

Top 25 Kafka interview questions

1) What is Apache Kafka and why is it used?

Apache Kafka is a distributed streaming platform used for building real-time data pipelines and streaming applications.

2) How does Kafka differ from traditional messaging systems?

Kafka is designed for fault tolerance, high throughput, and scalability, unlike traditional messaging systems that may not handle large data streams efficiently.

3) What are Producers and Consumers in Kafka?

Producers publish messages to Kafka topics. Consumers read messages from topics.

```
// Producer
producer.send(new ProducerRecord<String, String>("topic", "key",
"value"));

// Consumer
consumer.subscribe(Arrays.asList("topic"));
```

4) What is a Kafka Topic?

A Topic is a category to which records are published by producers and from which records are consumed by consumers.

```
kafka-topics.sh --create --topic my_topic --bootstrap-server
localhost:9092
```

5) How does Kafka ensure durability and fault-tolerance?

Kafka replicates data across multiple brokers. Consumers read from leader replicas, and follower replicas synchronize data.

6) What is a Kafka Partition?

Partitions allow Kafka to horizontally scale as each partition can be hosted on a different server.

7) What is Zookeeper's role in a Kafka ecosystem?

Zookeeper manages brokers, maintains metadata, and helps in leader election for partitions.

8) How can you secure Kafka?

Kafka can be secured using SSL for encryption, SASL for authentication, and ACLs for authorization.

9) What is Kafka Streams?

Kafka Streams is a client library for building real-time, highly scalable, fault-tolerant stream processing applications.

```
KStream<String, String> stream = builder.stream("input-topic");  
stream.to("output-topic");
```

10) What are some use-cases for Kafka?

Kafka is used for real-time analytics, data lakes, aggregating data from different sources, and acting as a buffer to handle burst data loads.

11) How do you integrate Kafka with Spring Boot?

You can use the Spring Kafka library, which provides `@KafkaListener` for consumers and `KafkaTemplate` for producers.

```
@KafkaListener(topics = "myTopic")  
public void listen(String message) {
```

```
// Handle message  
}
```

12) How do you send a message to a Kafka topic using Spring Kafka?

Use `KafkaTemplate` to send messages.

```
kafkaTemplate.send("myTopic", "myMessage");
```

13) How do you consume messages from a Kafka topic in Spring?

Use the `@KafkaListener` annotation to mark a method as a Kafka message consumer.

```
@KafkaListener(topics = "myTopic")  
public void consume(String message) {  
    // Process message  
}
```

14) How do you handle message deserialization errors in Spring Kafka?

Use the `ErrorHandlingDeserializer` to wrap the actual deserializer and catch deserialization errors.

15) How do you ensure ordered message processing in Spring Kafka?

Set the `concurrency` property of `@KafkaListener` to 1 to ensure single-threaded consumption for each partition.

```
@KafkaListener(topics = "myTopic", concurrency = "1")
```

16) How do you batch-process messages from Kafka in Spring?

Use the `@KafkaListener` annotation with the `batchListener` property set to `true`.

```
@KafkaListener(topics = "myTopic", batchListener = true)
public void consume(List<String> messages) {
    // Process messages
}
```

17) How do you filter messages in Spring Kafka?

Implement a `RecordFilterStrategy` to filter out unwanted records before they reach the `@KafkaListener`.

Create a class that implements `RecordFilterStrategy`:

```
import org.apache.kafka.clients.consumer.ConsumerRecord;
import
org.springframework.kafka.listener.adapter.RecordFilterStrategy;

public class MyRecordFilterStrategy implements
RecordFilterStrategy<String, String> {

    @Override
    public boolean filter(ConsumerRecord<String, String>
consumerRecord) {
        // Return true to filter out the record, false to include it
        return !consumerRecord.value().contains("important");
    }
}
```

Now, configure your `ConcurrentKafkaListenerContainerFactory` to use this filter:

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
```

```

org.springframework.kafka.config.ConcurrentKafkaListenerContainerFact
ory;
import org.springframework.kafka.core.ConsumerFactory;

@Configuration
public class KafkaConsumerConfig {

    @Bean
    public ConcurrentKafkaListenerContainerFactory<String, String>
kafkaListenerContainerFactory(
        ConsumerFactory<String, String> consumerFactory) {
        ConcurrentKafkaListenerContainerFactory<String, String>
factory = new ConcurrentKafkaListenerContainerFactory<>();
        factory.setConsumerFactory(consumerFactory);
        factory.setRecordFilterStrategy(new
MyRecordFilterStrategy());
        return factory;
    }
}

```

Finally, use the `@KafkaListener` annotation to consume messages:

```

import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Service;

@Service
public class MyKafkaConsumer {

    @KafkaListener(topics = "myTopic")
    public void consume(String message) {
        System.out.println("Consumed message: " + message);
    }
}

```

```
}  
  
}
```

18) How do you handle retries for message processing in Spring Kafka?

Configure a `SeekToCurrentErrorHandler` or implement a custom error handler to manage retries.

19) How can you produce and consume Avro messages in Spring Kafka?

Use the Apache Avro serializer and deserializer along with Spring Kafka's `KafkaTemplate` and `@KafkaListener`.

20) How do you secure Kafka communication in a Spring application?

Configure SSL properties in the `application.yml` or `application.properties` file for secure communication.

```
spring.kafka.properties.security.protocol: SSL
```

21) What are the key differences between Spring AMQP and Spring Pub-Sub?

Spring AMQP is based on the AMQP protocol and is often used with RabbitMQ. It supports complex routing and is suitable for enterprise-level applications. Spring Pub-Sub is generally used with messaging systems like Kafka and is more geared towards high-throughput data streaming.

22) How do message delivery semantics differ between Spring AMQP and Spring Pub-Sub?

Spring AMQP provides more granular control over message acknowledgment and transactions. Spring Pub-Sub, especially with Kafka, focuses on high-throughput and allows at-least-once, at-most-once, and exactly-once semantics.

Configure the producer for exactly-once semantics by setting the `transactional.id` and `acks` properties:

```
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;

public class ExactlyOnceProducer {
    public static void main(String[] args) {
        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
"org.apache.kafka.common.serialization.StringSerializer");
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
"org.apache.kafka.common.serialization.StringSerializer");
        props.put(ProducerConfig.TRANSACTIONAL_ID_CONFIG, "my-
transactional-id");
        props.put(ProducerConfig.ACKS_CONFIG, "all");

        Producer<String, String> producer = new KafkaProducer<>
(props);
        producer.initTransactions();

        try {
            producer.beginTransaction();
            for (int i = 0; i < 100; i++) {
                producer.send(new ProducerRecord<>("my-topic",
```

```

Integer.toString(i), Integer.toString(i)));
        }
        producer.commitTransaction();
    } catch (Exception e) {
        producer.abortTransaction();
    }
    producer.close();
}
}

```

Configure the consumer to read committed messages:

```

java
Copy code
import org.apache.kafka.clients.consumer.Consumer;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;

import java.time.Duration;
import java.util.Collections;
import java.util.Properties;

public class ExactlyOnceConsumer {
    public static void main(String[] args) {
        Properties props = new Properties();
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092");
        props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-group");
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
"org.apache.kafka.common.serialization.StringDeserializer");
    }
}

```



```

        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put(ConsumerConfig.ISOLATION_LEVEL_CONFIG,
"read_committed");

        Consumer<String, String> consumer = new KafkaConsumer<>
(props);
        consumer.subscribe(Collections.singletonList("my-topic"));

        while (true) {
            ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(100));
            records.forEach(record -> {
                System.out.printf("Consumed record with key %s and
value %s\n", record.key(), record.value());
            });
        }
    }
}

```

23) How do you handle message ordering in Spring AMQP and Spring Pub-Sub?

In Spring AMQP, message ordering is generally maintained by the queue. In Spring Pub-Sub with Kafka, message ordering is guaranteed within a partition but not across partitions.

24) How can you configure dead letter queues (DLQ) in Kafka and Spring?

Kafka itself doesn't provide DLQs, but you can configure Spring Kafka to handle failed messages and send them to a DLQ.


Define a Kafka listener with error handling:

```
@KafkaListener(topics = "myTopic")
public void listen(String message) {
    try {
        // Process message
    } catch (Exception e) {
        // Send to DLQ
        kafkaTemplate.send("myDLQ", message);
    }
}
```

25) What is a Kafka Connector?

A Kafka Connector is a tool for connecting Kafka with other systems, such as databases, through source and sink connectors.

Feel free to reach out or mail me at abilashs.work@gmail.com, for help with Interview preparation or resume reviews.

Do follow [@Abilash](#) Shankarpalli for more such posts 

Happy Learning!