Day08课堂笔记

课程环境切换命令：第二本书是rh134，切换到134的环境去做第二本教材的练习

rht-clearcourse 0

rht-setcourse rh134

rht-vmctl start classroom

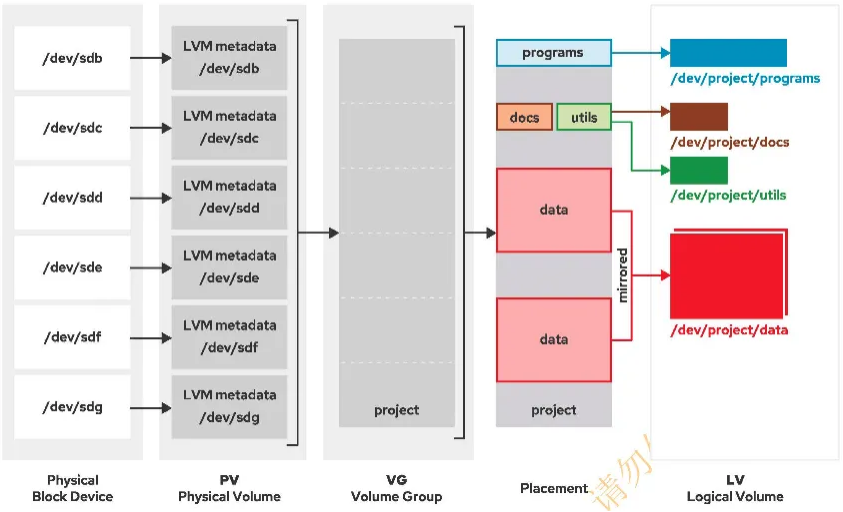
rht-vmctl start all

问题1: ssh登录servera或者serverb失败

原因：servera没有正常启动，打开虚拟机管理平台virt-manager，双击servera，看有没有登陆界面，机器性能不佳可能虚拟机需要花点时间。等。。

1. 管理逻辑卷LVM

1.1. 概述



逻辑卷的特点：

1. 逻辑卷可以动态扩展或缩小存储空间，
2. 还可以把逻辑区块迁移到新的磁盘
3. 逻辑卷四个组成部分
    a. 物理设备：一块硬盘、分区、RAID
    b. 物理卷PV：逻辑卷可以识别的底层格式
    c. 卷组VG：逻辑卷的存储池，一个PV只能属于一个VG，VG可以包含多个PV
    d. 逻辑卷LV：逻辑卷最后的存储设备，可以挂载给目录使用

1.2. 常见命令

Bash

```
创建逻辑卷：可以使用之前的分区奇数parted、fdisk，注意分区的磁盘格式是Linux LVM（0x8e），也可以使用完整磁盘或者阵列卡

#创建物理卷：
pvcreate /dev/vdb1  /dev/vdb2
#创建卷组：
vgcreate vg01 /dev/vdb1  /dev/vdb2
-s 16 PE大小，默认是4M
#创建逻辑卷：
lvcreate –n lv01 –L 700M vg01
-L 直接指定大小
lvcreate –n lv01 –l 70 vg01
-l 指定PE的个数
#格式化文件系统：
mkfs –t xfs /dev/vg01/lv01
#挂载：
/etc/fatab，按名称挂载逻辑卷等同于按UUID
#删除逻辑卷：
umount /mnt/data–lvremove–vgremove–pvremove
#查看状态：
pvdisplay
vgdisplay
lvdisplay

扩展卷组：增加额外的物理卷PV来增加更多的磁盘空间
缩减卷组：未使用的物理卷PV可以从卷组中删除，首先需要用pvmove将数据从一块物理卷组上迁移到另一块物理卷组上
#扩展PV：
pvcreat /dev/vdb3
#扩展VG：
vgextend vg01 /dev/vdb3
#缩减VG：
pvmove /dev/vdb3
vgreduce vg01 /dev/vdb3


#扩展LV：
lvextend –L +300M /dev/vg01/lv01
#扩展文件系统xfs格式：
xfs_growfs /mnt/data（在扩展LV之后一定要记得扩展格式文件系统）
#扩展文件系统ext4格式：
resize2fs /dev/vg01/lv01（在扩展LV之后一定要记得扩展格式文件系统）
#扩展文件系统swap格式：
mkswap /dev/vg01/lv01（在扩展LV之后一定要记得扩展格式文件系统）

普通存储空间
# 1. 备份数据
tar -czf backup.tar.gz /data

# 2. 卸载文件系统
umount /dev/sda2

# 3. 删除并重建分区（风险！）
fdisk /dev/sda  # 删除sda2，重建更大分区

# 4. 检查并恢复文件系统
fsck -f /dev/sda2
mount /dev/sda2 /data

# 5. 恢复数据
tar -xzf backup.tar.gz -C /data
LVM存储空间
# 在线扩展，无需卸载
lvextend -L +5G /dev/myvg/data_lv
resize2fs /dev/myvg/data_lv  # 对于ext4
# 或 xfs_growfs /data  # 对于xfs

# 整个过程服务无需中断
```

普通存储：

bash 复制

```
# 查看普通分区
$ fdisk -l /dev/sda
Device     Boot    Start       End   Sectors Size Id Type
/dev/sda1  *        2048   2099199   2097152   1G 83 Linux
/dev/sda2        2099200 62935039 60835840  29G 83 Linux
/dev/sda3       62935040 83886079 20951040  10G 83 Linux

# 问题：sda2 空间不足，sda3 空间闲置，但无法直接调整
```

