cluster(集群)模式-docker版哈希槽分区进行亿级数据存储

- 面试题：

```
1~2亿条数据需要缓存，请问如何设计这个存储案例
```

- 回答：

```
  单机单台100%不可能，肯定是分布式存储，用redis如何落地？
```

上述问题阿里P6~P7工程案例和场景设计类必考题目，一般业界有3中解决方案

1、哈希取余分区

- 原理：2亿条记录就是2亿个k,v，我们单机不行必须要分布式多机，假设有3台机器构成一个集群，用户每次读写操作都是根据公式：hash(key) % N个机器台数，计算出哈希值，用来决定数据映射到哪一个节点上
- 优点： 简单粗暴，直接有效，只需要预估好数据规划好节点，例如3台、8台、10台，就能保证一段时间的数据支撑。使用Hash算法让固定的一部分请求落到同一台服务器上，这样每台服务器固定处理一部分请求（并维护这些请求的信息），起到负载均衡+分而治之的作用。
- 缺点： 原来规划好的节点，进行扩容或者缩容就比较麻烦了额，不管扩缩，每次数据变动导致节点有变动，映射关系需要重新进行计算，在服务器个数固定不变时没有问题，如果需要弹性扩容或故障停机的情况下，原来的取模公式就会发生变化：Hash(key)/3会变成Hash(key)/?。此时地址经过取余运算的结果将发生很大变化，根据公式获取的服务器也会变得不可控。某个redis机器宕机了，由于台数数量变化，会导致hash取余全部数据重新洗牌。

2、一直性哈希算法分区

- 能干嘛?

  提出一致性Hash解决方案。目的是当服务器个数发生变动时尽量减少影响客户端到服务器的映射关系

- 3大步骤：

  算法构建一致性哈希环： *致性哈希算法必然有个hash函数并按照算法产生hash值，这个算法的所有可能哈希值会构成一个全量集，这个集合可以成为一个hash空间[0,2^32-1]，这个是一个线性空间，但是在算法中，我们通过适当的逻辑控制将它首尾相连(0 = 2^32),这样让它逻辑上形成了一个环形空间。*

  服务器IP节点映射

  将集群中各个IP节点或者主机名作为关键字进行哈希映射到环上的某一个位置。

  key落到服务器规则：

当我们需要存储一个kv键值对时，首先计算key的hash值，hash(key)，将这个key使用相同的函数Hash计算出哈希值并确定此数据在环上的位置，从此位置沿环顺时针"行走"，第一台遇到的服务器就是其应该定位到的服务器，并将该键值对存储在该节点上。

- 优点：

  容错性： *假设Node C宕机，可以看到此时对象A、B、D不会受到影响，只有C对象被重定位到Node D。一般的，在一致性Hash算法中，如果一台服务器不可用，则受影响的数据仅仅是此服务器到其环空间中前一台服务器（即沿着逆时针方向行走遇到的第一台服务器）之间数据，其它不会受到影响。简单说，就是C挂了，受到影响的只是B、C之间的数据，并且这些数据会转移到D进行存储。*

  容错性：

  数据量增加了，需要增加一台节点NodeX，X的位置在A和B之间，那收到影响的也就是A到X之间的数据，重新把A到X的数据录入到X上即可，不会导致hash取余全部数据重新洗牌。

- 缺点：

  Hash环的数据倾斜问题

  一致性Hash算法在服务节点太少时，容易因为节点分布不均匀而造成数据倾斜（被缓存的对象大部分集中缓存在某一台服务器上）问题，

  例如系统中只有两台服务器：一台服务器宕机后会有大量数据打到一台服务器上.

3、哈希槽分区

- 是什么?

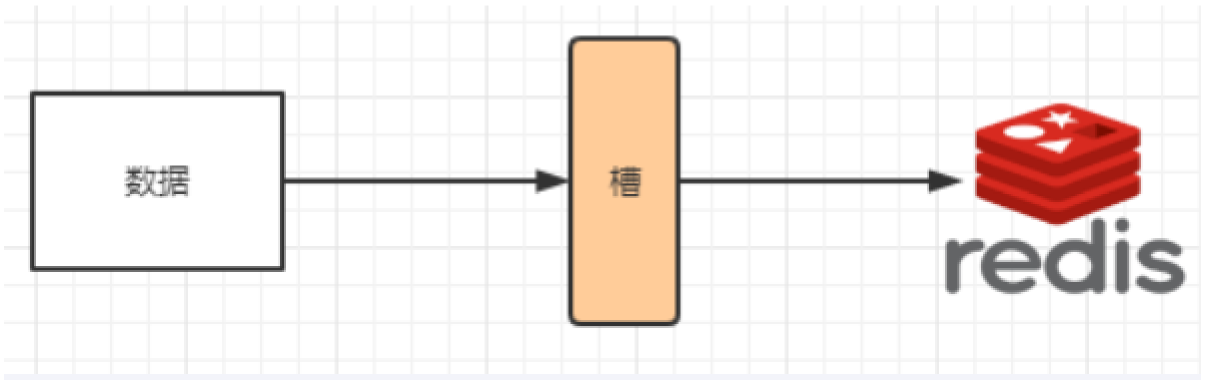
  为什么出现?

  解决一致性哈希算法的数据倾斜问题

  哈希槽的实质就是一个数组，数组[0,2^14-1]形成hash slot空间

  能干什么?

  解决均匀分配的问题，在数据和节点之间又加入了一层，把这层称为哈希槽（slot），用于管理数据和节点之间的关系，现在就相当于节点上放的是槽，槽里放的是数据。



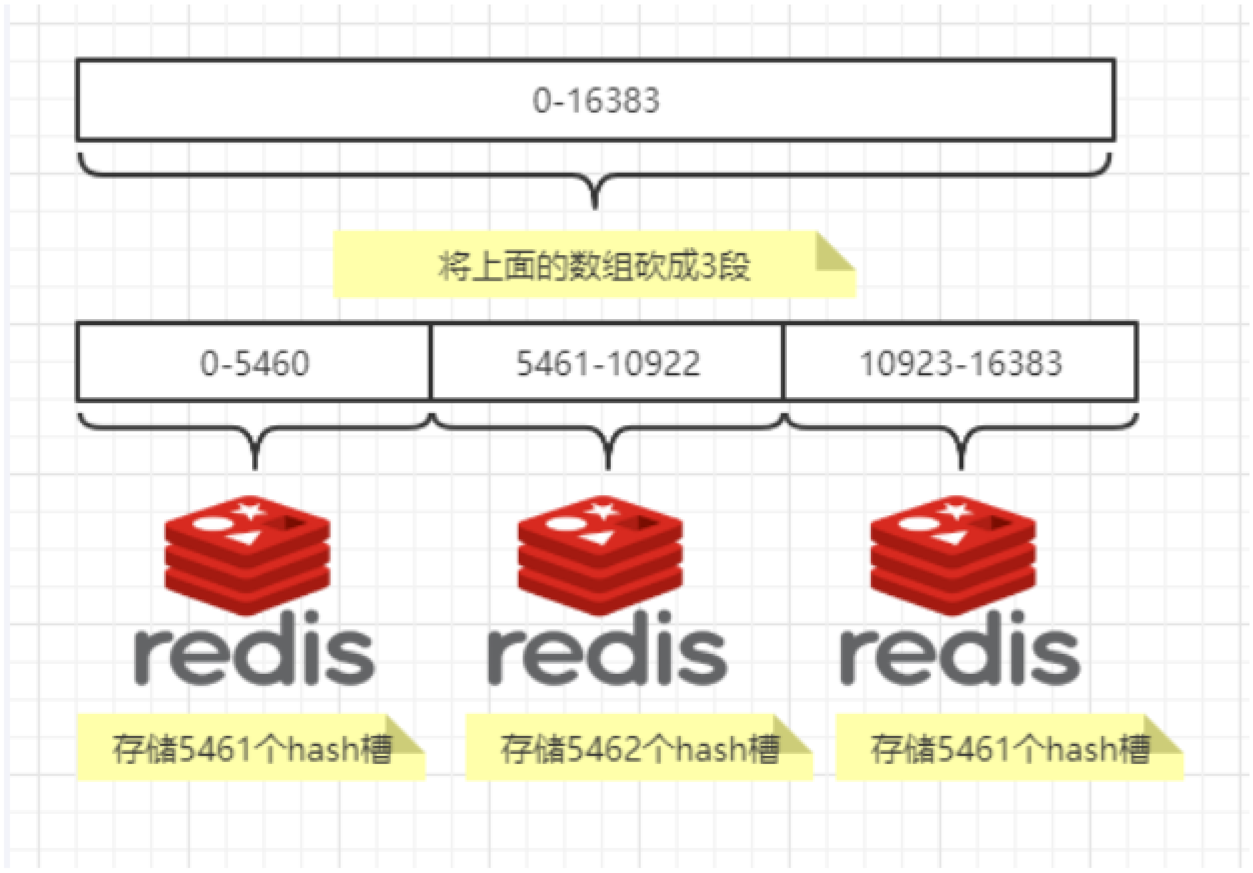
槽解决的是粒度问题，相当于把粒度变大了，这样便于数据移动。哈希解决的是映射问题，使用key的哈希值来计算所在的槽，便于数据分配。

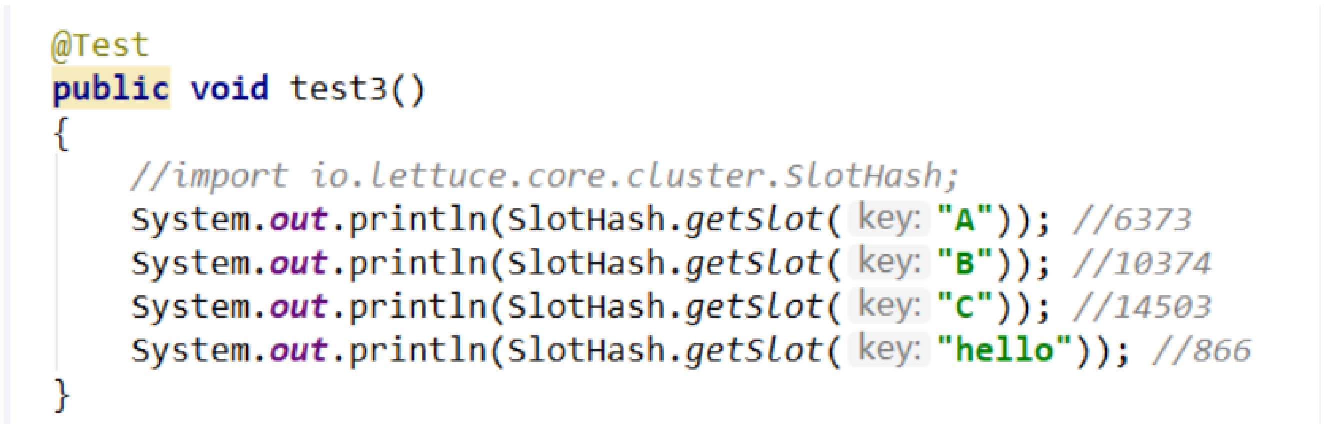
多少个hash槽

一个集群只能有16384个槽，编号0-16383（0-2^14-1）。这些槽会分配给集群中的所有主节点，分配策略没有要求。可以指定哪些编号的槽分配给哪个主节点。集群会记录节点和槽的对应关系。解决了节点和槽的关系后，接下来就需要对key求哈希值，然后对16384取余，余数是几key就落入对应的槽里。slot = CRC16(key) % 16384。以槽为单位移动数据，因为槽的数目是固定的，处理起来比较容易，这样数据移动问题就解决了。

- 哈希槽计算

  Redis 集群中内置了 16384 个哈希槽，redis 会根据节点数量大致均等的将哈希槽映射到不同的节点。当需要在 Redis 集群中放置一个 key-value时，redis 先对 key 使用 crc16 算法算出一个结果，然后把结果对 16384 求余数，这样每个 key 都会对应一个编号在 0-16383 之间的哈希槽，也就是映射到某个节点上。如下代码，key之A 、B在Node2， key之C落在Node3上




```
@Test
public void test3()
{
    //import io.lettuce.core.cluster.SlotHash;
    System.out.println(SlotHash.getSlot( key: "A")); //6373
    System.out.println(SlotHash.getSlot( key: "B")); //10374
    System.out.println(SlotHash.getSlot( key: "C")); //14503
    System.out.println(SlotHash.getSlot( key: "hello")); //866
}
```

