**下面的练习目的是为了加深对Redis槽道原理的理解。**

**1.将集群清空**

将已经创建的集群.恢复到初始状态

**1.1停**

**redis-cli -c -p 8000 shutdown**

[root@localhost redis-3.2.11]# redis-cli -c -p 8000 shutdown

[root@localhost redis-3.2.11]# redis-cli -c -p 8001 shutdown

[root@localhost redis-3.2.11]# redis-cli -c -p 8002 shutdown

[root@localhost redis-3.2.11]# redis-cli -c -p 8003 shutdown

[root@localhost redis-3.2.11]# redis-cli -c -p 8004 shutdown

[root@localhost redis-3.2.11]# redis-cli -c -p 8005 shutdown

**1.2删除持久化文件**

node文件,dump文件,append文件

[root@localhost redis-3.2.11]# rm -rf appendonly800\*

[root@localhost redis-3.2.11]# rm -rf dump800\*

[root@localhost redis-3.2.11]# rm -rf nodes-800\*

文本

描述已自动生成

**1.3启动所有节点(初始化的6个节点)**

[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 8003/redis-cluster.conf

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

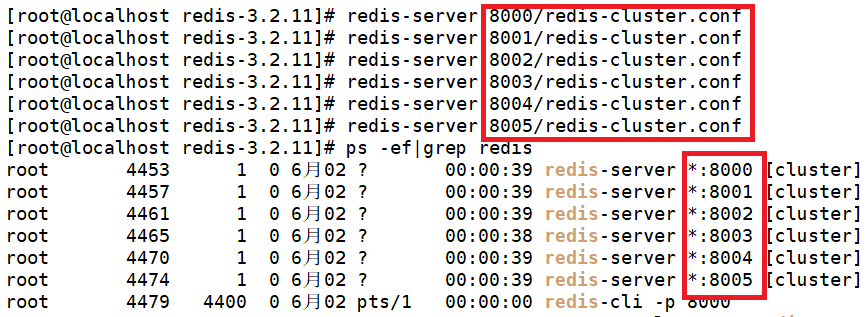
**2.操作redis的cluster命令完成测试**

**2.1手动搭建集群**

手动搭建:调用集群命令cluster meet 节点碰撞

**注意:手动搭建集群,登录时用-h携带对外访问地址**

* **启动节点 8000-8005全启动**



* **节点握手**

登录任何启动的节点; >cluster meet ip port(ip 端口是其他节点信息)

注意:**跨服务器**执行集群的搭建,需要登录时使用对外访问ip地址;

