**CentOS6.4上面Redis3.0的安装**

# 环境检查

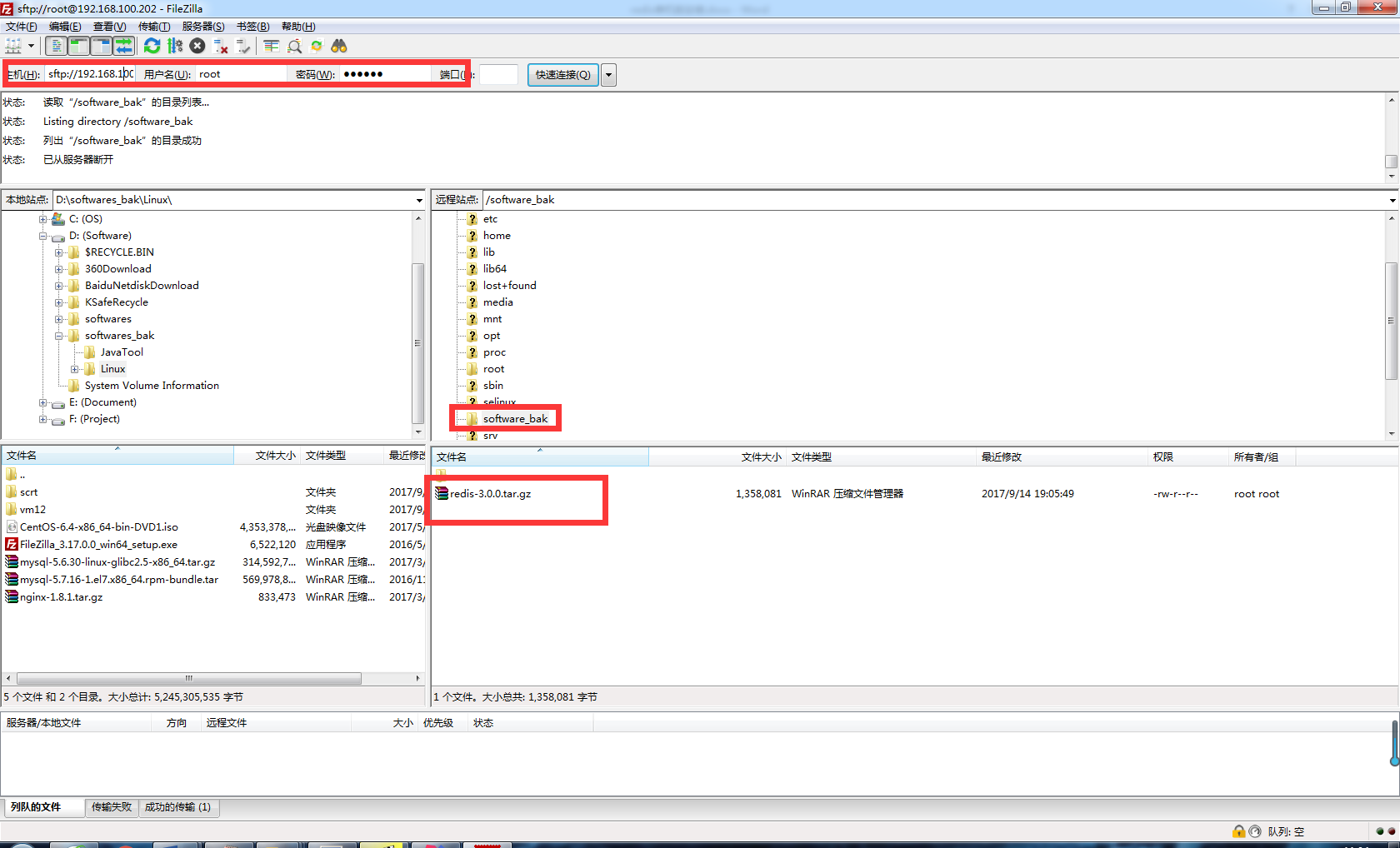
|  |
| --- |
| [root@root bin]# cat /etc/issue  CentOS release 6.4 (Final)  Kernel \r on an \m |

