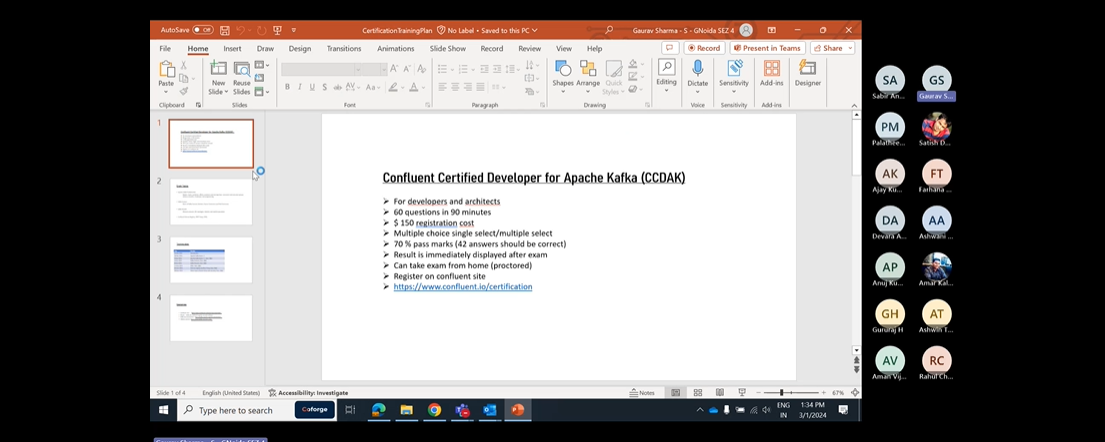
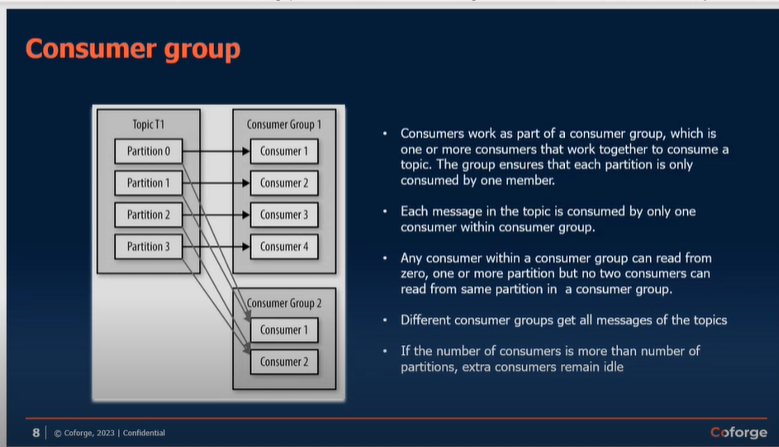
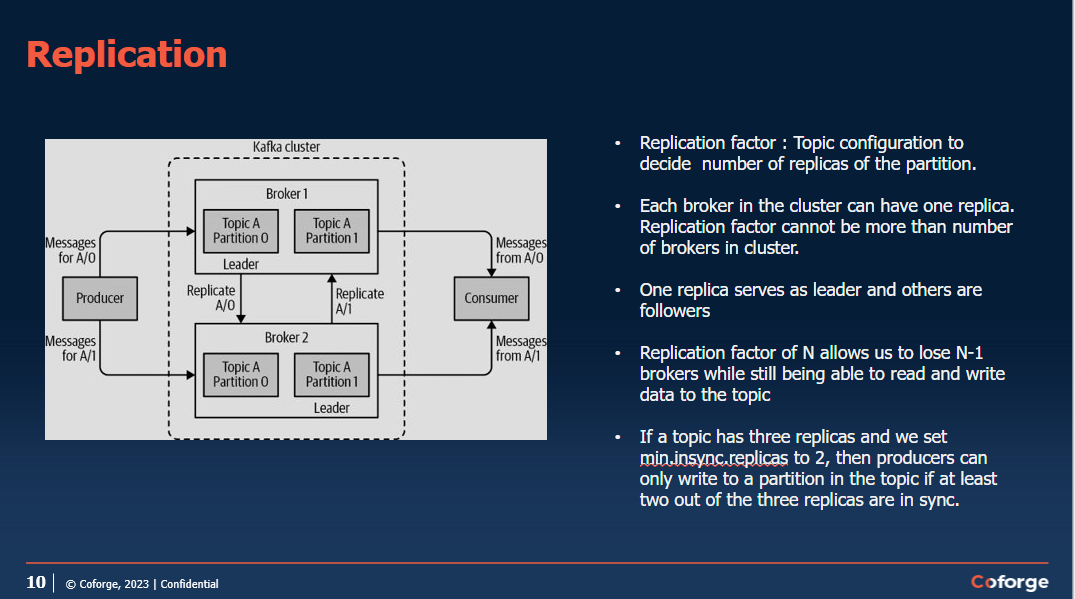
CERTIFIED DEVELOPER FOR KAFKA



One partition can be consumed by only one consumer with in a consumer group.



