Apache Kafka

Types of data pipelines

1. Batch data pipeline → Spark, Hive
2. Near real time data pipeline → Spark Streaming
3. Real time data pipeline → Event driven (Apache Flink)

where data needs to be computed in the batches or some kind of historical data → batch data pipeline

frequency

1. Daily
2. Weekly
3. Monthly

wherever we process the data in micro batches then it's kind of NRT.

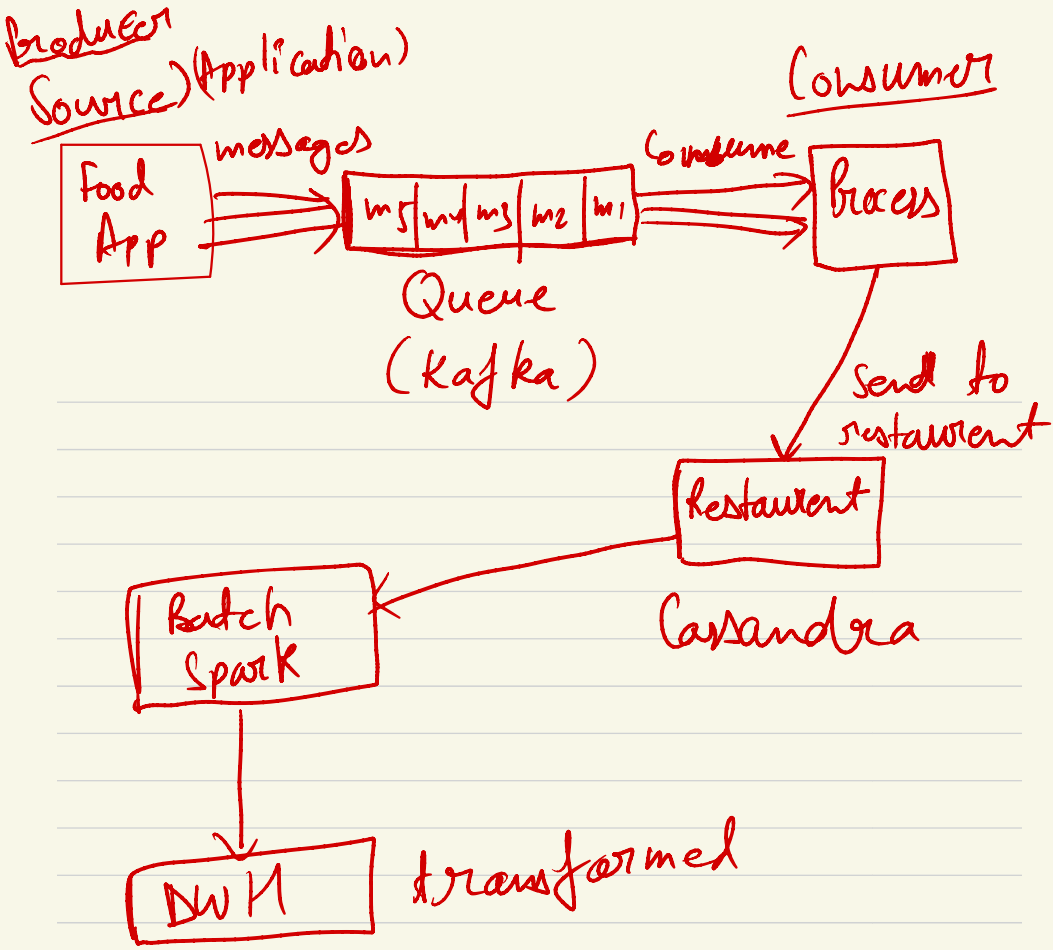
Real time Data pipeline

- Completely event driven
- happens in fraction of seconds
- Example → Database CDC events
  ⇓
  Change data capture
- live leaderboard
- Twitter streaming data
- Bank transcation
- fraud detection
- Cricket scoreboard

In order to build real time data pipelines we need messaging Queue kind of frameworks, so that data can be holded for some time

Apache Kafka (Messaging Queue)

- Holds data for real time process
- Messaging Queue kind of system
- Invented at Linkedin.

