

Streaming Data Analytics with Amazon Kinesis Firehose and Amazon Redshift

Darin Briskman
Databases, Analytics, and AI
briskman@amazon.com

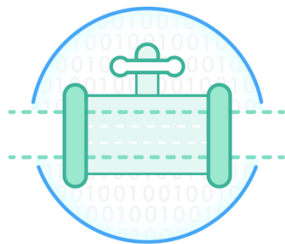
October 2017



What to Expect From This Session

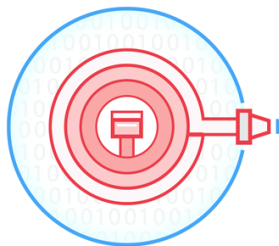
Amazon Kinesis streaming data on the AWS cloud

- Amazon Kinesis **Streams**
- Amazon Kinesis **Firehose**
- Amazon Kinesis **Analytics**



Amazon Kinesis Streams

Build your own custom applications that process or analyze streaming data



Amazon Kinesis Firehose

Easily load massive volumes of streaming data into Amazon S3 and Redshift



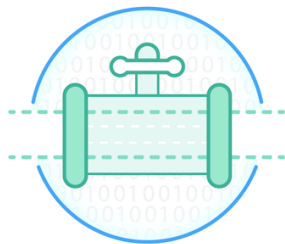
Amazon Kinesis Analytics

Easily analyze data streams using standard SQL queries

What to Expect From This Session

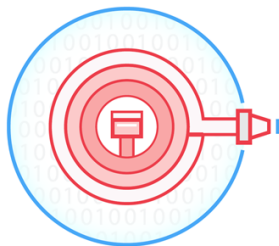
Amazon Kinesis streaming data on the AWS cloud

- Amazon Kinesis Streams
- Amazon Kinesis **Firehose** (focus of this session)
- Amazon Kinesis Analytics



Amazon Kinesis Streams

Build your own custom applications that process or analyze streaming data



Amazon Kinesis Firehose

Easily load massive volumes of streaming data into Amazon S3 and Redshift



Amazon Kinesis Analytics

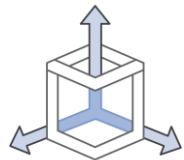
Easily analyze data streams using standard SQL queries

