Redis-Sentinel模式

sentinel masters

sentinel slaves mymaster #其中mymaster是 master名字

带密码登录

redis-cli -h yourIp*-p yourPort -a youPassword*

不带密码登录

redis-cli -h yourIp*-p yourPort*

*auth password*

批量删除key 带密码的

redis-cli -h 10.100.2.41 -p 6379 -a yinghuo keys "zodi\*" | xargs redis-cli -h 10.100.2.41 -p 6379 -a yinghuo del

redis 持久化方式RDB（redis database）和AOF（append only file）

参考文档：<https://www.cnblogs.com/itdragon/p/7906481.html>

哨兵模式和集群模式

<https://www.cnblogs.com/ibethfy/p/9965902.html>

<https://www.cnblogs.com/kevingrace/p/9004460.html>

哨兵模式下

查看节点信息

src/redis-cli -h 10.10.2.38 –p 6379 info

[root@localhost redis-5.0.5]# src/redis-cli -h 10.10.2.38 –p 6379 info |grep role

role:master

查看sentinel节点信息

redis-cli -h 192.168.10.205 -p 26379 info Sentinel

每个redis服务，同机器，启动一个sentinel服务

Redis主从，在从服务器中添加replicaof host port

Sentinel都监控主redis服务

这样就行成了redis高可用方案

Redis集群搭建

参考文档<https://redis.io/topics/cluster-tutorial>

这里以6个借点来说明，三主三副

新建6个文件夹 7000至7005

6个文件夹中放入redis.conf，相应配置如下，端口需要修改一下

port 7000

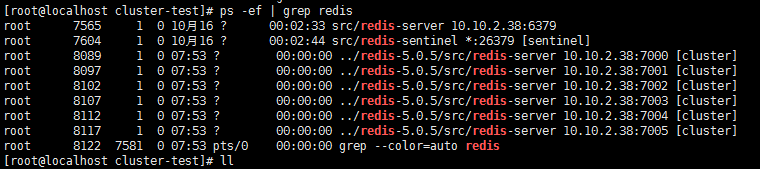
cluster-enabled yes

cluster-config-file nodes.conf

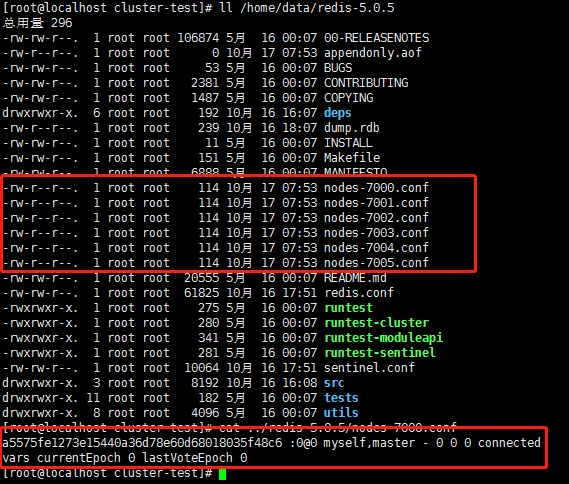
cluster-node-timeout 5000

appendonly yes

然后启动6个redis实例



产生6个文件



we need to create our cluster by writing some meaningful configuration to the nodes

create new clusters, check or reshard an existing cluster, and so forth

5版本后，可以方便的使用redis-cli

Create cluster

redis-cli --cluster create 127.0.0.1:7000 127.0.0.1:7001 \

127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005 \

--cluster-replicas 1

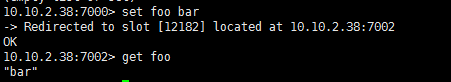
The option --cluster-replicas 1 means that we want a slave for every master created. The other arguments are the list of addresses of the instances I want to use to create the new cluster.

