

Base Concepts

A **cluster** consists of **brokers**=nodes that stores **topics**; each topic consists of 1+ **partitions**

Each **topic** has a **replication factor** that defines how many copies of each partitions are stored across different brokers

Producers send data to the **leader partition**, other **follower partitions** are stored for fault tolerance

Consumers usually read from the leader partition, but could be configured to read from replicas

Smart clients in producers and consumers; clients pull **cluster meta data** from any broker and know who is the leader and where they need to send or fetch data.

Architecture

Kafka cluster consists of brokers. Everything else are *outside* of a cluster.

Smart clients in producers and consumers; clients pull **cluster meta data** from a broker and decide where they need to send or request data. via Consumer and Producer APIs

Kafka Connect for export/import data to/from external systems like databases

Kafka Streams provides client based streaming framework, that involves **RockDB** client library + additional **topics**, made by streams framework, for stateful streams are stores

Producer Configuration

Broker Configuration

Schema Evolution

Kafka Tools

Kafka Availability

Kafka Throughput Configuration

Internals

A cluster consists of a **control plane** and **data plane**. Both these planes are processes that run on broker nodes;

Control Plane responsible for **cluster metadata**: new **KRaft consensus protocol**, old **Zookeeper**. One controller is active (an elected leader) - does not manipulate partitions, but only cluster meta-data

Data plane responsible for **client request processing** and **data=partition replication**.

Data Replication: **ISR - In-Sync Replicas** is a list of "broker ids" that are fully in-sync with a leader partition. So further, only brokers from ISR can be participated in leader election.

Data Replication: partition leader doesn't send data to followers, yet followers replicate from the leader.

Data Replication: a broker will be **removed from ISR** when it is lagging behind the leader for longer than **replica.lag.time.max.ms**

Data Replication: Consumers could fetch records only **by a HW (high watermark offset)**. Follower brokers **get HW with fetch response** from a leader.

Partition Leader Balancing: as a leader does more work than followers, kafka tries **evenly distribute partition leaders across all brokers**.

Partition Leader Balancing: **auto.leader.rebalance.enable** enables Kafka automatically rebalance partition leaders to their preferred replicas.

Partition Leader Balancing: If manual rebalance is desired, the **kafka-preferred-replica-election.sh**. It explicitly triggers a **preferred replica election** for specified partitions or the entire cluster, forcing leaders back to their preferred replicas.

Partition Distribution: **kafka-reassign-partitions.sh** redistribute **partitions across brokers**, which inherently influences **leader distribution**, especially when adding new brokers or addressing specific load imbalances.

Consumer Configuration



Security

Kafka Streams

Kafka Connect

Kafka Low Latency Configuration

Kafka Low Latency Configuration



By **Basim** (BasimDev)
cheatography.com/basimdev/

Not published yet.
Last updated 30th October, 2025.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>