1. chmctrl+alt+fn切换虚拟控制台（tty1图形界面、tt2~tty6字符界面）
2. pwd查看当前工作目录
3. cd切换工作目录

格式：cd [目标文件位置]

（以/开始的绝对路径或~user表示用户user家目录。点表示当前目录，点点表示上一层目录）

1. ls查看目录内容

格式：ls [选项]... [目录或文件名]...

选项：-l：以长格式显示

-A：包含名称以.开头的隐藏文件

-d：显示目录本身（而不是内容）的属性

-h：提供易读的容量单位（K,M等）

-R：表示递归显示

1. cat /etc/redhat-release显示发行信息
2. uname -r列出内核版本
3. lscpu列出cpu处理信息
4. cat /proc/meminfo检查内存信息、空闲情况
5. hostname列出当前主机名称

hostname 新主机名 设置主机名（临时修改）

1. ifconfig列出已激活网卡信息

ifconfig [接口名]

ifconfig -a

ifconfig [接口名] IPv4地址/掩码长度（临时修改）

1. poweroff关机
2. reboot重启
3. mkdir创建目录

格式：mkdir [-p] [/路径/]目录名...

联通父目录一起创建

1. touch创建空文档

格式：touch 文件名...

1. less分屏阅读工具（支持查找、前后翻页等功能）

格式：less [选项] 文件名...

1. head显示文档前几行

格式：head -n 数字（行数，默认是10行） 文件名

1. tail显示文档后几行

格式：tail -n 数字 文件名

1. grep输出包含指定字符串的行

格式：grep [选项]... ‘查找条件’ 目标文件

命令行 | grep [选项] ‘匹配模式’

常用选项

-v，取反匹配

-i，忽略大小写

-r，递归查询

常用的匹配模式

word包含字符串word

^word以字符串开头的word开头

word$以字符串word结尾

1. cat查看文本文件内容

格式：cat -n（加上行号功能） 文本名字

1. ctrl+l：清空整个屏幕

ctrl+u：清空至行首

ctrl+w：往回删除一个单词（以空格为界定）

ctrl+c：废弃当前编辑的命令行

esc+.：粘贴上一个命令参数

1. mount挂载操作

格式：mount 设备路径 挂载点目录

showmount -e [服务器地址] 显示挂载路径

mount -a 自动挂载

1. umount卸载挂载

格式：umount 挂载点目录

1. 通配符

\*表示任意多个字符

？表示单个字符

1. z]多个字符或连续范围中的一个，若无则忽略，只能匹配0-9

{a,min,xy}多组不同的字符串，全匹配

1. 别名：简化复杂指令

查看已设置的别名：alias 别名名称

定义新的别名名称：alias 别名名称=’实际执行的命令行’

取消已设置的别名：unalias [别名名称]（临时有效）

临时取消别名（加反斜杠）：\cp

1. cp复制

格式：cp [选项]... 原文件... 目标路径

选项：-r：递归，复制目录时必须有此选项

-p：复制时保留原文件的属性和权限

1. rm删除

格式：rm [选项]... 文件或目录...

选项：-r -f：递归删除（含目录）、强制删除，删除前无提示

1. mv移动

格式：mv [选项]... 原文件... 目标路径

1. vim编辑工具
   1. 命令模式操作
      1. 移动光标：↑↓←→上下左右
      2. 光标行内跳转：home键或^、数字0 跳转至行首

end键或$键 跳转至行尾

* + 1. 全文翻页：pgup键、pgdn键 向上翻页、向下翻页
    2. 光标行内跳转：1G或gg 跳转至文件的行首

G 跳转至文件的末尾行

* + 1. 复制：yy、#yy 复制到光标处的一行、#行
    2. 粘贴：p、P 粘贴到光标处之后、之前
    3. 删除：x或delete 删除光标处的一行、#行（或剪切）

d^ 从光标处之前删除至行首

d$ 从光标处删除至行尾

C（大写C） 从光标处删除至行尾，并进入插入模式

* + 1. 文本查找：/word 向后查找字符串“word”

n/N 跳至后/前一个结果

* + 1. 撤销编辑：u 撤销最近的一次操作（可以撤销多次）

U 撤销对当前行的所有操作

ctrl+r 取消前一次撤销操作（反撤销）

* + 1. 保存退出：ZZ 保存修改并退出
  1. 末行模式
     1. 存盘及退出：:w 保存当前文件

:q! 放弃已有更改后强制退出

:wq或:x 保存已有修改后退出

* + 1. 文件操作：:w /root/newfile 另存为其他文件

:r /etc/filesystems 读入其他文件内容

* + 1. 行内替换字符串：:s/old/new 替换当前行第一个“old”

:s/old/new/g 替换当前行所有的“old”

* + 1. 区域内替换字符串：:n,m s/old/new/g 替换n-m行所有的“old”

:%s/old/new/g 替换文件内所有的“old”

* + 1. 编辑器设置：:set nu|nonu 显示/不显示行号

:set ai|noai 启用/关闭自动缩进

补充：vim 可视模式 在命令模式下 按ctrl+v 进入可视视图 可以选择列 按大写I进入插入模式，输入内容 按Esc退出命令模式

1. ctrl+shift+t在当前终端中再开一个终端（ctrl+pageup ctrl+pagedown切换终端）
2. ctrl+shift+’+’全屏下字体变大 ctrl+shift+’-’全屏下字体变小
3. ssh -X 用户名@IP地址 远程管理
4. 使用rpm命令管理软件

rpm -q 软件名：查询软件

rpm -ivh 软件名-版本信息.rpm：安装软件

rpm -e 软件名：卸载软件

rpm -qa | grep nfs：根据关键字过滤软件包

1. yum软件包管理

yum repolist列仓库

yum list 软件包 列软件

yum clean all 清缓存

yum -y install 软件名 安装软件

yum -y remove 软件名 卸载软件

yum-config-manager --add-repo 软件仓库地址 创建yum源

yum provides 文件名 查看该文件名是哪个软件产生的

yum仓库搭建

1. 网络yum
   1. 服务器

安装httpd服务，并启动:systemctl start httpd

创建文件夹，挂载，检测：

mkdir /var/www/html/rhel7

mount /ISO/rhel-server-7.2-x86\_64-dev.iso /var/www/html/rhel7

开机自动挂载

/ISO/rhel-server-7.2-x86\_64-dev.iso /var/www/html/rhel7 iso9660 defaults 0 0

* 1. 客户机

修改配置文件vim /etc/yum.repos.d/rhel7.repo

[rhel7]

name=rhel7

baseurl=http://192.168.4.254/rhel7

enabled=1

gpgcheck=0

1. 本地Yum

创建文件夹，挂载，测试：

mkdir /var/www/html/rhel7

mount /ISO/rhel-server-7.2-x86\_64-dev.iso /var/www/html/rhel7

开机自动挂载：

/ISO/rhel-server-7.2-x86\_64-dev.iso /var/www/html/rhel7 iso9660 defaults 0 0

指定yum仓库

[other]

name=other

baseurl=file:///var/www/html/rhel7

enabled=1

gpgcheck=0

1. 自定义yum

准备安装包\*.rpm

使用createrepo工具创建仓库档案crateerepo 包目录

指定新yum仓库

[other]

name=other

baseurl=file:///tools/other

enabled=1

gpgcheck=0

1. wget下载工具

wget 软件包的UIRL网址

wget 软件包的UIRL网址 -O /目录新路径/新文件名

1. nmcli网卡相关配置

nmcli connection show

nmcli connection show “连接名” 查看网络连接

nmcli connection modify “连接名” 参数1 值1... 修改网络连接配置

常用参数：

ipv4.method auto | manual

ipv4.address “IP地址/掩码长度 [默认网关]”

ipv4.dns DNS服务器地址

coiinnection.autoconnect yes|no

nmcli connection dowm “连接名” 禁用网络连接

nmcli connection up “连接名” 启用网络连接

nmcli connection add type ethernet con-name eth1 ifname eth1 添加网卡

nmcli connection delete eth0 删除网卡信息

1. host DNS测试

host 域名或IP地址

host域名或IP地址 DNS服务器地址

1. id 用户名 查看用户相关信息
2. 管道命令

命令 | 命令 将前面命令的输出结果，交由后面的命令再处理一次

1. 重定向

>：覆盖重定向

>>：追加重定向

1. echo 内容 在屏幕上输出想输出的内容
2. useradd添加用户

useradd [选项]...用户名

常用选项

-u用户id

-d家目录路径

-s登录的shell

-g基本组

-G附加组

1. passwd设置用户登录密码

passwd [用户名]

echo ‘密码’ | passwd --stdin 用户名

1. usermod修改用户属性

usermod [选项]... 用户名

常用选项

-u 用户id

-d 家目录位置

-s登录shell

-g基本组

-G附加组

1. id查看用户信息

id 用户名

1. userdel删除用户

userdel [-r] 用户名

1. groupadd添加组

groupadd [-g 组ID] 组名

1. gpasswd管理组成员

gpasswd -a 用户名 组名 添加成员

gpasswd -d 用户名 组名 删除成员

1. groupmod修改组属性

groupmod [-g 组ID] [-n 新组名] 组名

1. groupdel删除组

groupdel 组名

1. tar

制作tar备份包

tar -zcf 备份文件.tar.gz 被备份的文档...

tar -jcf 备份文件.tar.bz2 被备份的文档...

tar -Jcf 备份文件.tar.xz 被备份的文档...