Replication factor in Kafka-

<https://richiebachala.medium.com/11-steps-for-understanding-confluent-kafka-for-data-engineers-47ab9a398bed>

go through for kafka confluent cluster details.

Zookepers are always odd numbered in Kafka Cluster.for normal organisation count is 3 and for very large organisation count can be 5.

# REST Proxy-

It is used to expose API to expose rest end point to external API’s. we cannot give access directly external applications for the kafka cluster.

Rest proxy so rest proxy is basically it will just provide the rest API OK and when the applications invoke that those rest API's they can post the messages to Kafka broker or they can just read from Kafka broker. So those kind of features that we provide rest proxy.

OK, so they don't have to connect to the Kafka broker, they can just directly put it to this API and internally this rest proxy will send it to the desired topic. OK we I will see this in details in next sessions

# Confluent KSQL-

When you want to copy some data from the topic and paste it to another topic via queries. It is done

With the help of KSQl.

Go through these things at-<https://richiebachala.medium.com/11-steps-for-understanding-confluent-kafka-for-data-engineers-47ab9a398bed>

Kafka Connect-

Kafka Streams App-

Microservice-

Confluent Control Center-

What is the use of Schema Registry-

Schema Registry is in the Avro format i.e. it will have data type and variable.in json we do not have any data types.

OK. Again, it depends on the producing and consuming applications which schema registry like if they want to use schema registry or not. OK suppose you serialise your message so whenever you create a record OK and you want to send it to Kafka so there are two things you need to do. One is you need to make sure that it is in proper format. OK so that consuming application can do. Second thing is you you need the Kafka has to convert it to.

A stream of bytes. OK, serialise it. OK, so you need to mention the configuration property, key serializer and value serializers. OK. There you will mention Avro and you need to give the schema registry URL. So that means it will be serialised according to your Avro schema. Whatever you have mentioned in the schema registry. OK.

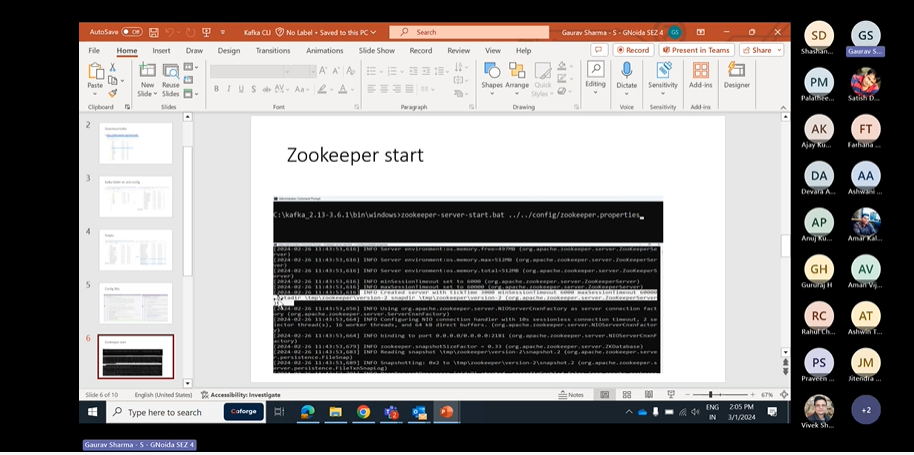
Use of brokers-

* The broker receives messages from producers, assigns offsets to them, and writes the messages to storage on disk.

Controller is selected by a zookeeper.

* The controller is responsible for administrative operations, including assigning partitions to brokers and monitoring for broker failures.
* A partition is owned by a single broker in the cluster, and that broker is called the leader of the partition. Replicated partition is assigned to additional brokers, called followers of the partition

All producers must connect to the leader to publish messages, but consumers may fetch from either the leader or one of the followers.



Zookeeper can be started anytime with the properties configured.

