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...

连同父目录一起创建

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

vimdiff filename filename1对比文件不同，并高亮显示

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

rpm -Uvh 软件名：升级安装软件

rpm2cpio rpm安包名 | cpio -div：提取rpm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

nmcli connection modify eth0 ipv4.gateway ip地址 添加网关

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判断为本域邮件（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;

}

删除不需要的模块（需要的模块用--with加载，不需要的用--without禁用）

./configure --without-http\_autoindex.module

make && make install

修改版本信息

修改版本信息（修改源码）

vim src/http/ngx\_http\_header\_filter\_module.c

static u\_char ngx\_http\_server\_string[] = “Server: aaa” CRLF;

static u\_char ngx\_http\_server\_full\_string[] = “Server: aaa” CRLF;

static u\_char ngx\_http\_server\_build\_string[] = “Server: aaa” CRLF;

隐藏版本信息（server\_tokens off）

vim /data/nginx/conf/nginx.conf

http {

server\_tokens off;

}

限制并发（该模块可以降低DDOS攻击风险）

vim /data/nginx/conf/nginx.conf

http {

limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;

//将客户端IP信息存储在名为one多大共享内存中，空间为10M，每秒种仅接受一个请求，多余的放入漏斗，处理完一个，则会从漏斗里面取出一个再处理

server {

limit\_req zone=one burst=5;

//漏斗超过5个则会报错和丢弃

}

}

拒绝非法请求

if ($request\_method !~ ^(GET|POST)$) {

return 444;

}

防止buffer溢出（防止客户端请求数据溢出，有效降低Dos攻击风险）

http {

client\_body\_buffer\_size 1k;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 16k;

large\_client\_header\_buffers 4 4k;

}

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. 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（openjdk）jdk1.8

yum -y install java-1.8.0-openjdk

yum -y install java-1.8.0-openjdk-headless

安装（javajdk）

下载jdk，解压

新建目录mkdir /usr/local/kencery

粘贴并重命名mv jdk1.8.0\_65 javajdk

配置环境变量vim /etc/profile

JAVA\_HOME=/usr/local/kencery/javajdk

PATH=$JAVA\_HOME/bin:$PATH

CLASSPATH=$JAVA\_HOME/jre/lib/ext:$JAVA\_HOME/lib/tools.jar

export PATH JAVA\_HOME CLASSPATH

立即生效source /etc/profile

检验java -version

安装部署tomcat（openjdk）

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

安装部署tomcat（javajdk）

将javajdk添加到tomcatc的配置文件中

vim /usr/local/kencery/tomcat/bin/catalina.sh

export JAVA\_HOME=/usr/local/kencery/javajdk #javajdk的安装路径，使 用echo $JAVA\_HOME命令可以读取

　　　　export CATALINA\_HOME=/usr/local/kencery/tomcat

　　　　export CATALINA\_BASE=/usr/local/kencery/tomcat

　　　　export CATALINA\_TMPDIR=/usr/local/kencery/tomcat/temp

设置开机启动

vim /etc/rc.d/rc.local

export JAVA\_HOME=/usr/local/kencery/javajdk

export CLASSPATH=.:$JAVA\_HOME/jre/lib/rt.jar:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

　　export PATH=$PATH:$JAVA\_HOME/bin

　　export CATALINA\_HOME=/usr/local/kencery/tomcat/

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;

}

}

}

隐藏版本信息

cd /data/tomcat/lib/

jar -xf catalina.jar

vim org/apache/catalina/util/ServerInfo.properties

server.info=xx/1.0.0.1

server.number=1.0.0.1

server.built=Mar 5 2018 09:34:35 UTC

vim /data/tomcat/conf/server.xml

<Connector port=”8080” protocol=”HTTP/1.1”

connectionTimeout=”2000”

redirectPort=”8443” server=”xxxx”（添加server字段）>

./shutdown.sh

./startup.sh

降权启动

使用非root启动Tomcat服务

useradd tomcat

chown -R tomcat:tomcat /data/tomcat

su -c /data/tomcat/bin/startup.sh tomcat

开机启动

chmod +x /etc/rc.local

su -c /data/tomcat/bin/startup.sh tomcat

删除默认启动页面

rm -rf /data/tomcat/webapps/\*

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

1. VPN服务器

GRE VPN（缺少加密机制）

加载模块：lsmod | grep ip\_grp //显示模块列表

modprobe ip\_grp //加载模版

modinfo ip\_grp //查看模块信息

创建VPN隧道

client

modprobe ip\_gre

ip tunnel add tun0 mode gre remote 201.1.2.5 local 201.1.2.10

ip link set tun0 up

ip addr add 10.10.10.10/24 peer 10.10.10.5/24 dev tun0

firewall-cmd --set-default-zone=trusted

vpn服务器

modeprobe ip\_gre

ip tunnel add tun0 mode gre remote 201.1.2.10 local 201.1.2.5

ip link set tun0 up

ip addr add 10.10.10.5/24 peer 10.10.10.10/24 dev tun0

echo “1” > /proc/sys/net/ipv4/ip\_forward

firewall-cmd --set-default-zone=trusted

测试联通性

client

ping 10.10.10.5

ping 192.168.4.5

服务器

ping 10.10.10.10

PPTP VPN（支持密码身份验证、支持MPPE加密）

安装软件

yum install pptpd-1.4.0-2.el7.x86\_64.rpm

rpm -qc pptpd

修改配置文件

vim /etc/pptpd.conf

localip 201.1.2.5 //服务器本地IP

remoteip 192.168.3.1-50 //分配给客户端的IP池

vim /etc/ppp/options.pptpd

require=mppe-128 //使用MPPE加密数据（帐号）

ms-dns 8.8.8.8 //DNS服务器

vim /etc/ppp/chap-secrets

jacob \* 123456 \*

echo “1” > /proc/sys/net/ipv4/ip\_forward

启动服务：systemctl start pptpd

翻墙设置：iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -j SNAT --to-source 201.1.2.5

iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.0/24 -j MASQUERADE

iptables -I FORDWARD -p tcp -n tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu

L2TP+IPSec VPN

安装软件

yum -y install libreswan

yum -y install xl2tpd-1.3.8-2.el7.x86\_64.rom

创建IPSec加密文件

vim /etc/ipsec.d/myipsec.conf

conn IDC-PSK-NAT

rightsubnet=vhost:%priv //允许的VPN虚拟网络

also=IDC-PSK-noNAT

conn IDC-PSK-noNAT

authby=secret //加密认证

ike=3des-sha1;modp1024 //算法

phase2alg=aes256-sha1;modp2048 //算法

pfs=no

auto=add

keyingtries=3

rekey=no

ikelifetime=8h

keylife=3h

type=transport

left=201.1.2.200 //服务器本机的外网IP

leftprotoport=17/1701

right=%any //允许任何客户端连接

leftprotoport=17/any

新建IPSec预定义共享密钥

vim /etc/ipsec.d/mpass.secrets

201.1.2.200 %any: PSK “123456”

启动IPSec服务（4500和500端口）

systemctl start ipsec

ss -ntulp | grep pluto

修改xl2tp配置文件

vim /etc/xl2tpd/xl2tpd.conf

ip range=192.168.3.128-192.168.3.254 //分配给客户端的IP池

local ip=201.1.2.200 //服务器IP地址

vim /etc/ppp/options.xl2tpd

require-mschap-v2 //添加一行

#crtscts //注释或删除该行

#lock //注释或删除该行

vim /etc/ppp/chap-secrets //修改密码文件

jacob \* 123456 \*

启动XL2TP服务（1701端口）

systemctl start xl2tpd

ss -ntulp | grep xl2tpd

1. NTP时间同步

服务器：

装包

yum -y install chrony

rpm -qc chrony

修改配置文件

主配置文件/etc/vhrony.conf

allow 192.168.4.0/24

deny 192.168.4.1

local stratum 10

启动服务器

systemctl restart chronyd

systemctl enable chronyd

客户端：

修改配置文件

vim /etc/chrony.conf

server 192.168.4.5 iburst

验证时间

设置时间

重启服务

查看时间

1. PSSH批量远程工具

安装软件rpm -ivh pssh-2.3.1-5.el7.noarch.rpm

常用选项：

-A：使用密码远程连接其他主机（默认使用密钥）

-i：将输出显示在屏幕

-H：设置需要连接的主机

-h：设置主机列表文件

-p：设置并发数量（几个线程）

-t：设置超时时间

-o dir：设置标准输出信息保存的目录

-e dir：设置错误信息保存的目录

-x：传递参数给ssh

pssh -i -A -H “192.168.1.4 192.168.1.5” -x ‘-o StrictHostKeyChecking=no’ echo hello使用密码连接多台主机并执行echo命令

pssh -i -A -h “host.txt” -x ‘ echo hello读取文件连接多台主机

pscp.pssh -h “/root/host.txt” /etc/hosts /tmp/将本地的文件拷贝到远程主机的/tmp目录下

pslurp -r -h host.txt /etc/tmp将远程主机的host.txt目录拷贝到/etc/tmp目录下

pnuke -h host.txt sleep杀死进程sleep

1. 账户安全

chage工具

chage -E yyyy-mm-dd username指定username失效日期

chage -l username查看username相关信息

账户锁定

passwd命令

-l锁定、-u解锁、-S查看状态

强制定期修改密码（修改配置文件）

vim /etc/login.defs

PASS\_MAX\_DAYS 99999密码最长有效

PASS\_MIN\_DAYS 0密码最短有效期（多长时间内可以再次修改）

PASS\_MIN\_LEN 8密码最短长度

PASS\_WARN\_AGE 7密码过期几天前警告

伪登录提示（修改相关配置文件）

/etc/issue /etc/issue.net

1. 文件和系统安全

程序和服务控制

禁用非必要的系统服务

锁定/解锁保护文件

chattr、lsattr

+ - =控制方式

属性i：所有都不可变，不能增删改

属性a：仅可以追加（只能echo，不能vim），不能改，删

1. 用户切换与提权

su切换用户身份

su - 目标用户

su - 目标用户 -c 指令

root切换其他用户不需要输入密码，普通用户切换其他用户，要输入对方密码

sudo提升执行权限

sudo 特权命令

sudo [-u 目标用户] 特权命令

配置sudo授权：visudo、vim /etc/sudoers

授权记录格式：用户名 主机列表=身份 命令

sudo别名设置：Cmnd\_Alias SERVICES=/sbin/service（别名必须大小）

ljh ALL=(ALL) SERVICE

添加日志文件：Defaults logfile=”/var/log/sudo”

1. SSH访问控制

改配置文件（/etc/ssh/sshd\_config）

port 3389//改用非标准端口

protocol2//启用SSH V2协议

permitrootlogin no//禁止root登录

usedns no//不解析客户机地址

logingracetime 1m//登录限时

maxauthtries 3//每连接最多认证次数

ssh黑/白名单设置

DenyUsers user1 user2

AllowGroups group1 group2

AllowUsers user1@host

ssh密钥对验证

创建密钥对：ssh-keygen

部署公钥：ssh-copy-id 3.1.221.97

测试：ssh 3.1.221.97

1. selinux安全防护

模式临时切换：setenforce 0切换为permissice

setenforce 1切换为enforceing

永久切换：修改配置文件（/etc/selinux/config）

修改安全上下文：chcon --reference=a b设置b和a一样（a为标准）

restorecon b重置b，自动继承上级目录标签

chcon -t pubic\_content\_t b单独设置标签

调整布尔值：getsebool -a查看所有布尔值

setsebool -P allow\_ftpd\_full\_access=1永久更改布尔值

1. md5sum校验工具

md5sum 文件名 生成md5码

1. GPG加密工具

对称加密：

gpg -c love.txt（-c加密）

gpg -d love.txt.gpg > love.txt（解密）

非对称加密

userb创建密钥对：gpg --gpg-key

gpg --list-keys查看密钥

userb导出公钥：gpg -a --export niuben > niuben.pub（--export的作用是导出公钥，-a的作用是导出的公钥存储为ASCII格式，否则可能是乱码）

scp niuben.pub [root@116.196.101.172:/root发送公钥到服务器](mailto:root@116.196.101.172:/root发送公钥到服务器)

usera服务器导入公钥：gpg --import niuben.pub

gpg --list-keys

usera对文件进行加密：gpg -e -r niuben love.txt（-e是加密操作，-r指定使用哪个密钥）

scp love.txt.gpg 132.232.56.166:/root传送加密文件

userb进行解密：gpg -d love.txt.gpg > love

GPG软件签名与验证（私钥做签名）

--default-key 密钥（指定密钥进行签名）

gpg -b log.tar对此文件进行签名

scp log.tar.\*（log.tar.sig和log.tar） [root@116.196.101.172:/root将签名文件和签名传给对方](mailto:root@116.196.101.172:/root将签名文件和签名传给对方)

gpg --verify log.tar.sig log.tar验证签名

1. AIDE入侵检测系统

yum -y install aide安装相关软件包

修改配置文件：/etc/aide.conf

初始化检查（在被入侵前）：aide --init

备份数据库（cp命令）：可以移动到U盘等设备

执行入侵后检查：mv aide.db.new.gz aide.db.gz

aide --check

1. nmap扫描

nmap [扫描类型] [选项] <扫描目标...>

常用扫描类型：

-sS，TCP SYN扫描（半开，不建议使用）

-sT，TCP连接扫描（全开）

-sU，UDP扫描

-sP，ICMP扫描

-A，目标系统全面分析

nmap -sP 116.196.101.172

1. tcpdump网络抓包工具

tcpdump [选项][过滤条件]

常见监控选项

1. 指定监控的网络接口
2. 转换为ACSII码，方便阅读

w，将数据包信息保存到指定文件

r，从指定文件读取数据包信息

tcpdump过滤条件

类型：host、net、port、portrange

方向：src、dst

协议：tcp、udp、ip、wlan、arp、.....

多个条件组合：and、or、not

tcpdump -A host 172.27.0.4 and tcp 21抓取172.27.0.4且端口是21的包

tcpdump -w ftp.log -A host 172.27.0.4 and tcp 21将抓取的数据保存到ftp.log中

tcpdump -A -r gtp.log | egrep “(USER|PASS)”查看并筛选内容

http需要再加以下两步：

authorization:basic dG9t0jEyMzQ1Ng==（需要在抓包记录中找这行）

echo dG9t0jEyMzQ1Ng== | base64 -d翻译密码

1. wireshark协议分析器

需要的包：wireshark、wireshark-gnome

1. curl

curl [选项] url

常用选项：

什么也不加，默认显示页面内容

-I：只显示头部信息

-i：显示头部和网页内容

-X：指定请求办法

-A：修改返回头信息

1. 审计audit

yum -y install audit

主配置文件/etc/audit/audit.conf

配置审计（临时）

auditctl命令

auditctl命令控制审计系统并设置规则决定哪些行为会被记录日志

auditctl -s //查询状态

auditctl -l //查看规则

auditctl -D //删除所有规则

定义文件系统，语法格式如下：

auditctl -w path -p 权限 -k key\_name

auditctl -w /etc/passwd -p wa passwd\_change

权限可以是r,w,x,a（文件或目录的属性发生变化）

key\_name为可选项，方便识别哪些规则生成特定的日志项

定义永久规则

修改配置文件/etc/audit/rules.d/audit.rules

-w /etc/sshd\_config -p rwxa -k sshd\_config

查看日志

tailf /var/log/audit/audit.log

搜索日志（默认程序会搜索/var/log/audit/audit.log）

ausearch options -if filename可以指定文件名

1. 数据库安全

初始化安全脚本mysql\_secure\_installation（前提已经启动服务，按步骤操作）

密码安全

修改mysql密码的若干方法

mysqladmin -uroot -p123456 password ‘mysql’

mysql -uroot -p123456

set password for root@’localhost’=password(‘redhat’)

清理历史（不显示明文密码）：.mysql\_history、.bash\_history、binlog（mysql目录里，5.6版本后已修复）

数据库的备份与恢复

备份

mysqldump -uroot -predhat mydb table > table.sql

mysqldump -uroot -predhat mydb > mydb.sql

mysqldump -uroot -predhat --all-databases > all.sql

恢复

mysql -uroot -predhat mydb < table.sql //还原表

mysql -uroot -predhat mydb < mydb.sql //还原数据库

mysql -uroot -predhat <all.sql //还原所有

数据安全：取消远程用户

1. linux打补丁（diff生成补丁包，patch打补丁）

diff：

diff 选项 文件或目录1 文件或目录2

选项：

-u 对比差异，怎么从第一个文件变为第二个文件（打补丁使用）

-r 递归对比目录中的所有资源（可以对比目录）

-a 所有文件均视为文本（包括二进制程序）

-N 无文件视为空文件（空文件怎么变成第二个文件）

//A 目录下没有txt文件，B目录下有txt文件

//diff 比较两个目录时，默认会提示txt仅在B目录有（无法根据补丁修复A缺失的文件）

//diff比较时使用N选项时，则diff会拿B下的txt与A下的空文件对比

//补丁信息会明确说明空文件修改后变成txt文件，打补丁即可成功

diff文件对比

diff -u test1.sh test2.sh

diff目录对比

准备环境

mkdir {source1,source2}

echo “hello world” > source1/test.sh

echo “hello the world” > source2/test.sh

echo “test” > source2/tmp.txt

cp /bin/find source1/

cp /bin/find source2/

echo “1” >> source2/find

对比差异

diff -u source1 source2//仅对比文本文件test.sh;二进制文件、tmp都没有对比差异，仅提示

diff -Nu source1/ source2/ //对比了test.sh，并且使用source2目录的tmp.txt与source1的空文件对比差异

diff -Nua source1 source2 //对比了test.sh、tmp.txt、find（程序）

diff -u test1.sh test2.sh > test.patch生成补丁文件

patch

diff -u test1.sh test2.sh > test.patch生成补丁文件

yum -y install patch

patch -p0 < test.patch打补丁

patch -RE <test.patch//还原旧版本（反向修复）

//patch -pnum指定删除补丁文件中多少层路径前缀

//如原始路径为/u/howard/src/blufl/blurfl.c

//-p0则整个路径不变

//-p1则修改路径为howard/src/blufl/blurfl.c

//-p4则修改路径为blurfl.c

//-R(reverse)反向修复，-E修复后如果文件为空，则删除该文件

1. iptables防火墙（四表五链：四表：filter（过滤信息）、nat（地址转换）、mangle（对进来的数据包修改）、raw（跟踪数据包），五链：OUTPUT出站规则、INPUT入站规则、FORWORD转发规则、PREROUTING路由前规则、POSTROUTING路由后规则：地址转换时用）

安装：

yum -y install iptables

systemctl start iptables

基本用法：

iptables [-t 表名] 选项（增删改查） [链名] [条件] [-j 怎么处理]

iptables -t filter -I INPUT -p icmp -j REJECT

注意：

可以不指定表，默认为filter表

可以不指定链，默认为对应表的所有链

除非设置默认策略，否则必须指定匹配条件

选项/链名/目标操作用大写字母，其余都用小写

基本的目标操作

ACCEPT：允许通过/放行

DROP：直接丢弃

REJECT：拒接通过，必要时会给出提示

LOG：记录日志，然后传给下一条规则（匹配即停止的唯一例外，会记录在/var/log/message）

常用的管理选项

添加规则：-A 在链的末尾追加一条规则

-I 在链的开头（或者指定序号）插入一条规则

查看规则：-L 列出所有的规则条目

-n 以数字形式显示地址、端口等信息

--link-numbers 查看规则时，显示规则的序号

删除规则：-D 删除链内指定序号（或内容）的一条规则

-F 清空所有规则

默认策略：-P 为指定的链设置默认规则

基本匹配策略

通用匹配规则：协议匹配 -p 协议名

地址匹配 -s 源地址、-d 目的地址

接口匹配 -i 收数据的入站网卡、-o 发数据的出站网卡

隐含匹配：端口匹配 --sport 源端口、--dport 目标端口

icmp类型匹配 --icmp-type ICMP类型

需要取反时，用！感叹号

iptables -t filter -AA INPUT -p tcp -j ACCEPT

iptables -I INPUT -p udp -j ACCEPT

iptables -I INPUT 2 -p icmp --icmp-type echo-request -j DROP

iptables -I INPUT 2 -p icmp ! --icmp-type echo-request -j DROP

iptables -nL INPUT

iptables -L INPUT --line-numbers

iptables -D INPUT 3

iptables -F

iptables -t filter -P INPUT DROP（设置默认策略，只能设置DROP或者ACCEPT）

iptables -A INPUT -s 192.168.4.120 -j drop

开启内核的IP转发

作为网关、路由的必要条件

echo ‘net.ipv4.ip\_forward=1’ >> /etc/sysctl.conf（永久）

或者

echo 1 >/ proc/sys/net/ipv4/ip\_forward（临时）

iptables -A FORWARD -s 192.168.0.0/16 -p tcp --dport 80 -j DROP

扩展匹配

iptables 选项 链名称 -m 扩展模块 --具体扩展模块 -j 动作

常见扩展条件类型

状态匹配 -m stat --state 状态值

MAC地址匹配 -m mac --mac-source MAC地址

多端口匹配 -m multiport --sport 源端口列表

-m multiport --dport 目标端口列表

IP范围匹配 -m iprange --src-range ip1-ip2

-m iprange --dst-range ip1-ip2

iptables -A INPUT -m mac --mac-source 00:0c:29:74:be:21 -j DROP

iptables -A INPUT -p tcp -m multiport --dports 20:22,80 -j ACCEPT（开放端口，连续的端口用冒号隔开，不连续的用逗号隔开）

iptables -A INPUT -p tcp --dport 22 -m iprange --src-range 192.168.4.10-192.168.4.20 -j ACCEPT

