1. 目录

[目录 1](#_Toc1314900577)

[ADMIN 1](#_Toc877646987)

[Day 01 开学典礼，云计算介绍，TCP/IP协议及其配置 1](#_Toc1938867591)

[一、 云计算介绍 1](#_Toc168151331)

[二、 服务器架构 1](#_Toc642126715)

[1. 服务器：为其他计算机提供服务的高级主机 1](#_Toc1743669835)

[2. 典型服务模式 1](#_Toc1881531834)

[三、 TCP/IP协议及配置 2](#_Toc1604437381)

[TCP/IP协议：统一准则； 2](#_Toc435038995)

[1. 主机通信三要素： 2](#_Toc1861676748)

[2. IP地址：网络中标识节点的逻辑地址 2](#_Toc1386069976)

[Day 02 Linux系统简介 安装Rhel7系统 Rhel7基本操作 4](#_Toc353708346)

[一、 Linux系统简介 4](#_Toc2043532240)

[1. Unix诞生（Linux时间的顶点）：1970-1-1； 4](#_Toc1236194027)

[2. Linux的诞生： 4](#_Toc1047191)

[二、 安装RHEL7系统 4](#_Toc527623591)

[1. 硬盘的使用： 5](#_Toc2034886240)

[2. 鼠标回到真机：ctrl+alt 5](#_Toc645032912)

[3. Linux管理员用户：root 5](#_Toc2131337654)

[4. 使用虚拟机软件 5](#_Toc526479952)

[三、 Linux目录结构 5](#_Toc2047902566)

[1. 最顶层为根目录(/)：所有数据都在此目录下（Linux系统的起点） 6](#_Toc2094644619)

[2. /dev：设备所在目录; 6](#_Toc260240103)

[四、 命令行基本操作 6](#_Toc1131964293)

[1. 获取命令行界面 6](#_Toc436916325)

[Day 03 命令行基础、目录和文件管理、教学环境介绍 9](#_Toc1965045972)

[一、 命令行基础 10](#_Toc1826234880)

[1. 如何编写命令行 10](#_Toc1313572182)

[2. 访问光盘的内容： 11](#_Toc183361006)

[二、 目录和文件管理 13](#_Toc993651809)

[1. 查看及切换目录 13](#_Toc43735521)

[2. 新建文档 14](#_Toc2122228597)

[3. 复制/删除/移动 15](#_Toc1161803141)

[二、 教学环境： 16](#_Toc685862236)

[1. 每个学员机上有三台虚拟机： 16](#_Toc1718414784)

[2. 利用别名方便每一次还原三台虚拟机 16](#_Toc895851327)

[Day 04 软件包管理 配置网络 文本/文件查找 17](#_Toc5970132)

[一、 远程管理： 18](#_Toc610044427)

[1. Linux远程管理：ssh 对方用户名@服务器IP地址 18](#_Toc1528885945)

[2. RPM软件包的管理 18](#_Toc1708281509)

[3. Yum软件包仓库 20](#_Toc963752773)

[二、 文本查找 22](#_Toc1424934537)

[1. grep过滤操作 22](#_Toc796991888)

[2. 重定向输出 22](#_Toc964799965)

[3. echo 'xxx' 将字符串输出到终端 23](#_Toc1952558128)

[4. 重定向整理 23](#_Toc684394481)

[三、 wget 命令行下载工具 23](#_Toc1609832877)

[四、 网络参数的配置 23](#_Toc1936412134)

[1. 设置永久主机名：修改配置文件 /etc/hostname 23](#_Toc1210874433)

[2. 配置永久ip地址，子网掩码，网关地址 23](#_Toc1510251795)

[3. nmcli连接管理 23](#_Toc1883573105)

[4. 配置永久DNS服务器地址：将网站域名解析为对应的IP地址 24](#_Toc1471114537)

[五、 find查找文档位置 24](#_Toc494732441)

[1. 根据预设的条件递归查找对应的文件 24](#_Toc173005782)

[2. find结果处理 25](#_Toc1075624242)

[Day 05 管理用户组 tar备份与恢复 NTP时间同步 cron计划任务 26](#_Toc1999240663)

[一、 管理用户和组 27](#_Toc241712776)

[1. 用户帐号： 27](#_Toc495655772)

[2. 添加用户 27](#_Toc845408824)

[3. 更改密码 28](#_Toc285448297)

[4. 修改用户属性 29](#_Toc470400721)

[5. 删除用户 29](#_Toc2007211965)

[6. 密码信息的存放 29](#_Toc971310533)

[7. 添加组 29](#_Toc41331858)

[8. 修改组属性 30](#_Toc755579645)

[9. 删除组 30](#_Toc1114126502)

[二、 tar备份与恢复 30](#_Toc47301990)

[1. 归档和压缩 30](#_Toc1365624072)

[2. tar工具的常用选项 31](#_Toc495528799)

[三、 NTP时间同步 32](#_Toc1755583499)

[1. 查看日期和时间 32](#_Toc181893198)

[2. NTP时间同步 32](#_Toc1920463336)

[四、 cron计划任务 33](#_Toc405091739)

[1. cron任务概述 33](#_Toc1146693163)

[2. 如何编写crontab任务记录 34](#_Toc1725537816)

[Day 06 权限和归属 使用LDAP认证 家目录漫游 34](#_Toc1089486220)

[一、 权限和归属 35](#_Toc609042392)

[1. 基本权限 35](#_Toc1514466302)

[2. 查看权限 35](#_Toc152877006)

[3. 附加权限（特殊权限） 36](#_Toc2119294188)

[4. acl访问控制列表 37](#_Toc1250555759)

[二、 使用LDAP认证 38](#_Toc1623991543)

[1. 什么是LDAP？ 38](#_Toc466542981)

[三、 家目录漫游 39](#_Toc1423561542)

[1. 共享服务 39](#_Toc552132137)

[四、 autofs工具配置 39](#_Toc778837746)

[1. 装包 39](#_Toc1275318557)

[2. 修改主配置文件 40](#_Toc793844913)

[3. 修改次级配置 40](#_Toc1274493518)

[4. 重启服务 40](#_Toc2120727381)

[ENGINEER 40](#_Toc1744894240)

[Day 01 分区规划及使用 LVM逻辑卷 40](#_Toc1980455699)

[一、 分区规划及使用 42](#_Toc2050603743)

[1. 硬盘分区管理 42](#_Toc1786226098)

[2. 分区规划 42](#_Toc588551696)

[3. 分区格式化 43](#_Toc1017246597)

[4. 挂载使用 43](#_Toc1833528088)

[5. 综合分区 44](#_Toc1954175768)

[二、 LVM逻辑卷 45](#_Toc1512775396)

[1. 作用： 45](#_Toc1441627939)

[2. 创建逻辑卷： 45](#_Toc2136068966)

[3. 扩展逻辑卷 46](#_Toc1285755084)

[4. 卷组划分空间的单位：PE 47](#_Toc1846719678)

[5. 删除逻辑卷 47](#_Toc1135278481)

[Day 02 Shell脚本基础 使用变量 条件测试及选择 列表式循环 47](#_Toc863809252)

[一、 Shell脚本基础 48](#_Toc788722251)

[1. 脚本： 48](#_Toc1744320874)

[2. 简单脚本技巧 48](#_Toc230791906)

[3. 变量的种类 50](#_Toc941599257)

[二、 条件测试及选择 51](#_Toc1716131414)

[1. 条件测试 51](#_Toc1481347665)

[2. if选择结构 51](#_Toc418107152)

[三、 列表式循环 52](#_Toc35190747)

[1. 适用于反复工作的场景 52](#_Toc757425559)

[2. for循环处理 53](#_Toc970239289)

[Day 03 系统安全保护 配置用户环境 配置高级连接 防火墙策略管理 53](#_Toc814028493)

[一、 系统安全保护 54](#_Toc2032744116)

[1. SELinux概述 54](#_Toc1764084202)

[2. 防火墙策略管理 55](#_Toc2088522012)

[3. RHEL7的防火墙体系 56](#_Toc2005987850)

[二、 端口：数字 编号，用来标识进程或程序 57](#_Toc695893764)

[1. 默认端口号 57](#_Toc1685932604)

[2. 端口转发 57](#_Toc1838959901)

[三、 配置高级连接 58](#_Toc599013859)

[1. 配置高级连接（聚合连接 网卡绑定 链路聚合） 参考 man teamd.conf 全文查找/example 按n跳转匹配项 58](#_Toc1324675054)

[2. 制作网卡绑定 58](#_Toc280027949)

[3. 配置team0的ip地址与激活 59](#_Toc1616260456)

[四、 IPv6地址 60](#_Toc1010719494)

[1. IPv4地址：32个二进制 点分隔4个部分 十进制表示 60](#_Toc86720069)

[2. IPv6地址：128个二进制 冒号分隔8个部分 十六进制表示 60](#_Toc981552204)

[Day 04 配置SMB共享 配置NFS共享 60](#_Toc304863785)

[一、 配置永久主机名：server0.example.com 61](#_Toc75305388)

[二、 配置静态ip地址 61](#_Toc119823640)

[三、 配置DNS服务器地址：172.25.254.254 61](#_Toc4099815)

[四、 搭建yum服务端 61](#_Toc1210583869)

[五、 配置SMB共享 61](#_Toc983632892)

[1. 设置双端防火墙默认区域为trusted 61](#_Toc792822066)

[2. 服务名：smb 61](#_Toc807421095)

[3. 配置SMB共享 跨平台的共享 Windows与Linux 62](#_Toc1214424798)

[4. Samba服务基础 62](#_Toc1734421323)

[5. 服务端：server0.example.com 62](#_Toc376068861)

[6. 客户端：desktop0.example.com 64](#_Toc548288815)

[7. 采用更加方便科学的访问方式(挂载访问) 64](#_Toc5044827)

[8. 实现读写的samba共享 65](#_Toc411259608)

[六、 配置NFS共享 66](#_Toc1305714375)

[1. NFS共享概述 66](#_Toc975284116)

[2. 搭建NFS 66](#_Toc1225288102)

[七、 分区规划 67](#_Toc1190974843)

[1. 查看识别的磁盘 lsblk 67](#_Toc591884670)

[2. 划分分区 67](#_Toc1166326466)

[3. 将第一个主分区格式化为xfs文件系统，开机自动挂载到/mnt/mypart 68](#_Toc1049479045)

[4. 将第二个主分区和第三个主分区组成卷组systemvg 68](#_Toc1287778434)

[5. 划分逻辑卷lvtest，大小为3G，格式化为ext4格式，挂载到/mnt/mylv 68](#_Toc704775422)

[6. 逻辑卷lvtest扩大到6G 68](#_Toc740955298)

[Day 05 iSCSI技术应用 数据库服务基础 管理表数据 69](#_Toc1886792293)

[远程端笔记下载方法： 70](#_Toc2029450476)

[一、 iSCSI共享(磁盘共享)internet scsi 70](#_Toc1020983247)

[1. iSCSI磁盘的工作模式，Internet SCSI，网际SCSI接口 70](#_Toc1355569101)

[2. 环境前提: 71](#_Toc892686322)

[3. 实验步骤： 71](#_Toc1107703317)

[二、 数据库服务基础 73](#_Toc189637657)

[1. DB(database)数据库：存放数据的地方 73](#_Toc1197550107)

[2. MySQL AB ==> Sun MySQL ==> Oralce MySQL 73](#_Toc1183008705)

[3. 在server0操作 73](#_Toc309461297)

[4. 数据库的备份与还原 76](#_Toc1201649922)

[5. 用户的创建 76](#_Toc246108926)

[6. 表记录的操作（表数据的操作）:增删查改 77](#_Toc1293094189)

[7. 禁止空密码root用户访问数据库 77](#_Toc1994471989)

[Day 06 HTTP服务基础 网页内容访问 部署动态网站 78](#_Toc1053530022)

[一、 HTTP服务基础 79](#_Toc360035339)

[1. 基于B/S(Browser/Server)架构的网页服务 79](#_Toc1581409664)

[2. 实验一:搭建基本的WEB服务器 79](#_Toc1429598883)

[3. 实验二:配置网站的DNS域名 80](#_Toc908324155)

[4. 实验三:修改网页文件存放路径 81](#_Toc1586454492)

[二、 虚拟Web主机 81](#_Toc1840858492)

[1. 作用:由同一台服务器提供多个不同的Web站点 81](#_Toc66554882)

[2. 实验四:搭建基于域名的虚拟Web主机 82](#_Toc414254960)

[3. 实验五:配置网页内容访问 84](#_Toc918662946)

[4. 使用自定义Web根目录 85](#_Toc1257529725)

[三、 读部署动态网站 86](#_Toc1006139631)

[1. 虚拟机Server0 86](#_Toc2084989412)

[Day 07 综合串讲 综合练习 88](#_Toc159525123)

[一、 HTTPS网站加密 89](#_Toc146434417)

[1. 加密算法： 89](#_Toc642281186)

[2. 对称加密（加密和解密是一把钥匙）适合单机加密 89](#_Toc900480421)

[3. 非对称密码（加密和解密不是一把钥匙）： 89](#_Toc2033226711)

[4. 虚拟机Server0 89](#_Toc524248014)

[二、 基础邮件服务 91](#_Toc1921463669)

[1. 种类 91](#_Toc1241312164)

[三、 分区工具fdisk,parted 93](#_Toc1416934336)

[四、 交换空间 95](#_Toc881683338)

[1. 创建交换空间 95](#_Toc1430949822)

[SERVICES 96](#_Toc467000795)

[Day 01 扩展的几个应用 发布网络YUM源 vim编辑技巧 源码编译安装 systemctl控制 96](#_Toc2064692043)

[一、 补充应用 97](#_Toc1740411119)

[1. man hier 97](#_Toc1668650717)

[2. 搭建教学环境 97](#_Toc163317321)

[3. 权限的数值表示 99](#_Toc886021661)

[4. 历史命令 100](#_Toc1515639058)

[5. 统计文件的占用空间 100](#_Toc1216847343)

[6. date查看计算机时间 100](#_Toc1246057000)

[7. 软链接和硬链接 101](#_Toc949565075)

[8. 查看帮助的方法 102](#_Toc498962579)

[9. zip归档工具,跨平台的压缩归档工具 102](#_Toc6897507)

[10. vim编辑技巧 103](#_Toc388535919)

[11. 源码包安装软件 104](#_Toc192337423)

[12. systemctl命令 106](#_Toc73452389)

[二、 发布网络Yum源 107](#_Toc802790879)

[1. yum排错 107](#_Toc1111000369)

[2. 传输tool包到虚拟机svr7 108](#_Toc1330982115)

[3. 虚拟机svr7: 108](#_Toc1808930510)

[Day 02 DNS服务基础 特殊解析 DNS子域授权 缓存DNS 109](#_Toc1490507238)

[一、 i权限：加上i权限，所有用户包括root都不能修改或删除文件 110](#_Toc1955364928)

[1. 利用root进入虚拟机 密码123456 110](#_Toc1690787319)

[2. 在真机上配置远程管理的别名，进行远程管理（/root/.bashrc） 110](#_Toc243504011)

[3. 为虚拟机A与虚拟机B 搭建Yum仓库 111](#_Toc1841107991)

[4. 检查防火墙状态和SELinux状态 111](#_Toc67551685)

[二、 DNS服务基础 111](#_Toc17484032)

[1. DNS解析的作用 111](#_Toc934936507)

[2. DNS域名管理 111](#_Toc1484486021)

[3. 域名体系 112](#_Toc899167370)

[4. BIND域名服务 112](#_Toc218402681)

[5. 搭建基本的DNS服务 113](#_Toc1951486816)

[三、 多区域的DNS服务器 115](#_Toc816375765)

[四、 特殊的解析记录 116](#_Toc1958813801)

[1. DNS负载均衡(解析结果的轮询) 116](#_Toc1472653885)

[2. 泛域名解析 116](#_Toc979693087)

[3. 有规律的泛域名解析 116](#_Toc697351814)

[4. 解析记录的别名 117](#_Toc840809296)

[五、 DNS子域授权 117](#_Toc49056782)

[1. 父域与子域 117](#_Toc1943408814)

[2. 虚拟机B： 118](#_Toc1790374371)

[3. 虚拟机Ａ： 118](#_Toc548019361)

[六、 主机映射文件/etc/hosts(为本机提供DNS域名解析) 119](#_Toc1950306322)

[七、 缓存DNS,加速解析效率 119](#_Toc31426642)

[1. 真机挂载光盘 119](#_Toc740356784)

[2. 真机书写配置文件 120](#_Toc2023758711)

[3. 搭建缓存DNS服务器 120](#_Toc834217521)

[八、 管理运行级别（运行模式） 120](#_Toc1851357153)

[Day 03 Split分离解析 RAID磁盘阵列 进程管理 日志管理 121](#_Toc1207257178)

[一、 回顾 122](#_Toc495664384)

[1. 常见的DNS服务器有哪些 122](#_Toc752379638)

[1. 常见的DNS资源解析记录 122](#_Toc550280768)

[二、 Split分离解析(视图解析) 122](#_Toc303545664)

[1. 概述 122](#_Toc295683309)

[2. 环境及需求 123](#_Toc793784780)

[3. 分离解析 123](#_Toc2144653655)

[三、 RAID磁盘阵列 124](#_Toc363234994)

[四、 进程管理 126](#_Toc811268812)

[1. 查看进程树 126](#_Toc932106514)

[2. 进程动态排名 128](#_Toc1847721015)

[3. 进程检索 128](#_Toc1710436183)

[4. 控制进程 129](#_Toc1150509196)

[5. 进程处决 130](#_Toc1651724183)

[五、 日志管理 131](#_Toc379328300)

[1. 日志的功能 131](#_Toc961839349)

[2. 查看文本日志消息 131](#_Toc976894421)

[3. 用户登陆分析 132](#_Toc1359021387)

[4. 日志消息的优先级 132](#_Toc1659191163)

[5. 使用journalctl工具 133](#_Toc1817703717)

[Day 04 系统&服务管理进阶 批量装机环境 配置PXE引导 kickstart自动应答 133](#_Toc1455116329)

[一、 网络批量装机环境 134](#_Toc1460594440)

[1. 部署DHCP服务器 134](#_Toc1956097531)

[2. 网络装机概述 136](#_Toc1257939003)

[二、 搭建TFTP服务 138](#_Toc1492021082)

[1. 安装tftp-server包 138](#_Toc548970668)

[2. 启动tftp服务 138](#_Toc1134214067)

[3. 部署pxelinux.0文件 139](#_Toc178754955)

[4. 部署菜单文件 139](#_Toc252844173)

[5. 部署引导文件(启动内核) 139](#_Toc193987597)

[6. 修改菜单文件内容 140](#_Toc674419339)

[三、 构建httpd服务,利用web共享众多的rpm包 141](#_Toc1005223812)

[1. 安装软件包 141](#_Toc744268366)

[2. 建立共享路径 141](#_Toc977965003)

[3. 测试 141](#_Toc1300907121)

[四、 部署无人值守安装,生成应答文件 141](#_Toc1538053146)

[1. 图形生成应答文件工具system-config-kickstart 141](#_Toc975135010)

[2. 运行工具system-config-kickstart 142](#_Toc1664142116)

[3. 查看应答文件 142](#_Toc201838310)

[4. 利用web共享ks应答文件,传递给客户端 142](#_Toc1907241525)

[五、 修改菜单文件,指定ks应答文件 142](#_Toc1364379483)

[Day 05 rsync同步操作 inotify实时同步 Cobbler装机平台 143](#_Toc1912274493)

[一、 rsync同步操作 144](#_Toc910267073)

[1. 命令用法 144](#_Toc868620019)

[二、 inotify实时同步 146](#_Toc144119146)

[1. 实时远程同步 146](#_Toc1872106422)

[1. 目录内容监控 147](#_Toc1845514440)

[三、 Cobbler装机平台 148](#_Toc1503140533)

[1. 环境准备 148](#_Toc1383813937)

[2. 基本用法 149](#_Toc1515734509)

[3. 书写Shell脚本 149](#_Toc763735055)

[4. 搭建Cobbler装机平台 150](#_Toc691446618)

[5. Cobbler简介 150](#_Toc828845301)

[Day 06 日志管理 systemctl服务管理 PATH变量应用 154](#_Toc1949385622)

[一、 日志管理 155](#_Toc173382735)

[1. 日志的功能 155](#_Toc1121319607)

[2. 查看文本日志消息 155](#_Toc936116041)

[3. 用户登陆分析 156](#_Toc352137690)

[4. 日志消息的优先级 156](#_Toc1374163780)

[5. 使用journalctl工具 157](#_Toc1130103638)

[二、 systemctl服务管理 157](#_Toc1026557030)

[三、 Cobbler装机平台 160](#_Toc231903944)

[1. 环境准备 160](#_Toc1874372004)

[2. 基本用法 161](#_Toc2004522033)

[3. 书写Shell脚本 161](#_Toc1532811066)

[4. 搭建Cobbler装机平台 162](#_Toc1264941502)

[四、 PATH变量应用:提供命令搜寻路径 162](#_Toc832173396)

[NETWORK 162](#_Toc1049469534)

[Day 01 计算机网络概述 网络设备及拓扑 OSI参考模型 交换机命令基础 162](#_Toc1466779813)

[一、 什么是计算机网络 163](#_Toc591931273)

[1. 什么是计算机网络 163](#_Toc266365369)

[1. 计算机网络的功能 163](#_Toc1231570658)

[2. 计算机网络发展阶段 163](#_Toc1502198346)

[3. 网络标准 164](#_Toc1134985388)

[4. WAN与LAN 164](#_Toc1375689804)

[二、 网络设备及拓扑 164](#_Toc1226821120)

[1. 网络设备生产厂商 164](#_Toc833016180)

[2. 网络拓扑结构 165](#_Toc731346690)

[三、 OSI参考模型 166](#_Toc463151409)

[1. 网络分层 166](#_Toc201267041)

[2. OSI的七层框架 166](#_Toc1495081745)

[3. TCP/IP五层模型 166](#_Toc1154598027)

[4. 协议 167](#_Toc1030112342)

[5. 协议数据单元 167](#_Toc2067430684)

[四、 物理层 167](#_Toc956500001)

[1. 以太网接口 167](#_Toc1203495077)

[2. 网络接口卡(网卡) 169](#_Toc1041266643)

[五、 Packet Tracer软件 169](#_Toc1892616042)

[六、 交换机命令行模式 169](#_Toc1555632768)

[1. 用户模式 169](#_Toc267946776)

[2. 特权模式(一般用于查看配置信息) 170](#_Toc875236033)

[3. 全局配置模式(所做的配置对整个设备生效) 170](#_Toc434706150)

[4. 接口模式 170](#_Toc499850720)

[5. 配置主机名 170](#_Toc602124389)

[6. 配置enable明文口令 170](#_Toc291744535)

[7. 保存交换机的配置 171](#_Toc2032661786)

[8. 恢复设备出厂默认值 171](#_Toc1867065892)

[9. 重启：reload 171](#_Toc1123917931)

[七、 设备配置的准备工作 171](#_Toc934647672)

[1. 空闲一段时间后，重回初始界面的问题 171](#_Toc1186362057)

[2. 禁用DNS查询 171](#_Toc1715849204)

[3. 配置输出日志同步 171](#_Toc1201013042)

[Day 02 数据链路层 172](#_Toc1070563902)

[一、 以太网 172](#_Toc188514782)

[1. 以太网MAC地址(48位,用16进制表示) 172](#_Toc1646138872)

[2. 单播 1对1 173](#_Toc149901374)

[3. 帧格式 173](#_Toc1021530963)

[二、 以太交换机 173](#_Toc230001914)

[1. 什么是交换机 173](#_Toc613052783)

[2. 交换机的转发原理 174](#_Toc1222798004)

[3. 查看MAC地址表 174](#_Toc1725083659)

[三、 广播域 174](#_Toc1767650811)

[1. VLAN 174](#_Toc105426699)

[四、 Trunk 175](#_Toc1645030696)

[1. 用于多广播域多交换机之间作为中继链路 175](#_Toc576667164)

[2. 配置以太通道 176](#_Toc1308921776)

[Day 03 网络层 177](#_Toc321799559)

[一、 网络层的功能 177](#_Toc717070896)

[二、 路由概述 178](#_Toc806760467)

[三、 路由器工作: 178](#_Toc1197035592)

[四、 如何获得路由表 179](#_Toc1151777046)

[五、 静态路由 179](#_Toc1306611188)

[六、 缺省路由 180](#_Toc1799159981)

[七、 三层交换机 180](#_Toc1443521582)

[八、 动态路由 181](#_Toc1191789326)

[Day 04 传输层 182](#_Toc419955865)

[一、 传输层作用 182](#_Toc2126436999)

[二、 传输层的协议 183](#_Toc557620634)

[三、 TCP协议 183](#_Toc2135805070)

[1. TCP的封装格式 183](#_Toc1179966393)

[2. TCP连接过程:三次握手 183](#_Toc828069702)

[3. TCP的四次断开 184](#_Toc1058885324)

[4. TCP的应用 184](#_Toc1368481175)

[四、 UDP协议 184](#_Toc326724926)

[1. UDP的流控和差错控制 184](#_Toc1208786699)

[2. UDP的应用 185](#_Toc242528490)

[五、 访问控制列表概述 185](#_Toc556726840)

[1. 访问控制列表(ACL) 185](#_Toc1821839482)

[2. 访问控制列表的类型 185](#_Toc1465326495)

[3. 标准访问控制列表的配置 186](#_Toc134326851)

[4. 扩展访问控制列表的配置 187](#_Toc1442006645)

[六、 NAT概述 187](#_Toc1570753194)

[1. NAT作用 187](#_Toc1779357547)

[2. NAT实现方式 187](#_Toc2018673810)

[3. 静态转换 188](#_Toc732191322)

[4. NAT端口映射 189](#_Toc170687591)

[5. 端口多路复用(PAT) 189](#_Toc192989721)

[Day 05 应用层 189](#_Toc1449262219)

[一、 STP生成树算法 189](#_Toc977448058)

[1. 广播风暴的产生 190](#_Toc1390025313)

[2. STP概述 190](#_Toc453555617)

[3. 选择根网桥 190](#_Toc136575598)

[二、 STP配置 191](#_Toc1041701646)

[1. PVST+的配置命令 191](#_Toc1897077199)

[2. HSRP组成员 191](#_Toc1328364925)

[3. HSRP原理 192](#_Toc412960224)

[4. HSRP配置 192](#_Toc169549417)

[Day06综合项目完整实验步骤 193](#_Toc1307318276)

[一、 二层交换机 193](#_Toc970580858)

[二、 三层交换机 196](#_Toc157870839)

[三、 配置生成树协议，产生负载均衡效果。 198](#_Toc339801021)

[SHELL 203](#_Toc1798650560)

[Day 01 Shell概述 编写及执行脚本 Shell变量 总结和答疑 203](#_Toc1216756163)

[一、 Shell环境及特性 204](#_Toc1708282196)

[1. 什么是shell 204](#_Toc2125375486)

[2. Shell的使用方式 一阶段回顾 205](#_Toc278059214)

[3. Shell脚本格式 208](#_Toc1950810687)

[4. Shell脚本运行方式 208](#_Toc534618678)

[5. 变量 209](#_Toc2099898697)

[Day 02 Shell中的数值运算 条件测试操作 使用if选择结构 210](#_Toc668945530)

[一、 Shell中的数值运算 211](#_Toc1394421694)

[1. 整数运算工具 211](#_Toc691923080)

[2. 条件测试 213](#_Toc300819429)

[Day 03 循环结构 case语句 、 函数及中断控制 214](#_Toc1424114402)

[一、 for循环 215](#_Toc471507020)

[1. 不能代入变量的格式 215](#_Toc1458601577)

[2. 可以代入变量的格式 215](#_Toc725892973)

[3. c函数格式 215](#_Toc1448955079)

[二、 while循环 216](#_Toc701143242)

[1. 死循环一般格式 216](#_Toc1179448591)

[三、 case分支编写脚本 216](#_Toc1585530677)

[四、 函数 216](#_Toc1742844889)

[1. 格式 216](#_Toc929042142)

[2. 函数调用 217](#_Toc766411954)

[3. 编写mycolor.sh脚本 217](#_Toc8321465)

[4. Shell版本的fork炸弹(一键死机) 218](#_Toc1098591559)

[5. 中断与退出 218](#_Toc2073730230)

[Day 04 字符串处理 扩展的脚本技巧 正则表达式 218](#_Toc1256462398)

[一、 总结: 219](#_Toc266047603)

[1. 变量相关 219](#_Toc630069236)

[2. 正则 219](#_Toc325734914)

[二、 字符串截取及切割 220](#_Toc1974329800)

[1. 字符串截取的三种方法 220](#_Toc607961074)

[2. 子串替换的两种方法 221](#_Toc603794128)

[3. 字符串掐头去尾 222](#_Toc1777656839)

[4. 字符串初值的处理 223](#_Toc1142579753)

[三、 expect工具 224](#_Toc556209177)

[1. 安装expect工具 224](#_Toc898826725)

[2. 写脚本 224](#_Toc1811525283)

[四、 正则表达式 224](#_Toc1950630872)

[1. 基本正则表 224](#_Toc1590749805)

[2. 扩展正则 225](#_Toc2112344712)

[3. 应用案例 226](#_Toc1068759080)

[Day 05 sed基本用法 sed文本块处理 、 sed高级应用 226](#_Toc867380559)

[一、 sed工具 227](#_Toc436368085)

[1. sed工具选项 227](#_Toc379877010)

[二、 使用awk提取文本 229](#_Toc1593273533)

[1. awk工具概述 229](#_Toc1885323164)

[2. 主要用法 229](#_Toc1081020252)

[三、 格式化输出 231](#_Toc625238476)

[1. awk处理的时机 231](#_Toc1323370193)

[2. 使用数值/字符串比较设置条件 232](#_Toc676381493)

[3. 分支结构 233](#_Toc1554280618)

[四、 Sed的一些补充 233](#_Toc2089782148)

[Day 06 awk工具 234](#_Toc505388530)

[一、 使用awk提取文本 234](#_Toc2016028730)

[1. awk工具概述 235](#_Toc1663605282)

[2. 主要用法 235](#_Toc1761850928)

[二、 格式化输出 237](#_Toc134592686)

[1. awk处理的时机 237](#_Toc146190870)

[2. 使用数值/字符串比较设置条件 238](#_Toc2087585842)

[3. 分支结构 238](#_Toc2108922486)

[三、 awk中数组的使用 239](#_Toc754151944)

[OPERATION 240](#_Toc543896323)

[Day 01 Nginx安装与升级 Nginx服务器 Nginx虚拟主机 、 HTTPS加密网站 240](#_Toc1739095677)

[一、 Nginx安装与升级 241](#_Toc1896731697)

[1. Web服务器对比 241](#_Toc1100105500)

[2. nginx升级 242](#_Toc490438754)

[二、 nginx虚拟主机与网站加密 242](#_Toc1560773332)

[1. 配置文件 242](#_Toc903252724)

[2. 虚拟主机配置 243](#_Toc2081188559)

[3. 网站加密 243](#_Toc1525634397)

[4. vim批量修改 244](#_Toc1972011805)

[5. SSL虚拟主机 244](#_Toc801085470)

[Day 02 部署LNMP Nginx+FastCGI 、 Nginx高级技术 245](#_Toc204405167)

[一、 部署LNMP(Linux,Nginx,Mariadb,PHP) 246](#_Toc246875355)

[1. 基础知识 246](#_Toc1699841998)

[2. 环境准备 247](#_Toc1285425419)

[3. 构建LNMP平台 248](#_Toc872113831)

[4. 动静分离 251](#_Toc875728543)

[二、 地址重写 251](#_Toc1961806913)

[1. 问题:通过调整Nginx服务端配置,实现以下目标: 251](#_Toc278910802)

[2. 方案 252](#_Toc818027043)

[3. 步骤 252](#_Toc499026223)

[Day 03 Nginx调度器 Nginx常见问题 256](#_Toc686572126)

[一、 Nginx反向代理 258](#_Toc15147857)

[1. Nginx调度算法 258](#_Toc398666612)

[二、 Nginx常见问题处理 260](#_Toc821164812)

[1. 压力测试与服务器优化 260](#_Toc161338727)

[2. 浏览器本地静态缓存数据 261](#_Toc338768807)

[3. 查看服务器状态信息 261](#_Toc782603650)

[4. 对页面进行压缩处理 262](#_Toc915490672)

[5. 服务器内存缓存 263](#_Toc882665130)

[Day 04 memcached原理 部署memcached 、 Session共享 263](#_Toc664738721)

[一、 构建memcached服务 264](#_Toc1982770630)

[1. 传统Web架构的问题 264](#_Toc864654433)

[2. rhel7中yum源安装的服务配置文件目录 264](#_Toc78028406)

[3. memcached数据库操作 265](#_Toc738539707)

[4. PHP实现session共享 265](#_Toc798359344)

[Day 05 Tomcat服务器 Tomcat应用案例 、 Varnish代理服务器 266](#_Toc563067864)

[一、 安装部署Tomcat服务器 267](#_Toc1599444814)

[1. 部署环境 267](#_Toc1418181637)

[2. 验证 267](#_Toc767473031)

[二、 使用Tomcat部署虚拟主机 267](#_Toc1846320170)

[1. 修改server.xml配置文件，创建两个域名的虚拟主机，修改如下两个参数块： 268](#_Toc970539987)

[2. 修改www.b.com网站的首页目录为base 268](#_Toc2052898450)

[3. 跳转 : 当用户访问http://www.a.com/test打开/var/www/html目录下的页面 268](#_Toc570950353)

[4. 配置Tomcat支持SSL加密网站 269](#_Toc1846268530)

[三、 Varnish 270](#_Toc1867221715)

[1. 代理服务器概念: 270](#_Toc849861155)

[2. 使用Varnish加速Web 270](#_Toc516811926)

[Day 06 版本控制 SVN基础 实战案例 、 RPM打包 272](#_Toc218764291)

[一、 Subversion基本操作(SVN) 273](#_Toc1634160487)

[1. 什么是Subversion 273](#_Toc1203384052)

[2. Repository--仓库(repo) 273](#_Toc233912148)

[3. 安装SVN服务器 274](#_Toc2032827100)

[4. 客户端测试(192.168.2.200) 275](#_Toc2024548864)

[二、 使用Subversion协同工作 277](#_Toc395250876)