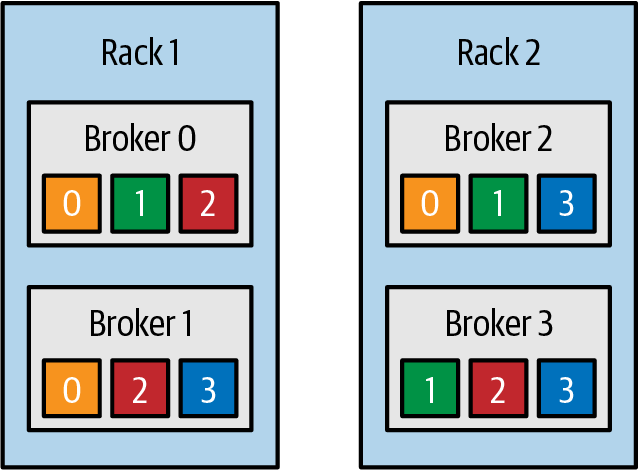
* Zookeeper cluster is called ensemble.
* Odd number of servers (typically 3 or 5)
* All servers need to have common configuration file
* Quorum : majority of the servers should be running to form a quorum
* Configurations:
  + tickTime:2000
  + dataDir:/var/lib/zookeeper
  + clientPort:2181
  + Server1 : hostname: peerPort: leaderPort
  + Server2
  + Server3
  + initTime : the amount of time to allow followers to connect
  + syncLimit : value limits how long out-of-sync followers can be with the leader.

# Cluster membership

* Each broker has a numeric broker id , mentioned in the config file.
* Broker registers itself with the zookeeper using this id creating ephemeral node
* Each broker in a cluster has unique id
* The first broker to register itself with the zookeeper becomes controller.
* If controller goes down, other brokers attempt to become controller
* Controller helps in selecting leader partitions.
* Zookeeper selects controller and stores metadata (brokers, topics, partition, replicas)

# Partition allocation

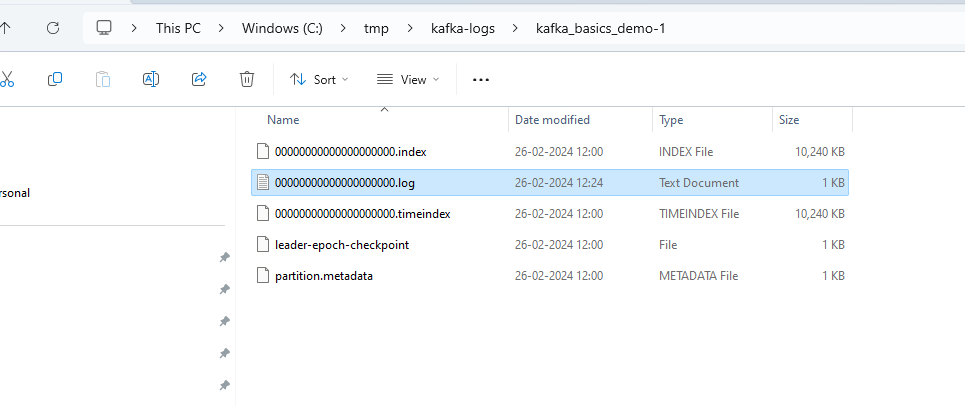
* Assume 4 brokers, new topic with 4 partitions and a replication factor of 3

