



Linux 操作系统实训

王良明



目录

1 基础实训.....	1
1.1 安装虚拟机软件	1
1.2 安装 Linux	3
1.2.1 安装 CentOS 9	4
1.2.2 安装扫尾	12
1.3 创建用户和组	18
1.4 硬盘分区	20
1.4.1 对已用硬盘分区	20
1.4.2 对新硬盘分区.....	21
1.4.3 对 loop 存储设备分区	22
1.5 常用命令用法	24
1.6 vim 实训	30
1.7 systemctl 命令	33
2 服务实训.....	37
2.1 时钟同步服务	38
2.2 FTP 服务	40
2.3 DHCP 服务	45
2.4 DNS 服务	50
2.4.1 域名解析	50
2.4.2 资源记录	54
2.4.3 安装和配置 DNS	55
2.5 网站服务	58
2.5.1 网站简介	58
2.5.2 搭建过程	59

2.6 网络文件系统 NFS	67
2.6.1 NFS 介绍	67
2.6.2 搭建 NFS	67
2.7 搭建邮件系统	71
2.7.1 邮件系统简介	71
2.7.2 搭建邮件系统	71
2.7.3 使用邮件系统	72
2.8 SAMBA 服务	77
2.9 防火墙	80
2.9.1 基础知识	80
2.9.2 内核中的 netfilter 模块	81
2.9.3 配置规则	82
2.9.4 实战案例	88
2.10 Docker 容器	92
2.10.1 安装	95
2.10.2 体验一遍	95
2.10.3 定制镜像	96
2.10.4 构建镜像	98
2.10.5 使用外部存储	99
2.10.6 容器组合	101
2.10.7 容器网络	103
2.11 虚拟机	108

1 基础实训

1.1 安装虚拟机软件

从 VMWare 官网 <https://www.vmware.com/go/getworkstation-win> 下载 VMware-workstation-full-17.0.0-20800274.exe, 然后双击下载的文件安装虚拟机软件。



这样就在电脑桌面上多了一个图标，双击它启动，然后点击菜单“帮助”》“输入许可证密钥”，尝试使用下面的激活密钥：4A4RR-813DK-M81A9-4U35H-06KND、NZ4RR-FTK5H-H81C1-Q30QH-1V2LA、JU090-6039P-08409-8J0QH-2YR7F、4Y09U-AJK97-089Z0-A3054-83KLA、4C21U-2KK9Q-M8130-4V2QH-CF810、MC60H-DWHD5-H80U9-6V85M-8280D。

VMware 是商业软件，需要购买。另一个开源的虚拟机软件是 virtualbox，可以直接从官网 (<https://www.virtualbox.org/wiki/Downloads>) 下载使用。

启动 VMWare 虚拟机软件后，按 Ctrl+n 新建一台虚拟机，一直点击“下一步”直到出现如图 1-1 左上角所示的画面。

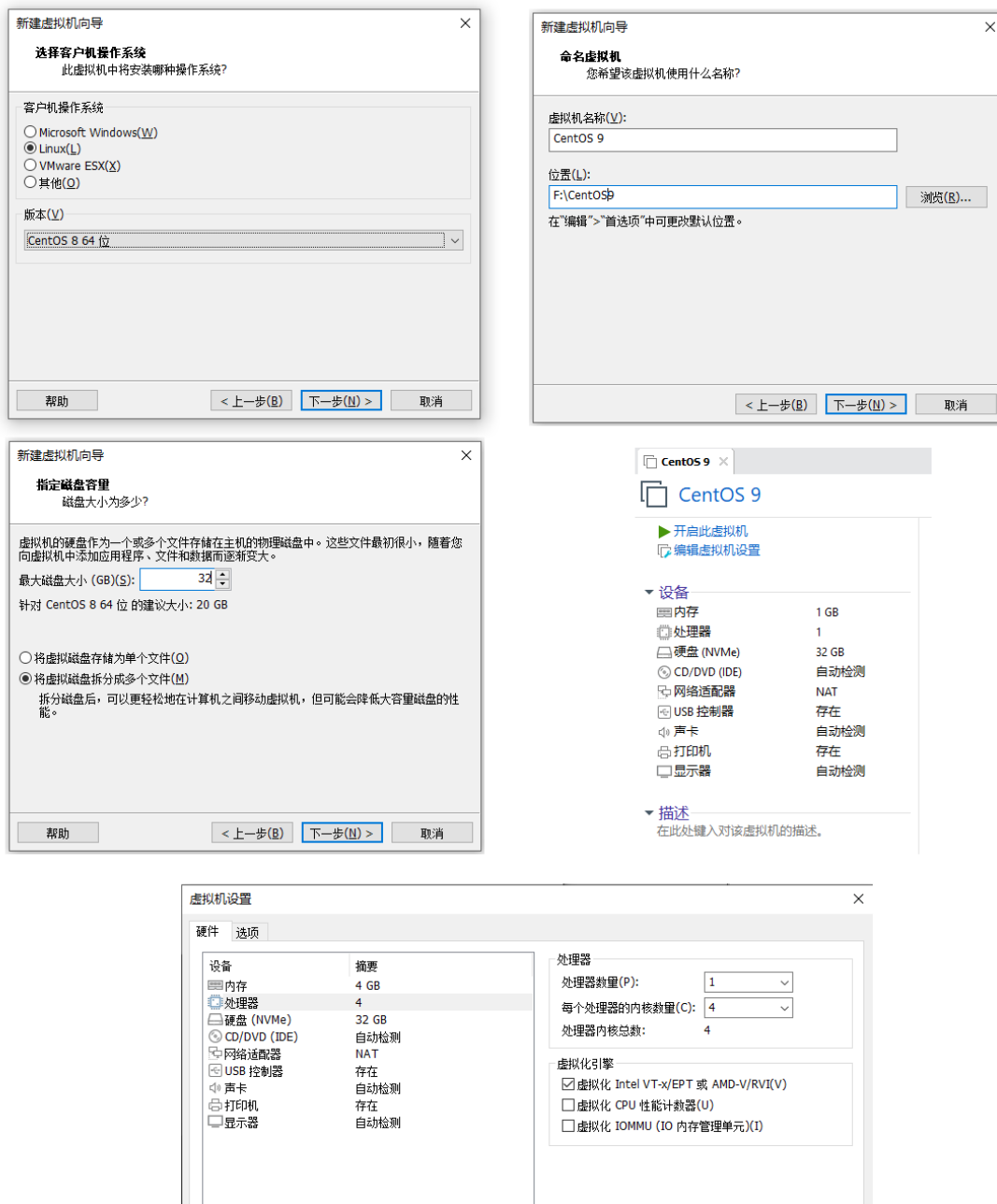


图 1-1 新建虚拟机

接下来的几个步骤参见图 1-1 操作。这里选择虚拟机里要安装 Linux 操作系统 CentOS 9 64 位，如果没有没 9 就选择 8 版本。不过即使这里选择的操作系统错了，也不会影响以后在虚拟机里安装 Linux。虚拟机的内存 2GB 以上，cpu 核心 2 个以上，且 cpu 要开启虚拟化功能（参见图 1-1 最下部分）。

1.2 安装 Linux

准备工作：一台计算机或虚拟机（在 BIOS 里配置先从光盘启动），安装光盘或 ISO 安装镜像文件（虚拟机安装可以采用），最好接上网线并保证能连上 Internet。

从 http://mirror.stream.centos.org/9-stream/BaseOS/x86_64/iso/CentOS-Stream-9-latest-x86_64-boot.iso 下载 CentOS 9。

国内可以从南京大学下载，速度很快：http://mirror.nju.edu.cn/centos-stream/9-stream/BaseOS/x86_64/iso/CentOS-Stream-9-latest-x86_64-boot.iso

首先新建一台虚拟机 CentOS 9，参考图 1-2。



图 1-2 新建虚拟机

注意：下面的安装界面上的按钮上出现的字母表示快捷键，同时按 **Alt** 和字母相当于单击了那个按钮。

1.2.1 安装 CentOS 9

首先把下载的镜像文件放入虚拟机的光驱里。启动虚拟机后出现如图 1-3 所示的画面，在里面单击一下鼠标，然后使用光标键选择第一项，即“Install CentOS Stream 9”并按 Enter 键进入下一画面。

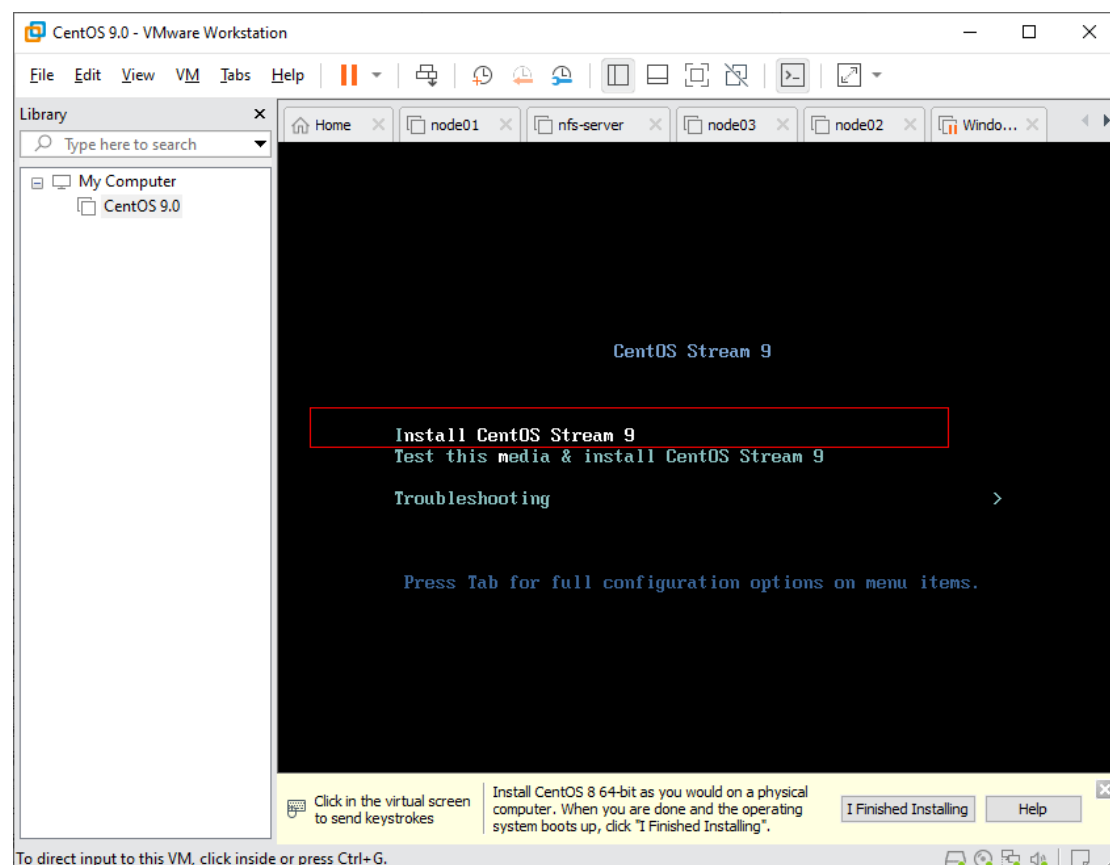


图 1-3 安装 CentOS 9

这时鼠标被虚拟机捕获，如果要让虚拟机释放鼠标，就按键盘的 Ctrl+Alt 键。接下来选中语言，参见图 1-4 所示。

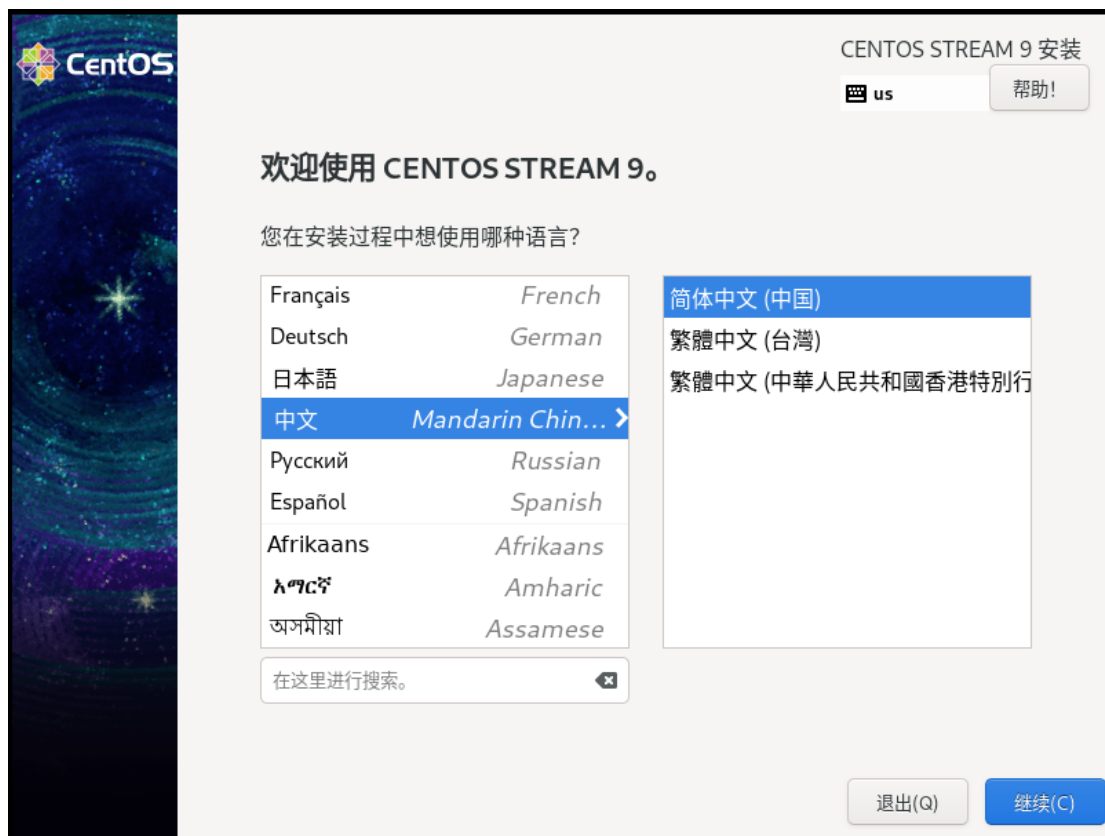


图 1-5 选择中文语言

单击“继续”后出现如图 1-6 所示的安装摘要画面。



图 1-6 安装摘要

单击“安装目的地”后出现图 1-7 所示的画面，点选“自定义”，接下来我们来手工创建分区。



图 1-7 安装目标位置

单击“完成”后，在“新挂载点将使用以下分区方案”下面选择“标准分区”，参考图 1-8 画面。



图 1-8 手动分区画面

单击左下角的“+”添加一个分区，我们接下来创建两个分区：交换区和根分区，见图 1-9 所示，左侧是建交换区，右侧是建根分区。



图 1-9 创建分区

建好分区后的画面见图 1-10，在这里可以修改分区、删除分区。



图 1-10 创建了两个分区

单击左上角的“完成”，出现图 1-11 的警告画面，直接单击“接受更改”。



图 1-11 警告

此时又回到图 1-6 所示的安装摘要画面，单击“软件选择”，出现图 1-12 所示的画面，点

选第一项（Server with GUI），在右侧可以再选择一些需要额外安装的软件，如图形管理工具、远程桌面客户端等，当然这些软件以后可随时安装。默认是带 GUI 的服务器，最小安装是没有图形界面的。然后单击“完成”。

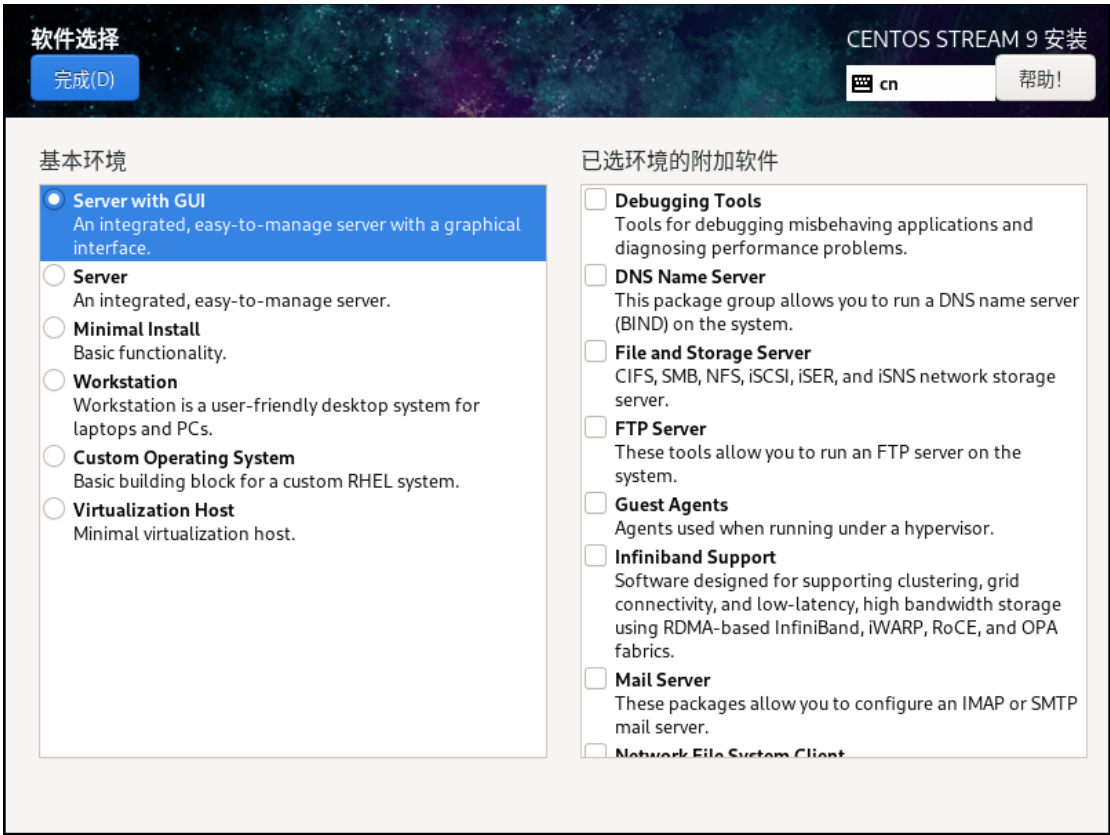


图 1-12 选择安装的软件

此时再次回到图 1-6 所示的安装摘要画面，然后单击“时间和日期”设置上海时区，接下来在图 1-6 所示的画面上选择“root 密码”，然后参考图 1-13 左侧部分所示给超级用户 root 设置密码，在图 1-6 所示的画面上选择“创建用户”，创建一个普通的工作用户 moodisk。



图 1-13 给 root 设置密码并创建普通用户 moodisk

最后直接单击“开始安装”出现图 1-14 所示的画面。

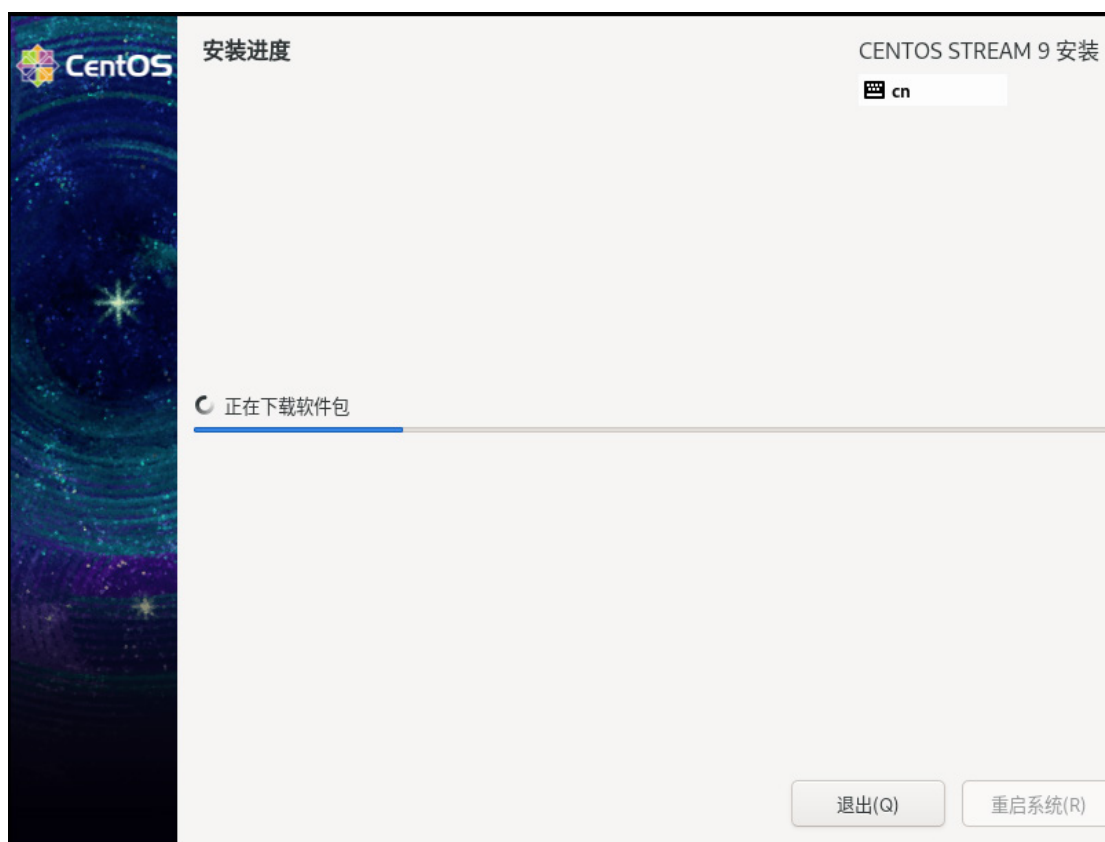


图 1-14 开始安装

整个安装过程大概 20 分钟，安装完后单击“重启系统”按钮，重启后出现图 1-15 所示的登录画面。到此 CentOS 9 安装完毕。

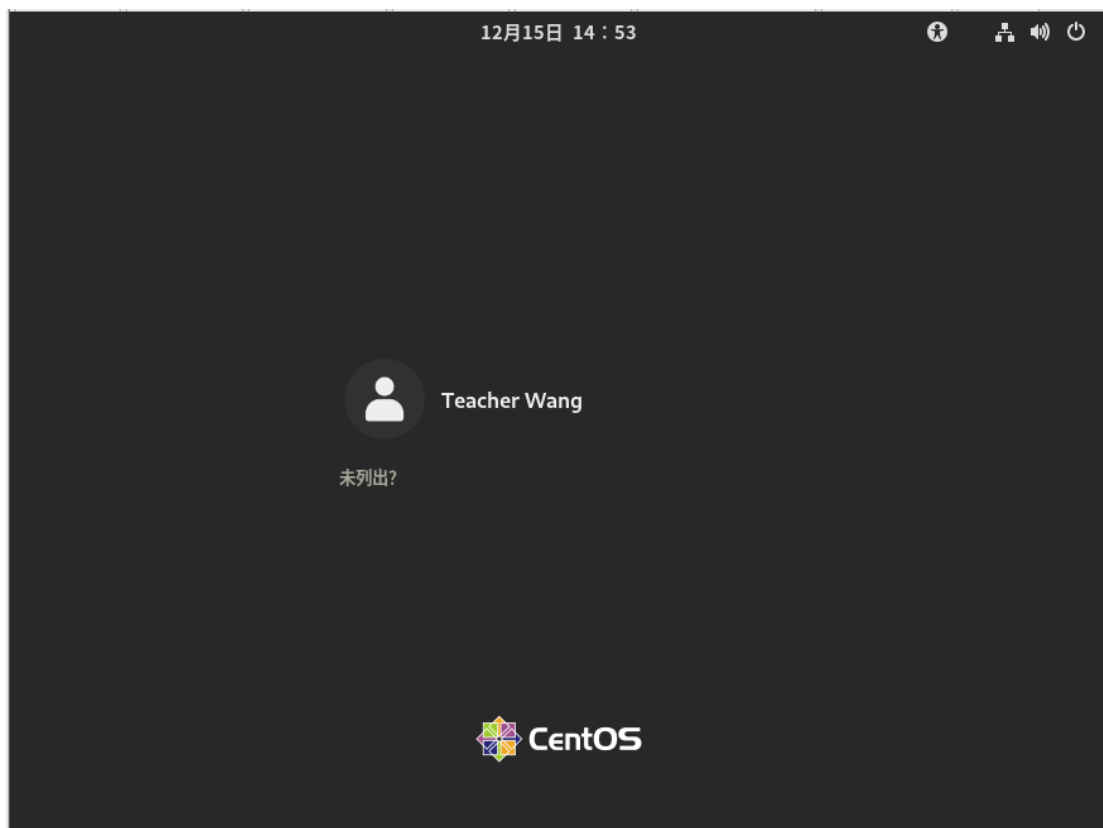


图 1-15 登录画面

如果要用 root 登录，单击图 1-15 中的“未列出？”，然后输入 root 账号和密码即可登录了。

1.2.2 安装扫尾

登录之后看到的画面如图 1-16 所示。

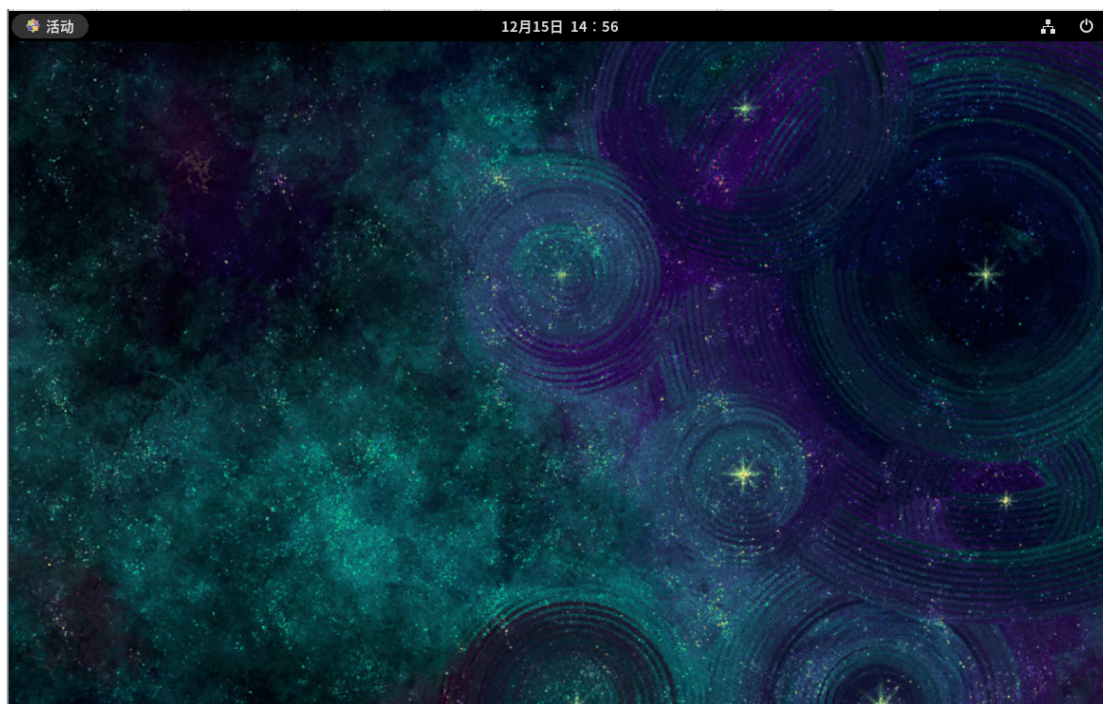


图 1-16 CentOS 9 的图形桌面

点击“活动”，再点击屏幕下面的“终端”图标，打开一个命令终端窗口，参考图 1-17 所示。

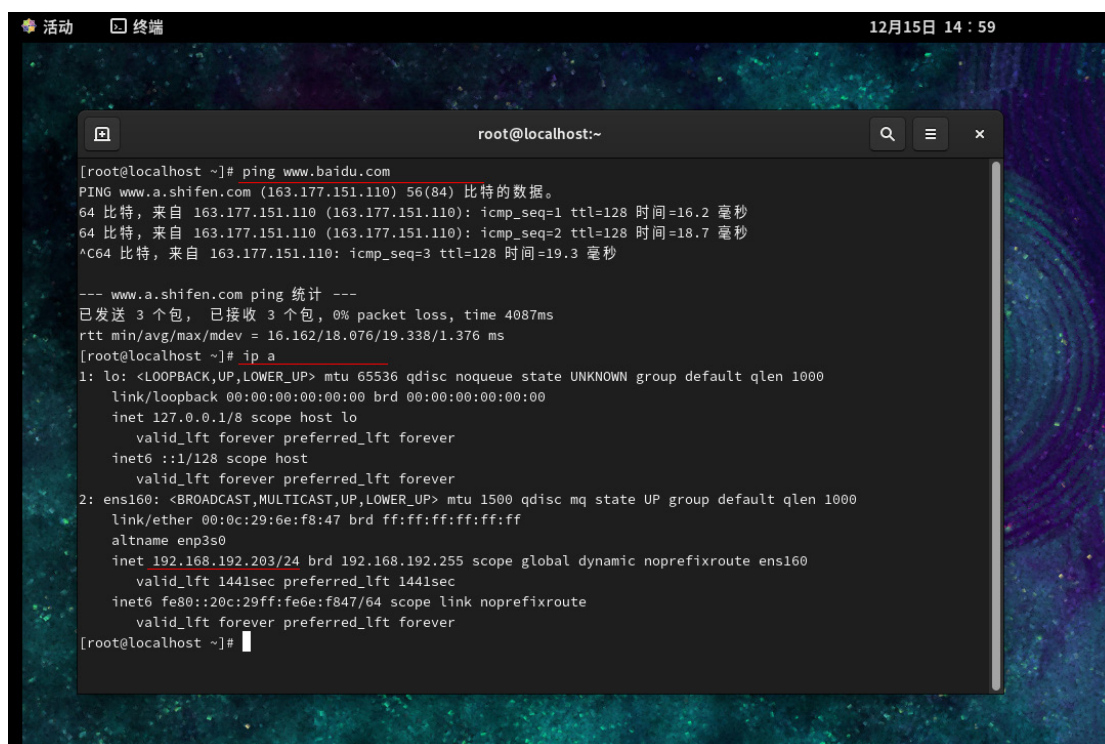


图 1-17 打开命令窗口

输入命令 `ping www.baidu.com` 测试网络是否接通互联网，再输入命令 `ip a` 查看网卡的 `ip` 地址，比如我的虚拟机的网卡 `ip` 为 `192.168.192.203`。

从 windows 登录到虚拟机中，在 windows 这边按 `win+r`，输入 `cmd` 并回车，打开 windows 的命令窗口，参考图 1-18 所示。

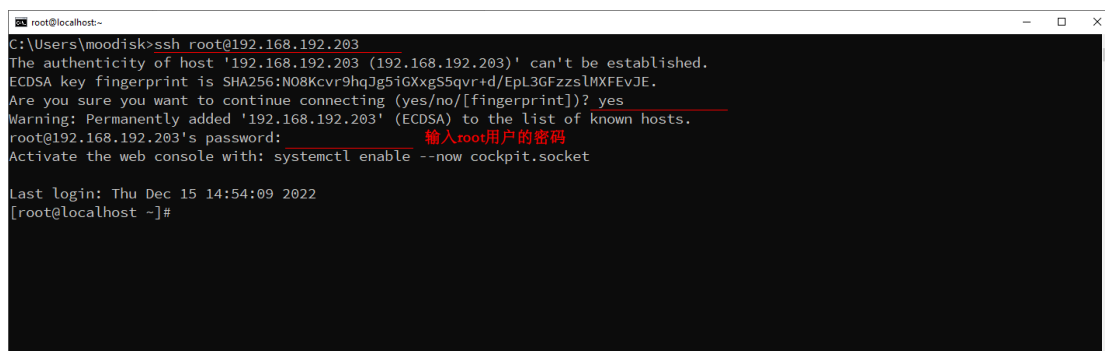


图 1-18 windows 的命令窗口

输入命令 `ssh root@192.168.192.203` 命令登录虚拟机，参考图 1-18。

一些扫尾工作：

```
[root@localhost ~]# dnf upgrade -y #操作系统打上最新的补丁
[root@localhost ~]# systemctl --now disable firewalld #禁用防火墙
[root@localhost ~]# vim /etc/sysconfig/selinux #禁用 SELinux
.....
SELINUX=disabled #把 enforcing 改为 disabled
.....
```

