**1.redis简介**

NoSQL（Not only SQL）属于非关系型数据库；redis就属于非关系型数据库

**为什么需要NoSQL，主要应对以下问题**：

高并发读写

海量数据的高效率存储与访问

高可扩展性和高可用性

**NoSQL产品**：MangoDB、Redis（主流）

**NoSQL的特点**：

易扩展

灵活的数据模型

大数据量、高性能、高可用性

**高性能键值对数据库，支持的键值数据类型**：

字符串类型

列表类型

有序集合类型

散列类型

集合类型

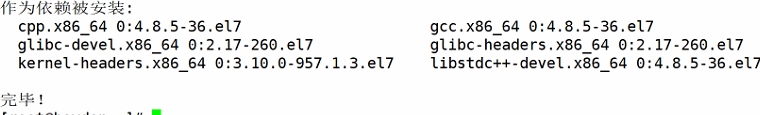
**Redis的应用场景**

缓存、任务队列、网站访问统计、数据过期处理、应用排行榜、分布式集群架构中的session分离

**2.redis安装**

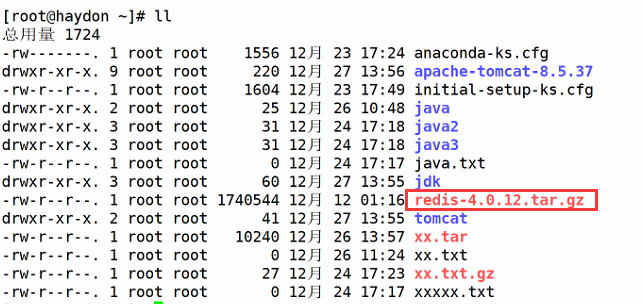
**安装gcc**

yum –y install gcc-c++



Wget方式下载redis压缩包，并解压，以及编译

wget http://download.redis.io/releases/redis-4.0.12.tar.gz



**解压：**

**tar –zxvf /root/redis-xxxxx**

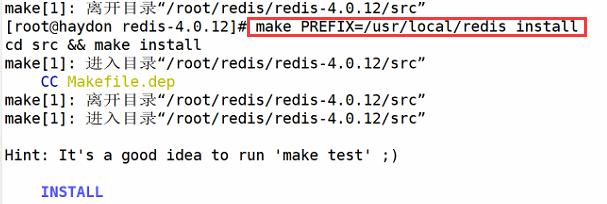


**编译**

进入redis的目录，使用make

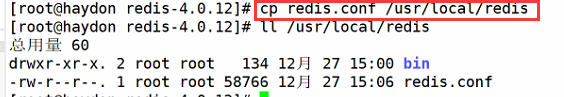
**安装redis**

make PREFIX=/usr/local/redis install





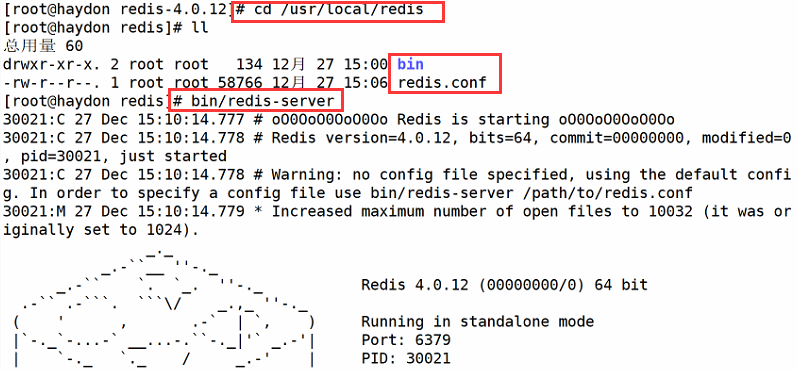
Cd回到root，我们需要把一个配置文件 复制到redis下 后台启动用到



**启动和关闭redis服务**

进入/usr/local/redis目录，

然后使用bin/redis-server启动redis

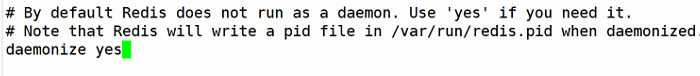


通过ctrl+c退出当前程序

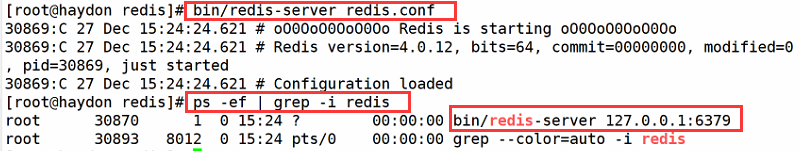
正式因为这个原因，必须让其后台运行

通过修改配置文件redis.conf





进入redis目录，加载配置文件运行



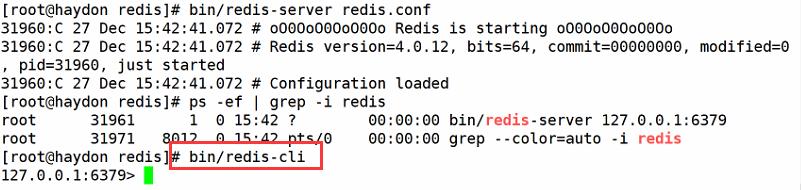
**关闭redis服务**

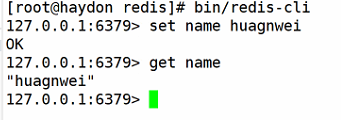
bin/redis-cli shutdown



