# 云计算全阶段笔记

第一阶段课程内容共19天

云计算系统管理、 云计算应用管理、系统&服务管理进阶

云计算系统管理

DAY01 开学典礼、云网络基础

DAY02 Linux系统简介、安装Linux系统、Linux基本操作

DAY03 命令行基础、目录和文件管理

DAY04 管理用户和组、tar备份与恢复、cron计划任务

DAY05 教学环境介绍、权限和归属

DAY06 软件包管理、分区规划及使用、NTP时间同步

DAY07 LVM逻辑卷、综合串讲、综合练习

云计算应用管理

DAY01 Shell脚本基础、使用变量、条件测试及选择、列表式循环

DAY02 系统安全保护、配置用户环境、防火墙策略管理

DAY03 ISCSI共享存储、数据库服务基础、管理表数据

DAY04 配置NFS共享、HTTP服务基础

DAY05 网页内容访问、部署动态网站 、安全Web服务

DAY06 综合串讲、综合练习

系统&服务管理进阶

DAY01 扩展的几个应用、发布网络YUM源、vim编辑技巧、源码编译安装

DAY02 KVM构建及管理、virsh控制工具、镜像管理、虚拟机快建技术

DAY03 DNS服务基础、特殊解析、DNS子域授权、缓存DNS

DAY04 Split分离解析、RAID磁盘阵列、进程管理、日志管理、systemctl控制

DAY05 批量装机环境、配置PXE引导、kickstart自动应答

DAY06 rsync同步操作、inotify实时同步、Cobbler网络装机

##################################################################################################

准备笔记本与笔（默写），U盘（后期存储备份教学数据）

先讲解后练习，勤奋的练习

##################################################################

一 教学环境

1.真机是Linux 账户:student 密码:

2.虚拟化软件: 虚拟众多的硬件

虚拟机被损坏,与真机无关

########################################

典型服务模式

• C/S,Client/Server架构

– 由服务器提供资源或某种功能

– 客户机使用资源或功能

##############################################

TCP/IP协议简介

• TCP/IP是最广泛支持的通信协议集合

– 包括大量Internet应用中的标准协议

– 支持跨网络架构、跨操作系统平台的通信

• 主机与主机之间通信的三个要素

– IP地址(IP address)

– 子网掩码(subnet mask)

– IP路由(IP router):涉及路由器

#################################################

IP地址(IP address):唯一标识网络中一个节点地址

• 地址组成(点分十进制):

– 一共32个二进制数

　11110000.00001111.10101010.11111100

00000000.00000000.00000000.00000000

11111111.11111111.11111111.11111111

　　 11111111 = 255

例子： 　1.2.3.4 192.168.1.1 10.11.12.13

#################################################

为虚拟机win2008配置IP地址

利用管理员进行登录，密码为：tedu

配置IP地址：　　192.168.1.1　　子网掩码：255.255.255.0

– 右击桌面网络 --＞属性 --＞更改适配器设置

– 双击“本地连接” --＞ 属性

– 双击“Internet协议版本4(TCP/IPv4)”

– 配置完成后,单击“确定”完成

查看IP地址：

– 右击桌面网络 --＞属性 --＞更改适配器设置

– 双击“本地连接” --＞ 详细信息

采用ipconfig命令的方式，查看IP地址：

　windows键　＋　ｒ＝　开始＋运行

　　　输入cmd-->敲回车--＞输入　ipconfig-->敲回车

　　　--＞输入　exit　关闭命令行界面

################################################

• IP地址的分类

– A类:1 ~ 127 　网+主+主+主

– B类:128 ~ 191 　 网+网+主+主

– C类:192 ~ 223 　网+网+网+主

• 组播及科研专用

– D类:224 ~ 239 组播

– E类:240 ~ 254 科研

192.168.1.1=只看ip地址的第一个数字,进行判别分类

################################################

IP地址由网络位与主机位，两部来组成

网络位=座机号码的区号，标识一个网络

主机位=编号

子网掩码：用来区分IP地址的网络位与主机位

　　　　　利用二进制的１标识网络位

　　　　　利用二进制的0标识主机位

　192.168.1.1＝11000000.10101000.00000001.00000001

11111111.11111111.11111111.00000000

255.255.255.0

　座机号码：区号－号码（编号）

　　北京：010－12345678＝北京区域，编号为12345678的座机

　　石家庄：0311－7354444＝石家庄区域,编号为7354444的座机

　区号：标识区域

　192.168.1.1　＝　192.168.1网络，编号为１的主机

　192.168.１.2　＝　192.168.１网络，编号为２的主机

　相同网络通信，网络位相同

　192.168.1.1 255.255.255.0 = 192.168.1.1/24

192.168.1.1/24 :该IP地址有24个网络位

　1.该IP地址是哪一类 　 2.判别网络位与主机位

################################################

克隆虚拟机win2008

1.双击桌面windows还原,打开桌面虚拟系统管理器,双击win2008,选择查看,详情,把内存4096修改成2048,应用.

2.进行虚拟机克隆

　双击＂虚拟系统管理器＂--＞右击＂win2008＂--＞克隆--＞克隆

3. 配置二台win2008的网络参数

配置win2008的IP地址 192.168.1.10

子网掩码 255.255.255.0

网关 192.168.1.254

dns 202.106.0.20

– 右击桌面网络 --＞属性 --＞更改适配器设置

– 双击“本地连接” --＞ 属性

– 双击“Internet协议版本4(TCP/IPv4)”

– 配置完成后,单击“确定”完成

配置win2008-clone的IP地址：192.168.1.20

子网掩码 255.255.255.0

网关 192.168.1.254

dns 202.106.0.20

– 右击桌面网络 --＞属性 --＞更改适配器设置

– 双击“本地连接” --＞ 属性

– 双击“Internet协议版本4(TCP/IPv4)”

– 配置完成后,单击“确定”完成

采用ipconfig命令的方式，查看IP地址：

　windows键　＋　ｒ＝　开始＋运行

　　　输入cmd-->敲回车--＞输入　ipconfig-->敲回车

　　　--＞输入　exit　关闭命令行界面

################################################

采用命令方式，测试网络联通性：

　　ping　　命令

　　ping 对方IP地址　　　ping命令是双向，有去有回

　　ping 　192.168.1.20

　　ping 　192.168.1.２

###############################################

　ftp://172.40.50.114

###############################################

网关地址:一个网络到另一个网络的关口　（解决不同网络通信）

DNS服务器：将网站的域名解析为,对应的IP地址

www.qq.com-------＞腾讯的服务器

www.baidu.com-------＞百度的服务器

###############################################

192.168.0.1 255.255.255.0 192.168.0.0 1

192.168.0.2 255.255.255.0 192.168.0.0 2

11000000.10101000.00000000.00000001

11111111.11111111.11111111.00000000

----------------------------------------------------------------------------------

11000000.10101000.00000000.00000000

192.168.0.0

10.0.0.1 255.0.0.0 = 10.0.0.1/8

192.168.20.100 255.255.255.0 = 192.168.20.100/24

## Day02

#################################################################

Linux系统简介

什么是Linux?

Linux是一种操作系统：可以让计算机硬件正常工作

Unix/Linux发展史

• UNIX诞生，1970-1-1

Linux的诞生

• Linux之父,Linus Torwalds

– 1991年10月,发布0.02版(第一个公开版)内核

– 1994年03月,发布1.0版内核

– 标准读音:　哩呐科斯

版本号:主版本.次版本.修订号

Linux发行版本

• 发行版的名称/版本由发行方决定

– Red Hat Enterprise Linux（RHEL） 5/6/7

– Suse Linux Enterprise 12

– Debian Linux 7.8

– Ubuntu Linux 14.10/15.04

– ......

################################################

• CentOS,社区企业操作系统

– Community Enterprise Operating System

– http://www.centos.org/

• 基于Linux的企业服务器

• 嵌入式系统

• 高性能大型运算

###############################################

安装Linux系统

CentOS,社区企业操作系统,建议2G以上内存

Ctrl + Alt = 鼠标回到真机

#################################################

预备知识

Linux目录结构：树型结构

• 最顶层为根目录(/)

– Unix/Linux的基本哲学理念:一切皆文件

　　根目录(/)：所有的数据都在此目录下（Linux系统的起点）

路径：/abc/nsd/1.txt

/dev：存放设备(硬盘或键盘或鼠标或显示器.....)相关的数据

Linux管理员用户为:root 　　进行登录

Linux磁盘表示

　hd,表示IDE设备

　sd,表示SCSI设备

　vd,表示虚拟设备

　/dev/sda:表示SCSI设备，第一块

　/dev/sdb:表示SCSI设备，第二块

　/dev/sdc:表示SCSI设备，第三块

　/dev/sdd:表示SCSI设备，第四块

#################################################

Linux基本操作

命令行基本操作

获取命令行界面

• 虚拟控制台切换( Ctrl + Alt + Fn 组合键)

– tty1:图形桌面

– tty2~tty6:字符控制台

• 右键 "打开终端"

编辑--->配置文件首选项--->修改字体大小

• 命令行提示标识的含义

– [当前用户@主机名 工作目录]$

– 若当前用户是root,则最后一个字符为 #

[root@svr7 桌面]#

– 否则,最后一个字符为 $

[teacher@svr7 桌面]$

Ctrl shift 　+ =　终端字体变大

Ctrl － =　终端字体变小

##################################################

查看及切换目录

• pwd — Print Working Directory

– 用途:查看当前工作目录

• cd — Change Directory

– 用途:切换工作目录

– 格式:cd [目标文件夹位置]

• ls — List

– 格式:ls [选项]... [目录或文件名]...

[root@localhost ~]# pwd #显示当前所在位置

[root@localhost ~]# cd /abc　　　　＃切换到/abc目录

bash: cd: /abc: 没有那个文件或目录

[root@localhost ~]# cd /　　　　　＃切换到　/　目录

[root@localhost /]# pwd

[root@localhost /]# ls　　　　　＃显示当前目录内容

[root@localhost /]# cd /boot

[root@localhost /boot]# pwd

[root@localhost /boot]# ls

[root@localhost /]# cd /opt

[root@localhost /opt]# pwd

[root@localhost /opt]# ls

　颜色：

　　蓝色：目录

　　黑色：文件

[root@localhost /]# ls

[root@localhost /]# ls /root

[root@localhost /]# ls /home

[root@localhost /]# ls /opt

[root@localhost /]# ls /boot

[root@localhost /]# ls /root

]# ls /root/anaconda-ks.cfg

]# cat /root/anaconda-ks.cfg #查看文件内容

]# cat /etc/passwd　　　#查看文件内容

]# cat /etc/hosts　　　　#查看文件内容

]# cat /etc/fstab　　　　#查看文件内容

]# cat /etc/redhat-release　#查看当前系统版本

###############################################

以 / 开始的绝对路径

以当前为参照的相对路径

[root@localhost /]# cd /etc/pki/

[root@localhost pki]# pwd

[root@localhost pki]# ls

[root@localhost pki]# cd /etc/pki/CA #绝对路径

[root@localhost CA]# pwd

[root@localhost CA]# cd /etc/pki/

[root@localhost pki]# pwd

[root@localhost pki]# ls

[root@localhost pki]# cd CA　　　　＃相对路径

[root@localhost CA]# pwd

[root@localhost CA]# cd /

[root@localhost /]# ls root #相对路径

[root@localhost /]# ls /root #绝对路径

###############################################

　　.. :表示上一级目录

[root@localhost /]# cd .. 　　　　#后退

[root@localhost /]# cd /etc/pki/

[root@localhost pki]# pwd

/etc/pki

[root@localhost pki]# cd ..

[root@localhost etc]# pwd

/etc

[root@localhost etc]# cd ..

[root@localhost /]# pwd

/

[root@localhost /]# cd /etc/pki/

[root@localhost pki]# cd ../..　　　＃一次性后退两层

[root@localhost /]# pwd

#################################################

• 真机列出CPU处理器信息

[root@svr7 桌面]# lscpu

.......

CPU(s): 4

.......

型号名称： Intel(R) Core(TM) i5-4430S CPU @ 2.70GHz

.......

• 真机检查内存大小、空闲情况

[root@svr7 桌面]# cat /proc/meminfo

MemTotal: 16330728 kB

.......

查看主机名和IP信息

• 列出当前系统的主机名称

[root@localhost /]# hostname　　#查看当前主机名

localhost.localdomain

[root@localhost /]# hostname　 A.tedu.cn #临时设置

[root@localhost /]# hostname　　#查看当前主机名

[root@localhost /]# exit 　　 #关闭当前终端

开启一个新的终端进行验证：

[root@A /]#　hostname

• 列出已激活的网卡连接信息

[root@A ~]# ifconfig　　　　　　　　　　　#查看IP地址信息

[root@A ~]# ifconfig eth0 192.168.1.1　#临时设置IP地址

[root@A ~]# ifconfig eth0

[root@A ~]# ping 192.168.1.1

按Ctrl + c 结束正在运行的命令

lo: 本地回环接口，专用于测试

inet 127.0.0.1 netmask 255.0.0.0

　　127.0.0.1：永久代表本机

[root@A ~]# ping 127.0.0.1

按Ctrl + c 结束正在运行的命令

##########################################

可以永久设置主机名与IP地址与子网掩码与网关地址与DNS服务器地址

修改主机名：设置系统主机名-->输入新的名字-->确定-->确定

[root@A ~]# nmtui

[root@nsd1903 ~]# hostname

nsd1903.tedu.cn

[root@nsd1903 ~]#

修改IP地址与子网掩码

[root@A ~]#　nmtui

　编辑连接 -->eth0-->编辑ipv4-->显示-->将自动修改手动

　-->按空格　勾选　自动连接 　 #每次开机自动生效

　-->按空格　勾选　对所有用户有效

激活配置

[root@A ~]#　nmtui

　启用连接 -->eth0-->敲回车　禁用-->敲回车　激活

　最终eth0前面有一个＊代表激活状态

###############################################

创建文档

• mkdir — Make Directory

– 格式:mkdir [/路径/]目录名...

[root@nsd1903 ~]# mkdir nsd01 　　 #当前创建目录

[root@nsd1903 ~]# pwd

/root

[root@nsd1903 ~]# ls

[root@nsd1903 ~]# mkdir /opt/nsd02　＃在指定路径下创建

[root@nsd1903 ~]# ls /opt/

• touch命令

– 用途:新建空文件　格式:touch 文件名...

[root@nsd1903 ~]# touch 1.txt　　　　#当前创建文件

[root@nsd1903 ~]# ls

[root@nsd1903 ~]# touch /opt/2.txt　＃在指定路径下创建

[root@nsd1903 ~]# ls /opt/

文本内容操作

• less分屏阅读工具

• 格式:less [选项] 文件名...

– 优势:支持前后翻页

• 基本交互操作

– 按 / 键向后查找(n、N切换)

[root@nsd1903 ~]# less /etc/passwd

　　按键盘　上　下　可以进行滚动

　　/a 　　#全文查找ａ　(n、N切换)

　　按ｑ退出

• head、tail 命令

– 格式:　head -n 数字 文件名

　　　　tail -n 数字 文件名

[root@nsd1903 ~]# head -2 /etc/passwd #显示头两行内容

[root@nsd1903 ~]# head -3 /etc/passwd

[root@nsd1903 ~]# tail -2 /etc/passwd　＃显示尾两行内容

[root@nsd1903 ~]# tail -3 /etc/passwd

• grep工具

– 用途:输出包含指定字符串的行

– 格式:grep 　'查找条件' 　　目标文件

[root@nsd1903 ~]# grep　 root 　/etc/passwd

[root@nsd1903 ~]# grep 　a　 /etc/passwd

#############################################

关机及重启操作

• 关机:poweroff

[root@svr7 ~]# poweroff

• 重启:reboot

[root@svr7 ~]# reboot

#############################################

编辑命令或路径的技巧：　Tab可以进行补全

[root@nsd1903 ~]# if(tab)(tab)

if ifdown ifup

ifcfg ifenslave

ifconfig ifstat

[root@nsd1903 ~]# ifco(tab)

[root@nsd1903 ~]# cat /et(tab)/red(tab)

CentOS Linux release 7.5.1804 (Core)

]# ls /etc/sysconfig/network-scripts/ifcfg-eth0

]# ls /et(tab)/sysco(tab)/netw(tab)-(tab)/ifc(tab)-e(tab)

## Day02 下

一、Linux系统简介

1.什么是Linux?

Linux是一种操作系统，主要用在企业服务器上面

2.Linux系统内核

版本号:主版本.次版本.修订号

3.RedHat系列版本

– Red Hat Enterprise Linux（RHEL） 5/6/7

– CentOS

–Fedora Core社区版

二、安装Linux系统

1.Linux 一切皆文件

Linux严格区分大小写

2.Linux磁盘表示

/dev/xdy

/dev 表示/dev目录

xd hd ide接口硬盘

sd 非ide接口硬盘（SATA、SCSI）

vd 虚拟机磁盘

y a 第一块硬盘

b 第二块硬盘

c 第三块硬盘

...

3.Linux管理员 root

三、命令行基本操作

1.Linux虚拟控制台

tty1 ~ tty6

tty1 图形

tty2~tty6 字符

切换：

图形切换到字符 ctrl+alt+(F2~F6)

字符之间切换 alt+(F2~F6)

字符到图形 alt+F1

2.Linux命令行提示符

[登录用户@主机名 工作目录] 身份标识

# 代表管理员

$ 代表普通用户

3.Linux基本命令

# pwd //显示当前工作目录

# cd //切换路径

# ls //列出目录内容/文件权限

# cat //查看文件内容

# lscpu //查看cpu信息

# cat /proc/meminfo //查看内存信息

# hostname //查看主机名

# ifconfig //查看ip地址

# mkdir //创建目录

# touch //创建文件

# less //查看大文件内容分屏查看

# head //默认查看文件头10行

# tail //默认查看文件尾10行

# grep //过滤包含关键字的行

# nmtui //配置网络参数（ip地址、子网掩码、网关、dns、主机名）

# poweroff //关机

# reboot //重启

编辑命令或路径的技巧：　Tab可以进行补全

[root@nsd1903 ~]# if(tab)(tab)

if ifdown ifup

ifcfg ifenslave

ifconfig ifstat

[root@nsd1903 ~]# ifco(tab)

[root@nsd1903 ~]# cat /et(tab)/red(tab)

CentOS Linux release 7.5.1804 (Core)

]# ls /etc/sysconfig/network-scripts/ifcfg-eth0

]# ls /et(tab)/sysco(tab)/netw(tab)-(tab)/ifc(tab)-e(tab)

## Day03

牛犇

#################################################

命令行基础

什么是命令?

• Linux命令

– 用来实现某一类功能的指令或程序

执行大多数命令,都要找到相应的程序

[root@localhost ~]# ls /usr/bin/hostname

/usr/bin/hostname #绿色可以执行的程序

[root@localhost ~]#

– 命令的执行依赖于解释器(默认解释器:/bin/bash)

用户------->解释器------->内核------->硬件

[root@localhost ~]# cat /etc/shells

/bin/sh

/bin/bash

/sbin/nologin

/usr/bin/sh

/usr/bin/bash

/usr/sbin/nologin

/bin/tcsh

/bin/csh

[root@localhost ~]#

###################################################

• Linux命令的分类

– 内部命令:属于解释器的一部分

– 外部命令:解释器之外的其他程序

命令行的一般格式

• 基本用法

– 命令字 [选项]... [参数1] [参数2]...

打 周老师 .......

查看文本文件内容

[root@localhost ~]# cat /etc/redhat-release

[root@localhost ~]# cat --help

[root@localhost ~]# cat -n /etc/redhat-release

[root@localhost ~]# cat -n /etc/passwd

[root@localhost ~]# cat /etc/redhat-release /etc/hosts

[root@localhost ~]# ls /root /boot

[root@localhost ~]# ls --help

################################################

快速编辑技巧

• Tab键自动补全

– 可补齐命令字、选项、参数、文件路径、软件名、服务名

• 快捷键

– Ctrl + l:清空整个屏幕

– Ctrl + c:废弃当前编辑的命令行(结束正在运行的命令)

– Esc + . 或 Alt + .:粘贴上一个命令的参数

[root@localhost ~]# ls /etc/hosts

/etc/hosts

[root@localhost ~]# cat Alt + .

[root@localhost ~]# cat -n Alt + .

– Ctrl + u:清空至行首

– Ctrl + w:往回删除一个单词(以空格界定)

##################################################

mount挂载操作:让目录成为设备的访问点

一 安装软件包,前提需要基本软件安装包

二 从光盘中获得软件包

三 让系统识别光盘内容

Windows系统:

光盘文件---->光驱设备---->CD驱动器图标(访问点)

Linux系统:

光盘文件---->光驱设备---->访问点(目录/文件夹)

/dev/sr0

/dev/hdc

一切皆文件

1.图形鼠标将光盘镜像文件放入虚拟光驱设备

2.查看光驱设备

[root@localhost ~]# ls /dev/sr0

/dev/sr0

[root@localhost ~]# ls /dev/cdrom

/dev/cdrom #青色:代表快捷方式颜色

[root@localhost ~]#

3.命令行挂载设备

[root@localhost ~]# mkdir /dvd

[root@localhost ~]# ls /dvd

[root@localhost ~]# mount /dev/cdrom /dvd

mount: /dev/sr0 写保护，将以只读方式挂载

[root@localhost ~]# ls /dvd/

4.卸载

[root@localhost ~]# umount /dvd/

[root@localhost ~]# ls /dvd/

[root@localhost ~]# umount /dvd/

umount: /dvd/：未挂载

[root@localhost ~]# mkdir /nsd1904

[root@localhost ~]# mount /dev/cdrom /nsd1904

[root@localhost ~]# ls /nsd1904

注意事项:

1.卸载,当前不要在挂载点(访问点)目录下

[root@localhost dvd]# umount /dvd/

umount: /dvd：目标忙。

(有些情况下通过 lsof(8) 或 fuser(1) 可以

找到有关使用该设备的进程的有用信息)

[root@localhost dvd]#

2.挂载允许一个设备具备多个挂载点(访问点)

3.不允许多个设备挂载到一个挂载点(访问点)

4.强烈建议挂载点(访问点)选择自己创建的目录

##################################################

查看及切换目录

使用pwd、cd

• pwd — Print Working Directory

– 用途:查看当前工作目录

• cd — Change Directory

– 用途:切换工作目录

– 格式:cd [目标文件夹位置]

~:用户的家目录

~user表示用户user的家目录

/root:管理员的家目录

/home:存放所有普通用户家目录

[root@localhost ~]# useradd tom #创建普通用户tom

[root@localhost ~]# cd ~tom

[root@localhost tom]# pwd

/home/tom

[root@localhost tom]# useradd dc #创建普通用户dc

[root@localhost tom]# cd ~dc

[root@localhost dc]# pwd

/home/dc

[root@localhost dc]# useradd tc #创建普通用户tc

[root@localhost dc]# cd ~tc

[root@localhost tc]# pwd

/home/tc

[root@localhost tc]# cd ~root #切换到root用户的家目录

[root@localhost ~]# pwd

####################################################

. 表示当前目录

####################################################

ls 列出文档及属性

• ls — List

– 格式:ls [选项]... [目录或文件名]

• 常用命令选项

– -l:以长格式显示,显示目录内容的详细属性

– -A:包括名称以 . 开头的隐藏文档

– -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)

[root@localhost ~]# ls /root

[root@localhost ~]# ls -l /root #显示目录内容的详细属性

[root@localhost ~]# ls -ld /root #显示目录本身的详细属性

[root@localhost ~]# ls -ld /

[root@localhost ~]# ls -l /dev/cdrom #显示文件详细属性

[root@localhost ~]# ls -l /etc/passwd

[root@localhost ~]# ls -l /etc/redhat-release

[root@localhost ~]# ls -l /etc/passwd

[root@localhost ~]# ls -lh /etc/passwd

[root@localhost ~]# ls -lh /boot

[root@localhost ~]# ls -A /root

[root@localhost ~]# ls -A /boot

百度搜索 EB容量单位

百度搜索 正则表达式

###################################################

使用通配符

• 针对不确定的文档名称,以特殊字符表示

– \*:任意多个任意字符

– ?:单个字符

[root@localhost ~]# ls /boot/vm\* #以vm开头

[root@localhost ~]# ls /root/a\* #以a开头

[root@localhost ~]# ls /dev/tty\* #以tty开头

[root@localhost ~]# ls /etc/\*tab #以tab结尾

[root@localhost ~]# ls /etc/\*.conf #以.conf结尾

[root@localhost ~]# ls /etc/r\*.conf #以r开头并且.conf结尾

[root@localhost ~]# ls /etc/\*passwd\*

[root@localhost ~]# ls /dev/tty?

[root@localhost ~]# ls /dev/tty??

• 针对不确定的文档名称,以特殊字符表示

– [a-z]:多个字符或连续范围中的一个,若无则忽略

– {a,min,xy}:多组不同的字符串,全匹配

[root@localhost ~]# ls /dev/tty[3-8]

[root@localhost ~]# ls /dev/tty[1-9]

[root@localhost ~]# ls /dev/tty{1,3,5,7,9}

[root@localhost ~]# ls /dev/tty{1,3,5,7,9,23}

[root@localhost ~]# ls /dev/tty{1,3,S0,S1}

[root@localhost ~]# ls /dev/tty{1,3,S0,haha}

#################################################

别名的定义:简化复杂命令

• 查看已设置的别名

– alias [别名名称]

• 定义新的别名

– 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]

[root@localhost ~]# hostname

localhost.localdomain

[root@localhost ~]# hn

bash: hn: 未找到命令...

[root@localhost ~]# alias hn='hostname' #定义别名

[root@localhost ~]# hn #运行别名测试

[root@localhost ~]# alias #查看当前系统有效的别名

[root@localhost ~]# unalias hn #删除别名

[root@localhost ~]# hn

bash: hn: 未找到命令...

[root@localhost ~]#

####################################################

mkdir 创建目录

• mkdir — Make Directory

– 格式:mkdir [-p] [/路径/]目录名...

-p:创建父目录

[root@localhost ~]# mkdir /opt/nsd01

[root@localhost ~]# ls /opt/

[root@localhost ~]# mkdir /opt/nsd02 /opt/nsd03

[root@localhost ~]# ls /opt/

[root@localhost ~]# mkdir -p /vod/movie/cartoon

[root@localhost ~]# ls /

[root@localhost ~]# mkdir -p /opt/aa/bb/cc/dd

[root@localhost ~]# ls /opt/

-R:

递归:目录本身以及目录下所有

[root@server0 ~]# ls -R /vod

/vod:

movie

/vod/movie:

cartoon

/vod/movie/cartoon:

#################################################

rm 删除

• rm — Remove

– 格式:rm [选项]... 文件或目录...

• 常用命令选项

– -r、-f:递归删除(含目录)、强制删除

[root@localhost ~]# rm -rf /opt/nsd03

[root@localhost ~]# ls /opt/

[root@localhost ~]# rm -rf /opt/rh

[root@localhost ~]# ls /opt/

##################################################

mv 移动/改名

• mv — Move

– 格式: mv [选项]... 原文件... 目标路径

[root@localhost ~]# mkdir /opt/nsd01

[root@localhost ~]# touch /opt/1.txt

[root@localhost ~]# ls /opt/

[root@localhost ~]# mv /opt/1.txt /opt/nsd01/

[root@localhost ~]# ls /opt/

[root@localhost ~]# ls /opt/nsd01/

重命名:路径不变的移动

[root@localhost ~]# mv /opt/nsd01/ /opt/haha01

[root@localhost ~]# ls /opt/

[root@localhost ~]# mv /opt/haha01/ /opt/stu01

[root@localhost ~]# ls /opt/

####################################################

cp 复制

• cp — Copy

– 格式:cp [选项]... 原文件... 目标路径

• 常用命令选项

– -r:递归,复制目录时必须有此选项

1.基本用法

[root@localhost ~]# cp /etc/passwd /opt/

[root@localhost ~]# ls /opt/

[root@localhost ~]# cp -r /home/ /opt/

[root@localhost ~]# ls /opt/

2.可以支持两个以上的参数,永远将最后一个参数做目标,其他所有参数作为源

[root@localhost ~]# cp -r /etc/hosts /etc/fstab /boot/ /opt/

[root@localhost ~]# ls /opt/

3.cp经常与点 . 连用,将数据复制到当前路径下

.:当前目录

[root@localhost ~]# cd /opt/

[root@localhost opt]# pwd

[root@localhost opt]# cp -r /etc/shadow .

[root@localhost opt]# ls

[root@localhost /]# cd /etc/sysconfig/network-scripts/

[root@localhost network-scripts]# pwd

[root@localhost network-scripts]# cp /etc/passwd .

[root@localhost network-scripts]# ls

4.复制到目标路径下,可以重新命名

[root@localhost /]# rm -rf /opt/\*

[root@localhost /]# ls /opt/

[root@localhost /]# cp /etc/passwd /opt/

[root@localhost /]# ls /opt/

[root@localhost /]# cp /etc/passwd /opt/pd

[root@localhost /]# ls /opt/

[root@localhost /]# cp -r /home/ /opt/abc

[root@localhost /]# ls /opt/

请分析下列两次复制的不同:

[root@localhost /]# rm -rf /opt/\*

[root@localhost /]# cp -r /home/ /opt/redhat

将/home目录复制到/opt/并且重命名为redhat

[root@localhost /]# cp -r /home/ /opt/redhat

将/home目录复制到/opt/redhat目录下

##################################################

grep针对文本文件内容进行过滤

在文本文件内容进行过滤,输出包含指定字符串的行

/etc/passwd系统级配置文件

[root@localhost /]# grep root /etc/passwd

• 常用命令选项

– -v,取反匹配

– -i,忽略大小写

[root@localhost /]# grep -v root /etc/passwd

[root@localhost /]# grep ROOT /etc/passwd

[root@localhost /]# grep -i ROOT /etc/passwd

• 常用的匹配模式

– word 包含字符串word

– ^word 以字符串word开头

– word$ 以字符串word结尾

[root@localhost /]# grep ^root /etc/passwd

[root@localhost /]# grep root$ /etc/passwd

[root@localhost /]# grep bash$ /etc/passwd

#################################################

vim命令行文本编辑器

三个模式:

命令模式 插入模式(输入模式) 末行模式

[root@server0 ~]# vim /opt/haxi.txt

命----i 键---->插入模式(Esc 键回到命令模式)

令

模

式----: 键---->末行模式(Esc 键回到命令模式)

末行模式 :wq 保存并退出

末行模式 :q! 强制不保存并退出

###################################################

## Day04

#################################################

复制时出现重名,强制覆盖

在本次复制,临时取消别名

[root@A ~]# cp -r /boot/ /opt/

[root@A ~]# cp -r /boot/ /opt/

[root@A ~]# \cp -r /boot/ /opt/

#################################################

命令输出重定向

>:覆盖重定向

>>:追加重定向

[root@A ~]# ls --help

[root@A ~]# ls --help > /opt/ls.txt

[root@A ~]# ls /opt/

[root@A ~]# less /opt/ls.txt

[root@A ~]# hostname

[root@A ~]# hostname > /opt/ls.txt

[root@A ~]# cat /opt/ls.txt

[root@A ~]# hostname >> /opt/ls.txt

[root@A ~]# cat /opt/ls.txt

[root@A ~]# echo haha

[root@A ~]# cat /opt/ls.txt

[root@A ~]# echo 123456 >> /opt/ls.txt

[root@A ~]# cat /opt/ls.txt

[root@A ~]# echo hello > /opt/ls.txt

[root@A ~]# cat /opt/ls.txt

##################################################

| 管道操作: 将前面命令的输出,交由后面命令处理

显示文件内容/etc/passwd的8~12行

[root@A ~]# head -12 /etc/passwd | tail -5

[root@A ~]# head -12 /etc/passwd | tail -5 | cat -n

[root@A ~]# cat -n /etc/passwd | head -12 | tail -5

[root@A ~]# ifconfig | less

[root@A ~]# ifconfig | head -2

##################################################

管理用户和组

用户:登录操作系统,不同的用户具备不同的权限

组:方便管理用户

用户唯一标识: UID

组唯一标识: GID

管理员root的UID为0

组的分类: 基本组(私有组) 附加组(从属组)

原则:一个用户必须至少属于一个组

基本组(私有组) :由系统创建与用户同名的组

附加组(从属组):由管理员root创建的组

[root@A ~]# useradd nsd01

[root@A ~]# 组:nsd01

附加组(从属组): caiwu meinv haha jishu

###################################################

添加用户

用户基本信息存放在 /etc/passwd 文件

[root@A ~]# head -1 /etc/passwd

root:x:0:0:root:/root:/bin/bash

用户名:密码占位符:UID:基本组GID:用户描述信息:家目录:解释器程序

• 使用 useradd 命令

– useradd [选项]... 用户名

• 常用命令选项

– -u 用户id、-d 家目录路径、-s 登录解释器、-G 附加组

[root@A ~]# useradd nsd02 #创建nsd02用户

[root@A ~]# id nsd02 #查看nsd02用户基本信息

[root@A ~]# useradd nsd03

[root@A ~]# id nsd03

[root@A ~]# useradd -u 1500 nsd04 #指定用UID值创建

[root@A ~]# id nsd04

[root@A ~]# useradd -d /opt/abc nsd05 #指定用户家目录创建

[root@A ~]# ls /opt/

[root@A ~]# grep nsd05 /etc/passwd

用户名:密码占位符:UID:基本组GID:用户描述信息:家目录:解释器程序

[root@A ~]# useradd -d /opt/nsd06 nsd06

[root@A ~]# ls /opt/

[root@A ~]# grep nsd06 /etc/passwd

###################################################

-s 登录解释器

/sbin/nologin:禁止登录操作系统

[root@A ~]# useradd -s /sbin/nologin nsd06

[root@A ~]# grep nsd06 /etc/passwd

-G 附加组

[root@A ~]# groupadd stugrp #创建stugrp组

[root@A ~]# useradd -G stugrp nsd07

[root@A ~]# id nsd07

[root@A ~]# useradd -G stugrp nsd08

[root@A ~]# id nsd08

##################################################

设置登录密码

用户密码信息存放在 /etc/shadow 文件

[root@A ~]# tail -1 /etc/shadow

nsd08:$6$VjFxZ9GD$XaYwBkrcR0eVnFNZrO/om1bQKFYzlsXW.aHMCx.nOGguVde19zLyyl1jBekeU6r49UfFcvWy4sUDw8iG2OSv2.:18022:0:99999:7:::

用户名:密码加密字符串:上一次修改密码的时间

1970-1-1~2019-5-6 18022

[root@A ~]# useradd nsd10

[root@A ~]# tail -1 /etc/shadow

nsd10:!!:18022:0:99999:7:::

第二个字段为!!,代表用户为锁定状态

• 使用 passwd 命令

– passwd [用户名]

[root@A ~]# useradd nsd08

[root@A ~]# passwd nsd08 #交互方式设置密码

更改用户 nsd08 的密码 。

新的 密码： #输入新的密码

无效的密码： 密码少于 8 个字符

重新输入新的 密码： #重新输入新的密码

passwd：所有的身份验证令牌已经成功更新。

[root@A ~]# su - nsd08 #命令行临时切换身份

[nsd08@A ~]$ passwd

更改用户 nsd08 的密码 。

为 nsd08 更改 STRESS 密码。

（当前）UNIX 密码： #输入旧密码

新的 密码： #输入新的密码

重新输入新的 密码： #重新输入新的密码

passwd：所有的身份验证令牌已经成功更新。

[nsd08@A ~]$ exit #退出回到root

非交互式设置密码

echo '密码' | passwd --stdin 用户名

[root@A ~]# echo 123 | passwd --stdin nsd08

[root@A ~]# echo redhat | passwd --stdin nsd08

[root@A ~]# echo 654321 | passwd --stdin nsd08

#################################################

修改用户属性

• 使用 usermod 命令

– usermod [选项]... 用户名

• 常用命令选项

– -u 用户id、-d 家目录路径、-s 登录解释器、-G 附加组

[root@A ~]# useradd nsd11

[root@A ~]# id nsd11

[root@A ~]# grep nsd11 /etc/passwd

[root@A ~]# usermod -u 1800 -d /opt/test -s /sbin/nologin -G stugrp nsd11

[root@A ~]# id nsd11

[root@A ~]# grep nsd11 /etc/passwd

[root@A ~]# useradd nsd13

[root@A ~]# grep nsd13 /etc/passwd

[root@A ~]# ls /home/

[root@A ~]# usermod -d /opt/n13 nsd13

[root@A ~]# grep nsd13 /etc/passwd

[root@A ~]# ls /opt/n13

ls: 无法访问/opt/n13: 没有那个文件或目录

################################################

检查用户ID信息

• 使用 id 命令

– id 用户名

##################################################

删除用户

• 使用 userdel 命令

– userdel [-r] 用户名

-r:连同用户家目录一并删除

[root@A ~]# ls /home/

[root@A ~]# userdel nsd01

[root@A ~]# ls /home/

[root@A ~]# id nsd01

id: nsd01: no such user

[root@A ~]# rm -rf /home/nsd01/

[root@A ~]# ls /home/

[root@A ~]# ls /home/

[root@A ~]# userdel -r nsd02

[root@A ~]# ls /home/

####################################################

管理组账号

添加组

组基本信息存放在 /etc/group 文件

[root@A ~]# grep linux /etc/group

linux:x:1802:

组名:组的密码占位符:组的编号:组成员列表

• 使用 groupadd 命令

– groupadd [-g 组ID] 组名

[root@A ~]# groupadd linux

[root@A ~]# grep linux /etc/group

[root@A ~]# useradd natasha

[root@A ~]# useradd kaka

[root@A ~]# useradd jack

[root@A ~]# useradd xiha

管理组成员

组成员信息存放在 /etc/gshadow 文件

• 使用 gpasswd 命令

– gpasswd -a 用户名 组名

– gpasswd -d 用户名 组名

[root@A ~]# grep linux /etc/group

[root@A ~]# gpasswd -a natasha linux #将用户加入组

[root@A ~]# grep linux /etc/group

[root@A ~]# id natasha

[root@A ~]# gpasswd -a kaka linux

[root@A ~]# grep linux /etc/group

[root@A ~]# gpasswd -a jack linux

[root@A ~]# grep linux /etc/group

[root@A ~]# gpasswd -d kaka linux #将用户从组中删除

[root@A ~]# grep linux /etc/group

案例1:配置用户和组账号

• 新建用户 alex,其用户ID为3456,密码是flectrag

[root@A ~]# useradd -u 3456 alex

[root@A ~]# id alex

[root@A ~]# echo flectrag | passwd --stdin alex

• 创建下列用户、组以及组的成员关系:创建一个名为 adminuser 的组

[root@A ~]# groupadd adminuser

[root@A ~]# grep adminuser /etc/group

• 创建natasha 的用户,其属于 adminuser 组,这个组是该用户的从属组

[root@A ~]# gpasswd -a natasha adminuser

[root@A ~]# grep adminuser /etc/group

• 创建harry 的用户,其属于 adminuser 组,这个组是该用户的从属组

[root@A ~]# gpasswd -a harry adminuser

[root@A ~]# grep adminuser /etc/group

• 创建 sarah 的用户,其在系统中没有可交互的Shell(解释器),不是 adminuser 组的成员

[root@A ~]# useradd -s /sbin/nologin sarah

[root@A ~]# grep sarah /etc/passwd

• natasha 、harry、sarah 的密码都要设置为 flectrag

[root@A ~]# echo flectrag | passwd --stdin natasha

[root@A ~]# echo flectrag | passwd --stdin harry

[root@A ~]# echo flectrag | passwd --stdin sarah

###################################################

删除组

• 使用 groupdel 命令

– groupdel 组名

[root@A ~]# groupdel linux

[root@A ~]# grep linux /etc/group

###############################################

tar备份与恢复

作用:减少占用的空间,零散的文件整理到一个压缩包

压缩格式:

gzip-----> .gz

bzip2-----> .bz2

xz -----> .xz

制作tar包:打包

格式: tar 选项 /路径/压缩包的名字 /路径/源文档 .......

解包

格式: tar 选项 /路径/压缩包的名字 /释放位置

• tar 集成备份工具

– -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

– -C:指定释放路径

f选项必须放在,所有选项的最后

制作tar包:打包

]# tar -zcf /opt/file.tar.gz /home/ /etc/passwd

]# ls /opt/

]# tar -jcf /opt/nsd.tar.bz2 /home/ /etc/passwd

]# ls /opt/

]# tar -Jcf /opt/abc.tar.xz /home/ /etc/passwd

]# ls /opt/

-z:gzip压缩格式

-j:bzip2压缩格式

-J:xz压缩格式

[root@A ~]# mkdir /nsd02

[root@A ~]# tar -xf /opt/file.tar.gz -C /nsd02

[root@A ~]# ls /nsd02/

[root@A ~]# ls /nsd02/etc/

[root@A ~]# ls /nsd02/home/

[root@A ~]# mkdir /nsd03

[root@A ~]# tar -xf /opt/nsd.tar.bz2 -C /nsd03

[root@A ~]# ls /nsd03

[root@A ~]# tar -tf /opt/abc.tar.xz #查看包里内容

案例2:创建一个备份包

使用 tar 工具完成以下备份任务:

– 创建一个名为 /root/backup.tar.bz2 的归档文件

– 其中包含 /usr/local 目录中的内容

– tar 归档必须使用 bzip2 进行压缩

[root@A ~]# tar -jcf /root/backup.tar.bz2 /usr/local/

[root@A ~]# ls /root/

[root@A ~]# tar -tf /root/backup.tar.bz2 #查看包里内容

#####################################################

周期性任务

cron任务概述

• 用途:按照设置的时间间隔为用户反复执行某一项固定的系统任务

• 日志文件:/var/log/crond

执行周期 配置说明

分钟 从0到59之间的整数

小时 从0到23之间的整数

日期 从1到31之间的整数

月份 从1到12之间的整数

星期 0~7之间的整数,0或7代表星期日

• 使用 crontab 命令

– 编辑:crontab -e [-u 用户名]

– 查看:crontab -l [-u 用户名]

– 清除:crontab -r [-u 用户名]

如何编写crontab任务记录

– 分 时 日 月 周 任务命令行(绝对路径)

30 8 \* \* \* #每天早晨的8:30

30 23 \* \* 5 #每周的周五晚上11:30

30 8 1 \* \* #每月的1号早晨的8:30

30 23 \* \* 1-5 #周一至周五晚上11:30

30 23 \* \* 1,3,5 #周一或周三或周五晚上11:30

\*:匹配范围内任意时间

,:分隔多个不连续的时间点

-:指定连续时间范围

/n:指定时间频率,每n ...

1 \*/2 \* \* \* #每隔2个小时执行一次

\*/5 \* \* \* \* #每隔5分钟执行一次

30 23 1 \* 1 #周一或每月一号 晚上11:30

##################################################

每分钟记录当前系统时间,将系统时间每分钟写入文件/opt/time.txt

[root@A ~]# date

[root@A ~]# date >> /opt/time.txt

[root@A ~]# cat /opt/time.txt

[root@A ~]# crontab -e -u root

\* \* \* \* \* date >> /opt/time.txt

[root@A ~]# crontab -l -u root

[root@A ~]# cat /opt/time.txt

###################################################

[root@A ~]# crontab -e -u root

\*/2 \* \* \* \* date >> /opt/time.txt

[root@A ~]# crontab -l -u root

[root@A ~]# cat /opt/time.txt

###################################################

## Day05

教学环境介绍

使用教学虚拟机

• 每个学员机上有三台预先配置好的虚拟机

– server —— 作为练习用服务器

– desktop —— 作为练习用客户机

– classroom —— 提供网关/DNS/软件素材等资源

优先开启classroom虚拟机

真机上:使用rht-vmctl辅助工具

• 控制教学用虚拟机

– 格式:rht-vmctl 控制指令 虚拟机名

– 常用控制指令:

[student@room9pc01 ~]$ rht-vmctl 　reset　 classroom

classroom [OK]

域 classroom 已开始

[student@room9pc01 ~]$ rht-vmctl　 reset　 server

server [OK]

域 server 已开始

[student@room9pc01 ~]$ rht-vmctl 　reset　 desktop

desktop [OK]

域 desktop 已开始

虚拟机server与虚拟机desktop,利用root登录密码为redhat

#################################################

字体变大：ctrl shift +

字体变小：ctrl -

#################################################

虚拟机server:

1.查看系统的版本:RHEL7.0

2.查看eth0网卡的IP地址:172.25.0.11/24

3.查看主机名:server0.example.com

虚拟机desktop:

1.查看系统的版本:RHEL7.0

2.查看eth0网卡的IP地址:172.25.0.10/24

3.查看主机名:desktop0.example.com

#################################################

利用真机,远程管理两台虚拟机

1.前提真机可以与两台虚拟机通信

[student@room9pc01 ~]$ ping 172.25.0.11

[student@room9pc01 ~]$ ping 172.25.0.10

2.真机上,ssh命令进行远程管理

ssh 对方用户名@对方IP地址

[student@room9pc01 ~]$ ssh　root@172.25.0.11

[student@room9pc01 ~]$ ssh　root@172.25.0.10

开启一个新的终端：ctrl + shift + t

-X(大写):在远程管理时,开启对方的图形程序

[root@server0 ~]# exit

登出

Connection to 172.25.0.11 closed.

[student@room9pc01 ~]$ ssh -X root@172.25.0.11

[root@server0 ~]# firefox #火狐浏览器

[root@server0 ~]# firewall-config #防火墙图形配置工具

常见提示:dconf桌面服务报错,与远程管理无关

(process:30405): dconf-WARNING \*\*: failed to commit changes to dconf: 无法连接：拒绝连接

###################################################

补充:为真机,设置永久的别名,修改系统级配置文件

~/.bashrc #设置永久别名配置文件,每开一个终端会被执行一次

[student@room9pc01 ~]$ vim /home/student/.bashrc

# .bashrc

alias gos='ssh -X root@172.25.0.11'

alias god='ssh -X root@172.25.0.10'

新开一个终端验证:

[student@room9pc01 ~]$ gos

[student@room9pc01 ~]$ god

#################################################

权限和归属

基本权限

• 访问方式(权限)

– 读取:允许查看内容-read r

– 写入:允许修改内容-write w

– 可执行:允许运行和切换-execute x

对文本文件:

r: cat less head tail

w: vim 保存 > >>

x: 书写Shell脚本或Python脚本

• 权限适用对象(归属)

– 所有者(属主):拥有此文件/目录的用户-user u

– 所属组(属组):拥有此文件/目录的组-group g

– 其他用户:除所有者、所属组以外的用户-other o

zhangsan 1.txt

所有者的基本组成为该文件的所属组

####################################################

查看权限

• 使用 ls -l 命令

– ls -ld 文件或目录...

以d开头:目录

以-开头:文件

以l开头:快捷方式

[root@server0 ~]# ls -ld /etc/

[root@server0 ~]# ls -l /etc/passwd

[root@server0 ~]# ls -l /etc/shadow

[root@server0 ~]# ls -ld /home/student

[root@server0 ~]# ls -ld /tmp

####################################################

设置基本权限

• 使用 chmod 命令

– chmod 归属关系+-=权限类别 文档...

[-R]:递归设置权限

[root@server0 ~]# mkdir /nsd01

[root@server0 ~]# ls -ld /nsd01

[root@server0 ~]# chmod g+w /nsd01

[root@server0 ~]# ls -ld /nsd01

[root@server0 ~]# chmod u-w /nsd01

[root@server0 ~]# ls -ld /nsd01

[root@server0 ~]# chmod o=--- /nsd01

[root@server0 ~]# ls -ld /nsd01

[root@server0 ~]# chmod u=rwx,g=rx,o=rx /nsd01

[root@server0 ~]# ls -ld /nsd01

[root@server0 ~]# chmod ugo=rwx /nsd01

[root@server0 ~]# ls -ld /nsd01

[root@server0 ~]# mkdir -p /opt/aa/bb/cc

[root@server0 ~]# ls -ld /opt/aa

[root@server0 ~]# ls -ld /opt/aa/bb/

[root@server0 ~]# ls -ld /opt/aa/bb/cc/

[root@server0 ~]# chmod -R o=--- /opt/aa

[root@server0 ~]# ls -ld /opt/aa

##################################################

Linux判定用户具备的权限 匹配及停止

1.判定用户对该文档的身份 所有者>所属组>其他人

2.查看相应身份 权限位的权限

Permission denied :权限不足

目录的 r 权限:能够 ls 浏览此目录内容

目录的 w 权限:能够执行 rm/mv/cp/mkdir/touch/等更改目录内容的操作

目录的 x 权限:能够 cd 切换到此目录

以root用户新建/nsddir目录，在此目录下新建readme.txt文件

并进一步完成下列操作

1）使用户lisi能够在此目录下创建子目录 切换用户 su - lisi

chmod o+w /nsddir/

2）使用户lisi不能够在此目录下创建子目录

chmod o-w /nsddir/

3）使用户lisi能够修改readme.txt文件内容

chmod o+w /nsddir/readme.txt

4）调整此目录的权限，使所有用户都不能cd进入此目录

chmod u-x,g-x,o-x /nsddir/

5）为此目录及其下所有文档设置权限 rwxr-x---

chmod -R u=rwx,g=rx,o=--- /nsddir/

　　　-R：递归设置权限，目录下及目录下所有

##################################################

设置文档归属

• 使用 chown 命令

– chown [-R] 属主 文档...

– chown [-R] :属组 文档...

– chown [-R] 属主:属组 文档...

[root@server0 ~]# groupadd tedu

[root@server0 ~]# mkdir /nsd03

[root@server0 ~]# ls -ld /nsd03

[root@server0 ~]# chown lisi:tedu /nsd03

[root@server0 ~]# ls -ld /nsd03

[root@server0 ~]# chown zhangsan /nsd03

[root@server0 ~]# ls -ld /nsd03

[root@server0 ~]# chown :root /nsd03

[root@server0 ~]# ls -ld /nsd03

################################################

利用root用户新建/nsd06目录，并进一步完成下列操作

1）将属主设为gelin01，属组设为tarena组

[root@server0 /]# useradd gelin01

[root@server0 /]# useradd gelin02

[root@server0 /]# groupadd tarena

[root@server0 /]# chown gelin01:tarena /nsd06

2）使用户gelin01对此目录具有rwx权限

除属主与属组之外的人，对此目录无任何权限

[root@server0 /]# chmod o=--- /nsd06

3）使用户gelin02能进入、查看此目录内容

[root@server0 /]# gpasswd -a gelin02 tarena

4）将gelin01加入tarena组, 将nsd06目录的权限设为rw-r-x---

再测试gelin01用户能否进入此目录

[root@server0 /]# gpasswd -a gelin01 tarena

[root@server0 /]# chmod u=rw,g=rx /nsd06

###################################################

让lisi用户可以读取/etc/shadow文件内容,您有几种办法

1.利用其他用户

chmod o+r /etc/shadow

2.利用所属组

chown :lisi /etc/shadow

chmod g+r /etc/shadow

3.利用所有者

chown lisi /etc/shadow

chmod u+r /etc/shadow

###################################################

附加权限(特殊权限)

Set GID

• 附加在属组的 x 位上

属组的权限标识会变为 s

适用于目录,Set GID可以使目录下新增的文档自动设置与父目录相同的属组

让父目录下,新增的子文档继承父目录的所属组身份

[root@server0 ~]# mkdir /nsd07

[root@server0 ~]# chown :tedu /nsd07 #修改所属组

[root@server0 ~]# ls -ld /nsd07

[root@server0 ~]# mkdir /nsd07/abc

[root@server0 ~]# ls -ld /nsd07/abc

[root@server0 ~]# chmod g+s /nsd07/ #设置Set GID权限

[root@server0 ~]# ls -ld /nsd07/

[root@server0 ~]# mkdir /nsd07/test

[root@server0 ~]# ls -l /nsd07/

[root@server0 ~]# touch /nsd07/1.txt

[root@server0 ~]# ls -l /nsd07/

###################################################

/nsd09 rwx r-x --- 存放打卡记录

2019-5-6.txt

2019-5-7.txt

2019-5-8.txt

2019-5-9.txt

groupadd caiwu

chown :caiwu /nsd09

chmod g+s /nsd09

##################################################

Set UID

• 附加在属主的 x 位上

属主的权限标识会变为 s

适用于可执行文件,Set UID可以让使用者具有文件属主的身份及部分权限

[root@server0 ~]# /usr/bin/mkdir /opt/haha

[root@server0 ~]# ls /opt/

[root@server0 ~]# cp /usr/bin/mkdir /usr/bin/hahadir

[root@server0 ~]# /usr/bin/hahadir /opt/xixi

[root@server0 ~]# ls /opt/

[root@server0 ~]# chmod u+s /usr/bin/hahadir

[root@server0 ~]# ls -l /usr/bin/hahadir

[root@server0 ~]# su - lisi

[lisi@server0 ~]$ /usr/bin/mkdir test01

[lisi@server0 ~]$ /usr/bin/hahadir test02

[lisi@server0 ~]$ ls

test01 test02

[lisi@server0 ~]$ ls -l

[lisi@server0 ~]$ exit

###################################################

Sticky Bit

• 附加在其他人的 x 位上

其他人的权限标识会变为 t

适用于开放 w 权限的目录,可以阻止用户滥用 w 写入权限(禁止操作别人的文档)

[root@server0 ~]# mkdir /public

[root@server0 ~]# chmod ugo=rwx /public

[root@server0 ~]# ls -ld /public

[root@server0 ~]# chmod o+t /public/

[root@server0 ~]# ls -ld /public/

##############################################

## Day06

ACL权限(策略)

acl策略的作用

• 文档归属的局限性

– 任何人只属于三种角色:属主、属组、其他人

– 无法实现更精细的控制

• acl访问策略

– 能够对个别用户、个别组设置独立的权限

– 大多数挂载的EXT3/4、XFS文件系统默认已支持

设置acl访问控制策略

• 使用 getfacl、setfacl 命令

– getfacl 文档...

– setfacl [-R] -m u:用户名:权限类别 文档...

– setfacl [-R] -m g:组名:权限类别 文档...

– setfacl [-R] -b 文档... #删除所有ACL策略

– setfacl [-R] -x u:用户名 文档... #删除指定ACL策略

让lisi用户可以读取/etc/shadow文件内容,您有几种办法

1.利用其他用户

chmod o+r /etc/shadow

2.利用所属组

chown :lisi /etc/shadow

chmod g+r /etc/shadow

3.利用所有者

chown lisi /etc/shadow

chmod u+r /etc/shadow

4.利用ACL策略

setfacl -m u:lisi:r /etc/shadow

[root@server0 ~]# mkdir /nsd01

[root@server0 ~]# chmod o=--- /nsd01

[root@server0 ~]# ls -ld /nsd01

[root@server0 ~]# su - student

[student@server0 ~]$ cd /nsd01

-bash: cd: /nsd01: Permission denied

[student@server0 ~]$ exit

logout

[root@server0 ~]# setfacl -m u:student:rx /nsd01

[root@server0 ~]# getfacl /nsd01 #查看ACL策略

[root@server0 ~]# su - student

[student@server0 ~]$ cd /nsd01

[student@server0 nsd01]$ pwd

#####################################################

[root@server0 ~]# mkdir /nsd02

[root@server0 ~]# setfacl -m u:student:rwx /nsd02

[root@server0 ~]# setfacl -m u:dc:rx /nsd02

[root@server0 ~]# setfacl -m u:lisi:rx /nsd02

[root@server0 ~]# setfacl -m u:zhangsan:rwx /nsd02

[root@server0 ~]# getfacl /nsd02

[root@server0 ~]# setfacl -x u:dc /nsd02 #删除指定ACL

[root@server0 ~]# getfacl /nsd02

[root@server0 ~]# setfacl -b /nsd02 #删除所有ACL策略

[root@server0 ~]# getfacl /nsd02

###################################################

设置黑名单

[root@server0 ~]# mkdir /public

[root@server0 ~]# chmod ugo=rwx /public

[root@server0 ~]# ls -ld /public

[root@server0 ~]# setfacl -m u:lisi:--- /public

[root@server0 ~]# getfacl /public

[root@server0 ~]# su - lisi

[lisi@server0 ~]$ cd /public/

-bash: cd: /public/: Permission denied

[lisi@server0 ~]$ ls /public/

ls: cannot open directory /public/: Permission denied

[lisi@server0 ~]$ exit

###################################################

软件包管理

一 零散软件管理

1.前提具备软件包

2.虚拟机classroom搭建Web服务器,共享RHEL7.0光盘所有内容

Web服务:提供网页内容的服务

3.访问虚拟机classroom网页

真机利用火狐浏览器: classroom.example.com

classroom.example.com/content/rhel7.0/x86\_64/dvd/

4.下载软件包到虚拟机server:

• 使用wget下载工具

– wget 软件包的URL网址

– wget 软件包的URL网址 -O /目录路径/新文件名

Packages目录中

在火狐浏览器中,按键盘 Ctrl + f 进行当前页面搜索

[root@server0 ~]# wget 软件包的网络路径

[root@server0 ~]# wget http://classroom.example.com/content/rhel7.0/x86\_64/dvd/Packages/vsftpd-3.0.2-9.el7.x86\_64.rpm

[root@server0 ~]# ls

使用rpm命令管理软件

• RPM Package Manager,RPM包管理器

– rpm -q 软件名...

– rpm -ivh 软件名-版本信息.rpm...

– rpm -e 软件名...

1 查询软件是否安装

[root@server0 ~]# rpm -q firefox

[root@server0 ~]# rpm -q zip

[root@server0 ~]# rpm -q hostname

[root@server0 ~]# rpm -q vsftpd #查询软件软件安装

[root@server0 ~]# rpm -q haha

2.安装软件包

[root@server0 ~]# rpm -q vsftpd

未安装软件包 vsftpd

]# rpm -ivh /root/vsftpd-3.0.2-9.el7.x86\_64.rpm

[root@server0 ~]# rpm -q vsftpd

[root@server0 ~]# rpm -e vsftpd #卸载

[root@server0 ~]# rpm -q vsftpd #查询是否卸载成功

未安装软件包 vsftpd

[root@server0 ~]#

补充内容:

[root@server0 ~]# rpm -ql vsftpd #已安装软件的清单

了解:导入红帽签名信息

]# wget http://classroom.example.com/content/rhel7.0/x86\_64/dvd/RPM-GPG-KEY-redhat-release

]# ls

]# rpm --import /root/RPM-GPG-KEY-redhat-release

]# rpm -e vsftpd

]# rpm -ivh /root/vsftpd-3.0.2-9.el7.x86\_64.rpm

##################################################

常见报错信息:

]# rpm -ivh /root/bind-chroot-9.9.4-14.el7.x86\_64.rpm

错误：依赖检测失败：

bind = 32:9.9.4-14.el7 被 bind-chroot-32:9.9.4-14.el7.x86\_64 需要

#################################################

Yum软件包仓库:自动解决依赖关系安装软件

构建Yum服务

服务:自动解决依赖关系安装软件

服务端: 1.众多的软件包 2.仓库数据文件 3.构建FTP或Web服务

都在光盘内容里

虚拟机classroom:构建Web服务,共享光盘所有内容

客户端: 虚拟机server 指定服务端位置

书写配置文件: /etc/yum.repos.d/\*.repo

错误的文件会影响正确的配置文件

[root@server0 /]# rm -rf /etc/yum.repos.d/\*

[root@server0 /]# vim /etc/yum.repos.d/dvd.repo

[rhel7] #仓库标识

name=rhel7.0 #仓库描述信息

baseurl=http://classroom.example.com/content/rhel7.0/x86\_64/dvd/ #指定服务端位置

enabled=1 #是否启用本文件

gpgcheck=0 #是否检测软件包的红帽签名

[root@server0 /]# yum repolist #显示仓库基本信息

Yum仓库的使用:

[root@server0 /]# yum -y install httpd

[root@server0 /]# yum -y install sssd

[root@server0 /]# yum -y install gcc

[root@server0 /]# yum -y install system-config-kickstart

[root@server0 /]# yum -y install tftp-server

卸载软件包:

[root@server0 /]# yum remove httpd

[root@server0 /]# yum remove gcc

[root@server0 /]# yum remove sssd

清空Yum缓存:

[root@server0 /]# yum repolist

[root@server0 /]# yum clean all #清空Yum缓存

[root@server0 /]# yum repolist

#################################################

升级内核

1.下载新的内核

[root@server0 ~]# wget http://classroom.example.com/content/rhel7.0/x86\_64/errata/Packages/kernel-3.10.0-123.1.2.el7.x86\_64.rpm

[root@server0 ~]# ls

2.查看当前系统内核版本

[root@server0 ~]# uname -r

3.10.0-123.el7.x86\_64

3.安装新的内核软件包

]# rpm -ivh /root/kernel-3.10.0-123.1.2.el7.x86\_64.rpm

4.重启虚拟机server验证:

[root@server0 ~]# reboot

Connection to 172.25.0.11 closed by remote host.

Connection to 172.25.0.11 closed.