LVM 存储：

bash

```
# 创建灵活的存储池
$ pvcreate /dev/sdb1 /dev/sdc1
$ vgcreate myvg /dev/sdb1 /dev/sdc1
$ lvcreate -L 20G -n data_lv myvg
$ lvcreate -L 10G -n backup_lv myvg

# 空间不足时动态扩展
$ lvextend -L +5G /dev/myvg/data_lv
$ resize2fs /dev/myvg/data_lv  # 扩展文件系统
```

一、核心区别总览

特性	普通存储	LVM
扩展性	困难，需要备份-重建-恢复	在线动态扩展/收缩
灵活性	固定分区大小	逻辑卷大小可调
管理性	每个分区独立管理	统一池化管理
高级功能	基本功能	快照、条带化、镜像等
性能优化	有限	支持条带化等性能优化

普通存储架构

复制

```
物理磁盘 → 分区表 → 文件系统 → 数据
    |
    └── 每个分区独立管理，大小固定
```

LVM 架构

复制

```
物理磁盘 → 物理卷(PV) → 卷组(VG) → 逻辑卷(LV) → 文件系统 → 数据
    |                        |
    └── 统一池化管理          └── 灵活分配，动态调整
```

```
实验：电子档教材-P177
纸质P168

parted /dev/vdb mklabel gpt
parted /dev/vdb mkpart first 1M 258M
parted /dev/vdb mkpart second 258M 514M
parted /dev/vdb set 1 lvm on
parted /dev/vdb set 2 lvm on
parted /dev/vdb print

pvcreate /dev/vdb1 /dev/vdb2
vgcreate servera_group /dev/vdb1 /dev/vdb2
lvcreate -n servera_volume -L 400M servera_group

mkfs.xfs /dev/servera_group/servera_volume

mkdir /data
mount /dev/servera_group/servera_volume /data/

cp -a /etc/*.conf /data/
ls -l /data |wc -l
```

1.3. 补充（缩减逻辑卷）

这是一个完整的 LVM 创建和使用流程。让我为您逐行解释并分析这个脚本的执行结果：一、命令逐行解析

1. 创建 GPT 分区表parted /dev/vdb mklabel gpt在磁盘 /dev/vdb上创建 GPT 分区表清除所有现有分区和数据
2. 创建第一个分区parted /dev/vdb mkpart first 1M 258M分区名：first起始：1MB（跳过初始空间）结束：258MB大小：257MB（258M - 1M）
3. 创建第二个分区parted /dev/vdb mkpart second 258M 514M分区名：second起始：258MB结束：514MB大小：256MB（514M - 258M）
4. 设置 LVM 标志
parted /dev/vdb set 1 lvm on
parted /dev/vdb set 2 lvm on
将两个分区的 LVM 标志设为开启标记为可用于 LVM 物理卷
5. 查看分区信息
parted /dev/vdb print显示创建的分区表信息
6. 创建物理卷pv
pvcreate /dev/vdb1 /dev/vdb2将两个分区初始化为 LVM 物理卷
注意：会擦除分区上的任何现有数据　#然后可以pvdisplay查看具体分区内容
7. 创建卷组
vgcreate servera_group /dev/vdb1 /dev/vdb2创建名为 servera_group的卷组包含两个物理卷 /dev/vdb1和 /dev/vdb2　　#然后可以vgdisplay查看具体分区内容
8. 创建逻辑卷
lvcreate -n servera_volume -L 400M servera_group在 servera_group卷组中创建逻辑卷
名称：servera_volume
大小：400MB
9. 创建文件系统
mkfs.xfs /dev/servera_group/servera_volume在逻辑卷上创建 XFS 文件系统　mkfs：Make FileSystem（创建文件系统）

注意事项：（xfs格式不支持，减操作，ext4格式支持）

1. 不能在线缩减，一定要先umount
2. 确保缩减以后的空间大小还能存储原有的数据
3. 在缩减之前要强制检查文件，保证文件系统处于一致的状态

场景1:

直接缩减逻辑卷100M----》50M，数据不受影响

umount /data

xfs_repair /dev/servera_group/servera_volume

vgextend servera_group /dev/vdb3
把vdb3加入到已有的vg servera-group中
lvextend -L 750M /dev/servera_group/servera_volume
扩展到750M
xfs_growfs data/
刷新挂载点
然后在df -Th查看mount的目录

mkdir /file 前面加一个斜杠就是把它创建到根目录的意思

场景2:

3块盘在vg里面，对硬盘做维护，vdb1里面的数据转移到其他盘，把vdb1从pv移除

10. 挂载使用mkdir /data
mount /dev/servera_group/servera_volume /data/创建挂载点 /data挂载逻辑卷到该目录
或者进入vim /etc/fstab 中添加自动挂载 /dev/server_group/servera_volume /data xfs defaults 0 0 然后df -Th 查看挂载详情
11. 复制文件cp -a /etc/*.conf /data/复制 /etc/目录下所有 .conf配置文件到 /data/，-a选项保留文件属性
12. 统计文件数量ls -l /data |wc -l列出 /data目录内容统计行数（文件数量 + 1）

执行过程详解
第一步：磁盘分区布局
/dev/vdb
├──/dev/vdb1: 257MB (1M-258M)
└──/dev/vdb2: 256MB (258M-514M)
总计: 513MB
第二步：LVM 结构
物理卷: /dev/vdb1 (257MB) + /dev/vdb2 (256MB) = 513MB
卷组: servera_group (513MB)
逻辑卷: servera_volume (400MB)
剩余: 113MB 未分配
第三步：存储使用
逻辑卷: 400MB → XFS 文件系统 → 挂载到 /data
实际文件: 复制 /etc/*.conf 文件
物理卷 (PV) - 最底层
1. 什么是 PV？
物理存储的封装：磁盘或分区经过 pvcreate初始化
LVM 的基础单元：没有 PV 就没有 LVM
可被 VG 管理：PV 必须加入 VG 才能使用
卷组 (VG) - 中间层
1. 什么是 VG？
存储资源池：汇集一个或多个 PV
容量总和：VG 大小 = 所有 PV 大小之和
管理单元：从 VG 中分配空间给 LV
. 什么是 LV？
可用的存储空间：从 VG 中分配的逻辑块
最终使用单元：格式化为文件系统后挂载使用
灵活可变：大小可动态调整

