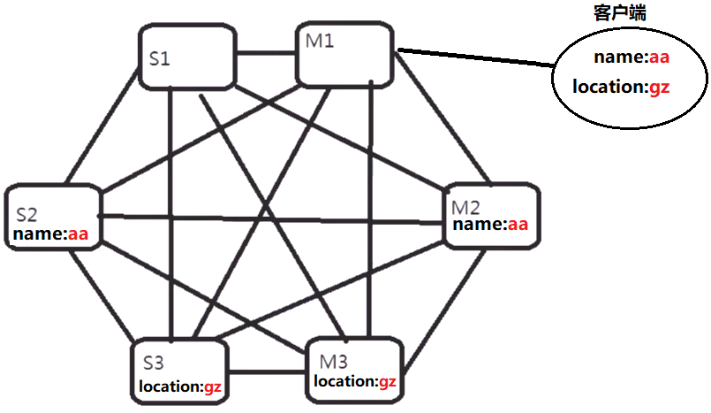
**Redis Cluster**

**1.介绍**

Redis3.0版本出现的一个新的redis集群结构,引入了很多新的概念和逻辑.将之前学习的主从复制.哨兵监听的逻辑整合到了这个新的结构当中.

**2.redis-cluster的集群结构**

****

**2.1集群结构中的特性和功能**

* 集群节点两两互联,底层依赖二进制传输协议的,优化传输速度，内部之间传输很快.(集群的基础)
* 集群的监控监听机制,交给主节点投票(哨兵集成到了集群的主节点中),任意的主节点宕机,只要剩下的主节点满足过半.
* 客户端连接任意一个节点,都可以向集群传递命令存储和获取数据,数据会在集群内部做节点的连接跳转.
* 引入了新的分布式分片计算逻辑--hash取模(取余,**CRC16()散**列计算);取模结果[0,..16383],这个范围区间,称之为槽道号--hash slot，每个master负责管理一批槽道号，由于槽道的存在,key值做hash取模运算,根据key的取模结果判断是否归哪个节点管理,从而做到了解耦,key和节点的解耦key--slot--node.

**2.2 Redis集群计算Key的算法**

* Redis集群计算Key的算法

<https://segmentfault.com/a/1190000018094567?utm_source=tag-newest>

**Key=CRC16(name)%16384=5798 所以key的范围0-16383**

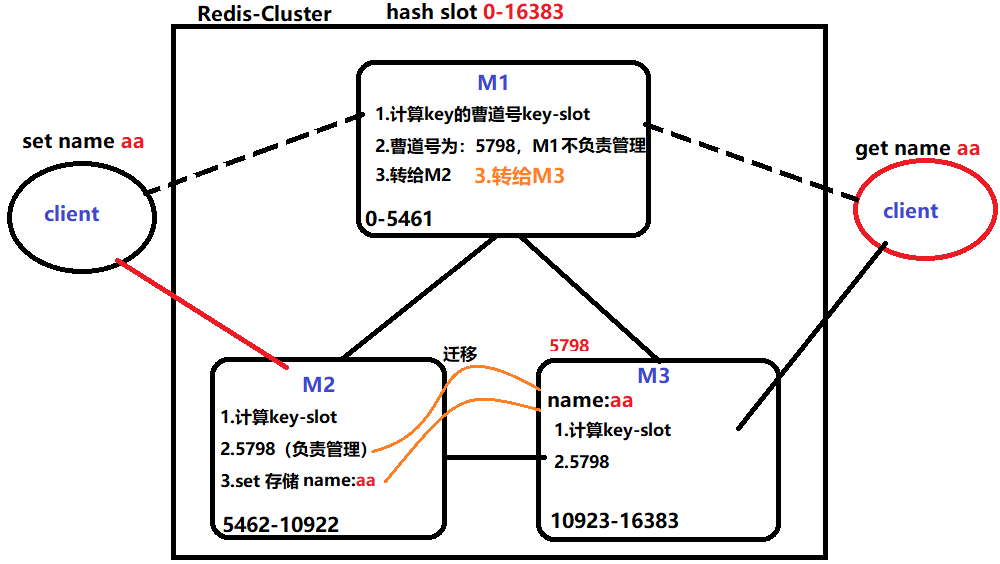
**Key= CRC16(location)%16384=7307**

* Redis之间进行数据交换的时候消息头所占空间为CLUSTER\_SLOTS/8，

也就是16384÷8÷1024=2kb

**2.3槽道管理的逻辑**

Redis Cluster要求一个集群最少需要3个主节点(最少的集群容忍度1,3个才能容忍宕一个),由于槽道的限制,理论上最多16384(官方版本提示1000节点左右)



每一个主节点都可以具备槽道管理权,也可以不具备槽道管理权(只具备投票权利)

**2.4槽道原理需要了解解决的问题**

* 计算CRC16()%16384,得到的整数，如何判断是否为当前节点的管理范围?
* 如果不是当前节点管理的槽道范围,如何知道正确的槽道管理者是谁?