[1. 多人协同工作 277](#_Toc224112259)

[2. 使用dump指令备份版本库数据 279](#_Toc659668866)

[三、 制作nginx的RPM包 279](#_Toc1310741548)

[1. 安装rpm-build软件包 279](#_Toc1106777389)

[2. ppmbuild创建RPM软件包 281](#_Toc1033884545)

[四、 git的使用 281](#_Toc1975480269)

[Day 07 VPN服务器 NTP时间同步 PSSH远程工具 281](#_Toc942064371)

[一、 VPN服务器 282](#_Toc1898538978)

[1. VPN概述 282](#_Toc2053508675)

[2. GRE模块VPN 282](#_Toc1680604078)

[3. 创建PPTP VPN 283](#_Toc549414674)

[4. 创建L2TP+IPSec VPN 285](#_Toc1509687830)

[二、 NTP时间同步 288](#_Toc96188294)

[1. NTP协议概述 288](#_Toc1375840)

[2. 部署NTP服务 288](#_Toc780385819)

[3. 配置客户端 289](#_Toc863661325)

[4. pssh远程套件工具 289](#_Toc1847696010)

[CLUSTER 293](#_Toc1750925806)

[Day 01 存储技术与应用 iSCSI技术应用 、 udev配置 NFS网络文件系统 、 Multipath多路径 、 NFS网络文件系统 、 udev配置 293](#_Toc769076128)

[一、 关于硬盘的二三事 294](#_Toc271162716)

[二、 准备两台虚拟机 294](#_Toc1449710689)

[1. 在proxy上执行 294](#_Toc488814195)

[2. 客户端client 295](#_Toc1121023871)

[三、 附加课外实验：多台FTP或者http主机使用共享存储 296](#_Toc1966522615)

[1. 这里以FTP为例，web1和web2主机都安装vsftpd软件，使用统一的后端共享存储设备 296](#_Toc707578486)

[四、 部署multipath多路径环境 296](#_Toc607700711)

[1. 通过multipath,实现以下目标 296](#_Toc1022423019)

[五、 配置并访问NFS共享 298](#_Toc941490635)

[1. 服务器利用NFS机制发布2个共享目录，要求如下： 298](#_Toc493044163)

[2. 配置proxy的nfs文件 299](#_Toc899488235)

[3. 客户端查看 299](#_Toc1336741511)

[4. NFS依赖服务 299](#_Toc717156422)

[六、 编写udev规则 299](#_Toc1559157101)

[1. 编写udev规则，实现以下目标： 299](#_Toc499999411)

[2. udev(2.6) 3.10 300](#_Toc1823933811)

[Day 02 集群及LVS简介 LVS-NAT集群 LVS-DR集群 301](#_Toc327996032)

[一、 集群及LVS简介 302](#_Toc196613328)

[1. 集群简介 302](#_Toc234021060)

[2. LVS概述 303](#_Toc151634965)

[二、 LVS-NAT集群 306](#_Toc746028002)

[1. 软件安装 306](#_Toc1743708890)

[2. LVS-NAT案例 308](#_Toc247823259)

[三、 部署LVS-DR集群 309](#_Toc747403842)

[1. 问题描述 309](#_Toc376611062)

[2. 方案 310](#_Toc1111484585)

[Day 03 Keepalived热备 Keepalived+LVS 、 HAProxy服务器 313](#_Toc1880560713)

[一、 nginx与LVS调度器区别 314](#_Toc718778921)

[1. nginx代理(我帮你去访问) 314](#_Toc1429763909)

[2. LVS调度(转发数据包) LVS-NAT LVS-DR LVS-TUN 314](#_Toc221891260)

[二、 Keepalived热备 314](#_Toc1839802792)

[1. Keepalived概述 314](#_Toc1248802876)

[2. Keepalived服务 315](#_Toc929469747)

[\* 当web1的keepalived服务停止时,192.168.4.80会自动切换到web2上 317](#_Toc300019855)

[三、 Keepalived+LVS服务器 317](#_Toc123742247)

[1. 问题 317](#_Toc1870960382)

[2. 操作 317](#_Toc793064018)

[3. 部署Keepalived实现LVS-DR模式调度器的高可用 318](#_Toc1023230482)

[四、 配置HAProxy负载均衡集群 319](#_Toc1060218245)

[1. 集群调度软件对比 319](#_Toc1510220440)

[2. 准备4台Linux服务器,两台做web服务器,1台安装HAproxy,1台做客户端,实现以下功能 321](#_Toc434903935)

[3. 清除前面的实验环境 322](#_Toc1560217656)

[4. 配置后端服务器 322](#_Toc1186670603)

[5. 两种方式定义集群: 324](#_Toc880461933)

[Day 04 Ceph概述 部署Ceph集群 Ceph块存储 324](#_Toc1888213688)

[一、 Ceph概述 325](#_Toc1805185138)

[1. 基础知识 325](#_Toc1077075261)

[2. 实验环境准备 328](#_Toc2122234748)

[二、 部署Ceph集群 330](#_Toc1956820103)

[1. 准备部署环境 330](#_Toc1823103263)

[2. 部署存储集群 330](#_Toc1718459991)

[三、 Ceph块存储 333](#_Toc57159714)

[1. 概述 333](#_Toc423023458)

[2. 块存储集群 334](#_Toc2095071053)

[Day 05 块存储应用案例 分布式文件系统 对象存储 338](#_Toc870639663)

[一、 块存储应用案例 339](#_Toc2075124273)

[1. 准备实验环境 339](#_Toc901721364)

[2. 创建KVM虚拟机 340](#_Toc1589418584)

[二、 分布式文件系统 342](#_Toc1357404535)

[1. 基础知识 342](#_Toc1123612625)

[2. 元数据服务器 344](#_Toc1281737728)

[3. 文件系统服务器 345](#_Toc458723763)

[三、 对象存储 346](#_Toc2053082372)

[1. 概述 346](#_Toc1581757584)

[2. 对象存储 348](#_Toc582466011)

[四、 总结和答疑 350](#_Toc1776559106)

[DBA1 350](#_Toc1605696493)

[Day 01 数据库服务概述 构建MySQL服务器 、 数据库基本管理 MySQL数据类型 350](#_Toc689293703)

[一、 数据库服务概述 351](#_Toc1737558395)

[1. 什么是数据库 351](#_Toc2040600429)

[2. MySQL数据库介绍 352](#_Toc102027711)

[二、 构建MySQL服务器 353](#_Toc776745350)

[1. 准备工作(非必须操作)： 353](#_Toc773578714)

[2. 安装部署MySQL 353](#_Toc1990241399)

[3. 启动服务 353](#_Toc434446840)

[4. 配置MySQL管理员密码(默认数据库管理员账户为root) 353](#_Toc1850653976)

[三、 数据库基本管理 354](#_Toc1964992500)

[1. 数据库基本操作流程 354](#_Toc243783295)

[2. 数据库相关指令练习 355](#_Toc1526273591)

[3. 数据表相关指令练习 355](#_Toc1535968843)

[四、 MySQL数据类型 356](#_Toc300943010)

[1. 数字类型 356](#_Toc1949297049)

[2. 字符类型 358](#_Toc1483556248)

[3. 日期时间类型 359](#_Toc1469587309)

[4. 枚举类型(选择类型) 360](#_Toc672453064)

[五、 总结和答疑 361](#_Toc1411196873)

[Day 02 约束条件 修改表结构 MySQL键值 361](#_Toc114388000)

[一、 约束条件 362](#_Toc621117760)

[1. 约束条件 362](#_Toc1347437651)

[二、 修改表结构 362](#_Toc1396125729)

[1. 语法结构 362](#_Toc1079841524)

[2. 添加新字段 363](#_Toc1253036375)

[3. 修改字段类型 363](#_Toc830399665)

[4. 修改字段名 364](#_Toc1662307535)

[5. 删除字段 364](#_Toc882111833)

[6. 修改表名 364](#_Toc1057737619)

[三、 MySQL键值 365](#_Toc1120520380)

[1. MySQL索引概述 365](#_Toc1571405536)

[2. MySQL键值类型 366](#_Toc647812366)

[Day 03 MySQL存储引擎 、 数据导入导出 管理表记录 匹配条件 371](#_Toc1673433247)

[一、 MySQL存储引擎 372](#_Toc1424557717)

[1. MySQL工作原理 372](#_Toc1787215876)

[2. 存储引擎的配置 373](#_Toc1516190998)

[3. 存储引擎特点 373](#_Toc1859004557)

[二、 数据导入导出 376](#_Toc1490386204)

[1. 设置搜索路径 376](#_Toc1333699850)

[2. 数据导入 376](#_Toc2102787853)

[3. 数据导出 378](#_Toc869176147)

[三、 管理表记录 378](#_Toc722185045)

[1. 增加表记录 378](#_Toc256247215)

[2. 查询表记录 379](#_Toc670989549)

[3. 更新表记录 379](#_Toc58257645)

[4. 删除表记录 379](#_Toc1725834524)

[四、 匹配条件 379](#_Toc1343442613)

[1. 基本匹配条件 379](#_Toc1469454519)

[2. 高级匹配条件 381](#_Toc1949659550)

[3. 操作查询结果 382](#_Toc1457830614)

[五、 总结和答疑 384](#_Toc2090572279)

[Day 04 多表查询 MySQL管理工具 、 用户授权及撤销 384](#_Toc706472695)

[一、 多表查询 385](#_Toc1022930155)

[1. 复制表 385](#_Toc255166280)

[2. 多表查询 385](#_Toc1536872360)

[二、 MySQL管理工具 387](#_Toc537754042)

[1. MySQL管理方式 387](#_Toc1137278113)

[2. MySQL管理工具 387](#_Toc447126331)

[三、 用户授权及撤销 388](#_Toc1658274423)

[1. 密码恢复及设置 388](#_Toc561200001)

[2. 创建授权 389](#_Toc1094938698)

[3. 撤销授权 391](#_Toc524427936)

[四、 总结和答疑 391](#_Toc87149600)

[Day 05 mysqldump 实时增量备份 、 innobackupex 391](#_Toc164160164)

[一、 mysqldump 392](#_Toc1603340599)

[1. 数据库备份概述 392](#_Toc83533676)

[2. 物理备份 392](#_Toc789556801)

[3. 逻辑备份 393](#_Toc38837881)

[二、 实时增量备份/恢复 394](#_Toc376238868)

[1. binlog简介 394](#_Toc1511741847)

[2. 使用binlog日志 395](#_Toc295085096)

[3. 利用binlog恢复数据 397](#_Toc1047228417)

[三、 innocackupex 397](#_Toc1569999492)

[1. MySQL备份工具 397](#_Toc2020919621)

[2. 安装percona 398](#_Toc243187382)

[3. innobk应用案例 399](#_Toc891970363)

[4. 在完全备份文件中恢复单个表 401](#_Toc1823095523)

[四、 总结和答疑 402](#_Toc1701017996)

[1. innobackupex增量备份总结： 402](#_Toc835058995)

[DBA2 403](#_Toc825225429)

[Day 01 MySQL主从同步 、 主从同步模式 403](#_Toc260007043)

[一、 MySQL主从同步 404](#_Toc1857989150)

[1. 主从同步概述 404](#_Toc1080391709)

[2. 构建组从同步 405](#_Toc1796879403)

[3. 常用配置选项 407](#_Toc248259545)

[二、 主从同步模式 408](#_Toc70186175)

[1. 结构类型 408](#_Toc96522087)

[2. 配置主从从结构 410](#_Toc1906533968)

[3. 复制模式 410](#_Toc631386176)

[Day 02 MySQL读写分离 MySQL多实例 、 MySQL性能调优 411](#_Toc283478256)

[一、 MySQL读写分离 412](#_Toc718535777)

[1. 读写分离概述 412](#_Toc1563473552)

[2. 构建读写分离 413](#_Toc447638421)

[二、 MySQL多实例 416](#_Toc174392728)

[1. 多实例概述 416](#_Toc1647007228)

[2. 配置多实例 416](#_Toc2102184789)

[三、 MySQL性能调优 417](#_Toc963949529)

[1. MySQL工作原理 417](#_Toc1685845110)

[2. MySQL性能调优 418](#_Toc330940009)

[3. SQL查询优化 421](#_Toc328207728)

[Day 03 MHA集群概述 、 部署MHA集群 测试配置 422](#_Toc1378168426)

[一、 MHA集群概述 423](#_Toc1898207221)

[1. MHA介绍 423](#_Toc1854366179)

[2. MHA工作过程 423](#_Toc1621355809)

[二、 部署MHA集群 424](#_Toc642693936)

[1. 拓扑结构 424](#_Toc1529978055)

[2. 准备集群环境 425](#_Toc1174890157)

[3. 配置MHA集群 426](#_Toc1477752931)

[三、 测试配置 428](#_Toc207719836)

[1. 测试集群配置 428](#_Toc1434897201)

[2. 测试高可用配置 429](#_Toc1188258434)

[Day 04 MySQL视图 MySQL存储过程 429](#_Toc1084292956)

[一、 MySQL视图 430](#_Toc1436517979)

[1. 视图概述 430](#_Toc1358297720)

[2. 视图基本使用 431](#_Toc1180815043)

[3. 视图进阶 432](#_Toc1195568299)

[二、 MySQL存储过程 432](#_Toc1989683897)

[1. 存储过程概述 432](#_Toc224792180)

[2. 基本使用 433](#_Toc1479046555)

[3. 存储过程进阶 434](#_Toc560736026)

[4. 流程控制 436](#_Toc1788265732)

[Day 05 分库分表概述 配置mycat 439](#_Toc735128754)

[一、 分库分表概述 440](#_Toc1287789313)

[1. 相关概念 440](#_Toc1881386118)

[2. mycat介绍 440](#_Toc1699078283)

[二、 配置mycat 442](#_Toc826150775)

[NOSQL 445](#_Toc64842479)

[Day 01 NoSQL概述、部署Redis服务、部署LNMP+Redis 445](#_Toc2027286012)

[一、 NoSQL概述 447](#_Toc659597333)

[1. 数据库类型 447](#_Toc1443010906)

[二、 部署Redis服务 448](#_Toc1778009585)

[1. 部署Redis服务 448](#_Toc366479865)

[2. 配置文件解析 450](#_Toc916883067)

[一、 部署LNMP+Redis 452](#_Toc273219873)

[1. 部署LNMP 452](#_Toc1896457920)

[2. PHP+Redis支持 453](#_Toc2091773224)

[Day 02 创建Redis集群、管理集群 454](#_Toc1750972805)

[一、 创建Redis集群 454](#_Toc2104177756)

[1. 准备集群环境 455](#_Toc1379186777)

[2. 创建Redis集群 456](#_Toc791747591)

[一、 管理集群 459](#_Toc1244805653)

[1. 管理命令 459](#_Toc315996086)

[2. master选举测试 460](#_Toc80781922)

[3. 添加新节点 460](#_Toc455619726)

[4. 移除节点 461](#_Toc1496811129)

[Day 03: redis主从同步 存储持久化 redis数据类型 462](#_Toc1276350221)

[一、 Redis主从复制 463](#_Toc297819975)

[1. 主从复制概述 463](#_Toc1721603310)

[2. 配置主从复制 463](#_Toc607913128)

[二、 RDB/AOF持久化 465](#_Toc858556001)

[1. 持久化之RDB 465](#_Toc1362385394)

[2. 持久化之AOF 467](#_Toc387114457)

[三、 数据类型 469](#_Toc1593684755)

[1. String字符串 469](#_Toc502691059)

[2. List列表 471](#_Toc121016927)

[3. Hash表 473](#_Toc1145279390)

[Day 04 MongoDB安装、MongoDB命令、MongoDB数据类型、MongoDB导入导出、MongoDB备份还原 474](#_Toc1328841834)

[一、 部署MongoDB服务 475](#_Toc185859406)

[1. MongoDB概述 475](#_Toc1025081754)

[2. 搭建MDB服务器 475](#_Toc1988439168)

[二、 MongoDB基本使用 476](#_Toc1628870312)

[1. 常用管理命令 476](#_Toc655607691)

[2. 基本数据类型 479](#_Toc207435385)

[3. 数据导入导出 481](#_Toc398269731)

[4. 数据备份与恢复 482](#_Toc928827565)

[Day 05 MongoDB副本集、MongoDB文档管理 483](#_Toc2103893305)

[一、 MongoDB副本集 484](#_Toc342559308)

[1. 环境准备 484](#_Toc532316722)

[2. 配置Replica Sets 485](#_Toc2060587413)

[二、 文档管理 486](#_Toc1721746085)

[1. 插入文档 486](#_Toc1324064313)

[2. 查询文档 487](#_Toc1157909418)

[3. 更新文档 489](#_Toc2037742171)

[4. 删除文档 490](#_Toc1404846235)

[SECURITY 491](#_Toc1613529144)

[Day 01 Linux基本防护 用户切换与提权 SSH访问控制 SELinux安全 、 SSH访问控制 SELinux安全 491](#_Toc1387069653)

[Day 04 iptables防火墙 filter表控制 扩展匹配 nat表典型应用 522](#_Toc533712808)

[一、 iptables防火墙 522](#_Toc1911349119)

[1. 防火墙概述 522](#_Toc961189315)

[2. iptables用法解析 524](#_Toc1141625936)

[3. 规则管理示例 524](#_Toc622421472)

[4. iptables命令的基本使用方法 524](#_Toc176091061)

[CLOUD 535](#_Toc68622579)

[Day 01 KVM简介、Virsh管理 、自定义虚拟机、虚拟设备管理 535](#_Toc678782121)

[一、 阶段简介 537](#_Toc1649757320)

[二、 配置yum源 537](#_Toc1213901970)

[1. 安装vsftpd服务 537](#_Toc2007623955)

[2. 自定义yum源 539](#_Toc1835616727)

[3. dns转发+时间同步 539](#_Toc91500076)

[三、 KVM简介 540](#_Toc1848579475)

[1. 搭建KVM服务器 540](#_Toc1317003391)

[2. 管理KVM平台 542](#_Toc747107768)

[二、 virsh管理 545](#_Toc2056014860)

[1. virsh虚拟机管理 546](#_Toc1715273123)

[2. xml管理 547](#_Toc1675935333)

[三、 自定义虚拟机 548](#_Toc2012424517)

[1. 自定义虚拟机安装 548](#_Toc2057832431)

[2. 虚拟机模板制作 549](#_Toc60768407)

[四、 虚拟设备管理 551](#_Toc1925528282)

[1. xml文件 551](#_Toc1632094868)

[2. 创建访问虚拟机 555](#_Toc1384832720)

[Day 02 Openstack概述、部署安装环境、部署Openstack OpenStack操作基础 556](#_Toc935954053)

[一、 Openstack概述 557](#_Toc1522353392)

[1. 云计算简介 557](#_Toc642195307)

[2. Openstack简介 558](#_Toc401999549)

[二、 部署安装环境 561](#_Toc761939397)

[1. Openstack环境准备 561](#_Toc1175908115)

[三、 部署Openstack 562](#_Toc165865021)

[1. 基础环境准备(旧) 562](#_Toc1723128712)

[2. 基础环境准备(新) 563](#_Toc170050403)

[3. 安装Openstack 564](#_Toc788286493)

[4. 网络配置 566](#_Toc1899219773)

[四、 Openstack操作基础 568](#_Toc1698790797)

[1. Horizon配置 568](#_Toc856909073)

[Day 03 用户配额管理、云主机类型管理、镜像管理、网络管理案例和实例管理、安装额外计算节点 570](#_Toc430518246)

[一、 项目配额、用户管理 571](#_Toc1201064469)

[1. 项目管理 571](#_Toc2070811043)

[2. 用户管理 572](#_Toc290658554)

[3. 配额管理 574](#_Toc889197548)

[二、 云主机类型管理 575](#_Toc14827471)

[1. 云主机类型 575](#_Toc2139238029)

[三、 镜像管理 577](#_Toc58717292)

[1. 镜像基础 577](#_Toc761935239)

[2. 镜像应用 578](#_Toc2047769242)

[四、 网络管理 579](#_Toc1773990415)

[1. 网络和路由 579](#_Toc290386924)

[2. 浮动IP地址 582](#_Toc1912710111)

[五、 安全和实例管理 583](#_Toc1684339198)

[1. 安全管理 583](#_Toc351155331)

[2. 实例管理 584](#_Toc1690754746)

[六、 计算节点扩容 585](#_Toc1168950418)

[1. 环境准备 585](#_Toc1735988051)

[2. 安装计算节点 586](#_Toc479225151)

[Day 04 阿里云简介、ECS实战、华为云简介、云计算测评报告 588](#_Toc543820162)

[一、 阿里云产品 589](#_Toc230699710)

[1. 云计算基础 589](#_Toc881224700)

[2. 阿里云产品和技术 590](#_Toc1305759559)

[3. 阿里云应用架构 590](#_Toc1406607825)

[二、 华为云 592](#_Toc1047089721)

[1. 简介 592](#_Toc881404623)

[2. 产品介绍 593](#_Toc1576658229)

[三、 ECS云服务器管理 593](#_Toc1835376215)

[1. 华为云控制台 593](#_Toc633140749)

[2. ECS选购及操作 594](#_Toc1127965378)

[3. ECS选购及操作 595](#_Toc544801640)

[一、 596](#_Toc1063658995)

[Day 06 Docker概述、部署Docker、Docker镜像、Docker基本命令 596](#_Toc181546199)

[一、 Docker概述 597](#_Toc468129035)

[1. 概述 597](#_Toc1354317549)

[2. Docker特性 597](#_Toc1070743748)

[二、 部署Docker 598](#_Toc482956506)

[1. 安装前准备 598](#_Toc1346071931)

[2. 安装Docker 598](#_Toc1129461040)

[三、 Docker镜像 600](#_Toc1244891746)

[1. 基本概念 600](#_Toc1246357525)

[2. 镜像操作 602](#_Toc755967807)

[3. 启动不同镜像中遇到的问题 603](#_Toc1535278670)

[四、 Docker基本命令 604](#_Toc1011583988)

[1. 镜像常用命令 604](#_Toc292823357)

[2. 容器常用命令 605](#_Toc1886434002)

[Day 07 自定义镜像及仓库、持久化存储、Docker网络架构 607](#_Toc1461773775)

[一、 自定义镜像与仓库 608](#_Toc1474938405)

[1. 自定义镜像 608](#_Toc1034080237)

[2. 自定义镜像仓库 609](#_Toc2005593938)

[二、 持久化存储 612](#_Toc1705638116)

[1. 存储卷 612](#_Toc1915304938)

[2. 共享存储 613](#_Toc1163869849)

[三、 Docker网络架构 614](#_Toc964762293)

[1. Linux网桥 614](#_Toc814911011)

[2. Docker网络拓扑 614](#_Toc2045274473)

[ARCHITECTURE 615](#_Toc393936874)

[Day 01 ansible概述、ansible基础 、 ad-hoc、批量配置管理 615](#_Toc502803578)

[一、 absible基础 616](#_Toc530931574)

[1. ansible简介 616](#_Toc1521902252)

[2. ansible安装 618](#_Toc1047605218)

[二、 ad-hoc(执行临时操作) 620](#_Toc1594590569)

[1. 主机管理 620](#_Toc1703448452)

[2. 批量执行 626](#_Toc1515734253)

[三、 批量配置管理 627](#_Toc801424471)

[1. 模块 627](#_Toc626708552)

[Day 02 playbook基础、playbook进阶 632](#_Toc1998690760)

[一、 Playbook基础 633](#_Toc12754)

[1. 基础知识 633](#_Toc1756169592)

[2. playbook 639](#_Toc1096098858)

[二、 Playbook进阶 642](#_Toc1246370279)

[1. 语法进阶 642](#_Toc364653751)

[2. 调试 648](#_Toc483893880)

[Day 03 分布式ELK平台、ES安装、扩展插件 649](#_Toc110470619)

[一、 分布式ELK平台 650](#_Toc657477108)

[1. 重定向 650](#_Toc222844234)

[2. ELK简介 651](#_Toc665325706)

[3. Elasticsearch 652](#_Toc2119250883)

[二、 ES集群安装 654](#_Toc1697782640)

[1. ES基础 654](#_Toc1699405943)

[2. ES插件的安装与使用 659](#_Toc1977361173)

[三、 扩展插件 661](#_Toc1255937108)

[1. RESTful API 661](#_Toc1467227233)

[Day 04 Kibana安装、Kibana使用、Logstash配置扩展插件 665](#_Toc993747375)

[一、 Kibana安装与使用 666](#_Toc73215753)

[1. Kibana 666](#_Toc134654597)

[二、 Logstash配置扩展插件 669](#_Toc891538200)

[1. Logstash 669](#_Toc467152628)

1. ADMIN
2. Day 01 开学典礼，云计算介绍，TCP/IP协议及其配置
   1. 云计算介绍

· 云中的计算机，提供云服务

* 1. 服务器架构
     1. 服务器：为其他计算机提供服务的高级主机

- 机架式

- 塔式

- 机柜式

- 刀片式

* + 1. 典型服务模式

- Client/Server(C/S)架构：由服务器提供资源和功能，客户机使用资源和功能

* 1. TCP/IP协议及配置

TCP/IP协议：统一准则；

1. 主机通信三要素：

- IP地址 ip address

- 子网掩码 subnet mask

- IP路由 ip router

1. IP地址：网络中标识节点的逻辑地址
2. 特点：

· 唯一标识

· 长

· 二进制

1. 组成（点分十进制）

- 32个二进制

- 表示为4个十进制数，用.隔开

1. 分类：

· A类:1-126 网 主 主 主

· B类:128-191 网 网 主 主

· C类:192-223 网 网 网 主

· D类:224-239 组播用

· E类:240-254 科研用

· ip 127.0.0.1代表本机

1. 给企业的私有IP划分有：

· A类：10.0.0.1-10.255.255.254

· B类：172.16.0.1-172.31.255.254

· C类：192.168.0.1-192.168.255.254

1. 子网掩码：标识IP地址的主机位和网络位，1表示网络位，0表示主机位

· A类:255.0.0.0

· B类:255.255.0.0

· C类:255.255.255.0

『192.168.1.1 255.255.255.0』= 192.168.1.1/24 #24个网络位

DHCP协议：自动配置ip地址

1. 网关：

· 从一个网络连接到另一个网络的“关口”（不同的网络之间的通信）

1. DNS域名解析服务器：将域名解析为IP地址
2. 在Window系统的cmd命令行界面中：

· 查看ip的命令：ipconfig 若要查看更多信息，则使用ipconfig /all

· 测试通信的命令：ping

1. Day 02 Linux系统简介 安装Rhel7系统 Rhel7基本操作
2. Linux系统简介
3. Unix诞生（Linux时间的顶点）：1970-1-1；
4. Linux的诞生：
5. Linux之父：Linus Torwalds

· 1991.10发布0.02版内核；

· 1994.3发布1.0版内核；

内核作用：调配所有的硬件

内核版本号：主版本.次版本.修订号

1. 一套公开发布的完整的Linux系统=Linux内核+应用程序
2. Linux系统的应用：

· 基于Linux系统的企业服务器

· 嵌入式系统

· 高性能大型运算

1. 安装RHEL7系统
2. 硬盘的使用：

· 物理硬盘 ---> 分区规划 ---> 格式化 ---> 读/写

· -格式化：赋予空间文件系统，决定数据在空间中如何存储（排列规则）

· Windows文件系统：FAT，NTFS

· Linux文件系统： RHEL6 ext4

RHEL7 xfs

SWAP 交换空间（虚拟内存）

· 格式化的基本作用：

· 定义向磁盘介质上存储文档的方法和数据结构，以及读取文档的规则；

· 不同类型的文件系统，其存储/读取方式不一样；

· 格式化操作就是建立新的文件系统；

1. 鼠标回到真机：ctrl+alt
2. Linux管理员用户：root
3. 使用虚拟机软件

· 通过软件来模拟真实计算机的一类软件程序

· 一台物理机可运行多个虚拟机

· 每个虚拟机提供一个相对独立的操作系统环境

1. Linux目录结构
2. 最顶层为根目录(/)：所有数据都在此目录下（Linux系统的起点）
3. /dev：设备所在目录;
4. hd表示IDE设备;sd表示SCSI设备;vd表示虚拟机硬盘;
5. /dev/sda5

· a:磁盘号ls

· 5:分区号

1. 命令行基本操作
2. 获取命令行界面
3. 虚拟控制台切换(ctrl+alt+Fx)

· tty1-tty2:图形桌面

· tty3-tty6:字符控制台

· 右键“在终端中打开”

· 应用程序 ---> 工具 ---> 终端

1. 伪字符终端

· 提示符：

管理员:[用户名@主机名 当前目录]#

用户:[用户名@主机名 当前目录]$

~:用户的家(Home)目录

/root:管理员的家目录

/home:用户的家目录

1. 命令行的一般格式：

· 命令字 [选项]... [参数1] [参数2]...

· 命令

1. pwd - Print Working Directory：查看当前工作目录
2. cd - Change Directory：切换工作目录

· 格式：cd [目标路径]

· 路径：分为 绝对路径 和 相对路径

· 在相对路径中：

- . ：当前目录

- .. ：上一级目录

- cd .. 返回上一级

1. ls - List：列出当前所在目录内容

· 格式：ls [选项]... [目录或文件名]...

· 支持多个参数

· 选项： -l：显示目录内容的详细信息

1. cat：查看文本文件内容

· 选项： -n 显示同时标注行号

· 格式：cat /root/1.txt

· cat /proc/meminfo：列出内存信息

· 查看系统版本

· cat /etc/redhat-release

1. clear & Ctrl+L：清屏
2. uname -r：列出内核版本

[root@localhost ~]# uname -r

3.10.0-693.el7.x86\_64

1. cp - copy：拷贝

· 选项： -r 拷贝整个目录

1. mv - move：移动（剪切）

· 改文件名也使用mv

1. less：分屏浏览文本

· 查看大文件

· 全文查找 “/”+查找的内容

· 按q退出浏览模式

1. head -(数字n) 文件名 看头n行（默认10）
2. tail -(数字n) 文件名 看尾n行（默认10）
3. man：查看命令帮助信息

· 格式：man 命令

1. lscpu：列出cpu信息
2. cat /proc/meminfo：查看内存信息
3. hostname：临时查看/修改主机名

· 查看： [root@xxd ~]# hostname

xxd

· 修改： [root@xxd ~]# hostname localhost

1. ifconfig：列出已激活的网卡连接信息

· ifconfig eth0 192.168.1.1 临时配置ip

1. ping：测试网络连通性

· Ctrl+C：结束正在运行的命令

1. poweroff：关机
2. reboot：重启
3. mkdir - Make Directory：创建目录
4. touch

· 用途：创建空文件

· 格式：touch 文件名

1. grep：

· 用途：输出包含指定字符串的行

· 格式：grep [选项] '查找内容' 目标文件

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

或者[root@localhost 桌面]# grep 'root' /etc/passwd

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

operator:x:11:0:operator:/root:/sbin/nologin

1. Linux系统中：

文本色：文件

蓝色： 目录

1. /etc/redhat-release：储存本机系统的具体版本信息
2. 技巧：<Tab> 补全键：补全路径和命令

连续两下Tab键：列出可能的所有命令;

1. Day 03 命令行基础、目录和文件管理、教学环境介绍
2. 命令行基础
3. 如何编写命令行

i.什么是命令

- 用来实现某一类功能的指令或程序

- 命令的执行依赖于解释器（例如：/bin/bash）

1.绿色：可以执行的程序

青色：快捷方式

2. 提示“bash:firefox：未找到命令...”的原因

1) 命令有误

2) 该命令没有安装

3. poweroff ==> 解释器 ==> 内核 ==> 硬件

4. 解释器：Shell;

Linux下默认的解释器程序：/bin/bash

ii. Linux命令分类

- 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

iii. 命令行的一般格式

·命令字 [选项]... [参数1][参数2]...

· 选项：调控命令的方式

-短选项：-l、-A、-c、-d...

-多个短选项 --> 复合选项：-lh、-lA...

-长选项：--help...

iv. 快速编辑技巧

·快捷键：

- ctrl + l ：清空整个屏幕

- ctrl + u ：清空至行首

- ctrl + w ：往回删除一个单词

- ctrl + c ：废弃当前编辑的命令行

- Esc + . ：粘贴上一个命令的参数

b)mount挂载操作

· 挂载/装载：

- 将光盘/U盘/分区/网络存储等设备装到某个Linux目录

- 各种命令工具通过访问Linux目录来操作这些设备

1. 访问光盘的内容：

Windows：

光盘 => 光驱设备 => CD驱动器(访问点)

Linux：

光盘 => 光驱设备 => 访问点(手动配置)

挂载点 = 访问点 <==> 目录

1. 图形界面下将光盘放入光驱设备
2. 查看光驱设备

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

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

lrwxrwxrwx. 1 root root 3 2月 1 10:34 /dev/cdrom -> sr0

1. 访问光驱设备，可以提供设备的访问点

· mount命令：挂载

[root@localhost ~]# mkdir /dvd

[root@localhost ~]# ls /dvd

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

/dev/cdrom

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

mount: /dev/sr0 写保护，将以只读方式挂载

[root@localhost ~]# ls /dvd

（光盘文件）

[root@localhost ~]# ls /dvd/Packages/

（包文件爱你哟~）

1. 卸载已挂载的设备或分区

· umount命令：卸载

· 卸载和重装示例：

[root@localhost ~]# umount /dvd/

[root@localhost ~]# ls /dvd/

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

mount: /dev/sr0 写保护，将以只读方式挂载

[root@localhost ~]# ls /mnt/

（光盘文件）

· 在Linux中访问设备资源内容，必须通过目录作为访问点进行访问

1. 注意事项：

· 进行挂载时，挂载点目录必须要存在

· 进行挂载时，挂载点目录不要是根目录以下已经存在的目录（建议）

· 卸载时，确认当前没有任何人在挂载点中，否则出现“目标忙”提示

· 同一设备可以在两个目录上同时挂载

1. 目录和文件管理
   * 1. 查看及切换目录
2. 使用pwd、cd

- cd ：

· 用途：切换工作目录

· 格式：cd [目标文件夹位置]

· /home：存放所有普通用户的家目录

· ~user表示user的家目录

- useradd ：创建新用户

1. 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

- -A ：包括以.开头的隐藏文档

- -d ：显示目录本身的属性

- -h ：提供易读的容量(K、M等)

* + 1. 新建文档

1. 命令的别名：简化复杂命令的执行

· alias 别名 = '真正执行的命令'

· unalias 删除别名

1. 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

- [a-z]：多个字符或连续范围中的一个

- {a,min,xy}：多组不同的字符串，全匹配

· 请利用通配符显示/dev/目录下tty20至tty30?

