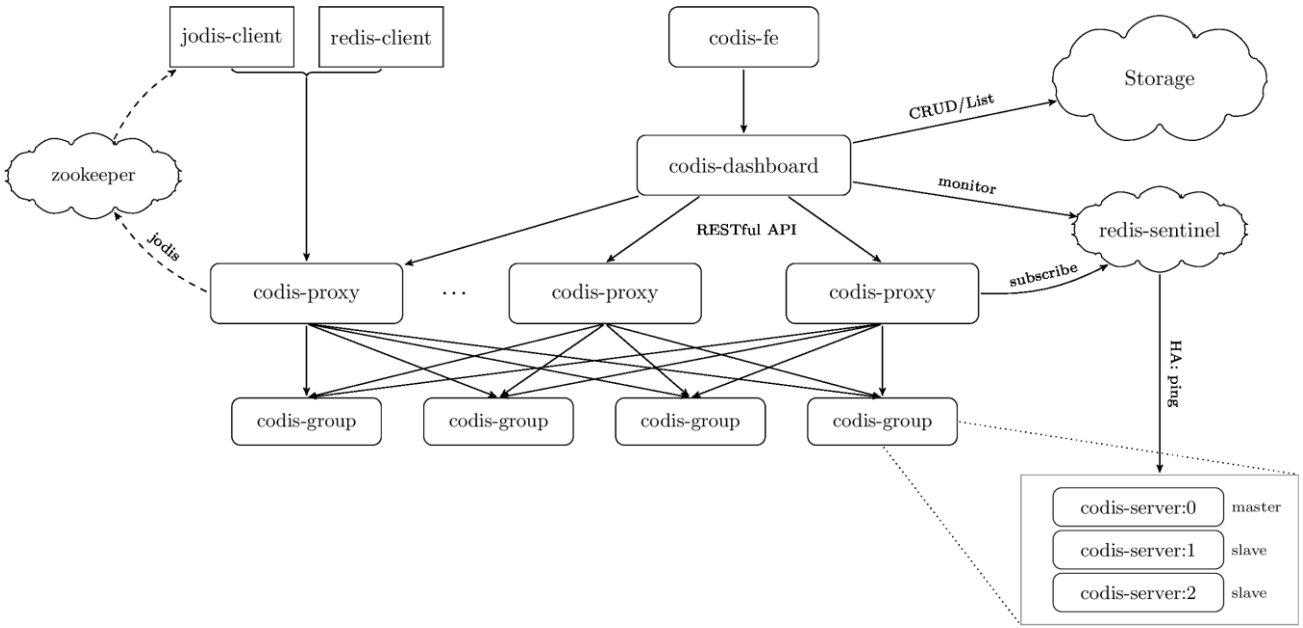
codis cluster

2017 年 4 月 23 日
16:44
codis3.2.8 集群环境部署（by：一苇）

目录

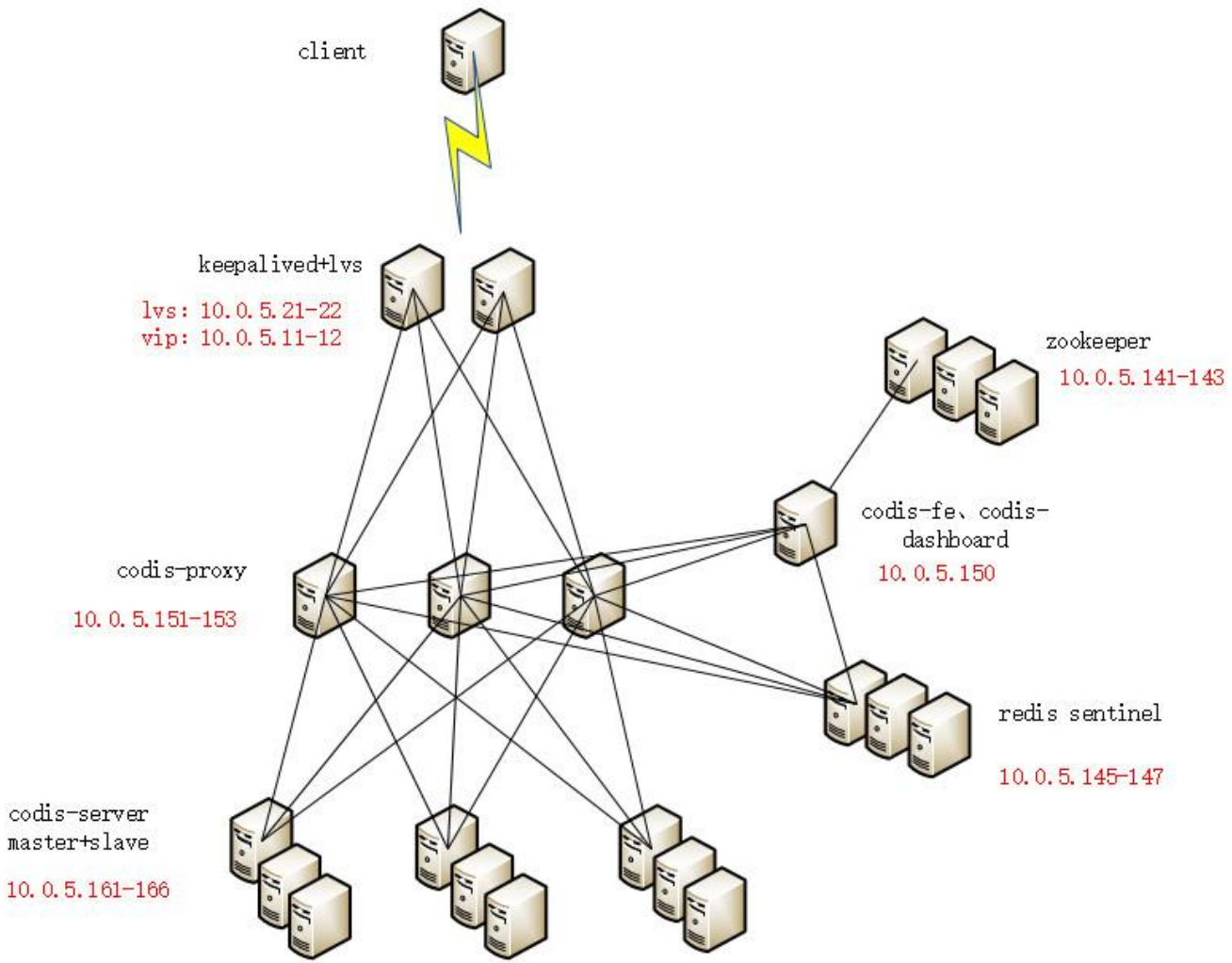
一：基础环境

1.1. codis 产品架构



1.2. 逻辑拓补图

codis集群架构



1.3.硬件规划：

lvs 环境	IP	业务端口	角色	服务描述

lvs 四层负载均衡	vip：10.0.5.11		对外业务 IP:port	
	vip：10.0.5.12		对外业务 IP:port	
	10.0.5.21	19000	keepalived+lvs	lvs 四层反代负载均衡集群；keepalived 高可用 lvs，双主模型，两个 vip 分布于两台主机，各负载一半请求；
	10.0.5.22	19000	keepalived+lvs	lvs 四层反代负载均衡集群；keepalived 高可用 lvs，双主模型，两个 vip 分布于两台主机，各负载一半请求；

codis 环境	角色	IP	业务端口	服务描述
codis 集群	zookeeper	10.0.5.141	client：2181，cluster：2888、3888	服务注册
	zookeeper	10.0.5.142	client：2181，cluster：2888、3888	服务注册
	zookeeper	10.0.5.143	client：2181，cluster：2888、3888	服务注册
	redis-sentinel	10.0.5.145	26379	codis-server 高可用监控节点
	redis-sentinel	10.0.5.146	26379	codis-server 高可用监控节点
	redis-sentinel	10.0.5.147	26379	codis-server 高可用监控节点
	codis-fe、codis-dashboard	10.0.5.150	dashboard：18080，fe：18090	codis 集群环境 web 端管理后台和面板
	codis-proxy	10.0.5.151	admin：11080，data：19000	codis-server 高可用代理节点
	codis-proxy	10.0.5.152	admin：11080，data：19000	codis-server 高可用代理节点
	codis-proxy	10.0.5.153	admin：11080，data：19000	codis-server 高可用代理节点
	codis-server	10.0.5.161	6379	codis-server 主从节点、数据分片存储节点

	codis-server	10.0.5.162	6379	codis-server 主从节点、数据分片存储节点
	codis-server	10.0.5.163	6379	codis-server 主从节点、数据分片存储节点
	codis-server	10.0.5.164	6379	codis-server 主从节点、数据分片存储节点
	codis-server	10.0.5.165	6379	codis-server 主从节点、数据分片存储节点
	codis-server	10.0.5.166	6379	codis-server 主从节点、数据分片存储节点