[student@room9pc01 ~]$ gos

[root@server0 ~]# uname -r

3.10.0-123.1.2.el7.x86\_64

[root@server0 ~]#

####################################################

分区规划及使用

扇区默认的大小512字节

• 识别硬盘 => 分区规划 => 格式化 => 挂载使用

毛坯楼层 => 打隔断 => 装修 =>入驻

一 识别硬盘

[root@server0 ~]# lsblk

NAME SIZE TYPE MOUNTPOINT

vda 10G disk

└─vda1 10G part /

vdb 10G disk

[root@server0 ~]# ls /dev/vdb

关机添加一块新的硬盘60G硬盘

[root@server0 ~]# poweroff

Connection to 172.25.0.11 closed by remote host.

Connection to 172.25.0.11 closed.

[student@room9pc01 ~]$ gos

[root@server0 ~]# lsblk

[root@server0 ~]# ls /dev/vdc

二 分区规划

分区模式:

MBR主引导记录的分区模式

– 最大支持容量为 2.2TB 的磁盘

– 1~4个主分区,或者 3个主分区+1个扩展分区(n个逻辑分区)

– 扩展分区不能格式化

• 修改硬盘的分区表

[root@server0 ~]# fdisk /dev/vdb

n 创建新的分区----->回车----->回车---->回车----->在last结束时 +2G

p 查看分区表

n 创建新的分区----->回车----->回车---->回车----->在last结束时 +1G

d 删除分区

w 保存并退出

[root@server0 ~]# lsblk

[root@server0 ~]# ls /dev/vdb[1-2]

/dev/sda5设备表示的含义:

第一块SCSI接口的硬盘,第5个分区

第一块SCSI接口的硬盘,第1个逻辑分区

三 格式化:赋予空间文件系统的过程

文件系统:空间存储数据的规则

Windows: NTFS FAT32

Linux: ext4(RHEL6) xfs(RHEL7)

[root@server0 ~]# mkfs.ext4 /dev/vdb1 #格式化ext4文件系统

[root@server0 ~]# blkid /dev/vdb1 #查看分区文件系统

[root@server0 ~]# mkfs.xfs /dev/vdb2 #格式化xfs文件系统

[root@server0 ~]# blkid /dev/vdb2 #查看分区文件系统

四 挂载使用

[root@server0 ~]# mkdir /mypart1

[root@server0 ~]# mount /dev/vdb1 /mypart1

[root@server0 ~]# df -h #查看正在挂载的设备,使用情况

[root@server0 ~]# mkdir /mypart2

[root@server0 ~]# mount /dev/vdb2 /mypart2

[root@server0 ~]# df -h #查看正在挂载的设备,使用情况

五 开机自动挂载

• 配置文件 /etc/fstab 的记录格式

设备路径 挂载点 类型 参数 备份标记 检测顺序

[root@server0 ~]# vim /etc/fstab

#命令模式 按 o可以另起新的一行进入 插入模式

.......

/dev/vdb1 /mypart1 ext4 defaults 0 0

/dev/vdb2 /mypart2 xfs defaults 0 0

[root@server0 ~]# umount /mypart1

[root@server0 ~]# umount /mypart2

[root@server0 ~]# df -h

[root@server0 ~]# mount -a

检测/etc/fstab开机自动挂载配置文件,格式是否正确

检测/etc/fstab中,书写完成,但当前没有挂载的设备,进行挂载

[root@server0 ~]# df -h

六 综合分区

综合分区，/dev/vdb继续分区

最终有3个主分区，分别为2G、1G、1G

创建扩展分区 ---> 两个逻辑分区，分别为1G、1G

[root@server0 ~]# fdisk /dev/vdb

p 查看分区表

n 创建主分区----->回车----->回车---->回车----->在last结束时 +1G

p 查看分区表

n 创建扩展分区

----->回车---->起始回车----->结束回车 将所有空间给扩展分区

p 查看分区表

n 创建逻辑分区----->起始回车------>结束+1G

n 创建逻辑分区----->起始回车------>结束+1G

p 查看分区表

w 保存并退出

[root@server0 ~]# partprobe #刷新分区表

[root@server0 ~]# lsblk

####################################################

总结：

1.识别硬盘　lsblk

2.分区规划 fdisk

3.刷新分区表　　partprobe

4.进行格式化　　mkfs.xfs mkfs.ext4 blkid

5.挂载使用　mount /etc/fstab开机自动挂载 mount -a df -h

## Day07

图形添加一块硬盘为80G

[root@server0 ~]# lsblk

NAME SIZE TYPE MOUNTPOINT

vda 10G disk

└─vda1 10G part /

vdb 10G disk

vdc 80G disk

[root@server0 ~]# ls /dev/vdc

综合分区:

划分三个主分区10G,一个扩展分区,二个逻辑分区10G

[root@server0 ~]# fdisk /dev/vdc

n 创建主分区----->回车----->回车---->回车----->在last结束时 +10G

连续划分三个主分区

p 查看分区表

n 创建扩展分区

----->回车---->起始回车----->结束回车 将所有空间给扩展分区

p 查看分区表

n 创建逻辑分区----->起始回车------>结束+10G

n 创建逻辑分区----->起始回车------>结束+10G

p 查看分区表

w 保存并退出

[root@server0 ~]# lsblk

[root@server0 ~]# ls /dev/vdc[1-6]

###################################################

LVM逻辑卷

作用:1.整合分散的空间 2.空间可以扩大

– 零散空闲存储 ---- 整合的虚拟磁盘 ---- 虚拟的分区

将众多的物理卷(PV),组建成卷组(VG),再从卷组中划分逻辑卷(LV)

LVM管理工具集

功能 物理卷管理 卷组管理 逻辑卷管理

Scan 扫描 pvs vgs lvs

Create 创建 pvcreate vgcreate lvcreate

Display 显示 pvdisplay vgdisplay lvdisplay

Remove 删除 pvremove vgremove lvremove

Extend 扩展 / vgextend lvextend

###################################################

创建逻辑卷

1.创建卷组

命令格式: vgcreate 卷组名 分区设备路径......

[root@server0 ~]# vgcreate systemvg /dev/vdc[1-2]

[root@server0 ~]# pvs #查看物理卷基本信息

[root@server0 ~]# vgs #查看卷组基本信息

2.创建逻辑卷

命令格式: lvcreate -n 逻辑卷名 -L 大小 基于的卷组名

[root@server0 ~]# lvcreate -n mylv -L 16G systemvg

[root@server0 ~]# vgs

[root@server0 ~]# lvs #查看逻辑卷基本信息

3.使用逻辑卷

[root@server0 ~]# ls /dev/dm-0

[root@server0 ~]# ls /dev/systemvg/mylv

[root@server0 ~]# ls -l /dev/systemvg/mylv

[root@server0 ~]# mkfs.xfs /dev/systemvg/mylv

[root@server0 ~]# blkid /dev/systemvg/mylv

[root@server0 ~]# vim /etc/fstab

/dev/systemvg/mylv /nsd xfs defaults 0 0

[root@server0 ~]# mkdir /nsd

[root@server0 ~]# mount -a

[root@server0 ~]# df -h

#################################################

逻辑卷扩大

一 卷组有足够的剩余空间

1.空间的扩展

[root@server0 ~]# lvs

[root@server0 ~]# vgs

[root@server0 ~]# lvextend -L 18G /dev/systemvg/mylv

[root@server0 ~]# lvs

2.文件系统的扩展

扩展xfs文件系统命令: xfs\_growfs

扩展ext4文件系统命令: resize2fs

[root@server0 ~]# df -h

[root@server0 ~]# xfs\_growfs /dev/systemvg/mylv

[root@server0 ~]# df -h

二 卷组没有足够的剩余空间

1.先扩展卷组的空间

[root@server0 ~]# vgextend systemvg /dev/vdb

[root@server0 ~]# vgs

2.逻辑卷空间的扩展

[root@server0 ~]# lvs

[root@server0 ~]# vgs

[root@server0 ~]# lvextend -L 25G /dev/systemvg/mylv

[root@server0 ~]# lvs

2.逻辑卷文件系统的扩展

[root@server0 ~]# df -h

[root@server0 ~]# xfs\_growfs /dev/systemvg/mylv

[root@server0 ~]# df -h

###################################################

补充: 逻辑卷也支持缩减

ext4文件系统支持缩减

xfs文件系统不支持缩减

##################################################

卷组划分空间的单位 PE

[root@server0 ~]# vgdisplay systemvg #显示卷组详细信息

PE Size 4.00 MiB

请划分一个大小为250M的逻辑卷

[root@server0 ~]# vgchange -s 1M systemvg #修改PE大小

[root@server0 ~]# vgdisplay systemvg #查看详细信息

[root@server0 ~]# lvcreate -n lvtest -L 250M systemvg

[root@server0 ~]# lvs

请划分一个为800个PE大小的逻辑卷 -l:指定PE的个数

[root@server0 ~]# lvcreate -n lvnsd -l 800 systemvg

[root@server0 ~]# lvs

###################################################

逻辑卷的删除

删除卷组前提:基于该卷组创建的,所有逻辑卷都要删除

[root@server0 ~]# lvremove /dev/systemvg/lvnsd

Do you really want to remove active logical volume lvnsd? [y/n]: y

Logical volume "lvnsd" successfully removed

[root@server0 ~]#

[root@server0 ~]# lvremove /dev/systemvg/lvtest

Do you really want to remove active logical volume lvtest? [y/n]: y

Logical volume "lvtest" successfully removed

[root@server0 ~]# lvremove /dev/systemvg/mylv

Logical volume systemvg/mylv contains a filesystem in use.

[root@server0 ~]# umount /nsd/

[root@server0 ~]# lvremove /dev/systemvg/mylv

Do you really want to remove active logical volume mylv? [y/n]: y

Logical volume "mylv" successfully removed

[root@server0 ~]# lvs

[root@server0 ~]# vgs

##################################################

find高级使用

– find [目录] [条件1] [-a|-o] [条件2] ...

– 常用条件表示:

-type 类型(f、d、l)

-name "文档名称"

-size +|-文件大小(k、M、G)

-user 用户名

-mtime 根据文件修改时间

###############################################

-type 类型(f文本文件、d目录、l快捷方式)

[root@server0 /]# find /boot -type l

[root@server0 /]# ls -l /boot/grub/menu.lst

[root@server0 /]# find /boot -type d

[root@server0 /]# find /boot -type f

[root@server0 /]# find /root -type d

[root@server0 /]# find /root -type f

-name '文档名称'

[root@server0 /]# find /etc/ -name '\*tab'

[root@server0 /]# find /etc/ -name 'passwd'

[root@server0 /]# find /etc/ -name 'passwd\*'

[root@server0 /]# find /etc/ -name '\*.conf'

[root@server0 /]# find /boot -name 'vm\*'

[root@server0 /]# mkdir /root/nsd01

[root@server0 /]# mkdir /root/nsd02

[root@server0 /]# touch /root/nsd03.txt

[root@server0 /]# find /root/ -name 'nsd\*'

[root@server0 /]# find /root/ -name 'nsd\*' -type f

[root@server0 /]# find /root/ -name 'nsd\*' -type d

两个条件满足其一即可

[root@server0 /]# find /root/ -name 'nsd\*' -o -type d

-size +|-文件大小(k、M、G)

[root@server0 /]# find /boot/ -size +10M

[root@server0 /]# ls -lh /boot/initramfs-\*

[root@server0 /]# find /boot/ -size -10M

-mtime 根据文件修改时间(都是过去时间)

+10:十天之前

-10:最近十天之内

[root@server0 /]# find /var/ -mtime +90

[root@server0 /]# find /var/ -mtime +1000

[root@server0 /]# find /root/ -mtime -2

-user 用户名

[root@server0 /]# find /home -user student

[root@server0 /]# ls -l /home

[root@server0 /]# find / -user student

[root@server0 /]# ls -l /var/spool/student

#################################################

find扩展使用

• 使用find命令的 -exec 操作

– find .. .. -exec 处理命令 {} \;

– 优势:以 {} 代替每一个结果,逐个处理,遇 \; 结束

]# find /boot/ -size +10M

]# find /boot/ -size +10M -exec cp {} /opt/ \;

]# ls /opt/

]# find / -user student -type f -exec cp {} /opt \;

]# ls -A /opt/

]# find /boot/ -name 'vm\*'

]# find /boot/ -name 'vm\*' -exec cp {} /opt/ \;

]# ls -A /opt/

##################################################

NTP时间同步

[root@server0 /]# date #查看系统时间

[root@server0 /]# date -s '2000-10-1 12:10:08'

[root@server0 /]# date

[root@server0 /]# date -s '年-月-日 时:分:秒' #修改时间格式

NTP网络时间协议

• Network Time Protocol

– NTP服务器为客户机提供标准时间

– NTP客户机需要与NTP服务器保持沟通

NTP时间服务器:虚拟机classroom

NTP客户机:虚拟机server

1.安装chrony软件包,与时间服务器沟通软件

[root@server0 /]# yum -y install chrony

[root@server0 /]# rpm -q chrony

[root@server0 /]# rpm -ql chrony

/etc/NetworkManager/dispatcher.d/20-chrony

/etc/chrony.conf

2.修改配置/etc/chrony.conf,指定时间服务器位置

[root@server0 /]# vim /etc/chrony.conf

以#开头的为注释行

#server 0.rhel.pool.ntp.org iburst

#server 1.rhel.pool.ntp.org iburst

#server 2.rhel.pool.ntp.org iburst

server classroom.example.com iburst #指定服务端位置

3.重启服务(程序)

[root@server0 /]# systemctl restart chronyd

[root@server0 /]# systemctl enable chronyd #开机自启动

daemon

英 [ˈdiːmən] 美 [ˈdiːmən]

(古希腊神话中的)半神半人精灵

守护进程

4.验证:

[root@server0 /]# date -s '2000-1-1 1:1:1'

2000年 01月 01日 星期六 01:01:01 CST

[root@server0 /]# systemctl restart chronyd

[root@server0 /]# date

2000年 01月 01日 星期六 01:01:58 CST

[root@server0 /]# date

2019年 05月 09日 星期四 16:19:00 CST

###################################################

grep补充

^$:匹配空行

[root@server0 /]# grep ^root /etc/passwd

[root@server0 /]# grep bash$ /etc/passwd

显示/etc/login.defs文件有效信息(去除注释 去除空行)

]# grep -v ^# /etc/login.defs

]# grep -v ^# /etc/login.defs | grep -v ^$

]# grep -v ^# /etc/login.defs | grep -v ^$ > /opt/1.txt

################################################

修改用户家目录

[root@server0 /]# useradd tom

[root@server0 /]# id tom

[root@server0 /]# grep tom /etc/passwd

[root@server0 /]# usermod -d /opt/abc tom

[root@server0 /]# grep tom /etc/passwd

[root@server0 /]# grep tom /etc/passwd

[root@server0 /]# ls /opt/abc #默认没有创建该用户的家目录

[root@server0 /]# cp -r /home/tom /opt/abc

[root@server0 /]# chown -R tom:tom /opt/abc

[root@server0 /]# su - tom

[tom@server0 ~]$ exit

###########################################

## Day01

什么是Shell脚本

• 提前设计可执行语句,用来完成特定任务的文件

– 解释型程序

– 顺序、批量执行

案例1:Shell脚本的编写及测试

1. 编写一个面世问候 /root/helloworld.sh 脚本

– 显示出一段话 Hello World

[root@server0 ~]# vim /root/hello.sh

echo Hello World

[root@server0 ~]# chmod +x /root/hello.sh #赋予执行权限

[root@server0 ~]# /root/hello.sh #绝对路径运行

规范Shell脚本的一般组成

• #! 环境声明,以下代码由那个程序进行解释

• # 注释文本

• 可执行代码

1)输出当前红帽系统的版本信息

2)输出当前使用的内核版本

3)输出当前系统的主机名

[root@server0 ~]# vim /root/hello.sh

#!/bin/bash

echo hello world

cat /etc/redhat-release

uname -r

hostname

ifconfig | head -2

[root@server0 ~]# /root/hello.sh

#################################################

简单脚本技巧

重定向输出

>:只收集前面命令的正确输出信息

2>:只收集前面命令的错误输出信息

&>:收集前面命令的正确与错误信息

[root@server0 ~]# cat /etc/redhat-release

[root@server0 ~]# cat /xixi

[root@server0 ~]# cat /etc/redhat-release /xixi

]# cat /etc/redhat-release /xixi > /opt/1.txt

]# cat /opt/1.txt

]# cat /etc/redhat-release /xixi 2> /opt/2.txt

]# cat /opt/2.txt

]# cat /etc/redhat-release /xixi &> /opt/3.txt

]# cat /opt/3.txt

####################################################

案例:书写一个创建用户设置密码的脚本

/dev/null:黑洞设备

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

useradd nsd04 &> /dev/null

echo 用户nsd04创建成功

echo 123 | passwd --stdin nsd04 &> /dev/null

echo 用户nsd04密码设置成功

[root@server0 ~]# chmod +x /root/user.sh

[root@server0 ~]# /root/user.sh

为了让脚本,适应多变需求,所以使用变量

变量:会变化的量 类似容器

以不变的名称,存储可以变化的值

定义变量:

变量名=存储的值

使用变量储存的值: $变量名

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

a=dc

useradd $a &> /dev/null

echo 用户$a创建成功

echo 123 | passwd --stdin $a &> /dev/null

echo 用户$a密码设置成功

提高用户体验度,产生交互

read

1.直接产生交互,等待用户输入

2.记录用户在键盘上的输入

3.将记录的内容,赋值给一个变量储存

-p '屏幕提示信息'

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

read -p '请您输入要创建的用户名:' a

read -p '请您输入要设置的密码:' b

useradd $a &> /dev/null

echo 用户$a创建成功

echo $b | passwd --stdin $a &> /dev/null

echo 用户$a密码设置成功

[root@server0 ~]#

什么是变量

• 以不变的名称存放的可能会变化的值

– 变量名=变量值

– 方便以固定名称重复使用某个值

– 提高对任务需求、运行环境变化的适应能力

定义/赋值变量

• 设置变量时的注意事项

– 等号两边不要有空格

– 变量名只能由字母/数字/下划线组成,区分大小写

– 变量名不能以数字开头,不要使用关键字和特殊字符

– 若指定的变量名已存在,相当于为此变量重新赋值

以下有效变量名为: B

A.2008file B.tun\_01 C a-b D a\_ccd\_9-abc

查看/引用变量

• 基本格式

– 引用变量值:$变量名

– 查看变量值:echo $变量名、echo ${变量名}

[root@server0 ~]# a=rhel

[root@server0 ~]# echo $a

rhel

[root@server0 ~]# echo $a7

[root@server0 ~]# echo ${a}7

rhel7

################################################

单引号' ':取消所有特殊字符的意义,原样输出

[root@server0 ~]# echo '\*\*\*\*\*\*\*\*'

\*\*\*\*\*\*\*\*

[root@server0 ~]# echo '| ? \* {} [] > '

| ? \* {} [] >

[root@server0 ~]#

反撇号 ` ` 或 $( )

:将命令的输出,直接参与另一条命令的运行,作为另一条命令的参数

[root@server0 ~]# mkdir /opt/nsd`date +%F`

[root@server0 ~]# ls /opt/

[root@server0 ~]# mkdir /opt/`hostname`-`date +%F`

[root@server0 ~]# ls /opt/

[root@server0 ~]# mkdir /opt/abc-$(date +%F)

[root@server0 ~]# ls /opt/

###################################################

环境变量

变量名为大写字母,由系统定义并且赋值完成

USER=当前登录的用户名

[root@server0 ~]# echo $USER

root

[root@server0 ~]# su - student

[student@server0 ~]$ echo $USER

student

[student@server0 ~]$ exit

logout

[root@server0 ~]# su - nsd01

[nsd01@server0 ~]$ echo $USER

nsd01

[nsd01@server0 ~]$

#################################################

位置变量

• 由系统定义并且赋值完成

• 在执行脚本时提供的命令行参数

– 表示为 $n,n为序号

– $1、$2、.. .. ${10}、${11}、.. ..

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

#read -p '请您输入要创建的用户名:' a

#read -p '请您输入要设置的密码:' b

useradd $1 &> /dev/null

echo 用户$1创建成功

echo 123 | passwd --stdin $1 &> /dev/null

echo 用户$1密码设置成功

[root@server0 ~]# /root/user.sh natasha

[root@server0 ~]# vim /root/1.sh

#!/bin/bash

cat -n $1 | head -$2

echo 您输入的文件为$1

echo 显示文件内容的头$2行

[root@server0 ~]# /root/1.sh /etc/passwd 3

####################################################

预定义变量

• 由系统定义并且赋值完成

$# 已加载的位置变量的个数

$? 程序退出后的状态值,0表示正常,其他值异常

[root@server0 ~]# vim /root/2.sh

#!/bin/bash

echo $1

echo $2

echo $3

echo $#

[root@server0 ~]# chmod +x /root/2.sh

[root@server0 ~]# /root/2.sh haha xixi hehe lele

###################################################

条件测试

[ 测试表达式 ] 每一部分都要有空格

常用的测试选项

• 检查文件状态

-e:文档存在为真

-d:文档存在,且必须为目录才为真

-f:文档存在,且必须为文件才为真

[root@server0 ~]# [ -e /etc ]

[root@server0 ~]# echo $?

0

[root@server0 ~]# [ -e /xixi ]

[root@server0 ~]# echo $?

1

[root@server0 ~]# [ -d /etc/passwd ]

[root@server0 ~]# echo $?

1

[root@server0 ~]# [ -f /etc/passwd ]

[root@server0 ~]# echo $?

0

[root@server0 ~]#

• 比较整数大小

-gt:大于

-ge:大于等于

-eq:等于

-ne:不等于

-lt:小于

-le:小于等于

[root@server0 ~]# [ 1 -eq 2 ]

[root@server0 ~]# echo $?

1

[root@server0 ~]# [ 1 -eq 1 ]

[root@server0 ~]# echo $?

0

[root@server0 ~]# [ 10 -gt 9 ]

[root@server0 ~]# echo $?

0

[root@server0 ~]# [ 9 -ge 9 ]

[root@server0 ~]# echo $?

• 字符串比对

==:一致为真

!=:不一致为真

[root@server0 ~]# [ student == dc ]

[root@server0 ~]# echo $?

1

[root@server0 ~]# [ student == root ]

[root@server0 ~]# echo $?

1

[root@server0 ~]# [ student != root ]

[root@server0 ~]# echo $?

0

[root@server0 ~]# [ $USER == root ]

[root@server0 ~]# echo $?

0

[root@server0 ~]#

##################################################

if双分支处理

if [条件测试];then

命令序列xx

else

命令序列yy

fi

[root@server0 ~]# vim /root/if01.sh

#!/bin/bash

if [ $1 -eq 1 ];then

echo hello

else

echo hi

fi

[root@server0 ~]# /root/if01.sh 1

[root@server0 ~]# /root/if01.sh 2

案例:利用read获取,用户所输入的用户名

进行判断,如果用户名当前系统已存在,则输出 用户已存在

进行判断,如果用户名当前系统不存在,则输出 用户不存在

[root@server0 ~]# vim /root/if02.sh

#!/bin/bash

read -p '请输入您要测试的用户名:' a

id $a &> /dev/null

if [ $? -eq 0 ];then

echo $a用户已存在

else

echo $a用户不存在

fi

[root@server0 ~]#

案例:利用read获取,用户所输入的IP地址

进行判断,如果当前系统可以与该IP地址通信,则输出 可以ping通

进行判断,如果当前系统不可以与该IP地址通信,则输出 不可以ping通

[root@server0 ~]# vim /root/if03.sh

#!/bin/bash

read -p '请输入您要测试的IP地址:' ip

ping -c 2 $ip &> /dev/null

if [ $? -eq 0 ];then

echo $ip可以通信

else

echo $ip不可以通信

fi

##################################################

if多分支处理

if [条件测试1];then

命令序列xx

elif [条件测试2];then

命令序列yy

elif [条件测试3];then

命令序列aa

..........

else

命令序列zz

fi

案例:利用read获取,用户所输入的成绩

进行判断

如果大于等于90,则输出 优秀

如果大于等于80,则输出 良好

如果大于等于70,则输出 一般

如果大于等于60,则输出 合格

以上条件均不满足,则输出 再牛的肖邦,也弹不出哥的悲伤

[root@server0 ~]# vim /root/if04.sh

#!/bin/bash

read -p '请输入您的成绩:' num

if [ $num -ge 90 ];then

echo 优秀

elif [ $num -ge 80 ];then

echo 良好

elif [ $num -ge 70 ];then

echo 一般

elif [ $num -ge 60 ];then

echo 合格

else

echo '再牛的肖邦,也弹不出哥的悲伤!'

fi

####################################################

列表式循环,反复执行类似的操作

for循环处理

• 遍历/列表式循环

– 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

for 献血车 in zhangsan lisi wangwu

do

抽血

done

[root@server0 ~]# vim /root/for01.sh

#!/bin/bash

for a in zhangsan lisi wangwu natasha harry jack

do

useradd $a &> /dev/null

echo $a创建成功

done

[root@server0 ~]#

列表值可以不参与循环执行的操作

造数工具: {起始..结束}

{1..20}:制造1到20之间连续的数字

[root@server0 ~]# vim /root/for01.sh

#!/bin/bash

for a in {1..20}

do

echo Hello World

done

###############################################

案例4:编写一个判断脚本

在 server0 上创建 /root/foo.sh 脚本

1)当运行/root/foo.sh redhat,输出为fedora

2)当运行/root/foo.sh fedora,输出为redhat

3)当没有任何参数或者参数不是 redhat 或者fedora时,

其错误输出产生以下信息:

/root/foo.sh redhat|fedora

[root@server0 ~]# vim /root/foo.sh

#!/bin/bash

if [ $# -eq 0 ];then #判断用户是否输入参数

echo '/root/foo.sh redhat|fedora' >&2 #将正确输出变成错误输出

exit 38 #脚本退出返回值

elif [ $1 == redhat ];then #判断参数是否是redhat

echo fedora

elif [ $1 == fedora ];then #判断参数是否是fedora

echo redhat

else

echo '/root/foo.sh redhat|fedora' >&2 #将正确输出变成错误输出

exit 48 #脚本退出返回值

fi

##############################################

## Day02

案例5:编写一个批量添加用户脚本

在 server0 上创建 /root/batchusers 脚本

1)此脚本要求提供用户名列表文件作为参数

2)如果没有提供参数,此脚本应该给出提示

Usage: /root/batchusers,退出并返回相应值

3)如果提供一个不存在的文件,此脚本应该给出提

示 Input file not found,退出并返回相应值

4)新用户的登录Shell为 /bin/false,无需设置密码

5)用户列表测试文件:

http://classroom.example.com/pub/materials/userlist

[root@server0 ~]# vim /root/batchusers

#!/bin/bash

if [ $# -eq 0 ];then #判断用户是否输入参数

echo 'Usage: /root/batchusers' >&2

exit 2

elif [ -f $1 ];then　　　　　　　　＃判断用户输入的文件是否存在

for a in $(cat $1)

do

useradd -s /bin/false $a &> /dev/null

echo $a创建成功

done

else

echo 'Input file not found' >&2

exit 3

fi

[root@server0 ~]# /root/batchusers /root/userlist

###########################################################

系统安全保护

SELinux安全机制

• Security-Enhanced Linux

– 美国NSA国家安全局主导开发,一套增强Linux系统安

全的强制访问控制体系

– 集成到Linux内核(2.6及以上)中运行

– RHEL7基于SELinux体系针对用户、进程、目录和文件

提供了预设的保护策略,以及管理工具

• SELinux的运行模式

– enforcing(强制)、permissive(宽松)

– disabled(彻底禁用)

所有状态变成disabled(彻底禁用),都要经历重启系统

• 切换运行模式

– 临时切换:setenforce 1|0

– 固定配置:/etc/selinux/config 文件

虚拟机server

[root@server0 ~]# getenforce #查看SELinux当前系统状态

Enforcing

[root@server0 ~]# setenforce 0 #修改SELinux状态,为宽松模式

[root@server0 ~]# getenforce

Permissive

[root@server0 ~]# vim /etc/selinux/config#SELinux状态永久配置文件

SELINUX=permissive

虚拟机desktop

[root@desktop0 ~]# getenforce

Enforcing

[root@desktop0 ~]# setenforce 0

[root@desktop0 ~]# getenforce

Permissive

[root@desktop0 ~]# vim /etc/selinux/config

SELINUX=permissive

##########################################################

用户个性化配置文件

• 影响指定用户的 bash 解释环境

– ~/.bashrc,每次开启 bash 终端时生效

全局环境配置

• 影响所有用户的 bash 解释环境

– /etc/bashrc,每次开启 bash 终端时生效

虚拟机server0

[root@server0 ~]# vim /home/student/.bashrc

alias hello='echo hello'

[root@server0 ~]# vim /root/.bashrc

alias hi='echo hi'

[root@server0 ~]# vim /etc/bashrc

alias dc='echo dc'

#######################################################

搭建基本Web服务,提供一个网页内容

http协议:超文本传输协议,专用于传输页面内容

服务端:虚拟机server

1.安装httpd软件包

[root@server0 ~]# yum -y install httpd

2.重启httpd服务

[root@server0 ~]# systemctl restart httpd

[root@server0 ~]# systemctl enable httpd

ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service'

3.本机测试

[root@server0 ~]# firefox 172.25.0.11

4.书写自己的页面

默认存放文件文件路径:/var/www/html/

默认网页文件名称:index.html

[root@server0 ~]# cat /var/www/html/index.html

<marquee><font color=red><h1>NSD1904hahaxixi

滚动 字体颜色 标题字体

[root@server0 ~]#

#######################################################

搭建基本的FTP服务

FTP:文件传输协议

默认FTP服务共享位置:/var/ftp/

虚拟机server

1.安装vsftpd软件

[root@server0 ~]# yum -y install vsftpd

2.重启vsftpd服务

[root@server0 ~]# systemctl restart vsftpd

[root@server0 ~]# systemctl enable vsftpd

3.本机验证

[root@server0 ~]# firefox ftp://172.25.0.11

##########################################################

防火墙策略管理

作用:隔离,进行入站过滤

硬件防火墙

软件防火墙

firewalld服务基础

Linux的防火墙体系

• 系统服务:firewalld

• 管理工具:firewall-cmd、firewall-config

预设安全区域

• 根据所在的网络场所区分,预设保护规则集

– public:仅允许访问本机的sshd dhcp ping

– trusted:允许任何访问

– block:拒绝任何来访请求(明确拒绝回应)

– drop:丢弃任何来访的数据包(直接丢弃不给回应,节省服务器资源)

数据包: 源IP地址 目标IP地址 数据

防火墙的判定规则

1.查看请求数据包,中客户端来源IP地址,查看自己所有区域规则,如果某一个区域有该源IP地址的规则,则进入该区域

2.进入默认区域public

#######################################################

防火墙默认区域修改

虚拟机server

]# firewall-cmd --get-default-zone #查看默认区域

虚拟机desktop

]# ping 172.25.0.11 #可以通信

虚拟机server

]# firewall-cmd --set-default-zone=block #修改默认区域为block

]# firewall-cmd --get-default-zone #查看默认区域

虚拟机desktop

]# ping 172.25.0.11 #不可以通信,有回应

虚拟机server

]# firewall-cmd --set-default-zone=drop #修改默认区域为drop

]# firewall-cmd --get-default-zone #查看默认区域

虚拟机desktop

]# ping 172.25.0.11 #不可以通信,没有回应

###########################################################

常见的互联网协议:

http:超文本传输协议 默认端口:80

FTP:文本传输协议 默认端口:21

https:安全的超文本传输协议 默认端口:443

DNS:域名解析协议 默认端口:53

telnet:远程管理协议 默认端口:23

tftp:简单的文本传输协议 默认端口:69

smtp:发邮件协议 默认端口:25

pop3:收邮件协议 默认端口:110

snmp:简单的网络管理协议 默认端口:161

在默认区域添加协议:

虚拟机server:

]# firewall-cmd --set-default-zone=public #修改默认区域为public

]# firewall-cmd --get-default-zone

]# firewall-cmd --zone=public --list-all #查看区域详细规则

虚拟机desktop

]# firefox 172.25.0.11 #失败

]# firefox ftp://172.25.0.11 #失败

虚拟机server:

]# firewall-cmd --zone=public --list-all

]# firewall-cmd --zone=public --add-service=http #添加允许的协议

]# firewall-cmd --zone=public --add-service=ftp #添加允许的协议

]# firewall-cmd --zone=public --list-all

虚拟机desktop

]# firefox 172.25.0.11 #成功

]# firefox ftp://172.25.0.11 #成功

##########################################################

防火墙永久策略

– 永久(permanent)

]# firewall-cmd --reload #重新加载防火墙所有配置

]# firewall-cmd --permanent --zone=public --add-service=http

]# firewall-cmd --permanent --zone=public --add-service=ftp

]# firewall-cmd --zone=public --list-all

]# firewall-cmd --reload #重新加载防火墙所有配置

]# firewall-cmd --zone=public --list-all

#########################################################

单独拒绝虚拟机desktop的访问

]# firewall-cmd --zone=block --add-source=172.25.0.10

]# firewall-cmd --zone=block --list-all

#########################################################

端口:编号 标识协议或服务 可以具备多个

由root指定

数据包: 源IP地址 目标IP地址 端口号 数据

实现本机的端口映射

• 本地应用的端口重定向(端口1 --> 端口2)

– 从客户机访问 端口1 的请求,自动映射到本机 端口2

– 比如,访问以下两个地址可以看到相同的页面:

http://172.25.0.11:5423 ---> http://172.25.0.11:80

虚拟机server

]# firewall-cmd --permanent --zone=public --add-forward-port=port=5423:proto=tcp:toport=80

]# firewall-cmd --reload

]# firewall-cmd --zone=public --list-all

虚拟机desktop

]# firefox 172.25.0.11:5423

#########################################################

## Day03

一 环境设置

虚拟机server

[root@server0 ~]# firewall-cmd --set-default-zone=trusted

[root@server0 ~]# firewall-cmd --get-default-zone

虚拟机desktop

[root@desktop0 ~]# firewall-cmd --set-default-zone=trusted

[root@desktop0 ~]# firewall-cmd --get-default-zone

##########################################################

iSCSI网络磁盘

• Internet SCSI,网际SCSI接口(默认端口:3260)

– 一种基于C/S架构的虚拟磁盘技术

– 服务器提供磁盘空间,客户机连接并当成本地磁盘使用

一 划分一个主分区3G

[root@server0 ~]# fdisk /dev/vdb

[root@server0 ~]# lsblk

[root@server0 ~]# ls /dev/vdb1

二 服务端选择虚拟机server,构建iSCSI共享存储

• backstore,后端存储

– 对应到服务端提供实际存储空间的设备,需要起一个管理名称

• target,磁盘组

– 是客户端的访问目标,作为一个框架,由多个lun组成

• ISCSI Qualified Name(iqn) 名称规范

– iqn.yyyy-mm.倒序域名:自定义标识

– 用来识别 target 磁盘组,也用来识别客户机身份

• 名称示例

– iqn.2016-02.com.example:server0

• lun,逻辑单元

– 每一个lun需要关联到某一个后端存储设备,在客户端会视为一块虚拟硬盘

冰箱------->纸质的包装 木质的箱子

1.安装targetcli软件包

[root@server0 ~]# yum -y install targetcli

2.配置

a)建立backstore,后端存储(冰箱)

[root@server0 ~]# targetcli

/> ls

/> backstores/block create name=nsd dev=/dev/vdb1

/> ls

b)建立target磁盘组(建立木质的箱子)

iqn.yyyy-mm.倒序域名:自定义标识

/> iscsi/ create iqn.2019-05.com.example:server0

/> ls

c)进行lun关联(冰箱----->木质的箱子)

/> iscsi/iqn.2019-05.com.example:server0/tpg1/luns create /backstores/block/nsd

/> ls

d)设置访问控制ACL,设置客户端声称的名字(符合iqn命名规范)

/> iscsi/iqn.2019-05.com.example:server0/tpg1/acls create

iqn.2019-05.com.example:abc

/> ls

e)指定本机开放共享服务的IP地址及端口

/> iscsi/iqn.2019-05.com.example:server0/tpg1/portals create 172.25.0.11

/> ls

[root@server0 ~]# systemctl restart target #重启服务

[root@server0 ~]# systemctl enable target #设置开机自动

ln -s '/usr/lib/systemd/system/target.service' '/etc/systemd/system/multi-user.target.wants/target.service'

[root@server0 ~]#

###########################################################

虚拟机desktop:客户端

1.安装客户端软件

• 所需软件包:iscsi-initiator-utils

Yum安装软件时可以补全包名:

前提1:必须有Yum缓存, yum repolist 可以直接生成缓存

前提2:必须是当前系统没有安装

[root@desktop0 ~]# rpm -q iscsi-initiator-utils

iscsi-initiator-utils-6.2.0.873-21.el7.x86\_64

[root@desktop0 ~]# yum -y install iscsi(tab)

2.配置客户端声称的名字

[root@desktop0 ~]# vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2019-05.com.example:abc

3.客户端刷新声称名字的服务

[root@desktop0 ~]# systemctl restart iscsid

Warning: Unit file of iscsid.service changed on

disk, 'systemctl daemon-reload' recommended.

[root@desktop0 ~]# systemctl daemon-reload

[root@desktop0 ~]# systemctl restart iscsid

4.发现服务端共享存储

Linux查看命令帮助的方法 : man iscsiadm

全文查找/example 按n项下跳转匹配,看到EXAMPLES

Ctrl - ：减小字体

Ctrl Shift + ：变大字体

[root@desktop0 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 172.25.0.11 --discover

172.25.0.11:3260,1 iqn.2019-05.com.example:server0

5.使用服务端共享存储

[root@desktop0 ~]# lsblk

[root@desktop0 ~]# systemctl restart iscsi #重启客户端服务

[root@desktop0 ~]# lsblk

[root@desktop0 ~]# systemctl enable iscsi #设置开机自启动服务

#########################################################

总结:

服务端:

1.修改防火墙默认区域trusted

2.安装targetcli软件

3.运行targetcli命令进行配置

1)建立后端存储

2)建立target磁盘

3)进行lun关联

4)设置ACL访问控制

5)设置启用IP地址 默认端口为3260

4.重启target服务

客户端:

1.修改防火墙默认区域trusted

2.所需软件包:iscsi-initiator-utils

3.修改配置文件/etc/iscsi/initiatorname.iscsi

4.客户端刷新声称名字的服务 systemctl restart iscsid

5.发现共享存储

]# iscsiadm --mode discoverydb --type sendtargets --portal 172.25.0.11 --discover

6.使用共享存储

]# systemctl enable iscsi

########################################################

数据库服务基础

什么是数据库:存放数据的仓库

– 社区开源版 MariaDB

部署mariadb数据库服务器

• RHEL7 中的 MariaDB 相关包

– mariadb-server:提供服务端有关的系统程序

[root@server0 ~]# yum -y install mariadb-server

[root@server0 ~]# systemctl restart mariadb

在数据库中,有众多的库,在每一个库中有很多的表格

一 数据库的基本操作

[root@server0 ~]# mysql

MariaDB [(none)]> show databases; #显示所有库

MariaDB [(none)]> create database nsd01; #创建库nsd01

MariaDB [(none)]> show databases;

MariaDB [(none)]> drop database nsd01; #删除nsd01库

MariaDB [(none)]> show databases;

MariaDB [(none)]> create database nsd1904;

MariaDB [(none)]> show databases;

MariaDB [(none)]> exit

######################################################

二 为数据库管理员设置密码

数据库管理员root,针对数据库具有最高权限,信息存放mysql库中user表

系统管理员root,针对系统具有最高权限,信息存放/etc/passwd

命令格式: mysqladmin [-u用户名] [-p[旧密码]] password '新密码'

[root@server0 ~]# mysqladmin -u root password '123'

[root@server0 ~]# mysql -u root -p #交互式

Enter password:

[root@server0 ~]# mysql -u root -p123 #非交互式

已知旧密码修改新密码

[root@server0 ~]# mysqladmin -u root -p123 password '321'

#########################################################

数据库的主配置文件:/etc/my.cnf

#########################################################

• 导入/恢复到数据库

– mysql [-u用户名] [-p[密码]] 数据库名 < 备份文件.sql

]# wget http://classroom.example.com/pub/materials/users.sql

]# ls

]# mysql -u root -p123 nsd1904 < /root/users.sql

• 查看库中有哪些表格

[root@server0 ~]# mysql -u root -p123

MariaDB [(none)]> use nsd1904; #切换到nsd1904库

MariaDB [nsd1904]> show tables; #显示当前库有那些表格

表格: 表字段 表记录

编号 姓名 联系方式 居住地址

1 dc 138..... 东村

2 tc 138..... 西村

表格: 增(insert) 删(delete) 改(update) 查(select)

查(select)

格式: select 表字段,表字段..... from 库名.表名称

[root@server0 ~]# mysql -u root -p123

MariaDB [(none)]> use nsd1904; #进入nsd1904库

MariaDB [(none)]> show tables; #显示库中所有表格

MariaDB [(none)]> select \* from base;

MariaDB [(none)]> select \* from location;

MariaDB [(none)]> select name,password from base;

MariaDB [(none)]> use mysql;

MariaDB [(none)]> select \* from nsd1904.base;

########################################################

查看表结构: desc 表名字;

MariaDB [(none)]> desc nsd1904.base;

MariaDB [(none)]> desc mysql.user;

MariaDB [nsd1904]> select user,host,password from mysql.user;

########################################################

条件查询:

MariaDB [nsd1904]> use nsd1904;

MariaDB [nsd1904]> select \* from base;

MariaDB [nsd1904]> select \* from base where password='123';

MariaDB [nsd1904]> select \* from base where password='456';

MariaDB [nsd1904]> select \* from base where name='tom';

MariaDB [nsd1904]> select \* from base where id='4';

########################################################

除了root用户,此nsd1904库只能被用户lisi查询,此用户的密码为123

数据库的授权

GRANT 权限列表 ON 数据库名.表名 TO 用户名@客户机地址

IDENTIFIED BY '密码';

grant select on nsd1904.\* to lisi@localhost identified by '123';

当lisi通过本地localhost登录,输入密码为123,将会获得nsd1904库中所有表格的查询权限.

[root@server0 ~]# mysql -u root -p123

MariaDB [(none)]> grant select on nsd1904.\* to lisi@localhost identified by '123';

MariaDB [(none)]> select user,host,password from mysql.user;

MariaDB [(none)]> exit

[root@server0 ~]# mysql -u lisi -p123 #检测lisi登录

MariaDB [(none)]> show databases;

MariaDB [(none)]> exit

#########################################################

案例4:使用数据库查询

2. 在系统 server0 上使用数据库 nsd1904,并使用相

应的 SQL 查询以回答下列问题:

1)密码是 solicitous 的人的名字?

[root@server0 ~]# mysql -u root -p123

> use nsd1904;

> select \* from base;

> select \* from base where password='solicitous';

2)有多少人的姓名是 Barbara 同时居住在 Sunnyvale?

> select \* from base,location

where base.name='Barbara' and location.city='Sunnyvale' and

base.id=location.id ;

> select count(\*) from base,location

where base.name='Barbara' and location.city='Sunnyvale' and

base.id=location.id ;

> insert base values ('6','Barbara','789');

> select \* from base;

> insert location values ('6','Sunnyvale');

> select \* from location;

1. 禁止空密码root用户访问 mariadb 数据库

> use mysql;

> select user,host,password from user;

> select user,host,password from user where password='';

#删除表中password值为空的记录

> delete from user where password='';

> select user,host,password from user;

> flush privileges; #刷新user表记录

> exit

验证登录失败:

[root@server0 ~]# mysql -u root -h server0.example.com

ERROR 1130 (HY000): Host 'server0.example.com' is not allowed to connect to this MariaDB server

###############################################

## Day04

配置NFS共享

• Network File System,网络文件系统

– 用途:为客户机提供共享使用的文件夹

– 协议:NFS(TCP/UDP 2049)、RPC(TCP/UDP 111)

• 所需软件包:nfs-utils

• 系统服务:nfs-server

一 环境设置

虚拟机server

[root@server0 ~]# firewall-cmd --set-default-zone=trusted

[root@server0 ~]# firewall-cmd --get-default-zone

虚拟机desktop

[root@desktop0 ~]# firewall-cmd --set-default-zone=trusted

[root@desktop0 ~]# firewall-cmd --get-default-zone

二 虚拟机server构建NFS共享

1.安装nfs-utils软件包

[root@server0 ~]# rpm -q nfs-utils

nfs-utils-1.3.0-0.el7.x86\_64

[root@server0 ~]#

2.修改配置文件

[root@server0 ~]# mkdir /nsd01 #创建共享目录

[root@server0 ~]# echo haha > /nsd01/a.txt #建立测试文件

[root@server0 ~]# ls /nsd01

[root@server0 ~]# vim /etc/exports

文件夹路径 客户机地址(权限) 客户机地址(权限) .. ..

/nsd01 \*(ro) #将/nsd01目录共享给所有客户端,可以只读方式访问

[root@server0 ~]# systemctl restart nfs-server #重启服务

[root@server0 ~]# systemctl enable nfs-server #设置开机自启动

虚拟机desktop

[root@desktop0 ~]# mkdir /mnt/test

[root@desktop0 ~]# ls /mnt/test

[root@desktop0 ~]# showmount -e 172.25.0.11 #显示服务端共享路径

[root@desktop0 ~]# mount 172.25.0.11:/nsd01 /mnt/test/

[root@desktop0 ~]# df -h

开机自动挂载

\_netdev:声明网络设备,Linux先部署网络所有参数,在进行挂载该设备

[root@desktop0 ~]# vim /etc/fstab

172.25.0.11:/nsd01 /mnt/test nfs defaults,\_netdev 0 0

[root@desktop0 ~]# umount /mnt/test/

[root@desktop0 ~]# df -h

[root@desktop0 ~]# mount -a

[root@desktop0 ~]# df -h

#########################################################

HTTP服务基础

Web通信基本概念

• 基于 B/S (Browser/Server)架构的网页服务

– 服务端提供网页

– 浏览器下载并显示网页

• Hyper Text Transfer Protocol(http),超文本传输协议

• Hyper Text Markup Language(html),超文本标记语言

提供Web服务的软件

httpd(Apache)、nginx、tomcat

• 软件包:httpd

• 系统服务:httpd

• 提供的默认配置

– Listen:监听地址:端口(80)

– ServerName:本站点注册的DNS名称(空缺)

– DocumentRoot:网页根目录(/var/www/html)

虚拟机server:构建基本的Web服务

1.安装httpd

2.重启httpd服务

]# systemctl restart httpd

3.设置httpd服务开机自启

4.书写一个网站内容

]# echo '<h1>NSD1904 Web1' > /var/www/html/index.html

]# cat /var/www/html/index.html

]# firefox 172.25.0.11

########################################################

DocumentRoot:设置存放网页文件的路径(设置网页文件根目录)

虚拟机server:

]# mkdir /var/www/myweb

]# echo '<h1>wo shi myweb' > /var/www/myweb/index.html

]# vim /etc/httpd/conf/httpd.conf #全文命令模式下 搜索 /Doc

DocumentRoot "/var/www/myweb"

]# systemctl restart httpd

]# firefox 172.25.0.11

######################################################

实际路径与网络路径(浏览器中输入的路径)

客户端:firefox 172.25.0.11:80 ----->服务端 httpd---->主配置 DocumenRoot /var/www/myweb -----> /var/www/myweb

客户端:firefox 172.25.0.11

服务端:/var/www/myweb

客户端:firefox 172.25.0.11/abc

服务端:/var/www/myweb/abc

客户端:firefox 172.25.0.11/nsd

服务端:/var/www/myweb/nsd

DocumenRoot /var/www/myweb

客户端:firefox 172.25.0.11/var/www/myweb

服务端:/var/www/myweb/var/www/myweb

虚拟机server

]# mkdir /var/www/myweb/abc

]# echo '<h1>wo shi abc' > /var/www/myweb/abc/index.html

]# firefox 172.25.0.11/abc

#########################################################

Web访问控制,默认继承父目录访问控制

访问控制的配置字段:

<Directory "/var/www">

Require all granted #允许所有人访问

</Directory>

<Directory "/var/www">

Require ip 172.25.0.10 #仅允许172.25.0.10访问

</Directory>

<Directory "/var/www">

Require all denied #拒绝所有人访问

</Directory>

#######################################################

<Directory "/var/www">

Require all denied #拒绝所有人访问

</Directory>

<Directory "/var/www/test">

Require all granted #允许所有人访问

</Directory>

/var/www/路径下的页面,拒绝所有人访问

/var/www/test路径下的页面,允许所有人访问

/var/www/abc路径下的页面,拒绝所有人访问

/var/www/abc/nsd路径下的页面,拒绝所有人访问

#######################################################

案例:

修改网页文件,存放路径为/webroot

1.创建路径写入网页文件

]# mkdir /webroot

]# echo '<h1>wo shi webroot' > /webroot/index.html

]# ls /webroot/

2.修改配置文件

[root@server0 ~]# vim /etc/httpd/conf/httpd.conf

DocumentRoot "/webroot"

<Directory "/webroot"> #针对/webroot进行访问控制配置

Require all granted #允许所有人访问

</Directory>

[root@server0 ~]# setenforce 0 #将SELinux设置宽松模式

[root@server0 ~]# getenforce #查看SELinux状态

[root@server0 ~]# systemctl restart httpd

[root@server0 ~]# firefox 172.25.0.11

#########################################################

虚拟Web主机

– 由同一台服务器提供多个不同的Web页面

• 区分方式

– 基于域名的虚拟主机

– 基于端口的虚拟主机

– 基于IP地址的虚拟主机

域名可以DNS解析 :

DNS服务器虚拟机classroom

server0.example.com

www0.example.com

webapp0.example.com

• 为每个虚拟站点添加配置

<VirtualHost IP地址:端口>

ServerName 此站点的DNS名称

DocumentRoot 此站点的网页根目录

</VirtualHost>

• 配置文件路径

– /etc/httpd/conf/httpd.conf 主配置文件

– /etc/httpd/conf.d/\*.conf 调用配置文件

虚拟机server

[root@server0 ~]# vim /etc/httpd/conf.d/nsd01.conf #调用配置文件

<VirtualHost \*:80> #启用虚拟Web主机功能

ServerName server0.example.com #设置访问的域名

DocumentRoot /var/www/qq #设置存放网页文件目录

</VirtualHost>

<VirtualHost \*:80> #启用虚拟Web主机功能

ServerName www0.example.com #设置访问的域名

DocumentRoot /var/www/baidu #设置存放网页文件目录

</VirtualHost>

[root@server0 ~]# mkdir /var/www/qq /var/www/baidu

[root@server0 ~]# echo '<h1>企鹅' > /var/www/qq/index.html

[root@server0 ~]# echo '<h1>百度' > /var/www/baidu/index.html

[root@server0 ~]# systemctl restart httpd

[root@desktop0 ~]# firefox server0.example.com

[root@desktop0 ~]# firefox www0.example.com

#########################################################

一旦使用了虚拟Web主机功能,所有的网站都必须使用虚拟Web主机实现

如果直接使用IP地址进行访问,默认会将第一个虚拟Web主机页面内容呈现

[root@server0 ~]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName server0.example.com

DocumentRoot /var/www/qq

</VirtualHost>

<VirtualHost \*:80>

ServerName www0.example.com

DocumentRoot /var/www/baidu

</VirtualHost>

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/myweb

</VirtualHost>

[root@server0 ~]# systemctl restart httpd

###############################################

## Day05

一 环境设置

虚拟机server

[root@server0 ~]# firewall-cmd --set-default-zone=trusted

[root@server0 ~]# firewall-cmd --get-default-zone

虚拟机desktop

[root@desktop0 ~]# firewall-cmd --set-default-zone=trusted

[root@desktop0 ~]# firewall-cmd --get-default-zone

二 虚拟机server构建基于域名的虚拟Web主机

- 实现三个网站的部署

- 实现客户端访问server0.example.com网页内容为 卖女孩的小火柴

- 实现客户端访问www0.example.com网页内容为 奔跑吧骆驼

- 实现客户端访问webapp0.example.com网页内容为 20里春风不如你

[root@server0 ~]# yum -y install httpd

[root@server0 ~]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName server0.example.com

DocumentRoot /var/www/nsd01

</VirtualHost>

<VirtualHost \*:80>

ServerName www0.example.com

DocumentRoot /var/www/nsd02

</VirtualHost>

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/nsd03

</VirtualHost>

]# cd /var/www/

]# mkdir nsd01 nsd02 nsd03

]# echo '<h1>卖女孩的小火柴' > /var/www/nsd01/index.html

]# echo '<h1>奔跑吧骆驼' > /var/www/nsd02/index.html

]# echo '<h1>20春风不如你' > /var/www/nsd03/index.html

]# systemctl restart httpd

###########################################################

文件夹权限

• 针对 DocumentRoot 网页目录的权限控制

• 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设置

案例1:虚拟机server配置网页内容访问

在 Web 网站 http://server0.example.com 的

DocumentRoot 目录下创建一个名为 private 的子目录,要求如下:

1.查看server0.example.com的DocumentRoot目录路径

]# cat /etc/httpd/conf.d/nsd01.conf

]# mkdir /var/www/nsd01/private

]# echo '<h1>wo shi private' > /var/www/nsd01/private/index.html

]# firefox server0.example.com/private

2.从虚拟机server0上,任何人都可以浏览 private 的内容

但是从其他系统不能访问这个目录的内容

[root@server0 ~]# vim /etc/httpd/conf.d/nsd02.conf

<Directory "/var/www/nsd01/private">

Require ip 172.25.0.11 #仅允许172.25.0.11进行访问

</Directory>

[root@server0 ~]# systemctl restart httpd

虚拟机desktop0:访问测试权限不足

]# firefox server0.example.com/private

Forbidden

You don't have permission to access /private/ on this server.

##########################################################

案例2:使用自定Web根目录

调整 Web 站点 http://server0.example.com 的网页

目录,要求如下:

1)新建目录 /webroot,作为此站点新的网页目录

]# mkdir /webroot

]# echo '<h1>wo shi Webroot' > /webroot/index.html

]# ls /webroot/

[root@server0 ~]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName server0.example.com

DocumentRoot /webroot

</VirtualHost>

........

[root@server0 ~]# vim /etc/httpd/conf.d/nsd02.conf

........

<Directory "/webroot">

Require all granted

</Directory>

[root@server0 ~]# systemctl restart httpd

SELinux三大策略: 1.布尔值 2.安全上下文 3.非默认端口的开放

[root@server0 ~]# setenforce 1 #开启SELinux

[root@server0 ~]# getenforce #查看SELinux状态

SELinux安全上下文(标签值)

[root@server0 ~]# ls -Zd /var/www/ #查看目录的上下为值

[root@server0 ~]# ls -Zd /webroot/

• 方式1:参照标准目录,重设新目录的属性

– chcon [-R] --reference=模板目录 新目录

[root@server0 ~]# chcon -R --reference=/var/www /webroot/

[root@server0 ~]# ls -Zd /webroot/

########################################################

部署动态网站

静态网站的运行

• 服务端的原始网页 = 浏览器访问到的网页

– 由Web服务软件处理所有请求

– 文本(txt/html)、图片(jpg/png)等静态资源

动态网站的运行

• 服务端的原始网页 ≠ 浏览器访问到的网页

– 由Web服务软件接受请求,动态程序转后端模块处理

– PHP网页、Python网页、JSP网页......

虚拟机server新建一个动态站点站点名称为 webapp0.example.com

1.部署Python页面

[root@server0 ~]# cat /etc/httpd/conf.d/nsd01.conf

[root@server0 ~]# cd /var/www/nsd03

[root@server0 nsd03]# wget http://classroom.example.com/pub/materials/webinfo.wsgi

[root@server0 nsd03]# cat webinfo.wsgi

2.方便用户的访问,页面跳转(页面别名 地址重写)

浏览器中输入: firefox webapp0.example.com

服务端:/var/www/nsd03/webinfo.wsgi

Alias 网络路径 服务端实际路径

[root@server0 /]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/nsd03

Alias / /var/www/nsd03/webinfo.wsgi

#当客户端直接访问网页文件根目录时,呈现/var/www/nsd03/webinfo.wsgi页面内容

</VirtualHost>

[root@server0 /]# systemctl restart httpd

[root@server0 /]# firefox webapp0.example.com

3.安装mod\_wsgi软件包,专用于翻译Python页面

[root@server0 /]# yum -y install mod\_wsgi

[root@server0 /]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/nsd03

WsgiScriptAlias / /var/www/nsd03/webinfo.wsgi

</VirtualHost>

[root@server0 /]# systemctl restart httpd

[root@desktop0 ~]# firefox webapp0.example.com

显示内容为 UNIX时间戳:从1970-1-1 0:0:0到达当前系统时间,所经历的秒数

4.此虚拟webapp0.example.com主机侦听在端口8909

[root@server0 /]# vim /etc/httpd/conf.d/nsd01.conf

Listen 8909 #让httpd程序监听8909端口

<VirtualHost \*:8909> #本虚拟Web主机使用8909端口

ServerName webapp0.example.com

DocumentRoot /var/www/nsd03

WsgiScriptAlias / /var/www/nsd03/webinfo.wsgi

</VirtualHost>

5.SELinux非默认端口的开放

-l:列出 -a:添加 -t:类型 -p:协议

]# semanage port -l | grep http

]# semanage port -a -t http\_port\_t -p tcp 8909 #添加端口

]# systemctl restart httpd

]# firefox webapp0.example.com:8909

#########################################################

1.端口的优先级最高

2.默认将第一个虚拟Web主机内容呈现

#########################################################

安全Web服务,实现https(完全的超文本传输协议)

PKI公钥基础设施

• Public Key Infrastructure,公钥基础设施

– 公钥:主要用来加密数据

– 私钥:主要用来解密数据(与相应的公钥匹配)

– 数字证书:证明拥有者的合法性/权威性(单位名称、

有效期、公钥、颁发机构及签名、......)

– Certificate Authority,数字证书授权中心:负责证书

的申请/审核/颁发/鉴定/撤销等管理工作

虚拟机server:构建安全的Web服务

1.部署网站证书

]# cd /etc/pki/tls/certs/

]# wget http://classroom.example.com/pub/tls/certs/server0.crt

]# ls

2.部署根证书

]# cd /etc/pki/tls/certs/

]# wget http://classroom.example.com/pub/example-ca.crt

]# ls

3.部署私钥(解密数据)

]# cd /etc/pki/tls/private/

]# wget http://classroom.example.com/pub/tls/private/server0.key

]# ls

4.安装支持安全的软件

[root@server0 /]# yum -y install mod\_ssl

[root@server0 /]# ls /etc/httpd/conf.d/ssl.conf

[root@server0 /]# vim /etc/httpd/conf.d/ssl.conf

vim末行模式 :set nu #开启行号功能

59 DocumentRoot "/var/www/html"

60 ServerName www0.example.com:443

#指定网站证书的路径及名称

100 SSLCertificateFile /etc/pki/tls/certs/server0.crt

#指定解密的私钥

107 SSLCertificateKeyFile /etc/pki/tls/private/server0.key

#指定根证书

122 SSLCACertificateFile /etc/pki/tls/certs/example-ca.crt

5.书写一个网站

]# echo '<h1>haha xixi' > /var/www/html/index.html

6.重启httpd服务

]# systemctl restart httpd

7.验证

[root@desktop0 ~]# firefox https://www0.example.com

我已了解可能的风险--->添加例外---->确认例外

## Day06

一 环境设置

虚拟机server

[root@server0 ~]# firewall-cmd --set-default-zone=trusted

[root@server0 ~]# firewall-cmd --get-default-zone

虚拟机desktop

[root@desktop0 ~]# firewall-cmd --set-default-zone=trusted

[root@desktop0 ~]# firewall-cmd --get-default-zone

#########################################################

二 网络参数配置

方式一:nmtui

方式二:

1.设置永久设置主机名,修改/etc/hostname配置文件

[root@server0 ~]# echo A.tedu.cn > /etc/hostname

[root@server0 ~]# hostname

A.tedu.cn

[root@server0 ~]# exit

登出

Connection to 172.25.0.11 closed.

[student@room9pc01 ~]$ gos

Last login: Fri May 17 09:04:25 2019 from 172.25.0.250

[root@A ~]#

2.配置IP地址,子网掩码,网关地址

1)查看识别的网卡名称

[root@A ~]# nmcli connection show

名称

System eth0

2)修改IP地址,子网掩码,网关地址

]# nmcli connection modify 'System eth0'

ipv4.method manual

ipv4.addresses '172.25.0.112/24 172.25.0.254'

connection.autoconnect yes

]# nmcli connection 修改 '识别的网卡名'

ipv4.方法 手工配置IP地址

ipv4.地址 'IP地址/子网掩码 网关地址'

每次开机自动启用

3)激活配置

[root@A ~]# ifconfig | head -2

[root@A ~]# nmcli connection up 'System eth0'

######################################################

DNS服务器地址:将域名解析为对应的IP地址

/etc/resolv.conf #指定DNS服务器最终有效配置文件

]# echo 'nameserver 172.25.254.254' > /etc/resolv.conf

]# cat /etc/resolv.conf

nameserver 172.25.254.254

]# nslookup www0.example.com #测试域名解析

Server: 172.25.254.254

Address: 172.25.254.254#53

www0.example.com canonical name = server0.example.com.

