Points on Kafka Broker-partitions

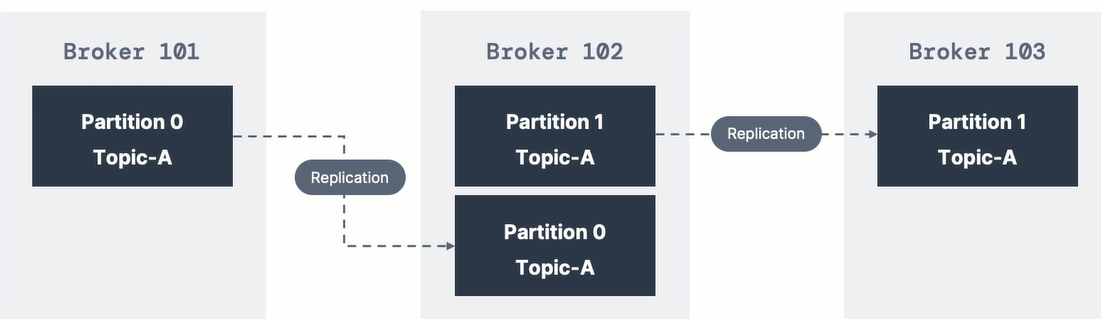
Kafka is always a cluster its set of nodes running on different machines

1. For every message u send u will definitely get acknowledgement
2. And kafkaTemplate.send(msg,callback) and attach a call back u will get a metadata where u can find that message went into which partition number, and what was the offset id for it
3. If u send 1 million messages, u will get 1 million acknowledgements, and attach a callback for each message , for all 1 million messages callback will be fired and u can see partition number and offset id of each stored messages
4. What if leader is down ? always we should maintain some in sync replicas

Means those replicas will continuously ask leader for new data and they will keep themselves in sync with each other. So when leader is down if 1 broker is in In sync replica list then that ISR can become the leader as previous , leader is down , always cfg minimum ISR list =2+

Assignment:- send 1 lakh message and see all the acknowledgements

1. Topics are durable its not like queues once a consumer consumes they are not lost we can retain then, topics stores the data in log files we can configure the retention policy for those messages
2. 1 topic will have many partitions –each partition will be on a separate node to protect from broker failures



If you see the above image if topic-A is having 2 partitions , both partitions are on different node to

1. If all partitions are on same broker, if that broker goes down all partitions will not be available
2. since partitions got splitted and stored even if that broker went down all partitions will not go down

Producer

1. Partition is determined by the key- While sending messages ProducerRecord, send partition num , if u send it will go to that partition only, else it will use the key , Partition number will decide the key same like hash code ,
2. If no key then all messages will be evenly distributed to all partitions in round robin fashion   
   If all messages are having same key they all will be sitting on same partition   
   use case:- lets say if we are sending employee objects send emp objects having the key as employee work location so that all hyd emp data will go to partition -1 , all employee objects having Chennai work location those emp objects will goto partition-2.. so if u want data to flow to certain partitions then prefer sending the key along with the messages
3. If we are sending data we always send to leader broker, consumer should request for the data and he should provide acknowledgement once he properly consumed the message , else producer may think and he may retry and consumer should commit the offset after consuming that message

Questions

1) Can we tune partition size -mostly hyd employee Objects will go to partition-1 , if all data goes to p-1 for hyd then partition should be bigg right

What is an Event in Kafka World?

Event is anything that is happened, manual event or automatic event

When user clicks something it’s also an event or any user interaction is an event

Ex:- transaction is an event, sensor data this is also an event

# Kafka properties

1. **Auto timestamping**There are 2 types of timestamps to which a message can be set

At origin we can set the timestamp

Setting producer time

**Message.timestamp.type=0** this is the default and recommended one means producer will set the time to that message when he is sending to the broker

Setting broker time 1 means , timestamp will be set to the TimeStamp field of ProducerRecord not by us ,by broker it will be set when the message received by the broker

1. Buffer.memory=32MB this is producer buffer memory ,

This is a producer config and can be set to properties obj which we give to producer

Producer.send(ProducerRecord); will actually send the record to producer buffer

IO threads are responsible to send the data from buffer to the cluster

1. Min in sync replicas must be configured =2

**Min**.insync.replicas=2 its like in a team ensuring remaining team members are having equal knowledge as like team leader, so that if team leader goes offline remaining 1 of few members can become next team leader

--config min.insync.replicas=2 this is a topic/ broker config

So while creating a topic u have to add to the existing script

It will make sure always 2 replicas are present in the ISR list, if we don’t do this

Then incase if follower is far behind the broker then they will be removed from the ISR list

And eventhough u cfgd acks=null, since no body is present in ISR list, broker will give ack only when it is received by the broker itself

#story already written in another word file named “kafka fresco notes”

1. Acks=all

Means broker will give back acknowledgement back to the producer only when all the followers in the IN sync replicas also received the message

1. Enable.idempotence=true (this is a producer configuration)

This is only for producer retries, in this case producer while sending the messages it will stamp the producer id and message sequence number , so when producer resends ,he will resend with same id and broker will identify and rejects it

This is for exactly once scenario to avoid duplicates, in atleast scenario there is a chance of duplicate occurances , eventhough message received successfully if broker while sending back acknowledgement if it lost, producer will retry sending the same message which is a duplicate scenario , to avoid that set this flag,

# Practice – assignments

1. Auto timestamping –set the producer time and setting the broker time to the message
2. Practice 03- multi threaded producer which demonstrates 1 kafka producer object is enough , multiple threads can use same kafka producer objects

# Kafka standards

Kafka can process million messages per second with cluster of brokers

1. In an application ,if you are sending 4 messages to kafka make sure you are sending all of them in a transaction
2. Don’t tolerate loss of messages
3. Don’t tolerate any duplicate messaging