Types of Datapipelines

① Batch Data processing → wait for Sometime → Hadoop, Hive, Apache Spark *
② Near Real Time (NRT) * 30 Sec, 1 Min, 2 min
③ Real Time Data Processing * Instantly

→ Hours
→ Day
→ Week
→ Month
→ Years
→ Batch Data pipeline
Wait time (threshold)

* Messaging Queues for Real Time Data Process

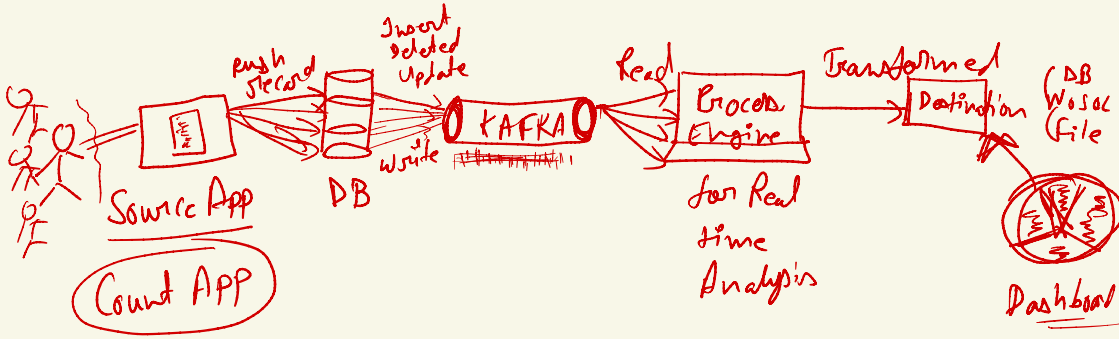
KAFKA
→ Hold the data for Realtime process
→ messaging system for stream process
→ LinkedIn

Real Time processes

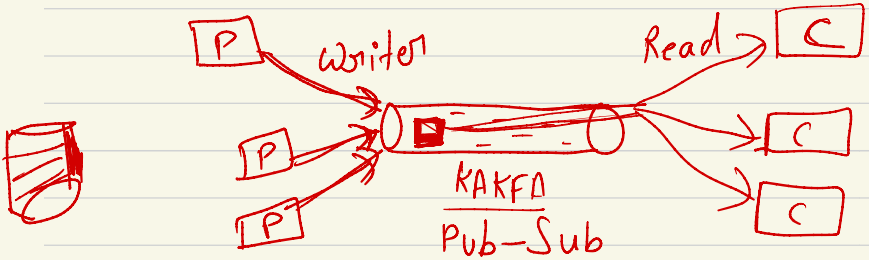
① Bank Transaction
② Application for Live cricket scores and aggregations
③ Gaming Application (live tournaments)

④ Trending Twitter Hashtags

⑤ Any type of real time dashboards
- ↳ Stock exchange Apps
- ↳ Live Count in a mall



KAFKA Basic Actors
- ↳ Producer *
- ↳ Consumer *
- ↳ Connector (Practical)



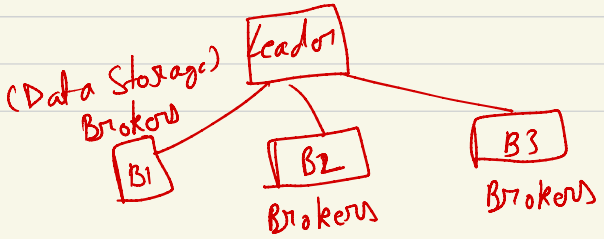
→ Producers are publishing Data
→ Consumers basically subscribe

KAFKA Components

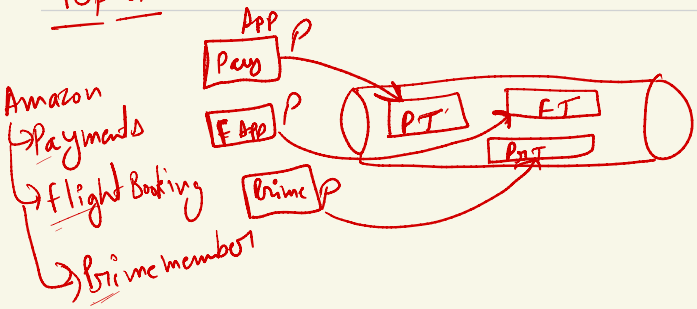
- Broker
- Topics
- Partitions
- Replicas
- Offset
- Commits
  - Sync
  - Async
- Consumer Groups
- Parallelism in KAFKA
- Backpressure

