# Redis高性能高可用集群

## 1、 前言

本章节我们使用：Sentinel+Twemproxy+Keepalive+Redis搭建高可用集群

我们通过之前的学习发现了一个问题，哨兵模式虽然可以自动选举，但是选举的过程中会出现服务短暂不可用的问题，随意我们就在想，能不能使用通过多个集群的方式来实现负载均衡和故障转移呢？答案是可以的！

各元素职责：

Redis：缓存服务器

Sentinel:主要作用于redis的主从复制集群的master故障后从新选举新的master

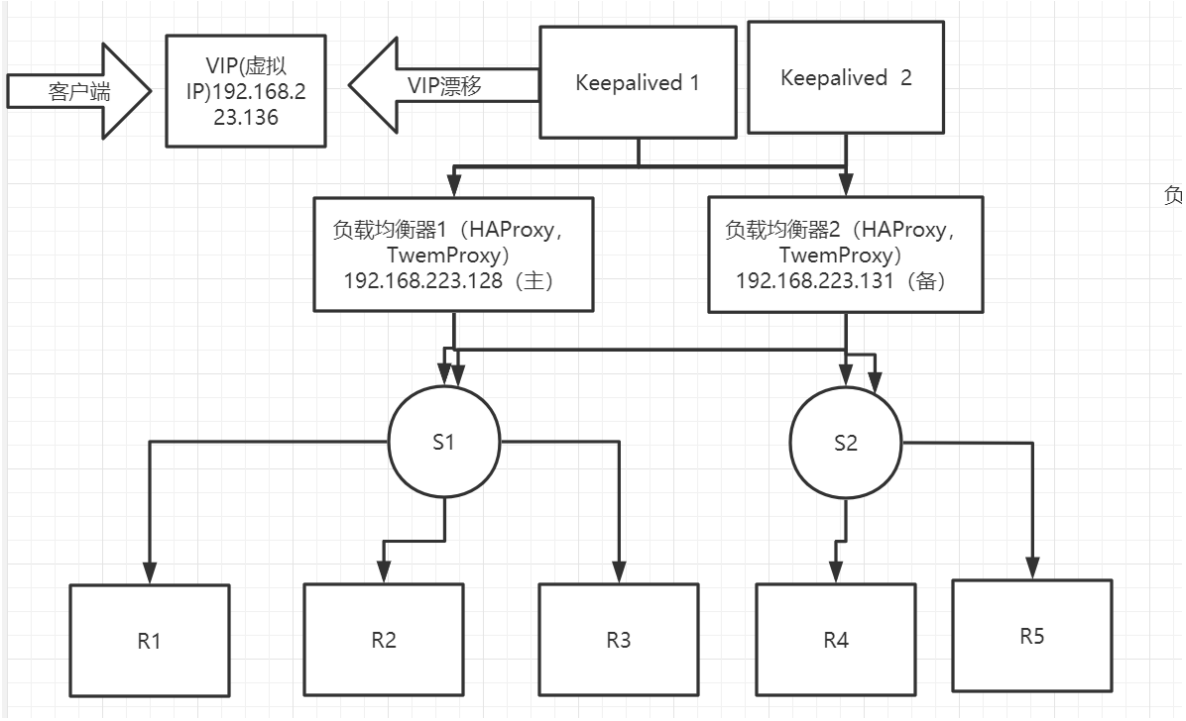
Twemproxy：redis的负载均衡代理服务器，主要对redis的多主从复制集群进行负载均衡

Keepalive：主要作用是对twemproxy进行容灾，实现twemproxy的高可用

矩阵图如下

| 名称 | 192.168.223.128 | 192.168.223.129 | 192.168.223.130 | 192.168.223.131 | 192.168.223.132 | 192.168.223.133 |
|---|---|---|---|---|---|---|
| keepalived1 | 1 | | | | | |
| keepalived2 | | | | 1 | | |
| twemproxy1 | 1 | | | | | |
| twemproxy2 | | | | 1 | | |
| 哨兵进程1 | 1 | | | | | |
| 哨兵进程2 | | | | 1 | | |
| master1 | 1 | | | | | |
| slave1-1 | | 1 | | | | |
| slave1-2 | | | 1 | | | |
| master2 | | | | 1 | | |
| slave2-1 | | | | | 1 | |
| slave2-2 | | | | | | 1 |