如果要配置静态 `ip` 地址，参考下面的 命令：

```
[root@localhost ~]# vim /etc/NetworkManager/system-connections/ens160.nmconnection
.....
[ipv4]
method=manual #manual-静态, auto-动态
address1=192.168.192.20/24,192.168.192.2 #ip 地址/掩码位数[,网关]
dns=202.96.128.86;8.8.8.8 #中间用分号隔开
.....
```

修改了网卡配置文件，可执行命令 `nmcli c reload;nmcli c up ens160` 使得新地址起作用。DNS 与 windows 一样，你可执行 `ipconfig /all` 查看 windows 的 DNS，至于 NAT 网络的静态 `ip` 地址范围、网关和掩码，请参考图 1-19 查看。

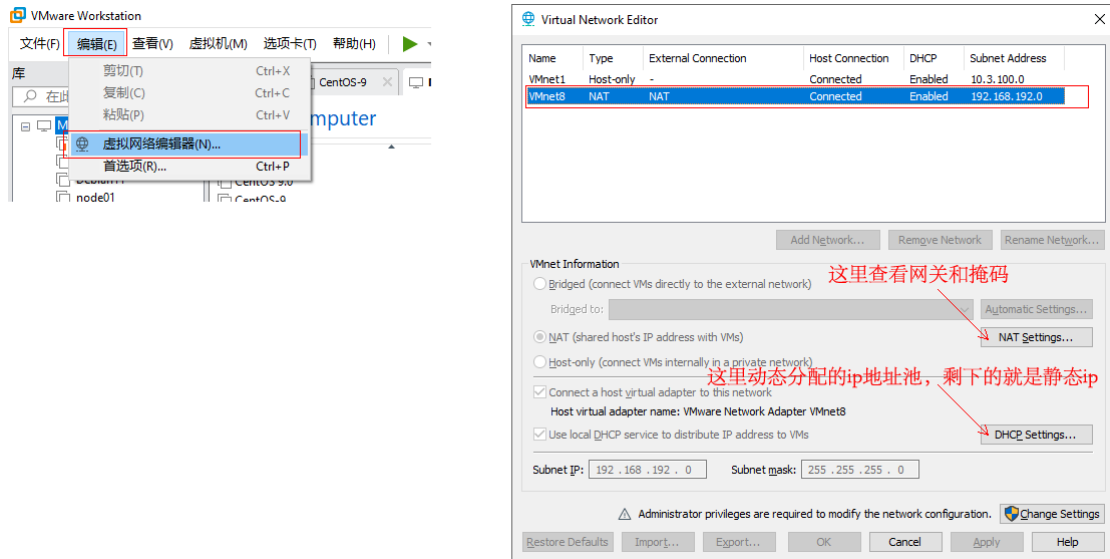


图 1-19 查看 NAT 网络参数

最后关机：

```
[root@localhost ~]# shutdown -h now #关机
```

由于后面的实训需要用到几台虚拟机，所以现在先克隆一台出来，参考图 1-20。

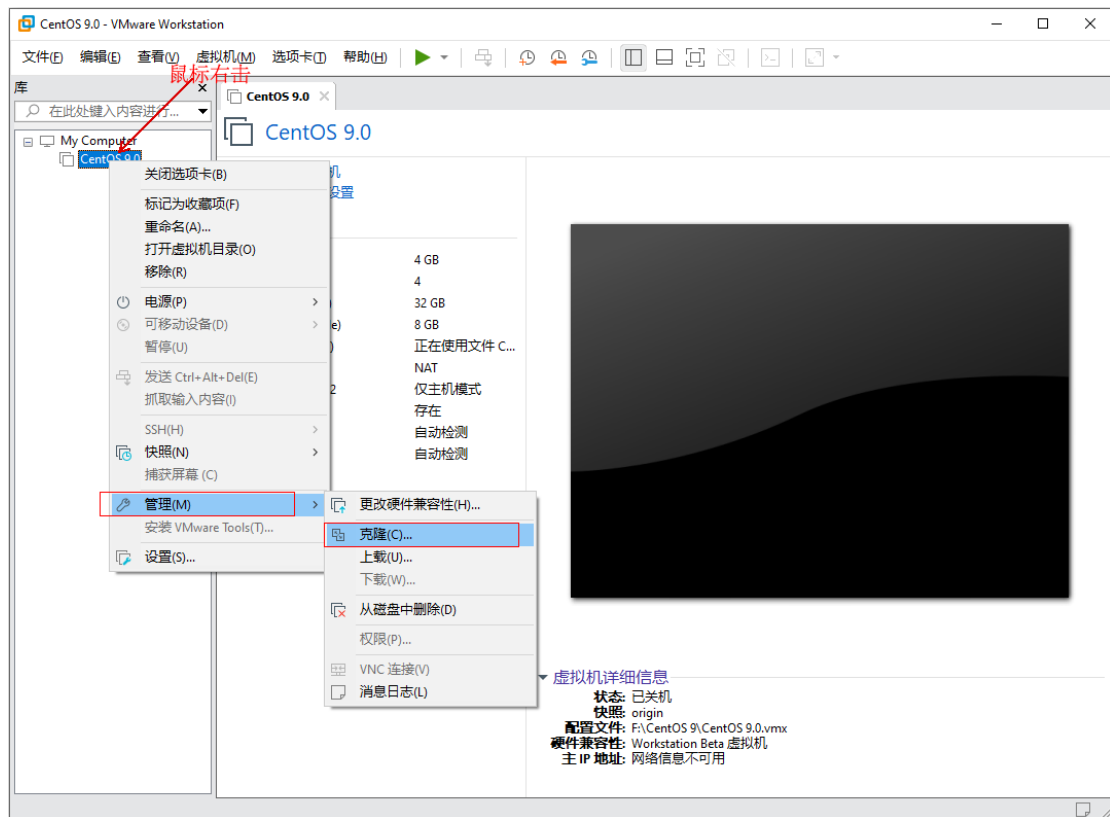


图 1-20 克隆虚拟机

然后创建一个快照，这样以后可随时回滚到干净的原始状态。参考图 1-21 操作。

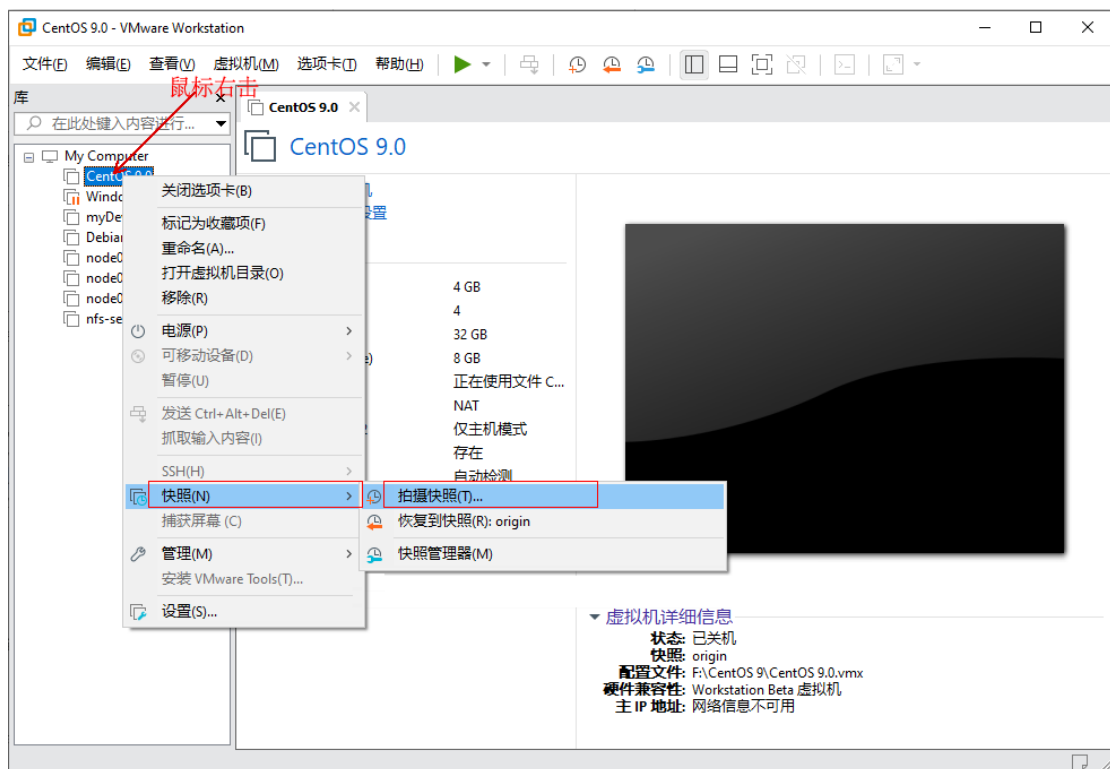


图 1-21 创建快照

1.3 创建用户和组

只有超级用户 `root` 才有权限创建用户和组，以 `root` 登录 Linux 图形界面，打开一个终端，或者远程登录到虚拟机。

实训 1：通过实操完成表 1-1 中空白部分或者问答题。

表 1-1 创建用户和组练习

序号	命 令	说 明
1	<code>groupadd class1</code>	
2		创建组 <code>class2</code> ，组的 ID 号为 1101
3	<code>groupadd -o -g 1101 class3</code>	
4	<code>tail /etc/group</code>	组 <code>class1</code> 的组号是
5		
6	<code>groupmod -g 1102 class1</code>	
7		修改组 <code>class2</code> 的组名为 <code>grade2</code>
8	<code>useradd -u 1200 -d /home/abc -s /bin/sh -g class1 -G grade2 zsan</code>	
9		新建用户 <code>lisi</code> ，登录 SHELL 是 <code>/bin/bash</code> 。
10	<code>useradd -g 1101 -c "wang er" wanger</code>	
11	<code>useradd -G class1, grade2 wlm</code>	
12	<code>useradd user1</code>	
13	<code>useradd -o -u 0 root2</code>	
14	<code>tail /etc/passwd</code>	新建用户的 ID 号分别是：

15	<code>id root2</code>	UID= , GID= , 附加组=
16	<code>tail /etc/group</code>	组 <code>grade2</code> 里成员用户有:
17	<code>usermod -d /opt/wlm -u 1210 wlm</code>	
18		修改 <code>user1</code> 的备注为 “Test user”, 主要组群为 <code>grade2</code>
19	<code>usermod -l aliasroot root2</code>	
20	<code>passwd zsan</code>	
21		修改用户 <code>wlm</code> 的密码
22	<code>passwd -d user1</code>	
23	<code>passwd -l aliasroot</code>	
24		解锁用户 <code>aliasroot</code>
25	切换到其他字符屏幕, 并测试分别用账号 <code>zsan</code> , <code>wlm</code> , <code>user1</code> 登录	屏幕切换热键 <code>Ctrl + Alt + F1, F2, F3, ...</code>
26		删除用户 <code>user1</code> , 用户家目录一并删除
27		删除用户 <code>wlm</code>
28	<code>userdel zsan</code>	
29		删除组 <code>class1</code> 。注意: 只能删除没有成员的空组
30		删除组 <code>class3</code>
31	<code>groupdel grade2</code>	能成功删除组 <code>grade2</code> 吗? 如果不能, 原因是什么? 如何删除它?

实训 2: 创建组 `sales`; 创建用户 `alice`, 要求 `alice` 加入 `sales` 组; 给 `alice` 设置密码; 用 `alice` 远程登录到虚拟机。

1.4 硬盘分区

1.4.1 对已用硬盘分区

采用下面的命令列出全部的硬盘以及每个硬盘的分区：

```
fdisk -l
```

硬盘参数包括总容量（多少 G、字节和扇区数），见图 1-21 所示。

```
Select root@localhost:~
[root@localhost ~]# fdisk -l
Disk /dev/nvme0n1: 32 GiB, 34359738368 字节, 67108864 个扇区
磁盘型号: VMware Virtual NVMe Disk
单元: 扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理): 512 字节 / 512 字节
I/O 大小(最小/最佳): 512 字节 / 512 字节
磁盘标签类型: dos
磁盘标识符: 0x329f476d

设备            启动      起点      末尾      扇区    大小 Id 类型
/dev/nvme0n1p1      2048    3907583    3905536    1.9G 82 Linux swap / Solaris
/dev/nvme0n1p2 *    3907584 41943039 38035456    18.1G 83 Linux
[root@localhost ~]#
```

图 1-21 查看硬盘分区

从上图中可以看出，硬盘还有约 10GB 的空闲空间，下面在这个空闲区域里面新建分区。

```
[root@localhost ~]# fdisk /dev/nvme0n1                                #对硬盘/dev/nvme0n1 进行分区操作
Command (m for help):
```

等待输入命令，按 m 显示全部可用的命令，常用命令有 p:显示分区表，n:新建分区，d:删除分区，w:存盘，q:退出 fdisk。

```
Command (m for help):p                                              #先看看已有分区表信息
Disk /dev/nvme0n1: 32 GiB, 34359738368 字节, 67108864 个扇区
磁盘型号: VMware Virtual NVMe Disk
单元: 扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理): 512 字节 / 512 字节
I/O 大小(最小/最佳): 512 字节 / 512 字节
磁盘标签类型: dos
磁盘标识符: 0x329f476d

设备            启动      起点      末尾      扇区    大小 Id 类型
/dev/nvme0n1p1      2048    3907583    3905536    1.9G 82 Linux swap / Solaris
/dev/nvme0n1p2 *    3907584 41943039 38035456    18.1G 83 Linux

Command (m for help):n                                              #开始新建一个分区
分区类型
  p  主分区 (2 primary, 0 extended, 2 free)
  e  扩展分区 (逻辑分区容器)
选择 (默认 p):                                                    #输入 p 表示新建一个主分区

将使用默认回应 p。
分区号 (3,4, 默认 3):
第一个扇区 (41943040-67108863, 默认 41943040):
```

```

最后一个扇区, +/-sectors 或 +size{K,M,G,T,P} (41943040-67108863, 默认 67108863):

创建了一个新分区 3, 类型为“Linux”, 大小为 12 GiB。

Command (m for help):p                                     #再显示一下分区表信息, 发现多了一个
.....
设备          启动          起点          末尾          扇区    大小 Id 类型
/dev/nvme0n1p1      2048      3907583      3905536      1.9G 82 Linux swap / Solaris
/dev/nvme0n1p2 *    3907584      41943039      38035456      18.1G 83 Linux
/dev/nvme0n1p3      41943040      67108863      25165824      12G 83 Linux

Command (m for help):w                                     #保存分区表信息并退出 fdisk 命令

[root@localhost ~]# partprobe                               #通知内核分区表已经改变, 如果报错, 就干脆重启计算机
[root@localhost ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sr0              11:0    1   855M  0 rom
nvme0n1          259:0    0    32G  0 disk
├─nvme0n1p1      259:1    0    1.9G  0 part [SWAP]
├─nvme0n1p2      259:2    0   18.1G  0 part /
└─nvme0n1p3      259:3    0    12G  0 part

```

格式化成 ext4 文件系统:

```

[root@localhost ~]# mkfs.ext4 /dev/nvme0n1p3
mke2fs 1.46.5 (30-Dec-2021)
创建含有 3145728 个块 (每块 4k) 和 786432 个 inode 的文件系统
文件系统 UUID: 11ed9cd6-cc18-46a0-a680-68d4df4cef2f
超级块的备份存储于下列块:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

正在分配组表: 完成
正在写入 inode 表: 完成
创建日志 (16384 个块) 完成
写入超级块和文件系统账户统计信息: 已完成

```

挂载分区:

```

[root@localhost ~]# mount /dev/nvme0n1p3 /mnt                #挂载到/mnt 空目录上
[root@localhost ~]# cp /etc/profile /mnt                    #往分区拷贝文件
[root@localhost ~]# df -T
文件系统      类型      1K-块    已用    可用 已用% 挂载点
devtmpfs      devtmpfs    4096         0    4096    0% /dev
tmpfs         tmpfs      1992224         0   1992224    0% /dev/shm
tmpfs         tmpfs       796892    9784    787108    2% /run
/dev/nvme0n1p2 xfs       19007488 5014776 13992712   27% /
tmpfs         tmpfs       398444     52    398392    1% /run/user/42
tmpfs         tmpfs       398444     36    398408    1% /run/user/0
/dev/nvme0n1p3 ext4      12278920     24   11633368    1% /mnt
[root@localhost ~]# umount /mnt                             #卸载分区

```

如果想每次开机自动挂载, 那么操作以下几步:

```

[root@localhost ~]# mkdir -p /opt/data
[root@localhost ~]# echo "/dev/ nvme0n1p3 /opt/data ext4 defaults 1 1" >>/etc/fstab
[root@localhost ~]# mount -a 或者 reboot                    #然后再用 df -T 命令可以看到已经挂载了

```

1.4.2 对新硬盘分区

首先关闭虚拟机, 增加一块 8GB 的硬盘, 然后启动虚拟机, 先看看新加的硬盘是什么:

```

[root@localhost ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sr0              11:0    1   855M  0 rom
nvme0n1          259:0    0    32G  0 disk
├─nvme0n1p1      259:1    0    1.9G  0 part [SWAP]

```

```
└─nvme0n1p2 259:2    0 18.1G 0 part /
└─nvme0n1p3 259:3    0 12G 0 part
nvme0n2      259:4    0 8G 0 disk
```

新加的硬盘为 `nvme0n2`，下面对 `/dev/nvme0n2` 进行分区。

```
[root@localhost ~]# fdisk /dev/nvme0n2
.....
命令(输入 m 获取帮助): n                                #新建分区
分区类型
  p  主分区 (0 primary, 0 extended, 4 free)
  e  扩展分区 (逻辑分区容器)
选择 (默认 p): p                                          #主分区
分区号 (1-4, 默认 1):
第一个扇区 (2048-16777215, 默认 2048):
最后一个扇区, +/-sectors 或 +size{K,M,G,T,P} (2048-16777215, 默认 16777215): +4G      #分区为 4G

创建了一个新分区 1, 类型为 "Linux", 大小为 4 GiB。

命令(输入 m 获取帮助): p                                #显示分区表
.....
设备            启动   起点   末尾   扇区 大小 Id 类型
/dev/nvme0n2p1    2048 8390655 8388608   4G 83 Linux

命令(输入 m 获取帮助): w                                #存盘并退出分区命令
```

通知内核分区表已经改变，如果报错，就干脆重启计算机：

```
[root@localhost ~]# partprobe
[root@localhost ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sr0          11:0    1 855M  0 rom
nvme0n1      259:0    0   32G  0 disk
└─nvme0n1p1 259:1    0   1.9G  0 part [SWAP]
└─nvme0n1p2 259:2    0  18.1G  0 part /
└─nvme0n1p3 259:3    0   12G  0 part
nvme0n2      259:4    0    8G  0 disk
└─nvme0n2p1 259:6    0    4G  0 part
```

然后参照前面的方法对分区 `/dev/nvme0n2p1` 做格式化和挂载操作。

1.4.3 对 loop 存储设备分区

如果计算机硬盘没有空闲空间又不能新增硬盘时，可以采取 `loop` 存储设备来做分区实验。

(1) 加载 `loop` 内核模块：

```
[root@localhost ~]# modprobe loop max_part=8              #加载 loop 模块
[root@localhost ~]# echo loop >> /etc/modules-load.d/modules #启动时自动加载
```

(2) 创建一个全 0 且 50MB 的文件 `hdisk`

```
[root@localhost ~]# dd if=/dev/zero of=hdisk bs=1024 count=50000
记录了 50000+0 的读入
记录了 50000+0 的写出
51200000 字节 (51 MB, 49 MiB) 已复制, 0.0484218 s, 1.1 GB/s
```

(3) 设置 `loop` 设备 `loop0`，指向 `hdisk` 文件：

```
[root@localhost ~]# losetup /dev/loop0 hdisk
```

```
[root@localhost ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0        7:0      0 48.8M 0 loop
sr0          11:0     1 855M 0 rom
nvme0n1      259:0    0   32G 0 disk
├─nvme0n1p1 259:1    0   1.9G 0 part [SWAP]
├─nvme0n1p2 259:2    0  18.1G 0 part /
└─nvme0n1p3 259:3    0   12G 0 part
nvme0n2      259:4    0    8G 0 disk
└─nvme0n2p1 259:5    0    4G 0 part
```

(4) 接下来对 `/dev/loop0` 进行分区和格式化:

```
[root@localhost ~]# fdisk /dev/loop0

.....
命令(输入 m 获取帮助): n
分区类型
  p 主分区 (0 primary, 0 extended, 4 free)
  e 扩展分区 (逻辑分区容器)
选择 (默认 p):

将使用默认回应 p。
分区号 (1-4, 默认 1):
第一个扇区 (2048-99999, 默认 2048):
最后一个扇区, +/-sectors 或 +size[K,M,G,T,P] (2048-99999, 默认 99999):

创建了一个新分区 1, 类型为 "Linux", 大小为 47.8 MiB。

命令(输入 m 获取帮助): w

[root@localhost ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0        7:0      0 48.8M 0 loop
├─loop0p1    7:1      0 47.8M 0 part
sr0          11:0     1 855M 0 rom
nvme0n1      259:0    0   32G 0 disk
├─nvme0n1p1 259:1    0   1.9G 0 part [SWAP]
├─nvme0n1p2 259:2    0  18.1G 0 part /
└─nvme0n1p3 259:3    0   12G 0 part
nvme0n2      259:4    0    8G 0 disk
└─nvme0n2p1 259:5    0    4G 0 part
```

需要注意的是分区设备名类似 `/dev/loop0pN`, 其中 `N=1,2,3,4,5, ...`。下面就对分区 `/dev/loop0p1` 进行格式化。

```
[root@localhost ~]# mkfs.xfs /dev/loop0p1 #格式化 xfs 文件系统
[root@localhost ~]# mkdir /opt/disk
[root@localhost ~]# mount /dev/loop0p1 /opt/disk
[root@localhost ~]# df -T
```

文件系统	类型	1K-块	已用	可用	已用%	挂载点
devtmpfs	devtmpfs	4096	0	4096	0%	/dev
tmpfs	tmpfs	1992224	0	1992224	0%	/dev/shm
tmpfs	tmpfs	796892	9800	787092	2%	/run
/dev/nvme0n1p2	xfs	19007488	5021860	13985628	27%	/
tmpfs	tmpfs	398444	52	398392	1%	/run/user/42
tmpfs	tmpfs	398444	36	398408	1%	/run/user/0
/dev/loop0p1	xfs	43504	2904	40600	7%	/opt/disk

此后对目录 `/opt/disk` 操作就是对新建的分区操作了, 使用完了就可以卸载它。

```
[root@localhost ~]# umount /dev/loop0p1
```

1.5 常用命令用法

在表 1-2 中的空白处写上答案。

表 1-2 常用命令练习

序号	命 令	说 明
1	shutdown -h now	
2	exit	
3		重启计算机
4	telinit 1	
5		显示全部历史命令
6	history -c	
7		修改组 class2 的组名为 grade2
8	alias l='ls -la'	
9		显示全部的别名
10		定义 man 的别名 woman
11	unalias l	
12	echo "Hello World!"	
13	echo \$?	
14		获取 sleep 命令的帮助信息
15	help bg	
16		定义用户环境变量 HISTSIZE=5000
17	echo "export OK=1" >> ~/.bashrc	

18	<code>unset OK</code>	
19		显示全部的用户环境变量
20	<code>. ~/.bashrc</code>	
21	<code>date</code>	
22		设置系统时间为 2011 年 7 月 17 日 14:40
23	<code>cal</code>	
24		显示当年整年的日历
25	<code>uname -a</code>	
26		显示已挂载硬盘分区的使用率
27	<code>hostname</code>	
28	<code>du -sh /usr</code>	
29		查看目录/boot 占用磁盘的大小
30	<code>cd /usr/share</code>	
31	<code>cd ../../etc/init.d</code>	
32	<code>cd</code> 或者 <code>cd \$HOME</code> 或者 <code>cd ~</code>	
33	<code>ls -la /boot</code>	
34		列出家目录下的全部文件（包含隐藏文件）
35	<code>ls -R /</code>	
36		进入目录/tmp
37	<code>mkdir abc</code>	
38		创建目录 123
39	<code>mkdir -p a/b/c 1/2/3</code>	

40	cp /etc/profile abc/	
41	ln -s abc/profile 123/linkfile	
42		创建硬链接/tmp/file, 指向/etc/profile
43	cd a/b && touch newfile	
44	cd -	
45	cd 1/2/3 && ls -l / >filelist.txt	
46		强行删除目录/tmp/下的子目录 123
47	more /etc/passwd	
48		查看文件/etc/profile 的后八行
49	head -15 /etc/profile	
50	cat \$HOME/.bashrc	
51	chmod 643 /tmp/1/2/3/filelist.txt	
52		把文件/tmp/1/2/3/filelist.txt 属组改为 fuse
53	chown -R nobody /tmp/abc	
54	mv /tmp/{abc, cba}	
55	find /tmp -name new*	
56	find /etc -name *.conf -exec touch 201107110800 '{}' \;	
57		递归查找/etc 目录下文件名以 conf 结尾的文件
58	grep -R boot /tmp/*	
59		递归查找/etc/sshd 目录下包含 UsePAM 的文件中的行

60	<code>ls / tee /tmp/file.txt</code>	
61	<code>cd /usr/doc && echo "ok" echo "no such directory"</code>	
62	<code>cat /etc/passwd awk -F: '{print \$1}' sort wc -l</code>	对于这样的命令，建议先看 <code>cat /etc/passwd</code> 的输出结果，然后再看 <code>cat /etc/passwd awk -F: '{print \$1}'</code> 的结果，借着再看 <code>cat /etc/passwd awk -F: '{print \$1}' sort</code> 的结果，最后看整个命令的输出结果。
63	<code>grep bash /etc/passwd wc -l</code>	
64	<code>find . xargs -i -t sed -i 's women men g' {}</code>	
65	<code>tar -cjf /tmp/etc.tar.bz2 /etc/init.d &</code>	
66		解压解包 <code>/tmp/etc.tar.bz2</code> 到当前目录下
67	<code>ps axjf</code>	
68	<code>ps -ef</code>	
69	<code>pstree</code>	
70		把进程号 1234 强行杀掉
71	<code>kill -l</code>	
72	<code>ip a</code>	
73	<code>ip a show dev enp0s3</code>	
74	<code>ethtool enp0s3</code>	
75	<code>ip route</code>	
76	<code>ping www.veryopen.org</code>	
77	<code>ss -tnlp</code>	
78	<code>ss -unlp</code>	

79	systemctl restart ssh	
80	wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.16.9.tar.xz	
82		关闭 crond 服务
83		用 bc 命令计算 $123 * (1234 - 432) \% 11$ 的值
84	cat >abc.txt <<EOF What's your name? I'm wlm Oh, long time no seeee, how've you been I am fine, many thanks EOF	
85	passwd alice << PAS A1b2C3 A1b2C3 PAS	修改 alice 的密码为 A1b2C3
86	对于 CentOS 9	软件包管理
87	rpm -qa more	
88	rpm -qa grep boost	
89	rpm -ql acl	
90	rpm -qpl linuxqq-v1.0.2-beta1.i386.rpm	
91	rpm -ivh linuxqq-v1.0.2-beta1.i386.rpm	
92	rpm -e linuxqq	
93		查看已安装包 filesystem 所包含的文件
94	dnf search mutt	

95	dnf install mutt	
96	dnf group list	
97	dnf install "@FTP server"	
98	dnf remove "@FTP server"	

1.6 vim 实训

编辑文件/tmp/practice.txt，参考表 1-3。

表 1-3 vim 练习

行号	操作描述	vim 命令
1	输入如下内容： I'm happy that you're my teacher; I enjoy each lesson you teach. As my role model you inspire me To dream and to work and to reach. With your kindness you get my attention; Every day you are planting a seed Of curiosity and motivation To know and to grow and succeed. You help me fulfill my potential; I'm thankful for all that you've done. I admire you each day, and I just want to say, As a teacher, you're number one!	
2	退出插入模式，回到命令模式，显示行号	
3	光标快速跳到行首第一个字符处	
4	采用光标移动键 hjkl 把光标移到第 8 行第 11 个字符处	
5	删除光标处的两个字符	
6	光标快速跳到文件的末尾	
7	新增一行，内容是： ----- Written by WLM	
8	在文章开头新增一行： MY TEACHER	

9	在第 8 行末尾增加一个逗号和空格，然后把第 9 行合并到第 8 行的末尾	
10	删除第 3 行	
11	把第 6 行复制到第 8 行下面	
12	把 11,12 行剪贴到命名寄存器 a 中	
13	将 2~6 行复制到命名寄存器 b 中	
14	把命名寄存器 a 中的内容复制到文件末尾	
15	把命名寄存器 b 中的内容复制到文件的开头	
16	搜索 kindness 关键字，并查找下一个	
17	把全部的 attention 替换为 OK	
18	一次性删除 1~5 行	
19	把第 13 行剪贴到文件尾	
	最终文件内容如下： <pre> MY TEACHER I'm happy that you're my teacher; As my role model you inspire me To dream and to work and to reach. With your kindness you get my OK; Every day you are planting a seed Of curiosi and motivation To know and to grow and succeed. With your kindness you get my OK; I admire you each day, and I just want to say, As a teacher, you're number one! You help me fulfill my potential; I'm thankful for all that you've done. ----- Written by WLM </pre>	

20	存盘退出	
----	------	--

1.7 systemctl 命令

Systemd 通过配置好的“单元”来管理各种服务的启动和停止，systemctl 命令是我们与 systemd 守护进程交互的工具，此命令功能很强且使用复杂，其语法如下：

systemctl	[选项]	[操作]
<pre>-t, --type=<类型,...> 类型列表 --state=<状态,...> 状态列表 ---message=<内容> 重启或关机时的广播内容 --now 激活的同时启动单元 -H, --host=<[user@]host[:port]> 通过ssh远程执行命令 -a, --all 全部加载的活动和非活动的单元</pre>		1、管理单元 <pre>list-units [PATTERN ...] 列出满足匹配模式的单元 list-sockets [PATTERN...] 列出满足条件的socket类型的单元 list-timers [PATTERN...] 列出满足条件的timers类型的单元 is-active PATTERN... 查看单元是否活跃 is-failed PATTERN... 查看单元是否失败 status [PATTERN...] 查看目标单元的状态 show [PATTERN...] 查看目标单元的属性 cat PATTERN... 显示单元文件的内容 list-dependencies [UNIT...] 列出单元依赖的其他单元 start PATTERN... 启动单元 stop PATTERN... 关闭单元 reload PATTERN... 重新加载配置 restart PATTERN... 重启单元 isolate UNIT 启动单元及其依赖的单元，其他的单元全部关闭 daemon-reload 重新加载systemd的配置</pre> 2、管理单元文件 <pre>list-unit-files [PATTERN...] 列出目标单元文件 enable UNIT..., enable PATH... 激活单元或目录中的全部单元 disable UNIT... 禁用单元（不会自动启动，但可以手工启动） reenable UNIT... 先禁用然后激活单元 is-enabled UNIT... 是否激活 edit UNIT... 编辑单元文件 get-default 显示系统默认目标（启动时进入的目标） mask UNIT... 屏蔽单元（自会启动，也不能手工启动），unmask反方向操作 set-default TARGET 设置默认目标</pre> 3、管理系统 <pre>is-system-running 查看系统运行状态 default 进入默认目标 rescue 进入系统营救模式 halt 关机,poweroff 关机,reboot 重启,suspend 挂起系统,hibernate 休眠 switch-root ROOT [INIT] 切换到另一个根文件系统</pre>

-t 参数指名操纵的若干单元类型，类型之间用逗号分隔。--state 指明若干单元的状态，单元的类型不同，状态也不同。可用命令 `systemctl --state=help` 显示全部的状态。下面举例说明，供大家参考。

（1）翻页显示全部已经装载的活跃单元：

```
[root@localhost ~]# systemctl
```

此命令等价于 `systemctl list-units` 命令，因为 `list-units` 是默认的操作。按空格键翻页，按 `q` 退出。

（2）显示全部已经装载的活跃和非活跃单元：

```
[root@localhost ~]# systemctl --all
```

（3）列出类型为 `mount`、`socket` 的活跃单元：

```
[root@localhost ~]# systemctl -t mount,socket
```

（4）显示名称以 `network` 开头的单元：

```
[root@localhost ~]# systemctl list-units network\*
```

注意：匹配模式 `network*` 是 `systemctl` 命令的，而不是登录 SHELL 的，所以 `*` 前要加转义符 `\`，告诉 `bash` 星号是一个普通的字符而不是通配符。这样 `systemctl` 命令就能完

