1.What are various components in Kafka?

The four major components of Kafka are:

- **Topic – a stream of messages belonging to the same type**
- **Producer – that can publish messages to a topic**
- **Brokers – a set of servers where the publishes messages are stored**
- **Consumer – that subscribes to various topics and pulls data from the brokers.**

2.What is the role of Zookeeper in Kafka?

- The basic responsibility of Zookeeper is to build coordination between different nodes in a cluster.
- Since Zookeeper works as periodically commit offset so that if any node fails, it will be used to recover from previously committed to offset.
- The ZooKeeper is also responsible for configuration management, leader detection, detecting if any node leaves or joins the cluster, synchronization, etc.
- Kafka uses Zookeeper to store offsets of messages consumed for a specific topic and partition by a specific Consumer Group.

3.What is the role of the Kafka Producer API?

The role of Kafka's Producer API is to wrap the two producers – kafka.producer.SyncProducer and the kafka.producer.async.AsyncProducer.

The goal is to expose all the producer functionality through a single API to the client.

4.In the Producer, when does QueueFullException occur?

- QueueFullException typically occurs when the Producer attempts to send messages at a pace that the Broker cannot handle.
- Since the Producer doesn't block, users will need to add enough brokers to collaboratively handle the increased load.

5.What is an Offset?

- Messages contained in the partitions are assigned a unique ID number that is called the offset.
- The role of the offset is to uniquely identify every message within the partition.
- With the aid of Zookeeper Kafka stores the offsets of messages consumed for a specific topic and partition by this consumer group.

6.What are consumers or users in Kafka?

Kafka provides single consumer abstractions that discover both queuing and publish-subscribe Consumer Group. They tag themselves with a user group and every communication available on a topic is distributed to one user case within every promising user group. User instances are in disconnected process. We can determine the messaging model of the consumer based on the consumer groups.

- If all consumer instances have the same consumer set, then this works like a conventional queue adjusting load over the consumers.
- If all customer instances have dissimilar consumer groups, then this works like a publish-subscribe and all messages are transmitted to all the consumers.

7.What is the concept of leader and follower in Kafka?

Every partition in Kafka has one server which plays the role of a Leader, and none or more servers that act as Followers.

The Leader performs the task of all read and write requests for the partition, while the role of the Followers is to passively replicate the leader.

In the event of the Leader failing, one of the Followers will take on the role of the Leader. This ensures load balancing of the server.

8.What is the difference between partition and replica of a topic in Kafka cluster?

Partitions: A single piece of a Kafka topic. The number of partitions is configurable on a per topic basis. More partitions allow for great parallelism when reading from the topics. The number of partitions determines how many consumers you have in a consumer group. This partition number is somewhat hard to determine until you know how fast you are producing data and how fast you are consuming the data. If you have a topic that you know will be high volume, you will need to have more partitions.

Replicas: These are copies of the partitions. They are never written to or read. Their only purpose is for data redundancy. If your topic has n replicas, n-1 brokers can fail before there is any data loss. Additionally, you cannot have a topic a replication factor greater than the number of brokers that you have.

9.Distinguish between the Kafka and Flume?

- Flume's major use-case is to gulp down the data into Hadoop. The Flume is incorporated with the Hadoop's monitoring system, file formats, file system and utilities such as Morphlines. Flume's design of sinks, sources and channels mean that with the aid of Flume one can shift data among other systems lithely, but the main feature is its Hadoop integration.

The Flume is the best option used when you have non-relational data sources if you have a long file to stream into the Hadoop.

- Kafka's major use-case is a distributed publish-subscribe messaging system. Kafka is not developed specifically for Hadoop and using Kafka to read and write data to Hadoop is considerably trickier than it is in Flume.

Kafka can be used when you particularly need a highly reliable and scalable enterprise messaging system to connect many multiple systems like Hadoop.

10.What are the main APIs of Kafka?

Apache Kafka has 4 main APIs:

1. Producer API
2. Consumer API
3. Streams API
4. Connector API

11.What is a topic in Kafka?

A topic is a category or feed name to which records are published. Topics in Kafka are always multi-subscriber; that is, a topic can have zero, one, or many consumers that subscribe to the data written to it. For each topic, the Kafka cluster maintains a partitioned log.

12.What is Geo-Replication in Kafka?

Kafka MirrorMaker provides geo-replication support for your clusters. With MirrorMaker, messages are replicated across multiple datacenters or cloud regions. You can use this in active/passive scenarios for backup and recovery, or inactive/active scenarios to place data closer to your users, or support data locality requirements.

13.Mention What Is The Maximum Size Of The Message Does Kafka Server Can Receive?

The maximum size of the message that Kafka server can receive is 1000000 bytes.

