Redis集群部署文档(centos6系统)

（要让集群正常工作至少需要3个主节点，在这里我们要创建6个redis节点，其中三个为主节点，三个为从节点，对应的redis节点的ip和端口对应关系如下）

127.0.0.1:7000  
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

1：下载redis。官网下载3.0.0版本，之前2.几的版本不支持集群模式

下载地址：<https://github.com/antirez/redis/archive/3.0.0-rc2.tar.gz>

2：上传服务器，解压，编译

|  |
| --- |
| tar -zxvf redis-3.0.0-rc2.tar.gz  mv redis-3.0.0-rc2.tar.gz redis3.0  cd /usr/local/redis3.0  make  make install |

3：创建集群需要的目录

|  |
| --- |
| mkdir -p /usr.local/cluster  cd /usr.local/cluster  mkdir 7000  mkdir 7001  mkdir 7002  mkdir 7003  mkdir 7004  mkdir 7005 |

4：修改配置文件redis.conf

|  |
| --- |
| cp /usr/local/redis3.0/redis.conf  /usr.local/cluster  vi redis.conf  ##修改配置文件中的下面选项  port 7000  daemonize yes  cluster-enabled yes  cluster-config-file nodes.conf  cluster-node-timeout 5000  appendonly yes  ##修改完redis.conf配置文件中的这些配置项之后把这个配置文件分别拷贝到7000/7001/7002/7003/7004/7005目录下面  cp /usr/local/cluster/redis.conf /usr/local/cluster/7000  cp /usr/local/cluster/redis.conf /usr/local/cluster/7001  cp /usr/local/cluster/redis.conf /usr/local/cluster/7002  cp /usr/local/cluster/redis.conf /usr/local/cluster/7003  cp /usr/local/cluster/redis.conf /usr/local/cluster/7004  cp /usr/local/cluster/redis.conf /usr/local/cluster/7005    ##注意：拷贝完成之后要修改7001/7002/7003/7004/7005目录下面redis.conf文件中的port参数，分别改为对应的文件夹的名称 |

5：分别启动这6个redis实例

|  |
| --- |
| cd /usr/local/cluster/7000  redis-server redis.conf  cd /usr/local/cluster/7001  redis-server redis.conf  cd /usr/local/cluster/7002  redis-server redis.conf  cd /usr/local/cluster/7003  redis-server redis.conf  cd /usr/local/cluster/7004  redis-server redis.conf  cd /usr/local/cluster/7005  redis-server redis.conf      ##启动之后使用命令查看redis的启动情况ps -ef|grep redis  如下图显示则说明启动成功 |

6：执行redis的创建集群命令创建集群

|  |
| --- |
| cd /usr/local/redis3.0/src  ./redis-trib.rb  create --replicas 1 127.0.0.1:7000 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 |

6.1执行上面的命令的时候会报错，因为是执行的ruby的脚本，需要ruby的环境

错误内容：/usr/bin/env: ruby: No such file or directory

所以需要安装ruby的环境，这里推荐使用yum install ruby安装

|  |
| --- |
| yum install ruby |

6.2然后再执行第6步的创建集群命令，还会报错，提示缺少rubygems组件，使用yum安装

错误内容：

./redis-trib.rb:24:in `require': no such file to load -- rubygems (LoadError)

from ./redis-trib.rb:24

|  |
| --- |
| yum install rubygems |

6.3再次执行第6步的命令，还会报错，提示不能加载redis，是因为缺少redis和ruby的接口，使用gem 安装

错误内容：

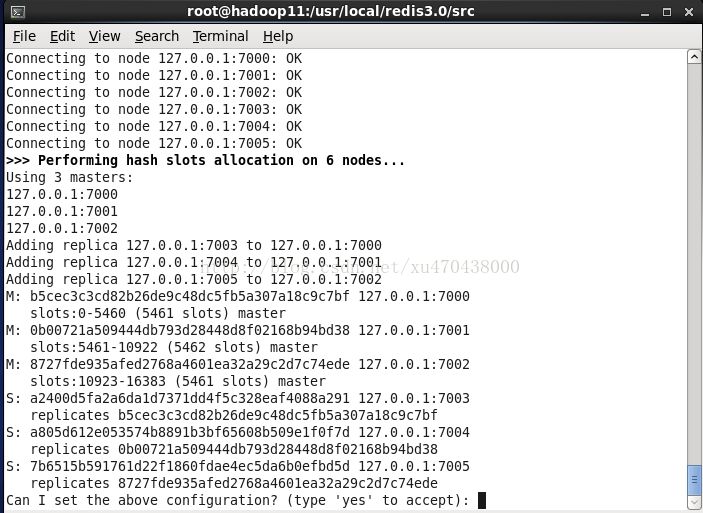
/usr/lib/ruby/site\_ruby/1.8/rubygems/custom\_require.rb:31:in `gem\_original\_require': no such file to load -- redis (LoadError)