整获取“network*”了。

(5) 显示全部的套接口单元:

```
[root@localhost ~]# systemctl list-sockets
```

(6) 检查 `sshd.service` 单元是否活跃:

```
[root@localhost ~]# systemctl is-active sshd
```

注意: 命令中的单元后缀允许省略。

(7) 查看 `gdm.service` 单元的状态:

```
[root@localhost ~]# systemctl status gdm
```

(8) 显示全部已经装载的活跃和非活跃单元:

```
[root@localhost ~]# systemctl --all
```

(9) 显示 `dbus.service` 单元的配置内容:

```
[root@localhost ~]# systemctl cat dbus.service
```

(10) 查看 `sshd.service` 单元的全部依赖单元:

```
[root@localhost ~]# systemctl list-dependencies sshd.service
```

类似的命令“`systemd-analyze critical-chain`”查看默认单元所依赖的目标单元链。

(11) “启动”单元 `sshd.service`:

```
[root@localhost ~]# systemctl start sshd
```

单元的类型不同,“启动”的含义就不同,例如对于服务单元,“启动”就是运行相应的程序,而对于挂载单元,“启动”就是挂载某个分区了。对于 `stop`、`restart` 和 `reload` 可作类似的理解。`reload` 是重载的意思,当修改了配置文件之后,`reload` 就是通知有关进程重新读取配置,例如我修改了 `nginx` 网站服务的配置,然后就可以执行下面的命令通知 `nginx` 重新读取配置。当然也可以先 `stop` 再 `start`,或者直接 `restart`,不过这样做会产生短暂的网页不可访问的错误,这对于流量非常大的生产系统来说是难以接受的。

```
[root@localhost ~]# systemctl reload nginx.service
```

(12) 退出图形系统,进入多用户模式:

```
[root@localhost ~]# systemctl isolate multi-user.target
```

(13) 操作系统进入图形桌面模式:

```
[root@localhost ~]# systemctl isolate graphical.target
```

(14) 列出全部的单元文件:

```
[root@localhost ~]# systemctl list-unit-files
```

(15) 列出名字以 `time` 开头的单元文件:

```
[root@localhost ~]# systemctl list-unit-files time\*
```

(16) 激活 `sshd.service` 单元:

```
[root@localhost ~]# systemctl enable sshd
```

被激活的单元在机器启动时会自动启动。

(17) 禁用 `sshd.service` 单元, 被禁用的单元可以手工启动:

```
[root@localhost ~]# systemctl disable sshd.service
```

(18) 屏蔽 `sound.target` 目标, 被屏蔽的单元手工也无法启动:

```
[root@localhost ~]# systemctl mask sound.target
```

(19) 把 `multi-user.target` 设为默认目标:

```
[root@localhost ~]# systemctl set-default multi-user.target
```

这样重启计算机之后就进入多用户模式, 不再启动图形系统了。如果把 `graphical.target` 设为默认目标, 就会启动图形桌面。

(20) 检查操作系统的运行状态:

```
[root@localhost ~]# systemctl is-system-running
```

操作系统的运行状态参见表 1-4。

表 1-4 操作系统的运行状态

状 态	说 明	返回值
initializing	启动的早期阶段, 即尚未到达 <code>basic.target</code> 、 <code>rescue.target</code> 、 <code>emergency.target</code> 阶段	>0
starting	启动的后期阶段, 即任务队列首次达到空闲之前的阶段, 或者已经进入了某个救援目标中	>0
running	完全可用状态	0
degraded	类似 <code>running</code> 状态, 但存在若干单元启动失败	>0
maintenance	进入急救模式	>0
stopping	正在关机	>0
offline	系统可用, 但是 1 号进程不是 <code>systemd</code>	>0
unknown	状态不可知	>0

degraded 状态表明存在一些单元启动异常，这时可用下面的命令显示这些异常的单元：

```
[root@localhost ~]# systemctl --state=failed
```

然后采用命令 **systemctl status ...** 进一步排错。

(21) 直接进入默认目标：

```
[root@localhost ~]# systemctl default
```

如果修改了默认目标，此命令直接进入，当然能重启也是可以达到同样的目的。

(22) 关机：

```
[root@localhost ~]# systemctl poweroff
```

halt 是停机，**reboot** 是重启。停机与关机的区别是前者不彻底断掉主板的电源，还可以通过远程唤醒开机，而关机就是彻底断电了。

(23) 进入另一个根文件系统：

```
[root@localhost ~]# systemctl switch-root /mnt
```

能切换成功的前提是 **/mnt** 目录上挂载了一个根分区。类似的命令还有 **chroot**。

2 服务实训

搭建一个服务的目标分为三种，分别是功能目标、性能目标和产能目标。功能目标是最低要求，比如搭建邮件系统必须具有收、发和阅读邮件的功能，否则就是搭建失败。性能目标要求一个服务系统在预设的用户体验下能满足预期的访问量，体现的是系统的抗压能力，通常通过压力测试来评价，比如邮件系统能满足一万人正常使用。而产能目标体现服务的安全性和可用性，通常用多少个“9”来衡量，比如邮件系统的可用性为三个“9”，表示一年内 99.9% 的时间内用户可以正常收发邮件。在生产环境中使用的服务系统一般要求同时满足功能、性能和产能目标，否则都会出问题，比如一个邮件系统不能发邮件，那肯定没人会使用它，如果能收、发和阅读邮件但是当多几个人同时使用时速度很慢，还是没有人愿意使用它，如果功能和性能都满足，但是邮件内容很容易泄密或者隔三差五死机，这样的邮件系统仍然没人会用。

本章搭建的服务要求满足功能目标，性能目标一般是做横向扩展（采用集群、负载均衡等技术实现），实现一定的产能目标就更复杂了，涉及入侵检测、防火墙、容错、三防、灾备等等。另外本章搭建服务的方法是截止到 2022 年 12 月份的，由于各种服务软件的版本都在变，相应的配置方法肯定也会变，因此在将来的某个时间点，你再参照本章的方法可能存在少量问题。最好的途径是通过谷歌搜索相关软件的官方网站，然后参考官方网站上的最新文档。

假设下面的操作都是在 ip 地址为 192.168.192.20 的机器上进行。为了减少复杂性，对于 CentOS 9，在搭建各种服务之前建议先关闭 SELinux 服务，用 vim 修改 /etc/selinux/config 文件，把里面的 SELINUX=enforcing 改为 SELINUX=disabled，然后重启操作系统。命令 sestatus 可以查看 SELinux 的状态，如果返回 “SELinux status: disabled” 表示 SELinux 已经关闭。

2.1 时钟同步服务

在安装了 CentOS 9 的计算机上实现时钟同步, 是非常简单的, 前提是计算机连上互联网, 以试图参见图 2-1。

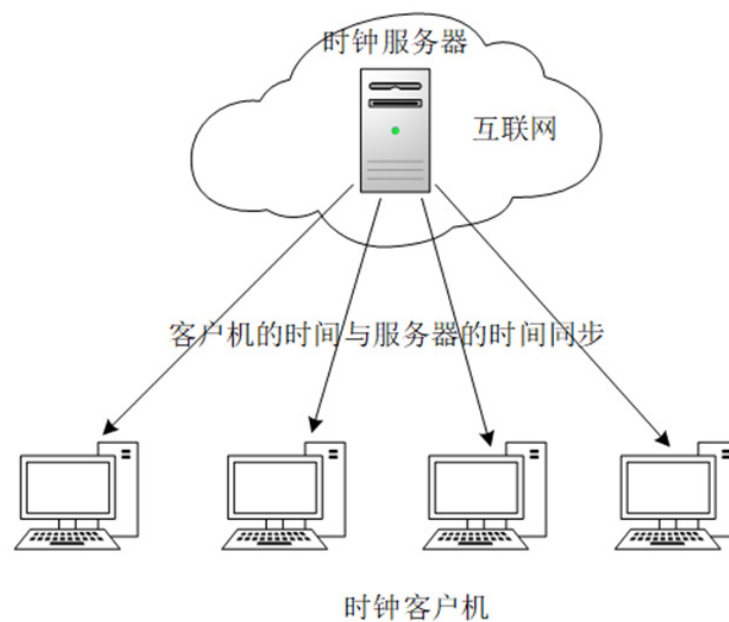


图 2-1 时钟同步的两种情况

首先确保时区选择正确, 运行命令 `timedatectl` 查看当前时区, 对于中国内地来说, 应该是 `Asia/Shanghai` (CST, +0800) 时区, 如果不对, 采用以下命令调正:

```
[root@localhost ~]# timedatectl set-timezone Asia/Shanghai

[root@localhost ~]# timedatectl
          Local time: 五 2022-12-16 09:18:17 CST
          Universal time: 五 2022-12-16 01:18:17 UTC
             RTC time: 五 2022-12-16 01:18:18
            Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

1) 安装时钟同步软件 chrony:

```
[root@localhost ~]# dnf install chrony -y
[root@localhost ~]# vim /etc/chrony.conf
.....
pool 2.centos.pool.ntp.org iburst                                #因特网上的时钟服务器
.....
makestep 1.0 3                                                  #如果时间相差 1 秒以上, 就在前三次更新中步进调整系统时钟
.....
```

```
[root@localhost ~]# systemctl restart chronyd.service

[root@localhost ~]# systemctl --now enable chronyd.service

[root@localhost ~]# chronyc -a makestep #立即同步
[root@localhost ~]# chronyc sources -v
.-- Source mode  '^' = server, '=' = peer, '#' = local clock.
/ .- Source state '*' = current best, '+' = combined, '-' = not combined,
| /      'x' = may be in error, '~' = too variable, '?' = unusable.
||
||      Reachability register (octal) --.      .- xxxx [ yyyy ] +/- zzzz
||      Log2(Polling interval) --.      |      xxxx = adjusted offset,
||                                  |      yyyy = measured offset,
||                                  |      zzzz = estimated error.
||
||
MS Name/IP address         Stratum Poll Reach LastRx Last sample
=====
^~ tick.ntp.infomaniak.ch   1   6   177   50   +7263us[+9360us] +/- 110ms
^* 120.25.115.20           2   6   275   51   -61us[+2039us] +/- 13ms #*表示同步服务器
^~ time.cloudflare.com      3   6    77   29   +45ms[ +45ms] +/- 145ms
^? ntp6.flashdance.cx       2   7    3   242   -36ms[ -34ms] +/- 195ms
```

验证，故意把系统时间改错，看看能否自动纠正过来：

```
[root@localhost ~]# date 102108302021
2021 年 10 月 21 日 星期四 08:30:00 CST

[root@localhost ~]# date #过一会儿再查看时间，发现同步了
2022 年 12 月 16 日 星期五 09:12:07 CST
```

与时钟服务器的网络延时越大，时间越不准确，我们使用的网络时钟服务器其实都是时间中继，原子钟最准确，1 层中继准确性次之，2 层又次之，15 层最不准确，参考示意图 2-2。

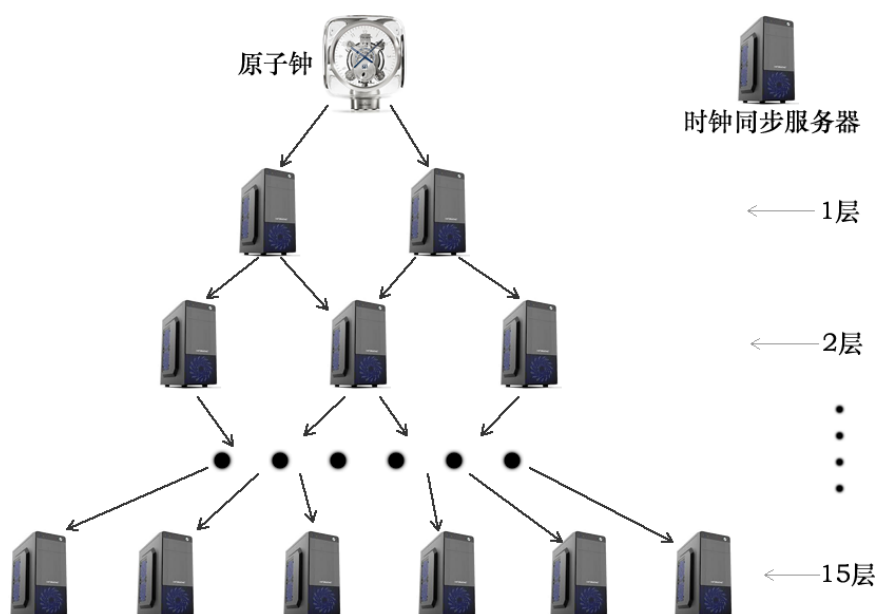


图 2-2 时钟同步服务层级

2.2 FTP 服务

FTP (File Transfer Protocol) 就是文件传输协议的简称, 用于不同计算机之间的文件传输。不同于 SFTP 用密文传输, FTP 是明文传输的, 所以不建议用它来传输敏感信息, 对安全要求高的环境, 建议还是采用 openssh 搭建 sftp 服务。在 Linux 上实现 ftp 功能的软件有 ftpd、VsFTPD 和 ProFTPD 等, 其中 vsFTP (Very Secure FTP) 是一种在 UNIX/Linux 中非常安全且快速稳定的 FTP 服务器。下面讲述 vsFTPD 的安装和配置, 实验目标是有 50 个虚拟用户, 其中前 10 个可以上传和下载文件, 后 40 个用户只能下载文件。

1. 安装 vsFTPD

```
[root@localhost ~]# dnf install vsftpd -y
```

然后修改配置文件 `/etc/vsftpd/vsftpd.conf`, 确保如下内容没有被注释。

```
[root@localhost ~]# vim /etc/vsftpd/vsftpd.conf
anonymous_enable=NO
dirmessage_enable=YES
ftpd_banner=欢迎进入李木生的 ftp 服务器!
guest_enable=YES
guest_username=vsftpduser
listen=NO
listen_ipv6=YES
local_enable=YES
pam_service_name=vsftpd
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
user_config_dir=/etc/vsftpd/user_conf
local_umask=022
allow_writeable_chroot=YES
```

注意: 每一行的行首和行尾不能出现空格, 至于这些参数的含义, 请参考在线文档 `man 5 vsftpd.conf`。

2. 建立虚拟用户库

1) 采用下面一条命令产生 50 个虚拟用户和密码并保存在临时文件 `/tmp/users.list` 中 (奇数行是用户名, 偶数行是密码):

```
[root@localhost ~]# for((i=1;i<51;i++)); do echo -e "ftpuser$i\npassword$i">>/tmp/users.list; done
```

用户名是 `ftpuser1`、`ftpuser2`、`ftpuser3`……, 密码分别是 `password1`、`password2`……。或者 `vim /tmp/users.list`

2) 安装贝克利库管理软件包:

```
[root@localhost ~]# dnf install libdb-utils -y
```

3) 产生用户库文件:

```
[root@localhost ~]# db_load -T -t hash -f /tmp/users.list /etc/vsftpd/vsftpd_login.db
[root@localhost ~]# chmod 600 /etc/vsftpd/vsftpd_login.db
[root@localhost ~]# rm -rf /tmp/users.list
```

3. 配置

1) 创建系统用户 vsftpduser:

```
[root@localhost ~]# useradd -s /usr/sbin/nologin vsftpduser
```

2) 为每个虚拟用户创建家目录:

```
[root@localhost ~]# for((i=1;i<51;i++)); do mkdir /home/vsftpduser/ftpuser$i; done
[root@localhost ~]# chown -R vsftpduser: /home/vsftpduser
[root@localhost ~]# chmod -R 700 /home/vsftpduser
```

虚拟用户的家目录为/home/vsftpduser/ftpuserX, X=1,2,3,...,50。

3) 为每个虚拟用户创建配置文件

```
[root@localhost ~]# mkdir /etc/vsftpd/user_conf && cd /etc/vsftpd/user_conf
[root@localhost user_conf]# for((i=1;i<51;i++)); do echo "anon_world_readable_only=NO">ftpuser$i; done
[root@localhost user_conf]# for((i=1;i<51;i++)); do echo "local_root=/home/vsftpduser/ftpuser$i">>ftpuser$i; done
```

前十个用户允许上传文件:

```
[root@localhost user_conf]# for((i=1;i<11;i++)); do echo -e \
"write_enable=YES\nanon_upload_enable=YES\nanon_mkdir_write_enable=YES\
\nanon_other_write_enable=YES">>ftpuser$i; done
```

4) 下面创建 pam 身份鉴别机制

```
[root@localhost ~]# echo "auth required pam_userdb.so db=/etc/vsftpd/vsftpd_login" >/etc/pam.d/vsftpd
[root@localhost ~]# echo "account required pam_userdb.so db=/etc/vsftpd/vsftpd_login" >>/etc/pam.d/vsftpd
```

4. 重启 vsftpd 并测试

```
[root@localhost ~]# systemctl restart vsftpd #重启服务
[root@localhost ~]# systemctl status vsftpd #查看服务的状态
● vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-12-16 10:28:34 CST; 6s ago
   Process: 31650 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status=0/SUCCESS)
   Main PID: 31651 (vsftpd)
     Tasks: 1 (limit: 24524)
    Memory: 716.0K
       CPU: 2ms
    CGroup: /system.slice/vsftpd.service
            └─31651 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf
.....

[root@localhost ~]# dnf install ftp -y #安装 ftp 客户端
[root@localhost ~]# ftp localhost #与本机建立 ftp 链接
Connected to 127.0.0.1.
220 欢迎进入李木生的 ftp 服务器!
Name (127.0.0.1:root): ftpuser1
331 Please specify the password.
Password: #输入密码 password1
230 Login successful.
Remote system type is UNIX.
```

```
Using binary mode to transfer files.  
ftp>
```

#建立链接成功

这样表明与 **ftp** 服务器建立了链接，接下来就可以上传和下载文件了。输入 **help** 列出全部可用的命令。

```
ftp> help  
!  
$  
account  
append  
ascii  
bell  
binary  
bye  
case  
cd  
cdup  
chmod  
close  
cr  
delete  
debug  
ftp>  
dir  
disconnect  
exit  
form  
get  
glob  
hash  
help  
idle  
image  
ipany  
ipv4  
ipv6  
lcd  
ls  
macdef  
mdelete  
mdir  
mget  
mkdir  
mls  
mode  
modtime  
mput  
newer  
nmap  
nlist  
ntrans  
open  
prompt  
passive  
proxy  
qc  
sendport  
put  
pwd  
quit  
quote  
recv  
reget  
rstatus  
CentOS  
rename  
reset  
restart  
rmdir  
runique  
send  
site  
size  
status  
struct  
system  
sunique  
tenex  
tick  
trace  
type  
user  
umask  
verbose  
?
```

其中最常用的命令有！（执行本地命令）、**dir**（列出服务器上的文件）、**lcd**（改变本地目录）、**mkdir**（在服务器上创建目录）、**get**（下载文件）、**put**（上传文件）。采用 **help** 可以获取命令的用法，如：

```
ftp> help get  
ftp> lcd /etc  
ftp> put profile  
ftp> dir  
200 PORT command successful. Consider using PASV.  
150 Here comes the directory listing.  
-rw----- 1 2004 2004 665 Aug 26 17:07 123.txt  
-rw----- 1 2004 2004 665 Aug 26 17:45 profile  
226 Directory send OK.  
ftp> lcd /tmp  
ftp> get 123.txt  
ftp> mkdir musics  
ftp> cd musics  
ftp> put beijing.mp3 ./beijingbeijing.mp3  
ftp> exit  
#进入本地的/etc 目录  
#上传文件 profile  
#本地进入/tmp 目录  
#下载文件 123.txt  
#在服务器上创建目录 musics  
#进入服务器上的目录 musics  
#上传文件  
#退出 ftp
```

5. Windows 上的 ftp 客户端

Windows 10+操作系统自带了 **ftp** 命令，直接使用即可，参考图 2-3。另外常见的图形化 **ftp** 客户端工具还有 **FileZilla**、**Cyberduck** 和 **FlashFXP**，其中 **FileZilla** 不但支持 **ftp** 协议，还支持 **sftp** 协议。

2.3 DHCP 服务

DHCP 为 Dynamic Host Configuration Protocol 的首字母缩写，意为动态主机配置协议，主要功能是给局域网内的机器分配网络参数（如 ip、网关、DNS 等等），属于基础网络服务。

一台计算机启动时自动配置网络参数的过程参见图 2-5。

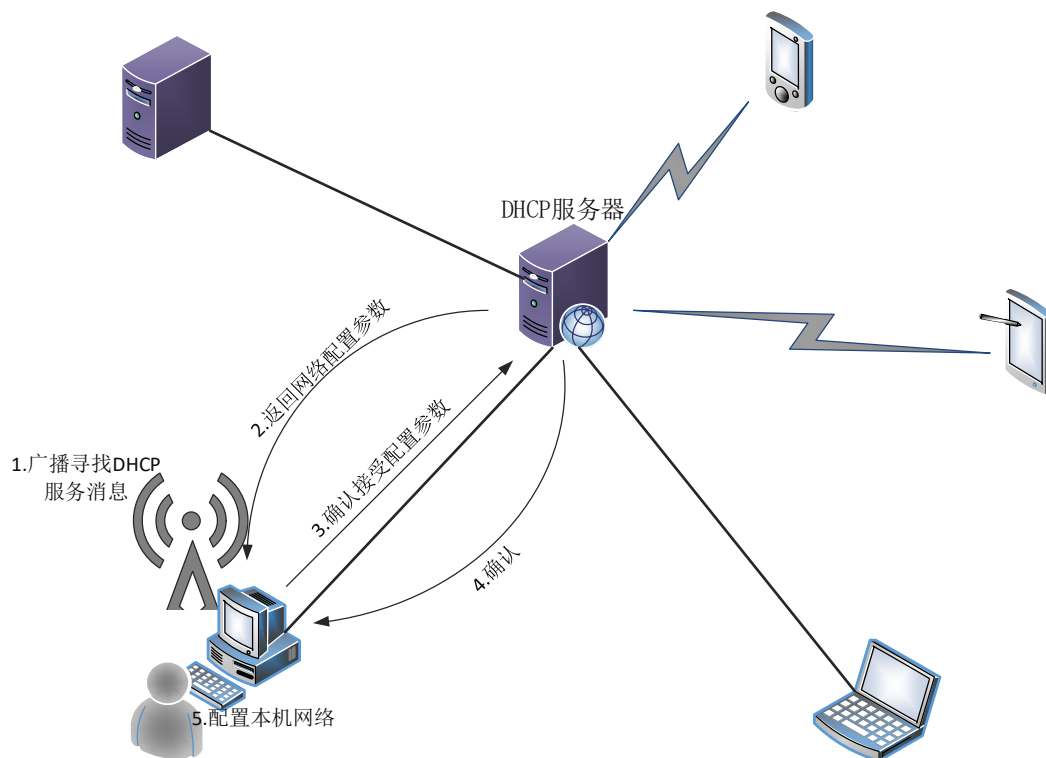


图 2-5 自动配置网络

机器自动配置网络的过程是这样的：

1. 客户机首先广播“寻找 DHCP 服务器”的消息（即 DHCPDiscover）。
2. DHCP 服务器返回网络配置参数（客户机可能会收到多个 DHCP 服务器提供的参数）。
3. 客户机向其中一台 DHCP 服务器发送确认接受其提供的配置参数的消息（一般是最近的那台 DHCP 服务器）。
4. DHCP 服务器返回确认消息。
5. 客户机配置自己的网络。

下面在 CentOS 9 上搭建 DHCP 服务，首先给虚拟机增加一块网卡，模式为 `host-only`，并给此网卡配置 IP 为 `192.168.10.3`，假设仅主机模式（`host-only`）网络为 `192.168.10.0/24`，可参考图 2-6 查看虚拟机软件的 `host-only` 网络（点击菜单“编辑”>>“虚拟机网络编辑器”）。

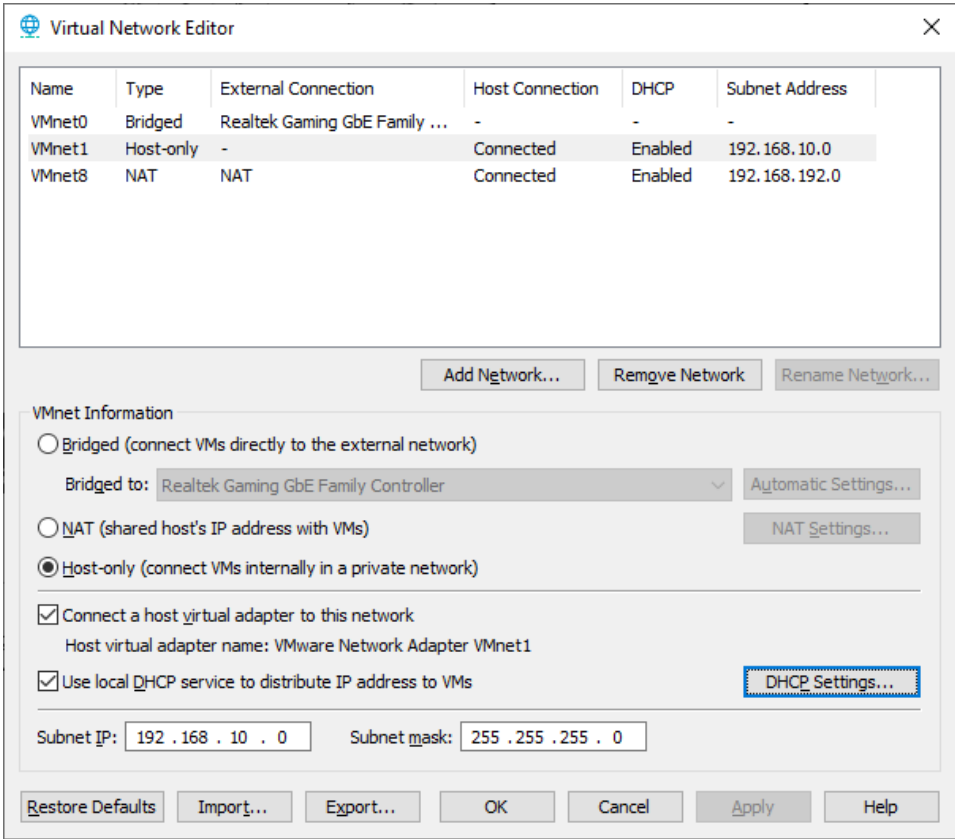


图 2-6 查看虚拟机网络

1. 安装

```
[root@localhost ~]# dnf -y install dhcp-server dhcp-client
[root@localhost ~]# systemctl enable dhcpd.service
```

服务单元是 `dhcpd.service`，服务程序是 `/usr/sbin/dhcpd`，启动后监听 67 号端口，主配置文件目录 `/etc/dhcp`，创建了系统用户 `dhcpd`，此用户专门用于运行 `dhcp` 服务，主配置文件是 `/etc/dhcp/dhcpd.conf` 和 `/etc/dhcp/dhcpd6.conf`，后者是 IPv6 的。

2. 配置

要求：服务器采用固定的网络参数，在 `192.168.10.201~192.168.10.224` 范围内分

配，办公计算机采用动态的，地址范围是 192.168.10.51~192.168.10.200，固定 ip 范围是 192.168.10.5~192.168.10.50。

用 vim 编辑/etc/dhcp/dhcpd.conf，输入如下内容：

```
#
# DHCP Server Configuration file.
# see /usr/share/doc/dhcp*/dhcpd.conf.example
# see dhcpd.conf(5) man page
#

#全局参数：
option domain-name "weisuan.com";
option domain-search "weisuan.com";
option domain-name-servers 202.96.128.86,8.8.8.8;
option routers 192.168.10.1;
option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

#根据服务器网卡的 MAC 地址来分配固定的 ip 和主机名：
group {
    use-host-decl-names on;
    host erp-server{ hardware ethernet 08:00:46:AC:A9:8E; fixed-address 192.168.10.201;}
    host database-server{ hardware ethernet 00:0C:29:54:94:45; fixed-address 192.168.10.202;}
    host webserver{ hardware ethernet 00:0c:29:dd:9a:48; fixed-address 192.168.10.205;}
}

#办公计算机动态 ip:
subnet 192.168.10.0 netmask 255.255.255.0 {
    range 192.168.10.51 192.168.10.200;
}
```

实操时，ip 和 MAC 地址要根据实际情况修改。最后重启 dhcpd 服务：

```
[root@localhost ~]# systemctl restart dhcpd.service
```

使用下面的命令查看 dhcpd 运行情况：

```
[root@localhost ~]# systemctl status -l dhcpd.service
#查看服务单元的运行状态
[root@localhost ~]# ss -u -l -p -n
#查看 67 号 UDP 端口是否被监听
```

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port	Process
UNCONN	0	0	0.0.0.0:67	0.0.0.0:*	users:(("dhcpd",pid=2534,fd=7))
UNCONN	0	0	127.0.0.1:323	0.0.0.0:*	users:(("chronyd",pid=736,fd=5))
UNCONN	0	0	0.0.0.0:37765	0.0.0.0:*	users:(("avahi-daemon",pid=718,fd=14))
UNCONN	0	0	0.0.0.0:5353	0.0.0.0:*	users:(("avahi-daemon",pid=718,fd=12))
UNCONN	0	0	:::1:323	:::1:*	users:(("chronyd",pid=736,fd=6))
UNCONN	0	0	:::52348	:::1:*	users:(("avahi-daemon",pid=718,fd=15))
UNCONN	0	0	:::5353	:::1:*	users:(("avahi-daemon",pid=718,fd=13))

3. 测试

给另外一台虚拟机（不妨称为 B 虚拟机）添加一块 host-only 网卡，启动后把网卡设成 dhcp 配置模式，然后关机。再 windows 这边按 win+r，输入 services.msc 并回车，接下来参考图 2-7 关闭 VMware DHCP 服务。

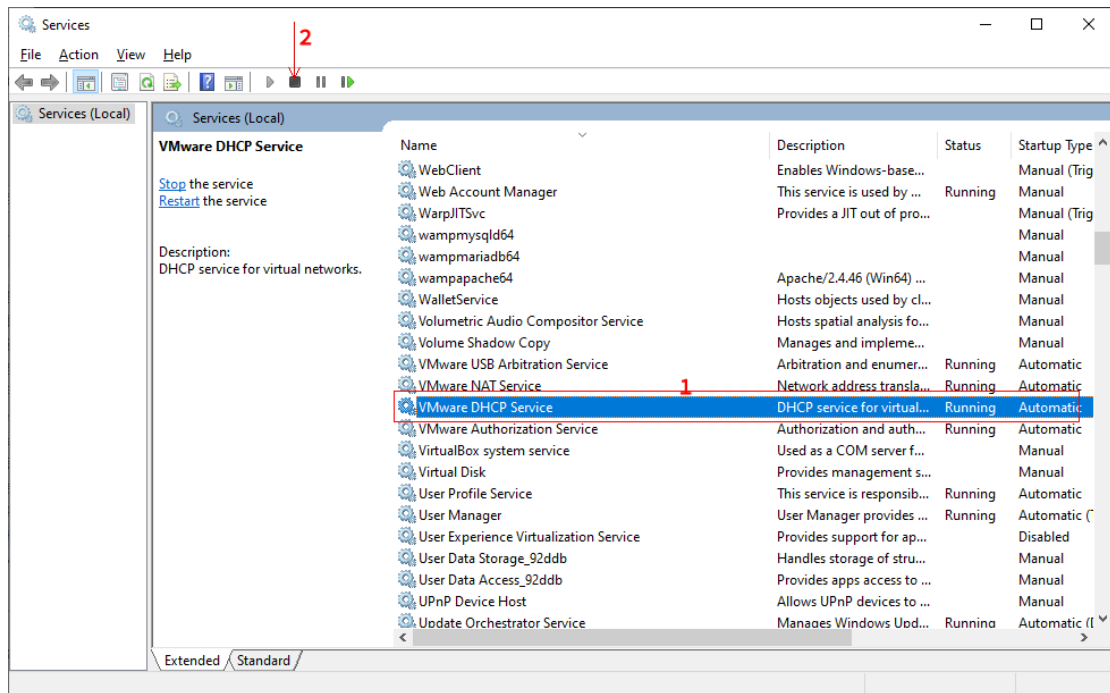


图 2-7 关闭 VMware DHCP 服务

启动 B 虚拟机，看看网卡能否自动配置 ip，然后关机。把 B 虚拟机的 host-only 网卡的 MAC 地址设为 08:00:46:AC:A9:8E（参考图 2-8 所示操作），开机后看看是否自动配置了 ip 地址 192.168.10.201。MAC 地址 08:00:46:AC:A9:8E 与 ip 地址 192.168.10.201 的对应关系参考前面的“2. 配置”。

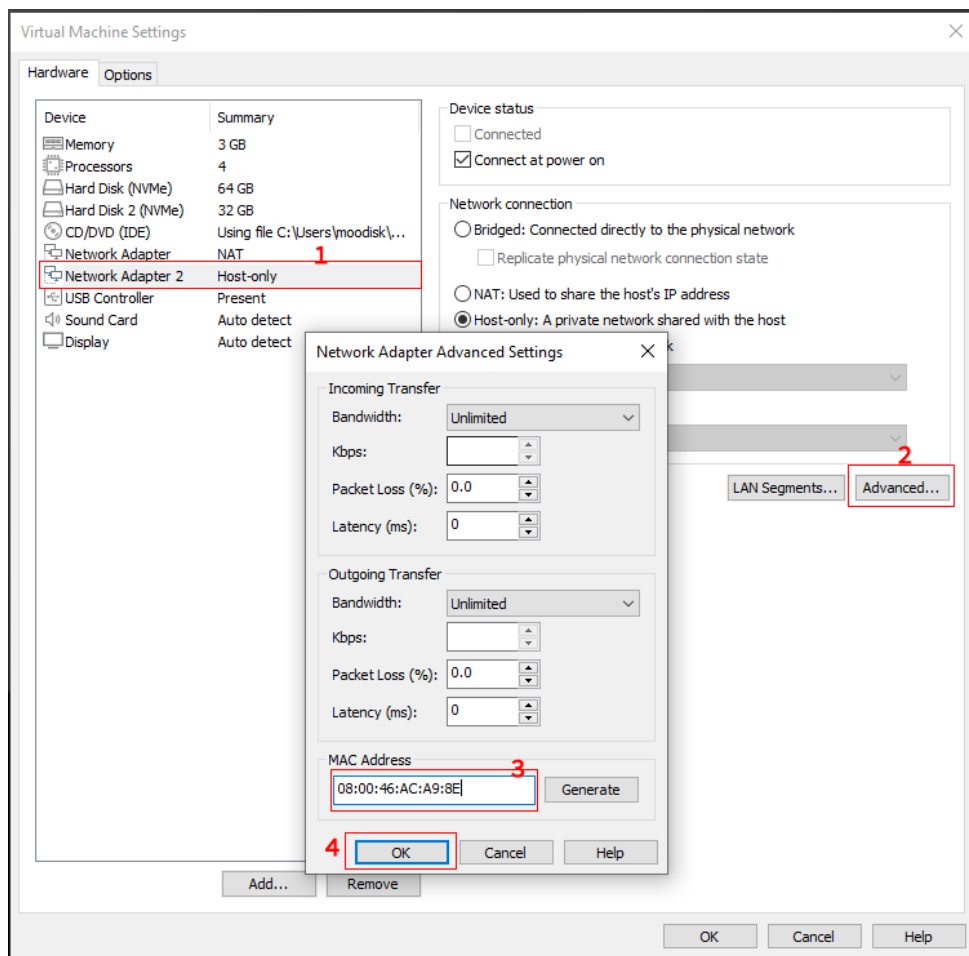


图 2-8 修改网卡的 MAC 地址

采用 `ip a` 查看地址，结果如图 2-9 所示，显示自动配置了 `10.3.100.201`。

