目录

[服务器规格概述 2](#_Toc25019)

[服务器架构简述 2](#_Toc7697)

[17玩直播平台架构 2](#_Toc31454)

[HDT主站高可用架构 3](#_Toc23038)

[服务器登陆操作 4](#_Toc6972)

[服务器安全权限 5](#_Toc15896)

[服务器应用 5](#_Toc23151)

[nginx 5](#_Toc3219)

[Install 5](#_Toc24282)

[Configuration 5](#_Toc29534)

[php-fpm 6](#_Toc890)

[Install 6](#_Toc23328)

[Configuration 7](#_Toc10608)

[mariadb 8](#_Toc14118)

[Install 8](#_Toc31428)

[Configuration 9](#_Toc9630)

[supervisor 9](#_Toc20512)

[Install 9](#_Toc27353)

[Configuration 9](#_Toc14752)

[project server management 10](#_Toc26472)

[Go server(17玩) 10](#_Toc27368)

[Go server(互动力) 11](#_Toc26702)

[redis 11](#_Toc3357)

[Install 11](#_Toc27039)

[Configuration 12](#_Toc19830)

[vsftpd 16](#_Toc26440)

[Install 16](#_Toc30078)

[Configuration 16](#_Toc29901)

[docker cluster 17](#_Toc574)

[Install 17](#_Toc23616)

[一、搭建docker集群环境 18](#_Toc6634)

[二、配置consul集群 19](#_Toc17864)

[三、registrator状态获取 20](#_Toc31689)

[四、安装shipyard、swarm 21](#_Toc13413)

[五、安装haproxy或者nginx（192.168.1.14） 22](#_Toc9317)

[六、安装consul-template 实现服务自动发现 26](#_Toc18906)

[七、测试是否自动发现 32](#_Toc28752)

[服务器脚本简述 33](#_Toc4834)

[服务器定时任务 33](#_Toc6655)

[香港主机定时任务 33](#_Toc26142)

[大陆服务器定时任务 34](#_Toc5001)

###### 服务器规格概述

一

计费方式：包年包月

地域：华南（随机分配）

实例规格：ecs.sn2.xlarge（8 核 32GB，通用型(原独享) sn2）

I/O 优化：I/O 优化实例

网络：公网带宽 100Mbps（经典网络）

镜像：CentOS 7.4 64位

系统盘：SSD 云盘（500 GB）

流量计费：按量计费（1元/GB）

购买量：1 年 4台

二

计费方式：包年包月

地域：香港（随机分配）

实例规格：ecs.sn2.xlarge（8 核 32GB，通用型(原独享) sn2）

I/O 优化：I/O 优化实例

网络：公网带宽 100Mbps（经典网络）

镜像：CentOS 7.4 64位

系统盘：SSD 云盘（500 GB）

流量计费：按量计费（1元/GB）

购买量：1 年 1台

###### 服务器架构简述

Linux服务器均使用Centos7.4系统

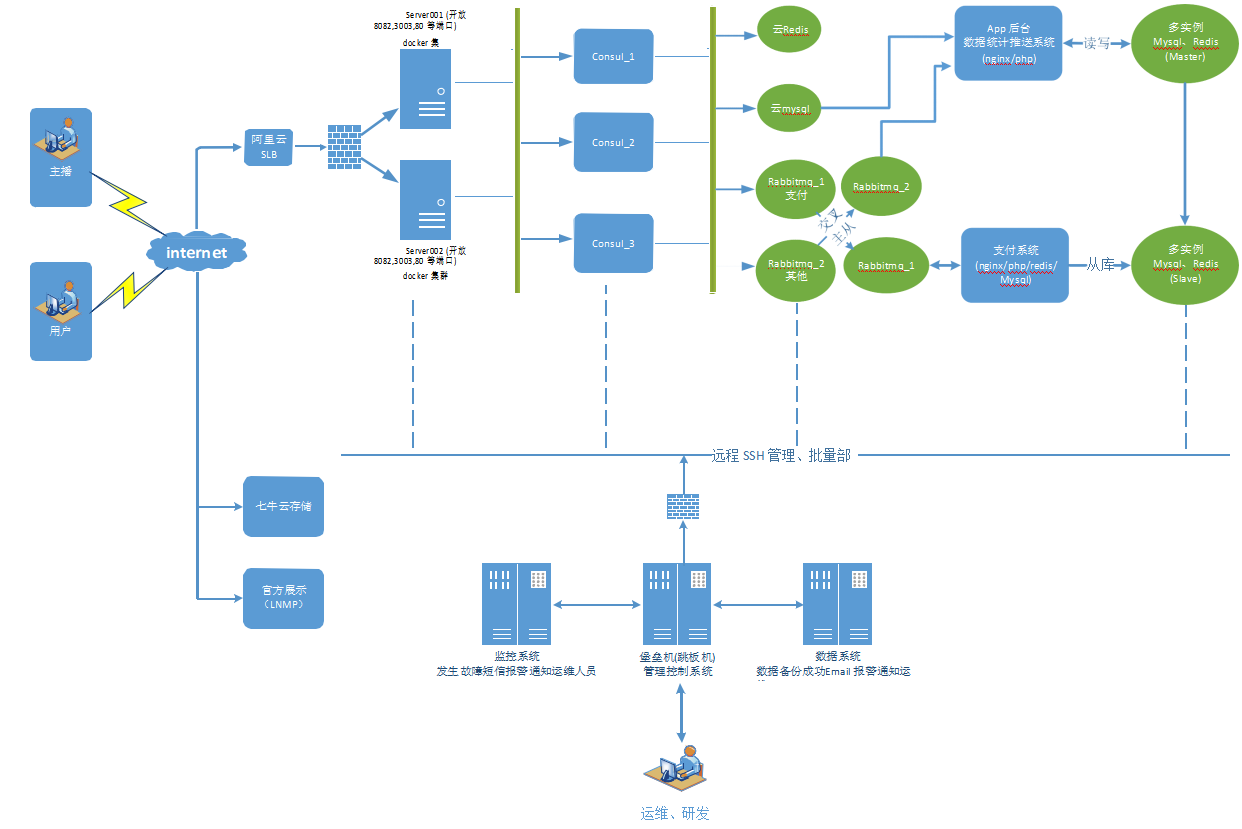
Linux服务器有部署nginx-1.13.1，php-7，mariadb-10.1.10的WEB架构（LNMP）