# 软件准备

|  |
| --- |
| redis-3.0.0.tar.gz |

# 上传到服务器

使用filezilla上传到服务器。



# 安装

要求：将Redis安装到/usr/local/redis目录下面去。

## 安装gcc编译环境

由于Redis是由ANSI C语言编写，故需要c语言的编译环境。

|  |
| --- |
| // 安装编译环境  [root@root software\_bak]# yum install gcc-c++  // 上传后解压  [root@root software\_bak]# tar -zxvf redis-3.0.0.tar.gz |

## 编译源码

|  |
| --- |
| // 进入解压目录  [root@root software\_bak]# cd redis-3.0.0  // 编译源码  [root@root redis-3.0.0]# make |

注意：执行make命令的时候可能会报“找不到make命令”的错误，如果有这种错误出现，需要安装yum install gcc automake autoconf libtool make，进行安装make工具包

## 安装

|  |
| --- |
| // 指定安装目录为/usr/local/redis  [root@root redis-3.0.0]# make install PREFIX=/usr/local/redis |

安装目录就是/usr/local/redis。

## 启动

### 前端启动模式

进入安装目录，然后启动即可。

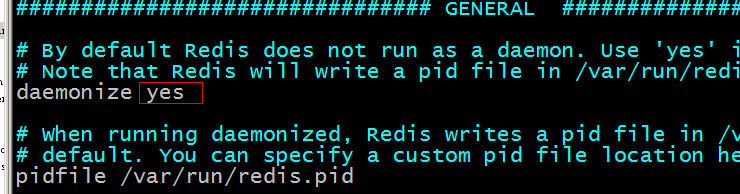
|  |
| --- |
| [root@root bin]# ./redis-server |

### 后端启动模式

后端启动模式需要指定配置文件，所以，需要将解压后的源码包中的配置文件redis.conf文件拷贝到安装目录下面的bin目录下面，然后通过指定配置文件方式启动。

|  |
| --- |
| // 拷贝源码包中的配置文件到安装目录下面的bin目录下  [root@root redis-3.0.0]# cp redis.conf /usr/local/redis/bin/ |

修改配置文件中的相关选项



保存后启动。

|  |
| --- |
| [root@root bin]# ./redis-server redis.conf |

## 测试

|  |
| --- |
| [root@root bin]# ./redis-cli  127.0.0.1:6379> ping  PONG  127.0.0.1:6379> set a 10  OK  127.0.0.1:6379> get a  "10"  127.0.0.1:6379> incr a  (integer) 11  127.0.0.1:6379> get a  "11"  127.0.0.1:6379> decr a  (integer) 10  127.0.0.1:6379> keys \*  1) "a"  127.0.0.1:6379> del a  (integer) 1  127.0.0.1:6379> keys \*  (empty list or set)  127.0.0.1:6379> exit // 关闭客户端 |

## 关闭

|  |
| --- |
| // 第一种：在客户端中直接关闭  127.0.0.1:6379> shutdown // 关闭服务器即可  // 第二种：使用客户端命令直接关闭  [root@root bin]# ./redis-cli –p 6379 |

# 集群环境的搭建

## 集群理论

创建Redis集群，每个主节点都有一个从节点进行备份，让主节点发生错误时，从节点自动连接，实现集群的高可用。

### 集群架构



redis-cluster把所有的物理节点映射到[0-16383]slot上,cluster 负责维护node<->slot<->value的映射。

Redis 集群中内置了 16384 个哈希槽，当需要在 Redis 集群中放置一个 key-value 时，redis 先对 key 使用 crc16 算法算出一个结果，然后把结果对 16384（2的14次方） 求余数，这样每个 key 都会对应一个编号在 0-16383 之间的哈希槽，redis 会根据节点数量大致均等的将哈希槽映射到不同的节点。