常用参数：

-c：创建归档

-x：释放归档

-f：指定归档文件名称

-z、-j、-J：调用.gz、.bz2、.xz格式的工具进行处理

-t：显示归档中的文件清单

-P：保持归档内文件的绝对路径

-C：指定释放位置

查看tar备份包内容

tar -tf 备份文件.tar.gz

从tar备份包恢复文档

tar -zxf 备份文件.tar.gz [-C 目标文件]

1. date查看/调整系统日期时间

date +%Y显示年份

date +%m显示月份

date +%d显示日期

date +%Y-%m-%d显示年月日

date +%F显示年月日

date -s “年-月-日 时:分:秒”设置时间

date +%R显示时间

date +”%Y-%m-%d %H:%M:%S”显示时间

1. chmod设置基本权限

chmod -R 归属关系+-=权限类别 文档

u：属主

g：属组

o：其他人

r：读

w：写

x：执行

权限数值化：r=4,w=2,x=1

采用数值形式设置权限

chmod -R nnn 文档

chmod -R xnnn 文档（x为特殊权限）

1. chown设置文档归属

chown -R 属主 文档...

chown -R :属组 文档...

chown -R 属主:属组 文档...

1. set gid设置组特殊权限（继承父目录的所属组身份，适用于目录），数值为2

chmod g+s 目录...

1. set uid设置用户权限位（适用于文件，会发生身份的转变），数值为4

chmod u+s 文档...

1. set bit设置其他人的x位（阻止用户滥用w写入权限，适用于文件），数值为1

chmod o+t 文档...

1. acl访问控制列表

getfacl 文档... 查看策略

setfacl -R -m u:用户名:权限类别 文档... 添加用户策略

setfacl -R -m g:组名:权限类别 文档... 添加组策略

setfacl -R -b 文档... 清除所有acl

setfacl -x u:用户名 文档... 删除指定的acl

1. find查找文件

find [目录] [条件1] [-o（或者，二者选一，不写就是全部）] [条件2]...

常用条件表示：

-type 类型（f文件、d目录、l快捷方式）

-name “文档名称”

-size +|-（大于|小于） 文件大小（k,M,G）

-user 用户名

-group 组名

-exec 将find的结果做进一步动作（以{}代替find的每一个结果，逐个处理，遇\;结束）

-iname忽略大小写按名称查找

-maxdepth限制目录查找的深度（最大层数）

-mtime根据文件修改时间查找（都是过去的时间，+10过去的10天之前，-10，过去的10天之内）

1. cron计划任务（24小时制）

crontab -e [-u 用户名] 编辑

crontab -l [-u 用户名] 查看

crontab -r [-u 用户名] 删除

分 时 日 月 周 任务命令行（绝对路径）

1. 分区相关命令

使用fdisk分区工具（MBR）

fdisk -l /dev/sda查看分区表

fdisk 硬盘设备修改硬盘的分区表

常用的交互指令：

m：列出帮助指令

p：查看现有的分区表

n：新建分区

d：删除分区

q：放弃更改并退出

w：保存更改并退出

partprobe识别新分区

mkfs格式化分区（格式化多次xfs需要加-f，进行强制格式化）

mkfs工具集

mkfs.ext3 分区设备路径

mkfs.ext4 分区设备路径

mkfs.xfs 分区设备路径

mkfs.vfat -F 32分区设备路径

访问文件系统

使用mount命令挂载，并访问测试

mkdir /mnt/part2

mount /dev/vdb2 /mnt/part2

实施开机挂载

配置文件/etc/fstab（设备路径 挂载点 类型 参数 备份标记 检测顺序）

blkid 设备名称查看文件系统

lsblk列出所有blk设备

1. LVM相关命令
2. 添加
   1. 准备至少一个分区（/dev/vdc1）
   2. 创建物理卷：pvcreate 空闲分区（/dev/vdc1）
   3. 创建卷组：vgcreate 卷组名（aaa） 空闲分区（/dev/vdc1）
   4. 创建逻辑卷：lvcreate -L 大小 -n 名称（bbb） 卷组名（aaa）
   5. 格式化并挂载使用：mkfs.ext4 /dev/aaa/bbb