Name: server0.example.com

Address: 172.25.0.11

#######################################################

配置IPv6地址

• IPv4 地址表示

– 32个二进制位,点分隔的十进制数

– 例如:172.25.0.11、127.0.0.1

• IPv6 地址表示

– 128个二进制位,冒号分隔的十六进制数

– 每段内连续的前置 0 可省略、连续的多个 : 可简化为 ::

– 例如: 2003:ac18:0000:0000:0000:0000:0000:0305

2003:ac18::305/64

1)查看识别的网卡名称

[root@A ~]# nmcli connection show

2)修改IP地址,子网掩码,网关地址

]# nmcli connection modify 'System eth0'

ipv6.method manual

ipv6.addresses 2003:ac18::305/64

connection.autoconnect yes

]# nmcli connection 修改 '识别的网卡名'

ipv6.方法 手工配置IP地址

ipv6.地址 IP地址/子网掩码

每次开机自动启用

3)激活配置

[root@A ~]# ifconfig | head -4

[root@A ~]# nmcli connection up 'System eth0'

[root@A ~]# ifconfig | head -4

[root@A ~]# ping6 2003:ac18::305

#########################################################

链路聚合(聚合链路 网卡绑定)

作用:提高网卡设备的可靠性,提供冗余

eth1 eth2

虚拟网卡:team0

192.168.1.1

1.建立虚拟网卡team0,参考 man teamd.conf 全文搜索 /example

]# nmcli connection add type team

con-name team0 ifname team0 autoconnect yes

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

]# nmcli connection 添加 类型 team

配置文件名 team0 网卡名 team0 每次开机自动启用

链路聚合管理方式为 热备份方式

]# ifconfig #查看生成的虚拟网卡team0

生成配置文件位置:/etc/sysconfig/network-scripts/

2.添加成员

]# 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

]# nmcli connection 添加 类型 team-成员

配置文件名 team0-1 网卡名 eth1 主设备为 team0

生成配置文件位置:/etc/sysconfig/network-scripts/

3.配置IP地址激活

[root@A ~]# nmcli connection modify team0 ipv4.method manual ipv4.addresses 192.168.1.1/24 connection.autoconnect yes

[root@A ~]# nmcli connection up team0 #激活配置

[root@A ~]# teamdctl team0 state #专用于查看team0信息

[root@A ~]# ifconfig eth2 down #禁用eth2网卡

[root@A ~]# teamdctl team0 state

如果无法激活,删除所有配置重新来过

[root@A ~]# nmcli connection delete team0

[root@A ~]# nmcli connection delete team0-1

[root@A ~]# nmcli connection delete team0-2

###################################################

parted分区工具,进行gpt分区模式

128个主分区 最大空间支持18EB

[root@A ~]# parted /dev/vdb

(parted) mktable gpt #指定分区表类型为gpt

(parted) print #输出分区表信息

(parted) mkpart #划分新的分区

分区名称？ []? haha #分区名称随意写

文件系统类型？ [ext2]? ext4 #文件系统随意些,不起实际作用

起始点？ 0 #分区起始点

结束点？ 2G #分区结束点

警告: The resulting partition is not properly aligned

for best performance.

忽略/Ignore/放弃/Cancel? Ignore #选择ignore忽略

(parted) print

(parted) unit GB #使用GB最为单位

(parted) print

(parted) mkpart

分区名称？ []? haha

文件系统类型？ [ext2]? ext4

起始点？ 2G

结束点？ 5G

(parted) print

(parted) quit

###################################################

基础邮件服务

邮件域名:

server0.example.com

邮箱帐号:

yg@server0.example.com

• 电子邮件服务器的基本功能

– 为用户提供电子邮箱存储空间(用户名@邮件域名)

– 处理用户发出的邮件 —— 传递给收件服务器

– 处理用户收到的邮件 —— 投递到邮箱

快速部署postfix邮件服务器

1.装postfix软件包

[root@A ~]# rpm -q postfix

postfix-2.10.1-6.el7.x86\_64

2.修改配置文件/etc/postfix/main.cf

vim末行模式 :set nu #开启行号

99 myorigin = server0.example.com #默认补全域名后缀

116 inet\_interfaces = all #允许本机所有网卡提供邮件服务

164 mydestination = server0.example.com #此值判断为本域邮件

3.重启服务

[root@A ~]# systemctl restart postfix

测试邮件收发

[root@A ~]# useradd yg

[root@A ~]# useradd xln

•发信操作: mail -s '邮件标题' -r 发件人 收件人

[root@A ~]# mail -s 'test01' -r yg xln

ahahxixihehelele

.

EOT

[root@A ~]# echo AAA | mail -s 'test02' -r yg xln

•收信操作: mail [-u 用户名]

[root@A ~]# mail -u xln

>N 1 yg@server0.example.c Fri May 17 15:28 18/565

& 1 #输入邮件编号1回车查看邮件的内容

####################################################

使用LDAP认证,实现网络用户认证,达到用户的集中管理

　　RHCSA　红帽认证管理员

　　RHCE　红帽认证工程师

　　RHCA　红帽认证架构师

网络用户:用户信息会放在网络中LDAP服务器

本地用户:用户信息会放在/etc/passwd

LDAP服务器:虚拟机classroom

客户端:虚拟机server

1.安装sssd软件,与LDAP服务器沟通

[root@server0 /]# yum -y install sssd

[root@server0 /]# rpm -q sssd

2.安装authconfig-gtk图形的工具,进行配置客户端sssd软件

[root@server0 /]# yum -y install authconfig-gtk

[root@server0 /]# authconfig-gtk

完全合格的主机名 www.baidu.com　　　www.qq.com

classroom.example.com

选择LDAP

dc=example,dc=com #指定服务端域名

classroom.example.com #指定服务端主机名

勾选TLS加密

使用证书加密:

http://classroom.example.com/pub/example-ca.crt

选择LDAP密码

3.重起服务

[root@server0 ~]# systemctl restart sssd #重起服务

[root@server0 ~]# systemctl enable sssd #设置开机自启动

[root@server0 ~]# grep ldapuser0 /etc/passwd

[root@server0 ~]# id ldapuser0 #验证LDAP用户信息

###################################################

# NSD Service全部笔记

## Day01

搭建新教学环境

一 关闭虚拟机classroom与server与desktop

二 真机建立全新的两台虚拟机

[student@room9pc01 ~]$ clone-vm7

Enter VM number: 4

[student@room9pc01 ~]$ clone-vm7

Enter VM number: 5

vi是Linux最基本的编辑器

虚拟机root的密码:123456

##################################################

三 配置虚拟机A

1.为虚拟机A:配置eth0的IP地址为192.168.4.7/24

真机管理本机虚拟机,前提关闭所有虚拟机图形窗口

[student@room9pc01 ~]$ virsh console A

Ctrl + ]:退出管理

[root@localhost ~]# nmcli connection modify eth0 ipv4.method manual ipv4.address 192.168.4.7/24 connection.autoconnect yes

[root@localhost ~]# nmcli connection up eth0

[root@localhost ~]# ifconfig | head -2

2.为虚拟机A配置主机名:svr7.tedu.cn

[root@localhost ~]# echo svr7.tedu.cn > /etc/hostname

svr7.tedu.cn

[root@localhost ~]# hostname svr7.tedu.cn

[root@localhost ~]# hostname

svr7.tedu.cn

[root@svr7 ~]# cat /etc/motd #每次开机自动显示内容

###################################################

真机利用virsh console管理虚拟机B进行配置

[root@localhost ~]# hostname pc207.tedu.cn

[root@localhost ~]# echo pc207.tedu.cn > /etc/hostname

[root@localhost ~]# nmcli connection modify eth0 ipv4.method manual ipv4.address 192.168.4.207/24 connection.autoconnect yes

[root@localhost ~]# nmcli connection up eth0

[root@localhost ~]# ifconfig | head -2

###################################################

利用真机进行ssh远程管理

真机上设置

[student@room9pc01 ~]$ vim /home/student/.bashrc

alias goa='ssh -X root@192.168.4.7'

alias gob='ssh -X root@192.168.4.207'

开启新的终端才能验证

###################################################

为虚拟机A与虚拟机B构建Yum仓库

一 真机:服务端

构建Web服务或FTP服务,共享光盘所有内容

1.构建FTP服务,共享光盘所有内容

[student@room9pc01 ~]$ rpm -q vsftpd

vsftpd-3.0.2-22.el7.x86\_64

[student@room9pc01 ~]$ ls /var/ftp/centos-1804/

[student@room9pc01 ~]$ systemctl status vsftpd

[student@room9pc01 ~]$ vim /etc/fstab

..........

/var/lib/libvirt/images/iso/CentOS7-1804.iso /var/ftp/centos-1804 iso9660 defaults 0 0

]$ firefox ftp://192.168.4.254/centos-1804

二 客户端:虚拟机A与虚拟机B

[root@svr7 ~]# vim /etc/yum.repos.d/local.repo

[local\_repo]

name=CentOS-$releasever - Base

baseurl=ftp://192.168.4.254/centos-1804

enabled=1

gpgcheck=0

[root@svr7 ~]# yum repolist

[root@svr7 ~]# yum -y install xeyes

[root@svr7 ~]# xeyes

###################################################

总结新建虚拟机:

1.真机利用clone-vm7生成虚拟机,开启虚拟机

2.真机利用virsh console 虚拟机名 设置IP地址

3.真机利用ssh远程管理虚拟机,配置主机名与Yum客户端配置文件

####################################################

虚拟机C

1.配置eth0的IP地址:192.168.4.10/24

2.配置主机名:svr10.tedu.cn

3.配置Yum仓库,以真机提供FTP作为服务端

虚拟机D

1.配置eth0的IP地址:192.168.4.20/24

2.配置主机名:pc20.tedu.cn

3.配置Yum仓库,以真机提供FTP作为服务端

####################################################

目录结构

• 认识Linux的目录层次:

– man hier

• 常见一级目录的用途

/boot 存放系统引导必需的文件,包括内核、启动配置

/bin、/sbin 存放各种命令程序

/dev 存放硬盘、键盘、鼠标、光驱等各种设备文件

/etc 存放Linux系统及各种程序的配置文件

/root、/home/用户名 分别是管理员root、普通用户的默认家目录

/var 存放日志文件、邮箱目录等经常变化的文件

/proc 存放内存中的映射数据,不占用磁盘

/tmp 存放系统运行过程中使用的一些临时文件

权限的数值表示

• 权限的数值化

– 基本权限:r = 4,w = 2,x = 1

– 附加权限:SUID = 4,SGID = 2,Sticky Bit = 1

[root@svr7 ~]# mkdir /nsd01

[root@svr7 ~]# ls -ld /nsd01

[root@svr7 ~]# chmod 700 /nsd01

[root@svr7 ~]# ls -ld /nsd01

[root@svr7 ~]# chmod 007 /nsd01

[root@svr7 ~]# ls -ld /nsd01

[root@svr7 ~]# chmod 755 /nsd01

[root@svr7 ~]# ls -ld /nsd01

[root@svr7 ~]# chmod 750 /nsd01

[root@svr7 ~]# ls -ld /nsd01

[root@svr7 ~]# chmod 3755 /nsd01

[root@svr7 ~]# ls -ld /nsd01

###################################################

历史命令

• 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

[root@svr7 ~]# vim /etc/profile

HISTSIZE=1000 #默认记录1000条

[root@svr7 ~]# history #查看历史命令

[root@svr7 ~]# history -c #清空历史命令

[root@svr7 ~]# history

1 history

[root@svr7 ~]# cat /etc/redhat-release

[root@svr7 ~]# ls -l /etc/redhat-release

[root@svr7 ~]# cat /etc/hostname

[root@svr7 ~]# history

[root@svr7 ~]# !cat #执行历史命令中最近一条cat开头

###################################################

实用小命令工具

• 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的总空间大小

– -h:提供易读容量单位(K、M等)

[root@svr7 ~]# du -s /root/

[root@svr7 ~]# du -sh /root/

[root@svr7 ~]# du -sh /boot/ /etc/

[root@svr7 ~]# du -sh /

################################################

bc 交互式计算器

[root@svr7 ~]# bc

-bash: bc: 未找到命令

[root@svr7 ~]# yum -y install bc

[root@svr7 ~]# bc #进入交互式模式

+ - \* / %(取余数运算)

公式是: 被除数 ÷ 除数 = 商

余数一定小于除数

##################################################

• date,查看/调整系统日期时间

– date +%F、date +%R

– date +"%Y-%m-%d %H:%M:%S"

– date -s "yyyy-mm-dd HH:MM:SS"

[root@svr7 ~]# date +%F #显示年月日

[root@svr7 ~]# date +%R #显示时分

[root@svr7 ~]# date +%Y #显示年

[root@svr7 ~]# date +%m #显示月

[root@svr7 ~]# date +%d #显示日期

[root@svr7 ~]# date +%H #显示时

[root@svr7 ~]# date +%M #显示分

[root@svr7 ~]# date +%S #显示秒

#####################################################

制作快捷方式(连接文件 符号文件)

格式:ln -s /路径/源文件 /路径/生成快捷方式名称 #软连接

[root@svr7 ~]# cat /etc/hostname

[root@svr7 ~]# ln -s /etc/hostname /

[root@svr7 ~]# ls /

[root@svr7 ~]# ls -l /hostname

[root@svr7 ~]# ln -s /etc/hostname /hn

[root@svr7 ~]# ls /

[root@svr7 ~]# ls -l /hn

[root@svr7 ~]# cat /hn

[root@svr7 ~]# cat /hostname

若原始文件或目录被删除,连接文件将失效

软连接可存放在不同分区/文件系统,也可以针对目录

[root@svr7 ~]# echo 123 > /opt/A

[root@svr7 ~]# cat /opt/A

123

[root@svr7 ~]# ln -s /opt/A /opt/B

[root@svr7 ~]# ls /opt/

A B

[root@svr7 ~]# rm -rf /opt/A

[root@svr7 ~]# ls /opt/

B

[root@svr7 ~]# cat /opt/B

ln,创建硬连接

– ln 原始文件 硬连接文件

若原始文件被删除,连接文件仍可用

硬连接与原始文件必须在同一分区/文件系统,必须时文件

[root@svr7 ~]# echo haha > /opt/1.txt

[root@svr7 ~]# ln /opt/1.txt /opt/2.txt

[root@svr7 ~]# ls /opt/

[root@svr7 ~]# cat /opt/2.txt

[root@svr7 ~]# rm -rf /opt/1.txt

[root@svr7 ~]# ls /opt/

[root@svr7 ~]# cat /opt/2.txt

####################################################

获取命令帮助

[root@svr7 ~]# date --help

[root@svr7 ~]# man date

[root@svr7 ~]# man passwd

[root@svr7 ~]# man 5 passwd #配置文件帮助信息

###################################################

zip归档工具(跨平台的压缩工具)

• 归档+压缩操作

– zip [-r] 备份文件.zip 被归档的文档...

[root@svr7 ~]# yum -y install zip

[root@svr7 ~]# zip -r /opt/file.zip /etc/passwd /home

[root@svr7 ~]# ls /opt/

[root@svr7 ~]# zip -r /opt/abc.zip /etc/fstab /etc/hosts

[root@svr7 ~]# ls /opt/

• 释放归档+解压操作

– unzip 备份文件.zip [-d 目标文件夹]

[root@svr7 ~]# yum -y install unzip

[root@svr7 ~]# mkdir /nsd02

[root@svr7 ~]# unzip /opt/file.zip -d /nsd02

[root@svr7 ~]# ls /nsd02

[root@svr7 ~]# ls /nsd02/etc/

[root@svr7 ~]# ls /nsd02/home/

####################################################

发布自定义Yum仓库

把从互联网下载的软件包,采用Yum的机制进行管理

1.具备从互联网下载的软件包

真机上:

[student@room9pc01 ~]$ ls /linux-soft/01

Cobbler.zip tools.tar.gz

2.将真机tools.tar.gz上传到虚拟机A的/root目录下

scp=ssh+cp结合

scp /本地路径/源文件 root@对方IP地址:/目标路径/

scp /linux-soft/01/tools.tar.gz root@192.168.4.7:/root

3.验证在虚拟机A进行查看

[root@svr7 ~]# ls /root/

Desktop tools.tar.gz

[root@svr7 ~]#

4.虚拟机A解包:

[root@svr7 ~]# tar -xf /root/tools.tar.gz -C /

[root@svr7 ~]# ls /

[root@svr7 ~]# ls /tools/

[root@svr7 ~]# ls /tools/other/

5.生成仓库清单(仓库数据文件)

[root@svr7 ~]# yum -y install createrepo

[root@svr7 ~]# createrepo /tools/other/

[root@svr7 ~]# ls /tools/other/

6.修改配置文件指定新的Yum服务端

[root@svr7 ~]# ls /etc/yum.repos.d/local.repo

[local\_repo]

name=CentOS-$releasever - Base

baseurl=ftp://192.168.4.254/centos-1804

enabled=1

gpgcheck=0

[myrpm]

name=myrpm

baseurl=file:///tools/other #指定本机为服务端

enabled=1

gpgcheck=0

[root@svr7 ~]# yum repolist

##################################################

[root@svr7 ~]# yum -y install cmatrix

[root@svr7 ~]# cmatrix

[root@svr7 ~]# yum -y install sl

[root@svr7 ~]# sl

[root@svr7 ~]# yum -y install oneko

[root@svr7 ~]# oneko &

关闭终端,可以结束oneko程序

##################################################

vim编辑技巧

命令模式

[root@svr7 ~]# cp /etc/passwd /opt/pa.txt

[root@svr7 ~]# vim /opt/pa.txt

光标跳转

Home 键 或 ^、数字 0 跳转到行首

End 键 或“$”键 跳转到行尾

1G 或 gg 跳转到文件的首行

G 跳转到文件的末尾行

复制/粘贴/删除

复制 yy、#yy 复制光标处的一行、#行

粘贴 p、P 粘贴到光标处之后、之前

x 或 Delete键 删除光标处的单个字符

dd、#dd 删除光标处的一行、#行

d^ 从光标处之前删除至行首

d$或D(大写) 从光标处删除到行尾

C(大写) 从光标处删除到行尾,并且进入插入模式

u 撤销

查找/撤销/保存

/word 向后查找字符串“word”

n、N 跳至后/前一个结果

u 撤销最近的一次操作

U 撤销对当前行的所有修改

Ctrl + r 取消前一次撤销操作

ZZ(大写) 保存修改并退出

[root@svr7 ~]# cp /etc/passwd /opt/pa.txt

cp：是否覆盖"/opt/pa.txt"？ y

[root@svr7 ~]# vim /opt/pa.txt

####################################################

末行模式操作

读入其他文件内容

[root@svr7 ~]# echo 123 > /opt/a.txt

[root@svr7 ~]# echo abc > /opt/b.txt

[root@svr7 ~]# vim /opt/a.txt

末行模式下 :r /opt/b.txt #读入/opt/b.txt内容到当前文件

末行模式下 :r /etc/hostname

末行模式下 :r /etc/hosts

字符串替换

:s/root/admin 替换光标所在的当前行第一个“root”

:s/root/admin/g 替换光标所在的当前行所有的“root”

:5,10 s/root/admin/g 替换第5-10行所有的“root”

:% s/root/admin/g 替换文件内所有的“root”

[root@svr7 ~]# cp /etc/passwd /opt/p.txt

[root@svr7 ~]# vim /opt/p.txt

开关参数的控制

:set nu或nonu 显示/不显示行号

:set ai或noai 启用/关闭自动缩进

###############################################

源码编译安装

RPM包: rpm -ivh yum

源码包----gcc与make---->可以执行的文件------->运行安装

源码编译安装的优势

• 主要优点

– 获得软件的最新版,及时修复bug

– 软件功能可按需选择/定制,有更多软件可供选择

– 源码包适用各种平台

步骤1:安装依赖关系包(安装开发工具)

[root@svr7 ~]# yum -y install gcc make

[root@svr7 ~]# rpm -q gcc

gcc-4.8.5-28.el7.x86\_64

[root@svr7 ~]# rpm -q make

make-3.82-23.el7.x86\_64

步骤2:tar解包,释放源代码至指定目录

]# tar -xf /tools/inotify-tools-3.13.tar.gz -C /

]# ls /

]# cd /inotify-tools-3.13/

]# ls

步骤3: ./configure 配置,指定安装目录/功能模块等选项

./=当前路径下运行 作用1:检测本机是否安装gcc

作用2:指定安装目录/功能模块等选项

--prefix=指定安装位置

]# cd /inotify-tools-3.13/

]# ./configure --prefix=/mnt/myrpm

常见报错提示:没有安装gcc

checking for gcc... no

checking for cc... no

checking for cl.exe... no

configure: error: no acceptable C compiler found in $PATH

See `config.log' for more details.

步骤4:make 编译,生成可执行的二进制程序文件

]# cd /inotify-tools-3.13/

]# make

步骤5:make install 安装,将编译好的文件复制到安装目录

]# cd /inotify-tools-3.13/

]# make install

]# ls /mnt/

]# ls /mnt/myrpm/ #查看安装生成的目录

]# ls /mnt/myrpm/bin/

####################################################

一 真机远程管理虚拟机A

虚拟化概述

• virtualization 资源管理

– x个物理资源 --> y个逻辑资源

– 实现程度:完全、部分、硬件辅助(CPU)

CPU能够直接识别运算虚拟机指令

虚拟机A:

[root@svr7 ~]# lscpu | grep vmx #查看CPU是否支持虚拟化

真机查看CPU是否支持虚拟化

[student@room9pc01 ~]$ lscpu | grep vmx

####################################################

• 虚拟化主要厂商及产品

VMware VMware Workstation、vSphere

Microsoft VirtualPC、Hyper-V

RedHat KVM、RHEV

Citrix Xen

Oracle Oracle VM VirtualBox

####################################################

二 手动安装一台有图形的虚拟机nsd01

操作系统为:CentOS 7.5

内存为:2048M

硬盘大小:9G

cpu:1

虚拟名为:nsd01

网络类型:private1

软件包选择:带GUI的服务器

分区:自动分区

root密码设置为1 创建lisi用户密码设置为1

####################################################

三 针对虚拟机nsd01进行配置

1.关闭SELinux

[root@localhost ~]# getenforce

[root@localhost ~]# setenforce 0

[root@localhost ~]# vim /etc/selinux/config

SELINUX=permissive

2.防火墙设置为trusted

3.设置IP地址为:192.168.4.30/24

]# nmcli connection modify eth0

ipv4.method manual ipv4.addresses 192.168.4.30/24

connection.autoconnect yes

]# nmcli connection up eth0

]# ifconfig | head -2

4.主机名:kvm.tedu.cn

5.构建Yum仓库,使用真机FTP最为服务端

[root@localhost yum.repos.d]# mkdir repo

[root@localhost yum.repos.d]# mv \*.repo repo

###################################################

虚拟机nsd01:安装虚拟化服务器平台

• 主要软件包

– qemu-kvm:为 kvm 提供底层仿真支持

– libvirt-daemon:libvirtd 守护进程,管理虚拟机

– libvirt-client:用户端软件,提供客户端管理命令

– libvirt-daemon-driver-qemu:libvirtd 连接 qemu 的驱动

– virt-manager:图形管理工具

[root@kvm ~]# yum -y install qemu-kvm

[root@kvm ~]# yum -y install libvirt-daemon

[root@kvm ~]# yum -y install libvirt-daemon-driver-qemu

[root@kvm ~]# yum -y install libvirt-client

[root@kvm ~]# yum -y install virt-manager

[root@kvm ~]# virt-manager #图形虚拟系统管理器

####################################################

虚拟化服务:

[root@kvm ~]# systemctl status libvirtd

####################################################

真机上进行虚拟机管理:

virsh命令工具介绍

• 查看KVM节点(服务器)信息

– virsh nodeinfo

• 列出虚拟机

– virsh list [--all]

• 查看指定虚拟机的信息

– virsh dominfo 虚拟机名称

• 将指定的虚拟机设为开机自动运行

– virsh autostart [--disable] 虚拟机名称

• 强制关闭指定的虚拟机

– virsh destroy 虚拟机名称

• 运行|重启|关闭指定的虚拟机

– virsh start 虚拟机名称

###################################################

xml配置文件(虚拟机描述文件)

默认存放路径:

[student@room9pc01 ~]$ ls /etc/libvirt/qemu

输出虚拟机xml文件内容

[student@room9pc01 ~]$ virsh dumpxml nsd01 | less

虚拟机名称: <name>nsd01</name>

<uuid>cf4e78cb-65bb-48df-a84b-56741a470fcb</uuid>

<source file='/var/lib/libvirt/images/nsd01.qcow2'/>

网卡MAC地址:网卡唯一编号<mac address='52:54:00:20:37:74'/>

###################################################

一台KVM虚拟机的组成

– xml配置文件:定义虚拟机的名称、UUID、CPU、内

存、虚拟磁盘、网卡等各种参数设置

默认路径:/etc/libvirt/qemu

– 磁盘镜像文件:保存虚拟机的操作系统及文档数据,

镜像路径取决于xml配置文件中的定义

默认路径:/var/lib/libvirt/images/

真机:手动创建一个虚拟机abc01

1.创建虚拟机磁盘镜像文件

]$ cd /var/lib/libvirt/images/

]$ cp .node\_tedu.qcow2 abc01.qcow2

2.创建虚拟机xml配置文件

]$ virsh dumpxml nsd01 > /etc/libvirt/qemu/abc01.xml

]$ ls /etc/libvirt/qemu

]$ ls -l /etc/libvirt/qemu/abc01.xml

]$ vim /etc/libvirt/qemu/abc01.xml

虚拟机名称: <name>abc01</name>

<uuid>删除整行内容,如果删多了按u进行撤销

<source file='/var/lib/libvirt/images/abc01.qcow2'/>

<mac address='52:54:00:20:37:74'/> 删除整行内容

3.导入虚拟机信息

]$ virsh define /etc/libvirt/qemu/abc01.xml

]$ virsh list --all

]$ virsh start abc01

]$ virsh list

###################################################

手动创建虚拟机:

1.创建新的虚拟机磁盘文件

2.导出虚拟机xml配置文件

]$ virsh dumpxml nsd01 > /etc/libvirt/qemu/abc01.xml

]$ vim /etc/libvirt/qemu/abc01.xml

虚拟机名称: <name>abc01</name>

<uuid>删除整行内容,如果删多了按u进行撤销

<source file='/var/lib/libvirt/images/abc01.qcow2'/>

<mac address='52:54:00:20:37:74'/> 删除整行内容

3.导入虚拟机信息

]$ virsh define /etc/libvirt/qemu/abc01.xml

##################################################

三合一的命令:导出xml配置文件 修改 导入

1.创建一个虚拟机磁盘文件

]$ cd /var/lib/libvirt/images/

]$ cp .node\_tedu.qcow2 abc02.qcow2

2.三合一的命令:导出xml配置文件 修改 导入

[student@room9pc01 /]$ virsh edit nsd01

虚拟机名称: <name>abc02</name>

<uuid>删除整行内容,如果删多了按u进行撤销

<source file='/var/lib/libvirt/images/abc02.qcow2'/>

<mac address='52:54:00:20:37:74'/> 删除整行内容

[student@room9pc01 /]$ virsh list --all

[student@room9pc01 /]$ virsh start abc02

###################################################

虚拟机命令行的删除

[student@room9pc01 /]$ virsh list

[student@room9pc01 /]$ virsh destroy abc02 #强制关机

域 abc02 被删除

[student@room9pc01 /]$ virsh list --all

[student@room9pc01 /]$ virsh undefine abc02 #取消定义

域 abc02 已经被取消定义

]$ virsh list --all

]$ cd /var/lib/libvirt/images/

]$ ls abc02.qcow2

]$ rm -rf abc02.qcow2 #删除磁盘文件

]$ ls

##################################################

COW(Copy On Write)机制 写时复制

• 虚拟机的磁盘镜像文件格式

特点\类型 RAW QCOW2

KVM默认 否 是

I/O效率 高 较高

占用空间 大 小

压缩 不支持 支持

后端盘复用 不支持 支持

快照 不支持 支持

– 直接映射原始盘的数据内容

– 当前端盘的数据有修改时,在修改之前自动将原始盘的旧数据存入前端盘

– 对前端盘的修改不回写到原始盘

前端盘大小最好比原始盘大或相等

1.基于原始盘.node\_tedu.qcow2生成前端盘test.qcow2

]$ cd /var/lib/libvirt/images/

]$ qemu-img create -f qcow2 -b .node\_tedu.qcow2 test.qcow2 10G

]$ qemu-img info test.qcow2 #查看磁盘文件信息

2.修改虚拟机xml文件内容

]$ virsh edit nsd01

虚拟机名称: <name>test</name>

<uuid>删除整行内容,如果删多了按u进行撤销

<source file='/var/lib/libvirt/images/test.qcow2'/>

<mac address='52:54:00:20:37:74'/> 删除整行内容

]$ virsh start test

]$ virsh list --all

###################################################

离线访问虚拟机:虚拟机关机状态,直接修改磁盘文件里面的数据

]$ guestmount -a /var/lib/libvirt/images/tedu\_node09.img -i /home/student/nsdfile/

##################################################

• DNS服务器的功能

– 正向解析:根据注册的域名查找其对应的IP地址

– 反向解析:根据IP地址查找对应的注册域名,不常用

所有的域名都必须以点作为结尾

根域：　　　　　　　　．

一级域名：　.cn .us .tw .hk .jp .kr

二级域名：　tedu.cn .com.cn .net.cn .org.cn

三级域名：　nb.com.cn haha.com.cn haxi.com.cn

完全合格的主机名: 主机头部+域名

该网站的FQDN为: www.qq.com

Full Qualified Domain Name(FQDN),完全合格主机名

###################################################

bind-9.9.4-29.el7.x86\_64 //域名服务包

bind-chroot-9.9.4-29.el7.x86\_64 //提供虚拟根支持,牢笼政策

– 系统服务:named

– 默认端口:TCP/UDP 53

– 运行时的虚拟根环境:/var/named/chroot/

• 主配置文件:/etc/named.conf #设置负责解析域名 tedu.cn

• 地址库文件:/var/named/ #记录域名与IP地址对应关系

虚拟机A构建DNS服务

1.安装软件

[root@svr7 ~]# yum -y install bind bind-chroot

2.修改主配置文件

[root@svr7 ~]# cp /etc/named.conf /etc/named.bak

[root@svr7 ~]# vim /etc/named.conf

options {

directory "/var/named"; #地址库文件存放路径

};

zone "tedu.cn" IN { #设置负责解析的域名

type master; #类型为主DNS服务器

file "tedu.cn.zone"; #地址库文件名称

};

3.建立地址库文件

[root@svr7 ~]# cd /var/named/

[root@svr7 named]# cp -p named.localhost tedu.cn.zone

-p:保持源文件属性不变

[root@svr7 named]# ls -l tedu.cn.zone

[root@svr7 named]# vim tedu.cn.zone

所有的域名都必须以点结尾

如果没有以点结尾,那么自动补全本地址库负责的域名

tedu.cn. NS svr7

svr7 A 192.168.4.7

www A 1.1.1.1

ftp A 2.2.2.2

[root@svr7 named]# systemctl restart named

虚拟机B测试域名解析

1.指定本机DNS服务器/etc/resolv.conf

]# echo nameserver 192.168.4.7 > /etc/resolv.conf

]# cat /etc/resolv.conf

]# nslookup www.tedu.cn

]# nslookup ftp.tedu.cn

##################################################

多区域的DNS服务器

1.修改主配置文件

options {

directory "/var/named";

};

zone "tedu.cn" IN {

type master;

file "tedu.cn.zone";

};

zone "qq.com" IN {

type master;

file "qq.com.zone";

};

2.建立地址库文件qq.com.zone

qq.com. NS svr7

svr7 A 192.168.4.7

www A 3.3.3.3

ftp A 4.4.4.4

3.重启named服务

###################################################

搭建DNS服务器,实现www.sina.com解析结果为192.168.4.100

###################################################

DNS服务器资源解析记录:

NS解析记录

A正向地址解析记录

CNAME解析记录的别名

##################################################

特殊的解析记录

1.DNS轮询,基于DNS的站点负载均衡

– 一个域名 ---> 多个不同IP地址

[root@svr7 /]# vim /var/named/tedu.cn.zone

tedu.cn. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.1

www A 192.168.4.2

www A 192.168.4.3

ftp A 2.2.2.2

2.泛域名解析

虚拟机A

[root@svr7 /]# vim /var/named/tedu.cn.zone

最后追加写入

\* A 1.2.3.4

tedu.cn. A 10.20.30.40

[root@svr7 /]# systemctl restart named

虚拟机B

[root@pc207 /]# nslookup wwww.tedu.cn

[root@pc207 /]# nslookup haxi.tedu.cn

[root@pc207 /]# nslookup tedu.cn

3.有规律的泛域名解析

pc1.tedu.cn ------>192.168.10.1

pc2.tedu.cn ------>192.168.10.2

pc3.tedu.cn ------>192.168.10.3

pc4.tedu.cn ------>192.168.10.4

..........

pc50.tedu.cn ------>192.168.10.50

内置函数: $GENERATE 制造连续范围数字

虚拟机A

[root@svr7 /]# vim /var/named/tedu.cn.zone

最后追加写入

$GENERATE 1-50 pc$ A 192.168.10.$

[root@svr7 /]# systemctl restart named

虚拟机B

[root@pc207 /]# nslookup pc1.tedu.cn

[root@pc207 /]# nslookup pc2.tedu.cn

4.解析记录的别名

[root@svr7 /]# vim /var/named/tedu.cn.zone

最后追加写入

tts CNAME ftp

[root@svr7 /]# systemctl restart named

虚拟机B

[root@pc207 /]# nslookup tts.tedu.cn

###################################################

主机名映射文件:/etc/hosts

1.解析域名过程中,最高优先级

2.只为本机提供解析

虚拟机A:

[root@svr7 /]# vim /etc/hosts

[root@svr7 /]# tail -1 /etc/hosts

最后追加写入

192.168.4.110 www.360.com

[root@svr7 /]# ping www.360.com

###################################################

虚拟机B,构建DNS服务器,负责解析bj.qq.com

1.安装软件bind bind-chroot

2.修改主配置文件

options {

directory "/var/named";

};

zone "bj.qq.com" IN {

type master;

file "bj.qq.com.zone";

};

[root@pc207 ~]# cd /var/named/

[root@pc207 named]# cp -p named.localhost bj.qq.com.zone

[root@pc207 named]# vim bj.qq.com.zone

bj.qq.com. NS pc207

pc207 A 192.168.4.207

www A 11.12.13.14

[root@pc207 named]# systemctl restart named

####################################################

DNS子域授权

父域:www.qq.com

子域:www.bj.qq.com

虚拟机A为DNS服务,负责qq.com域名的解析

虚拟机B为DNS服务,负责bj.qq.com域名的解析

[root@pc207 /]# nslookup www.qq.com 192.168.4.7

[root@pc207 /]# nslookup www.bj.qq.com 192.168.4.207

解析请求www.bj.qq.com询问父域DNS服务器虚拟机A,也能够解析

虚拟机A:

[root@svr7 ~]# vim /var/named/qq.com.zone

qq.com. NS svr7

bj.qq.com. NS pc207

svr7 A 192.168.4.7

pc207 A 192.168.4.207

www A 3.3.3.3

ftp A 4.4.4.4

[root@svr7 ~]# systemctl restart named

[root@pc207 /]# nslookup www.bj.qq.com 192.168.4.7

Server: 192.168.4.7

Address: 192.168.4.7#53

Non-authoritative answer: #非权威解答

Name: www.bj.qq.com

Address: 11.12.13.14

递归查询

客户端请求域名解析,主DNS服务器与其他DNS服务器交互,最终将解析结果带回来

[root@svr7 ~]# vim /etc/named.conf

options {

directory "/var/named";

recursion no; #禁止DNS递归查询

};

迭代查询

主DNS服务器与其他DNS服务器交互

####################################################

缓存DNS服务器,缓存解析记录加快解析

虚拟机A:互联网真DNS服务器

虚拟机B:缓存DNS服务器

虚拟机C:测试客户端

虚拟机C解析请求发送给虚拟机B,虚拟机B将请求发送给虚拟机A

一 准备虚拟机C,配置IP地址为192.168.4.10/24

二 配置虚拟机B

[root@pc207 /]# vim /etc/named.conf

options {

directory "/var/named";

forwarders { 192.168.4.7; }; #转发给192.168.4.7

};

[root@pc207 /]# systemctl restart named

[root@pc207 /]# nslookup www.tedu.cn 192.168.4.207

###################################################

综合实验:

虚拟机A:DNS服务器

虚拟机B:Web服务器,提供www.tedu.cn与www.qq.com

虚拟机C:客户端测试

虚拟机B:

1.安装httpd软件包

[root@pc207 /]# yum -y install httpd

2.建立调用配置文件

[root@pc207 /]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName www.tedu.cn

DocumentRoot /var/www/tedu

</VirtualHost>

<VirtualHost \*:80>

ServerName www.qq.com

DocumentRoot /var/www/qq

</VirtualHost>

[root@pc207 /]# mkdir /var/www/tedu /var/www/qq

[root@pc207 /]# echo '<h1>tedu' > /var/www/tedu/index.html

[root@pc207 /]# echo '<h1>QQ' > /var/www/qq/index.html

[root@pc207 /]# systemctl restart httpd

虚拟机A:

1.修改地址库文件记录

[root@svr7 ~]# vim /var/named/tedu.cn.zone

tedu.cn. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.207

[root@svr7 ~]# vim /var/named/qq.com.zone

qq.com. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.207

[root@svr7 ~]# !sys

systemctl restart named

[root@svr7 ~]#

虚拟机C:测试

1.指定DNS服务器地址

]# echo nameserver 192.168.4.7 > /etc/resolv.conf

]# nslookup www.qq.com

]# nslookup www.tedu.cn

2.测试访问Web页面

]# firefox www.qq.com

]# firefox www.tedu.cn

Split分离解析(视图解析)

• 当收到客户机的DNS查询请求的时候

– 能够区分客户机的来源地址

– 为不同类别的客户机提供不同的解析结果(IP地址)

– 为不同的客户端,提供最近的解析结果

案例需求及要点

• 环境及需求

– 权威DNS:svr7.tedu.cn 192.168.4.7

– 负责区域:tedu.cn

– A记录分离解析 —— 以 www.tedu.cn 为例

客户机来自 解析结果

192.168.4.207 ---www.tedu.cn----> 192.168.4.100

其他地址 ---www.tedu.cn-----> 1.2.3.4

分离解析配置:

1.客户端匹配原则为,由上到下,匹配即停止

2.每一个客户端都必须找到自己的分类

3.所有的zone都必须在view中

虚拟机A:

1.修改主配置文件

options {

directory "/var/named";

};

view "nsd" {

match-clients { 192.168.4.207; };

zone "tedu.cn" IN {

type master;

file "tedu.cn.zone";---->写入解析结果192.168.4.100

};

};

view "other" {

match-clients { any; };

zone "tedu.cn" IN {

type master;

file "tedu.cn.other";---->写入解析结果1.2.3.4

};

};

2.建立不同地址库文件

####################################################

多区域的分离解析

客户机来自 解析结果

192.168.4.207 -------> 192.168.4.100

其他地址 --------> 1.2.3.4

1.每一个view中zone的个数,以及zone负责的域名均要一致

view "nsd" {

match-clients { 192.168.4.207; };

zone "tedu.cn" IN {

type master;

file "tedu.cn.zone";

};

zone "qq.com" IN {

type master;

file "qq.com.zone";

};

};

view "other" {

match-clients { any; };

zone "tedu.cn" IN {

type master;

file "tedu.cn.other";

};

zone "qq.com" IN {

type master;

file "qq.com.other";

};

};

###################################################

了解内容:

acl地址列表

acl test { 192.168.4.207; 192.168.4.1; 192.168.7.0/24; 192.168.4.2; 192.168.4.27; };

view "nsd" {

match-clients { test; };

zone "tedu.cn" IN {

type master;

file "tedu.cn.zone";

};

zone "qq.com" IN {

type master;

file "qq.com.zone";

};

};

####################################################

RAID磁盘阵列

• 廉价冗余磁盘阵列

– Redundant Arrays of Inexpensive Disks

– 通过硬件/软件技术,将多个较小/低速的磁盘整合成一个大磁盘

– 阵列的价值:提升I/O效率、硬件级别的数据冗余

– 不同RAID级别的功能、特性各不相同

• RAID 0,条带模式

– 同一个文档分散存放在不同磁盘

– 并行写入以提高效率

– 至少由两块磁盘组成

• RAID 1,镜像模式

– 一个文档复制成多份,分别写入不同磁盘

– 多份拷贝提高可靠性,效率无提升

– 至少由两块磁盘组成

• RAID5,高性价比模式

– 相当于RAID0和RAID1的折中方案

– 需要至少一块磁盘的容量来存放校验数据

– 至少由三块磁盘组成

• RAID6,高性价比/可靠模式

– 相当于扩展的RAID5阵列,提供2份独立校验方案

– 需要至少两块磁盘的容量来存放校验数据

– 至少由四块磁盘组成

• RAID 0+1/RAID 1+0

– 整合RAID 0、RAID 1的优势

– 并行存取提高效率、镜像写入提高可靠性

– 至少由四块磁盘组成

####################################################

进程管理

程序:静态的代码 占用磁盘空间

进程:动态执行的代码 占用CPU 内存

父进程与子进程 结构:树型结构

僵尸进程 孤儿进程

PID:进程的编号

查看进程树

• pstree — Processes Tree

– 格式:pstree [选项] [PID或用户名]

systemd:上帝进程,所有进程的父进程

[root@svr7 /]# pstree

• 常用命令选项

– -a:显示完整的命令行

– -p:列出对应PID编号

[root@svr7 ~]# pstree lisi #查看lisi用户开启的进程

bash───vim

[root@svr7 ~]# pstree -p lisi

bash(3394)───vim(3419)

[root@svr7 ~]# pstree -ap lisi

bash,3394

└─vim,3419 a.txt

[root@svr7 ~]#

• ps aux 操作

– 列出正在运行的所有进程,信息非常全面

用户 进程ID %CPU %内存 虚拟内存 固定内存 终端 状态 起始时间 CPU时间 程序指令

• ps -elf 操作

– 列出正在运行的所有进程,可以显示该进程的父进程的PID

PPID:父进程的PID号

请计算当前系统中运行的进程个数?

]# wc -l /etc/passwd

]# ps aux | wc -l

]# find /etc/ -name '\*tab' -type f

]# find /etc/ -name '\*tab' -type f | wc -l

]# find /etc/ -name '\*.conf' -type f

]# find /etc/ -name '\*.conf' -type f | wc -l

###################################################

进程动态排名

• top 交互式工具

– 格式:top [-d 刷新秒数] [-U 用户名]

[root@svr7 ~]# top -d 1

P(大写)进行CPU排序

M(大写)进行内存排序

###################################################

检索进程

• pgrep — Process Grep

– 用途:pgrep [选项]... 查询条件

• 常用命令选项

– -l:输出进程名,而不仅仅是 PID

– -U:检索指定用户的进程

– -x:精确匹配完整的进程名

[root@svr7 ~]# pgrep -l a

[root@svr7 ~]# pgrep -l log

[root@svr7 ~]# pgrep -U lisi

[root@svr7 ~]# pgrep -lU lisi

[root@svr7 ~]# pstree -ap lisi

################################################

进程的前后台调度

• 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

• bg 命令

– 激活后台被挂起的任务

[root@svr7 ~]# sleep 1000 & #将进程正在运行放入后台

[root@svr7 ~]# jobs #查看后台进程信息

[root@svr7 ~]# sleep 800

^Z #按Ctrl+z 暂停放入后台

[2]+ 已停止 sleep 800

[root@svr7 ~]# jobs

[root@svr7 ~]# bg 2 #将后台编号为2的进程,继续运行

[root@svr7 ~]# jobs

[root@svr7 ~]# fg 1 #将后台编号为1的进程,恢复到前台运行

sleep 1000

^C #按Ctrl+c结束

[root@svr7 ~]# jobs

[root@svr7 ~]# fg 2 #将后台编号为2的进程,恢复到前台运行

sleep 800

^C #按Ctrl+c结束

###################################################

• 干掉进程的不同方法

– Ctrl+c 组合键,中断当前命令程序

– kill [-9] PID... 、kill [-9] %后台任务编号

– killall [-9] 进程名...

– pkill 查找条件

[root@svr7 ~]# sleep 100 &

[root@svr7 ~]# sleep 200 &

[root@svr7 ~]# sleep 300 &

[root@svr7 ~]# jobs -l

[root@svr7 ~]# kill 3573

[root@svr7 ~]# jobs

[root@svr7 ~]# killall sleep

[root@svr7 ~]# jobs

[root@svr7 ~]# sleep 400 &

[root@svr7 ~]# jobs

[root@svr7 ~]# killall -9 sleep #强制杀

[root@svr7 ~]# jobs

强制踢出一个用户(杀死一个用户开启的所有进程)

[root@svr7 ~]# killall -9 -u lisi

#################################################

日志管理

日志的功能

• 系统和程序的“日记本”

– 记录系统、程序运行中发生的各种事件

– 通过查看日志,了解及排除故障

– 信息安全控制的 依据

• 由系统服务rsyslog统一记录/管理

• 常见的日志文件

/var/log/messages 记录内核消息、各种服务的公共消息

/var/log/dmesg 记录系统启动过程的各种消息

/var/log/cron 记录与cron计划任务相关的消息

/var/log/maillog 记录邮件收发相关的消息

/var/log/secure 记录与访问限制相关的安全消息

日志分析

• 通用分析工具

– tail、tailf、less、grep等文本浏览/检索命令

– awk、sed等格式化过滤工具

tailf:实时跟踪日志消息

[root@svr7 ~]# echo 123 > /opt/1.txt

[root@svr7 ~]# cat /opt/1.txt

123

[root@svr7 ~]# tailf /opt/1.txt

123

haha

• users、who、w 命令

– 查看已登录的用户信息,详细度不同

pts/0:图形终端

• last、lastb 命令

– 查看最近登录成功/失败的用户信息

[root@svr7 ~]# users

[root@svr7 ~]# who

[root@svr7 ~]# w

[root@svr7 ~]# last -2 #最近登录成功的用户记录

[root@svr7 ~]# lastb -2 #最近登录失败的用户记录

################################################

日志消息的优先级

• Linux内核定义的事件紧急程度

– 分为 0~7 共8种优先级别

– 其数值越小,表示对应事件越紧急/重要

0 EMERG（紧急） 会导致主机系统不可用的情况

1 ALERT（警告） 必须马上采取措施解决的问题

2 CRIT（严重） 比较严重的情况

3 ERR（错误） 运行出现错误

4 WARNING（提醒） 可能会影响系统功能的事件

5 NOTICE（注意） 不会影响系统但值得注意

6 INFO（信息） 一般信息

7 DEBUG（调试） 程序或系统调试信息等

##################################################

使用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"

[root@svr7 ~]# journalctl -u httpd

[root@svr7 ~]# journalctl -xe

###############################################

systemctl控制

• Linux系统和服务管理器

– 是内核引导之后加载的第一个初始化进程(PID=1)

– 负责掌控整个Linux的运行/服务资源组合

systemd

• 一个更高效的系统&服务管理器

– 开机服务并行启动,各系统服务间的精确依赖

– 配置目录:/etc/systemd/system/

– 服务目录:/lib/systemd/system/

– 主要管理工具:systemctl

对于服务的管理

systemctl restart 服务名 #重起服务

systemctl start 服务名 #开启服务

systemctl stop 服务名 #停止服务

systemctl status 服务名 #查看服务当前的状态

systemctl enable 服务名 #设置服务开机自启动

systemctl disable 服务名 #设置服务不开机自启动

RHEL6 运行级别

0：关机

1：单用户模式（基本功能的实现，破解Linux密码）

2：多用户字符界面（不支持网络）

3：多用户字符界面（支持网络）服务器默认的运行级别

4：未定义

5：图形界面

6：重起

切换运行级别：init 5

RHEL7 运行模式

字符模式：multi-user.target

图形模式：graphical.target

[student@room9pc01 ~]$ rht-vmctl reset classroom

classroom [OK]

域 classroom 已开始

[student@room9pc01 ~]$ rht-vmctl reset server

server [OK]

域 server 已开始

[root@svr7 /]# ls -l /lib/systemd/system/

当前直接切换到字符模式

[root@svr7 /]# systemctl isolate multi-user.target

当前直接切换到图形模式

[root@svr7 /]# systemctl isolate graphical.target

查看每次开机默认进入模式

[root@svr7 /]# systemctl get-default

graphical.target

设置永久策略，每次开机自动进入graphical.target

# systemctl set-default multi-user.target

# reboot

#################################################

综合实验:

1.构建虚拟机C与虚拟机D的Web服务器

虚拟机C:

[root@svr10 ~]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName www.qq.com

DocumentRoot /var/www/qq

</VirtualHost>

<VirtualHost \*:80>

ServerName www.163.com

DocumentRoot /var/www/163

</VirtualHost>

[root@svr10 ~]# mkdir /var/www/qq /var/www/163

[root@svr10 ~]# echo '<h1>Web1 QQ' > /var/www/qq/index.html

[root@svr10 ~]# echo '<h1>Web1 163' > /var/www/163/index.html

虚拟机D

[root@svr10 ~]# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName www.qq.com

DocumentRoot /var/www/qq

</VirtualHost>

<VirtualHost \*:80>

ServerName www.163.com

DocumentRoot /var/www/163

</VirtualHost>

[root@svr10 ~]# mkdir /var/www/qq /var/www/163

[root@svr10 ~]# echo '<h1>Web2 QQ' > /var/www/qq/index.html

[root@svr10 ~]# echo '<h1>Web2 163' > /var/www/163/index.html

2.虚拟机A构建DNS服务器

[root@svr7 ~]# vim /etc/named.conf

options {

directory "/var/named";

};

view "nsd" {

match-clients { 192.168.4.7; };

zone "163.com" IN {

type master;

file "163.com.zone";

};

zone "qq.com" IN {

type master;

file "qq.com.zone";

};

};

view "other" {

match-clients { any; };

zone "163.com" IN {

type master;

file "163.com.other";

};

zone "qq.com" IN {

type master;

file "qq.com.other";

};

};

[root@svr7 ~]# cd /var/named/

[root@svr7 named]# vim qq.com.zone

qq.com. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.10

[root@svr7 named]# vim qq.com.other

qq.com. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.20

[root@svr7 named]# cp -p qq.com.zone 163.com.zone

[root@svr7 named]# vim 163.com.zone

163.com. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.10

[root@svr7 named]# cp -p 163.com.zone 163.com.other

[root@svr7 named]# vim 163.com.other

163.com. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.20

[root@svr7 /]# systemctl restart named

3.指定DNS服务器地址

[root@svr7 /]# echo 'nameserver 192.168.4.7' > /etc/resolv.conf

[root@pc207 ~]# echo 'nameserver 192.168.4.7' > /etc/resolv.conf

[root@svr10 ~]# echo 'nameserver 192.168.4.7' > /etc/resolv.conf

[root@pc20 ~]# echo 'nameserver 192.168.4.7' > /etc/resolv.conf

###################################################

部署DHCP服务器

• Dynamic Host Configuration Protocol

– 动态主机配置协议,由 IETF(Internet 网络工程师任

务小组)组织制定,用来简化主机地址分配管理

• 主要分配以下入网参数

– IP地址/子网掩码/广播地址

– 默认网关地址、DNS服务器地址

• DHCP地址分配的四次会话(以广播形式进行,先到先得)

– DISCOVERY --> OFFER --> REQUEST -->ACK

在一个网络中,只能有一个DHCP服务器

• 服务端基本概念

– 租期:允许客户机租用IP地址的时间期限,单位为秒

– 作用域:分配给客户机的IP地址所在的网段

– 地址池:用来动态分配的IP地址的范围

虚拟机A:

1.安装dhcp软件包

[root@svr7 ~]# yum -y install dhcp

2.修改配置文件

[root@svr7 ~]# vim /etc/dhcp/dhcpd.conf

末行模式下:

:r /usr/share/doc/dhcp\*/dhcpd.conf.example

subnet 192.168.4.0 netmask 255.255.255.0 { #分配的网段

range 192.168.4.100 192.168.4.200; #分配的IP范围

option domain-name-servers 192.168.4.7; #分配DNS地址

option routers 192.168.4.254; #分配网关地址

default-lease-time 600;

max-lease-time 7200;

}

3.重启dhcpd服务

[root@svr7 ~]# systemctl restart dhcpd

###################################################

网络装机概述

网络装机的优势

• 规模化:同时装配多台主机

• 自动化:装系统、配置各种服务

• 远程实现:不需要光盘、U盘等物理安装介质

什么是PXE网络

• PXE,Pre-boot eXecution Environment

– 预启动执行环境,在操作系统之前运行

– 可用于远程安装

• 工作模式

– PXE client 集成在网卡的启动芯片中

– 当计算机引导时,从网卡芯片中把PXE client调入内存

执行,获取PXE server配置、显示菜单,根据用户选

择将远程引导程序下载到本机运行

•服务端需要哪些服务组件

– DHCP服务,分配IP地址、定位引导程序

– TFTP服务,提供引导程序下载

– HTTP服务,提供yum安装源

###################################################

在虚拟机A构建PXE网络装机服务器

一 配置DHCP服务器,指定下一个服务器地址

[root@svr7 ~]# vim /etc/dhcp/dhcpd.conf

subnet 192.168.4.0 netmask 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;

next-server 192.168.4.7; #指定下一个服务器地址

filename "pxelinux.0"; #指定网卡引导文件名称

}

[root@svr7 ~]# systemctl restart dhcpd

pxelinux.0:网卡引导文件 ,安装说明书,二进制文件

安装一个软件即可获得,默认生成的名字pxelinux.0

####################################################

二 构建tftp服务

tftp:简单的文件传输协议

端口默认为:69

默认共享位置:/var/lib/tftpboot

1.安装软件包为tftp-server

[root@svr7 ~]# yum -y install tftp-server

2.重启服务

[root@svr7 ~]# systemctl restart tftp

3.部署pxelinux.0引导文件

]# yum provides \*/pxelinux.0 #查询那个包产生的该文件

]# yum -y install syslinux

]# rpm -ql syslinux | grep pxelinux.0 #查询软件包安装清单

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

]# ls /var/lib/tftpboot/

客户端----->DHCP---->next-server--->pxelinux.0

pxelinux.0---->/var/lib/tftpboot/pxelinux.cfg/default

4.部署默认菜单文件

首先关闭虚拟机A,图形添加一个光驱设备,放入光盘文件

[root@svr7 ~]# ls /dev/cdrom

[root@svr7 ~]# mount /dev/cdrom /mnt/

[root@svr7 ~]# ls /mnt/

]# mkdir /var/lib/tftpboot/pxelinux.cfg

]# cp /mnt/isolinux/isolinux.cfg /var/lib/tftpboot/pxelinux.cfg/default

]# ls -l /var/lib/tftpboot/pxelinux.cfg/default

5.部署图形模块(vesamenu.c32)和背景图片(splash.png)

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

]# ls /var/lib/tftpboot/

pxelinux.0 pxelinux.cfg splash.png vesamenu.c32

6.部署启动内核(vmlinuz)和驱动程序(initrd.img)

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

]# ls /var/lib/tftpboot/

initrd.img pxelinux.cfg vesamenu.c32

pxelinux.0 splash.png vmlinuz

##################################################

总结:

dhcp--->IP地址 next-server filename

tftp--->pxelinux.0

pxelinux.0--->读取default菜单文件

default--->vesamenu.c32 splash.png vmlinuz initrd.img

7.修改菜单文件内容

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

末行模式 :set nu

1 default vesamenu.c32 #默认加载运行图形的模块

2 timeout 600 #默认读秒时间 1/10秒

10 menu background splash.png #背景图片

11 menu title NSD1904 PXE Server #显示标题信息

61 label linux

62 menu label Install CentOS7.5 #屏幕显示信息

63 menu default #读秒结束后默认选项

63 kernel vmlinuz #指定启动内核

64 append initrd=initrd.img #指定驱动程序

##################################################

[root@svr7 ~]# systemctl restart dhcpd

[root@svr7 ~]# systemctl restart tftp

##################################################

初步测试:

新建一台全新的虚拟机

1.安装方式选择:PXE网络引导安装

2.内存必须2G

3.网络类型选择:private1

总结:

dhcp--->IP地址 next-server filename

tftp--->pxelinux.0

pxelinux.0--->读取default菜单文件

default--->vesamenu.c32 splash.png vmlinuz initrd.img

#################################################

三 构建Web服务器,将光盘内容共享给客户端

1.安装软件包

[root@svr7 ~]# yum -y install httpd

[root@svr7 ~]# mkdir /var/www/html/centos

[root@svr7 ~]# ls /var/www/html/centos

[root@svr7 ~]# mount /dev/cdrom /var/www/html/centos

mount: /dev/sr0 写保护，将以只读方式挂载

[root@svr7 ~]# ls /var/www/html/centos

[root@svr7 ~]# systemctl restart httpd

[root@svr7 ~]# firefox http://192.168.4.7/centos

##################################################

四 部署无人值守安装,生成应答文件

1.安装一个system-config-kickstart图形化软件,生成应答文件

[root@svr7 ~]# yum -y install system-config-kickstart

2.运行图形化工具

以英文语言进行运行

[root@svr7 ~]# LANG=en system-config-kickstart

点击 "软件包选择(Package Selection)" 查看是否可以进行选择

需要Yum仓库的支持:指向光盘的内容仓库,标识必须为 [development]

[root@svr7 ~]# vim /etc/yum.repos.d/local.repo

[development]

name=CentOS-$releasever - Base

baseurl=ftp://192.168.4.254/centos-1804

enabled=1

gpgcheck=0

[root@svr7 ~]# LANG=en system-config-kickstart

点击 "软件包选择(Package Selection)" 查看是否可以进行选择

[root@svr7 ~]# ls /root/ks.cfg

/root/ks.cfg

[root@svr7 ~]# vim /root/ks.cfg

3.利用Web服务器共享应答文件

[root@svr7 ~]# cp /root/ks.cfg /var/www/html/

[root@svr7 ~]# ls /var/www/html/

centos ks.cfg

[root@svr7 ~]# firefox http://192.168.4.7/ks.cfg

4.通过菜单文件,指定ks应答文件获取方式

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

label linux

menu label Install CentOS7.5

menu default

kernel vmlinuz

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

]#

##################################################

总结:

dhcp--->IP地址 next-server filename

tftp--->pxelinux.0

pxelinux.0--->读取default菜单文件

default--->图形模块 背景 内核 驱动 ks应答文件

ks应答文件--->语言 分区 密码 时区 ... 指定获取软件包的方式

获取软件包的方式---> --url="http://192.168.4.7/centos"

###################################################

总结:

dhcp--->IP地址 next-server filename

tftp--->pxelinux.0

pxelinux.0--->读取default菜单文件

default--->图形模块 背景 内核 驱动 ks应答文件

ks应答文件--->语言 分区 密码 时区 ... 指定获取软件包的方式

获取软件包的方式---> --url="http://192.168.4.7/centos"

###################################################

rsync同步操作

rsync同步操作

• 命令用法

– rsync [选项...] 源目录 目标目录

• 同步与复制的差异

– 复制:完全拷贝源到目标

– 同步:增量拷贝,只传输变化过的数据

同步控制

• rsync操作选项

– -n:测试同步过程,不做实际修改

– --delete:删除目标文件夹内多余的文档

– -a:归档模式,相当于-rlptgoD

– -v:显示详细操作信息

– -z:传输过程中启用压缩/解压

[root@svr7 ~]# mkdir /dir01 /abc

[root@svr7 ~]# cp /etc/passwd /etc/fstab /dir01

[root@svr7 ~]# ls /dir01

[root@svr7 ~]# rsync -avz /dir01 /abc/ #同步目录本身

[root@svr7 ~]# ls /abc

[root@svr7 ~]# ls /abc/dir01/

[root@svr7 ~]# rsync -avz /dir01/ /abc/ #同步目录内容

[root@svr7 ~]# ls /abc/

[root@svr7 ~]# touch /dir01/1.txt

[root@svr7 ~]# ls /dir01/

[root@svr7 ~]# rsync -avz /dir01/ /abc/

[root@svr7 ~]# ls /abc/

[root@svr7 ~]# ls /dir01/

[root@svr7 ~]# ls /abc/

[root@svr7 ~]# touch /abc/haha.txt

[root@svr7 ~]# ls /abc/

[root@svr7 ~]# rsync -avz --delete /dir01/ /abc/

[root@svr7 ~]# ls /abc/

[root@svr7 ~]# touch /abc/{1..5}.txt

[root@svr7 ~]# ls /abc/

[root@svr7 ~]# rsync -avz --delete /dir01/ /abc/

[root@svr7 ~]# ls /abc/

###################################################

rsync+SSH同步

• 与远程的 SSH目录保持同步

– 下行:rsync [...] user@host:远程目录 本地目录

– 上行:rsync [...] 本地目录 user@host:远程目录

虚拟机A:

[root@svr7 ~]# rsync -avz --delete /dir01/ root@192.168.4.207:/opt/

虚拟机B:

[root@pc207 ~]# ls /opt/

###################################################

虚拟机A:

[root@svr7 ~]# touch /dir01/xixi.txt

[root@svr7 ~]# rsync -avz --delete /dir01/ root@192.168.4.207:/opt/

虚拟机B:

[root@pc207 ~]# ls /opt

###################################################

实时同步

虚拟机A

一 生成公钥与私钥,实现ssh无密码验证

[root@svr7 ~]# ssh-keygen #一路回车

[root@svr7 ~]# ls /root/.ssh/

authorized\_keys(别的机器传递过来的公钥) id\_rsa(私钥) id\_rsa.pub(公钥)

[root@svr7 ~]# ssh-copy-id root@192.168.4.207

[root@svr7 ~]# rsync -avz --delete /dir01/ root@192.168.4.207:/opt/

二 监控目录内容的变化

安装inotify-tools工具

真机上:

[student@room9pc01 ~]$ ls /linux-soft/01/

Cobbler.zip tools.tar.gz

[student@room9pc01 ~]$ scp /linux-soft/01/tools.tar.gz root@192.168.4.7:/root

1.让虚拟机A具备inotify-tools-3.13.tar.gz

[root@svr7 /]# ls /root/

[root@svr7 /]# mkdir /test

[root@svr7 /]# tar -xf /root/tools.tar.gz -C /test

[root@svr7 /]# ls /test

[root@svr7 /]# ls /test/tools/

步骤1:安装依赖关系包(安装开发工具)

]# yum -y install gcc make

步骤2:tar解包,释放源代码至指定目录

]# tar -xf /test/tools/inotify-tools-3.13.tar.gz -C /opt

]# ls /opt

]# cd /opt/inotify-tools-3.13/

]# ls

步骤3: ./configure 配置,指定安装目录/功能模块等选项

]# cd /opt/inotify-tools-3.13/

]# ./configure

步骤4:make 编译,生成可执行的二进制程序文件

]# make

步骤5:make install 安装,将编译好的文件复制到安装目录

]# make install

]# ls /usr/local/bin/inotifywait

inotifywait监控

• 基本用法

– inotifywait [选项] 目标文件夹

• 常用命令选项

– -m,持续监控(捕获一个事件后不退出)

– -r,递归监控、包括子目录及文件

– -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

三 书写Shell脚本

for循环:适合有规定次数的循环

while适合死循环:次数无限及无法确定

while [条件]

do

重复执行操作

done

[root@svr7 /]# vim /opt/rsync.sh

#!/bin/bash

while inotifywait -rqq /dir01

do

rsync -az --delete /dir01/ root@192.168.4.207:/opt/

done

[root@svr7 /]# chmod +x /opt/rsync.sh

[root@svr7 /]# /opt/rsync.sh

#################################################

Cobbler装机平台,不同版本的多系统的安装

教学环境虚拟机3.1版本进行扩容

[root@svr7 ~]# df -h

文件系统 容量 已用 可用 已用% 挂载点

/dev/vda1 2.0G 1.8G 256M 88% /

[root@svr7 ~]# LANG=en growpart /dev/vda 1

[root@svr7 ~]# df -h

[root@svr7 ~]# xfs\_growfs /dev/vda1

[root@svr7 ~]# df -h

文件系统 容量 已用 可用 已用% 挂载点

/dev/vda1 20G 1.8G 19G 9% /

一 具备一个CentOS虚拟机

[root@svr7 ~]# cat /etc/redhat-release

CentOS Linux release 7.5.1804 (Core)

二、虚拟机设置

1.设置防火墙为trusted

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

2.当前及永久设置SELinux状态为permissive

]# setenforce 0 #当前临时关闭

]# getenforce

]# vim /etc/selinux/config

SELINUX=

##################################################

[root@svr7 ~]# LANG=en growpart /dev/vda 1

[root@svr7 ~]# df -h

[root@svr7 ~]# xfs\_growfs /dev/vda1

##################################################

三、利用scp真机传递cobbler.zip包到虚拟机192.168.4.7中

# scp /home/student/桌面/cobbler.zip root@192.168.4.7:/root/

##################################################

四、搭建Cobbler装机平台

Cobbler概述软件，管理dhcp、tftp、Web服务

自由的导入镜像与ks应答文件

1.解压cobbler.zip包

[root@svr7 ~]# yum -y install unzip

[root@svr7 ~]# unzip /root/cobbler.zip -d /

[root@svr7 ~]# ls /cobbler/

cobbler\_boot.tar.gz cobbler\_web.png

cobbler\_rpm.zip

[root@svr7 ~]#

[root@svr7 ~]# unzip /cobbler/cobbler\_rpm.zip -d /opt/

[root@svr7 ~]# ls /opt/cobbler/

#####################################################

五、安装cobbler主程序、工具包等

]# yum -y install dhcp httpd mod\_ssl

]# yum -y install /opt/cobbler/\*.rpm

[root@svr7 ~]# rpm -q cobbler

cobbler-2.8.2-1.el7.x86\_64

[root@svr7 ~]#

################################################################

cobbler网络装机部署

1.安装软件 cobbler cobbler-web dhcp tftp-server pykickstart httpd tftp-server

cobbler #cobbler程序包

cobbler-web #cobbler的web服务包

pykickstart #cobbler检查kickstart语法错误

httpd #Apache web服务

dhcp #dhcp服务

tftp-server #tftp服务

2.配置cobbler

[root@svr7 /]# vim /etc/cobbler/settings

next\_server: 192.168.4.7 #设置下一个服务器还为本机

server: 192.168.4.7 #设置本机为cobbler服务器

manage\_dhcp: 1 #设置cobbler管理dhcp服务

pxe\_just\_once: 1 #防止客户端重复安装操作系统

开机启动项： 匹配及停止

1.硬盘启动 2.光驱设备 3.U盘 4.网络引导

3.配置cobbler的dhcp

[root@svr7 /]# vim /etc/cobbler/dhcp.template

:%s /旧/新/g #全文替换

:%s /192.168.1/192.168.4/g #全文替换

5 次替换，共 4 行

4.绝对路径解压cobbler\_boot.tar.gz #众多的引导文件

# tar -tf /cobbler/cobbler\_boot.tar.gz #查看包里面内容

# tar -xPf /cobbler/cobbler\_boot.tar.gz #绝对路径释放

# ls /var/lib/cobbler/loaders/

5.启动相关服务

]# systemctl restart cobblerd

]# systemctl enable cobblerd

]# systemctl restart httpd

]# systemctl enable httpd

]# systemctl restart tftp

]# systemctl enable tftp

]# systemctl restart rsyncd #同步服务

]# systemctl enable rsyncd

6.同步刷新cobbler配置

[root@svr7 /]# cobbler sync #检测所有配置是否正确

......