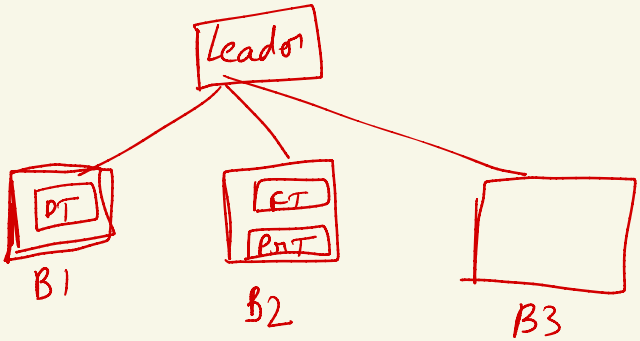
KAFKA Architecture:

- KAFKA Cluster
- Brokers means servers in a cluster



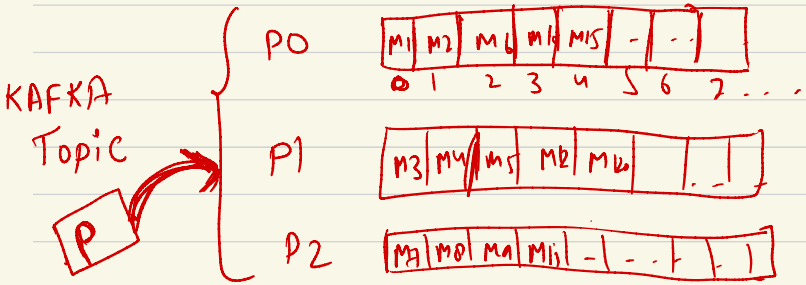
Topics → Small segments of A space name given within KAFKA

