

Ubuntu 22.04 安装CUPS共享打印服务

1.安装cups及配置

```
apt install cups

vim /etc/cups/cupsd.conf
> 修改18行 Listen为0.0.0.0:631
> 修改22行 Browsing Off为Browsing On
> 在29行下面增加 ServerAlias *
> 在所有的Order Allow,Deny行下方增加Allow all
```

2.编译hplip准备工作

```
sudo apt install python-is-python3 libavahi-core-dev
sudo apt install hplip hplip-data printer-driver-hpcups -y
sudo apt-get install --assume-yes python3-pip
sudo pip3 install --upgrade pip
sudo apt-get install --assume-yes liblptonica-dev
sudo apt-get install --assume-yes tesseract-ocr tesseract-ocr-all
sudo apt-get install --assume-yes libtesseract-dev
sudo -H pip3 install tesseract
sudo -H pip3 install opencv-python
sudo -H pip3 install PyPDF2
sudo -H pip3 install imutil
sudo -H pip3 install scikit-image
sudo -H pip3 install scipy
sudo apt install ocrmypdf
```

3.在非root用户下执行 ./hplip-3.22.04.run

4.执行安装 hp-plugin:

```
hp-plugin -i
```

5.运行lsusb查看 系统检测到的USB设备

6.将linux用户添加到lpadmin用户组中

```
sudo usermod -aG lpadmin $USER
```

7.设置开启自启cups

```
sudo systemctl enable cups
```

Ubuntu 24.04 安装FileManager服务

1. 禁用无用的系统服务

```
systemctl disable --now unattended-upgrades
sudo apt-get update -y
sudo apt remove cloud-init -y
sudo apt install vim tree net-tools iputils-ping software-properties-common -y
```

2. 同步时间

```
timedatectl set-timezone Asia/Shanghai
apt install ntpdate -y
ntpdate ntp.aliyun.com
hwclock -w
设置24小时时间
vim /etc/default/locale
添加
LANG=en_US.UTF-8
LC_TIME=en_DK.UTF-8
```

3. 删除snap服务

```
snap list
snap remove lxd
snap remove core20
snap remove snapd
systemctl stop snapd
apt remove snapd -y
```

4. 开启root登录,方便传代码

```
修改/etc/ssh/ssd_config
找到PermitRootLogin行, 修改为 PermitRootLogin yes
systemctl daemon-reload
设置root的密码: passwd root
```

5. 修改网卡等待时间:

```
vim /etc/systemd/system/network-online.target.wants/systemd-networkd-wait-online.service
在Service段中添加 TimeoutStartSec=2sec
systemctl daemon-reload
```

6.修改pip为国内源

1. 在/home/linux下创建.pip文件夹与pip.conf文件
2. `mkdir -p /home/linux/.pip`
3. `vim /home/linux/.pip/pip.conf`

```
[global]
index-url = https://mirrors.aliyun.com/pypi/simple/
[install]
trusted-host=mirrors.aliyun.com
```
4. `mkdir -p /root/.pip`
5. `cp /home/linux/.pip/pip.conf /root/.pip/pip.conf`

7.安装zerotier

```
curl -s https://install.zerotier.com | sudo bash

systemctl enable zerotier-one
zerotier-cli join 93afae5963a766fc
mkdir /var/lib/zerotier-one/mons.d
wget https://rowsea.com/000000b5b91d9c08.moon
mv 000000b5b91d9c08.moon /var/lib/zerotier-one/mons.d/
zerotier-cli orbit b5b91d9c08 b5b91d9c08
systemctl restart zerotier-one
```

8.安装软raid

1. `apt install mdadm quota`
2. 检查磁盘状态,确认哪些磁盘未被使用,例如 `/dev/sdb /dev/sdc /dev/sdd`

```
lsblk
fdisk -l
```
3. 清除磁盘旧配置

```
sudo mdadm --zero-superblock /dev/sd[b-d]
```
4. 创建raid阵列

```
sudo mdadm --create --verbose /dev/md0 --level=10 --raid-devices=4 /dev/sd[b-e]
```
5. 创建文件系统并挂载

```
sudo mkfs.ext4 -O project,quota /dev/md0
sudo mkdir -p /mnt/raid
sudo mount /dev/md0 /mnt/raid
```
6. 持久化配置

```
sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf
sudo update-initramfs -u
```
7. 添加到 `/etc/fstab`:

```
echo '/dev/md0 /mnt/raid ext4 defaults,nofail,discard,prjquota 0 0' | sudo tee -a /etc/fstab
```
8. 开启磁盘路径配额

```
quotaon -P /mnt/raid
```
9. 查看是否开启磁盘路径配额

```
quotaon -Ppv /mnt/raid
```
10. 监控阵列同步状态

```
watch cat /proc/mdstat
```
11. 常见问题与排查方法

阵列创建失败或卡住？原因是 某些磁盘可能已有旧RAID签名，使用 `-zero-superblock` 清除所有参与盘的RAID信息

重启后阵列未挂载？检测`/etc/mdadm/mdadm.conf`和`/etc/fstab`是否正确写入，是否RAID设备名变动

磁盘掉线或降级模式？使用`mdadm --detail /dev/md0`查看状态

替换损坏盘后重新添加：`mdadm --add /dev/md0 /dev/sdx`

性能不达预期：

检测磁盘是否为SMR类型（不推荐用于RAID）

检测是否开启写缓存（使用`hdparm -w`）

优化文件系统挂载参数，如`noatime,nobarrier`

9.重装系统后恢复RAID1硬盘阵列

1. 查看阵列是否存在

```
cat /proc/mdstat
```

2. 导入

```
mdadm --assemble --scan
```

3. 恢复配置并保存

```
mdadm --detail --scan >> /etc/mdadm/mdadm.conf
update-initramfs -u
```

4. 删除`/etc/mdadm/mdadm.conf`类似下面这个后重启

```
ARRAY /dev/md/0
```

10.安装smartctl

```
apt install g++ gcc make -y
```

```
tar -zxvf https://rowsea.com/smartmontools-7.5.tar.gz
```

```
cd smartmontools-7.5
```

```
./configure --disable-dependency-tracking
```

```
make && make install
```

```
/usr/local/sbin/smartctl -a /dev/sda
```

10.双网卡指定网段走指定网卡网关方法

```
route add default gw 10.129.168.1 # 添加默认网关10.129.168.1
```

```
route add -net 10.129.168.0/21 gw 10.129.168.1 eth0 # 10.129.168.0网段走eth0网卡
```

```
# 删除路由 route del -net 10.129.168.0/21 gw 10.129.168.1
```

win10添加双网卡路由

```
route print #查看网卡编号(第一列)
```

```
route add 10.129.168.0 mask 255.255.248.0 10.129.168.1 if <网卡编号> -p
```

```
#删除路由 route delete 10.129.168.0
```

11.安装samba

```
apt install samba
```

```
systemctl enable smbd
```

```
mkdir -p /etc/samba/users/
```

`/etc/samba/smb.conf`配置文件修改为以下内容：

```
[global]
```

```

workgroup = WORKGROUP
server string = %h server (Samba, Ubuntu)
log file = /var/log/samba/log.%m
max log size = 1000
logging = file
security = user
;map to guest = bad user
#netbios name = Share Server
max connections = 1000
dns proxy = no
min protocol = NT1
# 隐藏特殊文件,如socket, 设备文件, FIFO等
hide special files = yes
# 隐藏没有读取权限的文件
hide unreadable = yes
;vfs objects = trim_perms full_audit
;full_audit:prefix = %u
;full_audit:syslog = true
;full_audit:success = create_file mkdirat renameat mknodat unlinkat
pwrite_recv pwrite_send pread_recv pread_send
include = /etc/samba/users/%U.share.conf
;以下解决中文乱码
;dos charset = UTF-8
;unix charset = utf-8
;display charset = utf-8

