亚马逊云科技

Developer Guide

Amazon Data Firehose

Amazon Data Firehose: Developer Guide

Table of Contents

Amazon Data Firehose was previously known as Amazon Kinesis Data Firehose

What Is Amazon Data Firehose?

Amazon Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service, Amazon OpenSearch Serverless, Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers, including Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Coralogix, and Elastic. With Amazon Data Firehose, you don't need to write applications or manage resources. You configure your data producers to send data to Amazon Data Firehose, and it automatically delivers the data to the destination that you specified. You can also configure Amazon Data Firehose to transform your data before delivering it.

For more information about Amazon big data solutions, see Big Data on Amazon. For more information about Amazon streaming data solutions, see What is Streaming Data?

> **Note**
>
> Note the latest Amazon Streaming Data Solution for Amazon MSK that provides Amazon CloudFormation templates where data flows through producers, streaming storage, consumers, and destinations.

Key Concepts

As you get started with Amazon Data Firehose, you can benefit from understanding the following concepts:

Firehose stream

The underlying entity of Amazon Data Firehose. You use Amazon Data Firehose by creating a Firehose stream and then sending data to it. For more information, see Creating a Firehose stream and Sending Data to a Firehose stream.

record

The data of interest that your data producer sends to a Firehose stream. A record can be as large as 1,000 KB.

data producer

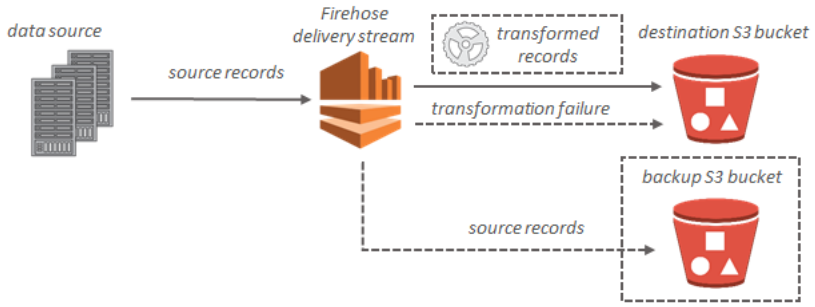
Producers send records to Firehose streams. For example, a web server that sends log data to a Firehose stream is a data producer. You can also configure your Firehose stream to automatically read data from an existing Kinesis data stream, and load it into destinations. For more information, see Sending Data to a Firehose stream.

buffer size and buffer interval

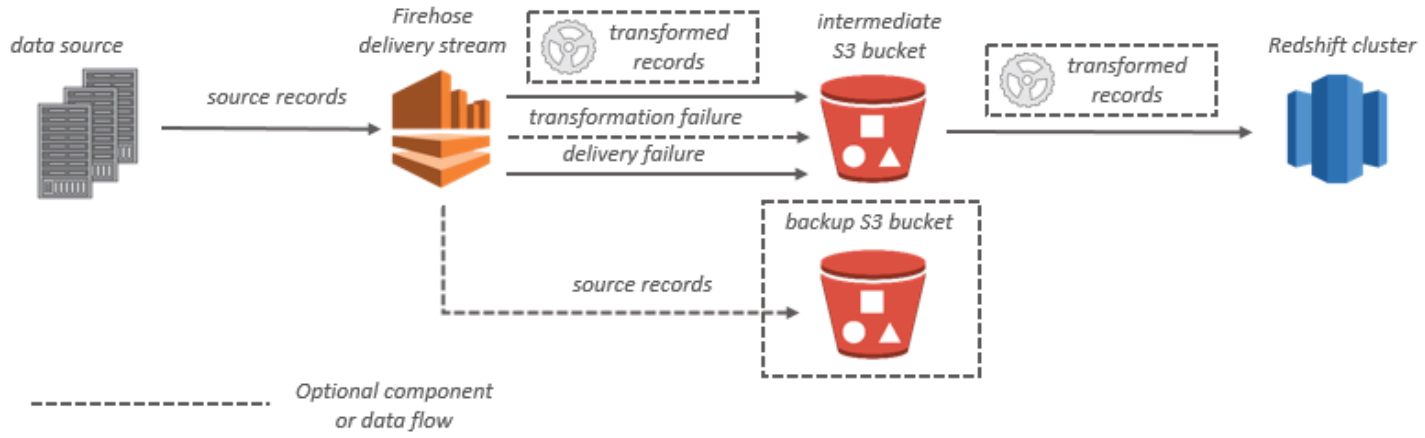
Amazon Data Firehose buffers incoming streaming data to a certain size or for a certain period of time before delivering it to destinations. **Buffer Size** is in MBs and **Buffer Interval** is in seconds.

Data Flow

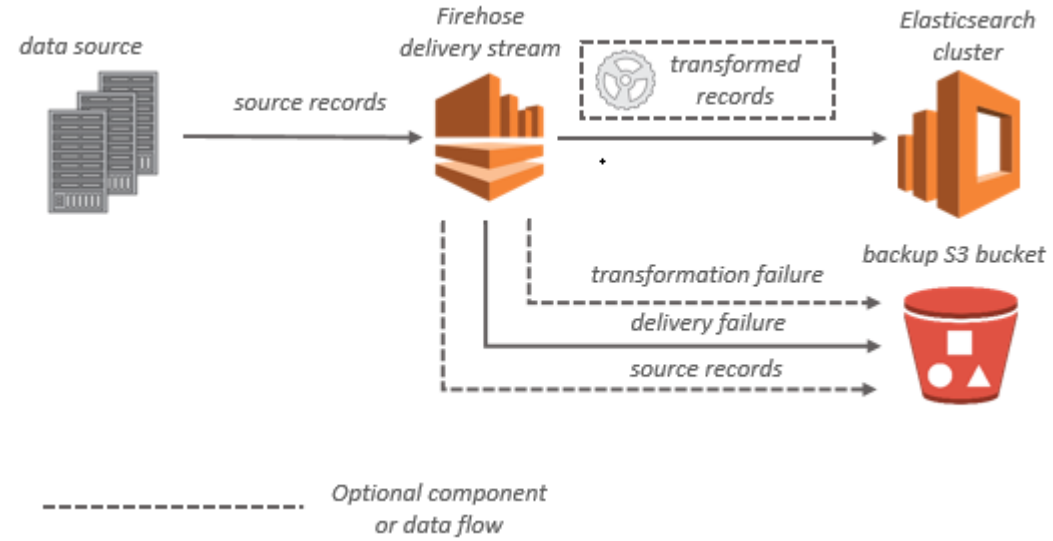
For Amazon S3 destinations, streaming data is delivered to your S3 bucket. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket.



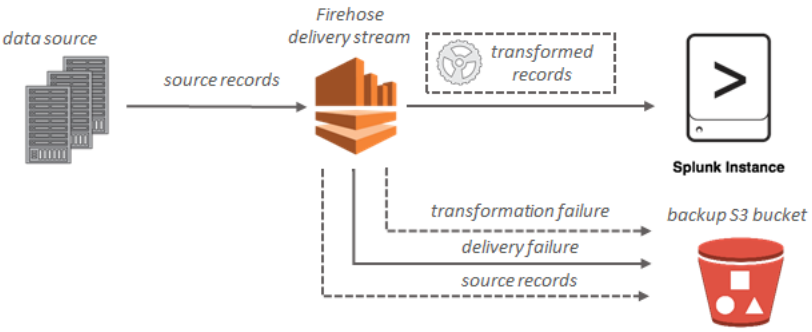
For Amazon Redshift destinations, streaming data is delivered to your S3 bucket first. Amazon Data Firehose then issues an Amazon Redshift **COPY** command to load data from your S3 bucket to your Amazon Redshift cluster. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket.



For OpenSearch Service destinations, streaming data is delivered to your OpenSearch Service cluster, and it can optionally be backed up to your S3 bucket concurrently.



For Splunk destinations, streaming data is delivered to Splunk, and it can optionally be backed up to your S3 bucket concurrently.



Setting Up for Amazon Data Firehose

Before you use Amazon Data Firehose for the first time, complete the following tasks.

Tasks

- Sign Up for Amazon
- Optional: Download Libraries and Tools

Sign Up for Amazon

When you sign up for Amazon Web Services (Amazon), your Amazon account is automatically signed up for all services in Amazon, including Amazon Data Firehose. You are charged only for the services that you use.

If you have an Amazon account already, skip to the next task. If you don't have an Amazon account, use the following procedure to create one.

To sign up for an Amazon account

1. Open https://portal.amazonaws.cn/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an Amazon Web Services account, an *Amazon Web Services account root user* is created. The root user has access to all Amazon Web Services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

Optional: Download Libraries and Tools

The following libraries and tools will help you work with Amazon Data Firehose programmatically and from the command line:

- The Firehose API Operations is the basic set of operations that Amazon Data Firehose supports.

- The Amazon SDKs for Go, Java, .NET, Node.js, Python, and Ruby include Amazon Data Firehose support and samples.

  If your version of the Amazon SDK for Java does not include samples for Amazon Data Firehose, you can also download the latest Amazon SDK from GitHub.
- The Amazon Command Line Interface supports Amazon Data Firehose. The Amazon CLI enables you to control multiple Amazon services from the command line and automate them through scripts.

Creating a Firehose stream

You can use the Amazon Web Services Management Console or an Amazon SDK to create a Firehose stream to your chosen destination.

You can update the configuration of your Firehose stream at any time after it's created, using the Amazon Data Firehose console or UpdateDestination. Your Firehose stream remains in the `ACTIVE` state while your configuration is updated, and you can continue to send data. The updated configuration normally takes effect within a few minutes. The version number of a Firehose stream is increased by a value of 1 after you update the configuration. It is reflected in the delivered Amazon S3 object name. For more information, see Amazon S3 Object Name Format.

The following topics describe how to create a Firehose stream:

Topics

Source, Destination, and Name

1. Sign in to the Amazon Web Services Management Console and open the Amazon Data Firehose console at https://console.aws.amazon.com/firehose
2. Choose **Create Firehose stream**.
3. Enter values for the following fields:

   Source

   - **Direct PUT:** Choose this option to create a Firehose stream that producer applications write to directly. Currently, the following are Amazon services and agents and open source services that are integrated with Direct PUT in Amazon Data Firehose:
     - Amazon SDK
     - Amazon Lambda

- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- Amazon Cloud Metric Streams
- Amazon IOT
- Amazon Eventbridge
- Amazon Simple Email Service
- Amazon SNS
- Amazon WAF web ACL logs
- Amazon API Gateway - Access logs
- Amazon Pinpoint
- Amazon MSK Broker Logs
- Amazon Route 53 Resolver query logs
- Amazon Network Firewall Alerts Logs
- Amazon Network Firewall Flow Logs
- Amazon Elasticache Redis SLOWLOG
- Kinesis Agent (linux)
- Kinesis Tap (windows)
- Fluentbit
- Fluentd
- Apache Nifi
- Snowflake

- **Kinesis stream:** Choose this option to configure a Firehose stream that uses a Kinesis data stream as a data source. You can then use Amazon Data Firehose to read data easily from an existing Kinesis data stream and load it into destinations. For more information about using Kinesis Data Streams as your data source, see Writing to Amazon Data Firehose Using Kinesis Data Streams.
- **Amazon MSK:** Choose this option to configure a Firehose stream that uses Amazon MSK as a data source. You can then use Firehose to read data easily from an existing Amazon MSK clusters and load it into specified S3 buckets. For more information about using Amazon MSK as your data source, see Writing to Amazon Data Firehose Using Amazon MSK.

Firehose stream destination

The destination of your Firehose stream. Amazon Data Firehose can send data records to various destinations, including Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service, and any HTTP endpoint that is owned by you or any of your third-party service providers. The following are the supported destinations:

- Amazon OpenSearch Service
- Amazon OpenSearch Serverless
- Amazon Redshift
- Amazon S3
- Coralogix
- Datadog
- Dynatrace
- Elastic
- HTTP Endpoint
- Honeycomb
- Logic Monitor
- Logz.io
- MongoDB Cloud
- New Relic
- Splunk
- Splunk Observability Cloud
- Sumo Logic
- Snowflake

Firehose stream name

The name of your Firehose stream.

Record Transformation and Format Conversion

Configure Amazon Data Firehose to transform and convert your record data.

- If you choose Amazon MSK as the source for your delivery stream:

1. In the **Transform source records with Amazon Lambda** section, provide values for the following field:

   Data transformation

   To create a Firehose stream that doesn't transform incoming data, do not check the **Enable data transformation** checkbox.

   To specify a Lambda function for Firehose to invoke and use to transform incoming data before delivering it, check the **Enable data transformation** checkbox. You can configure a new Lambda function using one of the Lambda blueprints or choose an existing Lambda function. Your Lambda function must contain the status model that is required by Firehose. For more information, see Amazon Data Firehose Data Transformation.

2. In the **Convert record format** section, provide values for the following field:

   Record format conversion

   To create a Firehose stream that doesn't convert the format of the incoming data records, choose **Disabled**.

   To convert the format of the incoming records, choose **Enabled**, then specify the output format you want. You need to specify an Amazon Glue table that holds the schema that you want Firehose to use to convert your record format. For more information, see *Record Format Conversion*.

   For an example of how to set up record format conversion with Amazon CloudFormation, see Amazon::KinesisFirehose::DeliveryStream.

- If you choose Managed Service for Apache Flink or Direct PUT as the source for your delivery stream, in the **Source settings** section:

  1. Under **Transform records**, choose one of the following:

     a. If your destination is Amazon S3 or Splunk, in the **Decompress source records Amazon CloudWatch Logs** section, choose **Turn on decompression**.

     b. In the **Transform source records with Amazon Lambda** section, provide values for the following field:

Data transformation

To create a Firehose stream that doesn't transform incoming data, do not check the **Enable data transformation** checkbox.

To specify a Lambda function for Amazon Data Firehose to invoke and use to transform incoming data before delivering it, check the **Enable data transformation** checkbox. You can configure a new Lambda function using one of the Lambda blueprints or choose an existing Lambda function. Your Lambda function must contain the status model that is required by Amazon Data Firehose. For more information, see Amazon Data Firehose Data Transformation.

2. In the **Convert record format** section, provide values for the following field:

Record format conversion

To create a Firehose stream that doesn't convert the format of the incoming data records, choose **Disabled**.

To convert the format of the incoming records, choose **Enabled**, then specify the output format you want. You need to specify an Amazon Glue table that holds the schema that you want Amazon Data Firehose to use to convert your record format. For more information, see *Record Format Conversion*.

For an example of how to set up record format conversion with Amazon CloudFormation, see Amazon::KinesisFirehose::DeliveryStream.

Destination Settings

This topic describes the destination settings for your delivery stream. For more information on buffering hints, see Buffering hints.

Topics

- Choose Datadog for Your Destination
- Choose Honeycomb for Your Destination
- Choose Coralogix for Your Destination
- Choose Dynatrace for Your Destination
- Choose LogicMonitor for Your Destination
- Choose Logz.io for Your Destination
- Choose MongoDB Cloud for Your Destination
- Choose New Relic for Your Destination
- Choose Snowflake for Your Destination
- Choose Splunk for Your Destination
- Choose Splunk Observability Cloud for Your Destination
- Choose Sumo Logic for Your Destination
- Choose Elastic for Your Destination

Choose Amazon S3 for Your Destination

You must specify the following settings in order to use Amazon S3 as the destination for your Firehose stream.

- Enter values for the following fields.

  S3 bucket

  Choose an S3 bucket that you own where the streaming data should be delivered. You can create a new S3 bucket or choose an existing one.

  New line delimiter

  You can configure your delivery stream to add a new line delimiter between records in objects that are delivered to Amazon S3. To do so, choose **Enabled**. To not add a new line delimiter between records in objects that are delivered to Amazon S3, choose **Disabled**. If you plan to use Athena to query S3 objects with aggregated records, enable this option.

  Dynamic partitioning

  Choose **Enabled** to enable and configure dynamic partitioning.

Multi record deaggregation

This is the process of parsing through the records in the delivery stream and separating them based either on valid JSON or on the specified new line delimiter.

If you aggregate multiple events, logs, or records into a single PutRecord and PutRecordBatch API call, you can still enable and configure dynamic partitioning. With aggregated data, when you enable dynamic partitioning, Amazon Data Firehose parses the records and looks for multiple valid JSON objects within each API call. When the Firehose stream is configured with Kinesis Data Stream as a source, you can also use the built-in aggregation in the Kinesis Producer Library (KPL). Data partition functionality is executed after data is de-aggregated. Therefore, each record in each API call can be delivered to different Amazon S3 prefixes. You can also leverage the Lambda function integration to perform any other de-aggregation or any other transformation before the data partitioning functionality.

> **Important**
>
> If your data is aggregated, dynamic partitioning can be applied only after data deaggregation is performed. So if you enable dynamic partitioning to your aggregated data, you must choose **Enabled** to enable multi record deaggregation.

Firehose stream preforms the following processing steps in the following order: KPL (protobuf) de-aggregation, JSON or delimiter de-aggregation, Lambda processing, data partitioning, data format conversion, and Amazon S3 delivery.

Multi record deaggregation type

If you enabled multi record deaggregation, you must specify the method for Firehose to deaggregate your data. Use the drop-down menu to choose either **JSON** or **Delimited**.

Inline parsing

This is one of the supported mechanisms to dynamically partition your data that is bound for Amazon S3. To use inline parsing for dynamic partitioning of your data, you must specify data record parameters to be used as partitioning keys and provide a value for each specified partitioning key. Choose **Enabled** to enable and configure inline parsing.

> **Important**
>
> If you specified an Amazon Lambda function in the steps above for transforming your source records, you can use this function to dynamically partition your data that is bound to S3 and you can still create your partitioning keys with inline parsing. With dynamic partitioning, you can use either inline parsing or your Amazon Lambda function to create your partitioning keys. Or you can use both inline parsing and your Amazon Lambda function at the same time to create your partitioning keys.

Dynamic partitioning keys

You can use the **Key** and **Value** fields to specify the data record parameters to be used as dynamic partitioning keys and jq queries to generate dynamic partitioning key values. Firehose supports jq 1.6 only. You can specify up to 50 dynamic partitioning keys. You must enter valid jq expressions for your dynamic partitioning key values in order to successfully configure dynamic partitioning for your Firehose stream.

S3 bucket prefix

When you enable and configure dynamic partitioning, you must specify the S3 bucket prefixes to which Amazon Data Firehose is to deliver partitioned data.

In order for dynamic partitioning to be configured correctly, the number of the S3 bucket prefixes must be identical to the number of the specified partitioning keys.

You can partition your source data with inline parsing or with your specified Amazon Lambda function. If you specified an Amazon Lambda function to create partitioning keys for your source data, you must manually type in the S3 bucket prefix value(s) using the following format: "partitionKeyFromLambda:keyID". If you are using inline parsing to specify the partitioning keys for your source data, you can either manually type in the S3 bucket preview values using the following format: "partitionKeyFromQuery:keyID" or you can choose the **Apply dynamic partitioning keys** button to use your dynamic partitioning key/value pairs to auto-generate your S3 bucket prefixes. While partitioning your data with either inline parsing or Amazon Lambda, you can also use the following expression forms in your S3 bucket prefix: !{namespace:value}, where namespace can be either partitionKeyFromQuery or partitionKeyFromLambda.

S3 bucket and S3 error output prefix time zone

Choose a time zone that you want to use for date and time in Custom Prefixes for Amazon Simple Storage Service Objects. By default, Firehose adds a time prefix in UTC. You can change the time zone used in S3 prefixes if you want to use different time zone.

Buffering hints

Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

S3 compression

Choose GZIP, Snappy, Zip, or Hadoop-Compatible Snappy data compression, or no data compression. Snappy, Zip, and Hadoop-Compatible Snappy compression is not available for delivery streams with Amazon Redshift as the destination.

S3 file extension format (optional)

Specify a file extension format for objects delivered to Amazon S3 destination bucket. If you enable this feature, specified file extension will override default file extensions appended by Data Format Conversion or S3 compression features such as .parquet or .gz. Make sure if you configured the right file extension when you use this feature with Data Format Conversion or S3 compression. File extension must start with a period (.) and can contain allowed characters: 0-9a-z!-_.*'(). File extension cannot exceed 128 characters.

S3 encryption

Firehose supports Amazon S3 server-side encryption with Amazon Key Management Service (SSE-KMS) for encrypting delivered data in Amazon S3. You can choose to use the default encryption type specified in the destination S3 bucket or to encrypt with a key from the list of Amazon KMS keys that you own. If you encrypt the data with Amazon KMS keys, you can use either the default Amazon managed key (aws/s3) or a customer managed key. For more information, see Protecting Data Using Server-Side Encryption with Amazon KMS-Managed Keys (SSE-KMS).

Choose Amazon Redshift for Your Destination

This section describes settings for using Amazon Redshift as your Firehose stream destination.

Choose either of the following procedures based on whether you have an Amazon Redshift provisioned cluster or an Amazon Redshift Serverless workgroup.

- Amazon Redshift Provisioned Cluster
- Amazon Redshift Serverless Workgroup

Amazon Redshift Provisioned Cluster

This section describes settings for using Amazon Redshift provisioned cluster as your Firehose stream destination.

- Enter values for the following fields:

  Cluster

  The Amazon Redshift cluster to which S3 bucket data is copied. Configure the Amazon Redshift cluster to be publicly accessible and unblock Amazon Data Firehose IP addresses. For more information, see Grant Amazon Data Firehose Access to an Amazon Redshift Destination .

  User name

  An Amazon Redshift user with permissions to access the Amazon Redshift cluster. This user must have the Amazon Redshift `INSERT` permission for copying data from the S3 bucket to the Amazon Redshift cluster.

  Password

  The password for the user who has permissions to access the cluster.

  Database

  The Amazon Redshift database to where the data is copied.

  Table

  The Amazon Redshift table to where the data is copied.

  Columns

  (Optional) The specific columns of the table to which the data is copied. Use this option if the number of columns defined in your Amazon S3 objects is less than the number of columns within the Amazon Redshift table.

Intermediate S3 destination

Firehose delivers your data to your S3 bucket first and then issues an Amazon Redshift **COPY** command to load the data into your Amazon Redshift cluster. Specify an S3 bucket that you own where the streaming data should be delivered. Create a new S3 bucket, or choose an existing bucket that you own.

Firehose doesn't delete the data from your S3 bucket after loading it to your Amazon Redshift cluster. You can manage the data in your S3 bucket using a lifecycle configuration. For more information, see Object Lifecycle Management in the *Amazon Simple Storage Service User Guide*.

Intermediate S3 prefix

(Optional) To use the default prefix for Amazon S3 objects, leave this option blank. Firehose automatically uses a prefix in "YYYY/MM/dd/HH" UTC time format for delivered Amazon S3 objects. You can add to the start of this prefix. For more information, see Amazon S3 Object Name Format.

COPY options

Parameters that you can specify in the Amazon Redshift **COPY** command. These might be required for your configuration. For example, "GZIP" is required if Amazon S3 data compression is enabled. "REGION" is required if your S3 bucket isn't in the same Amazon Region as your Amazon Redshift cluster. For more information, see COPY in the *Amazon Redshift Database Developer Guide*.

COPY command

The Amazon Redshift **COPY** command. For more information, see COPY in the *Amazon Redshift Database Developer Guide*.

Retry duration

Time duration (0–7200 seconds) for Firehose to retry if data **COPY** to your Amazon Redshift cluster fails. Firehose retries every 5 minutes until the retry duration ends. If you set the retry duration to 0 (zero) seconds, Firehose does not retry upon a **COPY** command failure.

Buffering hints

Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

S3 compression

Choose GZIP, Snappy, Zip, or Hadoop-Compatible Snappy data compression, or no data compression. Snappy, Zip, and Hadoop-Compatible Snappy compression is not available for delivery streams with Amazon Redshift as the destination.

S3 file extension format (optional)

S3 file extension format (optional) – Specify a file extension format for objects delivered to Amazon S3 destination bucket. If you enable this feature, specified file extension will override default file extensions appended by Data Format Conversion or S3 compression features such as .parquet or .gz. Make sure if you configured the right file extension when you use this feature with Data Format Conversion or S3 compression. File extension must start with a period (.) and can contain allowed characters: 0-9a-z!-_.*'(). File extension cannot exceed 128 characters.

S3 encryption

Firehose supports Amazon S3 server-side encryption with Amazon Key Management Service (SSE-KMS) for encrypting delivered data in Amazon S3. You can choose to use the default encryption type specified in the destination S3 bucket or to encrypt with a key from the list of Amazon KMS keys that you own. If you encrypt the data with Amazon KMS keys, you can use either the default Amazon managed key (aws/s3) or a customer managed key. For more information, see Protecting Data Using Server-Side Encryption with Amazon KMS-Managed Keys (SSE-KMS).

Amazon Redshift Serverless Workgroup

This section describes settings for using Amazon Redshift Serverless workgroup as your Firehose stream destination.

- Enter values for the following fields:

Workgroup name

The Amazon Redshift Serverless workgroup to which S3 bucket data is copied. Configure the Amazon Redshift Serverless workgroup to be publicly accessible and unblock the Firehose IP addresses. For more information, see the Connect to a publicly accessible

Amazon Redshift Serverless instance section in Connecting to Amazon Redshift Serverless and also Grant Amazon Data Firehose Access to an Amazon Redshift Destination .

User name

An Amazon Redshift user with permissions to access the Amazon Redshift Serverless workgroup. This user must have the Amazon Redshift `INSERT` permission for copying data from the S3 bucket to the Amazon Redshift Serverless workgroup.

Password

The password for the user who has permissions to access the Amazon Redshift Serverless workgroup.

Database

The Amazon Redshift database to where the data is copied.

Table

The Amazon Redshift table to where the data is copied.

Columns

(Optional) The specific columns of the table to which the data is copied. Use this option if the number of columns defined in your Amazon S3 objects is less than the number of columns within the Amazon Redshift table.

Intermediate S3 destination

Amazon Data Firehose delivers your data to your S3 bucket first and then issues an Amazon Redshift **COPY** command to load the data into your Amazon Redshift Serverless workgroup. Specify an S3 bucket that you own where the streaming data should be delivered. Create a new S3 bucket, or choose an existing bucket that you own.

Firehose doesn't delete the data from your S3 bucket after loading it to your Amazon Redshift Serverless workgroup. You can manage the data in your S3 bucket using a lifecycle configuration. For more information, see Object Lifecycle Management in the *Amazon Simple Storage Service User Guide*.

Intermediate S3 prefix

(Optional) To use the default prefix for Amazon S3 objects, leave this option blank. Firehose automatically uses a prefix in "`YYYY/MM/dd/HH`" UTC time format for delivered Amazon

S3 objects. You can add to the start of this prefix. For more information, see Amazon S3 Object Name Format.

COPY options

Parameters that you can specify in the Amazon Redshift **COPY** command. These might be required for your configuration. For example, "`GZIP`" is required if Amazon S3 data compression is enabled. "`REGION`" is required if your S3 bucket isn't in the same Amazon Region as your Amazon Redshift Serverless workgroup. For more information, see COPY in the *Amazon Redshift Database Developer Guide*.

COPY command

The Amazon Redshift **COPY** command. For more information, see COPY in the *Amazon Redshift Database Developer Guide*.

Retry duration

Time duration (0–7200 seconds) for Firehose to retry if data **COPY** to your Amazon Redshift Serverless workgroup fails. Firehose retries every 5 minutes until the retry duration ends. If you set the retry duration to 0 (zero) seconds, Firehose does not retry upon a **COPY** command failure.

Buffering hints

Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

S3 compression

Choose GZIP, Snappy, Zip, or Hadoop-Compatible Snappy data compression, or no data compression. Snappy, Zip, and Hadoop-Compatible Snappy compression is not available for delivery streams with Amazon Redshift as the destination.

S3 file extension format (optional)

S3 file extension format (optional) – Specify a file extension format for objects delivered to Amazon S3 destination bucket. If you enable this feature, specified file extension will override default file extensions appended by Data Format Conversion or S3 compression features such as .parquet or .gz. Make sure if you configured the right file extension when you use this feature with Data Format Conversion or S3 compression. File extension must

start with a period (.) and can contain allowed characters: 0-9a-z!-_.*'(). File extension cannot exceed 128 characters.

S3 encryption

Firehose supports Amazon S3 server-side encryption with Amazon Key Management Service (SSE-KMS) for encrypting delivered data in Amazon S3. You can choose to use the default encryption type specified in the destination S3 bucket or to encrypt with a key from the list of Amazon KMS keys that you own. If you encrypt the data with Amazon KMS keys, you can use either the default Amazon managed key (aws/s3) or a customer managed key. For more information, see Protecting Data Using Server-Side Encryption with Amazon KMS-Managed Keys (SSE-KMS).

Choose OpenSearch Service for Your Destination

This section describes options for using OpenSearch Service for your destination.

- Enter values for the following fields:

OpenSearch Service domain

The OpenSearch Service domain to which your data is delivered.

Index

The OpenSearch Service index name to be used when indexing data to your OpenSearch Service cluster.

Index rotation

Choose whether and how often the OpenSearch Service index should be rotated. If index rotation is enabled, Amazon Data Firehose appends the corresponding timestamp to the specified index name and rotates. For more information, see Index Rotation for the OpenSearch Service Destination.

Type

The OpenSearch Service type name to be used when indexing data to your OpenSearch Service cluster. For Elasticsearch 7.x and OpenSearch 1.x, there can be only one type per index. If you try to specify a new type for an existing index that already has another type, Firehose returns an error during runtime.

For Elasticsearch 7.x, leave this field empty.

Retry duration

Time duration (0–7200 seconds) for Firehose to retry if an index request to your OpenSearch Service cluster fails. Firehose retries every 5 minutes until the retry duration ends. If you set the retry duration to 0 (zero) seconds, Firehose does not retry upon an index request failure.

DocumentID type

Indicates the method for setting up document ID. The supported methods are Firehose-generated document ID and OpenSearch Service-generated document ID. Firehose-generated document ID is the default option when the document ID value is not set. OpenSearch Service-generated document ID is the recommended option because it supports write-heavy operations, including log analytics and observability, consuming fewer CPU resources at the OpenSearch Service domain and thus, resulting in improved performance.

Destination VPC connectivity

If your OpenSearch Service domain is in a private VPC, use this section to specify that VPC. Also specify the subnets and subgroups that you want Amazon Data Firehose to use when it sends data to your OpenSearch Service domain. You can use the same security groups that the OpenSearch Service domain is using. If you specify different security groups, ensure that they allow outbound HTTPS traffic to the OpenSearch Service domain's security group. Also ensure that the OpenSearch Service domain's security group allows HTTPS traffic from the security groups that you specified when you configured your Firehose stream. If you use the same security group for both your Firehose stream and the OpenSearch Service domain, make sure the security group's inbound rule allows HTTPS traffic. For more information about security group rules, see Security group rules in the Amazon VPC documentation.

Important

When you specify subnets for delivering data to the destination in a private VPC, make sure you have enough number of free IP addresses in chosen subnets. If there is no available free IP address in a specified subnet, Firehose cannot create or add

ENIs for the data delivery in the private VPC, and the delivery will be degraded or fail.

Buffer hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose OpenSearch Serverless for Your Destination

This section describes options for using OpenSearch Serverless for your destination.

- Enter values for the following fields:

OpenSearch Serverless collection

The endpoint for a group of OpenSearch Serverless indexes to which your data is delivered.

Index

The OpenSearch Serverless index name to be used when indexing data to your OpenSearch Serverless collection.

Destination VPC connectivity

If your OpenSearch Serverless collection is in a private VPC, use this section to specify that VPC. Also specify the subnets and subgroups that you want Amazon Data Firehose to use when it sends data to your OpenSearch Serverless collection.

> **Important**
>
> When you specify subnets for delivering data to the destination in a private VPC, make sure you have enough number of free IP addresses in chosen subnets. If there is no available free IP address in a specified subnet, Firehose cannot create or add ENIs for the data delivery in the private VPC, and the delivery will be degraded or fail.

Retry duration

Time duration (0–7200 seconds) for Amazon Data Firehose to retry if an index request to your OpenSearch Serverless collection fails. Amazon Data Firehose retries every 5 minutes until the retry duration ends. If you set the retry duration to 0 (zero) seconds, Amazon Data Firehose does not retry upon an index request failure.

Buffer hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose HTTP Endpoint for Your Destination

This section describes options for using **HTTP endpoint** for your destination.

> **Important**
>
> If you choose an HTTP endpoint as your destination, review and follow the instructions in Appendix - HTTP Endpoint Delivery Request and Response Specifications.

- Provide values for the following fields:

  HTTP endpoint name - optional

  Specify a user friendly name for the HTTP endpoint. For example, `My HTTP Endpoint Destination`.

  HTTP endpoint URL

  Specify the URL for the HTTP endpoint in the following format: `https://xyz.httpendpoint.com`. The URL must be an HTTPS URL.

  Access key - optional

  Contact the endpoint owner to obtain the access key (if it is required) to enable data delivery to their endpoint from Firehose.

Content encoding

Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

Retry duration

Specify how long Amazon Data Firehose retries sending data to the selected HTTP endpoint.

After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn’t arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

> **Important**
>
> For the HTTP endpoint destinations, if you are seeing 413 response codes from the destination endpoint in CloudWatch Logs, lower the buffering hint size on your Firehose stream and try again.

Choose Datadog for Your Destination

This section describes options for using **Datadog** for your destination. For more information about Datadog, see https://docs.datadoghq.com/integrations/amazon_web_services/.

- Provide values for the following fields:

  HTTP endpoint URL

  Choose the HTTP endpoint URL from the following options in the drop down menu:

  - **Datadog logs - US1**
  - **Datadog logs - US5**
  - **Datadog logs - EU**
  - **Datadog logs - GOV**
  - **Datadog metrics - US**
  - **Datadog metrics - EU**

  API key

  Contact Datadog to obtain the API key required to enable data delivery to this endpoint from Amazon Data Firehose.

  Content encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

Retry duration

Specify how long Amazon Data Firehose retries sending data to the selected HTTP endpoint.

After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn’t arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose Honeycomb for Your Destination

This section describes options for using **Honeycomb** for your destination. For more information about Honeycomb, see https://docs.honeycomb.io/getting-data-in/metrics/aws-cloudwatch-metrics/ .

- Provide values for the following fields:

  Honeycomb Kinesis endpoint

  Specify the URL for the HTTP endpoint in the following format: https://api.honeycomb.io/1/kinesis_events/{{dataset}}

  API key

  Contact Honeycomb to obtain the API key required to enable data delivery to this endpoint from Amazon Data Firehose.

  Content encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** to enable content encoding of your request. This is the recommended option for the Honeycomb destination.

  Retry duration

  Specify how long Amazon Data Firehose retries sending data to the selected HTTP endpoint.

  After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn’t arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

  Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

  Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

  If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose Coralogix for Your Destination

This section describes options for using **Coralogix** for your destination. For more information about Coralogix, see https://coralogix.com/integrations/aws-firehose .

- Provide values for the following fields:

  HTTP endpoint URL

  Choose the HTTP endpoint URL from the following options in the drop down menu:

  - **Coralogix - US**
  - **Coralogix - SINGAPORE**
  - **Coralogix - IRELAND**
  - **Coralogix - INDIA**
  - **Coralogix - STOCKHOLM**

  Private key

  Contact Coralogix to obtain the private key required to enable data delivery to this endpoint from Amazon Data Firehose.

  Content encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** to enable content encoding of your request. This is the recommended option for the Coralogix destination.

Retry duration

Specify how long Amazon Data Firehose retries sending data to the selected HTTP endpoint.

After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn’t arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

- applicationName: the environment where you are running Data Firehose
- subsystemName: the name of the Data Firehose integration
- computerName: the name of the Firehose stream in use

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose Dynatrace for Your Destination

This section describes options for using **Dynatrace** for your destination. For more information, see https://www.dynatrace.com/support/help/technology-support/cloud-platforms/amazon-web-services/integrations/cloudwatch-metric-streams/.

- Provide values for the following fields:

  HTTP endpoint URL

  Choose the HTTP endpoint URL (**Dynatrace US**, **Dynatrace EU**, or **Dynatrace Global**) from the drop down menu.

  API token

  Generate the Dynatrace API token required for data delivery from Amazon Data Firehose. For more information, see https://www.dynatrace.com/support/help/dynatrace-api/basics/dynatrace-api-authentication/.

  API URL

  Provide the API URL of your Dynatrace environment.

  Content encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

  Retry duration

  Specify how long Amazon Data Firehose retries sending data to the selected HTTP endpoint.

  After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn't arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose LogicMonitor for Your Destination

This section describes options for using **LogicMonitor** for your destination. For more information, see https://www.logicmonitor.com.

- Provide values for the following fields:

  HTTP endpoint URL

  Specify the URL for the HTTP endpoint in the following format: https://ACCOUNT.logicmonitor.com

  API key

  Contact LogicMonitor to obtain the API key required to enable data delivery to this endpoint from Amazon Data Firehose.

Content encoding

Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

Retry duration

Specify how long Amazon Data Firehose retries sending data to the selected HTTP endpoint.

After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn’t arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose Logz.io for Your Destination

This section describes options for using **Logz.io** for your destination. For more information, see https://logz.io/.

> **Note**
>
> In the Europe (Milan) region, Logz.io is not supported as an Amazon Data Firehose destination.

- Provide values for the following fields:

  HTTP endpoint URL

  Specify the URL for the HTTP endpoint in the following format: `https://listener-aws-metrics-stream-<region>.logz.io/`. For example, `https://listener-aws-metrics-stream-us.logz.io/`. The URL must be an HTTPS URL.

