centos 6.5

rabbitmq 无网络安装：

rpm -ivh rabbitmq-server-3.7.14-1.el6.noarch.rpm

发现需要以下

1:socat：rpm -ivh socat-1.7.3.2-2.el7.x86\_64.rpm --nodeps

#使用 strings /lib64/libc.so.6 | grep GLIBC 可以查看 glib 号

# 需要glib

# wget http://ftp.gnu.org/gnu/glibc/glibc-2.14.tar.gz

# tar xf glibc-2.14.tar.gz

# cd glibc-2.14

# mkdir build

# cd build

# ../configure --prefix=/opt/glibc-2.14 --disable-profile --enable-add-ons --with-headers=/usr/include --with-binutils=/usr/bin

#make

# make install

###2:logrotate：

3:erlang : rpm -ivh erlang-21.3.3-1.el6.x86\_64.rpm

开启服务 ：service rabbitmq-server start

开启控制台管理插件：rabbitmq-plugins enable rabbitmq\_management

添加用户 ：rabbitmqctl add\_user admin 123456

赋予角色 : rabbitmqctl set\_user\_tags admin administrator

防火墙 添加端口：

vi /etc/sysconfig/iptables

#RabbitMQ

-A INPUT -m state --state NEW -m tcp -p tcp --dport 15672 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 25672 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 5672 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 4369 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 5671 -j ACCEPT

#RabbitMQ

相关命令：

rpm -qa | grep jdk

rpm -e --nodeps 要卸载的软件包

# 集群模式配置

设计架构

可以如下：在一个集群里，有 4 台机器，其中 1 台使用磁盘模式，另 2 台使用内存模式。

2 台内存模式的节点，无疑速度更快，因此客户端（consumer、producer）连接访问它们。而磁盘模式的节点，

由于磁盘 IO 相对较慢，因此仅作数据备份使用，另外一台作为反向代理。

注意：！！！！

先修改 hosts文件

vi /etc/hosts

192.168.56.103 rabbitmq\_node1

192.168.56.104 rabbitmq\_node2

再用主机名登录

ssh root@rabbitmq\_node1

保证相同的Erlang Cookie（各个节点的rabbitmq .erlang.cookie 这个文件相同）

我们使用rpm等安装包方式进行安装的，那么 .erlang.cookie 这个文件会在/var/lib/rabbitmq目录下。

[root@localhost rabbitmq]# pwd

/var/lib/rabbitmq

[root@localhost rabbitmq]# ls -a

. .. .erlang.cookie mnesia

下面是我通过rpm包安装的rabbitmq的启动日志。

=INFO REPORT==== 21-Sep-2016::21:17:42 ===

node : rabbit@he07

home dir : /var/lib/rabbitmq

config file(s) : /etc/rabbitmq/rabbitmq.config (not found)

cookie hash : qhOGp9TtH4Rn+BekiYXxIg==

log : /var/log/rabbitmq/rabbit@he07.log

sasl log : /var/log/rabbitmq/rabbit@he07-sasl.log

database dir : /var/lib/rabbitmq/mnesia/rabbit@he07

保证相同的Erlang Cookie

我这里是把103上的.erlang.cookie以scp的方式拷贝到另外两台机器

chmod 777 /var/lib/rabbitmq/.erlang.cookie

scp /var/lib/rabbitmq/.erlang.cookie root@192.168.56.104:/var/lib/rabbitmq

chmod 400 /var/lib/rabbitmq/.erlang.cookie

先把103的rabbitmq停止 ，仅仅关闭应用： rabbitmqctl stop\_app (关闭整个节点（包括应用）：./rabbitmqctl stop)

rabbitmqctl join\_cluster rabbit@rabbitmq\_node2

[root@rabbitmq\_node1 ~]# rabbitmqctl stop\_app

Stopping rabbit application on node rabbit@rabbitmq\_node1 ...

[root@rabbitmq\_node1 ~]# rabbitmqctl join\_cluster --ram rabbit@rabbitmq\_node2

Clustering node rabbit@rabbitmq\_node1 with rabbit@rabbitmq\_node2

然后再启动103 ： rabbitmqctl start\_app

使用命令查看 集群状态 ：rabbitmqctl cluster\_status

[root@rabbitmq\_node1 ~]# rabbitmqctl cluster\_status

Cluster status of node rabbit@rabbitmq\_node1 ...

[{nodes,[{disc,[rabbit@rabbitmq\_node2]},{ram,[rabbit@rabbitmq\_node1]}]},

{running\_nodes,[rabbit@rabbitmq\_node2,rabbit@rabbitmq\_node1]},

{cluster\_name,<<"rabbit@rabbitmq\_node2">>},

{partitions,[]},

{alarms,[{rabbit@rabbitmq\_node2,[]},{rabbit@rabbitmq\_node1,[]}]}]