架构图：

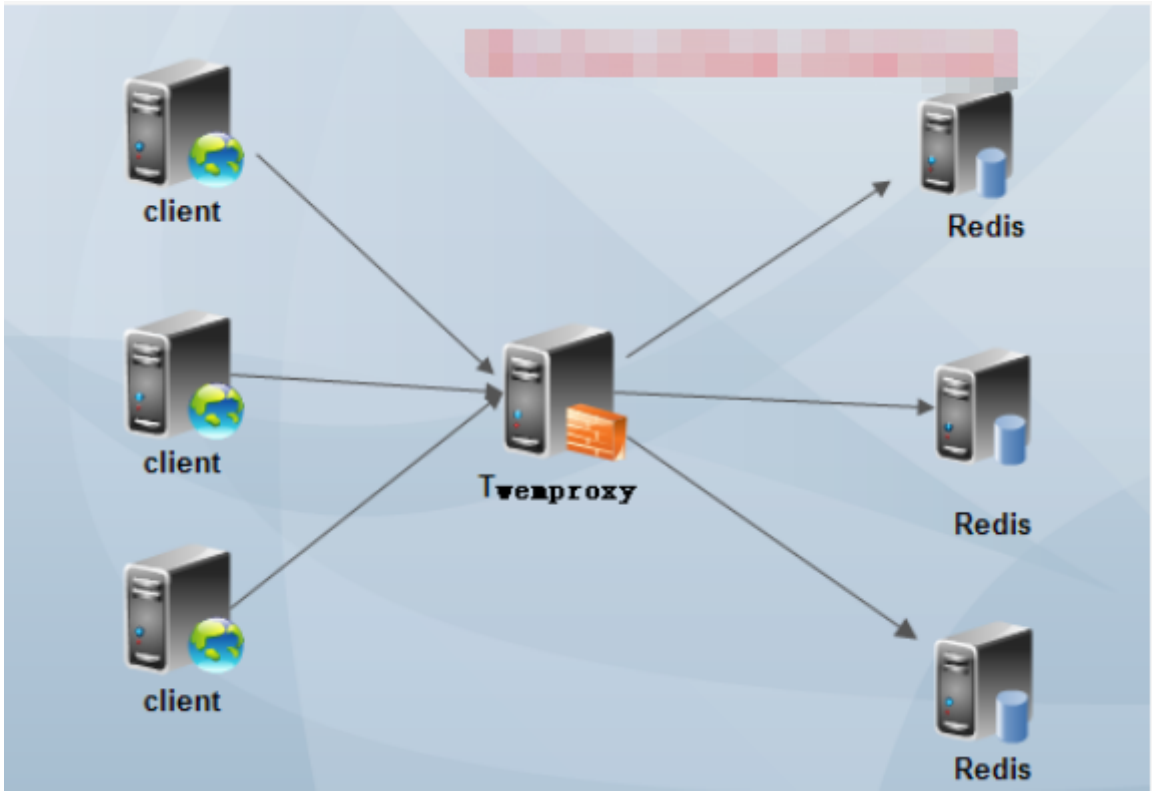



# 2、初识TwemProxy

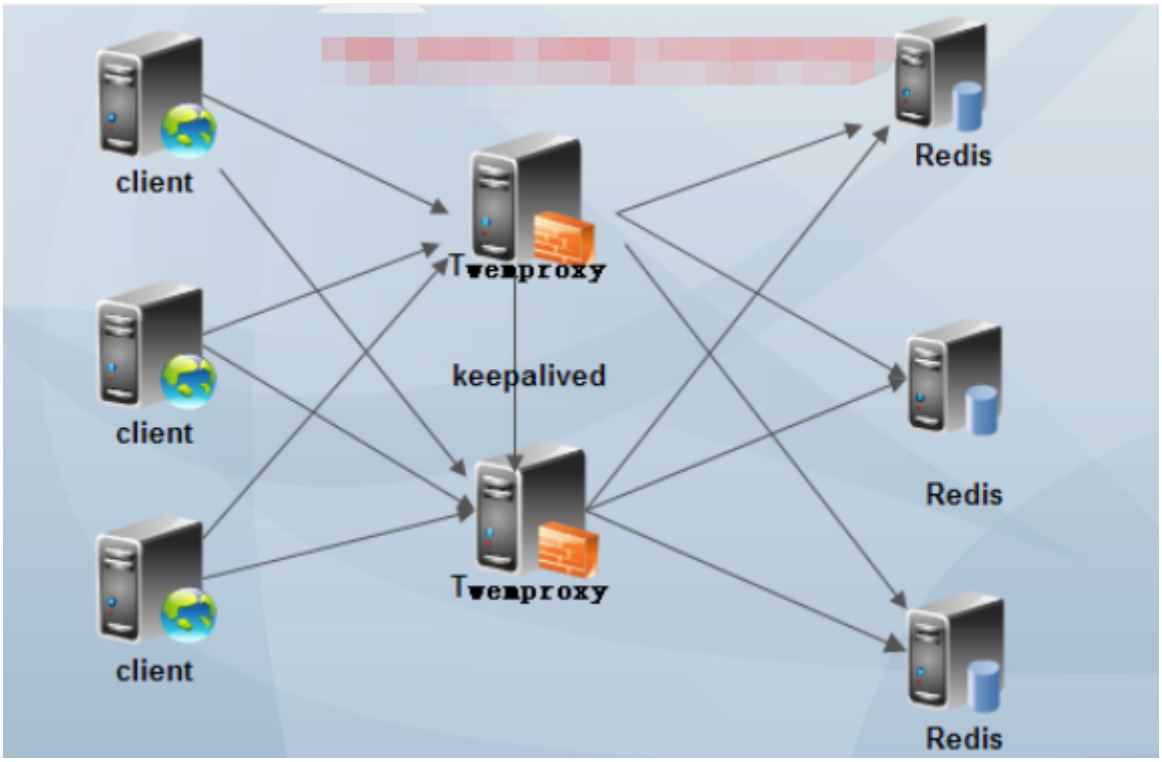
## 2.1 什么是TwemProxy

Twemproxy是一种代理分片机制，由Twitter开源。**Twemproxy作为代理，可接受来自多个程序的访问，按照路由规则，转发给后台的各个Redis服务器，再原路返回。**该方案很好的解决了单个Redis实例承载能力的问题。当然，Twemproxy本身也是单点，需要用Keepalived做高可用方案。通过Twemproxy可以使用多台服务器来水平扩张redis服务，可以有效的避免单点故障问题。虽然使用Twemproxy需要更多的硬件资源和在redis性能有一定的损失（twitter测试约20%），但是能够提高整个系统的HA也是相当划算的。不熟悉twemproxy的同学，如果玩过nginx反向代理或者mysql proxy，那么你肯定也懂twemproxy了。其实twemproxy不光实现了redis协议，还实现了memcached协议，什么意思？换句话说，twemproxy不光可以代理redis，还可以代理memcached

