Solved Projects Customer Reviews Custom Project Path New Blog Write for ProjectPro End to End Projects

100+ Kafka Interview Questions and Answers for 2022

Last Updated: 30 Jun 2022

Your search for Apache Kafka interview questions ends right here! This blog brings you the most popular Kafka interview questions and answers divided into various categories such as Apache Kafka interview questions for beginners, Advanced Kafka interview questions/Apache Kafka interview questions for experienced, Apache Kafka Zookeeper interview questions, etc.

Let us now dive directly into the Apache Kafka interview questions and answers and help you get started with your Big Data interview preparation!

Table of Contents

Relevant Projects

Machine Learning Projects

Data Science Projects

Python Projects for Data Science

Top 100+ Kafka Interview Questions and Answers

Suppose you are a student, a fresher in the industry, or an experienced IT professional making a career transition into analytics. In that case, this blog on the most popular 100+ Apache Kafka interview questions and answers will help you nail your next big data job interview.

The following sections will walk you through the important Apache Kafka interview questions and answers subdivided into different categories based on experience levels, specific topics, and Apache Kafka job roles.

- Data Science Projects in R
- Machine Learning Projects for Beginners
- Deep Learning Projects
- Neural Network Projects
- Tensorflow Projects
- NLP Projects
- Kaggle Projects
- IoT Projects
- Big Data Projects
- Hadoop Real-Time Projects Examples
- Spark Projects
- Data Analytics Projects for Students

You might also like

- Data Scientist Salary
- How to Become a Data Scientist
- Data Analyst vs Data Scientist
- Data Scientist Resume
- Data Science Projects for Beginners
- Machine Learning Engineer
- Machine Learning Projects for Beginners
- Datasets
- Pandas Dataframe
- Machine Learning Algorithms
- Regression Analysis

Apache Kafka Interview Questions for Beginners

This section comprises the basic yet commonly asked Apache Kafka interview questions. These Kafka interview questions mostly revolve around the fundamental components of Apache Kafka, such as topics, partitions, consumer group, load balancing, Kafka APIs, etc.

1. What are topics in Apache Kafka?

A stream of messages that belong to a particular category is called a topic in Kafka. Kafka stores data in topics that are split into partitions.

2. Explain partitions in Apache Kafka.

Topics in Kafka are divided into partitions. One or more consumers can read data from a Kafka topic simultaneously by reading from each partition. The partitions are separated in order. While creating a topic, you need to specify the number of partitions, although this number is arbitrary and can be changed later.

3. How are partitions distributed in an Apache Kafka cluster?

Partitions of a Kafka topic are distributed across servers in a Kafka cluster. Each Kafka server handles the data and requests with its share of partitions. Partitions can be replicated across multiple servers to ensure fault tolerance. Every partition has one Kafka server that plays the role of a leader for that partition. The leader takes care of all the read and write requests for that particular partition. A leader can have zero or more followers. The leader and follower relationship is such that the followers passively replicate the leader. In the case where the leader fails, one of the followers can take on the role of the leader. This is how the leader and follower concept works in a Kafka cluster.

4. What are consumers in Apache Kafka?

Consumers read data from the brokers. Consumers can subscribe to one or more topics and receive published messages from these topics by pulling data from the brokers. Consumers pull the data at their own pace.

5. What are producers in Apache Kafka?

Producers can publish messages on one or more Kafka topics. Producers send data to the Kafka brokers. Whenever a producer publishes messages to the broker, the broker appends the published messages to a partition. The producer can also send messages to a partition of their choice.

6. What is a broker in Apache Kafka?

A Kafka cluster contains one or more servers that are known as brokers. A broker works as a container that holds multiple topics with various partitions. A broker in a cluster can only be identified by the integer ID associated with it. Connection with any one broker in a cluster implies a connection with the whole cluster. Brokers in Kafka do not contain the complete data, but they know about other brokers, topics, and partitions of the cluster.

MNIST Dataset

Data Science Interview Questions

Python Data Science Interview Questions

Spark Interview Questions

Hadoop Interview Questions

Data Analyst Interview Questions

Machine Learning Interview Questions

AWS vs Azure

Hadoop Architecture

Spark Architecture

Tutorials

Data Science Tutorial

Snowflake Data Warehouse Tutorial for Beginners with Examples

Jupyter Notebook Tutorial - A Complete Beginners Guide

Best Python NumPy Tutorial for Beginners

Tableau Tutorial for Beginners - Step by Step Guide

MLOps Python Tutorial for Beginners -Get Started with MLOps

Alteryx Tutorial for Beginners to Master Alteryx in 2021

Free Microsoft Power BI Tutorial for Beginners with Examples

Theano Deep Learning Tutorial for Beginners

7. What is a consumer group in Apache Kafka?

In Kafka, a consumer group is a set of one or more consumers who cooperate to consume data from the same topic or the same set of topics. A consumer group basically represents the name of an application. Consumers in Kafka generally belong to a particular consumer group. To consume messages from a consumer group, '-group' command has to be used.

8. Mention the APIs provided by Apache Kafka.

There are four main APIs provided by Apache Kafka-

- Kafka Producer API: The producer API allows applications to publish messages in the form of a stream of records to one or more Kafka topics.
- Kafka Consumer API: The consumer API allows applications to subscribe to one or more Kafka topics. The consumer API also allows applications to process streams of messages that are produced for those topics.
- Kafka Streams API: The Kafka streams API allows applications to process data in a stream processing paradigm. The application can fetch data in the form of input streams for one or more topics, process these streams, and then send output streams to one or more topics.
- Kafka Connector API: The connector API helps connect applications to Kafka topics. It provides features for managing the running of producers and consumers and handling the connections between them.

9. What is meant by ZooKeeper in Apache Kafka?

The ZooKeeper in Kafka is responsible for managing and coordinating the Kafka cluster. It ensures coordination between different nodes in a cluster. When there is any change in the topology of the Kafka cluster, the ZooKeeper notifies all the nodes of these changes. The changes include when brokers or topics are removed or added.

10. Can Kafka be used without a ZooKeeper?

It is not possible to bypass the ZooKeeper in Kafka and connect directly to the Apache Kafka Server. Hence, the answer is no. If, for any reason, the ZooKeeper is down, it will not be possible to service any client requests.

11. How is load balancing maintained in Kafka?

Load balancing in Kafka is handled by the producers. The message load is spread out between the various partitions while maintaining the order of the message. By default, the producer selects the next partition to take up message data using a round-robin approach. If a different approach other than round-robin is to be used, users can also specify exact partitions for a message.

Computer Vision Tutorial for Beginners | Learn Computer Vision

Python Pandas Tutorial for Beginners - The A-Z Guide

NumPy Python Tutorial for Beginners

Hadoop Online Tutorial – Hadoop HDFS Commands Guide

MapReduce Tutorial–Learn to implement Hadoop WordCount Example

Hadoop Hive Tutorial-Usage of Hive Commands in HQL

Hive Tutorial-Getting Started with Hive Installation on Ubuntu

Learn Java for Hadoop Tutorial: Inheritance and Interfaces

Learn Java for Hadoop Tutorial: Classes and Objects

Learn Java for Hadoop Tutorial: Arrays

Apache Spark Tutorial - Run your First Spark Program

Best PySpark Tutorial for Beginners-Learn Spark with Python

R Tutorial- Learn Data Visualization with R using GGVIS

Neural Network Training Tutorial

Python List Tutorial

MatPlotLib Tutorial

Decision Tree Tutorial

Neural Network Tutorial

Performance Metrics for Machine Learning Algorithms

R Tutorial: Data.Table

SciPy Tutorial

Step-by-Step Apache Spark Installation Tutorial

12. What are some differences between Apache Kafka and Flume?

Apache Kafka and Flume are distributed data systems, but there is a certain difference between Kafka and Flume in terms of features, scalability, etc. The below table lists all the major differences between Apache Kafka and Flume-

Apache Kafka	Apache Flume
Kafka is optimized to ingest data and process streaming data in real-time.	Flume is mainly used for collecting and aggregating large amounts of log data from multiple sources to a centralized data location.
Easy to scale.	Not as easy to scale as Kafka.
It can be supported across various applications.	Specifically designed for Hadoop.
Apache Kafka runs as a cluster and supports automatic recovery if resilient to node failure.	Tool to collect log data from distributed web servers.

13. What is the maximum size of a message that can be received by Apache Kafka?

The maximum size for a Kafka message, by default, is 1MB (megabyte). The size can be changed in the broker settings. However, Kafka is optimized to handle smaller messages of the size of 1KB.

14. Explain the retention period in an Apache Kafka cluster.

Messages sent to Kafka clusters get appended to one of the multiple partition logs. These messages remain in multiple partition logs even after being consumed, for a configurable period of time, or until a configurable size is reached. This configurable amount of time for which the message remains in the log is known as the retention period. The message will be available for the amount of time specified by the retention period. Kafka allows users to configure the retention period for messages on a per-topic basis. The default retention period for a message is seven days.

