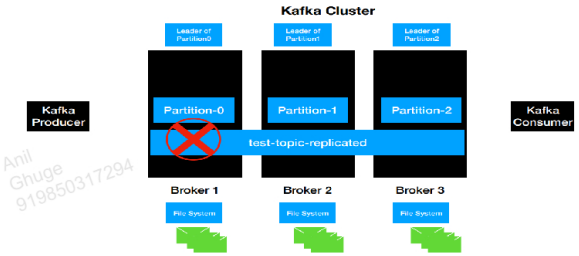
Apache Kafka

Data Loss Handling By Kafka-Replication

Lets have the kafka cluster as below under the representation of the how the topic is distributed accross the cluster and we know in the kafka cluster always have the producers and consumers will talk to the leader of the partition to retrive the data from the broker file system.



Assume now, the broker goes down for a specific reason, which is the leader partiion-0 of broker-1.

All the data which is written to Partion-0 will reside in the file system of the broker-1. once it goes down there will be no way for the clients to access the data. **this is called as Data loss and it is a big problem.**

Kafka Handles this issue with the solution called Replication. we have created the topic using the command

> **.bin/kafka-topics.sh --bootstrap-server localhost:2181**

--create --topic test-topic-replicated

--replication-factor 3

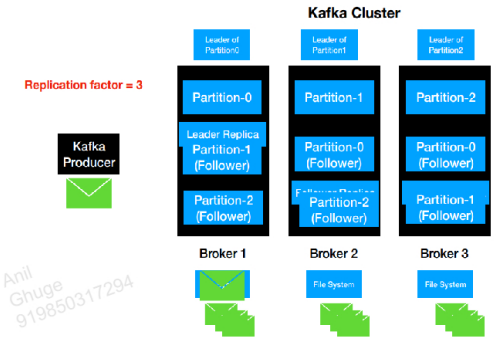
--partitions 3

Replication Factor:

when the replication factor is 3, the sender send the message will goes to the leader partiion-0 of broker-1 and then save into the broker-1 file system. So when we say replication factor is 3 , we need 2 more copies of the same data .

replication factor= no.of copies of the same message.

Apache Kafka



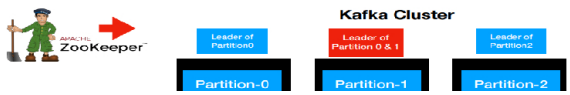
Hence the same copy of the actual message copied onto broker-2 file system now broker-2 is the follower of the partition-0 which is also knows as follower replica.

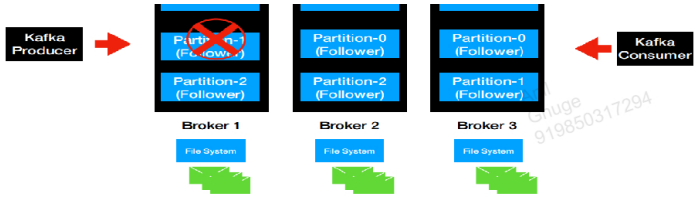
Also the same copy of the actual message copied onto broker-3 file system now broker-3 is the follower of the partition-0.

Now we have the 3 copies of the same data avialble on all the brokers. In kafka terminology this concept is called Replication and the repliaca of the leader is called as leader relica and other two are called as follower replica.

the same concept will be applies to broker-2 and 3 as well. that means any message will be sent to broker-2's leader partition, the message will be saved into the broker-2 file system and the partition of the broker-2 is called as leader replica followed by the other two as follower replica on the remaining brokers where each copy of the broker-2 will be maintained.

Now we have the leader replica for each partition and the follower replicas. Lets say the broker-1 is down, but still the data of the partition is available on the broker-2 and broker-3 partitions.





Apache Kafka

So when the broker-1 goes down, the zookeeper gets notified about it un availbility and immediately the Zookeeper will routes the clients to the next availble broker leader partitions i.e. broker-2 partition.

As already a copy of the message of broker-1 partion is there on broker-2 as well, it just will retrive from there.

In-Sync Replica(ISR)

This will Represents the number of replica in sync with each other in the cluster. **This includes both the leader and follower replica.**

- ✓ The In-Sync replica is always recommended to be greater than 1.
- ✓ The ideal value of ISR == Replication Factor
- ✓ The ISR Can be controlled by the property **min.insync.replias** property. This can be set at the broker level or at the topic level.

The command to see the leader replicas, ISR value is:

>.bin/kafka-topics.bat --bootstrap-server localhost:9092 --describe (or)

>.bin/kafka-topics.bat --bootstrap-server localhost:9092 --describe --topic test-topic-replicated

Lets bring one broker down I.e. broker-1 down and execute the above command again, then we can see the change in value of the ISR.

Fault Tolerance in Kafka

Lets have one zookeeper and 3 broker running. lets open the console producer and console consumer in 2 terminals.

Producer

>.bin/kafka-console-producer.bat --broker-list localhost:9092 --topic test-topic-replicated

Consumer

>.bin/kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test-topic-replicated

Now when we send the message from producer it will received to the consumer as we have the active producer and consumer.

Now lets bring down of the broker. as soon as we did that, we see a warning as Broker May Not be Availble.

Now when we send a message from producer, still the consumer is able to receive the message. So Producer/consumer doesn't know if even some issue going on at the cluster like broker down. Still Producer and Consumer will works as expected. **this is called as Fault Tolerance.**