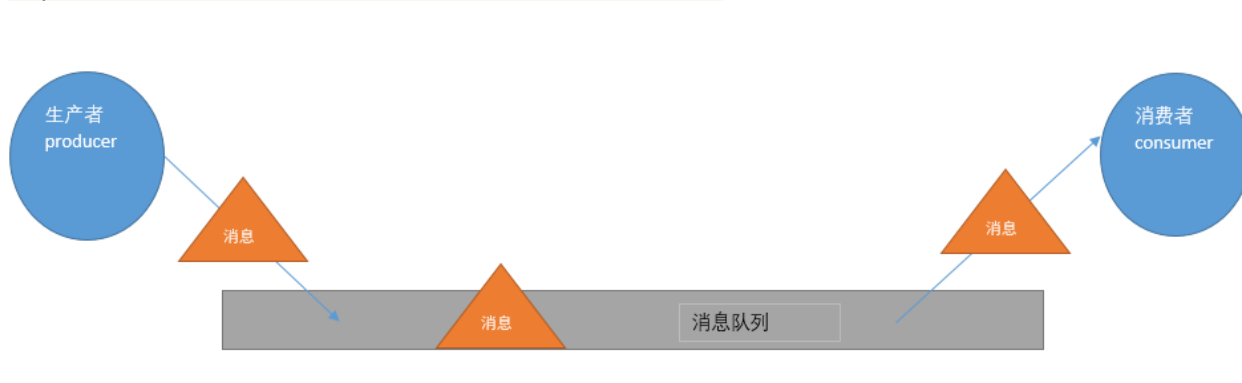


一、RabbitMQ概念

RabbitMQ是流行的开源消息队列系统，是AMQP（Advanced Message Queuing Protocol高级消息队列协议）的标准实现，用erlang语言开发。RabbitMQ据说具有良好的性能和时效性，同时还能够非常好的支持集群和负载部署，非常适合在较大规模的分布式系统中使用。

消息中间件的工作过程可以用生产者消费者模型来表示。即，生产者不断的向消息队列发送信息，而消费者从消息队列中消费信息。具体过程如下：



从上图可看出，对于消息队列来说，生产者，消息队列，消费者是最重要的三个概念，生产者发消息到消息队列中去，消费者监听指定的消息队列，并且当消息队列收到消息之后，接收消息队列传来的消息，并且给予相应的处理。消息队列常用于分布式系统之间互相信息的传递。

Rabbit模式大概分为以下三种：单一模式、普通模式、镜像模式

单一模式：最简单的情况，非集群模式，即单实例服务。

普通模式：默认的集群模式。

queue创建之后，如果没有其它policy，则queue就会按照普通模式集群。对于Queue来说，消息实体只存在于其中一个节点，A、B两个节点仅有相同的元数据，即队列结构，但队列的元数据仅保存有一份，即创建该队列的rabbitmq节点（A节点），当A节点宕机，你可以去其B节点查看，`./rabbitmqctl list_queues`发现该队列已经丢失，但声明的exchange还存在。

当消息进入A节点的Queue中后，consumer从B节点拉取时，RabbitMQ会临时在A、B间进行消息传输，把A中的消息实体取出并经过B发送给consumer。

所以consumer应尽量连接每一个节点，从中取消息。即对于同一个逻辑队列，要在多个节点建立物理Queue。否则无论consumer连A或B，出口总在A，会产生瓶颈。

该模式存在一个问题就是当A节点故障后，B节点无法取到A节点中还未消费的消息实体。

如果做了消息持久化，那么得等A节点恢复，然后才可被消费；如果没有持久化的话，队列数据就丢失了。

镜像模式：把需要的队列做成镜像队列，存在于多个节点，属于RabbitMQ的HA方案。

该模式解决了上述问题，其实质和普通模式不同之处在于，消息实体会主动在镜像节点间同步，而不是在consumer取数据时临时拉取。

该模式带来的副作用也很明显，除了降低系统性能外，如果镜像队列数量过多，加之大量的消息进入，集群内部的网络带宽将会被这种同步通讯大大消耗掉。

所以在对可靠性要求较高的场合中适用，一个队列想做成镜像队列，需要先设置policy，然后客户端创建队列的时候，rabbitmq集群根据“队列名称”自动设置是普通集群模式或镜像队列。具体如下：