1.4.软件环境：

系统环境：centos6.8
软件版本：go1.8.linux-amd64.tar.gz release3.2.zip zookeeper-3.4.10.tar.gz java-1.8.0-openjdk
软件安装据目录：/app
站点数据目录：/appdata

软件下载：

```
wget http://golangtc.com/static/go/1.8/go1.8.linux-amd64.tar.gz &
wget http://mirrors.hust.edu.cn/apache/zookeeper/zookeeper-3.4.10/zookeeper-3.4.10.tar.gz &
wget https://github.com/CodisLabs/codis/archive/release3.2.zip &
```

各主机在部署前完成初始化配置，包括：ntp、yum、hostname 等

如：

```
[root ansible 14:19:15] ~
-- # crontab -l
0 */2 * * * /usr/sbin/ntpdate 10.0.1.12 &>/dev/null

[root ansible 14:24:17] ~
-- # ls /etc/yum.repos.d/
epel_aliyun.repo  local_source.repo  zabbix_aliyun.repo

[root ansible 14:24:33] ~
-- # cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=ansible
```

1.5.参考文档及交流群：

参考文档：

官方：https://github.com/CodisLabs/codis/blob/release3.2/doc/tutorial_zh.md

codis 3.1 安装搭建 作者：夏末终年

Codis3.1集群搭建文档 作者：冷月宫主

Codis 交流群：343595434

二：codis 集群部署

2.1.zookeeper 集群部署

2.1.1.部署 java

zookeeper 依赖 java 环境，需要先部署 java 环境；

```
# yum install java-1.8.0-openjdk-devel
```

配置 JAVA_HOME

```
# vim /etc/profile.d/java.sh
export JAVA_HOME=/usr

# source /etc/profile.d/java.sh

# java -version
openjdk version "1.8.0_91"
OpenJDK Runtime Environment (build 1.8.0_91-b14)
OpenJDK 64-Bit Server VM (build 25.91-b14, mixed mode)
```

2.1.2.部署 zookeeper

配置 zk141：

下载 zookeeper：

```
# mkdir /app
# cd /app
# wget http://mirrors.hust.edu.cn/apache/zookeeper/zookeeper-3.4.10/zookeeper-3.4.10.tar.gz &

# tar -xf ./zookeeper-3.4.10.tar.gz
# ln -sv ./zookeeper -3.4.10 ./zookeeper
```

配置 zookeeper：
```
# cd /app/zookeeper
# cp /app/zookeeper/conf/zoo_sample.cfg /app/zookeeper/conf/zoo.cfg

[root zk141 12:41:40] /app/zookeeper
-- # grep "^[^#]" /app/zookeeper/conf/zoo.cfg
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/app/zookeeper/data
dataLogDir=/app/zookeeper/log
clientPort=2181
server.141=10.0.5.141:2888:3888
server.142=10.0.5.142:2888:3888
server.143=10.0.5.143:2888:3888
```

创建 zookeeper 的 data、log 目录：
```
# mkdir /app/zookeeper/{data,log}
```

生成 myid 文件：
```
# echo "141" > /app/zookeeper/data/myid  //各 zookeeper 集群主机 myid 内容 ID，需和配置文件中的 server.ID=10.0.5.141:2888:3888 相同；
```

配置 PATH：
```
# cat /etc/profile.d/zookeeper.sh
export PATH=$PATH:/app/zookeeper-3.4.10/bin
```

添加开机自启动：
```
# cat /etc/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
```

```
/app/zookeeper-3.4.10/bin/zkServer.sh start
```

启动 zookeeper 服务：

```
# /app/zookeeper/bin/zkServer.sh start

# /app/zookeeper/bin/zkServer.sh -h
ZooKeeper JMX enabled by default
Using config: /app/zookeeper/bin/../conf/zoo.cfg
Usage: /app/zookeeper/bin/zkServer.sh {start|start-foreground|stop|restart|status|upgrade|print-cmd}
```

按照 zk141 的部署步骤依次配置 zk142、zk143：

```
[root zk142 20:31:12] /app
-- # grep "^[^#].*" ./zookeeper/conf/zoo.cfg
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/app/zookeeper/data
dataLogDir=/app/zookeeper/log
clientPort=2181
server.141=10.0.5.141:2888:3888
server.142=10.0.5.142:2888:3888
server.143=10.0.5.143:2888:3888


[root zk143 03:48:48] ~
-- # grep "^[^#]" /app/zookeeper/conf/zoo.cfg
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/app/zookeeper/data
dataLogDir=/app/zookeeper/log
clientPort=2181
server.141=10.0.5.141:2888:3888
server.142=10.0.5.142:2888:3888
```

server.143=10.0.5.143:2888:3888

查看 zookeeper 集群状态：

```
[root zk141 20:45:57] /app/zookeeper
-- # /app/zookeeper/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /app/zookeeper/bin/../conf/zoo.cfg
Mode: follower

[root zk142 20:44:57] /app
-- # /app/zookeeper/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /app/zookeeper/bin/../conf/zoo.cfg
Mode: leader

[root zk143 20:46:55] ~
-- # /app/zookeeper/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /app/zookeeper/bin/../conf/zoo.cfg
Mode: follower
```

