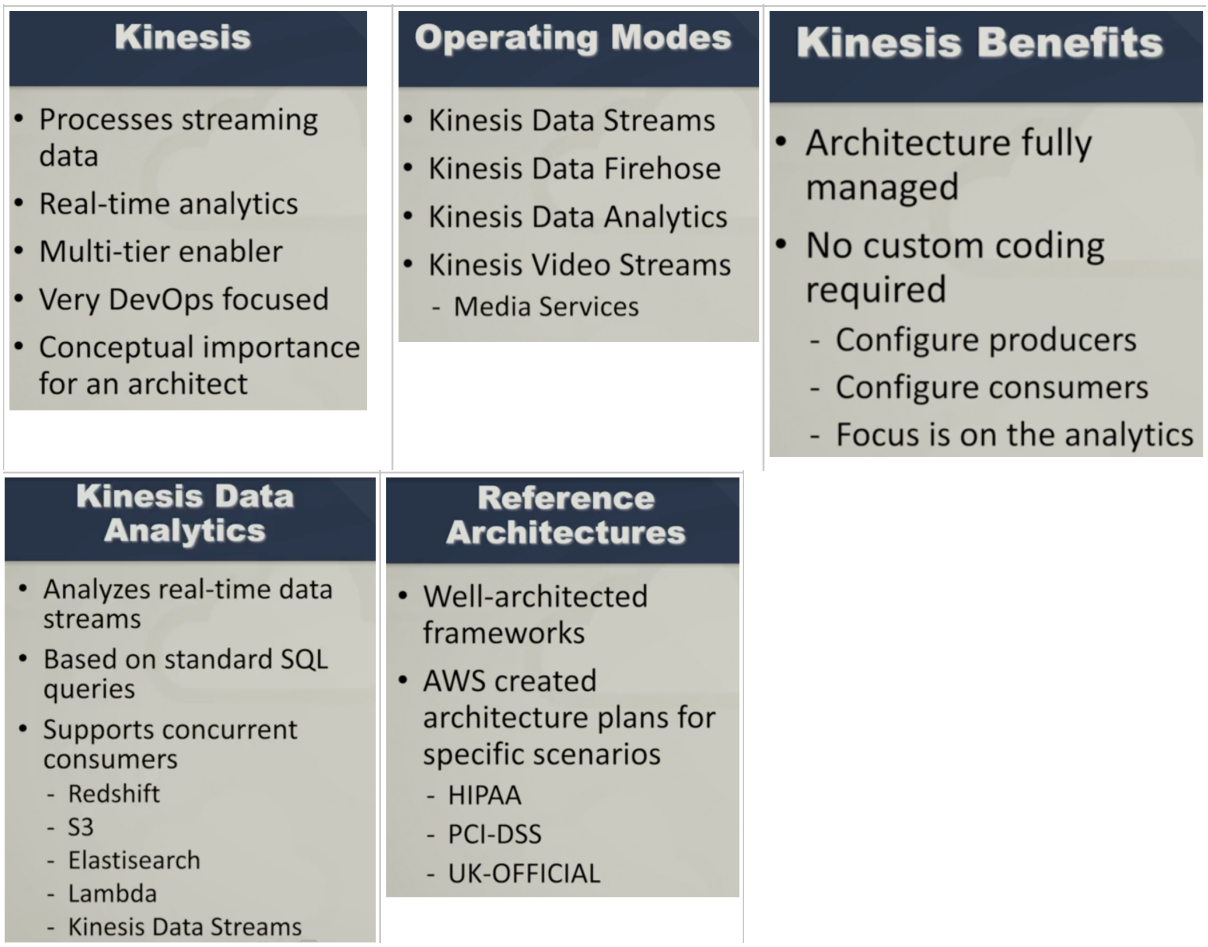
Kinesis

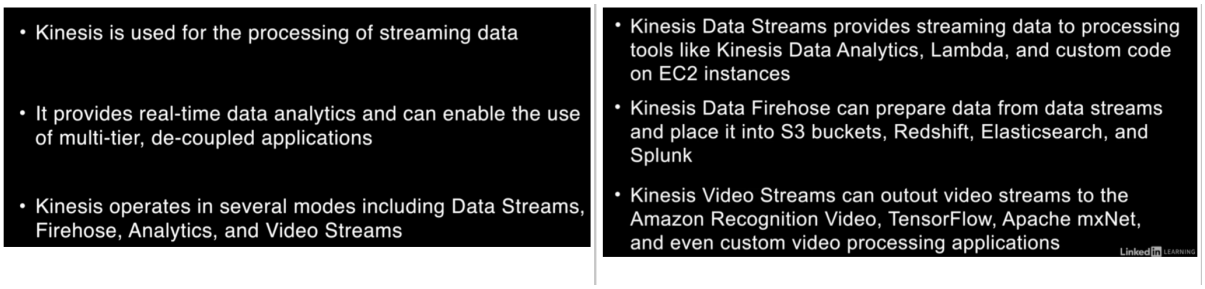