```

12. samba手动添加共享命令

```
vim /etc/samba/smb.conf
```

[共享名]

```

comment = Share Folder
path = /mnt/raid/www
guest ok = no
read only = no
browseable = yes
writeable = yes
create mask = 0644
directory mask = 0755
force user = <文件夹的用户>
valid user = <系统用户>

```

添加用户

```

useradd -M -s /usr/sbin/nologin <用户名>
echo -e "<密码>\n<密码>" | (smbpasswd -a -s <用户名>)

```

删除用户

```

smbpasswd -x <用户名>
userdel -rf <用户名>

```

重启服务

```
smbcontrol all reload-config
```

12.安装supervisor,开启python服务

```
apt install python3-pip python3-flask supervisor -y
```

```
systemctl enable supervisor
```

修改vim /etc/supervisor/conf.d/pysmb.conf

```
[program:pysmb]
```

```
directory=/mnt/raid/www
```

```
command=/usr/bin/python3 pysmb.py
```

```
process_name=pysmb
```

```
autostart=true
```

```
autorestart=true
```

```
startretries=10
```

```
redirect_stderr=true
```

```
stdout_logfile=/tmp/pysmb.log
```

重新加载supervisorctl reload

查看supervisorctl status

13.安装PHP7.4

```
sudo add-apt-repository ppa:ondrej/php
```

```
sudo apt update
```

```
sudo apt install -y php7.4 php7.4-cli php7.4-fpm php7.4-mysql php7.4-xml php7.4-curl php7.4-zip php7.4-mbstring php7.4-gd php7.4-iconv php7.4-pdo php7.4-sqlite3
```

修改/etc/php/7.4/php.ini中 以下字段的值

```
upload_max_filesize = 20000M
```

```
max_execution_time = 60
```

```
memory_limit = 256M
```

```
doc_root = /mnt/raid/www
```

修改/etc/php/7.4/php-fpm.d/www.conf中以下字段的值

```
listen = /dev/shm/php-fpm.sock
```

```
systemctl enable php7.4-fpm
```

14.安装nginx

```
apt install nginx
```

在/etc/nginx/nginx.conf中 include /etc/nginx/conf.d/*.conf上方添加以下字段

```
charset 'utf-8';
```

```
server_tokens off;
```

注释 include /etc/nginx/sites-enabled/*;

在/etc/nginx/conf.d/下新建file.conf文件,添加以下内容

```
server {
```

```
    server_name default_server;
```

```
    listen 80;
```

```
    root /mnt/raid/www;
```

```
    add_header X-Frame-Options "SAMEORIGIN";
```

```
    add_header X-XSS-Protection "1; mode=block";
```

```
    add_header X-Content-Type-Options "nosniff";
```

```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains;
preload" always;

proxy_send_timeout 300s;
proxy_read_timeout 300s;
resolver 8.8.8.8 ipv6=off;
client_max_body_size 20000M;
client_body_buffer_size 128k;

# log files
access_log /var/log/nginx/file_access.log;
error_log /var/log/nginx/file_error.log;

index index.php index.html;

location / {
    try_files $uri $uri/ /index.php?$query_string;
}

location /favicon.ico {
    root /mnt/raid/www;
}

location ~ \.(php|phar)(/.*)?$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.(?:php|phar))(/.*)$;

    fastcgi_intercept_errors on;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
    fastcgi_pass unix:/dev/shm/php-fpm.sock;
}

location /static {
    root /mnt/raid/www;
}
}

systemctl enable nginx
```

15.上传代码

```
mkdir -p /mnt/raid/www
开始上传代码
chown -R www-data:root /mnt/raid/www
```

16.测试

1. 查看python服务是否启动
2. 查看nginx, samba等服务

17.安装ttyd服务

```
wget https://rowsea.com/ttyd.x86_64
mv ttyd.x86_64 /usr/local/bin/ttyd
chmod -R 755 /usr/local/bin/ttyd
/usr/local/bin/ttyd -w -p 8082 login

vim /etc/systemd/system/ttyd.service
[Unit]
Description=ttyd - Web-based Terminal
After=network.target

[Service]
User=root # 以登录的用户身份运行，或者指定 root
# 如果是使用 root 权限，则使用以下这一行，通常不需要额外参数
ExecStart=/usr/local/bin/ttyd -w -p 7681 login # 如果是bash就直接登录
# 如果需要指定具体的用户和工作目录，可以使用下面这一行（示例）
# ExecStart=/usr/local/bin/ttyd -w -i 127.0.0.1 -p 7681 -U -l /home/user -c
user:password bash
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target

systemctl daemon-reload
systemctl enable ttyd
systemctl start ttyd
```

18.nginx配置

```
location /ssh {
    #auth_basic "Restricted Access";
    #auth_basic_user_file /etc/nginx/.htpasswd;
    proxy_pass http://127.0.0.1:7681/;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
```

19.双网卡添加持久化路由

```
vim /etc/systemd/system/static-route.service

[Unit]
Description=Static Route Configuration
After=network.target

[Service]
Type=oneshot
```

```
ExecStart=/sbin/ip route add 10.0.0.0/8 via 10.122.44.1 dev enp2s0
RemainAfterExit=yes
```

```
[Install]
WantedBy=multi-user.target

systemctl enable static-route
systemctl start static-route
```

20.安装redis

```
apt-get purge --auto-remove redis redis-server
apt install redis-server
systemctl enable redis
#systemctl stop redis
#修改redis存储路径
#vim /etc/redis/redis.conf
#找到 dir /var/lib/redis
#修改为 /mnt/raid/redis_data
#同步数据
#rsync -avz /var/lib/redis /mnt/raid/redis_data/
#chown -R redis:redis /mnt/raid/redis_data/
#重启redis
systemctl restart redis.service
```

21.安装mysql 5.7

```
vim /etc/apt/sources.list.d/mysql.list

deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-apt-config
deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-5.7
deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-tools
deb-src http://repo.mysql.com/apt/ubuntu/ bionic mysql-5.7

apt update
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys <错误ID>
apt-cache policy mysql-server

apt install mysql-client=5.7.40-1ubuntu18.04
apt install mysql-server=5.7.40-1ubuntu18.04

systemctl enable mysql
systemctl start mysql

开启远程访问：
vim /etc/mysql/mysql.conf.d/mysqld.conf
bind-address = 0.0.0.0
character-set-server=utf8mb4
vim /etc/mysql/conf.d/mysql.conf
default-character-set=utf8mb4

mysql -uroot -p
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' identified by '密码' WITH GRANT OPTION;
flush tables;
```

```
flush privileges;
```

22.使用certbot申请泛域名证书

```
apt install certbot python-certbot-nginx
```

执行 `certbot certonly --manual --preferred-challenges dns --cert-name lzloyo.cn -d *.lzloyo.cn` 提示添加txt记录

更新命令: `certbot renew --force-renewal`

nginx配置文件中添加

```
listen 443 ssl;
server_name it.lzloyo.cn;
ssl_certificate /etc/letsencrypt/live/lzloyo.cn/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/lzloyo.cn/privkey.pem;
include /etc/letsencrypt/options-ssl-nginx.conf;
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
```