**Redis基本使用**

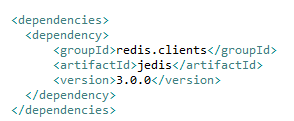
bin/redis-cli

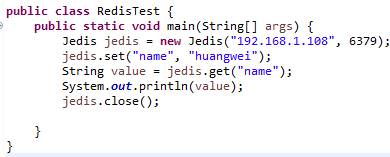


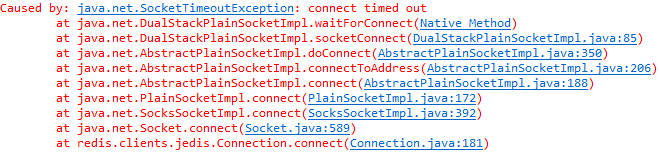


**3.使用jedis连接redis**

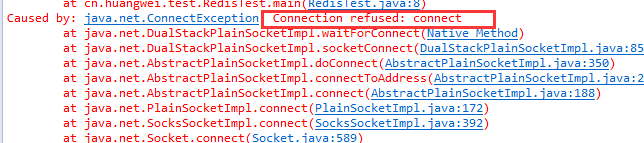
新建maven项目，在pom.xml中导入依赖







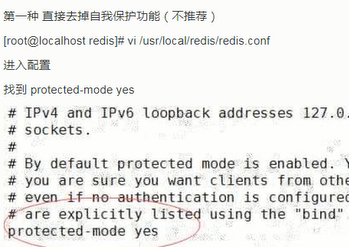
连接超时，因为redis的6379端口没有开放



注释掉这一行

修改后还会出错





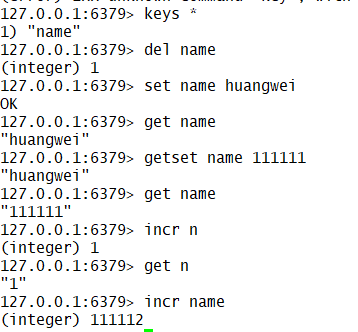


**Jedis连接池的使用**

|  |
| --- |
| **package** cn.huangwei.test;  **import** redis.clients.jedis.Jedis;  **import** redis.clients.jedis.JedisPool;  **import** redis.clients.jedis.JedisPoolConfig;  **public** **class** JedisPoolTest {  **public** **static** **void** main(String[] args) {  JedisPoolConfig config = **new** JedisPoolConfig();  config.setMaxTotal(100);//设置最大链接数  config.setMaxIdle(10);//设置最大空闲链接数  //创建连接池  JedisPool pool = **new** JedisPool(config, "192.168.220.128", 6379);  Jedis jedis = **null**;  **try** {  jedis = pool.getResource();  jedis.auth("123456");  jedis.set("name", "huangwei");  String value = jedis.get("name");  System.***out***.println(value);  }**catch**(Exception e) {  e.printStackTrace();  }**finally** {  **if**(jedis != **null**) {  jedis.close();  }    **if**(pool != **null**) {  pool.close();  }  }  }  } |