队列通过策略来使能镜像。策略能在任何时刻改变，rabbitmq队列也尽可能的将队列随着策略变化而变化；非镜像队列和镜像队列之间是有区别的，前者缺乏额外的镜像基础设施，没有任何slave，因此会运行得更快。

为了使队列称为镜像队列，你将会创建一个策略来匹配队列，设置策略有两个键“ha-mode 和 ha-params（可选）”。

了解集群中的基本概念：

RabbitMQ的集群节点包括内存节点、磁盘节点。顾名思义内存节点就是将所有数据放在内存，磁盘节点将数据放在磁盘。不过，如前文所述，如果在投递消息时，打开了消息的持久化，那么即使是内存节点，数据还是安全的放在磁盘。

一个rabbitmq集群中可以共享user，vhost，queue，exchange等，所有的数据和状态都是必须在所有节点上复制的，一个例外是，那些当前只属于创建它的节点的消息队列，尽管它们可见且可被所有节点读取。rabbitmq节点可以动态的加入到集群中，一个节点它可以加入到集群中，也可以从集群环集群会进行一个基本的负载均衡。

集群中有两种节点：

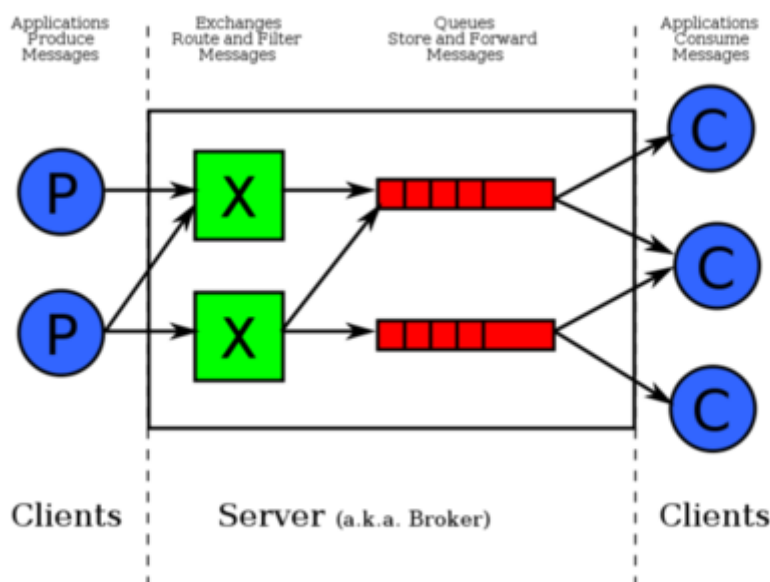
1 内存节点：只保存状态到内存（一个例外的情况是：持久的queue的持久内容将被保存到disk）

2 磁盘节点：保存状态到内存和磁盘。

内存节点虽然不写入磁盘，但是它执行比磁盘节点要好。集群中，只需要一个磁盘节点来保存状态就足够了如果集群中只有内存节点，那么不能停止它们，否则所有的状态，消息等都会丢失。

RabbitMQ的工作流程

对于RabbitMQ来说,除了这三个基本模块以外,还添加了一个模块,即交换机(Exchange).它使得生产者和消息队列之间产生了隔离,生产者将消息发送给交换机,而交换机则根据调度策略把相应的消息转发给对应的消息队列.那么RabbitMQ的工作流程如下所示:



交换机的主要作用是接收相应的消息并且绑定到指定的队列.交换机有四种类型,分别为 Direct,topic,headers,Fanout.

1).Direct是RabbitMQ默认的交换机模式,也是最简单的模式.即创建消息队列的时候,指定一个BindingKey.当发送者发送消息的时候,指定对应的Key.当Key和消息队列的BindingKey一致的时候,消息将会被发送到该消息队列中.