项目服务器端采用go语言开发

阿里云华南区域服务器搭建了基于docker的服务器集群（Swarm，swarm-agent，consul，shipyard，registrator）

17玩直播平台架构

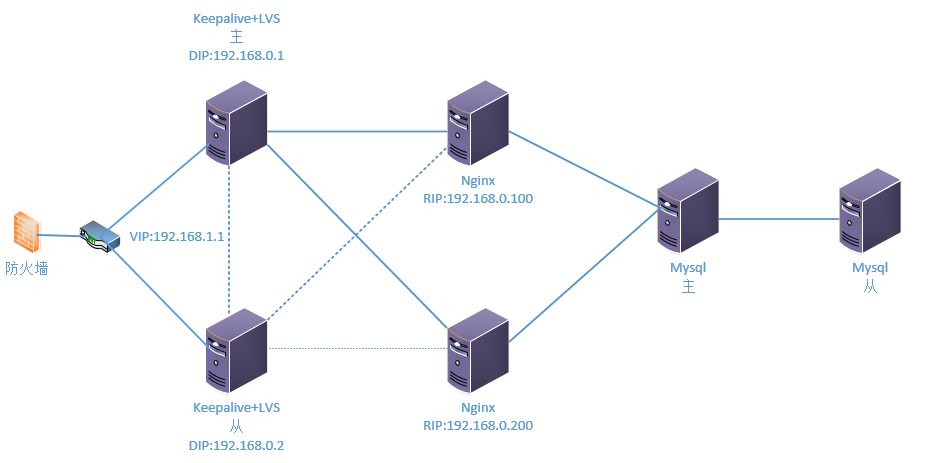
前端用阿里云SLB做负载均衡，



HDT主站高可用架构

|  |  |  |  |
| --- | --- | --- | --- |
| 地址规划表 | | | |
| 节点类型 | 主机名 | IP列表 | 网络类型 | |
| 主DS | DR1 | eth0:192.168.0.1 | Public IP | |
|  | 192.168.1.1 | Virtual IP | |
| 从DS | DR2 | eth0:192.168.0.2 | Public IP | |
|  | 192.168.1.1 | Virtual IP | |
| Real server 1 | RS1 | Eth0:192.168.0.100 | Public IP | |
| Real server 2 | RS2 | Eth0:192.168.0.200 | Public IP | |
| Mysqldb 1 | DB1 | Eth0:192.168.0.50 | Public IP | |
| Mysqldb 2 | DB2 | Eth0:192.168.0.52 | Public IP | |

keepalived是以VRRP协议为实现基础的，VRRP全称Virtual Router Redundancy Protocol，即虚拟路由冗余协议。通过VRRP协议结合LVS，对组群服务器监控情况，若master出现宕机情况，则将VIP漂移到backup机上。实现了分布式系统高可用。可以理解为：keepalived是LVS的管理软件，根据监控情况，将宕机服务器从ipvsadm移除掉。后端真是服务器Real server连接的数据库为主从同步，读写分离，保证数据库的高可用



###### 服务器登陆操作

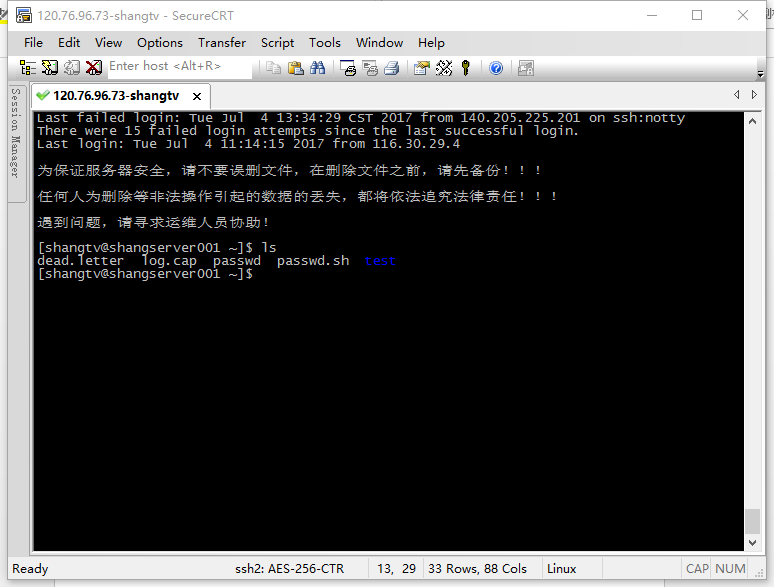
可使用ssh工具登陆服务器，比如 CecureCRT、putty等等

创建用户可使用/scripts/adduser.sh脚本创建用户

用户第一次登陆强制修改密码

用户使用个人账号，请勿使用公共账号登陆，已禁用root远程登陆

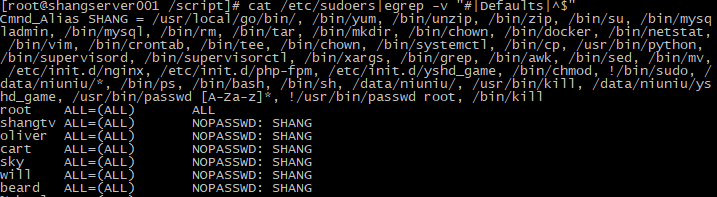
可以修改个人密码：sudo passwd {*username*}



###### 服务器安全权限

便于管理和安全，创建的用户已添加了运行命令权限限制，如遇到某个命令无法运行，请联系运维！

权限是通过 sudoers进行限制配置文件/etc/sudoers



###### 服务器应用

所有应用安装的源码包统一放在/software

nginx

Nginx主配置文件：/usr/local/nginx/conf/nginx.conf

Nignx站点配置目录：/usr/local/nginx/conf.d/

服务热重启命令：sudo /etc/init.d/nginx reload

服务完全重启命令：sudo /etc/init.d/nginx restart

Install

useradd www -s /sbin/nologin

yum -y install pcre pcre-devel zlib zlib-devel gcc-c++ gcc openssl\*

tar zxvf nginx-1.12.0.tar.gz

cd nginx-1.12.0/

./configure --user=www --group=www --prefix=/usr/local/nginx --with-http\_realip\_module --with-http\_sub\_module --with-http\_gzip\_static\_module --with-http\_stub\_status\_module --with-pcre

make && make install

Configuration

ln -s /usr/local/nginx/sbin/nginx /sbin/nginx

cat >> /usr/lib/systemd/system/nginx.service << EOF

[Unit]