  Content encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

  Retry duration

  Specify how long Amazon Data Firehose retries sending data to Logz.io.

  After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn't arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

  Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose MongoDB Cloud for Your Destination

This section describes options for using **MongoDB Cloud** for your destination. For more information, see https://www.mongodb.com.

- Provide values for the following fields:

  MongoDB Realm webhook URL

  Specify the URL for the HTTP endpoint in the following format: `https://webhooks.mongodb-realm.com`. The URL must be an HTTPS URL.

  API key

  Contact MongoDB Cloud to obtain the API key required to enable data delivery to this endpoint from Amazon Data Firehose.

  Content encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

Retry duration

Specify how long Amazon Data Firehose retries sending data to the selected third-party provider.

After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn't arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Choose New Relic for Your Destination

This section describes options for using **New Relic** for your destination. For more information, see https://newrelic.com.

- Provide values for the following fields:

HTTP endpoint URL

Choose the HTTP endpoint URL from the following options in the drop down menu:

- **New Relic logs - US**
- **New Relic metrics - US**
- **New Relic metrics - EU**

API key

Enter your License Key (40-characters hexadecimal string) from your New Relic One Account settings. This API key is required to enable data delivery to this endpoint from Firehose.

Content encoding

Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

Retry duration

Specify how long Amazon Data Firehose retries sending data to the New Relic HTTP endpoint.

After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn’t arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose Snowflake for Your Destination

This section describes options for using Snowflake for your destination.

Connection settings

- Provide values for the following fields:

  Snowflake account URL

  Specify a regional account URL provided by Snowflake. Firehose does not support a region-less URL in preview when private link is enabled. Refer to Snowflake documentation on how to determine your account URL. For example: `xy12345.us-east-1.aws.snowflakecomputing.com`. Note that port number must not be specified, whereas protocol (https://) is optional.

  User login

  Specify the Snowflake user to be used for loading data. Make sure the user has access to insert data into the Snowflake table.

  Private key

  Specify the user's private key of the key pair used for authentication with Snowflake. Make sure the private key is in PKCS8 format. Do not include PEM header and footer as part of Private Key. If the key is split across multiple lines, remove the line breaks.

Passphrase

Passphrase to decrypt the private key when the key is encrypted. Leave this field empty if private key is not encrypted. For information, see Using Key Pair Authentication & Key Rotation.

Role configuration

Use default Snowflake role – If this option is selected, Firehose will not pass any role to Snowflake. Default role is assumed to load data. Please make sure the default role has permission to insert data in to Snowflake table.

Use custom Snowflake role – Enter a non-default Snowflake role to be assumed by Firehose when loading data into Snowflake table.

Snowflake connectivity

Options are **Private** or **Public**.

Private VPCE ID (optional)

The VPCE ID for Firehose to privately connect with Snowflake. The ID format is com.amazonaws.vpce.[region].vpce-svc-*[id]*. For more information, see Amazon PrivateLink & Snowflake.

> **Note**
>
> During the public preview, make sure that your Snowflake network permits access to Firehose. To gain access, either contact Amazon Web Services Support to add a Firehose VPC endpoint to your allow list, or consider disabling the network policy on your Snowflake cluster.

Database configuration

- You must specify the following settings in order to use Snowflake as the destination for your Firehose delivery stream:
  - Snowflake database – All data in Snowflake is maintained in databases.
  - Snowflake schema – Each database consists of one or more schemas, which are logical groupings of database objects, such as tables and views

- Snowflake table – All data in Snowflake is stored in database tables, logically structured as collections of columns and rows.

Data loading options for your Snowflake table

- Use JSON keys as column names
- Use VARIANT columns
  - Content column name – Specify a column name in the table, where the raw data has to be loaded.
  - Metadata column name (optional) – Specify a column name in the table, where the metadata information has to be loaded.

Choose Splunk for Your Destination

This section describes options for using Splunk for your destination.

> **Note**
>
> Firehose delivers data to Splunk clusters configured with Classic Load Balancer or an Application Load Balancer.

- Provide values for the following fields:

  Splunk cluster endpoint

  To determine the endpoint, see Configure Amazon Data Firehose to Send Data to the Splunk Platform in the Splunk documentation.

  Splunk endpoint type

  Choose `Raw endpoint` in most cases. Choose `Event endpoint` if you preprocessed your data using Amazon Lambda to send data to different indexes by event type. For information about what endpoint to use, see Configure Amazon Data Firehose to send data to the Splunk platform in the Splunk documentation.

Authentication token

To set up a Splunk endpoint that can receive data from Amazon Data Firehose, see Installation and configuration overview for the Splunk Add-on for Amazon Data Firehose in the Splunk documentation. Save the token that you get from Splunk when you set up the endpoint for this Firehose stream, and add it here.

HEC acknowledgement timeout

Specify how long Amazon Data Firehose waits for the index acknowledgement from Splunk. If Splunk doesn't send the acknowledgment before the timeout is reached, Amazon Data Firehose considers it a data delivery failure. Amazon Data Firehose then either retries or backs up the data to your Amazon S3 bucket, depending on the retry duration value that you set.

Retry duration

Specify how long Amazon Data Firehose retries sending data to Splunk.

After sending data, Amazon Data Firehose first waits for an acknowledgment from Splunk. If an error occurs or the acknowledgment doesn't arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to Splunk (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from Splunk.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Choose Splunk Observability Cloud for Your Destination

This section describes options for using **Splunk Observability Cloud** for your destination. For more information, see https://docs.splunk.com/observability/en/gdi/get-data-in/connect/aws/aws-apiconfig.html#connect-to-aws-using-the-splunk-observability-cloud-api.

- Provide values for the following fields:

  Cloud Ingest Endpoint URL

  You can find your Splunk Observability Cloud's Real-time Data Ingest URL in Profile > Organizations > Real-time Data Ingest Endpoint in Splunk Observability console.

  Access Token

  Copy your Splunk Observability access token with INGEST authorization scope from Settings > Access Tokens in Splunk Observability console

  Content Encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

  Retry duration

  Specify how long Amazon Data Firehose retries sending data to the selected HTTP endpoint.

  After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn't arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

  Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

  Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the

acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the destination varies from service provider to service provider.

Choose Sumo Logic for Your Destination

This section describes options for using **Sumo Logic** for your destination. For more information, see https://www.sumologic.com.

- Provide values for the following fields:

  HTTP endpoint URL

  Specify the URL for the HTTP endpoint in the following format: `https://deployment name.sumologic.net/receiver/v1/kinesis/dataType/access token`. The URL must be an HTTPS URL.

  Content encoding

  Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** or **Disabled** to enable/disable content encoding of your request.

  Retry duration

  Specify how long Amazon Data Firehose retries sending data to Sumo Logic.

  After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn't arrive within

the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the Elastic destination varies from service provider to service provider.

Choose Elastic for Your Destination

This section describes options for using **Elastic** for your destination.

- Provide values for the following fields:

  Elastic endpoint URL

  Specify the URL for the HTTP endpoint in the following format: `https://<cluster-id>.es.<region>.aws.elastic-cloud.com`. The URL must be an HTTPS URL.

API key

Contact Elastic service to obtain the API key required to enable data delivery to their service from Amazon Data Firehose.

Content encoding

Amazon Data Firehose uses content encoding to compress the body of a request before sending it to the destination. Choose **GZIP** (which is what selected by default) or **Disabled** to enable/disable content encoding of your request.

Retry duration

Specify how long Amazon Data Firehose retries sending data to Elastic.

After sending data, Amazon Data Firehose first waits for an acknowledgment from the HTTP endpoint. If an error occurs or the acknowledgment doesn't arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time that Amazon Data Firehose sends data to the HTTP endpoint (either the initial attempt or a retry), it restarts the acknowledgement timeout counter and waits for an acknowledgement from the HTTP endpoint.

Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout period is reached. If the acknowledgment times out, Amazon Data Firehose determines whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

If you don't want Amazon Data Firehose to retry sending data, set this value to 0.

Parameters - optional

Amazon Data Firehose includes these key-value pairs in each HTTP call. These parameters can help you identify and organize your destinations.

Buffering hints

Amazon Data Firehose buffers incoming data before delivering it to the specified destination. The recommended buffer size for the Elastic destination is 1 MiB.

Backup and Advanced Settings

This topic describes how to configure the backup and the advanced settings for your Firehose stream.

Backup Settings

Amazon Data Firehose uses Amazon S3 to backup all or failed only data that it attempts to deliver to your chosen destination.

> **Important**
>
> Backup settings are only supported if the source for your Firehose stream is Direct PUT or Kinesis Data Streams.

You can specify the S3 backup settings for your Firehose stream if you made one of the following choices:

- If you set Amazon S3 as the destination for your Amazon Data Firehose Firehose stream and you choose to specify an Amazon Lambda function to transform data records or if you choose to convert data record formats for your delivery stream.
- If you set Amazon Redshift as the destination for your Amazon Data Firehose Firehose stream and you choose to specify an Amazon Lambda function to transform data records.
- If you set any of the following services as the destination for your Firehose Firehose stream: Amazon OpenSearch Service, Datadog, Dynatrace, HTTP Endpoint, LogicMonitor, MongoDB Cloud, New Relic, Splunk, or Sumo Logic.

The following are the backup settings for your Amazon Data Firehose delivery stream:

- Source record backup in Amazon S3 - if S3 or Amazon Redshift is your selected destination, this setting indicates whether you want to enable source data backup or keep it disabled. If any other supported service (other than S3 or Amazon Redshift) is set as your selected destination, then this setting indicates if you want to backup all your source data or failed data only.
- S3 backup bucket - this is the S3 bucket where Amazon Data Firehose backs up your data.
- S3 backup bucket prefix - this is the prefix where Amazon Data Firehose backs up your data.

- S3 backup bucket error output prefix - all failed data is backed up in the this S3 bucket error output prefix.
- Buffering hints, compression and encryption for backup - Amazon Data Firehose uses Amazon S3 to backup all or failed only data that it attempts to deliver to your chosen destination. Amazon Data Firehose buffers incoming data before delivering it (backing it up) to Amazon S3. You can choose a buffer size of 1–128 MiBs and a buffer interval of 60–900 seconds. The condition that is satisfied first triggers data delivery to Amazon S3. If you enable data transformation, the buffer interval applies from the time transformed data is received by Amazon Data Firehose to the data delivery to Amazon S3. If data delivery to the destination falls behind data writing to the Firehose stream, Amazon Data Firehose raises the buffer size dynamically to catch up. This action helps ensure that all data is delivered to the destination.
- S3 compression - choose GZIP, Snappy, Zip, or Hadoop-Compatible Snappy data compression, or no data compression. Snappy, Zip, and Hadoop-Compatible Snappy compression is not available for delivery streams with Amazon Redshift as the destination.
- S3 file extension format (optional) – Specify a file extension format for objects delivered to Amazon S3 destination bucket. If you enable this feature, specified file extension will override default file extensions appended by Data Format Conversion or S3 compression features such as .parquet or .gz. Make sure if you configured the right file extension when you use this feature with Data Format Conversion or S3 compression. File extension must start with a period (.) and can contain allowed characters: 0-9a-z!-_.*‘(). File extension cannot exceed 128 characters.
- Firehose supports Amazon S3 server-side encryption with Amazon Key Management Service (SSE-KMS) for encrypting delivered data in Amazon S3. You can choose to use the default encryption type specified in the destination S3 bucket or to encrypt with a key from the list of Amazon KMS keys that you own. If you encrypt the data with Amazon KMS keys, you can use either the default Amazon managed key (aws/s3) or a customer managed key. For more information, see Protecting Data Using Server-Side Encryption with Amazon KMS-Managed Keys (SSE-KMS).

Advanced Settings

The following are the advanced settings for your Amazon Data Firehose delivery stream:

- Server-side encryption - Amazon Data Firehose supports Amazon S3 server-side encryption with Amazon Key Management Service (Amazon KMS) for encrypting delivered data in Amazon S3. For more information, see Protecting Data Using Server-Side Encryption with Amazon KMS–Managed Keys (SSE-KMS).

- Error logging - Amazon Data Firehose logs errors related to processing and delivery. Additionally, when data transformation is enabled, it can log Lambda invocations and send data delivery errors to CloudWatch Logs. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Logs.

  > **Important**
  >
  > While optional, enabling Amazon Data Firehose error logging during Firehose stream creation is strongly recommended. This practice ensures that you can access error details in case of record processing or delivery failures.

- Permissions - Amazon Data Firehose uses IAM roles for all the permissions that the Firehose stream needs. You can choose to create a new role where required permissions are assigned automatically, or choose an existing role created for Amazon Data Firehose. The role is used to grant Firehose access to various services, including your S3 bucket, Amazon KMS key (if data encryption is enabled), and Lambda function (if data transformation is enabled). The console might create a role with placeholders. For more information, see What is IAM?.
- Tags - You can add tags to organize your Amazon resources, track costs, and control access.

  If you specify tags in the `CreateDeliveryStream` action, Amazon Data Firehose performs an additional authorization on the `firehose:TagDeliveryStream` action to verify if users have permissions to create tags. If you do not provide this permission, requests to create new Firehose delivery streams with IAM resource tags will fail with an `AccessDeniedException` such as following.

  ```
  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.
  ```

  The following example demonstrates a policy that allows users to create a delivery stream and apply tags.

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

```
            "Resource": "*",
            }
        },
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            }
        }
    ]
}
```

Once you've chosen your backup and advanced settings, review your choices, and then choose **Create Firehose stream**.

The new Firehose stream takes a few moments in the **Creating** state before it is available. After your Firehose stream is in an **Active** state, you can start sending data to it from your producer.

Buffering hints

Amazon Data Firehose buffers incoming streaming data in memory to a certain size (buffering size) and for a certain period of time (buffering interval) before delivering it to the specified destinations. You would use buffering hints when you want to deliver optimal sized files to Amazon S3 and get better performance from data processing applications or to adjust Firehose delivery rate to match destination speed.

You can configure the buffering size and the buffer interval while creating new delivery streams or update the buffering size and the buffering interval on your existing delivery streams. Buffering size is measured in MBs and buffering interval is measured in seconds. However, if you specify a value for one of them, you must also provide a value for the other. The first buffer condition that is satisfied triggers Firehose to deliver the data. If you don't configure the buffering values, then the default values are used.

You can configure Firehose buffering hints through the Amazon Web Services Management Console, Amazon Command Line Interface, or Amazon SDKs. For existing streams, you can reconfigure buffering hints with a value that suits your use cases using the **Edit** option in the console or using the UpdateDestination API. For new streams, you can configure buffering hints as part of new stream creation using the console or using the CreateDeliveryStream API. To

adjust the buffering size, set `SizeInMBs` and `IntervalInSeconds` in the destination specific `DestinationConfiguration` parameter of the CreateDeliveryStream or UpdateDestination API.

> **Note**
>
> To meet lower latencies of real-time use cases, you can use zero buffering interval hint. When you configure buffering interval as zero seconds, Firehose will not buffer data and will deliver data within a few seconds. Before you change buffering hints to a lower value, check with the vendor for recommended buffering hints of Firehose for their destinations.

> **Note**
>
> Firehose uses multi-part upload for S3 destination when you configure a buffer time interval less than 60 seconds to offer lower latencies. Due to multi-part upload for S3 destination, you will see some increase in S3 PUT API costs if you choose a buffer time interval less than 60 seconds.

For destination specific buffering hint ranges and default values, see the following table:

Destination	Buffering size in MB (default in parenthesis)	Buffering interval in seconds (default in parenthesis)
S3	1-128 (5)	0-900 (300)
Redshift	1-128 (5)	0-900 (300)
OpenSearch Serverless	1-100 (5)	0-900 (300)
OpenSearch	1-100 (5)	0-900 (300)
Splunk	1-5(5)	0-60 (60)
Datadog	1-4 (4)	0-900 (60)

Destination	Buffering size in MB (default in parenthesis)	Buffering interval in seconds (default in parenthesis)
Coralogix	1-64 (6)	0-900 (60)
Dynatrace	1-64 (5)	0-900 (60)
Elastic	1	0-900 (60)
Honeycomb	1-64 (15)	0-900 (60)
HTTP endpoint	1-64 (5)	0-900 (60)
LogicMonitor	1-64 (5)	0-900 (60)
Logzio	1-64 (5)	0-900 (60)
mongoDB	1-16 (5)	0-900 (60)
newRelic	1-64 (5)	0-900 (60)
sumoLogic	1-64 (1)	0-900 (60)
Splunk Observability Cloud	1-64(1)	0-900 (60)

Testing Your Firehose stream Using Sample Data

You can use the Amazon Web Services Management Console to ingest simulated stock ticker data. The console runs a script in your browser to put sample records in your Firehose stream. This enables you to test the configuration of your Firehose stream without having to generate your own test data.

The following is an example from the simulated data:

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

Note that standard Amazon Data Firehose charges apply when your Firehose stream transmits the data, but there is no charge when the data is generated. To stop incurring these charges, you can stop the sample stream from the console at any time.

Contents

Prerequisites

Before you begin, create a Firehose stream. For more information, see Creating a Firehose stream.

Test Using Amazon S3 as the Destination

Use the following procedure to test your Firehose stream using Amazon Simple Storage Service (Amazon S3) as the destination.

To test a Firehose stream using Amazon S3

1. Open the Firehose console at https://console.amazonaws.cn/firehose/.
2. Choose the Firehose stream.

3. Under **Test with demo data**, choose **Start sending demo data** to generate sample stock ticker data.
4. Follow the onscreen instructions to verify that data is being delivered to your S3 bucket. Note that it might take a few minutes for new objects to appear in your bucket, based on the buffering configuration of your bucket.
5. When the test is complete, choose **Stop sending demo data** to stop incurring usage charges.

Test Using Amazon Redshift as the Destination

Use the following procedure to test your Firehose stream using Amazon Redshift as the destination.

To test a Firehose stream using Amazon Redshift

1. Your Firehose stream expects a table to be present in your Amazon Redshift cluster. Connect to Amazon Redshift through a SQL interface and run the following statement to create a table that accepts the sample data.

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

2. Open the Firehose console at https://console.amazonaws.cn/firehose/.
3. Choose the Firehose stream.
4. Edit the destination details for your Firehose stream to point to the newly created `firehose_test_table` table.
5. Under **Test with demo data**, choose **Start sending demo data** to generate sample stock ticker data.
6. Follow the onscreen instructions to verify that data is being delivered to your table. Note that it might take a few minutes for new rows to appear in your table, based on the buffering configuration.
7. When the test is complete, choose **Stop sending demo data** to stop incurring usage charges.
8. Edit the destination details for your Firehose stream to point to another table.

9. (Optional) Delete the `firehose_test_table` table.

Test Using OpenSearch Service as the Destination

Use the following procedure to test your Firehose stream using Amazon OpenSearch Service as the destination.

To test a Firehose stream using OpenSearch Service

1. Open the Firehose console at https://console.amazonaws.cn/firehose/.
2. Choose the Firehose stream.
3. Under **Test with demo data**, choose **Start sending demo data** to generate sample stock ticker data.
4. Follow the onscreen instructions to verify that data is being delivered to your OpenSearch Service domain. For more information, see Searching Documents in an OpenSearch Service Domain in the *Amazon OpenSearch Service Developer Guide*.
5. When the test is complete, choose **Stop sending demo data** to stop incurring usage charges.

Test Using Splunk as the Destination

Use the following procedure to test your Firehose stream using Splunk as the destination.

To test a Firehose stream using Splunk

1. Open the Firehose console at https://console.amazonaws.cn/firehose/.
2. Choose the Firehose stream.
3. Under **Test with demo data**, choose **Start sending demo data** to generate sample stock ticker data.
4. Check whether the data is being delivered to your Splunk index. Example search terms in Splunk are `sourcetype="aws:firehose:json"` and `index="name-of-your-splunk-index"`. For more information about how to search for events in Splunk, see Search Manual in the Splunk documentation.

   If the test data doesn't appear in your Splunk index, check your Amazon S3 bucket for failed events. Also see Data Not Delivered to Splunk.
5. When you finish testing, choose **Stop sending demo data** to stop incurring usage charges.

Sending Data to a Firehose stream

You can send data to your Firehose stream using different types of sources: You can use a Kinesis data stream, the Kinesis Agent, or the Amazon Data Firehose API using the Amazon SDK. You can also use Amazon CloudWatch Logs, CloudWatch Events, or Amazon IoT as your data source. If you are new to Amazon Data Firehose, take some time to become familiar with the concepts and terminology presented in What Is Amazon Data Firehose?.

Note

Some Amazon services can only send messages and events to a Firehose stream that is in the same Region. If your Firehose stream doesn't appear as an option when you're configuring a target for Amazon CloudWatch Logs, CloudWatch Events, or Amazon IoT, verify that your Firehose stream is in the same Region as your other services.

Topics

Writing to Amazon Data Firehose Using Kinesis Data Streams

You can configure Amazon Kinesis Data Streams to send information to a Firehose stream.

Important

If you use the Kinesis Producer Library (KPL) to write data to a Kinesis data stream, you can use aggregation to combine the records that you write to that Kinesis data stream. If you then use that data stream as a source for your Firehose stream, Amazon Data Firehose

de-aggregates the records before it delivers them to the destination. If you configure your delivery stream to transform the data, Amazon Data Firehose de-aggregates the records before it delivers them to Amazon Lambda. For more information, see Developing Amazon Kinesis Data Streams Producers Using the Kinesis Producer Library and Aggregation.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Data Firehose console at https://console.amazonaws.cn/firehose/.
2. Choose **Create Firehose stream**. On the **Name and source** page, provide values for the following fields:

   Firehose stream name

   The name of your Firehose stream.

   Source

   Choose **Kinesis stream** to configure a Firehose stream that uses a Kinesis data stream as a data source. You can then use Amazon Data Firehose to read data easily from an existing data stream and load it into destinations.

   To use a Kinesis data stream as a source, choose an existing stream in the **Kinesis stream** list, or choose **Create new** to create a new Kinesis data stream. After you create a new stream, choose **Refresh** to update the **Kinesis stream** list. If you have a large number of streams, filter the list using **Filter by name**.

   > **Note**
   >
   > When you configure a Kinesis data stream as the source of a Firehose stream, the Amazon Data Firehose `PutRecord` and `PutRecordBatch` operations are disabled. To add data to your Firehose stream in this case, use the Kinesis Data Streams `PutRecord` and `PutRecords` operations.

   Amazon Data Firehose starts reading data from the `LATEST` position of your Kinesis stream. For more information about Kinesis Data Streams positions, see GetShardIterator. Amazon Data Firehose calls the Kinesis Data Streams GetRecords operation once per second for each shard.

More than one Firehose stream can read from the same Kinesis stream. Other Kinesis applications (consumers) can also read from the same stream. Each call from any Firehose stream or other consumer application counts against the overall throttling limit for the shard. To avoid getting throttled, plan your applications carefully. For more information about Kinesis Data Streams limits, see Amazon Kinesis Streams Limits.

3. Choose **Next** to advance to the Record Transformation and Format Conversion page.

Writing to Amazon Data Firehose Using Amazon MSK

You can configure Amazon MSK to send information to a Firehose stream.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Data Firehose console at https://console.amazonaws.cn/firehose/.
2. Choose **Create Firehose stream**.

   In the **Choose source and destination** section of the page, provide values for the following fields:

   Source

   Choose **Amazon MSK** to configure a Firehose stream that uses Amazon MSK as a data source. You can choose between MSK provisioned and MSK-Serverless clusters. You can then use Amazon Data Firehose to read data easily from a specific Amazon MSK cluster and topic and load it into the specified S3 destination.

   Destination

   Choose Amazon S3 as the destination for your Firehose stream.

   In the **Source settings** section of the page, provide values for the following fields:

   Amazon MSK cluster connectivity

   Choose either the **Private bootstrap brokers** (recommended) or **Public bootstrap brokers** option based on your cluster configuration. Bootstrap brokers is what Apache Kafka client uses as a starting point to connect to the cluster. Public bootstrap brokers are intended for public access from outside of Amazon, while private bootstrap brokers are intended

for access from within Amazon. For more information about Amazon MSK, see Amazon Managed Streaming for Apache Kafka.

To connect to a provisioned or serverless Amazon MSK cluster through private bootstrap brokers, the cluster must meet all of the following requirements.

- The cluster must be active.
- The cluster must have IAM as one of its access control methods.
- Multi-VPC private connectivity must be enabled for the IAM access control method.
- You must add to this cluster a resource-based policy which grants Amazon Data Firehose service principal the permission to invoke the Amazon MSK CreateVpcConnection API.

To connect to a provisioned Amazon MSK cluster through public bootstrap brokers, the cluster must meet all of the following requirements.

- The cluster must be active.
- The cluster must have IAM as one of its access control methods.
- The cluster must be public-accessible.

Amazon MSK cluster

For the same account scenario, specify the ARN of the Amazon MSK cluster from where your Firehose stream will read data.

For a cross-account scenario, see Cross-Account Delivery from Amazon MSK.

Topic

Specify the Apache Kafka topic from which you want your delivery stream to ingest data. Once the Firehose stream is created, you cannot update this topic.

In the **Firehose stream name** section of the page, provide values for the following fields:

Firehose stream name

Specify the name for your Firehose stream.

3. Next, you can complete the optional step of configuring record transformation and record format conversion. For more information, see Record Transformation and Format Conversion.

Writing to Amazon Data Firehose Using Kinesis Agent

Amazon Kinesis agent is a standalone Java software application that serves as a reference implementation to show how you can collect and send data to Firehose. The agent continuously monitors a set of files and sends new data to your Firehose delivery stream. The agent shows how you can handle file rotation, checkpointing, and retry upon failures. It shows how you can deliver your data in a reliable, timely, and simple manner. It also shows how you can emit CloudWatch metrics to better monitor and troubleshoot the streaming process. To learn more, awslabs/amazon-kinesis-agent.

By default, records are parsed from each file based on the newline (`'\n'`) character. However, the agent can also be configured to parse multi-line records (see Agent Configuration Settings).

You can install the agent on Linux-based server environments such as web servers, log servers, and database servers. After installing the agent, configure it by specifying the files to monitor and the Firehose stream for the data. After the agent is configured, it durably collects data from the files and reliably sends it to the Firehose stream.

Topics

Prerequisites

- Your operating system must be Amazon Linux, or Red Hat Enterprise Linux version 7 or later.
- Agent version 2.0.0 or later runs using JRE version 1.8 or later. Agent version 1.1.x runs using JRE 1.7 or later.

- If you are using Amazon EC2 to run your agent, launch your EC2 instance.
- The IAM role or Amazon credentials that you specify must have permission to perform the Amazon Data Firehose PutRecordBatch operation for the agent to send data to your Firehose stream. If you enable CloudWatch monitoring for the agent, permission to perform the CloudWatch PutMetricData operation is also needed. For more information, see Controlling Access with Amazon Data Firehose , Monitoring Kinesis Agent Health, and Authentication and Access Control for Amazon CloudWatch.

Credentials

Manage your Amazon credentials using one of the following methods:

- Create a custom credentials provider. For details, see the section called "Custom Credential Providers".
- Specify an IAM role when you launch your EC2 instance.
- Specify Amazon credentials when you configure the agent (see the entries for `awsAccessKeyId` and `awsSecretAccessKey` in the configuration table under the section called "Agent Configuration Settings").
- Edit `/etc/sysconfig/aws-kinesis-agent` to specify your Amazon Region and Amazon access keys.
- If your EC2 instance is in a different Amazon account, create an IAM role to provide access to the Amazon Data Firehose service. Specify that role when you configure the agent (see assumeRoleARN and assumeRoleExternalId). Use one of the previous methods to specify the Amazon credentials of a user in the other account who has permission to assume this role.

Custom Credential Providers

You can create a custom credentials provider and give its class name and jar path to the Kinesis agent in the following configuration settings: `userDefinedCredentialsProvider.classname` and `userDefinedCredentialsProvider.location`. For the descriptions of these two configuration settings, see the section called "Agent Configuration Settings".

To create a custom credentials provider, define a class that implements the `AmazonCredentialsProvider` interface, like the one in the following example.

```
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() {
    }
}
```

Your class must have a constructor that takes no arguments.

Amazon invokes the refresh method periodically to get updated credentials. If you want your credentials provider to provide different credentials throughout its lifetime, include code to refresh the credentials in this method. Alternatively, you can leave this method empty if you want a credentials provider that vends static (non-changing) credentials.

Download and Install the Agent

First, connect to your instance. For more information, see Connect to Your Instance in the *Amazon EC2 User Guide for Linux Instances*. If you have trouble connecting, see Troubleshooting Connecting to Your Instance in the *Amazon EC2 User Guide for Linux Instances*.

Next, install the agent using one of the following methods.

- **To set up the agent from the Amazon Linux repositories**

  This method works only for Amazon Linux instances. Use the following command:

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

  Agent v 2.0.0 or later is installed on computers with operating system Amazon Linux 2 (AL2). This agent version requires Java 1.8 or later. If required Java version is not yet present, the agent

installation process installs it. For more information regarding Amazon Linux 2 see https://aws.amazon.com/amazon-linux-2/.

- **To set up the agent from the Amazon S3 repository**

  This method works for Red Hat Enterprise Linux, as well as Amazon Linux 2 instances because it installs the agent from the publicly available repository. Use the following command to download and install the latest version of the agent version 2.x.x:

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

  To install a specific version of the agent, specify the version number in the command. For example, the following command installs agent v 2.0.1.

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

  If you have Java 1.7 and you don't want to upgrade it, you can download agent version 1.x.x, which is compatible with Java 1.7. For example, to download agent v1.1.6, you can use the following command:

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

  The latest agent v1.x.x can be downloaded using the following command:

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

- **To set up the agent from the GitHub repo**

  1. First, make sure that you have required Java version installed, depending on agent version.

2. Download the agent from the awslabs/amazon-kinesis-agent GitHub repo.
3. Install the agent by navigating to the download directory and running the following command:

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

- **To set up the agent in a Docker container**

  Kinesis Agent can be run in a container as well via the amazonlinux container base. Use the following Dockerfile and then run `docker build`.

```
FROM amazonlinux

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

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

Configure and Start the Agent

To configure and start the agent

1. Open and edit the configuration file (as superuser if using default file access permissions): `/etc/aws-kinesis/agent.json`

   In this configuration file, specify the files (`"filePattern"`) from which the agent collects data, and the name of the Firehose stream (`"deliveryStream"`) to which the agent sends data. The file name is a pattern, and the agent recognizes file rotations. You can rotate files or create new files no more than once per second. The agent uses the file creation time stamp to determine which files to track and tail into your Firehose stream. Creating new files or rotating files more frequently than once per second does not allow the agent to differentiate properly between them.

```
{
   "flows": [
        {
            "filePattern": "/tmp/app.log*",
```

```
            "deliveryStream": "yourdeliverystream"
        }
    ]
}
```

The default Amazon Region is `us-east-1`. If you are using a different Region, add the `firehose.endpoint` setting to the configuration file, specifying the endpoint for your Region. For more information, see Agent Configuration Settings.

2. Start the agent manually:

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

3. (Optional) Configure the agent to start on system startup:

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

The agent is now running as a system service in the background. It continuously monitors the specified files and sends data to the specified Firehose stream. Agent activity is logged in `/var/log/aws-kinesis-agent/aws-kinesis-agent.log`.

Agent Configuration Settings

The agent supports two mandatory configuration settings, `filePattern` and `deliveryStream`, plus optional configuration settings for additional features. You can specify both mandatory and optional configuration settings in `/etc/aws-kinesis/agent.json`.

Whenever you change the configuration file, you must stop and start the agent, using the following commands:

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

Alternatively, you could use the following command:

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

The following are the general configuration settings.

Configuration Setting	Description
`assumeRoleARN`	The Amazon Resource Name (ARN) of the role to be assumed by the user. For more information, see Delegate Access Across Amazon Accounts Using IAM Roles in the *IAM User Guide*.
`assumeRoleExternalId`	An optional identifier that determines who can assume the role. For more information, see How to Use an External ID in the *IAM User Guide*.
`awsAccessKeyId`	Amazon access key ID that overrides the default credentials. This setting takes precedence over all other credential providers.
`awsSecretAccessKey`	Amazon secret key that overrides the default credentials. This setting takes precedence over all other credential providers.
`cloudwatch.emitMetrics`	Enables the agent to emit metrics to CloudWatch if set (true). Default: true
`cloudwatch.endpoint`	The regional endpoint for CloudWatch. Default: `monitoring.us-east-1.amazonaws.com`
`firehose.endpoint`	The regional endpoint for Amazon Data Firehose. Default: `firehose.us-east-1.amazonaws.com`
`sts.endpoint`	The regional endpoint for the Amazon Security Token Service. Default: `https://sts.amazonaws.com`
`userDefinedCredentialsProvider.classname`	If you define a custom credentials provider, provide its fully-qualified class name using this setting. Don't include `.class` at the end of the class name.
`userDefinedCredent`	If you define a custom credentials provider, use this setting to specify the absolute path of the jar that contains the custom credentials

Configuration Setting	Description
ialsProvider.location	provider. The agent also looks for the jar file in the following location: /usr/share/aws-kinesis-agent/lib/ .

The following are the flow configuration settings.

Configuration Setting	Description
aggregatedRecordSizeBytes	To make the agent aggregate records and then put them to the Firehose stream in one operation, specify this setting. Set it to the size that you want the aggregate record to have before the agent puts it to the Firehose stream. Default: 0 (no aggregation)
dataProcessingOptions	The list of processing options applied to each parsed record before it is sent to the Firehose stream. The processing options are performed in the specified order. For more information, see Use the agent to Preprocess Data.
deliveryStream	[Required] The name of the Firehose stream.
filePattern	[Required] A glob for the files that need to be monitored by the agent. Any file that matches this pattern is picked up by the agent automatically and monitored. For all files matching this pattern, grant read permission to aws-kinesis-agent-user . For the directory containing the files, grant read and execute permissions to aws-kinesis-agent-user . **Important** The agent picks up any file that matches this pattern. To ensure that the agent doesn't pick up unintended records, choose this pattern carefully.

Configuration Setting	Description
initialPosition	The initial position from which the file started to be parsed. Valid values are START_OF_FILE and END_OF_FILE . Default: END_OF_FILE
maxBuffer AgeMillis	The maximum time, in milliseconds, for which the agent buffers data before sending it to the Firehose stream. Value range: 1,000–900,000 (1 second to 15 minutes) Default: 60,000 (1 minute)
maxBuffer SizeBytes	The maximum size, in bytes, for which the agent buffers data before sending it to the Firehose stream. Value range: 1–4,194,304 (4 MB) Default: 4,194,304 (4 MB)
maxBuffer SizeRecords	The maximum number of records for which the agent buffers data before sending it to the Firehose stream. Value range: 1–500 Default: 500
minTimeBe tweenFile PollsMillis	The time interval, in milliseconds, at which the agent polls and parses the monitored files for new data. Value range: 1 or more Default: 100
multiLine StartPattern	The pattern for identifying the start of a record. A record is made of a line that matches the pattern and any following lines that don't match the pattern. The valid values are regular expressions. By default, each new line in the log files is parsed as one record.

Configuration Setting	Description
skipHeaderLines	The number of lines for the agent to skip parsing at the beginning of monitored files. Value range: 0 or more Default: 0 (zero)
truncatedRecordTerminator	The string that the agent uses to truncate a parsed record when the record size exceeds the Amazon Data Firehose record size limit. (1,000 KB) Default: '\n' (newline)

Monitor Multiple File Directories and Write to Multiple Streams

By specifying multiple flow configuration settings, you can configure the agent to monitor multiple file directories and send data to multiple streams. In the following configuration example, the agent monitors two file directories and sends data to a Kinesis data stream and a Firehose stream respectively. You can specify different endpoints for Kinesis Data Streams and Amazon Data Firehose so that your data stream and Firehose stream don't need to be in the same Region.

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

For more detailed information about using the agent with Amazon Kinesis Data Streams, see Writing to Amazon Kinesis Data Streams with Kinesis Agent.

Use the agent to Preprocess Data

The agent can pre-process the records parsed from monitored files before sending them to your Firehose stream. You can enable this feature by adding the `dataProcessingOptions` configuration setting to your file flow. One or more processing options can be added, and they are performed in the specified order.

The agent supports the following processing options. Because the agent is open source, you can further develop and extend its processing options. You can download the agent from Kinesis Agent.

Processing Options

`SINGLELINE`

Converts a multi-line record to a single-line record by removing newline characters, leading spaces, and trailing spaces.

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

`CSVTOJSON`

Converts a record from delimiter-separated format to JSON format.

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

`customFieldNames`

[Required] The field names used as keys in each JSON key value pair. For example, if you specify `["f1", "f2"]`, the record "v1, v2" is converted to `{"f1":"v1","f2":"v2"}`.

`delimiter`

The string used as the delimiter in the record. The default is a comma (,).

LOGTOJSON

Converts a record from a log format to JSON format. The supported log formats are **Apache Common Log**, **Apache Combined Log**, **Apache Error Log**, and **RFC3164 Syslog**.

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

logFormat

[Required] The log entry format. The following are possible values:

- COMMONAPACHELOG — The Apache Common Log format. Each log entry has the following pattern by default: "%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes}".
- COMBINEDAPACHELOG — The Apache Combined Log format. Each log entry has the following pattern by default: "%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes} %{referrer} %{agent}".
- APACHEERRORLOG — The Apache Error Log format. Each log entry has the following pattern by default: "[%{timestamp}] [%{module}:%{severity}] [pid %{processid}:tid %{threadid}] [client: %{client}] %{message}".
- SYSLOG — The RFC3164 Syslog format. Each log entry has the following pattern by default: "%{timestamp} %{hostname} %{program}[%{processid}]: %{message}".

matchPattern

Overrides the default pattern for the specified log format. Use this setting to extract values from log entries if they use a custom format. If you specify matchPattern, you must also specify customFieldNames.

customFieldNames

The custom field names used as keys in each JSON key value pair. You can use this setting to define field names for values extracted from matchPattern, or override the default field names of predefined log formats.

Example : LOGTOJSON Configuration

Here is one example of a LOGTOJSON configuration for an Apache Common Log entry converted to JSON format:

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

Before conversion:

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

After conversion:

```
{"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 Configuration With Custom Fields

Here is another example LOGTOJSON configuration:

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

With this configuration setting, the same Apache Common Log entry from the previous example is converted to JSON format as follows:

```
{"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 : Convert Apache Common Log Entry

The following flow configuration converts an Apache Common Log entry to a single-line record in JSON format:

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

Example : Convert Multi-Line Records

The following flow configuration parses multi-line records whose first line starts with "[SEQUENCE=". Each record is first converted to a single-line record. Then, values are extracted from the record based on a tab delimiter. Extracted values are mapped to specified `customFieldNames` values to form a single-line record in 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 Configuration with Match Pattern

Here is one example of a LOGTOJSON configuration for an Apache Common Log entry converted to JSON format, with the last field (bytes) omitted:

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

Before conversion:

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

After conversion:

```
{"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"}
```

Agent CLI Commands

Automatically start the agent on system startup:

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

Check the status of the agent:

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

Stop the agent:

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

Read the agent's log file from this location:

```
/var/log/aws-kinesis-agent/aws-kinesis-agent.log
```

Uninstall the agent:

```
sudo yum remove aws-kinesis-agent
```

FAQ

Is there a Kinesis Agent for Windows?

Kinesis Agent for Windows is different software than Kinesis Agent for Linux platforms.

Why is Kinesis Agent slowing down and/or `RecordSendErrors` increasing?

This is usually due to throttling from Kinesis. Check the `WriteProvisionedThroughputExceeded` metric for Kinesis Data Streams or the `ThrottledRecords` metric for Firehose streams. Any increase from 0 in these metrics indicates that the stream limits need to be increased. For more information, see Kinesis Data Stream limits and Firehose streams.

Once you rule out throttling, see if the Kinesis Agent is configured to tail a large amount of small files. There is a delay when Kinesis Agent tails a new file, so Kinesis Agent should be tailing a small amount of larger files. Try consolidating your log files into larger files.

Why am I getting `java.lang.OutOfMemoryError` exceptions?

Kinesis Agent does not have enough memory to handle its current workload. Try increasing `JAVA_START_HEAP` and `JAVA_MAX_HEAP` in `/usr/bin/start-aws-kinesis-agent` and restarting the agent.

Why am I getting `IllegalStateException : connection pool shut down` exceptions?

Kinesis Agent does not have enough connections to handle its current workload. Try increasing `maxConnections` and `maxSendingThreads` in your general agent configuration settings at `/etc/aws-kinesis/agent.json`. The default value for these fields is 12 times the runtime processors available. See AgentConfiguration.java for more about advanced agent configurations settings.

How can I debug another issue with Kinesis Agent?

DEBUG level logs can be enabled in `/etc/aws-kinesis/log4j.xml` .

How should I configure Kinesis Agent?

The smaller the `maxBufferSizeBytes`, the more frequently Kinesis Agent will send data. This can be good as it decreases delivery time of records, but it also increases the requests per second to Kinesis.

Why is Kinesis Agent sending duplicate records?

This occurs due to a misconfiguration in file tailing. Make sure that each `fileFlow's filePattern` is only matching one file. This can also occur if the `logrotate` mode being used is in `copytruncate` mode. Try changing the mode to the default or create mode to avoid duplication. For more information on handling duplicate records, see Handling Duplicate Records.

Writing to Amazon Data Firehose Using the Amazon SDK

You can use the Amazon Data Firehose API to send data to a Firehose stream using the Amazon SDK for Java, .NET, Node.js, Python, or Ruby. If you are new to Amazon Data Firehose, take some time to become familiar with the concepts and terminology presented in What Is Amazon Data Firehose?. For more information, see Start Developing with Amazon Web Services.

These examples do not represent production-ready code, in that they do not check for all possible exceptions, or account for all possible security or performance considerations.

The Amazon Data Firehose API offers two operations for sending data to your Firehose stream: PutRecord and PutRecordBatch. `PutRecord()` sends one data record within one call and `PutRecordBatch()` can send multiple data records within one call.

Topics

- Single Write Operations Using PutRecord
- Batch Write Operations Using PutRecordBatch

Single Write Operations Using PutRecord

Putting data requires only the Firehose stream name and a byte buffer (<=1000 KB). Because Amazon Data Firehose batches multiple records before loading the file into Amazon S3, you may

want to add a record separator. To put data one record at a time into a Firehose stream, use the following code:

```
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);
```

For more code context, see the sample code included in the Amazon SDK. For information about request and response syntax, see the relevant topic in Firehose API Operations.

Batch Write Operations Using PutRecordBatch

Putting data requires only the Firehose stream name and a list of records. Because Amazon Data Firehose batches multiple records before loading the file into Amazon S3, you may want to add a record separator. To put data records in batches into a Firehose stream, use the following code:

```
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();
```

For more code context, see the sample code included in the Amazon SDK. For information about request and response syntax, see the relevant topic in Firehose API Operations.

Writing to Amazon Data Firehose Using CloudWatch Logs

CloudWatch Logs events can be sent to Firehose using CloudWatch subscription filters. For more information, see Subscription filters with Amazon Data Firehose.

CloudWatch Logs events are sent to Firehose in compressed gzip format. If you want to deliver decompressed log events to Firehose destinations, you can use the decompression feature in Firehose to automatically decompress CloudWatch Logs.

> **Important**
>
> Currently, Firehose does not support the delivery of CloudWatch Logs to Amazon OpenSearch Service destination because Amazon CloudWatch combines multiple log events into one Firehose record and Amazon OpenSearch Service cannot accept multiple log events in one record. As an alternative, you can consider Using subscription filter for Amazon OpenSearch Service in CloudWatch Logs.

Decompression of CloudWatch Logs

If you are using Firehose to deliver CloudWatch Logs and want to deliver decompressed data to your delivery stream destination, use Firehose Data Format Conversion (Parquet, ORC) or Dynamic partitioning. You must enable decompression for your Firehose delivery stream.

You can enable decompression using the Amazon Web Services Management Console, Amazon Command Line Interface or Amazon SDKs.

> **Note**
>
> If you enable the decompression feature on a stream, use that stream exclusively for CloudWatch Logs subscriptions filters, and not for Vended Logs. If you enable the decompression feature on a stream that is used to ingest both CloudWatch Logs and Vended Logs, the Vended Logs ingestion to Firehose fails. This decompression feature is only for CloudWatch Logs.

Message extraction after decompression of CloudWatch Logs

When you enable decompression, you have the option to also enable message extraction. When using message extraction, Firehose filters out all metadata, such as owner, loggroup, logstream, and others from the decompressed CloudWatch Logs records and delivers only the content inside the message fields. If you are delivering data to a Splunk destination, you must turn on message

extraction for Splunk to parse the data. Following are sample outputs after decompression with and without message extraction.

Fig 1: Sample output after decompression without message extraction:


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


Fig 2: Sample output after decompression with message extraction:


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


Enabling and disabling decompression

You can enable and disable decompression using the Amazon Web Services Management Console, Amazon Command Line Interface or Amazon SDKs.

Enabling decompression on a new data stream using the Amazon Web Services Management Console

To enable decompression on a new data stream using the Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Kinesis console at https://console.amazonaws.cn/kinesis.
2. Choose **Data Firehose** in the navigation pane.
3. Choose **Create delivery stream**.
4. Under **Choose source and destination**

   Delivery stream source

   The source of your Firehose stream. Choose one of the following sources:

   - **Direct PUT** – Choose this option to create a Firehose stream that producer applications write to directly. Currently, the following are Amazon services and agents and open source services that are integrated with Direct PUT in Firehose:
   - **Kinesis stream:** Choose this option to configure a Firehose stream that uses a Kinesis data stream as a data source. You can then use Firehose to read data easily from an existing Kinesis data stream and load it into destinations. For more information, see Writing to Firehose Using Kinesis Data Streams

   Destination

   The destination of your Firehose stream. Choose one of the following:

   - Amazon S3
   - Splunk
5. Under **Delivery stream name**, enter a name for your stream.
6. Under **Transform records - *optional***:

   - In the **Decompress source records from Amazon CloudWatch Logs** section, choose **Turn on decompression**.
   - If you want to use message extraction after decompression, choose **Turn on message extraction**.

Enabling decompression on an existing data stream using the Amazon Web Services Management Console

If you have a Firehose stream with a Lambda function to perform decompression, you can replace it with the Firehose decompression feature. Before you proceed, review your Lambda function code to confirm that it only performs decompression or message extraction. The output of your Lambda function should look similar to the examples shown in Fig 1 or Fig 2 in the previous section. If the output looks similar, you can replace the Lambda function using the following steps.

1. Replace your current Lambda function with this blueprint. The new blueprint Lambda function automatically detects whether the incoming data is compressed or decompressed. It only performs decompression if its input data is compressed.
2. Turn on decompression using the built-in Firehose option for decompression.
3. Enable CloudWatch metrics for your Firehose stream if it's not already enabled. Monitor the metric CloudWatchProcessorLambda_IncomingCompressedData and wait until this metric changes to zero. This confirms that all input data sent to your Lambda function is decompressed and the Lambda function is no longer required.
4. Remove the Lambda data transformation because you no longer need it to decompress your stream.

Disabling decompression using the Amazon Web Services Management Console

To disable decompression on a data stream using the Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Kinesis console at https://console.amazonaws.cn/kinesis.
2. Choose **Data Firehose** in the navigation pane.
3. Choose the delivery stream you wish to edit.
4. Choose **Configuration**.
5. In the **Transform records** section, choose **Edit**.
6. Under **Decompress source records from Amazon CloudWatch Logs**, clear **Turn on decompression** and then choose **Save changes**.

FAQ

What happens to the source data in case of an error during decompression?

If Amazon Data Firehose is not able to decompress the record, the record is delivered as is (in compressed format) to error S3 bucket you specified during delivery stream creation time. Along with the record, the delivered object also includes error code and error message and these objects will be delivered to an S3 bucket prefix called `decompression-failed`. Firehose will continue to process other records after a failed decompression of a record.

What happens to the source data in case of an error in the processing pipeline after successful decompression?

If Amazon Data Firehose errors out in the processing steps after decompression like Dynamic Partitioning and Data Format Conversion, the record is delivered in compressed format to the error S3 bucket you specified during delivery stream creation time. Along with the record, the delivered object also includes error code and error message.

How are you informed in case of an error or an exception?

In case of an error or an exception during decompression, if you configure CloudWatch Logs, Firehose will log error messages into CloudWatch Logs. Additionally, Firehose sends metrics to CloudWatch metrics that you can monitor. You can also optionally create alarms based on metrics emitted by Firehose.

What happens when `put` operations don't come from CloudWatch Logs?

When customer `puts` do not come from CloudWatch Logs, then the following error message is returned:

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

What metrics does Firehose emit for the decompression feature?

Firehose emits metrics for decompression of every record. You should select the period (1 min), statistic (sum), date range to get the number of `DecompressedRecords` failed or succeeded or `DecompressedBytes` failed or succeeded. For more information, see CloudWatch Logs Decompression Metrics.

Writing to Amazon Data Firehose Using CloudWatch Events

You can configure Amazon CloudWatch to send events to a Firehose stream by adding a target to a CloudWatch Events rule.

To create a target for a CloudWatch Events rule that sends events to an existing delivery stream

1. Sign in to the Amazon Web Services Management Console and open the CloudWatch console at https://console.amazonaws.cn/cloudwatch/.
2. Choose **Create rule**.
3. On the **Step 1: Create rule** page, for **Targets**, choose **Add target**, and then choose **Firehose Firehose stream**.
4. For **Firehose stream**, choose an existing Amazon Data Firehose delivery stream.

For more information about creating CloudWatch Events rules, see Getting Started with Amazon CloudWatch Events.

Writing to Amazon Data Firehose Using Amazon IoT

You can configure Amazon IoT to send information to a Firehose stream by adding an action.

To create an action that sends events to an existing Firehose stream

1. When creating a rule in the Amazon IoT console, on the **Create a rule** page, under **Set one or more actions**, choose **Add action**.
2. Choose **Send messages to an Amazon Kinesis Firehose stream**.
3. Choose **Configure action**.
4. For **Stream name**, choose an existing Firehose stream.
5. For **Separator**, choose a separator character to be inserted between records.
6. For **IAM role name**, choose an existing IAM role or choose **Create a new role**.
7. Choose **Add action**.

For more information about creating Amazon IoT rules, see Amazon IoT Rule Tutorials.

Security in Amazon Data Firehose

Cloud security at Amazon is the highest priority. As an Amazon customer, you will benefit from a data center and network architecture built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between Amazon and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – Amazon is responsible for protecting the infrastructure that runs Amazon services in the Amazon Cloud. Amazon also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the Amazon compliance programs. To learn about the compliance programs that apply to Data Firehose, see Amazon Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the Amazon service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Data Firehose. The following topics show you how to configure Data Firehose to meet your security and compliance objectives. You'll also learn how to use other Amazon services that can help you to monitor and secure your Data Firehose resources.

Topics

Data Protection in Amazon Data Firehose

Amazon Data Firehose encrypts all data in transit using TLS protocol. Furthermore, for data stored in interim storage during processing, Amazon Data Firehose encrypts data using Amazon Key Management Service and verifies data integrity using checksum verification.

If you have sensitive data, you can enable server-side data encryption when you use Amazon Data Firehose. How you do this depends on the source of your data.

> **Note**
>
> If you require FIPS 140-2 validated cryptographic modules when accessing Amazon through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

Server-Side Encryption with Kinesis Data Streams as the Data Source

When you send data from your data producers to your data stream, Kinesis Data Streams encrypts your data using an Amazon Key Management Service (Amazon KMS) key before storing the data at rest. When your Amazon Data Firehose Firehose stream reads the data from your data stream, Kinesis Data Streams first decrypts the data and then sends it to Amazon Data Firehose. Amazon Data Firehose buffers the data in memory based on the buffering hints that you specify. It then delivers it to your destinations without storing the unencrypted data at rest.

For information about how to enable server-side encryption for Kinesis Data Streams, see Using Server-Side Encryption in the *Amazon Kinesis Data Streams Developer Guide*.

Server-Side Encryption with Direct PUT or Other Data Sources

If you send data to your Firehose stream using PutRecord or PutRecordBatch, or if you send the data using Amazon IoT, Amazon CloudWatch Logs, or CloudWatch Events, you can turn on server-side encryption by using the StartDeliveryStreamEncryption operation.

To stop server-side-encryption, use the StopDeliveryStreamEncryption operation.

You can also enable SSE when you create the Firehose stream. To do that, specify DeliveryStreamEncryptionConfigurationInput when you invoke CreateDeliveryStream.

When the CMK is of type CUSTOMER_MANAGED_CMK, if the Amazon Data Firehose service is unable to decrypt records because of a KMSNotFoundException, a KMSInvalidStateException, a KMSDisabledException, or a KMSAccessDeniedException, the service waits up to 24 hours (the retention period) for you to resolve the problem. If the problem persists beyond the retention period, the service skips those records that have passed the retention period and couldn't be decrypted, and then discards the data. Amazon Data Firehose provides the following four CloudWatch metrics that you can use to track the four Amazon KMS exceptions:

- KMSKeyAccessDenied
- KMSKeyDisabled
- KMSKeyInvalidState
- KMSKeyNotFound

For more information about these four metrics, see the section called “Monitoring with CloudWatch Metrics”.

Important

To encrypt your Firehose stream, use symmetric CMKs. Amazon Data Firehose doesn't support asymmetric CMKs. For information about symmetric and asymmetric CMKs, see About Symmetric and Asymmetric CMKs in the Amazon Key Management Service developer guide.

Note

When you use a customer managed key (CUSTOMER_MANAGED_CMK) to enable server-side encryption (SSE) for your Firehose delivery stream, the Firehose service sets an encryption context whenever it uses your key. Since this encryption context represents an occurrence where a key owned by your Amazon account was used, it is logged as part of Amazon CloudTrail event logs for your Amazon account. This encryption context is system generated by the Firehose service. Your application should not make any assumptions about the format or content of the encryption context set by the Firehose service.

Controlling Access with Amazon Data Firehose

The following sections cover how to control access to and from your Amazon Data Firehose resources. The information they cover includes how to grant your application access so it can send data to your Firehose stream. They also describe how you can grant Amazon Data Firehose access to your Amazon Simple Storage Service (Amazon S3) bucket, Amazon Redshift cluster, or Amazon OpenSearch Service cluster, as well as the access permissions you need if you use Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Splunk, or Sumo Logic as your destination. Finally, you'll find in this topic guidance on how to configure Amazon Data Firehose so it can deliver data to a destination that belongs to a different Amazon account. The technology for managing all these forms of access is Amazon Identity and Access Management (IAM). For more information about IAM, see What is IAM?.

Contents

Grant Your Application Access to Your Amazon Data Firehose Resources

To give your application access to your Firehose stream, use a policy similar to this example. You can adjust the individual API operations to which you grant access by modifying the `Action` section, or grant access to all operations with `"firehose:*"`.

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

Grant Amazon Data Firehose Access to your Private Amazon MSK Cluster

If the source of your Firehose stream is a private Amazon MSK cluster, then use a policy similar to this example.

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

```
    "Action": [
      "kafka:CreateVpcConnection"
    ],
    "Resource": "cluster-arn"
    }
  ]
}
```

Allow Amazon Data Firehose to Assume an IAM Role

This section describes the permissions and policies that grant Amazon Data Firehose access to ingest, process, and deliver data from source to destination.

> **Note**
>
> If you use the console to create a Firehose stream and choose the option to create a new role, Amazon attaches the required trust policy to the role. If you want Amazon Data Firehose to use an existing IAM role or if you create a role on your own, attach the following trust policies to that role so that Amazon Data Firehose can assume it. Edit the policies to replace *account-id* with your Amazon account ID. For information about how to modify the trust relationship of a role, see Modifying a Role.

Amazon Data Firehose uses an IAM role for all the permissions that the delivery stream needs to process and deliver data. Make sure that the following trust policies are attached to that role so that Amazon Data Firehose can assume it.

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
   "Service": "firehose.amazonaws.com"
  },
  "Action": "sts:AssumeRole",
  "Condition": {
   "StringEquals": {
    "sts:ExternalId": "account-id"
   }
  }
```

```
 }]
}
```

This policy uses the `sts:ExternalId` condition context key to ensure that only Amazon Data Firehose activity originating from your Amazon account can assume this IAM role. For more information about preventing unauthorized use of IAM roles, see The confused deputy problem in the *IAM User Guide*.

If you choose Amazon MSK as the source for your Firehose stream, you must specify another IAM role that grants Amazon Data Firehose permissions to ingest source data from the specified Amazon MSK cluster. Make sure that the following trust policies are attached to that role so that Amazon Data Firehose can assume it.

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

Make sure that this role that grants Amazon Data Firehose permissions to ingest source data from the specified Amazon MSK cluster grants the following permissions:

```
{
   "Version": "2012-10-17",
   "Statement": [{
        "Effect":"Allow",
        "Action": [
           "kafka:GetBootstrapBrokers",
           "kafka:DescribeCluster",
```

```
            "kafka:DescribeClusterV2",
            "kafka-cluster:Connect"
          ],
          "Resource": "CLUSTER-ARN"
        },
        {
          "Effect":"Allow",
          "Action": [
            "kafka-cluster:DescribeTopic",
            "kafka-cluster:DescribeTopicDynamicConfiguration",
            "kafka-cluster:ReadData"
          ],
          "Resource": "TOPIC-ARN"
        }]
}
```

Grant Amazon Data Firehose Access to Amazon Glue for Data Format Conversion

If your Firehose stream performs data-format conversion, Amazon Data Firehose references table definitions stored in Amazon Glue. To give Amazon Data Firehose the necessary access to Amazon Glue, add the following statement to your policy. For information on how to find the ARN of the table, see Specifying Amazon Glue Resource ARNs.

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

The recommended policy for getting schemas from schema registry has no resource restrictions. For more information, see IAM examples for deserializers in the Amazon Glue Developer Guide.

> **Note**
>
> Currently, Amazon Glue is not supported in the Israel (Tel Aviv), Asia Pacific (Jakarta) or Middle East (UAE) Regions. If you are working with Amazon Data Firehose in the Asia Pacific (Jakarta) Region or Middle East (UAE) Region, make sure to give your Amazon Data Firehose access to Amazon Glue in one of the Regions where Amazon Glue is currently supported. Cross-region interoperability between Data Firehose and Amazon Glue is supported. For more information on regions where Amazon Glue is supported, see https://docs.aws.amazon.com/general/latest/gr/glue.html

Grant Amazon Data Firehose Access to an Amazon S3 Destination

When you're using an Amazon S3 destination, Amazon Data Firehose delivers data to your S3 bucket and can optionally use an Amazon KMS key that you own for data encryption. If error logging is enabled, Amazon Data Firehose also sends data delivery errors to your CloudWatch log group and streams. You are required to have an IAM role when creating a Firehose stream. Amazon Data Firehose assumes that IAM role and gains access to the specified bucket, key, and CloudWatch log group and streams.

Use the following access policy to enable Amazon Data Firehose to access your S3 bucket and Amazon KMS key. If you don't own the S3 bucket, add `s3:PutObjectAcl` to the list of Amazon S3 actions. This grants the bucket owner full access to the objects delivered by Amazon Data Firehose.

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
```

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

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

The policy above also has a statement that allows access to Amazon Kinesis Data Streams. If you don't use Kinesis Data Streams as your data source, you can remove that statement. If you use Amazon MSK as your source, then you can substitute that statement with the following:

```
{
   "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}}/*"
}
```

For more information about allowing other Amazon services to access your Amazon resources, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.

To learn how to grant Amazon Data Firehose access to an Amazon S3 destination in another account, see the section called “Cross-Account Delivery to an Amazon S3 Destination”.

Grant Amazon Data Firehose Access to an Amazon Redshift Destination

Refer to the following when you are granting access to Amazon Data Firehose when using an Amazon Redshift destination.

Topics

- IAM Role and Access Policy
- VPC Access to an Amazon Redshift Provisioned Cluster or Amazon Redshift Serverless Workgroup

IAM Role and Access Policy

When you're using an Amazon Redshift destination, Amazon Data Firehose delivers data to your S3 bucket as an intermediate location. It can optionally use an Amazon KMS key you own for data encryption. Amazon Data Firehose then loads the data from the S3 bucket to your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup. If error logging is enabled, Amazon Data Firehose also sends data delivery errors to your CloudWatch log group and streams. Amazon Data Firehose uses the specified Amazon Redshift user name and password to access your provisioned cluster or Amazon Redshift Serverless workgroup, and uses an IAM role to access the specified bucket, key, CloudWatch log group, and streams. You are required to have an IAM role when creating a Firehose stream.

Use the following access policy to enable Amazon Data Firehose to access your S3 bucket and Amazon KMS key. If you don't own the S3 bucket, add `s3:PutObjectAcl` to the list of Amazon

S3 actions, which grants the bucket owner full access to the objects delivered by Amazon Data Firehose. This policy also has a statement that allows access to Amazon Kinesis Data Streams. If you don't use Kinesis Data Streams as your data source, you can remove that statement.

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

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

For more information about allowing other Amazon services to access your Amazon resources, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.

VPC Access to an Amazon Redshift Provisioned Cluster or Amazon Redshift Serverless Workgroup

If your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup is in a virtual private cloud (VPC), it must be publicly accessible with a public IP address. Also, grant Amazon Data Firehose access to your Amazon Redshift provisioned cluster or Amazon Redshift

Serverless workgroup by unblocking the Amazon Data Firehose IP addresses. Amazon Data Firehose currently uses one CIDR block for each available Region:

- 13.58.135.96/27 for US East (Ohio)
- 52.70.63.192/27 for US East (N. Virginia)
- 13.57.135.192/27 for US West (N. California)
- 52.89.255.224/27 for US West (Oregon)
- 18.253.138.96/27 for Amazon GovCloud (US-East)
- 52.61.204.160/27 for Amazon GovCloud (US-West)
- 35.183.92.128/27 for Canada (Central)
- 40.176.98.192/27 for Canada West (Calgary)
- 18.162.221.32/27 for Asia Pacific (Hong Kong)
- 13.232.67.32/27 for Asia Pacific (Mumbai)
- 18.60.192.128/27 for Asia Pacific (Hyderabad)
- 13.209.1.64/27 for Asia Pacific (Seoul)
- 13.228.64.192/27 for Asia Pacific (Singapore)
- 13.210.67.224/27 for Asia Pacific (Sydney)
- 108.136.221.64/27 for Asia Pacific (Jakarta)
- 13.113.196.224/27 for Asia Pacific (Tokyo)
- 13.208.177.192/27 for Asia Pacific (Osaka)
- 52.81.151.32/27 for China (Beijing)
- 161.189.23.64/27 for China (Ningxia)
- 16.62.183.32/27 for Europe (Zurich)
- 35.158.127.160/27 for Europe (Frankfurt)
- 52.19.239.192/27 for Europe (Ireland)
- 18.130.1.96/27 for Europe (London)
- 35.180.1.96/27 for Europe (Paris)
- 13.53.63.224/27 for Europe (Stockholm)
- 15.185.91.0/27 for Middle East (Bahrain)
- 18.228.1.128/27 for South America (São Paulo)
- 15.161.135.128/27 for Europe (Milan)

- `13.244.121.224/27` for Africa (Cape Town)
- `3.28.159.32/27` for Middle East (UAE)
- `51.16.102.0/27` for Israel (Tel Aviv)
- `16.50.161.128/27` for Asia Pacific (Melbourne)

For more information about how to unblock IP addresses, see the step Authorize Access to the Cluster in the *Amazon Redshift Getting Started Guide* guide.

Grant Amazon Data Firehose Access to a Public OpenSearch Service Destination

When you're using an OpenSearch Service destination, Amazon Data Firehose delivers data to your OpenSearch Service cluster, and concurrently backs up failed or all documents to your S3 bucket. If error logging is enabled, Amazon Data Firehose also sends data delivery errors to your CloudWatch log group and streams. Amazon Data Firehose uses an IAM role to access the specified OpenSearch Service domain, S3 bucket, Amazon KMS key, and CloudWatch log group and streams. You are required to have an IAM role when creating a Firehose stream.

Use the following access policy to enable Amazon Data Firehose to access your S3 bucket, OpenSearch Service domain, and Amazon KMS key. If you do not own the S3 bucket, add `s3:PutObjectAcl` to the list of Amazon S3 actions, which grants the bucket owner full access to the objects delivered by Amazon Data Firehose. This policy also has a statement that allows access to Amazon Kinesis Data Streams. If you don't use Kinesis Data Streams as your data source, you can remove that statement.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:region:account-id:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.region.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/
prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "es:DescribeDomain",
                "es:DescribeDomains",
                "es:DescribeDomainConfig",
                "es:ESHttpPost",
                "es:ESHttpPut"
            ],
          "Resource": [
              "arn:aws:es:region:account-id:domain/domain-name",
              "arn:aws:es:region:account-id:domain/domain-name/*"
          ]
       },
       {
          "Effect": "Allow",
          "Action": [
              "es:ESHttpGet"
          ],
          "Resource": [
              "arn:aws:es:region:account-id:domain/domain-name/_all/_settings",
```

```
                "arn:aws:es:region:account-id:domain/domain-name/_cluster/stats",
                "arn:aws:es:region:account-id:domain/domain-name/index-name*/
_mapping/type-name",
                "arn:aws:es:region:account-id:domain/domain-name/_nodes",
                "arn:aws:es:region:account-id:domain/domain-name/_nodes/stats",
                "arn:aws:es:region:account-id:domain/domain-name/_nodes/*/stats",
                "arn:aws:es:region:account-id:domain/domain-name/_stats",
                "arn:aws:es:region:account-id:domain/domain-name/index-name*/_stats",
                "arn:aws:es:region:account-id:domain/domain-name/"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:log-
stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:region:account-id:function:function-name:function-
version"
            ]
        }
    ]
```

```
}
```

For more information about allowing other Amazon services to access your Amazon resources, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.

To learn how to grant Amazon Data Firehose access to an OpenSearch Service cluster in another account, see the section called "Cross-Account Delivery to an OpenSearch Service Destination".

Grant Amazon Data Firehose Access to an OpenSearch Service Destination in a VPC

If your OpenSearch Service domain is in a VPC, make sure you give Amazon Data Firehose the permissions that are described in the previous section. In addition, you need to give Amazon Data Firehose the following permissions to enable it to access your OpenSearch Service domain's VPC.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

Important

Do not revoke these permissions after you create the delivery stream. If you revoke these permissions, your Firehose stream will be degraded or stop delivering data to your OpenSearch service domain whenever the service attempts to query or update ENIs.

Important

When you specify subnets for delivering data to the destination in a private VPC, make sure you have enough number of free IP addresses in chosen subnets. If there is no available

free IP address in a specified subnet, Firehose cannot create or add ENIs for the data delivery in the private VPC, and the delivery will be degraded or fail.

When you create or update your delivery stream, you specify a security group for Firehose to use when it sends data to your OpenSearch Service domain. You can use the same security group that the OpenSearch Service domain uses or a different one. If you specify a different security group, ensure that it allows outbound HTTPS traffic to the OpenSearch Service domain's security group. Also ensure that the OpenSearch Service domain's security group allows HTTPS traffic from the security group you specified when you configured your Firehose stream. If you use the same security group for both your Firehose stream and the OpenSearch Service domain, make sure the security group inbound rule allows HTTPS traffic. For more information about security group rules, see Security group rules in the Amazon VPC documentation.

Grant Amazon Data Firehose Access to a Public OpenSearch Serverless Destination

When you're using an OpenSearch Serverless destination, Amazon Data Firehose delivers data to your OpenSearch Serverless collection, and concurrently backs up failed or all documents to your S3 bucket. If error logging is enabled, Amazon Data Firehose also sends data delivery errors to your CloudWatch log group and streams. Amazon Data Firehose uses an IAM role to access the specified OpenSearch Serverless collection, S3 bucket, Amazon KMS key, and CloudWatch log group and streams. You are required to have an IAM role when creating a Firehose stream.

Use the following access policy to enable Amazon Data Firehose to access your S3 bucket, OpenSearch Serverless domain, and Amazon KMS key. If you do not own the S3 bucket, add `s3:PutObjectAcl` to the list of Amazon S3 actions, which grants the bucket owner full access to the objects delivered by Amazon Data Firehose. This policy also has a statement that allows access to Amazon Kinesis Data Streams. If you don't use Kinesis Data Streams as your data source, you can remove that statement.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
```

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

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

In addition to the policy above, you must also configure Amazon Data Firehose to have the following minimum permissions assigned in a data access policy:

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/target-collection/target-index"
            ],
            "Permission":[
               "aoss:WriteDocument",
               "aoss:UpdateIndex",
               "aoss:CreateIndex"
            ]
         }
      ],
```

```
      "Principal":[
         "arn:aws:sts::account-id:assumed-role/firehose-delivery-role-name/*"
      ]
   }
]
```

For more information about allowing other Amazon services to access your Amazon resources, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.

Grant Amazon Data Firehose Access to an OpenSearch Serverless Destination in a VPC

If your OpenSearch Serverless collection is in a VPC, make sure you give Amazon Data Firehose the permissions that are described in the previous section. In addition, you need to give Amazon Data Firehose the following permissions to enable it to access your OpenSearch Serverless collection's VPC.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Do not revoke these permissions after you create the delivery stream. If you revoke these permissions, your Firehose stream will be degraded or stop delivering data to your OpenSearch service domain whenever the service attempts to query or update ENIs.

> **Important**
>
> When you specify subnets for delivering data to the destination in a private VPC, make sure you have enough number of free IP addresses in chosen subnets. If there is no available free IP address in a specified subnet, Firehose cannot create or add ENIs for the data delivery in the private VPC, and the delivery will be degraded or fail.

When you create or update your delivery stream, you specify a security group for Firehose to use when it sends data to your OpenSearch Serverless collection. You can use the same security group that the OpenSearch Serverless collection uses or a different one. If you specify a different security group, ensure that it allows outbound HTTPS traffic to the OpenSearch Serverless collection's security group. Also ensure that the OpenSearch Serverless collection's security group allows HTTPS traffic from the security group you specified when you configured your Firehose stream. If you use the same security group for both your Firehose stream and the OpenSearch Serverless collection, make sure the security group inbound rule allows HTTPS traffic. For more information about security group rules, see Security group rules in the Amazon VPC documentation.

Grant Amazon Data Firehose Access to a Splunk Destination

When you're using a Splunk destination, Amazon Data Firehose delivers data to your Splunk HTTP Event Collector (HEC) endpoint. It also backs up that data to the Amazon S3 bucket that you specify, and you can optionally use an Amazon KMS key that you own for Amazon S3 server-side encryption. If error logging is enabled, Firehose sends data delivery errors to your CloudWatch log streams. You can also use Amazon Lambda for data transformation.

If you use an Amazon load balancer, make sure that it is a Classic Load Balancer or an Application Load Balancer. Also, enable duration-based sticky sessions with cookie expiration disabled for Classic Load Balancer and expiration is set to the maximum (7 days) for Application Load Balancer. For information about how to do this, see Duration-Based Session Stickiness for Classic Load Balancer or an Application Load Balancer.

You are required to have an IAM role when creating a delivery stream. Firehose assumes that IAM role and gains access to the specified bucket, key, and CloudWatch log group and streams.

Use the following access policy to enable Amazon Data Firehose to access your S3 bucket. If you don't own the S3 bucket, add `s3:PutObjectAcl` to the list of Amazon S3 actions, which grants the bucket owner full access to the objects delivered by Amazon Data Firehose. This policy also

grants Amazon Data Firehose access to CloudWatch for error logging and to Amazon Lambda for data transformation. The policy also has a statement that allows access to Amazon Kinesis Data Streams. If you don't use Kinesis Data Streams as your data source, you can remove that statement. Amazon Data Firehose doesn't use IAM to access Splunk. For accessing Splunk, it uses your HEC token.

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

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

For more information about allowing other Amazon services to access your Amazon resources, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.

Access to Splunk in VPC

If your Splunk platform is in a VPC, it must be publicly accessible with a public IP address. Also, grant Amazon Data Firehose access to your Splunk platform by unblocking the Amazon Data Firehose IP addresses. Amazon Data Firehose currently uses the following CIDR blocks.

- 18.216.68.160/27, 18.216.170.64/27, 18.216.170.96/27 for US East (Ohio)
- 34.238.188.128/26, 34.238.188.192/26, 34.238.195.0/26 for US East (N. Virginia)
- 13.57.180.0/26 for US West (N. California)
- 34.216.24.32/27, 34.216.24.192/27, 34.216.24.224/27 for US West (Oregon)
- 18.253.138.192/26 for Amazon GovCloud (US-East)
- 52.61.204.192/26 for Amazon GovCloud (US-West)
- 18.162.221.64/26 for Asia Pacific (Hong Kong)
- 13.232.67.64/26 for Asia Pacific (Mumbai)
- 13.209.71.0/26 for Asia Pacific (Seoul)
- 13.229.187.128/26 for Asia Pacific (Singapore)
- 13.211.12.0/26 for Asia Pacific (Sydney)
- 13.230.21.0/27, 13.230.21.32/27 for Asia Pacific (Tokyo)
- 51.16.102.64/26 for Israel (Tel Aviv)
- 35.183.92.64/26 for Canada (Central)
- 40.176.98.128/26 for Canada West (Calgary)
- 18.194.95.192/27, 18.194.95.224/27, 18.195.48.0/27 for Europe (Frankfurt)
- 34.241.197.32/27, 34.241.197.64/27, 34.241.197.96/27 for Europe (Ireland)
- 18.130.91.0/26 for Europe (London)
- 35.180.112.0/26 for Europe (Paris)
- 13.53.191.0/26 for Europe (Stockholm)
- 15.185.91.64/26 for Middle East (Bahrain)
- 18.228.1.192/26 for South America (São Paulo)
- 15.161.135.192/26 for Europe (Milan)
- 13.244.165.128/26 for Africa (Cape Town)
- 13.208.217.0/26 for Asia Pacific (Osaka)
- 52.81.151.64/26 for China (Beijing)
- 161.189.23.128/26 for China (Ningxia)
- 108.136.221.128/26 for Asia Pacific (Jakarta)

- `3.28.159.64/26` for Middle East (UAE)
- `51.16.102.64/26` for Israel (Tel Aviv)
- `16.62.183.64/26` for Europe (Zurich)
- `18.60.192.192/26` for Asia Pacific (Hyderabad)
- `16.50.161.192/26` for Asia Pacific (Melbourne)

Access to Snowflake or HTTP end point

There is no subset of Amazon IP address ranges specific to Amazon Data Firehose when the destination is HTTP end point or Snowflake.