客户端连接 zookeeper：

```
[root zk141 20:41:40] /app/zookeeper
-- # /app/zookeeper/bin/zkCli.sh -server 10.0.5.141:2181
Connecting to 10.0.5.141:2181
2017-04-24 20:44:18,702 [myid:] - INFO  [main:Environment@100] - Client environment:zookeeper.version=3.4.10-
39d3a4f269333c922ed3db283be479f9deacaa0f, built on 03/23/2017 10:13 GMT
2017-04-24 20:44:18,705 [myid:] - INFO  [main:Environment@100] - Client environment:host.name=<NA>
2017-04-24 20:44:18,705 [myid:] - INFO  [main:Environment@100] - Client environment:java.version=1.8.0_91
2017-04-24 20:44:18,706 [myid:] - INFO  [main:Environment@100] - Client environment:java.vendor=Oracle Corporation
2017-04-24 20:44:18,706 [myid:] - INFO  [main:Environment@100] - Client environment:java.home=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.91-
1.b14.el6.x86_64/jre
2017-04-24 20:44:18,706 [myid:] - INFO  [main:Environment@100] - Client
environment:java.class.path=/app/zookeeper/bin/../build/classes:/app/zookeeper/bin/../build/lib/*.jar:/app/zookeeper/bin/../lib/slf4j-
```

```
log4j12-1.6.1.jar:/app/zookeeper/bin/../lib/slf4j-api-1.6.1.jar:/app/zookeeper/bin/../lib/netty-
3.10.5.Final.jar:/app/zookeeper/bin/../lib/log4j-1.2.16.jar:/app/zookeeper/bin/../lib/jline-0.9.94.jar:/app/zookeeper/bin/../zookeeper-
3.4.10.jar:/app/zookeeper/bin/../src/java/lib/*.jar:/app/zookeeper/bin/../conf:
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client
environment:java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:java.io.tmpdir=/tmp
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:java.compiler=<NA>
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:os.name=Linux
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:os.arch=amd64
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:os.version=2.6.32-642.el6.x86_64
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:user.name=root
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:user.home=/root
2017-04-24 20:44:18,707 [myid:] - INFO  [main:Environment@100] - Client environment:user.dir=/app/zookeeper-3.4.10
2017-04-24 20:44:18,708 [myid:] - INFO  [main:ZooKeeper@438] - Initiating client connection, connectString=10.0.5.141:2181
sessionTimeout=30000 watcher=org.apache.zookeeper.ZooKeeperMain$MyWatcher@25f38edc
Welcome to ZooKeeper!
2017-04-24 20:44:18,724 [myid:] - INFO  [main-SendThread(10.0.5.141:2181):ClientCnxn$SendThread@1032] - Opening socket connection to
server 10.0.5.141/10.0.5.141:2181. Will not attempt to authenticate using SASL (unknown error)
JLine support is enabled
2017-04-24 20:44:18,774 [myid:] - INFO  [main-SendThread(10.0.5.141:2181):ClientCnxn$SendThread@876] - Socket connection established to
10.0.5.141/10.0.5.141:2181, initiating session
2017-04-24 20:44:18,781 [myid:] - INFO  [main-SendThread(10.0.5.141:2181):ClientCnxn$SendThread@1299] - Session establishment complete
on server 10.0.5.141/10.0.5.141:2181, sessionid = 0x8d5b9959a09c0009, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[zk: 10.0.5.141:2181(CONNECTED) 0] ls /
[jodis, codis3, zookeeper]
[zk: 10.0.5.141:2181(CONNECTED) 1] help
ZooKeeper -server host:port cmd args
        stat path [watch]
        set path data [version]
        ls path [watch]
        delquota [-n|-b] path
        ls2 path [watch]
```

```
setAcl path acl
setquota -n|-b val path
history
redo cmdno
printwatches on|off
delete path [version]
sync path
listquota path
rmr path
get path [watch]
create [-s] [-e] path data acl
addauth scheme auth
quit
getAcl path
close
connect host:port
```

2.2.codis 编译安装

2.2.1.部署 go

codis 编译依赖 go 语言环境，需要先部署 go 环境；

```
# mkdir -pv /app/gopkg
# cd /app
# wget http://golangtc.com/static/go/1.8/go1.8.linux-amd64.tar.gz &

# tar -xf /app/go1.8.linux-amd64.tar.gz
```

修改 PATH：

```
# cat /etc/profile.d/go.sh
export GOROOT=/app/go
export GOPATH=/app/gopkg
export PATH=$PATH:$GOROOT/bin

# source /etc/profile.d/go.sh
```

查看 go 版本：

```
# go version
go version go1.8 linux/amd64
```

2.2.2.编译安装 codis

安装编译环境：

```
# yum -y install gcc gcc-c++ autoconf make unzip
```

创建 codis 编译目录：

```
# mkdir /app/gopkg/src/github.com/CodisLabs
```

创建 codis 安装目录、数据及日志目录：

```
# mkdir -pv /app/codis/{data,log}
```

下载 codis-release3.2：

```
# cd /app/gopkg/src/github.com/CodisLabs
# wget https://github.com/CodisLabs/codis/archive/release3.2.zip &
# unzip /app/gopkg/src/github.com/CodisLabs/release3.2.zip
```

编译 codis-release3.2：

```
# ln -sv /app/gopkg/src/github.com/CodisLabs/codis-release3.2 /app/gopkg/src/github.com/CodisLabs/codis
# cd /app/gopkg/src/github.com/CodisLabs/codis-release3.2
# make MALLOC=libc
```

编译完成后 bin 目录生成如下文件：

```
# ll /app/gopkg/src/github.com/CodisLabs/codis-release3.2/bin
total 67640
drwxr-xr-x 4 root root     4096 Jun 19 23:05 assets
-rwxr-xr-x 1 root root 15206342 Jun 19 23:05 codis-admin
-rwxr-xr-x 1 root root 16775582 Jun 19 23:05 codis-dashboard
-rwxr-xr-x 1 root root 14938086 Jun 19 23:05 codis-fe
-rwxr-xr-x 1 root root 18968628 Jun 19 23:05 codis-proxy
-rwxr-xr-x 1 root root  2674841 Jun 19 23:04 codis-server
-rwxr-xr-x 1 root root   274044 Jun 19 23:04 redis-benchmark
-rwxr-xr-x 1 root root   405285 Jun 19 23:04 redis-cli
-rw-r--r-- 1 root root       94 Jun 19 23:04 version
```

复制编译后的 codis/bin 目录到 codis 安装目录：
cp /app/gopkg/src/github.com/CodisLabs/codis-release3.2/bin /app/codis/

修改 PATH：
cat /etc/profile.d/codis.sh
export PATH=$PATH:/app/codis/bin

source /etc/profile.d/codis.sh

后续 codis-xxx 各种角色部署均基于 codis 编译安装的结果；

codis 相关目录约定：
编译目录：/app/gopkg/src/github.com/CodisLabs/codis-release3.2/
安装目录：/app/codis
命令目录：/app/codis/bin
数据目录：/app/codis/data
日志目录：/app/codis/log

2.3.codis-dashboard 部署

完成 codis 编译安装；

生成 codis-dashboard 配置文件：
./codis-dashboard --default-config |tee /app/codis/dashboard.conf

```
##################################################
#                                                #
#                  Codis-Dashboard               #
#                                                #
##################################################

# Set Coordinator, only accept "zookeeper" & "etcd" & "filesystem".
# Quick Start
coordinator_name = "zookeeper"
coordinator_addr = "10.0.5.141:2181,10.0.5.142:2181,10.0.5.143:2181"
```

```
#coordinator_addr = "10.0.5.141:2181,10.0.5.142:2181,10.0.5.143:2181"
#coordinator_name = "zookeeper"
#coordinator_addr = "127.0.0.1:2181"

# Set Codis Product Name/Auth.
product_name = "codis-test"  //因为是搭建测试环境、此处我将 codis-demo 修改为 codis-test 用于标记；如果此处做了修改后续 codis-fe 配置文件中要对应；
product_auth = ""

# Set bind address for admin(rpc), tcp only.
admin_addr = "0.0.0.0:18080"

# 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_quorum = 2
sentinel_parallel_syncs = 1
sentinel_down_after = "30s"
sentinel_failover_timeout = "5m"
sentinel_notification_script = ""
sentinel_client_reconfig_script = ""
```

启动 codis-dashboard 服务：

```
# nohup /app/codis/bin/codis-dashboard --ncpu=4 --config=/app/codis/dashboard.conf --log=/app/codis/log/dashboard.log --log-level=WARN &
```

正常关闭 codis-dashboard 服务：

```
# /app/codis/bin/codis-admin --dashboard=10.0.5.150:18080 --shutdown

# codis-dashboard -h
```

```
Usage:
     codis-dashboard [--ncpu=N] [--config=CONF] [--log=FILE] [--log-level=LEVEL] [--host-admin=ADDR] [--pidfile=FILE]
     codis-dashboard  --default-config
     codis-dashboard  --version

Options:
     --ncpu=N                    set runtime.GOMAXPROCS to N, default is runtime.NumCPU().
     -c CONF, --config=CONF      run with the specific configuration.
     -l FILE, --log=FILE         set path/name of daliy rotated log file.
     --log-level=LEVEL           set the log-level, should be INFO,WARN,DEBUG or ERROR, default is INFO.
```

2.4.codis-fe 部署

完成 codis 编译安装；

生成 codis-fe 配置文件：

```
# /app/codis/bin/codis-admin --dashboard-list --zookeeper=10.0.5.141:2181 |tee ./codis.json
[
    {
        "name": "codis-test",    //与 codis-dashboard 中的名字相同
        "dashboard": "10.0.5.150:18080"
    }
]
```

