1、主从环境搭建

注意防火墙还要开放集群总线端口，就是redis端口号+10000

1.1、3主3从

```
docker run -d --name=redis-node-1 --net host --privileged=true -v
/data/redis/share/redis-node-1:/data redis --cluster-enabled yes --appendonly
yes --port 6381

docker run -d --name=redis-node-2 --net host --privileged=true -v
/data/redis/share/redis-node-2:/data redis --cluster-enabled yes --appendonly
yes --port 6382

docker run -d --name=redis-node-3 --net host --privileged=true -v
/data/redis/share/redis-node-3:/data redis --cluster-enabled yes --appendonly
yes --port 6383

docker run -d --name=redis-node-4 --net host --privileged=true -v
/data/redis/share/redis-node-4:/data redis --cluster-enabled yes --appendonly
yes --port 6384

docker run -d --name=redis-node-5 --net host --privileged=true -v
/data/redis/share/redis-node-5:/data redis --cluster-enabled yes --appendonly
yes --port 6385

docker run -d --name=redis-node-6 --net host --privileged=true -v
/data/redis/share/redis-node-6:/data redis --cluster-enabled yes --appendonly
yes --port 6386
```

宿主机输入`ifconfig`可得`172.19.212.192`的`ip`

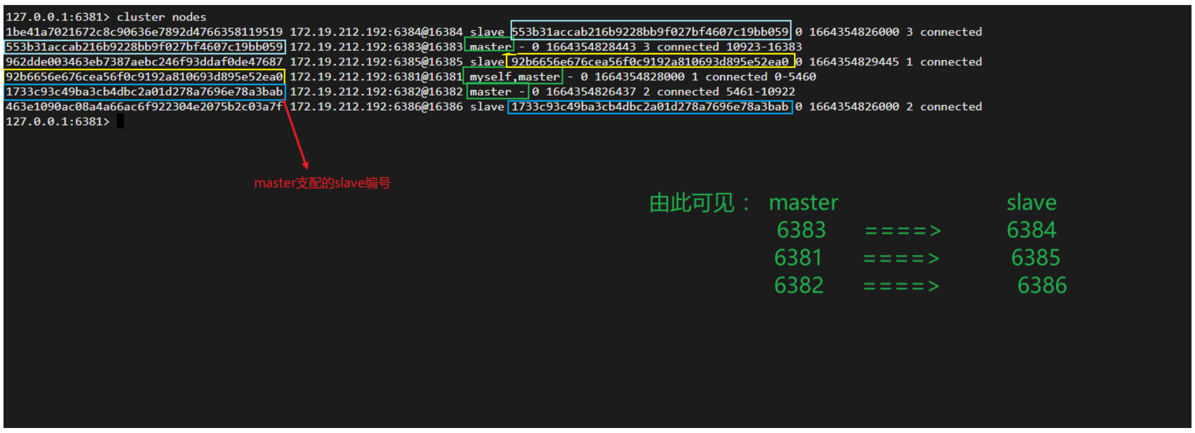
```
docker exec -it redis-node-1 /bin/bash

redis-cli --cluster create 172.19.212.192:6381 172.19.212.192:6382
172.19.212.192:6383 172.19.212.192:6384 172.19.212.192:6385 172.19.212.192:6386
--cluster-replicas 1
```

成功

```
docker exec -it redis-node-1 /bin/bash
redis-cli -p 6381 # 单机模式下查看



127.0.0.1:6381> cluster info
cluster_state:ok
cluster_slots_assigned:16384
cluster_slots_ok:16384
```

```
cluster_slots_pfail:0
cluster_slots_fail:0
cluster_known_nodes:6
cluster_size:3
cluster_current_epoch:6
cluster_my_epoch:1
cluster_stats_messages_ping_sent:214
cluster_stats_messages_pong_sent:216
cluster_stats_messages_sent:430
cluster_stats_messages_ping_received:211
cluster_stats_messages_pong_received:214
cluster_stats_messages_meet_received:5
cluster_stats_messages_received:430
127.0.0.1:6381>


127.0.0.1:6381> cluster nodes
1be41a7021672c8c90636e7892d4766358119519 172.19.212.192:6384@16384 slave
553b31accab216b9228bb9f027bf4607c19bb059 0 1664354826000 3 connected
553b31accab216b9228bb9f027bf4607c19bb059 172.19.212.192:6383@16383 master - 0
1664354828443 3 connected 10923-16383
962dde003463eb7387aebc246f93ddaf0de47687 172.19.212.192:6385@16385 slave
92b6656e676cea56f0c9192a810693d895e52ea0 0 1664354829445 1 connected
92b6656e676cea56f0c9192a810693d895e52ea0 172.19.212.192:6381@16381 myself,master
- 0 1664354828000 1 connected 0-5460
1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 172.19.212.192:6382@16382 master - 0
1664354826437 2 connected 5461-10922
463e1090ac08a4a66ac6f922304e2075b2c03a7f 172.19.212.192:6386@16386 slave
1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 0 1664354826000 2 connected
127.0.0.1:6381> info replication
```

`cluster nodes`

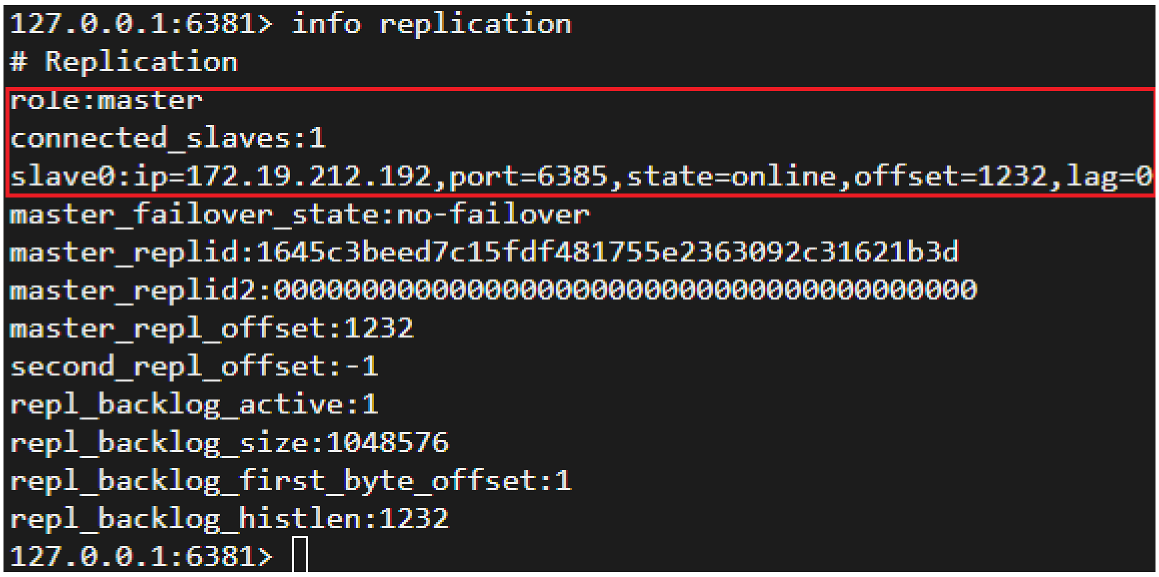


```
127.0.0.1:6381> cluster nodes
1be41a7021672c8c90636e7892d4766358119519 172.19.212.192:6384@16384 slave 553b31accab216b9228bb9f027bf4607c19bb059 0 1664354826000 3 connected
553b31accab216b9228bb9f027bf4607c19bb059 172.19.212.192:6383@16383 master - 0 1664354828443 3 connected 10923-16383
962dde003463eb7387aebc246f93ddaf0de47687 172.19.212.192:6385@16385 slave 92b6656e676cea56f0c9192a810693d895e52ea0 0 1664354829445 1 connected
92b6656e676cea56f0c9192a810693d895e52ea0 172.19.212.192:6381@16381 myself,master - 0 1664354828000 1 connected 0-5460
1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 172.19.212.192:6382@16382 master - 0 1664354826437 2 connected 5461-10922
463e1090ac08a4a66ac6f922304e2075b2c03a7f 172.19.212.192:6386@16386 slave 1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 0 1664354826000 2 connected
127.0.0.1:6381>
```

