Amazon Kinesis Firehose

Firehose

Amazon Kinesis Firehose: Firehose

Table of Contents

What is Amazon Kinesis Firehose?

Amazon Kinesis Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, or Amazon Elasticsearch Service (Amazon ES). Firehose is part of the Amazon Kinesis streaming data platform, along with Amazon Kinesis Streams and Amazon Kinesis Analytics. With Firehose, you do not need to write applications or manage resources. You configure your data producers to send data to Firehose and it automatically delivers the data to the destination that you specified. You can also configure Firehose to transform your data before data delivery.

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

Key Concepts

As you get started with Firehose, you'll benefit from understanding the following concepts:

Firehose delivery stream
The underlying entity of Firehose; you use Firehose by creating a Firehose delivery stream and then sending data to it. For more information, see Creating an Amazon Kinesis Firehose Delivery Stream (p. 6) and Sending Data to an Amazon Kinesis Firehose Delivery Stream (p. 16).

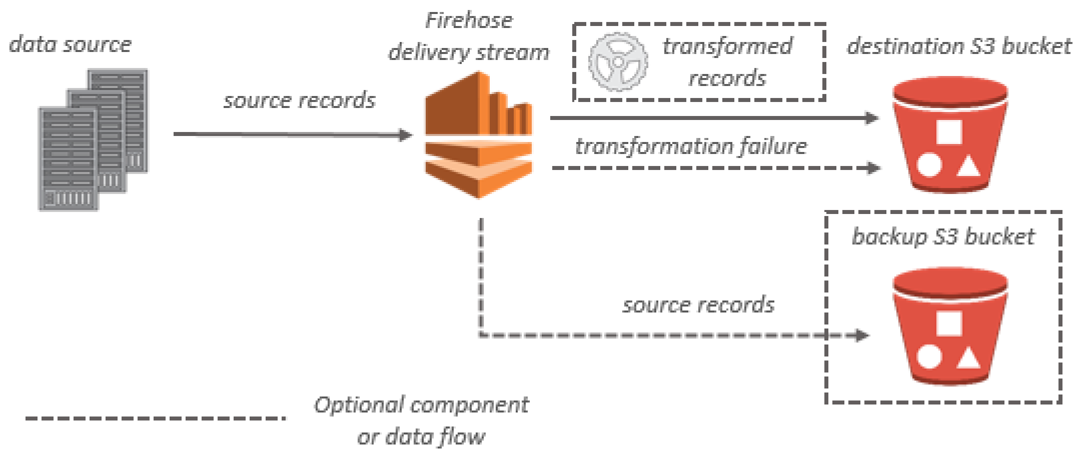
record
The data of interest that your data producer sends to a Firehose delivery stream. A record can be as large as 1000 KB.

data producer
Producers send records to Firehose delivery streams. For example, a web server sending log data to a Firehose delivery stream is a data producer. For more information, see Sending Data to an Amazon Kinesis Firehose Delivery Stream (p. 16).

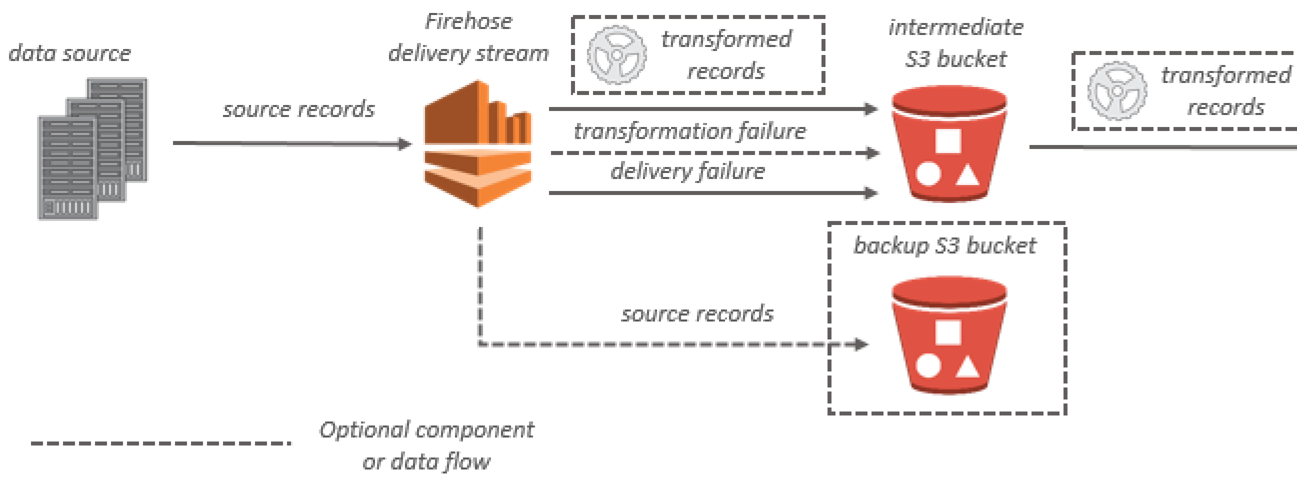
buffer size and buffer interval
Firehose buffers incoming streaming data to a certain size or for a certain period of time before delivering to destinations. **Buffer Size** is in MBs and **Buffer Interval** is in seconds.

Data Flow

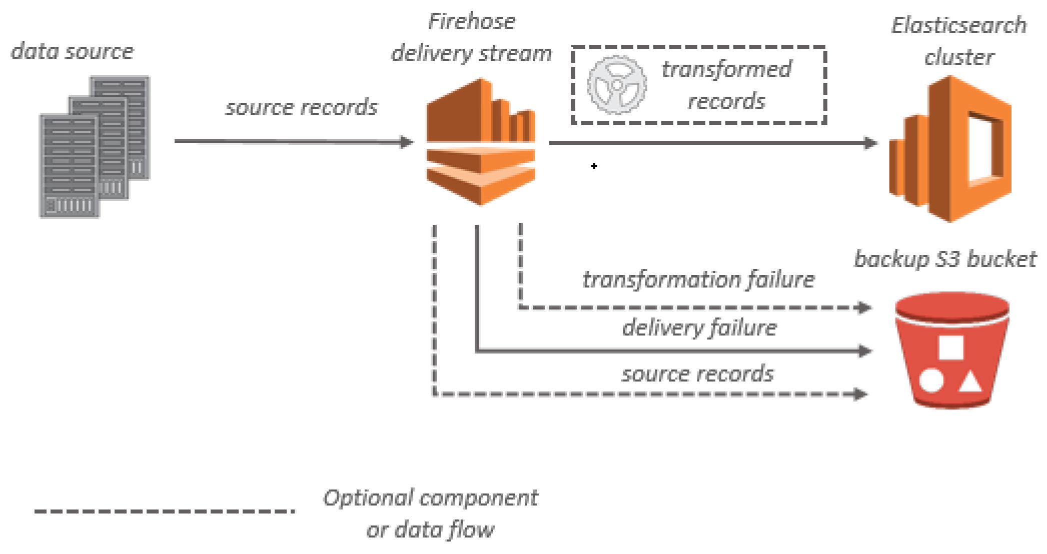
For Amazon S3 destinations, streaming data is delivered to your S3 bucket. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket.



