<https://www.cnblogs.com/zhoujinyi/p/9249873.html>

下载安装包：

wget <http://mirror.bit.edu.cn/apache/zookeeper/zookeeper-3.4.14/zookeeper-3.4.14.tar.gz>

wget https://dl.google.com/go/go1.10.3.linux-amd64.tar.gz

git clone https://github.com/CodisLabs/codis.git -b release3.2

**1）**[**zookeeper安装**](https://www.jianshu.com/p/de90172ea680)**：**

zookeeper依赖JDK，需要先安装JDK：

#jdk：

yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel

1：解压：

tar -xf zookeeper-3.4.14.tar.gz

mv zookeeper-3.4.14 /usr/local/zookeeper

2：编辑配置文件：

cd /usr/local/zookeeper/conf

cp zoo\_sample.cfg zoo.cfg

配置文件(zoo.cfg)见下面

创建需要的目录：

mkdir -p /data/zookeeper/data

mkdir -p /data/zookeeper/log

设置环境变量：

vim /etc/profile

#添加

# ZooKeeper Env

export ZOOKEEPER\_HOME=/usr/local/zookeeper

export PATH=$PATH:$ZOOKEEPER\_HOME/bin

#应用环境变量

**① 单机模式**

zoo.cfg配置文件：

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/data/zookeeper/data

dataLogDir=/data/zookeeper/log

clientPort=2181

**启动：zkServer.sh start**

root@test1:~# **zkServer.sh start**

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

**查看状态：zkServer.sh status**

root@test1:~# **zkServer.sh status**

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Mode: **standalone**

**关闭：zkServer.sh stop**

root@test1:~# **zkServer.sh stop**

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Stopping zookeeper ... STOPPED

**注意：**在单机模式中, Mode 的值是 "standalone"。

**② 集群模式(3台）**

zoo.cfg配置文件：对比单机模式多了**server.ID**

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/data/zookeeper/data

dataLogDir=/data/zookeeper/log

clientPort=2181

#要是一台装三个zk，可以按照端口区分：192.168.163.131:2887:3887/192.168.163.132:2887:3887/192.168.163.133:2887:3887

server.1=192.168.163.131:2888:3888

server.2=192.168.163.132:2888:3888

server.3=192.168.163.133:2888:3888

**注意：**2888表示zookeeper监听端口，3888表示zookeeper选举通信端口；以上server.1 server.2 server.3都要配置到三台zookeeper的zoo.cfg文件。

#在第1台zookeeper（192.168.163.131）上设置id=1

echo "1" >/data/zookeeper/data/myid

#在第2台zookeeper（192.168.163.132）上设置id=2

echo "2" >/data/zookeeper/data/myid

#在第3台zookeeper（192.168.163.133）上设置id=3

echo "3" >/data/zookeeper/data/myid

三台启动：**zkServer.sh start**

root@test1:~# zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

三台查看状态：**zkServer.sh status**

root@test1:~# zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Mode: **leader**

root@test2:~# zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Mode: **follower**

root@test3:~# zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Mode: **follower**

**到此Zookeeper安装完毕。**

**2）Codis安装：**

安装go环境，codis基于go开发：

#解压：

sudo tar -C /usr/local -xzf go1.10.3.linux-amd64.tar.gz

设置环境变量：

vim /etc/profile

添加：

##go

export GOROOT=/usr/local/go

export PATH=$PATH:$GOROOT/bin

#用于安装codis的目录

export GOPATH=/opt/gowork ##mkdir -p /opt/gowork

生效环境变量：

source /etc/profile

验证：

root@test2:~# go version

go version go1.10.3 linux/amd64

创建需要的文件：

mkdir /opt/gowork

以上go的依赖环境已经安装完毕，开始安装Codis。

#创建目录

mkdir -p /opt/gowork/src/github.com/CodisLabs

#git clone迁移

mv codis /opt/gowork/src/github.com/CodisLabs/

#进入目录

cd /opt/gowork/src/github.com/CodisLabs/codis

#编译

Make

几种报错：

如果报错make[2]: \*\*\* [adlist.o] Error 127

make[2]: \*\*\* Waiting for unfinished jobs....

make[2]: \*\*\* [quicklist.o] Error 127

make[2]: \*\*\* [ae.o] Error 127

CC anet.o

/bin/sh: cc: command not found

make[2]: \*\*\* [anet.o] Error 127

make[2]: Leaving directory `/opt/gowork/src/github.com/CodisLabs/codis/extern/redis-3.2.11/src'

make[1]: \*\*\* [all] Error 2

make[1]: Leaving directory `/opt/gowork/src/github.com/CodisLabs/codis/extern/redis-3.2.11'

make: \*\*\* [codis-server] Error 2

因为Redis是C实现的，需要gcc来进行编译，所以原因是系统未安装gcc，按顺序执行：

yum install cpp

yum install binutils

yum install glibc

yum install glibc-kernheaders

yum install glibc-common

yum install glibc-devel

yum install gcc

yum install make

**jemalloc/jemalloc.h：没有那个文件或目录**

分配器allocator， 如果有MALLOC  这个 环境变量， 会有用这个环境变量的 去建立Redis。

而且libc 并不是默认的 分配器， 默认的是 jemalloc, 因为 jemalloc 被证明 有更少的 fragmentation problems 比libc。

但是如果你又没有jemalloc 而只有 libc 当然 make 出错。 所以加这么一个参数,运行如下命令：

make MALLOC=libc

错误说明

[ocetl@einvoice123 codis]$ make

make -j4 -C extern/redis-3.2.11/

make[1]: Entering directory `/home/ocetl/support/gopath/src/github.com/CodisLabs/codis/extern/redis-3.2.11'

cd src && make all

make[2]: Entering directory `/home/ocetl/support/gopath/src/github.com/CodisLabs/codis/extern/redis-3.2.11/src'

Hint: It's a good idea to run 'make test' ;)

make[2]: Leaving directory `/home/ocetl/support/gopath/src/github.com/CodisLabs/codis/extern/redis-3.2.11/src'

make[1]: Leaving directory `/home/ocetl/support/gopath/src/github.com/CodisLabs/codis/extern/redis-3.2.11'

version: line 3: git: command not found

autoconf

./autogen.sh: line 5: autoconf: command not found

Error 0 in autoconf

make[2]: \*\*\* [config] Error 1

make[1]: \*\*\* [build] Error 2

make: \*\*\* [codis-deps] Error 2

原因 缺少依赖autoconf automake libtool

解决方法

yum install autoconf automake libtool -y

make gotest

编译安装完毕之后先设置一个软连接：

ln -s /opt/gowork/src/github.com/CodisLabs/codis/ /usr/local/codis

再设置环境变量：

vim /etc/profile

#添加

# Codis

export CODIS\_HOME=/usr/local/codis

export PATH=$PATH:$CODIS\_HOME/bin

#应用环境变量

source /etc/profile

到这步了

**环境**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **机器** | **服务** | **端口** | **端口说明** | **依赖** |
| 192.168.163.131/132/133(Ubuntu 16.04) | Codis | 7021/7022 | server端口:主/从（三台） | GO |
| 11080 | proxy管理端口（三台） |
| 18080 | dashboard管理端口（一台） |
| 18090 | fe管理端口（一台） |
| 192.168.163.131/132/133(Ubuntu 16.04) | zookeeper | 2181 | zk客户端监听端口（三台） | JDK |
| 2888 | zk内部通讯端口（三台） |
| 3888 | zk选举端口（三台） |

**配置启动Codis**

① 新建目录专门存放codis的配置（包括在一台上安装（131）dashboard、proxy、fe等相关服务进程的配置）

mkdir -p /etc/codis/codis-dashboard（一台）

mkdir -p /etc/codis/codis-proxy （三台）

mkdir -p /etc/codis/codis-server （三台）

mkdir -p /etc/codis/codis-fe（一台）

mkdir –p /etc/codis/codis-ha （一台）

② 新建codis-server（redis）：三台上都安装

cp /usr/local/codis/extern/redis-3.2.11/redis.conf /etc/codis/codis-server/redis7021.conf

修改redis7021.conf：

# bind 127.0.0.1  
**protected-mode no**

port 7021

daemonize yes

pidfile /var/lib/redis\_7021/redis\_7021.pid

logfile "/var/lib/redis\_7021/redis\_7021.log"

dbfilename 7021dump.rdb

dir /var/lib/redis\_7021/

appendfilename "7021appendonly.aof"

[复制代码](javascript:void(0);)

创建配置文件里所需的目录：

mkdir /var/lib/redis\_7021

再新建一个codis-server：

[复制代码](javascript:void(0);)

cd /etc/codis/codis-server

cp redis7021.conf redis7022.conf

sed -i "s/7021/7022/g" redis7022.conf

mkdir /var/lib/redis\_7022

[复制代码](javascript:void(0);)