from /usr/lib/ruby/site\_ruby/1.8/rubygems/custom\_require.rb:31:in `require'

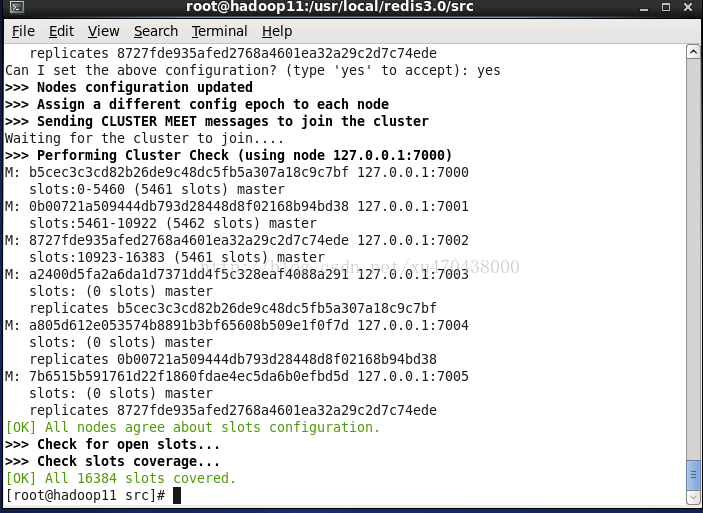
from ./redis-trib.rb:25

|  |
| --- |
| gem install redis |

6.4 再次执行第6步的命令，正常执行



输入yes，然后配置完成。



至此redis集群即搭建成功！

7：使用redis-cli命令进入集群环境

|  |
| --- |
| redis-cli -c -p 7000 |

8.查看集群信息：

Redis-cli –p 端口号 cluster nodes

9.集群的时候使用命令行添加数据必须加一个参数 -C

Redis-cli –c –p 7000

然后这样登陆以后才能在集群的模式下添加成功；

动态添加节点部分

**一、概述**

要让集群正常工作至少需要3个主节点，在这里我们要创建6个redis节点，其中三个为主节点，三个为从节点，对应的redis节点的ip和端口对应关系如下

127.0.0.1:7000  
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

**二、准备环境**

**安装ruby**

**集群环境需要ruby，所以需要安装下**

yum install ruby

yum install ruby-rdoc

ruby setup.rb

安装完毕后启动依然报错

需要安装redis库

gem install redis

问题解决

下载最新redis版本

3.06 并进行构建

创建用户

useradd redis

passwd redis

切换用户

su – redis

上传redis包到home目录

tar -zxvf redis-3.0.6.tar.gz

mv redis-3.0.6 redis3.0

cd redis3.0

make

本环境为伪集群环境，所以在一台机器，多机情况相同

mkdir  /home/redis/ cluster-test

cd /home/redis/ cluster-test

mkdir 7000

mkdir 7001

mkdir 7002

mkdir 7003

mkdir 7004

mkdir 7005

**三、修改配置**

**将配置文件CP到各个集群节点目录下并修改端口号**

**将redis-server cp到各个集群几点目录下**

**详见配置文件范例**

**四、启动环境**

**运行 startAll.sh文件即可启动集群**

**Client 连接集群任意一个节点即可**

**五、            添加新主节点**

**使用命令./redis-trib.rb add-node 10.4.89.16:7006 10.4.89.16:7000**

**第一个为新节点IP端口 第二个参数为任意节点IP**

**添加后日志信息**

**>>> Addingnode 10.4.89.16:7006 to cluster 10.4.89.16:7000**

**>>>Performing Cluster Check (using node 10.4.89.16:7000)**

**M:7e81d801c3ce503868c67dd6b47b82b8e04458ce 10.4.89.16:7000**

**slots:0-5460 (5461 slots) master**

**1 additional replica(s)**

**M:c744b599c7cf31c05ed5a95f831461de8d5dfb95 10.4.89.16:7002**

**slots:10923-16383 (5461 slots) master**

**1 additional replica(s)**

**M:d5297c5cee8a142123775224f39cce499fd7c435 10.4.89.16:7001**

**slots:5461-10922 (5462 slots) master**

**1 additional replica(s)**

**S:d2c5ca33820cda4f8f1e98fbfcd72cf91bc7b676 10.4.89.16:7004**

**slots: (0 slots) slave**

**replicatesd5297c5cee8a142123775224f39cce499fd7c435**

**S:b501af87c221ba53f7edd76bf1b5a77352d963be 10.4.89.16:7003**

**slots: (0 slots) slave**

**replicates7e81d801c3ce503868c67dd6b47b82b8e04458ce**

**S:77472bc34b72345b939b585157f4fc70fea81327 10.4.89.16:7005**

**slots: (0 slots) slave**

**replicatesc744b599c7cf31c05ed5a95f831461de8d5dfb95**

**[OK] All nodes agreeabout slots configuration.