[root@localhost ~]# ls /dev/tty{2?,30}

1. 创建目录

· mkdir - Make Directory

- 格式：mkdir [-p] [/路径/]目录名...

mkdir -p ：创建多层的目录，当父目录没有的时候创建父目录

ls -R 递归显示：目录本身的内容 所有子目录的内容 依次展开

1. vim文本编辑器

- 格式：vim [[/目录/]文件名]

- 若目标文件不存在，则新建空文件并编辑

- 若目标文件不存在，则打开此文件并编辑

· 命令模式下：

- 按i进入输入模式编辑，按Esc回到命令模式

- 按':'进入末行模式，按Esc回到命令模式

· 输入 :wq 保存并退出

· 输入 :q! 不保存强制退出

* + 1. 复制/删除/移动

1. rm —— Remove

- 格式：rm [选项]... 文件或目录...

· 常用命令选项

-r：递归删除（含目录）

-f：强制删除

· rm -rf 爆破警告

1. mv —— 移动/改名

- 格式： mv [选项]... 原文件... 目标路径

- 功能：

· 移动并改名

· 重命名：路径不变的移动

1. 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

-p：保持源文件的权限、修改时间等属性不变

· 当拷贝出现两个以上参数时，永远将最后一个参数作为目标，其他作为源

· cp支持通配符

· cp与.连用

· cp支持重命名

* 1. 教学环境：
     1. 每个学员机上有三台虚拟机：

- server --- 服务器

- desktop --- 客户机

- classroom --- 提供网关/DNS/软件素材等资源

首先开启classroom，然后再看其server和desktop

还原指令：物理机：vim /user/local/bin/rht-vmctl

然后rht-vmctl reset classroom

rht-vmctl reset server

rht-vmctl reset desktop

* + 1. 利用别名方便每一次还原三台虚拟机

#每次开机root都会读取的文件/root/.bashrc

alias c='rht-vmctl reset classroom'

alias d='rht-vmctl reset desktop'

alias s='rht-vmctl reset server'

开启一个新的终端验证

1. Day 04 软件包管理 配置网络 文本/文件查找
2. 远程管理：
   * 1. Linux远程管理：ssh 对方用户名@服务器IP地址

server虚拟机 ：IP地址 172.25.0.11 主机名：server0.example.com

desktop虚拟机 ：IP地址 172.25.0.10 主机名：desktop0.example.com

[root@room9pc01 ~]# ssh -X root@172.25.0.11

-X（大写）：在远程管理时，可以在本机运行对端的图形程序

ctrl + shift + t ：开启一个新的终端

* + 1. RPM软件包的管理

· RPM Package Manager，RPM包管理器

- rpm -q 软件名... #检测是否有安装程序

- rpm -ivh 软件名-版本信息.rpm... #安装软件包

- rpm -e 软件名... #卸载软件包

[root@room9pc01 bin]# rpm -q vsftpd

1.从光盘内容安装到系统中，首先关闭虚拟机，添加光驱设备

2.查看是否具备光驱设备

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

/dev/cdrom

3.挂载光驱设备

[root@server0 ~]# mkdir /dvd

[root@server0 ~]# mount /dev/cdrom /dvd/

mount: /dev/sr0 写保护，将以只读方式挂载

[root@server0 ~]# ls /dvd/

[root@server0 ~]# ls /dvd/Packages/vsftpd-3.0.2-22.el7.x86\_64.rpm

/dvd/Packages/vsftpd-3.0.2-22.el7.x86\_64.rpm

4.安装包

4.1.一般安装

[root@server0 ~]# rpm -q vsftpd

未安装软件包 vsftpd

[root@server0 ~]# rpm -ivh /dvd/Packages/vsftpd-3.0.2-22.el7.x86\_64.rpm

警告：/dvd/Packages/vsftpd-3.0.2-22.el7.x86\_64.rpm: 头V3 RSA/SHA256 Signature, 密钥 ID fd431d51: NOKEY

准备中... ################################# [100%]

正在升级/安装...

1:vsftpd-3.0.2-22.el7 ################################# [100%]

[root@server0 ~]# rpm -q vsftpd

vsftpd-3.0.2-22.el7.x86\_64

· 警告原因：系统不能识别红帽认证信息

· 导入红帽签名信息：

[root@server0 ~]# rpm --import /dvd/RPM-GPG-KEY-redhat-release

· 导入后重装：

[root@server0 ~]# rpm -ivh /dvd/Packages/vsftpd-3.0.2-22.el7.x86\_64.rpm

准备中... ################################# [100%]

正在升级/安装...

1:vsftpd-3.0.2-22.el7 ################################# [100%]

5.卸载包

[root@server0 ~]# rpm -e vsftpd

[root@server0 ~]# rpm -q vsftpd

未安装软件包 vsftpd

* + 1. Yum软件包仓库

· 自动解决依赖关系

· Yellowdog Update Manager，黄狗升级管理器

- yum repolist 列仓库

- yum list [软件名]... 列软件

- yum clean all 清缓存

- yum [-y] install 软件名... 安装软件

- yum [-y] remove 软件名... 卸载软件

· Yum服务端：为客户端自动解决依赖关系，安装软件包 classroom.example.com

搭建完Web服务，共享了光盘所有的内容

验证：http://classroom.example.com

1. 指定yum软件源：

- 服务端（软件仓库）

集中提供软件安装包，并提供依赖支持

- 客户端（yum命令及配置）

· yum相关文件位置

- /etc/yum.conf

- /etc/yum.repos.d/\*.repo

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

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

#用vim编辑器修改repo文件信息

[rhel7] #仓库标识

name=rhel 7.0 #仓库描述信息

baseurl=http://classroom.example.com/content/rhel7.0/x86\_64/dvd/ #仓库路径

enabled=1 #本文件是否生效，1为Y，0为N

gpgcheck=0 #是否检测红帽签名，1为Y，0为N

[root@server0 ~]# yum repolist #列出仓库信息，检测是否能发现Yum服务端

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

[root@server0 ~]# yum -y install sssd #安装sssd软件

[root@server0 ~]# yum -y install gcc #安装gcc软件

· 真机搭建Yum仓库：

· 真机 服务端：将CentOS 7.4光盘内容显示在系统中：

[root@room9pc01 ~]# mkdir /dvd

[root@room9pc01 ~]# mount /iso/rhel-server-7.4-x86\_64-dvd.iso /dvd

mount: /dev/loop1 写保护，将以只读方式挂载

[root@room9pc01 ~]# ls /dvd/

多媒体：包组

yum groups list [hidden] #查看列表

yum groups install 多媒体 #安装

* 1. 文本查找
     1. grep过滤操作

· 根据字符串提取文本行

- grep [选项] '指定字符串'

· 选项：

· -v ：取反匹配

· -i ：忽略大小写

[root@server0 ~]# grep 'root' /etc/passwd

[root@server0 ~]# grep -i 'root' /etc/passwd #忽略大小写

[root@server0 ~]# grep -v 'root' /etc/passwd #不包含root的行

· 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@server0 ~]# grep 'root' /etc/passwd

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

[root@server0 ~]# grep 'root$' /etc/passwd

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

* + 1. 重定向输出

· 将命令行的正常输出保存到文件

- 覆盖式：命令行 > 文件

- 追加式：命令行 >> 文件

* + 1. echo 'xxx' 将字符串输出到终端
    2. 重定向整理

· 重定向

< 读重定向

> 写重定向

>> 追加重定向

<< 终止符

stdin 0 标准输入(读)

stdout 1 标准输出(写)

stderr 2 标准错误(写)

* 1. wget 命令行下载工具

1. [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
   1. 网络参数的配置
      1. 设置永久主机名：修改配置文件 /etc/hostname
      2. 配置永久ip地址，子网掩码，网关地址

[root@Gay ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0

* + 1. nmcli连接管理

a) 查看识别的网卡的名字

[root@Gay ~]# nmcli connection show

b) 配置ip地址 子网掩码 网关地址

[root@Gay ~]# nmcli connection modify 'System eth0'

连接网络 修改 修改网卡的名称

ipv4.method manual/auto

修改ipv4的方法 手动/自动 模式选择

ipv4.addresses '172.25.0.100/24 172.25.0.254'

配置ipv4的地址 ip + 掩码 + 网关

connection.autoconnect yes

每次开机自动启用

c) 激活配置：

[root@Gay ~]# nmcli connection up 'System eth0'

* + 1. 配置永久DNS服务器地址：将网站域名解析为对应的IP地址

a)修改配置文件：/etc/resolv.conf

[root@Gay ~]# echo 'nameserver 172.25.254.254' > /etc/resolv.conf

[root@Gay ~]# cat /etc/resolv.conf

nameserver 172.25.254.254

[root@Gay ~]# nslookup server0.example.com

Server: 172.25.254.254

Address: 172.25.254.254#53

Name: server0.example.com

Address: 172.25.0.11

* 1. find查找文档位置
     1. 根据预设的条件递归查找对应的文件

- find [目录] [条件1] [-a|-o] [条件2]...

- 常用条件表示：

-type 类型(f文本文件、d目录、l快捷方式)

-name "文档名称"

-size +|-文件大小(k,M,G)

-user 用户名

-maxdepth n 查找深度

-mtime +10 十天之前

-mtime -10 最近十天之内

[root@Gay ~]# ls /boot/grub/menu.lst

[root@Gay ~]# ll /boot/grub/menu.lst f

[root@Gay ~]# find /boot/ -type d

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

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

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

[root@Gay ~]# touch nsd01.txt nsd02.txt

[root@Gay ~]# mkdir nsd1801

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

/root/nsd01.txt

/root/nsd02.txt

/root/nsd1801

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

/root/nsd1801

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

/root/nsd01.txt

/root/nsd02.txt

* + 1. find结果处理

- 使用find命令的 -exec操作

- find.. .. -exec处理命令{} \;

- 优势：以{}代表所有结果，逐个处理，遇 \;结束

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

/boot/initramfs-0-rescue-946cb0e817ea4adb916183df8c4fc817.img

/boot/initramfs-3.10.0-123.el7.x86\_64.img

/boot/initramfs-3.10.0-123.1.2.el7.x86\_64.img

[root@Gay ~]# find /boot/ -size +10M -exec cp {} /opt/ \;

[root@Gay ~]# ls /opt/

initramfs-0-rescue-946cb0e817ea4adb916183df8c4fc817.img

initramfs-3.10.0-123.1.2.el7.x86\_64.img

initramfs-3.10.0-123.el7.x86\_64.img

rh

[root@Gay ~]# find /boot/ -name "vm\*"

/boot/vmlinuz-3.10.0-123.el7.x86\_64

/boot/vmlinuz-0-rescue-946cb0e817ea4adb916183df8c4fc817

/boot/vmlinuz-3.10.0-123.1.2.el7.x86\_64

[root@Gay ~]# find /boot/ -name "vm\*" -exec cp {} /opt/ \;

[root@Gay ~]# ls /opt/

initramfs-0-rescue-946cb0e817ea4adb916183df8c4fc817.img

initramfs-3.10.0-123.1.2.el7.x86\_64.img

initramfs-3.10.0-123.el7.x86\_64.img

rh

vmlinuz-0-rescue-946cb0e817ea4adb916183df8c4fc817

vmlinuz-3.10.0-123.1.2.el7.x86\_64

vmlinuz-3.10.0-123.el7.x86\_64

[root@Gay ~]# mkdir /root/findfiles

[root@Gay ~]# find /root/ -name "nsd\*" -type f

/root/nsd01.txt

/root/nsd02.txt

[root@Gay ~]# find /root/ -name "nsd\*" -type f -exec cp {} /mnt/ \;

[root@Gay ~]# ls /mnt/

nsd01.txt nsd02.txt

1. Day 05 管理用户组 tar备份与恢复 NTP时间同步 cron计划任务
2. 管理用户和组
   * 1. 用户帐号：

a)可以登录系统 b)实现访问控制（不同用户不同权限）

组帐号：方便对用户帐号进行管理 权限方面

唯一标识：UID GID

· 管理（root）：UID永远是0

· 组帐号：基本组（私有组） 附加组（公共组 从属组）

· Linux中，一个用户至少属于一个组

* + 1. 添加用户

用户基本信息存放在/etc/passwd文件（户口本）

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

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

用户名:密码占位符号:UID:基本GID:用户描述信息:家目录位置:默认的解释器

· 使用useradd命令名

· 常用命令选

- useradd [选项]... 用户项

- 用户uid、-d家目录路径、-s登陆为Shell（指定登陆的解释器程序）、-G附加组

- /sbin/nologin：不能登陆操作系统

- id 用户名 查看用户基本信息，如用户不存在，显示未查找到该用户

- groupadd 创建组

实例：

[root@server ~]# useradd nsd01

[root@server ~]# grep 'nsd01' /etc/passwd

nsd01:x:1001:1001::/home/nsd01:/bin/bash

[root@server ~]# useradd -u 2333 nsd02

[root@server ~]# grep 'nsd02' /etc/passwd

nsd02:x:2333:2333::/home/nsd02:/bin/bash

[root@server ~]# useradd -d /opt/nsd03 nsd03

[root@server ~]# grep 'nsd03' /etc/passwd

nsd03:x:2334:2334::/opt/nsd03:/bin/bash

[root@server ~]# useradd -s /sbin/nologin nsd04

[root@server ~]# grep 'nsd04' /etc/passwd

nsd04:x:2335:2335::/home/nsd04:/sbin/nologin

[root@server ~]# groupadd terena #创建组terena

[root@server ~]# useradd -G terena nsd05

[root@server ~]# id nsd05

uid=2336(nsd05) gid=2337(nsd05) 组=2337(nsd05),2336(terena)

* + 1. 更改密码

· 使用passwd命令

- passwd [用户名] #交互式设置密码

- echo '密码' | passwd --stdin 用户名 #非交互式设置密码

· su命令：可以实现命令行临时切换身份

· 管道命令符"|"：将前面命令的输出作为后面命令的输入

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

halt:x:7:0:halt:/sbin:/sbin/halt

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

* + 1. 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登陆Shell

- -g基本组

- -G附加组

* + 1. 删除用户

· 使用userdel命令

- userdel [-r] 用户名 #包括家目录一并删除

* + 1. 密码信息的存放

/etc/shadow

* + 1. 添加组

组基本信息存放在/etc/group文件中

[root@server ~]# head -1 /etc/group

root:x:0:

组名:组密码占位符:组ID:本组成员列表

· 使用groupadd命令

- groupadd [-g 组ID] 组名

· 使用gpasswd命令

- gpasswd -a 用户名 组名 #添加成员

- gpasswd -d 用户名 组名 #删除成员

* + 1. 修改组属性

· 使用groupmod命令

- groupmod [-g 组ID] [-n 新组名] 组名

* + 1. 删除组

· 使用groupdel命令

- groupdel 组名

* 1. tar备份与恢复
     1. 归档和压缩

· 归档的含义

- 将许多零散的文件整理成一个文件

- 文件总大小基本不变

· 压缩的含义

- 案某种算法减小文件所占用空间的大小

- 恢复时按对应的逆向算法解压

Windows：归档、压缩一气呵成

Linux：先归档，再压缩

Linux独有的压缩工具：gzip bzip2 xz

* + 1. tar工具的常用选项

· 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

- -P：保持归档内文件的绝对路径

注意：所有的操作都有f选项，且必须放在所有选项的最后

tar 选项 tar包名字 被归档的文件1 被归档的文件2...

root@server0 opt]# tar -cPf file01.tar /etc/passwd /home/ /boot/

[root@server0 opt]# ls

file01.tar rh #归档结果

[root@server0 opt]# gzip file01.tar

file01.tar.gz rh #打包结果

[root@server0 opt]# tar -czPf file02.tar.gz /etc/passwd /home/ /boot/

/etc/passwd #一步到胃

[root@server0 opt]# ls

file01.tar.gz file02.tar.gz rh #打包结果

tar解包

[root@server0 opt]# tar -xf file01.tar.gz

tar: 从成员名中删除开头的“/”

[root@server0 opt]# ls

boot etc file01.tar.gz file02.tar.gz home rh

指定解压路径的选项 -C

[root@server0 opt]# tar -xf file01.tar.gz -C /root/opt

案例：

[root@server0 opt]# tar -cjPf /root/backup.tar.bz2 /usr/local/

[root@server0 opt]# ls /root/backup.tar.bz2

/root/backup.tar.bz2

* 1. NTP时间同步
     1. 查看日期和时间

[root@server0 opt]# date

2018年 02月 05日 星期一 16:08:54 CST

· 修改时间

[root@server0 opt]# date -s "年-月-日 时-分-秒"

* + 1. NTP时间同步

· 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

服务端：Linux系统上的软件，服务端软件 classroom.example.com

客户端：客户端软件 server.example.com

a)安装chrony客户端软件

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

[root@server0 opt]# rpm -q chrony

chrony-1.29.1-1.el7.x86\_64

b) 配置chrony客户端软件

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

server classroom.example.com iburst

c)重启chrony客户端软件服务

[root@server0 opt]# systemctl restart chronyd #重启服务

[root@server0 opt]# systemctl enable chronyd #设置开机自启程序

d: daemon：守护进程

d)验证：

[root@server0 opt]# date -s "1997-5-20 20:20"

1997年 05月 20日 星期二 20:20:00 CST

[root@server0 opt]# systemctl restart chronyd

[root@server0 opt]# date

2018年 02月 05日 星期一 16:44:08 CST

* 1. cron计划任务
     1. 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

· 软件包：cronie、crontabs

· 系统服务：crond

· 日志文件：/var/log/crond

* + 1. 如何编写crontab任务记录

· 配置格式可参考/etc/crontab文件

时间 执行任务

- 分 时 日 月 周 任务命令行（绝对路径）

\* \* \* \* \*

# "\*" 代表任意，"\*/n"代表频率，"-"代表连续时间，","代表不连续时间，24小时制

· 使用crontab命令

- 编辑：crontab -e [-u 用户名]

· grep 中匹配空行

- '^$' 代表空行

- grep -v '^$' 目录

1. Day 06 权限和归属 使用LDAP认证 家目录漫游
2. 权限和归属
   * 1. 基本权限

a) 访问方式（权限）

- 读取：允许查看内容-read

- 写入：允许修改内容-write

- 可执行：允许运行和切换-execute

· 对于文本文件：

r:cat less head tail

w:vim 能够保存

x:Shell脚本

b) 权限适用对象（归属）

- 所有者：拥有此文件/目录的用户-user

- 所属组：拥有此文件/目录的组-group

- 其他用户：除所有者、所属组以外的用户-other

* + 1. 查看权限

a)命令ls -l

b)开头含义：

· 以 - 开头：文本文件

· 以d开头：目录

· 以l开头：快捷方式（链接）

c) 设置基本权限

· 使用chmod命令

- chmod [-R] 归属关系+-=权限类别 文档

[-R]：递归设置

d) 如何判断Linux中用户的权限

· 查看用户对于该文档 所属的身份 匹配即停止

· 查看相应身份的权限

常见的提示信息：权限不足

Permission denied

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

e) 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...

* + 1. 附加权限（特殊权限）

a) Set GID

· 附加在属组的x位上

- 属组的权限标识会变为s（s覆盖执行权限x，分大小写区别原执行权限）

- 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

- 继承父目录所属组身份

b) 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（禁止操作别人文档）

c) Set UID

· 附加再属主的x位上

- 属主的执行标识会变成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

* + 1. acl访问控制列表

a)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

b) 命令设置：

· 使用getfacl、setfacl

- setfacl [-R] -m u:用户名:权限类别 文档... #添加策略

- setfacl [-R] -m g:组名:权限类别 文档...

- setfacl [-R] -x u:用户名 文档... #删除指定策略

- setfacl [-R] -b 文档... #清空策略

* 1. 使用LDAP认证
     1. 什么是LDAP？

· 轻量级目录访问协议

- 由服务器来集中存储并向客户端提供的信息，存储方式类似于DNS分层结构

- 提供的信息包括：用户名、密码、通信录、主机名映射记录、……

· 为一组客户机提供可登陆的用户帐号

- 本地用户：/etc/passwd /etc/shadow

- 网络用户：用户名、密码信息存储在LDAP服务端

服务端：classroom.example.com

客户端：指定LDAP服务端位置（域名）

a)安装一个客户端sssd软件

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

b) authconfig-gtk图形的配置

[root@server0 ~]# yum -y install authconfig-gtk

c) authconfig-gtk 启动

选择LPAD

dc=example,dc=com

classroom.example.com

TLS加密，证书加密

systemctl restart sssd

systemctl enable sssd

* 1. 家目录漫游
     1. 共享服务

· NFS共享：网络文件系统

- 由NFS服务器将指定的文件夹共享给客户机

- 客户机将此共享目录mount到本地目录，访问此共享资源就像访问本地目录一样方便

- 类似于EXT4、XFS等类型，只不过资源在网上

· 客户端：server0.example.com

1)查看共享classroom.example.com

[root@server0 ~]# showmount -e classroom.example.com

Export list for classroom.example.com:

/home/guests 172.25.0.0/255.255.0.0

[root@server0 ~]# mkdir /home/guests

[root@server0 ~]# mount 172.25.254.254:/home/guests /home/guests

[root@server0 ~]# su - ldapuser0

* 1. autofs工具配置
     1. 装包

yum -y install autofs

* + 1. 修改主配置文件

vim /etc/auto.master

/home/guests /etc/auto.guests #设置触发式挂载

:wq

* + 1. 修改次级配置

\* -rw classroom.example.com:/home/guests/& (&:和前一个\*内容相同)

:wq

* + 1. 重启服务

systemctl restart autofs

systemctl enable autofs

1. ENGINEER
2. Day 01 分区规划及使用 LVM逻辑卷
3. 分区规划及使用
   * 1. 硬盘分区管理

· 每个扇区，512字节

· 识别硬盘 => 分区规划 => 格式化 => 挂载使用

· 格式化 -> 文件系统的作用：数据在空间中的排列规则

· lsblk 查看本机识别的硬盘

[root@server0 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

vda 253:0 0 10G 0 disk

└─vda1 253:1 0 10G 0 part /

vdb 253:16 0 10G 0 disk

* + 1. 分区规划

· MBR/msdos分区模式（主引导记录分区方案）

- 1~4个主分区，或者0~3个主分区+1个扩展分区(n个逻辑分区)

- 最大支持容量为2.2TB的硬盘

- 扩展分区不能格式化

- 分区类型：主分区 扩展分区 逻辑分区

- 一个硬盘最多只能有4个主分区

· 使用fdisk分区工具 - 交互式管理工具

- 查看分区表

fdisk -l /dev/sda

- 修改硬盘分区表

fdisk 硬盘设备

· 常用交互指令

m 列出指令帮助

p 查看现有分区表

n 新建分区

d 删除分区

q 放弃更改并退出

w 保存更改并退出

* + 1. 分区格式化

· 常用的格式化工具

- mkfs工具集

mkfs.ext3 分区设备路径

mkfs.ext4 分区设备路径

mkfs.xfs 分区设备路径

mkfs.vfat -F 32 分区设备路径

- 查看分区设备信息

blkid 设备路径

* + 1. 挂载使用

df -h 查看挂载分区信息

· 挂载vdb1分区

[root@server0 ~]# mkdir /part1

[root@server0 ~]# mount /dev/vdb1 /part1/

[root@server0 ~]# df -h

文件系统 容量 已用 可用 已用% 挂载点

/dev/vda1 10G 3.0G 7.0G 31% /

devtmpfs 906M 0 906M 0% /dev

tmpfs 921M 80K 921M 1% /dev/shm

tmpfs 921M 17M 904M 2% /run

tmpfs 921M 0 921M 0% /sys/fs/cgroup

/dev/vdb1 2.0G 33M 2.0G 2% /part1

/dev/vdb2 2.0G 33M 2.0G 2% /part2

* + 1. 综合分区

· 最多4个主分区，如果需要创建更多的分区，则将一个主分区换成扩展分区，在扩展分区上创建逻辑分区

· 刷新命令

- partprobe

· 使用磁盘空间：

1.查看识别的磁盘 lsblk

2.划分分区 fdisk

3.刷新分区 partprobe

4.格式化分区 mkfs.格式 设备分区

5.挂载使用 mount

6.实现开机自动挂载 /etc/fstab

· 实现开机自动挂载

- 配置文件/etc/fstab的记录格式

设备路径 挂载点 文件系统类型 参数 备份标记 检测顺序

/dev/vdb1 /part1 xfs defults 0 0

/dev/vdb2 /part2 xfs defults 0 0

* 1. LVM逻辑卷
     1. 作用：

a)整合分散的空间

b)逻辑卷空间可以扩大

c)支持线上操作（不需要umount）

* + 1. 创建逻辑卷：

a)创建物理卷

b)创建卷组（可以同时创建物理卷）

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

Physical volume "/dev/vdc1" successfully created

Physical volume "/dev/vdc2" successfully created

Volume group "systemvg" successfully created

[root@server0 ~]# pvs #查看物理卷信息

PV VG Fmt Attr PSize PFree

/dev/vdc1 systemvg lvm2 a-- 10.00g 10.00g

/dev/vdc2 systemvg lvm2 a-- 10.00g 10.00g

[root@server0 ~]# vgs #查看卷组信息

VG #PV #LV #SN Attr VSize VFree

systemvg 2 0 0 wz--n- 19.99g 19.99g

c) 从卷组中划分逻辑卷

· 命令格式lvcreate -L 大小 -n逻辑卷名字 卷组名字

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

Logical volume "mylv" created

[root@server0 ~]# lvs

LV VG Attr LSize Pool Origin Data% Move Log Cpy%Sync Convert

mylv systemvg -wi-a----- 16.00g

d) 使用逻辑卷

[root@server0 ~]# mkfs.ext4 /dev/systemvg/mylv

[root@server0 ~]# mkdir /test

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

/dev/systemvg/mylv /test ext4 defaults 0 0

[root@server0 ~]# mount -a

[root@server0 ~]# df -h

文件系统 容量 已用 可用 已用% 挂载点

/dev/vda1 10G 3.0G 7.0G 31% /

devtmpfs 906M 0 906M 0% /dev

tmpfs 921M 80K 921M 1% /dev/shm

tmpfs 921M 17M 904M 2% /run

tmpfs 921M 0 921M 0% /sys/fs/cgroup

/dev/vdb1 2.0G 33M 2.0G 2% /part1

/dev/vdb2 2.0G 33M 2.0G 2% /part2

/dev/mapper/systemvg-mylv 16G 45M 15G 1% /test

* + 1. 扩展逻辑卷

a)卷组有足够的空间

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

Extending logical volume mylv to 18.00 GiB

Logical volume mylv successfully resized

[root@server0 ~]# lvs

LV VG Attr LSize Pool Origin Data% Move Log Cpy%Sync Convert

mylv systemvg -wi-ao---- 18.00g

[root@server0 ~]# vgs

VG #PV #LV #SN Attr VSize VFree

systemvg 2 1 0 wz--n- 19.99g 1.99g

[root@server0 ~]# df -h|tail -1

/dev/mapper/systemvg-mylv 16G 45M 15G 1% /test

· 扩展空间后，还需要扩展文件系统的大小

1)扩展文件系统的大小

ext4:resize2fs

xfs:xfs\_growfs

[root@server0 ~]# resize2fs /dev/systemvg/mylv #刷新文件系统

[root@server0 ~]# df -h|tail -1

/dev/mapper/systemvg-mylv 18G 44M 17G 1% /test

b)卷组没有足够的空间

扩展逻辑卷的空间

· 逻辑卷也可以缩减

a)缩减文件系统

b)缩减空间

* + 1. 卷组划分空间的单位：PE

PE的大小可以更改，vgchange -s命令

* + 1. 删除逻辑卷

删除逻辑卷(lv) --> 删除卷组 --> 删除物理卷

1. Day 02 Shell脚本基础 使用变量 条件测试及选择 列表式循环
2. Shell脚本基础
   * 1. 脚本：

· 一个可以运行的文本文件，运行后可实现某种功能

· 提前设计可执行语句，用来完成特定的任务的文件

- 解释型程序

- 顺序、批量执行

· 将命令都写入文本文件，然后赋予文本文件执行权限（命令的堆积）

· 非交互式

- 需要提前设计、智能化难度大

- 批量执行、效率高

- 方便在后台静默执行

· 规范：

- #! 环境声明（声明下列可执行代码用什么程序翻译）

- # 注释文本

- 可执行代码

* + 1. 简单脚本技巧

· 使用 | 管道操作

- 将前一条命令的标准输出交给后一条命令处理

· 免交互处理

- 脚本一般在后台执行，要尽量减少人工交互的语句

· 脚本输出信息的处理

- 记录有价值的信息（>> /var/log/foo.log）

- 屏蔽无价值的、干扰性的信息:黑洞设备(垃圾场w)（&> /dev/null）

- 关于重定向输出：

> ：只收集前面命令的正确输出，将其写入文本文件

2> ：只收集前面命令的错误输出，将其写入文本文件

&> ：收集前面命令的正确和错误输出，将其写入文本文件

- 自定义输出：echo '文本字符串'

3.为了增加脚本适应多变的环境、多变的需求，以及为了方便性，我们要使用变量

- 变量：会变化的量，以不变的名称，存储可以变化的值

- 变量的定义：变量命=变量值

· 为了降低脚本使用的复杂性，使用交互式

- read：产生交互的方式，将用户从键盘的输入记录作为输出赋值给一个变量

read -p '请输入你要创建的用户名:' user

· 设置变量时的注意事项

- 若指定的变量名已存在，相当于为此变量重新赋值

- 等号两边不能有空格

- 变量名由字母/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

· 基本格式

- 引用变量值：$变量名

- 查看变量值：echo $变量名;和常量混用：echo ${变量名}常量

· 补充：

$[] ：运算

' ' ：让所有的特殊字符变成普通字符

$() ：将命令的输出结果作为参数，与``作用相同

* + 1. 变量的种类

· 环境变量

- 常见的环境变量

PWD、PATH、USER、LOGNAME

SHELL、HOME

USER:当前登陆的用户名

- 环境变量区分字母大小写

· 位置变量

- 在执行脚本时提供的命令行参数

· 预定义变量

$# 已加载的位置变量的个数

$\* 所有位置变量的值

$? 程序退出后的状态值，0表示正常，其他值异常

* 1. 条件测试及选择
     1. 条件测试

· 常用的测试选项

-e ：文档存在为真

-d ：存在并且为目录，才为真

-f ：存在并且为文本文件，才为真

-r ：存在并且有读取权限，才为真

-w ：存在并且有写入权限，才为真

-x ：存在并且有执行权限，才为真

· 比较整数（带e字母的都有 等于 二字）

-gt ：大于

-ge ：大于等于

-eq ：等于

-ne ：不等于

-lt ：小于

-le ：小于等于

· 字符串对比

==:相等为真

!=:不相等为真

* + 1. if选择结构

· 双分支：当条件满足/不满足时，分别作xx、yy处理

· 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

经典脚本之成绩：

用户输入考试成绩

大于等于85 输出 优秀

大于等于70 输出 良好

大于等于60 输出 合格

以上条件均不满足 输出 再牛的肖邦，也弹不出哥的悲伤

* 1. 列表式循环
     1. 适用于反复工作的场景
     2. for循环处理

· for 变量名 in 值列表

do

扎针 抽血

done

1. Day 03 系统安全保护 配置用户环境 配置高级连接 防火墙策略管理
2. 系统安全保护
   * 1. SELinux概述

· 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及其以上）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)

|

|重启

|reboot

V

- disable(彻底禁用)

查看SELinux状态：getenforce

· 切换运行模式

- 临时切换：setenforce 1|0

- 固定配置：/etc/selinux/config 文件

* + 1. 防火墙策略管理

· 作用：隔离

- 硬件防火墙

- 软件防火墙

· 搭建web

- 服务端：server0.example.com

1)安装服务端软件 httpd(完善，均衡) Nginx(并发量高，自动化运维) Tomcat(Java)

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

2)启动httpd服务，设置为开机自启

[root@server0 ~]# systemctl restart httpd

[root@server0 ~]# systemctl enable httpd

3)写自己的主页

默认网页文件路径：/var/www/html

默认网页文件的名字：index.html

默认书写网页内容的语言：html

- 客户端：desktop0.example.com

· 搭建ftp

- 服务端：server0.example.com

1) 安装服务端软件

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

2) 启动vsftpd服务，设置为开机自启

[root@server0 ~]# systemctl restart vsftpd

[root@server0 ~]# systemctl enable vsftpd

3) 默认共享路径：/var/ftp

- 客户端：desktop0.example.com

· 防火墙：firewalld服务基础

* + 1. RHEL7的防火墙体系

a) 系统服务：firewalld

b) 管理工具：firewall-cmd、firewall-config

c) 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等少数几个服务

- trusted：允许任何访问

- block：阻塞任何来访请求

- drop：丢弃任何来访的数据包

- ......

