BIG DATA on AWS

Data Collection

Agenda

01 What is AWS Kinesis?

02 Why AWS Kinesis?

03 How does AWS Kinesis Work?

04 Features of AWS Kinesis

05 AWS Kinesis Components

06 Enhanced Fan Out

07 Kinesis Firehose

What is AWS Kinesis?

What is AWS Kinesis?

AWS Kinesis is a cloud-based service which enables you to **collect, process** and **analyze** real-time, streaming data

Streaming data is data which is generated continuously from thousands of data sources, and these data sources can send the data records simultaneously and in small sizes

What is AWS Kinesis?

Some Kinesis Use Cases:

- Real-Time Data Log and Data Entry Ingestion
- Real-Time Graph/Metrics availability for Monitoring Kinesis Stream Statistics
- Real-Time Data Analytics

amazon Kinesis

Real-Time Results that can be used for timely insights and quick reactions/responses

How does AWS Kinesis Work?

How does AWS Kinesis Work?

Before we understand the working of Kinesis, we need to understand some key terminologies

Shard

Record

How does AWS Kinesis Work?

Shard

Record

Shard is the base throughput unit of an Amazon Kinesis data stream. One shard provides a capacity of 1MB/sec data input and 2MB/sec data output. One shard can support up to 1000 PUT records per second. You will specify the number of shards needed when you create a data stream.

Shard ID is the unique identifier of the shard within the stream

Shard Iterator iterates on all the records of a Kinesis Shard

How does AWS Kinesis Work?

Shard

Record

Record is a unit of data contained in a Kinesis Stream

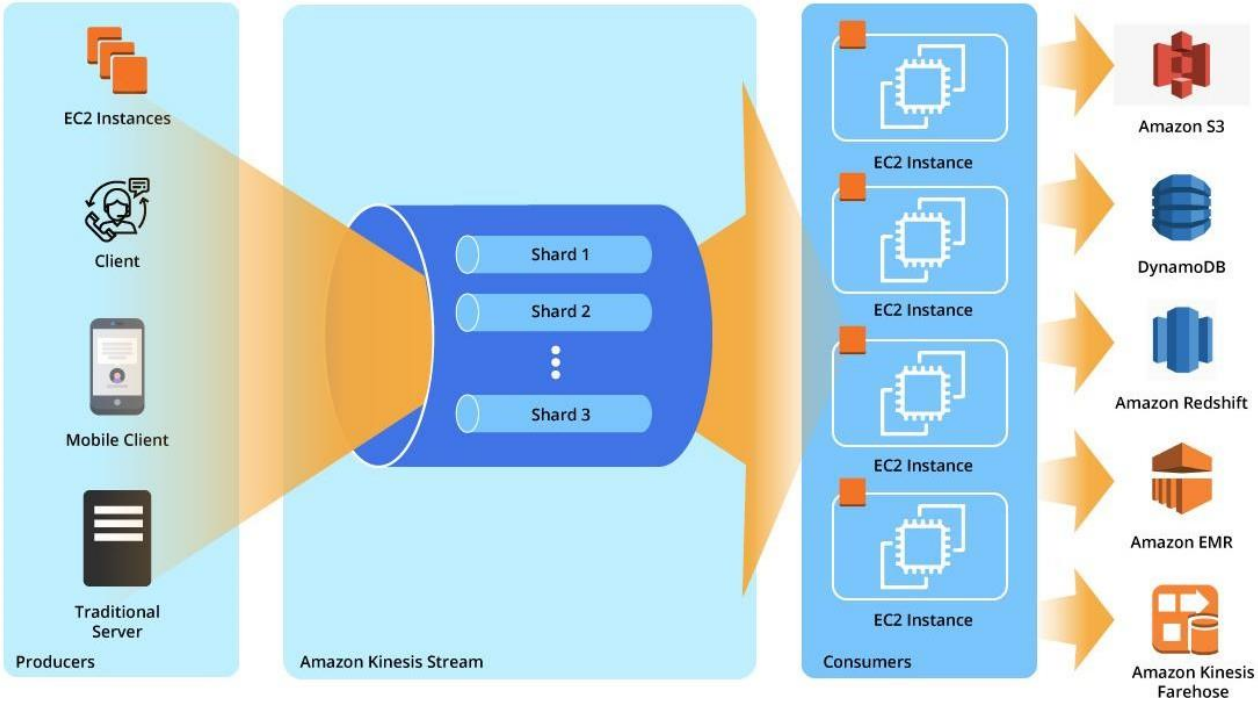
Partition key is used to segregate and route records to different shards of a data stream. A partition key is specified by your data producer while adding data to an Amazon Kinesis data stream

