Kafka 101

Kafka is an Event Streaming Platform. The events are organized as key/value pairs binary encoded in some format (ex. Avro, Json). The key/value themselves can have any schema as defined by business logic, but Kafka stores and serves key/values as bytes

Topics

- A topic is a unit of organizing events. It represents a Log of events

Partitioning

- Topics are stored as log files on disk and are broken down into partitions to be able to scale out on multiple machines

Brokers

- Brokers are Kafka cluster nodes that can run on bare metal, VM, Docker etc. and perform
  - Write events to new partition
  - Read events from existing partitions
  - Replicate partitions
- Brokers don't do any computation over messages or routing of messages between topics.

Replication

- Each topic has a configurable replication factor that determines how many of these copies will exist in the cluster in total. One of the replicas is elected to be the leader, and it is to this replica that all writes are produced and from which all reads are probably consumed
- The other replicas are called *followers*, and it is their job to stay up to date with the leader and be eligible for election as the new leader if the broker hosting the current leader goes down.

Producers

- A producer is an external application that writes messages to a Kafka cluster, communicating with the cluster using Kafka's network protocol.
- The producer library manages all of the non-trivial network plumbing between your client program and the cluster and also makes decisions like how to assign new messages to topic partitions.

Consumers

- The consumer is an external application that reads messages from Kafka topics and does some work with them, like filtering, aggregating, or enriching them with other information sources.
- Like the producer, it relies on a client library to handle the low-level network interface in addition to some other pretty sophisticated functionality. A consumer can be just a single instance, or it can be many instances of the same program: a consumer group.
- Consumer groups are elastically scalable by default, but the library only manages some of the challenges associated with scale-out and fault tolerance. For example, if your consumer is stateful (and it probably is), then you'll have some extra work to do to manage that state during failover or scaling operations.