Amazon Kinesis: Streaming Data Done the AWS Way

Makes it easy to capture, deliver, and process real-time data streams



Easy to provision, deploy, and manage



Elastically scalable



Real-time latencies



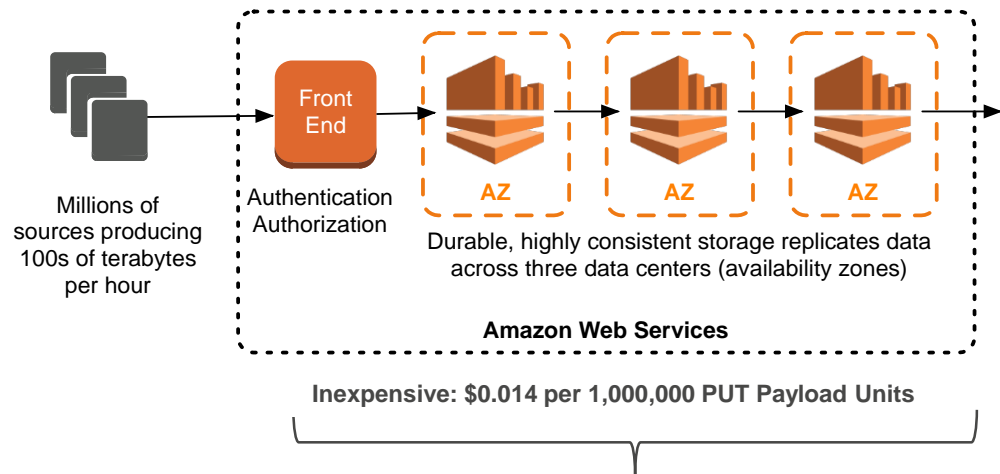
Pay as you go, no up-front costs



Right services for your specific use cases

Amazon Kinesis Streams

Fully managed service for real-time processing of streaming data



Real-Time Streaming Data Ingestion

Select Kinesis Customer **Case Studies**

Ad Tech



Gaming



IoT

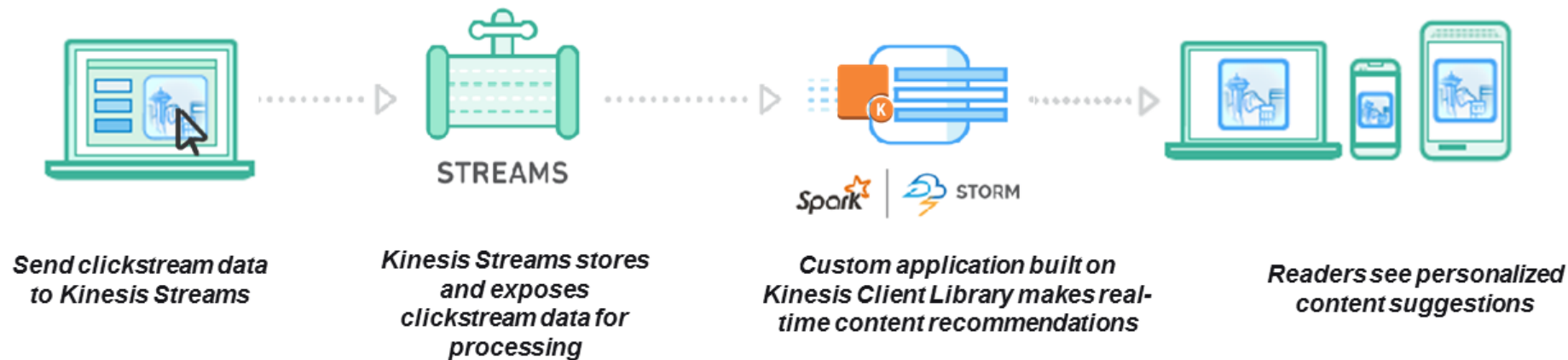




We have listened to our customers...

Amazon Kinesis Streams

Build your own data streaming applications



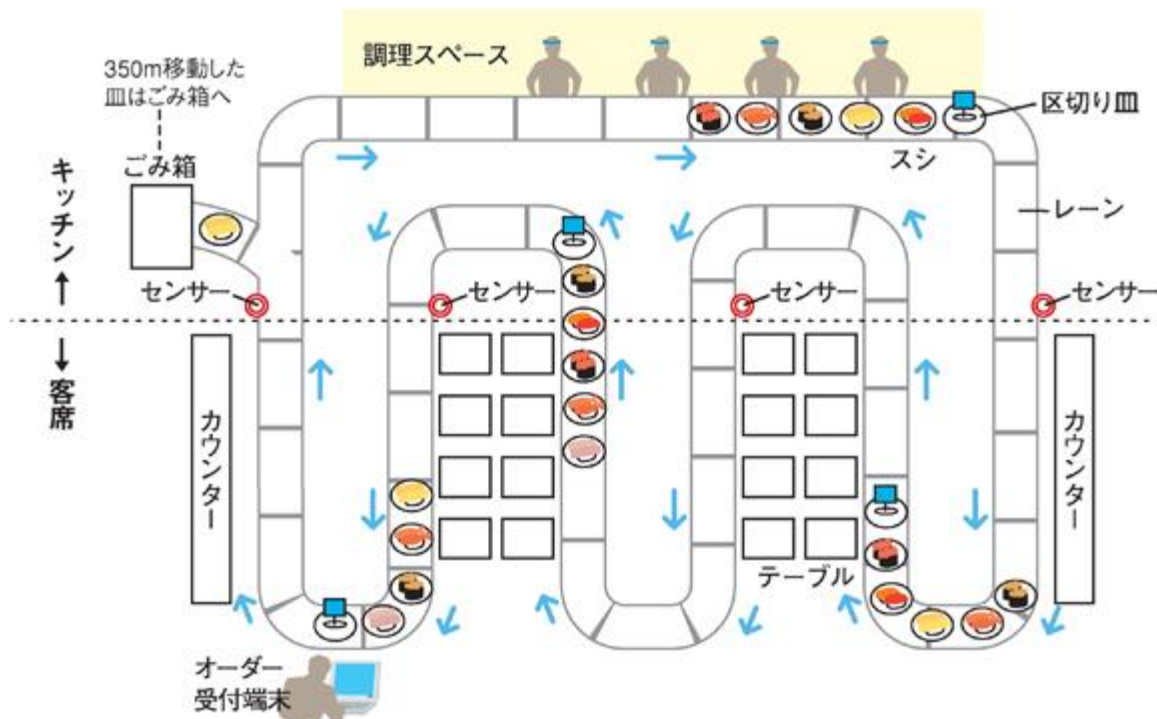
Easy administration: Simply create a new stream, and set the desired level of capacity with shards. Scale to match your data throughput rate and volume.

