Data Firehose verwendet Inhaltskodierung, um den Hauptteil einer Anfrage zu komprimieren, bevor sie an das Ziel gesendet wird. Wählen Sie GZIP (was standardmäßig ausgewählt ist) oder Deaktiviert für die enable/disable Inhaltskodierung Ihrer Anfrage.

Retry duration

Geben Sie an, wie lange Amazon Data Firehose erneut versucht, Daten an Elastic zu senden.

Nach dem Senden von Daten wartet Amazon Data Firehose zunächst auf eine Bestätigung vom HTTP-Endpunkt. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Es wird so lange wiederholt, bis die Dauer des Wiederholungsversuchs abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an den HTTP-Endpunkt sendet (entweder beim ersten Versuch oder bei einem erneuten Versuch), wird der Timeout-Zähler für die Bestätigung neu gestartet und auf eine Bestätigung vom HTTP-Endpunkt gewartet.

Selbst wenn die Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, bestimmt Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Wenn Sie nicht möchten, dass Amazon Data Firehose erneut versucht, Daten zu senden, setzen Sie diesen Wert auf 0.

Parameter – optional

Amazon Data Firehose schließt diese Schlüssel-Wert-Paare in jedem HTTP-Aufruf ein. Diese Parameter können Ihnen helfen Ihnen, Ihre Ziele zu identifizieren und zu organisieren.

Hinweise zum Puffern

Amazon Data Firehose puffert eingehende Daten, bevor sie an das angegebene Ziel gesendet werden. Die empfohlene Puffergröße für das Elastic-Ziel beträgt 1 MiB.

Konfigurieren Sie die Backup-Einstellungen

Amazon Data Firehose verwendet Amazon S3, um alle oder nur fehlgeschlagene Daten zu sichern, die versucht werden, an das von Ihnen gewählte Ziel zu liefern.

> **Important**
>
> - Backup-Einstellungen werden nur unterstützt, wenn die Quelle für Ihren Firehose-Stream Direct PUT oder Kinesis Data Streams ist.
> - Die Funktion Zero Buffering ist nur für die Anwendungsziele und nicht für das Amazon S3 S3-Backup-Ziel verfügbar.

Sie können die S3-Backup-Einstellungen für Ihren Firehose-Stream angeben, wenn Sie eine der folgenden Optionen getroffen haben.

- Wenn Sie Amazon S3 als Ziel für Ihren Firehose-Stream festlegen und eine AWS Lambda-Funktion zur Transformation von Datensätzen angeben oder wenn Sie Datensatzformate für Ihren Firehose-Stream konvertieren möchten.
- Wenn Sie Amazon Redshift als Ziel für Ihren Firehose-Stream festlegen und eine AWS Lambda-Funktion zur Transformation von Datensätzen angeben.
- Wenn Sie einen der folgenden Dienste als Ziel für Ihren Firehose-Stream festlegen: Amazon OpenSearch Service, Datadog, Dynatrace, HTTP Endpoint,, MongoDB Cloud, New Relic LogicMonitor, Splunk oder Sumo Logic, Snowflake, Apache Iceberg Tables.

Im Folgenden sind die Backup-Einstellungen für Ihren Firehose-Stream aufgeführt.

- Sicherung von Quelldatensätzen in Amazon S3 – wenn S3 oder Amazon Redshift Ihr ausgewähltes Ziel ist, gibt diese Einstellung an, ob Sie die Quelldatensicherung aktivieren oder deaktivieren möchten. Wenn ein anderer unterstützter Service (außer S3 oder Amazon Redshift) als Ihr ausgewähltes Ziel festgelegt ist, gibt diese Einstellung an, ob Sie alle Ihre Quelldaten oder nur fehlerhafte Daten sichern möchten.
- S3-Backup-Bucket — das ist der S3-Bucket, in dem Amazon Data Firehose Ihre Daten sichert.
- S3-Backup-Bucket-Präfix — Dies ist das Präfix, mit dem Amazon Data Firehose Ihre Daten sichert.
- Ausgabepräfix für Fehler im S3-Backup-Bucket – alle fehlgeschlagenen Daten werden in diesem S3-Bucket-Fehlerausgabepräfix gesichert.
- Pufferhinweise, Komprimierung und Verschlüsselung für Backups — Amazon Data Firehose verwendet Amazon S3, um alle oder nur fehlgeschlagene Daten zu sichern, die versucht werden, an das von Ihnen gewählte Ziel zu liefern. Amazon Data Firehose puffert eingehende Daten, bevor sie an Amazon S3 übermittelt (gesichert) werden. Sie können eine Puffergröße von 1—128 MiBs und ein Pufferintervall von 60—900 Sekunden wählen. Die Bedingung, die erfüllt ist, löst eine erste Datenübermittlung an Amazon S3 aus. Wenn Sie die Datentransformation aktivieren, gilt das Pufferintervall vom Empfang der transformierten Daten bei Amazon Data Firehose bis zur Datenlieferung an Amazon S3. Wenn die Datenlieferung an das Ziel hinter dem Schreiben von Daten in den Firehose-Stream zurückbleibt, erhöht Amazon Data Firehose die Puffergröße dynamisch, um catch. Diese Aktion trägt dazu bei, dass alle Daten an das Ziel geliefert werden.
- S3-Komprimierung — wählen Sie GZIP-, Snappy-, Zip- oder Hadoop-kompatible Snappy-Datenkomprimierung oder keine Datenkomprimierung. Snappy-, Zip- und Hadoop-kompatible Snappy-Komprimierung ist für Firehose-Streams mit Amazon Redshift als Ziel nicht verfügbar.
- S3-Dateierweiterungsformat (optional) — Geben Sie ein Dateierweiterungsformat für Objekte an, die an den Amazon S3 S3-Ziel-Bucket geliefert werden. Wenn Sie diese Funktion aktivieren, überschreibt die angegebene Dateierweiterung die Standarddateierweiterungen, die durch Datenformatkonvertierungs- oder S3-Komprimierungsfunktionen wie .parquet oder .gz hinzugefügt wurden. Vergewissern Sie sich, dass Sie die richtige Dateierweiterung konfiguriert haben, wenn Sie diese Funktion mit Datenformatkonvertierung oder S3-Komprimierung verwenden. Die Dateierweiterung muss mit einem Punkt (.) beginnen und kann die zulässigen Zeichen enthalten: 0-9a-z! -_.*' (). Die Dateierweiterung darf 128 Zeichen nicht überschreiten.
- Firehose unterstützt die serverseitige Amazon S3-Verschlüsselung mit AWS Key Management Service (SSE-KMS) zur Verschlüsselung von gelieferten Daten in Amazon S3. Sie können wählen, ob Sie den im Ziel-S3-Bucket angegebenen Standardverschlüsselungstyp verwenden oder mit einem Schlüssel aus der Liste der Schlüssel verschlüsseln möchten, die Sie besitzen. AWS KMS Wenn Sie die Daten mit AWS KMS Schlüsseln verschlüsseln, können Sie entweder den AWS

verwalteten Standardschlüssel (aws/s3) oder einen vom Kunden verwalteten Schlüssel verwenden. Weitere Informationen finden Sie unter Schutz von Daten mithilfe serverseitiger Verschlüsselung mit AWS KMS-verwalteten Schlüsseln (SSE-KMS).

Pufferhinweise konfigurieren

Amazon Data Firehose puffert eingehende Streaming-Daten im Speicher auf eine bestimmte Größe (Puffergröße) und für einen bestimmten Zeitraum (Pufferintervall), bevor sie an die angegebenen Ziele gesendet werden. Sie würden Pufferhinweise verwenden, wenn Sie Dateien mit optimaler Größe an Amazon S3 senden und eine bessere Leistung von Datenverarbeitungsanwendungen erzielen möchten oder um die Firehose-Zustellungsrate an die Zielgeschwindigkeit anzupassen.

Sie können die Puffergröße und das Pufferintervall beim Erstellen neuer Firehose-Streams konfigurieren oder die Puffergröße und das Pufferintervall für Ihre vorhandenen Firehose aktualisieren. Die Puffergröße wird in Sekunden gemessen MBs und das Pufferintervall wird in Sekunden gemessen. Wenn Sie jedoch für einen dieser beiden Parameter einen Wert angeben, müssen Sie auch für den anderen Parameter einen Wert angeben. Die erste Pufferbedingung, die erfüllt ist, veranlasst Firehose, die Daten zu liefern. Wenn Sie die Pufferwerte nicht konfigurieren, werden die Standardwerte verwendet.

Sie können Firehose-Pufferhinweise über AWS-Managementkonsole AWS Command Line Interface, oder konfigurieren. AWS SDKs Für bestehende Streams können Sie die Pufferhinweise mit einem Wert neu konfigurieren, der Ihren Anwendungsfällen entspricht, indem Sie die Option Bearbeiten in der Konsole oder die API verwenden. UpdateDestination Für neue Streams können Sie Pufferhinweise als Teil der Erstellung neuer Streams mithilfe der Konsole oder mithilfe der API konfigurieren. CreateDeliveryStream Um die Puffergröße anzupassen, legen Sie `SizeInMBs` und `IntervalInSeconds` in den zielspezifischen `DestinationConfiguration` Parameter der CreateDeliveryStreamUpdateDestinationOR-API fest.

> **Note**
>
> - Pufferhinweise werden auf Shard- oder Partitionsebene angewendet, während Pufferhinweise für dynamische Partitionierungen auf Stream- oder Themenebene angewendet werden.
> - Um geringeren Latenzen bei Echtzeit-Anwendungsfällen gerecht zu werden, können Sie einen Hinweis ohne Pufferintervall verwenden. Wenn Sie das Pufferintervall auf Null Sekunden konfigurieren, puffert Firehose keine Daten und liefert Daten innerhalb weniger

Sekunden. Bevor Sie die Pufferhinweise auf einen niedrigeren Wert ändern, erkundigen Sie sich beim Anbieter nach den empfohlenen Pufferhinweisen von Firehose für deren Ziele.

- Die Funktion Zero Buffering ist nur für die Anwendungsziele und nicht für das Amazon S3 S3-Backup-Ziel verfügbar.
- Die Funktion Zero Buffering ist für dynamische Partitionierung nicht verfügbar.
- Firehose verwendet mehrteiligen Upload für das S3-Ziel, wenn Sie ein Pufferzeitintervall von weniger als 60 Sekunden konfigurieren, um geringere Latenzen zu bieten. Aufgrund des mehrteiligen Uploads für das S3-Ziel werden Sie einen gewissen Anstieg der PUT S3-API-Kosten feststellen, wenn Sie ein Pufferzeitintervall von weniger als 60 Sekunden wählen.

Die Bereiche und Standardwerte für zielspezifische Pufferhinweise finden Sie in der folgenden Tabelle:

Ziel	Puffergröße in MB (Standard in Klammern)	Pufferintervall in Sekunden (Standard in Klammern)
Amazon S3	1-128 (5)	0-900 (300)
Apache Iceberg-Tabellen	1-128 (5)	0-900 (300)
Amazon Redshift	1-128 (5)	0-900 (300)
OpenSearch Serverlos	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)

Ziel	Puffergröße in MB (Standard in Klammern)	Pufferintervall in Sekunden (Standard in Klammern)
Dynatrace	1-64 (5)	0-900 (60)
Elastic	1	0-900 (60)
Honeycomb	1-64 (15)	0-900 (60)
HTTP-Endpunkt	1-64 (5)	0-900 (60)
LogicMonitor	1-64 (5)	0-900 (60)
Logik	1-64 (5)	0-900 (60)
MongoDB	1-16 (5)	0-900 (60)
Neues Relikt	1-64 (5)	0-900 (60)
SumoLogic	1-64 (1)	0-900 (60)
Splunk Observability Cloud	1-64 (1)	0-900 (60)
Snowflake	1-128 (1)	0 bis 900 (0)

Konfigurieren von erweiterten Einstellungen

Der folgende Abschnitt enthält Details zu den erweiterten Einstellungen für Ihren Firehose-Stream.

- Serverseitige Verschlüsselung — Amazon Data Firehose unterstützt die serverseitige Amazon S3-Verschlüsselung mit AWS Key Management Service (AWS KMS) zur Verschlüsselung der in Amazon S3 übermittelten Daten. Weitere Informationen finden Sie unter Schutz von Daten mithilfe serverseitiger Verschlüsselung mit AWS KMS-verwalteten Schlüsseln (SSE-KMS).
- Fehlerprotokollierung — Amazon Data Firehose protokolliert Fehler im Zusammenhang mit der Verarbeitung und Lieferung. Wenn die Datentransformation aktiviert ist, kann sie außerdem Lambda-Aufrufe protokollieren und Fehler bei der Datenübermittlung an Logs senden. CloudWatch

Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch .

> **Important**
>
> Obwohl optional, wird dringend empfohlen, die Amazon Data Firehose-Fehlerprotokollierung während Firehose Firehose-Stream-Erstellung zu aktivieren. Diese Vorgehensweise stellt sicher, dass Sie im Falle von Fehlern bei der Verarbeitung oder Übermittlung von Datensätzen auf Fehlerdetails zugreifen können.

- Berechtigungen — Amazon Data Firehose verwendet IAM-Rollen für alle Berechtigungen, die der Firehose-Stream benötigt. Sie können wählen, ob Sie eine neue Rolle erstellen, bei der die erforderlichen Berechtigungen automatisch zugewiesen werden, oder eine bestehende Rolle wählen, die für Amazon Data Firehose erstellt wurde. Die Rolle wird verwendet, um Firehose Zugriff auf verschiedene Dienste zu gewähren, darunter Ihren S3-Bucket, Ihren AWS KMS-Schlüssel (wenn die Datenverschlüsselung aktiviert ist) und die Lambda-Funktion (wenn die Datentransformation aktiviert ist). Die Konsole erstellt möglicherweise eine Rolle mit Platzhaltern. Weitere Informationen finden Sie unter Was ist IAM?.

  > **Note**
  >
  > Die IAM-Rolle (einschließlich Platzhaltern) wird auf der Grundlage der Konfiguration erstellt, die Sie beim Erstellen eines Firehose-Streams ausgewählt haben. Wenn Sie Änderungen an der Firehose-Stream-Quelle oder dem Ziel vornehmen, müssen Sie die IAM-Rolle manuell aktualisieren.

- Tags — Sie können Tags hinzufügen, um Ihre AWS Ressourcen zu organisieren, Kosten zu verfolgen und den Zugriff zu kontrollieren.

  Wenn Sie in der `CreateDeliveryStream` Aktion Tags angeben, führt Amazon Data Firehose eine zusätzliche Autorisierung für die `firehose:TagDeliveryStream` Aktion durch, um zu überprüfen, ob Benutzer berechtigt sind, Tags zu erstellen. Wenn Sie diese Berechtigung nicht erteilen, schlagen Anfragen zum Erstellen neuer Firehose-Streams mit IAM-Ressourcen-Tags fehl, und zwar mit einem `AccessDeniedException` solchen Fehler wie dem Folgenden.

  ```
  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.
```

Das folgende Beispiel zeigt eine Richtlinie, die es Benutzern ermöglicht, einen Firehose-Stream zu erstellen und Tags anzuwenden.

Nachdem Sie Ihr Backup und Ihre erweiterten Einstellungen ausgewählt haben, überprüfen Sie Ihre Auswahl und wählen Sie dann Firehose-Stream erstellen.

Der neue Firehose-Stream benötigt im Status Creating einen Moment, bis er verfügbar ist. Sobald sich Ihr Firehose-Stream im Status Aktiv befindet, können Sie damit beginnen, Daten von Ihrem Producer an ihn zu senden.

Testen des Firehose-Streams mit Beispieldaten

Sie können das verwenden AWS-Managementkonsole , um simulierte Börsentickerdaten aufzunehmen. Die Konsole führt ein Skript in Ihrem Browser aus, um Beispieldatensätze in Ihren Firehose-Stream einzufügen. Auf diese Weise können Sie die Konfiguration Ihres Firehose-Streams testen, ohne Ihre eigenen Testdaten generieren zu müssen.

Es folgt ein Beispiel für simulierte Daten:

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

Beachten Sie, dass die Standardgebühren von Amazon Data Firehose anfallen, wenn Ihr Firehose-Stream die Daten überträgt, aber keine Gebühren anfallen, wenn die Daten generiert werden. Damit diese Gebühren nicht mehr anfallen, können Sie den Beispiel-Stream über die Konsole jederzeit beenden.

Voraussetzungen

Bevor Sie beginnen, erstellen Sie einen Firehose-Stream. Weitere Informationen finden Sie unter Tutorial: Einen Firehose-Stream von der Konsole aus erstellen.

Testen Sie mit Amazon S3

Verwenden Sie das folgende Verfahren, um Ihren Firehose-Stream mit Amazon Simple Storage Service (Amazon S3) als Ziel zu testen.

Um einen Firehose-Stream mit Amazon S3 zu testen

1. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie einen aktiven Firehose-Stream. Der Firehose-Stream muss den Status Aktiv haben, bevor Sie mit dem Senden von Daten beginnen können.
3. Wählen Sie unter Test with demo data die Option Start sending demo data, um Börsenticker-Beispieldaten zu generieren.
4. Befolgen Sie die Anweisungen auf dem Bildschirm, um zu überprüfen, ob die Daten an Ihren S3-Bucket übermittelt werden. Beachten Sie, dass es je nach der Pufferkonfiguration Ihres Buckets einige Minuten dauern kann, bis neue Objekte in Ihrem Bucket angezeigt werden.

5. Wenn der Test abgeschlossen ist, wählen Sie Stop sending demo data, damit keine nutzungsabhängigen Gebühren mehr anfallen.

Testen Sie mit Amazon Redshift

Verwenden Sie das folgende Verfahren, um Ihren Firehose-Stream mit Amazon Redshift als Ziel zu testen.

So testen Sie einen Firehose-Stream mit Amazon Redshift

1. Ihr Firehose-Stream erwartet, dass eine Tabelle in Ihrem Amazon Redshift Redshift-Cluster vorhanden ist. Stellen Sie eine Verbindung mit Amazon Redshift über eine SQL-Schnittstelle her und führen Sie die folgende Anweisung aus, um eine Tabelle zu erstellen, die die Beispieldaten akzeptiert.

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

2. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
3. Wählen Sie einen aktiven Firehose-Stream. Der Firehose-Stream muss den Status Aktiv haben, bevor Sie mit dem Senden von Daten beginnen können.
4. Bearbeiten Sie die Zieldetails für Ihren Firehose-Stream so, dass sie auf die neu erstellte `firehose_test_table` Tabelle verweisen.
5. Wählen Sie unter Test with demo data die Option Start sending demo data, um Börsenticker-Beispieldaten zu generieren.
6. Befolgen Sie die Anweisungen auf dem Bildschirm, um zu überprüfen, ob die Daten an Ihre Tabelle übermittelt werden. Beachten Sie, dass es je nach der Pufferkonfiguration einige Minuten dauern kann, bis neue Zeilen in Ihrer Tabelle angezeigt werden.
7. Wenn der Test abgeschlossen ist, wählen Sie Stop sending demo data, damit keine nutzungsabhängigen Gebühren mehr anfallen.
8. Bearbeiten Sie die Zieldetails für Ihren Firehose-Stream so, dass sie auf eine andere Tabelle verweisen.

9. (Optional) Löschen Sie die `firehose_test_table`-Tabelle.

Testen Sie mit Service OpenSearch

Verwenden Sie das folgende Verfahren, um Ihren Firehose-Stream mit Amazon OpenSearch Service als Ziel zu testen.

Um einen Firehose-Stream mit OpenSearch Service zu testen

1. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie einen aktiven Firehose-Stream. Der Firehose-Stream muss den Status Aktiv haben, bevor Sie mit dem Senden von Daten beginnen können.
3. Wählen Sie unter Test with demo data die Option Start sending demo data, um Börsenticker-Beispieldaten zu generieren.
4. Folgen Sie den Anweisungen auf dem Bildschirm, um zu überprüfen, ob Daten an Ihre OpenSearch Service-Domain übermittelt werden. Weitere Informationen finden Sie unter Suchen nach Dokumenten in einer OpenSearch Service-Domain im Amazon OpenSearch Service Developer Guide.
5. Wenn der Test abgeschlossen ist, wählen Sie Stop sending demo data, damit keine nutzungsabhängigen Gebühren mehr anfallen.

Testen Sie mit Splunk

Verwenden Sie das folgende Verfahren, um Ihren Firehose-Stream mit Splunk als Ziel zu testen.

So testen Sie einen Firehose-Stream mit Splunk

1. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie einen aktiven Firehose-Stream. Der Firehose-Stream muss den Status Aktiv haben, bevor Sie mit dem Senden von Daten beginnen können.
3. Wählen Sie unter Test with demo data die Option Start sending demo data, um Börsenticker-Beispieldaten zu generieren.
4. Überprüfen Sie, ob die Daten an Ihren Splunk-Index gesendet werden. Beispiel für Suchbegriffe in Splunk sind `sourcetype="aws:firehose:json"` und `index="name-of-your-splunk-index"`. Weitere Informationen zum Suchen von Ereignissen in Splunk finden Sie unter Search Manual in der Splunk-Dokumentation.

Wenn die Testdaten nicht in Ihrem Splunk-Index erscheinen, überprüfen Sie Ihren Amazon-S3-Bucket auf fehlgeschlagene Ereignisse. Lesen Sie auch unter Daten wurden nicht an Splunk bereitgestellt nach.

5. Wenn Sie den Test abgeschlossen haben, wählen Sie Stop sending demo data, damit keine nutzungsabhängigen Gebühren mehr anfallen.

Testen Sie mit Apache Iceberg-Tabellen

Verwenden Sie das folgende Verfahren, um Ihren Firehose-Stream mit Apache Iceberg Tables als Ziel zu testen.

Um einen Firehose-Stream mit Apache Iceberg Tables zu testen

1. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie einen aktiven Firehose-Stream. Der Firehose-Stream muss den Status Aktiv haben, bevor Sie mit dem Senden von Daten beginnen können.
3. Wählen Sie unter Test with demo data die Option Start sending demo data, um Börsenticker-Beispieldaten zu generieren.
4. Folgen Sie den Anweisungen auf dem Bildschirm, um zu überprüfen, ob Daten an Ihre Apache Iceberg-Tabellen übertragen werden. Beachten Sie, dass es je nach Pufferkonfiguration einige Minuten dauern kann, bis neue Objekte in Ihrem Bucket erscheinen.
5. Wenn die Testdaten nicht in Ihren Apache Iceberg-Tabellen erscheinen, überprüfen Sie Ihren Amazon S3 S3-Bucket auf fehlgeschlagene Ereignisse.
6. Wenn Sie den Test abgeschlossen haben, wählen Sie Stop sending demo data, damit keine nutzungsabhängigen Gebühren mehr anfallen.

Daten an einen Firehose-Stream senden

In diesem Abschnitt wird beschrieben, wie Sie verschiedene Datenquellen verwenden können, um Daten an Ihren Firehose-Stream zu senden. Wenn Sie Amazon Data Firehose noch nicht kennen, nehmen Sie sich etwas Zeit, um sich mit den Konzepten und der Terminologie in Was ist Amazon Data Firehose? vertraut zu machen.

> **Note**
>
> Einige AWS Dienste können nur Nachrichten und Ereignisse an einen Firehose-Stream senden, der sich in derselben Region befindet. Wenn Ihr Firehose-Stream bei der Konfiguration eines Ziels für Amazon CloudWatch Logs, CloudWatch Events oder nicht als Option angezeigt wird, stellen Sie sicher AWS IoT, dass sich Ihr Firehose-Stream in derselben Region wie Ihre anderen Dienste befindet. Informationen zu Service-Endpunkten für jede Region finden Sie unter Amazon Data Firehose-Endpunkte.

Sie können Daten aus den folgenden Datenquellen an Ihren Firehose-Stream senden.

Themen

Kinesis-Agent für das Senden von Daten konfigurieren

Amazon Kinesis Agent ist eine eigenständige Java-Softwareanwendung, die als Referenzimplementierung dient und zeigt, wie Sie Daten sammeln und an Firehose senden können. Der Agent überwacht kontinuierlich eine Reihe von Dateien und sendet neue Daten an Ihren Firehose-Stream. Der Agent zeigt, wie Sie mit Dateirotation, Checkpoints und Wiederholungsversuchen bei Fehlern umgehen können. Er zeigt, wie Sie Ihre Daten zuverlässig, zeitnah und einfach bereitstellen können. Außerdem wird gezeigt, wie Sie CloudWatch Metriken

ausgeben können, um den Streaming-Prozess besser zu überwachen und Fehler zu beheben. Weitere Informationen finden Sie unter awslabs/ amazon-kinesis-agent.

Standardmäßig werden Datensätze aus den einzelnen Dateien anhand des Zeilenumbruchzeichens (`'\n'`) analysiert. Der Agent kann jedoch auch für die Analyse mehrzeiliger Datensätze konfiguriert werden (siehe Geben Sie die Einstellungen der Agentenkonfiguration an).

Sie können den Agenten in Linux-Serverumgebungen installieren, beispielsweise auf Webservern, Protokollservern und Datenbankservern. Nachdem Sie den Agenten installiert haben, konfigurieren Sie ihn, indem Sie die zu überwachenden Dateien und den Firehose-Stream für die Daten angeben. Nach der Konfiguration sammelt der Agent dauerhaft Daten aus den Dateien und sendet sie zuverlässig an den Firehose-Stream.

Voraussetzungen

Bevor Sie Kinesis Agent verwenden, stellen Sie sicher, dass Sie die folgenden Voraussetzungen erfüllen.

- Ihr Betriebssystem muss Amazon Linux oder Red Hat Enterprise Linux Version 7 oder höher sein.
- Agent-Version 2.0.0 oder höher wird mit JRE-Version 1.8 oder höher ausgeführt. Agent-Version 1.1.x wird mit JRE 1.7 oder höher ausgeführt.
- Wenn Sie Amazon verwenden EC2 , um Ihren Agenten auszuführen, starten Sie Ihre EC2 Instance.
- Die von Ihnen angegebene IAM-Rolle oder die AWS Anmeldeinformationen müssen berechtigt sein, den Amazon Data PutRecordBatchFirehose-Vorgang auszuführen, damit der Agent Daten an Ihren Firehose-Stream senden kann. Wenn Sie die CloudWatch Überwachung für den Agenten aktivieren, ist auch eine Genehmigung zur Durchführung des CloudWatch PutMetricDataVorgangs erforderlich. Weitere Informationen finden Sie unter Zugriffskontrolle mit Amazon Data FirehoseÜberwachen Sie den Zustand des Kinesis-Agenten, und Authentifizierung und Zugriffskontrolle für Amazon CloudWatch.

AWS Zugangsdaten verwalten

Verwalten Sie Ihre AWS Anmeldeinformationen mit einer der folgenden Methoden:

- Erstellen Sie einen Anbieter benutzerdefinierter Anmeldeinformationen. Details hierzu finden Sie unter the section called “Erstellen Sie benutzerdefinierte Anbieter für Anmeldeinformationen”.
- Geben Sie beim Starten Ihrer EC2 Instance eine IAM-Rolle an.

- Geben Sie bei der Konfiguration des Agenten AWS Anmeldeinformationen an (siehe die Einträge für `awsAccessKeyId` und `awsSecretAccessKey` in der Konfigurationstabelle unterthe section called “Geben Sie die Einstellungen der Agentenkonfiguration an”).
- Bearbeiten Sie`/etc/sysconfig/aws-kinesis-agent`, um Ihre AWS Region und Ihre AWS Zugriffsschlüssel anzugeben.
- Wenn sich Ihre EC2 Instance in einem anderen AWS Konto befindet, erstellen Sie eine IAM-Rolle, um Zugriff auf den Amazon Data Firehose-Service zu gewähren. Geben Sie diese Rolle bei der Konfiguration des Agenten an (siehe assumeRoleARN und assumeRoleExternal Id). Verwenden Sie eine der vorherigen Methoden, um die AWS Anmeldeinformationen eines Benutzers in dem anderen Konto anzugeben, der berechtigt ist, diese Rolle anzunehmen.

Erstellen Sie benutzerdefinierte Anbieter für Anmeldeinformationen

Sie können einen Anbieter für benutzerdefinierte Anmeldeinformationen erstellen und den Klassennamen und den JAR-Pfad zum Kinesis Agent in den folgenden Konfigurationseinstellungen angeben: `userDefinedCredentialsProvider.classname` und `userDefinedCredentialsProvider.location`. Die Beschreibungen dieser beiden Konfigurationseinstellungen finden Sie unter the section called “Geben Sie die Einstellungen der Agentenkonfiguration an”.

Um einen Anbieter benutzerdefinierte Anmeldeinformationen zu erstellen, definieren Sie eine Klasse, die die `AWS CredentialsProvider`-Schnittstelle implementiert, wie im folgenden Beispiel.

```
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() {
    }
}
```

Ihre Klasse muss über einen Konstruktor verfügen, der keine Argumente annimmt.

AWS ruft die Refresh-Methode regelmäßig auf, um aktualisierte Anmeldeinformationen abzurufen. Wenn Ihr Anmeldeinformationsanbieter während seiner gesamten Lebensdauer unterschiedliche Anmeldeinformationen bereitstellen soll, fügen Sie Code ein, um die Anmeldeinformationen in dieser Methode zu aktualisieren. Alternativ können Sie diese Methode leer lassen, wenn Sie einen Anmeldeinformationsanbieter wünschen, der statische (nicht ändernde) Anmeldeinformationen vergibt.

Laden Sie den Agenten herunter und installieren Sie ihn

Stellen Sie zunächst eine Verbindung mit Ihrer Instance her. Weitere Informationen finden Sie unter Connect to Your Instance im EC2 Amazon-Benutzerhandbuch. Wenn Sie Probleme mit der Verbindung haben, finden Sie weitere Informationen unter Problembehandlung beim Herstellen einer Verbindung zu Ihrer Instance im EC2 Amazon-Benutzerhandbuch.

Installieren Sie als Nächstes mithilfe einer der folgenden Methoden den Agenten.

- So richten Sie den Agenten über die Amazon-Linux-Repositorys ein

  Diese Methode funktioniert nur für Amazon-Linux-Instances. Verwenden Sie den folgenden Befehl:

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

  Agent v 2.0.0 oder höher ist auf Computern mit dem Betriebssystem Amazon Linux 2 (AL2) installiert. Diese Agent-Version erfordert Java-Version 1.8 oder höher. Falls die erforderliche Java-Version noch nicht vorhanden ist, wird sie bei der Agenteninstallation installiert. Weitere Informationen zu Amazon Linux 2 finden Sie unter https://aws.amazon.com/amazon-linux-2/.
- So richten Sie den Agenten über die Amazon-S3-Repositorys ein

  Diese Methode funktioniert sowohl für Instances von Red Hat Enterprise Linux als auch von Amazon Linux 2, da sie den Agenten aus dem öffentlich verfügbaren Repository installiert. Verwenden Sie den folgenden Befehl, um die neueste Version der Agentenversion 2.x.x. herunterzuladen und zu installieren:

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

Um eine bestimmte Version des Agents zu installieren, geben Sie die Versionsnummer im Befehl an. Mit dem folgenden Befehl wird beispielsweise Agent v 2.0.1. installiert.

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

Wenn Sie Java 1.7 haben und es nicht aktualisieren möchten, können Sie die Agentenversion 1.x.x herunterladen, die mit Java 1.7 kompatibel ist. Um beispielsweise Agent v1.1.6 herunterzuladen, können Sie den folgenden Befehl verwenden:

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

Sie können den neuesten Agenten mit dem folgenden Befehl herunterladen

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

- Um den Agenten aus dem GitHub Repo einzurichten

  1. Stellen Sie zunächst sicher, dass die erforderliche Java-Version installiert ist, je nach Agentenversion.
  2. Laden Sie den Agenten aus dem amazon-kinesis-agent GitHub awslabs/Repo herunter.
  3. Installieren Sie den Agenten, indem Sie zum Download-Verzeichnis navigieren und den folgenden Befehl ausführen:

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

- So richten Sie den Agenten in einem Docker-Container ein

Kinesis Agent kann auch in einem Container über die amazonlinux-Containerbasis ausgeführt werden. Verwenden Sie die folgende Docker-Datei und führen Sie dann `docker build` aus.

```
FROM amazonlinux

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

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

Konfigurieren und starten Sie den Agenten

So konfigurieren und starten Sie den Agenten

1. Öffnen und bearbeiten Sie die Konfigurationsdatei (als Superuser, wenn Sie standardmäßige Dateizugriffsberechtigungen nutzen): `/etc/aws-kinesis/agent.json`

   Geben Sie in dieser Konfigurationsdatei die Dateien (`"filePattern"`) an, aus denen der Agent Daten sammelt, und den Namen des Firehose-Streams (`"deliveryStream"`), an den der Agent Daten sendet. Der Dateiname ein Muster ist und der Agent Dateirotationen erkennt. Sie können nur einmal pro Sekunde Dateien rotieren oder neue Dateien erstellen. Der Agent verwendet den Zeitstempel der Dateierstellung, um zu bestimmen, welche Dateien nachverfolgt und in Ihren Firehose-Stream aufgenommen werden sollen. Wenn Daten häufiger als einmal pro Sekunde neu erstellt oder rotiert werden, kann der Agent nicht richtig zwischen den Dateien unterscheiden.

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

   Die AWS Standardregion ist`us-east-1`. Wenn Sie eine andere Region verwenden, fügen Sie der Konfigurationsdatei die Einstellung `firehose.endpoint` hinzu, um den Endpunkt für Ihre

Region anzugeben. Weitere Informationen finden Sie unter Geben Sie die Einstellungen der Agentenkonfiguration an.

2. Starten Sie den Agenten manuell:

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

3. (Optional) Konfigurieren Sie den Agenten so, dass er beim Startup des Systems gestartet wird:

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

Der Agent wird jetzt als Systemdienst im Hintergrund ausgeführt. Es überwacht kontinuierlich die angegebenen Dateien und sendet Daten an den angegebenen Firehose-Stream. Die Agentenaktivität wird in `/var/log/aws-kinesis-agent/aws-kinesis-agent.log` protokolliert.

Geben Sie die Einstellungen der Agentenkonfiguration an

Der Agent unterstützt zwei obligatorische Konfigurationseinstellungen, `filePattern` und `deliveryStream`, sowie optionale Konfigurationseinstellungen für zusätzliche Funktionen. Sie können sowohl die obligatorischen als auch die optionalen Konfigurationseinstellungen in `/etc/aws-kinesis/agent.json` festlegen.

Wenn Sie die Konfigurationsdatei ändern, müssen Sie den Agenten mit den folgenden Befehlen anhalten und starten:

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

Alternativ können Sie auch den folgenden Befehl nutzen:

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

Im Folgenden finden Sie die allgemeinen Konfigurationseinstellungen.

Konfigurationseinstellung	Beschreibung
`assumeRoleARN`	Der Amazon-Ressourcenname (ARN) der Rolle, die der Benutzer übernehmen soll. Weitere Informationen finden Sie unter AWS

Konfigurationseinstellung	Beschreibung
	Kontenübergreifendes Delegieren des Zugriffs mithilfe von IAM-Rollen im IAM-Benutzerhandbuch.
`assumeRoleExternalId`	Eine optionale Kennung, die festlegt, wer die Rolle übernehmen kann. Weitere Informationen finden Sie unter Verwendung einer externen ID im IAM-Benutzerhandbuch.
`awsAccessKeyId`	AWS Zugriffsschlüssel-ID, die die Standardanmeldedaten überschreibt. Diese Einstellung hat Vorrang vor allen anderen Anbietern von Anmeldeinformationen.
`awsSecretAccessKey`	AWS geheimer Schlüssel, der die Standardanmeldedaten überschreibt. Diese Einstellung hat Vorrang vor allen anderen Anbietern von Anmeldeinformationen.
`cloudwatch.emitMetrics`	Ermöglicht dem Agenten, Metriken auszusenden, CloudWatch sofern diese Einstellung gesetzt ist (true). Standard: true
`cloudwatch.endpoint`	Der regionale Endpunkt für CloudWatch. Standard: `monitoring.us-east-1.amazonaws.com`
`firehose.endpoint`	Der regionale Endpunkt für Amazon Data Firehose. Standard: `firehose.us-east-1.amazonaws.com`
`sts.endpoint`	Der regionale Endpunkt für den AWS Security Token Service. Standard: `https://sts.amazonaws.com`
`userDefinedCredentialsProvider.classname`	Wenn Sie einen Anbieter für benutzerdefinierte Anmeldeinformationen definieren, geben Sie den vollständig qualifizierten Klassennamen mit dieser Einstellung an. Fügen Sie `.class` nicht am Ende des Klassennamens ein.

Konfigurationseinstellung	Beschreibung
`userDefinedCredentialsProvider.location`	Wenn Sie einen Anbieter für benutzerdefinierte Anmeldeinformationen definieren, verwenden Sie diese Einstellung, um den absoluten JAR-Pfad anzugeben, der den Anbieter für benutzerdefinierte Anmeldeinformationen enthält. Der Agent sucht auch am folgenden Speicherort nach der JAR-Datei: `/usr/share/aws-kinesis-agent/lib/` .

Im Folgenden finden Sie die Konfigurationseinstellungen für den Ablauf.

Konfigurationseinstellung	Beschreibung
`aggregatedRecordSizeBytes`	Geben Sie diese Einstellung an, damit der Agent Datensätze aggregiert und sie dann in einem Vorgang in den Firehose-Stream einfügt. Stellen Sie ihn auf die Größe ein, die der Aggregatdatensatz haben soll, bevor der Agent ihn in den Firehose-Stream einfügt. Standard: 0 (keine Aggregation)
`dataProcessingOptions`	Die Liste der Verarbeitungsoptionen, die auf jeden analysierten Datensatz angewendet werden, bevor er an den Firehose-Stream gesendet wird. Die Verarbeitungsoptionen werden in der angegebenen Reihenfolge ausgeführt. Weitere Informationen finden Sie unter Daten mit Agenten vorverarbeiten.
`deliveryStream`	[Erforderlich] Der Name des Firehose-Streams.
`filePattern`	[Erforderlich] Glob für die Dateien, die vom Agent überwacht werden müssen. Eine Datei, die mit diesem Muster übereinstimmt, wird vom Agenten automatisch erfasst und überwacht. Gewähren Sie `aws-kinesis-agent-user` Leseberechtigung für alle Dateien, die diesem Muster entsprechen. Gewähren Sie `aws-kinesis-agent-user` Lese- und Ausführungsberechtigungen für das Verzeichnis mit den Dateien.

Konfigurationseinstellung	Beschreibung
	Important Der Agent verarbeitet jede Datei, die diesem Muster entspricht. Dieses Muster muss sorgfältig so ausgewählt werden, dass der Agent nur die gewünschten Datensätze verarbeitet.
`initialPosition`	Die Position, an der mit der Analyse der Datei begonnen wurde. Gültige Werte sind `START_OF_FILE` und `END_OF_FILE` . Standard: `END_OF_FILE`
`maxBufferAgeMillis`	Die maximale Zeit in Millisekunden, für die der Agent Daten zwischenspeichert, bevor er sie an den Firehose-Stream sendet. Wertebereich: 1 000–900 000 (1 Sekunde bis 15 Minuten) Standard: 60.000 (1 Minute)
`maxBufferSizeBytes`	Die maximale Größe in Byte, für die der Agent Daten puffert, bevor er sie an den Firehose-Stream sendet. Wertebereich: 1–4 194 304 (4 MB) Standard: 4.194.304 (4 MB)
`maxBufferSizeRecords`	Die maximale Anzahl von Datensätzen, für die der Agent Daten zwischenspeichert, bevor er sie an den Firehose-Stream sendet. Wertebereich: 1–500 Standard: 500

Konfigurationseinstellung	Beschreibung
`minTimeBetweenFilePollsMillis`	Das Zeitintervall (in Millisekunden), in dem der Agent die überwachten Dateien auf neue Daten abfragt und analysiert. Wertbereich: 1 oder höher Standard: 100
`multiLineStartPattern`	Das Muster für die Identifizierung des Datensatzbeginns. Ein Datensatz besteht aus einer Zeile, die mit dem angegebenen Muster übereinstimmt, und allen folgenden Zeilen, die nicht dem Muster entsprechen. Gültige Werte sind reguläre Ausdrücke. Standardmäßig wird jede neue Zeile in den Protokolldateien als einziger Datensatz analysiert.
`skipHeaderLines`	Die Anzahl der Zeilen, die der Agent überspringt, ehe mit der Analyse der überwachten Dateien begonnen wird. Wertbereich: 0 oder höher Standard: 0 (null)
`truncatedRecordTerminator`	Die Zeichenfolge, die der Agent verwendet, um einen analysierten Datensatz zu kürzen, wenn die Datensatzgröße die Datensatzgrößenbeschränkung von Amazon Data Firehose überschreitet. (1,000 KB) Standard: `'\n'` (Zeilenumbruch)

Konfigurieren Sie mehrere Dateiverzeichnisse und Streams

Wenn Sie mehrere Ablaufkonfigurationseinstellungen angeben, können Sie den Agenten so konfigurieren, dass er mehrere Dateiverzeichnisse überwacht und Daten an verschiedene Streams sendet. Im folgenden Konfigurationsbeispiel überwacht der Agent zwei Dateiverzeichnisse und sendet Daten an einen Kinesis-Datenstream bzw. einen Firehose-Stream. Sie können unterschiedliche Endpunkte für Kinesis Data Streams und Amazon Data Firehose angeben, sodass sich Ihr Datenstream und Ihr Firehose-Stream nicht in derselben Region befinden müssen.

```
{
```

```
    "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"
        }
    ]
}
```

Ausführlichere Informationen zur Verwendung des Agenten mit Amazon Kinesis Data Streams finden Sie unter Schreiben in Amazon Kinesis Data Streams mit Kinesis Agent.

Daten mit Agenten vorverarbeiten

Der Agent kann die aus den überwachten Dateien analysierten Datensätze vorverarbeiten, bevor er sie an Ihren Firehose-Stream sendet. Sie können dieses Feature aktivieren, indem Sie Ihrem Dateifluss die Konfigurationseinstellung `dataProcessingOptions` hinzufügen. Sie können eine oder mehrere Verarbeitungsoptionen hinzufügen. Diese werden in der angegebenen Reihenfolge ausgeführt.

Der Agent unterstützt die folgenden Verarbeitungsoptionen. Der Agent ist ein Open-Source-Tool, sodass Sie dessen Verarbeitungsoptionen optimieren und erweitern können. Sie können den Agenten von Kinesis Agent herunterladen.

Verarbeitungsoptionen

`SINGLELINE`

Konvertiert einen mehrzeiligen Datensatz in einen einzeiligen Datensatz, indem Zeilenumbruchzeichen sowie vorangestellte und folgende Leerzeichen entfernt werden.

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

`CSVTOJSON`

Konvertiert einen Datensatz aus dem durch Trennzeichen getrennten Format in einen Datensatz im JSON-Format.

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

`customFieldNames`

[Erforderlich] Die Feldnamen, die als Schlüssel in den einzelnen JSON-Schlüssel-Wert-Paaren verwendet werden. Wenn Sie beispielsweise `["f1", "f2"]` angeben, wird der Datensatz „v1, v2“ in `{"f1":"v1","f2":"v2"}` konvertiert.

`delimiter`

Die Zeichenfolge, die als Trennzeichen im Datensatz verwendet wird. Standardmäßig wird ein Komma (,) verwendet.

`LOGTOJSON`

Konvertiert einen Datensatz aus einem Protokollformat in einen Datensatz im JSON-Format. Folgende Protokollformate werden unterstützt: Apache Common Log, Apache Combined Log, Apache Error Log und RFC3164 Syslog.

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

`logFormat`

[Erforderlich] Das Format des Protokolleintrags. Folgende Werte sind möglich:

- `COMMONAPACHELOG` – Das Apache-Common-Log-Format. Jeder Protokolleintrag weist standardmäßig das folgende Muster auf: „`%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes}`“.

- COMBINEDAPACHELOG – Das Apache-Combined-Log-Format. Jeder Protokolleintrag weist standardmäßig das folgende Muster auf: „%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes} %{referrer} %{agent}“.
- APACHEERRORLOG – Das Apache-Error-Log-Format. Jeder Protokolleintrag weist standardmäßig das folgende Muster auf: „[%{timestamp}] [%{module}: %{severity}] [pid %{processid}:tid %{threadid}] [client: %{client}] %{message}“.
- SYSLOG— Das RFC3164 Syslog-Format. Jeder Protokolleintrag weist standardmäßig das folgende Muster auf: „%{timestamp} %{hostname} %{program}[%{processid}]: %{message}“.

matchPattern

Überschreibt das Standardmuster für das angegebene Protokollformat. Verwenden Sie diese Einstellung, um Werte aus Protokolleinträgen zu extrahieren, wenn sie ein benutzerdefiniertes Format verwenden. Wenn Sie matchPattern angeben, müssen Sie auch customFieldNames angeben.

customFieldNames

Die benutzerdefinierten Feldnamen, die als Schlüssel in den einzelnen JSON-Schlüssel-Wert-Paaren verwendet werden. Mit dieser Einstellung können Sie Feldnamen für Werte definieren, die aus matchPattern extrahiert wurden, oder die Standardfeldnamen von vordefinierten Protokollformaten überschreiben.

Example : LOGTOJSON-Konfiguration

Nachfolgend ein Beispiel einer LOGTOJSON-Konfiguration für einen Apache Common Log-Eintrag, der in ein JSON-Format konvertiert wurde:

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

Vor der Konvertierung:

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

Nach der Konvertierung:

```
{"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 : LOGTOJSON-Konfiguration mit benutzerdefinierten Feldern

Im Folgenden ein weiteres Beispiel einer LOGTOJSON-Konfiguration:

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

Durch diese Konfigurationseinstellung wird der Apache Common Log-Eintrag aus dem vorherigen Beispiel wie folgt in ein JSON-Format konvertiert:

```
{"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 : Konvertieren eines Apache Common Log-Eintrags

Bei der folgenden Ablaufkonfiguration wird ein Apache Common Log-Eintrag in einen einzeiligen Datensatz im JSON-Format umgewandelt:

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

```
        }
    ]
}
```

Example : Konvertieren mehrzeiliger Datensätze

Bei der folgenden Ablaufkonfiguration werden mehrzeilige Datensätze analysiert, deren erste Zeile mit „`[SEQUENCE=`“ beginnt. Jeder Datensatz wird in einen einzeiligen Datensatz konvertiert. Anschließend werden Werte aus dem Datensatz basierend auf einem Tabulatortrennzeichen extrahiert. Die extrahierten Werte werden zu angegebenen `customFieldNames`-Werten zugeordnet und ergeben so einen einzeiligen Datensatz im JSON-Format.

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

Example : LOGTOJSON-Konfiguration mit Übereinstimmungsmuster

Nachfolgend ein Beispiel einer `LOGTOJSON`-Konfiguration für einen Apache Common Log-Eintrag, der in das JSON-Format konvertiert wurde. Das letzte Feld (Bytes) wurde ausgelassen:

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

```
    "customFieldNames": ["host", "ident", "authuser", "datetime", "request",
 "response"]
}
```

Vor der Konvertierung:

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

Nach der Konvertierung:

```
{"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"}
```

Verwenden Sie allgemeine Agent-CLI-Befehle

Die folgende Tabelle enthält eine Reihe gängiger Anwendungsfälle und die entsprechenden Befehle für die Arbeit mit dem AWS Kinesis-Agenten.

Anwendungsfall	Befehl
Startet den Agenten beim Systemstart automatisch	`sudo chkconfig aws-kinesis-agent on`
Überprüfen Sie den Status des Agenten	`sudo service aws-kinesis-agent status`
Stoppen Sie den Agenten	`sudo service aws-kinesis-agent stop`
Lesen Sie die Protokolldatei des Agenten von diesem Speicherort	`/var/log/aws-kinesis-agent/aws-kinesis-agent.log`
Deinstallieren Sie den Agenten	`sudo yum remove aws-kinesis-agent`

Probleme beim Senden über Kinesis Agent beheben

Diese Tabelle enthält Informationen zur Fehlerbehebung und Lösungen für häufig auftretende Probleme bei der Verwendung des Amazon Kinesis Agent.

Problem	Lösung
Warum funktioniert Kinesis Agent nicht unter Windows?	Kinesis Agent für Windows ist eine andere Software als Kinesis Agent für Linux-Plattformen.
Warum verlangsamt sich Kinesis Agent und/oder `RecordSendErrors` nimmt zu?	Dies ist normalerweise auf die Drosselung durch Kinesis zurückzuführen. Überprüfen Sie die `WriteProvisionedThroughputExceeded` Metrik für Kinesis Data Streams oder die `ThrottledRecords` Metrik für Firehose-Streams. Jede Erhöhung dieser Metriken von 0 zeigt an, dass die Stream-Grenzwerte erhöht werden müssen. Weitere Informationen finden Sie unter Kinesis Data Stream-Grenzwerte und Firehose-Streams. Sobald Sie die Drosselung ausgeschlossen haben, überprüfen Sie, ob der Kinesis Agent so konfiguriert ist, dass er eine große Menge kleiner Dateien durchsucht. Es gibt eine Verzögerung, wenn der Kinesis Agent eine neue Datei überwacht, daher sollte der Kinesis-Agent eine kleine Menge größerer Dateien überwachen. Versuchen Sie, Ihre Protokolldateien in größeren Dateien zusammenzufassen.
Wie behebt man die Ausnahmen? `java.lang.OutOfMemoryError`	Dies passiert, wenn der Kinesis Agent nicht über genügend Arbeitsspeicher verfügt, um seine aktuelle Arbeitslast zu bewältigen. Versuchen Sie, `JAVA_START_HEAP` und `JAVA_MAX_HEAP` in `/usr/bin/start-aws-kinesis-agent` zu erhöhen und den Agenten neu zu starten.
Wie behebt man die Ausnahmen? `IllegalStateException : connection pool shut down`	Kinesis Agent verfügt nicht über genügend Verbindungen, um seinen aktuellen Workload zu bewältigen. Versuchen Sie, `maxConnections` und `maxSendingThreads`

Problem	Lösung
	in den allgemeinen Konfigurationseinstellungen des Agenten unter `/etc/aws-kinesis/agent.json` zu erhöhen. Der Standardwert für diese Felder ist das 12-fache der verfügbaren Laufzeitprozessoren. Weitere Informationen zu den Einstellungen für erweiterte Agentenkonfigurationen finden Sie unter AgentConfiguration.java.
Wie kann ich ein anderes Problem mit Kinesis Agent beheben?	DEBUG-Level-Protokolle können in `/etc/aws-kinesis/log4j.xml` aktiviert werden.
Wie sollte ich Kinesis Agent konfigurieren?	Je kleiner das `maxBufferSizeBytes` , desto häufiger sendet der Kinesis Agent Daten. Dies kann nützlich sein, da es die Lieferzeit von Datensätzen verkürzt, aber es erhöht auch die Anfragen pro Sekunde an Kinesis.
Warum sendet Kinesis Agent doppelte Datensätze?	Dies ist auf eine Fehlkonfiguration bei der Dateiüberwachung zurückzuführen. Stellen Sie sicher, dass jedes `fileFlow's filePattern` nur einer Datei entspricht. Dies kann auch auftreten, wenn der verwendete `logrotate` -Modus im `copytruncate` -Modus ist. Versuchen Sie, den Modus auf den Standard- oder Erstellungsmodus zu ändern, um Duplikate zu vermeiden. Weitere Informationen zum Umgang mit doppelten Datensätzen finden Sie unter Umgang mit doppelten Datensätzen.

Daten mit AWS SDK senden

Sie können die Amazon Data Firehose-API verwenden, um Daten mit dem AWS SDK for Java, .NET, Node.js, Python oder Ruby an einen Firehose-Stream zu senden. Wenn Sie Amazon Data Firehose noch nicht kennen, nehmen Sie sich etwas Zeit, um sich mit den Konzepten und der Terminologie in Was ist Amazon Data Firehose? vertraut zu machen. Weitere Informationen finden Sie unter Entwickeln Sie Ihre Apps mit Amazon Web Services.

Diese Beispiele stellen keinen produktionsbereiten Code dar, d. h. es werden nicht alle möglichen Ausnahmen geprüft und es werden nicht alle möglichen Sicherheits- oder Leistungsüberlegungen berücksichtigt.

Die Amazon Data Firehose-API bietet zwei Operationen zum Senden von Daten an Ihren Firehose-Stream: PutRecordund. PutRecordBatch `PutRecord()`sendet einen Datensatz innerhalb eines Anrufs und `PutRecordBatch()` kann mehrere Datensätze innerhalb eines Anrufs senden.

Einzelne Schreiboperationen mit PutRecord

Für das Einfügen von Daten sind nur der Firehose-Streamname und ein Bytepuffer (<=1000 KB) erforderlich. Da Amazon Data Firehose mehrere Datensätze stapelt, bevor die Datei in Amazon S3 geladen wird, möchten Sie möglicherweise ein Datensatztrennzeichen hinzufügen. Verwenden Sie den folgenden Code, um Daten datensatzweise in einen Firehose-Stream einzufügen:

```
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);
```

Weitere Informationen zum Codekontext finden Sie im Beispielcode, der im AWS SDK enthalten ist. Informationen zur Anfrage- und Antwortsyntax finden Sie im entsprechenden Thema unter Firehose API Operations.

Batch-Schreiboperationen mit PutRecordBatch

Für das Einfügen von Daten sind nur der Firehose-Stream-Name und eine Liste von Datensätzen erforderlich. Da Amazon Data Firehose mehrere Datensätze stapelt, bevor die Datei in Amazon S3 geladen wird, möchten Sie möglicherweise ein Datensatztrennzeichen hinzufügen. Verwenden Sie den folgenden Code, um Datensätze stapelweise in einen Firehose-Stream einzufügen:

```
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();
```

Weitere Informationen zum Codekontext finden Sie im Beispielcode, der AWS im SDK enthalten ist. Informationen zur Anfrage- und Antwortsyntax finden Sie im entsprechenden Thema unter Firehose API Operations.

CloudWatch Logs an Firehose senden

CloudWatch Protokollereignisse können mithilfe von CloudWatch Abonnementfiltern an Firehose gesendet werden. Weitere Informationen finden Sie unter Abonnementfilter mit Amazon Data Firehose.

CloudWatch Protokollereignisse werden im komprimierten GZIP-Format an Firehose gesendet. Wenn Sie dekomprimierte Protokollereignisse an Firehose-Ziele senden möchten, können Sie die Dekomprimierungsfunktion in Firehose verwenden, um Logs automatisch zu dekomprimieren. CloudWatch

> **Important**
>
> Derzeit unterstützt Firehose die Übermittlung von CloudWatch Protokollen an das Amazon OpenSearch Service-Ziel nicht, da Amazon mehrere Protokollereignisse zu einem Firehose-Datensatz CloudWatch zusammenfasst und Amazon OpenSearch Service nicht mehrere Protokollereignisse in einem Datensatz akzeptieren kann. Als Alternative können Sie erwägen, den Abonnementfilter für Amazon OpenSearch Service in CloudWatch Logs zu verwenden.

Dekomprimieren Sie Protokolle CloudWatch

Wenn Sie Firehose zur Übertragung von CloudWatch Logs verwenden und dekomprimierte Daten an Ihr Firehose-Stream-Ziel liefern möchten, verwenden Sie Firehose Data Format Conversion (Parquet, ORC) oder Dynamic Partitioning. Sie müssen die Dekomprimierung für Ihren Firehose-Stream aktivieren.

Sie können die Dekomprimierung mit dem AWS-Managementkonsole, oder aktivieren. AWS Command Line Interface AWS SDKs

> **Note**
>
> Wenn Sie die Dekomprimierungsfunktion für einen Stream aktivieren, verwenden Sie diesen Stream ausschließlich für CloudWatch Logs-Abonnementfilter und nicht für Vending Logs. Wenn Sie die Dekomprimierungsfunktion für einen Stream aktivieren, der sowohl zum Ingestieren von Logs als auch von Vended CloudWatch Logs verwendet wird, schlägt das Ingestieren von Vended-Logs in Firehose fehl. Diese Dekomprimierungsfunktion ist nur für Logs verfügbar. CloudWatch

Extrahieren Sie die Nachricht nach der Dekomprimierung der Protokolle CloudWatch

Wenn Sie die Dekomprimierung aktivieren, haben Sie die Möglichkeit, auch die Nachrichtenextraktion zu aktivieren. Bei der Verwendung der Nachrichtenextraktion filtert Firehose alle Metadaten wie Besitzer, Loggroup, Logstream und andere aus den dekomprimierten CloudWatch Logs-Datensätzen heraus und liefert nur den Inhalt in den Nachrichtenfeldern. Wenn Sie Daten an ein Splunk-Ziel senden, müssen Sie die Nachrichtenextraktion aktivieren, damit Splunk die Daten analysieren kann. Im Folgenden finden Sie Beispielausgaben nach der Dekomprimierung mit und ohne Nachrichtenextraktion.

Abb. 1: Beispielausgabe nach der Dekomprimierung ohne Nachrichtenextraktion:


```
{
 "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\"}"
 }
 ]
}
```

Abb. 2: Beispielausgabe nach der Dekomprimierung mit Nachrichtenextraktion:

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

Aktivieren Sie die Dekomprimierung für einen neuen Firehose-Stream von der Konsole

Um die Dekomprimierung in einem neuen Firehose-Stream zu aktivieren, verwenden Sie den AWS-Managementkonsole

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Navigationsbereich Amazon Data Firehose aus.
3. Wählen Sie Create Firehose stream (Firehose-Stream erstellen) aus.
4. Unter Quelle und Ziel auswählen

   Quelle

   Die Quelle deines Firehose-Streams. Wählen Sie eine der folgenden Quellen:

   - Direct PUT — Wählen Sie diese Option, um einen Firehose zu erstellen, in den Producer-Anwendungen direkt schreiben. Eine Liste der AWS Dienste und Agenten sowie Open-Source-Dienste, die in Direct PUT in Firehose integriert sind, finden Sie in diesem Abschnitt.

- Kinesis-Stream: Wählen Sie diese Option, um einen Firehose-Stream zu konfigurieren, der einen Kinesis-Datenstream als Datenquelle verwendet. Anschließend können Sie Firehose verwenden, um Daten einfach aus einem vorhandenen Kinesis-Datenstrom zu lesen und in Ziele zu laden. Weitere Informationen finden Sie unter Mit Kinesis Data Streams in Firehose schreiben

Zieladresse

Das Ziel Ihres Firehose-Streams. Wählen Sie eine der folgenden Optionen aus:

- Amazon S3
- Splunk

5. Geben Sie Firehose Firehose-Streamname einen Namen für Ihren Stream ein.
6. (Optional) Unter Datensätze transformieren:
   - Wählen Sie im Abschnitt Quelldatensätze aus Amazon CloudWatch Logs dekomprimieren die Option Dekomprimierung aktivieren aus.
   - Wenn Sie die Nachrichtenextraktion nach der Dekomprimierung verwenden möchten, wählen Sie Nachrichtenextraktion einschalten.

Aktivieren Sie die Dekomprimierung für einen vorhandenen Firehose-Stream

Dieser Abschnitt enthält Anweisungen zum Aktivieren der Dekomprimierung vorhandener Firehose-Streams. Es deckt zwei Szenarien ab: Streams mit deaktivierter Lambda-Verarbeitung und Streams mit bereits aktivierter Lambda-Verarbeitung. In den folgenden Abschnitten step-by-step werden die Verfahren für jeden Fall beschrieben, einschließlich der Erstellung oder Änderung von Lambda-Funktionen, der Aktualisierung der Firehose-Einstellungen und der Überwachung von CloudWatch Metriken, um eine erfolgreiche Implementierung der integrierten Firehose-Dekomprimierungsfunktion sicherzustellen.

Dekomprimierung aktivieren, wenn die Lambda-Verarbeitung deaktiviert ist

Um die Dekomprimierung für einen vorhandenen Firehose mit deaktivierter Lambda-Verarbeitung zu aktivieren, müssen Sie zuerst die Lambda-Verarbeitung aktivieren. Diese Bedingung gilt nur für bestehende Streams. Die folgenden Schritte zeigen, wie die Dekomprimierung für bestehende Streams aktiviert wird, für die die Lambda-Verarbeitung nicht aktiviert ist.

1. Erstellen Sie eine Lambda-Funktion. Sie können entweder einen Dummy-Record-Passthrough erstellen oder diesen Blueprint verwenden, um eine neue Lambda-Funktion zu erstellen.
2. Aktualisieren Sie Ihren aktuellen Firehose-Stream, um die Lambda-Verarbeitung zu aktivieren, und verwenden Sie die Lambda-Funktion, die Sie für die Verarbeitung erstellt haben.
3. Sobald Sie den Stream mit der neuen Lambda-Funktion aktualisiert haben, kehren Sie zur Firehose-Konsole zurück und aktivieren Sie die Dekomprimierung.
4. Deaktivieren Sie die Lambda-Verarbeitung, die Sie in Schritt 1 aktiviert haben. Sie können jetzt die Funktion löschen, die Sie in Schritt 1 erstellt haben.

Dekomprimierung aktivieren, wenn die Lambda-Verarbeitung aktiviert ist

Wenn Sie bereits über einen Firehose-Stream mit einer Lambda-Funktion verfügen, können Sie ihn zur Durchführung der Dekomprimierung durch die Firehose-Dekomprimierungsfunktion ersetzen. Bevor Sie fortfahren, überprüfen Sie Ihren Lambda-Funktionscode, um sicherzustellen, dass er nur Dekomprimierung oder Nachrichtenextraktion durchführt. Die Ausgabe Ihrer Lambda-Funktion sollte den Beispielen in Abb. 1 oder Abb. 2 ähneln. Wenn die Ausgabe ähnlich aussieht, können Sie die Lambda-Funktion mithilfe der folgenden Schritte ersetzen.

1. Ersetzen Sie Ihre aktuelle Lambda-Funktion durch diesen Blueprint. Die neue Blueprint-Lambda-Funktion erkennt automatisch, ob die eingehenden Daten komprimiert oder dekomprimiert sind. Sie führt nur dann eine Dekomprimierung durch, wenn ihre Eingabedaten komprimiert sind.
2. Schalten Sie die Dekomprimierung mit der integrierten Firehose für die Dekomprimierung ein.
3. Aktivieren Sie CloudWatch Metriken für Ihren Firehose-Stream, falls er noch nicht aktiviert ist. Überwachen Sie die Metrik `CloudWatchProcessorLambda_IncomingCompressedData` und warten Sie, bis sich diese Metrik auf Null ändert. Dadurch wird bestätigt, dass alle an Ihre Lambda-Funktion gesendeten Eingabedaten dekomprimiert sind und die Lambda-Funktion nicht mehr benötigt wird.
4. Entfernen Sie die Lambda-Datentransformation, da Sie sie nicht mehr benötigen, um Ihren Stream zu dekomprimieren.

Deaktivieren Sie die Dekomprimierung im Firehose-Stream

Um die Dekomprimierung eines Datenstroms zu deaktivieren, verwenden Sie AWS-Managementkonsole

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Kinesis-Konsole unter https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Navigationsbereich Amazon Data Firehose aus.
3. Wählen Sie den Firehose-Stream aus, den Sie bearbeiten möchten.
4. Wählen Sie auf der Seite mit den Firehose-Stream-Details die Registerkarte Konfiguration aus.
5. Wählen Sie im Abschnitt Datensätze transformieren und konvertieren die Option Bearbeiten.
6. Deaktivieren Sie unter Quelldatensätze aus Amazon CloudWatch Logs dekomprimieren die Option Dekomprimierung aktivieren und wählen Sie dann Änderungen speichern.

Fehlerbehebung bei der Dekomprimierung in Firehose

Die folgende Tabelle zeigt, wie Firehose mit Fehlern bei der Datendekomprimierung und -verarbeitung umgeht, einschließlich der Übermittlung von Datensätzen an einen Fehler-S3-Bucket, der Protokollierung von Fehlern und der Ausgabe von Metriken. Es erklärt auch die Fehlermeldung, die bei nicht autorisierten Dateneingabevorgängen zurückgegeben wird.

Problem	Lösung
Was passiert mit den Quelldaten im Falle eines Fehlers bei der Dekomprimierung?	Wenn Amazon Data Firehose den Datensatz nicht dekomprimieren kann, wird der Datensatz unverändert (im komprimierten Format) an den Fehler S3-Bucket geliefert, den Sie bei der Erstellung des Firehose-Streams angegeben haben. Zusammen mit dem Datensatz enthält das gelieferte Objekt auch den Fehlercode und die Fehlermeldung. Diese Objekte werden an ein S3-Bucket-Präfix mit dem Namen gesendet. `decompression-failed` Firehose verarbeitet nach einer fehlgeschlagenen Dekomprimierung eines Datensatzes weiterhin andere Datensätze.
Was passiert mit den Quelldaten im Falle eines Fehlers in der Verarbeitungspipeline nach erfolgreicher Dekomprimierung?	Wenn Amazon Data Firehose bei den Verarbeitungsschritten nach der Dekomprimierung Fehler macht, z. B. dynamische Partitionierung und Datenformatkonvertierung, wird der Datensatz in komprimiertem Format an den Fehler-S3-Bucket übermittelt, den Sie bei der

Problem	Lösung
	Erstellung des Firehose-Streams angegeben haben. Zusammen mit dem Datensatz enthält das gelieferte Objekt auch einen Fehlercode und eine Fehlermeldung.
Wie werden Sie im Falle eines Fehlers oder einer Ausnahme informiert?	Falls bei der Dekomprimierung ein Fehler oder eine Ausnahme auftritt und Sie CloudWatch Logs konfiguri eren, protokolliert Firehose Fehlermeldungen in CloudWatch Logs. Darüber hinaus sendet Firehose Metriken an CloudWatch Metriken, die Sie überwachen können. Sie können optional auch Alarme erstellen, die auf von Firehose ausgegebenen Metriken basieren.
Was passiert, wenn `put` Operation en nicht aus CloudWatch Logs stammen?	Wenn der Kunde `puts` nicht aus den CloudWatch Logs kommt, wird die folgende Fehlermeldung zurückgegeben: `Put to Firehose failed for AccountId: <accountI D>, FirehoseName: <firehosename> because the request is not originating from allowed source types.`
Welche Metriken gibt Firehose für die Dekomprimierungsfunktion aus?	Firehose gibt Metriken für die Dekomprimierung jedes Datensatzes aus. Sie sollten den Zeitraum (1 Minute), die Statistik (Summe) und den Datumsbereich auswählen, um die Anzahl der fehlgeschlagenen oder erfolgreichen oder `DecompressedRecords` fehlgeschlagenen oder erfolgreichen Daten zu ermitteln. `DecompressedBytes` Weitere Informationen finden Sie unter CloudWatch Protokolliert Dekomprimierungsmetriken.

CloudWatch Ereignisse an Firehose senden

Sie können Amazon so konfigurieren, dass Ereignisse CloudWatch an einen Firehose-Stream gesendet werden, indem Sie einer CloudWatch Event-Regel ein Ziel hinzufügen.

Um ein Ziel für eine CloudWatch Event-Regel zu erstellen, die Ereignisse an einen vorhandenen Firehose sendet

1. Melden Sie sich bei an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Regel erstellen aus.
3. Wählen Sie auf der Seite Schritt 1: Regel erstellen für Ziele die Option Ziel hinzufügen und dann Firehose-Stream aus.
4. Wählen Sie einen vorhandenen Firehose-Stream aus.

Weitere Informationen zum Erstellen von CloudWatch Event-Regeln finden Sie unter Erste Schritte mit Amazon CloudWatch Events.

So konfigurieren AWS IoT , dass Daten an Firehose gesendet werden

Sie können so konfigurieren AWS IoT , dass Informationen an einen Firehose-Stream gesendet werden, indem Sie eine Aktion hinzufügen.

Um eine Aktion zu erstellen, die Ereignisse an einen vorhandenen Firehose sendet

1. Wenn Sie eine Regel in der AWS IoT Konsole erstellen, wählen Sie auf der Seite Regel erstellen unter Eine oder mehrere Aktionen festlegen die Option Aktion hinzufügen aus.
2. Wählen Sie Send messages to an Amazon Kinesis Firehose stream (Senden von Nachrichten an einen Amazon-Kinesis-Firehose-Stream).
3. Wählen Sie Configure action.
4. Wählen Sie als Streamname einen vorhandenen Firehose aus.
5. Wählen Sie für Separator ein Trennzeichen, das zwischen die Datensätze gesetzt werden soll.
6. Wählen Sie für IAM role name (IAM-Rollenname) eine vorhandene IAM;-Rolle oder wählen Sie Create a new role (Eine neue Rolle erstellen).
7. Wählen Sie Aktion hinzufügen aus.

Weitere Informationen zum Erstellen von AWS IoT Regeln finden Sie unter Tutorials zu AWS IoT-Regeln.

Transformieren Sie Quelldaten in Amazon Data Firehose

Amazon Data Firehose kann Ihre Lambda-Funktion aufrufen, um eingehende Quelldaten zu transformieren und die transformierten Daten an Ziele weiterzuleiten. Sie können die Amazon Data Firehose-Datentransformation aktivieren, wenn Sie Ihren Firehose-Stream erstellen.

Verstehen Sie den Ablauf der Datentransformation

Wenn Sie die Firehose-Datentransformation aktivieren, puffert Firehose eingehende Daten. Der Hinweis zur Puffergröße liegt zwischen 0,2 MB und 3 MB. Der standardmäßige Hinweis zur Lambda-Puffergröße beträgt 1 MB für alle Ziele, außer Splunk und Snowflake. Für Splunk und Snowflake beträgt der Standard-Pufferhinweis 256 KB. Der Hinweis für das Lambda-Pufferintervall liegt zwischen 0 und 900 Sekunden. Der standardmäßige Hinweis für das Lambda-Pufferintervall beträgt für alle Ziele außer Snowflake sechzig Sekunden. Für Snowflake beträgt das Standardintervall für Pufferhinweise 30 Sekunden. Um die Puffergröße anzupassen, setzen Sie den ProcessingConfigurationParameter der CreateDeliveryStreamUpdateDestinationOR-API mit dem aufgerufenen und. ProcessorParameter`BufferSizeInMBsIntervalInSeconds` Firehose ruft dann die angegebene Lambda-Funktion synchron mit jedem gepufferten Batch im synchronen Aufrufmodus auf. AWS Lambda Die transformierten Daten werden von Lambda an Firehose gesendet. Firehose sendet es dann an das Ziel, wenn die angegebene Puffergröße oder das angegebene Pufferintervall erreicht ist, je nachdem, was zuerst eintritt.

> **Important**
>
> Der synchrone Lambda-Aufrufmodus hat ein Nutzlastgrößenlimit von 6 MB sowohl für die Anforderung als auch für die Antwort. Die Puffergröße für das Senden der Anforderung an die Funktion muss kleiner oder gleich 6 MB sein. Außerdem darf die von der Funktion zurückgegebene Antwort 6 MB nicht übersteigen.

Dauer des Lambda-Aufrufs

Amazon Data Firehose unterstützt eine Lambda-Aufrufzeit von bis zu 5 Minuten. Wenn die Ausführung Ihrer Lambda-Funktion länger als 5 Minuten dauert, wird die folgende Fehlermeldung angezeigt: Firehose hat beim Aufrufen von Lambda auf Timeout-Fehler gestoßen. AWS Das maximal unterstützte Funktions-Timeout beträgt 5 Minuten.

Informationen darüber, was Amazon Data Firehose tut, wenn ein solcher Fehler auftritt, finden Sie unterthe section called “Behandeln Sie Fehler bei der Datentransformation ”.

Erforderliche Parameter für die Datentransformation

Alle transformierten Datensätze von Lambda müssen die folgenden Parameter enthalten. Andernfalls lehnt Amazon Data Firehose sie ab und behandelt dies als Fehler bei der Datentransformation.

For Kinesis Data Streams and Direct PUT

Die folgenden Parameter sind für alle transformierten Datensätze von Lambda erforderlich.

- `recordId`— Die Datensatz-ID wird während des Aufrufs von Amazon Data Firehose an Lambda übergeben. Der transformierte Datensatz muss dieselbe Datensatz-ID enthalten. Jede fehlende Übereinstimmung zwischen der ID des ursprünglichen Datensatzes und der ID des transformierten Datensatzes wird als Datentransformationsfehler behandelt.
- `result`— Der Status der Datentransformation des Datensatzes. Die möglichen Werte sind `Ok` (der Datensatz wurde erfolgreich transformiert), `Dropped` (der Datensatz wurde absichtlich von Ihrer Verarbeitungslogik fallengelassen) und `ProcessingFailed` (der Datensatz konnte nicht transformiert werden). Wenn ein Datensatz den Status `Ok` oder hat`Dropped`, geht Amazon Data Firehose davon aus, dass er erfolgreich verarbeitet wurde. Andernfalls betrachtet Amazon Data Firehose die Verarbeitung als erfolglos.
- `data`— Die transformierte Datennutzlast nach der Base64-Kodierung.

  Im Folgenden finden Sie ein Beispiel für eine Lambda-Ergebnisausgabe:

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

For Amazon MSK

Die folgenden Parameter sind für alle transformierten Datensätze von Lambda erforderlich.

- `recordId`— Die Datensatz-ID wird während des Aufrufs von Firehose an Lambda übergeben. Der transformierte Datensatz muss dieselbe Datensatz-ID enthalten. Jede

fehlende Übereinstimmung zwischen der ID des ursprünglichen Datensatzes und der ID des transformierten Datensatzes wird als Datentransformationsfehler behandelt.

- `result`— Der Status der Datentransformation des Datensatzes. Die möglichen Werte sind `Ok` (der Datensatz wurde erfolgreich transformiert), `Dropped` (der Datensatz wurde absichtlich von Ihrer Verarbeitungslogik fallengelassen) und `ProcessingFailed` (der Datensatz konnte nicht transformiert werden). Wenn ein Datensatz den Status `Ok` oder hat, geht Firehose davon aus`Dropped`, dass er erfolgreich verarbeitet wurde. Andernfalls betrachtet Firehose es als erfolglos verarbeitet.
- `KafkaRecordValue`— Die transformierte Datennutzlast nach der Base64-Kodierung.

  Im Folgenden finden Sie ein Beispiel für eine Lambda-Ergebnisausgabe:

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

Unterstützte Lambda-Blueprints

Diese Blueprints zeigen, wie Sie AWS Lambda-Funktionen erstellen und verwenden können, um Daten in Ihren Amazon Data Firehose-Datenströmen zu transformieren.

Um die Blueprints zu sehen, die in der Konsole verfügbar sind AWS Lambda

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie Create function (Funktion erstellen) und anschließend die Option Use a blueprint (Eine Vorlage verwenden) aus.
3. Suchen Sie im Feld Blueprints nach dem Schlüsselwort, `firehose` um die Amazon Data Firehose Lambda-Blueprints zu finden.

Liste der Vorlagen:

- An den Amazon Data Firehose-Stream gesendete Datensätze verarbeiten (Node.js, Python)

Dieser Blueprint zeigt ein grundlegendes Beispiel für die Verarbeitung von Daten in Ihrem Firehose-Datenstrom mithilfe von AWS Lambda.

Datum der letzten Veröffentlichung: November 2016.

Versionshinweise: keine.

- An CloudWatch Firehose gesendete Prozessprotokolle

  Dieser Blueprint ist veraltet. Verwenden Sie diesen Blueprint nicht. Es können hohe Gebühren anfallen, wenn die dekomprimierten CloudWatch Logs-Daten mehr als 6 MB groß sind (Lambda-Limit). Informationen zur Verarbeitung von an Firehose gesendeten CloudWatch Logs finden Sie unter Writing to Firehose Using CloudWatch Logs.
- Amazon Data Firehose-Stream-Datensätze im Syslog-Format in JSON konvertieren (Node.js)

  Dieser Blueprint zeigt, wie Sie Eingabedatensätze im RFC3164 Syslog-Format in JSON konvertieren können.

  Datum der letzten Veröffentlichung: November 2016.

  Versionshinweise: keine.

Um die Blueprints zu sehen, die verfügbar sind in AWS Serverless Application Repository

1. Wechseln Sie zu AWS Serverless Application Repository.
2. Wählen Sie Alle Anwendungen durchsuchen aus.
3. Suchen Sie im Feld Applications (Anwendungen) nach dem Schlüsselwort `firehose`.

Sie können auch eine Lambda-Funktion erstellen, ohne eine Vorlage zu verwenden. Siehe Erste Schritte mit AWS Lambda.

Behandeln Sie Fehler bei der Datentransformation

Wenn Ihr Lambda-Funktionsaufruf aufgrund eines Netzwerk-Timeouts fehlschlägt oder weil Sie das Lambda-Aufruflimit erreicht haben, wiederholt Amazon Data Firehose den Aufruf standardmäßig dreimal. Wenn der Aufruf nicht erfolgreich ist, überspringt Amazon Data Firehose diesen Datensatzstapel. Die übersprungenen Datensätze werden als nicht erfolgreich verarbeitete Datensätze behandelt. Sie können die Wiederholungsoptionen mithilfe der API oder angeben oder

überschreiben. CreateDeliveryStreamUpdateDestination Für diese Art von Fehler können Sie Aufruffehler in Amazon CloudWatch Logs protokollieren. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch .

Wenn der Status der Datentransformation eines Datensatzes lautet`ProcessingFailed`, behandelt Amazon Data Firehose den Datensatz als nicht erfolgreich verarbeitet. Für diese Art von Fehler können Sie von Ihrer Lambda-Funktion aus CloudWatch Fehlerprotokolle an Amazon Logs senden. Weitere Informationen finden Sie unter Zugreifen auf Amazon CloudWatch Logs für AWS Lambda im AWS Lambda Entwicklerhandbuch.

Wenn eine Datentransformation fehlschlägt, werden die erfolglos verarbeiteten Datensätze im `processing-failed` Ordner an Ihren S3-Bucket übermittelt. Die Datensätze haben das folgende Format:

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

`attemptsMade`

Die Anzahl der versuchten Aufrufanforderungen.

`arrivalTimestamp`

Der Zeitpunkt, zu dem der Datensatz bei Amazon Data Firehose eingegangen ist.

`errorCode`

Der von Lambda ausgegebene HTTP-Fehlercode.

`errorMessage`

Die von Lambda ausgegebene HTTP-Fehlermeldung.

`attemptEndingTimestamp`

Der Zeitpunkt, zu dem Amazon Data Firehose aufgehört hat, Lambda-Aufrufe zu versuchen.

`rawData`

Die base64-verschlüsselten Daten.

`lambdaArn`

Der Amazon-Ressourcenname (ARN) der -Lambda-Funktion.

Quelldatensätze sichern

Amazon Data Firehose kann alle nicht transformierten Datensätze gleichzeitig in Ihrem S3-Bucket sichern und gleichzeitig transformierte Datensätze an das Ziel liefern. Sie können die Sicherung von Quelldatensätzen aktivieren, wenn Sie Ihren Firehose-Stream erstellen oder aktualisieren. Nach de Aktivierung kann die Sicherung der Quelldatensätze nicht mehr deaktiviert werden.

Streaming-Daten in Amazon Data Firehose partitionieren

Dynamische Partitionierung ermöglicht es Ihnen, Streaming-Daten in Firehose kontinuierlich zu partitionieren, indem Sie Schlüssel innerhalb von Daten verwenden (z. B. `customer_id` oder`transaction_id`) und dann die nach diesen Schlüsseln gruppierten Daten in den entsprechenden Amazon Simple Storage Service (Amazon S3) -Präfixen bereitstellen. Dies macht es einfacher, leistungsstarke und kosteneffiziente Analysen von Streaming-Daten in Amazon S3 mithilfe verschiedener Dienste wie Amazon Athena, Amazon EMR, Amazon Redshift Spectrum und Amazon durchzuführen. QuickSight Darüber hinaus kann AWS Glue anspruchsvollere Extraktions-, Transformations- und Ladeaufträge (ETL) ausführen, nachdem die dynamisch partitionierten Streaming-Daten an Amazon S3 geliefert wurden, in Anwendungsfällen, in denen zusätzliche Verarbeitung erforderlich ist.

Durch die Partitionierung Ihrer Daten wird die Menge der gescannten Daten minimiert, die Leistung optimiert und die Kosten Ihrer Analyseabfragen auf Amazon S3 gesenkt. Es verbessert auch den granulierten Zugriff auf Ihre Daten. Firehose-Streams werden traditionell verwendet, um Daten zu erfassen und in Amazon S3 zu laden. Um einen Streaming-Datensatz für Amazon-S3-basierte Analysen zu partitionieren, müssten Sie Partitionierungsanwendungen zwischen Amazon-S3-Buckets ausführen, bevor Sie die Daten für Analysen verfügbar machen können, was kompliziert oder kostspielig werden könnte.

Mit dynamischer Partitionierung gruppiert Firehose fortlaufend übertragene Daten mithilfe dynamisch oder statisch definierter Datenschlüssel und liefert die Daten nach Schlüsseln an einzelne Amazon S3 S3-Präfixe. Dies reduziert time-to-insight sich um Minuten oder Stunden. Es reduziert auch die Kosten und vereinfacht Architekturen.

Themen

Dynamische Partitionierung in Amazon Data Firehose aktivieren

Sie können die dynamische Partitionierung für Ihre Firehose-Streams über die Amazon Data Firehose Management Console, CLI oder die konfigurieren. APIs

> **Important**
>
> Sie können die dynamische Partitionierung nur aktivieren, wenn Sie einen neuen Firehose-Stream erstellen. Sie können die dynamische Partitionierung nicht für einen vorhandenen Firehose-Stream aktivieren, für den die dynamische Partitionierung noch nicht aktiviert ist.

Ausführliche Schritte zur Aktivierung und Konfiguration der dynamischen Partitionierung über die Firehose-Verwaltungskonsole bei der Erstellung eines neuen Firehose-Streams finden Sie unter Amazon Firehose-Stream erstellen. Wenn Sie das Ziel für Ihren Firehose-Stream angeben möchten, befolgen Sie unbedingt die Schritte im Zieleinstellungen konfigurieren Abschnitt, da die dynamische Partitionierung derzeit nur für Firehose-Streams unterstützt wird, die Amazon S3 als Ziel verwenden.

Sobald die dynamische Partitionierung auf einem aktiven Firehose-Stream aktiviert ist, können Sie die Konfiguration aktualisieren, indem Sie neue Partitionierungsschlüssel und die S3-Präfixausdrücke hinzufügen oder vorhandene entfernen oder aktualisieren. Nach der Aktualisierung verwendet Firehose die neuen Schlüssel und die neuen S3-Präfixausdrücke.

> **Important**
>
> Sobald Sie die dynamische Partitionierung in einem Firehose-Stream aktiviert haben, kann sie in diesem Firehose-Stream nicht mehr deaktiviert werden.

Verstehen Sie die Partitionierungsschlüssel

Mit dynamischem Partitionierungs-Mechanismus erstellen Sie gezielte Datensätze aus den Streaming-S3-Daten, indem sie auf der Grundlage von Partitionsschlüsseln partitioniert werden. Mit Partitionierungsschlüsseln können Sie Ihre Streaming-Daten anhand bestimmter Werte filtern. Wenn Sie Ihre Daten beispielsweise nach Kunden-ID und Land filtern müssen, können Sie das Datenfeld der `customer_id` als einen Partitionsschlüssel und das Datenfeld des `country` als einen weiteren Partitionsschlüssel angeben. Anschließend geben Sie die Ausdrücke (unter Verwendung

der unterstützten Formate) an, um die S3-Bucket-Präfixe zu definieren, an die die dynamisch partitionierten Datensätze geliefert werden sollen.

Sie können Partitionierungsschlüssel mit den folgenden Methoden erstellen.

- Inline-Parsing — Diese Methode verwendet den integrierten Unterstützungsmechanismus von Firehose, einen JQ-Parser, zum Extrahieren der Schlüssel für die Partitionierung aus Datensätzen im JSON-Format. Derzeit unterstützen wir nur die Version. `jq 1.6`
- AWS Lambda-Funktion — Diese Methode verwendet eine angegebene AWS Lambda Funktion, um die für die Partitionierung benötigten Datenfelder zu extrahieren und zurückzugeben.

> **Important**
>
> Wenn Sie die dynamische Partitionierung aktivieren, müssen Sie mindestens eine dieser Methoden konfigurieren, um Ihre Daten zu partitionieren. Sie können eine dieser Methoden konfigurieren, um Ihre Partitionierungsschlüssel anzugeben, oder beide gleichzeitig.

Erstellen Sie Partitionierungsschlüssel mit Inline-Parsing

Um Inline-Parsing als dynamische Partitionierungsmethode für Ihre Streaming-Daten zu konfigurieren, müssen Sie Datensatzparameter auswählen, die als Partitionierungsschlüssel verwendet werden sollen, und für jeden angegebenen Partitionierungsschlüssel einen Wert angeben.

Der folgende Beispieldatensatz zeigt, wie Sie mit Inline-Parsing Partitionierungsschlüssel dafür definieren können. Beachten Sie, dass die Daten im Base64-Format codiert sein sollten. Sie können sich auch auf das CLI-Beispiel beziehen.

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

Sie können beispielsweise wählen, ob Sie Ihre Daten auf der Grundlage des `customer_id`-Parameters oder des `event_timestamp`-Parameters partitionieren möchten. Das bedeutet, dass Sie möchten, dass der Wert des `customer_id`-Parameters oder des `event_timestamp`-Parameters in jedem Datensatz verwendet wird, um das S3-Präfix zu bestimmen, an das der Datensatz gesendet werden soll. Sie können auch einen verschachtelten Parameter wählen, z. B. `device` bei einem `.type.device`-Ausdruck. Ihre dynamische Partitionierungslogik kann von mehreren Parametern abhängen.

Nachdem Sie Datenparameter für Ihre Partitionierungsschlüssel ausgewählt haben, ordnen Sie jeden Parameter einem gültigen JQ-Ausdruck zu. Die folgende Tabelle zeigt eine solche Zuordnung von Parametern zu JQ-Ausdrücken:

Parameter	jq-Ausdruck
`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")

Zur Laufzeit verwendet Firehose die rechte Spalte oben, um die Parameter auf der Grundlage der Daten in den einzelnen Datensätzen auszuwerten.

Erstellen Sie Partitionierungsschlüssel mit einer Funktion AWS Lambda

Für komprimierte oder verschlüsselte Datensätze oder Daten, die in einem anderen Dateiformat als JSON vorliegen, können Sie die integrierte AWS Lambda Funktion mit Ihrem eigenen benutzerdefinierten Code verwenden, um die Datensätze zu dekomprimieren, zu entschlüsseln oder zu transformieren, um die für die Partitionierung benötigten Datenfelder zu extrahieren und zurückzugeben. Dies ist eine Erweiterung der bestehenden Transform-Lambda-Funktion, die heute mit Firehose verfügbar ist. Sie können die Datenfelder transformieren, analysieren und zurückgeben, die Sie dann mit derselben Lambda-Funktion für die dynamische Partitionierung verwenden können.

Im Folgenden finden Sie ein Beispiel für eine Lambda-Funktion zur Firehose-Stream-Verarbeitung in Python, die jeden gelesenen Datensatz von der Eingabe bis zur Ausgabe wiedergibt und Partitionierungsschlüssel aus den Datensätzen extrahiert.

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

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

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

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

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

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

        # Create output Firehose record and add modified payload and record ID to it.
```

```
        firehose_record_output = {'recordId': firehose_record_input['recordId'],
                                  'data': firehose_record_input['data'],
                                  'result': 'Ok',
                                  'metadata': { 'partitionKeys': partition_keys }}

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

        firehose_records_output['records'].append(firehose_record_output)

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

Im Folgenden finden Sie ein Beispiel für eine Lambda-Funktion zur Firehose-Stream-Verarbeitung in Go, die jeden gelesenen Datensatz von der Eingabe bis zur Ausgabe wiedergibt und Partitionierungsschlüssel aus den Datensätzen extrahiert.

```
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)
}
```

Verwenden Sie das Amazon S3 S3-Bucket-Präfix, um Daten zu liefern

Wenn Sie einen Firehose-Stream erstellen, der Amazon S3 als Ziel verwendet, müssen Sie einen Amazon S3 S3-Bucket angeben, in den Firehose Ihre Daten liefern soll. Amazon-S3-Bucket kann Präfixe verwenden, um die Daten zu organisieren, die Sie in Ihren S3-Buckets speichern. Ein

Amazon-S3-Bucket-Präfix ähnelt einem Verzeichnis, mit dem Sie ähnliche Objekte gruppieren können.

Bei der dynamischen Partitionierung werden Ihre partitionierten Daten in die angegebenen Amazon-S3-Präfixe übertragen. Wenn Sie die dynamische Partitionierung nicht aktivieren, ist die Angabe eines S3-Bucket-Präfix für Ihren Firehose-Stream optional. Wenn Sie sich jedoch dafür entscheiden, die dynamische Partitionierung zu aktivieren, müssen Sie die S3-Bucket-Präfixe angeben, an die Firehose partitionierte Daten liefert.

In jedem Firehose-Stream, in dem Sie die dynamische Partitionierung aktivieren, besteht der S3-Bucket-Präfixwert aus Ausdrücken, die auf den angegebenen Partitionierungsschlüsseln für diesen Firehose-Stream basieren. Wenn Sie das obige Datensatzbeispiel erneut verwenden, können Sie den folgenden S3-Präfixwert erstellen, der aus Ausdrücken besteht, die auf den oben definierten Partitionierungsschlüsseln basieren:

```
"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 wertet den obigen Ausdruck zur Laufzeit aus. Es gruppiert Datensätze, die demselben ausgewerteten S3-Präfixausdruck entsprechen, zu einem einzigen Datensatz. Firehose liefert dann jeden Datensatz an das ausgewertete S3-Präfix. Die Häufigkeit der Übermittlung von Datensätzen an S3 wird durch die Firehose-Stream-Puffereinstellung bestimmt. Daher wird der Datensatz in diesem Beispiel an den folgenden S3-Objektschlüssel übermittelt:

```
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
```

Für die dynamische Partitionierung müssen Sie das folgende Ausdrucksformat in Ihrem S3-Bucket-Präfix verwenden: `!{namespace:value}`, wobei Namespace entweder

`partitionKeyFromQuery`, `partitionKeyFromLambda` oder beides sein kann. Wenn Sie Inline-Parsing verwenden, um die Partitionierungsschlüssel für Ihre Quelldaten zu erstellen, müssen Sie einen S3-Bucket-Präfixwert angeben, der aus Ausdrücken besteht, die im folgenden Format angegeben sind: `"partitionKeyFromQuery:keyID"`. Wenn Sie AWS -Lambda-Funktion verwenden, um die Partitionierungsschlüssel für Ihre Quelldaten zu erstellen, müssen Sie einen S3-Bucket-Präfixwert angeben, der aus Ausdrücken besteht, die im folgenden Format angegeben sind: `"partitionKeyFromLambda:keyID"`.

> **Note**
>
> Sie können den S3-Bucket-Präfixwert auch im Hive-Format angeben, zum Beispiel customer_id=! {query:Customer_ID}. partitionKeyFrom

Weitere Informationen finden Sie unter „Wählen Sie Amazon S3 für Ihr Ziel“ unter Erstellen eines Amazon Firehose-Streams und benutzerdefinierte Präfixe für Amazon S3 S3-Objekte.

Fügen Sie bei der Übertragung von Daten an Amazon S3 ein neues Zeilentrennzeichen hinzu

Sie können New Line Delimiter aktivieren, um ein neues Zeilentrennzeichen zwischen Datensätzen in Objekten hinzuzufügen, die an Amazon S3 geliefert werden. Dies kann hilfreich sein, um Objekte in Amazon S3 zu analysieren. Dies ist auch besonders nützlich, wenn dynamische Partitionierung auf aggregierte Daten angewendet wird, da die Deaggregation mehrerer Datensätze (die auf aggregierte Daten angewendet werden muss, bevor sie dynamisch partitioniert werden können) im Rahmen des Analyseprozesses neue Zeilen aus Datensätzen entfernt.

Wenden Sie dynamische Partitionierung auf aggregierte Daten an

Sie können dynamische Partitionierung auf aggregierte Daten anwenden (z. B. mehrere Ereignisse, Protokolle oder Datensätze, die zu einem einzigen `PutRecord`- und `PutRecordBatch`-API-Aufruf zusammengefasst wurden), aber diese Daten müssen zuerst deaggregiert werden. Sie können Ihre Daten deaggregieren, indem Sie die Deaggregation mehrerer Datensätze aktivieren. Dabei werden die Datensätze im Firehose-Stream analysiert und getrennt.

Die Deaggregation mehrerer Datensätze kann entweder `JSON` vom Typ Typ sein, was bedeutet, dass die Trennung von Datensätzen auf aufeinanderfolgenden JSON-Objekten basiert. Die

Deaggregation kann auch vom Typ sein`Delimited`, was bedeutet, dass die Trennung von Datensätzen auf der Grundlage eines angegebenen benutzerdefinierten Trennzeichens erfolgt. Bei diesem benutzerdefinierten Trennzeichen muss es sich um eine Base-64-kodierte Zeichenfolge handeln. Wenn Sie beispielsweise die folgende Zeichenfolge als benutzerdefiniertes Trennzeichen verwenden möchten`####`, müssen Sie sie im Base-64-kodierten Format angeben, was sie übersetzt. `IyMjIw==` Die Deaggregation von Datensätzen nach JSON oder nach Trennzeichen ist auf 500 pro Datensatz begrenzt.

> **Note**
>
> Achten Sie beim Deaggregieren von JSON-Datensätzen darauf, dass Ihre Eingabe weiterhin im unterstützten JSON-Format dargestellt wird. JSON-Objekte dürfen sich in einer einzigen Zeile ohne Trennzeichen oder nur mit Zeilenumbruch (JSONL) befinden. Ein Array von JSON-Objekten ist keine gültige Eingabe.
> Dies sind Beispiele für korrekte Eingaben: `{"a":1}{"a":2}` and `{"a":1}\n{"a":2}`
> Dies ist ein Beispiel für die falsche Eingabe: `[{"a":1}, {"a":2}]`

Wenn Sie bei aggregierten Daten die dynamische Partitionierung aktivieren, analysiert Firehose die Datensätze und sucht in jedem API-Aufruf entweder nach gültigen JSON-Objekten oder nach getrennten Datensätzen, basierend auf dem angegebenen Deaggregationstyp für mehrere Datensätze.

> **Important**
>
> Wenn Ihre Daten aggregiert sind, kann die dynamische Partitionierung nur angewendet werden, wenn Ihre Daten zuerst deaggregiert wurden.

> **Important**
>
> Wenn Sie die Datentransformationsfunktion in Firehose verwenden, wird die Deaggregation vor der Datentransformation angewendet. Daten, die in Firehose eingehen, werden in der folgenden Reihenfolge verarbeitet: Deaggregation → Datentransformation via Lambda → Partitioning Keys.

Beheben Sie Fehler bei der dynamischen Partitionierung

Wenn Amazon Data Firehose nicht in der Lage ist, Datensätze in Ihrem Firehose-Stream zu analysieren oder die angegebenen Partitionierungsschlüssel nicht zu extrahieren oder die im S3-Präfixwert enthaltenen Ausdrücke auszuwerten, werden diese Datensätze an das S3-Fehler-Bucket-Präfix übermittelt, das Sie angeben müssen, wenn Sie den Firehose-Stream erstellen, in dem Sie die dynamische Partitionierung aktivieren. Das Präfix S3-Fehler-Bucket enthält alle Datensätze, die Firehose nicht an das angegebene S3-Ziel liefern kann. Diese Datensätze sind nach dem Fehlertyp geordnet. Neben dem Datensatz enthält das gelieferte Objekt auch Informationen über den Fehler, um den Fehler besser zu verstehen und zu beheben.

Sie müssen ein S3-Fehler-Bucket-Präfix für einen Firehose-Stream angeben, wenn Sie die dynamische Partitionierung für diesen Firehose-Stream aktivieren möchten. Wenn Sie die dynamische Partitionierung für einen Firehose-Stream nicht aktivieren möchten, ist die Angabe eines S3-Fehler-Bucket-Präfixes optional.

Pufferdaten für die dynamische Partitionierung

Amazon Data Firehose puffert eingehende Streaming-Daten bis zu einer bestimmten Größe und für einen bestimmten Zeitraum, bevor sie an die angegebenen Ziele gesendet werden. Sie können die Puffergröße und das Pufferintervall beim Erstellen neuer Firehose konfigurieren oder die Puffergröße und das Pufferintervall für Ihre vorhandenen Firehose-Streams aktualisieren. Eine Puffergröße wird in Sekunden gemessen MBs und ein Pufferintervall wird in Sekunden gemessen.

> **Note**
>
> Die Funktion „Nullpuffer“ ist für die dynamische Partitionierung nicht verfügbar.

Wenn die dynamische Partitionierung aktiviert ist, puffert Firehose intern Datensätze, die zu einer bestimmten Partition gehören, basierend auf dem konfigurierten Pufferhinweis (Größe und Zeit), bevor diese Datensätze an Ihren Amazon S3-Bucket gesendet werden. Um Objekte mit maximaler Größe zu liefern, verwendet Firehose intern eine mehrstufige Pufferung. Daher kann die end-to-end Verzögerung eines Batches von Datensätzen das 1,5-fache der konfigurierten Pufferhinweiszeit betragen. Dies wirkt sich auf die Datenaktualität eines Firehose-Streams aus.

Die Anzahl der aktiven Partitionen ist die Gesamtzahl der aktiven Partitionen innerhalb des Bereitstellungspuffers. Wenn die dynamische Partitionierungsabfrage beispielsweise 3 Partitionen

pro Sekunde erstellt und Sie eine Konfiguration mit Pufferhinweisen haben, die alle 60 Sekunden eine Übermittlung auslöst, dann haben Sie im Durchschnitt 180 aktive Partitionen. Wenn Firehose die Daten in einer Partition nicht an ein Ziel liefern kann, wird diese Partition im Lieferpuffer als aktiv gezählt, bis sie zugestellt werden kann.

Eine neue Partition wird erstellt, wenn ein S3-Präfix auf der Grundlage der Datensatzdatenfelder und der S3-Präfixausdrücke zu einem neuen Wert ausgewertet wird. Für jede aktive Partition wird ein neuer Puffer erstellt. Jeder nachfolgende Datensatz mit demselben ausgewerteten S3-Präfix wird an diesen Puffer geliefert.

Sobald der Puffer die Puffergrößenbeschränkung oder das Pufferzeitintervall erreicht, erstellt Firehose ein Objekt mit den Pufferdaten und liefert es an das angegebene Amazon S3 S3-Präfix. Nachdem das Objekt geliefert wurde, werden der Puffer für diese Partition und die Partition selbst gelöscht und aus der Anzahl der aktiven Partitionen entfernt.

Firehose liefert alle Pufferdaten als einzelnes Objekt, sobald die Puffergröße oder das Intervall für jede Partition separat erreicht sind. Sobald die Anzahl der aktiven Partitionen ein Limit von 500 pro Firehose-Stream erreicht, werden die restlichen Datensätze im Firehose-Stream an das angegebene S3-Fehler-Bucket-Präfix () activePartitionExceeded übermittelt. Sie können das Formular Amazon Data Firehose Limits verwenden, um eine Erhöhung dieses Kontingents auf bis zu 5000 aktive Partitionen pro gegebenem Firehose-Stream zu beantragen. Wenn Sie mehr Partitionen benötigen, können Sie mehr Firehose-Streams erstellen und die aktiven Partitionen auf diese verteilen.

Konvertieren Sie das Eingabedatenformat in Amazon Data Firehose

Amazon Data Firehose kann das Format Ihrer Eingabedaten von JSON in Apache Parquet oder Apache ORC konvertieren, bevor die Daten in Amazon S3 gespeichert werden. Parquet und ORC sind spaltenbasierte Datenformate, die Speicherplatz sparen und schnellere Abfragen im Vergleich zu zeilenorientierten Formaten wie JSON unterstützen. Wenn Sie ein anderes Eingabeformat als JSON konvertieren möchten, z. B. kommagetrennte Werte (CSV) oder strukturierten Text, können Sie es zunächst in JSON AWS Lambda umwandeln. Weitere Informationen finden Sie unter Transformieren Sie Quelldaten.

Sie können das Format Ihrer Daten konvertieren, auch wenn Sie Ihre Datensätze aggregieren, bevor Sie sie an Amazon Data Firehose senden.

Amazon Data Firehose benötigt die folgenden drei Elemente, um das Format Ihrer Datensatzdaten zu konvertieren:

Deserializer

Amazon Data Firehose benötigt einen Deserializer, um das JSON Ihrer Eingabedaten zu lesen. Sie können einen der folgenden zwei Deserializer-Typen wählen.

Wenn Sie mehrere JSON-Dokumente zu demselben Datensatz kombinieren, stellen Sie sicher, dass Ihre Eingabe weiterhin im unterstützten JSON-Format dargestellt wird. Ein Array von JSON-Dokumenten ist keine gültige Eingabe.

Dies ist beispielsweise die richtige Eingabe: `{"a": 1}{"b": 1}` und dies ist die falsche Eingabe:`[{"a":1}, {"a":2}]`.

- Apache Hive JSON SerDe
- OpenX JSON SerDe

Wählen Sie den JSON-Deserializer

Wählen Sie OpenX JSON, SerDe wenn Ihr Eingabe-JSON Zeitstempel in den folgenden Formaten enthält:

- yyyy-MM-dd'T'hh:mm:ss [.S] 'Z', wobei der Bruch bis zu 9 Ziffern haben kann — zum Beispiel. `2017-02-07T15:13:01.39256Z`
- yyyy-[M]M-[d]d HH:mm:ss[.S], wobei der Bruchteil bis zu 9 Stellen haben kann – z. B. `2017-02-07 15:13:01.14`.
- Epoch-Sekunden – z. B. `1518033528`.
- Epoch-Millisekunden – z. B. `1518033528123`.
- Fließkomma-Epoch-Sekunden – z. B. `1518033528.123`.

Das OpenX-JSON SerDe kann Punkte (`.`) in Unterstriche (`_`) konvertieren. Es kann außerdem JSON-Schlüssel in Kleinbuchstaben konvertieren, bevor er sie deserialisiert. Weitere Informationen zu den Optionen, die mit diesem Deserializer über Amazon Data Firehose verfügbar sind, finden Sie unter Öffnen. XJson SerDe

Wenn Sie sich nicht sicher sind, welchen Deserializer Sie wählen sollen, verwenden Sie OpenX JSON, es sei denn SerDe, Sie haben Zeitstempel, die er nicht unterstützt.

Wenn Sie Zeitstempel in anderen als den zuvor aufgeführten Formaten haben, verwenden Sie Apache Hive JSON. SerDe Wenn Sie diesen Deserializer wählen, können Sie die zu verwendenden Zeitstempel-Formate angeben. Dazu wenden Sie die Mustersyntax der Joda-Time `DateTimeFormat`-Formatzeichenfolgen an. Weitere Informationen finden Sie unter Class DateTimeFormat.

Sie können auch den speziellen Wert `millis` zum Analysieren von Zeitstempeln in Epoch-Millisekunden verwenden. Wenn Sie kein Format angeben, verwendet Amazon Data Firehose `java.sql.Timestamp::valueOf` standardmäßig.

Das Hive-JSON erlaubt Folgendes SerDe nicht:

- Punkte (`.`) in Spaltennamen.
- Felder mit dem Typ `uniontype`.
- Felder, für die numerische Typen im Schema angegeben sind, die aber im JSON Zeichenfolgen sind. Wenn das Schema beispielsweise (ein Int) und das JSON ist`{"a":"123"}`, SerDe gibt Hive einen Fehler aus.

Der Hive konvertiert verschachteltes JSON SerDe nicht in Zeichenketten. Wenn Sie zum Beispiel `{"a":{"inner":1}}` haben, behandelt er `{"inner":1}` nicht als Zeichenfolge.

Schema

Amazon Data Firehose benötigt ein Schema, um zu bestimmen, wie diese Daten interpretiert werden sollen. Verwenden Sie AWS Glue, um ein Schema in der zu erstellen AWS Glue Data Catalog. Amazon Data Firehose referenziert dann dieses Schema und verwendet es, um Ihre Eingabedaten zu interpretieren. Sie können dasselbe Schema verwenden, um sowohl Amazon Data Firehose als auch Ihre Analysesoftware zu konfigurieren. Weitere Informationen finden Sie unter Füllen des AWS Glue-Datenkatalogs im AWS Glue Entwicklerhandbuch.

> **Note**
>
> Das im AWS Glue Datenkatalog erstellte Schema sollte der Eingabedatenstruktur entsprechen. Andernfalls enthalten die konvertierten Daten keine Attribute, die nicht im Schema angegeben sind. Wenn Sie verschachteltes JSON verwenden, verwenden Sie einen STRUCT-Typ im Schema, der die Struktur Ihrer JSON-Daten widerspiegelt. In diesem Beispiel erfahren Sie, wie Sie verschachteltes JSON mit einem STRUCT-Typ behandeln.

> **Important**
>
> Für Datentypen, die keine Größenbeschränkung angeben, gibt es eine praktische Grenze von 32 MBs für alle Daten in einer einzigen Zeile.
> Wenn Sie die Länge für `CHAR` oder angeben`VARCHAR`, kürzt Firehose die Zeichenketten beim Lesen der Eingabedaten auf die angegebene Länge. Wenn die zugrunde liegende Datenzeichenfolge länger ist, bleibt sie unverändert.

Serializer

Firehose benötigt einen Serializer, um die Daten in das spaltenförmige Zielspeicherformat (Parquet oder ORC) zu konvertieren. Sie können einen der folgenden beiden Serialisierertypen wählen.

- ORC SerDe
- Parkett SerDe

Wählen Sie den Serializer

Welchen Serializer Sie auswählen sollten, hängt von den Anforderungen Ihres Unternehmens ab. Weitere Informationen zu den beiden Serializer-Optionen finden Sie unter ORC und Parquet. SerDe SerDe

Aktivieren Sie die Konvertierung des Datensatzformats

Wenn Sie die Konvertierung von Datensatzformaten aktivieren, können Sie Ihr Amazon Data Firehose-Ziel nicht auf Amazon OpenSearch Service, Amazon Redshift oder Splunk festlegen. Wenn die Formatkonvertierung aktiviert ist, ist Amazon S3 das einzige Ziel, das Sie für Ihren Firehose-Stream verwenden können. Der folgende Abschnitt zeigt, wie Sie die Konvertierung von Datensatzformaten aus Konsolen- und Firehose-API-Vorgängen aktivieren. Ein Beispiel für die Einrichtung der Konvertierung von Datensatzformaten mit CloudFormation finden Sie unter AWS:::DataFirehose: DeliveryStream.

Aktivieren Sie die Konvertierung des Datensatzformats von der Konsole aus

Sie können die Datenformatkonvertierung auf der Konsole aktivieren, wenn Sie einen Firehose-Stream erstellen oder aktualisieren. Wenn die Datenformatkonvertierung aktiviert ist, ist Amazon S3 das einzige Ziel, das Sie für den Firehose-Stream konfigurieren können. Außerdem wird beim Aktivieren einer Formatkonvertierung die Amazon-S3-Komprimierung deaktiviert. Die Snappy-Komprimierung erfolgt jedoch automatisch als Teil des Konvertierungsvorgangs. Das Framing-Format für Snappy, das Amazon Data Firehose in diesem Fall verwendet, ist mit Hadoop kompatibel. Das bedeutet, dass Sie die Ergebnisse der Snappy-Komprimierung verwenden und für diese Daten Abfragen in Athena ausführen können. Informationen zum Snappy-Framing-Format, auf das Hadoop angewiesen ist, finden Sie unter .java. BlockCompressorStream

Um die Datenformatkonvertierung für einen Firehose-Datenstream zu aktivieren

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Data Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie einen Firehose-Stream aus, der aktualisiert werden soll, oder erstellen Sie einen neuen Firehose-Stream, indem Sie die Schritte unter befolgen. Tutorial: Einen Firehose-Stream von der Konsole aus erstellen
3. Setzen Sie unter Convert record format (Datensatzformat konvertieren) die Option Record format conversion (Datensatzformat-Konvertierung) auf Enabled (Aktiviert).

4. Wählen Sie die Option aus, die Sie hinzufügen möchten. Weitere Informationen zu den beiden Optionen finden Sie unter Apache Parquet und Apache ORC.
5. Wählen Sie eine AWS Glue Tabelle aus, um ein Schema für Ihre Quelldatensätze anzugeben. Legen Sie die Region, Datenbank, Tabelle und Tabellenversion fest.

Verwalten Sie die Konvertierung des Datensatzformats über die Firehose-API

Wenn Sie möchten, dass Amazon Data Firehose das Format Ihrer Eingabedaten von JSON nach Parquet oder ORC konvertiert, geben Sie das optionale DataFormatConversionConfigurationElement in ExtendedS3 oder in ExtendedS3 DestinationConfiguration an. DestinationUpdate Wenn Sie dies angeben, gelten die folgenden Einschränkungen. DataFormatConversionConfiguration

- BufferingHintsIn können Sie keinen Wert unter 64 festlegen`SizeInMBs`, wenn Sie die Konvertierung des Datensatzformats aktivieren. Wenn Formatkonvertierung nicht aktiviert ist, lautet der Standardwert 5. Bei Aktivierung wird der Wert 128.
- Sie müssen in ExtendedS3 DestinationConfiguration oder `CompressionFormat` in ExtendedS3 auf einstellen. DestinationUpdate `UNCOMPRESSED` Der Standardwert für den `CompressionFormat` beträgt `UNCOMPRESSED`. Daher können Sie es in ExtendedS3 auch unspezifiziert lassen. DestinationConfiguration Die Daten werden als Teil der Serialisierungsprozesses dennoch komprimiert. Dazu wird standardmäßig die Snappy-Komprimierung verwendet. Das Framing-Format für Snappy, das Amazon Data Firehose in diesem Fall verwendet, ist mit Hadoop kompatibel. Das bedeutet, dass Sie die Ergebnisse der Snappy-Komprimierung verwenden und für diese Daten Abfragen in Athena ausführen können. Informationen zum Snappy-Framing-Format, auf das Hadoop angewiesen ist, finden Sie unter .java. BlockCompressorStream Wenn Sie den Serializer konfigurieren, können Sie andere Arten der Komprimierung auswählen.

Behandlung von Fehlern bei der Konvertierung von Datenformaten

Wenn Amazon Data Firehose einen Datensatz nicht analysieren oder deserialisieren kann (z. B. wenn die Daten nicht dem Schema entsprechen), schreibt es ihn mit einem Fehlerpräfix in Amazon S3. Wenn dieser Schreibvorgang fehlschlägt, wiederholt Amazon Data Firehose den Vorgang für immer, wodurch die weitere Zustellung blockiert wird. Für jeden fehlgeschlagenen Datensatz schreibt Amazon Data Firehose ein JSON-Dokument mit dem folgenden Schema:

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

Verstehen Sie die Datenlieferung in Amazon Data Firehose

Wenn Sie Daten an Ihren Firehose-Stream senden, werden sie automatisch an das von Ihnen gewählte Ziel gesendet. In der folgenden Tabelle wird die Datenzustellung an verschiedene Ziele erklärt.

Bestimmungsort	Details
Amazon S3	Für die Datenlieferung an Amazon S3 verkettet Firehose mehrere eingehende Datensätze auf der Grundlage der Pufferkonfiguration Ihres Firehose-Streams. Anschließend übermittelt es die Datensätze als Amazon-S3-Objekt an Amazon S3. Standardmäßig verkettet Firehose Daten ohne Trennzeichen. Wenn Sie neue Zeilentrennzeichen zwischen Datensätzen haben möchten, können Sie neue Zeilentrennzeichen hinzufügen, indem Sie die Funktion in der Firehose-Konsolenkonfiguration oder im API-Parameter aktivieren. Die Datenübermittlung zwischen Firehose und dem Amazon S3 S3-Ziel ist mit TLS (HTTPS) verschlüsselt.
Amazon Redshift	Für die Datenlieferung an Amazon Redshift liefert Firehose zunächst eingehende Daten in dem zuvor beschriebenen Format an Ihren S3-Bucket. Firehose gibt dann einen Amazon COPY Redshift-Befehl aus, um die Daten aus Ihrem S3-Bucket in Ihren von Amazon Redshift bereitgestellten Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe zu laden. Stellen Sie sicher, dass, nachdem Amazon Data Firehose mehrere eingehende Datensätze zu einem Amazon S3 S3-Objekt verkettet hat, das Amazon S3 S3-Objekt in Ihren von Amazon Redshift bereitgestellten Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe kopiert werden kann. Weitere Informationen finden Sie unter Amazon Redshift COPY Command Data Format Parameters.
OpenSearch OpenSearch Service und Serverless	Für die Datenlieferung an OpenSearch Service und OpenSearch Serverless puffert Amazon Data Firehose eingehende Datensätze auf der Grundlage der Pufferkonfiguration Ihres Firehose-Streams. Anschließend generiert es eine OpenSearch Service- oder OpenSearch Serverless-Massenanforderung, um mehrere Datensätz

Bestimmungsort	Details
	e in Ihrem Service-Cluster oder Ihrer OpenSearch Serverless-Sammlung zu indizieren. OpenSearch Stellen Sie sicher, dass Ihr Datensatz UTF-8-kodiert und auf ein einzeiliges JSON-Objekt reduziert ist, bevor Sie ihn an Amazon Data Firehose senden. Außerdem muss die `rest.action.multi.allow_explicit_index` Option für Ihren OpenSearch Service-Cluster auf true (Standard) gesetzt sein, um Massenanfragen mit einem expliziten Index entgegenzunehmen, der pro Datensatz festgelegt wird. Weitere Informationen finden Sie unter OpenSearch Service Configure Advanced Options im Amazon OpenSearch Service Developer Guide.
Splunk	Für die Datenlieferung an Splunk verkettet Amazon Data Firehose die von Ihnen gesendeten Bytes. Wenn Sie Trennzeichen in Ihren Daten wünschen, wie z. B. ein Neue-Zeile-Zeichen, müssen Sie sie selbst einfügen. Stellen Sie sicher, dass Splunk so konfiguriert ist, dass diese Trennzeichen bei der Analyse berücksichtigt werden. Gehen Sie wie in der Splunk-Dokumentation beschrieben vor, um die Daten, die an den S3-Fehler-Bucket (S3-Backup) übermittelt wurden, wieder an Splunk weiterzuleiten.
HTTP-Endpunkt	Für die Datenlieferung an einen HTTP-Endpunkt, der einem unterstützten Drittanbieter gehört, können Sie den integrierten Amazon Lambda-Service verwenden, um eine Funktion zu erstellen, um die eingehenden Datensätze in das Format umzuwandeln, das dem Format entspricht, das die Integration des Dienstanbieters erwartet. Wenden Sie sich an den Drittanbieter, dessen HTTP-Endpunkt Sie für Ihr Ziel ausgewählt haben, um mehr über das akzeptierte Datensatzformat zu erfahren.

Bestimmungsort	Details
Snowflake	Für die Datenlieferung an Snowflake puffert Amazon Data Firehose intern Daten für eine Sekunde und verwendet Snowflake-Streamin g-API-Operationen, um Daten in Snowflake einzufügen. Standardm äßig werden Datensätze, die Sie einfügen, jede Sekunde geleert und in die Snowflake-Tabelle übernommen. Nachdem Sie den Insert-Au fruf ausgeführt haben, gibt Firehose eine CloudWatch Metrik aus, die misst, wie lange es gedauert hat, bis die Daten an Snowflake übergeben wurden. Firehose unterstützt derzeit nur ein einzelnes JSON-Element als Datensatznutzlast und unterstützt keine JSON-Arrays. Stellen Sie sicher, dass Ihre Eingabe-Payload ein gültiges JSON-Objekt ist und ohne zusätzliche doppelte Anführungszeichen, Anführungszeichen oder Escape-Zeichen korrekt formatiert ist.

Jedes Firehose-Ziel hat seine eigene Datenlieferfrequenz. Weitere Informationen finden Sie unter Pufferhinweise konfigurieren.

Doppelte Datensätze

Amazon Data Firehose verwendet at-least-once Semantik für die Datenlieferung. Unter bestimmten Umständen, z. B. wenn das Zeitlimit für die Datenzustellung überschritten wird, kann es bei erneuten Zustellungsversuchen von Amazon Data Firehose zu Duplikaten kommen, wenn die ursprüngliche Datenlieferanforderung irgendwann durchgeht. Dies gilt für alle Zieltypen, die Amazon Data Firehose unterstützt, mit Ausnahme von Amazon S3 S3-Zielen, Apache Iceberg Tables und Snowflake-Zielen.

Themen

- Verstehen Sie den Versand zwischen Konten und Regionen AWS
- Verstehen Sie die Anforderungen- und Antwortspezifikationen für die HTTP-Endpunktzustellung
- Behandeln Sie Fehler bei der Datenzustellung
- Amazon S3 S3-Objektnamenformat konfigurieren
- Konfigurieren Sie die Indexrotation für Service OpenSearch
- Die Datenübermittlung unterbrechen und fortsetzen

Verstehen Sie den Versand zwischen Konten und Regionen AWS

Amazon Data Firehose unterstützt die AWS kontenübergreifende Datenübermittlung an HTTP-Endpunktziele. Der Firehose-Stream und der HTTP-Endpunkt, den Sie als Ziel wählen, können zu unterschiedlichen AWS Konten gehören.

Amazon Data Firehose unterstützt auch die Datenzustellung an HTTP-Endpunktziele in allen AWS Regionen. Sie können Daten aus einem Firehose-Stream in einer AWS Region an einen HTTP-Endpunkt in einer anderen AWS Region liefern. Sie können Daten auch von einem Firehose-Stream an ein HTTP-Endpunktziel außerhalb von AWS Regionen liefern, z. B. an Ihren eigenen lokalen Server, indem Sie die HTTP-Endpunkt-URL auf Ihr gewünschtes Ziel setzen. In diesen Szenarien werden zusätzliche Datenübertragungsgebühren zu Ihren Lieferkosten hinzugefügt. Weitere Informationen finden Sie im Abschnitt Datenübertragung auf der Seite On-Demand-Preise.

Verstehen Sie die Anforderungen- und Antwortspezifikationen für die HTTP-Endpunktzustellung

Damit Amazon Data Firehose erfolgreich Daten an benutzerdefinierte HTTP-Endpunkte liefern kann, müssen diese Endpunkte Anfragen annehmen und Antworten in bestimmten Amazon Data Firehose-Anfrage- und Antwortformaten senden. In diesem Abschnitt werden die Formatspezifikationen der HTTP-Anfragen beschrieben, die der Amazon Data Firehose-Service an benutzerdefinierte HTTP-Endpunkte sendet, sowie die Formatspezifikationen der HTTP-Antworten, die der Amazon Data Firehose-Service erwartet. HTTP-Endpunkte haben 3 Minuten Zeit, um auf eine Anfrage zu antworten, bevor Amazon Data Firehose das Zeitlimit für diese Anfrage überschreitet. Amazon Data Firehose behandelt Antworten, die nicht dem richtigen Format entsprechen, als Zustellungsfehler.

Anforderungsformat

Pfad- und URL-Parameter

Diese werden direkt von Ihnen als Teil eines einzigen URL-Felds konfiguriert. Amazon Data Firehose sendet sie wie konfiguriert ohne Änderung. Es werden nur HTTPS-Ziele unterstützt. URL-Einschränkungen werden bei der Konfiguration des Bereitstellungsdatenstroms angewendet.

> **Note**
>
> Derzeit wird nur Port 443 für die Bereitstellung von HTTP-Endpunktdaten unterstützt.

HTTP-Header — -Version X-Amz-Firehose-Protocol

Dieser Header wird verwendet, um die Version der Anforderungs-/Antwortformate anzugeben. Derzeit gibt es nur die Version 1.0.

HTTP-Header - -ID X-Amz-Firehose-Request

Der Wert dieses Headers ist eine undurchsichtige GUID, die für Debugging- und Deduplizierungszwecke verwendet werden kann. Endpunktimplementierungen sollten den Wert dieses Headers nach Möglichkeit sowohl für erfolgreiche als auch für erfolglose Anfragen protokollieren. Die Anforderungs-ID wird bei mehreren Versuchen derselben Anfrage unverändert beibehalten.

HTTP-Header – Inhaltstyp

Der Wert des Content-Type-Headers ist immer `application/json`.

HTTP-Header – Inhaltskodierung

Ein Firehose-Stream kann so konfiguriert werden, dass er GZIP verwendet, um den Hauptteil beim Senden von Anfragen zu komprimieren. Wenn diese Komprimierung aktiviert ist, wird der Wert des Content-Encoding-Headers gemäß der Standardpraxis auf gzip gesetzt. Wenn die Komprimierung nicht aktiviert ist, fehlt der Content-Encoding-Header vollständig.

HTTP-Header – Inhaltslänge

Dies wird standardmäßig verwendet.

HTTP-Header - -Arn: X-Amz-Firehose-Source

Der ARN des Firehose-Streams, dargestellt im ASCII-String-Format. Der ARN kodiert die Region, die AWS Konto-ID und den Streamnamen. Beispiel, `arn:aws:firehose:us-east-1:123456789:deliverystream/testStream`.

HTTP-Header — -Schlüssel X-Amz-Firehose-Access

Dieser Header enthält einen API-Schlüssel oder andere Anmeldeinformationen. Sie haben die Möglichkeit, den API-Schlüssel (auch Autorisierungstoken genannt) zu erstellen oder zu aktualisieren, wenn Sie Ihren Bereitstellungsdatenstrom erstellen oder aktualisieren. Amazon Data Firehose beschränkt die Größe des Zugriffsschlüssels auf 4096 Byte. Amazon Data Firehose versucht in keiner Weise, diesen Schlüssel zu interpretieren. Der konfigurierte Schlüssel wird wortwörtlich in den Wert dieses Headers kopiert. Wenn Sie jedoch Secrets Manager verwenden, um den Schlüssel zu konfigurieren, muss der geheime Schlüssel einem bestimmten JSON-Objektformat folgen:`{"api_key": "..."}`.

Der Inhalt kann beliebig sein und möglicherweise ein JWT-Token oder einen ACCESS_KEY darstellen. Wenn für einen Endpunkt Anmeldeinformationen mit mehreren Feldern erforderlich sind (z. B. Benutzername und Passwort), sollten die Werte aller Felder zusammen in einem einzigen Zugriffsschlüssel in einem Format gespeichert werden, das der Endpunkt versteht (JSON oder CSV). Dieses Feld kann Base-64-kodiert sein, wenn der ursprüngliche Inhalt binär ist. Amazon Data Firehose ändert den and/or konfigurierten Wert nicht und verwendet den Inhalt unverändert.