2).topic转发信息主要是依据通配符,队列和交换机的绑定主要是依据一种模式(通配符+字符串),而当发送消息的时候,只有指定的Key和该模式相匹配的时候,消息才会被发送到该消息队列中.

3).headers也是根据一个规则进行匹配,在消息队列和交换机绑定的时候会指定一组键值对规则,而发送消息的时候也会指定一组键值对规则,当两组键值对规则相匹配的时候,消息会被发送到匹配的消息队列中.

4).Fanout是路由广播的形式,将会把消息发给绑定它的全部队列,即便设置了key,也会被忽略.

1.centos7中安装RabbitMQ

1).首先需要安装erlang

```
#rpm -Uvh http://download.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-8.noarch.rpm
```

```
#yum install erlang
```

安装过程中会有提示,一路输入“y”即可。

测试是否安装成功:

```
[root@localhost rabbitMQ]# erl
Erlang/OTP 19 [erts-8.0.3] [source] [64-bit] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V8.0.3 (abort with ^G)
1> 5+6.
11
2> halt().
[root@localhost rabbitMQ]#
```

下载安装包:

wget <http://www.rabbitmq.com/releases/rabbitmq-server/v3.6.6/rabbitmq-server-3.6.6-1.el7.noarch.rpm>

安装: `rpm -ivh rabbitmq-server-3.6.6-1.el7.noarch.rpm`

第四、启动和关闭:---/sbin为根目录

`/sbin/service rabbitmq-server stop` #关闭

`/sbin/service rabbitmq-server start` #启动

`/sbin/service rabbitmq-server status` #状态

第五、cd 到/sbin目录下:

```
[root@localhost rabbitMQ]# cd /sbin/
```

`./rabbitmq-plugins list`

```
[root@localhost sbin]# ./rabbitmq-plugins list
Configured: E = explicitly enabled; e = implicitly enabled
| Status:  * = running on rabbit@localhost
|/
[ ] amqp_client          3.6.6
[ ] cowboy              1.0.3
[ ] cowlib              1.0.1
[ ] mochiweb            2.13.1
[ ] rabbitmq_amqp1_0    3.6.6
[ ] rabbitmq_auth_backend_ldap 3.6.6
[ ] rabbitmq_auth_mechanism_ssl 3.6.6
[ ] rabbitmq_consistent_hash_exchange 3.6.6
[ ] rabbitmq_event_exchange 3.6.6
[ ] rabbitmq_federation 3.6.6
[ ] rabbitmq_federation_management 3.6.6
[ ] rabbitmq_jms_topic_exchange 3.6.6
[ ] rabbitmq_management 3.6.6
[ ] rabbitmq_management_agent 3.6.6
[ ] rabbitmq_management_visualiser 3.6.6
[ ] rabbitmq_mqtt        3.6.6
[ ] rabbitmq_recent_history_exchange 1.2.1
[ ] rabbitmq_sharding    0.1.0
[ ] rabbitmq_shovel      3.6.6
[ ] rabbitmq_shovel_management 3.6.6
[ ] rabbitmq_stomp       3.6.6
[ ] rabbitmq_top         3.6.6
[ ] rabbitmq_tracing     3.6.6
[ ] rabbitmq_trust_store 3.6.6
[ ] rabbitmq_web_dispatch 3.6.6
[ ] rabbitmq_web_stomp   3.6.6
[ ] rabbitmq_web_stomp_examples 3.6.6
[ ] sockjs              0.3.4
[ ] webmachine          1.10.3
```