Redis-cli will propose you a configuration. Accept the proposed configuration by typing **yes**. The cluster will be configured and joined, which means, instances will be bootstrapped into talking with each other

创建集群



用redis-cli和redis cluster交互，在命令中加 –c参数



Set命令后，客户端重定向到7002这个redis实例

可以观察到redis-cli会重定向到正确的node。还有很多客户端做的很好，会缓存hash槽和节点地址的对应关系，然后直接用相应的连接到正确的客户端。对应关系在集群配置发生变化时会刷新的，比如失败转移，添加节点，减少节点。

#redis-cli批量删除key

redis-cli -n 7 keys "RP\*" | xargs redis-cli -n 7 del

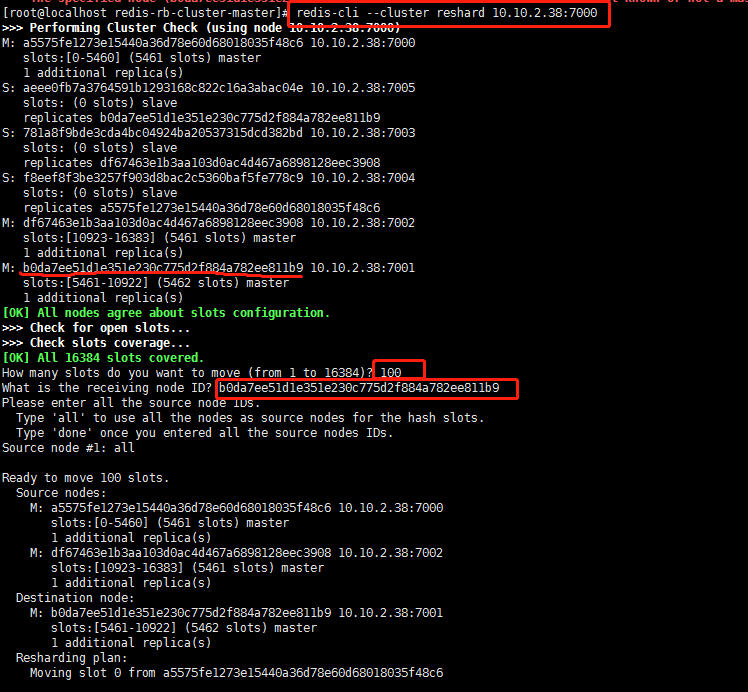
-n 7 指定数据库，相当于select 7

keys "RP\*" 就是 keys命令

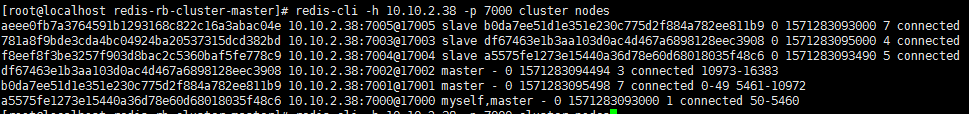
| 竖线是linux命令中的管道

Reshard重新分片

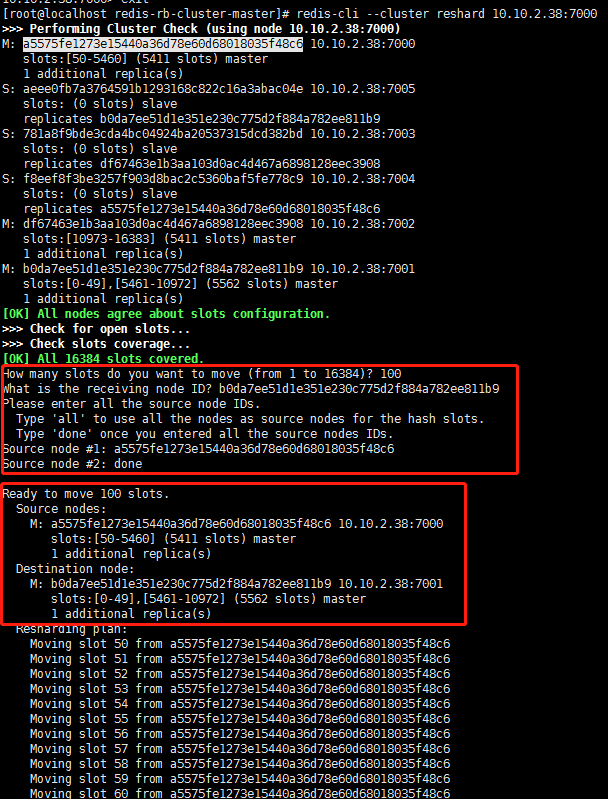
从其他节点平均将槽分到目标节点b0da7ee51d1e351e230c775d2f884a782ee811b9