mount /dev/aaa/bbb /mylvm

1. 扩展逻辑卷
   1. 卷组有足够的空间
      1. 直接扩展逻辑卷的空间

lventend -L 新大小（18G或者+2G） 设备名称（/dev/卷组名/逻辑卷名）

* + 1. 扩展逻辑卷文件系统（格式化过的逻辑卷要通知linux内核，否则df看到的大小仍是旧大小）

resize2fx ext3/4 设备

xfs\_growfs xfs 设备

* 1. 卷组没有足够的空间
     1. 扩展卷组

vgentend 卷组名（aaa） 磁盘设备名（/dev/vdc3）

* + 1. 扩展逻辑卷的空间

lventend -L 新大小（25G或者+7G） 设备名称（/dev/卷组名/逻辑卷名）

* + 1. 扩展逻辑卷文件系统

resize2fx ext3/4 设备

xfs\_growfs xfs 设备

1. 删除逻辑卷
   1. 删除LV

umount /mylvm

lvremov 逻辑卷名（/dev/aaa/bbb）

* 1. 删除VG（必须全删除lv，才能卸载）

vgremove 卷组名（aaa）

* 1. 删除PV

pvremove 设备名（/dev/vdc1）

1. 综合分区规划
   1. 指定PE的大小

vgcreate -s PE大小（1M） 卷组名（aaa） 空闲分区（/dev/vdb6）

* 1. 创建逻辑卷的时候指定PE的个数

lvcreate -l PE个数（50） -n 逻辑卷名（bbb） 卷组名（aaa）

* 1. 修改逻辑卷PE的大小

vgchange -s 大小（1M） 卷组名（aaa）

pvs查看物理卷

vgs查看卷组

lvs查看逻辑卷

1. shell脚本

组成：#!环境申明

#注释文本

可执行代码

调试脚本：sh -x 脚本名称

编写技巧：管道使用|：将钱一条命令的标准输出交给后一条命令处理

>：将前面命令的正确输出，写入到文本文件中，只收集正确信息

1. ：将前面命令的错误输出写入到文本文件中，只收集错误信息

&>：将前面命令的正确与错误输出，写入到文本文件中，正确与错误都收集

echo ‘文本字符串’自定义输出

&>/dev/null屏蔽无价值、干扰性的信息

>>/var/log/foo.log记录有价值的信息

‘’（单引）：取消所有特殊字符意义

$[]：运算+-\*/%

$()与``反撇号：将命令的输出结果，作为参数

造数工具{20..80}：列出20-80的所有数字

>&2把正确的转换为错误的

条件测试及选择：

$?：命令行/程序的退出状态值（0表示成功，非0表示异常或失败）

常用的测试状态

检查文件的状态

-e：文档是否存在，存在为真，不存在为假

-d：存在且为目录，存在为真，不存在为假

-f：存在且为文件，存在为真，不存在为假

-r：存在且具备读权限，存在为真，不存在为假

-w：存在且具备写权限，存在为真，不存在为假

-x：存在且具备执行权限，存在为真，不存在为假

比较整数大小（带e都有等于等于二字，g代表大于，l代表小于）

-gt：大于

-ge：大于等于

-eq：等于

-ne：不等于

-lt：小于

-le：小于等于

字符串比对

==：字符串相等为真

!=：字符串不相等为真

if选择结构：

if单分支结构（当条件满足时，作xx处理）

if [条件判断];then

命令序列xx

fi

if双分支处理（当条件满足/不满足时，分别做xx、yy处理）

if [条件判断];then

命令序列xx

else

命令序列yy

fi

if多分支处理

if [条件判断1];then

命令序列xx

elif [条件判断2];then

命令序列yy

else

命令序列zz

fi

for列表循环（遍历/列表循环，根据变量的不同取值，重复执行xx处理）

for 变量名 in 值列表

do

命令序列

done

1. selinux

三种运行模式（enforcing强制、permissive宽松、disable彻底禁用，变成disable的模式都需要更改配置文件并重启）

永久切换模式：修改配置文件/etc/selinux/config

临时切换配置：setenforce 0临时设置成宽松模式

setenforce 1 临时设置成强制模式

查看selinux配置：getenforce

1. filewall防火墙配置

规则：先看源IP，再看有没有源IP对应的策略，如果有，则直接进入该区域，如果没有，进入默认区域

常用的四个区域：public：仅允许主机的sshd、dhcp、ping等少数几个服务

trusted：允许任何访问

block：阻塞任何来访请求，明确拒绝，有返回

drop：丢弃任何来访的数据包，节省资源，没返回

查看防火墙规则列表

firewall-cmd --list-all [--zone=区域名]

firewall-cmd --list-all-zones

firewall-cmd --get-zones

firewall-cmd --get-services

firewall-cmd --get-default-zone

指定默认安装区域：firewall-cmd --set-default-zone=区域名

添加源IP的规则设置（禁止某一IP访问）：firewall-cmd --zone=区域名 --add-source=ip地址

设置成为永久有效：firwall-cmd --permanent --zone=区域名

重载：firwall-cmd --reload（要使策略重新生效，需重载）

添加网段源地址：firewall-cmd --zone=区域名 --add-source=网段地址或IP

添加服务：firewall-cmd --zone=区域名 --add-service=服务名

删除服务：firewall-cmd --permanent --zone=区域名 --remove-service=服务名

端口转发（防火墙功能）：filewall-cmd --permanent --zone=区域名 --add-forward-port=port=源端口号:proto=tcp:toport=目的端口号

1. 配置用户环境

个人：~/.bashrc 全局：/etc/bashrc

自定义命令：alias qstat=’/bin/ps -Ao pid,tt,user,fname,rsz’

1. 配置聚合链路（网卡绑定、链路聚合）

作用1：轮询式的流量负载均衡

作用2：热备份连接冗余（连接类型是team）

步骤1：网络接口准备（2块或2块以上物理网卡）

步骤2：新建虚拟网卡：为聚合连接提供配置（类型、连接名、运行器）

nmcli connection add type team

con-name team0（配置文件的名字） ifname team0 autoconnect yes

config '{"runner": {"name": "activebackup"}}'

如果敲错：nmcli connection delete team0

步骤3：添加成员（添加奴隶）：为成员网卡提供配置（类型学、连接名、主连接）

nmcli connection add type team-slave con- name team0-1 ifname eth1 master team0

nmcli connection add type team-slave con- name team0-2 ifname eth2 master team0

如果错误nmcli connection delete team0-1

nmcli connection delete team0-2

步骤4：配置team0的IP地址

nmcli connection modify team0 ipv4.method manual ipv4.addresses 192.168.1.1/24 connection.autoconnect yes

步骤5：激活所有配置

nmcli connection up team0

nmcli connection up team0-1

nmcli connection up team0-2

如果激活失败（失败哪个删除哪个）

nmcli connection delete team0

nmcli connection delete team0-1

nmcli connection delete team0-2

步骤6：验证

teamdctl team0 state

1. IPV6配置
2. 使用命令行修改连接参数

nmcli connection show 获取连接名称

nmcli connection modify ‘System eth0’ ipv6.method manual ipv6.addresses 2003:ac18::3055/64 connection.autoconnect yes

1. 激活更改过的连接（必要时先down再up）

nmcli connection up ‘System eth0’

1. ping6 2003:ac18::305
2. SMB共享

服务端：

1. yum -y install samba装包
2. pdbedit -a 用户名 添加用户

pdbedit -L 用户名 查询用户

pdbedit -x 用户名 删除用户

1. mkdir /common 创建共享目录
2. vim /etc/samba/smb.conf修改配置文件

public：公共的为no，默认的为no，表示需要进行密码验证才可以进入

browseable：是否可以浏览，默认为yes，表示是否可见，是否为隐藏共享

read only：是否为只读，默认为yes

writye list：可读用户列表，默认无，需要个别指定

valid user：有效用户，默认任何用户，也可以指定

host allow：允许访问的客户机地址，需禁止，一般不用，直接配置防火墙

host deny：禁止访问的客户机地址，需禁止，一般不用，直接配置防火墙

1. systemctl restart smb 重启smb服务
2. firewall-cmd --set-default-zone=trusted 设置防火墙的默认安全区域是trusted
3. setsebool -P samba\_export\_all\_rw=on打开读写权限

setsebool -P samba\_export\_all\_ro=on打开只读权限

1. setfacl -m u:chihiro:rwx /devops 设置用户acl（读写共享时需要）

客户端

1. yum -y install cifs-utils安装客户端软件包
2. mount //服务器地址/共享名 挂载点 挂载文件系统
3. vim /etc/fstab编辑开机挂载配置

//服务器地址/共享名 挂载点 defaults,cifs \_netdev 0 0

1. NFS共享

服务端：

1. yum -y install nfs-utils安装服务端所用软件（rpm -q nfs-utils查询是否已经安装）
2. vim /etc/exports修改配置文件

/etc \*(rw)

/ \*(ro)

1. exportfs -r重新更新配置
2. systemctl restart nfs-server重启服务

客户端：

1. showmount -e 服务器地址 列出有哪些NFS共享资源
2. mount 服务器地址:文件夹路径 挂载点 手动挂载NFS共享
3. vim /etc/fstab配置开机挂载

服务器地址:文件夹路径 挂载点 defaults,nfs \_netdev 0 0

1. iscsi网络磁盘

服务端：

1. yum -y install targetcli安装targetcli软件
2. fdisk划分分区
3. 进行配置共享存储
   1. backstores/block create nsd /dev/vdb1建立后端存储
   2. iqn.yyyy-mm.倒序域名:自定义标识（用来识别target磁盘组，也用来标识客户机身份） 创建箱子并起名（建立iqn磁盘组）
   3. iscsi/iqn.2019-03.com.example:server0/tpg1/luns create /backstores/block/nsd 将nsd后端存储放入箱子中（磁盘组 绑定 后端存储 luns）
   4. iscsi/iqn.2019-03.com.example:server0/tpg1/acls create iqn.2019-03.com.example:test 设置客户端访问控制acl（客户端访问时声称的名字）
   5. iscsi/iqn.2019-03.com.example:server0/tpg1/portals create 172.25.0.11设置提供服务的IP地址

客户端：

1. yum -y install iscsi-initiator-utils.i686安装客户端软件
2. vim /etc/iscsi/initiator.iscsi修改备份文件指定客户端名字
3. systemctl restart iscsid刷新iqn标识
4. iscsiadm --mode discoverydb --type sendtargets --portal 127.25.0.11 --discover寻找服务端，发现iscsi磁盘
5. systemctl restart iscsi重启客户端iscsi服务
6. 挂载进行操作
7. apache

独立web站点创建

1. yum -y install httpd装包（httpd）
2. echo ‘first web’ > /var/www/html/index.html配置（部署首页文件index.html）
3. systemctl restart httpd起服务（httpd）

虚拟web主机

<virtualhost IP地址:端口>

servername 此站点的DNS域名

documentroot 此站点的网页根目录

</virtualhost>

注意：一旦使用了虚拟web主机功能，所有的站点必须使用虚拟web来实现

网页内容访问

文件夹访问权限：httpd运行身份（用户/组）：apache

客户端地址限制

使用<directory>配置区段

每个文件夹自动继承其父目录的acl访问权限

除非针对子目录有明确限制

<directory 目录的绝对路径>

require all denied禁止任何客户机访问

require all granted允许任何客户机访问

require ip IP或网段地址 仅允许部分客户机访问

</directory>

selinux策略保护

标准web目录

semanage fcontext -l | grep httpd\_sys\_content查看相关目录的上下文

restorecon -R 新建的目录名 新建目录初始化

新增web目录

方式1：参照标准目录，重设新目录的属性：chcon -R --reference=模版目录 新目录

方式2：将新目录增加到预设的标准目录范围：semanage fcontext -a -t httpd\_sys\_content\_t ‘新目录(/.\*)?’