\*\*\* TASK COMPLETE \*\*\*

]# firefox https://192.168.4.7/cobbler\_web

用户名：cobbler

密码：cobbler

##########################################################################################################

cobbler应用

cobbler import --path=挂载点 --name=导入系统命名（随意起）

导入安装镜像数据

# mkdir /dvd

# mount /dev/cdrom /dvd

# ls /dvd/

# cobbler import --path=/dvd --name=CentOS7

cobbler导入的镜像放在：/var/www/cobbler/ks\_mirror

[root@cobbler /]# cobbler list #查看有哪些系统

distros:

CentOS7-x86\_64 #安装客户端至少2G内存

profiles:

CentOS7-x86\_64

##############################################

删除

# cobbler list

# cobbler profile remove --name=CentOS7-x86\_64 #删除菜单信息

# cobbler distro remove --name=CentOS7-x86\_64 #删除镜像信息

###########################################

# umount /dvd/

# mount /dev/cdrom /dvd 将光盘换成RHEL7

# ls /dvd/

安装完成机器 默认root 密码 cobbler

############################################################################################################

自定义应答文件：开头注释行删除

[root@cobbler ~]# system-config-kickstart #生成ks文件

必须默认kickstart文件存放位置：/var/lib/cobbler/kickstarts/

[root@cobbler ~]# cobbler list

修改kickstart文件：

[root@cobbler ~]# cobbler profile edit --name=CentOS7 --kickstart=/var/lib/cobbler/kickstarts/自定义.cfg

[root@cobbler ~]# cobbler profile report

[root@cobbler ~]# cobbler sync #同步配置

############################################################################################################

day01

200行

cat /etc/shells //查看本机可用解释器

准备环境

rht-vmctl reset classroom

rht-vmctl reset desktop

rht-vmctl reset server

进入desktop或者server中:

yum -y install ksh //安装新解释器

> >> 收集正确信息

2> 2>> 收集错误信息

&> &>> 收集所有信息

< 输入重定向

bash优点:

历史记录,tab键,快捷键,管道, 重定向

一个标准的shell脚本包含:

1, 声明解释器

2, 注释,脚本功能,变量的含义等

3,执行指令(脚本内容)

#!/bin/bash

#这是一个测试脚本

echo "你好"

执行脚本的方式

1,添加完x权限后,使用完整路径

chmod +x 脚本.sh

/opt/脚本.sh

2,bash 脚本.sh

3,source 脚本.sh

bash 执行脚本时开启子进程

source 执行脚本不开启子进程

pstree可以看到进程树

测试用以下脚本:

#!/bin/bash

mkdir abcd

cd abcd

bash看不到进入目录的效果,新开启的bash进程进入目录后的效果随着脚本结束而消失

source可以看到,因为不开启子进程,只使用当前解释器的进程

编写自动配置yum仓库的脚本:

#!/bin/bash

rm -rf /etc/yum.repos.d/\*.repo

echo "[test]

name=abcdefg

baseurl=http://172.25.0.254/content/rhel7.0/x86\_64/dvd/

enabled=1

gpgcheck=0" > /etc/yum.repos.d/test.repo

执行脚本后,使用yum repolist检查

编写自动安装ftp并设置开机自启的脚本:

#!/bin/bash

yum -y install vsftpd &> /dev/null //安装过程不显示

systemctl start vsftpd

systemctl enable vsftpd

systemctl stop firewalld //运行脚本后关闭防火墙用真机链接虚拟机的ftp测试

vim /etc/vsftpd/vsftpd.conf //ftp的配置文件可以根据需要开启匿名上传,修改第29行, 另外/var/ftp/pub需要给写权限

anon\_upload\_enable=YES //创建文件

anon\_mkdir\_write\_enable=YES //创建目录

如果要测试写入权限,需要修改配置后重启服务,并关掉selinux

systemctl restart vsftpd

setenforce 0

变量,使用变量名代替未知量,大大增加脚本灵活

变量名的定义,使用字母,数字,下划线,不能以数字开头,不能用特殊符号

User=10 //等号两边不要出现空格

User=20 //再次给变量赋值会覆盖之前的值

unset User //删除变量

变量种类:

1,自定义变量

2,系统变量,包含

环境变量 PWD,HOME,UID,HOSTNAME,PATH,USER,SHELL,PS1,PS2

位置变量 $1 $2 $3 ....

预定义变量 $0 $? $$ $\* $#

$0 显示脚本名

$$ 显示脚本进程号

$? 显示上一条指令是否成功 ,0是成功,非0是失败

$# 统计位置变量的个数

$\* 显示所有位置变量的内容

#!/bin/bash

echo $1

echo $2

echo $3

echo $0 显示脚本名

echo $\* 所有参数

echo $# 参数的个数

echo $? 上一任务是否成功,0成功,非0失败

echo $$ 显示进程号(PID)

编写脚本实现 创建用户abc,并配置密码123

#!/bin/bash

useradd abc

echo 123 | passwd --stdin abc

改良版

#!/bin/bash

useradd $1

echo $2 | passwd --stdin $1

env 显示所有环境变量

set 显示所有变量

'' 界定范围,屏蔽特殊符号的效果

"" 界定范围

touch a b // 创建2个文件

touch "a b" //创建1个文件

touch 'a b' //创建1个文件

a=10

echo $a //显示变量内容

echo "$a" //显示变量内容

echo '$a' //显示$a ,$的特殊含义被单引号屏蔽

`` 获取命令结果

mv a `date +%Y%m%d` //将a文件改名为当天日期

tar -czf log-`date +%Y%m%d`.tar.gz /var/log //备份日志改名为log-当天日期.tar.gz

编写创建用户配置密码的脚本:

#!/bin/bash

read -p "请输入用户名" User

useradd $User

stty -echo

read -p "请输入密码" Pass

stty echo

echo $Pass | passwd --stdin $User

stty -echo //关闭回显,屏蔽用户输入的内容,不影响系统弹出的信息

stty echo //开启回显

export 发布全局变量,使子进程可以使用父进程定义的变量

通常没有发布全局变量时,定义的变量为局部效果

测试:

a=10 //定义局部变量

export a //修改为全局变量

export b=20 //新定义+发布全局变量

bash //进入子进程

echo $a //检测全局变量是否可用

echo $b

exit //退出子进程

export -n a //取消变量a的全局效果,恢复局部变量效果

bash //进入子进程

echo $a //检测效果,发现无内容

shell中的运算

方法一:

expr + - \\* / %求模(取余数)

注意,运算符号两边必须加空格,乘号需要使用\(转义符号)或者''(单引号)取消原本含义

方法二:使用$[]或$(())表达式

echo $[1+1] 或 echo $((1+1))

echo $[2-1]

echo $[2\*2]

echo $[4/2]

echo $[10%3]

方法三:使用let命令,let在运算时不显示结果,可以方便的对变量进行自增减

let a=a+10 相当于 a+=10,后面是主流写法

a=a-10 a-=10

a=a+1 a++

a=a-1 a--

a=a\*10 a\*=10

a=a/10 a/=10

a=a%10 a%=10

验证可以用echo $a

方法四: bc,可以进行小数运算,scale可以确定小数点后面位数

[root@desktop0 opt]# echo "1+1" | bc

2

[root@desktop0 opt]# echo "11.1+1" | bc

12.1

[root@desktop0 opt]# echo "10/3" | bc

3

[root@desktop0 opt]# echo "scale=3;10/3" | bc

3.333

[root@desktop0 opt]# echo "scale=3;10/3;1+1" | bc

3.333

2

day02

条件测试:

一.字符串匹配

语法格式

使用“test 表达式”或者[ 表达式 ]都可以，表达式两边至少要留一个空格。

1）== 比较两个字符串是否相同

[zengye@svr5 ~]$ [ $USER == "root" ]

[zengye@svr5 ~]$ echo $? //查看结果0为对，非0为错

2）!= 比较两个字符串是否不相同

[zengye@svr5 ~]$ [ $USER != "root" ]

3）一行执行多条命令的情况

# A || B //仅当A命令执行失败，才执行B命令 或

# A && B //仅当A命令执行成功，才执行B命令 且

# A ; B //执行A命令后执行B命令，两者没有逻辑关系

# A && B || C //A且B,或C

4) -z 检查变量的值是否未设置（空值）

! -z 与 - n 一样

[ -z $a ] //测试变量a是否为空,是空返回0

多个逻辑符号连用时:

|| 或,前后任务有一个成功就算成功

&& 且,前后任务都成功,才算成功

[ ! -z $a ] 或 [ -n $a ] //测试变量a,不是空返回0

[ ! -z $a ] && echo "a不是空的" || echo "a是空的"

##############################################################################

二.比较数值大小

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

脚本:每两分钟检查登录用户数量,超过三人报警

[root@server0 opt]# vim ceshi.sh

#!/bin/bash

a=`who|wc -l`

[ $a -gt 3 ] && echo "已有$a人登录" | mail -s test root

[root@server0 opt]# crontab -e -u root

\*/2 \* \* \* \* /bin/bash /opt/ceshi.sh

###################################################

3.识别文件/目录的状态

-r -w -x

-e存在 -f存在且是文件 -d存在且是目录

[ -r txt ] //测试当前用户对文件txt有没有读权限(对root无效)

[ -w txt ] //测试当前用户对文件txt有没有写权限(对root无效)

[ -x txt ] //测试当前用户对文件txt有没有执行权限

[ -e txt ] //txt是否存在,不在乎文件类型

[ -f txt ] //txt是否存在,必须是文件

[ -d txt ] //txt是否存在,必须是目录

4.多个条件/操作的逻辑组合

&& ||

== != -z !-z(-n) -eq -ne -gt -ge -lt -le -r -w -x -e -f -d

##########################################################################################################

if判断

单分支

if 条件测试;then

执行指令

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

if 条件测试;then

命令序列1

elif 条件测试;then

命令序列2

else

命令序列3

fi

########################################################

for循环

for 变量名 in 参数1 参数2 参数3 .....

do

任务指令

done

1.有变量有关,而且关心循环次数

2.只关心循环次数

大括号造数

#!/bin/bash

for i in {0..8}

do

echo nb

done

seq造数工具:

#!/bin/bash

for i in `seq 8`

do

echo nb

done

seq造数工具:

#!/bin/bash

x=5

for i in `seq $x`

do

echo nb

done

批量创建用户:

#!/bin/bash

for i in `cat /opt/user`

do

useradd $i &> /dev/null

done

修改vim tab间隔几个字符

vim /etc/vimrc

第一行添加 set ts=4 //修改为间隔四个字符

编辑脚本实现一次性检测12台主机,显示结果及总数

#!/bin/bash

a=0

b=0

for i in {1..12}

do

ping -c 3 -i 0.2 -W 1 172.25.0.$i &> /dev/null

if [ $? -eq 0 ];then

echo "172.25.0.$i 通了"

let a++

else

echo "172.25.0.$i 不通"

let b++

fi

done

echo "$a台通了,$b 台没通"

while 循环

while 条件测试

do

任务序列

done

day02 2

bash 快捷键,tab,管道,历史记录,重定向

1,声明解释器 2注释脚本功能,重要变量的含义 3脚本代码

1,自定义变量

2,系统变量,包含环境变量 PWD HOSTNAME USER HOME UID PATH PS1 PS2 SHELL ,位置变量 $1 $2 $3.... ,预定义变量 $0 $\* $# $? $$

'' "" `` read -p stty -echo stty echo

export a=10 export -n unset

运算,1 expr + - \(转义)\* / %

2 echo $[1+1]

3 let a\*=10 a=a\*10

a++ a--

条件测试: 使用“test 表达式”或者[ 表达式 ]都可以

1,字符串匹配

[ a == a ] //判断a是否等于a

echo $? //结果为0,正确

[ a != a ] //判断a是否不等于a

echo $? //结果为非0,错误

[ $USER == root ] //判断当前用户是不是管理员

&& 且,前置命令成功后才执行后面命令

|| 或,前置命令失败后才执行后面命令

[ a == a ] && echo ok

[ a == b ] && echo ok

[ a == a ] || echo "前置任务失败了"

[ a == b ] || echo "前置任务失败了"

[ -z $a ] //测试变量a是否为空,是空返回0

[ ! -z $a ] 或 [ -n $a ] //测试变量a,不是空的时候返回0

多个逻辑符号连用时:

&& 且, 前后任务都成功,才算成功

|| 或 ,前后任务有一个成功,就算成功

[ -z $a ] && echo "a是空的" || echo "a不是空的"

[ ! -z $a ] && echo "a不是空的" || echo "a是空的"

2,比较整数值的大小

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

X=20 //定义一个测试变量

[ $X -eq 20 ] && echo "相等" || echo "不相等"

[ $X -eq 30 ] && echo "相等" || echo "不相等"

[ $X -ne 20 ] && echo "不等于" || echo "等于"

[ $X -ne 30 ] && echo "不等于" || echo "等于"

[ $X -gt 10 ] && echo "大于" || echo "否"

[ $X -gt 20 ] && echo "大于" || echo "否"

[ $X -gt 30 ] && echo "大于" || echo "否"

写一个脚本,实现:

每2分钟检测服务器登录用户的数量,如果超过3人,则发邮件给管理员报警

素材:

crontab

who | wc -l

[ -gt 3 ] && echo "报警" | mail -s test root

#!/bin/bash

n=`who | wc -l` //定义变量n,为目前登录账户的数量

[ $n -gt 3 ] && echo "报警" | mail -s test root //用变量n与3对比,如果大于3,则发邮件给管理员,邮件标题是test,内容是"报警"

n=`who | wc -l` 与 [ $n -gt 3 ] 也可以写成[ `who | wc -l` -gt 3 ]

可以提前准备邮件文本,使用输入重定向,邮件内容就是文本内容

mail -s test root < txt

脚本写完后 添加x权限

并在计划任务中添加具体执行时间与脚本位置

crontab -e

\*/2 \* \* \* \* /opt/test1.sh //每2分钟执行一次脚本

/var/spool/mail/root //管理员所有邮件所在的文件,测试前可以先删除此文件

3,识别文件/目录的状态

-r读 -w写 -x执行

-e存在 -f存在且是文件 -d存在且是目录

[ -r txt ] //测试当前用户对文件txt有没有读权限(对root无效)

[ -w txt ] //测试当前用户对文件txt有没有写权限(对root无效)

[ -x txt ] //测试当前用户对文件txt有没有执行权限

[ -e txt ] //txt是否存在,不在乎文件类型

[ -f txt ] //txt是否存在,还必须是文件

[ -d txt ] //txt是否存在,还必须是目录

4,多个条件/操作的逻辑组合

&& ||

== != -z ! -z (-n) -eq -ne -gt -ge -lt -le -r -w -x -e -f -d

if判断

单分支

if 条件测试;then

执行指令

fi

if 条件测试

then

执行指令

fi

双分支

if 条件测试;then

命令序列1

else

命令序列2

fi

ping

使用if双分支编写脚本,测试与其他主机的链接情况:

ping命令可以使用选项增加工作效率

-c 定义ping的次数 -i 定义ping的间隔时间,单位秒 -W 定义反馈结果的时间,单位秒

#!/bin/bash

ping -c 3 -i 0.2 -W 1 $1 &> /dev/null

if [ $? -eq 0 ];then

echo "通了"

else

echo "不通"

fi

#!/bin/bash

[ -z $1 ] && echo "请在脚本名后输入ip" && exit

ping -c 3 -i 0.2 -W 1 $1 &> /dev/null

if [ $? -eq 0 ];then

echo "通了"

else

echo "不通"

fi

#!/bin/bash

read -p "请输入ip地址" ip

ping -c 3 -i 0.2 -W 1 $ip &> /dev/null

if [ $? -eq 0 ];then

echo "通了"

else

echo "不通"

fi

多分支

if 条件测试;then

命令序列1

elif 条件测试;then

命令序列2

else

命令序列3

fi

使用RANDOM获取随机数字,编写猜大小脚本

#!/bin/bash

x=$[RANDOM%11] //使用随机数除11,取余数范围在0-10之间,为变量x

read -p "请输入一个数字(0-10)" n //用户提供一个数字,为变量n

if [ $n -eq $x ];then //利用if多分支将x与n进行对比,如果相等

echo "猜对啦" //反馈信息

elif [ $n -gt $x ];then //如果用户给的数字大于随机数

echo "猜大了" //反馈信息

else

echo "猜小了" //最后一种情况

fi

for循环

for 变量名 in 参数1 参数2 参数3......

do

任务指令

done

1,与变量有关,而且关心循环次数

2,只关心循环次数

#!/bin/bash

for i in {1..8}

do

echo nb

done

#!/bin/bash

for i in `seq 8`

do

echo nb

done

#!/bin/bash

x=5

for i in `seq $x`

do

echo nb

done

利用文本创建账户,账户数量由文本决定

#!/bin/bash

for i in `cat user` //查看文本user后,将内容依次交给i循环

do

useradd $i //第一次循环创建的是文本中的第一个用户,以此类推

done

vim /etc/vimrc

set ts=4 //ts是tabstop,vim编辑器中每次tab间隔4个空格,默认8个

编写脚本实现,一次检测12台主机172.25.0.1~172.25.0.12的网络状况,仅仅显示结果

#!/bin/bash

x=0

y=0

for i in {1..12}

do

ping -c 3 -i 0.2 -W 1 172.25.0.$i &> /dev/null

if [ $? -eq 0 ];then

echo "172.25.0.$i 通了"

let x++ //每通一台,x就+1

else

echo "172.25.0.$i 不通"

let y++ //每不通一台,y就+1

fi

done

echo "$x台通了,$y台不通"

while 条件测试

do

任务序列

done

使用while循环编写猜数脚本,猜对为止,并统计猜的次数

#!/bin/bash

x=$[RANDOM%101]

c=0

while :

do

read -p "请输入一个数字(0-100)" n

let c++

if [ $n -eq $x ];then

echo "猜对了,猜了$c次"

exit

elif [ $n -gt $x ];then

echo "猜大了"

else

echo "猜小了"

fi

done

day03

&& || if

case分支,简化版本if,功能不如if强大,代码比if简洁

case 变量 in

模式1)

命令序列1;;

模式2)

命令序列2;;

\*)

命令序列3

esac

########################################

方便使用nginx的脚本

#1/bin/bash

case $1 in

start)

/usr/local/nginx/sbin/nginx;; //开启服务

stop)

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

re)

/usr/local/nginx/sbin/nginx -s stop //先关闭

/usr/local/nginx/sbin/nginx;; //后开启,重启

status)

netstat -ntulp |grep -q nginx

[ $? -eq 0 ] && echo "服务已开启" || echo "服务未开启";; //查询是否开启

\*)

echo "请输入start|stop|re|status"

esac

break //终止循环,继续执行循环之后的任务

continue //终止本次循环,继续下一次循环

exit //直接退出脚本

day03 2

&& || if

case分支,简化版本ｉｆ，功能不如ｉｆ强大，代码比ｉｆ简洁

case 变量 in

模式1)

命令序列1;;

模式2)

命令序列2;;

模式3)

命令序列3;;

\*)

命令序列4

esac

httpd nginx

编写一键源代码编译安装nginx软件的脚本

源码安装Nginx需要提前安装依赖包软件gcc,openssl-devel,pcre-devel

#!/bin/bash

yum -y install gcc openssl-devel pcre-devel

tar -xf nginx-1.12.2.tar.gz

cd nginx-1.12.2

./configure

make

make install

运行脚本后执行:

/usr/local/nginx/sbin/nginx //开启nginx服务

systemctl stop firewalld //关闭防火墙后,用真机浏览器测试虚拟机的web服务是否正常 http://172.25.0.10

netstat -ntulp | grep nginx

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

编写方便使用nginx服务的工具脚本

#!/bin/bash

case $1 in

start)

netstat -ntulp | grep -q nginx

[ $? -eq 0 ] && echo "服务已经开启" && exit

/usr/local/nginx/sbin/nginx;; //开启服务

stop)

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

re)

/usr/local/nginx/sbin/nginx -s stop //先关闭再开启

/usr/local/nginx/sbin/nginx;; 可以实现重启效果

status)

netstat -ntulp | grep -q nginx //检查服务状态

[ $? -eq 0 ] && echo "服务开启" || echo "服务未开启";;

\*)

echo "start | stop | re | status"

esac

在Shell脚本中，将一些需重复使用的操作，定义为公共的语句块，即可称为函数。通过使用函数，可以使脚本代码更加简洁，增强易读性，提高Shell脚本的执行效率

格式1：

function 函数名 {

命令序列

.. ..

}

格式2：

函数名() {

命令序列

.. ..

}

#!/bin/bash

a(){ //定义函数,名为a

echo abc //函数中的指令,可以是多条

echo xyz

}

a //执行函数

a //再次执行函数,多次执行可以体现函数优势

#!/bin/bash

cecho(){

echo -e "\033[$1m$2\033[0m"

}

cecho 31 ABCDEFG

cecho 32 ABCDEFG

cecho 33 ABCDEFG

cecho 34 ABCDEFG

cecho 35 ABCDEFG

cecho 36 ABCDEFG

break //终止循环,继续执行循环之后的任务

continue //终止本次循环,继续下一次循环

exit //直接退出脚本

编写为用户提供的数字求和脚本,0是结束

#!/bin/bash

x=0

while :

do

read -p "请输入一个数字(0是结束)" n

[ $n -eq 0 ] && break

let x+=n

done

echo "总和是 $x"

编写脚本,找出1~20中谁是6的倍数

#!/bin/bash

for i in {1..20}

do

x=$[i%6] //依次用数字除6取余数

[ $x -ne 0 ] && continue //余数不等于0的话,结束本次循环

echo $i 进入下一次循环

done

字串处理

1,字串截取

${变量名:起始位置:长度}

a=abcdef

echo ${a:0:2} //截取变量a中的字串,从第一位开始截取2位

echo ${a::2} //0可以省略

echo ${a:3:1} //截取第4位字符

获取1位随机字符

n=$[RANDOM%62]

echo ${x:n:1}

如何获取8位?

编写获取8位随机字符的脚本

#!/bin/bash

x=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQR

STUVWXYZ0123456789

for i in {1..8}

do

n=$[RANDOM%62] //求出随机数,范围是0~61之间

txt=${x:n:1} //截取出对应的字符

pass=$pass$txt //累加放在变量pass里面

done

echo $pass //循环结束后,得出结果

2,字串替换

${变量名/old/new}

a=111

echo ${a/1/a} //将1个1替换为a

echo ${a//1/a} //将所有1都替换为a

echo ${a/11/aa} //将11替换为aa

echo ${a//1/} //替换为空,等于删除

3,字串删除

掐头

${变量名#被删除内容} //从左往右删除

a=`head -1 /etc/passwd` //准备素材

echo ${a#root} //从左往右删除到root

echo ${a#x} //从左往右删除到x,无效,因为第一个字符不是x

echo ${a#root:x} //手工输入需要删除的内容

echo ${a#\*x} //使用通配符删除到x以及x左边所有内容,效果同上

echo ${a##\*root} //从左往右删除删除到最后一个root,以及root左边所有内容

echo ${a#\*:root} //从左往右删除到第一个:root,以及左边所有

去尾

echo ${变量名称%被删除的内容}

echo ${a%bash} //去尾,删除到第一个bash

echo ${a%root\*} //删除到第一个root,以及此root右边所有

echo ${a%%root\*} //删除到最后一个root,以及此root右边所有

echo ${a%%:\*} //删除到最后一个: ,以及右边所有

利用字串删除功能编写脚本,用来实现批量修改扩展名

将所有.txt 文件 修改为 .doc

#!/bin/bash

for i in `ls \*.txt` //找到所有.txt文件 依次交给for循环

do

n=${i%.\*} //把找到的每个文件去尾,得到无扩展名的文件名

mv $i $n.doc //修改文件,将名称与.doc组合

done

改良

#!/bin/bash

for i in `ls \*.$1`

do

n=${i%.\*}

mv $i $n.$2

done

定义变量初值(备用值)

aa= //aa等于空

echo ${aa:-123} //定义初值,如果aa是空,则a的内容就是123

非空的话,则显示具体内容

aa=10 //aa非空

echo ${aa:-123} //不会显示123

编写创建用户并配置密码的脚本,如果用户没有输入密码直接回车,则统一使用123,如果用户输入了密码则不使用123

#!/bin/bash

read -p "请输入用户名" n

useradd $n

read -p "请输入密码" p

echo ${p:-123} | passwd --stdin $n

休息

day04

case 变量 in

模式1)

命令序列1;;

模式2)

命令序列2;;

模式3)

命令序列3;;

\*)

命令序列4

esac

netstat -ntulp

函数名称(){

}

break 终止循环,继续执行循环之后的任务

continue 结束本次循环,继续下一次循环

exit 直接退出脚本

字串处理:

${变量名:} 截取

${变量名/} 替换

${变量名#} 掐头

${变量名%} 去尾

${变量名:-} 定义初值

正则表达式,用来查找过滤文档的工具符号

grep "root" user //在user文档中查找有单词root的行

grep "^root" user //在user文档中查找以单词root开头的行

grep "bash$" user //在user文档中查找以单词bash结尾的行

grep "^$" user //查找空行

grep "^ " user //查找以空格开头的行

grep "[rot]" user //查找字母r或者o或者t

grep "roo[tn]" user //查找root或者roon

grep "[^rot]" user //查找字母r或者o或者t之外的内容

grep "[a-z]" user //查找小写字母

grep "[A-Z]" user //查找大写字母

grep "[a-Z]" user //查找所有字母

grep "[0-9]" user //查找数字

grep "r..t" user //查找以r开头t结尾中间有2个任意字符的单词

grep "ro\*" user //查找以r开头后面有或没有字母o的单词,o可以出现任意次

grep "o\{1,2\}" user //查找o出现1次到2次的行

grep "o\{2,3\}" user //查找o出现2次到3次的行

grep "o\{2\}" user //查找o出现2次的行

grep "o\{1,\} " user //查找o出现1次以及1次以上

grep "\(:0\)\{2\}" user //查找:0(数字零)连续出现2次的行

扩展正则

egrep "o+" user //查找o出现1次以及1次以上

egrep "o?" user //查找o出现0次或1次

egrep "o{1,2}" user //查找o出现1次到2次的行

egrep "o{2,3}" user //查找o出现2次到3次的行

egrep "o{2}" user //查找o出现2次的行

egrep "o{1,}" user //查找o出现1次以及1次以上

egrep "(:0){2}" user //查找:0(数字零)连续出现2次的行

egrep "bash|nologin" user //查找bash或者nologin

egrep "\bthe\b" a //查找单词the,前后是空格或者特殊符号

vim 交互式文本编辑工具

sed 流式编辑器,能对文档执行增删改查等操作,逐行操作,非交互式

sed使用方式:

前置命令 | sed 选项 (定址符)指令

sed 选项 (定址符)指令 文档

选项

-n 屏蔽默认输出

-i 写入到文件

-r 使用扩展正则

指令

p 输出文档内容

d 删除

s 替换

sed -n 'p' user //使用sed输出user文档中所有行

sed -n '1p' user //使用sed输出user文档中第1行

sed -n '1,3p' user //使用sed输出user文档中第1行到第3行

sed -n '1,+3p' user //输出user文档中第1行以及后面的3行

sed -n '1p;4p' user //输出第1行和第4行

sed -n '1~2p' user //输出奇数行

sed -n '2~2p' user //输出偶数行

使用正则做定址符

sed -n '/root/p' /etc/passwd //指令前面添加2个斜杠,中间使用正则,查找有root单词的行

sed -n '/bash$/p' /etc/passwd

sed -rn '/bash|nologin/p' user //竖线"|" 是扩展正则,要加-r选项

sed -n '=' user //显示行号

sed -n '$=' user //显示最后一行的行号

sed 'd' user //使用sed删除user文档中所有行

sed '1d' user //使用sed删除user文档中第1行

sed '1,3d' user //使用sed删除user文档中第1行到第3行

sed '1,+3d' user //使用sed删除user文档中第1行以及后面3行

替换:

s/旧内容/新内容/ 或者 s$旧内容$新内容$ 等符号

sed 's/2017/AAAA/' test.txt //替换所有行的第1个2017为AAAA

sed 's/2017/AAAA/g' test.txt //替换所有行的所有2017为AAAA

sed 's/2017/AAAA/2' test.txt //替换所有行的第2个2017为AAAA

sed '1s/2017/AAAA/' test.txt //替换第1行的第1个2017为AAAA

sed '2s/2017/AAAA/2' test.txt //替换第2行的第2个2017为AAAA

sed '3s/2017/AAAA/3;3s/2017/AAAA/2;' test.txt //替换第3行的第3个, 然后在替换第3行的第2个

sed '3s/2017/AAAA/2;3s/2017/AAAA/2;' test.txt //效果同上

如何将 /bin/bash 替换为 /sbin/sh

sed 's#/bin/bash#/sbin/sh#' user

\(\)

()

保留=复制 \1 \2 \3

sed 's/.//2 ; s/.$//' nssw.txt //利用替换功能,删除文档中每行的第2个字符和最后一个字符

sed -r 's/^(.)(.\*)(.)$/\3\2\1/' nssw.txt //将每行的首个字符和最后一个字符互换,首先利用^(.)复制行首字符,定义为\1 , 然后复制行尾字符 (.)$ ,定义为\3 ,中间的字符无论多少都使用(.\*)涵盖,定义为\2, 然后根据需求粘贴.

sed -r 's/^(.)(.)(.)(.)$/\4\3\2\1/' aa //aa文档中有abcd,将abcd 换成 dcba

sed 's/[0-9]//g' nssw.txt //删除所有数字

sed -r 's/([A-Z])/[\1]/g' nssw.txt //将所有大写字母添加中括号

编写一键部署ftp服务的脚本,要求ftp具备匿名上传功能

vim /etc/vsftpd/vsftpd.conf

#anon\_upload\_enable=YES

#anon\_mkdir\_write\_enable=YES

sed 's/^#anon\_/anon\_/'

#!/bin/bash

yum -y install vsftpd &> /dev/null

sed -i 's/^#anon\_/anon\_/' /etc/vsftpd/vsftpd.conf

systemctl restart vsftpd

systemctl enable vsftpd

systemctl stop firewalld

setenforce 0

chmod o+w /var/ftp/pub

虚拟机执行脚本后,使用真机访问虚拟机的ftp,可以在pub目录中成功上传文件即可,关闭防火墙为了测试此脚本,生产环境中不可关闭

day05

前置指令 | sed 选项 (定址符)指令

sed 选项 (定址符)指令 文件

-n -r -i

/ /p 2d 3s

a行后追加 i行前插入 c替换整行

sed '1a AAAA' user //在第1行的后面追加AAAA

sed '/^bin/a AAAA' user //在以bin开头的行的后面追加AAAA

sed '1i AAAA' user //在第1行的前面追加AAAA

sed '/^bin/i AAAA' user //在以bin开头的行的前面追加AAAA

sed '1c AAAA' user //把第1行替换成AAAA

sed '/^bin/c AAAA' user //把bin开头的行替换成AAAA

sed '$a 1.1.1.1 www.baidu.com' /etc/hosts //在hosts文件的最后一行后面追加ip地址与域名

sed 修改,查找 文件

awk 查找输出文件的内容

awk使用方式:

1, 前置命令 | awk 选项 条件{指令}

2, awk 选项 条件{指令} 文件

指令 : print $1 $2 $3

选项 : -F 分隔符

awk '{print $1}' nssa //在nssa文件中查找所有行的第1列并输出

awk '{print}' nssa //输出文档所有内容

awk -F: '{print $6}' /etc/passwd //以冒号做分隔符,输出第6列,相当于显示所有用户的家目录

提取网卡流量:

ifconfig eth0 | sed -rn '/RX p|TX p/p' | awk -F[\(\)] '{print $2}' //首先使用前置命令查看eth0网卡所有信息, 然后使用sed过滤显示有RX p(接收的流量)和TX p(发送的流量)的行, 最后使用awk 定义小括号为分隔符 ,并输出第2列

ifconfig eth0 | awk -F[\(\)] '/RX p|TX p/{print $2}' //awk也支持正则, 效果同上

awk支持常量输出,放在双引号中间即可

df -h | awk '/vda1/{print "当前主机根分区剩余空间是"$4}' //配合常量输出

awk '/Failed/{print "登录本机失败的IP地址",$11}' /var/log/secure //在安全日志文件中利用awk筛选出访问本机失败的ip,可加逗号做空格

awk的内置变量:

$0显示所有 $1 $2 $3... NR显示行号 NF显示列号

awk处理的时机: 相当于逐行任务之外的任务

BEGIN{print } //在逐行任务之前添加任务,执行1次

{print} //逐行任务,执行n次

END{print } //在逐行任务之后添加任务,执行1次

awk 'BEGIN{print "早上好"}{print}END{print "再见"}' nssw

awk '{print}END{print NR}' nssw //显示所有内容,以及最后一行的行号

awk -F: 'BEGIN{print "User\tUid\tHome"}{print $1"\t"$3"\t"$6}END{print "总计" NR "行"}' user //使用awk处理时机,输出表头,内容,和结尾

awk 'BEGIN{x=0}{x++}END{print x}' user

awk处理条件:

1,使用正则

awk -F: '/root/{print}' /etc/passwd //输入有root的行

awk -F: '$1~/root/{print}' /etc/passwd //输入第一列包含root的行

awk -F: '$1!~/root/{print}' /etc/passwd //输入第一列不包含root的行

2,使用字符或数字做条件,可以配合符号 == != > < >= <=

awk -F: '$1=="root"{print}' user //找第1列是root的行

awk -F: 'NR==1{print}' user //找第一行

awk -F: '$3>=1000{print $1,$3}' /etc/passwd //找第3列大于等于1000的行,并输出第1列与第3列

awk -F: '$3<10{print $1,$3}' /etc/passwd //找第3列小于10的行,并输出第1列与第3列

awk -F: '$3==10{print $1,$3}' /etc/passwd //找第3列等于10的行,找不到则无任何显示

3,使用逻辑符号 && 且 || 或

awk -F: '$3>10 && $3<20{print}' /etc/passwd //找第3列大于10并且第3列小于20的行, {print}可以省略不写

awk -F: '$1=="root" ||$1=="bin"{print}' /etc/passwd //找第1列是root或是bin的行

awk -F: '$1=="rootabc" ||$1=="bin"{print}' /etc/passwd //找第1列是rootabc或是bin的行

awk -F: '$1=="root" && $1=="bin"{print}' /etc/passwd //找第1列既是root又是bin的行,逻辑错误,找不到

awk -F: '$3>100 && $3<10{print}' /etc/passwd //逻辑错误找不到

awk -F: '$3>10 || $3<1000print}' /etc/passwd //找所有行

4,运算

awk 'BEGIN{print 1+1}'

awk 'BEGIN{print 1.1+1.1}'

awk 'BEGIN{print 2\*2}'

awk 'BEGIN{print 2/2}'

awk 'BEGIN{print 2%2}'

awk 'BEGIN{x=2;y=5;print x+y}'

列出UID间于1~1000的用户详细信息

输出/etc/hosts映射文件内以127或者192开头的记录

列出100以内整数中7的倍数或是含7的数

seq 100 | awk '$1%7==0 || $1~/7/'

方法一:

#/bin/bash

A=$(awk -F: '/bash$/{print $1}' /etc/passwd)

for i in $A

do

grep $i /etc/shadow | awk -F: '{print $1,"-->",$2}'

done

方法二:

#!/bin/bash

u=`awk -F: '/bash$/{print $1}' /etc/passwd`

for i in $u

do

awk -F: -v a=$i '$1==a{print $1,"-->" ,$2}' /etc/shadow

done

day06

awk 选项 条件 指令

-F -v 正则,字符和数字,逻辑符号&& ||,运算 {print $1 $2}

变量只可以存储一个值,再次赋值时,上一个值会被覆盖

a=10

a=20

数组可以利用多个下标,仅仅使用一个数组名存储多个值

数组名[下标]=值

a[0]=10

a[1]=20

awk 'BEGIN{a[0]=10;a[1]=20;print a[1],a[0]}' //首先定义1个数组,名称a,再定义多个下标对应的值, 然后输出

awk 'BEGIN{a[0]="a";a[1]="b";print a[1],a[0]}' //除了数字,还可以使用字符,注意要使用双引号

awk 'BEGIN{a["abc"]=10;a["xyz"]=20;print a["abc"],a["xyz"]}'

awk 'BEGIN{a[0]++;a[1]++;a[1]++;print a[0],a[1]}'

awk 'BEGIN{a[0]=0;a[1]=11;a[2]=22; for(i in a){print i,a[i]}}' //使用awk数组+for循环显示数组的所有内容

首先准备素材:

abc

abc

xyz

qqq

xyz

abc

awk '{cba[$1]++}END{for(i in cba){print cba[i],i}}' abc

//使用数组+for循环, 输出该文档中每行的第1列是什么内容,出现了几次. 首先定义数组名为cba,下标是每行的第1列, 然后执行++运算,逐行执行,都执行完后得到结果:

cba[abc]=3

cba[xyz]=2

cba[qqq]=1

然后这个结果再交给for循环处理,循环显示的变量i是下标.

ip[192.168.0.2]=3

ip[192.168.0.1]=2

ip[172.25.0.1]=1

ip[8.8.8.8]=1

awk '{ip[$1]++}END{for(i in ip){print i,ip[i]}}' abc | sort -rk 2 //将最后处理的结果交给sort排序, -k是定义按照哪列排序, -r是降序排列

netstat -ntulp | grep :80

使用awk统计网站访问者的ip与点击量

1,安装httpd与测试工具

yum -y install httpd elinks

2,开启httpd服务

3,使用测试工具访问本机网站

elinks -dump 127.0.0.1

4,查看httpd访问量日志

vim /var/log/httpd/access\_log

再使用其他主机访问此http服务(注意关闭防火墙),再次查看日志发现增加了新记录

5,使用awk查看最终结果

awk '{ip[$1]++}END{for(i in ip){print ip[i],i}}' /var/log/httpd/access\_log | sort -r

6,可以配合脚本增加访问次数再次检测

#!/bin/bash

for i in {1..5000}

do

elinks -dump 172.25.0.10 &> /dev/null

done

编写监控脚本,可以方便查看主机的若干参数

#!/bin/bash

while :

do

uptime |awk '{print "cpu平均负载:"$10,$11,$12}'

ifconfig eth0 |awk -F[\(\)] '/RX p/{print "网卡eth0接收的数据量是:"$2}'

ifconfig eth0 |awk -F[\(\)] '/TX p/{print "网卡eth0发送的数据量是:"$2}'

free -m | awk '/^Mem:/{print "主机剩余内存容量是:"$4"m"}'

df -h | awk '/vda1/{print "磁盘剩余空间是:"$4}'

u=`cat /etc/passwd | wc -l`

echo "计算机账户数量是:$u"

user=`who | wc -l`

echo "当前登录的用户数量是:$user"

p=`ps -aux | wc -l`

echo "目前开启的进程数量是:$p"

r=`rpm -qa | wc -l`

echo "目前安装的软件包数量是:$r"

sleep 2

clear

done

使用awk找出安全日志中对本机访问失败的ip记录与次数统计

awk '/Failed/{ip[$11]++}END{for(i in ip){print ip[i],i}}' /var/log/secure

#!/bin/bash

abc=`awk '/Failed/{ip[$11]++}END{for(i in ip){print ip[i]","i}}' /var/log/secure`

for i in $abc

do

aa=${i%,\*}

bb=${i#\*,}

[ $aa -ge 10 ] && echo "$bb猜错10次以上密码了,赶紧收拾他" | mail -s test root

done

\w 匹配数字字母下划线 a-z A-Z 0-9

\W 取反

grep "\w" aaaa //在aaaa文件中找所有的数字字母下划线

\s 各种空白,如空格,tab制表符等

\S 取反

day01

网络功能：

资源共享

信息传递

提高可靠性 云盘

增加系统处理能力 web

分组交换

tcp/ip

web技术

iso

ieee

点到点 广域网

星形 易于实现 方便故障排查

网状 提高可靠性

osi七层模型 （理论框架）

tcp/ip五层模型 （实际应用）

10000Mbit

100000000bit

1k=1000

1M=1000k

物理层 数据链路层 网络层 传输层 会话层 表示层 应用层

http https dns ftp tftp smtp snmp

tcp udp

icmp igmp arp ip

物理层 数据链路层 网络层 传输层 应用层

比特流 数据帧 数据包 数据段

网卡 交换机 路由器 防火墙 计算机

100M cat5

100M cat5e 超五类

1000M cat6

10000M cat7

UTP 非屏蔽双绞线 STP 屏蔽双绞线

568A

568B

直通线

交叉线

全反线

准备win2008环境，密码Taren1

用户模式

Switch>

特权模式（一般用于查看配置信息）

Switch>enable

Switch#

全局配置模式（所做的配置对整个设备生效）

Switch#configure terminal

Switch(config)#

接口模式

Switch(config)#interface fastEthernet 0/1

Switch(config-if)#

exit 返回上一模式

end或快捷键 crtl+z 返回特权模式

思科设备命令行不区分大小写，与Linux命令行不同

命令输入错误卡住时同时按键盘ctrl+shift+6这三个键即可

远程控制终端软件：

SecureCRT

Xshell

no

Switch(config)# hostname s1 //配置主机名

s1# show running-config //查看当前的运行配置

Switch(config)#enable password 123 //配置密码为123，之后退出到用户模式再次进入特权模式测试密码

Switch#copy running-config startup-config //保存配置

Switch#write //保存配置，同上效果

Switch#erase startup-config //删除配置文件，恢复出厂设置

Erasing the nvram filesystem will remove all configuration files! Continue? [confirm] //此时系统询问是否确定删除，按回车即可

[OK]

Erase of nvram: complete

Switch#reload //重启设备

switch(config)#no ip domain-lookup //禁止dns查询，不会因为错误的命令卡住

Switch(config)#line console 0

Switch(config-line)#logging synchronous //实现输入同步，不被系统信息干扰

Switch(config)#line console 0

Switch(config-line)#exec-timeout 10 0 //修改10分钟后锁屏，设置0 0 是永不超时

day02

资源共享

信息传递

增加可靠

提高系统处理能力

osi 7层

tcp/ip 5层

点对点 广域网

星形 好实现 方便故障排查 易于扩扩展

网状 提高可靠

wan 广域网 lan 局域网

物理层 数据链路层 网络层 传输层 应用层

rj-45

双绞线

568A

568B

cat5

cat5e 100M

cat6

cat7

ip 32位 十进制 跨网络的数据传递 192.168.0.1

mac 48位 十六进制 同局域网范围内的数据传递

全球唯一

前24位 厂商标示

后24位 产品的唯一编号

fe54：00:00:00:0b

单播 1对1

组播 1对多

广播 1对所有

MAC 物理地址 硬件地址

8 bit 比特 = 1 byte 字节

100Mb 12+MB

mtu 最大传输单元

交换机查看MAC地址表

Tarena-sw1#show mac-address-table

windows中查看网络信息，比如mac地址使用 ipconfig /all

学习 源MAC地址

广播

转发

更新 300秒

vlan数量 默认 允许创建 4096个 0～4095

1000

vlan1是默认存在的

Switch(config)#vlan 2 //在全局配置模式下创建VLAN2

Switch(config)#no vlan 2 //删除VLAN2

Switch(config)# interface f0/1 //进入f0/1接口

Switch(config-if)# switchport access vlan 2 //将接口加入vlan2

Switch(config-if)# no switchport access vlan 2 //取消加入vlan2

将多个接口添加到某个VLAN中：

Switch(config)# interface range f0/1–10 //进入组接口模式

Switch# show vlan brief //查看vlan摘要信息

提前熟悉路由器基本操作

使用2911路由器可以测试以下配置：

模式的切换（注意，路由器接口是千兆的 gigabitethernet 0/1），配密码，改主机名，保存，重启，清空配置等。。。

思考：

如何使同vlan设备互通

trunk 中继链路，可以实现不通vlan的数据在一条链路中传递

Switch(config-if)#switchport mode trunk //配置接口为trunk模式

access 接入链路 可以承载1个vlan

trunk 中继链路 可以承载多个vlan

以太通道，提高带宽，增加可靠性

Switch(config)# interface range fastEthernet 0/1 – 2 //同时进入1口与2口

Switch(config-if-range)#channel-group 1 mode on //捆绑为以太通道

Switch(config-if)#shutdown //关闭接口

Switch(config-if)#no shutdown //开启接口

Switch# show etherchannel summary //查看以太通道汇总信息

day02 2

广播泛滥

vlan 虚拟局域网

1.控制广播

2.提高安全

3.提高带宽利用

4.降低数据传输延迟

理论最大创4096个 (0~4095) 厂商基本1000+ 1 开始

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1个广播域=1个部门=1个vlan=1个网段

]vlan 2 //创建vlan 2

vlan配置: (系统视图)

port link-type access //设置为access模式

port default vlan 2 //设置为vlan 2

port link-type trunk //设置为trunk模式

port trunk allow-pass vlan all //允许所有vlan通过

配置思路:

1.规划vlan,需要创建多少

2.将对应的接口加入vlan

3.交换机之间的链路配置为trunk

4.最后配置好ip做ping测试

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

单点故障

链路聚合

链路聚合,提高链路可靠,提高链路带宽

注意:配置之前检查接口是否为默认状态,否则要清空配置

clean configueation interface ethernet 0/0/7

in e0/0/7

undo shutdown

1.找到要捆绑的接口,至少两个

2.创建链路聚合接口

interface Eth-trunk 1 //创建1号链路聚合接口

3.将物理接口加入链路聚合接口

trunkport ethernet 0/0/7 0/0/8 //将7 8 号绑在一起

port link-type trunk //继续将1号链路聚合口配置为中继链路

port trunk allow-pass vlan all //放行所有vlan

undo terminal monitor //关闭某些信息

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

路由表指明当前设备可以前往的区域(网段)

display ip routing-table //查看路由表

display ip interface brief //查看配置的ip

直连路由:在设备接口配置好IP地址并且打开后自动产生

静态路由:由管理员手工设置,小规模网络使用

ip route-static 192.168.3.0 24 192.168.2.2 //配置静态IP

day03

MAC 48位 6字节 前24位厂商标示 后24位产品唯一编号

MTU 最大传输单元 1500

学习 广播 转发 更新 超300秒，链路down

vlan 广播控制 增加安全 提高带宽利用 降低数据延迟

trunk 打标记

以太通道 增加带宽，提高可靠

192.168.1.1/24

网络位 主机位

255.255.255.0

网络ID

192.168.1.0

广播地址

192.168.1.255

192.168.1.2/24

A 1～127

255.0.0.0

B 128～191

255.255.0.0

C 192～223

255.255.255.0

192.168.1.1/16

192.168.10.1/16

172.16.0.1

172.16.0.0

172.16.10.1

172.16.0.0

Router(config)#interface gigabitEthernet 0/0 //进入千兆接口

Router(config-if)#ip address 192.168.1.254 255.255.255.0 //配置ip地址

Router(config-if)#no shutdown //开启接口

Router#show ip route //查看路由表

C 直连路由，接口配置好ip，并开启后自动生成

S 静态路由 ip route

S\* 默认路由 ip route 0.0.0.0 0.0.0.0 下一跳

动态路由

只允许配置对外方向，否则容易出现路由环路

路由器1配置的静态路由

Router(config)#ip route 192.168.3.0 255.255.255.0 192.168.2.2

路由器2配置的静态路由

Router(config)#ip route 192.168.1.0 255.255.255.0 192.168.2.1

静态路由如果配置错误要及时删除

Router(config)#no ip route 192.168.1.0 255.255.255.0 192.168.2.1

配置思路：

1，检查路由表 （show ip route），找到无法到达的网段

2，使用静态路由，添加网段

三层交换机=二层转发+三层路由

在三层交换机启用路由功能

Switch(config)# ip routing

配置虚接口的IP

Switch(config)# interface vlan 1

Switch(config-if)# ip address 192.168.1.254 255.255.255.0

Switch(config-if)# no shutdown

在三层交换机上配置Trunk前要先指定接口封装为802.1q

Switch(config)#interface fastEthernet 0/24

Switch(config-if)#switchport trunk encapsulation dot1q

Switch(config-if)#switchport mode trunk

排错思路：

1，检查ip地址与网关

2，检查接口是否加入相应的vlan

3，网关设备（三层交换）的vlan虚接口ip是否配置正确并开启

4，交换机与交换机之间的链路是否配置为trunk

5，三层交换的路由功能是否开启

show ip interface brief //查看接口摘要信息，是否配置ip，是否开启

三层交换机配置默认路由前往外网：

Switch(config)#ip route 0.0.0.0 0.0.0.0 192.168.4.2

day04

宣告 向动态路由邻居通告自身所连接的网段

准备使用动态路由前将路由器的静态路由删除

三层交换机的默认路由保留

首先在三层交换机配置：

Switch(config)# router ospf 1 //启动OSPF路由进程

指定OSPF协议运行的接口和所在的区域

Switch(config-router)# network 192.168.1.0 0.0.0.255 area 0

Switch(config-router)# network 192.168.2.0 0.0.0.255 area 0

Switch(config-router)# network 192.168.3.0 0.0.0.255 area 0

Switch(config-router)# network 192.168.4.0 0.0.0.255 area 0

然后在路由器中配置

Route(config)# router ospf 1 //启动OSPF路由进程

指定OSPF协议运行的接口和所在的区域

Route(config-router)# network 192.168.4.0 0.0.0.255 area 0

排错，分别检查三个方面：

vlan

ip

路由

传输层实现了端到端的数据传递

0～65535 端口总数

1～1023 常用端口

SYN 想与对方建立连接

ACK 确认

FIN 想与对方断开连接

TCP 可靠 ，面向连接，效率低

UDP 不可靠，无连接，效率高

访问控制列表（ACL）

读取第三层、第四层 头部信息

根据预先定义好的规则对数据进行过滤

对已经建设好的网络，可以通过ACL管理，使网络环境更加健康

标准ACL

基于源IP地址过滤数据包

标准访问控制列表的访问控制列表号是1～99

扩展ACL

基于源IP地址、目的IP地址、指定协议、端口来过滤数据包

扩展访问控制列表的访问控制列表号是100～199

192.168.2.1

255.255.255.0

0.0.0.255

0.0.0.0

0严格匹配

1不匹配

1题 禁止192.168.2.1与192.168.1.1通信，不影响其他数据

Router(config)#access-list 1 deny 192.168.2.1 0.0.0.0 //拒绝192.168.2.1通过

Router(config)#access-list 1 permit any //允许其他所有人通过

Router(config)# in g0/1

Router(config-if)# ip access-group 1 in //在接口应用acl

2题 只允许192.168.2.1与192.168.1.1通信

Router(config)#access-list 2 permit 192.168.2.1 0.0.0.0 //允许192.168.2.1通过

Router(config)# in g0/1

Router(config-if)# ip access-group 2 in //在接口应用acl

使用扩展acl限制192.168.2.1访问192.168.1.1的ftp服务

Router(config)#access-list 100 deny tcp 192.168.2.1 0.0.0.0 192.168.1.1 0.0.0.0 eq 21

Router(config)# in g0/1

Router(config-if)# ip access-group 100 in //在接口应用acl

使用扩展acl限制192.168.2.2访问192.168.1.1的www服务

Router(config)#access-list 100 deny tcp 192.168.2.2 0.0.0.0 192.168.1.1 0.0.0.0 eq 80

Router(config)#access-list 100 permit ip any any //放行其他所有数据

A

B

C

IPV4 32位 42亿

100.0.0.1 20M 3万

200.0.0.1

NAT 网络地址转换，可以将内部私有ip转换为外部公有ip

私有ip地址范围

A 10.0.0.0～10.255.255.255

B 172.16.0.0～172.31.255.255

C 192.168.0.0～192.168.255.255

IPV6 128位

2，配置静态nat转换

Router(config)#ip nat inside source static 192.168.1.1 100.0.0.2

Router(config)#ip nat inside source static 192.168.1.2 100.0.0.3

Router(config)#interface g0/1

Router(config-if)#ip nat outside

Router(config)#interface g0/0

Router(config-if)#ip nat inside

静态 一对一 服务器发布服务时才使用

PAT 一对多 端口多路复用，通常只用来访问外网

使用PAT技术，让内网所有主机利用外网接口的公网地址出门

首先使用acl定义内部ip地址

Router(config)#access-list 1 permit any

使用pat复用外网接口地址

Router(config)#ip nat inside source list 1 interface g0/1 overload

day05

传输层

TCP 面向连接

三次握手

SYN----ACK，SYN----ACK

四次断开

FIN----ACK----FIN----ACK

HTTP 80

DNS 53

SSH 22

FTP 21

SMTP 25

HTTPS 443

UDP 不可靠 无连接 效率高

DNS 53

TFTP 69

NTP 123

ACL 访问控制列表

标准 针对源ip限制 1～99

扩展 源ip ，目标ip，协议，端口 100～199

NAT 网络地址转换，私有ip转换为公有ip

A 10.0.0.0～10.255.255.255

B 172.16.0.0～172.31.255.255

C 192.168.0.0～192.168.255.255

静态 一对一

PAT 一对多

STP 生成树协议，通过临时断开某接口避免广播风暴，当其他链路断开后，此接口会自动恢复，起到备份链路的效果

决定断开接口的是主根网桥与次根网桥

通过修改优先级可以实现主根与次根的配置

优先级默认 32768，越小越优先，如果要修改，必须为4096的倍数

1，在Switch1中配置

Switch(config)#spanning-tree vlan 1 priority 24576

Switch#show spanning-tree //查看所有vlan的生成树信息

Switch#show spanning-tree vlan 1 //查看vlan1的生成树信息

阻塞接口 BLK

转发接口 FWD

Switch(config)#spanning-tree vlan 1 root primary //将当前设备配置为主根

Switch(config)#spanning-tree vlan 1 root secondary //将当前设备配置为次根

vlan1 vlan2

MS1 vlan1的主根 vlan2的次根

MS2 vlan1的次根 vlan2的主根

HSRP 热备份路由选择协议

主要功能：网关备份

三层交换机物理接口配置ip首先转换为路由器接口

no switchport 再开启路由功能 ip routing

in vlan 1 //进入vlan1 配置ip

路由器

outer(config)#router ospf 1

Router(config-router)#network 192.168.4.0 0.0.0.255 area 0

Router(config-router)#network 192.168.3.0 0.0.0.255 area 0

Router(config-router)#network 192.168.2.0 0.0.0.255 area 0

MS1

Switch(config)#ip routing

Switch(config)#router ospf 1

Switch(config-router)#network 192.168.1.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.2.0 0.0.0.255 area 0

