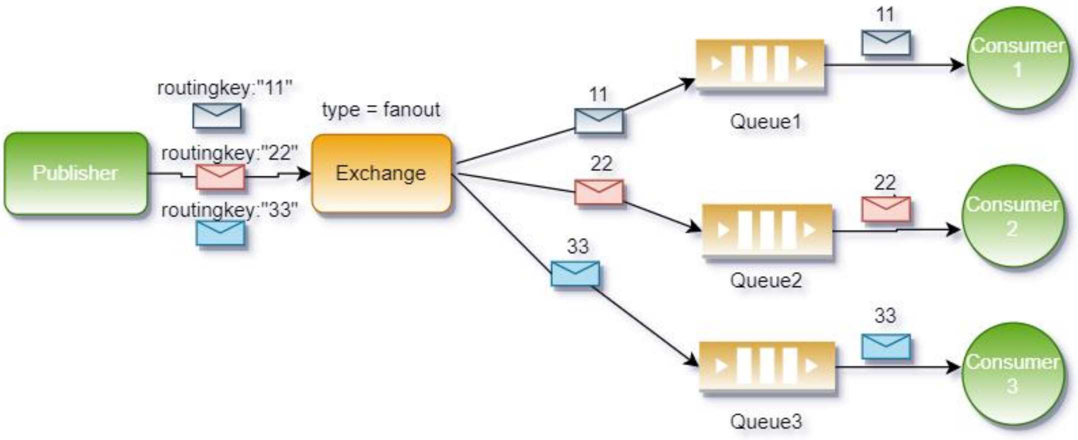
RabbitMQ open source message Broker software.

Why do we need a message A queue?

1. It helps application asynchronous communication করার জন্য message queue ব্যবহার করা হয়।
2. Background job process করার জন্য message queue ব্যবহার করা হয়।
3. Enable load balancing:
   - ○ এক queue তে multiple consumer থাকে।
   - ○ Consumer message গ্রহণ করে এবং Round Robin করে কাজ করে।

Message Flow:

1. Producer message send করবে।
2. Exchange message receive করে binding rules এর মাধ্যমে ঠিক করে কোন queue-তে send করবে।
3. ঐ queue থেকে consumer message receive করবে।
4. Consumer Ack করলে queue থেকে remove করে দিবে।



RabbitMQ Exchange:

RabbitMQ-তে **Exchange** হলো একটি মেকানিজম যা প্রাপ্ত মেসেজগুলোকে Queue-তে রাউট করার দায়িত্ব পালন করে। RabbitMQ-তে চার ধরনের Exchange আছে, এবং প্রতিটির আলাদা কাজ এবং ব্যবহারের পদ্ধতি রয়েছে। নিচে তাদের প্রধান পার্থক্য তুলে ধরা হলো:

---

1. Direct Exchange

- **বর্ণনা**: মেসেজ **routing key** এর উপর ভিত্তি করে সরাসরি নির্দিষ্ট Queue-তে পাঠানো হয়।
- **ব্যবহার**: যদি নির্দিষ্টভাবে একটি Queue-তে মেসেজ পাঠাতে চান।
- **উদাহরণ**:
    - Routing Key: email.queue
    - যদি Queue email.queue এর সাথে binding থাকে, তবে মেসেজ সরাসরি সেই Queue-তে যাবে।

2. Fanout Exchange

- **বর্ণনা**: মেসেজ কোনো **routing key** বিবেচনা না করে, Exchange এর সাথে সংযুক্ত সমস্ত Queue-তে পাঠানো হয়।
- **ব্যবহার**: ব্রডকাস্টের জন্য। যখন একই মেসেজ অনেকগুলো Queue-তে পাঠাতে হয়।
- **উদাহরণ**:
  - একবার মেসেজ প্রকাশ করলে, এটি Queue-1, Queue-2, এবং Queue-3 এ একসাথে চলে যাবে।

---

3. Topic Exchange

- **বর্ণনা**: **Wildcard routing key** ব্যবহার করে মেসেজ রাউট করা হয়। এটি নির্দিষ্ট বা আংশিক মিলিয়ে Queue-তে মেসেজ পাঠায়।
- **ব্যবহার**: ডাইনামিক রাউটিং এর জন্য, যেখানে **routing key** আংশিক মিলে মেসেজ পাঠাতে হয়।
- **উদাহরণ**:
  - Routing Key: log.error
  - Binding: log.* -> এই ক্ষেত্রে log.error, log.info ইত্যাদি মেসেজ মিলে যাবে।
  - Binding: *.error -> এই ক্ষেত্রে system.error, log.error মিলে যাবে।

---

4. Headers Exchange

- **বর্ণনা**: মেসেজের **header properties** এর ভিত্তিতে Queue-তে রাউট করা হয়, **routing key** এর প্রয়োজন নেই।
- **ব্যবহার**: যখন মেসেজের কাস্টম অ্যাট্রিবিউটের উপর ভিত্তি করে রাউটিং দরকার হয়।
- **উদাহরণ**:
  - Header: type = pdf
  - শুধুমাত্র সেই Queue-তে মেসেজ যাবে, যার header type=pdf।