master支配的slave编号

由此可见： master ====> slave
6383 ====> 6384
6381 ====> 6385
6382 ====> 6386

`info replication`



```
127.0.0.1:6381> info replication
# Replication
role:master
connected_slaves:1
slave0:ip=172.19.212.192,port=6385,state=online,offset=1232,lag=0
master_failover_state:no-failover
master_replid:1645c3beed7c15fdf481755e2363092c31621b3d
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:1232
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:1232
127.0.0.1:6381>
```

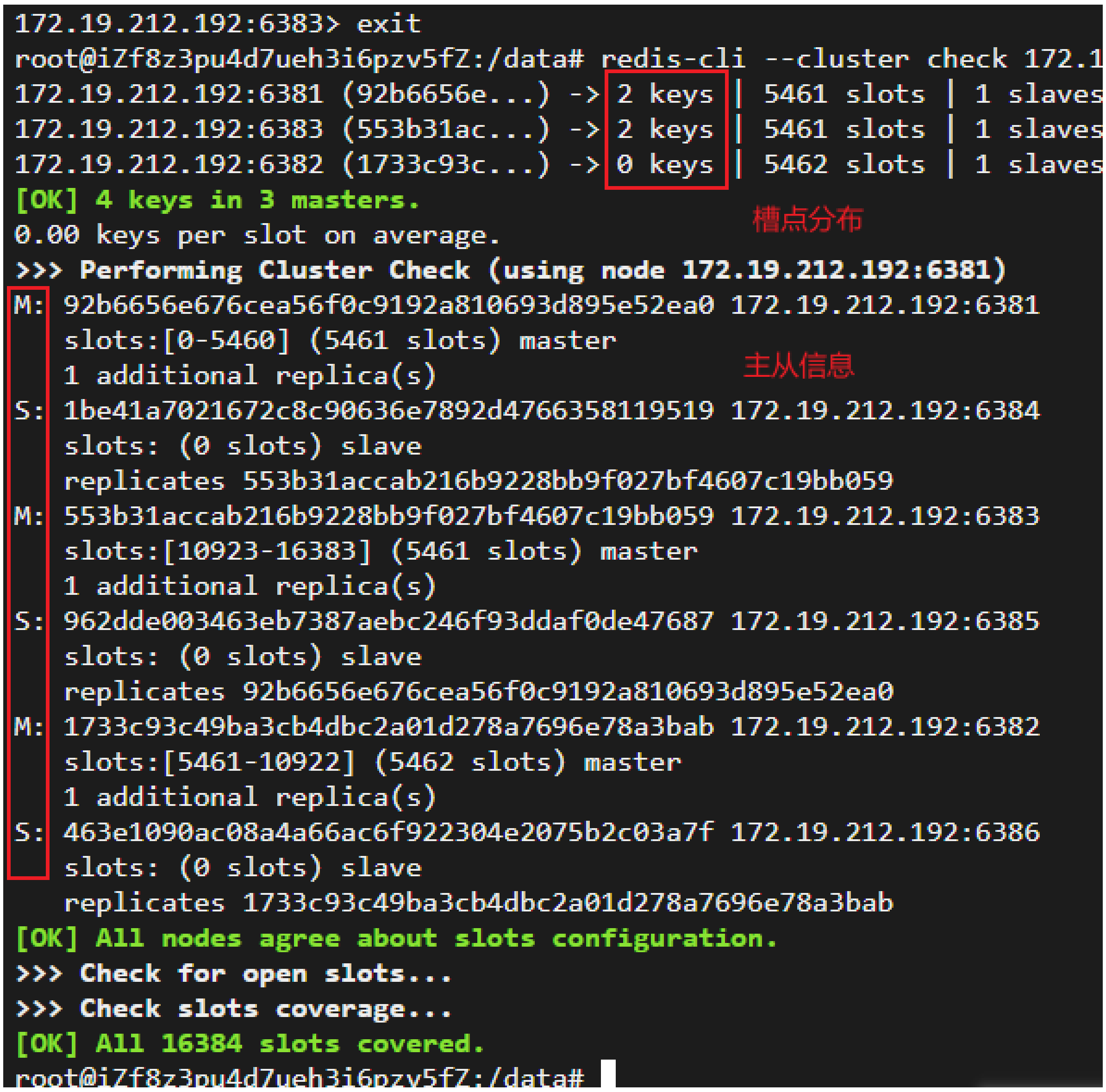
1.2、检查集群环境

集群环境连接

```
docker exec -it redis-node-1 /bin/bash
redis-cli -p 6381 -c
# 重定向
127.0.0.1:6381> set k1 v1
-> Redirected to slot [12706] located at 172.19.212.192:6383
OK
172.19.212.192:6383>
```

查看集群信息

```
172.19.212.192:6383> exit
# 在redis容器内
redis-cli --cluster check 172.19.212.192:6381
```