**打开新的窗口**

**[root@localhost ~]# cd /home/software/redis-3.2.11**

**[root@localhost redis-3.2.11]# redis-cli -c -p 8000 -h 192.168.126.128**

**192.168.126.128:8000> cluster meet 192.168.126.128 8001**

**OK**

**添加其他节点到集群**

图片包含 室内, 桌子, 瓶子, 照片

描述已自动生成

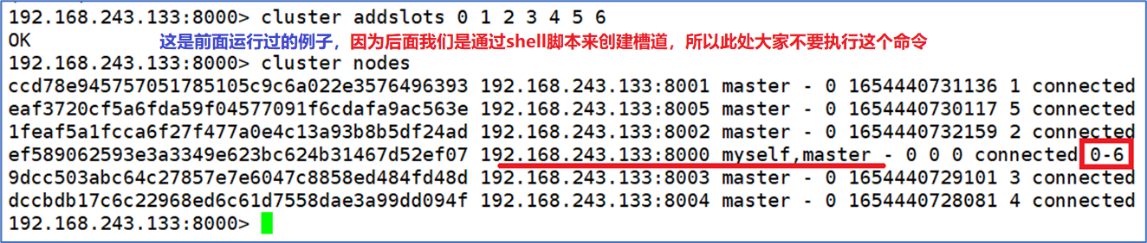
观察集群节点，此时所有节点都还没分配槽道

文本

描述已自动生成

下面将8000-8002设为主节点,8003-8005为从节点。

* 先为8000-8002三个主节点分配槽道
* 8000>cluster addslots 槽道号 :可以1个,也可以多个用空格隔开

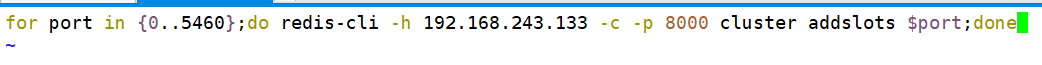


我们可以通过shell脚本来完成槽道的分配,编写脚本文件8000slots.sh;

[root@localhost redis-3.2.11]# vim 8000slots.sh

代码内容：（**此处不要复制，复制容易出错**）

for port in {0..5460};do redis-cli -h 192.168.126.128 -c -p 8000 cluster addslots $port;done

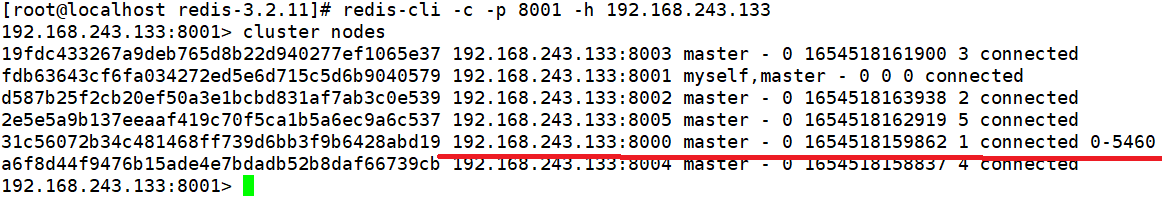
执行shell脚本，得到5461个OK，表示8000节点的槽道分配完毕。

[root@localhost redis-3.2.11]# sh 8000slots.sh

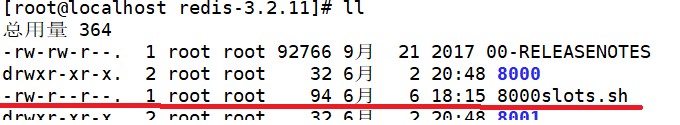
图片包含 图形用户界面

描述已自动生成

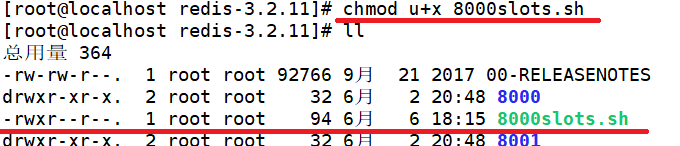
查看集群节点信息,发现0-5460槽道分配完毕



**Q&A:**运行了脚本如果没有任何反应，cluster nodes 也查看不到槽道，请检查shell脚本没有打错，确定没打错，则有可能是不够权限的问题，查看：



**发现没有x权限（执行权限）**，添加x权限后，再次执行8000slots.sh



addslot执行完成后,**当前节点和其他节点**的**索引数组**/**共享数组**发生变动,0~5460号元素内容替换为8000的节点信息；

二进制数组中，除了8000节点的0~5460号元素内容替换为1，其他节点没变化。

**编辑8001的shell脚本,执行脚本,完成 8001(5461-10922)的槽道分配**

对应的shell脚本为：

for port in {5461..10922};do redis-cli -h 192.168.126.128 -c -p 8001 cluster addslots $port;done



