Kinesis

19 June 2023 15:42

Amazon **Kinesis** is a scalable and durable real-time data streaming service to ingest and analyze data in real-time from multiple data sources. Its Amazon's fully managed service for collecting, processing and analyzing streaming data in the cloud. Some examples where Kinesis can be used streaming data are trading data, geospatial data, social networks data, any realtime monitoring solutions etc.

There are 4 different types of Kinesis streams

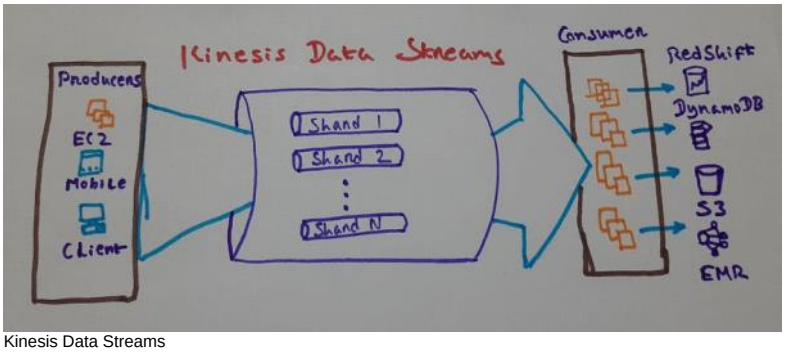
1. Kinesis Data Streams

2. Kinesis Firehose Delivery Streams

3. Kinesis Video Analytics

4. Kinesis Data Analytics

Kinesis Data Streams :



Kinesis Data Streams

Producers will produce data from different sources and ingested into the Kinesis Data Stream and it distributes the data amongst its shards and finally send it to the consumers. Consumers need to be configured by us programmatically as per the requirement. We can have multiple consumers.

When data enters the data stream, it can be persistent from 24 hours to 168 hours before it disappears from the stream.

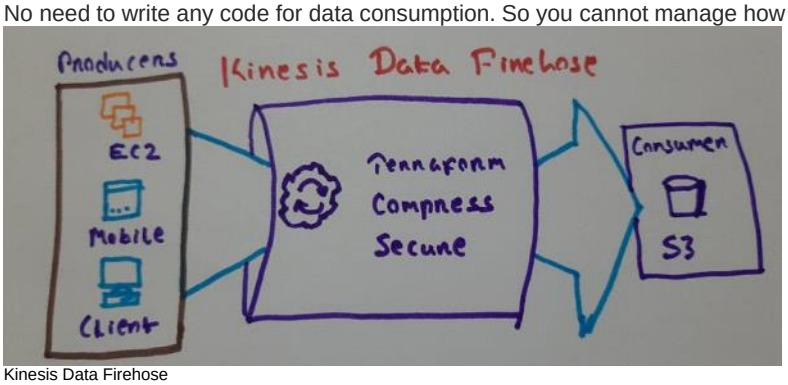
The way you pay is as per the running shard.

Kinesis Data Firehose :

It is similar to data stream, rather it is lot simpler. The difference is as soon as the data consumed by consumers, it immediately disappears from the queue, so no persistent of data.

Here, we can choose one consumer from a pre-defined list like S3, RedShift, ElasticSearch or Splunk.

No need to write any code for data consumption. So you cannot manage how you want to consume the data.



