Rabbit MQ

## 历史

rabbitMQ是一个由erLang语言开发的一个AMQP标准的开源实现,旨在给异步消息处理提供商业支持。

## 应用场景及优点

### 1.应用场景

1. 异步处理：例如短信通知、终端状态推送、App推送、用户注册等
2. 数据同步：业务数据推送同步
3. 重试补偿：记账失败重试
4. 系统解耦：通讯上下行、终端异常监控、分布式事件中心
5. 流量消峰：秒杀场景下的下单处理
6. 发布订阅：HSF的服务状态变化通知、分布式事件中心
7. 高并发缓冲：日志服务、监控上报

### **2.优点**

1. 降低数据发送者和接受者的耦合。
2. 对优先级高的请求可以优先处理。
3. 有效的均衡接受者的负载。

….

## 三. 安装及常用命令

### 1. 安装步骤(windows环境)：

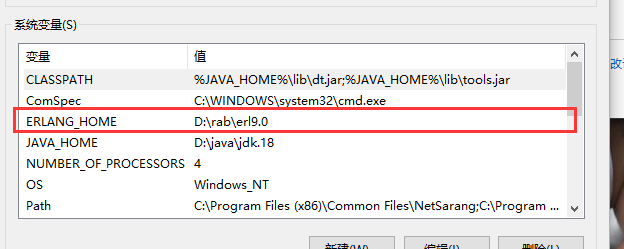
#### **1.软件安装**

1. 首先要安装erLang语言环境，直接取官网下载.exe文件傻瓜式安装即可( <http://www.erlang.org/downloads>)
2. 然后安装rabbitmq,直接去官网下载.exe文件安装即可

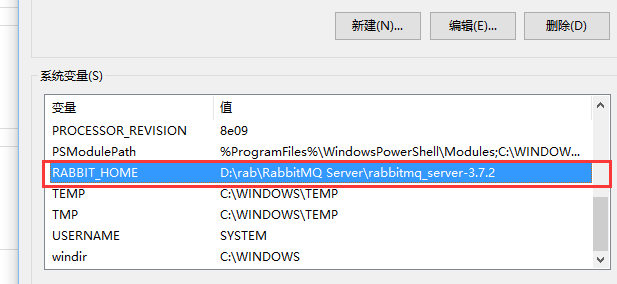
(<http://www.rabbitmq.com/download.html> )