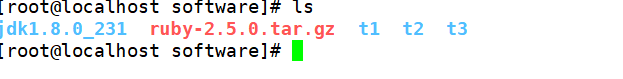
**3.利用ruby的脚本命令操作Redis集群**

**请按照视频的指引完成文档后面的实验。**

**3.1环境中需要具备ruby语言的环境**

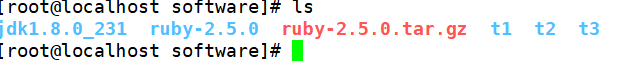
* **安装ruby语言**

1. 下载ruby-2.5.0.tar.gz，将ruby-2.5.0.tar.gz上传到/home/software下



2）解压ruby-2.5.0.tar.gz

**[root@localhost software]# tar -zxvf ruby-2.5.0.tar.gz**

****

**3）**./configure的意思是执行ruby根目录下面的configure文件

**[root@localhost software]# cd ruby-2.5.0**

**[root@localhost ruby-2.5.0]# ./configure**

4）执行make&&make install

**[root@localhost ruby-2.5.0]# make&&make install**

5）查看版本号

[root@localhost software]# ruby -v

ruby 2.5.0p0 (2017-12-25 revision 61468) [x86\_64-linux]

[root@localhost software]#

6）安裝Redis接口

[root@localhost software]# gem install redis

文本

描述已自动生成

**3.2准备一下搭建集群的配置文件环境**

* 每一个配置文件单独放在一个用端口号命名的文件夹里 8000/redis.conf 8001/redis.conf...
* 在redis根目录创建6个文件夹8000-8005，搭建集群需要6个节点对应6个文件夹

图片包含 图表

描述已自动生成

表格

中度可信度描述已自动生成

* 将Redis根目录下的redis.conf复制，复制一份redis-cluster节点启动的模板文件

[root@10-9-100-26 redis-3.2.11]# cp redis.conf **redis-cluster.conf**

* 打开redis-cluster.conf模板文件修改内容

其中我前面配置过的bind,protected-mode daimonize dump log 等不变(6379)

这里只需要配置与集群节点启动有关的内容.

**P595行**追加模式 aop的持久化策略，开启aop模式的持久化策略

**P598** 指定一个端口号对应的持久化文件

图片包含 图形用户界面

描述已自动生成

两种redis的持久化策略对比

|  |  |
| --- | --- |
| rdb | aof |
| 使用dump.rdb为默认的持久化文件,存储的是数据内容key-value | 一种追加模式的日志文件，存储的是命令的日志 |
| 可以一次性存储多条数据(执行效率高) | 默认每秒同步执行的写命令(执行效率低) |
| save时间间隔长,对数据的备份能力要求高时不推荐使用rdb | 备份数据时间比较短,最多最多,在极大的高并发先只会丢失1-2秒的数据 |
| 一次性存储数据过大时,有可能造成阻塞,影响使用 | 细水长流 |
| 数据量并发少的时候rdb性能更高 | 数据量并发到达一定程度aof反而性能高于rdb |

