Apache KAFKA :-- Apache KAFKA is communication system / messaging system between sender and receiver.

- kafka works on publishing and subscribing model.
- Sender will publish the data to kafka and receiver will subscribe the data from kafka.
- There can be multiple receiver to subscribe the data from kafka.

Advantage of KAFKA :-

1. High Throughput :-- It is able to handle lots of transactions, operations in distributed across the clusters.

2. Fault Tollerance :-- Fault tolerance in Kafka is done by copying the partition data to other brokers which are known as replicas.

Kafka is a distributed system. The topic is divided into partitions and kept in different brokers. If any broker fails, data should not be lost. For fault-tolerance purposes, the partition is replicated and stored in different brokers. If leader brokers fail, then the controller will elects one of the replicas as the leader. Even controller brokers can fail, in this case, Zookeeper will help in electing the broker as the controller.

In a Kafka cluster, each server is called a broker. Brokers store data and serve client requests. Kafka's fault tolerance is primarily achieved through replication. Each topic in Kafka can be configured with a replication factor, which defines the number of copies of the data that will be kept across different brokers

3. scalable :-- Kafka is designed to be scalable. As our needs change we can scale out by adding brokers to a cluster or scale in by removing brokers.
We Scale data consumption horizontally and The main way we scale data consumption from a Kafka topic is by adding more consumers to the consumer group. It is a common operation for Kafka consumers to do high-latency operations such as writing to databases or a time-consuming computation.

Key Concepts in Kafka :-- Apache Kafka is a publish-subscribe messaging system which lets you send messages between processes, applications, and servers. Applications which need to read data from Kafka use a KafkaConsumer to subscribe to Kafka topics and receive messages from these topics.

key concepts in Kafka are :--

1.Topics :- Similar to a table in database, topics is a particular stream of data. Topics consist of one or more partitions, ordered, immutable sequences of messages to which Kafka appends new messages.

2. Partitions and Offset :-- Topics are split in partitions. Each message within a partition gets an incremental id, called offset.

3. Brokers : -- A Kafka cluster consists of one or more brokers. Partitions are spread across these brokers. After connecting to any broker, you will be connected to the entire cluster.

4.Replicas :-- Topics have a replication factor to make sure if one broker is down, another broker can serve the data.

5. Zookeeper :-- Zookeeper manages brokers, helps in performing leader election for the partition.

6. Producer :-- Producers write data to topics.

7. Consumer :-- Consumer read data from a topic.

8. Lag :-- A consumer is lagging when it’s unable to read from a partition as fast as messages are produced to it. Lag is expressed as the number of offsets that are behind the head of the partition.

What is Consumer Group?

Consumer can be grouped together for a given topic for maximizing read throughput. Each consumer in a group read from mutually exclusive partitions.
If all the consumer instances have the same consumer group id, then the records will effectively be load-balanced over the consumer instances and each consumer in the group will receive messages from a different subset of the partitions in the topic.

Kafka Components :--

1 Topics :-- A stream of messages belonging to a particular category is called a topic. Data is stored in topics.

Topics are split into partitions. For each topic, Kafka keeps a mini-mum of one partition. Each such partition contains messages in an immutable ordered sequence. A partition is implemented as a set of segment files of equal sizes.

2 Partition :-- Topics may have many partitions, so it can handle an arbitrary amount of data.

3 Partition offset :-- Each partitioned message has a unique sequence id called as "offset".

4 Replicas of partition :-- Replicas are nothing but "backups" of a partition. Replicas are never read or write data. They are used to prevent data loss.

5 Brokers :--

- Brokers are simple system responsible for maintaining the pub-lished data. Each broker may have zero or more partitions per topic. Assume, if there are N partitions in a topic and N number of brokers, each broker will have one partition.
- Assume if there are N partitions in a topic and more than N brokers (n + m), the first N broker will have one partition and the next M broker will not have any partition for that particular topic.
- Assume if there are N partitions in a topic and less than N brokers (n-m), each broker will have one or more partition sharing among them. This scenario is not recommended due to unequal load distri-bution among the broker.

6 Kafka Cluster :-- Kafka's having more than one broker are called as Kafka cluster. A Kafka cluster can be expanded without downtime. These clusters are used to manage the persistence and replication of message data.

7 Producers :-- Producers are the publisher of messages to one or more Kafka topics. Producers send data to Kafka brokers. Every time a producer pub-lishes a message to a broker, the broker simply appends the message to the last segment file. Actually, the message will be appended to a partition. Producer can also send messages to a partition of their choice.

8 Consumers :-- Consumers read data from brokers. Consumers subscribes to one or more topics and consume published messages by pulling data from the brokers.

9 Leader :-- **"Leader"** is the node responsible for all reads and writes for the given partition. Every partition has one server acting as a leader.

10 Follower :-- Node which follows leader instructions are called as follower. If the leader fails, one of the follower will automatically become the new leader. A follower acts as normal consumer, pulls messages and up-dates its own data store.