Data Blob is the actual data contained in the record

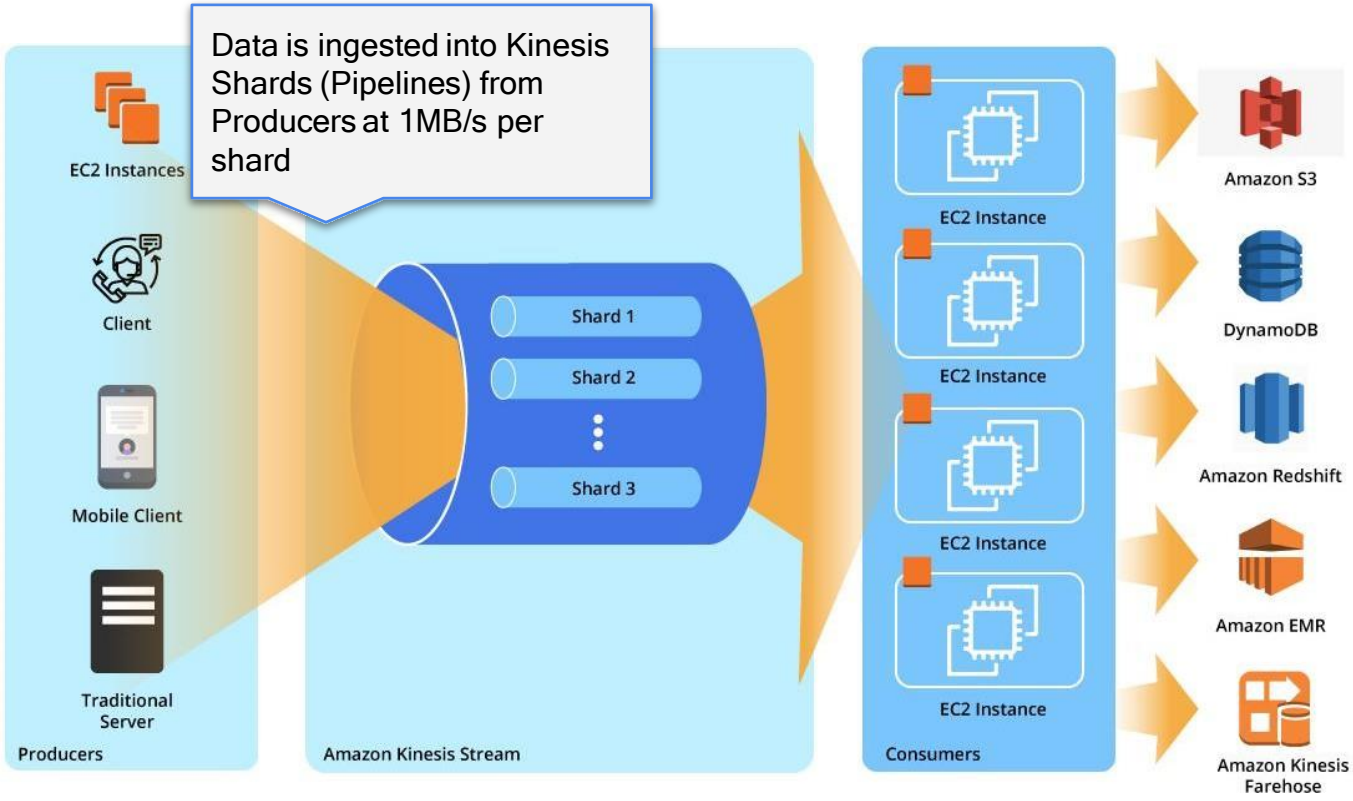
A **sequence number** is a unique identifier for each record

How does AWS Kinesis Work?