MS2

Switch(config)#ip routing

Switch(config)#router ospf 1

Switch(config-router)#network 192.168.1.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.3.0 0.0.0.255 area 0

三层交换机开启HSRP功能：

Switch(config)#interface vlan 1

Switch(config-if)#standby 1 ip 192.168.1.254

Switch(config-if)#standby 1 priority 105

Switch(config-if)#standby 1 preempt

综合拓扑：

1，为每台2960交换机分别添加10、20、30、40这四个vlan

2，将每台2960交换机的f0/5口分别加入vlan10、20、30、40

注意：最后一台要加5口与6口到vlan40中

SW1#conf t

SW1(config)#in f0/5

SW1(config-if)#switchport access vlan 10 //第一台加vlan10

3，为三层交换机分别添加10、20、30、40这四个vlan

4，将所有交换机与交换机之间的链路配置为trunk

Switch(config)#interface range fastEthernet 0/1-5

Switch(config-if)#switchport trunk encapsulation dot1q

Switch(config-if)#switchport mode trunk

5，通过配置生成树协议，实现负载均衡

MS1 vlan10、20 主根 vlan30、40 次根 stp

vlan10、20 活跃 vlan30、40 备份 hsrp

MS2 vlan10、20 次根 vlan30、40 主根 stp

vlan10、20 备份 vlan30、40 活跃 hsrp

第一台三层交换机配置：

Switch(config)#spanning-tree vlan 10 root primary //将当前设备配置为vlan10的主根

Switch(config)#spanning-tree vlan 20 root primary //将当前设备配置为vlan20的主根

Switch(config)#spanning-tree vlan 30 root secondary //将当前设备配置为vlan30的次根

Switch(config)#spanning-tree vlan 40 root secondary //将当前设备配置为vlan40的次根

第二台三层交换机配置：

Switch(config)#spanning-tree vlan 40 root primary //将当前设备配置为vlan40的主根

Switch(config)#spanning-tree vlan 30 root primary //将当前设备配置为vlan30的主根

Switch(config)#spanning-tree vlan 20 root secondary //将当前设备配置为vlan20的次根

Switch(config)#spanning-tree vlan 10 root secondary //将当前设备配置为vlan10的次根

HSRP配置

MS1：

Switch(config)#interface vlan 10

Switch(config-if)#ip add 192.168.10.252 255.255.255.0

Switch(config-if)#standby 10 ip 192.168.10.254

Switch(config-if)#standby 10 priority 105

Switch(config-if)#standby 10 preempt

Switch(config)#interface vlan 20

Switch(config-if)#ip add 192.168.20.252 255.255.255.0

Switch(config-if)#standby 20 ip 192.168.20.254

Switch(config-if)#standby 20 priority 105

Switch(config-if)#standby 20 preempt

Switch(config)#interface vlan 30

Switch(config-if)#ip add 192.168.30.252 255.255.255.0

Switch(config-if)#standby 30 ip 192.168.30.254

Switch(config)#interface vlan 40

Switch(config-if)#ip add 192.168.40.252 255.255.255.0

Switch(config-if)#standby 40 ip 192.168.40.254

MS2：

Switch(config)#interface vlan 40

Switch(config-if)#ip add 192.168.40.253 255.255.255.0

Switch(config-if)#standby 40 ip 192.168.40.254

Switch(config-if)#standby 40 priority 105

Switch(config-if)#standby 40 preempt

Switch(config)#interface vlan 30

Switch(config-if)#ip add 192.168.30.253 255.255.255.0

Switch(config-if)#standby 30 ip 192.168.30.254

Switch(config-if)#standby 30 priority 105

Switch(config-if)#standby 30 preempt

Switch(config)#interface vlan 20

Switch(config-if)#ip add 192.168.20.253 255.255.255.0

Switch(config-if)#standby 20 ip 192.168.20.254

Switch(config)#interface vlan 10

Switch(config-if)#ip add 192.168.10.253 255.255.255.0

Switch(config-if)#standby 10 ip 192.168.10.254

6，开启所有三层交换机的路由功能，再给服务器配置ip与网关，实现全网互通

7，在ms1中开启ospf动态路由，并宣告直连网段

Switch(config)#router ospf 1

Switch(config-router)#network 192.168.10.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.20.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.30.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.40.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.50.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.60.0 0.0.0.255 area 0

在ms2中开启ospf动态路由，并宣告直连网段

Switch(config)#router ospf 1

Switch(config-router)#network 192.168.10.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.20.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.30.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.40.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.70.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.80.0 0.0.0.255 area 0

在r1中开启ospf动态路由，并宣告直连网段

Router(config)#router ospf 1

Router(config-router)#network 192.168.50.0 0.0.0.255 area 0

Router(config-router)#network 192.168.70.0 0.0.0.255 area 0

在r2中开启ospf动态路由，并宣告直连网段

Router(config)#router ospf 1

Router(config-router)#network 192.168.60.0 0.0.0.255 area 0

Router(config-router)#network 192.168.80.0 0.0.0.255 area 0

8，在两台路由器中设置默认路由并在ospf中向其他设备传递该信息

Router(config)#ip route 0.0.0.0 0.0.0.0 100.0.0.10

Router(config)#router ospf 1

Router(config-router)#default-information originate

9，在两台路由器中设置nat，使内网192.168.40.1设备可以访问外网

Router(config)#ip nat inside source static 192.168.40.1 100.0.0.3

Router(config)#in g0/2

Router(config-if)#ip nat outside

Router(config-if)#in range g0/0-1

Router(config-if-range)#ip nat inside

# CLUSTER全部笔记

调度器是集群系统的核心技术

目的:

提高性能

降低成本

提高扩展性

增加可靠性

集群分类:

高性能集群(HPC)

负载均衡集群(LB)

高可用集群(HA)

LVS集群组成:

前端:负载均衡层

中间层:服务器群组层

底端(后端):数据共享存储层

LVS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 --真正提供服务的服务器

VIP:虚拟IP地址 ---用户访问的虚拟IP

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

CIP:客户端IP地址

web1(RIP)

proxy

(VIP) (DIP) web2(RIP)

LVS原理就是路由+调度

负载均衡调度十种算法:

常用六种:

轮询 (rr)

加权轮询(wrr)

最少连接(lc)

加权最少链接(wlc)

源地址哈希(source hash == sh)

目标地址哈希(dh)

################################################################

部署LVS-NAT集群

集群对外公网IP地址为192.168.4.5

调度器内网IP地址为192.168.2.5

真实Web服务器地址分别为192.168.2.100、192.168.2.200

使用加权轮询调度算法，真实服务器权重分别为1和2

步骤一：配置基础环境

1）设置Web服务器（以web1为例）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html

2）启动Web服务器软件

[root@web1 ~]# systemctl restart httpd

3)关闭防火墙与SELinux

[root@web1 ~]# systmctl stop firewalld

[root@web1 ~]# setenforce 0

步骤三：部署LVS-NAT模式调度器

1)确认调度器的路由转发功能(如果已经开启，可以忽略)

[root@proxy ~]# echo 1 > /proc/sys/net/ipv4/ip\_forward

[root@proxy ~]# cat /proc/sys/net/ipv4/ip\_forward

1

[root@proxy ~]# echo "net.ipv4.ip\_forward = 1" >> /etc/sysctl.conf

#修改配置文件，设置永久规则

2）创建集群服务器

[root@proxy ~]# yum -y install ipvsadm

[root@proxy ~]# ipvsadm -A -t 192.168.4.5:80 -s wrr

2）添加真实服务器

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100 -w 1 -m

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200 -w 1 -m

3）查看规则列表，并保存规则

[root@proxy ~]# ipvsadm -Ln

[root@proxy ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

步骤四：客户端测试

客户端使用curl命令反复连接http://192.168.4.5，查看访问的页面是否会轮询到不同的后端真实服务器。

########################################################################

部署LVS-DR集群

CIP是客户端的IP地址；

VIP是对客户端提供服务的IP地址；

RIP是后端服务器的真实IP地址；

DIP是调度器与后端服务器通信的IP地址（VIP必须配置在虚拟接口）。

步骤一：配置实验网络环境

1）设置Proxy代理服务器的VIP和DIP

注意：为了防止冲突，VIP必须要配置在网卡的虚拟接口！！！

[root@proxy ~]# cd /etc/sysconfig/network-scripts/

[root@proxy ~]# cp ifcfg-eth0{,:0}

[root@proxy ~]# vim ifcfg-eth0

TYPE=Ethernet

BOOTPROTO=none

NAME=eth0

DEVICE=eth0

ONBOOT=yes

IPADDR=192.168.4.5

PREFIX=24

[root@proxy ~]# vim ifcfg-eth0:0

TYPE=Ethernet

BOOTPROTO=none

DEFROUTE=yes

NAME=eth0:0

DEVICE=eth0:0

ONBOOT=yes

IPADDR=192.168.4.15

PREFIX=24

[root@proxy ~]# systemctl restart network

//启动失败需要关闭 NetworkManager (systemctl stop NetworkManager)

2）设置Web1服务器网络参数

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

接下来给web1配置VIP地址。

注意：这里的子网掩码必须是32（也就是全255），网络地址与IP地址一样，广播地址与IP地址也一样。

[root@web1 ~]# cd /etc/sysconfig/network-scripts/

[root@web1 ~]# cp ifcfg-lo{,:0}

[root@web1 ~]# vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

防止地址冲突的问题：

这里因为web1也配置与代理一样的VIP地址，默认肯定会出现地址冲突；

sysctl.conf文件写入这下面四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应，这样防止地址冲突的问题。

[root@web1 ~]# vim /etc/sysctl.conf

#手动写入如下4行内容

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

net.ipv4.conf.all.arp\_announce = 2

#当有arp广播问谁是192.168.4.15时，本机忽略该ARP广播，不做任何回应

#本机不要向外宣告自己的lo回环地址是192.168.4.15

[root@web1 ~]# sysctl -p

重启网络服务，设置防火墙与SELinux

[root@web1 ~]# systemctl restart network

//启动失败需要关闭 NetworkManager (systemctl stop NetworkManager)

[root@web1 ~]# ifconfig

[root@web1 ~]# systemctl stop firewalld

[root@web1 ~]# setenforce 0

步骤二：配置后端Web服务器

1）自定义Web页面

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.4.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.4.200" > /var/www/html/index.html

2）启动Web服务器软件

[root@web1 ~]# systemctl restart httpd

[root@web2 ~]# systemctl restart httpd

步骤三：proxy调度器安装软件并部署LVS-DR模式调度器

1）安装软件（如果已经安装，此步骤可以忽略）

[root@proxy ~]# yum -y install ipvsadm

2）清理之前实验的规则，创建新的集群服务器规则

[root@proxy ~]# ipvsadm -C #清空所有规则

[root@proxy ~]# ipvsadm -A -t 192.168.4.15:80 -s wrr

3）添加真实服务器(-g参数设置LVS工作模式为DR模式，-w设置权重)

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100 -g -w 1

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200 -g -w 1

4）查看规则列表，并保存规则

[root@proxy ~]# ipvsadm -Ln

TCP 192.168.4.15:80 wrr

-> 192.168.4.100:80 Route 1 0 0

-> 192.168.4.200:80 Route 2 0 0

[root@proxy ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

步骤四：客户端测试

客户端使用curl命令反复连接http://192.168.4.15，查看访问的页面是否会轮询到不同的后端真实服务器。

扩展知识：默认LVS不带健康检查功能，需要自己手动编写动态检测脚本，实现该功能：(参考脚本如下，仅供参考)

[root@proxy ~]# vim check.sh

#!/bin/bash

VIP=192.168.4.15:80

RIP1=192.168.4.100

RIP2=192.168.4.200

while :

do

for IP in $RIP1 $RIP2

do

curl -s http://$IP &>/dev/vnull

if [ $? -eq 0 ];then

ipvsadm -Ln |grep -q $IP || ipvsadm -a -t $VIP -r $IP

else

ipvsadm -Ln |grep -q $IP && ipvsadm -d -t $VIP -r $IP

fi

done

sleep 1

done

LVS[内核级调度器] 路由转发

web1

vip

web2

VIP给用户的IP

DIP调度连接web

RIP真实服务器的IP

NAT模式:从哪进从哪出,最简单

1.lvs开启路由转发

2.web需求配置网管

ipvsadm -A -t VIP -s rr|wrr|lc|wlc

ipvsadm -a -t VIP -r RIP -g|m

....

DR模式:

1.真实服务器配置4.0网段的IP

2.真实服务器伪装VIP,并且防止地址冲突,配置到回环上

3.lvs配置VIP在虚拟接口上 eth0 eth0:0

ipvsadm -A -t VIP -s rr

ipvsadm -a -t VIP -r RIP

keepalived

路由VRRP

vip router1 (ip1) 优先级

电脑

vip router2 (ip2) 优先级

keepalived功能:

1.实现VRRP

2.自动配置LVS规则 //提前写到配置文件

3.健康检查功能

TCP\_CHECK{

检查时间

}

ssl\_get{}

http\_chech{}

步骤一：配置网络环境（如果在前面课程已经完成该配置，可以忽略此步骤）

1）设置Web1服务器网络参数、配置Web服务

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.4.100" > /var/www/html/index.html

[root@web1 ~]# systemctl restart httpd

2）设置Web2服务器网络参数、配置Web服务

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.4.200" > /var/www/html/index.html

[root@web2 ~]# systemctl restart httpd

3）配置proxy主机的网络参数（如果已经设置，可以忽略此步骤）

[root@proxy ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth0

步骤二：安装Keepalived软件

注意：两台Web服务器做相同的操作。

[root@web1 ~]# yum install -y keepalived

[root@web2 ~]# yum install -y keepalived

步骤三：部署Keepalived服务

1）修改web1服务器Keepalived配置文件

[root@web1 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

admin@tarena.com.cn //设置报警收件人邮箱

}

notification\_email\_from ka@localhost //设置发件人

smtp\_server 127.0.0.1 //定义邮件服务器

smtp\_connect\_timeout 30

router\_id web1 //设置路由ID号（实验需要修改）

}

vrrp\_instance VI\_1 {

state MASTER //主服务器为MASTER（备服务器需要修改为BACKUP）

interface eth0 //定义网络接口

virtual\_router\_id 51 //主备服务器VRID号必须一致

priority 100 //服务器优先级,优先级高优先获取VIP

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主备服务器密码必须一致

}

virtual\_ipaddress { //谁是主服务器谁获得该VIP（实验需要修改）

192.168.4.80

}

}

2）修改web2服务器Keepalived配置文件

[root@web2 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

admin@tarena.com.cn //设置报警收件人邮箱

}

notification\_email\_from ka@localhost //设置发件人

smtp\_server 127.0.0.1 //定义邮件服务器

smtp\_connect\_timeout 30

router\_id web2 //设置路由ID号（实验需要修改）

}

vrrp\_instance VI\_1 {

state BACKUP //备服务器为BACKUP（实验需要修改）

interface eth0 //定义网络接口

virtual\_router\_id 51 //主辅VRID号必须一致

priority 50 //服务器优先级（实验需要修改）

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主辅服务器密码必须一致

}

virtual\_ipaddress { //谁是主服务器谁配置VIP（实验需要修改）

192.168.4.80

}

}

3）启动服务

[root@web1 ~]# systemctl start keepalived

[root@web2 ~]# systemctl start keepalived

4）配置防火墙和SELinux

启动keepalived会自动添加一个drop的防火墙规则，需要清空！

[root@web1 ~]# iptables -F

[root@web1 ~]# setenforce 0

[root@web2 ~]# iptables -F

[root@web1 ~]# setenforce 0

步骤四：测试

1）登录两台Web服务器查看VIP信息

[root@web1 ~]# ip addr show eth0

[root@web2 ~]# ip addr show eth0

2) 客户端访问

客户端使用curl命令连接http://192.168.4.80，查看Web页面；关闭Web1服务器的网卡，

客户端再次访问http://192.168.4.80，验证是否可以正常访问服务。

########################################################################################

Keepalived+LVS服务器

步骤一：配置网络环境

1）设置Web1服务器的网络参数

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

接下来给web1配置VIP地址

注意：这里的子网掩码必须是32（也就是全255），网络地址与IP地址一样，广播地址与IP地址也一样。

[root@web1 ~]# cd /etc/sysconfig/network-scripts/

[root@web1 ~]# cp ifcfg-lo{,:0}

[root@web1 ~]# vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

注意：这里因为web1也配置与调度器一样的VIP地址，默认肯定会出现地址冲突。

写入这四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应。

[root@web1 ~]# vim /etc/sysctl.conf

#手动写入如下4行内容

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

net.ipv4.conf.all.arp\_announce = 2

#当有arp广播问谁是192.168.4.15时，本机忽略该ARP广播，不做任何回应

#本机不要向外宣告自己的lo回环地址是192.168.4.15

重启网络服务，设置防火墙与SELinux

[root@web1 ~]# systemctl stop NetworkManager

[root@web1 ~]# systemctl disable NetworkManager

[root@web1 ~]# systemctl restart network

[root@web1 ~]# ifconfig

[root@web1 ~]# systemctl stop firewalld

[root@web1 ~]# setenforce 0

2）设置Web2服务器的网络参数

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

接下来给web2配置VIP地址

注意：这里的子网掩码必须是32（也就是全255），网络地址与IP地址一样，广播地址与IP地址也一样。

[root@web2 ~]# cd /etc/sysconfig/network-scripts/

[root@web2 ~]# cp ifcfg-lo{,:0}

[root@web2 ~]# vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

注意：这里因为web2也配置与代理一样的VIP地址，默认肯定会出现地址冲突。

写入这四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应。

[root@web2 ~]# vim /etc/sysctl.conf

#手动写入如下4行内容

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

net.ipv4.conf.all.arp\_announce = 2

#当有arp广播问谁是192.168.4.15时，本机忽略该ARP广播，不做任何回应

#本机不要向外宣告自己的lo回环地址是192.168.4.15

重启网络服务，设置防火墙与SELinux

[root@web2 ~]# systemctl stop NetworkManager

[root@web1 ~]# systemctl disable NetworkManager

[root@web2 ~]# systemctl restart network

[root@web2 ~]# ifconfig

[root@web2 ~]# systemctl stop firewalld

[root@web2 ~]# setenforce 0

3）配置proxy1主机的网络参数(不配置VIP，由keepalvied自动配置)

[root@proxy1 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy1 ~]# nmcli connection up eth0

4）配置proxy2主机的网络参数(不配置VIP，由keepalvied自动配置)

注意：按照前面的课程环境，默认没有该虚拟机，需要重新建一台虚拟机proxy2。

[root@proxy2 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.6/24 connection.autoconnect yes

[root@proxy2 ~]# nmcli connection up eth0

步骤二：配置后台web服务

1）安装软件，自定义Web页面（web1和web2主机）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.4.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.4.200" > /var/www/html/index.html

3）启动Web服务器软件(web1和web2主机)

[root@web1 ~]# systemctl start httpd ; systemctl enable httpd

[root@web2 ~]# systemctl start httpd ; systemctl enable httpd

步骤三：调度器安装Keepalived与ipvsadm软件

注意：两台LVS调度器执行相同的操作（如何已经安装软件，可用忽略此步骤）。

安装软件

[root@proxy1 ~]# yum install -y keepalived

[root@proxy1 ~]# systemctl enable keepalived

[root@proxy1 ~]# yum install -y ipvsadm

[root@proxy1 ~]# ipvsadm -C

[root@proxy2 ~]# yum install -y keepalived

[root@proxy2 ~]# systemctl enable keepalived

[root@proxy2 ~]# yum install -y ipvsadm

[root@proxy2 ~]# ipvsadm -C

步骤四：部署Keepalived实现LVS-DR模式调度器的高可用

1）LVS1调度器设置Keepalived，并启动服务

[root@proxy1 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

admin@tarena.com.cn //设置报警收件人邮箱

}

notification\_email\_from ka@localhost //设置发件人

smtp\_server 127.0.0.1 //定义邮件服务器

smtp\_connect\_timeout 30

router\_id lvs1 //设置路由ID号(实验需要修改)

}

vrrp\_instance VI\_1 {

state MASTER //主服务器为MASTER

interface eth0 //定义网络接口

virtual\_router\_id 50 //主辅VRID号必须一致

priority 100 //服务器优先级

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主辅服务器密码必须一致

}

virtual\_ipaddress { //配置VIP（实验需要修改）

192.168.4.15

}

}

virtual\_server 192.168.4.15 80 { //设置ipvsadm的VIP规则（实验需要修改）

delay\_loop 6

lb\_algo wrr //设置LVS调度算法为WRR

lb\_kind DR //设置LVS的模式为DR

#persistence\_timeout 50

#注意这样的作用是保持连接，开启后，客户端在一定时间内始终访问相同服务器

protocol TCP

real\_server 192.168.4.100 80 { //设置后端web服务器真实IP（实验需要修改）

weight 1 //设置权重为1

TCP\_CHECK { //对后台real\_server做健康检查

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.200 80 { //设置后端web服务器真实IP（实验需要修改）

weight 2 //设置权重为2

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

[root@proxy1 ~]# systemctl start keepalived

[root@proxy1 ~]# ipvsadm -Ln #查看LVS规则

[root@proxy1 ~]# ip a s #查看VIP配置

2）LVS2调度器设置Keepalived

[root@proxy2 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

admin@tarena.com.cn //设置报警收件人邮箱

}

notification\_email\_from ka@localhost //设置发件人

smtp\_server 127.0.0.1 //定义邮件服务器

smtp\_connect\_timeout 30

router\_id lvs2 //设置路由ID号（实验需要修改）

}

vrrp\_instance VI\_1 {

state BACKUP //从服务器为BACKUP（实验需要修改）

interface eth0 //定义网络接口

virtual\_router\_id 50 //主辅VRID号必须一致

priority 50 //服务器优先级（实验需要修改）

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主辅服务器密码必须一致

}

virtual\_ipaddress { //设置VIP（实验需要修改）

192.168.4.15

}

}

virtual\_server 192.168.4.15 80 { //自动设置LVS规则（实验需要修改）

delay\_loop 6

lb\_algo wrr //设置LVS调度算法为WRR

lb\_kind DR //设置LVS的模式为DR

# persistence\_timeout 50

#注意这样的作用是保持连接，开启后，客户端在一定时间内始终访问相同服务器

protocol TCP

real\_server 192.168.4.100 80 { //设置后端web服务器的真实IP（实验需要修改）

weight 1 //设置权重为1

TCP\_CHECK { //对后台real\_server做健康检查

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.200 80 { //设置后端web服务器的真实IP（实验需要修改）

weight 2 //设置权重为2

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

[root@proxy2 ~]# systemctl start keepalived

[root@proxy2 ~]# ipvsadm -Ln #查看LVS规则

[root@proxy2 ~]# ip a s #查看VIP设置

步骤五：客户端测试

客户端使用curl命令反复连接http://192.168.4.15，查看访问的页面是否会轮询到不同的后端真实服务器。

####################################################################

配置HAProxy负载平衡集群

注意事项：

将前面实验VIP、LVS等实验的内容清理干净！！！！！！

删除所有设备的VIP，清空所有LVS设置，关闭keepalived！！！

web1关闭多余的网卡与VIP，配置本地真实IP地址。

[root@web1 ~]# ifdown eth0

[root@web1 ~]# ifdown lo:0

[root@web1 ~]# nmcli connection modify eth1 ipv4.method manual \

ipv4.addresses 192.168.2.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth1

Web2关闭多余的网卡与VIP，配置本地真实IP地址。

[root@web2 ~]# ifdown eth0

[root@web2 ~]# ifdown lo:0

[root@web2 ~]# nmcli connection modify eth1 ipv4.method manual \

ipv4.addresses 192.168.2.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth1

proxy关闭keepalived服务，清理LVS规则。

[root@proxy ~]# systemctl stop keepalived

[root@proxy ~]# systemctl disable keepalived

[root@proxy ~]# ipvsadm -C

[root@proxy ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth0

[root@proxy ~]# nmcli connection modify eth1 ipv4.method manual \

ipv4.addresses 192.168.2.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth1

步骤一：配置后端Web服务器

设置两台后端Web服务（如果已经配置完成，可用忽略此步骤）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# systemctl start httpd

[root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# systemctl start httpd

[root@web2 ~]# echo "192.168.2.200" > /var/www/html/index.html

步骤二：部署HAProxy服务器

1）配置网络，安装软件

[root@haproxy ~]# yum -y install haproxy

2）修改配置文件

[root@haproxy ~]# vim /etc/haproxy/haproxy.cfg

global

log 127.0.0.1 local2 ###[err warning info debug]

chroot /usr/local/haproxy

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

maxconn 4000 ###最大连接数，默认4000

user haproxy

group haproxy

daemon ###创建1个进程进入deamon模式运行

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ip

option redispatch ###serverid服务器挂掉后强制定向到其他健康服务器

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 ###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 0.0.0.0:1080 #监听端口

stats refresh 30s #统计页面自动刷新时间

stats uri /stats #统计页面url

stats realm Haproxy Manager #进入管理解面查看状态信息

stats auth admin:admin #统计页面用户名和密码设置

#stats hide-version #隐藏统计页面上HAProxy的版本信息

listen websrv-rewrite 0.0.0.0:80

balance roundrobin

server web1 192.168.2.100:80 check inter 2000 rise 2 fall 5

server web2 192.168.2.200:80 check inter 2000 rise 2 fall 5

3）启动服务器并设置开机启动

[root@haproxy ~]# systemctl start haproxy

[root@haproxy ~]# systemctl enable haproxy

步骤三：客户端验证

客户端配置与HAProxy相同网络的IP地址，并使用火狐浏览器访问http://192.168.4.5，测试调度器是否正常工作，客户端访问http://192.168.4.5:1080/stats测试状态监控页面是否正常。访问状态监控页的内容，参考图-4所示。

图-4

备注：

Queue队列数据的信息（当前队列数量，最大值，队列限制数量）；

Session rate每秒会话率（当前值，最大值，限制数量）；

Sessions总会话量（当前值，最大值，总量，Lbtot: total number of times a server was selected选中一台服务器所用的总时间）；

Bytes（入站、出站流量）；

Denied（拒绝请求、拒绝回应）；

Errors（错误请求、错误连接、错误回应）；

Warnings（重新尝试警告retry、重新连接redispatches）；

Server(状态、最后检查的时间（多久前执行的最后一次检查）、权重、备份服务器数量、down机服务器数量、down机时长)。

listen 名称 \*:80

balance roundribin

server web1

server web2

存储类型:

DAS存储(直连存储,家用电脑)接口有限,容量有限

NAS存储(网络附加存储,网盘)文件系统共享:nfs,samba,http

ext3,ext4,xfs,ntfs,fat32

SAN存储(存储区域网络,网盘)块共享: iscsi

独立网络存储

SDS(软件定义存储)分布式存储(software define storage)

a.mp4(默认3副本)

Ceph分布式文件系统

具有高拓展,高可用,高性能特点

可以提供对象存储,块存储,文件系统存储

#######################################################

实验环境

准备四台KVM虚拟机，其三台作为存储集群节点，一台安装为客户端，实现如下功能：

创建1台客户端虚拟机

创建3台存储集群虚拟机

配置主机名、IP地址、YUM源

修改所有主机的主机名

配置无密码SSH连接

配置NTP时间同步

创建虚拟机磁盘

步骤一：安装前准备

1）物理机为所有节点配置yum源服务器。

提示：ceph10.iso在/linux-soft/02目录。

[root@room9pc01 ~]# mkdir /var/ftp/ceph

[root@room9pc01 ~]# mount ceph10.iso /var/ftp/ceph/

2）配置无密码连接(包括自己远程自己也不需要密码)，在node1操作。

[root@node1 ~]# ssh-keygen -f /root/.ssh/id\_rsa -N ''

[root@node1 ~]# for i in 10 11 12 13

do

ssh-copy-id 192.168.4.$i

done

3）修改/etc/hosts并同步到所有主机。

警告：/etc/hosts解析的域名必须与本机主机名一致！！！！

[root@node1 ~]# cat /etc/hosts

... ...

192.168.4.10 client

192.168.4.11 node1

192.168.4.12 node2

192.168.4.13 node3

4）修改所有节点都需要配置YUM源，并同步到所有主机。

[root@node1 ~]# cat /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=ftp://192.168.4.254/ceph/MON

gpgcheck=0

[osd]

name=osd

baseurl=ftp://192.168.4.254/ceph/OSD

gpgcheck=0

[tools]

name=tools

baseurl=ftp://192.168.4.254/ceph/Tools

gpgcheck=0

[root@node1 ~]# yum repolist #验证YUM源软件数量

源标识 源名称 状态

Dvd redhat 9,911

Mon mon 41

Osd osd 28

Tools tools 33

repolist: 10,013

[root@node1 ~]# for i in 10 11 12 13

do

scp /etc/yum.repos.d/ceph.repo 192.168.4.$i:/etc/yum.repos.d/

done

5）所有节点主机与真实主机的NTP服务器同步时间。

提示：默认真实物理机已经配置为NTP服务器。

[root@node1 ~]# vim /etc/chrony.conf

… …

server 192.168.4.254 iburst

[root@node1 ~]# for i in 10 11 12 13

do

scp /etc/chrony.conf 192.168.4.$i:/etc/

ssh 192.168.4.$i "systemctl restart chronyd"

done

步骤三：准备存储磁盘

物理机上为每个虚拟机准备3块磁盘（可以使用命令，也可以使用图形直接添加）。

[root@room9pc01 ~]# virt-manager

##############################################

部署ceph集群

步骤一：安装部署软件ceph-deploy

1）在node1安装部署工具，学习工具的语法格式。

[root@node1 ~]# yum -y install ceph-deploy

[root@node1 ~]# ceph-deploy --help

[root@node1 ~]# ceph-deploy mon --help

2）创建目录

[root@node1 ~]# mkdir ceph-cluster //后期必须在这个文件下执行

[root@node1 ~]# cd ceph-cluster/

步骤二：部署Ceph集群

1）创建Ceph集群配置,在ceph-cluster目录下生成Ceph配置文件。

在ceph.conf配置文件中定义monitor主机是谁。

[root@node1 ceph-cluster]# ceph-deploy new node1 node2 node3 //在当前生成配置文件(ceph.conf)

2）给所有节点安装ceph相关软件包。

[root@node1 ceph-cluster]# for i in node1 node2 node3

do

ssh $i "yum -y install ceph-mon ceph-osd ceph-mds ceph-radosgw"

done

3）初始化所有节点的mon服务，也就是启动mon服务（主机名解析必须对）。

[root@node1 ceph-cluster]# ceph-deploy mon create-initial

步骤三：创建OSD

[root@node1 ceph-cluster]# for i in node1 node2 node3

do

ssh $i "parted /dev/vdb mklabel gpt"

ssh $i "parted /dev/vdb mkpart primary 1 50%"

ssh $i "parted /dev/vdb mkpart primary 50% 100%"

done

2）磁盘分区后的默认权限无法让ceph软件对其进行读写操作，需要修改权限。

node1、node2、node3都需要操作，这里以node1为例。

[root@node1 ceph-cluster]# chown ceph.ceph /dev/vdb1

[root@node1 ceph-cluster]# chown ceph.ceph /dev/vdb2

#上面的权限修改为临时操作，重启计算机后，权限会再次被重置。

#我们还需要将规则写到配置文件实现永久有效。

#规则：如果设备名称为/dev/vdb1则设备文件的所有者和所属组都设置为ceph。

#规则：如果设备名称为/dev/vdb2则设备文件的所有者和所属组都设置为ceph。

[root@node1 ceph-cluster]# vim /etc/udev/rules.d/70-vdb.rules

ENV{DEVNAME}=="/dev/vdb1",OWNER="ceph",GROUP="ceph"

ENV{DEVNAME}=="/dev/vdb2",OWNER="ceph",GROUP="ceph"

3）初始化清空磁盘数据（仅node1操作即可）。

[root@node1 ceph-cluster]# ceph-deploy disk zap node1:vdc node1:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node2:vdc node2:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node3:vdc node3:vdd

4）创建OSD存储空间（仅node1操作即可）

重要：很多同学在这里会出错！将主机名、设备名称输入错误！！！

[root@node1 ceph-cluster]# ceph-deploy osd create \

node1:vdc:/dev/vdb1 node1:vdd:/dev/vdb2

//创建osd存储设备，vdc为集群提供存储空间，vdb1提供JOURNAL缓存，

//一个存储设备对应一个缓存设备，缓存需要SSD，不需要很大

[root@node1 ceph-cluster]# ceph-deploy osd create \

node2:vdc:/dev/vdb1 node2:vdd:/dev/vdb2

[root@node1 ceph-cluster]# ceph-deploy osd create \

node3:vdc:/dev/vdb1 node3:vdd:/dev/vdb2

步骤四：验证测试

1) 查看集群状态。

[root@node1 ~]# ceph -s

有警告的话查看一下时间同步

3 案例3：创建Ceph块存储

#################################################################################

步骤一：创建镜像

1）查看存储池。

[root@node1 ~]# ceph osd lspools

2）创建镜像、查看镜像

[root@node1 ~]# rbd create demo-image --image-feature layering --size 10G

[root@node1 ~]# rbd create rbd/image --image-feature layering --size 10G

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info demo-image //查看详细信息

步骤二：动态调整

1）缩小容量

[root@node1 ~]# rbd resize --size 7G image --allow-shrink

[root@node1 ~]# rbd info image

2）扩容容量

[root@node1 ~]# rbd resize --size 15G image

[root@node1 ~]# rbd info image

步骤三：通过KRBD访问

1）客户端通过KRBD访问

#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）

#拷贝连接密钥（否则无连接权限）

[root@client ~]# yum -y install ceph-common

[root@client ~]# scp 192.168.4.11:/etc/ceph/ceph.conf /etc/ceph/

[root@client ~]# scp 192.168.4.11:/etc/ceph/ceph.client.admin.keyring \

/etc/ceph/

[root@client ~]# rbd map image

[root@client ~]# lsblk

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

2) 客户端格式化、挂载分区

[root@client ~]# mkfs.xfs /dev/rbd0

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# echo "test" > /mnt/test.txt

步骤四：创建镜像快照

1)

[root@node1 ~]# rbd snap ls image

2) 给镜像创建快照。

[root@node1 ~]# rbd snap create image --snap image-snap1

#为image镜像创建快照，快照名称为image-snap1

[root@node1 ~]# rbd snap ls image

3) 删除客户端写入的测试文件

4) 还原快照

[root@node1 ~]# rbd snap rollback image --snap image-snap1

#客户端重新挂载分区

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# ls /mnt

步骤四：创建快照克隆

1）克隆快照

[root@node1 ~]# rbd snap protect image --snap image-snap1

[root@node1 ~]# rbd snap rm image --snap image-snap1 //会失败

[root@node1 ~]# rbd clone \

image --snap image-snap1 image-clone --image-feature layering

//使用image的快照image-snap1克隆一个新的名称为image-clone镜像

2）查看克隆镜像与父镜像快照的关系

[root@node1 ~]# rbd info image-clone

rbd image 'image-clone':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.d3f53d1b58ba

format: 2

features: layering

flags:

parent: rbd/image@image-snap1

#克隆镜像很多数据都来自于快照链

#如果希望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！！！

[root@node1 ~]# rbd flatten image-clone

[root@node1 ~]# rbd info image-clone

rbd image 'image-clone':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.d3f53d1b58ba

format: 2

features: layering

flags:

#注意，父快照信息没了！

[root@node1 ~]# rbd snap unprotect image --snap image-snap1 #取消快照保护

[root@node1 ~]# rbd snap rm image --snap image-snap1 #可以删除快照

步骤四：其他操作

1） 客户端撤销磁盘映射

[root@client ~]# umount /mnt

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

//语法格式:

[root@client ~]# rbd unmap /dev/rbd0

以上为案例

###########################################################################################

部署ceph集群:

yum -y installceph-deploy

ceph-deploy --help

1.部署ceph-mon

ceph-deploy new node1 node2 node3

yum -y install ceph-osd ceph-mon ceph-mds ceph-radosgw

ceph-deploy mon create-initial

ceph -s (no osds)

2.部署ceph-osd

有磁盘,格式化,启动共享

/dev/vdb1 ,vdb2

ceph-deploy disk zap node1:/vdc node1:/vdd

ceph-deploy osd create node1:/vdc:/dev/vdb1

.....

ceph 共享池

[共享镜像 共享镜像 共享镜像]

1.分区parted [dev/vdb1 dev/vdb2]

2.修改权限

chown ceph:ceph

vim /etc/udev/rules.d/xxx.rules

文件系统 二线ext3 ext4 xfs ntfs

inode(256) block(4k)

元数据(metedata) 数据

软件包:

ceph-mon ceph-osd ceph-mds ceph-radosgw

创建新机步骤

ceph-deploy new node1 node2 node3

ceph-deploy mon create initial

ceph-deploy0sd create node1 node2 node3

ceph-deploy mds create node3

ceph-deploy rgw create node3

案例1：块存储应用案例

1）创建磁盘镜像。

[root@node1 ~]# rbd create vm1-image --image-feature layering --size 10G

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info vm1-image

2）Ceph认证账户（仅查看即可）。

[root@node1 ~]# cat /etc/ceph/ceph.conf //配置文件

[global]

mon\_initial\_members = node1, node2, node3

mon\_host = 192.168.2.10,192.168.2.20,192.168.2.30

auth\_cluster\_required = cephx //开启认证

auth\_service\_required = cephx //开启认证

auth\_client\_required = cephx //开启认证

[root@node1 ~]# cat /etc/ceph/ceph.client.admin.keyring //账户文件

[client.admin]

key = AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==

3）创建KVM虚拟机（注意：这里使用真实机当客户端！！！）。

使用virt-manager创建2台普通的KVM虚拟机。

4）配置libvirt secret（注意：这里使用真实机当客户端！！！）。

编写账户信息文件，让KVM知道ceph的账户名称。

[root@room9pc01 ~]# vim secret.xml //新建临时文件，内容如下

<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

[root@room9pc01 ~]# virsh secret-define secret.xml

733f0fd1-e3d6-4c25-a69f-6681fc19802b

//随机的UUID，这个UUID对应的有账户信息

给secret绑定admin账户的密码，密码参考ceph.client.admin.keyring文件。

[root@room9pc01] virsh secret-set-value \

--secret 733f0fd1-e3d6-4c25-a69f-6681fc19802b \

--base64 AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg

//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

6）虚拟机的XML配置文件。

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。

[root@room9pc01 ~]# vim /etc/libvirt/qemu/vm1.xml

//修改前内容如下

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2'/>

<source file='/var/lib/libvirt/images/vm1.qcow2'/>

<target dev='vda' bus='virtio'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>

</disk>

不推荐直接使用vim修改配置文件，推荐使用virsh edit修改配置文件，效果如下：

[root@room9pc01] virsh edit vm1 //vm1为虚拟机名称

<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='192.168.4.11' port='6789'/>  
 </source>

<target dev='vda' bus='virtio'/>

<address type='pci' domain='0x0000' bus='0x09' slot='0x08' function='0x0'/>

</disk>

注意：如果有设备编号冲突的情况下，需要修改设备编号，任意修改一个数字即可。

################## 需要正常关机

#####################################################################################

Ceph文件系统

步骤

实现此案例需要按照如下步骤进行。

1）添加一台新的虚拟机，要求如下：

IP地址:192.168.4.14

主机名:node4

配置yum源（包括rhel、ceph的源）

与Client主机同步时间

node1允许无密码远程node4

2）部署元数据服务器

登陆node4，安装ceph-mds软件包

[root@node4 ~]# yum -y install ceph-mds

登陆node1部署节点操作

[root@node1 ~]# cd /root/ceph-cluster

//该目录，是最早部署ceph集群时，创建的目录

[root@node1 ceph-cluster]# ceph-deploy mds create node4

//给nod4拷贝配置文件，启动mds服务

同步配置文件和key

[root@node1 ceph-cluster]# ceph-deploy admin node4

3）创建存储池

[root@node4 ~]# ceph osd pool create cephfs\_data 128

//创建存储池，对应128个PG

[root@node4 ~]# ceph osd pool create cephfs\_metadata 128

//创建存储池，对应128个PG

5）创建Ceph文件系统

[root@node4 ~]# ceph mds stat //查看mds状态

e2:, 1 up:standby

[root@node4 ~]# ceph fs new myfs1 cephfs\_metadata cephfs\_data

new fs with metadata pool 2 and data pool 1

//注意，先写medadata池，再写data池

//默认，只能创建1个文件系统，多余的会报错

[root@node4 ~]# ceph fs ls

name: myfs1, metadata pool: cephfs\_metadata, data pools: [cephfs\_data ]

[root@node4 ~]# ceph mds stat

e4: 1/1/1 up {0=node4=up:creating}

6）客户端挂载

[root@client ~]# mount -t ceph 192.168.4.11:6789:/ /mnt/cephfs/ \

-o name=admin,secret=AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==

//注意:文件系统类型为ceph

//192.168.4.11为MON节点的IP（不是MDS节点）

//admin是用户名,secret是密钥

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

##############################

案例3：创建对象存储服务器

步骤一：部署对象存储服务器

1）准备实验环境，要求如下：

IP地址:192.168.4.15

主机名:node5

配置yum源（包括rhel、ceph的源）

与Client主机同步时间

node1允许无密码远程node5

修改node1的/etc/hosts，并同步到所有node主机

2）部署RGW软件包

[root@node1 ~]# ceph-deploy install --rgw node5

同步配置文件与密钥到node5

[root@node1 ~]# cd /root/ceph-cluster

[root@node1 ~]# ceph-deploy admin node5

3）新建网关实例

启动一个rgw服务

[root@node1 ~]# ceph-deploy rgw create node5

登陆node5验证服务是否启动

[root@node5 ~]# ps aux |grep radosgw

ceph 4109 0.2 1.4 2289196 14972 ? Ssl 22:53 0:00 /usr/bin/radosgw -f --cluster ceph --name client.rgw.node4 --setuser ceph --setgroup ceph

[root@node5 ~]# systemctl status ceph-radosgw@\\*

4）修改服务端口

登陆node5，RGW默认服务端口为7480，修改为8000或80更方便客户端记忆和使用

[root@node5 ~]# vim /etc/ceph/ceph.conf

[client.rgw.node5]

host = node5

rgw\_frontends = "civetweb port=8000"

//node5为主机名

//civetweb是RGW内置的一个web服务

步骤二：客户端测试（扩展选做实验）

1）curl测试

[root@client ~]# curl 192.168.4.15:8000

<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID><DisplayName></DisplayName></Owner><Buckets></Buckets></ListAllMyBucketsResult>

2）使用第三方软件访问

登陆node5（RGW）创建账户

[root@node5 ~]# radosgw-admin user create \

--uid="testuser" --display-name="First User"

… …

"keys": [

{

"user": "testuser",

"access\_key": "5E42OEGB1M95Y49IBG7B",

"secret\_key": "i8YtM8cs7QDCK3rTRopb0TTPBFJVXdEryRbeLGK6"

}

],

... ...

#

[root@node5 ~]# radosgw-admin user info --uid=testuser

//testuser为用户，key是账户访问密钥

3）客户端安装软件

[root@client ~]# yum install s3cmd-2.0.1-1.el7.noarch.rpm

以下未作出

修改软件配置（注意，除了下面设置的内容，其他提示都默认回车）

[root@client ~]# s3cmd --configure

Access Key: 5E42OEGB1M95Y49IBG7B  
Secret Key: i8YtM8cs7QDCK3rTRopb0TTPBFJVXdEryRbeLGK6

S3 Endpoint [s3.amazonaws.com]: 192.168.4.15:8000

[%(bucket)s.s3.amazonaws.com]: %(bucket)s.192.168.4.15:8000

Use HTTPS protocol [Yes]: No

Test access with supplied credentials? [Y/n] n

Save settings? [y/N] y

//注意，其他提示都默认回车

4）创建存储数据的bucket（类似于存储数据的目录）

[root@client ~]# s3cmd ls

[root@client ~]# s3cmd mb s3://my\_bucket

Bucket 's3://my\_bucket/' created

[root@client ~]# s3cmd ls

2018-05-09 08:14 s3://my\_bucket

[root@client ~]# s3cmd put /var/log/messages s3://my\_bucket/log/

[root@client ~]# s3cmd ls

2018-05-09 08:14 s3://my\_bucket

[root@client ~]# s3cmd ls s3://my\_bucket

DIR s3://my\_bucket/log/

[root@client ~]# s3cmd ls s3://my\_bucket/log/

2018-05-09 08:19 309034 s3://my\_bucket/log/messages

测试下载功能

[root@client ~]# s3cmd get s3://my\_bucket/log/messages /tmp/

测试删除功能

[root@client ~]# s3cmd del s3://my\_bucket/log/messages

# OPERATION全部笔记

192.168.4.5 :nginx服务器

web服务器: (httpd\nginx\lighttpd

tomcat\weblogic\websphere\jboss)

nginx(taobao)-->tengine

F12 server: taobao (tengine)

./configure --help // 帮助 with 安装 without 不安装

netstat命令可以查看系统中启动的端口信息，该命令常用选项如下：

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

相对的是nginx 的安装目录(/usr/local/nginx/)

index index.html index.htm; //默认首页,默认页面

1)源码安装nginx

1.安装依赖包 gcc pcre-devel openssl-devel

2.安全 创建用户

3.解包 cd

4.configure --user --group --prefix --with-名称

make make install

5.升级 手动拷贝nginx ,重启

2)默认有一个网站(允许所有人访问)

用户认证

vim nginx.conf

server {

auth\_basic "字符";

auth\_basic\_user\_file "文件";

}

yum -y install httpd-tools

htpasswd -c 文件 用户名

>密码

nginx -s reload //重启

3)虚拟主机(做多个网站)域名

vim nginx.conf

server {

charset utf-8; //使网站支持中文

server\_name www.a.com:

}

server {

server\_name www.b.com

}

#mkdir /usr/local/nginx/www

#echo 内容 > 主页

#yum -y install \*fonts\*

4)httpd加密网站

对称 123 123

非对称 123 abc

//机制问题网站不能使用对称加密

cd /usr/local/nginx/conf

openssl genrsa > cert.key //生成私钥

openssl req -new -x509 -key cert.key > cert.pem //生成证书

LNMP(linux+nginx+mysql+php), LAMP

静态

动态

web:/var/www/html

/usr/local/nginx/html

yum安装软件

nginx

mariadb mariadb-server mariadb-devel

php php-fpm php-mysql

1）启动MySQL服务

[root@proxy ~]# systemctl start mariadb //启动服务器

[root@proxy ~]# systemctl status mariadb //查看服务状态

[root@proxy ~]# systemctl enable mariadb //设置开机启动

2）启动PHP-FPM服务

[root@proxy ~]# systemctl start php-fpm //启动服务

[root@proxy ~]# systemctl status php-fpm //查看服务状态

[root@proxy ~]# systemctl enable php-fpm //设置开机启动

实现动静分离(nginx.conf),做判断

如果用户访问的静态页面,则直接返回

如果用户访问的是php脚本,则把脚本转发给9000端口

location语句是匹配用户的地址栏(从域名后面开始)

location 支持正则

location ~ /abc {

}

php是多进程软件

################################

程序 (program)

-进程(process)

-线程(thread)

-线程

-进程

-线程

################################

Nginx的默认访问日志文件为/usr/local/nginx/logs/access.log

Nginx的默认错误日志文件为/usr/local/nginx/logs/error.log

PHP默认错误日志文件为/var/log/php-fpm/www-error.log

如果动态网站访问失败，可用参考错误日志，查找错误信息

使用tailf (与tail区别,动态查看)

创建新的动态页面:

[root@proxy conf]# cd /root/lnmp\_soft/php\_scripts/

[root@proxy php\_scripts]# cp mysql.php /usr/local/nginx/html/

[root@client ~]# firefox www.a.com/mysql.php

查看变化:

mysql创建用户

grant all on \*.\* to 'dc'@'1.1.1.1' identified by '123';

####################################################

地址重写:

rewrite 旧地址 新地址 [选项]

rewrite 旧地址 新地址 [选项];

last 不再读其他rewrite

break 不再读其他语句，结束请求

redirect 临时重定向

permament 永久重定向

配置默认安装,没安装的话要按升级步骤重新启用

访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面

rewrite /(.\*) www.tmooc.cn/$1; //$1代表前面括号里的值

1.安装软件 7个

2.启动服务 nginx,php-fpm,mariadb

3.修改配置 location ~ \.php${ }

4.nginx -s reload

5.firefox http://www.a.com/test.php

nginx web服务器

代理服务器

功能:

1.调度(轮询)

2.健康检查

upstream abc{

server 192.168.2.100 weight=2;

server 192.168.2.200 max\_fails= fail\_timeout=;

}

server{

proxy\_pass http://abc;

proxy\_pass abc;

}

#####################################################

修改默认404页面:主配置文件下的error..404(去#号)

error\_page 404 /404.html;

error\_page 402 /402.html;

error\_page 401 /401.html;

error\_page 401 402 403 404 405 /a.html;

#####################################################

web代理服务器/web调度器(http协议)

默认不安装

./configure --with-stream

nginx服务器:

关心的问题:

谁在访问你

访问量多少

并发量多少

PV是多少(page view 页面被点击多少次)

UV是多少(user view 有多少用户点击)

有没有等待用户,是多少

统计信息模块:

--with-http\_stub\_status\_module

[root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf

… …

location /status {

stub\_status on;

#allow IP地址;

#deny IP地址;

}

… …

[root@proxy ~]# nginx

ss命令可以查看系统中启动的端口信息，该命令常用选项如下：

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

########################################################

tcp/udp:

3次握手,4次断开

http 一次连接,一次请求

keepalive 一次连接多次请求

#############################################

PV是多少(page view 页面被点击多少次)

UV是多少(user view 有多少用户点击)

/usr/local/nginx/logs/access.log

awk '{IP[$1]++} END{for (i in IP){print IP[i],i}}' access.log

#############################################

客户端的电脑(客户端)--------------------服务器

缓存的功能

存什么,存多久

动态数据

静态数据(图片视频音频)

location ~\* \.(jpg|png) {

expires 30d; //如果访问图片,则缓存30天

}

#################################

kill 作用是给程序发送 信号

kill -l //查看

####################################################

cd /var/lib/php/session/ //查看服务器本地的Session信息

web页面更改背景颜色:

<body bgcolor=blue>

####################################################

PHP实现session共享

[root@web1 ~]# yum -y install php-pecl-memcache

[root@web1 ~]# vim /etc/php-fpm.d/www.conf //修改该配置文件的两个参数

//文件的最后2行

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

php\_value[session.save\_path] = /var/lib/php/session

//原始文件，默认定义Sessoin会话信息本地计算机（默认在/var/lib/php/session）

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

php\_value[session.save\_path] = "tcp://192.168.2.5:11211"

session,cookie

HTTP 无状态协议

web1

client--------proxy memcached(数据库)

web2

web1创建session (文件名=状态)

web1创建session (连接memcached(234df89=状态))

静态数据(图片视频音频文档)

动态数据(脚本)

PHP,java,python,perl,ruby,shell

java{跨平台的开发语言}

Windows{jdk.JVM java虚拟机}

linux{jdk,JVM,java虚拟机}

MacOS{jdk,JVM,java虚拟机}

电视盒子l{jdk,JVM,java虚拟机}

tomcat 就是用java写的代码

######################################################

web虚拟主机:

apache:

<viryualhost \*:80>

servername

documentroot

</virtualhost>

nginx:

server{

listen 80;

server\_name;

root html;

}

ls /dev/random

ln -s /dev/urandom /dev/random //tomcat启动需要大量随机字符

tomcat:

<engin ...defaulthost=www.a.com> //输ip默认打开页面

<host name=www.a.com appBase=a> //appBase=a(路径)

</Host> //与上面对应,首字母大写

<host name=www.b.com appBase=b> //appBase=b(路径)

<Context path="" docBase="base"/ > //指定网页文件路径

</Host> //与上面对应

</Eigine> //与上面对应

##########################################################

二:修改www.b.com网站的首页目录为base

1）使用docBase参数可以修改默认网站首页路径

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base"/>

</Host>

… …

[root@web1 ~]# mkdir /usr/local/tomcat/b/base

[root@web1 ~]# echo "BASE" > /usr/local/tomcat/b/base/index.html

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh //关闭tomcat

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh //开启tomcat

##################################################

三:跳转

1）当用户访问http://www.a.com/test打开/var/www/html目录下的页面

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

<Context path="/test" docBase="/var/www/html/" />

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base" />

</Host>

[root@web1 ~]# echo "Test" > /var/www/html/index.html

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

测试查看页面是否正确

################################################

四:配置Tomcat支持SSL加密网站:

1）创建加密用的私钥和证书文件

[root@web1 ~]# keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore //提示输入密码为:123456

//-genkeypair 生成密钥对

//-alias tomcat 密钥别名

//-keyalg RSA 定义密钥算法为RSA算法

//-keystore 定义密钥文件存储在:/usr/local/tomcat/keystore

2)再次修改server.xml配置文件，创建支持加密连接的Connector

###############################################################

五：配置Tomcat日志

vim /usr/local/tomcat/conf/server.xml

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

<Context path="/test" docBase="/var/www/html/" />

#从默认localhost虚拟主机中把Valve这段复制过来，适当修改下即可

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

prefix=" a\_access" suffix=".txt"

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

</Host>

重启Tomcat服务器

nginx[代理加软件]

varnish代理软件

varnish[代理加软件] CDN(content delivery network) 内容分发网络

版本控制软件

人,文件就一个

机器,版本有很多[自动保存所有版本]

集中版本控制: SVN(subversion)

git init /仓库名称 --bare

分布式版本控制: git

##################################

全览:

web1: /var/git/project(仓库)

web2: git clone ip:/var/git/project

project(工作目录)

project/.git(本地仓库)

project/新建文件a.txt

git add . 建索引

git commit -m "注释" 把a.txt拷贝到.git(提交本地仓库)

project/新建文件a.txt(修改文件)()v2

git add . 建索引

git commit -m "注释" .git (a.txt-v1 a.txt-v2 a.txt-v3)

......

git push //提交到网络服务器

git pull //下载服务器的新版本

最常用四个命令:

git add .

git commit

git push

git pull

###########################################################

一：部署Git服务器

1）YUM安装Git软件。

[root@web1 ~]# yum -y install git

[root@web1 ~]# git --version //查看版本号

2)初始化一个空仓库。

[root@web1 ~]# mkdir /var/git //创建一个普通目录

[root@web1 ~]# git init /var/git/project --bare 初始化仓库

[root@web1 ~]# ls /var/git/ /查看

二:客户端

1) clone克隆服务器仓库到本地

[root@web2 ~]# yum -y install git

[root@web2 ~]# git clone root@192.168.2.100:/var/git/project //克隆服务器仓库

2) 修改git配置

[root@web2 project]# git config --global user.email "you@example.com" //邮箱

[root@web2 project]# git config --global user.name "Your Name" //用户名

[root@web2 project]# cat ~/.gitconfig //查看自己配置

3） 本地工作区对数据进行增删改查(必须要先进入仓库再操作数据)。

root@web2 project]# echo "init date" > init.txt //创建文档

[root@web2 project]# mkdir demo

[root@web2 project]# cp /etc/hosts demo

4） 查看仓库中数据的状态。

[root@web2 project]# git status

5） 将工作区的修改提交到暂存区。

[root@web2 project]# git add . //. 代表所有,可写文档名单加

6) 将暂存区修改提交到本地仓库。

[root@web2 project]# git commit -m "注释，可以为任意字符" //提交到仓库

[root@web2 project]# git status

查看日志方法:

git log //查看提交日志

git log --pretty=oneline //在一行显示提交日志

git log --oneline //查看版本码前几位

git reflog //查看版本码前几位

Windows需要安装git和tortoiseGit

#######################################################

Git分支操作

[root@web2 project]# git status //查看当前工作分支

[root@web2 project]# git branch -v //查看当前工作分支

[root@web2 project]# git branch hotfix //创建后提醒分支(git branch 分支名)

[root@web2 project]# git checkout hotfix //切换hotfix 分支

在hotfix修改代码

[root@web2 project]# git checkout master //切回master分支

[root@web2 project]# git merge hotfix //合并分支

[root@web2 project]# git push //提交到服务器

#####################################################################

git的几种服务器:###

SSH协议服务器（支持读写操作）

1）创建基于密码验证的SSH协议服务器（web1主机操作）。

[root@web1 ~]# git init --bare /var/git/base\_ssh

Initialized empty Git repository in /var/git/base\_ssh/

2)客户端访问的方式（web2主机操作）。

[root@web2 ~]# git clone root@192.168.2.100:/var/git/base\_ssh

3）客户端生成SSH密钥，实现免密码登陆git服务器（web2主机操作）。

设置免密登录:

ssh-keygen -f /root/.ssh/id\_rsa -N '' //生成秘钥,无交互

ssh-copy-id 192.168.2.100 //传递秘钥

[root@web2 ~]# git clone root@192.168.2.100:/var/git

Git协议服务器（只读操作的服务器）

1）安装git-daemon软件包（web1主机操作）。

[root@web1 ~]# yum -y install git-daemon

2）创建版本库（web1主机操作）。

[root@web1 ~]# git init --bare /var/git/base\_git

Initialized empty Git repository in /var/git/base\_git/

3）修改配置文件，启动git服务（web1主机操作）。

[root@web1 ~]# vim /usr/lib/systemd/system/git@.service

修改前内容如下：

ExecStart=-/usr/libexec/git-core/git-daemon --base-path=/var/lib/git

--export-all --user-path=public\_git --syslog --inetd –verbose

修改后内容如下：

ExecStart=-/usr/libexec/git-core/git-daemon --base-path=/var/git

--export-all --user-path=public\_git --syslog --inetd –verbose

[root@web1 ~]# systemctl start git.socket

4）客户端访问方式（web2主机操作）

[root@web2 ~]# git clone git://192.168.2.100/base\_git

步骤三：HTTP协议服务器（只读操作的服务器）

1）安装gitweb、httpd软件包（web1主机操作）。

[root@web1 ~]# yum -y install httpd gitweb

2）修改配置文件，设置仓库根目录（web1主机操作）。

[root@web1 ~]# vim +11 /etc/gitweb.conf

$projectroot = "/var/git"; #添加一行

3) 创建版本仓库（web1主机操作）

[root@web1 ~]# git init --bare /var/git/base\_http

4）启动httpd服务器

[root@web1 ~]# systemctl start httpd

5）客户端访问方式（web2主机操作）

注意：调用虚拟机中的firefox浏览器，需要在远程时使用ssh -X 服务器IP，并且确保真实主机的firefox已经关闭。

[root@web2 ~]# firefox http://192.168.2.100/git/

###############################################################################

制作nginx的RPM包

RPM是一个压缩包

把/usr/local/nginx 打包压缩nginx.rpm

rpm -ivh nginx.rpm

/usr/local.nginx

yum -y ins install nginx

####################################################################

步骤一：安装rpm-build软件

1)[root@web1 ~]# yum -y install rpm-build

2）生成rpmbuild目录结构

[root@web1 ~]# rpmbuild -ba nginx.spec //会报错，没有文件或目录

[root@web1 ~]# ls /root/rpmbuild //自动生成的目录结构

BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

3）准备工作，将源码软件复制到SOURCES目录

[root@web1 ~]# cp nginx-1.12.2.tar.gz /root/rpmbuild/SOURCES/

4）创建并修改SPEC配置文件

[root@web1 ~]# vim /root/rpmbuild/SPECS/nginx.spec

Name:nginx //源码包软件名称

Version:1.12.2 //源码包软件的版本号

Release: 10 //制作的RPM包版本号

Summary: Nginx is a web server software. //RPM软件的概述

#Group //注释掉

License:GPL //软件的协议

URL: www.test.com //网址

Source0:nginx-1.12.2.tar.gz //源码包文件的全称

#BuildRequires: //制作RPM时的依赖关系

#Requires: //安装RPM时的依赖关系

%description

nginx [engine x] is an HTTP and reverse proxy server. //软件的详细描述

%post //需要自己添加,安装后脚本

useradd nginx //非必需操作：安装后脚本(创建账户)

%prep

%setup -q //自动解压源码包，并cd进入目录

%build

./configure

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%doc

/usr/local/nginx/\* //对哪些文件与目录打包

%changelog

1)三个不能改错的:

name version source

2)编译安装,打包

./configure

%file

/usr/local/nginx/\*

目前主流VPN技术(GRE,PPTP,l2tp+ipsec)

难度:

GRE<PPTP<l2tp+ipsec

安全:

GRE<PPTP<l2tp+ipsec

#################################################################

linux操作(模块化设计)

lsmod //查看启动多少模块

modprobe 模块名 //激活模块

###############################################################

三步:

ip tunnel add tun0 mode gre remote 201.1.2.10 local 201.1.2.5 //建立隧道

ip link set tun0 up //启动网卡

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

两边都要做,ip需互换

步骤一：启用GRE模块

[root@client ~]# lsmod //显示模块列表

[root@client ~]# lsmod | grep ip\_gre //确定是否加载了gre模块

2)加载模块ip\_gre

[root@client ~]# modprobe ip\_gre

3）查看模块信息

[root@client ~]# modinfo ip\_gre

步骤二：Client主机创建VPN隧道

1）创建隧道

[root@client ~]# ip tunnel add tun0 mode gre remote 201.1.2.5 local 201.1.2.10

//ip tunnel add创建隧道（隧道名称为tun0），ip tunnel help可以查看帮助

//mode设置隧道使用gre模式

//local后面跟本机的IP地址，remote后面是与其他主机建立隧道的对方IP地址

2）启用该隧道（类似与设置网卡up）

[root@client ~]# ip link show

[root@client ~]# ip link set tun0 up //设置UP

[root@client ~]# ip link show

3）为VPN配置隧道IP地址

[root@client ~]# ip addr add 10.10.10.10/24 peer 10.10.10.5/24 dev tun0

//为隧道tun0设置本地IP地址（10.10.10.10.10/24）

//隧道对面的主机IP的隧道IP为10.10.10.5/24

[root@client ~]# ip a s //查看IP地址

开启路由转发、关闭防火墙

[root@proxy ~]# echo "1" > /proc/sys/net/ipv4/ip\_forward //此文件只能echo,不能vim编辑,1开启,0关闭

###################################################

案例2：创建PPTP VPN {服务器设置}

部署VPN服务器

1）安装软件包（软件包参考lnmp\_soft）

[root@proxy ~]# yum localinstall pptpd-1.4.0-2.el7.x86\_64.rpm

[root@proxy ~]# rpm -qc pptpd

/etc/ppp/options.pptpd

/etc/pptpd.conf

/etc/sysconfig/pptpd

2)修改配置文件

[root@proxy ~]# 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

[root@proxy ~]# vim /etc/ppp/options.pptpd

require-mppe-128 //使用MPPE加密数据

ms-dns 8.8.8.8 //DNS服务器

[root@proxy ~]# vim /etc/ppp/chap-secrets //修改账户配置文件

jacob \* 123456 \*

//用户名 服务器标记 密码 客户端

[root@proxy ~]# echo "1" > /proc/sys/net/ipv4/ip\_forward //开启路由转发

3）启动服务

[root@proxy ~]# systemctl start pptpd

[root@proxy ~]# systemctl enable pptpd

[root@proxy ~]# firewall-cmd --set-default-zone=trusted

4）翻墙设置（非必需操作）

[root@proxy ~]# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 \

> -j SNAT --to-source 201.1.2.5

47.88.32.119#########

esc.console.aliyun#################

#############################################################

创建L2TP+IPSec VPN {服务器设置}

1）安装软件包

[root@client ~]# yum -y install libreswan

2)新建IPSec密钥验证配置文件

[root@client ~]# cat /etc/ipsec.conf //仅查看一下该主配置文件

.. ..

include /etc/ipsec.d/\*.conf //加载该目录下的所有配置文件

[root@client ~]# vim /etc/ipsec.d/myipsec.conf

//新建该文件，参考lnmp\_soft/vpn/myipsec.conf

conn IDC-PSK-NAT

rightsubnet=vhost:%priv //允许建立的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=201.1.2.10 //重要，服务器本机的外网IP

leftprotoport=17/1701

right=%any //允许任何客户端连接

rightprotoport=17/%any

3)创建IPSec预定义共享密钥

[root@client ~]# cat /etc/ipsec.secrets //仅查看，不要修改该文件

include /etc/ipsec.d/\*.secrets

[root@client ~]# vim /etc/ipsec.d/mypass.secrets //新建该文件

201.1.2.10 %any: PSK "randpass" //randpass为预共享密钥

//201.1.2.10是VPN服务器的IP

4)启动IPSec服务

[root@client ~]# systemctl start ipsec

[root@client ~]# netstat -ntulp |grep pluto

udp 0 0 127.0.0.1:4500 0.0.0.0:\* 3148/pluto

udp 0 0 192.168.4.10:4500 0.0.0.0:\* 3148/pluto

udp 0 0 201.1.2.10:4500 0.0.0.0:\* 3148/pluto

udp 0 0 127.0.0.1:500 0.0.0.0:\* 3148/pluto

udp 0 0 192.168.4.10:500 0.0.0.0:\* 3148/pluto

udp 0 0 201.1.2.10:500 0.0.0.0:\* 3148/pluto

udp6 0 0 ::1:500 :::\* 3148/pluto

步骤二：部署XL2TP服务

1）安装软件包（软件包参考lnmp\_soft）

[root@client ~]# yum localinstall xl2tpd-1.3.8-2.el7.x86\_64.rpm

2) 修改xl2tp配置文件（修改3个配置文件的内容）

[root@client ~]# vim /etc/xl2tpd/xl2tpd.conf //修改主配置文件

[global]

.. ..