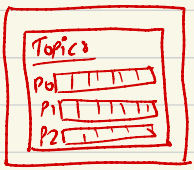




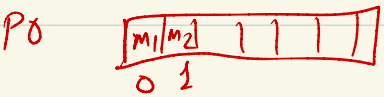
Partitions & Message = Record

→ Current Partition Index = Offset



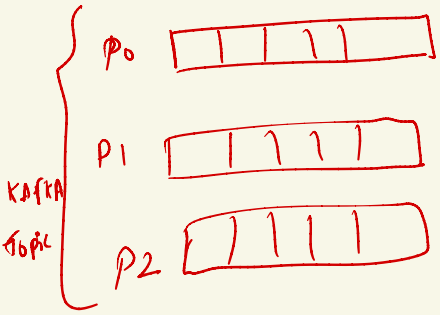


offset = 1

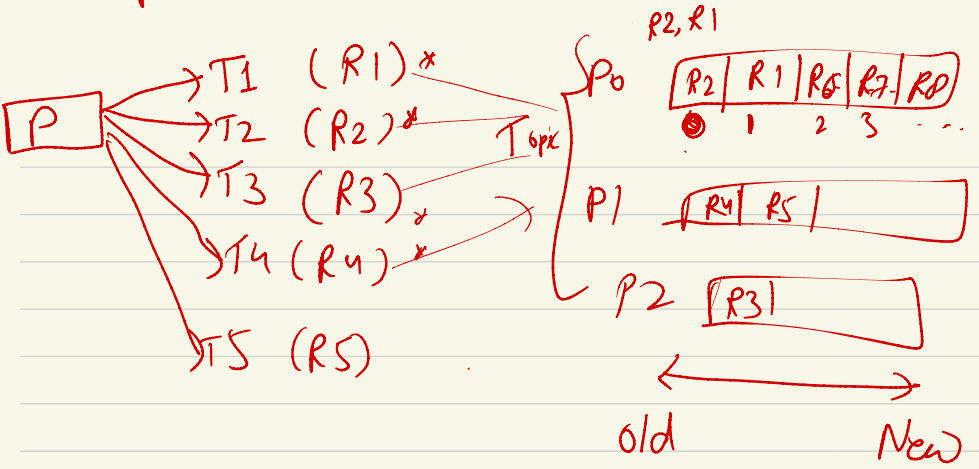


Offset - 1

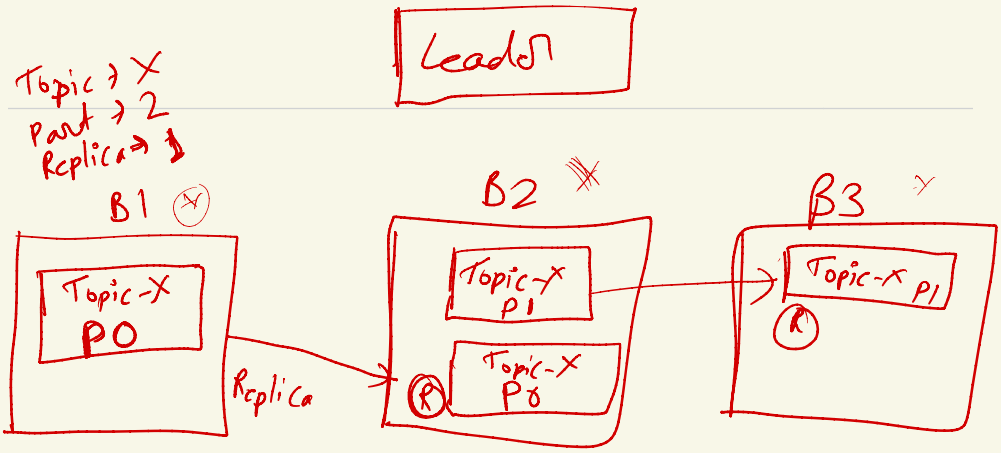
↳ unique id number for a message within a partition

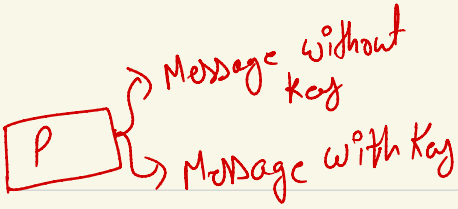


Order of messages will be maintained within a partition



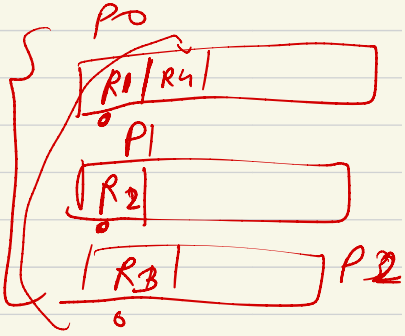
Partitions
- ↳ fault tolerance
- ↳ Parallelism