标配web端口：semanage port -l | grep http\_port查看web端口

允许开放其他web端口：semanage port -a -t 类型 -p tcp 端口号（-a添加、-t类型、-p协议）

动态网站部署

1. 配置虚拟主机

<virtualhost IP地址:端口>

servername 此站点的DNS域名

documentroot 此站点的网页根目录

alias / 新目录

</virtualhost>

1. 下载测试页面

wget http://classroom/pub/materials/webinfo.wsgi

1. 重启服务

systemctl restart httpd

1. 安装可以解释python页面的程序，配置解释该页面

systemctl restart httpd

1. 添加配置文件内容，调用刚才的程序

<virtualhost \*:80>

servername webapp0.example.com

documentroot /var/www/nsd02

wsgiscriptalias / /var/www/nsd02/webinfo.wsgi

</virtualhost>

1. 改变端口为8909

listen 8909

<virtualhost \*:8909>

servername webapp0.example.com

documentroot /var/www/nsd02

wsgiscriptalias / /var/www/nsd02/webinfo.wsgi

</virtualhost>

1. 设置selinux

semanage port -a -t http\_port\_t -p tcp 8909

semanage port -l | grep http\_port 确认结果

1. 重启服务并检查结果

systemctl restart httpd

安全web服务（https）

1. 安装相关服务

yum -y install mod\_ssl

1. 下载证书文件

cd /etc/pki/tls/certs

wget <http://classroom/pub/tls/certs/server0.crt（网站证书）>

wget <http://classroom/pub/example-ca.crt（根证书）>

1. 下载密钥（私钥）

cd /etc/pki/tls/private/

wget <http://classroom/pub/tls/private/server0.key>（私钥）

1. 调整web配置

vim /etc/httpd/conf.d/ssl.conf

document “/var/www/html”

servername server0.example.com:443

sslcertificatefile /etc/pki/tls/certs/server0.crt指定网站证书

sslcertificatkeyfile /etc/pki/tls/private/server0.key指定密钥（私钥）

sslcacertificatefile /etc/pki/tls/certs/example-ca.cert指定根证书

1. 重启服务并验证

systemctl restart httpd

https://server0.example.com

1. parted分区工具（gpt分区）

步骤1：重建分区表

mktable gpt

步骤2：新建分区

mkpart primary （分区名） 文件系统类型 起始位置 结束位置

步骤3：输出磁盘信息

print

步骤4：选择显示单位

unit

步骤5：退出

quit

1. 构建交换分区

步骤1：格式化分区设备

mkswap 空闲分区或设备

步骤2：交换分区的相关操作

swapon 交换设备 启用

swapon -s 查看

swapoff 交换设备 停用

自动配置交换分区

修改/etc/fstab配置文件

/dev/vdb1 swap swap defaults 0 0

swapon -a用于检测/etc/fstab的信息并挂载

1. 部署邮件服务postfix
2. 安装包（默认已经安装并开机启动）

yum -y install postfix

1. 修改/etc/postfix/main.cf配置文件

myorign = server0.example.com默认的域名后缀，是@后面的部分

inet\_interfaces = all允许本机所有网卡提供邮件服务，只要能ping通，就能利用该网卡发邮件

mydestination = server0.example.com判断为笨鱼邮件（当收件人的域名后缀与目的地址一致时，判断为本域邮件）

1. 重启服务

systemctl restart postfix

1. mail命令

mail发信操作

mail -s ‘邮件主题’ -r 发信人 [收件人[@收件域]（邮件内容遇到.结束，-s主题，-r源头，指发件人，不写后缀即默认后缀）](mailto:收件人[@收件域]（邮件内容遇到.结束）)

mail收信操作

mail [-u 用户名]（N代表新邮件，输入数字并回车就可以查看邮件）

1. 历史命令history

管理/调用曾经执行过的命令

history：查看历史命令列表

history -c：清空历史命令

!n：执行命令历史中的第n条命令

!str：执行最近一次以str开头的历史命令

调整历史命令的数量vim /etc/profile

HISTSIZE=1000

1. du，统计文件的占用空间

du [选项]...[目录或文件]...

-s只统计每个参数所占用总空间的大小

-h：提供易读容量单位（K、M等）

1. wc统计命令

wc -l：只统计行数

1. ln（软连接、硬链接）

ln，创建软连接（对原始文件做的快捷方式）

ln -s 原始文佳或目录 软连接文件

ln，创建硬链接（对i节点做的快捷方式）

ln 原始文件 硬链接文件

1. man获取命令帮助

man，格式化手册阅读工具

按↑↓pgup pgon键滚动及翻页

按q退出

按/键向后查找关键字（n、N切换）

1. zip归档工具（跨平台压缩工具）

制作.zip压缩包：zip -r 备份文件.zip 被归档的文档...

释放.zip压缩文件（释放压缩文件+解压操作）：unzip 备份文件.zip [-d 目标文件]

1. 源码编译安装软件

安装gcc、gcc-c++、make等工具

下载源码包

./configure配置，指定安装目录/功能模块等选项

make编译，生成可执行的文件

make install安装，将编译好的文件复制到安装目录

测试及应用/维护软件

1. systemctl控制

开启服务：systemctl start 服务名

停止服务：systemctl stop 服务名

重启服务：systemctl restart 服务名

设置开机自动启动：systemctl enable 服务名

设置开机不自启：systtemctl disable 服务名

查看当前是不是开机自启：systemctl is-enabled 服务名

列出活动的系统服务：systemctl -t service

列出所有的系统服务（包含不活动的）：systemctl -t service --all

列出可用运行级别：systemctl -t target

切换到文本模式：systemctl isolate multi-user.target

切换到图形模式：systemctl isolate graphical.target

查看默认级别：systemctl get-default

设置默认级别：systemctl set-default multi-user.target

systemctl set-default graphical.target

1. DNS域名解析服务

单区域解析

服务器：

1. 安装包bind bind-chroot
2. 设置主配置文件/etc/named.conf
3. 修改地址库文件/var/named
4. 重启named服务

客户机：

1. nslookup 域名 或 host 域名检查

多区域（负责多域名的解析）：

服务端：

1. 主配置文件再加一个zone
2. 再建一个地址库文件
3. 起服务

客户端：

1. nslookup 域名 或 host 域名检查

DNS轮询：

服务器：

1. 修改地址库文件（写入三条一样的）
2. 重启服务

客户端：

1. 进行Ping操作

别名：

地址库文件加CNAME

泛域名解析：

服务器：

1. 修改地址库文件（添加一条\*记录）
2. 重启服务

客户端

1. nslookup或host

有规律的泛域名解析

1. 修改配置文件

$GENERATE 1-50 stu$ A 192.168.10.$

DNS子域授权

1. 配置父DNS
2. 配置子DNS
3. 在父域中加，添加子域授权

子域域名. NS 子域的FQDN.

子域的FQDN. A 子DNS的IP地址

1. 测试子域FQDN查询

关闭递归解析：主配置文件（/etc/named.conf，默认开启）：recursion no;

缓存DNS

1. 主服务器：配置文件（/etc/named.conf，添加内容）：forwarders {公共DNS;};
2. 测试：nslookup [www.qq.com](http://www.qq.com) 192.168.4.254

split分离解析

服务器：

1. 配置DNS主配置文件

view “联通”{

match-clients{来源地址1;.....;};

zone “12306.cn” IN {

地址库1;

};

};

acl”联通”{

IP地址1;IP地址2;...

网段1;网段2;...

};

简化match-clients

1. 建立相关地址库文件
2. 重启named服务

客户端：测试分离结果

1. pstree查看进程树

格式：pstree [选项] [PID或用户名]

常用选项：-a：显示完整的命令行

-p：列出对应PID编号

1. ps查看进程快照

格式：ps [选项]...