#### **2.配置环境变量**

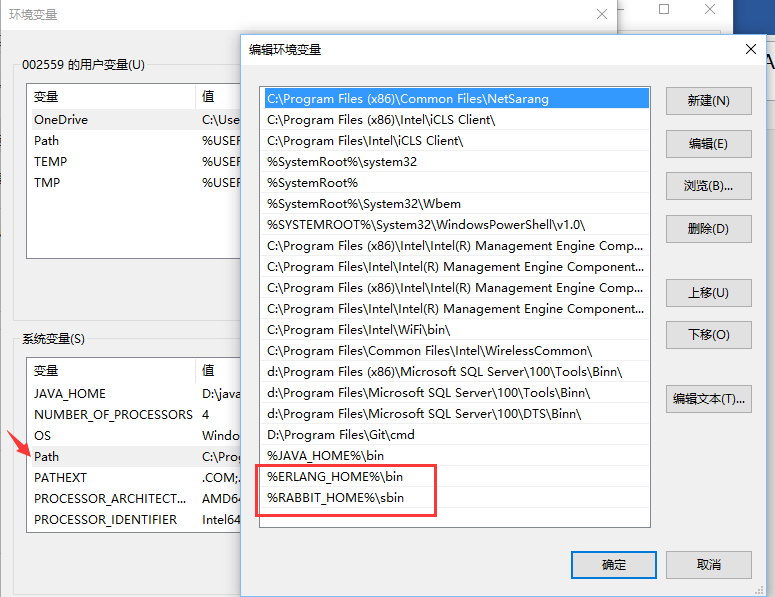
a．首先添加erLang安装路径变量



b．然后添加rabbitmq安装路径变量



1. 最后在path路径中追加命令路径



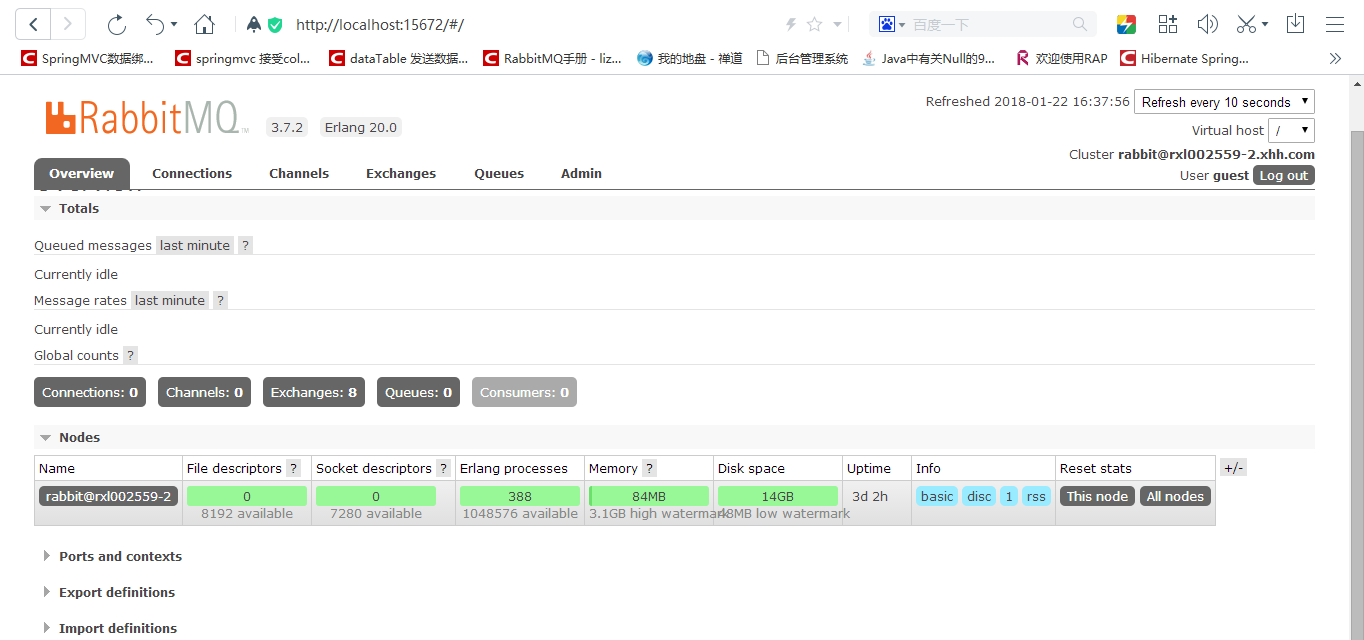
到此环境就准备好了。

### 2. 常用命令

1. 启动:rabbitmq-service start
2. 关闭:rabbitmq-service stop
3. 启动监控管理器:rabbitmq-plugins enable rabbitmq\_management
4. 关闭监控管理器:rabbitmq-plugins disable rabbitmq\_management
5. 查看所有队列:rabbitmqctl list\_queues
6. 清除所有队列:rabbitmqctl reset
7. 关闭应用:rabbitmqctl stop\_app
8. 开启应用:rabbitmqctl start\_app

*--用户及设置权限命令:*

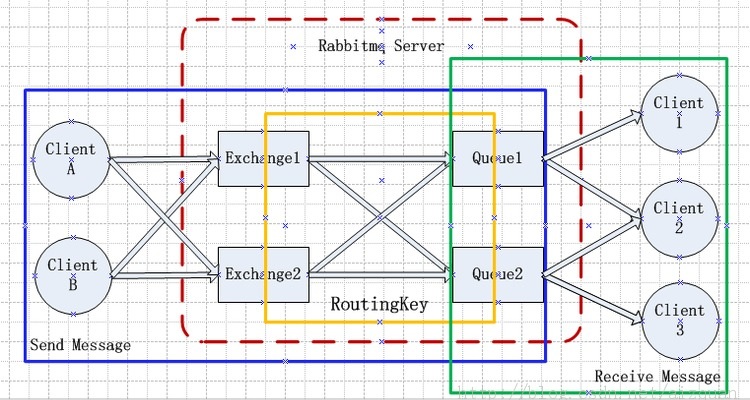
1. 查看所有用户:rabbitmqctl list\_users
2. 新增一个用户:rabbitmqctl add\_user Username Password
3. 删除一个用户:rabbitmqctl delete\_user Username
4. 修改用户密码:rabbitmqctl change\_passowrd Username Newpassword
5. 设置用户角色:rabbitmqctl set\_user\_tags Username Tag
6. 设置用户权限:rabbitmqctl set\_permissions -p VHostPath Username Confgp WriteP readP
7. 查看所有用户的权限:Rabbitmqctl list\_permissions [-p VHostPath]
8. 查看指定用户权限:rabbitmqctl list\_user\_permissions Username
9. 清楚用户的权限信息:rabbitmqctl clear\_permissions [-p VHostPath] Username