```

root@nfs-server:~
root@nfs-server ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:3f:17:9a brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 192.168.192.10/24 brd 192.168.192.255 scope global noprefixroute ens160
        valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe3f:179a/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 08:00:46:ac:a9:8e brd ff:ff:ff:ff:ff:ff
    altname enp19s0
    inet 10.3.100.201/24 brd 10.3.100.255 scope global dynamic noprefixroute ens224
        valid_lft 559sec preferred_lft 559sec
    inet6 fe80::a00:46ff:feac:a98e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@nfs-server ~]#
  
```

图 2-9 查看网卡的 ip

2.4 DNS 服务

实训目的：搭建 DNS 服务，并能解析一个域名。

域名服务（domain name service, DNS）主要完成这样的任务：用域名查询 ip 地址，或者用 ip 查询域名。DNS 属于基础网络服务，在现代的互联网时代扮演着重要的角色。

2.4.1 域名解析

人们都是通过域名来访问网站，诸如 `www.google.com` 的域名便于记忆，但是计算机最终只能通过 ip 地址才能访问网站服务器，通过域名找到对应的 ip 地址就是 DNS 服务器的重要功能——称为域名解析（或正向解析），示意图参见图 2-9。反向解析就是通过 ip 地址查询域名，这是 DNS 服务器的一个次要功能。我在帮企业建网站时必做的三件事就是：注册域名，比如在 `www.godaddy.com` 上注册 `www.weisuan.com` 域名；租赁网站空间，比如我在谷歌云平台上租的 vps，ip 地址是 `59.188.87.107`；在 DNS 服务器增加一条 A 记录“`www.weisuan.com A 59.188.87.107`”（登录 `www.godaddy.com` 操作）。这样任何人访问 `www.weisuan.com` 网站时 DNS 服务器都会解析到 ip 地址 `59.188.87.107`，这个地址就是我的 vps，网站就放在那里。

目前世界上的域名达十几亿之多，由分布在各个国家的成千上万台 DNS 服务器承担解析任务，这些服务器之间的关系要么是上/下级关系，要么是主/从关系，要么是镜像关系，要么是缓存关系。用户在一台 DNS 服务器上添加了 A 记录，通常要过数小时后才能正确解析，原因是把新加的 A 记录同步到世界各地的主从、镜像和缓存服务器需要花费数小时的时间。那么又怎么来理解 DNS 服务器的上/下级关系呢？这要从域名的结构说起，域名 `www.weisuan.com` 的完整写法应该是 `www.weisuan.com.`，最后还有一个点。所有这种采用点分结构的域名组成一颗倒树，参见图 2-10。

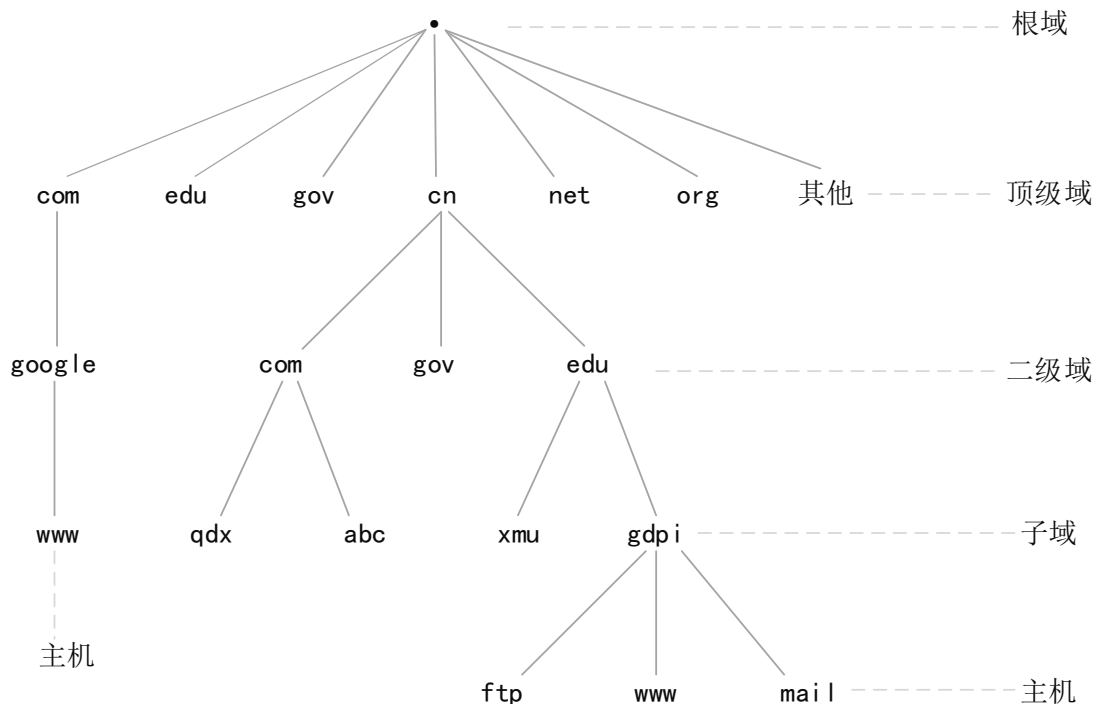


图 2-10 域名倒树结构

在图 2-10 中，用“.”连接从叶子到根的每一个节点，就成了一个完整的域名，比如 `www.gdpi.edu.cn.`、`www.google.com.`、`ftp.gdpi.edu.cn.` 等等。每一层都由若干台 DNS 服务器管理，上一层负责管理下一层服务器的信息。对域名做解析时是从域名的右侧开始的，比如解析域名 `www.veryopen.org.` 的过程是这样的：最右侧是“.”，因此到根 DNS 服务器去查询专门负责解析 `org` 顶级域的 DNS 服务器的 ip 地址，然后再通过 `org` 对应的 DNS 服务器（每一层都有多个 DNS 服务器）去查询专门负责解析 `veryopen` 二级域名的 DNS 服务器的 ip 地址，最后再通过子域 DNS 服务器去查询主机名为 `www` 的这台主机的资源记录，从而得到 `www.veryopen.org.` 的 IP 地址。我们可以通过 `dig +trace www.veryopen.org.` 命令来追踪整个 DNS 查询的过程。

```

[root@localhost ~]# dig +trace www.veryopen.org.

; <<>> DiG 9.16.23-RH <<>> +trace www.veryopen.org.
;; global options: +cmd
.                5      IN      NS      m.root-servers.net.
.                5      IN      NS      c.root-servers.net.
.                5      IN      NS      l.root-servers.net.
.                5      IN      NS      e.root-servers.net.
.                5      IN      NS      k.root-servers.net.
.                5      IN      NS      d.root-servers.net.
.                5      IN      NS      h.root-servers.net.
.                5      IN      NS      b.root-servers.net.
.                5      IN      NS      g.root-servers.net.
.                5      IN      NS      f.root-servers.net.
.                5      IN      NS      j.root-servers.net.
.                5      IN      NS      i.root-servers.net.
.                5      IN      NS      a.root-servers.net.
.....
;; Received 525 bytes from 192.168.192.2#53(192.168.192.2) in 2014 ms

org.             172800 IN      NS      a2.org.afiliat-nst.info.
  
```

```

org.          172800 IN      NS      c0.org.afilias-nst.info.
org.          172800 IN      NS      b2.org.afilias-nst.org.
org.          172800 IN      NS      a0.org.afilias-nst.info.
org.          172800 IN      NS      d0.org.afilias-nst.org.
org.          172800 IN      NS      b0.org.afilias-nst.org.
.....
;; Received 816 bytes from 192.36.148.17#53(i.root-servers.net) in 124 ms

veryopen.org. 3600    IN      NS      ns67.domaincontrol.com.
veryopen.org. 3600    IN      NS      ns68.domaincontrol.com.
.....
;; Received 601 bytes from 199.19.56.1#53(a0.org.afilias-nst.info) in 120 ms

www.veryopen.org. 3600    IN      CNAME   veryopen.org.
veryopen.org. 3600    IN      A       34.133.211.54
veryopen.org. 3600    IN      NS      ns68.domaincontrol.com.
veryopen.org. 3600    IN      NS      ns67.domaincontrol.com.
;; Received 130 bytes from 173.201.71.44#53(ns68.domaincontrol.com) in 121 ms

```

我们看到整个 DNS 查询过程就是根据域名从右到左一步一步查询，. root DNS→org DNS→veryopen DNS→www。图 2-11 很好的阐释了域名的迭代解析过程（递归解析请参阅其他资料）。

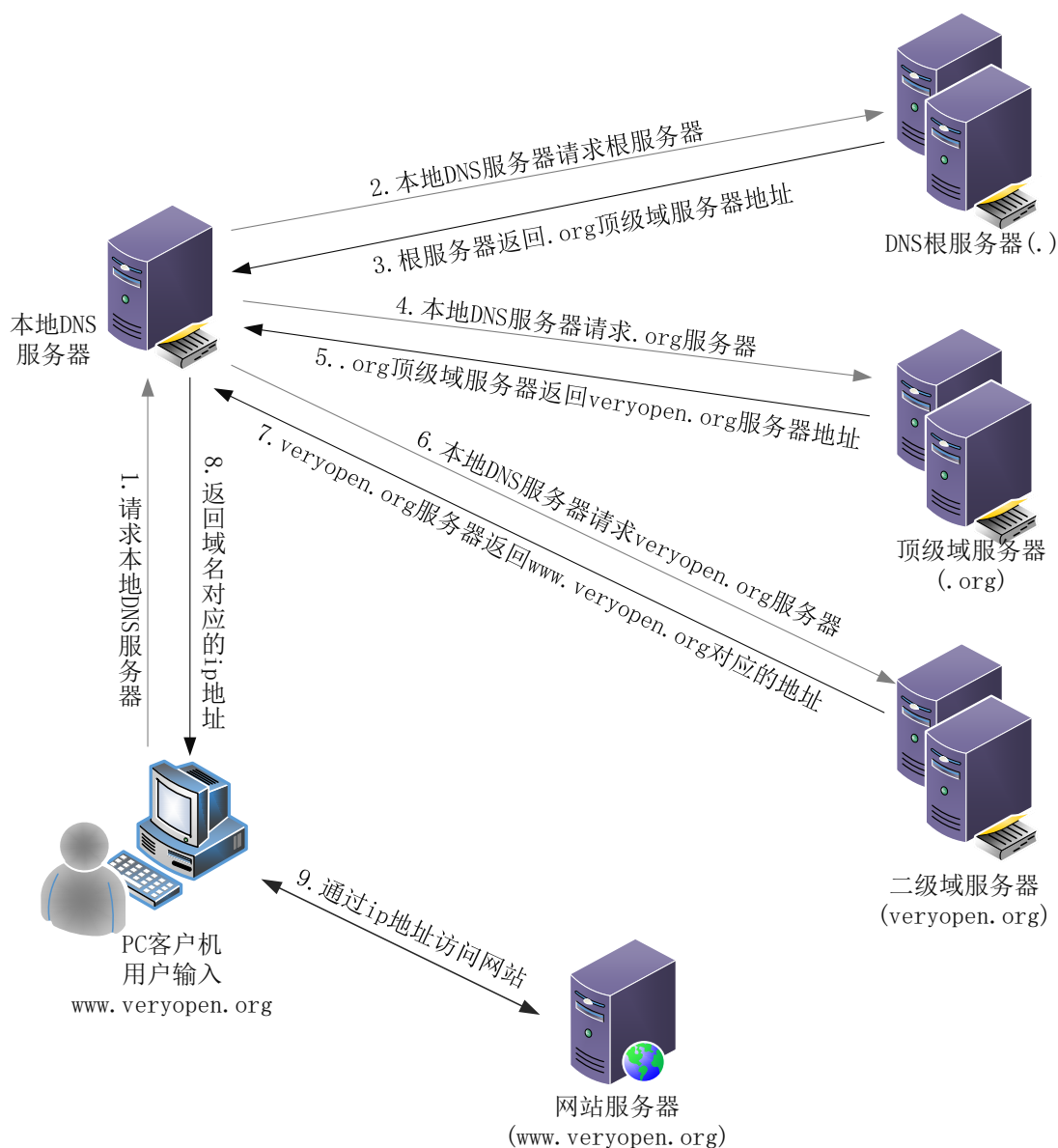


图 2-11 域名的迭代解析

为了加快域名解析速度，当前的 DNS 服务器都有本地缓存的功能，比如只要之前有人请求过 `.com` 的域名，那么本地 DNS 服务器就缓存了顶级域服务器的 ip 地址，如果也有人曾经请求过 `.veryopen.org` 的域名，那么本地 DNS 服务器就缓存了二级域服务器的 ip 地址，如果有人已经访问过 `www.veryopen.org` 网站，那么本地 DNS 服务器和用户自己的计算机就缓存了此域名对应的 ip 地址，以后再次访问这个域名，本地 DNS 服务器直接返回 ip，根本不用去请求其他 DNS 服务器，所以域名解析的速度很快。另外用户也可以把域名和对应的 ip 添加到 PC 客户机的 hosts 文件中（Linux 下是 `/etc/hosts`，Windows 下是 `C:\windows\system32\drivers\etc\hosts`），这样在此 PC 机上浏览 `www.veryopen.org` 时根本不用请求本地 DNS 服务器。可采用下面的命令往 `/etc/hosts` 文件添加一行：

```
[root@localhost ~]# echo "www.veryopen.org 59.188.87.107" >>/etc/hosts
```

2.4.2 资源记录

DNS 服务器上的信息都是通过资源记录 (RR=Resource Record) 的格式进行保存, RR 不仅能够保存域名到 IP 地址的对应关系, 还能保持其他很多的信息。一条 RR 记录的格式是:

NAME	CLASS	TYPE	RDATA
------	-------	------	-------

其中 NAME 指主机名, CLASS 指类别 (通常等于 IN, 即 Internet), TYPE 指记录的类型, RDATA 指具体的信息。如下面是一些常见的资源记录:

veryopen.org.	IN	A	59.188.87.107
mail	IN	A	192.168.1.2
www	IN	CNAME	weisuan.com.
IN MX 10		mail.gmail.com.	

表 2-1 列举了一些常见的资源类型 TYPE。

表 2-1 DNS 资源类型

序号	类型	说明
1	A	IPV4 地址
2	AAAA	IPV6 地址
3	MX	邮件记录
4	CNAME	别名
5	PTR	指针 (用于反向解析, 即由 ip 地址查询域名)
6	SRV	服务资源记录
7	NS	域名服务器记录

可以采用 “dig -t TYPE 域名” 来查询一个域名某种类型的记录, 比如:

```
[root@localhost ~]# dig -t A www.baidu.com
; <<>> DiG 9.16.23-RH <<>> -t A www.baidu.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45916
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
;; QUESTION SECTION:
;www.baidu.com.                IN      A

;; ANSWER SECTION:
www.baidu.com.      5      IN      CNAME   www.a.shifen.com.
www.a.shifen.com.   5      IN      A       163.177.151.109
www.a.shifen.com.   5      IN      A       163.177.151.110

;; Query time: 2025 msec
;; SERVER: 192.168.192.2#53(192.168.192.2)
;; WHEN: Fri Dec 16 16:18:42 CST 2022
;; MSG SIZE rcvd: 101
```

2.4.3 安装和配置 DNS

我们在 CentOS 9 上搭建一个主 DNS 服务器，采用经典的 bind 开源软件，资源记录保存在文件中。不妨假设本机器的 ip 为 192.168.192.20。

1. 安装

```
[root@localhost ~]# dnf -y install bind bind-chroot bind-utils
[root@localhost ~]# systemctl --now enable named-chroot.service #激活并启动单元
```

DNS 服务软件包是 bind，bind-utils 是配置工具，而 bind-chroot 是 chroot 环境——让 bind 运行在封闭的沙箱里（伪根）以增强安全性。DNS 服务是 named-chroot.service，对应的服务程序是/usr/sbin/named，启动后它监听 53 号端口。安装 bind 时自动创建了系统用户 named，此用户专门用来运行 DNS 服务进程。

2. 配置

配置文件是/etc/named.conf 和/etc/named.rfc1912.zones。资源记录保存在目录 /var/named 下各个文件中。为了安全起见，bind 运行在伪根环境（即 /var/named/chroot），所以配置文件的目录改变了，如下所示：

/var/named/chroot/etc/named.conf

/var/named/chroot/etc/named.rfc1912.zones

/var/named/chroot/var/named

不过伪根中的文件和目录都是采用 mount 命令绑定到根文件系统中相应的文件和目录的，例如文件/var/named/chroot/etc/named.conf 就是采用下面的命令绑定的：

```
[root@localhost ~]# touch /var/named/chroot/etc/named.conf #新建空文件
```

```
[root@localhost ~]# mount --bind /etc/named.conf /var/named/chroot/etc/named.conf #绑定
```

这样修改/etc/named.conf 等于修改了/var/named/chroot/etc/named.conf，因为它们是一个文件。

首先修改/etc/named.conf，确保下面两个参数的值：

```
listen-on port 53 { any; }; #监听计算机上的所有网卡 Ip
allow-query { any; }; #允许任何人查询 DNS
```

修改/var/named/chroot/etc/named.rfc1912.zones，在文件末尾添加下面的内容：

```
zone "weisuan.com" IN {
    type master;
    file "weisuan.local";
    allow-update { none; };
};

zone "0.168.192.in-addr.arpa" IN {
    type master;
    file "weisuan.zone";
    allow-update { none; };
};
```

下面再来编辑两个资源文件 weisuan.local 和 weisuan.zone。

/var/named/chroot/var/named/weisuan.local 文件的内容如下：

```
$TTL 1D ;生存期为 1 天
@      IN SOA  ns.weisuan.com. root.weisuan.com. (
                                0      ; 更新序列号
                                1D     ; 刷新时间是 1 天
                                1H     ; 重试延时 1 小时
                                1W     ; 有效期为 1 星期
                                3H )   ; 无效解析记录的缓存时间为 3 小时

@      IN     NS      ns.weisuan.com.
ns     IN     A       192.168.0.100
www    IN     A       192.168.0.100
www1   IN     A       192.168.0.100
ftp    IN     CNAME   www
mail   IN     A       192.168.0.101
video  IN     A       192.168.0.100
```

此文件做正向解析，即从域名解析 ip。

/var/named/chroot/var/named/weisuan.zone 文件的内容如下：

```
$TTL 1D
@      IN SOA  ns.weisuan.com. root.weisuan.com. (
                                0      ; serial
                                1D     ; refresh
                                1H     ; retry
                                1W     ; expire
                                3H )   ; minimum

@      IN     NS      ns.weisuan.com.
ns     IN     A       192.168.0.100
100    IN     PTR     www.weisuan.com.
       IN     PTR     ftp.weisuan.com.
       IN     PTR     www1.weisuan.com.
       IN     PTR     video.weisuan.com.
101    IN     PTR     mail.weisuan.com.
```

这个资源文件主要是完成反向解析，即由 ip 查询域名。

写好配置文件后可以采用命令 named-checkconf 和 named-checkzone 检查语法，比

如：

```
[root@localhost ~]# named-checkconf /var/named/chroot/etc/named.conf
[root@localhost ~]# named-checkzone weisuan.com /var/named/chroot/var/named/weisuan.local
zone weisuan.com/IN: loaded serial 0
OK
[root@localhost ~]# named-checkzone weisuan.zone /var/named/chroot/var/named/weisuan.zone
zone weisuan.zone/IN: loaded serial 0
OK
```

修改新加的两个资源文件的权限：

```
[root@localhost ~]# chmod o-rwx,g+r /var/named/chroot/var/named/*
[root@localhost ~]# chgrp named /var/named/chroot/var/named/*
```

最后启动 DNS 服务，named 进程首先陷入伪根/var/named/chroot/目录，然后才读取配置文件/etc/named.conf（注意：在真根目录下，此配置文件对应/var/named/chroot/etc/named.conf）：

```
[root@localhost ~]# systemctl restart named-chroot.service #重启 DNS 服务
```

如果报错，那么运行命令 `systemctl status -l named-chroot.service` 查看错误信息。

在另一台安装了 linux 的计算机上安装 bind-utils，此软件包含命令 nslookup，然后设置 DNS 服务器为 192.168.192.20，然后运行命令 nslookup 验证正向和反向解析是否成功。在 Linux 下还可使用命令 `host www.weisuan.com` 或 `host 192.168.0.100`，前者返回 ip，后者返回域名。

```
[root@nfs-server ~]# nslookup
> www.weisuan.com <-----输入域名
Server:          192.168.192.20
Address:         192.168.192.20#53

Name:   www.weisuan.com
Address: 192.168.0.100
> 192.168.0.100 <-----输入 ip
100.0.168.192.in-addr.arpa    name = ftp.weisuan.com.
100.0.168.192.in-addr.arpa    name = www.weisuan.com.
100.0.168.192.in-addr.arpa    name = www1.weisuan.com.
100.0.168.192.in-addr.arpa    name = video.weisuan.com.
> exit

[root@nfs-server ~]# host www.weisuan.com
www.weisuan.com has address 192.168.0.100

[root@nfs-server ~]# host 192.168.0.100
100.0.168.192.in-addr.arpa domain name pointer www.weisuan.com.
100.0.168.192.in-addr.arpa domain name pointer ftp.weisuan.com.
100.0.168.192.in-addr.arpa domain name pointer www1.weisuan.com.
100.0.168.192.in-addr.arpa domain name pointer video.weisuan.com.
```

2.5 网站服务

实训目的：在安装了 CentOS 9 的计算机上搭建 LAMP 网站系统，然后在浏览器里能打开一个静态网页和一个动态网页。

2.5.1 网站简介

网页分为静态和动态两种，静态网页文件就是 HTML/Javascript/CSS 编码文件，当用户浏览静态网页时，服务器就把 html 网页文件发过来，因此用户每次看到的内容都一样，即内容是静态的，没有变化的。图 2-13 展示了这个浏览过程。

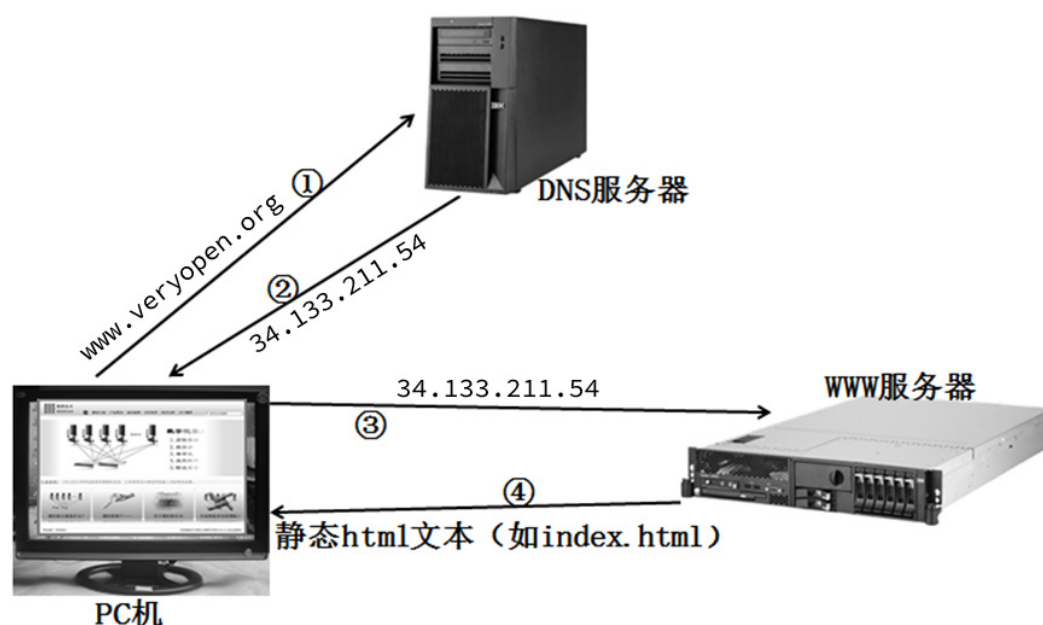


图 2-13 浏览静态网页的过程

我们在浏览器里输入网址（域名）`www.verypen.org` 并按 Enter 键，马上就看到网页的内容，图 10.11 中的①、②、③、④步是计算机自动完成的。域名便于人们记忆，但是计算机实际上还是通过 ip 地址访问网站服务器的，DNS 服务器的主要功能就是登记域名与 ip 地址的对应关系，当有人通过域名查询 ip 时返回对应的 IP 地址。

动态网页是服务器上的程序，此程序可以用 C、bash、php、Java、Python 等编程语言编写，人们更热衷于采用 php 或 python 来编写动态网页文件。当用户浏览某个动态网页文件时，服务器临时去执行这个网页文件（实际上就是程序），执行的结果就是生成一份完整的静态网页文件，然后服务器再把这个 HTML 文件发给用户，只不过此 HTML 文件保存在内存里，而不是硬盘上。由于用户看到的内容是程序执行的结果，程序可以访问数据库并把

数据库里的内容输出，因此用户浏览同一个网页可能每次看到的内容不同。图 2-14 展示了用户浏览动态网页的过程。

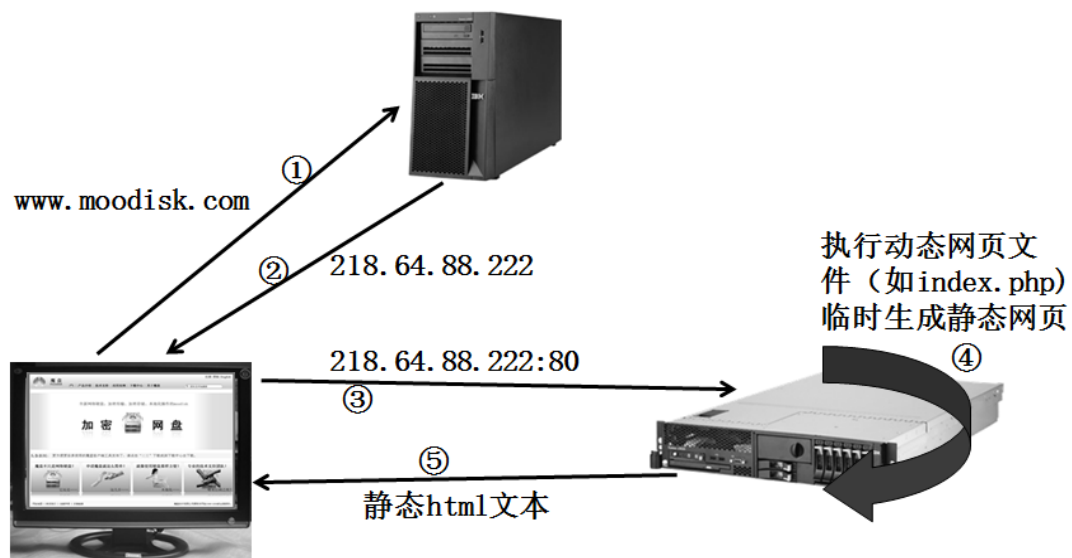


图 2-14 浏览动态网页的过程

LAMP (Linux + Apache + MariaDB + Php/Python) 网站套件目前最流行，世界上超过一半的网站都是基于它搭建的。Apache (阿帕奇) 是 web 服务器，是完全开源的，可靠、速度快，俄罗斯人在 `apache` 的基础上开发出 `nginx` 服务器 (相应的建站套件就是 LNMP)，性能提升不少，俄罗斯访问量最大的几个网站都是采用 `nginx`，`apache` 和 `nginx` 配置文档是相互兼容的。MariaDB 是一个短小精悍的开源数据库，为 MySQL 的后继版本。`php` 是非常流行的用于编写动态网页的脚本语言，其他的比较流行的脚本语言还有 `python`、`perl` 等，facebook、优酷、google 等大型网站和 Odoo 系统都是采用 `python` 语言开发的。

2.5.2 搭建过程

1. 准备工作

以 `root` 用户登录 Linux，然后执行如下命令设置主机名为 `webserver.mine.com`：

```
[root@localhost ~]# hostnamectl set-hostname webserver.mine.com
[root@localhost ~]# echo "127.0.0.1 webserver.mine.com" >> /etc/hosts
[root@localhost ~]# shutdown -r now
```

不妨假设本机的 Ip 为 192.168.192.20。

2. 安装 apache

```
[root@localhost ~]# dnf install httpd -y                                     #安装 apache
[root@localhost ~]# systemctl --now enable httpd                         #激活并启动服务
[root@localhost ~]# ss -antpl | grep httpd                               #检查监听的端口
LISTEN  0      511      *:80      *:~      users:((("httpd",pid=2860,fd=4),("httpd",pid=2859,fd=4),("httpd",pid=2858,fd=4),("httpd",pid=2856,fd=4))