Introduction to Apache Spark Tutorial

R Tutorial: Importing Data from Web

R Tutorial: Importing Data from Relational Database

R Tutorial: Importing Data from Excel

Introduction to Machine Learning Tutorial

Machine Learning Tutorial: Linear Regression

Machine Learning Tutorial: Logistic Regression

Support Vector Machine Tutorial (SVM)

K-Means Clustering Tutorial

dplyr Manipulation Verbs

Introduction to dplyr package

Importing Data from Flat Files in R

Principal Component Analysis Tutorial

Pandas Tutorial Part-3

Pandas Tutorial Part-2

Pandas Tutorial Part-1

Tutorial- Hadoop Multinode Cluster Setup on Ubuntu

Data Visualizations Tools in R

R Statistical and Language tutorial

Introduction to Data Science with R

Apache Pig Tutorial: User Defined Function Example

Apache Pig Tutorial Example: Web Log Server Analytics

Impala Case Study: Web Traffic

Impala Case Study: Flight Data Analysis

Hadoop Impala Tutorial

15. How long are messages retained in Apache Kafka?

Messages sent to Kafka are retained regardless of whether they are published or not for a specific period that is referred to as the retention period. The retention period can be configured for a topic. The default retention time is 7 days.

16. What is the role of the Partitioning Key?

Messages are sent to various partitions associated with a topic in a round-robin fashion. If there is a requirement to send a message to a particular partition, then it is possible to associate a key with the message. The key determines which partition that particular message will go to. All messages with the same key will go to the same partition. If a key is not specified for a message, the producer will choose the partition in a round-robin fashion.

17. When does QueueFullException occur in the Producer API?

The QueueFullException occurs when the producer sends messages to the broker at a pace that the broker cannot handle. A solution here is to add more brokers to handle the pace of messages coming in from the producer.

18. Explain the roles of leader and follower in Apache Kafka.

Every partition in the Kafka server has one server that plays the role of a leader. The leader performs all the read and write data tasks for a partition. A partition can have no followers, one follower, or more than one follower. The job of the follower is to replicate the leader. In such a case, if there is a failure in the leader, then one of the followers can take on the leader's load.

19. What is the role of replicas in Apache Kafka?

Replicas are the backups for partitions in Kafka. They are never actually read or written; rather, they are used to prevent data loss in case of failure. The partitions of a topic are published to several servers in an Apache cluster.

- Apache Hive Tutorial: Tables
- Flume Hadoop Tutorial: Twitter Data Extraction
- Flume Hadoop Tutorial: Website Log Aggregation
- Hadoop Sqoop Tutorial: Example Data Export
- Hadoop Sqoop Tutorial: Example of Data Aggregation
- Apache Zookepeer Tutorial: Example of Watch Notification
- Apache Zookepeer Tutorial: Centralized Configuration Management
- Hadoop Zookeeper Tutorial for Beginners
- Hadoop Sqoop Tutorial
- Hadoop PIG Tutorial
- Hadoop Oozie Tutorial
- Hadoop NoSQL Database Tutorial
- Hadoop Hive Tutorial
- Hadoop HDFS Tutorial
- Hadoop hBase Tutorial
- Hadoop Flume Tutorial
- Hadoop 2.0 YARN Tutorial
- Hadoop MapReduce Tutorial
- Big Data Hadoop Tutorial for Beginners- Hadoop Installation

Top 15 Latest Recipes

- Explain the features of Amazon Personalize
- Introduction to Amazon Personalize and its use cases
- Explain the features of Amazon Nimble Studio

There is one Kafka server that is considered to be the leader for that partition.

The leader handles all reads and writes for a particular partition. There can be none or more followers in the cluster, where the partitions of the topics get replicated. In the event of a failure in the leader, the data is not lost because of the presence of replicas in other servers. In addition, one of the followers will take on the role of the new leader.

20. What is the purpose of ISR in Apache Kafka?

ISR - in-synch replicas refers to all the replicated partitions that are completely synced up with the leader. A replica has to be fully caught up with the leader within a configurable amount of time. By default, this time is 10 seconds. After this period of time, if a follower is not caught up with the leader, the leader will drop the follower from its ISR and writes will continue on the remaining replicas in the ISR. If the follower comes back, it will first truncate its log to the last point, which was checked, and then catch up on all the messages after the last checkpoint from the leader. Only when the follower fully catches up will the leader add it back to the ISR.

21. What is meant by partition offset in Apache Kafka?

Every time message or record is assigned to a partition in Kafka, it is provided with an offset. The offset denotes the position of the record in that partition. A record can be uniquely identified within a partition using the offset value. The partition offset only carries meaning within that particular partition. Records are always added to the ends of partitions, and therefore, older records will have a lower offset.

22. Explain fault tolerance in Apache Kafka.

In Kafka, the partition data is copied to other brokers, which are known as replicas. If there is a point of failure in the partition data in one node, then there are other nodes that will provide a backup and ensure that the data is still available. This is how Kafka allows fault tolerance.

23. What is the importance of replication in Kafka?

In Kafka, replication provides fault tolerance by ensuring that published messages are not permanently lost. Even if a node fails and they are lost on one node due to program error, machine error, or even due to software upgrades, then there is a replica present on another node that can be recovered.

24. What is the best way to start the Kafka server?

Once you download the latest version of Apache Kafka, remember to extract it.

To run Kafka, remember that your local environment must have Java 8+ installed on it.

If you want to start the Kafka server, the following commands have to be run in order so that all the services can be started in the correct order:

Introduction to Amazon Nimble Studio and its use cases

Explain the features of Amazon Neptune

Introduction to Amazon Neptune and its use cases

Explain the features of Amazon MQ

Introduction to Amazon MQ and its use cases

Explain the features of Amazon Monitron for Redis

Introduction to Amazon Monitron and its use cases

Explain the features of Amazon MemoryDB for Redis

Introduction to Amazon MemoryDB for Redis and its use cases

Explain the features of Amazon Grafana

Introduction to Amazon Managed Grafana and its use cases

Explain the features of Amazon Managed Blockchain

Start the ZooKeeper service:

```
$bin/zookeeper-server-start.sh config/zookeeper.properties
```

Open another terminal and run the following to start the Kafka broker service:

```
$ bin/kafka-server-start.sh config/server.properties
```

25. What is Geo-Replication in Kafka?

Geo-Replication in Kafka is a process by which you can duplicate messages in one cluster across other data centers or cloud regions. Geo-replication involves copying all the files and allows them to be stored across the globe if required. In Kafka, Geo-replication can be achieved by using Kafka's MirrorMaker Tool. Geo-replication is a way to ensure that the data is backed up.

26. What is meant by multi-tenancy in Apache Kafka?

Multi-tenancy refers to the mode of operation of software where there are multiple instances of one or multiple applications operating in a shared environment independent from each other. The instances are said to be logically isolated but physically integrated. For a system to support multi-tenancy, the level of logical isolation must be complete, but the level of physical integration may vary. Kafka can be said to be multi-tenant as it allows for configuring different topics for which data can be consumed or produced on the same cluster.

27. Explain the topic replication factor.

Topic replication factor refers to the number of copies of the topic that are present over multiple brokers. The replication factor should be greater than 1 for fault tolerance. In such cases, there will be a replica of the data in another broker from where the data can be retrieved if necessary.

28. Differentiate between partitions and replicas in a Kafka cluster.

- In Kafka, topics are divided into parts which are called partitions. Partitions allow one or more consumers to read data from servers in parallel. Read and write responsibility for one particular partition is managed on one server, called the leader for that partition. A cluster may have zero or more followers in which replicas of the data will be created. Replicas are merely copies of the data in a particular partition. The followers do not have to read or write the partitions separately; rather, they just copy the leader.
- Partitions in Kafka are used to increase throughput. Replicas ensure fault tolerance.

29. Mention some disadvantages of Apache Kafka.

- Tweaking of messages in Kafka causes performance issues in Kafka. Kafka works well in cases where the message does not need to be changed.

- Kafka does not support wildcard topic selection. The exact topic name has to be matched.
- In the case of large messages, brokers and consumers reduce the performance of Kafka by compressing and decompressing the messages. This affects the throughput and performance of Kafka.
- Kafka does not support certain message paradigms like a point-to-point queue and client request/reply.

30. Mention some real-world use cases of Apache Kaka.

- Message Broker: Kafka is capable of appropriate metadata handling, i.e., a large volume of similar types of messages or data, due to its high throughput value. Kafka can be used as a publish-subscribe messaging system that allows data to be read and written conveniently.
- Monitor Operational Data: Kafka can be used to monitor the metrics associated with certain technologies, such as security logs.
- Tracking website activities: Kafka can be used to ensure that data is successfully sent and received by websites. Kafka can handle the huge amount of data generated by websites for each particular page and users' activities.
- Data logging: Kafka's feature of data replication across the nodes can be used to restore data on failed nodes. Kafka also offers a replicated log service across multiple sources and makes replicated data available to the customers.
- Kafka for Stream Processing: Kafka can handle streaming data wherein data is read from a topic, processed, and written to a new topic. The new topic containing the processed data will be available to users and applications.

31. Why is Apache Kafka preferred over traditional messaging techniques?

- Unlike the traditional message transfer method, Apache Kafka is more scalable as it allows the addition of more partitions.
- Kafka does not slow down with the addition of new consumers, unlike the traditional method of message transfer, where both the queue and topic performance sees a decline in performance with a rise in the number of consumers.
- In Kafka, the messages contain a key-value pair. The key is used for partitioning purposes and to place related messages in the same partition. The traditional message transfer method usually does not have any such method of grouping messages.
- Apache Kafka comes with a checksum method that is used to detect corruption of messages on the various servers; the traditional method of message transfer has no such way of verifying whether message integrity is maintained.

integrity is maintained.

- Apache Kafka supports the creation of message replicas, i.e., messages in Kafka are not deleted once consumed and are available for a while as specified by the retention time. This also makes it possible for consumers to retrieve any message once more and reprocess it. In any traditional messaging system, the broker would either delete a successfully processed message or try to re-deliver an unprocessed one which could cause performance degradation due to messages getting stuck in the queue.

