**Apache Kafka** is an open-source distributed **event streaming platform** used for building real-time data pipelines and streaming apps.

It’s used to:

* **Publish** and **subscribe** to streams of records (like a message queue)
* **Store** data in a distributed, fault-tolerant way
* **Process** streams in real-time

| **Term** | **Description** |
| --- | --- |
| **Producer** | Sends messages to Kafka topics |
| **Consumer** | Reads messages from topics |
| **Broker** | Kafka server that stores messages |
| **Topic** | Category or feed name to which messages are published |
| **Partition** | Topic split to allow scalability and parallelism |
| **Offset** | A unique ID for each message in a partition |
| **Consumer Group** | A group of consumers sharing the load of reading messages |
| **Zookeeper** | (Pre-Kafka 3.x) Manages metadata and cluster coordination (being phased out) |

**Kafka Interview Questions and Answers**

**🔹 1. What is Kafka and how does it work?**

**Answer:**  
Kafka is a distributed streaming platform used to handle real-time data feeds. Producers send records to **topics**, which are divided into **partitions** and stored in **brokers**. Consumers read these messages using offsets. Kafka ensures durability, scalability, and fault-tolerance.

**🔹 2. What is the role of ZooKeeper in Kafka?**

**Answer:**  
ZooKeeper helps Kafka manage:

* Cluster metadata
* Leader election for partitions
* Configuration and coordination

🆕 As of **Kafka 3.0+**, Kafka is moving toward a **KRaft mode** (Kafka Raft) to remove dependency on ZooKeeper.

**🔹 3. What is a Kafka topic?**

**Answer:**  
A topic is a logical channel where records are published. Topics are split into **partitions** to improve scalability and performance.

**🔹 4. What is the difference between Kafka and traditional messaging systems (like RabbitMQ)?**

| **Kafka** | **Traditional MQ** |
| --- | --- |
| Distributed & scalable | Often centralized |
| High-throughput | Medium throughput |
| Persistent logs | Queue semantics |
| Pull-based consumers | Often push-based |

**🔹 5. How does Kafka ensure message durability?**

**Answer:**

* Data is **written to disk** (log file)
* Replication across brokers
* Acks ensure producers wait for message persistence

**🔹 6. What are consumer groups in Kafka?**

**Answer:**  
A **consumer group** allows multiple consumers to share load by assigning **partitions** to different members of the group. Each partition is read by only one consumer in the group.

**🔹 7. What happens if a consumer fails in a consumer group?**

**Answer:**  
Kafka reassigns the failed consumer’s partitions to other members in the group during **rebalance**.

**🔹 8. What are the different types of Kafka acknowledgments?**

**Answer:**

* acks=0: Producer doesn’t wait for any acknowledgment.
* acks=1: Wait for the leader to acknowledge.
* acks=all: Wait for all in-sync replicas to acknowledge (strongest durability).

**🔹 9. How does Kafka guarantee message ordering?**

**Answer:**  
Messages within a **partition** are strictly ordered. If a consumer reads from a single partition, it receives messages in the same order as produced.

**🔹 10. What is idempotence in Kafka?**

**Answer:**  
Kafka producers can be configured to **avoid duplicate messages** using **idempotent producer settings**, ensuring exactly-once delivery semantics.

**🚀 Bonus: Sample Code Snippet (Kafka Producer)**

java

CopyEdit

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092");

props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

Producer<String, String> producer = new KafkaProducer<>(props);

producer.send(new ProducerRecord<>("my-topic", "key", "Hello, Kafka!"));

producer.close();

Would you like a follow-up with **Kafka architecture diagrams**, **Spring Boot + Kafka integration**, or questions for **Kafka with microservices**?

| **Feature** | **Database** | **Kafka** |
| --- | --- | --- |
| Stores data | In tables | In topic logs (append-only) |
| Query | SQL | Offset-based read |
| Access | Random (any row) | Sequential (like a queue) |
| Data lost? | No | No – kept as long as configured |