HTTP-Header — -Attribute X-Amz-Firehose-Common

Dieser Header enthält die gemeinsamen Attribute (Metadaten), die sich auf die gesamte Anfrage beziehen, und zwar and/or auf alle Datensätze innerhalb der Anfrage. Diese werden direkt von Ihnen konfiguriert, wenn Sie einen Firehose-Stream erstellen. Der Wert dieses Attributs ist als JSON-Objekt mit dem folgenden Schema kodiert:

```
"$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
```

Ein Beispiel:

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

Hauptteil – maximale Größe

Die maximale Körpergröße wird von Ihnen konfiguriert und kann vor der Komprimierung bis zu 64 MiB betragen.

Körper – Schema

Der Hauptteil enthält ein einzelnes JSON-Dokument mit dem folgenden JSON-Schema (in YAML geschrieben):

```
"$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
```

Ein Beispiel:

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

Reaktionsformat

Standardverhalten bei einem Fehler

Wenn eine Antwort die folgenden Anforderungen nicht erfüllt, behandelt der Firehose-Server sie so, als ob sie einen Statuscode 500 ohne Text hätte.

Statuscode

Der HTTP-Statuscode MUSS im 2XX-, 4XX- oder 5XX-Bereich liegen.

Der Amazon Data Firehose-Server folgt KEINEN Weiterleitungen (3XX-Statuscodes). Nur der Antwortcode 200 gilt als erfolgreiche Übermittlung der Datensätze an HTTP/EP. Der Antwortcode 413 (Größe überschritten) wird als permanenter Fehler betrachtet und der Datensatzstapel wird nicht an den Fehler-Bucket gesendet, wenn er konfiguriert ist. Alle anderen Antwortcodes gelten als wiederherstellbare Fehler und unterliegen einem Back-off-Wiederholungsalgorithmus, der später erklärt wird.

Header – Inhaltstyp

Der einzig akzeptable Inhaltstyp ist application/json.

HTTP-Header – Inhaltskodierung

Content-Encoding DARF NICHT verwendet werden. Der Hauptteil MUSS unkomprimiert sein.

HTTP-Header – Inhaltslänge

Der Content-Length-Header MUSS vorhanden sein, wenn die Antwort einen Hauptteil hat.

Hauptteil – maximale Größe

Der Antworttext darf höchstens 1 MiB groß sein.

```
"$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
```

Ein Beispiel:

```
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
}
```

Umgang mit Fehlerreaktionen

In allen Fehlerfällen versucht der Amazon Data Firehose-Server erneut, denselben Stapel von Datensätzen mithilfe eines exponentiellen Back-off-Algorithmus zuzustellen. Die Wiederholungen werden anhand einer anfänglichen Back-Off-Zeit (1 Sekunde) mit einem Jitterfaktor von (15%) zurückgesetzt, und jede weitere Wiederholung wird anhand der Formel (initial-backoff-time * (multiplier (2) ^ retry_count)) mit zusätzlichem Jitter zurückgestellt. Die Backoff-Zeit ist auf ein maximales Intervall von 2 Minuten begrenzt. Beispielsweise beträgt die Back-Off-Zeit bei der 'n-ten Wiederholung = MAX (120, 2^n) * random (0,85, 1,15).

Die in der vorherigen Gleichung angegebenen Parameter können sich ändern. Die genaue anfängliche Backoff-Zeit, AWS die maximale Backoff-Zeit, den Multiplikator und die Jitter-Prozentsätze, die im exponentiellen Backoff-Algorithmus verwendet werden, finden Sie in der Firehose-Dokumentation.

Bei jedem nachfolgenden Wiederholungsversuch kann sich das and/or Zugriffsschlüsselziel, an das Datensätze übermittelt werden, basierend auf der aktualisierten Konfiguration des Firehose ändern. Der Amazon Data Firehose-Service verwendet dieselbe Anforderungs-ID für alle Wiederholungen nach bestem Wissen und Gewissen. Das letzte Feature kann vom HTTP-

Endpunktserver zur Deduplizierung verwendet werden. Wenn die Anfrage nach Ablauf der maximal zulässigen Zeit (basierend auf der Firehose-Stream-Konfiguration) immer noch nicht zugestellt wird, kann der Datensatzstapel optional auf der Grundlage der Stream-Konfiguration an einen Fehler-Bucket übermittelt werden.

Beispiele

Beispiel für eine Anfrage mit CWLog Herkunft.

```
{
  "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"
        }
      ]
    }
   }
  ]
}
```

Behandeln Sie Fehler bei der Datenzustellung

Jedes Amazon Data Firehose-Ziel hat seine eigene Behandlung bei Datenlieferfehlern.

Wenn Sie einen Firehose für viele Ziele wie Splunk- und HTTP-Endpunkte einrichten OpenSearch, richten Sie auch einen S3-Bucket ein, in dem Daten, die nicht zugestellt werden können, gesichert werden können. Weitere Informationen darüber, wie Firehose Daten bei fehlgeschlagenen Lieferungen sichert, finden Sie in den entsprechenden Zielabschnitten auf dieser Seite. Weitere Informationen darüber, wie Sie Zugriff auf S3-Buckets gewähren, in denen Daten gesichert werden können, die nicht zugestellt werden können, finden Sie unter Firehose-Zugriff auf ein Amazon S3 S3-Ziel gewähren. Wenn Firehose (a) keine Daten an das Stream-Ziel liefert und (b) keine Daten für fehlgeschlagene Lieferungen in den Backup-S3-Bucket schreibt, wird die Stream-Übertragung effektiv angehalten, bis Daten entweder an das Ziel geliefert oder an den Backup-S3-Speicherort geschrieben werden können.

Amazon S3

Die Datenbereitstellung zu Ihrem S3-Bucket kann aus verschiedenen Gründen fehlschlagen. Beispielsweise ist der Bucket möglicherweise nicht mehr vorhanden, die IAM-Rolle, von der Amazon Data Firehose annimmt, dass sie keinen Zugriff auf den Bucket hat, dass das Netzwerk ausgefallen ist oder ähnliche Ereignisse auftreten. Unter diesen Bedingungen versucht Amazon Data Firehose bis zu 24 Stunden lang erneut, bis die Lieferung erfolgreich ist. Die maximale Datenspeicherzeit von Amazon Data Firehose beträgt 24 Stunden. Falls die Datenbereitstellung länger als 24 Stunden fehlschlägt, gehen die Daten verloren.

Die Datenlieferung an Ihren S3-Bucket kann aus verschiedenen Gründen fehlschlagen, z. B.:

- Der Bucket ist nicht mehr vorhanden.
- Die von Amazon Data Firehose übernommene IAM-Rolle hat keinen Zugriff auf den Bucket.
- Probleme mit dem Netzwerk.
- S3-Fehler, wie HTTP 500s oder andere API-Fehler.

In diesen Fällen versucht Amazon Data Firehose die Lieferung erneut:

- DirectPut Quellen: Wiederholungen dauern bis zu 24 Stunden an.
- Kinesis Data Streams- oder Amazon MSK-Quellen: Wiederholungen werden auf unbestimmte Zeit fortgesetzt, bis die für den Stream definierte Aufbewahrungsrichtlinie eingehalten wird.

Amazon Data Firehose übermittelt fehlgeschlagene Datensätze nur dann an einen S3-Fehler-Bucket, wenn die Lambda-Verarbeitung oder die Parquet-Konvertierung fehlschlägt. Andere Fehlerszenarien führen zu kontinuierlichen Wiederholungsversuchen mit S3, bis die Aufbewahrungsfrist erreicht ist. Wenn Firehose erfolgreich Datensätze an S3 übermittelt, erstellt es eine S3-Objektdatei, und bei teilweisen Datensatzfehlern versucht es automatisch, erneut zuzustellen und aktualisiert dieselbe S3-Objektdatei mit den erfolgreich verarbeiteten Datensätzen.

Amazon Redshift

Für ein Amazon Redshift Redshift-Ziel können Sie beim Erstellen eines Firehose-Streams eine Wiederholungsdauer (0—7200 Sekunden) angeben.

Die Datenbereitstellung an Ihren bereitgestellten Amazon-Redshift-Cluster oder Ihre Arbeitsgruppe von Amazon Redshift Serverless kann aus verschiedenen Gründen fehlschlagen. Möglicherweise haben Sie beispielsweise eine falsche Clusterkonfiguration Ihres Firehose-Streams, einen Cluster oder eine Arbeitsgruppe, die gewartet wird, oder es liegt ein Netzwerkausfall vor. Unter diesen Bedingungen versucht Amazon Data Firehose es für die angegebene Zeitdauer erneut und überspringt diesen bestimmten Stapel von Amazon S3 S3-Objekten. Die Informationen zu den übersprungenen Objekten werden Ihrem S3-Bucket als Manifestdatei im Ordner `errors/` bereitgestellt. Sie können diesen für manuelle Backfill-Vorgänge verwenden. Informationen darüber, wie Sie Daten manuell mit Manifestdateien kopieren, finden Sie unter Verwenden eines Manifests für die Angabe von Datendateien.

Amazon OpenSearch Service und OpenSearch Serverless

Für das OpenSearch Service- und OpenSearch Serverless-Ziel können Sie bei der Erstellung des Firehose-Streams eine Wiederholungsdauer (0—7200 Sekunden) angeben.

Die Datenzustellung an Ihren OpenSearch Service-Cluster oder Ihre OpenSearch serverlose Sammlung kann aus verschiedenen Gründen fehlschlagen. Möglicherweise haben Sie beispielsweise eine falsche OpenSearch Service Cluster- oder OpenSearch Serverless Collection-Konfiguration Ihres Firehose-Streams, einen OpenSearch Service-Cluster oder eine OpenSearch Serverless-Sammlung, die gerade gewartet wird, ein Netzwerkausfall oder ähnliche Ereignisse vorliegen. Unter diesen Bedingungen versucht Amazon Data Firehose es für die angegebene Zeitdauer erneut und überspringt dann die jeweilige Indexanforderung. Die übersprungenen Dokumente werden Ihrem S3-Bucket im Ordner `Amazon0penSearchService_failed/` bereitgestellt. Sie können diesen für manuelle Backfill-Vorgänge verwenden.

Für OpenSearch Service hat jedes Dokument das folgende JSON-Format:

```
{
    "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)"
}
```

Bei OpenSearch Serverless hat jedes Dokument das folgende JSON-Format:

```
{
    "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

Wenn Amazon Data Firehose Daten an Splunk sendet, wartet es auf eine Bestätigung von Splunk. Wenn ein Fehler auftritt oder die Bestätigung nicht innerhalb des Zeitlimits für die Bestätigung eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Der Vorgang wird wiederholt, bis die Wiederholungsdauer abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an Splunk sendet, unabhängig davon, ob es sich um einen ersten Versuch oder einen erneuten Versuch handelt, wird der Timeout-Zähler für die Bestätigung neu gestartet. Er wartet dann auf eine Bestätigung von Splunk. Selbst wenn die

Dauer des Wiederholungsversuchs abläuft, wartet Amazon Data Firehose immer noch auf die Bestätigung, bis sie eingeht oder das Bestätigungs-Timeout erreicht ist. Wenn bei der Bestätigung eine Zeitüberschreitung eintritt, prüft Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Bestätigung erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Eine fehlende Bestätigung ist nicht der einzige Fehlertyp, der bei einer Datenübermittlung auftreten kann. Weitere Informationen zu den anderen Fehlertypen finden Sie unter Fehler bei der Datenbereitstellung für Splunk. Ein Fehler bei der Datenbereitstellung löst die Wiederhollogik aus, wenn Ihre Wiederholdauer größer als 0 ist.

Nachfolgend sehen Sie ein Beispiel für einen Fehlerdatensatz.

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

HTTP-Endpunktziel

Wenn Amazon Data Firehose Daten an ein HTTP-Endpunktziel sendet, wartet es auf eine Antwort von diesem Ziel. Wenn ein Fehler auftritt oder die Antwort nicht innerhalb des Antwort-Timeouts eingeht, startet Amazon Data Firehose den Zähler für die Dauer der Wiederholungsversuche. Der Vorgang wird wiederholt, bis die Wiederholungsdauer abgelaufen ist. Danach betrachtet Amazon Data Firehose den Fehler bei der Datenübermittlung und sichert die Daten in Ihrem Amazon S3 S3-Bucket.

Jedes Mal, wenn Amazon Data Firehose Daten an ein HTTP-Endpunktziel sendet, unabhängig davon, ob es sich um den ersten Versuch oder einen erneuten Versuch handelt, wird der Antwort-Timeout-Zähler neu gestartet. Anschließend wartet es darauf, dass eine Antwort vom HTTP-Endpunktziel eingeht. Selbst wenn die Wiederholungsdauer abläuft, wartet Amazon Data Firehose

immer noch auf die Antwort, bis sie eingeht oder das Antwort-Timeout erreicht ist. Wenn bei der Antwort ein Timeout auftritt, prüft Amazon Data Firehose, ob im Wiederholungszähler noch Zeit übrig ist. Ist noch Zeit übrig, führt es erneut eine Wiederholung durch und wiederholt die Logik, bis es eine Response erhält, oder feststellt, dass die Wiederholungszeitdauer abgelaufen ist.

Eine fehlende Response ist nicht der einzige Fehlertyp, der bei einer Datenübermittlung auftreten kann. Weitere Informationen zu den anderen Fehlertypen finden Sie unter Fehler bei der Datenbereitstellung für den HTTP-Endpunkt

Nachfolgend sehen Sie ein Beispiel für einen Fehlerdatensatz.

```
{
 "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

Für das Snowflake-Ziel können Sie beim Erstellen Firehose Firehose-Streams eine optionale Wiederholungsdauer (0-7200 Sekunden) angeben. Der Standardwert für die Dauer der Wiederholungen beträgt 60 Sekunden.

Die Datenübermittlung an Ihre Snowflake-Tabelle kann aus verschiedenen Gründen fehlschlagen, z. B. aufgrund einer falschen Snowflake-Zielkonfiguration, eines Snowflake-Ausfalls, eines Netzwerkausfalls usw. Die Wiederholungsrichtlinie gilt nicht für Fehler, die nicht behoben werden können. Wenn Snowflake beispielsweise Ihre JSON-Nutzlast ablehnt, weil sie eine zusätzliche Spalte hatte, die in der Tabelle fehlt, versucht Firehose nicht, sie erneut zu liefern. Stattdessen wird eine Sicherungskopie für alle Einfügefehler erstellt, die auf Probleme mit der JSON-Nutzlast in Ihrem S3-Fehler-Bucket zurückzuführen sind.

Wenn die Lieferung aufgrund einer falschen Rolle, Tabelle oder Datenbank fehlschlägt, versucht Firehose ebenfalls nicht erneut und schreibt die Daten in Ihren S3-Bucket. Die Dauer

der Wiederholungsversuche gilt nur für Fehler aufgrund eines Snowflake-Dienstproblems, vorübergehender Netzwerkstörungen usw. Unter diesen Bedingungen versucht Firehose für die angegebene Zeitdauer erneut, bevor sie an S3 gesendet werden. Die fehlgeschlagenen Datensätze werden im Ordner snowflake-failed/ geliefert, den Sie für manuelles Auffüllen verwenden können.

Im Folgenden finden Sie ein JSON-Beispiel für jeden Datensatz, den Sie an S3 liefern.

```
{
    "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 S3-Objektnamenformat konfigurieren

Wenn Firehose Daten an Amazon S3 liefert, folgt der Name des S3-Objektschlüssels dem Format <evaluated prefix><suffix>, wobei das Suffix das Format - - - - - - <Firehose stream name><Firehose stream version><year><month><day><hour><minute><second>hat, <uuid><file extension><Firehose stream version>mit 1 beginnt und bei jeder Konfigurationsänderung des Firehose-Streams um 1 erhöht wird. Sie können die Firehose-Stream-Konfigurationen ändern (z. B. den Namen des S3-Buckets, Pufferhinweise, Komprimierung und Verschlüsselung). Sie können dies mithilfe der Firehose-Konsole oder der UpdateDestinationAPI-Operation tun.

Denn <evaluated prefix>Firehose fügt dem Format `YYYY/MM/dd/HH` ein Standard-Zeitpräfix hinzu. Dieses Präfix erstellt eine logische Hierarchie im Bucket, wobei jeder Schrägstrich (/) eine Ebene in der Hierarchie erzeugt. Sie können diese Struktur ändern, indem Sie ein benutzerdefiniertes Präfix angeben, das Ausdrücke enthält, die zur Laufzeit ausgewertet werden. Informationen zur Angabe eines benutzerdefinierten Präfixes finden Sie unter Benutzerdefinierte Präfixe für Amazon Simple Storage Service Objects.

Standardmäßig ist die Zeitzone, die für das Zeitpräfix und das Suffix verwendet wird, UTC, aber Sie können sie in eine Zeitzone ändern, die Sie bevorzugen. Um beispielsweise Japan Standard Time anstelle von UTC zu verwenden, können Sie die Zeitzone für Asien/Tokio in der AWS-Managementkonsole oder in der API-Parametereinstellung () CustomTimeZone konfigurieren. Die folgende Liste enthält Zeitzonen, die Firehose für die S3-Präfixkonfiguration unterstützt.

Unterstützte Zeitzonen

Im Folgenden finden Sie eine Liste der Zeitzonen, die Firehose für die S3-Präfixkonfiguration unterstützt.

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>Sie können das Suffixfeld nur ändern. Wenn Sie die Konvertierung oder Komprimierung von Datenformaten aktivieren, hängt Firehose eine auf der Konfiguration basierende Dateierweiterung an. In der folgenden Tabelle wird die von Firehose angehängte Standarddateierweiterung erklärt:

Konfiguration	Dateierweiterung
Konvertierung von Datenformaten: Parquet	.parquet
Konvertierung von Datenformaten: ORC	.orc
Komprimierung: Gzip	.gz
Komprimierung: Zip	.zip

Konfiguration	Dateierweiterung
Komprimierung: Snappy	.snappy
Komprimierung: Hadoop-Snappy	.hsnappy

Sie können auch eine von Ihnen bevorzugte Dateierweiterung in der Firehose-Konsole oder -API angeben. Die Dateierweiterung muss mit einem Punkt (.) beginnen und kann die zulässigen Zeichen enthalten: 0-9a-z! -_.*' (). Die Dateierweiterung darf 128 Zeichen nicht überschreiten.

> **Note**
>
> Wenn Sie eine Dateierweiterung angeben, überschreibt diese die Standarddateierweiterung, die Firehose hinzufügt, wenn die Datenformatkonvertierung oder -komprimierung aktiviert ist.

Verstehen Sie benutzerdefinierte Präfixe für Amazon S3 S3-Objekte

Objekte, die an Amazon S3 geliefert werden, folgen dem Namensformat von <evaluated prefix><suffix>. Sie können Ihr benutzerdefiniertes Präfix angeben, das Ausdrücke enthält, die zur Laufzeit ausgewertet werden. Das von Ihnen angegebene benutzerdefinierte Präfix überschreibt das Standardpräfix von`yyyy/MM/dd/HH`.

Sie können Ausdrücke der folgenden Formen in Ihrem benutzerdefinierten Präfix verwenden: `!{namespace:`*`value`*`}`, wobei `namespace` einer von den beiden sein kann, wie in den folgenden Abschnitten erläutert.

- `firehose`
- `timestamp`
- `partitionKeyFromQuery`
- `partitionKeyFromLambda`

Wenn ein Präfix mit einem Schrägstrich endet, wird es als Ordner im Amazon-S3-Bucket angezeigt. Weitere Informationen finden Sie unter Amazon S3 Object Name Format im Amazon Data FirehoseDeveloper Guide.

Namespace **timestamp**

Gültige Werte für diesen Namespace sind Zeichenketten, die gültige DateTimeFormatterJava-Zeichenketten sind. Beispiel: Im Jahr 2018 wird der Ausdruck `!{timestamp:yyyy}` als `2018` ausgewertet.

Bei der Auswertung von Zeitstempeln verwendet Firehose den ungefähren Ankunftszeitstempel des ältesten Datensatzes, der in dem zu schreibenden Amazon S3 S3-Objekt enthalten ist.

Standardmäßig ist der Zeitstempel in UTC. Sie können jedoch eine Zeitzone angeben, die Sie bevorzugen. Sie können beispielsweise die Zeitzone für Asien/Tokio in der AWS-Managementkonsole oder in der API-Parametereinstellung (CustomTimeZone) konfigurieren, wenn Sie Japan Standard Time anstelle von UTC verwenden möchten. Eine Liste der unterstützten Zeitzonen finden Sie unter Amazon S3 Object Name Format.

Wenn Sie den Namespace `timestamp` mehr als einmal in demselben Präfixausdruck verwenden, werden alle Instances mit demselben Zeitpunkt ausgewertet.

Namespace **firehose**

Es gibt zwei Werte, die Sie mit diesem Namespace verwenden können: `error-output-type` und `random-string`. In der folgenden Tabelle wird beschrieben, wie Sie diese verwenden.

Die **firehose**-Namespace-Werte

Konvertierung	Beschreibung	Beispieleingabe	Beispielausgabe	Hinweise
`error-output-type`	Ergibt je nach Konfiguration Ihres Firehose-Streams und der Ursache des Fehlers eine der folgenden Zeichenketten: {processing-failed, -failed, AmazonOpenSearchService splunk-fa	`myPrefix/result=!{firehose:error-output-type}/!{timestamp:yyyy/MM/dd}`	`myPrefix/result=processing-failed/2018/08/03`	Der error-output-type Wert kann nur in dem Feld verwendet werden. ErrorOutputPrefix

Konvertierung	Beschreibung	Beispieleingabe	Beispielausgabe	Hinweise
	iled,,}. format-co nversion-failed http-endpoint-fail ed Wenn Sie ihn mehr als einmal in demselben Ausdruck verwenden , werden alle Instances als dieselbe Fehlerzei chenfolge ausgewertet.			

Konvertierung	Beschreibung	Beispieleingabe	Beispielausgabe	Hinweise
`random-string`	Wird als zufällige Zeichenfolge von 11 Zeichen ausgewertet. Wenn Sie ihn mehr als einmal in demselben Ausdruck verwenden, werden alle Instances als neue zufällige Zeichenfolge ausgewertet.	`myPrefix/!{firehose:random-string}/`	`myPrefix/046b6c7f-0b/`	Sie können ihn mit beiden Präfixtypen verwenden. Sie können ihn an den Anfang der Formatzeichenfolge setzen, um ein zufälliges Präfix abzurufen. Dies ist manchmal erforderlich, wenn sie einen extrem hohen Durchsatz mit Amazon S3 erreichen möchten.

partitionKeyFromLambda- und **partitionKeyFromQuery**-Namespaces

Für die dynamische Partitionierung müssen Sie das folgende Ausdrucksformat in Ihrem S3-Bucket-Präfix verwenden: `!{namespace:value}`, wobei Namespace entweder `partitionKeyFromQuery`, `partitionKeyFromLambda` oder beides sein kann. Wenn Sie Inline-Parsing verwenden, um die Partitionierungsschlüssel für Ihre Quelldaten zu erstellen, müssen Sie einen S3-Bucket-Präfixwert angeben, der aus Ausdrücken besteht, die im folgenden Format angegeben sind: `"partitionKeyFromQuery:keyID"`. Wenn Sie AWS -Lambda-Funktion verwenden, um die Partitionierungsschlüssel für Ihre Quelldaten zu erstellen, müssen Sie einen S3-Bucket-Präfixwert angeben, der aus Ausdrücken besteht, die im folgenden Format angegeben sind: `"partitionKeyFromLambda:keyID"`. Weitere Informationen finden Sie unter „Wählen Sie Amazon S3 für Ihr Ziel“ unter Amazon Firehose-Stream erstellen.

Semantische Regeln

Folgende Regeln gelten für die Ausdrücke `Prefix` und `ErrorOutputPrefix`.

- Für den Namespace `timestamp` wird jedes Zeichen ausgewertet, das nicht in einfache Anführungszeichen gesetzt ist. Anders ausgedrückt: Alle Zeichenfolgen mit durch Escape-Zeichen geschützten einfachen Anführungszeichen im Wertefeld werden unverändert übernommen.
- Wenn Sie ein Präfix angeben, das keinen Timestamp-Namespace-Ausdruck enthält, hängt Firehose den Ausdruck an den Wert im `!{timestamp:yyyy/MM/dd/HH/}` Feld an. `Prefix`
- Die Sequenz `!{` kann nur in `!{namespace:value}`-Ausdrücken angezeigt werden.
- `ErrorOutputPrefix` kann nur dann Null sein, wenn `Prefix` keine Ausdrücke enthält. In diesem Fall wird `Prefix` als `<specified-prefix>yyyy/MM/DDD/HH/` und `ErrorOutputPrefix` als `<specified-prefix><error-output-type>yyyy/MM/DDD/HH/` ausgewertet. `DDD` repräsentiert den Tag des Jahres.
- Wenn Sie einen Ausdruck für `ErrorOutputPrefix` angeben, müssen Sie mindestens eine Instance von `!{firehose:error-output-type}` einschließen.
- `Prefix` kann nicht `!{firehose:error-output-type}` enthalten.
- Weder `Prefix` noch `ErrorOutputPrefix` können nach der Auswertung länger als 512 Zeichen sein.
- Wenn das Ziel Amazon Redshift ist, darf `Prefix` keine Ausdrücke enthalten und `ErrorOutputPrefix` muss Null sein.
- Wenn das Ziel Amazon OpenSearch Service oder Splunk ist und kein Ziel angegeben `ErrorOutputPrefix` ist, verwendet Firehose das `Prefix` Feld für fehlgeschlagene Datensätze.
- Wenn das Ziel Amazon S3 ist, werden das `Prefix` und `ErrorOutputPrefix` in der Amazon-S3-Zielkonfiguration für erfolgreiche Datensätze bzw. fehlgeschlagene Datensätze verwendet. Wenn Sie die AWS CLI oder die API verwenden, können Sie mit der `ExtendedS3DestinationConfiguration` eine Amazon-S3-Backup-Konfiguration mit einem eigenen `Prefix` und `ErrorOutputPrefix` angeben.
- Wenn Sie Amazon S3 verwenden AWS-Managementkonsole und das Ziel auf Amazon S3 setzen, verwendet Firehose das `Prefix` und `ErrorOutputPrefix` in der Zielkonfiguration für erfolgreiche bzw. fehlgeschlagene Datensätze. Wenn Sie ein Präfix mithilfe von Ausdrücken angeben, müssen Sie das Fehlerpräfix einschließlich `!{firehose:error-output-type}` angeben.

- Wenn Sie `ExtendedS3DestinationConfiguration` mit der AWS CLI, der API oder, wenn Sie eine angeben CloudFormation, Firehose verwenden`S3BackupConfiguration`, stellt Firehose keinen Standard `ErrorOutputPrefix` bereit.
- Sie können beim Erstellen von `partitionKeyFromLambda` Ausdrücken keine `partitionKeyFromQuery` Namespaces verwenden. ErrorOutputPrefix

Beispielpräfixe

Beispiele für **`Prefix`** und **`ErrorOutputPrefix`**

Eingabe	Ausgewertetes Präfix (10:30 UTC am 27. August 2018)

`Prefix`: Nicht angegeben
 	
`ErrorOutputPrefix` : `myFirehoseFailures/!{firehose:error-output-type}/`
	`Prefix`: `2018/08/27/10`
	
	`ErrorOutputPrefix` : `myFirehoseFailures/processing-failed/`

`Prefix`: `!{timestamp:yyyy/MM/dd}` `ErrorOutputPrefix` : Nicht angegeben	Ungültige Eingabe: `ErrorOutputPrefix` kann nicht Null sein, wenn Präfix Ausdrücke enthält

`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=!`
	`Prefix`: `myPrefix/year=2018/month=07/day=06/hour=23/`
	
	`ErrorOutputPrefix` : `myErrorPrefix/year=2018/month=07/day=06/hour=23/processing-failed`

Eingabe	Ausgewertetes Präfix (10:30 UTC am 27. August 2018)
`{timestamp:MM}/day=!{timesta mp:dd}/hour=!{timestamp:HH}/! {firehose:error-output-type}`	

`Prefix`: `myFirehosePrefix/`
 	
`ErrorOutputPrefix` : Nicht angegeben
	`Prefix`: `myFirehosePrefix/2 018/08/27/`
	
	`ErrorOutputPrefix` : `myFirehos ePrefix/processing-failed/2 018/08/27/`

Konfigurieren Sie die Indexrotation für Service OpenSearch

Für das OpenSearch Serviceziel können Sie eine zeitbasierte Indexrotationsoption aus einer der folgenden fünf Optionen angeben:NoRotation,, OneHour OneDayOneWeek, oder. OneMonth

Abhängig von der von Ihnen gewählten Rotationsoption hängt Amazon Data Firehose einen Teil des UTC-Ankunftszeitstempels an Ihren angegebenen Indexnamen an. Es rotiert den angefügten Zeitstempel entsprechend. Das folgende Beispiel zeigt den resultierenden Indexnamen in OpenSearch Service für jede Indexrotationsoption, wobei sich der angegebene Indexname myindex und der Ankunftszeitstempel befinden. `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**
>
> Mit der `OneWeek`-Option erstellt Data Firehose automatisch Indizes im Format <YEAR>-w <WEEK NUMBER>(z. B. `2020-w33`), wobei die Wochennummer anhand der UTC-Zeit und gemäß den folgenden US-Konventionen berechnet wird:
>
> - Eine Woche beginnt am Sonntag
> - Die erste Woche des Jahres ist die erste Woche, die in diesem Jahr einen Samstag enthält

Die Datenübermittlung unterbrechen und fortsetzen

Nachdem Sie einen Firehose-Stream eingerichtet haben, werden die in der Stream-Quelle verfügbaren Daten kontinuierlich an das Ziel übermittelt. Wenn Sie auf Situationen stoßen, in denen Ihr Stream-Ziel vorübergehend nicht verfügbar ist (z. B. bei geplanten Wartungsarbeiten), sollten Sie die Datenübermittlung vorübergehend unterbrechen und fortsetzen, sobald das Ziel wieder verfügbar ist.

> **Important**
>
> Wenn Sie den unten beschriebenen Ansatz verwenden, um einen Stream anzuhalten und fortzusetzen, werden Sie nach der Wiederaufnahme des Streams feststellen, dass nur wenige Datensätze in den Fehler-Bucket in Amazon S3 zugestellt werden, während der Rest des Streams weiterhin an das Ziel zugestellt wird. Dies ist eine bekannte Einschränkung dieses Ansatzes, die darauf zurückzuführen ist, dass eine kleine Anzahl von Datensätzen, die zuvor nach mehreren Wiederholungen nicht an das Ziel zugestellt werden konnten, als fehlgeschlagen eingestuft werden.

Einen Firehose-Stream anhalten

Um die Stream-Übertragung in Firehose zu unterbrechen, entfernen Sie zunächst die Berechtigungen für Firehose, für fehlgeschlagene Lieferungen in den S3-Backup-Speicherort zu schreiben. Wenn Sie beispielsweise den Firehose-Stream mit einem OpenSearch Ziel pausieren möchten, können Sie dies tun, indem Sie die Berechtigungen aktualisieren. Weitere Informationen finden Sie unter Firehose Access to a Public OpenSearch Service Destination gewähren.

Entfernen Sie die `"Effect": "Allow"`-Berechtigung für die `s3:PutObject`-Aktion und fügen Sie explizit eine Anweisung hinzu, die die `Effect": "Deny"`-Berechtigung auf die `s3:PutObject`-Aktion für den S3-Bucket anwendet, der für die Sicherung fehlgeschlagener Lieferungen verwendet wird. Schalten Sie als Nächstes das Stream-Ziel aus (z. B. indem Sie die OpenSearch Zieldomäne ausschalten) oder entfernen Sie Firehose die Schreibberechtigungen für das Ziel. Um die Berechtigungen für andere Ziele zu aktualisieren, überprüfen Sie den Abschnitt für Ihr Ziel unter Zugriffskontrolle mit Amazon Data Firehose. Nachdem Sie diese beiden Aktionen abgeschlossen haben, stellt Firehose die Bereitstellung von Streams ein, und Sie können dies mithilfe von CloudWatch Metriken für Firehose überwachen.

> **Important**
>
> Wenn Sie die Stream-Übertragung in Firehose unterbrechen, müssen Sie sicherstellen, dass die Quelle des Streams (z. B. in Kinesis Data Streams oder in Managed Service for Kafka) so konfiguriert ist, dass Daten beibehalten werden, bis die Stream-Zustellung wieder aufgenommen wird und die Daten an das Ziel geliefert werden. Wenn die Quelle DirectPut ist, speichert Firehose die Daten 24 Stunden lang. Es kann zu Datenverlusten kommen, wenn Sie den Stream nicht fortsetzen und die Daten nicht vor Ablauf der Datenaufbewahrungsfrist bereitstellen.

Einen Firehose-Stream fortsetzen

Um die Zustellung fortzusetzen, machen Sie zunächst die zuvor am Stream-Ziel vorgenommene Änderung rückgängig, indem Sie das Ziel aktivieren und sicherstellen, dass Firehose über die Berechtigungen verfügt, den Stream an das Ziel zu senden. Machen Sie als Nächstes die zuvor vorgenommenen Änderungen an den Berechtigungen rückgängig, die auf den S3-Bucket angewendet wurden, um fehlgeschlagene Lieferungen zu sichern. Wenden Sie die `"Effect": "Allow"`-Berechtigung für die `s3:PutObject`-Aktion an und entfernen Sie die `"Effect": "Deny"`-Berechtigung für die `s3:PutObject`-Aktion für den S3-Bucket, der für die Sicherung fehlgeschlagener Lieferungen verwendet wird. Überwachen Sie abschließend mithilfe von CloudWatch Metriken für Firehose, ob der Stream an das Ziel geliefert wird. Verwenden Sie Amazon CloudWatch Logs Monitoring for Firehose, um Fehler anzuzeigen und zu beheben.

Liefern Sie Daten mit Amazon Data Firehose an Apache Iceberg Tables

Apache Iceberg ist ein leistungsstarkes Open-Source-Tabellenformat für die Durchführung von Big-Data-Analysen. Apache Iceberg bringt die Zuverlässigkeit und Einfachheit von SQL-Tabellen in Amazon S3 S3-Data Lakes und ermöglicht es Open-Source-Analyse-Engines wie Spark, Flink, Trino, Hive und Impala, gleichzeitig mit denselben Daten zu arbeiten. Weitere Informationen finden Sie unter Apache Iceberg und. Überlegungen und Einschränkungen

Sie können Firehose verwenden, um Streaming-Daten an Apache Iceberg Tables in Amazon S3 zu übermitteln. Ihre Apache Iceberg-Tabellen können in Amazon S3 selbst verwaltet oder in Amazon S3-Tabellen gehostet werden. In selbstverwalteten Iceberg-Tabellen verwalten Sie alle Tabellenoptimierungen wie Komprimierung und Ablauf von Snapshots. Amazon S3 S3-Tabellen bieten Speicher, der für umfangreiche Analyse-Workloads optimiert ist, mit Funktionen, die die Abfrageleistung kontinuierlich verbessern und die Speicherkosten für tabellarische Daten senken. Weitere Informationen zu Amazon S3 S3-Tabellen finden Sie unter Amazon S3 S3-Tabellen.

Mit dieser Funktion können Sie Datensätze aus einem einzelnen Stream in verschiedene Apache Iceberg-Tabellen weiterleiten. Sie können automatisch Einfüge-, Aktualisierungs- und Löschvorgänge auf Datensätze in diesen Tabellen anwenden. Es unterstützt auch eine detaillierte Datenzugriffskontrolle für Apache Iceberg-Tabellen in Amazon S3 mit. AWS Lake Formation Sie können Zugriffskontrollen zentral in AWS Lake Formation Firehose festlegen und detailliertere Berechtigungen auf Tabellen- und Spaltenebene bereitstellen.

Überlegungen und Einschränkungen

> **Note**
>
> Firehose unterstützt Apache Iceberg Tables als Ziel in allen Regionen AWS-Regionenaußer China und im asiatisch-pazifischen Raum (Malaysia). AWS GovCloud (US) Regions

Die Firehose-Unterstützung für Apache Iceberg-Tabellen hat die folgenden Überlegungen und Einschränkungen.

- Durchsatz — Wenn Sie Direct PUT als Quelle für die Bereitstellung von Daten an Apache Iceberg-Tabellen verwenden, beträgt der maximale Durchsatz pro Stream 5 MiB/second in den Regionen

USA Ost (Nord-Virginia), USA West (Oregon) und Europa (Irland) und 1 MiB/second in allen anderen AWS-Regionen Regionen. Wenn Sie Daten ohne Aktualisierungen und Löschungen in Iceberg-Tabellen einfügen und einen höheren Durchsatz für Ihren Stream wünschen, können Sie das Formular Firehose Limits verwenden, um eine Erhöhung des Durchsatzlimits zu beantragen.

Sie können die `AppendOnly` Markierung auch auf setzen, `True` wenn Sie nur Daten einfügen und keine Aktualisierungen und Löschungen durchführen möchten. Wenn Sie die `AppendOnly` Flagge auf setzen`True`, passt sich Firehose automatisch Ihrem Durchsatz an. Derzeit können Sie dieses Flag nur bei der CreateDeliveryStreamAPI-Operation setzen.

Wenn bei einem Direct PUT-Stream eine Drosselung aufgrund höherer Datenaufnahmemengen auftritt, die die Durchsatzkapazität eines Firehose-Streams überschreiten, erhöht Firehose automatisch die Durchsatzgrenze des Streams, bis die Drosselung eingedämmt ist. Je nach erhöhtem Durchsatz und Drosselung kann es länger dauern, bis Firehose den Durchsatz eines Streams auf das gewünschte Niveau erhöht. Aus diesem Grund sollten Sie die fehlgeschlagenen Datenaufnahmesätze erneut versuchen. Wenn Sie erwarten, dass das Datenvolumen bei plötzlichen großen Datenausbrüchen ansteigt, oder wenn Ihr neuer Stream einen höheren Durchsatz als den standardmäßigen Durchsatzgrenzwert benötigt, fordern Sie eine Erhöhung des Durchsatzlimits an.

- S3-Transaktion pro Sekunde (TPS) — Um die S3-Leistung zu optimieren, empfehlen wir, den Quelldatensatz mit einem geeigneten Partitionsschlüssel zu partitionieren, wenn Sie Kinesis Data Streams oder Amazon MSK als Quelle verwenden. Auf diese Weise werden Datensätze, die an dieselbe Iceberg-Tabelle weitergeleitet werden, einer oder mehreren Quellpartitionen zugeordnet, die als Shards bezeichnet werden. Wenn möglich, verteilen Sie Datensätze, die zu verschiedenen Ziel-Iceberg-Tabellen gehören, auf verschiedene. partitions/shards, so that you can use all the aggregate throughput available across all the partitions/shards of the source topic/stream
- Spalten — Für Spaltennamen und Werte verwendet Firehose nur die erste Ebene von Knoten in einem mehrstufigen verschachtelten JSON. Firehose wählt beispielsweise die Knoten aus, die in der ersten Ebene verfügbar sind, einschließlich des Positionsfeldes. Die Spaltennamen und die Datentypen der Quelldaten müssen exakt mit denen der Zieltabellen übereinstimmen, damit Firehose erfolgreich liefern kann. In diesem Fall erwartet Firehose, dass Sie in Ihren Iceberg-Tabellen entweder eine Spalte vom Datentyp Struct oder Map haben, die dem Positionsfeld entspricht. Firehose unterstützt 16 Verschachtelungsebenen. Im Folgenden finden Sie ein Beispiel für ein verschachteltes JSON.

```
{
   "version":"2016-04-01",
```

```
   "deviceId":"<solution_unique_device_id>",
   "sensorId":"<device_sensor_id>",
   "timestamp":"2024-01-11T20:42:45.000Z",
   "value":"<actual_value>",
   "position":{
      "x":143.595901,
      "y":476.399628,
      "z":0.24234876
   }
}
```

Wenn die Spaltennamen oder Datentypen nicht übereinstimmen, gibt Firehose einen Fehler aus und übermittelt Daten an den S3-Fehler-Bucket. Wenn alle Spaltennamen und Datentypen in den Apache Iceberg-Tabellen übereinstimmen, der Quelldatensatz jedoch ein zusätzliches Feld enthält, überspringt Firehose das neue Feld.

- Ein JSON-Objekt pro Datensatz — Sie können nur ein JSON-Objekt in einem Firehose senden. Wenn Sie mehrere JSON-Objekte innerhalb eines Datensatzes aggregieren und senden, gibt Firehose einen Fehler aus und übermittelt Daten an den S3-Fehler-Bucket. Wenn Sie Datensätze mit KPL aggregieren und Daten mit Amazon Kinesis Data Streams als Quelle in Firehose aufnehmen, deaggregiert Firehose automatisch und verwendet ein JSON-Objekt pro Datensatz.
- Komprimierung und Speicheroptimierung — Jedes Mal, wenn Sie mit Firehose in Iceberg-Tabellen schreiben, werden Snapshots, Datendateien und gelöschte Dateien festgeschrieben und generiert. Viele Datendateien erhöhen den Metadaten-Overhead und beeinträchtigen die Leseleistung. Um eine effiziente Abfrageleistung zu erzielen, sollten Sie eine Lösung in Betracht ziehen, bei der in regelmäßigen Abständen kleine Datendateien in weniger größere Datendateien umgeschrieben werden. Dieser Vorgang wird als Komprimierung bezeichnet. AWS Glue Data Catalog unterstützt die automatische Komprimierung Ihrer Apache Iceberg-Tabellen. Weitere Informationen finden Sie unter Verdichtungsmanagement im AWS Glue-Benutzerhandbuch. Weitere Informationen finden Sie unter Automatische Komprimierung von Apache Iceberg-Tabellen. Alternativ können Sie den Befehl Athena Optimize ausführen, um die Komprimierung manuell durchzuführen. Weitere Informationen zum Befehl Optimize finden Sie unter Athena Optimize.

  Neben der Komprimierung von Datendateien können Sie auch den Speicherverbrauch mit der VACUUM-Anweisung optimieren, die Tabellenverwaltung für Apache Iceberg-Tabellen durchführt, z. B. das Ablaufen von Snapshots und das Entfernen verwaister Dateien. Alternativ können Sie AWS Glue Data Catalog damit auch die verwaltete Tabellenoptimierung von Apache Iceberg-Tabellen unterstützen, indem die Datendateien, verwaisten Dateien und abgelaufenen Snapshots,

die nicht mehr benötigt werden, automatisch entfernt werden. Weitere Informationen finden Sie in diesem Blogbeitrag zur Speicheroptimierung von Apache Iceberg-Tabellen.

- Wir unterstützen die Amazon MSK Serverless-Quelle für Apache Iceberg Tables nicht als Ziel.
- Für einen Aktualisierungsvorgang legt Firehose eine Löschdatei gefolgt von einem Einfügevorgang ab. Für das Ablegen gelöschter Dateien fallen Gebühren für Amazon S3 an.
- Firehose empfiehlt nicht, mehrere Firehose-Streams zu verwenden, um Daten in dieselbe Apache Iceberg-Tabelle zu schreiben. Dies liegt daran, dass Apache Iceberg auf Optimistic Concurrency Control (OCC) angewiesen ist. Wenn mehrere Firehose-Streams versuchen, gleichzeitig in eine einzelne Iceberg-Tabelle zu schreiben, kann nur ein Stream die Daten zu einem bestimmten Zeitpunkt erfolgreich festschreiben. Die anderen Streams, bei denen der Commit fehlschlägt, ziehen sich zurück und wiederholen den Commit-Vorgang, bis die konfigurierte Wiederholungsdauer abgelaufen ist. Sobald die Wiederholungsdauer abgelaufen ist, werden die Daten und die Schlüssel zum Löschen von Dateien (Amazon S3 S3-Pfade) an das konfigurierte Amazon S3 S3-Fehlerpräfix gesendet.
- Die aktuelle Version der Iceberg Library, die Firehose unterstützt, ist Version 1.5.2.
- Um verschlüsselte Daten an Amazon S3 S3-Tabellen zu liefern, sollten Sie AWS Key Management Service Parameter in Amazon S3 S3-Tabellen und nicht in der Firehose-Konfiguration konfigurieren. Wenn Sie in Firehose AWS Key Management Service Parameter für die Übertragung verschlüsselter Daten an Amazon S3 S3-Tabellen konfigurieren, kann Firehose diese Parameter nicht zum Verschlüsseln verwenden. Weitere Informationen finden Sie unter Serverseitige Verschlüsselung mit Schlüsseln verwenden. AWS KMS
- Firehose-Streams unterstützen nur die Bereitstellung an Datenbanken und Tabellen, die über die Iceberg-API erstellt wurden. GlueCatalog Die Lieferung an Datenbanken und Tabellen, die mit dem Glue SDK erstellt wurden, wird nicht unterstützt. Beachten Sie, dass ein Bindestrich (-) kein unterstütztes Zeichen für die Datenbank und den Tabellennamen in der Iceberg-Bibliothek ist. Weitere Informationen finden Sie in der Glue-Datenbank-Regex und der Glue-Tabellen-Regex, die von der Iceberg-Bibliothek unterstützt werden.
- Alle von Firehose geschriebenen Dateien werden anhand der Partition berechnet, die im Datensatz vorhanden ist. Dies gilt auch für gelöschte Dateien. Globale Löschungen, wie das Schreiben unpartitionierter Löschdateien für eine partitionierte Tabelle, werden nicht unterstützt.

Voraussetzungen für die Verwendung von Apache Iceberg Tables als Ziel

Wählen Sie aus den folgenden Optionen, um die erforderlichen Voraussetzungen zu erfüllen.

Topics

Voraussetzungen für die Lieferung an Iceberg Tables in Amazon S3

Bevor Sie beginnen, müssen Sie die folgenden Voraussetzungen erfüllen.

- Erstellen Sie einen Amazon S3 S3-Bucket — Sie müssen einen Amazon S3 S3-Bucket erstellen, um bei der Tabellenerstellung einen Metadatendateipfad hinzuzufügen. Weitere Informationen finden Sie unter Einen S3-Bucket erstellen.
- Erstellen Sie eine IAM-Rolle mit den erforderlichen Berechtigungen — Firehose benötigt eine IAM-Rolle mit bestimmten Berechtigungen, um auf AWS Glue Tabellen zuzugreifen und Daten in Amazon S3 zu schreiben. Dieselbe Rolle wird verwendet, um AWS Glue Zugriff auf Amazon S3 S3-Buckets zu gewähren. Sie benötigen diese IAM-Rolle, wenn Sie eine Iceberg-Tabelle und einen Firehose-Stream erstellen. Weitere Informationen finden Sie unter Firehose Zugriff auf Amazon S3 S3-Tabellen gewähren.
- Apache Iceberg-Tabellen erstellen — Wenn Sie eindeutige Schlüssel im Firehose-Stream für Aktualisierungen und Löschungen konfigurieren, überprüft Firehose, ob die Tabelle und die eindeutigen Schlüssel als Teil der Stream-Erstellung existieren. Für dieses Szenario müssen Sie Tabellen erstellen, bevor Sie den Firehose-Stream erstellen. Sie können es verwenden AWS Glue , um Apache Iceberg-Tabellen zu erstellen. Weitere Informationen finden Sie unter Creating Apache Iceberg tables. Wenn Sie keine eindeutigen Schlüssel im Firehose-Stream konfigurieren, müssen Sie keine Iceberg-Tabellen erstellen, bevor Sie einen Firehose-Stream erstellen.

> **Note**
>
> Firehose unterstützt die folgende Tabellenversion und das folgende Format für Apache Iceberg-Tabellen.

- Version im Tabellenformat — Firehose unterstützt nur das V2-Tabellenformat. Erstellen Sie keine Tabellen im V1-Format, da Sie sonst eine Fehlermeldung erhalten und stattdessen Daten an den S3-Fehler-Bucket gesendet werden.
- Datenspeicherformat — Firehose schreibt Daten im Parquet-Format in Apache Iceberg-Tabellen.
- Operation auf Zeilenebene — Firehose unterstützt den Modus Merge-on-Read (MOR) zum Schreiben von Daten in Apache Iceberg-Tabellen.

Voraussetzungen für die Lieferung an Amazon S3 S3-Tabellen

Um Daten an Amazon S3 S3-Tabellen-Buckets zu liefern, müssen Sie die folgenden Voraussetzungen erfüllen.

- Erstellen Sie einen S3-Tabellen-Bucket, einen Namespace, Tabellen im Tabellen-Bucket und andere Integrationsschritte, die unter Erste Schritte mit Amazon S3 S3-Tabellen beschrieben sind. Spaltennamen müssen aufgrund der Einschränkungen, die durch die S3-Tabellen-Katalogintegration auferlegt werden, in Kleinbuchstaben geschrieben werden, wie unter Einschränkungen der S3-Tabellen-Katalogintegration beschrieben.
- Erstellen Sie eine IAM-Rolle mit den erforderlichen Berechtigungen — Firehose benötigt eine IAM-Rolle mit bestimmten Berechtigungen, um auf AWS Glue Tabellen zuzugreifen und Daten in Tabellen in einem Amazon S3 S3-Tabellen-Bucket zu schreiben. Um in Tabellen in einem S3-Tabellen-Bucket zu schreiben, müssen Sie der IAM-Rolle außerdem die erforderlichen Berechtigungen in geben. AWS Lake Formation Sie konfigurieren diese IAM-Rolle, wenn Sie einen Firehose-Stream erstellen. Weitere Informationen finden Sie unter Firehose Zugriff auf Amazon S3 S3-Tabellen gewähren.
- AWS Lake Formation Berechtigungen konfigurieren — AWS Lake Formation verwaltet den Zugriff auf Ihre Tabellenressourcen. Lake Formation verwendet ein eigenes Berechtigungsmodell, das eine differenzierte Zugriffskontrolle für Datenkatalogressourcen ermöglicht.

Informationen zur step-by-step Integration finden Sie im Blog Build a Data Lake for Streaming Data with Amazon S3 Tables and Amazon Data Firehose. Weitere Informationen finden Sie auch unter Amazon S3-Tabellen mit AWS Analysediensten verwenden.

Richten Sie den Firehose-Stream ein

Um einen Firehose-Stream mit Apache Iceberg Tables als Ziel zu erstellen, müssen Sie Folgendes konfigurieren.

> **Note**
>
> Die Einrichtung eines Firehose-Streams für die Lieferung an Tabellen in S3-Tabellen-Buckets entspricht der von Apache Iceberg Tables in Amazon S3.

Quelle und Ziel konfigurieren

Um Daten an Apache Iceberg Tables zu liefern, wählen Sie die Quelle für Ihren Stream aus.

Informationen zur Konfiguration Ihrer Quelle für Ihren Stream finden Sie unter Quelleinstellungen konfigurieren.

Wählen Sie als Nächstes Apache Iceberg Tables als Ziel und geben Sie einen Firehose-Stream-Namen an.

Konfigurieren Sie die Datentransformation

Um benutzerdefinierte Transformationen an Ihren Daten durchzuführen, z. B. Datensätze in Ihrem eingehenden Stream hinzuzufügen oder zu ändern, können Sie Ihrem Firehose-Stream eine Lambda-Funktion hinzufügen. Weitere Informationen zur Datentransformation mit Lambda in einem Firehose-Stream finden Sie unter. Transformieren Sie Quelldaten in Amazon Data Firehose

Für Apache Iceberg-Tabellen müssen Sie angeben, wie eingehende Datensätze an verschiedene Zieltabellen weitergeleitet werden sollen und welche Operationen Sie ausführen möchten. Eine Möglichkeit, Firehose die erforderlichen Routing-Informationen zur Verfügung zu stellen, ist die Verwendung einer Lambda-Funktion.

Weitere Informationen finden Sie unter Datensätze an verschiedene Iceberg-Tabellen weiterleiten.

Datenkatalog Connect

Apache Iceberg benötigt einen Datenkatalog, um in Apache Iceberg-Tabellen schreiben zu können. Firehose lässt sich in Apache Iceberg Tables integrieren. AWS Glue Data Catalog

Sie können es AWS Glue Data Catalog in demselben Konto wie Ihr Firehose-Stream oder in einem kontoübergreifenden und in derselben Region wie Ihr Firehose-Stream (Standard) oder in einer anderen Region verwenden.

Wenn Sie an eine Amazon S3 S3-Tabelle liefern und die Konsole verwenden, um Ihren Firehose-Stream einzurichten, wählen Sie den Katalog aus, der Ihrem Amazon S3 S3-Table-Katalog entspricht. Wenn Sie die CLI verwenden, um Ihren Firehose-Stream einzurichten, verwenden Sie ihn in der `CatalogConfiguration` Eingabe `CatalogARN` mit dem Format:`arn:aws:glue:<region>:<account-id>:catalog/s3tablescatalog/<s3 table bucket name>`. Weitere Informationen finden Sie unter Einen Firehose-Stream zu Amazon S3 S3-Tabellen einrichten.

> **Note**
>
> Firehose unterstützt drei Operationen für Iceberg-Tabellen: Einfügen, Aktualisieren und Löschen. Ohne eine angegebene Operation verwendet Firehose standardmäßig das Einfügen, fügt jeden eingehenden Datensatz als neue Zeile hinzu und behält Duplikate bei. Um stattdessen bestehende Datensätze zu ändern, geben Sie den Vorgang „Update“ an, bei dem Primärschlüssel verwendet werden, um vorhandene Zeilen zu finden und zu ändern. Beispiel:
>
> - Standard (Einfügen): Mehrere identische Kundendatensätze erzeugen doppelte Zeilen.
> - Angegebenes Update: Durch die neue Kundenadresse wird der bestehende Datensatz aktualisiert.

Konfigurieren Sie JQ-Ausdrücke

Für Apache Iceberg-Tabellen müssen Sie angeben, wie eingehende Datensätze an verschiedene Zieltabellen weitergeleitet werden sollen und welche Operationen wie Einfügen, Aktualisieren und Löschen Sie ausführen möchten. Sie können dies tun, indem Sie JQ-Ausdrücke für Firehose konfigurieren, um die erforderlichen Informationen zu analysieren und abzurufen. Weitere Informationen finden Sie unter ???.

Konfigurieren Sie eindeutige Schlüssel

Aktualisierungen und Löschungen mit mehr als einer Tabelle — Eindeutige Schlüssel sind ein oder mehrere Felder in Ihrem Quelldatensatz, die eine Zeile in Apache Iceberg-Tabellen eindeutig

identifizieren. Wenn Sie nur ein Szenario mit mehr als einer Tabelle einfügen haben, müssen Sie keine eindeutigen Schlüssel konfigurieren. Wenn Sie Aktualisierungen und Löschungen an bestimmten Tabellen vornehmen möchten, müssen Sie eindeutige Schlüssel für diese erforderlichen Tabellen konfigurieren. Beachten Sie, dass das Update die Zeile automatisch einfügt, wenn die Zeile in den Tabellen fehlt. Wenn Sie nur eine einzige Tabelle haben, können Sie eindeutige Schlüssel konfigurieren. Für einen Aktualisierungsvorgang legt Firehose eine Löschdatei gefolgt von einer Einfügung ab.

Sie können entweder eindeutige Schlüssel pro Tabelle als Teil der Firehose-Stream-Erstellung konfigurieren oder Sie können sie identifier-field-idsnativ in Iceberg während der Operation „Tabelle erstellen“ oder „Tabelle ändern“ festlegen. Die Konfiguration eindeutiger Schlüssel pro Tabelle während der Stream-Erstellung ist optional. Wenn Sie bei der Stream-Erstellung keine eindeutigen Schlüssel pro Tabelle konfigurieren, `identifier-field-ids` sucht Firehose nach erforderlichen Tabellen und verwendet sie als eindeutige Schlüssel. Wenn beide nicht konfiguriert sind, schlägt die Übertragung von Daten mit Aktualisierungs- und Löschvorgängen fehl.

Um diesen Abschnitt zu konfigurieren, geben Sie den Datenbanknamen, den Tabellennamen und die eindeutigen Schlüssel für die Tabellen an, in denen Sie Daten aktualisieren oder löschen möchten. Sie können in der Konfiguration nur Einträge für jede Tabelle angeben. Sie müssen diesen Abschnitt nicht für reine Append-Szenarien konfigurieren. Optional können Sie auch ein Fehler-Bucket-Präfix angeben, falls Daten aus der Tabelle nicht zugestellt werden können, wie im folgenden Beispiel gezeigt.

```
[
  {
    "DestinationDatabaseName": "MySampleDatabase",
    "DestinationTableName": "MySampleTable",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

Firehose unterstützt die Konfiguration eindeutiger Schlüssel, wenn der angegebene Spaltenname in der gesamten Tabelle eindeutig ist. Vollqualifizierte Spaltennamen als eindeutige Schlüssel werden jedoch nicht unterstützt. Beispielsweise `top._id` wird ein benannter Schlüssel nicht als eindeutiger Schlüssel betrachtet, wenn der Spaltenname auch auf der obersten Ebene vorhanden `_id` ist. Wenn `_id` ein Wert in der gesamten Tabelle eindeutig ist, wird er unabhängig von seiner Position innerhalb

der Tabellenstruktur verwendet. Dabei spielt es keine Rolle, ob es sich um eine Spalte der obersten Ebene oder um eine verschachtelte Spalte handelt. Im folgenden Beispiel `_id` handelt es sich um einen gültigen eindeutigen Schlüssel für das Schema, da der Spaltenname im gesamten Schema eindeutig ist.

```
[
 "schema": {
  "type": "struct",
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "user", "type": "string" }
  ]
}
]
```

Im folgenden Beispiel `_id` ist dies kein gültiger eindeutiger Schlüssel für das Schema, da er sowohl in der Spalte auf oberster Ebene als auch in der verschachtelten Struktur verwendet wird.

```
[
"schema": {
  "type": "struct",
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "_id", "type": "string" }
```

```
  ]
}
]
```

Geben Sie die Dauer des erneuten Versuchs an

Sie können diese Konfiguration verwenden, um die Dauer in Sekunden anzugeben, für die Firehose versuchen soll, es erneut zu versuchen, falls beim Schreiben in Apache Iceberg-Tabellen in Amazon S3 Fehler auftreten. Sie können einen beliebigen Wert zwischen 0 und 7200 Sekunden für die Durchführung von Wiederholungsversuchen festlegen. Standardmäßig versucht Firehose es 300 Sekunden lang erneut.

Behandeln Sie die fehlgeschlagene Lieferung oder Verarbeitung

Sie müssen Firehose so konfigurieren, dass Datensätze an einen S3-Backup-Bucket gesendet werden, falls nach Ablauf der Wiederholungsdauer Fehler bei der Verarbeitung oder Bereitstellung eines Streams auftreten. Konfigurieren Sie dazu das Ausgabepräfix für den S3-Backup-Bucket und den S3-Backup-Bucket-Fehler in den Backup-Einstellungen in der Konsole.

Handhaben von Fehlern

Firehose sendet alle Lieferfehler an CloudWatch Logs und Amazon S3 S3-Fehler-Buckets.

Liste der Fehler:

Fehlermeldung	Beschreibung
`Iceberg.NoSuchTable`	Firehose schreibt in eine Tabelle, die nicht existiert, oder die Tabelle ist nicht im V2-Format . Firehose unterstützt keine Tabellen im V1-Format.
`Iceberg.InvalidTableName`	Ein Null-Tabellenname oder ein leerer Tabellenname wird übergeben, oder die Tabelle ist nicht im V2-Format. Firehose unterstützt keine Tabellen im V1-Format.
`S3.AccessDenied`	Stellen Sie sicher, dass die im Schritt „Voraussetzungen“ erstellte IAM-Rolle über die

Fehlermeldung	Beschreibung
	erforderlichen Berechtigungen und Vertrauen srichtlinien verfügt.
`Glue.AccessDenied`	Stellen Sie sicher, dass die im Schritt „Voraussetzungen“ erstellte IAM-Rolle über die erforderlichen Berechtigungen und die erforderl ichen Vertrauensrichtlinien verfügt.

Pufferhinweise konfigurieren

Firehose puffert eingehende Streaming-Daten im Speicher auf eine bestimmte Größe (Puffergröße) und für einen bestimmten Zeitraum (Pufferintervall), bevor sie an Apache Iceberg Tables übermittelt werden. Sie können eine Puffergröße von 1—128 MiBs und ein Pufferintervall von 0—900 Sekunden wählen. Höhere Pufferhinweise führen zu einer geringeren Anzahl von S3-Schreibvorgängen, geringeren Kosten für die Komprimierung aufgrund größerer Datendateien und einer schnelleren Abfragelaufzeit, jedoch mit einer höheren Latenz. Niedrigere Werte für Pufferhinweise liefern die Daten mit geringerer Latenz.

Konfigurieren von erweiterten Einstellungen

Sie können serverseitige Verschlüsselung, Fehlerprotokollierung, Berechtigungen und Tags für Ihre Apache Iceberg-Tabellen konfigurieren. Weitere Informationen finden Sie unter Konfigurieren von erweiterten Einstellungen. Sie müssen die IAM-Rolle hinzufügen, die Sie als Teil von erstellt haben. ??? Firehose übernimmt die Rolle für den Zugriff auf AWS Glue Tabellen und das Schreiben in Amazon S3 S3-Buckets.

Die Erstellung eines Firehose-Streams kann mehrere Minuten dauern. Nachdem Sie den Firehose-Stream erfolgreich erstellt haben, können Sie damit beginnen, Daten in ihn aufzunehmen und die Daten in Apache Iceberg-Tabellen anzuzeigen.