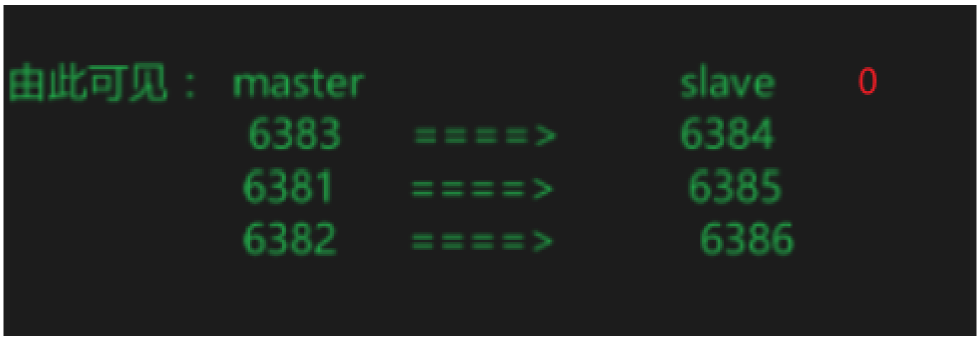


```
172.19.212.192:6383> exit
root@iZf8z3pu4d7ueh3i6pzv5fZ:/data# redis-cli --cluster check 172.1
172.19.212.192:6381 (92b6656e...) -> 2 keys | 5461 slots | 1 slaves
172.19.212.192:6383 (553b31ac...) -> 2 keys | 5461 slots | 1 slaves
172.19.212.192:6382 (1733c93c...) -> 0 keys | 5462 slots | 1 slaves
[OK] 4 keys in 3 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 172.19.212.192:6381)
M: 92b6656e676cea56f0c9192a810693d895e52ea0 172.19.212.192:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 1be41a7021672c8c90636e7892d4766358119519 172.19.212.192:6384
   slots: (0 slots) slave
   replicates 553b31accab216b9228bb9f027bf4607c19bb059
M: 553b31accab216b9228bb9f027bf4607c19bb059 172.19.212.192:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: 962dde003463eb7387aebc246f93ddaf0de47687 172.19.212.192:6385
   slots: (0 slots) slave
   replicates 92b6656e676cea56f0c9192a810693d895e52ea0
M: 1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 172.19.212.192:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
S: 463e1090ac08a4a66ac6f922304e2075b2c03a7f 172.19.212.192:6386
   slots: (0 slots) slave
   replicates 1733c93c49ba3cb4dbc2a01d278a7696e78a3bab
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
root@iZf8z3pu4d7ueh3i6pzv5fZ:/data#
```

2、主从应用

docker部署redis集群的步骤示例图

> 初始集群信息



```
由此可见： master            slave      0
           6383    ====>    6384
           6381    ====>    6385
           6382    ====>     6386
```

2.1、主从扩容

> 预计效果

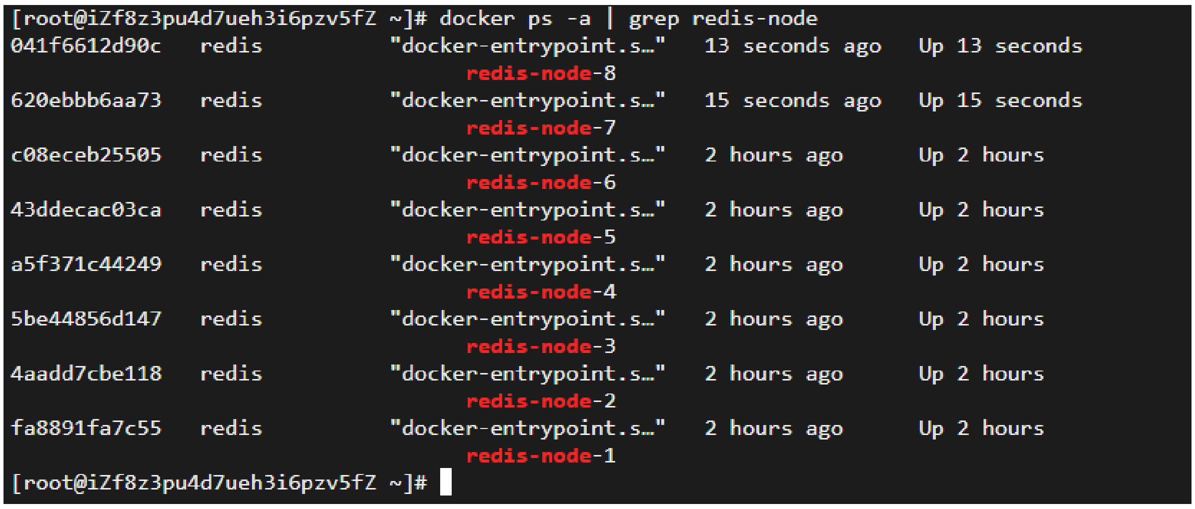
master : 6387

slave : 6388

> 步骤示例

1.1 创建两个镜像实例

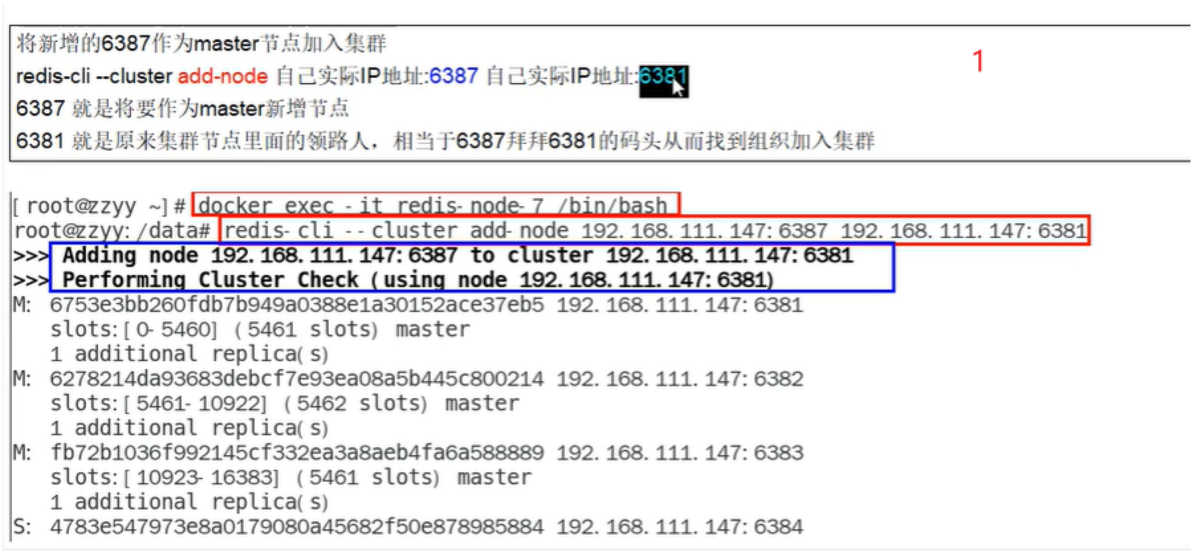
```
docker run -d --name=redis-node-7 --net host --privileged=true -v
/data/redis/share/redis-node-7:/data redis --cluster-enabled yes --appendonly
yes --port 6387

docker run -d --name=redis-node-8 --net host --privileged=true -v
/data/redis/share/redis-node-8:/data redis --cluster-enabled yes --appendonly
yes --port 6388


# 查看是否成功
docker ps -a | grep redis-node
```



```
[root@iZf8z3pu4d7ueh3i6pzv5fZ ~]# docker ps -a | grep redis-node
041f6612d90c   redis          "docker-entrypoint.s…"   13 seconds ago   Up 13 seconds
                                     redis-node-8
620ebbb6aa73   redis          "docker-entrypoint.s…"   15 seconds ago   Up 15 seconds
                                     redis-node-7
c08eceb25505   redis          "docker-entrypoint.s…"   2 hours ago      Up 2 hours
                                     redis-node-6
43ddecac03ca   redis          "docker-entrypoint.s…"   2 hours ago      Up 2 hours
                                     redis-node-5
a5f371c44249   redis          "docker-entrypoint.s…"   2 hours ago      Up 2 hours
                                     redis-node-4
5be44856d147   redis          "docker-entrypoint.s…"   2 hours ago      Up 2 hours
                                     redis-node-3
4aadd7cbe118   redis          "docker-entrypoint.s…"   2 hours ago      Up 2 hours
                                     redis-node-2
fa8891fa7c55   redis          "docker-entrypoint.s…"   2 hours ago      Up 2 hours
                                     redis-node-1
[root@iZf8z3pu4d7ueh3i6pzv5fZ ~]#
```