23.nginx配置心跳包

```
location /ping {
    default_type application/json;
    return 200 '{"code": 0, "version": "1.x"}';
    #alias /var/www/version.json; // 返回更多内容时
}
```

RAID完整操作

1.mdadm操作 使用本地回环创建3个块设备

```
$ truncate -s 1G md0.img md1.img md2.img
$ sudo losetup -f --show md0.img
/dev/loop0
$ sudo losetup -f --show md1.img
/dev/loop1
$ sudo losetup -f --show md2.img
/dev/loop2
```

2.创建RAID,先以RAID 0为例, 创建一个跨3块盘的RAID 0

```
$ sudo mdadm --create /dev/md0 --level=0 --raid-devices=3 /dev/loop0 /dev/loop1
/dev/loop2
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

得到的 `/dev/md0` 就是创建完成的 RAID 块设备, 它的大小是约 3G, 可以使用 `mdadm --detail` 查看详细信息。

```
$ sudo mdadm --detail /dev/md0
/dev/md0:
    Version : 1.2
  Creation Time : Wed Feb 21 00:54:02 2024
    Raid Level : raid0
    Array Size : 3139584 (2.99 GiB 3.21 GB)
    Raid Devices : 3
    Total Devices : 3
    Persistence : Superblock is persistent

    Update Time : Wed Feb 21 00:54:02 2024
      State : clean
    Active Devices : 3
    Working Devices : 3
    Failed Devices : 0
    Spare Devices : 0


    Layout : -unknown-
    Chunk Size : 512K

Consistency Policy : none


    Name : hostname:0 (local to host hostname)
    UUID : 119661ca:ff0a76b7:28909e39:18af76dc
    Events : 0

   Number   Major   Minor   RaidDevice State
     0         7         0         0     active sync   /dev/loop0
     1         7         1         1     active sync   /dev/loop1
     2         7         2         2     active sync   /dev/loop2
```

对应的状态也可以通过 `/proc/mdstat` 查看:

```
$ cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [raid1] [raid0]
md0 : active raid0 loop2[2] loop1[1] loop0[0]
      3139584 blocks super 1.2 512k chunks

unused devices: <none>
```

使用 `--stop` 选项停止 RAID:

```
$ sudo mdadm --stop /dev/md0
mdadm: stopped /dev/md0
$ ls -lha /dev/md0
ls: cannot access '/dev/md0': No such file or directory
```

在 stop 之后, 如果需要再恢复, 可以使用 `--assemble` 选项:

```
$ sudo mdadm --assemble /dev/md0 /dev/loop0 /dev/loop1 /dev/loop2
mdadm: /dev/md0 has been started with 3 drives.
$ # 或者使用 --scan 参数:
$ sudo mdadm --assemble --scan
mdadm: /dev/md0 has been started with 3 drives.
```

将这个 RAID 0 拆掉，然后试试组建 RAID 1、RAID 5：

```
$ # RAID 1
$ sudo mdadm --stop /dev/md0
mdadm: stopped /dev/md0
$ sudo mdadm --misc --zero-superblock /dev/loop0 /dev/loop1 /dev/loop2
$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=3 /dev/loop0 /dev/loop1
/dev/loop2
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device.  If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
$ # RAID 5
$ sudo mdadm --stop /dev/md0
mdadm: stopped /dev/md0
$ sudo mdadm --misc --zero-superblock /dev/loop0 /dev/loop1 /dev/loop2
$ sudo mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/loop0 /dev/loop1
/dev/loop2
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

如果在重新创建 mdadm 阵列之前不清空 superblock，会输出类似以下的警告信息：

```
mdadm: /dev/loop0 appears to be part of a raid array:
      level=raid0 devices=3 ctime=Wed Feb 21 00:54:02 2024
mdadm: /dev/loop1 appears to be part of a raid array:
      level=raid0 devices=3 ctime=Wed Feb 21 00:54:02 2024
mdadm: /dev/loop2 appears to be part of a raid array:
      level=raid0 devices=3 ctime=Wed Feb 21 00:54:02 2024
```

3.重建操作

与 LVM 一章类似，这里展示在一块盘丢失（损坏）情况下的操作：

```
$ sudo mdadm --stop /dev/md0
mdadm: stopped /dev/md0
$ sudo losetup -D
$ sudo losetup -f --show md0.img
/dev/loop0
$ sudo losetup -f --show md1.img
/dev/loop1
$ # 此时 /dev/md0 已经自动构建，并且处于丢失一块盘的状态
$ sudo mdadm --detail /dev/md0
```

/dev/md0:

Version : 1.2
Creation Time : Thu Feb 22 13:05:14 2024
Raid Level : raid5
Array Size : 2093056 (2044.00 MiB 2143.29 MB)
Used Dev Size : 1046528 (1022.00 MiB 1071.64 MB)
Raid Devices : 3
Total Devices : 2
Persistence : Superblock is persistent

Update Time : Thu Feb 22 13:05:19 2024
State : clean, degraded
Active Devices : 2
Working Devices : 2
Failed Devices : 0
Spare Devices : 0

Layout : left-symmetric
Chunk Size : 512K

Consistency Policy : resync

Name : hostname:0 (local to host hostname)
UUID : 3e4455f5:65e4251c:b21c1c81:13b44a8b
Events : 18

Number	Major	Minor	RaidDevice	State	
0	7	0	0	active sync	/dev/loop0
1	7	1	1	active sync	/dev/loop1
-	0	0	2	removed	

\$ # 添加新的盘

\$ truncate -s 1G md3.img

\$ sudo losetup /dev/loop3 md3.img

\$ sudo mdadm --add /dev/md0 /dev/loop3

\$ sudo mdadm --detail /dev/md0

/dev/md0:

Version : 1.2
Creation Time : Thu Feb 22 13:05:14 2024
Raid Level : raid5
Array Size : 2093056 (2044.00 MiB 2143.29 MB)
Used Dev Size : 1046528 (1022.00 MiB 1071.64 MB)
Raid Devices : 3
Total Devices : 3
Persistence : Superblock is persistent

Update Time : Thu Feb 22 13:52:32 2024
State : clean, degraded, recovering
Active Devices : 2
Working Devices : 3
Failed Devices : 0
Spare Devices : 1

Layout : left-symmetric
Chunk Size : 512K

Consistency Policy : resync

Rebuild Status : 35% complete

Name : hostname:0 (local to host hostname)
UUID : 3e4455f5:65e4251c:b21c1c81:13b44a8b
Events : 26

Number	Major	Minor	RaidDevice	State
0	7	0	0	active sync /dev/loop0
1	7	1	1	active sync /dev/loop1
3	7	3	2	spare rebuilding /dev/loop3