常用选项：aux：显示当前终端所有进程（a）、当前用户在所有终端下的进程（x），以用户格式输出（u）

-elf：显示系统内所有进程（-e）、以长格式输出（-l）信息，包括最完整的进程信息（-f）

1. top进程动态排名交互式工具

格式：top [-d 刷新秒数] [-U 用户名]

交互操作指令

?：查看帮助（列出可用的按键指令）

1. M：根据%CPU、%MEM降序排列

T：根据消耗进程的TIME降序排列

k：杀死指定的进程

q：退出程序

1. pgrep检索进程（包含就算）

格式：pgrep [选项]... 查询条件

常用选项

-l：输出进程名，而不仅仅是PID

-U：检索指令用户进程

-t：检索指定终端的进程

-x：精确匹配完整的进程名

1. 控制进程

前台启动：输入正常命令行，运行期间占用当前终端

后台启动：在命令行末尾添加“&”符号，不占用当前终端

ctrl=z组合键：挂起当前进程（暂停并转入后台）

jobs命令：查看后台任务列表

fg命令：将后台任务恢复到前台运行（默认为最近的一个任务，后面可以接任务序号）

bg命令：激活后台被挂起任务（默认为最近1个任务 后面可以接任务序号）

干掉进程的不同方法

ctrl=c，终端当前命令程序

kill [-9] PID...、kill [-9] %后台任务编号（只能杀死一个进程）

killall [-9] 进程名（杀死多个进程）

pkill 查找文件（包含就算，误杀几率较高）

-9代表强制杀，不加-9进程自己衰亡

killall -8 -u 用户名 杀死该用户开启的所有进程

1. 查看文本日志消息tail、tailf（实时跟踪日志信息）、less、grep

awk、sed等格式化过滤工具

1. 日志专用分析工具webmin系统管理套件

webalizer、awstats等日志统计套件

1. 用户登录分析

user、who、w命令

查看已登录的用户信息，详细度不同

last（最近登录成功）、lastb（最近登录失败）命令

查看最近登录成功/失败的用户命令

1. journalctl工具

提取由systemd-journal服务搜集的日志

主要包括内核/系统日志、服务日志

常见用法

journalctl | grep 关键词

journalctl -u 服务名 [-p 优先级]

journalctl -n 消息条数

journalctl --since=”yyyy-mm-dd HH:MM:SS” --until=”yyyy-mm-dd HH:MM:SS”

1. DHCP服务器
2. 装包

yum -y install dhcp

1. 配置文件/etc/dhcp/dhcp.conf

subnet 192.168.4.0 network 255.255.255.0 {

range 192.168.4.100 192.168.4.200;

option domain-name-servers 192.168.4.7;

option routers 192.168.4.254;

default-lease-time 600;

max-lease-time 7200;

}

1. 起服务

systemctl restart dhcpd

1. pxe+kickstart网络装机
2. 准备RHEL7安装源（HTTP方式yum库）
3. 修改DHCP配置文件并启动服务

next-server 192.168.4.7

filename “pxelinux.0”

1. 安装tftp服务，提供装机用的内核、初始化文件

yum -y install tftp-server

systemctl restart tftp

1. 提供PXE引导程序、配置启动菜单（部署pxelinux.0文件）

yum provides \*/pxelinux.0查找对应的包

yum -y install syslinux-4.05-12.el7.x86\_64

rpm -ql syslinux | grep pxelinux.0查找pxelinux.0文件

cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/

mkdir /var/lib/tftpboot/pxelinux.cfg

mount /dev/cdrom /mnt/

cp /mnt/isolinux/isolinux.cfg /var/var/lib/tftpboot/pxelinux.cfg/default部署/读取菜单文件

1. 部署内核 启动驱动程序（vmlinuz启动内核 initrd.img启动驱动程序）

cp /mnt/isolinux/vmlinuz /mnt/isolinux/initrd.img /var/lib/tftpboot

1. 部署 图形的模块 背景图片

cp /mnt/isolinux/vesamenu.c32 /mnt/isolinux/splash.png /var/lib/tftpboot

1. 修改菜单文件内容

vim /var/lib/tftpboot/pxelinux.cfg/default

1. 为客户机提供软件源

yum -y install httpd

mount /dev/cdrom /var/www/html/rhel7

1. kickstart自动应答，无人安装

Yum -y install system-config-kickstart

修改yum文件标题为[development]，否则无法安装软件

system-config-kickstart 设置相关选项后保存成文件

1. 利用web服务，共享应答文件

cp /root/ks.cfg /var/www/html

1. 修改菜单文件，指定应答文件

vim /var/lib/tftboot/pexlinux.cfg

append initrd=initrd.img ks=http://192.168.4.7/ks.cfg

1. 新建虚拟机安装测试
2. rsync同步操作

rsync [选项...] 源目标 目标目录

常用选项

-n：测试同步结果，不做实际修改

--delete：删除目标文件夹内多余的文档

-a：递归模式，全有，相当于-rlptgoD

-r：递归模式，包括目录/子目录

-l：保留符号链接文件

-p、-t：保留文件权限、时间标记

-o、-g：保留文件的属主/属组标记

-D：保留设备文件及其他特殊设备

-v：显示详细操作信息

-z：传输过程中启用压缩/解压

rsync+ssh同步

rsync user@host:远程目录/ 列出SSH服务端资源

rsync [...] user@host:远程目录 本地目录（下行）与远程SSH目录保持同步

rsync [...] 本地目录 user@host:远程目录（上行）与远程SSH目录保持同步

1. inotify实时同步

inotifywait [选项] 目标文件夹

-m，持续监控（捕获一个事件后不退出）

-r，递归监控、包括子目录

-q，减少屏幕输出信息

-e，指定监视的modify、move、create、delete、attrib等事件类别

1. cobbler装机平台
2. 准备centos虚拟机

设置防火墙trusted

配置selinux

配置IP地址192.168.4.180/24

利用本地/dev/cdrom挂载，搭建本地yum仓库

主机名cobbler.tedu.cn

1. 安装cobbler组件

yum -y install cobbler cobbler-web dhcp tftp-server pykickstart httpd xinetd

1. 修改配置文件（cobbler）/etc/cobbler/settings

next\_server:192.168.4.180

server:192.168.4.180

manage\_dhcp:1

pxe\_just\_once:1

1. 配置dhcp服务，修改配置文件/etc/cobbler/dhcp.template

:%s /192.168.1/192.168.4/g

1. 下载并解压引导文件（绝对路径）

tar -xPf /Cobbler/cobbler\_boot.tar.gz

1. 开启所需的服务

systemctl restart httpd

systemctl enable httpd

systemctl restart cobblerd.service

systemctl enable cobblerd.service

systemctl restart rsyncd

systemctl enable rsyncd

systemctl restart tftp

systemctl enable tftp

1. 同步刷新cobbler配置

cobbler sync

firefox <https://192.168.4.180/cobbler_web>

用户名：cobbler

密码：cobbler

1. 导入镜像（放在/var/www/cobbler/ks\_mirror）

cobbler import --path=挂载点 --name=导入系统名称（随意起）

1. 进行安装系统测试
2. rhel7密码破解
3. 重启系统，进入recovery恢复模式

按e键，找到linux16行，末尾添加rd.break console=tty0

按ctrl+x启动

1. 以可写方式重新挂载/sysroot，并切换到此环境

switch\_root#mount -o remolunt,rw /sysroot

switch\_root#chroot /sysroot

sh-4.2#

1. 将root密码设置为redhat

echo redhat | passwd --stdin root

1. 重设slinux安全策略标签

touch /.autorelabel 让selinux失忆

1. 先后执行exit、reboot完成修复

exit

reboot

1. shell脚本编程

切换shell环境（默认/bin/bash）

usermod -s 解释器 用户名

useradd -s 解释器 用户名

命令行环境

快捷卷 tab键补齐

历史命令history

history查看历史命令

history -c：清空历史命令

!78：执行历史命令第78条命令

!str：执行最近一次以str开头的历史命令

命令别名alias unalias

标准输入输出

标准输入（从该设备接收用户输入的数据）：设备文件/dev/stdin 文件描述号0 默认设备键盘

标准输出（通过该涉笔向用户输出数据）：设备文件/dev/stdout 文件描述号1 默认设备显示器

标准准错误输出（通过该设备报告执行中的错误信息）：设备文件/dev/stderr 文件描述号2 默认设备显示器

重定向

< 将文本输入来源由键盘改为指定的文件

> 将命令行的正常执行输出保存到文件，而不是直接显示在屏幕上

>> 与>类似，操作是追加

2> 将命令行的执行错误输出信息保存到文件，而不是直接显示在屏幕上

2>> 与2>类似，操作是追加

&> 相当于>和2>，覆盖到同一文件

管道操作（将一条命令的标准输出交给另一条的标准输入，可以使用多个管道）

脚本构成：

#!脚本声明（使用哪种解释器）

#注释信息（步骤、思路、用途、变量含义等）

可执行的语句

脚本执行方式

方法一：指定脚本文件的路径，前提是有x权限

方法二，作为“参数”