1、3主3从redis集群配置

关闭防火墙+启动docker后台服务

```
systemctl start docker
```

新建6个docker容器redis实例

创建需要映射的目录：

```
mkdir -p /data/docker/redis-cluster/redis-node-1/data /data/docker/redis-cluster/redis-node-1/conf
mkdir -p /data/docker/redis-cluster/redis-node-2/data /data/docker/redis-cluster/redis-node-2/conf
mkdir -p /data/docker/redis-cluster/redis-node-3/data /data/docker/redis-cluster/redis-node-3/conf
mkdir -p /data/docker/redis-cluster/redis-node-4/data /data/docker/redis-cluster/redis-node-4/conf
mkdir -p /data/docker/redis-cluster/redis-node-5/data /data/docker/redis-cluster/redis-node-5/conf
mkdir -p /data/docker/redis-cluster/redis-node-6/data /data/docker/redis-cluster/redis-node-6/conf
```

端口规划：6381~6386

```
docker run -d --name redis-node-1 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-1:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6381
docker run -d --name redis-node-2 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-2:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6382
docker run -d --name redis-node-3 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-3:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6383
docker run -d --name redis-node-4 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-4:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port
```

```
6384
docker run -d --name redis-node-5 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-5:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6385
docker run -d --name redis-node-6 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-6:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6386
```

命令分步解释：

docker run：创建并运行docker容器实例

--name redis-node-1:容器名字

--net host:使用主机的IP和端口，默认

--privileged=true:获取宿主机root用户权限

-v /data/docker/redis-cluster/redis-node-6:/data：容器卷，宿主机地址：docker内部地址

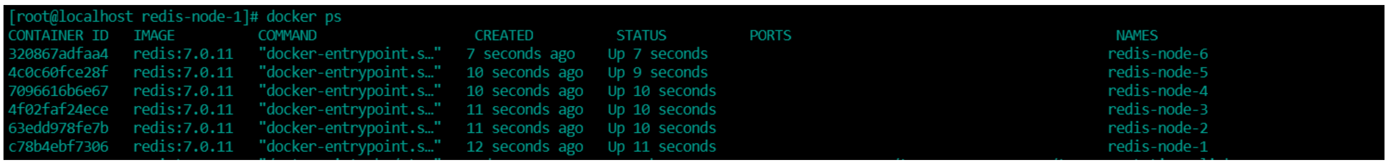
reids:7.0.11:redis镜像和版本号

--cluster-enabled yes:开启redis集群

--appendonly yes：开启持久化

--port 6386:redis端口号

查看reids是否启动



```
[root@localhost redis-node-1]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                       NAMES
320867adfaa4   redis:7.0.11   "docker-entrypoint.s…"   7 seconds ago    Up 7 seconds                                                redis-node-6
4c0c60fce28f   redis:7.0.11   "docker-entrypoint.s…"   10 seconds ago   Up 9 seconds                                                redis-node-5
7096616b6e67   redis:7.0.11   "docker-entrypoint.s…"   10 seconds ago   Up 10 seconds                                               redis-node-4
4f02faf24ece   redis:7.0.11   "docker-entrypoint.s…"   11 seconds ago   Up 10 seconds                                               redis-node-3
63edd978fe7b   redis:7.0.11   "docker-entrypoint.s…"   11 seconds ago   Up 10 seconds                                               redis-node-2
c78b4ebf7306   redis:7.0.11   "docker-entrypoint.s…"   12 seconds ago   Up 11 seconds                                               redis-node-1
```

进入容器redis-node-1并为6台机器构建集群关系

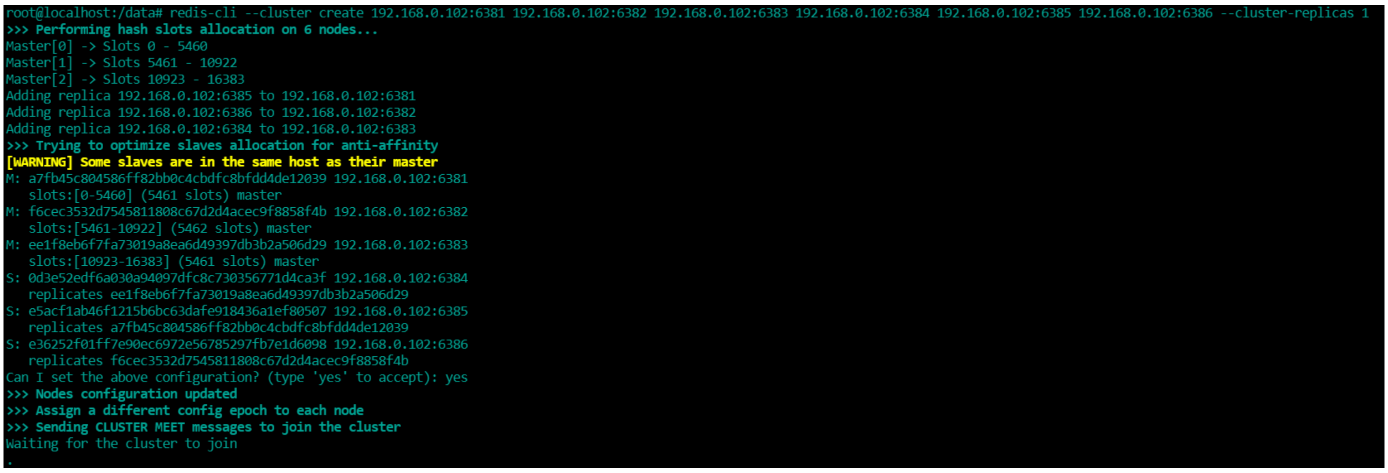
进入容器

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

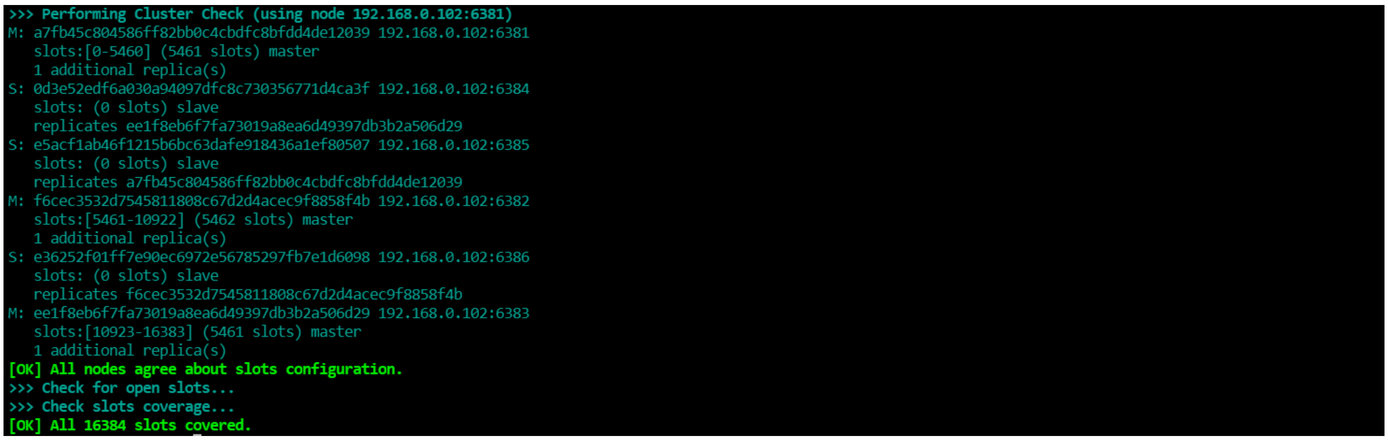
构建组从关系(容器内执行)

```
redis-cli --cluster create 192.168.0.102:6381 192.168.0.102:6382 192.168.0.102:6383 192.168.0.102:6384 192.168.0.102:6385 192.168.0.102:6386 --cluster-replicas 1
```

--cluster-replicas 1 表示为每个master创建一个slave节点



```
root@localhost:/data# redis-cli --cluster create 192.168.0.102:6381 192.168.0.102:6382 192.168.0.102:6383 192.168.0.102:6384 192.168.0.102:6385 192.168.0.102:6386 --cluster-replicas 1
>>> Performing hash slots allocation on 6 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
Adding replica 192.168.0.102:6385 to 192.168.0.102:6381
Adding replica 192.168.0.102:6386 to 192.168.0.102:6382
Adding replica 192.168.0.102:6384 to 192.168.0.102:6383
>>> Trying to optimize slaves allocation for anti-affinity
[WARNING] Some slaves are in the same host as their master
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-5460] (5461 slots) master
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[5461-10922] (5462 slots) master
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[10923-16383] (5461 slots) master
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
Can I set the above configuration? (type 'yes' to accept): yes
>>> Nodes configuration updated
>>> Assign a different config epoch to each node
>>> Sending CLUSTER MEET messages to join the cluster
Waiting for the cluster to join
.
```

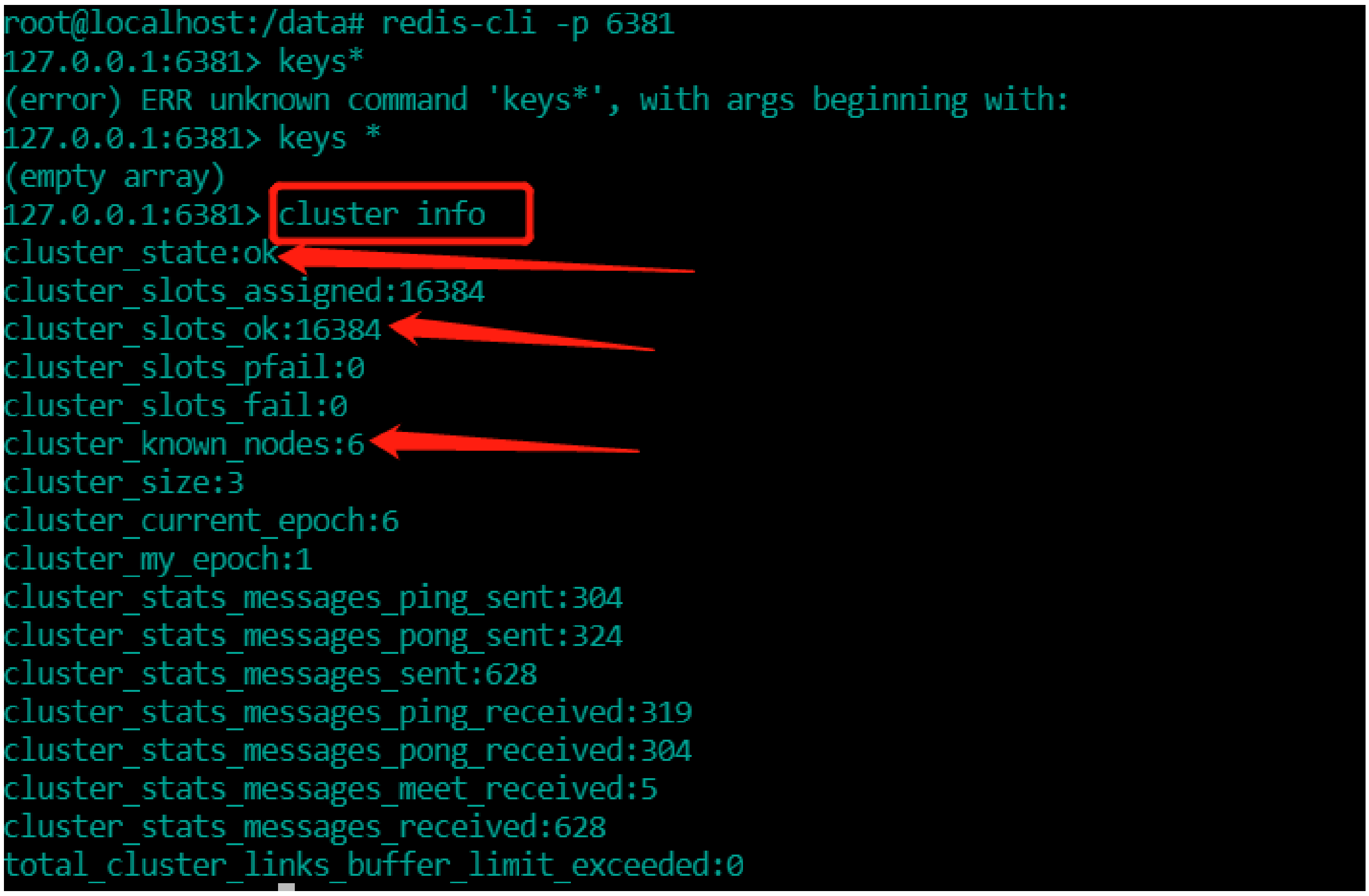


```
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

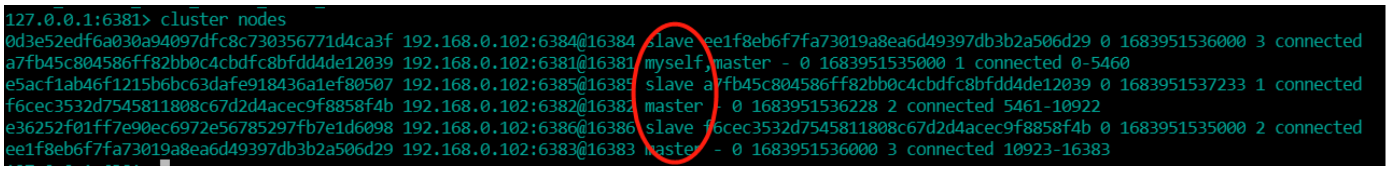
如上图显示3主3从构建成功

链接进入6381作为切入点，查看集群状态

