**Apache Kafka and Messaging Systems**

1. **What is Apache Kafka and why is it used in distributed systems?** Kafka is a distributed event streaming platform used for building real-time data pipelines and streaming applications. It helps decouple microservices by acting as a messaging layer. Example use cases include event sourcing, log aggregation, and real-time analytics.
2. **Explain Kafka's architecture.** Kafka consists of Producers, Consumers, Topics, Partitions, and Brokers:
   * Producers send messages to topics.
   * Topics are split into partitions.
   * Brokers manage partitions and store data.
   * Consumers read messages from topics. Kafka achieves fault tolerance using replication and high throughput with partitioning.
3. **What are Kafka partitions and why are they important?** Partitions allow Kafka to parallelize data handling. Each partition is an ordered sequence of records and a unit of parallelism and scalability. More partitions = more concurrent consumers.
4. **How does Kafka ensure message durability and reliability?** Kafka writes messages to disk and replicates them across brokers. The leader-follower replication model ensures data is not lost if a broker fails. acks=all can be set to ensure full commit acknowledgment.
5. **What is the role of Zookeeper in Kafka?** Zookeeper manages cluster metadata, broker status, topic configurations, and controller elections. As of newer versions, Kafka can run without external Zookeeper (KRaft mode).
6. **What are consumer groups in Kafka?** A consumer group allows multiple consumers to share the load of processing messages from a topic. Each message is delivered to only one consumer in the group.
7. **How do producers send data to Kafka?** Producers serialize messages into byte arrays and send them to a specific topic and optionally a partition. They can be configured for compression and retries. Example: Java clients use KafkaProducer APIs.
8. **How can you achieve message ordering in Kafka?** Kafka guarantees ordering per partition. If all messages for a key go to the same partition, order is preserved. Use a key-based partitioning strategy for this.
9. **How does Kafka handle message retention?** Kafka retains messages for a configurable time or size even after they're consumed. Useful for reprocessing and audit trails. Example: retention.ms=604800000 (7 days)
10. **What is a Kafka offset and how is it managed?** Offset is a unique ID of a message in a partition. Consumers track which offsets they have read. Kafka supports auto and manual offset commit.
11. **How do you scale Kafka consumers?** By increasing the number of partitions and assigning more consumers in a group, processing can be parallelized. Each partition is read by one consumer in a group.
12. **How do Kafka producers achieve high throughput?** Techniques include:
    * Batching messages
    * Compression (snappy/gzip)
    * Async sends
    * Proper tuning of linger.ms and batch.size
13. **What are Kafka topics and how are they created?** Topics categorize messages. They can be created manually or automatically (if auto.create.topics.enable=true). Example: kafka-topics.sh --create --topic orders --partitions 3 --replication-factor 2
14. **What happens when a Kafka broker fails?** The leader partitions it hosted are reassigned to followers. Kafka continues operating normally unless replication is under-provisioned.
15. **What are the pros and cons of Kafka compared to RabbitMQ?** Kafka:
    * Pros: High throughput, scalability, built-in durability
    * Cons: More complex setup, requires tuning RabbitMQ:
    * Pros: Simpler, good for RPC patterns
    * Cons: Lower throughput, less suitable for big data streams
16. **How do you monitor Kafka clusters?** Use tools like Prometheus + Grafana, Kafka Manager, Burrow. Monitor lag, broker health, disk usage, and throughput.
17. **How is security handled in Kafka?** Kafka supports SSL encryption, SASL authentication, and ACLs for authorization. Example: Use SCRAM for username/password auth.
18. **What is idempotence in Kafka producers?** Idempotent producers ensure that even if a message is sent more than once due to retries, it is written exactly once to the partition. Enable by setting enable.idempotence=true.
19. **Explain Kafka Connect.** Kafka Connect is a framework to stream data between Kafka and external systems like databases and file systems. It supports source and sink connectors.
20. **Explain the role of schema registry in Kafka.** When using Avro/Protobuf, a schema registry helps manage versions of data schemas. It ensures compatibility and validation of message formats.

**Kafka One-Word Answers / Basics**

1. **Default message size** – 1MB
2. **Max message size (configurable)** – 100MB
3. **Default batch size** – 16KB
4. **Compression types** – gzip, snappy, lz4, zstd
5. **Default compression** – none
6. **Message key used for?** – Partitioning
7. **Default retention time** – 7days
8. **Retention unit** – ms
9. **Default number of partitions** – 1
10. **Default replication factor** – 1
11. **High availability through** – Replication
12. **Kafka stores messages on** – Disk
13. **Kafka is** – Distributed
14. **Leader election managed by** – Zookeeper
15. **Kafka replacement for Zookeeper** – KRaft
16. **Consumer offset stored in** – \_\_consumer\_offsets
17. **Offset type** – Long
18. **Consumer offset commit types** – Auto, Manual
19. **Consumer pull or push model?** – Pull
20. **Log compaction ensures** – LatestValue
21. **Kafka message format** – Key-Value
22. **Kafka durability via** – Replication
23. **Ordering guaranteed in** – Partition
24. **Kafka client API language** – Java
25. **Kafka protocol type** – TCP
26. **Kafka Connect is a** – DataPipeline
27. **Kafka Schema registry used with** – Avro
28. **Kafka Streams used for** – Processing
29. **Broker default port** – 9092
30. **ZooKeeper port** – 2181
31. **Kafka topic is a** – Log
32. **Consumer group enables** – Parallelism
33. **Producer acknowledgement setting** – acks
34. **acks=0 means** – NoAck
35. **acks=1 means** – LeaderAck
36. **acks=all means** – FullSync
37. **Kafka logs stored as** – Segments
38. **Kafka log file format** – Binary
39. **Kafka CLI tool** – kafka-topics.sh
40. **Kafka Connect runs as** – Worker
41. **Message timestamp stored as** – EpochMillis
42. **Kafka fault tolerance** – Replication
43. **Exactly-once semantics via** – Idempotence
44. **Enable idempotent producer** – true
45. **Partition assignment strategy** – Range, RoundRobin
46. **Kafka Java client class** – KafkaProducer
47. **Retention policy options** – Delete, Compact
48. **Kafka record is also known as** – Message
49. **Rebalance triggered on** – Join/Leave
50. **Tool to monitor consumer lag** – Burrow