查看：

图片包含 日历

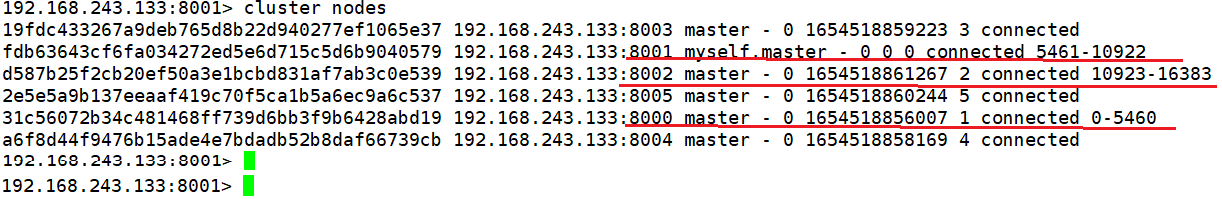
描述已自动生成

**编辑8002的shell脚本,执行脚本,完成8002(10923-16383)的槽道分配**

for port in {10923..16383};do redis-cli -h 192.168.126.128 -c -p 8002 cluster addslots $port;done



查看：



查看集群状态,ok

>cluster info

文本

描述已自动生成

成功执行set命令。

图片包含 文本

描述已自动生成

**2.2手动挂接从节点**

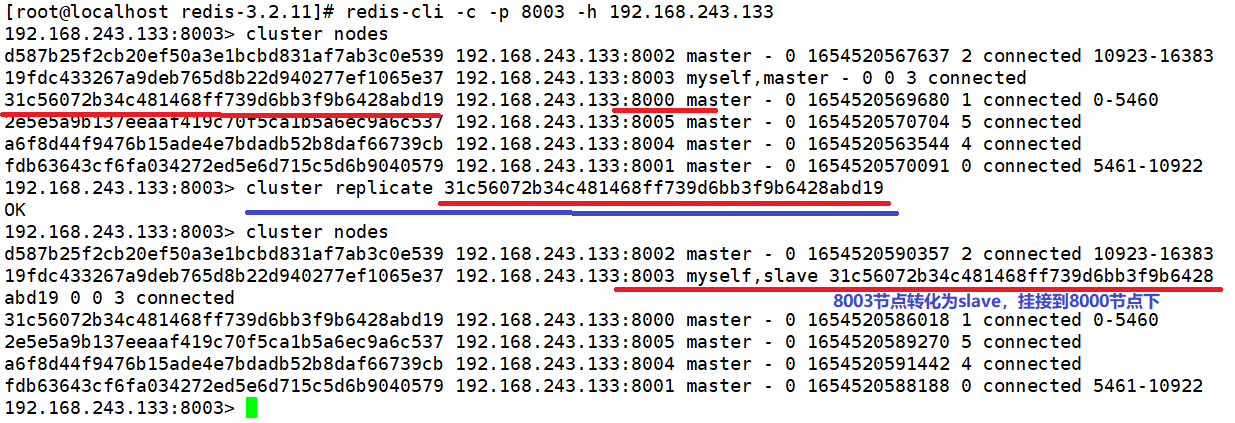
meet进来的节点都是master节点，需要对其做角色转换。

下面将8003挂接到8000节点

