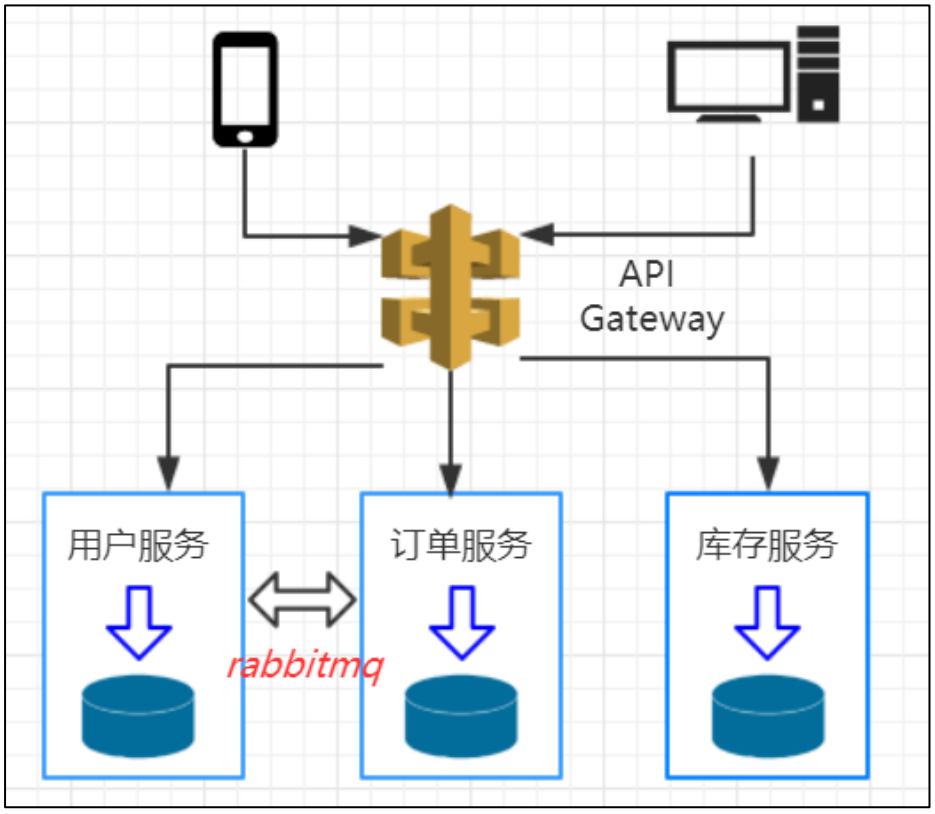
消息队列的微服务场景

当下网购日益普遍，购物节、秒杀大促越来越多，传统的电商系统如果不向微服务转型的话，无法应对大规模订单处理等情况。因此，研发团队将平台解耦成一个个独立的微服务，服务应用之间通过消息传输数据，并以消息队列（Message Queue）作为消息的载体进行通信。

消息发送后由消息系统来确保消息的最终投递，消息发送者和接收者不知道对方的存在，只与消息队列建立联系。



目前流行的消息队列产品很多，大家熟悉的有 ActiveMQ、Kafka、ZeroMQ、RabbitMQ，本章主要讲述结业实践中用到的 RabbitMQ，有关它的原理和工作流程。

RabbitMQ 基本概念

RabbitMQ 是由 Erlang 语言开发完成的，是 AMQP(Advanved Message Queue)的开源实现，遵循 Mozilla Public License 开源协议。