R1 → {key, R1} / {R}

KAFKA Topic

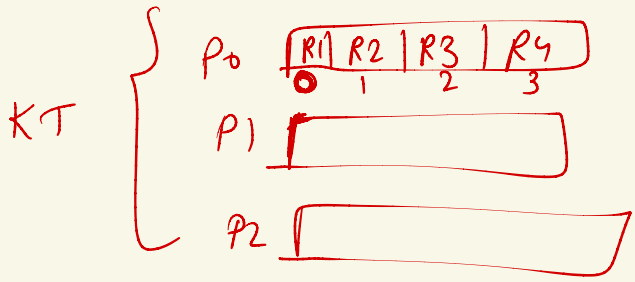


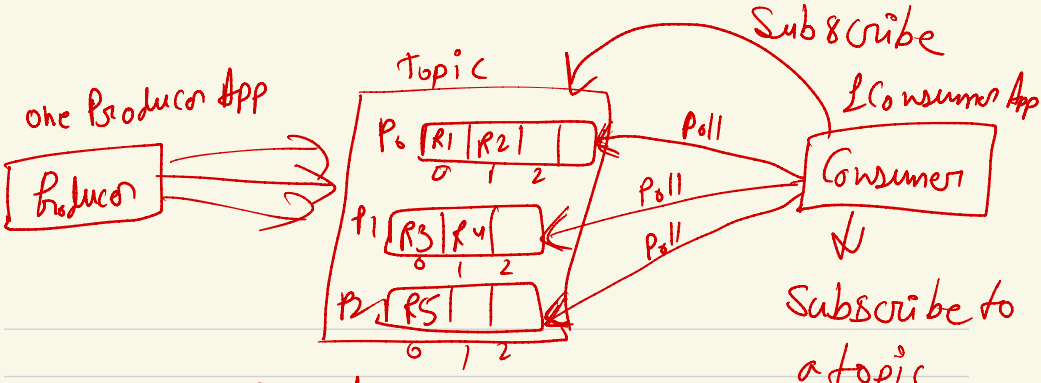
{6, R1} → Key = 6, 6%3 = 0

{8, R1} → 8%3 = 2

{7, R1} → 7%3 = 1

{6, R1} , {6, R2} , {6, R3} {6, R4}



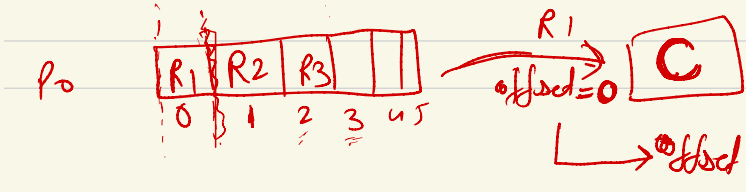


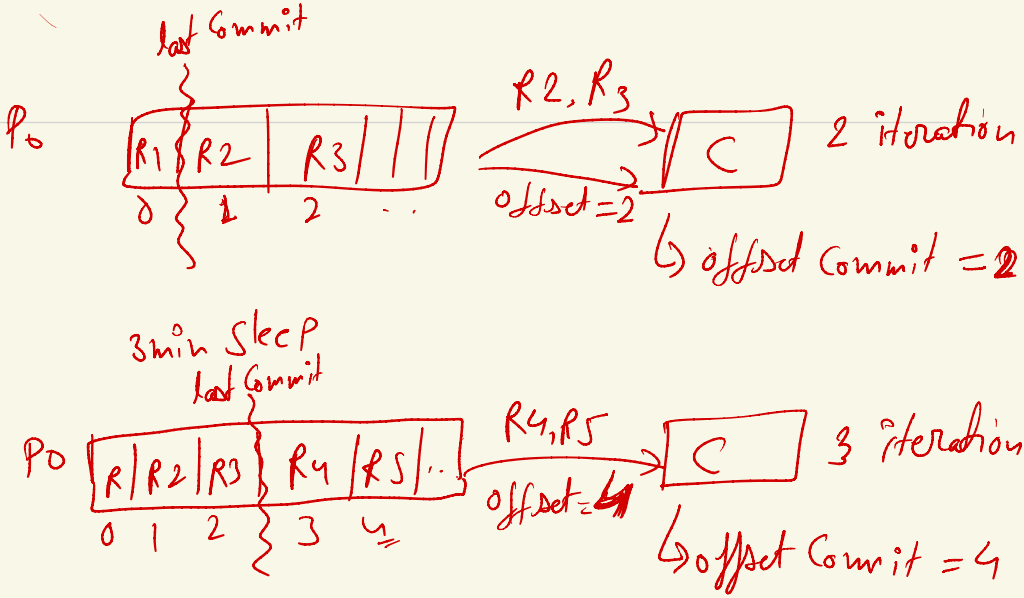
Consumer Properties

→ Latest (Read data from each partition from current offset)

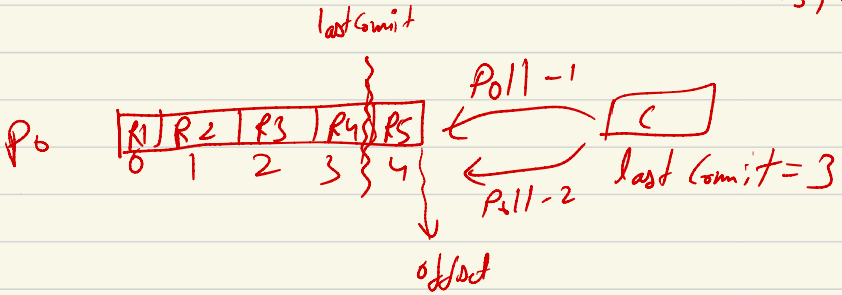
→ Earliest (Read for each partition from start offset)