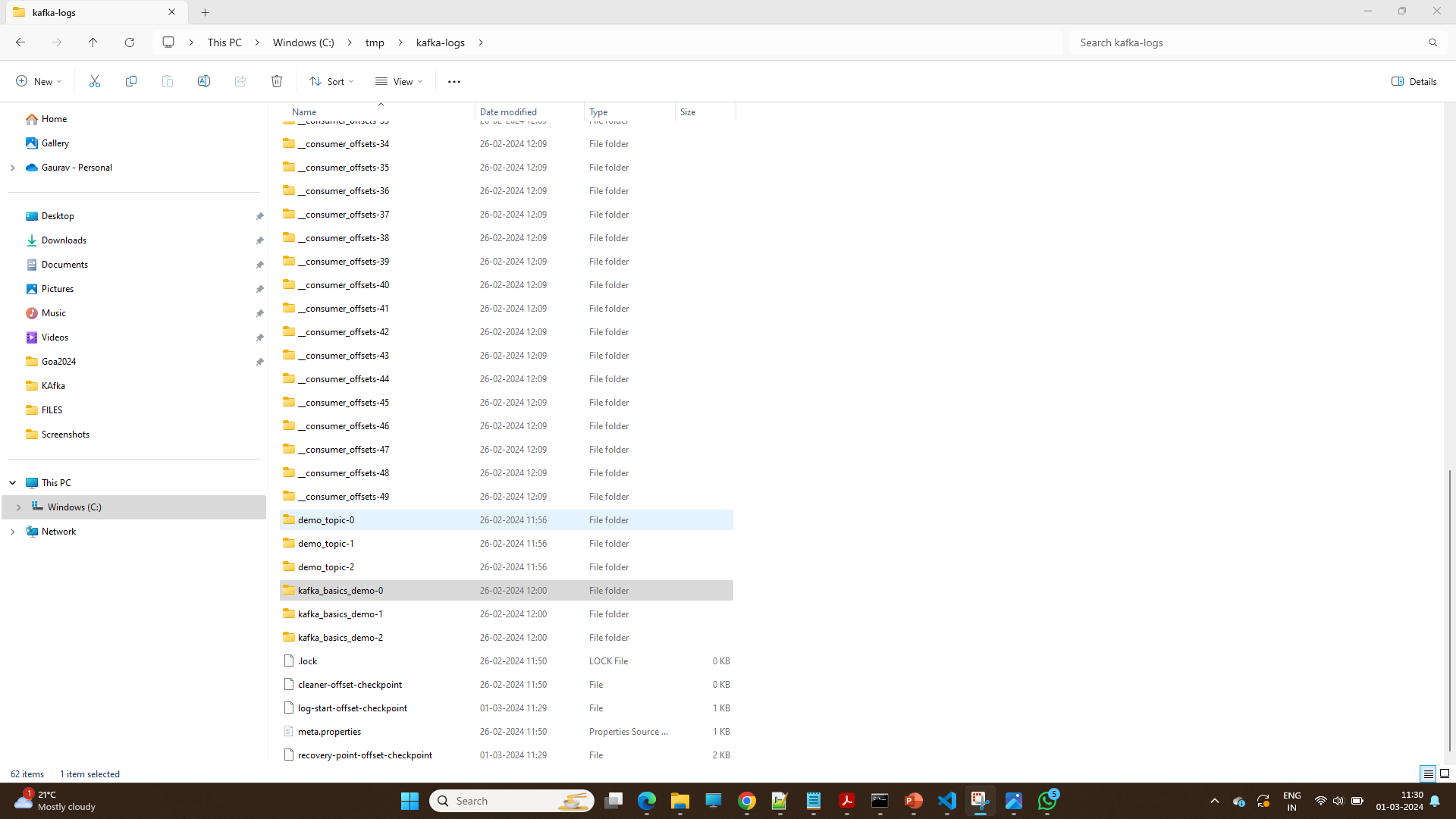


Topic is divided into partitions, partition is divided into segments for the purpose of writing data to the disk. Segment file can be retained for time duration or based size limit.

Active segment is not deleted.

If fewer messages are expected, retention should be maintained on time basis

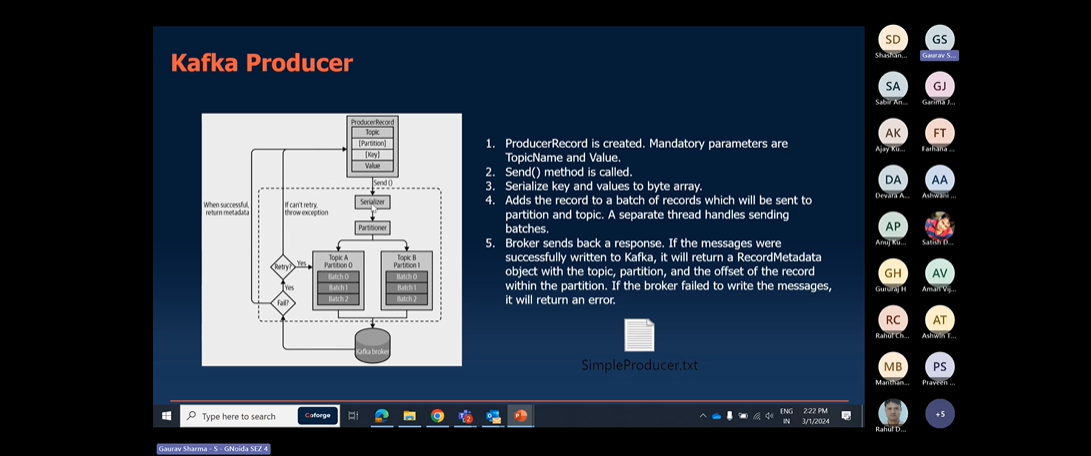




In kafka Topic is divided into partitions, partition is divided into segments for the purpose of writing data to the disk. Data is stored in the form of segments once the segment is complete to specific space.it will create another segments to store the data.