```
# 查看集群信息
cluster info
# 查看集群节点信息
cluster nodes
```



```
root@localhost:/data# redis-cli -p 6381
127.0.0.1:6381> keys*
(error) ERR unknown command 'keys*', with args beginning with:
127.0.0.1:6381> keys *
(empty array)
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:304
cluster_stats_messages_pong_sent:324
cluster_stats_messages_sent:628
cluster_stats_messages_ping_received:319
cluster_stats_messages_pong_received:304
cluster_stats_messages_meet_received:5
cluster_stats_messages_received:628
total_cluster_links_buffer_limit_exceeded:0
```

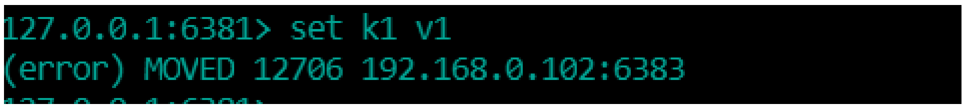


```
127.0.0.1:6381> cluster nodes
0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384@16384 slave ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 0 1683951536000 3 connected
a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381@16381 myself,master - 0 1683951535000 1 connected 0-5460
e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385@16385 slave a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 0 1683951537233 1 connected
f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382@16382 master - 0 1683951536228 2 connected 5461-10922
e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386@16386 slave f6cec3532d7545811808c67d2d4acec9f8858f4b 0 1683951535000 2 connected
ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383@16383 master - 0 1683951536000 3 connected 10923-16383
```

2、主从容错切换迁移案例

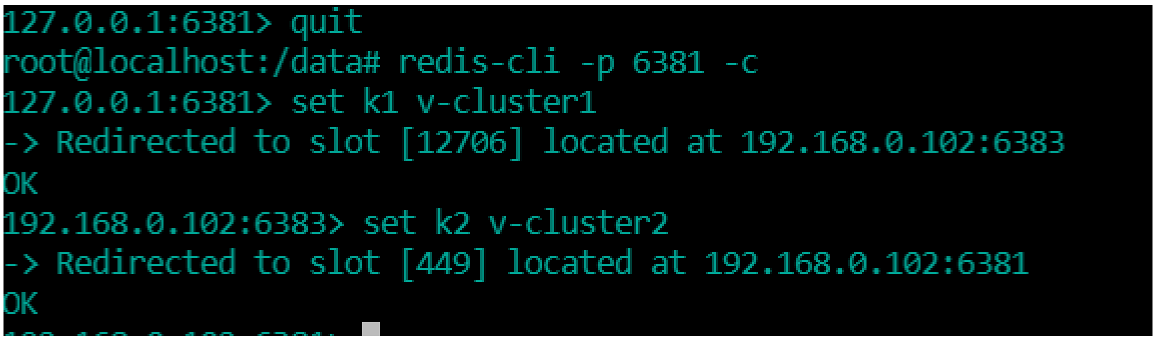
数据读写存储

- 启动6机构成的集群
- 对6381新增两个key
- 防止路由失效加参数-c并新增两个key



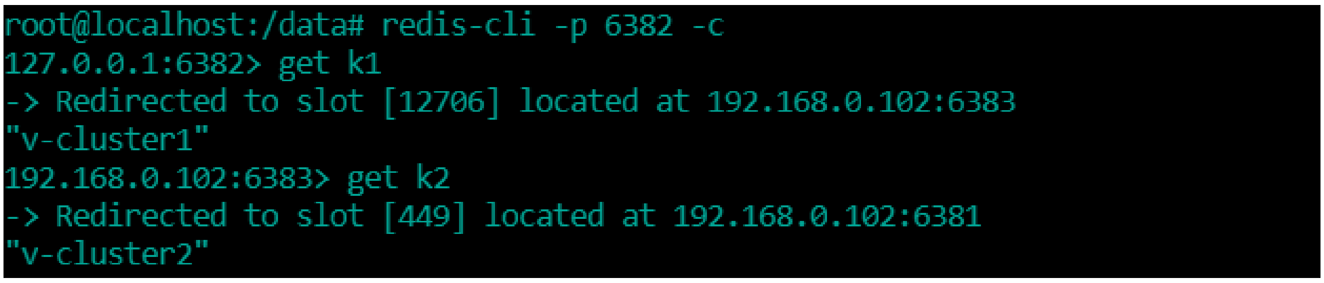
```
127.0.0.1:6381> set k1 v1
(error) MOVED 12706 192.168.0.102:6383
```

```
redis-cli -p 6381 -c
```



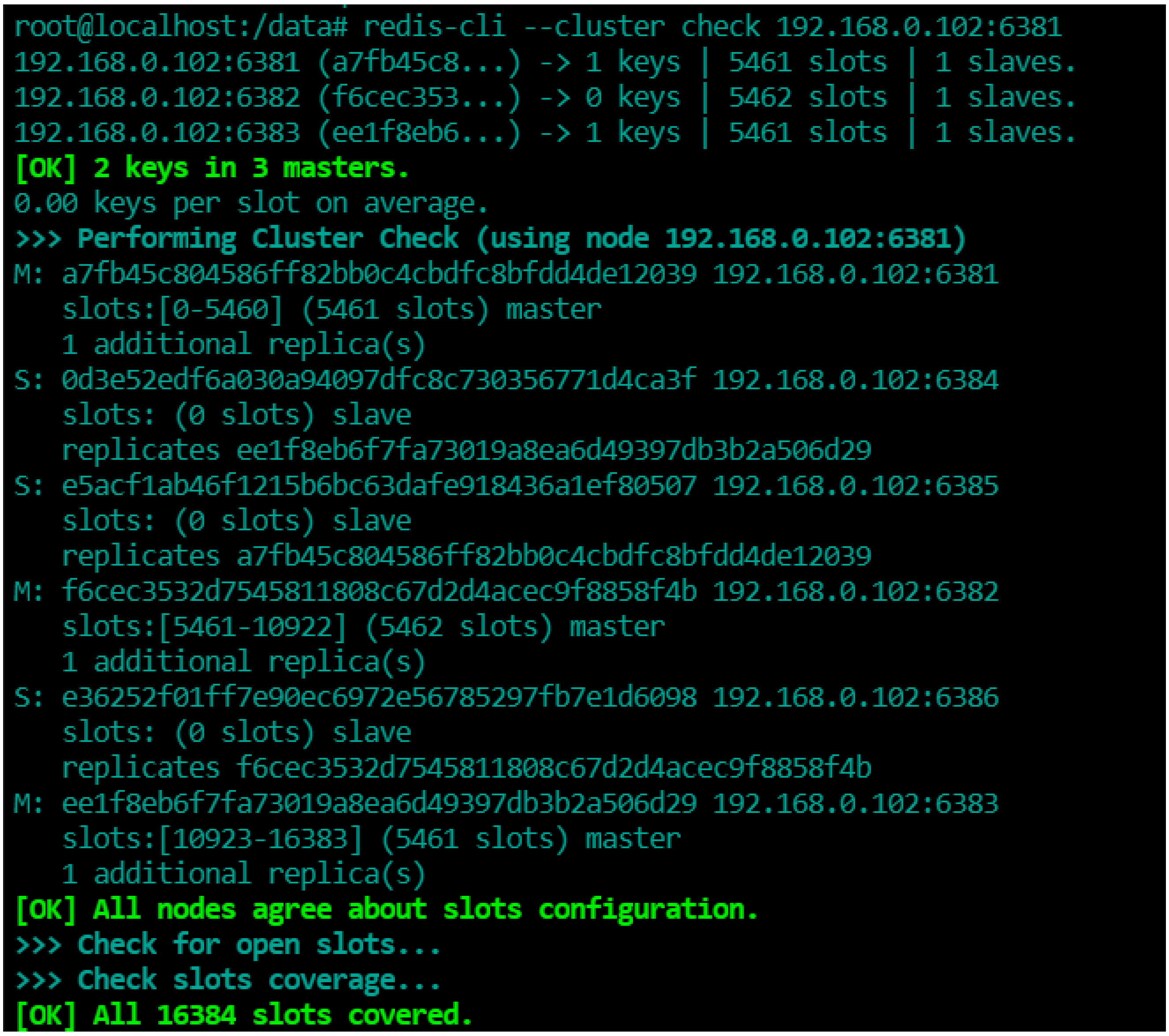
```
127.0.0.1:6381> quit
root@localhost:/data# redis-cli -p 6381 -c
127.0.0.1:6381> set k1 v-cluster1
-> Redirected to slot [12706] located at 192.168.0.102:6383
OK
192.168.0.102:6383> set k2 v-cluster2
-> Redirected to slot [449] located at 192.168.0.102:6381
OK
```

- 查看集群信息



```
root@localhost:/data# redis-cli -p 6382 -c
127.0.0.1:6382> get k1
-> Redirected to slot [12706] located at 192.168.0.102:6383
"v-cluster1"
192.168.0.102:6383> get k2
-> Redirected to slot [449] located at 192.168.0.102:6381
"v-cluster2"
```

```
redis-cli --cluster check 192.168.0.102:6381
```



```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 1 keys | 5461 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 5462 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 5461 slots | 1 slaves.
[OK] 2 keys in 3 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

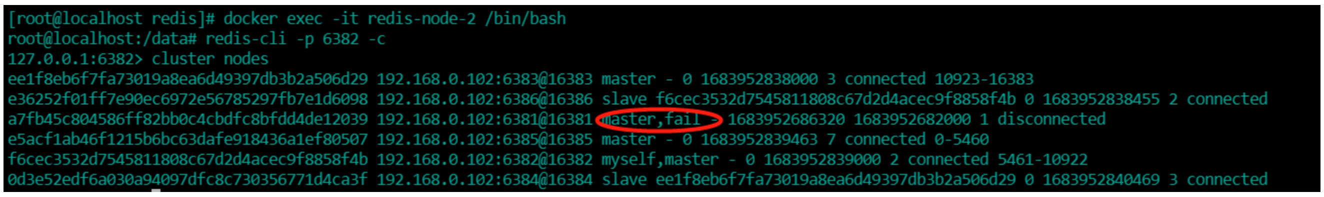
容错切换迁移

- 主6381和从机切换，先停止主机6381

  ```
  6381主机停了，对应的真实从机上位

  6381作为1号主机分配的从机以实际情况为准 ，具体是几号机器就是几号
  ```

- 再次查看集群信息

  ```
  docker exec -it redis-node-2 /bin/bash
  redis-cli -p 6382 -c
  cluster nodes
  ```

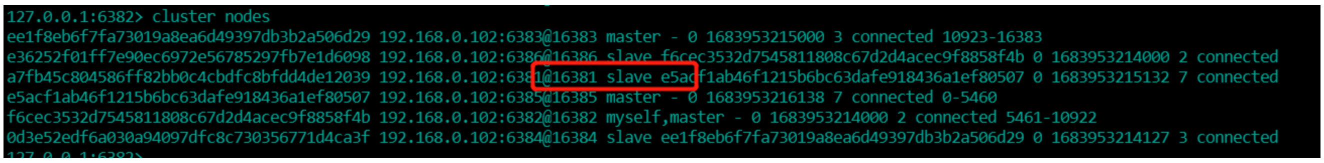
  ```
  [root@localhost redis]# docker exec -it redis-node-2 /bin/bash
  root@localhost:/data# redis-cli -p 6382 -c
  127.0.0.1:6382> cluster nodes
  ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383@16383 master - 0 1683952838000 3 connected 10923-16383
  e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386@16386 slave f6cec3532d7545811808c67d2d4acec9f8858f4b 0 1683952838455 2 connected
  a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381@16381 master,fail - 1683952686320 1683952682000 1 disconnected
  e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385@16385 master - 0 1683952839463 7 connected 0-5460
  f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382@16382 myself,master - 0 1683952839000 2 connected 5461-10922
  0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384@16384 slave ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 0 1683952840469 3 connected
  ```

  可以看到原来6381节点的从节点6385已经变成了主节点

- 还原之前的3主3从

  先启动6381(此时6381节点还是从节点，并不会自动变为主节点)