For Amazon Redshift destinations, streaming data is delivered to your S3 bucket first. 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 Amazon ES destinations, streaming data is delivered to your Amazon ES cluster, and can optionally be backed up to your S3 bucket concurrently.


data source
source records
Firehose delivery stream
transformed records
Elasticsearch cluster
backup S3 bucket
transformation failure
delivery failure
source records
Optional component or data flow

Setting Up for Amazon Kinesis Firehose

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

Tasks

Sign Up for AWS

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

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

To sign up for an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Optional: Download Libraries and Tools

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

- The Amazon Kinesis Firehose API Reference is the basic set of operations that Firehose supports.
- The AWS SDKs for Java, .NET, Node.js, Python, and Ruby include Firehose support and samples.

  If your version of the AWS SDK for Java does not include samples for Firehose, you can also download the latest AWS SDK from GitHub.

- The AWS Command Line Interface supports Firehose. The AWS CLI enables you to control multiple AWS services from the command line and automate them through scripts.

Creating an Amazon Kinesis Firehose Delivery Stream

You can use the AWS Management Console or an SDK to create a Firehose delivery stream to your chosen destination. After you create a Firehose delivery stream, you can update the destination.

Contents

Create a Firehose Delivery Stream to Amazon S3

The following procedure shows how to use the console to create a Firehose delivery stream with Amazon S3 as the destination.

To create a delivery stream

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.
2. Choose **Create Firehose Delivery Stream**.
3. Choose Amazon S3 as the destination.
4. Specify the Firehose delivery stream and destination information as follows.

   Delivery stream name
   The name of your Firehose delivery stream.

   S3 bucket
   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.

   S3 prefix
   Firehose automatically uses a prefix in "YYYY/MM/DD/HH" UTC time format for delivered S3 objects. You can specify an extra prefix to be added in front of the time format prefix. Note that if the prefix ends with a forward slash (`/`), it appears as a folder in the S3 bucket. This value is optional.
5. Choose **Next** to change the following **Configuration** settings:

Data transformation with Lambda
Firehose can invoke your Lambda function to transform incoming data before delivering it. 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 required by Firehose. For more information, see Amazon Kinesis Firehose Data Transformation (p. 27).

Source record backup
If you enable data transformation with Lambda, you can enable source record backup to deliver untransformed incoming data to a separate S3 bucket. You can specify an extra prefix to be added in front of the "YYYY/MM/DD/HH" UTC time prefix generated by Firehose. Note that you cannot disable source record backup after you enable it.

S3, Buffer size, Buffer interval
Firehose buffers incoming data before delivering it to Amazon S3. You can choose a buffer size (1-128 MBs) or buffer interval (60-900 seconds); whichever condition 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 Firehose to the data delivery to Amazon S3. Note that in circumstances where data delivery to the destination falls behind data writing to the delivery stream, Firehose raises the buffer size dynamically to catch up and ensure that all data is delivered to the destination.

S3, Data compression
You can choose from three compression formats (GZIP, ZIP, or SNAPPY), or no data compression.

S3, Data encryption
Firehose supports Amazon S3 server-side encryption with AWS Key Management Service (AWS KMS) for encrypting delivered data in Amazon S3. You can choose not to encrypt the data or to encrypt with a key from the list of AWS KMS keys that you own. For more information, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS).

Error logging
Firehose can log Lambda invocation, if data transformation is enabled, and data delivery errors to CloudWatch Logs so that you can view the specific error logs is Lambda invocation or data delivery fails. For more information, see Monitoring with Amazon CloudWatch Logs (p. 36).

IAM role
This IAM role is used to grant Firehose access to your S3 bucket, AWS KMS key (if data encryption is enabled), and Lambda function (if data transformation is enabled). You can choose to create a new IAM role where required permissions are assigned automatically, or choose an existing role created for Firehose. For more information, see Grant Firehose Access to an Amazon S3 Destination (p. 48).

The console might create a role with placeholders. You can safely ignore or safely delete lines with %FIREHOSE_BUCKET_NAME%, %FIREHOSE_DEFAULT_FUNCTION%, or %FIREHOSE_DEFAULT_VERSION%.

6. When settings are final, choose **Next** to review and then choose **Create Delivery Stream**.

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

Create a Firehose Delivery Stream to Amazon Redshift

The following procedure shows the steps for using the AWS console to create a Firehose delivery stream with Amazon Redshift as the destination.

To create a delivery stream

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.
2. Choose **Create Firehose Delivery Stream**.
3. Choose Amazon Redshift for the destination.
4. Specify the Firehose delivery stream and destination information.

    Delivery stream name
    The name of your Firehose delivery stream.

    S3 bucket
    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. You have the option of creating a new S3 bucket or choosing an existing bucket that you own.

    Note that 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 Developer Guide*.

    S3 prefix
    Firehose automatically uses a prefix in "YYYY/MM/DD/HH" UTC time format for delivered S3 objects. You can specify an extra prefix to be added in front of the time format prefix. Note that if the prefix ends with a forward slash (`/`), it appears as a folder in the S3 bucket. This value is optional.

    Redshift cluster
    Your Amazon Redshift cluster to which your S3 bucket data will be copied. Note that you need to configure your Amazon Redshift cluster to be publicly accessible and unblock Firehose IP addresses. For more information, see Grant Firehose Access to an Amazon Redshift Destination (p. 50).

    Redshift table
    The Amazon Redshift table where the data will be copied.

    Redshift table columns
    The specific columns of the table to which the data will be copied. Use this option if the number of columns defined in your S3 objects is less than the number of columns within the Amazon Redshift table. This value is optional.

    Redshift username
    Amazon Redshift user with permission to access your Amazon Redshift cluster. This user needs to have Amazon Redshift INSERT privilege for copying data from S3 bucket to the Amazon Redshift cluster.

    Redshift password
    The password for the user above.

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

Retry duration (sec)
Time duration (0-7200 seconds) for Firehose to retry if data **COPY** to your Amazon Redshift cluster fails. If 0 (zero) seconds is configured, Firehose will not retry upon **COPY** command failure.

5. Choose **Next** to change the following **Configuration** settings:

Data transformation with Lambda
Firehose can invoke your Lambda function to transform incoming data before delivering it. 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 required by Firehose. For more information, see Amazon Kinesis Firehose Data Transformation (p. 27).

Source record backup
If you enable data transformation with Lambda, you can enable source record backup to deliver untransformed incoming data to a separate S3 bucket. You can specify an extra prefix to be added in front of the "YYYY/MM/DD/HH" UTC time prefix generated by Firehose. Note that you cannot disable source record backup after you enable it.

S3, Buffer size, Buffer interval
Firehose buffers incoming data before delivering it to Amazon S3. You can choose a buffer size (1-128 MBs) or buffer interval (60-900 seconds); whichever condition 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 Firehose to the data delivery to Amazon S3. Note that in circumstances where data delivery to the destination falls behind data writing to the delivery stream, Firehose raises the buffer size dynamically to catch up and ensure that all data is delivered to the destination.