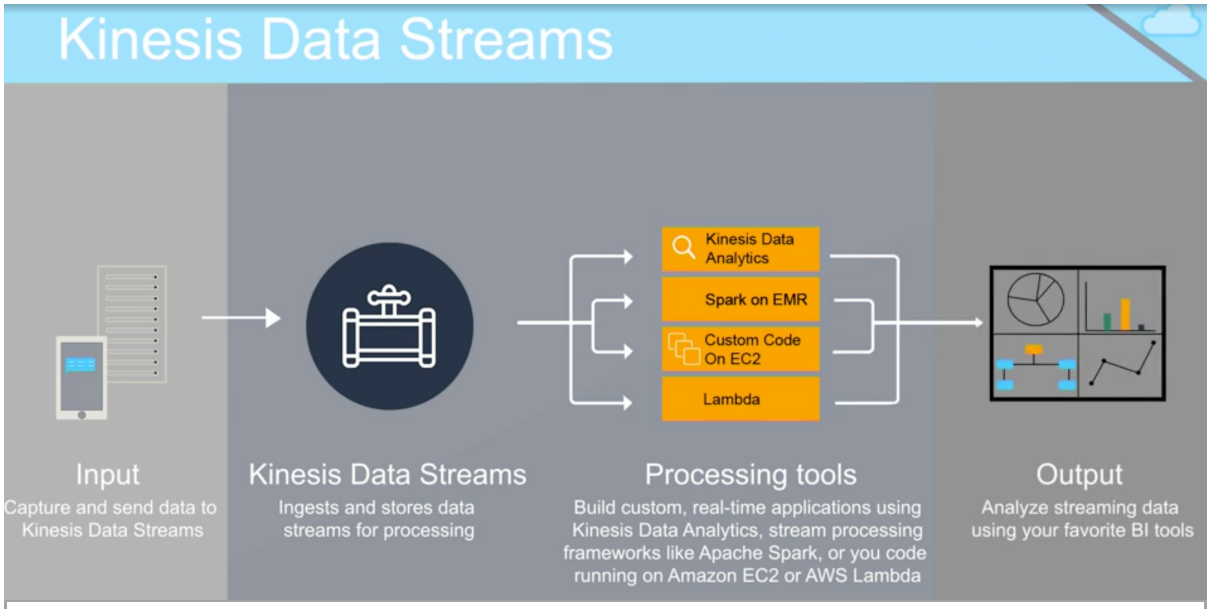
References:

https://aws.amazon.com/kinesis/data-firehose/

https://aws.amazon.com/kinesis/data-streams/faqs/

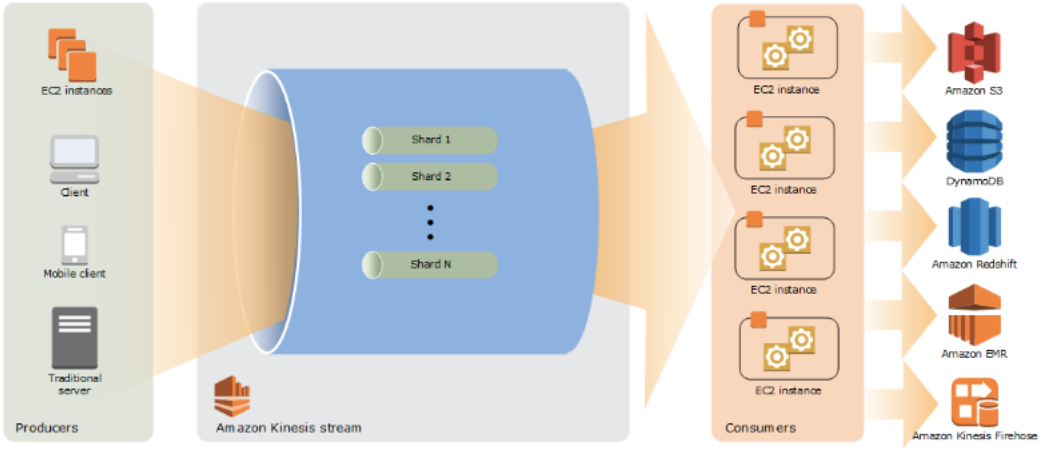
https://aws.amazon.com/blogs/compute/increasing-real-time-stream-processing-performance-with-amazon-kinesis-data-streams-enhanced-fan-out-and-aws-lambda/



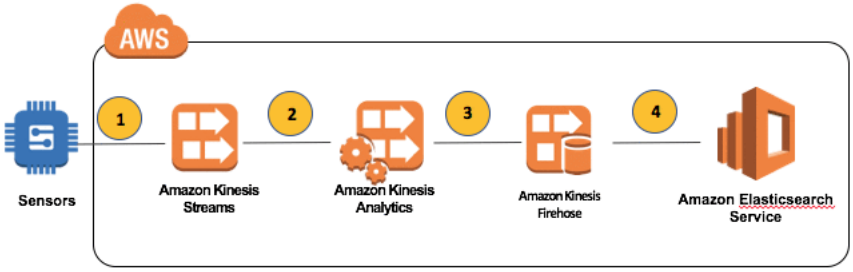


Kinesis Data Streams
Kinesis Data Analytics
Spark on EMR
Custom Code On EC2
Lambda
Input
Capture and send data to Kinesis Data Streams
Kinesis Data Streams
Ingests and stores data streams for processing
Processing tools
Build custom, real-time applications using Kinesis Data Analytics, stream processing frameworks like Apache Spark, or you code running on Amazon EC2 or AWS Lambda
Output
Analyze streaming data using your favorite BI tools

- **Amazon Kinesis Data Streams** is used to collect and process large streams of data records in real time.
- You can use Kinesis Data Streams for rapid and continuous data intake and aggregation.
- The type of data used includes IT infrastructure log data, application logs, social media, market data feeds, and web clickstream data. Because the response time for the data intake and processing is in real time, the processing is typically lightweight.
- The following diagram illustrates the high-level architecture of Kinesis Data Streams. The producers continually push data to Kinesis Data Streams, and the consumers process the data in real time. Consumers (such as a custom application running on Amazon EC2 or an Amazon Kinesis Data Firehose delivery stream) can store their results using an AWS service such as Amazon DynamoDB, Amazon Redshift, or Amazon S3.



- Kinesis Data Streams supports changes to the data record retention period of your stream. A Kinesis data stream is an ordered sequence of data records meant to be written to and read from in real-time. Data records are therefore stored in shards in your stream temporarily.



- The time period from when a record is added to when it is no longer accessible is called the *retention period*. A Kinesis data stream stores records from **24 hours by default** to a maximum of 168 hours.
- This is the reason why there are missing data in your S3 bucket. To fix this, you can either configure your sensors to send the data everyday instead of every other day or alternatively, you can increase the retention period of your Kinesis data stream.

