**redis4.0.x集群搭建**

## 一.集群搭建前提环境-ruby

因为redis集群环境基于ruby，所以需要先安装ruby运行环境。

采用rvm来安装或者升级ruby，因为有些服务器已经有了ruby，比如我买的阿里云好几台都是ruby2.0，你在安装redis集群时，执行gem install redis 安装接口命令就会报错：

[**redis需要Ruby版本> = 2.2.2**](https://blog.csdn.net/xinshui151/article/details/79216406)

告诉你最起码需要2.2.2的版本，因为现在redis4.0.x比较多，2.0版本低了，redis3.0貌似不会有这个问题。

---------------------------------------------------

### 1.安装RVM：（以下每一行都是一条命令）

gpg2 --keyserver hkp://keys.gnupg.net --recv-keys D39DC0E3

curl -L get.rvm.io | bash -s stable

网速低的话装起来稍微慢点，装完后执行find进行查找，找到标红的rvm。

find / -name rvm -print

/ usr / local / rvm

/ usr / local / rvm / src / rvm

/usr/local/rvm/src/rvm/bin/rvm

/usr/local/rvm/src/rvm/lib/rvm

/usr/local/rvm / src / rvm / scripts / rvm

/ usr / local / rvm / bin / rvm

/ usr / local / rvm / lib / rvm

/ usr / local / rvm / scripts / rvm

然后执行source命令：

source /usr/local/rvm/scripts/rvm

### 2.查看RVM库中已知的ruby版本

rvm list known

[ruby-] 1.8.6 [-p420]

[ruby-] 1.8.7 [-head]＃在头上发布的安全性

[ruby-] 1.9.1 [-p431]

[ruby-] 1.9.2 [-p330 ]

[ruby-] 1.9.3 [-p551]

[ruby-] 2.0.0 [-p648]

[ruby-] 2.1 [.10]

[ruby-] 2.2 [.6]

[ruby-] 2.3 [.3]

[ruby-] 2.4 [.0]

### 3. 安装一个版本

rvm install 2.3.3

### 4.安装完成后，使用这个版本并设置默认版本。

rvm use 2.3.3 --default

### 5.卸载目前的2.0版本（没有就不用管）

rvm remove 2.0.0

### 6. 查看是不是安装的版本

ruby --version或ruby -v

### 7.然后安装redis接口

gem install redis

获取：redis-4.0.0.gem（100％）

.....

....

成功安装redis-4.0.0 redis-4.0.0什么什么的....

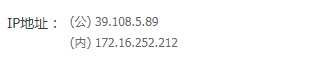
## 二.搭建redis集群

redis集群有投票形式的容错机制，所以要超过半数投票来确定某个节点是否挂掉了，辣么最少就需要3个节点（2个节点投什么票对吧？），而一般一个主节点需要有从节点，达到高可用的目的，辣么redis最少就需要6个节点来集群。

假设有6台服务器搭建redis集群，但资源欠缺，就在一台服务器上创建6个redis实例来搭建一套伪集群，真实环境操作反而比伪集群更简单，所以会搭建伪集群就没有问题。

假设搭建的6台端口分别为：7001,7002,7003,7004,7005,7006。

以我的阿里云服务器为例：



### 1. 将之前跑的redis单例停掉

redis-cli 命令进入，然后shutdown save 关闭。

ps -ef|grep redis 验证是否已关闭。

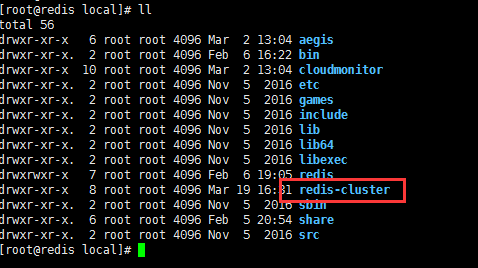
### 2.创建redis-cluster目录作为集群目录

找到之前安装redis的目录，我的是/usr/local，创建redis-cluster目录，将之前装过的redis复制一份到redis-cluster目录中去并重命名为redis01。

cd /usr/local

mkdir redis-cluster

cp redis redis-cluster/redis01



### 3.编辑配置文件并复制出其它5个节点

进入redis01，编辑redis.conf

vim redis01/redis.conf

修改以下一些参数：

# 端口号，redis01的改为7001.

**port 7001**

# 绑定IP，这是个大坑很容易误解，这里不是指绑定能访问redis服务器的外部服务器IP，而是指绑定本机可以接受访问的IP。我的阿里云ifconfig查看的IP就是172.16.252.212.用阿里云的公网IP绑定后启动都不行，原因目前未知。配置成0.0.0.0也是可以的，而且可以用jedis通过外网访问，前提是后面出现的问题都解决了，再回头改成0.0.0.0试试。

