General Kafka Questions:

22 May 2025 14:21

1.What is Apache Kafka, and why is it used?

- Answer: Kafka is a distributed event streaming platform designed for high-throughput, fault-tolerant messaging. It is used for real-time data streaming, log aggregation, and building event-driven architectures.

2.Can you explain Kafka's architecture?

-Answer: Kafka consists of four core components:

-Topics: Categories to which records are sent.

-Partitions: Subdivisions of topics for scalability and parallelism.

-Producers: Send records to topics.

-Consumers: Read records from topics.

-Zookeeper: (Optional in newer versions) Manages cluster metadata and leader election.

-Broker: A Kafka server that stores messages on disk.

3.What is a Kafka topic?

-Answer: A Kafka topic is a category or feed name to which messages are published by producers and from which consumers read. It is partitioned to allow scalability and parallelism.

4.What are Kafka partitions, and why are they important?

-Answer: Partitions are sub-segments of a topic, allowing parallel processing. Each partition is stored and replicated across brokers, enabling scalability and fault tolerance.

5.What is a replication factor in Kafka?

-Answer: The replication factor determines how many copies of a partition exist. A higher replication factor improves fault tolerance.

6.How does Kafka ensure message reliability?

-Answer: Kafka ensures reliability through replication, acknowledgments (ACKs), ISR (In-Sync Replicas), and producer retries. Messages are stored durably on disk.

Advanced Kafka Questions:

7.What is the role of Zookeeper in Kafka?

-Answer: Zookeeper manages metadata, tracks the status of brokers, and handles leader elections. Newer versions of Kafka have started moving away from Zookeeper.

8.What are Kafka consumer groups?

-Answer: A consumer group is a collection of consumers that work together to read data from a topic. Each partition in the topic is assigned to one consumer in the group.

9.How does Kafka handle offset management?

-Answer: Kafka stores offsets in a special topic, `__consumer offsets`. Consumers can manually commit offsets or let Kafka handle it automatically.

10.How do you handle high throughput in Kafka?
-Answer: By increasing the number of partitions, tuning producer and consumer configurations (e.g., batch size, linger.ms), and using efficient serialization formats like Avro or Protobuf.

11.What are ISR (In-Sync Replicas)?
-Answer: ISR are replicas of a partition that are fully caught up with the leader. Kafka only acknowledges writes that are replicated to all ISR members, ensuring fault tolerance.

Practical Kafka Questions:

12.How do you create a Kafka producer and consumer?
-Answer: Producers use Kafka APIs to send messages to a topic. Consumers subscribe to topics and process messages using consumer APIs. Both can be configured with properties like `acks`, `batch. Size`, and `auto.offset.reset`.

13.What are some common challenges in Kafka-based systems?
-Answer: Challenges include handling message ordering, offset management, backpressure, data retention policies, and tuning configurations for performance.

14.How do you monitor Kafka?
-Answer: By using tools like Kafka Manager, Conductor, Prometheus, Grafana, and Kafka's built-in metrics. Logs and monitoring APIs can also track health and performance.

15.What are Kafka Streams and how are they different from Kafka?
-Answer: Kafka Streams is a library for real-time stream processing of data from Kafka topics. Unlike Kafka, which focuses on message brokering, Kafka Streams allows you to process and transform the messages.

Integration with Other Systems:

16.How did you use Kafka with webMethods in your projects?
-Answer: I used Kafka producers and consumers to enable real-time streaming of data between webMethods services and external systems. Kafka was integrated with webMethods for asynchronous messaging and event-driven data flows.

17.What is the difference between JMS and Kafka?
-Answer: JMS is a standard API for message-oriented middleware and is often used in point-to-point or publish-subscribe models. Kafka, on the other hand, is a distributed streaming platform designed for scalability, high throughput, and durability.

18.How do you implement fault tolerance in Kafka?
-Answer: Fault tolerance is implemented through replication factors, ISR, retries, and producer acknowledgments. Kafka ensures no data loss even in case of broker failure.

19.What tools do you use for Kafka cluster management?
-Answer: Tools like Conduktor, Kafka Manager, and scripts provided by Kafka are used for cluster management. Docker and Kubernetes are used for deploying and scaling Kafka clusters.

Scenario-Based Questions:

20.How would you optimize Kafka for low-latency data processing?
-Answer: Reduce partition count (if appropriate), optimize producer batch size and linger.ms, adjust replication settings for low overhead, and use efficient serialization formats.

21.What happens if a Kafka broker goes down?
-Answer: The leader of a partition is reassigned to another ISR member. If the replication factor is sufficient, there is no data loss.

22.How do you handle a scenario where a Kafka topic is receiving duplicate messages?
-Answer: Enable idempotent producers and configure deduplication logic in consumers or storage systems.

23.How do you design a Kafka architecture for high availability?
-Answer: Use a replication factor of at least 3, deploy brokers across multiple availability zones, and enable ISR and producer retries.

24.What are retention policies in Kafka?
-Answer: Retention policies determine how long messages are stored. Kafka allows you to configure time-based (`retention.ms`) or size-based (`retention. Bytes`) retention.

General System Design

25.How do you design an event-driven system using Kafka?
-Answer: Use producers to publish events to Kafka topics. Consumers in different microservices subscribe to topics to process the events asynchronously. Use partitions for scalability and replication for fault tolerance.

26.What serialization formats have you used with Kafka?
-Answer: JSON, Avro, Protobuf, or custom formats depending on the use case. Avro is preferred for its compact size and schema evolution support.

