**TOPIC**

A TOPIC can have multiple partitions and each partition is assigned a broker process, known as its Leader broker.

In order to write to a specific partition, the message needs to be sent to its corresponding leader. The leader takes care of updating its log file as well as replicating that partition to other copies.

It is not possible to change the number of partitions in a topic once it is created.

**Broker**

A kafka broker is the center brain behind everything that kafka does

A broker is nothing but a running Kafka instance.

It is a physical process that runs on the base operating system and executes all Kafka functions.

The Kafka broker receives messages from producers and stores them locally in logs.

Consumers subscribe to specific topics within the Kafka broker. The broker keeps track of all active consumers.

**Kafka brokers manage the life cycle of topics.**

**Kafka Cluster**

Multiple Kafka brokers can be clustered together to form a single Kafka cluster.

Within a Kafka cluster, there is one Kafka broker instance that will act as the active controller for that cluster.

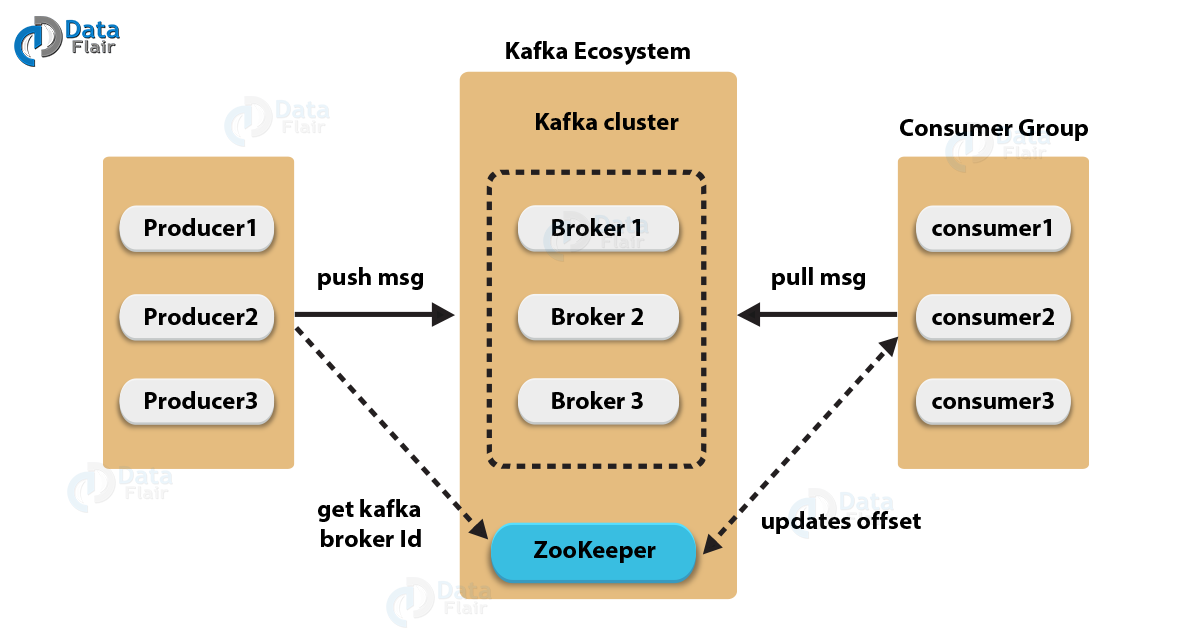
In addition, each partition will have a corresponding Kafka broker as its leader. The leader then manages that specific partition.

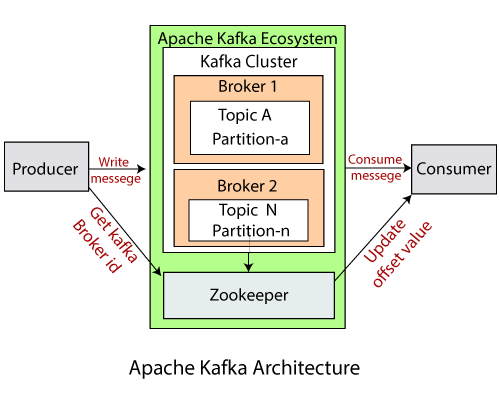
 A Kafka broker also takes care of replicating topic partitions across multiple brokers. So even if one broker goes down, the other brokers can take over the corresponding topic partitions. This provides fall tolerance for Kafka.

**LOGS :-**

 There are multiple log files created in Kafka. Each broker will have its own log territory. And in these directories, there are separate files for each topic and partition. These are rolling files so when a file gets filled up it's rolled over and a new file is created to continue with the logging process. So, each partition will have multiple log files in the log directory.

***Data in Kafka is only kept for a configured interval of time. The default is seven days.***





**Zookeeper**

 ZooKeeper serves as the central information store for Kafka.

ZooKeeper helps Kafka in broker management.

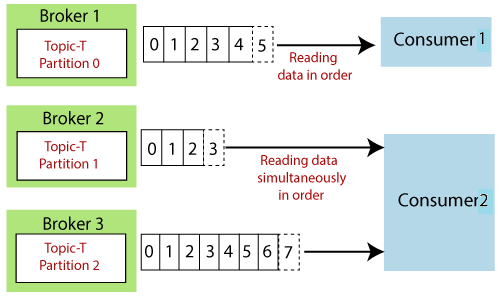
When each Kafka broker starts up, they register themselves with ZooKeeper. They also discover about other Kafka brokers from the same ZooKeeper.

 One of the Kafka brokers, typically the first broker that starts up, registers itself as the active controller. It then controls and manages other brokers in the cluster. If the active controller fails, one of the other brokers will take up that role. This synchronization is handled through ZooKeeper.

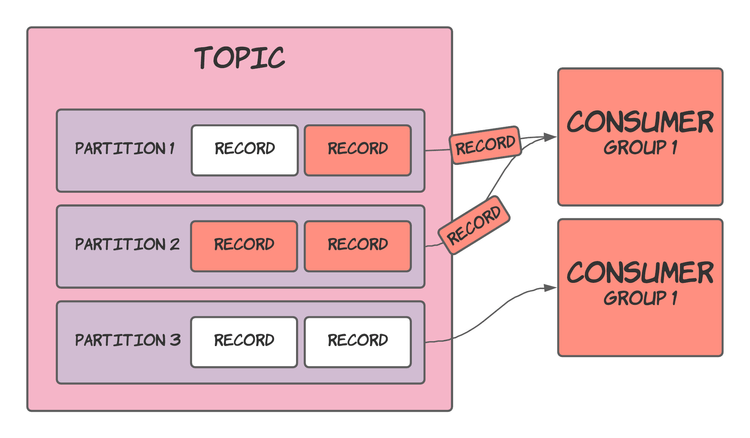
**Consumer Group**

Same set of messages can be accessed by two different consumer groups.

But only one message can be accessed by a consumer within a group.



**Partitions within a single consumer group**



**Shares a topic’s partitions between consumers in the same group**