Build real-time applications: Perform continual processing on streaming big data using Kinesis Client Library (KCL), Apache Spark/Storm, AWS Lambda, and more.

Low cost: Cost-efficient for workloads of any scale.

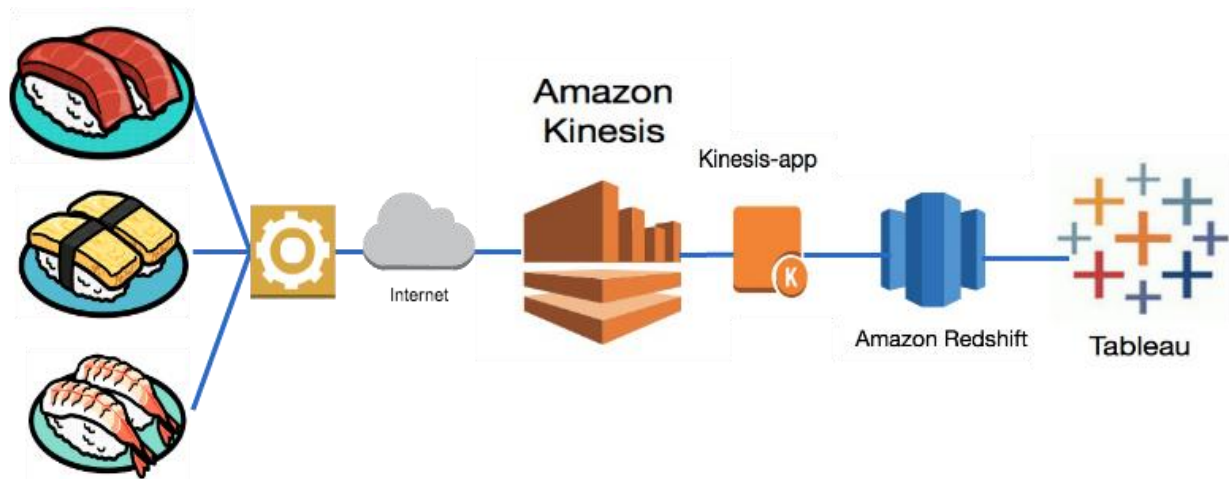
Sushiro: Kaiten Sushi Restaurants

380 stores stream data from sushi plate sensors and stream to Kinesis



Sushiro: Kaiten Sushi Restaurants

380 stores stream data from sushi plate sensors and stream to Kinesis



Sushiro: Kaiten Sushi Restaurants

380 stores stream data from sushi plate sensors and stream to Kinesis



Amazon Kinesis Firehose

Load massive volumes of streaming data into Amazon S3 and Amazon Redshift



Zero administration: Capture and deliver streaming data into S3, Redshift, and other destinations [without writing an application](#) or [managing infrastructure](#).

Direct-to-data store integration: [Batch](#), [compress](#), and [encrypt](#) streaming data for delivery into data destinations [in as little as 60 secs](#) using simple configurations.

Seamless elasticity: Seamlessly scales to match data throughput w/o intervention


Amazon Kinesis Firehose

3 Simple Concepts

1. **Delivery Stream:** The underlying entity of Firehose. Use Firehose by creating a delivery stream to a specified destination and send data to it.
 - You **do not** have to create a stream or provision shards.
 - You **do not** have to specify partition keys.
2. **Records:** The data producer sends data blobs as large as 1,000 KB to a delivery stream. That data blob is called a record.
3. **Data Producers:** Producers send records to a Delivery Stream. For example, a web server sends log data to a delivery stream is a data producer.

Amazon Kinesis Firehose Console Experience

Unified Console Experience for Firehose and Streams

 AWS Services Edit

Adi Krishnan N. Virginia Support

Amazon Kinesis

Kinesis Streams

Kinesis Firehose

Welcome to Amazon Kinesis Firehose


Amazon Kinesis Firehose is a fully managed, elastic service to easily deliver real-time data streams to destinations such as Amazon S3 and Amazon Redshift. You can start using Firehose by:

1. Creating a delivery stream
2. Sending your data to your delivery stream via Kinesis Agent or Kinesis Firehose APIs

Data will be automatically delivered to your specified destination.


