一、简要说明：

- 依据**ip**从小到大的顺序，各个节点分别为**comput01 comput02 comput03**
- 三个服务器`adm`用户的密码为 **123654789**
- 代码位于每个服务器的`tmp`目录下：
  - `host_list` 记录各节点其private ip 和 对应的 hostname

```
172.17.0.14 comput01
172.17.0.15 comput02
172.17.0.16 comput03
```

  - `nopasswd.sh` 用于免密登录，需要在每个服务器都运行一次(对于adm账户而言)。

```
# Usage: nopasswd.sh hosts_file username
adm@comput01: cd /tmp && ./nopasswd.sh host_list adm
adm@comput02: cd ~ && ./nopasswd.sh host_list adm
adm@comput03: cd ~ && ./nopasswd.sh host_list adm
```

  - `lay_nfsnis` 修改主机名，配置hosts 并配置 NFS NIS 服务。

```
# Usage: lay_nfsnis server hosts_file server_dir
#        lay_nfsnis client hosts_file server_dir local_dir
root@comput01: cd /tmp && ./lay_nfsnis server host_list /data
root@comput02: cd /tmp && ./lay_nfsnis client host_list /data /data
root@comput03: cd /tmp && ./lay_nfsnis client host_list /data /data
```

- `adduser` 用于创建用户和同步用户信息。和 useradd 一样的使用方法。

```
# 新建用户并更新信息
root@comput01: cd /tmp && ./adduser -g group_nam -e 2020-01-01 user_name
# 删除用户后(修改用户信息的操作)使用 adduser更新信息
root@comput01: cd /tmp && userdel -r username && ./adduser
# 客户端的用户可以使用 yppasswd 命令修改密码并自动更新
```

- `show_usage.sh` 用户统计用户磁盘占用。

```
#Usage: show_usage.sh '/data /tmp /log'
#Path should be absolute path.
root@comput01: cd /tmp && ./show_usage.sh '/data /tmp'
```

二、详细说明：

---

1. 在三台服务器上各创建一个名为adm的账户（设置密码），禁用root远程登陆，修改主机名分别为comput01, comput02, comput03，并设置三台服务器之间免密登陆。

1. 分别登录三台服务器 `yum -y update` 更新软件

    1. 记录各节点**private ip,** 按照以下格式保存到**comput01** 的 `/tmp/host_list` 文件中, 并拷贝到各个节点的**/tmp**目录下。

```
172.17.0.13 comput01
172.17.0.14 comput02
172.17.0.15 comput03
```

2. 创建**adm**用户

    1. 在每台服务器中以**root**权限创建用户**adm,** 家目录设置为**/home/adm(**希望在不同节点上的**adm**用户是独立的**)**

```
# 已经存在adm用户
userdel adm
useradd -d /home/adm -m adm
# 修改密码为 123654789
passwd adm
# 拷贝一份host_list文件到adm 家目录下并转移所有权
cp /tmp/host_list /home/adm
chown adm.adm /home/adm/host_list
```

2. 在本地使用**adm**登录三台服务器，并使用**su root**切换到**root** 确保后续出错可以修正。

3. 禁用**root**远程登录

1. 修改**sshd**配置文件

```
vim /etc/ssh/sshd_config
# 修改(存在注释先取消注释) `PermitRootLogin yes`
为 `PermitRootLogin no`
```

2. 重启**sshd** 服务 `systemctl restart sshd`， `exit` 退出

3. 使用**root**登录三台机器会出现错误： **Permission denied,** 表明上一步成功。

4. 设置三台服务器**adm**用户之间免密登录

1. `yum install openssh-clients` 按照**openssh** 客户端

2. 登录**adm@comput01(**实际为**ip),** 在家目录下执行：

```
bash nopasswd.sh
# >> Usage: nopasswd.sh hosts_file username
bash nopasswd.sh host_list adm
# 之后要按照提示输入comput02 comput03 的密码
```

3. 在当前**shell**中执行`ssh adm@ip of comput02` 执行与第一步相同的操作**(**无需再次输入密码**)**。

4. 在当前**shell**中执行`ssh adm@ip of comput03` 执行第一步相同的操作**(**无需再次输入密码**)**。

5. 在各主机中分别以`ssh adm@ip if node`的形式连接对方，无需输入密码，表明成功。

6. `nopasswd.sh`脚本

```
# !/bin/bash
if [ $# -ne 2 ]; then
    echo "Usage: $(basename $0) hosts_file username"
    exit
fi
hosts=$1
username=$2
while IFS=' ' read -r addr name
do
    # 如果是本机则先生成密钥
```

```
    if [ $addr = `hostname -I` ]; then
        if [ ! -f ~/.ssh/id_rsa.pub ]; then
        cat /dev/zero | ssh-keygen -q -N ""
        fi
        cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
        chmod 600 ~/.ssh/authorized_keys
    # 非本机则ssh远程连接，把本机pub密钥添加到远程主机的~/.ssh/authorized_keys中
    # 并且把本脚本和host_list  scp 传送到远程主机中
    else
        cat ~/.ssh/id_rsa.pub | ssh $username@$addr \
        "cat > ~/tmp_file; if [ ! -d  !/.ssh ]; then cat /dev/zero | ssh-keygen -q -N
''; fi;\
        cat tmp_file >> ~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys;\
        rm -f ~/tmp_file 2>/dev/null"
        scp -p $(basename $0) $username@${addr}:~/
        scp -p $1 $username@${addr}:~/
    fi
 done < "$hosts"
```

5. 更改主机名 切换到**root**用户

    1. 修改主机名的步骤放在下面配置**NFS/NIS**的时候一起设置，这里仅介绍方法。

    2. 不在该步骤中修改本机名，如果需要修改主机名的话，执行**(**以**comput01**为例**)**：

```
# 修改 /etc/hostname
echo 'comput01' > /etc/hostname
hostname comput01
```

    3. 同理不在该步骤中修改**hosts**文件，如果需要修改的话：执行**(**以**comput01**为例**)**：

```
su root
cat host_list >> /etc/hosts
```

---

2. 在第一台服务器中创建**"/data/"**目录，并将其挂载到另外两台服务器的**/data/**位置。（可以使用**NFS**）

3. 在第一台服务器中，编写**Bash**脚本用于创建用户，要求输入的参数有：用户名、用户所属组、账户有效期，用户的根目录为**"/data/**用户名**/"**。并在执行后自动将用户同步到另外两台服务器中。并使用编写的**Bash**创建四个用户，分别隶属两个组（可以使用**NIS**）