Description=nginx - high performance web server

Documentation=http://nginx.org/en/docs/

After=network-online.target remote-fs.target nss-lookup.target

[Service]

Type=forking

PIDFile=/usr/local/nginx/logs/nginx.pid

ExecStartPre=/usr/sbin/nginx -t

ExecStart=/usr/sbin/nginx

ExecReload=/usr/sbin/nginx -s reload

ExecStop=/usr/sbin/nginx -s stop

PrivateTmp=true

[Install]

WantedBy=multi-user.target

EOF

systemctl daemon-reload

systemctl start nginx

systemctl enable nginx

systemctl status nginx

php-fpm

port：9000

php配置文件：/usr/local/php/etc/php.ini

php-fpm配置文件:/usr/local/php/etc/php-fpm.conf

/usr/local/php/etc/php-fpm.d/www.conf

服务热重启命令：sudo /etc/init.d/php-fpm reload

服务完全重启命令：sudo /etc/init.d/php-fpm restart

Install

cd /software

tar zxvf php-7.1.4.tar.gz

cd php-7.1.4/

./configure --help

yum -y install libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib zlib-devel curl curl-devel openssl openssl-devel

#./configure --prefix=/usr/local/php --enable-fpm --with-fpm-user=nginx --with-fpm-group=nginx --with-mysqli --with-zlib --with-curl --with-gd --with-jpeg-dir --with-png-dir --with-freetype-dir --with-openssl --enable-mbstring --enable-xml --enable-session --enable-ftp --enable-pdo -enable-tokenizer --enable-zip

./configure --prefix=/usr/local/php \

--with-config-file-path=/usr/local/php/etc \

--enable-mysqlnd \

--with-mysql=shared,mysqlnd \

--with-mysqli=shared,mysqlnd \

--with-pdo-mysql=shared,mysqlnd \

--with-mysql-sock=/var/lib/mysql/mysql.sock \

--with-mysqli=/usr/bin/mysql\_config \

--with-gd \

--with-png-dir \

--with-jpeg-dir \

--with-freetype-dir \

--with-xpm-dir \

--with-zlib-dir \

--with-iconv \

--enable-fpm \

--with-fpm-user=nginx \

--with-fpm-group=nginx \

--enable-libxml \

--enable-xml \

--enable-bcmath \

--enable-shmop \

--enable-sysvsem \

--enable-inline-optimization \

--enable-opcache \

--enable-mbregex \

--enable-mbstring \

--enable-ftp \

--enable-gd-native-ttf \

--with-openssl \

--enable-pcntl \

--enable-sockets \

--with-xmlrpc \

--enable-zip \

--enable-soap \

--without-pear \

--with-gettext \

--enable-session \

--with-mcrypt \

--with-curl \

--enable-exif \

--with-mhash \

--enable-ctype

make && make install

Configuration

cp php.ini-development /usr/local/php/lib/php.ini

grep mysqli.default\_socket /usr/local/php/lib/php.ini

sed -i 's#mysqli.default\_socket =#mysqli.default\_socket = /usr/local/mysql/mysql.sock#' /usr/local/php/lib/php.ini

grep mysqli.default\_socket /usr/local/php/lib/php.ini

grep date.timezone /usr/local/php/lib/php.ini

sed -i 's#;date.timezone =#date.timezone = Asia/Shanghai#' /usr/local/php/lib/php.ini

grep date.timezone /usr/local/php/lib/php.ini

/usr/local/php/bin/php -v

/usr/local/php/bin/php -m

cp /usr/local/php/etc/php-fpm.conf.default

cp /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf

cp /usr/local/php/etc/php-fpm.d/www.conf.default /usr/local/php/etc/php-fpm.d/www.conf

grep -E 'user =|group =' /usr/local/php/etc/php-fpm.d/www.conf

sed -i 's#user = nginx#user = www#' /usr/local/php/etc/php-fpm.d/www.conf

sed -i 's#group = nginx#group = www#' /usr/local/php/etc/php-fpm.d/www.conf

grep -E 'user =|group =' /usr/local/php/etc/php-fpm.d/www.conf

cp sapi/fpm/php-fpm.service /usr/lib/systemd/system/

grep -E 'PIDFile|ExecStart' /usr/lib/systemd/system/php-fpm.service

systemctl daemon-reload

systemctl enable php-fpm

systemctl start php-fpm

systemctl status php-fpm

mariadb

port：3306

mariadb配置文件：/etc/my.cnf

重启服务：/etc/init.d/mysql restart

（需要在root下操作）

Install

cat > /etc/yum.repos.d/mariadb.repo <<EOF

[mariadb]

name = MariaDB

baseurl = http://yum.mariadb.org/10.1.10/centos7-amd64

gpgkey = https://yum.mariadb.org/RPM-GPG-KEY-MariaDB

gpgcheck = 1

EOF

yum clean all

yum makecache

#install Development tools

#yum -y groupinstall "Development Tools"

#install mariadb

yum -y install MariaDB-\*

Configuration

#start MariaDB

service mysql start

#setting mariadb

mysql\_secure\_installation

supervisor

用途：服务器进程管理，可用于服务加入到守护进程

supervisorctl端口：3001

停止supervisord服务：systemctl stop supervisord

或者 kill -9



启动supervisord服务：

sudo supervisord –c /etc/supervisor/supervisord.conf

Install

系统：centos7，默认在root用户下安装，使用普通用户，可使用sudo

yum install python-setuptools -y

easy\_install supervisor #如果没看到报错信息，就说明安装成功了，可以使用echo\_supervisord\_conf查看配置详情，而后生成配置文件

mkdir -m 755 -p /etc/supervisor/

echo\_supervisord\_conf > /etc/supervisor/supervisord.conf

Configuration

#创建存放配置文件的目录

mkdir -m 755 /etc/supervisor/conf.d

#在主配置文件中引入配置

vim /etc/supervisor/supervisord.conf

#注释去掉

[include]

