**Redis集群**

目录

[**1. Redis集群的两种模式 1**](#_Toc535421353)

[**1.1 Sentinel哨兵模式 2**](#_Toc535421354)

[**1.2 Cluster模式 2**](#_Toc535421355)

[**2. Cluster模式 3**](#_Toc535421356)

[**2.1 修改配置文件 4**](#_Toc535421357)

[**2.2 启动容器 4**](#_Toc535421358)

[**2.3 建立集群关系 6**](#_Toc535421359)

[**3. Sentinel哨兵模式 8**](#_Toc535421360)

[**3.1 Master结点 8**](#_Toc535421361)

[**3.2 Slave结点 9**](#_Toc535421362)

[**3.3 sentinel哨兵 10**](#_Toc535421363)

[**3.4 测试 11**](#_Toc535421364)

[**4. 问题和解决方法 11**](#_Toc535421365)

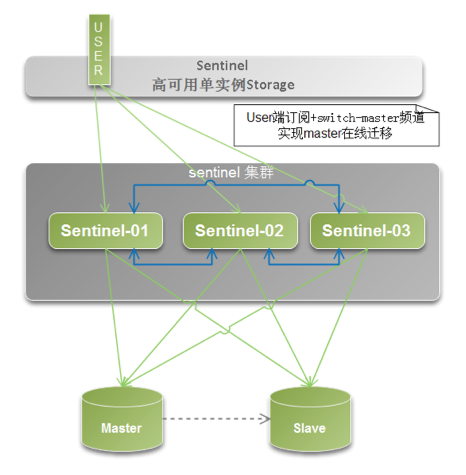
[**4.1 Cluster集群模式问题 11**](#_Toc535421366)

[**5. 扩展 14**](#_Toc535421367)

[**5.1 redis-cli --cluster常用命令 14**](#_Toc535421368)

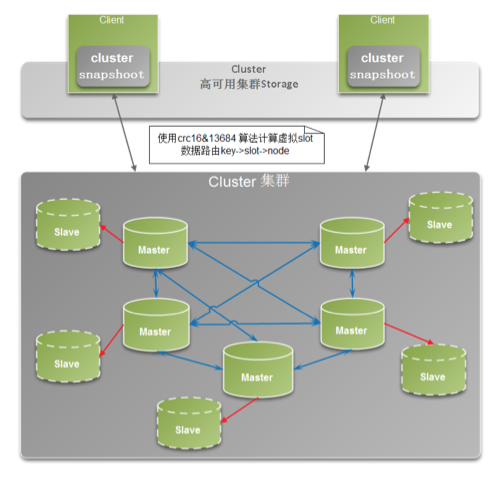
1. **Redis集群的两种模式**
   1. **Sentinel哨兵模式**

Redis-Sentinel是Redis官方推荐的高可用性(HA)解决方案，当用Redis做Master-slave的高可用方案时，假如master宕机了，Redis本身(包括它的很多客户端)都没有实现自动进行主备切换，而Redis-sentinel本身也是一个独立运行的进程，它能监控多个master-slave集群，发现master宕机后能进行自动切换。



* 1. **Cluster模式**

Redis Cluster是Redis的分布式解决方案，在Redis 3.0版本正式推出的，有效解决了Redis分布式方面的需求。当遇到单机内存、并发、流量等瓶颈时，可以采用Cluster架构达到负载均衡的目的。



**注：红线为主从复制**

1. **Cluster模式**

****

## 2.1 修改配置文件

redis.conf（nodes.conf）

1. port  6379  # 修改端口号，从6379到6384
2. cluster-enabled  yes  # 开启cluster，去掉注释
3. cluster-config-file  nodes.conf   # nodes.conf文件会自动生成
4. cluster-node-timeout  30000   # 集群结点超时时长，默认15000
5. appendonly  yes   # 开启持久化

**注：6份配置文件（nodes[1-6].conf）只需要修改端口，其他配置相同**

## 2.2 启动容器

**结点1：**

docker run --net=host --name=redis-node1 -v /u06/root/pkg/others/redis/nodes1.conf:/etc/redis/redis.conf -v /data/redis/data1:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

**结点2：**

docker run --net=host --name=redis-node2 -v /u06/root/pkg/others/redis/nodes2.conf:/etc/redis/redis.conf -v /data/redis/data2:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

**结点3：**

docker run --net=host --name=redis-node3 -v /u06/root/pkg/others/redis/nodes3.conf:/etc/redis/redis.conf -v /data/redis/data3:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