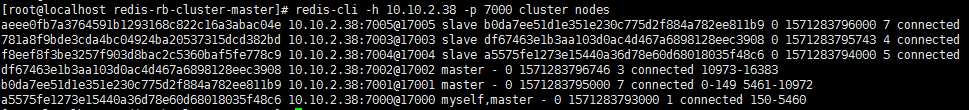
查看reshard后的槽情况



将某些节点上的槽分到指定节点b0da7ee51d1e351e230c775d2f884a782ee811b9



查看reshard后的槽情况



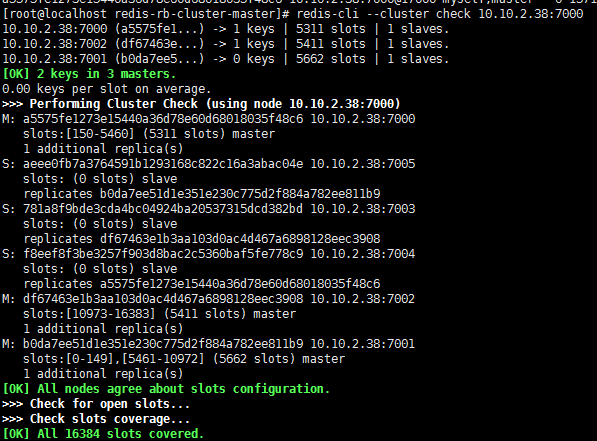
Reshard操作会询问你要reshard多少个槽，接收槽的节点id是什么，从哪些节点划分槽（all，或者指定节点，多个节点的话，是均匀的从源节点划分槽）。是一个比较麻的操作，还要确认槽的划分情况。

While the resharding is in progress you should be able to see your example program running unaffected. You can stop and restart it multiple times during the resharding if you want.

在reshard过程中不会影响客户端使用

检查集群健康状况

redis-cli --cluster check 10.10.2.38:7000



集群reshard可以不用交互的方式输入参数，如下：

redis-cli reshard <host>:<port> --cluster-from <node-id> --cluster-to <node-id> --cluster-slots <number of slots> --cluster-yes

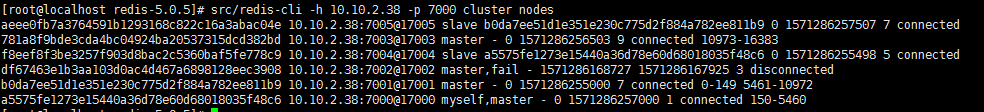
测试failover

用debug segfault

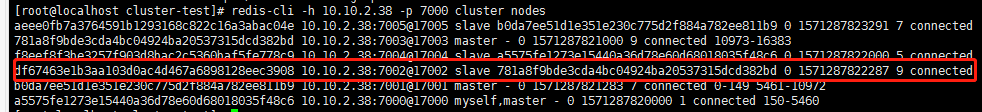
src/redis-cli -h 10.10.2.38 -p 7002 debug segfault



7002的主服务出问题，自动切换到slave上



失败的主节点，重启后是slave角色了



手动failover，升级master redis版本，在slave节点上执行手动failover命令

手动failover是比较安全的，不会造成数据丢失，处理程序会将数据同步完成后在切换到新的master。

客户端不会阻塞

在slave上面执行 cluster failover [force | takeover]