**在cmd命令窗口中。可以先启动(命令：rabbitmq-service start) 服务,然后开启监控管理器(命令：rabbitmq-plugins enable rabbitmq\_management),成功后浏览器访问** **<http://localhost:15672> 地址，直观感受下安装成功后的样子.如图：（默认用户: User:guest Passowrd:guest）**9

## 四. 系统应用架构

在深入学习之前必须要清楚其中几个重要的概念和它的运行原理。

图：



### **概念**

**RabbitMQ Server（**Broker**）**: 它是一个维护生产者到消费者路线，保证数据能够按照指定的方式进行传输的一种传输服务。但是这个也不是100%保证,但对于普通应用已经足够了。对于商业系统来说,再做一层数据一致性的校验,就可以彻底保证系统的一致性。

**ClientA & B (Producer)**:这个是所谓的生产者(Producer),也就是数据的发送方,负责发送数据(Message)。通常一个Message包含两个部分:一是数据部分,二描述了数据信息的标识部分,它是Exchanage的名字或者一个标签,RabbitMQ也是通过这个标识决定将Message发送给哪个消费者。

**Exchanage1 & 2**:顾名思义这个东西叫交换器。用于接受消息，转发消息到指定的队列。交换机有四种类型:**direct、topic、headers、fanou**t

1. **direct** : 如果routingKey匹配,那么Message就会被传递到相应的队列中。其实queue在创建时,它会自动的以queue的名字作为routingKey来绑定到那个Exchanage中。（指定routingKey的模式）
2. **topic ：** 这个类型的交换器,会模式匹配分析消息的routingKey属性.

**Queue:** 消息的载体,每个消息都会被投到一个或多个队列。

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

**Routing Key:** 路由关键字,exchange根据这个关键字进行消息投递

**vhost:** 虚拟主机,一个broker里可以有多个vhost，用作不同用户的权限分离。

### **考虑到学习深度，下面详细说明这些概念**

## 五. 消息通信

### 1. 生产者和消费者

如果把RabbitMQ比作是快递服务，那么我们在网上买东西时，卖家就是生产者，而我们则是消费者。在寄送快递时，卖家需要将商品和收件地址给到快递员。由快递员根据收件地址传送到我们手中。

不同于快递服务的是，rabbitMQ无需知道收件人地址，它只需要将包裹(消息),投放到指定的地点，然后有专门的人(消费者)会到这个地址来取走包裹。

当我们的应用程序连接到rabbitMQ的时候，我们必须要做一个决定：我是消费者还是生产者，即是我是要发送消息还是要接受消息?。

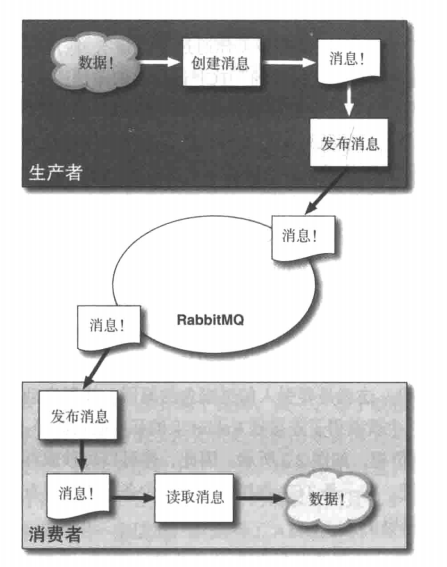
#### **生产者**

生产者即创建消息，然后将消息发送到代理服务器(rabbitmq server)

