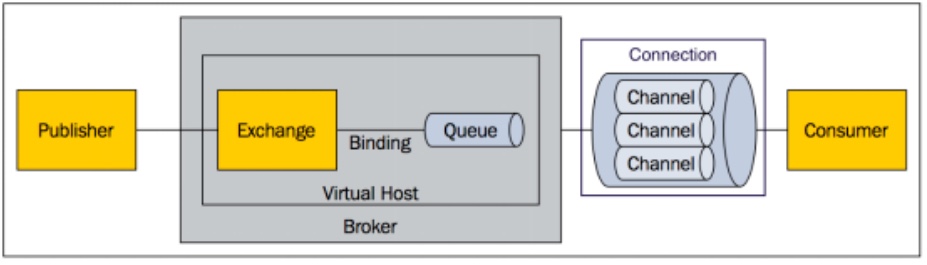
# 一．RabbitMQ基本概念

看一下RabbitMQ是怎么具体来实现消息队列的：

[[](https://s3.mogucdn.com/mlcdn/c45406/180527_7bdj5djf0ae798la210a9ila83874_927x265.jpg)](https://s3.mogucdn.com/mlcdn/c45406/180527_7bdj5djf0ae798la210a9ila83874_927x265.jpg)

[内部结构图](https://s3.mogucdn.com/mlcdn/c45406/180527_7bdj5djf0ae798la210a9ila83874_927x265.jpg)

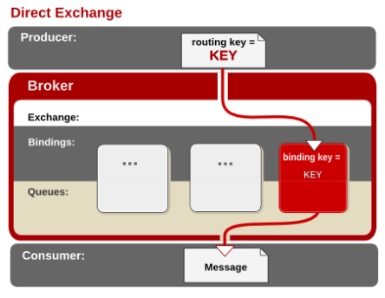
* Message：消息，包含消息头（即附属的配置信息）和消息体（即消息的实体内容）
* Publisher：生产者，向交换机发布消息的主体
* **Exchange：交换机，用来接收生产者发送的消息并将这些消息路由给服务器中的队列**
* Binding：绑定，用于给Exchange和Queue建立关系，就是我们熟知的配对的红娘
* Queue：消息队列，用来保存消息直到发送给消费者。它是消息的容器，也是消息的终点。一个消息可投入一个或多个队列。消息一直在队列里面，等待消费者连接到这个队列将其取走。
* Connection：连接
* Channel：通道，MQ与外部打交道都是通过Channel来的，发布消息、订阅队列还是接收消息，这些动作都是通过Channel完成；简单来说就是消息通过Channel塞进队列或者流出队列
* Consumer：消费者，从消息队列中获取消息的主体
* Virtual Host: 虚拟主机，表示一批交换器、消息队列和相关对象。虚拟主机是共享相同的身份认证和加密环境的独立服务器域。每个 vhost 本质上就是一个 mini 版的 RabbitMQ 服务器，拥有自己的队列、交换器、绑定和权限机制。**vhost 是 AMQP 概念的基础，必须在连接时指定，RabbitMQ 默认的 vhost 是 /**
* Broker：消息队列服务器实体

上面是一些专业的概念，那么可以怎么映射到前面的快递上呢？

# 二．Exchange类型

生产者，将消息投递给Exchange，然后由Exchange将消息路由到对应的Queue上，供消费者消费，那么这个路由有哪些方式呢？

1. Direct策略

[[](https://s3.mogucdn.com/mlcdn/c45406/180527_1fe0g1g39eafjdj4e0k1d8hj65bif_385x298.jpg)](https://s3.mogucdn.com/mlcdn/c45406/180527_1fe0g1g39eafjdj4e0k1d8hj65bif_385x298.jpg)

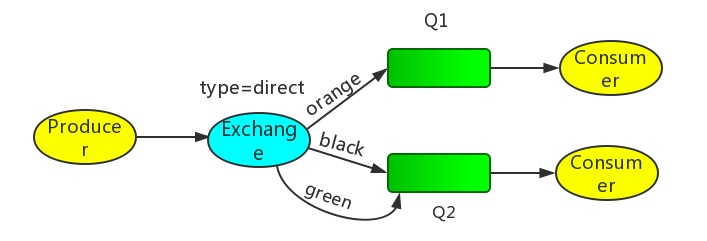
[IMAGE](https://s3.mogucdn.com/mlcdn/c45406/180527_1fe0g1g39eafjdj4e0k1d8hj65bif_385x298.jpg)

消息中的路由键（routing key）如果和 Binding 中的 binding key 一致， 交换器就将消息发到对应的队列中

简单来讲，就是路由键与队列名完全匹配

* 如果一个队列绑定到交换机要求路由键为“dog”
* 只转发 routing key 标记为“dog”的消息，
* 不会转发“dog.puppy”，也不会转发“dog.guard”等等
* 它是完全匹配、单播的模式