开启codis-server：

codis-server /etc/codis/codis-server/redis7021.conf

codis-server /etc/codis/codis-server/redis7022.conf

说明：每台机器上有2个codis-server实例，端口为：7021、7022，没有做主从。为了防止单点的问题，可以交错的设置主从，防止一台服务器挂掉，codis-server不可用。

|  |  |  |
| --- | --- | --- |
| Group | 主 | 从 |
| 1 | 192.168.163.131:7021 | 192.168.163.132:7022 |
| 2 | 192.168.163.132:7021 | 192.168.163.133:7022 |
| 3 | 192.168.163.133:7021 | 192.168.163.131:7022 |

通过codis-fe添加各个Group节点：需要先开启codis-dashboard和codis-fe，在**⑤**之后操作。

**③ 配置**[**codis-dashboard**](https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md#21-codis-dashboard)**（一台）**

cd /etc/codis/codis-dashboard/

cp /usr/local/codis/config/dashboard.toml /etc/codis/codis-dashboard/

修改配置：vim /etc/codis/codis-dashboard/dashboard.toml

[复制代码](javascript:void(0);)

##################################################

# #

# Codis-Dashboard #

# #

##################################################

# Set Coordinator, only accept "zookeeper" & "etcd" & "filesystem".

# for zookeeper/etcd, coorinator\_auth accept "user:password"

# Quick Start

#coordinator\_name = "filesystem"

#coordinator\_addr = "/tmp/codis"

**coordinator\_name** = "**zookeeper**"

**coordinator\_addr** = "**192.168.163.131:2181,192.168.163.132:2181,192.168.163.133:2181**" #zk地址，多个逗号隔开

#coordinator\_auth = ""

# Set Codis Product Name/Auth.

**product\_name** = "codis-testX" #集群名称

**product\_auth** = "" #集群密码

# Set bind address for admin(rpc), tcp only.

**admin\_addr** = "192.168.163.131:18080" restful api地址，

# Set arguments for data migration (only accept 'sync' & 'semi-async').

migration\_method = "semi-async"

migration\_parallel\_slots = 100

migration\_async\_maxbulks = 200

migration\_async\_maxbytes = "32mb"

migration\_async\_numkeys = 500

migration\_timeout = "30s"

# Set configs for redis sentinel.

sentinel\_client\_timeout = "10s"

sentinel\_quorum = 2

sentinel\_parallel\_syncs = 1

sentinel\_down\_after = "30s"

sentinel\_failover\_timeout = "5m"

sentinel\_notification\_script = ""

sentinel\_client\_reconfig\_script = ""

[复制代码](javascript:void(0);)

[参数说明](https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md#212-%E8%AF%A6%E7%BB%86%E8%AF%B4%E6%98%8E)：

| **参数** | **说明** |
| --- | --- |
| coordinator\_name | 外部存储类型，接受 zookeeper/etcd |
| coordinator\_addr | 外部存储地址 |
| product\_name | 集群名称，满足正则 \w[\w\.\-]\* |
| product\_auth | 集群密码，默认为空 |
| admin\_addr | RESTful API 端口 |

创建codis日志目录（存放codis所有log）：

mkdir /usr/local/codis/logs

[启动](https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md#211-%E5%90%AF%E5%8A%A8%E5%91%BD%E4%BB%A4)codis-dashboard服务：（一台）

[复制代码](javascript:void(0);)

codis-dashboard --ncpu=1 --config=/etc/codis/codis-dashboard/dashboard.toml --log=/usr/local/codis/logs/dashboard.log --log-level=warn &

[参数说明：](https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md" \l "212-%E8%AF%A6%E7%BB%86%E8%AF%B4%E6%98%8E" \t "_blank)

##--ncpu=N 最大使用 CPU 个数;

##-c CONF, --config=CONF 指定启动配置文件;

##-l FILE, --log=FILE 设置 log 输出文件;

##--log-level=LEVEL 设置 log 输出等级：INFO,WARN,DEBUG,ERROR；默认INFO，推荐WARN;

##对于同一个业务集群而言，可以同时部署多个codis-proxy 实例;

##不同 codis-proxy 之间由 codis-dashboard 保证状态同步。

[复制代码](javascript:void(0);)

**关闭**codis-dashboard服务：

codis-admin --dashboard=192.168.163.131:18080 --shutdown

**④：配置**[**codis-proxy**](https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md#22-codis-proxy)**（三台）**：每台配置一个Proxy，也可以一台配置多个Proxy。

注意参数：proxy\_max\_clients

cd /etc/codis/codis-proxy/

cp /usr/local/codis/config/proxy.toml /etc/codis/codis-proxy/

修改配置：vim /etc/codis/codis-proxy/proxy.toml

[复制代码](javascript:void(0);)

##################################################

# #

# Codis-Proxy #

# #

##################################################

# Set Codis Product Name/Auth.

**product\_name** = "codis-testX" #和dashboard对应

**product\_auth** = ""

# Set auth for client session

# 1. product\_auth is used for auth validation among codis-dashboard,

# codis-proxy and codis-server.

# 2. session\_auth is different from product\_auth, it requires clients

# to issue AUTH <PASSWORD> before processing any other commands.

**session\_auth** = ""

# Set bind address for admin(rpc), tcp only.

**admin\_addr** = "192.168.163.131:11080" **#同一台服务器可以根据端口创建多个Proxy**

# Set bind address for proxy, proto\_type can be "tcp", "tcp4", "tcp6", "unix" or "unixpacket".

proto\_type = "tcp4"

**proxy\_addr** = "192.168.163.131:19000" **#同一台服务器可以根据端口创建多个Proxy**

# Set jodis address & session timeout

# 1. jodis\_name is short for jodis\_coordinator\_name, only accept "zookeeper" & "etcd".

# 2. jodis\_addr is short for jodis\_coordinator\_addr

# 3. jodis\_auth is short for jodis\_coordinator\_auth, for zookeeper/etcd, "user:password" is accepted.

# 4. proxy will be registered as node:

# if jodis\_compatible = true (not suggested):

# /zk/codis/db\_{PRODUCT\_NAME}/proxy-{HASHID} (compatible with Codis2.0)

# or else

# /jodis/{PRODUCT\_NAME}/proxy-{HASHID}

**jodis\_name** = "zookeeper"

**jodis\_addr** = "192.168.163.131:2181,192.168.163.132:2181,192.168.163.133:2181"

**jodis\_auth** = ""

jodis\_timeout = "20s"

**jodis\_compatible** = false

...

...

[复制代码](javascript:void(0);)

参数说明：

| **参数** | **说明** |
| --- | --- |
| product\_name | 集群名称，参考 dashboard 参数说明 |
| product\_auth | 集群密码，默认为空 |
| admin\_addr | RESTful API 端口 |
| proto\_type | Redis 端口类型，接受 tcp/tcp4/tcp6/unix/unixpacket |
| proxy\_addr | Redis 端口地址或者路径 |
| jodis\_addr | Jodis 注册 zookeeper 地址 |
| jodis\_timeout | Jodis 注册 session timeout 时间，单位 second |
| jodis\_compatible | Jodis 注册 zookeeper 的路径 |
| backend\_ping\_period | 与 codis-server 探活周期，单位 second，0 表示禁止 |
| session\_max\_timeout | 与 client 连接最大读超时，单位 second，0 表示禁止 |
| session\_max\_bufsize | 与 client 连接读写缓冲区大小，单位 byte |
| session\_max\_pipeline | 与 client 连接最大的 pipeline 大小 |
| session\_keepalive\_period | 与 client 的 tcp keepalive 周期，仅 tcp 有效，0 表示禁止 |

[启动](https://www.cnblogs.com/zhoujinyi/p/codis-proxy%20--ncpu=1%20--config=/etc/codis/codis-proxy/proxy.toml%20--log=/usr/local/codis/logs/proxy.log%20--log-level=warn%20&)codis-proxy服务（三台）：

codis-proxy --ncpu=1 --config=/etc/codis/codis-proxy/proxy.toml --log=/usr/local/codis/logs/proxy.log --log-level=warn &

codis-proxy 启动后，处于 waiting online 状态(日志查询)，监听 proxy\_addr 地址，但是不会 accept 连接，添加到集群并完成集群状态的同步，才能改变状态为 online。添加的方法有以下两种：

* 通过 codis-fe 添加：通过 Add Proxy 按钮，将 admin\_addr 加入到集群中；⑤之后。
* 通过 codis-admin 命令行工具添加，方法如下：

codis-admin --dashboard=192.168.163.131:18080 --create-proxy -x 192.168.163.131:11080

其中 192.168.163.131:18080 以及 192.168.163.131:11080 分别为 dashboard 和 proxy 的 admin\_addr 地址；可以在后面的codis-fe里看到

添加过程中，dashboard 会完成如下一系列动作：

* 获取 proxy 信息，对集群 name 以及 auth 进行验证，并将其信息写入到外部存储中（zookeeper）；
* 同步 slots 状态；
* 标记 proxy 状态为 online，此后 proxy 开始 accept 连接并开始提供服务；