files = ./conf.d/\*.ini

#在/etc/supervisor/conf.d下面创建go-http-server.ini

;--------------------------------------------------------

[program:yshd\_game]

directory = /data/niuniu

command= /data/niuniu/yshd\_game

user=root

process\_name= yshd\_game

autostart=true

autorestart=true

startretries=3

startsecs=10

stdout\_logfile=/var/log/yshd\_game-server.log

stdout\_logfile\_maxbytes=1MB

stdout\_logfile\_backups=10

stdout\_capture\_maxbytes=1MB

stderr\_logfile=/var/log/yshd\_game-server.log

stderr\_logfile\_maxbytes=1MB

stderr\_logfile\_backups=10

stderr\_capture\_maxbytes=1MB

;-----------------------------------------------------------

#启动supervisor服务，注意：启动服务的时候，不需要启动goweb，不然会报错

/usr/bin/supervisord -c /etc/supervisor/supervisord.conf

查看supervisor运行状态

supervisorctl

supervisor>status

修改配置文件重新加载

supervisorctl reload

supervisorctl shutdown #关闭supervisor

supervisorctl restart #重启supervisor

project server management

Go server(17玩)

服务器公网IP： 120.76.96.73

服务器内网IP：10.45.44.249

服务器存放区域：阿里云华南机房

port：3003

服务器进程管理使用supervisorctl工具，具体配置参照上面supervisor

重启服务器端：sudo supervisorctl restart yshd\_game

服务器更新脚本：sh /script/server-options\_17wan.sh

规则如下

1.为避免升级版本错乱，版本升级统一由oliver管理，cart和sky只需负责上传编译好的程序包，如遇特殊情况，可以自行处理

2.版本升级应在固定时间处理，比如：每周四升级服务器版本，不需要频繁更新，待测试通过没问题之后才能上线到服务器

3.服务器升级统一使用脚本运行，不能使用手动kill的方式，如需手动停止服务器，使用命令 sudo supervisorctl stop yshd\_game

4.cart和sky的都使用自己分配的用户登陆服务器，这样可以看到是哪个用户上传的程序

5.每次编译包之前，先合并代码，避免没有编译最新的代码

6.程序统一打包成zip文件之后再上传，上传zip的文件夹： /data/server-package/，如果文件夹已经有yshd\_game.zip包，则先删除，再上传，脚本会自动在更新服务器后自动把包名后面添加时间戳

7.脚本运行之后，都会自动保存前一个版本的程序文件。保存路径：/data/server-backup/

8.日志文件：/var/log/yshd\_game.log

9.定时更新任务，操作用户是shangtv，配置crontab，如需定时更新，只需要把crontab的内容#注释去掉保存即可，默认是每周四4点运行脚本 sudo crontab -u shangtv -e

Go server(互动力)

服务器公网IP：47.75.16.43

服务器内网IP：10.80.218.19

服务器存放区域：阿里云香港机房

Port:3003

服务器进程管理使用supervisorctl工具，具体配置参照上面supervisor

重启服务端：sudo supervisorctl restart gateway

sudo supervisorctl restart usermeta

项目路径：/project/hdtap

redis

port：6379

配置文件：/etc/redis.conf

重启redis服务：/etc/init.d/redis restart

需要在root权限下操作

Install

1.安装redis组件

#yum install tcl

2.下载redis源码包

cd /usr/local/src

wget http://download.redis.io/releases/redis-3.0.7.tar.gz

#tar -zvxf redis-3.0.7.tar.gz

#cd redis-3.0.7

#make

#make install #按提示install

#cd src/

#make install

下面提示就说明成功

Hint: To run 'make test' is a good idea ;)

INSTALL install

INSTALL install

INSTALL install

INSTALL install

INSTALL install

以上redis已完成编译安装

在/usr/local/bin 下面就存在了redis-benchmark redis-check-aof redis-check-dump redis-cli redis-sentinel redis-server六个文件，其中redis-server为启动服务

配置redis文件

#cp /usr/local/src/redis-3.0.7/redis.conf /etc/

#chmod 755 /etc/redis.conf

启动redis

#/usr/local/bin/redis-server /etc/redis.conf

Configuration

修改redis.conf配置文件 vim /etc/redis.conf

下面是redis.conf的主要配置参数的意义：

daemonize：是否以后台daemon方式运行

pidfile：pid文件位置

port：监听的端口号

timeout：请求超时时间

loglevel：log信息级别

logfile：log文件位置

databases：开启数据库的数量

save \* \*：保存快照的频率，第一个\*表示多长时间，第三个\*表示执行多少次写操作。在一定时间内执行一定数量的写操作时，自动保存快照。可设置多个条件。

rdbcompression：是否使用压缩

requirepass: redis管理员密码

dbfilename：数据快照文件名（只是文件名，不包括目录）

dir：数据快照的保存目录（这个是目录）

appendonly：是否开启appendonlylog，开启的话每次写操作会记一条log，这会提高数据抗风险能力，但影响效率。

appendfsync：appendonlylog如何同步到磁盘（三个选项，分别是每次写都强制调用fsync、每秒启用一次fsync、不调用fsync等待系统自己同步）

4.修改 daemonize yes,以后台运行

这时你可以打开一个终端进行测试了，配置文件中默认的监听端口是6379

5.建立用户和日志目录

第一次启动时建议为Redis建立用户和日志目录

#groupadd redis

#useradd -g redis redis -s /bin/false

#mkdir -p /var/lib/redis #db文件放在这里，需要修改redis.conf

#mkdir -p /var/log/redis #日志放在这里

#chown redis.redis /var/lib/redis

#chown redis.redis /var/log/redis

vim /etc/redis.conf

# The working directory.

#

# The DB will be written inside this directory, with the filename specified

# above using the 'dbfilename' configuration directive.

#

# Also the Append Only File will be created inside this directory.

#

# Note that you must specify a directory here, not a file name.

dir /var/lib/redis

---------------------------------------------------------------------------------------