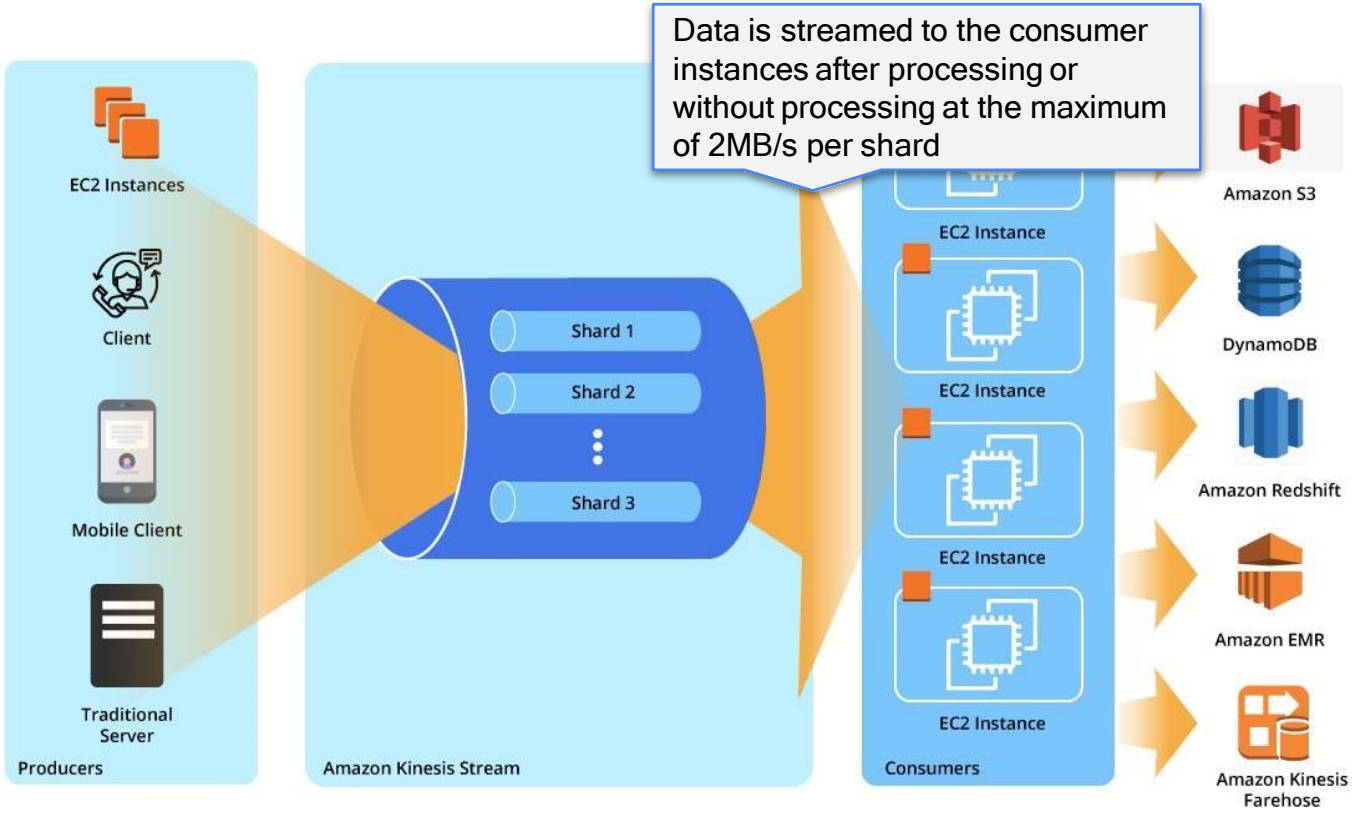
Kinesis Architecture



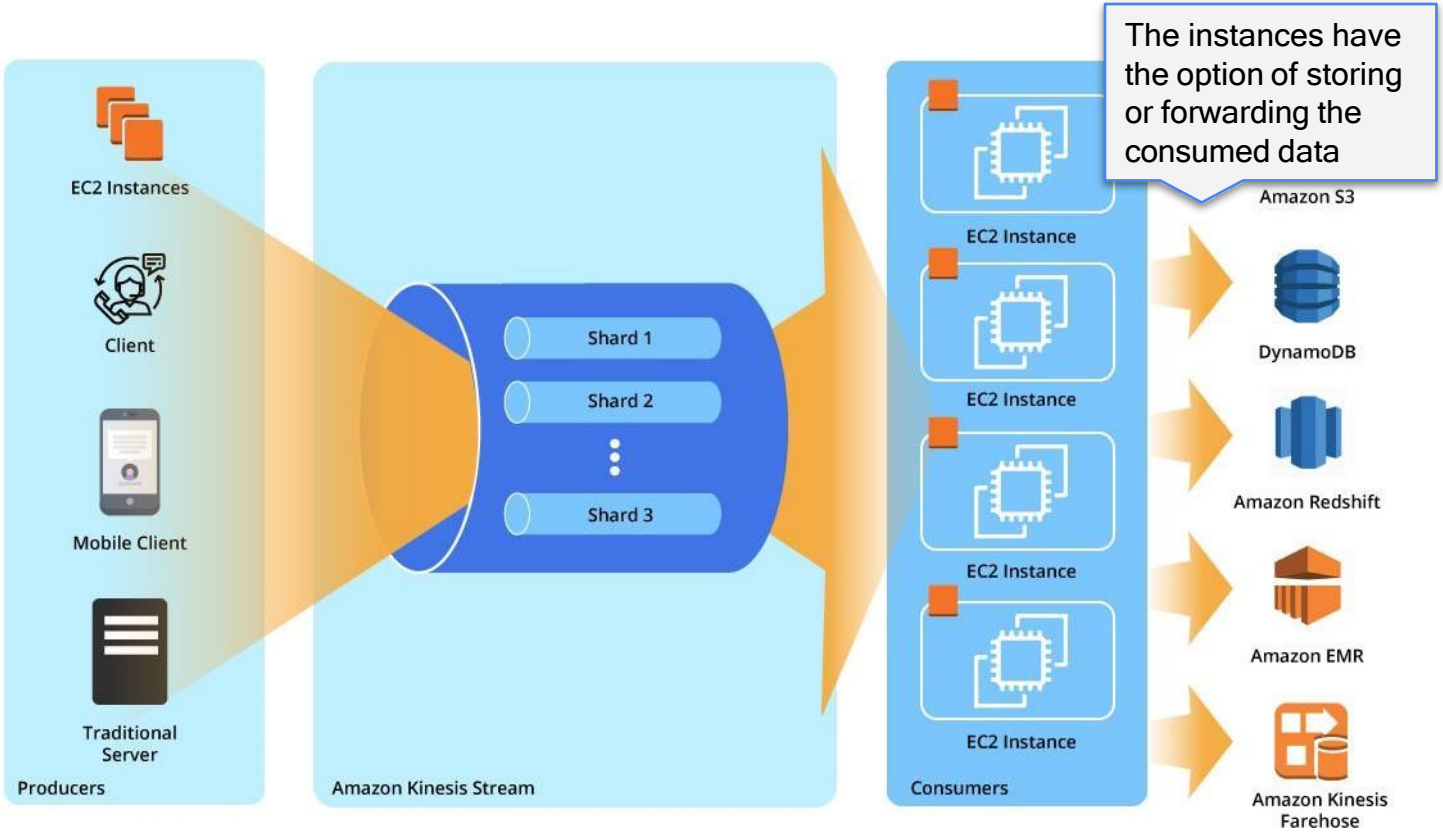
How does AWS Kinesis Work?



How does AWS Kinesis Work?



How does AWS Kinesis Work?



Hands-On
(Managing Kinesis Shards)

Manage a Kinesis Shard

- Create a Stream
- Describe the Stream
- List all the streams that you've created
- Add data to the stream (Put a Record)
- Get the Shard Iterator
- Use the Shard Iterator to get the Records (Valid for only 300 seconds post creating a Shard Iterator)
- Delete a Stream

Why AWS Kinesis?

Why AWS Kinesis?

Why is **Kinesis** better than Apache Kafka?

Amazon
Lambda

- In Kinesis, you pay according to your usage and you don't have to rent out or buy entire hardware resources irrespective of usage.
- Kafka gives you more flexibility, but you must dedicate an entire DevOps team to manage and maintain it.
- Kinesis manages and maintains things for you as it is a paid service provided to you by Amazon.
- We can also analyze/manipulate Kinesis data using lambda functions before forwarding it to S3 or RedShift.

Why AWS Kinesis?

