Kafka Producers & Message Keys

1. Producers Basics (Producer Kya Hai?)

Humne dekha ki Topics mein data store hota hai. Lekin wo data wahan pahunchta kaise hai? Ye kaam **Producers** ka hota hai.

- Producers **Topic Partitions** mein data write karte hain.
- **Important:** Producer ko **advance mein pata hota hai** ki data kis partition mein aur kis Kafka Broker (Server) par write hone wala hai.
- Ye decision Kafka Server nahi leta, balki **Producer khud decide karta hai**.
- Agar koi Broker fail ho jaye, toh Producer automatically recover kar leta hai.

---

2. Message Keys Strategy (Data Kahan Jayega?)

Producer jab message bhejta hai, toh wo message ke saath ek **Key** bhi bhej sakta hai. Ye Key optional hoti hai (String, Number, Binary kuch bhi ho sakti hai). Iske do main scenarios hain:

Case A: Key is NULL (No Key)

- Agar aapne Key provide nahi ki (`Key = null`), toh data **Round Robin** fashion mein distribute hoga.
- **Load Balancing:** Iska matlab data barabar divide hoga partitions ke beech.
  - Message 1 -> Partition 0
  - Message 2 -> Partition 1
  - Message 3 -> Partition 0
  - Message 4 -> Partition 1

Case B: Key is NOT NULL (Key Provided)

- Agar aapne Key provide ki hai (e.g., "Truck_123"), toh Kafka ek **Hashing Strategy** use karta hai.
- **Guarantee:** Jiske paas **Same Key** hogi, wo hamesha **Same Partition** mein jayega.
- **Use Case (Why use Keys?):** Jab aapko **Ordering** chahiye hoti hai kisi specific field ke liye.

Example (Truck GPS Data): Agar aap chahte hain ki ek specific Truck (ID: 123) ka data hamesha line se (order mein) mile, toh aap `Truck ID` ko Key bana denge.

- Truck 123 -> Always Partition 0
- Truck 456 -> Always Partition 1
- Truck 123 (Next msg) -> Always Partition 0

Text Diagram: Key vs No Key

```
Scenario 1: NO KEY (Round Robin)
Producer  --->  Msg 1  --->  Partition 0
Producer  --->  Msg 2  --->  Partition 1
Producer  --->  Msg 3  --->  Partition 2
(Data evenly distributed)

Scenario 2: WITH KEY (Hashing)
Producer (Key="Truck A")  --->  Partition 0
Producer (Key="Truck B")  --->  Partition 1
Producer (Key="Truck A")  --->  Partition 0
(Truck A ka data hamesha Partition 0 mein hi jayega)
```

---

3. Anatomy of a Kafka Message (Message mein kya hota hai?)

Jab Producer ek message create karta hai, toh uske andar ye cheezein hoti hain:

1. **Key:** (Optional, Binary format) - Routing ke liye.
2. **Value:** (Message Content, Binary format) - Main data.

3. **Compression Type:** (Optional) - Message chhota karne ke liye (gzip, snappy, lz4, zstd).
4. **Headers:** (Optional) - Metadata key-value pairs.
5. **Partition + Offset:** Kahan store hua.
6. **Timestamp:** Kab create hua.

[Image of Kafka message structure]

---

4. Kafka Message Serializer

Kafka bahut optimized hai kyunki wo input aur output mein sirf **Bytes (0s and 1s)** accept karta hai. Lekin hum code mein Objects (Strings, Integers) use karte hain. Isliye humein **Serialization** karni padti hai.

- **Serialization:** Apne data/objects ko Bytes mein convert karna.
- Producer ko batana padta hai ki Key aur Value ko kaise convert karein.

Example Process:

1. **Key:** Integer `123` --> **IntegerSerializer** --> `010101` (Bytes)
2. **Value:** String `"Hello"` --> **StringSerializer** --> `110010` (Bytes)
3. Ye Bytes phir Kafka Cluster ko bheje jate hain.

Text Diagram: Serialization Flow

```
[ Code Object ]       [ Serializer ]        [ Kafka ]
 Key: 123       --->  IntSerializer    --->  Binary (Bytes)
 Value: "Hello" --->  StringSerializer --->  Binary (Bytes)
```

---

5. The Partitioner (Advanced - Optional)

- Producer ke andar ek logic hota hai jise **Partitioner** kehte hain.
- Ye `Murmur2` algorithm use karke Key ko Hash karta hai.
- **Formula:** `targetPartition = Math.abs(Utils.murmur2(keyBytes)) % (numPartitions)`
- Isi logic ki wajah se Producer ko pehle se pata hota hai ki message kis partition mein land karega.

Summary: Producers data write karte hain. Agar **Key nahi** hai, toh data random/round-robin (load balanced) hota hai. Agar **Key hai**, toh same key wala data hamesha same partition mein jata hai (ordering ke liye). Kafka sirf bytes samajhta hai, isliye Serializers use hote hain.