找到日志logfile

# Specify the log file name. Also 'stdout' can be used to force

# Redis to log on the standard output. Note that if you use standard

# output for logging but daemonize, logs will be sent to /dev/null

logfile /var/log/redis/redislog

或者编写启动脚本，加入到服务

配置Init脚本

Redis管理脚本基于Ubuntu 的发行版上的，在Centos linux 上并不能用，下面有个脚本可以用于CentOS 。

用这个脚本管理之前，需要先配置下面的内核参数，否则Redis脚本在重启或停止redis时，将会报错，并且不能自动在停止服务前同步数据到磁盘上：

# vim /etc/sysctl.conf

vm.overcommit\_memory = 1

然后应用生效：

sysctl –p

建立redis启动脚本：

vim /etc/init.d/redis

#!/bin/bash

#

# Init file for redis

#

# chkconfig: - 80 12

# description: redis daemon

#

# processname: redis

# config: /etc/redis.conf

# pidfile: /var/run/redis.pid

source /etc/init.d/functions

#BIN="/usr/local/bin"

BIN="/usr/local/bin"

CONFIG="/etc/redis.conf"

PIDFILE="/var/run/redis.pid"

### Read configuration

[ -r "$SYSCONFIG" ] && source "$SYSCONFIG"

RETVAL=0

prog="redis-server"

desc="Redis Server"

start() {

if [ -e $PIDFILE ];then

echo "$desc already running...."

exit 1

fi

echo -n $"Starting $desc: "

daemon $BIN/$prog $CONFIG

RETVAL=$?

echo

[ $RETVAL -eq 0 ] && touch /var/lock/subsys/$prog

return $RETVAL

}

stop() {

echo -n $"Stop $desc: "

killproc $prog

RETVAL=$?

echo

[ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog $PIDFILE

return $RETVAL

}

restart() {

stop

start

}

case "$1" in

start)

start

;;

stop)

stop

;;

restart)

restart

;;

condrestart)

[ -e /var/lock/subsys/$prog ] && restart

RETVAL=$?

;;

status)

status $prog

RETVAL=$?

;;

\*)

echo $"Usage: $0 {start|stop|restart|condrestart|status}"

RETVAL=1

esac

exit $RETVAL

然后增加服务并开机自启动：

chmod 755 /etc/init.d/redis

chkconfig --add redis

chkconfig --level 345 redis on

chkconfig --list redis

/etc/init.d/redis restart

vsftpd

port：21

配置文件：/etc/vsftpd/vsftpd.conf

重启服务命令：sudo systemctl restart vsftpd

Vsftpuser：ftpuser

Passwd：\*\*\*\*\*\*

sshd

port：22

配置文件：/etc/ssh/sshd\_config

重启服务命令：sudo systemctl restart sshd

Install

用yum直接可安装

# yum install vsftpd

安装完成后在/etc/vsftpd/目录下面会生成下面5个文件

ftpusers user\_list vsftpd.conf vsftpd.conf\_bak vsftpd\_conf\_migrate.sh

vsftpd.conf就是vsftpd的主配置文件

Configuration

贴出配置，参照修改就行了

[root@shangserver001 ~]# cat /etc/vsftpd/vsftpd.conf|grep -v "#"

anonymous\_enable=NO #是否打开匿名访问

local\_enable=YES #是否本地目录访问

write\_enable=YES #是否可写

local\_umask=022 #默认umask权限

anon\_upload\_enable=YES #是否允许上传

anon\_mkdir\_write\_enable=YES #是否可以创建文件夹

dirmessage\_enable=YES

xferlog\_enable=YES

connect\_from\_port\_20=YES

xferlog\_file=/var/log/vsftpd.log

xferlog\_std\_format=NO

ftpd\_banner=欢迎进入云尚互动FTP服务.

chroot\_local\_user=YES

local\_root=/ftp #FTP根目录

allow\_writeable\_chroot=yes

chroot\_list\_file=/etc/vsftpd/chroot\_list

listen=no

listen\_ipv6=YES

pam\_service\_name=vsftpd

userlist\_enable=YES

tcp\_wrappers=YES

ftp目录与nginx目录关联

采用的mount的方式，具体操作如下

1. 创建ftp项目目录，最好和nginx站点目录一直，便于管理
2. 编辑/etc/fstab，在最下面添加如下，仅参考，具体需要修改对应的目录

/usr/local/nginx/html/hdt\_web /ftp/hdt\_web none rw,bind 0 0

/usr/local/nginx/html/hdt\_admin /ftp/hdt\_admin none rw,bind 0 0

3.修改完成之后保存，执行mount -a即可

docker cluster

sudo netstat –ntlp | grep docker 查看docker端口



查看docker容器命令

docker ps –a

查看docker images命令

docker images

停止docker 容器

docker stop {CONTAINER ID}

启动docker 容器

docker stop {CONTAINER ID}

重启docker服务

sudo systemctl daemon-reload

sudo systemctl restart docker

Install

环境：centos7.2

192.168.1.14 master swarm-manager rethinkdb controller swarm-agent consul-s1 registrator consul-template(nginx)

192.168.1.15 slave-1 registrator swarm-agent consul-s2

192.168.1.16 slave-2 registrator swarm-agent consul-s3

docker-engine 17.05.0-ce

一、搭建docker集群环境

1、先检查是否安装旧版本docker

rpm -qa|grep docker

如果有就先卸载

yum remove docker\*

2、添加docker.repo安装源，写入文件

tee /etc/yum.repos.d/docker.repo<<EOF

[dockerrepo]

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/centos/7/

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

EOF

3、安装docker

yum install docker-engine

4、配置防火墙

firewall-cmd --permanent --add-port={2375/tcp,3375/tcp,8500/tcp,8300/tcp,8301/tcp,8301/udp,8302/tcp,8302/udp,8400/tcp,8500/tcp,8600/tcp,8600/udp,8080/tcp,28015/tcp,29015/tcp}

firewall-cmd --reload

firewall-cmd --list-all

------------------------------------------------------------------------------------------------

5、增加tcp监听端口,并配置docker加速

修改/lib/systemd/system/docker.service

阿里云加速

sed -i 's/ExecStart=.\*/ExecStart=\/usr\/bin\/dockerd -H unix\:\/\/\/var\/run\/docker.sock -D -H tcp\:\/\/0.0.0.0\:2375 --registry-mirror=https\:\/\/0xl18ug0.mirror.aliyuncs.com --live-restore/g' /lib/systemd/system/docker.service