集群相关命令：

更改节点属性

#rabbitmqctl stop\_app –停止rabbitmq服务

#rabbitmqctl change\_cluster\_node\_type disc/ram –更改节点为磁盘或内存节点

#rabbitmqctl start\_app –开启rabbitmq服务

RABBITMQ退出集群

假设要把 rabbitmq\_node1 退出集群

在 rabbitmq\_node1 上执行

#rabbitmqctl stop\_app

#rabbitmqctl reset

#rabbitmqctl start\_app

RABBITMQ集群重启

集群重启时，最后一个挂掉的节点应该第一个重启，如果因特殊原因（比如同时断电），而不知道哪个节点最后一个挂掉。可用以下方法重启：

先在一个节点上执行

#rabbitmqctl force\_boot

#service rabbitmq-server start

在其他节点上执行

#service rabbitmq-server start

查看cluster状态是否正常（要在所有节点上查询）。

#rabbitmqctl cluster\_status

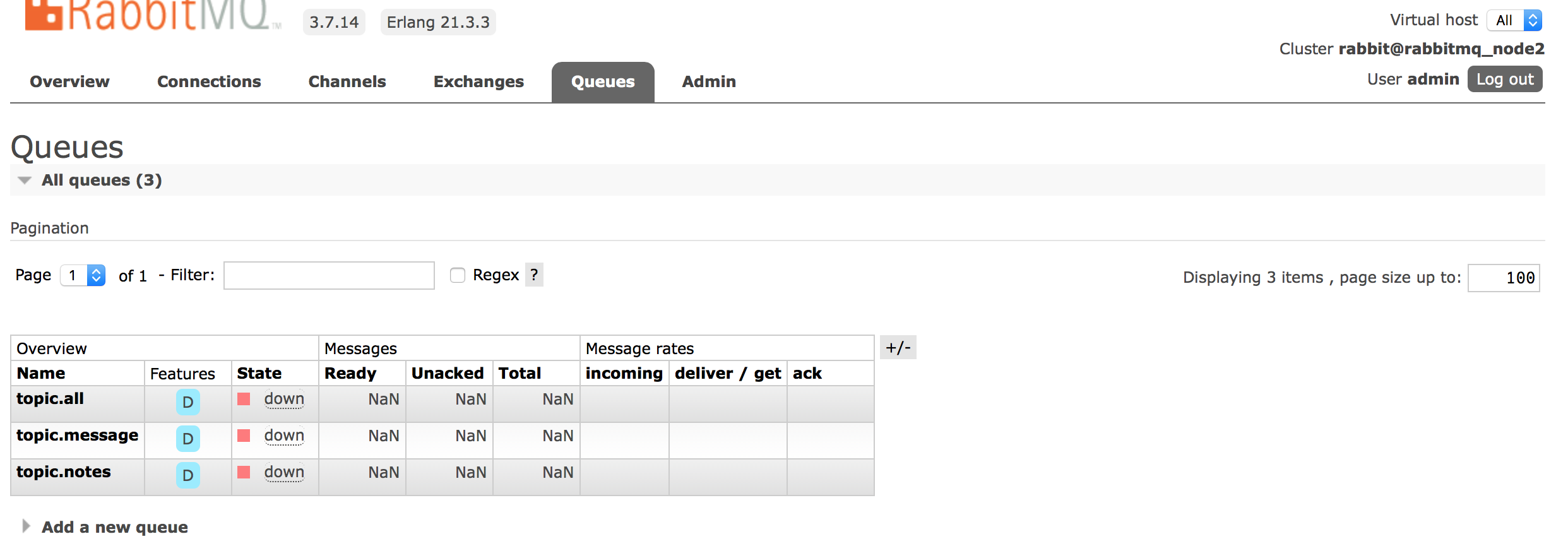
集群测试：普通模式：默认的集群模式

刚开始，rabbitmq\_node1 和rabbitmq\_node2都是disc （硬盘）模式

1：测试发送消息 到 rabbitmq\_node1 上， rabbitmq\_node2 的控制台 queue中 也能查看到消息

customer 连接到rabbitmq\_node2上也能消费到数据

2：关闭消费者，发送消息 到 rabbitmq\_node1（） 上，然后再关闭rabbitmq\_node1服务，发现控制台中 的queue 全部不可用



启动消费者去消费rabbitmq\_node2数据，发现报错连不上

重新启动rabbitmq\_node1 数据还存在，可以被连接在rabbitmq\_node2的消费

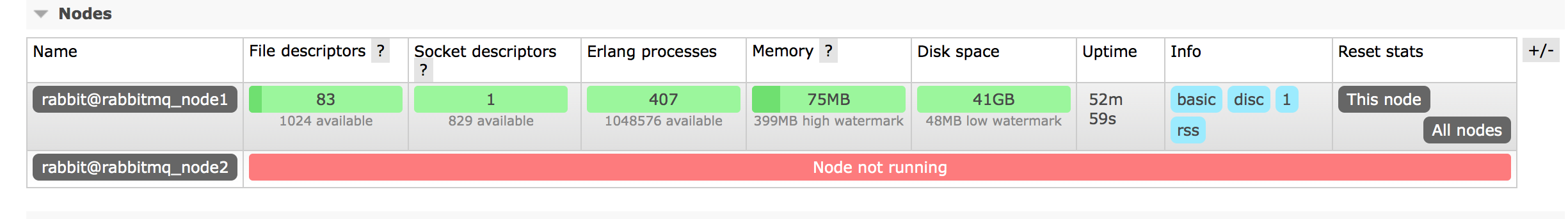
将rabbitmq\_node1改为ram 模式，启动后发现 原先的数据丢失，关掉消费者， 发送消息到rabbitmq\_node1，在关闭abbitmq\_node1，测试数据会不会丢失，结果数据还在,

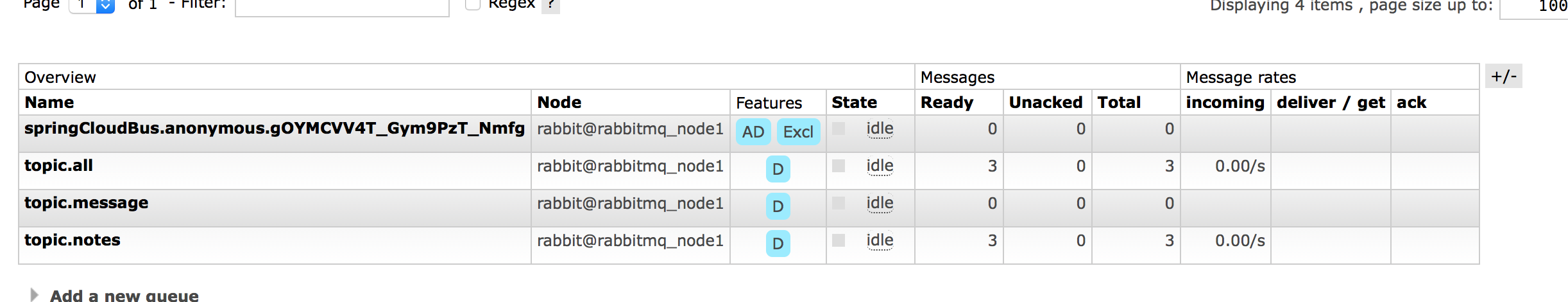
Rabbitmq broker 切换 ram和disc 模式的时候会丢失全部数据