bash 脚本文件路径（开子进程）

sh 脚本文件路径（开子进程）

source 脚本文件路径（不开子进程）

. 脚本文件路径（不开子进程）

调试脚本

通过sh -x开启调试模式

shell变量

变量名=变量值 定义/赋值变量

$变量名 引用变量值

echo $变量名、echo ${变量名} 查看变量值

退出定义变量的shell环境时，变量会自动失效

unset 变量名 手动取消

变量种类

环境变量

env列出所有的环境变量

set列出所有变量

PWD、PATH、USER、LOGNAME、UID、SHELL、HOME、PS1、PS2常见的环境变量

预定义变量

$0 当前所在的进程或脚本名

$$ 当前运行进程的PID号

$? 命令执行后的返回状态，0表示正常

$# 已加载的位置变量的个数

$\* 所有位置变量的值

位置变量

$n代表第n个参数

三种定界符

双引号””：允许特殊符号的功能，如$

单引号’’：禁用特殊符号的功能

反撇号``：将命令的执行输出作为参数

read标准输入取值

read [-p “提示信息” ] 变量名

-p可选，-t可指定超过秒数

stty -echo关闭终端输出（无显示）

stty echo 恢复终端输出（显示）

export 局部变量[=变量值]...：为全局变量添加全局属性

export -n 全局变量名...：取消指定的全局属性

算数运算

整数：expr：expr 整数1 运算符 整数2

$[]：$[整数1 运算符 整数2]

let：let 变量=整数1（变量） 运算符 整数2

小数：bc：echo “数值1 比较符 数值2” | bc

条件测试

字符串判断 [= =]等于 [!=]不等于 [-z]为空 [!-z]非空

数字-eq等于 -ne不等于 -gt大于 -ge大于等于 -lt小于 -le小于等于

文件或目录-e存在 -f是否为文件且存在 -d是否为目录且存在 -r是否有可读权限 -w是否有可写权限 -x是否可执行

if：单分支：if 条件测试

then

fi

双分支：if 条件测试

then 命令序列1

else 命令序列2

fi

多分支（嵌套）：if 条件测试1

then 命令序列1

elif 条件测试2

then 命令序列2

else 命令序列n

fi

循环结构

for：

for 变量名 In 值列表

do

命令序列

done

while

while 条件测试

do

命令序列

done

case语句

case 变量值 in

模式1)

命令序列1;;

模式2)

命令序列2;;

....

\*)

默认序列

esac

函数

function 函数名{

命令序列

}

函数名(){

命令序列

}

中断/退出

break：跳出所有循环

countinue：结束本次循环

exit：退出脚本，默认值为0

字符串处理

字串截取

方法一：${var:起始位置:截取长度}（默认编号从0开始）

方法二：expr substr “$var” 起始位置 长度（起始位置，编号从1开始）

方式三：echo $var | cut -b 起始位置-结束位置

路径分割

取目录位置：dirname “字符串”

取文档的基本名称：basename “字符串”

字串替换

只替换第一个匹配的结果：${var/old/new}

替换全部匹配结果：${var//old/new}

按条件掐头去尾

字符串掐头

从左向右，最短匹配删除：${变量名#\*关键字}（#用来删除头部，\*通配）

从左向右，最长匹配删除：${变量名##\*关键字}（#用来删除头部，\*通配）

字符串去尾

从右向左，最短匹配删除：${变量名%关键字\*}

从右向左，最长匹配删除：${变量名%%关键字\*}

变量的初始值处理

取值：${var:-word}

赋值：${var:=word}

shell数组（一个变量存多个值）

定义/赋值数组

方法一，整体赋值：数组名=(值1 值2 .... 值N)

方法二，为单个元素赋值：数组名[下标]=值（下标从0开始）

输出数组元素

获取单个数组元素：${数组名[下标]}

获取所有数组元素：${数组名[\*]}

获取数组元素个数：${#数组名[\*]}

获取连续的多个数组元素：${数组名[\*]:起始下标:元素个数}

expect预期交互

yum -y install expectt

expect << EOF

spawn 需要交互的命令

expect “命令的关键字” “所要输入的内容”（可以有多条）

expect exit

EOF

正则表达式

egrep过滤工具

基本用法：格式1：egrep [选项] “正则表达式” 文件...

格式2：前置命令 | egrep [选项] “正则表达式”

常用选项：-i：忽略大小写

-v：条件取反

-c：统计匹配的行数

-q：静默，无任何输出，一般用于检测

-n：显示出匹配结果的行号

--color：标记显示匹配字符串

基本正则表达式

^匹配行首，^abc

$匹配行尾，abc$

[]集合，匹配集合中的任意单个字符

[^]对集合取反

.匹配任意单个字符

\*匹配前一个字符任意次（不允许单独使用）

\{n,m\}匹配前一个字符n到m次

\{n\}匹配前一个字符n次

\{n,\}匹配前一个字符至少n次

\(\)保留

扩展正则表达式

+至少匹配一次

?最多匹配一次

{n,m}匹配n到m次

()组合为整体，保留

|或者

\b单词边界

sed流式编辑器（非交互）：

主要用法：

格式1：前置命令 | sed [选项] ‘条件指令’（指令：增删改查）

格式2：sed [选项] ‘条件指令’ 文件...

常用命令选项

-n：屏蔽默认输出，默认sed会输出读取文档的全部内容（全部文本）

-i：直接修改文件内容，默认sed只是通过内存临时修改文件，源文件无影响

-r：启用正则表达式，若与其他选项一起连用，应作为首个选项条件，可以是行号或/正则/

行号可以使用单个数字表示单行或者3，5表示连续的多行

省略条件，则默认处理全部文本

匹配正则时，需要使用//

基本处理动作

p打印行：2,4输出2 3 4行；2p;4p输出第2行和第4行

d删除行：2,4d删除2 3 4行

s字符串替换：s/old/new/将每行第1个old替换为new

s/old/new/3将每行的第3个old替换为new

s/old/new/g将所有的old都替换为new

替换操作的分割“/”可改用其他字符，如#、&等

文本块处理操作

i行前插入文本：2iYY在第2行之前添加文本行yy

4,7iYY在第4-7行每一行前添加文本行

a行后插入文本：2aYY在第2行后面插入文本

/^XX/aYY在XX开头的行后添加文本YY

c替换当前行：2cYY将第2行的内容修改为YY

文本的导入/导出操作

r读取文件 ：3r b.txt在第三行下方插入文件b.txt

w保存到文件：3w c.txt将第3行另存为文件c.txt

4,7w c.txt将第4-7行另存为文件c.txt

流控制

!取反

复制剪切

H:模式空间---[追加]--->保持空间（复制）

h:模式空间---[覆盖]---->保持空间（复制）

G：保持空间---[追加]--->模式空间（粘贴）

g:保持空间---[覆盖]--->模式空间（粘贴）

awk基本用法

主要用法：

格式1：前置命令 | awk [选项] ‘[条件]{指令}’

格式2：awk [选项] ‘[条件]{指令}’ 文件

格式3：awk ‘BEGIN{指令} 条件{指令} END{指令}’ 文件

BEGIN：在所有行之前处理，执行一次

{}：逐行处理，逐行读入文本相应的处理，执行n次

END：在所有行后处理，执行一次

常用选项：

-F：指定分隔符，可省略（默认空格或TAB位）

常用指令：

print打印，若有多条指令，可以用分号隔开

内置变量：

FS：保存或设置字段分隔符，例如FS=”:”