Create Delivery Stream

Firehose Benefits




Easy to Use

Capture and deliver streaming data into destinations without writing any application or managing any infrastructure.



Direct to Data Stores

Batch, compress, and encrypt streaming data for delivery into your S3 bucket or Redshift cluster in as little as sixty seconds.




Zero Maintenance

Scale elastically to handle spikes in streaming data without intervention. Monitor the metrics for streaming data flowing into destinations.

Amazon Kinesis Firehose Console Experience

Create fully managed resources for delivery without building an app

 AWS Services Edit

Adi Krishnan N. Virginia Support

Amazon Kinesis

Kinesis Streams

Kinesis Firehose

Create Delivery Stream

Step 1: Destination

Step 2: Configuration

Step 3: Review

Destination ?

Select the destination where your streaming data will be delivered.

Destination* Amazon S3

Delivery stream name* sensorexhaust

S3 Bucket

S3 bucket* fhtest52

S3 prefix S3 prefix

IAM role* firehose_delivery_role

Firehose needs an IAM role to access your destination S3 bucket and KMS key. [Learn more](#)

*Required

Cancel Skip To Review Next

Amazon Kinesis Firehose Console Experience

Configure data delivery options simply using the console

AWS ▾

Services ▾

Edit ▾

Adi Krishnan ▾

Select a Region ▾

Create Firehose

Step 1: Choose Destination

Step 2: Configure Firehose

Step 3: Review

Configure Firehose

Data put into Firehose will be buffered and delivered into the destination using the configuration options you have selected.

Buffering Hints

Choose how you want your data buffered. You can choose to specify a buffer size in MB or a buffer interval in seconds. If both are specified, the condition that is met first is how data will be buffered.

Buffer Size (MB)*

?

Buffer Interval (s)*

?

Compression and encryption

Compression and encryption will be applied after data has been buffered on the server side.

Data Compression

?

Data Encryption using KMS

?

*required

Cancel

Previous

Next

Amazon Kinesis Firehose to Redshift

A two-step process

1

Use customer-provided S3 bucket as an intermediate destination

- Still the most efficient way to do large-scale loads to Redshift.
- Never lose data, always safe, and available in your S3 bucket.

Amazon Kinesis Firehose to Redshift

A two-step process

1

Use customer-provided S3 bucket as an intermediate destination

- Still the most efficient way to do large scale loads to Redshift.
- Never lose data, always safe, and available in your S3 bucket.

2

Firehose issues **customer-provided COPY command** synchronously. It continuously issues a COPY command once the previous COPY command is finished and acknowledged back from Redshift.

Amazon Kinesis Firehose Console

Configure data delivery to Redshift simply using the console

▼

Edit ▼

Adi: Krishnan

Create Delivery Stream

Step 1: Choose Destination

Step 2: Configure Destination

Step 3: Review

Choose Destination

Choose destination type for delivery stream to store your streaming data.

Destination* ⓘ

Delivery stream name* ⓘ

Intermediate S3 Bucket

S3 bucket for storing aggregated streaming data before copying to Redshift.

S3 destination bucket* ⓘ

S3 prefix ⓘ

IAM role* ⓘ

Firehose needs an IAM security role to access the AWS resources it needs. [Learn more](#) about Firehose IAM roles.

Redshift cluster

Redshift destination details.

Redshift cluster* ⓘ

Redshift database* ⓘ

Redshift table* ⓘ

Redshift table columns ⓘ

User name* ⓘ

Password* ⓘ

Copy Command Options

Command for copying data from intermediate S3 bucket to Redshift. See [Run the Copy Command](#) for instructions and samples.

*Required

Cancel

Skip To Review

Next

Amazon Kinesis Agent

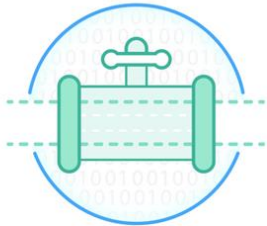
Software agent makes submitting data to Firehose easy

- Monitors files and sends new data records to your delivery stream
- Handles file rotation, checkpointing, and retry upon failures
- Delivers all data in a reliable, timely, and simple manner
- Emits AWS CloudWatch metrics to help you better monitor and troubleshoot the streaming process
 - Supported on Amazon Linux AMI with version 2015.09 or later, or Red Hat Enterprise Linux version 7 or later. Install on Linux-based server environments such as web servers, front ends, log servers, and more
- Also enabled for Streams