· 配置规则的位置

- 运行时 (runtime)

- 永久配置选项 (-permanent)

· 匹配规则：原则 匹配即停止

· 防火墙判定进入哪一个区域的规则：

1) 查看客户端请求数据包中，源ip地址，查看自己所有的区域，哪一个区域有该源ip地址进入哪一个区域

2) 如果规则1不符合则进入默认区域

· 预设安全区域

- public：仅允许访问本机的sshd等少数几个服务

- trusted：允许任何访问

- block：阻塞任何来访请求（明确拒绝，客户端会接收到拒绝信息）

- drop：丢弃任何来访的数据包（直接丢弃，节省服务器的资源）

d) 防火墙对源ip地址的控制

· 方式1-宽松模式：默认区域未trusted，将想要拒绝的源ip地址放入到block或drop中

· 方式2-严格模式：默认区域未block或drop，将想要允许的源ip地址放入到trusted中

* 1. 端口：数字 编号，用来标识进程或程序
     1. 默认端口号

· ftp：21

· ssh:22

· Telnet:23

· SMTP:25

· DNS:53

· DHCP:UDP 67/68

· tftp:69

· http：80

· NTP:123

· https:443

* + 1. 端口转发

· 利用防火墙进行对指定端口的转发

# firewall-cmd --permanent --zone=public

--add-forward-port=port=5423:proto=tcp:toport=80

* 1. 配置高级连接
     1. 配置高级连接（聚合连接 网卡绑定 链路聚合） 参考 man teamd.conf 全文查找/example 按n跳转匹配项

· 热备份（activebackupo）连接冗余（活跃状态 备份状态）

eth1 eth2

team0虚拟网卡

* + 1. 制作网卡绑定

1. 制作虚拟网卡team0

# nmcli connection add type team

autoconnect yes con-name team0 ifname team0

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

# nmcli connection 添加 类型为 team 的设备

每次开机自动启用 配置文件命名为 team0 网卡显示的名字为 team0

team0网卡内部成员工作模式为 '{"runner": {"name": "activebackup"}}' （热备）

# ifconfig

1. 为 team0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

# 添加 类型为 team-slave 的设备 配置文件命名为 team0-1 网卡为eth1

主设备为team0

* + 1. 配置team0的ip地址与激活

# nmcli connection modify team0 ipv4.method manual

ipv4.addresses 192.168.1.1/24 connection.autoconnect yes

# nmcli connection up team0 #激活team0网卡

# nmcli connection up team0-1 #激活team0-1成员

# nmcli connection up team0-2 #激活team0-2成员

如果错了：

1. 删除

[root@server0 ~]# nmcli connection delete team0

[root@server0 ~]# nmcli connection delete team0-1

[root@server0 ~]# nmcli connection delete team0-2

1. 重新配置

专用于team测试查看的命令

[root@server0 ~]# teamdctl team0 state #查看team0信息

[root@server0 ~]# ifconfig eth1 down #禁用eth1网卡

[root@server0 ~]# teamdctl team0 state #查看team0信息

* 1. IPv6地址
     1. IPv4地址：32个二进制 点分隔4个部分 十进制表示
     2. IPv6地址：128个二进制 冒号分隔8个部分 十六进制表示

1. Day 04 配置SMB共享 配置NFS共享
   1. 配置永久主机名：server0.example.com

/etc/hostname

* 1. 配置静态ip地址

nmcli connection modify 'System eth0' ipv4.method manual ipv4.addresses '172.25.0.11/24 172.25.0.254' connection.autoconnect yes

nmcli connection up 'System eth0'

* 1. 配置DNS服务器地址：172.25.254.254

将域名解析成IP地址

echo 'nameserver 172.25.254.254' > /etc/resolv.conf

cat /etc/resolv.conf

nslookup classroom.example.com

* 1. 搭建yum服务端

echo '[rhel7]

> name=rhel 7.0

> baseurl=http://classroom.example.com/content/rhel7.0/x86\_64/dvd/

> enabled=1

> gpgcheck=0' > /etc/yum.repos.d/dvd.repo

* 1. 配置SMB共享
     1. 设置双端防火墙默认区域为trusted

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

* + 1. 服务名：smb

包名：samba

* + 1. 配置SMB共享 跨平台的共享 Windows与Linux
    2. Samba服务基础

· Samba软件项目

- 用途：为客户机提供共享使用的文件夹

- 协议：SMB(TCP 139)、CIFS(TCP 445)

· 所需软件包：samba

· 系统服务：smb

* + 1. 服务端：server0.example.com

1. 安装软件包：samba

yum -y install samba

1. 建立samba共享帐号

useradd harry

useradd kenji

useradd chihiro

echo redhat | passwd --stdin harry

echo redhat | passwd --stdin kenji

echo redhat | passwd --stdin chihiro

1. 管理共享帐号

· Samba用户——专用来访问共享文件夹的用户

- 采用独立设置的密码

- 但需要提前建立同名系统用户

· 使用pdbedit管理工具

- 添加用户： pdbedit -a 用户名

- 查询用户： pdbedit -L 用户名

- 删除用户： pdbedit -x 用户名

· 示例：

pdbedit -a harry #将本地用户harry设置为Samba共享帐号

pdbedit -a kenji

pdbedit -a chihiro

交互设置密码为123

pdbedit -L #显示本地有哪些Samba共享帐号

harry:1001:

chihiro:1003:

kenji:1002:

· 配置文件及参数

修改/etc/samba/smb.conf

[自定共享名]

path = 文件夹绝对路径

[global]

workgroup = STAFF

[common]

path = /common

1. 重启smb服务

systemctl restart smb

systemctl enable smb

1. 设置SELinux

布尔值——功能的开关

· 查看服务状态(加-P永久实现)

getsebool -a | grep samba

setsebool samba\_export\_all\_ro on

* + 1. 客户端：desktop0.example.com

1)所需软件包：samba-client

yum -y install samba-client

2) 列出共享资源

- smbclient -L 服务器地址

3) 连接到共享文件夹

- smbclient -U harry //server0/common

* + 1. 采用更加方便科学的访问方式(挂载访问)

· 客户机上操作：

1)安装软件包

yum -y install cifs-utils

2)挂载

mount -o user=harry,pass=123 //172.25.0.11/common /common

3)查看挂载结果

ls /common

4)客户端实现开机自动挂载 /etc/fstab

//172.25.0.11/common /common cifs defaults,user=harry,pass=123,\_netdev 0 0

#\_netdev：标识本设备为网络设备(先启动网络服务具备ip地址等网络参数后，再进行挂载)

umount /common #卸载已挂载目录

df -h #查看卸载是否成功

mount -a #挂载

df -h #检验挂载结果

* + 1. 实现读写的samba共享

· 服务端：server0.example.com

1) 修改配置文件，设置新的共享

vim /etc/samba/smb.conf

[devops]

path = /devops

write list = chihiro

2) 重建相应的目录

mkdir /devops

echo abc > /devops/abc.txt

ls /devops

3) 重启smb服务，设置开机自启

systemctl restart smb

systemctl enable smb

4) 设置本地权限

setfacl u:chihiro:rwx /devops

getfacl /devops

5) 修改SELinux功能开关

getsebool -a | grep samba

setsebool samba\_export\_all\_rw on

6) 修改防火墙设置

· 客户端：desktop0.example.com

1)vim /etc/fstab

//172.25.0.11/devops /devops cifs defaults,user=chihiro,pass=123,\_netdev 0 0

2)systemctl restart smb

7.多用户(multiuser)的samba共享,专为普通用户设计

· SMB客户端的multiuser挂载技术

- 管理员只需要作一次挂载

- 客户端在访问挂载点时，若需要不同权限，可以临时切换为新的共享用户(无需重新挂载)

· 实现方式

1)挂载SMB共享时启用multiuser支持

2)使用cifscreds临时切换身份

7.1 客户端操作：

· 修改/etc/fstab配置文件，添加参数

- multiuser，提供客户端多个用户身份的区分支持

- sec=ntlmssp，提供NT局域网管理安全支持

vim /etc/fstab

//172.25.0.11/devops /devops cifs defaults,user=kenji,pass=123,\_netdev,multiuser,sec=ntlmssp 0 0

umount /devops

mount -a

df -h

su - student

cifscreds add -u chihiro 172.25.0.11

Password:

ls /mnt/dev

touch /mnt/dev/haha.txt

exit

* 1. 配置NFS共享
     1. NFS共享概述

· 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

* + 1. 搭建NFS

· 服务端：server0.example.com

rpm -q nfs-utils

1）修改/etc/exports

- 文件夹路径 客户机地址（权限） 客户机地址（权限）

使用exportfs可以重载更新过的配置

- exportfs -r

vim /etc/exports

/public \*(ro)

2）创建共享目录

mkdir /public

cp /etc/passwd /public/passwd.txt

systemctl restart nfs-server

systemctl enable nfs-server

· 客户端：

mkdir /public

showmount -e 172.25.0.11 #查看有哪些nfs共享

vim /etc/fstab #实现开机自动挂载

172.25.0.11:/public /public nfs defaults,\_netdev 0 0

mount -a

df -h

* 1. 分区规划
     1. 查看识别的磁盘 lsblk
     2. 划分分区

fdisk /dev/vdb

lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

vda 253:0 0 10G 0 disk

└─vda1 253:1 0 10G 0 part /

vdb 253:16 0 10G 0 disk

├─vdb1 253:17 0 2G 0 part

├─vdb2 253:18 0 2G 0 part

└─vdb3 253:19 0 2G 0 part

* + 1. 将第一个主分区格式化为xfs文件系统，开机自动挂载到/mnt/mypart

# mkfs.xfs /dev/vdb1

# blkid /dev/vdb1 //查看文件系统格式

# mkdir /mnt/mypart

# echo '/dev/vdb1 /mnt/mypart xfs defaults 0 0' >> /etc/fstab

# tail -1 /etc/fstab

# mount -a

# df -h | tail -1

* + 1. 将第二个主分区和第三个主分区组成卷组systemvg

vgcreate systemvg /dev/vdb[2-3]

* + 1. 划分逻辑卷lvtest，大小为3G，格式化为ext4格式，挂载到/mnt/mylv

lvcreate -L 3G -n lvtest systemvg

mkfs.ext4 /dev/systemvg/lvtest

blkid /dev/systemvg/lvtest

mkdir /mnt/mylv

mount /dev/systemvg/lvtest

df -h | tail -1

* + 1. 逻辑卷lvtest扩大到6G

1)通过fdisk分出4G的逻辑分区vdb5

2)partprobe

3)vgextend /dev/vdb5

4)lvextend -L 6G /dev/systemvg/lvtest

5)resize2fs /dev/systemvg/lvtest #如果是xfs格式，则用xfs\_growfs命令刷新文件系统

1. Day 05 iSCSI技术应用 数据库服务基础 管理表数据

远程端笔记下载方法：

yum -y install git

git clone http://github.com/redhatedu/course //完整下载

cd course

git

pull //更新下载（必须在course目录下执行）

git checkout 文件名 //下载某个文件

1. iSCSI共享(磁盘共享)internet scsi
   * 1. iSCSI磁盘的工作模式，Internet SCSI，网际SCSI接口

- 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

· iSCSI磁盘的构成（需要的三个概念）

- 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

- 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

· target的共享名有要求(要符合iqn规范)

iqn.年-月.反转域名:任意字串

如：

iqn.2018-2.com.example:jpg

* + 1. 环境前提:

修改两台虚拟机的防火墙：

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

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

* + 1. 实验步骤：

a) 在server0上准备一个磁盘分区(vdb1)3G

b) 安装软件包，修改配置,重启服务

# yum -y install targetcli

# targetcli

/> ls

/> cd /

//建立后端存储

/> /backstore/block create (name=)nsd (dev=)/dev/vdb1

/\*\*

\* /> /iscsi create iqn.8102-12.com.example:server0

\* 创建一个iscsi共享（target磁盘组,客户端访问目标)

\* 磁盘组共享命名遵循iqn原则(qin.yyyy-mm.倒序域名:自定义标识)

\*\*/

/> /iscsi create iqn.2018-02.com.example

//创建逻辑单元，把共享名和后端的设备通过lun关联在一起

/> /iscsi/iqn.2018-02.com.example.com/tgp1/luns create /backstores/block/back\_store

/\*\*

\* 创建一个访问的口令:iqn.2018-02.com.example:desktop0，以后仅知道口令的客户端才可以访问共享

\*设置访问控制,设置客户端访问需要声明的名称

\*\*/

/> /iscsi/iqn.2018-02.com.example:data/tpg1/acls create iqn.2018-02.com.example:desktop0

//以后客户端访问本机（172.25.0.11）的3260端口(默认端口)可以访问到共享

/> /iscsi/iqn.2018-02.com.example:data/tpg1/portals create 172.25.0.11

//重启服务

# systemctl restart target

# systemctl enable target

c) 客户端访问共享

//查看硬盘设备

# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

vda 253:0 0 10G 0 disk

└─vda1 253:1 0 10G 0 part /

vdb 253:16 0 10G 0 disk

//实验环境可以跳过，生产环境需要检查iscsi是否安装

# yum -y install iscsi

//修改配置文件,指定口令,重启服务

# vim /etc/iscsi/initiatorname.iscsi

# systemctl daemon-reload

# systemctl restart iscsid

//查看、复制命令格式

# man iscsiadm

关键词:example

//挂载172.25.0.11服务器上的共享：

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

172.25.0.11:3260,1 iqn.2018-02.com.example:data

# iscsiadm --mode node --targetname iqn.8102-07.com.example:server0 --portal 172.25.0.11:3260 --login

//如果挂载提示authentication,口令不对

# systemctl restart iscsid

# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 3G 0 disk

vda 253:0 0 10G 0 disk

└─vda1 253:1 0 10G 0 part /

vdb 253:16 0 10G 0 disk

//可以看到sda硬盘成功挂载

* 1. 数据库服务基础
     1. DB(database)数据库：存放数据的地方

存放形式： 数据表，数据列，表头

DBMS（数据库管理系统）是具体的产品：excel,access,mysql,sql server,oracle

* + 1. MySQL AB ==> Sun MySQL ==> Oralce MySQL

MariaDB == MySQL

RHEL7(内置的数据库是MariaDB)

* + 1. 在server0操作

- mariadb-server:提供服务端有关的系统程序,默认端口3306

# yum -y install mariadb-server mariadb

//mariadb-server服务器

//mariadb客户端(为mariadb-server前置)

//使用客户端软件连接数据库服务器做增、删、改、查操作

# systemctl start mariadb

# systemctl enable mariadb

mysql

//查看有哪些目录

MariaDB [(none)]> show databases;

//进入数据库

MariaDB [(none)]> use mysql

//查看有哪些数据表

MariaDB [mysql]> show tables;

//查看数据表里的数据

MariaDB [mysql]> select \* from user;

//退出数据库

MariaDB [mysql]> exit

/\*为数据库管理员设置密码\*/

· 数据库管理员root:数据库MariaDB最高权限用户,存放在mysql库中user表下

· 系统管理员root:Linux系统最高权限用户,存放在/etc/passwd文件中

· 格式:# mysqladmin [-u用户名] [-p旧密码] password '新密码'

# mysqladmin -uroot password '123456' //给mariadb数据库的管理员root修改密码，密码为123456

//直接登陆数据库会失败

# mysql

/\*交互式设置密码\*/

# mysql -u root -p

Enter password:

//使用用户与密码登陆

# mysql -uroot -p123456

//创建数据库nb

MariaDB [(none)]> create database nb;

//切换至数据库nb

MariaDB [(none)]> use nb;

· 禁止监听,只服务于本机

# vim /etc/my.cnf

[mysql]

skip-network

# systemctl restart mariadb

MariaDB [(nb)]> create table info

-> (id int, //第一列是id，数据类型为int型

-> name varchar(10), //第二列是name，数据类型为字符串，10位

-> password varchar(20)); //第三列是password，数据类型为字符串，20位

//创建一个表，表名info

MariaDB [(nb)]> describe info;

//查看创建的表

MariaDB [nb]> insert into info values

-> (1,'tom','123456');

//向info中插入数据

MariaDB [nb]> insert into info values

-> (1,'tom','123456'),

-> (2,'dachui','654321'),

-> (3,'jerry','123456'),

-> (4,'lucy','123456');

//插入多条数据的格式

MariaDB [nb]> select \* from info;

· 跨库查询

MariaDB [nb]> select \* from mysql.user;

//查看info的所有数据

MariaDB [nb]> drop table nb;

//删除名为nb的表

* + 1. 数据库的备份与还原

# mysqldump -uroot (-p123456) nb > my.bak

//把名称为nb的数据库备份到my.bak

# mysql -uroot (-p123456) nb < my.bak

//使用my.bak将名为nb的数据库还原

* + 1. 用户的创建

- 默认mariadb数据库仅有一个帐号（root)

- 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'

· 创建新的账户

//授予账户tom对nb数据库下所有表的所有权限

//all指所有权限

//可以写具体的指令[select,insert]

//tom@'localhost'（%表示任意） 表示tom只能从本机登陆数据库

//identified by设置密码

# mysql -uroot -p123456

MariaDB [nb]> grant all on nb.\*

-> to tom@'localhost'

-> identified by '123456';

* + 1. 表记录的操作（表数据的操作）:增删查改

//查看表中所有数据

MariaDB [nb]> select \* from info;

//查看info表中name与password信息

MariaDB [nb]> select name,password from info;

//查看info表中满足password为123456的条件的信息

MariaDB [nb]> select \* from info where password='123456';

//查看表结构

MariaDB [mysql]> desc user[表名称];

//更新info表

MariaDB [nb]> update info set password = 'mmp'

-> where name='tom';

//更新name为Lily的行的密码为123

//删除info表中id为3的数据

MariaDB [nb]> delete from info where id=3

//查看info表中的数据

//数据符合密码为123456且id等于1

MariaDB [nb]> select \* from info

-> where password='123456' and id=1;

//多表查询

MariaDB [nb]> SELECT count(\*)

-> FROM base,location

-> WHERE base.name="Barbara" AND

-> location.city="Sunnyvale" AND

-> base.id=location.id;

* + 1. 禁止空密码root用户访问数据库

MariaDB [none]> use mysql;

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

MariaDB [mysql]> delete from user where password='';

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

MariaDB [mysql]> flush privileges;

MariaDB [mysql]> exit

# mysql -u root -h server0.example.com

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

· 登陆不成功为正确

1. Day 06 HTTP服务基础 网页内容访问 部署动态网站
2. HTTP服务基础
   * 1. 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 Markup Language,超文本标记语言(html)

· Hyper Text Transfer Protocol,超文本传输协议

- 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)

- DirectoryIndex:起始页/首页文件名(index.html)

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

* + 1. 实验一:搭建基本的WEB服务器

虚拟机Server0:

1.安装httpd(Apache)软件包

# systemctl install -y httpd

2.启动httpd服务,设置开机自启动

# systemctl start httpd

# systemctl enable httpd

3.书写页面

# vim var/www/html/index.html

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

4.虚拟机Desktop0访问测试

# firefox 172.25.0.11

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

DNS服务器:将域名解析为IP地址

classroom.example.com上的域名

server0.example.com

www0.example.com

webapp0.example.com

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

* + 1. 实验二:配置网站的DNS域名

ServerName:本站点注册的DNS名称(空缺)

虚拟机Server0:

1.修改配置文件:

路径: /etc/httpd/conf/httpd.conf

95行,注释去掉

ServerName server0.example.com:80

# systemctl restart httpd

虚拟机Desktop0:

# firefox server0.example.com

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

* + 1. 实验三:修改网页文件存放路径

DocumentRoot:网页根目录(var/www/html)

虚拟机Server0

1.修改配置文件

路径: /etc/httpd/conf/httpd.conf

2.创建路径及网页文件

# mkdir /var/www/myweb

# echo '<h1>Wo Shi MyWeb' > /var/www/myweb/index.html

3.重启服务

# systemctl restart httpd

4.访问测试

# firefox server0.example.com

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

server0.example.com

服务端:/var/www/myweb/pub

=====================================================

* 1. 虚拟Web主机
     1. 作用:由同一台服务器提供多个不同的Web站点

·区分方式(构建方式)

- 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

配置文件:

/etc/httpd/conf/httpd.conf //主配置文件

/etc/httpd/conf.d/\*.conf //子配置文件

IP:端口

<VirtualHost \*:80>

ServerName DNS名称

DocumentRoot 网页根目录

</VirtualHost>

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

* + 1. 实验四:搭建基于域名的虚拟Web主机

虚拟机Server0:

1.创建子配置文件

# vim /etc/httpd/conf.d/dc.conf

<VirtualHost \*:80>

ServerName www0.example.com

DocumentRoot /var/www/nsd01

</VirtualHost>

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/nsd02

</VirtualHost>

2.创建网页根目录

# mkdir /var/www/nsd01 /var/www/nsd02

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

# echo '<h1>wo shi nsd02' > /var/www/nsd02/index.html

3.重启服务

# systemctl restart httpd

注意:

一旦使用虚拟Web主机,主配置文件中ServerName与DocumentRoot失效,所有站点都需要使用虚拟Web主机来实现

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

配置网页内容访问

· 使用<Directory>配置区段

- 每个文件夹自动继承父目录的访问控制

- 除非针对子目录有明确设置

· 格式:

<Directory />

AllowOverride none

Require all denied

</Directory>

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

* + 1. 实验五: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

虚拟机Server0

1.查看http://server0.example.cpm 的DocumentRooot目录

# vim /etc/httpd/conf.d/nsd01.conf

2.创建目录,建立网页文件

# mkdir /var/www/myweb/private

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

# cat /var/www/myweb/private/index.html

<h1>wo shi private

3.本机及虚拟机desktop0分别测试

# firefox server0.example.com/private

# curl server0.example.com/private/

4.从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

# vim /etc/httpd/conf.d/tc.conf

<Directory /var/www/myweb/private>

Require ip 172.25.0.11

</Directory>

# systemctl restart httpd

5.本机及虚拟机Desktop0分别测试

# firefox server0.example.com/private

# curl server0.example.com/private/

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

* + 1. 使用自定义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

# cat /webroot/index.html

<h1>wo shi webroot

2.修改虚拟Web主机配置文件

/etc/httpd/conf.d/dc.conf

<VirtrualHost \*:80>

ServerName server0.example.com

DocumentRoot /webroot

</VirtualHost>

3.修改访问控制配置文件

/etc/httpd/conf.d/tc.conf

<Directory /webroot>

Require all granted

</Directory>

4.重启httpd服务

5.SELinux策略,安全上下文值(路径的进入及配置文件读取)

· 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

//查看安全上下文

# ls -Zd /var/www

# ls -Zd /webroot

# chcon -R --reference=/var/www /webroot

# ls -Zd /webroot

# chcon -t public\_content\_t /var/ftp/log2.tar

=======================================================

* 1. 读部署动态网站

· 静态页面

- 服务端原始页面

- 无数据处理

· 动态页面

- php

- wsgi

- jsp

* + 1. 虚拟机Server0

1.部署python页面

# cd /var/www/nsd02

# wget http://classroom.example.com/pub/materials/webinfo.wsgi

# cat webinfi.wsgi

2.方便用户的访问,页面跳转

# vim /etc/httpd/conf.d/nsd01.conf

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/nsd02/webinfo.wsgi

Alias / /var/www/nsd02/webinfo.wsgi

#当客户端访问网页站根目录时,实现页面跳转,将webinfo.wsgi呈现

</VirtualHost>

3.重启服务验证

# system restart httpd

# firefox webapp0.example.com

4.安装python脚本语言翻译软件,负责解析python页面的代码0

# yum install -y mod\_wsgi

5.修改虚拟Web主机配置文件,实现httpd进行转义和翻译

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/nsd02/

WSGIScriptAlias / /var/www/nsd02/webinfo.wsgi

</VirtualHost>

6.重启httpd服务验证

# systemctl restart httpd

# firefox webapp0.example.com

# curl webapp0.example.com

· UNIX时间戳:从1970-1-1 0:0:0算起,到达现在所经历q的秒数

7.修改虚拟主机侦听在端口8909

Listen 8909

<VirtualHost \*:8909>

ServerName webapp0.example.com

DocumentRoot /var/www/nsd02/

WSGIScriptAlias / /var/www/nsd02/webinfo.wsgi

</VirtualHost>

8.SELinux策略,非默认端口的开放

# semanage port -l

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

-a:添加 -t:类型 -p:协议

# systemctl restart httpd

# firefox webapp0.example.com

# curl webapp0.example.com

· 端口优先级大于域名

1. Day 07 综合串讲 综合练习
2. HTTPS网站加密

HTTP是明文协议，网络中传输的任何数据都是明文，包括用户和密码，如果有人抓包，所有数据都可以获得

* + 1. 加密算法：

对称算法（AES,DES）

非对称算法（RSA,DSA）

信息摘要（md5,sha512,sha265）

* + 1. 对称加密（加密和解密是一把钥匙）适合单机加密
    2. 非对称密码（加密和解密不是一把钥匙）：

公钥和私钥

信息摘要：

# md5sum 文件名

数据完整性校验（检查数据是否被人修改过）

* + 1. 虚拟机Server0

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

# wget http://classroom.example.com/pub/example-ca.crt

# ls

3.部署私钥

# cd /etc/pki/tls/private

# wget http://classroom.example.com/pub/tls/private/server0.key

# ls

4.安装软件包mod\_ssl,提供安全支持

# yum install -y mod\_ssl

5.修改配置文件

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

:set nu //添加行号

59 DocumentRoot "/var/www/html"

60 ServerName www0.example.com:443

//指定网站证书的位置

100 SSLCertificateFile /etc/pki/tls/certs/server0.crt

//指定私钥的位置

107 SSLCertificateKeyFile /etc/pki/tls/private/server0.key

//指定根证书的位置

122 SSLCACertificateFile /etc/pki/tls/certs/example-ca.crt

6.重启httpd服务

7.验证

# firefox https://www0.example.com

我了解风险 --> 添加例外 --> 确认添加例外

自己新建加密格式

<VirtualHost \*:443>

sslengine on

SSLCaCertificateFile /

SSLCertificateFile /

SSLCertificateKeyFile /

</VirtualHost>

=====================================================

* 1. 基础邮件服务
     1. 种类

· 本域邮件

· 外域邮件

· 电子邮件服务器的基本功能

- 为用户用户提供电子邮件存储空间(用户名@域名)

- 处理用户发出的邮件(SMTP) ---- 传递给收件服务器

- 处理用户收到的邮件(pop3 IMAP) ---- 投递到邮箱

· mail命令发信/收信

//发信

- mail -s '标题' -r 发件人 收件人[@收件域]

//收信

- mail [-u]

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

虚拟机Server0

1.安装postfix包

# yum -y install postfix

# rpm -q postfix

2.修改配置文件

# vim /etc/postfix/main.cf

:set nu //添加行号

99 myorigin = server0.example.com //默认补全的域名后缀

116 inet\_interfaces = all //允许所有人使用邮件服务

164 mydestination = server0.example.com //判断为本域邮件

3.重启服务

# systemctl restart postfix

4.创建本地用户进行收发邮件

# useradd yg //杨过

# useradd gg //姑姑

&1 //查看编号为1

&q //退出

===================================================

* 1. 分区工具fdisk,parted

· fdisk分区只能分4个主分区

· fdisk分区每个分区最大只能是2T

· parted可以分超过4个主分区（128），分区大小可以大于2T

· parted分区工具

- parted分区首先要选择分区的类型msdos,gpt模式

- msdos就是fdisk使用的模式

# parted /dev/vdb

(parted) mktable gpt //指定分区模式

(parted) print //查看分区表信息

(parted) mkpart //划分新的分区

分区名称？ [ ]? nsd //分区名称

文件系统类型？ [ext2]? ext4 //分区文件系统,未实装

起始点？ 0

结束点？ 2G

警告: The resulting partition is not properly aligned for best performance. //需要引导空间

忽略/Ignore/放弃/Cancel? Ignore //选择忽略

(parted) print

Model: Virtio Block Device (virtblk)

Disk /dev/vdb: 10.7GB

Sector size (logical/physical): 512B/512B

Partition Table: gpt

Disk Flags:

Number Start End Size File system Name 标志

1 17.4kB 2000MB 2000MB nsd

(parted) unit GB //选择显示容量单位为GB

(parted) mkpart

分区名称？ [ ]?

文件系统类型？ [ext2]? ext4

起始点？ 2G

结束点？ 4G

(parted) print

Model: Virtio Block Device (virtblk)

Disk /dev/vdb: 10.7GB

Sector size (logical/physical): 512B/512B

Partition Table: gpt

Disk Flags:

Number Start End Size File system Name 标志

1 0.00GB 2.00GB 2.00GB nsd

2 2.00GB 4.00GB 2.00GB

(parted) quit

# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

vda 253:0 0 10G 0 disk

└─vda1 253:1 0 10G 0 part /

vdb 253:16 0 10G 0 disk

├─vdb1 253:17 0 1.9G 0 part

└─vdb2 253:18 0 1.9G 0 part

=====================================================

* 1. 交换空间

· 相当于虚拟内存

- 当物理内存不够用时,使用磁盘空间来模拟内存

- 在一定程度上缓解内存不足的问题

- 交换分区:以空闲分区充当的交换空间

- 交换文件:以文件模拟的设备充当的交换空间

* + 1. 创建交换空间

1)前提:要有空闲的分区

2)格式化文件系统

# mkswap /dev/vdb1

3)启用交换空间

# swapon /dev/vdb1

4)查看启用的交换空间

# swapon -s

5)停用交换空间

# swapoff /dev/vdb1

· 交换空间开机自动挂载

# vim /etc/fstab

/dev/vdb1 swap swap defaults 0 0

/dev/vdb2 swap swap defaults 0 0

# swapon -a

# swapon -s

1. SERVICES
2. Day 01 扩展的几个应用 发布网络YUM源 vim编辑技巧 源码编译安装 systemctl控制
3. 补充应用
   * 1. man hier

· 主要用途

/boot 存放系统引导必须的文件,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

======================================================

* + 1. 搭建教学环境

· 采用真机和虚拟机架构

1.真机: 搭建FTP服务,共享光盘所有内容

1)安装vsftpd软件

# yum -y install vsftpd

# rpm -q vsftpd

2)查看服务启动

# systemctl status vsftpd

# systemctl restart vsftpd

# systemctl enable vsftpd

3)共享光盘所有内容

· 默认共享:/var/ftp

· 服务端: 1.众多的软件包 2.仓库清单文件 3.共享的服务

# vim /etc/fstab

/var/lib/libvirt/images/iso/rhel-server-7.4-x86\_64-dvd.iso /var/ftp/rhel7 iso9660 defaults 0 0

# ls /var/ftp/rhel7

利用真机FTP服务,共享rhel6及CentOs7光盘内容

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

· 虚拟机

· 利用root进行登录,密码为123456

虚拟机A:

1.主机名:

# ehco 'svr7.tedu.cn' > hostname

# hostname svr7.tedu.cn

# exit

# login

2.配置ip地址:192.168.4.7/24

3.配置Yum仓库(以真机作为源)

虚拟机B:

1.主机名:pc207.tedu.cn //同上

2.配置ip地址:192.168.4.207/24

3.配置Yum仓库(以真机作为源)

# clone-vm7

· 补充内容:本地数据传递给网络中其他主机

# scp /etc/yum.repos.d/rhel7.repo root@192.168.4.207:/etc/yum.repos.d/

==============================================================

* + 1. 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

# mkdir /nsd01

# ll -d /nsd01

# chmod 700 /nsd01

# ll -d /nsd01

# chmod 007 /nsd01

# ll-d /nsd01

# chmod 750 /nsd01

# ll -d /nsd01

· 附加权限

SUID，SGID，Sticky

SUID仅能对程序有效

/usr/bim/passwd(root root)程序属于root用户

以普通用户去执行有SUID的命令，执行时会获得root权限

# chmod 7777 文件 --> (rwsrwsrwt)

======================================================

* + 1. 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

# vim /etc/profile

HISTSIZE=1000 //默认记录1000条

* + 1. 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)

* + 1. date查看计算机时间

# date

# date +%F 日期

# date +%R 时间

# date +%Y%m%d 年，月，日year,month,day

# date +%Y:%m:%d 年，月，日

# date +%H:%M:%S 小时，分，秒

# date -s "年-月-日 时:分:秒" //修改计算机时间

# date -s "11:01" //仅修改时间

# date -s "2088-12-1 12:12"

======================================================

* + 1. 软链接和硬链接

· ln命令（给文件或目录创建快捷方式，链接）

· 软链接（符号链接）

- 软链接 --> 原始文档 --> i节点 --> 文档数据

软链接不占用空间，但是源文件删除，链接失效

# ln -s 源 目标

# ln -s /root/文档/xxd/ /root/桌面/xxd

· 硬链接

- 硬链接 --> i节点 --> 文档数据

硬链接不占用空间，源文件可以删除，链接依然能用

硬链接与原始文件必须在同一分区下

- i节点(编号) : 标识硬盘存储区域

# ln 源 目标

* + 1. 查看帮助的方法

# 命令 -h

# 命令 --help

可用 | grep 筛选有效信息

# man 命令

对man帮助时可以/搜索

- 5 配置文件的帮助信息

· man 5 passwd

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

* + 1. zip归档工具,跨平台的压缩归档工具

linux压缩常用格式：gzip,bz2

windows压缩：rar,zip

· 归档+压缩操作

# zip -r my.zip /var/log //把/var/log目录压缩

· 释放归档+压缩

# unzip my.zip //把my.zip解压到当前

# unzip my.zip -d /tmp //把my.zip解压到/tmp

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

* + 1. vim编辑技巧

· 命令模式：

yy,#yy 复制一行,#行

p,P 粘贴到下一行,上一行

dd,#dd 剪切一行,#行(多用于删除)

hjkl 左下上右

gg 到第一行

G 到最后一行

4G 移动光标到第4行（同 :4 ）

x(DELETE) 删除光标当前的一个字符

^(HOME) 光标到行首

$(END) 光标到行尾

d^ 从光标处删除到行首

d$ 从光标处删除到行尾

C 从光标处删除到行尾并且进入插入模式

u 撤销一步

Ctrl+r 取消撤销

ZZ 保存并退出

· 末行模式：

:4 移动到光标第4行

:w 保存（不退出）

:wq 保存退出

:q! 不保存退出

:r 读取其他文件 :r /etc/passwd

:w 另存为i/root/tmp.txt :w /root/tmp.txt

:s/旧/新 旧内容替换为新内容，仅替换当前行的第一个内容

:s/旧/新/g 替换当前行所有内容

:1,10 s/旧/新/g 替换1-10行所有的内容

:%s/旧/新/g 替换所有的内容

:set nu 显示行号

:set nonu 不显示行号

:set ai 启用自动缩进

:set noai 关闭自动缩进

=======================================================

* + 1. 源码包安装软件

a)编译安装的优势

· 二进制包可以安装软件[rpm,exe,msi,deb]

· 源码 ---> 编译 ---> 二进制

· Linux的软件多数都免费，开源

· 二进制包装软的缺点：

（不愿意花时间去封装二进制）

有些开源软件会封装成二进制，但时间会很久

· 源码编译安装：

- 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

b)从ftp下载

# wget ftp://172.25.0.250/share/inotify-tools-3.13.tar.gz

· 登陆server0使用源码安装软件

# tar -xf inotify-tools-3.13.tar.gz

//解压

# cd inotify-tools-3.13/

//计算机的系统软件一般都是用c语言写的

//如QQ，office,windows,linux,画图，播放器

java,php,python,shell,C,汇编

# ./configure 检查你的计算机环境

# yum -y install gcc

//gcc是linux里面的一个C语言的解释器

#./configure 检查环境，不报错没有Error

# make

//用gcc解释器把源码转换为二进制

# make install

//把编译好的二进制程序安装到你的计算机

# inotifywait 这个能<tab>出来（说明成功）

//备注：执行./configure --prefix=/路径

//configure可以通过prefix参数，指定安装路径

//如果没有指定prefix，则一般默认在usr/local/

inotifywait -mrq /root/

再开一个终端，在/root目录做一些操作

# touch /root/tmp.txt

# echo "1" > 1.txt

# rm -f 1.txt

=====================================================

* + 1. systemctl命令

# systemctl -t service

//列出启动的服务

# systemctl -t service --all

//列出所有的服务，包括没成功的

# systemctl stop 服务名称

//当前关闭，重启无效

# systemctl disable服务名称

//永久关闭

# systemctl enable 服务名称

//开启自启

# systemctl start 服务名称

//当前立刻启动

# systemctl restart 服务名称

//重启服务

# systemctl status 服务名称

//查看某个服务的状态

=======================================================

* 1. 发布网络Yum源
     1. yum排错

1) 格式是否正确

# cat /etc/yum.repos.d/rhel7.repo

2) 查看是否有光盘内容

# curl ftp://192.168.4.254/rhel7/

没有: 1.是否ping通 2.真机vsftpd服务是否开启

3.ls /var/ftp/rhel7 --> 没有挂载 --> /etc/fstab

4.真机防火墙是否关闭

* + 1. 传输tool包到虚拟机svr7

[root@svr7 ~]# mkdir /usr/xxd

[root@room9pc01 ~]# scp 桌面/tools.tar.gz root@192.168.4.7:/usr/xxd

[root@svr7 ~]# ls /usr/xxd

tools.tar.gz

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

* + 1. 虚拟机svr7:

· 创建自定义yum源:

1.解包

# tar -xf /usr/xxd/tools.tar.gz -C /usr/xxd/

# ls /usr/xxd/tools/other/

2.生成仓库数据文件

# createrepo /usr/xxd/tools/other/

3.修改配置文件

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

[myrpm]

name=my rpm

baseurl=file:///usr/xxd/tools/other

enable=1

gpgcheck=0

4.验证

# yum repolist

# rpm -q sl //查看是否安装

# rpm -ql sl //查看安装了什么

# oneko & //在后台运行

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

1. Day 02 DNS服务基础 特殊解析 DNS子域授权 缓存DNS
2. i权限：加上i权限，所有用户包括root都不能修改或删除文件

# lsattr 文档

//查看是否具有特殊权限

# chattr +i

# chattr -i

//添加/去除特殊权限

* + 1. 利用root进入虚拟机 密码123456

虚拟机A

1.配置eth0 永久静态ip地址：192.168.4.7/24

2.配置永久主机名：svr7.tedu.cn

# hostname svr7.tedu.cn

# echo 'svr7.tedu.cn' > /etc/hostname

虚拟机B

1.配置eth0 永久静态ip地址：192.168。4.207

2.配置永久主机名：pc207.tedu.cn

# hostname pc207.tedu.cn

# echo 'pc207.tedu.cn' > /etc/hostname

* + 1. 在真机上配置远程管理的别名，进行远程管理（/root/.bashrc）

alias goa='ssh -X root@192.168.4.7'

alias gob='ssh -X root@192.168.4.207'

* + 1. 为虚拟机A与虚拟机B 搭建Yum仓库

1. 光盘搭本地yum
2. 真机ftp/http搭网络yum
3. 客户端配置yum

[rhel]

name=rhel7

baseurl=ftp://rhel7

enabled=1

gpgcheck=0

* + 1. 检查防火墙状态和SELinux状态

===================================================

* 1. DNS服务基础

1. DNS解析的作用

· 为什么要DNS系统 -- 域名比IP好记

· 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

1. DNS域名管理

· IANA,互联网数字分配机构

· CNNIC,中国互联网络信息中心

1. 域名体系

a)所有域名,都要以'.'来结尾

根： 根.