S3, Data compression
Firehose can compress incoming data before delivering it to Amazon S3. You can choose GZIP compression format or no data compression. Note that if you specify compression you will need to make sure GZIP is in the Amazon Redshift **COPY** parameter from the previous step.

S3, Data encryption
Firehose supports Amazon S3 server-side encryption with AWS Key Management Service (AWS KMS) for encrypting delivered data in Amazon S3. You can choose not to encrypt the data or to encrypt with a key from the list of AWS KMS keys that you own. For more information about Amazon S3 server-side encryption with KMS keys, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS).

Error logging
Firehose can log Lambda invocations, if data transformation is enabled, and send data delivery errors to CloudWatch Logs so that you can view the specific error logs if Lambda invocation or data delivery fails. For more information, see Monitoring with Amazon CloudWatch Logs (p. 36).

IAM role
This IAM role is used to grant Firehose access to your S3 bucket, AWS KMS key (if data encryption is enabled), and Lambda function (if data transformation is enabled.) You can choose to create a new IAM role where required permissions are assigned automatically, or choose an existing role created for Firehose. For more information, see Grant Firehose Access to an Amazon Redshift Destination (p. 50).

The console might create a role with placeholders. You can safely ignore or safely delete lines with %FIREHOSE_BUCKET_NAME%, %FIREHOSE_DEFAULT_FUNCTION%, or %FIREHOSE_DEFAULT_VERSION%.

6. When settings are final, choose **Next** to review and then choose **Create Delivery Stream**.

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

Create a Firehose Delivery Stream to Amazon Elasticsearch Service

The following procedure shows the steps for using the AWS console to create a Firehose delivery stream with Amazon ES as the destination.

To create a delivery stream

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.
2. Choose **Create Firehose Delivery Stream**.
3. Choose Amazon ES for the destination.
4. Specify the Firehose delivery stream and destination information.

   Delivery stream name
   The name of your Firehose delivery stream.

   Elasticsearch domain
   The Amazon ES domain to which your data will be delivered.

   Index
   The Elasticsearch index name to be used when indexing data to your Amazon ES cluster.

   Index rotation
   Select whether and how often the Elasticsearch index should be rotated. If index rotation is enabled, Firehose appends the corresponding timestamp to the specified index name and rotates. For more information, see Index Rotation for the Amazon ES Destination (p. 32).

   Type
   The Amazon ES type name to be used when indexing data to your Amazon ES cluster.

   Retry duration (sec)
   Time duration (0-7200 seconds) for Firehose to retry if an index request to your Amazon ES cluster fails. If 0 (zero) seconds is configured, Firehose will not retry upon index request failure.
5. Specify backup S3 bucket information.

   Firehose can back up incoming data to Amazon S3 while delivering it to Amazon ES.

   Backup mode
   You can choose to either back up failed documents only or all documents. If you choose failed documents only, any data that Firehose could not deliver to your Amazon ES cluster or your Lambda function could not transform are backed up to the specified S3 bucket. If you choose all documents, Firehose backs up all incoming source data to your S3 bucket concurrently with data delivery to Amazon ES. For more information, see Data Delivery Failure Handling (p. 31) and Data Transformation Failure Handling (p. 28).

   S3 bucket
   An S3 bucket you own that is the target of the backup data. You have the option of creating a new S3 bucket or choosing an existing bucket that you own.

   S3 prefix
   Firehose automatically uses a prefix in "YYYY/MM/DD/HH" UTC time format for delivered S3 objects. You can specify an extra prefix to be added in front of the time format prefix. Note that if the prefix ends with a forward slash (`/`), it appears as a folder in the S3 bucket. This value is optional.
6. Choose **Next** to change the following **Configuration** settings.

Data transformation with Lambda
Firehose can invoke your Lambda function to transform incoming data before delivering it. 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 required by Firehose. For more information, see Amazon Kinesis Firehose Data Transformation (p. 27).

Elasticsearch, Buffer size, Buffer interval
Firehose buffers incoming data before delivering it to Amazon ES. You can choose **Buffer size** (1-100 MBs) or **Buffer interval** (60-900 seconds), and whichever condition is satisfied first triggers the data delivery to Amazon ES. Please note that in circumstances where data delivery to the destination is falling behind data writing to the delivery stream, Firehose raises the buffer size dynamically to catch up and make sure that all data is delivered to the destination.

S3, Buffer size, Buffer interval
Firehose buffers incoming data before delivering it to Amazon S3. You can choose a buffer size (1-128 MBs) or a buffer interval (60-900 seconds); whichever condition is satisfied first triggers the data delivery to Amazon S3. Note that in circumstances where data delivery to the destination falls behind data writing to the delivery stream, Firehose raises the buffer size dynamically to catch up and make sure that all data is delivered to the destination.

S3, Data compression
Firehose can compress incoming data before delivering it to Amazon S3. You can choose from three compression formats (GZIP, ZIP, or SNAPPY), or no data compression.

S3, Data encryption
Firehose supports Amazon S3 server-side encryption with AWS Key Management Service (AWS KMS) for encrypting delivered data in Amazon S3. You can choose not to encrypt the data or to encrypt with a key from the list of AWS KMS keys that you own. For more information about Amazon S3 server-side encryption with AWS KMS keys, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS).

Error logging
Firehose can log Lambda invocation, if data transformation is enabled, and send data delivery errors to CloudWatch Logs so that you can view the specific error logs if Lambda invocation or data delivery fails. For more information, see Monitoring with Amazon CloudWatch Logs (p. 36).

IAM role
This IAM role is used to grant Firehose access to your specified Amazon ES domain, S3 bucket, and AWS KMS key (if data encryption is enabled). You can choose to create a new IAM role where required permissions are assigned automatically, or choose an existing role created for Firehose. For more information, see Grant Firehose Access to an Amazon Elasticsearch Service Destination (p. 52).

The console might create a role with placeholders. You can safely ignore or safely delete lines with %FIREHOSE_BUCKET_NAME%, %FIREHOSE_DEFAULT_FUNCTION%, or %FIREHOSE_DEFAULT_VERSION%.

7. When settings are final, choose **Next** to review and then choose **Create Delivery Stream**.

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

Update Destination

You can update the configurations of your Firehose delivery stream at any time after it’s created. You can do so by using the Firehose console or the UpdateDestination operation. Your Firehose delivery stream remains in the `ACTIVE` state while your configurations are updated and you can continue to send data to your Firehose delivery stream. The updated configurations normally take effect within a few minutes. The version number of a Firehose delivery stream is increased by a value of `1` after updating Firehose delivery stream configurations. The version number is reflected in the delivered S3 object name. For more information, see Amazon S3 Object Name Format (p. 32).

Testing Your Delivery Stream Using Sample Data

You can use the AWS Management Console to ingest simulated stock ticker data. The console runs a script in your browser to put sample records in your Firehose delivery stream. This enables you to test the configuration of your delivery 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 Kinesis Firehose charges apply when your delivery 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 delivery stream. For more information, see Creating an Amazon Kinesis Firehose Delivery Stream (p. 6).