其基本概念如下：

Broker：简单来说就是消息队列服务器实体。

producer：生产者，就是投递消息的程序。

consumer：消费者，就是接受消息的程序。

Exchange：消息交换机，它指定消息按什么规则，路由到哪个队列。

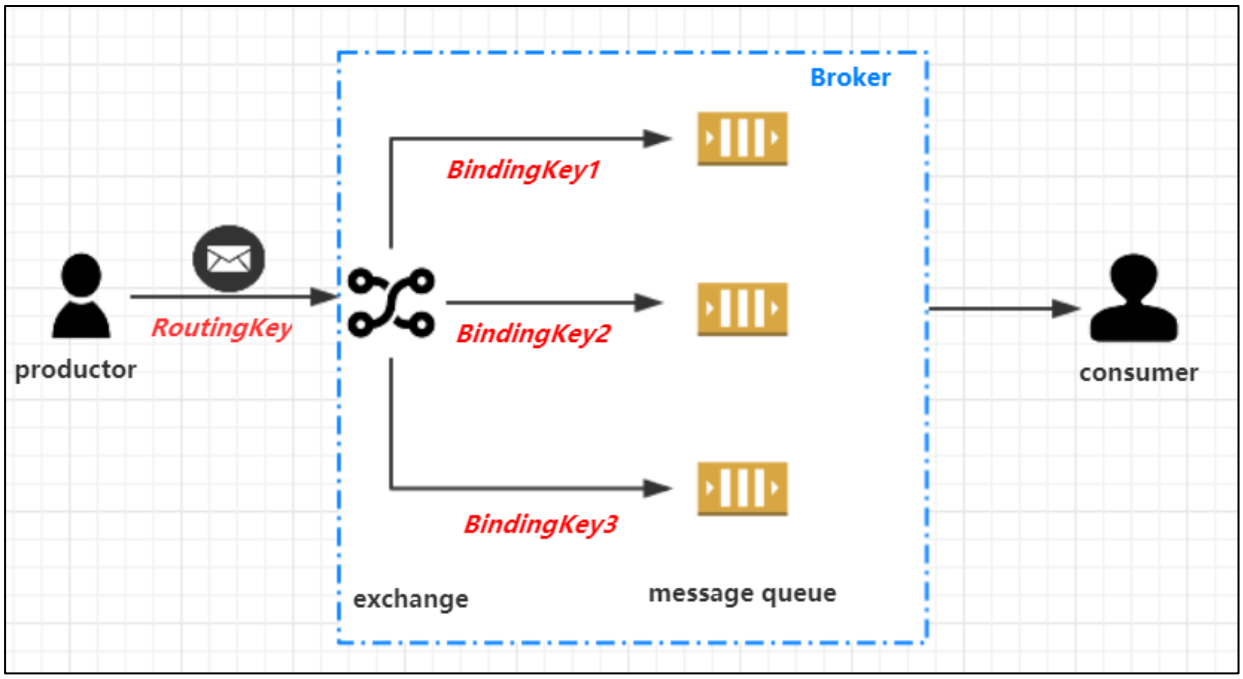
Routing Key：路由关键字，由生产者封装在消息头中，Exchange 根据这个关键字进行消息投递。

Message Queue：消息队列载体，每个消息都会被投入到一个或多个队列。

Binding：绑定，它的作用就是把 Exchange 和 queue 按照路由规则绑定起来。

Binding Key：Message Queue 对接收消息的限制条件，由消费者在 Binding 时指定。

消息的分发流程



1. Productor 发送消息；

2. Exchange 接收 Message，解析消息头得到 Routing Key;

3. Exchange 由 Exchange Type 决定 RoutingKey 和 BindingKey 匹配方式，或是否忽略 RoutingKey，广播发送；

4. Exchange 决策完成后将消息发送给满足条件的队列;

5. 监听该队列的 Consumer 读取消息

Exchange 类型

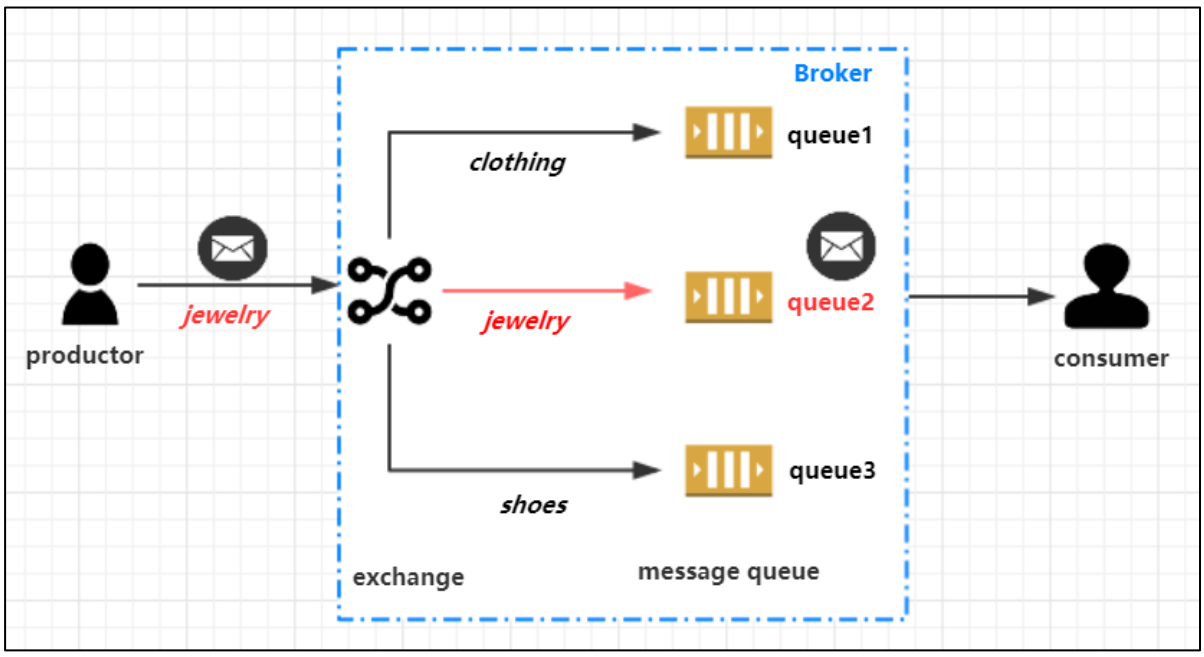
上文的消息分发流程中提到了 Exchange Type，它是 Exchange 在路由消息时的分发策略，目前 RabbitQueue 常见的类型有三种：Direct Exchange，Fanout Exchange，Topic Exchange，实际项目中会根据业务特点进行选型。