什么是消息：

消息包含两部分：有效载荷(payload)和标签(label)。**有效载荷**就 是你想要传输的数据,它可以是任何数据,如JSON、XML、MPEG4等等。

**标签**它描述了有效载荷，并且rabbitMQ用它来决定谁将获得消息的拷贝。AMQP只会用标签表述这条消息(一个交换机的名称和一个可选的主题标记),然后把消息发送给rabbitMQ。rabbitMQ会根据标签把消息发送给感兴趣的接收方。



#### **消费者**

消费者很简单。他们连接到代理服务器上，并订阅到队列上。每当有消息到达指定的队列时,RabbitMQ会发送给其中一个消费者。消费者接受到的消息只是有效载荷,标签不会传给消费者。如果消费者想要知道生产者是谁,那么就需要生产者在有效载荷中体现出来。

#### **信道**

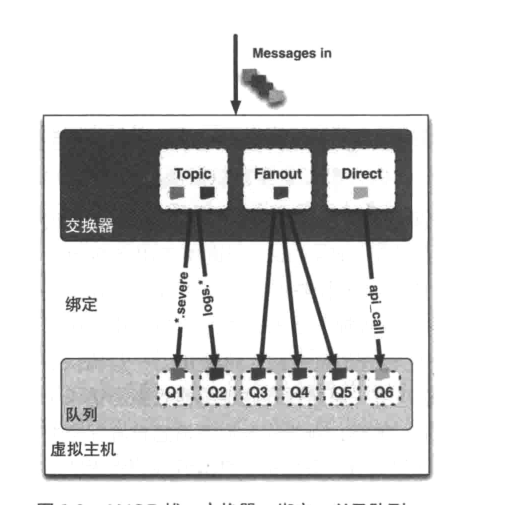
我们必须连接到RabbitMQ才能接受或发布消息。我们的应用程序和RabbitMQ创建一条TCP连接。连接打开后，应用程序就可以创建一条AMQP信道。信道是建立在真实TCP连接上的虚拟连接。AMQP的命令都是通过信息发送出去的。每条信道都会指派唯一的ID（AMQP会帮忙记住）。无论发布或者接受消息都是通过信道完成。

信道相比直接通过TCP连接提升了很大的性能,不断地创或销毁TCP的连接需要消耗操作系统的性能。

### 2. 队列

AMQP消息路由必须有三部分：交换机、队列和绑定。

生产者把消息发送到交换机上;消息最终到达指定的队列,并被消费者接受;绑定决定了消息如何路由到指定的队列。



* **那么消息到达队列后,是如何被消费者接受的？**

**有两种方式：**

1. 通过该AMQP的basic.consume命令订阅。这样做会把信道置为接受模式,知道取消订阅为止。订阅了消息后,消费者在消费(或拒绝)最近接受的那条消息后,就能从队列中自动接受下一条消息。
2. 某些时候,我们只想从队列中获取单条消息。通过AMQP的basic.get名称实现。这样做会使消费者从队列中获取下一条消息。如果想再次获得消息,那么需要再次发送basic.get命令。

* **如果消息到达队列后,无消费者订阅队列怎么办？**

在这种情况下,消息到达队列后会在队列中等待。一旦有消费者订阅该队列,那么队列上的消息就会发送给消费者。

* **如果有多个消费者订阅队列时,消费怎么分发？**

当队列有多个消费者时,队列接受到消息后将以循环的方式发送给消费者。每条消息只会发送给一个消费者。

假设有两个队列,当第一条消息过来时,发送给队列1,当第二条消息过来时发送给队列2,第三条又发送给队列1…,如此循环。

* **RabbitMQ怎么确认消费者正确接受到了消息？**

消费者接受到的每一条消息都必须进行确认。

消费者可以通过**两种方式**进行消息确认:

1. 通过AMQP的basic.ack命令显示地向RabbitMQ发送一个确认命令。
2. 或者在订阅队列的时候就将auto\_ack参数设置为ture。

* **如果消费者没有进行确认怎么办？**

如果消费者收到一条消息,然后确认之前从RabbitMQ上断开了连接(或取消订阅), RabbitMQ会认为这条消息没有分发,然后重新分发给下一个订阅的消费者。

如果程序上有bug忘记进行确认,那么在上一条消息确认之前,RabbitMQ将不会再给这个消费者发送消息,会认为该消费者没有准备好进行下一个消费确认。

如何程序明确的想拒绝这个消息,那么消费可以取消订阅或断开rabbitMQ连接，另一种方法可以使用AMQP的basic.reject命令。如果reject的requeue参数设置为true的话那么RabbitMQ会将消息发送给下一个消费者。而如果将requeue参数设置为false的话,那么该条消息会入“死信”队列,不会被处理也不会