添加新盘后，可以看到重建操作会自动开始。

4.完整性检查

这里展示三盘 RAID 1 下进行检查与修复的场景：

```
$ sudo mdadm --stop /dev/md0
mdadm: stopped /dev/md0
$ sudo mdadm --misc --zero-superblock /dev/loop0 /dev/loop1 /dev/loop3
$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=3 /dev/loop0 /dev/loop1 /dev/loop3
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device.  If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
$ # 向其中一块盘写入垃圾数据
$ sudo dd if=/dev/urandom of=/dev/loop0 bs=1M count=1 oseek=100
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00406889 s, 258 MB/s
$ sudo bash -c 'echo check > /sys/block/md0/md/sync_action'
$ # 在 /proc/mdstat 中可以看到检查的进度
$ cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [raid1] [raid0]
md0 : active raid1 loop3[2] loop1[1] loop0[0]
      1046528 blocks super 1.2 [3/3] [UUU]
      [=====>.]  check = 95.3% (998016/1046528) finish=0.0min
      speed=249504K/sec

unused devices: <none>
$ # 由于我们这里盘很小，所以检查很快就结束了
$ cat /sys/block/md0/md/mismatch_cnt # 获取不一致的块数
4096
$ # 由于我们有足够多的副本，可以尝试修复
$ sudo bash -c 'echo repair > /sys/block/md0/md/sync_action'
$ # 在 /proc/mdstat 中也可以看到修复的进度
$ cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [raid1] [raid0]
md0 : active raid1 loop3[2] loop1[1] loop0[0]
```

```
1046528 blocks super 1.2 [3/3] [UUU]
[=====>.....] resync = 57.4% (601984/1046528) finish=0.0min
speed=300992K/sec

unused devices: <none>
$ # 修复完成后, mismatch_cnt 中仍然记录的是不一致的数量
$ # 再执行一次 check 后 mismatch_cnt 值会变为 0
```

5.监控 SMART信息

阅读 SMART 信息可以帮助了解硬盘的健康状态。在服务器上, 如果使用硬件 RAID, 使用 `smartctl` 需要添加额外的参数来从 RAID 控制器获取真实的磁盘信息, 例如下面的例子:

```
$ sudo smartctl --scan
/dev/sda -d scsi # /dev/sda, SCSI device
/dev/sdb -d scsi # /dev/sdb, SCSI device
/dev/sdc -d scsi # /dev/sdc, SCSI device
/dev/sdd -d scsi # /dev/sdd, SCSI device
/dev/bus/4 -d megaraid,8 # /dev/bus/4 [megaraid_disk_08], SCSI device
/dev/bus/4 -d megaraid,9 # /dev/bus/4 [megaraid_disk_09], SCSI device
/dev/bus/4 -d megaraid,10 # /dev/bus/4 [megaraid_disk_10], SCSI device
/dev/bus/4 -d megaraid,11 # /dev/bus/4 [megaraid_disk_11], SCSI device
/dev/bus/4 -d megaraid,12 # /dev/bus/4 [megaraid_disk_12], SCSI device
/dev/bus/4 -d megaraid,13 # /dev/bus/4 [megaraid_disk_13], SCSI device
/dev/bus/4 -d megaraid,14 # /dev/bus/4 [megaraid_disk_14], SCSI device
/dev/bus/4 -d megaraid,15 # /dev/bus/4 [megaraid_disk_15], SCSI device
/dev/bus/0 -d megaraid,8 # /dev/bus/0 [megaraid_disk_08], SCSI device
/dev/bus/0 -d megaraid,9 # /dev/bus/0 [megaraid_disk_09], SCSI device
/dev/bus/0 -d megaraid,10 # /dev/bus/0 [megaraid_disk_10], SCSI device
/dev/bus/0 -d megaraid,11 # /dev/bus/0 [megaraid_disk_11], SCSI device
/dev/bus/0 -d megaraid,12 # /dev/bus/0 [megaraid_disk_12], SCSI device
/dev/bus/0 -d megaraid,13 # /dev/bus/0 [megaraid_disk_13], SCSI device
$ sudo smartctl -a /dev/sdd # 直接查询只能看到没有意义的控制器信息
smartctl 7.2 2020-12-30 r5155 [x86_64-linux-5.10.0-21-amd64] (local build)
Copyright (C) 2002-20, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Vendor:                AVAGO
Product:               MR9361-8i
Revision:              4.68
Compliance:            SPC-3
User Capacity:         1,919,816,826,880 bytes [1.91 TB]
Logical block size:    512 bytes
Physical block size:   4096 bytes
Logical Unit id:       0x600605b00f17786026223e2d33c1767b
Serial number:         007b76c1332d3e22266078170fb00506
Device type:           disk
Local Time is:         Sun Feb 11 18:40:45 2024 CST
SMART support is:      Unavailable - device lacks SMART capability.

=== START OF READ SMART DATA SECTION ===
Current Drive Temperature:    0 C
Drive Trip Temperature:       0 C

Error Counter logging not supported
```

```
Device does not support Self Test logging
$ sudo smartctl -a /dev/bus/4 -d megaraid,8 # 添加参数可以看到真实的磁盘信息
(内容省略)
```

`smartctl -a` 的输出主要分为两个部分：information section 和 smart data section。

Information section 展示硬盘的基本信息，包括型号、序列号、容量、固件版本等。

Smart data section 则展示了硬盘的 SMART 信息。其中**自检信息**与**错误记录**均会显示，其他的部分视硬盘类型而定。

SMART 还支持让磁盘自检，对应的参数为 `smartctl -t`，可以选择不同的测试参数（例如 short、long 等）

安装 smartmontools 之后，可以启用 smartd 服务（`smartd.service` 或 `smartmontools.service`）。该服务会每隔一段时间检查硬盘的 SMART 信息，并在发现问题时发送邮件通知管理员（在正确配置的情况下）。默认 Debian 提供的 `/etc/smartd.conf` 的有效内容如下：

```
DEVICESCAN -d removable -n standby -m root -M exec
/usr/share/smartmontools/smartd-runner
```

`/usr/share/smartmontools/smartd-runner` 会调用 `/etc/smartmontools/run.d/` 下的文件，其中默认提供的 `10mail` 会使用系统的 `mail` 命令发送邮件。

在配置中添加 `-M test` 可以在启动时发送测试邮件

6. 紧急救援

对于 RAID 0 类型的配置（包括 RAID 10/01），`mdadm` 工具支持拼接多个（没有 `mdadm superblock` 的）块设备的内容组成 RAID，参考命令如下：

```
sudo mdadm --build --assume-clean -c 128 --level=0 --raid-devices=8 --
size=195364584 /dev/md0 /dev/mapper/disk1 /dev/mapper/disk2 /dev/mapper/disk3
/dev/mapper/disk4 /dev/mapper/disk5 /dev/mapper/disk6 /dev/mapper/disk7
/dev/mapper/disk8
```

PVE中添加多网桥vmbr1,桥接物理接口

```
vim /etc/network/interfaces

auto vmbr1
iface vmbr1 inet static
    address 192.168.1.200/24
    bridge_ports enp2s0
    bridge_stp off
    bridge_fd 0
    post-up ip rule add from 192.168.1.0/24 table 101
    post-up ip route add default via 192.168.1.1 dev vmbr1 table 101
    pre-down ip rule del from 192.168.1.0/24 table 101
    pre-down ip route del default via 192.168.1.1 dev vmbr1 table 101
```

指定网卡PING

```
linux:
    ping -I <eth0> or <eth0的IP> www.baidu.com
win10:
    ping -S <网卡IP> www.baidu.com
```

安装NTP Server

1.修改时区

```
timedatectl set-timezone Asia/Shanghai
```

2.ubuntu设置24h显示

```
localectl set-locale LC_TIME="en_GB.UTF-8"
```

3.关闭systemd-timesyncd服务，防止与ntpd冲突

```
sudo systemctl stop systemd-timesyncd
sudo systemctl disable systemd-timesyncd
sudo apt install -y ntp
sudo systemctl enable ntp --now
```

