Step 2: High-Level Design - Matlab hai ki ab hum system ka "Blueprint" bana rahe hain. Interviewer ko yeh dikhana hai ki hum sirf code nahi likhte, hume architecture samajh aata hai.

Main isse tukdon mein todta hoon aur har decision ka **"Reason"** samjhata hoon.

1. The Basic Concept

Sabse pehle, basic flow samajhte hain.

```
+-------------+         +------------------------+         +-------------+
|   PRODUCER  |  -----> |      MESSAGE QUEUE     |  -----> |   CONSUMER  |
| (Mobile App)|         |      (The Buffer)      |         |  (Backend)  |
+-------------+         +------------------------+         +-------------+
```

Kyu chahiye ye? (Decoupling) Agar Producer seedha Consumer ko data bhejta aur Consumer down hota, toh data loss ho jata. Queue beech mein ek "Buffer" (store) ka kaam karta hai. Producer queue mein daal kar bhool jata hai, Consumer apni speed se utha leta hai.

---

2. Messaging Models (Baat karne ke tareeke)

Yahan candidate ne do models discuss kiye hain. Interviewer yeh difference zaroor puchega.

A. Point-to-Point (Queue Model) - Figure 3

- **Kaise kaam karta hai:** Ek message sirf **EK** consumer padhega.
- **Reason:** Load Balancing (Kaam baatna).

```
Message: [A, B, C]

[Queue] ------------------------> [Consumer 1] (Gets Message A)
   |
   +----------------------------> [Consumer 2] (Gets Message B)
   |
   +----------------------------> [Consumer 3] (Gets Message C)

(Har message alag consumer ke paas gaya. Kaam divide ho gaya.)
```

B. Publish-Subscribe (Pub-Sub Model) - Figure 4

- **Kaise kaam karta hai:** Ek message **MULTIPLE** consumers padh sakte hain.
- **Reason:** Broadcasting. Jab ek hi data alag-alag teams ko chahiye (e.g., Payment Notification -> Email Team + Analytics Team).

```
Message: [A]

           /------------------> [Consumer 1] (Gets Message A)
[Topic] ----+
           \------------------> [Consumer 2] (Gets Message A)

(Ek hi message sabko mila.)
```

Decision: Humara system **Pub-Sub** use karega kyunki yeh flexible hai. Isse hum Point-to-Point bhi simulate kar sakte hain.

---

3. Core Terminology (Dictionary)

Kafka/Queue system samajhne ke liye ye 3 shabd sabse zaroori hain.

A. Topic (Category)

- **Matlab:** Folder ka naam (e.g., `payments`, `clicks`).

B. Partition (Sharding/Scaling)

Yeh sabse important technical concept hai.

- **Problem:** `payments` topic mein 1 TB data hai. Ek server par rakhna mushkil hai.
- **Solution (Partition):** Hum Topic ko chote tukdon mein tod dete hain.

```
Topic: "Payments"
-------------------
|  Partition 0    |  ----> Stored on Broker 1 (Server A)
-------------------
|  Partition 1    |  ----> Stored on Broker 2 (Server B)
-------------------
|  Partition 2    |  ----> Stored on Broker 3 (Server C)
-------------------
```

- **Reason: Scalability.** Ab hum parallel mein 3 servers se data likh aur padh sakte hain.

---

4. Consumer Group (The Magic Trick)

Yeh concept thoda confusing ho sakta hai, dhyan se samjho. Consumer Group hi woh feature hai jo hume **Point-to-Point** aur **Pub-Sub** dono ki power deta hai.

Rule: Ek **Partition** ka data, ek **Group** ke andar sirf **Ek Consumer** padh sakta hai.

Diagram: Load Balancing using Consumer Groups (Maan lo hamare paas 2 Partitions hain aur 2 Consumers hain ek hi group mein).

```
Topic: "Orders"
+-------------+
| Partition 0 | -------------------------> [Consumer 1]  \
+-------------+                                           } Consumer Group A
+-------------+                                           /
| Partition 1 | -------------------------> [Consumer 2]  /
+-------------+
```

- **Result:** P0 ka data C1 lega, P1 ka data C2 lega. Kaam bat gaya (Parallel Processing).

Crucial Constraints (Interview Trap): Agar Partition 2 hain, aur Group mein **3 Consumers** hain, toh kya hoga?

```
+-------------+
| Partition 0 | ---------> [Consumer 1]
+-------------+
+-------------+
| Partition 1 | ---------> [Consumer 2]
+-------------+

                           [Consumer 3]  <---- IDLE (Khali baithega!)
```

- **Reason:** Ordering maintain karne ke liye hum 1 partition se 2 logon ko read karne nahi de sakte.

---

5. The Full Architecture

End mein, pura system judta hai.

```
                                +-----------------+
                                |   Zookeeper     |
                                | (Coordinator)   |
                                +-------+---------+
                                        |
                                        v
+----------+          +----------------------------------+          +-----------+
| PRODUCER |  ----->  |         BROKER CLUSTER           |  ----->  | CONSUMERS |
+----------+          |                                  |          +-----------+
                      |  [Broker 1]  [Broker 2] [Broker 3] |
                      |     |            |           |     |
                      |  [Part-0]     [Part-1]    [Part-2] |
                      +----------------------------------+
                                        ^
                                        |
                             +----------+------------+
                             |   Metadata Storage    |
                             | (Config & Retention)  |
                             +-----------------------+
```

1. **Broker:** Data store karta hai (Actual hard disk yahan hai).
2. **Coordination Service (Zookeeper/Etcd):**
   - **Role:** Class Monitor.
   - **Reason:** Distributed system mein servers marte rehte hain. Zookeeper track rakhta hai ki kaun zinda hai aur naya Leader kaun banega.
3. **Metadata Storage:**
   - **Role:** Settings Register.
   - **Reason:** Topic ka naam kya hai? Partition kahan hai? Yeh info yahan hoti hai.
4. **State Storage (Offsets):**
   - **Role:** Bookmark.
   - **Reason:** Consumer ko yaad rakhna padta hai ki "maine page 10 tak padh liya". Yeh "Page 10" (Offset) save hota hai taaki crash hone ke baad wahi se continue ho.

Summary for Beginner

Is pure design ka maqsad hai **Speed** aur **Reliability**.

- **Speed:** Partitions ki wajah se hum parallel processing kar rahe hain.
- **Reliability:** Zookeeper aur multiple Brokers ki wajah se agar ek server jalta bhi hai, system chalta rahega.