**bind 172.16.252.212**

# 密码也是个大坑，不设置还好，设置了对后面创建集群有很大影响，有的人不设置，但我认为真实环境为了安全考虑应该要设置的。

**requirepass 123456**

# 是否开启后台运行，是。

**daemonize yes**

# 存放redis数据的目录，比如快照、conf文件、aof日志等等，这个data目录如果没有自己创建，路径对应上就行。

**dir /usr/local/redis-cluster/redis01/data**

# 后台进程运行需要指定一个pid，虽然有默认的，但集群我们自己指定一个，自己命名。

**pidfile /var/run/redis\_7001.pid**

# 日志文件路径

**logfile /usr/local/redis-cluster/redis01/redis.log**

# 开启集群

**cluster-enabled yes**

# 集群的配置，配置文件首次启动自动生成，会放在上面指定的data目录里，自行验证。

**cluster-config-file nodes\_7001.conf**

# 请求超时，默认15秒，可自行设置，我设置为10秒。

**cluster-node-timeout 10100**

# aof日志开启，有需要就开启，它会每次写操作都记录一条日志，稍微耗性能一点，但实用性比快照要好。

**appendonly yes**

上述配置完成后记得保存，然后复制5份到同目录，分别为redis02、redis03、redis04、redis05、redis06。

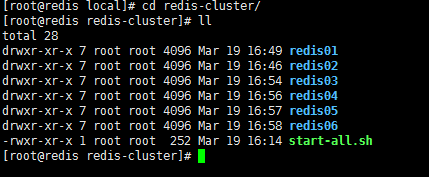
cp -r redis01 redis02

cp -r redis01 redis03

cp -r redis01 redis04

cp -r redis01 redis05

cp -r redis01 redis06



最后，分别修改redis02、redis03、redis04、redis05、redis06中的redis.conf文件，参照第3点中的配置，把port、dir、pidfile、logfile、cluster-config-file的名称路径什么的都改成和节点对应的。

### 4.编写shell启动和关闭脚本

1. 由于有6个节点，一个个启动比较麻烦，我们编写一个shell直接全部启动。

---------------------------------------------------------

vim start-all.sh 创建出脚本并命名为start-all.sh

shell启动脚本如下:

**cd redis01**

**redis-server ./redis.conf**

**cd ..**

**cd redis02**

**redis-server ./redis.conf**

**cd ..**

**cd redis03**

**redis-server ./redis.conf**

**cd ..**

**cd redis04**

**redis-server ./redis.conf**

**cd ..**

**cd redis05**

**redis-server ./redis.conf**

**cd ..**

**cd redis06**

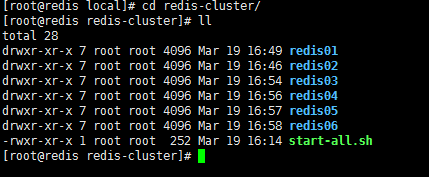
**redis-server ./redis.conf**

---------------------------------------------------------

这里有一点注意：如果是redis3.0.x版本的写法有所不同，主要是redis.conf的位置和4.0.x有区别，目录结构3.0.x和4.0.x有些不同。

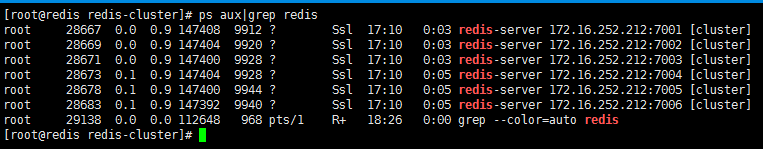
编写完shell脚本后保存退出，然后赋予权限,因为目前还不是可执行文件。变成绿色的就表示赋予权限成功，变成了可执行文件。

赋予权限：chmod +x start-all.sh



执行启动脚本：./start-all.sh

验证是否启动：ps aux|grep redis



如图，6个节点都启动了，并且是cluster模式，表示集群模式。

PS: 这里说明两个个问题

1.redis.conf有个bind配置，用自己的虚拟机来模拟的话配置成本机IP是没有问题的，但用阿里云服务器就会有问题，因为阿里云服务器分为公网IP和内网IP，bind公网IP会出现起不起来redis的现象，查看日志会报“**can not assign requested address**”这样的提示，表示不能分配请求地址，原因我不清楚，可能是阿里云的网络有什么限制。但换成内网IP后就可以起起来了，内网IP就是本机IP，用ifconfig可以查看。