**4. redis的数据类型**

**字符串**

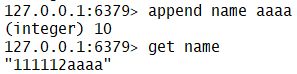


扩展： incrby 指定增量值

decrby 指定减量值

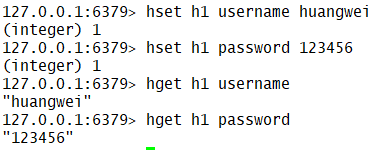
append 源字符串 追加字符串

假设没有定义 直接赋值



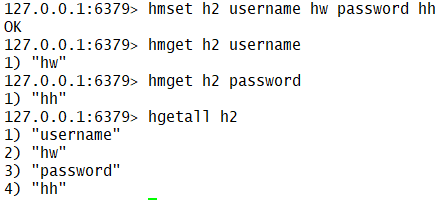
**哈希**

hset 哈希名 key value

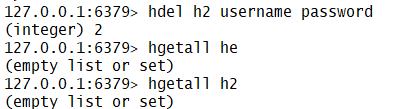


hmset 多个字段一起设置

hmset 哈希名 key value key value



删除



hincrby 增加数字

hincrby 哈希名 key 增量

hexists判断字段是否存在 1表示存在 0表示不存在

hexists 哈希名 key

hlen 哈希名 ：获取hash属性个数

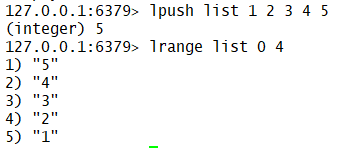
hkeys 哈希名: 获取所有属性的名称

hvals 哈希名：获取所有属性的值

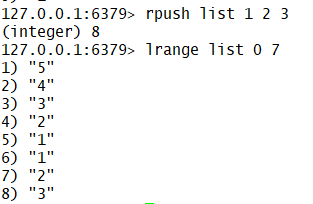
**List**

lpush 从左边开始添加

lrange 获取指定方位的集合元素

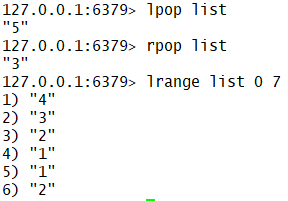


rpush 从右边开始添加



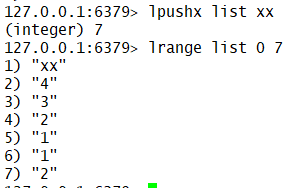
lpop 左侧弹出集合元素

rpop 右侧弹出集合元素



llen 查看集合元素的个数

lpushx 集合头部（前面）插入元素（一个）



lrem 从指定方向删除指定个数的指定元素

lrem 集合名 删除个数 删除的值

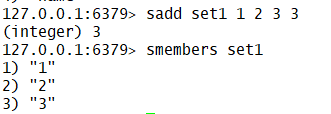
索引从0开始

lset 集合名 索引 设置值

**Set：不允许有重复的数据元素**

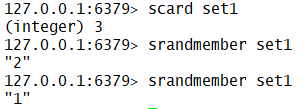
sadd key member:添加set元素

smembers key 查询指定key集合的元素



scard key：计算元素总数

srandmember key :随机取一个元素



sinterstore 把交集结果存到新集合中去

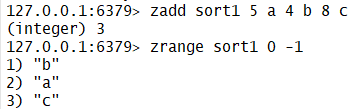
sinterstore 新集合 集合1 集合2

sunionstore 新集合 集合1 集合2：把并集结果存到新集合中

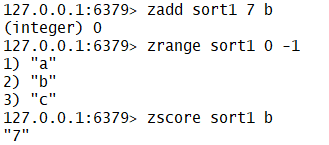
**Sorted-set**

zadd 添加元素 里面包括评分和值

zrange 查找元素 -1代表是最后一个



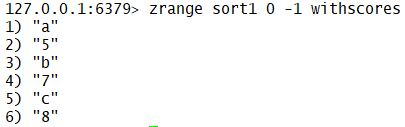
如果添加的时候，值已经有了，评分不一样，就会覆盖旧的评分



zrem key 值 ：删除元素

zcard key ：查找元素个数

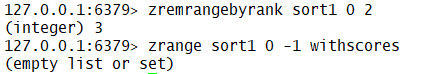
zrange key 0 -1 withscores 罗列评分和值



zrevrange 降序排列，用法与上面一致

zremrangebyrank 根据排名来删除元素 删除3个

zremrangebyrank sort1 0 2

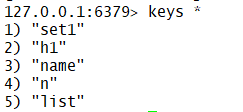


zremrangebyscore 根据具体评分范围来删除元素

zremrangebyscore sort1 10 16 删除评分10-16的

zrangebyscore sort1 18 23 withscores limit 0 2；显示18-13分数的第0个开始，每页显示两个