4. 在上述创建的两个组中分别删除两个用户，并在三个服务器中同步（如果有配置**NIS**，请使用**NIS**实现。如果没有配置**NIS**，请编写相应的脚本实现）。

1. 配置**NFS, NIS** 以下步骤均为在`root`用户下进行

1. 在**comput01** 中执行：

```
/tmp/lay_nfsnis
# Usage: lay_nfsnis server hosts_file server_dir
#        lay_nfsnis client hosts_file server_dir local_dir
# hosts_file format:
# 192.167.0.3 hostname1
# 192.167.0.5 hostname2
# 192.167.0.58 hostname3
# The first line represent the NFS/NIS server.
# This node's ip address must correspond to a certain line in hosts_file
/tmp/lay_nfsnis server /tmp/host_list /data
```

2. 在**comput02 comput03** 中执行：

```
/tmp/lay_nfsnis client /tmp/host_list /data /data
```

3. 在客户端中`df -h` 即可看到目录挂在成功：

```
# >> comput01:/data                       10G  1.8G  8.3G  18% /data
```

2. 编写脚本 `adduser`，添加用户并同步

1. 在**comput01** 中执行： `root`用户下进行

```
/tmp/adduser -g group1 -e 2020-01-01 -p 123456 user1
# >> Updated user infos.
#    Add user user1 success: user1:x:1033:1033::/data/user1:/bin/bash
/tmp/adduser -g group1 -e 2020-01-02 -p 123456 user2
/tmp/adduser -g group2 -e 2022-02-01 -p 123456 user3
/tmo/adduser -g group2 -e 2022-02-01 -p 123456 user4
# 如果上一步中没有给用户设置密码，还需要给用户设置密码
# 在comput01中修改了用户信息，只需要执行`/tmp/adduser` 一次就行
# /tmp/adduser 还有更新ypserv数据库的功能
/tmp/adduser
# >> Updated user infos.
```

2. 检查是否添加成功 以及客户端是否更新了用户信息

```
# 1. 服务端 comput01 中
tail -n 4 /etc/passwd
"user1:x:1033:1033::/data/user1:/bin/bash
 user2:x:1034:1033::/data/user2:/bin/bash
 user3:x:1035:1034::/data/user3:/bin/bash
 user4:x:1036:1034::/data/user4:/bin/bash"
tail -n 2 /etc/group
"group1:x:1033:
 group2:x:1034:"
# 2. 在客户端 comput02 comput03 中
id user1
```

```
"uid=1033(user1) gid=1033(group1) groups=1033(group1)"
yptest | tail -n 5
# 检查用户修改密码功能
su user1
yppasswd
# 检查目录是否同步
ll /data*
"
drwx------ 2 user1 group1 59 Jul  4 06:43 user1
drwx------ 2 user2 group1 59 Jul  4 06:52 user2
drwx------ 2 user3 group2 59 Jul  4 06:54 user3
drwx------ 2 user4 group2 59 Jul  4 06:54 user4"
# 由于/etc/login.defs 中的 UMASK 默认为077 所以这里目录的权限为700
# 如果要使得不同用户对不同用户的家目录有可读权限，只需要更改下UNMASK就行
```

3. 在两个组中删除用户并同步 `root用户下进行`

    1. **comput01** 中删除用户

```
# -r 选项表明删除用户的家目录和邮件目录
userdel -r user1
userdel -r user2
userdel -r user3
userdel -r user4
groupdel group1
groupdel group2
/tmp/adduser
# >> Updated user infos.
```

    2. 查看客户端中是否同步

```
id user1
# >> id: user1: no such user
ll /data*
# >> total 0
```

---

5. 编写脚本统计该服务器上所有用户的磁盘使用情况，并将结果输出成三列的文件（第一列为用户名，第二列为用户组，第三列为磁盘使用量，单位为GB）

```
# 生成test user1 user2 和 test 文件
/tmp/adduser user1
/tmp/adduser user2
cd /data/user1 && dd if=/dev/zero of=50M.file bs=1M count=50
cd /data/user2 && dd if=/dev/zero of=100M.file bs=1M count=100
cd /tmp && dd if=/dev/zero of=70M.file bs=1M count=70 && chown user1.user1 70M.file
cd /tmp && dd if=/dev/zero of=50M.file bs=1M count=50 && chown user2.user2 50M.file
# 调用脚本  该脚本支持多目录，单用户多目录统计
bash /tmp/show_usage.sh "/data /tmp"
```

```
# >>
# total   total   0.265GB
# user2   user2   0.146GB                ->  100 + 50 / 1024
# user1   user1   0.117GB                ->  70 + 50 / 1024
# root    root    0.002GB
# adm     adm     0.000GB
# mongod  mongod  0.000GB

# 删除文件，用户
rm -f /data/user1/50M.file /data/user2/100M.file /tmp/70M.file /tmp/50M.file
userdel -r user1
userdel -r user2
/tmp/adduser
```

---

6. 根据自己的兴趣尽可能完整地配置一个高可用网络服务器（如**LAMP、Django、Flask**等），调试网络请在主机（**218.199.68.107**）中打开**firefox**调试。

1. **comput01**使用`nginx`负载均衡, 反向代理，分发请求到**2**个后端服务器**(comput01, comput02),** 即可以分担请求负载，也可以实现故障自动转移**(**反向代理层到应用服务层的高可用**)**。
2. 在**comput02, comput03**分别使用`nginx`通过`unix socket`反向代理到多个`aiohttp`异步**web**服务器。
3. 数据库采用双主同步，利用`keepalived`存活探测，以`VIP`的形式提供`mysql`数据库服务。由于没有多余的主机，只能在**comput02**和**comput03** 上搭建`mysql`双主服务**(**正常情况下应该位于与**web**服务器不同的主机上**)**。这里使得数据库也具有高可用性。
4. 使用`supervisord`监控多个`aiohttp`服务器进程和`mysqld`数据库进程。
5. 所有**web**服务的配置文件和代码都位于**/home/www**目录下，且相互独立。三台服务器都创建了**www**用户**/**组，专门用于存放**web**服务的配置文件，在**comput02**和**comput03**中还存放了服务器**python**代码和网页代码。
6. 概要 **1** 表示**comput01 2,3**表示**comput02,comput03**

```
           mysql(2)<----------------------
        -> nginx(2) -> 5 * web app ->    |
nginx(1)                                VIP(172.17.1.101)
        -> nginx(3) -> 5 * web app ->    |
           mysql(3)<----------------------
```