最简单的TwemProxy架构：




复杂的TwemProxy架构：




上面的架构通常只有一台Twemproxy在工作，另外一台处于备机，当一台挂掉以后，vip自动漂移，备机接替工作

## 2.2 安装TwemProxy

安装Twemproxy,安装之前先安装或者升级 autoconf版本（具体版本自己看着办）；

```
查询当前版本
rpm -qf /usr/bin/autoconf
```

```
卸载当前版本
rpm -e --nodeps autoconf-2.64

下载新版本
wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.65.tar.gz

解压安装
tar zxvf autoconf-2.65.tar.gz
cd autoconf-2.65
./configure --prefix=/usr
make && make install

查看是否安装成功
/usr/bin/autoconf -V
```

```
#安装twemproxy
#1.下载安装包

git clone https://github.com/twitter/twemproxy.git

cd twemproxy/

CFLAGS="-ggdb3 -O0" autoreconf -fvi && ./configure --prefix=/usr/local/twemproxy
--enable-debug=log

#PS:如果最后一步报错如下，说明要先安装automake
#Can't exec "aclocal": No such file or directory at
/usr/share/autoconf/Autom4te/FileUtils.pm line 326.
# autoreconf: failed to run aclocal: No such file or directory

yum install automake

#如果报错如下：
#configure.ac:16: error: Autoconf version 2.65 or higher is required
#请按照要求安装对应的autoconf版本

#如果报错如下：
configure.ac:36: error: possibly undefined macro: AC_PROG_LIBTOOL
      If this token and others are legitimate, please use m4_pattern_allow.
      See the Autoconf documentation.

 #安装升级libtool
 yum install libtool -y

#然后继续执行：
CFLAGS="-ggdb3 -O0" autoreconf -fvi && ./configure --prefix=/usr/local/twemproxy
--enable-debug=log
#安装bin目录，如果你高兴，也可以直接使用src目录下的
make & make test & make install
```

## 2.3 配置TwemProxy

```
#编辑nutcracker.yml，设置哨兵进程信息
vim /usr/local/twemproxy/sbin/nutcracker.yml
```

```
alpha:
  #twemproxy对外监听端口
  listen: 0.0.0.0:22121
  #hash算法，有兴趣的自行百度研究
  hash: fnv1a_64
  #存在ketama一致性hash、modula直接取hash值和random3跟hash无关，随机选择一个服务器，有兴趣的可以自行百度研究
  distribution: ketama
  #控制twemproxy是否应该根据server的连接状态重建群集。这个连接状态是由server_failure_limit 阀值来控制
  auto_eject_hosts: true
  #识别到服务器的通讯协议是redis还是memcached
  redis: true
  #单位是毫秒，控制服务器连接的时间间隔，在auto_eject_host被设置为true的时候产生作用
  server_retry_timeout: 2000
  #控制连接服务器的次数，在auto_eject_host被设置为true的时候产生作用
  server_failure_limit: 1
  #可以使用redis单机也可以使用哨兵主节点，我们先使用redis单机进行数据分片和负载均衡；最后的两个1标识权重
  servers:
   - 192.168.223.128:6379:1
   - 192.168.223.131:6379:1

#启动：
cd /usr/local/twemproxy/sbin/
#调试启动
./nutcracker -c nutcracker.yml
#以守护进程启动
./nutcracker -d -c nutcracker.yml
```

## 2.4 测试TwemProxy

1）、先启动 192.168.223.128:6379；192.168.223.131:6379 两台redis服务

2）、启动TwemProxy,命令如上所述

3）、使用redis客户端连接TwemProxy代理：

```
#客户端登录，-h为twemproxy主机IP，-p为twemproxy主机port
[root@ydt1 redis-4.0.6]# ./bin/redis-cli -h 192.168.223.128 -p 22121
127.0.0.1:22121> get a
(nil)
127.0.0.1:22121> set name laohu
OK
127.0.0.1:22121> get name
"laohu"
127.0.0.1:22121> set age 18
OK
127.0.0.1:22121> set length 20
OK
127.0.0.1:22121> set sex boy
OK
127.0.0.1:22121> get sex
"boy"
```

大家分别登录启动的两台redis服务，可以看到数据均匀分布！

# 3、初识Keepalived

## 3.1 什么Keepalived

Keepalived 是一种高性能的服务器高可用或热备解决方案， Keepalived 可以用来防止服务器单点故障的发生，通过配合 Nginx 可以实现 web 前端服务的高可用，当然还可以用于其他场合。

Keepalived 以 VRRP 协议为实现基础，用 VRRP 协议来实现高可用性(HA)。 VRRP(Virtual RouterRedundancy Protocol)协议是用于实现路由器冗余的协议， VRRP 协议将两台或多台路由器设备虚拟成一个设备，对外提供虚拟路由器 IP(一个或多个)

而在路由器组内部，如果实际拥有这个对外 IP 的路由器如果工作正常的话就是 MASTER，或者是通过算法选举产生， MASTER 实现针对虚拟路由器 IP 的各种网络功能，如 ARP 请求， ICMP，以及数据的转发等；其他设备不拥有该虚拟 IP，状态是 BACKUP，除了接收 MASTER 的VRRP 状态通告信息外，不执行对外的网络功能。当主机失效时， BACKUP 将接管原先 MASTER 的网络功能。