Test Using Amazon S3 as the Destination

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

To test a delivery stream using Amazon S3

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.

2. Choose the delivery stream.
3. Under **Test with demo data**, choose **Start sending demo data** to generate sample stock ticker data.
4. Follow the on-screen 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 delivery stream using Amazon Redshift as the destination.

To test a delivery stream using Amazon Redshift

1. Your delivery stream expects a table to be present in your 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.aws.amazon.com/firehose/.
3. Choose the delivery stream.
4. Edit the destination details for your delivery 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 on-screen 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 delivery stream to point to another table.
9. (Optional) Delete the `firehose_test_table` table.

Test Using Amazon ES as the Destination

Use the following procedure to test your delivery stream using Amazon Elasticsearch Service (Amazon ES) as the destination.

To test a delivery stream using Amazon ES

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.
2. Choose the delivery stream.
3. Under **Test with demo data**, choose **Start sending demo data** to generate sample stock ticker data.

4. Follow the on-screen instructions to verify that data is being delivered to your Amazon ES domain. For more information, see Searching Documents in an Amazon ES Domain in the *Amazon Elasticsearch Service Developer Guide*.
5. When the test is complete, choose **Stop sending demo data** to stop incurring usage charges.

Sending Data to an Amazon Kinesis Firehose Delivery Stream

You can send data to your Firehose delivery stream using the Amazon Kinesis Agent or the Firehose API using the AWS SDK. If you are new to Firehose, take some time to become familiar with the concepts and terminology presented in What is Amazon Kinesis Firehose? (p. 1).

Topics

Writing to Amazon Kinesis Firehose Using Amazon Kinesis Agent

Amazon Kinesis Agent is a stand-alone Java software application that offers an easy way to 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 handles file rotation, checkpointing, and retry upon failures. It delivers all of your data in a reliable, timely, and simple manner. It also emits Amazon CloudWatch metrics to help you better monitor and troubleshoot the streaming process.

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 (p. 18)).

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 delivery stream for the data. After the agent is configured, it durably collects data from the files and reliably sends it to the delivery stream.

Topics

- Monitor Multiple File Directories and Write to Multiple Streams (p. 20)
- Use the Agent to Pre-process Data (p. 21)
- Agent CLI Commands (p. 24)

Prerequisites

- Your operating system must be either Amazon Linux AMI with version 2015.09 or later, or Red Hat Enterprise Linux version 7 or later.
- If you are using Amazon EC2 to run your agent, launch your EC2 instance.
- Manage your AWS credentials using one of the following methods:
  - Specify an IAM role when you launch your EC2 instance.
  - Specify AWS credentials when you configure the agent (see awsAccessKeyId (p.) and awsSecretAccessKey (p.)).
  - Edit `/etc/sysconfig/aws-kinesis-agent` to specify your region and AWS access keys.
  - If your EC2 instance is in a different AWS account, create an IAM role to provide access to the Firehose service, and specify that role when you configure the agent (see assumeRoleARN (p.) and assumeRoleExternalId (p.)). Use one of the previous methods to specify the AWS credentials of a user in the other account who has permission to assume this role.
- The IAM role or AWS credentials that you specify must have permission to perform the Firehose PutRecordBatch operation for the agent to send data to your delivery 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 Kinesis Firehose (p. 47), Monitoring Amazon Kinesis Agent Health (p. 41), and Authentication and Access Control for Amazon CloudWatch.

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 using the Amazon Linux AMI

Use the following command to download and install the agent:

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

To set up the agent using Red Hat Enterprise Linux

Use the following command to download and install the agent:

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

To set up the agent using GitHub

1. Download the agent from awlabs/amazon-kinesis-agent.
2. Install the agent by navigating to the download directory and running the following command:

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

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 delivery stream (`"deliveryStream"`) to which the agent sends data. Note that 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 timestamp to determine which files to track and tail into your delivery 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"
        }
   ]
}
```

   Note that the default 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 (p. 18).
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 delivery 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 ARN of the role to be assumed by the user. For more information, see Delegate Access Across AWS 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`	AWS access key ID that overrides the default credentials. This setting takes precedence over all other credential providers.
`awsSecretAccessKey`	AWS 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 Firehose. Default: `firehose.us-east-1.amazonaws.com`

The following are the flow configuration settings.

Configuration Setting	Description
`dataProcessingOptions`	The list of processing options applied to each parsed record before it is sent to the delivery stream. The processing options are performed in the specified order. For more information, see Use the Agent to Pre-process Data (p. 21).
`deliveryStream`	[Required] The name of the delivery 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, read permission must be granted to `aws-kinesis-agent-user`. For the directory containing the files, read and execute permissions must be granted to `aws-kinesis-agent-user`.
`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`
`maxBufferAgeMillis`	The maximum time, in milliseconds, for which the agent buffers data before sending it to the delivery stream.

Configuration Setting	Description
	Value range: 1,000 to 900,000 (1 second to 15 minutes) Default: 60,000 (1 minute)
`maxBufferSizeBytes`	The maximum size, in bytes, for which the agent buffers data before sending it to the delivery stream. Value range: 1 to 4,194,304 (4 MB) Default: 4,194,304 (4 MB)
`maxBufferSizeRecords`	The maximum number of records for which the agent buffers data before sending it to the delivery stream. Value range: 1 to 500 Default: 500
`minTimeBetweenFilePollsMillis`	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
`multiLineStartPattern`	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.
`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 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 an Amazon Kinesis stream and a Firehose delivery stream respectively. Note that you can specify different endpoints for Streams and Firehose so that your Amazon Kinesis stream and Firehose delivery 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 Streams, see Writing to Amazon Kinesis Streams with Amazon Kinesis Agent.

Use the Agent to Pre-process Data

The agent can pre-process the records parsed from monitored files before sending them to your delivery 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 will be 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 Amazon 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
```

Writing to a Firehose Delivery Stream Using the AWS SDK

You can use the Firehose API to send data to a Firehose delivery stream using the AWS SDK for Java, .NET, Node.js, Python, or Ruby. If you are new to Firehose, take some time to become familiar with the concepts and terminology presented in What is Amazon Kinesis Firehose? (p. 1). 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 Firehose API offers two operations for sending data to your Firehose delivery 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

Putting data requires only the Firehose delivery stream name and a byte buffer (<=1000 KB). Because 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 delivery stream, use the following code:

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

String data = line + "\n";

Record record = createRecord(data);
putRecordRequest.setRecord(record);

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

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

Batch Write Operations Using PutRecordBatch

Putting data requires only the Firehose delivery stream name and a list of records. Because 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 delivery 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 AWS SDK. For information about request and response syntax, see the relevant topic in Amazon Kinesis Firehose API Operations.

