AWS White Paper - Streaming Data Solutions on AWS with Amazon Kinesis

https://d0.awsstatic.com/whitepapers/whitepaper-streaming-data-solutions-on-aws-with-amazon-kinesis.pdf

Streaming Data Solutions on AWS with Amazon Kinesis

July 2017

they recognized that Kinesis Firehose can receive a stream of data records and insert them into Amazon Redshift. They created a Kinesis Firehose delivery stream and configured it so that it would copy data to their Amazon Redshift table every 15 minutes. Their current solution stores records to a file system as part of their batch process. As part of this new solution, they used the Amazon Kinesis Agent on their servers to forward their log data to Kinesis Firehose. Since Kinesis Firehose uses Amazon S3 to store raw streaming data before it is copied to Amazon Redshift, ABC Tolls didn't need to build another solution to archive their raw data.

Figure 1 depicts this solution.

Figure 1: New solution using Amazon Kinesis Firehose

Amazon Kinesis Firehose

Amazon Kinesis Firehose is the easiest way to load streaming data into AWS. It can capture, transform, and load streaming data into Amazon Kinesis Analytics, Amazon S3, Amazon Redshift, and Amazon Elasticsearch Service, enabling near real-time analytics with existing business intelligence tools and dashboards that you're already using today. It's a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security.

Sending Data to an Amazon Kinesis Firehose Delivery Stream

To send data to your delivery stream, there are several options. AWS offers SDKs for many popular programming languages, each of which provides APIs for Kinesis Firehose. AWS has also created a utility to help send data to your delivery stream.

Using the API

The Kinesis Firehose API offers two operations for sending data to your delivery stream. `PutRecord` sends one data record within one call. `PutRecordBatch` can send multiple data records within one call.

> In each method, you must specify the name of the delivery stream and the data record, or array of data records, when using the method. Each data record consists of a data blob that can be up to 1,000 KB in size and any kind of data.

Using the Amazon Kinesis Agent

The Amazon Kinesis Agent is a stand-alone Java software application that offers an easy way to collect and send data to Kinesis Streams and Kinesis Firehose. The agent continuously monitors a set of files and sends new data to your 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.

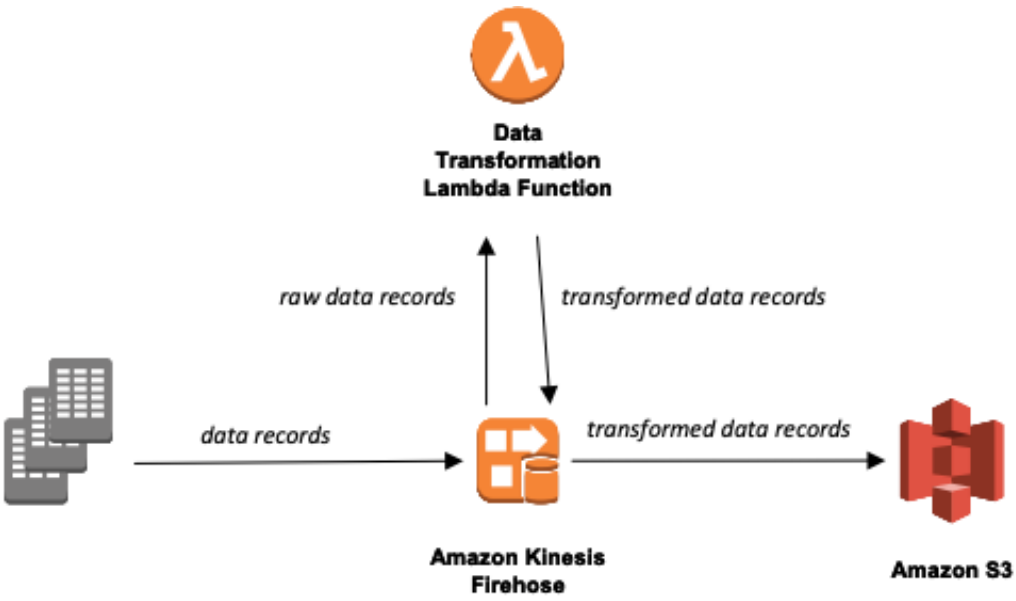
Data Transformation

In some scenarios, you might want to transform or enhance your streaming data before it is delivered to its destination. For example, data producers might send unstructured text in each data record, and you need to transform it to JSON before delivering it to Amazon Elasticsearch Service.

