RabbitMQ

RabbitMQ হলো একটি জনপ্রিয় ওপেন-সোর্স **Message Broker**।[1] এটি অনেকটা একটি "স্মার্ট ডাকঘর"-এর মতো কাজ করে। এটি বিভিন্ন অ্যাপ্লিকেশনের মধ্যে মেসেজ আদান-প্রদান করে এবং নিশ্চিত করে যে মেসেজটি সঠিক জায়গায় পৌঁছেছে।

নিচে RabbitMQ সম্পর্কে বিস্তারিত এবং সহজ আলোচনা করা হলো:

১. RabbitMQ কীভাবে কাজ করে? (The Post Office Analogy)

মনে করুন, আপনি একটি চিঠি পাঠাতে চান। আপনি চিঠিটি লিখে পোস্টবক্সে ফেলে দিলেন। ডাকপিয়ন সেই চিঠিটি নিয়ে সঠিক ঠিকানায় পৌঁছে দিল।

RabbitMQ-তে:

- **Producer:** যে অ্যাপ্লিকেশনটি মেসেজ পাঠায় (চিঠি লেখক)।
- **Queue:** যেখানে মেসেজগুলো জমা থাকে (ডাকঘর বা লেটারবক্স)।
- **Consumer:** যে অ্যাপ্লিকেশনটি মেসেজটি গ্রহণ করে এবং প্রসেস করে (চিঠি প্রাপক)।

২. RabbitMQ-এর মূল কনসেপ্টসমূহ

ক. Exchange (এক্সচেঞ্জ)

এটি RabbitMQ-এর একটি বিশেষ অংশ। প্রোডিউসার সরাসরি কিউতে (Queue) মেসেজ পাঠায় না, বরং এক্সচেঞ্জে পাঠায়। এক্সচেঞ্জ ঠিক করে দেয় মেসেজটি কোন কিউতে যাবে। এটি অনেকটা সর্টিং অফিসের মতো।

খ. Binding (বাইন্ডিং)

এক্সচেঞ্জ এবং কিউ-এর মধ্যে যে কানেকশন বা লিংক থাকে, তাকেই বাইন্ডিং বলে।

গ. Routing Key (রাউটিং কী)

এটি একটি নির্দিষ্ট কোড বা নাম যা এক্সচেঞ্জ ব্যবহার করে মেসেজটিকে সঠিক কিউতে পাঠানোর জন্য।

৩. কেন RabbitMQ ব্যবহার করবেন?

- **Asynchronous Processing:** ধরুন আপনার অ্যাপে কেউ রেজিস্ট্রেশন করল। এখন তাকে ইমেইল পাঠাতে ৫ সেকেন্ড সময় লাগে। আপনি ইউজারকে ৫ সেকেন্ড বসিয়ে না রেখে, ইমেইল পাঠানোর কাজটি RabbitMQ-তে পাঠিয়ে দিলেন। ইউজার সাথে সাথে "Success" মেসেজ পেল, আর ব্যাকগ্রাউন্ডে RabbitMQ ইমেইলটি পাঠিয়ে দিল।
- **Decoupling:** দুটি সার্ভিসকে একে অপরের ওপর সরাসরি নির্ভর করতে হয় না। একটি সার্ভিস ডাউন থাকলেও অন্যটি কাজ চালিয়ে যেতে পারে।
- **Load Balancing:** আপনার যদি অনেক বেশি কাজ জমে যায়, তবে আপনি একাধিক **Consumer** চালু করতে পারেন। RabbitMQ কাজগুলোকে সবার মধ্যে ভাগ করে দেবে।

৪. RabbitMQ বনাম Kafka (প্রধান পার্থক্য)

বৈশিষ্ট্য	RabbitMQ	Kafka
কাজের ধরন	ট্র্যাডিশনাল মেসেজ ব্রোকার।	ডিস্ট্রিবিউটেড স্ট্রিমিং প্ল্যাটফর্ম।
মেসেজ ডিলিট	কনজিউমার পড়ার সাথে সাথে মেসেজ সাধারণত ডিলিট হয়ে যায়।	মেসেজ নির্দিষ্ট সময় পর্যন্ত সেভ থাকে (Replay করা যায়)।
গতি	ছোট এবং মিডিয়াম ডাটার জন্য অত্যন্ত দ্রুত।	বিশাল পরিমাণ (Big Data) ডাটার জন্য সেরা।