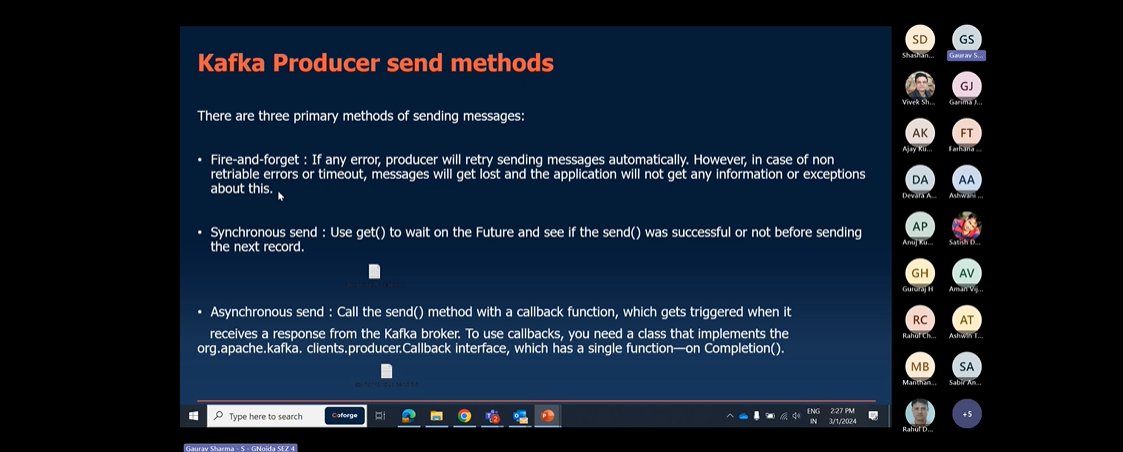
If producer is slow ,we can configure it will create another topic after specific time interval.

# KAFKA Producer-



Retry Mechanisms-

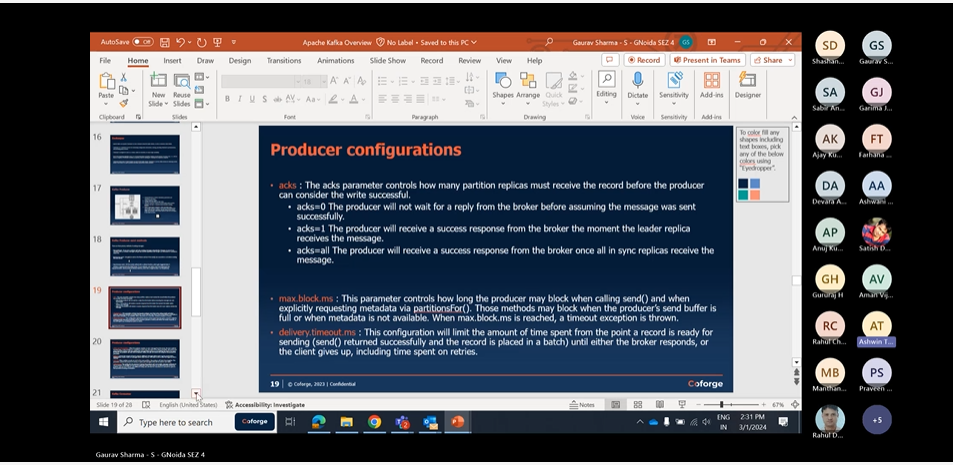




Fire-forget🡪send msg and forget ,if data lost it is not observed by our app.

Synchronous send🡪single thread mechanism it will wait until the acknowlwdgement received from kafka.

Asynchronous send🡪multithreaded mechanism , one thread will pass by and another thread will wait for Acknowledgement.

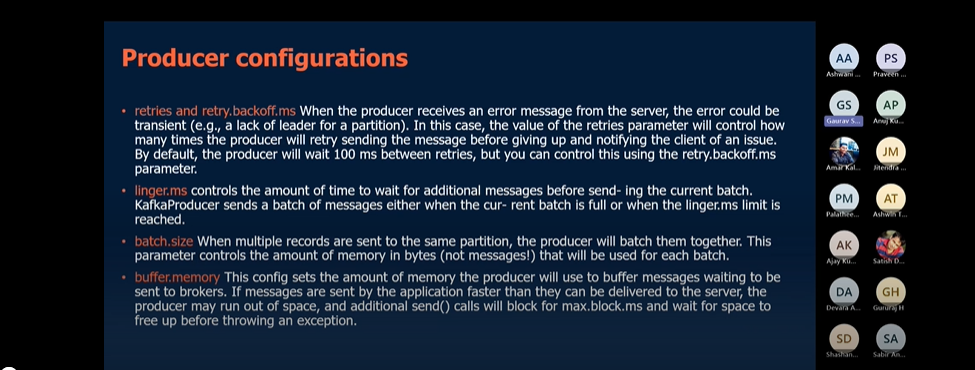


“Not Enough Replica”-

Like as if we want replica to backup. and we have Replication factor:4 and minimum insync replicas :3 and we have one of our broker down ,so it will not be possible for us to get acknowledgement giving “ “Not Enough Replica” error.