[root@webserver ~]# ps -ef|grep httpd
root      2856      1    0 19:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2857     2856    0 19:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2858     2856    0 19:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2859     2856    0 19:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    2860     2856    0 19:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
```

可见监听了端口 80，启动了四个 httpd 进程，这样就增大了并发性。是 apache 用户运行 httpd 进程，在安装 apache 时会自动创建 apache 用户。我们直接浏览 <http://192.168.192.20> 就可以看到如图 2-15 所示的默认静态页面了。

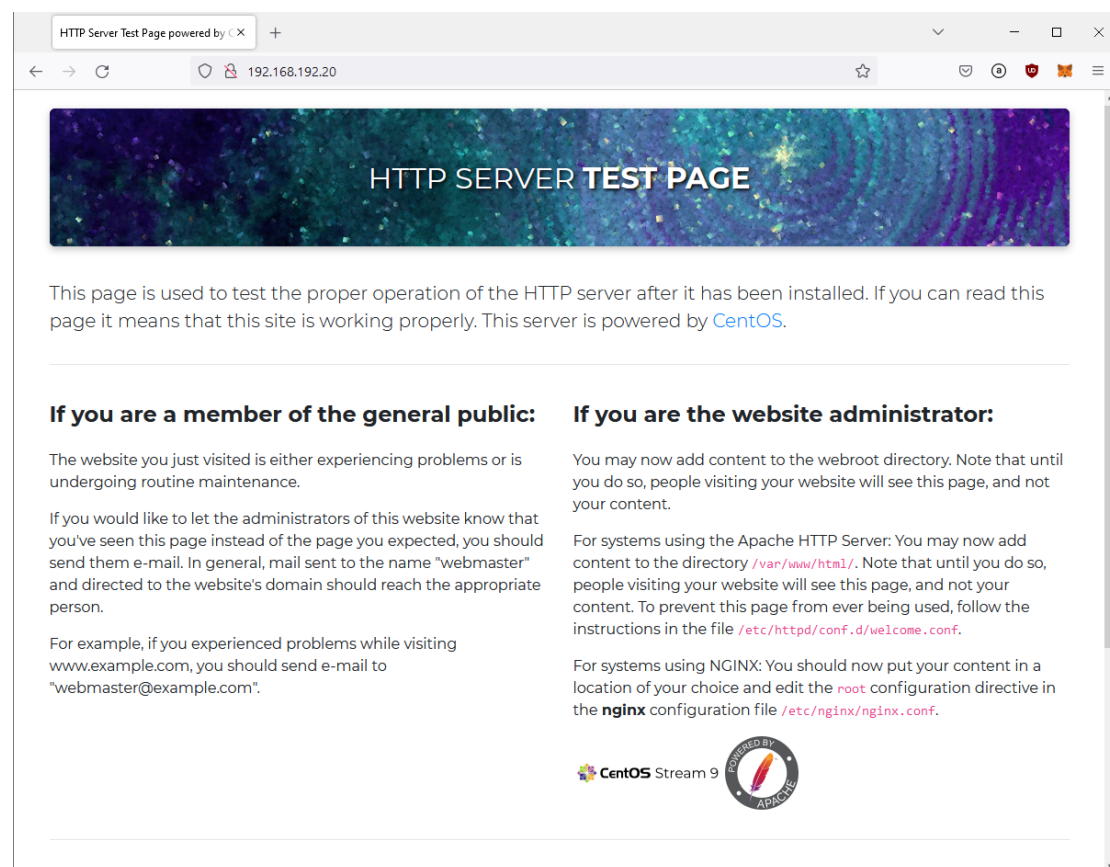


图 2-15 Apache 安装后的默认页面

默认网页的目录是 /var/www/html，静态网页文件是 index.html。apache 的主配置文件为 /etc/httpd/conf/httpd.conf，网站的配置文件在 /etc/httpd/conf.d 目录中。好了，现在用 vim 编辑 /var/www/html/index.html 文件，内容改为：

```
<h1>Hello Apache!</h1>
```

然后刷新页面，可以看到 Hello Apache!，参见图 2-16。

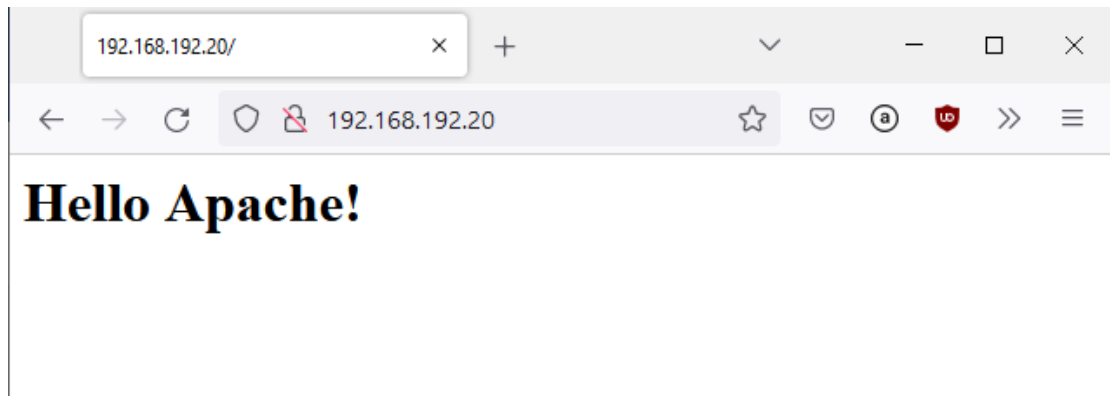


图 2-16 我写的第一个静态页面

3. 安装 php

```
[root@webserver ~]# dnf install php php-mysqlnd -y #安装 php
[root@webserver ~]# php -v #查看 Php 的版本号
PHP 8.0.20 (cli) (built: Jun 8 2022 00:33:06) ( NTS gcc x86_64 )
Copyright (c) The PHP Group
Zend Engine v4.0.20, Copyright (c) Zend Technologies
    with Zend OPcache v8.0.20, Copyright (c), by Zend Technologies
[root@webserver ~]# systemctl restart httpd #重启 web 服务
```

再在 /var/www/html 目录下编辑动态网页文件 info.php，内容如下：

```
<?php
phpinfo();
?>
```

然后浏览网址 <http://192.168.192.20/info.php>，应该看到如图 2-17 所示的画面，表示刚刚搭建的网站支持 php 动态网页。

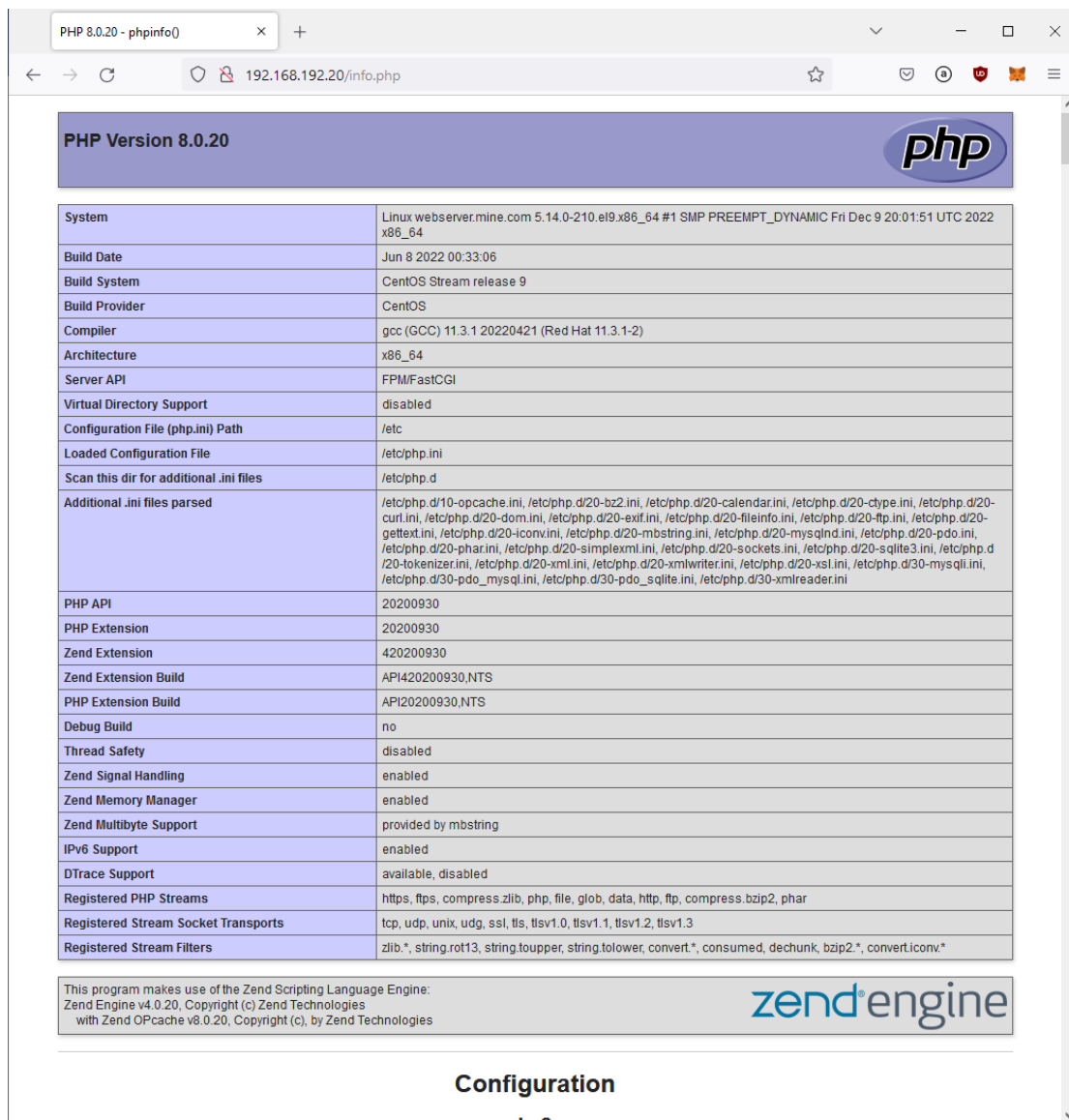


图 2-17 php 动态网页

4. 安装 MariaDB 数据库

```
[root@webserver ~]# dnf install mariadb-server -y
[root@webserver ~]# systemctl --now enable mariadb
[root@webserver ~]# mariadb-secure-installation
Enter current password for root (enter for none):
Switch to unix_socket authentication [Y/n] n
Change the root password? [Y/n] n
Remove anonymous users? [Y/n]
Disallow root login remotely? [Y/n]
Remove test database and access to it? [Y/n]
Reload privilege tables now? [Y/n]
```

#安装数据库
#激活并启动数据库
#配置

然后看看能否登录到数据库，执行下面的命令：

```
[root@webserver ~]# mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
```

```

Server version: 10.5.16-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database           |
+-----+
| information_schema |
| mysql               |
| performance_schema |
+-----+
3 rows in set (0.000 sec)

MariaDB [(none)]> use mysql;

Database changed
MariaDB [mysql]> quit
Bye

```

如果出现提示行 `MariaDB [(none)]>` 表明登录成功，输入 `sql` 语句 `show databases` 查看已有的数据库，输入语句 `use mysql` 打开 `mysql` 数据库，输入语句 `quit` 退出数据库。

5. 新建一个网站

一台计算机可以放置多个网站，每一个网站称为虚拟主机，不同的网站通过不同的域名来访问，每一个网站对应磁盘上单独的目录，比如 `www.weisuan.com` 域名的网页文件放在 `/var/www/weisuancom` 目录下，`www.veryopen.org` 对应 `/var/www/veryopenorg` 目录，等等。接下来我们来配置一个虚拟主机：对应 `www.weisuan.com` 域名，目录是 `/var/www/weisuancom`。

新建配置文件 `/etc/httpd/conf.d/weisuancom.conf`，输入如下内容：

```

<VirtualHost *:80>
    ServerName www.weisuan.com
    DocumentRoot /var/www/weisuancom
    DirectoryIndex index.php index.html
</VirtualHost>

```

这个配置文件定义了网站域名与网站目录的对应关系，即当用户浏览 `www.weisuan.com` 时，服务器会到 `/var/www/weisuancom` 目录中寻找相应的网页文件并发回给用户。

```
[root@webserver ~]# mkdir /var/www/weisuancom #创建网站对应的目录
```

编辑 `/var/www/weisuancom/index.php` 文件，输入如下的内容：

```

<?php
echo "<h1>这是我的动态网站，耶! </h1>";
?>

```

执行下面的命令使网站配置生效：

```
[root@webserver ~]# systemctl reload httpd
```

在用户计算机上临时对域名 `www.weisuan.com` 解析到服务器的 `ip` 地址

192.168.192.20 , 如 果 是 Windows , 在 文 件
c:\windows\System32\drivers\etc\hosts 末尾增加如下所示的一行:

```
192.168.192.20 www.weisuan.com
```

如果是 Linux, 则在/etc/hosts 文件中增加同样的一行内容即可。

最后浏览 <http://www.weisuan.com> 网站就能看到“这是我的动态网站, 耶!”。

大家可参照上面的方法在同一台计算机上创建更多的网站虚拟主机。

6. 编写访问数据库的页面

在数据库中建库建表建用户:

```
[root@webserver ~]# mariadb
.....
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE abc_db CHARACTER SET utf8;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> use abc_db;
Database changed
MariaDB [abc_db]> CREATE TABLE student(
  -> id CHAR(15),
  -> name CHAR(12),
  -> age SMALLINT
  -> );
Query OK, 0 rows affected (0.004 sec)

MariaDB [abc_db]> INSERT INTO student VALUES('2021098223','张三',20);
Query OK, 1 row affected (0.002 sec)

MariaDB [abc_db]> INSERT INTO student VALUES('2021098085','李四',21);
Query OK, 1 row affected (0.002 sec)

MariaDB [abc_db]> SELECT * FROM student;
+-----+-----+-----+
| id      | name  | age  |
+-----+-----+-----+
| 2021098223 | 张三  | 20   |
| 2021098085 | 李四  | 21   |
+-----+-----+-----+
2 rows in set (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON abc_db.* TO 'alice'@'localhost' IDENTIFIED BY 'A1b2C3';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON abc_db.* TO 'alice'@ '%' IDENTIFIED BY 'A1b2C3';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> exit
Bye
```

创建了库 abc_db, 表为 student, 表中插入了两条记录, 创建了数据库用户 alice, 密码 A1b2C3。

下面再编辑 php 页面:

```
[root@webserver ~]# vim /var/www/weisuancom/index.php
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>我的第一个动态页面</title>
</head>

<body>
  <?php
    $host = "localhost";
    $user = "alice";
    $password = "A1b2C3";
    $db = "abc_db";
    $port = 3306;

    $dbconnect = mysqli_connect($host, $user, $password, $db, $port);
    if ($dbconnect->connect_error) {
      die("链接数据库失败: " . $dbconnect->connect_error);
    }
    mysqli_set_charset($dbconnect, "utf8");
    ?>
    <table border="1" align="center">
      <caption>学生信息表</caption>
      <thead>
        <tr>
          <th>学号</th>
          <th>姓名</th>
          <th>年龄</th>
        </tr>
      </thead>
      <tbody>
        <?php
          $query = mysqli_query($dbconnect, "SELECT * FROM student") or
die(mysqli_error($dbconnect));
          while ($row = mysqli_fetch_array($query)) {
            echo "
              <tr>
                <td>{$row['id']}</td>
                <td>{$row['name']}</td>
                <td>{$row['age']}</td>
              </tr>";
          }
        ?>
      </tbody>
    </table>
  </body>
</html>
```

再次浏览 www.weisuan.com, 效果如图 2-18。



图 2-18 访问数据库的页面

这样网站前端从数据库中拿到了数据了。

2.6 网络文件系统 NFS

实训目的：搭建 nfs 服务器，并让客户机使用。

2.6.1 NFS 介绍

网络文件系统（Network File System, NFS）构建于 IP 协议之上，由太阳微系统公司于 1984 年开发发展而来的，目的是在安装了 UNIX 操作系统的计算机之间实现磁盘文件共享，它是标准的 C/S 架构——NFS 服务器输出（export）让别人共享的目录，多个客户端挂载（mount）各自感兴趣的目录，并加载到本地文件空间当中。位于 VFS（虚拟文件系统）之下，这样就像访问本地硬盘上的文件那样方便简单，参见图 2-19。尽管 NFS 目前被普遍使用，但是它的原创者太阳微系统公司却不存在了，被 Oracle 公司收购，真所谓物是人非也！不过 NFS4.0 以前的版本存在性能上的瓶颈，NFS 目前发展到了 4.1 版本，这个版本已经是 pNFS 了，即并行 NFS。



图 2-19 NFS 服务器输出目录与客户端挂载

2.6.2 搭建 NFS

搭建一个 NFS 环境非常简单，只需配置 NFS 服务器即可，客户端不用任何改动。假设 nfs 服务器的 Ip 地址为 192.168.192.20。

1. 安装 NFS:

红帽 9 支持 NFS 4.1。默认 NFS 服务器已经安装好，使用命令 “rpm -qa nfs*” 查看已经安装的软件包，如果没有安装，就执行下面的命令进行安装：

```
[root@webserver ~]# dnf install nfs-utils -y

然后执行下面的命令激活 nfs-server 服务单元和 rpcbind 套接口单元，并启动它们：

[root@webserver ~]# systemctl --now enable nfs-server rpcbind.socket

[root@webserver ~]# cat /proc/fs/nfsd/versions                                #检查支持的 nfs 版本
-2 +3 +4 +4.1 +4.2                                                         #表明不再支持 2 版了
[root@webserver ~]# useradd nfsuser -u 1036 -g kvm                          #创建用户 nfsuser
```

2. 配置 NFS:

把需要输出的目录加到文件 /etc/exports 中，一行输出一个目录，每一行的格式如下，这样客户端就可以挂载这些被输出的目录了。

```
目录  [-(参数 1,参数 2,...)]  客户主机 1(参数 1,参数 2,...)  客户主机 2(参数 1,参数 2,...)  ...
```

配置文件中的每一行也就是要回答清楚下面三个问题：

- （1）想让其他计算机共享本机磁盘上的什么目录？
- （2）此目录允许哪些客户机访问？
- （3）这些客户机访问的权限是什么？

可选部分 “[-(参数 1,参数 2,...)]” 定义默认参数。/etc/exports 文件中的空行被忽略，“#” 字符右侧直到行尾的字符也被忽略。

至于如何指定客户机，可参考下表 2-2。

表 2-2 定义客户机的方式

序号	定义方式	说明
1	单个客户机	直接采用客户机的 ip 地址或者域名，比如 192.168.10.50
2	通配符	采用通配符*、?，如*.moodisk.com，即凡是域名包含“.moodisk.com”字符串的主机。再比如*表示所有的客户机
3	网段	指明处于同一个网段内的主机，如 192.168.10.0/24 或者 192.168.10.0/255.255.255.0 均可

常用的参数见表 2-3。

表 2-3 nfs 常用参数说明

序号	参 数	说 明
1	secure	使用小于 1024 的安全 TCP/IP 端口，默认参数
2	insecure	NFS 通过大于 1024 的端口发送请求
3	ro	客户机只能读（下载），这是默认参数
4	rw	可读可写（上传和下载），不过能否上传还受到本地文件系统权限的限制
5	async	数据到达 NFS 服务器内存时就反馈客户端“写成功”，也就是数据还没写到磁盘，如果系统崩溃会丢掉数据。高效和风险同在。
6	sync	数据正式写到了磁盘上才反馈给客户端“写成功”。这是默认参数
7	subtree_check	如果输出一个文件系统的某个子目录时，强制 NFS 检查父目录的权限
8	no_subtree_check	不检查父目录权限，这是默认参数
9	pnfs	启用 pNFS 扩展协议，默认参数。
10	no_pnfs	不启用 pNFS 扩展协议
11	root_squash	把客户端的 root 账号的 uid/gid 映射成服务器上的匿名账号 nobody——如果客户机用 root 上传文件，上传后文件的主人是 nobody。
12	no_root_squash	关闭 root_squash，即不做 root 到 nobody 的映射
13	all_squash	所有用户映射成由 anonuid 参数定义的用户，组映的组由 anongid 指定
14	no_all_squash	即不做映射，这是默认参数
15	anonuid=<uid>, anongid=<gid>	映射到 uid/gid 的账号，这个账号事先在 NFS 服务器上已经存在，即客户端往共享目录里新建文件时，文件主人的 UID 就是<uid>，组号是<gid>

比如我们把 /opt/nfs1 输出，并允许 192.168.192.0/24 网络上的电脑上传和下载文件：

```
[root@webserver ~]# echo "/opt/nfs1 192.168.192.0/24(rw,all_squash,anonuid=1036,anongid=36)" >>/etc/exports
[root@webserver ~]# systemctl stop nfs-server rpcbind.socket
[root@webserver ~]# systemctl start nfs-server rpcbind.socket
[root@webserver ~]# chown -R nfsuser:kvm /opt/nfs1
[root@webserver ~]# chmod -R 0755 /opt/nfs1
```

其中“192.168.192.0/24(rw,all_squash,anonuid=1036,anongid=36)”表示允许 192.168.192.0/24 网络上的任何电脑读写 nfs 服务上的/opt/nfs1 下的内容，且不管客户端采用什么用户写，最终在 nfs 服务器上都被转换为 nfsuser 用户。

3. 客户端使用 NFS:

客户机挂载 NFS 服务器上的一个被输出目录，统一用如下命令格式：

```
mount -t nfs <NFS 服务器>:<被输出目> <本地空目录>
```

启动另一台虚拟机，首先安装 nfs 客户端软件。

```
[root@another ~]# dnf install nfs-utils -y #需要安装 nfs 客户端
[root@another ~]# showmount -e 192.168.192.20 #查看输出的目录
Export list for 192.168.192.20:
/opt/nfs1 192.168.192.0/24

[root@another ~]# mount -t nfs4 192.168.192.20:/opt/nfs1 /mnt #挂载

如果报错: mount.nfs4: Stale file handle, 说明/mnt 上已经挂载了文件系统，先执行 umount /mnt 卸载即可。

[root@another ~]# df -T #查看挂载的分区
文件系统 类型 1K-块 已用 可用 已用% 挂载点
devtmpfs devtmpfs 4096 0 4096 0% /dev
tmpfs tmpfs 1992220 0 1992220 0% /dev/shm
tmpfs tmpfs 796888 9924 786964 2% /run
/dev/nvme0n1p2 xfs 19007488 5402332 13605156 29% /
tmpfs tmpfs 398444 96 398348 1% /run/user/0
192.168.192.20:/opt/nfs1 nfs4 19007488 5402112 13605376 29% /mnt
```

然后往/mnt 目录拷贝文件实际上就是把文件上传到 NFS 服务器的/opt/nfs1 目录，把 /mnt 中的文件拷贝出来实际上就是下载文件。

```
[root@another ~]# cp /etc/profile /mnt/ #上传文件到 NFS 服务器
[root@another ~]# mkdir /mnt/another
[root@another ~]# umount /mnt #卸载
```

如果客户端不再使用 NFS 文件系统，就可以卸载它，例如上面的最后一条命令。可以把 NFS 文件系统挂载任何一个空目录上。

这时你应该可以在 nfs 服务器上的/opt/nfs1 中看到上传的文件 profile 和创建的目录 another，且文件的主人是 nfsuser，组是 kvm:

```
[root@webserver ~]# ll /opt/nfs1/
总用量 4
drwxr-xr-x 2 nfsuser kvm 6 12月 17 10:19 another
-rw-r--r-- 1 nfsuser kvm 1899 12月 17 10:19 profile
```

2.7 搭建邮件系统

实训目的：在自己的计算机上搭建一个邮件系统，能收、发和阅读邮件，而且能防病毒，过滤垃圾邮件。

假设：虚拟机的 Ip 地址为 192.168.192.20。

2.7.1 邮件系统简介

邮件系统比较复杂，涉及收/发邮件、过滤病毒和垃圾邮件等，建议大家去网上搜索并了解邮件系统的原理。这里建议采用 iRedMail 邮件安装和配置脚本，可以从这里 <http://www.iredmail.com/download.html> 下载，然后直接运行这个下载的 bash 程序，大概 20 分钟就自动搭建了一个全功能的企业级邮件系统了。

我们在 CentOS 9 上操作。首先把虚拟机回滚到最初的快照。

2.7.2 搭建邮件系统

确保计算机连上互联网，然后执行如下的命令进行安装：

```
[root@localhost ~]# hostnamectl set-hostname mx.veryopen.org           #主机名改成mx.veryopen.org
[root@localhost ~]# su -
[root@mx ~]# echo "127.0.0.1 mx.veryopen.org" >> /etc/hosts           #域名解析到127.0.0.1
[root@mx ~]# wget https://github.com/iredmail/iRedMail/archive/refs/tags/1.6.2.tar.gz      #下载
```

最近 github.com 被封，可能没法下载。请到 www.veryopen.org 网站的“资源下载”目录中去下载 iRedMail-1.6.2.tar.gz。

```
[root@mx ~]# tar -xf iRedMail-1.6.2.tar.gz                               #解压解包
[root@mx ~]# cd iRedMail-1.6.2                                           #进入解压后产生的目录
[root@mx iRedMail-1.6.2]# bash iRedMail.sh                               #开始部署邮件系统
```

安装邮件系统，按照向导一步步往下走即可，建议的选择是：采用 MariaDB 数据库，涉及的密码比较多，建议设成相同的。交互问答大致如下：

```
* Storage base directory:      /var/vmail
* Mailboxes:
* Daily backup of SQL/LDAP databases:
* Store mail accounts in:      MariaDB
* Web server:                   Nginx
* First mail domain name:      weisuan.com
* Mail domain admin:           postmaster@weisuan.com
* Additional components:       Roundcubemail netdata iRedAdmin Fail2ban
< Question > Continue? [y|N]y
```

接下来还会问几个问题，直接回车就可以了，部署成功后的总结内容如下：

```

*****
* URLs of installed web applications:
*
* - Roundcube webmail: https://mx.veryopen.org/mail/           <-----用户使用邮件的页面
* - netdata (monitor): https://mx.veryopen.org/netdata/       <-----邮件系统监控页面
*
* - Web admin panel (iRedAdmin): https://mx.veryopen.org/iredadmin/ <-----邮件系统管理页面
*
* You can login to above links with below credential:
*
* - Username: postmaster@weisuan.com
* - Password: *****
*
*****

```

最后重启电脑。

```
[root@mx iRedMail-1.6.2]# shutdown -r now #重启电脑
```

最后的管理网址、用户名和密码全部保存在 `iRedMail.tips` 文件中。`iRedMail` 是一个企业级邮件系统搭建脚本程序，主要采用了 `postfix`（邮件传输代理）、`amavisd`、`spamassassin`（反垃圾邮件）、`clamav`（反病毒）、`roundcubemail`（邮件 web 客户端）、`Dovecot`（IMAP 和 POP3 邮件服务器）等软件包。

2.7.3 使用邮件系统

1. 邮件系统管理后台

浏览网址 `https://192.168.192.20/iredadmin` 进入登录界面，见图 2-20。

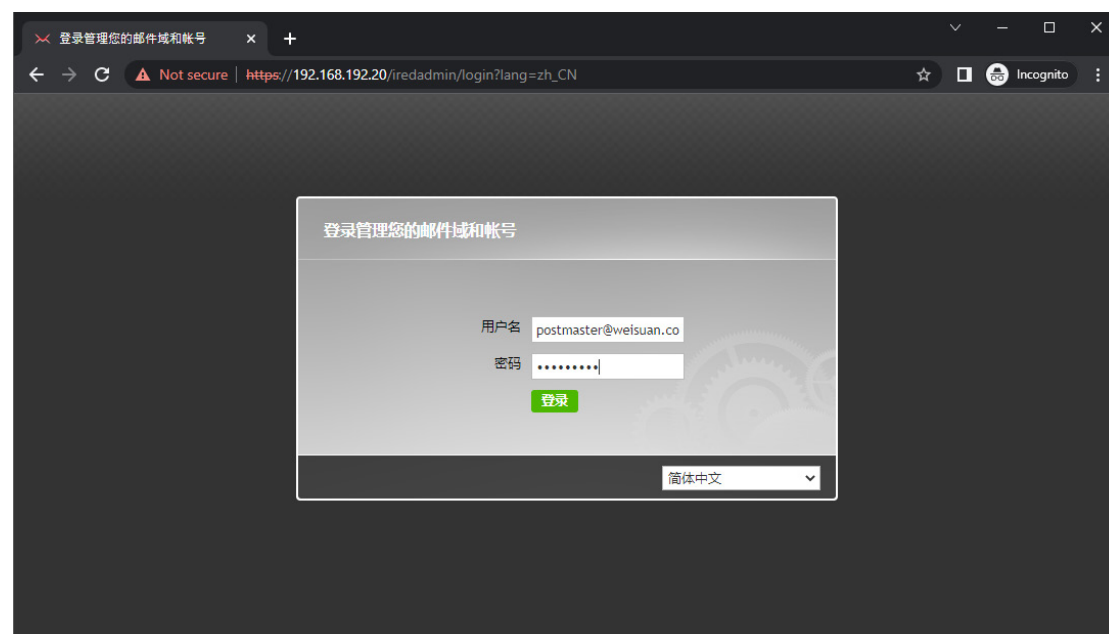


图 2-20 邮件系统的管理页面

用户名和密码在 `iRedMail.tips` 文件可以找到。在邮件管理后台可以管理邮件域、管理

每个域的管理员、管理邮件用户等。

我们再创建一个域 `veryopen.org`，点击“添加”下的“域名”，参考图 2-21。

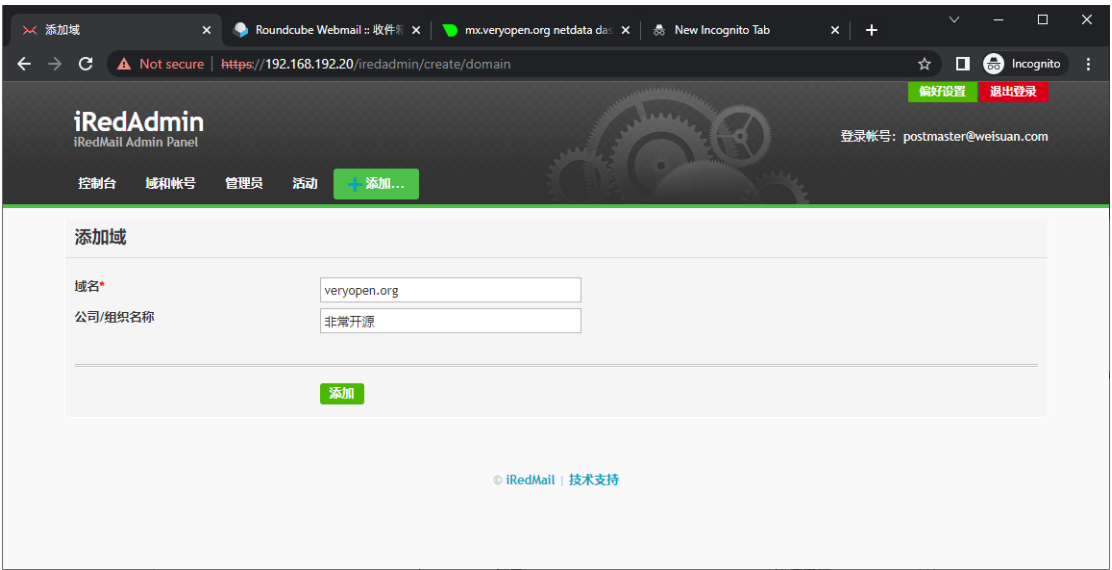


图 2-21 新建域

在域 `weisuan.com` 中新建一个邮件账户，点击“添加”下面的“用户”，参考图 2-22。

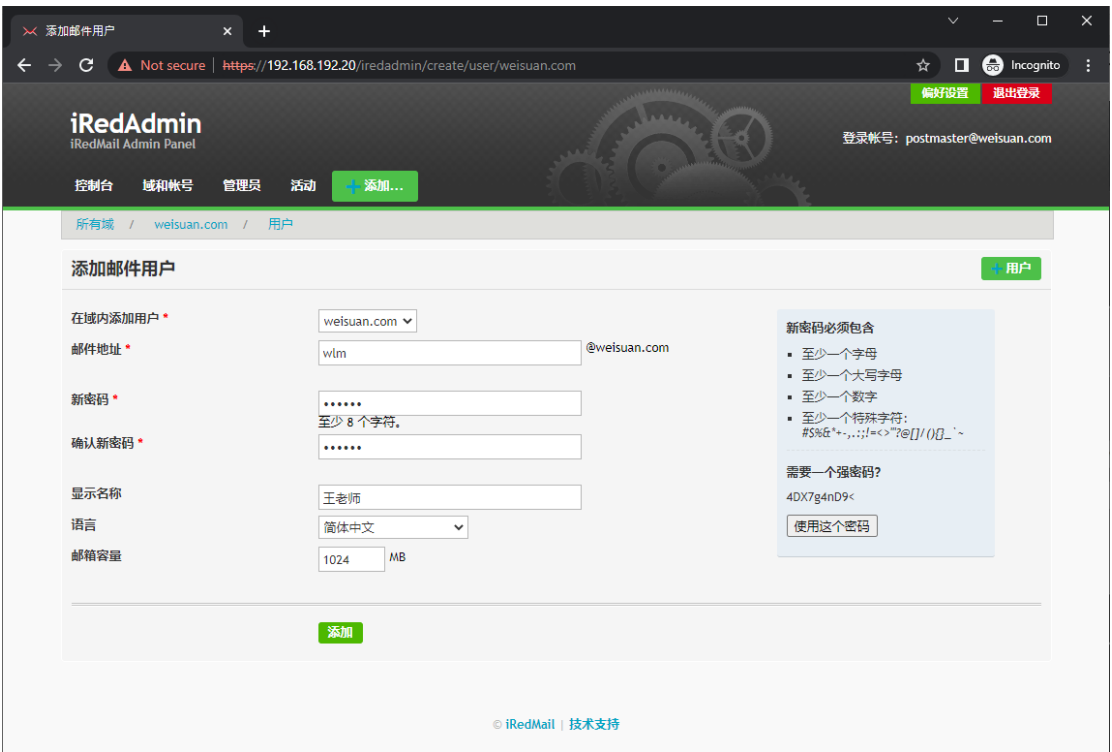


图 2-22 新建邮账号

注意密码不能少于 8 个字符，且要包含大小写字母、数字和特殊字符，这样我们新建了一个

邮件账号 `wlm@weisuan.com`，下面再看看这个账号如何收发邮件。参照类似方法在 `veryopen.org` 域中新建邮件账号 `alice@veryopen.org`。这样这两个邮箱可以互相收发邮件了，不过要确保文件 `/etc/hosts` 中存在以下两行：