[lns default]

.. ..

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

local ip = 201.1.2.10 //VPN服务器的IP地址

[root@client ~]# vim /etc/ppp/options.xl2tpd //认证配置

require-mschap-v2 //添加一行，强制要求认证

#crtscts //注释或删除该行

#lock //注释或删除该行

root@client ~]# vim /etc/ppp/chap-secrets //修改密码文件

jacob \* 123456 \* //账户名称 服务器标记 密码 客户端IP

3）启动服务

[root@client ~]# systemctl start xl2tpd

[root@client ~]# netstat -ntulp |grep xl2tpd

udp 0 0 0.0.0.0:1701 0.0.0.0:\* 3580/xl2tpd

4）设置路由转发，防火墙

[root@client ~]# echo "1" > /proc/sys/net/ipv4/ip\_forward

[root@client ~]# firewall-cmd --set-default-zone=trusted

5）翻墙设置（非必需操作）

[root@client ~]# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 \

> -j SNAT --to-source 201.1.2.10

######################################################################################

四: NTP时间同步

部署NTP服务

1）安装软件包

[root@proxy ~]# yum -y install chrony

[root@proxy ~]# rpm -qc chrony //查看配置文件列表

/etc/chrony.conf

/etc/chrony.keys

2)修改配置文件

[root@proxy ~]# cat /etc/chrony.conf

.. ..

server 0.centos.pool.ntp.org iburst //server用户客户端指向上层NTP服务器

allow 192.168.4.0/24 //允许那个IP或网络访问NTP

#deny 192.168.4.1 //拒绝那个IP或网络访问NTP

local stratum 10 //设置NTP服务器的层数量

3)启动NTP服务

[root@proxy ~]# systemctl restart chronyd

[root@proxy ~]# systemctl enable chronyd

配置客户端

1）安装软件包

[root@client ~]# yum -y install chrony

2) 修改配置文件

[root@client ~]# vim /etc/chrony.conf

server 192.168.4.5 iburst //设置与哪台服务器同步数据

//iburst参数设置重启服务后尽快同步时间

3) 将客户端时间修改为错误的时间

[root@client ~]# date -s "hour:minute" //调整时间（小时：分钟）

[root@client ~]# date //查看修改后的时间

4) 重启chrony与服务器同步时间

[root@client ~]# systemctl restart chronyd

5) 确认时间是否已经同步

[root@client ~]# date //多执行几次查看结果

第三阶段 （数据库管理员 DBA ） （运维工程师 DBA）

1 RDBMS---MYSQL 2 NOsql---Redis 3 监控与安全

RDBMS1---day01

一、搭建数据库服务器

1.1 相关概念

1.2 在50主机部署mysql数据库服务 练习到 11：35

解包

安装软件包

启动mysqld服务

查看服务进程和端口

查看初始登陆密码

使用初始密码登陆

修改登陆密码

断开连接

使用修改后的密码登陆

修改密码策略

根据新密码策略设置密码

使用新密码登陆

]# tar -xvf mysql-5.7.17.tar

]# ls \*.rpm

]# yum -y install mysql-community-\*.rpm

]# systemctl start mysqld

]# ls /var/lib/mysql

]# systemctl enable mysqld

]# netstat -utnlp | grep :3306

]# ps -C mysqld

]# grep password /var/log/mysqld.log

]# mysql -hlocalhost -uroot -p'f<tbekh-d8Y!'

mysql> alter user root@"localhost" identified by "123qqq...A";

mysql> exit

]# mysql -uroot -p123qqq...A

mysql> show variables like "%password%";

mysql> set global validate\_password\_length=6;

mysql> set global validate\_password\_policy=0;

mysql> alter user root@"localhost" identified by "tarena";

mysql> exit

]# mysql -uroot -ptarena

]# vim /etc/my.cnf

[mysqld]

validate\_password\_length=6

validate\_password\_policy=0

:wq

二、数据库服务的基本使用

2.1 连接方式

2.2 数据存储流程 ?

2.3 mysql管理环境 (SQL命令)

练习25分钟到 13：55

2.4 库管理命令 （创建新库 查看已有的库 切换库 删除库 库名命名规则）

2.5 表管理命令 （查看已有的表 创建新表 查看表结构 查看表记录）

2.6 记录管理命令（查看已有记录 插入新记录 修改记录 删除记录）

设置表记录可以写中文

]# LANG="zh\_CN.UTF-8"

mysql> create table db1.学生表(

姓名 char(10),

家庭地址 char(20)

) DEFAULT CHARSET=utf8;

mysql> insert into db1.学生表 values("张三丰","武当山");

mysql> insert into db1.学生表 values("张四丰","武台山");

mysql> select \* from db1.学生表;

三、MySQL数据类型

3.1 字符类型 char varchar

3.2 数值类型 练习到 15：35

整数类型 tinyint smallint int bigint

浮点类型 float double

mysql> create table db1.t5 (name char(10), level tinyint );

mysql> insert into db1.t5 values ("zbj",-1);

mysql> insert into db1.t5 values ("zbj",-129);

mysql> insert into db1.t5 values ("zbj",198);

mysql> create table db1.t6 (name char(10), level tinyint unsigned);

mysql> insert into db1.t6 values ("zbj",-1);

mysql> insert into db1.t6 values ("zbj",0);

mysql> insert into db1.t6 values ("zbj",256);

mysql> insert into db1.t6 values ("zbj",25);

mysql> insert into db1.t6 values ("zbj",25.22);

mysql> insert into db1.t6 values ("zbj",25.52);

mysql> create table db1.t3(pay float , id double);

mysql> insert into db1.t3 values(9999.23,9999.77);

mysql> create table db1.t4( pay float(5,2) , id double(4,2) );

mysql> insert into db1.t4 values(9999.23,9999.77);

mysql> create table db1.t7(stu\_num int , name char(5) ,

age tinyint ,pay float ,money float(7,2) );

mysql> desc db1.t7 ;

练习到 15：45

3.3 日期时间类型

年 日期 时间 日期时间

year date time datetime/ timestamp

create table db1.t8( 休息到 16：15

name char(15),

your\_s year,

birthday date,

up\_class time,

party datetime

);

desc db1.t8 ;

insert into db1.t8 values

("bob",1990,20191120,083000,20190607203000);

当year类型 使用2位数赋值时：

01~69视为 2001~2069

70~99视为 1970~1999

insert into db1.t8 values

("tom",77,20191120,083000,20190607203000);

select \* from db1.t8 ;

当未给timestamp字段赋值时，自动以当前系统时间赋值，而datetime 值为 NULL (空)

create table db1.t9( 练习到 16:30

meetting datetime,

party timestamp

) ;

insert into db1.t9 values (20190606162000,20190606162000);

insert into db1.t9(meetting) values (20180707093000);

insert into db1.t9(party) values (20160707093000);

select \* from db1.t9;

可以使用时间函数给字段赋值

mysql> insert into db1.t8 values

( "jack" , year(now()) , date(now()) , curtime() , now() );

3.4 枚举类型

enum 单选

set 多选

create table db1.t10 (

name char(10),

sex enum("boy","girl","no"),

likes set("eat","game","piao","du","film")

);

mysql> desc db1.t10;

insert into db1.t10 values ("jack","man","a,b,c");

insert into db1.t10 values ("jack","boy","eat,piao");

select \* from db1.t10 ;

RDBMS1\_day01 内容总结

一、搭建的mysql数据库服务器：什么是数据库？ 数据库服务软件？

mysql优点？ 应用场景？ LAMP / LNMP

安装mysql软件 启动服务 使用初始密码登陆

修改登陆密码 设置密码策略

二、基本使用？ 把数据库存储到mysql服务器上的步骤 ？

连接方式？

库的基本管理命令 表的基本管理命令 记录管理

SQL命令使用规则

三、数据类型： 字符 数值 日期时间 枚举

一. 表结构

| Null | Key | Default | Extra |

|　　空 |　键值｜　默认值　｜额外设置｜

1.1 字段约束条件 (控制字段赋值)

２．设置约束条件

创建t7表格

mysql> create table db2.t7(

-> name char(15) not null,

-> age tinyint unsigned default 25,

-> class char(9) not null default "nsd1904",

-> sex enum("男","女") default "男",

-> pay float(7,2) default 28000

-> );

Query OK, 0 rows affected (0.59 sec)

mysql> desc t7;

+-------+---------------------+------+-----+----------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+---------------------+------+-----+----------+-------+

| name | char(15) | NO | | NULL | |

| age | tinyint(3) unsigned | YES | | 25 | |

| class | char(9) | NO | | nsd1904 | |

| sex | enum('男','女') | YES | | 男 | |

| pay | float(7,2) | YES | | 28000.00 | |

+-------+---------------------+------+-----+----------+-------+

5 rows in set (0.00 sec)

案例一:

mysql> create table db2.t2(

-> class char(9),

-> name char(10) not null default "",

-> age tinyint(4) not null default 19,

-> likes set("a","b","c","d") default "a,b");

Query OK, 0 rows affected (0.60 sec)

mysql> desc db2.t2;

+-------+----------------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+----------------------+------+-----+---------+-------+

| class | char(9) | YES | | NULL | |

| name | char(10) | NO | | | |

| age | tinyint(4) | NO | | 19 | |

| likes | set('a','b','c','d') | YES | | a,b | |

+-------+----------------------+------+-----+---------+-------+

4 rows in set (0.00 sec)

二．myself

普通索引index

唯一索引inique

主键primary key

外 键 foreign key \*\*\*

全文索引 fulltext

作用：约束如何给字段赋值

普通索引（index）

什么是索引（索引介绍）

为什么要使用索引（索引优点与缺点）

使用索引：

使用规则 ？

查看 desc 库.表 key

show index from 表名 \G;

创建索引:

建表时创建索引字段

create table db1.t22(

class char(9),

name char(15),

age int,

index(name),index(age)

);

desc db1.t22;

show index from db1.t22\G;

在已有表里创建索引字段

mysql> create index xixi

on stuinfo(name);

删除索引:

mysql> drop index name on t22;

2.2 主 键 primary key

作用：限制字段赋值

使用规则？

使用主键：

1 创建主键

1.1 建表时创建主键

mysql> create table db1.t23(

-> name char(10) primary key,

-> age int,

-> pay float(7,2)

-> );

mysql> desc db1.t23 ;

mysql> insert into db1.t23 values(null,19,28800.88);

mysql> insert into db1.t23 values("bob",19,28800.88);

mysql> insert into db1.t23 values("bob",29,38800.88);

1.2 在已有表里添加主键

mysql> select \* from db1.classinfo;

mysql> delete from db1.classinfo;

mysql> alter table db1.classinfo add primary key(name);

mysql> desc db1.classinfo;

练习到 14：45

2 创建复合主键

约束方式： 主键字段的值不可以同时重复

PRI PRI

clientip serport status

1.1.1.1 22 yes

1.1.1.1 22 yes

create table db1.t24(

clientip char(15),

serport int,

status enum("yes","no"),

primary key(clientip,serport)

);

mysql> insert into db1.t24 values ("1.1.1.1",22,"yes");

mysql> insert into db1.t24 values ("1.1.1.1",22,"no");

mysql> insert into db1.t24 values ("1.1.1.1",22,"yes");

mysql> insert into db1.t24 values ("1.1.1.1",25,"yes");

mysql> insert into db1.t24 values ("1.1.1.1",25,"no");

mysql> insert into db1.t24 values ("2.1.1.1",25,"no");

3 primary key与auto\_increment连用

自增长

赋值时 字段的值字段加1

id name age sex

1 bob 19 boy

2 tom 21 boy

3 jerry 25 boy

4 lucy 18 girl

create table db1.t25(

id int primary key auto\_increment,

name char(15),

age tinyint ,

sex enum("boy","girl")

);

mysql> desc db1.t25 ;

mysql> insert into db1.t25(name,age,sex)values("bob",19,"boy");

mysql> insert into db1.t25(name,age,sex)values("tom",21,"boy");

mysql> select \* from db1.t25;

mysql> insert into db1.t25 values (6,"bob",19,"girl");

mysql> select \* from db1.t25;

mysql> insert into db1.t25(name,age,sex)values("jack",27,"boy");

mysql> select \* from db1.t25;

mysql> delete from db1.t25;

mysql> select \* from db1.t25;

mysql> insert into db1.t25(name,age,sex)values("jack",27,"boy");

mysql> select \* from db1.t25;

4 删除主键

mysql> alter table db1.t23 drop primary key;

mysql> alter table db1.t24 drop primary key;

mysql> alter table db1.t25 modify id int;

mysql> alter table db1.t25 drop primary key;

2.3 外 键 foreign key

作用：限制给字段赋值，插入记录时，

字段值在另一个表字段值范围内选择。

使用规则:

1 表存储引擎必须是innodb

2 字段类型要一致

3 被参照字段必须要是键值类型的某一种，通常是p rimary key

命令格式：

create table 库.表（

字段列表，

foreign key(字段名) references 表名(字段名)

on update cascade

on delete cascade

)engine=innodb;

yg 员工表

编号 姓名

1 bob

2 tom

3 bob

create table db1.yg(

yg\_id int primary key auto\_increment,

name char(20)

)engine=innodb;

insert into db1.yg(name)values("bob");

insert into db1.yg(name)values("tom");

insert into db1.yg(name)values("bob");

select \* from db1.yg;

gz 工资表

编号 工资

1 30000

2 50000

create table db1.gz(

gz\_id int,

pay float(7,2),

foreign key(gz\_id) references yg(yg\_id)

on update cascade on delete cascade

)engine=innodb;

mysql> desc db1.gz;

mysql> show create table db1.gz\G;

mysql> select \* from db1.gz;

测试外键：

mysql> insert into db1.gz values(4,50000);

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`db1`.`gz`, CONSTRAINT `gz\_ibfk\_1` FOREIGN KEY (`gz\_id`) REFERENCES `yg` (`yg\_id`) ON DELETE CASCADE ON UPDATE CASCADE)

mysql> insert into db1.gz values(1,50000);

mysql> insert into db1.gz values(2,45000);

mysql> insert into db1.gz values(3,35000);

mysql> select \* from db1.gz;

练习 到 17：52

mysql> update db1.yg set yg\_id=4 where yg\_id=3;

mysql> select \* from db1.yg;

mysql> select \* from db1.gz;

mysql> delete from db1.yg where yg\_id=1;

mysql> select \* from db1.yg;

mysql> select \* from db1.gz;

mysql> delete from db1.gz;

mysql> alter table db1.gz add primary key(gz\_id);

删除外键

mysql> show create table db1.gz\G;

mysql> alter table db1.gz drop foreign key gz\_ibfk\_1;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DBA1-DAY03\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.数据导入导出

检索目录

查看默认检索目录

show variables like "secure\_file\_priv";

修改检索目录

[root@mysql50 ~]# mkdir /myload

[root@mysql50 ~]# chown mysql /myload/

[root@mysql50 ~]# vim /etc/my.cnf

添加

secure\_file\_priv="/myload"

[root@mysql50 ~]# systemctl restart mysqld

数据导入:把系统文件的内容储存到表里

建表db3.user

mysql> desc user;

+----------+-----------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+-----------+------+-----+---------+-------+

| name | char(50) | YES | | NULL | |

| password | char(1) | YES | | NULL | |

| uid | int(11) | YES | | NULL | |

| gid | int(11) | YES | | NULL | |

| comment | char(150) | YES | | NULL | |

| homedir | char(150) | YES | | NULL | |

| shell | char(100) | YES | | NULL | |

+----------+-----------+------+-----+---------+-------+

7 rows in set (0.00 sec)

导入数据

mysql> load data infile "/myload/passwd" into table db3.user fields terminated by ":" lines terminated by "\n";

添加ID字段更易查看

mysql> alter table db3.user add id int primary key auto\_increment first;

数据导出:把表记录保存到系统文件里

命令 mysql> SQL查询命令 into outfile "目录/文件名"; //基本格式

不指定以什么分割:

mysql> select \* from user where id<=5 into outfile "/myload/user.txt";

打开导出的文件:

mysql> system cat /myload/user.txt;

1 root x 0 0 root /root /bin/bash

2 bin x 1 1 bin /bin /sbin/nologin

3 daemon x 2 2 daemon /sbin /sbin/nologin

4 adm x 3 4 adm /var/adm /sbin/nologin

5 lp x 4 7 lp /var/spool/lpd /sbin/nologin

以"###"分割列,"?"分隔行导出文件:

mysql> select \* from user where id<=5 into outfile "/myload/user2.txt"

>fields terminated by "###"

>lines terminated by "?";

打开导出的文件:

mysql> system cat /myload/user2.txt;

1###root###x###0###0###root###/root###/bin/bash?2###bin###x###1###1###bin###/

bin###/sbin/nologin?3###daemon###x###2###2###daemon###/sbin###/sbin/nologin?

4###adm###x###3###4###adm###/var/adm###/sbin/nologin?

5###lp###x###4###7###lp###/var/spool/lpd###/sbin/nologin?mysql>

#######################################################################################

2.管理表记录

插入表记录 insert into 表名 values(字段值列表);

全部插入一次:

mysql> insert into db3.user

- > values

- >(22,"bob","x",2000,2000,"test user","/home/test","/bin/bash");

全部插入多次:

mysql> insert into db3.user

- > values

- >(22,"bob","x",2000,2000,"test user","/home/test","/bin/bash");

- >(23,"haha","x",2001,2001,"test user","/home/test","/bin/bash");

- >(24,"dcc","x",2002,2002,"test user","/home/test","/bin/bash");

个别字段插入一次:

mysql> insert into db3.user

- > values(name)

- > ("licy");

个别字段插入多次:

mysql> insert into db3.user

- > values(name)

- > ("jeck"),

- > ("licy");

查询表记录 select \* from 库.表 where 条件表达式;

mysql> select name,shell from db3.user where id <=2;

mysql> select \* from db3.user where id<=2;

更新表记录 update 库.表 set 字段名=值 where 条件表达式;

mysql> update db3.user set password="a",comment="student"; //批量修改

mysql> update db3.user set password="a",comment="student" //按条件修改

-> where id=11;

删除表记录 delete from 库.表;

mysql> delete from db3.user where id>=25; //将大于等于25的行删除

mysql> delete from db3.user where id=18; //将18行删除

mysql> delete from db3.user; //将表删除

####################################################################################

3.匹配条件

3.1 基本匹配条件 (select,update,delete)

数值比较

= > >= < <= !=(不等于) //比较数值类型

mysql> select name from db3.user where id>15; //输出id大于15的名字

mysql> select name from db3.user where uid=gid; //输出uid等于gid的名字

mysql> select name,uid from db3.user where id<10; //输出id大于15的名字,uid

字符比较

= 相等 != 不相等 is null 空 is not null 非空

mysql> insert into db3.user(name) values(null),(NULL),(""),("null");//添加字段

mysql> select name,uid from db3.user where shell !="/sbin/nologin";

mysql> select name from db3.user where name = comment; //查询name等于comment字段的

mysql> select name,uid from db3.user where shell is not null; //非空查询

mysql> select id,name from db3.user where name is null; //查询空值

逻辑匹配(多个判断条件) (或且非)

逻辑与 and 或&& 多个条件同时成立

逻辑或 or 或 || 多个条件某一个条件成立即可

逻辑非 ! = 取反

mysql> select name,uid from db3.user where name="root" and uid=0; //uid和name都相等的

mysql> select name,uid from db3.user where name="root" or uid=1; //uid为1,或name为root

范围内匹配

in(值列表) 在....里....

mysql> select name from db3.user where name in ("root","sync","bin"); //查询name在root,sync,bin之间

not in 不在....里...

mysql> select name from db3.user where name not in ("root","sync","bin");

//查询name不在root,sync,bin之间

between数字and数字 在...之间...

mysql> select \* from db3.user where uid between 10 and 20;//在10到20之间

distinct 字段名 去重显示

mysql> select distinct shell from db3.user;//去重

3.2 高级匹配条件

模糊查询 like

where 字段名 like '通配符' \_ 表示一个字符 % 表示0~n个字符

mysql> select name from db3.user where name like "\_\_\_\_"; 匹配四个字符的

mysql> select name from db3.user where name like "%a%"; //包含a

mysql> select name from db3.user where name like "\_\_%\_\_"; //最少4个字符

正则表达式 regexp

where 字段名 regexp '正则表达式' ^ $ . [ ] \*　　　｜

mysql> select name from db3.user where name regexp '^r'; //以r开头

mysql> select name from db3.user where name regexp '^r.\*t$'; //以r开头,t结尾

mysql> select name from db3.user where name regexp '^r.\*t$'; //以r开头或者t结尾

四则运算(算术运算)

+ - \* / % () 加 减 乘 除 求模(取余) 提高优先级

添加字段:

mysql> alter table db3.user

-> add system int default 80,

-> add mysql int default 80,

-> add age tinyint default 18 after name;

mysql> select name,age, 2019 - age csnf from db3.user where name="root"; //root出生年份

mysql> select name,uid from db3.user where uid%2 !=0;//uid为奇数

mysql> select name,uid from db3.user where uid%2 =0;//uid为偶数

mysql> select name,uid from db3.user where uid < 10;

mysql> update db3.user set uid=uid+1 where uid < 10; //批量修改

mysql> select name,uid from db3.user where uid < 10;

mysql> select name ,system,mysql,system+mysql zfs,(system+mysql)/2 pzf from db3.user where name="root"; //算平均分

3.3 操作查询结果

1 聚集函数 (MySQL服务内置对数据做统计的命令)

avg(字段名) //平均值

sum(字段名) //求和

min(字段名) //最小值

max(字段名) //最大值

count(字段名) //值的个数

mysql> select max(uid) from db3.user; //格式

mysql> select sum(uid) from db3.user where id>10; //id大于10的uid之和

2 查询结果排序 order by 字段名 asc|desc; (升序|降序)

mysql> select name , uid from db3.user where uid >=10 and uid<=1000; //对比看结果

mysql> select name , uid from db3.user where uid >=10 and uid<=1000 order by uid; //默认升序

mysql> select name , uid from db3.user where uid >=10 and uid<=1000 order by uid desc; //降序

3 查询结果分组 group by 字段名;

mysql> select shell from db3.user;

mysql> select shell from db3.user group by shell; //有空值时,和去重输出不一样

mysql> select shell from db3.user where uid<=500;

mysql> select shell from db3.user where uid<=500 group by shell; //有空值时,和去重输出不一样

mysql> select shell from db3.user group by shell; //输出空值

4 查询结果过滤

mysql> select name from db3.user where shell!="/bin/bash"

-> having name ="bin"; //在输出结果里在过滤一遍

5 限制查询结果显示行数 limit 数字; limit 数字1,数字2; (起始行,总行数)

mysql> select \* from db3.user where uid <=15; //默认输出所有航

mysql> select \* from db3.user where uid <=15 limit 3; //只显示前三行

mysql> select \* from db3.user where uid <=15 limit 3,6; //从第四行开始显示6行( 0是第一行 )

4.MySQL管理工具

常用工具

mysql(命令行) MySQL-Workbench(图形) MySQLFront(图形) Navicat(图形) phpMyAdmin(浏览器)

1. 部署LAMP

2. 启动httpd服务

3.解压phpmyadmin包,移动到html目录

4.配置配置文件,指定服务器地址

5.浏览器访问

yum -y install httpd php php-mysql

tar -zxvf phpmyadmin

mv phpmyadmin /var/www/html

cp config.sample.inc.php config.inc.php

vim config.inc.php

17 $cfg['blowfish\_secret'] = 'dc';(等号后面随便填,不能为空)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DBA1\*DAY04\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

select user(); //显示当前登录用户 show grants; 显示当前登录用户权限

一: 用户授权

1.1 授权 : 添加用户并设置权限

1.2 格式

添加权限:

grant 权限列表 on 库名 to 用户名@"客户端地址" identified by "密码" with grant option;

all 所有权限 usage 无权限 select,update,insert 个别权限 select,update(字段1,字段2) 指定字段

\*.\* //所有库所有表 库名.\* // 一个库 库名.表名 //一张表

mysql> grant all on \*.\* to mydba@"%" identified by "123456" with grant option; //添加用户mydba对所有库所

有表有所有权限,并添加可授权权限

修改密码:

mysql> set password =password("123456"); //修改自己密码

mysql> set password for userone@"localhost" =password("111111"); //管理员修改其他人密码

1.3 授权信息 mysql(授权信息)

user表 已授权用户及权限表

db表 用户对数据库的权限

table\_priv表 已授权用户对表的访问权限

columns\_priv表 已授权用户对字段的访问权限

mysql> update mysql.db set Delete\_priv="N" where db="db3" and user="admin2" and host="localhost"\G

//可通过修改文件修改权限

mysql> flush privileges; //刷新

1.4 删除用户

mysql> drop user yaya99@"%"; //删除用户(用户名@"客户端地址")

1.5 撤销权限

mysql> show grants for mydba@"%";

+---------------------------------------------------------------------------------------------------------------+

| Grants for mydba@% |

+---------------------------------------------------------------------------------------------------------------+

| GRANT ALL PRIVILEGES ON \*.\* TO 'mydba'@'%' WITH GRANT OPTION |

+---------------------------------------------------------------------------------------------------------------+

mysql> revoke grant option on \*.\* from mydba@"%"; //撤销用户mydba 的授权权限

mysql> show grants for mydba@"%";

+------------------------------------------------------------------------------+

| Grants for mydba@% |

+-------------------------------------------------------------------------------+

| GRANT ALL PRIVILEGES ON \*.\* TO 'mydba'@'%' |

+------------------------------------------------------------------------------+

mysql> revoke all on \*.\* from mydba@"%"; //删除所有权限

Query OK, 0 rows affected (0.00 sec)

mysql> show grants for mydba@"%";

+--------------------------------------------------------------+

| Grants for mydba@% |

+--------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO 'mydba'@'%' |

+---------------------------------------------------------------+

1.6 恢复root密码

vim /etc/my.cnf

[mysqld]

skip-grant-tables //添加

systemctl restart mysqld //重启MySQL

mysql> select user,host,authentication\_string from mysql.user; //查看

mysql> update mysql.user set authentication\_string=password("aaa123...A")

-> where user="root" and host="localhost"; //修改密码

mysql> flush privileges; //刷新,然后重启,用新密码登录

1.7 修改数据库管理员root用户本机登录密码

[root#] mysqladmin -uroot -p password "123456" //输入后需要输入原密码

################################################################################################

二: 完全备份

物理备份

]# mkdir /dbbak

]# cp -r /var/lib/mysql /dbbak/mysql.bak

]# scp -r /dbbak/mysql.bak root@192.168.4.51:/root/

物理恢复

]# systemctl stop mysqld

]# rm -rf /var/lib/mysql

]# cp -r /root/mysql.bak /var/lib/mysql

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

]# systemctl start mysqld

]# mysql -uroot -p123456

三: 增量备份

binlog日志的使用

1.binlog日志介绍

2.启用日志

[root@mysql50 mysql]# vim/etc/my.cnf

log-bin //添加字段

server\_id=50

:wq

[root@mysql50 mysql]# systemctl restart mysqld

[root@mysql50 mysql]# ls /var/lib/mysql/\*-bin.\* //查看日志位置

[root@mysql50 mysql]# mysql -uroot -p123456 -e "flush logs;" //重启一次生成一个日志(-e在命令行执行数据库命令)

3.手动创建日志文件(三种方法)

1: systemctl restart mysqld

2: mysql -uroot -p123456 -e "show databases;"

mysql -uroot -p123456 -e "flush logs;"

3: mysql -uroot -p123456

mysql> flush logs;

删除已有文件(两种方法)

mysql> purge master logs to "mysql50-bin.000004";

mysql> reset master

4.修改日志格式

[root@mysql50 mysql]# vim/etc/my.cnf

[mysqld]

binlog\_format="mixed" //添加

:wq

[root@mysql50 mysql]# systemctl restart mysqld

[root@mysql50 mysql]# mysql -uroot -p123456

mysql> show variables like "binlog\_format";

mysql> show master status;

5.查看日志内容

mysql> show master status;

[root@mysql50 ~]# mysqlbinlog /mylog/db50.000001 | grep -i create

6.修改日志保存位置

[root@mysql50 ~]# mkdir /mylog

[root@mysql50 ~]# chown mysql /mylog

[root@mysql50 ~]# vim /etc/my.cnf

[mysqld]

log-bin=/mylog/db50 //db50是日志名字

:wq

[root@mysql50 ~]# systemctl restart mysqld

7. 恢复数据

[root@mysql50 ~]# scp /mylog/db50.000001 root@192.168.4.51:/root/ //拷贝给目标主机

[root@mysql51 ~]# mysqlbinlog /root/db50.000001 | mysql -uroot -p123456 //恢复数据,打开日志文件管道给登录系

统命令就可恢复数据,等于重新执行一遍

[root@mysql51 ~]# mysqlbinlog --start-position=2232 --stop-position=3656 /root/db50.000001 | mysql -

uroot -p123456 //恢复指定偏移量 (2232到3656) 的数据

命令格式

mysqlbinlog 日志文件名 | mysql -uroot -p密码

mysqlbinlog 选项 日志文件名 | mysql -uroot -p密码

选项 用途

--start-position数字 起始偏移量

--stop-position数字 结束偏移量

--start-datetime="yyyy-mm-dd hh:mm:ss" 起始时间

--stop-datetime="yyyy-mm-dd hh:mm:ss" 结束时间

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DBA\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

数据备份与恢复

rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

yum -y install percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

rpm -ql percona-xtrabackup-24

1. 使用innobackupex 对数做备份与恢复

1.1命令格式

[root@mysql50 ~]# innobackupex <选项>

1.2常用选项

--user --password --no-timestamp --apply-log --copy-back

用户 密码 不用日期命名备份文件储存的子目录文件名 准备恢复数据 拷贝备份文件,恢复数据

--export import

导出 导 入

1.3完全备份与恢复

[root@mysql50 ~]# innobackupex --user root --password 123456 /allbak --no-timestamp //完全备份

[root@mysql50 ~]# ls /allbak

[root@mysql50 ~]# scp -r /allbak/ root@192.168.4.51:/root/ //拷贝到51机器

[root@mysql51 ~]# rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

[root@mysql51 ~]# yum -y install percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

[root@mysql51 ~]# innobackupex --apply-log /root/allbak/ //准备恢复

[root@mysql51 ~]# cat /root/allbak/xtrabackup\_checkpoints //打开文件查看

[root@mysql51 ~]# innobackupex --copy-back /root/allbak/ //执行拷贝命令,将数据拷贝到mysql文件夹下

[root@mysql51 ~]# ls -l /var/lib/mysql //查看文件属性

[root@mysql51 ~]# chown -R mysql:mysql /var/lib/mysql //修改文件权限

[root@mysql51 ~]# systemctl start mysqld //启动服务进行验证

1.4恢复完全备份中的单张表

删除表空间 mysql> alter table db5.b discard tablespace;

导出表信息 [root@mysql50 ~]# innobackupex --apply-log --export /allbak

拷贝表信息文件到数据库目录下 [root@mysql50 ~]# cp /allbak/db5/b.{cfg,exp,ibd} /var/lib/mysql/db5/

修改表信息文件的所有者及组用户为mysql [root@mysql50 ~]# chown mysql:mysql /var/lib/mysql/db5/b.\*

导入表空间 mysql> alter table db5.b import tablespace;

删除数据库目录下的表信息文件 mysql> system rm -rf /var/lib/mysql/db5/b.{cfg,exp};

查看表记录 mysql> desc db5.b; mysql> select \* from db5.b;

1.5增量备份与恢复

增量备份:备份上次备份后所产生的新数据

首备份数据(完全备份) [root@mysql50 ~]# innobackupex --user root --password 123456 /allbak --no-timestamp

第二次备份(增量备份) [root@mysql50 ~]# innobackupex --user root --password 123456 --incremental /new2dir --

incremental-basedir=/datefull --no-timestamp

第三次备份(增量备份) [root@mysql50 ~]# innobackupex --user root --password 123456 --incremental /new3dir --

incremental-basedir=/new2dir --no-timestamp

增量恢复:

RDBMS1知识点总结:

数据库服务介绍

搭建mysql数据库服务器

数据库服务基本使用: 登录 断开 建表 建库 记录管理

mysql数据类型

修改表结构 mysql键值(index primary key 复合主键 外键)

对数据进行操作: 数据导入导出,记录的增删改查 匹配条件

用户授权 root密码 图形工具phpmyadmin

数据的备份与恢复: 物理备份(cp tar innobackupex) 逻辑备份(mysqldump binlog)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DBA\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

数据备份与恢复

rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

yum -y install percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

rpm -ql percona-xtrabackup-24

1. 使用innobackupex 对数做备份与恢复

1.1命令格式

[root@mysql50 ~]# innobackupex <选项>

1.2常用选项

--user --password --no-timestamp --apply-log --copy-back

用户 密码 不用日期命名备份文件储存的子目录文件名 准备恢复数据 拷贝备份文件,恢复数据

--export import

导出 导 入

1.3完全备份与恢复

[root@mysql50 ~]# innobackupex --user root --password 123456 /allbak --no-timestamp //完全备份

[root@mysql50 ~]# ls /allbak

[root@mysql50 ~]# scp -r /allbak/ root@192.168.4.51:/root/ //拷贝到51机器

[root@mysql51 ~]# rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

[root@mysql51 ~]# yum -y install percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

[root@mysql51 ~]# innobackupex --apply-log /root/allbak/ //准备恢复

[root@mysql51 ~]# cat /root/allbak/xtrabackup\_checkpoints //打开文件查看

[root@mysql51 ~]# innobackupex --copy-back /root/allbak/ //执行拷贝命令,将数据拷贝到mysql文件夹下

[root@mysql51 ~]# ls -l /var/lib/mysql //查看文件属性

[root@mysql51 ~]# chown -R mysql:mysql /var/lib/mysql //修改文件权限

[root@mysql51 ~]# systemctl start mysqld //启动服务进行验证

1.4恢复完全备份中的单张表

删除表空间 mysql> alter table db5.b discard tablespace;

导出表信息 [root@mysql50 ~]# innobackupex --apply-log --export /allbak

拷贝表信息文件到数据库目录下 [root@mysql50 ~]# cp /allbak/db5/b.{cfg,exp,ibd} /var/lib/mysql/db5/

修改表信息文件的所有者及组用户为mysql [root@mysql50 ~]# chown mysql:mysql /var/lib/mysql/db5/b.\*

导入表空间 mysql> alter table db5.b import tablespace;

删除数据库目录下的表信息文件 mysql> system rm -rf /var/lib/mysql/db5/b.{cfg,exp};

查看表记录 mysql> desc db5.b; mysql> select \* from db5.b;

1.5增量备份与恢复

增量备份:备份上次备份后所产生的新数据

首备份数据(完全备份) [root@mysql50 ~]# innobackupex --user root --password 123456 /allbak --no-timestamp

第二次备份(增量备份) [root@mysql50 ~]# innobackupex --user root --password 123456 --incremental /new2dir --

incremental-basedir=/datefull --no-timestamp

第三次备份(增量备份) [root@mysql50 ~]# innobackupex --user root --password 123456 --incremental /new3dir --

incremental-basedir=/new2dir --no-timestamp

增量恢复:

RDBMS1知识点总结:

数据库服务介绍

搭建mysql数据库服务器

数据库服务基本使用: 登录 断开 建表 建库 记录管理

mysql数据类型

修改表结构 mysql键值(index primary key 复合主键 外键)

对数据进行操作: 数据导入导出,记录的增删改查 匹配条件

用户授权 root密码 图形工具phpmyadmin

数据的备份与恢复: 物理备份(cp tar innobackupex) 逻辑备份(mysqldump binlog)

# NoSQL

## day01

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NoSQL\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NoSQL

-Not Only SQL不仅仅是数据库

-泛指非关系型数据库

-不需要预先定义数据库存储结构

-每条记录可以有不同的数据类型和字段个数

NoSQL软件

Memcached Redis MongoDB CouchDB Neo4j FlockDB

RDBMS软件

MySQL MariaDB Oracle DB2 SQL Server

一: 部署redis 软件 (redis 高性能分布式内存数据库)

1. redis介绍

2. 安装redis

yum -y install gcc

tar -zxf redis-4.0.8.tar.gz //一定要先安装gcc在解压

make && makeinstall

3. 初始化配置redis

./utils/install\_server.sh //执行初始化设置脚本

Please select the redis port for this instance: [6379] //端口号

Please select the redis config file name [/etc/redis/6379.conf] //主配置文件路径及名称

Please select the redis log file name [/var/log/redis\_6379.log] //日志文件路径及名称

Please select the data directory for this instance [/var/lib/redis/6379] //数据库目录

Please select the redis executable path [/usr/local/bin/redis-server] //启动程序目录

Port : 6379 //端口号

Config file : /etc/redis/6379.conf //配置文件目录

Log file : /var/log/redis\_6379.log //日志目录

Data dir : /var/lib/redis/6379 //数据库目录

Executable : /usr/local/bin/redis-server //启动程序的目录

Cli Executable : /usr/local/bin/redis-cli //命令行的连接工具

Is this ok? Then press ENTER to go on or Ctrl-C to abort. //回车完成配置

Copied /tmp/6379.conf => /etc/init.d/redis\_6379 //服务启动脚本

4. 服务管理

/etc/init.d/redis\_6379 start //启动服务

/etc/init.d/redis\_6379 stop //停止服务

/etc/init.d/redis\_6379 status //查看服务状态

/etc/init.d/redis\_6379 restart //重新启动

netstat -antupl |grep :6379 //查看端口

5. 连接服务,存取数据

redis-cli //默认连接127.0.0.1地址的 6379端口

set key名 key值 //储存一个key值

mset key名 key值 .... //储存多个key值

get key名 //获取key值

mget key名 ... //获取多个key值

keys \* //显示所有key名

type key名 //查看key类型

exists key名 //测试key名是否存在

ttl key名 //查看key生存时间(-2已过期 -1 永久存在,单位为秒)

expire key名 数字 //设置key有效时间

move key名 库编号 //移动key到指定库

select 数据库编号0-15 //切换库

del key名 //删除指定key

flushdb //删除所在库的所有key

flushall //删除内存所有key

save //保存所有key到硬盘

shutdown //停止服务

二:

## day02

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NoSQL\*DAY02\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 :集群环境准备

在51-56设置集群　6台redis服务器

　 /etc/init.d/redis\_6379 stop

　　 vim /etc/redis/6379.conf

　　 70 bind 192.168.4.51

　　　 93 port 6351

　　　 501 # requirepass foobared　　//密码

　　　 815 cluster-enabled yes　　//启用集群功能

　　　 823 cluster-config-file nodes-6379.conf　　//储备集群信息文件

　　　 829 cluster-node-timeout 5000　　//链接超时服务　　5秒

　　 rm -rf /var/lib/redis/6379/＊　　　//删除所有相关原来的东西

　　 vim +43 /etc/init.d/redis\_6379　　//修改脚本

　　 $CLIEXEC -p -h 192.168.4.51 -p 6351 shutdown

/etc/init.d/redis\_6379 start　　　　//重新启动

redis-cli -h 192.168.4.51 -p 6351

哈希池　0-16383　分3个大块　每个大块一主一从

0-5460 5461-10922 10923-16383

二 : 部署redis集群

1.配置管理主机51

部署ruby脚本运行环境

yum -y install rubygems //安装依赖包

rpm -q ruby

scp /linux-soft/03/redis/redis-3.2.1.gem root@192.168.4.51:/root/ //拷贝gem包

gem install redis-3.2.1.gem //安装

1 gem installed

创建集群管理脚本

cd redis-4.0.8/src/

mkdir /root/bin

cp redis-trib.rb /root/bin/

chmod +x redis-trib.rb

redis-trib.rb help

2. 创建集群

vim /etc/redis/6379.conf

815 cluster-enabled yes　　//启用集群功能

　　　 823 cluster-config-file nodes-6379.conf　　//储备集群信息文件

　　　 829 cluster-node-timeout 5000　　//链接超时服务　　5秒

　　　 /etc/init.d/redis\_6379 start

redis-trib.rb create --replicas 1 192.168.4.51:6351 192.168.4.52:6352 192.168.4.53:6353 192.168.4.54:6354 192.168.4.55:6355 192.168.4.56:6356

[OK] All 16384 slots covered.

redis-trib.rb create info //查看集群信息

192.168.4.53:6353> cluster info //查看集群信息

192.168.4.53:6353> cluster nodes //查看集群节点信息

redis-trib.rb check 192.168.4.52:6352 //检查集群主机角色

3. 访问集群

集群存取数据工作原理:

变量名和crc16算法做hash计算,得到一个数字 ,和16384取余,根据余数得值,找对应的redis服务器

redis-cli -c -h 192.168.4.52 -p 6352

redis-cli -c -h ip地址 -p 端口号 选项 -c 集群模式

三 : 管理redis集群

3.1 测试集群高可用功能

关闭53服务,启动53服务

redis-trib.rb check 192.168.4.56:6356

把主服务器的redis服务停止,对应的从服务器会自动升级为主服务器.

宕机的主服务器redis服务重新启动后,自动做当前主服务器的从库,并自动同步宕机期间产生的新数据

3.2 向集群中添加新主机

3.2.1 添加master角色主机 (管理主机51操作)

redis-trib.rb add-node 192.168.4.58:6358 192.168.4.53:6353 //53为master

redis-trib.rb info 192.168.4.53:6353 //查看是否添加成功

redis-trib.rb check 192.168.4.53:6353 //检测集群

redis-trib.rb reshard 192.168.4.53:6353

How many slots do you want to move (from 1 to 16384)?4096 //拿出4096个hash 槽给主机192.168.4.58

What is the receiving node ID? c5e0da48f335c46a2ec199faa99b830f537dd8a0 //主机192.168.4.58的id值

Source node #1:all //从当前所有master服务器获取hash槽

Do you want to proceed with the proposed reshard plan (yes/no)?yes //同意以上配置

redis-trib.rb info 192.168.4.53:6353

3.2.2 添加slave角色主机

redis-trib.rb add-node --slave 192.168.4.59:6359 192.168.4.51:6351 //执行添加命令

Configure node as replica of 192.168.4.58:6358. //提示添加完成

redis-trib.rb info 192.168.4.51:6351 //查看信息

redis-trib.rb check 192.168.4.53:6353 //检测集群

3.3 从集群中移除主机

3.3.1 移除slave角色主机

redis-trib.rb del-node 192.168.4.57:6357 d9fcd44a06e9661ff033c86bc1a8b071d1919c4c //执行移除命令

>>> SHUTDOWN the node. //停止移除服务的Redis服务

3.3.2 移除master角色主机

释放占用的hash槽

redis-trib.rb reshard 192.168.4.58:6358

How many slots do you want to move (from 1 to 16384)?4096 //移除4096个数槽

What is the receiving node ID? bc5c4e082a5a3391b634cf433a6486c867cfc44b

//要移动给谁的id即目标主机（这里可以随机写一个master的ID）

Source node #1: c5e0da48f335c46a2ec199faa99b830f537dd8a0 //从谁那移动即源主机,要删除主机的id

Source node #2: done //设置完毕

Do you want to proceed with the proposed reshard plan (yes/no)?yes //提交

移除主机

redis-trib.rb del-node 192.168.4.58:6358 43f844d7ff020a7650881ded2c1747a29e1bbe61

>>> Removing node 43f844d7ff020a7650881ded2c1747a29e1bbe61 from cluster 192.168.4.58:6358

>>> Sending CLUSTER FORGET messages to the cluster...

>>> SHUTDOWN the node. //关闭主机redis服务

把移除的redis主机添加到集群

1 启动redis服务(要添加的主机)

/etc/init.d/redis\_6379 start

2 重置集群信息(要添加的主机)

redis-cli -h 192.168.4.57 -p 6357

192.168.4.57:6357> cluster reset

3 添加主机到集群(管理主机)

[root@mysql51 ~]# redis-trib.rb add-node 192.168.4.57:6357 192.168.4.52:6352

4 查看集群信息

[root@mysql51 ~]# redis-trib.rb info 192.168.4.56:6356

还原操作:

停止服务

rm -rf /var/lib/redis/6379/\*

注释集群中的集群项

815 823 829

启动服务

## day03

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NoSQL\*DAY03\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : 主从复制(需要关闭集群) //主从同步加哨兵服务可以实现高可用,优点使用服务器少,两台就可以

1.1 主从复制结构模式

1.1.1一主一从

主服务器: 接收客户端访问

从服务器: 自动同步主服务器数据到本地

把主机52 配置为51的从服务器

redis-cli -h 192.168.4.51 -p 6351

192.168.4.51:6351> info replication //查看复制信息,默认都是主服务器

192.168.4.52:6352> slaveof 192.168.4.51 6351 //52设置为51的从服务器,临时有效,立即生效

[root@mysql52 ~]# vim /etc/redis/6379.conf

slaveof 192.168.4.51 6351 //永久有效

:wq

客户端50 连接51 存储数据 > set x 111

客户端50 连接52 查看数据 > keys \*

1.1.2一主多从

把主机53 也配置为51的从服务器

192.168.4.53:6353> slaveof 192.168.4.51 6351

在51主机查看复制信息 > 192.168.4.51:6351> info replication (从库个数是两个)

客户端50 连接51 存储数据 > set x 111

客户端50 分别连接从服务器 查看数据 > keys \*

1.1.3主从从

把主机54 配置为53的从服务器

192.168.4.54:6354> slaveof 192.168.4.53 6353 //把主机54 配置为53的从服务器

192.168.4.53:6353> info replication //查看复制信息

192.168.4.54:6354> info replication

客户端50 连接51 存储数据 > set x 111

客户端50 分别连接52/53/54 查看数据 > keys \*

1.1.4把从服务器恢复为独立的数据库服务器

临时恢复 > slaveof no one

192.168.4.53:6353> info replication //查看复制信息

永久恢复

[root@mysql54 ~]# vim /etc/redis/6379.conf

# slaveof 192.168.4.53 6353

:wq

[root@mysql54 ~]# /etc/init.d/redis\_6379 stop

[root@mysql54 ~]# /etc/init.d/redis\_6379 start

1.1.5配置带验证(主服务器带密码)的主从复制

给主服务器51设置连接密码

vim /etc/redis/6379.conf //修改主配置文件

501 requirepass 123456

vim +43 /init.d/redis\_6379 //修改脚本,重启服务

配置从服务器,指定主服务器的连接密码

(从服务器) vim /etc/redis/6379.conf

289 masterauth 123456

:wq

重启redis服务

info replication //查看服务状态

二 : 哨兵服务

2.1 部署一主一从服务

2.2 配置主配置文件

从服务器 viom /etc/sentinel.conf

bind 0.0.0.0 //服务地址

port 26379 //连接端口

sentinel monitor server51 192.168.4.51 6351 1

//主机名 //ip地址 //端口 //票数

sentinel auth-pass server51 123456 //连接密码

2.3 启动哨兵服务

[root@mysql52 redis-4.0.8]# redis-sentinel /etc/sentinel.conf //启动

2.4 测试

关闭主服务器

2.5 在从服务器查看复制信息(没密码的话,另一台从服务器自动做新主库从库)

info replication

2.6 查看哨兵服务的主配置文件,自动监视当前的主服务器

三 : 持久化(redis服务把内存里的数据 保存到硬盘的方式)

3.1 RDB

3.1.1 介绍

redis主配置文件 dbfilename "dump.rdb" //备份文件名

- save 300 10 //5分钟(300秒)且有10个key改变

save //阻塞写存盘

bgsave //不阻塞写存盘

3.1.2 使用RDB恢复数据

要备份的主机 :

192.168.4.50:6350> keys \*

1) "x3"

2) "x1"

192.168.4.50:6350> save

[root@mysql50 ~]# ls /var/lib/redis/6379/

dump.rdb

[root@mysql50 6379]# scp /var/lib/redis/6379/dump.rdb root@192.168.4.55:/root/

要恢复的主机 :

[root@mysql55 ~]# /etc/init.d/redis\_6379 stop

[root@mysql55 ~]# rm -rf /var/lib/redis/6379/\*

[root@mysql55 ~]# scp /root/dump.rdb /var/lib/redis/6379/

[root@mysql55 ~]# /etc/init.d/redis\_6379 start

[root@mysql55 ~]# redis-cli -h 192.168.4.55 -p 6355

192.168.4.55:6355> keys \*

1) "x"

2) "x1"

3.1.3 优缺点

优点 : 高性能的持久化实现----创建子进程来执行持久化,先写入临时文件,过程中没有IO操作

适合大规模数据恢复

缺点 : 意外宕机,丢失最后一次持久化的所有数据

3.2 AOF

3.2.1 开启AOF

192.168.4.50:6350> config set appendonly yes //开启AOF

192.168.4.50:6350> config rewrite //写入配置文件

192.168.4.50:6350> save //保存

[root@mysql50 6379]# ls /var/lib/redis/6379/ //查看

3.2.2 恢复数据

[root@mysql50 ~]# scp /var/lib/redis/6379/appendonly.aof root@192.168.4.55:/root/ //拷贝备份文件

[root@mysql55 ~]# /etc/init.d/redis\_6379 stop //停服务

[root@mysql55 ~]# vim /etc/redis/6379.conf //改配置文件

673 appendonly yes

[root@mysql55 ~]# rm -rf /var/lib/redis/6379/\* /删除数据库目录下的信息

[root@mysql55 ~]# cp /root/appendonly.aof /var/lib/redis/6379/ //拷贝备份文件

[root@mysql55 ~]# /etc/init.d/redis\_6379 start //启动

[root@mysql55 ~]# redis-cli -h 192.168.4.55 -p 6355

192.168.4.55:6355> keys \* //查看数据

3.2.3 优化配置

702 appendfsync always //实时记录,磁盘同步

703 appendfsync everysec //每秒记录,磁盘同步

704 appendfsync no //写入aof,不磁盘同步

744 auto-aof-rewrite-percentage 100 //再次重写触发值

745 auto-aof-rewrite-min-size 64mb //首次重写触发值

3.2.4 修复AOF文件

[root@mysql50 6379]# redis-check-aof --fix appendonly.aof

0x 109: Expected prefix '\*', got: 's'

AOF analyzed: size=276, ok\_up\_to=265, diff=11

This will shrink the AOF from 276 bytes, with 11 bytes, to 265 bytes

Continue? [y/N]: y

Successfully truncated AOF

[root@mysql50 6379]# rm -rf /var/run/redis\_6379.pid

[root@mysql50 6379]# /etc/init.d/redis\_6379 start

[root@mysql50 6379]# redis-cli -h 192.168.4.50 -p 6350 -a 123456

3.2.5 优缺点

优点 : 灵活设置持久化方式

出现意外宕机时,尽可能丢失一秒数据

缺点 : 体积大于RDB方式

执行fsync策略时速度可能会比RDB方式慢

四 数据类型

4.1 string字符串

1) set key value [ex seconds] [px milliseconds] [nx | xx]

设置key及值 过期时间秒 毫秒 不存在就改|存在就改

2) setrange key offset values

--从偏移量开始复写key的特定位的值(修改指定字段)

192.168.4.50:6350> set aaa 123abc456

OK

192.168.4.50:6350> SETRANGE aaa 3 "nnn"

(integer) 9

192.168.4.50:6350> get aaa

" 123nnn456"

3) strlen key 统计字串长度

192.168.4.50:6350> strlen aaa

(integer) 9

4) append key value

存在则追加,不存在则创建key及value,返回key长度

192.168.4.50:6350> get x1

"1130"

192.168.4.50:6350> append x1 30

(integer) 6

192.168.4.50:6350> get x1

"113030"

5) setbit key offset value

对key所储存的子串,设置或清除特定偏移量(特定位置)上的位(bit)

value的值可以为1或0,offset为0~2^32之间

key不存在,则创建

192.168.4.51:6379> setbit bit 0 1 //设置bit第0位为1

192.168.4.51:6379> setbit bit 1 0 //设置bit第1位为0

6) bitcount key

统计子串中被设置为1的比特位数量

192.168.4.51:6379> setbit bits 0 1 //0001

192.168.4.51:6379> setbit bits 3 1 //1001

192.168.4.51:6379> bitcount bits //结果为2

192.168.4.51:6379> setbit peter 100 1 //网站上线100天用户登录了一次

192.168.4.51:6379> setbit peter 105 1 //网站上线105天用户登录了一次

192.168.4.51:6379> bitcount peter

7) decr key

将key中的值减1，key不存在则先初始化为0，再减1

192.168.4.51:6379> decr bb

192.168.4.51:6379> decr bb

8) decrby key decrement

将key中的值，减去decrement

192.168.4.51:6379> set count 100

192.168.4.51:6379> DECRBY cc 20 //定义每次减少20（步长）

192.168.4.51:6379> DECRBY cc 20

9) getrange key start end

返回字串值中的子字串，截取范围为start和end，负数偏移量表示从末尾开始计数，-1表示最后一个字符，-2表示倒数第二个字符

192.168.4.51:6379> set x 123456789

192.168.4.51:6379> getrange x -5 -1

"56789"

192.168.4.51:6379> getrange x 0 4

"12345"

10) incr key

将key的值加1，如果key不存在，则初始为0后再加1，主要应用为计数器

192.168.4.51:6379> set page 20

192.168.4.51:6379> incr page

(integer) 21

11) incrby key increment

将key的值增加increment

192.168.4.51:6379> set x 10

192.168.4.51:6379> incr x

(integer) 11

192.168.4.51:6379> incr x

(integer) 12

12) incrbyfloat key increment

为key中所储存的值加上浮点数增量 increment

192.168.4.51:6379> set num 16.1

192.168.4.51:6379> incrbyfloat num 1.1

"17.2"

4.2 list 列表

4.2.1简介:

一个变量存储多个值

先进后出

Redis的list是一个字符队列

4.2.2 list操作

1) lpush key value [ value ]

将一个或多个值value插入到列表key的表头

key不存在,则创建

lpush list a b c d

2) lrange key start stop

从开始位置读取keyd的值到stop结束

lrange list 0 2

lrange list 0 -1

lrange list 0 -2

3) lpop key

移除并返回列表头元素数据，key不存在则返回nil

192.168.4.51:6379> lpop list //删除表头元素，可以多次执行

4) llen key

返回列表key的长度

192.168.4.51:6379> llen list

5) lindex key index

返回列表中第index个值

192.168.4.51:6379> lindex list 1

6) lset key index value

将key中index位置的值修改为value

192.168.4.51:6379> lpush list a b c d

(integer) 5

192.168.4.51:6379> lset list 3 test //将list中第3个值修改为test

OK

7) rpush key value [value…]

将value插入到key的末尾

192.168.4.51:6379> rpush list3 a b c //list3值为a b c

(integer) 3

192.168.4.51:6379> rpush list3 d //末尾插入d

(integer) 4

8) rpop key

删除并返回key末尾的值

192.168.4.51:6379> RPOP list3

"d"

4.3 hash表

4.3.1简介: 一个变量有多个列,每个列有不同值

4.3.2 Hash表操作

1) hset key field value

将hash表中field值设置为value

192.168.4.51:6379> hset site google 'www.g.cn'

(integer) 1

192.168.4.51:6379> hset site baidu 'www.baidu.com'

(integer) 1

2) hget key filed

获取hash表中field的值

192.168.4.51:6379> hget site google

"www.g.cn"

3) hmset key field value [field value…]

同时给hash表中的多个field赋值

192.168.4.51:6379> hmset site google www.g.cn baidu www.baidu.com

4) hmget key field [field…]

返回hash表中多个field的值

192.168.4.51:6379> hmget site google baidu

1) "www.g.cn"

2) "www.baidu.com"

5) hkeys key

返回hash表中所有field名称

192.168.4.51:6379> hmset site google www.g.cn baidu www.baidu.com

192.168.4.51:6379> hkeys site

1) "google"

2) "baidu"

6) hgetall key

返回hash表中所有key名和对应的值列表

192.168.4.51:6379> hgetall site

1) "google"

2) "www.g.cn"

3) "baidu"

4) "www.baidu.com"

7) hvals key

返回hash表中所有key的值

192.168.4.51:6379> hvals site

1) "www.g.cn"

2) "www.baidu.com"

8) hdel key field [field…]

删除hash表中多个field的值，不存在则忽略

192.168.4.51:6379> hdel site google baidu

(integer) 2

++++++++++++++SECURITY 安全与监控(6天)

监控(Zabbix)

安全 : 系统常规加固 服务安全 数据安全 网络安全

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SECURITY\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : 监控基础