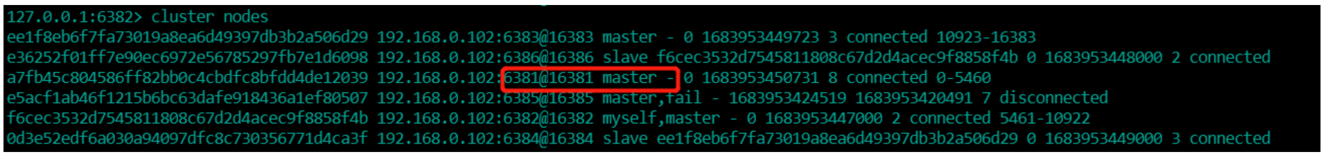
  ```
  docker start redis-node-1
  ```

  ```
  127.0.0.1:6382> cluster nodes
  ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383@16383 master - 0 1683953215000 3 connected 10923-16383
  e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386@16386 slave f6cec3532d7545811808c67d2d4acec9f8858f4b 0 1683953214000 2 connected
  a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381@16381 slave e5acf1ab46f1215b6bc63dafe918436a1ef80507 0 1683953215132 7 connected
  e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385@16385 master - 0 1683953216138 7 connected 0-5460
  f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382@16382 myself,master - 0 1683953214000 2 connected 5461-10922
  0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384@16384 slave ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 0 1683953214127 3 connected
  ```

  再停6385(让系统自动主从切换)

  ```
  docker stop redis-node-5
  ```

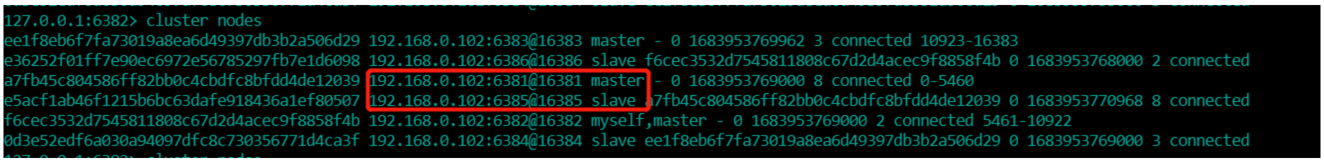
  

  ```
  127.0.0.1:6382> cluster nodes
  ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383@16383 master - 0 1683953449723 3 connected 10923-16383
  e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386@16386 slave f6cec3532d7545811808c67d2d4acec9f8858f4b 0 1683953448000 2 connected
  a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381@16381 master - 0 1683953450731 8 connected 0-5460
  e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385@16385 master,fail - 1683953424519 1683953420491 7 disconnected
  f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382@16382 myself,master - 0 1683953447000 2 connected 5461-10922
  0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384@16384 slave ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 0 1683953449000 3 connected
  ```

  可以看到6381又变成了主节点

  再启6385

  ```
  docker start redis-node-5
  ```

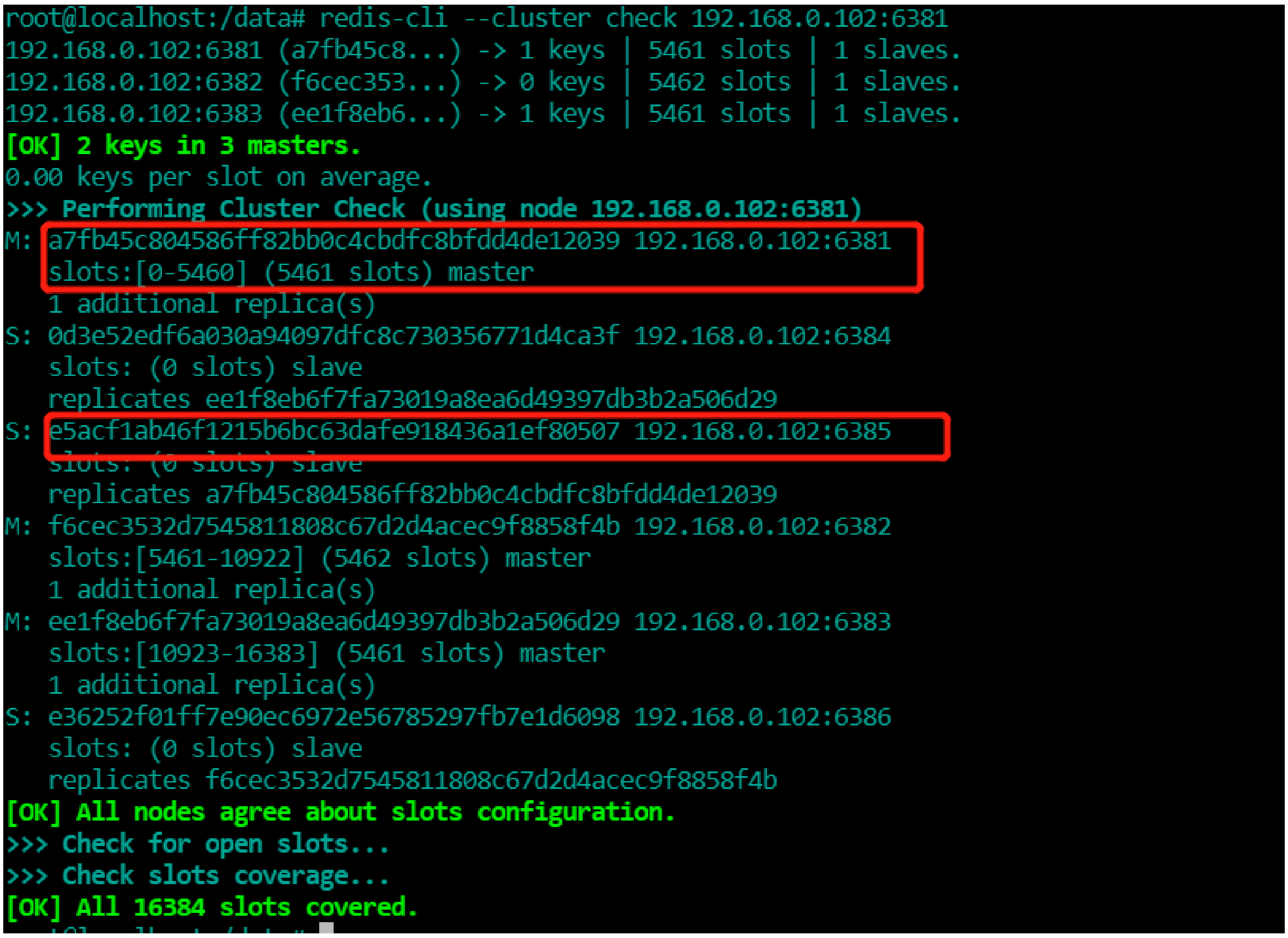


```
127.0.0.1:6382> cluster nodes
ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383@16383 master - 0 1683953769962 3 connected 10923-16383
e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386@16386 slave f6cec3532d7545811808c67d2d4acec9f8858f4b 0 1683953768000 2 connected
a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381@16381 master - 0 1683953769000 8 connected 0-5460
e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385@16385 slave a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 0 1683953770968 8 connected
f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382@16382 myself,master - 0 1683953769000 2 connected 5461-10922
0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384@16384 slave ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 0 1683953769000 3 connected
```

集群恢复正常

- 查看集群状态

```
docker exec -it redis-node-2 /bin/bash
redis-cli --cluster check 192.168.0.102:6381
```



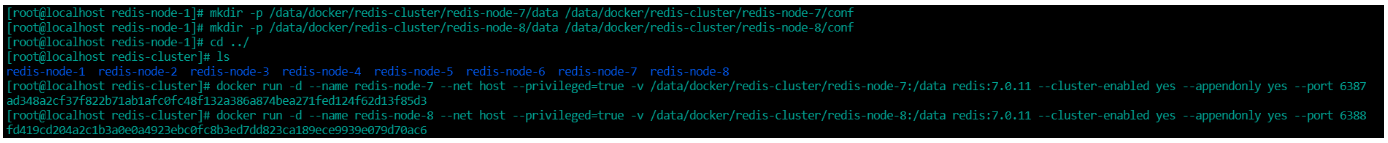
```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 1 keys | 5461 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 5462 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 5461 slots | 1 slaves.
[OK] 2 keys in 3 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

3、主从扩容案例

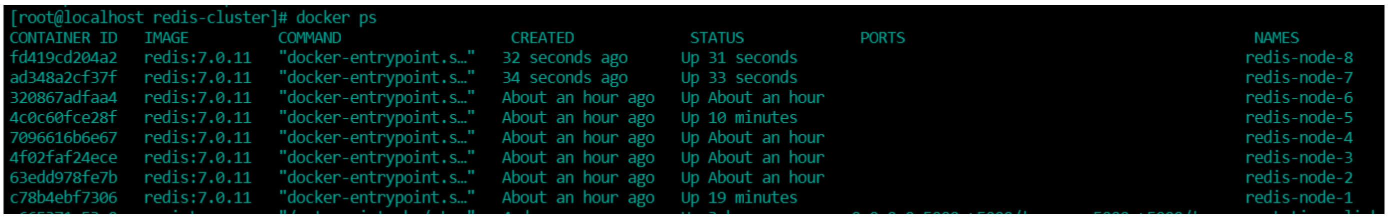
新建6387、6388两个节点+新建启动+查看是否8节点

```
mkdir -p /data/docker/redis-cluster/redis-node-7/data /data/docker/redis-cluster/redis-node-7/conf
mkdir -p /data/docker/redis-cluster/redis-node-8/data /data/docker/redis-cluster/redis-node-8/conf

docker run -d --name redis-node-7 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-7:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port
```

```
6387
docker run -d --name redis-node-8 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-8:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6388

docker ps
```



```
[root@localhost redis-node-1]# mkdir -p /data/docker/redis-cluster/redis-node-7/data /data/docker/redis-cluster/redis-node-7/conf
[root@localhost redis-node-1]# mkdir -p /data/docker/redis-cluster/redis-node-8/data /data/docker/redis-cluster/redis-node-8/conf
[root@localhost redis-node-1]# cd ../
[root@localhost redis-cluster]# ls
redis-node-1  redis-node-2  redis-node-3  redis-node-4  redis-node-5  redis-node-6  redis-node-7  redis-node-8
[root@localhost redis-cluster]# docker run -d --name redis-node-7 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-7:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6387
ad348a2cf37f822b71ab1afc0fc48f132a386a874bea271fed124f62d13f85d3
[root@localhost redis-cluster]# docker run -d --name redis-node-8 --net host --privileged=true -v /data/docker/redis-cluster/redis-node-8:/data redis:7.0.11 --cluster-enabled yes --appendonly yes --port 6388
fd419cd204a2c1b3a0e0a4923ebc0fc8b3ed7dd823ca189ece9939e079d70ac6
```



