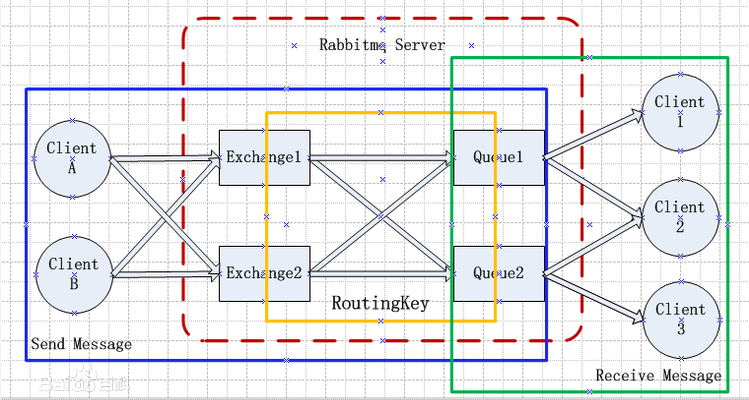
**初步了解RabbitMQ：**

RabbitMQ是一个轻量级的消息代理中间件，它支持多种消息通信协议，支持分布式部署，同事也支持运行于多个操作系统，它的灵活、高可用特性是它成为流行MQ中间件的重要因素。

RabbitMQ支持多种协议，这其中最重要的当属AMQP（Advanced Message Queuing Protocal 高级消息队列协议），何为AMQP，可以自行百度，反正它实现了客户端与消息代理中间件之间不受语言、产品等约束。

这里用一张图让大家容易的认识RabbitMQ的工作原理：



它在点到点、发布订阅的基础上还加入了中间这乱七八糟的一坨。这一坨是高级消息队列的关键，可以自由地规划路线，选择发送到哪一个队列上。具体怎么路由，客户端是不必关心的，客户端只负责如何将消息投进去，另一边负责消息来了如何处理。

消息生产者（Producer）：顾名思义产生消息的一端，也就是发送数据的一端，比如用户注册成功后，服务器要发送一封激活邮件，这个时候，为了提升性能，点击注册后肯定立马回给用户说你信息提交成功了，那么现在快去邮箱找找激活邮件，用户几乎感觉不到时间差了，可是将邮件存到数据库并推送到邮件服务器可是会受很多因素控制的，比如网速缓慢，那可能就要很长时间了，所以这一步完全可以做成异步。我们可以再用户点击注册提交表单信息之后，就产生一条消息，这个消息的实体就是邮件内容了。所以这里消息生产者是产生邮件的那个角色。（寄信人）

交换器（Exchanger）：负责接收来自生产者投递的消息，并把它转交给消息队列。（快递）

队列（Queue）:负责接收并储存来自交换器的消息。（收件人家的信箱）

消息消费者（Consumer）：会查看是否有消息过来，一旦消息过来了，它要负责处理这条消息，比如刚才那封邮件到达消息队列了，它要负责取出这封邮件，并将这封邮件处理掉，就算这条消息被消费了。（收信人）

作为生产者，所做的事情非常简单，消息投出去就完事儿了，怎么投，很容易，指定Exchange（交换器）就可以了，我们就可以把数据（也就是消息）扔进去。

作为交换器，就要麻烦一点了，它不仅要从生产者那里接过接力棒，要选择丢给合适的队列，怎么投呢，这就是RabbitMQ的强大之处了，有下面四种玩法：

1）direct: 介绍这种玩法之前，先了解这里面的一个名词：绑定。（它还有个好听的英文名：Binding, 不知道是先有的汉子还是先有的单词，反正发音就这么莫名和谐）。Binding就是交换器和队列之间的连接线（类似于媒人牵的红线），每一个线路就是一个Binding，并且都有一个唯一的标识符：binding key。当消息发送过来的时候，消息本身会带上一个routing key，交换器会根据这两个key来考虑如何分配路线。好了，回过头来，那么direct怎么玩呢，很简单粗暴，就是判断routing key是否等于binding key，完全匹配就丢给binding key对应的那个队列。

2）topic: 根据生产者传过来的routing key是否匹配一定的表达式来决定丢给那个队列，这里面可以模糊匹配，正则匹配等。

3）fanout: 广播推送，直接忽略routing key，将消息丢给所有队列（比1还要简单粗暴）

4）headers: 这个和2有点类似，是根据消息的头信息来决定将消息分给那些队列。

作为队列，要做的就是接受Exchange过来的消息，并存在队列里面，按先进先出的顺序。

作为消费者，就是监视消息队列是否有新消息过来，有的话就读取并处理掉。整个流程就结束了。

**客户端和RabbitMQ打交道的API的几个重要对象：**

ConnectionFactory，Connection，Channel

ConnectionFactory: 是产生Connection的工厂

Connection：是RabbitMQ的socket连接，它封装了socket协议相关的部分逻辑。Connection就是客户端和RabbitMQ的一个TCP连接，生产者和消费者都是通过connection连接到RabbitMQ server的，有点类似SQL server的连接。

Channel：是客户端与RabbitMQ打交道的最重要的接口，我们大部分的业务操作是通过这个完成的，包括定义Queue, Exchange, Binding, 发布消息等。

是不是读到这里有点疑惑，Connection为啥不能代替Channel干这些事情？

Channel是建立在TCP连接的基础上的，也就是必须先满足Connection 对象ready，然后才能通过Channel做一些操作的。如果直接在TCP上进行操作（即Connection），建立和关闭TCP连接都是需要代价的，我们需要做的操作非常多，频繁的TCP连接会影响系统性能，并且服务器的TCP的连接是有限制的，这就限制了系统处理高并发的能力，所以，RabbitMQ选择在TCP连接的基础上建立Channel（虚拟连接），这样不仅次数不受限制，建立和撤销的代价也非常小。

下节我们将了解下Spring Cloud Stream和RabbitMQ是怎样集成的。我们可以通过Spring Cloud Stream很方便的使用RabbitMQ。