14.What is the traditional method of message transfer?

The traditional method of message transfer includes two methods

- **Queuing**: In a queuing, a pool of consumers may read a message from the server and each message goes to one of them
- **Publish-Subscribe:** In this model, messages are broadcasted to all consumers Kafka caters single consumer abstraction that generalized both of the above- the consumer group

15.What Is The Benefits Of Apache Kafka Over The Traditional Technique?

Apache Kafka has following benefits above traditional messaging technique:

- **Fast:** A single Kafka broker can serve thousands of clients by handling megabytes of reads and writes per second
- **Scalable:** Data are partitioned and streamlined over a cluster of machines to enable larger data
- **Durable**: Messages are persistent and is replicated within the cluster to prevent data loss
- **Distributed by Design:** It provides fault tolerance guarantees and durability

16.What does ISR stand in Kafka environment?

ISR stands for In sync replicas.

They are classified as a set of message replicas which are synched to be leaders.

17.How does The process of Assigning partitions to broker Work?

When a consumer wants to join a group, it sends a JoinGroup request to the group coordinator. The first consumer to join the group becomes the group leader. The leader receives a list of all consumers in the group from the group coordinator and is responsible for assigning a subset of partitions to each consumer. It uses an implementation of PartitionAssignor to decide which partitions should be handled by which consumer.

After deciding on the partition assignment, the consumer group leader sends the list of assignments to the Group Coordinator, which sends this information to all the consumers. Each consumer only sees his own assignment—the leader is the only client process that has the full list of consumers in the group and their assignments. This process repeats every time a rebalance happens.

18.Why replication is required in Kafka?

Replication of message in Kafka ensures that any published message does not lose and can be consumed in case of machine error, program error or more common software upgrades.

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

If a replica remains out of ISR for an extended time, it indicates that the follower is unable to fetch data as fast as data accumulated at the leader.

20. Explain how you can get exactly once messaging from Kafka during data production?

During data, production to get exactly once messaging from Kafka you have to follow two things avoiding duplicates during data consumption and avoiding duplication during data production. Here are the two ways to get exactly one semantics while data production:

- Avail a single writer per partition, every time you get a network error checks the last message in that partition to see if your last write succeeded
- In the message include a primary key (UUID or something) and de-duplicate on the consumer.

1.Is it possible to get the message offset after producing?

You cannot do that from a class that behaves as a producer like in most queue systems, its role is to fire and forget the messages. The broker will do the rest of the work like appropriate metadata handling with id's, offsets, etc. As a consumer of the message, you can get the offset from a Kafka broker. If you gaze in the SimpleConsumer class, you will notice it fetches MultiFetchResponse objects that include offsets as a list. In addition to that, when you iterate the Kafka Message, you will have MessageAndOffset objects that include both, the offset and the message sent.

2.How to configure Kafka to ensure that events are stored reliably?

The following recommendations for Kafka configuration settings make it extremely difficult for data loss to occur.

1. **Producer**
   A. block.on.buffer.full=true
   B. retries=Long.MAX_VALUE
   C. acks=all
   D. max.in.flight.requests.per.connections=1
   E. Remember to close the producer when it is finished or when there is a long pause.
2. **Broker**
   A. Topic replication.factor >= 3
   B. Min.insync.replicas = 2
   C. Disable unclean leader election
3. **Consumer**
   A. Disable auto.offset.commit
   B. Commit offsets after messages are processed by your consumer client(s).

If you have more than 3 hosts, you can increase the broker settings appropriately on topics that need more protection against data loss.

3.How to rebalance the Kafka cluster?

This one comes up when a customer adds new nodes or disks to existing nodes. Partitions are not automatically balanced. If a topic already has a number of nodes equal to the replication factor (typically 3), then adding disks will not help with rebalancing.

Using the Kafka-reassign-partitions command after adding new hosts is the recommended method.

There are several caveats to using this command:

- It is highly recommended that you minimize the volume of replica changes. Say, instead of moving ten replicas with a single command, move two at a time to keep the cluster healthy.
- It is not possible to use this command to make an out-of-sync replica into the leader partition.
- If too many replicas are moved, then there could be a serious performance impact on the cluster. It is recommended that when using the Kafka-reassign-partitions command that you look at the partition counts and sizes. From there, you can test various partition sizes along with the --throttle flag to determine what volume of data can be copied without affecting broker performance significantly.
- Given the earlier restrictions, it is best to use this command only when all brokers and topics are healthy.

6.How Kafka communicate with clients and servers?

In Kafka the communication between the clients and the servers is done with a simple, high-performance, language agnostic TCP protocol. This protocol is versioned and maintains backwards compatibility with the older version.