Linux运维

- 网络运维
  - linux无ifconfig命令下查看IP地址
- 常用命令
  - linux下通过内置命令生成随机字符串
  - curl测试网站响应速度
- 系统备份
  - lsyncd定时备份
- 安全审计
  - 给shell命令执行命令添加记录时间
- 常规配置
  - .bash_profile和.bashrc的区别
- 硬盘管理
  - ubuntu查看剩余空间并扩充到文件系统
- Macos应用程序代码签名

网络运维

网络运维

linux无ifconfig命令下查看IP地址

docker镜像一般为了减少镜像空间都会减少不必要的工具，`ifconfig`命令可能都没安装，如果不想另外再安装`ifconfig`可以使用以下命令查看容器IP

```
ip addr show
```

常用命令

常用命令

linux下通过内置命令生成随机字符串

```
# 生成数字加大小写英文字母随机字符串
tr -dc a-z0-9A-Z </dev/urandom | head -c 16 && echo ''
# 所有字母与数字
tr -dc [:alnum:] </dev/urandom | head -c 16 && echo ''
# 随机数字
tr -dc [:digit:] </dev/urandom | head -c 16 && echo ''
# 所有可打印字符，不包括空格
tr -dc [:graph:] </dev/urandom | head -c 16 && echo ''
```

参考资料：

- CSDN:linux 命令：tr 详解

常用命令

curl测试网站响应速度

```
curl "-w" "time_namelookup: %{time_namelookup}\ntime_connect: %{time_connect}\ntime_starttransfer: %{time_starttransfer}\ntime_total: %{time_total}\n" <请求地址>
```

变量名称	作用
time_namelookup	域名解析时间，用来排查是否为dns解析导致请求慢的原因
time_connect	建立TCP连接所花费时间
time_starttransfer	从请求开始到响应开始传输的时间
time_total	总的花费时间

参考资料：

- Cizixs Write Here — 使用 curl 命令分析请求的耗时情况

系统备份

系统备份

lsyncd定时备份

前段时间2TB的固态硬盘突然坏了，才想起硬盘还是比较容易坏。还好通过diskgenius找回了大部分关键数据，从此还是要养成备份的习惯。准备一个U盘，将一些重要的数据定时备份到U盘中，这样多一份保险。

当前环境

- ubuntu server-22
- lsyncd-2.2.3

一、安装rsync+lsyncd

```
apt install rsync lsyncd
```

二、配置同步备份目录

添加备份目录直接添加`sync`配置就行，可能不同发行版本配置文件路径会不一样，具体路径请参考`/etc/init.d/lsyncd`文件

```
# apt安装时默认不存在，需要手动创建
mkdir /etc/lsyncd
# 以下命令将覆盖文件内容，执行前请确保该文件不存在
cat <<EOF > /etc/lsyncd/lsyncd.conf.lua
settings {
  logfile = "/var/log/lsyncd/lsyncd.log",
  statusFile = "/var/log/lsyncd/lsyncd.status",
  pidfile = "/var/run/lsyncd.pid",
  statusInterval = 10,
  maxProcesses = 1,
  maxDelays = 100,
}

sync {
  default.rsync,
  source = "<source-1>",
  target = "<target-1>",
  excludeFrom="/etc/lsyncd/exclude",
}

sync {
  default.rsync,
  source = "<source-2>",
  target = "<target-2>",
  excludeFrom="/etc/lsyncd/exclude",
}


EOF
# 生成不需要同步文件配置
cat <<EOF > /etc/lsyncd/exclude
*.swp
_.bak
_.tmp
EOF
```

三、启动服务

```
# 启动服务
sudo systemctl start lsyncd
# 注册开机服务
sudo systemctl enable lsyncd
```

三、配置日志轮询切割

为了防止日志过大，添加日志轮询切割配置

```
cat <<EOF > /etc/logrotate.d/lsyncd
/var/log/lsyncd/lsyncd.log {
  weekly
  missingok
  notifempty
  maxsize 5M
  rotate 14
  delaycompress
  # create 0640 rsync root
  sharedscripts
  postrotate
    [ ! -f /var/run/lsyncd.pid ] || kill -USR2 `cat /var/run/lsyncd.pid`
  endscript
}
EOF
```

参考资料：

- 【CSDN】 - lsyncd 配合 rsync 实时差异同步节点文件
- 【官网】 - Config Layer 4: Default Config

安全审计

安全审计

给shell命令执行命令添加记录时间

只对当前用户生效

```
# 以下两条任选一条
echo 'HISTTIMEFORMAT="%F %T "' >> ~/.bashrc
#
echo 'HISTTIMEFORMAT="%F %T "' >> ~/.bash_profile
```