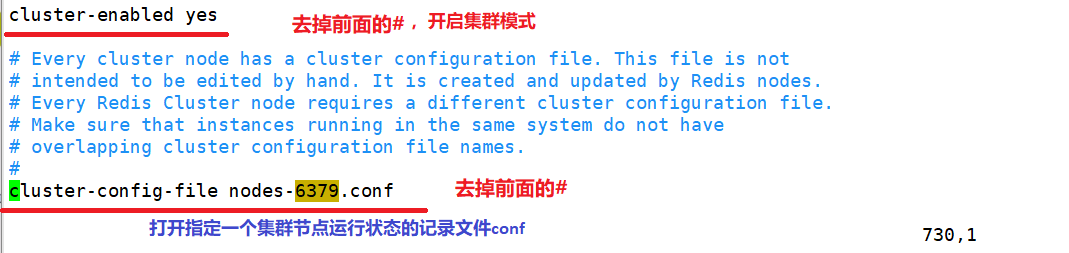
图片包含 图示

描述已自动生成

 如果同时开启aof和rdb,aof优先级高于rdb;重启之后最先加载了**appendOnly6379.aof;**

P723必须将节点启动定义为集群模式（开启集群模式）

P730 打开指定一个集群节点运行状态的记录文件conf



* 将模板文件拷贝到8000-8005的文件夹中,挨个修改端口号

[root@localhost redis-3.2.11]# cp redis-cluster.conf 8000/

文本

描述已自动生成

分别修改端口

[root@localhost redis-3.2.11]# vim 8000/redis-cluster.conf

:%s/6379/8000/g

[root@localhost redis-3.2.11]# vim 8001/redis-cluster.conf

:%s/6379/8001/g

[root@localhost redis-3.2.11]# vim 8002/redis-cluster.conf

:%s/6379/8002/g

[root@localhost redis-3.2.11]# vim 8003/redis-cluster.conf

:%s/6379/8003/g

[root@localhost redis-3.2.11]# vim 8004/redis-cluster.conf

:%s/6379/8004/g

[root@localhost redis-3.2.11]# vim 8005/redis-cluster.conf

:%s/6379/8005/g

图片包含 图表

描述已自动生成

* 启动所有节点

[root@localhost redis-3.2.11]# redis-server 8000/redis-cluster.conf

[root@localhost redis-3.2.11]# redis-server 8001/redis-cluster.conf

[root@localhost redis-3.2.11]# redis-server 8002/redis-cluster.conf

[root@localhost redis-3.2.11]# redis-server 8003/redis-cluster.conf

[root@localhost redis-3.2.11]# redis-server 8004/redis-cluster.conf

[root@localhost redis-3.2.11]# redis-server 8005/redis-cluster.conf

* 查看启动进程redis

文本

描述已自动生成

* 以集群模式启动的节点在搭建分配槽道之前(16384全部分配之前),无法直接操作。

文本

描述已自动生成

**3.3利用ruby脚本命令创建一个集群**

下面作一些测试练习,最终我们可以使用ruby脚本命令创建一个3主各自有一从的集群结构

* redis-trib.rb是使用ruby语言编写的具备很多操作集群底层命令的代码,利用它来实现创建集群的过程。

手机屏幕截图

描述已自动生成

文本

描述已自动生成

[root@localhost src]# ./redis-trib.rb

Usage: redis-trib <command> <options> <arguments ...>

create host1:port1 ... hostN:portN

--replicas <arg>

check host:port

info host:port

fix host:port

--timeout <arg>

reshard host:port

--from <arg>

--to <arg>

--slots <arg>

--yes

--timeout <arg>

--pipeline <arg>

rebalance host:port

--weight <arg>

--auto-weights

--use-empty-masters

--timeout <arg>

--simulate

--pipeline <arg>

--threshold <arg>

add-node new\_host:new\_port existing\_host:existing\_port

--slave

--master-id <arg>

del-node host:port node\_id

set-timeout host:port milliseconds

call host:port command arg arg .. arg

import host:port

--from <arg>

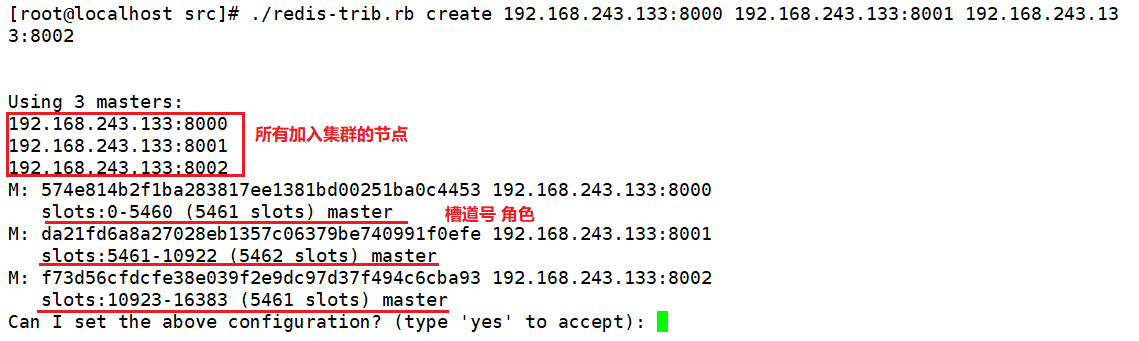
--copy

--replace

help (show this help)

* **使用ruby脚本搭建一个3主的结构**

[root@localhost src]# ./redis-trib.rb create 192.168.126.128:8000 192.168.126.128:8001 192.168.126.128:8002



