Topic Replication: Fault Tolerance & Safety

1. Replication Factor (Kyun chahiye?)

Jab aap apne local computer par Kafka seekh rahe hote hain, toh `Replication Factor = 1` chal jata hai. Lekin **Production (Real Companies)** mein hum risk nahi le sakte.

- **The Rule:** Production mein Replication Factor hamesha **> 1** hona chahiye (Usually **2 ya 3** rakha jata hai, 3 is standard).
- **Goal:** Agar koi Broker (Server) down ho jaye (maintainance ya crash ki wajah se), toh bhi data safe rahe aur system chalta rahe.

---

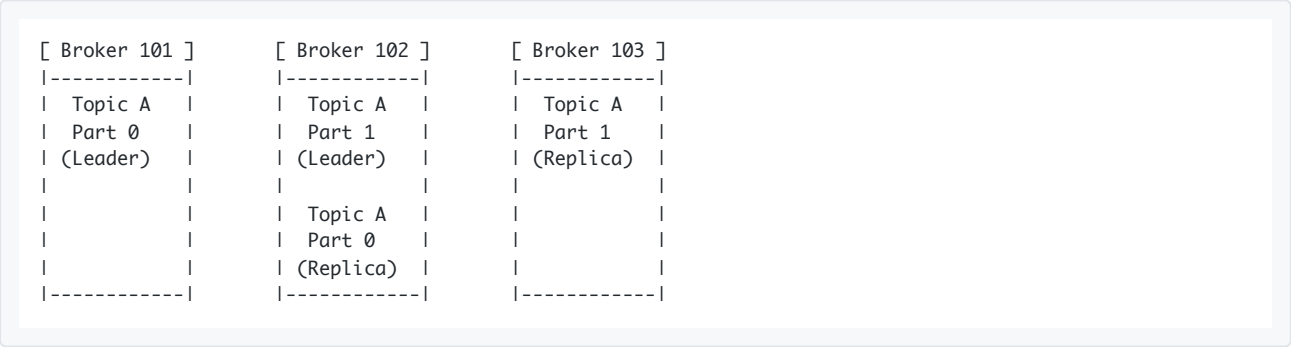
2. Example Scenario (Replication Factor = 2)

Maan lijiye humare paas **Topic A** hai jisme **2 Partitions** hain aur **Replication Factor 2** hai. Humare paas **3 Brokers** hain: 101, 102, 103.

Distribution kaise hogi?

- **Partition 0:** Original Broker 101 par hai -> Iski copy (Replica) Broker 102 par jayegi.
- **Partition 1:** Original Broker 102 par hai -> Iski copy (Replica) Broker 103 par jayegi.

Text Diagram: Cluster State



Scenario: Agar Broker 102 Down ho gaya (Crash)?

- **Partition 1:** Broker 102 ke paas tha, wo gaya. **Lekin** Broker 103 ke paas uski copy (Replica) hai, toh wo take over kar lega.
- **Partition 0:** Broker 101 ke paas hai (Safe). Uski replica 102 par thi jo lost hai, but main data safe hai.
- **Result:** System chalta rahega. User ko pata bhi nahi chalega ki server down hua.

---

3. Leader vs. Replica (Boss kaun hai?)

Kafka mein partitions ki copies (replicas) hoti hain, lekin sab barabar nahi hote.

1. **Leader:**
   - Har partition ka **sirf 1 Leader** hota hai at a time.
   - **Producers hamesha Leader ko hi data bhejte hain.** Wo replicas se baat nahi karte.
   - **Consumers (Default):** Hamesha Leader se hi data read karte hain.
2. **ISR (In-Sync Replica):**
   - Wo replica jo Leader ke saath perfectly sync mein hai (Fast data copy kar raha hai).
   - Agar koi replica slow ho jaye ya disconnect ho jaye, toh wo **ISR list** se bahar ho jata hai (Out of Sync).
   - Jab Leader fail hota hai, toh naya leader sirf **ISR** mein se hi chuna jata hai.

Text Diagram: Data Flow (Standard)

```
           | (Writes only to Leader)
           v
    +--------------+
    |  PARTITION 0 |  (Broker 101)
    |   LEADER     |
    +--------------+
      /        \
     / (Copies) \
    v            v
+--------+    +--------+
| P0 Rep |    | P0 Rep |
| (ISR)  |    | (ISR)  |
+--------+    +--------+

      ^
      | (Reads only from Leader)
      |
    [ CONSUMER ]
```

---

4. Consumer Replica Fetching (Kafka 2.4+ New Feature)

Pehle Kafka ka rule strict tha: *Only talk to Leader.* Lekin **Kafka v2.4** se ek naya feature aaya hai.

- **Closest Replica Fetching:**
  - Agar Consumer aur Leader door hain (alag data center mein), lekin ek Replica consumer ke paas wale data center mein hai.
  - Toh Consumer ab **Replica se bhi read kar sakta hai**.
- **Benefit:**
  1. **Latency:** Data jaldi milega.
  2. **Cost:** Cloud mein data transfer cost bachegi (kyunki same region/data center sasta hota hai).

Note: Ye advanced configuration hai. Default behavior abhi bhi Leader se padhna hi hai.

Summary Checklist

- [x] **Replication Factor:** Production mein 2 ya 3 rakhein taaki server fail hone par data bacha rahe.
- [x] **Leader:** Partition ka wo broker jo saare Writes aur Reads handle karta hai.
- [x] **ISR:** Replica jo leader ke saath update hai.
- [x] **Producer Rule:** Only writes to Leader.
- [x] **Consumer Rule:** Usually reads from Leader (but newer versions support closest replica).

Next Step: Would you like to create a **Producer Acknowledgement (Acks)** strategy diagram? Ye batata hai ki Producer ko kaise confirm hota hai ki data safe hai ya nahi.