32. Mention some of the system tools available in Apache Kafka.

The three main system tools available in Apache Kafka are:

Kafka Migration Tool − This tool is used to migrate the data in a Kafka cluster from one version to another.

Kafka MirrorMaker − This tool copies data from one Kafka cluster to another.

Consumer Offset Checker − This tool displays the Consumer Group, Topic, Partitions, Off-set, logSize, and Owner for specific Topics and Consumer Group sets.

33. Mention some benefits of Apache Kafka.

- High throughput: Kafka can handle thousands of messages per second. Due to low latency, Kafka supports incoming messages at a high volume and velocity.
- Low latency: Apache Kafka offers as low as ten milliseconds. This is because it decoupled messages on the broker, allowing the consumer to pull them at any time.
- Fault-tolerant: The use of replicas allows Apache Kafka to be fault-tolerant in cases of a failure within a cluster.
- Durability: With the replication feature, Apache Kafka allows data to remain more persistent on the cluster rather than the disk, thus making it more durable.
- Scalability: the ability of Kafka to handle messages of high volume and high velocity makes it very scalable.
- Ability to handle real-time data: Kafka can handle real-time data pipelines.

34. What method does Apache Kafka use to connect with clients and servers?

Apache Kafka uses a basic, high-performance, language-agnostic TCP protocol to facilitate communication between clients and servers. Backward compatibility exists between this protocol and its predecessor.

35. Define the role of Kafka Streams API and Kafka Connector API.

Streams API enables an application to work as a stream processor by efficiently changing input streams into output streams. Streams API is responsible for receiving input streams from one or more topics and sending output streams to one or more output topics.

Connector API Connects Kapfka topics to applications. The connector API enables the execution and building of reusable producers or consumers that link Kafka topics to pre-existing applications or data systems.

Advanced Kafka Interview Questions | Kafka Interview Questions for Experienced

The following set of Apache Kafka interview questions is ideal for those professionals who have a few years of work experience in the industry and are looking for a promotion to a senior role. These Kafka interview questions cover the concepts of Apache Kafka in depth.

36. What is meant by the Replication Tool?

The Replication Tool in Kafka is used for a high-level design to maintain Kafka replicas. Some of the replication tools available are

1. Preferred Replica Leader Election Tool: Partitions are distributed to multiple brokers in a cluster, each copy known as a replica. The preferred replica usually refers to the leader. The brokers distribute the leader role evenly across the cluster for various partitions. Still, an imbalance can occur over time due to failures, planned shutdowns, etc. in such cases, you can use the replication tool to maintain the load balancing by reassigning the preferred replicas and hence, the leaders.

2. Topics tool: Kafka topics tool is responsible for handling all management operations related to topics, which include

- Listing and describing topics
- Creating topics
- Changing topics
- Adding partitions to a topic
- Deleting topics

3. Reassign partitions tool: This tool changes the replicas assigned to a partition. This means adding or removing followers associated with a partition.

4. StateChangeLogMerger tool: This tool is used to collect data from the brokers in a particular cluster, formats it into a central log, and help to troubleshoot issues with state changes. Often, problems may arise with the leader election for a particular partition. This tool can be used to determine what caused the problem.

5. Change topic configuration tool: used to Add new config options, Change existing config options, and Remove config options

37. How can Kafka be tuned for optimal performance?

Tuning for optimal performance involves consideration of two key measures: latency measures, which denote the amount of time taken to process one event, and throughput measures, which refer to how many events can be processed in a specific time. Most systems are optimized for either latency or throughput, while Kafka can balance both. Tuning Kafka for optimal performance involves the following steps:

- Tuning Kafka producers: Data that the producers have to send to brokers is stored in a batch. When the batch is ready, the producer sends it to the broker. For latency and throughput, to tune the producers, two parameters must be taken care of: batch size and linger time. The batch size has to be selected very carefully. If the producer is sending messages all the time, a larger batch size is preferable to maximize throughput. However, if the batch size is chosen to be very large, then it may never get full or take a long time to fill up and, in turn, affect the latency. Batch size will have to be determined, taking into account the nature of the volume of messages sent from the producer. The linger time is added to create a delay to wait for more records to get filled up in the batch so that larger records are sent. A longer linger time will allow more messages to be sent in one batch, but this could compromise latency. On the other hand, a shorter linger time will result in fewer messages getting sent faster - reduced latency but reduced throughput as well.
- Tuning Kafka broker: Each partition in a topic is associated with a leader, which will further have 0 or more followers. It is important that the leaders are balanced properly and ensure that some nodes are not overworked compared to others.
- Tuning Kafka Consumers: It is recommended that the number of partitions for a topic is equal to the number of consumers so that the consumers can keep up with the producers. In the same consumer group, the partitions are split up among the consumers.

38. What are the differences between Redis and Kafka?

Redis is the short form for remote dictionary servers. It is a key-value store and can be used as a repository to read and write requests. Redis is a no-SQL database.

Redis	Apache Kafka
Redis supports push-based message delivery. This means that messages published to Redis will be automatically delivered to the consumers.	Apache Kafka supports the pull-based delivery of messages. The messages published to the Kafka broker are not delivered automatically to the consumers; rather, the consumers have to pull the messages when they are ready for them.

Redis does not support parallel processing.	Due to the partitioning system in Apache Kafka, one or more consumers of a specific consumer group can parallelly consume partitions of the topic at the same time.
Redis does not support message replicas. Once the messages are delivered to the consumers, they are deleted.	Apache Kafka supports creating message replicas in its log.
Redis is an in-memory store, which makes it faster than Kafka.	Apache Kafka uses disk space for storage, which makes it slower than Redis.
Since Redis is an in-memory store, it cannot handle large volumes of data.	Since Kafka uses disk space as its primary storage, it is capable of handling large volumes of data.

39. Is it possible to add partitions to an existing topic in Apache Kafka?

Apache Kafka provides the alter command to change a topic behavior and modify the configurations associated with it. The alter command can be used to add more partitions.

The command to increase the partitions to four is as follows:

```
./bin/kafka-topics.sh --alter --zookeeper localhost:2181 --topic my-topic --partitions 4
```

40. What is the optimal number of partitions for a topic?

The optimal number of partitions a topic should be divided into must be equal to the number of consumers.

41. How does one view a Kafka message?

The Kafka-console-consumer.sh command can be used to view the messages. The following command can be used to view the messages from a topic:

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning
```

42. How can all brokers available in a cluster be listed?

Two ways to get the list of available brokers in an Apache Kafka cluster are as follows:

- Using zookeeper-shell.sh

```
zookeeper-shell.sh :2181 ls /brokers/ids
```

Which will give an output like:

```
WATCHER::  WatchedEvent state:SyncConnected type:None path:null [0, 1, 2,
```

3]

This indicates that there are four alive brokers - 0,1,2 and 3

- Using zkCli.sh

  First, you have to log in to the ZooKeeper client

  zkCli.sh -server :2181

Then use the below command to list all the available brokers

ls /brokers/ids

Both the methods used above make use of the ZooKeeper to find out the list of available brokers.

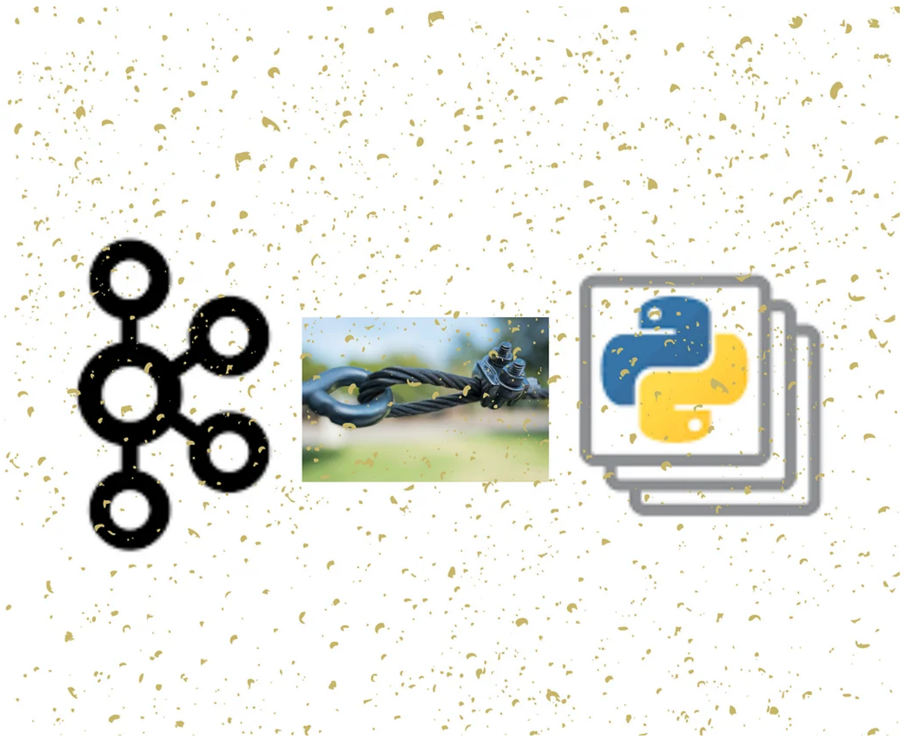
43. What is the Kafka MirrorMaker?

The Kafka MirrorMaker is a stand-alone tool that allows data to be copied from one Apache Kafka cluster to another. The Kafka MirrorMaker will read data from topics in the original cluster and write the topics to a destination cluster with the same topic name. The source and destination clusters are independent entities and can have different numbers of partitions and varying offset values.

44. What is the role of the Kafka Migration Tool?

The Kafka Migration tool is used to efficiently move from one environment to another. It can be used to move existing Kafka data from an older version of Kafka to a newer version.

45. How can Apache Kafka be used with Python?



There are several [libraries available in Python](https://www.projectpro.io) which allow access to Apache Kafka:

- Kafka-python: an open-source community-based library.
- PyKafka: maintained by Parsly, and claimed to be a 'Pythonic' API.
- Confluent Python Kafka has the best performance among the three. It is offered by Confluent as a wrapper around 'librdkafka' (C library implementation of Kafka).

46. How can you list the topics being used in Apache Kafka?

Once you start the ZooKeeper, you can list all the topics using

bin/kafka-topics.sh --list --zookeeper localhost:2181

47. What are name restrictions for Kafka topics?

According to Apache Kafka, there are some legal rules to be followed to name topics, which are as follows:

- The maximum length is 255 characters (symbols and letters). The length has been reduced from 255 to 249 in Kafka 0.10
- . (dot), _ (underscore), - (hyphen) can be used. However, topics with a dot (.) and underscore (_) could cause some confusion with internal data structures, and hence, it is advisable to use either but not both.

48. What is the Confluent Replicator?

The Confluent Replicator allows easy and reliable replication of topics from a source cluster to a destination cluster. It continuously copies messages from the source to the destination and even assigns the same names to the topics in the destination cluster.

49. How can load balancing be ensured in Apache Kafka when one Kafka fails?

When a Kafka server fails, if it was the leader for any partition, one of its followers will take on the role of being the new leader to ensure load balancing. For this to happen, the topic replication factor has to be more than one, i.e., the leader should have at least one follower to take on the new role of leader.

50. Where is the meta-information about topics stored in the Kafka cluster?

Currently, in Apache Kafka, meta-information about topics is stored in the ZooKeeper. Information regarding the location of the partitions and the configuration details related to a topic are stored in the ZooKeeper in a separate Kafka cluster.

51. How can large messages be sent in Apache Kafka?

By default, the largest size of a message that can be sent in Apache Kafka is 1MB. In order to send larger messages using Kafka, a few properties have to be adjusted. Here are the configuration details that have to be updated

- At the Consumer end – fetch.message.max.bytes
- At the Broker, end to create replica– replica.fetch.max.bytes
- At the Broker, the end to create a message – message.max.bytes
- At the Broker end for every topic – max.message.bytes

52. Explain the scalability of Apache Kafka.

In software terms, the scalability of an application is its ability to maintain its performance when it is exposed to changes in application and processing demands. In Apache Kafka, the messages corresponding to a particular topic are divided into partitions. This allows the topic size to be scaled beyond the size that will fit on a single server. Allowing a topic to be divided into partitions ensures that Kafka can guarantee load balancing over multiple consumer processes. In addition, the concept of the consumer group in Kafka also contributes to making it more scalable. In a consumer group, a particular partition is consumed by only one consumer in the group. This aids in the parallelism of consuming multiple messages on a topic.

53. What is the command to start ZooKeeper?

```
bin/zookeeper-server-start.sh
```

54. Explain how topics can be added and removed.

To create a topic:

```
kafka/bin/kafka-topics.sh --create \

--zookeeper localhost:2181 \

--replication-factor [replication factor] \

--partitions [number_of_partitions] \

--topic [unique-topic-name]
```

To Delete a topic:

- Go to ${kafka_home}/config/server.properties, and add the below line:

```
Delete.topic.enable = true
```

- Start the Kafka server once again with the new configuration:

```
${kafka_home}/bin/kafka-server-start.sh ~/kafka/config/server.properties
```

- Delete the topic:

```
${kafka_home}/bin/kafka-topics.sh --delete --zookeeper localhost:2181 --topic topic-name
```

55. Explain how topic configurations can be modified in Apache Kafka.

To add a config:

```
bin/kafka-configs.sh --zookeeper localhost:2181 --topics --topic_name --alter --add-config x=y
```

To remove a config:

```
bin/kafka-configs.sh --zookeeper localhost:2181 --topics --topic_name --alter --delete-config x
```

Where x is the particular configuration key that has to be changed.

56. Mention some differences between Kafka and JMS.

Kafka	JMS (Java Messaging Service)
The delivery system is based on a pull mechanism. Consumers pull the messages when they are ready to consume them.	The message delivery system is based on a push model. Messages are automatically delivered to consumers.
Messages are retained for a specific period of time, even after the consumer reads the messages.	Once the JMS queue receives an acknowledgment that the consumer has received the message, it gets permanently deleted.
Kafka guarantees that the partitions are delivered in the order that they were present in the message.	JMS is a queue that works on the FIFO system; it does not support any other form of order.
Kafka is more suitable for handling a large volume of data.	JMS is more suitable in highly complex systems with multi-node clusters.

57. What is meant by Kafka Connect?

Kafka Connect is a tool provided by Apache Kafka to allow scalable and reliable streaming data to move between Kafka and other systems. It makes it easier to define connectors that are responsible for moving large collections of data in and out of Kafka. Kafka Connect is able to process entire databases as input. It can also collect metrics from application servers into Kafka topics so that this data can be available for Kafka stream processing.

58. Explain message compression in Apache Kafka.

In Apache Kafka, producer applications write data to the brokers in JSON format. The data in the JSON format is stored in string form, which can result in several duplicated records getting stored in the Kafka topic. This leads to an increased occupation of disk space. Hence, to reduce this disk space, compression of messages or lingering the data is performed before sending the messages to Kafka. Message compression is done on the producer side, and hence there is no need to make any changes to the configuration of the and hence there is no need to make any changes to the configuration of the consumer or the broker.

59. What is the need for message compression in Apache Kafka?

Message compression in Kafka does not require any changes in the configuration of the broker or the consumer. It is beneficial for the following reasons:

- Due to reduced size, it reduces the latency in which messages are sent to Kafka.
- Reduced bandwidth allows the producers to send more net messages to the broker.
- When the data is stored in Kafka via cloud platforms, it can reduce the cost in cases where the cloud services are paid.
- Message compression leads to reduced disk load, which will lead to faster read and write requests.

60. What are some disadvantages of message compression in Apache Kafka?

- Producers end up using some CPU cycles for compression.
- Consumers use some CPU cycles for decompression.
- Compression and decompression result in greater CPU demand.

61. Explain producer batch in Apache Kafka.

Producers write messages to Kafka, one at a time. Kafka waits for the messages that are being sent to Kafka, creates a batch and puts the messages into the batch, and waits until this batch becomes full. Only then is the batch sent to Kafka. The batch here is known as the producer batch. The default size of a producer batch is 16KB, but it can be modified. The larger the batch size, the more is the compression and throughput of the producer requests.

62. Highlight some differences between Kafka Streams and Spark Streaming.

Kafka streams	Spark Streaming
Able to handle only real-time streams	Can handle real-time streams as well as batch processes.
The use of partitions and their replicas allows Kafka to be fault-tolerant.	Spark allows recovery of partitions using Cache and RDD (resilient distributed dataset)
Kafka does not provide any interactive modes. The broker simply consumes the data from the producer and waits for the client to read it.	Has interactive modes

Messages remain persistent in the Kafka log.	A dataframe or some other data structure has to be used to keep the data persistent.

63. Explain how Apache Kafka provides security.

There are three components to the security provided by Kafka:

- Encryption: All the message transfer processes between the Kafka broker and its various clients are secured through encryption. This ensures that other clients cannot intercept the data. All the messages are shared between the components in an encrypted format.
- Authentication: applications that are making use of the Kafka broker have to be authenticated before they can be connected to Kafka. Only authorized applications will be allowed to consume or publish messages. Authorized applications will have unique ids and passwords to identify themselves.
- Authorization: this is done after authentication. Once a client is authenticated, it is allowed to consume or publish messages. The authorization ensures that applications can be restricted from write access to prevent data pollution.

64. Can a consumer read more than one partition from a topic?

Yes, if the number of partitions is greater than the number of consumers in a consumer group, then a consumer will have to read more than one partition from a topic.

65. Can Apache Kafka be considered to be a distributed streaming platform? Elaborate.

Yes, Apache Kafka is considered to be a distributed streaming platform. A streaming platform can be called such if it has the following three capabilities:

- Be able to publish and subscribe to streams of data.

- Provide services similar to that of a message queue or scalable enterprise messaging system.
- Store streams of records in a durable and fault-tolerant manner.

Since Kafka meets all three of these requirements, it can be considered to be a streaming platform.

Furthermore, since a Kafka cluster consists of multiple servers that function as brokers, it is said to be distributed. Kafka topics are divided into multiple partitions to ensure load balancing. Brokers process these partitions parallelly and allow multiple producers and consumers to publish and retrieve messages in parallel.

Distributed streaming platforms handle large amounts of data in real-time by pushing them to multiple servers for real-time processing.

66. Mention some use cases where Apache Kafka is not suitable.

- Kafka is built to handle high volumes of data. If the requirement is to process only a small number of messages per day, a traditional messaging system would be more suitable.
- Kafka has a streaming API, but it is not suitable for performing data transformation operations. Kafka is to be avoided for ETL (extract, transform, load) jobs.
- In cases where a simple task queue is needed, there are better alternatives like the RabbitMQ.
- Kafka is not good if long-term storage is required. It supports saving data only for a specified retention period and not longer than that.

67. Define consumer lag in Apache Kafka.

Consumer lag refers to the lag between the Kafka producers and consumers. Consumer groups will have a lag if the data production rate far exceeds the rate at which the data is getting consumed. Consumer lag is the difference between the latest offset and the consumer offset.

68. What guarantees does Kafka provide?

- Consumers can see the messages in the same sequence in which the producers published them. The messaging order is preserved.
- The replication factor determines the number of replicas. If the replication factor is n, then there is a fault tolerance for up to n-1 servers in the Kafka cluster.
- Kafka can guarantee “at least one” delivery semantics per partition. This means that for multiple attempts at delivering a partition, Kafka guarantees that it will be delivered to a consumer at least once.

69. What do you know about log compaction in Kafka?

Log compaction is a method by which Kafka ensures that at least the last known value for each message key within the log of data is retained for a single topic partition. This makes it possible to restore the state after an application crashes or in the event of a system failure. It allows cache reloading once an application restarts during any operational maintenance. Log compaction ensures that any consumer processing the log from the start can view the final state of all records in the original order they were written.

70. What do you understand about quotas in Kafka?

As of Kafka 0.9, a Kafka cluster can enforce quotas on producers and fetch any client request. Quotas are byte-rate thresholds that are defined per client-id. A client-id is used to identify an application making a client request logically. A single client-id can refer to multiple producers and consumer instances. The quota will apply to all of them as a single entity. Quotas ensure that a single application does not monopolize the broker resources and cause network saturation by consuming very high volumes of data.

71. What is meant by cluster id in Kafka?

Kafka clusters are assigned unique and immutable identifiers. The identifier for a particular cluster is known as the cluster id. A cluster id can have a maximum number of 22 characters and has to follow the regular expression [a-zA-Z0-9_\-]+. It is generated when a broker with version 0.10.1 or later is successfully started for the first time. The broker attempts to get the cluster id from the znode during startup. If the znode does not exist, the broker generates a new cluster id and creates a znode with this cluster id.

72. Can Apache Kafka be integrated with Apache Storm? If yes, explain how.

Yes, Apache Kafka and Apache Storm naturally complement each other. Apache Storm is a distributed real-time processing system that allows the processing of very large amounts of data. Storm runs continuously consuming data from configured sources and passes it along the data pipeline to configured destinations.

In Storm, for streaming data processing, the following components work together:

- Spout: source of the stream. It is a continuous stream of log data
- Bolt: the bolt consumes input streams, processes them, and possibly emits new streams.

Here are some of the classes that can be used to integrate Apache Storm and Apache Kafka:

BrokerHosts:

BrokerHosts is an interface. ZkHosts and StaticHosts are two of their implementations. ZkHosts tracks the Kafka brokers dynamically and is used to maintain their details in the ZooKeeper. StaticHosts is used to manually set the Kafka brokers and their details.

SpoutConfig:

This class is an extension of the KafkaConfig class that supports additional ZooKeeper information. Its signature is as follows:

public SpoutConfig(BrokerHosts hosts, string topic, string zkRoot, string id)

Where:

- hosts: any implementation of BrokerHosts interface
- Topic: Name of the topic
- zkRoot: root path of the ZooKeeper
- Id: the spout stores the state of the offset that it has consumed in ZooKeeper. 'Id' here should uniquely identify the spout.

KafkaSpout API