Keys通用操作



keys s\*表示s开头的集合

key s？表示s开头后面只有1个字符 的集合

del 删除key，可以跟多个key

exists判断key是否存在 1表示存在，0表示不存在

get 获取元素

rename对key重命名

expire 设置n的有效时间 秒

expire n 100

ttl查看剩余时间

type 显示类型

**5.redis持久化**

RDB持久化方法：指定时间间隔写入硬盘

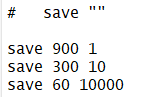
AOF持久化方法：以日志的方式，记录每一个操作，服务器启动后就构建数据库

配置可以金庸持久化功能，也可以同时使用两种方式

**5.1 Redis的持久化之RDB方式**

优势：只有一个文件，时间间隔的数据，可以归档为一个文件，方便压缩转移（就一个文件）

劣势：如果宕机，数据损失比较大，因为它是每一个时间段进行持久化操作的，也就是积攒的数据比较多。



每900秒内至少有1个key发生变化，就持久化

每300秒内至少有10个key发生变化，就持久化

每60秒内至少有10000个key发生变化，就持久化

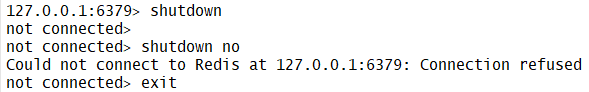
保存的持久化文件名 默认是dump.rdb



文件存储路径是当前路径

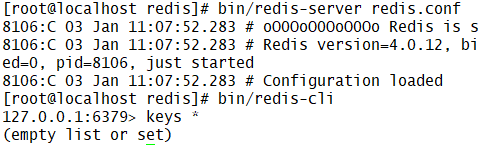
RDB备份和回复数据

假如遇到断电或者宕机或者自然灾害，我们需要恢复数据，模拟测试

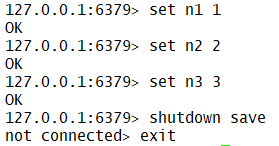


删除dump.rdb文件

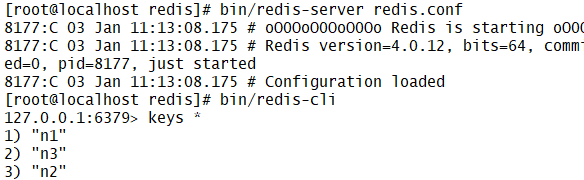
然后重新链接，查看keys是否为空



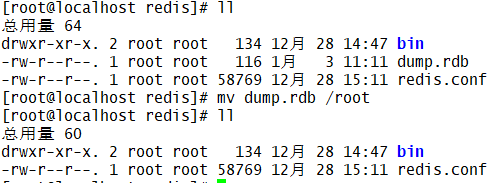
添加数据，然后shutdown保存退出



加入这个时候重启redis，这个时候启动过程会进行rdb check验证，然后加载redis目录下rdb文件；加载数据

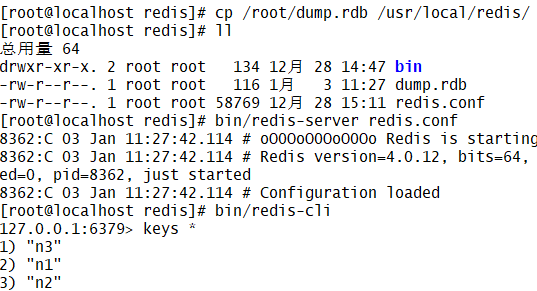


这里将redis下的rdb文件剪切到其他地方去，然后再启动下

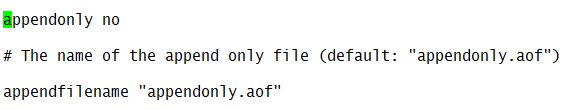




此时数据已经没了，如果需要恢复，只需要将刚刚的rdb文件拷贝到当前目录即可



**5.2 Redis的持久化之AOF方式**



默认关闭aof方式 修改成yes就会开启



三种同步策略：

Always是只要发生修改，立即同步（推荐使用，安全性最高）

Everysec每秒同步一次

No不同步

**6.redis集群概述**

redis cluster是去中心化，去中间件的，也就是说，集群中的每个节点都是平等关系，都是对等的，每个节点都保存各自的数据和整个集群的状态。每个节点都和其他所有节点链接，而且这些连接保持活跃，这样就保证了我们只需要连接集群中的任意一个节点，就可以获取到其他节点的数据。

Redis是如何合理分配这些节点和数据的呢？

Redis集群并没有使用传统的一致性哈希来分配数据，而是采用另外一种叫做哈希槽hash slot的方式来分配的。Redis集群默认分配16384个slot，当我们set一个key时，会用CRC16算法来取模得到所属的slot，然后将这个key分到哈希槽区间的节点上，具体算法就是CRC16(key) % 16384

必须要3个以后的主节点，否则创建集群时会失败。

所以如果我们假设现在又3个节点已经组成了集群，分别是：A、B、C三个节点，他们可以时一台机器上的三个端口，也可以是三台不同的服务器。那么，采用哈希槽方式来分配16384个slot的话，他们三个节点分别承担的slot区间是：