SNAT策略应用

iptables -t nat -A POSTROUTING -s 192.168.4.0/24 -p tcp --dport 80 -j SNAT --to-source 192.168.2.5

地址伪装策略（适合家庭使用，外网IP不固定，MASQUERADE可以自动识别，适用于ADSL宽带拨号）

iptables -t nat -A POSTROUTING -s 192.168.4.0/24 -p tcp --dport 80 -j MASQUERADE --to-source 192.168.2.5

service iptables save（保存规则，不保存重启就会丢掉）

1. zabbix

安装前准备：安装LNMP并启动相关服务

服务端：

yum -y install net-snmp-devel curl-devel libvent-devel

tar -xf -zabbix-3.4.4.tar.gz

cd zabbix-3.4.4.tar.gz

./configure --enable-server --enable-proxy --enable-agent --with-mysql=/usr/local/mysql\_config --with-net-snmp --with-libcurl

make && make install

初始化zabbix

创建数据库与数据库账户

mysql

create database zabbix character set utf8;

grant all on zabbix.\* to zabbix@’localhost’ identified by ‘zabbix’;

mysql -uzabbix -pzabbix zabbix <schema.sql

mysql -uzabbix -pzabbix zabbix <images.sql

mysql -uzabbix -pzabbix zabbix <data.sql

上线zabbix页面

cd /zabbix-3.4.4/frontends/php

cp -a \* /data/nginx/html

chmod -R 777 /data/nginx/html/\*

修改配置文件，启动zabbix server服务

useradd zabbix

vim /usr/local/etc/zabbix\_server.conf

DBhost=localhost

DBname=zabbix

DBUser=zabbix

DBPassword=zabbix

LogFile=/var/log/zabbix/zabbix\_server.log

zabbix\_server

修改配置文件，启动zabbix agent（被监控时使用）

vim /usr/local/etc/zabbix\_agent.conf

server=127.0.0.1,192.168.2.5

serveractive=127.0.0.1,192.168.2.5

hostname=zabbix\_server

logfile=/tmp/zabbix\_server.log

unsafeuserparameters=1

zabbix\_agentd

客户端：

安装

useradd zabbix

yum -y install gcc pcre-devel

tar -xf zabbix-3.4.4.tar.gz

cd zabbix-3.4.4

./configure --enable-agent

make && make install

拷贝启动脚本（可选操作）

cd misc/init.d/fegora/core

cp zabbix\_agent /etc/init.d

修改agent配置文件并启动服务

vim /usr/local/etc/zabbix\_agent.conf

server=127.0.0.1,192.168.2.5

serveractive=127.0.0.1,192.168.2.5

hostname=zabbix\_client\_web1

enableremotecommands=1

unsafeuserparameters=1

zabbix\_agent //启动服务

基础监控

配置->添加主机->应用监控模版->查看数据（检测中->最新数据）

自定义监控

被监控项修改agent配置文件

vim /data/zabbix/etc/zabbix\_agent.conf

Include=/data/zabbix/etc/zabbix\_agentd.conf.d/

创建自定义key

cd /data/zabbix/etc/zabbix\_agentd.conf.d/

vim count.line.passwd

UserParameter=count.line.passwd,wc -l /etc/passwd | awk ‘{print $1}’

重启agent

killall zabbix\_agentd

zabbix\_agentd

测试自定义key是否有效

zabbix\_get -s 127.0.0.1 -k count.line.passwd

创建监控模版

登录web管理页面

选择配置（configuration），模版（templates），创建模版

创建应用

模版添加后，默认没有任何应用、项目、触发器、图形等

点击模版后面的应用集（application）、刷新出的页面中选择创建应用集（create application）

设置application name点击添加

创建监控项目

配置（configuration）->模版（templates）->监控项目（item）->创建（create item）

设置监控参数

项目名称

自定义key（必须与配置文件一致）

应用集选择刚刚创建的应用

创建图形

与创建监控项类似

将模版关联主机

configuration->hosts->选择主机

点击monitoring->craphs->选择条件查看图形

zabbix报警机制

触发器

创建触发器

通过configuration->templates

选择模版后面的triggers->create trigger

触发器表达式

expression 表达式：触发异常的条件

{<server>:<function>(<parameter>)}<operator><constant>

{主机：key.函数(参数)}<表达式>常数

配置触发器

设置触发器名称

填写表达式：item监控项、function函数、N值、time shift时间偏移量

讯泽恩触发器报警级别

add创建该触发器

设置邮件

创建media报警媒介

设置邮件服务器

adminisration->media type->选择email邮件

设置邮件服务器信息

为账户添加media

在administration->users中找到admin账户

选择media菜单->点击add添加报警媒介

在media type中填写报警类型，收件人，时间等信息

创建active（行为）

configuration->actions->create action创建

填写名称

配置导致动作的触发条件

配置动作的具体操作行为（发送信息或执行远程命令）

zabbix自动发现

自动发现流程

创建自动发现规则

创建action动作，说明发现主机后自动执行什么操作

通过动作，执行添加主机，连接模版到主机等操作

自动发现规则

创建自动发现规则：configuration->discovery->create discovery rule

填写规则

自动发现的IP范围（逗号隔开可以写多个，横杠表示连续的IP）

多久发现一次（默认为1小时）

检查的方式：http ftp agent的自定义key等检查

创建动作

configuration>actions

event source(discovery)

注意：选择事件源为：自动发现

添加动作名称，添加触发动作的条件

操作（触发动作后要执行的操作指令），操作细节如下

添加主机到组，与模版关联（HTTP模版）

创建新主机

创建一台新的主机，验证zabbix是否可以自动发现该主机

登录zabbix服务器的web页面，查看主机列表

zabbix主动监控

创建新的被监控主机，并安装agent软件

修改配置文件

#Server=127.0.0.1,172.25.0.10、、注释该行，允许谁监控本机

StartAgents=0//被动监控时启动多个进程，设置为0，则禁止被动监控

ServerActive=172.25.0.10//主动监控模式，一定要取消127.0.0.1

RefreshActiveChexks=120//默认120秒检测一次

UnsafeUserParameters=1//允许自定义key

Include/usr/local/etc/zabbix\_agentd.conf.d/

zabbix\_agentd

克隆模版

configuration->templates

选择template os linux

全克隆该末班，新建一个新的模版

新模版名称为：template os Linux serveractive

修改监控项模式

将模版中的所有监控项目全部修改为主动监控模式

configuration->templates

选择新克隆的模版点击后面的items（监控项）

点击全选，选择所有监控项，点击批量更新

将类型修改为：zabbix agent(active主动模式)

批量修改监控项的监控模式以后，并非所有监控项目都支持主动模式

批量修改后，会发现有几个没有修改主动模式成功

说明这些监控项目不支持主动模式，关闭即可

可以点击类型排序，方便操作，点击状态关闭即可关闭

添加监控主机

在zabbix监控服务器中，添加被监控的主机（主动模式）

名称与被监控端的配置文件hostname一致，将主机添加到linux servers组

IP地址为0.0.0.0，端口为0，不填写IP无法创建成功

为主机添加监控模版

选择刚刚创建的模版（主动模式）

添加链接模版到主机

验证效果

查看数据图表

monitoring->graphs

选择需要查看的主机组、主机以及图形

查看图表时无数据：因为分区数据采用的是自动发现监控，与普通监控一样，改为主动模式即可

拓扑图与聚合图形

拓扑图

绘制拓扑图可以快速了解服务器架构

monitoring->maps（拓扑图）

新建拓扑图然后进行编辑即可

操作说明

icon（图形），添加新的设备后可以点击图标修改属性

shape（形状）

Link（连线），先选择两个图标，再选择连线

完成后，点击update更新

聚合图形

在一个页面显示多个数据图表，方便了解多组数据

monitoring->screens（聚合图形）->create screen

owner：使用默认的admin用户

name：名称设置为web2\_host

columns：列数设置为2列

rows：行数设置为4行

选择刚刚创建的聚合图形

点击后面的购在函数

点击change（更改），设置每行每列需要显示的数据图表

1. mysql安装

关闭iptables

关闭selinux

从官网下载软件包

安装软件包

从官网下载RPM包

htttp://dev/mysql.com/downloads/mysql

选择适用于当前系统的bundle集合包

mysql-community-client//客户端应用程序

mysql-community-common//数据库和客户端共享文件

mysql-community-devel//客户端应用程序的库和头文件

mysql-community-embedded//嵌入式函数库

mysql-community-embedded-compay//嵌入式janrong函数库

mysql-community-embedded-devel//头文件和库文件作为mysql的嵌入式库文件

mysql-community-libs//mysql数据库客户端应用程序的共享库

mysql-community-libs-compat//客户端应用程序的共享兼容库

准备工作

停止mariadb服务：systemct stop mariadb

删除文件/etc/my.cnf：rm -rf /etc/my.cnf

删除数据：rm -rf /var/lib/mysql/\*

卸载软件包：rpm -e --nodeps mariadb-server mariadb

至少安装server、client、share\*包

采用U升级安装，可替代冲突文件

推荐将devel也装上，用于支持其他软件

yum -y install perl-Date-Dumper perl-JSON perl-Time-HiRes autoconf libaio

tar -xf mysql-5.7.17-1.el7.x86\_64.rpm-bundel.tar

rpm -Uvh-community-\*.rpm

启动mysql数据库服务

服务脚本：/usr/lib/system/mysqld.service

systemctl enable mysqld

systemctl start mysqld

systemctl status mysqld

1. redis安装

装包

从源码编译安装

tar -xf redis-4.0.8.tar.gz

cd redis-4.0.8

make && make install

初始化配置（初始化完成默认开启服务）

配置服务运行参数

端口：6379

主配置文件：/etc/redis/6379.conf

日志：/var/log/redis\_6379.log

数据库目录（内存往硬盘存储的目录）：/var/lib/redis/6379

pid文件：/var/run/redis\_6379.pid

启动程序：/usr/local/bin/redis-server

./utils/install\_server.sh//初始化

启动/停止服务

启动服务

/etc/init.d/redis\_<portnumber> start

停止服务

/etc/init.d/redis\_<portnumber> stop

连接数据库服务

访问redis服务

ps -C redis

netstat -utnlp | grep redis

redis-cli//连接本机的redis数据库服务

redis-cli -h ip地址 -p 端口//连接远程数据库服务

1. LNMP+redis

部署LNMP

安装软件包

安装源码nginx和php

yum -y install gcc gcc-c++ pcre-devel zlib-devel

tar -xf nginx-1.12.2.tar.gz

./configure --prefix=/data/nginx

make && make install

yum -y install php-common

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

修改配置nginx文件使其支持PHP

启动服务

nginx

systemctl start php-fpm

配置PHP支持redis

安装PHP扩展

yum -y install autoconf automake

yum -y install php-devel=5.4.16-46.el7.x86\_64.rpm

tar -xf phpredis-2.2.4.tar.gz

cd phpredis-2.2.4

/usr/bin/phpize

./configure --with-php-config=/usr/bin/php-config

make && make install

vim /etc/php.ini

extension\_dir=”扩展路径”

extension=”redis.so”

systemctl restart php-fpm

1. mongodb安装

装包

免安装，解压后即可使用

mkdir /usr/lcoal/mongodb

tar -xf mongodb-linux-x86\_64-rhel70-3.63.tgz

cp -r mongodb-linux-x86\_64-rhel70-3.63.tgz/bin /usr/local/mongodb

cd /usr/local/mongodb

mkdir -p etc log data/db

创建配置文件

手动创建服务主配置文件

vim mongodb.conf

logpath=/usr/local/mongodb/log/mongodb.log

logappend=true//追加的方式记录日志信息

dbpath=/usr/local/mongodb/data/db//数据库目录

fork=true//守护进程方式运行

启动服务

./bin/mongod -f /usr/local/mongodb/etc/mongodb.conf

查看进程

ps -C mongod

查看端口

netstat -yunlp | grep 27017

连接服务

本地连接，默认没有密码

/usr/local/mongodb/bin/mongo

指定服务的IP地址和端口号

停止服务

killall mongod

mongod --shutdown -f /usr/local/mongodb/etc/mongodb.conf

修改配置文件

vim /mongodb.conf

bind\_ip=ip地址

port=端口号

启动服务

./bin/mongod -f /usr/local/mongodb/etc/mongodb.conf

连接服务

mongo --host ip地址 --port 端口号

1. iscsi配置（提供块设备，例如硬盘、U盘等）

是一种基于IP Storage理论的新型存储技术

将存储行业广泛应用的SCSI接口技术与IP网络相结合

可以在IP网络上构建SAN

优势

基于IP协议技术的标准

允许网络在TCP/IP协议上传输SCSI命令

相对于FC SAN，iscsi实现的ip san投资更低

解决了传输效率、存储容量、兼容性、开放性、安全性等方面的问题

没有距离限制

操作流程

服务端为客户端提供块设备，需要起名，这个名字是IQN，IQN名称规范：iqn.时间.域名的反写.字符串,并分配大小格式化，然后客户端启用服务，发现硬盘，使用硬盘。

客户端

iscsi initiator：软件实现，成本低、性能低

iscsi hba：硬件实现，性能好，成本较高

服务端

iscsi target

部署iscsi服务

服务端：

添加额外硬盘

安装target软件

查询yum仓库

yum list | grep target

安装

yum -y install targetcli

为新硬盘分区

parted /dev/vdb

mklabel gpt

mkpart primary 1M 50%

mkpart primary 50% 100%

lsblk

查看iscsi target信息

yum info targetcli

配置iscsi target

定义后端存储（给vdb1起个名字formysql，加入到iscsi管理中）

targetcli

ls

backstores/block create formysql /dev/vdb1

创建iqn对象（创建供客户端访问的iqn名字）

/iscsi create iqn.2018-01.cn.tedu.nsd1803

绑定存储（把存储绑定到iqn名字中）

/iscsi/iqn.2018-01.cn.tedu.nsd1803/tpg1/luns create backstores/block/formysql

授权客户机访问（配置acl，限定允许访问的客户端）

/iscsi/iqn.2018-01.cn.tedu.nsd1803/tpg1/acls create iqn.2018-01.cn.tedu.node02

/iscsi/iqn.2018-01.cn.tedu.nsd1803/tpg1/acls create iqn.2018-01.cn.tedu.node0

绑定监听地址

/iscsi/iqn.2018-01.cn.tedu.nsd1803/tpg1/portals create 0.0.0.0

保存配置

saveconfig

exit

服务管理

控制服务

systemctl start target

设置开机启动

systemctl enable target

查看端口

netstat -tunlp | grep 3260

客户端（客户端node2访问并验证）

查询Yum仓库

yum list | grep initiator

安装

yum -y install iscsi-initator-utils

查看iscsi target信息

yum info iscsi-initator-utils

启动服务

systemctl start iscsi

设置本机的iqn名称

vim /etc/iscsi/initatorname.iscsi

InitatorName=iqn.2018-01.cn.tedu.node02

发现远程target存储，此时可以发现名字，但是不会存储端

iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.1 --discover

重启服务，登录iscsi设备，出现新硬盘

systemctl restart iscsi

lsblk

systemctl enable iscsi

systemctl enable iscsid

重启后，本地会有sda，第一个是重启服务，第二个是重启守护进程

分区，格式化，进行使用

parted /dev/sda

mklabel gpt

mkpart primary 1M 100%

lsblk

mkfs.ext4 /dev/sda1

安装mysql进行检测

yum -y install mariadb-server mariadb

mount /dev/sda1 /var/lib/mysql

chown -R mysql:mysql /var/lib/mysql

systemctl start mariadb

如果客户端node2宕掉，可以使用node3进行切换

安装软件包

yum -y install iscsi-initator-utils

systemctl start scsi

设置本机iqn名称

vim /etc/iscsi/initatorname.iscsi

InitatorName=iqn.2018-01.cn.tedu.node03

发现远程target存储，此时可以发现名字，但是不会出现存储端

iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.1 --discover

重启服务，登录iscsi设备，出现新硬盘

systemctl restart iscsi

lsblk

systemctl enable iscsi

systemctl enable iscsid

安装数据库，进行检验

yum -y install mariadb-server mariadb

mount /dev/sda1 /var/lib/mysql

systemctl restart mysql

mysql -uroot -p

>show databases

注意：千万不要使用两个节点同时挂载相同的文件系统（分区），如果多个节点同时挂在这个文件系统，很可能导致文件系统的损坏，数据丢失！因为XFS/EXT4/EXT3这些文件系统都是单节点文件系统。红帽的GFS才能支持多节点同时挂载。

1. udev配置

概述

只有连到系统上来的设备才在/dev下创建设备文件

与主、次设备编号无关

为设备提供持久、一致的名字

如，把U盘接入到主机，主机会多一个硬盘文件，把U盘移除时，会少一个硬盘文件

接入设备事件链

内核发现设备并导入设备状态到sysfs

udev接到事件通知

udev创建设备节点或是运行指定程序

udev通知hald守护进程

HAL探测设备信息

HAL创建设备对象结构

HAL通过系统消息总线广播该事件

用户程序也可以监控该事件

udev的作用

从内核收集到添加/移除硬件事件时，udev将会分析：

/sys目录下信息

/etc/udev/rules.d目录中的规则

基于分析结果，udev会

处理设备命名

决定要创建哪些设备文件或链接

决定如何设置属性

决定触发哪些事件

配置udev

主配置文件/etc/udev/udev.conf

udev\_root：创建设备文件位置，默认为/dev

udev\_rules：udev规则文件位置，默认为/etc/udev/rules.d

udev\_log：syslog优先级，默认为err

文件格式及位置

/etc/udev/rules.d/数字-<rule\_name>.rules

规则格式

<match-key><op><value>[,...]<assignment=key><op>value[,...]

操作符

= =：表示匹配

!=：表示不匹配

= =：表示匹配

!=：表示不匹配

udev变量

可以简化或缩写规则

KERNEL= =”sda\*”,SYMLINK+=”iscsi%n”

常量替代变量

%k：内核所识别出来的设备名，如sdb1

%n：设备的内核编号，如sda3中的3

%p：设备路径，如/sys/block/sdb/sdb1

%%：%符号本身

实例

应用udev，给iscsi创建一个软连接叫idisk

1. 查看iscsi磁盘（sda）在sys目录中的位置

udevadm info --query=path --name=/dev/sda1

1. 通过上一步的路径，查看iscsi(sda)磁盘的信息（以和其他信息进行区别）

udevadm info --query=all --attribute-walk --path=上一步的路径

1. 创建规则文件

vim /etc/udev/rules.d/90-iscsi.rules

KERNEL==”sd[a-z]\*”,ACTION==”add”,SUBSYSTEMS==”scsi”,ATTRS[model]==”formysql ”,SYMLINK+=”idisk%n”

注：KENNEL==”sd[a-z]\*”表示内核识别出来的设备名

ACTION==”add”表示新接入设备

SUBSYSTEMS==”scsi”和ATTRS是上一步查到的属性

SYMLINK表示符号链接，+=表示额外创建，%n是分区号

1. 只有把磁盘移除再接入才能生效或是重启系统
   1. iscsiadm --mode node --targetname iqn-2018-01.cn.tedu.nsd1803 --portal 192.168.4.1:3260 --logout 退出
   2. iscsiadm --mode discoverrydb --type sendtargets --portal 192.168.4.1 --discover 再次查找
   3. systemctl restart iscsi 重启服务
   4. ll /dev/idisk\* 查看是否成功
2. NFS网络文件系统

不管是NFS，还是SAMBA，还是FTP，只要是读写共享，必须注意本地权限和配置文件内的授权

配置并访问NFS共享（只读，读写共享）

主要软件包

nfs-utils-1.3.0-0.48.el7.x86\_64

rpcbind-0.2.0-42.el7.x86\_64

系统服务脚本

nfs、rpcbind

主配置文件

/etc/exports

/etc/exports配置解析

共享目录 客户机地址（参数,参数... ...）...