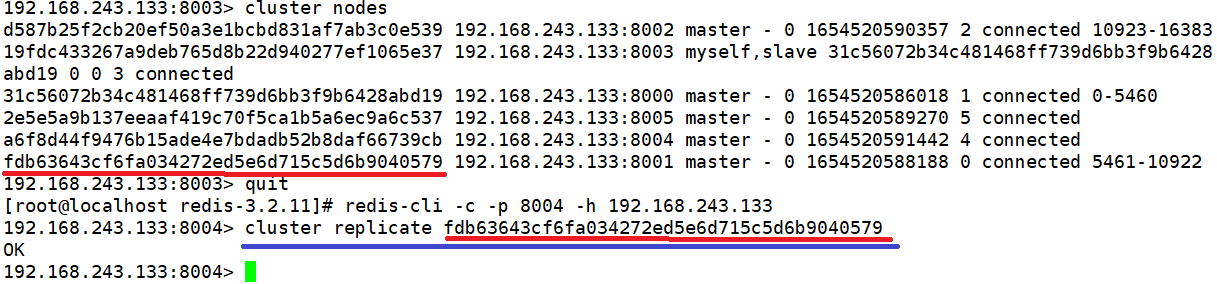
先登录到8003,然后调用命令将其**添加为8000**的从节点

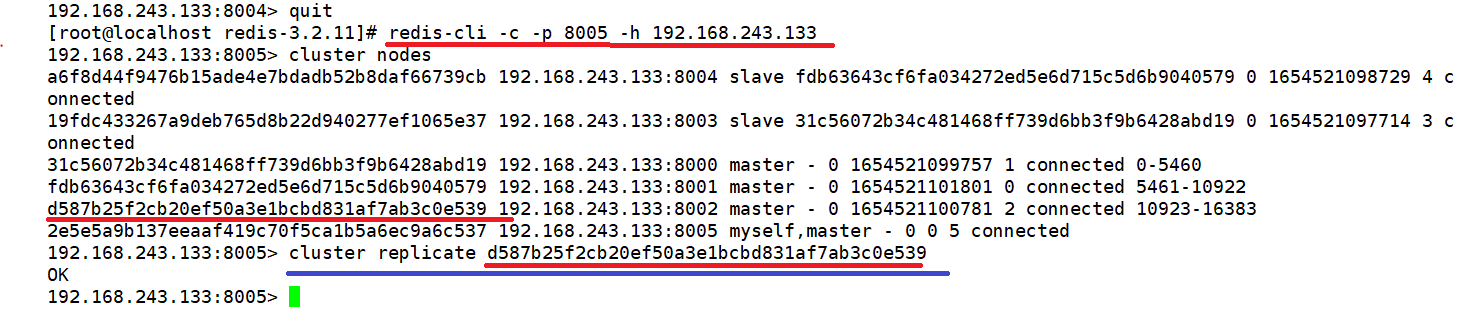
[root@localhost redis-3.2.11]# redis-cli -c -p 8003 -h 192.168.126.128

192.168.126.128:8003> cluster replicate 31c56072b34c481468ff739d6bb3f9b6428abd19

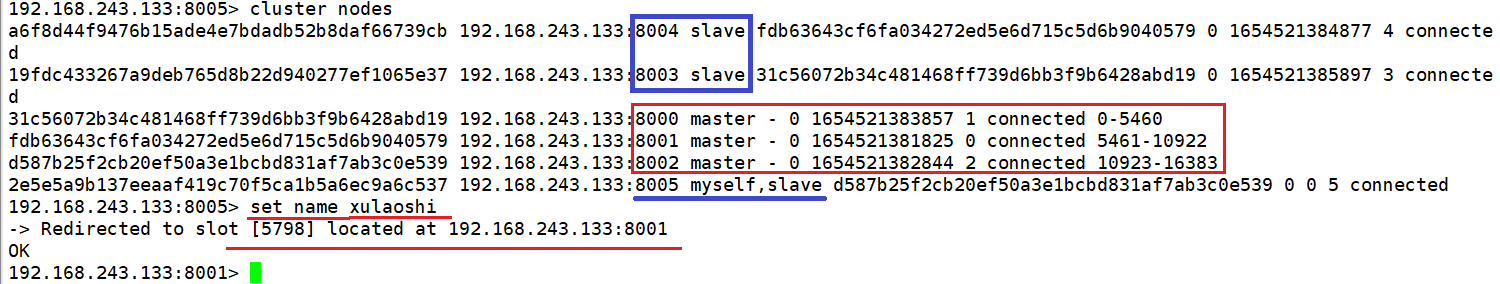


请按同样的步骤将8004节点挂接在8001上，8005节点挂接在8002上面。





测试集群



总结:redis-trib.rb 命令,就是利用ruby语言完成上述步骤的封装;对外提供方便的调用命令;

