APACHE KAFKA
SHORT QUESTIONS

PRESENTED BY

SHIZA IMRAN
MAHAM ASIF

1. How does Kafka achieve fault tolerance?

Through its distributed nature and replication of partitions across multiple brokers.

2. What is the purpose of partitions in Kafka?

To split topics into ordered sequences of records, enabling parallel processing and improved throughput.

3. How does Kafka handle message ordering within a partition?

Messages within a partition are ordered and immutable, with each message assigned an incremental offset value.

4. Describe the role of the Kafka Producer API.

It allows applications to write or publish data to Kafka topics or specific partitions.

5. What is the function of the Kafka Consumer API?

It enables applications to read or consume data from Kafka topics, associating with a consumer group.

6. How does Kafka ensure high throughput for message processing?

By distributing load across multiple brokers and using efficient write and read mechanisms.

7. Explain the concept of a distributed commit log in Kafka.

It ensures message durability by persisting records on disk as fast as possible.

8. What is the role of Zookeeper in a Kafka cluster?

To maintain cluster metadata, manage broker coordination, and handle controller election.

9. **How do you create a new Kafka topic with specific partitions and replication factor?**

Using the command: `kafka-topics --create --topic my_topic --bootstrap-server localhost:9092 --partitions 3 --replication-factor 2`.

10. **What steps are involved in setting up Kafka in a Java environment?**

Installing WSL2, Java, Kafka, starting Zookeeper, starting Kafka server, creating a topic, and starting Kafka producer and consumer.

11. **Describe the architecture of an Apache Kafka cluster and explain how it handles data distribution and fault tolerance.**

An Apache Kafka cluster consists of multiple brokers (servers) that manage message data. Each Kafka topic is divided into partitions, which are distributed across different brokers to balance the load. Kafka ensures fault tolerance through replication; each partition has multiple replicas on different brokers. If a broker fails, Kafka can switch to another broker that holds a replica of the data, ensuring no data loss and high availability. Zookeeper is used for cluster management, including leader election for partitions and maintaining metadata. Kafka brokers handle data persistence and retrieval, while producers send data to topics and consumers read data from topics. The distributed nature of Kafka allows it to scale horizontally by adding more brokers, enhancing both storage capacity and processing power.