1.2 先进入redis-node-1容器加入主机6387

> redis-cli --cluster add-node 172.19.212.192:6387 172.19.212.192:6381

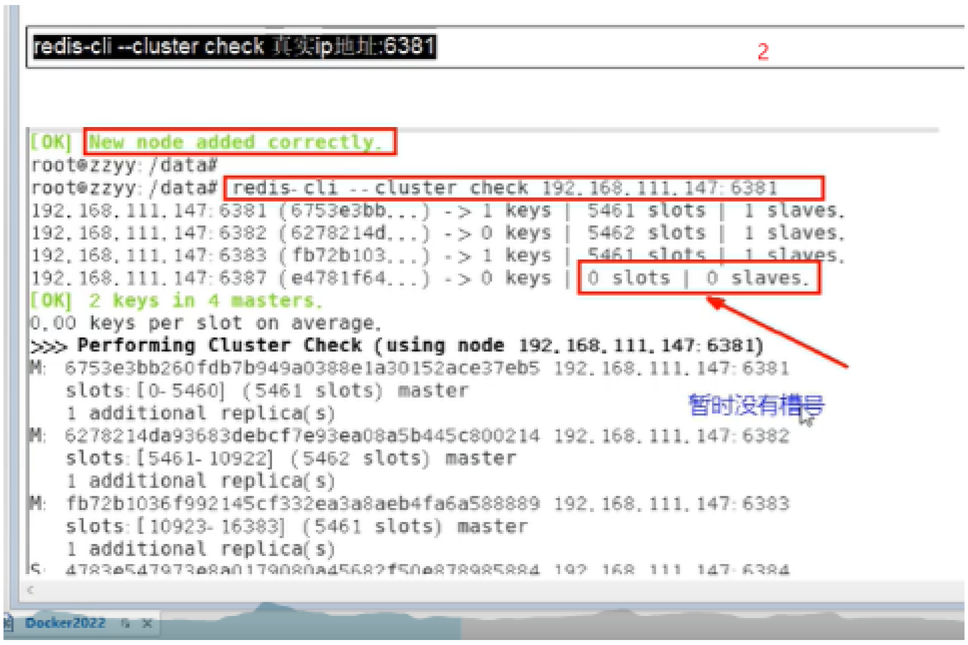


将新增的6387作为master节点加入集群
redis-cli --cluster add-node 自己实际IP地址:6387 自己实际IP地址:6381
6387 就是将要作为master新增节点
6381 就是原来集群节点里面的领路人，相当于6387拜拜6381的码头从而找到组织加入集群

```
[root@zzyy ~]# docker exec -it redis-node-7 /bin/bash
root@zzyy:/data# redis-cli --cluster add-node 192.168.111.147:6387 192.168.111.147:6381
>>> Adding node 192.168.111.147:6387 to cluster 192.168.111.147:6381
>>> Performing Cluster Check (using node 192.168.111.147:6381)
M: 6753e3bb260fdb7b949a0388e1a30152ace37eb5 192.168.111.147:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
M: 6278214da93683debcf7e93ea08a5b445c800214 192.168.111.147:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
M: fb72b1036f992145cf332ea3a8aeb4fa6a588889 192.168.111.147:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: 4783e547973e8a0179080a45682f50e878985884 192.168.111.147:6384
```

> redis-cli --cluster check ip:6381

以任一master节点出发检查当前的集群信息

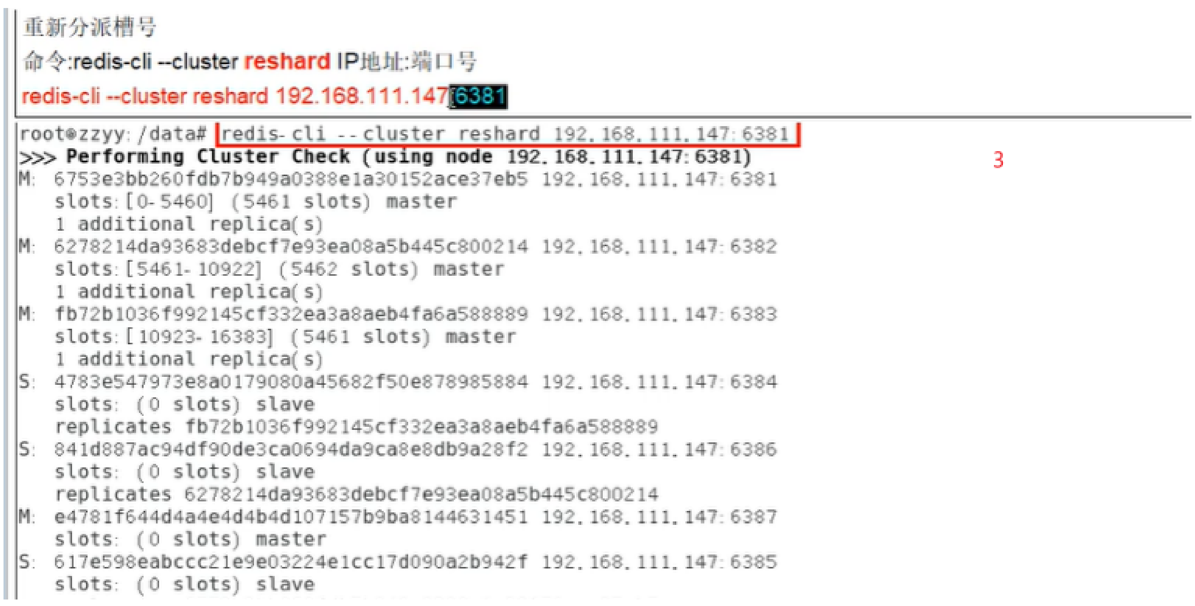


redis-cli --cluster check 真实ip地址:6381

```
[OK] New node added correctly.
root@zzyy:/data#
root@zzyy:/data# redis-cli --cluster check 192.168.111.147:6381
192.168.111.147:6381 (6753e3bb...) -> 1 keys | 5461 slots | 1 slaves.
192.168.111.147:6382 (6278214d...) -> 0 keys | 5462 slots | 1 slaves.
192.168.111.147:6383 (fb72b103...) -> 1 keys | 5461 slots | 1 slaves.
192.168.111.147:6387 (e4781f64...) -> 0 keys | 0 slots | 0 slaves.
[OK] 2 keys in 4 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.111.147:6381)
M: 6753e3bb260fdb7b949a0388e1a30152ace37eb5 192.168.111.147:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
M: 6278214da93683debcf7e93ea08a5b445c800214 192.168.111.147:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
M: fb72b1036f992145cf332ea3a8aeb4fa6a588889 192.168.111.147:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: 4783e547973e8a0179080a45682f50e878985884 192.168.111.147:6384
```