reduce value:”insync replicas “ .it will work

# Producer Configurations



Retry.backoff.ms🡪time between each retry.

Linger.ms🡪sometimes the producer is too slow and the producer will wait for this much time only before pushing messages to kafka.if this much time is passed it will push the messages.

Batch size🡪if your producer is fast than you can increase the batch size and you can push messages at faster rate.

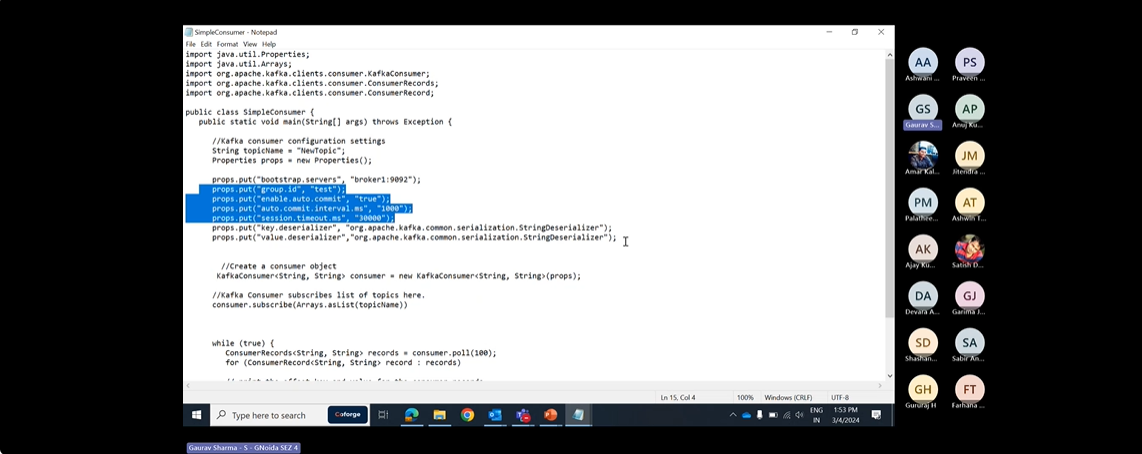
Kafka Consumer:-follow multiple configurations , without which it will not start the kafka Consumer.

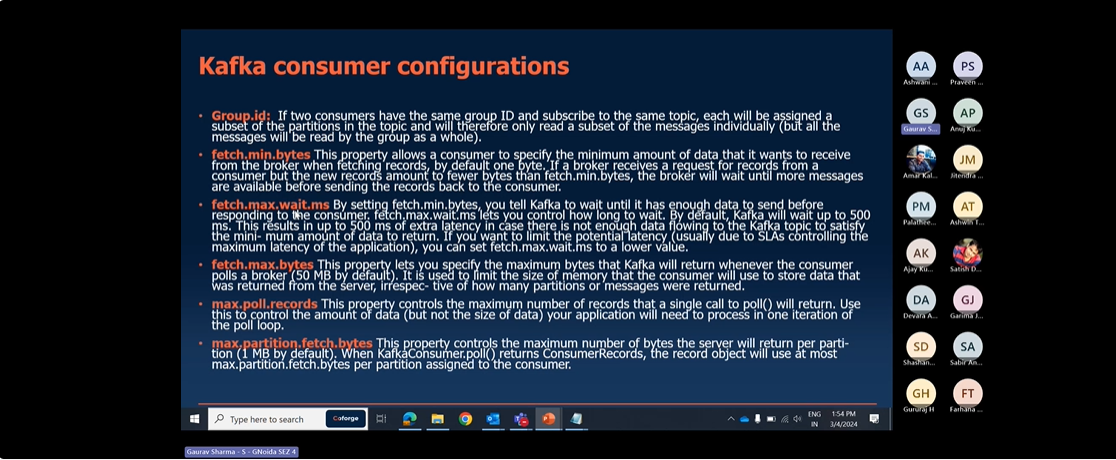
BootstrapServers

Key.deserializer

Value.deserializer

Kafka Consumer have following properties-:



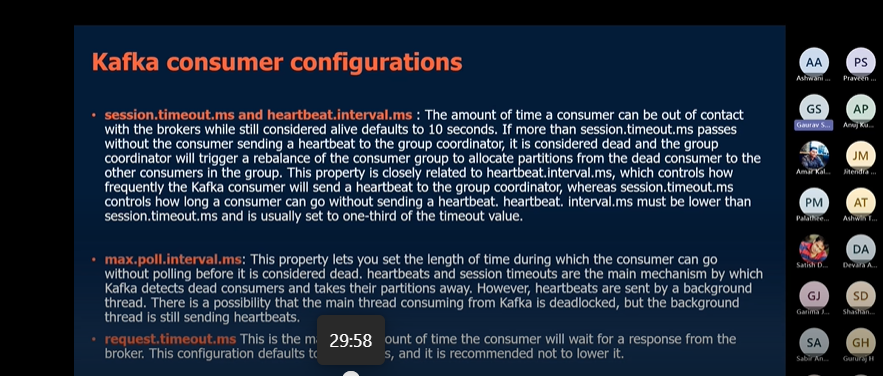


Fetch.min.bytes:-by default it is 1 byte.if a broker gets request,but a new records amount less than fetch.min.bytes.broker will wait for for more Records before sending.

Fetch.max.wait.ms-since the broker waits for records to come because of fetch.min.bytes it increase the lag.by default wait time is 500 ms .we can reduce it as per our wish.

Fetch.max.bytes-it specifies the maximum bytes that kafka will return whenever consumer polls a broker.it is used to limit the size of a memory that consumer will use to store the data that was returned from server.

Max.partition.fetch.bytes-it specifies the maximum number of bytes server return per partition.



Session.timeout.ms -The amount of time a consumer can be out of contact with brokers while still alive defaults to 10 seconds.if more than session.timeout.ms passes without the consumer sending a heartbeat to group coordinator,it is considerd dead and the group group coordinator will trigger a rebalance of a consumer group to allocate partitions from dead consumer to other consumers.

heartbeat.interval.ms -it controls how frequently kafka consumer will send heartbeat to group coordinator.

Generally Session.timeout.ms and heartbeat.interval.ms are closely related to each other. Session.timeout.ms basically defines the time in which consumer can go without heartbeat.it is generally Session.timeout.ms set to 3 times heartbeat.interval.ms

Max.poll.interval.ms-

Lets you set the length of time during which consumer can go without polling before it is considerd dead. Heartbeat and Session timeout are the main mechanisms by which kafka detects the dead consumer and takes their partition away.and there can be scenarios in which main thread is deadlocked and background thread is sending heartbeats.

Request.timeout.ms:-maximum amount of time consumer will wait for response from broker. The

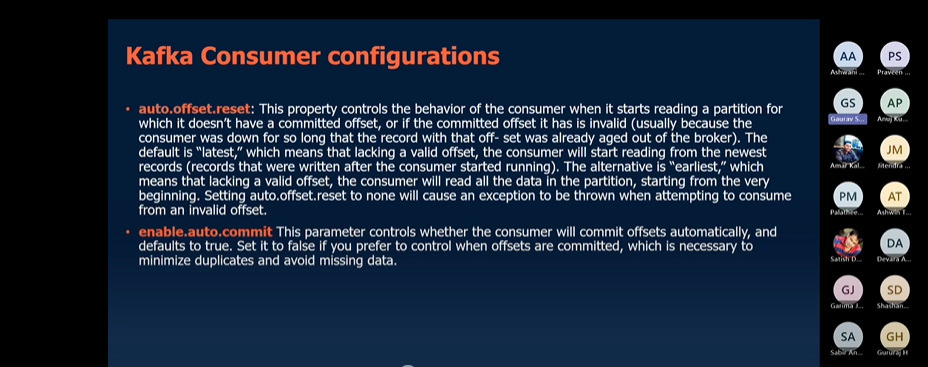
Configuration defaults to 30 s.it is recommended to lower it.

Auto.offset.reset-it controls the behavior of consumer when it starts reading a partition which doesnot have commited offset or if the offset it has commited is invalid.

