Question 1 of 15

What is Apache Kafka primarily designed for?

1 Relational database management

2 Distributed event streaming and message queuing

3 File system storage and backup

4 Static web content hosting

Correct!

Apache Kafka is an open-source distributed event streaming platform that can be used as either a message queue or stream processing system, designed for high performance, scalability, and durability.

Question 2 of 15

In Kafka's architecture, what is the relationship between topics, partitions, and brokers?

1 Topics are logical groupings of partitions, which are distributed across brokers

2 They are all equivalent concepts

3 Brokers contain topics which contain partitions

4 Partitions contain brokers which contain topics

Correct!

Topics are logical groupings of partitions used for publishing and subscribing to data. Partitions are the physical storage units that allow parallel processing, and they are distributed across multiple brokers (servers) in the cluster.

Question 3 of 15

How does Kafka determine which partition a message should be sent to?

1 By the size of the message content

2 By the timestamp of the message

3 By hashing the message key to assign it to a specific partition

4 Random assignment to any available partition

Correct!

Kafka uses a partitioning algorithm that hashes the message key to assign the message to a specific partition. This ensures that messages with the same key always go to the same partition, preserving order at the partition level.

Question 4 of 15

Kafka partitions are append-only, immutable logs where messages cannot be altered or deleted once written.

1 True

2 False

Correct!

Each partition functions as an append-only log file where messages are sequentially added to the end. This immutability is central to Kafka's performance, reliability, and simplifies replication and recovery processes.

Question 5 of 15

What is the purpose of consumer groups in Kafka?

1 To encrypt messages for security

2 To ensure each message is processed by only one consumer in the group ✓

3 To compress messages for storage efficiency

4 To group producers together for load balancing

Correct!

Consumer groups ensure that each message is processed by exactly one consumer within the group, enabling load balancing and parallel processing while preventing duplicate message processing.

Question 6 of 15

How does Kafka handle fault tolerance for message durability?

1 By using database transactions

2 Through leader-follower replication across multiple brokers

3 By storing all messages in a single location

4 By creating periodic backups to external storage

Correct!

Kafka uses a leader-follower replication model where each partition has a leader replica handling reads/writes and multiple follower replicas that passively replicate data, ensuring durability and availability even if brokers fail.

Question 7 of 15

Kafka uses a push-based model where brokers actively send messages to consumers.

1 True

2 False

Correct!

Kafka uses a pull-based model where consumers actively poll brokers for new messages. This design choice allows consumers to control their consumption rate, prevents overwhelming slow consumers, and enables efficient batching.

Question 8 of 15

What is the recommended maximum message size in Kafka for optimal performance?

1 1MB

2 10MB

3 100MB

4 100KB

Correct!

While message size can be configured, it's recommended to keep messages under 1MB to ensure optimal performance through reduced memory pressure and better network utilization.

Question 9 of 15

What is a 'hot partition' problem in Kafka?

1 Partitions that are corrupted

2 Partitions that are physically overheating

3 Partitions that are too small

4 Partitions receiving disproportionately more traffic than others

Correct!

A hot partition occurs when one partition receives much more traffic than others (e.g., a popular ad getting many clicks), potentially overwhelming that partition's broker while leaving other brokers underutilized.

Question 10 of 15

Which strategy is NOT typically used to handle hot partitions in Kafka?

1 Automatically redistributing hot partitions across more brokers

2 Using compound keys

3 Random partitioning with no key

4 Random salting of keys

Correct!

Kafka cannot automatically redistribute hot partitions. Solutions include random partitioning, salting keys with random values, using compound keys, or implementing back pressure - but not automatic redistribution.

Question 11 of 15

What does setting acks=all accomplish in Kafka producer configuration?

1 Compresses messages automatically

2 Ensures maximum durability by waiting for all replicas to acknowledge ✓

3 Speeds up message sending

4 Enables message batching

Correct!

Setting acks=all ensures that a message is acknowledged only when all in-sync replicas have received it, providing maximum durability guarantees at the cost of slightly higher latency.

Question 12 of 15

When a Kafka consumer fails and restarts, it can resume processing from where it left off by using committed offsets.

1 True

2 False

Correct!

Consumers commit their offsets to Kafka after processing messages. When a consumer restarts, it reads its last committed offset and resumes from there, ensuring no messages are missed or duplicated.

Question 13 of 15

Why doesn't Kafka support built-in consumer retries like some other message queues?

1 It's not technically feasible

2 Kafka focuses on high performance streaming rather than complex retry logic ✓

3 It would require too much storage

4 Retries would break message ordering

Correct!

Kafka is designed as a high-performance streaming platform rather than a traditional message queue. Consumer retry patterns must be implemented using separate retry topics and dead letter queues rather than built-in mechanisms.

Question 14 of 15

Which technique can significantly improve Kafka producer throughput?

1 Disabling compression completely

2 Batching messages before sending to reduce network overhead ✓

3 Using only a single partition per topic

4 Sending messages one at a time for guaranteed delivery

Correct!

Batching messages allows producers to send multiple messages in a single request, significantly reducing network overhead and improving throughput. This can be configured with maxSize and maxTime settings.

Question 15 of 15

Kafka's default retention policy keeps messages for 7 days with no size limit by default.

1 True

2 False

Correct!

Kafka topics have configurable retention policies controlled by retention.ms and retention.bytes settings. The default is to retain messages for 7 days (168 hours) with retention.bytes set to -1 (no size limit). Size limits can be configured but are not set by default.