To enable streaming data transformations, Kinesis Firehose uses an AWS Lambda function that you create to transform your data.[5]

Data Transformation Flow

When you enable Kinesis Firehose data transformation, Kinesis Firehose buffers incoming data up to 3 MB or the buffering size you specified for the delivery stream, whichever is smaller. Kinesis Firehose then invokes the specified Lambda function with each buffered batch asynchronously. The transformed data is sent from Lambda to Kinesis Firehose for buffering. Transformed data is delivered to the destination when the specified buffering size or buffering interval is reached, whichever happens first. Figure 2 depicts this process for a delivery stream that delivers data to Amazon S3.



Data Delivery Format

For data delivery to Amazon S3, Kinesis Firehose concatenates multiple incoming records based on the 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 Kinesis Firehose so that you can divide a delivered S3 object to individual records.

Data Delivery Frequency

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. Kinesis Firehose buffers incoming data before delivering it to Amazon S3. You can configure the values for the Amazon S3 buffer size (1 MB to 128 MB) or buffer interval (60 seconds to 900 seconds). 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, Kinesis Firehose raises the buffer size dynamically to catch up and make sure that all data is delivered to the destination.

Figure 3 shows the flow of data for Amazon S3 destinations.

Figure 3: Data flow from Kinesis Firehose to S3 buckets

Amazon Redshift

Amazon Redshift is a fast, fully managed data warehouse that makes it simple and cost-effective to analyze all your data using standard SQL and your existing business intelligence (BI) tools.[7] It allows you to run complex analytic queries against petabytes of structured data using sophisticated query optimization, columnar storage on high-performance local disks, and massively parallel query execution. Most results come back in seconds.

Data Delivery Format

For data delivery to Amazon Redshift, Kinesis Firehose first delivers incoming data to your S3 bucket in the format described earlier. Kinesis 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 Kinesis 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.

Data Flow

Figure 4 shows the flow of data for Amazon Redshift destinations.