Kinesis Data Firehose

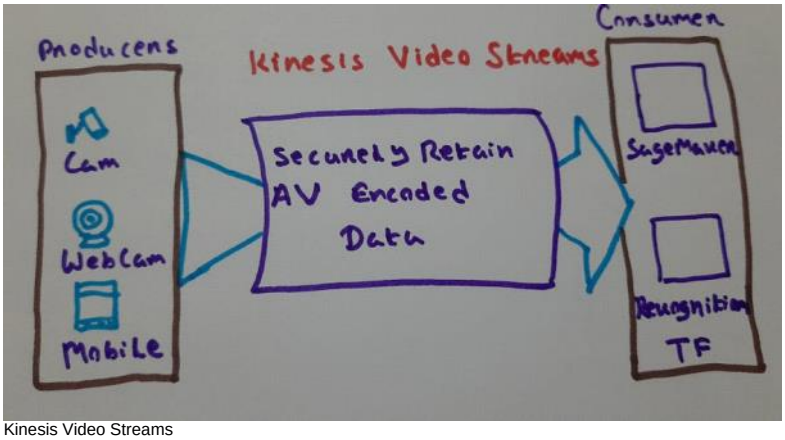
Firehose can transform the data to other formats like JSON, or can be compressed or secured data.

Here, you pay only for the data ingested, so inexpensive.

Kinesis Video Streams :

As the name suggests, it is used for ingesting encoded video and audio data from different devices or service.

The output audio or video data to ML services like SageMaker or Rekognition or any customised AV processing services, so we can analyse and process the AV streams.



Kinesis Video Streams

Kinesis Data Analytics :

Kinesis Data Analytics takes stream like data, Kinesis data stream or Kinesis firehose as inut and output. Data that pass through the Data Analytics is run through customized SQL and the results are send to output. This allows us for real-time analysis of data like realtime reporting. Here as you will be using two streams, so little bit expensive than others, but useful for the required application.

AWS Kinesis Data Streams

In today's world, data sources are growing aplenty due to which data is increasing at an explosive rate. It may includes logs, click streams, IoT sensory data etc. And it may be required to do an analysis and storage of this data on real time or near real time basis. This is where streaming data services comes into picture. they are expected to transmit data and use cases may need to be developed based on this data to analyse and provide output in real time using machine learning algorithms. Some of the common use cases could be stock market analysis, fraud detection, monitoring etc.

Streaming data is different from batch data that used to rule the roost until few years ago in way that the streaming data comes at a blistering pace and consumers of the data must also be built in such a way as to handle such data at scale and speeds. The table below may quickly elaborate differences between batch processing and stream processing.

	Batch	Stream
Scope	Queries or processing over all or most of the data in the dataset.	Queries or processing over data within a rolling time window
Size	Large batches of data	Individual records or micro batches consisting of a few records
Performance	Latencies in minutes to hours	Requires latency in the order of seconds or milliseconds
Analysis	Complex analytics	Simple response functions, aggregates, and rolling metrics

This is where AWS Kinesis Streams comes into picture. AWS Kinesis is a serverless streaming service that can be used to build highly scalable event driven applications. It makes capturing, storing and processing data much easier. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications. Amazon Kinesis enables you to process and analyze data as it arrives and respond instantly instead of having to wait until all your data is collected before the processing can begin. Some of the advantages of using Kinesis Streams are:

Scalable

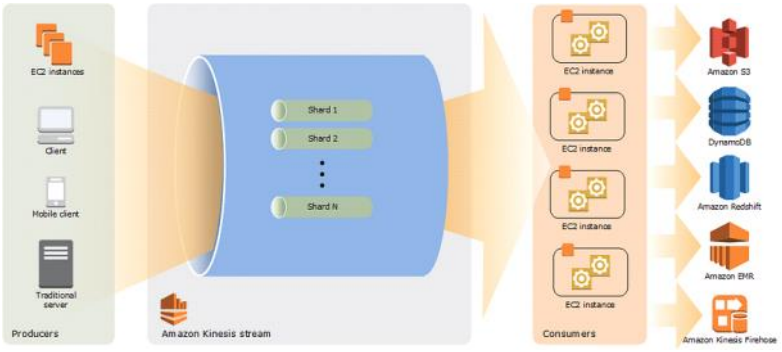
Amazon Kinesis can handle any amount of streaming data and process data from hundreds of thousands of sources with very low latencies.

Fully managed

Amazon Kinesis is fully managed and runs your streaming applications without requiring you to manage any infrastructure.