Amazon Kinesis Firehose API Overview

- **CreateDeliveryStream:** Create a DeliveryStream. You specify the S3 bucket information where your data will be delivered to.
- **DeleteDeliveryStream:** Delete a DeliveryStream.
- **DescribeDeliveryStream:** Describe a DeliveryStream and returns configuration information.
- **ListDeliveryStreams:** Return a list of DeliveryStreams under this account.
- **UpdateDestination:** Update the destination S3 bucket information for a DeliveryStream.
- **PutRecord:** Put a single data record (up to 1000KB) to a DeliveryStream.
- **PutRecordBatch:** Put multiple data records (up to 500 records OR 5MBs) to a DeliveryStream.

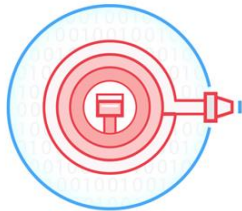
Amazon Kinesis Firehose **or** Amazon Kinesis Streams?



Amazon Kinesis Streams

Build your own custom applications that process or analyze streaming data

Amazon Kinesis Streams is a service for workloads that requires **custom processing**, per incoming record, with sub-1 second processing latency, and a choice of stream processing frameworks.



Amazon Kinesis Firehose

Easily load massive volumes of streaming data into Amazon S3 and Redshift

Amazon Kinesis Firehose is a service for workloads that require zero administration, ability to **use existing analytics tools based on S3, Amazon Elasticsearch Service, or Redshift**, and a data latency of 60 seconds or higher.

Data Delivery Methodology for Amazon S3

Controlling size and frequency of S3 objects

- Single delivery stream delivers to single S3 bucket
- Buffer size/interval values control size + frequency of data delivery
 - Buffer size – 1 MB to 128 MBs or
 - Buffer interval - 60 to 900 seconds
 - Firehose concatenates records into a single larger object
 - Condition satisfied first triggers data delivery
- (Optional) compress data after data is buffered
 - GZIP, ZIP, SNAPPY
 - Delivered S3 objects might be smaller than total data Put into Firehose
 - For Amazon Redshift, GZIP is only supported format currently

Data Delivery Methodology for Amazon S3

AWS IAM Roles and Encryption

- Firehose needs IAM role to access to your S3 bucket
 - Delivery stream assumes specified role to gain access to target S3 bucket
- (Optional) encrypt data with AWS Key Management Service
 - AWS KMS is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data
 - Uses Hardware Security Modules (HSMs) to protect keys
 - Firehose passes KMS-id to S3 which uses existing integration
 - S3 uses bucket and object name as encryption context

Data Delivery Methodology for Amazon S3

S3 Object Name Format

- Firehose adds a UTC time prefix in the format YYYY/MM/DD/HH before putting objects to Amazon S3. Translates to an Amazon S3 folder structure. Each '/' is a sub-folder
- Specify S3-prefix to get your own top-level folder
 - Modify this by adding your own top-level folder with '/' to get myApp/YYYY/MM/DD/HH
 - Or simply pre-pend text without a '/' to get myapp YYYY/MM/DD/HH
- Expect following naming pattern ***“DeliveryStreamName-DeliveryStreamVersion-YYYY-MM-DD-HH-MM-SS-RandomString”***
 - DeliveryStreamVersion == 1 if delivery stream config is not updated
 - DeliveryStreamVersion increases by 1 for every config change

Data Delivery Methodology for Amazon S3

If Firehose cannot reach your S3 bucket

- If your Amazon S3 bucket is not reachable by Amazon Kinesis Firehose, Firehose tries to access the S3 bucket every 5 seconds until the issue is resolved.
- Firehose durably stores your data for 24 hours
- It resumes delivery (per configurations) as soon as your S3 bucket becomes available and catches up
- Data can be lost if bucket is not accessible for > 24 hours

Data Delivery Methodology for Amazon Redshift

2-step process executed on your behalf

- Use customer-provided S3 bucket as an intermediate destination
 - Still the most efficient way to do large scale loads to Redshift
 - Never lose data, always safe, and available in your S3 bucket
- Firehose issues customer-provided Redshift COPY command synchronously
 - Loads data from intermediate-S3 bucket to Redshift cluster
 - Single delivery stream loads into a single Redshift cluster, database, and table