Leiten Sie eingehende Datensätze an eine einzelne Iceberg-Tabelle weiter

Wenn Sie möchten, dass Firehose Daten in eine einzelne Iceberg-Tabelle einfügt, konfigurieren Sie einfach eine einzelne Datenbank und Tabelle in Ihrer Stream-Konfiguration, wie im folgenden JSON-

Beispiel gezeigt. Für eine einzelne Tabelle benötigen Sie keinen JQ-Ausdruck und keine Lambda-Funktion, um Firehose die Routing-Informationen bereitzustellen. Wenn Sie diese Felder zusammen mit JQ oder Lambda angeben, nimmt Firehose Eingaben von JQ oder Lambda entgegen.

```
[
  {
    "DestinationDatabaseName": "UserEvents",
    "DestinationTableName": "customer_id",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

In diesem Beispiel leitet Firehose alle Eingabedatensätze an eine `customer_id` Tabelle in der `UserEvents` Datenbank weiter. Wenn Sie Aktualisierungs- oder Löschvorgänge für eine einzelne Tabelle ausführen möchten, müssen Sie Firehose die Operation für jeden eingehenden Datensatz entweder mit der JSONQueryMethode oder der Lambda-Methode zur Verfügung stellen.

Leiten Sie eingehende Datensätze an verschiedene Iceberg-Tabellen weiter

Amazon Data Firehose kann eingehende Datensätze in einem Stream basierend auf dem Inhalt des Datensatzes an verschiedene Iceberg-Tabellen weiterleiten. Die Aufzeichnungen werden nicht in der richtigen Reihenfolge aufbewahrt, wenn sie von Amazon Data Firehose geliefert werden. Betrachten Sie den folgenden Beispiel-Eingabedatensatz.

```
{
  "deviceId": "Device1234",
  "timestamp": "2024-11-28T11:30:00Z",
  "data": {
    "temperature": 21.5,
    "location": {
      "latitude": 37.3324,
      "longitude": -122.0311
    }
  },
  "powerlevel": 84,
```

```
  "status": "online"
}
```

```
{
  "deviceId": "Device4567",
  "timestamp": "2023-11-28T10:40:00Z",
  "data": {
    "pressure": 1012.4,
    "location": {
      "zipcode": 24567
    }
  },
  "powerlevel": 82,
  "status": "online"
}
```

In diesem Beispiel hat das **`deviceId`** Feld zwei mögliche Werte — `Device1234` und`Device4567`. Wenn ein eingehender Datensatz das **`deviceId`** Feld als enthält`Device1234`, möchten wir den Datensatz in eine Eisberg-Tabelle mit dem Namen schreiben`Device1234`, und wenn ein eingehender Datensatz das **`deviceId`** Feld als enthält`Device4567`, möchten wir den Datensatz in eine Tabelle mit dem Namen `Device4567` schreiben.

Beachten Sie, dass die Datensätze unterschiedliche Felder enthalten `Device1234` und `Device4567` möglicherweise unterschiedliche Felder haben, die unterschiedlichen Spalten in der entsprechenden Eisberg-Tabelle zugeordnet sind. Die eingehenden Datensätze haben möglicherweise eine verschachtelte JSON-Struktur, in der sie innerhalb des JSON-Datensatzes verschachtelt werden **`deviceId`**können. In den nächsten Abschnitten besprechen wir, wie Sie Datensätze an verschiedene Tabellen weiterleiten können, indem Sie Firehose in solchen Szenarien die entsprechenden Routing-Informationen zur Verfügung stellen.

Stellen Sie Firehose Routing-Informationen mit JSONQuery Ausdruck zur Verfügung

Die einfachste und kostengünstigste Möglichkeit, Firehose Datensatz-Routing-Informationen zur Verfügung zu stellen, ist die Bereitstellung eines JSONQuery Ausdrucks. Bei diesem Ansatz stellen Sie JSONQuery Ausdrücke für drei Parameter bereit — `Database Name``Table Name`, und (optional)`0peration`. Firehose verwendet den von Ihnen angegebenen Ausdruck, um Informationen aus Ihren eingehenden Stream-Datensätzen zu extrahieren, um die Datensätze weiterzuleiten.

Der `Database Name` Parameter gibt den Namen der Zieldatenbank an. Der `Table Name` Parameter gibt den Namen der Zieltabelle an. `Operation`ist ein optionaler Parameter, der angibt, ob der eingehende Stream-Datensatz als neuer Datensatz in die Zieltabelle eingefügt oder ein vorhandener Datensatz in der Zieltabelle geändert oder gelöscht werden soll. Das Feld Operation muss einen der folgenden Werte haben — `insert`, `update`, oder`delete`.

Für jeden dieser drei Parameter können Sie entweder einen statischen Wert oder einen dynamischen Ausdruck angeben, wobei der Wert aus dem eingehenden Datensatz abgerufen wird. Wenn Sie beispielsweise alle eingehenden Stream-Datensätze an eine einzige Datenbank mit dem Namen senden möchten`IoTevents`, hätte der Datenbankname den statischen Wert`"IoTevents"`. Wenn der Name der Zieltabelle aus einem Feld im eingehenden Datensatz abgerufen werden muss, ist der Tabellenname ein dynamischer Ausdruck, der das Feld im eingehenden Datensatz angibt, aus dem der Name der Zieltabelle abgerufen werden muss.

Im folgenden Beispiel verwenden wir einen statischen Wert für den Datenbanknamen, einen dynamischen Wert für den Tabellennamen und einen statischen Wert für den Vorgang. Beachten Sie, dass die Angabe der Operation optional ist. Wenn keine Operation angegeben ist, fügt Firehose die eingehenden Datensätze standardmäßig als neue Datensätze in die Zieltabelle ein.

```
Database Name : "IoTevents"
Table Name : .deviceId
Operation : "insert"
```

Wenn das `deviceId` Feld innerhalb des JSON-Datensatzes verschachtelt ist, geben wir den Tabellennamen mit den verschachtelten Feldinformationen als an. `.event.deviceId`

> **Note**
>
> - Wenn Sie die Operation als `update` oder angeben`delete`, müssen Sie entweder eindeutige Schlüssel für die Zieltabelle angeben, wenn Sie Ihren Firehose-Stream einrichten, oder identifier-field-idsin Iceberg festlegen, wenn Sie Operationen zum Erstellen von Tabellen oder Ändern von Tabellen in Iceberg ausführen. Wenn Sie dies nicht angeben, gibt Firehose einen Fehler aus und übermittelt Daten an einen S3-Fehler-Bucket.
> - Die `Table Name` Werte `Database Name` und müssen exakt mit Ihren Zieldatenbank- und Tabellennamen übereinstimmen. Wenn sie nicht übereinstimmen, gibt Firehose einen Fehler aus und übermittelt Daten an einen S3-Fehler-Bucket.

Stellen Sie Routing-Informationen mithilfe einer Funktion bereit AWS Lambda

Es kann Szenarien geben, in denen Sie über komplexe Regeln verfügen, die bestimmen, wie eingehende Datensätze an eine Zieltabelle weitergeleitet werden. Möglicherweise haben Sie eine Regel, die definiert, ob ein Feld den Wert A, B oder F enthält, der an eine Zieltabelle mit dem Namen weitergeleitet werden soll, `TableX` oder Sie möchten den Datensatz für den Eingangsstream erweitern, indem Sie zusätzliche Attribute hinzufügen. Wenn ein Datensatz beispielsweise ein Feld `device_id` als 1 enthält, möchten Sie vielleicht ein weiteres Feld mit dem Namen „Modem“ hinzufügen und das zusätzliche Feld in die Spalte der Zieltabelle schreiben. `device_type` In solchen Fällen können Sie den Quellstream mithilfe einer AWS Lambda Funktion in Firehose transformieren und Routing-Informationen als Teil der Ausgabe der Lambda-Transformationsfunktion bereitstellen. Informationen dazu, wie Sie den Quellstream mithilfe einer AWS Lambda Funktion in Firehose transformieren können, finden Sie unter Transformieren von Quelldaten in Amazon Data Firehose.

Wenn Sie Lambda für die Transformation eines Quellstreams in Firehose verwenden, muss die Ausgabe die Parameter `recordIdresult`, und `data` oder `KafkaRecordValue` enthalten. Der Parameter `recordId` enthält den Eingabestream-Datensatz, `result` gibt an, ob die Transformation erfolgreich war, und `data` enthält die Base64-kodierte transformierte Ausgabe Ihrer Lambda-Funktion. Weitere Informationen finden Sie unter ???.

```
{
  "recordId": "49655962066601463032522589543535113056108699331451682818000000",
  "result": "Ok",
  "data": "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1"
}
```

Um Firehose Routing-Informationen darüber zu geben, wie der Stream-Datensatz als Teil Ihrer Lambda-Funktion an eine Zieltabelle weitergeleitet werden soll, muss die Ausgabe Ihrer Lambda-Funktion einen zusätzlichen Abschnitt für enthalten. `metadata` Das folgende Beispiel zeigt, wie der Metadatenabschnitt zur Lambda-Ausgabe für einen Firehose-Stream hinzugefügt wird, der Kinesis Data Streams als Datenquelle verwendet, um Firehose anzuweisen, den Datensatz als neuen Datensatz in die Tabelle mit dem Namen der Datenbank einzufügen. `Device1234` `IoTevents`

```
 {
"recordId": "49655962066601463032522589543535113056108699331451682818000000",
    "result": "Ok",
```

```
    "data":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",

    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
    }
  }
```

In ähnlicher Weise zeigt das folgende Beispiel, wie Sie den Metadatenabschnitt zur Lambda-Ausgabe für eine Firehose hinzufügen können, die Amazon Managed Streaming for Apache Kafka als Datenquelle verwendet, um Firehose anzuweisen, den Datensatz als neuen Datensatz in eine in der Datenbank benannte Tabelle einzufügen. `Device1234` `IoTevents`

```
{
"recordId": "496559620666014630325225895435351130561086993314516828180000000",
    "result": "Ok",
    "kafkaRecordValue":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",

    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
    }
  }
```

Für dieses Beispiel

- `destinationDatabaseName`bezieht sich auf den Namen der Zieldatenbank und ist ein Pflichtfeld.
- `destinationTableName`bezieht sich auf den Namen der Zieltabelle und ist ein Pflichtfeld.
- `operation`ist ein optionales Feld mit möglichen Werten wie `insert`update, und`delete`. Wenn Sie keine Werte angeben, ist die Standardoperation`insert`.

Note

- Wenn Sie die Operation als `update` oder angeben`delete`, müssen Sie entweder eindeutige Schlüssel für die Zieltabelle angeben, wenn Sie Ihren Firehose-Stream einrichten, oder identifier-field-idsin Iceberg festlegen, wenn Sie Operationen zum Erstellen von Tabellen oder Ändern von Tabellen in Iceberg ausführen. Wenn Sie dies nicht angeben, gibt Firehose einen Fehler aus und übermittelt Daten an einen S3-Fehler-Bucket.
- Die `Table Name` Werte `Database Name` und müssen exakt mit Ihren Zieldatenbank- und Tabellennamen übereinstimmen. Wenn sie nicht übereinstimmen, gibt Firehose einen Fehler aus und übermittelt Daten an einen S3-Fehler-Bucket.
- Wenn Ihr Firehose-Stream sowohl eine Lambda-Transformationsfunktion als auch einen JSONQuery Ausdruck enthält, sucht Firehose zunächst nach dem Metadatenfeld in der Lambda-Ausgabe, um festzustellen, wie der Datensatz an die entsprechende Zieltabelle weitergeleitet werden soll, und sucht dann in der Ausgabe Ihres JSONQuery Ausdrucks nach fehlenden Feldern.

  Wenn das Lambda oder der JSONQuery Ausdruck nicht die erforderlichen Routing-Informationen bereitstellt, geht Firehose davon aus, dass es sich um ein Einzeltabellenszenario handelt, und sucht in der Konfiguration der eindeutigen Schlüssel nach Einzeltabelleninformationen.

  Weitere Informationen finden Sie unter Eingehende Datensätze an eine einzelne Iceberg-Tabelle weiterleiten. Wenn Firehose die Routing-Informationen nicht ermitteln und den Datensatz nicht mit einer bestimmten Zieltabelle abgleichen kann, werden die Daten an Ihren angegebenen S3-Fehler-Bucket gesendet.

Beispiel-Lambda-Funktion

Diese Lambda-Funktion ist ein Python-Beispielcode, der die eingehenden Stream-Datensätze analysiert und erforderliche Felder hinzufügt, um anzugeben, wie die Daten in bestimmte Tabellen geschrieben werden sollen. Sie können diesen Beispielcode verwenden, um den Metadatenbereich für Routing-Informationen hinzuzufügen.

```
import json
import base64
```

```
def lambda_handler(firehose_records_input, context):
    print("Received records for processing from DeliveryStream: " +
 firehose_records_input['deliveryStreamArn'])

    firehose_records_output = {}
    firehose_records_output['records'] = []


    for firehose_record_input in firehose_records_input['records']:

        # Get payload from Lambda input, it could be different with different sources
        if 'kafkaRecordValue' in firehose_record_input:
            payload_bytes =
 base64.b64decode(firehose_record_input['kafkaRecordValue']).decode('utf-8')
        else
            payload_bytes =
 base64.b64decode(firehose_record_input['data']).decode('utf-8')

        # perform data processing on customer payload bytes here

        # Create output with proper record ID, output data (may be different with
 different sources), result, and metadata
        firehose_record_output = {}

        if 'kafkaRecordValue' in firehose_record_input:
            firehose_record_output['kafkaRecordValue'] =
 base64.b64encode(payload_bytes.encode('utf-8'))
        else
            firehose_record_output['data'] =
 base64.b64encode(payload_bytes.encode('utf-8'))

        firehose_record_output['recordId'] = firehose_record_input['recordId']
        firehose_record_output['result'] =  'Ok'
        firehose_record_output['metadata'] = {
            'otfMetadata': {
                'destinationDatabaseName': 'your_destination_database',
                'destinationTableName': 'your_destination_table',
                'operation': 'insert'
                }
            }
        firehose_records_output['records'].append(firehose_record_output)
    return firehose_records_output
```

Überwachen von Metriken

Für die Datenlieferung an Apache Iceberg Tables gibt Firehose die folgenden CloudWatch Metriken auf Stream-Ebene aus.

Metrik	Description
`DeliveryToIceberg.Bytes`	Die Anzahl der Byte, die im angegebenen Zeitraum an Apache Iceberg Tables geliefert wurden. Einheiten: Byte
`DeliveryToIceberg.IncomingRowCount`	Anzahl der Datensätze, die Firehose versucht, an Apache Iceberg Tables zu liefern. Einheiten: Anzahl
`DeliveryToIceberg.SuccessfulRowCount`	Anzahl der erfolgreichen Zeilen, die an Apache Iceberg-Tabellen übermittelt wurden. Einheiten: Anzahl
`DeliveryToIceberg.FailedRowCount`	Anzahl der fehlgeschlagenen Zeilen, die an den S3-Backup-Bucket übermittelt wurden. Einheiten: Anzahl
`DeliveryToIceberg.DataFreshness`	Das Alter (von den Anfängen bei Firehose bis heute) der frühesten Aufzeichnungen in Firehose. Jeder Datensatz , der vor diesem Alter liegt, wurde an Apache Iceberg Tables übermittelt. Einheiten: Sekunden
`DeliveryToIceberg.Success`	Summe der erfolgreichen Commits für Apache Iceberg Tables.
`JQProcessing.Duration`	Die Zeit, die für die Ausführung des JQ-Ausdrucks benötigt wurde.

Metrik	Description
	Einheiten: Millisekunden

Verstehen Sie die unterstützten Datentypen

Firehose unterstützt alle primitiven und komplexen Datentypen, die Apache Iceberg unterstützt. Weitere Informationen finden Sie unter Schemas und Datentypen. Wenn Sie Binärdaten als Zeichenfolge senden, müssen Sie die von Firehose unterstützten Kodierungstypen verwenden: Basic Base64, MIME Base64, URL and filename safe Base64 und Hex. Bei Timestamp-Datentypen müssen Sie immer in Mikrosekunden senden.

Beispiele für Datentypen

Der folgende Abschnitt zeigt Beispiele für verschiedene Datentypen.

MapType

```
{
    "destination_column_0":
 {"WP5o0JOkuIQcDPcsvpJJygF1xzaOSq0wUlgTwuIeCEzgVneGxA":"PO3ReF3auyDqbfonx9Cd8NTmcQnqnw7JuZOCWwI
    "destination_column_1": "{\"{\\\"destination_nested_column_0\\\": \
\\"18:56:14.974\\\", \\\"destination_nested_column_1\\\": 241.86246}\":
\"M07kAvYdHvBh61F7RzfxtEd39YQI33LnM2NbGS67DOFFsRUyUUujKT5VnK7Wtfz1mHNeIix6FAY9cYpwTdedgr9XnFwG0
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 562.56384}\":
\"9GlxhDCt95LxBo51HybBZihqOqf6EU8jrDu7NMpxtGB2dY6q6kXpvxIrFuMdqHCJKIZIcDikwggLniUm8kgE4d
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 496.03268}\":
\"keTJZYLNvLRB50DMKzEI6M0AM4mueyNnA1m2YVnYdDwyxUpPqkb72Q6LiX0B9s8gCjZ6trW6C1PFk9KNBIpxYsj5Tc5Xs
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\\\", \\
\"destination_nested_column_1\\\": 559.0878}\":
\"mG0ZET84BUF28E312UCIWgmypyQFSUODH9NAMAnF3LJEutbooZWcBt97PP5AHaopNvC8pQZ4mGXB9hmVmjUNmuj5Qanyx
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 106.845245}\":
\"aidoVYrzu8gcLRkVVUyTKCN9gqTUFYi8uJQsrXEFEYl1f9ool7JhAtg9QKG5BBu67Ngb95ENsNKQyCHNImSu5x4hMnmHU
\"}"
}
```

DecimalType

```
{
    "destination_column_0": 9455262425851.1342772,
    "destination_column_1": "9455262425851.1342772",
    "destination_column_2": 9455262425852
}
```

BinaryType (base64-default, base64-mime, base64-url-safe, hex)

```
{
    "destination_column_0": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y
+k5cMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA
+CBAWXfIuiG420JSQP5iAx5xFG\/
mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAXg
+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK8j
+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/
r3MEqoEqt+nPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY
\/8Bvy+4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r
\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b9ydjR9z+fCSdG
+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX\/
W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF\/Tv2DUub465hzgpaLPur3+MB
+kfdN2YXUfqb
+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo
+bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwi
\/kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\/
pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/
fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx
+im7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL
+gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm
+NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_1": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y+k5c\r
\nMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA+CBAWXfI\r
\nuiG420JSQP5iAx5xFG\/mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6R\r
\nzdJlHhUDjoAXg+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK\r
\n8j+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/r3MEqoEqt+
\r\nnPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY\/8Bvy+4byoPlmr4Z5yhN1z
\r\n3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b
\r\n9ydjR9z+fCSdG+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZ
\r\ntyiZOaTGIj9rO0xX\/W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF
\/\r\nTv2DUub465hzgpaLPur3+MB+kfdN2YXUfqb+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvP
\r\nRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo+bjtai4ndRCGcuCaxT8KkOcXs\r
\nS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwidzDU\/k\r
```

```
\nFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\r
\n\/pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/fgr9jsTtxDg
\r\nZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx+\r
\nim7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL+gGNHFKDRL6wGIfhuYc
\r\nx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm+NO\r
\n5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_2": "AsYhnHD_Ra54hITl1daNV9glOjtWPEfopH-
PjgUKHYB6K7UcYi4K19b80wD4J_93x5tyh-0y-k5cMljVRlmfIkIuLxl9ERBiPPLhf4-
yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA-
CBAWXfIuiG420JSQP5iAx5xFG_mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAX
qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno-LYF5ZsySs2rB5AbVM73RfOPqdS_c_r3MEqoEqt-
nPx6eGam4WSA-0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY_8Bvy-4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv-
vDVsDBtItVazDwHgDy41r_hQNeNedPKrozc8TY9k7wZre_6V2lCa3BmT8Uu9b9ydjR9z-fCSdG-
VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX_W5dGe9_4YChs6LbD
MB-kfdN2YXUfqb-
xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7-
yJwCB8qxhTTryxo-bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8-
EYmcUIVB9XcNdke4zoUfhVQoruwidzDU_kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDS
Jq0U_jQ99GShlU283nZ85ot2ocbtMAgD_WsrSEh6lNt9RaI3HfA7_HcH_fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7
im7mte1sprf1-A24kksVU_MD9aP9N8_QDsQ13gkhOn5KwFMz3BC2Vw5gL-
gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm-
NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89-Rw==",
    "destination_column_3":
 "02c6219c70ff45ae788484e5d5d68d57d8253a3b563c47e8a47f8f8e050a1d807a2bb51c622e0ad7d6fcd300f827f
}
```


TimeType (Epoche in Mikrosekunden, Java-Objekt) LocalTime

```
{
    "destination_column_0": 68175096000,
    "destination_column_1": "18:56:15.096"
}
```

TimestampType.withZone (Epoche in Mikrosekunden, Java-Objekt, OffsetDateTime Java-Objekt) LocalDateTime

```
{
    "destination_column_0": 1725476175099000,
    "destination_column_1": "2024-09-04T18:56:15.099Z",
    "destination_column_2": "2024-09-04T18:56:15.099"
}
```

DoubleType

```
{
    "destination_column_0": 9.18477568715142,
    "destination_column_1": "9.18477568715142"
}
```

BooleanType

```
{
    "destination_column_0": true,
    "destination_column_1": "false",
    "destination_column_2": 1,
    "destination_column_3": 0
}
```

FloatType

```
{
    "destination_column_0": 0.6242226,
    "destination_column_1": "0.6242226"
}
```

IntegerType

```
{
    "destination_column_0": 7,
    "destination_column_1": "7"
}
```

TimestampType.withoutZone (Epoche in Mikrosekunden, Java-Objekt, Java-Objekt, Java-Objekt) LocalDateTime OffsetDateTime ZonedDateTime

```
{
    "destination_column_0": 1725476175114000,
    "destination_column_1": "2024-09-04T18:56:15.114",
    "destination_column_2": "2024-09-04T18:56:15.114Z",
    "destination_column_3": "2024-09-04T18:56:15.114-07:00"
}
```

DateType

```
{
    "destination_column_0": 19970,
    "destination_column_1": "2024-09-04"
}
```

LongType

```
{
    "destination_column_0": 8,
    "destination_column_1": "8"
}
```

UUIDType (UUID-Java-Objekt)

```
{
    "destination_column_0": "21c5521c-a6d4-48d4-b2c8-7f6d842f72c3"
}
```

ListType

```
{
    "destination_column_0":
 ["s1FSrgb0lGDxfn2iYT0EtlP47aHSjwmLZgrdr1JqRs0dmbeCcQoaLr4Xhi2KIVvmus9ppFdpWIcOHnJ0omhAPhXH0yns
    "destination_column_1": "[{\"destination_nested_column_0\":\"bb00f8e6-
db82-4241-a5c5-0d9c0d2f71a4\",\"destination_nested_column_1\":907.35345},
{\"destination_nested_column_0\":\"2c77b702-d405-4fe1-beee-fb541d7ab833\",
\"destination_nested_column_1\":544.0026},{\"destination_nested_column_0\":
\"68389200-d6b1-413d-bcd9-fdb931708395\",\"destination_nested_column_1\":153.683},
{\"destination_nested_column_0\":\"bc31cbaa-39cd-4e2f-b357-9ea9ce75532b\",
\"destination_nested_column_1\":977.5165},{\"destination_nested_column_0\":
\"b7d627f9-0d5b-41b7-903a-525488259fba\",\"destination_nested_column_1\":434.17215},
{\"destination_nested_column_0\":\"06b6ec1e-1952-4582-b285-46aaf40064b8\",
\"destination_nested_column_1\":580.33124},{\"destination_nested_column_0\":
\"f04b3bbf-61ad-4c5c-8740-6f666f57c431\",\"destination_nested_column_1\":550.75793}]"
}
```

Ressourcen

Verwenden Sie die folgenden Ressourcen, um mehr zu erfahren:

- Streamen Sie Echtzeitdaten mit Amazon Data Firehose in Apache Iceberg-Tabellen in Amazon S3
- Optimieren AWS WAF Sie die Protokollanalyse mit Apache Iceberg und Amazon Data Firehose
- Erstellen Sie mit Amazon S3-Tabellen und Amazon Data Firehose einen Data Lake für Streaming-Daten

Einen Firehose-Stream taggen

Sie können Firehose-Streams, die Sie in Amazon Data Firehose erstellen, Ihre eigenen Metadaten in Form von Tags zuweisen. Ein Tag ist ein Schlüssel-Wert-Paar, das Sie für einen Stream definieren. Die Verwendung von Tags ist eine einfache und dennoch leistungsstarke Möglichkeit, AWS Ressourcen zu verwalten und Daten, einschließlich Rechnungsdaten, zu organisieren.

Sie können Tags angeben, wenn Sie aufrufen, CreateDeliveryStreamum einen neuen Firehose-Stream zu erstellen. Für bestehende Firehose-Streams können Sie Tags mithilfe der folgenden drei Operationen hinzufügen, auflisten und entfernen:

- TagDeliveryStream
- ListTagsForDeliveryStream
- UntagDeliveryStream

Verstehen Sie die Grundlagen von Tags

Sie können die API-Operationen von Amazon Data Firehose verwenden, um die folgenden Aufgaben zu erledigen:

- Fügen Sie einem Firehose-Stream Tags hinzu.
- Listet die Tags für Ihre Firehose-Streams auf.
- Entfernen Sie Tags aus einem Firehose-Stream.

Sie können Tags verwenden, um Ihre Firehose-Streams zu kategorisieren. Sie können Firehose-Streams beispielsweise nach Zweck, Eigentümer oder Umgebung kategorisieren. Da Sie den Schlüssel und den Wert für jedes Tag definieren, können Sie einen benutzerdefinierten Satz von Kategorien erstellen, der Ihren spezifischen Anforderungen entspricht. Sie könnten beispielsweise eine Reihe von Tags definieren, mit deren Hilfe Sie Firehose-Streams nach Eigentümer und zugehöriger Anwendung verfolgen können.

Im Folgenden sehen Sie verschiedene Beispiele für Tags:

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`

- `Application: Application name`
- `Environment: Production`

Wenn Sie in der `CreateDeliveryStream` Aktion Tags angeben, führt Amazon Data Firehose eine zusätzliche Autorisierung für die `firehose:TagDeliveryStream` Aktion durch, um zu überprüfen, ob Benutzer berechtigt sind, Tags zu erstellen. Wenn Sie diese Berechtigung nicht erteilen, schlagen Anfragen zum Erstellen neuer Firehose-Streams mit IAM-Ressourcen-Tags fehl, und zwar mit einem `AccessDeniedException` solchen Fehler wie dem Folgenden.

```
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.
```

Das folgende Beispiel zeigt eine Richtlinie, die es Benutzern ermöglicht, einen Firehose-Stream zu erstellen und Tags anzuwenden.

Verfolgen Sie die Kosten mit Tagging

Sie können Tags verwenden, um Ihre AWS Kosten zu kategorisieren und nachzuverfolgen. Wenn Sie Tags auf Ihre AWS Ressourcen anwenden, einschließlich Firehose-Streams, enthält Ihr AWS Kostenzuordnungsbericht die Nutzung und die Kosten, die nach Tags zusammengefasst sind. Sie können die Kosten für mehrere Services organisieren, indem Sie Tags anwenden, die geschäftliche Kategorien (wie Kostenstellen, Anwendungsnamen oder Eigentümer) darstellen. Weitere Informationen finden Sie unter Verwenden von Kostenzuordnungs-Tags für benutzerdefinierte Fakturierungsberichte im AWS Billing -Benutzerhandbuch.

Kennen Sie die Einschränkungen von Tags

Die folgenden Einschränkungen gelten für Tags in Amazon Data Firehose.

Grundlegende Einschränkungen

- Die maximale Anzahl an Tags pro Ressource (Stream) beträgt 50.
- Bei Tag-Schlüsseln und -Werten wird zwischen Groß- und Kleinschreibung unterschieden.
- Sie können Tags für einen gelöschten Stream nicht ändern oder bearbeiten.

Einschränkungen für Tag-Schlüssel

- Jeder Tag-Schlüssel muss einmalig sein. Wenn Sie ein Tag mit einem Schlüssel hinzufügen, der bereits verwendet wird, überschreibt Ihr neues Tag das bestehende Schlüssel-Wert-Paar.
- Sie können einen Tag-Schlüssel nicht mit `aws:` beginnen, da dieses Präfix für die Verwendung durch AWS reserviert ist. AWS erstellt in Ihrem Namen Tags, die mit diesem Präfix beginnen, Sie können diese jedoch nicht bearbeiten oder löschen.
- Tag-Schlüssel müssen zwischen 1 und 128 Unicode-Zeichen lang sein.
- Tag-Schlüssel müssen die folgenden Zeichen enthalten: Unicode-Zeichen, Ziffern, Leerzeichen sowie die folgenden Sonderzeichen: `_ . / = + - @`.

Einschränkungen für den Tag-Wert

- Tag-Werte müssen zwischen 0 und 255 Unicode-Zeichen lang sein.
- Tag-Werte können leer sein. Ansonsten müssen sie die folgenden Zeichen enthalten: Unicode-Zeichen, Ziffern, Leerzeichen und eines der folgenden Sonderzeichen: `_ . / = + - @`.

Sicherheit bei Amazon Data Firehose

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die auf die Anforderungen der sicherheitssensibelsten Unternehmen zugeschnitten sind.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS -Compliance-Programms getestet und überprüft. Weitere Informationen zu den Compliance-Programmen, die für Data Firehose gelten, finden Sie unter AWS Services im Umfang nach Compliance-Programmen.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von Data Firehose anwenden können. In den folgenden Themen erfahren Sie, wie Sie Data Firehose konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, die Ihnen bei der Überwachung und Sicherung Ihrer Data Firehose-Ressourcen helfen können.

Topics

Datenschutz bei Amazon Data Firehose

Amazon Data Firehose verschlüsselt alle Daten während der Übertragung mithilfe des TLS-Protokolls. Darüber hinaus verschlüsselt Amazon Data Firehose Daten, die während der Verarbeitung im Zwischenspeicher gespeichert werden, mithilfe einer Prüfsummenüberprüfung AWS Key Management Serviceund überprüft deren Integrität.

Wenn Sie vertrauliche Daten haben, können Sie die serverseitige Datenverschlüsselung aktivieren, wenn Sie Amazon Data Firehose verwenden. Wie Sie dazu vorgehen, hängt von der Quelle Ihrer Daten ab.

> **Note**
>
> Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Serverseitige Verschlüsselung mit Kinesis Data Streams

Wenn Sie Daten von Ihren Datenproduzenten an Ihren Datenstream senden, verschlüsselt Kinesis Data Streams Ihre Daten mit einem AWS Key Management Service (AWS KMS) -Schlüssel, bevor die Daten im Ruhezustand gespeichert werden. Wenn Ihr Firehose-Stream die Daten aus Ihrem Datenstream liest, entschlüsselt Kinesis Data Streams zuerst die Daten und sendet sie dann an Amazon Data Firehose. Amazon Data Firehose puffert die Daten im Speicher auf der Grundlage der von Ihnen angegebenen Pufferhinweise. Dann liefert es sie ans Ziel, ohne die unverschlüsselten Daten im Ruhezustand zu speichern.

Informationen zur Aktivierung der serverseitigen Verschlüsselung für Kinesis Data Streams finden Sie unter Verwenden der serverseitigen Verschlüsselung im Entwicklerhandbuch zu Amazon Kinesis Data Streams.

Serverseitige Verschlüsselung mit Direct PUT oder anderen Datenquellen

Wenn Sie Daten mit oder an Ihren Firehose-Stream senden PutRecordBatch, PutRecordoder wenn Sie die Daten mithilfe von AWS IoT Amazon CloudWatch Logs oder CloudWatch Events senden, können Sie die serverseitige Verschlüsselung mithilfe des StartDeliveryStreamEncryptionVorgangs aktivieren.

Verwenden Sie den Vorgang server-side-encryption, um den StopDeliveryStreamEncryptionVorgang zu beenden.

Sie können SSE auch aktivieren, wenn Sie den Firehose erstellen. Geben Sie dazu an, DeliveryStreamEncryptionConfigurationInputwann Sie aufrufen CreateDeliveryStream.

Wenn das CMK vom Typ `CUSTOMER_MANAGED_CMK` ist und der Amazon Data Firehose-Service aufgrund von a`KMSNotFoundException`, a, a oder a keine Datensätze entschlüsseln kann`KMSDisabledException`, wartet der Service bis zu 24 Stunden (die Aufbewahrungsfrist)`KMSAccessDeniedException`, bis Sie das Problem behoben haben. `KMSInvalidStateException` Wenn das Problem über den Aufbewahrungszeitraum hinaus fortbesteht, überspringt der Service die Datensätze, die den Aufbewahrungszeitraum überschritten haben und nicht entschlüsselt werden konnten, und verwirft dann die Daten. Amazon Data Firehose bietet die folgenden vier CloudWatch Metriken, mit denen Sie die vier AWS KMS Ausnahmen verfolgen können:

- `KMSKeyAccessDenied`
- `KMSKeyDisabled`
- `KMSKeyInvalidState`
- `KMSKeyNotFound`

Weitere Informationen zu diesen vier Metriken finden Sie unter the section called “Überwachung mit Metriken CloudWatch ”.

> **Important**
>
> Verwenden Sie symmetrisch, um Firehose Firehose-Stream zu verschlüsseln. CMKs Amazon Data Firehose unterstützt keine Asymmetrie CMKs. Informationen zu symmetrisch und asymmetrisch finden Sie unter About Symmetric CMKs and Asymmetric im Entwicklerhandbuch. CMKs AWS Key Management Service

> **Note**
>
> Wenn Sie einen vom Kunden verwalteten Schlüssel (CUSTOMER_MANAGED_CMK) verwenden, um die serverseitige Verschlüsselung (SSE) für Ihren Firehose-Stream zu aktivieren, legt der Firehose-Dienst bei jeder Verwendung Ihres Schlüssels einen

Verschlüsselungskontext fest. Da dieser Verschlüsselungskontext ein Ereignis darstellt, bei dem ein Schlüssel verwendet wurde, der Ihrem AWS Konto gehört, wird er als Teil der Ereignisprotokolle für Ihr Konto protokolliert. AWS CloudTrail AWS Dieser Verschlüsselungskontext ist ein vom Firehose-Dienst generiertes System. Ihre Anwendung sollte keine Annahmen über das Format oder den Inhalt des vom Firehose-Dienst festgelegten Verschlüsselungskontextes treffen.

Zugriffskontrolle mit Amazon Data Firehose

In den folgenden Abschnitten wird beschrieben, wie Sie den Zugriff auf und von Ihren Amazon Data Firehose-Ressourcen kontrollieren können. Zu den Informationen, die sie behandeln, gehört, wie Sie Ihrer Anwendung Zugriff gewähren können, damit sie Daten an Ihren Firehose-Stream senden kann. Sie beschreiben auch, wie Sie Amazon Data Firehose Zugriff auf Ihren Amazon Simple Storage Service (Amazon S3) -Bucket, Amazon Redshift Redshift-Cluster oder Amazon OpenSearch Service-Cluster gewähren können, sowie die Zugriffsberechtigungen, die Sie benötigen, wenn Sie Datadog, Dynatrace,, MongoDB, New Relic LogicMonitor, Splunk oder Sumo Logic als Ziel verwenden. Schließlich finden Sie in diesem Thema Anleitungen zur Konfiguration von Amazon Data Firehose, sodass Daten an ein Ziel gesendet werden können, das zu einem anderen AWS Konto gehört. Die Technologie zur Verwaltung all dieser Zugriffsformen ist AWS Identity and Access Management (IAM). Weitere Informationen zu IAM finden Sie unter Was ist IAM?.

Inhalt

- Gewähren Sie Firehose Zugriff auf ein OpenSearch serverloses Ziel in einer VPC
- Firehose Zugriff auf ein Splunk-Ziel gewähren
- Zugreifen auf Splunk in VPC
- VPC-Flow-Logs mithilfe von Amazon Data Firehose in Splunk aufnehmen
- Zugreifen auf Snowflake oder den HTTP-Endpunkt
- Firehose Zugriff auf ein Snowflake-Ziel gewähren
- Zugreifen auf Snowflake in VPC
- Firehose Zugriff auf ein HTTP-Endpunktziel gewähren
- Kontenübergreifender Versand von Amazon MSK
- Kontoübergreifende Lieferung an ein Amazon S3 S3-Ziel
- Kontoübergreifende Lieferung an ein Serviceziel OpenSearch
- Verwendung von Tags zur Zugriffssteuerung

Gewähren Sie Zugriff auf Ihre Firehose-Ressourcen

Verwenden Sie eine Richtlinie, die diesem Beispiel ähnelt, um Ihrer Anwendung Zugriff auf Ihren Firehose-Stream zu gewähren. Sie können die einzelnen API-Operationen, für die Sie Zugriff gewähren, anpassen, indem Sie den Abschnitt `Action` ändern oder Zugriff auf alle Operationen mit `"firehose:*"` gewähren.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DeleteDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:UpdateDestination"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/delivery-
stream-name"
```

```
            ]
        }
    ]
}
```

Gewähren Sie Firehose Zugriff auf Ihren privaten Amazon MSK-Cluster

Wenn die Quelle Ihres Firehose-Streams ein privater Amazon MSK-Cluster ist, verwenden Sie eine Richtlinie, die diesem Beispiel ähnelt.

Sie müssen der ressourcenbasierten Richtlinie des Clusters eine solche Richtlinie hinzufügen, um dem Firehose Service Principal die Erlaubnis zu erteilen, den Amazon MSK-API-Vorgang aufzurufen. `CreateVpcConnection`

Erlauben Sie Firehose, eine IAM-Rolle anzunehmen

In diesem Abschnitt werden die Berechtigungen und Richtlinien beschrieben, die Amazon Data Firehose Zugriff auf die Erfassung, Verarbeitung und Übertragung von Daten von der Quelle bis zum Ziel gewähren.

> **Note**
>
> Wenn Sie die Konsole verwenden, um einen Firehose-Stream zu erstellen, und die Option zum Erstellen einer neuen Rolle wählen, wird die erforderliche Vertrauensrichtlinie an die Rolle AWS angehängt. Wenn Sie möchten, dass Amazon Data Firehose eine bestehende IAM-Rolle verwendet, oder wenn Sie selbst eine Rolle erstellen, fügen Sie dieser Rolle die folgenden Vertrauensrichtlinien hinzu, damit Amazon Data Firehose sie übernehmen kann. Bearbeiten Sie die Richtlinien, um sie durch Ihre AWS Konto-ID zu *`account-id`* ersetzen. Weitere Informationen zum Ändern der Vertrauensstellung einer Rolle finden Sie unter Ändern einer Rolle.

Amazon Data Firehose verwendet eine IAM-Rolle für alle Berechtigungen, die der Firehose-Stream zur Verarbeitung und Bereitstellung von Daten benötigt. Stellen Sie sicher, dass die folgenden Vertrauensrichtlinien mit dieser Rolle verknüpft sind, damit Amazon Data Firehose sie übernehmen kann.

Wenn Sie Amazon MSK als Quelle für Ihren Firehose-Stream wählen, müssen Sie eine andere IAM-Rolle angeben, die Amazon Data Firehose Berechtigungen zum Ingestieren von Quelldaten

aus dem angegebenen Amazon MSK-Cluster gewährt. Stellen Sie sicher, dass die folgenden Vertrauensrichtlinien mit dieser Rolle verknüpft sind, damit Amazon Data Firehose sie übernehmen kann.

JSON

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

Stellen Sie sicher, dass diese Rolle, die Amazon Data Firehose Berechtigungen zum Ingestieren von Quelldaten aus dem angegebenen Amazon MSK-Cluster erteilt, die folgenden Berechtigungen gewährt:

Gewähren Sie Firehose Zugriff auf AWS Glue für die Datenformatkonvertierung

Wenn Ihr Firehose-Stream eine Datenformatkonvertierung durchführt, verweist Amazon Data Firehose auf Tabellendefinitionen, die in gespeichert sind. AWS Glue Um Amazon Data Firehose den erforderlichen Zugriff zu gewähren AWS Glue, fügen Sie Ihrer Richtlinie die folgende Erklärung hinzu. Informationen darüber, wie Sie den ARN der Tabelle finden, finden Sie unter AWS Glue-Ressource angeben ARNs.

```
{
    "Sid": "",
    "Effect": "Allow",
```

```
    "Action": [
        "glue:GetTable",
        "glue:GetTableVersion",
        "glue:GetTableVersions"
    ],
    "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/b",
        "arn:aws:glue:us-east-1:123456789012:table/b/easd"
    ]
},
{
      actions: ['glue:GetSchemaVersion'],
      grantee: options.role,
      resourceArns: ['*'],
}
```

Die empfohlene Richtlinie zum Abrufen von Schemas aus der Schemaregistrierung hat keine Ressourceneinschränkungen. Weitere Informationen finden Sie in den IAM-Beispielen für Deserializer im Developer Guide. AWS Glue

Firehose Zugriff auf ein Amazon S3 S3-Ziel gewähren

Wenn Sie ein Amazon S3 S3-Ziel verwenden, liefert Amazon Data Firehose Daten an Ihren S3-Bucket und kann optional einen AWS KMS Schlüssel, den Sie besitzen, für die Datenverschlüsselung verwenden. Wenn die Fehlerprotokollierung aktiviert ist, sendet Amazon Data Firehose auch Fehler bei der Datenübermittlung an Ihre CloudWatch Protokollgruppe und Ihre Streams. Sie benötigen eine IAM-Rolle, wenn Sie einen Firehose-Stream erstellen. Amazon Data Firehose übernimmt diese IAM-Rolle und erhält Zugriff auf den angegebenen Bucket, den Schlüssel, die CloudWatch Protokollgruppe und die Streams.

Verwenden Sie die folgende Zugriffsrichtlinie, damit Amazon Data Firehose auf Ihren S3-Bucket und AWS KMS -Schlüssel zugreifen kann. Wenn Sie nicht Eigentümer des S3-Buckets sind, fügen Sie `s3:PutObjectAcl` der Liste der Amazon-S3-Aktionen hinzu. Dadurch erhält der Bucket-Besitzer vollen Zugriff auf die von Amazon Data Firehose gelieferten Objekte.

JSON

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

```
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

Die oben genannte Richtlinie enthält auch eine Erklärung, die den Zugriff auf Amazon Kinesis Data Streams ermöglicht. Wenn Sie keine Kinesis Data Streams als Datenquelle verwenden, können Sie diese Erklärung entfernen. Wenn Sie Amazon MSK als Quelle verwenden, können Sie diese Aussage durch Folgendes ersetzen:

```
{
   "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}}/*"
}
```

Weitere Informationen darüber, wie Sie anderen AWS Diensten den Zugriff auf Ihre AWS Ressourcen ermöglichen, finden Sie unter Creating a Role to Delegate Permissions to an AWS Service im IAM-Benutzerhandbuch.

Informationen darüber, wie Sie Amazon Data Firehose Zugriff auf ein Amazon S3 S3-Ziel in einem anderen Konto gewähren, finden Sie unterthe section called “Kontoübergreifende Lieferung an ein Amazon S3 S3-Ziel”.

Firehose Zugriff auf Amazon S3 S3-Tabellen gewähren

Sie müssen über eine IAM-Rolle verfügen, bevor Sie einen Firehose-Stream erstellen können. Gehen Sie wie folgt vor, um eine Richtlinie und eine IAM-Rolle zu erstellen. Firehose übernimmt diese IAM-Rolle und führt die erforderlichen Aktionen aus.

Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/

Erstellen Sie eine Richtlinie und wählen Sie im Richtlinieneditor JSON aus. Fügen Sie die folgende Inline-Richtlinie hinzu, die Amazon S3 read/write S3-Berechtigungen wie Berechtigungen, Berechtigungen zum Aktualisieren der Tabelle im Datenkatalog und andere gewährt.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "S3TableAccessViaGlueFederation",
      "Effect": "Allow",
      "Action": [
        "glue:GetTable",
        "glue:GetDatabase",
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog/*",
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog",
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/*",
        "arn:aws:glue:us-east-1:123456789012:table/*/*"
      ]
    },
    {
      "Sid": "S3DeliveryErrorBucketPermission",
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<error delivery bucket>",
        "arn:aws:s3:::<error delivery bucket>/*"
      ]
    },
    {
```

```
      "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStream",
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:ListShards"
      ],
      "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/<stream-name>"
    },
    {
      "Sid":
"RequiredWhenDoingMetadataReadsANDDataAndMetadataWriteViaLakeformation",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredWhenUsingKMSEncryptionForS3ErrorBucketDelivery",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:123456789012:key/<KMS-key-id>"
      ],
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.us-east-1.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::<error delivery
bucket>/prefix*"
        }
      }
    },
    {
      "Sid": "LoggingInCloudWatch",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
```

```
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name>:log-
stream:<log-stream-name>"
      ]
    },
    {
      "Sid": "RequiredWhenAttachingLambdaToFirehose",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:us-east-1:123456789012:function:<function-
name>:<function-version>"
      ]
    }
  ]
}
```

Die Richtlinie enthält Anweisungen, die den Zugriff auf Amazon Kinesis Data Streams, das Aufrufen von Lambda-Funktionen und den Zugriff auf Schlüssel ermöglichen. AWS KMS Wenn Sie keine dieser Ressourcen verwenden, können Sie die entsprechenden Anweisungen entfernen. Wenn die Fehlerprotokollierung aktiviert ist, sendet Amazon Data Firehose auch Fehler bei der Datenübermittlung an Ihre CloudWatch Protokollgruppe und Ihre Streams. Sie müssen die Namen der Protokollgruppen und der Protokolldatenströme konfigurieren, um diese Option verwenden zu können. Die Namen von Protokollgruppen und Protokolldatenströmen finden Sie unterÜberwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch .

Ersetzen Sie in den Inline-Richtlinien `<error delivery bucket>` durch Ihren Amazon S3 S3-Bucket-Namen `aws-account-id` und Region durch eine gültige AWS-Konto Nummer und Region der Ressource.

Nachdem Sie die Richtlinie erstellt haben, öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/und erstellen Sie eine IAM-Rolle mit dem AWS-ServiceEntitätstyp Vertrauenswürdig.

Wählen Sie für Service or use case (Service oder Anwendungsfall) die Option Kinesis aus. Wählen Sie als Anwendungsfall Kinesis Firehose aus.

Wählen Sie auf der nächsten Seite die im vorherigen Schritt erstellte Richtlinie aus, die dieser Rolle zugewiesen werden soll. Auf der Überprüfungsseite finden Sie eine Vertrauensrichtlinie, die dieser Rolle bereits zugeordnet ist und dem Firehose-Dienst die Erlaubnis gibt, diese Rolle zu übernehmen. Wenn Sie die Rolle erstellen, kann Amazon Data Firehose davon ausgehen, dass sie die erforderlichen Operationen an AWS Glue und S3-Buckets ausführt. Fügen Sie den Firehose-Dienstprinzipal zur Vertrauensrichtlinie der erstellten Rolle hinzu. Weitere Informationen finden Sie unter Erlauben Sie Firehose, eine IAM-Rolle anzunehmen.

Firehose Zugriff auf ein Apache Iceberg Tables-Ziel gewähren

Sie müssen über eine IAM-Rolle verfügen, bevor Sie einen Firehose-Stream und Apache Iceberg-Tabellen mit erstellen können. AWS Glue Gehen Sie wie folgt vor, um eine Richtlinie und eine IAM-Rolle zu erstellen. Firehose übernimmt diese IAM-Rolle und führt die erforderlichen Aktionen aus.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Erstellen Sie eine Richtlinie und wählen Sie im Richtlinien-Editor JSON aus.
3. Fügen Sie die folgende Inline-Richtlinie hinzu, die Amazon S3 read/write S3-Berechtigungen wie Berechtigungen, Berechtigungen zum Aktualisieren der Tabelle im Datenkatalog usw. gewährt.

   Diese Richtlinie enthält eine Anweisung, die den Zugriff auf Amazon Kinesis Data Streams, das Aufrufen von Lambda-Funktionen und den Zugriff auf KMS-Schlüssel ermöglicht. Wenn Sie keine dieser Ressourcen verwenden, können Sie die entsprechenden Anweisungen entfernen.

   Wenn die Fehlerprotokollierung aktiviert ist, sendet Firehose auch Datenübermittlungsfehler an Ihre CloudWatch Protokollgruppe und Ihre Streams. Dazu müssen Sie die Namen der Protokollgruppen und der Protokolldatenströme konfigurieren. Die Namen von Protokollgruppen und Protokolldatenströmen finden Sie unterÜberwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch .
4. Ersetzen Sie in den Inline-Richtlinien *amzn-s3-demo-bucket* durch Ihren Amazon S3 S3-Bucket-Namen aws-account-id und Region durch eine gültige AWS-Konto Nummer und Region der Ressourcen.

> **Note**
>
> Diese Rolle erteilt Berechtigungen für alle Datenbanken und Tabellen in Ihrem Datenkatalog. Wenn Sie möchten, können Sie nur bestimmten Tabellen und Datenbanken Berechtigungen erteilen.

5. Nachdem Sie die Richtlinie erstellt haben, öffnen Sie die IAM-Konsole und erstellen Sie eine IAM-Rolle mit dem AWS-ServiceEntitätstyp Vertrauenswürdig.
6. Wählen Sie für Service or use case (Service oder Anwendungsfall) die Option Kinesis aus. Wählen Sie für Use case (Anwendungsfall) die Option Kinesis Firehose aus.
7. Wählen Sie auf der nächsten Seite die im vorherigen Schritt erstellte Richtlinie aus, die Sie dieser Rolle zuordnen möchten. Auf der Überprüfungsseite finden Sie eine Vertrauensrichtlinie, die dieser Rolle bereits zugeordnet ist und dem Firehose-Dienst die Erlaubnis gibt, diese Rolle zu übernehmen. Wenn Sie die Rolle erstellen, kann Amazon Data Firehose davon ausgehen, dass sie die erforderlichen Operationen an AWS Glue und S3-Buckets ausführt.

Firehose Zugriff auf ein Amazon Redshift Redshift-Ziel gewähren

Beachten Sie Folgendes, wenn Sie Zugriff auf Amazon Data Firehose gewähren, wenn Sie ein Amazon Redshift Redshift-Ziel verwenden.

Themen

- IAM-Rolle und Zugriffsrichtlinie
- VPC-Zugriff auf einen von Amazon Redshift bereitgestellten Cluster oder eine Amazon Redshift Serverless-Arbeitsgruppe

IAM-Rolle und Zugriffsrichtlinie

Wenn Sie ein Amazon Redshift Redshift-Ziel verwenden, liefert Amazon Data Firehose Daten an Ihren S3-Bucket als Zwischenstandort. Es kann optional einen AWS KMS Schlüssel, den Sie besitzen, für die Datenverschlüsselung verwenden. Amazon Data Firehose lädt dann die Daten aus dem S3-Bucket in Ihren von Amazon Redshift bereitgestellten Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe. Wenn die Fehlerprotokollierung aktiviert ist, sendet Amazon Data Firehose auch Fehler bei der Datenübermittlung an Ihre CloudWatch Protokollgruppe und Ihre Streams. Amazon Data Firehose verwendet den angegebenen Amazon Redshift-Benutzernamen und das

angegebene Passwort für den Zugriff auf Ihren bereitgestellten Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe und verwendet eine IAM-Rolle, um auf den angegebenen Bucket, den Schlüssel, die Protokollgruppe und die Streams zuzugreifen. CloudWatch Sie benötigen eine IAM-Rolle, wenn Sie einen Firehose-Stream erstellen.

Verwenden Sie die folgende Zugriffsrichtlinie, damit Amazon Data Firehose auf Ihren S3-Bucket und AWS KMS -Schlüssel zugreifen kann. Wenn Sie den S3-Bucket nicht besitzen, fügen Sie `s3:PutObjectAcl` ihn der Liste der Amazon S3 S3-Aktionen hinzu, wodurch der Bucket-Besitzer vollen Zugriff auf die von Amazon Data Firehose bereitgestellten Objekte erhält. Diese Richtlinie enthält auch eine Erklärung, die den Zugriff auf Amazon Kinesis Data Streams ermöglicht. Wenn Sie keine Kinesis Data Streams als Datenquelle verwenden, können Sie diese Erklärung entfernen.

JSON

```
{
"Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
```

```
}
```

Weitere Informationen darüber, wie Sie anderen AWS Diensten den Zugriff auf Ihre AWS Ressourcen ermöglichen, finden Sie unter Creating a Role to Delegate Permissions to an AWS Service im IAM-Benutzerhandbuch.

VPC-Zugriff auf einen von Amazon Redshift bereitgestellten Cluster oder eine Amazon Redshift Serverless-Arbeitsgruppe

Wenn Ihr Amazon-Redshift-Cluster oder Ihre Arbeitsgruppe von Amazon Redshift Serverless sich in einer Virtual Private Cloud (VPC) befindet, muss er mit einer öffentlichen IP-Adresse öffentlich zugänglich sein. Gewähren Sie Amazon Data Firehose außerdem Zugriff auf Ihren von Amazon Redshift bereitgestellten Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe, indem Sie die Amazon Data Firehose-IP-Adressen entsperren. Amazon Data Firehose verwendet derzeit einen CIDR-Block für jede verfügbare Region.

Region	CIDR-Blöcke
US East (Ohio)	`13.58.135.96/27`
USA Ost (Nord-Virginia)	`52.70.63.192/27`
USA West (Nordkali fornien)	`13.57.135.192/27`
USA West (Oregon)	`52.89.255.224/27`
AWS GovCloud (US-Ost)	`18.253.138.96/27`
AWS GovCloud (US-West)	`52.61.204.160/27`
Kanada (Zentral)	`35.183.92.128/27`
Kanada West (Calgary)	`40.176.98.192/27`
Asien-Pazifik (Hongkong)	`18.162.221.32/27`
Asien-Pazifik (Mumbai)	`13.232.67.32/27`

Region	CIDR-Blöcke
Asien-Pazifik (Hyderabad)	18.60.192.128/27
Asien-Pazifik (Seoul)	13.209.1.64/27
Asien-Pazifik (Singapur)	13.228.64.192/27
Asien-Pazifik (Sydney)	13.210.67.224/27
Asien-Pazifik (Jakarta)	108.136.221.64/27
Asien-Pazifik (Tokio)	13.113.196.224/27
Asien-Pazifik (Osaka)	13.208.177.192/27
Asien-Pazifik (Thailand)	43.208.112.96/27
Asien-Pazifik (Taipeh)	43.212.53.160/27
China (Peking)	52.81.151.32/27
China (Ningxia)	161.189.23.64/27
Europa (Zürich)	16.62.183.32/27
Europe (Frankfurt)	35.158.127.160/27
Europa (Irland)	52.19.239.192/27
Europa (London)	18.130.1.96/27
Europe (Paris)	35.180.1.96/27
Europa (Stockholm)	13.53.63.224/27
Europa (Spain)	18.100.71.96/27
Middle East (Bahrain)	15.185.91.0/27
Mexiko (Zentral)	78.12.207.32/27

Region	CIDR-Blöcke
Südamerika (São Paulo)	`18.228.1.128/27`
Europa (Milan)	`15.161.135.128/27`
Afrika (Kapstadt)	`13.244.121.224/27`
Naher Osten (VAE)	`3.28.159.32/27`
Israel (Tel Aviv)	`51.16.102.0/27`
Asien-Pazifik (Melbourne)	`16.50.161.128/27`
Asien-Pazifik (Malaysia)	`43.216.58.0/27`

Weitere Informationen zum Entsperren von IP-Adressen finden Sie unter dem Schritt Autorisieren des Zugriffs auf den Cluster im Benutzerhandbuch zu Erste Schritte mit Amazon Redshift.

Firehose Zugang zu einem Ziel des öffentlichen OpenSearch Dienstes gewähren

Wenn Sie ein OpenSearch Serviceziel verwenden, liefert Amazon Data Firehose Daten an Ihren OpenSearch Service-Cluster und sichert gleichzeitig fehlgeschlagene oder alle Dokumente in Ihrem S3-Bucket. Wenn die Fehlerprotokollierung aktiviert ist, sendet Amazon Data Firehose auch Fehler bei der Datenübermittlung an Ihre CloudWatch Protokollgruppe und Ihre Streams. Amazon Data Firehose verwendet eine IAM-Rolle, um auf die angegebene OpenSearch Service-Domain, den S3-Bucket, den AWS KMS Schlüssel und die CloudWatch Protokollgruppe und die Streams zuzugreifen. Sie benötigen eine IAM-Rolle, wenn Sie einen Firehose-Stream erstellen.

Verwenden Sie die folgende Zugriffsrichtlinie, um Amazon Data Firehose den Zugriff auf Ihren S3-Bucket, Ihre OpenSearch Service-Domain und Ihren AWS KMS Schlüssel zu ermöglichen. Wenn Sie den S3-Bucket nicht besitzen, fügen Sie `s3:PutObjectAcl` ihn der Liste der Amazon S3 S3-Aktionen hinzu, wodurch der Bucket-Besitzer vollen Zugriff auf die von Amazon Data Firehose bereitgestellten Objekte erhält. Diese Richtlinie enthält auch eine Erklärung, die den Zugriff auf Amazon Kinesis Data Streams ermöglicht. Wenn Sie keine Kinesis Data Streams als Datenquelle verwenden, können Sie diese Erklärung entfernen.

Weitere Informationen darüber, wie Sie anderen AWS Diensten den Zugriff auf Ihre AWS Ressourcen ermöglichen, finden Sie unter Creating a Role to Delegate Permissions to an AWS Service im IAM-Benutzerhandbuch.

Informationen darüber, wie Sie Amazon Data Firehose Zugriff auf einen OpenSearch Service-Cluster in einem anderen Konto gewähren, finden Sie unterthe section called “Kontoübergreifende Lieferung an ein Serviceziel OpenSearch ”.

Gewähren Sie Firehose Zugriff auf ein OpenSearch Serviceziel in einer VPC

Wenn sich Ihre OpenSearch Service-Domain in einer VPC befindet, stellen Sie sicher, dass Sie Amazon Data Firehose die im vorherigen Abschnitt beschriebenen Berechtigungen erteilen. Darüber hinaus müssen Sie Amazon Data Firehose die folgenden Berechtigungen erteilen, damit Amazon Data Firehose auf die VPC Ihrer OpenSearch Service-Domain zugreifen kann.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Widerrufen Sie diese Berechtigungen nicht, nachdem Sie den Firehose-Stream erstellt haben. Wenn Sie diese Berechtigungen widerrufen, wird Ihr Firehose-Stream beeinträchtigt oder es werden keine Daten mehr an Ihre OpenSearch Dienstdomäne gesendet, wenn der Dienst versucht, eine Abfrage oder Aktualisierung durchzuführen. ENIs

> **Important**
>
> Wenn Sie Subnetze für die Übertragung von Daten an das Ziel in einer privaten VPC angeben, stellen Sie sicher, dass Sie über genügend freie IP-Adressen in den ausgewählten Subnetzen verfügen. Wenn in einem bestimmten Subnetz keine kostenlose IP-Adresse verfügbar ist, kann Firehose die Datenlieferung in der privaten VPC nicht erstellen oder hinzufügen ENIs , und die Lieferung wird beeinträchtigt oder schlägt fehl.

Wenn Sie Ihren Firehose-Stream erstellen oder aktualisieren, geben Sie eine Sicherheitsgruppe an, die Firehose verwenden soll, wenn es Daten an Ihre OpenSearch Service-Domain sendet. Sie können dieselbe Sicherheitsgruppe verwenden, die die OpenSearch Service-Domain verwendet, oder eine andere. Wenn Sie eine andere Sicherheitsgruppe angeben, stellen Sie sicher, dass sie ausgehenden HTTPS-Verkehr zur Sicherheitsgruppe der OpenSearch Dienstdomäne zulässt. Stellen Sie außerdem sicher, dass die Sicherheitsgruppe der OpenSearch Service-Domain HTTPS-Verkehr von der Sicherheitsgruppe zulässt, die Sie bei der Konfiguration Ihres Firehose-Streams angegeben haben. Wenn Sie dieselbe Sicherheitsgruppe sowohl für Ihren Firehose-Stream als auch für die OpenSearch Service-Domain verwenden, stellen Sie sicher, dass die Sicherheitsgruppenregel für eingehenden Datenverkehr HTTPS-Verkehr zulässt. Weitere Informationen zu den Regeln der Sicherheitsgruppe finden Sie unter Sicherheitsgruppenregeln im Amazon-VPC-Benutzerhandbuch.

Gewähren Sie Firehose Zugriff auf ein öffentliches OpenSearch serverloses Ziel

Wenn Sie ein OpenSearch serverloses Ziel verwenden, liefert Amazon Data Firehose Daten an Ihre OpenSearch serverlose Sammlung und sichert gleichzeitig fehlgeschlagene oder alle Dokumente in Ihrem S3-Bucket. Wenn die Fehlerprotokollierung aktiviert ist, sendet Amazon Data Firehose auch Fehler bei der Datenübermittlung an Ihre CloudWatch Protokollgruppe und Ihre Streams. Amazon Data Firehose verwendet eine IAM-Rolle, um auf die angegebene OpenSearch serverlose Sammlung, den S3-Bucket, den AWS KMS Schlüssel und die CloudWatch Protokollgruppe und die angegebenen Streams zuzugreifen. Sie benötigen eine IAM-Rolle, wenn Sie einen Firehose-Stream erstellen.