Real-time

Amazon Kinesis enables you to ingest, buffer, and process streaming data in real-time, so you can derive insights in seconds or minutes instead of hours or days.



Key Concepts

Kinesis Data Stream

A Kinesis data stream is a set of shards. Each shard has a sequence of data records. Each data record has a sequence number that is assigned by Kinesis Data Streams.

Data Record

A data record is the unit of data stored in a Kinesis data stream. Data records are composed of a sequence number, a partition key, and a data blob, which is an immutable sequence of bytes. Kinesis Data Streams does not inspect, interpret, or change the data in the blob in any way. A data blob can be up to 1 MB.

Capacity Mode

A data stream capacity mode determines how capacity is managed and how you are charged for the usage of your data stream. Currenly, in Kinesis Data Streams, you can choose between an **on-demand** mode and a **provisioned** mode for your data streams.

With the **on-demand** mode, Kinesis Data Streams automatically manages the shards in order to provide the necessary throughput. You are charged only for the actual throughput that you use and Kinesis Data Streams automatically accommodates your workloads' throughput needs as they ramp up or down.

With the **provisioned** mode, you must specify the number of shards for the data stream. The total capacity of a data stream is the sum of the capacities of its shards. You can increase or decrease the number of shards in a data stream as needed and you are charged for the number of shards at an hourly rate.

In today's world, data sources are growing aplenty due to which data is increasing at an explosive rate. It may includes logs, click streams, IoT sensory data etc. And it may be required to do an analysis and storage of this data on real time or near real time basis. This is where streaming data services comes into picture. they are expected to transmit data and use cases may need to be developed based on this data to analyse and provide output in real time using machine learning algorithms. Some of the common use cases could be stock market analysis, fraud detection, monitoring etc.

Streaming data is different from batch data that used to rule the roost until few years ago in way that the streaming data comes at a blistering pace and consumers of the data must also be built in such a way as to handle such data at scale and speeds. The table below may quickly elaborate differences between batch processing and stream processing.

	Batch	Stream
Scope	Queries or processing over all or most of the data in the dataset.	Queries or processing over data within a rolling time window
Size	Large batches of data	Individual records or micro batches consisting of a few records
Performance	Latencies in minutes to hours	Requires latency in the order of seconds or milliseconds
Analysis	Complex analytics	Simple response functions, aggregates, and rolling metrics

This is where AWS Kinesis Streams comes into picture. AWS Kinesis is a serverless streaming service that can be used to build highly scalable event driven applications. It makes capturing, storing and processing data much easier. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon

Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications. Amazon Kinesis enables you to process and analyze data as it arrives and respond instantly instead of having to wait until all your data is collected before the processing can begin. Some of the advantages of using Kinesis Streams are:

Scalable

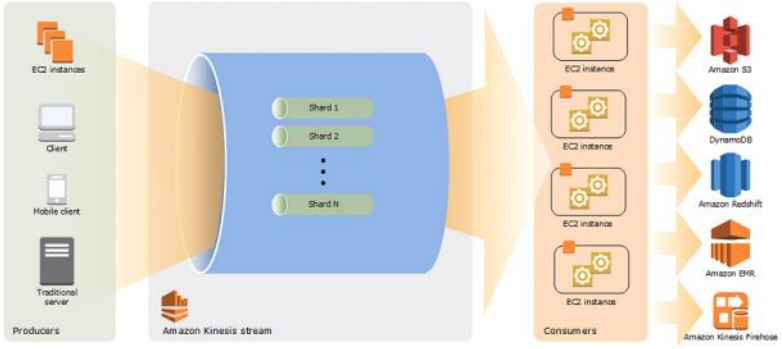
Amazon Kinesis can handle any amount of streaming data and process data from hundreds of thousands of sources with very low latencies.

Fully managed

Amazon Kinesis is fully managed and runs your streaming applications without requiring you to manage any infrastructure.

Real-time

Amazon Kinesis enables you to ingest, buffer, and process streaming data in real-time, so you can derive insights in seconds or minutes instead of hours or days.