/etc/fstab 文件的作用
/etc/fstab 是 FileSystem TABle 的缩写，中文意思是"文件系统表"。它的主要作用是：
核心功能：
定义系统启动时需要自动挂载的文件系统和存储设备
类比理解：
就像 Windows 的"磁盘管理"，给每个分区分配盘符或像"自动挂载清单"，告诉系统："开机时请自动把这些设备挂载到指定位置

ext4格式，减操作 Bash

```
缩小LVM及删盘笔记－－－缩小LVM要先停相关卷的使用，会影响业务运行

1、查看已mount的盘
df -h
du -sh /data/                    xfs_db -c "check" /dev/servera_group/servera_volume
90K     /data/


2、卸载磁盘
umount /data

3、检查磁盘
e2fsck -f /dev/servera_group/servera_volume

4、缩小磁盘到100G，多缩减一些，以免还有内容在要移除的盘上
resize2fs /dev/servera_group/servera_volume 100M

5、缩小LV到100G
lvresize --size 100M /dev/servera_group/servera_volume

6、查看要缩减的vg name
lvdisplay

7.再次挂载，查看文件是否有变化
 mount /dev/servera_group/servera_volume /data/
 ls -l /data/ | wc -l


------------------场景2的要求------------------
8. 把数据从要移除的盘移动到其他的盘
pvmove /dev/vdb2

9、缩减vg
vgreduce servera_group /dev/vdb2

10、从系统中移除磁盘
pvremove /dev/vdb2
完全删除/dev/vdb2

11.再次挂载，查看文件是否有变化
 mount /dev/servera_group/servera_volume /data/
 ls -l /data/ | wc -l
```

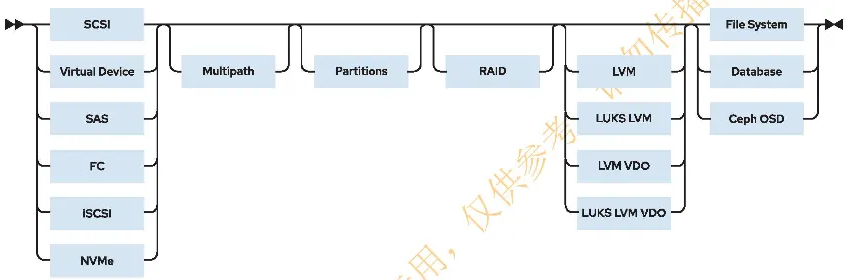
2. 高级存储管理stratis

2.1. 概述

1. Stratis以管理物理存储设备池的服务形式运行
2. Stratis也支持当前在LVM、XFS和设备映射器中使用的所有高级存储功能
3. Stratis是卷管理文件系统方式，在创建文件系统及调整其大小是以动态、透明的方式来管理卷层。
4. Stratis创建的文件系统是以精简配置的方式内置于磁盘设备的共享池中，所有Stratis文件系统没有固定大小，也不预分配未使用的块空间
5. 多个文件系统可以驻留在同一磁盘设备池中，共享可用空间
6. 文件系统也可以保留池空间，以便在需要的时候保证可用性
7. Stratis使用元数据识别所管理的池、卷和文件系统，所以不能对其创建的文件系统进行手动重新格式化或者重新配置，只能使用Stratis工具和命令对其进行管理
8. 每个池中最多可以创建2**24个文件系统
9. 创建快照（文件快照）