NFS选项

no\_root\_squash表示保留文件的属主属组还是root，默认写入到共享目录中的文件属主属组是nfsnobody（只对root有效）

all\_squash作用是客户端任何写入的文件属主属组都是nfsnbody

注意事项：1.从未授权的客户端将无法挂载此共享

2.若未启用no\_root\_squash，挂载后会无权限浏览（750）

3.sync、async同步写入、异步写入

只读共享

服务端

1. 安装软件包

yum -y install nfs-utils

1. 创建共享目录并拷贝文件

mkdir -p /nfsroot/nfsro

cp /etc/hosts /nfsroot/nfsro

1. 修改配置文件

vim /etc/wxports

/nfsroot/nfsro \*(ro)

1. 启动服务并验证

systemctl restart nfs

showmount -e 192.168.4.1

客户端

1. 创建挂载点

mkdir /mnt/nfshare

1. 查看共享挂载

showmount -e 192.168.4.1

mount 192.168.4.1:/nfsroot/nfsro /mnt/nfshare

1. 检查

ls /mnt/nfshare

读写共享

服务端

1. 服务器创建用于读写的共享目录

mkdir -m 777 /nfsroot/nfsrw

1. 编辑配置文件

vim /etc/exports

/nfsroot/nfsrw 192.168.4.\*(rw,sync)

1. 重新输出共享

exports -rv

1. 检查

showmount -e 192.168.4.1

客户端

1. 创建挂载点

mkdir /mnt/rwnfs

1. 挂载测试

showmount -e 192.168.4.1

mount 192..168.4.1:/nfsroot/nfsrw /mnt/rwnfs

1. Mulitipath多路径

多路径描述

当服务器到某一存储设备有多条路径时，每条路径都会识别为一个单独的设备

多路径允许您将服务器节点和存储阵列间的多个I/O路径配置为一个单一设备

这些I/O路径是可包含独立电缆、交换机和控制器的实体SAN连接

多路径集合了I/O路径，并生成由这些集合路径组成的新设备

主要功能

冗余

主备模式，高可用

改进模式

主主模式，负载均衡

多路径设备

若没有DM Multipath，从服务器节点到储存控制器的每一条路径都会被系统视为独立的设备，即使I/O路径连接的是相同的服务器节点到相同的储存控制器也是如此

DM Multipath提供了有逻辑的管理I/O路径的方法，即在基础设备顶端生成单一多路径设备

每个多路径设备都有一个WWID（全球识别符），它是全球唯一的，无法更改的好吗

默认情况下会将多路径设备的名称设定为它的WWID

可以在多路径配置文件中设置user\_friendly\_names选项，该选项可以将别名设为格式为mpathn的节点唯一名称

也可以自定义存储名称（合并出来的多路径设备，一般起名为mpath[a-z]）

配置案例：

准备共享存储

配置iscsi服务器

准备共享介质（分区、LV或磁盘镜像）

安装scsi-target-utils

准备规划iqn名称

修改target.conf配置文件，以提供存储

访问共享存储

因为达到共享存储有两条路径，所以需要在两条路径上都执行发现命令

iscsiadm --mode discovery --type sendtargets --portal 192.168.4.1 --discovery

iscsiadm --mode discovery --type sendtargets --portal 192.168.4.1 --discovery

设置开机自启

systemctl enable iscsi

systemctl enable iscsid

安装多路径软件包

yum -y install device-mapper-multpath

获取WWID

加入共享存储在本地被识别为/dev/sdb和/dev/sdc，那么获取它WWID的方法为

/lib/udev/scsi\_id --whitelisted --device=/dev/sdb

/lib/udev/scsi\_id --whitelisted --device=/dev/sdc

因为两个设备虽然名称不一样，但是实际上是一个设备，所以他们的WWID是相同的

使用mpathconf命令创建配置文件并启用多路径

mpathconf --user\_friendly\_names n

若无需编辑该配置文件，可使用此命令启动多路径守护程序

修改配置文件

在配置文件中声明获取WWID的方法

vim /etc/multipath.conf

defaluts{

user\_friendly\_names no

find\_multipaths yes

}

为多路径设备配置别名

根据得到的WWID，为多路径设备配置别名

multipaths{

multipath{

wwid “”

alias mpatha

}

}

启动服务并验证

启动服务

systemctl start multipathd

systemjctl enable multipathd

验证

ls /dev/mapper #mpatha即为多路径设备

multpath -rr #重新加载多路径信息

multpath -ll #查看多路径信息

分区

为/dev/mapper/mpatha分区，得到的第一个分区名为/dev/mapper/mpatha1

1. lvs调度器

介绍

LVS可以实现高可用、可伸缩的web、email、cache和media等网络服务

最终目的是利用linux操作系统和lvs集群软件实现一个高可行、高性能、低成本的服务器应用集群

组成

前端：负载均衡层

由一台或多台负载调度器构成

中间：服务器群组层

由一组实际运行应用程序的服务器组成

底层：数据共享存储

提供共享存储空间的存储区域

术语

director server：调度服务器

将负载分发到real server的服务器

real server：真实服务器

真正提供应用服务的服务器

VIP：虚拟IP地址

公布给用户访问的虚拟IP地址

RIP：真实IP地址

集群节点上使用的IP地址

DIP：调度器连接节点服务器的IP地址

工作模式

VS/NAT

通过网络地址转发实现的虚拟服务器

大并发访问时，调度器的性能成为瓶颈

VS/DR

直接使用路由技术实现虚拟服务器

节点服务器需要配置VIP，注意MAC地址广播

VS/TUN

通过隧道方式实现虚拟服务器

负载均衡调度算法（10种，常用前4种）

轮询（round robin）

将客户端请求平均分发到real server

加权轮询（weighted round robin）

根据real server权重值进行轮询调度

最少连接（least connections）

选择连接数最少的服务器

加权最少连接（weight least connections）

根据real server权重值，选择连接数最少的服务器

源地址散列（source hashing）

根据请求的目标IP地址，作为散列键从静态分配的散列表找出对应的服务器

其他调度算法

基于局部性的最少连接

带复制性的最少连接

目标地址散列（destination hashing）

最短的期望的延迟

最少队列调度

安装软件

两台真实服务器，可以运行web服务

一台服务器做调度服务器

LVS的ip负载是通过IPVS模块实现的

IPVS模块是Linux的组成部分

安装ipvsadm

yum -y install ipvsadm

ipvsadm用法

创建虚拟服务器

-A添加虚拟服务器

-t 设置群集地址（VIP地址，tcp连接，udp是-u）

-s指定负载调度算法

-D删除虚拟服务器

添加、删除服务器节点

-a添加真实服务器

-d删除真实服务器

-r指定真实服务器的地址

-m使用NAT模式;-g、-i扥别对应DR、TUN模式

-w为节点服务器设置权重，默认为1

查看IPVS

ipvsdm -LN

LVS-NAT集群

操作流程

lvs调度器接受到用户请求，根据调度算法分配给底下的服务器进行工作，然后返回给调度器，由调度器再返回给用户结果，调度器是双网卡

real server:

配置web服务器

配置网关为调度器地址

director server：

安装并启用ipvsadm

创建虚拟服务器

向虚拟服务器中加入节点

client：

连接虚拟机服务器测试

real server

配置web页面

配置网关为调度器的IP地址

nmtui

ifdown eth0;ifup eth0

部署LVS调度器

添加双网卡，并配置IP地址，因为一个是对外的，一个是real server网关

打开ip\_forword（rhel7默认已经打开）

vim /etc/sysctl.conf

net.ipv4.ip\_forward=1

sysctl -p

装包

yum -y install ipvsadm

创建虚拟服务器，VIP为201.1.1.4:80，采用调度算法为round robin

ipvsadm -A -t 201.1.1.4:80 -s rr

-A添加虚拟服务器，-t是tcp模式

向虚拟服务器中加入节点

ipvsadm -a -t 201.1.1.4:80 -r 192.168.4.2 -m

ipvsadm -a -t 201.1.1.4:80 -r 192.168.4.3 -m

-a加入真机，-r是真机节点IP，-m是nat模式，-w是设置权重

查看配置

ipvsadm -Ln

保存配置

ipvsadm-save > /etc/sysconfig/ipvsadm

删除配置

ipvsadm -d -t 201.1.1.4:80 -r 192.168.4.3

从虚拟服务器中删除RIP

ipvsadm -D -t 201.1.1.4:80

删除虚拟服务器

启动ipvsadm

systemctl start ipvsadm

systemctl enable ipvsadm

修改director调度算法为ERR

ipvsadm -E -t 201.1.1.4:80 -s wrr

client验证

通过浏览器访问

每秒运行ipvsadm -Ln

watch -n1 ipvsadm -Ln

LVS-DR集群

操作流程

调度器是单网卡地址，且需配虚拟IP地址，客户端发送请求到虚拟IP，调度器响应请求，发送给真实服务器，服务器响应请求，使用虚拟IP地址回应客户端，所以真实服务器需要配置VIP到回环地址，且更改内核参数，不响应用户请求，只有调度器响应

real server

配置web服务器

配置辅助IP地址，调整内核参数

director server

安装并启用invsadm

配置虚拟服务器，向虚拟服务器中加入节点

client

连接虚拟服务器测试

配置real server服务器

配置web服务器

配置虚拟ip地址

cp /etc/sysconfig/network-scripts/ifcfg-lo{,:0}

vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.100

NETMASK=255.255.255.255

NSETWORK=192.168.4.100

ONBOOT=yes

NAME=lo:0

ifup lo:0

调整内核参数

echo 1 > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/lo/arp\_announce

echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp\_announce

sysctl -p

配置LVS调度器

配置虚拟IP地址

cp /etc/sysconfig/network-scripts/ifcfg-eth0{,:0}

vim ifcfg-eth0:0

DEVICE=eth0:0

BOOTPROTO=none

ONBOOT=yes

TYPE=Ethernet

NAME=eth0:0

IPADDR=192.168.4.100

PREFIX=24

ifup eth0:0

创建虚拟服务器

ipsdam -A -t 192.168.4.100:80 -s wlc

向服务器中添加节点

ipsvadm -a -t 192.168.4.100:80 -g -r 192.168.4.1 -w 1

ipsvadm -a -t 192.168.4.100:80 -g -r 192.168.4.2 -w 2

client端验证

通过浏览器访问

配置永久规则

新建/etc/sysconfig/ipvsadm文件，并写入内容，重启服务即可

1. keepalived（高可用）热备

概述

解决调度器出现单点故障的问题，实现了高可用集群

最初为LVS设计的，专门监控各服务器节点的状态

后来加入了VRRP（虚拟冗余路由协议，IETF公共标准）功能，放置单点故障

运行原理

keepalived检测每个服务器节点状态

服务器节点异常或工作出现故障，keepalived将故障节点从集群系统中移除

故障节点恢复后，keepalived再将其加入到集群系统中

所有工作自动完成，无需人工干预

高可用web拓扑

多台web服务器。由于没有过多用户访问，只有一台服务器工作即可，另一台起到备份作用，并在其上面都配置keepalived。

配置web服务器

（web1）ifconfig eth0 192.168.4.1

yum -y install httpd

systemctl start httpd;systemctl enable httpd

（web2）ifconfig eth0 192.168.4.2

yum -y install httpd

systemctl start httpd;systemctl enable httpd

使用keepalived为服务器提供VIP

主：

global\_defs {

notification\_email {

[acassen@firewall.loc](mailto:acassen@firewall.locl)

[failover@firewall.loc](mailto:failover@firewall.loc)

[sysadmin@firewall.loc](mailto:sysadmin@firewall.loc)

}

notification\_email\_from [Alexandre.Cassen@firewall.loc](mailto:Alexandre.Cassen@firewall.loc)

smtp\_server 192.168.200.1

smtp\_connectiom timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

# vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gan\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER #辅助写BACKUP

interface eth0

virtual\_router\_id 51 #虚拟路由器ID号

priority 150 #优先级

advert\_int 1 #心跳消息发送间隔

authenrication {

auth\_type PASS

auth\_pass 1111 #集群成员共享密码

}

virtual\_ipaddress {

192.168.4.100 #VIP地址

}

}

systemctl restart keepalived

备：

global\_defs {

notification\_email {

[acassen@firewall.loc](mailto:acassen@firewall.locl)

[failover@firewall.loc](mailto:failover@firewall.loc)

[sysadmin@firewall.loc](mailto:sysadmin@firewall.loc)

}

notification\_email\_from [Alexandre.Cassen@firewall.loc](mailto:Alexandre.Cassen@firewall.loc)

smtp\_server 192.168.200.1

smtp\_connectiom timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

# vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gan\_interval 0

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

virtual\_router\_id 51 #虚拟路由器ID号

priority 100 #优先级

advert\_int 1 #心跳消息发送间隔

authenrication {

auth\_type PASS

auth\_pass 1111 #集群成员共享密码

}

virtual\_ipaddress {

192.168.4.100 #VIP地址

}

}

systemctl restart keepalived

查看：

ip a s eth0

2为主，3为辅，当2宕机后，3会成为主，VIP会到3上，当故障节点恢复后，VIP会回到原来的主上

双主配置：两台主机，两个服务，每个主机是一个服务的主

1. keepalived+lvs（高可用、负载均衡的web集群）

keepalived+lvs拓扑

使用keepalived高可用解决调度器单点失败的问题

主、备调度器上配置LVS

主调度器异常时，keepalived启用备用调度器

配置步骤

1. 两台web服务器，两台lvs（调度器）服务器作为主备服务器
2. 在web服务器的lo上配置VIP，并调整内核参数

配置VIP

cp /etc/sysconfig/network-scripts/ifcfg-lo{,:0}

vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.100

NETMASK=255.255.255.255

NSETWORK=192.168.4.100

ONBOOT=yes

NAME=lo:0

ifup lo:0

调整内核参数

echo 1 > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/lo/arp\_announce

echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp\_announce

1. 调度器上不要手工为eth0配置VIP了，因为VIP出现在活跃的调度器上，活跃设备由keepalived决定。应该把已经存在的VIP移除
2. 将LVS规则清掉，因为LVS规则由keepalived配置文件指定
3. 调度器上安装ipvsadm和keepalived

yum -y install ipvsadm keepalived

1. 编辑keepalived配置文件

lvs相关信息通过keepalived配置即可

主：

global\_defs {

notification email {

root@localhost #邮件接收地址

}

notification\_email\_from [admin@tedu.cn](mailto:admin@tedu.cn) #邮件发送地址

smtp\_server 127.0.0.1 #邮件服务器地址

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

# vrrp\_strict

vrrp\_grap\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 150

advert\_int 1

authentication {

auto\_type PASS

auth\_pass 1111

}

virtual ipaddress {

192.168.4.100

}

}

virtual\_server 192.168.4.100 80 {

delay\_loop 6 #延迟轮询的时间

lb\_algo rr

lb\_kind DR

persistebce\_timeout 50

protocol TCP

real\_server 192.168.4.2 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.1 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

备：

global\_defs {

notification email {

root@localhost #邮件接收地址

}

notification\_email\_from [admin@tedu.cn](mailto:admin@tedu.cn) #邮件发送地址

smtp\_server 127.0.0.1 #邮件服务器地址

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

# vrrp\_strict #不禁用的启动的话会把iptables设置一下

vrrp\_grap\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auto\_type PASS

auth\_pass 1111

}

virtual ipaddress {

192.168.4.100

}

}

virtual\_server 192.168.4.100 80 {

delay\_loop 6 #延迟轮询的时间

lb\_algo rr

lb\_kind DR

persistebce\_timeout 50

protocol TCP

real\_server 192.168.4.2 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.1 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

1. 启动服务并验证

systemctl restart keepalived

ipvsadm -Ln

宕机一个服务器，ipvsadm规则少一个，并有邮件提醒

1. HAProxy服务器

概述

是免费、快速并且可靠的一种解决方案

适用于那些负载特别大的web站点，这些站点通常又需要会话保持或七层处理

提高高可用、负载均衡以及TCP和HTTP应用的代理

工作模式

mode http

客户端请求被深度分析后再发往服务器

mode tcp

客户端与服务器之间建立会话，不检查第七层信息

mode health

仅做检查状态，不建议使用

HTTP协议

当HAProxy运行在HTTP模式下，HTTP请求（request）和响应（response）均被完全分析和索引，这样有利于创建恰当的匹配规则

事务模型（每一个请求仅能对应一个响应）

HTTP close（将请求回应给客户端立即断开）

客户端向服务器建立一个tcp连接

客户端发送请求给服务器

服务器响应客户端请求后立即断开连接

如果客户端到服务器的请求不只一个，那么就要不断的去建立连接

TCP三次握手消耗较大的系统资源，同时延迟较大

keep-alive（整个会话结束后，才断开连接）

一次可以传输多个请求

客户端需要知道传输内容的长度，以避免无限期的等待传输结束

降低两个HTTP事务间的延迟

需要相对较少的服务器资源

pipelining（打开多个图片，会同时出现）

仍然使用keep-alive

在发送后续请求后，不用等前面的请求已经得到回应

适用于有大量图片的页面

降低了多次请求之间的网络延迟

HAProxy配置

工作原理

它的工作原理是proxy代理。客户端把请求发送到HAProxy后，HAProxy代替用户发送请求到web服务器，webweb服务器响应HAProxy，把页面发给HAProxy。HAProxy再把页面发送给客户端

准备

一个HAProxy服务器（供用户访问）

多个WEB服务器

HAProxy安装

yum -y install haproxy

修改配置文件

配置参数来源

命令行：总是具有最高优先级（临时生效）

global部分：全局设置进程级别参数

代理声明部分

来自于default、listen、frontend和backend

配置文件可由如下部分构成

defaults

为后续的其他部分设置缺省参数

缺省参数可以被后续部分重置

frontend

描述接受客户端侦听套接字（socket）集

backend

描述转发链接的服务器集

listen

把frontend和backend结合到一起的完整声明

/etc/haproxy/haproxy.cfg

global

log 127.0.0.1 local2

chroot /usr/local/haproxy

pidfile /var/run/haproxy.pid #haproxy的pid存放路径

maxconn 4000 #最大连接数

user haproxy

group haproxy

daemon

defaults

mode http #默认的模式mode{tcp|http|health}log global采用全局定义的日志

option dontlognull #不记录健康检查的日志信息

option httpclose #每次请求完毕后主动关闭http通道

option httplog #日志类别http日志格式

option forwardfor #后端服务器可以从http header中获得客户端id

timeout connect 10000 #如果backend没有指定，默认为10s

timeout client 300000 #客户端连接超时

timeout server 300000 #服务器连接超时

maxconn 60000 #最大连接数

retries #3次连接失败就认为服务不可用，也可以通过后面设置

listen stats

bind 0.0.0.0:1080 #监控页面端口号

stats refresh 30s #页面自动刷新时间是30秒

stats uri /monitor #页面网址

stats realm haproxy manager

stats auth admin:admin #用户名：密码

stats hide-version

listen myweb 0.0.0.0:80

cookie SERVERID rewrite

balance roundrobin #调度算法是rr

server web1 192.168.1.252:80 cookie ljh1 check inter 2000 rise 2 fall 5 # 2000毫秒检查一次服务器，2次成功是好的，5次失败是故障

server web2 192.168.1.213:80 cookie ljh2 check inter 2000 rise 2 fall 5

管理服务

启动服务

systemctl start haproxy

停止服务

systemctl stop haproxy

查看状态

systemctl status haproxy

配置本机接受通过网络发来的日志

vim /etc/rsyslog.conf（去注释）

$ModLoad imudp

$UDPServerRun 514

$ModLoad imcp

$InputTCPServerRun 514

systemctl restart rsyslog

查看

tail -f /var/log/messages

监控HAProxy状态

浏览器输入ip地址/monitor

输入ip地址浏览网页

1. ceph分布式文件系统（分布式存储）

分布式文件系统是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连

分布式文件系统的设计基于客户机/服务器模式

常用的分布式文件系统

Lustre Hadoop FastDFS ceps glusterFS

是一个分布式的文件系统，可以直接往里头加存储空间

具有高扩展、高可用、高性能的特点

ceph可以提供对象存储、块存储、文件系统存储

可以提供PB级别的存储空间

ceph组件

OSDs存储设备。真正存储数据的组件。一般来说，每款参与存储的磁盘都需要一个OSD进程

monitors集群监控组件。MON通过保存一系列集群状态map来监视集群状态，所以需要多台；另外mon需要是基数，如果出现分歧，则采用投票机制，少数服从多数

MDSs：存放文件系统的元数据（对象存储和块存储不需要该组件），只有cephfs需要它。元数据：metadata，存储数据的数据。比如一本书内容是数据，那么书的作者、出版社、出版时间之类的信息就是元数据