## 3.2 安装keepalive

```
#下载安装基础依赖包
yum install gcc
yum -y install openssl-devel
yum -y install libnl libnl-devel
yum -y install libnfnetlink-devel
yum -y install net-tools

#方法一：
yum -y install keepalived（如果你本地安装了mysql，可能会有环境冲突）

#方法二：
cd /usr/local
wget http://www.keepalived.org/software/keepalived-2.0.19.tar.gz
#解压文件
tar -zxvf keepalived-2.0.19.tar.gz
#编译
cd keepalived-2.0.19/
#--prefix 指定安装地址
#/usr/local/keepalived/ 安装的目录，不要和解压文件一个目录，不然可能报错
./configure --prefix=/usr/local/keepalived/
#编译并安装
 make && make install

#运行前配置（加载到系统服务）
cp /usr/local/keepalived-2.0.19/keepalived/etc/init.d/keepalived /etc/init.d/
mkdir /etc/keepalived
cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/
cp /usr/local/keepalived-2.0.19/keepalived/etc/sysconfig/keepalived
/etc/sysconfig/
cp /usr/local/keepalived/sbin/keepalived /usr/sbin/

#启动keepalive命令（后面配置好了再开）
service keepalived start
# 配置开机自启动（不建议开启）
systemctl enable keepalived
#查看服务启动情况
```

```
ps -aux |grep keepalived
```

## 3.3 配置keepalive启动测试

1）、准备两台TwemProxy：192.168.223.128,192.168.223.131，配置跟之前一样，分别启动

2）、配置两台Keepalived服务，我们也姑且配置在192.168.223.128,192.168.223.131上：

**192.168.223.128配置**

```
vim /etc/keepalived/keepalived.conf
#就加入如下配置即可，其他的可以不要
------------------------------------------------------------
! Configuration File for keepalived
global_defs {
   #不与其他漂移节点重名即可，如果是漂移的同一个虚拟IP，需要保持一致
   router_id redis_twemproxy
}

vrrp_instance TWEMPROXY {
    state BACKUP         #两台都使用BACKUP，具体根据priority值来判断即可
    interface eth0       #指定虚拟ip的网卡接口
    virtual_router_id 55     #路由器标识，MASTER和BACKUP必须是一致的
    priority 100             #定义优先级，数字越大，优先级越高，在同一个vrrp_instance下，
MASTER的优先级必须大于BACKUP的优先级。这样MASTER故障恢复后，就可以将VIP资源再次抢回来
    virtual_ipaddress {
        192.168.223.136
    }
}
#虚拟服务器信息,对外端口6379
virtual_server 192.168.223.136 6379 {
    delay_loop 3
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
    protocol TCP
    #真实服务器
    real_server 192.168.223.128 22121 {
        TCP_CHECK{
            connect_timeout 3
            retry 3
            delay_before_retry 3
        }
    }
}
```

**192.168.223.131配置：**

```
vim /etc/keepalived/keepalived.conf
#就加入如下配置即可，其他的可以不要
------------------------------------------------------------
! Configuration File for keepalived
global_defs {
   #不与其他漂移节点重名即可，如果是漂移的同一个虚拟IP，需要保持一致
   router_id redis_twemproxy
}
```

```
vrrp_instance TWEMPROXY {
    state BACKUP        #两台都使用BACKUP，具体根据priority值来判断即可
    interface eth0      #指定虚拟ip的网卡接口
    virtual_router_id 55     #路由器标识，MASTER和BACKUP必须是一致的
    priority 99             #定义优先级，数字越大，优先级越高，在同一个vrrp_instance下，
MASTER的优先级必须大于BACKUP的优先级。这样MASTER故障恢复后，就可以将VIP资源再次抢回来
    virtual_ipaddress {
        192.168.223.136
    }
}
#虚拟服务器信息,对外端口6379
virtual_server 192.168.223.136 6379 {
    delay_loop 3
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
    protocol TCP
    #真实服务器
    real_server 192.168.223.131 22121 {
        TCP_CHECK{
            connect_timeout 3
            retry 3
            delay_before_retry 3
        }
    }
}
```