4.修改/etc/ntp.conf

```
# 1) 记录频偏
driftfile /var/lib/ntp/ntp.drift
leapfile /usr/share/zoneinfo/leap-seconds.list

# 2) 上游时间源：至少 3~4 个，分布不同机构/地域
pool 0.cn.pool.ntp.org iburst
pool 1.cn.pool.ntp.org iburst
pool 2.cn.pool.ntp.org iburst
pool 3.cn.pool.ntp.org iburst
# 如需更稳，可替换为企业可信源或本地区权威源

# 3) 访问控制（默认收紧）
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery

# 允许本机与本地网段查询/同步（示例：10.0.0.0/24）
restrict 127.0.0.1
restrict ::1
restrict 10.0.0.0 mask 255.0.0.0 nomodify notrap
```

```
# 4) 接口控制 (可选: 仅监听内网口)
# interface ignore wildcard
# interface listen eth0

# 5) 异常容错 (可选): 上游全不可达时有序参考
# tos orphan 10
```

```
sudo systemctl restart ntp
sudo ufw allow 123/udp
sudo ufw reload
```

5.运行状态检查

```
# 进程/端口
sudo systemctl status ntp
sudo ss -ltnp | grep :123

# 同步状态与上游列表
ntpq -p -n

# 粗粒度健康度
ntpstat
```

vim去掉文件中的^M

编辑模式下替换: %s/\r//g

ZeroTier安装及路由moon节点加速

zerotier定义了几个专业名词:
PLANET 行星级的服务器, zerotier各地的根服务器, 有日本、新加坡等地。
MOON 卫星级服务器, 用户自建的私有根服务器, 起到中转加速的作用。
LEAF 相当于各个枝叶, 就是每台连接到该网络的机器节点

充当MOON的机器最好有公网IP

移动网络对zerotier默认的PLANET节点不友好,需要自建moon节点进行通信

```
windows: C:\ProgramData\ZeroTier\One
Macintosh: /Library/Application Support/ZeroTier/One (在 Terminal 中应为
/Library/Application\ Support/ZeroTier/One)
Linux: /var/lib/zerotier-one
FreeBSD/OpenBSD: /var/db/zerotier-one

zerotier-idtool initmoon identity.public >>moon.json //生成moon.json文件
```

修改生成的 moon.json 文件 将"stableEndpoints":[] 修改下, 放上分配的ip地址及zerotier的端口

如下:

```
{
  "id": "22f6cf7653",
  "objtype": "world",
  "roots": [
    {
      "identity":
"22f6cf7653:0:03577cdd2129d3f0e5d689e54a7c5a221d50e8b0497c745a85301276e8aadf6eae1
0fa185913753b61f61926aa22cb6e72b285eb0f40c4ef1eea414edd5e969a",
      "stableEndpoints": ["192.168.192.88/9993"]
    }
  ],
  "signingKey":
"4224275d0a577dbf5a3e459f1b48c121b9dc943bf163329855a8cfcaf0519e36676878aaa1984c6a
6552115fe1ff766945611de0373b25a5dead4682c5c063f7",
  "signingKey_SECRET":
"5227bd8e8341ef1fea2a0c9f13090f3f64c54daf72fea4040532b6ee170fa751841f5001df64e5d8
49e2b933391a0c056a4dc4a0fea1212d32f214012f13dc74",
  "updatesMustBeSignedBy":
"4224275d0a577dbf5a3e459f1b48c121b9dc943bf163329855a8cfcaf0519e36676878aaa1984c6a
6552115fe1ff766945611de0373b25a5dead4682c5c063f7",
  "worldType": "moon"
}
```

修改完保存, 执行:

```
zerotier-idtool genmoon moon.json
//显示结果
# wrote 00000022f6cf7653.moon (signed world with timestamp 1550806005779)
//目录下生成了00000022f6cf7653.moon的文件
mkdir moons.d //新建文件夹
mv 00000022f6cf7653.moon moons.d //移动到moons.d文件夹
killall -9 zerotier-one //重启
zerotier-one -d
```

将生成的 00000022f6cf7653.moon 复制到其它机器的 /var/lib/zerotier-one/moons.d 目录下,并重启zerotier服务

或者使用 zerotier-cli orbit 22f6cf7653 22f6cf7653 将MOON节点加入到客户端节点上

重启zerotier后查看 zerotier-cli listpeers

注意, 阿里云机器需要在安全组 开启 9993的udp协议

如需要使用TCP进行中继:

Linux设备路径: /var/lib/zerotier-one/local.conf

Windows设备: c:\ProgramData\ZeroTier\One\local.conf

Mac设备: /Library/Application Support/ZeroTier/One/local.conf

添加以下配置内容来强制TCP中继:

```
{
  "settings": {
    "forceTcpRelay": true
  }
}
```

保存后重启zerotier服务即可

命令行启动chrome的socks5

```
"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --show-app-list --
proxy-server="SOCKS5://127.0.0.1:7090"
```

pyinstall打包

添加静态文件及动态导入包

项目中如果有用到aiohttp, 不能打包

```
pyi-makespec -w xxx.py
```

```
pyinstaller -D xxx.spec
```

```
pyinstaller -F server.py -n server #这样就只生成一个二进制文件
```

```
# -*- mode: python -*-

block_cipher = None

a = Analysis(['server.py'],
             pathex=['/Users/tony/Downloads/www/rowsea'],
             binaries=[],
             datas=[('./app/hyy/static', 'app/hyy/static'),
                   ('./app/hyy/templates', 'app/hyy/templates')],
             hiddenimports=['app', 'app.hyy', 'app.hyy.config', 'app.hyy.urls',
                            'app.hyy.alembic'],
             hookspath=[],
             runtime_hooks=[],
             excludes=[],
             win_no_prefer_redirects=False,
             win_private_assemblies=False,
             cipher=block_cipher,
             noarchive=False)
pyz = PYZ(a.pure, a.zipped_data,
          cipher=block_cipher)
exe = EXE(pyz,
          a.scripts,
          [],
          exclude_binaries=True,
```

```
        name='server',
        debug=False,
        bootloader_ignore_signals=False,
        strip=False,
        upx=True,
        console=True )
coll = COLLECT(exe,
                a.binaries,
                a.zipfiles,
                a.datas,
                strip=False,
                upx=True,
                name='server')
```

tcpdump抓包header

```
tcpdump -i br0 port 5850 -s 1024 -l -A
```

```
tcpdump tcp -i eth1 port 443 -s 0 -l -A -w ./target.cap
```