一级域名：.cn .kr .hk .tw .jp .us .com .net .org ...

二级域名：.com.cn .edu.cn .net.cn .gov.cn .mil.cn ...

三级域名：.dawai.com.cn .nb.com.cn .haxi.com.cn .dc.com.cn

完整主机名：www.dawai.com.cn ftp.dawai.com.cn tts.dawai.com.cn

· FQDN完全合格主机名

- 站点名.域名后缀

- 站点名. .. .. .二级域.一级域

· 域名代理/注册/购买服务商

- 新网

- 万网

- 中国互联

1. BIND域名服务

· BIND

- 伯克利 Internet 域名服务

· 主配置文件:/etc/named.conf 　　//设置本机负责解析的域名

· 地址库配置文件:/var/named/ 　　//所有的完整的主机名与ip对应关系

－系统服务：named

－默认端口：TCP/UDP 53

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

1. 搭建基本的DNS服务

· 虚拟机A:

1.安装软件包

yum -y install bind-chroot bind

//域名服务包

bind

//提供虚拟根支持(牢笼政策)

bind-chroot

2.修改主配置文件

· 养成备份的习惯

# cp /etc/named.conf /root/named.bak

· 配置文件字段

//以下语句未定义时,默认为any

listen-on port 53 { any; };

allow-query { any; };

options {

directory "/var/named"; //指定地址库文件存放路径

};

zone "tedu.cn" IN { //指定本DNS服务器负责解析的域名

type master; //指定本机为权威DNS服务器(主DNS)

file "tedu.cn.zone"; //指定地址库文件的名字

};

3.建立地址库文件/var/named/tedu.cn.zone

//保证named用户对地址库文件有读权限

# cp -p /var/named/named.localhost /var/named/tedu.cn.zone

# ll /var/named/tedu.cn.zone

-rw-r-----. 1 root named 152 6月 21 2007 /var/named/tedu.cn.zone

# vim /var/named/tedu.cn.zone

tedu.cn. NS svr7

svr7 A 192.168.4.7

www A 1.2.3.4

ftp A 5.6.7.8

NS 字段前:本机解析的域名 字段后:本机域名

A字段前:域名 字段后:ip

AAAA -- ipv6

· 第一个A字段:本机 字段前:本机域名 字段后:本机ip

· 从第二个A字段开始 解析其他主机(DNS主要功能)

tips:

· 地址库文件所有域名都要以'.'结尾(解析域名和本机域名)

· 没有以'.'结尾则默认补全以地址库

· A:正向解析记录

4.重启named服务

虚拟机B:验证

1.指定DNS服务器

# echo nameserver 192.168.4.7 > /etc/resolv.conf

# nslookup www.tedu.cn

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

* 1. 多区域的DNS服务器

· 虚拟机A:

1.修改主配置文件

# vim /etc/named.conf

· 添加第二个解析域名

zone "qq.com" IN {

type master;

file "qq.com.zone";

};

2.添加地址库文件

# cp -p /var/named/named.localhost /var/named/qq.com.zone

# vim /var/named/qq.com.zone

qq.com. NS svr7

svr7 A 192.168.4.7

www A 1.2.3.4

ftp A 3.3.3.3

3.重启named服务

# systemctl restart named

· 虚拟机B:验证

# nslookup www.qq.com

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

* 1. 特殊的解析记录

1. DNS负载均衡(解析结果的轮询)

　qq.com. NS svr7

　svr7 A 192.168.4.7

　www A 1.2.3.1

　www A 1.2.3.2

　www A 1.2.3.3

　www A 1.2.3.4

　ftp A 3.3.3.3

1. 泛域名解析

qq.com. NS svr7

　svr7 A 192.168.4.7

　www A 1.2.3.1

　www A 1.2.3.2

www A 1.2.3.3

　www A 1.2.3.4

ftp A 3.3.3.3

\* A 11.22.33.44

qq.com. A 6.6.6.6

1. 有规律的泛域名解析

　pc1.qq.com --> 192.168.10.1

　pc2.qq.com --> 192.168.10.2

　pc3.qq.com --> 192.168.10.3

　......

pc50.qq.com --> 192.168.10.50

DNS服务内置的变量：$GENERATE 可以产生连续的数字

$GENERATE 1-50 pc$ A 192.168.10.$

1. 解析记录的别名

· 虚拟机A:

# vim /var/named/qq.com.zone

　　tts　　CNAME　　ftp

# tts解析结果与ftp相同

# systemctl restart named

· 虚拟机B:

# nslookup tts.qq.com

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

* 1. DNS子域授权

1. 父域与子域

· 父域:www.tedu.cn

· 子域:www.bj.tedu.cn www.sz.tedu.cn www.zz.tedu.cn

虚拟机A负责解析父域tedu.cn

虚拟机B负责解析父域bj.tedu.cn

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

1. 虚拟机B：

1.装包 bind-chroot bind

2.修改配置文件 vim /etc/named.vonf

3.建立地址库文件　cp -p /var/named/named.localhost /var/named/xxx

4.重启服务　systemctl restart named

1. 虚拟机Ａ：

1.子域授权

# vim /var/named/tedu.cn.zone

tedu.cn. NS svr7

bj.tedu.cn. NS pc207

svr7 A 192.168.4.7

pc207 A 192.168.4.207

www A 1.2.3.4

ftp A 5.6.7.8

2.重启服务

虚拟机Ｂ:验证

# nslookup www.bj.tedu.cn

Server: 192.168.4.7

Address: 192.168.4.7#53

Non-authoritative answer: //非权威解答

Name: www.bj.tedu.cn

Address: 1.2.3.4

· 递归解析：首选DNS服务器，跑到相应其他DNS服务器上，询问最终将结果带回来的过

程（客户端与首选DNS服务器的交互）

options {

directory "/var/named";

recursion no

};

· 迭代解析：首选DNS服务器与其他DNS服务器之间的交互

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

* 1. 主机映射文件/etc/hosts(为本机提供DNS域名解析)

# vim /etc/hosts

192.168.4.110 www.sina.com

# ping www.sina.com

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

总结：客户端DNS解析域名过程

1.客户端查询/etc/hosts

2.查询/etc/resolv.conf 查看DNS服务器

3.DNS服务器进行反馈(递归查询与迭代查询)

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

* 1. 缓存DNS,加速解析效率
     1. 真机挂载光盘

# mkdir /dvd

# mount /ISO/CentOS-7-x86\_64-DVD-1708.iso /dvd

* + 1. 真机书写配置文件

# cd /etc/yum.repos.d/

# mkdir repos

# mv \*repo repos/

# vim dvd.repo

[dvd]

name=CentOS 7.4

baseurl=file:///dvd

enabled=1

gpgcheck=0

# yum repolist

# yum -y install bind-chroot bind

* + 1. 搭建缓存DNS服务器

a)改配置

　# vim /etc/named.conf

　options {

　 directory "/var/named";

　 forwarders { 176.19.0.26; }; //转发给达内DNS服务器

　};

b)启服务

　# systemctl restart named

====================================================

* 1. 管理运行级别（运行模式）

RHEL5、RHEL6 //切换运行级别的命令 init

0：关机

1：单用户模式（破解密码、修复系统）

2：字符模式（不支持网络）

3：字符模式（支持网络）

4：无定义

5：图形模式

6：重启

=====================================================

RHEL7 运行模式

multi-user.target 字符模式（支持网络）

graphical.target 图形模式

临时切换

# systemctl isolate multi-user.target

# systemctl isolate graphical.target

永久改变默认的运行模式

# systemctl get-default

//查看当前默认的运行模式

# systemctl set-default graphical.target

//修改默认的运行模式

1. Day 03 Split分离解析 RAID磁盘阵列 进程管理 日志管理
2. 回顾
3. 常见的DNS服务器有哪些

主DNS 一级域名DNS 二级域名DNS 三级域名DNS 根域DNS

* + 1. 常见的DNS资源解析记录

NS A PTR(反向解析) CNAME

=============================================================

* 1. Split分离解析(视图解析)
     1. 概述

· 当受到客户机的DNS查询请求的时候

- 能够区分客户机的来源地址

- 能够为不同类别的客户机提供不同的解析结果(IP地址)

- 判断客户端来源,不同客户端解析同一个域名得到不同解析结果

- 意义:让客户端访问网络中最近的服务器

· 典型适用场景:CDN提供的内容分发服务

· 注意点:

1.客户端分类合理,所有客户端都要匹配分类

2.分类匹配由上到下,依次匹配,匹配即停止

1）同一个区域(sina.com)在多个视图内分别定义，其他地址库文件相互独立，从而实现解析结果的分离

2）定义view视图后，不允许在view以外出现zone配置

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

* + 1. 环境及需求

- 权威DNS:svr7.tedu.cn 192.168.4.7

- 负责区域:sina.com

- svr7记录分离解析:以www.sina.com为例

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. 分离解析

-修改配置文件

view "nsd"{

match-clients { 192.168.4.207;.. ..; }; //客户端源IP地址

zone "sina.com" IN {

type master;

file "sina.com.zone";

};

};

view "other"{

match-clients { any; };

zone "sina.com" IN {

type master;

file "sina.com.other";

};

};

-建立地址库文件sina.com.zone和sina.com.other

-重启服务

· 对客户机ip抽象封装

· 多区域的分离解析

acl myip{ 192.168.4.8; 192.168.4.123; 192.168.4.66; 192.168.4.58 };

view "nsd" {

match-clients { myip };

zone

...

};

==================================================================

* 1. RAID磁盘阵列

· 廉价冗余磁盘阵列

- 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

- 至少需要2块磁盘

· 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

· 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4块磁盘

· RAID 5，高性价比模式

- 相当于RAID0和RAID1的折中方案

- 需要至少3块磁盘的容量来存放校验数据

· RAID 6，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4块磁盘的容量来存放校验数据

RAID阵列实现方式

· 硬RAID:由RAID控制卡管理阵列

- 主板 --> 阵列卡 --> 磁盘 --> 操作系统 --> 数据

· 软RAID:由操作系统来管理阵列

- 主板 --> 磁盘 --> 操作系统 --> RAID软件 --> 数据

==============================================================

* 1. 进程管理

程序:静态的代码,占用硬盘空间

进程:动态执行的代码,占用内存和CPU

父进程 子进程 --- 树型结构

进程的唯一编号:PID 越小越优先运行

* + 1. 查看进程树

· pstree

-格式:pstree [选项] [PID或用户名]

· 常用命令选项

-a:显示完整的命令行

-p:列出对应PID编号

systemd - 所有进程的父进程

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

# pstree

# pstree lisi

# pstree -p lisi

# pstree -ap lisi

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

· ps—— Processes Snapshot

-格式:ps [选项] ...

· 常用命令选项

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

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

· ps aux 操作

- 列出正在运行的所有进程

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

- 列出正在运行的所有进程

- 可以看到进程的父进程PID

PPID：父进程的PID号

PRI/NI：进程优先级，数值越小优先级越高

· wc -l统计行数

统计进程数:

# ps -aux | wc -l

# ps -aux | cat -n | tail -1

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

* + 1. 进程动态排名

# top [-d 刷新秒数] [-U 用户名]

· top交互操作指令

- ?：查看帮助

- P、M：根据%CPU、%MEM降序排列

- T：根据进程消耗的TIME降序排列

- k：杀死指定的进程

- q：退出top程序

* + 1. 进程检索

# pgrep [选项] ... 查询条件

· 支持模糊查询

· 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

-t：检索指定终端的进程

-x：精确匹配完整的进程名

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

# pgrep - log

# pgrep -lU lisi -t pts/1

# pgrep -lU lisi -t pts/2

//pts/1 <--> 终端号

=====================================================

* + 1. 控制进程

· 进程的前后台调度

· 前台启动

- 输入正常命令行，运行期间占用当前终端

· 后台启动

- 在命令行末尾添加"&"符号，不占用当前终端

· Ctrl + z组合键

- 挂起当前进程（暂停并转入后台）

· jobs命令

- 查看后台任务列表

· fg命令

- 将后台任务恢复到前台运行

· bg命令

- 激活后台被挂起的任务

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

# sleep 1000 & //将进程放入后台运行

# sleep 800

ctrl+z //将进程暂停放入后台

# bg 2 //将后台编号为2的挂起进程恢复运行

# jobs //查看后台进程信息

[1]- 运行中 sleep 1000 &

[2]+ 运行中 sleep 800 &

# fg 2 //将后台编号为2的进程恢复前台运行

sleep 800

==============================================================

* + 1. 进程处决

· 干掉进程的不同方法

- Ctrl+c组合键，中断当前命令

- kill [-9] PID...、kill [-9] %后台任务编号

- killall [-9] 进程名...

- pkill 关键字... //模糊查询查杀(包含即死)

· 强制踢出一个用户

killall -9 -u 用户名 //杀死该用户开启的所有进程（用户由登陆变成未登陆）

书写防火墙或SELinux禁止用户登录

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

# sleep 900 &

[3] 5905

# jobs -l

[3]+ 5905 运行中 sleep 900 &

# kill 5905 //根据PID杀死进程

# jobs

[3]+ 已终止 sleep 900

# sleep 1000 &

[1] 5928

# killall -9 sleep //根据进程名,强制杀死进程

[1]+ 已杀死 sleep 1000

===============================================================

* 1. 日志管理
     1. 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"

* + 1. 查看文本日志消息

· 通过分析工具

- tail、tailf、less、grep等文本浏览/检索命令

- awk、sed等格式化过滤工具

· 专用分析工具

- Webmin系统管理套件

- Webalizer、AWStats等日志统计套件

· 路径

/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

· 由系统服务rsyslog统一记录/管理

- 日志消息采集用文本格式

- 主要记录时间发生的时间,主机,进程,内容

* + 1. 用户登陆分析

· users、who、w命令

- 查看已登陆的用户信息，详细度不同

· last、lastb命令

- 查看最近登陆成功/失败的用户信息

* + 1. 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

* + 1. 使用journalctl工具

· 提取由sysremd-journal服务收集的日志

- 主要包括内核/系统日志、服务日志

· 常见用法

- journalctl | grep 关键词

- journalctl -u 服务名 [-p 优先级]

- journalctl -n 消息条数

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

1. Day 04 系统&服务管理进阶 批量装机环境 配置PXE引导 kickstart自动应答

补充:虚拟机命令行管理

# virt-manager //启用图形虚拟系统管理器

# virsh list //列出正在运行的虚拟机

# virsh list --all //列出所有虚拟机

# clone-vm7 //产生一个新的虚拟机

# virsh list --all

# virsh start xxx //开启虚拟机

# virsh list --all

# virsh console xxx //直接控制虚拟机,不需要IP

退出终端管理模式:Ctrl + ]

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

1. 网络批量装机环境
   * 1. 部署DHCP服务器

1)DHCP概述及原理

· Dynamic Host Configuration Protocol

-动态主机配置协议,由IETF(Internet 网络工程师人物小组)组织指定,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工作分配原理(广播进行)

- 一个网络中只能有一个DHCP服务器

- 判断机制:先到先得

· 地址分配的四次会话

- DISCOVERY --> OFFER --> REQUIST --> ACK

客户机广播 路由反馈

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

· 配置dhcpd地址分配服务

虚拟机svr7:

1. 安装dhcp服务包

# yum -y install dhcp

1. 修改配置文件/etc/dhcp/dhcpd.conf

末行模式输入 :r /user/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地址范围

option domain-name-servers 192.168.4.7; #分配的DNS地址

option routers 192.168.4.254; #分配的网关地址

default-lease-time 600; #IP地址默认租期

max-lease-time 7200; #IP地址最大租期

}

1. 重启服务

# systemctl restart dhcpd

* + 1. 网络装机概述

1)网络装机的优势

-规模化:同时装配多台主机

-自动化:装系统,配置等各种服务

-远程实现:不需要光盘,U盘等物理安装介质

2)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

3)PXE组件及过程分析

·需要哪些服务组件

-DHCP服务,分配IP,定位引导程序

-TFTP服务,提供引导程序下载 //TFTP:简单的文件传输协议,端口69

-HTTP服务(或FTP/NFS),提供yum安装源

·简略过程示意图

next-server

-------------->

客户端 DHCP <----------------- TFTP

｜ <-------------- 众多的引导文件 |

｜ IP地址 |

------<-----------------<----------------<-------

rpm包

图vim版

next-server

-------------------->

客户端 DHCP <----------------- TFTP

| <-------------------- 众多的引导文件 |

| IP地址 |

-------<-----------------------<-----------------------------<----------

rpm包

图txt版

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

1.修改配置文件

# 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

filename "pxelinux.0"; #指定网卡引导文件名称

}

2.重启dhcpd服务

# systemctl restart dhcpd

pxelinix.0:网卡引导文件(网络装机说明书)二进制文件

安装一个软件可以自动产生默认叫pxelinux.0

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

* 1. 搭建TFTP服务

简单的文件传输协议 端口69

tftp默认共享数据路径 /var/lib/tftpboot

* + 1. 安装tftp-server包

# yum -y install tftp-server

* + 1. 启动tftp服务

# systemctl restart tftp

# ls /var/lib/tftpboot

* + 1. 部署pxelinux.0文件

# yum provides \*/pxelinux.0 //查询仓库中哪个软件包产生该文件

# yum -y install syslinux-4.05-13.el7.x86\_64

# 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

* + 1. 部署菜单文件

pxelinux ---> /var/lib/tftpbppt/pxelinux.cfg/default

1)创建菜单目录

# mkdir /var/lib/tftpboot/pxelinux.cfg

# ls /var/lib/tftpboot/

pxelinux.0 pxelinux.cfg

2)从光盘拷贝菜单配置文件

# scp /var/ftp/rhel7/isolinux/isolinux.cfg root@192.168.4.7:/var/lib/tftpboot/pxelinux.cfg/default

· 为了修改方便,给管理员加上写权限

# chmod u+w /var/lib/tftpboot/pxelinux.cfg/default

# ll /var/lib/tftpboot/pxelinux.cfg/default

-rw-r--r--. 1 root root 3166 7月 24 14:25 /var/lib/tftpboot/pxelinux.cfg/default

* + 1. 部署引导文件(启动内核)

vesamenu.c32 图形模块,呈现背景图片与颜色等

vmlinuz 启动内核

initrd.img 驱动程序

splash.png 黑色背景图片

# scp /var/ftp/rhel7/isolinux/vesamenu.c32 /var/ftp/rhel7/isolinux/vmlinuz /var/ftp/rhel7/isolinux/initrd.img /var/ftp/rhel7/isolinux/splash.png root@192.168.4.7:/var/lib/tftpboot/

# ls /var/lib/tftpboot/

initrd.img pxelinux.0 pxelinux.cfg splash.png vesamenu.c32 vmlinuz

* + 1. 修改菜单文件内容

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

1 default vesamenu.c32 #默认加载图形模块

2 timeout 600 #默认读秒

11 menu title NSD1806 PXE SERVER #修改标题

61 label linux

62 menu label ^Install RHEL7.4

menu default #读秒结束默认选择

63 kernel vmlinuz

64 append initrd=initrd.img

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

初步测试:

1.新建虚拟机 选择PXE网络引导安装

2.网络类型选择private1

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

* 1. 构建httpd服务,利用web共享众多的rpm包
     1. 安装软件包

# yum -y install httpd

# systemctl restart httpd

* + 1. 建立共享路径

# mkdir /var/www/html/rhel7

# vim /etc/exports //真机通过nfs共享给svr7

/var/ftp/rhel7 192.168.4.0/24(ro)

# mount 192.168.4.254:/var/ftp/rhel7 /var/www/html/rhel7/

# ls /var/www/html/rhel7

* + 1. 测试

# firefox 192.168.4.7/rhel7

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

* 1. 部署无人值守安装,生成应答文件
     1. 图形生成应答文件工具system-config-kickstart

# yum -y install system-config-kickstart

* + 1. 运行工具system-config-kickstart

· 检查"软件包选择"是否可以选择

与本机Yum仓库标识有关: [development]

* + 1. 查看应答文件

# ls /root/ks.cfg

# vim /root/ks.cfg

* + 1. 利用web共享ks应答文件,传递给客户端

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

# ls /var/www/html

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

* 1. 修改菜单文件,指定ks应答文件

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

label linux

menu label ^Install RHEL7.4

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 --> /var/lib/tftpboot/pxelinux.cfg/default

default --> vesamenu.c32\splash.png\vmlinuz\inittrd.img

ks="http://192.168.4.7/rhel7"

1. Day 05 rsync同步操作 inotify实时同步 Cobbler装机平台

环境准备:

·检查Yum是否可用

虚拟机A:

# yum clean all //清空Yum缓存

# yum repolist

虚拟机B:同上

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

* 1. rsync同步操作

1. 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

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

· 虚拟机svr7本机同步

# mkdir /nsd /test

# cp /etc/passwd /nsd/

# touch /nsd/1.txt

# rsync -avz /nsd/ /test/

sending incremental file list

./

1.txt

passwd

sent 995 bytes received 53 bytes 2096.00 bytes/sec

total size is 2201 speedup is 2.10

# touch /nsd/2.txt

# rsync -avz /nsd/ /test/

sending incremental file list

./

2.txt

sent 112 bytes received 34 bytes 292.00 bytes/sec

total size is 2201 speedup is 15.08

# rsync -avz --delete /nsd/ /test/ //同步并删除目标多余文档

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

· rsync+SSH远程同步

与远程的SSH目录保持同步,格式类似scp

- 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

# rsync -avz --delete /opt/ root@192.168.4.207:/opt/

虚拟机B

# ls /opt

虚拟机A

# touch /opt/{1..5}.txt

# rsync -avz --delete /opt/ root@192.168.4.207:/opt/

虚拟机B

# ls /opt

* 1. inotify实时同步

1. 实时远程同步

· 无密码验证

a)生成公钥和私钥

# ssh-keygen //一路回车

/root/.ssh/known\_hosts //记录ssh远程客户端标识文件

·公钥

·私钥

b)传递公钥到指定机器

# ssh-copy-id root@192.168.4.207

c)测试免验证登录

# rsync -avz --delete /opt/ root@192.168.4.207:/opt/

* + 1. 目录内容监控

a)安装inotify-tools控制工具可调用此机制实现监控

· 标准的源码,编译安装

rpm包:yum,rpm -ivh

源码包:

· 安装流程

源码包 --> 开发工具(gcc与make) --> 可执行的程序 --> 运行安装

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

虚拟机svr7:

1.安装gcc与make

# yum -y install gcc make

2.tar进行解包

# tar -xf /root/xxd/tools/inotify-tools-3.13.tar.gz -C /

# ls /

# cd /inotify-tools-3.13

# ls

3.'./configure'配置,指定安装目录/功能模块等选项

检测系统是否安装gcc

--prefix=路径 #指定安装目录

# cd /inotify-tools-3.13

# ./configure

4.make编译,生成可执行的二进制程序文件

# cd /inotify-tools-3.13

# make

5.make install 安装,将编译好的文件复制到安装目录

# make install

# which inotifywait //查看是否具备该程序

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

· 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

====================================================================

* 1. Cobbler装机平台
     1. 环境准备

·安装CentOS 7虚拟机:

1.图形方式进行安装

2.内存2G

3.磁盘大小:至少50G以上

4.网络类型private1

5.分区选择自动分区

6.软件包选择"带GUI的服务器"

7.设置root密码 创建普通用户lisi

8.将CentOS放入光驱设备,搭建本地Yum仓库

9.配置IP地址:192.168.4.168/24

10.配置主机名:Cobbler.tedu.cn

11.设置防火墙默认区域为trusted

12.当前及永久设置SELinux状态为Disabled

13.利用scp将真机的Cobbler.zip包传到虚拟机192.168.4.168的/root/目录下

* + 1. 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midify,move,create,delete,attrib等事件类别

* + 1. 书写Shell脚本

循环:for循环:适合次数固定的事件

while循环:适合死循环的事件

while [条件]

do

循环执行语句

done

# vim /root/rsync.sh

#!/bin/bash

while inotifywait -rqq /opt/

do

rsync -az --delete /opt/ root@192.168.4.207:/opt/

done &

# chmod +x /root/rsync.sh

* + 1. 搭建Cobbler装机平台

Cobbler概述软件,管理dhcp,tftp,web服务

自由地导入镜像与ks应答文件

1)解压Cobbler.zip包

# unzip /root/Cobbler.zip -d /

# cd /Cobbler/

# ls

# unzip /Cobbler/cobbler.zip -d /opt

# ls /opt/cobbler

* + 1. Cobbler简介

· 基本概念

-Cobbler是一款快速的网络系统部署工具

-集中管理所需服务,如DHCP,DNS,TFTP,Web

-Cobbler内部集成了一个镜像版本仓库

-Cobbler内部集成了一个ks应答文件仓库

-Cobbler还提供了包括yum源管理,Web界面管理,API接口,电源管理功能

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

一.安装cobbler主程序,工具包等

# yum -y install /opt/cobbler/\*.rpm

1.安装软件 cobbler cobbler-web dhcp tftp-server pykickstart httpd

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

# vim /etc/cobbler/settings

next\_server: 192.168.4.168 #设置下一个服务器还为本机

server: 192.168.4.168 #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:匹配即停止

1.硬盘启动 2.光驱设备 3.U盘 4.网络引导

1.网络引导 2.光驱设备 3.U盘 4.硬盘启动

3.配置cobbler的dhcp

# vim /etc/cobbler/dhcp.template

末行模式输入,将192.168.1默认网段改为192.168.4网段

:%s/192.168.1/192.168.4/g

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

# cobbler sync

...

\*\*\* TASK COMPLETE \*\*\*

# firefox https://192.168.4.168/cobbler\_web

用户名:cobbler

密码:cobbler

7.Cobbler应用

cobbler import --path=挂载点 --name=导入系统命名(随意起)导入镜像安装

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

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=CentOS7.4-A --kickstart=/var/lib/cobbler/kickstarts/自定义.cfg

[root@cobbler ~]# cobbler profile report

[root@cobbler ~]# cobbler sync #同步配置

1. Day 06 日志管理 systemctl服务管理 PATH变量应用
2. 日志管理
   * 1. 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"

* + 1. 查看文本日志消息

· 通过分析工具

- tail、tailf(实时跟踪日志消息)、less、grep等文本浏览/检索命令

- awk、sed等格式化过滤工具

· 专用分析工具

- Webmin系统管理套件

- Webalizer、AWStats等日志统计套件

· 路径

/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

· 由系统服务rsyslog统一记录/管理

- 日志消息采集用文本格式

- 主要记录时间发生的时间,主机,进程,内容

* + 1. 用户登陆分析

· users、who、w命令

- 查看已登陆的用户信息，详细度不同

· last、lastb命令

- 查看最近登陆成功/失败的用户信息

# users

# who

# w

# last #查看登陆成功的用户信息

# lasttb #查看登录失败的用户信息

* + 1. 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

* + 1. 使用journalctl工具

· 提取由sysremd-journal服务收集的日志

- 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"

# systemctl restart httpd

# journalctl -u httpd

=============================================================

* 1. systemctl服务管理

· 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

· 服务的管理

systemvtl start 服务名 #启动服务

systemvtl stop 服务名 #停止服务

systemvtl restart 服务名 #重启服务

systemvtl ststus 服务名 #查看服务当前状态

systemvtl enable 服务名 #开机自启

systemvtl disable 服务名 #开机不自启

systemvtl is-enabled 服务名 #查看服务自启状态

· 运行模式的管理(运行级别)

RHEL5,RHEL6:

0：关机

1：单用户模式（破解密码、修复系统）

2：字符模式（不支持网络）

3：字符模式（支持网络）

4：无定义

5：图形模式

6：重启

0

开机启动 v

6

- 切换运行级别的命令 init

# init 3 切换到字符界面

# init 5 切换到图形界面

RHEL7: 运行模式

systemctl -t --all

multi-user.target :字符模式

graphical.target:图形模式

· 修改当前运行模式(重启失效)

systemctl isolate multi-user.target

systemctl isolate grphical.target

· 修改默认运行模式(重启有效)

systemctl set-default graphical.target

systemctl set-default multi-user.target

· unit配置单元

· 不同的unit决定了一组相关的启动任务

- service:后台独立服务

- socket:套接字,类似于xinetd管理的临时服务

- target:一套配置单元的组合,类似于传统"运行级别"

- device:对应udev规则标记的某个设备

- mount,automount:挂载点,触发挂载点

=========================================================

* 1. Cobbler装机平台

1. 环境准备

·安装CentOS 7虚拟机:

1.图形方式进行安装

2.内存2G

3.磁盘大小:至少50G以上

4.网络类型private1

5.分区选择自动分区

6.软件包选择"带GUI的服务器"

7.设置root密码 创建普通用户lisi

8.将CentOS放入光驱设备,搭建本地Yum仓库

9.配置IP地址:192.168.4.168/24

10.配置主机名:Cobbler.tedu.cn

11.设置防火墙默认区域为trusted

12.当前及永久设置SELinux状态为Disabled

13.利用scp将真机的Cobbler.zip包传到虚拟机192.168.4.168的/root/目录下

1. 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midify,move,create,delete,attrib等事件类别

1. 书写Shell脚本

循环:for循环:适合次数固定的事件

while循环:适合死循环的事件

while [条件]

do

循环执行语句

done

# vim /root/rsync.sh

#!/bin/bash

while inotifywait -rqq /opt/

do

rsync -az --delete /opt/ root@192.168.4.207:/opt/

done &

# chmod +x /root/rsync.sh

1. 搭建Cobbler装机平台

Cobbler概述软件,管理dhcp,tftp,web服务

自由地导入镜像与ks应答文件

1)解压Cobbler.zip包

# unzip /root/Cobbler.zip -d /

# cd /Cobbler/

# ls

# unzip /Cobbler/cobbler.zip -d /opt

# ls /opt/cobbler

=========================================================

* 1. PATH变量应用:提供命令搜寻路径

系统环境变量:由系统定义完成且赋值完成

# echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

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

[root@svr7 ~]# vim /usr/local/bin/hello.sh

[root@svr7 ~]# chmod +x /usr/local/bin/hello.sh

[root@svr7 ~]# hello.sh

hello world!