There are certain key advantages of Kinesis being an AWS Service

Cloud-Based

Divide and Conquer

Fast

Managed

Scalable

Why AWS Kinesis?

- **Cloud-Based**
- **Divide and Conquer**
- **Fast**
- **Managed**
- **Scalable**

AWS provides cloud-based services to customers. This eliminates the need to house/own computational hardware locally

Why AWS Kinesis?

- Cloud-Based
- Divide and Conquer
- Fast
- Managed
- Scalable

Data Collection is separate from Data Processing making both the processes independent

Why AWS Kinesis?

- Cloud-Based
- Divide and Conquer
- **Fast**
- Managed
- Scalable

1 Kinesis Shard provides 1MB/s data ingestion speed and 2MB/s data output speed making it extremely efficient for real-time streaming

Why AWS Kinesis?

- Cloud-Based
- Divide and Conquer
- Fast
- **Managed**
- Scalable

All AWS Services are fully managed by Amazon and the user only needs to concern himself with the task at hand and no maintenance

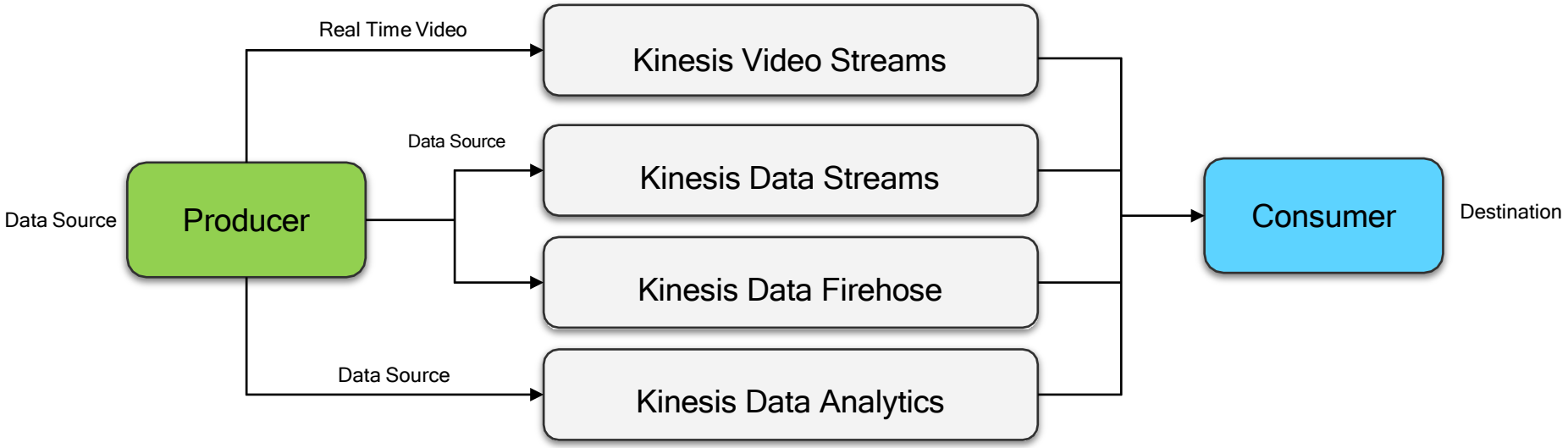
Why AWS Kinesis?

- Cloud-Based
- Divide and Conquer
- Fast
- Managed
-

You can increase or decrease the scale of your streaming operation depending on your use case

Features of AWS Kinesis

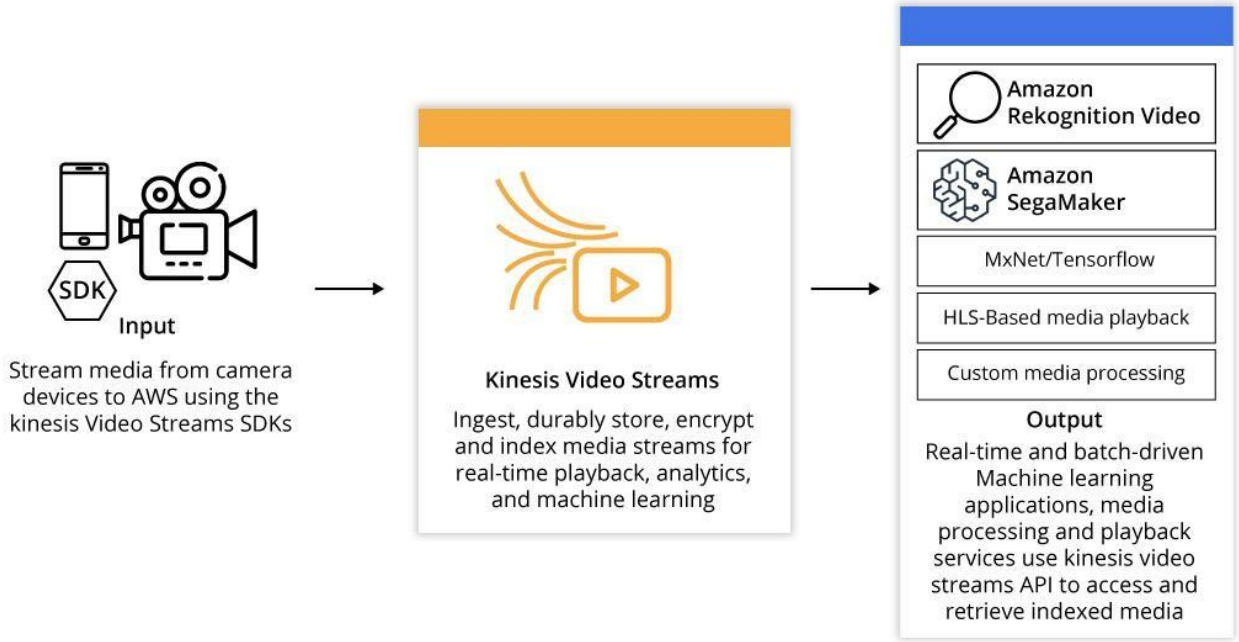
Features of AWS Kinesis



Kinesis provides four features. The user can deploy the service which is the most efficient for his use case