To add Amazon Data Firehose to an allow list in Snowflake Network policy or to your public HTTP or HTTPS endpoints, add all the current Amazon IP address ranges to your ingress rules.

> **Note**
>
> Notifications aren't always sourced from IP addresses in the same Amazon Region as their associated topic. You must include the Amazon IP address range for all Regions.

Grant Firehose Access to a Snowflake Destination

When you're using Snowflake as a destination, Firehose delivers data to a Snowflake account using your Snowflake account URL. It also backs up error data to the Amazon Simple Storage Service bucket that you specify, and you can optionally use an Amazon Key Management Service key that you own for Amazon S3 server-side encryption. If error logging is enabled, Firehose sends data delivery errors to your CloudWatch Logs streams.

You are required to have an IAM role when creating a delivery stream. Firehose assumes that IAM role and gains access to the specified bucket, key, and CloudWatch Logs group and streams. Use the following access policy to enable Firehose to access your S3 bucket. If you don't own the S3 bucket, add `s3:PutObjectAcl` to the list of Amazon Simple Storage Service actions, which grants the bucket owner full access to the objects delivered by Firehose. This policy also grants Firehose access to CloudWatch for error logging. The policy also has a statement that allows access to Amazon Kinesis Data Streams. If you don't use Kinesis Data Streams as your data source, you can

remove that statement. Firehose doesn't use IAM to access Snowflake. For accessing Snowflake, it uses your Snowflake account Url and PrivateLink Vpce Id in the case of a private cluster.

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

```
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
"Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:*"
            ]
        }
    ]
}
```

For more information about allowing other Amazon services to access your Amazon resources, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.

Grant Amazon Data Firehose Access to an HTTP Endpoint Destination

You can use Amazon Data Firehose to deliver data to any HTTP endpoint destination. Amazon Data Firehose also backs up that data to the Amazon S3 bucket that you specify, and you can optionally use an Amazon KMS key that you own for Amazon S3 server-side encryption. If error logging is enabled, Amazon Data Firehose sends data delivery errors to your CloudWatch log streams. You can also use Amazon Lambda for data transformation.

You are required to have an IAM role when creating a Firehose stream. Amazon Data Firehose assumes that IAM role and gains access to the specified bucket, key, and CloudWatch log group and streams.

Use the following access policy to enable Amazon Data Firehose to access the S3 bucket that you specified for data backup. If you don't own the S3 bucket, add `s3:PutObjectAcl` to the list of Amazon S3 actions, which grants the bucket owner full access to the objects delivered by Amazon Data Firehose. This policy also grants Amazon Data Firehose access to CloudWatch for error logging and to Amazon Lambda for data transformation. The policy also has a statement that allows access to Amazon Kinesis Data Streams. If you don't use Kinesis Data Streams as your data source, you can remove that statement.

Important

Amazon Data Firehose doesn't use IAM to access HTTP endpoint destinations owned by supported third-party service providers, including Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Splunk, or Sumo Logic. For accessing a specified HTTP endpoint destination owned by a supported third-party service provider, contact that service provider to obtain the API key or the access key that is required to enable data delivery to that service from Amazon Data Firehose.

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

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

For more information about allowing other Amazon services to access your Amazon resources, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.

Important

Currently Amazon Data Firehose does NOT support data delivery to HTTP endpoints in a VPC.

Cross-Account Delivery from Amazon MSK

If yours is a cross-account scenario where you're creating a delivery stream from your Firehose account (for example, Account B) and your source is an MSK cluster in another Amazon account (Account A), you must have the following configurations in place:

Account A:

1. In the Amazon MSK console, choose the provisioned cluster and then choose **Properties**.
2. Under **Network settings**, choose **Edit** and turn on **Multi-VPC connectivity**.
3. Under **Security settings** choose **Edit cluster policy**.

    a. If the cluster does not already have a policy configured, check **Include Firehose service principal** and **Enable Firehose cross-account S3 delivery**. The Amazon Web Services Management Console will automatically generate a policy with the appropriate permissions.

    b. If the cluster already has a policy configured, add the following permissions to the existing policy:

```
{
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::arn:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
      ],
      "Resource": "arn:aws:kafka:us-east-1:arn:cluster/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20"  // ARN of the
 cluster
    },
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::arn:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
      ],
      "Resource": "arn:aws:kafka:us-east-1:arn:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*"//topic of the
 cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::233450236687:role/mskaasTestDeliveryRole"
      },
      "Action": "kafka-cluster:DescribeGroup",
      "Resource": "arn:aws:kafka:us-east-1:arn:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of
 the cluster
    },
 }
```

4. Under **Amazon principal**, enter the principal ID from Account B.
5. Under **Topic**, specify the Apache Kafka topic from which you want your delivery stream to ingest data. Once the delivery stream is created, you cannot update this topic.
6. Choose **Save changes**

Account B:

1. In the Firehose console, choose **Create delivery stream** using Account B.
2. Under **Source**, choose **Amazon Managed Streaming for Apache Kafka**.
3. Under **Source settings**, for the **Amazon Managed Streaming for Apache Kafka cluster**, enter the ARN of the Amazon MSK cluster in Account A.
4. Under **Topic**, specify the Apache Kafka topic from which you want your delivery stream to ingest data. Once the delivery stream is created, you cannot update this topic.
5. In **Delivery stream name** specify the name for your delivery stream.

In Account B when you're creating your delivery stream, you must have an IAM role (created by default when using the Amazon Web Services Management Console) that grants the delivery stream 'read' access to the cross-account Amazon MSK cluster for the configured topic.

The following is what gets configured by the Amazon Web Services Management Console:

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
        ],
    "Resource": "arn:aws:kafka:us-east-1:arn:cluster/DO-NOT-TOUCH-mskaas-provisioned-
privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:topic/DO-NOT-TOUCH-mskaas-provisioned-
privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/mskaas_test_topic" //topic of the
 cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeGroup"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:group/DO-NOT-TOUCH-mskaas-provisioned-
privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the cluster
    },
 }
```

Next, you can complete the optional step of configuring record transformation and record format conversion. For more information, see Record Transformation and Format Conversion.

Cross-Account Delivery to an Amazon S3 Destination

You can use the Amazon CLI or the Amazon Data Firehose APIs to create a Firehose stream in one Amazon account with an Amazon S3 destination in a different account. The following procedure shows an example of configuring a Firehose stream owned by account A to deliver data to an Amazon S3 bucket owned by account B.

1. Create an IAM role under account A using steps described in Grant Firehose Access to an Amazon S3 Destination.

   > **Note**
   >
   > The Amazon S3 bucket specified in the access policy is owned by account B in this case. Make sure you add `s3:PutObjectAcl` to the list of Amazon S3 actions in the access policy, which grants account B full access to the objects delivered by Amazon Data Firehose. This permission is required for cross account delivery. Amazon Data Firehose sets the "x-amz-acl" header on the request to "bucket-owner-full-control".

2. To allow access from the IAM role previously created, create an S3 bucket policy under account B. The following code is an example of the bucket policy. For more information, see Using Bucket Policies and User Policies.

```
{

    "Version": "2012-10-17",
    "Id": "PolicyID",
    "Statement": [
        {
            "Sid": "StmtID",
            "Effect": "Allow",
            "Principal": {
                "Amazon": "arn:aws:iam::accountA-id:role/iam-role-name"
            },
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutObjectAcl"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

3. Create a Firehose stream under account A using the IAM role that you created in step 1.

Cross-Account Delivery to an OpenSearch Service Destination

You can use the Amazon CLI or the Amazon Data Firehose APIs to create a Firehose stream in one Amazon account with an OpenSearch Service destination in a different account. The following procedure shows an example of how you can create a Firehose stream under account A and configure it to deliver data to an OpenSearch Service destination owned by account B.

1. Create an IAM role under account A using the steps described in the section called "Grant Amazon Data Firehose Access to a Public OpenSearch Service Destination".
2. To allow access from the IAM role that you created in the previous step, create an OpenSearch Service policy under account B. The following JSON is an example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Amazon": "arn:aws:iam::Account-A-ID:role/firehose_delivery_role "
      },
      "Action": "es:ESHttpGet",
      "Resource": [
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_all/
_settings",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_cluster/
stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/roletest*/
_mapping/roletest",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_nodes",
```

```
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_nodes/
stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_nodes/*/
stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/roletest*/
_stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/"
      ]
    }
  ]
}
```

3. Create a Firehose stream under account A using the IAM role that you created in step 1. When you create the Firehose stream, use the Amazon CLI or the Amazon Data Firehose APIs and specify the `ClusterEndpoint` field instead of `DomainARN` for OpenSearch Service.

> **Note**
>
> To create a Firehose stream in one Amazon account with an OpenSearch Service destination in a different account, you must use the Amazon CLI or the Amazon Data Firehose APIs. You can't use the Amazon Web Services Management Console to create this kind of cross-account configuration.

Using Tags to Control Access

You can use the optional `Condition` element (or `Condition` *block*) in an IAM policy to fine-tune access to Amazon Data Firehose operations based on tag keys and values. The following subsections describe how to do this for the different Amazon Data Firehose operations. For more on the use of the `Condition` element and the operators that you can use within it, see IAM JSON Policy Elements: Condition.

CreateDeliveryStream

For the `CreateDeliveryStream` operation, use the `aws:RequestTag` condition key. In the following example, `MyKey` and `MyValue` represent the key and corresponding value for a tag. For more information, see Tag Basics

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:CreateDeliveryStream",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/MyKey": "MyValue"
                }
            }
        }
    ]
}
```

TagDeliveryStream

For the `TagDeliveryStream` operation, use the `aws:TagKeys` condition key. In the following example, `MyKey` is an example tag key.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                 }
            }
        }
    ]
}
```

UntagDeliveryStream

For the `UntagDeliveryStream` operation, use the `aws:TagKeys` condition key. In the following example, `MyKey` is an example tag key.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:UntagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                }
            }
        }
    ]
}
```

ListDeliveryStreams

You can't use tag-based access control with `ListDeliveryStreams`.

Other Amazon Data Firehose Operations

For all Amazon Data Firehose operations other than `CreateDeliveryStream`, `TagDeliveryStream`, `UntagDeliveryStream`, and `ListDeliveryStreams`, use the `aws:RequestTag` condition key. In the following example, `MyKey` and `MyValue` represent the key and corresponding value for a tag.

`ListDeliveryStreams`, use the `firehose:ResourceTag` condition key to control access based on the tags on that Firehose stream.

In the following example, `MyKey` and `MyValue` represent the key and corresponding value for a tag. The policy would only apply to Data Firehose streams having a tag named `MyKey` with a value of `MyValue`. For more information about controlling access based on resource tags, see Controlling access to Amazon resources using tags in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
            "Effect": "Deny",
            "Action": "firehose:DescribeDeliveryStream",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                      "firehose:ResourceTag/MyKey": "MyValue"
                  }
              }
          }
      ]
}
```

Monitoring Amazon Data Firehose

Amazon Data Firehose provides monitoring functionality for your Firehose streams. For more information, see *Monitoring*.

Compliance Validation for Amazon Data Firehose

Third-party auditors assess the security and compliance of Amazon Data Firehose as part of multiple Amazon compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of Amazon services in scope of specific compliance programs, see Amazon Services in Scope by Compliance Program. For general information, see Amazon Compliance Programs.

You can download third-party audit reports using Amazon Artifact. For more information, see Downloading Reports in Amazon Artifact.

Your compliance responsibility when using Data Firehose is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. If your use of Data Firehose is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, Amazon provides resources to help:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on Amazon.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use Amazon to create HIPAA-compliant applications.
- Amazon Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Amazon Config – This Amazon service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.

- Amazon Security Hub – This Amazon service provides a comprehensive view of your security state within Amazon that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon Data Firehose

The Amazon global infrastructure is built around Amazon Regions and Availability Zones. Amazon Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about Amazon Regions and Availability Zones, see Amazon Global Infrastructure.

In addition to the Amazon global infrastructure, Data Firehose offers several features to help support your data resiliency and backup needs.

Disaster Recovery

Amazon Data Firehose runs in a serverless mode, and takes care of host degradations, Availability Zone availability, and other infrastructure related issues by performing automatic migration. When this happens, Amazon Data Firehose ensures that the Firehose stream is migrated without any loss of data.

Infrastructure Security in Amazon Data Firehose

As a managed service, Amazon Data Firehose is protected by Amazon global network security. For information about Amazon security services and how Amazon protects infrastructure, see Amazon Cloud Security. To design your Amazon environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar Amazon Well-Architected Framework*.

You use Amazon published API calls to access Firehose through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.

- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the Amazon Security Token Service (Amazon STS) to generate temporary security credentials to sign requests.

> **Note**
>
> For outgoing HTTPS requests, Amazon Data Firehose uses an HTTP library that automatically selects the highest TLS protocol version supported at the destination side.

VPC Endpoints (PrivateLink)

Amazon Data Firehose provides support for VPC endpoints (PrivateLink). For more information, see Using Amazon Data Firehose with Amazon PrivateLink.

Security Best Practices for Amazon Data Firehose

Amazon Data Firehose provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

Implement least privilege access

When granting permissions, you decide who is getting what permissions to which Amazon Data Firehose resources. You enable specific actions that you want to allow on those resources. Therefore you should grant only the permissions that are required to perform a task. Implementing least privilege access is fundamental in reducing security risk and the impact that could result from errors or malicious intent.

Use IAM roles

Producer and client applications must have valid credentials to access Amazon Data Firehose delivery streams, and your Firehose stream must have valid credentials to access destinations. You

should not store Amazon credentials directly in a client application or in an Amazon S3 bucket. These are long-term credentials that are not automatically rotated and could have a significant business impact if they are compromised.

Instead, you should use an IAM role to manage temporary credentials for your producer and client applications to access Firehose streams. When you use a role, you don't have to use long-term credentials (such as a user name and password or access keys) to access other resources.

For more information, see the following topics in the *IAM User Guide*:

- IAM Roles
- Common Scenarios for Roles: Users, Applications, and Services

Implement Server-Side Encryption in Dependent Resources

Data at rest and data in transit can be encrypted in Amazon Data Firehose. For more information, see Data Protection in Amazon Amazon Data Firehose.

Use CloudTrail to Monitor API Calls

Amazon Data Firehose is integrated with Amazon CloudTrail, a service that provides a record of actions taken by a user, role, or an Amazon service in Amazon Data Firehose.

Using the information collected by CloudTrail, you can determine the request that was made to Amazon Data Firehose, the IP address from which the request was made, who made the request, when it was made, and additional details.

For more information, see the section called “Logging Amazon Data Firehose API Calls with Amazon CloudTrail”.

Amazon Data Firehose Data Transformation

Amazon Data Firehose can invoke your Lambda function to transform incoming source data and deliver the transformed data to destinations. You can enable Amazon Data Firehose data transformation when you create your Firehose stream.

Data Transformation Flow

When you enable Firehose data transformation, Firehose buffers incoming data. The buffering size hint ranges between 0.2 MB and 3MB. The default Lambda buffering size hint is 1 MB for all destinations, except Splunk and Snowflake. For Splunk and Snowflake, the default buffering hint is 256 KB. The Lambda buffering interval hint ranges between 0 and 900 seconds. The default Lambda buffering interval hint is sixty seconds for all destinations except Snowflake. For Snowflake, the default buffering hint interval is 30 seconds. To adjust the buffering size, set the ProcessingConfiguration parameter of the CreateDeliveryStream or UpdateDestination API with the ProcessorParameter called `BufferSizeInMBs` and `IntervalInSeconds`. Firehose then invokes the specified Lambda function asynchronously with each buffered batch using the Amazon Lambda synchronous invocation mode. The transformed data is sent from Lambda to Firehose. Firehose then sends it to the destination when the specified destination buffering size or buffering interval is reached, whichever happens first.

> **Important**
>
> The Lambda synchronous invocation mode has a payload size limit of 6 MB for both the request and the response. Make sure that your buffering size for sending the request to the function is less than or equal to 6 MB. Also ensure that the response that your function returns doesn't exceed 6 MB.

Data Transformation and Status Model

All transformed records from Lambda must contain the following parameters, or Amazon Data Firehose rejects them and treats that as a data transformation failure.

For Kinesis Data Streams and Direct PUT:

recordId

The record ID is passed from Amazon Data Firehose to Lambda during the invocation. The transformed record must contain the same record ID. Any mismatch between the ID of the original record and the ID of the transformed record is treated as a data transformation failure.

result

The status of the data transformation of the record. The possible values are: `Ok` (the record was transformed successfully), `Dropped` (the record was dropped intentionally by your processing logic), and `ProcessingFailed` (the record could not be transformed). If a record has a status of `Ok` or `Dropped`, Amazon Data Firehose considers it successfully processed. Otherwise, Amazon Data Firehose considers it unsuccessfully processed.

data

The transformed data payload, after base64-encoding.

Following is a sample Lambda result output:

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

For Amazon MSK

recordId

The record ID is passed from Firehose to Lambda during the invocation. The transformed record must contain the same record ID. Any mismatch between the ID of the original record and the ID of the transformed record is treated as a data transformation failure.

result

The status of the data transformation of the record. The possible values are: `Ok` (the record was transformed successfully), `Dropped` (the record was dropped intentionally by your processing logic), and `ProcessingFailed` (the record could not be transformed). If a record has a status of `Ok` or `Dropped`, Firehose considers it successfully processed. Otherwise, Firehose considers it unsuccessfully processed.

KafkaRecordValue

The transformed data payload, after base64-encoding.

Following is a sample Lambda result output:

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

Lambda Blueprints

These blueprints demonstrate how you can create and use Amazon Lambda functions to transform data in your Amazon Data Firehose data streams.

To see the blueprints that are available in the Amazon Lambda console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.
2. Choose **Create function**, and then choose **Use a blueprint**.
3. In the **Blueprints** field, search for the keyword `firehose` to find the Amazon Data Firehose Lambda blueprints.

List of blueprints:

- **Process records sent to Amazon Data Firehose stream (Node.js, Python)**

  This blueprint shows a basic example of how to process data in your Firehose data stream using Amazon Lambda.

  Latest release date: November, 2016.

  Release notes: none.
- **Process CloudWatch logs sent to Firehose**

  This blueprint is deprecated. For information on processing CloudWatch Logs sent to Firehose, see Writing to Firehose Using CloudWatch Logs.

- **Convert Amazon Data Firehose stream records in syslog format to JSON (Node.js)**

  This blueprint shows how you can convert input records in RFC3164 Syslog format to JSON.

  Latest release date: Nov, 2016.

  Release notes: none.

To see the blueprints that are available in the Amazon Serverless Application Repository

1. Go to Amazon Serverless Application Repository.
2. Choose **Browse all applications**.
3. In the **Applications** field, search for the keyword `firehose`.

You can also create a Lambda function without using a blueprint. See Getting Started with Amazon Lambda.

Data Transformation Failure Handling

If your Lambda function invocation fails because of a network timeout or because you've reached the Lambda invocation limit, Amazon Data Firehose retries the invocation three times by default. If the invocation does not succeed, Amazon Data Firehose then skips that batch of records. The skipped records are treated as unsuccessfully processed records. You can specify or override the retry options using the CreateDeliveryStream or `UpdateDestination` API. For this type of failure, you can log invocation errors to Amazon CloudWatch Logs. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Logs.

If the status of the data transformation of a record is `ProcessingFailed`, Amazon Data Firehose treats the record as unsuccessfully processed. For this type of failure, you can emit error logs to Amazon CloudWatch Logs from your Lambda function. For more information, see Accessing Amazon CloudWatch Logs for Amazon Lambda in the *Amazon Lambda Developer Guide*.

If data transformation fails, the unsuccessfully processed records are delivered to your S3 bucket in the `processing-failed` folder. The records have the following format:

```
{
    "attemptsMade": "count",
    "arrivalTimestamp": "timestamp",
```

```
    "errorCode": "code",
    "errorMessage": "message",
    "attemptEndingTimestamp": "timestamp",
    "rawData": "data",
    "lambdaArn": "arn"
}
```

`attemptsMade`

The number of invocation requests attempted.

`arrivalTimestamp`

The time that the record was received by Amazon Data Firehose.

`errorCode`

The HTTP error code returned by Lambda.

`errorMessage`

The error message returned by Lambda.

`attemptEndingTimestamp`

The time that Amazon Data Firehose stopped attempting Lambda invocations.

`rawData`

The base64-encoded record data.

`lambdaArn`

The Amazon Resource Name (ARN) of the Lambda function.

Duration of a Lambda Invocation

Amazon Data Firehose supports a Lambda invocation time of up to 5 minutes. If your Lambda function takes more than 5 minutes to complete, you get the following error: Firehose encountered timeout errors when calling Amazon Lambda. The maximum supported function timeout is 5 minutes.

For information about what Amazon Data Firehose does if such an error occurs, see the section called “Data Transformation Failure Handling”.

Source Record Backup

Amazon Data Firehose can back up all untransformed records to your S3 bucket concurrently while delivering transformed records to the destination. You can enable source record backup when you create or update your Firehose stream. You cannot disable source record backup after you enable it.

Dynamic Partitioning in Amazon Data Firehose

Dynamic partitioning enables you to continuously partition streaming data in Firehose by using keys within data (for example, `customer_id` or `transaction_id`) and then deliver the data grouped by these keys into corresponding Amazon Simple Storage Service (Amazon S3) prefixes. This makes it easier to run high performance, cost-efficient analytics on streaming data in Amazon S3 using various services such as Amazon Athena, Amazon EMR, Amazon Redshift Spectrum, and Amazon QuickSight. In addition, Amazon Glue can perform more sophisticated extract, transform, and load (ETL) jobs after the dynamically partitioned streaming data is delivered to Amazon S3, in use-cases where additional processing is required.

Partitioning your data minimizes the amount of data scanned, optimizes performance, and reduces costs of your analytics queries on Amazon S3. It also increases granular access to your data. Firehose streams are traditionally used in order to capture and load data into Amazon S3. To partition a streaming data set for Amazon S3-based analytics, you would need to run partitioning applications between Amazon S3 buckets prior to making the data available for analysis, which could become complicated or costly.

With dynamic partitioning, Firehose continuously groups in-transit data using dynamically or statically defined data keys, and delivers the data to individual Amazon S3 prefixes by key. This reduces time-to-insight by minutes or hours. It also reduces costs and simplifies architectures.

Topics

Partitioning keys

With dynamic partitioning, you create targeted data sets from the streaming S3 data by partitioning the data based on partitioning keys. Partitioning keys enable you to filter your

streaming data based on specific values. For example, if you need to filter your data based on customer ID and country, you can specify the data field of `customer_id` as one partitioning key and the data field of `country` as another partitioning key. Then, you specify the expressions (using the supported formats) to define the S3 bucket prefixes to which the dynamically partitioned data records are to be delivered.

The following are the supported methods of creating partitioning keys:

- **Inline parsing** - this method uses Firehose built-in support mechanism, a jq parser, for extracting the keys for partitioning from data records that are in JSON format. Currently, we only support `jq 1.6` version.
- **Amazon Lambda function** - this method uses a specified Amazon Lambda function to extract and return the data fields needed for partitioning.

> **Important**
>
> When you enable dynamic partitioning, you must configure at least one of these methods to partition your data. You can configure either of these methods to specify your partitioning keys or both of them at the same time.

Creating partitioning keys with inline parsing

To configure inline parsing as the dynamic partitioning method for your streaming data, you must choose data record parameters to be used as partitioning keys and provide a value for each specified partitioning key.

Let's look at the following sample data record and see how you can define partitioning keys for it with inline parsing:

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

For example, you can choose to partition your data based on the `customer_id` parameter or the `event_timestamp` parameter. This means that you want the value of the `customer_id` parameter or the `event_timestamp` parameter in each record to be used in determining the S3 prefix to which the record is to be delivered. You can also choose a nested parameter, like `device` with an expression `.type.device`. Your dynamic partitioning logic can depend on multiple parameters.

After selecting data parameters for your partitioning keys, you then map each parameter to a valid jq expression. The following table shows such a mapping of parameters to jq expressions:

Parameter	jq expression
`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")

At runtime, Firehose uses the right column above to evaluate the parameters based on the data in each record.

Creating partitioning keys with an Amazon Lambda function

For compressed or encrypted data records, or data that is in any file format other than JSON, you can use the integrated Amazon Lambda function with your own custom code to decompress, decrypt, or transform the records in order to extract and return the data fields needed for partitioning. This is an expansion of the existing transform Lambda function that is available today with Firehose. You can transform, parse and return the data fields that you can then use for dynamic partitioning using the same Lambda function.

The following is an example Firehose stream processing Lambda function in Python that replays every read record from input to output and extracts partitioning keys from the records.

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

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

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

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

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

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

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

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

        firehose_records_output['records'].append(firehose_record_output)

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

The following is an example Firehose stream processing Lambda function in Go that replays every read record from input to output and extracts partitioning keys from the records.

```
package main

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

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

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

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

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

 var response events.DataFirehoseResponse

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

  var transformedRecord events.DataFirehoseResponseRecord
```

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

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

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

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

 return response, nil
}

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

Amazon S3 Bucket Prefix for Dynamic Partitioning

When you create a Firehose stream that uses Amazon S3 as the destination, you must specify an Amazon S3 bucket where Firehose is to deliver your data. Amazon S3 bucket prefixes are used to organize the data that you store in your S3 buckets. An Amazon S3 bucket prefix is similar to a directory that enables you to group similar objects together.

With dynamic partitioning, your partitioned data is delivered into the specified Amazon S3 prefixes. If you don't enable dynamic partitioning, specifying an S3 bucket prefix for your Firehose stream is optional. However, if you choose to enable dynamic partitioning, you must specify the S3 bucket prefixes to which Firehose delivers partitioned data.

In every Firehose stream where you enable dynamic partitioning, the S3 bucket prefix value consists of expressions based on the specified partitioning keys for that delivery stream. Using the above data record example again, you can build the following S3 prefix value that consists of expressions based on the partitioning keys defined above:

```
"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 evaluates the above expression at runtime. It groups records that match the same evaluated S3 prefix expression into a single data set. Firehose then delivers each data set to the evaluated S3 prefix. The frequency of data set delivery to S3 is determined by the Firehose stream buffer setting. As a result, the record in this example is delivered to the following S3 object key:

```
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
```

For dynamic partitioning, you must use the following expression format in your S3 bucket prefix: `!{namespace:value}`, where namespace can be either `partitionKeyFromQuery` or `partitionKeyFromLambda`, or both. If you are using inline parsing to create the partitioning keys for your source data, you must specify an S3 bucket prefix value that consists of expressions specified in the following format: `"partitionKeyFromQuery:keyID"`. If you are using an Amazon Lambda function to create partitioning keys for your source data, you must specify an S3 bucket prefix value that consists of expressions specified in the following format: `"partitionKeyFromLambda:keyID"`.

Note

You can also specify the S3 bucket prefix value using the hive style format, for example customer_id=!{partitionKeyFromQuery:customer_id}.

For more information, see the "Choose Amazon S3 for Your Destination" in Creating an Amazon Firehose stream and Custom Prefixes for Amazon S3 Objects.

Dynamic partitioning of aggregated data

You can apply dynamic partitioning to aggregated data (for example, multiple events, logs, or records aggregated into a single `PutRecord` and `PutRecordBatch` API call) but this data must first be deaggregated. You can deaggregate your data by enabling multi record deaggregation - the process of parsing through the records in the Firehose stream and separating them. Multi record deaggregation can either be of `JSON` type, meaning that the separation of records is performed based on valid JSON. Or it can be of the `Delimited` type, meaning that the separation of records is performed based on a specified custom delimiter. This custom delimiter must be a base-64 encoded string. For example, if you want to use the following string as your custom delimiter `####`, you must specify it in the base-64 encoded format, which translates it to `IyMjIw==`.

With aggregated data, when you enable dynamic partitioning, Firehose parses the records and looks for either valid JSON objects or delimited records within each API call based on the specified multi record deaggregation type.

Important

If your data is aggregated, dynamic partitioning can be only be applied if your data is first deaggregated.

Important

When you use Data Transformation feature in Firehose, the deaggregation will be applied before the Data Transformation. Data coming into Firehose will be processed in the following order: Deaggregation → Data Transformation via Lambda → Partitioning Keys.

Adding a new line delimiter when delivering data to S3

You can enable **New Line Delimiter** to add a new line delimiter between records in objects that are delivered to Amazon S3. This can be helpful for parsing objects in Amazon S3. This is also particularly useful when dynamic partitioning is applied to aggregated data because multi-record deaggregation (which must be applied to aggregated data before it can be dynamically partitioned) removes new lines from records as part of the parsing process.

How to enable dynamic partitioning

You can configure dynamic partitioning for your Firehose streams through the Amazon Data Firehose Management Console, CLI, or the APIs.

> **Important**
>
> You can enable dynamic partitioning only when you create a new Firehose stream. You cannot enable dynamic partitioning for an existing Firehose stream that does not have dynamic partitioning already enabled.

For detailed steps on how to enable and configure dynamic partitioning through the Firehose management console while creating a new Firehose stream, see Creating an Amazon Firehose stream. When you get to the task of specifying the destination for your Firehose stream, make sure to follow the steps in the Choose Amazon S3 for Your Destination section, since currently, dynamic partitioning is only supported for Firehose streams that use Amazon S3 as the destination.

Once dynamic partitioning on an active Firehose stream is enabled, you can update the configuration by adding new or removing or updating existing partitioning keys and the S3 prefix expressions. Once updated, Firehose starts using the new keys and the new S3 prefix expressions.

> **Important**
>
> Once you enable dynamic partitioning on a Firehose stream, it cannot be disabled on this Firehose stream.

Dynamic Partitioning Error Handling

If Amazon Data Firehose is not able to parse data records in your Firehose stream or it fails to extract the specified partitioning keys, or to evaluate the expressions included in the S3 prefix value, these data records are delivered to the S3 error bucket prefix that you must specify when you create the Firehose stream where you enable dynamic partitioning. The S3 error bucket prefix contains all the records that Firehose is not able to deliver to the specified S3 destination. These records are organized based on the error type. Along with the record, the delivered object also includes information about the error to help understand and resolve the error.

You must specify an S3 error bucket prefix for a Firehose stream if you want to enable dynamic partitioning for this Firehose stream. If you don't want to enable dynamic partitioning for a Firehose stream, specifying an S3 error bucket prefix is optional.

Data buffering and dynamic partitioning

Amazon Data Firehose buffers incoming streaming data to a certain size and for a certain period of time before delivering it to the specified destinations. You can configure the buffer size and the buffer interval while creating new Firehose streams or update the buffer size and the buffer interval on your existing Firehose streams. A buffer size is measured in MBs and a buffer interval is measured in seconds.

When dynamic partitioning is enabled, Firehose internally buffers records that belong to a given partition based on the configured buffering hint (size and time) before delivering these records to your Amazon S3 bucket. In order to deliver maximum size objects, Firehose uses multi-stage buffering internally. Therefore, end-to-end delay of a batch of records might be 1.5 times of the configured buffering hint time. This affects the data freshness of a Firehose stream.

The active partition count is the total number of active partitions within the delivery buffer. For example, if the dynamic partitioning query constructs 3 partitions per second and you have a buffer hint configuration triggering delivery every 60 seconds, then on average you would have 180 active partitions. If Firehose cannot deliver the data in a partition to a destination, this partition is counted as active in the delivery buffer until it can be delivered.

A new partition is created when an S3 prefix is evaluated to a new value based on the record data fields and the S3 prefix expressions. A new buffer is created for each active partition. Every subsequent record with the same evaluated S3 prefix is delivered to that buffer. Once the buffer meets the buffer size limit or the buffer time interval, Firehose creates an object with the buffer

data and delivers it to the specified Amazon S3 prefix. Once the object is delivered, the buffer for that partition and the partition itself are deleted and removed from the active partitions count. Firehose delivers each buffer data as a single object once the buffer size or interval are met for each partition separately. Once the number of active partitions reaches the limit of 500 per deliver stream, the rest of the records in the Firehose stream are delivered to the specified S3 error bucket prefix.

Converting Your Input Record Format in Firehose

Amazon Data Firehose can convert the format of your input data from JSON to Apache Parquet or Apache ORC before storing the data in Amazon S3. Parquet and ORC are columnar data formats that save space and enable faster queries compared to row-oriented formats like JSON. If you want to convert an input format other than JSON, such as comma-separated values (CSV) or structured text, you can use Amazon Lambda to transform it to JSON first. For more information, see *Data Transformation*.

Topics

Record Format Conversion Requirements

Amazon Data Firehose requires the following three elements to convert the format of your record data:

- **A deserializer to read the JSON of your input data** – You can choose one of two types of deserializers: Apache Hive JSON SerDe or OpenX JSON SerDe.

> **Note**
>
> When combining multiple JSON documents into the same record, make sure that your input is still presented in the supported JSON format. An array of JSON documents is NOT a valid input.
> For example, this is the correct input: `{"a":1}{"a":2}`
> And this is the INCORRECT input: `[{"a":1}, {"a":2}]`

- **A schema to determine how to interpret that data** – Use Amazon Glue to create a schema in the Amazon Glue Data Catalog. Amazon Data Firehose then references that schema and uses it to interpret your input data. You can use the same schema to configure both Amazon Data Firehose and your analytics software. For more information, see Populating the Amazon Glue Data Catalog in the *Amazon Glue Developer Guide*.

  > **Note**
  >
  > The schema created in Amazon Glue Data Catalog should match the input data structure. Otherwise, the converted data will not contain attributes that are not specified in the schema. If you use nested JSON, use a STRUCT type in the schema that mirrors the structure of your JSON data. See this example for how to handle nested JSON with a STRUCT type.

- **A serializer to convert the data to the target columnar storage format (Parquet or ORC)** – You can choose one of two types of serializers: ORC SerDe or Parquet SerDe.

> **Important**
>
> If you enable record format conversion, you can't set your Amazon Data Firehose destination to be Amazon OpenSearch Service, Amazon Redshift, or Splunk. With format conversion enabled, Amazon S3 is the only destination that you can use for your Firehose stream.

You can convert the format of your data even if you aggregate your records before sending them to Amazon Data Firehose.

Choosing the JSON Deserializer

Choose the OpenX JSON SerDe if your input JSON contains time stamps in the following formats:

- yyyy-MM-dd'T'HH:mm:ss[.S]'Z', where the fraction can have up to 9 digits – For example, `2017-02-07T15:13:01.39256Z`.
- yyyy-[M]M-[d]d HH:mm:ss[.S], where the fraction can have up to 9 digits – For example, `2017-02-07 15:13:01.14`.
- Epoch seconds – For example, `1518033528`.

- Epoch milliseconds – For example, `1518033528123`.
- Floating point epoch seconds – For example, `1518033528.123`.

The OpenX JSON SerDe can convert periods (`.`) to underscores (_). It can also convert JSON keys to lowercase before deserializing them. For more information about the options that are available with this deserializer through Amazon Data Firehose, see OpenXJsonSerDe.

If you're not sure which deserializer to choose, use the OpenX JSON SerDe, unless you have time stamps that it doesn't support.

If you have time stamps in formats other than those listed previously, use the Apache Hive JSON SerDe. When you choose this deserializer, you can specify the time stamp formats to use. To do this, follow the pattern syntax of the Joda-Time `DateTimeFormat` format strings. For more information, see Class DateTimeFormat.

You can also use the special value `millis` to parse time stamps in epoch milliseconds. If you don't specify a format, Amazon Data Firehose uses `java.sql.Timestamp::valueOf` by default.

The Hive JSON SerDe doesn't allow the following:

- Periods (`.`) in column names.
- Fields whose type is `uniontype`.
- Fields that have numerical types in the schema, but that are strings in the JSON. For example, if the schema is (an int), and the JSON is `{"a":"123"}`, the Hive SerDe gives an error.

The Hive SerDe doesn't convert nested JSON into strings. For example, if you have `{"a": {"inner":1}}`, it doesn't treat `{"inner":1}` as a string.

Choosing the Serializer

The serializer that you choose depends on your business needs. To learn more about the two serializer options, see ORC SerDe and Parquet SerDe.

Converting Input Record Format (Console)

You can enable data format conversion on the console when you create or update a Firehose stream. With data format conversion enabled, Amazon S3 is the only destination that you

can configure for the Firehose stream. Also, Amazon S3 compression gets disabled when you enable format conversion. However, Snappy compression happens automatically as part of the conversion process. The framing format for Snappy that Amazon Data Firehose uses in this case is compatible with Hadoop. This means that you can use the results of the Snappy compression and run queries on this data in Athena. For the Snappy framing format that Hadoop relies on, see BlockCompressorStream.java.

To enable data format conversion for a data Firehose stream

1. Sign in to the Amazon Web Services Management Console, and open the Amazon Data Firehose console at https://console.amazonaws.cn/firehose/.
2. Choose a Firehose stream to update, or create a new Firehose stream by following the steps in Creating a Firehose stream.
3. Under **Convert record format**, set **Record format conversion** to **Enabled**.
4. Choose the output format that you want. For more information about the two options, see Apache Parquet and Apache ORC.
5. Choose an Amazon Glue table to specify a schema for your source records. Set the Region, database, table, and table version.