Key Concepts

Kinesis Data Stream

A Kinesis data stream is a set of shards. Each shard has a sequence of data records. Each data record has a sequence number that is assigned by Kinesis Data Streams.

Data Record

A data record is the unit of data stored in a Kinesis data stream. Data records are composed of a sequence number, a partition key, and a data blob, which is an immutable sequence of bytes. Kinesis Data Streams does not inspect, interpret, or change the data in the blob in any way. A data blob can be up to 1 MB.

Capacity Mode

A data stream capacity mode determines how capacity is managed and how you are charged for the usage of your data stream. Currenly, in Kinesis Data Streams, you can choose between an **on-demand** mode and a **provisioned** mode for your data streams.

With the **on-demand** mode, Kinesis Data Streams automatically manages the shards in order to provide the necessary throughput. You are charged only for the actual throughput that you use and Kinesis Data Streams automatically accommodates your workloads' throughput needs as they ramp up or down.

With the **provisioned** mode, you must specify the number of shards for the data stream. The total capacity of a data stream is the sum of the capacities of its shards. You can increase or decrease the number of shards in a data stream as needed and you are charged for the number of shards at an hourly rate.

Retention Period

The retention period is the length of time that data records are accessible after they are added to the stream. A stream's retention period is set to a default of 24 hours after creation. You can increase the retention period up to 8760 hours (365 days) using the IncreaseStreamRetentionPeriod operation, and decrease the retention period down to a minimum of 24 hours using the DecreaseStreamRetentionPeriod operation. Additional charges apply for streams with a retention period set to more than 24 hours. For more information, see Amazon Kinesis Data Streams Pricing.

Producer

Producers put records into Amazon Kinesis Data Streams. For example, a web server sending log data to a stream is a producer.

Consumer

Consumers get records from Amazon Kinesis Data Streams and process them. These consumers are known as Amazon Kinesis Data Streams

Application.
Fan-Out Consumers

a) **shared fan-out consumers**: If multiple consumers are listening to same Stream, then the total bandwidth available is divided among all the consumers. For eg, a Shard gives 2MBps read bandwidth. If there are 10 shards being used for a Stream, then total available bandwidth will be 20mbps. If there are 4 Consumers, then each will get a maximum of 5 mbps only.

b) **enhanced fan-out consumers**: If multiple consumers are listening to same Stream, then the total bandwidth available per consumer does not gets affected. For eg, a Shard gives 2MBps read bandwidth. If there are 10 shards being used for a Stream, then total available bandwidth will be 20mbps. If there are 'n' Consumers, each will get a bandwidth of 20mbps. (This does incur additional charge, as expected)
The output of a Kinesis Data Streams application can be input for another stream, enabling you to create complex topologies that process data in real time. An application can also send data to a variety of other AWS services. There can be multiple applications for one stream, and each application can consume data from the stream independently and concurrently.

Shard
A shard is a uniquely identified sequence of data records in a stream. A stream is composed of one or more shards, each of which provides a fixed unit of capacity. Each shard can support up to 5 transactions per second for reads, up to a maximum total data read rate of 2 MB per second and up to 1,000 records per second for writes, up to a maximum total data write rate of 1 MB per second (including partition keys). The data capacity of your stream is a function of the number of shards that you specify for the stream. The total capacity of the stream is the sum of the capacities of its shards.
If your data rate increases, you can increase or decrease the number of shards allocated to your stream.

Partition Key
A partition key is used to group data by shard within a stream. Kinesis Data Streams segregates the data records belonging to a stream into multiple shards. It uses the partition key that is associated with each data record to determine which shard a given data record belongs to. Partition keys are Unicode strings, with a maximum length limit of 256 characters for each key. An MD5 hash function is used to map partition keys to 128-bit integer values and to map associated data records to shards using the hash key ranges of the shards. When an application puts data into a stream, it must specify a partition key. A record with a particular Partition Key goes to a specific Shard throughout the life of the Shard. This is how order of records are maintained in the Shard. AWS guarantees the order of consuming to always be the same in which the records were inserted.