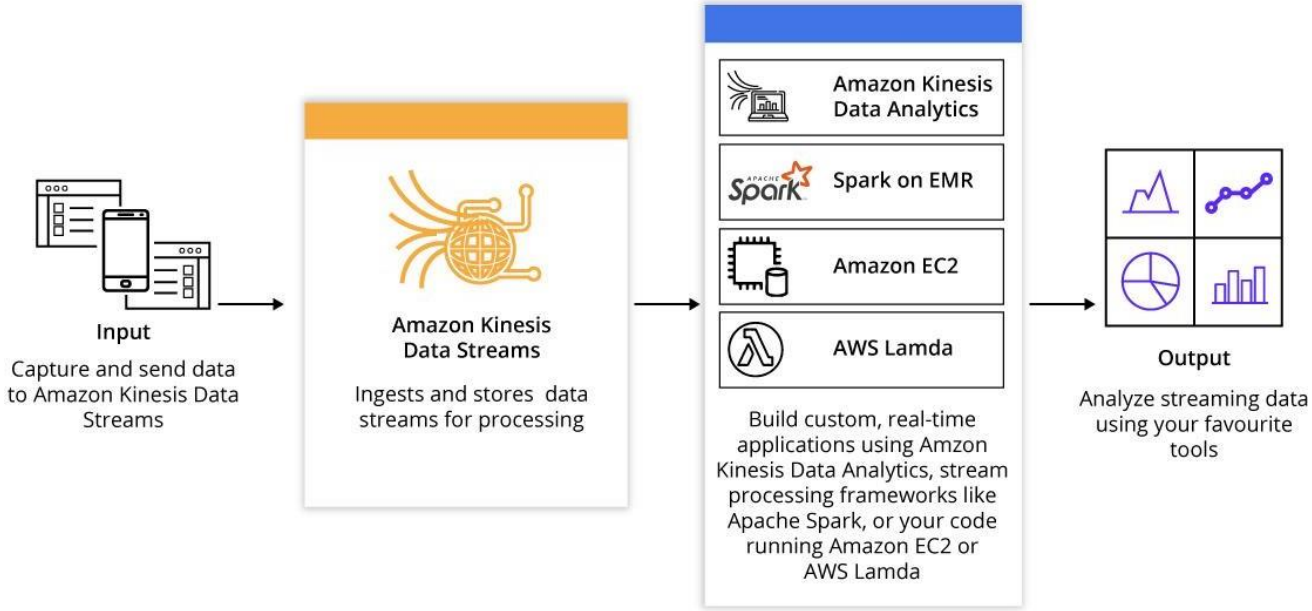
Features of AWS Kinesis

Kinesis Video Streaming Architecture and Use Case



Features of AWS Kinesis

Kinesis Data Streaming Architecture and Use Case

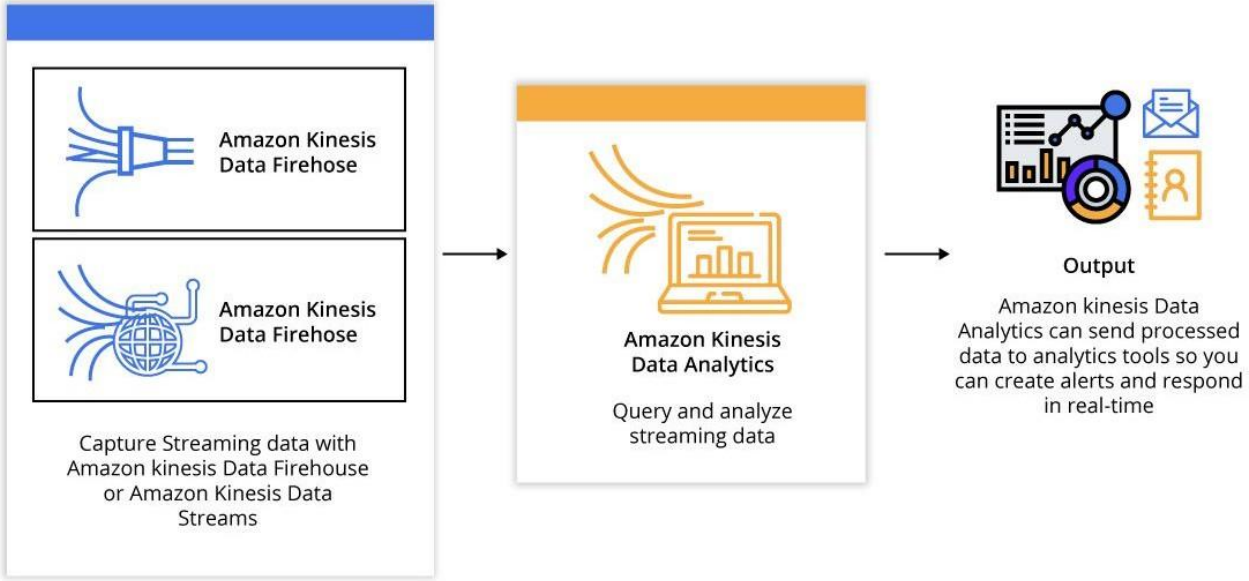


Features of AWS Kinesis

Kinesis Data Firehose Architecture and Use Case

Features of AWS Kinesis

Kinesis Data Analysis Architecture and Use Case



AWS Kinesis Components

AWS Kinesis Components

Producer

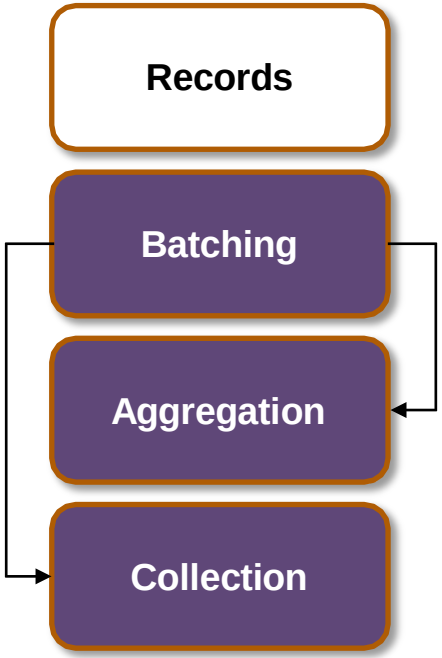
An Amazon Kinesis Data Streams **producer** is an application that puts user data records into a Kinesis data stream (also called data ingestion).

The **Kinesis Producer Library (KPL)** simplifies producer application development, allowing developers to achieve high write throughput to a Kinesis data stream.

- Writes to one or more Kinesis data 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 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

AWS Kinesis Components

KPL Terminology

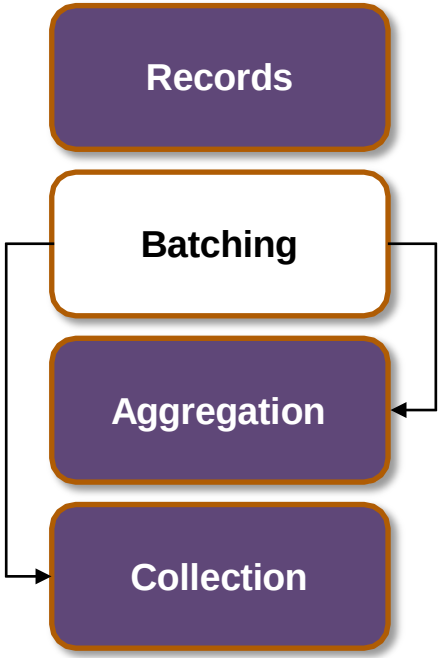


There is a distinction between **KPL user records** and **Kinesis Streams records**.

A KPL user record alludes to a blob of data that has meaning to the user. For example, a JSON blob representing events on a web server, or log files from an application

AWS Kinesis Components

KPL Terminology



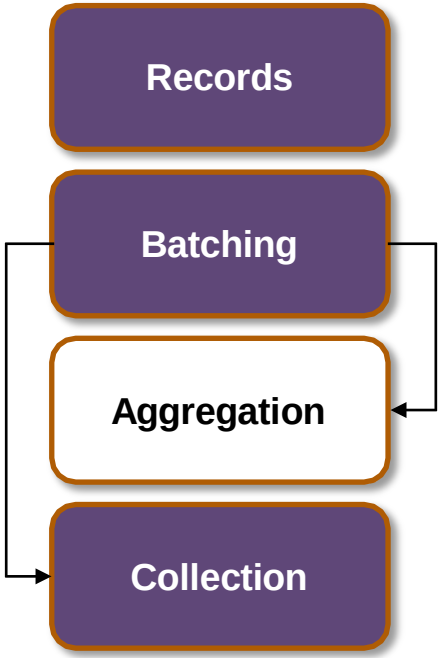