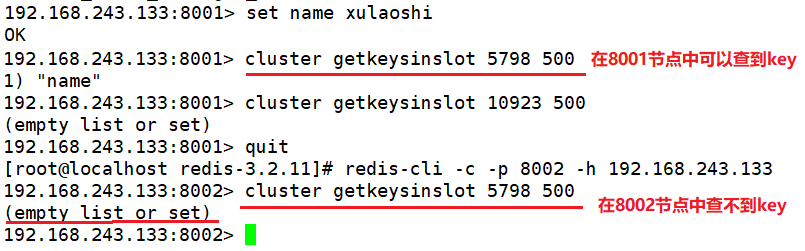
**3.槽道的迁移(数据微调)**

**3.1空槽道迁移(将8002节点中从10923槽道迁移到8001节点)**

分4个步骤完成：

**1）判断是否为空槽道**

判空命令必须**在管理该槽道的节点**上运行，否则判断不准确。例如我们执行set name xulaoshi命令。在8001查看5798槽道是否为空，查到不为空，到8002查看，查到的是empty，**所以必须在槽道坐在节点查看**。



下面检查从10923 槽道开始，后面的500个范围是否为空

由于10923是归8002管，必须在8002节点查看。

192.168.126.128:8002> cluster getkeysinslot 10923 500

图形用户界面, 文本

描述已自动生成

**2）目标节点设置导入槽道**

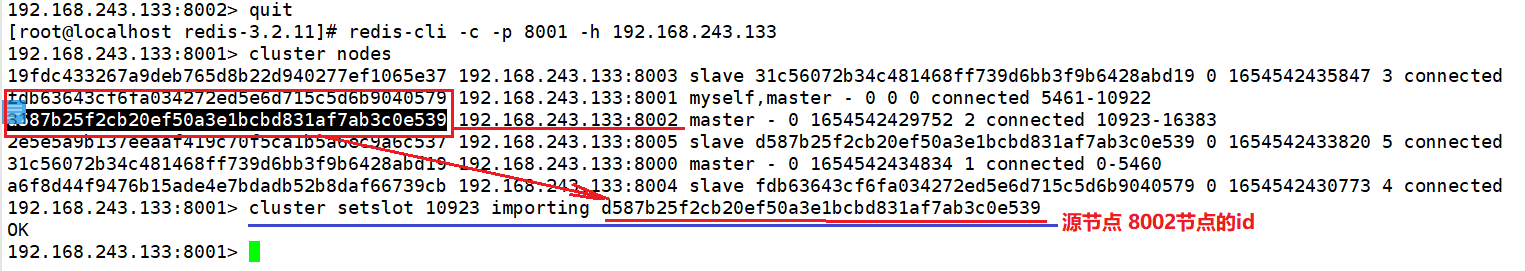
在目标节点(8001)设置导入槽道（10923）为导入状态(目标节点中二进制不变, 数组当前槽道的状态变为导入)

槽道有四种状态：正常状态是stable ,导入状态importing, 导出状态migrating,删除状态deleted

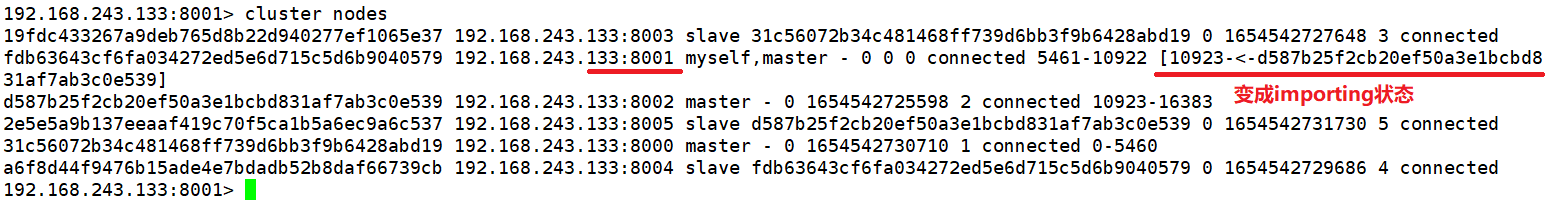
cluster setslot 10923 importing 源节点id

登录8001节点,执行命令,10923是操作的槽道,importing将8001上的数组的10923元素中的槽道状态修改为导入importing，源节点id 是8002的id