1.1 监控目的 : 报告系统运行情况 提前发现问题

1.2 监控资源类别 : 公开数据 (应用服务) 私有数据(cpu 内存等)

1.3 自动化监控系统: CaCti Nagios Zabbix

1.4 系统监控命令

traceroute (路由追踪) 装包traceroute-2.0.22-2.el7.x86\_64

iostat (查看磁盘读写速度) 装包 sysstat-10.1.5-13.el7.x86\_64

swapon -s (查看交换分区)

二 : Zabbix 简介

2.1 高度集成的监控解决方案

实现企业级的开源分布式监控

C/S模式采集数据

B/S模式实现WEB管理

2.2 部署运行环境 LNMP

yum install gcc openssl-devel zlib-devel pcre-devel //安装依赖包

tar -xf nginx-1.12.2.tar.gz

cd nginx-1.12.2/

./configure --with-http\_ssl\_module //装载ssl模块

make && make install

yum -y install php php-mysql mariadb mariadb-devel mariadb-server php-fpm //安装依赖包

vim /usr/local/nginx/conf/nginx.conf //65-71行 去掉注释(删除69行)

/usr/local/nginx/sbin/nginx -s reload //启动服务

systemctl start mariadb

systemctl enable mariadb

systemctl start php-fpm

systemctl enable php-fpm

三 : 部署Zabbix服务

3.1安装源码Zabbix

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

tar -xf zabbix-3.4.4.tar.gz

cd zabbix-3.4.4/

./configure \

--enable-server \ //监控服务服务器端

--enable-proxy \ //支持分布式监控

--enable-agent \ //当需要监控本机时安装

--with-mysql=/usr/bin/mysql\_config \ //支持mysql服务,及路径

--with-net-snmp \ //支持SNMP协议(简单的网络管理协议)

--with-libcurl //启用使用curl访问

make install

ls /usr/local/etc/ //检查

ls /usr/local/bin/

ls /usr/local/sbin/

3.2 初始化准备

3.2.1 准备存储数据时的库 ,表 ,及连接用户

[root@zabbix-server ~]# mysql

MariaDB [(none)]> create database zabbix character set utf8; //建库

MariaDB [(none)]> grant all on zabbix.\* to zabbix@'localhost' identified by 'zabbix'; //创建连接用户

[root@zabbix-server ~]# cd zabbix-3.4.4/database/mysql/

[root@zabbix-server mysql]# mysql -uzabbix -pzabbix zabbix < schema.sql //导入数据(有先后顺序)

[root@zabbix-server mysql]# mysql -uzabbix -pzabbix zabbix < images.sql

[root@zabbix-server mysql]# mysql -uzabbix -pzabbix zabbix < data.sql

3.2.2 上线Zabbix页面

cd zabbix-3.4.4/frontends/php/

cp -a \* /usr/local/nginx/html/ //保证属性不变

chmod -R 777 /usr/local/nginx/html/\* //递归修改权限

3.3 初始化

3.3.1 解决依赖

修改nginx配置文件

[root@zabbixserver ~]# vim /usr/local/nginx/conf/nginx.conf

… …

http{

fastcgi\_buffers 8 16k; //缓存php生成的页面内容，8个16k

fastcgi\_buffer\_size 32k; //缓存php生产的头部信息

fastcgi\_connect\_timeout 300; //连接PHP的超时时间

fastcgi\_send\_timeout 300; //发送请求的超时时间

fastcgi\_read\_timeout 300; //读取请求的超时时间

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

yum -y install php-gd php-xml php-ldap //安装依赖包

yum -y install php-bcmath php-mbstring

vim /etc/php.ini //修改php文件

672 post\_max\_size = 16M

384 max\_execution\_time = 300

394 max\_input\_time = 300

878 date.timezone =Asia/Shanghai

/usr/local/nginx/sbin/nginx -s stop //重启服务

/usr/local/nginx/sbin/nginx

systemctl restart php-fpm

3.3.2 打开真机浏览器输入http://192.168.2.5/index.php

cat /usr/local/nginx/html/conf/zabbix.conf.php //查看安装信息

3.4 登录管理页面

用户名admin 密码zabbix

修改语言 修改登录密码

3.5启动Zabbix监控服务

3.5.1 修改主配置文件

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

38 LogFile=/tmp/zabbix\_server.log //日志文件

85 DBHost=localhost //主机

95 DBName=zabbix //主机名

111 DBUser=zabbix //用户

119 DBPassword=zabbix //密码

3.5.2 启动服务

useradd zabbix //不添加zabbix用户会启动失败

zabbix\_server //启动服务命令

3.5.3 查看服务信息

ss -nulatp | grep 10051 //默认端口

四 : Zabbix监控服务

4.1 基础监控

4.1.1 配置被监控端(客户端)

安装Zabbix软件 ,提供Zabbix\_agentd服务

yum -y install pcre-devel gcc //依赖包

cd zabbix-3.4.4/

./configure --enable-agent //客户端只需要加载agent模块

make install

ls /usr/local/etc/ //查看信息

ls /usr/local/bin/

ls /usr/local/sbin/

修改服务的主配置文件

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

93 Server=127.0.0.1, 192.168.2.5 //允许访问本机agent服务的ip地址, 谁可以监控本机（被动监控模式）

134 ServerActive=192.168.2.5:10051 //谁可以监控本机（主动监控模式）

30 LogFile=/tmp/zabbix\_agentd.log //日志文件

启动服务

useradd zabbix //创建zabbix用户,没有会启动失败

zabbix\_agentd /启动

查看服务信息

ss -nulatp | grep 10050 //查看端口信息

ls /tmp/zabbix\_agentd.log //日志文件

4.1.2 配置监控服务器 , 管理员登录管理管理界面作如下操作:

1.添加监控主机

配置-->主机-->创建主机

主机名称,可见的名称(自定义)

群组(可自定义)

ip写要监控的主机ip

2. 选择监控模板

(主机名)-->模板-->添加-->更新

3.查看监控数据

检测中--> 最新数据-->群组-->主机

4.2 自定义监控(客户端定义监控命令给监控服务器使用)

4.2.1 配置客户端 192.168.2.100

启用自定义监控

修改配置文件

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

280 UnsafeUserParameters=1 //启用自定义监控

# Format: UserParameter=<key>,<shell command> //定义监控命令格式

265 Include=/usr/local/etc/zabbix\_agentd.conf.d/\*.conf //加载监控命令配置文件目录

cd /usr/local/etc/zabbix\_agentd.conf.d/ //配置文件目录

自定义key

vim count.line.passwd.conf //文件名自定义,要有标示性

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

重启服务

killall -9 zabbix\_agentd

zabbix\_agentd

ss -nulatp | grep zabbix\_agentd //重启成功就有端口

测试配置

zabbix\_get -s 127.0.0.1 -p 10050 -k count.line.passwd //本机测试连环回口,因为没设置允许其他人

useradd yaya //添加用户在测试

zabbix\_get -s 127.0.0.1 -p 10050 -k count.line.passwd

//-s ip地址 -p 端口 -k 命令

4.2.2 配置监控服务器

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SECURITY\*DAY02\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : Zabbix报警机制

1.1 相关概念

自定义的监控项默认不会自动报警，

首页也不会提示错误，

需要配置触发器与报警动作才可以自定报警。

什么是触发器（trigger）？

当触发条件发生后，会导致一个触发事件

触发事件会执行某个动作

什么是动作（action）？

动作是触发器的条件被触发后所执行的行为

可以是发送邮件、也可以是重启某个服务等

1.2 配置监控报警

1.2.1 创建触发器

配置-->模板--触发器-->创建触发器

1.2.2 创建动作

设置邮件服务器

yum -y install postfix //安装邮件服务

systemctl start postfix //启动

ss -nulatp |grep :25

echo "127.0.0.1 zabbixserver" >> /etc/hosts //Zabbix与本机绑定

yum -y install mailx //命令行发邮件的安装包

mail -s "aaa" zabbix < /etc/hosts //发邮件给zabbix

登录管理页面指定邮件服务器 :

管理-->报警媒介类型-->电子邮件-->(内容)-->更新

SMTP服务器 : localhost

SMTP HELO : zabbixserver

SMTP电邮 : (发件人邮箱) root@localhost

登录管理页面为账户指定Media(指定收件人) :

管理-->用户-->admin-->报警媒介-->添加-->收件人

登录管理页面创建动作 :

配置-->动作-->创建动作-->(名称自定义)-->触发条件(触发器)-->操作-->新的--用户添加

效果测试 : 1 添加用户

2 查看监控数据

3 查看触发器是否触发

4 查看邮件

二 : Zabbix进阶操作

2.1 自动发现

2.1.1简介:

当Zabbix需要监控的设备越来越多，手动添加监控设备越来越有挑战，

此时，可以考虑使用自动发现功能，自动添加被监控主机，

实现自动批量添加一组监控主机功能。

自动发现可以实现：

自动发现、添加主机，自动添加主机到组；

自动连接模板到主机，自动创建监控项目与图形等

2.1.2配置自动发现

1)创建自动发现规则

配置-->自动发现-->创建发现规则-->(设置)-->添加

2)创建动作

配置-->动作-->(自动发现)创建动作-->添加

3)创建新的虚拟机 配置ip地址并运行网站服务

2.2 主被动监控

2.2.1 介绍 :

默认zabbix采用的是被动监控，主动和被动都是对被监控端主机而言的！

被动监控：Server向Agent发起连接，发送监控key，Agent接受请求，响应监控数据。

主动监控：Agent向Server发起连接，Agent请求需要检测的监控项目列表，Server响应Agent发送一个items列表，

Agent确认收到监控列表，TCP连接完成，会话关闭，Agent开始周期性地收集数据。

区别： Server不用每次需要数据都连接Agent，Agent会自己收集数据并处理数据，Server仅需要保存数据即可。

当监控主机达到一定量级后，Zabbix服务器会越来越慢，此时，可以考虑使用主动监控，释放服务器的压力。

2.2.2 配置主动监控

1) 配置客户端 192.168.2.201

scp zabbix-3.4.4.tar.gz root@192.168.2.201:/root/

安装源码软件

yum -y install gcc pcre-devel

./configure --enable-agent

make install

修改配置文件

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

93 # Server=127.0.0.1 //注释该行，允许谁监控本机

118 StartAgents=0 //被动监控时启动多个进程 设置为0，则禁止被动监控，不启动zabbix\_agentd服务

134 ServerActive=192.168.2.5 //允许哪些主机监控本机（主动模式），一定要取消127.0.0.1

145 Hostname=web201 //告诉监控服务器，是谁发的数据信息,一定要和zabbix服务器配置的监控主机名称一致（后面设置）

183 RefreshActiveChecks=120 //默认120秒检测一次

启动服务

useradd zabbix

zabbix\_agentd

查看服务信息

ps -C zabbix\_agentd //只有进程没端口

2) 配置服务器 192.168.2.5

克隆模板

配置-->模板-->模板名-->全克隆-->(改名)添加

修改监控项模式

监控项-->(全选)批量更新-->类型(主动式)-->更新-->(将三项不能主动式的禁用)

添加监控主机

配置-->主机-->创建主机-->(主机名写配置文件里的)-->ip0.0.0.0端口0 -->添加

为主机添加监控模板

模板--> 自定义那个-->更新

查看数据图表

2.3 拓扑图与聚合图形

2.3.1 介绍

绘制拓扑图可以快速了解服务器架构

2.3.2创建拓扑图

监测中-->拓扑图-->创建拓扑图-->构造函数

2.3.3创见聚合图形 (聚合图形可以在一个页面显示多个数据图表，方便了解多组数据)

三 : 监控案例(在客户端编写监控脚本,给监控服务器使用)

环境准备 :

在200主机运行Zabbix\_agent服务 :

vim /usr/local/etc/zabbix\_agentd.conf //修改配置文件,启用被动监控

93 Server=127.0.0.1, 192.168.2.5

134 ServerActive=192.168.2.5:10051

265 Include=/usr/local/etc/zabbix\_agentd.conf.d/\*.conf

280 UnsafeUserParameters=1

启动服务

在200主机上运行nginx服务 :

./configure --with-http\_stub\_status\_module

make && make install

启动服务

vim /usr/local/nginx/conf/nginx.conf //添加启用status模块

server {

location /status {

stub\_status on;

}

......

}

3.1监控nginx服务状态

3.1.1编写脚本

vim /usr/local/bin/nginx\_status.sh //编写脚本,利用case命令(与if相近)

case $1 in

"Active")

curl -s http://localhost/status | awk 'NR==1{print $3}';;

"accepts")

curl -s http://localhost/status | awk 'NR==3{print $1}';;

"Waiting")

curl -s http://localhost/status | awk 'NR==4{print $6}'

esac

chmod +x /usr/local/bin/nginx\_status.sh //加权限

/usr/local/bin/nginx\_status.sh Waiting //运行测试

3.1.2 将脚本定义为监控命令

vim /usr/local/etc/zabbix\_agentd.conf.d/status.conf //编写监控命令

UserParameter=nginx\_status[\*],/usr/local/bin/nginx\_status.sh $1 // \* 代表传给$1的值

zabbix\_get -s 127.0.0.1 -p 10050 -k nginx\_status[Waiting] //测试

zabbix\_get -s 127.0.0.1 -p 10050 -k nginx\_status[accepts]

zabbix\_get -s 127.0.0.1 -p 10050 -k nginx\_status[Active]

3.1.3 创建监控模板 ATMP2

配置-->模板-->创建模板

3.1.4 创建应用集 mon\_nginx

(接上)模板-->应用集-->创建应用集

3.1.5 创建监控项

(接上)监控项-->创建监控项-->键值(写命令)-->应用集(mon\_nginx)-->添加 //应用及相当于目录

3.1.6 使用新模板监控200主机

200主机添加新模板

3.1.7 查看监控数据

ab -n 1000 -c 1000 http://127.0.0.1/status //压力测试

3.2监控网络连接状态

[root@zabbixclient\_web1 ~]# ss -antup

//-a显示所有

//-t显示TCP连接状态

//-u显示UDP连接状态

//-n以数字形式显示端口号和IP地址

//-p显示连接对应的进程名称

3.2.1编写监控脚本

vim /usr/local/bin/tcp\_status.sh

case $1 in

"estab")

ss -antp | awk '/^ESTAB/{x++} END{print x}';;

"close\_wait")

ss -antp | awk '/^CLOSE-WAIT/{x++} END{print x}';;

"time\_wait")

ss -antp | awk '/^TIME-WAIT/{x++} END{print x}';;

esac

chmod +x /usr/local/bin/tcp\_status.sh //加运行权限

3.2.2编写监控配置文件

vim /usr/local/etc/zabbix\_agentd.conf.d/status.conf //配置(也可新建)

UserParameter=net.status[\*],/usr/local/bin/tcp\_status.sh $1 //添加配置文件内容

killall -9 zabbix\_agentd //重启

zabbix\_agentd

zabbix\_get -s 127.0.0.1 -p 10050 -k net.status[estab] //测试

zabbix\_get -s 127.0.0.1 -p 10050 -k net.status[time\_wait]

zabbix\_get -s 127.0.0.1 -p 10050 -k net.status[close\_wait]

3.2.3配置监控服务器,登录管理页面作如下配置 :

1) 在监控模板ATMP2里创建新应用集 mon\_tcp

2) 创建监控项

3) 查看监控数据

3.3课外扩展

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SECURITY\*DAY03\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 :Linux 基本防护

1.1 用户账号安全

1.1.1修改用户zhangsan的账户属性

chage命令的语法格式：

chage –l 账户名称 //查看账户信息

chage –E 时间 账户名称 //修改账户有效期

chage -E 2019/07/31 bob

chage -l bob

echo 123456 | passwd --stdin bob

chage -d 0 bob //下次登录必须更改密码才能登陆

-d, --lastday 最近日期 将最近一次密码设置时间设为“最近日期”

-E, --expiredate 过期日期 将帐户过期时间设为“过期日期”

-h, --help 显示此帮助信息并推出

-I, --inactive INACITVE 过期 INACTIVE 天数后，设定密码为失效状态

-l, --list 显示帐户年龄信息

-m, --mindays 最小天数 将两次改变密码之间相距的最小天数设为“最小天数”

-M, --maxdays 最大天数 将两次改变密码之间相距的最大天数设为“最大天数”

-R, --root CHROOT\_DIR chroot 到的目录

-W, --warndays 警告天数 将过期警告天数设为“警告天数”

定义默认有效期

/etc/login.defs这个配置文件，决定了账户密码的默认有效期。

grep -v '#' /etc/login.defs //查看文件有效信息

MAIL\_DIR /var/spool/mail //创建邮箱

PASS\_MAX\_DAYS 99999 //密码最长有效期

PASS\_MIN\_DAYS 0 //密码最短有效期

PASS\_MIN\_LEN 5 //密码最短长度

PASS\_WARN\_AGE 7 //密码过期前几天提示警告信息

UID\_MIN 1000 //最小UID

UID\_MAX 60000 //UID最大值

SYS\_UID\_MIN 201

SYS\_UID\_MAX 999

GID\_MIN 1000 //最小GID

GID\_MAX 60000 //最大GID

SYS\_GID\_MIN 201

SYS\_GID\_MAX 999

CREATE\_HOME yes //创建家目录

UMASK 077

USERGROUPS\_ENAB yes //创建从属组

ENCRYPT\_METHOD SHA512 //加密算法

1.1.2临时锁定用户zhangsan的账户，使其无法登录，验证效果后解除锁定

passwd -l zhangsan //锁定用户账号lock

passwd -S zhangsan //查看状态status

passwd -u zhangsan //解锁用户账号

1.1.3修改tty登录的提示信息，隐藏系统版本

cat /etc/issue //确认原始文件

Kernel \r on an \m

cp /etc/issue /etc/issue.origin //备份文件

vim /etc/issue //修改文件内容

Windows Server 2012 Enterprise R2

NT 6.2 Hybrid

1.2 文件系统安全

1.2.1禁用非必要的系统服务RHEL7

systemctl disable httpd //关闭开机自启

systemctl enable httpd //开机自启

1.2.2禁用非必要的系统服务RHEL6及之前

chkconfig httpd off //开机自启

chkconfig httpd on //关闭开机自启

1.2.3锁定/解锁保护文件(给文件属性)

属性i : chattr +i 文件名 chattr -i 文件名 //不可变

属性a: chattr +a 文件名 chattr -a 文件名 //仅追加

lsattr 文件名 //查看文件特殊属性

chattr +a /etc/passwd //锁定文件

lsattr /etc/passwd //查看

echo "192.168.4.1 xyz" > /etc/hosts //失败

echo "192.168.4.1 xyz" >> /etc/hosts //成功

chattr -a /etc/passwd //解锁文件

　 chattr +i /etc/resolv.conf //加了权限不可删除

lsattr /etc/resolv.conf

----i---------- /etc/resolv.conf

rm -rf /etc/resolv.conf

rm: 无法删除"/etc/resolv.conf": 不允许的操作

二 :用户切换与提权

2.1 用户切换

su root //切换用户不换环境

su - root //切换用户及环境

su - -c "systemctl restart sshd" root //以管理员重启服务(要输入密码)

2.2用户提权(设置系统普通用户可以执行root命令的权限)

cat /var/log/secure //记录su 操作的日志文件

2.2.1 主配置文件 // /etc/sudoers

打开主配置文件方法1 visudo (没颜色,错了不提示)

打开主配置文件方法2 vim /etc/sudoers (保存要用 :wq! )

格式1 用户名 主机名列表=命令列表

格式2 %用户组名 主机名列表=命令列表

99 # %wheel ALL=(ALL) ALL //工作中一定要这行注释掉,否则将用户加入wheel组就可以获得root的所有权限

sudo -l //普通用户查看可以sudo命令

启用日志

Defaults logfile="/var/log/sudo" //主配置文件里添加(日志路径)

tail /var/log/sudo

vim /etc/sudoers //添加sudo命令

aa localhost,localhost=/usr/bin/vim /etc/my.cnf , /usr/bin/systemctl \* mysql

//用户 aa 可以在 localhost 上运行 /usr/bin/vim /etc/my.cnf , /usr/bin/systemctl \* mysql

mike localhost,localhost=/sbin/\* , !/sbin/ifconfig eth0 //开启用户mike/sbin/下除了ifconfig eth0的所有权限

softadm ALL=(ALL) /usr/bin/systemctl //授权softadm以root身份执行systemctl命令（ALL包括root）

配置别名 (别名名称必须大写)

用户别名 User\_Alias 别名名称=值列表

主机别名 Host\_Alias 别名名称=值列表

命令别名 Cmnd\_Alias 别名名称=值列表

三 : 提高SSH服务安全

3.1 基本防护

3.1.1修改ssh服务运行参数

vim /etc/ssh/sshd\_config

Port 2222 //改用非标准端口

Protocol 2 //SSH协议

PermitRootLogin no //禁止root用户登录

PermitEmptyPasswords no //禁止密码为空的用户登录

UseDNS no //不解析客户机地址

LoginGraceTime 1m //登录限时,输错三次后1分钟之内不能登录

MaxAuthTries 3 //每连接最多认证次数(防止暴力破解)

ListenAddress //监听地址

3.1.2 设置用户黑白命令(控制客户端连接使用的用户)

白名单 : 定义仅允许连接的用户 (服务需重启)

## AllowGroups GROUP1 GROUP2

## AllowUsers USER1 USER2

tail -1 /etc/ssh/sshd\_config

AllowUsers root@192.168.4.254 ass //白名单,只允许真机使用root用户登录,所有主机使用ass用户登录

黑名单: 定义仅不允许连接的用户 (sssh服务需重启)

## DenyGroups GROUP1 GROUP2

## DenyUsers USER1 USER2

DenyUsers mike aa //禁止mike aa用户登录

3.2密钥对验证

ssh服务验证用户登录方式有两种 : 口令 密钥对

rm -rf ~/.ssh //清除密钥对

3.2.1配置12主机可以用root用户,无密码登录11主机

11主机配置

修改配置文件仅允许密钥对登录(传完密钥对在关闭口令登录)

65 PasswordAuthentication yes //密码验证允许(传完密钥对在关闭口令登录)

43 #PubkeyAuthentication yes //密钥对验证

46 AuthorizedKeysFile .ssh/authorized\_keys //秘钥文件路径

重启

12主机

rm -rf ~/.ssh //清除密钥对

ssh-keygen //创建密钥对

ls ~/.ssh //查看

ssh-copy-id root@192.168.4.11 //传输密钥对

四 :SELinux安全防护

教学环境 : 将Linux服务器的SELinux设为enforcing强制模式

vim /etc/selinux/config

SELINUX=enforcing //设置SELinux为强制模式

SELINUXTYPE=targeted //保护策略为保护主要的网络服务安全

touch /.autorelabel //环境问题,需要创建文件

reboot

4.1 SELinux介绍

getenforce //查看当前状态

setenforce 1 //设置SELinux为强制模式

setenforce 0 //设置SELinux为强制模式

sestatus //查看当前状态

SELinux status: enabled

SELinuxfs mount: /sys/fs/selinux

SELinux root directory: /etc/selinux

Loaded policy name: targeted //保护策略

Current mode: enforcing //工作模式

Mode from config file: enforcing

Policy MLS status: enabled

Policy deny\_unknown status: allowed

Max kernel policy version: 31 //版本

4.2 查看安全上下文 (-Z)

查文件:

ls -lZ /etc/passwd

-rw-r--r--. root root system\_u:object\_r:passwd\_file\_t:s0 /etc/passwd

属性构成 : system\_u:object\_r:passwd\_file\_t:s0

查目录:

ls -ldZ /etc/

drwxr-xr-x. root root system\_u:object\_r:etc\_t:s0 /etc/

属性构成 : system\_u:object\_r:etc\_t:s0

查进程:

ps aux -Z | grep httpd

system\_u:system\_r:httpd\_t:s0 root 1002 0.1 0.3 224020 5020 ? Ss 16:18 0:00 /usr/sbin/httpd -DFOREGROUND

属性构成 : system\_u:system\_r:httpd\_t:s0

属性构成 : system\_u:object\_r:passwd\_file\_t:s0

system\_u:object\_r:etc\_t:s0

system\_u:system\_r:httpd\_t:s0

用户:角色:访问类型:选项...

一般操作规律 :

移动的文件, 原有上下文属性不变

复制的文件, 自动继承目标位置的上下文

新建的文件, 自动继承父目录的上下文

[root@localhost ~]# touch /var/www/html/aaa.html

[root@localhost ~]# touch /root/bbb.html

[root@localhost ~]# ls -lZ /var/www/html/aaa.html

-rw-r--r--. root root unconfined\_u:object\_r:httpd\_sys\_content\_t:s0 /var/www/html/aaa.html

[root@localhost ~]# ls -ldZ /var/www/html/

drwxr-xr-x. root root system\_u:object\_r:httpd\_sys\_content\_t:s0 /var/www/html/

[root@localhost ~]# ls -ldZ /root/

dr-xr-x---. root root system\_u:object\_r:admin\_home\_t:s0 /root/

[root@localhost ~]# ls -lZ /root/bbb.html

-rw-r--r--. root root unconfined\_u:object\_r:admin\_home\_t:s0 /root/bbb.html

[root@localhost ~]# mv /root/bbb.html /var/www/html/bbb.html //标签不对,浏览器没法访问,提示权限不足

修改安全上下文:

指定安全上下文:

chcon -t httpd\_sys\_content\_t /var/www/html/b.html

恢复安全上下文 :

restorecon /var/www/html/b.html //不用指定安全上下文

touch /.autorelabel //下次重启全部重置安全上下文

4.3 调整SELinux布尔值 (功能开关)

getsebool -a //查看所有服务的布尔值

setsebool -P samba\_share\_nfs on // -P是永久生效

setsebool -P 配置项 on | off //格式

案例 : 在50主机SELinux启用状态下，调整策略打开vsftpd服务的匿名上传访问

配置一个允许匿名上传的vsftpd服务作为测试环境

yum -y install vsftpd

vim +29 /etc/vsftpd/vsftpd.conf //修改配置文件

anon\_upload\_enable=YES //注释去掉,开启匿名上传

mkdir /var/ftp/sharedir //目录默认没有,手动创建

chmod o+w /var/ftp/sharedir //给其他人加权限

systemctl start vsftpd //启动

cp /etc/passwd /var/ftp/sharedir //拷贝文件

客户端配置 :

yum -y install ftp //装包,命令行登录ftp命令

ftp 192.168.4.13 //登录

Name (192.168.4.13:root): ftp //匿名登录默认用户名

Password: //密码为空,直接回车

ftp> cd sharedir //切换目录

ftp> ls

-rw-r--r-- 1 0 0 982 Jul 27 09:20 passwd

ftp> get passwd //下载passwd文件

ftp> lcd /etc/ //lcd 是进本地目录, cd是进ftp目录

ftp> put hosts

553 Could not create file. (失败) //SELinux开启默认不允许上传

ftp> bye //退出

50主机修改 :

setsebool -P ftpd\_anon\_write on //设置匿名用户可写

setsebool -P ftpd\_full\_access on //开启完全权限

getsebool -a | grep ftp //查看状态

修改后上传 :

ftp> put shadow

1110 bytes sent in 8.6e-05 secs (12906.98 Kbytes/sec)

ftp> ls

....

-rw------- 1 14 50 1110 Jul 27 09:47 shadow

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CLOUD\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

git clone git://43.254.90.134/nsd1904.git //克隆笔记命令

git pull //更新笔记 (必须进入git目录)

一：virsh基本操作

虚拟机操作:

1 启动虚拟机 [root@room9pc01 ~]# virsh start node1

2 重启虚拟机 [root@room9pc01 ~]# virsh reboot node1

3 强制关闭虚拟机 [root@room9pc01 ~]# virsh destroy node1

4 设置虚拟机开机自动运行 [root@room9pc01 ~]# virsh autostart node1 //取消加 --disable

5 查看虚拟机网卡信息 [root@room9pc01 ~]# virsh domiflist node1

6 查看虚拟机硬盘信息 [root@room9pc01 ~]# virsh domblklist node1

7 列出当前正在运行的虚拟机 [root@room9pc01 ~]# virsh list

8 查看虚拟机的信息 [root@room9pc01 ~]# virsh dominfo nova

virsh虚拟网络管理:

net-list [--all] 列出所有

net-start 启动

net-destroy 强制停止

net-define 创建

net-undefine 删除

net-edit 修改配置

net-autostart 自启动

二 : qemu-img基本操作管理

1. 创建一个新的镜像盘文件

qemu-img命令格式：qemu-img 命令 参数 块文件名称 大小

qemu-img create -f qcow2 disk.img 50G //qcow2为创建的格式

Formatting 'disk.img', fmt=qcow2 size=53687091200 encryption=off cluster\_size=65536 lazy\_refcounts=off

2. 查询镜像盘文件信息

qemu-img info 磁盘路径

qemu-img info aaaa.img

3. 使用后端模板文件创建一个新的镜像盘文件

-b使用后端模板文件

[root@room9pc01 ~]# qemu-img create -b disk.img -f qcow2 disk1.img 16G

4. 修改ip地址(配置文件)

[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0

# Generated by dracut initrd

DEVICE="eth0" //网卡名

ONBOOT="yes" //开机启动

NM\_CONTROLLED="no" //不受NetworkManager控制

TYPE="Ethernet" //类型

BOOTPROTO="static" //协议(dhcp|static|none(聚合链路用))

IPADDR="192.168.1.100" //ip地址

NETMASK="255.255.255.0" //子网掩码

GATEWAY="192.168.1.254" //默认网管

5. 磁盘在线扩容

[root@room9pc01 images]# virsh blockresize --path /var/lib/libvirt/images/dc.img --size 50G dc //真机

[root@localhost ~]# LANG=C //虚拟机 设置语言为英文

[root@localhost ~]# groepart /dev/vda 1 //虚拟机 扩容第一个分区

[root@localhost ~]# xfs\_growfs / //虚拟机 扩容文件系统

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CLOUD\*DAY02\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : Oenstack

Horizon :

用于管理Openstack服务的,基于web的管理接口

nove :

在节点上用于管理虚拟机的服务

是一个分布式服务,能够与Keystone交互实现认证

Glance :

管理镜像盘(共享)

Swift:

对象存储,解决数据通讯的问题

Quantun:

解决网络问题,网络组件

Cinder:

卷存储,lvs,块存储

Keystone:

集中式认证管理,提供令牌,所有组件的认证管理

二 :虚拟机准备

openstack

2CPU 6G内存(最小) 50G硬盘

nova01,nova02

2CPU 4.5G内存 100G硬盘

三台互ping主机名可以通

修改resolv.conf文件去掉search开头的行

准备三台虚拟机(也可手动创建,可以省略扩容磁盘)

1. [student@room9pc01 images]$base-vm openstack nova01 nova02

2. opensatck主机扩容为50G, nova01 和 nova02 主机扩容为100G

物理机]$ virsh blockresize --path /var/lib/libvirt/images/openstack.img --size 100G openstack

[student@room9pc01 images]$ virsh console openstack

[root@localhost ~]# /usr/bin/growpart /dev/vda 1

[root@localhost ~]# /usr/sbin/xfs\_growfs /

[root@localhost ~]# df -h

3. 三台主机配置静态ip（以一台为例）

openstack：192.168.1.10

nova01： 192.168.1.11

nova02： 192.168.1.12

[root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0

IPADDR=192.168.1.10

NEMASK=255.255.255.0

GATEWAY=192.168.1.254

4. 三台主机修改主机名，配置/etc/hosts，和/etc/resolv.conf文件（以一台为例）

[root@localhost ~]# vim /etc/resolv.conf //去掉search开头的行

; generated by /usr/sbin/dhclient-script

nameserver 192.168.1.254

5. 修改三台主机的内存（openstack6G，nova01 和nova02 4G）

[student@room9pc01 ~]$ virsh edit openstack

<memory unit='KiB'>6588282</memory>

<currentMemory unit='KiB'>6588282</currentMemory>

三 : 配置yum仓库

1. 挂载镜像

[student@room9pc01 ~]$ cd /linux-soft/04/openstack/

[student@room9pc01 openstack]$ ls

cirros.qcow2 RHEL7-extras.iso RHEL7OSP-10.iso small.qcow2

[student@room9pc01 openstack]$ mkdir /var/ftp/RHEL7-extras

[student@room9pc01 openstack]$ mkdir /var/ftp/RHEL7OSP-10

[student@room9pc01 openstack]$ mount RHEL7-extras.iso /var/ftp/RHEL7-extras/

mount: /dev/loop1 写保护，将以只读方式挂载

[student@room9pc01 openstack]$ mount RHEL7OSP-10.iso /var/ftp/RHEL7OSP-10/

mount: /dev/loop2 写保护，将以只读方式挂载

2. 配置yum源

[root@openstack ~]# vim /etc/yum.repos.d/openstack.repo

[local\_repo]

name=CentOS-$releasever - Base

baseurl="ftp://192.168.1.254/centos-1804"

enabled=1

gpgcheck=1

[RHEL7-extras]

name=RHEL7-extras

baseurl="ftp://192.168.1.254/RHEL7-extras"

enabled=1

gpgcheck=0

[RHEL7OSP-package]

name=RHEL7OSP-package

baseurl="ftp://192.168.1.254/RHEL7OSP-10/rhel-7-server-openstack-10-rpms"

enabled=1

gpgcheck=0

[RHEL7OSP-devtools]

name=RHEL7OSP-devtools

baseurl="ftp://192.168.1.254/RHEL7OSP-10/rhel-7-server-openstack-10-devtools-rpms"

enabled=1

gpgcheck=0

四 : 检查基础环境

安装额外的软件包

是否卸载firewalld 和 NetworkManager

检查配置主机网络参数（静态IP）

主机名必须能够相互 ping 通

检查配置主机yum源（4个，10670）

依赖软件包是否安装

检查NTP服务器是否可用

检查 /etc/resolv.conf 不能有 search 开头的行

1. 安装额外软件包（三台机器操作，这里以一台为例）

[root@openstack]# yum install -y qemu-kvm libvirt-client libvirt-daemon libvirt-daemon-driver-qemu

python-setuptools

2. 是否卸载firewalld 和 NetworkManager

[root@openstack ~]# rpm -qa | grep NetworkManager\*

[root@openstack ~]# rpm -qa | grep firewalld\*

3. 检查配置主机网络参数

[root@openstack ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0

# Generated by dracut initrd

DEVICE="eth0"

ONBOOT="yes"

NM\_CONTROLLED="no"

TYPE="Ethernet"

BOOTPROTO="static"

PERSISTENT\_DHCLIENT="yes"

IPADDR=192.168.1.10

NEMASK=255.255.255.0

GATEWAY=192.168.1.254

4. 验证主机名是否互通

[root@openstack ~]# ping openstack

[root@openstack ~]# ping nova01

[root@openstack ~]# ping nova02

5. 检查配置主机yum源

[root@openstack ~]# yum repolist //总10670

6. 检查时间同步是否可用

[root@openstack ~]# chronyc sources -v

^\* gateway 3 7 377 28 +31us[ +89us] +/- 25ms

7 .检查/etc/resolv.conf 不能有 search 开头的行

[root@openstack ~]# cat /etc/resolv.conf

; generated by /usr/sbin/dhclient-script

nameserver 192.168.1.254

8. 启动libvirtd服务

[root@openstack libvirt]# systemctl start libvirtd

[root@openstack libvirt]# systemctl enable libvirtd

9. virsh list //不报错就对

五 : 部署Openstack

[root@openstack ~]# yum install -y openstack-packstack

[root@openstack ~]# packstack --gen-answer-file answer.ini

//answer.ini与answer.txt是一样的，只是用vim打开answer.ini文件有颜色

Packstack changed given value to required value /root/.ssh/id\_rsa.pub

[root@openstack ~]# vim answer.ini

42 CONFIG\_SWIFT\_INSTALL=n

45 CONFIG\_CEILOMETER\_INSTALL=n //计费相关模块

49 CONFIG\_AODH\_INSTALL=n //计费相关模块

53 CONFIG\_GNOCCHI\_INSTALL=n //计费相关模块

75 CONFIG\_NTP\_SERVERS=192.168.1.254 //时间服务器的地址

98 CONFIG\_COMPUTE\_HOSTS=192.168.1.11

102 CONFIG\_NETWORK\_HOSTS=192.168.1.10,192.168.1.11

333 CONFIG\_KEYSTONE\_ADMIN\_PW=a //修改管理员的密码

840 CONFIG\_NEUTRON\_ML2\_TYPE\_DRIVERS=flat,vxlan //驱动类型

876 CONFIG\_NEUTRON\_ML2\_VXLAN\_GROUP=239.1.1.5 //设置组播地址,最后一个随意不能为0和255,其他固定

910 CONFIG\_NEUTRON\_OVS\_BRIDGE\_MAPPINGS=physnet1:br-ex //物理网桥的名称

921 CONFIG\_NEUTRON\_OVS\_BRIDGE\_IFACES=br-ex:eth0 //br-ex桥的名称与eth0连接，管理eth0,网桥与哪个物理网卡连接

1179 CONFIG\_PROVISION\_DEMO=n //DEMO是否测试

[root@openstack ~]# packstack --answer-file=answer.ini //安装会持续半小时左右

tallation log file is available at: /var/tmp/packstack/20190423-170603-b43g\_i/openstack-setup.log

Installing:

Clean Up [ DONE ]

Discovering ip protocol version [ DONE ]

root@192.168.1.11's password:

root@192.168.1.10's password:

Setting up ssh keys

\*\*\*\* Installation completed successfully \*\*\*\*\*\* //出现这个为成功

六 : 网络管理

1. 查看外部OVS网桥

1）查看br-ex网桥配置（br-ex为OVS网桥设备）

[root@openstack ~]# cat /etc/sysconfig/network-scripts/ifcfg-br-ex

ONBOOT="yes"

NM\_CONTROLLED="no"

IPADDR="192.168.1.10"

PREFIX=24

GATEWAY=192.168.1.254

DEVICE=br-ex

NAME=br-ex

DEVICETYPE=ovs

OVSBOOTPROTO="static"

TYPE=OVSBridge

2）查看eth0网卡配置（该网卡为OVS网桥的接口）

[root@openstack ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

NAME=eth0

DEVICETYPE=ovs

TYPE=OVSPort

OVS\_BRIDGE=br-ex

ONBOOT=yes

BOOTPROTO=none

3）验证OVS配置

[root@openstack ~]# ovs-vsctl show

Bridge br-ex

Controller "tcp:127.0.0.1:6633"

is\_connected: true

fail\_mode: secure

Port br-ex

Interface br-ex

type: internal

Port phy-br-ex

Interface phy-br-ex

type: patch

options: {peer=int-br-ex}

Port "eth0"

Interface "eth0"

ovs\_version: "2.5.0"

七 : 登录openstack

1）浏览器访问

[root@openstack ~]# firefox 192.168.1.10 //访问失败

2）需要改配置文件并重新加

[root@openstack conf.d]# vim /etc/httpd/conf.d/15-horizon\_vhost.conf

35 WSGIProcessGroup apache

36 WSGIApplicationGroup %{GLOBAL} //添加这一行

[root@openstack conf.d]# apachectl graceful //重新载入配置文件

[root@openstack conf.d]# systemctl reload httpd //或者这条命令

3）查看用户名和密码

[root@openstack conf.d]# cd

[root@openstack ~]# ls

answer.ini keystonerc\_admin //keystonerc\_admin生成的文件，里面有用户名和密码

[root@openstack ~]# cat keystonerc\_admin

unset OS\_SERVICE\_TOKEN

export OS\_USERNAME=admin //用户名

export OS\_PASSWORD=a //密码

export OS\_AUTH\_URL=http://192.168.1.10:5000/v2.0

export PS1='[\u@\h \W(keystone\_admin)]\$ '

export OS\_TENANT\_NAME=admin

export OS\_REGION\_NAME=RegionOne

八: 故障排错

1）ntp时间不同步

解决办法：查看ntp时间服务器，是否出现\*号，若没有，查看配置文件，配置ntp服务器步骤在案例3，可以参考

[root@room9pc01 ~]# chronyc sources -v //出现\*号代表NTP时间可用

^\* 120.25.115.20 2 6 17 62 -753us[-7003us] +/- 24ms

[root@openstack ~]# chronyc sources -v

^\* 192.168.1.254 3 9 377 504 +50us[-20us] +/- 24ms

[root@nova ~]# chronyc sources -v

^\* 192.168.1.254 3 9 377 159 -202us[-226us] +/- 24ms

2）网桥名称写错

解决办法：检查配置文件

[root@openstack ~]# vim answer.ini

921 CONFIG\_NEUTRON\_OVS\_BRIDGE\_IFACES=br-ex:eth0 //br-ex桥的名称与eth0连接，管理eth0,网桥与哪个物理网卡连接

3）若/root/.ssh/id\_rsa.pub，提示password，同样是配置文件没有写对

4）yum源没有配置正确

解决办法：检查yum是否为10853个软件包，查看是否是yum源没有配置正确，之后安装oprnstack-dashboard

5）出现Cannot allocate memory

解决办法：内存不足，重新启动主机

6）出现/usr/bin/systemctl start openvswith ... falied，说明是ssse3指令集的错误

解决办法：编辑openstack的xml文件，在里面添加

<cpu mode='host-passthrough'>

</cpu>

7）若出现 Could not prefetch... ‘openstack’。

配置文件里面有中文符号

8）访问openstack出错

没有修改Apache配置文件

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CLOUD\*DAY03\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 :

项目 : 一组隔离的资源与对象,由一组关联的用户进行管理(一般一个项目对应一个企业)

云主机类型 : 就是资源模板

重置非管理员用户密码 :

[root@openstack ~]# source keystonerc\_admin //命令行登录

[root@openstack ~(keystone\_admin)]# openstack user set --password redhat nsd1904

二 : 管理

创建项目

身份管理-->项目-->创建项目

创建用户

身份管理-->用户-->创建用户-->(角色选member)(nsd1904)

创建云主机类型

管理员-->云主机类型-->创建云主机类型

创建镜像

管理员-->镜像-->创建镜像(格式要一致)

创建网络

管理员-->网络-->创建网络-->(共享的及外部网络要打开)

三 : 客户端(项目管理用户)管理

登录nsd1904

创建网络

项目-->网络-->网络拓扑-->创建网络(共享的关闭)(网关一般选网段最后一个)

创建主机

项目-->计算-->云主机数量-->创建云主机

若出现云主机处于错误状态(No vaild host was found.)

解决办法：

可能是主机down掉

可能是内存不足

可能是内网出现了问题，检查内网，或者把内网删除（不会建立的可以参考案例4），重新建立，之后重新启动openstack

[root@openstack ~]# systemctl restart openstack-nova-compute

1006 报错:

[root@nova01 ~]# grep vncserver\_proxyclient\_address /etc/nova/nova.conf

# Deprecated group;name - DEFAULT;vncserver\_proxyclient\_address

#vncserver\_proxyclient\_address=127.0.0.1

vncserver\_proxyclient\_address=nova01

复制nova01写到nova /etc/hosts文件下

创建浮动IP

项目-->计算-->访问安全-->浮动IP-->创建浮动IP-->绑定到主机

项目-->计算-->访问安全-->安全组-->创建安全组

项目-->计算-->云主机数量-->主机列最后下拉菜单-->编辑安全组

做完这几步后物理机可以ping同云主机(用浮动ip)

四 : 节点扩容

环境准备:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CLOUD\*DAY04\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : Docker介绍

Docker 是完整的一套容器管理系统

Docker 不是虚拟化,没有系统

优点: 比虚拟化简洁高效

共享库和程序

不需要安装系统

缺点: 隔离没虚拟化强

共用Linux内核,安全性有缺陷

监控容器和容器排错难

SELinux难驾驭

二 : 部署Docker

2.1安装前准备:

需要64位系统

RHEL6.5以上,推荐7

关闭防火墙(会接管防火墙)

Docker软件在RHEL7-extras.iso中,先挂载到ftp上

mount -t iso9660 -o ro,loop RHEL7-extras.iso /var/ftp/extras/

2.2安装Docker:

[root@31 ~]vim /etc/yum.repos.d/local.repo //添加yum源

[extras]

name=extras

baseurl=ftp://192.168.1.254/extras

enabled=1

gpgcheck=0

[root@31 ~]yum -y install docker //需要修改yum源,添加ftp目录

[root@31 ~]ifconfig //启动后会对多出docker0网卡

[root@31 ~]systemctl start docker

[root@31 ~]systemctl enable docker

[root@31 ~]ifconfig

2.3 Docker镜像

镜像不一定要有系统,但一定要有数据

在Docker中容器是基于镜像启动的

镜像采用分层设计,使用快照的COW技术,确保不丢失数据

[root@31 ~]# docker images //查看当前系统有什么镜像

[root@31 ~]# docker search busybox //查找busybox镜像 docker search 查找镜像

[root@31 ~]# docker pull docker.io/busybox //下载镜像

[root@31 ~]# docker images

:(相当于文件路径) TAG 标签(相当于文件名) IMAGE ID CREATED SIZE

docker.io/busybox latest(默认标签) db8ee88ad75f 2 weeks ago 1.224 MB

[root@31 ~]# docker save docker.io/busybox:latest -o busybox.tar

//导出镜像(备份) 格式: docker save REPOSITORY:latest-o 文件名

[root@32 ~]# docker load -i busybox.tar //导入镜像(恢复)

[root@32 ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

docker.io/busybox latest db8ee88ad75f 2 weeks ago 1.224 MB

[root@room9pc01 ~]# cd /linux-soft/04/docker

[root@room9pc01 docker]# scp centos.tar nginx.tar redis.tar ubuntu.tar root@192.168.1.32:/root/ //拷贝镜像

19 docker load -i ubuntu.tar //恢复镜像

20 docker load -i nginx.tar

21 docker load -i centos.tar

22 docker load -i redis.tar

23 docker images

2.4 镜像基本操作

[root@31 ~]# docker run -it docker.io/centos:latest /bin/bash //创建 并 启动centos 镜像

//要看镜像内容启动,有的有操作系统,有的只是个服务

容器内运行的程序,物理机能查到,而虚拟机不可以

[root@1476e1688a9f /]# sleep 10000

[root@31 ~]# ps -ef | grep sleep

root 25324 25271 0 14:31 pts/1 00:00:00 sleep 10000 //物理机可杀掉此进程

命令列表:

docker images //查看镜像列表

[root@32 ~]# docker images

docker history 镜像名 //查看镜像制作历史

[root@32 ~]# docker history centos

docker inspect 镜像名 // 查看镜像底层信息

[root@32 ~]# docker inspect centos

docker pull //下载镜像

[root@32 ~]# docker pull docker.io/busybo

docker push //上传镜像

docker rmi //删除本地镜像

[root@32 ~]# docker rmi docker.io/nginx

[root@docker1 docker]# docker rmi nginx //nginx为镜像名,启动容器时删除镜像会失败，先删除容器,再删除镜像

docker save //镜像另存为tar包

[root@32 ~]# docker save docker.io/busybox:latest -o busybox.tar

docker load //使用tar包导入镜像

[root@32 ~]# docker load -i nginx.tar

docker search //搜索镜像

[root@32 ~]# docker search centos

docker tag //修改镜像名称和标签

[root@32 ~]# docker tag docker.io/centos:latest nginx:aaa

[root@docker1 docker]# docker ps -qa //a所有 ,q只显示id,显示所有的容器,包括没有启动的

2.5 Docker基本操作

循环删除 docker rm $(docker ps -qa) //删除所有容器

[root@0f ~]# yum -y install vim net-tools iproute psmisc bash-completion

//安装vim ifconfig IP命令 pstree tab键常用命令

1) 关闭容器

[root@docker1 docker]# docker stop 0f //0f为容器ID

2) 启动容器

[root@docker1 docker]# docker start 0f

3) 重启容器

[root@docker1 docker]# docker restart 0f

4) 删除容器(运行中删除不掉，先关闭容器)

[root@docker1 docker]# docker rm 0f //删除失败

Error response from daemon:

[root@docker1 docker]# docker stop 0f //关闭容器

[root@docker1 docker]# docker rm 0f //删除成功

[root@docker1 docker]# docker rm $(docker stop 0f) //或者这一条命令就行

5) 连接容器attach|exec (attach不杀死上帝进程退出用 Ctrl+p 后按q)

[root@docker1 docker]# docker attach 0f //exit后会关闭容器

[root@docker1 docker]# docker exec -it 0f /bin/bash //exit不会关闭容器

6) 查看容器底层信息

[root@docker1 docker]# docker inspect 0f //查看容器底层信息

7) 查看容器进程列表

[root@docker1 docker]# docker top 0f //查看容器进程列表

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COULD\*DAY05\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : 制作自定义镜像

1.1使用镜像启动容器

1）在该容器基础上修改yum源

[root@docker1 docker\_images]# docker run -it docker.io/centos

[root@8d07ecd7e345 /]# rm -rf /etc/yum.repos.d/\*

[root@8d07ecd7e345 /]# vi /etc/yum.repos.d/dvd.repo

2）安装测试软件

[root@8d07ecd7e345 /]# yum -y install net-tools iproute psmisc vim-enhanced

3）ifconfig查看

[root@8d07ecd7e345 /]# ifconfig

4)另存为另外一个镜像

1 创建新建镜像

[root@docker1 docker\_images]# docker start 8d07ecd7e345 //可以简写为8d，要保证唯一性

[root@docker1 docker\_images]# docker commit 8d07ecd7e345 docker.io/myos:latest //保存镜像

2 查看新建的镜像

[root@docker1 ~]# docker images

3 验证新建镜像

[root@docker1 docker\_images]# docker run -it docker.io/myos:latest

c7b4664bf /]# ifconfig

1.2使用Dockerfile文件创建一个新的镜像文件

Dockerfile语法格式：

– FROM:基础镜像

– MAINTAINER:镜像创建者信息(说明)

– EXPOSE:开放的端口

– ENV:设置环境变量

– ADD:复制文件到镜像

– RUN:制作镜像时执行的命令,可以有多个

– WORKDIR:定义容器默认工作目录

– CMD:容器启动时执行的命令,仅可以有一条CMD 格式 : CMD ["ls","-l","-a"]

1）创建一个Apache的镜像文件

[root@docker1 ~]# mkdir bulid

[root@docker1 ~]# cd bulid

[root@docker1 bulid]# touch Dockerfile //Dockerfile文件第一个字母要大写

[root@docker1 bulid]# cp /etc/yum.repos.d/local.repo ./

[root@docker1 bulid]# vi Dockerfile

FROM docker.io/centos:latest

RUN rm -rf /etc/yum.repos.d/\*.repo

ADD local.repo /etc/yum.repos.d/local.repo

RUN yum -y install vim net-tools iproute psmisc bash-completion

[root@docker1 bulid]# docker build -t test:latest . //后面有点,代表当前目录

[root@31 abc]# docker images //查看是否制作成功

[root@31 aaa]# vim Dockerfile //Dockerfile文件第一个字母要大写 //创建sshd服务镜像

FROM docker.io/myos:latest

RUN yum -y install openssh-server initscripts

RUN sshd-keygen

RUN echo "a" | passwd --stdin root

EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]

[root@31 aaa]# docker build -t myos:sshd .

[root@31 a]# vim Dockerfile //创建httpd服务镜像

FROM docker.io/myos:latest

RUN yum -y install httpd

WORKDIR /var/www/html //定义容器默认工作目录

ADD index.html index.html

EXPOSE 80 //设置开放端口号

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

[root@31 a]# docker build -t httpd:test .

[root@31 a]# docker run -itd httpd:test

二 : 创建私有镜像仓库

服务端 :

[root@31 ~]# yum -y install docker-distribution //装包

[root@31 ~]# systemctl start docker-distribution //启动

[root@31 ~]# systemctl enable docker-distribution

[root@31 ~]# ss -ltun //查看端口

[root@31 ~]# curl 192.168.1.31:5000/v2/ //访问

{}[root@31 ~]# //开头有大括号就成功

客户端 :

[root@32 ~]# vim /etc/sysconfig/docker

13 ADD\_REGISTRY='--add-registry 192.168.1.31:5000' //Docker仓库地址

24 INSECURE\_REGISTRY='--insecure-registry 192.168.1.31:5000' //允许非加密方式访问仓库

[root@32 ~]# systemctl restart docker //记得关闭所有在运行的docker

服务端将镜像上传到服务器 :

298 docker tag docker.io/busybox:latest 192.168.1.31:5000/busybox:latest

299 docker images

300 docker push 192.168.1.31:5000/busybox:latest

301 docker tag httpd:test 192.168.1.31:5000/httpd:test

302 docker images

303 docker push 192.168.1.31:5000/httpd:test

304 docker tag myos:sshd 192.168.1.31:5000/myos:sshd

305 docker push 192.168.1.31:5000/myos:sshd

306 docker tag docker.io/myos:latest 192.168.1.31:5000/myos:latest

307 docker push 192.168.1.31:5000/myos:latest

客户端可以直接启动镜像:

[root@32 ~]# docker run -itd myos:latest

[root@32 ~]# curl 192.168.1.31:5000/v2/\_catalog //查看有哪些镜像

{"repositories":["busybox","httpd","myos"]}

三 : 共享存储

共享存储端:

yum -y install nfs-utils //装包

vim /etc/exports //修改共享目录

/nfsdir \*(rw,no\_root\_squash)

systemctl start nfs-server //启动

systemctl enable nfs-server

showmount -e 192.168.1.33 //查看共享目录

exportfs -rv //查看共享目录

chmod 777 /nfsdir //默认权限是root的,nfs没权限

echo abcd > index.html //编写主页面

客户端:

yum -y install nfs-utils //装包

vim /etc/fstab //修改默认挂载文件

192.168.1.33:/nfsdir /nfsdir nfs defaults 0 0

mkdir /nfsdir

mount -a //重新挂载

docker run -itd -v /nfsdir/:/var/www/html 192.168.1.31:5000/httpd:test //创建docker -v将真机目录挂载到容器中

docker exec -it a1 /bin/bash //链接查看

curl 172.17.0.2

四 : 自定义网桥

不指定默认为docke0网桥

[root@31 ~]# docker run -it docker.io/myos:latest //创建容器,不指定默认为docke0网桥

[root@5050efc90f26 /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 172.17.0.3 netmask 255.255.0.0 broadcast 0.0.0.0

[root@31 ~]# docker run -it docker.io/myos:latest

[root@bcc492aeac8b /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 172.17.0.4 netmask 255.255.0.0 broadcast 0.0.0.0

[root@bcc492aeac8b /]# ping 172.17.0.3 //可以互通

创建docker1网桥,并指定创建容器

[root@31 ~]# docker network create --subnet=10.10.10.0/24 docker1

[root@31 ~]# docker run -it --network=docker1 docker.io/myos:latest

[root@61a4f7651c4b /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 10.10.10.2 netmask 255.255.255.0 broadcast 0.0.0.0

[root@31 ~]# docker run -it --network=docker1 docker.io/myos:latest

[root@1a4a3b4b71d1 /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 10.10.10.3 netmask 255.255.255.0 broadcast 0.0.0.0

[root@1a4a3b4b71d1 /]# ping 10.10.10.2 //可以通

[root@1a4a3b4b71d1 /]# ping 172.17.0.3 //失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COURSE\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : ansible介绍

是IT自动化和DevOps软件,基于Python开发,批量管理机器的工具

自动化部署APP,自动化管理配置项,自动化持续交付,自动化云服务管理

两大块 : 主机集合,命令集合

二 : 安装ansible

要求Python 2.6或2.7版本

开启ssh服务

2.1 真机配置yum仓库

[student@room9pc01~]$ mkdir /var/ftp/ansible

[student@room9pc01 ~]$ cd /linux-soft/04

[student@room9pc01 ansible]$ cp \* /var/ftp/ansible/

[student@room9pc01 ansible]$ createrepo /var/ftp/ansible/

2.2 修改主机名（容易区分，6台机器都需要修改）,及hosts文件(仅ansible修改就可以)

[root@ansible ~]# echo xxx > /etc/hostname

[root@ansible ~]# vim /etc/hosts

2.3 配置yum源，在管理节点ansible上面配置

[local]

name=local

baseurl="ftp://192.168.1.254/ansible"

enabled=1

gpgcheck=0

2.4 安装ansible

[root@ansible ~]# yum -y install ansible

[root@ansible ~]# ansible --version

ansible 2.4.2.0 //显示版本说明安装成功

config file = /etc/ansible/ansible.cfg //配置文件

configured module search path = [u'/root/.ansible/plugins/modules',

u'/usr/share/ansible/plugins/modules'] //配置的模块搜索路径

ansible python module location = /usr/lib/python2.7/site-packages/ansible //模块位置

executable location = /usr/bin/ansible //命令位置

python version = 2.7.5 (default, Apr 11 2018, 07:36:10) [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)]

三 : 配置ansible

3.1 [root@ansible ~]# vim /etc/ansible/ansible.cfg //配置文件

14 inventory = /etc/ansible/hosts //定义主机集合(默认注释掉了)

61 host\_key\_checking = False //(默认注释了)为True则等待输入yes,为False不需要输入yes

3.2 [root@ansible ~]# vim /etc/ansible/hosts //定义主机集合

[web]

web[1:2] //多个连续的写法aaa[x:y]

[db]

db1

db2

[other]

cache

3.3 列出要执行的主机

[root@ansible ansible]# ansible web --list-hosts //列出web组的主机

[root@ansible ansible]# ansible db,web --list-hosts //列出web组及db组的主机

[root@ansible ansible]# ansible all --list-hosts //列出所有主机

[root@ansible ansible]# ansible db,cache --list-hosts //列出db组及cache主机

3.4 批量检测主机

[root@ansible ansible]# ansible web -m ping -k //-k 使用交互式登录密码 ping代表ssh链接主机

3.5部署证书文件(每次登陆输密码太麻烦)

[root@ansible myansible]# cd /root/.ssh/

[root@ansible .ssh]# ssh-keygen -t rsa -b 2048 -N '' -f key //创建密钥

[root@ansible .ssh]# ssh-copy-id -i key.pub 主机名称

[root@ansible .ssh]# ansible all -m ping //失败

[root@ansible ~ ]# vim /etc/ansible/hosts

[all:vars] //指定ssh秘钥文件路径及名称

ansible\_ssh\_private\_key\_file="/root/.ssh/key"

[root@ansible .ssh]# ansible all -m ping //成功

[root@ansible .ssh]# ssh -i key cache //不需要输入密码,可以直接登陆

[root@db2 ~]# vim /etc/ssh/sshd\_config //改变ssh端口客户端修改

17 Port 222

[root@db2 ~]# systemctl restart sshd //重启

[root@ansible ~]# vim /etc/ansible/hosts //修改端口

db2 ansible\_ssh\_port=222

[root@ansible ~]# rm -rf /root/.ansible/cp/ //清除缓存

[root@ansible ~]# ansible all -m ping

四 : 主机定义及分组

[root@ansible ~]# vim /etc/ansible/hosts

[app:children] //定义分组app 里面包含web组和db组 ,格式[组名:children]

web

db

[root@ansible ~]# ansible app --list-hosts

五 : 练习理解批量执行

5.1 ansible-doc模块 : 模块的手册,相当于shell的man帮助

ansible-doc -l //列出所有模块

ansible-doc 模块名 //查看帮助

[root@ansible ~]# ansible-doc yum //查看yum模块帮助

5.2 ping模块 : 测试网络连通性(ssh的),ping模块没有参数

ansible host-pattern -m ping

[root@ansible ~]# ansible all -m ping

5.3 command模块 : 默认模块远程执行命令

ansible host-pattern -m command -a '[args]' //中括号里为参数

[root@ansible ~]# ansible all -m command -a 'uptime' //查看系统负载

5.4 shell模块 : 用法与command一样,区别是可以执行任意命令(除了交互式命令,例如vim top)

[root@ansible ~]# ansible web1,db2 -m shell -a 'cat /etc/passwd | grep root'

双引号与单引号的区别

[root@ansible ~]# ansible web -m shell -a "echo ${HOSTNAME}"

web1 | SUCCESS | rc=0 >>

ansible

web2 | SUCCESS | rc=0 >>

ansible

[root@ansible ~]# ansible web -m shell -a 'echo ${HOSTNAME}'

web2 | SUCCESS | rc=0 >>

web2

web1 | SUCCESS | rc=0 >>

web1

ansible命令式二次解析,第一次本机,第二次执行机器需要第二次解析的变量要转义(\)

切换目录用chdir代替cd命令

ansible cache -m shell -a 'chdir=/tmp touch testfile'

添加用户并设置密码:

[root@ansible ~]# ansible web1,db2 -m shell -a "useradd nb"

[root@ansible ~]# ansible web1,db2 -m shell -a "echo 123 | passwd --stdin nb"

5.5 script模块 : 在本地写脚本,然后使用script模块批量执行

添加用户wk,不能与nb在同一台主机出现

[root@ansible ~]# vim user.sh

#!/bin/bash

id nb

if [ $? != 0 ];then

useradd wk

echo 456 | passwd --stdin wk

fi

[root@ansible ~]# ansible web -m script -a './user.sh'

5.6 yum模块管理软件包 : name:要安装软件包名字(多个用逗号分割) state:动作 install ==installed remove == removed

service模块(启动服务) :name:服务名称 enabled : 是否开机启动yes|no state :启动started 停止stopped重启restarted