**

**>>> Checkfor open slots...**

**>>> Checkslots coverage...**

**[OK] All 16384 slotscovered.**

**>>> SendCLUSTER MEET to node 10.4.89.16:7006 to make it join the cluster.**

**脚本先检查了集群状态，后添加了节点，添加后节点是空节点**

**./redis-cli -c -p7000 cluster nodes**

**查看后发现**

**2e9bdb7ea488a3d5ee5b92f32d65a176bf7b41bd10.4.89.16:7006 master - 0 1453774165361 0 connected**

**是一个master节点，但是没有分配任何哈希槽，所以此节点不参加任何集群选取策略**

**下面我们为新节点分布数据槽**

**./redis-trib.rb  reshard 10.4.89.16:7006**

**会询问向移动多少哈希槽到此节点**

**>>>Performing Cluster Check (using node 10.4.89.16:7006)**

**M:2e9bdb7ea488a3d5ee5b92f32d65a176bf7b41bd 10.4.89.16:7006**

**slots: (0 slots) master**

**0 additional replica(s)**

**S: b501af87c221ba53f7edd76bf1b5a77352d963be10.4.89.16:7003**

**slots: (0 slots) slave**

**replicates7e81d801c3ce503868c67dd6b47b82b8e04458ce**

**S:77472bc34b72345b939b585157f4fc70fea81327 10.4.89.16:7005**

**slots: (0 slots) slave**

**replicates c744b599c7cf31c05ed5a95f831461de8d5dfb95**

**M:7e81d801c3ce503868c67dd6b47b82b8e04458ce 10.4.89.16:7000**

**slots:0-5460 (5461 slots) master**

**1 additional replica(s)**

**S:d2c5ca33820cda4f8f1e98fbfcd72cf91bc7b676 10.4.89.16:7004**

**slots: (0 slots) slave**

**replicates d5297c5cee8a142123775224f39cce499fd7c435**

**M:d5297c5cee8a142123775224f39cce499fd7c435 10.4.89.16:7001**

**slots:5461-10922 (5462 slots) master**

**1 additional replica(s)**

**M:c744b599c7cf31c05ed5a95f831461de8d5dfb95 10.4.89.16:7002**

**slots:10923-16383 (5461 slots) master**

**1 additional replica(s)**

**[OK] All nodes agreeabout slots configuration.**

**>>> Checkfor open slots...**

**>>> Checkslots coverage...**

**[OK] All 16384 slotscovered.**

**How many slots doyou want to move (from 1 to 16384)?**

**此次我们移动3000**

**What is thereceiving node ID?**

**接收这些哈希槽的ID**

**Source node #1:**

**想从哪个节点移动哈希槽，输入节点ID即可，如果输入all则各个节点分散获取**

**./redis-cli -c -p7000 cluster nodes 查看现在状态发现已经有哈希槽了**

**2e9bdb7ea488a3d5ee5b92f32d65a176bf7b41bd10.4.89.16:7006 master - 0 1453774829906 7 connected 0-998 5461-646110923-11921**