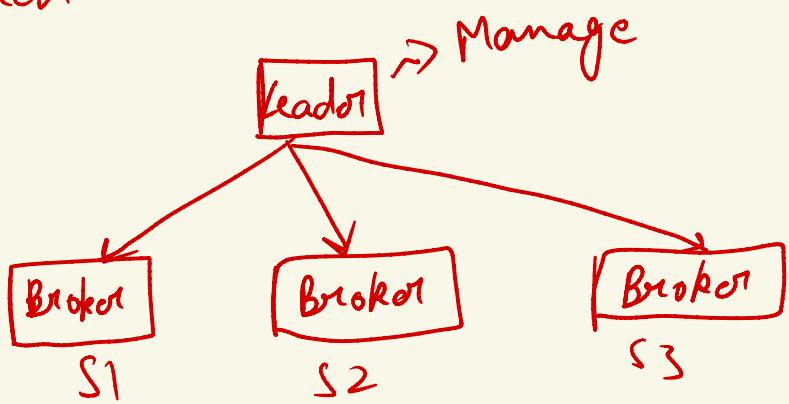


Kafka Components

- Producer ⇒ Application which publishes data in Kafka
- Consumer ⇒ Application which consumes from Kafka
- Cluster → Kafka is a distributed system
- Broker
- Topic
- Partitions
- Replicas
- Offset
- Commits → Sync / async
- Consumer Groups
- Parallelism in Kafka
- Backpressure

Kafka Architecture

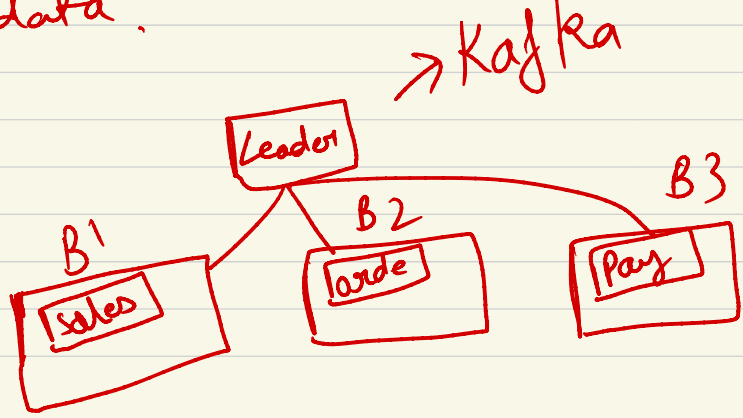
↳ It is basically master-slave arch



Broker → Node in Kafka cluster where actual data gets stored.

Topics → Small segments or a space given within Kafka to store the data.

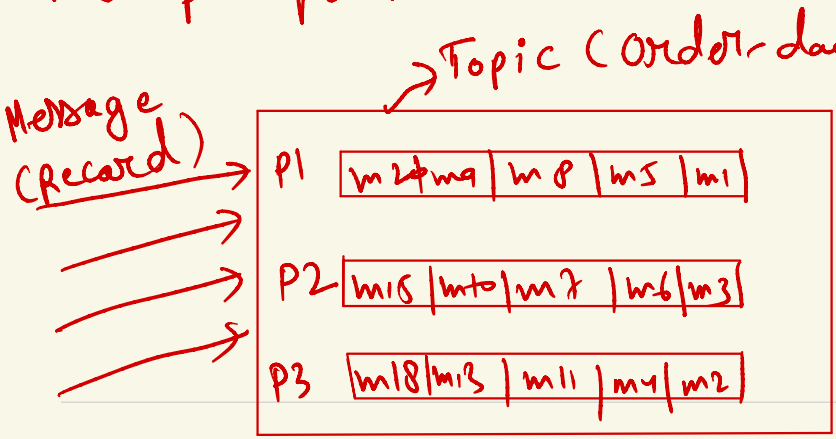
E-comm data
↳ Sales
↳ Sales-order
↳ Payment
↳ Return



Partitions

↳ Dividing the data in multiple parts inside a topic.