`./rabbitmqctl status`

```
[root@localhost sbin]# ./rabbitmqctl status
Status of node rabbit@localhost ...
[{pid,29560},
 {running_applications,[{rabbit,"RabbitMQ","3.6.6"},
                        {mnesia,"MNESIA CXC 138 12","4.14"},
                        {rabbit_common,[],"3.6.6"},
                        {os_mon,"CPO CXC 138 46","2.4.1"},
                        {xmerl,"XML parser","1.3.11"},
                        {franch,"Socket acceptor pool for TCP protocols.",
                                "1.2.1"},
                        {sasldb,"SASL CXC 138 11","3.0"},
                        {stdlib,"ERTS CXC 138 10","3.0.1"},
                        {kernel,"ERTS CXC 138 10","5.0.1"}]},
 {os,{unix,linux}},
 {erlang_version,"Erlang/OTP 19 [erts-8.0.3] [source] [64-bit] [async-threads:64] [hipec] [kernel-poll:true]\n"},
 {memory,[{total,38658424},
          {connection_readers,0},
          {connection_writers,0},
          {connection_channels,0},
          {connection_other,0},
          {queue_procs,2688},
          {queue_slave_procs,0},
          {plugins,0},
          {other_proc,13299896},
          {mnesia,58200},
          {mgmt_db,0},
          {msg_index,43744},
          {other_ets,929872},
          {binary,10848},
          {code,17760058},
          {atom,752561},
          {other_system,5800557}}],
 {alarms,[]},
 {listeners,[{clustering,25672,"::"},{amqp,5672,"::"}]},
 {vm_memory_high_watermark,0.4},
 {vm_memory_limit,410145587},
 {disk_free_limit,500000000},
 {disk_free,16946331648},
 {file_descriptors,[{total_limit,924},
                    {total_used,2},
                    {sockets_limit,829},
                    {sockets_used,0}]},
 {processes,[{limit,1048576},{used,137}]},
 {run_queue,0},
 {uptime,552},
 {kernel,{net_ticktime,60}}]
```

第六、其他

运行如下的命令，增加用户admin，密码admin

#添加用户

#./rabbitmqctl add_user 账号 密码

./rabbitmqctl add_user admin admin

#分配用户标签(admin为要赋予administrator权限的刚创建的那个账号的名字)

./rabbitmqctl set_user_tags admin administrator

#设置权限<即开启远程访问>(如果需要远程连接,例如java项目中需要调用mq,则一定要配置,否则无法连接到mq,admin为要赋予远程访问权限的刚创建的那个账号的名字,必须运行着rabbitmq此命令才能执行)

./rabbitmqctl set_permissions -p "/" admin ".*" ".*" ".*"

[root@localhost sbin]# ./rabbitmqctl list_users

Listing users ...

admin [administratr]

guest [administrator]

...done.

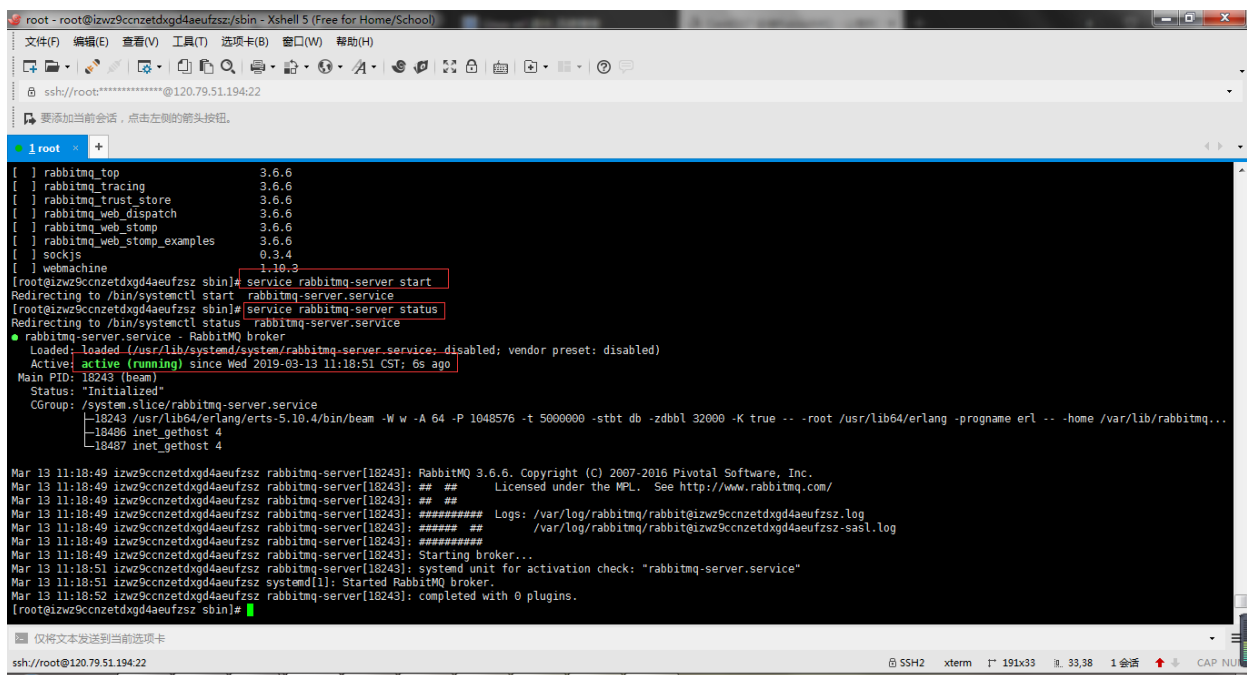
2、执行rabbitmq命令行工具 (rabbitmqctl) :