CONCEPTS

- **Concepts**
  - **Data Producer** – An application that typically emits data records as they are generated to a Kinesis data stream. Data producers assign partition keys to records. Partition keys ultimately determine which shard ingests the data record for a data stream.
  - **Data Consumer** – A distributed Kinesis application or AWS service retrieving data from all shards in a stream as it is generated. Most data consumers are retrieving the most recent data in a shard, enabling real-time analytics or handling of data.
  - **Data Stream** – A logical grouping of shards. There are no bounds on the number of shards within a data stream. A data stream will retain data for **24 hours, or up to 7 days** when extended retention is enabled.
  - **Shard** – The base throughput unit of a Kinesis data stream.
    - A shard is an append-only log and a unit of streaming capability. A shard contains an ordered sequence of records ordered by arrival time.
    - Add or remove shards from your stream dynamically as your data throughput changes.
    - One shard can ingest up to 1000 data records per second, or 1MB/sec. Add more shards to increase your ingestion capability.
    - When consumers use **enhanced fan-out**, one shard provides 1MB/sec data input and 2MB/sec data output for each data consumer registered to use enhanced fan-out.
    - When consumers do **no**t use **enhanced fan-out**, a shard provides 1MB/sec of input and 2MB/sec of data output, and this output is shared with any consumer not using enhanced fan-out.
    - You will specify the number of shards needed when you create a stream and can change the quantity at any time.
  - **Data Record**
    - A record is the unit of data stored in a Kinesis stream. A record is composed of a sequence number, partition key, and data blob.
    - A data blob is the data of interest your data producer adds to a stream. The maximum size of a data blob is 1 MB.

- **Partition Key**
  - A partition key is typically a meaningful identifier, such as a user ID or timestamp. It is specified by your data producer while putting data into a Kinesis data stream, and useful for consumers as they can use the partition key to replay or build a history associated with the partition key.
  - The partition key is also used to segregate and route data records to different shards of a stream.
- **Sequence Number**
  - A sequence number is a unique identifier for each data record. Sequence number is assigned by Kinesis Data Streams when a data producer calls *PutRecord* or *PutRecords* API to add data to a Kinesis data stream.

- **Monitoring**
  - You can monitor shard-level metrics in Kinesis Data Streams.
  - You can monitor your data streams in Amazon Kinesis Data Streams using CloudWatch, Kinesis Agent, Kinesis libraries.
  - Log API calls with CloudTrail.

- **Security**
  - Kinesis Data Streams can automatically encrypt sensitive data as a producer enters it into a stream. Kinesis Data Streams uses AWS KMS master keys for encryption.
  - Use IAM for managing access controls.
  - You can use an interface VPC endpoint to keep traffic between your Amazon VPC and Kinesis Data Streams from leaving the Amazon network.

- **Limits**
  - There is no upper limit on the number of shards you can have in a stream or account.
  - There is no upper limit on the number of streams you can have in an account.
  - A single shard can ingest up to 1 MiB of data per second (including partition keys) or 1,000 records per second for writes.
  - The default shard limit is 500 shards for the following AWS Regions: US East (N. Virginia), US West (Oregon), and EU (Ireland). For all other Regions, the default shard limit is 200 shards.
  - **Each shard can support up to five read transactions per second.**

KINESIS DATA STREAM - RESHARDING

- Amazon Kinesis Data Streams supports *resharding*, which lets you adjust the number of shards in your stream to adapt to changes in the rate of data flow through the stream. Resharding is considered an advanced operation.
- There are two types of resharding operations:
  - shard split and
  - shard merge.
- In a shard split, you divide a single shard into two shards.
- In a shard merge, you combine two shards into a single shard.
- Resharding is always *pairwise* in the sense that you cannot split into more than two shards in a single operation, and you cannot merge more than two shards in a single operation.
- The shard or pair of shards that the resharding operation acts on are referred to as *parent* shards. The shard or pair of shards that result from the resharding operation are referred to as *child* shards.

- Splitting increases the number of shards in your stream and therefore increases the data capacity of the stream. Because you are charged on a per-shard basis, splitting increases the cost of your stream.
- Similarly, merging reduces the number of shards in your stream and therefore decreases the data capacity—and cost—of the stream.

- If your data rate increases, you can also increase the number of shards allocated to your stream to maintain the application performance.
- You can reshard your stream using the **UpdateShardCount** API. The throughput of an Amazon Kinesis data stream is designed to scale without limits via increasing the number of shards within a data stream. Hence, the correct answer is to **increase the number of shards of the Kinesis stream by using the UpdateShardCount command**.