启动 codis-fe 服务：

```
# nohup /app/codis/bin/codis-fe --ncpu=4 --log=/app/codis/log/fe.log --log-level=WARN --dashboard-list=/app/codis/codis.json --
listen=0.0.0.0:18090 &

# codis-fe -h
Usage:
     codis-fe [--ncpu=N] [--log=FILE] [--log-level=LEVEL] [--assets-dir=PATH] [--pidfile=FILE] (--dashboard-list=FILE|--
     zookeeper=ADDR|--etcd=ADDR|--filesystem=ROOT) --listen=ADDR
     codis-fe  --version

Options:
     --ncpu=N                        set runtime.GOMAXPROCS to N, default is runtime.NumCPU().
```

```
      -d FILE, --dashboard-list=FILE  set list of dashboard, can be generated by codis-admin.
      -l FILE, --log=FILE             set path/name of daliy rotated log file.
      --log-level=LEVEL               set the log-level, should be INFO,WARN,DEBUG or ERROR, default is INFO.
      --listen=ADDR                   set the listen address.
```

2.5.codis-proxy 部署

完成 codis 编译安装；

生成 codis-proxy 配置文件：

```
# /app/codis/bin/codis-proxy --default-config |tee /app/codis/proxy.conf

##################################################
#                                                #
#                  Codis-Proxy                   #
#                                                #
##################################################

# Set Codis Product Name/Auth.
product_name = "codis-test"
product_auth = ""

# Set bind address for admin(rpc), tcp only.
admin_addr = "0.0.0.0:11080"

# Set bind address for proxy, proto_type can be "tcp", "tcp4", "tcp6", "unix" or "unixpacket".
proto_type = "tcp4"
proxy_addr = "0.0.0.0:19000"

# 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. 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 = "10.0.5.141:2181,10.0.5.142:2181,10.0.5.143:2181,"
jodis_timeout = "20s"
jodis_compatible = false

# Set datacenter of proxy.
proxy_datacenter = ""

# Set max number of alive sessions.
proxy_max_clients = 1000

# Set max offheap memory size. (0 to disable)
proxy_max_offheap_size = "1024mb"

# Set heap placeholder to reduce GC frequency.
proxy_heap_placeholder = "256mb"

# Proxy will ping backend redis (and clear 'MASTERDOWN' state) in a predefined interval. (0 to disable)
backend_ping_period = "5s"

# Set backend recv buffer size & timeout.
backend_recv_bufsize = "128kb"
backend_recv_timeout = "30s"

# Set backend send buffer & timeout.
backend_send_bufsize = "128kb"
backend_send_timeout = "30s"

# Set backend pipeline buffer size.
backend_max_pipeline = 1024

# Set backend never read replica groups, default is false
backend_primary_only = false

# Set backend parallel connections per server
```

```
backend_primary_parallel = 1
backend_replica_parallel = 1

# Set backend tcp keepalive period. (0 to disable)
backend_keepalive_period = "75s"

# Set number of databases of backend.
backend_number_databases = 16

# If there is no request from client for a long time, the connection will be closed. (0 to disable)
# Set session recv buffer size & timeout.
session_recv_bufsize = "128kb"
session_recv_timeout = "30m"

# Set session send buffer size & timeout.
session_send_bufsize = "64kb"
session_send_timeout = "30s"

# Make sure this is higher than the max number of requests for each pipeline request, or your client may be blocked.
# Set session pipeline buffer size.
session_max_pipeline = 10000

# Set session tcp keepalive period. (0 to disable)
session_keepalive_period = "75s"

# Set session to be sensitive to failures. Default is false, instead of closing socket, proxy will send an error response to client.
session_break_on_failure = false

# Set metrics server (such as http://localhost:28000), proxy will report json formatted metrics to specified server in a predefined
period.
metrics_report_server = ""
metrics_report_period = "1s"

# Set influxdb server (such as http://localhost:8086), proxy will report metrics to influxdb.
metrics_report_influxdb_server = ""
metrics_report_influxdb_period = "1s"
```

```
metrics_report_influxdb_username = ""
metrics_report_influxdb_password = ""
metrics_report_influxdb_database = ""

# Set statsd server (such as localhost:8125), proxy will report metrics to statsd.
metrics_report_statsd_server = ""
metrics_report_statsd_period = "1s"
metrics_report_statsd_prefix = ""
```

启动 codis-proxy 服务：

```
#  nohup /app/codis/bin/codis-proxy --ncpu=4 --config=/app/codis/proxy.conf --log=/app/codis/log/proxy.log --log-level=WARN &
```

正常关闭 codis-proxy 服务：

```
# /app/codis/bin/codis-admin --proxy=10.0.5.150:11080 --auth="xxx"（有密码就加，没有就不加） --shutdown
```

```
# codis-proxy -h
Usage:
     codis-proxy [--ncpu=N [--max-ncpu=MAX]] [--config=CONF] [--log=FILE] [--log-level=LEVEL] [--host-admin=ADDR] [--host-proxy=ADDR]
     [--dashboard=ADDR|--zookeeper=ADDR|--etcd=ADDR|--filesystem=ROOT|--fillslots=FILE] [--ulimit=NLIMIT] [--pidfile=FILE]
     codis-proxy  --default-config
     codis-proxy  --version

Options:
     --ncpu=N                    set runtime.GOMAXPROCS to N, default is runtime.NumCPU().
     -c CONF, --config=CONF      run with the specific configuration.
     -l FILE, --log=FILE         set path/name of daliy rotated log file.
     --log-level=LEVEL           set the log-level, should be INFO,WARN,DEBUG or ERROR, default is INFO.
     --ulimit=NLIMIT             run 'ulimit -n' to check the maximum number of open file descriptors.
```

2.6. codis-server 部署

完成 codis 编译安装；

此处配置单机单实例的 codis-server；单机多实例的 codis-server 请参考另一篇文档（codis-AIO）；

生成 codis-server（即 redis-server）配置文件；
cp /app/gopkg/src/github.com/CodisLabs/codis-release3.2/extern/redis-3.2.8/redis.conf /app/codis/redis.conf

修改 codis-server 配置文件：
grep "^[^#]" /app/codis/redis.conf
bind 0.0.0.0
protected-mode no
port 6379
tcp-backlog 511
timeout 60
tcp-keepalive 300
daemonize yes
supervised no
pidfile /tmp/redis_6379.pid
loglevel notice
logfile "/app/codis/log/codis_server.log"
databases 16
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename rdb_6379.rdb
dir /app/codis/data
slave-serve-stale-data yes
slave-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
repl-backlog-size 4mb
slave-priority 100
maxmemory 2gb
appendonly yes
appendfilename "aof_6379.aof"
appendfsync everysec

```
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
lua-time-limit 5000
slowlog-log-slower-than 10000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
hash-max-ziplist-entries 512
hash-max-ziplist-value 64
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
activerehashing yes
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit slave 256mb 64mb 60
client-output-buffer-limit pubsub 32mb 8mb 60
hz 10
aof-rewrite-incremental-fsync yes
```

注意：此时不用在配置文件里指明 slaveof 节点，后续会在 codis-fe 管理面板里指定；

启动 codis-server 服务：
/app/codis/bin/codis-server /app/codis/redis.conf

各 codis-server 节点按此配置即可；

2.7.redis-sentinel 部署

codis/bin 默认没有集成 redis-sentinel 工具；只需从已编译好 codis 的节点将 redis-sentinel 命令和 sentinel.conf 复制到 sentinel 节点，然后启动服务即可；

创建 redis-sentinel 目录，复制 redis-sentinel 命令和配置文件：

```
# mkdir -pv /app/codis/bin
# scp /app/gopkg/src/github.com/CodisLabs/codis-release3.2/extern/redis-3.2.8/src/redis-sentinel 10.0.5.145:/app/codis/bin/
# scp /app/gopkg/src/github.com/CodisLabs/codis-release3.2/extern/redis-3.2.8/sentinel.conf 10.0.5.145:/app/codis/
```