发送消息到 rabbitmq\_node1 再关闭rabbitmq\_node2 ， 查看集群情况

结果：

rabbitmq\_node1 正常

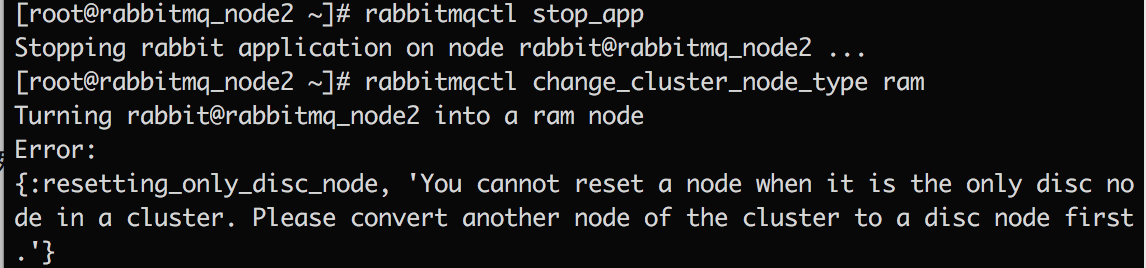




关闭机器电源，测试数据是否会丢

结果： 数据不丢失

将所有节点设置为 ram 模式， 会提示



不让修改，至少保证一个节点为disc模式

**一、普通模式：**默认的集群模式

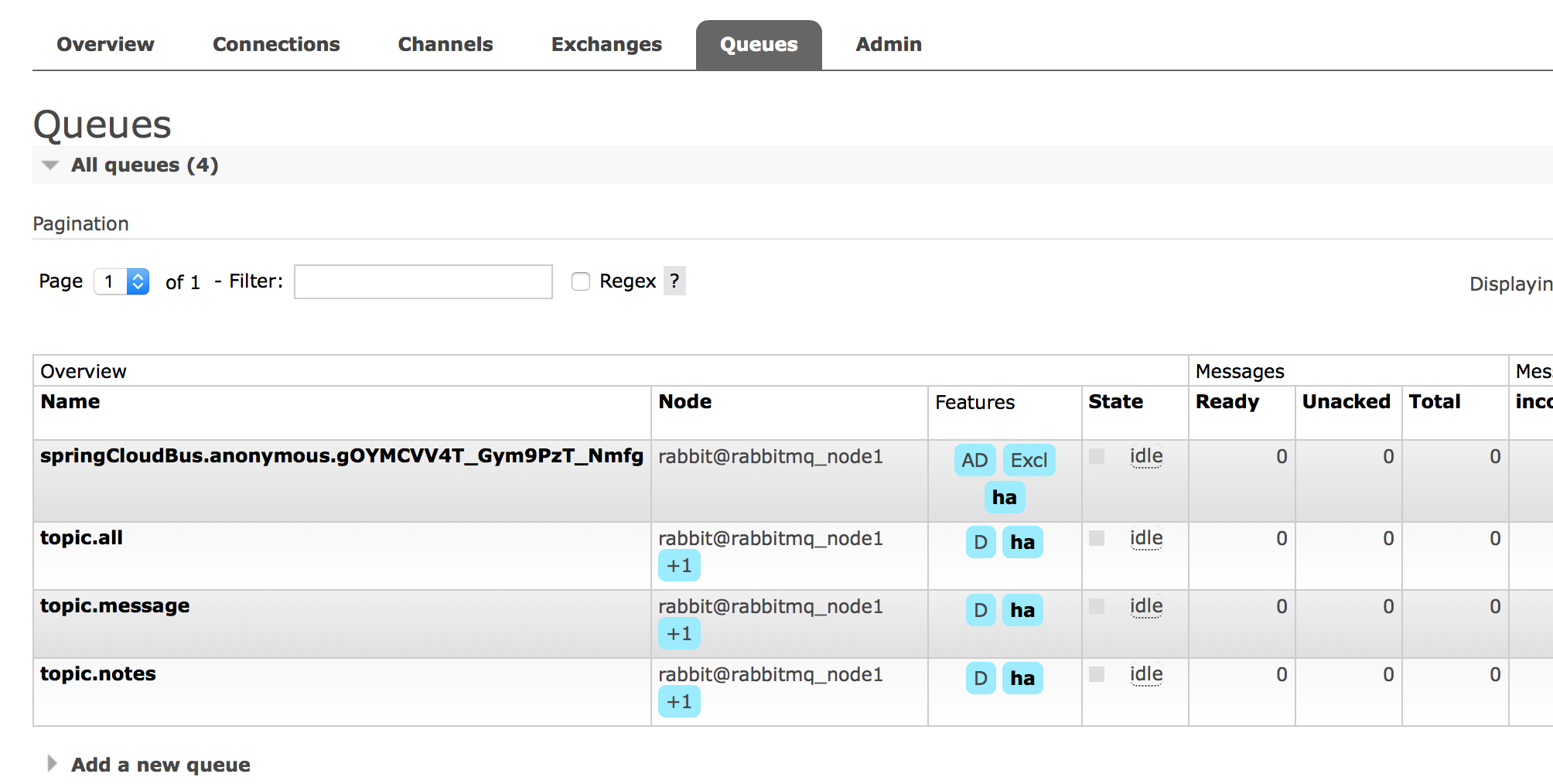
默认的集群模式，queue创建之后，如果没有其它policy，则queue就会按照普通模式集群。对于Queue来说，消息实体只存在于其中一个节点，A、B两个节点仅有相同的元数据，即队列结构，但队列的元数据仅保存有一份，即创建该队列的rabbitmq节点（A节点），当A节点宕机，你可以去其B节点查看，./rabbitmqctl list\_queues 发现该队列已经丢失，但声明的exchange还存在。当消息进入A节点的Queue中后，consumer从B节点拉取时，RabbitMQ会临时在A、B间进行消息传输，把A中的消息实体取出并经过B发送给consumer，所以consumer应平均连接每一个节点，从中取消息。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。如果做了队列持久化或消息持久化，那么得等A节点恢复，然后才可被消费，并且在A节点恢复之前其它节点不能再创建A节点已经创建过的持久队列；如果没有持久化的话，消息就会失丢。这种模式更适合非持久化队列，只有该队列是非持久的，客户端才能重新连接到集群里的其他节点，并重新创建队列。假如该队列是持久化的，那么唯一办法是将故障节点恢复起来。