RADOS：可靠自主分布式对象存储。它是ceph存储的基础，保证一切都以对象形式存储

RDB：RADOS块设备，提供块存储

cephfs：提供文件系统界别的存储

RGW：RADOS网关，提供对象存储

应用

1. 块存储：使用最多的方式
2. cephfs：了解，不成熟
3. 对象存储：了解，使用亚马逊s3

部署ceph集群（一台客户机、三台存储集群虚拟机）

编写Hosts（在node1上做，同步到其他主机）

for i in {1..6}

do

echo -e “192.168.4.$i\tnode$i.tedu.cn\tnode$i” >> /etc/hosts

done

for i in node{1..6}

do

scp /etc/hosts $i:/etc

done

配置yum源，挂载rhcs2.0-rhosp9-20161113-86\_64.iso镜像文件（在Node1上做，Node1为管理节点）

yum -y install httpd

mkdir /var/www/html/ceph

mount -o loop rhcs2.0-rhosp9-20161113-86\_64.iso /var/www/html/ceph

systemctl restart httpd

vim /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=http://192.168.4.1/ceph/rhceph-2.0-rhel-7-x86\_64/MON

enable=1

gpgcheck=0

[osd]

name=osd

baseurl=http://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/OSD

enable=1

gpgcheck=0

[tools]

name=tools

baseurl=http://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/Tools

enable=1

gpgcheck=0

for i in node{1..6}

do

scp /etc/yum.repos.d/ceph.repo $i:/etc/yum./repos.d/

done

配置ssh无密码连接

非交互生成密钥对

ssh-keygen -f /root/.ssh/id\_rsa -N ‘ ’

发送密钥到各个主机（包括自己）

for i in node{1..6}

do

ssh-copy-id 192.168.4.$i

done

NTP时间同步（配置node6为时间服务器）

创建NFP服务器

yum -y install chrony

vim /etc/chrony.conf

server0.centos.pool.ntp.org iburst

allow 192.168.4.0/24

local stratum 10

systemctl resstart chronyd

其他所有主机与其同步时间（以node1为例）

vim /etc/chrony.conf

server 192.168.4.6 iburst

systemctl restart chronyd

for i in {2..4}

do

scp /etc/chrony.conf node$i:/etc

done

for i in {2..4}

do

ssh node$i systemctl restart chronyd

done

准备存储磁盘

物理机上为每个虚拟机创建3个磁盘（node1-node3各添加三个10GB磁盘）

cd /var/lib/libvirt/images

qemu-img create -f qcow2 node1-vdb.vol 10G

qemu-img create -f qcow2 node1-vdb.vol 10G

qemu-img create -f qcow2 node1-vdb.vol 10G

qemu-img create -f qcow2 node2-vdb.vol 10G

qemu-img create -f qcow2 node2-vdb.vol 10G

qemu-img create -f qcow2 node2-vdb.vol 10G

qemu-img create -f qcow2 node3-vdb.vol 10G

qemu-img create -f qcow2 node3-vdb.vol 10G

qemu-img create -f qcow2 node3-vdb.vol 10G

在图形环境中为虚拟机添加磁盘

virt-manager

部署ceph集群

安装部署软件

使用node1作为部署主机

yum -y install ceph-deploy

ceph-deploy命令与子命令都支持--help查看帮助

ceph-deploy --help

创建目录

为部署工具创建目录，存放密钥与配置文件

mkdir ceph-cluster

cd ceph-cluster

创建ceph集群配置（所有节点都为mon）

ceph-deploy new node{1..3]

给所有节点都安装ceph软件包

ceph-deploy install node{1..3}

初始化所有节点的mon服务（主机名解析必须对）

ceph-deploy mon create-initial

这里没有指定主机，因为第一步创建的配置文件里头已经有了，所以要求主机名解析必须对，否则连接不到对应的主机

如果有报错：

[ERROR]admin\_socket:expection getting command descriptions:[Errno2]no such file or deirectory

解决方案：

vim ceph.conf最下面加入public\_network=192.168.0.0/24

再执行以下命令：ceph-deploy --overwrite-conf config push node1 node2 node3

创建OSD

把node1-3的vdb作为日志盘。ext/xfs都是日志文件系统，一个分区分成日志区和数据区。为了更好的性能，vdb专门作为vdc和vdd的日志盘（以node1为例）

for i in node{1..3}

do

ssh $i parted /dev/vdb mklabel gpt

done

for i in node{1..3};do ssh $i parted /dev/vdb mkpart primary 1M 50%;done

for i in node{1..3};do ssh $i parted /dev/vdb mkpart primary 50% 100%;done

for i in node{1..3};do ssh$i chown ceph.ceph /dev/vdb?;done

创建udev规则，使得vdb1和vbd2重启后，属主属组仍然是ceph

vim /etc/udev/rules.d/90-cephdisk.rules

ACTION==”add”,KERNEL==”vdb?”,OWNER=”ceph”,group=”ceph”

reboot

for i in node{1..3}; do scp /etc/udev/rules.d/90-cephdisk.rules $i:/etc/udev/rules.d/;done

初始化清空磁盘数据（仅node1操作即可）

for i in {1..3}

do

ceph-deploy disk zap node$i:vdc node$i:vdd

done

创建OSD存储空间（仅node1操作即可）

for i in {1..3}

do

ceph-deploy osd create node$i:vdc:/dev/vdb1 node$i:vdd:/dev/vdb2

done

创建osd存储设备，vdc为集群提供存储空间,vdb1提供日志，不需要很大

验证

查看

ceph -s（ok即可）

可能出现的错误

osd create创建OSD存储空间，如提示run ‘gatherkeys’

ceph-deploy gatherkeys node1 node2 node3

ceph -s查看状态，如果失败

systemctl restart ceph\\*.service ceph\\*.target

在所有节点，或仅在失败的节点重启服务

重启后状态为error，连接超时

systemctl restart ceph\\*

ceph块存储（在创建时，如果创建的镜像容量比真实大小小的话，在使用满之后需要进行扩容操作，如果创建的镜像比真实容量大，则只能使用真实容量大小）

什么是块设备

单机块设备

光盘

磁盘

分布式块存储

ceph

cinder

ceph块设备也叫做RADOS块

RADOS block device：RBD

RBD驱动已经很好的集成在了linux内核中

RBD提供了企业功能，如快照、COW克隆等等

RBD还支持缓存，从而大大提高性能

linux内核可以直接访问ceph块存储

KVM可用借助于librbd访问

块存储集群

创建镜像

查看存储池（默认只有一个rbd池）

ceph osd lspools

创建镜像、查看镜像

创建名为demo-image的镜像，大小为10GB

rbd create demo-image --image-feature layering --size 10G

创建镜像，名为image，指定它位于rbd池中

rbd create rbd/image --image-feature layering --size 10G

rbd list查看所有镜像列表

rbd info demo-image查看指定镜像信息

动态调整大小

缩小容量

rbd resize --size 7G image --allow-shrink

rbd info image

扩大容量

rbd resize --size 15G image

rbd info image

客户端通过KEDB访问

客户端需要安装ceph-common软件包

yum -y install ceph-common

拷贝配置文件（否则不知道集群在哪里，该配置文件里面记录了ceph集群访问的方式和地址）

scp node1:/etc/ceph/ceph.conf /etc/ceph

拷贝连接密钥（否则无连接权限）

scp node1:/etc/ceph/ceph.client.admin.keyring /etc/ceph

ceph.client.admin.keyring是client.admin用户的密钥文件，client是类型

映射镜像到本地磁盘

rbd map image查看映射到本地的文件名称

rbd showmapped

客户端格式化、挂载分区

mkfs.xfs /dev/rbd0

molunt /dev/rdb0 /mnt

创建镜像快照

查看镜像快照

rbd snap ls image

创建镜像快照

rbd snap create image --snap image-snap1//给image创建image-snap1的快照

rbd snap ls image

使用快照恢复数据

删除客户端写入的测试文件

rm -rf /mnt/test.txt

卸载挂载

umount /mnt

还原快照

rbd snap rollback image --snap image-snap1

客户端重新挂载分区

mount /dev/rdb0 /mnt

ls /mnt

快照克隆（克隆出来的是镜像，也就是说一个快照可以创建多个镜像）

如果想从快照恢复一个新的镜像，则可以使用克隆。注意：克隆前，需要对快照进行<保护>操作，被保护的快照无法删除，如果要删除，需要取消保护（unprotect）

克隆快照，首先要先将快照保护请求，防止误删除之类的操作

rbd snap protect image --snap image-snap1

克隆image-snap1快照，克隆名称是image-cl1

rbd clone image --snap image-snap1 image-cl1 --image-feature layering

使用image的快照image-snap1克隆一个新的image-cl1镜像

查看信息（克隆镜像与父镜像快照关系）

rbd info image-cl1

合并

克隆镜像很多数据来自于快照链，如果希望和克隆镜像可以独立工作，就需要将父快照中的数据，全部拷贝一份，但比较耗时

rbd flatten image-cl1

rbd info image-cl1

客户端撤销磁盘映射

umount挂载点

umount /mnt

取消RBD磁盘映射

rbd showmapped

rbd unmap /dev/rbd/rbd/image

删除快照与镜像

删除快照（确保快照未被保护）

rbd snap ls image

rbd snap unprotect image --snap image-snap1（在保护的状态下，取消保护）

rbd snap rm image --snap image-snap1

删除镜像

rbd list

rbd rm image

块存储应用案例（安装KVM虚拟机，是用ceph存储提供的镜像作为硬盘）

1. 创建磁盘镜像

为虚拟机创建磁盘镜像

rbd create vm1-image --image-feature layering --size 10G

查看镜像

rbd info vm1-image

rbd list

qemu-img info rbd:rbd/vm1-image

ceph认证账户（不需要更改，查看即可，要记住密码文件路径，淫威远程访问需要）

ceph默认开启用户认证，客户端需要账户才可以访问

默认账户名称为client.admin,key是账户的密钥

可以使用ceph auth添加新账户（案例使用默认账户）

cat /etc/ceph/ceph.conf

ceph.client.admin.keyring

1. 部署客户端环境（将物理机作为客户端，安装软件包，拷贝配置文件）

注意：这里使用真实机当客户端

客户端需要安装ceph-common软件包

拷贝配置文件（否则不知道集群在哪里）

拷贝连接密钥（否则无连接权限）

yum -y install ceph-common

scp 192.168.4.1:/etc/ceph/ceph.conf /etc/ceph

scp 192.168.4.1:/etc/ceph/ceph.client.admin.keyring /etc/ceph

1. 创建虚拟机

使用virt-manager创建一台普通的KVM虚拟机rhel7

创建后不要启动虚拟机，关闭虚拟机

1. 导出虚拟机声明文件，然后将其删除

virsh dumpxml rhel7 > /tmp/rhel7.xml

1. 虚拟机使用ceph存储，需要认证。方式是虚拟机先生成secret，再将secret与ceph账户映射

配置libvirt secret

KVM虚拟机需要使用librbd才可以访问ceph集群

librbd访问ceph需要账户认证

所以，需要给libvirt设置账户信息

编写账户信息文件（真实机操作）

vim /tmp/secret.xml

<secret ephemeral=’no’ private=’no’>

<usage type=’ceph’>

<name>client.admin secret</name>

</usage>

</secret>

使用XML配置文件创建secret

virsh secret-define --file /tmp/secret.xml

查看secret

virsh secret

将虚拟机软件的secret与ceph的管理员用户关联（真实机操作）

查看管理员的密钥

ceph auth get-key client.admin

//获取client.admin的key，或直接查看密钥文件

cat /etc/ceph/ceph.client.admin.keyring

关联secret和ceph的管理员用户

virsh secret-set-value --secret 733f0fd1-e3d6-4c25-a69f-6681fc19802b --base64 AQBTsdRapUxBKRAANXtteNUyoEmQhveb75blsg

//这里secret后面是之前创建的secret的UUID

//base64后面是client.admin账户的密码

//现在secret中既有账户信息又有密钥信息

1. 修改虚拟机的配置文件/tmp/rhel7.xml，把管理员信息写到该文件中，并指定虚拟磁盘使用的ceph镜像

每个虚拟机都会有一个XML配置文件，包括（虚拟机的名称、内存、CPU、磁盘、网卡等信息）

vim /tmp/rhel7.xml

<disk type=’network’ device=’disk’>

<driver name=’qemu’ type=’raw’/>

<auth username=’admin’>

<secret type=’ceph’ uuid=’733f0fd1-e3d6-4c25-a69f-6681fc19802b’>

</auth>

<source protocol=’rbd’ name=’rbd/vm1-image’>

<host name=’192.168.4.1’ port=’6789’/>

</source>

...

1. 利用xml文件生成虚拟机并验证

virsh define /tmp/rhel7.xml

分布式文件系统（可以像NFS或SAMBA那样，提供共享文件夹，客户端通过挂载目录的方法，使用ceph的存储）

什么是cephfs

分布式文件系统是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连

环境准备

准备一台新的服务器，作为元数据（任何文件系统中的数据分为数据和元数据，数据是指普通文件中的实际数据，元数据是指用来描述一个文件的特征的系统数据，比如访问权限、文件拥有者以及文件数据块的分布信息灯，所以cepfs必须有MDSs节点）服务器，需要配置IP地址和主机名node4，同步hosts,yum源，与NTP服务器同步时间，惯例节点需要无密码登录新节点

登录node4

yum -y install ceph-mds

登录node1进行元数据节点部署

cd /root/ceph-cluster

//该目录是最早部署ceph集群的时候，创建的目录

ceph-deploy mds create node4

//给node4拷贝配置文件，启动mds服务

ceph-deploy admin node4

//同步配置文件和key给node4

创建存储池

文件系统最少需要2个池（1个用于存储数据，名字为cephfs\_data，一个用于存储元数据，名字为cephfs\_metadata）

ceph osd pool create cephfs\_data 128

ceph osd pool create cephfs\_metadata 128

128表示PG的数目是128。PG是归置组，文件存到PG中，PG存在池中

创建ceph文件系统

使用前面创建的池，创建名为myfs1的文件系统

ceph mds stat//查看mds状态

ceph fs new myfs1 cephfs\_metadata cephfs\_daata

//注意，先写metadata池，再写data池

//默认只能创建1个文件系统，多余的会报错

客户端挂载

linux内核支持ceph文件系统（不需要装软件）

mount -t ceph 192.168.4.1:6789:/ /mnt/cephfs -o name=admin,secret=AQBTsdRapUxBKRAANXtteNUyoEmQhveb75blsg

//注意：文件系统类型为ceph

//192.168.4.1为mon节点的ip（不是mds节点）

//admin是用户.secret是密钥

//密钥可以在/etc/ceph/ceph.client.admin.keyring中找到

ceph对象存储

对象存储也就是键值存储，通其接口指令，也就是简单的GET、PUT、DEL和其他扩展，向存储服务上传下载数据，对象存储中所有数据都被认为是一个对象，所以，任何数据都可以存入对象服务器，如图片、视频、音频等，RGW是ceph对象存储网关，用于向客户端应用呈现存储界面，提供API访问接口，可以部署多台

准备新的服务器，作为元数据服务器需要有IP地址和主机名node5，配置yum源，与NTP服务器同步时间，管理节点允许无密码登录此电脑，同步hosts

部署RGW软件包

用户通过RGW访问存储集群

通过node1安装ceph-radosgw软件包

ceph-deploy install --rgw node5

同步配置文件与密钥

ceph-deploy admin node5

新建网关

启动一个rgw服务（在控制节点做）

ceph-deploy rgw create node5

登录node5验证服务是否启动成功

ps aux | grep radosgw

systemctl status ceph-radosgw@\\*

修改服务端口

登录node5，RGW默认端口为7480，修改为8000或80更方便客户端记忆和使用

vim /etc/ceph/ceph.conf

[client.rgw.node5]

host=node5//node5为主机名

rgw\_frontends=”civetweb port=80”//civetweb是RGW内置的一个web服务

systemctl restart ceph-radosgw\\*

客户端测试

仅测试RGW是否正常工作

curl loclhost

使用第三方软件访问

登录node5（RGW）创建用户

radosgw-admin user create --uid=”ljh” --display-name=”firstuser”

注意access\_key和secret\_key

客户端安装软件

yum install s3cmd-2.0.1+-1.el7.noarch.rpm

配置软件

s3cmd --configure

access key:上头创建的用户的access key

secret key:上头创建的用户的secret key

S3 Endpoint[s3.amazonaws.com]:rgw IP地址:端口号

[%(bucket)s.s3.amazonaws.com]:%(bucket)s.rgw IP地址:端口号

Use HTTPS protocol[Yes]:no

test access with supplied credentials?[y/n]:y

save settings?[y/n]:y

//其它提示都回车即可

客户端测试

s3cmd ls

创建存储数据的bucket（类似于存储数据的目录，格式要求xxx\_xxx）

s3cmd mb s3://my\_bucket

s3cmd ls

上传

s3cmd put /etc/hosts s3://my\_bucket

s3cmd ls s3://my\_bucket

下载

s3cmd get s3://my\_bucket/hosts /tmp

删除

s3cmd del s3://my\_bucket/hosts

小公司nfs iscsi就够了，大一点的常用fastdfs，再大的会上ceph glusterfs hdfs

1. KVM虚拟化

虚拟化概述

virtualization资源管理

x个物理资源->y个逻辑资源

虚拟机组成

内核虚拟化模块（KVM）

系统设备仿真（QEMU，虚拟硬件）

虚拟机实例程序（LIBVIRT）

一个XML文件（虚拟机配置声明文件，位置/etc/libvirt/qemu）

一个磁盘镜像文件（虚拟机的硬盘，位置/var/lib/libvirt/images）

安装虚拟化服务器平台

KVM/QEMU/LIBVIRTD

KVM是linux内核的模块，它需要CPU的支持，采用硬件服务虚拟化技术intel-vt,amd-v,内存的相关如intel的EPT和AMD的RVI技术

QEMU是一个虚拟化仿真工具，通过ioctl与内核KVM交互完成对硬件的虚拟化支持

libvirt是一个对虚拟化管理的接口和工具，提供用户端程序virsh,virt-install,virt-manager,virt-view与用户交互

必备软件

qemu-kvm #为kvm提供底层支持

libvirt-client #virsh等软件

libvirt-daemon #libvirtd服务进程

libvirt-daemon-driver-qemu #virsh的qemu驱动

yum -y install qemu-kvm libvirt-client libvirt-daemon libvirt-daemon-driver-qemu

可选功能

virt-install #系统安装工具

virt-manager #图形管理工具

virt-v2v #虚拟机迁移工具

virt-p2v #物理机迁移工具

配置虚拟机上网（启动libvirtd服务端）

systemctl start libvirtd

cd /etc/libvirt/qemu/networks

调用dnsmasq提供DNS、DHCP等功能

建立vbr虚拟网络（192.168.1.0/24）配置文件

vim vbr.xml

<network>

<name>vbr</name>

<bridge name=”vbr”/>

<forward mode=”nat”/>

<ip address=”192.168.1.254” netmask=”255.255.255.0”>

<dhcp>

<range start=”192.168.1.100” end=”192.168.1.200”/>

</dhcp>

</ip>

</network>

虚拟网络管理

virsh管理虚拟网络

virsh net-list查看虚拟网络

virsh net-define vbr.xml创建虚拟网络

virsh net-undefine vbr删除虚拟网络

virsh net-start vbr启动虚拟网络

virsh net-destory vbr停止虚拟网络

virsh net-edit vbr修改vbr网络的配置

virsh net-autostart vbr设置vbr虚拟网络开机自动启动

virsh控制管理器

操作虚拟机

virsh命令工具介绍

提供管理各虚拟机的命令接口

支持交互模式，查看/创建/停止/关闭

进入virsh命令或者命令行都可以操作

格式：virsh 控制指令 [虚拟机名称] [参数]

查看虚拟化信息

查看KVM节点（服务器）信息

virsh nodeinfo

列出虚拟机

virsh list [--all]

all会把关机状态的也列出来

列出虚拟网络

virsh net-list [--all]

查看指定虚拟机的信息

virsh dominfo虚拟机名称

开关机操作

运行|重启|关闭指定的虚拟机

virst start|reboot|shutdown 虚拟机名称

强制关闭指定的虚拟机

virsh destroy 虚拟机名称

将指定的虚拟机设置为开机启动

virsh autostart [--disable] 虚拟机名称

管理KVM平台

连接本地/远程KVM

使用本地/远程KVM

连接本地

virsh#connect qemu:///system（默认选项）

virsh -c gemu:///system

连接远程

virsh#connect qemu+ssh://user@ip地址:port/system

virsh -c qemu+ssh://user@ip地址:port/system

使用virt-manager客户端管理工具

virsh-manager也可以通过add connection管理其他机器上的虚拟机，一般通过“remote tunnel over ssh”就可以了。需要注意的是virt-manager需要使用ssh免密码登录，如果没有免密钥登录需要单独安装python的ssh相关模块，这里使用部署key的方法解决