此外，由于集群环境中只有2的14次方（16384）个slot（哈希槽），所以，集群环境中最多只能有16384台主机，此时，每个主机上面只有一个slot。

### 集群容错机制



(1)master A在连接master B时，发现master B没有响应，则master A就会向集群中广播，然后开始投票。领着投票过程是集群中所有master参与，如果半数以上master节点与master B节点通信出现(cluster-node-timeout)错误，则认为master B节点挂掉。集群中节点之间的通信是通过ping-pong协议通信的。

(2)什么时候整个集群不可用(cluster\_state:fail)?

a:如果集群任意master挂掉,且当前master没有slave。集群进入fail状态，也可以理解成集群的slot映射[0-16383]不完成时进入fail状态。ps : redis-3.0.0.rc1加入cluster-require-full-coverage参数，默认关闭，打开集群兼容部分失败。

b:如果集群超过半数以上master挂掉，无论是否有slave集群进入fail状态。ps:当集群不可用时，所有对集群的操作做都不可用，收到((error) CLUSTERDOWN The cluster is down)错误。

## 伪集群环境搭建

由于使用的是window环境下虚拟机的运行环境，如果要进行测试集群环境的话，至少需要六台虚拟机，故采用伪集群式进行搭建。这样的话，就只使用一台虚拟机，然后建立六台Redis实例即可。

为什么至少六个实例？因为集群环境中进行容错机制测试时，需要半数投票通过，所以至少需要三个实例，而每一个实例又有一个备份实例，所以，至少需要有六个实例。

我们要求搭建一个伪集群环境，其中有六个Redis实例，占用的端口是7001~7006。

### 环境准备

在创建完实例时，由于需要使用/redis-3.0.0/src/redis-trib.rb脚本进行创建集群，而/redis-3.0.0/src/redis-trib.rb脚本的运行，又需要使用一个叫做redis-3.0.0.gem的ruby包，所以需要安装Ruby的运行环境（此过程可以类比Java，一个工具包的运行依赖另外一个工具包，而另外一个工具包需要有Java的运行环境）。

此过程也可放到六个实例启动之后、创建集群之前进行。

|  |
| --- |
| // 安装ruby运行环境  [root@study src]# yum install ruby  // 安装ruby包的运行环境  [root@study ~]# yum install rubygems  // 安装redis-trib.rb的依赖包【要先上传到服务器】  [root@study software\_bak]# gem install redis-3.0.0.gem  Successfully installed redis-3.0.0  1 gem installed  Installing ri documentation for redis-3.0.0...  Installing RDoc documentation for redis-3.0.0... |

### 创建实例

如果虚拟机上面本来就有一个Redis实例则需要在安装目录下面建立redis-cluster目录，用来安装六个实例，原有实例不需要加入到集群中。

如果虚拟机上面没有Redis实例，则需要在安装过程中直接指定安装目录为redis-cluster。

redis-cluster目录并不是特定或默认安装目录，是便于管理的一个文件夹。

我们假设虚拟机上面已经有一个存在的redis实例，端口为默认端口6379安装目录为/usr/local/redis（同第4章节安装步骤）。集群安装完成之后进入安装目录，然后在原有实例安装目录下面建立集群的安装目录redis-cluster。

|  |
| --- |
| // 在安装目录下面创建  [root@study local]# mkdir redis-cluster  [root@study local]# cd redis-cluster/  [root@study redis-cluster]# pwd  /usr/local/redis-cluster  // 将原有实例拷贝一份到集群安装目录下面  [root@study redis-cluster]# cp -r /usr/local/redis/bin /usr/local/redis-cluster/  [root@study redis-cluster]# ll  总用量 4  drwxr-xr-x. 2 root root 4096 9月 15 00:57 bin  // 修改文件名称  [root@study redis-cluster]# mv bin redis01  [root@study redis-cluster]# ll  总用量 4  drwxr-xr-x. 2 root root 4096 9月 15 00:57 redis01 |

### 修改配置文件

如果此实例非别处拷贝而来，则需要从源码包中拷贝配置文件（redis.conf）到redis01中，并且还需要修改一个参数daemonize=yes（如第4章节安装过程）。