* **怎么创建队列？**

消费者和生产者都能创建队列:使用AMQP的queue.declare命令来创建队列。

**但是创建有条件:**

1. 如果消费者在同一条信道上订阅了其他队列,则无法再创建队列,

需要取消订阅才可创建。

**在创建队列时,有几个有用的参数:**

1. exclusive：如果设置为true的话,队列将变为私有,此时只有我们自己的程序才能消费队列消息。
2. auto-delete:当最后一个消费者取消订阅时,队列就会自动移除。

* **队列是AMQP消息通信的基础模块**

1. 为消息提供了场所,消息在此等待消费。
2. 对负载均衡来说,队列绝对是很好的方案。只需要加一堆消费者,并让rabbitMQ以循环的方式均匀分配发来的消息。
3. 队列是RabbitMQ中消息的最后的终点。

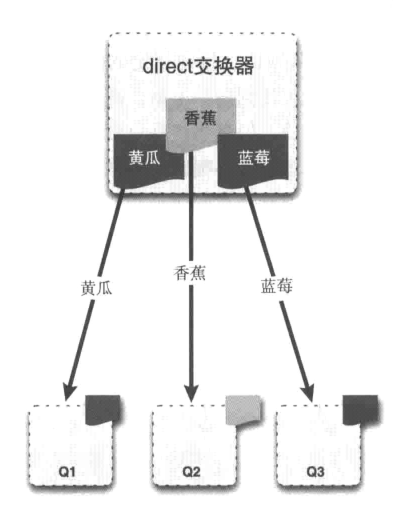
### 3. 交换机和绑定

当我们想把消息投递到队列上,通过把消息交给**交换机**来完成。然后,根据确定的**路由键**,RabbitMQ将会决定消息该投递到哪个队列。

* **一共有四种类型的交换机**

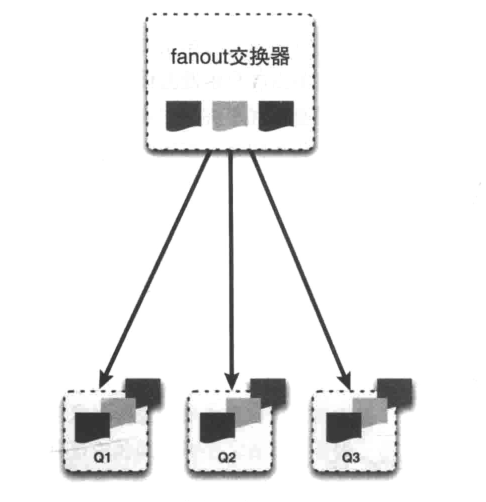
1. **direct交换机:**

如果路由键匹配的话,消息就被投递到对应的队列。



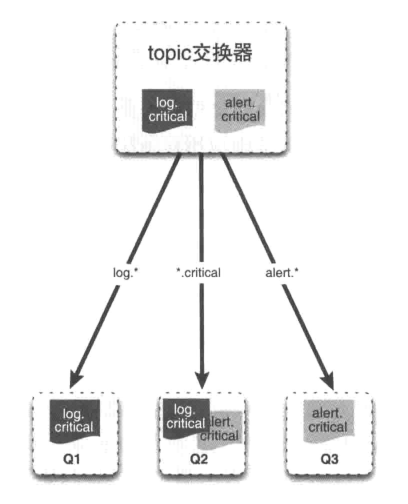
**2.fanout交换器:**

当我们发送一条消息到fanout交换器的时候,它会把消息投递到所有附加在交换机上的队列。



**3. topic交换器**

该交换器可以使来自不同源头的消息能够到达同一个队列。



### 4. 虚拟主机

每一个虚拟主机本质上是一个迷你版的RabbitMQ服务器,拥有自己的队列，交换机和绑定还有权限控制。

Vhost通过在实例间的逻辑分离,允许为不同应用程序安全保密的运行。

可以区各个客户,避免队列和交换机命名冲突。