**分配槽点后数据会自动到新的节点上来。**

**六、            添加从节点**

添加从节点和添加主节点的策略是一样的

先启动需要添加的节点，然后开始添加新节点

./redis-trib.rb add-node --slave10.4.89.16:7007 10.4.89.16:7000

第一个参数为新节点IP  第二个为任意节点

[redis@api cluster-test]$ ./redis-trib.rbadd-node --slave 10.4.89.16:7007 10.4.89.16:7000

>>> Adding node 10.4.89.16:7007 tocluster 10.4.89.16:7000

>>> Performing Cluster Check(using node 10.4.89.16:7000)

M: 7e81d801c3ce503868c67dd6b47b82b8e04458ce10.4.89.16:7000

  slots:999-5460 (4462 slots) master

   1additional replica(s)

M: c744b599c7cf31c05ed5a95f831461de8d5dfb9510.4.89.16:7002

  slots:11922-16383 (4462 slots) master

   1additional replica(s)

M: d5297c5cee8a142123775224f39cce499fd7c43510.4.89.16:7001

  slots:6462-10922 (4461 slots) master

   1additional replica(s)

S: d2c5ca33820cda4f8f1e98fbfcd72cf91bc7b67610.4.89.16:7004

  slots: (0 slots) slave

  replicates d5297c5cee8a142123775224f39cce499fd7c435

S: b501af87c221ba53f7edd76bf1b5a77352d963be10.4.89.16:7003

  slots: (0 slots) slave

  replicates 7e81d801c3ce503868c67dd6b47b82b8e04458ce

S: 77472bc34b72345b939b585157f4fc70fea8132710.4.89.16:7005

  slots: (0 slots) slave

  replicates c744b599c7cf31c05ed5a95f831461de8d5dfb95

M: 2e9bdb7ea488a3d5ee5b92f32d65a176bf7b41bd10.4.89.16:7006

  slots:0-998,5461-6461,10923-11921 (2999 slots) master

   0additional replica(s)

[OK] All nodes agree about slotsconfiguration.

>>> Check for open slots...

>>> Check slots coverage...

[OK] All 16384 slots covered.

Automatically selected master10.4.89.16:7006

>>> Send CLUSTER MEET to node10.4.89.16:7007 to make it join the cluster.

Waiting for the cluster to join.

>>> Configure node as replica of10.4.89.16:7006.

[OK] New node added correctly.

其实此节点加给了7006

看下集群信息./redis-cli -c -p 7000 cluster nodes

发现确实加给了7007

看下info信息

./redis-cli -c -p 7000 cluster info

一切正常

**七、           移除节点**

通过 redis-trib 提供的 del-node 命令可以移除一个slave节点:

./redis-trib del-node 127.0.0.1:7000 `<node-id>`

**在移除master节点之前必须确保它是空的否则你的集群有可能会不可用**

如果你要移除的master节点不是空的，你需要先用重新分片命令来把数据移到其他的节点。另外一个移除master节点的方法是先进行一次手动的失效备援，等它的slave被选举为新的master，并且它被作为一个新的slave被重新加到集群中来之后再移除它。很明显，如果你是想要减少集群中的master数量，这种做法没什么用。在这种情况下你还是需要用重新分片来移除数据后再移除它。

