Kafka Internals

1. Cluster Membership

Apache Kafka maintains the list of brokers that are currently members of a cluster in Apache Zookeeper so each broker has a unique identifier that is either set in the broker configuration file or automatically generated and registered with the Zookeeper. We get an error if we try to start another broker with the same ID because we already have a zookeeper node for the same broker ID.

2. The Controller

Apache Kafka has the controller node which is responsible to select the partition leader. During the startup up of the Apache Kafka cluster, the node which has started first will act as a controller. It creates an ephemeral node in ZooKeeper. Apart from this if other nodes will try to create a node receive an error that "the node is already present" and the Kafka cluster has already a controller node.

3. Replication

Apache Kafka Replication is the core of Kafka's architecture also Kafka is known for its other features such as distributed, partitioned, and replicated commit log service. In Apache Kafka, the data is organized into topics and each topic is partitioned and each topic has multiple replicas. So in case of any node failures, Kafka makes data available and durable as well.

There are two types of replicas as mentioned below.

3.1 Leader Replica

In this type of replica, every partition has a single replica assigned as a leader. To maintain consistency all produce and consume request goes through the leader only.

3.2 Follower Replica

The replica for a partition, which is not having any leader, is called followers. Followers do not process the request from the client, their main job is to replicate messages from the leader and stay up to date with the most recent messages a leader has. If a leader replica for a partition crashes then one of the followers becomes the new leader of that partition.

4. Request Processing

Apache Kafka broker process requests sent to the partition leaders from clients, partition replicas, and the controller. Clients always start connections and send requests, and the broker processes the requests and responds to them. The broker follows the order in which it receives the file from the client to process it.

All requests sent by the client have a standard header which includes the below detail.

- Request type which is also called an API key.
- Request version so that brokers can handle clients of different versions and respond accordingly.
- The correlation ID is a number that uniquely identifies the request and also appears in the response and the error logs (the ID is used for troubleshooting.
- Client ID is used to identify the application that sent the request.

Once requests are placed on the request queue, then request handler threads are responsible for picking them up and processing them.

The most common types of requests are as below.

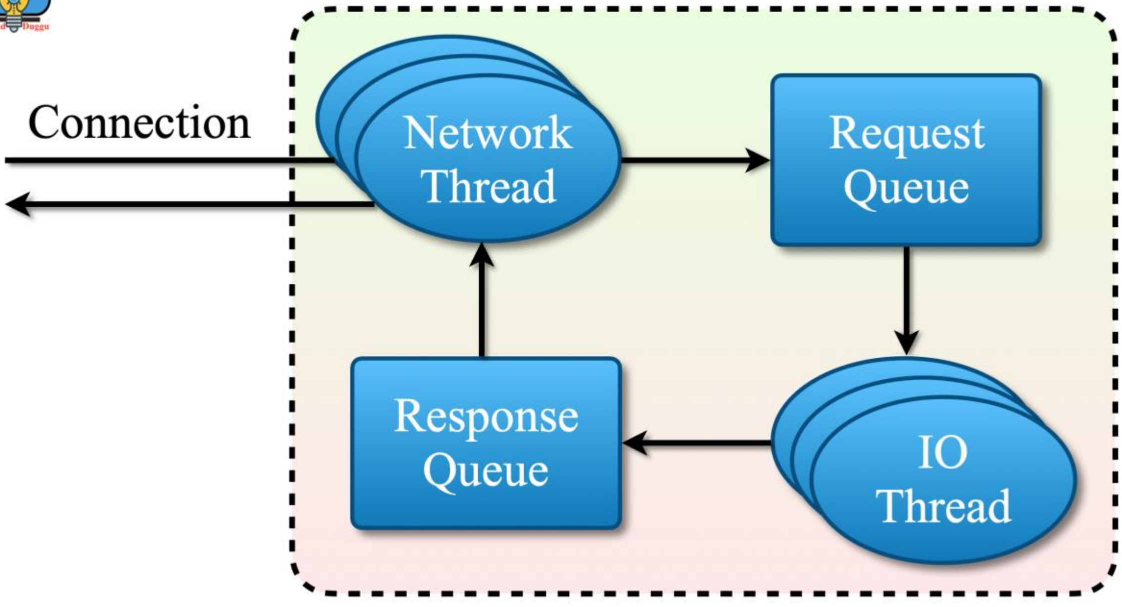
4.1 Produce requests

This request is sent by producers, which contains messages the clients write to Kafka brokers.

4.2 Fetch requests

This request is sent by consumers and follower replicas when they read messages from Kafka brokers.

The following figure shows the request processing inside Apache Kafka.



5. Physical Storage

Apache Kafka's basic unit of storage is a partition replica. Partitions cannot be split between multiple brokers and not even between multiple disks on the same broker and hence the size of the partition is limited on a single mount point. During configuration of Kafka and admin defines the list of directories in which partition will be stored.

Let us see some components of Physical Storage.

5.1 Partition Allocation

If we create a topic then Kafka decides how to allocate the partitions between brokers. For example, if we have 6 brokers and we decide to create a topic with 10 partitions and a replication factor of 3 then Kafka has 30 partition replicas to allocate to 6 brokers.

5.2 File Format

Apache Kafka segments are stored in the data file and the format of data on the disk is alike to the format of the messages which the producer sent to the broker and after that broker sent it to consumers.

5.3 Indexes

Apache Kafka keeps a track of each index that helps the broker to quickly locate the message. In case the index is corrupted then the same is generated by reading the messages.