7. 每台主机中**web**服务路径

```
-/home/www
    -awesome     app1的网页和运行代码
        -logs    app1的nginx log 目录
        -socks   app1的unix soceket 目录  (如果使用unix scoket 的话)
        -others
    -nginx
        - awesome.conf     app1 nginx config
    -supervisor
        - awsome.conf      app1 supervisord config
```

> aiohttp 是基于asyncio 实现的HTTP框架 web服务器代码和网页代码均来自
> `https://github.com/michaelliao/awesome-python3-webapp/tree/day-16`

步骤：

注意在下面的过程中由于要配置各种网络服务，需要把防火墙关闭， 无特殊说明均以**root**用户执行

```
yum install iptables-service
systemctl stop iptables
systemctl disable iptables
```

1. 安装必备软件 **ngxin, python3, supervisord。 (comput01, comput02, comput03)**

    1. 安装`nginx`

```
cd /tmp
yum install pcre pcre-devel zlib zlib-devel gcc wget
wget http://nginx.org/download/nginx-1.17.1.tar.gz && tar zxvf nginx-1.17.1.tar.gz
mkdir /usr/local/nginx
cd nginx-1.17.1 && ./configure --prefix=/usr/local/nginx
make && make install
```

    2. 安装`python3` (如果安装了`NetworkManager` 需要将其关闭)

```
yum install libffi-devel openssl openssl-devel
wget https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tgz && tar zxvf Python-
3.7.3.tgz
mkdir /usr/local/python
cd Python-3.7.3 && ./configure --with-ssl --prefix=/usr/local/python
make && make install
pip3 install aiohttp jinja2 motor aiomysql uvloop
# 之后再通过修改 ~/.vimrc 将/usr/local/python/bin 添加到环境变量PATH中
```

    3. 安装 `supervisor`, 并生成初始配置文件

```
pip install supervisor
# 生成初始配置文件
echo_supervisord_conf > /etc/supervisord.conf
```

2. 安装`mysql`数据库，开启双主复制服务，配置`keepalived`服务。 **(comput02, comput03)**

    1. 在**comput02, comput03** 中安装`mysql` 软件，并初始化，设置**root**密码

```
cd ~
wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
rpm -ivh mysql-community-release-el7-5.noarch.rpm
yum update
yum install mysql-server
# 注意初始化需要先切换到mysql用户下进行
su mysql
mysqld --initialize
exit
# 开启自启动
systemctl enable mysqld
systemctl start mysqld
# 修改mysql root密码
mysqladmin -u root password "server1324"
# 登录查看Mysql服务是否开启，以及密码是否正确
# 注意在执行命令的时候要想免验证登录，必须要  把-p和密码连写在一起
mysql -uroot -pserver1324
```

    2. 开启双主复制。

        1. 开启二进制日志服务，用于同步的服务器进行读取。

```
vim /etc/my.cnf
# 修改[mysqld] 区域的配置:
"[mysqld]
server-id=1
log-bin=mysql-bin
log-slave-updates=true"
# 注意上面的server id 两台主机要不一样，本次设置为 comput02: 1 comput03: 2
# 重启mysql服务
systemctl restart mysqld
```

        2. 创建用于同步的`mysql`复制用户。 一方必须要能够通过`复制用户`连接另一方，用于读取二进制日志文件。

```
  # 在两台机器上分别创建 rep_user
  # 标准形式为 grant previleges on db_name.table_name to user_name @ host identified by password;
  # *.*       左边的*表示任何数据库   右边的*表示任何table
  # 这里也可以写成网段的形式 172.14.0.0/255.255.0.0 或者 172.14.%          %也表示任何区域
  # 但是如果写成172.14.0.0/16这种形式，两台主机都连不上(连接时识别为域名，而又没有给该域名设置权限，但是mysql在配置master的时候却可以连接。尚不清楚具体原因。)
  # 这里写成 % 或者 172.14.0.0/255.255.0.0 的形式 以便于测试。
  root@comput02: mysql -u root -pserver1324
  > grant replication slave on *.* to 'rep_user'@'%' identified by '123456'; flush privileges;
  root@comput03: mysql -u root -pserver1324
  > grant replication slave on *.* to 'rep_user'@'%' identified by '123456'; flush privileges;
  # 在另一台机器上登录验证是否可以连接
  root@comput03: mysql -h comput03 -p123456
  # 连接成功
```

3. 双方互相设置为`master`

```
  # 1. 分别登录两台主机，获取master配置所需的信息
  # root@comput02:
  mysql -u root -pserver1324
  > show master status;
  # > | File             | Position |
  # > | mysql-bin.000002 |     7572 |
  # root@comput03:
  mysql -u root -pserver1324
  > show master status;
  # > | File             | Position |
  # > | mysql-bin.000001 |     10748 |
  # 2. 分别在两台主机上，设置对方为master
  # root@comput02:
  mysql> change master to master_user='rep_user',
  > master_host='comput03',
  > master_password='123456',
  > master_port=3306,
  > master_log_file='mysql-bin.000001',
  > master_log_pos=10748;
  start slave;
  # root@comput03:
  mysql> change master to master_user='rep_user',
  > master_host='comput02',
  > master_password='123456',
  > master_port=3306,
  > master_log_file='mysql-bin.000002',
  > master_log_pos=7572;
  start slave;
  # 在量主机中都检查是否开启成功            \G表示格式化输出
  root@comput02: mysql> show slave status\G;
  root@comput03: mysql> show slave status\G;
  # 只有都出现了下面的标示才表示master - slave 成功。
```

```
# > Slave_IO_Running: Yes
# > Slave_SQL_Running: Yes
# 如果未成功 检查是否可以远程登录rep_user 以及 防火墙是否开放

# 最后，在一台主机中登录root 添加用户,数据库,表， 在另外一台主机中旧可以看到 数据同步了。
```

3. 安装`keepalived` 软件并配置 两台主机都得进行配置**(comput02, comput03)**

    1. 安装软件

```
yum install -y popt-devel
cd /usr/local/src
wget https://www.keepalived.org/software/keepalived-2.0.17.tar.gz
tar zxvf keepalived-2.0.17.tar.gz
cd keepalived-2.0.17
./configure --prefix=/
make
make install
```