6、重启docker

systemctl daemon-reload

systemctl enable docker.service

systemctl restart docker.service

ps -ef |grep docker

#能看到docker启动以及2375端口

7、安装pip以及docker api

yum -y install epel-release

yum -y install python-pip

pip install docker-py docker-compose

8、创建consul用户及组

groupadd -g 1005 consul

useradd -u 105 -g 1005 -s /bin/false consul

9、创建consul数据存储文件夹

mkdir -p /opt/consul/{data,conf}

chown -R consul: /opt/consul

10、设置主机hosts，有多少台主机，就需要设置多少hosts

vim /etc/hosts

192.168.1.14 master.localhost.com

192.168.1.15 slave1.localhost.com

192.168.1.16 slave2.localhost.com

二、配置consul集群

1、拉取consul镜像

docker pull progrium/consul

#提示：目录没有官方出consul镜像，以上consul镜像是官方推荐的第三方docker image

2、启动consul server 192.168.1.14

docker run -d \

-p 8300:8300 \

-p 8301:8301 \

-p 8301:8301/udp \

-p 8302:8302 \

-p 8302:8302/udp \

-p 8400:8400 \

-p 8500:8500 \

-p 8600:53 \

-p 8600:53/udp \

-v /opt/consul/data:/data \

-h $HOSTNAME \

--restart=always \

--name=consul-s1 \

progrium/consul \

-server -bootstrap-expect=1 \

-ui-dir=/ui \

-client 0.0.0.0 \

-advertise 192.168.1.14

3、启动consul server 192.168.1.15

docker run -d \

-p 8300:8300 \

-p 8301:8301 \

-p 8301:8301/udp \

-p 8302:8302 \

-p 8302:8302/udp \

-p 8400:8400 \

-p 8500:8500 \

-p 8600:53 \

-p 8600:53/udp \

-v /opt/consul/data:/data \

-h consul-s2 \

--restart=always \

--name=consul-s2 \

progrium/consul \

-server \

-ui-dir=/ui \

-client 0.0.0.0 \

-advertise 192.168.1.15 -join 192.168.1.14

4、启动consul client 192.168.1.16

docker run -d \

-p 8300:8300 \

-p 8301:8301 \

-p 8301:8301/udp \

-p 8302:8302 \

-p 8302:8302/udp \

-p 8400:8400 \

-p 8500:8500 \

-p 8600:53 \

-p 8600:53/udp \

-v /opt/consul/data:/data \

-h consul-c1 \

--restart=always \

--name=consul-c1 \

progrium/consul \

-advertise 192.168.1.16 -join 192.168.1.14

三、registrator状态获取

依次启动

docker run -d \

--restart=always \

--name=registrator \

--net=host \

-v /var/run/docker.sock:/tmp/docker.sock \

gliderlabs/registrator \

-ip 192.168.1.14 \

consul://192.168.1.14:8500

docker run -d \

--restart=always \

--name=registrator \

--net=host \

-v /var/run/docker.sock:/tmp/docker.sock \

gliderlabs/registrator \

-ip 192.168.1.15 \

consul://192.168.1.15:8500

docker run -d \

--restart=always \

--name=registrator \

--net=host \

-v /var/run/docker.sock:/tmp/docker.sock \

gliderlabs/registrator \

-ip 192.168.1.16 \

consul://192.168.1.16:8500

四、安装shipyard、swarm

1、192.168.1.14

docker run -tid \

-p 3375:3375 \

--restart=always \

--name shipyard-swarm-manager \

swarm:latest \

manage --host tcp://0.0.0.0:3375 consul://192.168.1.14:8500

docker run -tid \

--restart=always \

--name=shipyard-rethinkdb \

-p 28015:28015 \

-p 29015:29015 \

-v /opt/rethinkdb:/data \

index.tenxcloud.com/docker\_library/rethinkdb

docker run -tid \

--restart=always \

--name shipyard-controller \

--link shipyard-rethinkdb:rethinkdb \

--link shipyard-swarm-manager:swarm \

-p 8080:8080 \

dockerclub/shipyard:latest \

server \

-d tcp://swarm:3375

docker run -tid \

--restart=always \

--name shipyard-swarm-agent \

swarm:latest \

join --addr 192.168.1.14:2375 consul://192.168.1.14:8500

2、安装swarm-agent

主机192.168.1.15操作

docker run -tid \

--restart=always \

--name shipyard-swarm-agent \

swarm:latest \

join --addr 192.168.1.15:2375 consul://192.168.1.14:8500

主机192.168.1.16操作

docker run -tid \

--restart=always \

--name shipyard-swarm-agent \

swarm:latest \

join --addr 192.168.1.16:2375 consul://192.168.1.14:8500

五、安装haproxy或者nginx（192.168.1.14）

1、安装haproxy

yum -y install git patch gcc gcc-c++ readline-devel zlib-devel libffi-devel \

openssl openssl-devel make autoconf automake libtool bison libxml2 \

libxml2-devel libxslt-devel libyaml-devel python python-docutils \

cmake imake expat-devel libaio libaio-devel bzr ncurses-devel wget \

libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel \

pcre-devel curl-devel libmcrypt libmcrypt-devel

cd /tmp

wget http://www.haproxy.org/download/1.7/src/haproxy-1.7.0.tar.gz

tar -xvf /tmp/haproxy-1.7.0.tar.gz

make TARGET=linux31 PREFIX=/opt/haproxy

make install PREFIX=/opt/haproxy

2、配置haproxy.conf

vim /opt/haproxy/conf/haproxy.conf

global

log 127.0.0.1 local0

#log 127.0.0.1 local1 notice

#log loghost local0 info

maxconn 50000

chroot /opt/haproxy

uid 99

gid 99

daemon

nbproc 2

pidfile /opt/haproxy/run/haproxy.pid

#debug

#quiet

defaults

mode tcp

option dontlognull

option forwardfor

option redispatch

retries 2

balance static-rr

stats enable

stats uri /ha?stats #haproxy运行状态查看 自定义uri

timeout connect 3000

timeout client 50000

timeout server 50000

listen admin\_stat

# 监听端口

bind \*:8888

# http的7层模式

mode http

#log global

# 统计页面自动刷新时间

stats refresh 30s

# 统计页面URL

stats uri /admin?stats

# 统计页面密码框上提示文本

stats realm Haproxy\ Statistics

# 统计页面用户名和密码设置

stats auth admin:admin

# 隐藏统计页面上HAProxy的版本信息

#stats hide-version

listen login

bind \*:9999

mode tcp

balance roundrobin

option httpchk

#maxconn 50000

#log 127.0.0.1 local0 debug

3、haproxy启动脚本 /etc/init.d/haproxy

#! /bin/bash

# chkconfig: - 85 15

# description: haproxy is a World Wide Web server. It is used to serve

PROGDIR=/opt/haproxy

PROGNAME=haproxy

DAEMON=$PROGDIR/sbin/$PROGNAME

CONFIG=$PROGDIR/conf/$PROGNAME.conf

PIDFILE=$PROGDIR/run/$PROGNAME.pid

DESC="HAProxy daemon"

SCRIPTNAME=/opt/haproxy/init.d/$PROGNAME

# Gracefully exit if the package has been removed.

test -x $DAEMON || exit 0

start()

{

echo -n "Starting $DESC: $PROGNAME"

$DAEMON -f $CONFIG

echo "."