**停止codis-proxy服务：**

codis-admin --proxy=192.168.163.131:11080 --shutdown

注意：直接kill Proxy进程zk的codis3里会有残留数据，建议codis-admin方式停codis-proxy服务

**⑤：配置**[**codis-fe**](https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md#24-codis-fe%E5%8F%AF%E9%80%89%E7%BB%84%E4%BB%B6)**（一台）**

配置文件 codis.json 可以手动编辑，也可以通过 codis-admin 从外部存储中拉取：

cd /etc/codis/codis-fe/

codis-admin --dashboard-list --zookeeper=192.168.163.131:2181 | tee codis.json

[启动](https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md#24-codis-fe%E5%8F%AF%E9%80%89%E7%BB%84%E4%BB%B6)codis-fe：

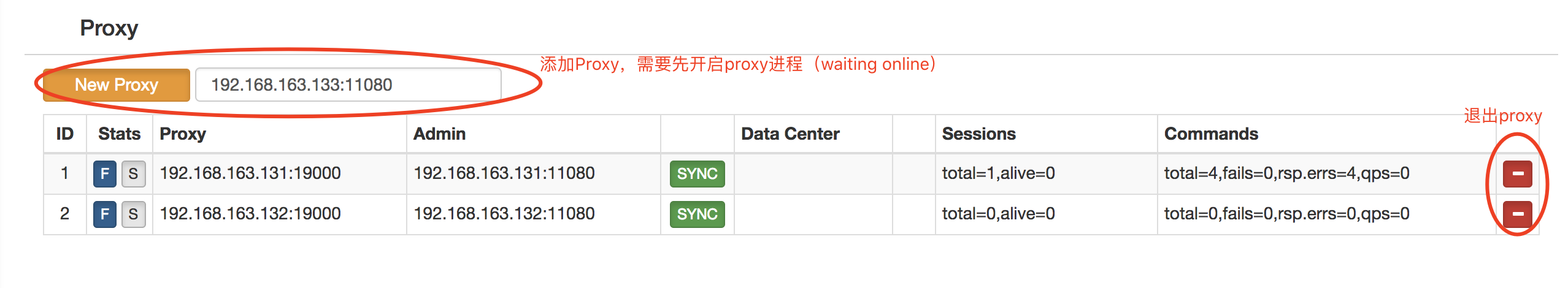
codis-fe --ncpu=1 --dashboard-list=/etc/codis/codis-fe/codis.json --listen=192.168.163.131:18090 --log=/usr/local/codis/logs/fe.log --log-level=warn --assets-dir=/usr/local/codis/bin/assets/ &

关闭codis-fe：

ps -ef|grep codis-fe|grep -v grep|awk '{print $2}'|xargs kill

**通过codis-fe进行web操作管理：**注意在fe上添加的时候需要保证这些进程存在，fe不会自动开启，只是对这些已有进程进行相关操作。

**1）添加proxy**：地址是配置文件中配置的admin\_addr信息，需要先开启codis-proxy（处于waiting online）。

  
  
其中SYNC表示：更新同步Proxy中的Slots信息，日志如下：

fill slot 0000, backend.addr = 192.168.163.133:7021, locked = false

fill slot 0001, backend.addr = 192.168.163.133:7021, locked = false

fill slot 0002, backend.addr = 192.168.163.133:7021, locked = false

fill slot 0003, backend.addr = 192.168.163.133:7021, locked = false

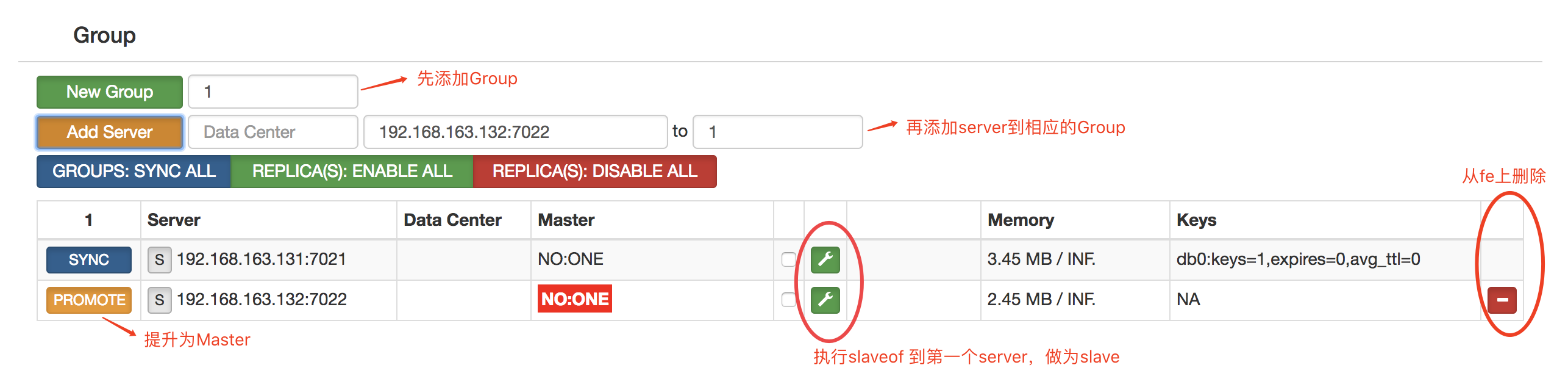
...

...

**2）添加Group**：本文说明的Group就是一主一从的环境，当然也可以一主多从。在上面介绍codis-server中，只是开启了实例，没有做主从关系。现在通过codis-fe来进行主从关系的创建，添加group：

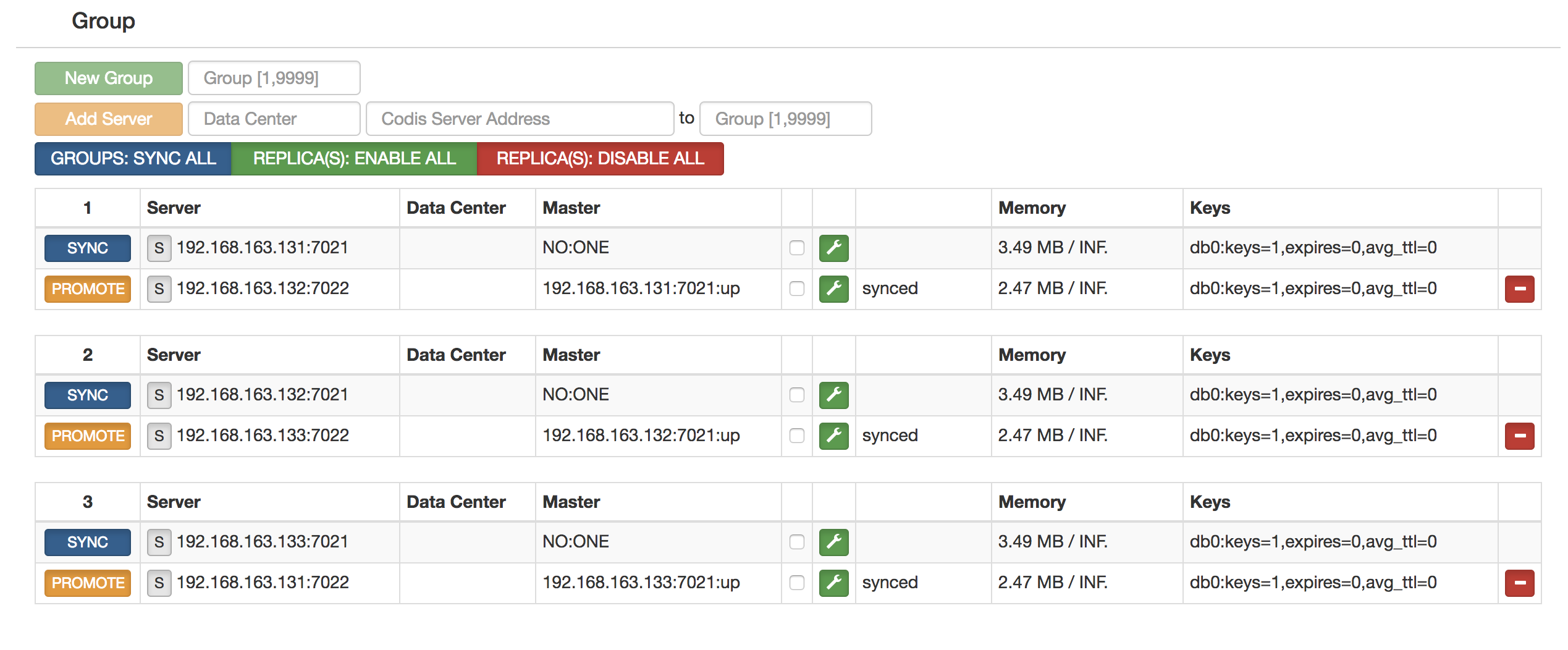
|  |  |  |
| --- | --- | --- |
| Group | 主 | 从 |
| 1 | 192.168.163.131:7021 | 192.168.163.132:7022 |