Verwenden Sie die folgende Zugriffsrichtlinie, um Amazon Data Firehose den Zugriff auf Ihren S3-Bucket, Ihre OpenSearch Serverless-Domain und AWS KMS Ihren Schlüssel zu ermöglichen. Wenn Sie den S3-Bucket nicht besitzen, fügen Sie `s3:PutObjectAcl` ihn der Liste der Amazon S3 S3-Aktionen hinzu, wodurch der Bucket-Besitzer vollen Zugriff auf die von Amazon Data Firehose

bereitgestellten Objekte erhält. Diese Richtlinie enthält auch eine Erklärung, die den Zugriff auf Amazon Kinesis Data Streams ermöglicht. Wenn Sie keine Kinesis Data Streams als Datenquelle verwenden, können Sie diese Erklärung entfernen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        },
        {
         "Effect": "Allow",
         "Action": "aoss:APIAccessAll",
         "Resource": "arn:aws:aoss:us-east-1:123456789012:collection/collection-
id"
      }
    ]
}
```

Zusätzlich zu der oben genannten Richtlinie müssen Sie Amazon Data Firehose auch so konfigurieren, dass Ihnen in einer Datenzugriffsrichtlinie die folgenden Mindestberechtigungen zugewiesen werden:

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/target-collection/target-index"
            ],
            "Permission":[
               "aoss:WriteDocument",
               "aoss:UpdateIndex",
               "aoss:CreateIndex"
            ]
         }
      ],
      "Principal":[
         "arn:aws:sts::123456789012:assumed-role/firehose-delivery-role-name/*"
      ]
   }
]
```

Weitere Informationen darüber, wie Sie anderen AWS Diensten den Zugriff auf Ihre AWS Ressourcen ermöglichen, finden Sie unter Creating a Role to Delegate Permissions to an AWS Service im IAM-Benutzerhandbuch.

Gewähren Sie Firehose Zugriff auf ein OpenSearch serverloses Ziel in einer VPC

Wenn sich Ihre OpenSearch serverlose Sammlung in einer VPC befindet, stellen Sie sicher, dass Sie Amazon Data Firehose die im vorherigen Abschnitt beschriebenen Berechtigungen erteilen. Darüber hinaus müssen Sie Amazon Data Firehose die folgenden Berechtigungen erteilen, damit Amazon Data Firehose auf die VPC Ihrer OpenSearch serverlosen Sammlung zugreifen kann.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`

- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Widerrufen Sie diese Berechtigungen nicht, nachdem Sie den Firehose-Stream erstellt haben. Wenn Sie diese Berechtigungen widerrufen, wird Ihr Firehose-Stream beeinträchtigt oder es werden keine Daten mehr an Ihre OpenSearch Dienstdomäne gesendet, wenn der Dienst versucht, eine Abfrage oder Aktualisierung durchzuführen. ENIs

> **Important**
>
> Wenn Sie Subnetze für die Übertragung von Daten an das Ziel in einer privaten VPC angeben, stellen Sie sicher, dass Sie über genügend freie IP-Adressen in den ausgewählten Subnetzen verfügen. Wenn in einem bestimmten Subnetz keine kostenlose IP-Adresse verfügbar ist, kann Firehose die Datenlieferung in der privaten VPC nicht erstellen oder hinzufügen ENIs , und die Lieferung wird beeinträchtigt oder schlägt fehl.

Wenn Sie Ihren Firehose-Stream erstellen oder aktualisieren, geben Sie eine Sicherheitsgruppe an, die Firehose verwenden soll, wenn es Daten an Ihre OpenSearch Serverless-Sammlung sendet. Sie können dieselbe Sicherheitsgruppe verwenden, die die OpenSearch Serverless-Sammlung verwendet, oder eine andere. Wenn Sie eine andere Sicherheitsgruppe angeben, stellen Sie sicher, dass sie ausgehenden HTTPS-Verkehr zur Sicherheitsgruppe der OpenSearch Serverless-Sammlung zulässt. Stellen Sie außerdem sicher, dass die Sicherheitsgruppe der OpenSearch Serverless Collection HTTPS-Verkehr von der Sicherheitsgruppe zulässt, die Sie bei der Konfiguration Ihres Firehose-Streams angegeben haben. Wenn Sie dieselbe Sicherheitsgruppe sowohl für Ihren Firehose-Stream als auch für die OpenSearch Serverless-Sammlung verwenden, stellen Sie sicher, dass die Sicherheitsgruppenregel für eingehenden Datenverkehr HTTPS-

Verkehr zulässt. Weitere Informationen zu den Regeln der Sicherheitsgruppe finden Sie unter Sicherheitsgruppenregeln im Amazon-VPC-Benutzerhandbuch.

Firehose Zugriff auf ein Splunk-Ziel gewähren

Wenn Sie ein Splunk-Ziel verwenden, liefert Amazon Data Firehose Daten an Ihren Splunk HTTP Event Collector (HEC) -Endpunkt. Außerdem werden diese Daten in dem von Ihnen angegebenen Amazon S3 S3-Bucket gesichert, und Sie können optional einen AWS KMS Schlüssel, den Sie besitzen, für die serverseitige Amazon S3 S3-Verschlüsselung verwenden. Wenn die Fehlerprotokollierung aktiviert ist, sendet Firehose Datenübermittlungsfehler an Ihre CloudWatch Protokollstreams. Sie können es auch AWS Lambda für die Datentransformation verwenden.

Wenn Sie einen Load AWS Balancer verwenden, stellen Sie sicher, dass es sich um einen Classic Load Balancer oder einen Application Load Balancer handelt. Aktivieren Sie außerdem dauerbasierte Sticky-Sitzungen mit deaktiviertem Cookie-Ablauf für Classic Load Balancer und mit maximaler Ablaufzeit (7 Tage) für Application Load Balancer. Informationen dazu finden Sie unter Duration-Based Session Stickiness für Classic Load Balancer oder einen Application Load Balancer.

Sie müssen über eine IAM-Rolle verfügen, wenn Sie einen Firehose erstellen. Firehose nimmt diese IAM-Rolle an und erhält Zugriff auf den angegebenen Bucket, den Schlüssel, die CloudWatch Protokollgruppe und die Streams.

Verwenden Sie die folgende Zugriffsrichtlinie, um Amazon Data Firehose den Zugriff auf Ihren S3-Bucket zu ermöglichen. Wenn Sie den S3-Bucket nicht besitzen, fügen Sie `s3:PutObjectAcl` ihn der Liste der Amazon S3 S3-Aktionen hinzu, wodurch der Bucket-Besitzer vollen Zugriff auf die von Amazon Data Firehose bereitgestellten Objekte erhält. Diese Richtlinie gewährt Amazon Data Firehose auch Zugriff auf die CloudWatch Fehlerprotokollierung und die AWS Lambda Datentransformation. Die Richtlinie enthält auch eine Erklärung, die den Zugriff auf Amazon Kinesis Data Streams ermöglicht. Wenn Sie keine Kinesis Data Streams als Datenquelle verwenden, können Sie diese Erklärung entfernen. Amazon Data Firehose verwendet IAM nicht für den Zugriff auf Splunk. Für den Zugriff auf Splunk, verwendet es Ihr HEC-Token.

JSON

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

```
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
        {
           "Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
           ],
           "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
```

```
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
           ],
           "Resource": [
               "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
           ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "lambda:InvokeFunction",
               "lambda:GetFunctionConfiguration"
           ],
           "Resource": [
               "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
           ]
        }
    ]
}
```

Weitere Informationen darüber, wie Sie anderen AWS Diensten den Zugriff auf Ihre AWS Ressourcen ermöglichen, finden Sie unter Creating a Role to Delegate Permissions to an AWS Service im IAM-Benutzerhandbuch.

Zugreifen auf Splunk in VPC

Wenn sich Ihre Splunk-Plattform in einer VPC befindet, muss sie mit einer öffentlichen IP-Adresse öffentlich zugänglich sein. Gewähren Sie Amazon Data Firehose außerdem Zugriff auf Ihre Splunk-Plattform, indem Sie die Amazon Data Firehose-IP-Adressen entsperren. Amazon Data Firehose verwendet derzeit die folgenden CIDR-Blöcke.

Region	CIDR-Blöcke
US East (Ohio)	`18.216.68.160/27, 18.216.170.64/27, 18.216.17 0.96/27 \`

Region	CIDR-Blöcke
USA Ost (Nord-Virginia)	34.238.188.128/26, 34.238.188.192/26, 34.238.195.0/26
USA West (Nordkalifornien)	13.57.180.0/26
USA West (Oregon)	34.216.24.32/27, 34.216.24.192/27, 34.216.24.224/27
AWS GovCloud (US-Ost)	18.253.138.192/26
AWS GovCloud (US-West)	52.61.204.192/26
Asien-Pazifik (Hongkong)	18.162.221.64/26
Asien-Pazifik (Mumbai)	13.232.67.64/26
Asia Pacific (Seoul)	13.209.71.0/26
Asien-Pazifik (Singapur)	13.229.187.128/26
Asien-Pazifik (Sydney)	13.211.12.0/26
Asien-Pazifik (Thailand)	43.208.112.128/26
Asien-Pazifik (Tokio)	13.230.21.0/27, 13.230.21.32/27
Canada (Central)	35.183.92.64/26
Kanada West (Calgary)	40.176.98.128/26
Europe (Frankfurt)	18.194.95.192/27, 18.194.95.224/27, 18.195.48.0/27
Europa (Irland)	34.241.197.32/27, 34.241.197.64/27, 34.241.197.96/27
Europa (London)	18.130.91.0/26

Region	CIDR-Blöcke
Europa (Paris)	35.180.112.0/26
Europa (Spain)	18.100.194.0/26
Europe (Stockholm)	13.53.191.0/26
Middle East (Bahrain)	15.185.91.64/26
Mexiko (Zentral)	78.12.207.64/26
Südamerika (São Paulo)	18.228.1.192/26
Europa (Milan)	15.161.135.192/26
Afrika (Kapstadt)	13.244.165.128/26
Asien-Pazifik (Osaka)	13.208.217.0/26
China (Peking)	52.81.151.64/26
China (Ningxia)	161.189.23.128/26
Asien-Pazifik (Jakarta)	108.136.221.128/26
Naher Osten (VAE)	3.28.159.64/26
Israel (Tel Aviv)	51.16.102.64/26
Europa (Zürich)	16.62.183.64/26
Asien-Pazifik (Hyderabad)	18.60.192.192/26
Asien-Pazifik (Melbourne)	16.50.161.192/26
Asien-Pazifik (Malaysia)	43.216.44.192/26

VPC-Flow-Logs mithilfe von Amazon Data Firehose in Splunk aufnehmen

Weitere Informationen darüber, wie Sie ein VPC-Flow-Log-Abonnement erstellen, in Firehose veröffentlichen und die VPC-Flow-Logs an ein unterstütztes Ziel senden, finden Sie unter VPC-Flow-Logs mit Amazon Data Firehose in Splunk aufnehmen.

Zugreifen auf Snowflake oder den HTTP-Endpunkt

Es gibt keine Untergruppe von AWS IP-Adressbereichen, die für Amazon Data Firehose spezifisch sind, wenn das Ziel ein HTTP-Endpunkt oder öffentliche Snowflake-Cluster ist.

Um Firehose zu einer Zulassungsliste für öffentliche Snowflake-Cluster oder zu Ihren öffentlichen HTTP- oder HTTPS-Endpunkten hinzuzufügen, fügen Sie Ihren Eingangsregeln alle aktuellen AWS IP-Adressbereiche hinzu.

> **Note**
>
> Benachrichtigungen stammen nicht immer von IP-Adressen in derselben AWS Region wie das zugehörige Thema. Sie müssen den AWS IP-Adressbereich für alle Regionen angeben.

Firehose Zugriff auf ein Snowflake-Ziel gewähren

Wenn Sie Snowflake als Ziel verwenden, übermittelt Firehose Daten über Ihre Snowflake-Konto-URL an ein Snowflake-Konto. Außerdem werden Fehlerdaten in dem von Ihnen angegebenen Amazon Simple Storage Service-Bucket gesichert, und Sie können optional einen AWS Key Management Service Schlüssel, den Sie besitzen, für die serverseitige Amazon S3 S3-Verschlüsselung verwenden. Wenn die Fehlerprotokollierung aktiviert ist, sendet Firehose Datenübermittlungsfehler an Ihre CloudWatch Logs-Streams.

Sie müssen über eine IAM-Rolle verfügen, bevor Sie einen Firehose-Stream erstellen können. Firehose nimmt diese IAM-Rolle an und erhält Zugriff auf den angegebenen Bucket, den Schlüssel und die CloudWatch Logs-Gruppe und die Streams. Verwenden Sie die folgende Zugriffsrichtlinie, damit Firehose auf Ihren S3-Bucket zugreifen kann. Wenn Sie den S3-Bucket nicht besitzen, fügen Sie `s3:PutObjectAcl` ihn der Liste der Amazon Simple Storage Service-Aktionen hinzu, wodurch der Bucket-Besitzer vollen Zugriff auf die von Firehose gelieferten Objekte erhält. Diese Richtlinie gewährt Firehose auch Zugriff auf die CloudWatch Fehlerprotokollierung. Die Richtlinie enthält

auch eine Erklärung, die den Zugriff auf Amazon Kinesis Data Streams ermöglicht. Wenn Sie keine Kinesis Data Streams als Datenquelle verwenden, können Sie diese Erklärung entfernen. Firehose verwendet IAM nicht, um auf Snowflake zuzugreifen. Für den Zugriff auf Snowflake werden die URL Ihres Snowflake-Kontos und die PrivateLink Vpce-ID im Fall eines privaten Clusters verwendet.

JSON

```
{
"Version":"2012-10-17",
    "Statement":
    [
        {
"Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
"Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
"StringEquals": {
"kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
"kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket/prefix*"
               }
```

```
            }
        },
        {
"Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
"Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
            ]
        }
    ]
}
```

Weitere Informationen darüber, wie Sie anderen AWS Diensten den Zugriff auf Ihre AWS Ressourcen ermöglichen, finden Sie unter Creating a Role to Delegate Permissions to an AWS Service im IAM-Benutzerhandbuch.

Zugreifen auf Snowflake in VPC

Wenn Ihr Snowflake-Cluster für private Links aktiviert ist, verwendet Firehose zum Zeitpunkt der Erstellung eines privaten Links einen der folgenden VPC-Endpunkte, um Daten an Ihren privaten Cluster zu übermitteln, ohne das öffentliche Internet zu nutzen. Erstellen Sie dazu Snowflake-Netzwerkregeln, um den Zugriff von folgenden Quellen für den Cluster zu ermöglichen, in dem sich Ihr Cluster befindet. `AwsVpceIds` AWS-Region Weitere Informationen finden Sie unter Netzwerkregel erstellen im Snowflake-Benutzerhandbuch.

Zu verwendende VPC-Endpunkt-IDs basierend auf Regionen, in denen sich Ihr Cluster befindet

AWS-Region	**VPCE IDs**
USA Ost (Ohio)	vpce-0d96cafcd96a50aeb vpce-0cec34343d48f537b
USA Ost (Nord-Virginia)	vpce-0b4d7e8478e141ba8 vpce-0b75cd681fb507352 vpce-01c03e63820ec00d8 vpce-0c2cfc51dc2882422 vpce-06ca862f019e4e056 vpce-020cda0cfa63f8d1c vpce-0b80504a1a783cd70 vpce-0289b9ff0b5259a96 vpce-0d7add8628bd69a12 vpce-02bfb5966cc59b2af vpce-09e707674af878bf2 vpce-049b52e96cc1a2165 vpce-0bb6c7b7a8a86cdbb vpce-03b22d599f51e80f3 vpce-01d60dc60fc106fe1 vpce-0186d20a4b24ecbef vpce-0533906401a36e416 vpce-05111fb13d396710e

AWS-Region	**VPCE IDs**
	vpce-0694613f4fbd6f514 vpce-09b21cb25fe4cc4f4 vpce-06029c3550e4d2399 vpce-00961862a21b033da vpce-01620b9ae33273587 vpce-078cf4ec226880ac9 vpce-0d711bf076ce56381 vpce-066b7e13cbfca6f6e vpce-0674541252d9ccc26 vpce-03540b88dedb4b000 vpce-0b1828e79ad394b95 vpce-0dc0e6f001fb1a60d vpce-0d8f82e71a244098a vpce-00e374d9e3f1af5ce vpce-0c1e3d6631ddb442f

AWS-Region	**VPCE IDs**
USA West (Oregon)	vpce-0f60f72da4cd1e4e7 vpce-0c60d21eb8b1669fd vpce-01c4e3e29afdafbef vpce-0cc6bf2a88da139de vpce-0797e08e169e50662 vpce-033cbe480381b5c0e vpce-00debbdd8f9eb10a5 vpce-08ec2f386c809e889 vpce-0856d14310857b545
Europa (Frankfurt)	vpce-068dbb7d71c9460fb vpce-0a7a7f095942d4ec9
Europa (Irland)	vpce-06857e59c005a6276 vpce-04390f4f8778b75f2 vpce-011fd2b1f0aa172fd
Asien-Pazifik (Tokio)	vpce-06369e5258144e68a vpce-0f2363cdb8926fbe8
Asien-Pazifik (Singapur)	vpce-049cd46cce7a12d52 vpce-0e8965a1a4bdb8941
Asien-Pazifik (Seoul)	vpce-0aa444d9001e1faa1 vpce-04a49d4dcfd02b884

AWS-Region	**VPCE IDs**
Asien-Pazifik (Sydney)	vpce-048a60a182c52be63 vpce-03c19949787fd1859
Asien-Pazifik (Mumbai)	vpce-0d68cb822f6f0db68 vpce-0517d32692ffcbde2
Europa (London)	vpce-0fd1874a0ba3b9374 vpce-08091b1a85e206029
Südamerika (São Paulo)	vpce-065169b8144e4f12e vpce-0493699f0e5762d63
Kanada (Zentral)	vpce-07e6ed81689d5271f vpce-0f53239730541394c
Europa (Paris)	vpce-09419680077e6488a vpce-0ea81ba2c08140c14
Asien-Pazifik (Osaka)	vpce-0a9f003e6a7e38c05 vpce-02886510b897b1c5a
Europa (Stockholm)	vpce-0d96410833219025a vpce-060a32f9a75ba969f
Asien-Pazifik (Jakarta)	vpce-00add4b9a25e5c649 vpce-004ae2de34338a856

Firehose Zugriff auf ein HTTP-Endpunktziel gewähren

Sie können Amazon Data Firehose verwenden, um Daten an jedes beliebige HTTP-Endpunktziel zu liefern. Amazon Data Firehose sichert diese Daten auch in dem von Ihnen angegebenen Amazon S3 S3-Bucket, und Sie können optional einen AWS KMS Schlüssel, den Sie besitzen, für die serverseitige Amazon S3 S3-Verschlüsselung verwenden. Wenn die Fehlerprotokollierung aktiviert ist, sendet Amazon Data Firehose Fehler bei der Datenübermittlung an Ihre CloudWatch Protokollstreams. Sie können es auch AWS Lambda für die Datentransformation verwenden.

Sie benötigen eine IAM-Rolle, wenn Sie einen Firehose-Stream erstellen. Amazon Data Firehose übernimmt diese IAM-Rolle und erhält Zugriff auf den angegebenen Bucket, den Schlüssel, die CloudWatch Protokollgruppe und die Streams.

Verwenden Sie die folgende Zugriffsrichtlinie, damit Amazon Data Firehose auf den S3-Bucket zugreifen kann, den Sie für die Datensicherung angegeben haben. Wenn Sie den S3-Bucket nicht besitzen, fügen Sie `s3:PutObjectAcl` ihn der Liste der Amazon S3 S3-Aktionen hinzu, wodurch der Bucket-Besitzer vollen Zugriff auf die von Amazon Data Firehose bereitgestellten Objekte erhält. Diese Richtlinie gewährt Amazon Data Firehose auch Zugriff auf die CloudWatch Fehlerprotokollierung und die AWS Lambda Datentransformation. Die Richtlinie enthält auch eine Erklärung, die den Zugriff auf Amazon Kinesis Data Streams ermöglicht. Wenn Sie keine Kinesis Data Streams als Datenquelle verwenden, können Sie diese Erklärung entfernen.

> **Important**
>
> Amazon Data Firehose verwendet IAM nicht für den Zugriff auf HTTP-Endpunktziele unterstützter Drittanbieter wie Datadog, Dynatrace, MongoDB, New Relic LogicMonitor, Splunk oder Sumo Logic. Wenn Sie auf ein bestimmtes HTTP-Endpunktziel zugreifen möchten, das einem unterstützten Drittanbieter gehört, wenden Sie sich an diesen Dienstanbieter, um den API-Schlüssel oder den Zugriffsschlüssel zu erhalten, der für die Datenlieferung an diesen Service von Amazon Data Firehose erforderlich ist.

Weitere Informationen darüber, wie Sie anderen AWS Diensten den Zugriff auf Ihre AWS Ressourcen ermöglichen, finden Sie unter Creating a Role to Delegate Permissions to an AWS Service im IAM-Benutzerhandbuch.

> **Important**
>
> Derzeit unterstützt Amazon Data Firehose KEINE Datenlieferung an HTTP-Endpunkte in einer VPC.

Kontenübergreifender Versand von Amazon MSK

Wenn Sie einen Firehose-Stream aus Ihrem Firehose (z. B. Konto B) erstellen und Ihre Quelle ein MSK-Cluster in einem anderen AWS Konto (Konto A) ist, müssen Sie über die folgenden Konfigurationen verfügen.

Konto A:

1. Wählen Sie in der Amazon-MSK-Konsole den bereitgestellten Cluster und dann Properties (Eigenschaften) aus.
2. Wählen Sie unter Netzwerkeinstellungen die Option Bearbeiten aus und aktivieren Sie die Multi-VPC-Konnektivität.
3. Wählen Sie unter Sicherheitseinstellungen die Option Clusterrichtlinie bearbeiten aus.

    a. Wenn für den Cluster noch keine Richtlinie konfiguriert ist, aktivieren Sie die Optionen Firehose-Dienstprinzipal einbeziehen und Kontoübergreifende Firehose-Zustellung für S3 aktivieren. Dadurch AWS-Managementkonsole wird automatisch eine Richtlinie mit den entsprechenden Berechtigungen generiert.

    b. Wenn für den Cluster bereits eine Richtlinie konfiguriert ist, fügen Sie der vorhandenen Richtlinie die folgenden Berechtigungen hinzu:

```
{
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
      ],
```

```
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/DO-NOT-TOUCH-
mskaas-provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20"  // ARN
 of the cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
      ],
      "Resource": "arn:aws:kafka:us-east-1:arn:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*"//topic of the
 cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": "kafka-cluster:DescribeGroup",
      "Resource": "arn:aws:kafka:us-east-1:arn:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of
 the cluster
    },
 }
```

4. Geben Sie unter AWS Principal die Prinzipal-ID von Konto B ein.
5. Geben Sie unter Thema das Apache Kafka-Thema an, aus dem Ihr Firehose-Stream Daten aufnehmen soll. Sobald der Firehose-Stream erstellt wurde, können Sie dieses Thema nicht mehr aktualisieren.
6. Wählen Sie Save Changes (Änderungen speichern)

Konto B:

1. Wählen Sie in der Firehose-Konsole die Option Firehose-Stream mithilfe von Account B erstellen.
2. Wählen Sie unter Quelle Amazon Managed Streaming für Apache Kafka.

3. Geben Sie unter Quelleinstellungen für den Cluster von Amazon Managed Streaming für Apache Kafka den ARN des Amazon-MSK-Clusters in Konto A ein.
4. Geben Sie unter Thema das Apache Kafka-Thema an, aus dem Ihr Firehose-Stream Daten aufnehmen soll. Sobald der Firehose-Stream erstellt wurde, können Sie dieses Thema nicht mehr aktualisieren.
5. Geben Sie unter Delivery Stream Name den Namen für Ihren Firehose-Stream an.

Wenn Sie Ihren Firehose-Stream erstellen, müssen Sie in Konto B über eine IAM-Rolle verfügen (standardmäßig erstellt, wenn Sie die verwenden AWS-Managementkonsole), die dem Firehose-Stream Lesezugriff auf den kontoübergreifenden Amazon MSK-Cluster für das konfigurierte Thema gewährt.

Folgendes wird von der AWS-Managementkonsole konfiguriert:

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
        ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws::cluster/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws::topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/mskaas_test_topic" //
topic of the cluster
    },
    {
    "Sid": "",
```

```
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeGroup"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
 }
```

Als Nächstes können Sie den optionalen Schritt der Konfiguration der Datensatztransformation und der Datensatzformatkonvertierung abschließen. Weitere Informationen finden Sie unter (Optional) Konfiguration der Datensatztransformation und Formatkonvertierung.

Kontoübergreifende Lieferung an ein Amazon S3 S3-Ziel

Sie können die AWS CLI oder die Amazon Data Firehose verwenden, APIs um einen Firehose-Stream in einem AWS Konto mit einem Amazon S3 S3-Ziel in einem anderen Konto zu erstellen. Das folgende Verfahren zeigt ein Beispiel für die Konfiguration eines Firehose-Streams, der Konto A gehört, um Daten an einen Amazon S3 S3-Bucket zu liefern, der Konto B gehört.

1. Erstellen Sie eine IAM-Rolle unter Konto A mithilfe der unter Grant Firehose Access to a Amazon S3 Destination beschriebenen Schritte.

   > **Note**
   >
   > Der in der Zugriffsrichtlinie angegebene Amazon-S3-Bucket befindet sich in diesem Fall jetzt im Besitz von Konto B. Stellen Sie sicher, dass `s3:PutObjectAcl` Sie der Liste der Amazon S3 S3-Aktionen in der Zugriffsrichtlinie hinzufügen, die Konto B vollen Zugriff auf die von Amazon Data Firehose gelieferten Objekte gewährt. Diese Genehmigung ist für die kontoübergreifende Lieferung erforderlich. Amazon Data Firehose setzt den Header x-amz-acl "" der Anfrage auf "bucket-owner-full-control“.

2. Um Zugriff von der zuvor erstellten IAM-Rolle zu gewähren, erstellen Sie eine S3-Bucket-Richtlinie unter Konto B. Der folgende Code ist ein Beispiel für die Bucket-Richtlinie. Weitere Informationen dazu finden Sie unter Verwendung von Bucket-Richtlinien und Benutzerrichtlinien.

JSON

```
{

    "Version":"2012-10-17",
    "Id": "PolicyID",
    "Statement": [
        {
            "Sid": "StmtID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/iam-role-name"
            },
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

3. Erstellen Sie einen Firehose-Stream unter Konto A mit der IAM-Rolle, die Sie in Schritt 1 erstellt haben.

Kontoübergreifende Lieferung an ein Serviceziel OpenSearch

Sie können die AWS CLI oder die Amazon Data Firehose verwenden APIs , um einen Firehose-Stream in einem AWS Konto mit einem OpenSearch Service-Ziel in einem anderen Konto zu erstellen. Das folgende Verfahren zeigt ein Beispiel dafür, wie Sie einen Firehose-Stream unter Konto

A erstellen und ihn so konfigurieren können, dass Daten an ein OpenSearch Serviceziel gesendet werden, das Konto B gehört.

1. Erstellen Sie eine IAM-Rolle unter Konto A mithilfe der unter the section called “Firehose Zugang zu einem Ziel des öffentlichen OpenSearch Dienstes gewähren” beschriebenen Schritte.
2. Um den Zugriff von der IAM-Rolle aus zu ermöglichen, die Sie im vorherigen Schritt erstellt haben, erstellen Sie eine OpenSearch Servicerichtlinie unter Konto B. Das folgende JSON ist ein Beispiel.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/firehose_delivery_role "
      },
      "Action": "es:ESHttpGet",
      "Resource": [
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/_all/
_settings",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_cluster/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_mapping/roletest",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/*/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/"
      ]
    }
  ]
```

```
}
```

3. Erstellen Sie einen Firehose-Stream unter Konto A mit der IAM-Rolle, die Sie in Schritt 1 erstellt haben. Wenn Sie den Firehose-Stream erstellen, verwenden Sie die AWS CLI oder die Amazon Data Firehose APIs und geben Sie das `ClusterEndpoint` Feld anstelle von Service `DomainARN` an OpenSearch .

> **Note**
>
> Um einen Firehose-Stream in einem AWS Konto mit einem OpenSearch Service-Ziel in einem anderen Konto zu erstellen, müssen Sie Amazon Data Firehose AWS CLI oder Amazon Data APIs Firehose verwenden. Sie können das nicht verwenden, AWS-Managementkonsole um diese Art von kontenübergreifender Konfiguration zu erstellen.

Verwendung von Tags zur Zugriffssteuerung

Sie können das optionale `Condition` Element (oder den `Condition` Block) in einer IAM-Richtlinie verwenden, um den Zugriff auf Amazon Data Firehose-Operationen auf der Grundlage von Tag-Schlüsseln und -Werten zu optimieren. In den folgenden Unterabschnitten wird beschrieben, wie Sie dies für die verschiedenen Amazon Data Firehose-Operationen tun können. Weitere Informationen zur Verwendung des Elements `Condition` und der Operatoren, die Sie mit diesem verwenden können, finden Sie unter IAM-JSON-Richtlinienelemente: Condition.

CreateDeliveryStream

Verwenden Sie für die Operation `CreateDeliveryStream` den Bedingungsschlüssel `aws:RequestTag`. Im folgenden Beispiel stellen `MyKey` und `MyValue` den Schlüssel und den entsprechenden Wert für einen Tag dar. Weitere Informationen finden Sie unter Verstehen Sie die Grundlagen von Tags.

JSON

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

```
            "firehose:CreateDeliveryStream",
            "firehose:TagDeliveryStream"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/MyKey": "MyValue"
            }
        }
    }]
}
```

TagDeliveryStream

Verwenden Sie für die Operation `TagDeliveryStream` den Bedingungsschlüssel `aws:TagKeys`. Im folgenden Beispiel ist `MyKey` ein Beispiel-Tag-Schlüssel.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                 }
            }
        }
    ]
}
```

UntagDeliveryStream

Verwenden Sie für die Operation `UntagDeliveryStream` den Bedingungsschlüssel `aws:TagKeys`. Im folgenden Beispiel ist `MyKey` ein Beispiel-Tag-Schlüssel.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:UntagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                 }
            }
        }
    ]
}
```

ListDeliveryStreams

Sie können die Tag-basierte Zugriffskontrolle nicht mit `ListDeliveryStreams` verwenden.

Andere Operationen

Verwenden Sie für Firehose Firehose-Operationen außer `CreateDeliveryStream` `TagDeliveryStreamUntagDeliveryStream`,`ListDeliveryStreams`, und den `aws:RequestTag` Bedingungsschlüssel. Im folgenden Beispiel stellen `MyKey` und `MyValue` den Schlüssel und den entsprechenden Wert für einen Tag dar.

`ListDeliveryStreams`, verwenden Sie den `firehose:ResourceTag` Bedingungsschlüssel, um den Zugriff auf der Grundlage der Tags in diesem Firehose-Stream zu steuern.

Im folgenden Beispiel stellen `MyKey` und `MyValue` den Schlüssel und den entsprechenden Wert für einen Tag dar. Die Richtlinie würde nur für Data Firehose-Streams gelten, deren Tag `MyKey` mit einem Wert von `MyValue` benannt ist. Weitere Informationen zur Steuerung des Zugriffs auf der Grundlage von Ressourcen-Tags finden Sie unter Steuern des Zugriffs auf AWS Ressourcen mithilfe von Tags im IAM-Benutzerhandbuch.

Authentifizieren Sie sich mit AWS Secrets Manager in Amazon Data Firehose

Amazon Data Firehose lässt sich integrieren AWS Secrets Manager , um sicheren Zugriff auf Ihre vertraulichen Daten zu ermöglichen und die Rotation von Anmeldeinformationen zu automatisieren. Diese Integration ermöglicht es Firehose, zur Laufzeit ein Geheimnis aus Secrets Manager abzurufen, um eine Verbindung zu zuvor genannten Streaming-Zielen herzustellen und Ihre Datenströme bereitzustellen. Dadurch sind Ihre Geheimnisse während des Workflows zur Stream-Erstellung weder in noch in API-Parametern im AWS-Managementkonsole Klartext sichtbar. Es bietet eine sichere Methode zur Verwaltung Ihrer Geheimnisse und entlastet Sie von komplexen Aktivitäten zur Verwaltung von Anmeldeinformationen wie der Einrichtung benutzerdefinierter Lambda-Funktionen zur Verwaltung von Passwortrotationen.

Weitere Informationen finden Sie im AWS Secrets Manager -Benutzerhandbuch.

Themen

Verstehen Sie Geheimnisse

Ein Geheimnis kann ein Passwort, ein Satz von Anmeldeinformationen wie ein Benutzername und ein Passwort, ein OAuth Token oder andere geheime Informationen sein, die Sie in verschlüsselter Form in Secrets Manager speichern.

Für jedes Ziel müssen Sie das geheime Schlüssel-Wert-Paar im richtigen JSON-Format angeben, wie im folgenden Abschnitt gezeigt. Amazon Data Firehose kann keine Verbindung zu Ihrem Ziel herstellen, wenn Ihr Secret nicht das richtige JSON-Format für das Ziel hat.

Geheimformat für Datenbanken wie MySQL und PostgreSQL

```
{
    "username":  "<username>",
    "password":  "<password>"
```

```
}
```

Geheimformat für Amazon Redshift Provisioned Cluster und Amazon Redshift Serverless Workgroup

```
{
    "username":  "<username>",
    "password":  "<password>"
}
```

Format des Geheimnisses für Splunk

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

Format des Geheimnisses für Snowflake

```
{
    "user":  "<snowflake-username>",
    "private_key":  "<snowflake-private-key>", // without the beginning and ending
 private key, remove all spaces and newlines
    "key_passphrase":  "<snowflake-private-key-passphrase>" // optional
}
```

Geheimformat für HTTP-Endpunkt, Coralogix, Datadog, Dynatrace, Elastic, Honeycomb, Logz.io, MongoDB Cloud und New LogicMonitor Relic

```
{
    "api_key":  "<apikey>"
}
```

Ein Secret erstellen

Um ein Geheimnis zu erstellen, folgen Sie den Schritten unter Ein AWS Secrets Manager Geheimnis erstellen im AWS Secrets Manager Benutzerhandbuch.

Verwenden Sie das Geheimnis

Wir empfehlen Ihnen, Ihre Anmeldeinformationen oder Schlüssel AWS Secrets Manager zu speichern, um eine Verbindung zu Streaming-Zielen wie Amazon Redshift, HTTP-Endpoint,

Snowflake, Splunk, Coralogix, Datadog, Dynatrace, Elastic, Honeycomb, Logz.io, MongoDB Cloud und New Relic herzustellen. LogicMonitor

Sie können die Authentifizierung mit Secrets Manager für diese Ziele über die AWS Management Console zum Zeitpunkt der Erstellung des Firehose-Streams konfigurieren. Weitere Informationen finden Sie unter Zieleinstellungen konfigurieren. Alternativ können Sie auch die UpdateDestinationAPI-Operationen CreateDeliveryStreamund verwenden, um die Authentifizierung mit Secrets Manager zu konfigurieren.

Firehose speichert die Geheimnisse mit einer Verschlüsselung zwischen und verwendet sie für jede Verbindung zu Zielen. Es aktualisiert den Cache alle 10 Minuten, um sicherzustellen, dass die neuesten Anmeldeinformationen verwendet werden.

Sie können die Funktion zum Abrufen von Geheimnissen aus Secrets Manager jederzeit während des Lebenszyklus des Streams deaktivieren. Wenn Sie Secrets Manager nicht zum Abrufen von Geheimnissen verwenden möchten, können Sie stattdessen den API-Schlüssel username/password oder verwenden.

> **Note**
>
> Für diese Funktion in Firehose fallen zwar keine zusätzlichen Kosten an, Ihnen werden jedoch der Zugriff und die Wartung von Secrets Manager in Rechnung gestellt. Weitere Informationen finden Sie auf der Seite mit den AWS Secrets ManagerPreisen.

Gewähren Sie Firehose Zugriff, um das Geheimnis abzurufen

Damit Firehose ein Geheimnis abrufen kann AWS Secrets Manager, müssen Sie Firehose die erforderlichen Berechtigungen für den Zugriff auf das Geheimnis und den Schlüssel, der Ihr Geheimnis verschlüsselt, gewähren.

Beim Speichern und Abrufen von AWS Secrets Manager Geheimnissen gibt es einige unterschiedliche Konfigurationsoptionen, je nachdem, wo das Geheimnis gespeichert und wie es verschlüsselt ist.

- Wenn das Geheimnis in demselben AWS Konto wie Ihre IAM-Rolle gespeichert und mit dem AWS verwalteten Standardschlüssel (`aws/secretsmanager`) verschlüsselt ist, benötigt die IAM-Rolle, von der Firehose annimmt, nur eine `secretsmanager:GetSecretValue` Genehmigung für das Geheimnis.

```
// secret role policy
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "Secret ARN"
        }
    ]
}
```

Weitere Informationen zu IAM-Richtlinien finden Sie unter Beispiele für Berechtigungsrichtlinien. AWS Secrets Manager

- Wenn der geheime Schlüssel in demselben Konto wie die Rolle gespeichert, aber mit einem vom Kunden verwalteten Schlüssel (CMK) verschlüsselt ist, benötigt die Rolle beides `secretsmanager:GetSecretValue` und `kms:Decrypt` Berechtigungen. Die CMK-Richtlinie muss auch die Ausführung der IAM-Rolle ermöglichen. `kms:Decrypt`
- Wenn das Geheimnis in einem anderen AWS Konto als Ihrer Rolle gespeichert und mit dem AWS verwalteten Standardschlüssel verschlüsselt ist, ist diese Konfiguration nicht möglich, da Secrets Manager keinen kontoübergreifenden Zugriff zulässt, wenn das Geheimnis mit einem AWS verwalteten Schlüssel verschlüsselt ist.
- Wenn das Geheimnis in einem anderen Konto gespeichert und mit einem CMK verschlüsselt ist, benötigt die IAM-Rolle eine Genehmigung für das Geheimnis und `kms:Decrypt` eine `secretsmanager:GetSecretValue` Genehmigung für den CMK. Die Ressourcenrichtlinie des Geheimnisses und die CMK-Richtlinie des anderen Kontos müssen der IAM-Rolle ebenfalls die erforderlichen Berechtigungen gewähren. Weitere Informationen finden Sie unter Kontoübergreifender Zugriff.

Drehe das Geheimnis

Rotation bedeutet, dass Sie ein Geheimnis regelmäßig aktualisieren. Sie können so konfigurieren AWS Secrets Manager , dass das Geheimnis nach einem von Ihnen festgelegten Zeitplan automatisch für Sie rotiert wird. Auf diese Weise können Sie langfristige Geheimnisse durch kurzfristige ersetzen. Dies trägt dazu bei, das Risiko von Kompromissen zu verringern. Weitere

Informationen finden Sie im AWS Secrets Manager Benutzerhandbuch unter Rotate AWS Secrets Manager Secrets.

Verwaltung von IAM-Rollen über die Amazon Data Firehose-Konsole

Amazon Data Firehose ist ein vollständig verwalteter Service, der Streaming-Daten in Echtzeit an Ziele liefert. Sie können Firehose auch so konfigurieren, dass das Format Ihrer Daten vor der Auslieferung transformiert und konvertiert wird. Um diese Funktionen nutzen zu können, müssen Sie zunächst IAM-Rollen bereitstellen, um Firehose beim Erstellen oder Bearbeiten eines Firehose-Streams Berechtigungen zu gewähren. Firehose verwendet diese IAM-Rolle für alle Berechtigungen, die der Firehose-Stream benötigt.

Stellen Sie sich zum Beispiel ein Szenario vor, in dem Sie einen Firehose erstellen, der Daten an Amazon S3 liefert, und für diesen Firehose-Stream Transform-Quelldatensätze mit aktivierter AWS Lambda Funktion vorhanden sind. In diesem Fall müssen Sie IAM-Rollen bereitstellen, um Firehose Berechtigungen für den Zugriff auf den S3-Bucket und das Aufrufen der Lambda-Funktion zu gewähren, wie im Folgenden gezeigt.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "lambdaProcessing",
        "Effect": "Allow",
        "Action": ["lambda:InvokeFunction", "lambda:GetFunctionConfiguration"],
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:<lambda
 function name>:<lambda function version>"
    }, {
        "Sid": "s3Permissions",
        "Effect": "Allow",
        "Action": ["s3:AbortMultipartUpload", "s3:GetBucketLocation",
 "s3:GetObject", "s3:ListBucket", "s3:ListBucketMultipartUploads",
 "s3:PutObject"],
        "Resource": ["arn:aws:s3:::<bucket name>", "arn:aws:s3:::<bucket name>/
*"]
    }]
}
```

Mit Firehose Firehose-Konsole können Sie wählen, wie Sie diese Rollen bereitstellen möchten. Sie können aus einer der folgenden Optionen wählen.

- Wählen Sie eine bestehende IAM-Rolle
- Erstellen Sie eine neue IAM-Rolle von der Konsole aus

Wählen Sie eine bestehende IAM-Rolle

Sie können aus einer vorhandenen IAM-Rolle wählen. Stellen Sie bei dieser Option sicher, dass die von Ihnen gewählte IAM-Rolle über die richtige Vertrauensrichtlinie und die für Ihre Quelle und Ihr Ziel erforderlichen Berechtigungen verfügt. Weitere Informationen finden Sie unter Zugriffskontrolle mit Amazon Data Firehose.

Erstellen Sie eine neue IAM-Rolle von der Konsole aus

Alternativ können Sie auch die Firehose-Konsole verwenden, um in Ihrem Namen eine neue Rolle zu erstellen.

Wenn Firehose in Ihrem Namen eine IAM-Rolle erstellt, enthält die Rolle automatisch alle Berechtigungs- und Vertrauensrichtlinien, die die erforderlichen Berechtigungen auf der Grundlage der Firehose-Stream-Konfiguration gewähren.

Wenn Sie beispielsweise die AWS Lambda Funktion Quelldatensätze mit transformieren nicht aktiviert haben, generiert die Konsole die folgende Anweisung in der Berechtigungsrichtlinie.

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
   "Action": [
     "lambda:InvokeFunction",
     "lambda:GetFunctionConfiguration"
   ],
   "Resource": "arn:aws:lambda:us-east-1123456789012:function:
%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%"
}
```

> **Note**
>
> Es ist unbedenklich, die folgenden Richtlinienerklärungen zu ignorieren, `%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%` da sie keine Berechtigungen für Ressourcen gewähren.

Die Konsole zum Erstellen und Bearbeiten von Firehose-Stream-Workflows erstellt auch eine Vertrauensrichtlinie und fügt sie der IAM-Rolle hinzu. Die Vertrauensrichtlinie ermöglicht es Firehose, die IAM-Rolle zu übernehmen. Im Folgenden finden Sie ein Beispiel für eine Vertrauensrichtlinie.

JSON

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

> **Important**
>
> - Sie sollten vermeiden, dieselbe, von der Konsole verwaltete IAM-Rolle für mehrere Firehose-Streams zu verwenden. Andernfalls könnte die IAM-Rolle zu freizügig werden oder zu Fehlern führen.
> - Um in einer Berechtigungsrichtlinie andere Richtlinienaussagen als in einer über die Konsole verwalteten IAM-Rolle zu verwenden, können Sie Ihre eigene IAM-Rolle erstellen und die Richtlinienanweisungen in eine Berechtigungsrichtlinie kopieren, die der neuen Rolle zugeordnet ist. Um die Rolle an den Firehose-Stream anzuhängen, wählen Sie im Dienstzugriff die Option Bestehende IAM-Rolle auswählen aus.
> - Die Konsole verwaltet alle IAM-Rollen, deren ARN die Zeichenfolge service-role enthält. Wenn Sie die vorhandene IAM-Rollenoption wählen, stellen Sie sicher, dass Sie eine IAM-

Rolle ohne die Zeichenfolge service-role in ihrem ARN auswählen, damit die Konsole keine Änderungen daran vornimmt.

Schritte zum Erstellen einer IAM-Rolle über die Konsole

1. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie Create Firehose stream (Firehose-Stream erstellen) aus.
3. Wählen Sie eine Quelle und ein Ziel. Weitere Informationen finden Sie unter Tutorial: Einen Firehose-Stream von der Konsole aus erstellen.
4. Wählen Sie die Zieleinstellungen. Weitere Informationen finden Sie unter Zieleinstellungen konfigurieren.
5. Wählen Sie unter Erweiterte Einstellungen für Dienstzugriff die Option IAM-Rolle erstellen oder aktualisieren aus.

   > **Note**
   >
   > Dies ist eine Standardoption. Um eine bestehende Rolle zu verwenden, wählen Sie die Option Bestehende IAM-Rolle auswählen. Die Firehose-Konsole nimmt keine Änderungen an Ihrer eigenen Rolle vor.

6. Wählen Sie Create Firehose stream (Firehose-Stream erstellen) aus.

Bearbeiten Sie die IAM-Rolle von der Konsole aus

Wenn Sie einen Firehose-Stream bearbeiten, aktualisiert Firehose die entsprechende Berechtigungsrichtlinie entsprechend, um die Änderungen an der Konfiguration und den Berechtigungen widerzuspiegeln.

Wenn Sie beispielsweise den Firehose-Stream bearbeiten und die AWS Lambda Funktion „Quelldatensätze mit der neuesten Version der Lambda-Funktion transformieren“ aktivieren`exampleLambdaFunction`, erhalten Sie die folgende Richtlinienanweisung in der Berechtigungsrichtlinie.

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
```

```
  "Action": [
    "lambda:InvokeFunction",
    "lambda:GetFunctionConfiguration"
  ],
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:exampleLambdaFunction:
$LATEST"
}
```

> **Important**
>
> Eine von der Konsole verwaltete IAM-Rolle ist so konzipiert, dass sie autonom ist. Es wird nicht empfohlen, die Berechtigungsrichtlinie oder die Vertrauensrichtlinie außerhalb der Konsole zu ändern.

Schritte zum Bearbeiten der IAM-Rolle von der Konsole aus

1. Öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com/firehose/.
2. Wählen Sie Firehose-Streams und wählen Sie den Namen eines Firehose-Streams, den Sie aktualisieren möchten.
3. Wählen Sie auf der Registerkarte Konfiguration im Abschnitt Serverzugriff die Option Bearbeiten aus.
4. Aktualisieren Sie die IAM-Rollenoption.

   > **Note**
   >
   > Standardmäßig aktualisiert die Konsole eine IAM-Rolle immer mit dem Pattern Service-Role in ihrem ARN. Wenn Sie die vorhandene IAM-Rollenoption wählen, stellen Sie sicher, dass Sie eine IAM-Rolle ohne die Zeichenfolge service-role in ihrem ARN auswählen, damit die Konsole keine Änderungen daran vornimmt.

5. Wählen Sie Änderungen speichern aus.

Verstehen Sie die Einhaltung von Vorschriften für Amazon Data Firehose

Externe Prüfer bewerten die Sicherheit und Konformität von Amazon Data Firehose im Rahmen mehrerer AWS Compliance-Programme. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Eine Liste der AWS Services im Rahmen bestimmter Compliance-Programme finden Sie unter AWS Services im Umfang nach Compliance-Programmen. Allgemeine Informationen finden Sie unter AWS -Compliance-Programme.

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte in AWS Artifact herunterladen.

Ihre Compliance-Verantwortung bei der Verwendung von Data Firehose hängt von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Falls Ihre Nutzung von Data Firehose der Einhaltung von Standards wie HIPAA, PCI oder FedRAMP unterliegt, bietet Firehose Ressourcen, die Ihnen helfen: AWS

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Umgebungen beschrieben, auf denen Sicherheit und Compliance im Vordergrund stehen. AWS
- Whitepaper „Architecting for HIPAA Security and Compliance“ — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-konforme Anwendungen entwickeln können. AWS
- AWS Ressourcen zur Einhaltung von Vorschriften — Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- AWS Config— Mit diesem AWS Service wird bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub CSPM— Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus und hilft Ihnen AWS , die Einhaltung der Sicherheitsstandards und bewährten Verfahren der Sicherheitsbranche zu überprüfen.

Resilienz in Amazon Data Firehose

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger

Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur bietet Data Firehose mehrere Funktionen, die Sie bei der Unterstützung Ihrer Datenausfallsicherheit und Ihrer Backup-Anforderungen unterstützen.

Notfallwiederherstellung

Amazon Data Firehose läuft im serverlosen Modus und kümmert sich durch automatische Migration um Hostverschlechterungen, Verfügbarkeit der Availability Zone und andere infrastrukturbezogene Probleme. In diesem Fall stellt Amazon Data Firehose sicher, dass der Firehose-Stream ohne Datenverlust migriert wird.

Verstehen Sie die Infrastruktursicherheit in Amazon Data Firehose

Als verwalteter Service ist Amazon Data Firehose durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Firehose zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

> **Note**
>
> Für ausgehende HTTPS-Anfragen verwendet Amazon Data Firehose eine HTTP-Bibliothek, die automatisch die höchste TLS-Protokollversion auswählt, die auf der Zielseite unterstützt wird.

Verwenden von Amazon Data Firehose mit AWS PrivateLink

Sie können einen VPC-Schnittstellen-Endpunkt (AWS PrivateLink) verwenden, um von Ihrer VPC aus auf Amazon Data Firehose zuzugreifen, ohne dass ein Internet Gateway oder ein NAT-Gateway erforderlich ist. Schnittstellen-VPC-Endpunkte benötigen kein Internet-Gateway, kein NAT-Gerät, keine VPN-Verbindung oder Direct Connect Verbindung. Interface VPC-Endpoints basieren auf einer AWS Technologie AWS PrivateLink, die private Kommunikation zwischen AWS Services über eine elastic network interface mit Private IPs in Ihrer Amazon VPC ermöglicht. Weitere Informationen finden Sie unter Amazon Virtual Private Cloud.

Verwenden von Schnittstellen-VPC-Endpunkten (AWS PrivateLink) für Firehose

Erstellen Sie zunächst einen VPC-Schnittstellen-Endpunkt, damit Ihr Amazon Data Firehose-Verkehr von Ihren Amazon VPC-Ressourcen über den Schnittstellen-VPC-Endpunkt fließen kann. Wenn Sie einen Endpunkt erstellen, können Sie ihm eine Endpunktrichtlinie hinzufügen, die den Zugriff auf Amazon Data Firehose steuert. Weitere Informationen zur Verwendung von Richtlinien zur Steuerung des Zugriffs von einem VPC-Endpunkt auf Amazon Data Firehose finden Sie unter Steuern des Zugriffs auf Services mit VPC-Endpunkten.

Das folgende Beispiel zeigt, wie Sie eine AWS Lambda Funktion in einer VPC einrichten und einen VPC-Endpunkt erstellen können, damit die Funktion sicher mit dem Amazon Data Firehose-Service kommunizieren kann. In diesem Beispiel verwenden Sie eine Richtlinie, die es der Lambda-Funktion ermöglicht, die Firehose-Streams in der aktuellen Region aufzulisten, aber keinen Firehose-Stream zu beschreiben.

Erstellen eines VPC-Endpunkts

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im VPC-Dashboard Endpoints (Endpunkte) aus.
3. Klicken Sie auf Endpunkt erstellen.

4. Wählen Sie in der Liste der Servicenamen `com.amazonaws.`*`your_region`*`.kinesis-firehose` aus.
5. Wählen Sie die VPC und ein oder mehrere Subnetze aus, in dem/denen Sie den Endpunkt erstellen möchten.
6. Wählen Sie eine oder mehrere Sicherheitsgruppen aus, die mit den Endpunkten verknüpft werden soll(en).
7. Wählen Sie für Policy (Richtlinie) Custom (Benutzerdefiniert) aus und fügen Sie die folgende Richtlinie ein:

```
{
    "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. Wählen Sie Endpunkt erstellen aus.

Erstellen einer IAM-Rolle zur Verwendung mit der Lambda-Funktion

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.

2. Wählen Sie im linken Navigationsbereich Rollen und dann Rolle erstellen aus.
3. Belassen Sie es unter Typ der vertrauenswürdigen Entität auswählen bei der Standard-Auswahl AWS -Service.
4. Wählen Sie unter Choose the service that will use this role (Die Rolle auswählen, die diese Rollen verwenden wird) die Option Lambda aus.
5. Wählen Sie Next: Permissions (Weiter: Berechtigungen) aus.
6. Suchen Sie in der Liste der Richtlinien nach den beiden Richtlinien mit den Namen `AWS LambdaVPCAccessExecutionRole` und `AmazonDataFirehoseReadOnlyAccess`.

   > **Important**
   >
   > Dies ist ein Beispiel. Möglicherweise benötigen Sie strengere Richtlinien für Ihre Produktionsumgebung.

7. Wählen Sie Weiter: Tags aus. Das Hinzufügen von Tags ist für diese Übung nicht zweckmäßig. Wählen Sie Weiter: Prüfen aus.
8. Geben Sie einen Namen für die Rolle ein, wählen Sie dann Rolle erstellen aus.

Erstellen einer Lambda-Funktion innerhalb der VPC

1. Öffnen Sie die AWS Lambda Konsole unter. https://console.aws.amazon.com/lambda/
2. Wählen Sie Funktion erstellen.
3. Wählen Sie Von Grund auf neu schreiben aus.
4. Geben Sie einen Namen für die Funktion ein und setzen Sie dann Runtime auf `Python 3.9` oder höher.
5. Erweitern Sie unter Permissions (Berechtigungen) den Bereich Choose or create an execution role (Ausführungsrolle wählen oder erstellen).
6. Wählen Sie in der Liste Execution role (Ausführungsrolle) die Option Use an existing role (Verwenden einer vorhandenen Rolle) aus.
7. Wählen Sie in der Liste Existing role (Vorhandene Rolle) die oben erstellte Rolle aus.
8. Wählen Sie Funktion erstellen.
9. Fügen Sie unter Function code (Funktionscode) den folgenden Code ein.

```
    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. Legen Sie unter Basic settings (Grundlegende Einstellungen) die Zeitüberschreitung auf 1 Minute fest.
11. Wählen Sie unter Network (Netzwerk) die VPC aus, für die Sie den obigen Endpunkt erstellt haben, und wählen Sie dann die Subnetze sowie die Sicherheitsgruppe aus, die mit dem Endpunkt bei der Erstellung verknüpft wurden.
12. Wählen Sie oben auf der Seite Speichern aus.
13. Wählen Sie Test aus.
14. Geben Sie einen Ereignisnamen ein und wählen Sie dann Erstellen.

15. Wählen Sie erneut Test (Testen) aus. Dies bewirkt, dass die Funktion ausgeführt wird. Erweitern Sie nach Anzeige des Ausführungsergebnisses den Bereich Details (Details) und vergleichen Sie die Protokollausgabe mit dem Funktionscode. Erfolgreiche Ergebnisse zeigen eine Liste der Firehose-Streams in der Region sowie die folgende Ausgabe:

```
Calling describe_delivery_stream.

AccessDeniedException

Access denied test succeeded.
```

Unterstützt AWS-Regionen

Schnittstellen-VPC-Endpunkte werden derzeit in den folgenden Regionen unterstützt.

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Thailand)
- Asien-Pazifik (Tokio)
- Asien-Pazifik (Hongkong)
- Kanada (Zentral)
- Kanada West (Calgary)
- China (Peking)
- China (Ningxia)
- Europe (Frankfurt)
- Europa (Irland)
- Europa (London)
- Europa (Paris)

- Mexiko (Zentral)
- Südamerika (São Paulo)
- AWS GovCloud (US-Ost)
- AWS GovCloud (US-West)
- Europa (Spain)
- Naher Osten (VAE)
- Asien-Pazifik (Jakarta)
- Asien-Pazifik (Osaka)
- Israel (Tel Aviv)
- Asien-Pazifik (Malaysia)

Implementieren Sie bewährte Sicherheitsmethoden für Amazon Data Firehose

Amazon Data Firehose bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Implementieren des Zugriffs mit geringsten Berechtigungen

Bei der Erteilung von Berechtigungen entscheiden Sie, wer welche Berechtigungen für welche Amazon Data Firehose erhält. Sie aktivieren die spezifischen Aktionen, die daraufhin für die betreffenden Ressourcen erlaubt sein sollen. Aus diesem Grund sollten Sie nur Berechtigungen gewähren, die zum Ausführen einer Aufgabe erforderlich sind. Die Implementierung der geringstmöglichen Zugriffsrechte ist eine grundlegende Voraussetzung zum Reduzieren des Sicherheitsrisikos und der Auswirkungen, die aufgrund von Fehlern oder böswilligen Absichten entstehen könnten.

Verwenden von IAM-Rollen

Producer- und Client-Anwendungen müssen über gültige Anmeldeinformationen für den Zugriff auf Firehose-Streams verfügen, und Ihr Firehose-Stream muss über gültige Anmeldeinformationen für den Zugriff auf Ziele verfügen. Sie sollten AWS Anmeldeinformationen nicht direkt in einer Client-

Anwendung oder in einem Amazon S3 S3-Bucket speichern. Dabei handelt es sich um langfristige Anmeldeinformationen, die nicht automatisch rotiert werden und bedeutende geschäftliche Auswirkungen haben könnten, wenn sie kompromittiert werden.

Stattdessen sollten Sie eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Ihre Producer- und Client-Anwendungen für den Zugriff auf Firehose-Streams zu verwalten. Wenn Sie eine Rolle verwenden, müssen Sie keine langfristigen Anmeldeinformationen (z. B. Benutzername und Passwort oder Zugriffsschlüssel) für den Zugriff auf andere Ressourcen verwenden.

Weitere Informationen finden Sie unter folgenden Themen im IAM-Benutzerhandbuch:

- IAM-Rollen
- Gängige Szenarien für Rollen: Benutzer, Anwendungen und Services

Implementieren Sie serverseitige Verschlüsselung in abhängigen Ressourcen

Daten im Ruhezustand und Daten während der Übertragung können in Amazon Data Firehose verschlüsselt werden. Weitere Informationen finden Sie unter Datenschutz bei Amazon Data Firehose.

Wird CloudTrail zur Überwachung von API-Aufrufen verwendet

Amazon Data Firehose ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in Amazon Data Firehose ausgeführt wurden.

Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Amazon Data Firehose gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen finden Sie unter the section called “Firehose-API-Aufrufe protokollieren”.

Überwachen Sie Amazon Data Firehose

Sie können Amazon Data Firehose mit den folgenden Funktionen überwachen:

Themen

Implementieren Sie bewährte Verfahren mit Alarmen CloudWatch

Fügen Sie CloudWatch Alarme hinzu, wenn die folgenden Metriken das Pufferlimit (maximal 15 Minuten) überschreiten.

- `DeliveryToS3.DataFreshness`
- `DeliveryToIceberg.DataFreshness`
- `DeliveryToSplunk.DataFreshness`
- `DeliveryToAmazonOpenSearchService.DataFreshness`
- `DeliveryToAmazonOpenSearchServerless.DataFreshness`
- `DeliveryToHttpEndpoint.DataFreshness`

Erstellen Sie außerdem Alarme basierend auf den folgenden metrischen mathematischen Ausdrücken.

- `IncomingBytes (Sum per 5 Minutes) / 300` nähert sich einem Prozentsatz von `BytesPerSecondLimit`.
- `IncomingRecords (Sum per 5 Minutes) / 300` nähert sich einem Prozentsatz von `RecordsPerSecondLimit`.

- `IncomingPutRequests (Sum per 5 Minutes) / 300` nähert sich einem Prozentsatz von `PutRequestsPerSecondLimit`.

Eine weitere Metrik, für die wir einen Alarm empfehlen, ist `ThrottledRecords`.

Weitere Informationen zur Fehlerbehebung, wenn Alarme in den `ALARM`-Status übergehen, finden Sie unter Beheben von Fehlern.

Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch

> **Important**
>
> Stellen Sie sicher, dass Sie Alarme für alle CloudWatch Messwerte aktivieren, die zu Ihrem Ziel gehören, um Fehler rechtzeitig zu erkennen.

Amazon Data Firehose ist in CloudWatch Amazon-Metriken integriert, sodass Sie CloudWatch Metriken für Ihre Firehose-Streams sammeln, anzeigen und analysieren können. Sie können beispielsweise die `IncomingBytes` und `IncomingRecords` -Metriken überwachen, um den Überblick über die Daten zu behalten, die von Datenproduzenten in Amazon Data Firehose aufgenommen wurden.

Amazon Data Firehose sammelt und veröffentlicht jede Minute CloudWatch Metriken. Wenn eingehende Datenmengen jedoch nur für einige Sekunden auftreten, werden sie möglicherweise nicht vollständig erfasst oder sind in den einminütigen Metriken nicht sichtbar. Dies liegt daran, dass CloudWatch Metriken von Amazon Data Firehose in Intervallen von einer Minute aggregiert werden.

Die für Firehose gesammelten Metriken sind kostenlos. Weitere Informationen zu Kinesis-Agent-Metriken finden Sie unter Überwachen Sie den Zustand des Kinesis-Agenten.