现在我们来移动7006的哈希槽到其他的机器，目前集群为4 M 4S 所以我们给另外3M每个上移动1000哈希槽

./redis-trib.rb rehard 10.4.89.16:7006

How many slots do you want to move (from 1to 16384)? 1000  移动1000

What is the receiving node ID? 7e81d801c3ce503868c67dd6b47b82b8e04458ce到7000

Source node#1:2e9bdb7ea488a3d5ee5b92f32d65a176bf7b41bd 这次只从7006移动

Source node #2：done

开始移动

./redis-cli -c -p 7000 cluster nodes 查看哈希槽数量

./redis-trib.rb  check 10.4.89.16:7000 使用查看集群正确性

现在我们可以安心的删除7006节点了，但是我们还有一个7007S节点，当然你也可以直接用命令删除，这里我们把它移动到另一个M下面

登录要转移的节点

./redis-cli -c -p 7007

输入转移命令

cluster replicate7e81d801c3ce503868c67dd6b47b82b8e04458ce

查看集群状态

./redis-trib.rb  check 10.4.89.16:7000

发现已经转移到7000下了

./redis-cli -c -p 7000 cluster nodes 查看哈希槽数量

现在就可以删除7006M节点了

./redis-trib.rb del-node 10.4.89.16:70062e9bdb7ea488a3d5ee5b92f32d65a176bf7b41bd

发现已经删除，检查集群正确性

发现已经没有7006M节点，删除完毕

**八、           复制迁移**

可以很好的提升系统抗灾性的策略

虽然在redis集群中通过以下命令是可以将一个slave节点重新配置为另外一个master的slave：

1. CLUSTER REPLICATE <master-node-id>

然而有时候你不想找系统管理员来帮忙，又想自动的将一个复制节点从一个master下移动到另外一个master下。 这种情况下的复制节点的自动重配置被称为复制迁移。复制迁移可以提升系统的可靠性。

在某种情况下，你想让集群的复制节点从一个master迁移到另一个master的原因可能是：集群的抗崩溃能力总是跟集群中master 拥有的平均slave数量成正比。  
比 如，如果一个集群中每个master只有一个slave，当master和slave都挂掉的时候这个集群就崩溃了。因为此时有一些哈希槽无法找到了。虽然网络分裂会把一堆节点从集群中孤立出来（这样你一下就会知道集群出问题了），但是其他的更常见的硬件或者软件的问题并不会在多台机器上同时发生，所以很 可能在你的这个集群（平均每个master只有一个slave）有一个slave在早上4点挂掉，然后他的master在随后的早上6点挂掉。这样依然会 导致集群崩溃。

我们可以通过给每个master都再多加一个slave节点来改进系统的可靠性，但是这样很昂贵。复制迁移允许只给某些master增加slave。比方说你的集群有20个节点，10个master，每个master都有1个slave。然后你增加3个 slave到集群中并把他们分配给某几个master节点，这样某些master就会拥有多于1个slave。

当某个 master失去了slave的时候，复制迁移可以将slave节点从拥有富余slave的master旗下迁移给没有slave的master。所以当 你的slave在早上4点挂掉的时候，另一个slave会被迁移过来取代它的位置，这样当master节点在早上5点挂掉的时候，依然有一个slave可 以被选举为master，集群依然可以正常运行。

所以简而言之你应该了解关于复制迁移的哪些方面？

* 集群在迁移的时候会尝试去迁移拥有最多slave数量的master旗下的slave。
* 想利用复制迁移特性来增加系统的可用性，你只需要增加一些slave节点给单个master（哪个master节点并不重要）。
* 复制迁移是由配置项cluster-migration-barrier控制的

**九、           升级节点**

升级从服务器节点很简单，因为你只需要停止节点然后用已更新的Redis版本重启。如果有客户端使用从服务器节点分离读请求，它们应该能够在某个节点不可用时重新连接另一个从服务器。   
    升级主服务器要稍微复杂一些，建议的步骤是：

1. 使用CLUSTER FAILOVER来触发一次手工故障转移主服务器(请看本文档的手工故障转移小节)。
2. 等待主服务器变为从服务器。
3. 像升级从服务器那样升级这个节点。
4. 如果你想让你刚刚升级的节点成为主服务器，触发一次新的手工故障转移，让升级的节点重新变回主服务器。

    你可以按照这些步骤来一个节点一个节点的升级，直到全部节点升级完毕。

**十、           数据迁移**

假设你的已存在数据集已经被拆分到了N个主服务器上，如果你没有已存在的分片的话N=1，你需要下面的步骤来迁移你的数据集到Redis集群：

1. 停止你的客户端。当前没有自动在线迁移(live-migration)到Redis集群的可能。你也许可以通过精心策划一次在你的程序或环境上下文中的在线迁移来办到。
2. 使用BGREWRITEOF命令为所有你的N个主服务器生成一个追加文件，然后等待AOF文件完全生成。
3. 按照aof-1到af-N保存你的AOF文件到某处。此时愿意的话你可以停掉你的旧实例(这很有用，因为在非虚拟化的部署中，你常常需要重用这些计算机)。
4. 创建一个由N个主服务器和0个从服务器组成的Redis集群。你可以稍后添加从服务器。确保所有你的节点都是使用追加文件来持久化。
5. 停止所有的集群节点，用你已存在的追加文件替换他们的朱家文件，aof-1替换第一节点，aof-2替换第二个节点，一直到aof-N。
6. 使用新的AOF文件来重启你的Redis集群。它们会抱怨按照配置有些键不应该出现。
7. 使用redis-trib fix命令来修正集群，这样键就会根据每个节点的哈希槽被迁移了。
8. 最后使用redis-trib check来确保集群是正常的。
9. 重启被修改为支持Redis集群的客户端。