Batching means performing a single operation on a collection of objects or items (a batch) rather than performing the same operation on multiple items individually

In KPL context, batching implies **sending records to the Kinesis Stream**. There are two types of batching:

- Aggregation
- Collection

AWS Kinesis Components

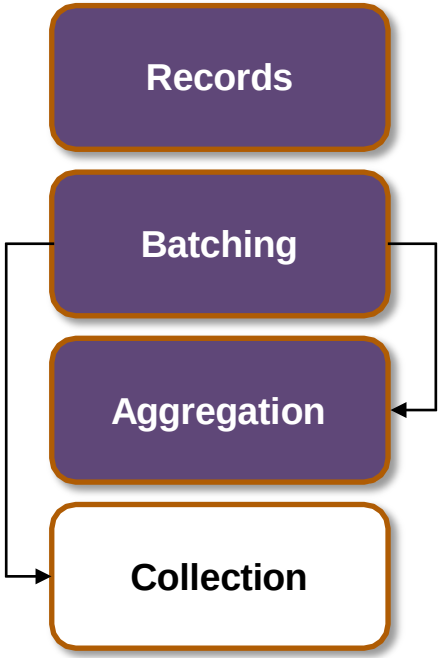
KPL Terminology



Aggregation is storing multiple records within a single Kinesis Data Streams record

AWS Kinesis Components

KPL Terminology



Collection is using the API operation: PutRecords to send multiple Kinesis Data Streams records to one or more shards in your Kinesis data stream

AWS Kinesis Components

A Basic **Producer** Application in Java:

```
// KinesisProducer gets credentials automatically like
// DefaultAWSCredentialsProviderChain.
// It also gets region automatically from the EC2 metadata
service. KinesisProducer kinesis = new KinesisProducer();
// Put some records
for (int i = 0; i < 100; ++i) {
    ByteBuffer data = ByteBuffer.wrap("myData".getBytes("UTF-8"));
    // doesn't block
    kinesis.addUserRecord("myStream", "myPartitionKey", data);
}
// Perform other functions ...
```

AWS Kinesis Components

Consumer

A **consumer**, known as an Amazon Kinesis Data Streams application, is an application that you build to read and process data records from Kinesis data streams.