    2. 创建检测`mysql存活`的脚本。**keepalive** 会自动调用该脚本，当**mysql**被杀掉后，**keepalived**服务关闭，**VIP**自动变到**BACKUP**服务器**(comput03)**的网卡**eth0**上。

```
# 注意该脚本 必须以 #!/bin/bash 开头，不然keepalived 不能执行该脚本
# 注意-p 一定要和密码连在一起，才可以免验证登录。
vim ~/check_mysql_alive.sh
"#!/bin/bash
host=localhost
user=root
passwd=server1324
mysql -h $host -u$user -p$passwd -e "show status;" > /dev/null
if [ $? = 0 ]; then
    echo "$host mysql is alive"
    exit 0
else
    systemctl stop keepalived
    exit 2
fi"
chmod u+x ~/check_mysql_alive.sh
systemctl enbale keepalived
```

    3. 配置`/etc/keepalived/keepalived.conf`。 需要注意的是，由于**VRRP**协议是通过广播的形式来选举**master**, 所以**VIP** 地址一定要设置为`同一网段`内的地址，对于本实验而言，网段为**172.17.0.0/16** ,在这里面随便选一个就行。

```
"vrrp_script check_mysql {                  # 指定检查的脚本
    script "/root/check_mysql_alive.sh"  # 检测Mysql存活的脚本
    interval 2                              # 检查周期
    weight 2
}
vrrp_instance VI_1 {
    state BACKUP    #(主机为MASTER(comput02)，备用机(一个或多个)BACKUP(comput03))
```

```
    interface eth0  #(HA监测网络接口，俩主机的网卡都是eth0)
    virtual_router_id 61 #(主、备机的virtual_router_id必须相同)
    priority 40 #(主、备机取不同的优先级，主机值较大，备份机值较小,值越大优先级越高)
    advert_int 1 #(VRRP Multicast广播周期秒数)
    authentication {
        auth_type PASS #(VRRP认证方式)       两主机必须一致
        auth_pass 1234 #(密码)               两主机必须一致
    }
    track_script {
        check_mysql #(调用nginx进程检测脚本)
    }
    virtual_ipaddress {
            172.17.1.101/16 #(VRRP HA虚拟地址) 必须属于统一网段
    }
}"
```

4. 开启**keepalived** 服务，并检查**VIP**是否可以进行转移，以及局域网内是否可以**ping**通。

    1. 两主机均开启**keepalived** 服务，并添加自启动项

```
systemctl start keepalived
echo "systemctl restart keepalived" >> /etc/rc.d/rc.local
```

    2. 检查**VIP**是否绑定在网卡上。正常的话会绑定在`MASTER`**(comput02)**上

```
# VIP不同通过ifconfig查看，需要使用ip addr，先安装
yum -y install initscripts
# comput02          可以看到 VIP 已经绑定在 comput02 的 eth0上
ip addr
"173: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
inet 172.17.0.15/16 brd 172.17.255.255 scope global eth0
valid_lft forever preferred_lft forever
inet 172.17.1.101/16 scope global secondary eth0
valid_lft forever preferred_lft forever"
# comput03          只有一个private IP
ip addr
"175: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
inet 172.17.0.16/16 brd 172.17.255.255 scope global eth0
valid_lft forever preferred_lft forever"
```

    3. 关闭**comput02** 的 `keepalived`，查看**VIP**是否自动转移到`comput03`上

```
# comput02      可以看到comput02 的 eth0上已经没有VIP了
systemctl stop keepalived
ip addr
"173: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
inet 172.17.0.15/16 brd 172.17.255.255 scope global eth0
valid_lft forever preferred_lft forever"
# comput03     可以看到comput03 的 eth0上已经绑定了VIP
```

```
  "175: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default
  inet 172.17.0.16/16 brd 172.17.255.255 scope global eth0
  valid_lft forever preferred_lft forever
  inet 172.17.1.101/16 scope global secondary eth0
  valid_lft forever preferred_lft forever"
  # comput02 重新开启
  systemctl start keepalived
```

4. 查看**VIP**是否可以在局域网内**ping**通

```
  ssh adm@comput01
  ping 172.17.1.101
  # [root@comput01 ~]# ping 172.17.1.101
  # PING 172.17.1.101 (172.17.1.101) 56(84) bytes of data.
  # 64 bytes from 172.17.1.101: icmp_seq=1 ttl=64 time=0.271 ms
  # 64 bytes from 172.17.1.101: icmp_seq=2 ttl=64 time=0.106 ms
```

5. 现在，只要`comput02`或者`comput03`的`mysql`服务还有一台存活，就可以在局域网内任何地方链接`mysql`服务了(前提是**mysql**端口(默认**3306**)防火墙关闭)。

3. 建立**www**用户/组, 配置**ngixn** 和 **supervisord**。 (**comput01, comput02, comput03**)

1. 创建**www**用户/组

```
# 由于不希望该用户家目录被共享，将家目录创建到/home中
groupadd www
useradd -d /home/www -g www -m
# 修改密码为 server1324
passwd www
# 创建目录分别存放web服务的 nginx 和 supervisor conf文件
mkdir /home/www/nginx
mkdir /home/www/supervisor
chown -R www.www /home/www
```

2. 配置`nginx.conf` 文件，由于可能会有多个**web**服务，所以使用**include**集中管理`/home/www/nginx/`中的`nginx`配置文件

```
# 1. 配置comput02 comput03 的 /homw/www/nginx/awesome.conf
vim /usr/local/nginx/cong/nginx.conf
"user  www;
worker_processes  8;
events {
    worker_connections  1024;
}
http {
        include       mime.types;
        default_type  application/octet-stream;
        sendfile        on;
        keepalive_timeout  65;
```

```
        proxy_connect_timeout 1;
        proxy_read_timeout 1;
        proxy_send_timeout 1;
        proxy_ignore_client_abort on;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        gzip  on;
        log_format main  "$remote_addr-$remote_user [$time_local]
$request $status $body_bytes_sent                                    $http_referer
$http_user_agent                                  $http_x_forwarded_for";

        include /home/www/nginx/*.conf;

 }"
```

