Kafka is a distributed event ledger which is redundant. Kafka uses messaging system semantics (producer, consumer, topics). Clustering is the core in Kafka. Kafka has durability & ordering guaranteed.

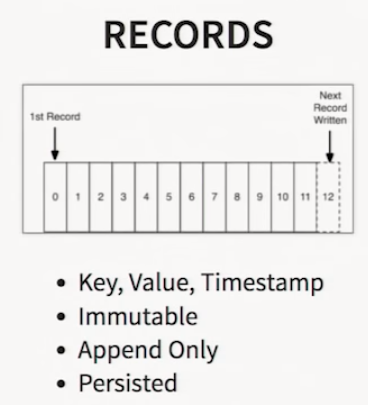
Kafka uses Zookeeper to maintain the state of the cluster, like if any of the node goes down, a new node will be spawn.



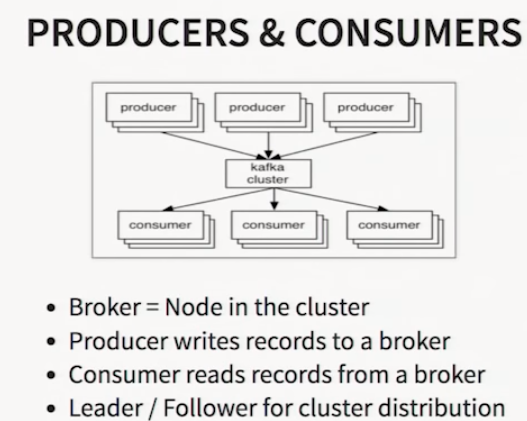
**Use Cases**

* Modern ETL or change data capture
* Data pipelines (basically kafka working as a hub for big data analysis where different systems plug onto kafka as producers and consumers)
* Big data Ingest (Kafka can easily get billions of data as buffer so that downstream systems can take their own time to consume)

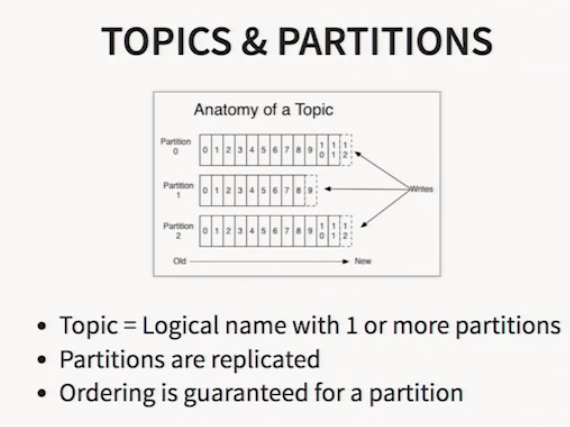
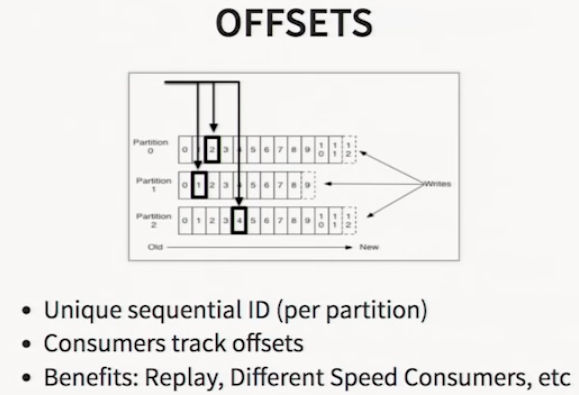
Kafka has records/message/events (key value, timestamp) which are immutable and append only. They are actually persisted on disk.



A Kafka node in the cluster is called broker. Producer writes records to a broker. Consumer reads record to a broker.



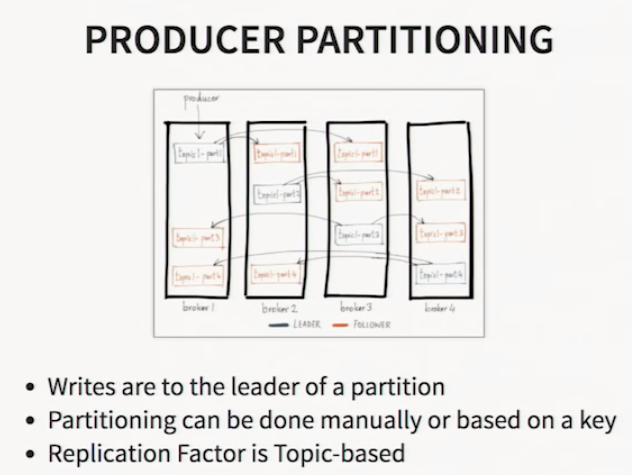
Leader/Follow for cluster distribution.

Inside Kafka, there are topics and partitions. A topic is a logical name for one or more partitions. Partitions are replicated and ordering is only guaranteed for a partition. Offsets are for keeping tracking of order in partition. It will have unique sequential ID and whenever we write, it will actually tell what’s the next sequential ID (offset).

Consumers track offsets (first time, it will be 0). If consumer reconnects, we can configure what will be the offset value. Benefits with Offset - replay the partition, support diff speed consumers.

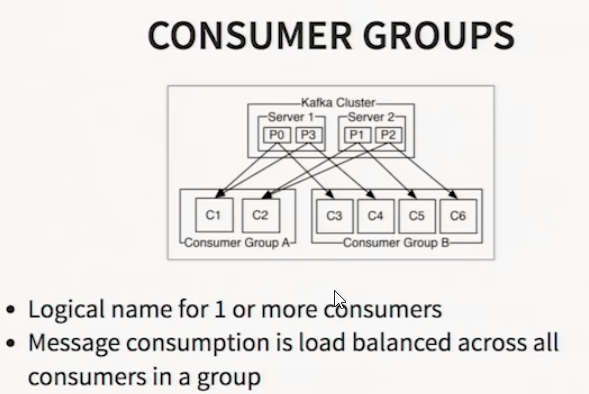
Producer Partitions



Write to the leader of the partition. There are followers which will replicate the partition to other nodes. Replication Factor is Topic based (if it 3, there will be 2 replicas in other nodes).

Consumer Groups

Kafka will ensure message on partition will be delivered to only one node of the consumer group.



Delivery Guarantees



It’s better to go with Async for producers as it is fast. But for critical data, Committed to Leader or Committed to Leader & Quorum (followers) may be needed but it may lead to other operations like writing to disk for confirmation to commitment (disk speed) and network latency (when calling other followers).

Consumer delivery guarantees have multiple approaches based on options to manage offset and criticality.

Cool features

Log Compaction – it’s an option, it runs background and will sweep through and compact the duplicate records. It checks whether the key is same and will compact it down if they are sequential.

Disk not Heap – It just uses disk cache not heap.

Pagecache to Socket – It can move the disk data to socket through network (for replication) instead of copying through JVMs

Balanced Partitions & Leaders – Automatic rebalancing of nodes and electing a new leader. Kafka clients will use Zookeeper to connect to Kafka events.

Producer & Consumer Quotas – configure events/second.

Heroku Kafka – Cloud Kafka, even AWS has it.

**Clients**

JVM, Polling based

**API**

For producer and consumer



**Reactive Kafka**



Apache Akka – Implementation of reactive streams

* + Source/Sink Stream programming
  + Back pressure (tell producer to slow down during spikes)
  + Reactive Kafka as adapter