> redis-cli --cluster reshard ip:6381

重新分配槽号



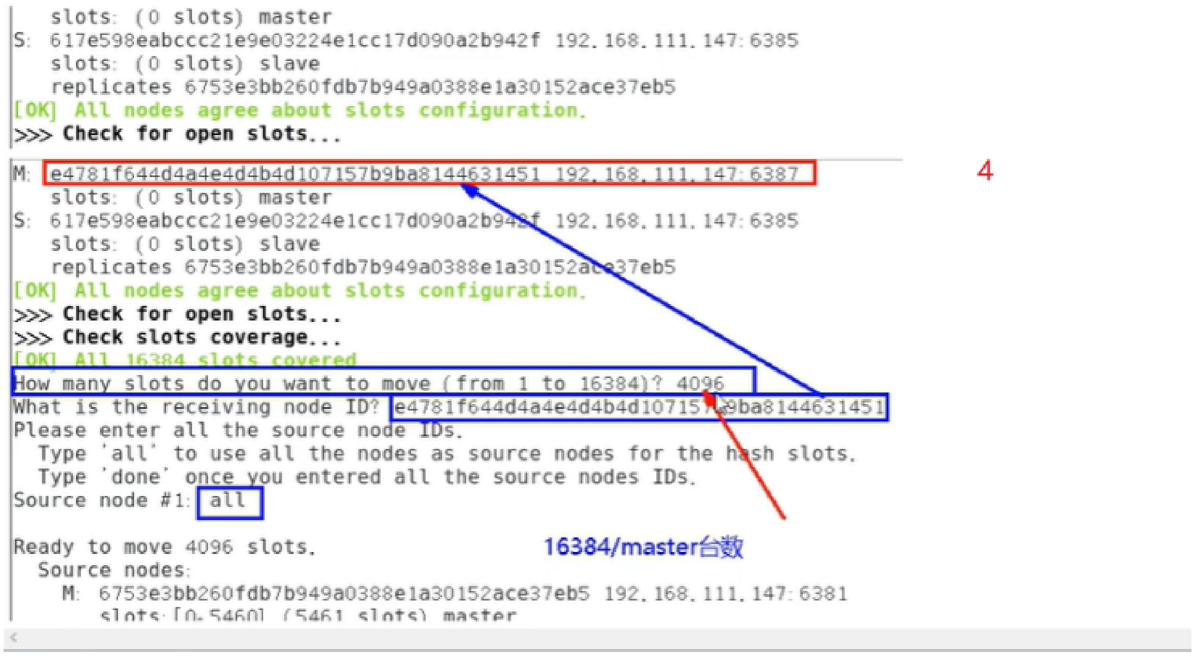
重新分派槽号

命令:redis-cli --cluster reshard IP地址:端口号

redis-cli --cluster reshard 192.168.111.147:6381

```
root@zzyy:/data# redis-cli --cluster reshard 192.168.111.147:6381
>>> Performing Cluster Check (using node 192.168.111.147:6381)
M: 6753e3bb260fdb7b949a0388e1a30152ace37eb5 192.168.111.147:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
M: 6278214da93683debcf7e93ea08a5b445c800214 192.168.111.147:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
M: fb72b1036f992145cf332ea3a8aeb4fa6a588889 192.168.111.147:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: 4783e547973e8a0179080a45682f50e878985884 192.168.111.147:6384
   slots: (0 slots) slave
   replicates fb72b1036f992145cf332ea3a8aeb4fa6a588889
S: 841d887ac94df90de3ca0694da9ca8e8db9a28f2 192.168.111.147:6386
   slots: (0 slots) slave
   replicates 6278214da93683debcf7e93ea08a5b445c800214
M: e4781f644d4a4e4d4b4d107157b9ba8144631451 192.168.111.147:6387
   slots: (0 slots) master
S: 617e598eabccc21e9e03224e1cc17d090a2b942f 192.168.111.147:6385
   slots: (0 slots) slave
```

3

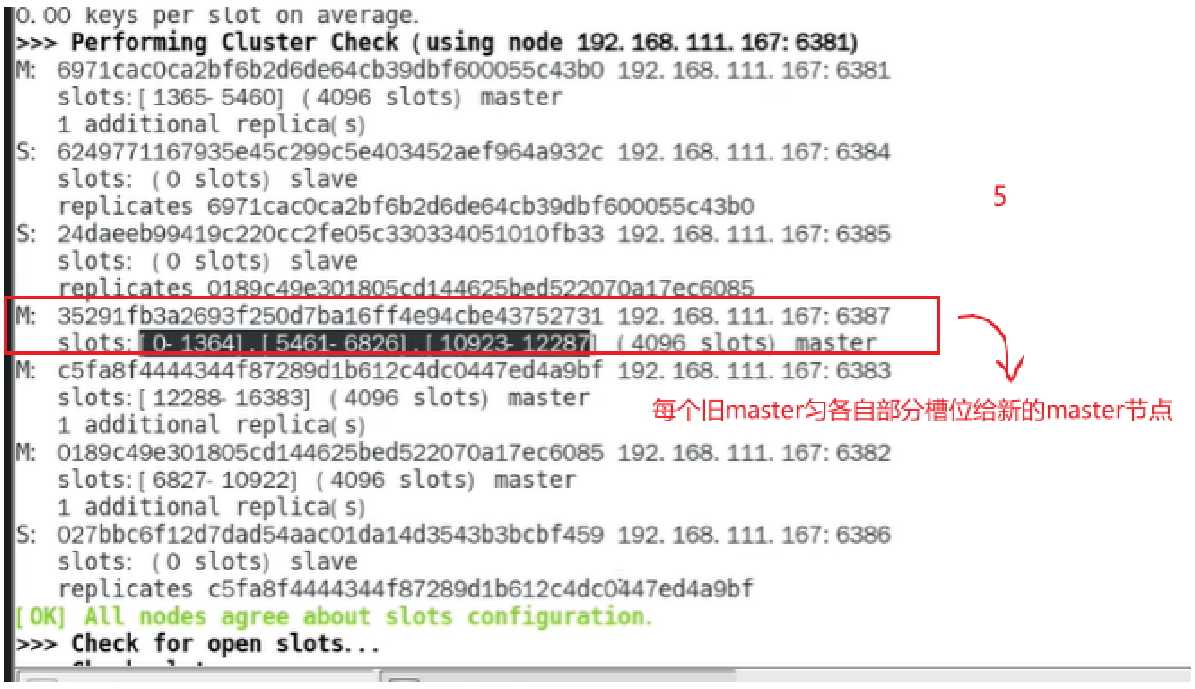


```
   slots: (0 slots) master
S: 617e598eabccc21e9e03224e1cc17d090a2b942f 192.168.111.147:6385
   slots: (0 slots) slave
   replicates 6753e3bb260fdb7b949a0388e1a30152ace37eb5
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
```

```
M: e4781f644d4a4e4d4b4d107157b9ba8144631451 192.168.111.147:6387
   slots: (0 slots) master
S: 617e598eabccc21e9e03224e1cc17d090a2b942f 192.168.111.147:6385
   slots: (0 slots) slave
   replicates 6753e3bb260fdb7b949a0388e1a30152ace37eb5
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered
How many slots do you want to move (from 1 to 16384)? 4096
What is the receiving node ID? e4781f644d4a4e4d4b4d107157b9ba8144631451
Please enter all the source node IDs.
  Type 'all' to use all the nodes as source nodes for the hash slots.
  Type 'done' once you entered all the source nodes IDs.
Source node #1: all

Ready to move 4096 slots.
  Source nodes:
    M: 6753e3bb260fdb7b949a0388e1a30152ace37eb5 192.168.111.147:6381
       slots:[0-5460] (5461 slots) master
```