}

stop()

{ echo -n "Stopping $DESC: $PROGNAME"

cat $PIDFILE | xargs kill

echo "."

}

reload()

{ echo -n "reloading $DESC: $PROGNAME"

$DAEMON -f $CONFIG -p $PIDFILE -sf $(cat $PIDFILE)

}

case "$1" in

start)

start

;;

stop)

stop

;;

reload)

reload

;;

\*)

echo "Usage: $SCRIPTNAME {start|stop|reload}" >&2

exit 1

;;

esac

exit 0

4、启动haproxy,并加入到开启启动

chmod +x /etc/init.d/haproxy

chkconfig haproxy op

service haproxy start

5、配置nginx 并支持数字证书（参照以上）

6、nginx配置文件

添加nginx默认web配置文件

/etc/nginx/conf.d/default.conf

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root /usr/local/nginx/html;

index index.php index.html index.htm;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

location ~ \.php$ {

root /usr/local/nginx/html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

location ~\* ^.+\.(jpg|jpeg|gif|png|bmp)$ {

access\_log off;

root opencart;

expires 30d;

break;

}

}

六、安装consul-template 实现服务自动发现

1、下载consul-template

cd /tmp

wget https://releases.hashicorp.com/consul-template/0.16.0/consul-template\_0.16.0\_linux\_amd64.zip

yum -y install unzip

unzip /tmp/consul-template\_0.16.0\_linux\_amd64.zip -d /usr/bin/

2、consul-template haproxy配置

cat > /opt/consul/conf/haproxy\_ctmpl.json << EOF

consul = "127.0.0.1:8500"

template {

source = "/opt/haproxy/conf/haproxy.ctmpl"

destination = "/opt/haproxy/conf/haproxy.conf"

command = "/etc/init.d/haproxy reload"

}

EOF

3、haproxy.ctmpl配置 /opt/haproxy/conf/haproxy.ctmpl

global

log 127.0.0.1 local0

#log 127.0.0.1 local1 notice

#log loghost local0 info

maxconn 50000

chroot /opt/haproxy

uid 99

gid 99

daemon

nbproc 2

pidfile /opt/haproxy/run/haproxy.pid

#debug

#quiet

defaults

mode tcp

option dontlognull

option forwardfor

option redispatch

retries 2

balance static-rr

stats enable

stats uri /ha?stats

timeout connect 3000

timeout client 50000

timeout server 50000

listen admin\_stat

bind \*:8888

mode http

#log global

stats refresh 30s

stats uri /admin?stats

stats realm Haproxy\ Statistics

stats auth admin:admin

#stats hide-version

frontend www

bind \*:80

mode http

acl apache hdr(HOST) apache.zone.com

acl nginx hdr(HOST) nginx.zone.com

use\_backend apache.qkazone.com if apache

use\_backend nginx.qkazone.com if nginx

backend apache.zone.com

balance roundrobin

mode http

{{range service "apache-php-80"}}

server apache {{.Address}}:{{.Port}} check {{end}}

backend nginx.zone.com

mode http

balance roundrobin

{{range service "nginx-80"}}

server nginx {{.Address}}:{{.Port}} check {{end}}

listen login

bind \*:9999

mode tcp

balance roundrobin

#log 127.0.0.1 local0 debug

{{range service "centos7"}}

server ssh {{.Address}}:{{.Port}} check {{end}}

4、配置consul-template haproxy 启动脚本 /etc/init.d/haproxy\_ctmpl

#!/bin/bash

# chkconfig: 2345 10 90

# description: Start and Stop redis

PATH=/usr/local/bin:/sbin:/usr/bin:/bin

EXEC=/usr/bin/consul-template

CONF="/opt/consul/conf/haproxy\_ctmpl.json"

case "$1" in

start)

PID=$(ps -ef | grep -v grep | grep "$EXEC -config $CONF" | awk '{print $2}')

if [ -n "$PID" ]

then

echo "haproxy\_ctmpl is running..."

else

echo "Starting haproxy\_ctmpl server..."

$EXEC -config $CONF > /tmp/haproxy\_ctmpl.out 2>&1 &

fi

;;

stop)

PID=$(ps -ef | grep -v grep | grep "$EXEC -config $CONF" | awk '{print $2}')

if [ -n "$PID" ]

then

echo "Stopping..."

kill -9 $PID

sleep 2

else

echo "haproxy\_ctmpl exists, process is not running."

fi

;;

restart|force-reload)

${0} stop

${0} start

;;

\*)

echo "Usage: /etc/init.d/tmpl {start|stop|restart|force-reload}" >&2

exit 1

esac

5、设置开机启动，启动

chmod +x /etc/init.d/haproxy\_ctmpl

chkconfig haproxy\_ctmpl on

service haproxy\_ctmpl start

6、consul-template nginx配置 /opt/consul/conf/nginx\_ctmpl.json

cat > /opt/consul/conf/nginx\_ctmpl.json << EOF

consul = "127.0.0.1:8500"

template {

source = "/etc/nginx/conf.d/nginx\_web.ctmpl"

destination = "/etc/nginx/conf.d/nginx\_web.conf"

command = "/usr/sbin/nginx -s reload"

}

EOF

7、/etc/nginx/conf.d/nginx\_web.ctmpl 配置

upstream apache {

ip\_hash;