```
[root@localhost redis-cluster]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS              PORTS                                         NAMES
fd419cd204a2   redis:7.0.11   "docker-entrypoint.s…"   32 seconds ago      Up 31 seconds                                                     redis-node-8
ad348a2cf37f   redis:7.0.11   "docker-entrypoint.s…"   34 seconds ago      Up 33 seconds                                                     redis-node-7
320867adfaa4   redis:7.0.11   "docker-entrypoint.s…"   About an hour ago   Up About an hour                                                  redis-node-6
4c0c60fce28f   redis:7.0.11   "docker-entrypoint.s…"   About an hour ago   Up 10 minutes                                                     redis-node-5
7096616b6e67   redis:7.0.11   "docker-entrypoint.s…"   About an hour ago   Up About an hour                                                  redis-node-4
4f02faf24ece   redis:7.0.11   "docker-entrypoint.s…"   About an hour ago   Up About an hour                                                  redis-node-3
63edd978fe7b   redis:7.0.11   "docker-entrypoint.s…"   About an hour ago   Up About an hour                                                  redis-node-2
c78b4ebf7306   redis:7.0.11   "docker-entrypoint.s…"   About an hour ago   Up 19 minutes                                                     redis-node-1
```

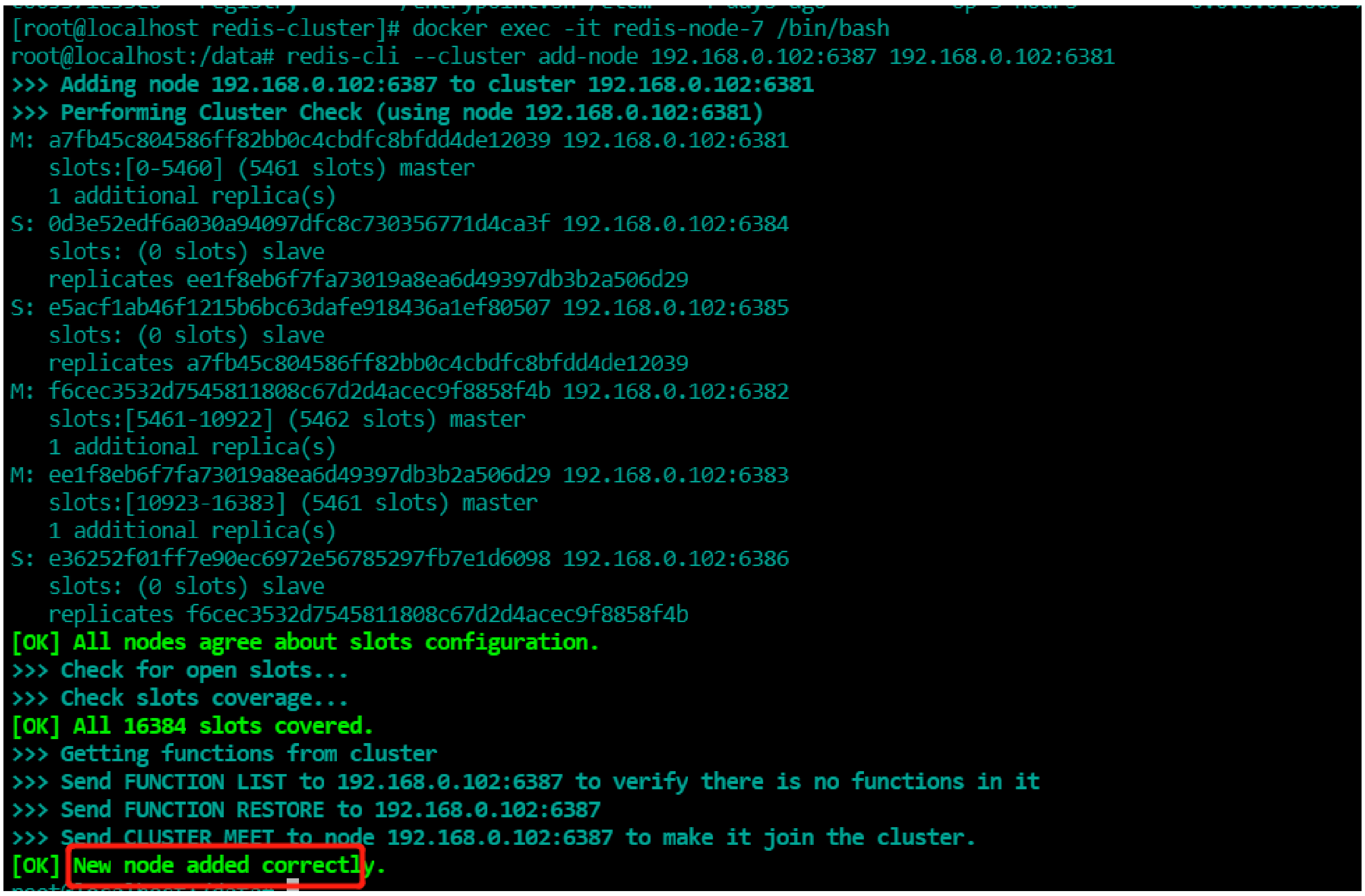
进入6378容器实例内部

```
docker exec -it redis-node-7 /bin/bash
```

将新增的6387节点(空槽号)作为master节点加入原集群

```
redis-cli --cluster add-node 192.168.0.102:6387 192.168.0.102:6381
```

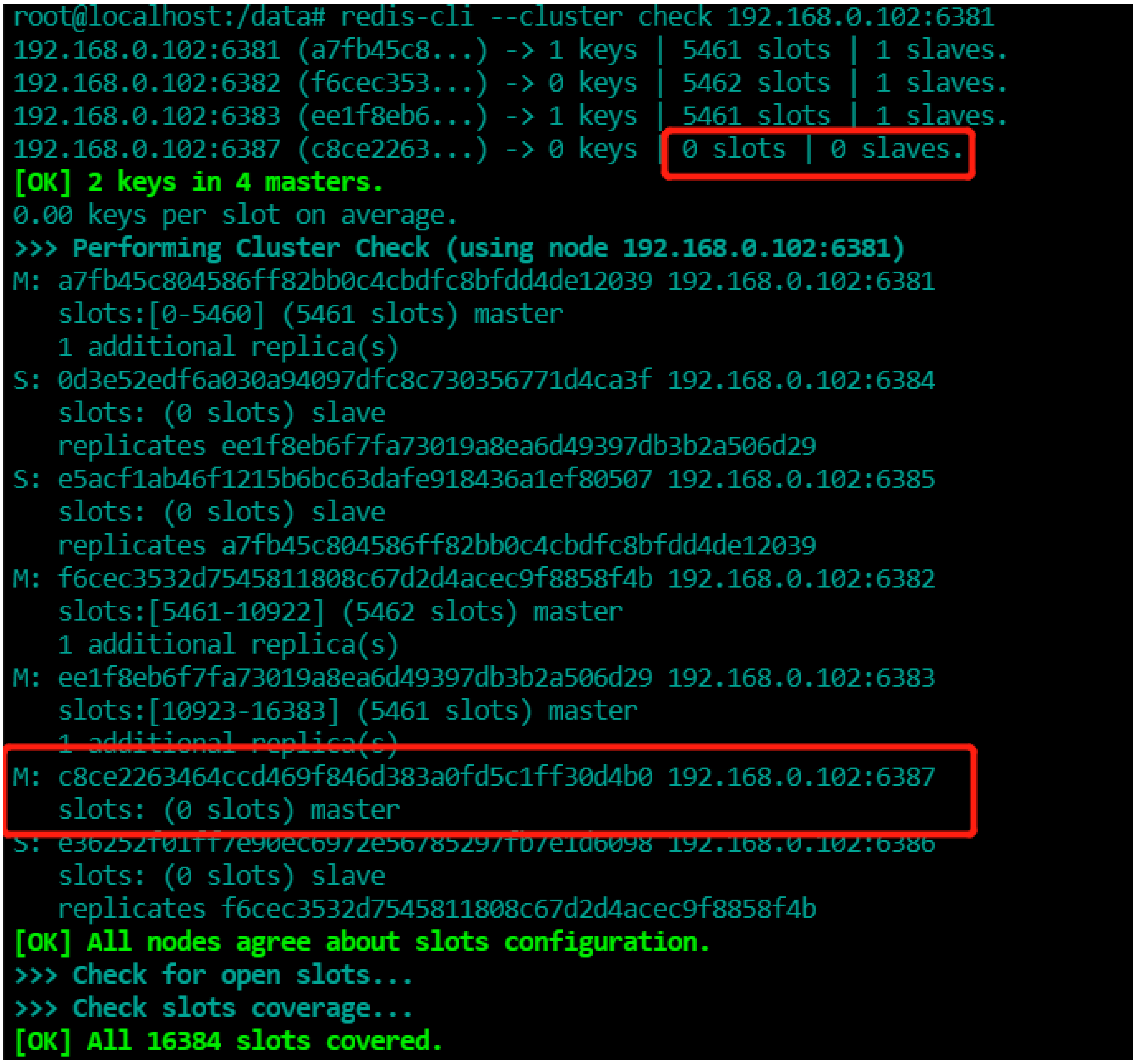
将新增的6387作为master节点加入集群 redis-cli --cluster add-node 自己实际IP地址:6387 自己实际IP地址:6381 6387 就是将要作为master新增节点 6381 就是原来集群节点里面的领路人，相当于6387拜拜6381的码头从而找到组织加入集群



```
[root@localhost redis-cluster]# docker exec -it redis-node-7 /bin/bash
root@localhost:/data# redis-cli --cluster add-node 192.168.0.102:6387 192.168.0.102:6381
>>> Adding node 192.168.0.102:6387 to cluster 192.168.0.102:6381
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
>>> Getting functions from cluster
>>> Send FUNCTION LIST to 192.168.0.102:6387 to verify there is no functions in it
>>> Send FUNCTION RESTORE to 192.168.0.102:6387
>>> Send CLUSTER MEET to node 192.168.0.102:6387 to make it join the cluster.
[OK] New node added correctly.
```

检查集群情况第1次

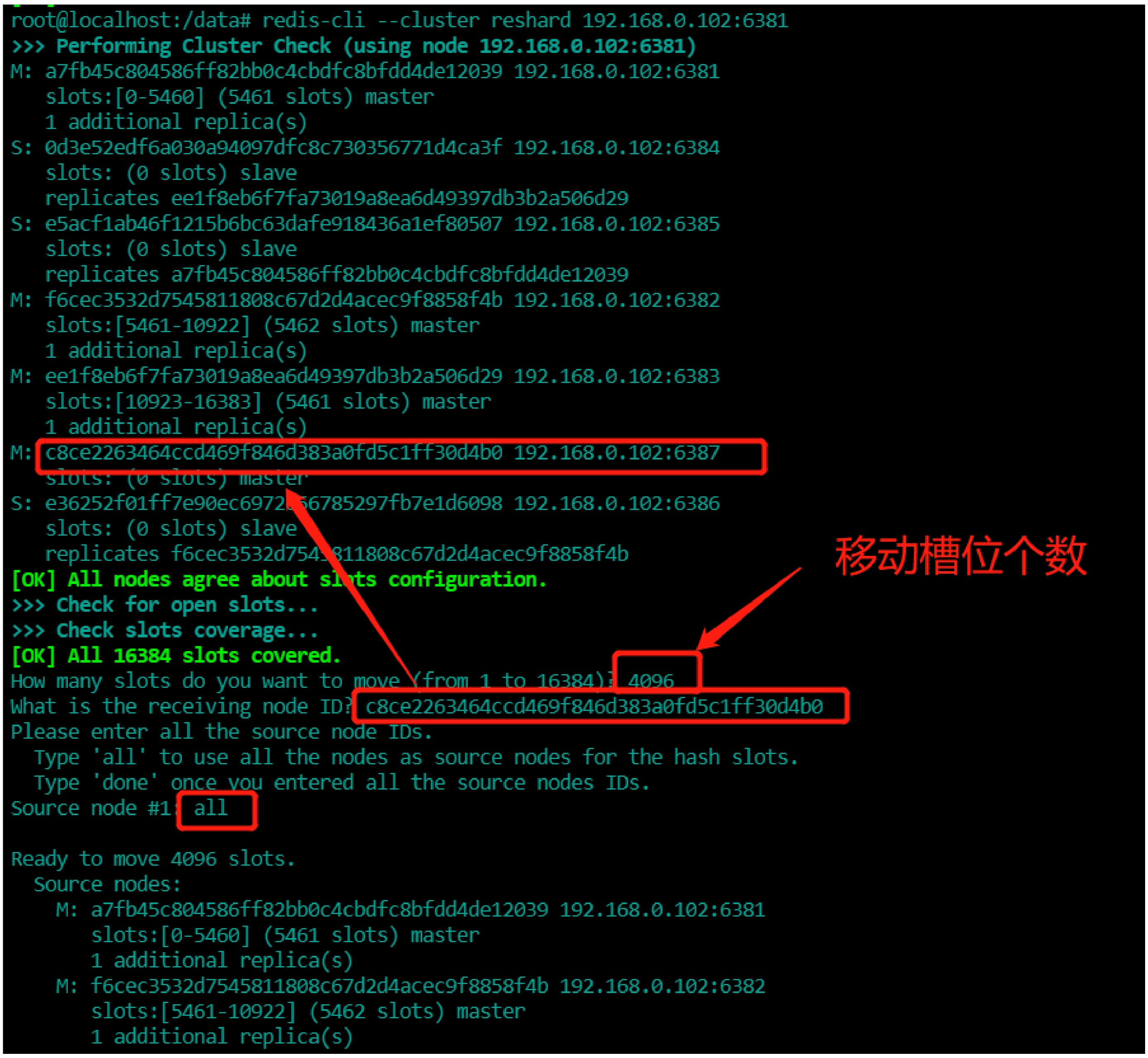
```
redis-cli --cluster check 192.168.0.102:6381
```



```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 1 keys | 5461 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 5462 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 5461 slots | 1 slaves.
192.168.0.102:6387 (c8ce2263...) -> 0 keys | 0 slots | 0 slaves.
[OK] 2 keys in 4 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots: (0 slots) master
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