KafkaSpout is our spout implementation, which will be integrated with Storm. It fetches the messages from the Kafka topic and emits them into the Storm ecosystem as tuples. KafkaSpout gets its configuration details from SpoutConfig.

IRichBolt

Bolt creation is done using the IRichBolt interface. Bolts take tuples as input, process the tuples, and produce new tuples as the output.

IRichBolt interface has the methods listed below:

- Prepare - this provides the bolt with an environment to execute. An executor can call this method to initialize the spout.
- Execute: this is used to process a single tuple of input.
- Cleanup - this is called when a bolt is ready to be shut down.
- declareOutputFields - this is used to specify the output schema of the tuple.

73. Why is the Kafka broker said to be "dumb"?

The Kafka broker does not keep a tab of which the consumers have read messages. It simply keeps all of the messages in its queue for a fixed time, known as the retention time, after which the messages are deleted. It is the responsibility of the consumer to pull the messages from the queue. Hence, Kafka is said to have a "smart-client, dumb-broker" architecture.

74. When does Kafka throw a BufferExhaustedException?

BufferExhaustedException is thrown when the producer cannot allocate memory to a record due to the buffer being too full. The exception is thrown if the producer is in non-blocking mode and the rate of data production exceeds the rate at which data is sent from the buffer for long enough for the allocated buffer to be exhausted.

75. What are the responsibilities of a Controller Broker in Kafka?

The main role of the Controller is to manage and coordinate the Kafka cluster, along with the Apache ZooKeeper. Any broker in the cluster can take on the role of the controller. However, once the application starts running, there can be only one controller broker in the cluster. When the broker starts, it will try to create a Controller node in ZooKeeper. The first broker that creates this controller node becomes the controller.

The controller is responsible for:

- creating and deleting topics
- Adding partitions and assigning leaders to the partitions
- Managing the brokers in a cluster - adding new brokers, active broker shutdown, and broker failures
- Leader Election
- Reallocation of partitions.

76. What causes OutOfMemoryException?

OutOfMemoryException can occur if the consumers are sending large messages or if there is a spike in the number of messages wherein the consumer is sending messages at a rate faster than the rate of downstream processing. This causes the message queue to fill up, taking up memory.

77. How can Kafka retention time be changed at runtime?

The retention time can be configured in Kafka for a topic. The default retention time for a topic is seven days. The retention time can be configured while a new topic is set up. Log.retention.hours is the property of a broker, which is used to set the retention time when a topic is created. However, when configurations have to be changed for a currently running topic, kafka-topic.sh will have to be used.

The correct command depends on the version of Kafka that is in use.

Up to 0.8.2 kafka-topics.sh --alter is the command to be used.

From 0.9.0 going forward, use kafka-configs.sh --alter

78. Explain the graceful shutdown in Kafka.

Any broker shutdown or failure will automatically be detected by the Apache cluster. In such a case, new leaders will be elected for partitions that were previously handled by that machine. This can occur due to server failure and even if it is intentionally brought down for maintenance or any configuration changes. In cases where the server is intentionally brought down, Kafka supports a graceful mechanism for stopping the server rather than just killing it.

Whenever a server is stopped:

- Kafka ensures that all of its logs are synced onto a disk to avoid needing any log recovery when it is restarted. Since log recovery takes time, this can speed up intentional restarts.
- Any partitions for which the server is the leader will be migrated to the replicas prior to shutting down. This ensures that the leadership transfer is faster, and the time during which each partition is unavailable will be reduced to a few milliseconds.

79. Can the number of partitions for a topic be reduced?

No, Kafka does not allow reducing the number of partitions for a topic. The partitions can only be increased but not decreased.

80. How can a cluster be expanded in Kafka?

In order to add a server to a Kafka cluster, it just has to be assigned a unique broker id, and Kafka has to be started on this new server. However, a new server will not automatically be assigned any of the data partitions until a new topic is created. Hence, when a new machine is added to the cluster, it becomes necessary to migrate some existing data to these machines. The partition reassignment tool can be used to move some partitions to the new broker. Kafka will add the new server as a follower of the partition that it is migrating to and allow it to completely replicate the data on that particular partition. When this data is fully replicated, the new server can join the ISR; one of the existing replicas will delete the data that it has with respect to that particular partition.

81. Explain customer serialization and deserialization in Kafka.

In Kafka, message transfer among the producer, broker, and consumers is done by making use of a standardized binary message format. The process of converting the data into a stream of bytes for the purpose of the transmission is known as serialization. Deserialization is the process of converting the bytes of arrays into the desired data format. Custom serializers are used at the producer end to let the producer know how to convert the message into byte arrays. Deserializers are used at the consumer end to convert the byte arrays back into the message.

82. What is meant by the Kafka schema registry?

For both the producers and consumers associated with a Kafka cluster, a Schema Registry is present, which stores Avro schemas. Avro schemas allow the configuration of compatibility settings between the producers and the consumers for seamless serialization and deserialization. Kafka Schema Registry is used to ensure that there is no difference in the schema that is being used by the consumer and the one that is being used by the producer. While using the Confluent schema registry in Kafka, the producers only need to send the schema ID and not the entire schema. The consumer uses the schema ID to look up the corresponding schema in the Schema Registry.

83. What can Kafka Monitoring be used to do?

- Track Resource Utilization of the system: it can be used to keep a close tab on the resources like memory, CPU, and disk utilization over time.
- Monitor threads and JVM usage: since Kafka relies on the Java garbage collector to free up memory, ensuring that the garbage collector runs frequently ensures that more activity occurs in the Kafka cluster.
- Watch statistics related to the broker, controller, and replication so that the states of partitions and replicas can be adjusted if required.
- Performance problems can be fixed quickly by finding out which applications cause excessive load and identifying performance bottlenecks.

84. How does Kafka ensure minimal data modification when data passes from the producer to the broker to the consumer?

Kafka uses a standardized binary message format that is shared by the producer, broker, and consumer to ensure that the data can pass without any modification.

85. Name the various types of Kafka producer API.

There are three types of Kafka producer API available-

- Fire and Forget
- Synchronous producer
- Asynchronous producer

86. What role does the Kafka consumer API and Kafka producer API play?

A consumer API enables an application to subscribe to one or more topics and process the stream of records provided to them.

You can see Kafka producer API play the role of a wrapper for the two producers—Sync Producer and Async Producer. The goal is to give the client access to every producer capability using a single API.

87. How can you write data from Kafka to a database?

There are two different frameworks- Connect Source and Connect Sink. You can send topics from Kafka to external databases and load the data from source databases.

88. What is the best method to determine the number of topics in a single Kafka broker?

The list command can be used to see all of the topics in a broker. Additionally, you can view subject details with the describe command.

you can view subject details with the describe command.

Apache Kafka ZooKeeper Interview Questions

Apache Zookeeper is a high-level program that supports naming and configuration management and serves as a centralized data store service. Kafka Zookeeper allows distributed systems to synchronize in a flexible and reliable manner. Zookeeper accomplishes its high availability and consistency by distributing the data over multiple nodes. If a node fails, Zookeeper is capable of performing instant failover migration; for instance, if a leader node fails, a new one is chosen in real-time using ensemble polling. If the initial node fails to respond, a client connecting to the server can search for a different node.

Let us quickly explore some of the important Kafka interview questions based on Apache Kafka Zookeeper concepts, for example, znodes, watches, quorum, barriers, etc.

89. Name the configuration file to be used to set up ZooKeeper properties in Kafka.

zookeeper.properties file.

90. What is the ZooKeeper ensemble?

ZooKeeper works as a coordination system for distributed systems and is a distributed system on its own. It follows a simple client-server model, where clients are the machines that make use of the service, and the servers are nodes that provide the service. The collection of ZooKeeper servers forms the ZooKeeper ensemble. Each ZooKeeper server is capable of handling a large number of clients.

91. What are Znodes?

Nodes in a ZooKeeper tree are referred to as znodes. Znodes maintain a structure that contains version numbers for data changes, acl changes, and also timestamps. The version number, along with the timestamp, allows ZooKeeper to validate the cache and ensure that updates are coordinated. The version number associated with Znode increases each time the znode's data changes.

92. What are the types of Znodes?

There are three types of Znodes, namely:

Persistence Znode: these are the znodes that remain alive even after the client who created that particular znode is disconnected. All znodes are persistent by default unless otherwise specified.

Ephemeral Znode: Ephemeral znodes remain active only until the client is alive. Ephemeral Znodes get deleted whenever the client that created them gets disconnected from the ZooKeeper ensemble. They play an important role in the leader election.

Sequential Znode: when znodes are created, it is possible to request the ZooKeeper to add an increasing counter to the end of the path. This counter is unique to the parent znode. Sequential nodes may be persistent or ephemeral.

93. How can we create Znodes?

Znodes are created within the given path.

Syntax:

```
create /path/data
```

Flags can be used to specify whether the znode created will be persistent, ephemeral, or sequential.

```
create -e /path/data
```

creates an ephemeral znode.

```
create -s /path/data
```

creates a sequential znode.

All znodes are persistent by default.

94. How can we remove Znodes?

```
rmr /path
```

It can be used to remove the znode specified and all its children.

95. What are ZooKeeper watches?

Clients can set watches on znodes. Any changes to that particular znode trigger the watch. When a watch triggers, ZooKeeper sends the client a notification. A watch event is a one-time trigger; another watch has to be set to get further notifications. Watches are maintained locally at the ZooKeeper server to which the client is connected.

96. What is ZooKeeper quorum?

The ZooKeeper quorum is the minimum number of server nodes that must be available for client requests. Updates done to the ZooKeeper tree by the clients must be stored in the number of servers that form the quorum for a transaction to be completed successfully.

Q = 2N + 1 is the recommended number of nodes required to form an Ensemble as defined by the quorum where:

Q - number of nodes required to form a healthy ensemble.

N - number of failure nodes that can be allowed

N - number of failure nodes that can be allowed.

97. What are the benefits of a distributed application?

Reliability: the failure of a single or a few systems does not cause the whole system to fail.

Scalability: The performance of the application can be increased as per requirements by adding more machines with minor changes in the configuration of the application with no downtime.

Transparency: the complexity of the system is masked from the users, as the application shows itself as a single entity.

98. What are some disadvantages of a distributed application?

Race conditions may arise: two or more machines trying to access the same resource can cause a race condition, as the resource can only be given to a single machine at a time.

Deadlock: in the process of trying to solve race conditions, deadlocks may arise where two or more operations may end up waiting for each other to complete indefinitely.

Inconsistency may arise due to partial failure in some of the systems.

99. What is meant by the ZooKeeper Atomic Broadcast (ZAB) Protocol?

The ZAB Protocol is the core of the ZooKeeper system, which ensures that all of the servers remain in sync. The ZAB protocol guarantees reliable delivery and ensures that messages are delivered in the same order to all the machines.

100. Where else is the ZooKeeper used?

Apart from Apache Kafka, where ZooKeeper is used for maintaining the leader and followers among the nodes in a cluster for topic partitions, ZooKeeper is also used in the following places:

- Apache Storm: manages its state of being a real-time processing framework using the ZooKeeper service.
- Apache YARN uses ZooKeeper for automatic failover of the master node.
- Yahoo! Performs the coordination and failure recovery service for Yahoo! Message Broker, which manages large amounts of data for replication and delivery. It is also used by the Fetching Service for Yahoo! Crawler for the purpose of failure recovery.

101. What are ZooKeeper barriers?

In ZooKeeper, barriers are primitives that allow a group of processes to remain in synchronization at the start and the end of a computation. There is a barrier node that serves as a parent for individual process nodes.

102. Explain Cages in ZooKeeper?

102. Explain Cages in ZooKeeper?

Cages refer to a distributed synchronization library for ZooKeeper. If ZooKeeper is being run on a machine or on a cluster, Cages can be used to synchronize and coordinate data access, manipulation, and processing of data, configuration management, and also membership of machines within the cluster.

103. What is the daemon name for ZooKeeper?

Quorumpeermain

104. Explain the CLI in ZooKeeper.

The ZooKeeper command-line interface or the ZooKeeper CLI is used to allow interaction with the ZooKeeper ensemble. It can be used for debugging and working with different options.

To perform CLI operations, first, turn on the ZooKeeper server and then the ZooKeeper client.

The ZooKeeper CLI can be used to perform the following commands:

- Create znodes.
- Watch znodes for changes
- Create children of a znode.
- List children of a znode.
- Remove/ delete a znode.
- Fetch data and the metadata associated with a znode.
- Set data for a particular znode.
- Check the status of a znode, such as a timestamp, version number, and data length.

RabbitMQ vs. Kafka Interview Questions

Messaging systems like Apache Kafka and RabbitMQ allow you to manage big data streams in distributed computing, including tasks such as data consumption, reading, writing, processing, etc. Although both systems are

equally good as they offer numerous features, they are suitable for different big data use cases. Take a look at some of the Apache Kafka interview questions based on the differences between Apache Kafka and RabbitMQ.

105. Differentiate between RabbitMQ and Apache Kafka.

RabbitMQ	Kafka
General-purpose message broker that uses variations of request/reply, point-to-point, and publish-subscribe messaging system.	High-volume streaming and publish-subscribe messaging system.
Does not support message ordering, but since RabbitMQ is a queue, messages are stored by default in a first-in-first-out format (FIFO).	Provides message ordering using partition keys.
Messages are removed from the RabbitMQ queue once consumed, and acknowledgment is provided.	Messages remain in the Kafka log for the amount of time specified by the retention period.
RabbitMQ allows specifying message priority.	No such feature is provided.

106. How does Kafka perform better than RabbitMQ?

Kafka has much higher performance than message brokers like RabbitMQ mainly due to its use of sequential disc I/O. It can generate high throughput (millions of messages per second) with limited resources, which is crucial for big data use cases. RabbitMQ can also deal with a million messages per second, though somewhat at the cost of greater resources (around 30 nodes).

107. Does Kafka follow the same approach as RabbitMQ for message handling?

No, Kafka does not follow the same approach as RabbitMQ for message handling.

Kafka adopts the pull approach. Consumers request a specific offset for batches of messages. Kafka enables long-pooling when there are no messages past the offset, which minimizes tight loops. A pull approach is appropriate due to Kafka's partitions. Kafka uses offsets for ordering messages in a partition with no conflicting users. Users can now leverage message batching for faster message delivery and higher efficiency.

RabbitMQ implements the push approach, which imposes a prefetch restriction on users to avoid them from becoming overburdened. This can be useful for low-latency messaging. The goal of the push model is to distribute messages separately and quickly, ensuring that work is fairly distributed and messages are processed roughly in the order they arrive in the queue.

Kafka Scenario-based Interview Questions

Every interview is incomplete without scenario-based questions. Such questions help the interviewer judge your critical thinking and problem-solving skills. Check out these Apache Kafka scenario-based interview questions to leave a solid impression in your next big data job interview.

108. Suppose you are sending messages to a Kafka topic using kafkaTemplate. You come across a requirement that states that if a failure occurs while delivering messages to a Kafka topic, you must retry sending the messages on the same partition with the same offset. How can you achieve this using kafkatemplate?

If you give the key while delivering the message, it will be stored in the same partition regardless of how many times you send it. The hashed key is used by Kafka to decide which partition needs to be updated.

The only way to ensure that a failed message has the same offset when retried is to ensure that nothing is put into the topic before retrying it.

109. Assume your brokers are hosted on AWS EC2. If you're a producer or consumer outside of the Kafka cluster network, you will only be capable of reaching the brokers over their public DNS, not their private DNS. Now, assume your client (producer or consumer) is outside your Kafka cluster's network, and you can only reach the brokers via their public DNS. The private DNS of the brokers hosting the leader partitions, not the public DNS, will

be returned by the broker. Unfortunately, since your client is not present on your Kafka cluster's network, they will be unable to resolve the private DNS, resulting in the LEADER NOT AVAILABLE error. How will you resolve this network error?

When you first start using Kafka brokers, you might have many listeners. Listeners are just a combination of hostname or IP, port, and protocol.

Each Kafka broker's server.properties file contains the properties listed below. The important property that will enable you to resolve this network error is *advertised.listeners.*

- listeners – a list of comma-separated hostnames and ports that Kafka brokers listen to.
- advertised.listeners – a list of comma-separated hostnames and ports that will be returned to clients. Only include hostnames that will be resolved at the client (producer or consumer) level, such as public DNS.
- inter.broker.listener.name – listeners used for internal traffic across brokers. These hostnames do not need to be resolved on the client side, but all of the cluster's brokers must resolve them.
- listener.security.protocol.map – lists the supported protocols for each listener.

110. Let's suppose a producer writes records to a Kafka topic at a rate of 10000 messages per second, but the consumer can only read 2500 messages per second. What are the various strategies for expanding your consumer group?

The solution to this question has two parts: topic partitions and consumer groups.

Partitions are used to split a Kafka topic. The producer's message is divided among the topic's partitions based on the message key. You can suppose that the key is chosen in such a way that messages are spread evenly between the partitions.

Consumer groups are a method of grouping consumers together to maximize a consumer application's throughput. Each consumer in a consumer group holds on to a topic partition. If the Kafka topic has four partitions and the consumer group has four consumers, each consumer will read from a single partition. If there are six partitions and four consumers, the data will be read in parallel from only four partitions. As a result, maintaining a 1-to-1 mapping of partition to the consumer in the consumer group is preferable.

Now, you can do two things to increase processing on the consumer side:

- You can increase the topic's partition count (say from existing 1 to 4).
- You can build a Kafka consumer group with four consumer instances tied

- You can build a Kafka consumer group with four consumer instances tied to it.

This would enable the consumers to read data from the topic in parallel, allowing it to expand from 2500 to 10000 messages per second.

Kafka Interview Questions for Java Developers | Kafka Developer Interview Questions

Kafka skills are highly in-demand these days, whether you are applying for a Big Data Engineer role, a Java Developer role, or a Kafka Developer role. This section will walk you through some Apache Kafka interview questions that are crucial for all the Java Developers and Kafka Developers out there.

111. What is Kafka's producer acknowledgment? What are the various types of acknowledgment settings that Kafka provides?

A broker sends an ack or acknowledgment to the producer to verify the reception of the message. Ack level is a configuration parameter in the Producer that specifies how many acknowledgments the producer must receive from the leader before a request is considered successful. The following types of acknowledgment are available:

- acks=0

In this setting, the producer does not wait for the broker's acknowledgment. There is no way to know if the broker has received the record.

- acks=1

In this situation, the leader logs the record to its local log file and answers without waiting for all of its followers to acknowledge it. The message can only be lost in this instance if the leader fails shortly after accepting the record but before the followers have copied it; otherwise, the record would be lost.

- acks=all

A set leader in this situation waits for all in-sync replica sets to acknowledge the record. As long as one replica is alive, the record will not be lost, and the best possible guarantee will be provided. However, because a leader must wait for all followers to acknowledge before replying, the throughput is significantly lower.

112. How do you get Kafka to perform in a FIFO manner?