修改 redis-sentinel 配置文件：

```
# grep "^[^#]" ./sentinel.conf
bind 0.0.0.0
protected-mode no
port 26379
dir "/tmp"
```

注意：此处不需要指定 monitor 节点信息，后续会在 codis-fe 管理面板里指定；

启动 redis-sentinel 服务：

```
/app/codis/bin/redis-sentinel /app/codis/sentinel.conf &
```

各 redis-sentinel 节点按此配置即可；

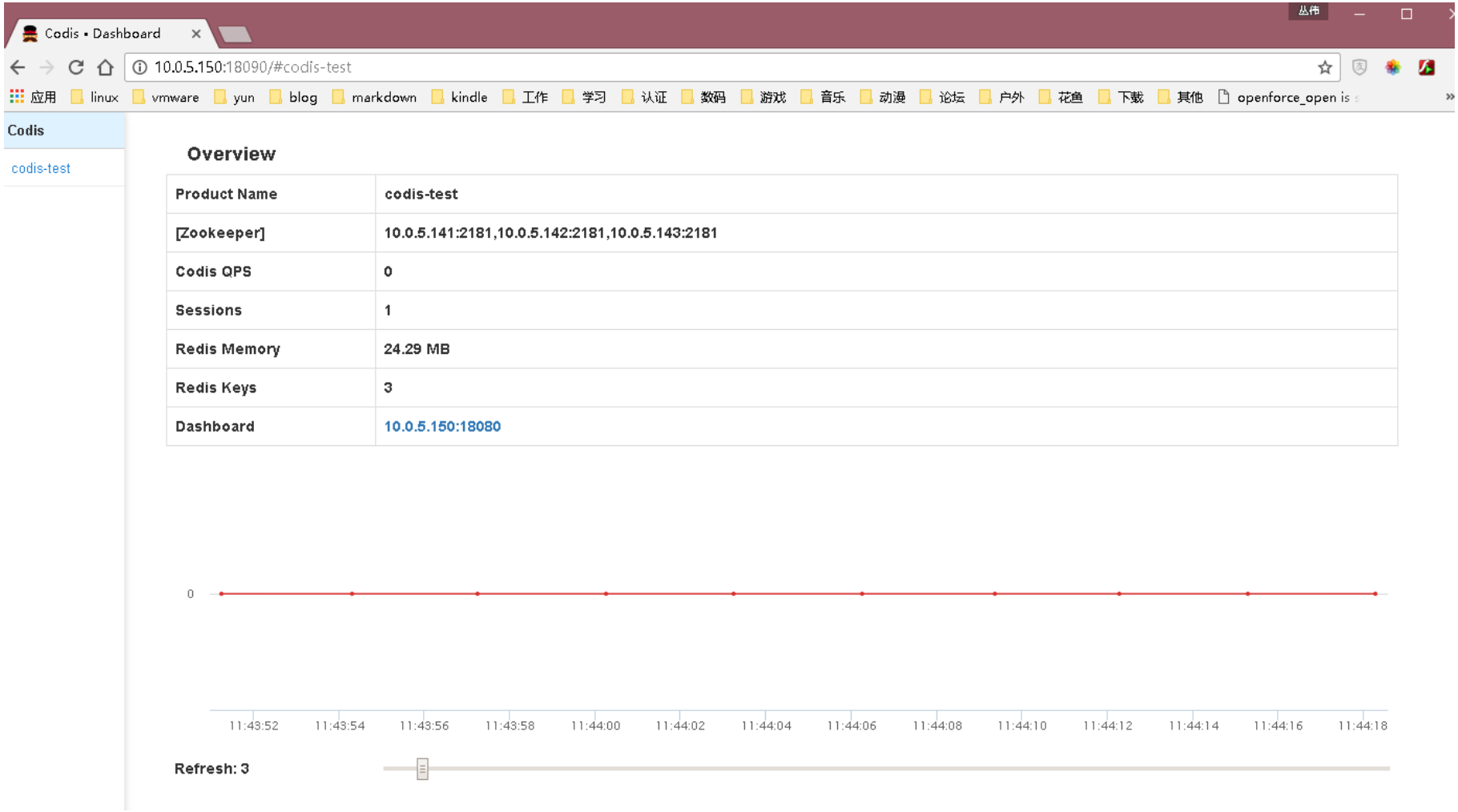
后续在 codis-fe 管理面板里指定 monitor 节点信息后，sentinel.conf 配置文件会自动添加相关信息：

```
# grep "^[^#]" ./sentinel.conf
bind 0.0.0.0
protected-mode no
port 26379
dir "/tmp"
sentinel myid 33d139f55709ba64a8e9e3691c631b735a222be9
sentinel monitor codis-test-3 10.0.5.165 6379 2
sentinel failover-timeout codis-test-3 300000
sentinel config-epoch codis-test-3 0
sentinel leader-epoch codis-test-3 0
sentinel known-slave codis-test-3 10.0.5.166 6379
sentinel known-sentinel codis-test-3 10.0.5.146 26379 4f4e175c3e72544d73951f54c90494b772ab88ac
sentinel known-sentinel codis-test-3 10.0.5.147 26379 088b0075a3353d00e29097430b46db70fe71c98e
sentinel monitor codis-test-1 10.0.5.161 6379 2
sentinel failover-timeout codis-test-1 300000
sentinel config-epoch codis-test-1 0
```

```
sentinel leader-epoch codis-test-1 0
sentinel known-slave codis-test-1 10.0.5.162 6379
sentinel known-sentinel codis-test-1 10.0.5.146 26379 4f4e175c3e72544d73951f54c90494b772ab88ac
sentinel known-sentinel codis-test-1 10.0.5.147 26379 088b0075a3353d00e29097430b46db70fe71c98e
sentinel monitor codis-test-2 10.0.5.163 6379 2
sentinel failover-timeout codis-test-2 300000
sentinel config-epoch codis-test-2 0
sentinel leader-epoch codis-test-2 0
sentinel known-slave codis-test-2 10.0.5.164 6379
sentinel known-sentinel codis-test-2 10.0.5.146 26379 4f4e175c3e72544d73951f54c90494b772ab88ac
sentinel known-sentinel codis-test-2 10.0.5.147 26379 088b0075a3353d00e29097430b46db70fe71c98e
sentinel current-epoch 0
```

三：通过 codis-fe 管理面板管理 codis 集群

使用浏览器打开 codis-fe 的管理面板： http://10.0.5.150:18090



添加 codis-proxy：

Proxy

New Proxy | 10.0.5.151:11080

ID	Stats	Proxy	Admin		Data Center		Sessions	Commands	
1	F S	10.0.5.151:19000	10.0.5.151:11080	SYNC			total=2259,alive=0	total=3,fails=0,errors=0,qps=0	−
2	F S	10.0.5.152:19000	10.0.5.152:11080	SYNC			total=2244,alive=0	total=0,fails=0,errors=0,qps=0	−
3	F S	10.0.5.153:19000	10.0.5.153:11080	SYNC			total=2247,alive=1	total=11,fails=0,errors=0,qps=0	−

添加 codis-server：

Group

New Group | Group [1,9999]

Add Server | Data Center | Codis Server Address | to | Group [1,9999]