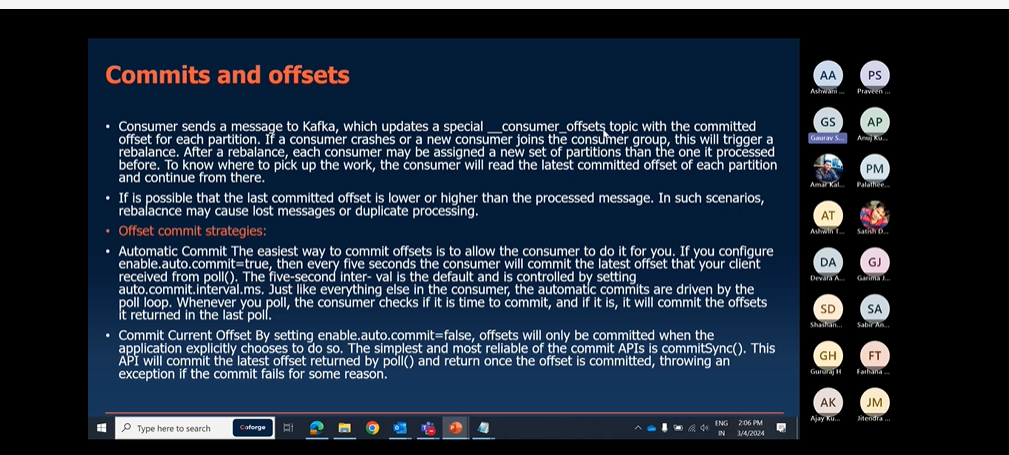
Earliest:means it starts reading from earliest offset

Latest: means it starts reading from the latest offset or new available offset

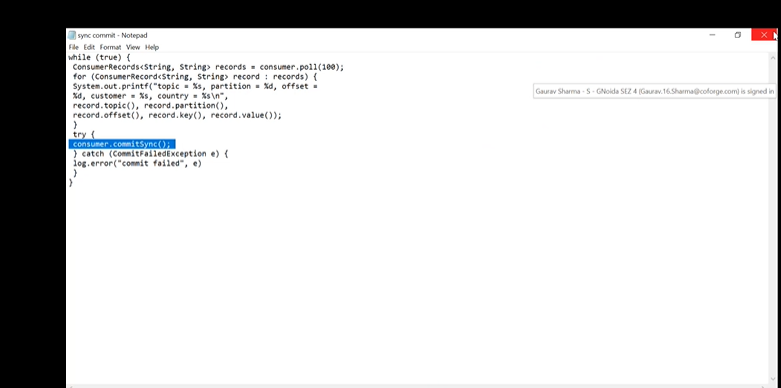
None:if we set it to none .it will automatically give an exception and consumer will be stopped automatically.

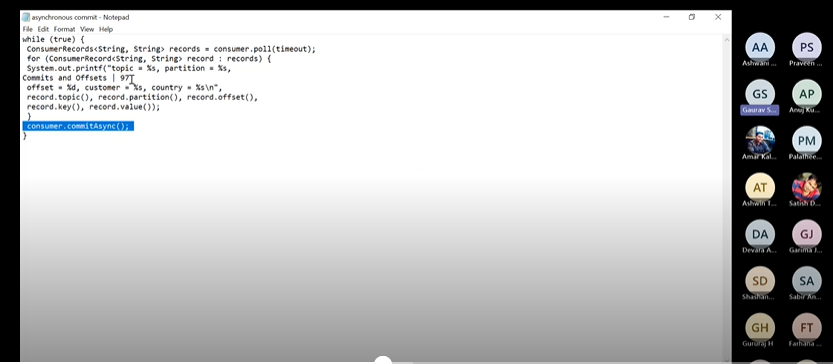


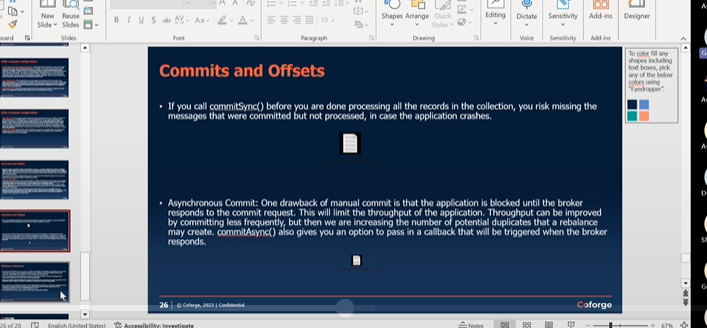
Enable.auto.commit:-either you will commit automatically or you will commit manually within code.



CommitSync-means you are Synchronously commiting the offset.In this we have risk of losing the records i.e commiting the records before processing.



Asynchronous Commit-:one drawback of manual commit is that application is blocked until the broker responds to commit request.This will limit throughput of the application.Throughput can be improved by commiting less frequently.



Rebalance:-