192.168.126.128:8001> cluster setslot 10923 importing d587b25f2cb20ef50a3e1bcbd831af7ab3c0e539



从8001检查当前集群的节点状态：



**3）源节点(8002)设置导出槽道(源节点二进制不变,数组槽道状态变为导出)**

在源节点(8002)设置导出槽道10923为导出状态(源节点二进制不变,数组槽道状态变为导出)

登录8002执行

cluster setslot 10923 migrating 目标节点id (8001节点id)

192.168.126.128:8002> cluster setslot 10923 migrating fdb63643cf6fa034272ed5e6d715c5d6b9040579

执行命令，并通过cluster nodes观察到8002的信息也发生变化

槽道的状态是正常,导入,导出,都不影响数据的插入

数据微调的时候,集群不能对外提供服务;数据迁移工作,在维护内容属于重大维护事项;

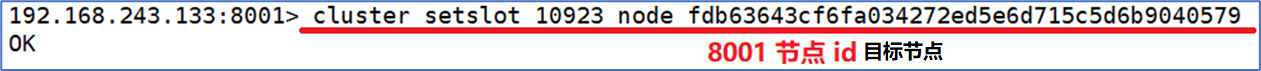
**4)通知所有集群过半主节点（包括当事人主节点）槽道迁移了(所有的节点的索引数组发生变化)**

cluster setslot 10923 node 目标节点8001节点id （**过半主节点**都执行这个操作）

通知8001,10923从现在开始,归8001所有（**在8001节点执行）**

192.168.126.128:8001> cluster setslot 10923 node fdb63643cf6fa034272ed5e6d715c5d6b9040579

(8001二进制10923下标的二进制从0变成1,数组变了)



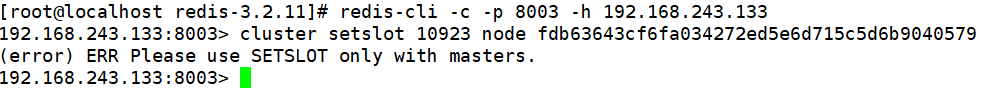
通知8002，10923从现在开始,归8001所有（**在8002节点执行**）

192.168.126.128:8002> cluster setslot 10923 node fdb63643cf6fa034272ed5e6d715c5d6b9040579

(8002二进制10923下标的二进制从0变成1,数组变了)