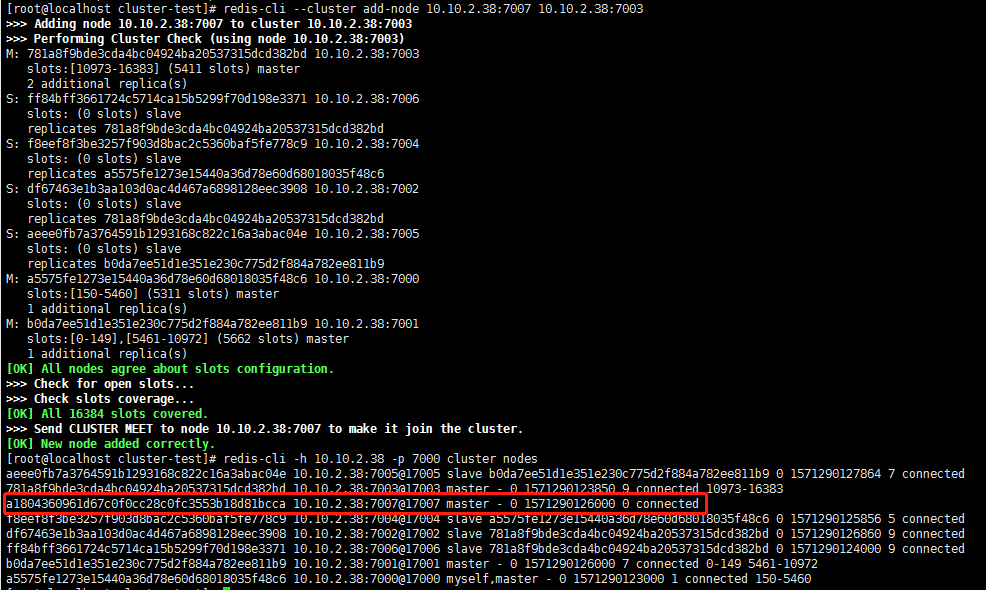
新增一个节点

可以是master，也可以是slave

先启动一个redis实例

然后用redis-cli客户端将空的实例加入redis集群

redis-cli --cluster add-node 10.10.2.38:7006 10.10.2.38:7000



新加进去的节点，是主节点，而且没有数据槽，需要重新reshard操作来分配hash槽。

将新加的节点作为replica，就是slave节点加进来

redis-cli --cluster add-node 127.0.0.1:7006 127.0.0.1:7000 --cluster-slave

a random master among the masters with less replicas

redis-cli --cluster add-node 127.0.0.1:7006 127.0.0.1:7000 --cluster-slave --cluster-master-id 3c3a0c74aae0b56170ccb03a76b60cfe7dc1912e

specify exactly what master you want to target

我们通常的做法是，将一个空的redis实例当做master节点加入集群，然后cluster replicate命令来变成slave节点。也可以。。。

在新加的空白主节点上执行

cluster replicate 3c3a0c74aae0b56170ccb03a76b60cfe7dc1912e

删除节点

redis-cli --cluster del-node 127.0.0.1:7000 `<node-id>`

You can remove a master node in the same way as well, **however in order to remove a master node it must be empty**. If the master is not empty you need to reshard data away from it to all the other master nodes before.

An alternative to remove a master node is to perform a manual failover of it over one of its slaves and remove the node after it turned into a slave of the new master. Obviously this does not help when you want to reduce the actual number of masters in your cluster, in that case, a resharding is needed.

副本迁移

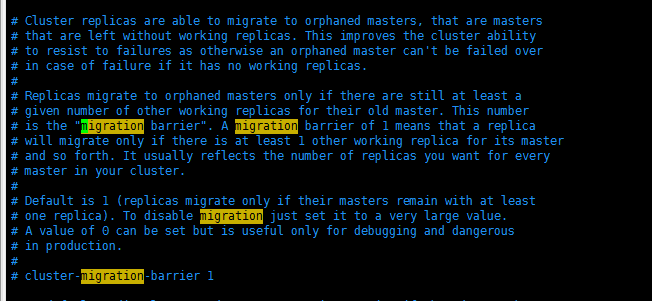
CLUSTER REPLICATE <master-node-id>

副本自动迁移，可以提高集群的可靠性

To improve reliability of the system we have the option to add additional replicas to every master, but this is expensive. Replica migration allows to add more slaves to just a few masters. So you have 10 masters with 1 slave each, for a total of 20 instances. However you add, for example, 3 instances more as slaves of some of your masters, so certain masters will have more than a single slave.

So what you should know about replicas migration in short?

* The cluster will try to migrate a replica from the master that has the greatest number of replicas in a given moment.
* To benefit from replica migration you have just to add a few more replicas to a single master in your cluster, it does not matter what master.
* There is a configuration parameter that controls the replica migration feature that is called cluster-migration-barrier: you can read more about it in the example redis.conf file provided with Redis Cluster.



数据迁移到redis集群