导出虚拟机

xml配置文件

定义了一个虚拟机的名称、CPU、内存、虚拟磁盘、网卡等各种参数设置

默认位于/etc/libvirt/qemu/虚拟机名.xml

导出xml配置文件

查看：virsh dumpxml 虚拟机名

备份：virsh dumpxml 虚拟机名 > 虚拟机名.xml

清除模版配置

virsh undefine node

编辑虚拟机设置

对虚拟机的设置进行调整

编辑：virsh edit 虚拟机名

若修改name、memory、disk、network，可自动保存为新的虚拟机配置

导入虚拟机

根据修改后的独立xml文件定义新虚拟机

virsh define xml描述文件

删除虚拟机

必要时可以去除多余的xml配置

比如虚拟机改名的情况

避免出现多个虚拟机的磁盘MAC地址冲突

virsh undefine 虚拟机名

镜像管理（默认的在/var/lib/libvirt/images/下面）

常见的虚拟机镜像盘类型

RAW：不是KVM默认，高IO效率，占用空间大，不支持压缩，不支持后端盘复用，不支持快照

QCOW2：是KVM默认，较高IO效率，占用空间较高，支持压缩，支持后端盘复用，支持快照

创建虚拟机磁盘镜像

创建新的镜像盘文件

qemu-img create -f 格式 磁盘路径 大小

qemu-img create -f qcow2 disk.img 50G

当使用量变大，它会变大，最大不悔炒锅预设值50G的大小

查看镜像盘文件的信息

qemu-img info 磁盘路径

qemu-img info disk.img

-b使用后端模版文件

qemu-img create -f qcow2 -b 模版文件 新磁盘名 size

size不指定，使用模版大小

qemu-img create -b disk.img -f qcow2 disk1.img

QCW2技术

cow技术原理

copy on write，写时复制

直接映射原始盘的数据

当原始盘的旧数据有修改时，在修改质检自动将旧数据存入前端盘

对前端盘的修改不会写到原始盘

原理：用户请求前端盘，如果没有数据，前端盘会转发到后端盘，如果要进行写操作，后端盘会把数据复制到前端盘，用户对前端盘修改，不会保存到后端盘

创建/还原/删除快照

快照的作用

在虚拟机磁盘镜像内记录不同时间点的状态数据备份，必要时可将虚拟机恢复到指定的快照

基本用法

qemu-img snapshot -c 快照名 qcow2磁盘（创建）

qemu-img snapshot -l qcow2磁盘（查看）

qemu-img snapshot -a 快照名 qcow2磁盘（还原）

qemu-img snapshot -d 快照名 qcow2磁盘（删除）

XML配置文件详解

配置文件  
 <!--是注释的开头

-->是注释的结尾

xml标签必须成对出现

<keyword>...</keyword>

配置文件组成

id个性化设置，不重要，可以不用管，生成模版时可以去掉

uuid个性化设置，不重要，可以不用管，生成模版时可以去掉

memory unit虚拟机最大使用内存，可以手动调整

currentmemory 创建虚拟机使用内存，free -m可以查看的

CPU的数量设置，可以调整

<vcpu placement=’static’>1</vcpu>

系统资源设置，不用修改

<resource>...</resource>

系统配置机引导设备，不需要更改

<os>...</os>

电源相关配置

<features>...</features>

CPU设置，可以修改成使用真机CPU，这里可以虚拟CPU，如果指令集写错，会出现卡顿状况，以下是修改后的结果

<cpu mode='host-passthrough'>

</cpu>

时钟相关设置，科一设置成与物理机时间一致，以下是修改的结果

<clock offset='localtime'/>

重启，关机，强制关闭对应的命令

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

内存及硬盘休眠相关设置，可以删除

<pm>...</pm>

仿真设备相关配置<devices>...</devices>

总线地址，别名配置，都可以删除

<address......

<alias...

硬盘配置，需要更改

<emulator>/usr/libexec/qemu-kvm</emulator>

<disk ...</disk>

usb相关配置，如果不用，可以删除

<controller type='usb'

type=’pci’ pci总线网络，可以删除

type=’virtio-serial’ 串口设备需要保留串口终端

type=’network’网络配置需要修改

type=’pty’串口终端，需要保留

type=’unix’虚拟通信控制接口

type=’spicevmc’图形图像相关配置，可以删除

type=’tablet’数位板，可以删除

type=’mouse’鼠标，保留

type=’keyboard’键盘保留

graphics、video图形图像显卡相关配置，可以删除

sounds声卡相关配置，可以删除

redirdev设备重定向，可以删除

memballoon内存气泡，可以动态调整内存

其它配置

驱动安全标签，可删除

<seclabel>......</seclabel>

自定义系统安装配置（后端文件）

系统安装

创建虚拟网络

cd /etc/libvirt/qemu/networks

vim vbr.xml

<network>

<name>vbr</name>

<bridge name=”vbr”/>

<forward mode=”nat”/>

<ip address=”192.168.1.254” netmask=”255.255.255.0”>

<dhcp>

<range start=”192.168.1.100” end=”192.168.1.200”/>

</dhcp>

</ip>

</network>

virsh net-define vbr.xml

virsh net-start vbr

virsh net-autostart vbr

创建网络yum源

mount -t iso9660 -o loop,ro xxxxxxx.iso /var/ftp/centos7

创建一个qcow2磁盘镜像文件

cd /var/lib/libvirt/images

qemu-img create -f qcow2 node.cow2 16G

使用刚刚创建的网络yum源安装系统

软件包要最小化，选择网络安装并输入URL，选择新创建的存储，网络 选择自己创建的，硬盘自己进行分区

初始化系统并制作模版

配置yum源

[local\_repo]

name=centos-$releaserver-base

baseurl=ftp://192.168.1.254/centos7

enable=1

gpgcheck=1

导入gpg key

rpm --import ftp://192.168.1.254/centos7/RPM-GPG-KEY-CentOS-7

禁用selinux

/etc/selinux/config

selinux=deisabled

修改网卡配置文件，去除UUID，MAC地址等

删除网络里个性化设置

/etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=”eth0”

ONBOOT=”yes”

IPV6INIT=”no”

TYPE=”Ethernet”

BOOTPROTO=”dhcp”

禁用空路由

/etc/sysconfig/network

NOZEROCONF=”yes”

systemctl restart network

卸载networkmanager,firewalld

systemctl stop networkmanager

yum remove -y firewalld-\* python-firewall NetworkManager-\*

安装常用工具软件net-tools vim-enhanced net-tools bridge-utils psmisc lftp

yum provides 命令名

console及磁盘分区配置

添加console配置及删除磁盘分区里的个性化信息

/etc/default/grub

GRUB\_CMDLINE\_LINUX=”biosdevname=0 net.ifnames=0 console=ttyS0,115200n8”

GRUB\_DISABLE\_LINUX\_UUIUD=”true”

GRUB\_ENABLE\_LINUX\_LABEL=”true”

最后两行添加，第一行修改，其它不需要动

重新生成grub.cfg

grub2-mkconfig -o /boot/grub2/grub.cfg

/etc/fstab文件中照到UUID手工修改成系统设备

blkid查看uuid对应的磁盘设备，修改fstab文件

去除个性化设置

安装分区扩展软件

yum install -y cloud-utils-growpart

设置第一次开机自动扩容根分区

chmod 755 /etc/rc.local

在rc.local里加入如下配置

###

/usr/bin/growpart /dev/vda 1（扩大分区）

/usr/sbin/xfs\_growfs /（扩大文件系统）

/usr/bin/sed’/^###/,$d’ -i /etc/rc.d/rc.local（执行完后删除）

关闭虚拟机后执行信息清理工作

virt-sysprep -d node（虚拟机名称）

yum -y install libguestfs-tools

关机并备份xml文件

virsh dumpxml rhel7 > /tmp/rhel7.xml

去除quest域（删除虚拟机）

虚拟机快建技术

xml配置文件：定义虚拟机名称、CPU、内存、虚拟磁盘、网卡等各种参数的设置

磁盘镜像文件：保存虚拟机的操作系统和文档数据，镜像取决于xml配置文件中的定义

思路：

准备一台模版虚拟机（镜像文件+xml配置文件）

模板虚拟机的选择

可以用现有的虚拟机，也可以再安装一台新的虚拟机

模板机的初始化

1. 预装软件/系统优化/关闭selinux等
2. 生产环境可以标记/.unconfigured，方便新虚拟机的配置（相当于出厂设置，提示用户初始化）
3. 备份好模板盘、xml配置文件
4. 删除模板机（undefine）

基于磁盘复用技术快建新虚拟机的磁盘

cd /var/lib/libvirt/images/

qemu-img通过-b选项复用指定后端盘

qemu-img create -f qcow2 -b 后端盘 前端盘

通过调整模版机的配置快建新虚拟机的xml配置文件

cp /etc/libvirt/qemu/aaa.xml /root

vim /root/aaa.xml

导入新虚拟机

virsh define /root/aaa.xml

virsh start aaa

离线访问虚拟机

使用guestmount工具

支持离线挂载raw、qcow2格式虚拟机磁盘

可以在虚拟机关机的情况下，直接修改磁盘中的文档

方便对虚拟机定制、修复、脚本维护

受selinux机制的影响

如何挂载虚拟盘

基本用法

guestmount -a 虚拟磁盘路径 -i /挂载点

ls 挂载点

chroot 挂载点（chroot进去进行操作，操作完以后exit推出，umount卸载挂载，如果修改了后端盘的话，需要重新制作前端盘）

1. openstack

openstack是一个由NASA（没过郭佳航空航天局）和rackspace合作研发并发起的项目，是一套IaaS解决方案，是一个开源的云计算管理平台，以apache许可证为授权

主要组件

horizon用于管理openstack各种服务的、基于web的管理端口，通过图形用户界面实现创建用户、管理服务、启动实例等操作

keystone为其他服务提供认证和集中身份管理服务，也提供了集中的目录服务

支持多种身份认证模式，如密码认证、令牌认证、以及AWS（亚马逊WEB服务）登录，为用户和其他服务提供了SSO认证服务

neutron一种软件定义网络服务，用于创建网络、子网、路由器、管理浮动IP地址，可以在项目中创建vpn

cinder为虚拟机管理存储卷的服务，为运行在nova（托管节点）中的实例提供永久的块存储，可以通过快照进行数据备份，经常应用在实例存储环境中，如数据库文件

nova在节点上用于管理虚拟机服务，nova是一个分布式的服务，能够与keystone交互实现认证，与glance交互实现镜像管理，nova被设计成在标准硬件上水平扩展，启动实例时，如果由需要则下载镜像

glance扮演虚拟机镜像注册角色，允许用户直接存储拷贝服务器镜像，这些 镜像可以用于新建虚拟机模版

部署环境（一台管理节点，多台nove节点，一台yum源、NTP、dns服务器）

yum仓库及名称解析（单独的一台服务器上安装）

配置yum仓库（12个yum源）

RHEL7光盘必须提供（1个）

RHEL7OSP光盘用有众多目录，每个目录都是一个仓库，因此需要把这些目录都配置为yum源（10个）

RRHEL7-extras扩展包在安装的时候也是需要的（1个）

设置DNS服务器

系统环境准备

OpenStack安装时候需要外部dns来解析域名，并且还需要外部事件服务器来保证所有节点的时间保持一致，需要创建一个DNS服务器，为主机提供域名解析，dns服务器不能与OpenStack安装在同一台主机上

yum -y install bind-chroot

vim /etc/named.conf

listen-on port 53 { any; };

// listen-on-v6 port 53 { ::1; };

directory "/var/named";

dump-file "/var/named/data/cache\_dump.db";

statistics-file "/var/named/data/named\_stats.txt";

memstatistics-file "/var/named/data/named\_mem\_stats.txt";

recursing-file "/var/named/data/named.recursing";

secroots-file "/var/named/data/named.secroots";

allow-query { any; };

recursion yes;

forwarders { 202.106.196.115; };

dnssec-enable no;

dnssec-validation no;

systemctl restart named

systemctl enable named

vim /etc/resolv.conf改成服务器ip（nove节点做）

NTP服务器

yum -y install chrony

vim /etc/chrony.conf

server ntp1.aliyun.com iburst

bindacqaddress 0.0.0.0

allow 0/0

cmdallow 127.0.0.1

systemctl start chronyd

systemctl enable chronyd

nova节点设置为服务器Ip地址

部署openstack

基础环境准备

配置IP地址（都是静态地址）

配置eth0为公共网络，网络地址为192.168.1.0/24

配置eth1为隧道接口，网络地址为192.168.4.0/24

关闭NetworkManager服务

禁用selinux

卸载firewalld

配置yum客户端

将RHEL7光盘作为基础的yum源

将RHEL7-extars光盘作为扩展的yum源

将RHEL7OSP光盘中所有目录作为yum源

配置卷组（管理节点安装）

OpenStack位虚拟机提供云硬盘，本质上是本地逻辑卷，逻辑卷创建于名为cinder-volumes的卷组，没有物理卷可以使用loop文件代替

pvcreate /dev/vdb

vgcreate cinder-volumes /dev/vdb

vgs//验证

导入公钥（所有节点）

安装OpenStack期间，系统会要求验证密钥，手动导入系统密钥，密钥文件在光盘文件中提供

rpm --import PRM-GPG-KEY-CentOS-7

安装额外软件包（所有节点）

在安装期间，有些软件所依赖的软件包，并没有在安装过程中安装，这些软件包需要提前安装，本地RPM包也可以通过yum进行安装

yum install -y qemu-kvm libvirt-client libvirt-daemon libvirt-daemon-driver-qemu python-setuptools

安装OpenStack

配置packstack

安装packstack（管理节点）

yum 0y install openstack-packstack

生成应答文件

packstack --gen-answer-file answer.ini

修改应答文件

CONFIG\_DEFAULT\_PASSWORD=123456 //设定密码

CONFIG\_SWIFT\_INSTALL=n //禁用swift存储

CONFIG\_NTP\_SERVERS=192.168.1.254 //配置NTP服务器

CONFIG\_COMPUTE\_HOSTS=192.168.1.123 //计算节点的IP地址，管理节点的主机IP，不需要更改

CONFIG\_NETWORK\_HOSTS=192.168.1.123 //配置主机的网络，哪些主机要变成这个网络，不需要更改

CONFIG\_CINDER\_VOLUMES\_CREATE=n //禁止自动创建卷，默认会从第一个硬盘的第一个设备创建

CONFIG\_NEUTRON\_ML2\_TYPE\_DRIVERS=flat,vxlan //扁平网络（外网,flat），出口网络，内网vxlan

CONFIG\_NEUTRON\_ML2\_VXLAN\_GROUP=239.1.1.5 //主播地址，vxlan设备互联互通，239开头

CONFIG\_NEUTRON\_OVS\_BRIDGE\_MAPPINGS=physnet1:br-ex //配置虚拟交换机（网桥），br-ex是ifconfig可以看见的设备，用于和外网进行通信

CONFIG\_NEUTRON\_OVS\_BRIDGE\_IFACES=br-ex:eth0 //虚拟交换机跟eth0连接，出外围用

CONFIG\_NEUTRON\_OVS\_TUNNEL\_IF=eth1 //承载多个vlan

CONFIG\_PROVISION\_DEMO=n //测试用的demo，不用测试，直接使用

一键部署OpenStack

安装过程大概20分钟左右，如果错误，根据屏幕上的日志文件进行排错

packstack --answer-file answer.ini

配置外部OVS网桥

配置br-ex为外部网桥（静态IP，自动生成，不用更改）

配置eth0为外部OVS网桥接口（自动更改，不需手动更改）

验证OVS配置

ovs-vsctl show

ip -o addr show

安装虽然没错，但是无法打开，是个bug，修改配置文件

vim /etc/httpd/conf.d/15-horizon\_vhost.conf

WSGIDaemonProcess apache group=apache processes=3 threads=10 user=apache

WSGIProcessGroup apache

WSGIApplicationGroup %{GLOBAL}

WSGIScriptAlias /dashboard "/usr/share/openstack-dashboard/openstack\_dashboard/wsgi/django.wsgi"

重新载入配置apachectl grancful

重启服务systemctl restart openstack\*.service

浏览器输入管理节点ip地址进入web控制页面

用户名密码在keystonerc\_admin中

用户、配额管理

用户管理

身份管理->项目->创建项目

身份管理->项目->创建用户

命令行管理用户

创建user2用户，指定密码位tedu.cn

[root@openstack~(keystone\_admin)]#openstack user create --password tedu.cn user2

设置user2的email地址

[root@openstack~(keystone\_admin)]#openstack user set --email user2@tedu.cn user2

列出所有用户

[root@openstack~(keystone\_admin)]#openstack user list

查看user2信息

[root@openstack~(keystone\_admin)]#openstack user show user2

指定user2可以访问myproject，角色为\_meber\_

[root@openstack~(keystone\_admin)]#openstack role add --user user2 --project myproject \_member\_

查看user2在myproject中的用户

[root@openstack~(keystone\_admin)]#openstack role list --projecct myproject --user user2

禁用用户

[root@openstack~(keystone\_admin)]#openstack user set --disable user2

激活用户

[root@openstack~(keystone\_admin)]#openstack user set --enable user2

修改user2的密码位redhat

[root@openstack~(keystone\_admin)]#openstack user set --password redhat user2

将user2从myproject中移除

[root@openstack~(keystone\_admin)]#openstack role remove --project myproject --user user2 \_member\_

删除user2用户

[root@openstack~(keystone\_admin)]#openstack user delete user2

配额管理

身份管理->项目->编辑项目->配额

通过命令行管理配额

列出项目的缺省配额

[root@openstack~(keystone\_admin)]#nova quota-defaults

列出myproject的配额

[root@openstack~(keystone\_admin)]#nova quota-show --tenant myproject

修改浮动IP地址配额

[root@openstack~(keystone\_admin)]#nova quota-update --floating-ips 20 myproject

云主机类型

管理员->云主机类型->创建云主机类型

通过命令行管理主机类型

列出所有的云主机类型

[root@openstack~(keystone\_admin)]#openstack flavor list

创建一个云主机类型

[root@openstack~(keystone\_admin)]#openstack flavor create --public demo.tiny --id auto --ram 512 --disk 10 --vcpus 1

删除云主机类型

[root@openstack~(keystone\_admin)]#openstack flavor delete demo.tiny

镜像管理

管理员->系统->镜像

通过命令行管理云主机类型

上传镜像

[root@openstack~(keystone\_admin)]#openstack image create --disk-format qcow2 --min-disk 10 --min-ram 512 --file /root/small.img small\_rhel6

列出镜像

[root@openstack~(keystone\_admin)]#openstack image list

查看镜像详情

[root@openstack~(keystone\_admin)]#openstack image show small\_rhel6

修改镜像属性

[root@openstack~(keystone\_admin)]#openstack image set --public small\_rhel6

另存镜像为本地文件

[root@openstack~(keystone\_admin)]#openstack image save --file /tmp/small\_rhel6.img small\_rhel6

网络管理

外网（管理员）：管理员->网络->新建网络

内网（用户）：项目->网络->网络->新建网络

路由（用户）：项目->网络->路由->新建路由

通过命令行管理网络

创建网络

[root@openstack~(keystone\_admin)]#openstack network create --project myproject --enable internal

创建子网

[root@openstack~(keystone\_admin)]#neutron subnet-create --name subnet3 --gateway 192.168.200.1 --enable-dhcp internal 192.168.200.0/24 --tenant-id ff38716297863f894cdd1c98597160c

查看网络详情

[root@openstack~(keystone\_admin)]#openstack network show internal

新建路由器

[root@openstack~(keystone\_admin)]#neutron router-create --tenant-id ff387162978643f894cdd1c98597160c router2

删除网络

[root@openstack~(keystone\_admin)]#openstack network delete internal

[root@openstack~(keystone\_admin)]#neutron router-delete router2

浮动IP

项目->计算->访问和安全->浮动IP->分配IP给项目

通过代码管理浮动IP

分配地址

[root@openstack~(keystone\_admin)]#neutron floatingip-create --tenant-id ff387162978943cdd1c98597160c WAN