优点总结：

1. 不需要指定filesystem大小–精简配置
2. 不需要格式化文件系统–以文件系统服务的方式在运行



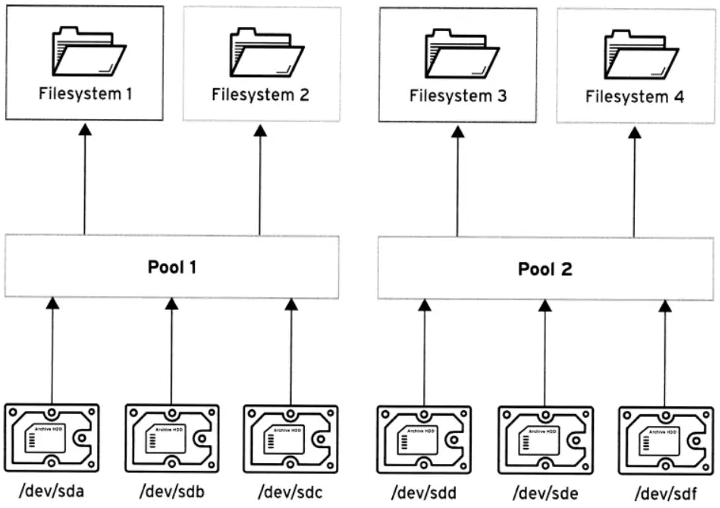


图 8.1: Stratis 的元素

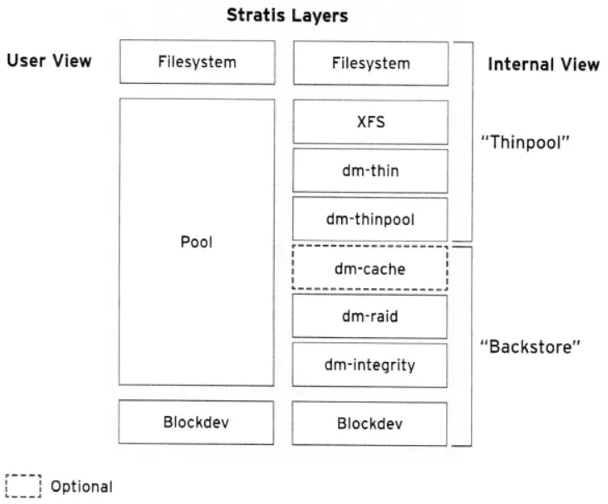


图 8.3: Stratis 层

2.2. 常见管理方法

```
安装Stratis：yum install stratis-cli stratisd
激活：systemctl enable - -now stratisd
创建一个或多个块设备的池：stratis pool create pool1 /dev/vdb
查看可用池：stratis pool list
向池中添加额外块设备：stratis pool add-data pool1 /dev/vdc
查看池中的块设备：stratis blockdev list pool1
在池中创建动态灵活的文件系统：stratis filesystem create pool1 filesystem1
创建文件系统快照：stratis filesystem snapshot pool1 filesystem1 snapshot1
查看可用文件系统列表：stratis filesystem list
开机自动挂载/etc/fstab：UUID=XXX /dir xfs defaults,x-system.requires=stratisd.service(实现延迟挂载，等待stratisd服务开启了再挂载) 0 0


实验：P189
纸质 P180
```