$n：指定分隔的第n个字段，分别表示第几列

$0：当前读入的整行文本内容

NF：记录当前处理行的字段个数（列数）

NR：记录当前已读入行的数量（行数）

条件的表现形式：

正则表达式：/ / ~匹配 !~不匹配

数值/字符串比较：= = != > >= < <=

逻辑比较：&& ||

运算符：+ - \* / % ++ -- += \*= /=

高级应用

分支结构

单分支：if(条件){编辑指令}

双分支：if(条件){编辑指令}

多分支：if(条件){编辑指令1}else if(条件){编辑指令2}....else{编辑指令N}

数组：

定义数组：

格式：数组名[下标]=元素值

调用数组：

格式：数组名[下标]

遍历数组：

for(变量名 in 数组名){print 数组名[变量]}

1. netstat查看监听端口

-t显示tcp连接的端口

-u显示udp连接的端口

-n以数字格式显示端口号

-l显示服务正在监听的端口信息

-a查看所有端口信息（包括未启动的）

-p显示监听端口的服务名称是什么（也就是程序名称）

1. uptime查看cpu负载
2. free查看内存信息
3. who登录用户数量
4. nginx web服务器

软件安装：yum -y install gcc pcre-devel openssl-devel

useradd nginx

tar -zxvf nginx-1.10.3.tar.gz

cd nginx-1.10.3

./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http\_ssl\_module --with-stream

make && make install

启动服务：/usr/local/nginx/sbin/nginx

关闭服务：/usr/local/nginx/sbin/nginx -s stop

重载配置文件：/usr/local/nginx/sbin/nginx -s reload

查看软件信息：/usr/local/nginx/sbin/nginx -V

快捷方式：ln -s /usr/local/nginx/sbin/nginx /sbin

nginx平滑升级：

tar -zxvf nginx-1.12.2.tar.gz

cd nginx-1.12.2

./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http\_ssl\_module

make

cd /usr/local/nginx/sbin

mv nginx nginxold

cd /root/inmp\_soft/nginx-1.12.2/objs

cp nginx /usr/local/nginx/sbin

make upgrade

./nginx -V（查看软件版本）

用户认证：

配置文件：auth\_basic “auth-domain”;

auth\_basic\_user\_file /usr.local/nginx/pass

yum -y install httpd-tools

htpasswd -c（追加的时候不要加c，仅第一次加c，如果都加c，第二次覆盖第一次，每次密码系统会参杂一些随机密钥） /usr/local/nginx/pass admin

虚拟主机：基于域名server\_name [www.aaa.com](http://www.aaa.com)

基于端口listen 8000

基于ip地址listen 192.168.0.1:80

加密https：

配置文件：ssl on;

ssl\_certificate cert.pem;（指定证书文件，也就是公钥）

ssl\_certificate\_key cert.key;（指定私钥）

生成密钥：openssl genrsa > cert.key 2048（生成私钥）

openssl req -new -x509 -key cert.key > cert.pem（生成公钥）

隐藏版本号：配置文件：server\_tokens off;

rewrite地址重写

基本语法：rewrite 旧地址 新地址 选项

if(条件){}

示例：rewrite /a.html /b.html访问路径更改

rewrite ^/ <http://www.tmooc.cn域名跳转>

rewrite ^/ (.\*) <http://www.tmooc.cn/$1;> 只域名变，后头的不变

if ( $http\_user\_agent ~\* firefox ) {

rewrite /(.\*) /firefox/$1;

}不同浏览器显示的页面不一样

选项：

last：停止执行其他重写规则，根据URI继续搜索其他location，地址栏不变（匹配即停止，不再读其他rewrite）

break：停止其他的重写规则，完成本次请求（不再读其他语句，结束请求，location之类的也不执行了）

redirect：302临时重定向，地址栏改变，不更新URI（地址栏会变，临时）

permanent：301永久重定向，地址栏改变，更新URI（地址栏会变，永久）

正则表达式

区分大小写匹配：~

不区分大小写匹配：~\*

区分大小写不匹配：!~

不区分大小写不匹配：!~\*

反向代理

http：

http{

upstream sergrp{

#ip\_hash;

server 192.168.8.5:80;

server 192.168.8.6:80;

}

server{

listen 80;

server\_name [www.aaa.com;](http://www.aaa.com;)

location / {

#root html;

proxy\_pass http://sergrp;

}

}

}

tcp/udp：

使用--with-stream开启该功能（从1.9才开启该功能）

stream{

upstream backend {

server 192.168.2.100:22;

server 192.168.2.200:33;

}

server {

listen 12345;

proxy\_pass backend;

}

}

优化：

多并发（调整进程数量）

nginx.conf

worker\_processes 2;（与CPU核心数量一致）

worker\_connections 655356;（每个worker最大并发连接数）

修改系统ulimit限制：

ulimit -Hn 100000（临时）

ulimit -Sn 100000（临时）

vim /etc/security/limits.conf

\* soft nofile 100000

\* hard nofile 100000

包头缓存不够（414）

client\_header\_buffer\_size 1k;（默认请求包头信息缓存）

large\_client\_header\_buffrts 4 4k;（大请求包头信息的缓存个数与容量）

客户端缓存部分数据（只适合静态数据、图片、视频、音频）

http{

location ~\* \.(jpg|jpeg|png|css|js|ico|xml)$ {

expires 30d;（缓存30天）

}

}

压缩

gzip on;（开启压缩）

gzip\_min\_length 1000;（小文件不压缩，单位是字节，多媒体文件不要压缩，压缩适用于文档）

gzip\_comp\_level 4;（压缩比率，有1-9几个等级，越大越好）

gzip\_types text/plain text/css application/json;（对特定文件压缩，类型参考minme.type）

处理大量静态文件，需要保持这些文件句柄为打开状态，避免后续再打开（打开服务器的文件缓存）

http{

open\_file\_cache max=2000 inactive=20s;（设置服务器最大缓存2000个文件句柄，关闭2-秒内无请求的文件句柄）

open\_file\_cache\_valid 60s;（文件句柄的有效时间是60秒，60秒后过期）

open\_file\_cache\_min\_uses 5;（只有访问次数超过5次会被缓存）

open\_file\_cache\_errors off;

}

如何返回错误页面

error\_page 404 /40x.html;（自定义错误页面，取消注释）

修改行馆页面文件

nginx状态页面

status模块

--with-http\_stub\_status\_modules开启该模块

修改配置文件

location /status{

stub\_status on;

}

1. LNMP

安装nginx：

tar -zxvf nginx-1.12.2.tar.gz

cd nginx-1.12.2

./configure --prefix=/data/nginx --with-http\_ssl\_module

make && make install

安装mariadb

yum -y install mariadb-server mariadb mariadb-devel

安装php

yum -y install php

yum -y installphp-fpm-5.4.16-42.el7.x86\_64.rpm

安装php扩展

yum -y install php-mysql

启动服务

/data/nginx/sbin/nginx

netstat -utnalp | grep :80

systemctl start php-fpm

systemctl status php-fpm

netstat -utnalp | grep :9000

nginx+fastcgi

修改php配置文件

vim /etc/php-fpm.d/www/conf

pm.max\_childdren=50最大进程数量

pm.start\_servers=15最小进程数量

pm.min\_spare\_servers=5最小需要几个空闲着的进程

pm.max\_spare\_servers=35最多允许几个进程处于空闲状态

修改nginx配置文件

location ~\.php${

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf

}

1. ab压力测试工具

ab -c 并发数 -n 总请求数 URL

1. memcached非关系型数据库（高性能的分布式缓存服务器）

部署memcached

安装软件

yum -y install memcached

rpm -qa memcached

启动memcached

systemctl start memcached

php+memcached（yum安装）

部署LNMP

安装php的memcached扩展

yum install php-pecl-memcache

rpm -ql php-pcel-memcache

systemctl restart php-fpm

php+memcached（编译安装）

wget <http://ftp.gnu.org/gnu/m4/m4-1.4.9.tar.gz>

tar -zvxf m4-1.4.9.tar.gz

cd m4-1.4.9/

./configure && make && make install

cd ../..

yum -y install autoconf

yum -y install gcc-c++

cd libmemcached-1.0.18

./configure --prefix=/data/libmemcached --enable-sasl

make && make install

cd ..