4

16384/master台数



```
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.111.167:6381)
M: 6971cac0ca2bf6b2d6de64cb39dbf600055c43b0 192.168.111.167:6381
   slots:[1365-5460] (4096 slots) master
   1 additional replica(s)
S: 6249771167935e45c299c5e403452aef964a932c 192.168.111.167:6384
   slots: (0 slots) slave
   replicates 6971cac0ca2bf6b2d6de64cb39dbf600055c43b0
S: 24daeeb99419c220cc2fe05c330334051010fb33 192.168.111.167:6385
   slots: (0 slots) slave
   replicates 0189c49e301805cd144625bed522070a17ec6085
M: 35291fb3a2693f250d7ba16ff4e94cbe43752731 192.168.111.167:6387
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
M: c5fa8f4444344f87289d1b612c4dc0447ed4a9bf 192.168.111.167:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
M: 0189c49e301805cd144625bed522070a17ec6085 192.168.111.167:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
S: 027bbc6f12d7dad54aac01da14d3543b3bcbf459 192.168.111.167:6386
   slots: (0 slots) slave
   replicates c5fa8f4444344f87289d1b612c4dc0447ed4a9bf
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
```

5

每个旧master匀各自部分槽位给新的master节点

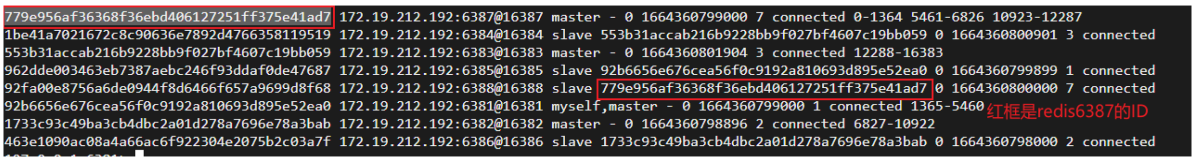
1.3 加入从机6388

```
redis-cli --cluster add-node ip:6388 ip:6387 --cluster-slave --cluster-master-id 填入6387的编号#我的是779e956af36368f36ebd406127251ff375e41ad7
```

redis-cli --cluster add-node 172.19.212.192:6388 172.19.212.192:6387 --cluster-slave --cluster-master-id 779e956af36368f36ebd406127251ff375e41ad7

1.4 确认集群信息

```
redis-cli --cluster check ip:6381
```



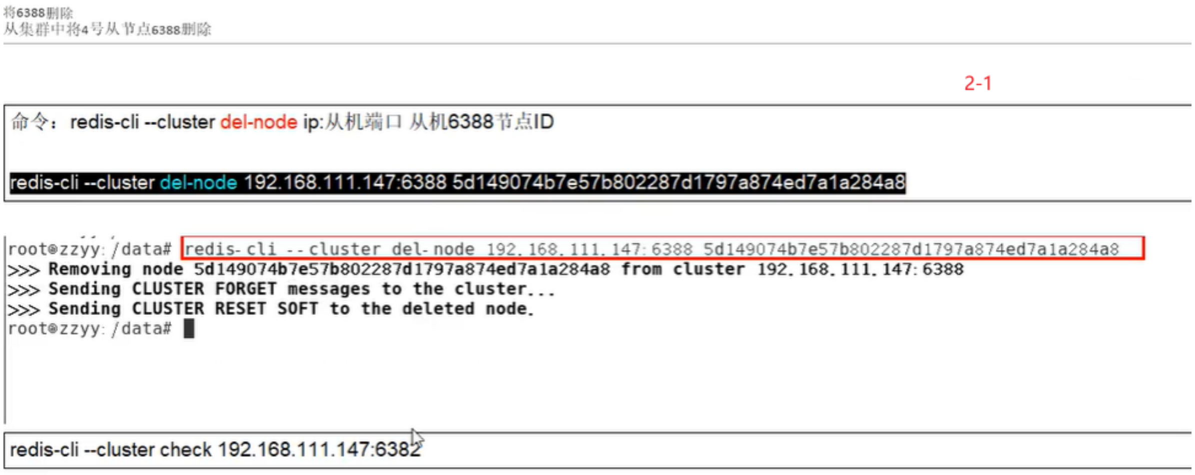
```
779e956af36368f36ebd406127251ff375e41ad7 172.19.212.192:6387@16387 master - 0 1664360799000 7 connected 0-1364 5461-6826 10923-12287
1be41a7021672c8c90636e7892d4766358119519 172.19.212.192:6384@16384 slave 553b31accab216b9228bb9f027bf4607c19bb059 0 1664360800901 3 connected
553b31accab216b9228bb9f027bf4607c19bb059 172.19.212.192:6383@16383 master - 0 1664360801904 3 connected 12288-16383
962dde003463eb7387aebc246f93ddaf0de47687 172.19.212.192:6385@16385 slave 92b6656e676cea56f0c9192a810693d895e52ea0 0 1664360799899 1 connected
92fa00e8756a6de0944f8d6466f657a9699d8f68 172.19.212.192:6388@16388 slave 779e956af36368f36ebd406127251ff375e41ad7 0 1664360800000 7 connected
92b6656e676cea56f0c9192a810693d895e52ea0 172.19.212.192:6381@16381 myself,master - 0 1664360799000 1 connected 1365-5460
1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 172.19.212.192:6382@16382 master - 0 1664360798896 2 connected 6827-10922
463e1090ac08a4a66ac6f922304e2075b2c03a7f 172.19.212.192:6386@16386 slave 1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 0 1664360798000 2 connected
```