```
127.0.0.1 mx.veryopen.org
127.0.0.1 mx.weisuan.com
```

2. 用户收发邮件

浏览网址 `https://192.168.192.20/mail`，如图 2-23 所示。

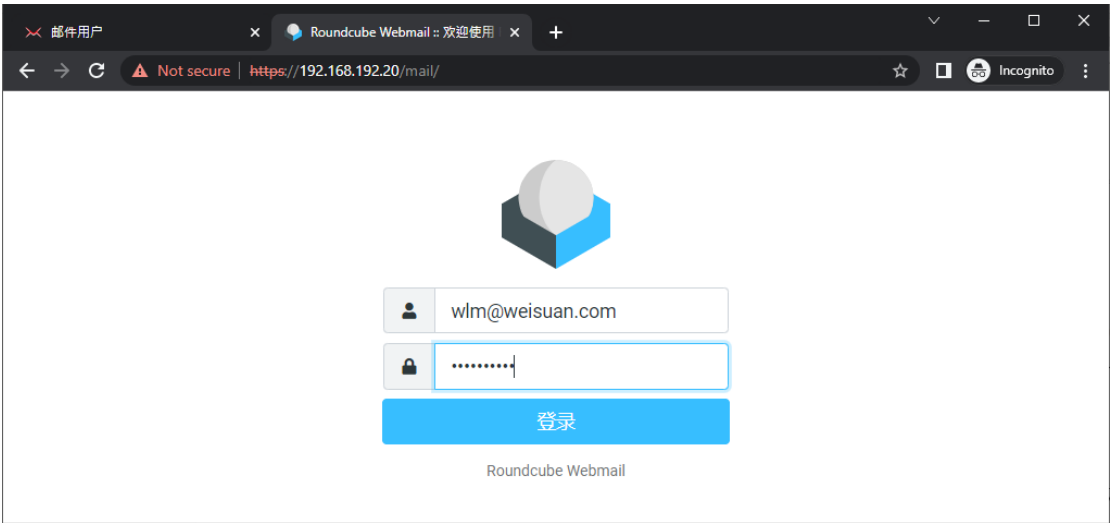


图 2-23 用户邮箱登录页面

登录后的画面如图 2-24 所示。

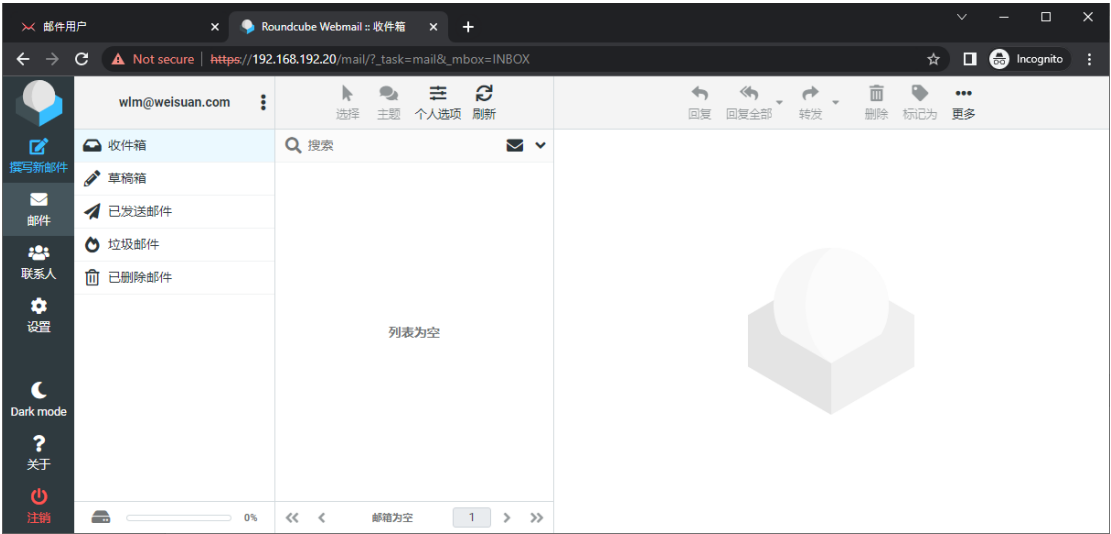


图 2-24 进入到用户邮箱

由于是新邮件用户，所以目前没有任何邮件。写一封邮件先，发到互联网上的 qq 邮箱，参考图 2-25 所示。

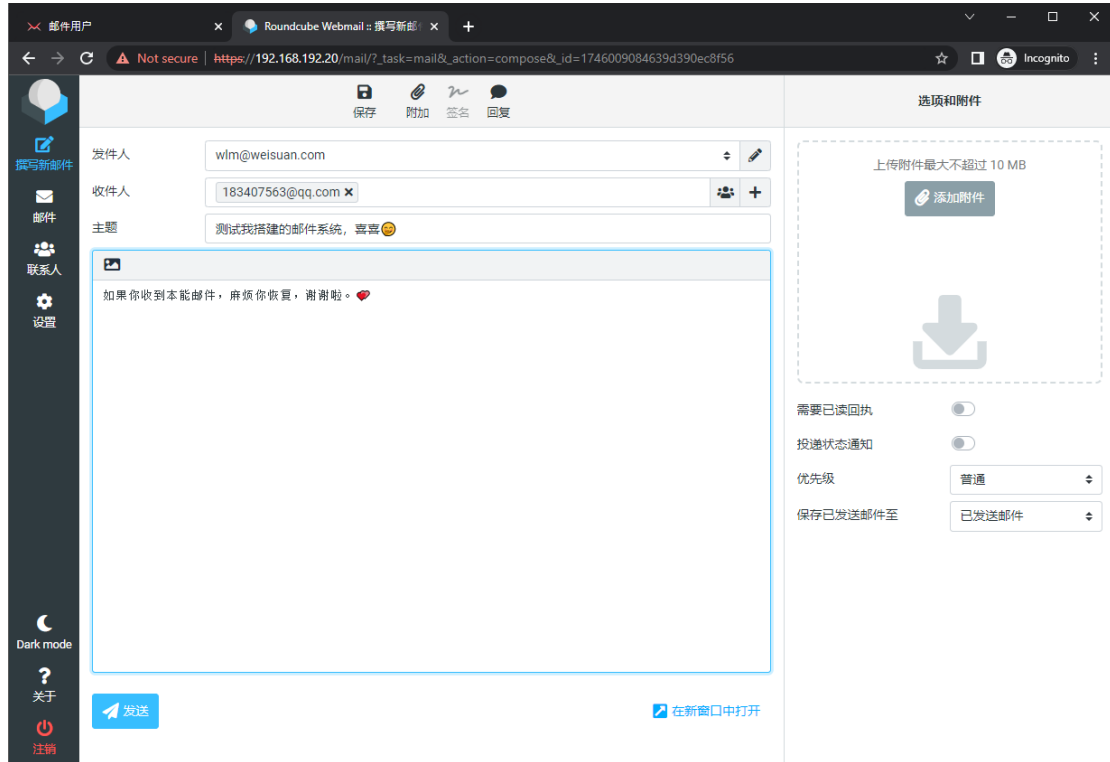


图 2-25 发一份新邮件

邮件能发送，对方确实能收到，但是对方回复后，邮件到达不了本机，因为对方没法解析域名 `mx.weisuan.com`，所以只有把邮件服务器放在互联网上（比如在阿里云的虚拟机里搭建邮件系统），然后申请正式的域名 `weisuan.com`，然后在 DNS 服务器中登记一条 A 记录——建立 `mx.weisuan.com` 与邮件系统的 ip 关系。

接下来可以验证：`wlm@weisuan.com` 可以向 `alice@veryopen.org` 发邮件，`alice@veryopen.org` 收到邮件后也能回复，而且 `wlm@weisuan.com` 能收到回复的邮件，因为 `mx.weisuan.com` 和 `mx.veryopen.org` 都在本机的 `/etc/hosts` 文件中做了解析。

浏览 `https://192.168.192.20/netdata/` 进入邮件服务器性能监控页面，参见图 2-26 所示，不过选要输入用户名 `postmaster@weisuan.com` 和密码（搭建邮件系统是设置的密码）。

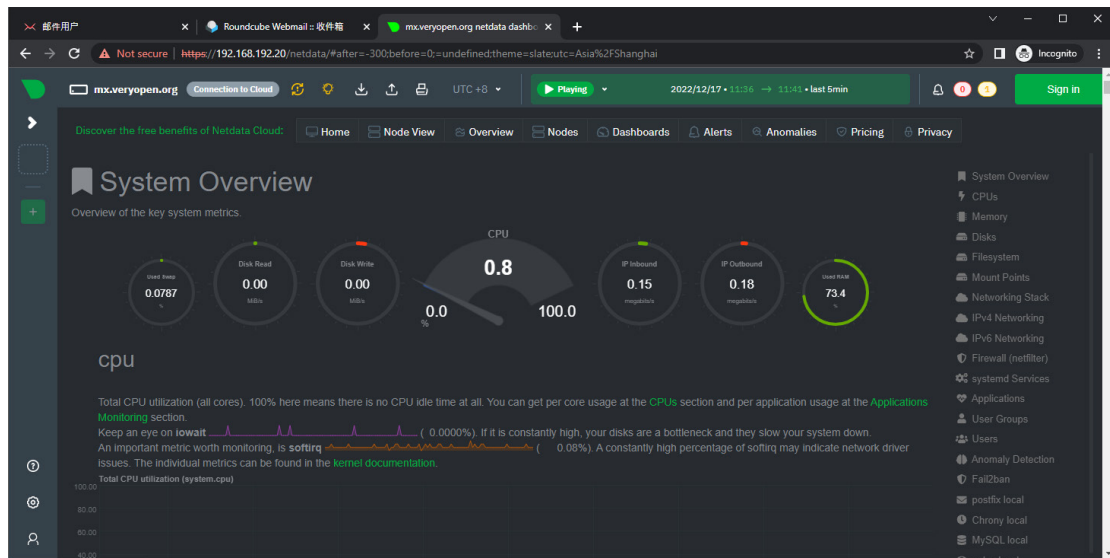


图 2-26 邮件服务器性能监控

2.8 SAMBA 服务

samba 在 Linux 上实现 smb 协议，此协议主要在计算机之间（主要与 Windows 机器）共享文件和打印机。

下面的实验把 samba 配置成独立服务器，与 Windows 之间共享目录。在 CentOS 9 机器上操作。

1. 安装

```
[root@localhost ~]# dnf install samba samba-client #安装
[root@localhost ~]# systemctl --now enable smb nmb #激活并启动 smb、nmb 单元
[root@localhost ~]# smb --version #显示版本号
Version 4.17.2
```

如果开启了防护墙，请执行下面的命令让防火墙通过 smb 或者干脆关闭防火墙。

```
firewall-cmd --permanent --add-service=samba
systemctl reload firewalld
```

主配置文件是 `/etc/samba/smb.conf`，服务单元分别是 `smb.service` 和 `nmb.service`，smb 协议的服务程序为 `/usr/sbin/smbd`，它监听 tcp 端口 139 和 445，NetBIOS 名字服务器程序为 `/usr/sbin/nmbd`，它监听 udp 端口 137 和 138。

2. 配置

(1) Windows 能匿名访问 Linux 上的目录。

```
[root@localhost ~]# mkdir -p /srv/samba/anonymous
[root@localhost ~]# chmod 0777 /srv/samba/anonymous
[root@localhost ~]# chown nobody:nobody /srv/samba/anonymous
[root@localhost ~]# cp /etc/samba/smb.conf /etc/samba/smb.conf.orig
```

编辑 `/etc/samba/smb.conf`，确保存在以下行：

```
[global]
workgroup = WORKGROUP
hosts allow = 127.0.0.1 192.168.192.0/24
map to guest = Bad User
netbios name = CentOS9
[Anonymous]
comment = Anonymous File Server Share
path = /srv/samba/anonymous
writeable = Yes
guest ok = Yes
browseable = Yes
```

注意：192.168.192.0/24 就是虚拟机网卡所在的网络。

至于各个参数的作用，请参考在线文档 `man 5 smb.conf`，文件中的其他参数不修改。运行下面的命令检查配置是否正确：

```
[root@localhost ~]# testparm #检查配置是否合法
[root@localhost ~]# systemctl restart smb.service nmb.service #重启 smb 和 nmb 单元
```

然后在 Windows 这边，按 win+r，然后输入\\192.168.192.20，按 Enter 键后即可看到 Linux 共享的目录 anonymous（参见图 2-27），假设安装了 samba 的 Linux 机器的 ip 地址为 192.168.192.20。

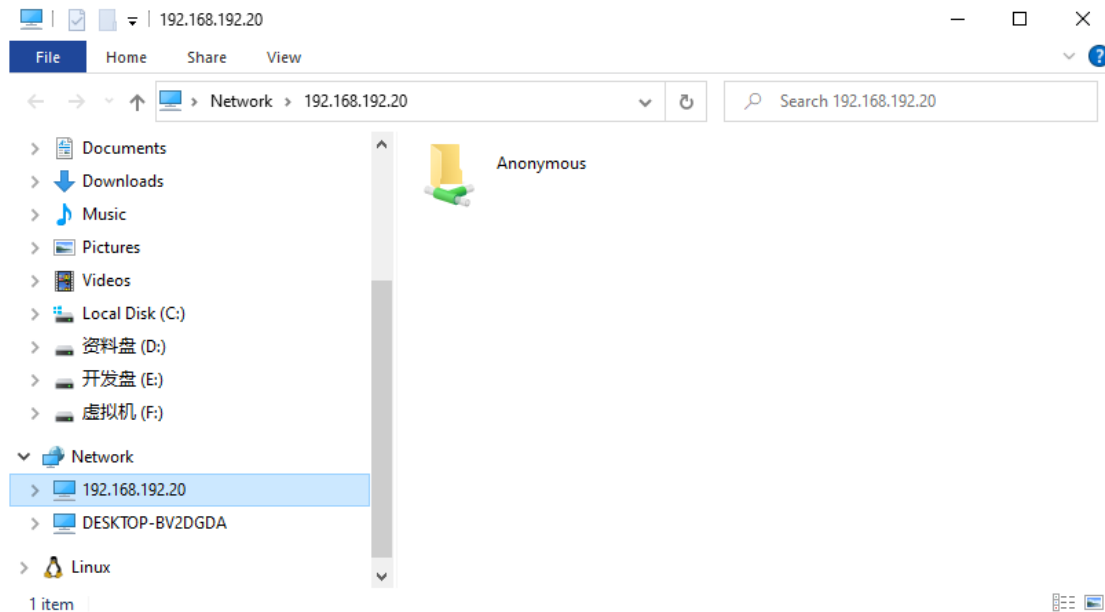


图 2-27 Windows 10 与 Linux 通过 samba 共享

然后就可以往 Anonymous 目录中拷贝文件了。

（2）Windows 通过用户和密码访问 Linux 上的目录。

```
[root@localhost ~]# groupadd smbgrp #创建组 smbgrp
[root@localhost ~]# useradd -g smbgrp smbuser #创建用户 smbuser
[root@localhost ~]# smbpasswd -a smbuser #把系统用户 smbuser 加到 smbpasswd 文件中，并设置密码
[root@localhost ~]# mkdir -p /srv/samba/secure
[root@localhost ~]# chmod 0770 /srv/samba/secure
[root@localhost ~]# chown root:smbgrp /srv/samba/secure
```

编辑/etc/samba/smb.conf，在末尾添加如下内容：

```
[Secure]
comment = Secure File Server Share
path = /srv/samba/secure
valid users = @smbgrp
guest ok = No
writable = Yes
browsable = Yes
```

检查配置合法性，并重启 smb 和 nmb 服务。

```
[root@localhost ~]# testparm
[root@localhost ~]# systemctl restart smb.service nmb.service
```

这时就可以在 Windows 访问 Linux 共享出来的目录了，只不过要输入用户名 smbuser 和

密码，参见图 2-28。

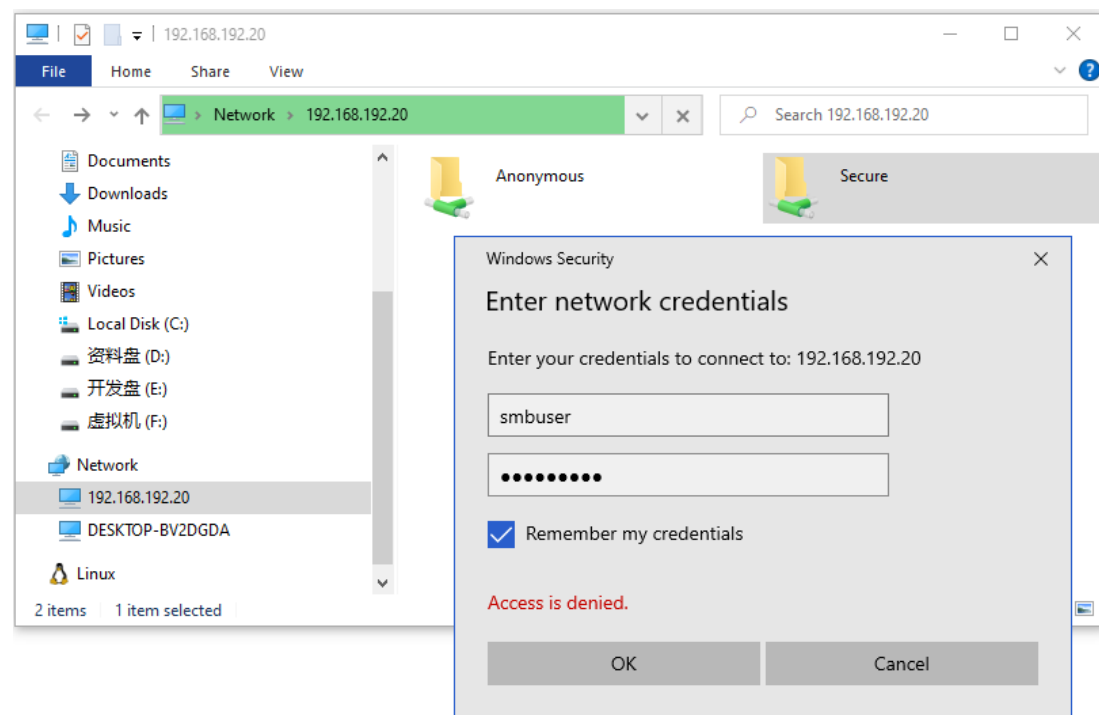


图 2-28 Windows 10 与 Linux 通过 samba 共享（需要密码）

2.9 防火墙

2.9.1 基础知识

绝大多数防火墙工作在网络层，其主要任务就是设“关卡”来审查流经防火墙的数据包，将原来完全互通的网络分割成不同的区域（zone）——内网区、外网区、公共区、DMZ 区、信任区等等。通过在防火墙内设置一系列的审查规则来判断哪些数据包允许通过，哪些数据包必须拦截，哪些数据包经过修改后放行，等等。

一个服务采用“ip 地址:端口”来唯一指定，比如“192.168.0.10:22”定义了 ip 地址为 192.168.0.10 这台计算机上的 ssh 服务（此服务监听端口 22）。另一个重要的概念是建立 tcp 链接需要三次握手，第一次握手时首先“伸手”的一方称为发起方，一般为客户端，比如 sshd 服务启动后监听 22 号端口，被动地等待客户端发起握手以便建立登录连接。图 2-29 描述了 tcp/ip 的通讯过程。

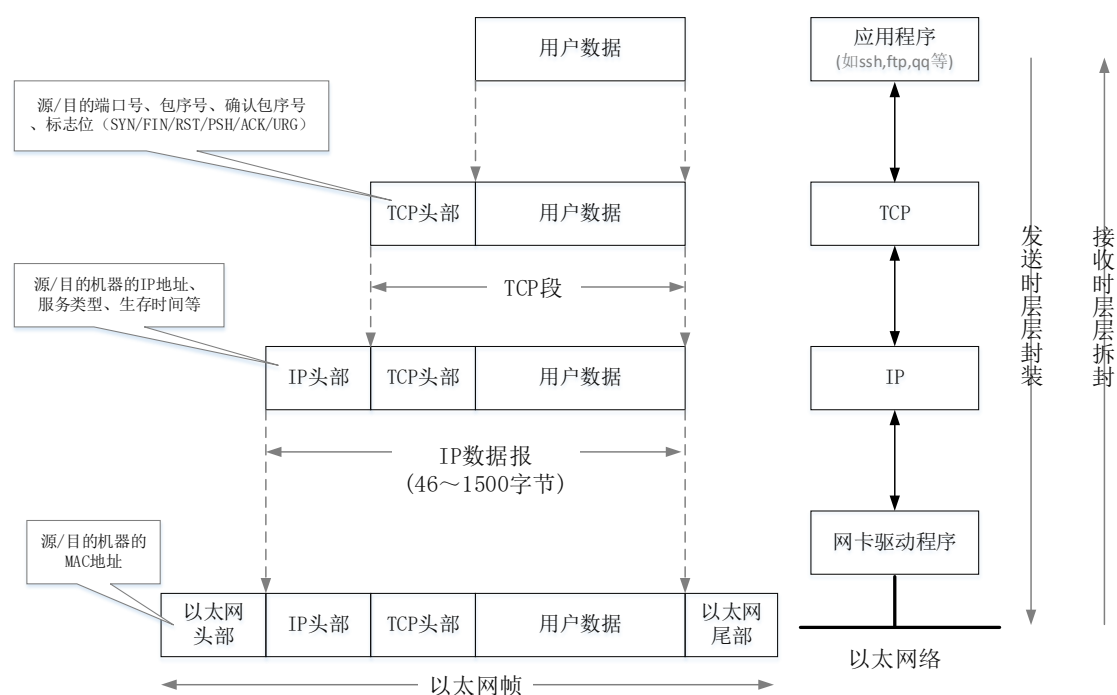


图 2-29 tcp/ip 通讯示意图

发送数据时从上（应用程序）到下（以太网网络）层层封装，相反，接收数据就从下到上层层拆封。

Linux 的 netfilter 防火墙模块工作在网卡驱动程序的上面，所以能看到 IP 数据报，报文里面的 IP 头部和 TCP 头部包含了重要的可用于制定审查规则的信息，如源/目的端口

号、源/目的机器的 IP 地址、包序号、标志位等等。

2.9.2 内核中的 netfilter 模块

在 Linux 系统中，tcp/ip 协议栈、netfilter nft 子系统、nf_tables 模块和 firewallld 的关系见图 2-30 所示。

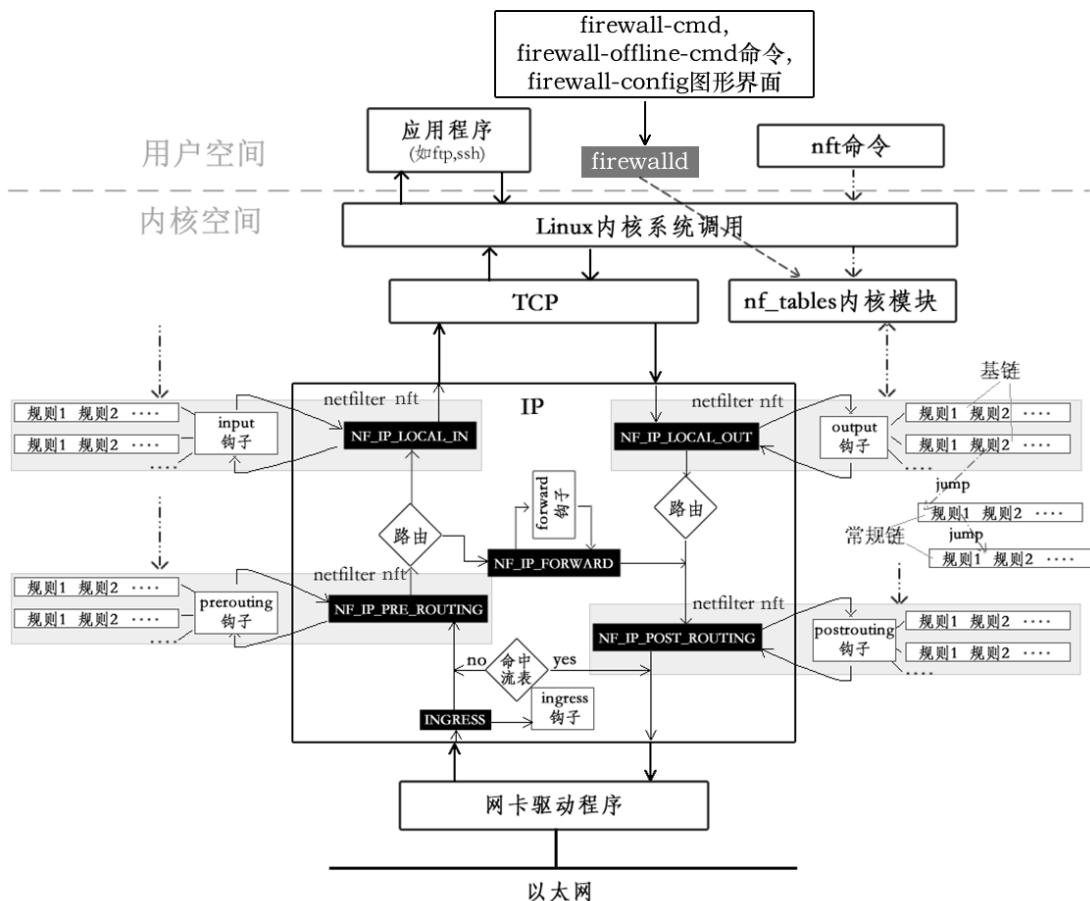


图 2-30 tcp/ip、netfilter 和 nf_tables 的关系

现代 Linux 操作系统的防火墙默认后端是 nftables，而不再使用 iptables 了，当然用户可以自行选择后端。nftables 比 iptables 相比具有明显的优势，比如速度就快很多。Netfilter 是 Linux 2.4.x 内核开始引入的一个通用的、抽象的框架，它提供一整套的钩子函数（hook）的管理机制，使得诸如数据包过滤、网络地址转换（NAT）和基于协议类型的连接跟踪的用户编程变得简化。Netfilter 在 TCP/IP 协议栈的 IP 层中的数据包路径上设置若干陷阱（如图 10.25 中的 INGRESS、NF_IP_PRE_ROUTING、NF_IP_LOCAL_IN、NF_IP_FORWARD、NF_IP_LOCAL_OUT 和 NF_IP_POST_ROUTING），

数据包一旦进入陷阱，就会被预先注册的各个钩子处理，钩子根据防火墙规则决定每个包的命运，要么放行，要么销毁，要么修改后放行，要么旁路等。

`nftables` 具体包含 `nf_tables` 内核模块和 `nft` 用户工具，`nf_tables` 模块构建于 `netfilter` 框架之上，它其实就是一台虚拟机，由 `nft` 把用户输入的配置内容（源代码）编译成 `nf_tables` 认识的指令并喂给它，`nf_tables` 执行这些指令并把结果输出到 `netfilter`。用户采用 `nft` 工具编写防火墙规则相当于是编程——一种新的计算机编程语言，不过比 C 语言要简单得多。

2.9.3 配置规则

在 `nftables` 中，一张表是一个名字空间，用作名字隔离，用户可以任意创建表，表中可包含链、集合、状态对象和变量等。一个表必须依附于一个具体的地址族（`ip`, `ip6`, `inet`, `arp`, `bridge`, `netdev`），因此表的唯一性引用就是“地址族 表名”，比如“`ip filter1`”、“`arp myRaw`”。链是规则的容器，有两种类型的链，一种称为基链，另一类称为常规链，基链必须挂在 `netfilter` 的某个钩子上，一共有 5 个钩子，分别是 `prerouting`、`input`、`forward`、`output`、`postrouting` 和 `ingress`（参见图 10.25），基链能看到 TCP/IP 协议栈上的数据包，被挂载在同一个钩子上的多条基链通过各自的优先级来排序处理。基链的优先级为有符号整数（数越小优先级越高），挂在同一个钩子上的多条基链，优先级高的被优先处理（比如优先级为 -10 的基链比 20 的基链先处理），但是如果某条基链拒绝或者删除了数据包，则后续的基链不再被处理。而常规链不挂在钩子上，是通过从其他链 `jump` 或 `goto` 过来的，一个不被 `jump` 或 `goto` 到的常规链没有什么作用，就像一个不被调用的函数那样永远得不到执行。被 `jump` 到的链需要返回，而 `goto` 不用返回。这里的规则就是防火墙规则，一条规则由表达式和语句组成，如果一个经过的数据包匹配表达式成功，那么就对该数据包执行规则中的语句。规则集、地址族、表、链和规则的关系参见图 2-31。

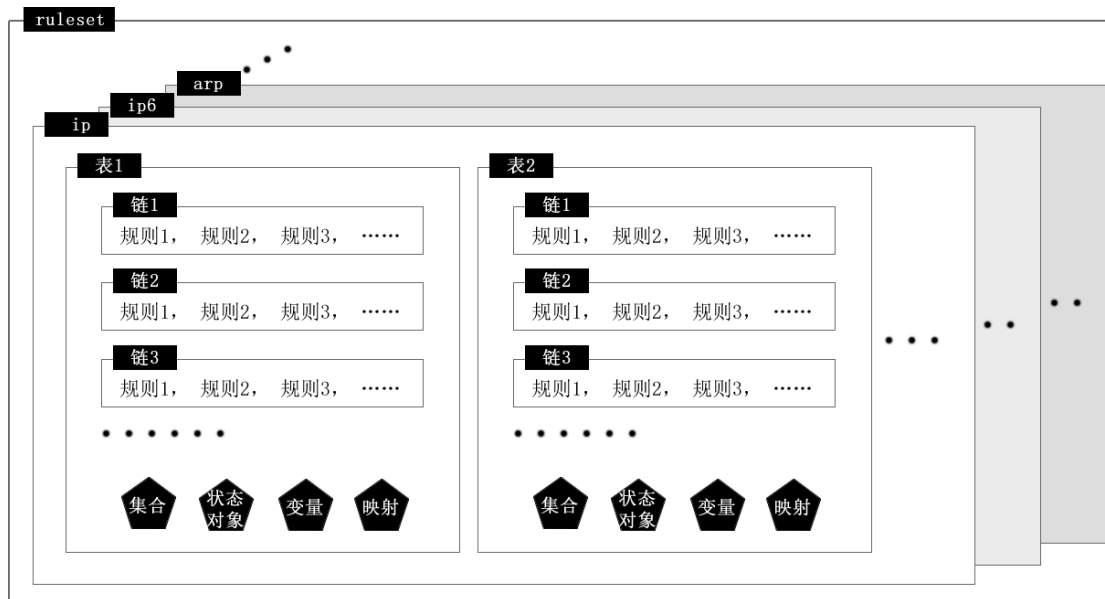


图 2-31 nftables 中的规则集、地址族、表、链和规则的关系

假设 netfilter 子系统的 prerouting 钩子上挂了 3 条基链，分别命名为基链 1、基链 2 和基链 3，优先级依次增大，基链 2 的某条规则 jump 到常规链 10。因此我们可以这样描述数据包被 nftables 处理的情景：一个数据包到达 netfilter 子系统的 prerouting 钩子（参见图 2-30），先由基链 1 处理数据包，基链 1 通过后再由基链 2 处理数据包，基链 2 中的某条规则 jump 到常规链 10，因此常规链 10 处理数据包，然后数据包被常规链 10 中某条规则删除了，因此数据包不会到达基链 3。

表对名字的隔离性给我们带来了极大的方便，应用程序可以单独创建自己的表、链和规则，不用考虑是否会影响其他应用，iptables 是没有这个优势的。

一条规则的语法如下：