输入yes搭建集群完毕(3个主节点),显示如下内容

图形用户界面, 文本

描述已自动生成

**16384个槽道分配完毕**

* **初次体验一下槽道逻辑**

操作集群节点,需要使用集群的登录模式

#redis-cli -c -p 8000,-c:以集群模式登录节点

登录8000,进行命令的输入

**#cluster info** 目前集群3个节点

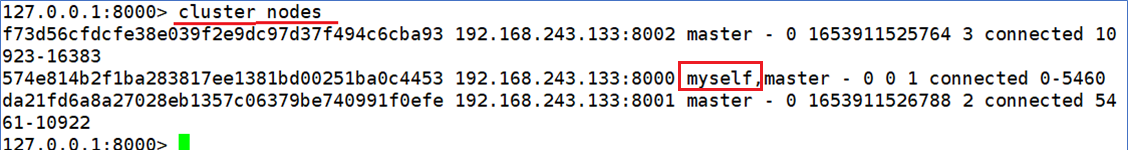
加入集群的节点，信息都是0：

文本

描述已自动生成

**#cluster nodes**

(显示的内容会保存在节点的node-800\*.conf文件中，重启时会根据文件中的节点信息来重新获取槽道等)



节点信息包含一下内容

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| f73d56cfdcfe38e039f2e9dc97d37f494c6cba93 | 192.168.126.128:8002 | master | - | 0 1653911525764 | 3 | connected | 10923-16383 |
| **节点id** | 节点host地址:端口 | 角色 | 节点对应的主节点id | 操作时间相关(时间戳，1970-1-1 0点到现在的秒数) | 序号 | 连接状态 | 管理槽道号 |

**8000后面的myself 表示当前查看信息的客户端信息**

将name=xu的key-value值set到8000的节点上

图片包含 图形用户界面

描述已自动生成

Key值为name的曹道号为5798，归8001节点管。通过槽道管理，不会造成数据和节点造成强耦合。

**3.4 动态添加一个节点**

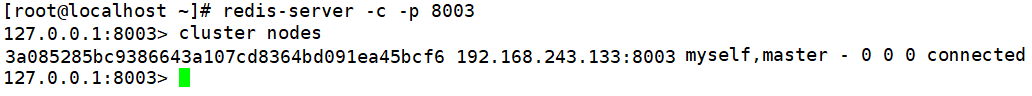
* **动态添加主节点**

已经搭建的3个主节点的集群,8000 8001 8002 正常运行,可以直接新增一个主节点;使用下面命令：

add-node 新节点ip:端口 已存在集群的任意节点ip:端口

**连接8003节点，可以查**

**看到8003还没加入到其他节点。**

****

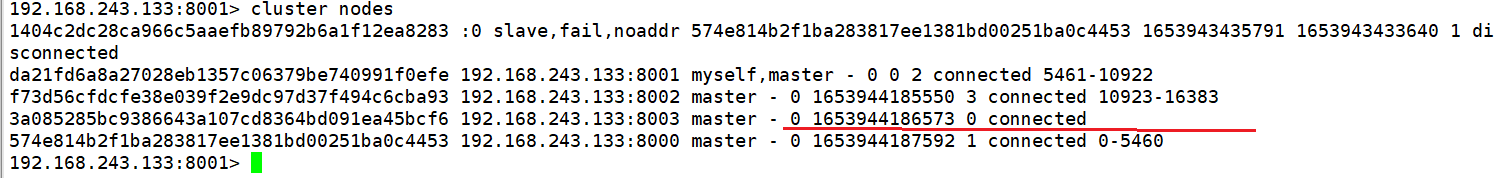
**添加节点：**

**[root@localhost src]# ./redis-trib.rb add-node 192.168.126.128:8003 192.168.126.128:8000**

电脑屏幕截图

描述已自动生成

查看节点信息

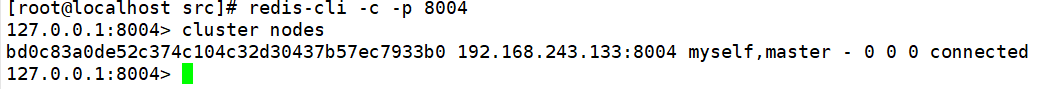


可以看到8003并没有获得任何槽道管理权

* **动态添加从节点**

add-node --slave(以**从节点角色**加入集群) --master-id 参数(主节点的id) 新节点ip:端口 已存在节点ip:端口

**先查看8004，目前没有加入到集群**



上面实验中可以查询到**8001**主节点的id值为：“da21fd6a8a27028eb1357c06379be740991f0efe”

8004挂接到8001;