A:覆盖0-5460 B：覆盖5461-10922 C: 覆盖10923-16383

如果现在设置一个key，my\_name, set my\_name wind

这样按照redis cluster的哈希算法：CRC（‘my\_name’） % 16384 = 2412，就会分配到A上了

同样，当连接ABC任何一个节点，想获取my\_name这个key时，也会这样的算法，从指定节点拿取数据

**6.1单机多节点集群操作**

**①Redis安装包里有个集群工具，要复制到/usr/local/bin里去**

cp /home/redis/redis-4.0.12/src/redis-trib.rb /usr/local/bin

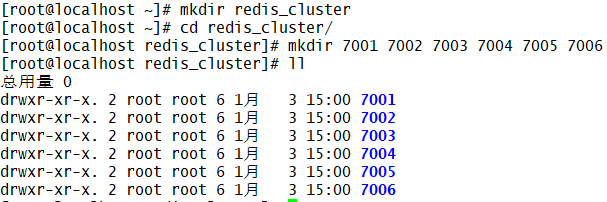
**②修改配置，创建节点**

弄六个节点，三主三从

端口分别是7001，7002，7003，7004，7005，7006

我们首先在root目录下新建一个redis\_cluster目录，然后在该目录下创建6个目录，分别是7001-7006，用来存储redis的配置文件

这里使用redis集群，首先要修改redis的配置文件redis.conf



**复制一份配置文件到7001目录下**



**修改以下几点**：

Port 7001 //其他几个文件夹也分别配置

Daemonize yes //redis后台运行

Pidfile /var/run/redis\_7001.pid //pidfile文件对应7001-7006

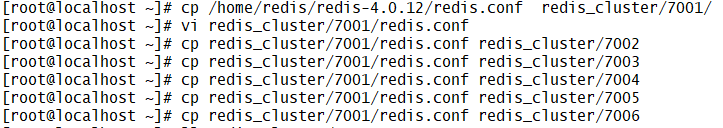
Cluster-enable yes //开启集群

Cluster-config-file nodes\_7001.conf //保存节点配置，自动创建，自动更新对应7001-7006

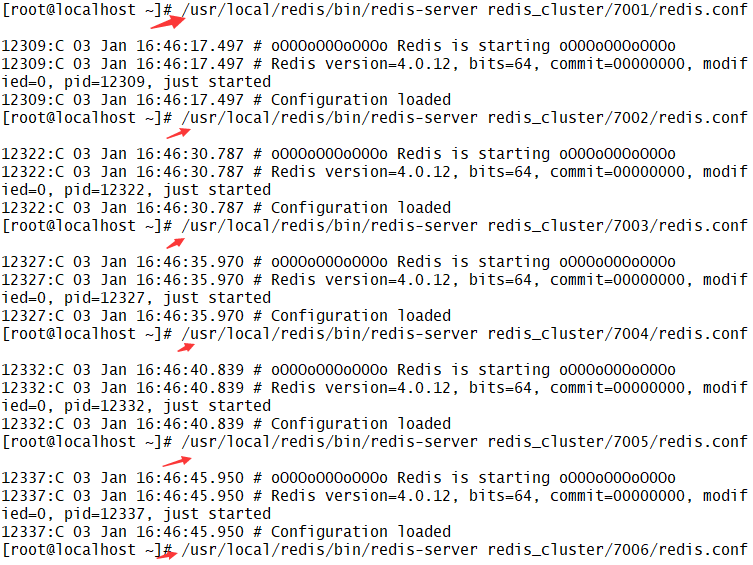
Cluster-node-timeout 5000 //集群超时时间，节点超过这个时间没反应就断定时宕机