References:

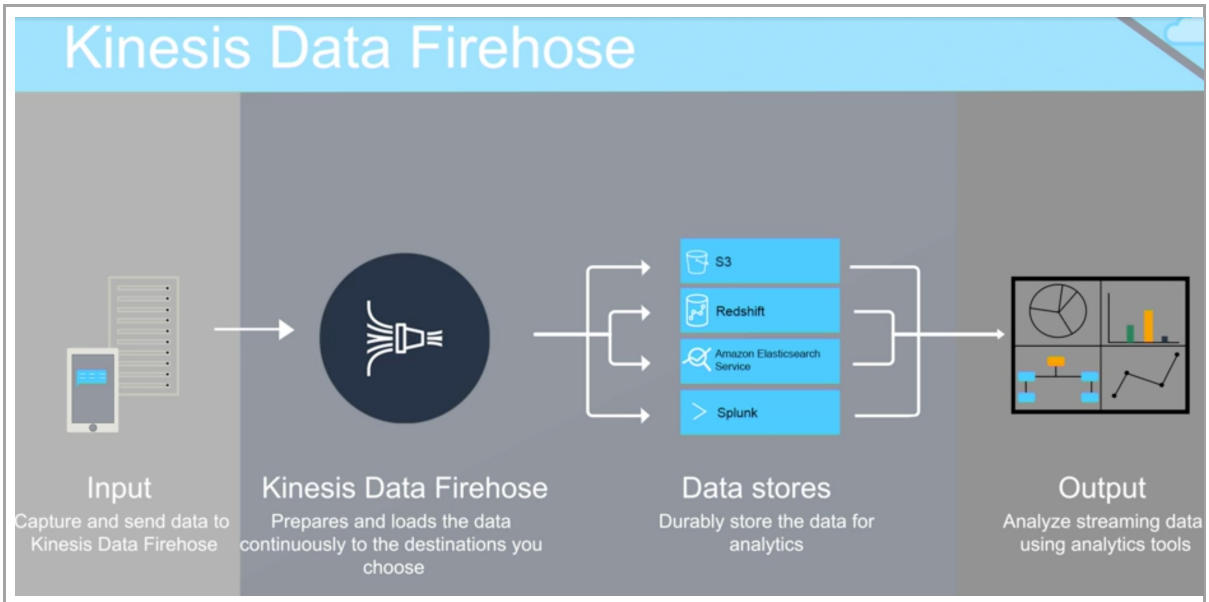
https://aws.amazon.com/blogs/big-data/scale-your-amazon-kinesis-stream-capacity-with-updateshardcount/

https://aws.amazon.com/kinesis/data-streams/faqs/

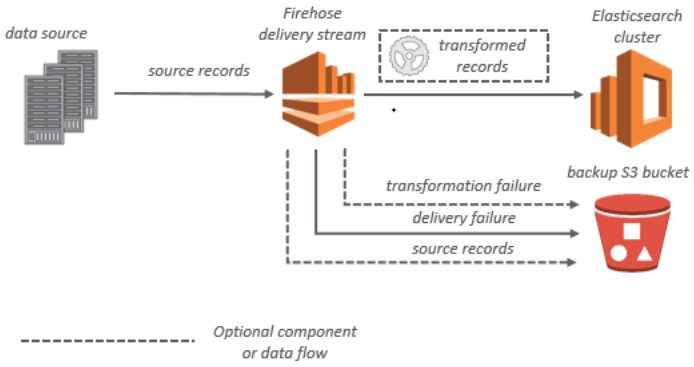
https://docs.aws.amazon.com/streams/latest/dev/kinesis-using-sdk-java-resharding.html

Check out this Amazon Kinesis Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-kinesis/



- Amazon Kinesis Data Firehose is the easiest way to load streaming data into 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.
- It is 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.