2.bind配成0.0.0.0不一定能跑起来，但文档里出现的问题都解决了的话，配置成0.0.0.0是没问题的，用jedis是可以访问，就是在本机用jedis通过外网访问redis服务器。这个可以在搭建好了之后放在最后测试。

1. 同理，由于6个节点，一个个关闭也比较麻烦，我们编写一个shell直接全部关闭。

vim shutdown-all.sh 创建关闭脚本，并命名为shutdown-all.sh

-----------------------------------------------------------------------------------

shell脚本如下：

**cd redis01**

**redis-cli -p 7001 -a '123456' shutdown**

**cd ..**

**cd redis02**

**redis-cli -p 7002 -a '123456' shutdown**

**cd ..**

**cd redis03**

**redis-cli -p 7003 -a '123456' shutdown**

**cd ..**

**cd redis04**

**redis-cli -p 7004 -a '123456' shutdown**

**cd ..**

**cd redis05**

**redis-cli -p 7005 -a '123456' shutdown**

**cd ..**

**cd redis06**

**redis-cli -p 7006 -a '123456' shutdown**

**cd ..**

**创建完成后保存并退出，然后赋予权限，让其变为可执行文件。**

**chmod +x** shutdown-all.sh

然后执行关闭脚本: ./shutdown-all.sh

验证是否全部关闭：ps aux|grep redis

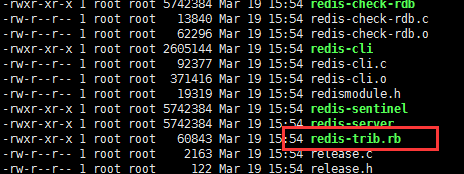
PS: 这里关闭也有一个坑，就是因为密码的原因，如果没有设置requirepass，shell脚本中就不需要加上 -a '123456' ，如果设置了密码，就必须加上这一句，否则一直关闭不成功。

### 5.创建集群，让6个节点都连接起来。

需要用到redis集群提供的一个工具来执行集群创建，让所有节点都连为一体。

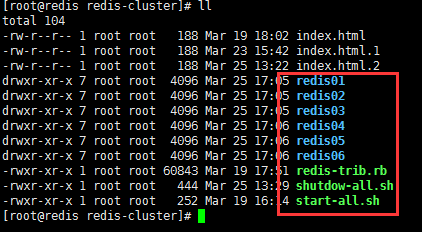
工具：redis-trib.rb

这个工具在redis目录中：redis01/src/ 下面，每个节点里面都有。



我们只需要随便进一个节点目录，将其拷出来就可以了，拷贝到redis-cluster目录下。

最后的redis集群目录如下：



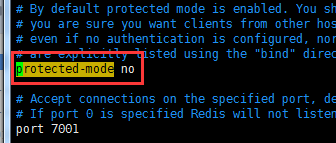
使用redis-trib.rb 创建集群，命令如下，很长但是理解很简单：

./redis-trib.rb create --replicas 1 172.16.252.212:7001 172.16.252.212:7002 172.16.252.212:7003 172.16.252.212:7004 172.16.252.212:7005 172.16.252.212:7006

执行之后，重头戏来了，这里不顺利的话会出现的问题非常多，你用自己虚拟机不会遇到，用接近真实环境的服务器都可能出现，这个问题就是能让人一直折腾的“ **cannot connected to node 172.16.252.212:7001** ”。

排查方向如下：

1. wget ip:port 验证下是不是connected；
2. 如果用的阿里云服务器或其他服务器，记得去管理平台的防火墙那里看看端口是不是都加上了，没加上的话必须加上，就算你在Linux中设置过也没用，这是云服务器自己的策略；
3. vim redis01/redis.conf，找到protected-mode，看看保护模式是什么，设为no然后保存退出，重启后，再执行命令试试看。



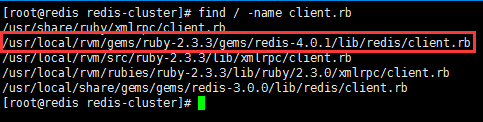
这里的protected-mode是保护模式，最明显的，你用redis-desktop-manager这个客户端工具连接redis服务器试试，如果这个设为yes，你怎么也连不上，设为no就可以连上。或者用本机cmd窗口，Telnet redis服务器IP端口，会报“DENIED Redis **is** running in protected modebecause protected mode **is** enabled”这样的错误。