3. 网络存储NFS

3.1. 概述

1.NFS（网络文件系统）使用互联网标准协议实现本地网络服务系统

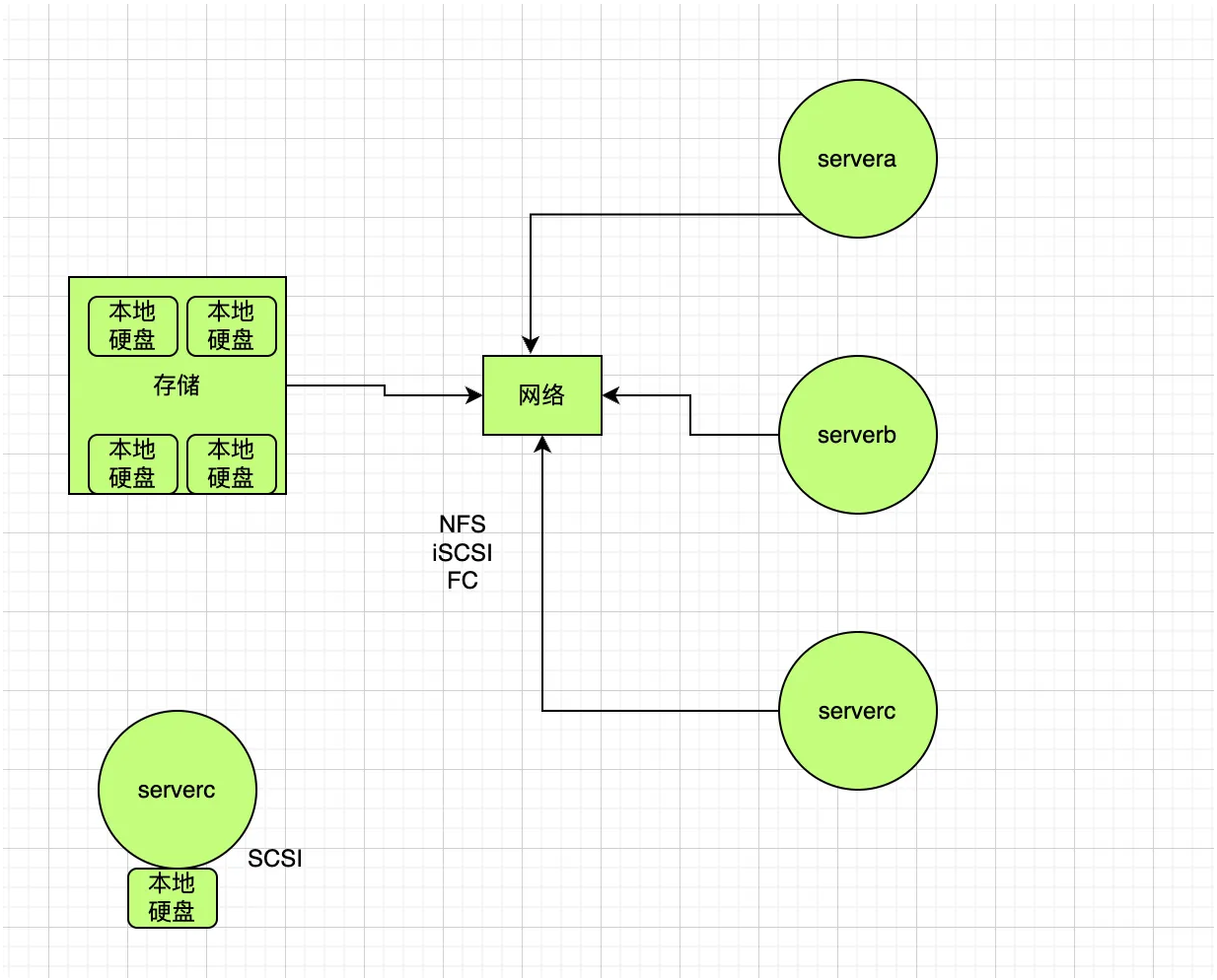
2.Linux8中默认使用NFS4.2版本，NFSv4仅使用TCP与服务器通信

3.NFS服务器导出共享目录，NFS客户端将导出的共享目录挂载到本地挂载点，该本地挂载点必须存在

4.可以使用三种方式挂载NFS共享：mount；/etc/fstab；autofs按需挂载

5.挂载NFS的三个步骤：识别–挂载点–挂载

6.NFS配置工具nfsconf的使用：nfsconf – –set nfsdtcp y



3.2. 挂载的第三种方式autofs+NFS

1.自动挂载器是一种服务autofs，可以根据需要自动挂载NFS共享，并将在不再使用NFS共享时自动卸载这些共享

2.用户无需具有root特权就可以mount和umount命令

3.自动挂载器中配置的NFS共享可以给所有用户使用，受访问权限约束

4.按需连接，不需要时随时释放网络资源

5.自动挂载器在客户端配置，服务器无需任何设置

6.自动挂载器与mount使用相同的选项，包括安全性选项

7.支持直接和间接挂载点映射，提供了挂载点的灵活性

8.可创建和删除间接挂载点，避免了手动管理

9.NFS是系统默认的自动挂载网络的文件系统，也可以自动挂载其他网络文件系统

10.挂载方式分为直接和间接两种

Bash

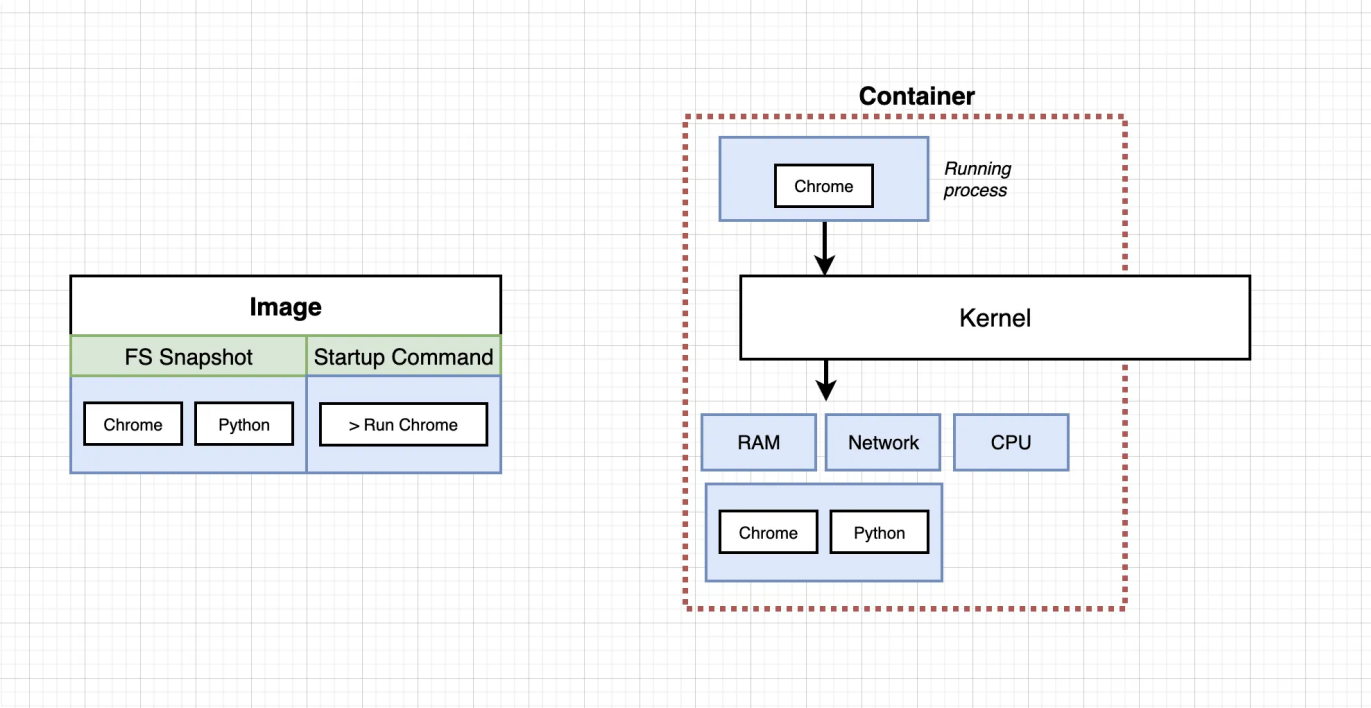
```
实验：电子教材P220
纸质P206



[dbuser1@servera ~]$ cat /etc/auto.master.d/shares.autofs
/remote /etc/auto.shares

[dbuser1@servera ~]$ cat /etc/auto.shares
* -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/student11
```

