

Kafka Consumers & Deserialization

1. Consumers Basics (Data Read Kaise Karte Hain?)

Ab tak humne data produce kiya, ab humein us data ko **read** karna hai. Ye kaam **Consumers** karte hain.

- **Pull Model:** Kafka Consumers **Pull Model** follow karte hain.
 - Iska matlab Kafka Broker (Server) data push nahi karta.
 - Consumer khud Broker se request karta hai aur data "pull" karta hai.
- **Smart Consumers:**
 - Consumer ko pata hota hai ki kaunsa broker kis partition ka leader hai.
 - Agar koi Broker fail ho jaye, toh Consumer automatic recover kar leta hai.

Reading Scenario:

- Ek Consumer sirf ek Partition se padh sakta hai.
- Ya fir ek Consumer multiple partitions (e.g., Partition 1 & 2) se bhi data padh sakta hai.

Text Diagram: Consumers Reading Data

```
Topic A
|-- Partition 0  <----- [ Consumer 1 ] (Reads only Part 0)
|
|-- Partition 1  <--\
|                   \__ [ Consumer 2 ] (Reads Part 1 & 2)
|-- Partition 2  <---/
```

2. Ordering Guarantees (Sequence Kaise maintain hota hai?)

Jab Consumer data read karta hai, toh wo hamesha **Low Offset to High Offset** (Order mein) read karta hai.

- *Sequence:* 0 -> 1 -> 2 -> 3 ...

Important Rule: Ordering sirf **within a partition** guarantee hoti hai.

- Consumer 2 jo Partition 1 aur 2 dono se padh raha hai, wo Partition 1 ka data order mein padhega aur Partition 2 ka data order mein padhega.
- Lekin Partition 1 aur Partition 2 ke beech mein kaunsa message pehle aayega, iski koi guarantee nahi hai (Parallel processing).

3. Deserialization (Bytes to Objects)

Kafka mein data **Bytes (0s and 1s)** ke form mein store hota hai. Lekin Consumer application ko **Objects** (String, Integer, etc.) chahiye hote hain. Yahan **Deserialization** kaam aata hai.

- **Process:** Binary Data (Kafka se) -> **Deserializer** -> Programming Object.
- **Matching Rule:** Jo Serializer Producer ne use kiya tha, wahi same type ka Deserializer Consumer ko use karna padega.
 - Producer: `IntegerSerializer` -> Consumer: `IntegerDeserializer`
 - Producer: `StringSerializer` -> Consumer: `StringDeserializer`

Example: Agar humare paas message hai:

- **Key:** `123` (Integer)
- **Value:** `"hello world"` (String)

Text Diagram: The Deserialization Flow

```
[ KAFKA ]           [ CONSUMER APP ]  
(Bytes)            (Objects)  
  
Key: 01010... -----> [Integer Deserializer] -----> Key: 123  
Value: 11001... -----> [String Deserializer] -----> Value: "hello world"
```

4. Critical Warning: Topic Lifecycle

Deserializer ko advance mein pata hona chahiye ki data ka format kya hai. Iska matlab aap **Topic ka Data Type change nahi kar sakte**.

- **The Problem:** Agar aapne pehle messages **Integers** mein bheje the, aur baad mein Producer ka code change karke **Float** bhejna shuru kar diya, toh **Consumer crash ho jayega** (kyunki wo Integer expect kar raha tha).
- **The Solution:** Agar aapko data format change karna hai, toh existing topic ko modify mat karo. Ek **New Topic** create karo aur naya format wahan use karo.

Summary Checklist

1. Consumers **Pull model** use karte hain (Data request karte hain).
2. Reading order **Low Offset to High Offset** hota hai.
3. **Deserialization** Bytes ko wapas usable Objects mein convert karta hai.
4. Topic ka data type **lifecycle ke dauran change nahi karna chahiye** (warna consumers break ho jayenge).

Next Step: Would you like to learn about **Consumer Groups** and how they coordinate to read data faster?