Appendonly yes //存储方式，aof，将写操作记录到保存日志中

7001的配置文件修改完之后，在复制到其他7002-7006，然后修改配置

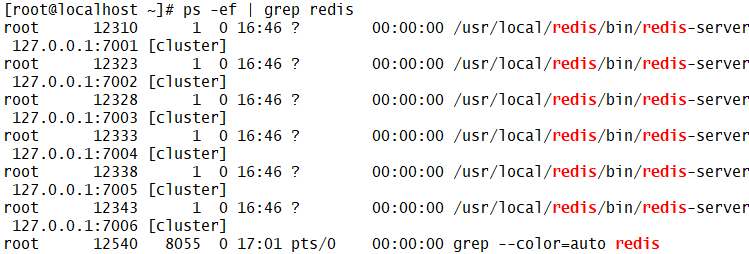


**③启动六个节点的redis**



**④查找redis进程**

ps –ef|grep redis

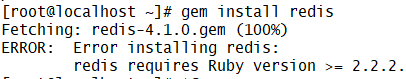


**⑤创建集群**

使用之前需要安装ruby，以及redis和ruby连接

yum –y install ruby ruby-devel rubygems rpm-build

gem install redis



解决：

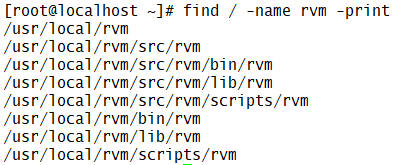
安装rvm

gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3 7D2BAF1CF37B13E2069D6956105BD0E739499BDB

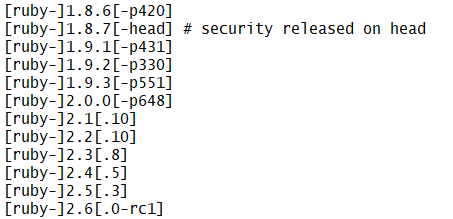
curl -sSL https://get.rvm.io | bash -s stable

find / -name rvm –print 查看是否安装成功

source /usr/local/rvm/scripts/rvm



查看rvm库中一直的ruby版本

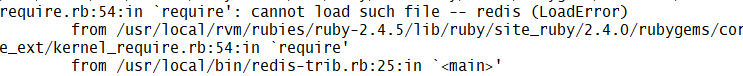


安装一个ruby的版本

rvm install 2.4.5

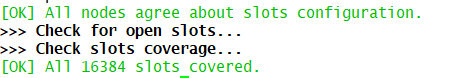
创建集群

redis-trib.rb create --replicas 1 127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005 127.0.0.1:7006