বৈশিষ্ট্য	RabbitMQ	Kafka
ব্যবহার	ইমেইল পাঠানো বা টাস্ক কিউ-এর জন্য সেরা।	রিয়েল-টাইম ডাটা অ্যানালিটিক্স বা লগ ট্র্যাকিংয়ের জন্য।

৫. এক্সচেঞ্জ টাইপ (Exchange Types)

RabbitMQ-তে মেসেজ রাউটিং করার ৪টি উপায় আছে:

1. **Direct:** রাউটিং কী মিলে গেলে সরাসরি ওই কিউতে পাঠায়।
2. **Fanout:** যতগুলো কিউ কানেক্টেড আছে, সবগুলোতে মেসেজ পাঠিয়ে দেয় (Broadcasting)।
3. **Topic:** প্যাটার্ন মিলিয়ে মেসেজ পাঠায় (যেমন: *.order.*)।
4. **Headers:** মেসেজ হেডারের তথ্যের ওপর ভিত্তি করে পাঠায়।

সংক্ষেপে সারাংশ:

RabbitMQ হলো একটি নির্ভরযোগ্য টুল যা আপনার অ্যাপ্লিকেশনের বিভিন্ন অংশকে একে অপরের সাথে নিরাপদে এবং দ্রুত কথা বলতে সাহায্য করে। এটি কাজগুলোকে লাইনে দাঁড় করিয়ে দেয় যাতে কোনো সার্ভিস ক্র্যাশ না করে।

🧠 RabbitMQ কী?

👉 **RabbitMQ = traditional message broker (AMQP based)**
Main goal:

Reliable task delivery

Kafka যেখানে **event streaming,**
RabbitMQ সেখানে **task/message queue**

🧱 Core Architecture

Producer → Exchange → Queue → Consumer

📌 RabbitMQ-তে **Exchange = brain**

1. Producer

👉 Message publish করে
👉 কোন queue জানে না
👉 শুধু exchange জানে

2. Exchange (🔥 Most Important)

Exchange decides:

কোন message কোন queue-তে যাবে

Types of Exchanges

🔹 Direct Exchange

routing_key = queue_key

✓ One-to-one mapping
✓ Simple task queue

🔹 Fanout Exchange

Message → ALL queues

✓ Broadcast
✓ Notification, email

🔹 **Topic Exchange (🔥 Popular)**

order.* → order.created

✓ Pattern-based routing
✓ Microservices eventing

🔹 **Headers Exchange**

- Based on headers
- Rarely used

3. Queue

👉 Actual message storage
👉 FIFO
👉 Consumers read from here

Features:

- Durable
- Auto-delete
- TTL

4. Consumer

👉 Pulls message
👉 Acknowledges (ACK)

ACK → message deleted

NO ACK → requeue

📌 Reliability controlled by ACK

5. Message Acknowledgement (VERY IMPORTANT)

Type	**Meaning**
Auto-ACK	Fast but risky
Manual ACK	Safe

Interview line:

“RabbitMQ ensures at-least-once delivery via acknowledgements.”

6. Prefetch (Backpressure Control)

👉 Limits how many messages consumer can take

prefetch = 5

✓ Prevents slow consumer overload

7. Persistence & Durability

For crash safety:

- Durable queue
- Persistent messages

📌 Both needed for true durability

8. RabbitMQ vs Kafka (🔥 Interview Gold)

Feature	RabbitMQ	Kafka
Model	Queue	Log
Storage	Memory-first	Disk-first
Replay	❌	✅
Ordering	Queue level	Partition level
Throughput	Medium	Very high
Use case	Task processing	Event streaming

🍔 Food Delivery Example

Order API

↓

RabbitMQ (order_queue)

↓

Payment Worker

↓

Delivery Worker

✓ Simple
✓ Reliable
✓ Fast processing

⚠️ When NOT to Use RabbitMQ?

- Event replay needed
- Analytics pipelines
- Very high throughput

🏆 FAANG Interview One-Liner

Say this:

“RabbitMQ is a message broker that uses exchanges and queues to reliably deliver messages with acknowledgements, ideal for task-based asynchronous processing.”

🧠 Memory Trick

RabbitMQ = Smart Router (Exchange) + Reliable Queue