查看地址

[root@openstack~(keystone\_admin)]#neutron floatingip-list

安全和实例管理

安全组

项目->计算->访问和安全->创建安全组

实例

项目->计算->云主机数量->创建虚拟机

云主机热迁移（admin）

管理员->云主机数量->选中云主机->云主机热迁移

镜像（admin）

管理员->镜像->创建镜像

云主机类型（admin）

管理员->云主机类型->创建云主机类型

安装计算节点

修改应答文件，只需要在管理节点上做即可

vim answer.ini

CONFIG\_COMPUTE\_HOSTS=192.168.1.143,新节点IP

CONFIG\_NETWORK\_HOSTS=192.168.1.143,新节点IP

在管理节点上执行

packstack --answer-file answer.ini

按提示输入远程密码，本机已经安装的服务，不会被覆盖，只有改动后的选项才需要重新配置

查看节点状态（admin）

管理员->主机聚合

1. docker

容器由以下几个核心技术组成

CGroups-资源管理，限制容器使用的资源

namespace-进程管理

selinux安全

由于是在物理机上实时隔离，启动一个容器，科一像启动一个进程一样快速

docker是完整的一套容器管理系统，提供了一组命令，让用户更加方便直接地使用容器技术，而不需要过多关心底层内核技术

docker特点

docker优点：相对于传统的虚拟化技术，容器更加简洁高效，传统虚拟机要给每个vm安装操作系统，容器使用的共享公共库和程序

docker缺点：容器的隔离性没有虚拟化强，安全性由先天缺陷，selinux难以驾驭，监控容器和容器排错是挑战

docker部署

工作原理：前端容器，后端镜像，一个后端盘可以创建多个容器，就像一个后端盘可以创建多个虚拟机，后端盘用镜像封装而成，采用分层设计

安装软件

安装前准备

需要64位系统

至少RHEL6.5以上系统，强烈推荐RHEL7

关闭防火墙（不是必须）

安装docker

软件包列表：docker-engine、docker-engine-selinux

rpm -ivh docker-engine-1.12.1-1.el7.centos.x86\_64.rpm

rpm -ivh docker-engine-selinux-1.12.1-1.el7.centos.noarch.rpm

systemctl start docker

systemctl enable docker

docker基本命令

镜像常用命令

docker version//查看版本

docker images//查看镜像列表，镜像仓库名称、镜像标签、镜像ID、创建时间、大小

docker history 镜像名//查看镜像制作历史，了解镜像制作过程

docker inspect 镜像名//查看镜像底层信息，了解镜像环境变量、存储卷、标签等信息

docker search 关键字//搜索镜像

docker pull 镜像名//下载镜像

docker push 镜像名//上传镜像

docker rmi 镜像名//删除镜像（如果报错，说明有容器存在，先删除容器）

docker save image\_name:tag > xx.tar//导出镜像（将本地镜像导出为tar包）

docker load < xx.tar//导入镜像（通过tar包文件导入镜像）

docker tag oldimage\_name:tag newimage\_name:tag//创建链接

容器常用命令

docker run -itd 镜像名 镜像内的命令//使用镜像启动容器，-i交互式，-t终端,-d后台进程，--name 指定名称

docker ps//查看正在运行的容器，-a查看所有容器列表，-q仅显示容器ID

docker stop 容器ID//关闭容器，支持使用docker命令结果，如docker stop $(docker ps -q)

docker start 容器ID//启动容器

docker restart 容器ID//重启容器

docker attach|exac//进入容器

docker attach 容器ID//进入容器，exit会导致容器关闭，ctrl+p+q是放到后台

docker exec -itd 容器ID号 容器内命令//进入容器，退出时不会关闭容器，因为会重新启动一个命令去执行

docker inspect 容器ID//查看容器底层信息，主要查看IP地址等信息

docker rm 容器ID//删除容器，注意：删除正在运行的容器时会报错

docker top 容器ID//查看容器进程列表（相当于在容器内部执行ps）

自定义镜像与仓库

自定义镜像

docker commit

使用镜像启动器，在该容器基础上修改，另存为一个镜像

docker run -itd centos /bin/bash

docker ps

docker exec -it IDS /bin/bash修改（增删改数据、安装软件、修改配置文件）

docker commit IDs name:label

docker images

dockerfile相当于创建自定义镜像的脚本

语法格式：

FROM 基础镜像

MAINTAINER 镜像创建者

EXPOSE 开放的端口

ENV 设置的变量

ADD 要复制的文件（本目录底下） 目标目录（容器里的目录）

RUN 制作镜像时执行的命令，可以有多个

WORKDIR 定义容器的默认工作目录

CMD 容器启动时执行的命令，仅可以有一条CMD，如CMD [“/usr/bin/python”]

工作流程：

mkdir build;cd build

vim Dockerfile

FROM ljh

RUN yum -y install httpd

ADD httpd.conf /etc/httpd/conf/httpd.conf

WORKDIR /var/www/html

RUN echo "hello world" > index.html

EXPOSE 80

MAINTAINER ljh

ENV EnvironmentFile=/etc/sysconfig/httpd

CMD ["/usr/sbin/httpd","-DFOREGROUND"]

docker build -t imagenamae dockerfile所在的文件夹

自定义私有仓库

服务器配置私有仓库

编写配置文件

vim /etc/docker/daemon.json

{

"insecure-registries" :["192.168.1.129:5000"]

}

重启服务

systemctl restart docker

启动私有仓库服务

docker run -d -p 5000:5000 registry

给镜像打标记

docker tag busybox:latest 192.168.1.129:5000/busybox:latest

进行上传

docker push 192.168.1.129:5000/busybox:latest

客户端使用

编写配置文件

vim /etc/docker/daemon.json

{

"insecure-registries" :["192.168.1.129:5000"]

}

重启服务

systemctl restart docker

启动容器，指定IP地址和端口，第一次要进行下载操作

docker run -it 192.168.1.129:5000/busybox:latest

查看私有仓库镜像

curl 192.168.1.129:5000/v2/\_catalog

现实镜像有几个标签

curl 192.168.1.129:5000/v2/busybox/tags/list

持久化存储

docker容器不保持任何数据，重要的数据请使用外部存储（数据持久化），容器可以挂载在真实机目录或共享存储为卷

主机卷映射：将真实机目录挂在到容器中提供持久化存储，-v后头冒号前头是本机目录，冒号后头是是容器内的目录，意思是用真机目录代替容器内目录

docker run -v /data:/data -it centos /bin/bash

docker网络架构

查看默认docker创建的网络模型docker network list

使用docker创建虚拟交换机

docker network create --driver bridge test01

docker network list

ip a

docker network inspect test01

创建指定网段的交换机

docker network create --subnet=192.168.100.0/24 test01

使用自定义网桥

docker run --network=test01 -it centos /bin/bash

客户端访问容器内的资源

默认容器可以访问外网，但是外部主机不能访问内网，所以需要端口映射，将物理机端口映射为容器端口

docker run -p 80:80 -id nginx

1. ansible批量主机管理

七种武器（工具）

第一种：ansible命令，用于执行临时性的工作，必须掌握

第二种：ansible-doc是ansible模块的文档说明，针对每个模块都有详细的用法说明及应用实例介绍，功能和linux的man类似，必须掌握

第三种：ansible-console是ansible为用户提供的一款交互式工具，用户可以在ansible-console虚拟出来的终端上像shell一样使用ansible内置的各种命令，这种习惯于使用shell交互方式的用户提供了良好的使用体验

第四种：ansible-galaxy从github上下载管理roles的一款工具，与python的pip类似

第五种：ansible-playbook是日常应用中使用频率最高的命令，其工作机制是：通过预先编写好的playbook文件实现批量管理。要实现的功能与命令ansible一样，可以理解为按一定条件组成的ansible任务集，必须掌握

第六种：ansible-vault主要用于配置文件加密，如编写的playbokk配置文件中包含敏感词汇，不希望别人随便看，这个可以加密和解密配置文件

第七种：anbsible-pull，有两种工作模式pull/push，默认使用push模式，主动管理被托管主机，pull模式是被托管主机定期访问管理主机，适用于有数量巨大的机器需要配置的情况下

环境准备

部署六台服务器（一个管理，五个托管）

管理主机：主机上配置ip与主机名对应关系/etc/hosts

免密码登录：管理主机可以登录托管主机无需密码

ssh-keygen

for i in web1 web2 db1 db2 cache;do ssh-copy-id -i id\_rsa.pub ${i};done

简介

ansible可以批量管理主机，基于python开发，实现了批量操作系统配置、批量程序部署、批量运行命令等功能，可以实现：自动部署APP、自动化管理配置项、自动化的交互使用、自动化（AWS）云服务管理

特性：

模块化设计，调用特定的模块来完成特定的任务

基于python语言实现

paramiko

pyYAML（半结构化语言）

jinja2

其模块支持JSON等标准输出格式，可采用任何编程语言重写

部署简单，仅用在管理端安装软件

主从模式工作

支持自定义模块

支持异构IT环境

工作流程：

程序读取配置文件，可以从多个静态文件，文件夹，脚本中读取机器，分组及其相关变量信息，读取机器列表，根据参数确定执行模块和配置，running执行返回，输出结果结束

ansible安装

软件依赖关系

对管理主机的要求

要求python2.6或python2.7

ansible使用了以下模块，都需要安装paramiko、PyYAML、jinja2、httplib2、six

被托管的主机

ansible默认通过SSH协议管理主机

被管理主机要开启SSH服务，允许ansible主机登录

在托管节点上也需要安装python2.5或以上版本

如果托管节点上启用了selinux，需要安装libselinux-python

安装ansible

ansible可以基于源码运行

源码安装

pip，需要安装扩展包源extras

git

yum -y install epel-release

yum -y install git python2-pip

pip安装依赖模块

pip install paramiko PyYAML jinja2 httplib2 six

源码下载

ansible源码下载

git clone git://github.com/ansible/ansible.git

yum install python-setuptools python-devel

python setup,py build

python setup.py install

pip方式安装

pip install ansible

yum扩展源安装简单，自动解决依赖关系

vim /etc/yum.repos.d/rhel7.repo

[ansible]

name=ansible

baseurl=http://mirror.centos.org/centos-7/7/extras/x86\_64/

enabled=1

gpgcheck=0

yum repolist

yum -y install ansible

安装完后验证

ansible --version

ad-hoc（执行临时的操作）

主机管理

主机定义与分组

ansible配置文件查找顺序

首先检测ANSIBLE\_CONFIG变量定义的配置文件

ansible配置文件查找顺序

首先检测ANSIBLE\_CONFIG变量定义的配置文件

其次检测当前目录下的./ansible.cfg文件

再次检测当前用户家目录下~/ansible.cfg文件

最后检查/etc/ansible/ansible.cfg文件

/etc/ansible/ansible.cfg默认配置文件路径

inventory是定义托管主机地址配置文件

inventory参数说明

ansible\_ssh\_host将要连接的远程主机名。与想要设定的主机的别名不同的话，可通过此变量设置

ansible\_ssh\_port ssh端口号，通过此变量设置

ansible\_ssh\_user默认的ssh用户名

ansible\_ssh\_pass ssh密码（这种方式并不安全，建议使用--ask-pass或SSH密码）

ansible\_sudo\_pass sudo密码（建议使用--ask-sudo-pass）

ansible\_sudo\_exe（new in version1.8）sudo命令路径（适用于1.8及以上版本）

ansible\_connection与主机连接的类型。比如:local、ssh或者paramiko。ansible1.2以前默认使用paramiko.1.2，以后默认使用’smart’,’smart’方式会提示是否支持control[ersist,来判断’ssh’方式

ansible\_ssh\_private\_key\_file ssh使用的私钥文件。适用于多个密钥，而不想使用SSH代理的情况

ansible\_python\_type目标系统的shell类型，默认情况下，命令的执行使用sh语法，可设置为csh或fish

ansible\_python\_interpreter目标主机的python路径，适用于系统中有多个python，或者命令路径不是/usr/bin/python

首先编辑/etc/ansible/hosts文件，写入一些远程主机地址，格式如下

[组名]

主机名称或ip地址，登录用户名，密码、端口等信息

[web]

web1

web2

[db]

db[1..2]

[cache]

192.168.1.10

[other]

cache ansible\_ssh\_user=”root” ansible\_ssh\_pass=”123456” ansible\_ssh\_port=22

[other:vars]（对一组进行配置）

ansible\_ssh\_user=”root”

ansible\_ssh\_pass=”123456”

[app:children]（字组定义，调用上面分组的主机）

web

db

测试：ansible [组名称] --list-hosts

自定义配置文件

创建文件夹myansible

创建配置文件ansible.cfg

[defaults]

inventory=myhost

配置主机文件

vim ./myhost

[nginx]

192.168.1.11

ansible nginx --list-hosts

动态主机（通过脚本调用）

ansible的inventory文件可以是精通也可以是脚本（输出格式为JSON），修改ansible.cfg文件指定脚本inverntory=script

注意事项：

1. 主机部分必须是列表格式的
2. hostdata行，其中”hosts”部分可以忽略，但如果使用时，必须是hosts
3. python和shell脚本都可以

事例：

vim /etc/ansible/ansible.cfg

inventory = /etc/ansible/hosts.sh

vim /etc/ansible/hosts.sh

#!/bin/bash

echo '

{

"web" :{

"hosts" :["web1","web2"],

"vars" :{

"ansible\_ssh\_user":"root",

"ansible\_ssh\_pass":"123456"

}

},

"db" :["db1","db2"],

"cache":{"ansible\_ssh\_user":"root","ansible\_ssh\_pass":"123456"}

}'

批量执行

ansible命令基础

ansible <host-parttern> [options]

host-pattern 主机或定义的分组

-M指定模块路径

-m使用模块，默认command模块

-a or --args模块参数

-i inventory文件路径或可执行脚本

-k使用交互式登录密码

-e定义变量-v详细信息

ansible all --list-hosts列出要执行的主机，不执行任何操作

ansible all -m ping批量检测主机

例子：ansible all -m command -a ‘id’ -k

批量部署ssh证书文件

ansible all -m authorized\_key -a "user=root exclusive=true manage\_dir=true key='$(</root/.ssh/id\_rsa.pub)'" -k

批量配置管理

模块

ansible-doc模块的手册，相当于shell的man

ansible-doc -l列出所有模块

ansible-doc modulename查看模块帮助

ping模块

测试网络连通性，ping模块没有参数

ansible host-pattern -m ping

command模块

默认模块，远程执行命令，用法：

ansible host-pattern -m command -a ‘[args]’

查看所有机器的负载

ansible all -m command -a ‘uptime’

注意事项：该模块通过-a跟上要执行的命令可以直接执行，不过命令里有bash的命令则执行不成功（<>|&）该模块不启动shell直接在ssh进程中执行，所有使用到shell特性的命令都会失败

shell | raw模块

shell模块用法基本和command一样，区别是shell模块是通过/bin/sh进行执行命令，可以执行任何命令

raw模块，用法和shell模块一样，可以执行任何命令，区别是raw没有chdir、creates、removes参数

ansible web -m shell -a ‘chdir=/tmp touch f3’

script模块

复杂命令直接在本地写脚本，然后使用script模块批量执行

ansible web -m script -a ‘/etc/aaa.sh’

提示：该脚本包含但不限于shell脚本

copy模块（适合同步配置文件）

复制文件到远程主机

src：要复制到远程主机的文件在本地的地址，可以是绝对路径，也可以是相对路径。如果路径是一个目录，它将递归复制。在这种情况下，如果路径使用”/”来结尾，则只复制目录里的内容，如果没有，则包含目录在内的整个内容全部复制，类似于rsync

dest：必选项。远程主机的绝对路径，如果源文件是一个目录，那么该路径也必须是一个目录

backup：在覆盖之前将原文件备份，备份文件包含时间信息。有两个选项：Yes|no，默认不备份

force：如果目标主机包含该文件，但内容不同，设置为yes时，则强制覆盖，如果为no，则只有当目标主机的目标位置不存在该文件时，才复制。默认为yes

复制文件：ansible t1 -m copy -a ‘src=/root/alog dest=/root/a.log’

复制目录：ansible t1 -m copy -a ‘src=urdir dest=/root/’

lineinfile（把整行删了重写）|replace（只替换匹配的字串）模块

类似sed的一种行编辑替换模式

path目的文件 regexp正则表达式 line替换后的结果

ansible other -m lineinfule -a 'path=/etc/sysconfig/network-scripts/ifcfg-eth0 regexp="^ONBOOT" line="ONBOOT=no"'

ansible other -m replace -a 'path=/etc/sysconfig/network-scripts/ifcfg-eth0 regexp="^(ONBOOT=).\*" replace="\1yes"'

yum模块

使用yum包管理器来管理软件包

config\_file：yum的配置文件

disable\_gpg\_check：关闭gpg\_check

disablerepo：不启用某个源

enablerepo：启用某个源

name：要进行操作的软件包名字，也可以传递一个url或者一个本地的rpm包的路径

state：状态（installed、removed），默认为installed

删除软件包：ansible web -m yum -a 'name="lrzsz" state=removed'

删除多个软件包：ansible web -m yum -a 'name="lrzsz,lftp" state=removed'

安装软件包：ansible web -m yum -a 'name="lrzsz" state=installed'

安装多个软件包：ansible web -m yum -a 'name="lrzsz，lftp" state=installed'

service模块

name：必选项，服务名称

enabled：是否开机启动yes|no

sleep：如果执行了restarted，在stop和start之间沉睡几秒

state：对当前服务执行启动，停止，重启，重新加载等操作（started，stopped,restarted,reloaded）

ansible other -m service -a 'name="ntpd" state="stopped"'

setup模块

主要用来获取主机信息。setup模块下经常使用的一个参数是filter参数

filter可以过滤到我们需要的信息，不加参数则过滤出所有信息

ansible web -m setup -a 'filter=ansible\_distribution'

playbook

JSON：

是javascript对象表示法，它是一种基于文本，独立于语言的轻量级数据交换格式。JSON中的分隔符限于单引号、小括号、中括号、大括号、冒号和逗号。特性是纯文本，数据由逗号分隔，大括号保存对象，中括号保存数组。书写格式是：名称/值对，字段名称在引号中，后面写一个冒号，然后是值，如：

{“讲师”:

[“aa”:”小逗比”,”负责阶段”:”1”]

[“bb”:”老逗比”,”负责阶段”:”2”]

}

YAML：

是一个可读性高，用来表达数据序列的格式，参考了其他多种语言，包括：C语言、python、perl，并从XML、电子邮件中获得灵感，目前数种编程语言或脚本语言支持解析这种语言，基本语法：YAML的结构通过空格来表示，数组用-来表示（短横线后面有一个空格），键值用:表示（冒号后面有一个空格），YAML使用一个固定的缩进风格表示数据层级结构关系（空格加缩进），一般每个缩进级别由两个以上的空格组成（不准使用tab键），# 表示注释（井号后面有空格）。YAML的键值表示方法：采用冒号分隔，冒号后面必须由一个空格，例如：

“讲师”

-

“aa”: “小逗比”

“阶段”: 1

-

“aaa”: “11”

jinja2模版

jinja2是基于python的模版引擎，包含变量和表达式两部分，这两者在模版求职的时候会被替换为值。模版中海油标签，控制模版的逻辑。playbook模版使用jinja2模版。基本语法：模版的表达式都是包含在分隔符{{ }}内的，控制语句都是包含在分隔符{% %}内的，另外，模版也支持注释，都包含在分隔符{# #}内，支持注释块，例如：

{{变量}}//调用变量

{%if name == ‘小逗比’%}

讲故事

{%elif name == ‘老逗比’%}

黑丝

{%else %}

沉迷学习，无法自拔

{%endif%}

jinja2过滤器

变量可以通过过滤器修改。过滤器与变量用管道符号分割，并且也可以用圆括号传递可选参数。多个过滤器可以链式调用，前一个过滤器的输出作为后一个过滤器的输入，例如：

{{list|join(‘,’)}}

playbook基础

playbook是ansible用于配置，部署和管理托管主机的脚本。

语法格式：由YAML语言编写，遵循YAML标准，在同一行中，#号后面的内容表示注释，同一列表中的元素应该保持相同的缩进，playbook由一个或多个play组成，play中的hosts,variables,roles,tasks等对象的表示方法都是键值中以”:”分隔表示，所有的YAML文件开始行都应该是---。这是YAML格式的一部分，表示一个文件的开始

playbook构成：hosts：定义将要执行playbook的远程主机，内容是一个或多个主机，以帐号分隔

remote\_user：是远程登录的用户名

var：定义playhbook运行时使用的变量

tasks：定义将要在远程主机上执行的任务列表，一个task在其所对应的所有足迹上执行完毕后，下一个task才会执行

handler：定义task执行完以后需要调用的任务

---

- host: web

remote\_user: root

tasks:

- user: name=plj

- shell: echo 123456|passwd --stdin plj

playbook执行结果：使用ansible-playbook运行playbook文件，得到输出内容为JSON格式。并且由不同颜色组成，便于识别。绿色代表成功，\*\*\*代表系统状态发生改变，红色表示执行四百。执行时-f设置并发进程数量，默认是5，不写也可以。

playbook进阶

变量

添加用户

给所有主机添加用户plj，默认设置密码123456

---

- hosts: web

remote\_user: root

vars:

un: nb

tasks:

- user:

name: “{{un}}”

group: users

password: ‘{{‘123456’|password\_hash(‘sha512’)}}’

- name: aaa

shell: chage -d 0 ‘{{}}’

error

ansible-playbook对错误的处理

默认情况判断$?，如果值不为0就停止

但某些情况需要忽略错误继续执行

加上ignore\_error: true

handlers

用于当关注的资源发生变化时采取一定的操作

“notify”这个action可用在每个play的最后被触发这样可以避免多次有改变时每次都执行指定的操作而代之仅在发生变化完成后一次性地执行指定操作

---

- hosts: all

remote\_user: root

tasks:

- name: config httpd.conf

copy: src=/root/httpd.conf dest=/etc/httpd/conf/httpd.conf

notify:

- restart httpd

handles:

- name: restart httpd

service:

name: httpd

state: restarted

注意：notify调用的是handler段name定义的串，必须一致，否则达不到触发的效果

执行完tasks里才会执行handlers，在tasks里有调用才会执行handles，不管调用几次，只执行一次，notify是调用

多个task触发同一notify时，同一服务只会触发一次

notify可以触发多个条件，在生产环境中往往涉及到某一配置文件的改变需要重启若干服务的场景，handler用到这里非常合适，结合vars可以写出非常合适的服务管理脚本

when

某些时候肯定需要在满足特定的条件下终止某个行为，这个时候就需要进行条件判断when: 条件

register

有时候需要根据结果处理后面的操作，这时候就需要register模块来保存前一个命令的返回状态，在后面进行调用

---

- hosts: web

remote\_user: root

tasks:

- shell: uptime | awk ‘{printd(‘%.2f’,$(NF-2))}’

register: aaa

- service: name=httpd state=stopped

when: result.stfout|float > 0.7

with\_items

with\_items是playbook标准循环，最常用到的就是它，with\_items可以用于迭代一个列表或字典，通过{{item}}获取每次迭代的值

---

- hosts: web2

remote\_user: root

tasks:

- user:

name: "{{item.name}}"

group: "{{item.group}}"

password: "{{'123456'|password\_hash('sha512')}}"

with\_items:

- {name: "nb", group: "root"}

- {name: "dd", group: "mail"}

- {name: "jj", group: "wheel"}

- {name: "lx", group: "root"}

with\_nested

嵌套循环

---

- hosts: cache

remote\_user: root

vars:

un: [a, b, c]

id: [1, 2, 3]

tasks:

- name: add users

shell: echo {{item}}

with\_nested:

- "{{un}}"

- "{{id}}"

tags

tags：给指定的任务定义一个调用标识;

使用格式：

name:aaa

module:arguments

tags:bbb

调用方式：

ansible-playbook --tags=bbb apache.yml

include and roles

在编写playbook的时候随着项目越来越大，playbook也越来越复杂，修改起来也越来越麻烦。这时候可以把一些plau、task或handler放到其他文件中，然后通过Include指令包含进来是一个不错的选择include:tasks/setup.yml

role像是加强版的Include，它可以引入一个项目的文件和目录，一般所需的目录层级有，加入引入了一个role，则都会把该目录底下所有的文件都添加到这个play

vars变量层

tasks任务层

handlers触发条件

template模版

defaults默认，优先级最低

调试debug

几种简单的排错语法：

检测语法

ansible-playbook --synta-check playbooki.yaml

测试运行（报错，有时候是正常现象，因为是模拟运行，不是最真实运行）

ansible-playbook -C playbook.yaml

显示收到文件的主机--list-hosts

显示工作的task --list-tasks

显示将要运行的tag --list-tags

debug模块可以输出更为详细的信息，帮助排错，例如

---

- hosts: web

remote\_user: root

tasks:

- shell: uptime | awk ‘{printd(‘%.2f’,$(NF-2))}’

register: aaa

- service: name=httpd state=stopped

when: result.stfout|float > 0.7

- name: show debug info

debug: var=result

1. 分布式ELK（分析日志）平台

ELK不是一个软件，是一套解决方案，是三个软件产品的首字母缩写

Elasticsearch：负责日志检索和存储，是一个数据库

logstash：负责日志的收集和分析、处理

kibana：负责日志的可视化，相当于web软件

ELK组件在海量系统的运维中，可用于解决：

分布式日志数量集中式查询和管理

系统监控，包含硬件喝应用各个组件的监控

故障排查

安全信息和时间管理

报表信息

elasticsearch（数据库）

elasticsearch是一个基于lucene的搜索服务器，它提供了一个分布式多用户能力的全文搜索引擎，基于RESTfulAPI的web接口。是用java开发的，并作为apache许可条件下的开放源码发布，是当前流行的企业级搜索引擎

主要特点：实时分析

分布式实时文件存储，并将每一个字段都编入索引

文档导向，所有的对象全部是文档

高可用性，易扩展，支持集群、分片和复制

接口友好，支持JSON

没有典型意义的事务

是一种面向文档的数据库

没有提供授权和认证特性

相关概念：node：装有一个ES服务器的节点

cluster：有多个node组成的集群

document：一个可被搜索的基础信息单元，相当于数据库中的行

index：拥有相似特征的文档的集合，相当于数据库中的库

type：一个索引中可以定义一种或多种类型，相当于数据库种的表

filed：一个索引中可以定义一种或多种类型，相当于数据库中的字段

shards：索引的分片，每个分片就是一个shard

replicas：索引的拷贝

ES集群搭建

安装单台ES服务器

设置主机名称和ip对应关系

/etc/hosts

192.168.4.11 es1

解决依赖关系，配置yum源，安装jdk和elasticsearch

elasticsearch要求至少java7，一般推荐使用openjdk1.8

mv accounts.json.gz accounts.json.gz elasticsearch-2.3.4.rpm filebeat-1.2.3-x86\_64.rpm kibana-4.5.2-1.x86\_64.rpm logstash-2.3.4-1.noarch.rpm /var/ftp/public

cd /var/ftp/public

createrepo --update .

yum -y install java-1.8.0-openjdk elasticsearch

修改配置文件

vim /etc/elasticsearch/elasticsearch.yml

network.host:0.0.0.0

启动服务

systemctl enable elasticsearch

systemctl start elasticsearch

检查服务

netstat -tunlp | grep 9200

netstat -tunlp | grep 9300

curl 192.168.4.11:9200

ES集群配置

ES集群配置也很简单，只需要对配置文件做少量的修改即可，其他步骤和单击完全一致

安装5台数据库节点，集群中的所有节点要相互能够ping通，要在所有集群主机上配置/etc/hosts中的主机名与ip对应关系

192.168.1.11 es1

192.168.1.12 es2

192.168.1.13 es3

192.168.1.14 es4

192.168.1.15 es5

集群中所有机器都要安装java环境和elasticsearch

yum -y install elasticsearch java-1.8.0-openjdk

修改配置文件

cluster.name: 集群名字（集群名称要求完全一致）

node.name: 本机主机名称（为当前节点标识，应配置本机的主机名）

network.host: 0.0.0.0

discovery.zen.ping.unicast.hosts: [“es1”, “es2”, “es3”]（为集群节点信息，不需要全部配置，启动时，会查找这些主机，定义的先启动，否则查找不到，会自定义为一个集群）

配置完成后启动所有节点服务（有可能有一定的延迟，需要等待几十秒）

systemctl start elasticsearch

验证集群，使用ES内置字段\_cluster/health?pretty

curl 192.168.1.11:9200/\_cluster/health?pretty

返回字段：

status”:”green”集群状态，绿色为正常，黄色表示有问题但不是很严重，红色表示严重故障

“number\_of\_nodes”:5,表示集群中节点的数量

"number\_of\_data\_nodes" : 5,

…… …… ……

"task\_max\_waiting\_in\_queue\_millis" : 0,

"active\_shards\_percent\_as\_number" : 100.0

ES插件的安装和使用

ES常用插件

head插件：它展现ES集群的拓扑结构，并可以通过它来进行索引和节点级别的操作。它提供了一组针对集群的查询API，并将结果以JSON和表格的形式返回。它提供一些快捷菜单，用以展现集群的各种状态

kopf插件：是一个elasticsearch的管理工具。它提供了对ES集群操作的API

bigdesk插件：是elasticsearch的一个集群监控工具，可以通过它来查看es集群的各种状态，如：CPI、内存、使用情况、索引数据、搜索情况等

ES插件安装、查看

查看安装的插件

/usr/share/elasticsearch/bin/plugin list

安装插件

/usr/share/elasticsearch/bin/plugin install [ftp://192.168.1.254/public/elasticsearch-head-master.zip](ftp://192.168.1.254/elasticsearch-head-master.zip)

192.168.1.12:9200/\_plugin/head

/usr/share/elasticsearch/bin/plugin install <ftp://192.168.1.254/public/elasticsearch-kopf-master.zip>

192.168.1.11:9200/\_plugin/kopf

/usr/share/elasticsearch/bin/plugin install <ftp://192.168.1.254/public/bigdesk-master.zip>

192.168.1.11:9200/\_plugin/bigdesk

这里必须以url的方式进行安装，如果文件在本地，也需要使用file://的方式指定路径，删除使用remove指令

HTTP与RESTful API

RESTful API调用

elsticsearch提供了一系列RESTful的API

检查集群、节点、索引的健康度、状态和统计

管理集群、节点、索引的数据及元数据

对索引进程CRUD操作及查询

执行其他高级操作如分页、排序、过滤等

POST或PUT数据使用json格式

简单使用

\_cat API查询集群状态，节点信息，v参数显示详细信息，help显示帮助信息

<http://192.168.1.11:9200/_cat/health?v>

<http://192.168.1.11:9200/_cat/health?help>

<http://192.168.1.11:9200/_cat/node?v>

<http://192.168.1.11:9200/_cat/indices?v>

API增加索引

创建一个索引，并设置分片数量与副本数量

curl -XPUT 'http://192.168.1.11:9200/aaa' -d '{

"settings":{

"index":{

"number\_of\_shards":5,

"number\_of\_replicas":1

}

}

}'

API插入数据

curl -XPUT 'http://192.168.1.11:9200/aaa/teacher/1' -d '{

"title":"阶段1",

"name":{"first":"老逗比","last":"丁丁"},

"age":52

}'

API更改数据

curl -XPOST 'http://192.168.1.11:9200/aaa/teacher/1/\_update' -d '{

"doc":{ "age":18 }

}'

API查询数据

curl -XGET 'http:192.168.1.11:9200/aaa/1?pretty'

API删除数据

curl -XDELETE 'http:192.168.1.11:9200/aaa/1?pretty'

kibana安装（数据可视化平台，相当于web浏览器）

特点：

灵活的分析和可视化平台

实时总结和流数据图标

为不同的用户显示直观的界面

即时分享和嵌入的仪表盘

kibana安装

配置hosts，因为要能够访问其他主机

scp /etc/hosts 192.168.1.16:/etc/hosts

安装kibana

yum -y install kibana

修改配置文件

默认安装在/opt/kibana下面，配置文件在/opt/kibana/config/kibana.yml，只需做少量的修改就可以启动,除了修改集群地址，其他默认即可

vim /opt/kibana/config/kibana.yml

server.port: 5601

server.host: "0.0.0.0"

elasticsearch.url: "http://es1:9200"

kibana.index: ".kibana"

kibana.defaultAppId: "discover"

elasticsearch.pingTimeout: 1500

elasticsearch.requestTimeout: 30000

elasticsearch.startupTimeout: 5000

启动并设置开机启动

systemctl enable kibana

systemctl start kibana

访问kibana

<http://192.168.1.16:5601>

查看status状态为green即可

数据导入

使用\_bulk批量导入，批量导入使用POST方式，数据格式为json,url编码使用data-binary

导入含有index配置的json文件：curl -XPOST <http://192.168.1.148:9200/_bulk> --data-binary @shakespeare.json

导入没有index配置的json文件，需要在uri里面指定index和type：curl -XPOST [http://192.168.1.148:9200/oo/xx/\_bulk](http://192.168.1.148:9200/_bulk) --data-binary @shakespeare.json

kibana使用

kibana里选择日志，支持通配符\*

1. 在设置里面选择索引名和文档名
2. 导入成功以后选择discover
3. 选择导入的时间
4. 数据以柱状图的形式显示出来
5. 也可以选择其他图，点击visualize进行创建
6. 保存后，可以在dashboard查看

logstash

logstash是一个数据采集、加工处理一起的传输工具

logstash特点：

所有类型的数据集中处理

不同模式和格式数据的正常化

自定义日志格式的迅速扩展

为定义数据源轻松添加插件

logstash安装

logstash依赖java环境，需要安装java-1.8.0-openjdk

logstash没有默认的配置文件，需要手动配置

logstash安装在/opt/logstash目录下

配置hosts，因为要与elasticsearch相连，所以要配置hosts，可以不写kibana,因为不与kibana相连

vim /etc/hosts

192.168.1.11 es1

192.168.1.12 es2

192.168.1.13 es3

192.168.1.14 es4

192.168.1.15 es5

安装相应软件包

yum -y install java-1.8.0-openjdk logstash

logstash工作结构

数据源发送到logstash，logstash内部input接受数据，然后给了filter进行处理，output存储到ES集群

logstash里面的类型

布尔值类型：ssl\_enable=>true

字节类型：bytes=>”1Mib”

字符串类型：name=>”xkops”

数值类型：port=>22

数组：match=>[“datetime”,”unix”]

哈希：option=>[k=>”v”,k2=>”v2”]

编码解码：codec=>”json”

路径：file\_path=>”/tmp/filename”

注释：#

logstash条件判断

等于：= =

不等于：!=

小于：<

大于：>

小于等于：<=

大于等于：>=

匹配正则：=~

不匹配正则：!~

包含：in

不包含：not in

与：and

或：or

非与：nand

非或：xor

符合表达式：()

取反符合：!()

logstash配置文件

/etc/logstash/logstash.conf

input{

stdin{}

}

filter

output{

stdout{}

}

logstash插件

在配置文件中使用了logstash-output-stdout和logstash-input-stdin两个插件，logstash还有filter和codec类插件，查看插件的方式是

/opt/logstash/bin/logstash-plugin list

<https://github.com/logstash-plugins插件使用详情地址>

codec类插件（三个里头都可用）

常用的插件：plain、json、json\_lines、rubydebug、multline等

input{

stdin{codec=>”json”}

}

filter{}

output{

stdout{codec=>”rubydebug”}

}

input file插件

file{

start\_position=>”beginning”

sincedb\_path=>”/var/lib/logstash/sicedb-access”

path=>[”/tmp/a.log”,”/tmp/b.log”]

type=>”filelog”

}

sincedb\_path记录读取文件的位置

stfart\_position配置第一次读取文件从什么地方开始 ，在没有指定读取文件的时候

path指从哪里读入

type打个标记，说明从哪种类型文件读入的

input tcp&udp插件

input{

tcp{

host=>”0.0.0.0”

port=>”8888”

type=>”tcplog”

mode=>”server”

}

udp{

port=>”7777”

type=>”udplog”

}

}

syslog插件

input{

syslog{

host=>”0.0.0.0”

port=>514

type=>”syslog”

}

}

filter grok插件

解析各种非结构化的日志数据插件

groke使用正则表达式把非结构化的数据结构化

在分组匹配，正则表达式需要根据具体数据结构编写

虽然编写困难，但适用性极广

几乎可以应用于各类数据

groke分组匹配

正则匹配(?<name>reg)

filter{

groke{

match=>[”meeage”,”(?<cliend\_ip>[a-z]+)”]

}

}

使用正则宏

match=>[“message”,”%{IP:client\_ip}”]

冒号前为调用的宏，冒号后为自定义名字

正则宏路径：/opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-2.0.5/patters

output ES插件

调用成功后，把数据写入ES集群

output{

if [type] ==”apache.log”{

elasticsearch{

hosts=>[“192.168.1.11:9200”,”192.168.1.15:9200”]

index=>”apachelog”

flush\_size=>2000

idle\_flush\_time=>10

}

}

}

input filebeat插件

这个插件主要作用是接受beats类软件发送过来的数据，由于logstash依赖java环境，而且占用资源非常大，启动慢，往往不希望所有集群的机器都部署java环境安装logstash，而是使用更轻量的filebeate代替

logstash配置文件

beats{

port=>5044

codec=>”json”

}

安装和配置filebeat

yum -y install filebeat

修改配置文件/etc/filebeat/filebeat.yml

filebeat:

prospectors:

-

paths:

- /var/log/httpd/access\_log

input\_type: log

document\_type: apache.log

registry\_file: /var/lib/filebeat/registry

output:

logstash:

hosts: ["192.168.1.16:5044"]

shipper:

logging:

files:

rotateeverybytes: 10485760 # = 10MB

启动并设置开机运行

systemctl enable filebeat

systemctl start filebeat

1. hadoop（一种分析和处理海量数据的软件平台，是用java开发的开源软件，提供一个分布式基础架构）

hadoop核心组件

HDFS：分布式文件系统

mapreduce：分布式计算框架

yarn：集群资源管理系统

hadoop常用组件

HDFS（hadoop分布式文件系统）

Mapreduce（分布式计算框架）

zookeeper（分布式协作服务）

hbase（分布式列存数据库）

hive（基于hadoop的数据仓库）

sqoop（数据同步工具）

pig（基于hadoop的数据流系统）

mahout（数据挖掘算法库）

Flume（日志收集工具）

HDFS结构（分布式文件系统）

是hadoop体系中数据存储管理的基础。它是一个高容错的系统，用于在低成本的通用硬件上运行

角色

client、namenode、secondarynode、datanode

client将数据切片发送给namenode，namenode存储到datanode节点上，并写入日志fsimage和fsedit，secondary namenode定期将两个日志进行整理，合并后发送给namenode

mapreduce结构（分布式计算框架）

角色

jobtracker、tasktracker、map task、reducer task

客户端发送请求给jobtracker，jobtracker分配任务给tasktracker，tasktracker启动map task和reduce task执行，执行完的结果返回给jobtracker，jobtracker再返回给客户端

yarn结构（集群资源管理）

角色

resourcemanager、nodemanager、applicationmaster、container、client

客户端发送请求给resourcemanager，resourcemanager将客户请求发送给nodemanager并启动applicationmaster程序，nodemanager处理来自resourcemanager的命令，处理完成后逐层返回

hadoop安装配置

部署的三种模式：单机、伪分布式（与完全分布式类似，只不过把所有组件安装在一台机器上，适用于学习和测试环境）、完全分布式

单机：

1. 配置hostname和hosts

echo node1 > /etc/hostname

echo ‘192.168.1.11 node1’ >> /etc/hosts

1. 安装Java环境，安装jps工具

yum -y install java-1.8.0-openjdk-devel java-1.8.0-openjdk

1. 安装hadoop

<http://hadoop.apache.org获取软件>

yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel

tar -xf hadoop-2.7.6.tar.gz

mv hadoop /usr/local/hadoop

1. 设置环境变量，启动服务

rpm -ql java-1.8.0-openjdk

vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86\_64/jre/

export HADOOP\_CONF\_DIR=/usr/local/hadoop/etc/hadoop

1. 验证并测试

./bin/hadoop version

测试：统计词频

cd /usr/local/hadoop

mkdir aaa

cp \*.txt aaa

./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar wordcount aaa b

调用/bin/hadoop文件执行jar脚本，脚本是share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar，用来统计aaa目录里词的数量，放到b里面，b必须是不存在的目录，否则会报错

完全分布式：

hadoop最大的优势就是分布式集群计算，所以在生产环境下都是搭建最后一种模式：完全分布式

系统规划

主机 角色 软件

192.168.1.10nn01 namenode、secondarynamenode、resourcemanager HDFS、yarn

192.168.1.11node1 datenode、nodemanager HDFS、yarn

192.168.1.12node2 datenode、nodemanager HDFS、yarn

192.168.1.13node3 datenode、nodemanager HDFS、yarn

环境准备

开启4台服务器，注意：只开启必要服务，关闭其他无关的系统服务，系统最小化，服务最小化，禁用selinux，禁用firewalld

在所有主机上安装java环境和调试工具jps

注意：保证所有机器系统版本及java版本的一致性，保证所有安装路径的一致性

yum -y install java-1.8.0-openjdk-devel

配置主机名和ip对应关系/etc/hosts

注意：所有主机的配置都要修改，master要能ping通所有主机，node要能ping通master

vim /etc/hosts

192.168.1.11 nn01

192.168.1.12 node1

192.168.1.13 node2

192.168.1.14 node3

配置ssh信任关系（namenode）

注意：不能出现要求输入yes的情况，每台机器都要成功，包括本机

ssh\_config

StrictHostKeyChecking no

ssh-keygen -b 2048 -t rsa -N ‘’ -f key

ssh-copy-id -i ip地址或主机名

HDFS完全分布式系统配置

环境配置文件hadoop-env.sh

核心配置文件core-site.xml

HDFS配置文件hfds-site.xml

节点配置文件slaves

slaves

node1

node2

node3

hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86\_64/jre/

export HADOOP\_CONF\_DIR=/usr/local/hadoop/etc/hadoop

core-site.xml

fs.defaultFS文件系统配置参数

hadoop.tmp.dir数据目录配置参数

<property>

<name>fs.defaultFS</name>

<value>hdfs://192.168.4.11:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/var/hadoop</value>