由于直接是从原有实例基础上面拷贝过来的，所以无需配置daemonize参数。直接进行下述配置即可：

|  |
| --- |
| // 删除原有快照文件【如果不是从原有实例中拷贝过来的，则不要要此步骤】  [root@study redis01]# rm -f dump.rdb  // 修改“端口”和“是否是集群”两个参数，保存关闭  [root@study redis01]# vi redis.conf  port 7001 🡪修改端口号：7001  cluster-enabled yes 🡪打开注释  // 拷贝剩余五个实例  [root@study redis01]# cd ..  [root@study redis-cluster]# cp -r redis01/ redis02  [root@study redis-cluster]# cp -r redis01/ redis03  [root@study redis-cluster]# cp -r redis01/ redis04  [root@study redis-cluster]# cp -r redis01/ redis05  [root@study redis-cluster]# cp -r redis01/ redis06  // 修改端口号  [root@study redis-cluster]# vi redis02/redis.conf  port 7002  [root@study redis-cluster]# vi redis03/redis.conf  port 7003  [root@study redis-cluster]# vi redis04/redis.conf  port 7004  [root@study redis-cluster]# vi redis05/redis.conf  port 7005  [root@study redis-cluster]# vi redis06/redis.conf  port 7006  // 查看端口号  [root@study redis-cluster]# find . -name 'redis\*' | cat redis\*/redis.conf | grep -a0b0 'port 700\*'  1726:port 7001  43128:port 7002  84530:port 7003  125932:port 7004  167334:port 7005  208736:port 7006  // 查看是否开启集群  [root@study redis-cluster]# find . -name 'redis\*' | cat redis\*/redis.conf | grep -a0b0 'cluster-enabled\*'  27140:cluster-enabled yes  68542:cluster-enabled yes  109944:cluster-enabled yes  151346:cluster-enabled yes  192748:cluster-enabled yes  234150:cluster-enabled yes |

Redis的持久化方案有哪些？

1. Rdb形式：
2. AOF形式：

【参见：redis的持久化方案.docx】

### 启动

创建集群前，需要启动六个实例。

|  |
| --- |
| // 在集群安装目录redis-cluster目录下面创建启动脚本  [root@study redis-cluster]# vi startall.sh  # 启动01  cd redis01  ./redis-server redis.conf  # 启动02  cd ../redis02  ./redis-server redis.conf  # 启动03  cd ../redis03  ./redis-server redis.conf  # 启动04  cd ../redis04  ./redis-server redis.conf  # 启动05  cd ../redis05  ./redis-server redis.conf  # 启动06  cd ../redis06  ./redis-server redis.conf  // 添加可执行属性  [root@study redis-cluster]# chmod +x startall.sh  // 运行启动  [root@study redis-cluster]# ./startall.sh  // 查看是否全部启动  [root@study redis-cluster]# ps aux |grep redis  root 1426 0.1 0.7 137452 7528 ? Ssl 02:16 0:00 ./redis-server \*:7001 [cluster]  root 1428 0.2 0.7 137452 7532 ? Ssl 02:16 0:00 ./redis-server \*:7002 [cluster]  root 1432 0.1 0.7 137452 7524 ? Ssl 02:16 0:00 ./redis-server \*:7003 [cluster]  root 1436 0.1 0.7 137452 7528 ? Ssl 02:16 0:00 ./redis-server \*:7004 [cluster]  root 1442 0.2 0.7 137452 7532 ? Ssl 02:16 0:00 ./redis-server \*:7005 [cluster]  root 1444 0.1 0.7 137452 7528 ? Ssl 02:16 0:00 ./redis-server \*:7006 [cluster]  root 1450 0.0 0.0 103260 888 pts/0 S+ 02:17 0:00 grep redis |