举例说明

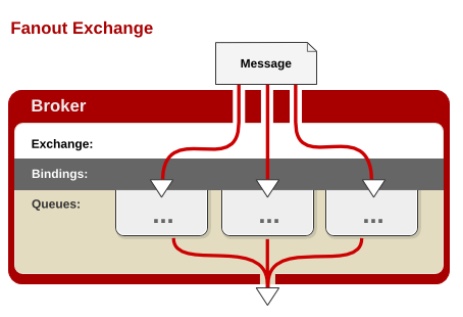
[[](https://s11.mogucdn.com/mlcdn/c45406/180527_7h1b1g7d1ajle2ablh7gk71dd3lca_707x241.jpg)](https://s11.mogucdn.com/mlcdn/c45406/180527_7h1b1g7d1ajle2ablh7gk71dd3lca_707x241.jpg)

[IMAGE](https://s11.mogucdn.com/mlcdn/c45406/180527_7h1b1g7d1ajle2ablh7gk71dd3lca_707x241.jpg)

Exchange和两个队列绑定在一起：

* Q1的bindingkey是orange
* Q2的binding key是black和green.
* 当Producer publish key是orange时, exchange会把它放到Q1上, 如果是black或green就会到Q2上, 其余的Message被丢弃

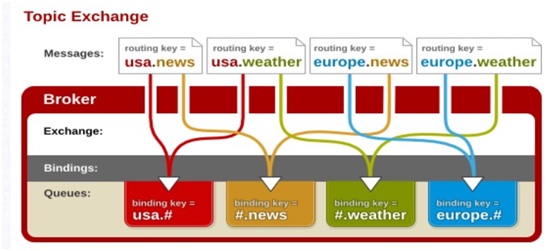
2. Fanout策略

[[](https://s3.mogucdn.com/mlcdn/c45406/180527_0j71hl8a1fhbjlb745hfi2d8acc6g_463x317.jpg)](https://s3.mogucdn.com/mlcdn/c45406/180527_0j71hl8a1fhbjlb745hfi2d8acc6g_463x317.jpg)

[IMAGE](https://s3.mogucdn.com/mlcdn/c45406/180527_0j71hl8a1fhbjlb745hfi2d8acc6g_463x317.jpg)

从上图也可以看出，这种策略，将忽略所谓的routing key,将消息分发到所有绑定的Queue上，更加类似我们理解的广播模式

3. Topic策略

[[](https://s11.mogucdn.com/mlcdn/c45406/180527_1af6hb4k3ja5df983cc98db3i44j6_558x251.jpg)](https://s11.mogucdn.com/mlcdn/c45406/180527_1af6hb4k3ja5df983cc98db3i44j6_558x251.jpg)

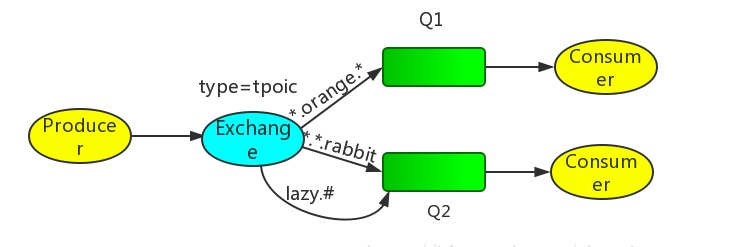
[IMAGE](https://s11.mogucdn.com/mlcdn/c45406/180527_1af6hb4k3ja5df983cc98db3i44j6_558x251.jpg)

topic 交换器通过模式匹配分配消息的路由键属性，将路由键和某个模式进行匹配，此时队列需要绑定到一个模式上

可以理解为直接策略的进阶版，直接策略是完全精确匹配，而topic则支持正则匹配，满足某类指定规则的（如以xxx开头的路由键），可以键消息分发过去

* # 匹配0个或多个单词
* \* 匹配不多不少一个单词

**一个更直观的实例如下**

[[](https://s11.mogucdn.com/mlcdn/c45406/180527_6ff26k8dh11gaeb570i055h07ba3d_731x247.jpg)](https://s11.mogucdn.com/mlcdn/c45406/180527_6ff26k8dh11gaeb570i055h07ba3d_731x247.jpg)

[IMAGE](https://s11.mogucdn.com/mlcdn/c45406/180527_6ff26k8dh11gaeb570i055h07ba3d_731x247.jpg)

Producer发送消息时需要设置routing\_key,

* Q1 的binding key 是”.orange.“
* Q2 是 “*.*.rabbit” 和 “lazy.#”：
* 产生一个 test.orange.mm 消息，则会路由到Q1；而如果是 test.orange则无法路由到Q1,因为Q1的规则是三个单词，中间一个为orange，不满足这个规则的都无效
* 产生一个 test.qq.rabbit 或者 lazy.qq 都可以分发到Q2；即路由key为三个单词，最后一个为rabbit或者不限制单词个数，主要第一个是lazy的消息，都可以分发过来
* 如果产生的是一个 test.orange.rabbit消息，则Q1和Q2都可以满足

4. Headers策略

这个实际上用得不多，它是根据Message的一些头部信息来分发过滤Message，忽略routing key的属性，如果Header信息和message消息的头信息相匹配

5. 小结

主要使用的消息分发策略有三个，直接，路由和扇形，简单的小结下应用场景和区别

a. Direct Exchange

直接完全匹配模式，适用于精准的消息分发

b. Topic Exchange

Routing Key的匹配模式，支持Routing Key的模糊匹配方式，更适用于多类消息的聚合

c. Fanout Exchange

忽略Routing Key, 将消息分配给所有的Queue，广播模式，适用于消息的复用场景