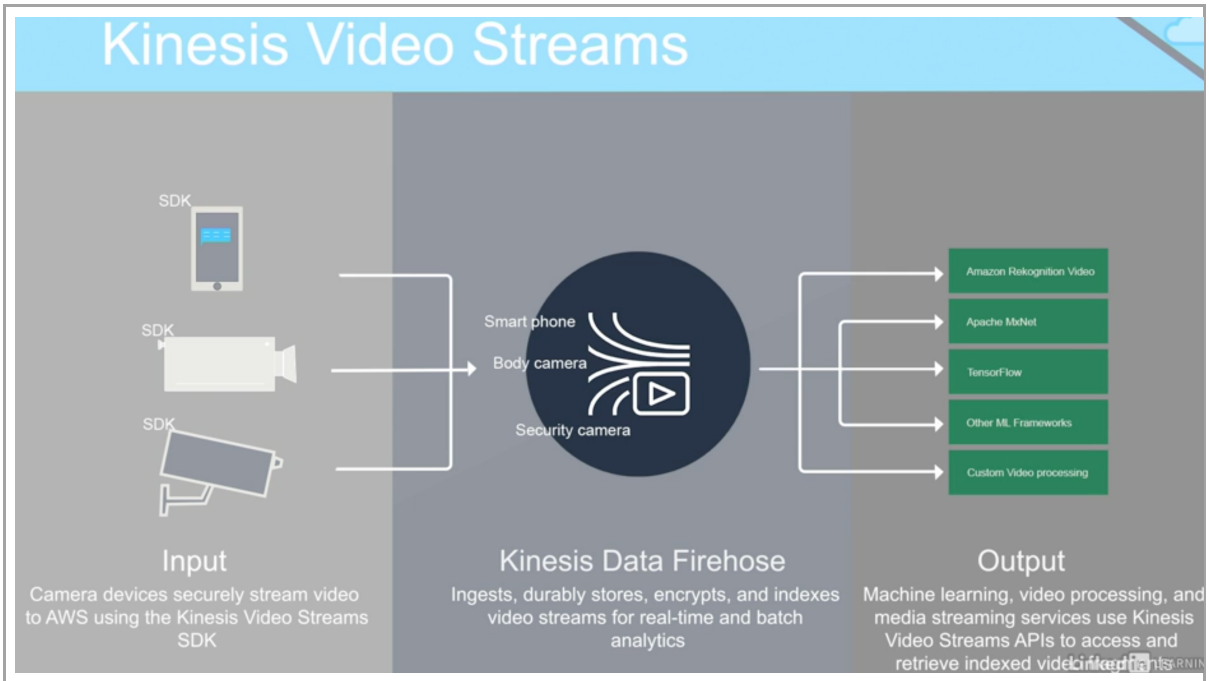


- **Concepts**
  - **Kinesis Data Firehose Delivery Stream** – The underlying entity of Kinesis Data Firehose. You use Kinesis Data Firehose by creating a Kinesis Data Firehose delivery stream and then sending data to it.
  - **Record** – The data of interest that your data producer sends to a Kinesis Data Firehose delivery stream. A record can be as large as 1,000 KB.
  - **Data Producer** – Producers send records to Kinesis Data Firehose delivery streams.
  - **Buffer Size and Buffer Interval** – Kinesis Data Firehose buffers incoming streaming data to a certain size or for a certain period of time before delivering it to destinations. Buffer Size is in MBs and Buffer Interval is in seconds.

- **Stream Sources**
  - You can send data to your Kinesis Data Firehose Delivery stream using different types of sources:
    - a Kinesis data stream,
    - the Kinesis Agent,
    - or the Kinesis Data Firehose API using the AWS SDK.
  - You can also use CloudWatch Logs, CloudWatch Events, or AWS IoT as your data source.
  - Some AWS services can only send messages and events to a Kinesis Data Firehose delivery stream that is in the same Region.