需要注意的是在fe上添加，**组内默认第一个Server是master**。



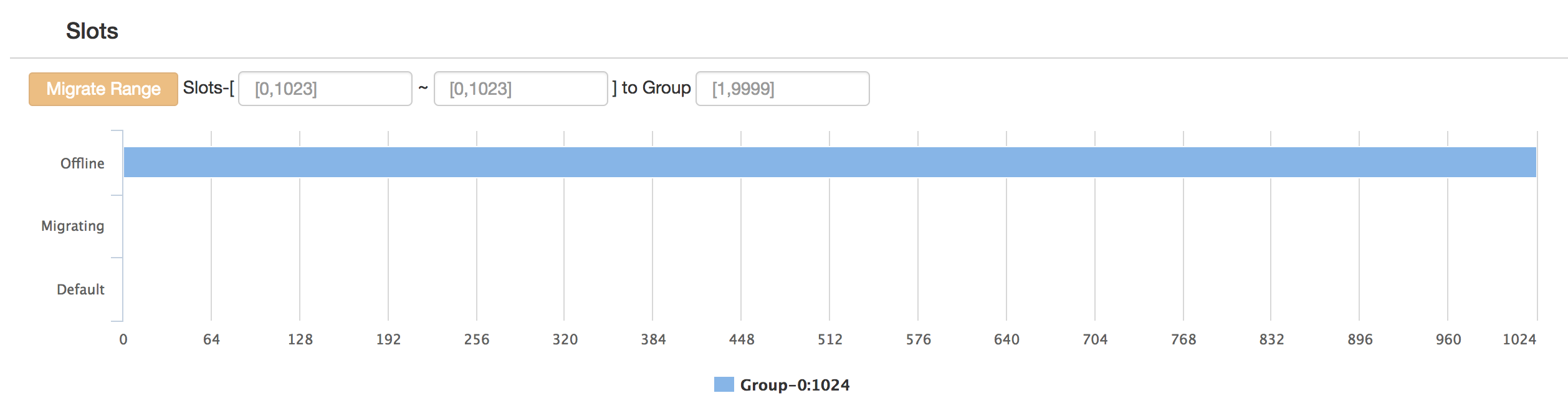
**注意：**点了PROMOTE之后，slave会被提升为master，但是老的master需要手动点才能对新主进行同步。

按照上面继续添加GROUP 2和3，最终图的结果为：

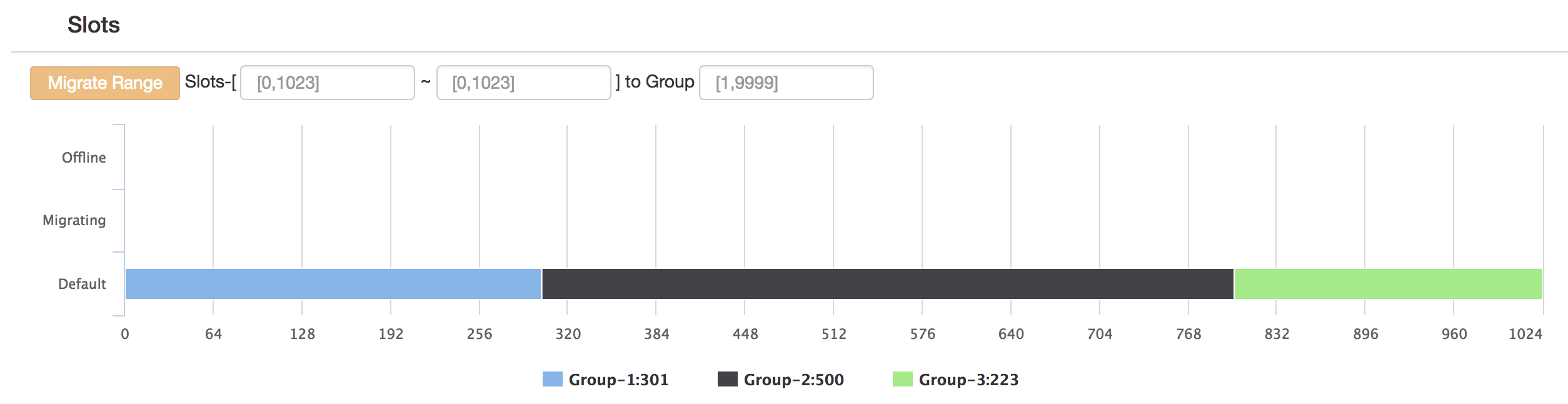


通过codis-fe已经把codis-server（redis）已经添加并自动做了主从复制。

3）初始化Slots：进行到这步的时候整个codis集群已经搭建完毕，最后只需要把Slots初始和分配下就可以使用了。



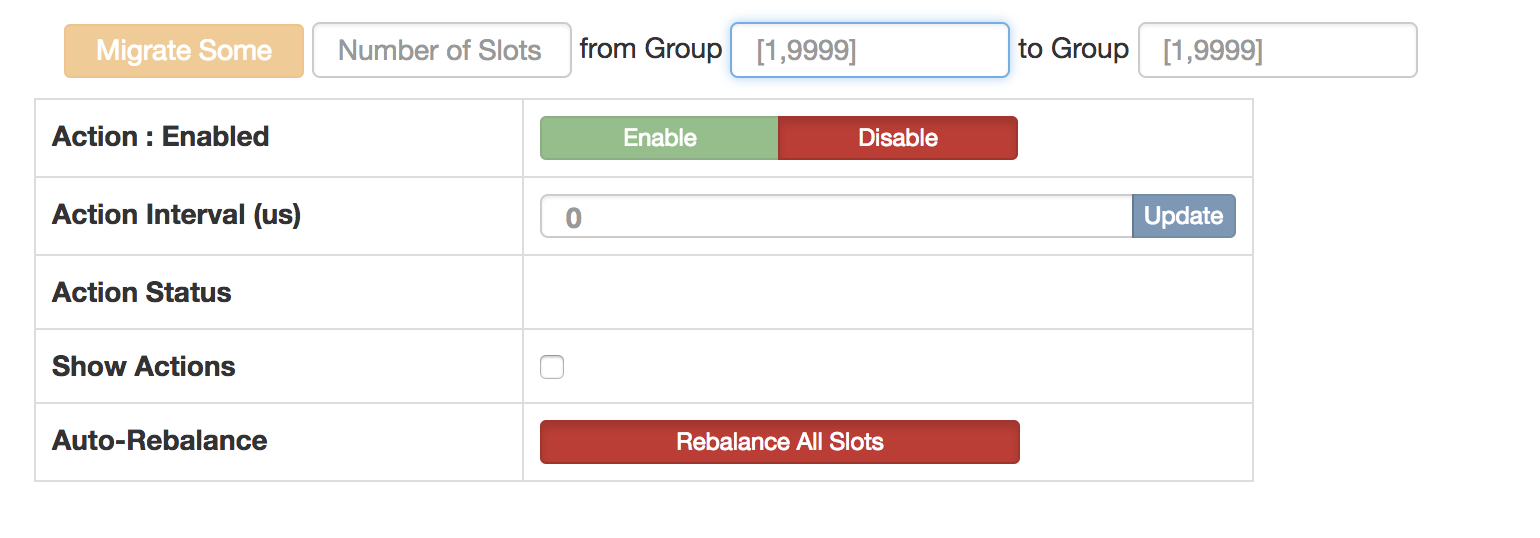
把0~300的slots分配给Group1，301~800的slots分配给Group2，801~1023的slots分配给Group3，最终结果图如下：



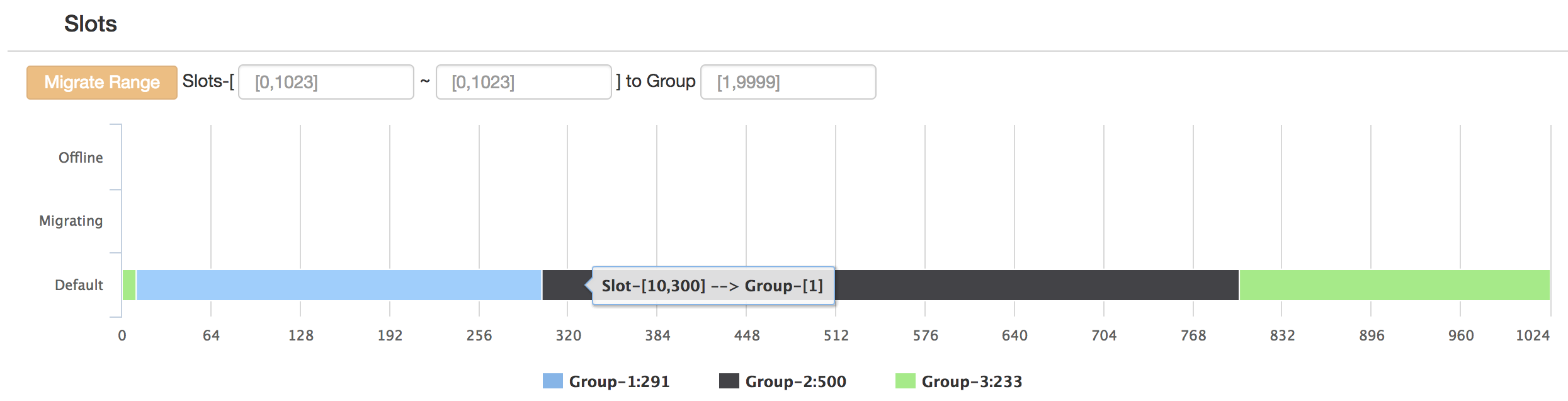
上面的图可以看到slots的分布信息，也可以通过condis-admin进行查询：

codis-admin --dashboard=192.168.163.131:18080 --slots-status

**迁移**Slots：



从Group1迁移10个slots到Group3，操作完之后的结果图：



在迁移Slots时候的zk的操作信息如下：

2018/07/03 12:54:09 zkclient.go:272: [DEBUG] zkclient update node /codis3/codis-testX/slots/slot-0008