Kafka organizes messages into topics, which are then divided into partitions. The partition is an immutable list of ordered messages that is updated regularly. A message in the partition is uniquely recognized by a sequential number called offset. FIFO behavior is possible only within the partitions. Following the methods below will help you achieve FIFO behavior:

- To begin, we first set the enable the auto-commit property to be false:

Set enable.auto.commit=false

- We should not call the consumer.commitSync(); method after the messages have been processed.
- Then we may "subscribe" to the topic and ensure that the consumer system's register is updated.
- You should use Listener consumerRebalance, and call a consumer inside a listener.

seek(topicPartition, offset).

- The offset related to the message should be kept together with the processed message once it has been processed.

113. How is it possible for a Kafka producer to retain exactly one semantics?

Kafka transactions assist Kafka brokers and clients in achieving precisely one semantics. To accomplish this, you must specify the properties enable.idempotence=true and transactional.id=some unique id> at the producer end. In order to prepare the producer for transactions, you must also call initTransaction. If the producer (identified by producer id>) delivers the same message to Kafka more than once with these properties set, the Kafka broker identifies and de-duplicates it.

Tricky Kafka Interview Questions

Here are a few tricky Apache Kafka interview questions for the times when your interviewer decides to test your knowledge and skills a little more but in a challenging manner-

114. Can a Kafka Consumer group have more than one consumer?

Yes, a Kafka consumer group consists of one or more consumers. In general, each customer may associate with one partition. The additional Kafka consumers in the group become inactive if the number of consumers in the specific consumer group exceeds the number of partitions. So, you must always ensure that a Kafka consumer group consists of the optimum number of Kafka consumers.

115. What does it indicate if a replica stays out of ISR for a long time?

A replica staying out of ISR for a long time indicates that the follower cannot fetch data at the same rate as data accumulated by the leader.

116. Describe how you can acquire precisely one messaging from Kafka during the data production process.

To acquire exactly one messaging from Kafka during data production, you must prevent duplication both during data production and data consumption. Below are the two methods for obtaining a single semantic during data production:

- Provide one writer per partition, and whenever a network error occurs, review the most recent message in that partition to determine if your latest write was successful.
- Include a primary key (a UUID or similar identifier) in the message and de-duplicate it for the recipient.

Kafka Admin Interview Questions | Kafka Technical Interview Questions

An individual working as a Kafka Admin is responsible for building and maintaining the entire Kafka messaging infrastructure and has expertise in concepts like Kafka brokers, Kafka server, etc.

Check out these important Apache Kafka interview questions that will help you crack any Kafka Admin job interview-

117. How can you create a Kafka topic?

[root@localhost kafka_2.9.2-0.8.1.1] #bin/kafka-topics.sh –create —zookeeper localhost:2181 –replication-factor 1 –partitions 1 –topic kafkatopic

The command above generates Kafkatopic as the topic name, and also, replication:1, partition:1.

118. What is the method to create Kafka producer API?

[root@localhost kafka_2.9.2-0.8.1.1] # bin/kafka-console-producer.sh —broker-list localhost:9092 –topic kafkatopic

This will create a producer client that receives messages from the command line and publishes messages to a single Kafka broker.

119. How do you start a single Kafka Broker?

[root@localhost kafka_2.9.2-0.8.1.1] # bin/kafka-server-start.sh config/server.properties

Why is Kafka technology significant in the Big Data industry?

Apache Kafka is an open-source stream processing platform developed by the Apache Software Foundation. It is written in Scala and Java. Kafka was originally created at LinkedIn and then open-sourced in 2011. Kafka aims to provide a platform for real-time handling data feeds and can handle trillions of events on a daily basis. Here are some features of Apache Kafka that make the Kafka technology significant in the Big Data industry:

- Scalability: Apache Kafka is able to handle messages of high volume and high velocity, making it very scalable without any downtime.
- High throughput: Apache Kafka can handle thousands of messages per second. Messages coming in at high volume and high velocity do not affect the performance of Kafka.
- Low latency: Apache Kafka offers low latency, which is low as ten

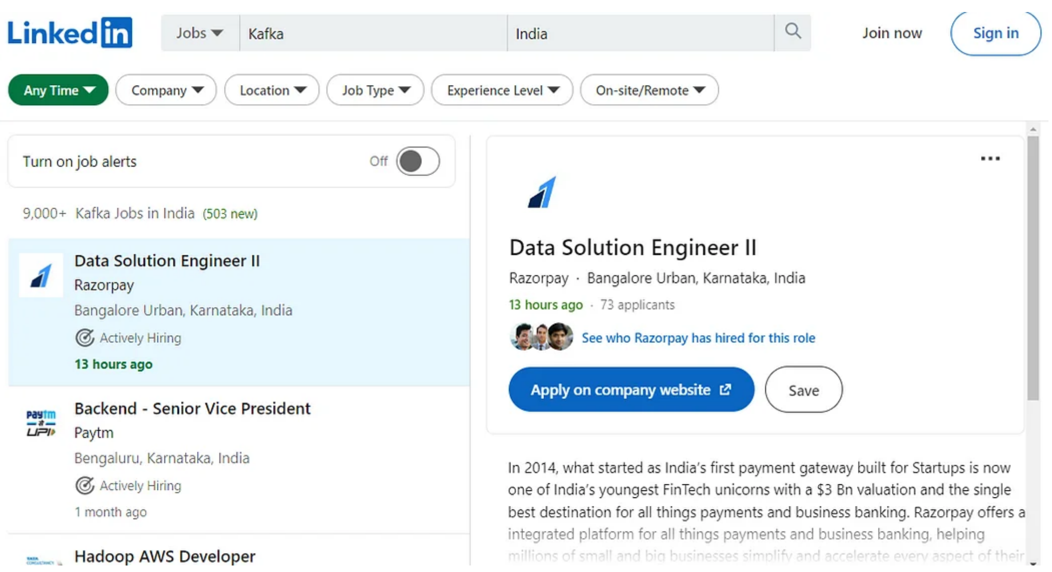
milliseconds.

- Fault tolerance: Kafka is able to handle failures at nodes in a cluster, ensuring that the data is kept secure and a process running is not disturbed.
- Reliability: Kafka is a distributed platform with high fault tolerance making it very reliable.
- Durability: Data on Kafka is allowed to remain more persistent on the cluster rather than on the disk, making it quite durable.
- Real-time data handling: Kafka can handle real-time data pipelines.

Apache Kafka Jobs Growth Trends- 2022

According to a report on 10 Hottest Technical Skills for 2020, over 30% of software engineering jobs and 11% of data engineer jobs listed Apache Kafka as a must-have skill. According to Dice, Apache Kafka is considered to be one of the top tech skills in 2020, and salaries associated with it were the highest of any skill set in 2019 - $134,557 per annum in the United States.

Close observation of the below two searches on LinkedIn for Kafka technology skills shows that there are over 38000 job openings just for Apache Kafka specialists in the United States and over 9000 job openings for Kafka specialists in India.



9,000+ Kafka Jobs in India (503 new)

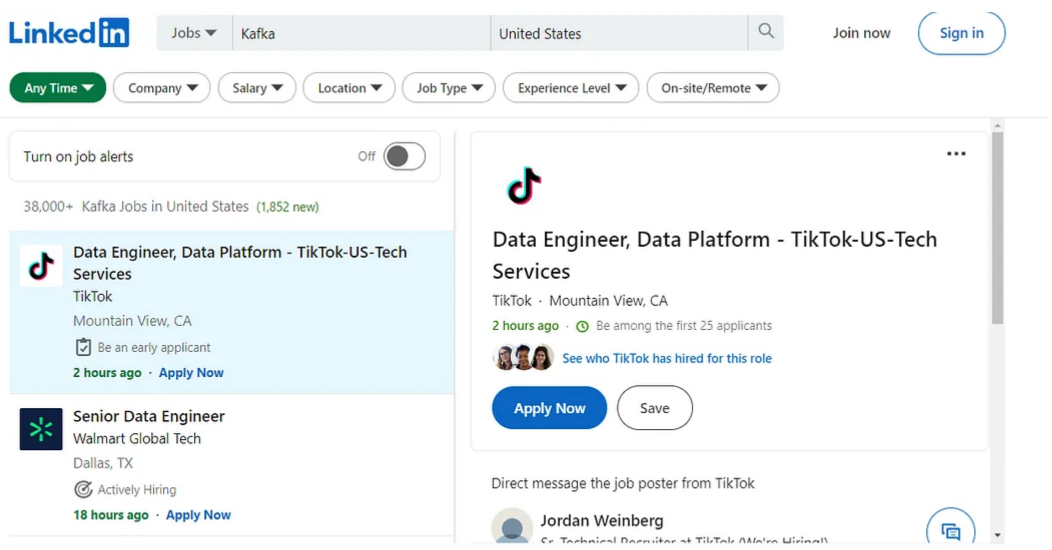
Data Solution Engineer II
Razorpay
Bangalore Urban, Karnataka, India
Actively Hiring
13 hours ago

Backend - Senior Vice President
Paytm
Bengaluru, Karnataka, India
Actively Hiring
1 month ago

Hadoop AWS Developer

Data Solution Engineer II
Razorpay · Bangalore Urban, Karnataka, India
13 hours ago · 73 applicants
See who Razorpay has hired for this role

In 2014, what started as India's first payment gateway built for Startups is now one of India's youngest FinTech unicorns with a $3 Bn valuation and the single best destination for all things payments and business banking. Razorpay offers a integrated platform for all things payments and business banking, helping millions of small and big businesses simplify and accelerate every aspect of their



38,000+ Kafka Jobs in United States (1,852 new)

Data Engineer, Data Platform - TikTok-US-Tech Services
TikTok
Mountain View, CA
Be an early applicant
2 hours ago · Apply Now

Senior Data Engineer
Walmart Global Tech
Dallas, TX
Actively Hiring
18 hours ago · Apply Now

Data Engineer, Data Platform - TikTok-US-Tech Services
TikTok · Mountain View, CA
2 hours ago · Be among the first 25 applicants
See who TikTok has hired for this role

Direct message the job poster from TikTok
Jordan Weinberg
Sr. Technical Recruiter at TikTok (We're Hiring!)

But let's put first things first and understand the demand for Kafka's skills in the industry.

The Market Demand for Kafka Skills

According to HG insights, there are over 18,000 companies that use Apache Kafka, and 3729 developers on StackShare have mentioned that they use

Kafka, and 3729 developers on StackShare have mentioned that they use Apache Kafka.

- Twitter uses Apache Storm and Apache Kafka as an integral part of it's stream processing infrastructure.
- Kafka was originally created by LinkedIn for the purpose of monitoring and tracking user activity stream data and calculating operational metrics. Kafka messaging system is an integral part of LinkedIn's popular products like LinkedIn today and LinkedIn Newsfeed.LinkedIn maintains over 100 Kafka clusters with over 4K brokers that serve 100,000+ topics and 7M partitions. Kafka deployments at LinkedIn handle over 7 trillion messages every day.
- Netflix uses Kafka for real-time monitoring and event processing.
- Oracle provides native connectivity to Kafka from its Enterprise Service Bus product, which it calls OSB or Oracle Service Bus. This allows developers to use built-in features of OSB to implement staged data pipelines.

Companies like Uber, PayPal, Spotify, Goldman Sachs, Tinder, Pinterest, and Tumbler also use Kafka stream processing and message passing features and claim Kafka technology to be one of the most popular big data tools in the world.

Don’t Just Stop at These Kafka Interview Questions

We hope these 100 Apache Kafka Interview Questions and Answers have helped you gain an in-depth understanding of this leading distributed streaming and queuing technology and also helped you prepare better for your next big data job interview. Apart from honing the required Kafka skills, a good way to further build up your portfolio to land your dream job is to work and practice some big data projects that use Kafka.

Preparing for a big data interview isn’t just about practicing a few interview questions and answers on relevant big data tools and technologies or acquiring the necessary skills. It also involves gaining expert-level practical knowledge of these tools. Explore some interesting Apache Kafka Projects with source code to get hands-on experience working with Kafka as a big data tool and build a project portfolio to showcase your employers.

FAQs on Kafka Interview Questions

1. What exactly does Kafka do?

Kafka generally creates real-time streaming data pipelines and data-adaptive systems. Data replication, node re-sync, and state restoration are all possible with Kafka. In addition to real-time data analytics and stream processing, Kafka helps in various tasks such as log aggregation, messaging, load balancing, click-stream tracking, audit trails, etc.

2. How to study for Kafka interview?

You can study for Kafka interview with the help of resources such as blogs and videos on Kafka interview preparation. These resources will guide you on the types of questions asked in Kafka interviews and their suitable answers.

Furthermore, you must try gaining hands-on experience on some real-world Kafka projects on Github, ProjectPro, etc.

3. What is Kafka used for?

Kafka is a distributed system used for creating real-time streaming data pipelines and applications.

4. What are main APIs of Kafka?

The five main APIs of Kafka are Kafka Producer API, Kafka Consumer API, Kafka Streams API, Kafka Connect API, and Kafka AdminClient API.

5. What are the major components of Kafka?

The major components of Kafka include topics, producers, consumers, consumer groups, clusters, brokers, partitions, message replicas, leaders, and followers.

6. What is Kafka partition?

Kafka's topics are separated into various partitions. The smallest storage unit that contains a subset of topic-owned records is called a partition.

7. What are the best Apache Kafka interview questions and answers for experienced?

Here are some of the best Apache Kafka interview questions and answers for experienced-

- What is Apache Kafka?
- What are topics in Kafka?
- What are the major features of Apache Kafka?
- What are the main APIs of Kafka?
- What is the role of the Zookeeper in Kafka?
- Name the main components of Kafka.
- Define Partitions in Kafka.
- Explain consumer group in Kafka.
- What is the method of starting a Kafka server?
- What are znodes in Kafka Zookeeper?

PREVIOUS NEXT

Trending Project Categories

Machine Learning Projects

Data Science Projects

Deep Learning Projects

Big Data Projects

Apache Hadoop Projects

Apache Spark Projects

Show more

Trending Projects

Walmart Sales Forecasting Data Science Project

BigMart Sales Prediction ML Project

Music Recommender System Project

Credit Card Fraud Detection Using Machine Learning

Resume Parser Python Project for Data Science

Time Series Forecasting Projects

Show more

Trending Blogs

Machine Learning Projects for Beginners with Source Code

Data Science Projects for Beginners with Source Code

Big Data Projects for Beginners with Source Code

IoT Projects for Beginners with Source Code

Data Analyst vs Data Scientist

Data Science Interview Questions and Answers

Show more

Trending Recipes

Search for a Value in Pandas DataFrame

Pandas Create New Column based on Multiple Condition

LSTM vs GRU

Plot ROC Curve in Python

Python Upload File to Google Drive

Optimize Logistic Regression Hyper Parameters

Show more

Trending Tutorials

PCA in Machine Learning Tutorial

PySpark Tutorial

Hive Commands Tutorial

MapReduce in Hadoop Tutorial

Apache Hive Tutorial -Tables

Linear Regression Tutorial

Show more

ProjectPro

About us

Contact us

Privacy policy

User policy

Write for ProjectPro