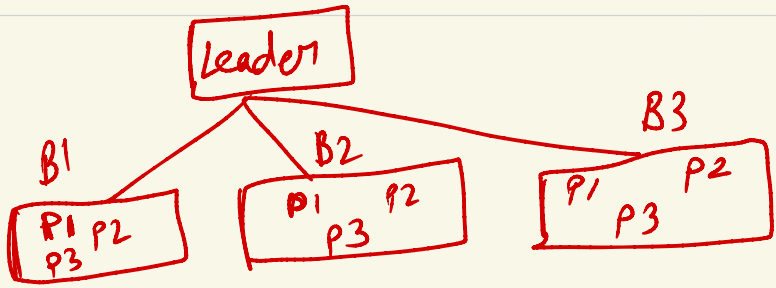
each Topic will be divided in multiple partitions.



in a partition messages will be maitained on their arrival sequence based on timestamp.

on Topic level sequencing is not there.

Replicas => multiple copies of Topic's partitions



Replica factor is 3, for fault tolerance.

Key, Value messages in Kafka

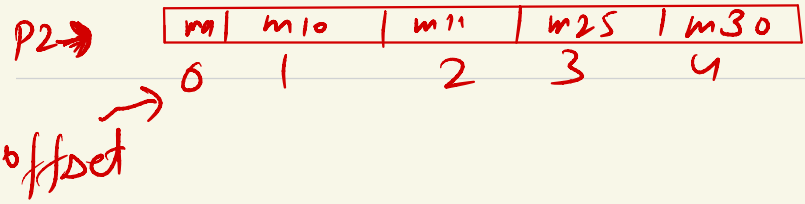
- Does data skewness problem exist in Kafka?
- We publish data in key value form

Offsets: Unique id number for a message within a partition

P1 ->	m1	m2	m3	m4	m5	m6
	0	1	2	3	4	5

offset ~~~> 0

m has been published to offset [0]



Commit in Kafka

↳ mechanism in kafka to maitain or keep a track of messages consumed from kafka topic.

We have topic named as order_data here internally kafka will maintain some information under <some>_orderdata and it will start storing the updated offset from which data was consumed and consumer application made a successfull commit.

P1 →

m5	m8	m15	m3	m9	m10
0	1	2	3	4	5

Consumer application (C)

after first read commit will happen and offset 0 will be maintained in the metadata.

now in second read data will be consumed from offset 1.

sync commit (Blocking Call)

→ async Commit (Response may come later)

Ex: Sync Commit

message:

Step A
Step B
sync-commit()

x↳ Step C

→ wherever we are concerned about data consistency then we should use sync-commit

→ impact will be on latency i.e; high latency

async-commit

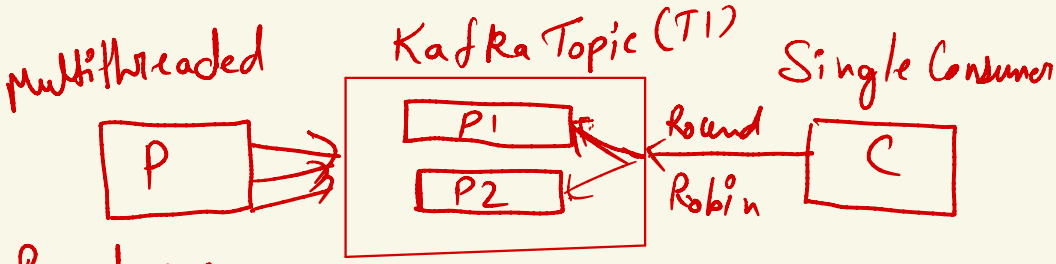
Ex:

- Step A
- Step B
- async-commit (Fire & Forget)
- Step C

→ advantage is low latency

→ Dis-advantage is data inconsistency.

Parallelism & Consumer Groups

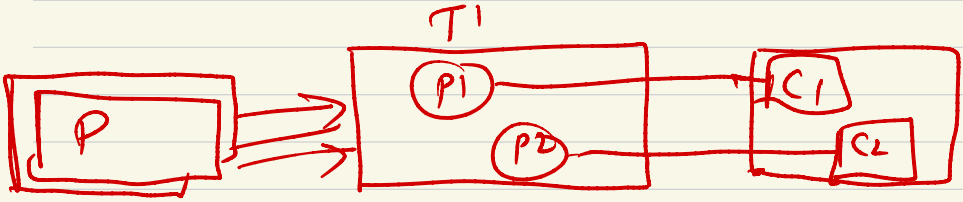


Producer

Production speed is higher than consumption.

