**Serializers**

Producer configuration includes mandatory serializers. String serializers are most commonly used. We can also use AVRO custom serializer.

ProducerRecord<String, GenericRecord> data = new ProducerRecord<String, GenericRecord>("customerContacts", name, customer);

1. **Above is the sample producer with topicName as customerContacts, Partition Id or key as name, value as customer.**
2. When the key is null and the default partitioner is used, the record will be sent to one of the available partitions of the topic at random. Round-robin algorithm will be used to balance the messages between the partitions.
3. If a key exists Kafka will hash the key, and use the result to map the message to a specific partition. Note that this time, it is important that a key will always get mapped to the same partition, so we use all the partitions in the topic to calculate the mapping and not just available partitions.
4. Also note that the mapping of keys to partitions is consistent only as long as the number of partitions in a topic does not change. So as long as the number of partitions is constant you can be sure that, for example, records regarding user 045189 will always get written to partition 34.

**Kafka clients are users of the system, and there are two basic types: producers and consumers.**

The consumer keeps track of which messages it has already consumed by keeping track of the offset of messages. The offset is another bit of integer metadata added by Kafka broker for each message. Consumer stores last consumed offset number in Zookeeper or in Kafka, so that it can stop and restart without losing its place.

**What is a broker and what are the duties of broker?**

1. A single Kafka server is called a broker.
2. The broker receives messages from producers, assigns offsets to them, and commits the messages to storage on disk.
3. It also services consumers, responding to fetch requests for partitions and responding with the messages that have been committed to disk
4. Kafka brokers are designed to operate as part of a cluster. A cluster can be formed within same nodes (running multiple kafka in same machines with different server-1.properties) or across different nodes.
5. Within a cluster of brokers, one will also function as the cluster controller (elected automatically from the live members of the cluster).
6. The controller is responsible for administrative operations, including assigning partitions to brokers and monitoring for broker failures.

The term stream is often used when discussing data within systems like Kafka. Most often, a stream is considered to be a single topic of data, regardless of the number of partitions. This represents a single stream of data moving from the producers to the consumers.

<https://www.mapr.com/blog/getting-started-sample-programs-apache-kafka-09>

Kafka is distributed replicated log.

It looks like messaging system and has messaging features, but not.

Kafka decouples data pipeline through broker system. So producer can feed data and consumer can process data.

Topic in Kafka is nothing but a logical grouping of partitions.

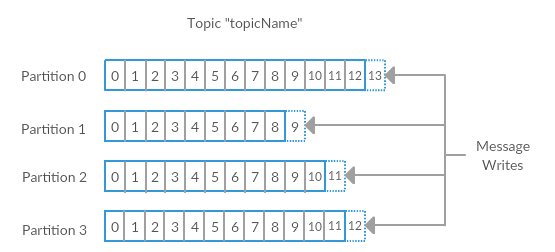
A partition can be think of an log file or directory of log file.

A partition is split into segments.

If there is no partition present on topic, message can be read directly from topic.

If partition is present then key needs to be given, which uniquely identifies a partition.

**A topic can be thought of as database table. A partition can be thought of as an row. The one difference is column will not grow in db table, whereas in partition column will grow or data is appended at end of column.**

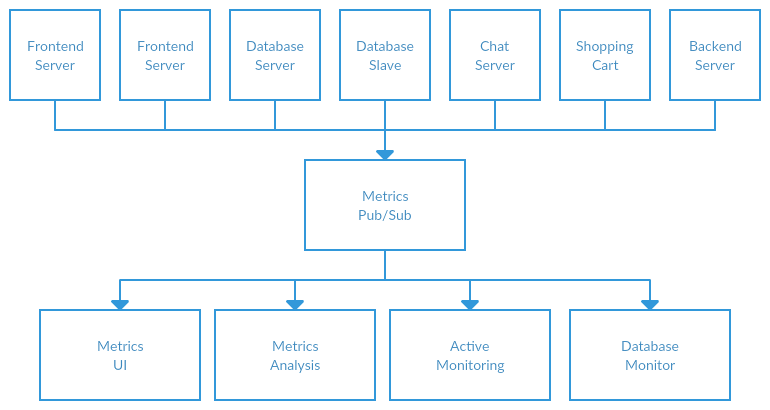


For efficiency, messages are written into Kafka in batches. A batch is just a collection of messages, all of which are being produced to the same topic and partition. **An individual round trip across the network for each message would result in excessive overhead, and collecting messages together into a batch reduces this.**

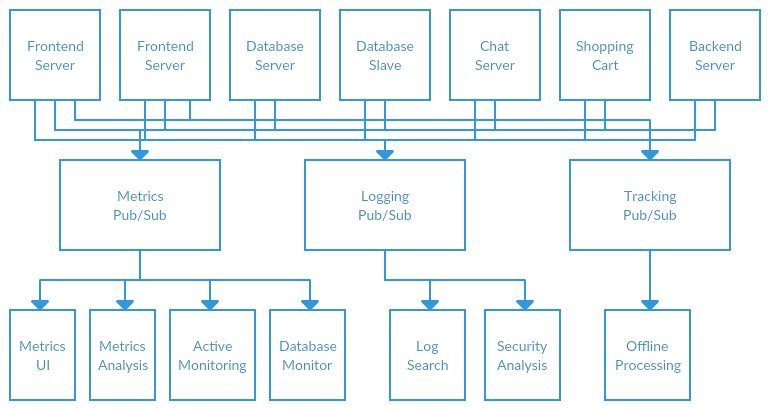
Message types are 1) byte array (default) 2)JSON 3) XML 4) Apache Avro

When we write to partition we are always appending log to end of file.

**"If all the consumer instances have the same consumer group like below diagram, then this works just like a traditional queue balancing load over the consumers."**



**Kafka guarantees each message is delivered exactly to one consumer from a group (with a same group id). So if you run 2 consumer listening for a message in a group only one consumer will get the message. Example in below diagram a single metrics message can be received and processed by only one of 4 consumers.**



Kafka is not traditional types message queue. Kafka is however a distributed system for storing continuous information / data in queue form. We can use Kafka as a replacement of traditional message queue framework depending on the requirements. Linkedin started the usage of Kafka as for log processing.

 The one drawback to Kafka is that each client needs to connect to every Kafka server in a particular cluster versus making only a single connection to the cluster. It needs to do this because the data being put into a Kafka cluster is split across members of the cluster.