1. NETWORK
2. Day 01 计算机网络概述 网络设备及拓扑 OSI参考模型 交换机命令基础
3. 什么是计算机网络
4. 什么是计算机网络

- 硬件方面:通过线缆将网络设备和计算机连接起来

- 软件方面:操作系统,应用软件,应用程序通过通信线路相连

- 实现资源共享,信息传递

* + 1. 计算机网络的功能

· 数据通信

· 资源共享

· 增加可靠性

· 提高系统处理能力

* + 1. 计算机网络发展阶段

· 60年代

-分组交换

· 70-80年代

-TCP/IP

· 90年代后

-Web技术

* + 1. 网络标准

· 标准化组织

-ISO(国际标准化组织)

-ANSI(美国国家标准化局)

-ITU-T(国际电信联盟-电信标准部)

-IEEE(电气和电子工程师学会)

* + 1. WAN与LAN

· 广域网(Wide-Area Network)

-范围:几十到几千千米

-作用:用于连接远距离的计算机网络

-典型应用:Internet

· 局域网(Local-Area Network)

-范围:1km左右

-作用:用于连接较短距离内的计算机

-典型应用:企业网,校园网

* 1. 网络设备及拓扑
     1. 网络设备生产厂商

· 网络设备生产厂商

- Cisco(思科)

- 华为

* + 1. 网络拓扑结构

· 线缆连接计算机和网络设备的布局

- 点对点

- 星型及扩展的星型

- 网状

1)点对点拓扑结构

-两台设备之间有一条单独的连接

-专用的广域网中电路连接的两台路由器

2)星型拓扑

-优点

易于实现

易于网络扩展

易于故障排查

-缺点

中心节点压力大

组网成本较高

3)网状拓扑

-一个节点与其他多个节点相连

-提供冗余性和容错性

-可靠性高

-组网成本高

* 1. OSI参考模型
     1. 网络分层

-数据以电子信号的形式穿越介质到达正确的计算机,然后转换成最初的形式,以便接收者能够阅读

-为了降低网络设计的复杂性,将协议进行了分层设计

* + 1. OSI的七层框架

应用层 网络服务与最终用户的一个借口

表示层 数据的表示,安全,压缩

会话层 建立,管理,中止会话

传输层 定义传输数据的协议端口号,以及流控和差错校验

网络层 进行逻辑地址寻址,实现不同网络之间的路径选择

数据链路层 建立逻辑连接,进行硬件地址寻址,差错校验等功能

物理层 建立,维护,断开物理连接

* + 1. TCP/IP五层模型

应用层 HTTP FTP TFTP SMTP SNMP DNS

传输层 TCP UDP

网络层 ICMP IGMP IP ARP(IP->MAC的解析) RARP(反解析)

数据链路层 由底层网络定义的协议

物理层

* + 1. 协议

· 什么是协议

- 为了使数据可以在网络上从源传递到目的地,网络上所有设备需要"讲"相同的"语言"

- 描述网络通信中语言规范的规则称为协议

* + 1. 协议数据单元

物理层 网卡 比特流 电信号

数据链路层 交换机 数据帧 MAC头部,IP头部,TCP头部,上层数据

网络层 路由器 数据包 IP头部,TCP头部,上层数据

传输层 防火墙 数据段 TCP头部,上层数据

应用层 计算机 上层数据

* 1. 物理层
     1. 以太网接口

· RJ - 45

-RJ使描述公用电信网络的接口,常用的有RJ-11和RJ-45(水晶头)

· 光纤接口

-用以稳定地但并不是永久地连接两根或多根光纤的无源组件

FC圆形带螺纹光纤接头

ST卡接式圆形光纤接头

SC方形光纤接头

LC窄体方形光纤接头

MT-RJ收发一体的方形光纤接头

· 双绞线

-双绞线TP是目前使用最广,价格相对便宜的一种传输介质

-由两根绝缘铜导线相互缠绕组成,以减少对邻近线的电气干扰

-由若干对双绞线构成的电缆被成为双绞线电缆

· 双绞线的标准

-非屏蔽双绞线UTP和屏蔽双绞线STP

-速率

cat5 100Mbps

cat5e 100Mbps

cat6 1000Mbps(1Gbps)

cat7 10000Mbps(10Gbps)

-网线长度一般不能超过150m

· 线缆的连接:

-T568A:白绿 绿 白橙 蓝 白蓝 橙 白棕 棕

-T568B:白橙 橙 白绿 蓝 白蓝 绿 白棕 棕

-直通线和交叉线

· 同类型设备用交叉,不同类型用直通

· 交换机之间用直通,路由器当主机处理

-全反线:连接设备console口

* + 1. 网络接口卡(网卡)

- 连接计算机和网络硬件

- 有一个唯一的网络节点地址

- 按照速率可分为10/100M,100/1000M自适应网卡

- 按照扩展类型可分为USB网卡,PCI网卡

- 按照提供的线缆接口类型可分为RJ-45接口网卡,光纤网卡等

· 中继器

-放大信号

-延长网络传输距离

* 1. Packet Tracer软件
  2. 交换机命令行模式
     1. 用户模式

Switch>用户模式

* + 1. 特权模式(一般用于查看配置信息)

Switch>enable

Switch#特权模式

* + 1. 全局配置模式(所做的配置对整个设备生效)

Switch#configure terminal

Switch(config)#全局配置模式

* + 1. 接口模式

Switch(config)#interface fastEhernet 0/1

Switch(config-if)#接口模式

* + 1. 配置主机名

>enable

#conf t

()#hostname S1

()#exit

#show running-config #查看配置信息

* + 1. 配置enable明文口令

全局配置模式：enable password 123

* + 1. 保存交换机的配置

#copy running-config startup-config

或

#write

* + 1. 恢复设备出厂默认值

特权：erase startup-config

* + 1. 重启：reload
  1. 设备配置的准备工作
     1. 空闲一段时间后，重回初始界面的问题

switch(config)#line con 0

switch(config-line)#exec-timeout 0 0

* + 1. 禁用DNS查询

switch(config)#no ip domain-lookup

* + 1. 配置输出日志同步

Switch(config)#line console 0

Switch(config-line)#logging synchronous

1. Day 02 数据链路层
2. 以太网
   * 1. 以太网MAC地址(48位,用16进制表示)

- 用来识别一个以太网上的某个单独的设备或一组设备

24比特供应商标识 24比特供应商对网卡的唯一编号

第八位对于目的地址:

0-物理地址(单播地址)

1-逻辑地址(组播地址)

* + 1. 单播 1对1

组播 1对多

广播 1对所有

* + 1. 帧格式

目的地址 6字节

源地址 6字节

类型/长度 2字节

数据 46-1500字节

帧校验序列 4字节

* 1. 以太交换机
     1. 什么是交换机

· 交换机是用来连接局域网的主要设备

-交换机能够根据以太网帧中目标地址智能转发数据,工作在数据链路层

* + 1. 交换机的转发原理

- 初始状态

- MAC地址学习

- 广播未知数据帧

- 接受方回应

·学习 ·广播 ·转发 ·更新

* + 1. 查看MAC地址表

#show mac-address-table

* 1. 广播域
     1. VLAN

·虚拟局域网是物理设备上连接的不受物理位置限制的用户的一个逻辑组

·为什么引入VLAN

-交换机的所有端口默认属于同一个广播域

-随着接入设备的增多,网络中广播增多,降低了网络效率

-为了分割广播域,引入了VLAN

·VLAN的作用

-广播控制

-安全性

-带宽利用

-延迟

·基于端口划分的静态VLAN

-静态VLAN的配置

-创建VLAN

-将端口加入到相应VLAN

-验证

·在全局模式下创建VLAN

()#vlan vlan-id

()#vlan vlan-name

·查看VLAN配置

#show vlan brief

#show vlan id $id

·删除已创建的VLAN

()#no vlan 5

·将端口加入VLAN

()#interface f0/2

()#switchport access vlan 2

·批量修改接口属性

()#interface range f0/1-2

()#switchport access vlan 2

* 1. Trunk
     1. 用于多广播域多交换机之间作为中继链路

·配置Trunk

()#interface 0/24

()#switchport mod trunk

Administrative Mode 动态自动(默认)

Operation Mode 正在使用的模式

-这样的链路中两台交换机之间的广播域vlan要对应

·ISL帧格式:前ISL头26字节,后CRC4字节

·ISL和802.1Q标记之间的异同

-相同点

·都是显式标记,即帧被显式标记了VLAN的信息

-不同点

·IEEE 802.1Q是公有的标记方式,ISL是Cisco私有的

·ISL采用外部标记的方法,802.1Q采用内部标记的方法

·ISL标记的长度为30字节,802.1Q标记长度为4字节

* + 1. 配置以太通道

1)概述:

·也称为以太端口捆绑,端口聚集或以太链路聚集

·以太通道为交换机提供了端口捆绑技术,允许两个交换机之间通过两个或多个端口并行连接,同时传输数据,以提供更高的带宽

2)配置以太网通道

()#interface range fastEthernet0/10-11

()#channel-group 1 mode on

Creating a port-channel interface Port-channel 1

3)查看以太网通道的配置

#show etherchannel summary

1. Day 03 网络层
   1. 网络层的功能

· 定义了基于IP协议的逻辑地址

· 连接不同的媒介类型

· 选择数据通过网络的最佳路径

* 1. 路由概述

· 将数据包从一个网络发送到另一个网络

- 需要依靠路由器完成

- 路由器只关心网络的状态,决定最佳路径

* 1. 路由器工作:

-识别数据包的目标IP地址

-识别数据包的源IP地址(用于策略路由)

-在路由表中发现可能的路径

-在路由表中选择到达目标最好的路径

-维护和检查路由信息

· 根据路由表选择最佳路径

-每个路由器都维护一张路由表,这是路由器转发数据包的关键

-每条路由表记录指明了:到达某个子网或主机应从路由器的那个物理端口发送,通过此端口可达到该路径的下一个路由器的地址

* 1. 如何获得路由表

· 静态,缺省路由

-由管理员在路由器上手动指定

-适合分支机构,家居办公等小型网络

· 动态路由

-根据网络拓扑或流量变化,由路由器通过路由协议自动设置

-适合ISP服务商,广域网,园区网等大型网络

* 1. 静态路由

· 主要特点

-由管理员手动配置,为单向条目

-通信双方的边缘路由器都需要指定,否则会导致数据包有去无回

· 配置静态路由

()#in g0/0

//设置接口ip

()#ip address 192.168.1.254 255.255.255.0

//启用接口

()#no shutdown

· 查看路由表

#show ip route

C直连路由

S静态路由

S\*默认路由

· 删除接口ip

()#no ip address

· 配置静态路由

()#ip route 目标网络ID 子网掩码 下一跳

* 1. 缺省路由

· 什么是缺省路由

-缺省路由是一种特殊的静态路由,对于末梢网络的主机来说,也被称为"默认网关",一般缺省路由也只应用在末梢网络,中继网络使用缺省路由会导致丢包

-缺省路由的.目标网络为0.0.0.0 0.0.0.0 ,可匹配任何目标地址

-只有当从路由表中找不到任何明确匹配的路由条目时,才会使用缺省路由

* 1. 三层交换机

·概述:

· 使用三层交换技术实现VLAN间通信

· 三层交换=二层交换+三层转发

·虚接口

·在三层交换机上配置的VLAN接口为虚接口

·使用SVI(交换虚拟端口)实现VLAN间路由

· 启用三层交换机的路由功能

()#ip routing

· 配置虚接口的IP

()#interface vlan $id

()#ip address $ip $netmask

()#no shutdown

· 配置路由接口

()#no switchport

· 在三层交换机上配置Trunk并指定接口封装为802.1Q

()#interface f0/24

()#switchport trunk encapsulation dot1q

()#switchport mode trunk

* 1. 动态路由

· 动态路由

-基于某种路由协议实现

· 动态路由的特点

-减少了管理任务

-占用了网络带宽

·OSPF

-邻居列表

-链路状态数据库

-路由表

·OSPF区域

-为了适应大型网络,OSPF在网络内部划分多个区域

-每个OSPF路由器只维护所在区域的完整链路状态信息

·区域ID

-区域ID可以表示成一个IP或十进制数字

·配置OSPF

启动OSPF路由进程

()#router ospf $id

指定OSPF协议运行的网络地址和所在区域

()#network $ip 反掩码 area $areaId

重启OSPF

#clear ip ospf process

1. Day 04 传输层
   1. 传输层作用

· 网络层提供点到点的连接

· 传输层提供端到端的连接

* 1. 传输层的协议

· TCP

-传输控制协议

-可靠的,面向连接的协议

-传输效率低

· UDP

-用户数据报协议

-不可靠的,无连接的服务

-传输效率高

* 1. TCP协议
     1. TCP的封装格式

· SYN 请求标记

· ACK 确定标记

* + 1. TCP连接过程:三次握手

客户端:SYN seq=100,ctl=SYN

服务端:SYN,ACK seq=300,ack=101,ctl=SYN,ACK

客户端:ACK seq=101,ack=301,ctl=ACK

* + 1. TCP的四次断开

客户端:发送 FIN,请求断开连接(FIN=1,ACK=1)

服务端:发送 ACK(ACK=1)

服务端:发送 FIN,请求断开连接(FIN=1,ACK=1)

客户端:发送ACK(ACK=1)

* + 1. TCP的应用

端口 协议 说　　明

21 FTP FTP服务器所开放的控制端口

23 TELNET 用于远程登录，可以远程控制管理目标计算机

25 SMTP SMTP服务器开放的端口，用于发送邮件

80 HTTP 超文本传输协议

53 DNS 域名服务，当用户输入网站的名称后，由DNS负责将它解析成IP地址，这个过程中用到的端口号是53

* 1. UDP协议
     1. UDP的流控和差错控制

· UDP缺乏可靠机制

· UDP只有校验和来提供差错控制

-需要上层协议来提供差错控制:如TFTP协议

* + 1. UDP的应用

端口 协议 说明

69 TFTP 简单文件传输协议

123 NTP 网络时间协议

53 DNS 域名服务

* 1. 访问控制列表概述
     1. 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

* + 1. 访问控制列表的类型

· 标准访问控制列表

- 基于源IP地址过滤数据包

- 标准访问控制列表的表号是1~99

· 扩展访问控制列表

- 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

* + 1. 标准访问控制列表的配置

· 创建ACL

()#access-list $number { permit | deny } source [ source-wildcard ]

· 实例

//允许192.168.1.0/24网段的流量通过

Router(config)# access-list 1 permit 192.168.1.0 0.0.0.255

//允许192.168.2.2主机的流量通过

Router(config)# access-list 1 permit 192.168.2.2 0.0.0.0

· 隐含的拒绝语句(拒绝所有人)

Router(config)# access-list 1 deny 0.0.0.0 255.255.255.255

· 关键字

-host

-any

· 将ACL应用于接口

Router(config-if)# ip access-group $number {in|out}

· 删除ACL

Router(config-if)# no ip access-group $number {in |out}

* + 1. 扩展访问控制列表的配置

· 创建ACL

Router(config)# access-list $number { permit | deny } protocol { source source-wildcard destination destination-wildcard } [ operator operan ]

* 1. NAT概述
     1. NAT作用

· NAT

-网络地址转换

· 作用

-通过将内部网络的私有IP地址翻译成全球唯一的公网IP地址,使内部网络可以连接到外部网络上

· NAT的优点

-节省公有合法IP地址

-处理地址重叠

-安全性

· NAT的缺点

-延迟增大

-配置和维护的复杂性

* + 1. NAT实现方式

· 静态转换

· 端口多路复用

* + 1. 静态转换

· IP地址的对应关系是一对一的,而且是不变的,借助静态转换,能实现外部网络对内部网络中某些特设定服务器的转换

· 静态NAT配置步骤

-接口IP地址配置

-决定需要转换的主机地址

-决定采用什么公有地址

-在内部和外部端口上启用NAT

()#ip nat inside source static $localIP $globalIP

· 将内网地址192.168.1.1静态转换为合法的外部地址100.0.0.2以便访问外网

· 设置外部端口的IP地址

Router(config)#interface g0/1

Router(config-if)#ip address 100.0.0.1 255.0.0.0

Router(config-if)#no shut

· 设置内部端口的IP地址

Router(config)#interface g0/0

Router(config-if)#ip address 192.168.1.254 255.255.255.0

Router(config-if)#no shut

· 建立静态地址转换

Router(config)#ip nat inside source static 192.168.1.1 100.0.0.2

· 在内部和外部端口上启用NAT

Router(config)#interface g0/1

Router(config-if)#ip nat outside

Router(config)#interface g0/0

* + 1. NAT端口映射

· 建立NAT端口映射

· 配置实例(只发布服务器的80端口)

Router(config)#ip nat inside source static tcp 192.168.1.6 80 61.159.62.133 80

Router(config-if)#ip nat inside

* + 1. 端口多路复用(PAT)

· ACL定义内部IP地址

Router(config)#access-list 1 permit 192.168.1.0 0.0.0.255

· 设置复用动态IP地址转换

//配置端口多路复用使企业内网192.168.1.0/24复用g0/1端口的ip,实现外部网络的访问

Router(config)#ip nat inside source list 1 interface g 0/1 overload

· PAT只适用于内网访问外网,无法架设服务

1. Day 05 应用层
2. STP生成树算法
   * 1. 广播风暴的产生

• 交换机工作原理

– 根据MAC地址表转发数据帧,如果地址未知,则广播

– 如果交换机接收到广播帧也会向所有端口发送

• 当网络中存在物理环路,会产生广播风暴

* + 1. STP概述

• STP简介

– STP - Spanning Tree Protocol(生成树协议)

逻辑上断开环路,防止广播风暴的产生

当线路故障,阻塞接口被激活,恢复通信,起备份线路的作用

* + 1. 选择根网桥

• 网桥ID(BID)

– 网桥ID是唯一的,交换机之间选择BID值最小的交换机作为网络中的根网桥

网桥优先级 2字节

网桥的MAC地址 6字节 取值范围:0 ~ 65535

缺省值:32768

* 1. STP配置
     1. PVST+的配置命令

• 启用生成树命令

Switch(config)#spanning-tree vlan $vlanList

• 指定根网桥

Switch(config)#spanning-tree vlan $vlanList priority Bridge-priority

Switch(config)#spanning-tree vlan $vlanList root { primary | secondary }

• 查看生成树的配置

Switch#show spanning-tree

• 查看某个VLAN的生成树详细信息

Switch#show spanning-tree vlan vlan-id

N.HSRP概述

n.HSRP的相关概念

1.热备份路由选择协议

-HSRP(Hot Standby Routing Protocol)

-是Cisco私有协议(VRRP公有协议了解一下)

* + 1. HSRP组成员

·两台物理路由器(一台活跃一台备份)

-活跃路由器

-备份路由器

-虚拟路由器

-其他路由器

* + 1. HSRP原理

一台活跃,一台热备,虚拟桥接,来回切换

* + 1. HSRP配置

• 配置为HSRP的成员

Switch(config-if)#standby group-number ip $vlanIP

• 配置HSRP的优先级

Switch(config-if)#standby group-number priority priority-value

• 查看HSRP摘要信息

Switch(config-if)#show standby brief

• MS1配置

MS1(config)#interface vlan 1

MS1(config-if)#ip address 192.168.1.252 255.255.255.0

MS1(config-if)#standby 1 ip 192.168.1.254

MS1(config-if)#standby 1 priority 105

• HSRP端口跟踪

– 跟踪端口不可用时,HSRP优先级降低

– 活跃路由器可以根据线路情况自动调整

• HSRP占先权

– 优先级高的路由器重新获得转发权,恢复成为活跃路由器

– HSRP占先权配置

()#standby group-number perrmpt

1. Day06综合项目完整实验步骤
   1. 二层交换机

分别创建VLAN10、20、30、40

sw1将f0/5接口加入vlan10

Switch(config)#interface fastEthernet 0/5

Switch(config-if)#switchport access vlan 10

sw2将f0/5接口加入vlan20

Switch(config)#interface fastEthernet 0/5

Switch(config-if)#switchport access vlan 20

sw3将f0/5接口加入vlan30

Switch(config)#interface fastEthernet 0/5

Switch(config-if)#switchport access vlan 30

sw4将f0/5接口加入vlan40

Switch(config)#interface fastEthernet 0/5

Switch(config-if)#switchport access vlan 40

每台设备捆绑以太通道，将f0/1与f0/2捆绑为通道1，f0/3与f0/4捆绑为通道2

Switch(config)#interface range f0/1-2

Switch(config-if-range)#channel-group 1 mode on

Switch(config)#interface range f0/3-4

Switch(config-if-range)#channel-group 1 mode on

查看以太通道汇总信息

Switch#show etherchannel summary

依次进入所有二层交换机的以太通道接口，配置中继链路

Switch(config)#interface port-channel 1

Switch(config-if)#switchport mode trunk

Switch(config)#interface port-channel 2

Switch(config-if)#switchport mode trunk

==========================================================

* 1. 三层交换机

每台设备分别创建VLAN10、20、30、40

1-2口捆绑为通道1

3-4口捆绑为通道2

5-6口捆绑为通道3

7-8口捆绑为通道4

9-10口捆绑为通道5

依次进入三层交换机的4个通道接口，配置中继链路（两台三层交换机配置相同）

Switch(config)#interface port-channel 1

Switch(config-if)# switchport trunk encapsulation dot1q

Switch(config-if)#switchport mode trunk

Switch(config)#interface port-channel 2

Switch(config-if)# switchport trunk encapsulation dot1q

Switch(config-if)#switchport mode trunk

Switch(config)#interface port-channel 3

Switch(config-if)# switchport trunk encapsulation dot1q

Switch(config-if)#switchport mode trunk

Switch(config)#interface port-channel 4

Switch(config-if)# switchport trunk encapsulation dot1q

Switch(config-if)#switchport mode trunk

Switch(config)#interface port-channel 5

Switch(config-if)# switchport trunk encapsulation dot1q

Switch(config-if)#switchport mode trunk

配置三层交换机vlan10、20、30、40的ip地址

Switch(config)#interface vlan 10

Switch(config-if)#ip address 192.168.10.252 255.255.255.0

Switch(config)#interface vlan 20

Switch(config-if)#ip address 192.168.20.252 255.255.255.0

Switch(config)#interface vlan 30

Switch(config-if)#ip address 192.168.30.252 255.255.255.0

Switch(config)#interface vlan 40

Switch(config-if)#ip address 192.168.40.252 255.255.255.0

注意：另外一台三层交换机配置的ip地址是253

Switch(config)#interface vlan 10

Switch(config-if)#ip address 192.168.10.253 255.255.255.0

Switch(config)#interface vlan 20

Switch(config-if)#ip address 192.168.20.253 255.255.255.0

Switch(config)#interface vlan 30

Switch(config-if)#ip address 192.168.30.253 255.255.255.0

Switch(config)#interface vlan 40

Switch(config-if)#ip address 192.168.40.253 255.255.255.0

===========================================================

* 1. 配置生成树协议，产生负载均衡效果。

MS1配置PVST+ 使其成为vlan10、20的主根 vlan30、40的次根

Switch(config)#spanning-tree vlan 10 root primary

Switch(config)#spanning-tree vlan 20 root primary

Switch(config)#spanning-tree vlan 30 root secondary

Switch(config)#spanning-tree vlan 40 root secondary

MS2配置PVST+ 使其成为vlan30、40的主根 vlan10、20的次根

Switch(config)#spanning-tree vlan 30 root primary

Switch(config)#spanning-tree vlan 40 root primary

Switch(config)#spanning-tree vlan 10 root secondary

Switch(config)#spanning-tree vlan 20 root secondary

配置热备份路由协议，完善负载均衡效果。

MS1配置HSRP 使其成为vlan10、20的活跃路由器 vlan30、40的备份路由器

Switch(config)#interface vlan 10

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)#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)#standby 30 ip 192.168.30.254

Switch(config)#interface vlan 40

Switch(config-if)#standby 40 ip 192.168.40.254

查看热备份状态

Switch#show standby brief

MS2配置HSRP 使其成为vlan30、40的活跃路由器 vlan10、20的备份路由器

Switch(config)#interface vlan 30

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 40

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 10

Switch(config-if)#standby 10 ip 192.168.10.254

Switch(config)#interface vlan 20

Switch(config-if)#standby 20 ip 192.168.20.254

开启两台三层交换机的路由功能，并设置每个服务器所在vlan的网关

Switch(config)#ip routing

然后测试目前网络是否可以达成全网互通。

============================================================

按图为路由器与三层交换机相连的接口配置ip

配置动态路由协议，使所有内网互通。

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

查看所有三层设备路由表，应该是统一状态

show ip route

配置r1与r2的nat功能，使内网服务器40.1映射到外网100.0.0.3，并在接口中开启

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

在r1与r2中配置默认路由之后，使用ospf宣告自己是默认信息源（表示自己有通往外网的默认路由）

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

三层交换机如果看不到从路由器学习来的O\*默认路由就去检查路由器G0/2地址是否配置？

验证从外网可以访问内网的web服务。

1. SHELL
2. Day 01 Shell概述 编写及执行脚本 Shell变量 总结和答疑
3. Shell环境及特性
   * 1. 什么是shell

· 在Linux内核与用户之间的解释器程序

-通常指/bin/bash

-负责向内核翻译及传达用户/程序指令

-相当于操作系统的"外壳"

·shell:概念的统称

·bash:具体的形式

# cat /etc/shells

/bin/sh

/bin/bash

/sbin/nologin

/usr/bin/sh

/usr/bin/bash

/usr/sbin/nologin

/bin/tcsh

/bin/csh

# yum -y install ksh

# cat /etc/shells

/bin/sh

/bin/bash

/sbin/nologin

/usr/bin/sh

/usr/bin/bash

/usr/sbin/nologin

/bin/tcsh

/bin/csh

/bin/ksh

# ksh

·切换到ksh解释器, 按exit退出

* + 1. Shell的使用方式 一阶段回顾

· 交互式 --命令行

-人工干预,智能化程度高

-逐条解释执行,效率低

· 非交互式 --脚本

-需要提前设计,智能化难度大

-批量执行,效率高

-方便在后台静默运行

· 计划任务

# at 21:00

at >ls

at >cd /

at >ls

· 快捷键:

ctrl+s:挂起,冻结终端

ctrl+q:解除挂起/冻结终端

ctrl+d:结束输入

· 历史记录文件配置:

# vim /etc/profile

...

HISTSIZE=1000

history | wc -l

1000

1031 grep 'zhangsan' /etc/passwd

//两种调用方法

!grep

!1031

· 清除历史命令

# history -c //清空自己的历史命令

> ~/.bash\_history //清空记录文件

· 别名

1)查看别名

# alias

alias cp='cp -i'

alias egrep='egrep --color=auto'

alias fgrep='fgrep --color=auto'

alias grep='grep --color=auto'

alias l.='ls -d .\* --color=auto'

alias ll='ls -l --color=auto'

alias ls='ls --color=auto'

alias mv='mv -i'

alias rm='rm -i'

alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'

tip:别名设置一般存放在用户的.bashrc内

2)取消别名

# unalias

· 标准输入/输出/错误输出

标准输入(stdin),描述号为0;

标准输出(stdout),描述号为1;

保准错误(stderr),描述号为2;

1)重定向标准输出

- 使用'>'将命令执行的正常输出重定向到文件

- '>' 先清空再覆盖

- '>>' 追加重定向

2)重定向标准错误

- 使用'2>'将错误信息重定向到文件

- 使用'2>>'将错误信息追加重定向到文件

3)全部重定向

- 使用'&>'将所有信息重定向到文件

4)分别重定向

//将正确的信息重定向到a.txt,错误信息重定向到b.txt

# ls a.txt nb.txt >a.txt 2>b.txt

# cat a.txt b.txt

a.txt

ls: 无法访问nb.txt: 没有那个文件或目录

· 编写一个脚本,建立user1-user200

for i in {1..200}

do

useradd user$i 2>> a.log

echo "12345" | passwd --stdin user$i > /dev/null

done

* + 1. Shell脚本格式

· 一个规范的Shell脚本构成包括：

- 脚本声明（需要的解释器、作者信息等）

- 注释信息（步骤、思路、用途、变量含义等）

- 可执行语句（操作代码）

* + 1. Shell脚本运行方式

· 给权限 -> 绝对路径

· bash/sh 脚本名 - - 开子进程

· source 脚本名 - - 不开子进程

example:

# bash sleep.sh

# pstree | grep bash

|-sshd-+-sshd---bash---bash---sleep

| `-sshd---bash-+-grep

# source sleep.sh

# pstree | grep bash

|-sshd-+-sshd---bash---sleep

| `-sshd---bash-+-grep

# history | wc -l

11

# vim /etc/profile

HISTSIZE=10

# bash /etc/profile

# history | wc -l

14

# source /etc/profile

# history | wc -l

10

* + 1. 变量

1)建立变量

test=11

2)取消变量

unset test

3)变量与常量的区分

echo ${test}RMB

4)系统环境变量

大写

env:查看所有环境变量

5)$0 - 脚本自身

$1-n - 脚本第n个参数

$\* - 脚本所有参数

$$ - 脚本pid

$? - 判断脚本(命令)是否执行成功

6)符号运用

引号: 将内容划分为一个整体参数

- 单引号:内部符号全部判定为字符串

- 双引号:内部特殊符号照常执行,其他判定为字符串

双撇: 执行内部语句

7)read用法

# read str 由用户输入,赋值给变量str

//通过[-p]选项给出提升

# read -p "请输入一个整数:"i

8)stty终端显示控制

· 将回显功能关闭(stty -echo)

· 将回显功能恢复(stty echo)

9)若希望定义的变量能被子进程使用，可以使用export命令将其发布为全局变量。使用export发布时，只需指定变量名（可以有多个）即可，也可以通过export命令直接设置新的全局变量

# export yy //发布已定义的变量

# export XX="1234" //发布新变量

1. Day 02 Shell中的数值运算 条件测试操作 使用if选择结构
2. Shell中的数值运算
   * 1. 整数运算工具

1)使用expr命令

· 乘法运算应用'\\*'转义,避免被认做通配符

[root@svr5 ~]# X=1234 //定义变量X

[root@svr5 ~]# expr $X + 78 //加法

1312

[root@svr5 ~]# expr $X - 78 //减法

1156

[root@svr5 ~]# expr $X \\* 78 //乘法，操作符应添加\转义

96252

[root@svr5 ~]# expr $X / 78 //除法，仅保留整除结果

15

[root@svr5 ~]# expr $X % 78 //求模

64

2)使用$[ ]或$(())表达式

[root@svr5 ~]# X=1234

[root@svr5 ~]# echo $[X+78]

1312

[root@svr5 ~]# echo $[X-78]

1156

[root@svr5 ~]# echo $[X\*78]

96252

[root@svr5 ~]# echo $[X/78]

15

[root@svr5 ~]# echo $[X%78]

64

3)使用let命令

expr或$[ ]、$(())方式只进行运算，并不会改变变量的值；而let命令可以直接对变量值做运算再保存新的值。因此变量X=1234，在执行let运算后的值会变更；另外，let运算操作并不显示结果，但是可以结合echo命令来查看：

[root@svr5 ~]# X=1234

[root@svr5 ~]# let y=X+22

[root@svr5 ~]# echo $y

1256

[root@svr5 ~]# let X++; echo $X # X++(X=X+1)

[root@svr5 ~]# let X--; echo $X # X--(X=X-1)

[root@svr5 ~]# let X+=78 ; echo $X # X+=78(X=X+78)

[root@svr5 ~]# let X-=78 ; echo $X # X-=78(X=X-78)

[root@svr5 ~]# let X\*=78 ; echo $X # X\*=78(X=X\*78)

[root@svr5 ~]# let X/=78 ; echo $X # X/=78(X=X/78)

[root@svr5 ~]# let X%=78 ; echo $X # X%=78(X=X%78)

4)bc交互式运算

[root@svr5 ~]# bc

12.34+56.78 //加法

69.12

12.34-56.78 //减法

-44.44

12.34\*56.78 //乘法

700.66

12.34/56.78 //除法

0

quit //退出交互计算器

5)bc非交互运算

[root@svr5 ~]# echo 'scale=4;12.34+5.678' | bc

18.018

[root@svr5 ~]# echo 'scale=4;12.34\*5.678' | bc

70.0665

[root@svr5 ~]# echo 'scale=4;12.34/5.678' | bc

2.1733

* + 1. 条件测试

· 子串判断 == != -z !-z

· 数字判断

-eq equal

-ne not equal

-gt greater than

-ge greater or equal

-lt less than

-le less or equal

· 文件或目录判断

-e exists 是否存在

-f file 是否为文件且存在

-d directory 是否为目录且存在

-r read 是否可读

-w write 是否可写

-x execute 是否可执行

1)'==' 比较两个字符串是否相同

2）!= 比较两个字符串是否不相同

3）一行执行多条命令的情况

# A && B //仅当A命令执行成功，才执行B命令

# A || B //仅当A命令执行失败，才执行B命令

# A ; B //执行A命令后执行B命令，两者没有逻辑关系

# A && B || C //思考？

4) -z 检查变量的值是否未设置（空值）

[root@svr5 ~]# var1="nb" ; var2=""

[root@svr5 ~]# [ -z "$var1" ] && echo "空值" || echo "非空值"

非空值

[root@svr5 ~]# [ -z $var2 ] && echo "空值" || echo "非空值"

空值 //变量var2已设置，但无任何值，视为空

[root@svr5 ~]# [ ! -z $var1 ] //测试var1是否为非空

还有一个-n可以测试变量是否不为空（相当于! -z）。

1. Day 03 循环结构 case语句 、 函数及中断控制
2. for循环
   * 1. 不能代入变量的格式

for i in {1..100}

ex:

for i in {1..10}

do

echo "$i"

done

* + 1. 可以代入变量的格式

for i in `seq n`

for i in `ls \*.txt`

ex:

for i in `cat /root/user.txt

do

useradd $i

echo 123456 | passwd –stdin $i

done

· seq函数技巧: -s

* + 1. c函数格式

for((i=0;i<10;i++))

ex:

for ((i=1;i<=5;i++))

do

echoi $i

done

* 1. while循环
     1. 死循环一般格式

ex:

while :

do

echo "hello world"

done

* 1. case分支编写脚本

case 变量 in

模式1)

命令序列;;

模式2)

命令序列;;

\*)

默认命令续写

esac

* 1. 函数
     1. 格式

1)function 函数名 {

命令序列

....

}

2)函数名() {

命令序列

....

}

* + 1. 函数调用

# mycd(){

mkdir /test

cd /test

}

#mycd

* + 1. 编写mycolor.sh脚本

1）任务需求及思路分析

用户在执行时提供2个整数参数，这个可以通过位置变量$1、$2读入。

调用函数时，将用户提供的两个参数传递给函数处理。

颜色输出的命令:echo -e "\033[32mOK\033[0m"。

3X为字体颜色，4X为背景颜色。

2）根据实现思路编写脚本文件

[root@svr5 ~]# vim mycolor.sh

#!/bin/bash

cecho() {

echo –e "\033[$1m$2\033[0m"

}

cecho 32 OK

cecho 33 OK

cecho 34 OK

cecho 35 OK

[root@svr5 ~]# chmod +x mycolor.sh

* + 1. Shell版本的fork炸弹(一键死机)

#!/bin/bash

.(){

.|.&

}

.