1. 经各种折腾，问题主要还是出在requirepass这个配置上，因为设置了密码，直接使用redis-trib.rb工具执行命令，通过不了密码认证，所以怎么都**cannot connected node。**

把requirepass给注释掉，你就发现可以创建集群了，但问题是密码有时也是必要的，为了安全性，那该怎么办呢。我百度查找了快一天都没找到解决方法，最后在一个外网上查到了，后来在查找其他问题的时候在中国网站也找到了，心态崩了。

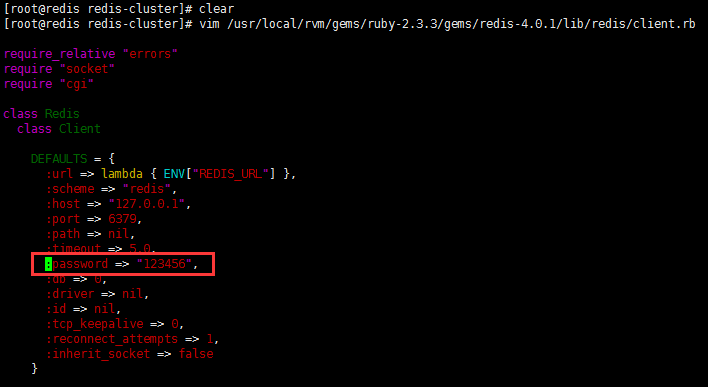
有一个工具叫client.rb，这里面有个密码的配置，对应的是redis.conf中requirepass的配置，如果没有做修改，怎么都无法通过认证，因为redis-trib.rb命令在你设置了requirepass之后很多命令都要通过密码认证才能执行了，比如redis-trib.rb check 172.16.252.212，设了密码而没改client.rb的配置执行这个命令就会报上面同样的错。

find / -name client.rb 找到这个东西在哪儿，如下图，因为我们的ruby是2.3.3的，redis是4.0的gem，所以就是红框这个，看清楚别找错了，是redis的lib下面的。



vim /usr/local/rvm/gems/ruby-2.3.3/gems/redis-4.0.1/lib/redis/client.rb，编辑这个工具。

如下图，有一个password，原本是nil，没有配置的，我们改成和requirepass一样的密码123456.



保存并退出。

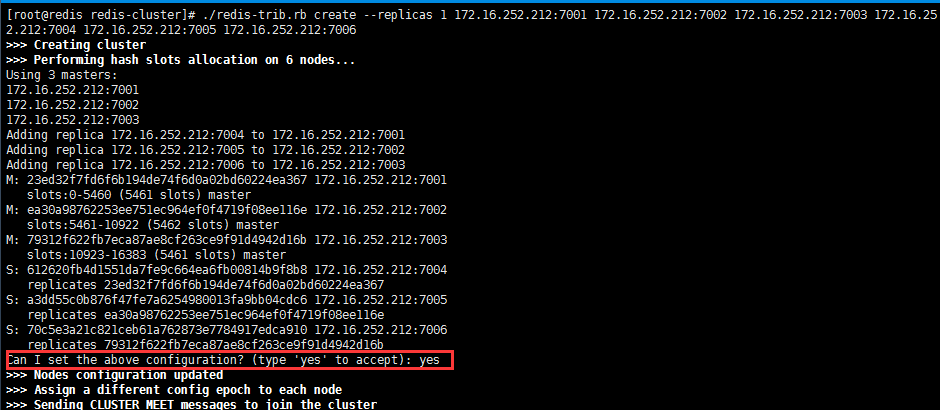
然后我们执行shell脚本全部关闭，然后全部开启，再次执行集群创建命令：

./redis-trib.rb create --replicas 1 172.16.252.212:7001 172.16.252.212:7002 172.16.252.212:7003 172.16.252.212:7004 172.16.252.212:7005 172.16.252.212:7006