2018/07/03 12:54:09 zkclient.go:280: [DEBUG] zkclient update OK

**到这里，codis的集群安装和初始化完毕，可以正常提供服务器了，不过这里还有个情况就是主Codis-Server挂了：**

模拟Group1的主挂掉，通过Proxy连上去看看查询到整个Group的Slots会怎么样：

root@test1:~# redis-cli -h 192.168.163.132 -p 19000 #随便连一个Proxy

192.168.163.133:19000> get age

(error) **ERR handle response**, backend conn reset

**HA如何保证？那就继续HA的部署说明。**

**⑥：配置启动codis-ha**

codis-ha --log=/usr/local/codis/logs/ha.log --log-level=warn --dashboard=192.168.163.131:18080&

**注意：**codis-ha启动之后，主挂掉虽然可以把从切换成主，但是老主再次启动会被codis-ha关闭（开不起来，除非关掉codis-ha），关闭掉codis-ha之后，还需要重新在codis-fe上添加这个老主进行同步，这个对于主从非常不友好，不推荐使用。还是使用Sentinel来替代codis-ha。

**⑦：配置Sentinel 3个节点（三台都安装），来替换codis-ha。**sentinel的说明可以参考[Redis 复制、Sentinel的搭建和原理说明](https://www.cnblogs.com/zhoujinyi/p/5570024.html)

**注意：**codis是通过sentinel来保证每个group下的Redis主从高可用，并且在codis-fe上配置的Sentinel是监控所有主机的，不需要单独配置监控的主，codis集群会自动添加。

建立所需的目录：

mkdir -p /var/lib/sentinel

修改配置文件：cp /usr/local/codis/extern/redis-3.2.11/sentinel.conf /etc/codis/codis-server/

vim /etc/codis/codis-server/sentinel.conf

[复制代码](javascript:void(0);)

port 10086

dir "/var/lib/sentinel"

logfile "/var/lib/sentinel/sentinel.log"

daemonize yes

protected-mode no

[复制代码](javascript:void(0);)

不需要启用其他的Sentinel开头的这些参数，codis-fe会自动发现处理。

**开启**Sentinel，三台保持一致即可：

codis-server /etc/codis/codis-server/sentinel.conf --sentinel

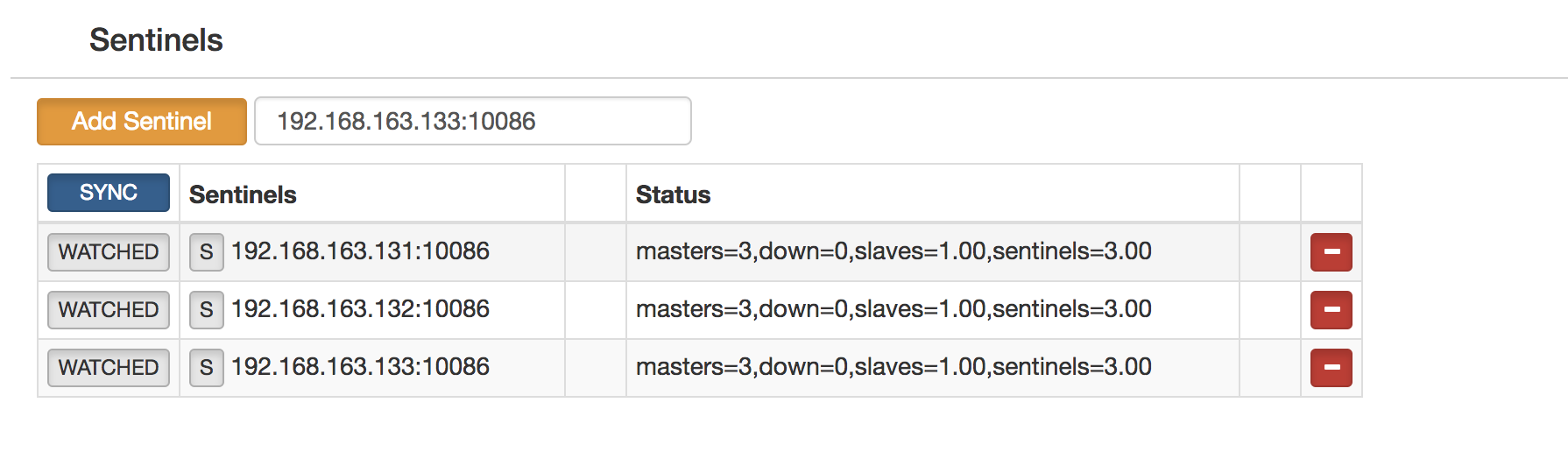
添加到集群（dashboard）：命令行模式

codis-admin --dashboard=192.168.163.131:18080 --sentinel-add --addr=192.168.163.131:10086

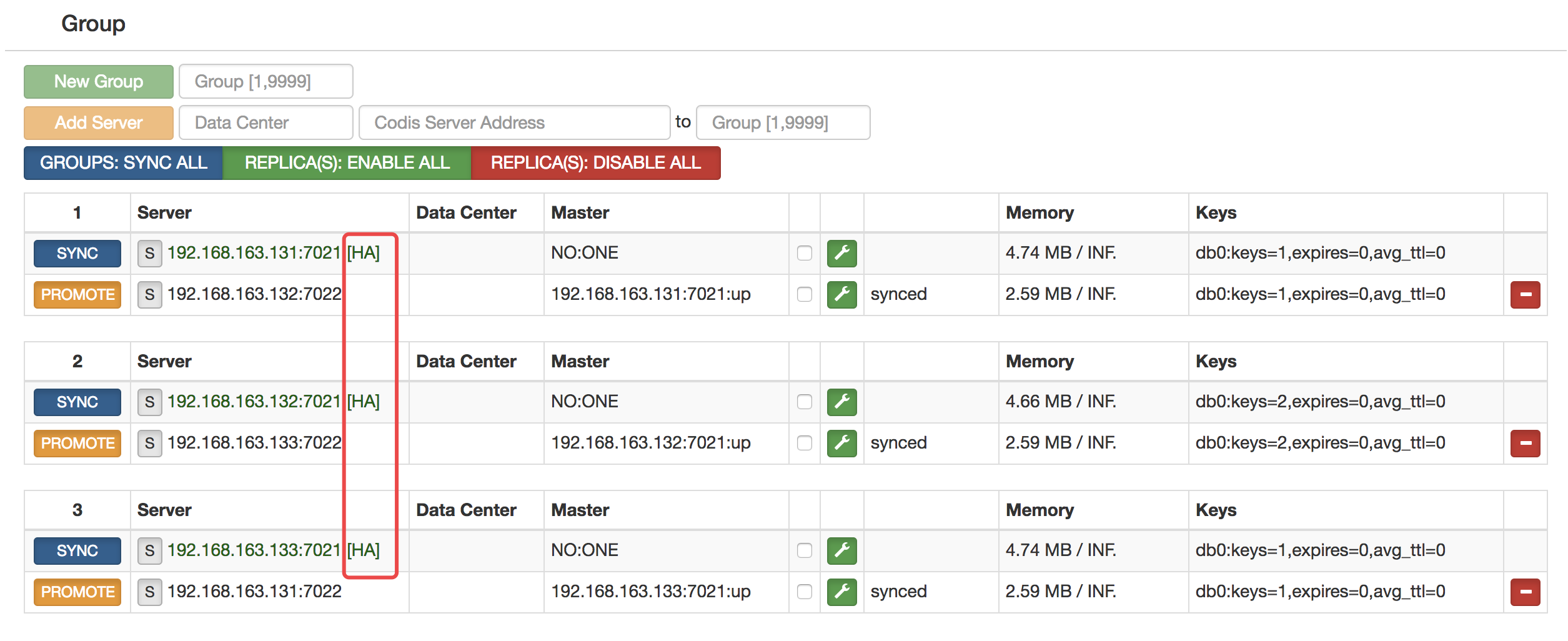
codis-admin --dashboard=192.168.163.131:18080 --sentinel-add --addr=192.168.163.132:10086

codis-admin --dashboard=192.168.163.131:18080 --sentinel-add --addr=192.168.163.133:10086