Converting Input Record Format (API)

If you want Amazon Data Firehose to convert the format of your input data from JSON to Parquet or ORC, specify the optional DataFormatConversionConfiguration element in ExtendedS3DestinationConfiguration or in ExtendedS3DestinationUpdate. If you specify DataFormatConversionConfiguration, the following restrictions apply:

- In BufferingHints, you can't set `SizeInMBs` to a value less than 64 if you enable record format conversion. Also, when format conversion isn't enabled, the default value is 5. The value becomes 128 when you enable it.
- You must set `CompressionFormat` in ExtendedS3DestinationConfiguration or in ExtendedS3DestinationUpdate to `UNCOMPRESSED`. The default value for `CompressionFormat` is `UNCOMPRESSED`. Therefore, you can also leave it unspecified in ExtendedS3DestinationConfiguration. The data still gets compressed as part of the serialization process, using Snappy compression by default. The framing format for Snappy that Amazon Data Firehose uses in this case is compatible with Hadoop. This means that you can use the results of the Snappy compression and run queries on this data in Athena. For the Snappy framing format

that Hadoop relies on, see BlockCompressorStream.java. When you configure the serializer, you can choose other types of compression.

Record Format Conversion Error Handling

When Amazon Data Firehose can't parse or deserialize a record (for example, when the data doesn't match the schema), it writes it to Amazon S3 with an error prefix. If this write fails, Amazon Data Firehose retries it forever, blocking further delivery. For each failed record, Amazon Data Firehose writes a JSON document with the following schema:

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

Record Format Conversion Example

For an example of how to set up record format conversion with Amazon CloudFormation, see Amazon::DataFirehose::DeliveryStream.

Using Amazon Managed Service for Apache Flink

With Amazon Managed Service for Apache Flink, you can use Java, Scala, or SQL to process and analyze streaming data. The service enables you to author and run code against streaming sources to perform time-series analytics, feed real-time dashboards, and create real-time metrics.

For an example of integrating with Amazon Managed Service for Apache Flink, see Example: Writing to Amazon Data Firehose.

In this exercise, you create an Apache Flink application that has a Kinesis data stream as a source and a Firehose stream as a sink. Using the sink, you can verify the output of the application in an Amazon S3 bucket.

Before you begin, set up the required prerequisites:

- Components of Managed Service for Apache Flink Application
- Prerequisites for Completing the Exercise

Amazon Data Firehose Data Delivery

After data is sent to your Firehose stream, it is automatically delivered to the destination you choose.

> **Important**
>
> If you use the Kinesis Producer Library (KPL) to write data to a Kinesis data stream, you can use aggregation to combine the records that you write to that Kinesis data stream. If you then use that data stream as a source for your Firehose stream, Amazon Data Firehose de-aggregates the records before it delivers them to the destination. If you configure your Firehose stream to transform the data, Amazon Data Firehose de-aggregates the records before it delivers them to Amazon Lambda. For more information, see Developing Amazon Kinesis Data Streams Producers Using the Kinesis Producer Library and Aggregation in the *Amazon Kinesis Data Streams Developer Guide*.

Topics

Data Delivery Format

For data delivery to Amazon Simple Storage Service (Amazon S3), Firehose concatenates multiple incoming records based on the buffering configuration of your delivery stream. It then delivers the records to Amazon S3 as an Amazon S3 object. By default, Firehose concatenates data without any delimiters. If you want to have new line delimiters between records, you can add new line delimiters by enabling the feature in the Firehoseconsole configuration or API parameter.

For data delivery to Amazon Redshift, Firehose first delivers incoming data to your S3 bucket in the format described earlier. Firehose then issues an Amazon Redshift **COPY** command to load the data from your S3 bucket to your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup. Ensure that after Amazon Data Firehose concatenates multiple incoming records to an Amazon S3 object, the Amazon S3 object can be copied to your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup. For more information, see Amazon Redshift COPY Command Data Format Parameters.

For data delivery to OpenSearch Service and OpenSearch Serverless, Amazon Data Firehose buffers incoming records based on the buffering configuration of your Firehose stream. It then generates an OpenSearch Service or OpenSearch Serverless bulk request to index multiple records to your OpenSearch Service cluster or OpenSearch Serverless collection. Make sure that your record is UTF-8 encoded and flattened to a single-line JSON object before you send it to Amazon Data Firehose. Also, the `rest.action.multi.allow_explicit_index` option for your OpenSearch Service cluster must be set to true (default) to take bulk requests with an explicit index that is set per record. For more information, see OpenSearch Service Configure Advanced Options in the *Amazon OpenSearch Service Developer Guide*.

For data delivery to Splunk, Amazon Data Firehose concatenates the bytes that you send. If you want delimiters in your data, such as a new line character, you must insert them yourself. Make sure that Splunk is configured to parse any such delimiters.

When delivering data to an HTTP endpoint owned by a supported third-party service provider, you can use the integrated Amazon Lambda service to create a function to transform the incoming record(s) to the format that matches the format the service provider's integration is expecting. Contact the third-party service provider whose HTTP endpoint you've chosen for your destination to learn more about their accepted record format.

Data Delivery Frequency

Each Firehose destination has its own data delivery frequency. For more information, see Buffering hints.

Data Delivery Failure Handling

Each Amazon Data Firehose destination has its own data delivery failure handling.

Amazon S3

Data delivery to your S3 bucket might fail for various reasons. For example, the bucket might not exist anymore, the IAM role that Amazon Data Firehose assumes might not have access to the bucket, the network failed, or similar events. Under these conditions, Amazon Data Firehose keeps retrying for up to 24 hours until the delivery succeeds. The maximum data storage time of Amazon Data Firehose is 24 hours. If data delivery fails for more than 24 hours, your data is lost.

Amazon Redshift

For an Amazon Redshift destination, you can specify a retry duration (0–7200 seconds) when creating a Firehose stream.

Data delivery to your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup might fail for several reasons. For example, you might have an incorrect cluster configuration of your Firehose stream, a cluster or workgroup under maintenance, or a network failure. Under these conditions, Amazon Data Firehose retries for the specified time duration and skips that particular batch of Amazon S3 objects. The skipped objects' information is delivered to your S3 bucket as a manifest file in the `errors/` folder, which you can use for manual backfill. For information about how to COPY data manually with manifest files, see Using a Manifest to Specify Data Files.

Amazon OpenSearch Service and OpenSearch Serverless

For the OpenSearch Service and OpenSearch Serverless destination, you can specify a retry duration (0–7200 seconds) when creating a delivery stream.

Data delivery to your OpenSearch Service cluster or OpenSearch Serverless collection might fail for several reasons. For example, you might have an incorrect OpenSearch Service cluster or OpenSearch Serverless collection configuration of your Firehose stream, an OpenSearch Service cluster or OpenSearch Serverless collection under maintenance, a network failure, or similar events. Under these conditions, Amazon Data Firehose retries for the specified time duration and then skips that particular index request. The skipped documents are delivered to your S3 bucket in the `AmazonOpenSearchService_failed/` folder, which you can use for manual backfill.

For OpenSearch Service, each document has the following 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)"
}
```

For OpenSearch Serverless, each document has the following 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

When Amazon Data Firehose sends data to Splunk, it waits for an acknowledgment from Splunk. If an error occurs, or the acknowledgment doesn’t arrive within the acknowledgment timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time Amazon Data Firehose sends data to Splunk, whether it's the initial attempt or a retry, it restarts the acknowledgement timeout counter. It then waits for an acknowledgement to arrive from Splunk. Even if the retry duration expires, Amazon Data Firehose still waits for the acknowledgment until it receives it or the acknowledgement timeout is reached. If the acknowledgment times out, Amazon Data Firehose checks to determine whether there's time

left in the retry counter. If there is time left, it retries again and repeats the logic until it receives an acknowledgment or determines that the retry time has expired.

A failure to receive an acknowledgement isn't the only type of data delivery error that can occur. For information about the other types of data delivery errors, see Splunk Data Delivery Errors. Any data delivery error triggers the retry logic if your retry duration is greater than 0.

The following is an example error record.

```
{
  "attemptsMade": 0,
  "arrivalTimestamp": 1506035354675,
  "errorCode": "Splunk.AckTimeout",
  "errorMessage": "Did not receive an acknowledgement from HEC before the HEC
 acknowledgement timeout expired. Despite the acknowledgement timeout, it's possible
 the data was indexed successfully in Splunk. Amazon Data Firehose backs up in
 Amazon S3 data for which the acknowledgement timeout expired.",
  "attemptEndingTimestamp": 13626284715507,
  "rawData":
"MiAyNTE2MjAyNzIyMDkgZW5pLTA1ZjMyMmQ1IDIxOC45Mi4xODguMjE0IDE3Mi4xNi4xLjE2NyAyNTIzMyAxNDMzID
  "EventId": "49577193928114147339600778471082492393164139877200035842.0"
}
```

HTTP endpoint destination

When Amazon Data Firehose sends data to an HTTP endpoint destination, it waits for a response from this destination. If an error occurs, or the response doesn't arrive within the response timeout period, Amazon Data Firehose starts the retry duration counter. It keeps retrying until the retry duration expires. After that, Amazon Data Firehose considers it a data delivery failure and backs up the data to your Amazon S3 bucket.

Every time Amazon Data Firehose sends data to an HTTP endpoint destination, whether it's the initial attempt or a retry, it restarts the response timeout counter. It then waits for a response to arrive from the HTTP endpoint destination. Even if the retry duration expires, Amazon Data Firehose still waits for the response until it receives it or the response timeout is reached. If the response times out, Amazon Data Firehose checks to determine whether there's time left in the retry counter. If there is time left, it retries again and repeats the logic until it receives a response or determines that the retry time has expired.

A failure to receive a response isn't the only type of data delivery error that can occur. For information about the other types of data delivery errors, see HTTP Endpoint Data Delivery Errors

The following is an example error record.

```
{
 "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"
}
```

Amazon S3 Object Name Format

When Firehose delivers data to Amazon S3, S3 object key name follows the format *<evaluated prefix><suffix>*, where the suffix has the format *<delivery stream name>-<delivery stream version>-<year>-<month>-<day>-<hour>-<minute>-<second>-<uuid><file extension> <delivery stream version>* begins with 1 and increases by 1 for every configuration change of the Firehose delivery stream. You can change delivery stream configurations (for example, the name of the S3 bucket, buffering hints, compression, and encryption). You can do so by using the Firehose console or the UpdateDestination API operation.

For *<evaluated prefix>*, Firehose adds a default time prefix in the format YYYY/MM/dd/HH. This prefix creates a logical hierarchy in the bucket, where each forward slash (/) creates a level in the hierarchy. You can modify this structure by specifying a custom prefix that includes expressions that are evaluated at runtime. For information about how to specify a custom prefix, see Custom Prefixes for Amazon Simple Storage Service Objects.

By default, time zone used for the time prefix is UTC, but you can change it to a time zone you prefer. For example, you can configure the time zone to Asia/Tokyo in the Amazon Web Services

Management Console or in API parameter setting (CustomTimeZone) if you want to use Japan Standard Time instead of UTC. The following list contains time zones supported for S3 prefix configuration in Firehose:

Time zones supported for S3 prefix configuration in Firehose

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
```

You cannot change the suffix field except *<file extension>*. When you enable data format conversion or compression, Firehose will append a file extension based on the configuration. The following table explains the default file extension appended by Firehose:

Configuration	File extension
Data Format Conversion: Parquet	.parquet
Data Format Conversion: ORC	.orc
Compression: Gzip	.gz
Compression: Zip	.zip

Configuration	File extension
Compression: Snappy	.snappy
Compression: Hadoop-Snappy	.hsnappy

You can also specify a file extension that you prefer in the Firehose console or API. File extension must start with a period (.) and can contain allowed characters: 0-9a-z!-_.*'(). File extension cannot exceed 128 characters.

> **Note**
>
> When you specify a file extension, it will override the default file extension that Firehose adds when data format conversion or compression is enabled.

Index Rotation for the OpenSearch Service Destination

For the OpenSearch Service destination, you can specify a time-based index rotation option from one of the following five options: **NoRotation**, **OneHour**, **OneDay**, **OneWeek**, or **OneMonth**.

Depending on the rotation option you choose, Amazon Data Firehose appends a portion of the UTC arrival timestamp to your specified index name. It rotates the appended timestamp accordingly. The following example shows the resulting index name in OpenSearch Service for each index rotation option, where the specified index name is **myindex** and the arrival timestamp is `2016-02-25T13:00:00Z`.

RotationPeriod	IndexName
`NoRotation`	`myindex`
`OneHour`	`myindex-2016-02-25-13`
`OneDay`	`myindex-2016-02-25`
`OneWeek`	`myindex-2016-w08`

RotationPeriod	IndexName
OneMonth	myindex-2016-02

Note

With the OneWeek option, Data Firehose auto-create indexes using the format of <YEAR>-w<WEEK NUMBER> (for example, 2020-w33), where the week number is calculated using UTC time and according to the following US conventions:

- A week starts on Sunday
- The first week of the year is the first week that contains a Saturday in this year

Delivery Across Amazon Accounts and Across Amazon Regions for HTTP Endpoint Destinations

Amazon Data Firehose supports data delivery to HTTP endpoint destinations across Amazon accounts. Amazon Data Firehose Firehose stream and the HTTP endpoint that you've chosen as your destination can be in different Amazon accounts.

Amazon Data Firehose also supports data delivery to HTTP endpoint destinations across Amazon regions. You can deliver data from a Firehose stream in one Amazon region to an HTTP endpoint in another Amazon region. You can also delivery data from a Firehose stream to an HTTP endpoint destination outside of Amazon regions, for example to your own on-premises server by setting the HTTP endpoint URL to your desired destination. For these scenarios, additional data transfer charges are added to your delivery costs. For more information, see the Data Transfer section in the "On-Demand Pricing" page.

Duplicated Records

Amazon Data Firehose uses at-least-once semantics for data delivery. In some circumstances, such as when data delivery times out, delivery retries by Amazon Data Firehose might introduce duplicates if the original data-delivery request eventually goes through. This applies to all destination types that Amazon Data Firehose supports.

How to Pause and Resume a Firehose delivery stream

After you setup a delivery stream in Firehose, data available in the stream source is continuously delivered to the destination. If you encounter situations where your stream destination is temporarily unavailable (for example, during planned maintenance operations), you may want to temporarily pause data delivery, and resume when the destination becomes available again. The following sections show how you can accomplish this:

Important

When you use the approach described below to pause and resume a stream, after you resume the stream, you will see that few records get delivered to the error bucket in Amazon S3 while the rest of the stream continues to get delivered to the destination. This is a known limitation of the approach, and it occurs because a small number of records that could not be previously delivered to the destination after multiple retries are tracked as failed.

Understanding how Firehose handles delivery failures

When you setup a delivery stream in Firehose, for many destinations such as OpenSearch, Splunk, and HTTP endpoints, you also setup an S3 bucket where data that fails to be delivered can be backed up. For more information about how Firehose backs up data in case of failed deliveries, see Data Delivery Failure Handling. For more information about how to grant access to S3 buckets where data that fails to be delivered can be backed up, see Grant Firehose Access to an Amazon S3 Destination. When Firehose (a) fails to deliver data to the stream destination, and (b) fails to write data to the backup S3 bucket for failed deliveries, it effectively pauses stream delivery until such time that data can either be delivered to the destination or written to the backup S3 location.

Pausing a Firehose delivery stream

To pause stream delivery in Firehose, first remove permissions for Firehose to write to the S3 backup location for failed deliveries. For example, if you want to pause the delivery stream with an OpenSearch destination, you can do this by updating permissions. For more information, see Grant Firehose Access to a Public OpenSearch Service Destination.

Remove the `"Effect": "Allow"` permission for the action `s3:PutObject`, and explicitly add a statement that applies `Effect": "Deny"` permission on the action `s3:PutObject` for the S3

bucket used for backing up failed deliveries. Next, turn off the stream destination (for example, turning off the destination OpenSearch domain), or remove permissions for Firehose to write to the destination. To update permissions for other destinations, check the section for your destination in Controlling Access with Amazon Data Firehose. After you complete these two actions, Firehose will stop delivering streams, and you can monitor this using CloudWatch metrics for Firehose.

> **Important**
>
> When you pause stream delivery in Firehose, you need to ensure that the source of the stream (for example, in Kinesis Data Streams or in Managed Service for Kafka) is configured to retain data until stream delivery is resumed and the data gets delivered to the destination. If the source is DirectPUT, Firehose will retain data for 24 hours. Data loss could happen if you do not resume the stream and deliver the data before the expiration of data retention period.

Resuming a Firehose delivery stream

To resume delivery, first revert the change made earlier to the stream destination by turning on the destination and ensuring that Firehose has permissions to deliver the stream to the destination. Next, revert the changes made earlier to permissions applied to the S3 bucket for backing up failed deliveries. That is, apply `"Effect": "Allow"` permission for the action `s3:PutObject`, and remove `"Effect": "Deny"` permission on the action `s3:PutObject` for the S3 bucket used for backing up failed deliveries. Finally, monitor using CloudWatch metrics for Firehose to confirm that the stream is being delivered to the destination. To view and troubleshoot errors, use Amazon CloudWatch Logs monitoring for Firehose.

Monitoring Amazon Data Firehose

You can monitor Amazon Data Firehose using the following features:

Topics

Best Practices with CloudWatch Alarms

Add CloudWatch alarms for when the following metrics exceed the buffering limit (a maximum of 15 minutes):

- `DeliveryToS3.DataFreshness`
- `DeliveryToSplunk.DataFreshness`
- `DeliveryToAmazonOpenSearchService.DataFreshness`
- `DeliveryToAmazonOpenSearchServerless.DataFreshness`
- `DeliveryToHttpEndpoint.DataFreshness`

Also, create alarms based on the following metric math expressions.

- `IncomingBytes (Sum per 5 Minutes) / 300` approaches a percentage of `BytesPerSecondLimit`.
- `IncomingRecords (Sum per 5 Minutes) / 300` approaches a percentage of `RecordsPerSecondLimit`.
- `IncomingPutRequests (Sum per 5 Minutes) / 300` approaches a percentage of `PutRequestsPerSecondLimit`.

Another metric for which we recommend an alarm is `ThrottledRecords`.

For information about troubleshooting when alarms go to the `ALARM` state, see *Troubleshooting*.

Monitoring Amazon Data Firehose Using CloudWatch Metrics

> **Important**
>
> Be sure to enable alarms on all CloudWatch metrics that belong to your destination in order to identify errors in timely manner.

Amazon Data Firehose integrates with Amazon CloudWatch metrics so that you can collect, view, and analyze CloudWatch metrics for your Firehose streams. For example, you can monitor the `IncomingBytes` and `IncomingRecords` metrics to keep track of data ingested into Amazon Data Firehose from data producers.

Amazon Data Firehose collects and publishes CloudWatch metrics every minute. However, if bursts of incoming data occur only for a few seconds, they may not be fully captured or visible in the one-minute metrics. This is because CloudWatch metrics are aggregated from Amazon Data Firehose over one-minute intervals.

The metrics collected for Firehose streams are free of charge. For information about Kinesis agent metrics, see Monitoring Kinesis Agent Health.

Topics

Dynamic Partitioning CloudWatch Metrics

If dynamic partitioning is enabled, the Amazon/Firehose namespace includes the following metrics.

Metric	Description
`ActivePartitionsLimit`	The maximum number of active partitions that a Firehose stream processes before sending data to the error bucket. Units: Count
`PartitionCount`	The number of partitions that are being processed, in other words, the active partition count. This number varies between 1 and the partition count limit of 500 (default). Units: Count
`PartitionCountExceeded`	This metric indicates if you are exceeding the partition count limit. It emits 1 or 0 based on whether limit is breached or not.
`JQProcessing.Duration`	Returns the amount of time it took to execute JQ expression in the JQ Lambda function. Units: Milliseconds
`PerPartitionThroughput`	Indicates the throughtput that is being processed per partition. This metric enables you to monitor the per partition throughput. Units: StandardUnit.BytesSecond
`DeliveryToS3.ObjectCount`	Indicates the number of objects that are being delivered to your S3 bucket. Units: Count

Data Delivery CloudWatch Metrics

The `Amazon/Firehose` namespace includes the following service-level metrics. If you see small drops in the average for `BackupToS3.Success`, `DeliveryToS3.Success`, `DeliveryToSplunk.Success`, `DeliveryToAmazonOpenSearchService.Success`, or `DeliveryToRedshift.Success`, that doesn't indicate that there's data loss. Amazon Data Firehose retries delivery errors and doesn't move forward until the records are successfully delivered either to the configured destination or to the backup S3 bucket.

Topics

Delivery to OpenSearch Service

Metric	Description
`DeliveryToAmazonOpenSearchService.Bytes`	The number of bytes indexed to OpenSearch Service over the specified time period. Units: Bytes
`DeliveryToAmazonOpenSearchService.DataFreshness`	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to OpenSearch Service. Units: Seconds
`DeliveryToAmazonOpenSearchService.Records`	The number of records indexed to OpenSearch Service over the specified time period.

Metric	Description
	Units: Count
DeliveryToAmazonOpenSearchService.Success	The sum of the successfully indexed records over the sum of records that were attempted.
DeliveryToS3.Bytes	The number of bytes delivered to Amazon S3 over the specified time period. Amazon Data Firehose emits this metric only when you enable backup for all documents. Units: Count
DeliveryToS3.DataFreshness	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the S3 bucket. Amazon Data Firehose emits this metric only when you enable backup for all documents. Units: Seconds
DeliveryToS3.Records	The number of records delivered to Amazon S3 over the specified time period. Amazon Data Firehose emits this metric only when you enable backup for all documents. Units: Count
DeliveryToS3.Success	The sum of successful Amazon S3 put commands over the sum of all Amazon S3 put commands. Amazon Data Firehose always emits this metric regardless of whether backup is enabled for failed documents only or for all documents.
DeliveryToAmazonOpenSearchService.AuthFailure	Authentication/authorization error. Verify the OS/ES cluster policy and role permissions. 0 indicates that there is no issue. 1 indicates authentication failure.

Metric	Description
DeliveryToAmazonOpenSearchService.DeliveryRejected	Delivery rejected error. Verify the OS/ES cluster policy and role permissions. 0 indicates that there is no issue. 1 indicates that there's a delivery failure.

Delivery to OpenSearch Serverless

Metric	Description
DeliveryToAmazonOpenSearchServerless.Bytes	The number of bytes indexed to OpenSearch Serverless over the specified time period. Units: Bytes
DeliveryToAmazonOpenSearchServerless.DataFreshness	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to OpenSearch Serverless. Units: Seconds
DeliveryToAmazonOpenSearchServerless.Records	The number of records indexed to OpenSearch Serverless over the specified time period. Units: Count
DeliveryToAmazonOpenSearchServerless.Success	The sum of the successfully indexed records over the sum of records that were attempted.
DeliveryToS3.Bytes	The number of bytes delivered to Amazon S3 over the specified time period. Amazon Data Firehose emits this metric only when you enable backup for all documents. Units: Count

Metric	Description
DeliveryToS3.DataFreshness	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the S3 bucket. Amazon Data Firehose emits this metric only when you enable backup for all documents. Units: Seconds
DeliveryToS3.Records	The number of records delivered to Amazon S3 over the specified time period. Amazon Data Firehose emits this metric only when you enable backup for all documents. Units: Count
DeliveryToS3.Success	The sum of successful Amazon S3 put commands over the sum of all Amazon S3 put commands. Amazon Data Firehose always emits this metric regardless of whether backup is enabled for failed documents only or for all documents.
DeliveryToAmazonOpenSearchServerless.AuthFailure	Authentication/authorization error. Verify the OS/ES cluster policy and role permissions. 0 indicates that there is no issue. 1 indicates that there is an authentication failure.
DeliveryToAmazonOpenSearchServerless.DeliveryRejected	Delivery rejected error. Verify the OS/ES cluster policy and role permissions. 0 indicates that there is no issue. 1 indicates that there is a delivery failure.

Delivery to Amazon Redshift

Metric	Description
DeliveryToRedshift.Bytes	The number of bytes copied to Amazon Redshift over the specified time period. Units: Count
DeliveryToRedshift.Records	The number of records copied to Amazon Redshift over the specified time period. Units: Count
DeliveryToRedshift.Success	The sum of successful Amazon Redshift COPY commands over the sum of all Amazon Redshift COPY commands.
DeliveryToS3.Bytes	The number of bytes delivered to Amazon S3 over the specified time period. Units: Bytes
DeliveryToS3.DataFreshness	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the S3 bucket. Units: Seconds
DeliveryToS3.Records	The number of records delivered to Amazon S3 over the specified time period. Units: Count
DeliveryToS3.Success	The sum of successful Amazon S3 put commands over the sum of all Amazon S3 put commands.

Metric	Description
BackupToS3.Bytes	The number of bytes delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose emits this metric when backup to Amazon S3 is enabled. Units: Count
BackupToS3.DataFreshness	Age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the Amazon S3 bucket for backup. Amazon Data Firehose emits this metric when backup to Amazon S3 is enabled. Units: Seconds
BackupToS3.Records	The number of records delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose emits this metric when backup to Amazon S3 is enabled. Units: Count
BackupToS3.Success	Sum of successful Amazon S3 put commands for backup over sum of all Amazon S3 backup put commands. Amazon Data Firehose emits this metric when backup to Amazon S3 is enabled.

Delivery to Amazon S3

The metrics in the following table are related to delivery to Amazon S3 when it is the main destination of the Firehose stream.

Metric	Description
DeliveryToS3.Bytes	The number of bytes delivered to Amazon S3 over the specified time period.

Metric	Description
	Units: Bytes
DeliveryToS3.DataFreshness	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the S3 bucket. Units: Seconds
DeliveryToS3.Records	The number of records delivered to Amazon S3 over the specified time period. Units: Count
DeliveryToS3.Success	The sum of successful Amazon S3 put commands over the sum of all Amazon S3 put commands.
BackupToS3.Bytes	The number of bytes delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose emits this metric when backup is enabled (which is only possible when data transformation is also enabled). Units: Count
BackupToS3.DataFreshness	Age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the Amazon S3 bucket for backup. Amazon Data Firehose emits this metric when backup is enabled (which is only possible when data transformation is also enabled). Units: Seconds

Metric	Description
BackupToS3.Records	The number of records delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose emits this metric when backup is enabled (which is only possible when data transformation is also enabled). Units: Count
BackupToS3.Success	Sum of successful Amazon S3 put commands for backup over sum of all Amazon S3 backup put commands. Amazon Data Firehose emits this metric when backup is enabled (which is only possible when data transform ation is also enabled).

Delivery to Snowflake

Metric	Description
DeliveryToSnowflake.Bytes	The number of bytes delivered to Snowflake over the specified time period. Units: Bytes
DeliveryToSnowflak e.DataFreshness	Age (from getting into Firehose to now) of the oldest record in Firehose. Any record older than this age has been delivered to Snowflake. Units: Seconds
DeliveryToSnowflak e.Records	The number of records delivered to Snowflake over the specified time period. Units: Count
DeliveryToSnowflak e.Success	The sum of the successfully delivered records over the sum of records that were attempted.

Delivery to Splunk

Metric	Description
DeliveryToSplunk.Bytes	The number of bytes delivered to Splunk over the specified time period. Units: Bytes
DeliveryToSplunk.DataAckLatency	The approximate duration it takes to receive an acknowledgement from Splunk after Amazon Data Firehose sends it data. The increasing or decreasing trend for this metric is more useful than the absolute approximate value. Increasing trends can indicate slower indexing and acknowledgement rates from Splunk indexers. Units: Seconds
DeliveryToSplunk.DataFreshness	Age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to Splunk. Units: Seconds
DeliveryToSplunk.Records	The number of records delivered to Splunk over the specified time period. Units: Count
DeliveryToSplunk.Success	The sum of the successfully indexed records over the sum of records that were attempted.
DeliveryToS3.Success	The sum of successful Amazon S3 put commands over the sum of all Amazon S3 put commands. This metric is emitted when backup to Amazon S3 is enabled.
BackupToS3.Bytes	The number of bytes delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose

Metric	Description
	emits this metric when the Firehose stream is configured to back up all documents. Units: Count
BackupToS3.DataFreshness	Age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the Amazon S3 bucket for backup. Amazon Data Firehose emits this metric when the Firehose stream is configured to back up all documents. Units: Seconds
BackupToS3.Records	The number of records delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose emits this metric when the Firehose stream is configured to back up all documents. Units: Count
BackupToS3.Success	Sum of successful Amazon S3 put commands for backup over sum of all Amazon S3 backup put commands. Amazon Data Firehose emits this metric when the Firehose stream is configured to back up all documents.

Delivery to HTTP Endpoints

Metric	Description
DeliveryToHttpEndpoint.Bytes	The number of bytes delivered successfully to the HTTP endpoint. Units: Bytes
DeliveryToHttpEndpoint.Records	The number of records delivered successfully to the HTTP endpoint.

Metric	Description
	Units: Counts
DeliveryToHttpEndpoint.DataFreshness	Age of the oldest record in Amazon Data Firehose. Units: Seconds
DeliveryToHttpEndpoint.Success	The sum of all successful data delivery requests to the HTTP endpoint Units: Count
DeliveryToHttpEndpoint.ProcessedBytes	The number of attempted processed bytes, including retries.
DeliveryToHttpEndpoint.ProcessedRecords	The number of attempted records including retries.

Data Ingestion Metrics

Topics

Data Ingestion Through Kinesis Data Streams

Metric	Description
DataReadFromKinesisStream.Bytes	When the data source is a Kinesis data stream, this metric indicates the number of bytes read from that data stream. This number includes rereads due to failovers. Units: Bytes

Metric	Description
DataReadFromKinesisStream.Records	When the data source is a Kinesis data stream, this metric indicates the number of records read from that data stream. This number includes rereads due to failovers. Units: Count
ThrottledDescribeStream	The total number of times the DescribeStream operation is throttled when the data source is a Kinesis data stream. Units: Count
ThrottledGetRecords	The total number of times the GetRecords operation is throttled when the data source is a Kinesis data stream. Units: Count
ThrottledGetShardIterator	The total number of times the GetShardIterator operation is throttled when the data source is a Kinesis data stream. Units: Count

Data Ingestion Through Direct PUT

Metric	Description
BackupToS3.Bytes	The number of bytes delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinations. Units: Bytes

Metric	Description
BackupToS3.DataFreshness	Age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the Amazon S3 bucket for backup. Amazon Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinations. Units: Seconds
BackupToS3.Records	The number of records delivered to Amazon S3 for backup over the specified time period. Amazon Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinati ons. Units: Count
BackupToS3.Success	Sum of successful Amazon S3 put commands for backup over sum of all Amazon S3 backup put commands. Amazon Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinations.
BytesPerSecondLimit	The current maximum number of bytes per second that a Firehose stream can ingest before throttling. To request an increase to this limit, go to the Amazon Support Center and choose **Create case**, then choose **Service limit increase**.
DataReadFromKinesi sStream.Bytes	When the data source is a Kinesis data stream, this metric indicates the number of bytes read from that data stream. This number includes rereads due to failovers. Units: Bytes

Metric	Description
DataReadFromKinesisStream.Records	When the data source is a Kinesis data stream, this metric indicates the number of records read from that data stream. This number includes rereads due to failovers. Units: Count
DeliveryToAmazonOpenSearchService.Bytes	The number of bytes indexed to OpenSearch Service over the specified time period. Units: Bytes
DeliveryToAmazonOpenSearchService.DataFreshness	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to OpenSearch Service. Units: Seconds
DeliveryToAmazonOpenSearchService.Records	The number of records indexed to OpenSearch Service over the specified time period. Units: Count
DeliveryToAmazonOpenSearchService.Success	The sum of the successfully indexed records over the sum of records that were attempted.
DeliveryToRedshift.Bytes	The number of bytes copied to Amazon Redshift over the specified time period. Units: Bytes
DeliveryToRedshift.Records	The number of records copied to Amazon Redshift over the specified time period. Units: Count

Metric	Description
DeliveryToRedshift.Success	The sum of successful Amazon Redshift COPY commands over the sum of all Amazon Redshift COPY commands.
DeliveryToS3.Bytes	The number of bytes delivered to Amazon S3 over the specified time period. Units: Bytes
DeliveryToS3.DataFreshness	The age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to the S3 bucket. Units: Seconds
DeliveryToS3.Records	The number of records delivered to Amazon S3 over the specified time period. Units: Count
DeliveryToS3.Success	The sum of successful Amazon S3 put commands over the sum of all Amazon S3 put commands.
DeliveryToSplunk.Bytes	The number of bytes delivered to Splunk over the specified time period. Units: Bytes
DeliveryToSplunk.DataAckLatency	The approximate duration it takes to receive an acknowledgement from Splunk after Amazon Data Firehose sends it data. The increasing or decreasing trend for this metric is more useful than the absolute approximate value. Increasing trends can indicate slower indexing and acknowledgement rates from Splunk indexers. Units: Seconds

Metric	Description
DeliveryToSplunk.DataFreshness	Age (from getting into Amazon Data Firehose to now) of the oldest record in Amazon Data Firehose. Any record older than this age has been delivered to Splunk. Units: Seconds
DeliveryToSplunk.Records	The number of records delivered to Splunk over the specified time period. Units: Count
DeliveryToSplunk.Success	The sum of the successfully indexed records over the sum of records that were attempted.
IncomingBytes	The number of bytes ingested successfully into the delivery stream over the specified time period. Data ingestion could be throttled when it exceeds one of the delivery stream limits. Throttled data will not be counted for IncomingBytes . Units: Bytes
IncomingPutRequests	The number of successful PutRecord and PutRecordBatch requests over a specified period of time. Units: Count
IncomingRecords	The number of records ingested successfully into the delivery stream over the specified time period. Data ingestion could be throttled when it exceeds one of the delivery stream limits. Throttled data will not be counted for IncomingRecords . Units: Count

Metric	Description
KinesisMillisBehindLatest	When the data source is a Kinesis data stream, this metric indicates the number of milliseconds that the last read record is behind the newest record in the Kinesis data stream. Units: Millisecond
RecordsPerSecondLimit	The current maximum number of records per second that a Firehose stream can ingest before throttling. Units: Count
ThrottledRecords	The number of records that were throttled because data ingestion exceeded one of the Firehose stream limits. Units: Count

Data Ingestion From MSK

Metric	Description
DataReadFromSource .Records	The number of records read from the source Kafka Topic. Units: Count
DataReadFromSource.Bytes	The number of bytes read from the source Kafka Topic. Units: Bytes
SourceThrottled.Delay	The amount of time that the source Kafka cluster is delayed in returning the records from the source Kafka Topic. Units: Milliseconds

Metric	Description
BytesPerSecondLimit	Current limit of throughput at which Firehose is going to read from each partition of the source Kafka Topic. Units: Bytes/sec
KafkaOffsetLag	The difference between the largest offset of the record that Firehose has read from the source Kafka Topic and the largest offset of the record available from the source Kafka Topic. Units: Count
FailedValidation.Records	The number of records that failed record validation. Units: Count
FailedValidation.Bytes	The number of bytes that failed record validation. Units: Bytes
DataReadFromSource .Backpressured	Indicates that a Firehose stream is delayed in reading records from the source partition either because BytesPerSecondLimit per partition has exceeded or that the normal flow of delivery is slow or has stopped Units: Boolean

API-Level CloudWatch Metrics

The Amazon/Firehose namespace includes the following API-level metrics.

Metric	Description
DescribeDeliverySt ream.Latency	The time taken per DescribeDeliveryStream operation, measured over the specified time period. Units: Milliseconds

Metric	Description
DescribeDeliveryStream.Requests	The total number of DescribeDeliveryStream requests. Units: Count
ListDeliveryStreams.Latency	The time taken per ListDeliveryStream operation, measured over the specified time period. Units: Milliseconds
ListDeliveryStreams.Requests	The total number of ListFirehose requests. Units: Count
PutRecord.Bytes	The number of bytes put to the Firehose stream using PutRecord over the specified time period. Units: Bytes
PutRecord.Latency	The time taken per PutRecord operation, measured over the specified time period. Units: Milliseconds
PutRecord.Requests	The total number of PutRecord requests, which is equal to total number of records from PutRecord operations. Units: Count
PutRecordBatch.Bytes	The number of bytes put to the Firehose stream using PutRecordBatch over the specified time period. Units: Bytes
PutRecordBatch.Latency	The time taken per PutRecordBatch operation, measured over the specified time period. Units: Milliseconds

Metric	Description
PutRecordBatch.Records	The total number of records from PutRecordBatch operations. Units: Count
PutRecordBatch.Requests	The total number of PutRecordBatch requests. Units: Count
PutRequestsPerSecondLimit	The maximum number of put requests per second that a Firehose stream can handle before throttling. This number includes PutRecord and PutRecordBatch requests. Units: Count
ThrottledDescribeStream	The total number of times the DescribeStream operation is throttled when the data source is a Kinesis data stream. Units: Count
ThrottledGetRecords	The total number of times the GetRecords operation is throttled when the data source is a Kinesis data stream. Units: Count
ThrottledGetShardIterator	The total number of times the GetShardIterator operation is throttled when the data source is a Kinesis data stream. Units: Count
UpdateDeliveryStream.Latency	The time taken per UpdateDeliveryStream operation, measured over the specified time period. Units: Milliseconds

Metric	Description
UpdateDeliveryStream.Requests	The total number of UpdateDeliveryStream requests. Units: Count

Data Transformation CloudWatch Metrics

If data transformation with Lambda is enabled, the AWS/Firehose namespace includes the following metrics.