如果出现问题，说明没有装ruby和redis 的链接接口

gem install redis

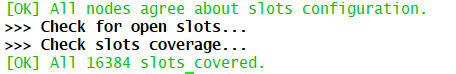




出现这个问题：

先把刚刚启动redis的进程给关闭，然后删除一些数据文件rdb，aof，node-conf等文件，这样重新进入就可以了

删除之后重新启动六个节点redis

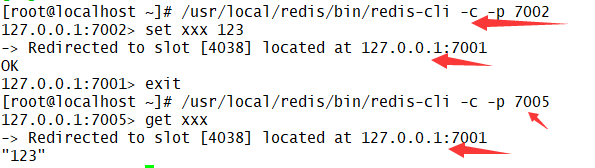


**⑥集群数据测试**

链接任意一个节点，添加一个key；

/usr/local/redis/bin/redis-cli –c –p 7002

-c表示集群，-p表示端口

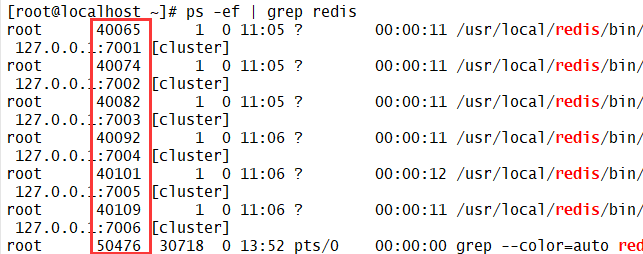


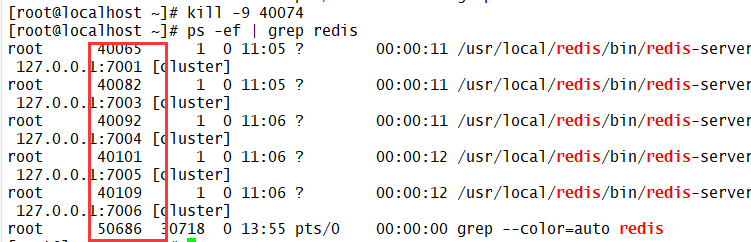
**⑦集群宕机的情况**

假如一个主节点宕机了，从节点会升级为主节点，，如果主从都宕机了，这个节点就会挂掉了

假设干掉7002这个主节点

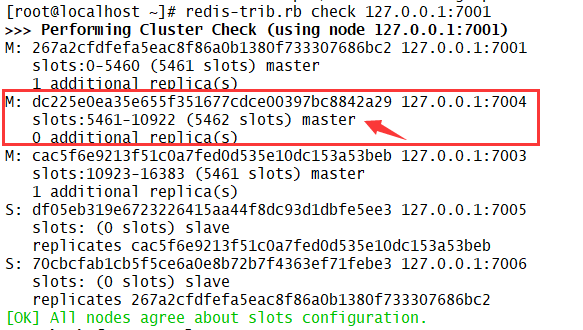
先查找redis的连接端口，然后找到7002对应的端口，kill -9 xxxx



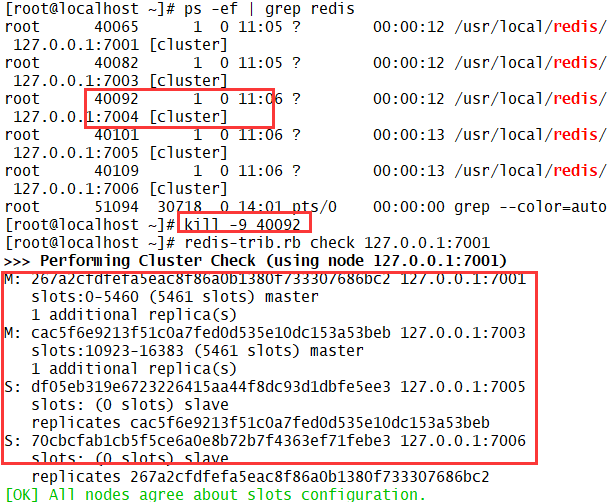


查看集群的情况

redis-trib.rb check 127.0.0.1:7001



在考虑把7004干掉，看会出现什么情况



这个的主从已经结束了，只剩下两个主，两个从