重新分配槽号

```
redis-cli --cluster reshard 192.168.0.102:6381
```

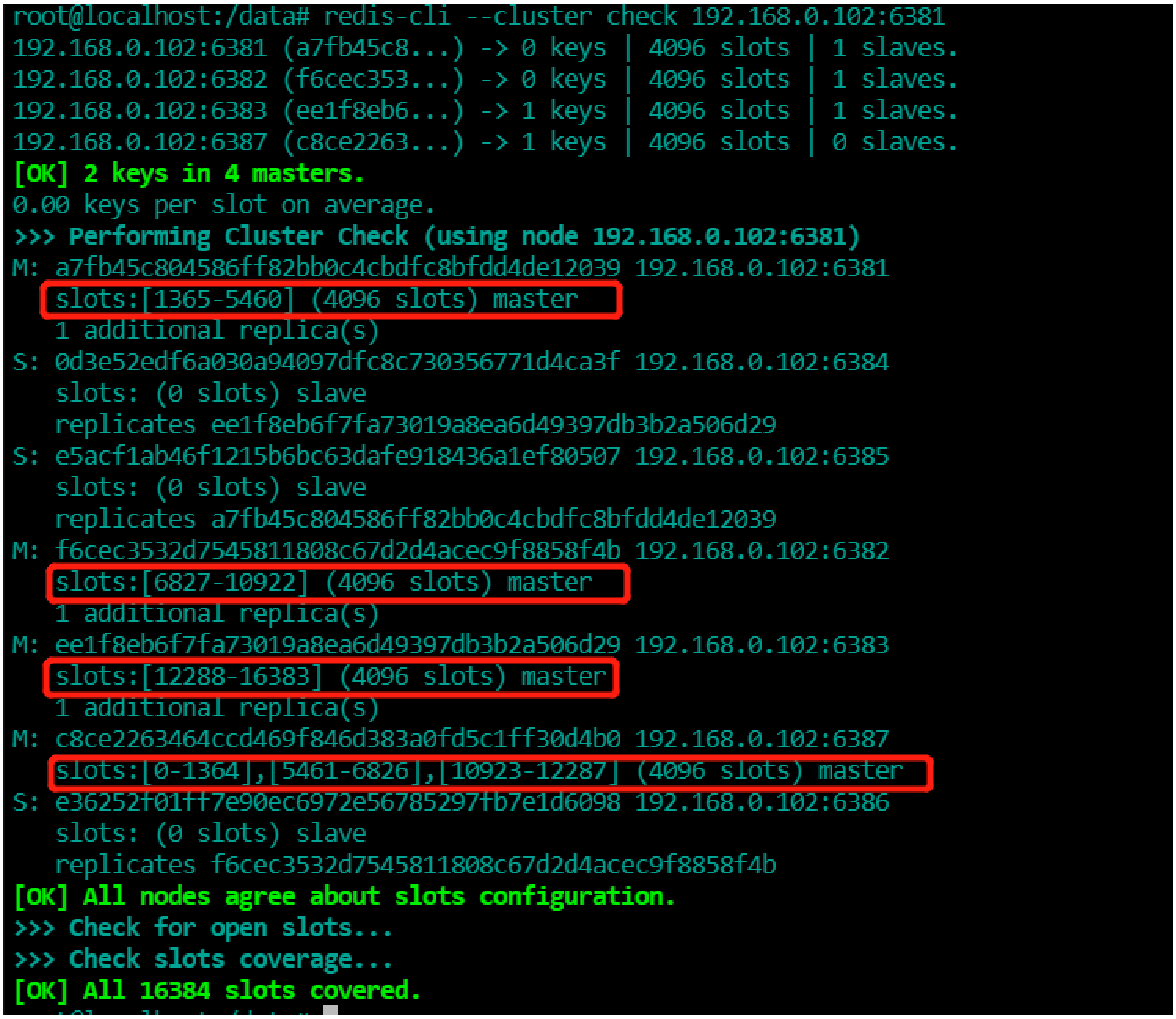


```
root@localhost:/data# redis-cli --cluster reshard 192.168.0.102:6381
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots: (0 slots) master
S: e36252f01ff7e90ec6972[illegible]56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[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? c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
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: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
       slots:[0-5460] (5461 slots) master
       1 additional replica(s)
    M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
       slots:[5461-10922] (5462 slots) master
       1 additional replica(s)
```

为了确保槽位的平均，移动槽位数计算公式：16384/master台数

检查集群情况第2次

```
redis-cli --cluster check 192.168.0.102:6381
```



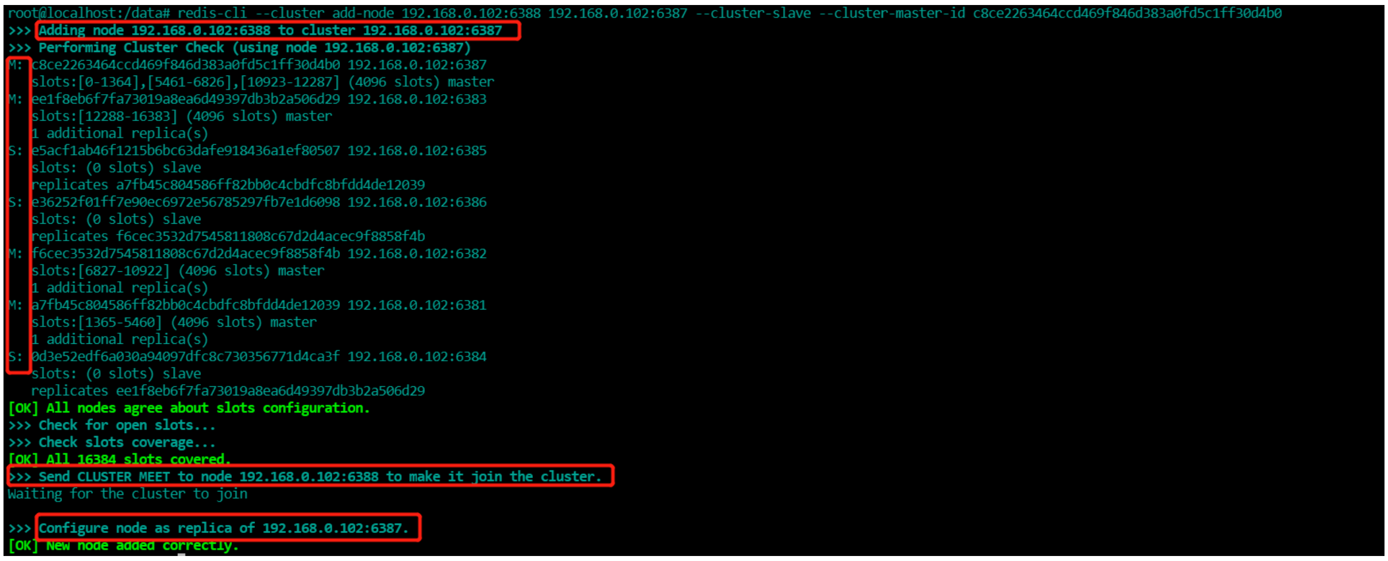
```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 4096 slots | 1 slaves.
192.168.0.102:6387 (c8ce2263...) -> 1 keys | 4096 slots | 0 slaves.
[OK] 2 keys in 4 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[1365-5460] (4096 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

可见6387上的4096个槽位是从其他三个master节点移动的

为什么6387是3个新的区间，以前的还是连续？ 重新分配成本太高，所以前3家各自匀出来一部分，从6381/6382/6383三个旧节点分别匀出1364个坑位给新节点6387

为主节点6387分配从节点6388

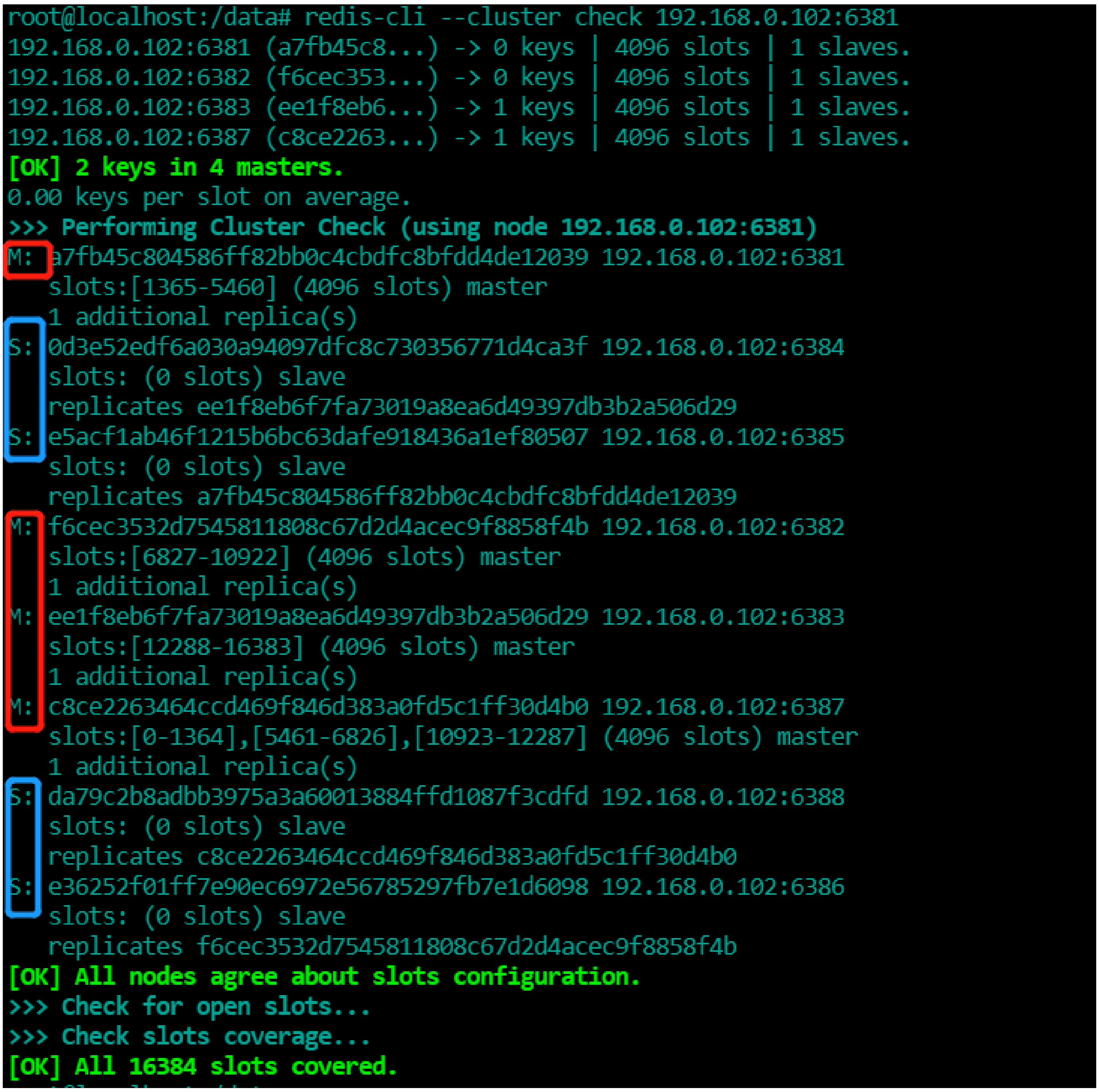
```
命令：redis-cli --cluster add-node ip:新slave端口 ip:新master端口 --cluster-slave --cluster-master-id 新主机节点ID

redis-cli --cluster add-node 192.168.0.102:6388 192.168.0.102:6387 --cluster-slave --cluster-master-id c8ce2263464ccd469f846d383a0fd5c1ff30d4b0-------这个是6387的编号，按照自己实际情况
```



```
root@localhost:/data# redis-cli --cluster add-node 192.168.0.102:6388 192.168.0.102:6387 --cluster-slave --cluster-master-id c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
>>> Adding node 192.168.0.102:6388 to cluster 192.168.0.102:6387
>>> Performing Cluster Check (using node 192.168.0.102:6387)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[1365-5460] (4096 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
>>> Send CLUSTER MEET to node 192.168.0.102:6388 to make it join the cluster.
Waiting for the cluster to join

>>> Configure node as replica of 192.168.0.102:6387.
[OK] New node added correctly.
```

检查集群情况第3次

```
redis-cli --cluster check 192.168.0.102:6381
```



```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 4096 slots | 1 slaves.
192.168.0.102:6387 (c8ce2263...) -> 1 keys | 4096 slots | 1 slaves.
[OK] 2 keys in 4 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[1365-5460] (4096 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
   1 additional replica(s)
S: da79c2b8adbb3975a3a60013884ffd1087f3cdfd 192.168.0.102:6388
   slots: (0 slots) slave
   replicates c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