You can develop a consumer application for Amazon Kinesis Data Streams using the Kinesis Client Library (KCL). The Kinesis Client Library (KCL) helps you consume and process data from a Kinesis data stream.

- Connects to the stream
- Enumerates the shard
- Coordinates shard associations with other workers (if any)
- Instantiates a record processor for every shard it manages
- Pulls data records from the stream
- Pushes the records to the corresponding record processor
- Checkpoints processed records
- Balances shard-worker associations when the worker instance count changes
- Balances shard-worker associations when shards are split or merged

AWS Kinesis Components

Consumer Application Code to Get Shard Iterator for a particular Shard in Java:

```
String shardIterator;
GetShardIteratorRequest getShardIteratorRequest = new
GetShardIteratorRequest();
getShardIteratorRequest.setStreamName(myStreamName);
getShardIteratorRequest.setShardId(shard.getShardId());
getShardIteratorRequest.setShardIteratorType("TRIM_HORIZON")
;

GetShardIteratorResult getShardIteratorResult =
client.getShardIterator(getShardIteratorRequest);
shardIterator = getShardIteratorResult.getShardIterator();
```

AWS Kinesis Components

Consumer Application Code to Get Records through Shard Iterator in Java

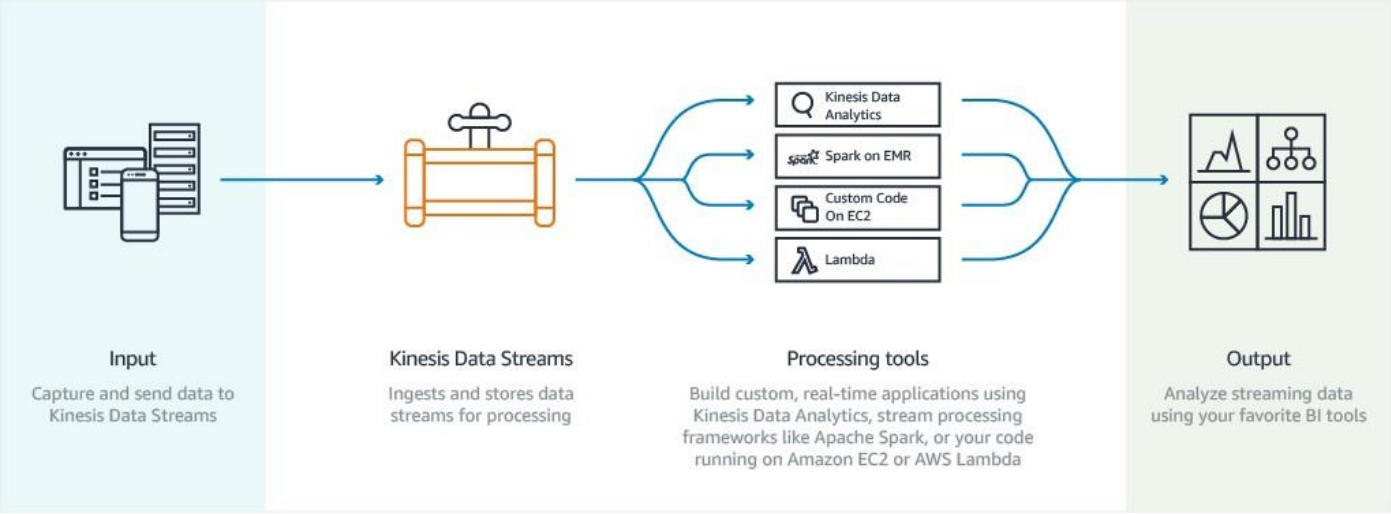
```
GetRecordsRequest getRecordsRequest = new GetRecordsRequest();
getRecordsRequest.setShardIterator(shardIterator);
getRecordsRequest.setLimit(25);

GetRecordsResult getRecordsResult =
client.getRecords(getRecordsRequest); List<Record> records =
getRecordsResult.getRecords();
```