神奇的事发生了，创建成功了，开始往下执行了！

### 6.创建集群的执行过程中会出现的问题

在第5点基础上，执行命令后会有一个执行过程一直往下进行，然后一直会出现一个提示，如下图：选yes就可以了，因为没得选。

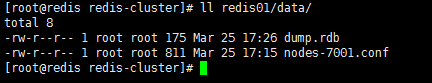


选完yes之后，会继续往下进行，但又有可能会报一些错误。

1. 报XXXX is not empty的错误



这个意思是你的redis节点中，快照、配置、日志等数据文件没有清除掉，进入redis01的data目录，也就是我们一开始指定的数据存放目录，会发现这些文件。

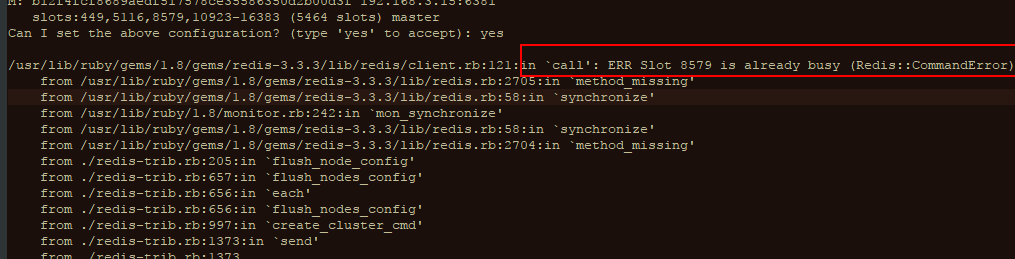


将这些都删掉，然后redis02、03、04、05、06这些节点下的也都删掉。

然后再次执行命令创建集群即可。

1. 延续1)中的过程，如果执行后又出现问题，如下：

**in `call': ERR Slot 8579 is already busy (Redis::CommandError)**

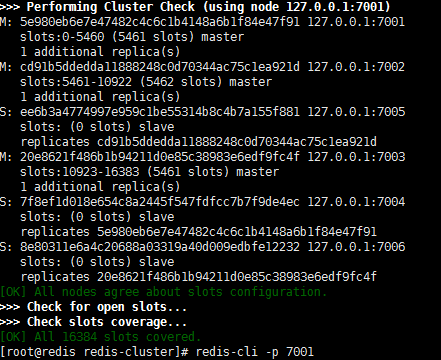


如上图，这个问题是因为slot插槽被占用了（这是 搭建集群前时，以前redis的旧数据和配置信息没有清理干净。）

redis-cli 命令进入每个redis节点，执行flushall  和 cluster reset 命令就可以了。(正式环境不可能用这些命令的，因为不安全，一般都禁用的，集群成功后也不可能动不动就清除和重置什么的。)

然后再执行集群创建命令，有提示就输入yes。

普天同庆的看到了OK！！集群创建成功！6个节点都连接起来了！而且7001,7002,7003显示的是master表示主节点，7004,7005,7006显示的是slave表示从节点！



这里如图显示的从节点的slots是0，表示没有插槽，可以使用redis-trib.rb reshard 命令什么的去分配槽，具体大家可以去百度，redis的东西还蛮多的。

### 7.验证集群

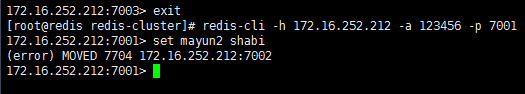
1. 内网验证，因为之前redis-trib.rb create 创建的集群是内网IP: 172.16.252.212，所以先来验证内网的。

执行命令：

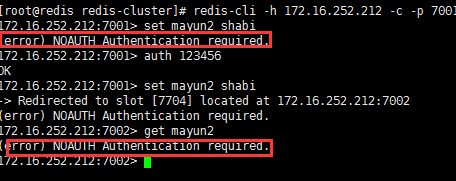
redis-cli -h 172.16.252.212 -c -a 123456 -p 7001

连上7001这个节点，必须要加 -c 和 -a 参数，-c表示是集群，-a表示是密码认证。

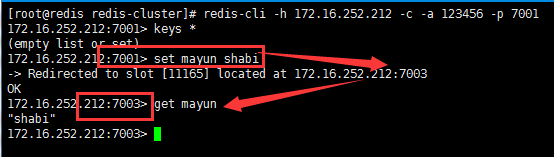
不加-c，插入值会报错：



不加-a，插入值也会报错：它会一直提示你密码认证。



都加上后，set一个值，就会出现redirected to XXX的提示，表示重定向set到了7003这个节点，并且会自动转到7003这个节点下，直接get，发现能拿到值。