`rabbitmqctl -q status` //打印了一些rabbitmq服务状态信息，包括内存，硬盘，和使用erlong的版本信息

`rabbitmqctl list_queues` //查看所有队列消息

`rabbitmqctl reset` 清除所有队列

启动：



```
root@izw29ccnztzxd4eufsz:~# service rabbitmq-server start
Redirecting to /bin/systemctl start rabbitmq-server.service
root@izw29ccnztzxd4eufsz:~# service rabbitmq-server status
Redirecting to /bin/systemctl status rabbitmq-server.service
● rabbitmq-server.service - RabbitMQ broker
   Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2019-03-13 11:18:51 CST; 6s ago
     Main PID: 18243 (beam)
    Status: "Initialized"
      CGroup: /system.slice/rabbitmq-server.service
              └─18243 /usr/lib64/erlang/erts-5.10.4/bin/beam -W w -A 64 -P 1048576 -t 5000000 -stbt db -zdbbl 32000 -K true -- -root /usr/lib64/erlang -programe erl -- -home /var/lib/rabbitmq...
              └─18486 inet_gethost 4
              └─18487 inet_gethost 4

Mar 13 11:18:49 izw29ccnztzxd4eufsz rabbitmq-server[18243]: RabbitMQ 3.6.6. Copyright (C) 2007-2016 Pivotal Software, Inc.
Mar 13 11:18:49 izw29ccnztzxd4eufsz rabbitmq-server[18243]: ## ## Licensed under the MPL. See http://www.rabbitmq.com/
Mar 13 11:18:49 izw29ccnztzxd4eufsz rabbitmq-server[18243]: ## ## Logs: /var/log/rabbitmq/rabbit@izw29ccnztzxd4eufsz.log
Mar 13 11:18:49 izw29ccnztzxd4eufsz rabbitmq-server[18243]: ## ## /var/log/rabbitmq/rabbit@izw29ccnztzxd4eufsz-sasl.log
Mar 13 11:18:49 izw29ccnztzxd4eufsz rabbitmq-server[18243]: ## ##
Mar 13 11:18:49 izw29ccnztzxd4eufsz rabbitmq-server[18243]: Starting broker...
Mar 13 11:18:51 izw29ccnztzxd4eufsz rabbitmq-server[18243]: systemd unit for activation check: "rabbitmq-server.service"
Mar 13 11:18:51 izw29ccnztzxd4eufsz systemd[1]: Started RabbitMQ broker.
Mar 13 11:18:52 izw29ccnztzxd4eufsz rabbitmq-server[18243]: completed with 0 plugins.
root@izw29ccnztzxd4eufsz:~#
```

`rabbitmq-server start` 或者 `service rabbitmq-server start` #启动rabbitmq

`rabbitmqctl list_exchanges`

`rabbitmqctl list_bindings`

`rabbitmqctl list_queues` #分别查看当前系统种存在的Exchange和Exchange上绑定的Queue信息。

`rabbitmqctl status` #查看运行信息

`rabbitmqctl stop` #停止运行rabbitmq