- **Data Delivery and Transformation**
  - Kinesis Data Firehose can invoke your Lambda function to transform incoming source data and deliver the transformed data to destinations.
  - Kinesis Data Firehose buffers incoming data up to 3 MB by default.
  - If your Lambda function invocation fails because of a network timeout or because you've reached the Lambda invocation limit, Kinesis Data Firehose retries the invocation three times by default.
  - Kinesis Data Firehose can convert the format of your input data from JSON to Apache Parquet or Apache ORC before storing the data in S3. Parquet and ORC are columnar data formats that save space and enable faster queries compared to row-oriented formats like JSON.
  - Data delivery format:
    - **For data delivery to S3**, Kinesis Data Firehose concatenates multiple incoming records based on buffering configuration of your delivery stream. It then delivers the records to S3 as an S3 object.
    - **For data delivery to Redshift**, Kinesis Data Firehose first delivers incoming data to your S3 bucket in the format described earlier. Kinesis Data Firehose then issues an Redshift *COPY* command to load the data from your S3 bucket to your Redshift cluster.
    - **For data delivery to ElasticSearch**, Kinesis Data Firehose buffers incoming records based on buffering configuration of your delivery stream. It then generates an Elasticsearch bulk request to index multiple records to your Elasticsearch cluster.
    - **For data delivery to Splunk**, Kinesis Data Firehose concatenates the bytes that you send.
  - Data delivery frequency
    - The frequency of data delivery to S3 is determined by the **S3 Buffer size** and **Buffer interval** value that you configured for your delivery stream.
    - The frequency of data *COPY* operations from S3 to Redshift is determined by how fast your Redshift cluster can finish the *COPY* command.
    - The frequency of data delivery to ElasticSearch is determined by the **Elasticsearch Buffer size** and **Buffer interval** values that you configured for your delivery stream.
    - Kinesis Data Firehose buffers incoming data before delivering it to Splunk. The buffer size is 5 MB, and the buffer interval is 60 seconds.

- **Monitoring**
  - Kinesis Data Firehose exposes several metrics through the console, as well as CloudWatch for monitoring.
  - Kinesis Agent publishes custom CloudWatch metrics, and helps assess whether the agent is healthy, submitting data into Kinesis Data Firehose as specified, and consuming the appropriate amount of CPU and memory resources on the data producer.
  - Log API calls with CloudTrail.