其实上边是官方给出的文档，这个坑可是花了我不少时间我还以为数据迁移完蛋了呢。其实很简单，就是如果你以前用的是sentinel的集群，那就导出一份数据就好了，然后先创建只有一个MASTER节点的集群，这点很重要，只能有一个节点。然后数据倒过来重启，会报错，告诉你什么已经有数据之类的，不要搭理他。**redis-trib.rb fix 用这个命令修复下就OK了**。然后你就会发现好了，它会自动check下，然后你就可以开始一个一个创建新的master了，根据上边的方法增加节点。

**十一、           注意**

**目前集群的启动只能空节点启动，当节点有数据时会有err警告，但是只要进行fix就好了 redis-trib.rb fix 用这个命令修复下就OK了，官方文档 讲解的很少当时居然没有理解，悲愤啊**

集群命令

CLUSTERINFO 打印集群的信息

CLUSTERNODES 列出集群当前已知的所有节点（node），以及这些节点的相关信息。

节点

CLUSTERMEET <ip> <port> 将 ip 和 port 所指定的节点添加到集群当中，让它成为集群的一份子。

CLUSTERFORGET <node\_id> 从集群中移除 node\_id 指定的节点。

CLUSTERREPLICATE <node\_id> 将当前节点设置为 node\_id 指定的节点的从节点。

CLUSTERSAVECONFIG 将节点的配置文件保存到硬盘里面。

槽(slot)

CLUSTERADDSLOTS <slot> [slot ...] 将一个或多个槽（slot）指派（assign）给当前节点。

CLUSTERDELSLOTS <slot> [slot ...] 移除一个或多个槽对当前节点的指派。

CLUSTERFLUSHSLOTS 移除指派给当前节点的所有槽，让当前节点变成一个没有指派任何槽的节点。

CLUSTERSETSLOT <slot> NODE <node\_id> 将槽 slot指派给 node\_id 指定的节点，如果槽已经指派给另一个节点，那么先让另一个节点删除该槽>，然后再进行指派。

CLUSTERSETSLOT <slot> MIGRATING <node\_id> 将本节点的槽 slot迁移到 node\_id 指定的节点中。

CLUSTERSETSLOT <slot> IMPORTING <node\_id> 从node\_id 指定的节点中导入槽 slot 到本节点。

CLUSTERSETSLOT <slot> STABLE 取消对槽 slot 的导入（import）或者迁移（migrate）。

键

CLUSTERKEYSLOT <key> 计算键 key 应该被放置在哪个槽上。

CLUSTERCOUNTKEYSINSLOT <slot> 返回槽 slot 目前包含的键值对数量。

CLUSTERGETKEYSINSLOT <slot> <count> 返回count 个 slot 槽中的键。

**手动故障转移**

有的时候在主节点没有任何问题的情况下强制手动故障转移也是很有必要的，比如想要升级主节点的Redis进程，我们可以通过故障转移将其转为slave再进行升级操作来避免对集群的可用性造成很大的影响。

Redis集群使用 CLUSTER FAILOVER命令来进行故障转移，不过要被转移的主节点的从节点上执行该命令

手动故障转移比主节点失败自动故障转移更加安全，因为手动故障转移时客户端的切换是在确保新的主节点完全复制了失败的旧的主节点数据的前提下下发生的，所以避免了数据的丢失。

查看所有节点

redis-cli -p 7000 cluster nodes

查看master节点

redis-cli -p 7000 cluster nodes | grep master

集群启动

 ./redis-trib.rb create --replicas 1 127.0.0.1:7000 127.0.0.1:7001127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005

添加集群从节点

./redis-trib.rb add-node --slave --master-id 3c3a0c74aae0b56170ccb03a76b60cfe7dc1912e 127.0.0.1:7006 127.0.0.1:7000

添加集群主节点

./redis-trib.rb add-node 127.0.0.1:7006 127.0.0.1:7000

**移除节点**

./redis-trib del-node 127.0.0.1:7000 <node-id>

对集群重新分片

./redis-trib.rb reshard 127.0.0.1:7000