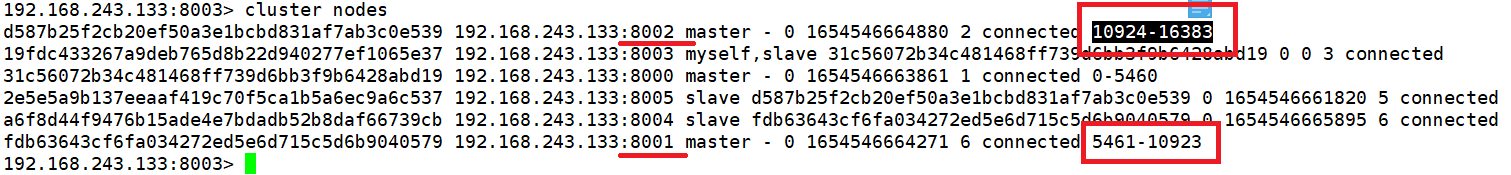
**在8003上执行操作,提示只能在master执行更新**



**在8003上执行操作,提示只能在master执行更新，**

**从节点同步主节点的信息,包括数组,和数据的备份**

**检查集群节点状态，空槽道迁移成功了**

****

**3.2非空槽道迁移(reshard命令不支持)**

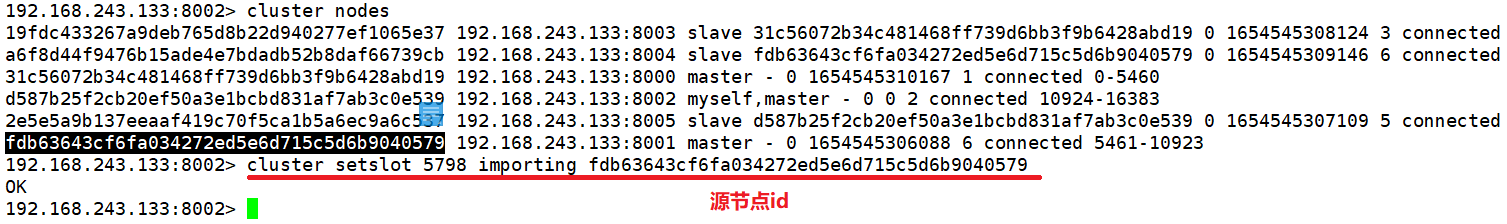
1）判断5798槽道是否非空

图片包含 文本

描述已自动生成

name 存储在5798上,迁移5798(**源节点8001**)=>**目标节点8002**

2）目标节点导入槽道 变为 importing



192.168.126.128:8002> cluster setslot 5798 importing fdb63643cf6fa034272ed5e6d715c5d6b9040579

3) 源节点导出槽道 变为migrating状态



192.168.126.128:8001> cluster setslot 5798 migrating d587b25f2cb20ef50a3e1bcbd831af7ab3c0e539

4) 将槽道的数据迁移到目标节点(登录源节点迁移数据)

* + **确定当前槽道包含的所有key(5798)**

192.168.126.128:8001> cluster getkeysinslot 5798 500

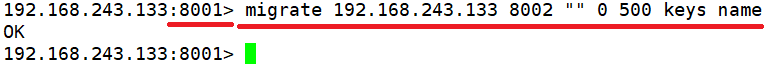
查看5798上500个范围内的所有key;

文本

低可信度描述已自动生成

* + **将获取的key值,进行迁移,从8001迁移到8002**

192.168.126.128:8001> migrate 192.168.126.128 8002 "" 0 500 keys name

****

"": 表示key的匹配; 可以使用正则(有待考证)