通过codis-fe添加：



该3个节点的Sentinel，监听着三组Group，即三个主从。最后一列表示从codis-fe上删除。添加完毕之后，在codis-fe上的Group中可以看到状态发生了改变：主上多了一个[HA]标识。



模拟Group1的主挂掉，通过Proxy连上去看看查询到整个Group的Slots会怎么样：

192.168.163.131:19000> get age

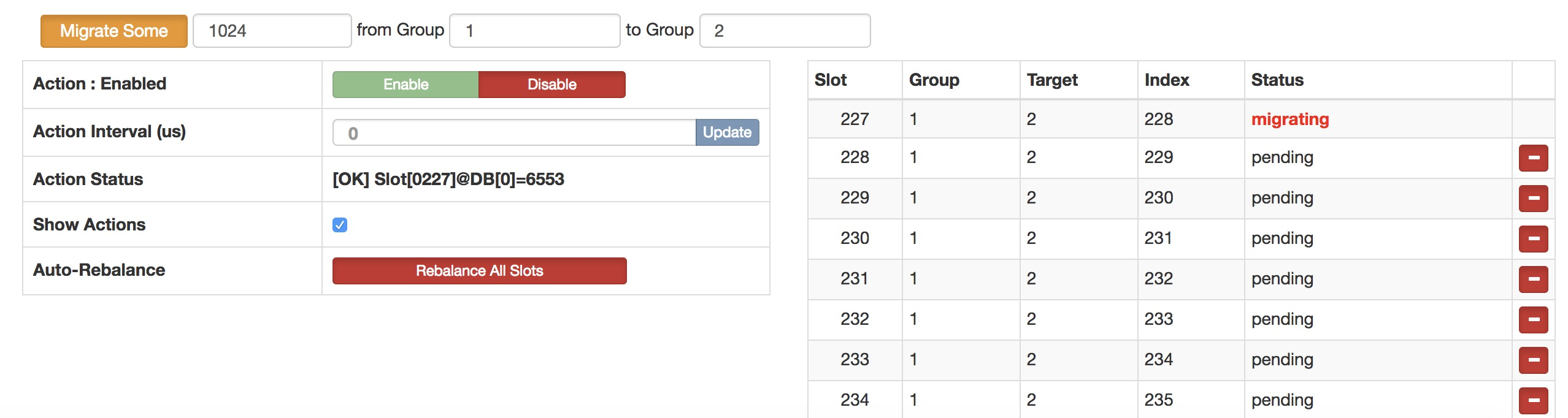
(error) ERR handle response, backend conn reset

192.168.163.131:19000> get age

"1233"

**说明：**当Group1的主被shutdown之后，Sentinel经过选举，选择新主（选举时间可配置）提供服务，选举期间该Group不可访问。老主启动后，Sentinel会自动的把老主slaveof 到新的主，形式主从集群，codis-fe不需要做主从同步的操作。最后出现“OUT OF SYNC”的标志，只需要手动SYNC下就可以了。另外说明下，如果codis-server添加了密码，则需要在codis相关组件的配置文件中的**auth**相关参数里添加密码**。**

**扩容：**如果Group1集群内存不足，需要扩容，则添加一个Group2集群，对指定数量的slots进行数据迁移，如：



迁移过程中，正在迁移的key可以访问（通过Proxy进行访问），不影响业务使用。

**Codis-admin的使用**

上面介绍的都是基于fe的web管理界面操作的，那如何通过命令行来实现呢？这时可以看下Codis-admin就派上用场了：

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

现在来逐一查看codis-admin实现的功能：

1. 查看Proxy的配置、model、状态、slots信息：具体的信息可以运行该命令

codis-admin [-v] --proxy=ADDR [--auth=AUTH] [config|model|stats|slots]

[复制代码](javascript:void(0);)

# codis-admin --proxy=192.168.163.132:11080 config

查看到proxy.toml的内容，如：proxy、zk、连接数等等

# codis-admin --proxy=192.168.163.132:11080 model

查看注册到ZK的信息：Proxy的地址，进群名称、主机名等等

# codis-admin --proxy=192.168.163.132:11080 stats

查看Proxy是否在线、sentinel信息、ops、qps、内存信息等等

# codis-admin --proxy=192.168.163.132:11080 slots

查看Slots信息：slot在哪个group，后端Redis Server地址

[复制代码](javascript:void(0);)

2. 开启关闭Proxy：这里的开启是值进入online状态。按照上面介绍的，开启codis-proxy 启动后，处于 waiting online 状态的，需要改成online状态进入zk（jodis）

codis-admin [-v] --proxy=ADDR [--auth=AUTH] --start/--shutdown

[复制代码](javascript:void(0);)

# codis-admin --proxy=192.168.163.132:11080 --start

日志中 proxy waiting online ... 会变成 proxy is working ...

并且注册到zk的jodis目录里：

API call /api/proxy/start/003acf3b450ebe66f56b8af4cc9c7d2d from 192.168.163.132:52426 []

jodis create node /jodis/codis-testX/proxy-1d24e313bee99f26174110c009714530

# codis-admin --proxy=192.168.163.132:11080 --shutdown

日志里记录关闭Proxy再把信息从zk里删除，最后再退出。

API call /api/proxy/shutdown/003acf3b450ebe66f56b8af4cc9c7d2d from 192.168.163.132:52428 []

admin shutdown

proxy shutdown

jodis remove node /jodis/codis-testX/proxy-1d24e313bee99f26174110c009714530

proxy is exiting ...

[复制代码](javascript:void(0);)

3. proxy 动态设置日志等级

codis-admin [-v] --proxy=ADDR [--auth=AUTH] --log-level=LEVEL

# codis-admin --proxy=192.168.163.132:11080 --log-level=info

日志里记录：

API call /api/proxy/loglevel/003acf3b450ebe66f56b8af4cc9c7d2d/INFO from 192.168.163.132:52432 []

set loglevel to INFO

4. 清除Proxy状态

codis-admin [-v] --proxy=ADDR [--auth=AUTH] --reset-stats

# codis-admin --proxy=192.168.163.132:11080 --reset-stats

API call /api/proxy/stats/reset/003acf3b450ebe66f56b8af4cc9c7d2d from 192.168.163.133:57436 []

清除Proxy的QPS、OPS、Fail、Errors等Commands信息

5. 强制清理Proxy内存

codis-admin [-v] --proxy=ADDR [--auth=AUTH] --forcegc

# codis-admin --proxy=192.168.163.132:11080 --forcegc

API call /api/proxy/forcegc/003acf3b450ebe66f56b8af4cc9c7d2d from 192.168.163.133:57437 []

控制Proxy内存的释放

6. 查看dashboard的配置、model、状态、slots、组和proxy信息：

codis-admin [-v] --dashboard=ADDR [config|model|stats|slots|group|proxy]

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 config

查看dashboard的配置，即dashboard.toml配置的信息

# codis-admin --dashboard=192.168.163.131:18080 model

查看model信息

# codis-admin --dashboard=192.168.163.131:18080 stats

包括slot和group的对应信息，group信息（group后端的server以及server的配置），proxy信息（sentinel信息、server信息、server主信息以及proxy CPU和内存使用情况），  
slots信息（迁移间隔、迁移状态），sentine信息（配置、状态），主server信息

# codis-admin --dashboard=192.168.163.131:18080 slots

slot和group的对应信息

# codis-admin --dashboard=192.168.163.131:18080 group

组信息，组下server的配置信息

# codis-admin --dashboard=192.168.163.131:18080 proxy

proxy信息，包括sentinel、server信息以及proxy cpu和内存使用情况，ops、qps等commands信息、连接数（session）等

[复制代码](javascript:void(0);)

7. 关闭和重新加载dashboard

codis-admin [-v] --dashboard=ADDR --shutdown/--reload

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --shutdown

关闭dashboard，清除zk里的topom

API call /api/topom/shutdown/4c0ca749efb5aad2b20b8d84b1bb6905 from 192.168.163.132:42772 []

admin exit on error

# codis-admin --dashboard=192.168.163.131:18080 --reload

修改dashboard之后，重新加载配置

API call /api/topom/reload/4c0ca749efb5aad2b20b8d84b1bb6905 from 192.168.163.132:42778 []

[复制代码](javascript:void(0);)

8. dashboard动态设置日志等级

codis-admin [-v] --dashboard=ADDR --log-level=LEVEL

# codis-admin --dashboard=192.168.163.131:18080 --log-level=info

API call /api/topom/loglevel/4c0ca749efb5aad2b20b8d84b1bb6905/INFO from 192.168.163.132:42780 []

set loglevel to INFO

9. proxy加入到dashboard，并online

codis-admin [-v] --dashboard=ADDR --create-proxy --addr=ADDR

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --create-proxy --addr=192.168.163.131:11080

proxy加入到dashboard中

[WARN] [0xc4202d17a0] API call /api/topom/proxy/create/4c0ca749efb5aad2b20b8d84b1bb6905/192.168.163.131:11080 from 192.168.163.132:42824 []

[WARN] create proxy-[1d24e313bee99f26174110c009714530]

...