Amazon Kinesis Firehose Data Transformation

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

Data Transformation Flow

When you enable Firehose data transformation, Firehose buffers incoming data up to 3 MB or the buffering size you specified for the delivery stream, whichever is smaller. Firehose then invokes the specified Lambda function with each buffered batch asynchronously. The transformed data is sent from Lambda to Firehose for buffering. Transformed data is delivered to the destination when the specified buffering size or buffering interval is reached, whichever happens first.

Data Transformation and Status Model

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

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.

data
: The transformed data payload, after base64-encoding.

Lambda Blueprints

Firehose provides the following Lambda blueprints that you can use to create a Lambda function for data transformation.

- **General Firehose Processing** — Contains the data transformation and status model described in the previous section. Use this blueprint for any custom transformation logic.
- **Apache Log to JSON** — Parses and converts Apache log lines to JSON objects, using predefined JSON field names.
- **Apache Log to CSV** — Parses and converts Apache log lines to CSV format.
- **Syslog to JSON** — Parses and converts Syslog lines to JSON objects, using predefined JSON field names.
- **Syslog to CSV** — Parses and converts Syslog lines to CSV format.

Data Transformation Failure Handling

If your Lambda function invocation fails because of a network timeout or because you've reached the Lambda invocation limit, Firehose retries the invocation three times by default and then skips that batch of records if the invocation does not succeed. 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 with Amazon CloudWatch Logs (p. 36).

If the status of the data transformation of a record is `ProcessingFailed`, 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 AWS Lambda in the *AWS 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 Firehose.

`errorCode`
: The HTTP error code returned by Lambda.

`errorMessage`
: The error message returned by Lambda.

`attemptEndingTimestamp`
: The time that Firehose stopped attempting Lambda invocations.

`rawData`
: The base64-encoded record data.

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

Source Record Backup

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 delivery stream. Note that you cannot disable source record backup after you enable it.

Amazon Kinesis Firehose Data Delivery

After data is sent to your delivery stream, it is automatically delivered to the destination you choose.

Topics

Data Delivery Format

For data delivery to Amazon S3, Firehose concatenates multiple incoming records based on buffering configuration of your delivery stream, and then delivers them to Amazon S3 as an S3 object. You may want to add a record separator at the end of each record before you send it to Firehose so that you can divide a delivered S3 object to individual records.

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 cluster. You need to make sure that after Firehose concatenates multiple incoming records to an S3 object, the S3 object can be copied to your Amazon Redshift cluster. For more information, see Amazon Redshift COPY Command Data Format Parameters.

For data delivery to Amazon ES, Firehose buffers incoming records based on buffering configuration of your delivery stream, and then generates an Elasticsearch bulk request to index multiple records to your Elasticsearch cluster. You need to make sure that your record is UTF-8 encoded and flattened to a single line JSON object before you send it to Firehose. Also, the `rest.action.multi.allow_explicit_index` option for your Elasticsearch cluster needs to be set to true (default) in order to take bulk requests with an explicit index that is set per record. For more information, see Amazon ES Configure Advanced Options in the *Amazon Elasticsearch Service Developer Guide*.

Data Delivery Frequency

Each Firehose destination has its own data delivery frequency.

Amazon S3
: The frequency of data delivery to Amazon S3 is determined by the S3 **Buffer size** and **Buffer interval** value you configured for your delivery stream. Firehose buffers incoming data before delivering it to Amazon S3. You can configure the values for S3 **Buffer size** (1 MB to 128 MB) or **Buffer interval** (60 to 900 seconds), and the condition satisfied first triggers data delivery to Amazon S3. Note that in circumstances where data delivery to the destination is falling behind data writing to the delivery stream, Firehose raises the buffer size dynamically to catch up and make sure that all data is delivered to the destination.

Amazon Redshift
: The frequency of data **COPY** operations from Amazon S3 to Amazon Redshift is determined by how fast your Amazon Redshift cluster can finish the **COPY** command. If there is still data to copy, Firehose issues a new **COPY** command as soon as the previous **COPY** command is successfully finished by Amazon Redshift.

Amazon Elasticsearch Service
: The frequency of data delivery to Amazon ES is determined by the Elasticsearch **Buffer size** and **Buffer interval** values that you configured for your delivery stream. Firehose buffers incoming data before delivering it to Amazon ES. You can configure the values for Elasticsearch **Buffer size** (1 MB to 100 MB) or **Buffer interval** (60 to 900 seconds), and the condition satisfied first triggers data delivery to Amazon ES. Note that in circumstances where data delivery to the destination is falling behind data writing to the delivery stream, Firehose raises the buffer size dynamically to catch up and make sure that all data is delivered to the destination.

Data Delivery Failure Handling

Each Firehose destination has its own data delivery failure handling.

Amazon S3
: Data delivery to your S3 bucket might fail for reasons such as the bucket doesn't exist anymore, the IAM role that Firehose assumes doesn't have access to the bucket, network failure, or similar events. Under these conditions, Firehose keeps retrying for up to 24 hours until the delivery succeeds. The maximum data storage time of Firehose is 24 hours and your data is lost if data delivery fails for more than 24 hours.

Amazon Redshift
: For the Amazon Redshift destination, you can specify a retry duration (0-7200 seconds) when creating a delivery stream.

  Data delivery to your Amazon Redshift cluster might fail for reasons such as incorrect Amazon Redshift cluster configuration of your delivery stream, Amazon Redshift cluster under maintenance, network failure, or similar events. Under these conditions, Firehose retries for the specified time duration and skips that particular batch of 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 Elasticsearch Service
: For the Amazon ES destination, you can specify a retry duration (0-7200 seconds) when creating a delivery stream.