# Refer: http://nginx.org/en/docs/http/ngx\_http\_upstream\_module.html#upstream

# least\_conn;

# least\_time;

{{range service "apache-php-80"}}

server {{.Address}}:{{.Port}} fail\_timeout=0;

{{end}}

keepalive 64;

}

server {

listen 80;

server\_name apache.zone.com;

location / {

client\_max\_body\_size 0;

proxy\_connect\_timeout 300s;

proxy\_send\_timeout 900;

proxy\_read\_timeout 900;

proxy\_buffer\_size 32k;

proxy\_buffers 4 32k;

proxy\_busy\_buffers\_size 64k;

proxy\_redirect off;

proxy\_hide\_header Vary;

proxy\_set\_header Accept-Encoding '';

proxy\_set\_header Host $host;

proxy\_set\_header Referer $http\_referer;

proxy\_set\_header Cookie $http\_cookie;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_headers\_hash\_max\_size 51200;

proxy\_headers\_hash\_bucket\_size 6400;

proxy\_pass http://apache/;

}

}

upstream nginx {

ip\_hash;

# Refer: http://nginx.org/en/docs/http/ngx\_http\_upstream\_module.html#upstream

# least\_conn;

# least\_time;

{{range service "nginx-80"}}

server {{.Address}}:{{.Port}} fail\_timeout=0;

{{end}}

keepalive 64;

}

server {

listen 80;

server\_name nginx.zone.com;

location / {

client\_max\_body\_size 0;

proxy\_connect\_timeout 300s;

proxy\_send\_timeout 900;

proxy\_read\_timeout 900;

proxy\_buffer\_size 32k;

proxy\_buffers 4 32k;

proxy\_busy\_buffers\_size 64k;

proxy\_redirect off;

proxy\_hide\_header Vary;

proxy\_set\_header Accept-Encoding '';

proxy\_set\_header Host $host;

proxy\_set\_header Referer $http\_referer;

proxy\_set\_header Cookie $http\_cookie;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_headers\_hash\_max\_size 51200;

proxy\_headers\_hash\_bucket\_size 6400;

proxy\_pass http://nginx/;

}

}

8、配置consul-template nginx启动脚本 /etc/init.d/nginx\_ctmpl

#!/bin/bash

# chkconfig: 2345 10 90

# description: Start and Stop redis

PATH=/usr/local/bin:/sbin:/usr/bin:/bin

EXEC=/usr/bin/consul-template

CONF="/opt/consul/conf/nginx\_ctmpl.json"

case "$1" in

start)

PID=$(ps -ef | grep -v grep | grep "$EXEC -config $CONF" | awk '{print $2}')

if [ -n "$PID" ]

then

echo "haproxy\_ctmpl is running..."

else

echo "Starting haproxy\_ctmpl server..."

$EXEC -config $CONF > /tmp/nginx\_ctmpl.out 2>&1 &

fi

;;

stop)

PID=$(ps -ef | grep -v grep | grep "$EXEC -config $CONF" | awk '{print $2}')

if [ -n "$PID" ]

then

echo "Stopping..."

kill -9 $PID

sleep 2

else

echo "haproxy\_ctmpl exists, process is not running."

fi

;;

restart|force-reload)

${0} stop

${0} start

;;

\*)

echo "Usage: /etc/init.d/tmpl {start|stop|restart|force-reload}" >&2

exit 1

esac

9、设置开机启动

chmod +x /etc/init.d/nginx\_ctmpl

chkconfig nginx\_ctmpl on

service nginx\_ctmpl start

七、测试是否自动发现

docker run -ti -d -p :80 eboraas/apache-php

docker run -d -ti -p :80 nginx

1、consul web

http://192.168.1.66:8500/ui/#/dc1/services

2、shipyard web

http://192.168.1.23:8080

账号admin密码 shipyard

3、haproxy web

http://192.168.1.14:8888/admin?stats

账号：admin 密码admin

###### 服务器脚本简述

脚本统一存放在/scripts/，运行脚本命令 /bin/bash {*script\_file*}



listen\_server.sh 实时监听17玩服务后端运行状态，后台运行

sh /scripts/listen\_server.sh &

server-options\_17wan.sh 17玩服务后端管理（升级、重启）

log\_cut.sh 17玩服务后端日志切割脚本

设置定时任务，每天运行一次

supervisord\_crond.sh 17玩服务后端报警脚本

设置定时任务，每次上午8点运行一次，推送邮件到指定邮箱

adduser.sh 添加用户，并发送邮件

mysqlbak.sh 数据库备份脚本

设置定时任务，每天23点59运行一次备份

telegram.sh php接口监听telegram

php定时脚本

dailyDataAdd.sh

updatechannelkeep.sh

updateYesterdayData.sh

updateRealtimeChannelData.sh

Telegram.sh

###### 服务器定时任务

香港主机定时任务

59 23 \* \* \* /bin/bash /scripts/mysql\_bakup.sh

00 04 \* \* \* /usr/bin/find /tmp/sessions -type f -atime +3 -delete

\*/10 \* \* \* \* /bin/bash /scripts/telegram.sh > /dev/null 2>&1

解释：

1、每天23点59分运行mysql备份脚本

2、每天4点清除1天之前的php session

3、每十分钟运行一个telegram.sh脚本

大陆服务器定时任务

59 23 \* \* \* /bin/bash /script/mysqlbak.sh

00 04 \* \* \* /usr/bin/find /tmp/sessions -type f -atime +1 -delete

01 04 \* \* \* /usr/bin/find /data/niuniu/image/log/ -type f -atime +30 -delete

00 00 \* \* \* /bin/bash /script/dailyDataAdd.sh > /dev/null 2>&1

00 11 \* \* \* /bin/bash /script/updateYesterdayData.sh > /dev/null 2>&1

59 23 \* \* \* /bin/bash /script/log\_cut.sh > /dev/null 2>&1

00 08 \* \* \* /bin/bash /script/supervisord\_crond.sh /dev/null 2>&1

解释：

1. 每天23点59分备份mysql数据库
2. 每天凌晨四点清除1天之前的php session
3. 每天4点1分清除17玩直播服务器30天之前的日志
4. 每天0点运行17玩直播后台php接口脚本
5. 每天0点11分运行17玩直播后台php接口脚本
6. 每天23点59分进行17玩直播服务器状态日志切割
7. 每天8点定时发送前一天服务器奔溃日志邮件