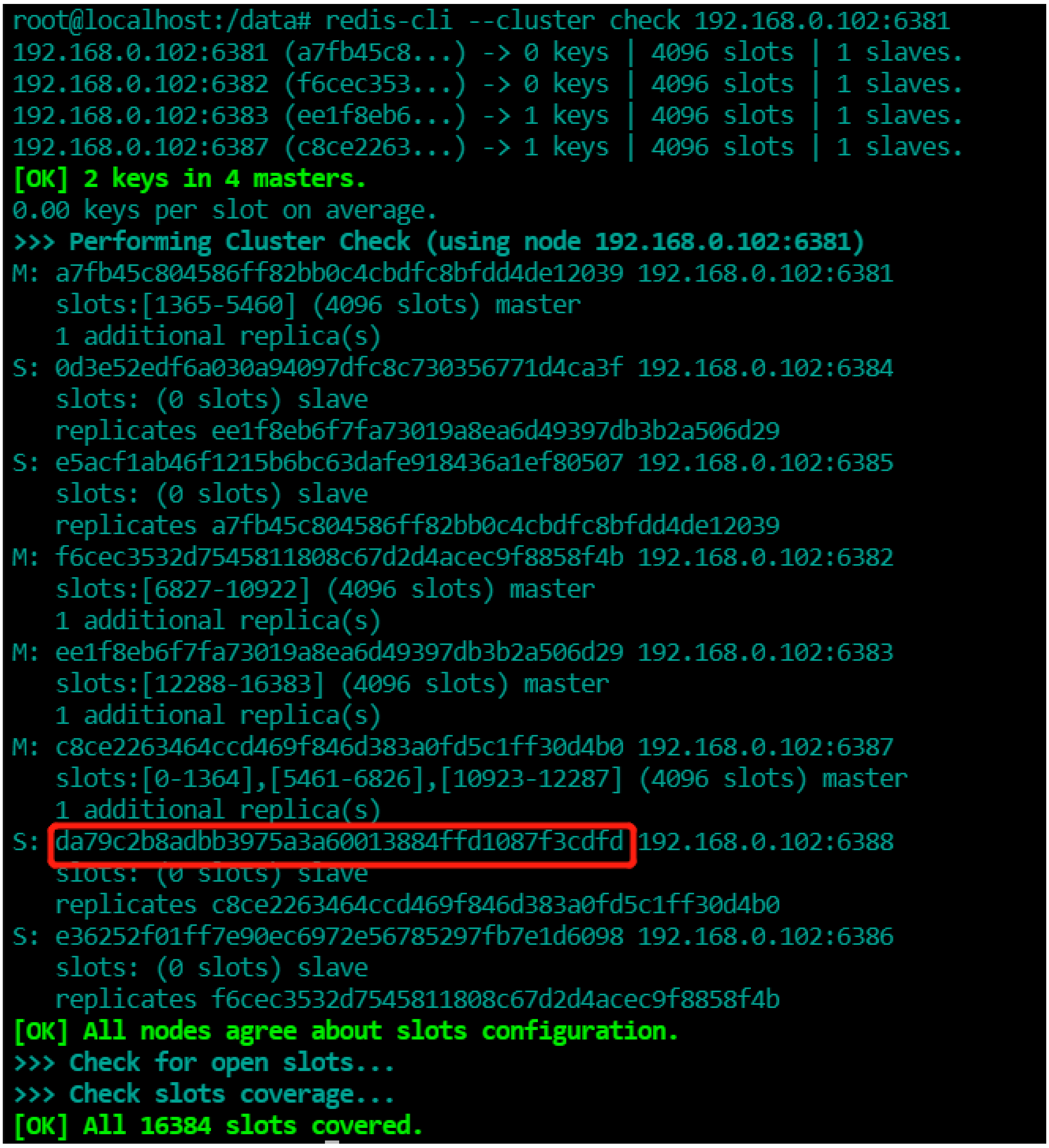
集群扩容完成

4、主从缩容案例

目的:下线6387和6388

检查集群情况1获得6388节点ID

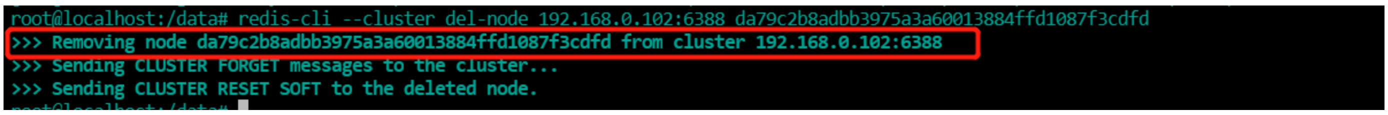
```
redis-cli --cluster check 192.168.0.102:6381
```



```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 4096 slots | 1 slaves.
192.168.0.102:6387 (c8ce2263...) -> 1 keys | 4096 slots | 1 slaves.
[OK] 2 keys in 4 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[1365-5460] (4096 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
   1 additional replica(s)
S: da79c2b8adbb3975a3a60013884ffd1087f3cdfd 192.168.0.102:6388
   slots: (0 slots) slave
   replicates c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

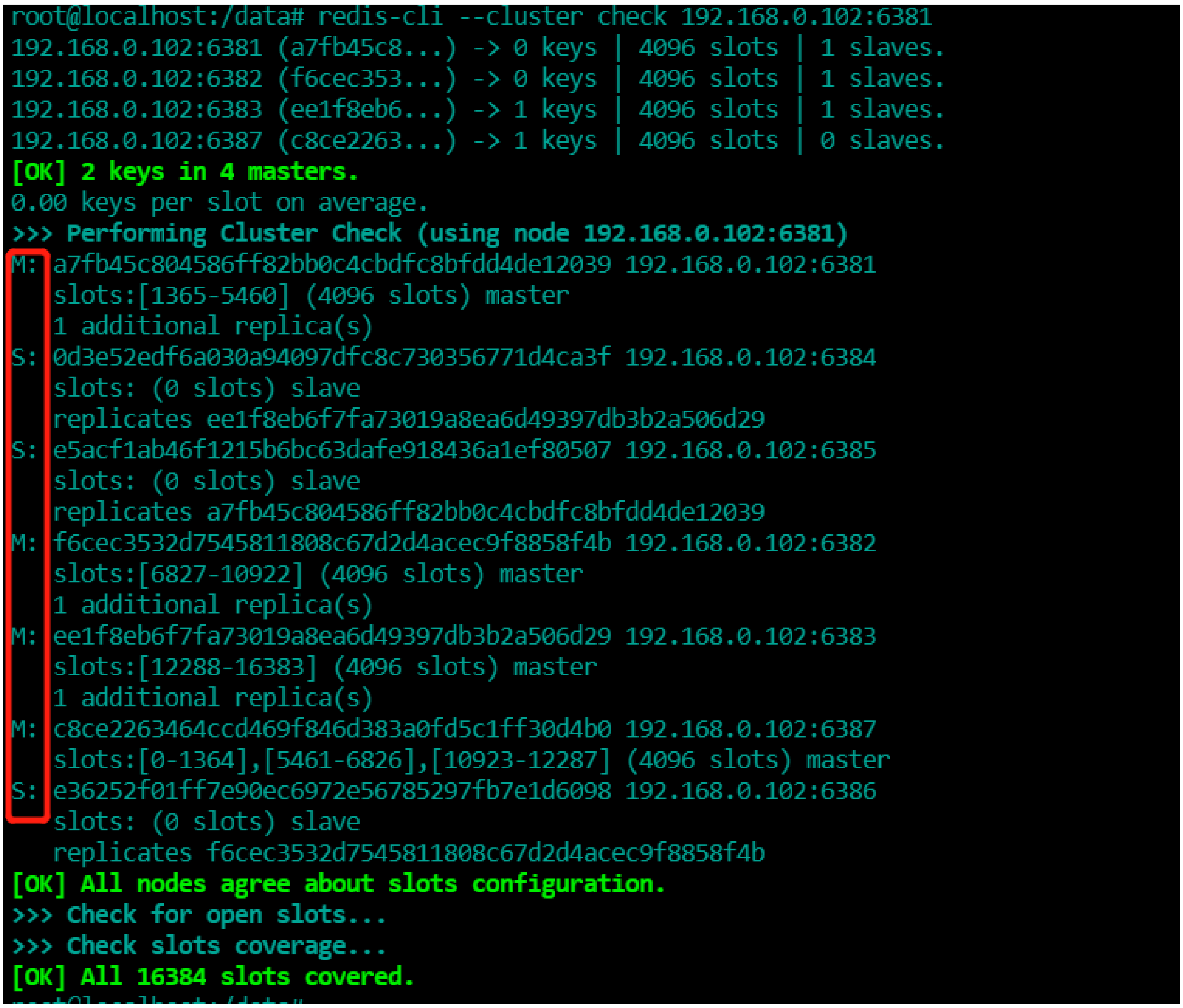
从集群中将从节点6388删除

```
命令：redis-cli --cluster del-node ip:从机端口 从机6388节点ID
redis-cli --cluster del-node 192.168.0.102:6388
da79c2b8adbb3975a3a60013884ffd1087f3cdfd
```



```
root@localhost:/data# redis-cli --cluster del-node 192.168.0.102:6388 da79c2b8adbb3975a3a60013884ffd1087f3cdfd
>>> Removing node da79c2b8adbb3975a3a60013884ffd1087f3cdfd from cluster 192.168.0.102:6388
>>> Sending CLUSTER FORGET messages to the cluster...
>>> Sending CLUSTER RESET SOFT to the deleted node.
root@localhost:/data#
```

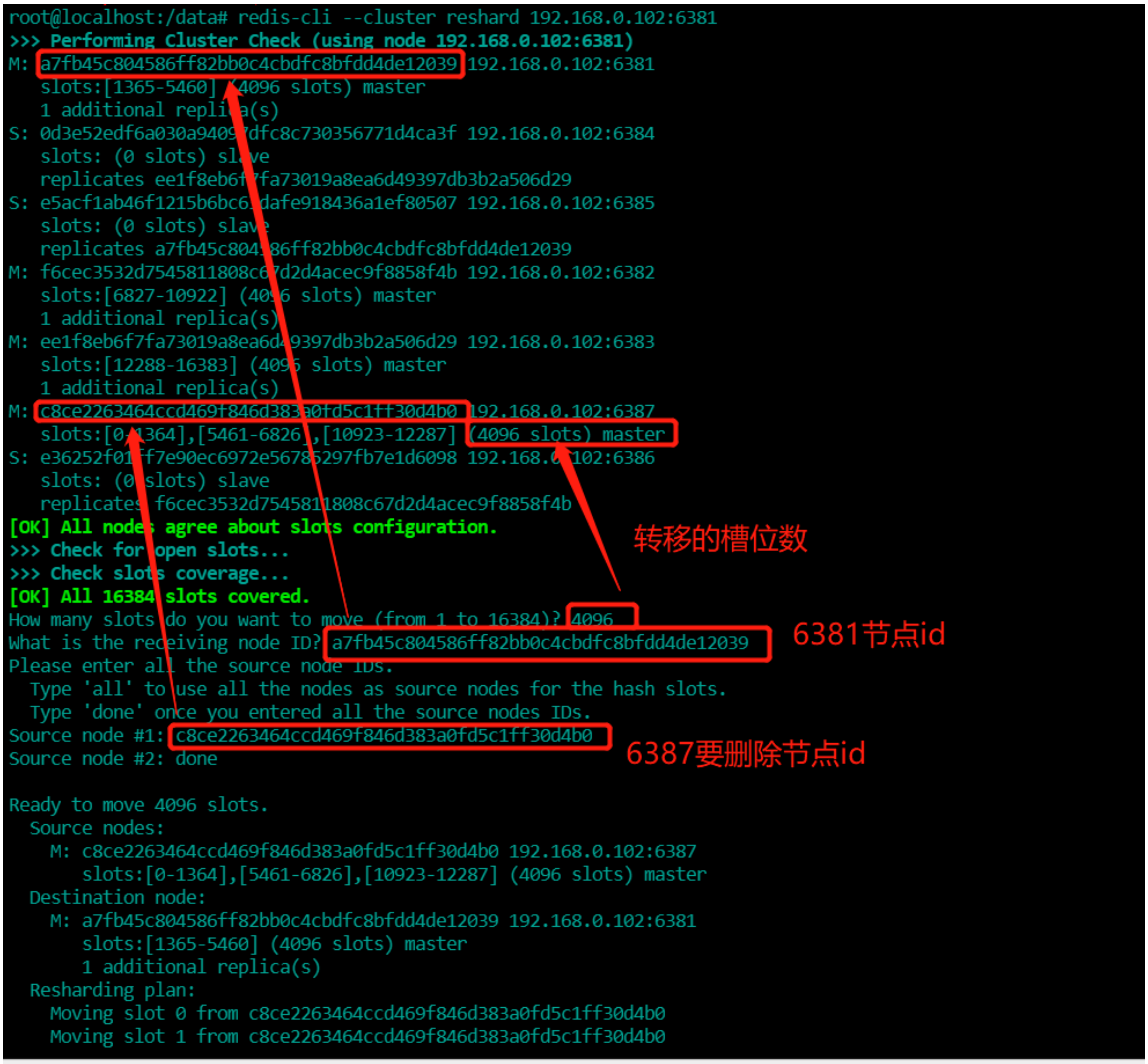
```
# 检查一下集群情况
redis-cli --cluster check 192.168.0.102:6388
```