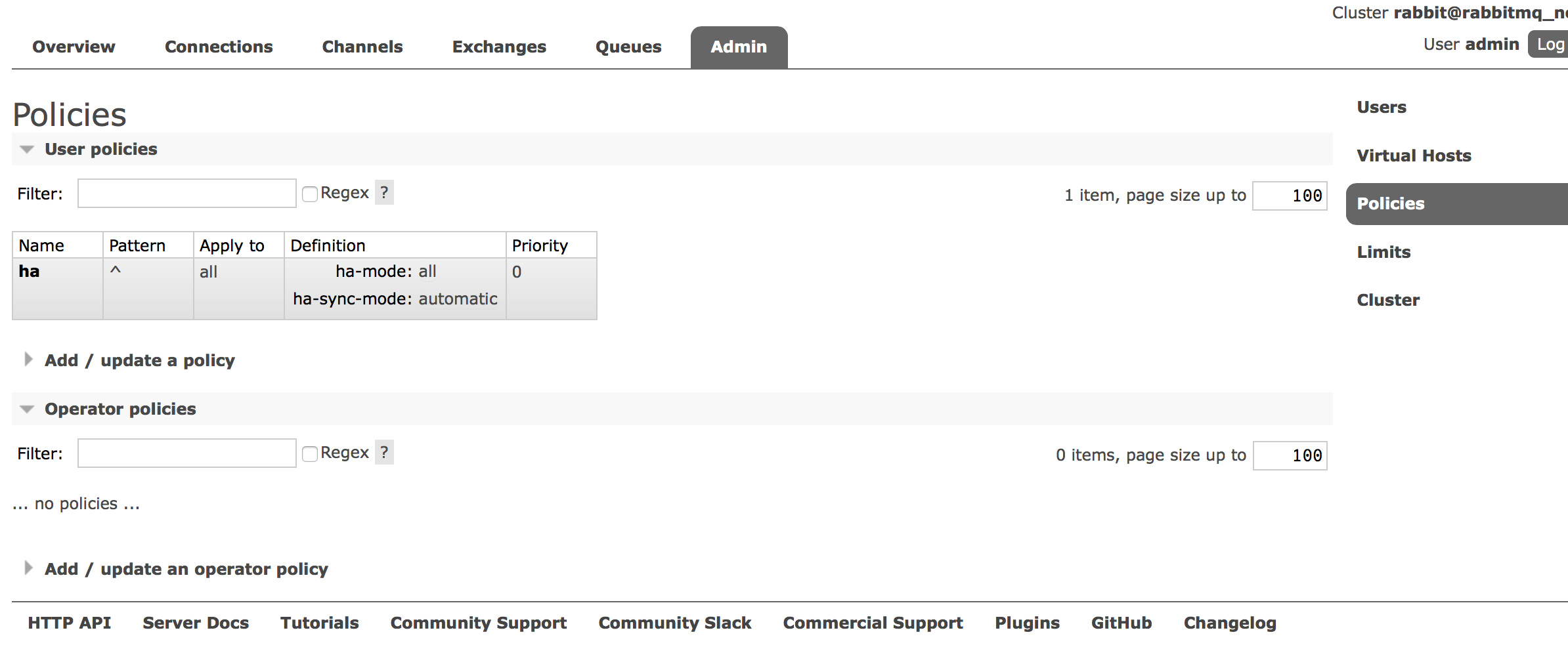
镜像模式测试：（镜像模式是在普通模式的基础上，增加一些镜像策略）

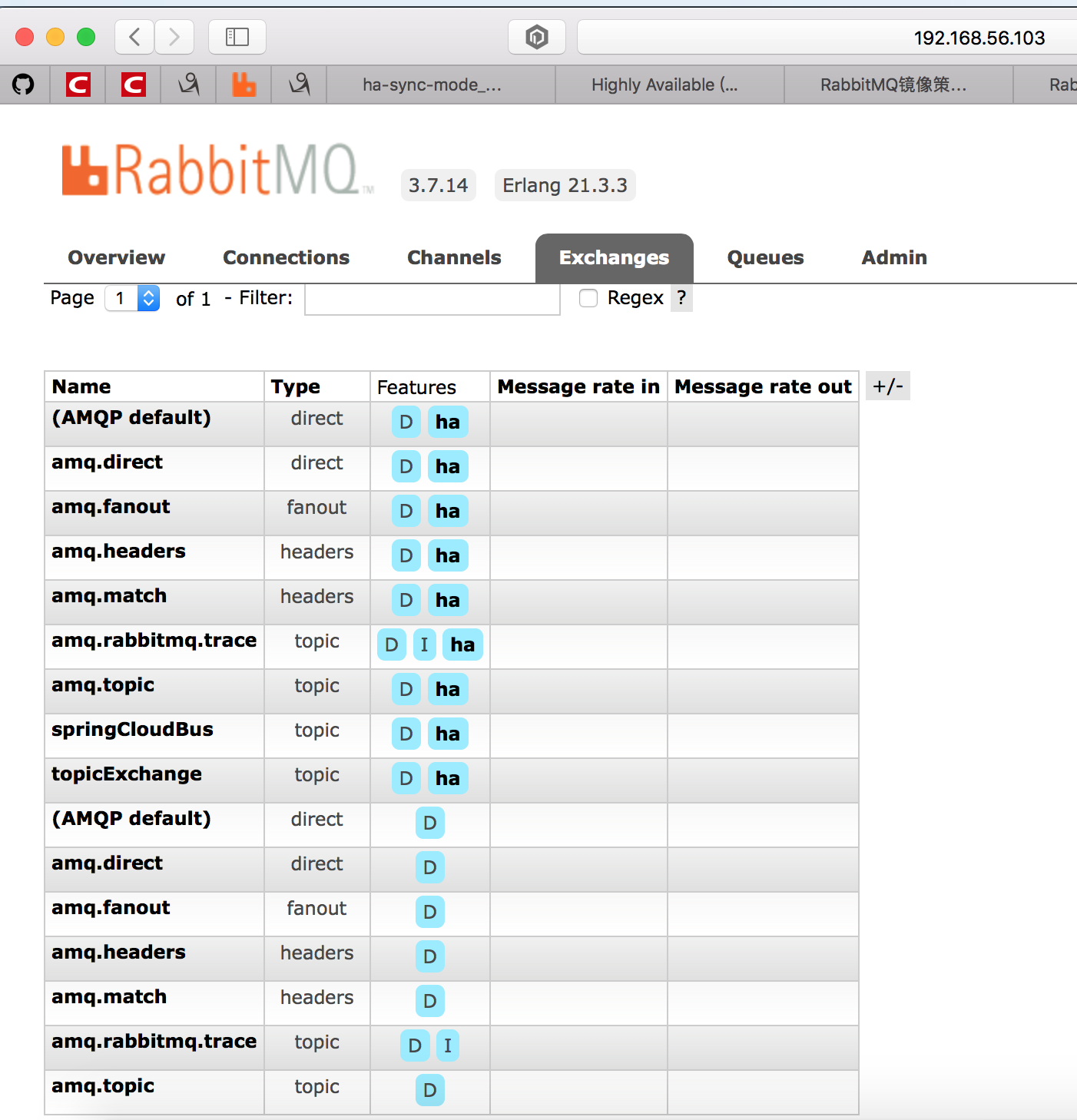
执行命令： rabbitmqctl set\_policy -p / ha "^" '{"ha-mode":"all","ha-sync-mode":"automatic"}'

在cluster中任意节点启用策略，策略会自动同步到集群节点   
  
这行命令在vhost名称为 / 创建了一个策略，策略名称为ha ,策略模式为 all 即复制到所有节点，包含新增节点，策略正则表达式为 “^” 表示所有匹配所有队列名称, HA同步模式w为automatic（自动模式）  
例如rabbitmqctl set\_policy -p hrsystem ha-allqueue "^message" '{"ha-mode":"all"}'  
注意："^message" 这个规则要根据自己修改，这个是指同步"message"开头的队列名称，我们配置时使用的应用于所有队列，所以表达式为"^"  
官方set\_policy说明参见  
set\_policy [-p vhostpath] {name} {pattern} {definition} [priority]  
（http://www.rabbitmq.com/man/rabbitmqctl.1.man.html）