2.2、主从缩容

先删从机6388

> redis-cli --cluster del-node ip:6388 加上6388的节点ID

redis-cli --cluster del-node 172.19.212.192:6388 92fa00e8756a6de0944f8d6466f657a9699d8f68



将6388删除
从集群中将4号从节点6388删除

2-1

命令：redis-cli --cluster del-node ip:从机端口 从机6388节点ID

redis-cli --cluster del-node 192.168.111.147:6388 5d149074b7e57b802287d1797a874ed7a1a284a8

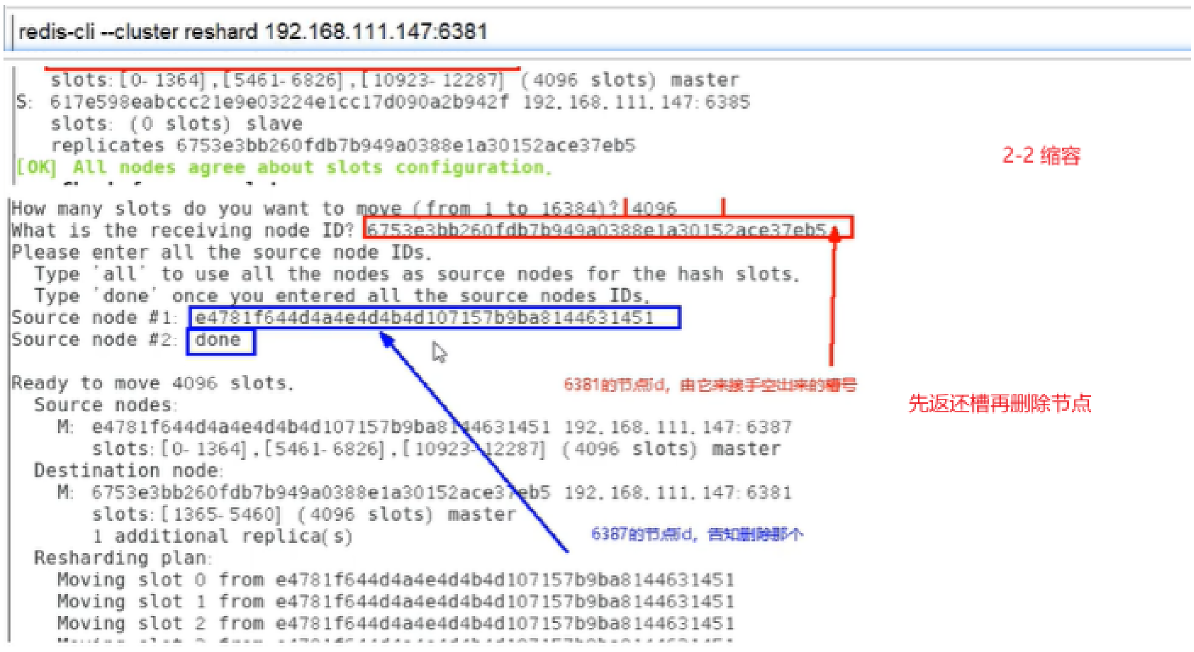
```
root@zzyy:/data# redis-cli --cluster del-node 192.168.111.147:6388 5d149074b7e57b802287d1797a874ed7a1a284a8
>>> Removing node 5d149074b7e57b802287d1797a874ed7a1a284a8 from cluster 192.168.111.147:6388
>>> Sending CLUSTER FORGET messages to the cluster...
>>> Sending CLUSTER RESET SOFT to the deleted node.
root@zzyy:/data# 
```

redis-cli --cluster check 192.168.111.147:6382

返还槽位

> redis-cli --cluster reshard ip:master的port （like redis-cli --cluster reshard ip:6381)

redis-cli --cluster reshard 172.19.212.192:6381



```
redis-cli --cluster reshard 192.168.111.147:6381
```

```
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
S: 617e598eabccc21e9e03224e1cc17d090a2b942f 192.168.111.147:6385
   slots: (0 slots) slave
   replicates 6753e3bb260fdb7b949a0388e1a30152ace37eb5
[OK] All nodes agree about slots configuration.

How many slots do you want to move (from 1 to 16384)? 4096
What is the receiving node ID? 6753e3bb260fdb7b949a0388e1a30152ace37eb5
Please enter all the source node IDs.
  Type 'all' to use all the nodes as source nodes for the hash slots.
  Type 'done' once you entered all the source nodes IDs.
Source node #1: e4781f644d4a4e4d4b4d107157b9ba8144631451
Source node #2: done

Ready to move 4096 slots.
  Source nodes:
    M: e4781f644d4a4e4d4b4d107157b9ba8144631451 192.168.111.147:6387
       slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
  Destination node:
    M: 6753e3bb260fdb7b949a0388e1a30152ace37eb5 192.168.111.147:6381
       slots:[1365-5460] (4096 slots) master
       1 additional replica(s)
  Resharding plan:
    Moving slot 0 from e4781f644d4a4e4d4b4d107157b9ba8144631451
    Moving slot 1 from e4781f644d4a4e4d4b4d107157b9ba8144631451
    Moving slot 2 from e4781f644d4a4e4d4b4d107157b9ba8144631451
```

2-2 缩容
6381的节点id，由它来接手空出来的槽号
先返还槽再删除节点
6387的节点id，告知删除那个

最后删除主机6387

> redis-cli --cluster del-node ip:6387 加上6387的ID

redis-cli --cluster del-node 172.19.212.192:6387 779e956af36368f36ebd406127251ff375e41ad7

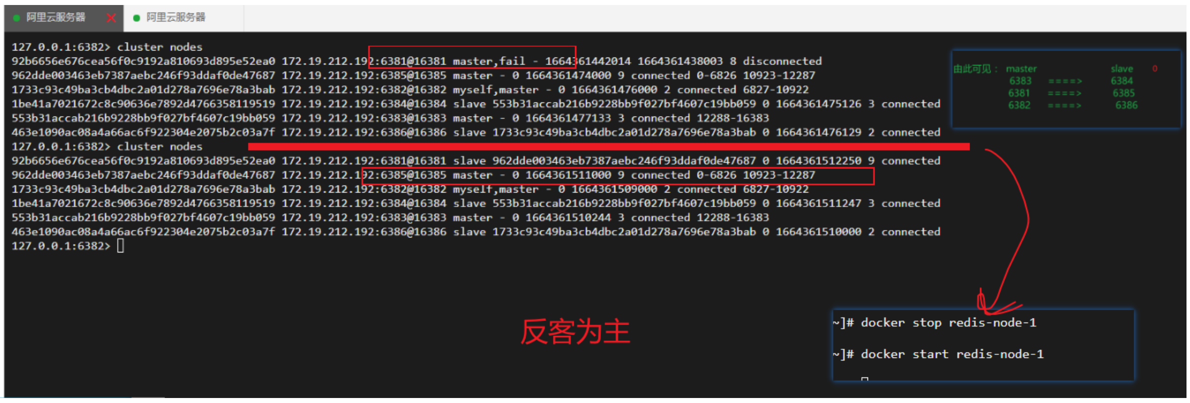
查看集群信息确认删除

> redis-cli --cluster check ip:6381

2.3、主从切换

```
docker exec -it redis-node-3 /bin/bash
# 查看初始的主从关系
redis-cli -p 6383 -c
127.0.0.1:6381> cluster nodes
# or
127.0.0.1:6381> exit
redis-cli --cluster check 172.19.212.192:6383



# 此时6385是6381的奴隶

# 在宿主机中
docker stop redis-node-1    （即6381主机）

# 再次查看

# 重启
docker start redis-node-1
```

```
# 此时6385成为6381的主人，反客为主
```