wget <http://pecl.php.net/get/memcached-2.1.0.tgz>

tar -xf memcached-2.1.0.tgz

cd memcached-2.1.0

/data/php/bin/phpize

./configure --enable-memcached --with-php-config=/data/php/bin/php-config --with-zlib-dir --with-libmemcached-dir=/data/libmemcached --prefix=/data/phpmemcached

make && make install

ls /data/php/lib/php/extensions/no-debug-non-zts-20131226/

#vim /data/php/etc/php.ini

#[Session]

#extension = memcached.so

#session.save\_handler = memcached

#session.save\_path = "54.238.130.196:11211"

/etc/init.d/php-fpm restart

/data/nginx/sbin/nginx -s reload

setenforce 0

session共享

session：存储在服务器端，保存用户名、密码等信息

cookies：服务端发送给客户端，保存在客户端的一个文件里。保存的文件主要包括：sessionid 账户名 过期时间 路径和域

本地session

部署Nginx调度器

安装nginx软件

yum -y install pcre pcre-devel openssl-devel

tar -zxvf nginx-1.12.2.tar.gz

./configure

make && make install

安装memcached软件

yum -y install memcached

修改nginx配置文件并启动

upstream webs{

server 192.168.2.100:80;

server 192.168.2.200:80;

}

server{

location / {

proxy\_pass http://webs;

}

}

nginx -s reload

部署后端服务器

nginx:

yum -y install pcre pcre-devel openssl-devel

tar -zxvf nginx-1.12.2.tar.gz

./configure

make && make install

mariadb

yum -y install mariadb mariadb-server mariadb-devel

php

yum -y insatll php php-mysql php-pcre-memcache

启动服务

nginx

systemctl restart php-fpm

systemctl restart mariadb

部署测试页面

客户端访问

验证session

ls /var/lib/php/session

共享session

修改php文件

vim /etc/php-fpm.d/www.conf

修改：

php\_value[session.save\_handler]=memcache

php\_value[session.save\_path]=”tcp://192.168.4.5:11211”

重启服务：

systemctl restart php-fpm

1. tomcat服务器安装

安装jdk1.8

yum -y install java-1.8.0-openjdk

yum -y install java-1.8.0-openjdk-headless

安装部署tomcat

tar -xf apache-tomcat-8.0.30.tar.gz

mv apache-tomcat-8.0.30 /data/tomcat

启动tomcat

/data/tomcat/bin/startup.sh

测试服务器localhost:8080

查看端口，如果启动8005端口较慢，可使用以下命令（非必须）

mv /dev/random /dev/random.bak

ln -s /dev/urandom /dev/random

基于域名的虚拟主机

修改主配置文件，添加Host主机

<Host name=”www.a.com” appBase=”a” unpackWARs=”true” autoDeploy=”true”>

</Host>

mkdir -p /data/tomcat/{a,b}/ROOT

echo a > /data/tomcat/a/ROOT/index.html

echo b > /data/tomcat/b/ROOT/index.html

appBase是项目位置

添加context，修改默认首页位置（默认为ROOT,修改为base）

<Host name=”www.a.com” appBase=”a” unpackWARs=”true” autoDeploy=”true”>

<Context path=”” docBase=”base” reloadable=”true” />

</Host>

mkdir -p /data/tomcat/a/base

echo base > /data/tomcat/a/base/index.html

添加context，修改目录

<Host name=”www.a.com” appBase=”a” unpackWARs=”true” autoDeploy=”true”>

<Context path=”/test” docBase=”/var/ww/html” reloadable=”true” />

</Host>

echo test > /var/ww/html/index.html

ssl加密站点

生成私钥证书文件（工具是安装上jdk以后就有的）

keytool -genkeypair -alias tomcat -keyalg RSA -keystore /data/tomcat/keystore

修改主配置文件

<Connector port="8443"

protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true"

keystoreFile="/data/tomcat/keystore" keystorePass="123456" clientAuth="false" sslProtocol="TLS" />

开启日志功能（在host中加）

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="ljh\_log" suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

tomcat集群（nginx反向代理tomcat，如果代理https需要开启nginx的443端口，并配置密钥）

修改Nginx配置文件

http{

upstream toms {

server 192.168.2.100:8080;

server 192.168.2.200:8080;

}

server {

listen80;

server\_name localhost;

locatin / {

proxy\_pass http://toms;

}

}

}

1. varnish代理服务器（具有缓存功能，用内存缓存，也可以指定文件，反向代理服务器，加速web服务）

编译安装

yum -y install gcc readline-devel

yum -y install ncurses-devel pcre-devel

yum -y install python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm

useradd -s /sbin/nologin varnish

tar -xf varnish-5.2.1.tar.gz

cd varnish-5.2.1

./configure && make && make install

cp etc/examle.vcl /usr/local/etc/default.vcl（复制启动脚本和配置文件）

快速部署加速服务器

修改配置文件：vim /usr/local/etc/default.vcl

backend default {

.host=”192.168.2.100”;

.port=”80”;

}

启动服务：varnishd -f /usr/local/etc/default.vcl

其他选项：varnishd -s malloc,128M //定义使用内存作为缓存，空间为128M

varishd -s file,/var/lib/varnish\_storage.bin,1G//定义使用文件作为缓存

清除缓存内容的命令格式：varnishadm ban.url index.html

varnishadm ban req.url ~ .\*

查看缓存清理列表：varnishadm ban.lst

varnish日志

varnishlog //详细信息

varishncsa//类apache日志

varnish状态

varnishstat

client connections accepted:表示客户端成功发送连接的总数量

client requests received:客户端发送HTTP请求的总数量

cache hits:命中缓存的次数

cache misses:缓存非命中的个数

代理多台后端服务器

修改配置文件

vcl 4.0

import directors;//新增一行，导入一个directors（管理）

backend web {

.host = ”192.168.2.100”

.port = “80”;

}

backend web2 {

.host = ”192.168.2.200”;

.port = “80”;

}

-------接上面代码接着定义调度算法及制定流量转发--------

sub vcl\_init {#在init子函数中定义

new bar = directors.round\_robin();//random（随机） round\_robin dns

#定义服务器组，让新的bar等于之前定义的变量directors，后面接轮询算法

bar.add\_backend(web1);

bar.add\_backend(web2);

}

sub vcl\_recv {

set req.backend\_hint = bar.backend();//将流量转发给所有节点 注意括号

}

1. svn版本控制

服务端：

安装svn

yum -y install subversion

rpm -q subversion

创建版本库

mkdir /vvar/svn

svnadmin create /var/svn/project

导入项目代码

cd /usr/lib/systemd/system

svn import ./ [file:///var/svn/project](file://var\\svn\\project) -m “init”

认证与授权

vim /var/svn/project/conf/svnserve.conf passwd authz

启动服务

svnserve -d 共享所有仓库

svnserve -d -r /var/svn/project共享单个仓库

客户端：

checkout URL [path]从服务器复制一份副本到本地

svn ci -m “node”提交代码

查看版本库信息

svn info svn://192.168.2.100查看版本仓库信息

svn log svn://192.168.2.100查看版本修改历史

添加新文件（add）

echo “test” > test.sh

svn add test.sh

svn ci -m “add”

删除文件（del/rm）

svn rm test.sh

svn ci -m “a”

新建目录（mkdir）

svn mkdir test

svn ci -m “mkdir”

数据对比（diff）

echo “add” > killall

svn diff killall对比单个文件

svn diff对比所有文件

查看内容（cat）

svn cat url+filename

版本回滚（revert、merge）

revert：本地副本修改后，但没有提交时回滚数据

merge：本地副本修改并提交后，使用该命令回滚

svn revert killall

svn merge -r 10:5 netfs将nefs从10版本还原回5版本

1. git版本控制

git注册账号，新建仓库

命令行的使用：yum -y install git

git clone <https://github.com/账户名、仓库（下载）>

修改文件并提交：git commit -m “xx”;git push

新建文件：git add .;git commit -m “xx”;git push

1. rpm打包

流程：准备源码软件

安装rpm-build

编写编译配置文件

编译RPM包

yum - y install rpm-build

rpmbuild -ba test.spec

cp nginx.tar.gz rpmbuild/sources

vim /root/rpmbuild/specs/nginx.spec

rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec

rpm -ivh xxx.rpm

rpm -qpi xxx.rpm