Kafka

Message Queue হলো একটি চিঠির বক্স বা বাফার, যা দুটি সফটওয়্যার বা সার্ভিসের মধ্যে সরাসরি কথা বলার বদলে মাঝখানে মেসেজ জমা রাখে। আর **Apache Kafka** হলো পৃথিবীর সবচেয়ে শক্তিশালী এবং দ্রুতগামী মেসেজিং সিস্টেম।

নিচে বিষয়টি বিস্তারিত এবং সহজভাবে বুঝিয়ে বলা হলো:

১. Message Queue কী?

সাধারণত একটি সার্ভিস যখন অন্যটিকে ডাটা পাঠায়, তখন দুটিকেই অনলাইনে থাকতে হয়। কিন্তু Message Queue থাকলে:

- **Producer:** মেসেজ পাঠিয়ে দিয়ে নিজের কাজে চলে যায়।
- **Queue:** মেসেজটি জমা রাখে।
- **Consumer:** নিজের সুবিধা অনুযায়ী সময়মতো মেসেজটি তুলে নেয়।

এটি ব্যবহার করা হয় যাতে একটি সার্ভিস স্লো হলে অন্যটি আটকে না থাকে (**Decoupling**)।

২. Apache Kafka কী?

Kafka শুধু একটি সাধারণ Message Queue নয়, একে বলা হয় **Distributed Event Streaming Platform**। এটি সেকেন্ডে কোটি কোটি মেসেজ হ্যান্ডেল করতে পারে। LinkedIn, Uber, এবং Netflix-এর মতো বড় বড় কোম্পানি তাদের বিশাল ডাটা প্রসেস করার জন্য Kafka ব্যবহার করে।

৩. Kafka-এর মূল কনসেপ্টসমূহ (Architecture)

Kafka বুঝতে হলে এই ৪টি শব্দ জানা জরুরি:

ক. Producer (প্রডিউসার)

যারা মেসেজ বা ডাটা তৈরি করে পাঠায়। যেমন: আপনার অ্যাপে কেউ ক্লিক করলে সেই তথ্য পাঠানো।

খ. Consumer (কনজিউমার)

যারা মেসেজগুলো পড়ে এবং সে অনুযায়ী কাজ করে। যেমন: ক্লিক করার তথ্য নিয়ে সেটি ডাটাবেসে সেভ করা।

গ. Topic (টপিক)

এটি অনেকটা ফোল্ডারের মতো। সব মেসেজ ঢালাওভাবে এক জায়গায় থাকে না। আলাদা আলাদা ক্যাটাগরিতে থাকে। যেমন: order-logs, user-signups ইত্যাদি।

ঘ. Broker (ব্রোকার)

Kafka যে সার্ভারে চলে, তাকেই ব্রোকার বলে। সাধারণত অনেকগুলো ব্রোকার মিলে একটি **Cluster** তৈরি করে, যাতে একটি সার্ভার নষ্ট হলেও ডাটা হারিয়ে না যায়।

৪. কেন Kafka সাধারণ কিউ (যেমন RabbitMQ) থেকে আলাদা?

- **Persistence (স্থায়িত্ব):** সাধারণ কিউতে মেসেজ পড়ার পর সেটি ডিলিট হয়ে যায়। কিন্তু Kafka-তে আপনি চাইলে মেসেজগুলো অনেক দিন (এমনকি চিরকাল) সেভ করে রাখতে পারেন।
- **Huge Scalability:** Kafka অনেকগুলো সার্ভারে ছড়িয়ে কাজ করতে পারে, তাই এটি অকল্পনীয় পরিমাণ ডাটা লোড নিতে পারে।
- **Replay:** আপনি চাইলে পুরনো মেসেজগুলো আবার প্রথম থেকে প্লে করতে পারেন (যেমন: কোনো বাগ বা ভুল হলে সব ডাটা আবার প্রসেস করা)।

৫. বাস্তব উদাহরণ: একটি ই-কমার্স ওয়েবসাইট

ধরুন একজন ইউজার একটি অর্ডার করলেন। তখন Kafka নিচের কাজগুলো করতে সাহায্য করবে:

1. **Producer:** অর্ডার হওয়ার সাথে সাথে orders টপিকে একটি মেসেজ পাঠাবে।
2. **Inventory Service (Consumer 1):** মেসেজটি নিয়ে স্টক থেকে পণ্য কমিয়ে দেবে।
3. **Email Service (Consumer 2):** মেসেজটি নিয়ে ইউজারকে একটি কনফার্মেশন মেইল পাঠাবে।
4. **Shipping Service (Consumer 3):** কুরিয়ার কোম্পানিকে তথ্য পাঠাবে।

এই সব সার্ভিস কিন্তু একে অপরের ওপর নির্ভর করছে না, তারা সবাই শুধু Kafka থেকে তথ্য নিচ্ছে।

সংক্ষেপে সারাংশ:

- **Message Queue:** মিডলম্যান যা মেসেজ জমা রাখে।
- **Kafka:** অত্যন্ত দ্রুত এবং বড় ডাটার জন্য ব্যবহৃত ডিস্ট্রিবিউটেড মেসেজিং প্ল্যাটফর্ম।

🧠 **Message Queue কী?**

👉 **Message Queue = asynchronous communication system**

Instead of:

Service A → Service B (sync)

We do:

Service A → Queue → Service B

📌 Producer আর Consumer **decoupled**

কেন দরকার?

- Traffic spike handle
- Services loosely coupled
- Reliability
- Async processing

🚀 Kafka কী?

👉 **Apache Kafka = distributed, high-throughput message streaming platform**

Kafka শুধু queue না —
👉 **commit log + stream processor**

Used by:

- Netflix
- Uber
- LinkedIn

🧱 Kafka Core Architecture

Producer → Kafka Broker → Consumer

Kafka cluster = multiple brokers

1. Topics

👉 **Topic = logical stream of messages**

Example:

order_created

user_signup

payment_done

📌 Topic ≠ Queue (conceptually stream)

2. Partitions (VERY IMPORTANT)

👉 Each topic is split into **partitions**

Topic: order_created

Partition-0 | Partition-1 | Partition-2

✓ Parallelism
✓ Scalability

📌 Ordering guaranteed **per partition**

3. Producers (Write Path)

Producer:

- Sends message to topic
- Partition decided by:
    - Key (hash)
    - Round-robin

Example:

orderId → same partition

📌 Interview line:

“Kafka guarantees ordering within a partition.”

4. Brokers

👉 Kafka servers

Responsibilities:

- Store messages on disk
- Serve producers & consumers
- Replicate partitions

5. Consumers & Consumer Groups

👉 Consumers read messages
👉 Consumer group = load balancing unit

Partition-0 → Consumer-1

Partition-1 → Consumer-2

📌 One partition → one consumer per group

6. Offset (🔥 Core Concept)

👉 Offset = message position in partition

Partition-0:

[0][1][2][3][4]

Consumer tracks offset.

✓ Replay possible
✓ Exactly-once semantics possible

7. Replication & Fault Tolerance

Each partition:

- Leader
- Followers

Writes go to leader → replicated

📌 Interview line:

“Kafka uses leader-follower replication for partitions.”

8. Message Delivery Semantics

Mode	Meaning
At most once	May lose
At least once	May duplicate
Exactly once	No loss, no duplicate

Kafka supports **Exactly Once** (with config)

9. Kafka vs Traditional Queue (RabbitMQ)

Aspect	Kafka	RabbitMQ
Storage	Disk	Memory
Throughput	Very high	Medium
Replay	Yes	No
Use case	Event streaming	Task queue

🍔 Food Delivery Example

Order Service → Kafka(order_created)

↓

Inventory Service

Delivery Service

Notification Service

✓ Async
✓ Scalable
✓ Fault tolerant

⚠️ When NOT to Use Kafka?

- Simple small apps
- Low throughput
- Strict request-response

🏆 FAANG Interview Killer Answer 💎

Say this:

“Kafka is a distributed event streaming platform that stores messages as an immutable log across partitions, enabling high-throughput, fault-tolerant, and scalable asynchronous communication.”

🧠 One-Line Memory Hook

Kafka = distributed commit log with consumers