3. 配置 **nginx.service**服务(自启动，以及方便管理)

```
vim /usr/lib/systemd/system/nginx.service
"[Unit]
Description=nginx - high performance web server
After=network.target remote-fs.target nss-lookup.target
[Service]
Type=forking
ExecStart=/usr/local/nginx/sbin/nginx
ExecReload=/usr/local/nginx/sbin/nginx -s reload
ExecStop=/usr/local/nginx/sbin -s stop
[Install]
WantedBy=multi-user.target"
# 添加自启动
systemctl enable nginx.service
echo "systemctl restart nginx" >> /etc/rc.d/rc.local
```

4. 配置`supervisord.conf` 文件, 与上面同理，为了不把所有配置文件都写在一个文件中，使用**include**统一管理配置文件。分为两个目录:

- 一般服务的**conf**文件放在 **/etc/supervisor/**下
- **web** 服务的**conf**文件放在 **/home/www/supervisor/** 下

```
vim /etc/supervisord.conf
# 添加
"[include]
files = /etc/supervisor/*.conf /home/www/supervisor/*.conf"
```

5. 配置 **supervisord.service**服务(自启动，以及方便管理)

```
vim /usr/lib/systemd/system/supervisord.service
"[Unit]
Description=Supervisor main
[Service]
Type=forking
```

```
ExecStart=/usr/local/python/bin/supervisord -c /etc/supervisord.conf
ExecStop=/usr/local/python/bin/supervisorctl shutdown
ExecReload=/usr/local/python/bin/supervisorctl reload
KillMode=process
Restart=on-failure
RestartSec=42s
user=root
[Install]
WantedBy=multi-user.target"
# 添加自启动
systemctl enable supervisord.service
supervisorctl shutdown
echo "systemctl restart supervisord" >> /etc/rc.d/rc.local
```

4. 配置**web**应用，配置**nginx** 和 **supervirod** 服务, 仅在 **comput02, comput03**中进行

    1. 下载**web**应用，修改源码，使用`unix socket` 以及 `uvloop`

```
cd /tmp
wget https://github.com/michaelliao/awesome-python3-webapp/archive/day-16.zip
yum install unzip && unzip day-16.zip
mv day-16 awesome
# 其目录结构为:
-awesome
    -www
    -backup
    -dist
    -ios
    -fabfile.py
# 删除除了 www 之外的文件 和目录
cd awesome && rm -rf !(www)
# 将awe 转移到 /home/awesome 目录下
mv awesome /home/www
# www 目录下的 app.py 文件是启动web服务的文件，为了使用unix sockt ，需要修改几行代码
vim /home/www/awesome/app.py
# 1. 修改为 create_unix_server 并传入一个path 命令行参数用以指定unix socket的path
# 2. 使用uvloop 替代默认的loop, 速度可以提高2-4倍
# ... 表示未修改的地方
"
...
import asyncio, uvloop
asyncio.set_event_loop_policy(uvloop.EventLoopPolicy())
...
@asyncio.coroutine
def init(loop, path):
    ...
    srv = yield from loop.create_unix_server(app.make_handler(), path=path)
    ...
import sys
path = sys.argv[1]
...
loop.run_until_complete(init(loop, path))
..."
```

```
# 创建socks 目录 和 log 目录，分别用于存放sock文件和nginx产生的log文件
mkdir /home/www/awesome/socks
mkdir /home/www/awesome/log
```

2. 在`comput01`中使用`nginx`做负载均衡，反向代理到`comput02`,`comput03`

```
# 配置comput01 的 /usr/local/nginx/conf/nginx.conf用于负载均衡
vim /usr/local/nginx/conf/nginx.conf
"server {
    listen      80;
    server_name www.awesome.com;
    access_log /home/www/awesome/log/access_log;
    error_log  /home/www/awesome/log/error_log;

    location / {
        proxy_pass http://awesome_pools;
    }
}

upstream awesome_pools {
    server comput02:80 weight=1;
    server comput03:80 weight=1;
    ip_hash;
}"
```

3. 在`comput02`和`comput03`中使用`nginx`代理多个`unix web server`并且使用`supervisord`管理多个**web server**

    1. 配置`awesome` **web**应用的`nginx.conf`

    ```
    # 主要部分为：    详情请见 /home/www/nginx/awesome.conf
    vim /home/www/nginx/awesome.conf
    " server {
        listen      80;
        server_name www.awesome.com;
        root        /home/www/awesome/www;
        access_log /home/www/awesome/log/access_log;
        error_log  /home/www/awesome/log/error_log;
        # 静态页面走 nginx
        location ~ ^\/static\/.*$ {
            root /home/www/awesome/www;
        }
        # 动态页面走后端的aiohttp
        location / {
            proxy_pass http://awesome;
        }
    }
    upstream awesome {
        server unix:/home/www/awesome/socks/awesome_1.sock fail_timeout=0;
        server unix:/home/www/awesome/socks/awesome_2.sock fail_timeout=0;
        server unix:/home/www/awesome/socks/awesome_3.sock fail_timeout=0;
        server unix:/home/www/awesome/socks/awesome_4.sock fail_timeout=0;
    ```

```
        server unix:/home/www/awesome/socks/awesome_5.sock fail_timeout=0;"
```

2. 配置 `awesome` **web**应用的`supervisord.conf`

```
  # 在 /home/www/supervisor 中配置 supervisord 的conf 文件
  vim /home/www/supervisor/awesome.conf
  "[program:awesome]
  numprocs = 5
  numprocs_start = 1
  process_name = awesome_%(process_num)s
  command=/usr/local/python/bin/python3 /home/www/awesome/www/app.py
/home/www/awesome/socks/awesome_%(process_num)s.sock
  user=www
  autostart=true
  autorestart=true"
  # 注意上面 利用命令行参数的方法指定socks文件路径
```

3. 配置虚拟域名 `www.awesome.com`, 如果有多个**web**应用的话，**nginx**可以通过**http**请求所带的域名信息，而分发请求到指定域名的**web**应用。

```
  # 1. 修改 hosts文件
  root@comput02 echo "172.17.0.15 www.awesome.com" >> /etc/hosts
  root@comput03 echo "172.17.0.16 www.awesome.com" >> /etc/hosts
  # 2. 在nginx 文件中设置 server_name
  # 在上一步中已经设置
```

4. 修改`/home/www`目录权限为 `www.www`, 由于指定了以**www**用户开启**nginx**子进程和**supervisord**, 所有必须得让**www**用户有**www**目录可读写的权限

```
  chown -R www.www /home/www
```

