Frequently Asked Questions on Apache Kafka

1. What is Apache Kafka?

Answer: Apache Kafka is an open-source distributed event streaming platform used to build real-time data pipelines and streaming applications. It is primarily used for publishing, subscribing to, storing, and processing streams of records in real time.

2. What are the main components of Kafka?

Answer: The core components of Kafka are:

- **Producer: Sends messages to Kafka topics.**
- **Consumer: Reads messages from Kafka topics.**
- **Broker: A Kafka server that stores data and serves clients (producers and consumers).**
- **Topic: A category or feed name to which messages are sent.**
- **Partition: Subdivision of topics to enable parallel processing.**
- **Zookeeper (deprecated as of KRaft mode): Manages the Kafka cluster's metadata and leader election for partitions.**

3. What is a Kafka Producer?

Answer: A Kafka producer is a client application that publishes messages (or records) to Kafka topics. Producers push data to topics in Kafka and are responsible for determining which partition of a topic a message will go to.

4. What is a Kafka Consumer?

Answer: A Kafka consumer is an application that reads messages from Kafka topics. Consumers can read from multiple partitions of a topic and process data as it arrives. They often work in consumer groups to distribute the load of reading messages.

5. What is a Kafka Broker?

Answer: A Kafka broker is a Kafka server that stores data and serves clients (producers and consumers). It handles the persistence and retrieval of data for Kafka topics and partitions.

Frequently Asked Questions on Apache Kafka

6. What is a Kafka Topic?

Answer: A topic in Kafka is a logical channel to which producers send messages and from which consumers consume messages. Topics are partitioned to allow scalability and parallel processing.

7. What is Kafka's message format?

Answer: Kafka messages are typically in a key-value pair format, with each message containing a key, value, and metadata such as a timestamp and offset.

8. What is the role of partitions in Kafka?

Answer: Partitions allow Kafka to scale horizontally. Each partition is a log that can be written to and read from independently. Kafka ensures data is distributed across partitions to balance load and support high availability.

9. What is the difference between Kafka and traditional messaging ?

Answer: Kafka is designed for high throughput and horizontal scalability, with a focus on streaming large volumes of data. Unlike traditional messaging systems, Kafka is a distributed, fault-tolerant, and highly scalable platform built to handle large-scale real-time data feeds. Kafka stores messages for longer durations, enabling replay and stream processing, while traditional systems often focus on short-lived messages and message queuing.

10. What is Kafka Consumer Group?

Answer: A Kafka consumer group is a group of consumers that share the responsibility of consuming messages from a topic. Each message in a topic partition is consumed by only one consumer in the group, which allows Kafka to scale message consumption across multiple consumers.

12. How does Kafka achieve fault tolerance?

Answer: Kafka achieves fault tolerance through data replication. Each partition can be replicated across multiple brokers. If a broker goes down, one of its replicas can take over, ensuring no data loss and continuous availability.

Frequently Asked Questions on Apache Kafka

13. What is Kafka Streams?

Answer: Kafka Streams is a client library for building real-time streaming applications that process data stored in Kafka topics. It allows for complex stream processing tasks like filtering, aggregating, and joining data streams.

14. What is the Kafka "offset"?

Answer: In Kafka, an offset is a unique identifier for each message within a partition. Consumers use offsets to track their position in a partition, enabling them to continue reading from where they left off.

15. What is the difference between Kafka and RabbitMQ?

Answer: Kafka is designed for high-throughput, distributed log-based storage and real-time stream processing, while RabbitMQ is optimized for reliable message queuing with support for routing, clustering, and guarantees for message delivery. Kafka excels in handling large-scale streaming data, while RabbitMQ is more suited for traditional message queuing scenarios.

16. What are the advantages of using Kafka?

Answer:

- High throughput: Kafka is designed to handle massive amounts of data with low latency.
- Scalability: Kafka clusters can easily scale by adding more brokers.
- Fault tolerance: Kafka replicates data across brokers to ensure reliability.
- Stream processing: Kafka is well-suited for real-time analytics and stream processing.
- Durability: Kafka persists messages on disk and ensures they are replicated to avoid data loss.