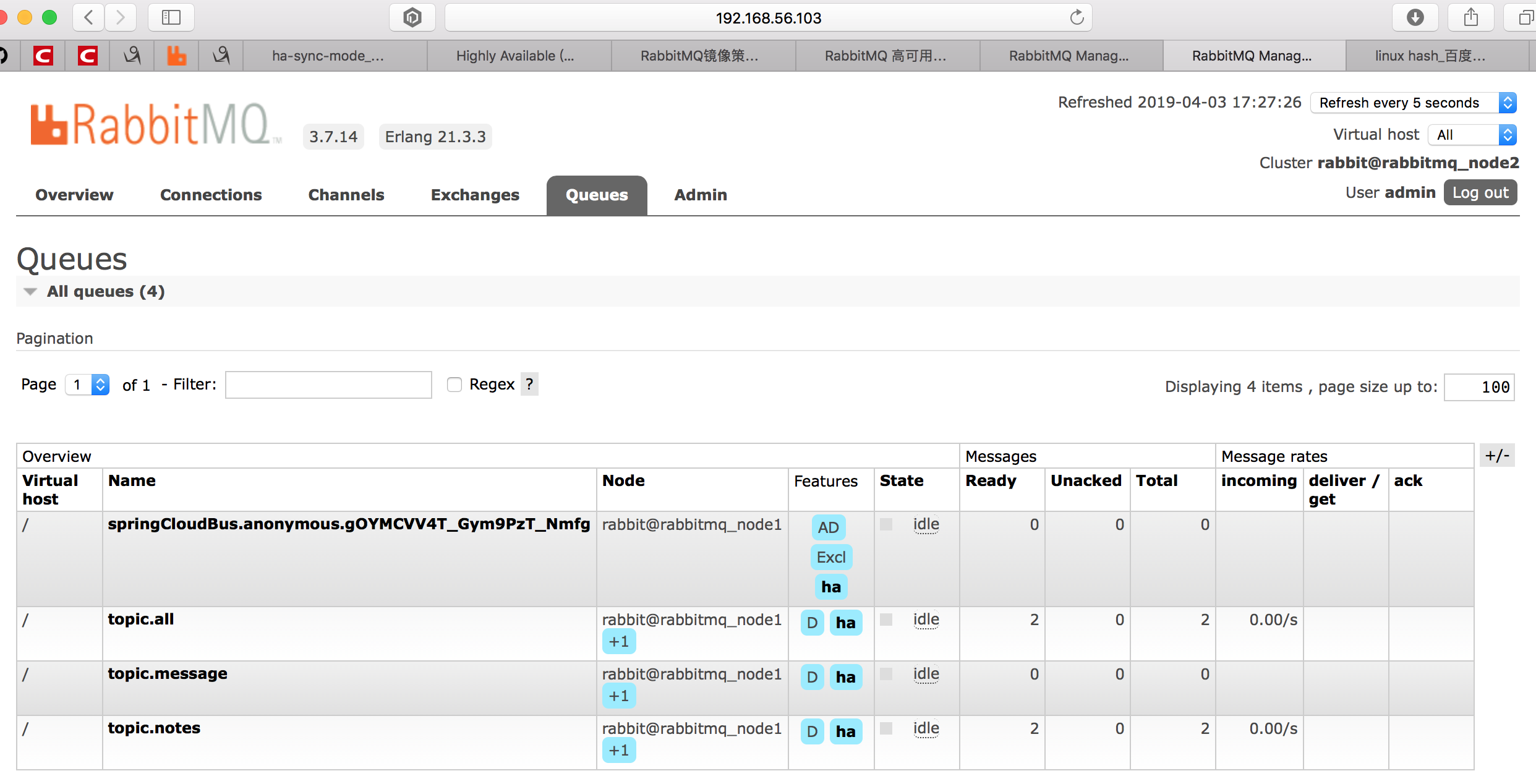
执行完控制台结果：



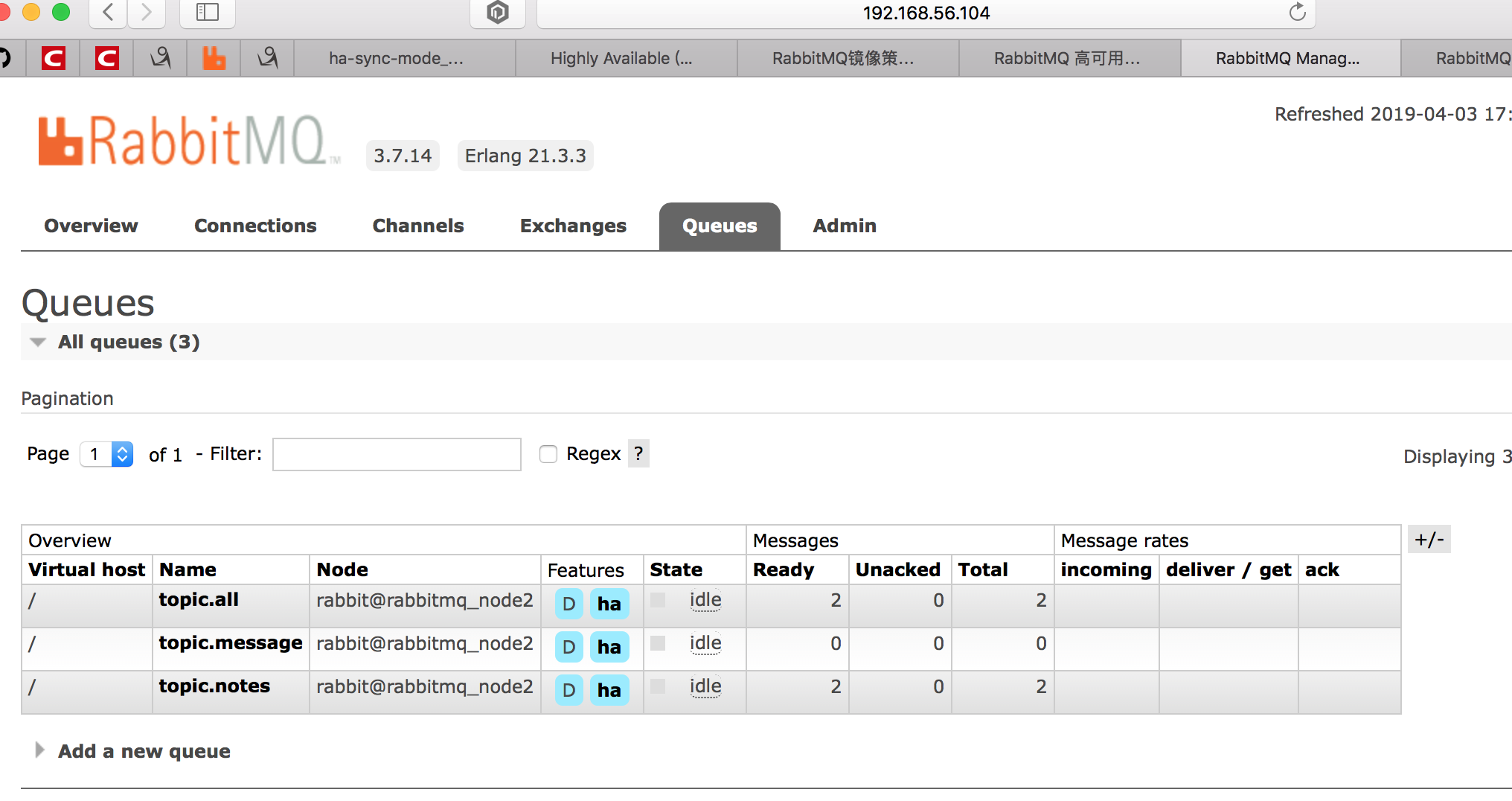




向 rabbitmq\_node1 发送消息

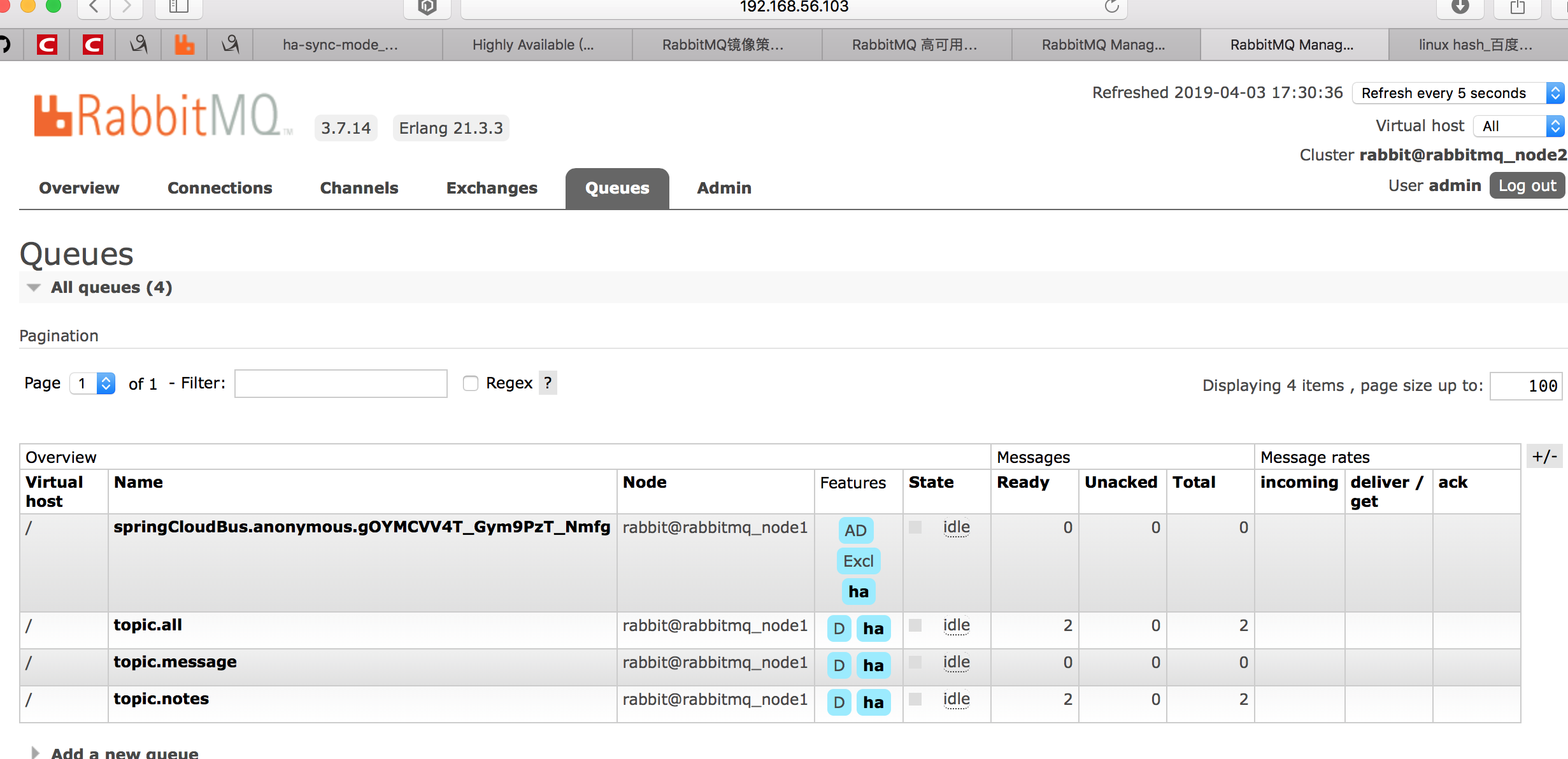


再关闭 node1,查看node2 上的情况



数据在， Nodes 自动切换为 rabbitmq\_node2

关闭node2 查看控制台情况



数据在， Nodes 自动切换为 rabbitmq\_node1

# Haproxy

将安装包上传到 106上

安装 haproxy ： tar -xvf haproxy-1.8.19.tar.gz

cd haproxy-1.8.19

#数说明：

#TARGET=linux3100

#使用uname -r查看内核，如：2.6.18-371.el5，此时该参数就为linux26

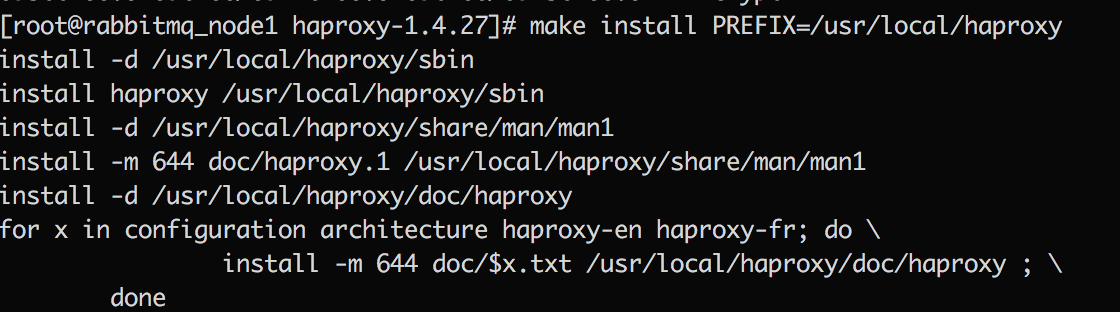
#kernel 大于2.6.28的用：TARGET=linux2628

#CPU=x86\_64 #使用uname -r查看系统信息，如x86\_64 x86\_64 x86\_64 GNU/Linux，此时该参数就为x86\_64

#PREFIX=/usr/local/haprpxy #/usr/local/haprpxy为haprpxy安装路径

根据当前环境命令为 ：make TARGET=linux2628 CPU=x86\_64 PREFIX=/usr/local/haproxy

make install PREFIX=/usr/local/haproxy



haproxy.cfg 文件内容

global

log 127.0.0.1 local0

log 127.0.0.1 local1 notice

daemon

nbproc 2

maxconn 51200

pidfile /usr/local/haproxy/haproxy.pid

defaults

log global

mode tcp

option tcplog

option dontlognull

retries 3

option abortonclose

timeout connect 5000ms