```
表达式 [ 表达式 表达式 ... ] 操作语句 [ 操作语句 ... ]
```

即：如果一个“到达”本规则的数据包匹配全部“表达式”，那么该数据包就被执行“操作语句”，可用的语句有：accept（通过）、drop（删除）、jump（调用其他常规链）、goto（跳转到其他常规链）、reject（拒绝）、log（记录日志）、snat（源地址转换）、dnat（目的地址转换）、meta（设置元数据：标记、优先级或跟踪线索）、queue（排队）、dup（复制）、mangle（修改包的头部）、counter（计数）、notrack（不做连接跟踪）等（详见 https://wiki.nftables.org/wiki-nftables/index.php/Main_Page）。

用户可以使用这些命令编写规则：nft、firewall-cmd、firewall-offline-cmd 和 firewall-config 等，其中 nft 命令比较复杂，firewall-config 是图形界面。

1. firewall-cmd 命令

firewall-cmd 命令直接与防火墙守护进程打交道，可以实时改变运行时的防火墙策略，也可以改变永久生效的防火墙策略。firewalld 引入了区域（zone）的概念——可以理解为应用场景，区域中包含若干条防火墙策略。通过应用不同的区域来达到快速切换防火墙策略的目的。firewall-cmd 命令的参数支持 Tab 键补齐，比如我输入“firewall-cmd -”，然后连续按两次 Tab 键，会显示全部的参数，再比如输入“firewall-cmd --add-”后连续按两次 Tab 键，会显示所有以--add-开头的参数。在线文档 man firewall-cmd 提供了详细的说明，下面举例说明，以抛砖引玉。

（1）查看防火墙是否正在运行：

```
[root@localhost ~]# firewall-cmd --state
not running
```

表明防火墙没有运行，采用下面的命令激活并启动防火墙。

```
[root@localhost ~]# systemctl --now enable firewalld
Created symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service → /usr/lib/systemd/system/firewalld.service.
Created symlink /etc/systemd/system/multi-user.target.wants/firewalld.service → /usr/lib/systemd/system/firewalld.service.
```

（2）查看预设的区域有哪些：

```
[root@localhost ~]# firewall-cmd --get-zones
block dmz drop external home internal nm-shared public trusted work
```

预设的区域有 block、dmz、drop、external、home、internal、nm-shared、public、trusted 和 work，可以根据实际的应用场景选择合适的预设区域，一块网卡每次只能绑定一个区域。这些预设区域的规则参见表 2-4。

表 2-4 预设区域的规则

序号	区域（noze）	默 认 策 略 规 则
1	trusted	允许所有的数据包进出（不设防）
2	home, internal	拒绝进入的流量，除非与出去的流量相关，但允许与 ssh、mdns、ipp-client、amba-client 与 dhcpv6-client 服务相关的数据包进入
3	work	拒绝进入的流量，除非与出去的流量相关，但允许与 ssh、ipp-client 与 dhcpv6-client 服务相关的数据包进入
4	public	拒绝进入的流量，除非与出去的流量相关，但允许与 ssh、dhcpv6-client 服务相关的数据包进入

5	external, dmz	拒绝进入的流量，除非与出去的流量相关；而如果流量与 <code>ssh</code> 服务相关，则允许进入
6	block, drop	拒绝进入的流量，除非与出去的流量相关

可以修改预设区域内的防火墙规则以便更好契合实际的应用场景，当然也可以新增区域。

(3) 查看当前正在使用的区域（称为活动区）：

```
[root@localhost ~]# firewall-cmd --get-active-zones
public
  interfaces: ens160

[root@localhost ~]# firewall-cmd --get-default-zone
public
```

表明区域 `public` 被绑定到网卡 `ens160` 上，即当前正在使用 `public` 区。

(4) 查看活动区内的防火墙规则：

```
[root@localhost ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

(5) 查看 `dmz` 区内的规则：

```
[root@localhost ~]# firewall-cmd --info-zone=dmz
dmz
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

即 `dmz` 区目前只有 `ssh` 服务可进，没有连接任何网卡。

(6) 活动区改为 work:

```
[root@localhost ~]# firewall-cmd --set-default-zone=work
success
```

(7) 新建区域 caidi:

```
[root@localhost ~]# firewall-cmd --permanent --new-zone=caidi
success
```

参数 `--permanent` 表示操纵永久生效的防火墙策略，否则就是临时的（重启计算机就没有了），新建的区的规则是空的——会话只能出不能进。

(8) 使用 caidi 区域（变为活动区）:

```
[root@localhost ~]# firewall-cmd --reload #加载永久性的防火墙策略
[root@localhost ~]# firewall-cmd --set-default-zone=caidi #使用 caidi 区域
```

可以验证，此时不能从外面登录到本机了。

(9) 在 caidi 区域中加入 ssh 服务，这样外面就能登录了:

```
[root@localhost ~]# firewall-cmd --zone=caidi --add-service ssh
```

如果想永久生效，就加上 `--permanent` 参数，如果去掉参数 `--zone=caidi`，就是操纵当前正在使用的区域。

(10) 删除区域 mySecure:

```
[root@localhost ~]# firewall-cmd --permanent --delete-zone=mySecure
```

注意：诸如 home、work 等这种预设的区不能删除。

(11) 使用 dmz 区域:

```
[root@localhost ~]# firewall-cmd --zone=work --remove-interface=ens160
[root@localhost ~]# firewall-cmd --zone=dmz --add-interface=ens160
[root@localhost ~]# firewall-cmd --set-default-zone=dmz
```

(12) 查看预设了防火墙策略的服务:

```
[root@localhost ~]# firewall-cmd --get-services
```

针对常见的应用服务（如 dhcp, dns, ftp, http, https, postgresql, pop3, nfs, mysql, rdp, samba, smtp, ssh, tftp, 等等），firewalld 预设了策略，我们直接在区域中加入这样的服务即可开通从外面进来的数据包，看下面的例子。

（13）在 8 小时内允许外面访问本机的网站：

```
[root@localhost ~]# firewall-cmd --add-service=http --timeout=8h
```

注意：--permanent 和--timeout 参数不能同时使用。

（14）不允许从外面登录本机：

```
[root@localhost ~]# firewall-cmd --remove-service=ssh
```

（15）打通 443 端口，即允许从外面访问本机的 https 网站：

```
[root@localhost ~]# firewall-cmd --add-port=443/tcp
```

（16）在 work 区域中，开通从 5901 至 5910 的 udp 端口进入本机：

```
[root@localhost ~]# firewall-cmd --permanent --zone=work --add-port=5901-5910/udp
```

（17）drop 区域内允许 icmp 协议流量进入，即允许 ping：

```
[root@localhost ~]# firewall-cmd --zone=drop --add-protocol=icmp
```

（18）work 区域允许来自 192.168.192.21 机器的任何会话：

```
[root@localhost ~]# firewall-cmd --zone=work --add-source=192.168.192.21
```

如果省略 --zone 参数，就修改默认区域。还可以定义网段，如 --add-source=10.10.2.0/24 表示接受来自 10.10.2.0/24 网络的任何链接。

注意：--add-*、--remove-*和--list-*分别表示增加、删除和显示相关策略的参数，如--add-port、--remove-port 和--list-ports。

（19）将访问本机 888 端口的流量转发到本机 22 端口：

```
[root@localhost ~]# firewall-cmd --add-forward-port=port=888:proto=tcp:toport=22
```

不过要先开通 888 端口（参考例子 15）。

（20）将访问本机 999 端口的流量转发到 ip 为 192.168.192.21 的主机的 22 端口（需要开启 masquerade）：

```
[root@localhost ~]# firewall-cmd --zone=caidi --add-masquerade
[root@localhost ~]# firewall-cmd --add-port=999/tcp
[root@localhost ~]# firewall-cmd --zone=caidi --add-forward-port=port=999:proto=tcp:toport=22:toaddr=192.168.192.21
```

（21）每分钟允许 10 个新连接访问 ftp 服务：

```
[root@localhost ~]# firewall-cmd --add-rich-rule="rule service name=ftp limit value=10/m accept"
```

（22）伪装，将来自局域网 192.168.2.0/24 网段访问外网的流量映射为网络出口公网 IP，即修改源 IP 地址：

```
[root@localhost ~]# firewall-cmd --zone=drop --add-masquerade
[root@localhost ~]# firewall-cmd --zone=drop --add-rich-rule="rule family=ipv4 source \
address=192.168.2.0/24 masquerade"
```

rich rule 称为富规则，使用富规则我们可以精细化定义防火墙规则，详情请参考在线文档 `man 5 firewalld.richlanguage`。

（23）进入紧急维护模式（中断所有对外的链接，仅本机控制台能登录）：

```
[root@localhost ~]# firewall-cmd --panic-on
```

（24）退出紧急维护模式：

```
[root@localhost ~]# firewall-cmd --panic-off
```

2.9.4 实战案例

首选采用下面的命令安装防火墙：

```
[root@localhost ~]# dnf install firewalld nftables firewall-config -y
```

同时会安装几个依赖包，诸如 `ipset`。`firewall-config` 是图形化的配置工具，可以不安装它。`firewalld` 主配置文件为 `/etc/firewalld/firewalld.conf`，防火墙策略放在 `/usr/lib/firewalld/` 目录中。下面我们以几个应用场景来做实验。

案例 1. 个人工作站

对于 Linux 工作站，防火墙的策略一般是：允许出去，只允许外面 ssh 远程登录进来，能访问本机的网站，能 ping 本机。最简单的方法是采用 `firewall-cmd` 命令切换到 work 区域：

```
[root@localhost ~]# firewall-cmd --set-default-zone=work
[root@localhost ~]# firewall-cmd --remove-interface=ens160
[root@localhost ~]# firewall-cmd --zone=work --add-interface=ens160
```

案例 2. 边界防火墙

如图 2-32 所示的边界防火墙是最流行的布局，我们用一台安装 Linux 和 nftables 的计算机充当防火墙。在边界防火墙上执行 `nft -f ruleset.nft` 即可。

图 2-32 边界防火墙

图 2-32 边界防火墙

1. 首先编辑五个文件：`ruleset.nft`、`defines.nft`、`inet-filter-sets.nft`、`inet-filter-forward.nft` 和 `inet-filter-local.nft`。

文件 `ruleset.nft` 内容：

```
flush ruleset

include "./defines.nft"

table inet filter_tb {
    chain global_ch {
        ct state established,related accept
        ct state invalid drop
        ip protocol icmp accept
        ip6 nexthdr icmpv6 accept
        udp dport 53 accept
    }
}
```

```

include "./inet-filter-sets.nft"
include "./inet-filter-forward.nft"
include "./inet-filter-local.nft"
}

```

文件 `defines.nft` 内容:

```

# interfaces
define nic_inet = ens160
define nic_dmz = bond0
define nic_lan = bond1

# network ranks
define net_ipv4_dmz = 10.0.1.0/24
define net_ipv6_dmz = fe00:1::/64
define net_ipv4_lan = 10.0.2.0/24
define net_ipv6_lan = fe00:2::/64

# some machines
define server1_ipv4 = 10.0.1.2
define server1_ipv6 = fe00:1::2
define workstation1_ipv4 = 10.0.2.2
define workstation1_ipv6 = fe00:2::2

```

文件 `inet-filter-sets.nft` 内容:

```

set myset_ipv4 {
    type ipv4_addr;
    elements = { $server1_ipv4 , $workstation1_ipv4 }
}

set myset_ipv6 {
    type ipv6_addr;
    elements = { $server1_ipv6 , $workstation1_ipv6 }
}

```

文件 `inet-filter-forward.nft` 内容:

```

chain dmz_in {
    # your rules for traffic to your dmz servers
    ip saddr @myset_ipv4
    ip6 saddr @myset_ipv6
}

chain dmz_out {
    # your rules for traffic from the dmz to internet
}

chain lan_in {
    # your rules for traffic to your LAN nodes
}

chain lan_out {
    # your rules for traffic from the LAN to the internet
}

chain forward_ch {
    type filter hook forward priority 0; policy drop;
    jump global_ch
    oifname vmap { $nic_dmz : jump dmz_in , $nic_lan : jump lan_in }
    oifname $nic_inet iifname vmap { $nic_dmz : jump dmz_out , $nic_lan : jump lan_out }
}

```

文件 `inet-filter-local.nft` 内容:

```

chain input_ch {
    type filter hook input priority 0 ; policy drop;
    jump global_ch
    # your rules for traffic to the firewall here
}

chain output_ch {
    type filter hook output priority 0 ; policy drop;
}

```

```
    jump global_ch
    # your rules for traffic originated from the firewall itself here
}
```

2. 执行命令提交:

```
[root@localhost ~]# nft -f ruleset.nft
```

2.10 Docker 容器

由于每台虚拟机里都要安装和运行操作系统，这样就浪费了很多计算资源，为此人们推出了容器技术。举一个简单的例子：假如一台计算机的配置是双核 3GHz 的 CPU、8GB 的内存、500GB 的硬盘，现在在这台计算机上创建 6 台虚拟机，每台虚拟机分配 1GB 内存、64GB 硬盘、1GHz CPU，虚拟机都安装 Windows 7 操作系统。那么当全部虚拟机启动后，几乎就很难再运行应用程序了，因为内存和 CPU 资源都被操作系统本身消耗掉了。为此，有公司专门推出了应用软件容器产品，即在操作系统层上创建一个容器，这些容器共享下层的操作系统内核和硬件资源，但是每个容器可单独限制 CPU、内存、硬盘和网络带宽容量，并且拥有单独的 IP 地址和操作系统管理员账户，可以关闭和重启。与虚拟机最大的不同是，容器里不用再安装操作系统，因此浪费的计算资源也就大大减少了，这样同样一台计算机就可以服务于更多的租户，示意图如图 6-1 所示。

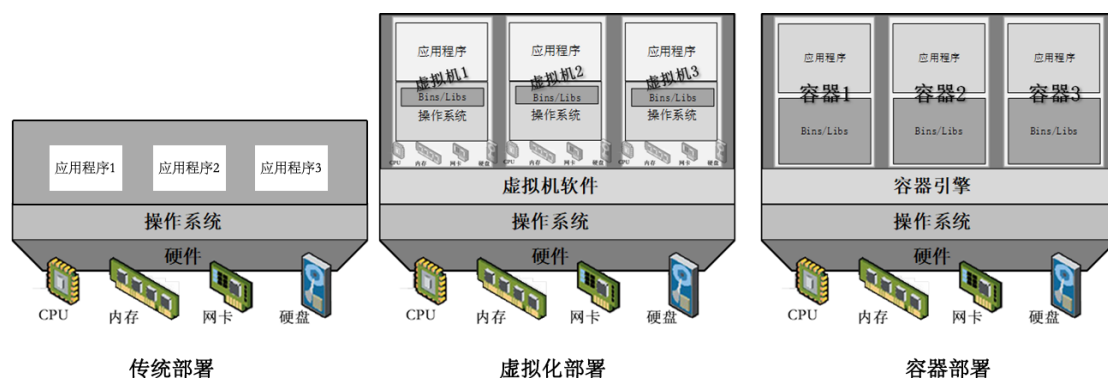


图 3-33 虚拟机和容器的区别

但是容器的隔离效果比虚拟机差，在安全性要求高的场合（比如公有云）中，虚拟机使用率要远远高于容器，而安全性要求不高的场合（比如私有云、公司内部 IT 环境）中，容器的使用率会越来越高。谷歌云目前的做法是在虚拟机里跑容器，并把 `kubernetes` 交给租户使用。开源产品 `kata` 和 `gVisor` 以及微软的闭源产品 `Hyper-V Container` 都是跨界产品——既接近于容器的轻量性，又接近于虚拟机的安全性。`gVisor` 是谷歌主推的开源项目，具备良好的发展势头。

一个容器原则上只运行一至两个应用，比如网站服务、邮件等，但是非常重型的应用不适合跑在容器中。使用容器的目的是方便开发、测试、发布、隔离和在集群中迁移应用程序，使得同一台电脑上跑很多应用程序而互不干扰，有点类似人们常说的绿色软件。所以说容器不是虚拟机的替代品，各有自己应用的领域，虚拟机里可以跑容器，但是反过来就不行。

目前有很多技术可以在 Linux 操作系统上实现容器，比如 `OpenVZ`、`Linux-VServer`、`LXC`、`Docker`、`FreeBSD jail`、`Solaris Containers` 和 `podman` 等，其中 `Docker`

几年前在谷歌公司的大力推动下发展迅速，而且谷歌发布了构建于 Docker 之上的开源的 Kubernetes 管理平台，这个平台使得管理运行在成千上万台电脑上的数十万个 Docker 容器变得异常轻松和简单。而由红帽推出的 podman 是后起之秀，它不用守护进程，也可在非 root 用户下运行，podman 的用法几乎于 docker 完全兼容。

在 docker 中，容器就是镜像（image）的实例化，镜像与容器的关系就像面向对象编程语言中的类与实例化的关系，一个镜像允许实例化多个容器，每个容器具有唯一的 ID 号。镜像和容器的关系参考图 2-34。

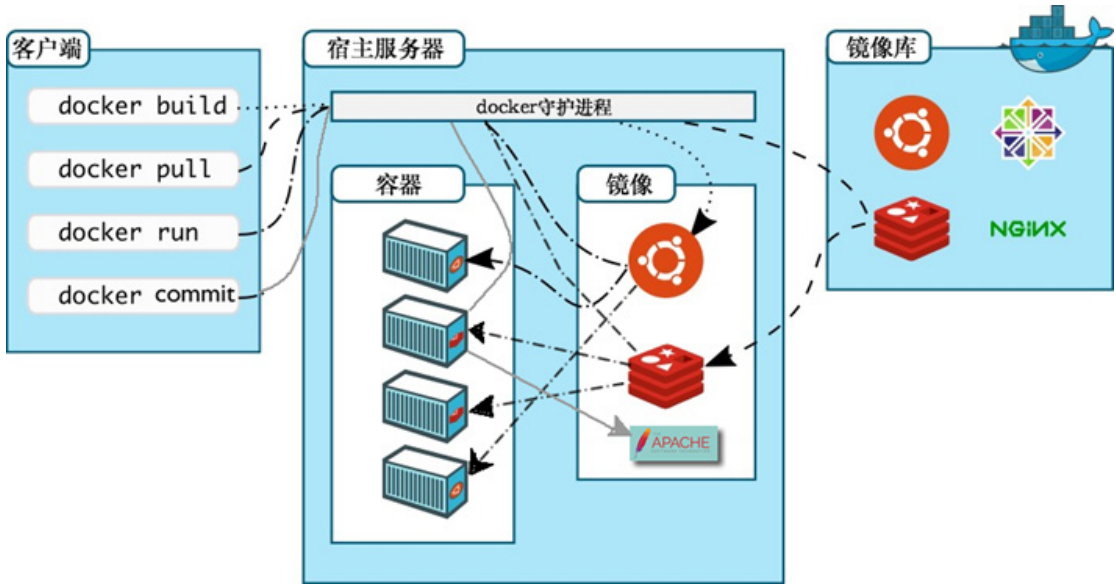


图 2-34 镜像和容器

镜像库一般位于公网上，用户可以把镜像下载到本地（即上图中的宿主服务器）（采用 `docker pull` 客户端命令），或者自己创建镜像（采用 `docker build` 客户端命令），或者从容器生成镜像（采用 `docker commit` 客户端命令）。注意图中的线型区别，比如用户执行命令 `docker pull` 向 `docker` 守护进程发指示，`docker` 守护进程就从网上下载镜像。

容器具有七种状态，状态转换参考图 2-35 所示。

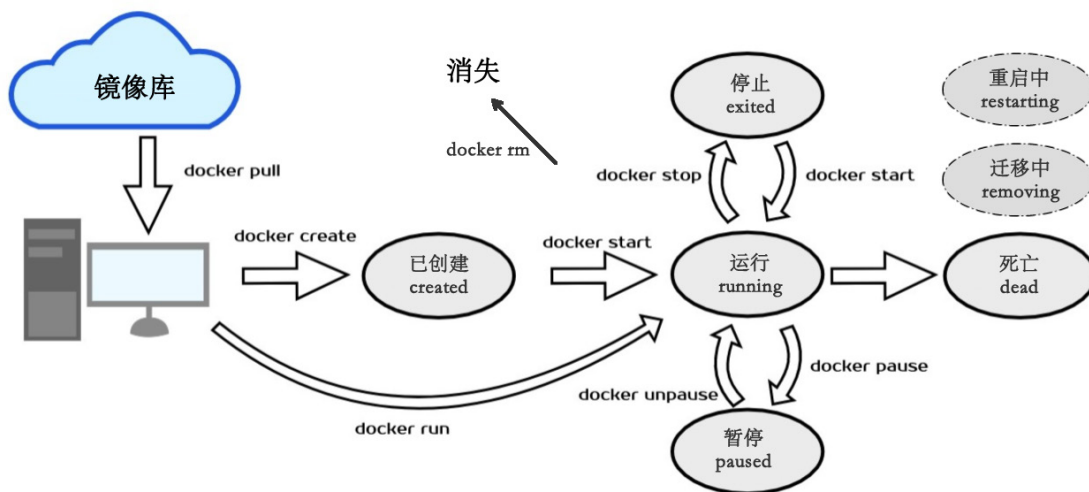


图 2-35 容器的状态

七种状态是已创建（created）、运行（running）、停止（exited）、暂停（paused）、死亡（dead）、重启中（restarting）和迁移中（removing），其中最后两种状态是过度状态。状态变换都是人为驱动的（运行相应的命令 `docker ??`）。参考图 2-36。

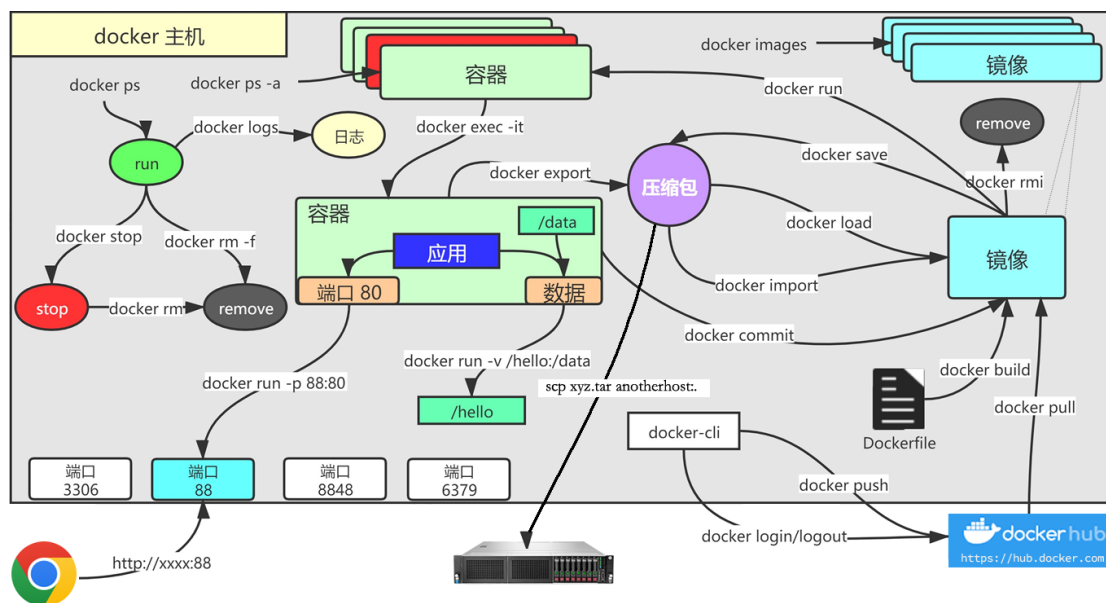


图 2-36 容器操作图景

Kubernetes 和 OpenStack 类似于 windows 及 Linux 操作系统，windows 和 Linux 操作系统管理进程，而 OpenStack 是专门管理虚拟机的，Kubernetes 是专门用来管理容器的。

2.10.1 安装

从 docker 官网下载安装源文件 `docker-ce.repo`，并禁用安装源：

```
[root@localhost ~]# curl https://download.docker.com/linux/centos/docker-ce.repo -o /etc/yum.repos.d/docker-ce.repo

[root@localhost ~]# dnf install docker-ce -y
[root@localhost ~]# systemctl enable --now docker
[root@localhost ~]# docker version
Client: Docker Engine - Community
Version:      20.10.22
API version:  1.41
Go version:   gol.18.7
Git commit:   baeda1f
Built:        Thu Dec 15 22:26:06 2022
OS/Arch:      linux/amd64
Context:      default
Experimental: true
.....
```

更换为国内的镜像库：

```
[root@localhost ~]# vim /etc/docker/daemon.json
{
    "registry-mirrors": [
        "http://hub-mirror.c.163.com",
        "https://docker.mirrors.ustc.edu.cn",
        "https://registry.docker-cn.com"
    ]
}
[root@localhost ~]# systemctl restart docker
[root@localhost ~]# docker info
.....
Experimental: false
Insecure Registries:
  127.0.0.0/8
Registry Mirrors:
  http://hub-mirror.c.163.com/
  https://docker.mirrors.ustc.edu.cn/
  https://registry.docker-cn.com/
Live Restore Enabled: false
```

2.10.2 体验一遍

```
# 搜索包含 debian 关键字的镜像:
[root@localhost ~]# docker search debian | head
NAME                DESCRIPTION                STARS     OFFICIAL   AUTOMATED
ubuntu              Ubuntu is a Debian-based Linux operating sys...  15160     [OK]
debian              Debian is a Linux distribution that's compos...  4478      [OK]
neurodebian          NeuroDebian provides neuroscience research s...  94        [OK]
bitnami/debian-base-buildpack  Debian base compilation image                2          [OK]
dockage/debian-runit  Docker image uses runit as a process supervi...  2          [OK]
treehouses/debian     Debian image that forms the base for some Do...  2          [OK]
dockage/debian        Official Debian Image with USTC Mirror         1
ustclug/debian        Debian arm64 Base Images                       1

# 下载 debian 镜像:
[root@localhost ~]# docker pull debian

# 列出本地的镜像:
[root@localhost ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
debian        latest   d8cacd17cfdc   9 days ago    124MB
```

```

# 查看镜像的详细信息:
[root@localhost ~]# docker inspect debian
[
  {
    "Id": "sha256:d8cacd17cfdcf44e296b41c96f5dab7ae82024c28c4f45bd9181f3823bca639f",
    "RepoTags": [
      "debian:latest"
    ],
    "RepoDigests": [
      "debian@sha256:bfe6615d017d1eebe19f349669de58cda36c668ef916e618be78071513c690e5"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2022-10-25T01:43:42.951329892Z",
    "Container": "580312df4fefc2e21c1da48866fde98215a64957ffa9d9d778f9d6f5426c5f102",
    "ContainerConfig": {
      "Hostname": "580312df4fef",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      ....
    }
  }
]

#运行容器并运行容器中的/bin/bash（进入容器），然后退出容器:
[root@localhost ~]# docker run -it debian /bin/bash
root@c02f2354b88c:/# uname -a
Linux c02f2354b88c 4.18.0-408.el8.x86_64 #1 SMP Mon Jul 18 17:42:52 UTC 2022 x86_64 GNU/Linux
root@c02f2354b88c:/# exit                                     #退出并关闭容器

# 查看全部状态的容器:
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
c02f2354b88c   debian   "/bin/bash"   53 seconds ago   Exited (0) 36 seconds ago           competent_wing

# 再次启动容器 c02f2354b88c:
[root@localhost ~]# docker start c02f2354b88c
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
c02f2354b88c   debian   "/bin/bash"   6 minutes ago   Up 5 seconds           competent_wing

# 进入处于运行状态的容器:
[root@localhost ~]# docker attach c02f2354b88c
root@c02f2354b88c:/# ls /
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr
var
root@c02f2354b88c:/# cat /etc/issue
Debian GNU/Linux 11 \n \l

root@c02f2354b88c:/#                                     #按 Ctrl+p, Ctrl+q 退出容器，但容器继续运行

[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
c02f2354b88c   debian   "/bin/bash"   10 minutes ago   Up 4 minutes           competent_wing

```

一旦采用 `exit` 命令退出，容器就被停止。如果退出后让容器继续运行，请采用 `Ctrl+p`、`Ctrl+q` 离开容器。

2.10.3 定制镜像

接下来我们定制镜像（安装 `nginx` 网站服务器）：

```
[root@localhost ~]# docker run -i -t debian /bin/bash
```

```
#启动并进入 debian 容器
```

```

root@91bac3636a16:/# sed -i 's|deb.debian.org|mirrors.aliyun.com|g' /etc/apt/sources.list
root@91bac3636a16:/# apt update
root@91bac3636a16:/# apt install nginx -y #安装 nginx
root@91bac3636a16:/# apt install iproute2 -y #安装网络工具

root@91bac3636a16:/# ip a | grep inet #查看网卡的 ip
    inet 127.0.0.1/8 scope host lo
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0

root@c02f2354b88c:/# update-rc.d nginx enable #开机时 nginx 自动启动
root@c02f2354b88c:/# exit
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
91bac3636a16   debian    "/bin/bash"             16 minutes ago Up 16 minutes
c02f2354b88c   debian    "/bin/bash"             27 minutes ago Exited (0) 8 seconds ago      competent_wing

[root@localhost ~]# docker commit -m="Nginx Web Server" -a="李木生" c02f2354b88c debian:www
sha256:db7d3fdd104456c8b7a648b86ca11aa5372e26e81c4dc03d22027c8d7aedff7f

[root@localhost ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
debian        www       db7d3fdd1044   22 seconds ago 211MB
debian        latest   d8cacd17cfdc   9 days ago    124MB

```

进入容器后首先更换安装源为阿里云（mirrors.aliyun.com），然后安装 nginx 网站服务器，退出（exit）容器后，容器停止了，然后从此容器生成镜像（上面的 docker commit … 命令），并备注为“Nginx Web Server”，指名作者为“李木生”，新产生的镜像名为 debian:www。

现在可以启动定制的镜像来对外提供 web 服务了。由上面最后一条命令可知，我们定制的容器镜像为 debian:www，id 号为 db7d3fdd1044，下面启动容器里的 nginx 服务。

```

[root@localhost ~]# docker run debian:www /usr/bin/whereis nginx #查看 nginx 的全路径
nginx: /usr/sbin/nginx /usr/lib/nginx /etc/nginx /usr/share/nginx /usr/share/man/man8/nginx.8.gz

# 启动容器中的 Nginx 服务:
[root@localhost ~]# docker run -t -d -p 8081:80 debian:www /usr/sbin/nginx -g "daemon off;"
3d2e6667c7f8c0f456ceec7d262ba8383478e4853651b36734d36ad1f98c0bf

[root@localhost ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
3d2e6667c7f8   debian:www "/usr/sbin/nginx -g ..." 23 seconds ago Up 22 seconds 0.0.0.0:8081->80/tcp, :::8081->80/tcp musing_napier

[root@localhost ~]# docker exec 3d2e6667c7f8 /bin/bash -c 'echo "Hello from Nginx on Docker Container" > /var/www/html/index.html' #产生一个简单的页面文件

[root@localhost ~]# curl localhost:8081 #浏览页面
Hello from Nginx on Docker Container

[root@localhost ~]# docker inspect 3d2e6667c7f8 | grep "IPAddress" #查看容器的 ip
    "IPAddress": "172.17.0.2",
    "IPAddress": "172.17.0.2",