27.How do you ensure message ordering in Kafka?
-Answer: By using a single partition or ensuring messages with the same key are routed to the same partition.

----

1. How will you handle a scenario with more consumers than partitions?
-Answer:
- In Kafka, each partition can only be assigned to a single consumer in a consumer group. If there are more consumers than partitions, the extra consumers remain idle because there are no partitions left to assign to them.
-Solutions:
  - Increase the number of partitions for the topic to match or exceed the number of consumers.
  - Redistribute consumers to other consumer groups to balance the load.
  - Consider the processing requirements—if all partitions are already fully utilized, adding consumers

will not improve throughput. In that case, optimize processing within existing partitions instead.

---

2. Explain Protobuf schema registry in detail.

-Answer:

-Protocol Buffers (Protobuf): A compact and efficient serialization format developed by Google. It is widely used in Kafka for serializing structured data before sending it to topics.

-Schema Registry: A centralized repository that manages Protobuf schemas and ensures compatibility between producers and consumers.

-Key Features of Schema Registry:

-Version Control: Tracks changes to schemas, enabling schema evolution without breaking existing consumers.

-Schema Compatibility: Validates whether new schemas are compatible with existing ones (backward, forward, or full compatibility).

-Centralized Storage: Stores schemas in a centralized system for easy access.

-Workflow with Protobuf:

- **Producers serialize data using a schema and send it to Kafka.**
- **Consumers retrieve the schema from the schema registry to deserialize the data.**
- **The schema registry ensures that only compatible changes are made to schemas, reducing the risk of serialization errors.**

-Benefits:

- **Efficient serialization and deserialization.**
- **Schema validation ensures data integrity.**
- **Reduces payload size compared to JSON or XML.**

---

3. Explain retention policies in Kafka in detail.

-Answer:

-Purpose: Retention policies determine how long Kafka stores messages in a topic, regardless of whether they have been consumed.

-Types of Retention Policies:

-Time-Based Retention (`retention.ms`): Messages are retained for a configured duration (e.g., 7 days).

-Size-Based Retention (`retention. Bytes`): Messages are retained until the total log size reaches a configured limit.

-Log Compaction (`cleanup.policy=compact`): Retains only the most recent message for a given key, useful for maintaining state.

-Configuration Options:

- **`log.retention.ms`: Time to retain messages.**
- **`log.retention.bytes`: Maximum size of logs before deletion.**
- **`log.segment.bytes`: Size of individual log segments. Kafka deletes segments that exceed the retention policy.**

-Scenarios:

- **For event-based systems, use time-based retention to ensure historical events are available.**
- **For stateful systems, use log compaction to maintain a compact, up-to-date dataset.**

-Impact of Retention Policies:

- **Messages exceeding retention limits are deleted automatically to free up disk space.**
- **Proper retention configuration ensures a balance between storage cost and data availability.**

---

4. What is the difference between Kafka's At-Least-Once and Exactly-Once semantics?

-Answer:

-At-Least-Once: Guarantees that messages are delivered at least once but may result in duplicates. Used when missing messages is unacceptable.

-Exactly-Once: Ensures messages are processed only once, even in failure scenarios. Achieved using Kafka's idempotent producers and transactions.

-Trade-offs: At-least-once has lower overhead and higher throughput, while exactly-once offers better data consistency at the cost of complexity.

---

5. How do you ensure high availability in a Kafka cluster?

-Answer:

-Replication Factor: Set a replication factor > 1 to replicate partitions across multiple brokers.

-ISR (In-Sync Replicas): Ensure all brokers in the ISR are up-to-date for failover.

-Multi-Data Centre Deployment: Use Mirror Maker or Confluent Replicator to replicate data across data centres.

-Zookeeper Quorum: Maintain a quorum of Zookeeper nodes for fault-tolerant cluster metadata management.

---

6. What is the difference between Consumer Offset Reset Policies?

-Answer:

-Earliest: Starts consuming messages from the beginning of the topic.

-Latest: Consumes messages that arrive after the consumer starts.

-None: Fails if no offset is stored.

-Use Case: "Earliest" is useful for processing historical data, while "Latest" is used for real-time processing.

---

7. What is the role of Kafka's Producer Acknowledgments (ACKs)?

-Answer:

- ACKs determine the guarantee level for message delivery:

  -acks=0: Producer does not wait for acknowledgment. Fast but unreliable.

  -acks=1: Waits for acknowledgment from the leader only.

  -acks=all: Waits for acknowledgment from all ISR replicas. Ensures the highest durability.

---

8. How would you optimize Kafka for high throughput?

-Answer:

- Increase the number of partitions.
- Tune producer configurations like `batch. Size`, `linger.ms`, and `compression. Type`.
- Use efficient serialization formats (e.g., Avro or Protobuf).
- Optimize consumer processing to match partition throughput.

---

9. What are Kafka Streams, and how are they different from Kafka?

-Answer:

-Kafka Streams: A library for processing and transforming streams of data within Kafka.

-Differences:

- Kafka focuses on message brokering, while Kafka Streams is used for processing messages in real-time.

- Kafka Streams allows stateful processing, windowing, and joins.

---

10. How does Kafka handle message ordering?

-Answer:

- Kafka ensures ordering within a partition. Messages with the same key are routed to the same partition, maintaining order for those keys.

---

11. How do you monitor and troubleshoot Kafka clusters?

-Answer:

- Use tools like Kafka Manager, Kafka-UI, Grafana, and Conductor.

- Monitor metrics like producer/consumer lag, broker disk usage, and ISR status.

- Analyse logs for broker errors or Zookeeper issues.