```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 4096 slots | 1 slaves.
192.168.0.102:6387 (c8ce2263...) -> 1 keys | 4096 slots | 0 slaves.
[OK] 2 keys in 4 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[1365-5460] (4096 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

将6387的槽号清空，重新分配，本例将清出来的槽号都给6381

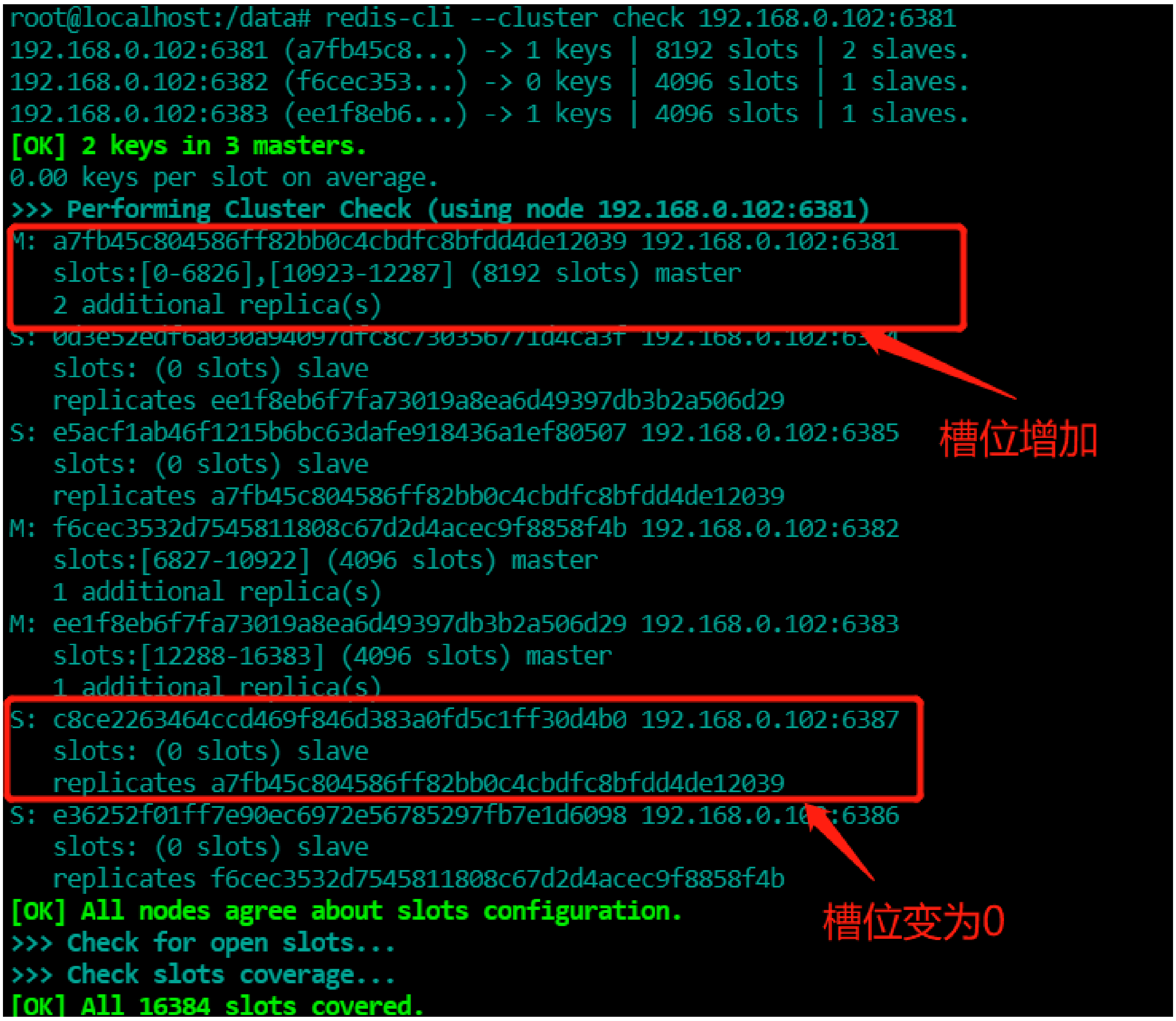
```
redis-cli --cluster reshard 192.168.0.102:6381
```



```
root@localhost:/data# redis-cli --cluster reshard 192.168.0.102:6381
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[1365-5460] (4096 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a9409 dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc6 dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
S: e36252f01 f7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[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? a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
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: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
Source node #2: done

Ready to move 4096 slots.
  Source nodes:
    M: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
       slots:[0-1364],[5461-6826],[10923-12287] (4096 slots) master
  Destination node:
    M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
       slots:[1365-5460] (4096 slots) master
       1 additional replica(s)
  Resharding plan:
    Moving slot 0 from c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
    Moving slot 1 from c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
```

检查集群情况第2次

```
redis-cli --cluster check 192.168.0.102:6381
```

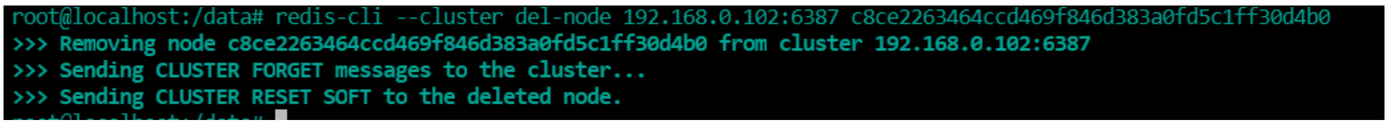


```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 1 keys | 8192 slots | 2 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 4096 slots | 1 slaves.
[OK] 2 keys in 3 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-6826],[10923-12287] (8192 slots) master
   2 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
S: c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 192.168.0.102:6387
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

6387节点槽位转移空后，立刻从主节点变成了从节点

集群删除6387节点

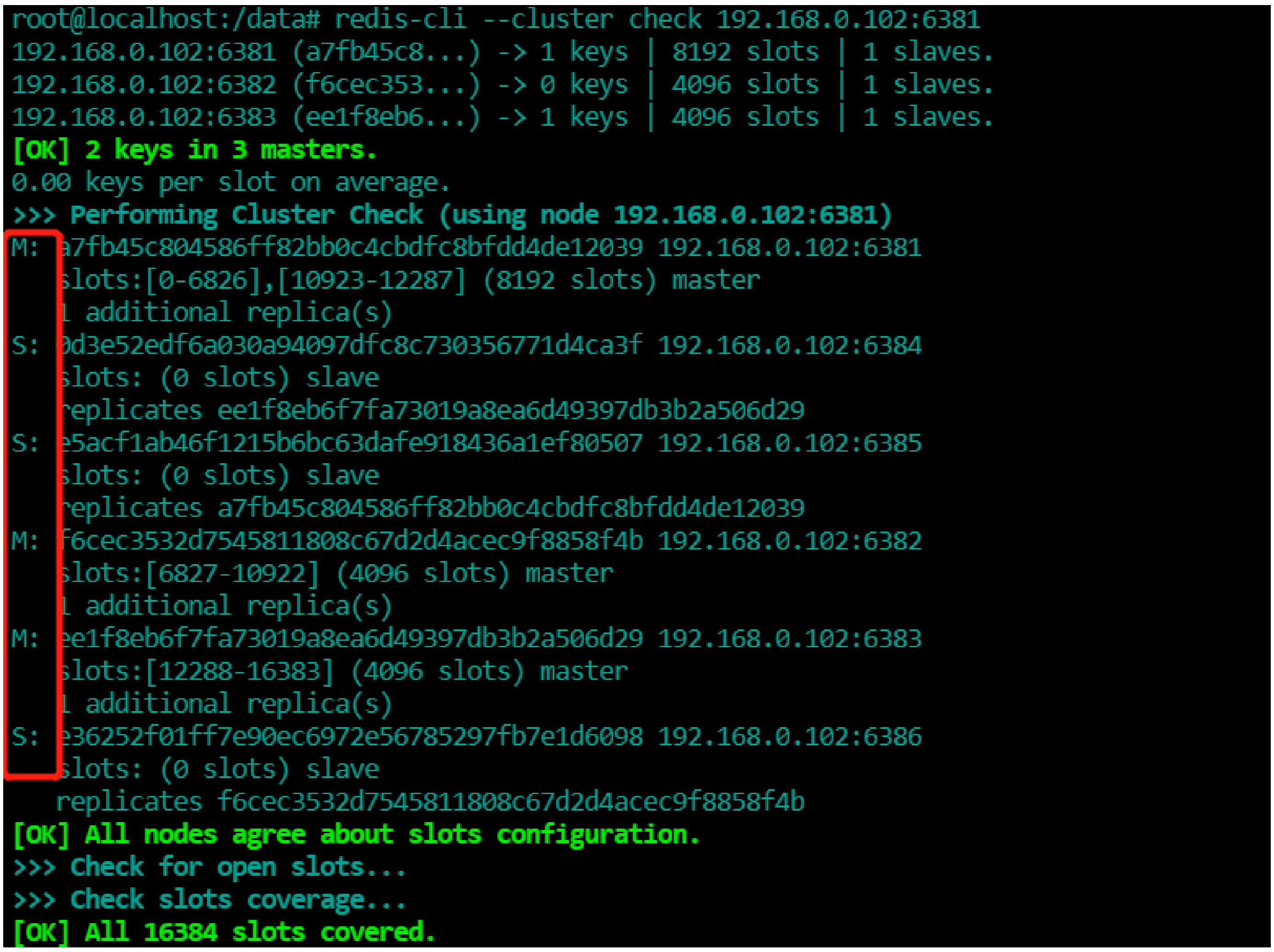
```
命令：redis-cli --cluster del-node ip:端口 6387节点ID
redis-cli --cluster del-node 192.168.0.102:6387
c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
```



```
root@localhost:/data# redis-cli --cluster del-node 192.168.0.102:6387 c8ce2263464ccd469f846d383a0fd5c1ff30d4b0
>>> Removing node c8ce2263464ccd469f846d383a0fd5c1ff30d4b0 from cluster 192.168.0.102:6387
>>> Sending CLUSTER FORGET messages to the cluster...
>>> Sending CLUSTER RESET SOFT to the deleted node.
root@localhost:/data#
```

检查集群情况第3次

```
redis-cli --cluster check 192.168.0.102:6381
```



```
root@localhost:/data# redis-cli --cluster check 192.168.0.102:6381
192.168.0.102:6381 (a7fb45c8...) -> 1 keys | 8192 slots | 1 slaves.
192.168.0.102:6382 (f6cec353...) -> 0 keys | 4096 slots | 1 slaves.
192.168.0.102:6383 (ee1f8eb6...) -> 1 keys | 4096 slots | 1 slaves.
[OK] 2 keys in 3 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.0.102:6381)
M: a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039 192.168.0.102:6381
   slots:[0-6826],[10923-12287] (8192 slots) master
   1 additional replica(s)
S: 0d3e52edf6a030a94097dfc8c730356771d4ca3f 192.168.0.102:6384
   slots: (0 slots) slave
   replicates ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29
S: e5acf1ab46f1215b6bc63dafe918436a1ef80507 192.168.0.102:6385
   slots: (0 slots) slave
   replicates a7fb45c804586ff82bb0c4cbdfc8bfdd4de12039
M: f6cec3532d7545811808c67d2d4acec9f8858f4b 192.168.0.102:6382
   slots:[6827-10922] (4096 slots) master
   1 additional replica(s)
M: ee1f8eb6f7fa73019a8ea6d49397db3b2a506d29 192.168.0.102:6383
   slots:[12288-16383] (4096 slots) master
   1 additional replica(s)
S: e36252f01ff7e90ec6972e56785297fb7e1d6098 192.168.0.102:6386
   slots: (0 slots) slave
   replicates f6cec3532d7545811808c67d2d4acec9f8858f4b
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

集群恢复3主3从

5、集群关闭、启动

docker集群

```
# 集群关闭
docker stop redis-node-1 redis-node-2 redis-node-3 redis-node-4 redis-node-5 redis-
node-6

# 集群启动
docker start redis-node-1 redis-node-2 redis-node-3 redis-node-4 redis-node-5 redis-
node-6
```

Liunxs

```
# 查看redis集群进程
ps -ef|grep redis
# kill 停止集群 在一台机器上时，如果是多个机器，一个一个进行停机即可
kill -9 进程id多个

# 重启集群，一个一个服务启动即可(进入redis/bin)
```

```
./redis-server /data/redis-cluster/redis01/redis.conf
./redis-server /data/redis-cluster/redis02/redis.conf
./redis-server /data/redis-cluster/redis03/redis.conf
./redis-server /data/redis-cluster/redis04/redis.conf
./redis-server /data/redis-cluster/redis05/redis.conf
./redis-server /data/redis-cluster/redis06/redis.conf
```