下面命令中的“da21fd6a8a27028eb1357c06379be740991f0efe”值是8001主节点的id。

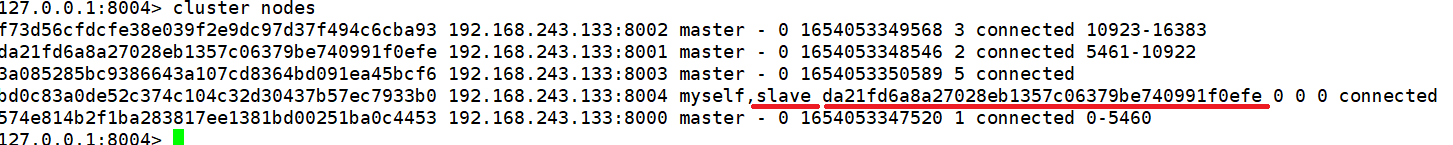
[root@localhost src]# ./redis-trib.rb add-node --slave --master-id da21fd6a8a27028eb1357c06379be740991f0efe 192.168.126.128:8004 192.168.126.128:8003

以slave角色来添加8004节点，将8004节点挂接到8001（id: da21fd6a8a27028eb1357c06379be740991f0efe）,挂接或加入时通过8003来通知集群的其他节点。

文本

中度可信度描述已自动生成

登录8004验证主从复制结构是否正确



测试：在8004端口中set 数据

文本

低可信度描述已自动生成

Key值为location的槽道号为7307，归8001节点管。

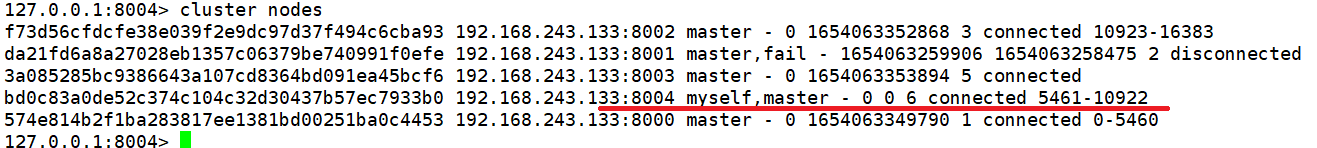
测试结果，8004正确挂接在8001节点下

* **直接验证集群结构的高可用**

1. 将主节点8001干掉,观察8004的顶替情况

手机屏幕截图

描述已自动生成



不仅顶替成功成为Master，还获取了槽道管理权

1. 顶替后是否能获取name 和location

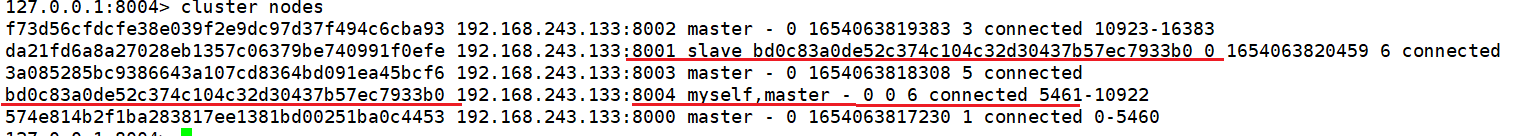
文本

中度可信度描述已自动生成

顶替后正确继承槽道管理权5461-10922，8004正确顶替,正确运行功能

* **8001重启观察角色(slave)**

重新挂接到8004作为从节点继续提供服务



**3.5槽道迁移**

将8002中的200个槽道交给8003管理

reshard 已存在集群的任一节点ip:端口

根据问题,作答.

[root@localhost src]# ./redis-trib.rb reshard 192.168.126.128:8001

回答问题1:迁移的槽道数量(没法迁移非空槽道 5798 7307两个槽道非空，无法迁移)

图片包含 徽标

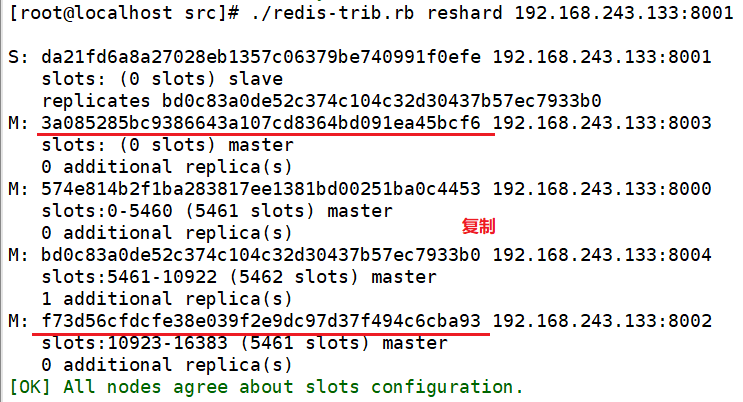
描述已自动生成

回答问题2:接收迁移槽道的节点id(8003的id)

图片包含 图表

描述已自动生成

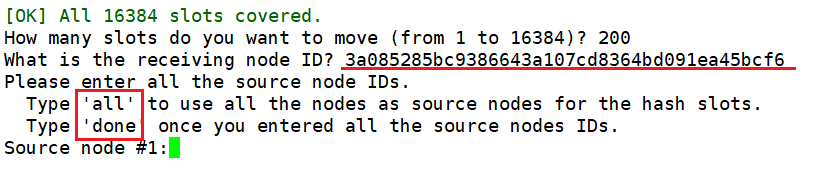
复制8003节点的id:



文本

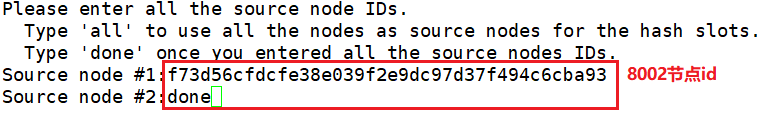
中度可信度描述已自动生成

回答问题3:选择槽道的来源(所有槽道均分all,指定节点提供槽道输入节点id,完毕后输入done)



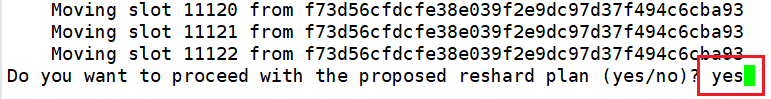
这里使用8002做槽道来源(200个槽道都从8002迁移)，指定多个节点表示从多个节点迁移

复制8002节点的id:



回答问题4:是否同意迁移的设置

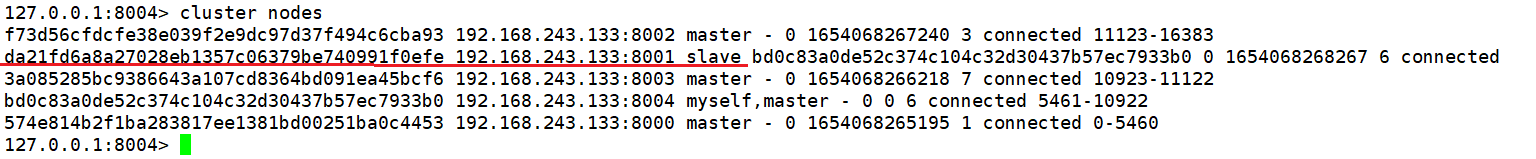
显示即将迁移的槽道。



reshard不支持非空迁移(有数据的槽道),可以使用cluster集群提供的底层命令一步一步完成有数据的迁移(数据微调)

**3.6删除节点**

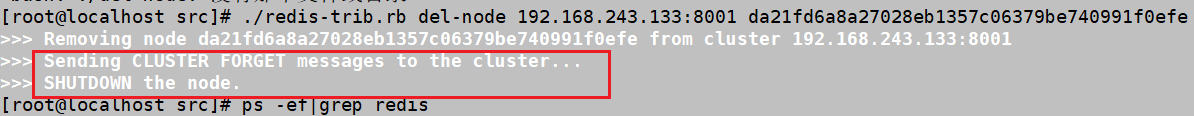
只能删除**从节点**或者**没有槽道管理权的主节点**,一旦删除,为了不影响集群使用,删除的节点自动关闭;**如果**删除的节点重新启动想重新添加到集群(**做不到**);



节点信息，其他节点都是主节点，并且都有槽道管理权，只能删除8001节点。

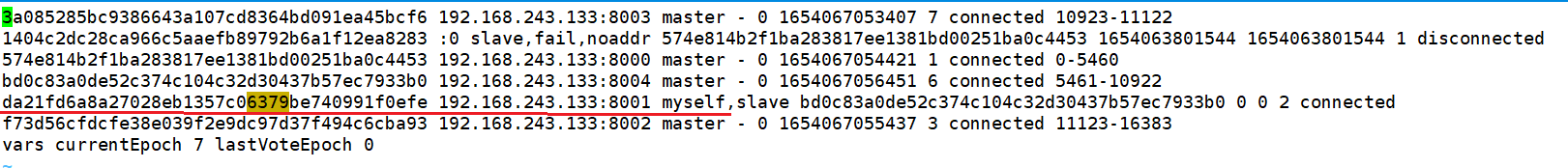
del-node 集群节点ip:端口 删除节点的id值

[root@localhost src]# ./redis-trib.rb **del-node** 192.168.126.128:8001 **da21fd6a8a27028eb1357c06379be740991f0efe**



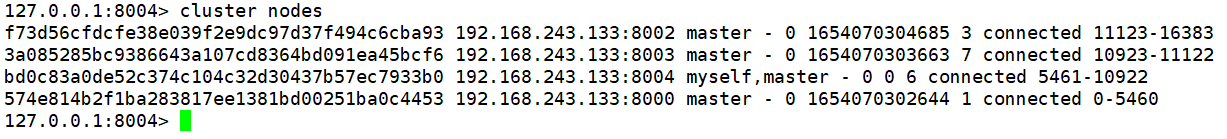
8001不仅被删除，还被关闭，关闭的原因,是因为删除的节点不会修改配置文件node-8001.conf,8001自己以为节点启动,依然会以集群的角色出现,但是其他节点都不知道8001。