[复制代码](javascript:void(0);)

10. proxy online初始化1024个槽，codis-admin [-v] --dashboard=ADDR --online-proxy --addr=ADDR

# codis-admin --dashboard=192.168.163.131:18080 --online-proxy --addr=192.168.163.133:11080

#proxy online

[WARN] [0xc4200be790] API call /api/proxy/start/f6eadfec468df5b262af66e292a27699 from 192.168.163.131:57976 []

[WARN] [0xc4200be790] API call /api/proxy/sentinels/f6eadfec468df5b262af66e292a27699 from 192.168.163.131:57976 []

[WARN] [0xc4200be790] set sentinels = []

11. 查看proxy 列表，codis-admin [-v] --dashboard=ADDR --list-proxy

# codis-admin --dashboard=192.168.163.131:18080 --list-proxy

查看该dashboard下的proxy列表和信息

12. 移除proxy，codis-admin [-v] --dashboard=ADDR --remove-proxy (--addr=ADDR|--token=TOKEN|--pid=ID) [--force]

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --remove-proxy --addr=192.168.163.133:11081

根据相关信息：ip、token、pid等移除proxy

[WARN] [0xc4200be790] API call /api/proxy/shutdown/f6eadfec468df5b262af66e292a27699 from 192.168.163.131:58144 []

[WARN] [0xc4200be790] proxy shutdown

[WARN] [0xc4200be790] admin shutdown

[WARN] jodis remove node /jodis/codis-testX/proxy-697e01c6c8b8aaf399599992e7108d35

[WARN] [0xc4200be790] proxy is exiting ...

[复制代码](javascript:void(0);)

13. reinit proxy：codis-admin [-v] --dashboard=ADDR --reinit-proxy (--addr=ADDR|--token=TOKEN|--pid=ID|--all) [--force]

# codis-admin --dashboard=192.168.163.131:18080 --reinit-proxy --addr=192.168.163.132:11080

初始化slots(1024)

[WARN] [0xc4200c68f0] API call /api/proxy/start/00d4e51e15eb811441ad228e44550b81 from 192.168.163.131:59242 []

[WARN] [0xc4200c68f0] API call /api/proxy/sentinels/00d4e51e15eb811441ad228e44550b81 from 192.168.163.131:59242 []

[WARN] [0xc4200c68f0] set sentinels = []

14 .获取proxy的信息：管理、代理端口。

# codis-admin --dashboard=192.168.163.131:18080 --proxy-status

[ ] proxy-1 [T] 4f07158ae347d67b1af825fd8e84b2f9 [A] 192.168.163.131:11080 [P] 192.168.163.131:19000

[ ] proxy-2 [T] 1d24e313bee99f26174110c009714530 [A] 192.168.163.132:11080 [P] 192.168.163.132:19000

[ ] proxy-3 [T] cfa02d4002da74e6b1f3b51f1416aa1d [A] 192.168.163.133:11080 [P] 192.168.163.133:19000

15. 查看group列表。codis-admin [-v] --dashboard=ADDR --list-group

# codis-admin --dashboard=192.168.163.131:18080 --list-group

查看group列表

16. 创建group。codis-admin [-v] --dashboard=ADDR --create-group --gid=ID

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --create-group --gid=1

创建一个group

[WARN] [0xc4202d17a0] API call /api/topom/group/create/4c0ca749efb5aad2b20b8d84b1bb6905/1 from 192.168.163.132:43114 []

[WARN] create group-[1]:

{

"id": 1,

"servers": [],

"promoting": {},

"out\_of\_sync": false

}

[复制代码](javascript:void(0);)

17. 移除group。codis-admin [-v] --dashboard=ADDR --remove-group --gid=ID

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --remove-group --gid=1

删除一个group

[WARN] [0xc4202d17a0] API call /api/topom/group/remove/4c0ca749efb5aad2b20b8d84b1bb6905/1 from 192.168.163.132:43116 []

[WARN] remove group-[1]:

{

"id": 1,

"servers": [],

"promoting": {},

"out\_of\_sync": false

}

[复制代码](javascript:void(0);)

18. group里添加server。codis-admin [-v] --dashboard=ADDR --group-add --gid=ID --addr=ADDR [--datacenter=DATACENTER]

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --group-add --gid=1 --addr=192.168.163.131:7021

group1里添加一个redis-server

[WARN] [0xc4202d17a0] API call /api/topom/group/add/4c0ca749efb5aad2b20b8d84b1bb6905/1/192.168.163.131:7021 from 192.168.163.132:43130 []

[WARN] update group-[1]:

{

"id": 1,

"servers": [

{

"server": "192.168.163.131:7021",

"datacenter": "",

"action": {},

"replica\_group": false

}

],

"promoting": {},

"out\_of\_sync": false

}

[复制代码](javascript:void(0);)

19.  group里删除server。codis-admin [-v] --dashboard=ADDR --group-del --gid=ID --addr=ADDR [--datacenter=DATACENTER]

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --group-del --gid=1 --addr=192.168.163.132:7021

删除group里的server

[WARN] [0xc4202d17a0] API call /api/topom/group/del/4c0ca749efb5aad2b20b8d84b1bb6905/1/192.168.163.132:7021 from 192.168.163.132:43140 []

[WARN] update group-[1]:

{

"id": 1,

"servers": [

{

"server": "192.168.163.131:7021",

"datacenter": "",

"action": {},

"replica\_group": false

}

],

"promoting": {},

"out\_of\_sync": false

}

[复制代码](javascript:void(0);)

20. 每个group事一个M-S结构的集群，设置每个group的复制关系：codis-admin [-v] --dashboard=ADDR --replica-groups --gid=ID --addr=ADDR (--enable|--disable)

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --replica-groups --gid=13 --addr=192.168.163.133:7021 --enable

# codis-admin --dashboard=192.168.163.131:18080 --replica-groups --gid=13 --addr=192.168.163.133:7022 --enable

同一个group的2个sever加入复制关系。

[WARN] [0xc4202d17a0] API call /api/topom/group/replica-groups/4c0ca749efb5aad2b20b8d84b1bb6905/13/192.168.163.133:7022/1 from 192.168.163.132:43440 []

[WARN] update group-[13]:

{

"id": 13,

"servers": [

{

"server": "192.168.163.133:7021",

"datacenter": "",

"action": {},

"replica\_group": true

},

{

"server": "192.168.163.133:7022",

"datacenter": "",

"action": {},

"replica\_group": true

}

],

"promoting": {},

"out\_of\_sync": false

}

[复制代码](javascript:void(0);)

21. 把组里面的第2个server做为从，形成主从关系：codis-admin [-v] --dashboard=ADDR --sync-action --create --addr=ADDR

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --sync-action --create --addr=192.168.163.133:7022

让组内的第2个server作为从

[WARN] [0xc4202d17a0] API call /api/topom/group/replica-groups/4c0ca749efb5aad2b20b8d84b1bb6905/13/192.168.163.133:7021/0 from 192.168.163.131:41954 [192.168.163.1]

[WARN] update group-[13]:

{

"id": 13,

"servers": [

{

"server": "192.168.163.133:7021",

"datacenter": "",

"action": {},

"replica\_group": false

},

{

"server": "192.168.163.133:7022",

"datacenter": "",

"action": {

"state": "synced"

},

"replica\_group": true

}

],

"promoting": {},

"out\_of\_sync": false

}

[复制代码](javascript:void(0);)

22. 查看group状态：codis-admin [-v] --dashboard=ADDR --group-status

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --group-status

查看主从复制的关系

[ ] group-11 [0] 192.168.163.131:7022 ==> NO:ONE

[ ] group-11 [1] 192.168.163.131:7021 ==> 192.168.163.131:7022:up

[ ] group-12 [0] 192.168.163.132:7021 ==> NO:ONE

[ ] group-12 [1] 192.168.163.132:7022 ==> 192.168.163.132:7021:up

[ ] group-13 [0] 192.168.163.133:7021 ==> NO:ONE

[ ] group-13 [1] 192.168.163.133:7022 ==> 192.168.163.133:7021:up

[复制代码](javascript:void(0);)

23. 提升组里的一个从作为主：codis-admin [-v] --dashboard=ADDR --promote-server --gid=ID --addr=ADDR

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --promote-server --gid=13 --addr=192.168.163.133:7022

让组13的服务器作为一个从库

# codis-admin --dashboard=192.168.163.131:18080 --group-status

[ ] group-11 [0] 192.168.163.131:7022 ==> NO:ONE

[ ] group-11 [1] 192.168.163.131:7021 ==> 192.168.163.131:7022:up

[ ] group-12 [0] 192.168.163.132:7021 ==> NO:ONE

[ ] group-12 [1] 192.168.163.132:7022 ==> 192.168.163.132:7021:up

[ ] group-13 [0] 192.168.163.133:7022 ==> NO:ONE

[X] group-13 [1] 192.168.163.133:7021 ==> NO:ONE

[WARN] group-[13] resync to prepared

提升了之后，老主需要手动执行，执行命令和**21**一样：

--sync-action --create

[复制代码](javascript:void(0);)