timeout client 30000ms

timeout server 60000ms

balance roundrobin

listen stats

bind 0.0.0.0:9080

mode http

option httplog

maxconn 10

stats refresh 30s

stats uri /stats

stats realm Haproxy Manager

stats auth admin:admin

stats hide-version

stats admin if TRUE

### rabbitmq 集群配置，转发到

listen rabbitmq\_cluster

bind \*:5672

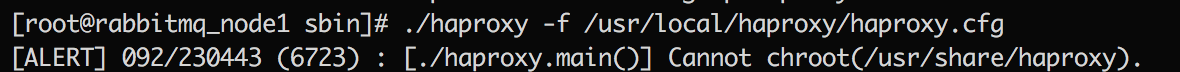
mode tcp

balance roundrobin

server rabbitnode1 192.168.56.103:5672 check inter 2000 rise 2 fall 3 weight 1

server rabbitnode2 192.168.56.104:5672 check inter 2000 rise 2 fall 3 weight 1

5672 端口根据情况配置，如果被占用的话还其他端口



将haproxy.cfg文件放入 /usr/local/haproxy 目录下

修改防火墙：

vi /etc/sysconfig/iptables

添加端口：

-A INPUT -m state --state NEW -m tcp -p tcp --dport 9080 -j ACCEPT

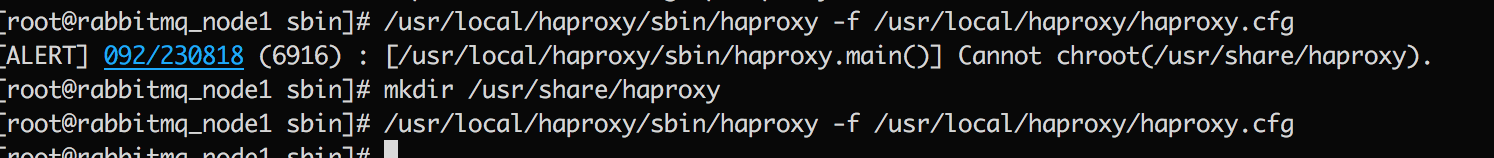
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5673 -j ACCEPT

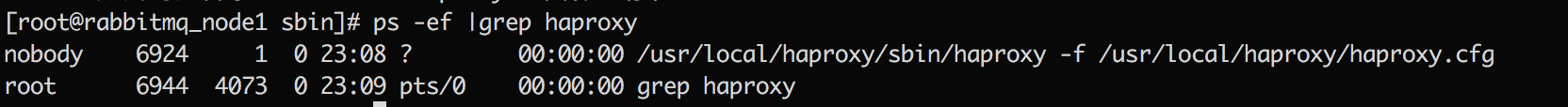
重启防火墙： service iptables restart

启动命令：/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/haproxy.cfg

如果启动时出现报错：/haproxy.main()] Cannot chroot(/usr/share/haproxy)