Sequence Number
Each data record has a sequence number that is unique per partition-key within its shard. Kinesis Data Streams assigns the sequence number after you write to the stream with `client.putRecords` or `client.putRecord`. Sequence numbers for the same partition key generally increase over time. The longer the time period between write requests, the larger the sequence numbers become.

Shard States
A shard can have following states in its lifetime:
a) OPEN: Default state of a shard once it is included in service. this state means records can be added to it and read from it.
b) CLOSED: This means that data records are no longer added to the shard. Data records that would have been added to this shard are now added to a child shard instead. However, data records can still be retrieved from the shard for a limited time.
c) EXPIRED: After the stream's retention period has expired, all the data records in the parent shard have expired and are no longer accessible. At this point, the shard itself transitions to an `EXPIRED` state. Calls to `getStreamDescription().getShards` to enumerate the shards in the stream do not include `EXPIRED` shards in the list shards returned.

Resharding
Let's assume you have an application writing to a Stream with 2 shards which was enough for your needs. Now the user base has increased and the 2 shards are not enough to cater to all the records inducing latency in your system, you decide to increase the number of shards to 5. AWS doesn't allow you to simply add 3 more shards to existing stream because the messages are being dispensed to current shards on the basis of your partition key and as we discussed above, a record with a partition key always goes to same shard throughout its life. Now we want to divide those messages to 5 shards instead of 2, so the only way is to close the current shards and make 5 new shards. This process is called Re-Sharding. Resharding will allow any new record to now go to the new shards but previous shards will still be in CLOSED status(available for reading records but closed to insert new records). Same is also applicable if you want to decrease the number of shards for any reasons. So, resharding is the activity of either reducing or increasing number of shards by closing existing ones and creating new ones.

Batching
Batching refers to performing a single action on multiple items instead of repeatedly performing the action on each individual item. In this context,

the “item” is a record, and the action is sending it to Kinesis Data Streams. In a non-batching situation, you would place each record in a separate Kinesis Data Streams record and make one HTTP request to send it to Kinesis Data Streams. With batching, each HTTP request can carry multiple records instead of just one.

Aggregation

Aggregation refers to the storage of multiple records in a Kinesis Data Streams record. Aggregation allows customers to increase the number of records sent per API call, which effectively increases producer throughput.

Kinesis Data Streams shards support up to 1,000 Kinesis Data Streams records per second, or 1 MB throughput. The Kinesis Data Streams records per second limit binds customers with records smaller than 1 KB. Record aggregation allows customers to combine multiple records into a single Kinesis Data Streams record. This allows customers to improve their per shard throughput.

Consider the case of one shard in region us-east-1 that is currently running at a constant rate of 1,000 records per second, with records that are 512 bytes each. With KPL aggregation, you can pack 1,000 records into only 10 Kinesis Data Streams records, reducing the RPS to 10 (at 50 KB each).

Collection

Collection refers to batching multiple Kinesis Data Streams records and sending them in a single HTTP request with a call to the API operation `PutRecords`, instead of sending each Kinesis Data Streams record in its own HTTP request.

This increases throughput compared to using no collection because it reduces the overhead of making many separate HTTP requests. In fact, `PutRecords` itself was specifically designed for this purpose.

Collection differs from aggregation in that it is working with groups of Kinesis Data Streams records. The Kinesis Data Streams records being collected can still contain multiple records from the user.

Accessing the Stream

Now that we are familiar with the terminology used, let’s dive into the ways to access the stream to either write/read records or to Manage the stream.

AWS Kinesis Agent

Kinesis Agent is a stand-alone Java software application that offers an easy way to collect and send data to Kinesis Data Streams. 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.

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

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

The agent can pre process the data single line at a time or multi line as well by enabling `dataProcessingOptions` configuration setting. It works with many common log formats like **Apache Common Log**, **Apache Combined Log**, **Apache Error Log** and **RFC3164 Syslog.**

Read here to get details about how to install and configure Kinesis Agent.