  Data delivery to your Amazon ES cluster might fail for reasons such as an incorrect Amazon ES cluster configuration of your delivery stream, an Amazon ES cluster under maintenance, network failure, or similar events. Under these conditions, 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 `elasticsearch_failed/` folder, which you can use for manual backfill. 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 Elasticsearch)",
    "errorMessage": "(error message returned by Elasticsearch)",
    "attemptEndingTimestamp": "(the time when Firehose stopped attempting
 index request)",
    "esDocumentId": "(intended Elasticsearch document ID)",
    "esIndexName": "(intended Elasticsearch index name)",
    "esTypeName": "(intended Elasticsearch type name)",
    "rawData": "(base64-encoded document data)"
}
```

Amazon S3 Object Name Format

Firehose adds a UTC time prefix in the format `YYYY/MM/DD/HH` before putting objects to Amazon S3. The prefix translates into the Amazon S3 folder structure, where each label separated by a forward slash (`/`) becomes a sub-folder. You can modify this folder structure by adding your own top-level folder with a forward slash (for example, `myApp/YYYY/MM/DD/HH`) or prepending text to the `YYYY` top-level folder name (for example, `myApp YYYY/MM/DD/HH`). This is accomplished by specifying an **S3 prefix** when creating the Firehose delivery stream, either by using the Firehose console or the API.

The S3 object name follows the pattern `DeliveryStreamName-DeliveryStreamVersion-YYYY-MM-DD-HH-MM-SS-RandomString`, where `DeliveryStreamVersion` begins with `1` and increases by 1 for every configuration change of the Firehose delivery stream. You can change Firehose delivery stream configurations (for example, the name of the S3 bucket, buffering hints, compression, and encryption) with the Firehose console, or by using the UpdateDestination API operation.

Index Rotation for the Amazon ES Destination

For the Amazon ES destination, you can specify a time-based index rotation option from one of the following five options: **NoRotation**, **OneHour**, **OneDay**, **OneWeek**, **OneMonth**.

Depending on the rotation option you choose, Firehose appends a portion of the UTC arrival timestamp to your specified index name, and rotates the appended timestamp accordingly. The following example shows the resulting index name in Amazon ES for each index rotation option, where 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`
OneMonth	`myindex-2016-02`

Monitoring Amazon Kinesis Firehose

You can monitor Amazon Kinesis Firehose using the following features:

- CloudWatch metrics (p. 33)— Firehose sends Amazon CloudWatch custom metrics with detailed monitoring for each delivery stream.
- CloudWatch logs (p. 36)— Firehose sends CloudWatch custom logs with detailed monitoring for each delivery stream.
- Amazon Kinesis Agent (p. 41)— Amazon Kinesis Agent publishes custom CloudWatch metrics to help assess if the agent is working as expected.
- API logging and history (p. 42)— Firehose uses AWS CloudTrail to log API calls and store the data in an Amazon S3 bucket, and to maintain API call history.

Monitoring with Amazon CloudWatch Metrics

Firehose integrates with CloudWatch metrics so that you can collect, view, and analyze CloudWatch metrics for your Firehose delivery streams. For example, you can monitor the `IncomingBytes` and `IncomingRecords` metrics to keep track of data ingested into Firehose from data producers.

The metrics that you configure for your Firehose delivery streams and agents are automatically collected and pushed to CloudWatch every five minutes. Metrics are archived for two weeks; after that period, the data is discarded.

The metrics collected for Firehose delivery streams are free of charge. For information about Amazon Kinesis agent metrics, see Monitoring Amazon Kinesis Agent Health (p. 41).

Topics

Service-level CloudWatch Metrics

The `AWS/Firehose` namespace includes the following service-level metrics.

Metric	Description
`DeliveryToElasticsearch.Bytes`	The number of bytes indexed to Amazon ES over the specified time period. Units: Bytes
`DeliveryToElasticsearch.Records`	The number of records indexed to Amazon ES over the specified time period. Units: Count
`DeliveryToElasticsearch.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
`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 Firehose to now) of the oldest record in 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.
`IncomingBytes`	The number of bytes ingested into the Firehose stream over the specified time period. Units: Bytes
`IncomingRecords`	The number of records ingested into the Firehose stream over the specified time period. Units: Count

API-Level CloudWatch Metrics

The `AWS/Firehose` namespace includes the following API-level metrics.

Metric	Description
`DescribeDeliveryStream.Latency`	The time taken per `DescribeDeliveryStream` operation, measured over the specified time period. Units: Milliseconds
`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 delivery 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 delivery 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
`PutRecordBatch.Records`	The total number of records from `PutRecordBatch` operations. Units: Count
`PutRecordBatch.Requests`	The total number of `PutRecordBatch` requests. Units: Count
`UpdateDeliveryStream.Latency`	The time taken per `UpdateDeliveryStream` operation, measured over the specified time period. Units: Milliseconds
`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: Seconds
`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

Accessing CloudWatch Metrics for Firehose

You can monitor metrics for 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.aws.amazon.com/cloudwatch/.
2. On the navigation bar, choose a region.
3. In the navigation pane, choose **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, choose **Firehose Metrics**.
5. Select the relevant row to view the statistics for the specified **MetricName** and **DeliveryStreamName** values.
6. (Optional) In the graph pane, select a statistic and a time period and then create a CloudWatch alarm using these settings.

To access metrics using the AWS CLI

Use the list-metrics and get-metric-statistics commands.

Monitoring with Amazon CloudWatch Logs

Firehose integrates with 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 Firehose error logging when you create your delivery stream.

If you enable Firehose error logging in the Firehose console, a log group and corresponding log streams are created for the delivery 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 delivery stream. The log stream name is **S3Delivery**, **RedshiftDelivery**, or **ElasticsearchDelivery**, depending on the delivery destination. Lambda invocation errors for data transformation are also logged to the log stream used for data delivery errors.

For example, if you create a delivery stream "MyStream" with Amazon Redshift as the destination and enable Firehose error logging, the following are created on your behalf: a log group named `aws/kinesisfirehose/MyStream` and two log streams named **S3Delivery** and **RedshiftDelivery**. In this example, the **S3Delivery** log stream is used for logging errors related to delivery failure to the intermediate S3 bucket, and the **RedshiftDelivery** log stream is used for logging errors related to Lambda invocation failure and delivery failure to your Amazon Redshift cluster.

If you enable Firehose error logging through the AWS CLI or an AWS SDK using the `CloudWatchLoggingOptions` configuration, you must create a log group and a log stream in advance. We recommend reserving that log group and log stream for Firehose error logging exclusively. Also ensure that the associated IAM policy has `"logs:putLogEvents"` permission. For more information, see Controlling Access with Amazon Kinesis Firehose (p. 47).

Note that Firehose does not guarantee that all delivery error logs are sent to CloudWatch Logs. In circumstances where delivery failure rate is high, 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 Firehose destination. Each error message also describes the proper action to take to fix the issue.

Errors

Amazon S3 Data Delivery Errors

Firehose can send the following Amazon S3-related errors to CloudWatch Logs.

Error Code	Error Message and Information
`S3.KMS.NotFoundException`	"The provided AWS KMS key was not found. If you are using what you believe to be a valid AWS KMS key with the correct role, check if there is a problem with the account to which the AWS 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 *AWS Key Management Service Developer Guide*.

Error Code	Error Message and Information
`S3.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 S3 bucket."
`S3.AccountProblem`	"There is a problem with your AWS account that prevents the operation from completing successfully. Contact AWS Support."
`S3.AllAccessDisabled`	"Access to the account provided has been disabled. Contact AWS Support."
`S3.InvalidPayer`	"Access to the account provided has been disabled. Contact AWS 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.MethodNotAllowed`	"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 AWS for resolution."

Amazon Redshift Data Delivery Errors

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." The destination table in Amazon Redshift to which data should be copied from S3 was not found. Note that 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 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."

Error Code	Error Message and Information
`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*.
`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 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."
`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 AWS for resolution."

Amazon Elasticsearch Service Data Delivery Errors

For the Amazon ES destination, Firehose sends errors to CloudWatch Logs as they are returned by Elasticsearch.

Lambda Invocation Errors

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 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 Firehose." For more information, see Data Transformation and Status Model (p. 27).
`Lambda.InvokeLimitExceeded`	"The Lambda concurrent execution limit is exceeded. Increase the concurrent execution limit." For more information, see AWS Lambda Limits in the *AWS 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 (p. 27).
`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 (p. 27).
`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`	"AWS 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*.
`Lambda.ENILimitReachedException`	"AWS 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*.

Accessing CloudWatch Logs for Firehose

You can view the error logs related to Firehose data delivery failure using the Firehose console or CloudWatch console. The following procedures show you how to access error logs using these two methods.

To access error logs using the Firehose console

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.
2. On the navigation bar, choose a region.
3. Select a delivery stream name to go to the delivery 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.aws.amazon.com/cloudwatch/.
2. On the navigation bar, choose a region.
3. In the navigation pane, choose **Logs**.
4. Select a log group and log stream to view a list of error logs related to data delivery failure.

Monitoring Amazon Kinesis Agent Health

Amazon Kinesis Agent publishes custom CloudWatch metrics with a namespace of **AWSKinesisAgent** to help assess if the agent is healthy, submitting data into 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 delivery 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 (p. 18).

There is a nominal charge for metrics emitted from Amazon Kinesis Agent, which are enabled by default. For more information, see Amazon CloudWatch Pricing.

Monitoring with CloudWatch

Amazon Kinesis Agent sends the following metrics to CloudWatch.

Metric	Description
`BytesSent`	The number of bytes sent to the Firehose delivery 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

Metric	Description
`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

Monitoring Amazon Kinesis Firehose API Calls with AWS CloudTrail

Amazon Kinesis Firehose is integrated with AWS CloudTrail, which captures API calls, delivers the log files to an Amazon S3 bucket that you specify, and maintains API call history. CloudTrail captures API calls made from the Firehose console or from your code to the Firehose API. With the information collected by CloudTrail, you can determine the request that was made to Firehose, the IP address from which the request was made, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the *AWS CloudTrail User Guide*.

Topics

Firehose and CloudTrail History

The CloudTrail API activity history feature lets you look up and filter events captured by CloudTrail. You can look up events related to the creation, modification, or deletion of resources in your AWS account on a per-region basis. Events can be looked up by using the CloudTrail console, or programmatically by using the AWS SDKs or AWS CLI.

The following actions are supported:

- CreateDeliveryStream
- DeleteDeliveryStream
- UpdateDestination

Firehose and CloudTrail Logging

When CloudTrail logging is enabled in your AWS account, API calls made to specific Firehose actions are tracked in CloudTrail log files. Firehose actions are written with other AWS service records. CloudTrail determines when to create and write to a new file based on the specified time period and file size.

The following actions are supported:

- CreateDeliveryStream
- DescribeDeliveryStream
- ListDeliveryStreams
- UpdateDestination
- DeleteDeliveryStream

Every log entry contains information about who generated the request. For example, if a request is made to create a delivery stream (**CreateDeliveryStream**), the identity of the person or service that made the request is logged. The identity information in the log entry helps you determine the following:

- Whether the request was made with root or 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 AWS service

For more information, see the CloudTrail userIdentity Element.

You can store your log files in your bucket for as long as needed but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

You can have CloudTrail publish SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For information, see Configuring Amazon SNS Notifications in the *AWS CloudTrail User Guide*.

You can also aggregate Firehose log files from multiple AWS regions and multiple AWS accounts into a single S3 bucket. For more information, see Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts in the *AWS CloudTrail User Guide*.

CloudTrail Log File Entries for Firehose

CloudTrail log files contain one or more log entries. Each entry lists multiple JSON-formatted events. A log entry 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. The log entries are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following is an example CloudTrail log entry. Note that for security reasons the username and password for **RedshiftDestinationConfiguration** will be omitted and returned as an empty string.

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

Controlling Access with Amazon Kinesis Firehose

The following topics cover how to control access to your Firehose resources and how to grant Firehose access to your Amazon Simple Storage Service (Amazon S3) bucket, Amazon Redshift cluster, or Amazon Elasticsearch Service cluster.

Contents

Firehose Resource Access

AWS Identity and Access Management (IAM) enables you to do the following:

- Create users and groups under your AWS account
- Assign unique security credentials to each user under your AWS account
- Control each user's permissions to perform tasks using AWS resources
- Allow the users in another AWS account to share your AWS resources
- Create roles for your AWS account and define the users or services that can assume them
- Use existing identities for your enterprise to grant permissions to perform tasks using AWS resources

By using IAM with Firehose, you can control whether users in your organization can perform a task using specific Firehose API actions and whether they can use specific AWS resources.

The following example JSON shows the format that your IAM policy should take. 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"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        }
    ]
}
```