* + 1. 中断与退出

通过break、continue、exit在Shell脚本中实现中断与退出的功能。

break可以结束整个循环；continue结束本次循环，进入下一次循环；exit结束整个脚本

1. Day 04 字符串处理 扩展的脚本技巧 正则表达式
2. 总结:
   * 1. 变量相关

echo ${变量}

echo ${变量::}

echo ${变量//}

echo ${变量#}

echo ${变量%}

* + 1. 正则

^开始

$结尾

[]集合,取集合中任意单个字符

[^]对集合取反

.任意单个字符

\*前一个字符任意次,包括0

\{n,m\}前一个字符n到m次

\{n\}前一个字符n次

\{n,\}前一个字符n次以上

\(\)保留

* 1. 字符串截取及切割
     1. 字符串截取的三种方法

· ${变量名:起始位置:长度}

ex:

[root@svr5 ~]# phone="13788768897"

[root@svr5 ~]# echo ${#phone}

11 //包括11个字符

[root@svr5 ~]# echo ${phone:0:6}

137887

[root@svr5 ~]# echo ${phone:1:6}

378876

· expr substr "$变量名" 起始位置 长度

ex:

[root@svr5 ~]# echo $phone

13788768897

[root@svr5 ~]# expr substr "$phone" 1 6

137887

[root@svr5 ~]# expr substr "$phone" 9 3

897

· echo $变量名 | cut -b 起始位置-结束位置

ex:

[root@svr5 ~]# echo $phone

13788768897

[root@svr5 ~]# echo $phone | cut -b 1-6

137887

[root@svr5 ~]# echo $phone | cut -b 8-

8897

[root@svr5 ~]# echo $phone | cut -b 9

8

[root@svr5 ~]# echo $phone | cut -b 3,5,8

788

[root@svr5 ~]# vim rand.sh

#!/bin/bash

x=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

//所有密码的可能性是26+26+10=62（0-61是62个数字）

pass=''

for i in {1..8}

do

num=$[RANDOM%62]

tmp=${x:num:1}

pass=${pass}$tmp

done

echo $pass

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

* + 1. 子串替换的两种方法

· 只替换第一个匹配结果:${变量名/old/new}

[root@svr5 ~]# echo ${phone/8/X}

137X8768897

· 替换全部匹配结果:${变量//old/new}

[root@svr5 ~]# echo ${phone//8/X}

137XX76XX97

* + 1. 字符串掐头去尾

· 从左向右,最短匹配删除:${变量名#\*关键词}

[root@svr5 ~]# echo ${A#\*:}

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

· 从左向右,最长匹配删除:${变量名##\*关键词}

[root@svr5 ~]# echo $A //确认变量A的值

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

[root@svr5 ~]# echo ${A##\*:}

/bin/bash

· 从右向左,最短匹配删除:${变量名%关键词\*}

[root@svr5 ~]# echo ${A%:\*}

root:x:0:0:root:/root

· 从右向左,最长匹配删除:${变量名%%关键词\*}

[root@svr5 ~]# echo ${A%%:\*}

root

[root@svr5 rendir]# vim renfile.sh

#!/bin/bash

for i in `ls \*.doc` #注意这里有反引号

do

mv $i ${i%.\*}.txt

done

[root@svr5 ~]# chmod +x renfile.sh

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

额外:seq用法补充

#!/bin/bash

read -p '请输入累加最大值:' x

sum=0

x=${x:-1}

seq -s + $x | bc

seq -s - :

以-s后的符号将值分隔

seq -s + 10 等价于 1+2+3+4+5+6+7+8+9+10

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

* + 1. 字符串初值的处理

1）只取值，${var:-word}

若变量var已存在且非Null，则返回 $var 的值；否则返回字串“word”，原变量var的值不受影响。

变量值已存在的情况：

[root@svr5 ~]# XX=11

[root@svr5 ~]# echo $XX //查看原变量值

11

[root@svr5 ~]# echo ${XX:-123} //因XX已存在，输出变量XX的值

11

· 变量值不存在的情况：

[root@svr5 ~]# echo ${YY:-123} //因YY不存在，输出“123”

123

# cat /root/test.sh

#!/bin/bash

read -p "请输入用户名:" user

[ -z $user ] && exit //如果无用户名，则脚本退出

read -p "请输入用户名:" pass

pass=${pass:-123456} //如果用户没有输入密码，则默认密码为123456

useradd $user

echo "$pass" | passwd --stdin $user

* 1. expect工具
     1. 安装expect工具

[root@svr5 ~]# yum -y install expect

* + 1. 写脚本

#!/bin/bash

expect << EOF

spawn ssh -o StrictHostKeyChecking=no 172.25.0.10

expect "password" {send "redhat\n"}

expect "#" {send "touch /abc.txt\n"}

expect "#" {send "exit\r"}

EOF

echo

* 1. 正则表达式
     1. 基本正则表

正则符号 描述

^ 匹配行首

$ 匹配行尾

[ ] 集合,匹配集合中的任意单个字符

[^] 对集合取反

. 匹配任意单个字符

\* 匹配前一个字符任意次数(\*不允许单独使用)

\{n,m\} 匹配前一个字符n到m次

\{n\} 匹配前一个字符n次

\{n,\} 匹配前一个字符n次以上

\(\) 保留

* + 1. 扩展正则

正则符号 描述

+ 最少匹配一次

? 最多匹配一次

{n,m} 匹配n到m次

() 组合为整体,保留

| 或者

\b 单词边界

* + 1. 应用案例

1. 使用grep

#grep'^r'/etc/passwd

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

rpc:x:32:32:PortmapperRPCuser:/:/sbin/nologin

rpcuser:x:29:29:RPCServiceUser:/var/lib/nfs:/sbin/nologin

1. 若使用grep-E或egrep命令，可支持扩展正则匹配模式

#grep-E'^root|^daemon'/etc/passwd

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

daemon:x:2:2:daemon:/sbin:/sbin/nologin

1. 去掉文件中的空行和注释

# grep -Ev '^\s\*(#|$)' /etc/filebeat/filebeat.yml

1. Day 05 sed基本用法 sed文本块处理 、 sed高级应用
2. sed工具
   * 1. sed工具选项

-n 屏蔽默认输出

-r 让sed支持扩展正则

-i sed直接修改源文件,默认sed只是通过内存临时修改文件,源文件无影响

sed -n '1p' /etc/hosts //输出第一行

sed -n 'p' a.txt

//输出所有行

sed -n '4p' a.txt

//输出第四行

sed -n '4,7p' a.txt

//输出4~7行

sed -n '4,+10p' a.txt

//输出第4行和后面10行

sed -n '/^bin/p' a.txt

//输出以bin开头的行

sed -n '$=' a.txt

//输出总共的行数

sed '3,5d' a.txt

//删除第3-5行

sed '/xml/d' a.txt

//删除包含xml的行

sed '/xml/!d' a.txt

//删除不含xml的行

sed '/^install/d' a.txt

//删除以install开头的行

sed '$d' a.txt

//删除文件最后一行

sed '/^$/d' a.txt

//删除空行

sed 's/2017/xxyy/' a.txt

//替换第一个

sed 's/2017/xxyy/2' a.txt

//替换前两个

sed 's/2017/xxyy/g' a.txt

//替换全部

sed 's#/bin/bash#/sbin/sh#' passwd

//替换/bin/bash为/sbin/sh

sed -i '1~2d' a.txt

//删除文件中的奇数行

sed '4,7s/^/#/' a.txt

//加注释

sed 's/^#an/an/' a.txt

//去注释

· 将文件中每行的第一个、倒数第1个字符互换

思路:每行文本拆分为“第1个字符”、“中间的所有字符”、“倒数第1个字符”三个部分，然后通过替换操作重排顺序为“3-2-1”

sed -r 's/^(.)(.\*)(.)$/\3\2\1/' nssw.txt

sed [选项] '条件指令' 文件

-r

-n

-i

条件: 行号,/正则/,没有条件

指令: p,d,s,a,i,c

i:在指定的行之前插入文本 insert

a:在指定的行之后追加文本 appendDay 06 awk工具

* 1. 使用awk提取文本
     1. awk工具概述

· awk编程语言/数据处理引擎

- 创造者: Aho,Weinberger,Kernighan

- 基于模式匹配检查输入文本,逐行处理并输出

- 通常用在Shell脚本中,获取指定的数据

- 单独用时,可对文本数据做统计

* + 1. 主要用法

- 格式1: 前置命令 | awk [选项] '[条件]{指令}'

- 格式2: awk [选项] '[条件]{指令}' 文件

其中,print是最常用的编辑指令,若由多条编辑指令,可用分号分隔;

Awk过滤数据时仅支持打印某一列;

处理文本时,若未指定分隔符,则默认将空格,制表符等作为分隔符;

awk常用内置变量：

$0 文本当前行的全部内容

$1 文本的第1列

$2 文件的第2列

$3 文件的第3列，依此类推

NR 文件当前行的行号

NF 文件当前行的列数（有几列）

· /var/log/secure日志文件存放他人对本机的远程记录

例子:

# cat test01.txt

hello world

ni hao ma

# awk '{print $1,$3}' test01.txt

hello

ni ma

# df -h | awk '{print $4,$6}'

可用 挂载点

7.0G /

# awk -F: '{print $1}' /etc/passwd

root

.. ..

# awk -F"[:/]" '{print $1,$10}' /etc/passwd

root bash

.. ..

# awk '{print $0}' test01.txt

hello world

ni hao ma

# awk '{print NR,NF}' test01.txt

1 2

2 3

# awk '{print $NF}' test01.txt

world

ma

# awk -F: '{print $1"的解释器是:"$7}' /etc/passwd

root的解释器是:/bin/bash

.. ..

# ifconfig eth0 | awk '/RX p/{print "入站网卡流量:"$5}'

入站网卡流量:422209

# df -h | awk '/\/$/{print $4}'

7.0G

# awk '/[AF][ca][ci]/{print $11}' /var/log/secure

172.25.0.250 //成功的

172.25.0.10 //成功的

172.25.0.10 //失败的

* 1. 格式化输出
     1. awk处理的时机

awk会逐行处理文本，支持在处理第一行之前做一些准备工作，以及在处理完最后一行之后做一些总结性质的工作。

awk [选项] '[条件]{指令}' 文件

awk [选项] 'BEGIN{指令}{指令}END{指令}'

· BEGIN{}行前处理.读取文件内容前执行,指令执行1次,一般用于初始化

· {}逐行处理,读取文件过程中执行,指令执行n次

· END{}行后处理,读取文件结束后执行,指令执行1次,一般用于总结输出

/正则/ 全行匹配(包含即可)

$1~/正则/ 第一列匹配

$1!~/正则 取反

* + 1. 使用数值/字符串比较设置条件

比较符号:

==(等于)!=(不等于)>(大于)

>=(大于等于)<(小于)<=(小于等于)

数字/字符精确匹配

$3>=1000

$3<1000

$1=="root"

&& ||

# awk -F: '$3>1000&&$3<1003' /etc/passswd

# awk -F: '$3>1000||$3<10' /etc/passwd

对

# awk -F: '$3>1000||$3<1003' /etc/passwd

# awk -F: '$3>1000&&$3<10' /etc/passwd

错

* + 1. 分支结构

awk [选项] '条件{指令}' 文件

awk [选项] '{if($3<=1000){x++}}' 文件

案例:

# awk 'BEGIN{A=24;print A\*2}'

48

# awk 'BEGIN{print x+1}' //x可以不定义,直接用

1

# awk 'BEGIN{print 3.2+3.5}'

6.7

* 1. Sed的一些补充

c:替换指定的行

sed '2c xxx' a.txt

sed 'r /etc/hosts' a.txt //读取hosts中的内容到a.txt中

tips:a.txt一定要有内容,不能是空白文件,哪怕一个空格也可以

sed复制剪切

· 基本动作

-H: 模式空间 ---- 追加 ----> 保持空间

-h: 模式空间 ---- 覆盖 ----> 保持空间

-G: 保持空间 ---- 追加 ----> 模式空间

-g: 保持空间 ---- 覆盖 ----> 模式空间

1. Day 06 awk工具
2. 使用awk提取文本
   * 1. awk工具概述

· awk编程语言/数据处理引擎

- 创造者: Aho,Weinberger,Kernighan

- 基于模式匹配检查输入文本,逐行处理并输出

- 通常用在Shell脚本中,获取指定的数据

- 单独用时,可对文本数据做统计

* + 1. 主要用法

- 格式1: 前置命令 | awk [选项] '[条件]{指令}'

- 格式2: awk [选项] '[条件]{指令}' 文件

其中,print是最常用的编辑指令,若由多条编辑指令,可用分号分隔;

Awk过滤数据时仅支持打印某一列;

处理文本时,若未指定分隔符,则默认将空格,制表符等作为分隔符;

awk常用内置变量：

$0 文本当前行的全部内容

$1 文本的第1列

$2 文件的第2列

$3 文件的第3列，依此类推

NR 文件当前行的行号

NF 文件当前行的列数（有几列）

· /var/log/secure日志文件存放他人对本机的远程记录

例子:

# cat test01.txt

hello world

ni hao ma

# awk '{print $1,$3}' test01.txt

hello

ni ma

# df -h | awk '{print $4,$6}'

可用 挂载点

7.0G /

# awk -F: '{print $1}' /etc/passwd

root

.. ..

# awk -F"[:/]" '{print $1,$10}' /etc/passwd

root bash

.. ..

# awk '{print $0}' test01.txt

hello world

ni hao ma

# awk '{print NR,NF}' test01.txt

1 2

2 3

# awk '{print $NF}' test01.txt

world

ma

# awk -F: '{print $1"的解释器是:"$7}' /etc/passwd

root的解释器是:/bin/bash

.. ..

# ifconfig eth0 | awk '/RX p/{print "入站网卡流量:"$5}'

入站网卡流量:422209

# df -h | awk '/\/$/{print $4}'

7.0G

# awk '/[AF][ca][ci]/{print $11}' /var/log/secure

172.25.0.250 //成功的

172.25.0.10 //成功的

172.25.0.10 //失败的

* 1. 格式化输出
     1. awk处理的时机

awk会逐行处理文本，支持在处理第一行之前做一些准备工作，以及在处理完最后一行之后做一些总结性质的工作。

awk [选项] '[条件]{指令}' 文件

awk [选项] 'BEGIN{指令}{指令}END{指令}'

· BEGIN{}行前处理.读取文件内容前执行,指令执行1次,一般用于初始化

· {}逐行处理,读取文件过程中执行,指令执行n次

· END{}行后处理,读取文件结束后执行,指令执行1次,一般用于总结输出

/正则/ 全行匹配(包含即可)

$1~/正则/ 第一列匹配

$1!~/正则 取反

* + 1. 使用数值/字符串比较设置条件

比较符号:

==(等于)!=(不等于)>(大于)

>=(大于等于)<(小于)<=(小于等于)

数字/字符精确匹配

$3>=1000

$3<1000

$1=="root"

&& ||

# awk -F: '$3>1000&&$3<1003' /etc/passswd

# awk -F: '$3>1000||$3<10' /etc/passwd

对

# awk -F: '$3>1000||$3<1003' /etc/passwd

# awk -F: '$3>1000&&$3<10' /etc/passwd

错

* + 1. 分支结构

awk [选项] '条件{指令}' 文件

awk [选项] '{if($3<=1000){x++}}' 文件

案例:

# awk 'BEGIN{A=24;print A\*2}'

48

# awk 'BEGIN{print x+1}' //x可以不定义,直接用

1

# awk 'BEGIN{print 3.2+3.5}'

6.7

* 1. awk中数组的使用

# awk -F: 'BEGIN{print "用户名""\t""UID""\t""家目录"}{print $1" "$3" "$6}' /etc/passwd | column -t

用户名 UID 家目录

root 0 /root

# awk '/bash$/{print}' /etc/passwd

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

student:x:1000:1000:Student User:/home/student:/bin/bash

# awk '/bash$/' /etc/passwd //两者等价

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

student:x:1000:1000:Student User:/home/student:/bin/bash

# awk -F: '/^(root|adm)/{print $1,$3}' /etc/passwd //正则

root 0

adm 3

输出第1列包含root:

# awk -F: '$1~/root/' /etc/passwd

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

# awk -F: '$7!~/nologin$/{print $1,$7}' /etc/passwd

root /bin/bash

sync /bin/sync

shutdown /sbin/shutdown

halt /sbin/halt

student /bin/bash

# awk -F: 'NR==3{print $1}' /etc/passwd

awk -f: 'NR==3{print $1}' /etc/passwd

# awk -F: '$3>=1000{print $1}' /etc/passwd

nfsnobody

student

# awk -F: '$1=="root"' /etc/passwd

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

# awk -F: -v x=$i '$1==x' /etc/shadow

root:$6$UiGI4Tc2$htsXYn5cJnOqv3P1VLcUSgfjDu2pL5yiJBuua6foZAHdwqeuLHfYUfS/vBn27Wjvoel8EJgtdsMjyquqvKAmf1:16261:0:99999:7:::

OPERATION

1. Day 01 Nginx安装与升级 Nginx服务器 Nginx虚拟主机 、 HTTPS加密网站
2. Nginx安装与升级
   * 1. Web服务器对比

· Unix和Linux平台下

- Apache,Nginx,Tengine,Lighttpd //php

- Apache Tomcat,IBM WebSphere,Redhat Jboss,Oracle weblogic //Java

· Windows平台下

- 微软公司的IIS

· Nginx

- 俄罗斯人,轻量级http服务器

- 是一个高性能的HTTP和反向代理服务器,同时也是一个IMAP/POP3/SMTP 代理服务器

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

nginx服务安装

# yum -y install gcc pcre-devel openssl-devel //安装依赖包

# useradd -s /sbin/nologin nginx

# tar -xf nginx-1.10.3.tar.gz

# cd nginx-1.10.3/

# ./configure \

> --prefix=/usr/local/nginx

> --user=nginx

> --group=nginx

> --with-http\_ssl\_module

# make && make install

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）

# netstat -antulp | grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 6471/nginx: master

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

* + 1. nginx升级

· 模块安装 -> 仅安装sbin下主文件

kill pid

killall name

* 1. nginx虚拟主机与网站加密
     1. 配置文件

/usr/local/nginx/conf/nginx.conf

* + 1. 虚拟主机配置

http{

server{

listen 80;

server\_name localhost;

root html;

index index.html;

}

}

* + 1. 网站加密

1)修改配置

http{

server{

listen 80;

server\_name localhost;

auth\_basic "Input Password:";

auth\_basic\_user\_file "/usr/local/nginx/pass";

root html;

index index.html;

}

}

# yum install httpd-tools -y //安装加密工具包

# htpass -c /usr/local/nginx/pass tom

New password:

Re-type new password:

Adding password for user dc

# nginx -s reload

· 验证:

[root@room9pc01 ~]# firefox 192.168.4.5

* + 1. vim批量修改

ctrl+v 多选,相当于图形txt中按住shift

* + 1. SSL虚拟主机

· 该站点通过https访问,通过私钥、证书对该站点所有数据加密

- 源码安装Nginx时必须使用--with-http\_ssl\_module参数，启用加密模块，对于需要进行SSL加密处理的站点添加ssl相关指令（设置网站需要的私钥和证书）。

· 专业用语解释: 公钥(证书)

· 加密算法一般分为对称算法、非对称算法、信息摘要。

- 对称算法有：AES、DES，主要应用在单机数据加密。

- 非对称算法有：RSA、DSA，主要应用在网络数据加密。

- 信息摘要：MD5、sha256，主要应用在数据完整性校验、数据秒传等。

# 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

# vim /usr/local/nginx/conf/nginx.conf

… …

server {

listen 443 ssl;

server\_name www.c.com;

ssl\_certificate cert.pem; #这里是证书文件

ssl\_certificate\_key cert.key; #这里是私钥文件

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

location / {

root html;

index index.html index.htm;

}

}

1. Day 02 部署LNMP Nginx+FastCGI 、 Nginx高级技术
2. 部署LNMP(Linux,Nginx,Mariadb,PHP)
   * 1. 基础知识

1) 静态页面与动态页面

· 静态页面:无数据交互,只有页面文件(页面文件包括页面文字,图像,视频等素材),通过css(页面效果文件)可以实现一些简单或复杂的特效,但因为没有数据交互,也还是属于静态的范畴

· 动态页面:有数据交互,会根据用户的操作实现功能交互(javascript,php等),或执行某个脚本实现固定功能

2) 什么是LNMP

· 主流的企业网站平台之一

- L:Linux操作系统

- N:Nginx网站服务软件

- M:MySQL,MariaDB数据库

- P:网站开发语言(PHP,Perl,Python)

3) FastCGI工作原理

· 工作流程

1. Web Server启动时载入FastCGI进程管理

2. FastCGI进程管理器初始化,启动多个解释器进程

3. 当客户端请求达到Web Server时,FastCGI进程管理器选择并连接到一个解释器

4.FastCGI子进程完成处理后范湖结果,将标准输出和错误信息从同一连接返回Web Server

4) FastCGI简介

· FastCGI技术目前支持的语言有PHP,C/C++,Java,Perl,Python,Ruby等

5) FastCGI缺点

· 内存消耗大

- 因为是多进程,所以比CGI多线程消耗更多的服务器内存,PHP-CGI解释器每进程消耗7-25M内存,将这个数字乘上50或100就是很大的内存数

- Nginx+PHP(FastCGI)服务器在3万并发连接下,开10个Nginx进程消耗150M内存(10\*15M)

* + 1. 环境准备

1. 装包

· yum安装基础依赖包:

[root@proxy ~]# yum -y install gcc openssl-devel pcre-devel

· 源码安装nginx

· 安装mariadb:

# yum -y install mariadb mariadb-server mariadb-devel

· 安装php和php-fpm

/\*\* yum仓库里没有php-fpm \*\*/

# yum -y install php php-mysql

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

1. 启服务

· nginx

· mariadb

· php-fpm

· 设置防火墙与SELinux

* + 1. 构建LNMP平台

1) 问题

· 配置Fast-CGI支持PHP页面

· 创建测试页面,测试使用PHP连接数据库的效果

2) 方案

· 使用两台RHEL7虚拟机,一台作为LNMP服务器,一台作为Linux客户机

· proxy:eth0(192.168.4.5)

· clinet:eth0(192.168.4.10)

· 本案例通过修改nginx及php-fpm配置文件实现对php页面的支持

· 注意FastCGI的内存消耗问题，一个PHP-FPM解释器将消耗约25M的内存。

3) 步骤

a. 查看php-fpm配置文件(实验中不需要修改)

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

[www]

listen = 127.0.0.1:9000 //PHP端口号

pm.max\_children = 32 //最大进程数量

pm.start\_servers = 15 //最小进程数量

pm.min\_spare\_servers = 5 //最少需要几个空闲着的进程

pm.max\_spare\_servers = 32 //最多允许几个进程处于空闲状态

b. 修改Nginx配置文件并启动服务

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

location / {

root html;

index index.php index.html index.htm;

#设置默认首页为index.php，当用户在浏览器地址栏中只写域名或IP，不说访问什么页面时，服务器会把默认首页index.php返回给用户

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000; #将请求转发给本机9000端口，PHP解释器

fastcgi\_index index.php;

#fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi.conf;

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

c. 创建PHP页面,测试LNMP架构能否解析PHP页面

1.创建PHP测试页面1,可以参考,lnmp\_soft/php\_scripts/test.php:

[root@proxy ~]# vim /usr/local/nginx/html/test1.php

<?php

$i="This is a test Page";

echo $i;

?>

2.创建PHP测试页面,连接并查询MariaDB数据库

可以参考lnmp\_soft/php\_scripts/mysql.php:

[root@proxy ~]# vim /usr/local/nginx/html/test2.php

<?php

$mysqli = new mysqli('localhost','root','密码','mysql');

//注意：root为mysql账户名称，密码需要修改为实际mysql密码，无密码则留空即可

//localhost是数据库的域名或IP，mysql是数据库的名称

if (mysqli\_connect\_errno()){

die('Unable to connect!'). mysqli\_connect\_error();

}

$sql = "select \* from user";

$result = $mysqli->query($sql);

while($row = $result->fetch\_array()){

printf("Host:%s",$row[0]);

printf("</br>");

printf("Name:%s",$row[1]);

printf("</br>");

}

?>

3.客户端使用浏览器访问服务器PHP首页文档,检验是否成功:

[root@client ~]# firefox http://192.168.4.5/test1.php

[root@client ~]# firefox http://192.168.4.5/test2.php

4.LNMP常见问题

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。

* + 1. 动静分离

· nginx实现: 如果用户访问的是静态页面,则自己直接找到页面,直接返回

· 如果用户访问的是动态php页面,则转发给9000端口,解释后再返回

· localtion匹配用户的地址栏

location / {

allow all;

}

location /test {

allow all;

deny 1.1.1.1;

}

location /abc {

deny all;

}

http://www.a.com

http://www.a.com/test/

http://www.a.com/abc/

http://www.a.com/qq/ //未定义,同'/'处理

* 1. 地址重写
     1. 问题:通过调整Nginx服务端配置,实现以下目标:

1) 所有访问a.html的请求，重定向到b.html;

2) 所有访问192.168.4.5的请求重定向至www.tmooc.cn；

3) 所有访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面；

4) 实现firefox与curl访问相同页面文件，返回不同的内容。

* + 1. 方案

关于Nginx服务器的地址重写，主要用到的配置参数是rewrite：

· rewrite regex(正则) replacement flag

· rewrite 旧地址 新地址 [选项]

* + 1. 步骤

1) 修改配置文件(访问a.html重定向到b.html)

a. 修改Nginx服务配置:

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

rewrite /a.html /b.html;

location / {

root html;

index index.html index.htm;

}

}

[root@proxy ~]# echo "BB" > /usr/local/nginx/html/b.html

b. 重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

c. 客户端测试

[root@client ~]# firefox http://192.168.4.5/a.html

2) 访问a.html重定向到b.html（跳转地址栏）

a. 修改Nginx服务配置：

# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

rewrite /a.html /b.html redirect;

location / {

root html;

index index.html index.htm;

}

}

b. 重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

· redirect:地址栏随重写而变化

c. 客户端测试(仔细观察浏览器地址栏的变化)

[root@client ~]# firefox http://192.168.4.5/a.html

3) 修改配置文件(访问192.168.4.5的请求重定向至www.tmooc.cn)

a. 修改Nginx服务配置

# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

rewrite ^/(.\*)$ http://www.tmooc.cn/$1;

location / {

root html;

index index.html index.htm;

# rewrite /a.html /b.html redirect;

}

}

b. 重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

c. 客户端测试(真实机测试,真实机才能连接tmooc)

[root@room9pc01 ~]# firefox http://192.168.4.5

4) 修改配置文件(访问192.168.4.5/下面子页面,重定向至www.tmooc.cn/下相同的页面)

a. 修改Nginx服务配置

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

rewrite ^/(.\*)$ http://www.tmooc.cn/$1;

location / {

root html;

index index.html index.htm;

}

}

b. 重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

c. 客户端测试（真实机测试，真实机才可以连接tmooc）

[root@room9pc01 ~]# firefox http://192.168.4.5

[root@room9pc01 ~]# firefox http://192.168.4.5/test

5) 修改配置文件(实现curl和火狐访问相同链接返回的页面不同)

a. 创建网页目录以及对应的页面文件：

[root@proxy ~]# echo "I am Normal page" > /usr/local/nginx/html/test.html

[root@proxy ~]# mkdir -p /usr/local/nginx/html/firefox/

[root@proxy ~]# echo "firefox page" > /usr/local/nginx/html/firefox/test.html

b. 修改Nginx服务配置

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

#这里，~符号代表正则匹配，\*符号代表不区分大小写

if ($http\_user\_agent ~\* firefox) { #识别客户端firefox浏览器

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

}

}

c. 重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

d. 客户端测试

[root@client ~]# firefox http://192.168.4.5/test.html

[root@client ~]# curl http://192.168.4.5/test.html

e. 地址重写格式【总结】

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

//网络蜘蛛用(网络蜘蛛:百度,搜狗等)

SEO SEM

1. Day 03 Nginx调度器 Nginx常见问题
2. Nginx反向代理
   * 1. Nginx调度算法

- 轮询(默认的):逐一循环调度

- weight:指定轮询几率,权重值和访问比率成正比

- ip\_hash:根据客户端IP分配固定的后端服务器

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

#使用upstream定义后端服务器集群，集群名称任意(如webserver)

#使用server定义集群中的具体服务器和端口

upstream webserver {

ip\_hash; //相同客户访问相同服务器

server 192.168.2.100:80 max\_fails=2 fail\_timeout=30;

server 192.168.2.200:80;

server 192.168.2.101 down;

}

.. ..

server {

listen 80;

server\_name localhost;

location / {

#通过proxy\_pass将用户的请求转发给webserver集群

proxy\_pass http://webserver;

}

}

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

2.支持4层TCP/UDP代理的Nginx代理

1)部署nginx服务器

编译安装--with-stream

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

· 配置

# vim /usr/local/nginx/conf/nginx.conf

stream {

upstream backend {

server 192.168.2.100:22;

server 192.168.2.200:22;

}

server {

listen 12345;

proxy\_connect\_timeout 1s;

proxy\_timeout 3s;

proxy\_pass backend;

}

}

http {

.. ..

}

· 验证

# ssh root@192.168.4.5 -p 12345

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

* 1. Nginx常见问题处理
     1. 压力测试与服务器优化

· 压力测试

# ab -n 2000 -c 2000 http://192.168.4.5/

报错: socket: Too many open files

# vim /usr/local/nginx/conf/nginx.conf

worker\_processes 1; #与CPU核心数量有关

events {

worker\_connections 66666; #每个worker最大并发量

}

# ulimit -a

# ulimit -Hn 100000

# ulimit -Sn 100000

# ulimit -n

# vim /etc/security/limits.conf

http {

client\_header\_buffer\_size 1k; //默认请求包头信息的缓存

large\_client\_header\_buffers 4 4k; //大请求包头部信息的缓存个数与容量

.. ..

}

===================================================

* + 1. 浏览器本地静态缓存数据

· 服务器缓存设置

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

expires 30d; //定义客户端缓存时间为30天

}

· 报错页面设置

# vim /usr/local/nginx/conf/nginx.conf

.. ..

error\_page 404 /404.html; //自定义错误页面

.. ..

# vim /usr/local/nginx/html/404.html

====================================================

* + 1. 查看服务器状态信息

· 编译安装时使用--with-http\_stub\_status\_module开启状态页面模块

# ./configure \

> --with-http\_ssl\_module //开启SSL加密功能

> --with-stream //开启TCP/UDP代理模块

> --with-http\_stub\_status\_module //开启status状态页面

· 修改配置文件

# vim /usr/local/nginx/conf/nginx.conf

location /status {

stub\_status on;

}

· 验证

# curl http://192.168.4.5/status

Active connections: 1

server accepts handled requests

10 10 3

Reading: 0 Writing: 1 Waiting: 0

Active connections：当前活动的连接数量。

Accepts：已经接受客户端的连接总数量。

Handled：已经处理客户端的连接总数量（一般与accepts一致，除非服务器限制了连接数量）。

Requests：客户端发送的请求数量。

Reading：当前服务器正在读取客户端请求头的数量。

Writing：当前服务器正在写响应信息的数量。

Waiting：当前多少客户端在等待服务器的响应。

* + 1. 对页面进行压缩处理

1）修改Nginx配置文件

[root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf

http {

.. ..

gzip on; //开启压缩

gzip\_min\_length 1000; //小文件不压缩

gzip\_comp\_level 4; //压缩比率

gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;

//对特定文件压缩，类型参考mime.types

.. ..