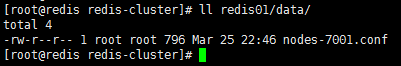


1. 外网验证

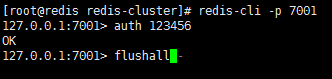
首先，把每个节点的bind这个配置改成0.0.0.0。



然后，把每个节点的data目录下的快照、日志、node\_conf这些只要有的统统都删掉。



最后，redis-cli进入每个节点，执行flushall和 cluster reset命令，都清除干净。





清理干净后，使用shell脚本关闭所有节点并重启。





然后再次执行创建集群命令：（这次用的IP就是阿里云的公网IP：39.108.5.89了，外网都能ping通的。）

./redis-trib.rb create --replicas 1 39.108.5.89:7001 39.108.5.89:7002 39.108.5.89:7003 39.108.5.89:7004 39.108.5.89:7005 39.108.5.89:7006



如果上述问题都解决过了，这里执行肯定可以成功的，如果有问题请参考第5和6点。

1. Jedis验证集群

在开发工具中写一段Junit的测试代码：

--------------------------------------------------------------------------------------------------

@Test

**public** **void** testJedisCluster() **throws** Exception {

//创建一连接，JedisCluster对象,在系统中是单例存在

Set<HostAndPort> nodes = **new** HashSet<>();

nodes.add(**new** HostAndPort("39.108.5.89", 7001));

nodes.add(**new** HostAndPort("39.108.5.89", 7002));

nodes.add(**new** HostAndPort("39.108.5.89", 7003));

nodes.add(**new** HostAndPort("39.108.5.89", 7004));

nodes.add(**new** HostAndPort("39.108.5.89", 7005));

nodes.add(**new** HostAndPort("39.108.5.89", 7006));

JedisCluster cluster = **new** JedisCluster(nodes, 10100, 3000, 5, "123456", **new** GenericObjectPoolConfig());

//执行JedisCluster对象中的方法

cluster.set("cluster-test", "my jedis cluster test");

String result = cluster.get("cluster-test");

System.***out***.println(result);

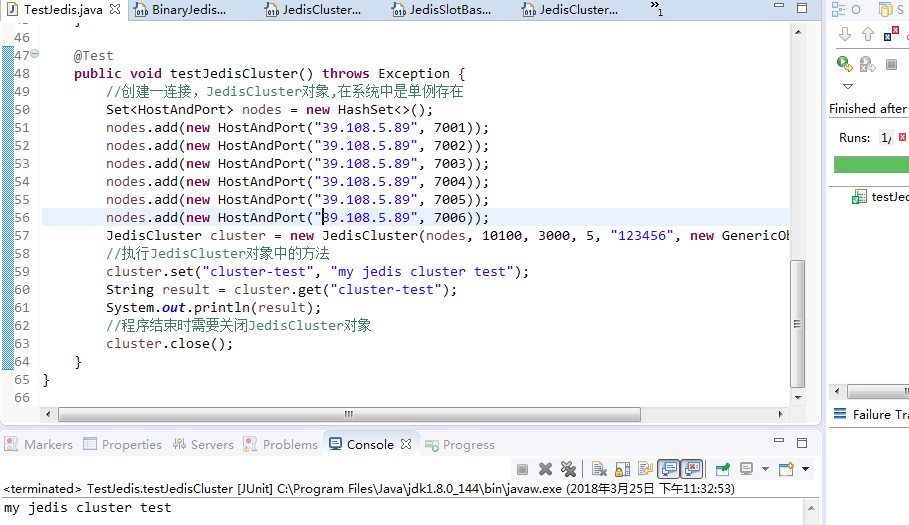
//程序结束时需要关闭JedisCluster对象

cluster.close();

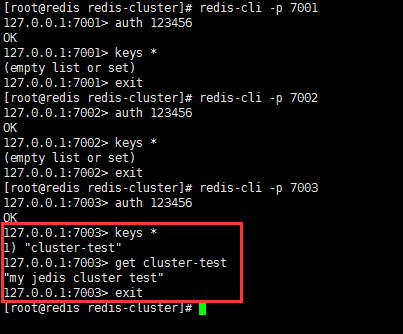
}

--------------------------------------------------------------------------------------------------

执行，成功！



同时进入redis查看是否set进来了，找一下，发现set进了7003这个节点。



-----------------------------------------------------------------------------------------------------------------------

至此，redis4.0.x的伪集群搭建就完成了，用真实服务器搭建要比想象中的更折腾一些，希望我遇到的这些问题能给诸位带来帮助，减少大家花费的时间。