Grant Firehose Access to an Amazon S3 Destination

When using an Amazon S3 destination, Firehose delivers data to your S3 bucket and can optionally use an AWS KMS key that you own for data encryption. If error logging is enabled, Firehose also sends data delivery errors to your CloudWatch log group and 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 log group and streams.

Use the following trust policy to enable Firehose to assume the role. Edit the policy below to replace *account-id* with your AWS account ID. This is so that only you can request Firehose to assume the IAM role.

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

Use the following access policy to enable Firehose to access your S3 bucket and AWS 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 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": [
               "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 AWS services to access your AWS resources, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.

Grant Firehose Access to an Amazon Redshift Destination

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

Topics

IAM Role and Access Policy

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

Use the following trust policy to enable Firehose to assume the role. Edit the policy below to replace *account-id* with your AWS account ID. This is so that only you can request Firehose to assume the IAM role.

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

Use the following access policy to enable Firehose to access your S3 bucket and AWS 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 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": [
               "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 AWS services to access your AWS resources, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.

VPC Access to an Amazon Redshift Cluster

If your Amazon Redshift cluster is in a VPC, it must be publicly accessible with a public IP address. You also need to grant Firehose access to your Amazon Redshift cluster by unblocking Firehose IP addresses. Firehose currently uses one CIDR block for each available region:

- `52.70.63.192/27` for US East (N. Virginia)
- `52.89.255.224/27` for US West (Oregon)
- `52.19.239.192/27` for EU (Ireland)

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

Grant Firehose Access to an Amazon Elasticsearch Service Destination

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

Use the following trust policy to enable Firehose to assume the role. Edit the policy below to replace *`account-id`* with your AWS account ID. This is so that only you can request Firehose to assume the IAM role.

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

Use the following access policy to enable Firehose to access your S3 bucket, Amazon ES domain, and AWS 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 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": [
               "es:DescribeElasticsearchDomain",
               "es:DescribeElasticsearchDomains",
               "es:DescribeElasticsearchDomainConfig",
               "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/indexname/
_mapping/typename",
               "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/indexname/
_stats"
           ]
        },
        {
           "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 AWS services to access your AWS resources, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.

Troubleshooting Amazon Kinesis Firehose

You may not see data delivered to your specified destinations. Use the following troubleshooting steps to solve common issues you might encounter while using Firehose.

Issues

Data Not Delivered to Amazon S3

Check the following if data is not delivered to your S3 bucket.