4. 容器podman（docker）

4.1. 容器的概念

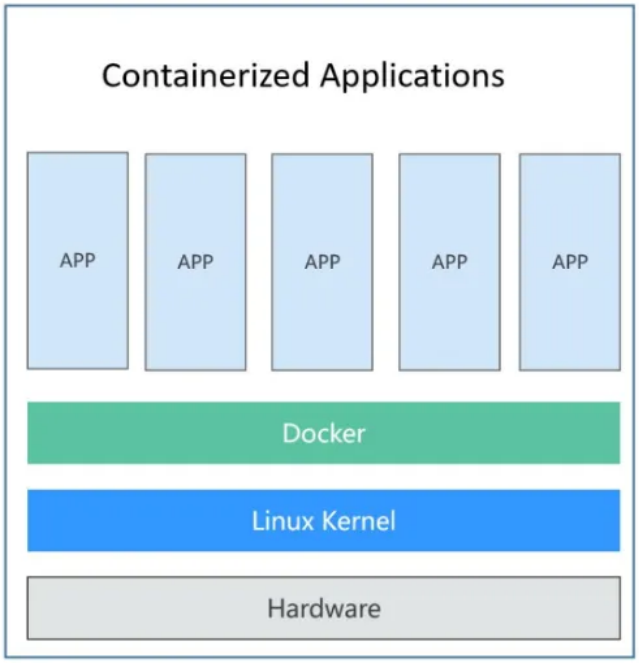
Docker的特点：

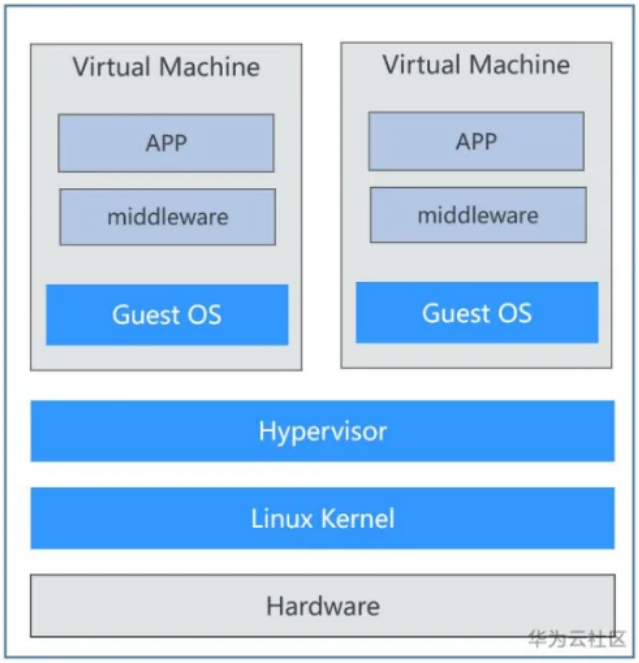
1. 基于linux的内核的应用
2. 应用包含：app+依赖包+文件+os+CMD------》打包成image
3. 容器是运行中的image
4. image是灵活跨平台运行，并且image可以同时运行成多个容器
5. 容器和容器之间，逻辑上相互隔离
6. 根本性解决运维和开发之间的矛盾（内核）



4.2. 容器和虚拟化的区别

> 区别：容器是没有灵魂OS的虚拟机





4.3. Docker的安装

安装链接：

https://download.docker.com

1. 修改docker镜像源为国内清华软件

sed –i 's#download.docker.com#mirrors.tuna.tsinghua.edu.cn/docker–ce#' /etc/yum.repos.d/docker–ce.repo

2. 安装docker–ce

yum install –y docker–ce

3. image也有仓库，
    a. 共有仓库：阿里mirrors.aliyun.com
    b. 私有仓库

[root@localhost ~]# cat /etc/docker/daemon.json

{

"registry–mirrors": ["https://用自己的.mirror.aliyuncs.com"]

}

4. 运行第一个容器

[root@localhost ~]# docker run hello–world

Unable to find image 'hello–world:latest' locally

latest: Pulling from library/hello–world

93288797bd35: Pull complete

Digest: sha256:2498fce14358aa50ead0cc6c19990fc6ff866ce72aeb5546e1d59caac3d0d60f

Status: Downloaded newer image for hello–world:latest

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

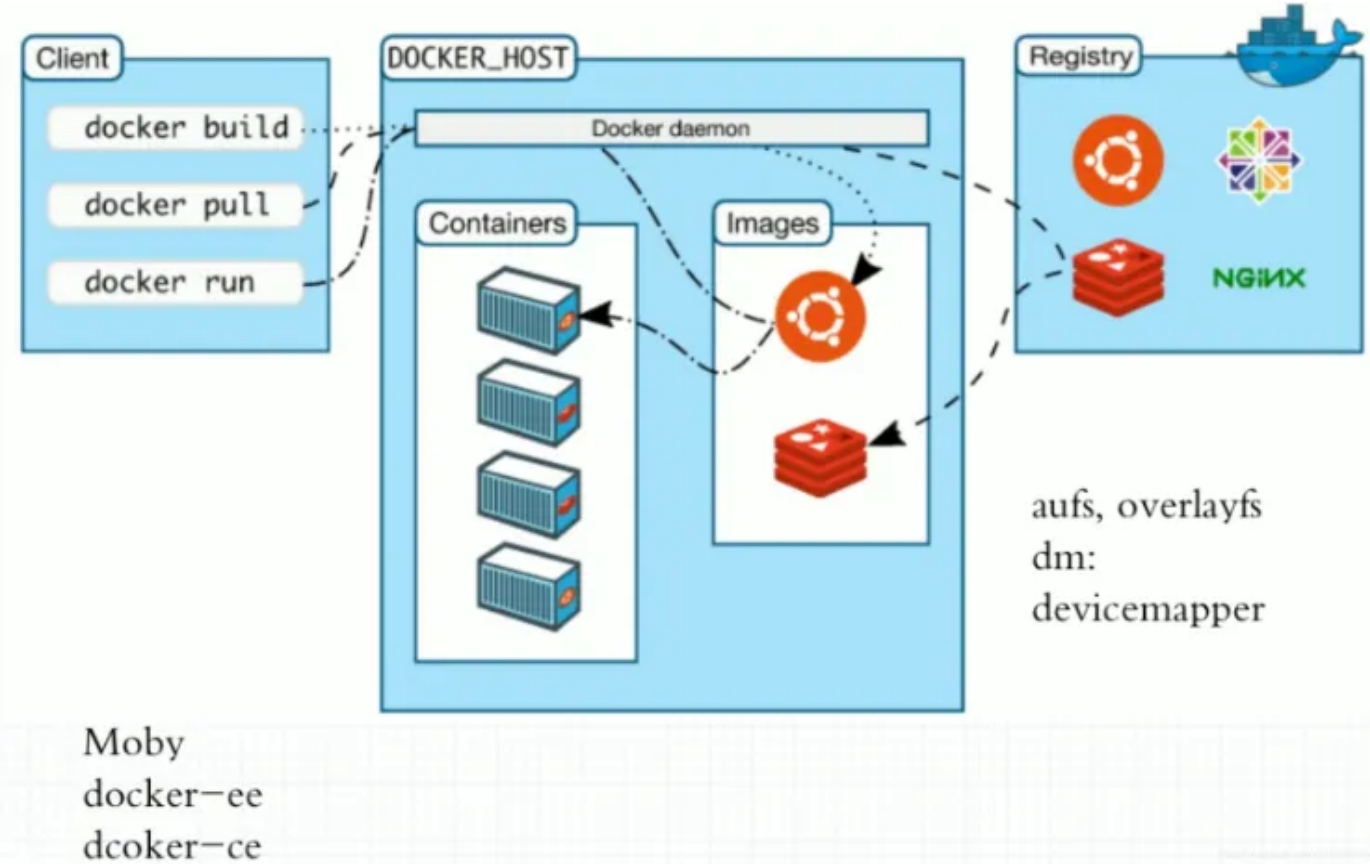
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (arm64v8)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