### 3.4 启动测试

```
#1、先启动 192.168.223.128:6379；192.168.223.131:6379 两台redis服务
#2、启动两台TwemProxy 192.168.223.128:22121；192.168.223.131:22121
#3、启动两台keepalived 192.168.223.128；192.168.223.131 漂移出来的VIP为：
192.168.223.136:6379
    service keepalived start

#4、开启虚拟IP及redis端口
    iptables -t nat -A PREROUTING -p tcp -d 192.168.223.136 --dport 6379 -j
REDIRECT
```

现在你可以使用192.168.223.136:6379来进行redis访问！

## 4 、高可用多哨兵集群

### 4.1 哨兵监控脚本配置

其实我们抱着美好的愿望，希望twemproxy能够监听到我们哨兵集群的结构变化，很可惜，暂时还不支持！所以我们需要在redis目录新增如下脚本配置文件，用来通知twemproxy集群结构有变化！让其自动更新master节点并且重启负载均衡服务！----所以我们哨兵进程和twemproxy必须在同一个主机上！

vim /usr/local/redis-4.0.6/client-reconfig.sh （128,131节点都需要配置）

```
 #!/bin/sh
##sentinel 触发执行此脚本时，会默认传递几个参数过来
#<master-name>|<role>|<state>|<from-ip>|<from-port>|<to-ip>|<to-port> ，如下：
# nutcracker|observer|start|t192.168.223.128|6379|192.168.223.129|6379

monitor_name="$1"   ##monitor master-group-name
master_old_ip="$4"
master_old_port="$5"
master_new_ip="$6"
master_new_port="$7"
twemproxy_name=$(echo $monitor_name |awk -F'_' '{print $1"_"$2}')   ##注意
## 记住一个地方 master-group-name ，我这边的命名规则编辑nutcracker ，这里我就是为了获取编辑nutcracker ，因为twemproxy 的配置文件名用的是nutcracker.yml## 这里通过获取  master-group-name 来修改 twemproxy 的配置文件，这里定的一点规范而已

twemproxy_bin="/usr/local/twemproxy/src/nutcracker"
twemproxy_conf="/usr/local/twemproxy/conf/${twemproxy_name}.yml"
twemproxy_cmd="${twemproxy_bin} -d -c ${twemproxy_conf}"

## 将新的master 端口和ip 替换掉 twemproxy 配置文件中旧的master 信息
sed -i
"s/${master_old_ip}:${master_old_port}/${master_new_ip}:${master_new_port}/"
${twemproxy_conf}

## kill 掉nutcracker 进程 ，并重新启动
killall nutcracker
${twemproxy_cmd}

sleep 1
ps -ef |grep "${twemproxy_cmd}" |grep -v grep
```

可以通过redis客户端动态修改192.168.223.128:26379 sentinel 的配置，添加 client-reconfig-script 项 (类VIP漂移)，当然我们也可以使用固定的方式将配置项配置到sentinel.conf文件中：

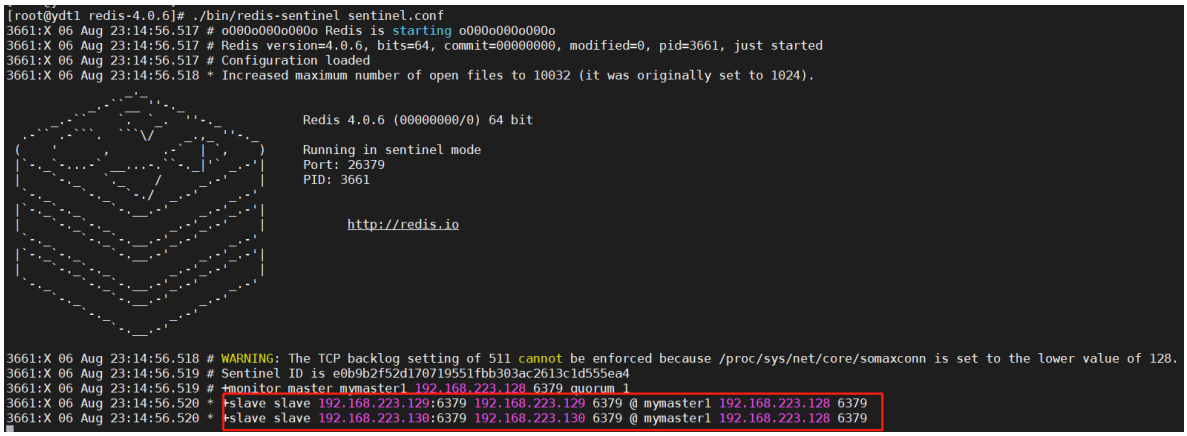
```
#节点都搞成nutcracker
sentinel client-reconfig-script nutcracker /usr/local/redis-4.0.6/client-reconfig.sh
```