5. 数据库创建和配置数据库信息

```
# 1. 数据库创建 用户，database，tables 以供awesome web应用使用
# 由于设置了双主模式，只要在任意一台机器上执行就行，数据会自动同步
root@comput02: mysql -u root -pserver1324
> create database awesome;
> grant all on awesome.* to 'www'@'172.17.0.0/255.255.0.0' identified by 'www';
> grant all on awesome.* to 'www'@'172.17.0.0/255.255.0.0' identified by 'www';
# 初始化tabel  init.sql文件见文档末
root@comput02: mysql -u root -p <init.sql

# 2. 编辑web app的配置文件
vim /home/www/awesome/www/config_default.py
"configs = {
    'debug': True,
    'db': {
        'host': '127.0.0.1',
        'port': 3306,
        'user': 'www',
        'password': 'www',
```

```
        'db': 'awesome'
    },
    'session': {
        'secret': 'Awesome'
    }
}"
vim /home/www/awesome/www/config_override.py
# 这里设置我们的VIP 地址
"configs = {
    'db': {
        'host': '172.17.1.101'
    }
}
"
```

6. 开启相关服务 在**comput01, comput02, comput03**中均需开启(虽然**supervisord**在**comput01**中无用)

```
supervisorctl shutdown
systemctl start supervisord
systemctl restart nginx
echo "systemctl restart supervisord" >> /etc/rc.d/rc.local
echo "systemctl restart nginx" >> /etc/rc.d/rc.local
# 1. 检查 supervisor 控制的5个web app是否开启  仅在comput02 comput03中有web app
supervisorctl status
# awesome:awesome_1                RUNNING   pid 14085, uptime 0:00:13
# awesome:awesome_2                RUNNING   pid 14086, uptime 0:00:13
# awesome:awesome_3                RUNNING   pid 14087, uptime 0:00:13
# awesome:awesome_4                RUNNING   pid 14088, uptime 0:00:13
# awesome:awesome_5                RUNNING   pid 14089, uptime 0:00:13
# 2. 检查 /home/www/awesome/log 下的 error_log
tail /home/www/awesome/log/error_log
```

7. 测试

```
ssh adm@comput02
curl comput01
curl www.awesome.com
wget comput01
# >
"
--2019-07-04 11:16:06--  http://comput01/
Resolving comput01 (comput01)... 172.17.0.13
Connecting to comput01 (comput01)|172.17.0.13|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4511 (4.4K) [text/html]
Saving to: 'index.html'

100%
[==========================================================================================
===>] 4,511       --.-K/s   in 0s
```

```
2019-07-04 11:16:06 (137 MB/s) - 'index.html' saved [4511/4511]
"
ssh adm@comput02
tail -n 2 /home/www/awesome/log/access_log
# 172.17.0.14 - - [04/Jul/2019:11:20:10 +0000] "GET / HTTP/1.1" 200 4511 "-" "curl/7.29.0"
# 172.17.0.14 - - [04/Jul/2019:11:20:10 +0000] "GET / HTTP/1.1" 200 4511 "-" "Wget/1.14
(linux-gnu)"
```

8. 故障测试

```
# 1. 关闭comput02 comput03 中的任意一台的nginx
root@comput03: systemctl stop nginx
adm@comput01: wget comput01
adm@comput02: wget comput01
adm@comput03: wget comput01
root@comput03: systemctl restart nginx
# 2. 关闭comput02 comput03 中的任意一台的mysqld
root@comput02: systemctl stop mysqld
# 这一步过后需要等待约10几秒。 因为需要重新连接数据库
adm@comput01: wget comput01
adm@comput02: wget comput01
adm@comput03: wget comput01
root@comput02: systemctl restart mysqld
root@comput02: systemctl restart keepalived
# 重启后开始几秒也不能访问
# 均可以访问
```

脚本介绍

下面的脚本均位于comput01:/tmp 目录下

`lay_nfsnis` 脚本介绍

- `lay_nfsnis` 脚本包含了运行失败后自动恢复原始配置文件，不重复添加配置信息，可多次运行，可添加节点等方面的功能。
- 脚本代码详情请见/comput01:tmp目录，以下仅为其大致流程。

1. 在三个服务器中分别修改**hosts**文件, 修改**hostname**, 以**comput01**为例

```
cat /tmp/host_list >> /etc/hosts
echo 'comput01' > /etc/hostname
hostname comput01
# 添加自启动项， 修改文件权限为可执行
echo 'hostname comput01' >> /rc.d/rc.local
chmod u+x /rc.d/rc.local
```

2. 安装必备软件： `vim rpcbind nfs-utils yp-tools ypserv ypbind`

3. 配置**NFS**

1. 服务端 在**comput01**进行

```
  mkdir /data && chmod 777 /data
  # 1. /data是要共享的目录
  # rw 表示开放读写权限, no_root_squash表示root不会被转换为nfsnobody身份
  echo '/data comput01(rw,no_root_squash) comput02(rw,no_root_squash)
comput03(rw,no_root_squash)' > /etc/exports
  # 2. 开启各服务, 顺序
  systemctl restart rpcbind
  systemctl restart nfs
  systemctl restart nfslock
  # 查看各端口是否开启 rpcbind: 111 nfs: 2049
  netstat -tulnp | grep -E '(rpc|nfs)'
  # 3. 挂在共享目录
  exportfs -arv
  showmount -e localhost
  # >> Export list for localhost:
  # >> /data comput03,comput02,comput01
  # 4. 添加自启动项
  echo 'systemctl restart rpcbind' >> /rc.d/rc.local
  echo 'systemctl restart nfs' >> /rc.d/rc.local
  echo 'systemctl restart nfslock' >> /rc.d/rc.local
  echo 'exportfs -arv' >> /rc.d/rc.local
```

2. 客户端 在**comput02 comput03** 进行

```
  # 1. 创建挂载目录,  开启服务
  mkdir /data
  systemctl restart rpcbind
  systemctl restart nfslock
  # 检查是否服务端开启了共享
  showmount -e comput01
  # >> Export list for comput01:
  # >> /data comput03,comput02,comput01
  # 2. 挂载
  mount -t nfs -o nosuid,nodev,rw,bg,soft,rsize=32768,wsize=32768 comput01:/data /data
  # 3. 添加自启动项
  echo 'systemctl restart rpcbind' >> /rc.d/rc.local
  echo 'systemctl restart nfslock' >> /rc.d/rc.local
  echo 'mount -t nfs -o nosuid,nodev,rw,bg,soft,rsize=32768,wsize=32768 comput01:/data
/data' >> /rc.d/rc.local
  # 4. 查看是否挂载成功
  df -h
  # >> comput01:/data                       10G  1.8G  8.3G  18% /data
```