সারাংশ:

- **Direct**: নির্দিষ্ট Queue-তে মেসেজ পাঠানোর জন্য।
- **Fanout**: সমস্ত Queue-তে মেসেজ ব্রডকাস্ট করার জন্য।
- **Topic**: আংশিক মিলে Queue-তে মেসেজ পাঠানোর জন্য।
- **Headers**: মেসেজের হেডারের ভিত্তিতে মেসেজ পাঠানোর জন্য।

RabbitMQ এর কাজের ধরণ বুঝে সঠিক Exchange নির্বাচন করলে মেসেজ রাউটিং আরো কার্যকর হয়।

Note:

1. যদি আমারা producer এর exchange type হিসাবে fanout ব্যবহার করি তাহলে ওই producer এর exchange এর সাথে connect প্রতিটা subscriber message পাবে কিন্তু subscriber সবার routingKey আলাদা হতে হবে।

   সেইসব subscriber message পাবে না যাদের routingKey duplicate কারণ একটা একটা queue একবার message পেলে ack দিয়ে দিলে আর message পাওয়া যায় না।

2. যদি আমারা producer এর exchange type হিসাবে topic ব্যবহার করি তাহলে ওই producer এর exchange এর সাথে connect ওইসব subscriber message পাবে যাদের routingKey producer এর routingKey এর wildcard সাথে মিলে।

Topic Exchange এর route key:

RabbitMQ Topic Exchange: * vs

*** (Single-Level Wildcard):**

- একটি মাত্র লেভেলের জন্য wildcard।
- * . এর পরে বা মধ্যে একটা key check করে।

- উদাহরণ:
  device.*.status → ম্যাচ করবে:

  - device.mobile.status
  - device.laptop.status
- **ম্যাচ করবে না**:

  - device.mobile.battery.status
  - device.status

 # (Multi-Level Wildcard):

- একাধিক লেভেল (০ বা বেশি) কভার করে।
- # . এর মধ্যে অনেক গুলো key check করে।
- উদাহরণ:
  device.# → ম্যাচ করবে:
  - device.mobile.status
  - device.laptop.battery.status
  - device.status
  - device

সংক্ষেপে:

- *: একটি লেভেল।
- #: ০ বা একাধিক লেভেল।

RabbitMQ RPC:

RabbitMQ RPC (Remote Procedure Call) **under the hood** সাধারণত **direct exchange** ব্যবহার করে। RabbitMQ RPC প্যাটার্নে একটি producer একটি request পাঠায় এবং একটি consumer সেই request প্রসেস করে একটি response পাঠিয়ে দেয়। এর পিছনে কাজের ধাপগুলো নিচে ব্যাখ্যা করা হলো:

RabbitMQ RPC এর কর্মপদ্ধতি:

1. **Request Producer**:
   - একটি direct exchange ব্যবহার করে একটি queue তে message পাঠানো হয়।
   - Producer তার জন্য একটি **unique reply queue** তৈরি করে (এই queue কে callback queue বলা হয়)।
   - reply-to property এর মাধ্যমে reply queue এর নাম উল্লেখ করা হয়।
   - Producer এই queue থেকে response এর জন্য অপেক্ষা করে।
2. **Consumer**:
   - Consumer সেই queue এর message consume করে।
   - Request process করার পর সেই response **producer এর callback queue তে পাঠিয়ে দেয়**।
3. **Response**:
   - Producer তার callback queue থেকে response consume করে এবং কাজ শেষ করে।

Exchange এর ভূমিকা:

- **Direct Exchange** ব্যবহার করে message গুলো নির্দিষ্ট queue তে পাঠানো হয়, কারণ direct exchange message এর **routing key** এর উপর ভিত্তি করে নির্দিষ্ট queue তে bind হয়ে যায়।
- RPC প্যাটার্নে একাধিক RPC call করা হলে প্রতিটি response আলাদা reply queue তে ফেরত পাঠানো হয়।

উদাহরণ:

Step-by-Step Flow:

1. Producer একটি RPC call করে এবং একটি unique reply queue তৈরি করে।
2. Producer মূল queue তে message পাঠায় direct exchange ব্যবহার করে।
3. Consumer সেই queue তে message consume করে এবং process করার পর reply queue তে response পাঠিয়ে দেয়।
4. Producer reply queue থেকে response consume করে এবং RPC call এর ফলাফল পায়।

Summary: RabbitMQ এর RPC প্যাটার্নের জন্য **direct exchange** ব্যবহার করা হয় কারণ এটি message গুলো নির্দিষ্ট routing key এর মাধ্যমে নির্দিষ্ট queue তে পাঠাতে সাহায্য করে।