Write Records — Kinesis Streams API with AWS SDK

AWS SDK provides few APIs to manage and insert data records. Most notable being *PutRecord* and *PutRecords.* 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. A sample code below for *PutRecords*

```
AmazonKinesisClientBuilder clientBuilder = AmazonKinesisClientBuilder.standard();

clientBuilder.setRegion(regionName);

clientBuilder.setCredentials(credentialsProvider);

clientBuilder.setClientConfiguration(config);

AmazonKinesis kinesisClient = clientBuilder.build();
```

```
PutRecordsRequest putRecordsRequest  = new PutRecordsRequest();

putRecordsRequest.setStreamName(streamName);

List <PutRecordsRequestEntry> putRecordsRequestEntryList  = new ArrayList<>();

for (int i = 0; i < 100; i++) {

PutRecordsRequestEntry putRecordsRequestEntry  = new PutRecordsRequestEntry();

putRecordsRequestEntry.setData(ByteBuffer.wrap(String.valueOf(i).getBytes()));

putRecordsRequestEntry.setPartitionKey(String.format("partitionKey-%d", i));

putRecordsRequestEntryList.add(putRecordsRequestEntry);

}putRecordsRequest.setRecords(putRecordsRequestEntryList);

PutRecordsResult putRecordsResult  = kinesisClient.putRecords(putRecordsRequest);

System.out.println("Put Result" + putRecordsResult);
```

A sample code for *PutRecord*

```
AmazonKinesisClientBuilder clientBuilder = AmazonKinesisClientBuilder.standard();




clientBuilder.setRegion(regionName);

clientBuilder.setCredentials(credentialsProvider);

clientBuilder.setClientConfiguration(config);




AmazonKinesis kinesisClient = clientBuilder.build();

for (int j = 0; j < 10; j++)

{

PutRecordRequest putRecordRequest = new PutRecordRequest();

putRecordRequest.setStreamName( myStreamName );

putRecordRequest.setData(ByteBuffer.wrap( String.format( "testData-%d", j ).getBytes() ));

putRecordRequest.setPartitionKey( String.format( "partitionKey-%d", j/5 ));
```

```
putRecordRequest.setSequenceNumberForOrdering( sequenceNumberOfPreviousRecord );

PutRecordResult putRecordResult = client.putRecord( putRecordRequest );

sequenceNumberOfPreviousRecord = putRecordResult.getSequenceNumber();

}
```

The disadvantage of the above is that the developer has to write a lot of boilerplate code to fetch messages, insert messages, handle batching, handle errors or exceptions, resharding, check pointing, decoding, managing order of messages, keep track of how many records have been read from the queue and to make sure there is no missed messages or duplicates in case of any events shutting down server(s). Around 90% of the code which will be needed for the above is going to remain the same for all applications using Kinesis Streams.
To avoid common pitfalls writing above common code, AWS provides Kinesis libraries where this boilerplate code is already written and user is tasked to just write code necessary to process these records once received from the Stream. the offerings are called Kinesis Consumer Library (KCL) and Kinesis Producer Library(KPL).

The retention period is the length of time that data records are accessible after they are added to the stream. A stream's retention period is set to a default of 24 hours after creation. You can increase the retention period up to 8760 hours (365 days) using the IncreaseStreamRetentionPeriod operation, and decrease the retention period down to a minimum of 24 hours using the DecreaseStreamRetentionPeriod operation. Additional charges apply for streams with a retention period set to more than 24 hours. For more information, see Amazon Kinesis Data Streams Pricing.

Producer
Producers put records into Amazon Kinesis Data Streams. For example, a web server sending log data to a stream is a producer.

Consumer
Consumers get records from Amazon Kinesis Data Streams and process them. These consumers are known as Amazon Kinesis Data Streams Application.
Fan-Out Consumers