GROUPS: SYNC ALL | REPLICA(S): ENABLE ALL | REPLICA(S): DISABLE ALL

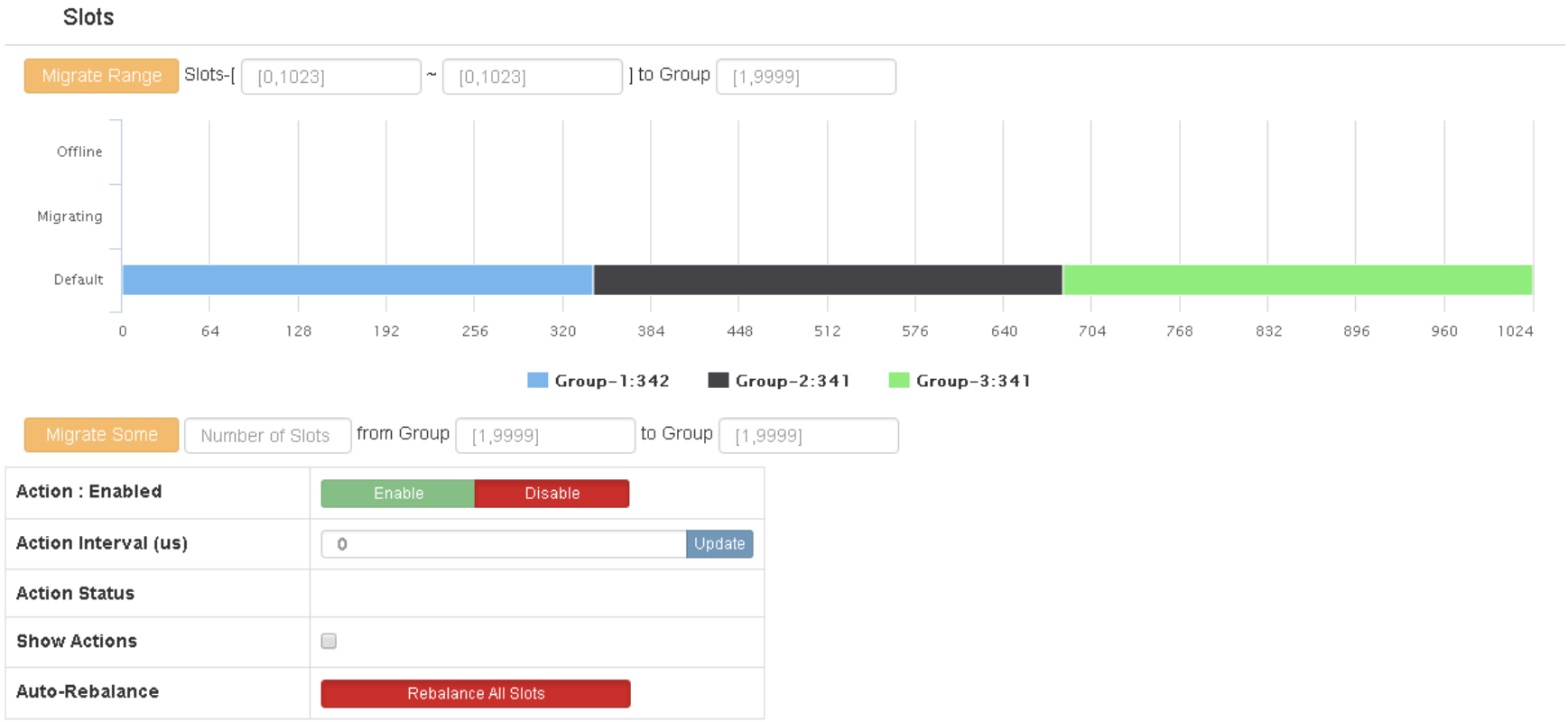
1	Server	Data Center	Master				Memory	Keys	
SYNC	S 10.0.5.161:6379 [HA]	1	NO:ONE	☐	🔧		7.98 MB / 2.00 GB	db0:keys=1,expires=0,avg_ttl=0	
PROMOTE	S 10.0.5.162:6379	1	10.0.5.161:6379:up	☐	🔧	synced	2.75 MB / 2.00 GB	db0:keys=1,expires=0,avg_ttl=0	−

2	Server	Data Center	Master				Memory	Keys	
SYNC	S 10.0.5.163:6379 [HA]	1	NO:ONE	☐	🔧		8.11 MB / 2.00 GB	db0:keys=2,expires=0,avg_ttl=0	
PROMOTE	S 10.0.5.164:6379	1	10.0.5.163:6379:up	☐	🔧	synced	2.75 MB / 2.00 GB	db0:keys=2,expires=0,avg_ttl=0	−

3	Server	Data Center	Master				Memory	Keys	
SYNC	S 10.0.5.165:6379 [HA]	1	NO:ONE	☐	🔧		8.11 MB / 2.00 GB	NA	
PROMOTE	S 10.0.5.166:6379	1	10.0.5.165:6379:up	☐	🔧	synced	2.75 MB / 2.00 GB	NA	−

添加 redis-sentinel：

分配 slot：



后续分配 codis-server 主从，同步 redis-sentinel、codis-proxy 即可；

四：keepalived+lvs 部署

4.1. lvs 规划：

lvs 环境	IP	业务端口	角色	服务描述
lvs 四层负载均衡	vip：10.0.5.11		对外业务 IP:port	
	vip：10.0.5.12		对外业务 IP:port	

	10.0.5.21	19000，	keepalived+lvs	lvs 四层反代负载均衡集群；keepalived 高可用 lvs，双主模型，两个 vip 分布于两台主机，各负载一半请求；
	10.0.5.22	19000，	keepalived+lvs	lvs 四层反代负载均衡集群；keepalived 高可用 lvs，双主模型，两个 vip 分布于两台主机，各负载一半请求；

RS 节点	角色	IP	业务端口	服务描述
	codis-proxy	10.0.5.151	admin：11080，data：19000	codis-server 高可用代理节点
	codis-proxy	10.0.5.152	admin：11080，data：19000	codis-server 高可用代理节点
	codis-proxy	10.0.5.153	admin：11080，data：19000	codis-server 高可用代理节点

vrrp，lvs 双主 vip：
vip1：10.0.5.11——10.0.5.21（MASTER）
vip2：10.0.5.12——10.0.5.22（MASTER）

real_server，codis-proxy 后端 RS 主机：
codis-proxy1：10.0.5.151
codis-proxy2：10.0.5.151
codis-proxy3：10.0.5.151

4.2.lvs+keepalived 配置

4.2.1.lvs21 配置：

```
# yum install keepalived
```

修改 keepalived 配置：

```
# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   notification_email {
     lasa_laka@xxx.cn
   }
   notification_email_from keepalived@lvs21
   smtp_server 42.120.219.25
```

```
   smtp_connect_timeout 30
   router_id 21
   vrrp_mcast_group 224.0.100.11
}


# vrrp config
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 11
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass lvs
    }
    virtual_ipaddress {
     10.0.5.11/24 dev eth0 label eth0:1
    }
    track_interface {
     eth0
   }
}

vrrp_instance VI_2 {
    state SLAVE
    interface eth0
    virtual_router_id 12
    priority 90
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass lvs
    }
    virtual_ipaddress {
```

```
     10.0.5.12/24 dev eth0 label eth0:2
    }
    track_interface {
     eth0
   }
}


# VS config for codis_proxy
virtual_server 10.0.5.11 19000 {
    delay_loop 3
    lb_algo wrr
    lb_kind DR
    nat_mask 255.255.255.0
    persistence_timeout 50
    protocol TCP

    sorry_server 127.0.0.1 80

    real_server 10.0.5.151 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }

    real_server 10.0.5.152 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
```

```
        }
    }

    real_server 10.0.5.153 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }
}

virtual_server 10.0.5.12 19000 {
    delay_loop 3
    lb_algo wrr
    lb_kind DR
    nat_mask 255.255.255.0
    persistence_timeout 50
    protocol TCP

    sorry_server 127.0.0.1 80

    real_server 10.0.5.151 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }

    real_server 10.0.5.152 19000 {
        weight 1
```

```
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }

    real_server 10.0.5.153 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }
}
```

启动 keepalived：

```
# service keepalived start
# chkconfig keepalived on
```

客户端连接：

```
# redis-cli -h 10.0.5.11 -p 19000
10.0.5.11:19000> GET name
"tom"
10.0.5.11:19000> HVALS city
1) "beijing"
2) "shanghai"
3) "wuhan"
4) "shenzhen"
```

