消息队列中间件是分布式系统中重要的组件，主要解决应用解耦，异步消息，流量削锋等问题，实现高性能，高可用，可伸缩和最终一致性架构。目前使用较多的消息队列有ActiveMQ，RabbitMQ，ZeroMQ，Kafka，MetaMQ，RocketMQ

异步处理

应用解耦

流量削峰

日志处理

信息通讯

ActiveMQ

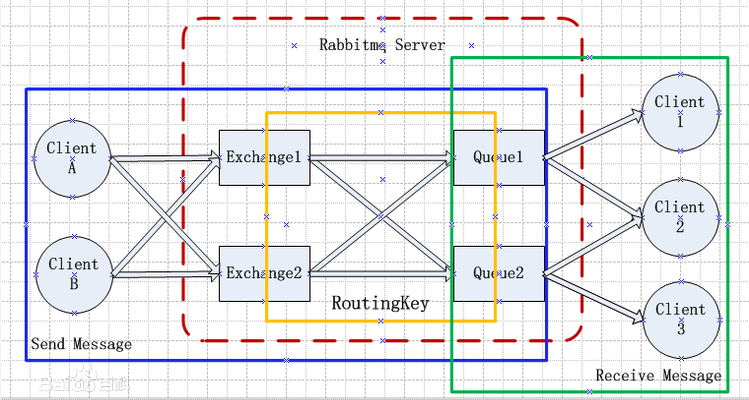
ActiveMQ是Apache下的开源项目，完全支持JMS1.1和J2EE1.4规范的JMS Provider实现。

启动命令：activemq start

Jetty

数据库持久化

RabbitMQ



Message

消息，消息是不具名的，它由消息头和消息体组成。消息是不透明的，而消息头则是由一系列的可选属性组成，这些属性包括rounting-key（路由键）、prority(相对于其他消息的优先权)、delivery-mode(指出该消息可能需要持久性存储)等。

Publisher

消息的生产者，也是一个向交换器发布消息得客户端应用程序。

Exchange

交换器，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，用于消息队列和交换器之间的关联。一个绑定就是基于路由键将交换器和消息队列连接起来的路由规则，所以可以将交换器理解成一个绑定构成的路由表。

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  
网络连接，比如一个TCP连接。

 Channel  
信道，多路复用连接中的一条独立的双向数据流通道。信道是建立在真实的TCP连接内地虚拟连接，AMQP 命令都是通过信道发出去的，不管是发布消息、订阅队列还是接收消息，这些动作都是通过信道完成。因为对于操作系统来说建立和销毁 TCP 都是非常昂贵的开销，所以引入了信道的概念，以复用一条 TCP 连接。

 Consumer  
消息的消费者，表示一个从消息队列中取得消息的客户端应用程序。

 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  
表示消息队列服务器实体。

Exchange分发消息时根据类型的不同分发策略有区别，目前共四种类型：direct、fanout、topic、headers。headers 匹配 AMQP 消息的 header 而不是路由键，此外 headers 交换器和 direct 交换器完全一致，但性能差很多，目前几乎用不到了。

Direct

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。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”，则只转发 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。

Fanout

每个发到 fanout 类型交换器的消息都会分到所有绑定的队列上去。fanout 交换器不处理路由键，只是简单的将队列绑定到交换器上，每个发送到交换器的消息都会被转发到与该交换器绑定的所有队列上。很像子网广播，每台子网内的主机都获得了一份复制的消息。fanout 类型转发消息是最快的。

topic

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。它将路由键和绑定键的字符串切分成单词，这些单词之间用点隔开。它同样也会识别两个通配符：符号“#”和符号“”。#匹配0个或多个单词，“”匹配不多不少一个单词。

激活rabbitmq\_management

"C:\Program Files\RabbitMQ Server\rabbitmq\_server-3.7.3\sbin\rabbitmq-plugins.bat" enable rabbitmq\_management

启动服务

net start RabbitMQ

关闭服务

net stop RabbitMQ

工作队列的两种分发方式，轮询分发（Round-robin）和 公平分发（Fair dispatch）

RabbitMQ派发消息默认采用的是轮询机制，轮询，顾名思义就是挨个的派发，就是第一个派发给C1，第二个派发给C2，第三个派发给C1，第四个派发给C4。正常情况下，这样很好，但是如果遇到某个消费者在消费某个消息时花费时间很长或者因为自身原因或者网络原因阻塞，那么按照这种轮询的策略就显得不合适了。

公平分发就是“能者多劳”. 当每个消费都设置了每次只会从队列取一条数据时，并且关闭自动应答，在每次处理完数据后手动给队列发送确认收到数据。这样队列就会公平给每个消息费者发送数据，消费一条再发第二条，而且可以在管理界面中看到数据是一条条随着消费者消费完从而减少的，并不是一下子全部分发完了。显然公平分发更符合系统设计。