24. 分配slot（单个slot），codis-admin [-v] --dashboard=ADDR --slot-action --create --sid=ID --gid=ID

# codis-admin --dashboard=192.168.163.131:18080 --slot-action --create --sid=0 --gid=11

分配slot0 到group11中

[WARN] [0xc4202d17a0] API call /api/topom/slots/action/create/4c0ca749efb5aad2b20b8d84b1bb6905/0/11 from 192.168.163.132:43524 []

[WARN] update slot-[0]:

pending -> preparing ->prepared ->migrating -> finished

25. 移除slot（单个slot），codis-admin [-v] --dashboard=ADDR --slot-action --remove --sid=ID

# codis-admin --dashboard=192.168.163.131:18080 --slot-action --remove --sid=1

26. 迁移某一段范围的slots到指定的group，codis-admin [-v] --dashboard=ADDR --slot-action --create-range --beg=ID --end=ID --gid=ID

# codis-admin --dashboard=192.168.163.131:18080 --slot-action --create-range --beg=5 --end=100 --gid=11

27. 停止开启slots迁移：codis-admin [-v] --dashboard=ADDR --slot-action --disabled=VALUE

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --slot-action --disabled=0

停止slots迁移

[WARN] [0xc4202d17a0] API call /api/topom/slots/action/disabled/4c0ca749efb5aad2b20b8d84b1bb6905/0 from 192.168.163.132:43936 []

[WARN] set action disabled = false

开启slots迁移

[WARN] [0xc4202d17a0] API call /api/topom/slots/action/disabled/4c0ca749efb5aad2b20b8d84b1bb6905/1 from 192.168.163.132:43938 []

[WARN] set action disabled = true

[复制代码](javascript:void(0);)

28. 平衡slots，各group平分slots，codis-admin [-v] --dashboard=ADDR --rebalance [--confirm]

# codis-admin --dashboard=192.168.163.131:18080 --rebalance

[0517,1023] => 12

done

# codis-admin --dashboard=192.168.163.131:18080 --rebalance

nothing changes

29. 添加sentinel，保证高可用。codis-admin [-v] --dashboard=ADDR --sentinel-add --addr=ADDR

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --sentinel-add --addr=192.168.163.131:10086

添加一台sentinel

[WARN] [0xc4202d17a0] API call /api/topom/sentinels/add/4c0ca749efb5aad2b20b8d84b1bb6905/192.168.163.131:10086 from 192.168.163.132:43950 []

[WARN] update sentinel:

{

"servers": [

"192.168.163.131:10086"

],

"out\_of\_sync": true

}

[复制代码](javascript:void(0);)

30. 删除sentinel，codis-admin [-v] --dashboard=ADDR --sentinel-del --addr=ADDR [--force]

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --sentinel-del --addr=192.168.163.133:10086

[WARN] [0xc4202d17a0] API call /api/topom/sentinels/del/4c0ca749efb5aad2b20b8d84b1bb6905/192.168.163.133:10086/0 from 192.168.163.132:43956 []

[WARN] update sentinel:

{

"servers": [

"192.168.163.131:10086",

"192.168.163.132:10086",

"192.168.163.133:10086"

],

"out\_of\_sync": true

}

[WARN] update sentinel:

{

"servers": [

"192.168.163.131:10086",

"192.168.163.132:10086"

],

"out\_of\_sync": true

}

[复制代码](javascript:void(0);)

31. 重新同步，codis-admin [-v] --dashboard=ADDR --sentinel-resync

[复制代码](javascript:void(0);)

# codis-admin --dashboard=192.168.163.131:18080 --sentinel-resync

如果出现异常，可以尝试resync

[WARN] [0xc4202d17a0] API call /api/topom/sentinels/resync-all/4c0ca749efb5aad2b20b8d84b1bb6905 from 192.168.163.132:43988 []

[WARN] update sentinel:

{

"servers": [

"192.168.163.131:10086",

"192.168.163.132:10086"

],

"out\_of\_sync": true

}

[WARN] rewatch sentinels = [192.168.163.131:10086 192.168.163.132:10086]

[WARN] update sentinel:

{

"servers": [

"192.168.163.131:10086",

"192.168.163.132:10086"

],

"out\_of\_sync": false

}

[复制代码](javascript:void(0);)

32. 从ZooKeeper或则其他外部存储里里获取dashboard信息，codis-admin [-v] --dashboard-list (--zookeeper=ADDR [--zookeeper-auth=USR:PWD]|--etcd=ADDR [--etcd-auth=USR:PWD]|--filesystem=ROOT)

[复制代码](javascript:void(0);)

# codis-admin --dashboard-list --zookeeper=127.0.0.1:2181

2018/11/12 17:36:14 zkclient.go:23: [INFO] zookeeper - zkclient setup new connection to 127.0.0.1:2181

2018/11/12 17:36:14 zkclient.go:23: [INFO] zookeeper - Connected to 127.0.0.1:2181

2018/11/12 17:36:14 zkclient.go:23: [INFO] zookeeper - Authenticated: id=144120780119670793, timeout=40000

2018/11/12 17:36:14 zkclient.go:23: [INFO] zookeeper - Re-submitting `0` credentials after reconnect

[

{

"name": "codis-testX",

"dashboard": "192.168.163.131:18080"

}

]

2018/11/12 17:36:15 zkclient.go:23: [INFO] zookeeper - Recv loop terminated: err=EOF

[复制代码](javascript:void(0);)

33. 从ZooKeeper或则外部存储里获取集群的信息：slots、proxy、group等。codis-admin [-v] --config-dump --product=NAME (--zookeeper=ADDR [--zookeeper-auth=USR:PWD]|--etcd=ADDR [--etcd-auth=USR:PWD]|--filesystem=ROOT) [-1]

[复制代码](javascript:void(0);)

# codis-admin --config-dump --product=codis-testX --zookeeper=127.0.0.1:2181

2018/11/12 17:40:34 zkclient.go:23: [INFO] zookeeper - zkclient setup new connection to 127.0.0.1:2181

2018/11/12 17:40:34 zkclient.go:23: [INFO] zookeeper - Connected to 127.0.0.1:2181

2018/11/12 17:40:34 zkclient.go:23: [INFO] zookeeper - Authenticated: id=144120780119670794, timeout=40000

2018/11/12 17:40:34 zkclient.go:23: [INFO] zookeeper - Re-submitting `0` credentials after reconnect

{

slots:

proxy:

groups:

}

[复制代码](javascript:void(0);)

34. 转换配置文件，codis-admin [-v] --config-convert=FILE

# codis-admin --config-convert codis\_v2.0.json | tee codis\_v3.0.json

该命令会将 Codis 2.x 版本的配置文件中有效信息提取出来，并转成 Codis 3.x 版本的配置文件并输出

35. 应用配置文件，codis-admin [-v] --config-restore=FILE --product=NAME (--zookeeper=ADDR [--zookeeper-auth=USR:PWD]|--etcd=ADDR [--etcd-auth=USR:PWD]|--filesystem=ROOT) [--confirm]

codis-admin --config-restore=codis\_v3.0.json --product=codis\_v3.0 --zookeeper=127.0.0.1:2181 --confirm

该命令会将 Codis 3.x 版本的配置文件提交到 /codis3/codis\_v3.0 目录下。

选项 --confirm 选项表示确认提交，缺省时该命令仅仅打印配置文件作为调试。

36. 清理残留信息，dashboard或proxy异常退出导致无法重新启动的情况下，可以使用--remove-lock清除zk里lock信息后再次尝试重启。

codis-admin [-v] --remove-lock --product=NAME (--zookeeper=ADDR [--zookeeper-auth=USR:PWD]|--etcd=ADDR [--etcd-auth=USR:PWD]|--filesystem=ROOT)

# codis-admin --remove-lock --product=codis-testX --zookeeper=127.0.0.1:2181

2018/11/12 18:00:25 zkclient.go:23: [INFO] zookeeper - zkclient setup new connection to 127.0.0.1:2181

2018/11/12 18:00:25 zkclient.go:23: [INFO] zookeeper - Connected to 127.0.0.1:2181

2018/11/12 18:00:25 zkclient.go:23: [INFO] zookeeper - Authenticated: id=216173149807312897, timeout=40000

2018/11/12 18:00:25 zkclient.go:23: [INFO] zookeeper - Re-submitting `0` credentials after reconnect

2018/11/12 18:00:25 zkclient.go:23: [INFO] zookeeper - Recv loop terminated: err=EOF

**总结**

到这里，redis codis 的高可用集群已经搭建完毕，从文中的介绍说明来看，Codis实现了Redis的高可用、动态扩展、对业务层透明以及如何连接。后续进行相关的测试说明。