查看 lvs 状态：

```
[root lvs21 11:21:03] ~
```

```
-- # ipvsadm -ln --stats
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port               Conns   InPkts  OutPkts  InBytes OutBytes
  -> RemoteAddress:Port
TCP  10.0.5.11:19000                    8       56        0     3266        0
  -> 10.0.5.151:19000                   0        0        0        0        0
  -> 10.0.5.152:19000                   0        0        0        0        0
  -> 10.0.5.153:19000                   8       56        0     3266        0
TCP  10.0.5.12:19000                    0        0        0        0        0
  -> 10.0.5.151:19000                   0        0        0        0        0
  -> 10.0.5.152:19000                   0        0        0        0        0
  -> 10.0.5.153:19000                   0        0        0        0        0

[root lvs21 11:21:57] ~
-- # ipvsadm -ln --stats
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port               Conns   InPkts  OutPkts  InBytes OutBytes
  -> RemoteAddress:Port
TCP  10.0.5.11:19000                    8       59        0     3422        0
  -> 10.0.5.151:19000                   0        0        0        0        0
  -> 10.0.5.152:19000                   0        0        0        0        0
  -> 10.0.5.153:19000                   8       59        0     3422        0
TCP  10.0.5.12:19000                    0        0        0        0        0
  -> 10.0.5.151:19000                   0        0        0        0        0
  -> 10.0.5.152:19000                   0        0        0        0        0
  -> 10.0.5.153:19000                   0        0        0        0        0
```

4.2.2.lvs22 配置：

```
# yum install keepalived
```

修改 keepalived 配置：

```
# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
```

```
   notification_email {
     lasa_laka@xxx.cn
   }
   notification_email_from keepalived@lvs22
   smtp_server 42.120.219.25
   smtp_connect_timeout 30
   router_id 22
   vrrp_mcast_group 224.0.100.11
}


# vrrp config
vrrp_instance VI_1 {
    state SLAVE
    interface eth0
    virtual_router_id 11
    priority 90
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass lvs
    }
    virtual_ipaddress {
     10.0.5.11/24 dev eth0 label eth0:1
    }
    track_interface {
     eth0
   }
}

vrrp_instance VI_2 {
    state MASTER
    interface eth0
    virtual_router_id 12
    priority 100
    advert_int 1
```

```
    authentication {
        auth_type PASS
        auth_pass lvs
    }
    virtual_ipaddress {
     10.0.5.12/24 dev eth0 label eth0:2
    }
    track_interface {
     eth0
   }
}


# VS config for codis_proxy
virtual_server 10.0.5.11 19000 {
    delay_loop 3
    lb_algo wrr
    lb_kind DR
    nat_mask 255.255.255.0
    persistence_timeout 50
    protocol TCP

    sorry_server 127.0.0.1 80

    real_server 10.0.5.151 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }

    real_server 10.0.5.152 19000 {
        weight 1
```

```
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }

    real_server 10.0.5.153 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }
}

virtual_server 10.0.5.12 19000 {
    delay_loop 3
    lb_algo wrr
    lb_kind DR
    nat_mask 255.255.255.0
    persistence_timeout 50
    protocol TCP

    sorry_server 127.0.0.1 80

    real_server 10.0.5.151 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
```

```
        }
    }

    real_server 10.0.5.152 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }

    real_server 10.0.5.153 19000 {
        weight 1
        TCP_CHECK {
         connect_port 19000
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }
}
```

启动 keepalived：

```
# service keepalived start
# chkconfig keepalived on
```

客户端连接：

```
# redis-cli -h 10.0.5.12 -p 19000
10.0.5.12:19000> get name
"tom"
10.0.5.12:19000> HVALS city
1) "beijing"
```

2) "shanghai"
3) "wuhan"
4) "shenzhen"

查看 lvs 状态：

```
[root lvs22 11:17:04] ~
-- # ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.5.11:19000 wrr persistent 50
  -> 10.0.5.151:19000             Route   1      0          0
  -> 10.0.5.152:19000             Route   1      0          0
  -> 10.0.5.153:19000             Route   1      0          0
TCP  10.0.5.12:19000 wrr persistent 50
  -> 10.0.5.151:19000             Route   1      0          0
  -> 10.0.5.152:19000             Route   1      0          0
  -> 10.0.5.153:19000             Route   1      1          0

[root lvs22 11:24:48] ~
-- # ipvsadm -ln --stats
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port               Conns   InPkts  OutPkts  InBytes OutBytes
  -> RemoteAddress:Port
TCP  10.0.5.11:19000                    0        0        0        0        0
  -> 10.0.5.151:19000                   0        0        0        0        0
  -> 10.0.5.152:19000                   0        0        0        0        0
  -> 10.0.5.153:19000                   0        0        0        0        0
TCP  10.0.5.12:19000                    1       15        0      867        0
  -> 10.0.5.151:19000                   0        0        0        0        0
  -> 10.0.5.152:19000                   0        0        0        0        0
  -> 10.0.5.153:19000                   1       15        0      867        0
```

4.3.后端 codis-proxy 的 RS 配置

后端 codis-proxy151 的 RS 配置脚本：

```
[root codis_proxy151 11:33:57] ~
-- # cat /etc/init.d/RS_lvs_dr.sh
#!/bin/bash
#
# chkconfig: 345 89 3
# description: lvs_dr Realserver config


vip1=10.0.5.11
vip2=10.0.5.12
mask='255.255.255.255'

case $1 in
start)
     echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
     echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore
     echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
     echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce

     ifconfig lo:1 $vip1 netmask $mask broadcast $vip1 up
     ifconfig lo:2 $vip2 netmask $mask broadcast $vip2 up
     route add -host $vip1 dev lo:1
     route add -host $vip2 dev lo:2
     ;;
stop)
     ifconfig lo:1 down
     ifconfig lo:2 down

     echo 0 > /proc/sys/net/ipv4/conf/all/arp_ignore
     echo 0 > /proc/sys/net/ipv4/conf/lo/arp_ignore
     echo 0 > /proc/sys/net/ipv4/conf/all/arp_announce
     echo 0 > /proc/sys/net/ipv4/conf/lo/arp_announce

     ;;
*)
```

```
        echo "Usage $(basename $0) start|stop"
        exit 1
        ;;
esac
```

启用 RS 配置：

```
#  /etc/init.d/RS_lvs_dr.sh  start
```

将 RS 配置脚本添加为服务，并设为开机自启动：

```
# chkconfig --add RS_lvs_dr.sh
# chkcofnig RS_lvs_dr.sh on
```

各 codis-proxy 的 RS_lvs_dr 配置按此配置即可；

五：codis 集群测试

5.1.可用性测试

测试 1：

在一台 codis-proxy 上写入数据，在另一台 codis-proxy 上读取数据；

```
# redis-cli -h 10.0.5.151 -p 19000
10.0.5.151:19000> SET name tom
OK
10.0.5.151:19000> HMSET city 1 beijing 2 shanghai 3 wuhan 4 shenzhen
OK


# redis-cli -h 10.0.5.152 -p 19000
10.0.5.152:19000> GET name
"tom"
10.0.5.152:19000> HGETALL city
1) "1"
2) "beijing"
3) "2"
4) "shanghai"
5) "3"
```

6) "wuhan"
7) "4"
8) "shenzhen"