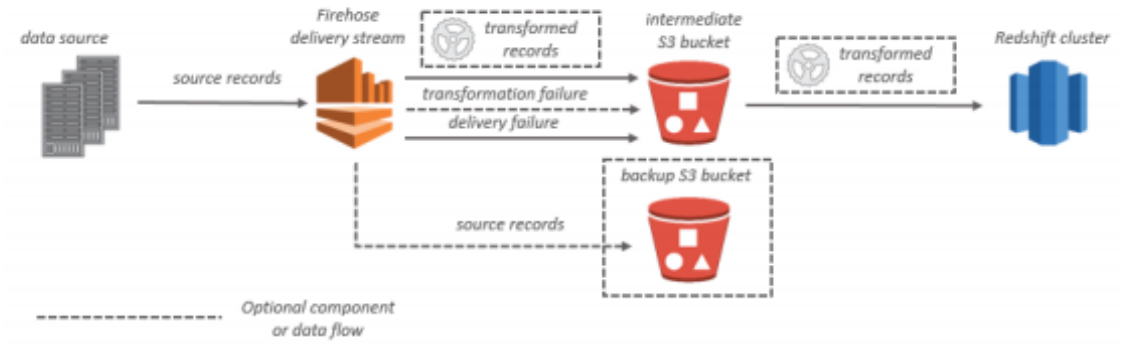


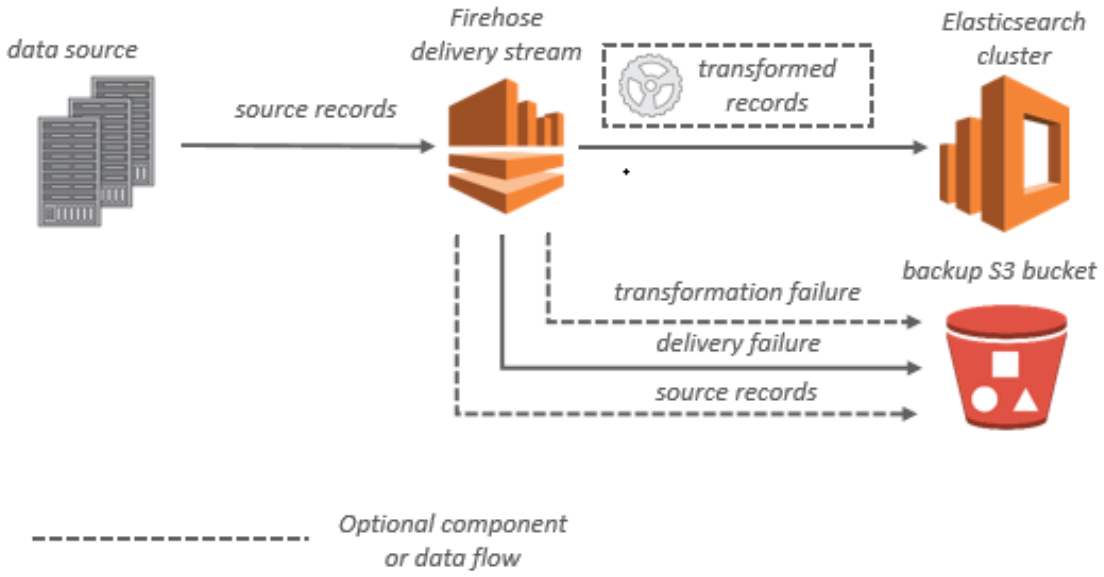
Figure 4: Data flow from Kinesis Firehose to Amazon Redshift

Amazon Elasticsearch Service

Amazon Elasticsearch Service (Amazon ES) is a fully managed service that delivers the Elasticsearch easy-to-use APIs and real-time capabilities along with the availability, scalability, and security required by production workloads.[8] Amazon ES makes it easy to deploy, operate, and scale Elasticsearch for log analytics, full text search, application monitoring, and more.

Data Delivery Format

For data delivery to Amazon ES, Kinesis Firehose buffers incoming records based on the 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 Kinesis Firehose.



Amazon Kinesis Analytics

With Kinesis Analytics, you can process and analyze streaming data using SQL. The service enables you to quickly author and run powerful SQL code against streaming sources to perform time series analytics, feed real-time dashboards, and create real-time metrics.

To get started with Kinesis Analytics, you create a Kinesis Analytics application that continuously reads and processes streaming data. The service supports ingesting data from Kinesis Streams and Kinesis Firehose streaming sources. You then author your SQL code using the interactive editor and test it with live streaming data. You can also configure destinations where you want Kinesis Analytics to persist the results. Kinesis Analytics supports Kinesis Firehose (Amazon S3, Amazon Redshift, and Amazon Elasticsearch Service), and Kinesis Streams as destinations.