Metric	Description
ExecuteProcessing.Duration	The time it takes for each Lambda function invocation performed by Firehose. Units: Milliseconds
ExecuteProcessing.Success	The sum of the successful Lambda function invocations over the sum of the total Lambda function invocations.
SucceedProcessing.Records	The number of successfully processed records over the specified time period. Units: Count
SucceedProcessing.Bytes	The number of successfully processed bytes over the specified time period. Units: Bytes

CloudWatch Logs Decompression Metrics

If decompression is enabled for CloudWatch Logs delivery, the AWS/Firehose namespace includes the following metrics.

Metric	Description
OutputDecompressedBytes.Success	Successful decompressed data in bytes Units: Bytes
OutputDecompressedBytes.Failed	Failed decompressed data in bytes Units: Bytes
OutputDecompressedRecords.Success	Number of successful decompressed records Units: Count
OutputDecompressedRecords.Failed	Number of failed decompressed records Units: Count

Format Conversion CloudWatch Metrics

If format conversion is enabled, the AWS/Firehose namespace includes the following metrics.

Metric	Description
SucceedConversion.Records	The number of successfully converted records. Units: Count
SucceedConversion.Bytes	The size of the successfully converted records. Units: Bytes
FailedConversion.Records	The number of records that could not be converted. Units: Count
FailedConversion.Bytes	The size of the records that could not be converted. Units: Bytes

Server-Side Encryption (SSE) CloudWatch Metrics

The `Amazon/Firehose` namespace includes the following metrics that are related to SSE.

Metric	Description
`KMSKeyAccessDenied`	The number of times the service encounters a `KMSAccessDeniedException` for the delivery stream. Units: Count
`KMSKeyDisabled`	The number of times the service encounters a `KMSDisabledException` for the delivery stream. Units: Count
`KMSKeyInvalidState`	The number of times the service encounters a `KMSInvalidStateException` for the delivery stream. Units: Count
`KMSKeyNotFound`	The number of times the service encounters a `KMSNotFoundException` for the delivery stream. Units: Count

Dimensions for Amazon Data Firehose

To filter metrics by Firehose stream, use the `DeliveryStreamName` dimension.

Amazon Data Firehose Usage Metrics

You can use CloudWatch usage metrics to provide visibility into your account's usage of resources. Use these metrics to visualize your current service usage on CloudWatch graphs and dashboards.

Service quota usage metrics are in the Amazon/Usage namespace and are collected every minute.

Currently, the only metric name in this namespace that CloudWatch publishes is `ResourceCount`. This metric is published with the dimensions `Service`, `Class`, `Type`, and `Resource`.

Metric	Description
`ResourceCount`	The number of the specified resources running in your account. The resources are defined by the dimensions associated with the metric. The most useful statistic for this metric is MAXIMUM, which represents the maximum number of resources used during the 1-minute period.

The following dimensions are used to refine the usage metrics that are published by Amazon Data Firehose.

Dimension	Description
`Service`	The name of the Amazon service containing the resource. For Amazon Data Firehose usage metrics, the value for this dimension is `Firehose`.
`Class`	The class of resource being tracked. Amazon Data Firehose API usage metrics use this dimension with a value of `None`.
`Type`	The type of resource being tracked. Currently, when the Service dimension is `Firehose`, the only valid value for Type is `Resource`.
`Resource`	The name of the Amazon resource. Currently, when the Service dimension is `Firehose`, the only valid value for Resource is `DeliveryStreams` .

Accessing CloudWatch Metrics for Amazon Data Firehose

You can monitor metrics for Amazon Data Firehose using the CloudWatch console, command line, or CloudWatch API. The following procedures show you how to access metrics using these different methods.

To access metrics using the CloudWatch console

1. Open the CloudWatch console at https://console.amazonaws.cn/cloudwatch/.
2. On the navigation bar, choose a region.
3. In the navigation pane, choose **Metrics**.
4. Choose the **Firehose** namespace.
5. Choose **Firehose stream Metrics** or **Firehose Metrics**.
6. Select a metric to add to the graph.

To access metrics using the Amazon CLI

Use the list-metrics and get-metric-statistics commands.

```
aws cloudwatch list-metrics --namespace "Amazon/Firehose"
```

```
aws cloudwatch get-metric-statistics --namespace "Amazon/Firehose" \
--metric-name DescribeDeliveryStream.Latency --statistics Average --period 3600 \
--start-time 2017-06-01T00:00:00Z --end-time 2017-06-30T00:00:00Z
```

Monitoring Amazon Data Firehose Using CloudWatch Logs

Amazon Data Firehose integrates with Amazon CloudWatch Logs so that you can view the specific error logs when the Lambda invocation for data transformation or data delivery fails. You can enable Amazon Data Firehose error logging when you create your Firehose stream.

If you enable Amazon Data Firehose error logging in the Amazon Data Firehose console, a log group and corresponding log streams are created for the Firehose stream on your behalf. The format of the log group name is `/aws/kinesisfirehose/`*`delivery-stream-name`*, where *`delivery-stream-name`* is the name of the corresponding Firehose stream. `DestinationDelivery` is log stream that is created and used to log any errors related to the

delivery to the primary destination. Another log stream called `BackupDelivery` is created only if S3 backup is enabled for the destination. The `BackupDelivery` log stream is used to log any errors related to the delivery to the S3 backup.

For example, if you create a Firehose stream "MyStream" with Amazon Redshift as the destination and enable Amazon Data Firehose error logging, the following are created on your behalf: a log group named `aws/kinesisfirehose/MyStream` and two log streams named `DestinationDelivery` and **BackupDelivery**. In this example, `DestinationDelivery` will be used to log any errors related to the delivery to the Amazon Redshift destination and also to the intermediate S3 destination. `BackupDelivery`, in case S3 backup is enabled, will be used to log any errors related to the delivery to the S3 backup bucket.

You can enable Amazon Data Firehose error logging through the Amazon CLI, the API, or Amazon CloudFormation using the `CloudWatchLoggingOptions` configuration. To do so, create a log group and a log stream in advance. We recommend reserving that log group and log stream for Amazon Data Firehose error logging exclusively. Also ensure that the associated IAM policy has `"logs:putLogEvents"` permission. For more information, see Controlling Access with Amazon Data Firehose .

Note that Amazon Data Firehose does not guarantee that all delivery error logs are sent to CloudWatch Logs. In circumstances where delivery failure rate is high, Amazon Data Firehose samples delivery error logs before sending them to CloudWatch Logs.

There is a nominal charge for error logs sent to CloudWatch Logs. For more information, see Amazon CloudWatch Pricing.

Contents

Data Delivery Errors

The following is a list of data delivery error codes and messages for each Amazon Data Firehose destination. Each error message also describes the proper action to take to fix the issue.

Errors

- Splunk Data Delivery Errors
- ElasticSearch Data Delivery Errors
- HTTPS Endpoint Data Delivery Errors
- Amazon OpenSearch Service Data Delivery Errors
- Lambda Invocation Errors
- Kinesis Invocation Errors
- Kinesis DirectPut Invocation Errors
- Amazon Glue Invocation Errors
- DataFormatConversion Invocation Errors

Amazon S3 Data Delivery Errors

Amazon Data Firehose can send the following Amazon S3-related errors to CloudWatch Logs.

Error Code	Error Message and Information
`S3.KMS.NotFoundException`	"The provided Amazon KMS key was not found. If you are using what you believe to be a valid Amazon KMS key with the correct role, check if there is a problem with the account to which the Amazon KMS key is attached."
`S3.KMS.RequestLimitExceeded`	"The KMS request per second limit was exceeded while attempting to encrypt S3 objects. Increase the request per second limit." For more information, see Limits in the *Amazon Key Management Service Developer Guide*.
`S3.AccessDenied`	"Access was denied. Ensure that the trust policy for the provided IAM role allows Amazon Data Firehose to assume the role, and the access policy allows access to the S3 bucket."
`S3.AccountProblem`	"There is a problem with your Amazon account that prevents the operation from completing successfully. Contact Amazon Support."
`S3.AllAccessDisabled`	"Access to the account provided has been disabled. Contact Amazon Support."

Error Code	Error Message and Information
S3.InvalidPayer	"Access to the account provided has been disabled. Contact Amazon Support."
S3.NotSignedUp	"The account is not signed up for Amazon S3. Sign the account up or use a different account."
S3.NoSuchBucket	"The specified bucket does not exist. Create the bucket or use a different bucket that does exist."
S3.Method NotAllowed	"The specified method is not allowed against this resource. Modify the bucket's policy to allow the correct Amazon S3 operation permissions."
InternalError	"An internal error occurred while attempting to deliver data. Delivery will be retried; if the error persists, then it will be reported to Amazon for resolution."
S3.KMS.KeyDisabled	"The provided KMS key is disabled. Enable the key or use a different key."
S3.KMS.InvalidStateException	"The provided KMS key is in an invalid state. Please use a different key."
KMS.InvalidStateException	"The provided KMS key is in an invalid state. Please use a different key."
KMS.DisabledException	"The provided KMS key is disabled. Please fix the key or use a different key."
S3.SlowDown	"The rate of put request to the specified bucket was too high. Increase Firehose stream buffer size or reduce put requests from other applications."
S3.SubscriptionRequired	"Access was denied when calling S3. Ensure that the IAM role and the KMS Key (if provided) passed in has Amazon S3 subscription."

Error Code	Error Message and Information
S3.InvalidToken	"The provided token is malformed or otherwise invalid. Please check the credentials provided."
S3.KMS.KeyNotConfigured	"KMS key not configured. Configure your KMSMasterKeyID, or disable encryption for your S3 bucket."
S3.KMS.AsymmetricCMKNotSupported	"Amazon S3 supports only symmetric CMKs. You cannot use an asymmetric CMK to encrypt your data in Amazon S3. To get the type of your CMK, use the KMS DescribeKey operation."
S3.IllegalLocationConstraintException	"Firehose currently uses s3 global endpoint for data delivery to the configured s3 bucket. The region of the configured s3 bucket doesn't support s3 global endpoint. Please create a Firehose stream in the same region as the s3 bucket or use s3 bucket in the region that supports global endpoint."
S3.InvalidPrefixConfigurationException	"The custom s3 prefix used for the timestamp evaluation is invalid. Check your s3 prefix contains valid expressions for the current date and time of the year."
DataFormatConversion.MalformedData	"Illegal character found between tokens."

Amazon Redshift Data Delivery Errors

Amazon Data Firehose can send the following Amazon Redshift-related errors to CloudWatch Logs.

Error Code	Error Message and Information
Redshift.TableNotFound	"The table to which to load data was not found. Ensure that the specified table exists."

Error Code	Error Message and Information
	The destination table in Amazon Redshift to which data should be copied from S3 was not found. Note that Amazon Data Firehose does not create the Amazon Redshift table if it does not exist.
`Redshift.SyntaxError`	"The COPY command contains a syntax error. Retry the command."
`Redshift.AuthenticationFailed`	"The provided user name and password failed authentication. Provide a valid user name and password."
`Redshift.AccessDenied`	"Access was denied. Ensure that the trust policy for the provided IAM role allows Amazon Data Firehose to assume the role."
`Redshift.S3BucketAccessDenied`	"The COPY command was unable to access the S3 bucket. Ensure that the access policy for the provided IAM role allows access to the S3 bucket."
`Redshift.DataLoadFailed`	"Loading data into the table failed. Check STL_LOAD_ERRORS system table for details."
`Redshift.ColumnNotFound`	"A column in the COPY command does not exist in the table. Specify a valid column name."
`Redshift.DatabaseNotFound`	"The database specified in the Amazon Redshift destination configuration or JDBC URL was not found. Specify a valid database name."
`Redshift.IncorrectCopyOptions`	"Conflicting or redundant COPY options were provided. Some options are not compatible in certain combinations. Check the COPY command reference for more info." For more information, see the Amazon Redshift COPY command in the *Amazon Redshift Database Developer Guide*.

Error Code	Error Message and Information
`Redshift.MissingColumn`	"There is a column defined in the table schema as NOT NULL without a DEFAULT value and not included in the column list. Exclude this column, ensure that the loaded data always provides a value for this column, or add a default value to the Amazon Redshift schema for this table."
`Redshift.ConnectionFailed`	"The connection to the specified Amazon Redshift cluster failed. Ensure that security settings allow Amazon Data Firehose connections, that the cluster or database specified in the Amazon Redshift destination configuration or JDBC URL is correct, and that the cluster is available."
`Redshift.ColumnMismatch`	"The number of jsonpaths in the COPY command and the number of columns in the destination table should match. Retry the command."
`Redshift.IncorrectOrMissingRegion`	"Amazon Redshift attempted to use the wrong region endpoint for accessing the S3 bucket. Either specify a correct region value in the COPY command options or ensure that the S3 bucket is in the same region as the Amazon Redshift database."
`Redshift.IncorrectJsonPathsFile`	"The provided jsonpaths file is not in a supported JSON format. Retry the command."
`Redshift.MissingS3File`	"One or more S3 files required by Amazon Redshift have been removed from the S3 bucket. Check the S3 bucket policies to remove any automatic deletion of S3 files."
`Redshift.InsufficientPrivilege`	"The user does not have permissions to load data into the table. Check the Amazon Redshift user permissions for the INSERT privilege."
`Redshift.ReadOnlyCluster`	"The query cannot be executed because the system is in resize mode. Try the query again later."

Error Code	Error Message and Information
`Redshift.DiskFull`	"Data could not be loaded because the disk is full. Increase the capacity of the Amazon Redshift cluster or delete unused data to free disk space."
`InternalError`	"An internal error occurred while attempting to deliver data. Delivery will be retried; if the error persists, then it will be reported to Amazon for resolution."
`Redshift.ArgumentNotSupported`	"The COPY command contains unsupported options."
`Redshift.AnalyzeTableAccessDenied`	"Access denied. Copy from S3 to Redshift is failing because analyze table can only be done by table or database owner."
`Redshift.SchemaNotFound`	"The schema specified in the DataTableName of Amazon Redshift destination configuration was not found. Specify a valid schema name."
`Redshift.ColumnSpecifiedMoreThanOnce`	"There is a column specified more than once in the column list. Ensure that duplicate columns are removed."
`Redshift.ColumnNotNullWithoutDefault`	"There is a non-null column without DEFAULT that is not included in the column list. Ensure that such columns are included in the column list."
`Redshift.IncorrectBucketRegion`	"Redshift attempted to use a bucket in a different region from the cluster. Please specify a bucket within the same region as the cluster."
`Redshift.S3SlowDown`	"High request rate to S3. Reduce the rate to avoid getting throttled."

Error Code	Error Message and Information
`Redshift.InvalidCopyOptionForJson`	"Please use either auto or a valid S3 path for json copyOption."
`Redshift.InvalidCopyOptionJSONPathFormat`	"COPY failed with error \"Invalid JSONPath format. Array index is out of range\". Please rectify the JSONPath expression."
`Redshift.InvalidCopyOptionRBACAclNotAllowed`	"COPY failed with error \"Cannot use RBAC acl framework while permission propagation is not enabled.\"
`Redshift.DiskSpaceQuotaExceeded`	"Transaction aborted due to disk space quota exceed. Free up disk space or request increased quota for the schema(s)."
`Redshift.ConnectionsLimitExceeded`	"Connection limit exceeded for user."
`Redshift.SslNotSupported`	"The connection to the specified Amazon Redshift cluster failed because the server does not support SSL. Please check your cluster settings."
`Redshift.HoseNotFound`	"The hose has been deleted. Please check the status of your hose."
`Redshift.Delimiter`	"The copyOptions delimiter in the copyCommand is invalid. Ensure that it is a single character."

Error Code	Error Message and Information
`Redshift.QueryCancelled`	"The user has canceled the COPY operation."
`Redshift.CompressionMismatch`	"Hose is configured with UNCOMPRESSED, but copyOption includes a compression format."
`Redshift.EncryptionCredentials`	"The ENCRYPTED option requires credentials in the format: 'aws_iam_role=...;master_symmetric_key=...' or 'aws_access_key_id=...;aws_secret_access_key=...[;token=...];master_symmetric_key=...'"
`Redshift.InvalidCopyOptions`	"Invalid COPY configuration options."
`Redshift.InvalidMessageFormat`	"Copy command contains an invalid character."
`Redshift.TransactionIdLimitReached`	"Transaction ID limit reached."
`Redshift.DestinationRemoved`	"Please verify that the redshift destination exists and is configured correctly in the Firehose configuration."
`Redshift.OutOfMemory`	"The Redshift cluster is running out of memory. Please ensure the cluster has sufficient capacity."
`Redshift.CannotForkProcess`	"The Redshift cluster is running out of memory. Please ensure the cluster has sufficient capacity."

Error Code	Error Message and Information
`Redshift.SslFailure`	"The SSL connection closed during the handshake."
`Redshift.Resize`	"The Redshift cluster is resizing. Firehose will not be able to deliver data while the cluster is resizing."
`Redshift.ImproperQualifiedName`	"The qualified name is improper (too many dotted names)."
`Redshift.InvalidJsonPathFormat`	"Invalid JSONPath Format."
`Redshift.TooManyConnectionsException`	"Too many connections to Redshift."
`Redshift.PSQLException`	"PSQlException observed from Redshift."
`Redshift.DuplicateSecondsSpecification`	"Duplicate seconds specification in date/time format."
`Redshift.RelationCouldNotBeOpened`	"Encountered Redshift error, relation could not be opened. Check Redshift logs for the specified DB."
`Redshift.TooManyClients`	"Encountered too many clients exception from Redshift. Revisit max connections to the database if there are multiple producers writing to it simultaneously."

Snowflake Data Delivery Errors

Firehose can send the following Snowflake-related errors to CloudWatch Logs.

Error Code	Error Message and Information
`Snowflake.InvalidUrl`	"Firehose is unable to connect to Snowflake. Please make sure that Account url is specified correctly in Snowflake destination configuration."
`Snowflake.InvalidUser`	"Firehose is unable to connect to Snowflake. Please make sure that User is specified correctly in Snowflake destination configuration."
`Snowflake.InvalidRole`	"The specified snowflake role does not exist or is not authorized. Please make sure that the role is granted to the user specified"
`Snowflake.InvalidTable`	"The supplied table does not exist or is not authorized"
`Snowflake.InvalidSchema`	"The supplied schema does not exist or is not authorized"
`Snowflake.InvalidDatabase`	"The supplied database does not exist or is not authorized"
`Snowflake.InvalidPrivateKeyOrPassphrase`	"The specified private key or passphrase is not valid. Note that the private key provided should be a valid PEM RSA private key"
`Snowflake.MissingColumns`	"The insert request is rejected due to missing columns in input payload. Make sure that values are specified for all non-nullable columns"
`Snowflake.ExtraColumns`	"The insert request is rejected due to extra columns. Columns not present in table shouldn't be specified"

Error Code	Error Message and Information
`Snowflake.InvalidInput`	"Delivery failed due to invalid input format. Make sure that the input payload provided is in the JSON format acceptable"
`Snowflake.IncorrectValue`	"Delivery failed due to incorrect data type in the input payload. Make sure that the JSON values specified in input payload adhere to the datatype declared in Snowflake table definition"

Splunk Data Delivery Errors

Amazon Data Firehose can send the following Splunk-related errors to CloudWatch Logs.

Error Code	Error Message and Information
`Splunk.ProxyWithoutStickySessions`	"If you have a proxy (ELB or other) between Amazon Data Firehose and the HEC node, you must enable sticky sessions to support HEC ACKs."
`Splunk.DisabledToken`	"The HEC token is disabled. Enable the token to allow data delivery to Splunk."
`Splunk.InvalidToken`	"The HEC token is invalid. Update Amazon Data Firehose with a valid HEC token."
`Splunk.InvalidDataFormat`	"The data is not formatted correctly. To see how to properly format data for Raw or Event HEC endpoints, see Splunk Event Data."
`Splunk.InvalidIndex`	"The HEC token or input is configured with an invalid index. Check your index configuration and try again."
`Splunk.ServerError`	"Data delivery to Splunk failed due to a server error from the HEC node. Amazon Data Firehose will retry sending the data if the retry duration in your Amazon Data Firehose is greater than 0. If all the retries fail, Amazon Data Firehose backs up the data to Amazon S3."

Error Code	Error Message and Information
`Splunk.DisabledAck`	"Indexer acknowledgement is disabled for the HEC token. Enable indexer acknowledgement and try again. For more info, see Enable indexer acknowledgement."
`Splunk.AckTimeout`	"Did not receive an acknowledgement from HEC before the HEC acknowledgement timeout expired. Despite the acknowledgement timeout, it's possible the data was indexed successfully in Splunk. Amazon Data Firehose backs up in Amazon S3 data for which the acknowledgement timeout expired."
`Splunk.MaxRetriesFailed`	"Failed to deliver data to Splunk or to receive acknowledgment. Check your HEC health and try again."
`Splunk.ConnectionTimeout`	"The connection to Splunk timed out. This might be a transient error and the request will be retried. Amazon Data Firehose backs up the data to Amazon S3 if all retries fail."
`Splunk.InvalidEndpoint`	"Could not connect to the HEC endpoint. Make sure that the HEC endpoint URL is valid and reachable from Amazon Data Firehose."
`Splunk.ConnectionClosed`	"Unable to send data to Splunk due to a connection failure. This might be a transient error. Increasing the retry duration in your Amazon Data Firehose configuration might guard against such transient failures."
`Splunk.SSLUnverified`	"Could not connect to the HEC endpoint. The host does not match the certificate provided by the peer. Make sure that the certificate and the host are valid."
`Splunk.SSLHandshake`	"Could not connect to the HEC endpoint. Make sure that the certificate and the host are valid."
`Splunk.URLNotFound`	"The requested URL was not found on the Splunk server. Please check the Splunk cluster and make sure it is configured correctly."

Error Code	Error Message and Information
`Splunk.ServerError.ContentTooLarge`	"Data delivery to Splunk failed due to a server error with a statusCode: 413, message: the request your client sent was too large. See splunk docs to configure max_content_length."
`Splunk.IndexerBusy`	"Data delivery to Splunk failed due to a server error from the HEC node. Make sure HEC endpoint or the Elastic Load Balancer is reachable and is healthy."
`Splunk.ConnectionRecycled`	"The connection from Firehose to Splunk has been recycled. Delivery will be retried."
`Splunk.AcknowledgementsDisabled`	"Could not get acknowledgements on POST. Make sure that acknowledgements are enabled on HEC endpoint."
`Splunk.InvalidHecResponseCharacter`	"Invalid characters found in HEC response, make sure to check to the service and HEC configuration."

ElasticSearch Data Delivery Errors

Amazon Data Firehose can send the following ElasticSearch errors to CloudWatch Logs.

Error Code	Error Message and Information
`ES.AccessDenied`	"Access was denied. Ensure that the provided IAM role associated with firehose is not deleted."
`ES.ResourceNotFound`	"The specified Amazon Elasticsearch domain does not exist."

HTTPS Endpoint Data Delivery Errors

Amazon Data Firehose can send the following HTTP Endpoint-related errors to CloudWatch Logs. If none of these errors are a match to the problem that you're experiencing, the default error is the following: "An internal error occurred while attempting to deliver data. Delivery will be retried; if the error persists, then it will be reported to Amazon for resolution."

Error Code	Error Message and Information
`HttpEndpoint.RequestTimeout`	The delivery timed out before a response was received and will be retried. If this error persists, contact the Amazon Firehose service team.
`HttpEndpoint.ResponseTooLarge`	"The response received from the endpoint is too large. Contact the owner of the endpoint to resolve this issue."
`HttpEndpoint.InvalidResponseFromDestination`	"The response received from the specified endpoint is invalid. Contact the owner of the endpoint to resolve the issue."
`HttpEndpoint.DestinationException`	"The following response was received from the endpoint destination."
`HttpEndpoint.ConnectionFailed`	"Unable to connect to the destination endpoint. Contact the owner of the endpoint to resolve this issue."
`HttpEndpoint.ConnectionReset`	"Unable to maintain connection with the endpoint. Contact the owner of the endpoint to resolve this issue."

Error Code	Error Message and Information
HttpEndpoint.ConnectionReset	"Trouble maintaining connection with the endpoint. Please reach out to the owner of the endpoint."
HttpEndpoint.ResponseReasonPhraseExceededLimit	"The response reason phrase received from the endpoint exceed the configured limit of 64 characters."
HttpEndpoint.InvalidResponseFromDestination	"The response received from the endpoint is invalid. See Troublesh ooting HTTP Endpoints in the Firehose documentation for more information. Reason: "
HttpEndpoint.DestinationException	"Delivery to the endpoint was unsuccessful. See Troubleshooting HTTP Endpoints in the Firehose documentation for more information. Response received with status code "
HttpEndpoint.InvalidStatusCode	"Received an invalid response status code."
HttpEndpoint.SSLHandshakeFailure	"Unable to complete an SSL Handshake with the endpoint. Contact the owner of the endpoint to resolve this issue."
HttpEndpoint.SSLHandshakeFailure	"Unable to complete an SSL Handshake with the endpoint. Contact the owner of the endpoint to resolve this issue."
HttpEndpoint.SSLFailure	"Unable to complete TLS handshake with the endpoint. Contact the owner of the endpoint to resolve this issue."

Error Code	Error Message and Information
HttpEndpoint.SSLHandshakeCertificatePathFailure	"Unable to complete an SSL Handshake with the endpoint due to invalid certification path. Contact the owner of the endpoint to resolve this issue."
HttpEndpoint.SSLHandshakeCertificatePathValidationFailure	"Unable to complete an SSL Handshake with the endpoint due to certification path validation failure. Contact the owner of the endpoint to resolve this issue."
HttpEndpoint.MakeRequestFailure.IllegalUriException	"HttpEndpoint request failed due to invalid input in URI. Please make sure all the characters in the input URI are valid."
HttpEndpoint.MakeRequestFailure.IllegalCharacterInHeaderValue	"HttpEndpoint request failed due to illegal response error. Illegal character '\n' in header value."
HttpEndpoint.IllegalResponseFailure	"HttpEndpoint request failed due to illegal response error. HTTP message must not contain more than one Content-Type header."

Error Code	Error Message and Information
`HttpEndpoint.IllegalMessageStart`	"HttpEndpoint request failed due to illegal response error. Illegal HTTP message start. See Troubleshooting HTTP Endpoints in the Firehose documentation for more information."

Amazon OpenSearch Service Data Delivery Errors

For the OpenSearch Service destination, Amazon Data Firehose sends errors to CloudWatch Logs as they are returned by OpenSearch Service.

In addition to errors that may return from OpenSearch clusters, you may encounter the following two errors:

- Authentication/authorization error occurs during attempt to deliver data to destination OpenSearch Service cluster. This can happen due to any permission issues and/or intermittently when your Amazon Data Firehose target OpenSearch Service domain configuration is modified. Please check the cluster policy and role permissions.
- Data couldn't be delivered to destination OpenSearch Service cluster due to authentication/authorization failures. This can happen due to any permission issues and/or intermittently when your Amazon Data Firehose target OpenSearch Service domain configuration is modified. Please check the cluster policy and role permissions.

Error Code	Error Message and Information
`OS.AccessDenied`	"Access was denied. Ensure that the trust policy for the provided IAM role allows Firehose to assume the role, and the access policy allows access to the Amazon OpenSearch Service API."
`OS.AccessDenied`	"Access was denied. Ensure that the trust policy for the provided IAM role allows Firehose to assume the role, and the access policy allows access to the Amazon OpenSearch Service API."
`OS.AccessDenied`	"Access was denied. Ensure that the provided IAM role associated with firehose is not deleted."

Error Code	Error Message and Information
OS.AccessDenied	"Access was denied. Ensure that the provided IAM role associated with firehose is not deleted."
OS.ResourceNotFound	"The specified Amazon OpenSearch Service domain does not exist."
OS.ResourceNotFound	"The specified Amazon OpenSearch Service domain does not exist."
OS.AccessDenied	"Access was denied. Ensure that the trust policy for the provided IAM role allows Firehose to assume the role, and the access policy allows access to the Amazon OpenSearch Service API."
OS.RequestTimeout	"Request to the Amazon OpenSearch Service cluster or OpenSearch Serverless collection timed out. Ensure that the cluster or collection has sufficient capacity for the current workload."
OS.ClusterError	"The Amazon OpenSearch Service cluster returned an unspecified error."
OS.RequestTimeout	"Request to the Amazon OpenSearch Service cluster timed out. Ensure that the cluster has sufficient capacity for the current workload."
OS.ConnectionFailed	"Trouble connecting to the Amazon OpenSearch Service cluster or OpenSearch Serverless collection. Ensure that the cluster or collection is healthy and reachable."
OS.ConnectionReset	"Unable to maintain connection with the Amazon OpenSearch Service cluster or OpenSearch Serverless collection. Contact the owner of the cluster or collection to resolve this issue."
OS.ConnectionReset	"Trouble maintaining connection with the Amazon OpenSearch Service cluster or OpenSearch Serverless collection. Ensure that the cluster or collection is healthy and has sufficient capacity for the current workload."

Error Code	Error Message and Information
OS.ConnectionReset	"Trouble maintaining connection with the Amazon OpenSearch Service cluster or OpenSearch Serverless collection. Ensure that the cluster or collection is healthy and has sufficient capacity for the current workload."
OS.AccessDenied	"Access was denied. Ensure that the access policy on the Amazon OpenSearch Service cluster grants access to the configured IAM role."
OS.ValidationException	"The OpenSearch cluster returned a ESServiceException. One of the reasons is that the cluster has been upgraded to OS 2.x or higher, but the hose still has the TypeName parameter configured. Update the hose configuration by setting the TypeName to an empty string, or change the endpoint to the cluster, that supports the Type parameter."
OS.ValidationException	"Member must satisfy regular expression pattern: [a-z][a-z0-9\\-]+
OS.JsonParseException	"The Amazon OpenSearch Service cluster returned a JsonParse Exception. Ensure that the data being put is valid."
OS.AmazonOpenSearchServiceParseException	"The Amazon OpenSearch Service cluster returned an AmazonOpenSearchServiceParseException. Ensure that the data being put is valid."
OS.ExplicitIndexInBulkNotAllowed	"Ensure rest.action.multi.allow_explicit_index is set to true on the Amazon OpenSearch Service cluster."
OS.ClusterError	"The Amazon OpenSearch Service cluster or OpenSearch Serverless collection returned an unspecified error."
OS.ClusterBlockException	"The cluster returned a ClusterBlockException. It may be overloaded."

Error Code	Error Message and Information
OS.InvalidARN	"The Amazon OpenSearch Service ARN provided is invalid. Please check your DeliveryStream configuration."
OS.MalformedData	"One or more records are malformed. Please ensure that each record is single valid JSON object and that it does not contain newlines."
OS.InternalError	"An internal error occurred when attempting to deliver data. Delivery will be retried; if the error persists, it will be reported to Amazon for resolution."
OS.AliasWithMultipleIndicesNotAllowed	"Alias has more than one indices associated with it. Ensure that the alias has only one index associated with it."
OS.UnsupportedVersion	"Amazon OpenSearch Service 6.0 is not currently supported by Amazon Data Firehose. Contact Amazon Support for more information."
OS.CharConversionException	"One or more records contained an invalid character."
OS.InvalidDomainNameLength	"The domain name length is not within valid OS limits."
OS.VPCDomainNotSupported	"Amazon OpenSearch Service domains within VPCs are currently not supported."
OS.ConnectionError	"The http server closed the connection unexpectedly, please verify the health of the Amazon OpenSearch Service cluster or OpenSearch Serverless collection."
OS.LargeFieldData	"The Amazon OpenSearch Service cluster aborted the request as it contained a field data larger than allowed."

Error Code	Error Message and Information
`OS.BadGateway`	"The Amazon OpenSearch Service cluster or OpenSearch Serverless collection aborted the request with a response: 502 Bad Gateway."
`OS.ServiceException`	"Error received from the Amazon OpenSearch Service cluster or OpenSearch Serverless collection. If the cluster or collection is behind a VPC, ensure network configuration allows connectivity."
`OS.GatewayTimeout`	"Firehose encountered timeout errors when connecting to the Amazon OpenSearch Service cluster or OpenSearch Serverless collection."
`OS.MalformedData`	"Amazon Data Firehose does not support Amazon OpenSearch Service Bulk API commands inside the Firehose record."
`OS.ResponseEntryCountMismatch`	"The response from the Bulk API contained more entries than the number of records sent. Ensure that each record contains only one JSON object and that there are no newlines."

Lambda Invocation Errors

Amazon Data Firehose can send the following Lambda invocation errors to CloudWatch Logs.

Error Code	Error Message and Information
`Lambda.AssumeRoleAccessDenied`	"Access was denied. Ensure that the trust policy for the provided IAM role allows Amazon Data Firehose to assume the role."
`Lambda.InvokeAccessDenied`	"Access was denied. Ensure that the access policy allows access to the Lambda function."
`Lambda.JsonProcessingException`	"There was an error parsing returned records from the Lambda function. Ensure that the returned records follow the status model required by Amazon Data Firehose." For more information, see Data Transformation and Status Model.

Error Code	Error Message and Information
`Lambda.InvokeLimitExceeded`	"The Lambda concurrent execution limit is exceeded. Increase the concurrent execution limit." For more information, see Amazon Lambda Limits in the *Amazon Lambda Developer Guide*.
`Lambda.DuplicatedRecordId`	"Multiple records were returned with the same record ID. Ensure that the Lambda function returns unique record IDs for each record." For more information, see Data Transformation and Status Model.
`Lambda.MissingRecordId`	"One or more record IDs were not returned. Ensure that the Lambda function returns all received record IDs." For more information, see Data Transformation and Status Model.
`Lambda.ResourceNotFound`	"The specified Lambda function does not exist. Use a different function that does exist."
`Lambda.InvalidSubnetIDException`	"The specified subnet ID in the Lambda function VPC configuration is invalid. Ensure that the subnet ID is valid."
`Lambda.InvalidSecurityGroupIDException`	"The specified security group ID in the Lambda function VPC configuration is invalid. Ensure that the security group ID is valid."
`Lambda.SubnetIPAddressLimitReachedException`	"Amazon Lambda was not able to set up the VPC access for the Lambda function because one or more configured subnets have no available IP addresses. Increase the IP address limit." For more information, see Amazon VPC Limits - VPC and Subnets in the *Amazon VPC User Guide*.

Error Code	Error Message and Information
`Lambda.ENILimitReachedException`	"Amazon Lambda was not able to create an Elastic Network Interface (ENI) in the VPC, specified as part of the Lambda function configuration, because the limit for network interfaces has been reached. Increase the network interface limit." For more information, see Amazon VPC Limits - Network Interfaces in the *Amazon VPC User Guide*.
Lambda.FunctionTimedOut	The Lambda function invocation timed out. Increase the Timeout setting in the Lambda function. For more information, see Configuring function timeout.
Lambda.FunctionError	This can be due to any of the following errors: • Invalid output structure. Check your function and make sure the output is in the required format. Also, make sure the processed records contain a valid result status of `Dropped`, `Ok`, or `ProcessingFailed` . • The Lambda function was successfully invoked but it returned an error result. • Lambda was unable to decrypt the environment variables because KMS access was denied. Check the function's KMS key settings as well as the key policy. For more information, see Troubleshooting Key Access.
Lambda.FunctionRequestTimedOut	Amazon Data Firehose encountered Request did not complete before the request timeout configuration error when invoking Lambda. Revisit the Lambda code to check if the Lambda code is menat to run beyond the configured timeout. If so, consider tuning Lambda configuration settings, including memory, timeout. For more information, see Configuring Lambda function options.
Lambda.TargetServerFailedToRespond	Amazon Data Firehose encountered an error. Target server failed to respond error when calling the Amazon Lambda service.

Error Code	Error Message and Information
Lambda.InvalidZipFileException	Amazon Data Firehose encountered InvalidZipFileException when invoking the Lambda function. Check your Lambda function configuration settings and the Lambda code zip file.
Lambda.InternalServerError	"Amazon Data Firehose encountered InternalServerError when calling the Amazon Lambda service. Amazon Data Firehose will retry sending data a fixed number of times. You can specify or override the retry options using the `CreateDeliveryStream` or `UpdateDestination` APIs. If the error persists, contact Amazon Lambda support team.
Lambda.ServiceUnavailable	Amazon Data Firehose encountered ServiceUnavailableException when calling the Amazon Lambda service. Amazon Data Firehose will retry sending data a fixed number of times. You can specify or override the retry options using the `CreateDeliveryStream` or `UpdateDestination` APIs. If the error persists, contact Amazon Lambda support.
Lambda.InvalidSecurityToken	Cannot invoke Lambda function due to invalid security token. Cross partition Lambda invocation is not supported.

Error Code	Error Message and Information
Lambda.InvocationFailure	This can be due to any of the following errors: • Amazon Data Firehose encountered errors when calling Amazon Lambda. The operation will be retried; if the error persists, it will be reported to Amazon for resolution." • Amazon Data Firehose encountered a KMSInvalidStateException from Lambda. Lambda was unable to decrypt the environment variables because the KMS key used is in an invalid state for Decrypt. Check the lambda function's KMS key. • Amazon Data Firehose encountered an AmazonLambdaException from Lambda. Lambda was unable to initialize the provided container image. Verify the image. • Amazon Data Firehose encountered timeout errors when calling Amazon Lambda. The maximum supported function timeout is 5 minutes. For more information, see Data Transformation Execution Duration.
Lambda.JsonMappingException	There was an error parsing returned records from the Lambda function. Ensure that data field is base-64 encoded.

Kinesis Invocation Errors

Amazon Data Firehose can send the following Kinesis invocation errors to CloudWatch Logs.