0: 表示迁移到目标节点的database(0-15) （片区）

500:连接超时毫秒

keys:标签确定迁移的key值名称

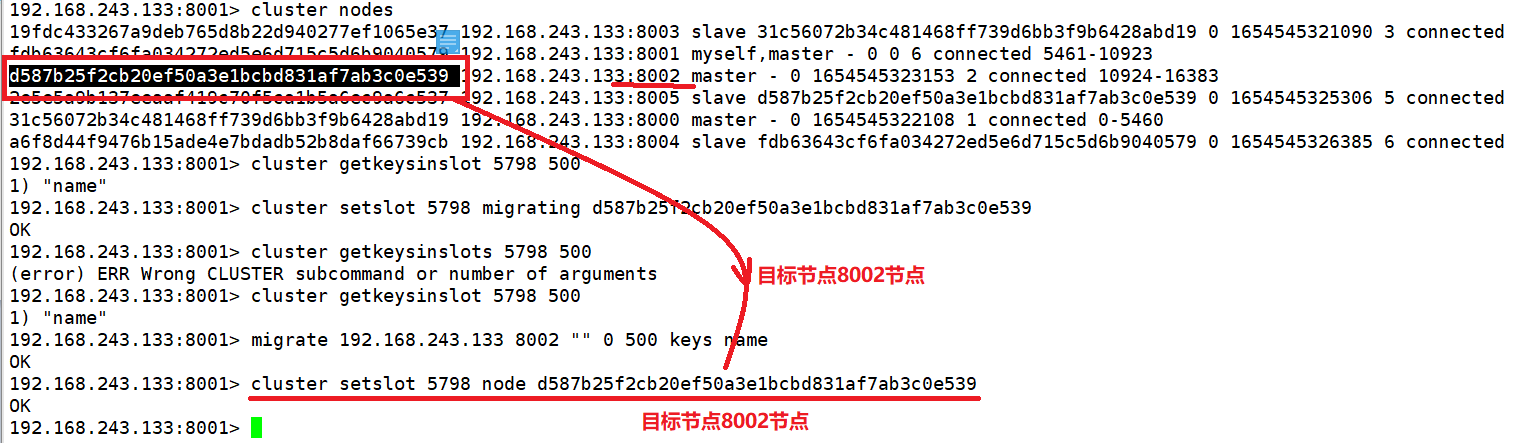
name:key

如果有多个key:keys name1 name2 name3 name4

除了key-value执行迁移,map记录{槽道:[key集合]}的内容一并迁移

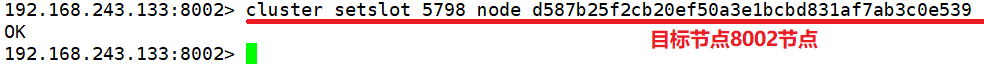
* + **通知过当事人节点（8001和8002）更新数据(槽道迁移)**

通知8001节点

****

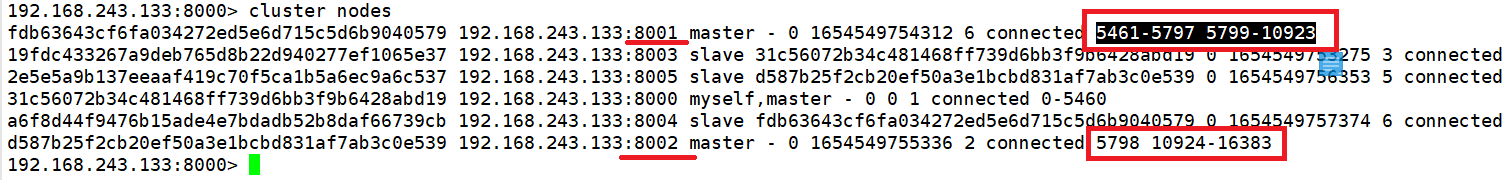
192.168.126.128:8001> cluster setslot 5798 node d587b25f2cb20ef50a3e1bcbd831af7ab3c0e539

通知8002节点

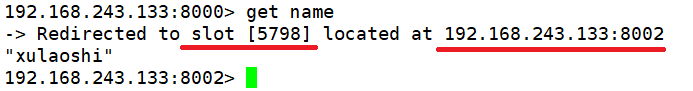


192.168.126.128:8002> cluster setslot 5798 node d587b25f2cb20ef50a3e1bcbd831af7ab3c0e539

查看nodes



测试，获取get数据：



**4.命令**

**4.1 集群命令**

cluster info ：打印集群的信息

cluster nodes ：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。

**4.2节点**

cluster meet <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。

cluster forget <node\_id> ：从集群中移除 node\_id 指定的节点(保证空槽道)。

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

cluster saveconfig ：将节点的配置文件保存到硬盘里面。

**4.3槽(slot)**

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

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

cluster flushslots ：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。

cluster setslot <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。

cluster setslot <slot> migrating <node\_id> ：将本节点的槽 slot 迁移到 node\_id 指定的节点中。

cluster setslot <slot> importing <node\_id> ：从 node\_id 指定的节点中导入槽 slot 到本节点。

cluster setslot <slot> stable ：取消对槽 slot 的导入（ import）或者迁移（ migrate）。

**4.5键**

cluster keyslot <key> ：计算键 key 应该被放置在哪个槽上。

cluster countkeysinslot <slot> ：返回槽 slot 目前包含的键值对数量。

cluster getkeysinslot <slot> <count> ：返回 count 个 slot 槽中的键