[root@localhost ~]# curl 172.17.0.2
Hello from Nginx on Docker Container

```

启动容器时，我们做了端口映射，即宿主机上的 8081 端口映射到容器中的 80 端口，因此可以用两种方式访问容器中的 web 服务：(1) http://<宿主机 ip>:8081；或者(2) http://<容器的 ip>。

或者直接在浏览器中打开 http://<所主机的 ip>:8081，参考图 2-37 所示。

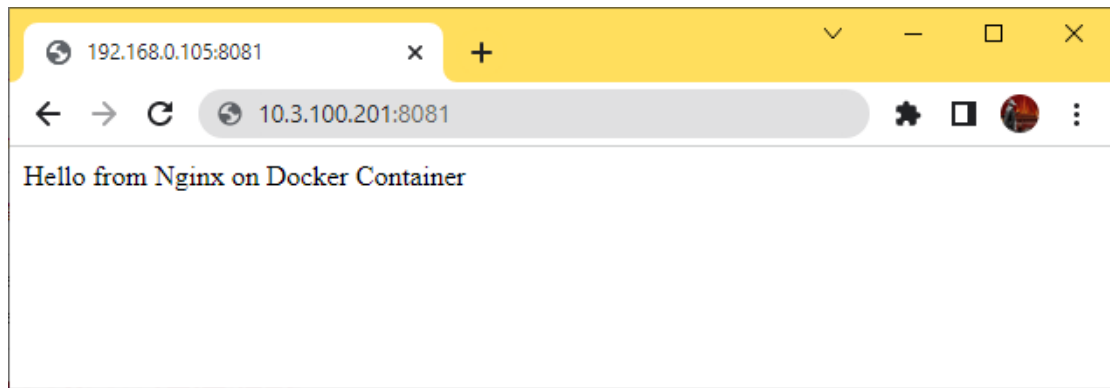


图 2-37 浏览容器里的页面

```
[root@localhost ~]# docker exec -i -t 3d2e6667c7f8 /bin/bash #进入容器
```

备份或发布容器镜像：

```
[root@localhost ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
debian         www       db7d3fdd1044   26 minutes ago 211MB
debian         latest    d8cacd17cfdc   9 days ago    124MB

[root@localhost ~]# docker save db7d3fdd1044 -o ./NginxDocker.tar debian:www #备份
[root@localhost ~]# xz NginxDocker.tar #压缩
[root@localhost ~]# ls -l NginxDocker.tar.xz
-rw----- 1 root root 64932784 Nov  3 15:39 NginxDocker.tar.xz
```

这样就可以把文件 `NginxDocker.tar.xz` 拷贝出来，或者分发给别人了。其他人得到这个文件后就可以恢复成容器镜像，参考下面的步骤。

恢复成容器镜像：

```
[root@localhost ~]# xz -d NginxDocker.tar #解压
[root@localhost ~]# docker load -i NginxDocker.tar #恢复
f84c270ad171: Loading layer [=====] 87.37MB/87.37MB
Loaded image ID: sha256:bc32c2703bac34fbf9ac9d5d0c209a7d3e53344b5587a58914140f2b0d0f3bda

[root@localhost ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
debian         www       bc32c2703bac   2 hours ago    210MB
```

有了容器镜像之后就可以启动容器了：

```
[root@localhost ~]# docker run -t -d -p 8081:80 debian:www /usr/sbin/nginx -g "daemon off;"
```

2.10.4 构建镜像

```
[root@localhost ~]# vim Dockerfile #新建一个文件，输入如下内容
FROM centos
MAINTAINER Jack Wang <wlm@veryopen.org>
```

```

RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-*
RUN dnf makecache
RUN dnf -y install httpd
RUN echo "Dockerfile Test on Aapche httpd" > /var/www/html/index.html

EXPOSE 80
CMD ["-D", "FOREGROUND"]
ENTRYPOINT ["/usr/sbin/httpd"]

```

开始构建，语法是 `docker build -t [image name]:[tag]`。

```

[root@localhost ~]# docker build -t centos-httpd:latest ./
Sending build context to Docker daemon 64.96MB
Step 1/10 : FROM centos
--> 5d0da3dc9764
Step 2/10 : MAINTAINER Jack Wang <wlm@veryopen.org>
--> Using cache
--> 436f41dd7ac7
Step 3/10 : RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
--> Running in 7d154873e57e
Removing intermediate container 7d154873e57e
--> 12d043779f8a
Step 4/10 : RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-*
--> Running in 3f7121017bca
Removing intermediate container 3f7121017bca
--> 729a7eb54425
Step 5/10 : RUN dnf makecache
--> Running in 39bffa7cb071
.....
Step 6/10 : RUN dnf -y install httpd
--> Running in 266477d5f08a
.....
Removing intermediate container 266477d5f08a
--> c989ad5151ea
Step 7/10 : RUN echo "Dockerfile Test on Aapche httpd" > /var/www/html/index.html
--> Running in f7079ec7492a
Removing intermediate container f7079ec7492a
--> da0d57565c24
Step 8/10 : EXPOSE 80
--> Running in 1eb5cfbee348
Removing intermediate container 1eb5cfbee348
--> 511d25733cf9
Step 9/10 : CMD ["-D", "FOREGROUND"]
--> Running in 77efc3a23810
Removing intermediate container 77efc3a23810
--> b1be81ed46e8
Step 10/10 : ENTRYPOINT ["/usr/sbin/httpd"]
--> Running in 655f1b29d22c
Removing intermediate container 655f1b29d22c
--> b76cbdd4e3c5
Successfully built b76cbdd4e3c5
Successfully tagged centos-httpd:latest

[root@localhost ~]# docker images
REPOSITORY      TAG       IMAGE ID       CREATED        SIZE
centos-httpd    latest    b76cbdd4e3c5   6 minutes ago  294MB

# 运行容器
[root@localhost ~]# docker run -d -p 8081:80 centos-httpd
1ae6ab1a393b4cd70d0d30ca7850d28635cd0ebdc5379eb2473095cd898f353b

[root@localhost ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
1ae6ab1a393b   centos-httpd  "/usr/sbin/httpd -D ..." 19 seconds ago Up 16 seconds 0.0.0.0:8081->80/tcp, :::8081->80/tcp fervent_noether

# 浏览页面
[root@localhost ~]# curl localhost:8081
Dockerfile Test on Aapche httpd

```

2.10.5 使用外部存储

容器一旦被删除，里面的数据全部丢失了，所以容器中的重要数据需要保存在外部的存储设备上。

方法 1:

```
[root@localhost ~]# mkdir -p /var/lib/docker/disk01
[root@localhost ~]# echo "永久保存的数据" >> /var/lib/docker/disk01/testfile.txt

# 运行容器，把外部的目录挂载到容器中的/mnt 目录上:
[root@localhost ~]# docker run -it -v /var/lib/docker/disk01:/mnt centos /bin/bash
[root@bb0d351c4204 /]# df -hT
Filesystem                Type      Size  Used Avail Use% Mounted on
overlay                    overlay   28G   3.4G   25G   13% /
tmpfs                      tmpfs     64M    0    64M    0% /dev
tmpfs                      tmpfs     1.4G    0    1.4G    0% /sys/fs/cgroup
shm                        tmpfs     64M    0    64M    0% /dev/shm
/dev/mapper/cs-root        xfs       28G   3.4G   25G   13% /mnt
tmpfs                      tmpfs     1.4G    0    1.4G    0% /proc/asound
tmpfs                      tmpfs     1.4G    0    1.4G    0% /proc/acpi
tmpfs                      tmpfs     1.4G    0    1.4G    0% /proc/scsi
tmpfs                      tmpfs     1.4G    0    1.4G    0% /sys/firmware

[root@bb0d351c4204 /]# cat /mnt/testfile.txt
永久保存的数据
[root@bb0d351c4204 /]# exit
```

方法 2:

```
# 创建卷 volume01:
[root@localhost ~]# docker volume create volume01
[root@localhost ~]# docker volume ls
DRIVER      VOLUME NAME
local       volume01

# 查看卷的详细信息:
[root@localhost ~]# docker volume inspect volume01
[
  {
    "CreatedAt": "2022-11-04T08:39:20+08:00",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/volume01/_data",
    "Name": "volume01",
    "Options": {},
    "Scope": "local"
  }
]
```

可见卷也是对象宿主机上的一个目录（如 /var/lib/docker/volumes/volume01/_data）。

```
# 运行容器，并把卷挂载到/mnt 目录上:
[root@localhost ~]# docker run -it -v volume01:/mnt centos /bin/bash
[root@7f745a9c5483 /]# df -hT /mnt
Filesystem                Type      Size  Used Avail Use% Mounted on
/dev/mapper/cs-root        xfs       28G   3.4G   25G   13% /mnt

[root@7f745a9c5483 /]# echo "由容器写入的数据资料" > /mnt/testfile.txt
[root@7f745a9c5483 /]# exit
[root@localhost ~]# cat /var/lib/docker/volumes/volume01/_data/testfile.txt
由容器写入的数据资料

# 同一个卷还可以挂载到其他容器中:
[root@localhost ~]# docker run -v volume01:/var/volume01 centos /usr/bin/cat /var/volume01/testfile.txt
由容器写入的数据资料

# 删除卷:
[root@localhost ~]# docker volume rm volume01
Error response from daemon: remove volume01: volume is in use -
[d97446251ba6ca6adeecbed6cb4d7e94a77cbc83bf2ee6a15ba5b1e7d918d6b,
35f196402ba86aa873cabda59ada3d80c98e022db2f69b70365be43b5e43299c]
```

```
# 被删除的卷正被容器使用，所以不能删，得先从容器中卸载卷或者干脆直接删除容器：
[root@localhost ~]# docker rm 35f196402ba86aa873cabda59ada3d80c98e022db2f69b70365be43b5e43299c
[root@localhost ~]# docker rm d97446251ba6ca6adeecbed6cb4d7e94a77cbc83bfb2ee6a15ba5b1e7d918d6b
[root@localhost ~]# docker volume rm volume01
```

2.10.6 容器组合

一个完整的应用可能涉及多个服务，比如企业的门户网站包括 **Web** 服务和数据库服务，前面已经讲过，一个容器一般只跑一个服务，这样就要组合两个容器来实现企业门户网站了。采用 **docker-compose** 工具很容易实现多个容器组合实现一个应用。

```
# 安装 docker-compose:
[root@localhost ~]# curl -L https://github.com/docker/compose/releases/download/v2.12.2/docker-
compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
[root@node1 ~]# chmod 755 /usr/local/bin/docker-compose
[root@node1 ~]# docker-compose --version
Docker Compose version v2.12.2

# 构建 Web 服务容器的配置文件:
[root@localhost ~]# vim Dockerfile
FROM debian
MAINTAINER 王先生 <admin@veryopen.org>

RUN sed -i 's|deb.debian.org|mirrors.aliyun.com|g' /etc/apt/sources.list
RUN apt update
RUN apt install nginx -y
RUN rm -rf /var/www/html/*

EXPOSE 80
CMD ["/usr/sbin/nginx", "-g", "daemon off;"]

# 定义应用:
[root@localhost ~]# vim docker-compose.yml
version: '3.9'
services:
  db_service:                                     #服务名称
    image: mariadb                                #容器镜像
    volumes:                                       #外部存储
      - /var/lib/docker/disk01:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: RootPass
      MYSQL_USER: john
      MYSQL_PASSWORD: JohnPass
      MYSQL_DATABASE: student_db
    ports:                                         #内外端口映射
      - "3306:3306"
  web_service:                                     #服务名称
    build: ./                                     #从本地文件 Dockerfile 构建容器
    ports:
      - "8081:80"
    volumes:
      - /var/lib/docker/disk02:/var/www/html
```

注意：

1. **docker-compose.yml** 文件包含四节：**version**、**services**、**networks** 和 **volumes**，分别定义版本号、服务、网络和存储卷，其中 **networks** 和 **volumes** 允许省略，整体结构如下：

```
version: "3.9"
services:
```

```

service1:
    配置参数
service2:
    配置参数
    .....

networks:
    frontend:
    backend:

volumes:
    db-data:

```

2. docker 的版本与 docker-compose.yml 文件中的 version 参数要一致，否则执行下面的构建命令时会报错。

版本对应关系是：docker-compose.yml 版本（docker 版本）→ 3.9(20.0+)、3.8(19.03.0+)、3.7(18.06.0+)、3.6(18.02.0+)、3.5(17.12.0+)、3.4(17.09.0+)、3.3(17.06.0+)、3.2(17.04.0+)、3.1(1.13.1+)、3.0(1.13.0+)。

```

# 构建容器并运行：
[root@localhost ~]# docker-compose up -d
[+] Building 83.4s (8/8) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 264B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/debian:latest 0.0s
=> CACHED [1/4] FROM docker.io/library/debian 0.0s
=> [2/4] RUN sed -i 's|deb.debian.org|mirrors.aliyun.com|g' /etc/apt/sources.list 0.5s
=> [3/4] RUN apt update 25.9s
=> [4/4] RUN apt install nginx -y 56.6s
=> exporting to image 0.3s
=> => exporting layers 0.3s
=> => writing image sha256:10f99946b42b981aa51cc6cd5c9ff9581e549460b3b9dd9487720ae55ceb986e
0.0s
=> => naming to docker.io/library/tmp-WEB_service 0.0s
[+] Running 2/2
:: Container tmp-web_service-1 Started 0.7s
:: Container tmp-db_service-1 Started

[root@localhost ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
aa25802470a0  mariadb   "docker-entrypoint.s..." 11 minutes ago Up 11 minutes 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp tmp-db_service-1
3f48c491d445  tmp-web_service  "/usr/sbin/nginx -g ..." 11 minutes ago Up 11 minutes 0.0.0.0:8080->80/tcp, :::8080->80/tcp tmp-web_service-1

# 验证：
[root@localhost ~]# dnf install mysql
[root@localhost ~]# mysql -h 127.0.0.1 -u root -p -e "show variables like 'hostname';"
Enter password: <输入密码 RootPass, 已经在 docker-compose.yml 中定义>
+-----+
| Variable_name | Value                |
+-----+
| hostname      | aa25802470a0        |
+-----+

[root@localhost ~]# mysql -h 127.0.0.1 -u john -p -e "show databases;"
Enter password:<输入密码 JohnPass>
+-----+
| Database      |
+-----+
| information_schema |
| student_db    |
+-----+

[root@localhost ~]# echo "Hello Docker Compose World" > /var/lib/docker/disk02/index.html
[root@localhost ~]# curl 127.0.0.1:8081
Hello Docker Compose World

```

针对容器组合的其他操作：

```
# 查看容器组合：
[root@localhost ~]# docker-compose ps
NAME                COMMAND                                SERVICE    STATUS    PORTS
tmp-DB_service-1    "docker-entrypoint.s..."            DB_service running   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp
tmp-WEB_service-1   "/usr/sbin/nginx -g ..."           WEB_service running    0.0.0.0:8080->80/tcp, :::8080->80/tcp

# 查看容器组合的日志：
[root@localhost ~]# docker-compose logs
Attaching to tmp-DB_service-1, tmp-WEB_service-1
tmp-DB_service-1 | 2022-11-04 02:02:18+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.9.3+maria-ubu2204 started.
tmp-DB_service-1 | 2022-11-04 02:02:18+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
.....
tmp-DB_service-1 | 2022-11-04 2:02:23 0 [Note] mariadbd: ready for connections.
tmp-DB_service-1 | Version: '10.9.3-MariaDB-1:10.9.3+maria-ubu2204' socket: '/run/mysqld/mysqld.sock' port: 3306 mariadb.org binary
distribution

# 在容器中执行命令，容器名称就是在 docker-compose.yml 文件中定义的：
[root@localhost ~]# docker-compose exec DB_service /bin/bash
root@aa25802470a0:/#

# 停止容器组合：
[root@localhost ~]# docker-compose stop
[+] Running 2/2
  :: Container tmp-WEB_service-1   Stopped      0.3s
  :: Container tmp-DB_service-1    Stopped

# 启动应用中的一个服务，顺带启动依赖服务：
[root@localhost ~]# docker-compose up -d WEB_service
[+] Running 1/1
  :: Container tmp-WEB_service-1   Started

[root@localhost ~]# docker-compose ps
NAME                COMMAND                                SERVICE    STATUS    PORTS
tmp-DB_service-1    "docker-entrypoint.s..."            DB_service exited (0)
tmp-WEB_service-1   "/usr/sbin/nginx -g ..."           WEB_service running    0.0.0.0:8080->80/tcp, :::8080->80/tcp

# 删除容器组合，如果还有容器在运行，就不能删除：
[root@localhost ~]# docker-compose stop
[root@localhost ~]# docker-compose rm
Going to remove tmp-DB_service-1
Are you sure? [yN] y
Removing root_db_1 ... done
```

2.10.7 容器网络

存在四种网络类型，分别是：

1. **host(主机)模式**：容器直接使用宿主机的 ip 与外界通信，所以服务端口不能与宿主机的服务端口冲突。运行容器时通过参数 `--network=host` 指定主机模式。参考图 2-38。

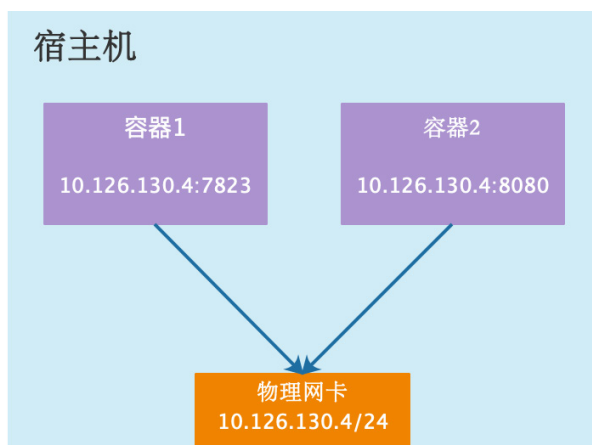


图 2-38 主机模式的网卡

2. **container(容器)模式**: 和另一个容器共享 ip 和地址范围, 运行容器时通过参数 `--network=container:NAME_or_ID` 指定容器模式。示意图参考 2-38。

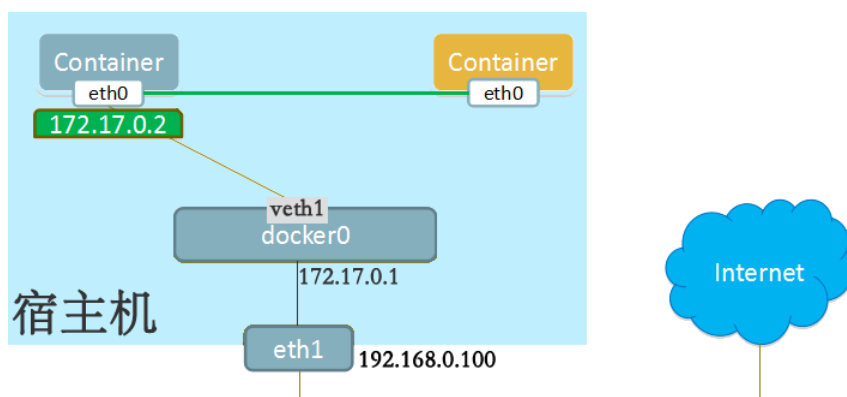


图 2-38 容器模式的网卡

3. **none 模式**: 容器没有网卡、IP、路由等信息, 需要我们自己事后为容器添加网卡、配置 IP 等。运行容器时通过参数 `--network=none` 指定 none 模式。参考图 2-39。

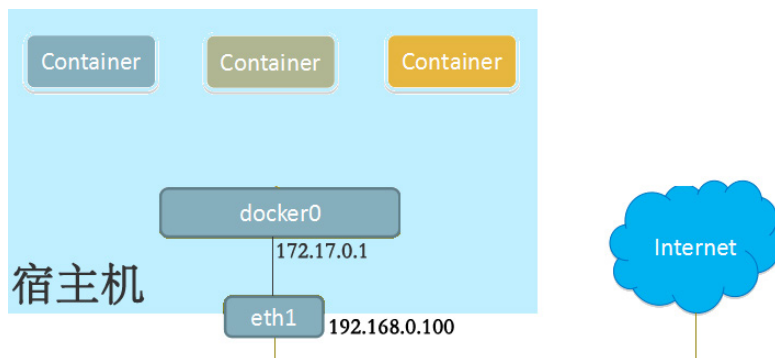


图 2-39 none 模式的网卡

4. **bridge(网桥)模式**: 这是默认的联网方式, 如果要指定参数, 格式为 `--network=bridge`。当 Docker 服务进程启动时, 会在宿主机上创建一个名为 `docker0` 的虚拟网桥并配置一个 `ip` 地址, 容器会连接到这个虚拟网桥上。虚拟网桥类似于物理交换机, 这样宿主机上的所有容器就通过交换机连在了一个二层网络中。从 `docker0` 子网中分配一个 `IP` 给容器使用, 并设置 `docker0` 的 `IP` 地址为容器的默认网关。示意图参考 2-40。

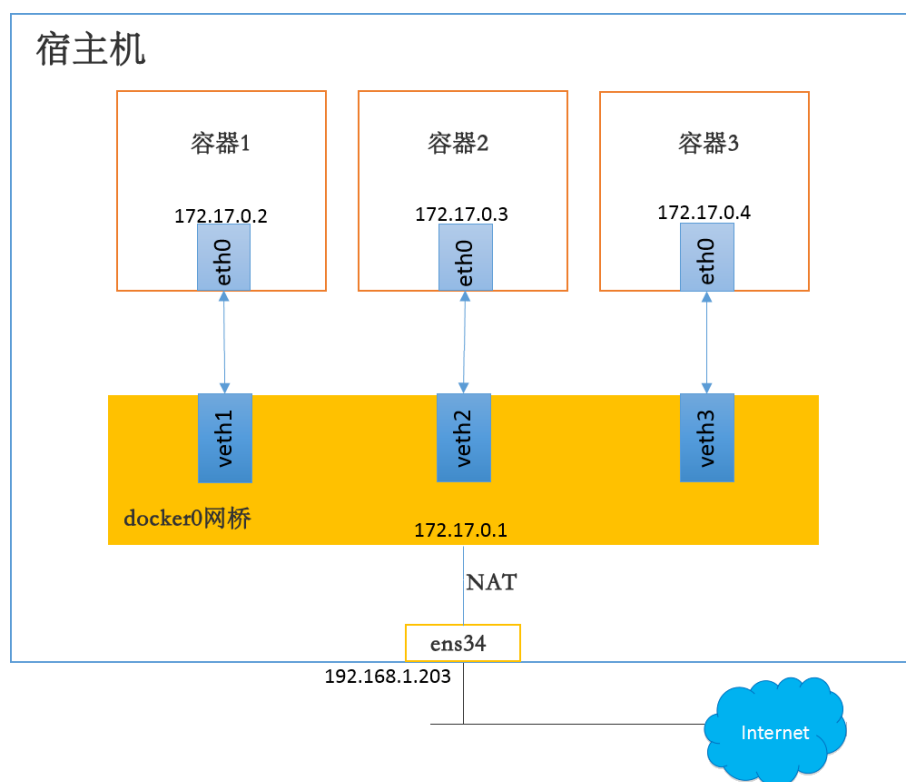


图 2-40 容器之间连网

```
# 显示网络:
[root@localhost ~]# docker network ls
NETWORK ID    NAME        DRIVER    SCOPE
f05052319d8e  bridge     bridge    local
1885c11bd3d4  host       host      local
0e3df7c1b4f6  none       null      local
f7d4310a19c5  tmp_default bridge     local

# 查看网桥的具体信息:
[root@localhost ~]# docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "f05052319d8e27de6cbe651d872ca33a254728c17fec8bee27df4f794d94f259",
    "Created": "2022-11-04T07:52:05.238644633+08:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
```

```

        "Gateway": "172.17.0.1"
    }
}
],
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
    "Network": ""
},
"ConfigOnly": false,
"Containers": {
    "b76dfc2c98003d2b8e3f7312770cd5e3c289605c29af62f403e032955f8a0a51": {
        "Name": "focused_hellman",
        "EndpointID": "53d0b09a734fa1983ef3e82ea5ac1392ab22192414f7c935127fa72583333569",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
    }
},
"Options": {
    "com.docker.network.bridge.default_bridge": "true",
    "com.docker.network.bridge.enable_icc": "true",
    "com.docker.network.bridge.enable_ip_masquerade": "true",
    "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
    "com.docker.network.bridge.name": "docker0",
    "com.docker.network.driver.mtu": "1500"
},
"Labels": {}
}
]

# 创建一个网络:
[root@localhost ~]# docker network create --subnet 192.168.100.0/24 network01

```

默认的类型为网桥，网关为 **192.168.100.1**，可以采用参数 **--driver** 和 **--gateway** 指定类型和网关。

```

# 运行容器并指定采用 network01 网络:
[root@localhost ~]# docker run -i -t --net network01 centos /bin/bash
[root@5e8b968fcfc2 /]# ip a
.....
49: eth0@if50: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a8:64:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.100.2/24 brd 192.168.100.255 scope global eth0
        valid_lft forever preferred_lft forever
[root@5e8b968fcfc2 /]# ip r
default via 192.168.100.1 dev eth0
192.168.100.0/24 dev eth0 proto kernel scope link src 192.168.100.2
[root@5e8b968fcfc2 /]# ping -c 3 www.veryopen.org
PING veryopen.org (34.133.211.54) 56(84) bytes of data:
64 bytes from 54.211.133.34.bc.googleusercontent.com (34.133.211.54): icmp_seq=1 ttl=50 time=204 ms
64 bytes from 54.211.133.34.bc.googleusercontent.com (34.133.211.54): icmp_seq=2 ttl=50 time=202 ms
64 bytes from 54.211.133.34.bc.googleusercontent.com (34.133.211.54): icmp_seq=3 ttl=50 time=205 ms

--- veryopen.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 9339ms
rtt min/avg/max/mdev = 201.922/203.588/204.994/1.370 ms
[root@5e8b968fcfc2 /]#

# 给正在运行的容器附加另一个网络上的 ip:
[root@localhost ~]# docker network connect --ip 192.168.100.10 network01 d84f22149b12
[root@localhost ~]# docker exec d84f22149b12 ip route
default via 172.17.0.1 dev eth0
172.17.0.0/16 dev eth0 proto kernel scope link src 172.17.0.2
192.168.100.0/24 dev eth1 proto kernel scope link src 192.168.100.10

# 从容器上断开网络:
[root@localhost ~]# docker network disconnect network01 d84f22149b12

# 删除网络:
[root@localhost ~]# docker network rm network01
# 删除全部不在使用的网络:
[root@localhost ~]# docker network prune
WARNING! This will remove all custom networks not used by at least one container.

```

```

Are you sure you want to continue? [y/N] y
Deleted Networks:
root_default

# 以 host 网络模式运行容器:
[root@localhost ~]# docker run -d --network=host centos-httpd
39989a03b9ec3ec65cb4677cda223e72abab9ea5a873c92bb215717354d5460b
[root@localhost ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
39989a03b9ec   centos-httpd   "/usr/sbin/httpd -D ..." 8 seconds ago  Up 6 seconds  80             upbeat_elbakyan

[root@localhost ~]# ss -tnlp
State      Recv-Q      Send-Q          Local Address:Port      Peer Address:Port      Process
LISTEN     0            128             0.0.0.0:22              0.0.0.0:*               users:((("sshd",pid=1014,fd=5))
LISTEN     0            128             [::]:22                 [::]:*                  users:((("sshd",pid=1014,fd=7))
LISTEN     0            128             *:80                    *:80                     *:*
users:((("httpd",pid=8872,fd=4)...

[root@localhost ~]# curl localhost
Dockerfile Test on Aapche httpd

```

2.11 虚拟机

各种 Linux 的最新发行版都融入了云计算的功能，云计算中最主要的技术就是虚拟机了，开源虚拟机 KVM 已经集成到 Linux 内核中，Windows Server 2008 及后续版本把虚拟机作为一种服务器角色（Hyper-V），可以随时安装和卸载，非常方便。针对虚拟机浪费资源（CPU、内存、存储等）较大的缺陷，谷歌公司力推 Docker 容器和管理平台 Kubernetes。

虚拟机顾名思义就是通过软件把一台计算机虚拟出很多台计算机（虚拟出来的计算机称为虚拟机），每台虚拟机里要单独安装操作系统以及各种应用软件，参见图 2-40。

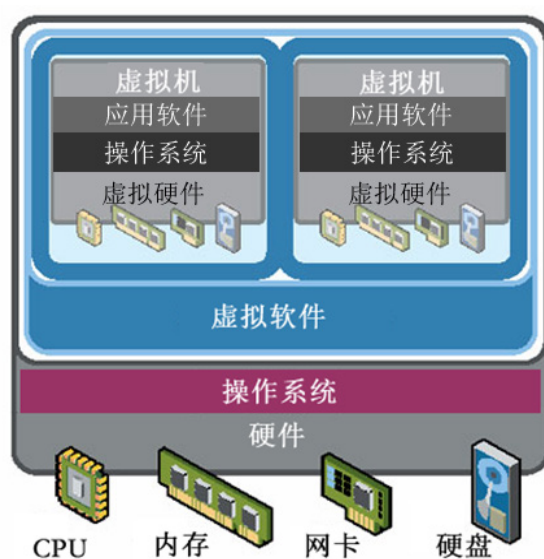


图 2-40 虚拟机示意图

虚拟机里安装的操作系统与物理机里安装的操作系统可以不同，也可以相同。接下来我们在 CentOS 9 上创建多个虚拟机，然后在一些虚拟机里安装 Windows 10 操作系统，在另一些虚拟机里安装 ubuntu 18.04。

首先安装虚拟机软件集：

```
[root@localhost ~]# install '@Virtualization Host'
```

这样一共安装了 qemu-kvm、libvirt-libs 等 73 个软件包。CentOS 9 集成了一个图形化管理页面，先安装它：

```
[root@localhost ~]# dnf install cockpit-* -y
[root@localhost ~]# systemctl --now enable cockpit.socket
```

然后用浏览器打开 <https://192.168.192.20:9090>，输入 root 和它密码即可登录。

如图 2-41 所示。

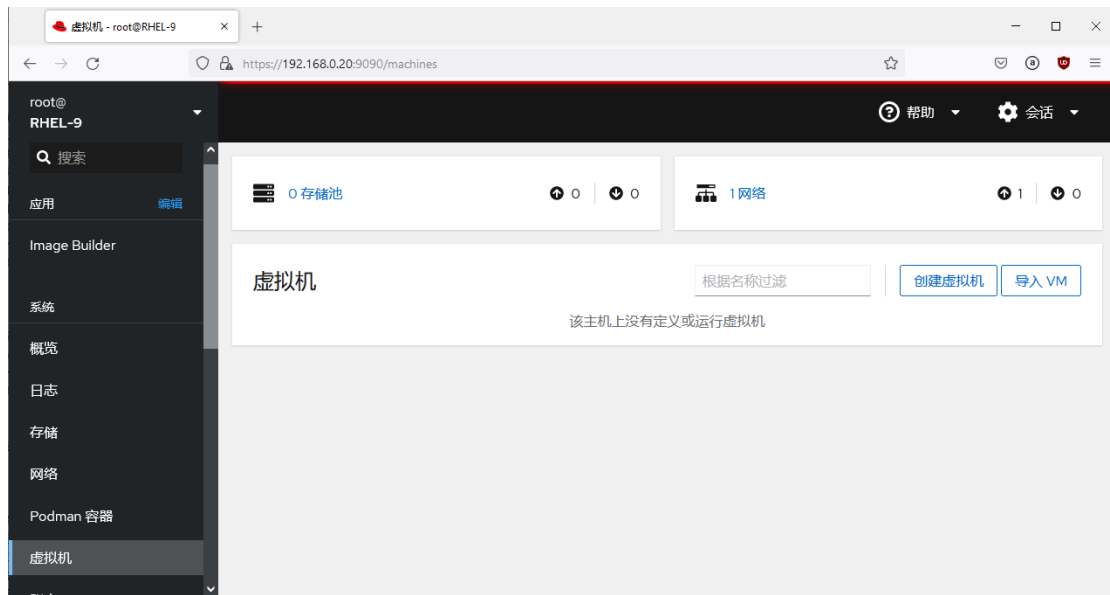


图 2-41 CentOS 9 的管理页面

单击“创建虚拟机”按钮后弹出如图 2-42 所示的画面。

创建新的虚拟机

名称

windows_10

安装类型

本地安装媒体 (ISO镜像或者发行版安装树)

安装源

/tmp/cn_windows_10_editions_version_1909_x64_dvd.iso

操作系统

Microsoft Windows Server 2019 (10.0)

存储

创建新卷

大小

32 GiB

内存

2 GiB

立即启动 VM

☒

创建

取消

图 2-42 新建虚拟机

操作系统的安装介质可以有三个来源，分别是光盘（“**sr0**”按钮）、临时从网上下载（“下载操作系统”按钮）和本地硬盘的镜像文件（“选择文件”按钮）。“输入 URL”按钮用于连接

远 程 桌 面 。 Windows 10 的 安 装 镜 像 文 件
cn_windows_10_editions_version_1909_x64_dvd.iso 需要先准备好, 放在/tmp
目录下。

企业级虚拟化平台 oVirt 适合部署在少于 50 台服务器的生产环境中。