Themen

- CloudWatch Protokolliert Dekomprimierungsmetriken
- CloudWatch Konvertierungsmetriken formatieren
- Metriken zur serverseitigen Verschlüsselung (SSE) CloudWatch
- Abmessungen für Amazon Data Firehose
- Nutzungsmetriken von Amazon Data Firehose

CloudWatch Metriken für die dynamische Partitionierung

Wenn die dynamische Partitionierung aktiviert ist, umfasst der AWS/Firehose-Namespace die folgenden Metriken.

Metrik	Beschreibung
`ActivePartitionsLimit`	Die maximale Anzahl aktiver Partitionen, die ein Firehose-Stream verarbeitet, bevor Daten an den Fehler-Bucket gesendet werden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`PartitionCount`	Die Anzahl der Partitionen, die verarbeitet werden, mit anderen Worten, die Anzahl der aktiven Partitionen. Diese Zahl variiert zwischen 1 und dem Limit für die Partitionsanzahl von 500 (Standard). Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`PartitionCountExceeded`	Diese Metrik gibt an, ob Sie das Limit für die Partition sanzahl überschreiten. Je nachdem, ob das Limit überschritten wurde oder nicht, gibt es 1 oder 0 aus.
`JQProcessing.Duration`	Gibt die Zeit zurück, die für die Ausführung des JQ-Ausdrucks in der JQ-Lambda-Funktion benötigt wurde.

Metrik	Beschreibung
	Einheiten: Millisekunden
`PerPartitionThroughput`	Gibt den Durchsatz an, der pro Partition verarbeitet wird. Mit dieser Metrik können Sie den Durchsatz pro Partition überwachen. Einheiten: StandardUnit. BytesSecond
`DeliveryToS3.ObjectCount`	Gibt die Anzahl der Objekte an, die an Ihren S3-Bucket geliefert werden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

CloudWatch Metriken für die Datenbereitstellung

Der `AWS/Firehose`-Namespace enthält die folgenden Service-Level-Metriken. Wenn Sie bei `BackupToS3.Success`, `DeliveryToS3.Success`, `DeliveryToSplunk.Success`, `DeliveryToAmazonOpenSearchService.Success` oder `DeliveryToRedshift.Success` einen leichten Rückgang des Durchschnitts feststellen, bedeutet das nicht, dass es zu einem Datenverlust gekommen ist. Amazon Data Firehose versucht erneut, Fehler bei der Zustellung zu melden, und fährt erst fort, wenn die Datensätze erfolgreich entweder an das konfigurierte Ziel oder an den Backup-S3-Bucket übermittelt wurden.

Themen

Lieferung zum Service OpenSearch

Metrik	Beschreibung
DeliveryToAmazonOpenSearchService.Bytes	Die Anzahl der Byte, die im angegebenen Zeitraum für den OpenSearch Service indexiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToAmazonOpenSearchService.DataFreshness	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Jeder Datensatz, der älter als dieses Alter ist, wurde an den OpenSearch Service übermittelt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Sekunden
DeliveryToAmazonOpenSearchService.Records	Die Anzahl der Datensätze, die im angegebenen Zeitraum für den OpenSearch Service indexiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToAmazonOpenSearchService.Success	Die Summe der erfolgreich indizierten Datensätze.
DeliveryToS3.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Amazon Data Firehose gibt diese Metrik nur aus, wenn Sie die Sicherung für alle Dokumente aktivieren. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben

Metrik	Beschreibung
	Einheiten: Anzahl
DeliveryToS3.DataFreshness	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den S3-Bucket bereitgestellt. Amazon Data Firehose gibt diese Metrik nur aus, wenn Sie die Sicherung für alle Dokumente aktivieren. Einheiten: Sekunden
DeliveryToS3.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Amazon Data Firehose gibt diese Metrik nur aus, wenn Sie die Sicherung für alle Dokumente aktivieren. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Befehle. Amazon Data Firehose gibt diese Metrik immer aus, unabhängig davon, ob die Sicherung nur für fehlgeschlagene Dokumente oder für alle Dokumente aktiviert ist.
DeliveryToAmazonOpenSearchService.AuthFailure	Authentication/authorization error. Verify the OS/ESCluster-Richtlinien und Rollenberechtigungen. 0 bedeutet, dass kein Problem vorhanden ist. 1 bedeutet, dass die Authentifizierung fehlgeschlagen ist.
DeliveryToAmazonOpenSearchService.DeliveryRejected	Fehler beim Ablehnen der Lieferung. Überprüfen Sie die OS/ES Clusterrichtlinie und die Rollenberechtigungen. 0 bedeutet, dass kein Problem vorliegt. 1 bedeutet, dass ein Zustellungsfehler vorliegt.

Lieferung an OpenSearch Serverless

Metrik	Beschreibung
`DeliveryToAmazonOpenSearchServerless.Bytes`	Die Anzahl der Byte, die im angegebenen Zeitraum für OpenSearch Serverless indexiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
`DeliveryToAmazonOpenSearchServerless.DataFreshness`	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Jeder Datensatz, der älter als dieses Alter ist, wurde an Serverless übermittelt OpenSearch . Einheiten: Sekunden
`DeliveryToAmazonOpenSearchServerless.Records`	Die Anzahl der Datensätze, die im angegebenen Zeitraum für OpenSearch Serverless indexiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`DeliveryToAmazonOpenSearchServerless.Success`	Die Summe der erfolgreich indizierten Datensätze.
`DeliveryToS3.Bytes`	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Amazon Data Firehose gibt diese Metrik nur aus, wenn Sie die Sicherung für alle Dokumente aktivieren. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
DeliveryToS3.DataFreshness	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den S3-Bucket bereitgestellt. Amazon Data Firehose gibt diese Metrik nur aus, wenn Sie die Sicherung für alle Dokumente aktivieren. Einheiten: Sekunden
DeliveryToS3.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Amazon Data Firehose gibt diese Metrik nur aus, wenn Sie die Sicherung für alle Dokumente aktivieren. Einheiten: Anzahl
DeliveryToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Befehle. Amazon Data Firehose gibt diese Metrik immer aus, unabhängig davon, ob die Sicherung nur für fehlgeschlagene Dokumente oder für alle Dokumente aktiviert ist.
DeliveryToAmazonOpenSearchServerless.AuthFailure	Authentication/authorization error. Verify the OS/ESCluster-Richtlinien und Rollenberechtigungen. 0 bedeutet, dass kein Problem vorhanden ist. 1 bedeutet, dass ein Authentifizierungsfehler vorliegt.
DeliveryToAmazonOpenSearchServerless.DeliveryRejected	Fehler beim Ablehnen der Lieferung. Überprüfen Sie die OS/ES Clusterrichtlinie und die Rollenberechtigungen. 0 bedeutet, dass kein Problem vorliegt. 1 bedeutet, dass ein Zustellungsfehler vorliegt.

Lieferung an Amazon Redshift

Metrik	Beschreibung
DeliveryToRedshift.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum in Amazon Redshift kopiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToRedshift.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum in Amazon Redshift kopiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToRedshift.Success	Die Summe der erfolgreichen Amazon Redshift COPY-Befehle.
DeliveryToS3.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToS3.DataFreshness	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Jeder Datensatz, der älter als dieses Alter ist, wird in den S3-Bucket übertragen. Einheiten: Sekunden
DeliveryToS3.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden.

Metrik	Beschreibung
	Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`DeliveryToS3.Success`	Die Summe der erfolgreichen Amazon S3 S3-Put-Be fehle.
`DeliveryToRedshift .DataFreshness`	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Jeder Datensatz, der älter als dieses Alter ist, wird an den Amazon Redshift Redshift-Cluster übermitte lt.
`BackupToS3.Bytes`	Die Anzahl der Bytes, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung auf Amazon S3 aktiviert ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`BackupToS3.DataFreshness`	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den Amazon-S3-Bucket bereitgestellt. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung auf Amazon S3 aktiviert ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden

Metrik	Beschreibung
BackupToS3.Records	Die Anzahl der Aufzeichnungen, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung auf Amazon S3 aktiviert ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
BackupToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Befehle für das Backup. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung auf Amazon S3 aktiviert ist.

Bereitstellung für Amazon S3

Die Metriken in der folgenden Tabelle beziehen sich auf die Lieferung an Amazon S3, wenn dies das Hauptziel des Firehose-Streams ist.

Metrik	Beschreibung
DeliveryToS3.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToS3.DataFreshness	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den S3-Bucket bereitgestellt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben

Metrik	Beschreibung
	Einheiten: Sekunden
DeliveryToS3.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Be fehle.
BackupToS3.Bytes	Die Anzahl der Bytes, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung aktiviert ist (was nur möglich ist, wenn auch die Datentran sformation aktiviert ist). Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
BackupToS3.DataFreshness	Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den Amazon-S3-Bucket bereitgestellt. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung aktiviert ist (was nur möglich ist, wenn auch die Datentransformation aktiviert ist). Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden

Metrik	Beschreibung
BackupToS3.Records	Die Anzahl der Aufzeichnungen, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung aktiviert ist (was nur möglich ist, wenn auch die Datentransformation aktiviert ist). Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
BackupToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Befehle für das Backup. Amazon Data Firehose gibt diese Metrik aus, wenn die Sicherung aktiviert ist (was nur möglich ist, wenn auch die Datentransformation aktiviert ist).

Lieferung nach Snowflake

Metrik	Beschreibung
DeliveryToSnowflake.Bytes	Die Anzahl der Byte, die im angegebenen Zeitraum an Snowflake geliefert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToSnowflake.DataFreshness	Alter (von den Anfängen bei Firehose bis heute) der ältesten Aufzeichnungen in Firehose. Jeder Datensatz, der älter als dieses Alter ist, wurde an Snowflake geliefert. Beachten Sie, dass es einige Sekunden dauern kann, Daten an Snowflake zu übertragen, nachdem der Firehose-Insert-Aufruf erfolgreich war. Die Zeit, die benötigt wird, um Daten an Snowflake zu übertrage

Metrik	Beschreibung
	n, finden Sie in der Metrik. `DeliveryToSnowflake.DataCommitLatency` Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
`DeliveryToSnowflake.DataCommitLatency`	Die Zeit, die benötigt wird, bis die Daten an Snowflake übergeben werden, nachdem Firehose erfolgreich Datensätze eingefügt hat. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
`DeliveryToSnowflake.Records`	Die Anzahl der Datensätze, die im angegebenen Zeitraum an Snowflake geliefert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`DeliveryToSnowflake.Success`	Die Summe der erfolgreichen Insert-Aufrufe an Snowflake.
`DeliveryToS3.Bytes`	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Diese Metrik ist nur verfügbar, wenn die Lieferung an Snowflake fehlschlägt und Firehose versucht, fehlgeschlagene Daten auf S3 zu sichern. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte

Metrik	Beschreibung
DeliveryToS3.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Diese Metrik ist nur verfügbar, wenn die Lieferung an Snowflake fehlschlägt und Firehose versucht, fehlgeschlagene Daten auf S3 zu sichern. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Befehle. Diese Metrik ist nur verfügbar, wenn die Lieferung an Snowflake fehlschlägt und Firehose versucht, fehlgeschlagene Daten auf S3 zu sichern.
BackupToS3.DataFreshness	Alter (von Into Firehose bis heute) der ältesten Aufzeichn ung in Firehose. Jeder Datensatz, der älter als dieses Alter ist, wird im Amazon S3 S3-Bucket gesichert. Diese Metrik ist verfügbar, wenn der Firehose-Stream so konfiguriert ist, dass er alle Daten sichert. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
BackupToS3.Records	Die Anzahl der Aufzeichnungen, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Diese Metrik ist verfügbar, wenn der Firehose-Stream so konfiguriert ist, dass er alle Daten sichert. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
BackupToS3.Bytes	Die Anzahl der Bytes, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Diese Metrik ist verfügbar, wenn der Firehose-Stream so konfiguriert ist, dass er alle Daten sichert. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
BackupToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Befehle für das Backup. Firehose gibt diese Metrik aus, wenn der Firehose-Stream so konfiguriert ist, dass er alle Daten sichert.

Bereitstellung für Splunk

Metrik	Beschreibung
DeliveryToSplunk.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Splunk bereitgestellt wurden Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToSplunk.DataAckLatency	Die ungefähre Dauer, die benötigt wird, um eine Bestätigung von Splunk zu erhalten, nachdem Amazon Data Firehose ihm Daten gesendet hat. Die Beobachtung des steigenden oder fallenden Trends für diese Metrik ist nützlicher als der ungefähre absolute Wert. Steigende Trends können auf langsamere Indizierungs- und Bestätigungsraten von Splunk-Indexerstellungsmodulen hindeuten.

Metrik	Beschreibung
	Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
`DeliveryToSplunk.DataFreshness`	Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für Splunk bereitgestellt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
`DeliveryToSplunk.Records`	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Splunk bereitgestellt wurden Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`DeliveryToSplunk.Success`	Die Summe der erfolgreich indizierten Datensätze.
`DeliveryToS3.Success`	Die Summe der erfolgreichen Amazon S3 S3-Put-Befehle. Diese Metrik wird ausgegeben, wenn die Sicherung in Amazon S3 aktiviert ist.
`BackupToS3.Bytes`	Die Anzahl der Bytes, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn der Firehose-Stream so konfiguriert ist, dass er alle Dokumente sichert. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
BackupToS3.DataFreshness	Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den Amazon-S3-Bucket bereitgestellt. Amazon Data Firehose gibt diese Metrik aus, wenn der Firehose-Stream so konfiguriert ist, dass er alle Dokumente sichert. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
BackupToS3.Records	Die Anzahl der Aufzeichnungen, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn der Firehose-Stream so konfiguriert ist, dass er alle Dokumente sichert. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
BackupToS3.Success	Summe der erfolgreichen Amazon S3 S3-Put-Befehle für das Backup. Amazon Data Firehose gibt diese Metrik aus, wenn der Firehose-Stream so konfiguriert ist, dass er alle Dokumente sichert.

Lieferung an HTTP-Endpunkte

Metrik	Beschreibung
DeliveryToHttpEndpoint.Bytes	Die Anzahl der Byte, die erfolgreich an den HTTP-Endpunkt übertragen wurden.

Metrik	Beschreibung
	Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
`DeliveryToHttpEndpoint.Records`	Die Anzahl der Aufzeichnungen, die erfolgreich an den HTTP-Endpunkt übertragen wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`DeliveryToHttpEndpoint.DataFreshness`	Alter des ältesten Datensatzes in Amazon Data Firehose. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
`DeliveryToHttpEndpoint.Success`	Die Summe aller erfolgreichen Datenlieferungsanfragen an den HTTP-Endpunkt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`DeliveryToHttpEndpoint.ProcessedBytes`	Die Anzahl der versuchten verarbeiteten Bytes, einschließlich Wiederholungen.
`DeliveryToHttpEndpoint.ProcessedRecords`	Die Anzahl der versuchten Datensätze, einschließlich Wiederholungen.

Metriken zur Datenaufnahme

Themen

- Datenaufnahme über Kinesis Data Streams

- Datenaufnahme über Direct PUT
- Datenaufnahme von MSK

Datenaufnahme über Kinesis Data Streams

Metrik	Beschreibung
`DataReadFromKinesisStream.Bytes`	Wenn die Datenquelle ein Kinesis Data Stream ist, gibt diese Metrik die Anzahl der aus diesem Datenstrom gelesenen Bytes an. Diese Zahl beinhaltet wiederholte Leseoperationen aufgrund von Failovers. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
`DataReadFromKinesisStream.Records`	Wenn die Datenquelle ein Kinesis Data Stream ist, gibt diese Metrik die Anzahl der aus diesem Datenstrom gelesenen Datensätze an. Diese Zahl beinhaltet wiederholte Leseoperationen aufgrund von Failovers. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`ThrottledDescribeStream`	Die Gesamtzahl, wie oft die `DescribeStream` -Operation gedrosselt wird, wenn die Datenquelle ein Kinesis-Datenstrom ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`ThrottledGetRecords`	Die Gesamtzahl, wie oft die `GetRecords` -Operation gedrosselt wird, wenn die Datenquelle ein Kinesis-Datenstrom ist.

Metrik	Beschreibung
	Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`ThrottledGetShardIterator`	Die Gesamtzahl, wie oft die `GetShardIterator` -Operation gedrosselt wird, wenn die Datenquelle ein Kinesis-Datenstrom ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`KinesisMillisBehindLatest`	Wenn die Datenquelle ein Kinesis-Datenstrom ist, gibt diese Metrik die Anzahl der Millisekunden an, die der zuletzt gelesene Datensatz hinter dem neuesten Datensatz im Kinesis-Datenstrom liegt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Millisekunden

Datenaufnahme über Direct PUT

Metrik	Beschreibung
`BackupToS3.Bytes`	Die Anzahl der Bytes, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn die Datentran sformation für Amazon S3- oder Amazon Redshift Redshift-Ziele aktiviert ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben

Metrik	Beschreibung
	Einheiten: Byte
BackupToS3.DataFreshness	Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den Amazon-S3-Bucket bereitgestellt. Amazon Data Firehose gibt diese Metrik aus, wenn die Datentransformation für Amazon S3- oder Amazon Redshift Redshift-Ziele aktiviert ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
BackupToS3.Records	Die Anzahl der Aufzeichnungen, die im angegebenen Zeitraum zum Backup an Amazon S3 geliefert wurden. Amazon Data Firehose gibt diese Metrik aus, wenn die Datentransformation für Amazon S3- oder Amazon Redshift Redshift-Ziele aktiviert ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
BackupToS3.Success	Summe der erfolgreichen Amazon S3 S3-Put-Befehle für das Backup. Amazon Data Firehose gibt diese Metrik aus, wenn die Datentransformation für Amazon S3- oder Amazon Redshift Redshift-Ziele aktiviert ist.
BytesPerSecondLimit	Die aktuelle maximale Anzahl von Byte pro Sekunde, die ein Firehose-Stream vor der Drosselung aufnehmen kann. Um eine Erhöhung dieses Limit zu beantragen, gehen Sie zum AWS -Support Center, wählen Sie Create case (Fall erstellen) und dann Service limit increase (Service-Limiterhöhung) aus.

Metrik	Beschreibung
DeliveryToAmazonOpenSearchService.Bytes	Die Anzahl der Byte, die im angegebenen Zeitraum für OpenSearch Service indexiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToAmazonOpenSearchService.DataFreshness	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Jeder Datensatz, der älter als dieses Alter ist, wurde an den OpenSearch Service übermittelt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
DeliveryToAmazonOpenSearchService.Records	Die Anzahl der Datensätze, die im angegebenen Zeitraum für den OpenSearch Service indexiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToAmazonOpenSearchService.Success	Die Summe der erfolgreich indizierten Datensätze.
DeliveryToRedshift.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum in Amazon Redshift kopiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte

Metrik	Beschreibung
DeliveryToRedshift.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum in Amazon Redshift kopiert wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToRedshift.Success	Die Summe der erfolgreichen Amazon Redshift COPY-Befehle.
DeliveryToS3.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToS3.DataFreshness	Das Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für den S3-Bucket bereitgestellt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
DeliveryToS3.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Amazon S3 bereitgestellt wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
DeliveryToS3.Success	Die Summe der erfolgreichen Amazon S3 S3-Put-Be fehle.
DeliveryToSplunk.Bytes	Die Anzahl der Bytes, die über den angegebenen Zeitraum für Splunk bereitgestellt wurden Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
DeliveryToSplunk.D ataAckLatency	Die ungefähre Dauer, die benötigt wird, um eine Bestätigu ng von Splunk zu erhalten, nachdem Amazon Data Firehose ihm Daten gesendet hat. Die Beobachtung des steigenden oder fallenden Trends für diese Metrik ist nützlicher als der ungefähre absolute Wert. Steigende Trends können auf langsamere Indizierungs- und Bestätigungsraten von Splunk-Indexerstellungsmodulen hindeuten. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden
DeliveryToSplunk.D ataFreshness	Alter (vom Einstieg in Amazon Data Firehose bis heute) des ältesten Datensatzes in Amazon Data Firehose. Alle älteren Datensätze wurden für Splunk bereitgestellt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Sekunden

Metrik	Beschreibung
DeliveryToSplunk.Records	Die Anzahl der Datensätze, die über den angegebenen Zeitraum für Splunk bereitgestellt wurden Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
DeliveryToSplunk.Success	Die Summe der erfolgreich indizierten Datensätze.
IncomingBytes	Die Anzahl der Byte, die über den angegebenen Zeitraum erfolgreich in den Firehose-Stream aufgenommen wurden. Die Datenaufnahme kann gedrosselt werden, wenn sie eines der Firehose-Stream-Grenzwerte überschreitet. Gedrosselte Daten werden für IncomingBytes nicht mitgezählt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
IncomingPutRequests	Die Anzahl erfolgreicher PutRecord und erfolgreicher PutRecordBatch Anfragen über einen bestimmten Zeitraum. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
`IncomingRecords`	Die Anzahl der Datensätze, die im angegebenen Zeitraum erfolgreich in Firehose Firehose-Stream aufgenommen wurden. Die Datenaufnahme kann gedrosselt werden, wenn sie eines der Firehose-Stream-Grenzwerte überschreitet. Gedrosselte Daten werden für `IncomingRecords` nicht mitgezählt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`RecordsPerSecondLimit`	Die aktuelle maximale Anzahl von Datensätzen pro Sekunde, die ein Firehose-Stream vor der Drosselung aufnehmen kann. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`ThrottledRecords`	Die Anzahl der Datensätze, die gedrosselt wurden, weil die Datenaufnahme eines der Firehose-Stream-Gr enzwerte überschritten hat. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Datenaufnahme von MSK

Metrik	Beschreibung
`DataReadFromSource .Records`	Die Anzahl der Datensätze, die aus dem Quell-Kafka-Thema gelesen wurden.

Metrik	Beschreibung
	Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`DataReadFromSource.Bytes`	Die Anzahl der Bytes, die aus dem Quell-Kafka-Thema gelesen wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
`SourceThrottled.Delay`	Der Zeitraum, um den der Quell-Kafka-Cluster bei der Rückgabe der Datensätze aus dem Quell-Kafka-Thema verzögert ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Millisekunden
`BytesPerSecondLimit`	Aktuelle Durchsatzgrenze, bei der Firehose von jeder Partition des Quell-Kafka-Topics liest. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Bytes/Sekunde
`KafkaOffsetLag`	Die Differenz zwischen dem größten Offset des Datensatzes, den Firehose aus dem Quell-Kafka-Topic gelesen hat, und dem größten Offset des Datensatzes, der aus dem Quell-Kafka-Topic verfügbar ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
`FailedValidation.Records`	Die Anzahl der Datensätze, die bei der Datensatz überprüfung fehlgeschlagen sind. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`FailedValidation.Bytes`	Die Anzahl der Bytes, die bei der Datensatzüberprüfung fehlgeschlagen sind. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
`DataReadFromSource .Backpressured`	Zeigt an, dass ein Firehose-Stream beim Lesen von Datensätzen aus der Quellpartition verzögert ist, entweder weil die Anzahl BytesPerSecondLimit pro Partition überschritten wurde oder weil der normale Zustellungsfluss langsam ist oder gestoppt wurde Einheiten: boolescher Wert

Metriken auf API-Ebene CloudWatch

Der `AWS/Firehose`-Namespace enthält die folgenden API-Metriken.

Metrik	Beschreibung
`DescribeDeliverySt ream.Latency`	Die Dauer pro `DescribeDeliveryStream` -Vorgang, gemessen im angegebenen Zeitraum. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Millisekunden

Metrik	Beschreibung
DescribeDeliveryStream.Requests	Die Gesamtanzahl der DescribeDeliveryStream -Anforderungen. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
ListDeliveryStreams.Latency	Die Dauer pro ListDeliveryStream -Vorgang, gemessen im angegebenen Zeitraum. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Millisekunden
ListDeliveryStreams.Requests	Die Gesamtanzahl der ListFirehose -Anforderungen. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
PutRecord.Bytes	Die Anzahl der Byte, die PutRecord über den angegebenen Zeitraum in den Firehose-Stream eingegeben wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
PutRecord.Latency	Die Dauer pro PutRecord -Vorgang, gemessen im angegebenen Zeitraum. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Millisekunden

Metrik	Beschreibung
PutRecord.Requests	Die Gesamtanzahl der PutRecord -Anforderungen, die der Gesamtanzahl der Datensätze der PutRecord -Vorgänge entspricht. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
PutRecordBatch.Bytes	Die Anzahl der Byte, die PutRecordBatch über den angegebenen Zeitraum in den Firehose-Stream eingegeben wurden. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
PutRecordBatch.Latency	Die Dauer pro PutRecordBatch -Vorgang, gemessen im angegebenen Zeitraum. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Millisekunden
PutRecordBatch.Records	Die Gesamtanzahl der Datensätze der PutRecord Batch -Vorgänge. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
PutRecordBatch.Requests	Die Gesamtanzahl der PutRecordBatch -Anforderungen. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
PutRequestsPerSecondLimit	Die maximale Anzahl von Put-Anfragen pro Sekunde, die ein Firehose-Stream vor der Drosselung verarbeiten kann. Diese Zahl beinhaltet auch Anfragen PutRecord . PutRecordBatch Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
ThrottledDescribeStream	Die Gesamtzahl, wie oft die DescribeStream -Operation gedrosselt wird, wenn die Datenquelle ein Kinesis-Datenstrom ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
ThrottledGetRecords	Die Gesamtzahl, wie oft die GetRecords -Operation gedrosselt wird, wenn die Datenquelle ein Kinesis-Datenstrom ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Metrik	Beschreibung
`ThrottledGetShardIterator`	Die Gesamtzahl, wie oft die `GetShardIterator` -Operation gedrosselt wird, wenn die Datenquelle ein Kinesis-Datenstrom ist. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`UpdateDeliveryStream.Latency`	Die Dauer pro `UpdateDeliveryStream` -Vorgang, gemessen im angegebenen Zeitraum. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Stichproben Einheiten: Millisekunden
`UpdateDeliveryStream.Requests`	Die Gesamtanzahl der `UpdateDeliveryStream` -Anforderungen. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

CloudWatch Metriken zur Datentransformation

Wenn die Datentransformation mit Lambda aktiviert ist, enthält der `AWS/Firehose`-Namespace die folgenden Metriken.

Metrik	Beschreibung
`ExecuteProcessing.Duration`	Die Zeit, die für jeden von Firehose ausgeführten Lambda-Funktionsaufruf benötigt wird. Einheiten: Millisekunden

Metrik	Beschreibung
`ExecutePr ocessing. Success`	Die Summe der erfolgreichen Lambda-Funktionsaufrufe im Vergleich zur Summe der gesamten Lambda-Funktionsaufrufe.
`SucceedPr ocessing. Records`	Anzahl der Aufzeichnungen, die im angegebenen Zeitraum erfolgreich übertragen wurden. Einheiten: Anzahl
`SucceedPr ocessing.Bytes`	Anzahl der Bytes, die im angegebenen Zeitraum erfolgreich übertragen wurden. Einheiten: Byte

CloudWatch Protokolliert Dekomprimierungsmetriken

Wenn die Dekomprimierung für die CloudWatch Protokollzustellung aktiviert ist, umfasst der `AWS/Firehose` Namespace die folgenden Metriken.

Metrik	Beschreibung
`OutputDecompressed Bytes.Success`	Daten wurden erfolgreich dekomprimiert (in Byte) Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
`OutputDecompressed Bytes.Failed`	Fehlgeschlagene dekomprimierte Daten in Byte Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Byte
`OutputDecompressed Records.Success`	Anzahl erfolgreicher dekomprimierter Datensätze

Metrik	Beschreibung
	Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`OutputDecompressedRecords.Failed`	Anzahl der fehlgeschlagenen dekomprimierten Datensätze Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

CloudWatch Konvertierungsmetriken formatieren

Wenn die Formatkonvertierung aktiviert ist, enthält der `AWS/Firehose`-Namespace enthält die folgenden Metriken.

Metrik	Beschreibung
`SucceedConversion.Records`	Die Anzahl der erfolgreich konvertierten Datensätze. Einheiten: Anzahl
`SucceedConversion.Bytes`	Die Größe der erfolgreich konvertierten Datensätze. Einheiten: Byte
`FailedConversion.Records`	Die Anzahl der Datensätze, die nicht konvertiert werden konnten. Einheiten: Anzahl
`FailedConversion.Bytes`	Die Größe der Datensätze, die nicht konvertiert werden konnten. Einheiten: Byte

Metriken zur serverseitigen Verschlüsselung (SSE) CloudWatch

Der `AWS/Firehose`-Namespace enthält die folgenden Metriken, die sich auf SSE beziehen.

Metrik	Beschreibung
`KMSKeyAccessDenied`	Gibt an, wie oft der Dienst auf einen `KMSAccessDeniedException` für den Firehose-Stream stößt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`KMSKeyDisabled`	Gibt an, wie oft der Dienst auf einen `KMSDisabledException` für den Firehose-Stream stößt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`KMSKeyInvalidState`	Gibt an, wie oft der Dienst auf einen `KMSInvalidStateException` für den Firehose-Stream stößt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl
`KMSKeyNotFound`	Gibt an, wie oft der Dienst auf einen `KMSNotFoundException` für den Firehose-Stream stößt. Gültige Statistiken: Minimum, Maximum, Durchschnitt, Summe, Stichproben Einheiten: Anzahl

Abmessungen für Amazon Data Firehose

Verwenden Sie die `DeliveryStreamName` Dimension, um Metriken nach Firehose-Stream zu filtern.

Nutzungsmetriken von Amazon Data Firehose

Sie können CloudWatch Nutzungsmetriken verwenden, um einen Überblick über die Ressourcennutzung Ihres Kontos zu erhalten. Verwenden Sie diese Metriken, um Ihre aktuelle Servicenutzung in CloudWatch Diagrammen und Dashboards zu visualisieren.

Messwerte zur Nutzung von Servicequoten befinden sich im Namespace AWS/Usage und werden alle drei Minuten erfasst.

Derzeit ist der einzige Metrikname in diesem Namespace, der veröffentlicht wird, CloudWatch . `ResourceCount` Diese Metrik wird mit den Dimensionen `Service`, `Class`, `Type` und`Resource` veröffentlicht.

Metrik	Beschreibung
`ResourceCount`	Die Anzahl der angegebenen Ressourcen, die in Ihrem Konto ausgeführt werden. Die Ressourcen werden durch die Dimensionen definiert, die der Metrik zugeordnet sind. Die nützlichste Statistik für diese Metrik ist MAXIMUM. Sie stellt die maximale Anzahl von Ressourcen dar, die während des 3-minütigen Zeitraums verwendet wurden.

Die folgenden Dimensionen werden verwendet, um die von Amazon Data Firehose veröffentlichten Nutzungsmetriken zu verfeinern.

Dimension	Beschreibung
`Service`	Der Name des AWS Dienstes, der die Ressource enthält. Für Nutzungsmetriken von Amazon Data Firehose lautet `Firehose` der Wert für diese Dimension.

Dimension	Beschreibung
`Class`	Die Klasse der nachverfolgten Ressource. Die API-Nutzungsmetriken von Amazon Data Firehose verwenden diese Dimension mit einem Wert von`None`.
`Type`	Der Typ der nachverfolgten Ressource. Wenn die Service-Dimension `Firehose` ist, ist aktuelle der einzige gültige Wert für „Type (Typ)“ `Resource`.
`Resource`	Der Name der AWS Ressource. Wenn die Service-Dimension `Firehose` ist, ist aktuelle der einzige gültige Wert für „Resource (Ressource)“ `DeliveryStreams` .

CloudWatch Zugriffsmetriken für Amazon Data Firehose

Sie können die Metriken für Amazon Data Firehose über die CloudWatch Konsole, die Befehlszeile oder die CloudWatch API überwachen. Die folgenden Verfahren zeigen, wie Sie mithilfe dieser verschiedenen Verfahren auf die Metriken zugreifen können.

So greifen Sie über die Konsole auf Metriken zu CloudWatch

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie auf der Navigationsleiste eine Region aus.
3. Wählen Sie im Navigationsbereich Metriken aus.
4. Wählen Sie den Namespace Firehose.
5. Wählen Sie Firehose Stream Metrics oder Firehose Metrics.
6. Wählen Sie eine Metrik für das Diagramm.

Um auf Metriken zuzugreifen, verwenden Sie AWS CLI

Verwenden Sie die Listen-Metriken und get-metric-statisticsBefehle.

```
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
```

Überwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch

Amazon Data Firehose ist in Amazon CloudWatch Logs integriert, sodass Sie die spezifischen Fehlerprotokolle einsehen können, wenn der Lambda-Aufruf für die Datentransformation oder Datenzustellung fehlschlägt. Sie können die Amazon Data Firehose-Fehlerprotokollierung aktivieren, wenn Sie Ihren Firehose-Stream erstellen.

Wenn Sie die Amazon Data Firehose-Fehlerprotokollierung in der Amazon Data Firehose-Konsole aktivieren, werden in Ihrem Namen eine Protokollgruppe und entsprechende Protokollstreams für Firehose Firehose-Stream erstellt. Das Format des Protokollgruppennamens ist`/aws/kinesisfirehose/`*`delivery-stream-name`*, wobei *`delivery-stream-name`* der Name des entsprechenden Firehose-Streams steht. `DestinationDelivery`ist der Protokollstream, der erstellt und verwendet wird, um alle Fehler im Zusammenhang mit der Übermittlung an das primäre Ziel zu protokollieren. Ein weiterer Protokollstream namens `BackupDelivery`, wird nur erstellt, wenn das S3-Backup für das Ziel aktiviert ist. Der `BackupDelivery`-Protokollstream wird verwendet, um alle Fehler im Zusammenhang mit der Lieferung an das S3-Backup zu protokollieren.

Wenn Sie beispielsweise einen Firehose-Stream "MyStream" mit Amazon Redshift als Ziel erstellen und die Amazon Data Firehose-Fehlerprotokollierung aktivieren, wird Folgendes in Ihrem Namen erstellt: eine Protokollgruppe mit dem Namen `aws/kinesisfirehose/MyStream` und zwei Protokollstreams mit dem Namen und. `DestinationDelivery` BackupDelivery In diesem Beispiel wird `DestinationDelivery` verwendet, um alle Fehler im Zusammenhang mit der Übermittlung an das Amazon-Redshift-Ziel und auch an das S3-Zwischenziel zu protokollieren. `BackupDelivery`, falls das S3-Backup aktiviert ist, wird verwendet, um alle Fehler im Zusammenhang mit der Lieferung an den S3-Backup-Bucket zu protokollieren.

Sie können die Amazon Data Firehose-Fehlerprotokollierung über die AWS CLI, die API oder CloudFormation mithilfe der `CloudWatchLoggingOptions` Konfiguration aktivieren. Erstellen Sie dazu im Voraus eine Protokollgruppe und einen Protokollstream. Wir empfehlen, diese Protokollgruppe und den Protokollstream ausschließlich für die Amazon Data Firehose-Fehlerprotokollierung zu reservieren. Achten Sie außerdem darauf, dass die zugehörige IAM-Richtlinie über die Berechtigung `"logs:putLogEvents"` verfügt. Weitere Informationen finden Sie unter Zugriffskontrolle mit Amazon Data Firehose.

Beachten Sie, dass Amazon Data Firehose nicht garantiert, dass alle Versandfehlerprotokolle an Logs gesendet CloudWatch werden. In Fällen, in denen die Rate an Lieferfehlern hoch ist, nimmt Amazon Data Firehose Stichproben von Lieferfehlerprotokollen vor, bevor sie an CloudWatch Logs gesendet werden.

Für Fehlerprotokolle, die an Logs gesendet werden, wird eine geringe Gebühr erhoben. CloudWatch Weitere Informationen finden Sie unter CloudWatch Amazon-Preise.

Inhalt

Fehler bei der Datenübermittlung

Im Folgenden finden Sie eine Liste der Fehlercodes und Meldungen bei der Datenübermittlung für jedes Amazon Data Firehose-Ziel. Jede Fehlermeldung beschreibt auch die korrekte Maßnahme zur Behebung des Problems.

Fehler

Fehler bei der Amazon S3 S3-Datenübermittlung

Amazon Data Firehose kann die folgenden Amazon S3-bezogenen Fehler an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`S3.KMS.NotFoundException`	„Der angegebene AWS KMS Schlüssel wurde nicht gefunden. Wenn Sie einen Ihrer Meinung nach gültigen AWS KMS Schlüssel mit der richtigen Rolle verwenden, überprüfen Sie, ob ein Problem mit dem Konto vorliegt, an das der AWS KMS Schlüssel angehängt ist.“
`S3.KMS.RequestLimitExceeded`	„Der Grenzwert für KMS-Anfragen pro Sekunde wurde beim Versuch der Verschlüsselung von S3-Objekten überschritten. Erhöhen Sie den Grenzwert für Anforderungen pro Sekunde.“ Weitere Informationen finden Sie unter Limits im AWS Key Management Service Entwicklerhandbuch.
`S3.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Vertrauensrichtlinie für die angegebene IAM-Rolle es Amazon Data Firehose ermöglicht, die Rolle zu übernehmen, und dass die Zugriffsrichtlinie den Zugriff auf den S3-Bucket ermöglicht.“
`S3.AccountProblem`	„Es liegt ein Problem mit Ihrem AWS Konto vor, das verhindert, dass der Vorgang erfolgreich abgeschlossen werden kann. Kontaktieren Sie den AWS -Support.“
`S3.AllAccessDisabled`	„Der Zugriff auf das bereitgestellte Konto wurde deaktiviert. Wenden Sie sich an den AWS Support.“
`S3.InvalidPayer`	„Der Zugriff auf das bereitgestellte Konto wurde deaktiviert. Wenden Sie sich an den AWS Support.“
`S3.NotSignedUp`	„Das Konto ist nicht für Amazon S3 registriert. Registrieren Sie das Konto, oder verwenden Sie ein anderes Konto.“
`S3.NoSuchBucket`	"Der angegebene Bucket ist nicht vorhanden. Erstellen Sie den Bucket, oder verwenden Sie einen anderen Bucket, der existiert.“

Fehlercode	Fehlermeldungen und Informationen
`S3.MethodNotAllowed`	„Die angegebene Methode ist für diese Ressource nicht zulässig. Ändern Sie die Bucket-Richtlinie, um die korrekten Amazon-S3-Operationsberechtigungen zuzulassen.“
`InternalError`	„Interner Fehler beim Versuch des Übermittelns von Daten. Die Zustellung wird erneut versucht. Wenn der Fehler weiterhin besteht, wird er AWS zur Lösung an uns gemeldet.“
`S3.KMS.KeyDisabled`	„Der bereitgestellte KMS-Schlüssel wurde deaktiviert. Aktivieren Sie den Schlüssel oder verwenden Sie einen anderen Schlüssel.“
`S3.KMS.InvalidStateException`	„Der angegebene KMS-Schlüssel hat den Status Ungültig. Bitte verwenden Sie einen anderen Schlüssel.“
`KMS.InvalidStateException`	„Der angegebene KMS-Schlüssel hat den Status Ungültig. Bitte verwenden Sie einen anderen Schlüssel.“
`KMS.DisabledException`	„Der bereitgestellte KMS-Schlüssel wurde deaktiviert. Bitte stellen Sie den Schlüssel ein oder verwenden Sie einen anderen Schlüssel.“
`S3.SlowDown`	„Die Rate der Put-Anfragen an den angegebenen Bucket war zu hoch. Erhöhen Sie die Größe des Firehose-Stream-Puffers oder reduzieren Sie Put-Anfragen von anderen Anwendungen.“
`S3.SubscriptionRequired`	„Beim Aufrufen von S3 wurde der Zugriff verweigert. Stellen Sie sicher, dass die IAM-Rolle und der übergebene KMS-Schlüssel (falls angegeben) über ein Amazon-S3-Abonnement verfügen.“
`S3.InvalidToken`	„Das bereitgestellte Token ist falsch formatiert oder anderweitig ungültig. Bitte überprüfen Sie die angegebenen Anmeldeinformationen.“
`S3.KMS.KeyNotConfigured`	„Der KMS-Schlüssel ist nicht konfiguriert. Konfigurieren Sie Ihre KMSMaster KeyID oder deaktivieren Sie die Verschlüsselung für Ihren S3-Bucket.“

Fehlercode	Fehlermeldungen und Informationen
`S3.KMS.AsymmetricCMKNotSupported`	„Amazon S3 unterstützt nur symmetrisch CMKs. Sie können keinen asymmetrischen CMK verwenden, um Daten in Amazon S3 zu verschlüsseln. Verwenden Sie den DescribeKey KMS-Vorgang, um den Typ Ihres CMK zu ermitteln.“
`S3.IllegalLocationConstraintException`	„Firehose verwendet derzeit den globalen S3-Endpunkt für die Datenlieferung an den konfigurierten S3-Bucket. Die Region des konfigurierten S3-Buckets unterstützt den globalen S3-Endpunkt nicht. Bitte erstellen Sie einen Firehose-Stream in derselben Region wie der S3-Bucket oder verwenden Sie den S3-Bucket in der Region, die den globalen Endpunkt unterstützt.“
`S3.InvalidPrefixConfigurationException`	„Das für die Zeitstempelauswertung verwendete benutzerdefinierte s3-Präfix ist ungültig. Prüfen Sie, ob Ihr s3-Präfix gültige Ausdrücke für das aktuelle Datum und die aktuelle Uhrzeit des Jahres enthält.“
`DataFormatConversion.MalformedData`	„Ungültiges Zeichen zwischen Token gefunden.“

Fehler bei der Datenübermittlung in Apache Iceberg Tables

Informationen zu Fehlern bei der Datenübermittlung in Apache Iceberg Tables finden Sie unter. Stellen Sie Daten an Apache Iceberg Tables bereit

Fehler bei der Amazon Redshift Redshift-Datenübermittlung

Amazon Data Firehose kann die folgenden Fehler im Zusammenhang mit Amazon Redshift an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`Redshift.TableNotFound`	„Die Tabelle für das Laden von Daten wurde nicht gefunden. Stellen Sie sicher, dass die angegebene Tabelle vorhanden ist.“

Fehlercode	Fehlermeldungen und Informationen
	Die Zieltabelle in Amazon Redshift, an die Daten von S3 kopiert werden sollten, wurde nicht gefunden. Beachten Sie, dass Amazon Data Firehose die Amazon Redshift Redshift-Tabelle nicht erstellt, wenn sie nicht existiert.
`Redshift.SyntaxError`	„Der COPY-Befehl enthält einen Syntaxfehler. Wiederholen Sie den Befehl.“
`Redshift.AuthenticationFailed`	„Der bereitgestellten Benutzernamen und das Passwort konnten nicht authentifiziert werden. Geben Sie einen gültigen Benutzernamen und ein gültiges Passwort ein.“
`Redshift.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Vertrauensrichtlinie für die angegebene IAM-Rolle es Amazon Data Firehose ermöglicht, die Rolle zu übernehmen.“
`Redshift.S3BucketAccessDenied`	„Der COPY-Befehl konnte nicht auf den S3-Bucket zugreifen. Stellen Sie sicher, dass die Zugriffsrichtlinie für die angegebene IAM-Rolle den Zugriff auf den S3-Bucket ermöglicht.“
`Redshift.DataLoadFailed`	„Das Laden von Daten in die Tabelle ist fehlgeschlagen. Prüfen Sie die STL_LOAD_ERRORS-Systemtabelle für Details.“
`Redshift.ColumnNotFound`	„Eine Spalte in dem COPY-Befehl ist in der Tabelle nicht vorhanden. Geben Sie einen gültigen Spaltennamen an.“
`Redshift.DatabaseNotFound`	„Die in der Amazon-Redshift-Zielkonfiguration oder der JDBC-URL angegebene Datenbank wurde nicht gefunden. Geben Sie einen gültigen Datenbanknamen an.“
`Redshift.IncorrectCopyOptions`	„Es wurden widersprüchliche oder redundante COPY-Optionen angegeben. Einige Optionen sind in bestimmten Kombinationen nicht kompatibel. Überprüfen Sie die COPY-Befehlsreferenz", um weitere Informationen zu erhalten.“ Weitere Informationen finden Sie unter Amazon Redshift COPY-Befehl im Datenbankentwicklerhandbuch zu Amazon Redshift.

Fehlercode	Fehlermeldungen und Informationen
`Redshift.MissingColumn`	„Eine Spalte im Tabellenschema ist als NOT NULL ohne DEFAULT-Wert spezifiziert und nicht in der Spaltenliste enthalten. Schließen Sie diese Spalte aus, stellen Sie sicher, dass die geladenen Daten immer einen Wert für diese Spalte angeben, oder fügen Sie dem Amazon-Redshift-Schema für diese Tabelle einen Standardwert hinzu.“
`Redshift.ConnectionFailed`	„Die Verbindung zum angegebenen Amazon-Redshift-Cluster ist fehlgeschlagen. Stellen Sie sicher, dass die Sicherheitseinstellungen Amazon Data Firehose-Verbindungen zulassen, dass der in der Amazon Redshift Redshift-Zielkonfiguration oder der JDBC-URL angegebene Cluster oder die Datenbank korrekt ist und dass der Cluster verfügbar ist.“
`Redshift.ColumnMismatch`	„Die Anzahl der jsonpaths in dem COPY-Befehl und die Anzahl der Spalten in der Zieltabelle sollten miteinander übereinstimmen. Wiederholen Sie den Befehl.“
`Redshift.IncorrectOrMissingRegion`	„Amazon Redshift hat versucht, den falschen Regionenendpunkt für den Zugriff auf den S3-Bucket zu verwenden. Geben Sie entweder einen korrekten Regionenwert in den Optionen für den COPY-Befehl an oder stellen Sie sicher, dass sich der S3-Bucket in derselben Region wie die Amazon-Redshift-Datenbank befindet.“
`Redshift.IncorrectJsonPathsFile`	„Die bereitgestellte jsonpaths-Datei hat kein unterstütztes JSON-Format. Wiederholen Sie den Befehl.“
`Redshift.MissingS3File`	„Eine oder mehrere für Amazon Redshift erforderliche S3-Dateien wurden aus dem S3-Bucket entfernt. Überprüfen Sie die S3-Bucket-Richtlinien, und entfernen Sie das automatische Löschen von S3-Dateien.“
`Redshift.InsufficientPrivilege`	„Der Benutzer hat keine Berechtigung zum Laden von Daten in die Tabelle. Überprüfen Sie die Amazon-Redshift-Benutzerberechtigungen auf die INSERT-Berechtigung.“

Fehlercode	Fehlermeldungen und Informationen
`Redshift.ReadOnlyCluster`	„Die Abfrage kann nicht ausgeführt werden, da sich das System im Resize-Modus befindet. Versuchen Sie die Abfrage später erneut.“
`Redshift.DiskFull`	„Die Daten konnten nicht geladen werden, da der Datenträger voll ist. Erhöhen Sie die Kapazität des Amazon-Redshift-Clusters oder löschen Sie nicht benötigte Daten, um Speicherplatz freizugeben.“
`InternalError`	„Interner Fehler beim Versuch des Übermittelns von Daten. Die Zustellung wird erneut versucht. Wenn der Fehler weiterhin besteht, wird er zur Lösung an uns gemeldet.“ AWS
`Redshift.ArgumentNotSupported`	„Der Befehl COPY enthält Optionen, die nicht unterstützt werden.“
`Redshift.AnalyzeTableAccessDenied`	Zugriff verweigert. Das Kopieren von S3 nach Redshift schlägt fehl, weil die Analyse der Tabelle nur vom Tabellen- oder Datenbankbesitzer durchgeführt werden kann.“
`Redshift.SchemaNotFound`	„Das in DataTableName der Amazon Redshift Redshift-Zielkonfiguration angegebene Schema wurde nicht gefunden. Geben Sie einen gültigen Schemanamen an.“
`Redshift.ColumnSpecifiedMoreThanOnce`	„In der Spaltenliste ist eine Spalte mehrfach angegeben. Stellen Sie sicher, dass doppelte Spalten entfernt werden.“
`Redshift.ColumnNotNullWithoutDefault`	„Es gibt eine Spalte ohne DEFAULT, die ungleich Null ist und die nicht in der Spaltenliste enthalten ist. Stellen Sie sicher, dass solche Spalten in der Spaltenliste enthalten sind.“

Fehlercode	Fehlermeldungen und Informationen
`Redshift.IncorrectBucketRegion`	„Redshift hat versucht, einen Bucket in einer anderen Region als der Cluster zu verwenden. Bitte geben Sie einen Bucket an, der sich in derselben Region wie der Cluster befindet.“
`Redshift.S3SlowDown`	„Hohe Anforderungsrate an S3. Reduzieren Sie die Rate, um eine Drosselung zu vermeiden.“
`Redshift.InvalidCopyOptionForJson`	„Bitte verwenden Sie entweder Auto oder einen gültigen S3-Pfad für json copyOption.“
`Redshift.InvalidCopyOptionJSONPathFormat`	„COPY ist mit dem Fehler\" fehlgeschlagen. Ungültiges JSONPath Format. Der Array-Index liegt außerhalb des zulässigen Bereichs\“. Bitte korrigieren Sie den JSONPath Ausdruck.“
`Redshift.InvalidCopyOptionRBACAclNotAllowed`	„COPY ist mit dem Fehler \"Das RBAC-ACL-Framework kann nicht verwendet werden, solange die Rechteweitergabe nicht aktiviert ist.\“ fehlgeschlagen
`Redshift.DiskSpaceQuotaExceeded`	„Die Transaktion wurde wegen Überschreitung des Speicherkontingents abgebrochen. Geben Sie Speicherplatz frei oder fordern Sie ein erhöhtes Kontingent für das/die Schema(s) an.“
`Redshift.ConnectionsLimitExceeded`	„Das Verbindungslimit für den Benutzer wurde überschritten.“
`Redshift.SslNotSupported`	„Die Verbindung zum angegebenen Amazon-Redshift-Cluster ist fehlgeschlagen, weil der Server SSL nicht unterstützt. Bitte überprüfen Sie Ihre Cluster-Einstellungen.“

Fehlercode	Fehlermeldungen und Informationen
`Redshift.HoseNotFound`	„Der Schlauch wurde gelöscht. Bitte überprüfen Sie den Status des Schlauchs.“
`Redshift.Delimiter`	„Das copyOptions-Trennzeichen im copyCommand-Trennzeichen ist ungültig. Stellen Sie sicher, dass es sich um ein einzelnes Zeichen handelt.“
`Redshift.QueryCancelled`	„Der Benutzer hat den COPY-Vorgang abgebrochen.“
`Redshift.CompressionMismatch`	„Hose ist mit UNCOMPRESSED konfiguriert, aber copyOption enthält ein Komprimierungsformat.“
`Redshift.EncryptionCredentials`	„Für die Option ENCRYPTED sind Anmeldeinformationen im folgenden Format erforderlich: 'aws_iam_role=...;master_symmetric_key=...' oder 'aws_access_key_id=...;aws_secret_access_key=...[;token=...];master_symmetric_key=...'“
`Redshift.InvalidCopyOptions`	„Ungültige COPY-Konfigurationsoptionen.“
`Redshift.InvalidMessageFormat`	„Der Befehl Copy enthält ein ungültiges Zeichen.“
`Redshift.TransactionIdLimitReached`	„Das Transaktions-ID-Limit wurde erreicht.“
`Redshift.DestinationRemoved`	„Bitte stellen Sie sicher, dass das Redshift-Ziel existiert und in der Firehose-Konfiguration korrekt konfiguriert ist.“

Fehlercode	Fehlermeldungen und Informationen
`Redshift.OutOfMemory`	„Der Redshift-Cluster verfügt nicht mehr über genügend Arbeitsspeicher. Bitte stellen Sie sicher, dass der Cluster über ausreichende Kapazität verfügt.“
`Redshift.CannotForkProcess`	„Der Redshift-Cluster verfügt nicht mehr über genügend Arbeitsspeicher. Bitte stellen Sie sicher, dass der Cluster über ausreichende Kapazität verfügt.“
`Redshift.SslFailure`	„Die SSL-Verbindung wurde während des Handshakes geschlossen.“
`Redshift.Resize`	„Der Redshift-Cluster gibt eine neue Größe an. Firehose wird keine Daten liefern können, während die Größe des Clusters geändert wird.“
`Redshift.ImproperQualifiedName`	„Der qualifizierte Name ist falsch (zu viele Namen mit Punkten).“
`Redshift.InvalidJsonPathFormat`	„Ungültiges JSONPath Format.“
`Redshift.TooManyConnectionsException`	„Zu viele Verbindungen zu Redshift.“
`Redshift.PSQLException`	„Bei Redshift wurde eine PSQl Ausnahme beobachtet.“
`Redshift.DuplicateSecondsSpecification`	„Doppelte Sekundenangabe im date/time Format.“

Fehlercode	Fehlermeldungen und Informationen
`Redshift.RelationCouldNotBeOpened`	„Redshift-Fehler aufgetreten, Beziehung konnte nicht geöffnet werden. Überprüfen Sie die Redshift-Protokolle für die angegebene Datenbank.“
`Redshift.TooManyClients`	„Ich bin auf zu viele Kunden gestoßen, mit Ausnahme von Redshift. Überprüfen Sie die maximale Anzahl der Verbindungen zur Datenbank erneut, wenn mehrere Produzenten gleichzeitig in die Datenbank schreiben.“

Fehler bei der Lieferung von Snowflake-Daten

Firehose kann die folgenden Snowflake-bezogenen Fehler an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`Snowflake.InvalidUrl`	„Firehose kann keine Verbindung zu Snowflake herstellen. Bitte stellen Sie sicher, dass die Konto-URL in der Snowflake-Zielkonfiguration korrekt angegeben ist.“
`Snowflake.InvalidUser`	„Firehose kann keine Verbindung zu Snowflake herstellen. Bitte stellen Sie sicher, dass der Benutzer in der Snowflake-Zielkonfiguration korrekt angegeben ist.“
`Snowflake.InvalidRole`	„Die angegebene Snowflake-Rolle existiert nicht oder ist nicht autorisiert. Bitte stellen Sie sicher, dass die Rolle dem angegebenen Benutzer gewährt wurde.“
`Snowflake.InvalidTable`	„Die mitgelieferte Tabelle existiert nicht oder ist nicht autorisiert“
`Snowflake.InvalidSchema`	„Das angegebene Schema existiert nicht oder ist nicht autorisiert“

Fehlercode	Fehlermeldungen und Informationen
`Snowflake.InvalidDatabase`	„Die angegebene Datenbank existiert nicht oder ist nicht autorisiert“
`Snowflake.InvalidPrivateKeyOrPassphrase`	„Der angegebene private Schlüssel oder die angegebene Passphrase ist nicht gültig. Beachten Sie, dass der angegebene private Schlüssel ein gültiger privater PEM-RSA-Schlüssel sein sollte.“
`Snowflake.MissingColumns`	„Die Einfügeanforderung wurde aufgrund fehlender Spalten in der Eingabe-Payload abgelehnt. Stellen Sie sicher, dass Werte für alle Spalten angegeben sind, für die keine NULL-Werte zulässig sind.“
`Snowflake.ExtraColumns`	„Die Einfügeanforderung wurde aufgrund zusätzlicher Spalten abgelehnt. Spalten, die in der Tabelle nicht vorhanden sind, sollten nicht angegeben werden.“
`Snowflake.InvalidInput`	„Die Lieferung ist aufgrund eines ungültigen Eingabeformats fehlgeschlagen. Stellen Sie sicher, dass die angegebene Eingabe-Payload im akzeptablen JSON-Format vorliegt.“
`Snowflake.IncorrectValue`	„Die Lieferung ist aufgrund eines falschen Datentyps in der Eingabe-Payload fehlgeschlagen. Stellen Sie sicher, dass die in der Eingabe-Payload angegebenen JSON-Werte dem in der Snowflake-Tabellen definition deklarierten Datentyp entsprechen.

Fehler bei der Bereitstellung von Splunk-Daten

Amazon Data Firehose kann die folgenden Splunk-bezogenen Fehler an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`Splunk.ProxyWithou`	„Wenn Sie einen Proxy (ELB oder ein anderer) zwischen Amazon Data Firehose und dem HEC-Knoten haben, müssen Sie Sticky-Sitzungen aktivieren, um ACKs HEC zu unterstützen.“

Fehlercode	Fehlermeldungen und Informationen
`tStickySessions`	
`Splunk.DisabledToken`	"Das angegebene HEC-Token ist nicht aktiviert. Aktivieren Sie das Token, um zu ermöglichen, dass Daten an Splunk geliefert werden."
`Splunk.InvalidToken`	"Das angegebene HEC-Token ist ungültig. Aktualisieren Sie Amazon Data Firehose mit einem gültigen HEC-Token.“
`Splunk.InvalidDataFormat`	"Die Daten sind nicht ordnungsgemäß formatiert. Informationen, wie Daten ordnungsgemäß für Raw-Format oder Ereignis-HEC-Endpunkte formatiert werden, finden Sie unter Splunk-Ereignisdaten."
`Splunk.InvalidIndex`	"Die HEC-Token oder Eingabe ist mit einem ungültigen Index konfiguriert worden. Überprüfen Sie Ihre Indexkonfiguration und versuchen Sie es erneut."
`Splunk.ServerError`	„Datenbereitstellung an Splunk ist aufgrund eines Server-Fehlers aus dem HEC-Knoten fehlgeschlagen. Amazon Data Firehose versucht erneut, die Daten zu senden, wenn die Wiederholungsdauer in Ihrer Amazon Data Firehose größer als 0 ist. Wenn alle Wiederholungen fehlschlagen, sichert Amazon Data Firehose die Daten auf Amazon S3.“
`Splunk.DisabledAck`	"Indexbestätigung für den HEC-Token ist nicht aktiviert. Aktivieren Sie die Indexbestätigung und versuchen Sie es erneut. Weitere Informationen finden Sie unter Aktivieren der Indexbestätigung.
`Splunk.AckTimeout`	"Habe keine Bestätigung von HEC vor Zeitablauf des HEC-Bestätigungs-Timeout erhalten. Trotz der Anerkennung des Timeouts ist es möglich, dass die Daten erfolgreich in Splunk indiziert wurden. Amazon Data Firehose sichert Daten, für die das Bestätigungs-Timeout abgelaufen ist, in Amazon S3.“
`Splunk.MaxRetriesFailed`	"Fehler beim Senden von Daten an Splunk oder beim Erhalt einer Bestätigung. Überprüfen Sie Ihre HEC-Gesundheit und versuchen Sie es erneut."

Fehlercode	Fehlermeldungen und Informationen
`Splunk.ConnectionTimeout`	"Zeitlimit bei der Verbindung zu Splunk ist überschritten. Dies kann ein vorübergehender Fehler sein, die Anforderung wird wiederholt. Amazon Data Firehose sichert die Daten auf Amazon S3, falls alle Wiederholungsversuche fehlschlagen.“
`Splunk.InvalidEndpoint`	"Es konnte keine Verbindung mit dem HEC-Endpunkt hergestellt werden. Stellen Sie sicher, dass die HEC-Endpunkt-URL gültig und von Amazon Data Firehose aus erreichbar ist.“
`Splunk.ConnectionClosed`	"Fehler beim Senden der Daten an Splunk aufgrund eines Verbindungsfehlers. Dies kann ein vorübergehender Fehler sein. Eine Verlängerung der Wiederholungsdauer in Ihrer Amazon Data Firehose-Konfiguration kann vor solchen vorübergehenden Ausfällen schützen.“
`Splunk.SSLUnverified`	"Es konnte keine Verbindung mit dem HEC-Endpunkt hergestellt werden. Der Host stimmt nicht mit dem vom Peer bereitgestellten Zertifikat überein. Stellen Sie sicher, dass das Zertifikat und der Host gültig sind."
`Splunk.SSLHandshake`	"Es konnte keine Verbindung mit dem HEC-Endpunkt hergestellt werden. Stellen Sie sicher, dass das Zertifikat und der Host gültig sind."
`Splunk.URLNotFound`	„Die angeforderte URL wurde auf dem Splunk-Server nicht gefunden. Bitte überprüfen Sie den Splunk-Cluster und stellen Sie sicher, dass er korrekt konfiguriert ist.“
`Splunk.ServerError.ContentTooLarge`	„Die Datenzustellung an Splunk ist aufgrund eines Serverfehlers mit dem statusCode: 413, Nachricht: Die Anfrage, die Ihr Kunde gesendet hat, war zu groß, fehlgeschlagen. Informationen zur Konfiguration von max_content_length finden Sie in der Splunk-Dokumentation.“
`Splunk.IndexerBusy`	„Datenbereitstellung an Splunk ist aufgrund eines Server-Fehlers aus dem HEC-Knoten fehlgeschlagen. Stellen Sie sicher, dass der HEC-Endpunkt oder der Elastic Load Balancer erreichbar und fehlerfrei sind.“

Fehlercode	Fehlermeldungen und Informationen
`Splunk.ConnectionRecycled`	„Die Verbindung von Firehose zu Splunk wurde recycelt. Die Lieferung wird erneut versucht.“
`Splunk.AcknowledgementsDisabled`	„Bei POST konnten keine Bestätigungen abgerufen werden. Stellen Sie sicher, dass Bestätigungen auf dem HEC-Endpunkt aktiviert sind.“
`Splunk.InvalidHecResponseCharacter`	„In der HEC-Antwort wurden ungültige Zeichen gefunden. Achten Sie darauf, den Dienst und die HEC-Konfiguration zu überprüfen.“

ElasticSearch Fehler bei der Datenübermittlung

Amazon Data Firehose kann die folgenden ElasticSearch Fehler an CloudWatch Logs senden.