4.4. 容器的架构



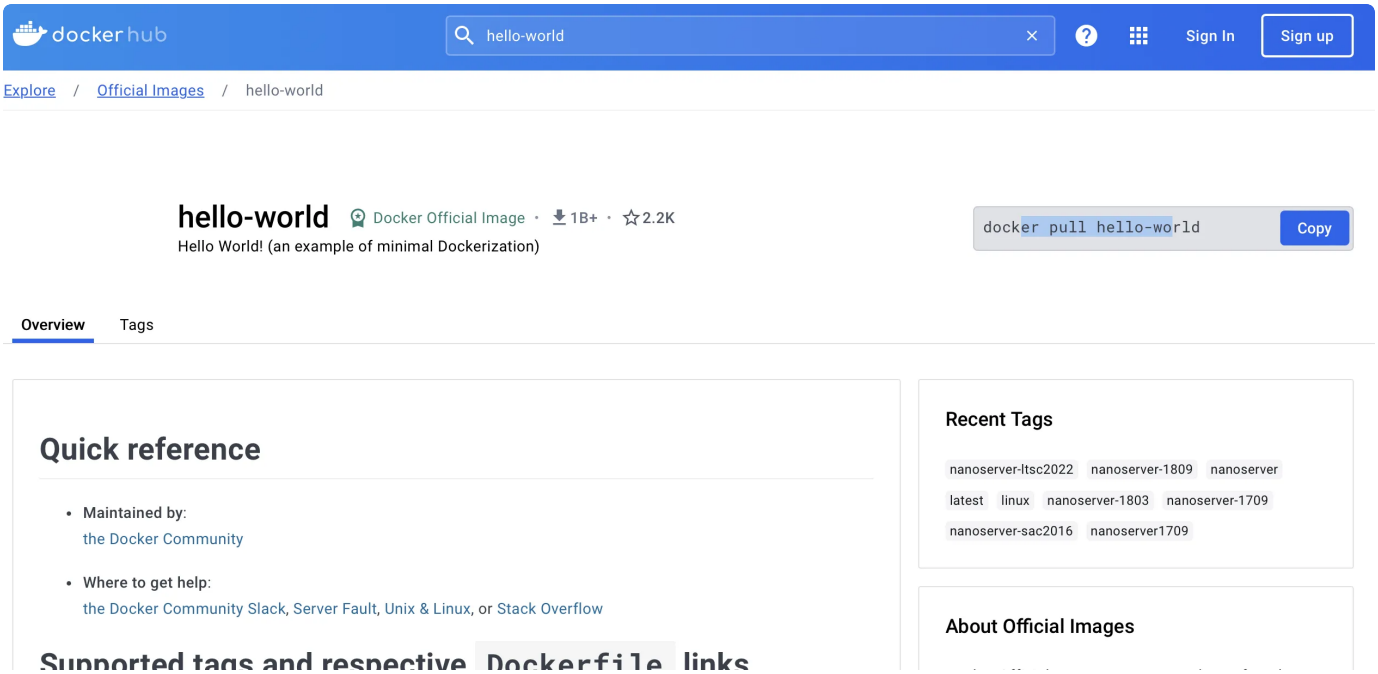
容器的架构：C-S架构

1. client:主要是前端用来控制管理docker镜像，容器的使用（podman/docker）
2. server：负责跟内核交互，实现容器部署的核心组件
3. registry：image仓库
    a. 公用：hub.docker.com mirrors.aliyun.com
    b. 私有镜像仓库：练习和考试环境用的就是这个
    c. 企业级的私有仓库：harbor仓库部署（高级运维）

