# RabbitMQ

## 下载安装

* 下载Erlang：<http://www.erlang.org/download/otp_win64_17.3.exe>。
* 下载RabbitMQ：<http://www.rabbitmq.com/download.html>。
* 配置两个的环境变量
* 配置RabbitMQ：安装目录/sbin，cmd，输入rabbitmq-plugins enable rabbitmq\_management用来启动管理插件，以管理员运行cmd，输入net stop(start) rabbitmq来启动关闭mq。如果失败的话，删除C:\Users\xuyabiao\AppData\Roaming\RabbitMQ\db\下面的内容，重新安装rabbitmq
* 进入<http://localhost:15672/>，默认账密guest

## 用户及vhost配置

### 添加用户

* admin >> add a user >> 输入账密及标签 >> add user。标签如下：
  + 超级管理员(administrator)：可登陆管理控制台，可查看所有的信息，并且可以对用户，策略(policy)进行操作。
  + 监控者(monitoring)：可登陆管理控制台，同时可以查看rabbitmq节点的相关信息(进程数，内存使用情况，磁盘使用情况等)
  + 策略制定者(policymaker)：可登陆管理控制台, 同时可以对policy进行管理。但无法查看节点的相关信息(上图红框标识的部分)。
  + 普通管理者(management)：仅可登陆管理控制台，无法看到节点信息，也无法对策略进行管理。
  + 其他：无法登陆管理控制台，通常就是普通的生产者和消费者。

## virtual hosts管理

**virtual hosts相当于mysql的db。**

### 添加virtual hosts

* virtual hosts >> add a new virtual hosts >> 输入name，一般以/开头 >> add virtual host

### 对用户进行授权

* 点击注册的host name >> set permission >> user >>　选择自己的hostname >> set permission

## 界面简介

|  |  |
| --- | --- |
|  | 协议amqp  端口5672  访问端口15673  集群端口25672 |

## 交换机(转发器)Exchange

### 作用：

* 接收生产者的消息。
* 向队列推送消息。
* 是第一次启动消息发送者才生成交换机，所以如果第一次先启动的是消费者，会报错，第二次(当然消息发送者得运行一次)就不会了。

### 类型

* fanout：不处理路由键。

|  |  |
| --- | --- |
| fanout不处理路由键，消息发送到交换机，则交换机会将消息发送到所有与之绑定的队列。 |  |
| direct处理路由键，交换机和队列都有一个key，如果一样的话，交换机会发送到此队列。用于路由模式 |  |
| Topic，根据key值进行匹配。\*匹配一个，#匹配多个。 |  |
|  |  |
|  |  |
|  |  |

## 消息应答acknowledgement

* autoAck为true：自动确认模式，默认是此模式，rabbitmq将消息发送给消费者，rabbitmq-server就会将消息从内存中删除。这种情况有可能会丢失消息。
* autoAck为false：手动模式，如果有一个消费者挂掉，rabbitmq-server会将消息发送给其它消费者。当消费者发送一个ack时，即告诉server处理成功，可以从server内存中删除消息。
* 消费者boolean autoAck = false; channel.basicConsume(QUEUE\_NAME, autoAck, consumer);

## 持久化durable

* 但是有了ack后还是会有问题，如果服务器挂了数据依然会丢失，所以要进行消息持久化。
* 生产者声明队列：
  + boolean durable = false;
  + channel.queueDeclare(QUEUE\_NAME, durable, false, false, null);
  + 注意，如果rabbitmq已经有一个durable为false的队列，修改为true，重新运行程序会失败。因为rabbitmq不允许重新定义一个已经存在的队列。除非在mq中删除此队列。或重新起个对列名。

## 五种队列

P：消息生产者。C：消费者。红框(或Q)：消息队列。X：交换机。

### simple-queues简单队列

|  |
| --- |
|  |
|  |

### work-queues工作队列

* 解决问题：消息生产者生产消息速度快，消费者需要处理业务逻辑，时间长，需要多个消费者。

|  |
| --- |
|  |

* 轮询发放Round-robin：将消息平均分给所有消费者，不管谁闲谁忙。
* 公平分发fair dispatch：谁干的快，就干的多，要设置生产者和消费者的basicQos和消费者的手动应答。

### 发布订阅模式publish/subscribe

* 1个P，多个C
* 每个C有自己的Q
* P没有直接把消息发送到Q，而是发送到X
* 每个Q都要绑定到X
* P发送的消息，经过X，到达Q就可实现一个消息被多个C消费。例如微信公众号

|  |
| --- |
|  |

* 注意rabbitmq中只有队列有储存消息的能力，交换机没有保存消息的能力。队列是在消费者中绑定的。

### 路由模式Routing

|  |  |
| --- | --- |
|  | 如果发送消息带error的key，则会发送到c1和c2所绑定队列。如果发送的消息的key为info或warning，则发送到c2。 |

也有缺点：路由表必须明确，且有些队列对应的key非常多，不可能全部列出来。

### 将路由和某个模式匹配Topic

|  |
| --- |
|  |
| #匹配一个或多个，\*匹配一个，routingKey匹配的才能发送消息。 |

# rabbitmq的消息确认机制(事务+confirm，生产者)

* 在rabbitmq中可以通过持久化数据解决rabbitmq服务器异常的数据丢失问题。
* 生产者将消息发送出去以后，消息到底有没有到达mq，默认不知道。可以通过两种方式解决：AMQP实现了事务机制，或者confirm模式。

## AMQP的事务机制

* **txSelect**：用户将当前channel设置为transation模式。channel.txSelect();
* txCommit：用于提交事务。
* txRollback：用于回滚事务。
* 会降低吞吐量，因为要多次连接。

## Confirm机制

### 原理

* 生产者将channel设置为confirm模式，一旦消息进入confirm模式，所有在该信道的消息都会被指派唯一id(从1开始)，一旦消息被投递到所有匹配的队列后，broker就会发送一个消息确认给生产者(包含消息的唯一id)，使得生产者知道消息已经正确到达队列，如果消息和队列可持久化，则会在消息写入磁盘后返回确认信息deliver-tag，其中包含了序列号，此外，broker也可以设置basic.ack的multiple属性，表示到这个序列号之前的消息都已经得到处理。
* 且有异步的优势。
* 发送异常会收到Nack消息。

### 开启方式

channel.confirmSelect();，有三种编程模式

* 普通单条：waitForConfirms();返回true，发送成功。每条确认。
* 批量：waitForConfirms();不用每个确认一次，而是一次性确认一批，但是如果出错的话，也是直接返回一批。
* 异步，提供回调：channel对象提供的ConfirmListener()回调方法包含deliveryTag(当前channel发出的序列号)，我们需要为自己为每一个Channel维护一个unconfirm的消息序号集合，每publish一条数据，集合中元素加1，每回调一次handleAck方法，unconfirm集合删掉相应的一条(multiple = false)或多条记录(multiple = true)，unconfirm最好采用SortedSet。