1. Direct Exchange

简单的直接匹配，通过 RoutingKey 和 BindingKey 匹配，匹配不上所有 BindingKey 的消息

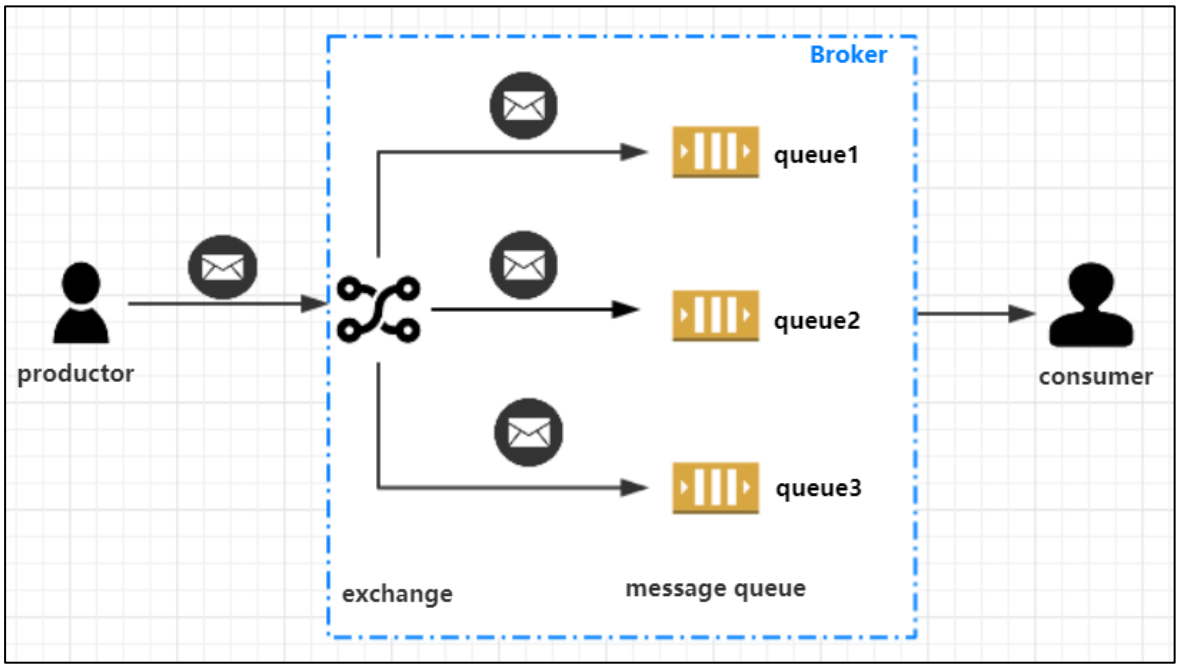
被丢弃。

例：Productor 发送的消息 RoutingKey 为 jewelry，exchange 与 queue2 的 BindingKey 也是 jewelry，两者匹配成功，消息被发送到 message queue。



2. Fanout Exchange

不使用 RoutingKey 匹配，当向所有消费者广播消息时，只要绑定到 Exchange 的队列都可以接收消息。



3. Topic Exchange

在 Direct Exchange 基础上增加了模糊匹配，BindingKey 可以使用*和#通配符，而 RoutingKey 中的多个单词用“.”隔开。通配符“*”代表匹配一个单词，“#”代表匹配零个或多个单词。

例：消息的 RoutingKey 中包含了三个关键词 discount.clothing.wholesale，queue1 对所有的打折产品感兴趣，queue2 对批发产品感兴趣，RoutingKey 满足 queue1、queue2 约束条件，因此都能接收到消息。