Error Code	Error Message and Information
`Kinesis.AccessDenied`	"Access was denied when calling Kinesis. Ensure the access policy on the IAM role used allows access to the appropriate Kinesis APIs."
`Kinesis.ResourceNotFound`	"Firehose failed to read from the stream. If the Firehose is attached with Kinesis Stream, the stream may not exist, or the shard may have been merged or split. If the Firehose is of DirectPut type, the Firehose may not exist any more."

Error Code	Error Message and Information
`Kinesis.Subscriptio nRequired`	"Access was denied when calling Kinesis. Ensure that the IAM role passed for Kinesis stream access has Amazon Kinesis subscription."
`Kinesis.Throttling`	"Throttling error encountered when calling Kinesis. This can be due to other applications calling the same APIs as the Firehose Firehose stream, or because you have created too many Firehose Firehose streams with the same Kinesis stream as the source."
`Kinesis.Throttling`	"Throttling error encountered when calling Kinesis. This can be due to other applications calling the same APIs as the Firehose Firehose stream, or because you have created too many Firehose Firehose streams with the same Kinesis stream as the source."
`Kinesis.AccessDenied`	"Access was denied when calling Kinesis. Ensure the access policy on the IAM role used allows access to the appropriate Kinesis APIs."
`Kinesis.AccessDenied`	"Access was denied while trying to call apis on the underlying Kinesis Stream. Ensure that the iam role is propagated and valid."
`Kinesis.KMS.AccessDeniedException`	"Firehose does not have access to the KMS Key used to encrypt/decrypt the Kinesis Stream. Please grant the Firehose delivery role access to the key."
`Kinesis.KMS.KeyDisabled`	"Firehose is unable to read from the source Kinesis Stream because the KMS key used to encrypt/decrypt it is disabled. Enable the key so that reads can proceed."
`Kinesis.KMS.InvalidStateException`	"Firehose is unable to read from the source Kinesis Stream because the KMS key used to encrypt it is in an invalid state."

Error Code	Error Message and Information
`Kinesis.KMS.NotFoundException`	"Firehose is unable to read from the source Kinesis Stream because the KMS key used to encrypt it was not found."

Kinesis DirectPut Invocation Errors

Amazon Data Firehose can send the following Kinesis DirectPut invocation errors to CloudWatch Logs.

Error Code	Error Message and Information
`Firehose.KMS.AccessDeniedException`	"Firehose does not have access to the KMS Key. Please check the key policy."
`Firehose.KMS.InvalidStateException`	"Firehose is unable to decrypt the data because the KMS key used to encrypt it is in an invalid state."
`Firehose.KMS.NotFoundException`	"Firehose is unable to decrypt the data because the KMS key used to encrypt it was not found."
`Firehose.KMS.KeyDisabled`	"Firehose is unable to decrypt the data because the KMS key used to encrypt the data is disabled. Enable the key so that data delivery can proceed."

Amazon Glue Invocation Errors

Amazon Data Firehose can send the following Amazon Glue invocation errors to CloudWatch Logs.

Error Code	Error Message and Information
DataFormatConversion.InvalidSchema	"The schema is invalid."
DataFormatConversion.EntityNotFound	"The specified table/database could not be found. Please ensure that the table/database exists and that the values provided in the schema configuration are correct, especially with regards to casing."
DataFormatConversion.InvalidInput	"Could not find a matching schema from glue. Please make sure the specified database with the supplied catalog ID exists."
DataFormatConversion.InvalidInput	"Could not find a matching schema from glue. Please make sure the passed ARN is in the correct format."
DataFormatConversion.InvalidInput	"Could not find a matching schema from glue. Please make sure the catalogId provided is valid."
DataFormatConversion.InvalidVersionId	"Could not find a matching schema from glue. Please make sure the specified version of the table exists."
DataFormatConversion.NonExistentColumns	"Could not find a matching schema from glue. Please make sure the table is configured with a non-null storage descriptor containing the target columns."

Error Code	Error Message and Information
DataFormatConversion.AccessDenied	"Access was denied when assuming role. Please ensure that the role specified in the data format conversion configuration has granted the Firehose service permission to assume it."
DataFormatConversion.ThrottledByGlue	"Throttling error encountered when calling Glue. Either increase the request rate limit or reduce the current rate of calling glue through other applications."
DataFormatConversion.AccessDenied	"Access was denied when calling Glue. Please ensure that the role specified in the data format conversion configuration has the necessary permissions."
DataFormatConversion.InvalidGlueRole	"Invalid role. Please ensure that the role specified in the data format conversion configuration exists."
DataFormatConversion.InvalidGlueRole	"The security token included in the request is invalid. Ensure that the provided IAM role associated with firehose is not deleted."
DataFormatConversion.GlueNotAvailableInRegion	"Amazon Glue is not yet available in the region you have specified; please specify a different region."
DataFormatConversion.GlueEncryptionException	"There was an error retrieving the master key. Ensure that the key exists and has the correct access permissions."

Error Code	Error Message and Information
`DataFormatConversion.SchemaValidationTimeout`	"Timed out while retrieving table from Glue. If you have a large number of Glue table versions, please add 'glue:GetTableVersion' permission (recommended) or delete unused table versions. If you do not have a large number of tables in Glue, please contact Amazon Support."
`DataFirehose.InternalError`	"Timed out while retrieving table from Glue. If you have a large number of Glue table versions, please add 'glue:GetTableVersion' permission (recommended) or delete unused table versions. If you do not have a large number of tables in Glue, please contact Amazon Support."
`DataFormatConversion.GlueEncryptionException`	"There was an error retrieving the master key. Ensure that the key exists and state is correct."

DataFormatConversion Invocation Errors

Amazon Data Firehose can send the following DataFormatConversion invocation errors to CloudWatch Logs.

Error Code	Error Message and Information
`DataFormatConversion.InvalidSchema`	"The schema is invalid."
`DataFormatConversion.ValidationException`	"Column names and types must be non-empty strings."

Error Code	Error Message and Information
DataFormatConversion.ParseError	"Encountered malformed JSON."
DataFormatConversion.MalformedData	"Data does not match the schema."
DataFormatConversion.MalformedData	"Length of json key must not be greater than 262144"
DataFormatConversion.MalformedData	"The data cannot be decoded as UTF-8."
DataFormatConversion.MalformedData	"Illegal character found between tokens."
DataFormatConversion.InvalidTypeFormat	"The type format is invalid. Check the type syntax."
DataFormatConversion.InvalidSchema	"Invalid Schema. Please ensure that there are no special characters or whitespaces in column names."

Error Code	Error Message and Information
DataFormatConversion.InvalidRecord	"Record is not as per schema. One or more map keys were invalid for map<string,string>."
DataFormatConversion.MalformedData	"The input JSON contained a primitive at the top level. The top level must be an object or array."
DataFormatConversion.MalformedData	"The input JSON contained a primitive at the top level. The top level must be an object or array."
DataFormatConversion.MalformedData	"The record was empty or contained only whitespace."
DataFormatConversion.MalformedData	"Encountered invalid characters."
DataFormatConversion.MalformedData	"Encountered invalid or unsupported timestamp format. Please see the Firehose developer guide for supported timestamp formats."
DataFormatConversion.MalformedData	"A scalar type was found in the data but a complex type was specified on the schema."

Error Code	Error Message and Information
DataFormatConversion.MalformedData	"Data does not match the schema."
DataFormatConversion.MalformedData	"A scalar type was found in the data but a complex type was specified on the schema."
DataFormatConversion.ConversionFailureException	"ConversionFailureException"
DataFormatConversion.DataFormatConversionCustomerErrorException	"DataFormatConversionCustomerErrorException"
DataFormatConversion.DataFormatConversionCustomerErrorException	"DataFormatConversionCustomerErrorException"

Error Code	Error Message and Information
DataFormatConversion.MalformedData	"Data does not match the schema."
DataFormatConversion.InvalidSchema	"The schema is invalid."
DataFormatConversion.MalformedData	"Data does not match the schema. Invalid format for one or more dates."
DataFormatConversion.MalformedData	"Data contains a highly nested JSON structure that is not supported."
DataFormatConversion.EntityNotFound	"The specified table/database could not be found. Please ensure that the table/database exists and that the values provided in the schema configuration are correct, especially with regards to casing."
DataFormatConversion.InvalidInput	"Could not find a matching schema from glue. Please make sure the specified database with the supplied catalog ID exists."
DataFormatConversion.InvalidInput	"Could not find a matching schema from glue. Please make sure the passed ARN is in the correct format."

Error Code	Error Message and Information
DataFormatConversion.InvalidInput	"Could not find a matching schema from glue. Please make sure the catalogId provided is valid."
DataFormatConversion.InvalidVersionId	"Could not find a matching schema from glue. Please make sure the specified version of the table exists."
DataFormatConversion.NonExistentColumns	"Could not find a matching schema from glue. Please make sure the table is configured with a non-null storage descriptor containing the target columns."
DataFormatConversion.AccessDenied	"Access was denied when assuming role. Please ensure that the role specified in the data format conversion configuration has granted the Firehose service permission to assume it."
DataFormatConversion.ThrottledByGlue	"Throttling error encountered when calling Glue. Either increase the request rate limit or reduce the current rate of calling glue through other applications."
DataFormatConversion.AccessDenied	"Access was denied when calling Glue. Please ensure that the role specified in the data format conversion configuration has the necessary permissions."
DataFormatConversion.InvalidGlueRole	"Invalid role. Please ensure that the role specified in the data format conversion configuration exists."

Error Code	Error Message and Information
DataFormatConversion.GlueNotAvailableInRegion	"Amazon Glue is not yet available in the region you have specified; please specify a different region."
DataFormatConversion.GlueEncryptionException	"There was an error retrieving the master key. Ensure that the key exists and has the correct access permissions."
DataFormatConversion.SchemaValidationTimeout	"Timed out while retrieving table from Glue. If you have a large number of Glue table versions, please add 'glue:GetTableVersion' permission (recommended) or delete unused table versions. If you do not have a large number of tables in Glue, please contact Amazon Support."
DataFirehose.InternalError	"Timed out while retrieving table from Glue. If you have a large number of Glue table versions, please add 'glue:GetTableVersion' permission (recommended) or delete unused table versions. If you do not have a large number of tables in Glue, please contact Amazon Support."
DataFormatConversion.MalformedData	"One or more fields have incorrect format."

Accessing CloudWatch Logs for Amazon Data Firehose

You can view the error logs related to Amazon Data Firehose data delivery failure using the Amazon Data Firehose console or the CloudWatch console. The following procedures show you how to access error logs using these two methods.

To access error logs using the Amazon Data Firehose console

1. Sign in to the Amazon Web Services Management Console and open the Firehose console at https://console.aws.amazon.com/firehose
2. On the navigation bar, choose an Amazon Region.
3. Choose a Firehose stream name to go to the Firehose stream details page.
4. Choose **Error Log** to view a list of error logs related to data delivery failure.

To access error logs using the CloudWatch console

1. Open the CloudWatch console at https://console.amazonaws.cn/cloudwatch/.
2. On the navigation bar, choose a Region.
3. In the navigation pane, choose **Logs**.
4. Choose a log group and log stream to view a list of error logs related to data delivery failure.

Monitoring Kinesis Agent Health

Kinesis Agent publishes custom CloudWatch metrics with a namespace of **AmazonKinesisAgent**. It helps assess whether the agent is healthy, submitting data into Amazon Data Firehose as specified, and consuming the appropriate amount of CPU and memory resources on the data producer.

Metrics such as number of records and bytes sent are useful to understand the rate at which the agent is submitting data to the Firehose stream. When these metrics fall below expected thresholds by some percentage or drop to zero, it could indicate configuration issues, network errors, or agent health issues. Metrics such as on-host CPU and memory consumption and agent error counters indicate data producer resource usage, and provide insights into potential configuration or host errors. Finally, the agent also logs service exceptions to help investigate agent issues.

The agent metrics are reported in the region specified in the agent configuration setting `cloudwatch.endpoint`. For more information, see Agent Configuration Settings.

Cloudwatch metrics published from multiple Kinesis Agents are aggregated or combined.

There is a nominal charge for metrics emitted from Kinesis Agent, which are enabled by default. For more information, see Amazon CloudWatch Pricing.

Monitoring with CloudWatch

Kinesis Agent sends the following metrics to CloudWatch.

Metric	Description
BytesSent	The number of bytes sent to the Firehose stream over the specified time period. Units: Bytes
RecordSendAttempts	The number of records attempted (either first time, or as a retry) in a call to PutRecordBatch over the specified time period. Units: Count
RecordSendErrors	The number of records that returned failure status in a call to PutRecordBatch , including retries, over the specified time period. Units: Count
ServiceErrors	The number of calls to PutRecordBatch that resulted in a service error (other than a throttling error) over the specified time period. Units: Count

Logging Amazon Data Firehose API Calls with Amazon CloudTrail

Amazon Data Firehose is integrated with Amazon CloudTrail, a service that provides a record of actions taken by a user, role, or an Amazon service in Amazon Data Firehose. CloudTrail captures all API calls for Amazon Data Firehose as events. The calls captured include calls from the Amazon Data Firehose console and code calls to the Amazon Data Firehose API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Data Firehose. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Data Firehose, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the Amazon CloudTrail User Guide.

Amazon Data Firehose Information in CloudTrail

CloudTrail is enabled on your Amazon account when you create the account. When supported event activity occurs in Amazon Data Firehose, that activity is recorded in a CloudTrail event along with other Amazon service events in **Event history**. You can view, search, and download recent events in your Amazon account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your Amazon account, including events for Amazon Data Firehose, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all Amazon Regions. The trail logs events from all Regions in the Amazon partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other Amazon services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon Data Firehose supports logging the following actions as events in CloudTrail log files:

- CreateDeliveryStream
- DeleteDeliveryStream
- DescribeDeliveryStream
- ListDeliveryStreams
- ListTagsForDeliveryStream
- TagDeliveryStream
- StartDeliveryStreamEncryption
- StopDeliveryStreamEncryption
- UntagDeliveryStream

- UpdateDestination

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or Amazon Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another Amazon service.

For more information, see the CloudTrail userIdentity Element.

Example: Amazon Data Firehose Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateDeliveryStream`, `DescribeDeliveryStream`, `ListDeliveryStreams`, `UpdateDestination`, and `DeleteDeliveryStream` actions.

```
{
  "Records":[
        {
            "eventVersion":"1.02",
            "userIdentity":{
                "type":"IAMUser",
                "principalId":"AKIAIOSFODNN7EXAMPLE",
                "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
                "accountId":"111122223333",
                "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
                "userName":"CloudTrail_Test_User"
            },
            "eventTime":"2016-02-24T18:08:22Z",
            "eventSource":"firehose.amazonaws.com",
            "eventName":"CreateDeliveryStream",
            "awsRegion":"us-east-1",
```

```
            "sourceIPAddress":"127.0.0.1",
            "userAgent":"aws-internal/3",
            "requestParameters":{
                "deliveryStreamName":"TestRedshiftStream",
                "redshiftDestinationConfiguration":{
                "s3Configuration":{
                    "compressionFormat":"GZIP",
                    "prefix":"prefix",
                    "bucketARN":"arn:aws:s3:::firehose-cloudtrail-test-bucket",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    }
                },
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "password":"",
                "username":"",
                "roleARN":"arn:aws:iam::111122223333:role/Firehose"
            }
        },
        "responseElements":{
            "deliveryStreamARN":"arn:aws:firehose:us-
east-1:111122223333:deliverystream/TestRedshiftStream"
        },
        "requestID":"958abf6a-db21-11e5-bb88-91ae9617edf5",
        "eventID":"875d2d68-476c-4ad5-bbc6-d02872cfc884",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
```

```
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:08:54Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DescribeDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"aa6ea5ed-db21-11e5-bb88-91ae9617edf5",
        "eventID":"d9b285d8-d690-4d5c-b9fe-d1ad5ab03f14",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:00Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"ListDeliveryStreams",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "limit":10
        },
        "responseElements":null,
        "requestID":"d1bf7f86-db21-11e5-bb88-91ae9617edf5",
        "eventID":"67f63c74-4335-48c0-9004-4ba35ce00128",
        "eventType":"AwsApiCall",
```

```
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:09Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"UpdateDestination",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "destinationId":"destinationId-000000000001",
            "deliveryStreamName":"TestRedshiftStream",
            "currentDeliveryStreamVersionId":"1",
            "redshiftDestinationUpdate":{
                "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "password":"",
                "username":"",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "s3Update":{
                    "bucketARN":"arn:aws:s3:::firehose-cloudtrail-test-bucket-update",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "compressionFormat":"GZIP",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
```

```
                    },
                    "prefix":"arn:aws:s3:::firehose-cloudtrail-test-bucket"
                }
            }
        },
        "responseElements":null,
        "requestID":"d549428d-db21-11e5-bb88-91ae9617edf5",
        "eventID":"1cb21e0b-416a-415d-bbf9-769b152a6585",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:12Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DeleteDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"d85968c1-db21-11e5-bb88-91ae9617edf5",
        "eventID":"dd46bb98-b4e9-42ff-a6af-32d57e636ad1",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    }
  ]
}
```

Custom Prefixes for Amazon S3 Objects

Objects delivered to Amazon S3 follow the name format of <evaluated prefix><suffix>. You can specify your custom prefix that includes expressions that are evaluated at runtime. Custom prefix you specify will override the default prefix of `YYYY/MM/dd/HH`.

You can use expressions of the following forms in your custom prefix: `!{namespace:value}`, where `namespace` can be one of the following, as explained in the following sections.

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

If a prefix ends with a slash, it appears as a folder in the Amazon S3 bucket. For more information, see Amazon S3 Object Name Format in the *Amazon Data FirehoseDeveloper Guide*.

The `timestamp` namespace

Valid values for this namespace are strings that are valid Java DateTimeFormatter strings. As an example, in the year 2018, the expression `!{timestamp:yyyy}` evaluates to `2018`.

When evaluating timestamps, Firehose uses the approximate arrival timestamp of the oldest record that's contained in the Amazon S3 object being written.

By default, timestamp is in UTC. But, you can specify a time zone that you prefer. For example, you can configure the time zone to Asia/Tokyo in the Amazon Web Services Management Console or in API parameter setting (CustomTimeZone) if you want to use Japan Standard Time instead of UTC. To see the list of supported time zones, see Amazon S3 Object Name Format.

If you use the `timestamp` namespace more than once in the same prefix expression, every instance evaluates to the same instant in time.

The `firehose` namespace

There are two values that you can use with this namespace: `error-output-type` and `random-string`. The following table explains how to use them.

The `firehose` namespace values

Conversion	Description	Example input	Example output	Notes
`error-output-type`	Evaluates to one of the following strings, depending on the configuration of your delivery stream, and the reason of failure: {processing-failed, AmazonOpenSearchService-failed, splunk-failed, format-conversion-failed, http-endpoint-failed}. If you use it more than once in the same expression, every instance evaluates to the same error string..	`myPrefix/result=!{firehose:error-output-type}/!{timestamp:yyyy/MM/dd}`	`myPrefix/result=processing-failed/2018/08/03`	The error-output-type value can only be used in the ErrorOutputPrefix field.
`random-string`	Evaluates to a random string of 11 characters. If you use it more than once in the same expressio	`myPrefix/!{firehose:random-string}/`	`myPrefix/046b6c7f-0b/`	You can use it with both prefix types. You can place it at the beginning

Conversion	Description	Example input	Example output	Notes
	n, every instance evaluates to a new random string.			of the format string to get a randomized prefix, which is sometimes necessary for attaining extremely high throughput with Amazon S3.

`partitionKeyFromLambda` and `partitionKeyFromQuery` namespaces

For dynamic partitioning, you must use the following expression format in your S3 bucket prefix: `!{namespace:value}`, where namespace can be either `partitionKeyFromQuery` or `partitionKeyFromLambda`, or both. If you are using inline parsing to create the partitioning keys for your source data, you must specify an S3 bucket prefix value that consists of expressions specified in the following format: `"partitionKeyFromQuery:keyID"`. If you are using an Amazon Lambda function to create partitioning keys for your source data, you must specify an S3 bucket prefix value that consists of expressions specified in the following format: `"partitionKeyFromLambda:keyID"`. For more information, see the "Choose Amazon S3 for Your Destination" in Creating an Amazon Data FirehoseDelivery Stream.

Semantic rules

The following rules apply to `Prefix` and `ErrorOutputPrefix` expressions.

- For the `timestamp` namespace, any character that isn't in single quotes is evaluated. In other words, any string escaped with single quotes in the value field is taken literally.
- If you specify a prefix that doesn't contain a timestamp namespace expression, Firehose appends the expression `!{timestamp:yyyy/MM/dd/HH/}`to the value in the `Prefix` field.
- The sequence `!{` can only appear in `!{namespace:value}` expressions.

- `ErrorOutputPrefix` can be null only if `Prefix` contains no expressions. In this case, `Prefix` evaluates to `<specified-prefix>yyyy/MM/DDD/HH/` and `ErrorOutputPrefix` evaluates to `<specified-prefix><error-output-type>YYYY/MM/DDD/HH/`. `DDD` represents the day of the year.
- If you specify an expression for `ErrorOutputPrefix`, you must include at least one instance of `!{firehose:error-output-type}`.
- `Prefix` can't contain `!{firehose:error-output-type}`.
- Neither `Prefix` nor `ErrorOutputPrefix` can be greater than 512 characters after they're evaluated.
- If the destination is Amazon Redshift, `Prefix` must not contain expressions and `ErrorOutputPrefix` must be null.
- When the destination is Amazon OpenSearch Service or Splunk, and no `ErrorOutputPrefix` is specified, Firehose uses the `Prefix` field for failed records.
- When the destination is Amazon S3, the `Prefix` and `ErrorOutputPrefix` in the Amazon S3 destination configuration are used for successful records and failed records, respectively. If you use the Amazon CLI or the API, you can use `ExtendedS3DestinationConfiguration` to specify an Amazon S3 *backup* configuration with its own `Prefix` and `ErrorOutputPrefix`.
- When you use the Amazon Web Services Management Console and set the destination to Amazon S3, Firehose uses the `Prefix` and `ErrorOutputPrefix` in the destination configuration for successful records and failed records, respectively. If you specify a prefix but no error prefix, Firehose automatically sets the error prefix to `!{firehose:error-output-type}/`.
- When you use `ExtendedS3DestinationConfiguration` with the Amazon CLI, the API, or Amazon CloudFormation, if you specify a `S3BackupConfiguration`, Firehose doesn't provide a default `ErrorOutputPrefix`.
- You cannot use `partitionKeyFromLambda` and `partitionKeyFromQuery` namespaces when creating ErrorOutputPrefix expressions.

Example prefixes

`Prefix` and `ErrorOutputPrefix` examples

Input	Evaluated prefix (at 10:30 AM UTC on Aug 27, 2018)

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

`Prefix`: `!{timestamp:yyyy/MM/dd}` `ErrorOutputPrefix` : Unspecified	Invalid input: `ErrorOutputPrefix` can't be null when Prefix contains expressions

`Prefix`: `myFirehose/DeliveredYear=!{timestamp:yyyy}/anyMonth/rand=!{firehose:random-string}`
 	
`ErrorOutputPrefix` : `myFirehoseFailures/!{firehose:error-output-type}/!{timestamp:yyyy}/anyMonth/!{timestamp:dd}`
	`Prefix`: `myFirehose/DeliveredYear=2018/anyMonth/rand=5abf82daaa5`
	
	`ErrorOutputPrefix` : `myFirehoseFailures/processing-failed/2018/anyMonth/10`

`Prefix`: `myPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/`
 	
`ErrorOutputPrefix` : `myErrorPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/!{firehose:error-output-type}`
	`Prefix`: `myPrefix/year=2018/month=07/day=06/hour=23/`
	
	`ErrorOutputPrefix` : `myErrorPrefix/year=2018/month=07/day=06/hour=23/processing-failed`

`Prefix`: `myFirehosePrefix/` `ErrorOutputPrefix` : Unspecified	`Prefix`: `myFirehosePrefix/2018/08/27/`

Input	Evaluated prefix (at 10:30 AM UTC on Aug 27, 2018)
	`ErrorOutputPrefix :myFirehosePrefix/processing-failed/2018/08/27/`

Using Amazon Data Firehose with Amazon PrivateLink

Interface VPC endpoints (Amazon PrivateLink) for Amazon Data Firehose

You can use an interface VPC endpoint to keep traffic between your Amazon VPC and Amazon Data Firehose from leaving the Amazon network. Interface VPC endpoints don't require an internet gateway, NAT device, VPN connection, or Amazon Direct Connect connection. Interface VPC endpoints are powered by Amazon PrivateLink, an Amazon technology that enables private communication between Amazon services using an elastic network interface with private IPs in your Amazon VPC. For more information, see Amazon Virtual Private Cloud.

Using interface VPC endpoints (Amazon PrivateLink) for Amazon Data Firehose

To get started, create an interface VPC endpoint in order for your Amazon Data Firehose traffic from your Amazon VPC resources to start flowing through the interface VPC endpoint. When you create an endpoint, you can attach an endpoint policy to it that controls access to Amazon Data Firehose. For more about using policies to control access from a VPC endpoint to Amazon Data Firehose, see Controlling Access to Services with VPC Endpoints.

The following example shows how you can set up an Amazon Lambda function in a VPC and create a VPC endpoint to allow the function to communicate securely with the Amazon Data Firehose service. In this example, you use a policy that allows the Lambda function to list the Firehose streams in the current Region but not to describe any Firehose stream.

Create a VPC endpoint

1. Sign in to the Amazon Web Services Management Console and open the Amazon VPC console at https://console.amazonaws.cn/vpc/.
2. In the VPC Dashboard choose **Endpoints**.
3. Choose **Create Endpoint**.
4. In the list of service names, choose `com.amazonaws.`*`your_region`*`.kinesis-firehose`.
5. Choose the VPC and one or more subnets in which to create the endpoint.
6. Choose one or more security groups to associate with the endpoint.

7. For **Policy**, choose **Custom** and paste the following policy:

```
{
    "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. Choose **Create endpoint**.

Create an IAM role to use with the Lambda function

1. Open the IAM console at https://console.amazonaws.cn/iam/.
2. In the left pane, chose **Roles** and then choose **Create role**.
3. Under **Select type of trusted entity**, leave the default selection **Amazon service**.
4. Under **Choose the service that will use this role**, choose **Lambda**.
5. Choose **Next: Permissions**.
6. In the list of policies, search for and add the two policies named `AmazonLambdaVPCAccessExecutionRole` and `AmazonDataFirehoseReadOnlyAccess`.

> **Important**
>
> This is an example. You might need stricter policies for your production environment.

7. Choose **Next: Tags**. You don't need to add tags for the purpose of this exercise. Choose **Next: Review**.
8. Enter a name for the role, then choose **Create role**.

Create a Lambda function inside the VPC

1. Open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.
2. Choose **Create function**.
3. Choose **Author from scratch**.
4. Enter a name for the function, then set **Runtime** to Python 3.6.
5. Under **Permissions**, expand **Choose or create an execution role**.
6. In the **Execution role** list, choose **Use an existing role**.
7. In the **Existing role** list, choose the role you created above.
8. Choose **Create function**.
9. Under **Function code**, paste the following code.

```
    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. Under **Basic settings**, set the timeout to 1 minute.
11. Under **Network**, choose the VPC where you created the endpoint above, then choose the subnets and security group that you associated with the endpoint when you created it.
12. Near the top of the page, choose **Save**.
13. Choose **Test**.
14. Enter an event name, then choose **Create**.
15. Choose **Test** again. This causes the function to run. After the execution result appears, expand **Details** and compare the log output to the function code. Successful results show a list of the Firehose streams in the Region, as well as the following output:

    ```
    Calling describe_delivery_stream.

    AccessDeniedException

    Access denied test succeeded.
    ```

Availability

Interface VPC endpoints are currently supported within the following Regions:

- US East (Ohio)

- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Asia Pacific (Hong Kong)
- Canada (Central)
- Canada West (Calgary)
- China (Beijing)
- China (Ningxia)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- South America (São Paulo)
- Amazon GovCloud (US-East)
- Amazon GovCloud (US-West)
- Europe (Spain)
- Middle East (UAE)
- Asia Pacific (Jakarta)
- Asia Pacific (Osaka)
- Israel (Tel Aviv)

Tagging Your Firehose streams in Amazon Data Firehose

You can assign your own metadata to Firehose streams that you create in Amazon Data Firehose in the form of *tags*. A tag is a key-value pair that you define for a stream. Using tags is a simple yet powerful way to manage Amazon resources and organize data, including billing data.

Topics

Tag Basics

You can use the Amazon Data Firehose API to complete the following tasks:

- Add tags to a Firehose stream.
- List the tags for your Firehose streams.
- Remove tags from a Firehose stream.

You can use tags to categorize your Firehose streams. For example, you can categorize Firehose streams by purpose, owner, or environment. Because you define the key and value for each tag, you can create a custom set of categories to meet your specific needs. For example, you might define a set of tags that helps you track Firehose streams by owner and associated application.

The following are several examples of tags:

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`
- `Application: Application name`
- `Environment: Production`

If you specify tags in the `CreateDeliveryStream` action, Amazon Data Firehose performs an additional authorization on the `firehose:TagDeliveryStream` action to verify if users have permissions to create tags. If you do not provide this permission, requests to create new Firehose delivery streams with IAM resource tags will fail with an `AccessDeniedException` such as following.

```
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.
```

The following example demonstrates a policy that allows users to create a delivery stream and apply tags.

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

Tracking Costs Using Tagging

You can use tags to categorize and track your Amazon costs. When you apply tags to your Amazon resources, including Firehose streams, your Amazon cost allocation report includes usage and costs aggregated by tags. You can organize your costs across multiple services by applying tags that represent business categories (such as cost centers, application names, or owners). For more information, see Use Cost Allocation Tags for Custom Billing Reports in the *Amazon Billing User Guide*.

Tag Restrictions

The following restrictions apply to tags in Amazon Data Firehose.

Basic restrictions

- The maximum number of tags per resource (stream) is 50.
- Tag keys and values are case-sensitive.
- You can't change or edit tags for a deleted stream.

Tag key restrictions

- Each tag key must be unique. If you add a tag with a key that's already in use, your new tag overwrites the existing key-value pair.
- You can't start a tag key with `aws:` because this prefix is reserved for use by Amazon. Amazon creates tags that begin with this prefix on your behalf, but you can't edit or delete them.
- Tag keys must be between 1 and 128 Unicode characters in length.
- Tag keys must consist of the following characters: Unicode letters, digits, white space, and the following special characters: `_ . / = + - @`.

Tag value restrictions

- Tag values must be between 0 and 255 Unicode characters in length.
- Tag values can be blank. Otherwise, they must consist of the following characters: Unicode letters, digits, white space, and any of the following special characters: `_ . / = + - @`.

Tagging Firehose streams Using the Amazon Data Firehose API

You can specify tags when you invoke CreateDeliveryStream to create a new Firehose stream. For existing delivery streams, you can add, list, and remove tags using the following three operations:

- TagDeliveryStream
- ListTagsForDeliveryStream
- UntagDeliveryStream

Tutorial: Ingest VPC flow logs into Splunk using Amazon Data Firehose

For a tutorial, see Ingest VPC flow logs into Splunk using Amazon Data Firehose.

Troubleshooting Amazon Data Firehose

If Firehose encounters errors while delivering or processing data, it retries until the configured retry duration expires. If the retry duration ends before the data is delivered successfully, Firehose backs up the data to the configured S3 backup bucket. If the destination is Amazon S3 and delivery fails or if delivery to the backup S3 bucket fails, Firehose keeps retrying until the retention period ends. For `DirectPut` delivery streams, Firehose retains the records for 24 hours. For a delivery stream whose data source is a Kinesis data stream, you can change the retention period as described in Changing the Data Retention Period.

If the data source is a Kinesis data stream, Firehose retries the following operations indefinitely: `DescribeStream`, `GetRecords`, and `GetShardIterator`.

If the delivery stream uses `DirectPut`, check the `IncomingBytes` and `IncomingRecords` metrics to see if there's incoming traffic. If you are using the `PutRecord` or `PutRecordBatch`, make sure you catch exceptions and retry. We recommend a retry policy with exponential back-off with jitter and several retries. Also, if you use the `PutRecordBatch` API, make sure your code checks the value of FailedPutCount in the response even when the API call succeeds.

If the delivery stream uses a Kinesis data stream as its source, check the `IncomingBytes` and `IncomingRecords` metrics for the source data stream. Additionally, ensure that the `DataReadFromKinesisStream.Bytes` and `DataReadFromKinesisStream.Records` metrics are being emitted for the delivery stream.

For information about tracking delivery errors using CloudWatch, see the section called "Monitoring with CloudWatch Logs".

Issues

- Troubleshooting Amazon S3
- Troubleshooting Amazon Redshift
- Troubleshooting Amazon OpenSearch Service
- Troubleshooting Splunk
- Troubleshooting Snowflake
- Troubleshooting Firehose endpoint reachability
- Troubleshooting HTTP Endpoints
- Troubleshooting MSK As Source

- Other

Troubleshooting Amazon S3

Check the following if data is not delivered to your Amazon Simple Storage Service (Amazon S3) bucket.

- Check the Firehose `IncomingBytes` and `IncomingRecords` metrics to make sure that data is sent to your Firehose stream successfully. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Metrics.
- If data transformation with Lambda is enabled, check the Firehose `ExecuteProcessingSuccess` metric to make sure that Firehose has tried to invoke your Lambda function. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Metrics.
- Check the Firehose `DeliveryToS3.Success` metric to make sure that Firehose has tried putting data to your Amazon S3 bucket. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Metrics.
- Enable error logging if it is not already enabled, and check error logs for delivery failure. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Logs.
- If you see an error message in the log saying *"Firehose encountered InternalServerError when calling Amazon S3 service. The operation will be retried; if the error persists, please contact S3 for resolution."*, it could be due to the significant increase in request rates on a single partition in S3. You can optimize S3 prefix design patterns to mitigate the issue. For more information, see Best practices design patterns: optimizing Amazon S3 performance. If this does not resolve the issue, contact Amazon Support for further assistance.
- Make sure that the Amazon S3 bucket that is specified in your Firehose stream still exists.
- If data transformation with Lambda is enabled, make sure that the Lambda function that is specified in your delivery stream still exists.
- Make sure that the IAM role that is specified in your Firehose stream has access to your S3 bucket and your Lambda function (if data transformation is enabled). Also, make sure that the IAM role has access to CloudWatch log group and log streams to check error logs. For more information, see Grant Amazon Data Firehose Access to an Amazon S3 Destination.
- If you're using data transformation, make sure that your Lambda function never returns responses whose payload size exceeds 6 MB. For more information, see Amazon Data FirehoseData Transformation.

Troubleshooting Amazon Redshift

Check the following if data is not delivered to your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup.

Data is delivered to your S3 bucket before loading into Amazon Redshift. If the data was not delivered to your S3 bucket, see Troubleshooting Amazon S3.

- Check the Firehose `DeliveryToRedshift.Success` metric to make sure that Firehose has tried to copy data from your S3 bucket to the Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Metrics.
- Enable error logging if it is not already enabled, and check error logs for delivery failure. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Logs.
- Check the Amazon Redshift `STL_CONNECTION_LOG` table to see if Firehose can make successful connections. In this table, you should be able to see connections and their status based on a user name. For more information, see `STL_CONNECTION_LOG` in the *Amazon Redshift Database Developer Guide*.
- If the previous check shows that connections are being established, check the Amazon Redshift `STL_LOAD_ERRORS` table to verify the reason for the COPY failure. For more information, see `STL_LOAD_ERRORS` in the *Amazon Redshift Database Developer Guide*.
- Make sure that the Amazon Redshift configuration in your Firehose stream is accurate and valid.
- Make sure that the IAM role that is specified in your Firehose stream can access the S3 bucket that Amazon Redshift copies data from, and also the Lambda function for data transformation (if data transformation is enabled). Also, make sure that the IAM role has access to CloudWatch log group and log streams to check error logs. For more information, see Grant Amazon Data Firehose Access to an Amazon Redshift Destination .
- If your Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup is in a virtual private cloud (VPC), make sure that the cluster allows access from Firehose IP addresses. For more information, see Grant Amazon Data Firehose Access to an Amazon Redshift Destination .
- Make sure that the Amazon Redshift provisioned cluster or Amazon Redshift Serverless workgroup is publicly available.
- If you're using data transformation, make sure that your Lambda function never returns responses whose payload size exceeds 6 MB. For more information, see Amazon Data FirehoseData Transformation.

Troubleshooting Amazon OpenSearch Service

Check the following if data is not delivered to your OpenSearch Service domain.

Data can be backed up to your Amazon S3 bucket concurrently. If data was not delivered to your S3 bucket, see Troubleshooting Amazon S3.