**结点4：**

docker run --net=host --name=redis-node4 -v /u06/root/pkg/others/redis/nodes4.conf:/etc/redis/redis.conf -v /data/redis/data4:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

**结点5：**

docker run --net=host --name=redis-node5 -v /u06/root/pkg/others/redis/nodes5.conf:/etc/redis/redis.conf -v /data/redis/data5:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

**结点6：**

docker run --net=host --name=redis-node6 -v /u06/root/pkg/others/redis/nodes6.conf:/etc/redis/redis.conf -v /data/redis/data6:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

## 2.3 建立集群关系

**（1）进入容器**

docker exec -it redis-node1 bash

**（2）使用redis-cli建立集群**

redis-cli --cluster create 192.168.9.214:6379 \

192.168.9.214:6380 \

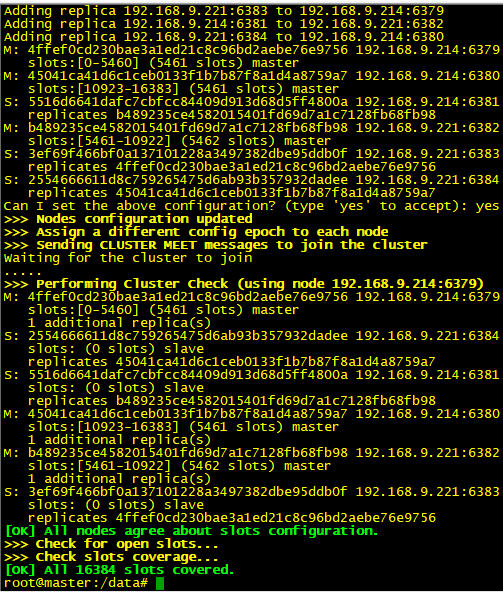
192.168.9.214:6381 \

192.168.9.221:6382 \

192.168.9.221:6383 \

192.168.9.221:6384 \

--cluster-replicas 1



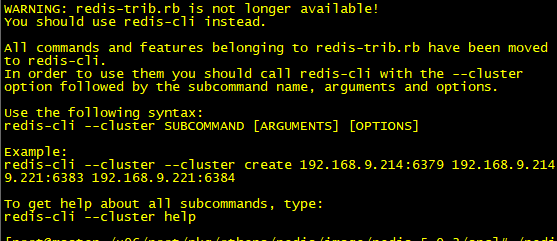
**注：此处使用的redis-cli --cluster是较新版本Redis 5.0以上的命令**

**使用低版本的redis-cli 启动高版本redis会出现错误：**

**Unrecognized option or bad number of args for: '--cluster'**



**使用redis源码中src目录下的redis-trib.rb建立高版本redis集群会出现错误：**



**因此，推荐进入容器使用容器内的redis-cli建立集群**

1. **Sentinel哨兵模式**

****

**3.1 Master结点**

**（1）redis.conf（master.conf）配置**

1. **port  6379**
2. **appendonly  yes**
3. **appendfilename  appendonly.aof**
4. **requirepass  1234 # 密码，可以不设置**

**（2）启动**

创建目录/data/redis/data1

docker run -p 6379:6379 --name=redis-master -v /u06/root/pkg/others/redis/master.conf:/etc/redis/redis.conf -v /data/redis/data1:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

## 3.2 Slave结点

**（1）redis.conf（slave.conf）配置**

1. **port  6379**
2. **appendonly  yes**
3. **appendfilename  appendonly.aof**
4. **slaveof  192.168.0.1 6389   # master结点ip 端口**
5. **masterauth  1234   # master结点密码**
6. **requirepass  1234   # Slave结点密码**

**（2）启动**

创建目录/data/redis/data2

docker run -p 6380:6379 --name=redis-slave -v /u06/root/pkg/others/redis/slave.conf:/etc/redis/redis.conf -v /data/redis/data2:/data -d redis redis-server /etc/redis/redis.conf --appendonly yes

## 3.3 sentinel哨兵

**（1）sentinel.conf配置（三个哨兵都一样的配置）**

1. **sentinel monitor mymaster 192.168.0.1 6379 1**
2. **sentinel auth-pass mymaster 1234  # master结点密码，与**
3. **# redis.conf对应**

sentinel monitor mymaster 192.168.9.214 6379 1：哨兵监控的主服务器名称为mymaster，ip为192.168.9.214，端口为6379，将这个主服务器标记为失效至少需要1个哨兵进程的同意

