Title: Apache Kafka Interview Questions and Answers (100+ Q&A)

Introduction: This guide contains over 100 interview-level questions and answers on Apache Kafka, covering Zookeeper, KRaft mode, producers, consumers, transactions, and integration with ELK and Splunk. It is intended as a comprehensive preparation resource for software engineers, DevOps, and data engineers.

---

1. Kafka Basics

1. **What is Apache Kafka?**
2. Answer: Kafka is a distributed streaming platform used for building real-time data pipelines and streaming applications.
3. **What are the main components of Kafka?**
4. Answer: Producer, Consumer, Broker, Topic, Partition, Zookeeper (optional in KRaft).
5. **What is a Kafka topic?**
6. Answer: A topic is a named logical channel to which messages are published by producers.
7. **What is a partition?**
8. Answer: A partition is a subset of a topic; partitions allow for parallelism and scalability.
9. **Difference between Kafka and traditional message queues?**
10. Answer: Kafka provides persistent storage, high throughput, horizontal scalability, and message replay, unlike traditional queues.
11. **Role of producer?**
12. Answer: Sends messages to Kafka topics.
13. **Role of consumer?**
14. Answer: Reads messages from topics and processes them.
15. **Push vs pull consumer model?**
16. Answer: Kafka uses a pull model where consumers request batches of messages.
17. **Kafka retention?**
18. Answer: Kafka retains messages for a configured time or size regardless of consumption.

19. **Durable vs ephemeral messaging?**
20. Answer: Kafka messages are durable; traditional queues may discard messages after delivery.

2. Kafka Architecture

1. **Explain Kafka broker.**
2. Answer: Broker is a server that stores messages and serves client requests.
3. **Kafka cluster?**
4. Answer: A set of brokers working together to provide scalability and fault tolerance.
5. **How does Kafka achieve scalability?**
6. Answer: By partitioning topics and distributing partitions across brokers.
7. **Leader and follower replicas?**
8. Answer: Leader handles reads/writes; followers replicate leader data.
9. **ISR (In-Sync Replica)?**
10. Answer: Set of replicas that are fully caught up with the leader.
11. **Failover handling?**
12. Answer: If a leader fails, a follower from ISR becomes leader.
13. **Role of Zookeeper?**
14. Answer: Maintains cluster metadata, broker status, leader election.
15. **Topics and partitions purpose?**
16. Answer: For organizing and scaling data streams.
17. **Replication factor vs partitions?**
18. Answer: Replication factor is number of copies; partitions enable parallelism.
19. **High throughput?**
20. Answer: Achieved via batching, compression, and sequential disk writes.

3. Kafka Producers and Consumers

1. **Producer acknowledgments (acks)?**

2. Answer: 0=fire-and-forget, 1=leader ack, all=all ISR ack.
3. **Sync vs async producer?**
4. Answer: Sync waits for ack; async sends without blocking.
5. **Idempotent producer?**
6. Answer: Ensures messages are not duplicated on retries.
7. **Consumer groups?**
8. Answer: Multiple consumers share topic partitions for parallel processing.
9. **Consumer offsets?**
10. Answer: Track last read position in partitions.
11. **Auto vs manual commit?**
12. Answer: Auto commits offsets automatically; manual allows controlled commit.
13. **Rebalancing?**
14. Answer: Redistributes partitions among consumers when group membership changes.
15. **Message key usage?**
16. Answer: Determines which partition a message is sent to.
17. **Message ordering?**
18. Answer: Preserved per partition.
19. **Backpressure handling?**
20. Answer: Consumers poll at their pace; producers can buffer messages.

4. Kafka Brokers, Partitions, and Replication

1. **Data replication?**
2. Answer: Leader replicates messages to followers.
3. **Leader election for partitions?**
4. Answer: Zookeeper (or KRaft) selects leader among replicas.
5. **Broker metadata management?**

6. Answer: Metadata includes broker IDs, topics, partitions, leader info.
7. **Log segment?**
8. Answer: Segment is a chunk of partition logs stored on disk.
9. **Log compaction?**
10. Answer: Retains only latest value per key.
11. **Parallelism via partitions?**
12. Answer: Consumers can read different partitions concurrently.
13. **Default replication factor?**
14. Answer: Typically 1 (configurable).
15. **Broker failure handling?**
16. Answer: ISR followers take over leader responsibilities.
17. **Data durability?**
18. Answer: Achieved via replication and disk storage.
19. **Committed vs uncommitted messages?**
20. Answer: Committed messages are acknowledged by required replicas.