4. 配置**NIS**

1. 服务端 在**comput01** 进行

```
 # 1. 设置NISDOMAINNAME 并且 设置端口(后续可以用防火墙进行控制)
 echo "NISDOMAIN=comput01" >> /etc/sysconfig/network
 echo "YPSERV_ARGS='-P 1011'" >> /etc/sysconfig/network
```

```
echo "YPPASSWDD_ARGS='--PORT 1012'" >> /etc/sysconfig/yppasswdd
nisdomainname comput01
# 2. 设置开放区域 多种设置方式， 本次选用了第一中方式
#     1. 直接设置ip
echo '172.17.0.14:*:*:none' >> /etc/ypserv.conf
echo '172.17.0.15:*:*:none' >> /etc/ypserv.conf
#     2. 对整个网段开放，之后通过防火墙进行控制可访问的客户端
echo '172.17.0.0/255.255.255.0:*:*:none' >> /etc/ypserv.conf
#     3. 对所有地址开放，之后再通过防火墙进行控制
echo '*:*:*:none' >> /etc/ypserv.conf
# 3. 开启服务    再配置NFS的时候已经开启了rpcbind 这一次可以跳过
systemctl restart ypserv
systemctl restart yppasswdd
# 4. 生成数据库(更新用户信息)
make -C /var/yp
# 5. 添加自启动项
echo 'nisdomainname comput01' >> /rc.d/rc.local
echo 'systemctl restart ypserv' >> /rc.d/rc.local
echo 'systemctl restart yppasswdd' >> /rc.d/rc.local
echo 'make -C /var/yp' >> /rc.d/rc.local
```

2. 客户端 在**comput02 comput03** 进行

```
# 1. 设置NISDOMAINNAME 和 YPSERV 端口
echo "NISDOMAIN=comput01" >> /etc/sysconfig/network
echo "YPSERV_ARGS='-p 1011'" >> /etc/sysconfig/network
nisdomainname comput01
# 2. 配置账户信息读取规则
vim /etc/nsswitch.conf
# 将相对应的地方修改为下面的形式:
# 将files 放在前面可以使得先读取本机存在的用户信息
"passwd: files nis sss
 shadow: files nis sss
 group: files nis sss
 hosts: nis files dns myhostname"
# 3. 配置/etc/yp.conf
echo "domain comput01 server comput01" >> /etc/yp.conf
echo "ypserver comput01" >> /etc/yp.conf
# 4. 设置账号登入认证机制
echo "USENIS=yes" > /etc/sysconfig/authconfig
# 5. 设置PAM授权
echo "password   sufficient    pam_unix.so md5shadow nis nullok
try_first_passuse_authtok" >> /etc/pam.d/system-auth
# 6. 开启服务 (配置NFS客户端时已经开启rpcbind，这里跳过)
systemctl restart ypbind
# 7. 测试
rpcinfo -p comput01
yptest
# 8. 添加自启动项
echo 'nisdomainname comput01' >> /rc.d/rc.local
echo 'systemctl restart ypbind' >> /rc.d/rc.local
```

`adduser` 脚本介绍

基本上是将用户输入的参数代理给`useradd`命令，使得该脚本支持`useradd`的所有功能，且有着些许的改动，比如:

1. 默认会给用户创建`/data`下面的家目录。
2. 当用户输入的组名不存在时会自动创建该组。
3. 每次运行脚本最后都会执行`make -C /var/yp` 进行**NIS** 服务的数据库更新的操作。
4. 让`useradd`进行繁琐的参数检验工作。

```
# !/bin/bash
NUM_RE="^[0-9]+$"
DEFAULT_HOME_PATH="/data"
FILE_NAME=$(basename $0)
show_help() {
    tmp_file=$(mktemp help_info.XXXXX)
    useradd --help >> $tmp_file
    IFS=$'\n'
    for line in $(cat $tmp_file | sed "s/useradd/$FILE_NAME/")
    do
        echo $line
    done
    rm -f $tmp_file 2> /dev/null
    exit
}
relay_arg() {
    if [ $# -eq 0 ]; then show_help; fi
    while [ -n "$1" ]; do
        case "$1" in
            --help|-h) show_help ;;
            -g) group=$2; shift  ;;
            *)                   ;;
        esac
        shift
    done
}
# 当用户组不存在时，创建用户组
check_grp() {
    if [ -z "$group" ]; then return 0; fi
    if ! [[ "$group" =~ $NUM_RE ]]; then
        record=$(cat /etc/group | cut -d ":" -f 1 | grep -w $group)
        if [ -z "$record" ]; then
            groupadd $group
            newgrp=TRUE
        fi
    else
        record=$(cat /etc/group | cut -d ":" -f 3 | grep -w $group)
        if [ -z "$record" ]; then
            echo "GUID not exit, please choose \
            a group name of an existed GUID."
            exit
        fi
```

```
    fi
}

make -C /var/yp 1>/dev/null
echo -e "Updated user infos.\n"
relay_arg $*
check_grp
useradd -D -b $DEFAULT_HOME_PATH
if `echo "$*" | grep -q '\-m'`; then
    useradd $*
else
    useradd -m $*
fi
# 当创建用户失败时， 删除可能创建的组
if [ $? != 0 ]; then
    if [ "$newgrp" = "TRUE" ]; then
        groupdel $group
    fi
    echo "Add user $user_name failure!"
else
    username=`cat /etc/passwd | tail -n 1 | cut -d ":" -f 1`
    make -C /var/yp 1>/dev/null
    echo "Add user $username success: $(cat /etc/passwd| tail -n 1)"
fi
```