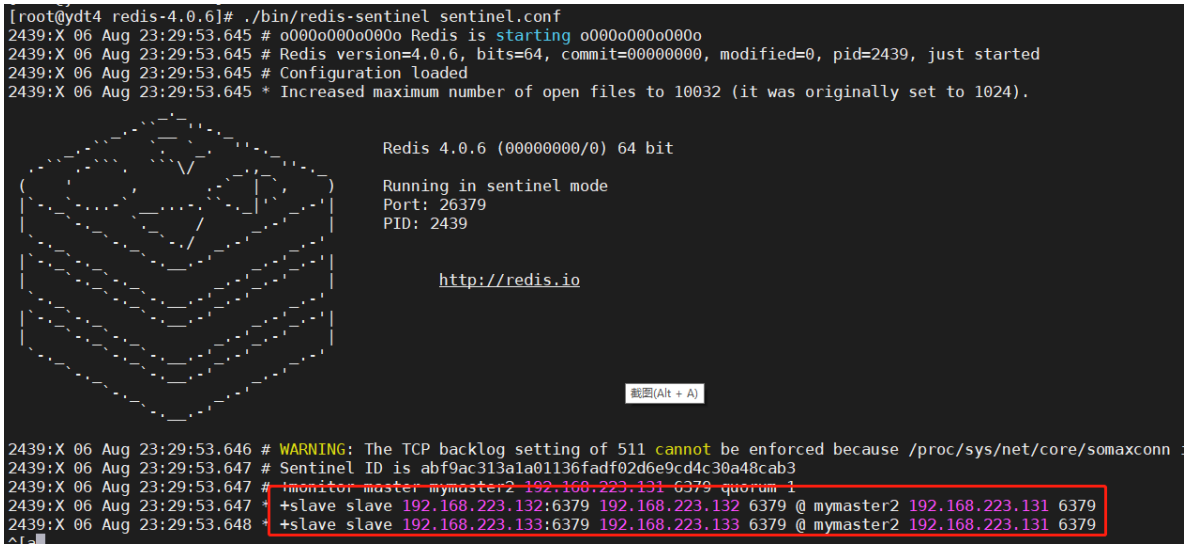
查看当前twemproxy服务下配置nutcracker.yml

```
lpha:
  listen: 0.0.0.0:22121
  hash: fnv1a_64
  distribution: ketama
  auto_eject_hosts: true
  redis: true
  server_retry_timeout: 2000
  server_failure_limit: 1
  #两个哨兵集群主节点
  servers:
   - 192.168.223.128:6379:1
```

## 4.2 哨兵集群启动

参考以上矩阵图以及前面课件中哨兵集群的搭建：




4.3 测试

再次断掉其中一个哨兵集群的主节点，你会发现哨兵进程打印了如下通知信息：


查看twemproxy配置文件，你发现文件中master节点已经变了

```
lpha:
  listen: 0.0.0.0:22121
  hash: fnv1a_64
  distribution: ketama
  auto_eject_hosts: true
  redis: true
  server_retry_timeout: 2000
  server_failure_limit: 1
  #两个哨兵集群主节点
  servers:
   - 192.168.223.129:6379:1
```

#说明已经在通知twemproxy已经重新选举了，更新了主节点ip和端口，达到了故障转移的效果！