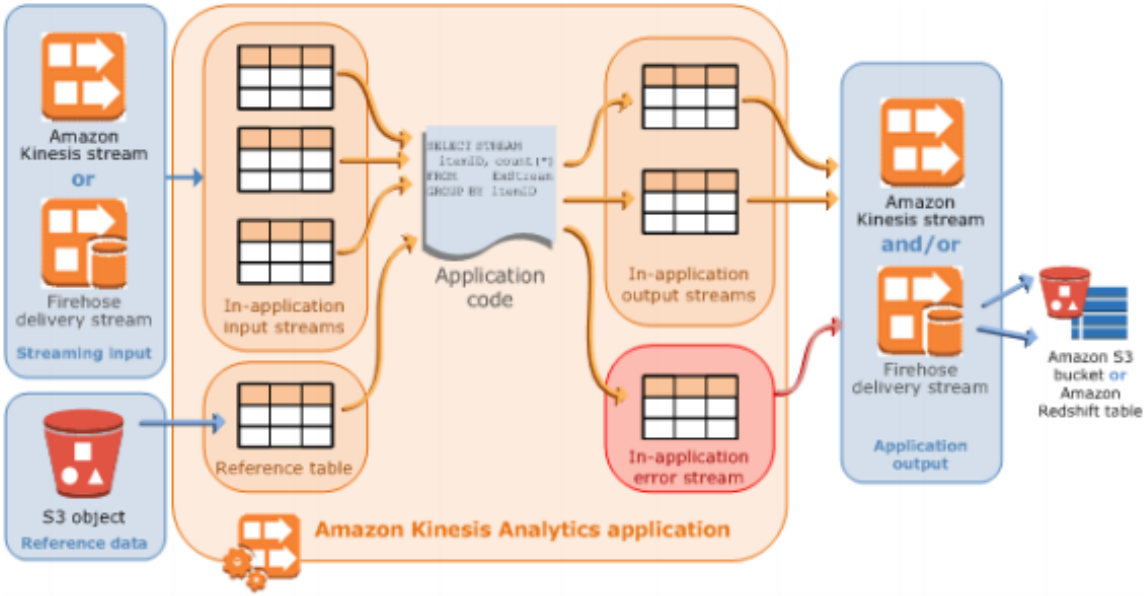


Figure 7: Architecture for a Kinesis Analytics application

Amazon Kinesis Streams

Amazon Kinesis Streams enables you to build custom, real-time applications using popular stream processing frameworks and load streaming data into any data store. You can configure hundreds of thousands of data producers to continuously put data into a Kinesis stream, for example, data from website clickstreams, application logs, IoT sensors, and social media feeds. Within less than a second, the data will be available for your application to read and process from the stream.

When implementing a solution with Kinesis Streams, you will create custom data-processing applications known as *Kinesis Streams applications*. A typical Kinesis Streams application reads data from a Kinesis stream as data records.

Sending Data to Amazon Kinesis Streams

There are several mechanisms to send data to your stream. AWS offers SDKs for many popular programming languages, each of which provides APIs for Kinesis Streams. AWS has also created several utilities to help send data to your stream. Let's review each of the approaches you can use and why you might choose each.

Amazon Kinesis Agent

The Amazon Kinesis Agent was discussed earlier as a tool that can be used to send data to Kinesis Firehose. The same tool can be used to send data to Kinesis Streams. For details on installing and configuring the Kinesis agent, see Writing to Amazon Kinesis Firehose Using Amazon Kinesis Agent.[10]

Amazon Kinesis Producer Library (KPL)

The KPL simplifies producer application development, allowing developers to achieve high write throughput to one or more Kinesis streams. The KPL is an easy-to-use, highly configurable library that you install on your hosts that generate the data that you wish to stream to Kinesis Streams. It acts as an intermediary between your producer application code and the Kinesis Streams API actions. The KPL performs the following primary tasks:

- Writes to one or more Kinesis streams with an automatic and configurable retry mechanism

- Collects records and uses `PutRecords` to write multiple records to multiple shards per request
- Aggregates user records to increase payload size and improve throughput
- Integrates seamlessly with the Amazon Kinesis Client Library (KCL) to de-aggregate batched records on the consumer
- Submits Amazon CloudWatch metrics on your behalf to provide visibility into producer performance

The KPL can be used in either synchronous or asynchronous use cases. We suggest using the higher performance of the asynchronous interface unless there is a specific reason to use synchronous behavior. For more information about these two use cases and example code, see Writing to your Streams Stream Using the KPL.[11]

Because the KPL buffers your records before they're sent to a Kinesis stream, the KPL can incur an additional processing delay, depending on the length of time you've configured the KPL to buffer records before sending them to Kinesis. Larger buffer time results in higher packing efficiencies and better performance. Applications that cannot tolerate this additional delay may need to use the AWS SDK directly.