测试 2：
通过 vip 访问 codis 服务读写数据；

测试 2：
任意宕机一台主的 codis-server，观察主从切换；

测试 2：
任意宕机一台 codis-proxy，观察数据读写是否有影响；

5.2.性能测试

```
# redis-benchmark -h 10.0.5.11 -p 19000 -q -d 100 |tee ./codis_test1
PING_INLINE: 50428.64 requests per second
PING_BULK: 58309.04 requests per second
SET: 45475.22 requests per second
GET: 51948.05 requests per second
INCR: 55187.64 requests per second
LPUSH: 49751.24 requests per second
RPUSH: 50428.64 requests per second
LPOP: 54406.96 requests per second
RPOP: 54112.55 requests per second
SADD: 61996.28 requests per second
SPOP: 62656.64 requests per second
LPUSH (needed to benchmark LRANGE): 50175.61 requests per second
LRANGE_100 (first 100 elements): 10636.03 requests per second
LRANGE_300 (first 300 elements): 3593.24 requests per second
LRANGE_500 (first 450 elements): 2400.90 requests per second
LRANGE_600 (first 600 elements): 1803.43 requests per second
MSET (10 keys): 13351.14 requests per second

# redis-benchmark -h 10.0.5.12 -p 19000 -n 100000 -c 100 |tee ./codis_test2
====== PING_INLINE ======
  100000 requests completed in 1.72 seconds
```

```
  100 parallel clients
  3 bytes payload
  keep alive: 1

23.51% <= 1 milliseconds
99.76% <= 2 milliseconds
99.96% <= 3 milliseconds
99.99% <= 4 milliseconds
100.00% <= 8 milliseconds
100.00% <= 8 milliseconds
58004.64 requests per second

====== PING_BULK ======
  100000 requests completed in 1.39 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

30.09% <= 1 milliseconds
99.98% <= 2 milliseconds
100.00% <= 2 milliseconds
72098.05 requests per second

====== SET ======
  100000 requests completed in 1.70 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

2.41% <= 1 milliseconds
98.91% <= 2 milliseconds
100.00% <= 2 milliseconds
58927.52 requests per second

====== GET ======
  100000 requests completed in 1.39 seconds
```

```
  100 parallel clients
  3 bytes payload
  keep alive: 1

20.67% <= 1 milliseconds
99.85% <= 2 milliseconds
99.98% <= 3 milliseconds
100.00% <= 3 milliseconds
71787.51 requests per second

====== INCR ======
  100000 requests completed in 1.34 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

31.98% <= 1 milliseconds
99.20% <= 2 milliseconds
99.90% <= 16 milliseconds
99.97% <= 17 milliseconds
100.00% <= 17 milliseconds
74626.87 requests per second

====== LPUSH ======
  100000 requests completed in 1.36 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

23.96% <= 1 milliseconds
99.44% <= 2 milliseconds
99.89% <= 3 milliseconds
99.90% <= 13 milliseconds
99.95% <= 14 milliseconds
100.00% <= 14 milliseconds
73637.70 requests per second
```

```
====== RPUSH ======
  100000 requests completed in 1.41 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

17.59% <= 1 milliseconds
99.79% <= 2 milliseconds
99.90% <= 3 milliseconds
99.97% <= 4 milliseconds
100.00% <= 4 milliseconds
71073.21 requests per second

====== LPOP ======
  100000 requests completed in 1.40 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

20.78% <= 1 milliseconds
99.88% <= 2 milliseconds
99.90% <= 3 milliseconds
99.95% <= 4 milliseconds
100.00% <= 5 milliseconds
71633.23 requests per second

====== RPOP ======
  100000 requests completed in 1.34 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

27.53% <= 1 milliseconds
99.93% <= 2 milliseconds
100.00% <= 2 milliseconds
```

```
74404.77 requests per second

====== SADD ======
  100000 requests completed in 1.27 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

59.23% <= 1 milliseconds
100.00% <= 2 milliseconds
78740.16 requests per second

====== SPOP ======
  100000 requests completed in 1.42 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

38.90% <= 1 milliseconds
99.97% <= 2 milliseconds
100.00% <= 2 milliseconds
70274.07 requests per second

====== LPUSH (needed to benchmark LRANGE) ======
  100000 requests completed in 1.31 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

31.29% <= 1 milliseconds
99.88% <= 2 milliseconds
99.92% <= 3 milliseconds
100.00% <= 4 milliseconds
100.00% <= 4 milliseconds
76394.20 requests per second
```

```
====== LRANGE_100 (first 100 elements) ======
  100000 requests completed in 3.33 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

0.00% <= 1 milliseconds
14.00% <= 2 milliseconds
98.09% <= 3 milliseconds
99.60% <= 4 milliseconds
99.79% <= 5 milliseconds
99.85% <= 6 milliseconds
99.90% <= 7 milliseconds
99.92% <= 8 milliseconds
99.95% <= 9 milliseconds
99.98% <= 10 milliseconds
100.00% <= 11 milliseconds
100.00% <= 11 milliseconds
30057.11 requests per second

====== LRANGE_300 (first 300 elements) ======
  100000 requests completed in 7.60 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

0.00% <= 1 milliseconds
0.01% <= 2 milliseconds
2.47% <= 3 milliseconds
46.39% <= 4 milliseconds
65.00% <= 5 milliseconds
95.20% <= 6 milliseconds
98.85% <= 7 milliseconds
99.43% <= 8 milliseconds
99.70% <= 9 milliseconds
99.85% <= 10 milliseconds
```

```
99.93% <= 11 milliseconds
99.97% <= 12 milliseconds
99.99% <= 13 milliseconds
100.00% <= 14 milliseconds
13154.43 requests per second

====== LRANGE_500 (first 450 elements) ======
  100000 requests completed in 11.12 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

0.00% <= 1 milliseconds
0.00% <= 2 milliseconds
0.02% <= 3 milliseconds
1.54% <= 4 milliseconds
20.80% <= 5 milliseconds
55.49% <= 6 milliseconds
69.58% <= 7 milliseconds
86.17% <= 8 milliseconds
96.50% <= 9 milliseconds
98.74% <= 10 milliseconds
99.12% <= 11 milliseconds
99.34% <= 12 milliseconds
99.57% <= 13 milliseconds
99.76% <= 14 milliseconds
99.87% <= 15 milliseconds
99.97% <= 16 milliseconds
100.00% <= 16 milliseconds
8989.57 requests per second

====== LRANGE_600 (first 600 elements) ======
  100000 requests completed in 15.37 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1
```

0.00% <= 1 milliseconds
0.00% <= 2 milliseconds
0.01% <= 3 milliseconds
0.03% <= 4 milliseconds
0.32% <= 5 milliseconds
5.43% <= 6 milliseconds
31.34% <= 7 milliseconds
50.53% <= 8 milliseconds
62.77% <= 9 milliseconds
78.34% <= 10 milliseconds
90.52% <= 11 milliseconds
95.72% <= 12 milliseconds
97.72% <= 13 milliseconds
98.38% <= 14 milliseconds
98.70% <= 15 milliseconds
98.93% <= 16 milliseconds
99.11% <= 17 milliseconds
99.32% <= 18 milliseconds
99.51% <= 19 milliseconds
99.66% <= 20 milliseconds
99.78% <= 21 milliseconds
99.88% <= 22 milliseconds
99.92% <= 23 milliseconds
99.94% <= 24 milliseconds
99.97% <= 25 milliseconds
99.99% <= 26 milliseconds
100.00% <= 27 milliseconds
100.00% <= 27 milliseconds
6506.60 requests per second

====== MSET (10 keys) ======
  100000 requests completed in 3.70 seconds
  100 parallel clients
  3 bytes payload
  keep alive: 1

```
0.00% <= 1 milliseconds
0.32% <= 2 milliseconds
26.71% <= 3 milliseconds
83.76% <= 4 milliseconds
93.04% <= 5 milliseconds
94.51% <= 6 milliseconds
94.70% <= 7 milliseconds
99.34% <= 8 milliseconds
99.83% <= 9 milliseconds
99.90% <= 10 milliseconds
99.97% <= 11 milliseconds
100.00% <= 11 milliseconds
27048.96 requests per second
```

六：总结

此文档对近两天来搭建 codis 环境做个梳理总结，文档内容偏向于实施部署，缺少对各配置项深入介绍，尚需读者查阅其他文档汇总梳理；
文档中还有需要完善的地方，如各 codis 服务脚本需要完善，服务探活、尤其是 codis-proxy 服务的高可用脚本需要完善；

最后，感谢马哥教育、感谢 redis/codis 技术交流群！

By：一苇