### 创建集群

|  |
| --- |
| // 将源码包中的ruby脚本拷贝到集群安装目录下，便于使用  [root@study redis-cluster]# cp /software\_bak/redis-3.0.0/src/\*.rb .  [root@study redis-cluster]# ll  总用量 72  drwxr-xr-x. 2 root root 4096 9月 15 01:34 redis01  drwxr-xr-x. 2 root root 4096 9月 15 01:43 redis02  drwxr-xr-x. 2 root root 4096 9月 15 01:38 redis03  drwxr-xr-x. 2 root root 4096 9月 15 01:40 redis04  drwxr-xr-x. 2 root root 4096 9月 15 01:40 redis05  drwxr-xr-x. 2 root root 4096 9月 15 01:40 redis06  -rwxr-xr-x. 1 root root 48141 9月 15 02:06 redis-trib.rb  // 执行脚本，创建集群  [root@study redis-cluster]# ./redis-trib.rb create --replicas 1 192.168.100.202:7001 192.168.100.202:7002 192.168.100.202:7003 192.168.100.202:7004 192.168.100.202:7005 192.168.100.202:7006  >>> Creating cluster  Connecting to node 192.168.100.202:7001: OK  Connecting to node 192.168.100.202:7002: OK  Connecting to node 192.168.100.202:7003: OK  Connecting to node 192.168.100.202:7004: OK  Connecting to node 192.168.100.202:7005: OK  Connecting to node 192.168.100.202:7006: OK  >>> Performing hash slots allocation on 6 nodes...  Using 3 masters:  192.168.100.202:7001  192.168.100.202:7002  192.168.100.202:7003  Adding replica 192.168.100.202:7004 to 192.168.100.202:7001  Adding replica 192.168.100.202:7005 to 192.168.100.202:7002  Adding replica 192.168.100.202:7006 to 192.168.100.202:7003  M: b978a0606a7cfd02b07ad271bf40037824f5344b 192.168.100.202:7001  slots:0-5460 (5461 slots) master  M: 869a15e47e5fa33237ee4d7b2276777fc679f00f 192.168.100.202:7002  slots:5461-10922 (5462 slots) master  M: 36ed2994cdbb6b75bd0f005e102351115d21a40c 192.168.100.202:7003  slots:10923-16383 (5461 slots) master  S: 712f8af71f954ce68efe43502093c90c20f390ed 192.168.100.202:7004  replicates b978a0606a7cfd02b07ad271bf40037824f5344b  S: 5ca3ec4abed620ec0f1f2ed63b38e044be91f3e9 192.168.100.202:7005  replicates 869a15e47e5fa33237ee4d7b2276777fc679f00f  S: 968344d168c949afe9015c8bee8c5ef0d3eeaaab 192.168.100.202:7006  replicates 36ed2994cdbb6b75bd0f005e102351115d21a40c  Can I set the above configuration? (type 'yes' to accept): yes  >>> Nodes configuration updated  >>> Assign a different config epoch to each node  >>> Sending CLUSTER MEET messages to join the cluster  Waiting for the cluster to join...  >>> Performing Cluster Check (using node 192.168.100.202:7001)  M: b978a0606a7cfd02b07ad271bf40037824f5344b 192.168.100.202:7001  slots:0-5460 (5461 slots) master  M: 869a15e47e5fa33237ee4d7b2276777fc679f00f 192.168.100.202:7002  slots:5461-10922 (5462 slots) master  M: 36ed2994cdbb6b75bd0f005e102351115d21a40c 192.168.100.202:7003  slots:10923-16383 (5461 slots) master  M: 712f8af71f954ce68efe43502093c90c20f390ed 192.168.100.202:7004  slots: (0 slots) master  replicates b978a0606a7cfd02b07ad271bf40037824f5344b  M: 5ca3ec4abed620ec0f1f2ed63b38e044be91f3e9 192.168.100.202:7005  slots: (0 slots) master  replicates 869a15e47e5fa33237ee4d7b2276777fc679f00f  M: 968344d168c949afe9015c8bee8c5ef0d3eeaaab 192.168.100.202:7006  slots: (0 slots) master  replicates 36ed2994cdbb6b75bd0f005e102351115d21a40c  [OK] All nodes agree about slots configuration.  >>> Check for open slots...  >>> Check slots coverage...  [OK] All 16384 slots covered. |