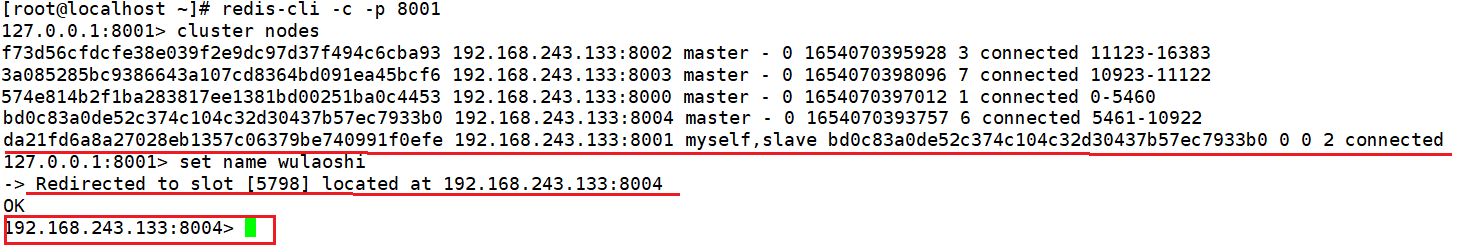
查看nodes-8001.conf，可以看到8001经过删除和关闭后，还认为自己在集群中。



如果不关闭，重启8001.

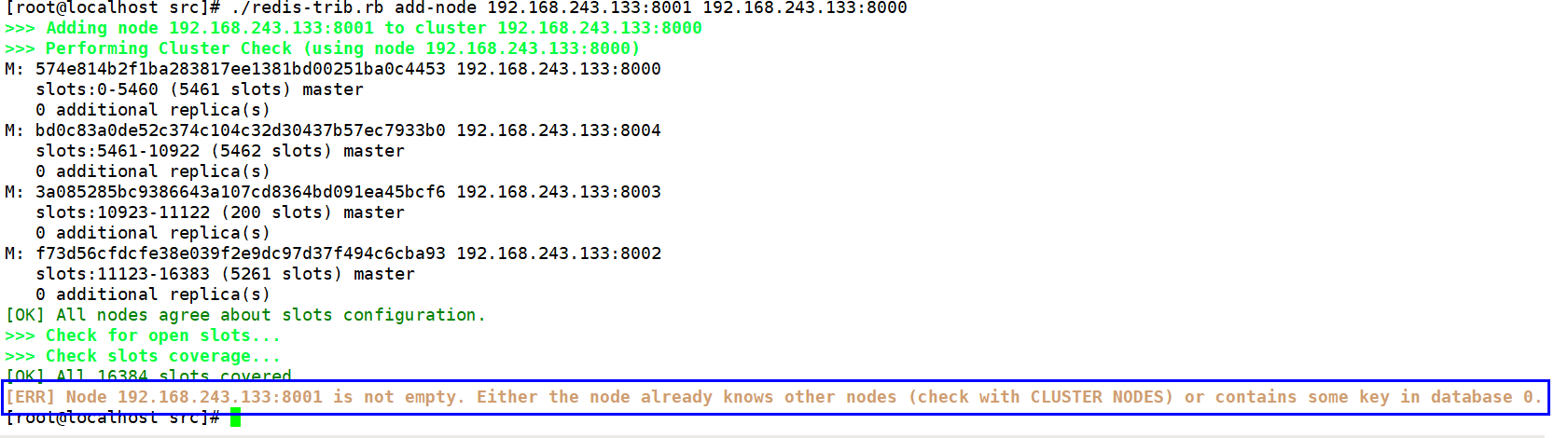
在8004节点查看，看不到8001，已经被删除



在8001自己节点查看，还看到8001，自己以为自己还在集群里面

再次将8001节点添加到集群，是碰到的问题:

[root@localhost src]# redis-trib.rb add-node 192.168.126.128:8001 192.168.126.128:8000



提示的问题:当前节点无法加入集群,不是节点有数据.就是节点已经被集群加入了.