- Check the Firehose `IncomingBytes` and `IncomingRecords` metrics to make sure that data is sent to your Firehose stream successfully. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Metrics.
- If data transformation with Lambda is enabled, check the Firehose `ExecuteProcessingSuccess` metric to make sure that Firehose has tried to invoke your Lambda function. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Metrics.
- Check the Firehose `DeliveryToAmazonOpenSearchService.Success` metric to make sure that Firehose has tried to index data to the OpenSearch Service cluster. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Metrics.
- Enable error logging if it is not already enabled, and check error logs for delivery failure. For more information, see Monitoring Amazon Data Firehose Using CloudWatch Logs.
- Make sure that the OpenSearch Service configuration in your delivery stream is accurate and valid.
- If data transformation with Lambda is enabled, make sure that the Lambda function that is specified in your delivery stream still exists. Also, make sure that the IAM role has access to CloudWatch log group and log streams to check error logs. For more information, see Grant FirehoseAccess to a Public OpenSearch Service Destination.
- Make sure that the IAM role that is specified in your delivery stream can access your OpenSearch Service cluster, S3 backup bucket, and Lambda function (if data transformation is enabled). Also, make sure that the IAM role has access to CloudWatch log group and log streams to check error logs. For more information, see Grant FirehoseAccess to a Public OpenSearch Service Destination.
- If you're using data transformation, make sure that your Lambda function never returns responses whose payload size exceeds 6 MB. For more information, see Amazon Data FirehoseData Transformation.
- Amazon Data Firehosecurrently does not support the delivery of CloudWatch Logs to Amazon OpenSearch Service destination because Amazon CloudWatch combines multiple log events into one Firehose record and Amazon OpenSearch Service cannot accept multiple log events in

one record. As an alternative, you can consider Using subscription filter for Amazon OpenSearch Service in CloudWatch Logs.

Troubleshooting Splunk

Check the following if data is not delivered to your Splunk endpoint.

- If your Splunk platform is in a VPC, make sure that Firehose can access it. For more information, see Access to Splunk in VPC.
- If you use an Amazon load balancer, make sure that it is a Classic Load Balancer or an Application Load Balancer. Also, enable duration-based sticky sessions with cookie expiration disabled for Classic Load Balancer and expiration is set to the maximum (7 days) for Application Load Balancer. For information about how to do this, see Duration-Based Session Stickiness for Classic Load Balancer or an Application Load Balancer.
- Review the Splunk platform requirements. The Splunk add-on for Firehose requires Splunk platform version 6.6.X or later. For more information, see Splunk Add-on for Amazon Kinesis Firehose.
- If you have a proxy (Elastic Load Balancing or other) between Firehose and the HTTP Event Collector (HEC) node, enable sticky sessions to support HEC acknowledgements (ACKs).
- Make sure that you are using a valid HEC token.
- Ensure that the HEC token is enabled. See Enable and disable Event Collector tokens.
- Check whether the data that you're sending to Splunk is formatted correctly. For more information, see Format events for HTTP Event Collector.
- Make sure that the HEC token and input event are configured with a valid index.
- When an upload to Splunk fails due to a server error from the HEC node, the request is automatically retried. If all retries fail, the data gets backed up to Amazon S3. Check if your data appears in Amazon S3, which is an indication of such a failure.
- Make sure that you enabled indexer acknowledgment on your HEC token. For more information, see Enable indexer acknowledgement.
- Increase the value of `HECAcknowledgmentTimeoutInSeconds` in the Splunk destination configuration of your Firehose delivery stream.
- Increase the value of `DurationInSeconds` under `RetryOptions` in the Splunk destination configuration of your Firehose delivery stream.

- Check your HEC health.
- If you're using data transformation, make sure that your Lambda function never returns responses whose payload size exceeds 6 MB. For more information, see Amazon Data FirehoseData Transformation.
- Make sure that the Splunk parameter named `ackIdleCleanup` is set to `true`. It is false by default. To set this parameter to `true`, do the following:
  - For a managed Splunk Cloud deployment, submit a case using the Splunk support portal. In this case, ask Splunk support to enable the HTTP event collector, set `ackIdleCleanup` to `true` in `inputs.conf`, and create or modify a load balancer to use with this add-on.
  - For a distributed Splunk Enterprise deployment, set the `ackIdleCleanup` parameter to true in the `inputs.conf` file. For *nix users, this file is located under `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/`. For Windows users, it is under `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\`.
  - For a single-instance Splunk Enterprise deployment, set the `ackIdleCleanup` parameter to `true` in the `inputs.conf` file. For *nix users, this file is located under `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/`. For Windows users, it is under `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\`.
- Make sure that the IAM role that is specified in your Firehosedelivery stream can access the S3 backup bucket and the Lambda function for data transformation (if data transformation is enabled). Also, make sure that the IAM role has access to CloudWatch Logs group and log streams to check error logs. For more information, see Grant FirehoseAccess to a Splunk Destination.
- See Troubleshoot the Splunk Add-on for Amazon Kinesis Firehose.

Troubleshooting Snowflake

This section describes common troubleshooting steps while using Snowflake as a destination

Firehose delivery stream creation fails

If delivery stream creation fails for a stream delivering data to a PrivateLink-enabled Snowflake Cluster, it indicates that the VPCE-ID is not reachable by Firehose. This can be due to one of the following reasons:

- Incorrect VPCE-ID. Confirm that there are no typographic errors.

- Firehose does not support region-less Snowflake URLs in preview. Provide the URL using Snowflake Account Locator. See Snowflake documentation for more details.
- Confirm that the Firehose delivery stream is created in the same Amazon Region as the Snowflake Region.
- If the issue persists, reach out to Amazon support.

Delivery failures

Check the following if data is not getting delivered to your Snowflake table. Snowflake delivery failed data will be delivered to the S3 error bucket along with an error code and an error message that corresponds to the payload. Following are few a common error scenarios. For the entire list of error codes, see Snowflake Data Delivery Errors.

- **Error code: Snowflake.DefaultRoleMissing**: Indicates that snowflake role is not configured while creating delivery stream. If Snowflake role is not configured, make sure you set a default role to the Snowflake user specified.
- **Error code: Snowflake.ExtraColumns**: Indicates that insert to Snowflake is rejected due to extra columns in the input payload. Columns not present in table shouldn't be specified. Note that Snowflake column names are case-sensitive. If the delivery is failing with this error despite column being present in table, make sure that the case of the column name in input payload matches the column name declared in table definition.
- **Error code: Snowflake.MissingColumns**: Indicates that insert to Snowflake is rejected due to missing columns in input payload. Make sure that values are specified for all non-nullable columns.
- **Error code: Snowflake.InvalidInput**: This could happen when Firehose failed to parse the input payload provided into valid JSON format. Make sure that the json payload is well formed, doesn't have extra double quotes, quotes, escape characters etc. Currently Firehose supports only single JSON item as record payload, JSON arrays are not supported.
- **Error code: Snowflake.InvalidValue**: Indicates that delivery failed due to incorrect data type in the input payload. Make sure that the JSON values specified in input payload adhere to the datatype declared in Snowflake table definition.
- **Error code: Snowflake.InvalidTableType**: Indicates that table type configured in the delivery stream is not supported. Refer to the limitations at Limitations) of snowpipe streaming for the supported tables, columns and data types.

Note

For any reason, if the table definition or role permissions are changed on your Snowflake destination after creating the delivery stream, it can take several minutes for Firehose to detect those changes. If you are seeing delivery errors due to this, try deleting and recreating the delivery stream.

Troubleshooting Firehose endpoint reachability

If the Firehose API encounters a timeout, perform the following steps to test endpoint reachability:

- Check if API requests are made from a host in a VPC. All traffic from a VPC requires setting up a Firehose VPC endpoint. For more information, see Using Firehose with Amazon PrivateLink.
- If traffic is coming from a public network or VPC with the Firehose VPC endpoint set up in a particular subnet, run the following commands from the host to check network connectivity. The Firehose endpoint can be found at Firehose endpoints and quotas.
  - Use tools like **traceroute** or **tcping** to check if the network setup is correct. If that fails, check your network setting:

    For example:

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

    or

    ```
    tcping firehose.us-east-2.amazonaws.com 443
    ```

  - If it appears the network setting is correct and the following command fails, check whether the Amazon CA (Certficate Authority) is in the trust chain.

    For example:

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

  If the above commands succeed, try the API again to see if there is a response returned from the API.

Troubleshooting HTTP Endpoints

This section describes common troubleshooting steps when dealing with Amazon Data Firehose delivering data to generic HTTP Endpoints destinations and to partner destinations, including Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Splunk, or Sumo Logic. For the purposes of this section, all applicable destinations are referred to as HTTP endpoints. Make sure that the IAM role that is specified in your Firehose delivery stream can access the S3 backup bucket and the Lambda function for data transformation (if data transformation is enabled). Also, make sure that the IAM role has access to CloudWatch log group and log streams to check error logs. For more information, see Grant Firehose Access to an HTTP Endpoint Destination.

Note

The information in this section does not apply to the following destinations: Splunk, OpenSearch Service, S3, and Redshift.

CloudWatch Logs

It is highly recommended that you enable CloudWatch Logging for Firehose. Logs are only published when there are errors delivering to your destination.

Destination Exceptions

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"
}
```

Destination exceptions indicate that Firehose **is** able to establish a connection to your endpoint and make an HTTP request, but **did not** receive a 200 response code. 2xx responses that are not 200s will also result in a destination exception. Amazon Data Firehose logs the response code and a truncated response payload received from the configured endpoint to CloudWatch Logs. Because Amazon Data Firehose logs the response code and payload without modification or interpretation, it is up to the endpoint to provide the exact reason why it rejected Amazon Data Firehose's HTTP delivery request. The following are the most common troubleshooting recommendations for these exceptions:

- **400**: Indicates that you are sending a bad request due to a misconfiguration of your Amazon Data Firehose. Make sure that you have the correct url, common attributes, content encoding, access key, and buffering hints for your destination. See the destination specific documentation on the required configuration.
- **401**: Indicates that the access key you configured for your Firehose stream is incorrect or missing.
- **403**: Indicates that the access key you configured for your Firehose stream does not have permissions to deliver data to the configured endpoint.
- **413**: Indicates that the request payload that Amazon Data Firehose sends to the endpoint is too large for the endpoint to handle. Try lowering the buffering hint to the recommended size for your destination.
- **429**: Indicates that Amazon Data Firehose is sending requests at a greater rate than the destination can handle. Fine tune your buffering hint by increasing your buffering time and/or increasing your buffering size (but still within the limit of your destination).
- **5xx**: Indicates that there is a problem with the destination. The Amazon Data Firehose service is still working properly.

> **Important**
>
> Important: While these are the common troubleshooting recommendations, specific endpoints may have different reasons for providing the response codes and the endpoint specific recommendations should be followed first.

Invalid Response

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"
}
```

Invalid response exceptions indicate that Amazon Data Firehose received an invalid response from the endpoint destination. The response must conform to the response specifications or Amazon Data Firehose will consider the delivery attempt a failure and will redeliver the same data until the configured retry duration is exceeded. Amazon Data Firehose treats responses that do not follow the response specifications as failures even if the response has a 200 status. If you are developing a Amazon Data Firehose compatible endpoint, follow the response specifications to ensure data is successfully delivered.

Below are some of the common types of invalid responses and how to fix them:

- **Invalid JSON or Unexpected Fields**: Indicates that the response can not be properly deserialized as JSON or has unexpected fields. Ensure that the response is not content-encoded.
- **Missing RequestId**: Indicates that the response does not contain a requestId.
- **RequestId does not match**: Indicates that the requestId in the response does not match the outgoing requestId.
- **Missing Timestamp**: Indicates that the response does not contain a timestamp field. The timestamp field must be a number and not a string.
- **Missing Content-Type Header**: Indicates that the response does not contain a “content-type: application/json” header. No other content-type is accepted.

> **Important**
>
> Important: Amazon Data Firehose can only deliver data to endpoints that follow the Firehose request and response specifications. If you are configuring your destination

to a third party service, ensure that you are using the correct Amazon Data Firehose compatible endpoint which will likely be different than the public ingestion endpoint. For example Datadog's Amazon Data Firehose endpoint is https://aws-kinesis-http-intake.logs.datadoghq.com/ while its public endpoint is https://api.datadoghq.com/.

Other Common Errors

Additional error codes and definitions are listed below.

- **Error Code: HttpEndpoint.RequestTimeout** - Indicates that the endpoint took longer than 3 minutes to respond. If you are the owner of the destination, decrease the response time of the destination endpoint. If you are not the owner of the destination, contact the owner and ask if anything can be done to lower the response time (i.e. decrease the buffering hint so there is less data being processed per request).
- **Error Code: HttpEndpoint.ResponseTooLarge** - Indicates that the response is too large. The response must be less than 1 MiB including headers.
- **Error Code: HttpEndpoint.ConnectionFailed** - Indicates a connection could not be established with the configured endpoint. This could be due to a typo in the configured url, the endpoint not being accessible to Amazon Data Firehose, or the endpoint taking too long to respond to the connection request.
- **Error Code: HttpEndpoint.ConnectionReset** - Indicates a connection was made but reset or prematurely closed by the endpoint.
- **Error Code: HttpEndpoint.SSLHandshakeFailure** - Indicates an SSL handshake could not be successfully completed with the configured endpoint.

Troubleshooting MSK As Source

This section describes common troubleshooting steps while using MSK As Source

> **Note**
>
> For troubleshooting processing, transformation or S3 delivery issues, please refer the earlier sections

Hose creation fails

Check the following if your hose with MSK As Source is failing creation

- Check that the source MSK cluster is in Active state.
- If you are using Private connectivity, ensure that Private Link on the cluster is turned on

  If you are using Public connectivity, ensure that Public access on the cluster is turned on
- If you are using Private connectivity, make sure that you add a resource based policy that allows Firehose to create Private Link. Also refer: MSK cross account permissions
- Ensure that the role in source configuration has permission to ingest data from cluster's Topic
- Ensure that your VPC security groups allow incoming traffic on ports used by the cluster's bootstrap servers

Hose Suspended

Check the following if your hose is in SUSPENDED state

- Check that the source MSK cluster is in Active state.
- Check that the source topic exists. In case the topic was deleted and re-created, you will have to delete and re-create the Firehose Firehose stream as well.

Hose Backpresurred

The value of DataReadFromSource.Backpressured will be 1 when BytesPerSecondLimit per partition is exceeded or that the normal flow of delivery is slow or stopped.

- If you are hitting BytesPerSecondLimit please check DataReadFromSource.Bytes metric and request a limit increase.
- Check the CloudWatch logs, destination metrics, Data Transformation metrics and Format Conversion metrics to identify the bottlenecks.

Incorrect Data Freshness

Data freshness seems incorrect

- Firehose calculates the data freshness based on the timestamp of the consumed record. To ensure that this timestamp is correctly recorded when the producer record is persisted in the Kafka's broker logs, set the Kafka topic timestamp type configuration to be `message.timestamp.type=LogAppendTime`.

MSK cluster connection issues

The following procedure explain how you can validate connectivity to MSK clusters. For details about setting up aneifjcbevlkrdcl Amazon MSK client, see Getting started using Amazon MSK in the *Amazon Managed Streaming for Apache Kafka Developer Guide*.

To validate connectivity to MSK clusters

1. Create a Unix-based (preferably AL2) Amazon EC2 instance. If you have only VPC connectivity enabled on your cluster then make sure your EC2 instance runs in the same VPC. SSH into the instance once its available. For more information, see this tutorial in the *Amazon EC2 User Guide for Linux Instances*.
2. Install Java using the Yum package manager by running the following command. For more information, see the installation instructions in the Amazon Corretto 8 User Guide.

```
sudo yum install java-1.8.0
```

3. Install the Amazon client by running the following command.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

4. Download the Apache Kafka client 2.6* version by running the following command.

```
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. Go to the `kafka_2.12-2.6.2/libs` directory, then run the following command to download the Amazon MSK IAM JAR file.

```
wget https://github.com/aws/aws-msk-iam-auth/releases/download/v1.1.3/aws-msk-iam-auth-1.1.3-all.jar
```

6. Create `client.properties` file in Kafka bin folder.

7. Replace `awsRoleArn` with the role ARN that you have used in your Firehose `SourceConfiguration` and verify the cert location. Allow your Amazon client user to assume role `awsRoleArn`. Amazon client user will attempt to assume the role that you specified here.

```
[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. Run the following Kafka command to list topics. If your connection is public, use the public endpoint Bootstrap servers. If your connection is private, use the private endpoint Bootstrap servers.

```
bin/kafka-topics.sh --list --bootstrap-server <bootstrap servers> --command-config
 bin/client.properties
```

If the request is successful, you should see an output similar to the following example.

```
[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. If you have any issues running the previous script, verify that the bootstrap servers you provided are reachable on the specified port. To do this, you could download and use **telnet** or a similar utility as shown in the following command.

   ```
   sudo yum install telnet
   telnet <bootstrap servers><port>
   ```

   If the request is successful, you will get the following output. This means that you're able to connect to your MSK cluster within your local VPC and bootstrap servers are healthy on the specified port.

   ```
   Connected to ..
   ```

10. If the request is unsuccessful, check inbound rules on your VPC security group. As an example, you could use the following properties on the inbound rule.

    ```
    Type: All traffic
    Port: Port used by the bootstrap server (e.g. 14001)
    Source: 0.0.0.0/0
    ```

    Retry the **telnet** connection as shown in the previous step. If you're still unable to connect or your Firehose connection is still failing, contact the Amazon support.

Other

Topics

- Delivery Stream Not Available as a Target for CloudWatch Logs, CloudWatch Events, or Amazon IoT Action
- Data Freshness Metric Increasing or Not Emitted
- Record Format Conversion to Apache Parquet Fails

- No Data at Destination Despite Good Metrics

Delivery Stream Not Available as a Target for CloudWatch Logs, CloudWatch Events, or Amazon IoT Action

Some Amazon services can only send messages and events to a Firehose delivery stream that is in the same Amazon Region. Verify that your Firehose delivery stream is located in the same Region as your other services.

Data Freshness Metric Increasing or Not Emitted

Data freshness is a measure of how current your data is within your delivery stream. It is the age of the oldest data record in the delivery stream, measured from the time that Firehose ingested the data to the present time. Firehose provides metrics that you can use to monitor data freshness. To identify the data-freshness metric for a given destination, see the section called “Monitoring with CloudWatch Metrics”.

If you enable backup for all events or all documents, monitor two separate data-freshness metrics: one for the main destination and one for the backup.

If the data-freshness metric isn't being emitted, this means that there is no active delivery for the delivery stream. This happens when data delivery is completely blocked or when there's no incoming data.

If the data-freshness metric is constantly increasing, this means that data delivery is falling behind. This can happen for one of the following reasons.

- The destination can't handle the rate of delivery. If Firehose encounters transient errors due to high traffic, then the delivery might fall behind. This can happen for destinations other than Amazon S3 (it can happen for OpenSearch Service, Amazon Redshift, or Splunk). Ensure that your destination has enough capacity to handle the incoming traffic.
- The destination is slow. Data delivery might fall behind if Firehose encounters high latency. Monitor the destination's latency metric.
- The Lambda function is slow. This might lead to a data delivery rate that is less than the data ingestion rate for the delivery stream. If possible, improve the efficiency of the Lambda function. For instance, if the function does network IO, use multiple threads or asynchronous IO to increase parallelism. Also, consider increasing the memory size of the Lambda function so that

the CPU allocation can increase accordingly. This might lead to faster Lambda invocations. For information about configuring Lambda functions, see Configuring Amazon Lambda Functions.

- There are failures during data delivery. For information about how to monitor errors using Amazon CloudWatch Logs, see the section called "Monitoring with CloudWatch Logs".
- If the data source of the delivery stream is a Kinesis data stream, throttling might be happening. Check the `ThrottledGetRecords`, `ThrottledGetShardIterator`, and `ThrottledDescribeStream` metrics. If there are multiple consumers attached to the Kinesis data stream, consider the following:
  - If the `ThrottledGetRecords` and `ThrottledGetShardIterator` metrics are high, we recommend you increase the number of shards provisioned for the data stream.
  - If the `ThrottledDescribeStream` is high, we recommend you add the `kinesis:listshards` permission to the role configured in KinesisStreamSourceConfiguration.
- Low buffering hints for the destination. This might increase the number of round trips that Firehose needs to make to the destination, which might cause delivery to fall behind. Consider increasing the value of the buffering hints. For more information, see BufferingHints.
- A high retry duration might cause delivery to fall behind when the errors are frequent. Consider reducing the retry duration. Also, monitor the errors and try to reduce them. For information about how to monitor errors using Amazon CloudWatch Logs, see the section called "Monitoring with CloudWatch Logs".
- If the destination is Splunk and `DeliveryToSplunk.DataFreshness` is high but `DeliveryToSplunk.Success` looks good, the Splunk cluster might be busy. Free the Splunk cluster if possible. Alternatively, contact Amazon Support and request an increase in the number of channels that Firehose is using to communicate with the Splunk cluster.

Record Format Conversion to Apache Parquet Fails

This happens if you take DynamoDB data that includes the `Set` type, stream it through Lambda to a delivery stream, and use an Amazon Glue Data Catalog to convert the record format to Apache Parquet.

When the Amazon Glue crawler indexes the DynamoDB set data types (`StringSet`, `NumberSet`, and `BinarySet`), it stores them in the data catalog as `SET<STRING>`, `SET<BIGINT>`, and `SET<BINARY>`, respectively. However, for Firehose to convert the data records to the Apache Parquet format, it requires Apache Hive data types. Because the set types aren't valid Apache Hive

data types, conversion fails. To get conversion to work, update the data catalog with Apache Hive data types. You can do that by changing `set` to `array` in the data catalog.

To change one or more data types from `set` to `array` in an Amazon Glue data catalog

1. Sign in to the Amazon Web Services Management Console and open the Amazon Glue console at https://console.amazonaws.cn/glue/.
2. In the left pane, under the **Data catalog** heading, choose **Tables**.
3. In the list of tables, choose the name of the table where you need to modify one or more data types. This takes you to the details page for the table.
4. Choose the **Edit schema** button in the top right corner of the details page.
5. In the **Data type** column choose the first `set` data type.
6. In the **Column type** drop-down list, change the type from `set` to `array`.
7. In the **ArraySchema** field, enter `array<string>`, `array<int>`, or `array<binary>`, depending on the appropriate type of data for your scenario.
8. Choose **Update**.
9. Repeat the previous steps to convert other `set` types to `array` types.
10. Choose **Save**.

No Data at Destination Despite Good Metrics

If there are no data ingestion problems and the metrics emitted for the delivery stream look good, but you don't see the data at the destination, check the reader logic. Make sure your reader is correctly parsing out all data.

Amazon Data Firehose Quota

Amazon Data Firehose has the following quota.

- With Amazon MSK as the source for the Firehose stream, each Firehose stream has a default quota of 10 MB/sec of read throughput per partition and 10MB max record size. You can use the Service quota increase to request an increase on the default quota of 10 MB/sec of read throughput per partition.
- With Amazon MSK as the source for the Firehose stream, there is a 6Mb maximum record size if Amazon Lambda is enabled, and 10Mb maximum record size if Lambda is disabled. Amazon Lambda caps its incoming record to 6 MB, and Amazon Data Firehose forwards records above 6Mb to an error S3 bucket. If Lambda is disabled, Firehose cap its incoming record to 10 MB. If Amazon Data Firehose receives a record size from Amazon MSK that is larger than 10MB, then Amazon Data Firehose delivers this record to S3 error bucket and emits Cloudwatch metrics to your account. For more information on Amazon Lambda limits, see: https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html.
- When dynamic partitioning on a delivery stream is enabled, there is a default quota of 500 active partitions that can be created for that delivery stream. The active partition count is the total number of active partitions within the delivery buffer. For example, if the dynamic partitioning query constructs 3 partitions per second and you have a buffer hint configuration that triggers delivery every 60 seconds, then, on average, you would have 180 active partitions. Once data is delivered in a partition, then this partition is no longer active. You can use the Amazon Data Firehose Limits form to request an increase of this quota up to 5000 active partitions per given delivery stream. If you need more partitions, you can create more delivery streams and distribute the active partitions across them.
- When dynamic partitioning on a delivery stream is enabled, a max throughput of 1 GB per second is supported for each active partition.
- Each account will have following quota for the number of Firehose delivery streams per Region:
  - US East (N. Virginia), US East (Ohio), US West (Oregon), Europe (Ireland), Asia Pacific (Tokyo): 5,000 delivery streams
  - Europe (Frankfurt), Europe (London), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Seoul), Asia Pacific (Mumbai), Amazon GovCloud (US-West), Canada (West), Canada (Central): 2,000 delivery streams

- Europe (Paris), Europe (Milan), Europe (Stockholm), Asia Pacific (Hong Kong), Asia Pacific (Osaka), South America (Sao Paulo), China (Ningxia), China (Beijing), Middle East (Bahrain), Amazon GovCloud (US-East), Africa (Cape Town): 500 delivery streams
- Europe (Zurich), Europe (Spain), Asia Pacific (Hyderabad), Asia Pacific (Jakarta), Asia Pacific (Melbourne), Middle East (UAE), Israel (Tel Aviv), Canada West (Calgary), Canada (Central): 100 delivery streams
- If you exceed this number, a call to CreateDeliveryStream results in a `LimitExceededException` exception. To increase this quota, you can use Service Quotas if it's available in your Region. For information about using Service Quotas, see Requesting a Quota Increase. If Service Quotas aren't available in your Region, you can use the Amazon Data Firehose Limits form to request an increase.

- When **Direct PUT** is configured as the data source, each Firehose stream provides the following combined quota for PutRecord and PutRecordBatch requests:
  - For US East (N. Virginia), US West (Oregon), and Europe (Ireland): 500,000 records/second, 2,000 requests/second, and 5 MiB/second.
  - For US East (Ohio), US West (N. California), Amazon GovCloud (US-East), Amazon GovCloud (US-West), Asia Pacific (Hong Kong), Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Singapore), China (Beijing), China (Ningxia), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), Canada West (Calgary), Europe (Frankfurt), Europe (London), Europe (Paris), Europe (Stockholm), Middle East (Bahrain), South America (São Paulo), Africa (Cape Town), and Europe (Milan): 100,000 records/second, 1,000 requests/second, and 1 MiB/second.

  To request an increase in quota, use the Amazon Data Firehose Limits form. The three quota scale proportionally. For example, if you increase the throughput quota in US East (N. Virginia), US West (Oregon), or Europe (Ireland) to 10 MiB/second, the other two quota increase to 4,000 requests/second and 1,000,000 records/second.

  > **Important**
  >
  > If the increased quota is much higher than the running traffic, it causes small delivery batches to destinations. This is inefficient and can result in higher costs at the destination services. Be sure to increase the quota only to match current running traffic, and increase the quota further if traffic increases.

Important

Note that smaller data records can lead to higher costs. Firehose ingestion pricing is based on the number of data records you send to the service, times the size of each record rounded up to the nearest 5KB (5120 bytes). So, for the same volume of incoming data (bytes), if there is a greater number of incoming records, the cost incurred would be higher. For example, if the total incoming data volume is 5MiB, sending 5MiB of data over 5,000 records costs more compared to sending the same amount of data using 1,000 records. For more information, see Amazon Data Firehose in the Amazon Calculator.

Note

When Kinesis Data Streams is configured as the data source, this quota doesn't apply, and Amazon Data Firehose scales up and down with no limit.

- Each Firehose stream stores data records for up to 24 hours in case the delivery destination is unavailable and if the source is DirectPut. If the source is Kinesis Data Streams (KDS) and the destination is unavailable, then the data will be retained based on your KDS configuration.
- The maximum size of a record sent to Amazon Data Firehose, before base64-encoding, is 1,000 KiB.
- The PutRecordBatch operation can take up to 500 records per call or 4 MiB per call, whichever is smaller. This quota cannot be changed.
- The following operations can provide up to five invocations per second (this is a hard limit): `CreateDeliveryStream`, `DeleteDeliveryStream`, `DescribeDeliveryStream`, `ListDeliveryStreams`, `UpdateDestination`, `TagDeliveryStream`, `UntagDeliveryStream`, `ListTagsForDeliveryStream`, `StartDeliveryStreamEncryption`, `StopDeliveryStreamEncryption`.
- The buffer interval hints range from 60 seconds to 900 seconds.
- For delivery from Amazon Data Firehose to Amazon Redshift, only publicly accessible Amazon Redshift clusters are supported.
- The retry duration range is from 0 seconds to 7,200 seconds for Amazon Redshift and OpenSearch Service delivery.

- Firehose supports Elasticsearch versions 1.5, 2.3, 5.1, 5.3, 5.5, 5.6, as well as all 6.* and 7.* versions and Amazon OpenSearch Service 2.x up to 2.11.
- When the destination is Amazon S3, Amazon Redshift, or OpenSearch Service, Amazon Data Firehose allows up to 5 outstanding Lambda invocations per shard. For Splunk, the quota is 10 outstanding Lambda invocations per shard.
- You can use a CMK of type CUSTOMER_MANAGED_CMK to encrypt up to 500 delivery streams.

Appendix - HTTP Endpoint Delivery Request and Response Specifications

For Amazon Data Firehose to successfully deliver data to custom HTTP endpoints, these endpoints must accept requests and send responses using certain Amazon Data Firehose request and response formats. This section describes the format specifications of the HTTP requests that the Amazon Data Firehose service sends to custom HTTP endpoints, as well as the format specifications of the HTTP responses that the Amazon Data Firehose service expects. HTTP endpoints have 3 minutes to respond to a request before Amazon Data Firehose times out that request. Amazon Data Firehose treats responses that do not adhere to the proper format as delivery failures.

Topics

Request Format

Path and URL Parameters

These are configured directly by you as part of a single URL field. Amazon Data Firehose sends them as configured without modification. Only https destinations are supported. URL restrictions are applied during delivery-stream configuration.

> **Note**
>
> Currently, only port 443 is supported for HTTP endpoint data delivery.

HTTP Headers - X-Amz-Firehose-Protocol-Version

This header is used to indicate the version of the request/response formats. Currently the only version is 1.0.

HTTP Headers - X-Amz-Firehose-Request-Id

The value of this header is an opaque GUID that can be used for debugging and deduplication purposes. Endpoint implementations should log the value of this header if possible, for both successful and unsuccessful requests. The request ID is kept the same between multiple attempts of the same request.

HTTP Headers - Content-Type

The value of the Content-Type header is always `application/json`.

HTTP Headers - Content-Encoding

A Firehose stream can be configured to use GZIP to compress the body when sending requests. When this compression is enabled, the value of the Content-Encoding header is set to gzip, as per standard practice. If compression is not enabled, the Content-Encoding header is absent altogether.

HTTP Headers - Content-Length

This is used in the standard way.

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

The ARN of the Firehose stream represented in ASCII string format. The ARN encodes region, Amazon account ID and the stream name. For example, `arn:aws:firehose:us-east-1:123456789:deliverystream/testStream`.

HTTP Headers - X-Amz-Firehose-Access-Key

This header carries an API key or other credentials. You have the ability to create or update the API-key (aka authorization token) when creating or updating your delivery-stream. Amazon Data Firehose restricts the size of the access key to 4096 bytes. Amazon Data Firehose does not attempt to interpret this key in any way. The configured key is copied verbatim into the value of this header.

The contents can be arbitrary and can potentially represent a JWT token or an ACCESS_KEY. If an endpoint requires multi-field credentials (for example, username and password), the values of all of the fields should be stored together within a single access-key in a format that the endpoint understands (JSON or CSV). This field can be base-64 encoded if the original contents are binary. Amazon Data Firehose does not modifiy and/or encode the configured value and uses the contents as is.

HTTP Headers - X-Amz-Firehose-Common-Attributes

This header carries the common attributes (metadata) that pertain to the entire request, and/or to all records within the request. These are configured directly by you when creating a Firehose stream. The value of this attribute is encoded as a JSON object with the following schema:

```
"$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
```

Here's an example:

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

Body - Max Size

The maximum body size is configured by you, and can be up to a maximum of 64 MiB, before compression.

Body - Schema

The body carries a single JSON document with the following JSON Schema (written in YAML):

```
"$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
```

Here's an example:

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

Response Format

Default Behavior on Error

If a response fails to conform to the requirements below, the Firehose server treats it as though it had a 500 status code with no body.

Status Code

The HTTP status code MUST be in the 2XX, 4XX or 5XX range.

The Amazon Data Firehose server does NOT follow redirects (3XX status codes). Only response code 200 is considered as a successful delivery of the records to HTTP/EP. Response code 413 (size exceeded) is considered as a permanent failure and the record batch is not sent to error bucket if configured. All other response codes are considered as retriable errors and are subjected to back-off retry algorithm explained later.

Headers - Content Type

The only acceptable content type is application/json.

HTTP Headers - Content-Encoding

Content-Encoding MUST NOT be used. The body MUST be uncompressed.

HTTP Headers - Content-Length

The Content-Length header MUST be present if the response has a body.

Body - Max Size

The response body must be 1 MiB or less in size.

```
"$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
```

Here's an example:

```
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
}
```

Error Response Handling

In all error cases the Amazon Data Firehose server reattempts delivery of the same batch of records using an exponential back-off algorithm. The retries are backed off using an initial back-off time (1 second) with a jitter factor of (15%) and each subsequent retry is backed off using the formula (initial-backoff-time * (multiplier(2) ^ retry_count)) with added jitter. The backoff time is capped by a maximum interval of 2 minutes. For example on the 'n'-th retry the back off time is = MAX(120sec, (1 * (2^n)) * random(0.85, 1,15).

The parameters specified in the previous equation are subject to change. Refer to the Amazon Firehose documentation for exact initial back off time, max backoff time, multiplier and jitter percentages used in exponential back off algorithm.

In each subsequent retry attempt the access key and/or destination to which records are delivered might change based on updated configuration of the Firehose stream. Amazon Data Firehose service uses the same request-id across retries in a best-effort manner. This last feature can be used for deduplication purpose by the HTTP end point server. If the request is still not delivered after the maximum time allowed (based on Firehose stream configuration) the batch of records can optionally be delivered to an error bucket based on stream configuration.

Examples

Example of a CWLog sourced request:

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

Document History

The following table describes the important changes to the Amazon Data Firehose documentation.

Change	Description	Date Changed
Amazon Kinesis Data Firehose is now known as Amazon Data Firehose	Amazon Kinesis Data Firehose has rebranded to Amazon Data Firehose. See *What Is Amazon Data Firehose?*	February 9, 2024
Added Snowflake as a destination	You can create a delivery stream with Snowflake as the destination. See the section called "Choose Snowflake for Your Destination".	January 19, 2024
Added automatic decompression of CloudWatch Logs	You can enable decompression on new or existing streams to send decompressed CloudWatch Logs data to Firehose destinations. See the section called "Writing Using CloudWatch Logs".	December 15, 2023
Added Splunk Observability Cloud as a destination	You can create a Firehose stream with Splunk Observability Cloud as the destination. See the section called "Choose Splunk Observability Cloud for Your Destination".	October 3, 2023
Added Amazon Managed Streaming for Apache Kafka as a data source	You can now configure Amazon MSK to send information to a Firehose stream. See the section called "Writing Using Amazon MSK".	September 26th, 2023
Added support for DocumentID type for the OpenSearch Service destination	If OpenSearch Service is your Firehose stream's destination, DocumentID type indicates the method for setting up document ID. The supported methods are Firehose generated document ID and OpenSearc	May 10th, 2023

Change	Description	Date Changed
	h Service generated document ID. See the section called “Destination Settings”.	
Added support dynamic partition ing	Added support for continuous dynamic partitioning of the streaming data in Amazon Data Firehose. See *Dynamic Partitioning*.	August 31, 2021
Added a topic on custom prefixes.	Added a topic about the expressions that you can use when building a custom prefix for data that is delivered to Amazon S3. See *Custom Amazon S3 Prefixes*.	December 20, 2018
Added New Amazon Data Firehose Tutorial	Added a tutorial that demonstrates how to send Amazon VPC flow logs to Splunk through Amazon Data Firehose. See Tutorial: Ingest VPC flow logs into Splunk using Amazon Data Firehose.	October 30, 2018
Added Four New Amazon Data Firehose Regions	Added Paris, Mumbai, Sao Paulo, and London. For more information, see Amazon Data Firehose Quota.	June 27, 2018
Added Two New Amazon Data Firehose Regions	Added Seoul and Montreal. For more information, see Amazon Data Firehose Quota.	June 13, 2018
New Kinesis Streams as Source feature	Added Kinesis Streams as a potential source for records for a Data Firehose Firehose stream. For more information, see Source, Destination, and Name.	August 18, 2017
Update to console documentation	The Firehose stream creation wizard was updated. For more information, see Creating a Firehose stream.	July 19, 2017

Change	Description	Date Changed
New data transformation	You can configure Amazon Data Firehose to transform your data before data delivery. For more information, see Amazon Data Firehose Data Transformation.	December 19, 2016
New Amazon Redshift COPY retry	You can configure Amazon Data Firehose to retry a COPY command to your Amazon Redshift cluster if it fails. For more information, see Creating a Firehose stream, Amazon Data Firehose Data Delivery, and Amazon Data Firehose Quota.	May 18, 2016
New Amazon Data Firehose destination, Amazon OpenSearch Service	You can create a Firehose stream with Amazon OpenSearch Service as the destination. For more information, see Creating a Firehose stream, Amazon Data Firehose Data Delivery, and Grant Amazon Data Firehose Access to a Public OpenSearch Service Destination.	April 19, 2016
New enhanced CloudWatch metrics and troubleshooting features	Updated Monitoring Amazon Data Firehose and Troubleshooting Amazon Data Firehose.	April 19, 2016
New enhanced Kinesis agent	Updated Writing to Amazon Data Firehose Using Kinesis Agent.	April 11, 2016
New Kinesis agents	Added Writing to Amazon Data Firehose Using Kinesis Agent.	October 2, 2015
Initial release	Initial release of the Amazon Data Firehose *Developer Guide*.	October 4, 2015

Amazon Glossary

For the latest Amazon terminology, see the Amazon glossary in the *Amazon Web Services Glossary Reference*.