Fehlercode	Fehlermeldungen und Informationen
`ES.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die bereitgestellte IAM-Rolle, die Firehose zugeordnet ist, nicht gelöscht wird.“
`ES.ResourceNotFound`	„Die angegebene AWS Elasticsearch-Domain existiert nicht.“

Fehler bei der Bereitstellung von HTTPS-Endpunktdaten

Amazon Data Firehose kann die folgenden Fehler im Zusammenhang mit HTTP-Endpunkten an Logs senden. CloudWatch Wenn keiner dieser Fehler mit dem aufgetretenen Problem übereinstimmt, lautet der Standardfehler wie folgt: „Beim Versuch, Daten zu liefern, ist ein interner Fehler aufgetreten. Die Lieferung wird erneut versucht. Wenn der Fehler weiterhin besteht, wird er zur Lösung an uns gemeldet.“ AWS

Fehlercode	Fehlermeldungen und Informationen
`HttpEndpoint.RequestTimeout`	Bei der Zustellung wurde das Zeitlimit überschritten, bevor eine Antwort eingegangen ist, und es wird erneut versucht. Wenn dieser Fehler weiterhin besteht, wenden Sie sich an das AWS -Firehose-Serviceteam.
`HttpEndpoint.ResponseTooLarge`	„Die vom Endpunkt empfangene Antwort ist zu umfangreich. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“
`HttpEndpoint.InvalidResponseFromDestination`	„Die vom angegebenen Endpunkt empfangene Antwort ist ungültig. Wenden Sie sich an den Besitzer des Endpunkts, um das Problem zu lösen.“
`HttpEndpoint.DestinationException`	„Die folgende Antwort wurde vom Endpunktziel empfangen.“
`HttpEndpoint.ConnectionFailed`	„Es konnte keine Verbindung zum Zielendpunkt hergestellt werden. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“
`HttpEndpoint.ConnectionReset`	„Die Verbindung mit dem Endpunkt konnte nicht aufrechterhalten werden. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“
`HttpEndpoint.ConnectionReset`	„Die Verbindung mit dem Endpunkt konnte nicht aufrechterhalten werden. Bitte wenden Sie sich an den Besitzer des Endpunkts.“
`HttpEndpoint.ResponseReasonPhraseExceededLimit`	„Der vom Endpunkt empfangene Satz zur Begründung der Antwort überschreitet den konfigurierten Grenzwert von 64 Zeichen.“

Fehlercode	Fehlermeldungen und Informationen
`HttpEndpoint.InvalidResponseFromDestination`	„Die vom Endpunkt empfangene Antwort ist ungültig. Weitere Informationen finden Sie unter Problembehandlung bei HTTP-Endpunkten in der Firehose-Dokumentation. Grund "
`HttpEndpoint.DestinationException`	„Die Lieferung an den Endpunkt war nicht erfolgreich. Weitere Informationen finden Sie unter Problembehandlung bei HTTP-Endpunkten in der Firehose-Dokumentation. Die Antwort wurde mit dem Statuscode“ empfangen
`HttpEndpoint.InvalidStatusCode`	„Ich habe einen ungültigen Antwortstatuscode erhalten.“
`HttpEndpoint.SSLHandshakeFailure`	„Ein SSL-Handshake mit dem Endpunkt konnte nicht abgeschlossen werden. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“
`HttpEndpoint.SSLHandshakeFailure`	„Ein SSL-Handshake mit dem Endpunkt konnte nicht abgeschlossen werden. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“
`HttpEndpoint.SSLFailure`	„Ein TLS-Handshake mit dem Endpunkt konnte nicht abgeschlossen werden. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“
`HttpEndpoint.SSLHandshakeCertificatePathFailure`	„Ein SSL-Handshake mit dem Endpunkt konnte aufgrund eines ungültigen Zertifizierungspfads nicht abgeschlossen werden. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“

Fehlercode	Fehlermeldungen und Informationen
`HttpEndpoint.SSLHandshakeCertificatePathValidationFailure`	„Ein SSL-Handshake mit dem Endpunkt konnte aufgrund eines Fehlers bei der Validierung des Zertifizierungspfads nicht abgeschlossen werden. Wenden Sie sich an den Besitzer des Endpunkts, um dieses Problem zu lösen.“
`HttpEndpoint.MakeRequestFailure.IllegalUriException`	„Die HttpEndpoint Anfrage ist aufgrund einer ungültigen Eingabe in der URI fehlgeschlagen. Bitte stellen Sie sicher, dass alle Zeichen in der Eingabe-URI gültig sind.“
`HttpEndpoint.MakeRequestFailure.IllegalCharacterInHeaderValue`	„Die HttpEndpoint Anfrage ist aufgrund eines ungültigen Antwortfehlers fehlgeschlagen. Ungültiges Zeichen '\n' im Header-Wert.“
`HttpEndpoint.IllegalResponseFailure`	„Die HttpEndpoint Anfrage ist aufgrund eines ungültigen Antwortfehlers fehlgeschlagen. Die HTTP-Nachricht darf nicht mehr als einen Content-Type-Header enthalten.“
`HttpEndpoint.IllegalMessageStart`	„Die HttpEndpoint Anfrage ist aufgrund eines ungültigen Antwortfehlers fehlgeschlagen. Ungültiger Start der HTTP-Nachricht. Weitere Informationen finden Sie unter Problembehandlung bei HTTP-Endpunkten in der Firehose-Dokumentation.“

Fehler bei der Lieferung von Amazon OpenSearch Service-Daten

Für das OpenSearch Serviceziel sendet Amazon Data Firehose Fehler an CloudWatch Logs, sobald sie vom OpenSearch Service zurückgegeben werden.

Zusätzlich zu Fehlern, die bei OpenSearch Clustern auftreten können, können die folgenden zwei Fehler auftreten:

- Authentication/authorization error occurs during attempt to deliver data to destination OpenSearch Service cluster. This can happen due to any permission issues and/orzeitweise, wenn Ihre Amazon Data OpenSearch Firehose-Zielservice-Domainkonfiguration geändert wird. Bitte überprüfen Sie die Clusterrichtlinie und die Rollenberechtigungen.
- Daten konnten aufgrund von Fehlern nicht an den OpenSearch Ziel-Servicecluster übermittelt werden. authentication/authorization Dies kann aufgrund von and/or zeitweise auftretenden Berechtigungsproblemen passieren, wenn die Konfiguration Ihrer Amazon Data OpenSearch Firehose-Zielservice-Domain geändert wird. Bitte überprüfen Sie die Clusterrichtlinie und die Rollenberechtigungen.

Fehlercode	Fehlermeldungen und Informationen
`OS.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Vertrauensrichtlinie für die angegebene IAM-Rolle Firehose erlaubt, die Rolle zu übernehmen, und dass die Zugriffsrichtlinie den Zugriff auf die Amazon OpenSearch Service API ermöglicht.“
`OS.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Vertrauensrichtlinie für die angegebene IAM-Rolle Firehose erlaubt, die Rolle zu übernehmen, und dass die Zugriffsrichtlinie den Zugriff auf die Amazon OpenSearch Service API ermöglicht.“
`OS.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die bereitgestellte IAM-Rolle, die Firehose zugeordnet ist, nicht gelöscht wird.“
`OS.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die bereitgestellte IAM-Rolle, die Firehose zugeordnet ist, nicht gelöscht wird.“
`OS.ResourceNotFound`	„Die angegebene Amazon OpenSearch Service-Domain existiert nicht.“

Fehlercode	Fehlermeldungen und Informationen
`OS.ResourceNotFound`	„Die angegebene Amazon OpenSearch Service-Domain existiert nicht.“
`OS.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Vertrauensrichtlinie für die angegebene IAM-Rolle Firehose erlaubt, die Rolle zu übernehmen, und dass die Zugriffsrichtlinie den Zugriff auf die Amazon OpenSearch Service API ermöglicht.“
`OS.RequestTimeout`	„Bei der Anfrage an den Amazon OpenSearch Service-Cluster oder bei der OpenSearch serverlosen Erfassung wurde das Zeitlimit überschritten. Stellen Sie sicher, dass der Cluster oder die Sammlung über ausreichend Kapazität für den aktuellen Workload verfügt.“
`OS.ClusterError`	„Der Amazon OpenSearch Service-Cluster hat einen nicht näher bezeichneten Fehler zurückgegeben.“
`OS.RequestTimeout`	„Bei der Anfrage an den Amazon OpenSearch Service-Cluster wurde das Zeitlimit überschritten. Stellen Sie sicher, dass der Cluster über ausreichend Kapazität für den aktuellen Workload verfügt.“
`OS.ConnectionFailed`	„Probleme beim Herstellen einer Verbindung zum Amazon OpenSearch Service-Cluster oder zur OpenSearch Serverless Collection. Stellen Sie sicher, dass der Cluster oder die Sammlung fehlerfrei und erreichbar ist.“
`OS.ConnectionReset`	„Die Verbindung mit dem Amazon OpenSearch Service-Cluster oder der OpenSearch serverlosen Sammlung konnte nicht aufrechterhalten werden. Wenden Sie sich an den Besitzer des Clusters oder der Sammlung, um dieses Problem zu lösen.“
`OS.ConnectionReset`	„Probleme bei der Aufrechterhaltung der Verbindung mit dem Amazon OpenSearch Service-Cluster oder der OpenSearch serverlosen Sammlung. Stellen Sie sicher, dass der Cluster oder die Sammlung intakt ist und über ausreichend Kapazität für den aktuellen Workload verfügt.“

Fehlercode	Fehlermeldungen und Informationen
`OS.ConnectionReset`	„Probleme bei der Aufrechterhaltung der Verbindung mit dem Amazon OpenSearch Service-Cluster oder der OpenSearch serverlosen Sammlung. Stellen Sie sicher, dass der Cluster oder die Sammlung intakt ist und über ausreichend Kapazität für den aktuellen Workload verfügt.“
`OS.AccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Zugriffsrichtlinie auf dem Amazon OpenSearch Service-Cluster Zugriff auf die konfigurierte IAM-Rolle gewährt.“
`OS.ValidationException`	„Der OpenSearch Cluster hat eine ESService Ausnahme zurückgegeben. Einer der Gründe ist, dass der Cluster auf OS 2.x oder höher aktualisiert wurde, aber der TypeName Parameter für den Schlauch immer noch konfiguriert ist. Aktualisieren Sie die Schlauchkonfiguration, indem Sie TypeName für eine leere Zeichenfolge angeben, oder ändern Sie den Endpunkt auf den Cluster, der den Type-Parameter unterstützt.“
`OS.ValidationException`	„Das Mitglied muss dem Muster für reguläre Ausdrücke entsprechen: [a-z] [a-z0-9\\-]+
`OS.JsonParseException`	„Der Amazon OpenSearch Service-Cluster hat a zurückgegeben JsonParseException. Stellen Sie sicher, dass die eingegebenen Daten gültig sind.“
`OS.AmazonOpenSearchServiceParseException`	„Der Amazon OpenSearch Service-Cluster hat eine zurückgegeben AmazonOpenSearchServiceParseException. Stellen Sie sicher, dass die eingegebenen Daten gültig sind.“
`OS.ExplicitIndexInBulkNotAllowed`	„Stellen Sie sicher, dass rest.action.multi.allow_explicit_index im Amazon Service-Cluster auf true gesetzt ist.“ OpenSearch
`OS.ClusterError`	„Der Amazon OpenSearch Service-Cluster oder die OpenSearch serverlose Sammlung haben einen nicht näher bezeichneten Fehler zurückgegeben.“

Fehlercode	Fehlermeldungen und Informationen
`OS.ClusterBlockException`	„Der Cluster hat a zurückgegeben. ClusterBlockException Es ist möglicherweise überlastet.“
`OS.InvalidARN`	„Der angegebene Amazon OpenSearch Service ARN ist ungültig. Bitte überprüfen Sie Ihre DeliveryStream Konfiguration.“
`OS.MalformedData`	„Ein oder mehrere Datensätze sind fehlerhaft formatiert. Bitte stellen Sie sicher, dass es sich bei jedem Datensatz um ein einzelnes gültiges JSON-Objekt handelt und dass er keine Zeilenumbrüche enthält.“
`OS.InternalError`	„Interner Fehler beim Versuch des Übermittelns von Daten. Die Lieferung wird erneut versucht. Wenn der Fehler weiterhin besteht, wird er AWS zur Lösung an uns gemeldet.“
`OS.AliasWithMultipleIndicesNotAllowed`	„Alias ist mit mehr als einem Index verknüpft. Stellen Sie sicher, dass dem Alias nur ein Index zugeordnet ist.“
`OS.UnsupportedVersion`	„Amazon OpenSearch Service 6.0 wird derzeit nicht von Amazon Data Firehose unterstützt. Wenden Sie sich für weitere Informationen an den AWS Support.“
`OS.CharConversionException`	„Ein oder mehrere Datensätze enthielten ein ungültiges Zeichen.“
`OS.InvalidDomainNameLength`	„Die Länge des Domainnamens liegt nicht innerhalb der gültigen Betriebssystemgrenzen.“
`OS.VPCDomainNotSupported`	„Amazon OpenSearch Service-Domains innerhalb VPCs werden derzeit nicht unterstützt.“

Fehlercode	Fehlermeldungen und Informationen
`OS.ConnectionError`	„Der HTTP-Server hat die Verbindung unerwartet geschlossen. Bitte überprüfen Sie den Zustand des Amazon OpenSearch Service-Clusters oder der OpenSearch Serverless Collection.“
`OS.LargeFieldData`	„Der Amazon OpenSearch Service-Cluster hat die Anfrage abgebrochen, da sie Felddaten enthielt, die größer als zulässig waren.“
`OS.BadGateway`	„Der Amazon OpenSearch Service-Cluster oder die OpenSearch Serverless Collection haben die Anfrage mit der folgenden Antwort abgebrochen: 502 Bad Gateway.“
`OS.ServiceException`	„Es wurde ein Fehler vom Amazon OpenSearch Service-Cluster oder der OpenSearch serverlosen Sammlung empfangen. Wenn sich der Cluster oder die Sammlung hinter einer VPC befindet, stellen Sie sicher, dass die Netzwerkkonfiguration Konnektivität zulässt.“
`OS.GatewayTimeout`	„Firehose hat beim Herstellen einer Verbindung zum Amazon OpenSearch Service-Cluster oder zur OpenSearch Serverless Collection Timeout-Fehler festgestellt.“
`OS.MalformedData`	„Amazon Data Firehose unterstützt keine Amazon OpenSearch Service Bulk-API-Befehle innerhalb des Firehose-Datensatzes.“
`OS.ResponseEntryCountMismatch`	„Die Antwort der Bulk-API enthielt mehr Einträge als die Anzahl der gesendeten Datensätze. Stellen Sie sicher, dass jeder Datensatz nur ein JSON-Objekt enthält und dass es keine Zeilenumbrüche gibt.“

Fehler Lambda Lambda-Aufruf

Amazon Data Firehose kann die folgenden Lambda-Aufruffehler an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`Lambda.AssumeRoleAccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Vertrauensrichtlinie für die angegebene IAM-Rolle es Amazon Data Firehose ermöglicht, die Rolle zu übernehmen.“

Fehlercode	Fehlermeldungen und Informationen
`Lambda.InvokeAccessDenied`	„Zugriff verweigert. Stellen Sie sicher, dass die Zugriffsrichtlinie den Zugriff auf die Lambda-Funktion zulässt.“
`Lambda.JsonProcessingException`	„Bei der Analyse der zurückgegebenen Datensätze von der Lambda-Funktion ist ein Fehler aufgetreten. Stellen Sie sicher, dass die zurückgesandten Datensätze dem von Amazon Data Firehose geforderten Statusmodell entsprechen.“ Weitere Informationen finden Sie unter Erforderliche Parameter für die Datentransformation.
`Lambda.InvokeLimitExceeded`	„Das Limit für die gleichzeitige Lambda-Ausführung wurde überschritten. Erhöhen Sie das Limit für die gleichzeitige Ausführung.“ Weitere Informationen finden Sie unter AWS Lambda Limits im AWS Lambda -Entwicklerhandbuch.
`Lambda.DuplicatedRecordId`	„Es wurden mehrere Datensätze mit der selben Datensatz-ID zurückgegeben. Stellen Sie sicher, dass die Lambda-Funktion IDs für jeden Datensatz einen eindeutigen Datensatz zurückgibt.“ Weitere Informationen finden Sie unter Erforderliche Parameter für die Datentransformation.
`Lambda.MissingRecordId`	„Ein oder mehrere Datensätze IDs wurden nicht zurückgegeben. Stellen Sie sicher, dass die Lambda-Funktion alle empfangenen Datensätze zurückgibt IDs.“ Weitere Informationen finden Sie unter Erforderliche Parameter für die Datentransformation.
`Lambda.ResourceNotFound`	„Die angegebene Lambda-Funktion ist nicht vorhanden. Verwenden Sie eine andere Funktion, die vorhanden ist.“

Fehlercode	Fehlermeldungen und Informationen
`Lambda.InvalidSubnetIDException`	„Die angegebenen Subnetz-ID in der Lambda-Funktions-VPC-Konfiguration ist ungültig. Stellen Sie sicher, dass die Subnetz-ID gültig ist.“
`Lambda.InvalidSecurityGroupIDException`	„Die angegebene Sicherheitsgruppen-ID in der Lambda-Funktions-VPC-Konfiguration ist ungültig. Stellen Sie sicher, dass die Sicherheitsgruppen-ID gültig ist.“
`Lambda.SubnetIPAddressLimitReachedException`	„AWS Lambda konnte den VPC-Zugriff für die Lambda-Funktion nicht einrichten, da für ein oder mehrere konfigurierte Subnetze keine verfügbaren IP-Adressen verfügbar sind. Erhöhen Sie das Limit für IP-Adressen.“ Weitere Informationen zu diesen Limits finden Sie unter Amazon VPC-Limits – VPC und Subnetze im Amazon-VPC-Benutzerhandbuch.
`Lambda.ENILimitReachedException`	„AWS Lambda konnte in der VPC, die als Teil der Lambda-Funktionskonfiguration angegeben wurde, kein Elastic Network Interface (ENI) erstellen, da das Limit für Netzwerkschnittstellen erreicht wurde. Erhöhen Sie das Limit für Netzwerkschnittstellen.“ Weitere Informationen zu diesen Limits finden Sie unter Amazon VPC-Limits – Netzwerkschnittstellen im Amazon-VPC-Benutzerhandbuch.
`Lambda.FunctionTimedOut`	Der Lambda-Funktions-Aufruf hat das Zeitlimit überschritten. Erhöhen Sie die Timeout-Einstellung in der Lambda-Funktion. Weitere Informationen erhalten Sie unter Zeitüberschreitung der Funktion konfigurieren.

Fehlercode	Fehlermeldungen und Informationen
`Lambda.FunctionError`	Dies kann an einen der folgenden zurückzuführen sein: • Ungültige Ausgabestruktur. Überprüfen Sie Ihre Funktion und stellen Sie sicher, dass die Ausgabe das erforderliche Format hat. Stellen Sie außerdem sicher, dass die verarbeiteten Datensätze den gültigen Ergebnisstatus von `Dropped`, `Ok` oder `ProcessingFailed` enthalten. • Die Lambda-Funktion wurde erfolgreich aufgerufen, hat aber ein Fehlerergebnis zurückgegeben. • Lambda konnte die Umgebungsvariablen nicht entschlüsseln, da der KMS-Zugriff verweigert wurde. Überprüfen Sie die KMS-Schlüsseleinstellungen der Funktion sowie die Schlüsselrichtlinie. Weitere Informationen finden Sie unter Fehlerbehebung beim Schlüsselzugriff.
`Lambda.FunctionRequestTimedOut`	Amazon Data Firehose ist beim Aufrufen von Lambda auf einen Konfigurationsfehler gestoßen, der beim Aufrufen von Lambda nicht vor dem Timeout der Anforderung abgeschlossen wurde. Rufen Sie den Lambda-Code erneut auf, um zu überprüfen, ob der Lambda-Code über das konfigurierte Timeout hinaus ausgeführt werden soll. Wenn ja, sollten Sie die Lambda-Konfigurationseinstellungen, einschließlich Speicher und Timeout, optimieren. Weitere Informationen erhalten Sie unter Konfigurieren von Lambda-Funktionsoptionen.
`Lambda.TargetServerFailedToRespond`	Amazon Data Firehose ist auf einen Fehler gestoßen. Der Zielserver hat beim Aufrufen des AWS Lambda-Dienstes nicht reagiert.
`Lambda.InvalidZipFileException`	Amazon Data Firehose ist InvalidZipFileException beim Aufrufen der Lambda-Funktion aufgetreten. Überprüfen Sie Ihre Lambda-Funktionskonfigurationseinstellungen und die Lambda-Code-ZIP-Datei.

Fehlercode	Fehlermeldungen und Informationen
`Lambda.InternalServerError`	„Amazon Data Firehose ist InternalServerError beim Aufrufen des AWS Lambda-Service aufgetreten. Amazon Data Firehose versucht erneut, Daten mit einer bestimmten Anzahl von Malen zu senden. Sie können die Wiederholungsoptionen mit der Taste oder angeben oder überschreiben. `CreateDeliveryStream` `UpdateDestination` APIs Wenn der Fehler weiterhin besteht, wenden Sie sich an das AWS Lambda-Supportteam.
`Lambda.ServiceUnavailable`	Amazon Data Firehose ist ServiceUnavailableException beim Aufrufen des AWS Lambda-Service aufgetreten. Amazon Data Firehose versucht erneut, Daten mit einer bestimmten Anzahl von Malen zu senden. Sie können die Wiederholungsoptionen mit der Taste oder angeben oder überschreiben. `CreateDeliveryStream` `UpdateDestination` APIs Wenn der Fehler weiterhin besteht, wenden Sie sich an den AWS Lambda-Support.
`Lambda.InvalidSecurityToken`	Die Lambda-Funktion kann aufgrund eines ungültigen Sicherheitstokens nicht aufgerufen werden. Partitionsübergreifender Lambda-Aufruf wird nicht unterstützt.

Fehlercode	Fehlermeldungen und Informationen
`Lambda.InvocationFailure`	Dies kann an einen der folgenden zurückzuführen sein: • Amazon Data Firehose ist beim Aufrufen von AWS Lambda auf Fehler gestoßen. Der Vorgang wird erneut versucht; wenn der Fehler bestehen bleibt, wird er zur Lösung an AWS gemeldet.“ • Amazon Data Firehose ist auf eine KMSInvalid StateException von Lambda gestoßen. Lambda konnte die Umgebungsvariablen nicht entschlüsseln, da der verwendete KMS-Schlüssel einen ungültigen Status für Entschlüsseln hat. Überprüfen Sie die Einstellungen des KMS-Schlüssels der Lambda-Funktion. • Amazon Data Firehose ist auf einen Fehler AWS LambdaException von Lambda gestoßen. Lambda konnte das bereitgestellte Container-Image nicht initialisieren. Überprüfen Sie das Bild. • Amazon Data Firehose ist beim Aufrufen AWS von Lambda auf Timeoutfehler gestoßen. Das maximal unterstützte Funktions-Timeout beträgt 5 Minuten. Weitere Informationen finden Sie unter Data Transformation Execution Duration.
`Lambda.JsonMappingException`	Bei der Analyse der zurückgegebenen Datensätze von der Lambda-Funktion ist ein Fehler aufgetreten. Stellen Sie sicher, dass das Datenfeld base-64-codiert ist.

Kinesis-Aufruffehler

Amazon Data Firehose kann die folgenden Kinesis-Aufruffehler an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`Kinesis.AccessDenied`	„Beim Aufrufen von Kinesis wurde der Zugriff verweigert. Stellen Sie sicher, dass die Zugriffsrichtlinie für die verwendete IAM-Rolle den Zugriff auf die entsprechende Kinesis APIs ermöglicht.“

Fehlercode	Fehlermeldungen und Informationen
`Kinesis.ResourceNotFound`	„Firehose konnte nicht aus dem Stream lesen. Wenn der Firehose mit Kinesis Stream verbunden ist, ist der Stream möglicherweise nicht vorhanden, oder der Shard wurde möglicherweise zusammengeführt oder aufgeteilt. Wenn der Firehose DirectPut vom Typ ist, existiert der Firehose möglicherweise nicht mehr.“
`Kinesis.SubscriptionRequired`	„Beim Aufrufen von Kinesis wurde der Zugriff verweigert. Stellen Sie sicher, dass die für den Kinesis-Stream-Zugriff übergebene IAM-Rolle ein AWS Kinesis-Abonnement hat.“
`Kinesis.Throttling`	„Beim Aufrufen von Kinesis ist ein Drosselungsfehler aufgetreten. Dies kann daran liegen, dass andere Anwendungen denselben Stream APIs wie den Firehose-Stream aufrufen, oder daran, dass Sie zu viele Firehose-Streams mit demselben Kinesis-Stream als Quelle erstellt haben.“
`Kinesis.Throttling`	„Beim Aufrufen von Kinesis ist ein Drosselungsfehler aufgetreten. Dies kann daran liegen, dass andere Anwendungen denselben Stream APIs wie den Firehose-Stream aufrufen, oder daran, dass Sie zu viele Firehose-Streams mit demselben Kinesis-Stream als Quelle erstellt haben.“
`Kinesis.AccessDenied`	„Beim Aufrufen von Kinesis wurde der Zugriff verweigert. Stellen Sie sicher, dass die Zugriffsrichtlinie für die verwendete IAM-Rolle den Zugriff auf die entsprechende Kinesis APIs ermöglicht.“
`Kinesis.AccessDenied`	„Beim Versuch, API-Operationen auf dem zugrunde liegenden Kinesis Stream aufzurufen, wurde der Zugriff verweigert. Stellen Sie sicher, dass die IAM-Rolle weitergegeben und gültig ist.“
`Kinesis.KMS.AccessDeniedException`	„Firehose hat keinen Zugriff auf den KMS-Schlüssel, der für encrypt/decrypt den Kinesis Stream verwendet wird. Bitte gewähren Sie der Firehose-Lieferrolle Zugriff auf den Schlüssel.“

Fehlercode	Fehlermeldungen und Informationen
`Kinesis.KMS.KeyDisabled`	„Firehose kann nicht aus dem Kinesis Stream der Quelle lesen, da der dafür verwendete KMS-Schlüssel deaktiviert encrypt/decrypt ist. Aktivieren Sie den Schlüssel, damit der Lesevorgang fortgesetzt werden kann.“
`Kinesis.KMS.InvalidStateException`	„Firehose kann nicht aus dem Quell-Kinesis-Stream lesen, da der KMS-Schlüssel, der zum Verschlüsseln verwendet wurde, in einem ungültigen Zustand ist.“
`Kinesis.KMS.NotFoundException`	„Firehose kann nicht aus dem Quell-Kinesis Stream lesen, da der KMS-Schlüssel, der zum Verschlüsseln verwendet wurde, nicht gefunden wurde.

DirectPut Kinesis-Aufruffehler

Amazon Data Firehose kann die folgenden Kinesis-Aufruffehler an DirectPut Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`Firehose.KMS.AccessDeniedException`	„Firehose hat keinen Zugriff auf den KMS-Schlüssel. Bitte überprüfen Sie die Schlüsselrichtlinie.“
`Firehose.KMS.InvalidStateException`	„Firehose kann die Daten nicht entschlüsseln, weil der zur Verschlüsselung verwendete KMS-Schlüssel ungültig ist.“
`Firehose.KMS.NotFoundException`	„Firehose ist nicht in der Lage, die Daten zu entschlüsseln, da der zur Verschlüsselung verwendete KMS-Schlüssel nicht gefunden wurde.“

Fehlercode	Fehlermeldungen und Informationen
`Firehose.KMS.KeyDisabled`	„Firehose ist nicht in der Lage, die Daten zu entschlüsseln, da der zur Verschlüsselung der Daten verwendete KMS-Schlüssel deaktiviert ist. Aktivieren Sie den Schlüssel, damit die Datenübermittlung fortgesetzt werden kann.“

AWS Glue Fehler beim Aufrufen

Amazon Data Firehose kann die folgenden Aufruffehler AWS Glue an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.InvalidSchema`	„Das Schema ist ungültig.“
`DataFormatConversion.EntityNotFound`	„Die angegebene Datei table/database konnte nicht gefunden werden. Bitte stellen Sie sicher, dass das table/database existiert und dass die in der Schemakonfiguration angegebenen Werte korrekt sind, insbesondere im Hinblick auf die Groß- und Kleinschreibung.“
`DataFormatConversion.InvalidInput`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die angegebene Datenbank mit der angegeben en Katalog-ID existiert.“
`DataFormatConversion.InvalidInput`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass der übergebene ARN das richtige Format hat.“
`DataFormatConversion.InvalidInput`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die angegebene catalogId gültig ist.“

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.InvalidVersionId`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die angegebene Version der Tabelle existiert.“
`DataFormatConversion.NonExistentColumns`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die Tabelle mit einem Speicherdeskriptor konfiguriert ist, der nicht Null ist und die Zielspalten enthält.“
`DataFormatConversion.AccessDenied`	Zugriff beim Übernehmen der Rolle verweigert Bitte vergewissern Sie sich, dass die in der Konfiguration der Datenformatkonvertierung angegebene Rolle dem Firehose-Dienst die Berechtigung erteilt hat, diese zu übernehmen.“
`DataFormatConversion.ThrottledByGlue`	„Beim Aufrufen von Glue ist ein Drosselungsfehler aufgetreten. Erhöhen Sie entweder das Limit für die Anforderungsrate oder verringern Sie die aktuelle Rate, mit der Glue über andere Anwendungen aufgerufen wird.“
`DataFormatConversion.AccessDenied`	„Beim Aufrufen von Glue wurde der Zugriff verweigert. Bitte stellen Sie sicher, dass die in der Konfiguration zur Datenformatkonvertierung angegebene Rolle dem Firehose-Dienst die Erlaubnis erteilt hat, diese Rolle zu übernehmen.“
`DataFormatConversion.InvalidGlueRole`	„Ungültige Rolle. Bitte stellen Sie sicher, dass die in der Konfiguration zur Datenformatkonvertierung angegebene Rolle existiert.“
`DataFormatConversion.InvalidGlueRole`	Das Sicherheits-Token der Anfrage ist ungültig. Stellen Sie sicher, dass die bereitgestellte IAM-Rolle, die Firehose zugeordnet ist, nicht gelöscht wird.“

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.GlueNotAvailableInRegion`	„AWS Glue ist in der von Ihnen angegebenen Region noch nicht verfügbar. Bitte geben Sie eine andere Region an.“
`DataFormatConversion.GlueEncryptionException`	„Beim Abrufen des Hauptschlüssels ist ein Fehler aufgetreten. Stellen Sie sicher, dass der Schlüssel vorhanden ist und über die richtigen Zugriffsberechtigungen verfügt.“
`DataFormatConversion.SchemaValidationTimeout`	„Beim Abrufen der Tabelle von Glue ist eine Zeitüberschreitung aufgetreten. Wenn Sie eine große Anzahl von Glue-Tabellenversionen haben, fügen Sie bitte die 'glue: GetTableVersion '-Berechtigung hinzu (empfohlen) oder löschen Sie unbenutzte Tabellenversionen. Wenn Sie nicht über eine große Anzahl von Tabellen in Glue verfügen, wenden Sie sich bitte an den AWS Support.“
`DataFirehose.InternalError`	„Beim Abrufen der Tabelle von Glue ist eine Zeitüberschreitung aufgetreten. Wenn Sie eine große Anzahl von Glue-Tabellenversionen haben, fügen Sie bitte die 'glue: GetTableVersion '-Berechtigung hinzu (empfohlen) oder löschen Sie unbenutzte Tabellenversionen. Wenn Sie nicht über eine große Anzahl von Tabellen in Glue verfügen, wenden Sie sich bitte an den AWS Support.“
`DataFormatConversion.GlueEncryptionException`	„Beim Abrufen des Hauptschlüssels ist ein Fehler aufgetreten. Stellen Sie sicher, dass der Schlüssel existiert und der Status korrekt ist.“

DataFormatConversion Fehler beim Aufrufen

Amazon Data Firehose kann die folgenden Aufruffehler DataFormatConversion an Logs senden. CloudWatch

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.InvalidSchema`	„Das Schema ist ungültig.“
`DataFormatConversion.ValidationException`	„Spaltennamen und -typen dürfen keine leeren Zeichenfolgen sein.“
`DataFormatConversion.ParseError`	„Auf falsch formatiertes JSON gestoßen.“
`DataFormatConversion.MalformedData`	„Die Daten stimmen nicht mit dem Schema überein.“
`DataFormatConversion.MalformedData`	„Die Länge des JSON-Schlüssels darf nicht größer als 262 144 sein“
`DataFormatConversion.MalformedData`	„Die Daten können nicht als UTF-8 dekodiert werden.“
`DataFormatConversi`	„Ungültiges Zeichen zwischen Token gefunden.“

Fehlercode	Fehlermeldungen und Informationen
`on.Malfor medData`	
`DataForma tConversi on.Invali dTypeFormat`	„Das Typformat ist ungültig. Überprüfen Sie die Typsyntax.“
`DataForma tConversi on.Invali dSchema`	„Ungültiges Schema. Bitte stellen Sie sicher, dass die Spaltennamen keine Sonderzeichen oder Leerzeichen enthalten.“
`DataForma tConversi on.Invali dRecord`	„Der Datensatz entspricht nicht dem Schema. Ein oder mehrere Map-Schlüssel waren für map<string,string> ungültig.“
`DataForma tConversi on.Malfor medData`	„Die Eingabe-JSON enthielt ein Primitiv auf der obersten Ebene. Die oberste Ebene muss ein Objekt oder Array sein.“
`DataForma tConversi on.Malfor medData`	„Die Eingabe-JSON enthielt ein Primitiv auf der obersten Ebene. Die oberste Ebene muss ein Objekt oder Array sein.“
`DataForma tConversi on.Malfor medData`	„Der Datensatz war leer oder enthielt nur Leerzeichen.“

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.MalformedData`	„Auf ungültige Zeichen gestoßen.“
`DataFormatConversion.MalformedData`	„Es wurde ein ungültiges oder nicht unterstütztes Zeitstempelformat festgestellt. Informationen zu den unterstützten Zeitstempelformaten finden Sie im Firehose-Entwicklerhandbuch.“
`DataFormatConversion.MalformedData`	„In den Daten wurde ein skalarer Typ gefunden, aber im Schema wurde ein komplexer Typ angegeben.“
`DataFormatConversion.MalformedData`	„Die Daten stimmen nicht mit dem Schema überein.“
`DataFormatConversion.MalformedData`	„In den Daten wurde ein skalarer Typ gefunden, aber im Schema wurde ein komplexer Typ angegeben.“
`DataFormatConversion.ConversionFailureException`	"ConversionFailureException"

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.DataFormatConversionCustomerErrorException`	"DataFormatConversionCustomerErrorException"
`DataFormatConversion.DataFormatConversionCustomerErrorException`	"DataFormatConversionCustomerErrorException"
`DataFormatConversion.MalformedData`	„Die Daten stimmen nicht mit dem Schema überein.“
`DataFormatConversion.InvalidSchema`	„Das Schema ist ungültig.“
`DataFormatConversion.MalformedData`	„Die Daten stimmen nicht mit dem Schema überein. Ungültiges Format für ein oder mehrere Daten.“
`DataFormatConversion.MalformedData`	„Daten enthalten eine stark verschachtelte JSON-Struktur, die nicht unterstützt wird.“

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.EntityNotFound`	„Die angegebene table/database Datei konnte nicht gefunden werden. Bitte stellen Sie sicher, dass das table/database existiert und dass die in der Schemakonfiguration angegebenen Werte korrekt sind, insbesondere im Hinblick auf die Groß- und Kleinschreibung.“
`DataFormatConversion.InvalidInput`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die angegebene Datenbank mit der angegeben en Katalog-ID existiert.“
`DataFormatConversion.InvalidInput`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass der übergebene ARN das richtige Format hat.“
`DataFormatConversion.InvalidInput`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die angegebene catalogId gültig ist.“
`DataFormatConversion.InvalidVersionId`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die angegebene Version der Tabelle existiert.“
`DataFormatConversion.NonExistentColumns`	„Es konnte kein passendes Schema von Glue gefunden werden. Bitte stellen Sie sicher, dass die Tabelle mit einem Speicherdeskriptor konfiguriert ist, der nicht Null ist und die Zielspalten enthält.“
`DataFormatConversion.AccessDenied`	Zugriff beim Übernehmen der Rolle verweigert Bitte vergewissern Sie sich, dass die in der Konfiguration der Datenformatkonvertierung angegebene Rolle dem Firehose-Dienst die Berechtigung erteilt hat, diese zu übernehmen.“

Fehlercode	Fehlermeldungen und Informationen
`DataFormatConversion.ThrottledByGlue`	„Beim Aufrufen von Glue ist ein Drosselungsfehler aufgetreten. Erhöhen Sie entweder das Limit für die Anforderungsrate oder verringern Sie die aktuelle Rate, mit der Glue über andere Anwendungen aufgerufen wird.“
`DataFormatConversion.AccessDenied`	„Beim Aufrufen von Glue wurde der Zugriff verweigert. Bitte stellen Sie sicher, dass die in der Konfiguration zur Datenformatkonvertierung angegebene Rolle dem Firehose-Dienst die Erlaubnis erteilt hat, diese Rolle zu übernehmen.“
`DataFormatConversion.InvalidGlueRole`	„Ungültige Rolle. Bitte stellen Sie sicher, dass die in der Konfiguration zur Datenformatkonvertierung angegebene Rolle existiert.“
`DataFormatConversion.GlueNotAvailableInRegion`	„AWS Glue ist in der von Ihnen angegebenen Region noch nicht verfügbar. Bitte geben Sie eine andere Region an.“
`DataFormatConversion.GlueEncryptionException`	„Beim Abrufen des Hauptschlüssels ist ein Fehler aufgetreten. Stellen Sie sicher, dass der Schlüssel vorhanden ist und über die richtigen Zugriffsberechtigungen verfügt.“
`DataFormatConversion.SchemaValidationTimeout`	„Beim Abrufen der Tabelle von Glue ist eine Zeitüberschreitung aufgetreten. Wenn Sie eine große Anzahl von Glue-Tabellenversionen haben, fügen Sie bitte die 'glue: GetTableVersion '-Berechtigung hinzu (empfohlen) oder löschen Sie unbenutzte Tabellenversionen. Wenn Sie nicht über eine große Anzahl von Tabellen in Glue verfügen, wenden Sie sich bitte an den AWS Support.“

Fehlercode	Fehlermeldungen und Informationen
`DataFirehose.InternalError`	„Beim Abrufen der Tabelle von Glue ist eine Zeitüberschreitung aufgetreten. Wenn Sie eine große Anzahl von Glue-Tabellenversionen haben, fügen Sie bitte die 'glue: GetTableVersion '-Berechtigung hinzu (empfohlen) oder löschen Sie unbenutzte Tabellenversionen. Wenn Sie nicht über eine große Anzahl von Tabellen in Glue verfügen, wenden Sie sich bitte an den AWS Support.“
`DataFormatConversion.MalformedData`	„Ein oder mehrere Felder haben ein falsches Format.“

CloudWatch Zugriffsprotokolle für Amazon Data Firehose

Sie können die Fehlerprotokolle im Zusammenhang mit dem Ausfall der Amazon Data Firehose-Datenzustellung in der Amazon Data Firehose-Konsole oder der CloudWatch Konsole einsehen. Die folgenden Verfahren zeigen, wie Sie mithilfe dieser zwei Methoden auf die Fehlerprotokolle zugreifen können.

So greifen Sie mit der Amazon Data Firehose-Konsole auf Fehlerprotokolle zu

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Firehose-Konsole unter https://console.aws.amazon.com /firehose
2. Wählen Sie in der Navigationsleiste eine Region aus. AWS
3. Wählen Sie einen Firehose-Stream-Namen, um zur Firehose-Stream-Detailseite zu gelangen.
4. Wählen Sie Error Log, um eine Liste der Fehlerprotokolle zur Datenbereitstellung anzuzeigen.

Um über die Konsole auf Fehlerprotokolle zuzugreifen CloudWatch

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste eine Region aus.
3. Wählen Sie im Navigationsbereich Protokolle aus.

4. Wählen Sie eine Protokollgruppe und einen Protokoll-Stream, um eine Liste der Fehlerprotokolle für die Datenbereitstellung anzuzeigen.

Überwachen Sie den Zustand des Kinesis-Agenten

Kinesis Agent veröffentlicht benutzerdefinierte CloudWatch Metriken mit einem Namespace von. AWS KinesisAgent Es hilft zu beurteilen, ob der Agent fehlerfrei ist, Daten wie angegeben an Amazon Data Firehose sendet und die entsprechende Menge an CPU- und Speicherressourcen auf dem Datenproduzenten verbraucht.

Metriken wie die Anzahl der gesendeten Datensätze und Byte sind nützlich, um die Geschwindigkeit zu verstehen, mit der der Agent Daten an den Firehose-Stream sendet. Wenn diese Metriken um einige Prozent unter die erwarteten Schwellenwerte oder auf Null sinken, kann dies auf Probleme mit der Konfiguration, Netzwerkfehler oder Probleme mit dem Zustand des Agenten hinweisen. Metriken wie On-Host CPU- und Speicherbelegung sowie die Anzahl der Agentenfehler zeigen die Nutzung der Ressourcen des Datenproduzenten an und informieren über potenzielle Konfigurations- oder Hostfehler. Schließlich protokolliert der Agent auf Serviceausnahmen, um die Untersuchung von Agentenproblemen zu unterstützen.

Der Agentenmetriken werden in der Region gemeldet, die in der Agentenkonfigurationseinstellung `cloudwatch.endpoint` angegeben ist. Weitere Informationen finden Sie unter Geben Sie die Einstellungen der Agentenkonfiguration an.

Cloudwatch-Metriken, die von mehreren Kinesis-Agenten veröffentlicht wurden, werden aggregiert oder kombiniert.

Für vom Kinesis-Agenten ausgegebene (standardmäßig aktivierte) Metriken fällt eine Schutzgebühr an. Weitere Informationen finden Sie unter CloudWatch Amazon-Preise.

Überwachen Sie mit CloudWatch

Kinesis Agent sendet die folgenden Metriken an CloudWatch.

Metrik	Beschreibung
`BytesSent`	Die Anzahl der Byte, die über den angegebenen Zeitraum an den Firehose-Stream gesendet wurden.

Metrik	Beschreibung
	Einheiten: Byte
`RecordSendAttempts`	Die Anzahl der versuchten Datensätze (zum ersten Mal oder wiederholt) in einem Aufruf an `PutRecordBatch` in dem angegebenen Zeitraum. Einheiten: Anzahl
`RecordSendErrors`	Die Anzahl der Datensätze mit Fehlerstatus in einem Aufruf an `PutRecordBatch` , einschließlich wiederholter Versuche, in dem angegebenen Zeitraum. Einheiten: Anzahl
`ServiceErrors`	Die Anzahl der Aufrufe an `PutRecordBatch` , die zu einem Servicefehler führten (außer Ablehnungsfehlern) in dem angegebenen Zeitraum. Einheiten: Anzahl

Protokollieren Sie API-Aufrufe von Amazon Data Firehose mit AWS CloudTrail

Amazon Data Firehose ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in Amazon Data Firehose ausgeführt wurden. CloudTrail erfasst alle API-Aufrufe für Amazon Data Firehose als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Amazon Data Firehose-Konsole und Code-Aufrufe an die Amazon Data Firehose-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Amazon Data Firehose. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf einsehen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Amazon Data Firehose gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen darüber CloudTrail, einschließlich der Konfiguration und Aktivierung, finden Sie im AWS CloudTrail Benutzerhandbuch.

Firehose-Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn unterstützte Ereignisaktivitäten in Amazon Data Firehose auftreten, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS -Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail -API-Ereignisverlauf.

Für eine fortlaufende Aufzeichnung von Ereignissen in Ihrem AWS Konto, einschließlich Ereignissen für Amazon Data Firehose, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole erstellen, gilt der Trail standardmäßig für alle AWS Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Amazon Data Firehose unterstützt die Protokollierung der folgenden Aktionen als Ereignisse in CloudTrail Protokolldateien:

- CreateDeliveryStream
- DeleteDeliveryStream
- DescribeDeliveryStream
- ListDeliveryStreams
- ListTagsForDeliveryStream
- TagDeliveryStream
- StartDeliveryStreamEncryption
- StopDeliveryStreamEncryption

- UntagDeliveryStream
- UpdateDestination

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Beispiel: Firehose-Logdateieinträge

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis ist eine einzelne Anforderung aus einer beliebigen Quelle und enthält Informationen zur angeforderten Aktion, zu Datum und Uhrzeit der Aktion, zu den Anforderungsparametern usw. CloudTrail -Protokolldateien stellen kein geordnetes Stack-Trace der öffentlichen API-Aufrufe dar. Daher werden sie nicht in einer bestimmten Reihenfolge angezeigt.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `DeleteDeliveryStream` Aktionen`CreateDeliveryStream`,`DescribeDeliveryStream`, `ListDeliveryStreams`, `UpdateDestination`, und demonstriert.

```
{
  "Records":[
        {
            "eventVersion":"1.02",
            "userIdentity":{
                "type":"IAMUser",
                "principalId":"AKIAIOSFODNN7EXAMPLE",
                "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
                "accountId":"111122223333",
                "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
                "userName":"CloudTrail_Test_User"
            },
```

```
            "eventTime":"2016-02-24T18:08:22Z",
            "eventSource":"firehose.amazonaws.com",
            "eventName":"CreateDeliveryStream",
            "awsRegion":"us-east-1",
            "sourceIPAddress":"127.0.0.1",
            "userAgent":"aws-internal/3",
            "requestParameters":{
                "deliveryStreamName":"TestRedshiftStream",
                "redshiftDestinationConfiguration":{
                "s3Configuration":{
                    "compressionFormat":"GZIP",
                    "prefix":"prefix",
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    }
                },
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "password":"",
                "username":"",
                "roleARN":"arn:aws:iam::111122223333:role/Firehose"
            }
        },
        "responseElements":{
            "deliveryStreamARN":"arn:aws:firehose:us-
east-1:111122223333:deliverystream/TestRedshiftStream"
        },
        "requestID":"958abf6a-db21-11e5-bb88-91ae9617edf5",
        "eventID":"875d2d68-476c-4ad5-bbc6-d02872cfc884",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
```

```
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:08:54Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DescribeDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"aa6ea5ed-db21-11e5-bb88-91ae9617edf5",
        "eventID":"d9b285d8-d690-4d5c-b9fe-d1ad5ab03f14",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:00Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"ListDeliveryStreams",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "limit":10
        },
```

```
        "responseElements":null,
        "requestID":"d1bf7f86-db21-11e5-bb88-91ae9617edf5",
        "eventID":"67f63c74-4335-48c0-9004-4ba35ce00128",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:09Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"UpdateDestination",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "destinationId":"destinationId-000000000001",
            "deliveryStreamName":"TestRedshiftStream",
            "currentDeliveryStreamVersionId":"1",
            "redshiftDestinationUpdate":{
                "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "password":"",
                "username":"",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "s3Update":{
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket-update",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "compressionFormat":"GZIP",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
```

```
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    },
                    "prefix":"arn:aws:s3:::amzn-s3-demo-bucket"
                }
            }
        },
        "responseElements":null,
        "requestID":"d549428d-db21-11e5-bb88-91ae9617edf5",
        "eventID":"1cb21e0b-416a-415d-bbf9-769b152a6585",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:12Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DeleteDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"d85968c1-db21-11e5-bb88-91ae9617edf5",
        "eventID":"dd46bb98-b4e9-42ff-a6af-32d57e636ad1",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    }
  ]
}
```

Codebeispiele für Firehose mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Firehose mit einem AWS Software Development Kit (SDK) verwendet wird.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterFirehose mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Firehose AWS SDKs
  - Aktionen für die Verwendung von Firehose AWS SDKs
    - Verwendung PutRecord mit einem AWS SDK oder CLI
    - Verwendung PutRecordBatch mit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Firehose AWS SDKs
  - Verwenden von Amazon Data Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze

Grundlegende Beispiele für die Verwendung von Firehose AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon Data Firehose mit AWS SDKs verwenden können.

Beispiele

- Aktionen für die Verwendung von Firehose AWS SDKs
  - Verwendung PutRecord mit einem AWS SDK oder CLI
  - Verwendung PutRecordBatch mit einem AWS SDK oder CLI

Aktionen für die Verwendung von Firehose AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Firehose-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Firehouse-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Firehose AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Data Firehose.

Beispiele

- Verwendung PutRecord mit einem AWS SDK oder CLI
- Verwendung PutRecordBatch mit einem AWS SDK oder CLI

Verwendung **`PutRecord`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRecord` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Datensätze in Firehose einfügen

CLI

AWS CLI

So schreiben Sie einen Datensatz in einen Stream

Im folgenden Beispiel für `put-record` werden Daten in einen Stream geschrieben. Die Daten sind im Base64-Format codiert.

```
aws firehose put-record \
    --delivery-stream-name my-stream \
    --record '{"Data":"SGVsbG8gd29ybGQ="}'
```

Ausgabe:

```
{
    "RecordId": "RjB5K/nnoGFHqwTsZlNd/
TTqvjE8V5dsyXZTQn2JXrdpMTOwssyEb6nfC8fwf1whhwnItt4mvrn+gsqeK5jB7QjuLg283+Ps4Sz/
j1Xujv31iDhnPdaLw4BOyM9Amv7PcCuB2079RuM0NhoakbyUymlwY8yt20G8X2420wu1jlFafhci4erAt7QhDEvpw
    "Encrypted": false
}
```

Weitere Informationen finden Sie unter Senden von Daten an einen Bereitstellungsstream von Amazon Kinesis Data Firehose im Entwicklerhandbuch zu Amazon Kinesis Data Firehose.

- Einzelheiten zur API finden Sie PutRecordin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
```

```
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

- Einzelheiten zur API finden Sie PutRecordin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
```

```
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- Einzelheiten zur API finden Sie PutRecordin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Firehose mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`PutRecordBatch`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRecordBatch` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Datensätze in Firehose einfügen

CLI

AWS CLI

So schreiben Sie mehrere Datensätze in einen Stream

Im folgenden Beispiel für `put-record-batch` werden drei Datensätze in einen Stream geschrieben. Die Daten sind im Base64-Format codiert.

```
aws firehose put-record-batch \
    --delivery-stream-name my-stream \
    --records file://records.json
```

Inhalt von `myfile.json`:

```
[
    {"Data": "Rmlyc3QgdGhpbmc="},
    {"Data": "U2Vjb25kIHRoaW5n"},
    {"Data": "VGhpcmQgdGhpbmc="}
]
```

Ausgabe:

```
{
```

```
    "FailedPutCount": 0,
    "Encrypted": false,
    "RequestResponses": [
        {
            "RecordId": "9D2OJ6t2EqCTZTXwGzeSv/EVHxRoRCw89xd+o3+sXg8DhYOaWKPSmZy/
CGlRVEys1u1xbeKh6VofEYKkoeiDrcjrxhQp9iF7sUW7pujiMEQ5LzlrzCkGosxQn
+3boDnURDEaD42V7GiixpOyLJkYZcae1i7HzlCEoy9LJhMr8EjDSi4Om/9Vc2uhwwuAtGE0XKpxJ2WD7ZRWtAnYlK
        },
        {
            "RecordId": "jFirejqxCLlK5xjH/UNmlMVcjktEN76I7916X9PaZ
+PVaOSXDfU1WGOqEZhxq2js7xcZ552eoeDxsuTU1MSq9nZTbVfb6cQTIXnm/
GsuF37Uhg67GKmR5z90l6XKJ+/
+pDloFv7Hh9a3oUS6wYm3DcNRLTHHAimANp1PhkQvWpvLRfzbuCUkBphR2QVzhP9OiHLbzGwy8/
DfH8sqWEUYASNJKS8GXP5s"
        },
        {
            "RecordId":
 "oy0amQ40o5Y2YV4vxzufdcMOOw6n3EPr3tpPJGoYVNKH4APPVqNcbUgefo1stEFRg4hTLrf2k6eliHu/9+YJ5R3
DTBt3qBlmTj7Xq8SKVbO1S7YvMTpWkMKA86f8JfmT8BMKoMb4XZS/sOkQLe+qh0sYKXWl"
        }
    ]
}
```

Weitere Informationen finden Sie unter Senden von Daten an einen Bereitstellungsstream von Amazon Kinesis Data Firehose im Entwicklerhandbuch zu Amazon Kinesis Data Firehose.