执行创建集群的脚本中：--replicas 1代表的是单个备份。

到此为止，整个伪集群环境已经搭建完毕。

### 测试

测试时，连接哪一个节点（不分主备）都可以。所以使用下面命令进行连接。

|  |
| --- |
| [root@study redis-cluster]# redis01/redis-cli -h 192.168.100.202 -p 7002 **–c**  192.168.100.202:7002> set a 10  -> Redirected to slot [15495] located at 192.168.100.202:7003  OK  192.168.100.202:7003> set hello nihao  -> Redirected to slot [866] located at 192.168.100.202:7001  OK  192.168.100.202:7001> set wxy gyl  -> Redirected to slot [13270] located at 192.168.100.202:7003  OK  192.168.100.202:7003> set beidian posun  -> Redirected to slot [7658] located at 192.168.100.202:7002  OK  192.168.100.202:7002> |

注意在连接命令中，**一定要加上-c参数**。不加-c参数也可以，但是运行的时候就不是集群了，而是单个实例。

### 关闭

|  |
| --- |
| // 在集群安装目录redis-cluster目录下面创建关闭脚本  [root@study redis-cluster]# vi shutdown.sh  # 使用哪一个客户端关闭无所谓，换句话说就是，没有必要非要用自己的客户端  redis01/redis-cli -p 7001 shutdown  redis01/redis-cli -p 7002 shutdown  redis01/redis-cli -p 7003 shutdown  redis01/redis-cli -p 7004 shutdown  redis01/redis-cli -p 7005 shutdown  redis01/redis-cli -p 7006 shutdown  // 修改脚本文件的权限  [root@study redis-cluster]# chmod +x shutdown.sh  // 测试  [root@study redis-cluster]# ps aux | grep redis  root 1273 0.0 0.0 103260 888 pts/0 S+ 17:09 0:00 grep redis  [root@study redis-cluster]# ./startall.sh  [root@study redis-cluster]# ps aux | grep redis  root 1280 0.2 0.7 137452 7400 ? Ssl 17:10 0:00 ./redis-server \*:7001 [cluster]  root 1284 1.2 0.7 137452 7396 ? Ssl 17:10 0:00 ./redis-server \*:7002 [cluster]  root 1288 1.3 0.7 137452 7400 ? Ssl 17:10 0:00 ./redis-server \*:7003 [cluster]  root 1292 0.0 0.7 137452 7388 ? Ssl 17:10 0:00 ./redis-server \*:7004 [cluster]  root 1296 0.0 0.7 137452 7388 ? Ssl 17:10 0:00 ./redis-server \*:7005 [cluster]  root 1300 0.0 0.7 137452 7384 ? Ssl 17:10 0:00 ./redis-server \*:7006 [cluster]  root 1307 0.0 0.0 103260 888 pts/0 S+ 17:10 0:00 grep redis  [root@study redis-cluster]# ./shutdown.sh  [root@study redis-cluster]# ps aux | grep redis  root 1344 0.0 0.0 103260 888 pts/0 S+ 17:12 0:00 grep redis |

## 真集群环境搭建

真集群环境的搭建过程跟伪集群的搭建过程一样，只不过是在使用ruby脚本创建集群时，需要指定的是主机地址和端口。只需要修改命令还要注意相关文件的配置即可。

|  |
| --- |
| ./redis-trib.rb create --replicas 1 192.168.100.X1:7001 192.168.100. X2:7002 192.168.100. X3:7003 192.168.100. X4:7004 192.168.100. X5:7005 192.168.100. X6:7006 |

# 客户端连接redis

## Redis-cli

命令行式客户端，最常用的客户端。可以使用不同主机上面的客户端连接不同的Redis实例。可以类比于Mysql客户端。

## Redis Manager Desktop

图形化客户端，只能连接单机版的Redis，不能连接集群。如果连接不上可以检查一下原因：

1. 是否开启Redis服务器
2. 是否开启防火墙中的6379端口
3. 修改配置文件中的bind参数，不要绑定IP登录

可类比MySQL相关操作。

## Jedis

1. 加入Jedis的依赖包
2. 使用Jedis连接测试，或者使用JedisPool连接测试。

此处不再赘述。

注意：在测试集群环境时，需要修改防火墙配置文件，开启7001~7006端口。