</property>

hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.namenode.http-address</name>

<value>nn01:50070</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>nn01:50090</value>

</property>

配置完成以后，把hadoop的文件夹拷贝到所有机器

for i in node{1..3};do

scp -r /usr/local/hadoop/ $i:/usr/local

done

在namenode上执行格式化操作

./bin/hdfs namenode -format

在没有报错的情况下启动集群

./sbin/start-dfs.sh

停止集群可以用./sbin/stop-dfs.sh

验证

角色

启动后分别在namenode和datanode执行命令

jps

成功的情况下应该可以看见

namenode

secondarynode

datanode

集群是否组件成功

./bin/hadoop dfsadmin -report

服务器启动日志

/usr/local/hadoop/logs

安装部署mapred、yarn、web访问hadoop

分布式文件系统：HDFS已经部署完毕

分布式计算框架：mapreduce

集群资源管理：yarn

分布式计算框架mapred-site.xml

mv mapred-site.xml.template mapred-site.xml

vim mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

yarn资源管理安装与配置yarn-site.xml

resourcemanager管理地址yarn.resourcemanager.hostname

nodemanager使用哪个框架计算yarn.nodemanager.aux-services

yarn-site.xml

<property>

<name>yarn.resourcemanager.hostname</name>

<value>nn01</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

配置完成后，同步到所有主机

for i in node{1..3};do scp -r /usr/local/hadoop $i:/usr/local;done

启动yarn服务

./sbin/start-yarn.sh

./sbin/start-all.sh

在所有主机上执行jps，查看是否成功，是否有resourcemanager和nodemanager，至此，hadoop完全分布式集群搭建完毕

使用web访问hadoop

namenode web页面（nn01）

http://192.168.1.10:50070

secondory namenode web页面（nn01）

http://192.168.1.10:50090

datanode web页面（node1,node2,node3）

http://192.168.1.11:50075

resourcemanager web页面（nn01）

http://192.168.1.10:8088

nodemanager web页面（node1,node2,node3）

<http://192.168.1.11:8042>

HDFS基本使用

./bin/hadoop fs -ls /查看集群根目录内容

./bin/hadoop fs -mkdir /abc在集群根目录底下创建abc目录

./bin/hadoop fs -rmdir /abc删除目录abc

./bin/hadoop fs -touchz /urfile在集群根目录底下创建文件urfile

./bin/hadoop fs -cat /urfile查看文件内容

./bin/hadoop fs -rm /urfile删除文件

./bin/hadoop fs -put \*.txt /asd上传txt文件到asd目录

./bin/hadoop fs -get /asd下载文件

提交验证

创建目录./bin/hadoop fs -mkdir /input

导入要分析的文件./bin/hadoop fs -put \*.txt /input

提交分析作业（要加/才行，否则是单机分析）./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar wordcount /input /output

查看结果./binhadoop fs -cat /output/\*

HDFS进阶应用

hadoop节点管理

HDFS增加节点

1. 新增一台主机，配置所有hadoop环境，包括主机名、ssh免密钥登录、禁用selinux、iptables、安装java环境

vim /etc/hosts

yum -y install java-1.8.0-openjdk-devel

hostname newnode

echo newnode > /etc/hostname

1. 修改namenode的slaves文件增加节点

vim slaves

node1

node2

node3

newnode

1. 把所有的配置文件复制到所有主机配置文件目录下
2. 在新节点启动datanode

./sbin/hadoop-daemon start datanode

1. 设置同步带宽，并同步数据（新节点）

./bin/hdfs dfsadmin -setBalancerBandwidth 67108864

./sbin/start-balancer.sh -threshold

1. 查看集群状态

./bin/hdfs fsadmin -report

HDFS删除节点

1. 配置namenode的hdfs-site.xml，增加dfs.hosts.exclude配置

<property>

<name>dfs.hosts.exclude</name>

<value>/usr/local/hadoop/etc/hadoop/exclude</value>

</property>

1. 增加exclude配置文件，写入要删除的节点主机名

vim /usr/local/hadoop/etc/hadoop/exclude

newnode

1. 管理节点更新数据

./bin/hdfs dfsadmin -refreshNodes

1. 删除节点状态

查看状态./bin/hdfs dfsadmin -report

normal正常状态

decommissioned in program数据正在迁移

decommissioned数据迁移完成

注意：只有当状态变成decommissioned，才能宕机下线

宕机下线（要删除的节点）：./sbin/hadoop-daemon.sh stop datanode

HDFS修复节点

单独配置一台新的datanode，主机名必须跟原来的一致

启动服务./sbin/hadoop-daemon.sh start datanode

数据恢复是自动的，上线以后会自动恢复数据，如果数据流巨大，可能需要一段时间

yarn的相关操作

增加节点：./sbin/yarn-daemon.sh start nodemanger

删除节点：./sbin/yarn-daemon.sh stop nodemanager

查看节点（resourcemanager）：./bin/yarn node-list

yarn的系统环境与hdfs的基础环境相同，这里就不重复列出了，由于yarn不包含数据，所以在增加删除修复节点的时候比较简单，hdfs要注意数据安全

NFS网关

用途：

1. 用户可以通过操作系统兼容的本地NFSv3客户端来阅读HDFS文件系统（可以mount）
2. 用户可以从HDFS文件系统下载文档到本地文件系统
3. 用户可以通过挂载点直接流化数据。支持文件附加，但是不支持随机写

特性与注意事项：

1. 不支持随机写
2. 在非安全模式，运行网关的是代理用户
3. 在安全模式时，kerberos keytab中的用户是代理用户

配置：

1. 新建一个主机，配置hosts
2. 添加用户（nn01 nfsgw）

groupadd -g 200 nsd1803

useradd -u 200 -g 200 nsd1803

1. 配置core-site.xml（停止集群，在nn01上进行配置和启动）

增加、同步到所有主机、启动集群

hadoop.proxyuser.{代理用户}.groups

hadoop.proxyuser.{代理用户}.hosts

这里的{代理用户}是机器上真实运行nfs3的用户，在非安全模式，运行nfs网关的用户为代理用户，groups为挂载点用户所使用的组，hosts为挂载点主机地址

core-site.xml

<property>

<name>hadoop.proxyuser.nsd1803.groups</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.nsd1803.hosts</name>

<value>\*</value>

</property>

for i in node{1..3};do scp core-site.xml $i:/usr/local/hadoop/etc/hadoop

./sbin/start-all.sh

1. 在nfsgw上检查是否禁用selinux、禁用firewalld，卸载rpcbind、nfs-utils，安装openjdk，同步namenode的hadoop到本机
2. 配置hdfs-site.xml（新增配置即可）

<property>

<name>nfs.exports.allowed.hosts</name>

<value>\* rw</value>

</property>

值字符串为机器名和访问策略，通过空格来分割。机器名的格式可以是单一的主机，java的正则表达式或者是ipv4地址。访问权限使用rw或ro来指定导出目录的读/写或者机器只读访问。如果访问策略没被提供，默认为只读的。每个条目使用分号来分割。默认“ro”

<property>

<name>nfs.dump.dir</name>

<value>/tmp/nfstmp</value>

</property>

nfs.dump.dir用户需要更新文件转储目录参数，配置完成后要创建目录并把该文件夹的属性改为代理用户

1. 创建临时转储目录，并把该文件夹的属组改成代理用户

mkdir /var/nfstmp

chown -R nsd1803:nsd1803 /var/nfstmp

1. 创建数据根目录

mkdir /var/hadoop

1. 启动与挂载

启动

设置/usr/local/hadoop/logs权限，为代理用户赋予读写执行权限

setfacl -m u:nsd1803:rwx /usr/local/hadoop/logs

使用root启动portmap服务

./sbin/hadoop-daemon.sh --script ./bin/hdfs start portmap

使用代理用户启动nfs3

su - nsd1803

./sbin/hadoop-daemon.sh --script ./bin/hdfs start nfs3

挂载

目前NFS v3仅使用TCP作为传输协议。不支持NLM（锁），因此需要安装”nolock”。强烈建议使用安装选项sync，因为它可以最小化或避免重新排序写入，这将导致更可预测的吞吐量。未指定同步选项可能会导致上传大文件时出现不可靠的行为

挂载NFS参数

vers=3、proto=tcp、nolock、noacl、noatime、sync

yum -y install nfs-utils

mount -t nfs -o vers=3,proto=tcp,nolock,noacl,noatime,sync ip.xx.xx.xx:/ /localdir

1. 挂载完成后进行读写测试

zookeeper

zookeeper是一个分布式的，开放源码的分布式应用程序协调服务（协调执行顺序）

应用场景

集群分布式锁

集群统一命名服务

分布式协调服务

角色与特性

leader：接受所有follower提案请求并统一协调发起天的投票，负责域所有的follower进行内部交换数据（老板负责制定员工的工作方向）

follower：直接为客户端服务并参与提案，同时与leader进行数据交换（经理负责安排活和向老板汇报）

observer：直接为客户端服务但并不参与提案的投票，同时也与leader进行数据交换（员工直接干具体的活）

角色选举规则

集群中有n/2+1台服务器选举投票，如果leader死亡，会进行重新选举，如果宕机数达到一半，集群就会停止工作。Observer不计算在投票总设备数里面

ZK集群安装配置

1. 安装openjdk环境
2. 配置Hosts，需要相互之间可以ping通

192.168.1.18 nn01

192.168.1.10 node1

192.168.1.11 node2

192.168.1.13 node3

1. 解压创建配置文件

tar -xf zookeeper-3.4.10.tar.gz

mv zookeeper-3.4.10 /usr/local/zookeeper

cd /usr/local/zookeeper/conf

mv zoo\_sample.cfg zoo.cfg

1. 修改配置文件，设置集群机器id、ip、port（配置文件在最后一行后面添加）

server.1=node1:2888:3888

server.2=node2:2888:3888

server.3=node3:2888:3888

server.4=nn01:2888:3888:observer

前三个是loader和follower，最后一个是observer，端口从2888到3888

1. 拷贝文件到所有集群节点

for i in node{1..3};do

scp -r /usr/local/zookeeper/ $i:/usr/local

done

1. 创建目录和myid文件

[root@nn01~]mkdir /tmp/zookeeper

[root@nn01~]echo 4 > /tmp/zookeeper/myid

[root@node1~]mkdir /tmp/zookeeper

[root@node1~]echo 1> /tmp/zookeeper/myid

[root@node2~]mkdir /tmp/zookeeper

[root@node2~]echo 2 > /tmp/zookeeper/myid

[root@node3~]mkdir /tmp/zookeeper

[root@node3~]echo 3 > /tmp/zookeeper/myid

1. 启动服务

[root@nn01~]cd /usr/local/zookeeper

[root@nn01 zookeeper]./bin/zkServer.sh start

[root@node1~]cd /usr/local/zookeeper

[root@node1 zookeeper]./bin/zkServer.sh start

[root@node2~]cd /usr/local/zookeeper

[root@node2 zookeeper]./bin/zkServer.sh start

[root@node3~]cd /usr/local/zookeeper

[root@node3 zookeeper]./bin/zkServer.sh start

需要分别启动，启动后稍等几秒，等待票选结果，即可启动成功，但启动一台无法查看结果，只有全部启动后才能查看

1. 查看状态

/usr/local/zookeeper/bin/zkServer.sh status

kafka集群

kafka是一个分布式消息系统，是由scala编写，是一种消息中间件

作用：

解耦，冗余，提高可扩展性，缓冲，保证顺序，灵活，削峰填谷，异步通信

角色与集群结构：

producer：生产者，负责发布信息

consumeri：消费者，负责读取处理信息

topic：消息的类别

parition：每个topic包含一个或多个partition

broker：kafka集群包含一个或多个服务器

kafka通过zookeeper管理集群配置，选举leader

安装配置

kafka集群的安装配置是依赖zookeeper的，搭建kafka集群之前，创建好一个可用的zookeeper集群

1. 安装openjdk运行环境
2. 解压包，修改配置文件

tar -xf kafka\_2.10-0.10.2.1.tgz

mv kafka\_2.10-0.10.2.1 /usr/local/kafka

cd /usr/local/kafka/config

vim server.properties

broker.id=11每台主机不一样，随便设定

zookeeper.connect=node1:218,node2:2181,node3:2181

1. 分发kafka拷贝到所有集群主机

for i in node{1..3};do

scp -r /usr/local/kafka $i:/usr/local

done

1. 启动与验证

在所有主机启服务

/usr/local/kafka/bin/kafka-server-start.sh - daemon /usr/local/kafka/config/server.properties

验证

jps能看到kafka模块

netstat命令能看到9092在监听

1. 集群验证与消息发布

创建一个topic

./bin/kafka-topics.sh --create --partitions 2 --replication-factor 2 --zookeeper node3:2181 --topic mymsg

查看已经存在的topic

./bin/kafka-topics.sh --list --zookeeper node2:2181

查看刚刚创建的topic

./bin/kafka-topics.sh --describe --zookeeper node1:2181 --topic mymsg

在两个终端里面，生产者发布消息，消费者读取消息

生产者

./bin/kafka-console-producer.sh --broker-list master:9092,node1:9092 --topic mymsg

消费者

./bin/kafka-console-consumer.sh --bootstrap-server node2:9092,node3:9092 --topic mymsg

--from-beginning表示从开始读取消息

hadoop高可用

namenode和resourcenode高可用

1. 系统规划

192.168.1.18 nn01

192.168.1.19 nn02

192.168.1.10 node1

zookeeper

192.168.1.11 node2

zookeeper

192.168.1.13 node3

zookeeper

1. 安装相应软件openjdk和psmic

yum -y install psmic

1. 所有主机配置hosts

192.168.1.18 nn01

192.168.1.10 node1

192.168.1.11 node2

192.168.1.13 node3

192.168.1.19 nn02

4. 所有机器除了zookeeper其他hadoop,kafka服务全部停掉

5. 所有机器初始化hdfs集群，删除/var/hadoop/\*

rm -rf /var/hadoop/\*

6. nn01、nn02关闭ssh key验证，部署公钥私钥，需要可以免密钥登录其他主机和自己

vim /etc/ssh/ssh\_config

strictHostKeyChecking

ssh-keygen

ssh-copy-id nn01（nn02 node1 node2 node3）

7. 修改配置文件

hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86\_64/jre/

export HADOOP\_CONF\_DIR=/usr/local/hadoop/etc/hadoop

slaves

node1

node2

node3

mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

core-site.xml

<property>

<name>fs.defaultFS</name>

<value>hdfs://mycluster</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/var/hadoop</value>

</property>

<property>

<name>ha.zookeeper.quorum</name>

<value>node1:2181,node2:2181,node3:2181</value>

</property>

hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

secondartnamenode在高可用里面没有用途，这里把他关闭

namenode在后面定义

<!--指定hdfs的nameservices名称为mycluster>

<property>

<name>dfs.nameservices</name>

<value>mycluster</value>

</property>

指定集群的两个namenode的名称分别为nn1，nn2

<property>

<name>dfs.ha.namenodes.mycluster</name>

<value>nn1,nn2</value>

</property>

配置nn1,nn2的namenode

<property>

<name>dfs.namenode.rpc-address.mycluster.nn1</name>

<value>nn01:8020</value>

</property>

<property>

<name>dfs.namenode.rpc-address.mycluster.nn2</name>

<value>nn02:8020</value>

</property>

配置nn1,nn2的datanode

<property>

<name>dfs.namenode.http-address.mycluster.nn1</name>

<value>nn01:50070</value>

</property>

<property>

<name>dfs.namenode.http-address.mycluster.nn2</name>

<value>nn02:50070</value>

</property>

指定namenode JNS日志的路径

<property>

<name>dfs.namenode.shared.edits.dir</name>

<value>qjournal://node1:8485;node2:8485;node3:8485/mycluster</value>

</property>

指定JNS日志文件存储的路径

<property>

<name>dfs.journalnode.edits.dir</name>

<value>/var/hadoop/journal</value>

</property>

指定HDFS客户端连接active namenode的java类

<property>

<name>dfs.client.failover.proxy.provider.mycluster</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

配置隔离机制为ssh

<property>

<name>dfs.ha.fencing.methods</name>

<value>sshfence</value>

</property>

指定密钥的位置

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/root/.ssh/id\_rsa</value>

</property>

开启自动故障转移

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

yarn-site-xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.resourcemanager.ha.rm-ids</name>

<value>rm1,rm2</value>

</property>

<property>

<name>yarn.resourcemanager.recovery.enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.resourcemanager.store.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>

</property>

<property>

<name>yarn.resourcemanager.zk-address</name>

<value>node1:2181,node2:2181,node3:2181</value>

</property>

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>yarn-ha</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm1</name>

<value>nn01</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm2</name>

<value>nn02</value>

</property>

8. 同步数据

删除服务器日志

rm -rf /usr/local/hadoop/logs/\*

同步配置到所有主机

scp -r /usr/local/hadoop nn02:/usr/local

for i in node{1..3};do

scp -r /usr/local/hadoop $i:/usr/local

done

在其中一台初始化zookeeper集群（nn01上操作）

cd /usr/local/hadoop

bin/hdfs zkfc-formatZK

在定义的节点启动journalnode

cd /usr/local/hadoop

sbin/hadoop-daemon.sh start journalnode

所有节点都要启动（node1、node2、node3）

在其中一台namenode上执行格式化命令

cd /usr/local/hadoop

bin/hdfs namenode -format

scp -r /var/hadoop nn02:/var/

注意是格式化其中一台，格式化以后把数据目录拷贝到另一台（nn01格式化，nn02进行同步）

初始化journalnode（nn01）

cd /usr/local/hadoop

./bin/hdfs namenode -initializeSharedEdits

停止journalnode（node1 node2 node3）

cd /usr/local/hadoop

sbin/hadoop-daemon.sh stop journalnode

启动dfs（nn01上执行）

cd /usr/local/hadoop

./sbin/start-all.sh

启动resourcemanager（nn02上执行）

cd /usr/local/hadoop

sbin/start-yarn.sh start

1. 高可用验证

验证配置

cd /usr/local/hadoop

bin/hadoop dfsadmin -report

查看集群状态

cd /usr/local/hadoop

bin/hdfs haadmin -getServiceState nn1

bin/hdfs haadmin -getServiceState nn2

bin/yarn rmadmin -getServiceState rm1

bin/yarn rmadmin -getServiceState rm2

文件读写

cd /usr/local/hadoop

bin/hadoop fs -ls hdfs://mycluster/

bin/hadoop fs -mkdir hdfs://mycluster/input

bin/hadoop fs -put \*.txt hdfs://mycluster/input

bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar wordcount hdfs://mycluster/input hdfs://mycluster/output

namenode故障转移

关闭主namenode

cd /usr/local/hadoop

./sbin/hadoop-daemon.sh stop namenode

bin/hdfs haadmin -getServiceState nn1

bin/hdfs haadmin -getServiceState nn2

bin/hadoop fs -cat hdfs://mycluster/output/\*

resourcemanager故障转移

关闭主resourcemanager

cd /usr/local/hadoop

./sbin/hadoop-daemon.sh stop resourcemanager

bin/yarn rmadmin -getServiceState rm1

bin/yarn rmadmin -getServiceState rm2

1. apache bench安装与使用

apache自带的压力测试工具，简单易用，大并发的时候需要自己查询错误结果

yum intsall httpd-tools

ab [option] <http://url>

支持get post自定义头等

-c并发数量

-n请求总数

-H自定义请求头

-p post文件

1. web bench安装与使用

webbech是一个在linux下使用的非常简单的网站压测工具，它使用fork()模拟多个客户端同时访问我们设定的URL，测试网站在压力下工作的性能，最多可以模拟3万个并发连接去测试网站的负载能力

安装依赖包yum install ctags

make

使用

./webbench -r -t 60 -c 200 -2 <http://url>

-r 发送刷新缓存要求

-t 测试时间

-c 并发数量

-2 使用http1.1，长连接

-h帮助信息

1. wrk压力测试软件

优势是支持线程，能更好发挥多核的优势

安装也很简单，直接make即可

-c表示的是并发

-d测试持续的时间

-t来控制线程数（缺省值为2）

/wrk -c 200 -d 10 -t 5 http://url