**StarLabs 2022 - Documentation**

**Apache Kafka**

***What is Apache Kafka?***

Apache Kafka is a distributed data store optimized for ingesting and processing streaming data in real-time. Streaming data is data that is continuously generated by thousands of data sources, which typically send the data records in simultaneously. A streaming platform needs to handle this constant influx of data, and process the data sequentially and incrementally. Kafka is primarily used to build real-time streaming data pipelines and applications that adapt to the data streams. It combines messaging, storage, and stream processing to allow storage and analysis of both historical and real-time data.

***Kafka provides three main functions to its users:***

1. Publish and subscribe to streams of records
2. Effectively store streams of records in the order in which records were generated
3. Process streams of records in real time

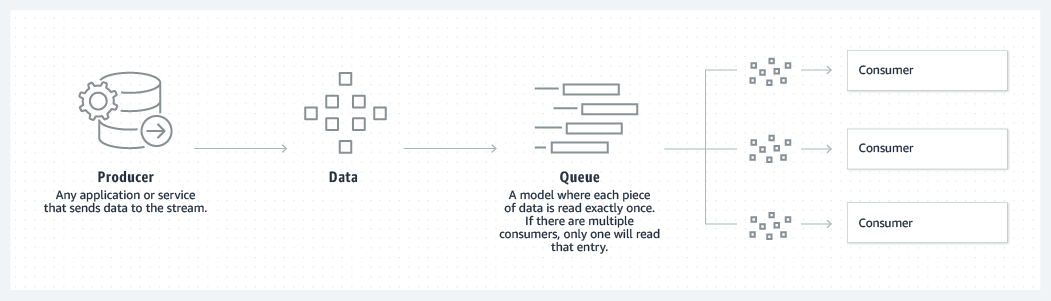
***Why would you use Kafka?***

Kafka is used to build real-time streaming data pipelines and real-time streaming applications. A data pipeline reliably processes and moves data from one system to another, and a streaming application is an application that consumes streams of data. For example, if you want to create a data pipeline that takes in user activity data to track how people use your website in real-time, Kafka would be used to ingest and store streaming data while serving reads for the applications powering the data pipeline. Kafka is also often used as a message broker solution, which is a platform that processes and mediates communication between two applications.

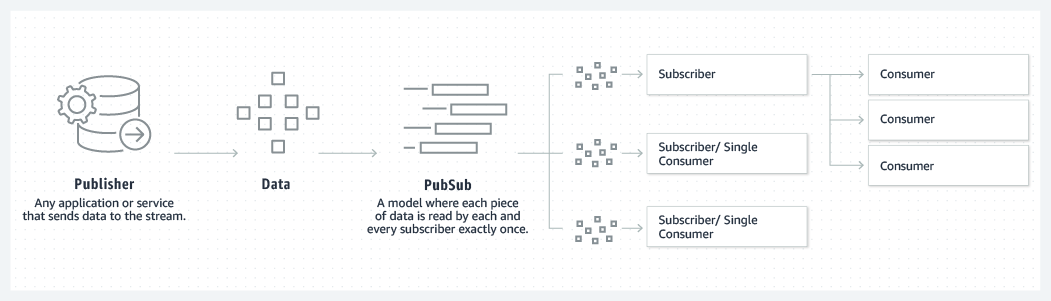
***How does Kafka work?***

Kafka combines two messaging models, queuing and publish-subscribe, to provide the key benefits of each to consumers. Queuing allows for data processing to be distributed across many consumer instances, making it highly scalable. However, traditional queues aren’t multi-subscriber. The publish-subscribe approach is multi-subscriber, but because every message goes to every subscriber it cannot be used to distribute work across multiple worker processes. Kafka uses a partitioned log model to stitch together these two solutions. A log is an ordered sequence of records, and these logs are broken up into segments, or partitions, that correspond to different subscribers. This means that there can be multiple subscribers to the same topic and each is assigned a partition to allow for higher scalability. Finally, Kafka’s model provides replay ability, which allows multiple independent applications reading from data streams to work independently at their own rate.

***Queuing:***

******

***Publish-Subscribe:***

******

[***Click here for a 10 min Kafka Intro (CTRL+Click)***](https://www.youtube.com/watch?v=FKgi3n-FyNU&t=1s)

Two different producer clients are publishing, independently from each other, new events to the topic by writing events over the network to the topic's partitions. Events with the same key (denoted by their color in the figure) are written to the same partition. Note that both producers can write to the same partition if appropriate.