**（2）启动**

创建目录/data/redis/sentinel1

/data/redis/sentinel2

/data/redis/sentinel3

docker run -p 26379:26379 --name=redis-sentinel1 -v /u06/root/pkg/others/redis/sentinel1.conf:/etc/redis/sentinel.conf -v /data/redis/sentinel1:/data -d redis redis-sentinel /etc/redis/sentinel.conf

docker run -p 26380:26379 --name=redis-sentinel2 -v /u06/root/pkg/others/redis/sentinel2.conf:/etc/redis/sentinel.conf -v /data/redis/sentinel2:/data -d redis redis-sentinel /etc/redis/sentinel.conf

docker run -p 26379:26379 --name=redis-sentinel3 -v /u06/root/pkg/others/redis/sentinel3.conf:/etc/redis/sentinel.conf -v /data/redis/sentinel3:/data -d redis redis-sentinel /etc/redis/sentinel.conf

## 3.4 测试

**（1）进入sentinel容器**

docker exec -it redis-sentinel1 bash

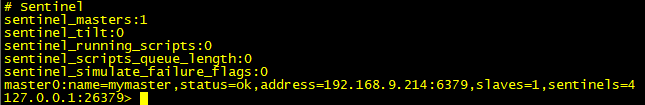
**（2）连接redis-sentinel**

redis-cli -p 26379

**（3）查看信息**

info

显示：



# 4. 问题和解决方法

## 4.1 Cluster集群模式问题

使用docker镜像运行redis时，会存在两个问题：

1. **使用容器ip和端口建立进群**

redis-cli --cluster create 10.10.0.1:6379 \

10.10.0.2:6379 \

10.10.0.3:6379 \

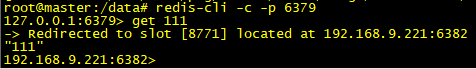
10.10.0.4:6379 \

10.10.0.5:6379 \

10.10.0.6:6379 \

--cluster-replicas 1

Cluster集群的特点是每个master结点存放部分数据，获取数据时，会先Redirected到存放该数据的master结点，然后获取Value，而这个Redirected是依据建立集群时的ip和端口的



如下图所示，若远程使用192.168.9.214:6379连接Redis，获取Key:1时，正常，获取Key:2时，会Redirected 到10.10.0.2:6379，但是宿主机以外的机器是无法连接10.10.0.2容器ip。



**解决方法：**

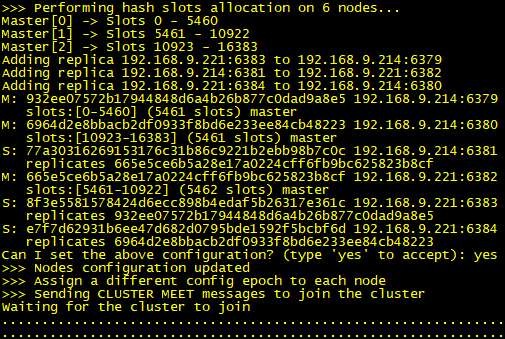
**建立集群的时候使用的ip和端口必须是宿主机的ip和端口**

**（2）使用宿主机ip和端口**

使用宿主机ip和端口解决了上面的问题1，但是也导致问题2出现

使用swarm集群部署redis集群：

即使用docker service create 创建redis实例时，必须使用overlay类型网络，而overlay网络是不能与宿主机相互连通。建立集群时，会出现一直停留在waiting for the cluster to join



**原因：**

结点间不能通过集群总线端口相连，集群总线端口号为redis启动端口+10000，如：6380对应的16380

但是，即使启动命令与上方docker run做了总线端口映射，也还是不能解决此问题。

# 5. 扩展

## 5.1 redis-cli --cluster常用命令

**（1）检查集群结点**

redis-cli --cluster check <ip:port>

1. **查看集群key、slot、slave分布信息**

redis-cli --cluster info <ip:port>

1. **添加集群节点**

redis-cli --cluster add-node <ip1:port1> <ip2:port2>

将ip1:port1添加到ip2:port2所在集群

1. **在线迁移槽**

redis-cli --cluster reshard <ip:port>

槽（slot）：Redis集群通过分片的方式来保存数据中的键值对：**集群中的数据库被分为16384个槽（slot），**数据库中的每个健都属于这16384个槽的其中一个，集群中的每个节点可以处理的0个或最多16384个槽。

1. **删除集群节点**

redis-cli --cluster del-node <ip:port>

必须是没有槽的节点，所以必须先移除槽

1. **平衡各节点槽数量**

redis-cli --cluster rebalance --cluster-threshold <ip:port>

1. **将集群外部redis实例中的数据导入到集群中去**

redis-cli --cluster import <ip1:port1> --cluster-from <ip2:port2> --cluster-copy