(1)tcp: ip icmp arp rarp 和 tcp、udp、icmp这些选项等都要放到第一个参数的位置，用来过滤数据包的类型

(2)-i eth1 : 只抓经过接口eth1的包

(3)-t : 不显示时间戳

(4)-s 0 : 抓取数据包时默认抓取长度为68字节。加上-s 0 后可以抓到完整的数据包

(5)-c 100 : 只抓取100个数据包

(6)dst port ! 22 : 不抓取目标端口是22的数据包

(7)src net 192.168.1.0/24 : 数据包的源网络地址为192.168.1.0/24

(8)-w ./target.cap : 保存成cap文件，方便用ethereal(即wireshark)分析

linux下grep递归搜索指定后缀名的内容

```
grep -rn --include='*.tpl' "basket"
```

linux 清空 last和lastb及history

清除登陆系统成功的记录

```
echo "">/var/log/wtmp #清空last 此文件默认打开时乱码，可查到ip等信息
```

清除登陆系统失败的记录

```
echo "">/var/log/btmp #清空lastb 此文件默认打开时乱码，可查到 登陆失败 信息
```

清除历史执行命令

```
history -c # 清空历史执行命令
```

linux ssh 生成socks5代理

```
ssh -D10000 root@101.0.20.11 -p 1821
```

ubuntu 多网卡配置指定路由

```
network:
  ethernets:
    enp3s0:
      addresses:
        - 192.168.1.253/24
      gateway4: 192.168.1.1
      nameservers:
        addresses:
          - 223.5.5.5
        search: []
    enp2s0:
      addresses:
        - 10.140.217.253/23
      routes:
        - to: 10.0.0.0/8
          via: 10.140.216.1
  version: 2
```

linux实现双网卡bond及桥接

1.介绍

将多块网卡绑定同一IP地址对外提供服务，可以实现高可用或者负载均衡。直接给两块网卡设置同一IP地址是不可以的。通过bonding，虚拟一块网卡对外提供连接，物理网卡的被修改为相同的MAC地址

2.Bonding工作模式

Linux网卡绑定共七种模式，分别是如下模式：

- mode=0 round-robin轮询策略(Round-robin policy)
- mode=1 active-backup主备策略(Active-backup policy)
- mode=2 load balancing (xor)异或策略(XOR policy)
- mode=3 fault-tolerance (broadcast)广播策略(Broadcast policy)
- mode=4 lacp IEEE 802.3ad 动态链路聚合(IEEE 802.3ad Dynamic link aggregation)
- mode=5 transmit load balancing适配器传输负载均衡(Adaptive transmit load balancing)
- mode=6 adaptive load balancing适配器负载均衡(Adaptive load balancing)

2.1 round-robin轮询策略

```
cat /proc/net/bonding/bond0
```

```
Bonding Mode: load balancing (round-robin)
```

该模式下，链路处于负载均衡状态，数据以轮询方式向每条链路发送报文，基于per packet方式发送。即每条链路各一个数据包。这模式好处在于增加了带宽，同时支持容错能力，当有链路出问题，会把流量切换到正常的链路上。该模式下，交换机端需要配置聚合口，在cisco交换机上叫port channel

2.2.active-backup主备策略

该模式拓扑图与上图相同

```
cat /proc/net/bonding/bond0
```

Bonding Mode: fault-tolerance (active-backup)

在该模式下，一个端口处于主状态，一个处于备状态，所有流量都在主链路上发出和接收，备链路不会有任何流量。当主端口down掉时，备端口接管主状态。同时可以设置primary网卡，若primary网卡出现故障，切换至备网卡，primary网卡回复后，流量自动回切。这种模式接入不需要交换机端支持。

2.3.load balancing (xor)异或策略

该模式拓扑图与上图相同

```
cat /proc/net/bonding/bond0
```

Bonding Mode: load balancing (xor)

在该模式下，通过源和目标mac做hash因子来做xor算法来选择链路，这样就使得到达特定对端的流量总是从同一个接口上发出。和balance-rr一样，交换机端口需要能配置为“port channel”。

值得注意的是，若选择这种模式，如果所有流量源和目标mac都固定了，例如使用“网关模式”，即所有对外的数据传输均固定走一个网关，那么根据该模式的描述，分发算法算出的线路就一直是同一条，另外一条链路不会有任何数据流，那么这种模式就没有多少意义了。

2.4.fault-tolerance (broadcast)广播策略

```
cat /proc/net/bonding/bond0
```

Bonding Mode: fault-tolerance (broadcast)

这种模式的特点是一个报文会复制两份往bond下的两个接口分别发送出去。当有对端交换机失效，我们感觉不到任何丢包。这个模式也需要交换机配置聚合口。

从这个ping信息可以看到，这种模式的特点是，同一个报文服务器会复制两份分别往两条线路发送，导致回复两份重复报文，虽然这种模式不能起到增加网络带宽的效果，反而给网络增加负担，但对于一些需要高可用的环境下，例如RAC的心跳网络，还是有一定价值的。

2.5.lacp IEEE 802.3ad 动态链路聚合

该模式拓扑结构与主备模式相同

```
cat /proc/net/bonding/bond0
```

Bonding Mode: IEEE 802.3ad Dynamic link aggregation

该模式是基于IEEE 802.3ad Dynamic link aggregation（动态链接聚合）协议，针对该协议的介绍，在公众号之前的文章中有所涉及。

在该模式下，操作系统和交换机都会创建一个聚合组，在同一聚合组下的网口共享同样的速率和双工设定。操作系统根据802.3ad 协议将多个slave 网卡绑定在一个聚合组下。聚合组向外发送数据选择哪一块儿网卡是基于传输hash 策略，该策略可以通过xmit_hash_policy 选项从缺省的XOR 策略改变到其他策略。

该模式的必要条件：

- ethtool 支持获取每个slave 的速率和双工设定；
- 交换机支持IEEE 802.3ad Dynamic link aggregation。

大多数交换机需要经过特定配置才能支持802.3ad 模式。

2.6.transmit load balancing适配器传输负载均衡

```
cat /proc/net/bonding/bond0
```

Bonding Mode: transmit load balancing

这种模式相较load balancing (xor)异或策略及LACP模式的hash策略相对智能，会主动根据对端的MAC地址上的流量，智能的分配流量从哪个网卡发出。但不足之处在于，仍使用一块网卡接收数据。存在的问题与load balancing (xor)也是一样，如果对端MAC地址是唯一的，那么策略就会失效。这个模式下bond成员使用各自的mac，而不是上面几种模式是使用bond0接口的mac。无需交换机支持

2.7.adaptive load balancing适配器负载均衡

该模式拓扑结构与上图一致。

```
cat /proc/net/bonding/bond0
```

Bonding Mode: adaptive load balancing

该模式除了balance-tlb适配器传输负载均衡模式的功能外，同时加上针对IPv4流量接收的负载均衡。接收负载均衡是通过ARP协商实现的。在进行ARP协商的过程中，bond模块将对端和本地的mac地址进行绑定，这样从同一端发出的数据，在本地也会一直使用同一块网卡来接收。若是网络上发出的广播包，则由不同网卡轮询的方式来进行接收。通过这种方式实现了接收的负载均衡。该模式同样无需交换机支持。

3.配置bonding

3.1 默认情况下bonding模块没有被加载，需先加载bonding模块

```
[root@c2 ~]# lsmod |grep bonding
[root@c2 ~]# modprobe bonding
[root@c2 ~]# lsmod |grep bonding
bonding                152656  0
```

3.2 新建bond网口的配置文件

```
[root@c2 ~]# cat /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
NAME=bond0
TYPE=Bond
BONDING_MASTER=yes
IPADDR=192.168.0.100
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
DNS1=8.8.8.8
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="mode=0 miimon=100"
```

3.3 网口eth0、eth1的配置文件

```
[root@c2 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
NAME="eth0"
DEVICE="eth0"
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
MASTER=bond0
SLAVE=yes
```

```
[root@c2 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
NAME="eth1"
DEVICE="eth1"
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
MASTER=bond0
SLAVE=yes
```

3.4 br0配置

```
[root@c2 network-scripts]# cat ifcfg-br0
TYPE=Bridge
BOOTPROTO=static
PEEDNS=yes
NAME=br0
DEVICE=br0
ONBOOT=yes
IPADDR=192.168.0.100
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
DNS1=8.8.8.8
USERCTL=no
```