重新加载reloaded sleep: 执行restarted,会在stop和start之间停几秒

copy模块 :复制文件到远程主机 src本地文件路径 dest:远程主机绝对路径

[root@ansible ~]# ansible all -m copy -a 'src=/etc/resolv.conf dest=/etc/resolv.conf' //拷贝到远程

[root@ansible ~]# ansible all -m copy -a 'src=/etc/yum.repos.d/ dest=/etc/yum.repos.d/'

[root@ansible ~]# ansible db -m yum -a 'name="mariadb-server" state=installed' //批量安装mariadb

[root@ansible ~]# ansible db -m service -a 'name="mariadb" enabled="yes" \

> state="started"' //启动

[root@ansible ~]# ansible db -m shell -a 'ss -nualtp | grep 3306'

[root@ansible ~]# vim my.cnf //修改配置文件

[root@ansible ~]# ansible db -m service -a 'name="mariadb" enabled="yes" \

> state="restarted"' //重启mariadb

5.7 lineinfile模块 : 类似sed的一种行编辑替换模块

path:目标文件路径 regexp:正则表达式,要修改的行 line: 最终修改的结果

replace模块 : path:目标文件路径 regexp:正则表达式,要修改的关键词 replace : 替换后的结果

[root@ansible ~]# ansible db -m lineinfile -a 'path="/etc/my.cnf" regexp="^binlog-"

line="binlog-format = row"' //修改整行,先删除后添加

[root@ansible ~]# ansible db -m shell -a 'grep -P "^(server|binlog-)" /etc/my.cnf' //查看结果

[root@ansible ~]# ansible db -m replace -a 'path="/etc/my.cnf"

regexp="row$" replace="mixed"' //替换

[root@ansible ~]# ansible db -m shell -a 'grep -P "^(server|binlog-)" /etc/my.cnf' //查看结果

5.8 setup模块 : 主要用于获取 主机信息 filter : 过滤所需信息

[root@ansible ~]# ansible cache -m setup -a 'filter=ansible\_hostname'

cache | SUCCESS => {

"ansible\_facts": {

"ansible\_hostname": "cache"

},

"changed": false

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COURSE\*DAY02\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : ansible七种武器

1 ansible命令,用于执行临时性工作

2 ansible-doc ,是ansible模块的文档说明,帮助信息

3 ansible-console ,为用户提供的交互式工具

4 ansible-galaxy ,从github上下载管理的工具,与Python的pip\类似

5 ansible-playbook,日常工作中使用频率最高的命令,工作机制 :通过事先编好的文件批量管理可以理解为一定条件组成的任务集

6 ansible-vault 用于配置文件加密,即可加密又可解密,不太实用

7 ansible-pull ansible有两种工作模式 pull|push,工作机制正好相反.push主动下发,pull需要客户机自己主动获取

二 : JSON简介

是JavaScript对象表示法,是一种基于文本独立与语言的轻量级数据交换格式

分隔符a ' () [] {} : ,

JSON特性: 纯文本 自我描述性(人类可读) 具有层级结构 可通过JavaScript解析

语法规则:

数据在名称/值对中

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:

名称/值对包括字段名称(在双引号中),后面一个冒号,然后是值 "诗仙":"李白"

数组语法:

{"诗人"

["a","b","c","d"]

}

符合复杂类型:

{"诗人":

[{a:b,c:d}

{e:f,g;h}

{r:o,t:s}

]

}

三: YAML简介

基础语法 :

结构通过空格来表示

数组使用"- "来表示

键值使用": "来表示

YAML使用一个固定的缩进风格来表示层级结构关系

一般每个缩进级别用两个空格组成

# 表示注释

哈希数组符合表达式例子:

"诗人":

- "李白"

- "杜甫"

- "白居易"

- "李贺"

高级复合表达式例子:

"诗人":

-

"李白":"诗仙"

"年代":"唐"

-

"杜甫":"诗圣"

"年代":"唐"

-

"白居易":"诗魔"

"年代":"唐"

四 : jinja2介绍

是基于Python的模板引擎,包含变量和表达式两部分

playbook的模板使用Python的jinja2模板来处理

模板表达式包含在分隔符"{{ }}"内

控制语句包含在"{% %}"内

注释包含在"{# #}"内

调用变量 {{变量名}}

计算 {{2+3}}

判断 {{1 in [1,2,3]}}

过滤器 "|" 前一个输出作为下一个输入

jinja2过滤器在线文档

http://docs.jinkan.org/docs/jinja2/templates.html

五 : playbook练习

[root@ansible ~]# vim ping.yml

---

- hosts: all

remote\_user: root

tasks:

- ping:

[root@ansible ~]# ansible-playbook ping.yml

练习一 : 安装httpd,修改端口8080 设置默认主页,启动服务,开机自启

---

- hosts: web

remote\_user: root

tasks:

- name: anzhuang

yum:

name: httpd

state: installed

- name: zhuyemian

shell:

echo "hello world" > /var/www/html/index.html

- name: xiugai8080

lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: ^Listen

line: Listen 8080

- name: qidong

service:

name: httpd

enabled: yes

state: started

练习二 : 创建用户利用传参

[root@ansible ~]# vim user.yml

---

- hosts: web

remote\_user: root

tasks:

- name: adduser "{{username}}"

user:

name: "{{username}}"

group: users

password: "{{'123qqq...A'|password\_hash('sha512')}}"

[root@ansible ~]# ansible-playbook user.yml -e '{"username": plj}' //使用json方式传参

[root@ansible ~]# vim args.yml

---

username:

plj

[root@ansible ~]# ansible-playbook user.yml -e @args.yml //使用yaml方式传参@链接文件(需提前编写)

练习三: tags标示练习

[root@ansible ~]# vim http\_conf.yml //编辑yaml文件

---

- hosts: web

remote\_user: root

tasks:

- copy:

src: httpd.conf

dest: /etc/httpd/conf/httpd.conf

owner: root

group: root

mode: 0644

tags: conf //添加标签

notify: //触发操作

- roload apache

- copy:

src: index.html

dest: /var/www/html/index.html

owner: apache

group: apache

mode: 0644

tags: index

handlers: //定义触发操作

- name: reload apache

service:

name: httpd

state: restarted

[root@ansible ~]# ansible-playbook http\_conf.yml -t conf //运行测试

[root@ansible ~]# ansible-playbook http\_conf.yml -t index //运行测试

练习四: when register 当系统负载大于0.5时关闭httpd服务

[root@ansible ~]# vim load.yml

---

- hosts: web

remote\_user: root

tasks:

- shell: uptime | awk '{printf("%.2f",$(NF-2))}'

register: result

- service:

name: httpd

state: stopped

when: result.stdout|float > 0.5

- name: show debug info //显示debug详细信息

debug: var=result //显示变量result详细信息

[root@ansible ~]# ansible-playbook load.yml

练习五: with\_item 创建多用户

---

- hosts: web

remote\_user: root

tasks:

- name: adduser "{{item}}"

user:

name: "{{item}}"

group: users

password: "{{'123qqq...A'|password\_hash('sha512')}}"

with\_items: ["aa","bb","cc","dd"] //逐一创建用户

with\_item进阶: 结合JSON批量添加用户,设置密码及加入到某一个组中

---

- hosts: web

remote\_user: root

tasks:

- name: adduser "{{item.name}}"

user:

name: "{{item.name}}"

group: "{{item.group}}"

password: "{{'123qqq...A'|password\_hash('sha512')}}"

with\_items:

- {name: 'ee',group: 'root'}

- {name: 'ff',group: 'bin'}

- {name: 'gg',group: 'ftp'}

- {name: 'hh',group: 'adm'}

练习六: debug调试模块

显示受到影响的主机 --list-hosts

[root@ansible ~]# ansible-playbook http\_conf.yml -t conf --list-hosts

显示工作的task(命令) --list-tasks

[root@ansible ~]# ansible-playbook http\_conf.yml -t conf --list-task

显示将要运行的tag --list-tags

[root@ansible ~]# ansible-playbook http\_conf.yml -t index --list-tags

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COURSE\*DAY03\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : ELK介绍

1.1ELK和lnmp的关系

L N M P

K(日志可视化) E(日志检索储存) L(日志分析处理)

1.2 相关概念

Node: 装有ES服务器的节点

Cluster: 多个Node组成的集群

Document: 可被搜索的基础信息单元

Index: 拥有相似特征的文档集合

Type: 一个索引中可以定义一种或多种类型

Filed: ES最小单位,相当于数据每一列

Shard是: 索引的分片,每一个分片就是Shard

Reolicas: 索引的拷贝

二 : ES集群安装

[root@room9pc01 elk]# mkdir /var/ftp/elk //创建ftp目录

[root@room9pc01 elk]# createrepo /var/ftp/elk //更新yum源

[root@es1 ~]# vim /etc/hosts

192.168.1.51 es1

192.168.1.52 es2

192.168.1.53 es3

192.168.1.54 es4

192.168.1.55 es5

[root@es1 ~]# vim /etc/yum.repos.d/local.repo //添加yum源

[elk]

name=CentOS-$releasever - elk

baseurl="ftp://192.168.1.254/elk"

enabled=1

gpgcheck=0

[root@es1 ~]# yum -y install java-1.8.0-openjdk elasticsearch //安装依赖包及软件包

[root@es1 ~]# vim /etc/elasticsearch/elasticsearch.yml //修改配置文件

17 cluster.name: nsd1904 //配置集群名字

23 node.name: es1 //当前主机名称

54 network.host: 0.0.0.0 // 0.0.0.0（监听所有地址）

68 discovery.zen.ping.unicast.hosts: ["es1", "es2","es3",] //声明集群里的主机成员有谁，不需要全部写进去

[root@es1 ~]# systemctl start elasticsearch

[root@es1 ~]# systemctl enable elasticsearch

[root@es1 ~]# ss -nualtp | grep 9200

[root@es1 ~]# ss -nualtp | grep 9300

三 : ES插件使用

3.1 真机共享插件

[root@room9pc01 elk]# scp bigdesk-master.zip elasticsearch-kopf-master.zip

elasticsearch-head-master.zip root@192.168.1.55:/root/

3.2 安装插件

head插件: ES集群的拓扑结构,并且可以通过它进行索引和节点级别的操作

kopf插件: 是一个Elasticsearch管理工具,提供了对ES集群操作的API

bigdesk插件: 集群监控工具,用来查看ES集群的各种状态

[root@es5 bin]# ./plugin install file:///root/elasticsearch-head-master.zip //安装head插件

[root@es5 bin]# ./plugin install file:///root/bigdesk-master.zip //安装bigdesk插件

[root@es5 bin]# ./plugin install file:///root/elasticsearch-kopf-master.zip //安装kopf插件

[root@es5 bin]# /usr/share/elasticsearch/bin/plugin list //列出已安装插件

也可网络安装

[root@es5 bin]# ./plugin install ftp://192.168.1.254/elk/elasticsearch-head-master.zip

四 : RESTful API

4.1 HTTP协议简介

请求行 消息报头 请求正文

http请求方法 GET POST HEAD OPTIONS PUT DELETE TRACE CONNECT

ES常用:

PUT --增 DELETE--删 POST--改 GET--查

4.2 RESTful API调用

-A 修改请求agent

-X 设置请求方法

-i 显示返回头信息

\_cat命令

curl http://es5:9200/\_cat/ //查看所有命令

curl http://es5:9200/\_cat/master?v //?v显示详细信息

curl http://es5:9200/\_cat/master?help //?help显示帮助信息

添加数据:

增加数据,使用PUT方法

调用方式: 数据库地址/索引/类型/id值

[root@es5 ~]# curl -X PUT http://es5:9200/tedu/student/1 -d '

> {

> "姓名": "dc",

> "爱好": "dcc",

> "年龄": "25"

> }'

修改数据:

修改数据,使用POST方法

在修改数据时必须调用\_update关键字

调用方式:数据库地址/索引/类型/id值/\_update

[root@es5 ~]# curl -XPOST http://es5:9200/tedu/teacher/1/\_update -d '

> {

> "doc":{

> "年代": "唐代"

> }

> }'

查询数据:

curl -X GET http://es5:9200/tedu/teacher/1

删除数据:

curl -X DELETE http://es5:9200/tedu/teacher/1

删除索引:

curl -X DELETE http://192.168.1.55:9200/tedu/

curl -X DELETE http://192.168.1.65:9200/\* //删除所有索引

4.3 批量导入数据

[root@es5 ~]# curl -XPOST http://es5:9200/\_bulk --data-binary @logs.json //调用\_bulk关键字导入

[root@es5 ~]# curl -XPOST http://es5:9200/\_bulk --data-binary @shakespeare.json

[root@es5 ~]# curl -XPOST http://es5:9200/oo/xx/\_bulk --data-binary @accounts.json

//因为备份文件没有索引和类型,所以要手动指定

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COURSE\*DAY04\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : kibana安装

是一款图形化查看软']

[root@kibana ~]# vim /etc/hosts

192.168. 1.51 es1

192.168.1.52 es2

192.168.1.53 es3

192.168.1.54 es4

192.168.1.55 es5

192.168.1.56 kibana

[root@kibana ~]# vim /etc/yum.repos.d/local.repo //配置yum源

[elk]

name=CentOS-$releasever - elk

baseurl="ftp://192.168.1.254/elk"

enabled=1

gpgcheck=0

[root@kibana ~]# yum -y install kibana

[root@kibana ~]# vim /opt/kibana/config/kibana.yml //修改配置文件

2 server.port: 5601 //若把端口改为80，可以成功启动kibana，但ss时没有端口，没有监听80端口，服务里面写死了，不能用80端口，只能是5601这个端口

5 server.host: "0.0.0.0" //服务器监听地址

15 elasticsearch.url: "http://192.168.1.51:9200" //配置了hosts文件可以写主机名,声明地址，从哪里查

23 kibana.index: ".kibana" //kibana自己创建的索引,名字

26 kibana.defaultAppId: "discover" //打开kibana页面时，默认打开的页面discover

53 elasticsearch.pingTimeout: 1500 //ping检测超时时间

57 elasticsearch.requestTimeout: 30000 //请求超时

64 elasticsearch.startupTimeout: 5000 //启动超时

[root@kibana ~]# systemctl start kibana

[root@kibana ~]# ss -nulatp | grep 5601

浏览器访问192.168.1.56:5601 查看状态

二 : logstash 配置

是一个数据采集,加工处理及传输的工具

logstash类型及条件判断

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: options => {k => "v",k2 => "v2"}

编码解码: codec => "json"

路径: file\_path => "/tmp/filename"

注释: #

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: !~

条件判断:

包含: in

不包含: not in

与: and

或: or

非与: nand

非或: xor

复合表达式: ()

取反符合: !()

[root@logstash ~]# vim /etc/yum.repos.d/local.repo

[local\_reo]

name=CentOS-$releasever- Base

baseurl="ftp://192.168.1.254/elk"

enabled=1

gpgcheck=0

[root@logstash ~]# yum -y install java-1.8.0-openjdk

[root@logstash ~]# yum -y install logstash

[root@logstash ~]# vim /etc/hosts

192.168.1.51 es1

192.168.1.52 es2

192.168.1.53 es3

192.168.1.54 es4

192.168.1.55 es5

192.168.1.56 kibana

192.168.1.57 web

192.168.1.58 logstash

[root@logstash ~]# /opt/logstash/bin/logstash-plugin list //查看已安装插件

2.1 编写配置文件

codec类插件常用插件:

plain json json\_lines rubydebug multiline

2.1.1 codec类:output rubydebug插件

[root@logstash bin]# vim /etc/logstash/logstash.conf

input{

stdin{codec => "json" } //数据输入,加载json模块(输入json格式)

}

filter{ } //数据处理

output{

stdout{ codec => "rubydebug" } //数据输出,加载调试模块(输出json格式)

}

[root@logstash ~]# /opt/logstash/bin/logstash -f /etc/logstash/logstash.conf //启动服务

2.1.2 codec类: input file插件

[root@logstash bin]# vim /etc/logstash/logstash.conf

input{

file { //用file模块

path => ["/tmp/a.log","/var/log/b.log"] //监控文件列表

type => "testlog" //类型名称,输出时添加,标签,便于识别

start\_position => "beginning" //配置第一次读取文件从什么地方开始,默认从结尾开始(易丢失数据)

sincedb\_path => "/var/lib/logsyash/sincedb-access" //记录读取文件的位置(默认在用户家目录,多用户时不方便,易丢数据)

}

}

filter{ }

output{

stdout{ codec => "rubydebug" }

}

[root@logstash ~]# /opt/logstash/bin/logstash -f /etc/logstash/logstash.conf //启动服务

2.1.3 codec类: filter grok插件

解析各种非结构化的日志数据文件

grok使用正则表达式把非结构化的数据结构化

适用性强,几乎可用于各类数据

grok{

match => ["message", (?<key>reg), "%{IP:clientip}"]

}

[root@logstash ~]# vim /etc/logstash/logstash.conf

input{

file {

path => ["/tmp/a.log"]

type => "testlog"

start\_position => "beginning"

sincedb\_path => "/dev/null"

}

}

filter{

grok {

match => { "message" => "%{COMBINEDAPACHELOG}" } //COMBINEDAPACHELOG为官方写好的模本,适用性差

}

}

output{

stdout{

codec => "rubydebug"

}

}

//COMBINEDAPACHELOG为官方写好的模本,适用性差,全部的在下面文件里

[root@logstash bin]# vim /opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-2.0.5

/patterns/grok-patterns

2.1.4 codec类 filebeat插件(监控其他机器)

被监控主机:

[root@web ~]# vim /etc/yum.repos.d/local.repo

[local\_rep]

name=CentO-$releasever - Base

baseurl="ftp://192.168.1.254/elk"

enabled=1

gpgcheck=0

[root@web ~]# yum -y install filebeat

[root@web ~]# vim /etc/filebeat/filebeat.yml

14 paths:

15 - /var/log/httpd/access\_log //日志的路径，短横线加空格代表yml格式

72 document\_type: apachelog //文档类型(名称自定义.有标示性)

183 #elasticsearch: //加上注释

188 #hosts: ["localhost:9200"] //加上注释

278 logstash: //去掉注释

280 hosts: ["192.168.1.58:5044"] //去掉注释,logstash那台主机的ip

[root@web ~]# systemctl start filebeat

[root@logstash logstash]# vim /etc/logstash/logstash.conf

input{

stdin {codec => "json"}

beats {

port => 5044

}

}

filter{

if [type] == "apachelog" { //配置文件定义的apachelog

grok {

match => { "message" => "%{COMBINEDAPACHELOG}" } //利用官方写好的

}}

}

output{

stdout{codec => "rubydebug" }

if [type] == "apachelog"{

elasticsearch {

hosts => ["192.168.1.51:9200","192.168.1.52:9200"]

index => "apachelog"

flush\_size => 2000

idle\_flush\_time => 10

}}

}

[root@logstash ~]# netstat -antup | grep 5044

[root@logstash logstash]# /opt/logstash/bin/logstash -f /etc/logstash/logstash.conf

[root@logstash ~]# curl web

浏览器访问Elasticsearch，有apachelog

[student@room9pc01 ~]$ firefox http://192.168.1.55:9200/\_plugin/head

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COURSE\*DAY05\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

大数据特性: 5V

Volume大体量 Variety多样性 Velocity时效性 Veracity准确性 Value大价值

一 : hadoop介绍

hadoop常用组件:

HDFS: Hadoop分布式文件系统(核心组件)

MapReduce: 分布式计算框架(核心组件)

Yarn: 集群资源管理系统(核心组件)

Zookeeper: 分布式协作系统

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角色及概念: 数据储存的基础,高度容错的系统,在低成本通用硬件上运行

NameNode: 管理名称空间和数据块映射信息,配置副本策略,处理客户端请求 //fsimages: 名称空间 fsedit: 文件变更日志

Secondary NameNode: 专门存储合并日志的,(NameNode秘书)

DataNode: 存储数据,汇报存储信息给NameNode

Client: 切分文件,访问NameNode和DataNode

MapReduce角色及概念: JAVA实现的分布式计算框架

JobTracker: master节点,只有一个分解任务,派发给TaskTracker

TaskTracker: Slave节点,多台运行Map Task和ReduceTask 与JobTracker交互,汇报任务状态

Map Task: 解析每条数据记录,传递给用户编写的map()并执行,将输出结果写入本地磁盘

Reduce Task: 从Map Task执行结果中,远程读取输入数据,对数据排序,传递给用户编写的reduce函数执行

Yarn结构: Hadoop的一个通用的资源管理系统

ResourceManager: 处理客户端请求,启动/监控ApplicationMaster,监控NodeManager,资源分配与调度

NodeManager: 单个节点资源管理,处理来自ApplicationMaster和ResourceManager的命令

Container: 多维资源及环境变量,启动命令等运行相关的信息资源调度分配

ApplicationMaster: 数据切分,申请资源进行分配,任务监控与容错

Client: 交互,监控状态,提交杀死应用程序

二 : 安装hadoop(单机模式)

[root@nn01 ~]# yum install java-1.8.0-openjdk-devel //安装环境

[root@nn01 ~]# tar -xf hadoop-2.7.7.tar.gz //解压hadoop

[root@nn01 ~]# mv hadoop-2.7.7 /usr/local/hadoop //移动

[root@nn01 ~]# jps

22674 Jps

[root@nn01 ~]# rpm -ql java-1.8.0-openjdk //查看javaopenjdk安装目录

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86\_64/jre/

[root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh //编写hadoop配置文件

25 export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86\_64/jre/ //java安装路径

33 export HADOOP\_CONF\_DIR=${HADOOP\_CONF\_DIR:-"/usr/local/hadoop/etc/hadoop"} //hadoop配置文件目录

[root@nn01 ~]# ./hadoop

[root@nn01 ~]# ./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar wordcount input output

命令 | 固定格式 | 参数 | 分析的文件夹| 输出的文件夹

三 : 环境准备

hadoop官方文档: http://hadoop.apache.org/docs/r2.7.7/

3.1 新的机器装环境及hosts文件:

[root@node1 ~]# yum install java-1.8.0-openjdk-devel //新的机器装环境及hosts文件

[root@nn01 hadoop]# vim /etc/hosts

192.168.1.60 nn01

192.168.1.61 node1

192.168.1.62 node2

192.168.1.63 node3

/etc/ssh/ssh\_config

StrictHostKeyChecking no //不要yes

3.2 [root@nn01 .ssh]# cd /root/.ssh/ //配置免密码登录(所有)

[root@nn01 .ssh]# ssh-keygen

[root@nn01 .ssh]# ssh-copy-id (node1,node2,node3 nn01)

[root@nn01 .ssh]# ssh node1

四 : 安装hadoop完全分布式

HDFS完全分布式系统配置

环境变量配置文件: hadoop-env.sh

核心配置文件: core-site.xml

HDFS配置文件: hdfs-site.xml

节点配置文件: slaves

4.1 编写hadoop配置文件:

[root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh //编写hadoop配置文件

25 export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86\_64/jre/ //java安装路径

33 export HADOOP\_CONF\_DIR=${HADOOP\_CONF\_DIR:-"/usr/local/hadoop/etc/hadoop"} //hadoop配置文件目录

固定格式:

<property>

<name></name>

<value></value>

</property>

4.2 修改核心配置文件:

官方文档位置: http://hadoop.apache.org/docs/r2.7.7/hadoop-project-dist/hadoop-common/core-default.xml

[root@nn01 hadoop]# vim /usr/local/hadoop/etc/hadoop/core-site.xml //修改核心配置文件

<configuration> //固定格式

<property>

<name>fs.defaultFS</name> //文件系统配置参数

<value>hdfs://nn01:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name> //数据目录配置参数(数据根目录)

<value>/var/hadoop</value>

</property>

</configuration>

4.3 修改HDFS配置文件:

官方文档位置: http://hadoop.apache.org/docs/r2.7.7/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml

[root@nn01 hadoop]# vim /usr/local/hadoop/etc/hadoop/hdfs-site.xml //修改HDFS配置文件

<configuration>

<property>

<name>dfs.namenode.http-address</name> //namenode地址声明

<value>nn01:50070</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name> //secondary.namenode地址声明

<value>nn01:50090</value>

</property>

<property>

<name>dfs.replication</name> //文件冗余份数

<value>2</value>

</property>

</configuration>

4.4 修改节点配置文件:

[root@nn01 hadoop]# vim /usr/local/hadoop/etc/hadoop/slave //节点配置文件,添加DataNode节点主机名称

node1 //只写主机名

node2

node3

4.5 同步配置

[root@nn01 hadoop]# for i in node{1..3}

> do

> rsync -aXSH --delete /usr/local/hadoop ${i}:/usr/local/ & //老师写的,利用rsync同步

> done

4.6 启动集群

[root@nn01 ~]# cd /usr/local/hadoop/sbin/

[root@nn01 sbin]# ./start-dfs.sh //启动

[root@nn01 sbin]# ./stop-dfs.sh //停止

4.7 查看状态

[root@node1 ~]# jps

23249 Jps

23074 DataNode //成功,没有就要排错

[root@node1 hadoop]# ./bin/hdfs dfsadmin -report //查看DataNode状态(能看到所有node)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COURSE\*DAY06\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : 文件配置

1.1 编辑分布式计算框架文件

[root@nn01 hadoop]# cd /usr/local/hadoop/etc/hadoop/

[root@nn01 hadoop]# mv mapred-site.xml.template mapred-site.xml

[root@nn01 hadoop]# vim mapred-site.xml //编辑分布式计算框架

<configuration>

<property>

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

<value>yarn</value> //只支持单机local和集群yarn

</property>

</configuration>

1.2 编辑资源管理文件

[root@nn01 hadoop]# vim yarn-site.xml //编辑资源管理文件

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name> //使用哪个计算框架

<value>mapreduce\_shuffle</value> //计算框架名称(要和开发人员沟通)

</property>

<!-- Site specific YARN configuration properties -->

<property>

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

<value>nn01</value>

</property>

</configuration>

1.3 同步配置文件

[root@nn01 hadoop]# for i in node{1..3}; do rsync -aXSH --delete /usr/local/hadoop ${i}:/usr/local/ & done

1.4 启动集群,验证状态

[root@nn01 hadoop]# /usr/local/hadoop/sbin/start-yarn.sh //启动集群

[root@nn01 hadoop]# jps //nn01验证

5203 SecondaryNameNode

5014 NameNode

5846 Jps

5582 ResourceManager

[root@node1 hadoop]# jps //node1验证

1809 Jps

1423 DataNode

1711 NodeManager

[root@node1 hadoop]# /usr/local/hadoop/bin/yarn node -list //查看是否组成集群

19/08/16 09:50:56 INFO client.RMProxy: Connecting to ResourceManager at nn01/192.168.1.60:8032

Total Nodes:3

Node-Id Node-State Node-Http-Address Number-of-Running-Containers

node2:35134 RUNNING node2:8042 0

node1:42069 RUNNING node1:8042 0

node3:32969 RUNNING node3:8042 0

1.5 web访问hadoop

http://192.168.1.60:50070/ //--namenode web页面（nn01）

http://192.168.1.60:50090/ //--secondory namenode web页面（nn01）

http://192.168.1.61:50075/ //--datanode web页面（node1,node2,node3）

http://192.168.1.60:8088/ //--resourcemanager web页面（nn01）

http://192.168.1.61:8042/ //--nodemanager web页面（node1,node2,node3）

二 : 集群使用

哪台都可以,只要有hadoop就行

[root@node1 hadoop]# ./bin/hadoop fs -mkdir /abc /集群中创建文件夹

[root@node1 hadoop]# ./bin/hadoop fs -ls / //查看

Found 1 items

drwxr-xr-x - root supergroup 0 2019-08-16 10:32 /abc

[root@node1 hadoop]# ./bin/hadoop fs -touchz /ooxx //创建文件(多z)

[root@node1 hadoop]# ./bin/hadoop fs -get /ooxx ./ //下载文件

[root@node1 hadoop]# ./bin/hadoop fs -put ./\*.txt /abc //上传文件

[root@node1 hadoop]# ./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar \

> wordcount /abc /bcd //分析/abc文件夹将结果放到/bcd文件夹里

[root@node1 hadoop]# ./bin/hadoop fs -ls /bcd //查看/bcd文件夹

[root@node1 hadoop]# ./bin/hadoop fs -cat /bcd/part-r-00000 //查看结果

[root@node1 hadoop]# ./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar \

> wordcount file:///etc/passwd /111 //分析本地文件passwd将结果放到HDFS的/111文件夹下

fs支持的命令:

[-appendToFile <localsrc> ... <dst>]

[-cat [-ignoreCrc] <src> ...]

[-checksum <src> ...]

[-chgrp [-R] GROUP PATH...]

[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]

[-chown [-R] [OWNER][:[GROUP]] PATH...]

[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]

[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

[-count [-q] [-h] <path> ...]

[-cp [-f] [-p | -p[topax]] <src> ... <dst>]

[-createSnapshot <snapshotDir> [<snapshotName>]]

[-deleteSnapshot <snapshotDir> <snapshotName>]

[-df [-h] [<path> ...]]

[-du [-s] [-h] <path> ...]

[-expunge]

[-find <path> ... <expression> ...]

[-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

[-getfacl [-R] <path>]

[-getfattr [-R] {-n name | -d} [-e en] <path>]

[-getmerge [-nl] <src> <localdst>]

[-help [cmd ...]]

[-ls [-d] [-h] [-R] [<path> ...]]

[-mkdir [-p] <path> ...]

[-moveFromLocal <localsrc> ... <dst>]

[-moveToLocal <src> <localdst>]

[-mv <src> ... <dst>]

[-put [-f] [-p] [-l] <localsrc> ... <dst>]

[-renameSnapshot <snapshotDir> <oldName> <newName>]

[-rm [-f] [-r|-R] [-skipTrash] <src> ...]

[-rmdir [--ignore-fail-on-non-empty] <dir> ...]

[-setfacl [-R] [{-b|-k} {-m|-x <acl\_spec>} <path>]|[--set <acl\_spec> <path>]]

[-setfattr {-n name [-v value] | -x name} <path>]

[-setrep [-R] [-w] <rep> <path> ...]

[-stat [format] <path> ...]

[-tail [-f] <file>]

[-test -[defsz] <path>]

[-text [-ignoreCrc] <src> ...]

[-touchz <path> ...]

[-truncate [-w] <length> <path> ...]

[-usage [cmd ...]]

三 : HDFS节点管理

3.1 增加新节点

1 添加新机器,配置环境

[root@node4 ~]# yum -y install java-1.8.0-openjdk-devel

2 编辑hosts文件

[root@nn01 ~]# vim /etc/hosts

192.168.1.64 node4

3 设置免密登录

[root@node4 ~]# ssh-copy

4 修改NameNode的slave文件增加node节点

[root@nn01 hadoop]# vim etc/hadoop/slaves

node4

5 拷贝文件

[root@nn01 hadoop]# for i in node{1..4}; do rsync -aXSH --delete /usr/local/hadoop ${i}:/usr/local/ & done

6 启动新节点,查看状态

[root@node4 sbin]# ./hadoop-daemon.sh start datanode //在新添加节点运行,启动DataNode命令

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -report //查看集群状态

Name: 192.168.1.61:50010 (node1)

Name: 192.168.1.62:50010 (node2)

Name: 192.168.1.63:50010 (node3)

Name: 192.168.1.64:50010 (node4)

7 设置同步带宽,同步数据

[root@nn01 hadoop]# ./bin/hdfs dfsadmin //直接回车看可使用命令

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -setBalancerBandwidth 50000000 //设置同步带宽,单位bytes

[root@nn01 hadoop]# ./sbin/start-balancer.sh //同步数据

3.2 删除节点

1 编辑配置文件

[root@nn01 hadoop]# vim etc/hadoop/hdfs-site.xml //编辑NameNode的hdfs-site.xml文件

<property>

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

<value>/usr/local/hadoop/etc/hadoop/exclude</value> //指定要删除的节点名称文件

</property>

[root@nn01 hadoop]# vim etc/hadoop/exclude //添加要删除的节点名

[root@nn01 hadoop]# vim etc/hadoop/slaves //删除node4

2 刷新数据

[root@nn01 hadoop]# ./bin/hdfs dfsadmin //查看命令

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -refreshNodes //更新数据刷新节点

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -report //查看节点状态

Decommission Status : Decommission in progress //数据正在迁移

Decommission Status : Decommissioned //数据迁移成功(出现这个才能对要删除的节点下线)

四 : YARN节点管理

[root@node4 hadoop]# ./sbin/yarn-daemon.sh start nodemanager //启动节点(添加节点)

[root@node4 hadoop]# ./sbin/yarn-daemon.sh stop nodemanager //删除节点

stopping nodemanager

[root@node4 hadoop]# ./bin/yarn node -list // 查看节点状态，还是有node4节点，要过一段时间才会消失

19/08/16 14:46:51 INFO client.RMProxy: Connecting to ResourceManager at nn01/192.168.1.60:8032

Total Nodes:5

Node-Id Node-State Node-Http-Address Number-of-Running-Containers

node2:35134 RUNNING node2:8042 0

node1:42069 RUNNING node1:8042 0

node3:32969 RUNNING node3:8042 0

node4:40707 RUNNING node4:8042 0

五 : NFS网关

5.1 添加用户

[root@nn01 hadoop]# groupadd -g 800 nfsuser

[root@nn01 hadoop]# useradd -u 800 -g 800 -r -d /var/hadoop nfsuser

[root@nn01 hadoop]# id nfsuser

uid=800(nfsuser) gid=800(nfsuser) 组=800(nfsuser)

[root@nfsgw ~]# groupadd -g 800 nfsuser

[root@nfsgw ~]# mkdir /usr/hadoop

[root@nfsgw ~]# useradd -u 800 -g 800 -r -d /var/hadoop nfsuser

[root@nfsgw ~]# id nfsuser

uid=800(nfsuser) gid=800(nfsuser) 组=800(nfsuser)

5.2 停止所有服务

[root@nn01 hadoop]# ./sbin/stop-all.sh

[root@nn01 hadoop]# jps //查看

8394 Jps

5.3 修改配置

[root@nn01 hadoop]# vim etc/hadoop/core-site.xml //修改配置

<configuration>

......

<property>

<name>hadoop.proxyuser.nfsuser.groups</name> //挂载点用户所使用的组

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.nfsuser.hosts</name> //挂载点主机地址

<value>\*</value>

</property>

</configuration>

5.4 同步配置到所有主机

[root@nn01 hadoop]# for i in node{1..3}; do rsync -aXSH --delete /usr/local/hadoop/etc \

> ${i}:/usr/local/hadoop & done //只同步配置文件的命令

5.5 启动hdfs

[root@nn01 hadoop]# ./sbin/start-dfs.sh //启动集群

[root@nn01 hadoop]# jps //查看集群

9383 NameNode

9578 SecondaryNameNode

9803 Jps

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -report //查看器群节点

Live datanodes (3):

Name: 192.168.1.61:50010 (node1)

Name: 192.168.1.62:50010 (node2)

Name: 192.168.1.63:50010 (node3)

六 : NFSGW配置

6.1 环境配置

卸载rpcbind nfs-utils

rpm -qa | grep rpcbind

rpm -qa | grep nfs

配置hosts文件

[root@nfsgw ~]# vim /etc/hosts

192.168.1.60 nn01

192.168.1.61 node1

192.168.1.62 node2

192.168.1.63 node3

192.168.1.65 nfsgw

安装JAVA运行环境

yum -y install java-1.8.0-openjdk-devel

同步NameNode的hadoop文件夹到本机

[root@nfsgw ~]# rsync -av 192.168.1.60:/usr/local/hadoop /usr/local/

6.2 修改配置

配置文件hdfs-site.xml

[root@nfsgw hadoop]# vim etc/hadoop/hdfs-site.xml

<configuration>

......

<configuration>

<property>

<name>nfs.exports.allowed.hosts</name> //共享文档权限(挂载用户)

<value>\* rw</value>

</property>

<property>

<name>nfs.dump.dir</name> //文件转储目录

<value>/var/nfstmp</value> //指定文件路径

</property>

</configuration>

创建文件夹,修改所有权

[root@nfsgw hadoop]# mkdir /var/nfstmp

[root@nfsgw hadoop]# chown nfsuser:nfsuser /var/nfstmp

[root@nfsgw hadoop]# ls -ld /var/nfstmp

drwxr-xr-x 2 nfsuser nfsuser 6 8月 16 16:41 /var/nfstmp

6.3 NFS启动与挂载

NFS启动

[root@nfsgw logs]# rm -rf /usr/local/hadoop/logs/\*

[root@nfsgw hadoop]# setfacl -m user:nfsuser:rwx logs/ //设置nfsuser可读权限

[root@nfsgw hadoop]# getfacl logs/

# file: logs/

# owner: root

# group: root

user::rwx

user:nfsuser:rwx

group::r-x

mask::rwx

other::r-x

[root@nfsgw hadoop]# ./sbin/hadoop-daemon.sh --script ./bin/hdfs start portmap //使用root启动portmap

[root@nfsgw hadoop]# sudo -u nfsuser ./sbin/hadoop-daemon.sh --script \

> ./bin/hdfs start nfs3 //使用nfsuser用户启动nfs3

[root@nfsgw hadoop]# jps //查看状态

23076 Portmap

23162 Nfs3

23213 Jps

[root@nfsgw hadoop]# ls logs/ //查看启动日志(有hadoop-nfsuser-nfs3就成功)

hadoop-nfsuser-nfs3-nfsgw.log hadoop-root-portmap-nfsgw.log SecurityAuth-nfsuser.audit

hadoop-nfsuser-nfs3-nfsgw.out hadoop-root-portmap-nfsgw.out SecurityAuth-root.audit

客户端挂载:

[root@client ~]# yum -y install nfs-utils //安装nfs软件

[root@client ~]# mount -t nfs -o vers=3,proto=tcp,noacl,nolock,noatime,sync 192.168.1.65:/ /mnt/

// |版本三|协议tcp|禁止acl扩展权限|不支持NLM|禁用access time时间更新

[root@client ~]# ls /mnt/

111 abc bcd ooxx system tmp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COURSE\*DAY06\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : 文件配置

1.1 编辑分布式计算框架文件

[root@nn01 hadoop]# cd /usr/local/hadoop/etc/hadoop/

[root@nn01 hadoop]# mv mapred-site.xml.template mapred-site.xml

[root@nn01 hadoop]# vim mapred-site.xml //编辑分布式计算框架

<configuration>

<property>

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

<value>yarn</value> //只支持单机local和集群yarn

</property>

</configuration>

1.2 编辑资源管理文件

[root@nn01 hadoop]# vim yarn-site.xml //编辑资源管理文件

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name> //使用哪个计算框架

<value>mapreduce\_shuffle</value> //计算框架名称(要和开发人员沟通)

</property>

<!-- Site specific YARN configuration properties -->

<property>

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

<value>nn01</value>

</property>

</configuration>

1.3 同步配置文件

[root@nn01 hadoop]# for i in node{1..3}; do rsync -aXSH --delete /usr/local/hadoop ${i}:/usr/local/ & done

1.4 启动集群,验证状态

[root@nn01 hadoop]# /usr/local/hadoop/sbin/start-yarn.sh //启动集群

[root@nn01 hadoop]# jps //nn01验证

5203 SecondaryNameNode

5014 NameNode

5846 Jps

5582 ResourceManager

[root@node1 hadoop]# jps //node1验证

1809 Jps

1423 DataNode

1711 NodeManager

[root@node1 hadoop]# /usr/local/hadoop/bin/yarn node -list //查看是否组成集群

19/08/16 09:50:56 INFO client.RMProxy: Connecting to ResourceManager at nn01/192.168.1.60:8032

Total Nodes:3

Node-Id Node-State Node-Http-Address Number-of-Running-Containers

node2:35134 RUNNING node2:8042 0

node1:42069 RUNNING node1:8042 0

node3:32969 RUNNING node3:8042 0

1.5 web访问hadoop

http://192.168.1.60:50070/ //--namenode web页面（nn01）

http://192.168.1.60:50090/ //--secondory namenode web页面（nn01）

http://192.168.1.61:50075/ //--datanode web页面（node1,node2,node3）

http://192.168.1.60:8088/ //--resourcemanager web页面（nn01）

http://192.168.1.61:8042/ //--nodemanager web页面（node1,node2,node3）

二 : 集群使用

哪台都可以,只要有hadoop就行

[root@node1 hadoop]# ./bin/hadoop fs -mkdir /abc /集群中创建文件夹

[root@node1 hadoop]# ./bin/hadoop fs -ls / //查看

Found 1 items

drwxr-xr-x - root supergroup 0 2019-08-16 10:32 /abc

[root@node1 hadoop]# ./bin/hadoop fs -touchz /ooxx //创建文件(多z)

[root@node1 hadoop]# ./bin/hadoop fs -get /ooxx ./ //下载文件

[root@node1 hadoop]# ./bin/hadoop fs -put ./\*.txt /abc //上传文件

[root@node1 hadoop]# ./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar \

> wordcount /abc /bcd //分析/abc文件夹将结果放到/bcd文件夹里

[root@node1 hadoop]# ./bin/hadoop fs -ls /bcd //查看/bcd文件夹

[root@node1 hadoop]# ./bin/hadoop fs -cat /bcd/part-r-00000 //查看结果

[root@node1 hadoop]# ./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar \

> wordcount file:///etc/passwd /111 //分析本地文件passwd将结果放到HDFS的/111文件夹下

fs支持的命令:

[-appendToFile <localsrc> ... <dst>]

[-cat [-ignoreCrc] <src> ...]

[-checksum <src> ...]

[-chgrp [-R] GROUP PATH...]

[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]

[-chown [-R] [OWNER][:[GROUP]] PATH...]

[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]

[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

[-count [-q] [-h] <path> ...]

[-cp [-f] [-p | -p[topax]] <src> ... <dst>]

[-createSnapshot <snapshotDir> [<snapshotName>]]

[-deleteSnapshot <snapshotDir> <snapshotName>]

[-df [-h] [<path> ...]]

[-du [-s] [-h] <path> ...]

[-expunge]

[-find <path> ... <expression> ...]

[-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

[-getfacl [-R] <path>]

[-getfattr [-R] {-n name | -d} [-e en] <path>]

[-getmerge [-nl] <src> <localdst>]

[-help [cmd ...]]

[-ls [-d] [-h] [-R] [<path> ...]]

[-mkdir [-p] <path> ...]

[-moveFromLocal <localsrc> ... <dst>]

[-moveToLocal <src> <localdst>]

[-mv <src> ... <dst>]

[-put [-f] [-p] [-l] <localsrc> ... <dst>]

[-renameSnapshot <snapshotDir> <oldName> <newName>]

[-rm [-f] [-r|-R] [-skipTrash] <src> ...]

[-rmdir [--ignore-fail-on-non-empty] <dir> ...]

[-setfacl [-R] [{-b|-k} {-m|-x <acl\_spec>} <path>]|[--set <acl\_spec> <path>]]

[-setfattr {-n name [-v value] | -x name} <path>]

[-setrep [-R] [-w] <rep> <path> ...]

[-stat [format] <path> ...]

[-tail [-f] <file>]

[-test -[defsz] <path>]

[-text [-ignoreCrc] <src> ...]

[-touchz <path> ...]

[-truncate [-w] <length> <path> ...]

[-usage [cmd ...]]

三 : HDFS节点管理

3.1 增加新节点

1 添加新机器,配置环境

[root@node4 ~]# yum -y install java-1.8.0-openjdk-devel

2 编辑hosts文件

[root@nn01 ~]# vim /etc/hosts

192.168.1.64 node4

3 设置免密登录

[root@node4 ~]# ssh-copy

4 修改NameNode的slave文件增加node节点

[root@nn01 hadoop]# vim etc/hadoop/slaves

node4

5 拷贝文件

[root@nn01 hadoop]# for i in node{1..4}; do rsync -aXSH --delete /usr/local/hadoop ${i}:/usr/local/ & done

6 启动新节点,查看状态

[root@node4 sbin]# ./hadoop-daemon.sh start datanode //在新添加节点运行,启动DataNode命令

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -report //查看集群状态

Name: 192.168.1.61:50010 (node1)

Name: 192.168.1.62:50010 (node2)

Name: 192.168.1.63:50010 (node3)

Name: 192.168.1.64:50010 (node4)

7 设置同步带宽,同步数据

[root@nn01 hadoop]# ./bin/hdfs dfsadmin //直接回车看可使用命令

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -setBalancerBandwidth 50000000 //设置同步带宽,单位bytes

[root@nn01 hadoop]# ./sbin/start-balancer.sh //同步数据

3.2 删除节点

1 编辑配置文件

[root@nn01 hadoop]# vim etc/hadoop/hdfs-site.xml //编辑NameNode的hdfs-site.xml文件

<property>

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

<value>/usr/local/hadoop/etc/hadoop/exclude</value> //指定要删除的节点名称文件

</property>

[root@nn01 hadoop]# vim etc/hadoop/exclude //添加要删除的节点名

[root@nn01 hadoop]# vim etc/hadoop/slaves //删除node4

2 刷新数据

[root@nn01 hadoop]# ./bin/hdfs dfsadmin //查看命令

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -refreshNodes //更新数据刷新节点

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -report //查看节点状态

Decommission Status : Decommission in progress //数据正在迁移

Decommission Status : Decommissioned //数据迁移成功(出现这个才能对要删除的节点下线)

四 : YARN节点管理

[root@node4 hadoop]# ./sbin/yarn-daemon.sh start nodemanager //启动节点(添加节点)

[root@node4 hadoop]# ./sbin/yarn-daemon.sh stop nodemanager //删除节点

stopping nodemanager

[root@node4 hadoop]# ./bin/yarn node -list // 查看节点状态，还是有node4节点，要过一段时间才会消失

19/08/16 14:46:51 INFO client.RMProxy: Connecting to ResourceManager at nn01/192.168.1.60:8032

Total Nodes:5

Node-Id Node-State Node-Http-Address Number-of-Running-Containers

node2:35134 RUNNING node2:8042 0

node1:42069 RUNNING node1:8042 0

node3:32969 RUNNING node3:8042 0

node4:40707 RUNNING node4:8042 0

五 : NFS网关

5.1 添加用户

[root@nn01 hadoop]# groupadd -g 800 nfsuser

[root@nn01 hadoop]# useradd -u 800 -g 800 -r -d /var/hadoop nfsuser

[root@nn01 hadoop]# id nfsuser

uid=800(nfsuser) gid=800(nfsuser) 组=800(nfsuser)

[root@nfsgw ~]# groupadd -g 800 nfsuser

[root@nfsgw ~]# mkdir /usr/hadoop

[root@nfsgw ~]# useradd -u 800 -g 800 -r -d /var/hadoop nfsuser

[root@nfsgw ~]# id nfsuser

uid=800(nfsuser) gid=800(nfsuser) 组=800(nfsuser)

5.2 停止所有服务

[root@nn01 hadoop]# ./sbin/stop-all.sh

[root@nn01 hadoop]# jps //查看

8394 Jps

5.3 修改配置

[root@nn01 hadoop]# vim etc/hadoop/core-site.xml //修改配置

<configuration>

......

<property>

<name>hadoop.proxyuser.nfsuser.groups</name> //挂载点用户所使用的组

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.nfsuser.hosts</name> //挂载点主机地址

<value>\*</value>

</property>

</configuration>

5.4 同步配置到所有主机

[root@nn01 hadoop]# for i in node{1..3}; do rsync -aXSH --delete /usr/local/hadoop/etc \

> ${i}:/usr/local/hadoop & done //只同步配置文件的命令

5.5 启动hdfs

[root@nn01 hadoop]# ./sbin/start-dfs.sh //启动集群

[root@nn01 hadoop]# jps //查看集群

9383 NameNode

9578 SecondaryNameNode

9803 Jps

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -report //查看器群节点

Live datanodes (3):

Name: 192.168.1.61:50010 (node1)

Name: 192.168.1.62:50010 (node2)

Name: 192.168.1.63:50010 (node3)

六 : NFSGW配置

6.1 环境配置

卸载rpcbind nfs-utils

rpm -qa | grep rpcbind

rpm -qa | grep nfs

配置hosts文件

[root@nfsgw ~]# vim /etc/hosts

192.168.1.60 nn01

192.168.1.61 node1

192.168.1.62 node2

192.168.1.63 node3

192.168.1.65 nfsgw

安装JAVA运行环境

yum -y install java-1.8.0-openjdk-devel

同步NameNode的hadoop文件夹到本机

[root@nfsgw ~]# rsync -av 192.168.1.60:/usr/local/hadoop /usr/local/

6.2 修改配置

配置文件hdfs-site.xml

[root@nfsgw hadoop]# vim etc/hadoop/hdfs-site.xml

<configuration>

......

<configuration>

<property>

<name>nfs.exports.allowed.hosts</name> //共享文档权限(挂载用户)

<value>\* rw</value>

</property>

<property>

<name>nfs.dump.dir</name> //文件转储目录

<value>/var/nfstmp</value> //指定文件路径

</property>

</configuration>

创建文件夹,修改所有权

[root@nfsgw hadoop]# mkdir /var/nfstmp

[root@nfsgw hadoop]# chown nfsuser:nfsuser /var/nfstmp

[root@nfsgw hadoop]# ls -ld /var/nfstmp

drwxr-xr-x 2 nfsuser nfsuser 6 8月 16 16:41 /var/nfstmp

6.3 NFS启动与挂载

NFS启动

[root@nfsgw logs]# rm -rf /usr/local/hadoop/logs/\*

[root@nfsgw hadoop]# setfacl -m user:nfsuser:rwx logs/ //设置nfsuser可读权限

[root@nfsgw hadoop]# getfacl logs/

# file: logs/

# owner: root

# group: root

user::rwx

user:nfsuser:rwx

group::r-x

mask::rwx

other::r-x

[root@nfsgw hadoop]# ./sbin/hadoop-daemon.sh --script ./bin/hdfs start portmap //使用root启动portmap

[root@nfsgw hadoop]# sudo -u nfsuser ./sbin/hadoop-daemon.sh --script \

> ./bin/hdfs start nfs3 //使用nfsuser用户启动nfs3

[root@nfsgw hadoop]# jps //查看状态

23076 Portmap

23162 Nfs3

23213 Jps

[root@nfsgw hadoop]# ls logs/ //查看启动日志(有hadoop-nfsuser-nfs3就成功)

hadoop-nfsuser-nfs3-nfsgw.log hadoop-root-portmap-nfsgw.log SecurityAuth-nfsuser.audit

hadoop-nfsuser-nfs3-nfsgw.out hadoop-root-portmap-nfsgw.out SecurityAuth-root.audit

客户端挂载:

[root@client ~]# yum -y install nfs-utils //安装nfs软件

[root@client ~]# mount -t nfs -o vers=3,proto=tcp,noacl,nolock,noatime,sync 192.168.1.65:/ /mnt/

// |版本三|协议tcp|禁止acl扩展权限|不支持NLM|禁用access time时间更新

[root@client ~]# ls /mnt/

111 abc bcd ooxx system tmp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PROJECT3\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : k8s介绍

\*\*Kubernetes\*\*的名字来自希腊语，意思是“舵手” 或 “领航员”。K8s是将8个字母“ubernete”替换为“8”的缩写

K8s 是容器集群管理系统，是一个开源的平台，可以实现容器集群的自动化部署、自动扩缩容、维护等功能

什么时候用

- 快速部署应用

- 快速扩展应用

- 无缝对接新的应用功能

- 节省资源，优化硬件资源的使用

在那用

- \*\*可移植\*\*: 支持公有云，私有云，混合云，多重云（multi-cloud）

- \*\*可扩展\*\*: 模块化, 插件化, 可挂载, 可组合

- \*\*自动化\*\*: 自动部署，自动重启，自动复制，自动伸缩/扩展

怎么用

- \*\*Build, Ship and Run （创建，发送和运行）\*\*

- \*\*Build once，Run anywhere（搭建一次，到处能用）\*\*

- Docker技术的三大核心概念，分别是：

1. 镜像（Image）

2. 容器（Container）

3. 仓库（Repository）

k8s 架构

一个K8S系统，通常称为一个\*\*K8S集群（Cluster）\*\*

这个集群主要包括两个部分：

- \*\*一个Master节点（主节点）\*\*

- Master节点包括API Server、Scheduler、Controller manager、etcd。

​ API Server是整个系统的对外接口，供客户端和其它组件调用，相当于“营业厅”。

​ Scheduler负责对集群内部的资源进行调度，相当于“调度室”。

​ Controller manager负责管理控制器，相当于“大总管”。

- \*\*一群Node节点（计算节点）\*\*

- Node节点包括\*\*Docker、kubelet、kube-proxy、Fluentd、kube-dns\*\*（可选），还有就是\*\*Pod\*\*。

​ \*\*Pod\*\*是Kubernetes最基本的操作单元。

​ 一个Pod代表着集群中运行的一个进程，它内部封装了一个或多个紧密相关的容器。

​ 除了Pod之外，K8S还有一个\*\*Service\*\*，一个Service可以看作一组提供相同服务的Pod的对外访问接口

​ Docker，不用说了，创建容器的。

​ Kubelet，主要负责监视指派到它所在Node上的Pod，包括创建、修改、监控、删除等。

​ Kube-proxy，主要负责为Pod对象提供代理。

​ Fluentd，主要负责日志收集、存储与查询。

二 : 部署k8s

准备docker环境:

1 设置 ip 和主机名

[root@master ~]# vim /etc/hosts

192.168.1.10 master

192.168.1.11 node1

192.168.1.12 node2

192.168.1.13 node3

192.168.1.100 repo

2 开启路由转发

[root@master ~]# vim /etc/sysctl.conf //开机读取文件

net.ipv4.ip\_forward=1 //开启路由转发

[root@master ~]# for i in master node1 node2 node3 repo; do scp /etc/sysctl.conf root@$i:/etc/; done

3 配置 yum 源

[root@master ~]# vim /etc/yum.repos.d/local.repo

[local\_aaa]

name=CentOS-$releasever - aaaa

baseurl="ftp://192.168.1.254/extras"

enabled=1

gpgcheck=0

[root@master ~]# for i in master node1 node2 node3 repo; do scp /etc/yum.repos.d/local.repo \

> root@$i:/etc/yum.repos.d/; done

4 配置私有仓库 registry

[root@repo ~]# yum install docker-distribution //repo

[root@master ~]# yum -y install docker //master node1 node2 node3

[root@master ~]# vim /etc/sysconfig/docker

13 ADD\_REGISTRY='--add-registry 192.168.1.100:5000' //添加注册表

24 INSECURE\_REGISTRY='--insecure-registry 192.168.1.100:5000' //添加不要密码验证的注册表

[root@master ~]# docker load -i /root/pod-infrastructure.tar //导入

[root@master ~]# docker load -i /root/centos.tar

[root@master ~]# docker load -i /root/kubernetes-dashboard.tar

[root@master ~]# docker images

[root@master ~]# docker run -itd docker.io/centos //启动

[root@master ~]# docker start 8d

[root@master ~]# docker exec -it 8d /bin/bash

[root@8d07ecd7e345 /]# yum -y install net-tools iproute psmisc vim-enhanced //安装常用命令

[root@master ~]# docker commit c8 192.168.1.100:5000/myos:test //保存为新镜像

[root@master ~]# docker tag pod-infrastructure:latest 192.168.1.100:5000/pod-infrastructure:latest //改标签

[root@master ~]# docker tag kubernetes-dashboard-amd64:v1.8.3 192.168.1.100:5000/kubernetes-dashboard-amd64:latest //改标签

[root@master ~]# docker push 192.168.1.100:5000/pod-infrastructure //上传到仓库

[root@master ~]# docker push 192.168.1.100:5000/myos

[root@master ~]# docker push 192.168.1.100:5000/kubernetes-dashboard-amd64:latest

[root@node1 ~]# docker run -itd myos:test /bin/bash //验证

配置kubernetes

1 配置kubernetes -master

配置yum源

[root@master ~]# vim /etc/yum.repos.d/local.repo

[local\_k8s]

name=CentOS-$releasever - k8s

baseurl="ftp://192.168.1.254/share/k8s"

enabled=1

gpgcheck=0

安装软件包

[root@master ~]# yum -y install kubernetes-master kubernetes-client etcd

修改etcd配置文件:

[root@master ~]# vim /etc/etcd/etcd.conf //etcd配置文件

6 ETCD\_LISTEN\_CLIENT\_URLS="http://0.0.0.0:2379"

21 ETCD\_ADVERTISE\_CLIENT\_URLS="http://192.168.1.10:2379"

[root@master ~]# systemctl start etcd

[root@master kubernetes]# vim /etc/kubernetes/config

22 KUBE\_MASTER="--master=http://192.168.1.10:8080"

[root@master kubernetes]# vim /etc/kubernetes/apiserver

8 KUBE\_API\_ADDRESS="--insecure-bind-address=0.0.0.0

17 KUBE\_ETCD\_SERVERS="--etcd-servers=http://192.168.1.10:2379"

23 KUBE\_ADMISSION\_CONTROL="--admission-control=NamespaceLifecycle,

LimitRanger,SecurityContextDeny,Re sourceQuota"

启动服务

[root@master ~]# systemctl start kube-apiserver kube-controller-manager \

>kube-scheduler

[root@master ~]# systemctl status etcd kube-apiserver kube-controller-manager \

>kube-scheduler

[root@master kubernetes]# kubectl get cs //查看服务状态

NAME STATUS MESSAGE ERROR

controller-manager Healthy ok

etcd-0 Healthy {"health":"true"}

scheduler Healthy ok

2 配置kubernetes -node

配置yum源

[root@master ~]# vim /etc/yum.repos.d/local.repo

[local\_k8s]

name=CentOS-$releasever - k8s

baseurl="ftp://192.168.1.254/share/k8s"

enabled=1

gpgcheck=0

装包

[root@node3 ~]# yum -y install kubernetes-node docker

修改配置文件

[root@node1 ~]# vim /etc/kubernetes/config

22 KUBE\_MASTER="--master=http://192.168.1.10:8080"

[root@node1 ~]# vim /etc/kubernetes/kubelet

5 KUBELET\_ADDRESS="--address=0.0.0.0"

11 KUBELET\_HOSTNAME="--hostname-override=node1"

14 KUBELET\_ARGS="--cgroup-driver=systemd --fail-swap-on=false

--kubeconfig=/etc/kubernetes/kubelet.kubeconfig //kubernetes配置文件

--pod-infra-container-image=pod-infrastructure:latest" //名字与共享仓库镜像名字一样

[root@node1 ~]# vim /etc/kubernetes/kubelet.kubeconfig //kubernetes配置文件

apiVersion: v1

kind: Config

clusters:

- cluster:

server: http://192.168.1.10:8080

name: local

contexts:

- context:

cluster: local

name: local

current-context: local

[root@node1 ~]# systemctl start kubelet kube-proxy docker //启动服务

设置网络

[root@master ~]# /etc/etcd/etcd.conf

6 ETCD\_LISTEN\_CLIENT\_URLS="http://0.0.0.0:2379"

[root@master ~]# etcdctl mk /atomic.io/network/config '{"Network": \

> "10.254.0.0/16", "Backend": {"Type": "vxlan"}}'

[root@node1 ~]# yum -y install flannel

[root@node1 ~]# vim /etc/sysconfig/flanneld //修改flanneld配置文件

4 FLANNEL\_ETCD\_ENDPOINTS="http://192.168.1.10:2379" //flannel连接IP

[root@node1 ~]# scp /etc/sysconfig/flanneld root@192.168.1.10:/etc/sysconfig/ //传给其他机器

[root@node1 ~]# systemctl restart flanneld docker //重启服务,一定先重启flanneld(网络组件,获取网关用)

[root@node1 ~]# docker run -it myos:test //测试,ip不一样且能ping通

[root@node2 ~]# docker run -it myos:test

使用kubernetes

[root@master ~]# vim kube-dashboard.yaml //创建docker的yaml

21 image: 192.168.1.100:5000/kubernetes-dashboard-amd64:latest

//共享仓库的镜像名称

26 - --apiserver-host=http://192.168.1.10:8080 //master的ip

[root@master ~]# vim baseos.yaml //管理kubernetes的脚本,自动创建docker

15 image: myos:test //使用哪个镜像创建

[root@master ~]# kubectl create -f kube-dashboard.yaml //使用kube创建docker

[root@master ~]# kubectl -n kube-system get pod //查看创建镜像状态

NAME READY STATUS RESTARTS AGE

kubernetes-dashboard-5d6c45f78d-f54h4 1/1 Running 0 1h

[root@master ~]# kubectl get node //查看node状态

NAME STATUS ROLES AGE VERSION

node1 Ready <none> 3h v1.10.3

[root@master ~]# kubectl get pod -o wide //查看docker的详细信息

[root@master ~]# kubectl exec -it test-os-696948b4b4-7x6rn /bin/bash

//进入查看,能相互ping通

[root@master ~]# kubectl exec -it test-os-696948b4b4-97tb2 /bin/bash

[root@master ~]# kubectl -n kube-system get pod -o wide

http://192.168.1.11:30090 //访问kubernetes集群