4.5. 常见的容器管理

4.5.1. 查找镜像

1. 方法1



2. 方法2

```
hellobaton/cypress                                                                0
[root@localhost ~]# docker search hello-world
NAME                                      DESCRIPTION                                     STARS     OFFICIAL
hello-world                               Hello World! (an example of minimal Dockeriz…   2198      [OK]
rancher/hello-world                       This container image is no longer maintained…   6
okteto/hello-world                                                                        0
tutum/hello-world                         Image to test docker deployments. Has Apache…   90
thomaspoignant/hello-world-rest-json      This project is a REST hello-world API to bu…   2
kitematic/hello-world-nginx               A light-weight nginx container that demonstr…   152
dockercloud/hello-world                   Hello World!                                    20
ansibleplaybookbundle/hello-world-apb     An APB which deploys a sample Hello World! a…   1
ansibleplaybookbundle/hello-world-db-apb  An APB which deploys a sample Hello World! a…   2
crccheck/hello-world                      Hello World web server in under 2.5 MB          23
strimzi/hello-world-consumer                                                              0
strimzi/hello-world-producer                                                              0
koudaiii/hello-world                                                                      0
businessgeeks00/hello-world-nodejs                                                        0
freddiedevops/hello-world-spring-boot                                                     0
strimzi/hello-world-streams                                                               0
```

4.5.2. 下载、运行镜像

> docker run #下载镜像+创建容器+运行容器，就会生成一个容器
>
> docker ps #查看正在运行的容器
>
> docker ps –a #查看所有容器（包含正在运行的和已经停止容器）
>
> docker rm 690b9b43a123 #删除容器（容器的数据就丢失了）
>
> docker rmi hello-world:latest #删除镜像（必须没有任何基于这个镜像的容器在运行）

docker images #查看镜像

docker pull hello-world #下载images，不运行

docker create hello-world #使用下载好的镜像创建容器

docker start c9001752785c #运行创建好的容器

docker run -d --name glab_nginx_test -p 81:80 nginx:alpine

run: 下载、创建、运行容器

-d：后台运行

--name：指定容器的名字（不要重复。可以不指系统可以自动生成）

-p：指定映射端口81:80，访问属主机的IP地址的81端口，转成容器的80端口

nginx:alpine：镜像名字

docker exec -it 0cc286a97d15 /bin/sh

exec：在容器中执行某个命令

-it：交互式进入容器

/bin/sh：进入容器时候运行的命令，进入shell，配置容器

docker cp ./test.sh ngnix_test_v2:/tmp/ #从属主机复制到容器

docker cp ngnix_test_v2:/tmp/aaa ./ #从容器复制到属主机

cp：属主机的文件复制到容器

4.5.3. 数据永久存放

问题：容器停止，数据全部丢失

docker run -d --name glab2 -p 86:80 -v /root/log_test:/usr/share/nginx/html/ nginx:alpine

> -v：属主机的目录映射到容器里面的目录，实现数据的永久存储

4.5.4. 容器的自动重启策略

容器的自动重启策略：

always：自动重启

unless-stopped：只在容器关闭、停止的时候重启

on-failure：只在失败的时候重启

默认行为：不自动重启

批量删除容器

docker rm `docker ps -aq`

docker run -d --name glab2 -p 82:80 --restart=always nginx:alpine

4.5.5. 日志查看

> docker logs glab2
>
> 注意点：
>
> 1. logs查看方法，只能看做了软链接的日志输出
>
> / # ls -l /dev/stdout
>
> lrwxrwxrwx 1 root root 15 Mar 9 08:08 /dev/stdout -> /proc/self/fd/1

```
/ # ls -l /dev/stderr
lrwxrwxrwx    1 root     root            15 Mar  9 08:08 /dev/stderr -> /proc/self/fd/2
```

4.6. Dockerfile

4.6.1. 概述

应用场景：通过一个文件Dockerfile，生成自定义镜像，自动生成镜像的方法，制作镜像

- 大规模使用容器的场景手动特别麻烦
- 编辑文件，编辑脚本的方式来自动按需创建我们需要的镜像
- 语法接近于playbook，便于维护
- 自动创建

4.6.2. 格式

Bash

```
[root@localhost ~]# cat Dockerfile
FROM nginx:alpine
LABEL author="GLAB"

COPY ./index.html /usr/share/nginx/html/

EXPOSE 80 8080 9090

CMD ["nginx","-g","daemon off;"]

```

FROM：指定basic 镜像 centos 。ubuntu　nginx

LABEL：标记容器的基本信息

COPY：属主机的文件送到容器里面 。cp

EXPOSE：暴露容器的端口，80

CMD：指定容器的入口命令 #容器能否长期运行的入口配置

构建镜像

docker build –t ngnix:glab_v1 .

一定要在Dockerfile文件的目录上

4.7. Podman的使用

```Bash
podman和docker
都是管理容器的前端命令

yum install -y containers-tool

#解决链接到内部私有仓库问题
#1配置文件
cp /etc/containers/registries.conf .config/containers/

[root@serverb ~]# cat .config/containers/registries.conf
unqualified-search-registries = ["registry.lab.example.com"]
[[registry]]
insecure = true
blocked = false
#2命令后登录内部仓库
podman login registry.lab.example.com
admin/redhat321

[root@serverb ~]# podman search registry.lab.example.com/
NAME                                          DESCRIPTION
registry.lab.example.com/rhel8/mariadb-103
registry.lab.example.com/rhel8/mariadb-105
registry.lab.example.com/rhel8/httpd-24
registry.lab.example.com/library/nginx
registry.lab.example.com/ubi7/ubi
registry.lab.example.com/ubi8/ubi
registry.lab.example.com/ubi9-beta/ubi
registry.lab.example.com/ubi8/python-38
registry.lab.example.com/ubi8/httpd-24
registry.lab.example.com/rhel8/php-74

#容器的常规操作
docker run -d --name test123 -v ./test:/dir1 -p 80:80 registry.lab.example.com/library/nginx

#把容器做成服务的形式（考题本有）
```

5. Shell脚本（占比较少，工作比较重要）