Amazon Kinesis API

After a stream is created, you can add your data records to it. A record is a data structure that contains the data to be processed in the form of a data blob. After you store the data in the record, Kinesis Streams does not inspect, interpret, or change the data in any way.

There are two different operations in the Kinesis Streams API that add data to a stream: `PutRecords` and `PutRecord`. The `PutRecords` operation sends multiple records to your stream per HTTP request, and the singular `PutRecord` operation sends records to your stream one at a time (a separate HTTP request is required for each record). You should prefer using `PutRecords` for most applications because it will achieve higher throughput per data producer.

Because the APIs are exposed in all AWS SDKs, using the API to write records provides the most flexible solution to send data to a Kinesis stream. If you are unable to use the Kinesis Agent or KPL (for example, you want to write messages directly from a mobile application, or you want to minimize message end-to-end latency as much as possible), then use the APIs to write records to your Kinesis stream.

Using the Amazon Kinesis Client Library (KCL)

You can develop a consumer application for Kinesis Streams using the KCL. Although you can use the Kinesis Streams API to get data from an Amazon Kinesis stream, we recommend using the design patterns and code for consumer applications provided by the KCL.

The KCL helps you consume and process data from a Kinesis stream. This type of application is also referred to as a consumer. The KCL takes care of many of the complex tasks associated with distributed computing, such as load balancing across multiple instances, responding to instance failures, checkpointing processed records, and reacting to resharding. The KCL enables you to focus on writing record-processing logic.

The KCL is a Java library; support for languages other than Java is provided using a multi-language interface. At run time, a KCL application instantiates a worker with configuration information, and then uses a record processor to process the data received from a Kinesis stream. You can run a KCL application on any number of instances. Multiple instances of the same application coordinate on failures and load-balance dynamically. You can also have multiple KCL applications working on the same stream, subject to throughput limits. The KCL acts as an intermediary between your record processing logic and Kinesis Streams.

Using AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers.[16] AWS Lambda executes your code only when needed and scales automatically. With AWS Lambda, you can run code with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and code monitoring and logging. All you need to do is supply your code in one of the languages that AWS Lambda supports.

You can subscribe Lambda functions to automatically read batches of records off your Kinesis stream and process them if records are detected on the stream.

AWS Lambda then polls the stream periodically (once per second) for new records. When it detects new records, it invokes your Lambda function by passing the new records as a parameter. If no new records are detected, your Lambda function is not invoked.

Using the API

For most use cases, you should use the KCL or AWS Lambda to retrieve and process data from a stream. However, if you prefer to write your own consumer application from scratch, there are several methods that enable this. The Kinesis Streams API provides the `GetShardIterator` and `GetRecords` methods to retrieve data from a stream. This is a pull model, where your code draws data directly from the shards of the stream. For more information about writing your own consumer application using the API, refer to Developing Amazon Kinesis Streams Consumers Using the Amazon Kinesis Streams API. Details about the API can be found in the Amazon Kinesis Streams API Reference.[18]

Choosing the Best Consumer Model for Your Application

How do you know which consumer model is best for your use case? Each approach has its own set of tradeoffs and you'll need to decide what's important to you. Here is some general guidance to help you choose the correct consumer model.

In most cases, consider starting with AWS Lambda. Its ease of use and the simple deployment model will enable you to quickly build a data consumer. The tradeoff to using AWS Lambda is that each invocation of your Lambda function should be considered stateless. That is, you can't easily use results from

previous invocations of your function (e.g., earlier batches of records from your stream). Also consider that the maximum execution time for a single Lambda function is 5 minutes. If a single batch of records takes longer than 5 minutes to process, AWS Lambda might not be the best consumer for your use case.

If you decide that you can't use AWS Lambda, consider building your own processing application with the KCL. Because you deploy KCL applications to EC2 instances within your AWS account, you have a lot of flexibility and control in the local data persistence and state requirements for your data.