- Check the Firehose **IncomingBytes** and **IncomingRecords** metrics to make sure that data is sent to your Firehose delivery stream successfully. For more information, see Monitoring with Amazon CloudWatch Metrics (p. 33).
- If data transformation with Lambda is enabled, check the Firehose **ExecuteProcessingSuccess** metric to make sure that Firehose has attempted to invoke your Lambda function. For more information, see Monitoring with Amazon CloudWatch Metrics (p. 33).
- Check the Firehose **DeliveryToS3.Success** metric to make sure that Firehose has attempted putting data to your S3 bucket. For more information, see Monitoring with Amazon CloudWatch Metrics (p. 33).
- Enable error logging if it is not already enabled, and check error logs for delivery failure. For more information, see Monitoring with Amazon CloudWatch Logs (p. 36).
- Make sure that the S3 bucket specified in your Firehose delivery stream still exists.
- If data transformation with Lambda is enabled, make sure that the Lambda function specified in your delivery stream still exists.
- Make sure that the IAM role specified in your Firehose delivery stream has access to your S3 bucket and your Lambda function (if data transformation is enabled). For more information, see Grant Firehose Access to an Amazon S3 Destination (p. 48).

Data Not Delivered to Amazon Redshift

Check the following if data is not delivered to your Amazon Redshift cluster.

Note that data is delivered to your S3 bucket before loading into Amazon Redshift If the data was not delivered to your S3 bucket, see Data Not Delivered to Amazon S3 (p. 55).

- Check the Firehose **DeliveryToRedshift.Success** metric to make sure that Firehose has attempted to copy data from your S3 bucket to the Amazon Redshift cluster. For more information, see Monitoring with Amazon CloudWatch Metrics (p. 33).
- Enable error logging if it is not already enabled, and check error logs for delivery failure. For more information, see Monitoring with Amazon CloudWatch Logs (p. 36).
- Check the Amazon Redshift STL_CONNECTION_LOG table to see if Firehose is able to 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 of 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 delivery stream is accurate and valid.
- Make sure that the IAM role specified in your Firehose delivery stream has access to the S3 bucket from which Amazon Redshift copies data and the Lambda function for data transformation (if data transformation is enabled). For more information, see Grant Firehose Access to an Amazon S3 Destination (p. 48).
- If your Amazon Redshift cluster is in a VPC, make sure that the cluster allows access from Firehose IP addresses. For more information, see Grant Firehose Access to an Amazon Redshift Destination (p. 50).
- Make sure that the Amazon Redshift cluster is publicly accessible.

Data Not Delivered to Amazon Elasticsearch Service

Check the following if data is not delivered to your Elasticsearch domain.

Note that data can be backed up to your S3 bucket concurrently. If data was not delivered to your S3 bucket, see Data Not Delivered to Amazon S3 (p. 55).

- Check the Firehose **IncomingBytes** and **IncomingRecords** metrics to make sure that data is sent to your Firehose delivery stream successfully. For more information, see Monitoring with Amazon CloudWatch Metrics (p. 33).
- If data transformation with Lambda is enabled, check the Firehose **ExecuteProcessingSuccess** metric to make sure that Firehose has attempted to invoke your Lambda function. For more information, see Monitoring with Amazon CloudWatch Metrics (p. 33).
- Check the Firehose **DeliveryToElasticsearch.Success** metric to make sure that Firehose has attempted to index data to the Amazon ES cluster. For more information, see Monitoring with Amazon CloudWatch Metrics (p. 33).
- Enable error logging if it is not already enabled, and check error logs for delivery failure. For more information, see Monitoring with Amazon CloudWatch Logs (p. 36).
- Make sure that the Amazon ES configuration in your delivery stream is accurate and valid.

- If data transformation with Lambda is enabled, make sure that the Lambda function specified in your delivery stream still exists.
- Make sure that the IAM role specified in your delivery stream has access to your Amazon ES cluster and Lambda function (if data transformation is enabled). For more information, see Grant Firehose Access to an Amazon Elasticsearch Service Destination (p. 52).

Amazon Kinesis Firehose Limits

Amazon Kinesis Firehose has the following limits.

- By default, each account can have up to 20 Firehose delivery streams per region. This limit can be increased using the Amazon Kinesis Firehose Limits form.
- By default, each Firehose delivery stream can accept a maximum of 2,000 transactions/second, 5,000 records/second, and 5 MB/second. You can submit a limit increase request using the Amazon Kinesis Firehose Limits form. The three limits scale proportionally. For example, if you increase the throughput limit to 10MB/second, the other two limits increase to 4,000 transactions/second and 10,000 records/second.

  Important
  If the increased limit is much higher than the running traffic, this causes very small delivery batches to destinations, which is inefficient and can be costly. Be sure to increase the limit only to match current running traffic, and increase the limit further if traffic increases.
- Each Firehose delivery stream stores data records for up to 24 hours in case the delivery destination is unavailable.
- The maximum size of a record sent to Firehose, before base64-encoding, is 1000 KB.
- The PutRecordBatch operation can take up to 500 records per call or 4 MB per call, whichever is smaller. This limit cannot be changed.
- The following operations can provide up to 5 transactions per second CreateDeliveryStream, DeleteDeliveryStream, DescribeDeliveryStream, ListDeliveryStreams, and UpdateDestination.
- The buffer sizes hints range from 1 MB to 128 MB for Amazon S3 delivery, and 1 MB to 100 MB for Amazon Elasticsearch Service delivery. The size threshold is applied to the buffer before compression.
- The buffer interval hints range from 60 seconds to 900 seconds.
- For Firehose to Amazon Redshift delivery, only publicly accessible Amazon Redshift clusters are supported.
- The retry duration range is from 0 seconds to 7200 seconds for Amazon Redshift and Amazon ES delivery.

Document History

The following table describes the important changes to the Amazon Kinesis Firehose documentation.

Change	Description	Date Changed
New data transformation	You can configure Firehose to transform your data before data delivery. For more information, see Amazon Kinesis Firehose Data Transformation (p. 27).	December 19, 2016
New Amazon Redshift COPY retry	You can configure Firehose to retry a COPY command to your Amazon Redshift cluster if it fails. For more information, see Create a Firehose Delivery Stream to Amazon Redshift (p. 8), Amazon Kinesis Firehose Data Delivery (p. 30), and Amazon Kinesis Firehose Limits (p. 58).	May 18, 2016
New Firehose destination, Amazon Elasticsearch Service	You can create a delivery stream with Amazon Elasticsearch Service as the destination. For more information, see Create a Firehose Delivery Stream to Amazon Elasticsearch Service (p. 10), Amazon Kinesis Firehose Data Delivery (p. 30), and Grant Firehose Access to an Amazon Elasticsearch Service Destination (p. 52).	April 19, 2016
New enhanced CloudWatch metrics and troubleshooting features	Updated Monitoring Amazon Kinesis Firehose (p. 33) and Troubleshooting Amazon Kinesis Firehose (p. 55).	April 19, 2016
New enhanced Amazon Kinesis agent	Updated Writing to Amazon Kinesis Firehose Using Amazon Kinesis Agent (p. 16).	April 11, 2016
New Amazon Kinesis agents	Added Writing to Amazon Kinesis Firehose Using Amazon Kinesis Agent (p. 16).	October 2, 2015
Initial release	Initial release of the Amazon Kinesis Firehose *Developer Guide*.	October 4, 2015

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.