- Einzelheiten zur API finden Sie PutRecordBatchin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
 stream.
     *
```

```
    * @param records           a list of maps representing the records to be
sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord =
objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose
record", e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();
```

```
                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }
```

- Einzelheiten zur API finden Sie PutRecordBatchin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
```

```
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
```

```
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")
```

- Einzelheiten zur API finden Sie PutRecordBatchin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn put_record_batch(
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- Einzelheiten zur API finden Sie PutRecordBatchin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterFirehose mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für die Verwendung von Firehose AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Firehose mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Firehouse oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Verwenden von Amazon Data Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze

Verwenden von Amazon Data Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze

Die folgenden Codebeispiele zeigen, wie Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze verwendet werden kann.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel werden Einzel- und Batch-Datensätze an Firehose übermittelt.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
```

```
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
 objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
 e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
```

```
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
```

```
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }


    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param records           a list of maps representing the records to be
sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
```

```
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord =
objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                                .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose
record", e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
```

```
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime,
Instant endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

.dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).bui
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
FirehoseScenario.class.getResourceAsStream("/" + fileName);
```

```
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8))
 {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - PutRecord
  - PutRecordBatch

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Skript werden Einzel- und Batch-Datensätze an Firehose übermittelt.

```
import json
import logging
import random
from datetime import datetime, timedelta
```

```
import backoff
import boto3

from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.

    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
```

```
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
```

```
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")


    def get_metric_statistics(
        self,
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
        statistics: list = ["Sum"],
    ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data
points.
            statistics (list): A list of statistics to retrieve. Default is
['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
```

```
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value":
self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords',
and 'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())

        metrics = {
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }

        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
```

```
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5
minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.

        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
```

```
        """
        Log the batch response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record_batch API
 call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
 {batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")


if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

Diese Datei enthält die Konfiguration für das oben genannte Skript.

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
```

```
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
        )


def get_config():
    return Config()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - PutRecord
  - PutRecordBatch

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterFirehose mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Fehler in Amazon Data Firehose beheben

Wenn Firehose bei der Bereitstellung oder Verarbeitung von Daten auf Fehler stößt, versucht es erneut, bis die konfigurierte Wiederholungsdauer abgelaufen ist. Wenn die Wiederholungsdauer endet, bevor die Daten erfolgreich übermittelt wurden, sichert Firehose die Daten im konfigurierten S3-Backup-Bucket. Wenn das Ziel Amazon S3 ist und die Lieferung fehlschlägt oder wenn die Lieferung an den Backup-S3-Bucket fehlschlägt, versucht Firehose es so lange erneut, bis die Aufbewahrungsfrist abgelaufen ist.

Informationen zur Nachverfolgung von Lieferfehlern mithilfe von finden Sie CloudWatch unter. the section called “Mit CloudWatch Protokollen überwachen”

Direct PUT

Bei `DirectPut` Firehose-Streams bewahrt Firehose die Aufzeichnungen 24 Stunden lang auf. Für einen Firehose-Stream, dessen Datenquelle ein Kinesis-Datenstream ist, können Sie den Aufbewahrungszeitraum ändern, wie unter Ändern des Datenaufbewahrungszeitraums beschrieben. In diesem Fall wiederholt Firehose die folgenden Operationen auf unbestimmte Zeit:`DescribeStream`, und`GetRecords`. `GetShardIterator`

Wenn der Firehose-Stream verwendet`DirectPut`, überprüfen Sie die `IncomingRecords` Metriken `IncomingBytes` und, um festzustellen, ob eingehender Datenverkehr vorhanden ist. Wenn Sie `PutRecord` oder `PutRecordBatch` verwenden, müssen Sie Ausnahmen abfangen und Wiederholungsversuche veranlassen. Wir empfehlen eine Wiederholungsrichtlinie mit exponentiellem Backoff mit Jitter und mehreren Wiederholungsversuchen. Wenn Sie die `PutRecordBatch` API verwenden, stellen Sie außerdem sicher, dass Ihr Code den Wert von FailedPutCountin der Antwort überprüft, auch wenn der API-Aufruf erfolgreich ist.

Kinesis Data Stream

Wenn der Firehose-Stream einen Kinesis-Datenstream als Quelle verwendet, überprüfen Sie die `IncomingBytes` `IncomingRecords` UND-Metriken für den Quelldatenstream. Stellen Sie außerdem sicher, dass die `DataReadFromKinesisStream.Records` Metriken `DataReadFromKinesisStream.Bytes` und für den Firehose-Stream ausgegeben werden.

Häufige Probleme

Im Folgenden finden Sie Tipps zur Fehlerbehebung, die Ihnen helfen sollen, häufig auftretende Probleme bei der Arbeit mit einem Firehose-Stream zu lösen.

Firehose-Stream nicht verfügbar

Der Firehose-Stream ist nicht als Ziel für CloudWatch Protokolle, CloudWatch Ereignisse oder AWS IoT-Aktionen verfügbar, da einige AWS Dienste nur Nachrichten und Ereignisse an einen Firehose-Stream senden können, der sich im selben befindet. AWS-Region Stellen Sie sicher, dass sich Ihr Firehose-Stream in derselben Region wie Ihre anderen Dienste befindet.

Keine Daten am Ziel

Wenn es keine Probleme bei der Datenaufnahme gibt und die für den Firehose-Stream ausgegebenen Metriken gut aussehen, Sie die Daten am Ziel jedoch nicht sehen, überprüfen Sie die Leselogik. Stellen Sie sicher, dass die Lesekomponente alle Daten richtig analysiert.

Die Metrik zur Datenaktualität nimmt zu oder wird nicht ausgegeben

Die Datenaktualität ist ein Maß dafür, wie aktuell Ihre Daten in Ihrem Firehose-Stream sind. Es ist das Alter des ältesten Datensatzes im Firehose-Stream, gemessen von der Zeit, als Firehose die Daten aufgenommen hat, bis heute. Firehose bietet Metriken, mit denen Sie die Datenaktualität überwachen können. Für Informationen zum Identifizieren der Datenaktualitätsmetrik für ein bestimmtes Ziel siehe the section called “Überwachung mit Metriken CloudWatch ”.

Wenn Sie die Sicherung für alle Ereignisse oder alle Dokumente aktivieren, sollten Sie zwei verschiedene Datenaktualitätsmetriken überwachen: eine für das Hauptziel und eine für die Sicherung.

Wenn die Datenaktualitätsmetrik nicht ausgegeben wird, bedeutet dies, dass für den Firehose-Stream keine aktive Bereitstellung erfolgt. Dies geschieht, wenn die Datenbereitstellung vollständig blockiert wurde oder keine Daten eingehen.

Wenn die Datenaktualitätsmetrik kontinuierlich ansteigt, bedeutet dies, dass die Daten immer stärker veralten. Dies kann aus einem der folgenden Gründe geschehen.

- Das Ziel kann mit der Bereitstellungsrate nicht Schritt halten. Wenn Firehose aufgrund des hohen Datenverkehrs auf vorübergehende Fehler stößt, kann es sein, dass die Lieferung in Verzug gerät. Dies kann für andere Ziele als Amazon S3 passieren (es kann für OpenSearch Service, Amazon

Redshift oder Splunk passieren). Stellen Sie sicher, dass das Ziel über genügend Kapazität verfügt, um den eingehenden Datenverkehr zu verarbeiten.

- Das Ziel ist langsam. Die Datenzustellung könnte ins Hintertreffen geraten, wenn Firehose auf eine hohe Latenz stößt. Überwachen Sie die Latenzmetrik des Ziels.
- Die Lambda-Funktion ist langsam. Dies kann zu einer Datenübermittlungsrate führen, die unter der Datenaufnahmerate für den Firehose-Stream liegt. Verbessern Sie die Effizienz der Lambda-Funktion (wenn möglich). Wenn die Funktion beispielsweise Netzwerk-I/O-Operationen ausführt, verwenden Sie mehrere Threads oder asynchrone I/O-Operationen, um die parallele Ausführung zu verbessern. Erwägen Sie außerdem, die Größe des Speichers für die Lambda-Funktion zu erhöhen, damit die CPU-Zuweisung entsprechend erhöht werden kann. Dies kann zu schnelleren Lambda-Aufrufen führen. Informationen zur Konfiguration von Lambda-Funktionen finden Sie unter Konfiguration von AWS Lambda-Funktionen.
- Es treten Fehler bei der Datenbereitstellung auf. Informationen zur Fehlerüberwachung mithilfe von Amazon CloudWatch Logs finden Sie unterthe section called “Mit CloudWatch Protokollen überwachen”.
- Wenn die Datenquelle des Firehose-Streams ein Kinesis-Datenstream ist, kann es zu einer Drosselung kommen. Prüfen Sie die Metriken `ThrottledGetRecords`, `ThrottledGetShardIterator` und `ThrottledDescribeStream`. Wenn an den Kinesis-Daten-Stream mehrere Konsumenten angefügt sind, müssen Sie Folgendes beachten:
  - Wenn die Metriken `ThrottledGetRecords` und `ThrottledGetShardIterator` hohe Werte aufweisen, sollten Sie die Anzahl der für den Daten-Stream bereitgestellten Shards erhöhen.
  - Wenn der Wert hoch `ThrottledDescribeStream` ist, empfehlen wir Ihnen, die `kinesis:listshards` Berechtigung zu der in konfigurierten Rolle hinzuzufügen. KinesisStreamSourceConfiguration
- Hinweise auf erschöpfte BufferingHints für das Ziel. Dies könnte die Anzahl der Hin- und Rückfahrten erhöhen, die Firehose zum Zielort unternehmen muss, was dazu führen könnte, dass die Lieferung ins Hintertreffen gerät. Erwägen Sie, den Wert für BufferingHints zu erhöhen. Weitere Informationen finden Sie unter BufferingHints.
- Eine hohe Anzahl Wiederholungsversuche kann dazu führen, dass die Bereitstellung zurückfällt, wenn die Fehler häufig auftreten. Erwägen Sie, den Wiederholungszeitraum zu verkürzen. Überwachen Sie außerdem die Fehler und versuchen Sie, deren Anzahl zu reduzieren. Informationen zur Fehlerüberwachung mithilfe von Amazon CloudWatch Logs finden Sie unterthe section called “Mit CloudWatch Protokollen überwachen”.
- Wenn das Ziel Splunk und `DeliveryToSplunk.DataFreshness` hoch ist, `DeliveryToSplunk.Success` jedoch gute Werte zeigt, ist der Splunk-Cluster möglicherweise

ausgelastet. Befreien Sie den Splunk-Cluster von Belastungen, wenn möglich. Wenden Sie sich alternativ an den AWS Support und fordern Sie eine Erhöhung der Anzahl der Kanäle an, die Firehose für die Kommunikation mit dem Splunk-Cluster verwendet.

Die Konvertierung des Datensatzformats in Apache Parquet schlägt fehl

Dies passiert, wenn Sie DynamoDB-Daten, die den `Set` Typ enthalten, über Lambda in einen Firehose-Stream streamen und das Datensatzformat mithilfe von in Apache Parquet AWS Glue Data Catalog konvertieren.

Wenn der AWS Glue Crawler die in DynamoDB festgelegten Datentypen (`StringSetNumberSet`, und`BinarySet`) indexiert, speichert er sie im Datenkatalog als `SET<STRING>SET<BIGINT>`, bzw.. `SET<BINARY>` Damit Firehose die Datensätze in das Apache Parquet-Format konvertieren kann, sind jedoch Apache Hive-Datentypen erforderlich. Da es sich bei den festgelegten Typen um keine gültigen Apache Hive-Datentypen handelt, schlägt die Konvertierung fehl. Damit die Konvertierung funktioniert, aktualisieren Sie den Datenkatalog mit Apache Hive-Datentypen. Sie können dies tun, indem Sie im Datenkatalog `set` in `array` ändern.

Um einen oder mehrere Datentypen **`array`** in einem AWS Glue Datenkatalog von **`set`** zu zu ändern

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die AWS Glue Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie im linken Bereich unter der Überschrift Data catalog (Datenkatalog) die Option Tables (Tabellen).
3. Wählen Sie in der Liste der Tabellen den Namen der Tabelle aus, in der Sie mindestens einen Datentyp ändern müssen. Dadurch gelangen Sie zur Seite mit den Details für die Tabelle.
4. Wählen Sie in der oberen rechten Ecke der Detailseite die Schaltfläche Schema bearbeiten
5. Wählen Sie in der Spalte Data type (Datentyp) den ersten `set`-Datentyp aus.
6. Ändern Sie in der Dropdownliste Column type (Spaltentyp) den Typ von `set` in `array`.
7. Geben Sie in das ArraySchemaFeld `array<string>array<int>`, oder ein`array<binary>`, je nachdem, welcher Datentyp für Ihr Szenario geeignet ist.
8. Wählen Sie Aktualisieren aus.
9. Wiederholen Sie die vorherigen Schritte, um andere `set`-Typen in `array`-Typen zu konvertieren.
10. Wählen Sie Speichern.

Fehlende Felder für transformiertes Objekt für Lambda

Wenn Sie die Lambda-Datentransformation verwenden, um JSON-Daten in ein Parquet-Objekt zu ändern, fehlen nach der Transformation möglicherweise einige Felder. Dies passiert, wenn Ihr JSON-Objekt Großbuchstaben enthält und die Berücksichtigung von Groß- und Kleinschreibung auf `false` eingestellt ist. Dies kann zu einer Nichtübereinstimmung der JSON-Schlüssel nach der Datentransformation führen, wodurch Daten im resultierenden Parquet-Objekt im s3-Bucket fehlen.

Um dieses Problem zu beheben, stellen Sie sicher, dass die Schlauchkonfiguration auf `deserializationOption: case.insensitive` eingestellt ist, `true` sodass die JSON-Schlüssel nach der Transformation übereinstimmen.

Fehlerbehebung für Amazon S3

Überprüfen Sie Folgendes, wenn Daten nicht an Ihren Amazon Simple Storage Service (Amazon S3)-Bucket geliefert werden.

- Überprüfen Sie Firehose `IncomingBytes` und die `IncomingRecords` Metriken, um sicherzustellen, dass Daten erfolgreich an Ihren Firehose-Stream gesendet wurden. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch .
- Wenn die Datentransformation mit Lambda aktiviert ist, überprüfen Sie die `ExecuteProcessingSuccess` Firehose-Metrik, um sicherzustellen, dass Firehose versucht hat, Ihre Lambda-Funktion aufzurufen. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch .
- Überprüfen Sie die `DeliveryToS3.Success` Firehose-Metrik, um sicherzustellen, dass Firehose versucht hat, Daten in Ihren Amazon S3 S3-Bucket zu laden. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch .
- Aktivieren Sie die Fehlerprotokollierung, falls noch nicht geschehen, und überprüfen Sie die Fehlerprotokolle auf Bereitstellungsfehler. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch .
- Wenn Sie im Protokoll eine Fehlermeldung sehen, die besagt, dass Firehose InternalServerError beim Aufrufen des Amazon S3 S3-Dienstes aufgetreten ist. Der Vorgang wird erneut versucht. Wenn der Fehler weiterhin besteht, wenden Sie sich bitte an S3, um eine Lösung zu finden.“ , dies könnte auf den deutlichen Anstieg der Anforderungsraten auf einer einzelnen Partition in S3 zurückzuführen sein. Sie können die Entwurfsmuster für S3-Präfixe optimieren, um das Problem zu beheben. Weitere Informationen finden Sie unter Bewährte Entwurfsmuster: Optimierung der

Amazon S3 S3-Leistung. Wenn das Problem dadurch nicht behoben wird, wenden Sie sich an den AWS Support, um weitere Unterstützung zu erhalten.

- Stellen Sie sicher, dass der Amazon S3 S3-Bucket, der in Ihrem Firehose-Stream angegeben ist, noch existiert.
- Wenn die Datentransformation mit Lambda aktiviert ist, stellen Sie sicher, dass die Lambda-Funktion, die in Ihrem Firehose-Stream angegeben ist, noch vorhanden ist.
- Stellen Sie sicher, dass die in Ihrem Firehose-Stream angegebene IAM-Rolle Zugriff auf Ihren S3-Bucket und Ihre Lambda-Funktion hat (sofern die Datentransformation aktiviert ist). Stellen Sie außerdem sicher, dass die IAM-Rolle Zugriff auf die CloudWatch Protokollgruppe und die Protokollstreams hat, um Fehlerprotokolle zu überprüfen. Weitere Informationen finden Sie unter Firehose Zugriff auf ein Amazon S3 S3-Ziel gewähren.
- Wenn Sie die Datentransformation verwenden, stellen Sie sicher, dass Ihre Lambda-Funktion keine Antworten zurückgibt, deren Nutzlast 6 MB überschreitet. Weitere Informationen finden Sie unter Amazon Data FirehoseData Transformation.

Fehlerbehebung für Amazon Redshift

Überprüfen Sie Folgendes, wenn Daten an Ihren bereitgestellten Amazon-Redshift-Cluster oder Ihre Arbeitsgruppe von Amazon Redshift Serverless nicht übermittelt werden.

Daten werden vor dem Laden in Amazon Redshift an Ihren S3-Bucket übermittelt. Wenn die Daten nicht an Ihren S3-Bucket gesendet wurden, vgl. Fehlerbehebung für Amazon S3.

- Überprüfen Sie die `DeliveryToRedshift.Success` Firehose-Metrik, um sicherzustellen, dass Firehose versucht hat, Daten aus Ihrem S3-Bucket in den von Amazon Redshift bereitgestellten Cluster oder die Amazon Redshift Serverless-Arbeitsgruppe zu kopieren. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch .
- Aktivieren Sie die Fehlerprotokollierung, falls noch nicht geschehen, und überprüfen Sie die Fehlerprotokolle auf Bereitstellungsfehler. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch .
- Sehen Sie in der Amazon Redshift `STL_CONNECTION_LOG` Redshift-Tabelle nach, ob Firehose erfolgreiche Verbindungen herstellen kann. In dieser Tabelle sehen Sie die Verbindungen und ihre Status auf der Grundlage eines Benutzernamens. Weitere Informationen finden Sie unter `STL_CONNECTION_LOG` im Leitfaden für Datenbankentwickler für Amazon Redshift.

- Wenn die vorige Prüfung zeigt, dass Verbindungen eingerichtet werden, prüfen Sie die STL_LOAD_ERRORS-Tabelle von Amazon Redshift, um die Ursache für den Fehler beim COPY-Befehl festzustellen. Weitere Informationen finden Sie unter STL_LOAD_ERRORS im Leitfaden für Datenbankentwickler für Amazon Redshift.
- Stellen Sie sicher, dass die Amazon Redshift Redshift-Konfiguration in Ihrem Firehose-Stream korrekt und gültig ist.
- Stellen Sie sicher, dass die in Ihrem Firehose-Stream angegebene IAM-Rolle auf den S3-Bucket zugreifen kann, aus dem Amazon Redshift Daten kopiert, sowie auf die Lambda-Funktion für die Datentransformation (sofern die Datentransformation aktiviert ist). Stellen Sie außerdem sicher, dass die IAM-Rolle Zugriff auf die CloudWatch Protokollgruppe und die Protokollstreams hat, um Fehlerprotokolle zu überprüfen. Weitere Informationen finden Sie unter Firehose Zugriff auf ein Amazon Redshift Redshift-Ziel gewähren .
- Wenn sich Ihr von Amazon Redshift bereitgestellter Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe in einer Virtual Private Cloud (VPC) befindet, stellen Sie sicher, dass der Cluster den Zugriff über Firehose-IP-Adressen ermöglicht. Weitere Informationen finden Sie unter Firehose Zugriff auf ein Amazon Redshift Redshift-Ziel gewähren .
- Stellen Sie sicher, dass der von Amazon Redshift bereitgestellte Cluster oder die Arbeitsgruppe von Amazon Redshift Serverless öffentlich verfügbar ist.
- Wenn Sie die Datentransformation verwenden, stellen Sie sicher, dass Ihre Lambda-Funktion keine Antworten zurückgibt, deren Nutzlast 6 MB überschreitet. Weitere Informationen finden Sie unter Amazon Data FirehoseData Transformation.

Problembehebung bei Amazon OpenSearch Service

Überprüfen Sie Folgendes, wenn Daten nicht an Ihre OpenSearch Service-Domain geliefert werden.

Daten können gleichzeitig in Ihrem Amazon-S3-Bucket gesichert werden. Wenn die Daten nicht an Ihren S3-Bucket gesendet wurden, vgl. Fehlerbehebung für Amazon S3.

- Überprüfen Sie Firehose IncomingBytes und die IncomingRecords Metriken, um sicherzustellen, dass Daten erfolgreich an Ihren Firehose-Stream gesendet wurden. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch .
- Wenn die Datentransformation mit Lambda aktiviert ist, überprüfen Sie die ExecuteProcessingSuccess Firehose-Metrik, um sicherzustellen, dass Firehose versucht hat, Ihre Lambda-Funktion aufzurufen. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch .

- Überprüfen Sie die `DeliveryToAmazonOpenSearchService.Success` Firehose-Metrik, um sicherzustellen, dass Firehose versucht hat, Daten für den OpenSearch Service-Cluster zu indizieren. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mit Metriken CloudWatch .
- Aktivieren Sie die Fehlerprotokollierung, falls noch nicht geschehen, und überprüfen Sie die Fehlerprotokolle auf Bereitstellungsfehler. Weitere Informationen finden Sie unter Überwachen Sie Amazon Data Firehose mithilfe von Protokollen CloudWatch .
- Stellen Sie sicher, dass die OpenSearch Dienstkonfiguration in Ihrem Firehose-Stream korrekt und gültig ist.
- Wenn die Datentransformation mit Lambda aktiviert ist, stellen Sie sicher, dass die Lambda-Funktion, die in Ihrem Firehose-Stream angegeben ist, noch vorhanden ist. Stellen Sie außerdem sicher, dass die IAM-Rolle Zugriff auf die CloudWatch Protokollgruppe und die Protokollstreams hat, um Fehlerprotokolle zu überprüfen. Weitere Informationen finden Sie unter Grant FirehoseAccess to a Public OpenSearch Service Destination.
- Stellen Sie sicher, dass die in Ihrem Firehose-Stream angegebene IAM-Rolle auf Ihren OpenSearch Service-Cluster, Ihren S3-Backup-Bucket und die Lambda-Funktion zugreifen kann (sofern die Datentransformation aktiviert ist). Stellen Sie außerdem sicher, dass die IAM-Rolle Zugriff auf die CloudWatch Protokollgruppe und die Protokollstreams hat, um Fehlerprotokolle zu überprüfen. Weitere Informationen finden Sie unter Firehose Zugang zu einem Ziel des öffentlichen OpenSearch Dienstes gewähren.
- Wenn Sie die Datentransformation verwenden, stellen Sie sicher, dass Ihre Lambda-Funktion keine Antworten zurückgibt, deren Nutzlast 6 MB überschreitet. Weitere Informationen finden Sie unter Amazon Data FirehoseData Transformation.
- Amazon Data Firehose unterstützt derzeit nicht die Übermittlung von CloudWatch Protokollen an das Amazon OpenSearch Service-Ziel, da Amazon mehrere Protokollereignisse zu einem Firehose-Datensatz CloudWatch zusammenfasst und Amazon OpenSearch Service nicht mehrere Protokollereignisse in einem Datensatz akzeptieren kann. Als Alternative können Sie erwägen, den Abonnementfilter für Amazon OpenSearch Service in CloudWatch Logs zu verwenden.

Fehlerbehebung bei Splunk

Prüfen Sie die folgenden Punkte, wenn die Daten nicht an Ihren Splunk endpoint übergeben wurden.

- Wenn sich Ihre Splunk-Plattform in einer VPC befindet, stellen Sie sicher, dass Firehose darauf zugreifen kann. Weitere Informationen finden Sie unter Zugreifen auf Splunk in VPC.

- Wenn Sie einen Load AWS Balancer verwenden, stellen Sie sicher, dass es sich um einen Classic Load Balancer oder einen Application Load Balancer handelt. Aktivieren Sie außerdem dauerbasierte Sticky-Sitzungen mit deaktiviertem Cookie-Ablauf für Classic Load Balancer und mit maximaler Ablaufzeit (7 Tage) für Application Load Balancer. Informationen dazu finden Sie unter Duration-Based Session Stickiness für Classic Load Balancer oder einen Application Load Balancer.
- Überprüfen Sie die Splunk-Plattformanforderungen. Das Splunk-Add-on für Firehose erfordert Splunk-Plattformversion 6.6.X oder höher. Weitere Informationen finden Sie unter Splunk-Add-On für Amazon Kinesis Firehose.
- Wenn Sie einen Proxy (Elastic Load Balancing oder ein anderer) zwischen Firehose und dem HTTP Event Collector (HEC) -Knoten haben, aktivieren Sie Sticky Sessions, um HEC-Bestätigungen () zu unterstützen. ACKs
- Stellen Sie sicher, dass Sie ein gültiges HEC-Token verwenden.
- Stellen Sie sicher, dass der HEC-Token aktiviert ist.
- Überprüfen Sie, ob die Daten, die Sie an Splunk senden, ordnungsgemäß formatiert sind. Weitere Informationen finden Sie unter Formatieren von Ereignissen für HTTP-Ereigniserfassung.
- Stellen Sie sicher, dass der HEC-Token und das Eingabeereignis mit einem gültigen Index konfiguriert sind.
- Wenn ein Upload an Splunk aufgrund eines Server-Fehlers im HEC-Knoten fehlschlägt, wird die Anforderung automatisch wiederholt. Wenn alle Wiederholungen fehlschlagen, werden die Daten in Amazon S3 gesichert. Überprüfen Sie, ob Ihre Daten in Amazon S3 angezeigt werden, was auf eine solche Fehlfunktion hinweist.
- Stellen Sie sicher, dass die Indexbestätigung auf Ihrem HEC-Token aktiviert ist.
- Erhöhen Sie den Wert von `HECAcknowledgmentTimeoutInSeconds` in der Splunk-Zielkonfiguration Ihres Firehose-Streams.
- Erhöhen Sie den Wert von `DurationInSeconds` under `RetryOptions` in der Splunk-Zielkonfiguration Ihres Firehose-Streams.
- Überprüfen Sie Ihre HEC-Lizenzen.
- Wenn Sie die Datentransformation verwenden, stellen Sie sicher, dass Ihre Lambda-Funktion keine Antworten zurückgibt, deren Nutzlast 6 MB überschreitet. Weitere Informationen finden Sie unter Amazon Data Firehose Data Transformation.
- Stellen Sie sicher, dass der Splunk-Parameter namens `ackIdleCleanup` auf `true` festgelegt ist. Standardmäßig lautet er "false". Um diesen Parameter auf `true` festzulegen, gehen Sie wie folgt vor:

  - Senden Sie eine Anfrage für eine verwaltete Splunk Cloud-Bereitstellung über das Splunk-Support-Portal. Bitten Sie den Splunk-Support in diesem Fall, die HTTP-Ereigniserfassung zu aktivieren, `ackIdleCleanup` in `inputs.conf` auf `true` festzulegen und einen Load Balancer, der mit diesem Add-On verwendet wird, zu erstellen oder zu ändern.
  - Für eine verteilte Splunk-Enterprise-Bereitstellung legen Sie für den Parameter `ackIdleCleanup` in der Datei `inputs.conf` den Wert „true“ fest. Für *nix-Benutzer befindet sich die Datei unter `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/`. Für Windows-Benutzer befindet sie sich unter `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\`.
  - Für eine Single-Instance-Splunk-Enterprise-Bereitstellung legen Sie für den Parameter `ackIdleCleanup` in der Datei `inputs.conf` den Wert „`true`“ fest. Für *nix-Benutzer befindet sich die Datei unter `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/`. Für Windows-Benutzer befindet sie sich unter `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\`.
- Stellen Sie sicher, dass die in Ihrem Firehose-Stream angegebene IAM-Rolle auf den S3-Backup-Bucket und die Lambda-Funktion für die Datentransformation zugreifen kann (sofern die Datentransformation aktiviert ist). Stellen Sie außerdem sicher, dass die IAM-Rolle Zugriff auf die Protokollgruppe und die Protokollstreams hat, um CloudWatch Fehlerprotokolle zu überprüfen. Weitere Informationen finden Sie unter Grant FirehoseAccess to a Splunk Destination.
- Gehen Sie wie in der Splunk-Dokumentation beschrieben vor, um die Daten, die an den S3-Fehler-Bucket (S3-Backup) übermittelt wurden, wieder an Splunk weiterzuleiten.
- Weitere Informationen finden Sie unter Fehlersuche für das Splunk Add-on für Amazon Kinesis Firehose.

Fehlerbehebung bei Snowflake

In diesem Abschnitt werden allgemeine Schritte zur Fehlerbehebung bei der Verwendung von Snowflake als Ziel beschrieben

Die Firehose-Stream-Erstellung schlägt fehl

Wenn die Firehose-Stream-Erstellung für einen Stream fehlschlägt, der Daten an einen PrivateLink -fähigen Snowflake-Cluster liefert, bedeutet dies, dass die VPCE-ID für Firehose nicht erreichbar ist. Dies kann einen der folgenden Gründe haben:

- Falsche VPCE-ID. Vergewissern Sie sich, dass keine Tippfehler vorliegen.

- Firehose unterstützt URLs Snowflake ohne Region in der Vorschauversion nicht. Geben Sie die URL mit dem Snowflake Account Locator an. Weitere Informationen finden Sie in der Snowflake-Dokumentation.
- Vergewissern Sie sich, dass der Firehose-Stream in derselben AWS Region wie die Snowflake-Region erstellt wurde.
- Wenn das Problem weiterhin besteht, wenden Sie sich an den Support. AWS

Fehler bei der Zustellung

Überprüfen Sie Folgendes, wenn Daten nicht an Ihre Snowflake-Tabelle übermittelt werden. Daten, die bei der Snowflake-Zustellung fehlgeschlagen sind, werden zusammen mit einem Fehlercode und einer Fehlermeldung, die der Payload entsprechen, an den S3-Fehler-Bucket übermittelt. Im Folgenden sind einige häufig auftretende Fehlerszenarien aufgeführt. Die gesamte Liste der Fehlercodes finden Sie unterFehler bei der Lieferung von Snowflake-Daten.

- Fehlercode: Snowflake. DefaultRoleMissing: Zeigt an, dass die Snowflake-Rolle beim Erstellen des Firehose-Streams nicht konfiguriert wurde. Wenn die Snowflake-Rolle nicht konfiguriert ist, stellen Sie sicher, dass Sie eine Standardrolle für den angegebenen Snowflake-Benutzer festlegen.
- Fehlercode: Snowflake. ExtraColumns: Zeigt an, dass das Einfügen in Snowflake aufgrund zusätzlicher Spalten in der Eingabe-Payload abgelehnt wurde. Spalten, die in der Tabelle nicht vorhanden sind, sollten nicht angegeben werden. Beachten Sie, dass bei Snowflake-Spaltennamen Groß- und Kleinschreibung beachtet wird. Wenn die Lieferung mit diesem Fehler fehlschlägt, obwohl eine Spalte in der Tabelle vorhanden ist, stellen Sie sicher, dass die Groß-/Kleinschreibung des Spaltennamens in der Eingabe-Payload mit dem in der Tabellendefinition deklarierten Spaltennamen übereinstimmt.
- Fehlercode: Snowflake. MissingColumns: Zeigt an, dass das Einfügen in Snowflake aufgrund fehlender Spalten in der Eingabe-Payload abgelehnt wurde. Stellen Sie sicher, dass Werte für alle Spalten angegeben sind, die keine NULL-Werte zulassen.
- Fehlercode: Snowflake. InvalidInput: Dies kann passieren, wenn Firehose die bereitgestellte Eingabe-Payload nicht in ein gültiges JSON-Format parsen konnte. Stellen Sie sicher, dass die JSON-Nutzlast gut geformt ist und keine zusätzlichen doppelten Anführungszeichen, Anführungszeichen, Escape-Zeichen usw. enthält. Derzeit unterstützt Firehose nur ein einzelnes JSON-Element als Datensatznutzlast, JSON-Arrays werden nicht unterstützt.
- Fehlercode: Snowflake. InvalidValue: Zeigt an, dass die Lieferung aufgrund eines falschen Datentyps in der Eingabe-Payload fehlgeschlagen ist. Stellen Sie sicher, dass die in der Eingabe-

Payload angegebenen JSON-Werte dem in der Snowflake-Tabellendefinition deklarierten Datentyp entsprechen.

- Fehlercode: Snowflake. InvalidTableType: Zeigt an, dass der im Firehose-Stream konfigurierte Tabellentyp nicht unterstützt wird. Informationen zu den unterstützten Tabellen, Spalten und Datentypen finden Sie in den Einschränkungen unter Einschränkungen) von Snowpipe-Streaming.

> **Note**
>
> Wenn die Tabellendefinition oder die Rollenberechtigungen an Ihrem Snowflake-Ziel nach der Erstellung des Firehose-Streams geändert werden, kann es aus irgendeinem Grund mehrere Minuten dauern, bis Firehose diese Änderungen erkennt. Wenn Sie aus diesem Grund Lieferfehler feststellen, versuchen Sie, den Firehose-Stream zu löschen und neu zu erstellen.

Fehlerbehebung bei der Erreichbarkeit von Firehose-Endpunkten

Wenn die Firehose auf ein Timeout stößt, führen Sie die folgenden Schritte aus, um die Erreichbarkeit der Endpunkte zu testen:

- Prüfen Sie, ob API-Anfragen von einem Host in einer VPC gestellt werden. Der gesamte Datenverkehr von einer VPC erfordert die Einrichtung eines Firehose-VPC-Endpunkts. Weitere Informationen finden Sie unter Firehose verwenden mit AWS PrivateLink.
- Wenn der Verkehr von einem öffentlichen Netzwerk oder einer VPC kommt, bei der der Firehose-VPC-Endpunkt in einem bestimmten Subnetz eingerichtet ist, führen Sie die folgenden Befehle vom Host aus, um die Netzwerkkonnektivität zu überprüfen. Den Firehose-Endpunkt finden Sie unter Firehose-Endpunkte und Kontingente.
  - Verwenden Sie Tools wie traceroute odertcping, um zu überprüfen, ob die Netzwerkkonfiguration korrekt ist. Wenn das fehlschlägt, überprüfen Sie Ihre Netzwerkeinstellungen:

    Zum Beispiel:

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

    oder

```
tcping firehose.us-east-2.amazonaws.com 443
```

- Wenn es den Anschein hat, dass die Netzwerkeinstellungen korrekt sind und der folgende Befehl fehlschlägt, überprüfen Sie, ob sich die Amazon CA (Certficate Authority) in der Vertrauenskette befindet.

  Zum Beispiel:

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

Wenn die obigen Befehle erfolgreich sind, versuchen Sie es erneut mit der API, um zu sehen, ob von der API eine Antwort zurückgegeben wurde.

Fehlerbehebung bei HTTP-Endpunkten

In diesem Abschnitt werden allgemeine Schritte zur Fehlerbehebung beschrieben, wenn Amazon Data Firehose Daten an generische HTTP-Endpunktziele und Partnerziele wie Datadog, Dynatrace,, MongoDB, New Relic LogicMonitor, Splunk oder Sumo Logic übermittelt. Für die Zwecke dieses Abschnitts werden alle zutreffenden Ziele als HTTP-Endpunkte bezeichnet. Stellen Sie sicher, dass die in Ihrem Firehose-Stream angegebene IAM-Rolle auf den S3-Backup-Bucket und die Lambda-Funktion für die Datentransformation zugreifen kann (sofern die Datentransformation aktiviert ist). Stellen Sie außerdem sicher, dass die IAM-Rolle Zugriff auf die CloudWatch Protokollgruppe und die Protokollstreams hat, um Fehlerprotokolle zu überprüfen. Weitere Informationen finden Sie unter Firehose-Zugriff auf ein HTTP-Endpunktziel gewähren.

> **Note**
>
> Die Informationen in diesem Abschnitt gelten nicht für die folgenden Ziele: Splunk, OpenSearch Service, S3 und Redshift.

CloudWatch Logs

Es wird dringend empfohlen, CloudWatch Logging für zu aktivieren. Protokolle werden nur veröffentlicht, wenn bei der Lieferung an Ihr Ziel Fehler auftreten.

Ausnahmen vom Zielort

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"
}
```

Zielausnahmen weisen darauf hin, dass Firehose in der Lage ist, eine Verbindung zu Ihrem Endpunkt herzustellen und eine HTTP-Anfrage zu stellen, aber keinen 200-Antwortcode erhalten hat. 2xx-Antworten, die nicht 200s sind, führen ebenfalls zu einer Zielausnahme. Amazon Data Firehose protokolliert den Antwortcode und eine gekürzte Antwortnutzlast, die vom konfigurierten Endpunkt empfangen wurde, in Logs. CloudWatch Da Amazon Data Firehose den Antwortcode und die Nutzdaten ohne Änderung oder Interpretation protokolliert, ist es Sache des Endpunkts, den genauen Grund anzugeben, warum er die HTTP-Lieferanforderung von Amazon Data Firehose abgelehnt hat. Im Folgenden sind die häufigsten Empfehlungen zur Problembehandlung für diese Ausnahmen:

- 400: Zeigt an, dass Sie aufgrund einer Fehlkonfiguration Ihrer Amazon Data Firehose eine fehlerhafte Anfrage senden. Stellen Sie sicher, dass Sie die richtige URL, die richtigen allgemeinen Attribute, die Inhaltskodierung, den Zugriffsschlüssel und die richtigen Pufferhinweise für Ihr Ziel haben. Informationen zur erforderlichen Konfiguration finden Sie in der zielspezifischen Dokumentation.
- 401: Zeigt an, dass der Zugriffsschlüssel, den Sie für Ihren Firehose konfiguriert haben, falsch ist oder fehlt.
- 403: Zeigt an, dass der Zugriffsschlüssel, den Sie für Ihren Firehose-Stream konfiguriert haben, nicht berechtigt ist, Daten an den konfigurierten Endpunkt zu liefern.
- 413: Zeigt an, dass die Anforderungsnutzlast, die Amazon Data Firehose an den Endpunkt sendet, zu groß ist, als dass der Endpunkt sie verarbeiten könnte. Versuchen Sie, den Pufferhinweis auf die empfohlene Größe für Ihr Ziel zu reduzieren.

- 429: Zeigt an, dass Amazon Data Firehose Anfragen mit einer höheren Geschwindigkeit sendet, als das Ziel verarbeiten kann. Optimieren Sie Ihren Pufferhinweis, indem Sie Ihre Pufferzeit verlängern und damit Ihre Puffergröße and/or erhöhen (aber immer noch innerhalb der Grenzen Ihres Ziels).
- 5xx: Zeigt an, dass ein Problem mit dem Ziel vorliegt. Der Amazon Data Firehose-Service funktioniert immer noch einwandfrei.

> **Important**
>
> Wichtig: Obwohl dies die allgemeinen Empfehlungen zur Fehlerbehebung sind, können für bestimmte Endpunkte unterschiedliche Gründe für die Bereitstellung der Antwortcodes vorliegen. Daher sollten die endpunktspezifischen Empfehlungen zuerst befolgt werden.

Ungültige Antwort

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"
}
```

Ausnahmen für ungültige Antworten weisen darauf hin, dass Amazon Data Firehose eine ungültige Antwort vom Endpunktziel erhalten hat. Die Antwort muss den Antwortspezifikationen entsprechen. Andernfalls betrachtet Amazon Data Firehose den Zustellungsversuch als Fehlschlag und übermittelt dieselben Daten erneut, bis die konfigurierte Wiederholungsdauer überschritten ist. Amazon Data Firehose behandelt Antworten, die nicht den Antwortspezifikationen entsprechen, als Fehlschläge, selbst wenn die Antwort den Status 200 hat. Wenn Sie einen Amazon Data Firehose-kompatiblen

Endpunkt entwickeln, befolgen Sie die Antwortspezifikationen, um sicherzustellen, dass die Daten erfolgreich übermittelt werden.

Im Folgenden finden Sie einige der häufigsten Arten von ungültigen Antworten und deren Behebung:

- Ungültiges JSON oder unerwartete Felder: Zeigt an, dass die Antwort nicht ordnungsgemäß als JSON deserialisiert werden kann oder unerwartete Felder enthält. Stellen Sie sicher, dass die Antwort nicht inhaltskodiert ist.
- Fehlt RequestId: Zeigt an, dass die Antwort keine requestId enthält.
- RequestId stimmt nicht überein: Zeigt an, dass die requestId in der Antwort nicht mit der ausgehenden requestId übereinstimmt.
- Fehlender Zeitstempel: Zeigt an, dass die Antwort kein Zeitstempelfeld enthält. Das Zeitstempelfeld muss eine Zahl und keine Zeichenfolge sein.
- Fehlender Content-Type-Header: Zeigt an, dass die Antwort keinen „content-type: application/json“-Header enthält. Es wird kein anderer Inhaltstyp akzeptiert.

> **Important**
>
> Wichtig: Amazon Data Firehose kann Daten nur an Endpunkte liefern, die den Anforderungen und Antworten von Firehose entsprechen. Wenn Sie Ihr Ziel für einen Drittanbieter-Service konfigurieren, stellen Sie sicher, dass Sie den richtigen Amazon Data Firehose-kompatiblen Endpunkt verwenden, der sich wahrscheinlich vom öffentlichen Aufnahmeendpunkt unterscheidet. Zum Beispiel ist der Amazon Data Firehose-Endpunkt von Datadog, `https://aws-kinesis-http-intake.logs.datadoghq.com/` während es sein öffentlicher Endpunkt ist. `https://api.datadoghq.com/`

Andere häufige Fehler

Zusätzliche Fehlercodes und Definitionen werden im Folgenden aufgelistet.

- Fehlercode:. HttpEndpoint RequestTimeout - Zeigt an, dass die Antwort des Endpunkts länger als 3 Minuten gedauert hat. Wenn Sie der Besitzer des Ziels sind, verringern Sie die Antwortzeit des Zielendpunkts. Wenn Sie nicht der Eigentümer des Ziels sind, wenden Sie sich an den Eigentümer und fragen Sie, ob etwas getan werden kann, um die Antwortzeit zu verkürzen (d. h. den Pufferhinweis zu verringern, sodass weniger Daten pro Anfrage verarbeitet werden).

- Fehlercode: HttpEndpoint. ResponseTooLarge - Zeigt an, dass die Antwort zu groß ist. Die Antwort muss weniger als 1 MiB einschließlich der Header betragen.
- Fehlercode: HttpEndpoint. ConnectionFailed - Zeigt an, dass keine Verbindung mit dem konfigurierten Endpunkt hergestellt werden konnte. Dies kann auf einen Tippfehler in der konfigurierten URL zurückzuführen sein, der Endpunkt ist für Amazon Data Firehose nicht zugänglich oder es dauert zu lange, bis der Endpunkt auf die Verbindungsanfrage reagiert.
- Fehlercode:. HttpEndpoint ConnectionReset - Zeigt an, dass eine Verbindung hergestellt, aber vom Endpunkt zurückgesetzt oder vorzeitig geschlossen wurde.
- Fehlercode:. HttpEndpoint SSLHandshakeFehler — Zeigt an, dass ein SSL-Handshake mit dem konfigurierten Endpunkt nicht erfolgreich abgeschlossen werden konnte.

Problembehandlung bei MSK As Source

In diesem Abschnitt werden allgemeine Schritte zur Fehlerbehebung bei der Verwendung von MSK As Source beschrieben

> **Note**
>
> Informationen zur Behebung von Problemen bei der Verarbeitung, Transformation oder S3-Bereitstellung finden Sie in den vorherigen Abschnitten

Fehler bei der Schlaucherstellung

Überprüfen Sie Folgendes, wenn Ihr Schlauch mit MSK als Quelle nicht erstellt werden kann:

- Stellen Sie sicher, dass sich der Quell-MSK-Cluster im Status Aktiv befindet.
- Wenn Sie private Konnektivität verwenden, stellen Sie sicher, dass Private Link auf dem Cluster aktiviert ist.

  Wenn Sie öffentliche Konnektivität verwenden, stellen Sie sicher, dass der öffentliche Zugriff auf dem Cluster aktiviert ist.
- Wenn Sie private Konnektivität verwenden, stellen Sie sicher, dass Sie eine ressourcenbasierte Richtlinie hinzufügen, die es Firehose ermöglicht, Private Links zu erstellen. Siehe auch: Kontoübergreifende MSK-Berechtigungen.

- Stellen Sie sicher, dass die Rolle in der Quellkonfiguration berechtigt ist, Daten aus dem Thema des Clusters aufzunehmen.
- Stellen Sie sicher, dass Ihre VPC-Sicherheitsgruppen eingehenden Datenverkehr an den Ports zulassen, die von den Bootstrap-Servern des Clusters verwendet werden.

Schlauch suspendiert

Prüfen Sie Folgendes, wenn sich Ihr Schlauch im Zustand SUSPENDIERT befindet

- Stellen Sie sicher, dass sich der Quell-MSK-Cluster im Status Aktiv befindet.
- Stellen Sie sicher, dass das Quellthema vorhanden ist. Falls das Thema gelöscht und neu erstellt wurde, müssen Sie auch den Firehose-Stream löschen und neu erstellen.

Schlauch mit Gegendruck

Der Wert von DataReadFromSource .Backpressured ist 1, wenn jede Partition überschritten wird oder wenn BytesPerSecondLimit der normale Übertragungsfluss langsam ist oder gestoppt wird.

- Wenn Sie darauf stoßen, überprüfen Sie BytesPerSecondLimit bitte die DataReadFromSource .Bytes-Metrik und fordern Sie eine Erhöhung des Limits an.
- Überprüfen Sie die CloudWatch Protokolle, Zielmetriken, Datenumwandlungsmetriken und Metriken zur Formatkonvertierung, um die Engpässe zu identifizieren.

Falsche Datenaktualität

Die Aktualität der Daten scheint falsch zu sein

- Firehose berechnet die Datenaktualität anhand des Zeitstempels des verbrauchten Datensatzes. Um sicherzustellen, dass dieser Zeitstempel korrekt aufgezeichnet wird, wenn der Produzentendatensatz in den Broker-Protokollen von Kafka gespeichert wird, stellen Sie die Konfiguration des Zeitstempeltyps Kafka-Thema auf `message.timestamp.type=LogAppendTime`.

Verbindungsprobleme mit dem MSK-Cluster

Im folgenden Verfahren wird erklärt, wie Sie die Konnektivität zu MSK-Clustern überprüfen können. Einzelheiten zur Einrichtung des Amazon MSK-Clients finden Sie unter Erste Schritte mit Amazon MSK im Amazon Managed Streaming for Apache Kafka Developer Guide.

Um die Konnektivität zu MSK-Clustern zu überprüfen

1. Erstellen Sie eine UNIX-basierte (vorzugsweise AL2) EC2 Amazon-Instance. Wenn Sie in Ihrem Cluster nur VPC-Konnektivität aktiviert haben, stellen Sie sicher, dass Ihre EC2 Instance in derselben VPC ausgeführt wird. Stellen Sie per SSH eine Verbindung zur Instanz her, sobald sie verfügbar ist. Weitere Informationen finden Sie in diesem Tutorial im EC2 Amazon-Benutzerhandbuch.
2. Installieren Sie Java mithilfe des Yum-Paketmanagers, indem Sie den folgenden Befehl ausführen. Weitere Informationen finden Sie in den Installationsanweisungen im Amazon Corretto 8-Benutzerhandbuch.

   ```
   sudo yum install java-1.8.0
   ```

3. Installieren Sie den AWS Client, indem Sie den folgenden Befehl ausführen.

   ```
   curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
   unzip awscliv2.zip
   sudo ./aws/install
   ```

4. Laden Sie die Version 2.6* des Apache Kafka-Clients herunter, indem Sie den folgenden Befehl ausführen.

   ```
   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. Wechseln Sie zum Verzeichnis `kafka_2.12-2.6.2/libs` und führen Sie dann den folgenden Befehl aus, um die Amazon-MSK-IAM-JAR-Datei herunterzuladen.

   ```
   wget https://github.com/aws/aws-msk-iam-auth/releases/download/v1.1.3/aws-msk-iam-auth-1.1.3-all.jar
   ```

6. Erstellen Sie die `client.properties` Datei im Kafka-Ordner bin.

7. `awsRoleArn`Ersetzen Sie es durch den Rollen-ARN, den Sie in Ihrer Firehose verwendet haben, `SourceConfiguration` und überprüfen Sie den Speicherort des Zertifikats. Erlauben Sie Ihrem AWS Client-Benutzer, die Rolle zu übernehmen. `awsRoleArn` AWS Der Client-Benutzer wird versuchen, die Rolle anzunehmen, die Sie hier angegeben haben.

```
[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. Führen Sie den folgenden Kafka-Befehl aus, um die Themen aufzulisten. Wenn Ihre Verbindung öffentlich ist, verwenden Sie die öffentlichen Endpunkt-Bootstrap-Server. Wenn Ihre Verbindung privat ist, verwenden Sie die privaten Endpunkt-Bootstrap-Server.

```
bin/kafka-topics.sh --list --bootstrap-server <bootstrap servers> --command-config
 bin/client.properties
```

   Wenn die Anfrage erfolgreich ist, sollten Sie eine Ausgabe sehen, die dem folgenden Beispiel ähnelt.

```
[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. Wenn Sie Probleme mit der Ausführung des vorherigen Skripts haben, überprüfen Sie, ob die von Ihnen angegebenen Bootstrap-Server über den angegebenen Port erreichbar sind. Zu diesem Zweck können Sie Telnet oder ein ähnliches Hilfsprogramm herunterladen und verwenden, wie im folgenden Befehl gezeigt.

   ```
   sudo yum install telnet
   telnet <bootstrap servers><port>
   ```

   Wenn die Anfrage erfolgreich ist, erhalten Sie die folgende Ausgabe. Das bedeutet, dass Sie innerhalb Ihrer lokalen VPC eine Verbindung zu Ihrem MSK-Cluster herstellen können und die Bootstrap-Server auf dem angegebenen Port fehlerfrei sind.

   ```
   Connected to ..
   ```

10. Wenn die Anfrage nicht erfolgreich ist, überprüfen Sie die Regeln für eingehende Nachrichten in Ihrer VPC-Sicherheitsgruppe. Als Beispiel könnten Sie die folgenden Eigenschaften für die Regel für eingehenden Datenverkehr verwenden.

    ```
    Type: All traffic
    Port: Port used by the bootstrap server (e.g. 14001)
    Source: 0.0.0.0/0
    ```

    Versuchen Sie erneut, die Telnet-Verbindung herzustellen, wie im vorherigen Schritt gezeigt. Wenn Sie immer noch keine Verbindung herstellen können oder Ihre Firehose-Verbindung immer noch ausfällt, wenden Sie sich an den AWS Support.

Amazon Data Firehose-Kontingent

In diesem Abschnitt werden aktuelle Kontingente, früher als Limits bezeichnet, innerhalb von Amazon Data Firehose beschrieben. Jedes Kontingent gilt pro Region, sofern nicht anders angegeben.

Die Servicekontingenten-Konsole ist ein zentraler Ort, an dem Sie Ihre Kontingente für AWS Dienste anzeigen und verwalten und eine Erhöhung des Kontingents für viele der von Ihnen verwendeten Ressourcen beantragen können. Verwenden Sie die von uns bereitgestellten Kontingentinformationen, um Ihre AWS Infrastruktur zu verwalten. Planen Sie Anfragen zur Erhöhung der Kontingente im Voraus vor dem Zeitpunkt, zu dem Sie sie benötigen.

Weitere Informationen finden Sie unter Amazon Data Firehose Endpoints and Quotas in der. Allgemeine Amazon Web Services-Referenz

Der folgende Abschnitt zeigt, dass Amazon Data Firehose das folgende Kontingent hat.

- Mit Amazon MSK als Quelle für den Firehose-Stream hat jeder Firehose-Stream ein Standardkontingent von 10% Lesedurchsatz pro Partition und eine maximale Datensatzgröße MB/sec von 10 MB.
- Mit Amazon MSK als Quelle für den Firehose-Stream gibt es eine maximale Datensatzgröße von 6 MB, wenn AWS Lambda aktiviert ist, und eine maximale Datensatzgröße von 10 MB, wenn Lambda deaktiviert ist. AWS Lambda begrenzt seinen eingehenden Datensatz auf 6 MB, und Amazon Data Firehose leitet Datensätze über 6 MB an einen Fehler-S3-Bucket weiter. Wenn Lambda deaktiviert ist, begrenzt Firehose seinen eingehenden Datensatz auf 10 MB. Wenn Amazon Data Firehose eine Datensatzgröße von Amazon MSK erhält, die größer als 10 MB ist, übermittelt Amazon Data Firehose diesen Datensatz an den S3-Fehler-Bucket und sendet Cloudwatch-Metriken an Ihr Konto. Weitere Informationen zu AWS Lambda-Grenzwerten finden Sie unter Lambda-Kontingente.
- Wenn die dynamische Partitionierung in einem Firehose-Stream aktiviert ist, gibt es ein Standardkontingent von 500 aktiven Partitionen, die für diesen Firehose-Stream erstellt werden können. Die Anzahl der aktiven Partitionen ist die Gesamtzahl der aktiven Partitionen innerhalb des Bereitstellungspuffers. Wenn die dynamische Partitionierungsabfrage beispielsweise 3 Partitionen pro Sekunde erstellt und Sie eine Konfiguration mit Pufferhinweisen haben, die alle 60 Sekunden eine Übermittlung auslöst, dann haben Sie im Durchschnitt 180 aktive Partitionen. Sobald Daten in einer Partition geliefert wurden, ist diese Partition nicht mehr aktiv. Wenn Sie mehr Partitionen benötigen, können Sie mehr Firehose-Streams erstellen und die aktiven Partitionen auf diese verteilen.

- Wenn die dynamische Partitionierung auf einem Firehose-Stream aktiviert ist, wird für jede aktive Partition ein maximaler Durchsatz von 1 GB pro Sekunde unterstützt.
- Für jedes Konto gilt das folgende Kontingent für die Anzahl der Firehose-Streams pro Region:
  - USA Ost (Nord-Virginia), USA Ost (Ohio), USA West (Oregon), Europa (Irland), Asien-Pazifik (Tokio): 5.000 Firehose-Streams
  - Europa (Frankfurt), Europa (London), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney), Asien-Pazifik (Seoul), Asien-Pazifik (Mumbai), AWS GovCloud (US-West), Kanada (West), Kanada (Zentral): 2.000 Firehose-Streams
  - Europa (Paris), Europa (Mailand), Europa (Stockholm), Asien-Pazifik (Hongkong), Asien-Pazifik (Osaka), Südamerika (Sao Paulo), China (Ningxia), China (Peking), Naher Osten (Bahrain), AWS GovCloud (US-Ost), Afrika (Kapstadt): 500 Firehose-Streams
  - Europa (Zürich), Europa (Spanien), Asien-Pazifik (Hyderabad), Asien-Pazifik (Jakarta), Asien-Pazifik (Melbourne), Naher Osten (VAE), Israel (Tel Aviv), Kanada West (Calgary), Kanada (Zentral), Asien-Pazifik (Malaysia), Asien-Pazifik (Thailand), Mexiko (Zentral): 100 Firehose-Streams
  - Wenn Sie diesen Wert überschreiten, hat ein Aufruf von CreateDeliveryStream die Ausnahme `LimitExceededException` zur Folge. Um dieses Kontingent zu erhöhen, können Sie Service Quotas (Service-Kontingente) verwenden, sofern in Ihrer Region verfügbar. Informationen zum Verwenden von Service Quotas finden Sie unter Service Quotas erhöhen.
- Wenn Direct PUT als Datenquelle konfiguriert ist, stellt jeder Firehose-Stream das folgende kombinierte Kontingent für PutRecordund PutRecordBatchAnfragen bereit:
  - Für USA Ost (Nord-Virginia), USA West (Oregon) und Europa (Irland): 500.000records/second, 2,000 requests/second, and 5 MiB/second.
  - Für andere AWS-Regionen: 100.000records/second, 1,000 requests/second, and 1 MiB/second.

  Wenn bei einem Direct PUT-Stream eine Drosselung aufgrund höherer Datenaufnahmemengen auftritt, die die Durchsatzkapazität eines Firehose-Streams überschreiten, erhöht Amazon Data Firehose automatisch das Durchsatzlimit des Streams, bis die Drosselung eingedämmt ist. Je nach erhöhtem Durchsatz und Drosselung kann es länger dauern, bis Firehose den Durchsatz eines Streams auf das gewünschte Niveau erhöht. Versuchen Sie daher weiterhin, die fehlgeschlagenen Dateneingabe-Datensätze erneut zu überprüfen. Wenn Sie erwarten, dass das Datenvolumen bei plötzlichen großen Datenstößen ansteigt, oder wenn Ihr neuer Stream einen höheren Durchsatz als den standardmäßigen Durchsatzgrenzwert benötigt, fordern Sie eine Erhöhung des Durchsatzlimits an.

Es gibt drei Quotenstufen, die proportional zu den Quoten sind. Wenn Sie beispielsweise die Durchsatzquote in USA Ost (Nord-Virginia), USA West (Oregon) oder Europa (Irland) auf 10 erhöhenMiB/second, the other two quota increase to 4,000 requests/second and 1,000,000 records/second.

> **Note**
>
> - Verwenden Sie Limits und Kontingente auf Ressourcenebene nicht dazu, Ihre Nutzung des Dienstes zu kontrollieren.
> - Wenn Kinesis Data Streams als Datenquelle konfiguriert ist, gilt dieses Kontingent nicht und Amazon Data Firehose skaliert ohne Limit nach oben und unten.
> - Wenn das erhöhte Kontingent wesentlich höher als der kontinuierliche Datenverkehr ist, werden kleine Datenpakete an die Ziele geliefert. Dies ist ineffizient und kann zu höheren Kosten beim Zielservice führen. Erhöhen Sie das Kontingent nur soweit, dass es dem aktuellen Datenverkehr entspricht. Steigt der Datenverkehr, können Sie das Kontingent erneut erhöhen.
> - Kleinere Datensätze können zu höheren Kosten führen. Die Preise für die Datenaufnahme von Firehose basieren auf der Anzahl der Datensätze, die Sie an den Service senden, multipliziert mit der Größe jedes Datensatzes, aufgerundet auf die nächsten 5 KB (5120 Byte). Bei gleichem Volumen eingehender Daten (Byte) wären die Kosten also höher, wenn es eine größere Anzahl eingehender Datensätze gibt. Wenn das gesamte eingehende Datenvolumen beispielsweise 5 MiB beträgt, kostet das Senden von 5 MiB an Daten über 5 000 Datensätze mehr als das Senden derselben Datenmenge mit 1 000 Datensätzen. Weitere Informationen finden Sie unter Amazon Data Firehose im AWS Rechner.

- Jeder Firehose-Stream speichert Datensätze für bis zu 24 Stunden, falls das Lieferziel nicht verfügbar ist und wenn die Quelle nicht verfügbar ist DirectPut. Wenn die Quelle Kinesis Data Streams (KDS) ist und das Ziel nicht verfügbar ist, werden die Daten basierend auf Ihrer KDS-Konfiguration beibehalten.
- Die maximale Größe eines Datensatzes, der vor der Base64-Kodierung an Amazon Data Firehose gesendet wird, beträgt 1.000 KiB.
- Der PutRecordBatch-Vorgang kann pro Aufruf bis zu 500 Datensätze oder 4 MiB aufnehmen, je nachdem, welcher Wert kleiner ist. Dieses Kontingent kann nicht geändert werden.

- Jeder der folgenden Operationen kann bis zu fünf Aufrufe pro Sekunde bereitstellen, was eine feste Grenze darstellt.
  - `CreateDeliveryStream`
  - `DeleteDeliveryStream`
  - `DescribeDeliveryStream`
  - `ListDeliveryStreams`
  - `UpdateDestination`
  - `TagDeliveryStream`
  - `UntagDeliveryStream`
  - `ListTagsForDeliveryStream`
  - `StartDeliveryStreamEncryption`
  - `StopDeliveryStreamEncryption`
- Die Pufferintervallspuren reichen von 60 Sekunden bis zu 900 Sekunden.
- Für die Lieferung von Amazon Data Firehose an Amazon Redshift werden nur öffentlich zugängliche Amazon Redshift Redshift-Cluster unterstützt.
- Die Dauer der Wiederholungsversuche liegt bei Amazon Redshift und Service Delivery zwischen 0 Sekunden und OpenSearch 7.200 Sekunden.
- Wenn das Ziel Amazon S3, Amazon Redshift oder OpenSearch Service ist, erlaubt Amazon Data Firehose bis zu 5 ausstehende Lambda-Aufrufe pro Shard. Für Splunk beträgt das Kontingent 10 ausstehende Lambda-Aufrufe pro Shard.
- Sie können einen CMK vom Typ verwenden, um bis `CUSTOMER_MANAGED_CMK` zu 500 Firehose-Streams zu verschlüsseln.

Dokumentverlauf

In der folgenden Tabelle werden die wichtigen Änderungen an der Amazon Data Firehose beschrieben.

Änderungen	Beschreibung	Änderungsdatum
Entfernung der Datenbank als Quelle (öffentliche Vorschau)	Datenbank als Quelle (öffentliche Vorschau) ist jetzt entfernt.	24. September 2025
Unterstützung für die Multi-Katalog-Hierarchie von Glue hinzugefügt	Dies vereinfacht die Firehose-Integration mit Amazon S3 S3-Tabellen, ohne dass Ressourcenlinks zwischen dem Standarddatenkatalog und `S3TablesCatalog` erforderlich sind. Siehe Einen Firehose-Stream für Amazon S3-Tabellen einrichten.	14. Mai 2025
Datenbank als Quelle hinzugefügt (öffentliche Vorschau)	Sie können jetzt Datenbankänderungen in Apache Iceberg Tables in Amazon S3 replizieren.	15. November 2024
Version für allgemeine Verfügbarkeit (GA) für Apache Iceberg-Tabellen als Ziel hinzugefügt	Sie können einen Firehose-Stream mit Apache Iceberg Tables als Ziel erstellen. Siehe Stellen Sie Daten an Apache Iceberg Tables bereit.	30. September 2024
Beispiele für Datentypen hinzugefügt	Es wurden Beispiele für unterstützte Datentypen für Apache Iceberg-Tabellen hinzugefügt. Siehe Verstehen Sie die unterstützten Datentypen.	22. August 2024
Start einer neuen Region	Amazon Data Firehose ist jetzt im asiatisch-pazifischen Raum (Malaysia) verfügbar. Siehe Amazon Data Firehose-Kontingent.	22. August 2024

Änderungen	Beschreibung	Änderungsdatum
Apache Iceberg Tables als Ziel hinzugefügt (öffentliche Vorschau)	Sie können einen Firehose-Stream mit Apache Iceberg Tables als Ziel erstellen. Siehe Stellen Sie Daten an Apache Iceberg Tables bereit.	25. Juli 2024
Hinweise zum Puffern für Snowflake	Snowflake unterstützt jetzt Pufferhinweise. Siehe the section called “Konfigurieren Sie die Zieleinstellungen für Snowflake”.	25. Juli 2024
Snowflake als Ziel in neuen Regionen	Snowflake ist jetzt als Reiseziel im asiatisch-pazifischen Raum (Singapur), im asiatisch-pazifischen Raum (Seoul) und im asiatisch-pazifischen Raum (Sydney) verfügbar. Siehe the section called “Konfigurieren Sie die Zieleinstellungen für Snowflake”.	25. Juli 2024
Die Abschnitte der Benutzerhandbücher wurden neu strukturiert	Vereinfachte Navigation für Abschnitte im Benutzerhandbuch. Siehe Daten an einen Firehose-Stream senden und Beheben von Fehlern.	5. Juli 2024
Amazon Data Firehose lässt sich integrieren mit AWS Secrets Manager	Mit Secrets Manager können Sie jetzt auf Ihre Geheimnisse zugreifen und die Rotation von Anmeldeinformationen sicher automatisieren. Siehe the section called “Authentifizieren Sie sich mit AWS Secrets Manager”.	06. Juni 2024
Unterstützung für die Aufnahme von Logs für Dynatrace wurde hinzugefügt	Sie können jetzt Protokolle und Ereignisse zur weiteren Analyse an Dynatrace senden. Siehe the section called “Konfigurieren Sie die Zieleinstellungen für Dynatrace”.	18. April 2024
Version mit allgemeiner Verfügbarkeit (GA) für Snowflake als Ziel	Snowflake ist jetzt allgemein als Ziel verfügbar. Siehe the section called “Konfigurieren Sie die Zieleinstellungen für Snowflake”.	17. April 2024

Änderungen	Beschreibung	Änderungsdatum
Amazon Kinesis Data Firehose ist jetzt als Amazon Data Firehose bekannt	Amazon Kinesis Data Firehose wurde in Amazon Data Firehose umbenannt. Siehe Was ist Amazon Data Firehose	9. Februar 2024
Snowflake wurde als Ziel hinzugefügt (öffentliche Vorschau)	Sie können einen Firehose-Stream mit Snowflake als Ziel erstellen. Siehe the section called “Konfigurieren Sie die Zieleinstellungen für Snowflake”.	19. Januar 2024
Automatische Dekomprimierung von Logs hinzugefügt CloudWatch	Sie können die Dekomprimierung für neue oder bestehende Streams aktivieren, um dekomprimierte CloudWatch Logdaten an Firehose-Ziele zu senden. Siehe the section called “ CloudWatch Logs an Firehose senden”.	15. Dezember 2023
Splunk Observability Cloud als Ziel hinzugefügt	Sie können einen Firehose-Stream mit Splunk Observability Cloud als Ziel erstellen. Siehe the section called “Konfigurieren Sie die Zieleinstellungen für Splunk Observability Cloud”.	3. Oktober 2023
Amazon Managed Streaming für Apache Kafka als Datenquelle hinzugefügt	Sie können Amazon MSK jetzt so konfigurieren, dass Informationen an einen Firehose-Stream gesendet werden. Siehe the section called “Quelleinstellungen für Amazon MSK konfigurieren”.	26. September 2023
Unterstützung für den Typ DocumentID für das Service-Ziel hinzugefügt OpenSearch	Wenn OpenSearch Service das Ziel Ihres Firehose-Streams ist, gibt der DocumentID-Typ die Methode zum Einrichten der Dokument-ID an. Die unterstützten Methoden sind die von Firehose generierte Dokument-ID und die vom OpenSearch Service generierte Dokument-ID. Siehe the section called “Zieleinstellungen konfigurieren”.	10. Mai 2023

Änderungen	Beschreibung	Änderungsdatum
Unterstützung für dynamische Partitionierung hinzugefügt	Unterstützung für die kontinuierliche dynamisch e Partitionierung der Streaming-Daten in Amazon Data Firehose hinzugefügt. Siehe Streaming-Daten partitionieren.	31. August 2021
Ein Thema zu benutzerdefinierte n Präfixen wurde hinzugefügt.	Ein Thema zu den Ausdrücken, die Sie für die Erstellung eines benutzerdefinierten Präfixes für an Amazon 3 übermittelte Daten verwenden können, wurde hinzugefügt. Siehe the section called “Verstehe n Sie benutzerdefinierte Präfixe für Amazon S3 S3-Objekte”.	20. Dezember 2018
Neues Amazon Data Firehose-Tutorial hinzugefü gt	Es wurde ein Tutorial hinzugefügt, das zeigt, wie Amazon VPC-Flow-Logs über Amazon Data Firehose an Splunk gesendet werden. Siehe VPC-Flow-Logs mithilfe von Amazon Data Firehose in Splunk aufnehmen.	30. Oktober 2018
Vier neue Amazon Data Firehose-Regionen hinzugefügt	Paris, Mumbai, Sao Paulo und London hinzugefügt. Weitere Informationen finden Sie unter Amazon Data Firehose-Kontingent.	27. Juni 2018
Zwei neue Amazon Data Firehose-Regionen hinzugefügt	Seoul und Montreal hinzugefügt. Weitere Informati onen finden Sie unter Amazon Data Firehose-Kontingent.	13. Juni 2018
Neue Kinesis Streams als Quell-Feature	Kinesis Streams als potenzielle Quelle für Datensätz e für einen Firehose-Stream hinzugefügt. Weitere Informationen finden Sie unter Wählen Sie Quelle und Ziel für Ihren Firehose-Stream.	18. August 2017

Änderungen	Beschreibung	Änderungsdatum
Aktualisierung der Konsolen-Dokumentation	Der Assistent zur Erstellung von Firehose-Streams wurde aktualisiert. Weitere Informationen finden Sie unter Tutorial: Einen Firehose-Stream von der Konsole aus erstellen.	19. Juli 2017
Neue Datentransformation	Sie können Amazon Data Firehose so konfigurieren, dass Ihre Daten vor der Datenlieferung transformiert werden. Weitere Informationen finden Sie unter Transformieren Sie Quelldaten in Amazon Data Firehose.	19. Dezember 2016
Neuer COPY-Wiederholungsversuch von Amazon Redshift	Sie können Amazon Data Firehose so konfigurieren, dass ein COPY-Befehl an Ihren Amazon Redshift Redshift-Cluster erneut ausgeführt wird, falls er fehlschlägt. Weitere Informationen finden Sie unter Tutorial: Einen Firehose-Stream von der Konsole aus erstellen, Verstehen Sie die Datenlieferung in Amazon Data Firehose und Amazon Data Firehose-Kontingent.	18. Mai 2016
Neues Amazon Data Firehose-Ziel, Amazon Service OpenSearch	Sie können einen Firehose-Stream mit Amazon OpenSearch Service als Ziel erstellen. Weitere Informationen finden Sie unter Tutorial: Einen Firehose-Stream von der Konsole aus erstellen, Verstehen Sie die Datenlieferung in Amazon Data Firehose und Firehose Zugang zu einem Ziel des öffentlichen OpenSearch Dienstes gewähren.	19. April 2016
Neue, verbesserte CloudWatch Metriken und Funktionen zur Fehlerbehebung	Überwachen Sie Amazon Data Firehose und Fehler in Amazon Data Firehose beheben wurden aktualisiert.	19. April 2016
Neuer verbesserter Kinesis-Agent	Aktualisiert Kinesis-Agent für das Senden von Daten konfigurieren.	11. April 2016

Änderungen	Beschreibung	Änderungsdatum
Neue Kinesis-Agenten	Kinesis-Agent für das Senden von Daten konfigurieren hinzugefügt.	2. Oktober 2015
Erstversion	Erste Veröffentlichung des Amazon Data Firehose Developer Guide.	4. Oktober 2015