对所有用户生效

```
echo 'HISTTIMEFORMAT="%F %T "' >> /etc/profile
```

参考资料：

- 【linux.cn】 - 让 history 命令显示日期和时间

常规配置

常规配置

.bash_profile和.bashrc的区别

> `/etc/profile` 此文件为系统的每个用户设置环境信息,当用户第一次登录时,该文件被执行.并从 `/etc/profile.d` 目录的配置文件中搜集shell的设置。
> `/etc/bashrc` 为每一个运行bash shell的用户执行此文件.当bash shell被打开时,该文件被读取。
> `~/.bash_profile` 每个用户都可使用该文件输入专用于自己使用的shell信息,当用户登录时,该文件**仅仅执行一次**! 默认情况下,他设置一些环境变量,执行用户的 `.bashrc` 文件。
> `~/.bashrc` 该文件包含专用于你的bash shell的bash信息,当登录时以及每次打开新的shell时,该该文件被读取。
> `~/.bash_logout` 少见，但是意味着当每次退出系统(退出bash shell)时,执行该文件。
>
> 另外 `/etc/profile` 中设定的变量(全局)的可以作用于任何用户, 而 `~/.bashrc` 等中设定的变量(局部)只能继承 `/etc/profile` 中的变量,他们是"父子"关系。
>
> `profile` 用于登录式shell, 而 `bashrc` 用于每个交互式shell
> `~/.bash_profile` 是交互式、login 方式进入 bash 运行的
> `~/.bashrc` 是交互式 non-login 方式进入 bash 运行的
> 通常二者设置大致相同，所以通常前者会调用后者。
> 所以一般优先把变量设置在 `.bashrc` 里面。比如在crontab里面执行一个命令， `.bashrc` 设置的环境变量会生效，而 `.bash_profile` 不会。

参考资料：

【简书】 - .bash_profile和.bashrc的区别

硬盘管理

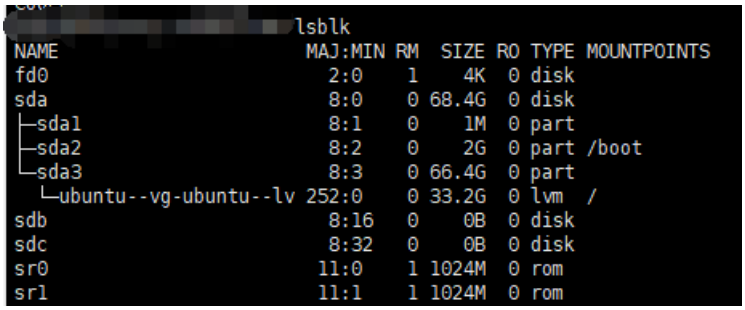
硬盘管理

ubuntu查看剩余空间并扩充到文件系统

> 操作系统：ubuntu-22.04.4

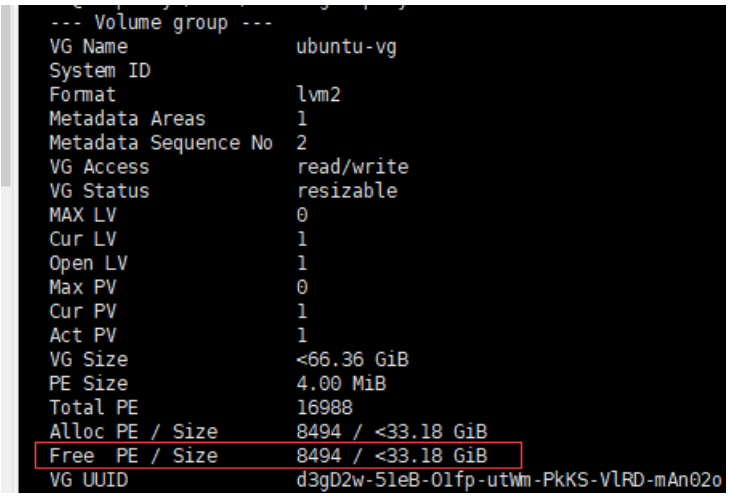
查看硬盘分区情况

```
lsblk
```



查看卷组信息

```
vgdisplay
```



执行扩容

```
# 扩充所有空闲空间
lvextend -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
# 执行变更
resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
```

参考资料

- 【CSDN】 - ubuntu系统将磁盘剩余容量扩到文件目录上

Macos应用程序代码签名

代码签名需要到苹果开发者中心订阅688/年的功能，

1. 创建Developer Application ID的证书（此操作只能账号持有者或管理员权限的人操作，开发者是没有权限）
2. 导出p12格式的证书
3. 在ci的机器上导入p12格式的证书，最好是单独建一个keychain
4. 每次构建前先解锁keychain

在gitlab-ci中使用代码签名

```
# 钥匙串访问密码
KEYCHAIN_PASSWORD=""
CERT_PATH="certs/mac_signing.p12"
CERT_PASS=""

#1️⃣创建 keychain（仅首次）
security create-keychain -p "${KEYCHAIN_PASSWORD}" gitlab-ci.keychain

#2️⃣禁止自动锁定（或设置超时时间）
security set-keychain-settings -lut 0 gitlab-ci.keychain

#3️⃣解锁
security unlock-keychain -p "${KEYCHAIN_PASSWORD}" gitlab-ci.keychain

#4️⃣导入证书（仅首次）
security import ${CERT_PATH} -k gitlab-ci.keychain -P "${CERT_PASS}" -T /usr/bin/codesign

#5️⃣授权 codesign 访问私钥
security set-key-partition-list -S apple-tool:,apple: -s -k "${KEYCHAIN_PASSWORD}" gitlab-ci.keychain
```

参考资料：

- 【稀土掘金】Electron 签名和公证