Kafka KRaft: Removing ZooKeeper (The Future)

1. History & Motivation (ZooKeeper ko kyun hata rahe hain?)

2020 mein Kafka community ne decide kiya ki wo ZooKeeper par dependency khatam karenge. Is project ko **KIP-500** (Kafka Improvement Proposal) naam diya gaya.

Purane System (with ZooKeeper) ki problems:

- **Scaling Limit:** ZooKeeper ke saath Kafka cluster max **100,000 partitions** hi handle kar pata tha. Isse zyada hone par system slow ho jata tha.
- **Complex Maintenance:** Admins ko do alag systems manage karne padte the (Kafka + ZooKeeper).
- **Double Security:** ZooKeeper ki security alag aur Kafka ki alag manage karni padti thi.

New System (KRaft Mode) ke fayde:

- **Massive Scaling:** Ab Kafka **Millions of Partitions** tak scale kar sakta hai.
- **Single Process:** Sirf Kafka start karna hai, ZooKeeper ki zarurat nahi.
- **Faster Recovery:** Agar system crash hota hai, toh KRaft mode mein recovery (restart) bahut fast hoti hai.
- **Unified Security:** Sirf Kafka ka security model manage karna hai.

---

2. Architecture Change (Before vs After)

Main difference ye hai ki management ka kaam (Controller) ab external ZooKeeper nahi karta, balki Kafka ke andar hi **Special Brokers** karte hain.

A. With ZooKeeper (Old Architecture) Yahan Controller (Boss) select karne ke liye ZooKeeper ki zarurat hoti thi.

```
[ ZooKeeper Quorum ]  (External Manager)
       |
       | (Manages Metadata & Leaders)
       v
[ Kafka Cluster ]
   |-- Broker 1
   |-- Broker 2 (Controller)
   |-- Broker 3
```

B. With KRaft (New Architecture) Yahan Kafka Brokers ke andar hi **Quorum Controller** chalta hai. Kafka khud apna boss hai.

```
[ Kafka Cluster (Self Managed) ]
   |
   |-- Broker 1 (Standard)
   |-- Broker 2 (Controller + Data) <--- Ye "Quorum Leader" hai
   |-- Broker 3 (Standard)
   |
   (No External ZooKeeper Required)
```

---

3. Version Timeline (Kaunsa version use karein?)

Ye transition raato-raat nahi hua. Iska ek proper timeline hai:

- **Kafka 3.0 - 3.2:** KRaft mode introduce hua (Lekin Experimental tha).
- **Kafka 3.3.1 (KIP-833): Production Ready.** Ab aap companies mein KRaft use kar sakte hain bina dar ke.
- **Kafka 4.0 (Future):** ZooKeeper puri tarah **Remove** ho jayega. Sirf KRaft chalega.

Comparison Table:

Feature	ZooKeeper Mode	KRaft Mode

Partitions Limit	~100,000	Millions
Startup Process	2 Process (ZK + Kafka)	1 Process (Kafka only)
Shutdown Time	Slow	Very Fast
Controller	Broker elected by ZK	Quorum Controller (Raft)

---

4. Summary

Ab Kafka **Raft** consensus protocol (KRaft) use karta hai metadata manage karne ke liye. Ye fast hai, simple hai, aur future-proof hai. Agar aap aaj naya cluster bana rahe hain (Kafka 3.3+), toh **KRaft** use karna better hai.

Next Step: Would you like to proceed with the **Practical Hands-on Setup** (Installing Kafka on your machine)? Bataiye aapka OS kaunsa hai (Windows, Mac, or Linux)?

Kafka Theory Roundup: A Quick Revision

Is lecture mein humne ab tak jo bhi **Kafka Theory** seekhi hai, uska ek pura recap (summary) kiya gaya hai. Ye saare concepts aage practicals mein use honge.

1. The Kafka Cluster (Infrastructure)

- **Brokers:** Cluster multiple **Brokers** (Servers) se banta hai. (e.g., 9 Brokers ka cluster).
- **Internal Concepts:**
  - **Topics & Partitions:** Data topics mein store hota hai jo partitions mein divided hote hain.
  - **Replication:** Data ki copies banti hain taaki agar broker fail ho toh data lost na ho.
  - **Leader & ISR:** Har partition ka ek Leader hota hai aur baaki replicas (ISR - In Sync Replicas) hote hain.
  - **__consumer_offsets:** Ye ek internal Kafka topic hai jo consumers ki progress save karta hai.

Text Diagram: The Big Picture

```
[ KAFKA CLUSTER ]
   |
   |-- [ Broker 101 ] -- (Leader for Part 0)
   |-- [ Broker 102 ] -- (Replica for Part 0)
   |-- [ Broker 103 ] -- (Replica for Part 0)
```

---

2. Producers (Sending Data)

Producers ka kaam hai Source System se data lena aur Kafka mein bhejna.

- **Routing Logic (Data kahan jayega?):**
  - **Round Robin:** Agar `Key = null` hai, toh data sabhi partitions mein barabar distribute hoga.
  - **Key-Based Ordering:** Agar `Key` di gayi hai (e.g., "TruckID"), toh same key hamesha same partition mein jayegi.
- **Acks Strategy (Safety Levels):**
  - `acks=0`: No confirmation (Fastest, Risky).
  - `acks=1`: Leader confirmation (Medium safe).
  - `acks=all`: Leader + Replicas confirmation (Safest, Slowest).

---

3. Consumers (Reading Data)

Consumers data ko pull karte hain aur process karte hain.

- **Consumer Groups:** Multiple consumers milkar ek group banate hain taaki load divide ho sake.
- **Offsets:** Consumers batate hain ki unhone kahan tak padh liya hai (Commit Offsets).

- **Delivery Semantics (Reading Modes):**
  - **At least once:** Message duplicate ho sakta hai par loss nahi hoga (Most common).
  - **At most once:** Message loss ho sakta hai par duplicate nahi hoga.
  - **Exactly once:** Message sirf ek baar process hoga (Ideal state).

---

4. Cluster Management (The Boss)

Cluster ko manage kaun karta hai?

- **ZooKeeper (The Old Way):**
  - Brokers ko manage karta hai.
  - Leader Election karta hai.
  - Metadata store karta hai.
- **KRaft Mode (The New Way):**
  - Kafka without ZooKeeper.
  - Kafka khud apne aap ko manage karta hai (using internal Controller).
  - Future mein yahi standard banega.

Text Diagram: Evolution

```
Past/Present:  [ ZooKeeper ] ----> manages ----> [ Kafka Cluster ]
Future:        [ Kafka Cluster (Self Managed / KRaft) ]
```

Conclusion

Humne puri theory cover kar li hai: **Brokers, Topics, Producers, Consumers, aur Zookeeper/KRaft.** Ab hum **Practical Implementation** ki taraf badhenge.

Next Step: Theory complete ho gayi hai! 🎉 Kya aap ab **Kafka Installation (Starting Kafka)** ke steps chahte hain? (Apna OS batayein: Windows, Mac, ya Linux).