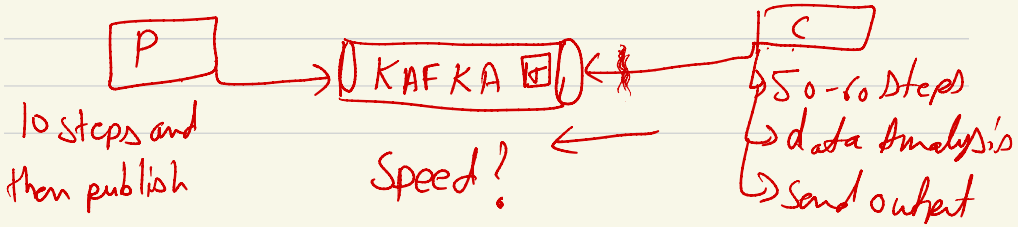
Consumer needs to do the offset Commit



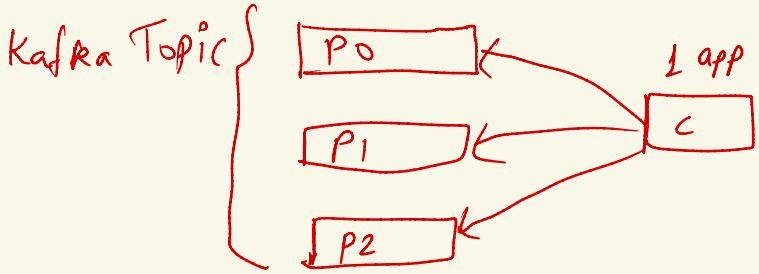


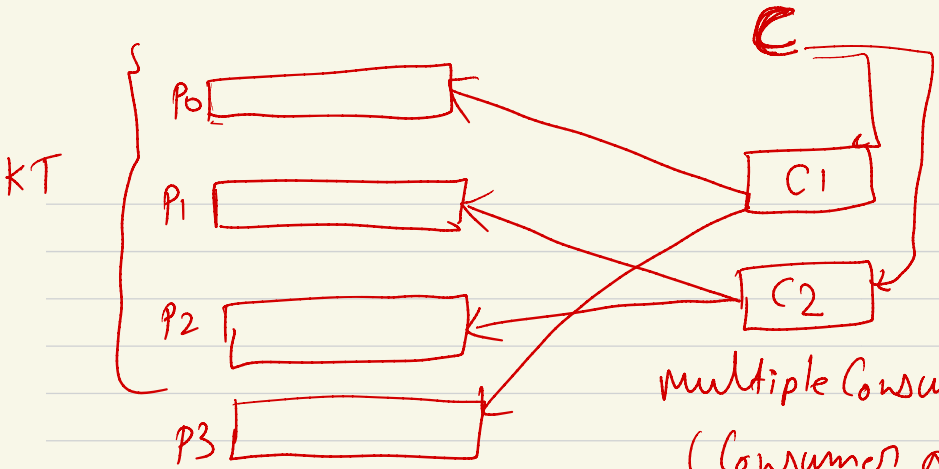
Consumer_0 = { 1 → 0, 2 → 2, 3 → 4 }





Backpressure } ⇒ because of Speed mismatch



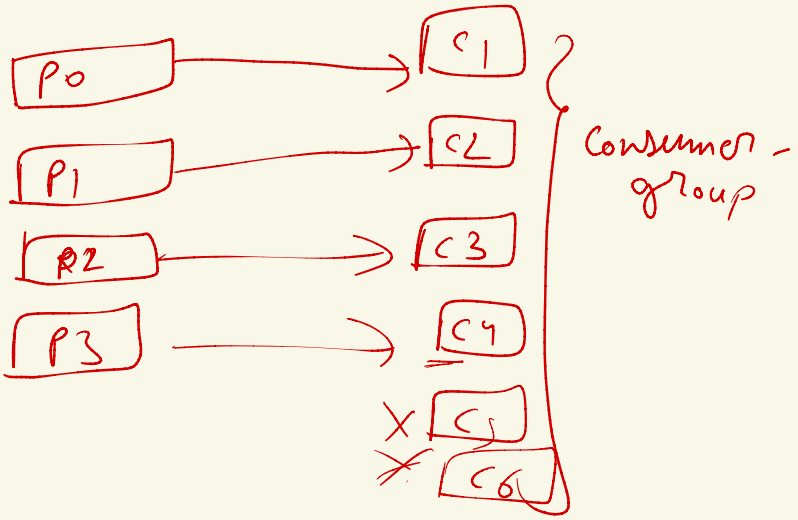


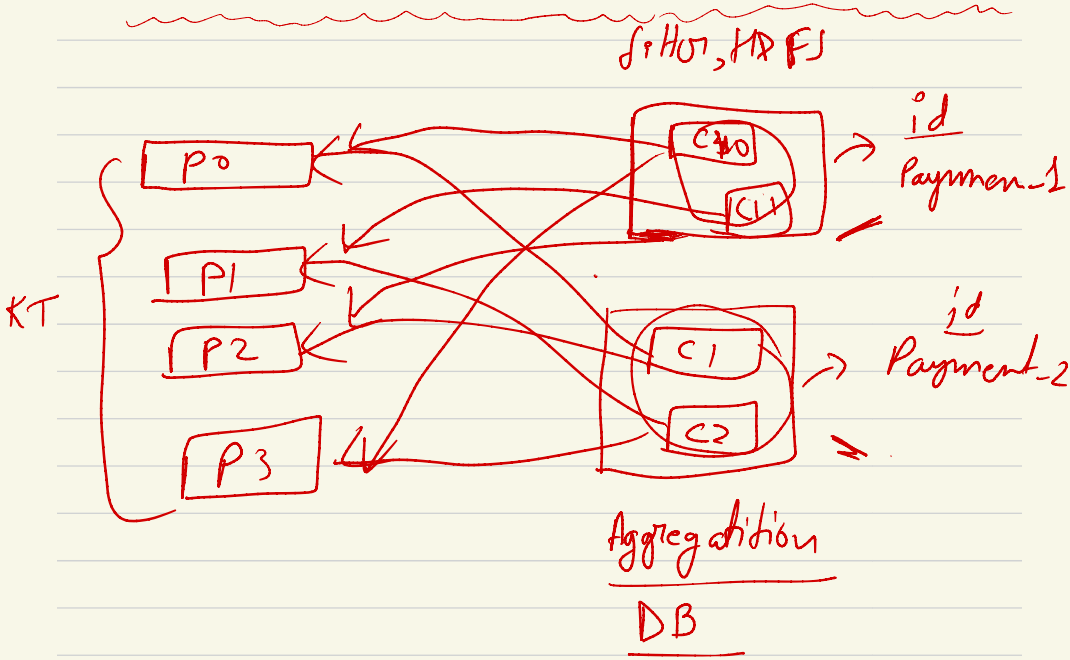
Multiple Consumers (Consumer group)

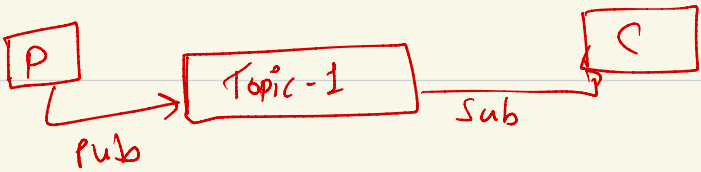
```
C = Consumer()
C.ConsumerGroupID("Payment_Consumer")
```

At a time only 1 Consumer of a Consumer group can read data from a specific

Partition

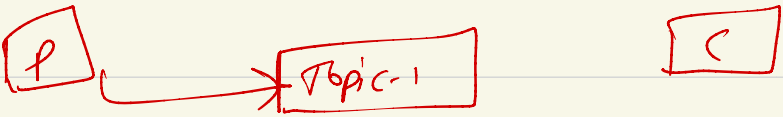

P0
P1
P2
P3
C1
C2
C3
C4
C5
C6
Consumer group


P0
P1
P2
P3
KT
C1
C2
C1
C2
id
Payment_1
id
Payment_2
Aggregation
DB



↳ Sync = wait until the operation is complete

↳ async = Means fire & forget
↳ Don't wait for anythin



step→ Poll the data 10Record
② → transform ×
③ → Dump into destination ×
④ → Commit offset →
⑤ → ↳ Sync Commit
⑥ → ↳ Async Commit

0 Sync Comit
↳ Where we are not concerned latency

- Data Consistency
- Avoid data loss while processing

2. Async Commit
   - where we need latency
   - Inconsistant data
   - data loss as well