则手动创建： # mkdir /usr/share/haproxy

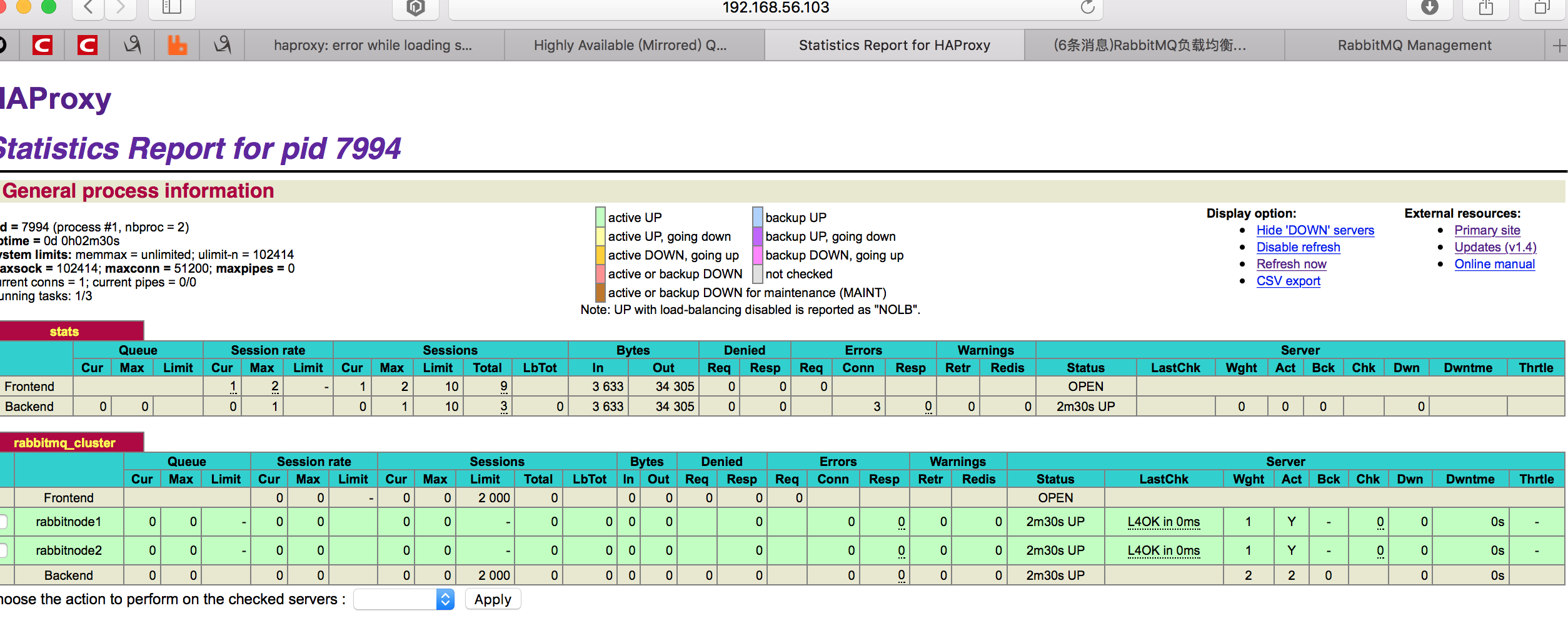




启动成功

访问地址： <http://192.168.56.106:9080/stats>

输入用户名密码 admin ,admin



测试：

使用 haproxy 的ip 和 绑定的端口 连接 rabbitmq ，发送和发送消息 成功

使用 haproxy 的ip 和 绑定的端口 连接 rabbitmq ，发送和消费消息 成功

关闭103 上的rabbitmq 查看生产消费情况：发送和消费成功

关闭104 上的rabbitmq 查看生产消费情况：发送和消费成功

设置 haproxy 开机启动：

将haproxy文件拷贝到/etc/init.d/ 下

chmod +x haproxy

注意：当配置了keepalived后，haproxy的启动和停止交给keepalived

不要再设置开机启动！

# chkconfig haproxy on

<https://www.cnblogs.com/grimm/p/5771038.html>

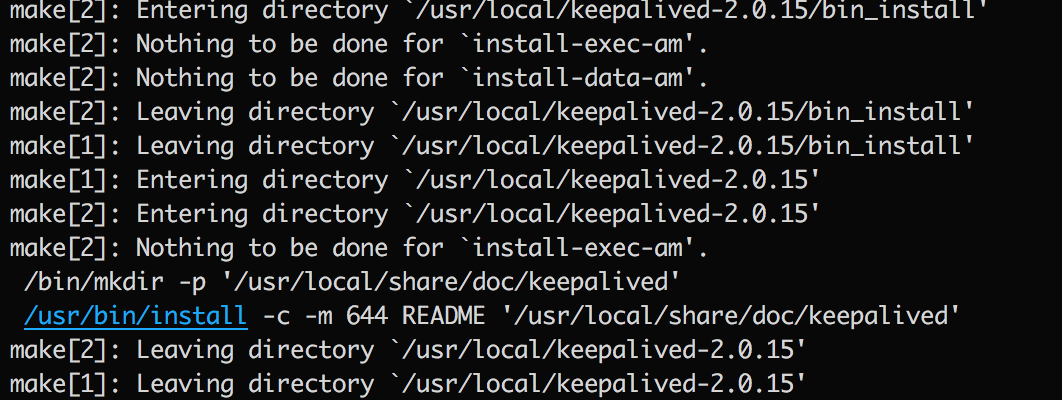
keepalived 安装：

tar -xvf keepalived-2.0.15.tar.gz

cd keepalived-2.0.15

./configure --prefix=/usr/local/keepalived

make && make install

****

cp /usr/local/keepalived-2.0.15/keepalived/etc/init.d/keepalived /etc/init.d

mkdir /etc/keepalived/

cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/

cp /usr/local/keepalived-2.0.15/keepalived/etc/sysconfig/keepalived /etc/sysconfig/keepalived

cp /usr/local/keepalived/sbin/keepalived /usr/sbin/

106（主） 和103 （备）安装haproxy

106 和103 安装keepalived

将主 下面的keepalived.conf 拷贝到106 /etc/keepalived/

将备 下面的keepalived.conf 拷贝到103 /etc/keepalived/

<https://blog.csdn.net/lvshaorong/article/details/81301976>

将脚本 check\_haproxy.sh ,to\_backup.sh ,to\_fault.sh, to\_master.sh 放到103和106 的 /etc/keepalived/script/ 下面。

脚本说明：

Keepalived组之间的心跳检查并不能察觉到HAproxy负载是否正常，所以需要使用此脚本。

在Keepalived主机上，开启此脚本检测HAproxy是否正常工作，如正常工作，记录日志。

如进程不存在，则尝试重启HAproxy，两秒后检测，如果还没有则关掉主Keepalived，此时备Keepalived检测到主Keepalive挂掉，接管VIP，继续服务