Data Delivery Methodology for Amazon Redshift

Frequency of loads into Redshift Cluster

- Based on delivery to S3 – buffer size and interval
- Redshift COPY command frequency
 - Continuously issues the Redshift COPY command once the previous one is acknowledged from Redshift.
 - Frequency of COPYs from Amazon S3 to Redshift is determined by how fast your Amazon Redshift cluster can finish the COPY command
- For efficiency, Firehose uses manifest copy
 - Intermediate S3 bucket contains 'manifests' folder – that holds a manifest of files that are to be copied into Redshift

Data Delivery Methodology for Amazon Redshift

Redshift COPY command mechanics

- Firehose issues the Redshift COPY command with zero error tolerance
- If a single record within a file fails, the whole batch of files within the COPY command fails
- Firehose does not perform partial file or batch copies to your Amazon Redshift table
- Skipped objects' information is delivered to S3 bucket as manifest file in the errors folder

Data Delivery Methodology for Amazon Redshift

If Redshift cluster is not accessible

- If Firehose cannot reach cluster it tries every 5 minutes for up to a total of 60 minutes
- If cluster is still not reachable after 60 minutes, it skips the current batch of S3 objects and moves on to next
- Like before, skipped objects' information is delivered to S3 bucket as manifest file in the errors folder
- Use info for manual backfill as appropriate

Amazon Kinesis Firehose Monitoring

Load to S3 specific metrics

Metrics	Description
Incoming. Bytes	The number of bytes ingested into the Firehose delivery stream.
Incoming. Records	The number of records ingested into the Firehose delivery stream.
DeliveryToS3. Bytes	The number of bytes delivered to Amazon S3
DeliveryToS3. DataFreshness	Age of the oldest record in Firehose. Any record older than this age has been delivered to the S3 bucket.
DeliveryToS3. Records	The number of records delivered to Amazon S3
DeliveryToS3. Success	Sum of successful Amazon S3 put commands over sum of all Amazon S3 put commands.

Amazon Kinesis Firehose Monitoring

Load to Redshift specific metrics

Metrics	Description
DeliveryToRedshift.Bytes	The number of bytes copied to Amazon Redshift
DeliveryToRedshift.Records	The number of records copied to Amazon Redshift
DeliveryToRedshift.Success	Sum of successful Amazon Redshift COPY commands over the sum of all Amazon Redshift COPY commands.

Amazon Kinesis Firehose Troubleshooting

Is data flowing end-to-end?

- Key question: Is data flowing through Firehose from producers to the S3 destination end-to-end?
- Key metrics: **Incoming.Bytes**, and **Incoming.Records** metrics along with **DeliveryToS3.Success** and **DeliveryToRedshift.Success** can be used to confirm that data is flowing end-to-end
- Additionally, check the Put* APIs Bytes, Latency, and Request metrics for Firehose and or S3

Amazon Kinesis Firehose Troubleshooting

What is being captured, and what is being delivered?

- Key question: How much data is being captured and delivered to the destination?
- Key metrics: Put* **Records**, **Bytes**, and **Requests** to determine the data volume captured by Firehose. Next check **DeliveryToS3.Bytes/ Records** and **DeliverytoRedshift.Bytes/Records**
- Additionally, check the S3 bucket related metrics above confirm the data volume delivered to S3. Verify with **Incoming.Records** and **Incoming.Bytes**

Amazon Kinesis Firehose Troubleshooting

Is my data showing 'on time'?

- Key question: Is Firehose delivering data in a timely way?
- Key metrics: **DeliveryToS3.DataFreshness** in seconds helps determine freshness of data. It shows the maximum age in seconds of oldest undelivered record in Firehose.
- Any record that is older than this age has been delivered into S3.

Amazon Kinesis Firehose Troubleshooting

Something wrong with my Firehose or bucket or cluster?

- Key question: Is something wrong with Firehose or with the customers' own configuration, say S3 bucket?
- Key metrics: The **DeliveryToS3.Success** metric computes the sum of successful S3 Puts over sum of all S3 puts managed by Firehose. Similarly **DeliveryToRedshift.Success** metric computes sum of successful COPY commands over sum of all COPY commands.
- If this trends below “1” it suggests potential issues with the S3 bucket configuration such that Firehose puts to S3 are failing.

Amazon Kinesis Firehose Troubleshooting

Something key steps if data isn't showing up in S3

- Check Incoming.Bytes and Incoming.Records metrics to ensure that data is Put successfully.
- Check DeliveryToS3.Success metric to ensure that Firehose is putting data to your S3 bucket.
- Ensure that the S3 bucket specified in your delivery stream still exists.
- Ensure that the IAM role specified in your delivery stream has access to your S3 bucket.