3.5 重新加载网络配置

```
[root@c2 ~]# nmcli connection reload
[root@c2 ~]# systemctl restart network.service
```

3.6 查看bonding状态

```
[root@c2 ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: adaptive load balancing
Primary Slave: None
Currently Active Slave: eth1
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
```

```
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:ba:03:9e
Slave queue ID: 0
```

3.7 删除网卡

```
nmcli con show
nmcli con delete team0
systemctl restart network
```

4. 双网卡配置team绑定实战

需要在生产环境下部署kvm虚拟化生产环境，因为网络采用的是叶脊拓扑，需要对多个网卡进行绑定，常用绑定技术为bond和team，team是centos7新支持的，性能和稳定性都高于bond,因此决定采用team对多个网卡进行绑定。需要绑定的两块万兆卡名称为ens7f0和ens7f1配置步骤如下：

4.1、配置ens7f0

```
[root@localhost network-scripts]# cat ifcfg-ens7f0
NAME=ens7f0
DEVICE=ens7f0
ONBOOT=yes
DEVICETYPE=TeamPort
TEAM_MASTER=team0
TEAM_PORT_CONFIG='{"prio":9}'
```

4.2、配置ens7f1

```
[root@localhost network-scripts]# cat ifcfg-ens7f1
NAME=ens7f1
DEVICE=ens7f1
DEVICETYPE=TeamPort
ONBOOT=yes
TEAM_MASTER=team0
TEAM_PORT_CONFIG={"prio": 10,"sticky": true}
```

4.3、配置team0

```
[root@localhost network-scripts]# cat ifcfg-team0
DEVICE=team0
NAME=team0
DEVICETYPE=Team
TEAM_CONFIG='{"runner": {"name": "activebackup"}, "link_watch": {"name": "ethtool", "delay_up": 2500, "delay_down": 1000}}'
ONBOOT=yes
BRIDGE=br0
```

4.4、配置br0

```
[root@localhost network-scripts]# cat ifcfg-br0
TYPE=Bridge
BOOTPROTO=static
NAME=br0
DEVICE=br0
ONBOOT=yes
IPADDR=192.168.217.12
NETMASK=255.255.255.0
GATEWAY=192.168.217.254
DNS1=119.29.29.29
DNS2=8.8.8.8
```

4.5、查看网卡状态

```
[root@localhost network-scripts]# teamdctl team0 state view
setup:
  runner: activebackup
ports:
  ens7f0
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
  ens7f1
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: ens7f0
```

至此，整个team端口组配置完毕，实测无任何问题

firewall常用命令及使用

1、管理端口

1. 列出 dmz 级别的被允许的进入端口

```
# firewall-cmd --zone=dmz --list-ports
```

2. 允许 tcp 端口 8080 至 dmz 级别

```
# firewall-cmd --zone=dmz --add-port=8080/tcp
```

3. 允许某范围的 udp 端口至 public 级别，并永久生效

```
# firewall-cmd --zone=public --add-port=5060-5059/udp --permanent
```

2、网卡接口

1. 列出 public zone 所有网卡

```
# firewall-cmd --zone=public --list-interfaces
```

2. 将 eth0 添加至 public zone，永久

```
# firewall-cmd --zone=public --permanent --add-interface=eth0
```

3. eth0 存在与 public zone，将该网卡添加至 work zone，并将之从 public zone 中删除

```
# firewall-cmd --zone=work --permanent --change-interface=eth0
```

4. 删除 public zone 中的 eth0，永久

```
# firewall-cmd --zone=public --permanent --remove-interface=eth0
```

3、管理服务

1. 添加 smtp 服务至 work zone

```
# firewall-cmd --zone=work --add-service=smtp
```

2. 移除 work zone 中的 smtp 服务

```
# firewall-cmd --zone=work --remove-service=smtp
```

4、配置 external zone 中的 ip 地址伪装

1. 查看

```
# firewall-cmd --zone=external --query-masquerade
```

2. 打开伪装

```
# firewall-cmd --zone=external --add-masquerade
```

3. 关闭伪装

```
# firewall-cmd --zone=external --remove-masquerade
```

5、配置 public zone 的端口转发

1. 要打开端口转发，则需要先

```
# firewall-cmd --zone=public --add-masquerade
```

2. 然后转发 tcp 22 端口至 3753

```
# firewall-cmd --zone=public --add-forward-port=port=22:proto=tcp:toport=3753
```

3. 转发 22 端口数据至另一个 ip 的相同端口上

```
# firewall-cmd --zone=public --add-forward-port=port=22:proto=tcp:toaddr=192.168.1.100
```

4. 转发 22 端口数据至另一 ip 的 2055 端口上

```
# firewall-cmd --zone=public --add-forward-port=port=22:proto=tcp:toport=2055:toaddr=192.168.1.100
```

6、配置 public zone 的 icmp

1. 查看所有支持的 icmp 类型

```
# firewall-cmd --get-icmptypes  
destination-unreachable echo-reply echo-request parameter-problem redirect  
router-advertisement router-solicitation source-quench time-exceeded
```

2. 列出

```
# firewall-cmd --zone=public --list-icmp-blocks
```

3. 添加 echo-request 屏蔽

```
# firewall-cmd --zone=public --add-icmp-block=echo-request [--timeout=seconds]
```

4. 移除 echo-reply 屏蔽

```
# firewall-cmd --zone=public --remove-icmp-block=echo-reply
```

7、IP 封禁（这个是我们平时用得最多的）

```
# firewall-cmd --permanent --add-rich-rule="rule family='ipv4' source address='222.222.222.222' reject" 单个IP
# firewall-cmd --permanent --add-rich-rule="rule family='ipv4' source address='222.222.222.0/24' reject" IP段
# firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=192.168.1.2 port port=80 protocol=tcp accept" 单个IP的某个端口
```

这个是我们用得最多的。封一个IP，和一个端口 reject 拒绝 accept 允许

当然，我们仍然可以通过 ipset 来封禁 ip

1. 封禁 ip

```
# firewall-cmd --permanent --zone=public --new-ipset=blacklist --type=hash:ip
# firewall-cmd --permanent --zone=public --ipset=blacklist --add-entry=222.222.222.222
```

2. 封禁网段

```
# firewall-cmd --permanent --zone=public --new-ipset=blacklist --type=hash:net
# firewall-cmd --permanent --zone=public --ipset=blacklist --add-entry=222.222.222.0/24
```

3. 倒入 ipset 规则

```
# firewall-cmd --permanent --zone=public --new-ipset-from-file=/path/blacklist.xml
```

4. 然后封禁 blacklist

```
# firewall-cmd --permanent --zone=public --add-rich-rule='rule source ipset=blacklist drop'
```

8、IP封禁和端口

```
# firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=192.168.1.2 port port=80 protocol=tcp accept"
```

只对192.168.1.2这个IP只能允许80端口访问（拒绝访问只需把 accept 换成 reject、删除该规则把 -add-rich-rule 改成 -remove-rich-rule即可）

```
# firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=192.168.1.2/24 port port=80 protocol=tcp accept"
```

只对192.168.1.2这个IP段只能允许80端口访问（拒绝访问只需把 accept 换成 reject、删除该规则把 -add-rich-rule 改成 -remove-rich-rule即可）

9、双网卡内网网卡不受防火墙限制

```
# firewall-cmd --permanent --zone=public --add-interface=eth1
```