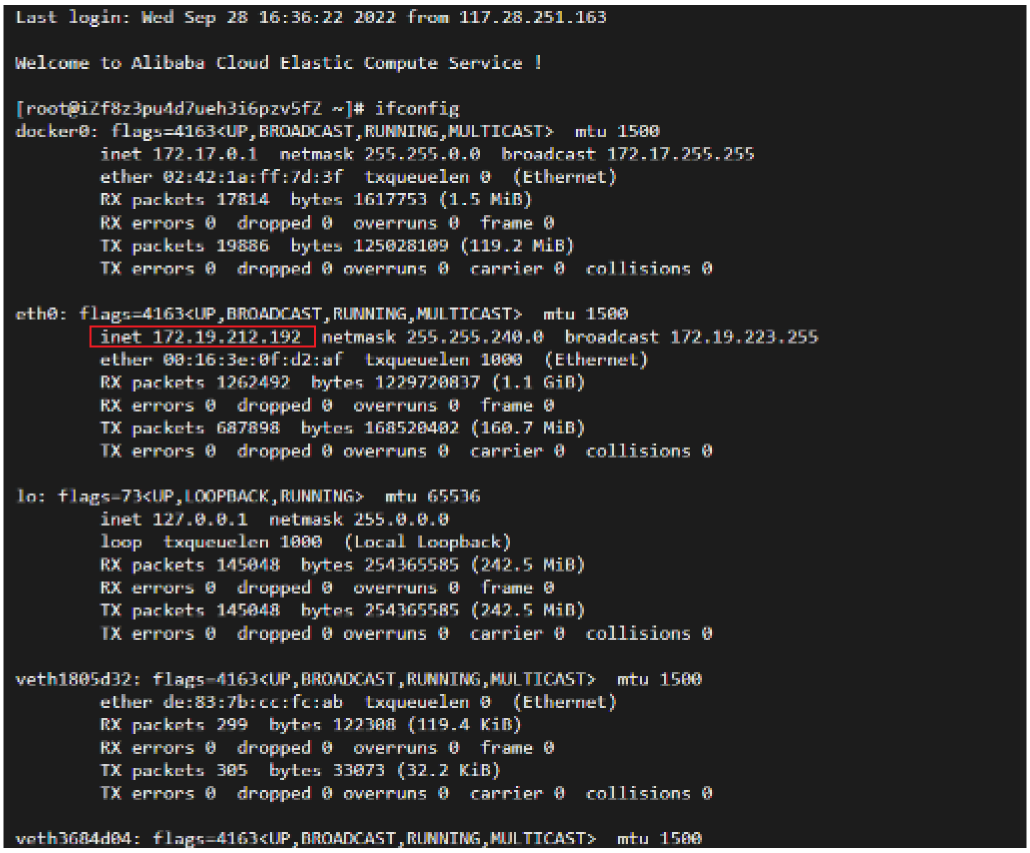
```
127.0.0.1:6382> cluster nodes
92b6656e676cea56f0c9192a810693d895e52ea0 172.19.212.192:6381@16381 master,fail - 1664361442014 1664361438003 8 disconnected
962dde003463eb7387aebc246f93ddaf0de47687 172.19.212.192:6385@16385 master - 0 1664361474000 9 connected 0-6826 10923-12287
1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 172.19.212.192:6382@16382 myself,master - 0 1664361476000 2 connected 6827-10922
1be41a7021672c8c90636e7892d4766358119519 172.19.212.192:6384@16384 slave 553b31accab216b9228bb9f027bf4607c19bb059 0 1664361475126 3 connected
553b31accab216b9228bb9f027bf4607c19bb059 172.19.212.192:6383@16383 master - 0 1664361477133 3 connected 12288-16383
463e1090ac08a4a66ac6f922304e2075b2c03a7f 172.19.212.192:6386@16386 slave 1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 0 1664361476129 2 connected
127.0.0.1:6382> cluster nodes
92b6656e676cea56f0c9192a810693d895e52ea0 172.19.212.192:6381@16381 slave 962dde003463eb7387aebc246f93ddaf0de47687 0 1664361512250 9 connected
962dde003463eb7387aebc246f93ddaf0de47687 172.19.212.192:6385@16385 master - 0 1664361511000 9 connected 0-6826 10923-12287
1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 172.19.212.192:6382@16382 myself,master - 0 1664361509000 2 connected 6827-10922
1be41a7021672c8c90636e7892d4766358119519 172.19.212.192:6384@16384 slave 553b31accab216b9228bb9f027bf4607c19bb059 0 1664361511247 3 connected
553b31accab216b9228bb9f027bf4607c19bb059 172.19.212.192:6383@16383 master - 0 1664361510244 3 connected 12288-16383
463e1090ac08a4a66ac6f922304e2075b2c03a7f 172.19.212.192:6386@16386 slave 1733c93c49ba3cb4dbc2a01d278a7696e78a3bab 0 1664361510000 2 connected
127.0.0.1:6382>
```

```
由此可见：  master          slave     0
            6383   ====>    6384
            6381   ====>    6385
            6382   ====>    6386
```

```
~]# docker stop redis-node-1

~]# docker start redis-node-1
```

注：

本机的ip查看： 172.19.212.192



```
Last login: Wed Sep 28 16:36:22 2022 from 117.28.251.163

Welcome to Alibaba Cloud Elastic Compute Service !

[root@iZf8z3pu4d7ueh3i6pzv5fZ ~]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:1a:ff:7d:3f  txqueuelen 0  (Ethernet)
        RX packets 17814  bytes 1617753 (1.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19886  bytes 125028109 (119.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.19.212.192  netmask 255.255.240.0  broadcast 172.19.223.255
        ether 00:16:3e:0f:d2:af  txqueuelen 1000  (Ethernet)
        RX packets 1262492  bytes 1229720837 (1.1 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 687898  bytes 168520402 (160.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 145048  bytes 254365585 (242.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 145048  bytes 254365585 (242.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth1805d32: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether de:83:7b:cc:fc:ab  txqueuelen 0  (Ethernet)
        RX packets 299  bytes 122308 (119.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 305  bytes 33073 (32.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth3684d04: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

对应的节点查看:

redis-cli --cluster check 172.19.212.192:6381

数据存在槽里的，槽重新分配，数据就有可能不在原来的节点上了