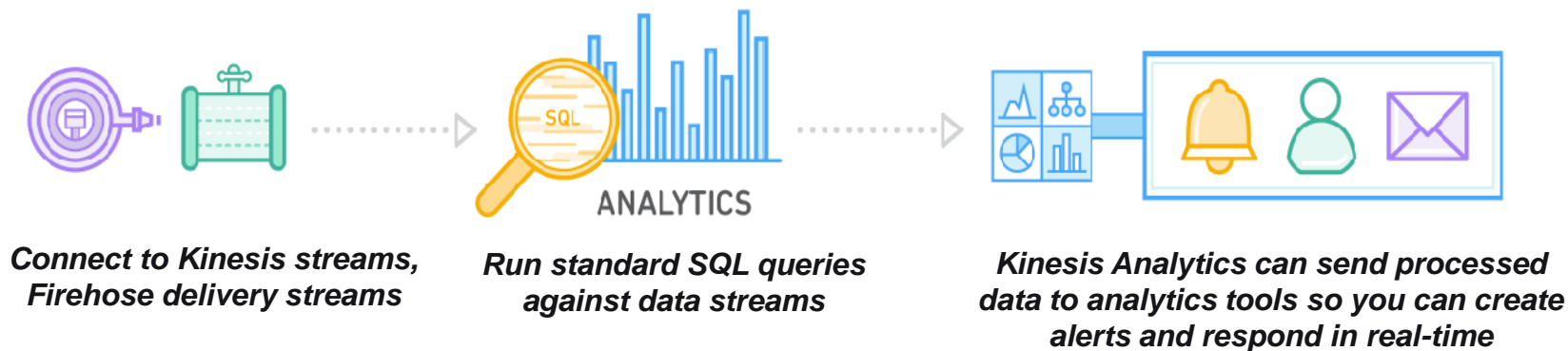
Amazon Kinesis Firehose Troubleshooting

Something key steps if data isn't showing up in Redshift

- Check Incoming.Bytes and Incoming.Records metrics to ensure that data is Put successfully, and the DeliveryToS3.Success metric to ensure that data is being staged in the S3 bucket.
- Check DeliveryToRedshift.Success metric to ensure that Firehose has copied data from your S3 bucket to the cluster.
- Check Redshift STL_LOAD_ERRORS table to verify the reason of the COPY failure.
- Ensure that Redshift config in Firehose is accurate

Amazon Kinesis Analytics

Analyze data streams continuously with standard SQL



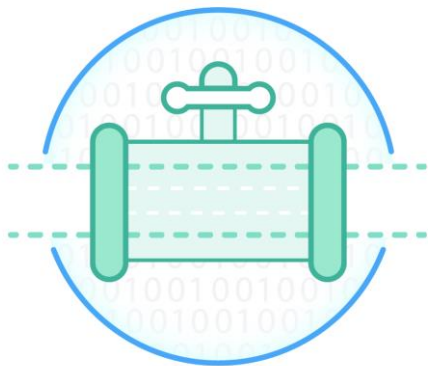
Apply SQL on streams: Easily connect to data streams and apply existing SQL skills.

Build real-time applications: Perform continual processing on streaming big data with sub-second processing latencies

Scale elastically: Elastically scales to match data throughput without any operator intervention.

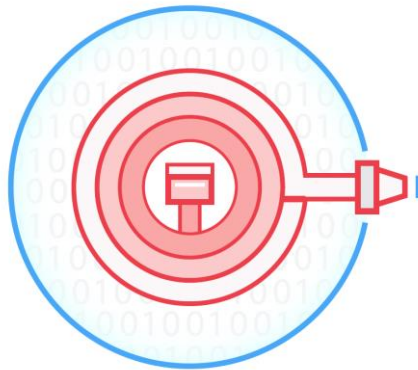
Amazon Kinesis: Streaming data made easy

Services make it easy to capture, deliver, and process streams on AWS



Amazon Kinesis Streams

Build your own custom applications that process or analyze streaming data



Amazon Kinesis Firehose

Easily load massive volumes of streaming data into Amazon S3 and Redshift



Amazon Kinesis Analytics

Easily analyze data streams using standard SQL queries

Q&A

Darin Briskman
briskman@amazon.com
aws.amazon.com/activate