Kafka Architecture & Core Concepts

This document explains Apache Kafka architecture and concepts in a clear, backend-developer-friendly way. It is suitable for learning, revision, and interview preparation.

1. What is Apache Kafka?

Apache Kafka is a distributed, fault-tolerant event streaming platform used for building real-time data pipelines and streaming applications. It is designed for high throughput, low latency, and scalability.

2. Kafka Architecture Overview

Kafka follows a distributed architecture consisting of Producers, Brokers, Topics, Partitions, Consumers, and Zookeeper/KRaft.

3. Core Components

Producer: Publishes messages to Kafka topics.
Broker: Kafka server that stores data and serves clients.
Topic: Logical stream of records.
Partition: Ordered, immutable sequence of records.
Consumer: Reads messages from topics.
Consumer Group: Group of consumers sharing partitions for parallelism.

4. Topics & Partitions

Each topic is divided into partitions. Each partition is an ordered log where records are appended sequentially. Partitions allow Kafka to scale horizontally.

5. Offset Management

Each message in a partition has an offset. Consumers track offsets to know which messages are processed. Offsets are stored in an internal Kafka topic (__consumer_offsets).

6. Consumer Groups & Rebalancing

Consumers in the same group divide partitions among themselves. When a consumer joins or leaves, a rebalance occurs, and partitions are reassigned.

7. Message Delivery Semantics

At-most-once: Message may be lost.
At-least-once: Message may be processed more than once.
Exactly-once: Message processed exactly once (using transactions).

8. Replication & Fault Tolerance

Each partition has replicas. One replica is the leader, others are followers. Kafka ensures durability by replicating data across brokers.

9. Zookeeper vs KRaft

Older Kafka versions used Zookeeper for metadata management. Modern Kafka versions use KRaft (Kafka Raft) which removes Zookeeper dependency.

10. Why Kafka is Fast

Kafka uses sequential disk writes, zero-copy transfer, batching, and efficient network usage to achieve high throughput.

11. Kafka Use Cases

- Event-driven microservices
- Log aggregation
- Real-time analytics
- Data pipelines
- Stream processing

12. Kafka in Real Projects

Kafka is widely used with Spring Boot, microservices, databases, and cloud platforms for reliable messaging and streaming architectures.