公网网卡-zone=public默认区域

```
# firewall-cmd --permanent --zone=trusted --add-interface=eth2
```

内网网卡-zone=trusted是受信任区域 可接受所有的网络连接

10、重新载入以生效

```
# firewall-cmd --reload
```

linux挂载大于2T硬盘并分区

下面是简单步骤

1, 将磁盘上原有的分区删除掉:

进入: `#parted /dev/sdb`

查看: `(parted) p`

删除: `(parted) rm 1`

`(parted) rm 2`

2, 将磁盘格式变成gpt的格式 (因为parted只能针对gpt格式的磁盘进行操作)

转换: `(parted) mklabel gpt`

设置单位为TB: `(parted) unit MB (GB, TB)`

分区: `(parted) mkpart primary 1 500` (分第一个主分区500MB)

分区: `(parted) mkpart primary 501 1000` (分第二个主分区500MB)

分区: `(parted) mkpart logical 1001 2000` (分第三个逻辑分区1000MB) (parted的逻辑分区不用先分扩展分区, 直接一步到位)

查看: `(parted) p`

退出: `(parted) quit` (parted分区自动保存, 不用手动保存)

3, 格式化已经分好的区

`# mkfs -t ext4 /dev/sdb1`

4, 挂载

`# mount /dev/sdb1 /mnt`

5, 开机自动挂载:

`# echo "/dev/sdb1 /mnt ext4 defaults 0 0" >>/etc/fstab`

lvm添加新盘步骤

```
fdisk /dev/vda
```

n # 新建分区

p # 主分区

回车 # 默认分区号

回车 # 默认开始位置

回车 # 默认结束位置

t # 指定分区格式

```
L # 列出可选格式
8e # 指定格式LVM
w # 保存操作

reboot

创建物理卷和分区
pvcreate /dev/sda3
vgextend centos /dev/sda3
vgcreate VGroup00 /dev/sdb (创建新的vg)
lvcreate -l +100%free -n Lvol100 VGroup00 (创建lv)

我们查看物理卷情况
vgdisplay

按照实际大小酌情增加，比如我现在需增加40GB。
lvresize -L +40G /dev/mapper/centos-root

然后动态扩容分区的大小：
resize2fs /dev/mapper/centos-root
报错的话，执行命令
xfs_growfs /dev/mapper/centos-root

.最后，查看最终的结果
df -Th

swap扩容后格式化
swapoff /dev/mapper/centos-swap

lvresize -L + 10G /dev/mapper/centos-swap
mkswap /dev/mapper/centos-swap
swapon /dev/mapper/centos-swap
```

npm修改国内源及安装electron

打开npm的配置文件

```
npm config edit
```

在空白处将下面几个配置添加上去,注意如果有原有的这几项配置，就修改

```
proxy=http://127.0.0.1:10807/
https-proxy=http://127.0.0.1:10807/
registry=https://registry.npmmirror.com
electron_mirror=https://cdn.npmmirror.com/binaries/electron/
electron_builder_binaries_mirror=https://npmmirror.com/mirrors/electron-builder-binaries/
```

修改源地址为国内 NPM 镜像

```
npm config set registry https://registry.npmmirror.com
```

修改源地址为官方源

```
npm config set registry https://registry.npmjs.org/
```

npm安装提示'current user("nobody") does not have permission'的解决

```
npm -g config set user root
```

linux下查看机器硬件名称与型号

查看机箱型号

```
dmidecode | grep "Product Name"
```

```
yum install smartmontools
```

#查看硬盘型号

```
smartctl --all /dev/sda
```

#查看cpu型号

```
cat /proc/cpuinfo
```

查看内存型号及插槽

```
dmidecode | grep -A16 "Memory Device$"
```

linux挂载img文件修改分区大小

1. 挂载img镜像

```
losetup -fP --show your_image.img
```

2. 扩容img文件本身大小 (如果需要扩大)

在末尾增加1GB

```
dd if=/dev/zero bs=1M count=1024 >> your_image.img
```

重新读取分区表

```
losetup -c /dev/loop0
```

3. 调整分区大小(使用parted)

```
parted /dev/loop0
```

```
(parted) print
```

记住要修改的分区编号 (例如 2)

```
(parted) resizepart 2 100% # 将2号分区扩大到末尾
```

```
(parted) quit
```

4. 调整文件系统大小

对于ext文件系统

```
e2fsck -f /dev/loop0p2 # 检查扩容的分区文件系统
resize2fs /dev/loop0p2 # 扩容
```

对于xfs文件系统

```
mount /dev/loop0p2 /mnt
xfs_growfs /mnt
```

5. 卸载镜像

```
losetup -d /dev/loop0
```

mysql5.7/8.0远程授权及修改密码

修改/etc/my.conf，添加
bind-address=0.0.0.0

你想从任何主机连接到mysql服务器的话。

```
GRANT ALL PRIVILEGES ON *.* TO '用户名'@'授权的IP' identified by '123345' WITH
GRANT OPTION ;
```

修改密码

```
update user set password=PASSWORD('123456') where user='root';
flush tables;
flush privileges;
```

修改默认字符

[mysql]

设置mysql客户端默认编码

default-character-set=utf8mb4

[mysqld]

允许最大连接

max_connections=200

服务端默认utf8编码

character-set-server=utf8mb4

默认存储

default-storage-engine=INNODB

[client]

#设置客户端编码

default-character-set=utf8mb4

mysql 8.0 的远程授权

```
CREATE USER 'root'@'%' IDENTIFIED BY 'root';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
```

#查看临时密码:

```
grep 'temporary password' /var/log/mysqld.log
```

mysql8.0 修改认证协议 支持旧版本客户端

1. 查看mysql使用的认证协议
use mysql;
select user,host,plugin,authentication_string from user;
2. 修改my.cnf文件, 添加
[mysqld]
default_authentication_plugin=mysql_native_password
3. 重启mysql
systemctl restart mysqld
4. 修改以前的帐号的认证方式
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'passwd';
FLUSH PRIVILEGES;

Ubuntu自定义Systemctl服务脚本

ubuntu的systemctl 脚本一般存放在: /etc/systemd/system/

目录下又存在两种类型的文件:

*.service # 服务unit文件

*.target # 开机级别unit

ubuntu的每一个服务以.service结尾, 一般会分为3部分: [Unit]、[Service]和[Install]

```
vim /etc/systemd/system/xxx.service
[Unit] # 主要是服务说明
Description=test # 简单描述服务
#After=network.target # 描述服务类别, 表示本服务需要在network服务启动后在启动
#Before=xxx.service # 表示需要在某些服务启动之前启动, After和Before字段只涉及启动顺序, 不涉及依赖关系。

[Service] # 核心区域
Type=forking # 表示后台运行模式。
User=user # 设置服务运行的用户
Group=user # 设置服务运行的用户组
KillMode=control-group # 定义systemd如何停止服务
PIDFile=/usr/local/test/test.pid # 存放PID的绝对路径
Restart=no # 定义服务进程退出后, systemd的重启方式, 默认是不重启
ExecStart=/usr/local/test/bin/startup.sh # 服务启动命令, 命令需要绝对路径
PrivateTmp=true # 表示给服务分配独立的临时空间

[Install]
WantedBy=multi-user.target # 多用户
```

php安装composer

```
yum -y update
```

```
yum install php-cli php-zip wget unzip
```

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
```

```
php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