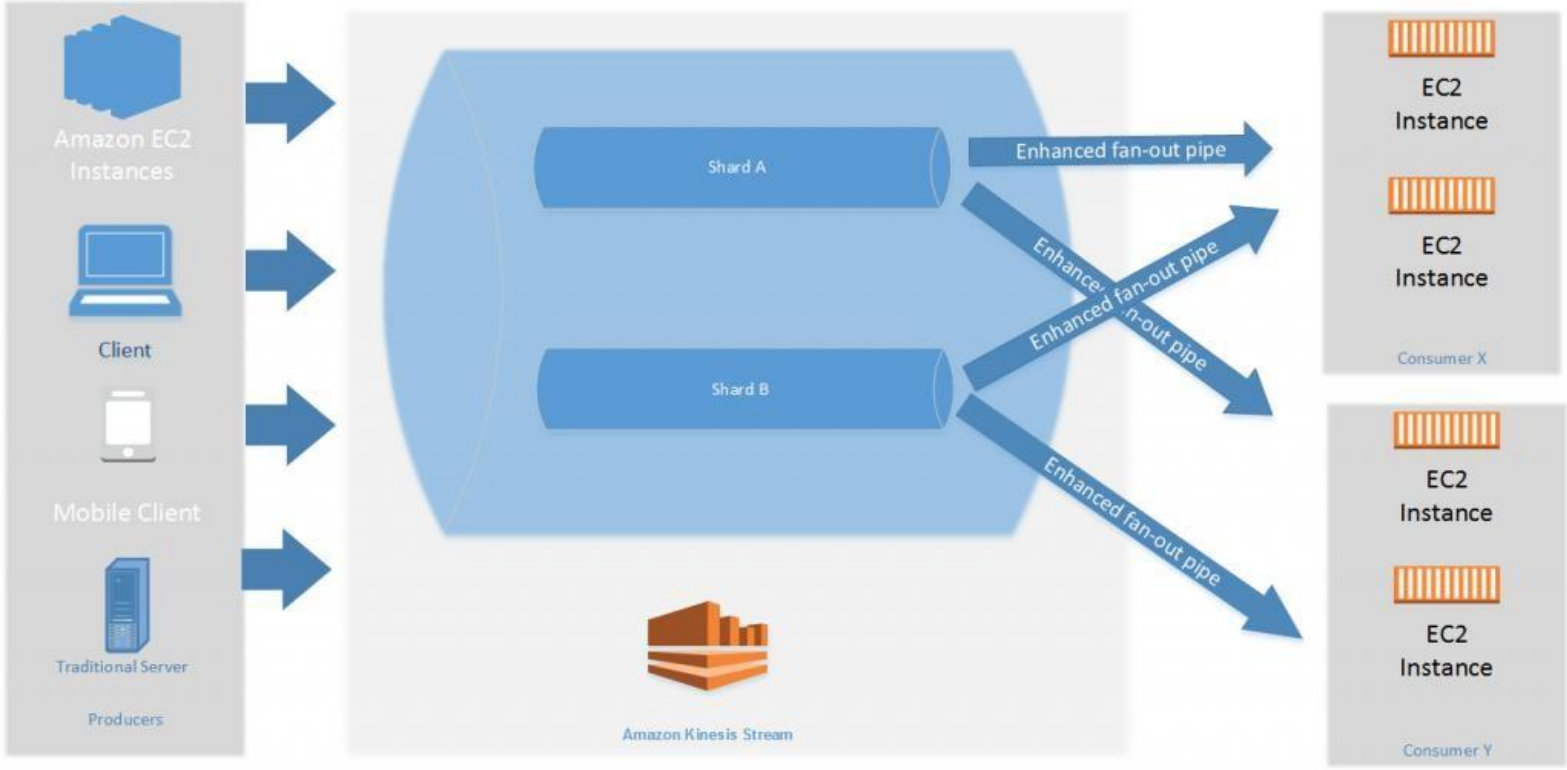
Enhanced Fan-Out

Enhanced Fan-Out

Enhanced fan-out is an optional feature for Kinesis Data Streams consumers that provides logical 2 MB/sec throughput pipes between consumers and shards. This allows customers to scale the number of consumers reading from a data stream in parallel, while maintaining high performance.



Enhanced Fan-Out



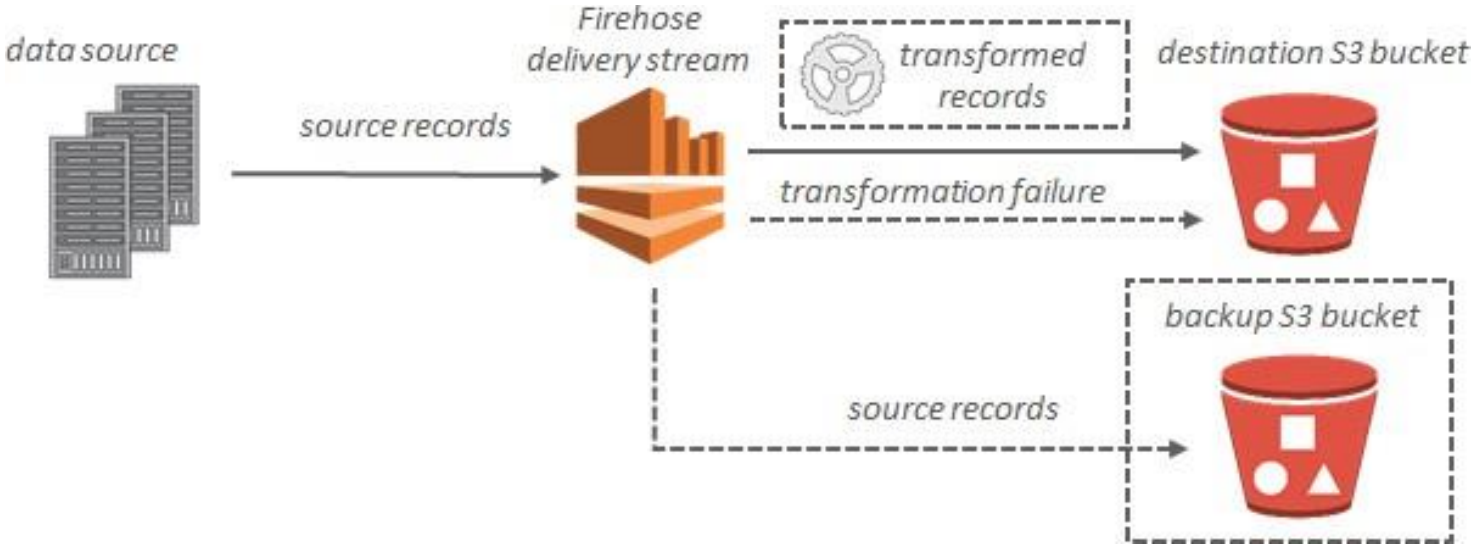
Enhanced Fan-Out Workflow

Kinesis Firehose

Kinesis Firehose

Amazon Kinesis Data Firehose is the easiest way to reliably load streaming data into data lakes, data stores and analytics tools. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards you're already using today.

Kinesis Firehose



Workflow for Firehose

+919030485102

rganesh0203@gmail.com

https://topmate.io/rganesh_0203