`show_usage.sh` 脚本介绍

1. 使用`du -sc`统计目录/文件的磁盘占用。
2. 使用`stat`获取目录/文件的所有者。
3. 使用字典储存不同用户的磁盘使用量。

```
# !/bin/bash
show_help() {
    echo "Usage: $(basename $0) '/data /tmp /log' "
    echo "Path should be absolute path."
    exit
}
DATE=$(date '+%y-%m-%d')
declare -A usage_dict
usage_dict[total]=0
dirs=$1
# Check if path exits
if [ $# -ne 1 ]; then show_help ;fi
for dir in $dirs
do
    if [ ! -d $dir ]; then
        echo "Error $dir not exit."
        show_help
    fi
done
```

```
# Use du to count usage, Use dict to store usage for each user
for dir in $dirs
do
    IFS=$'\n'
    for line in `du -sc ${dir}/* 2>/dev/null`
    do
        size=$(echo $line | cut -f 1)
        path=$(echo $line | cut -f 2)
        if [ ${#path} -gt 30 ]; then continue; fi
        if [ $path = total ]; then
            oldsize=${usage_dict[total]}
            usage_dict[total]=$[$oldsize + $size]
            continue
        fi
        user=`stat -c "%U:%G" $path `
        if [ -z ${usage_dict[$user]} ]; then
            usage_dict[$user]=$size
        else
            oldsize=${usage_dict[$user]}
            usage_dict[$user]=$[$oldsize + $size]
        fi
    done
done
# Convet kb to GB and display the usage info
tmp_file=$(mktemp usage.XXXX)
IFS=' '
for key in $(echo ${!usage_dict[*]})
do
    uname=$(echo $key | cut -d ":" -f 1)
    gname=$(echo $key | cut -d ":" -f 1)
    size=${usage_dict[$key]}
    size=$(awk "BEGIN{printf \"%.3f\" ,$size/1048576 }")
    echo -e "${uname}\t${gname}\t${size}GB" >> $tmp_file
done
echo "$DATE Usage summary for $dirs:"
sort -t "    " -k3.1rn $tmp_file
rm -f $tmp_file 2>/dev/null
```

---

init.sql 文件

```
# init.sql 文件
use awesome;
create table users (
    `id` varchar(50) not null,
    `email` varchar(50) not null,
    `passwd` varchar(50) not null,
    `admin` bool not null,
    `name` varchar(50) not null,
    `image` varchar(500) not null,
    `created_at` real not null,
    unique key `idx_email` (`email`),
```

```
    key `idx_created_at` (`created_at`),
    primary key (`id`)
) engine=innodb default charset=utf8;

create table blogs (
    `id` varchar(50) not null,
    `user_id` varchar(50) not null,
    `user_name` varchar(50) not null,
    `user_image` varchar(500) not null,
    `name` varchar(50) not null,
    `summary` varchar(200) not null,
    `content` mediumtext not null,
    `created_at` real not null,
    key `idx_created_at` (`created_at`),
    primary key (`id`)
) engine=innodb default charset=utf8;

create table comments (
    `id` varchar(50) not null,
    `blog_id` varchar(50) not null,
    `user_id` varchar(50) not null,
    `user_name` varchar(50) not null,
    `user_image` varchar(500) not null,
    `content` mediumtext not null,
    `created_at` real not null,
    key `idx_created_at` (`created_at`),
    primary key (`id`)
) engine=innodb default charset=utf8;
```

补充

防火墙设置 本次实验并未做任何防火墙限制，全部设置为开放。

1. 首先设置禁止**INPUT,** 开放**OUTPUT** 和 **FORWARD**的**Policy**。之后逐步开放特定的端口。
2. 开放**lo**接口的数据包。
3. 同意已经接受连接的所有数据包。
4. 同意**vrrp**协议的包传入，用于**keepalived**使用。
5. 开启**ipv4 ip_forward**。
6. **NFS**和**NIS**都使用到了**rpcbind, rpcbind**默认使用的端口是**111**，需要放行。
7. 由于需要使用**NFS**系统，所以需要开放特定的端口供**NFS**使用。**NFS**主程序默认的端口是**2049**，但是其还会开启一系列的端口，而默认情况下**NFS**各种服务的端口是随机生成的，所以需要先固定**NFS**各种服务使用的端口，之后放行这些端口的**INPUT**。
8. **NIS**服务的**YPSERV**我们之前指定了端口**1011**，**YPPASSWDD**指定的端口是**1012**，所以需要开放这两个端口供**YPBIND**使用。
9. 作为**mysql**服务器的主机，要对**web**应用所在的主机以及数据库所在的主机开放**3306**端口。

10. 开放一些必须的服务，比如**sshd**，还有开放特定的端口供**nginx**和**web**服务器之间进行反向代理。开放前端**nginx**的**80**端口。

为**NFS**指定特定的端口。**(**根据启动的服务所需开放的端口数量不一样，本次实验只开启了**nfs**和**ngslock**，如果还开启其他相关的**nfs**服务，还需指定开放更多端口**)**

```
# 修改/etc/sysconfig/nfs
vim /etc/sysconfig/nfs
# RQUOTAD_PORT=1001
# LOCKD_TCPPORT=1002
# LOCKD_UDPPORT=1003
# MOUNTD_PORT=1004
```

mongodb 安装

```
# 安装官网的流程安装
# https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/
echo "systemctl restart mongd" >> /etc/rc.d/rc.local
systemctl enable mongod
systemctl restart mongod
mongo
> use admin
> db.createUser({user:'admin', pwd:'server1324', roles:[{role:'root', db:'admin'}]})
# 开启验证
vim /etc/mongo.conf
"security:
    authrization: enabled"
systmctl restart mongod
```

keepalived 开启log

最初配置完**keepalived** 后，关闭**mysqld**, 发现**keepalived**并没有被关闭**(**导致**VIP**没有飘逸**)**，于是检查了**check_myhsql_alive.sh** 是否有问题，发现其可以正确的关闭**keepalived**服务后; 所以猜测是**keepalived** 并没有调用**check_mysql_alive.sh**。于是想要去检查日志，但日志文件没开启，所以需要先开启日志模式。

```
# 1. 安装rsyslog 服务
yum install rsyslog
systemctl enable rsyslog
systemctl start rsyslog

# 2. 开启keepalived 日志服务
vim /etc/sysconfig/keepalived
# 修改为开启日志系统
# KEEPALIVED_OPTIONS='-D -d -S 0'

# 3. 重启服务
systemctl restart rsyslog
systemctl restart keepalived

# 5. 查看日志错误信息
# Error exec-ing command '/root/check_mysql_alive.sh', error 8: Exec format error
```

```
# 说明格式出错，所以文件起始添加#!/bin/bash
```