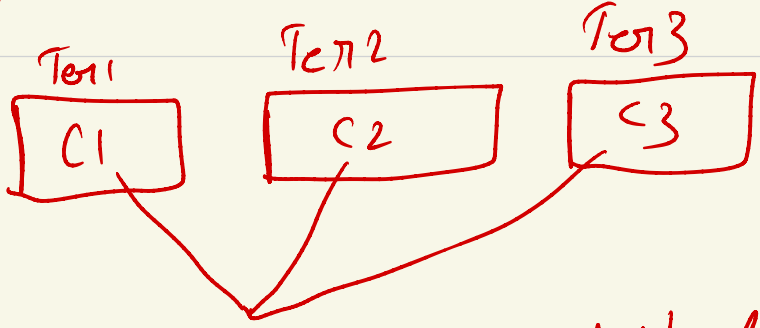
In this case backpressure will hit.

To solve this we should scale our application.



From one partition only one consumer of same group will be able to

Read the data.



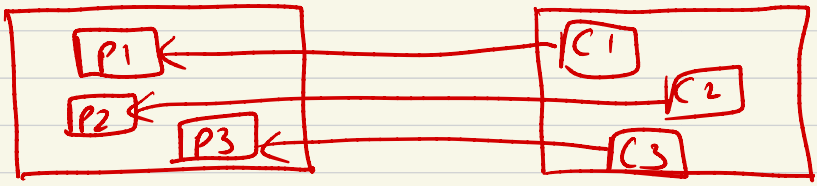
All three consumers should have the same consumer group id (custom name)

C1 → consume(topic, group_id = CKF_1)

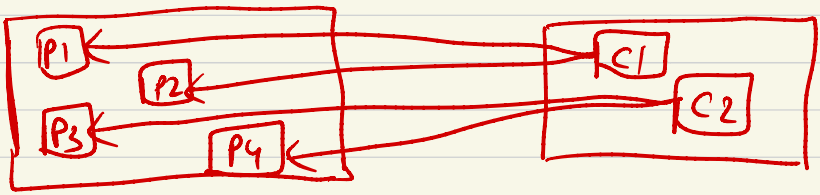
C2 → consume(topic, g_id = CKF_1)

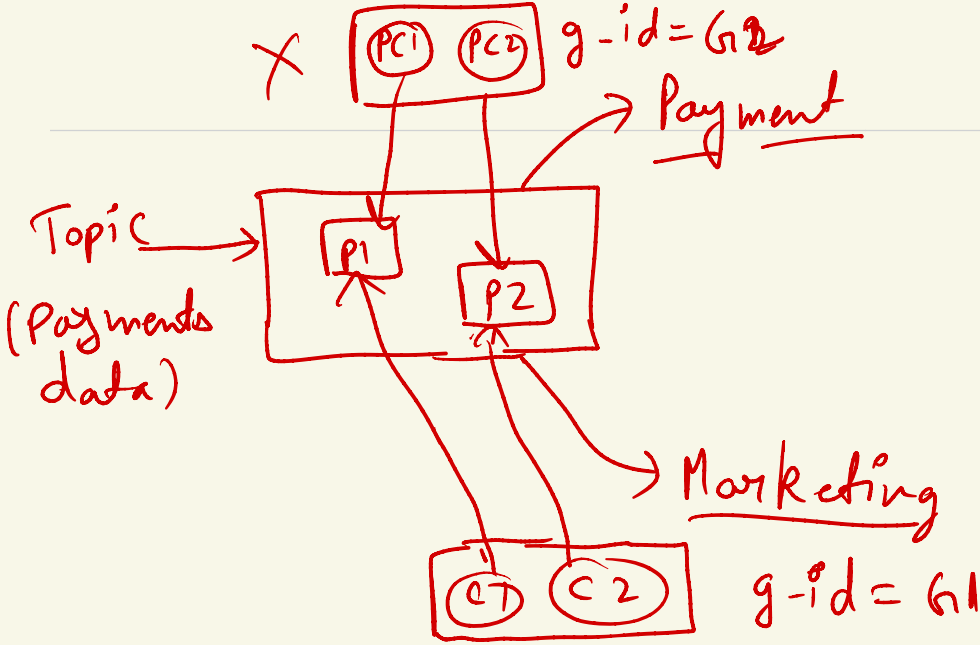
C3 → "

1



2





Parallelism

↳ We should increase the number of partitions.