删除的8001,想以新节点角色加入;

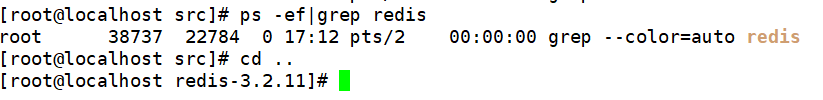
加入之前,删除node文件.删除dump文件，.aof也删除，重新启动

**4.重新搭建集群(测试环境)**

集群运行坏了的时候，重新搭建的步骤

停/删/启/建

* 将所有节点kill/shutdown



* 删除所有持久化文件（测试环境）
  + node\*
  + dump\*
  + append\*

图标

描述已自动生成

* 重新启动各自节点

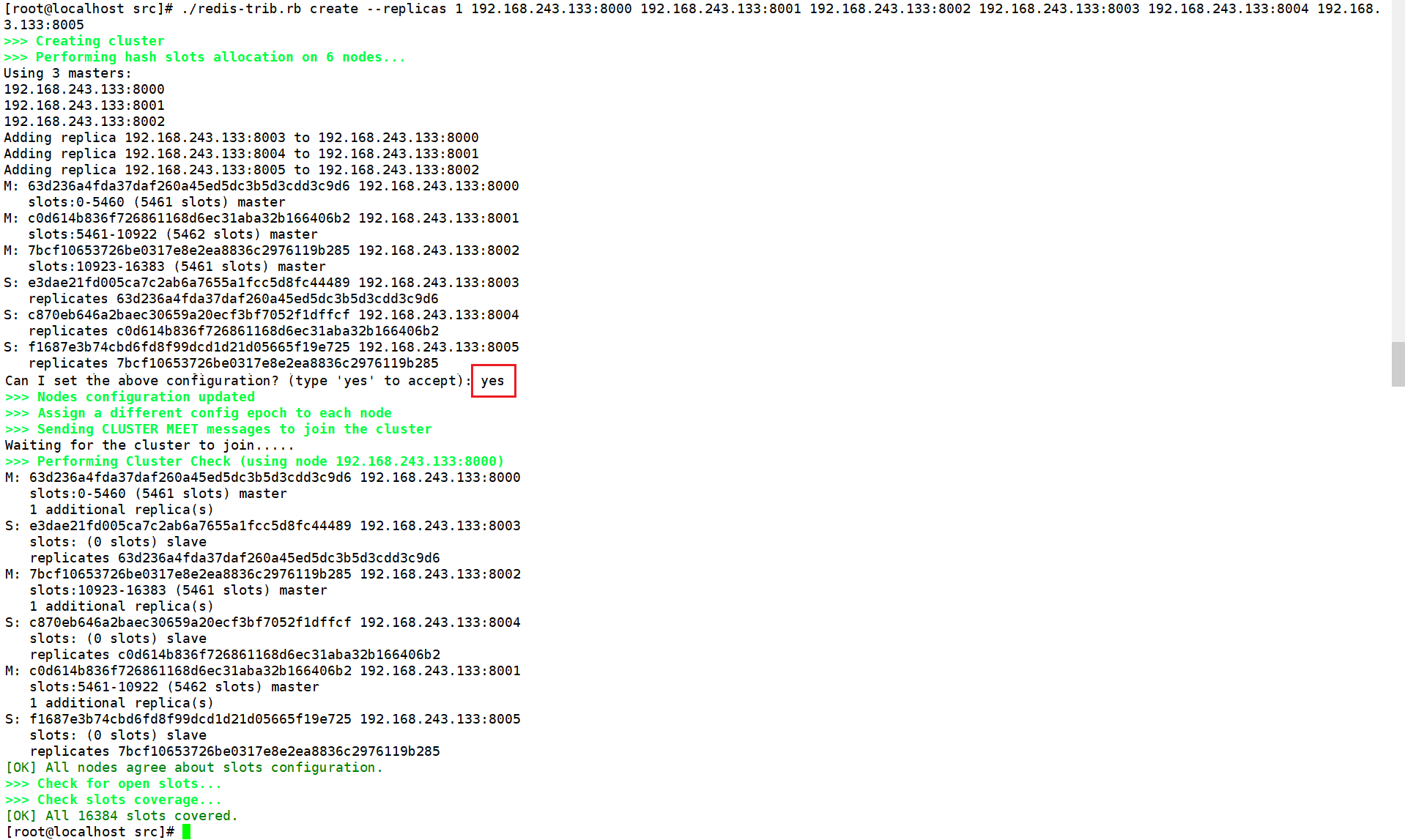
文本

描述已自动生成

* ./redis-trib.rb create 重新搭建 --replicas 1

[root@localhost src]# ./redis-trib.rb create **--replicas** 1 192.168.126.128:8000 192.168.126.128:8001 192.168.126.128:8002 192.168.126.128:8003 192.168.126.128:8004 192.168.3.133:8005

以主从结构的集群来搭建cluster,1每个主节点下至少配置一个从节点(6个节点 正好三主各自有一从)



连接任一节点查看nodes信息：正好一主一从