5. Kafka Zookeeper Role

1. **What is Zookeeper?**
2. Answer: Centralized service for configuration, coordination, and synchronization.
3. **Why Kafka used Zookeeper?**
4. Answer: To manage metadata, leader election, and cluster state.
5. **Metadata stored?**
6. Answer: Broker info, topic configs, partition leaders, ISR info.
7. **Leader elections?**
8. Answer: Zookeeper selects partition leaders when needed.
9. **Broker discovery?**

10. Answer: Brokers register with Zookeeper; clients query Zookeeper.
11. **Broker health monitoring?**
12. Answer: Zookeeper watches broker heartbeat nodes.
13. **Zookeeper vs KRaft?**
14. Answer: KRaft removes Zookeeper dependency, uses Raft internally.
15. **Topic metadata in Zookeeper mode?**
16. Answer: Stored in Zookeeper znodes.
17. **Limitations?**
18. Answer: External dependency, complex deployment, potential performance bottleneck.
19. **Securing Zookeeper?**
20. Answer: Use ACLs, TLS, authentication mechanisms.

6. Kafka KRaft Mode

1. **What is KRaft?**
2. Answer: Kafka Raft mode; manages metadata without Zookeeper.
3. **Why no Zookeeper?**
4. Answer: Raft quorum handles leader election and metadata internally.
5. **Kafka controller?**
6. Answer: Node responsible for cluster metadata and leadership.
7. **Metadata storage?**
8. Answer: Stored in internal topics (`__cluster_metadata`).
9. **Raft consensus?**
10. Answer: Ensures metadata replication across controllers.
11. **Advantages over Zookeeper?**
12. Answer: Simpler architecture, faster leader election, no external dependency.
13. **Migration from Zookeeper?**

14. Answer: Supported via migration tools in newer Kafka versions.
15. **Leader election in KRaft?**
16. Answer: Raft controller quorum elects leaders.
17. **Quorum controllers?**
18. Answer: Multiple controllers to maintain cluster consensus.
19. **Kafka versions supporting KRaft?**
20. Answer: Kafka 2.8+ with incremental improvements; production-ready 3.3+.

7. Kafka Transactions & Exactly-Once Semantics

1. **Exactly-once semantics (EOS)?**
2. Answer: Guarantees a message is processed exactly once from producer to consumer.
3. **How EOS achieved?**
4. Answer: Idempotent producers + transactional APIs.
5. **Idempotent producer?**
6. Answer: Prevents duplicate messages on retries.
7. **Transaction initiation?**
8. Answer: Producer begins transaction via API.
9. **Transaction IDs?**
10. Answer: Unique ID to identify and manage transactions.
11. **EOS vs at-least-once?**
12. Answer: EOS ensures single delivery; at-least-once may duplicate.
13. **Consumers handling transactional messages?**
14. Answer: Read only committed transactional messages.
15. **Broker failure during transaction?**
16. Answer: Transaction is aborted if not committed.
17. **Producer retries?**

18. Answer: Handled idempotently with transaction coordinator.

19. **Transactional timeouts?**

20. Answer: Configured via `transaction.timeout.ms`.

8. Kafka Integration with ELK and Splunk

1. **Kafka → ELK integration?**

2. Answer: Logstash consumes Kafka topics, transforms, sends to Elasticsearch.

3. **Logstash role?**

4. Answer: Parsing, filtering, and enriching Kafka messages before indexing.

5. **Send Kafka messages to Elasticsearch?**

6. Answer: Use Kafka input plugin in Logstash and Elasticsearch output plugin.

7. **Visualize in Kibana?**

8. Answer: Create dashboards on indexed topics.

9. **Kafka → Splunk integration?**

10. Answer: Use Splunk Connect for Kafka or push via HEC.

11. **Splunk HEC?**

12. Answer: HTTP Event Collector for ingesting JSON logs.

13. **Kafka consumers to HEC?**

14. Answer: Write consumer app or use connector to send messages to Splunk HEC.

15. **Kafka → ELK vs Kafka → Splunk?**

16. Answer: ELK supports filtering, dashboards; Splunk supports alerting, SPL queries.

17. **High throughput handling?**

18. Answer: Batch messages, compress,