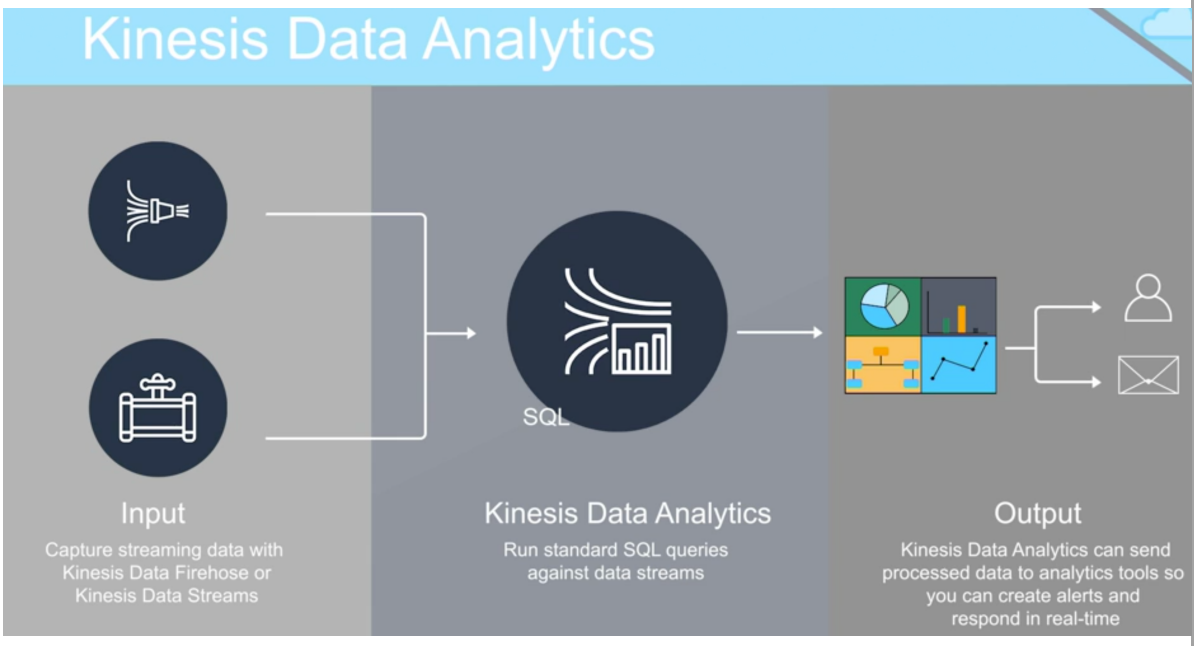


Kinesis Video Stream - Components

- Producer - Sender of the data
- Video Stream
  - Time-encoded data - Data records in time series data
  - Fragments - Self contained sequence of frames.
  - Chunks - Kinesis stores media set in Chunks.
- Consumer

<u>Video-Playback</u>

- HTTP Live Stream (HLS)
- GetMediaAPI



Kinesis - Event Store Patterns

- With the *event sourcing* pattern, instead of updating data stores directly, any events with significance to business logic–such as orders being placed, credit inquiries being made, or orders being processed or shipped–are added to a durable event log. Because each event record is stored individually, all updates are *atomic* (indivisible and irreducible).
- A key characteristic of this pattern is that the application state at any point in time can be rebuilt by simply reprocessing the stored events. Because data is stored as a series of events rather than through direct updates to data stores, various services can replay events from the event store to compute the appropriate state of their respective data stores.

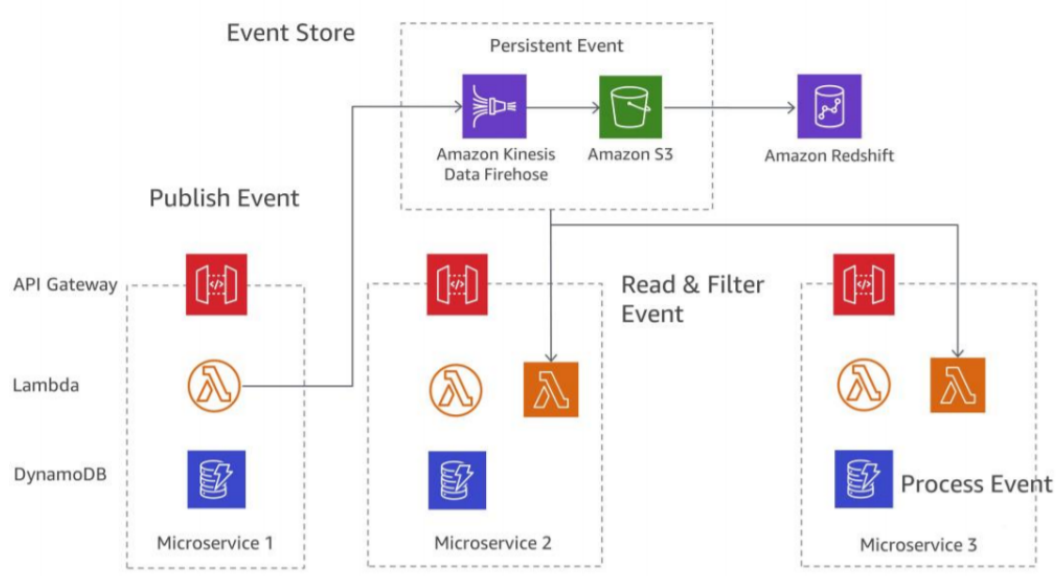


Figure 6: Event sourcing pattern on AWS

Event sourcing heavily relies on the persistence of events so it can rebuild/replay the state of your application at any given point in time in the past. Among the options given, Amazon Kinesis Data Firehose is the most suitable service to stream data as it provides an ordering of records, as well as the ability to read and/or replay records in the same order.

INCORRECT choices:

Configure the first microservice to send data to an Amazon SQS queue, then send the event log to an Amazon S3 bucket. Modify the second microservice to fetch data from the queue is incorrect because although SQS queue can be used as an event source, it does not have the functionality required in building an event store as it is just used to decouple applications by storing messages in the queue as they travel between computers. Amazon Kinesis Data Firehose Stream is capable of preserving the order of

records and can also replay the records in the same order which is the key characteristic of an event sourcing application.