添加可执行权限：

chmod +x /etc/keepalived/script/\*.sh

启动命令：service keepalived start

设置开机启动：

chkconfig keepalived on

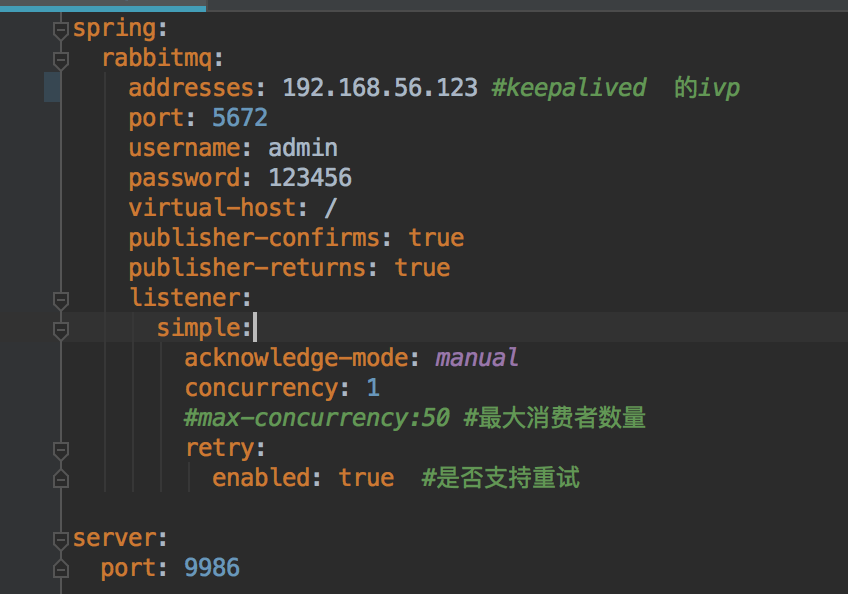
service keepalived status

出现 *keepalived dead but subsys locked*

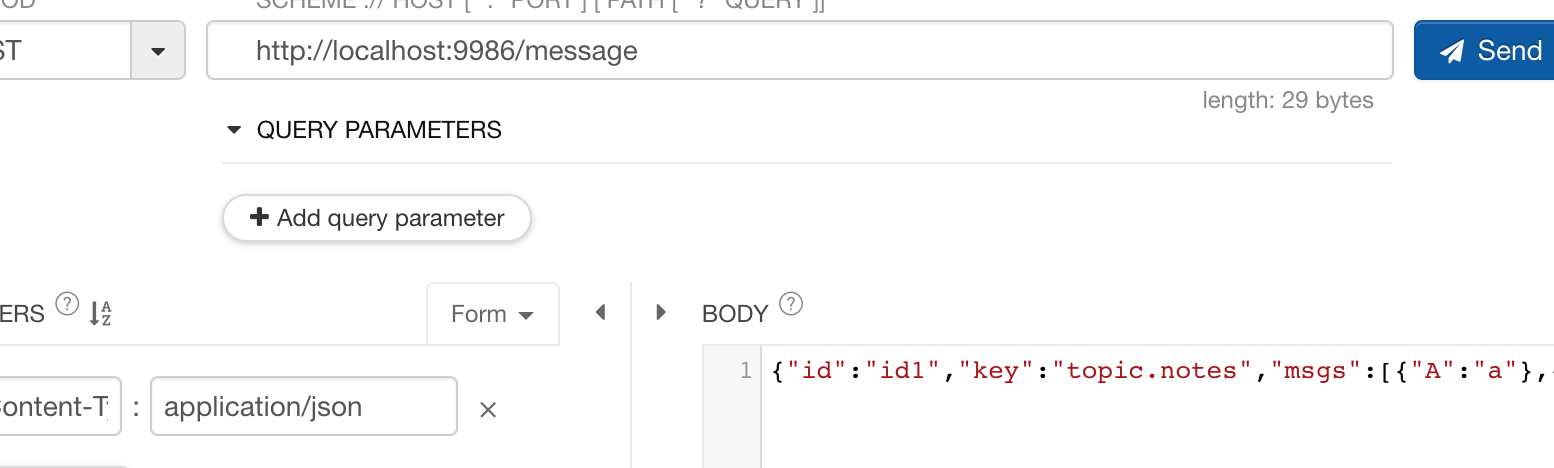
查看keepalived.conf 文件中的 vrrp\_instance VI\_1 interface 配置

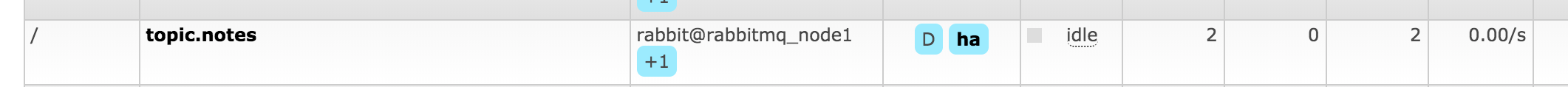
测试：

使用 vip 连接rabbitmq 集群：

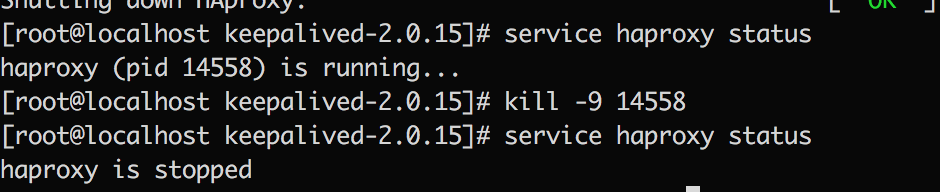


测试发送数据，发送成功

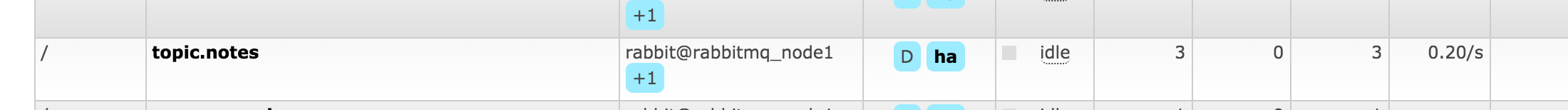




关闭 106（主）上的haproxy ，查看消息发送情况



发送成功：

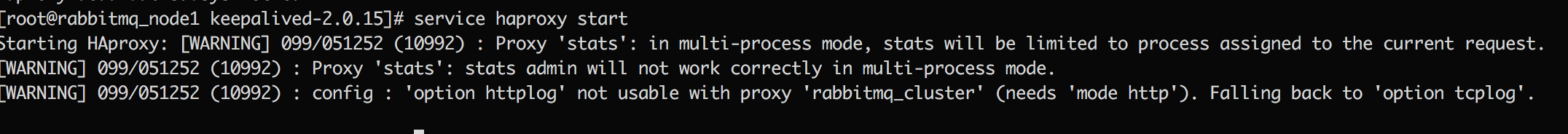


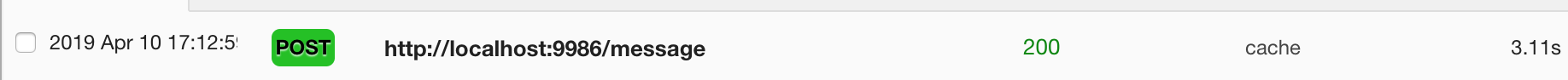
关闭所有haproxy， 测试发送

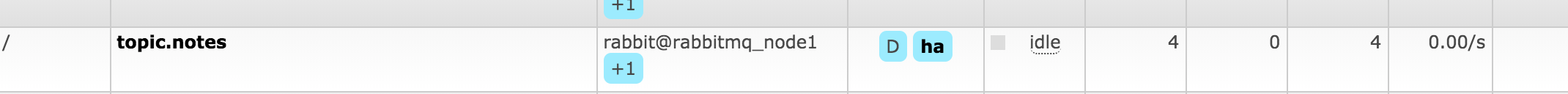


请求 报500错

重新启动 103 上的haproxy ，查看发送情况



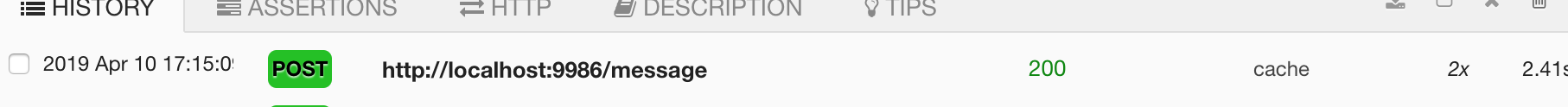


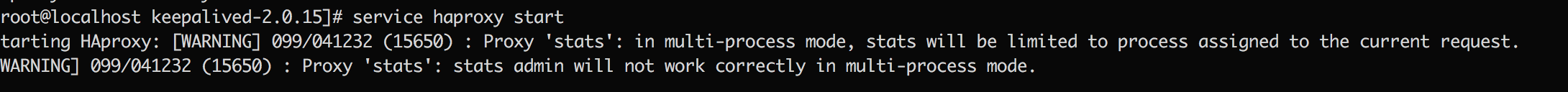


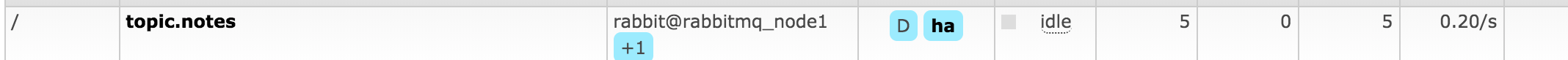
发送成功。

将106 上的haproxy也启动，查看发送情况

发送成功：







**RABBITMQ集群重启**

集群重启时，最后一个挂掉的节点应该第一个重启，如果因特殊原因（比如同时断电），而不知道哪个节点最后一个挂掉。可用以下方法重启：

先在一个节点上执行

#rabbitmqctl force\_boot

#service rabbitmq-server start

在其他节点上执行

#service rabbitmq-server start

查看cluster状态是否正常（要在所有节点上查询）。

#rabbitmqctl cluster\_status

如果有节点没加入集群，可以先退出集群，然后再重新加入集群。

上述方法不适合内存节点重启，内存节点重启的时候是会去磁盘节点同步数据，如果磁盘节点没起来，内存节点一直失败。