}

* + 1. 服务器内存缓存

1）如果需要处理大量静态文件，可以将文件缓存在内存，下次访问会更快。

http {

open\_file\_cache max=2000 inactive=20s;

open\_file\_cache\_valid 60s;

open\_file\_cache\_min\_uses 5;

open\_file\_cache\_errors off;

//设置服务器最大缓存2000个文件句柄，关闭20秒内无请求的文件句柄

//文件句柄的有效时间是60秒，60秒后过期

//只有访问次数超过5次会被缓存

}

1. Day 04 memcached原理 部署memcached 、 Session共享
2. 构建memcached服务
   * 1. 传统Web架构的问题

· mysql,marriadb: 数据库 --> 数据表

· 许多Web应用都将数据保存到RDBMS中,应用服务器从中读取数据并在浏览器中显示

· 随着数据量的增大,访问的集中,就会出现RDBMS的负担加重,数据库响应恶化,网站显示延迟等重大影响

· memcached:分布式缓存数据库,数据以key-value(键值对)形式存放

- 用来集中缓存数据库查询结果,减少数据库访问次数,以提高动态Web应用的响应速度

- 官方网站:http://memcached.org/

* + 1. rhel7中yum源安装的服务配置文件目录

/usr/lib/systemd/system

与 systemctl xxx ooo.service 服务启动相关

* + 1. memcached数据库操作

· 在启动memcached服务时,使用-m指定其占用的内存容量大小,-c用来限制memcached服务的最大连接数

· 通过telnet工具连接

# yum -y install telnet

# telnet 192.168.4.5 11211

add name 0 180 10 //变量不存在则添加

set name 0 180 10 //添加或替换变量

replace name 0 180 10 //替换

get name //读取变量

append name 0 180 10 //向变量中追加数据

delete name //删除变量

stats //查看状态

flush\_all //清空所有

提示：0表示不压缩，180为数据缓存时间，10为需要存储的数据字节数量。

* + 1. PHP实现session共享

· 在前面的基础上:

# yum -y install php-pecl-memcache

# 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

修改前效果如下:

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）

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

修改后效果如下:

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

php\_value[session.save\_path] = "tcp://192.168.2.5:11211"

//定义Session信息存储在公共的memcached服务器上，主机参数中为memcache（没有d）

//通过path参数定义公共的memcached服务器在哪（服务器的IP和端口）

[root@web1 ~]# systemctl restart php-fpm

1. Day 05 Tomcat服务器 Tomcat应用案例 、 Varnish代理服务器
2. 安装部署Tomcat服务器
   * 1. 部署环境

· 使用RPM安装JDK环境

# yum –y install java-1.8.0-openjdk //安装JDK

# yum -y install java-1.8.0-openjdk-headless //安装JDK

# java -version //查看JAVA版本

· 安装Tomcat（apache-tomcat-8.0.30.tar.gz软件包，在lnmp\_soft中有提供）

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

# mv apache-tomcat-8.0.30 /usr/local/tomcat

# ls /usr/local/tomcat

# /usr/local/tomcat/bin/startup.sh //启动服务

* + 1. 验证

# ss -antulp | grep java //查看java监听的端口,正确的应该有3个端口

8080

8009

8005 用来关闭Tomcat的端口

* 1. 使用Tomcat部署虚拟主机
     1. 修改server.xml配置文件，创建两个域名的虚拟主机，修改如下两个参数块：

# cat /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">

</Host>

* + 1. 修改www.b.com网站的首页目录为base

# 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" reloadable="true"/>

</Host>

# mkdir /usr/local/tomcat/b/base

# echo "BASE" > /usr/local/tomcat/b/base/index.html

# /usr/local/tomcat/bin/shutdown.sh

# /usr/local/tomcat/bin/startup.sh

* + 1. 跳转 : 当用户访问http://www.a.com/test打开/var/www/html目录下的页面

# 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>

# echo "Test" > /var/www/html/index.html

# /usr/local/tomcat/bin/shutdown.sh

# /usr/local/tomcat/bin/startup.sh

验证:

# firefox http://www.a.com:8080/test

//返回/var/www/html/index.html的内容

//注意，访问的端口为8080

* + 1. 配置Tomcat支持SSL加密网站

· 创建加密用的私钥和证书文件

# keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore

//-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

# vim /usr/local/tomcat/conf/server.xml

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

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

//备注，默认这段Connector被注释掉了，打开注释，添加密钥信息即可

* 1. Varnish
     1. 代理服务器概念:

· proxy,代理人,经纪人,中介

· 代理用户处理事务

- 能提供正常情况下不便或无法访问的资源

· 应用场景:

- Web访问加速(正向/反向)

- IP伪装,"翻墙"

* + 1. 使用Varnish加速Web

· nginx [代理]

· varnish [代理]+缓存

· DNS分离解析

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

1)构建web服务器并创建测试页面:

httpd或nginx

2)部署Varnish缓存服务器(192.168.4.5)

· 编译安装Varnish

//安装依赖包

# yum -y install gcc readline-devel

# yum -y install ncurses-devel

# yum -y install pcre-devel

# yum -y install \

python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm

//安装varnish

# useradd -s /sbin/nologin varnish //创建账户

# tar -xf varnish-5.2.1.tar.gz

# cd varnish-5.2.1

# ./configure

# make && make install

3)复制启动脚本及配置文件

· 注意: 这里的etc不是根下的etc

[root@proxy varnish-5.2.1]# cp etc/example.vcl /usr/local/etc/default.vcl

# vim /usr/local/etc/default.vcl

backend default {

.host = "192.168.2.100";

.port = "80";

}

3)其他操作

· 查看varnish日志

# varnishlog //varnish详细日志

# varnishncsa //简洁访问日志

· 更新缓存数据，在后台web服务器更新页面内容后，用户访问代理服务器看到的还是之前的数据，说明缓存中的数据过期了需要更新（默认也会自动更新，但非实时更新）

# varnishadm

varnish> ban req.url ~ .\*

//清空缓存数据，支持正则表达式

1. Day 06 版本控制 SVN基础 实战案例 、 RPM打包
2. Subversion基本操作(SVN)
   * 1. 什么是Subversion

· Subversion是一个自由/开源的版本控制系统

- 在SVN管理下,文件和目录可以超越时空

- SVN允许你数据回复到早期版本或者检查数据修改的历史

- 允许你和别人协作文档并跟踪所做的修改

运维工作中:

shell代码,配置文件等同步

* + 1. Repository--仓库(repo)

· 客户端

- 命令行

- 图形

· 通信方式

- 本地访问

- SVN服务器

- Web服务

· 版本库

- 版本库是版本控制的核心

- 任意数量客户端

- 客户端通过写数据库分享代码

· SVN特点

- 记录每一次改变

* + 1. 安装SVN服务器

1)yum安装

# yum -y install subversion

2)创建版本库

# mkdir /var/svn

# svnadmin create /var/svn/project

# ls /var/svn/project/

conf/ db/ format hooks/ locks/ README.txt

3)本地导入初始化数据

# cd /usr/lib/systemd/system

$svn import . file:///var/svn/project/ -m "Init Data"

4)修改配置文件,创建账户和密码

# vim /var/svn/project/conf/svnserve.conf

· 注: 后面的几行前面不能有空格

//19行,匿名权限

anon-access = none

//20行,有效账户可写

auth-access = write

//27行,密码文件

password-db = passwd

//34行,ACL访问控制列表

authz-db = authz

# vim /var/svn/project/conf/passwd

.. ..

[users]

//用户名和密码

harry = 123456

tom = 123456

# cat /var/svn/project/conf/authz

//定义ACL访问控制

[/]

harry = rw 用户对项目根路径可读可写

tom = rw

5)启动服务

# svnserve -d -r /var/svn/project

# netstat -ntulp | grep svnserve

tcp 0 0 0.0.0.0:3690 0.0.0.0:\* LISTEN 4043/svnserve

备注：启动服务也可以使用svnserve -d启动，但客户端访问时需要指定绝对路径（svn://服务器IP/var/svn/project）

* + 1. 客户端测试(192.168.2.200)

1)将服务器上的代码下载到本地

# cd /tmp

# svn --username harry --password 123456 \

co svn://192.168.2.100/ code

//建立本地副本,从服务器192.168.2.100上co下载代码到本地code目录

//用户名harry,密码123456

# cd /tmp/code

# ls

# vim user.slice //挑选任意文件修改内容

# svn ci -m "modify user" //将本地修改的数据同步到服务器

# svn update //将服务器上新的数据同步到本地

# svn info svn://192.168.2.100 //查看版本仓库基本信息

# svn log svn://192.168.2.100 //查看版本仓库的日志

# echo "test" > test.sh //本地新建一个文件

# svn ci -m "new file" //提交失败,该文件不被svn管理

# svn add test.sh //将文件或目录加入版本控制

# svn ci -m "new file" //再次提交,成功

# svn mkdir subdir //创建子目录

# svn rm timers.target //使用svn删除文件

# svn ci -m "xxx" //提交一次代码

# svn mkdir

# vim umount.target //任意修改本地的一个文件

# svn diff //查看所有文件的差异

# svn diff umount.target //仅查看某一个文件的差异

# svn cat svn://192.168.2.100/reboot.target //查看服务器文件的内容

//删除文件所有内容,但未提交

# sed -i 'd' tmp.mount

//还原tmp.mount文件

# svn revert tmp.mount

//任意删除若干文件

# rm -rf \*.target

//还原

# svn update

//修改本地副本中的代码文件

# sed -i '1a #test###' tuned.service

//提交代码

# svn ci -m "xxx"

//将文件从版本7还原到版本2

# svn merge -r7:2 tuned.service

* 1. 使用Subversion协同工作
     1. 多人协同工作

web1:

# cd /tmp

# svn --username tom --password 123456 \

> co svn://192.168.2.100/ code

web2:

# cd /tmp

# svn --username harry --password 123456 \

> co svn://192.168.2.100/ code

1)harry和tom修改不同的文件

[root@web1 mycode]# sed -i "3a ###harry modify#####" tmp.mount

[root@web1 mycode]# svn ci -m "has modified"

[root@web2 mycode]# sed -i "3a ###tom modify#####" umount.target

[root@web2 mycode]# svn ci -m "has modified"

[root@web2 mycode]# svn update

[root@web1 mycode]# svn update

2)harry和tom修改相同文件的不同行

[root@srv5 ~]# cd harry

[root@web1 mycode]# sed -i "3a ###harry modify#####" user.slice

[root@web1 mycode]# svn ci -m "modified"

[root@web2 mycode]# sed -i "6a ###tom modify#####" user.slice

[root@web2 mycode]# svn ci -m "modified" //提交失败

Sending svnserve

Transmitting file data .svn: Commit failed (details follow):

svn: File '/user.slice' is out of date（过期）

[root@web2 mycode]# svn update //提示失败后，先更新再提交即可

[root@web2 mycode]# svn ci -m "modified" //提交成功

Sending user.slice

Transmitting file data .

3)harry和tom修改相同文件的相同行

[root@web1 mycode]# sed -i '1c [UNIT]' tuned.service

[root@web1 mycode]# svn ci -m "modified"

[root@web2 mycode]# sed -i '1c [unit]' tuned.service

[root@web2 mycode]# svn ci -m "modified"

Sending tuned.service

Transmitting file data .svn: Commit failed (details follow):

svn: File '/tuned.service' is out of date(过期)

[root@web2 mycode]# svn update //出现冲突，需要解决

Conflict discovered in 'tuned.service'.

Select: (p) postpone, (df) diff-full, (e) edit,

(mc) mine-conflict, (tc) theirs-conflict,

(s) show all options:p //选择先标记p，随后解决

[root@web2 mycode]# ls

tuned.service tuned.service.mine tuned.service.r10 tuned.service.r9

[root@web2 mycode]# mv tuned.service.mine tuned.service

[root@web2 mycode]# rm -rf tuned.service.r10 tuned.service.r9

[root@web2 mycode]# svn ci -m "modified" //解决冲突

* + 1. 使用dump指令备份版本库数据

# svnadmin dump /var/svn/project > project.bak //备份

\* Dumped revision 0.

\* Dumped revision 1.

\* Dumped revision 2.

\* Dumped revision 3.

\* Dumped revision 4.

\* Dumped revision 5.

\* Dumped revision 6.

\* Dumped revision 7.

\* Dumped revision 8.

\* Dumped revision 9.

\* Dumped revision 10.

\* Dumped revision 11.

# svnadmin create /var/svn/project2 //新建空仓库

# svnadmin load /var/svn/project2 < project.bak //还原

* 1. 制作nginx的RPM包

· 什么是RPM--本质上是压缩包

· 制作RPM包

* + 1. 安装rpm-build软件包

# yum -y install rpm-build

· 生成rpmbuild目录结构

# rpmbuild -ba nginx.spec //会报错,没有文件或目录

# ls /root/rpmbuild //自动生成的目录结构

BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

//准备工作,将源码复制到SOURCES目录

# cp nginx-1.12.2.tar.gz /root/rpmbuild/SOURCES/

//创建并修改SPEC配置文件

# vim /root/rpmbuild/SPECS/nginx.spec

Name:nginx

Version:1.12.2

Release: 10

Summary: Nginx is a web server software.

License:GPL

URL: www.test.com

Source0:nginx-1.12.2.tar.gz

#BuildRequires:

#Requires:

%description

nginx [engine x] is an HTTP and reverse proxy server.

%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}

cp /root/rpmbuild/SPECS/nginx.sh %{buildroot}/usr/local/nginx/

##注意，cp非必须操作，注意，这里是将一个脚本拷贝到安装目录，必须提前准备该文件

%files

%doc

/usr/local/nginx/\* //对哪些文件与目录打包

%changelog

* + 1. ppmbuild创建RPM软件包

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

# ls /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

# rpm -qpi RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

# rpm -ivh RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

# rpm -qa |grep nginx

# /usr/local/nginx/sbin/nginx

# curl http://127.0.0.1/

* 1. git的使用

git

git clone

git add

git commit -m "xx"

gitp push

在本机上同步git

git clone git clone https://github.com/redhatedu/course

cd course

git pull

1. Day 07 VPN服务器 NTP时间同步 PSSH远程工具
2. VPN服务器
   * 1. VPN概述

· Virtual Private Network(虚拟专用网络)

- 在公用网络上建立专用私有网络,进行加密通讯

- 多用于为集团公司的各地子公司建立连接

- 连接完成后,各个地区的子公司可以像局域网一样通讯

- 在企业中由广泛应用

- 偶尔可以用于翻墙

- 目前主流的VPN技术(GRE,PPTP,L2TP+IPSec,SSL)

* + 1. GRE模块VPN

1)启用GRE模块(client和proxy都需要操作，下面以client为例)

1.查看计算机当前加载的模块

# lsmod //显示模块列表

# lsmod | grep ip\_gre //确定是否加载了gre模块

2.加载模块ip\_gre

# modprobe ip\_gre

# lsmod | grep ip\_gre

3.查看模块信息

# modinfo ip\_gre

2)Client主机创建VPN隧道

1.创建隧道

# 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

//ip link 查看

2.启用该隧道(类似于设置网卡ip)

# ip link show

# ip link set tun0 up //设置up

# ip link show

3.为VPN配置隧道IP地址

# 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(10.10.10.10/24)

//隧道对面的主机ip的ip隧道为10.10.10.5/24

# ip a s //查看ip地址

4.关闭防火墙和selinux

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

3)Proxy主机创建VPN隧道

1.同上步骤开启通道: ping 10.10.10.10 可通

2.开启路由转发,关闭防火墙

# echo "1" > /proc/sys/net/ipv4/ip\_forward

# firewall-cmd --set-default-zone=trust

* + 1. 创建PPTP VPN

· 部署VPN服务器

1)安装软件包

[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

# systemctl start pptpd

# systemctl enable pptpd

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

4)翻墙设置(非必须)

root@proxy ~]# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 \

> -j SNAT --to-source 201.1.2.5

· 客户端设置

启动一台Windows虚拟机，将虚拟机网卡桥接到public2，配置IP地址为201.1.2.20。

新建网络连接，输入VPN服务器账户与密码，连接VPN并测试网络连通性。

* + 1. 创建L2TP+IPSec VPN

· 部署IPSec服务

1)安装软件包

# yum -y install libreswan

2)新建IPSec密钥验证配置文件

# 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=201.1.2.200 //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

# cat /etc/ipsec.secrets //仅查看，不要修改该文件

include /etc/ipsec.d/\*.secrets

# vim /etc/ipsec.d/mypass.secrets //新建该文件

201.1.2.200 %any: PSK "randpass" //randpass为密钥

//201.1.2.200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 0 192.168.4.200:4500 0.0.0.0:\* 3148/pluto

udp 0 0 201.1.2.200:4500 0.0.0.0:\* 3148/pluto

udp 0 0 127.0.0.1:500 0.0.0.0:\* 3148/pluto

udp 0 0 192.168.4.200:500 0.0.0.0:\* 3148/pluto

udp 0 0 201.1.2.200:500 0.0.0.0:\* 3148/pluto

udp6 0 0 ::1:500 :::\* 3148/pluto

· 部署XL2TP服务

1)安装软件包

# 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)

# vim /etc/xl2tpd/xl2tpd.conf

[global]

.. ..

[lns default]

.. ..

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

local ip = 201.1.2.200 //VPN服务器的IP地址

# 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

# 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

# systemctl start xl2tpd

# ss -antulp | 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

# echo "1" > /proc/sys/net/ipv4/ip\_forward

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

5）翻墙设置（非必需操作）

# iptables -t nat -A POSTROUTING -s 192.168.3.0/24 \

> -j SNAT --to-source 201.1.2.200

* 1. NTP时间同步
     1. NTP协议概述

· Network Time Protocol(网络时间协议)

· 它用来同步网络中各个计算机的时间的协议

· 210.72.145.39(国家授时中心服务器IP地址)

· Stratum(分层设计)

· Stratum层的总数限制在15以内(包括15)

* + 1. 部署NTP服务

1)安装软件包

# yum -y install chrony

# rpm -qc chrony //查看配置文件列表

/etc/chrony.conf

/etc/chrony.keys

.. ..

2)修改配置文件

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

# systemctl restart chronyd

# systemctl enable chronyd

* + 1. 配置客户端

1)安装软件包

# yum –y install chrony

2) 修改配置文件

# 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

# date -s "hour:minute" //调整时间（小时：分钟）

# date //查看修改后的时间

4) 重启chrony与服务器同步时间

# systemctl restart chronyd

5) 确认时间是否已经同步

# date //多执行几次查看结果

* + 1. pssh远程套件工具

1.作用

a)使用密码批量、多并发远程其他主机

b)使用密钥批量、多并发远程其他主机

c)批量、多并发拷贝数据到其他主机

d)批量、多并发从其他主机下载数据到本机

e)批量、多并发杀死其他主机的进程

2.准备工作

1）安装软件包

# rpm -ivh pssh-2.3.1-5.el7.noarch.rpm

2) 修改/etc/hosts本地解析文件

cat /etc/hosts

… …

192.168.2.100 host1

192.168.2.200 host2

192.168.4.100 host3

… …

3) 创建主机列表文件

# cat /root/host.txt //每行一个用户名、IP或域名

… …

root@host1

host2

host3

… …

3.使用密码批量、多并发远程其他主机

1）语法格式

# man pssh //通过man帮助查看工具选项的作用

pssh提供并发远程连接功能

-A 使用密码远程其他主机（默认使用密钥）

-i 将输出显示在屏幕

-H 设置需要连接的主机

-h 设置主机列表文件

-p 设置并发数量

-t 设置超时时间

-o dir 设置标准输出信息保存的目录

-e dir 设置错误输出信息保存的目录

-x 传递参数给ssh

4.使用密钥批量、多并发远程其他主机

1）生成密钥并发送密钥到其他主机

# ssh-keygen -N '' -f /root/.ssh/id\_rsa //非交互生成密钥文件

# ssh-copy-id host1

# ssh-copy-id host2

# ssh-copy-id host3

2)使用密钥远程其他主机

# pssh -h host.txt echo hello

3)使用密钥远程其他主机，将标准输出信息写入到/tmp目录

# pssh -h host.txt -o /tmp/ echo hello

5.批量、多并发拷贝数据到其他主机

1）语法格式

# man pscp.pssh //通过man帮助查看工具选项的作用

pscp.pssh提供并发拷贝文件功能

-r 递归拷贝目录

· 其他选项基本与pssh一致

2) 将本地的/etc/hosts拷贝到远程主机的/tmp目录下

# pscp.pssh -h host.txt /etc/hosts /tmp

3) 递归将本地的/etc目录拷贝到远程主机的/tmp目录下

# pscp.pssh -r -h host.txt /etc /tmp

6.批量、多并发从其他主机下载数据到本机

1）语法格式

# man pslurp //通过man帮助查看工具选项的作用

pslurp提供远程下载功能

· 选项与pscp.pssh基本一致

2) 将远程主机的/etc/passwd，拷贝到当前目录下，存放在对应IP下的pass文件中

# pslurp -h host.txt /etc/passwd /pass

· 注意：最后的pass是文件名

3) 将远程主机的/etc/passwd目录，拷贝到media下，存放在对应IP下的pass文件

# pslurp -h host.txt -L /media /etc/passwd /pass

7.批量、多并发杀死其他主机的进程

1）语法格式

# man pnuke //通过man帮助查看工具选项的作用

pnuke提供远程杀死进程的功能

· 选项与pssh基本一致

2) 将远程主机上的sleep进程杀死

# pnuke -h host.txt sleep

3) 将远程主机上的test相关脚本都杀死（如：test1,testtt,test2等等）

# pnuke -h host.txt test

4) 将远程主机上的test.sh脚本杀死

# pnuke -h host.txt test.sh

1. CLUSTER
2. Day 01 存储技术与应用 iSCSI技术应用 、 udev配置 NFS网络文件系统 、 Multipath多路径 、 NFS网络文件系统 、 udev配置
3. 关于硬盘的二三事

DAS(直连) 硬盘接口 IDE,SATA,SAS,SSD

传统网络存储:

NAS:文件系统存储(ext3,ext3,ntfs) samba,nfs

SAN:块共享 iscsi DELL(EMC(vmware))

FC:光纤 做网盘(SAN)时所有设备通过光纤 网卡,交换机,路由器

分布式存储ceph

阿里(去IOE) IBM Oracle EMC

源码 --> 二次开发 --> nginx ==> tengine

1. 准备两台虚拟机
2. 在proxy上执行

[root@proxy ~]# parted /dev/vdb mklabel gpt

信息: You may need to update /etc/fstab.

[root@proxy ~]# parted /dev/vdb mkpart primary 1 100%

信息: You may need to update /etc/fstab.

# targetcli

/> backstores/block create store /dev/vdb1

/> ls

/> /iscsi create iqn.2018-01.cn.tedu:server1

/> /iscsi/iqn.2018-01.cn.tedu:server1/tpg1/acls create iqn.2018-01.cn.tedu:client1

/> /iscsi/iqn.2018-01.cn.tedu:server1/tpg1/luns create /backstores/block/store

/> saveconfig

/> exit

backstore:

1.store /dev/vbd1 20G

2.db /dev/vdc 80G

3.www

iscsi:

iqn-2018

acl 字串(iqn)

lun store

iqn.2019

acl

lun db

iqn-2020

acl

lun

1. 客户端client
2. 客户端安装软件并启动服务

# yum install iscsi-initiator-utils -y

1. 设置本机的iqn名称

# vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2018-01.cn.tedu:client1

1. 发现远程target存储

# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.5 --discover

# systemctl restart iscsi

1. 查看iscsi的加载是否成功

# lsblk

出现sda盘

1. 对sda盘进行分区和格式化,并挂载进行使用

# parted /dev/sda mklabel gpt

# parted /dev/sda mkpart 1 100%

# lsblk

/\*\* sda1出现 \*\*/

# mkfs.xfs /dev/sda1

# mount /dev/sda1 /mnt

1. 附加课外实验：多台FTP或者http主机使用共享存储
2. 这里以FTP为例，web1和web2主机都安装vsftpd软件，使用统一的后端共享存储设备

web1(ftp) /var/ftp

iscsi -- sda

mount /sda1 /var/ftp

web2(http)

iscsi -- sda

mount /dev/sda1 /var/www/html

iscsi块共享:

不能同时多人使用

1. 修改vsftpd配置文件，开启匿名上传功能。将下面2行默认的注释行打开。

#anon\_upload\_enable=YES

#anon\_mkdir\_write\_enable=YES

1. 部署multipath多路径环境
2. 通过multipath,实现以下目标

· 在共享存储服务器上配置ISCSI,为应用服务器共享存储空间

· 应用服务器上配置iSCSI，发现远程共享存储

· 应用服务器上配置Multipath，将相同的共享存储映射为同一个名称

web1(iscsi客户端)

eth0

eth1

双网卡访问

· proxy(iscsi服务端)

1. client上操作

# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.5:3260 --discover

192.168.2.5:3260,1 iqn.2018-01.cn.tedu:server1

# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5:3260 --discover

192.168.2.5:3260,1 iqn.2018-01.cn.tedu:server1

# systemctl restart iscsi

# lsblk

sda 8:0 0 20G 0 disk

└─sda1 8:1 0 20G 0 part

sdb 8:16 0 20G 0 disk

└─sdb1 8:17 0 20G 0 part

1. 使用多路径将两个硬盘合并为一个
2. 安装多路径软件包

# yum install device-mapper-multipath -y

1. 生成配置文件

# cd /usr/share/doc/device-mapper-multipath-0.4.9/

# cp multipath.conf /etc/

1. 查看两块硬盘的id号

# /usr/lib/udev/scsi\_id --whitelisted --device=/dev/sda

3600140541fc17254192472bae0438e5e

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

3600140541fc17254192472bae0438e5e

1. 修改配置文件

# vim /etc/multipath.conf

跳转至60行

multipaths {

multipath {

wwid 3600140541fc17254192472bae0438e5e

alias my\_scsi

}

}

1. 查看多路径设备是否正常

# multipath -ll

# multipath -rr

my\_scsi (3600140541fc17254192472bae0438e5e) dm-2 LIO-ORG ,store

size=20G features='0' hwhandler='0' wp=rw

|-+- policy='service-time 0' prio=1 status=active

| `- 2:0:0:0 sda 8:0 active ready running

`-+- policy='service-time 0' prio=1 status=enabled

`- 3:0:0:0 sdb 8:16 active ready running

· 关闭proxy的eth1网卡

[root@proxy ~]# ifdown eth1

# multipath -ll

my\_scsi (3600140541fc17254192472bae0438e5e) dm-2 LIO-ORG ,store

size=20G features='0' hwhandler='0' wp=rw

|-+- policy='service-time 0' prio=1 status=active

| `- 2:0:0:0 sda 8:0 active ready running

`-+- policy='service-time 0' prio=1 status=enabled

`- 3:0:0:0 sdb 8:16 failed ready running

· 挂载特殊硬盘my\_scsi

# mount /dev/mapper/my\_scsi1 /mnt

1. 为了防止和ceph存储冲突,需要取消当前实验

1.umount

# umount /mnt

2.stop multipath.service

# systemctl stop multipathd.service

3.logout

# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.4.5:3260 --logout

# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.2.5:3260 --logout

1. 配置并访问NFS共享
2. 服务器利用NFS机制发布2个共享目录，要求如下：

· 将目录/root共享给192.168.2.100，客户机的root用户有权限写入

· 将/usr/src目录共享给192.168.2.0/24网段，只开放读取权限

1. 配置proxy的nfs文件

# vim /etc/exports

/root 192.168.2.100(rw,no\_root\_squash)

/usr/src 192.168.2.0/24(ro)

/\*\*

\* 用root访问nfs时,默认会被降级,需要在选项中增加no\_root\_squash

\*\*/

1. 客户端查看

# showmount -e 192.168.2.5

/usr/src 192.168.2.0/24

/root 192.168.2.100

1. NFS依赖服务

# systemctl restart rpcbind;systemctl enabled rpcbind

/\*\*

\* NFS:随机端口 如1098

\* rpcbind:111端口

\* 注册端口:

\* nfs:1098

\*\*/

1. 编写udev规则
2. 编写udev规则，实现以下目标：

· 当插入一个U盘时，该U盘自动出现一个链接称为udisk

· U盘上的第1个分区名称为udisk1，以此类推

· 终端上出现提示信息”udisk plugged in”

1. udev(2.6) 3.10

udev设备管理

所有设备都存放在/dev/目录(静态)

udev动态管理

udev动态设备管理/sys /dev

对于Linux kernel 2.6及更新的操作系统版本会将设备的相关信息动态写入/sys文件系统中，而udev程序可以通过读取这些设备系信息，并根据自己的udev规则进行设备管理器，实现如下功能:

· 处理设备命名 ENV{设备属性}=="属性值" NAME="定义设备名称"

· 决定要创建哪些设备文件或链接 SYMLINK+="链接"

· 决定如何设置属性 MODE="权限" OWNER="属主" GROUP="属组"

· 决定触发哪些事件 RUN="命令"

/\*\*

\* ==判断 !=

\* =赋值

\* +=原有基础上的赋值

\* RUN+=

\* :=不允许被替换的规则

\* 判断条件1,判断条件2,判断条件3

\* ENV{品牌}==xx,ENV{序列号}==xx,ENV{条件}==值,RUN+=

\* KERNEL=="sd[a-z]1" 判断设备的内核名称

\*\*/

udev默认规则存放在/etc/udev/rules.d目录下，通过修改此目录下的规则实现设备的命名、属性、链接文件等。

1）查看设备属性

加载USB设备的同时实时查看设备的相关属性，可以使用monitor指令。

# udevadm monitor --property

ACTION=add

如果设备已经加载则无法使用monitor查看相关属性。可以使用下面的命令查看设备属性。

# udevadm info --query=path –name=/dev/sdb

# udevadm info --query=property –path=/block/sdb

# vim /etc/udev/rules.d/70-usb.rules

ENV{ID\_VENDOR}=="JMicron",ACTION=="add",RUN+="/usr/bin/systemctl start httpd"

ENV{ID\_VENDOR}=="JMicron",ACTION=="remove",RUN+="/usr/bin/systemctl stop httpd"

2. Day 02 集群及LVS简介 LVS-NAT集群 LVS-DR集群
3. 集群及LVS简介
4. 集群简介

1) 什么是集群

· 高速网互联

· 多服务器集中起来提供一种服务,在客户端看来就像是只有一个服务器

· 节省成本,低成本下获得在性能,可靠性和灵活性方面的相对较高的收益

· 任务调度时集群系统中的核心技术

2) 集群目的

· 提高性能

· 降低成本

· 提高扩展性

· 增强可靠性

3) 集群分类

· 高性能计算集群HPC

- 通过以集群开发的并行应用程序,解决复杂的科学问题

· 负载均衡(LB)集群

- 客户端负载在计算机集群中尽可能平均分摊

· 高可用(HA)集群

- 避免单点故障,当一个系统发生故障时,可以快速迁移

1. LVS概述

1) LVS项目介绍

· Linux虚拟服务器(LVS)是章文嵩在国防科大就读博士期间创建的

· LVs可以实现高可用的,可伸缩的Web,Mail,Cache和Media等网络服务

· 最终目标时利用Linux操作系统和LVS集群软件实现一个高可用,高性能,低成本的服务器应用集群

/\*\*

\* LVS:性能高,功能少,不支持正则

\* nginx:性能中,功能多,正则

\*\*/

1. LVS集群组成

· 前端:负载均衡层

- 由一台或多台负载调度器构成

· 中间:服务器群组层

- 由一组实际运行应用服务的服务器组成

· 底端:数据共享存储层

- 提供共享存储空间的存储区域

/\*\*

\* 用keeplived来保证LVS的高可用

\*\*/

3) 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

- 真正提供应用服务的服务器

· VIP[virtual IP]:虚拟IP地址

- Director Server公布给用户访问的虚拟IP地址

· RIP[real IP]:真实IP地址

- 集群节点[Real Server]上使用的IP地址

· DIP:调度器连接节点服务器的IP地址

· CIP[Client IP]:客户端IP

4) LVS工作模式

· NAT模式

- 通过网络地址转换实现的虚拟服务器;

- 用户通过互联网访问VIP,LVS可以看作一个路由器;

- 要求从哪里进,从哪里出;

- 与路由器的区别:路由器只是单纯把内网服务器发布到公网,而LVS可以实现负载均衡;

- 缺点:LVS的负荷高,调度器性能会成为整个集群的瓶颈;

- 适合数据量小的集群环境;

· TUN[隧道]模式

- 用户访问北京的Director Server;

- Real Server在上海;

- 通过LVS转发请求寻找Real Server服务;

· DR模式

- 直接使用路由技术实现虚拟服务器;

- 路由器将数据转给LVS,LVS转给服务器;

- 服务器将处理结果通过路由器传输给用户;

- 适合数据量较大的集群环境;

- 节点服务器需要配置VIP,注意MAC地址广播;

5) 负载均衡调度算法

· LVS目前实现了10种调度算法

· 常用调度算法有4种

- 轮询(Round Robin)

\* 将客户端请求平均分发到Real Server

- 加权轮询(Weight Round Robin)

\* 根据Real Server权重值进行轮询调度

- 最少连接(Least Connections)

\* 选择连接数最少的服务器

- 加权最少连接(Weight Least Connections)

\* 根据Real Server权重值,选择连接数最少的服务器

- 源地址散列(Source Hashing)

\* 根据请求的目标IP地址,作为散列键(Hash Key)从静态分配的散列表找出对应的服务器

- 其他调度算法

\* 给予局部性的最少链接

\* 带复制的基于局部性最少链接

\* 目标地址的三列(Destination Hashing)

\* 最短的期望的延迟

\* 最少队列调度

1. LVS-NAT集群
2. 软件安装

1) 安装前准备

· LVS的IP负载均衡技术是通过IPVS模块实现的

· IPVS模块已成为Linux组成部分

2) 安装ipvsadm

3) ipvsadm用法

ipvsadm -A 添加虚拟服务器

ipvsadm -E 修改虚拟服务器

ipvsadm -D 删除虚拟服务器

ipvsadm -C 清空所有

ipvsadm -a 添加真实服务器

ipvsadm -e 修改真实服务器

ipvsadm -d 删除真实服务器

ipvsadm -L 查看

-s [rr|wrr|lc|wlc] 指定负载调度算法

· 创建虚拟服务器

# ipvsadm -A -t|u 192.168.4.5:80 -s [算法]

-A 添加虚拟服务器

-t 协议为tcp

-u 协议为udp

-s 执行负载调度算法

· 添加/删除服务器节点

# ipvsadm -a -t|u 192.168.4.5:80 -r 192.168.2.100 [-g|i|m] [-w 权重]

-a 添加真实服务器

-d 删除真实服务器

-r 指定真实服务器的地址

-m 使用NAT模式

-g 使用DR模式 /\* 默认模式 \*/

-i 使用TUN模式

-w 为节点服务器设置权重,,默认为1

· 查看IPVS

# ipvsadm -Ln

· 永久保存所有规则

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

# cat /etc/sysconfig/ipvsadm

· 清除无关规则

# ipvsadm -C

# rm -f /etc/sysconfig/ipvsadm

Example:

/\*\* ipvsadm选项用法

\* -A添加集群

\* -a添加服务器

\*\*/

# ipvsadm -A -t 192.168.4.5:80 -s rr

# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 rr

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80

# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 rr

-> 192.168.2.100:80 Route 1 0 0

1. LVS-NAT案例

· 使用LVS实现NAT模式的集群调度服务器，为用户提供Web服务：

\* 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

· 机器配备:

\* client 192.168.4.10/24

\* proxy 192.168.4.5/24 192.168.2.5/24

\* web1 192.168.2.100/24

\* web2 192.168.2.200/24

/\*\* 注意: web1 和 web2 的eth0网卡需要关闭 \*\*/

\* 服务器节点配置(web1/2):

1) 设置Web服务器

2) 启动Web服务

3) 关闭防火墙与SELinux

# systmctl stop firewalld

# setenforce 0

\* 调度服务器配置(proxy)

1) 确认调度器的路由转发功能

# echo 1 > /proc/sys/net/ipv4/ip\_forward

# cat /proc/sys/net/ipv4/ip\_forward

1

/\*\* 上面为临时配置 \*\*/

# echo "net.ipv4.ip\_forward = 1" >> /etc/sysctl.conf

/\*\* 这个是永久配置,可以用sysctl -p命令查看规则 \*\*/

2) 创建集群服务器

# yum -y install ipvsadm

# ipvsadm -A -t 192.168.4.5:80 -s wrr

3) 添加真实服务器

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80 -w 1 -m

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200:80 -w 1 -m

4) 查看规则列表,并保存规则

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

\* 客户端测试(client)

1) 利用自带的curl浏览器反复访问Dicretor Server(proxy)

# curl http://192.168.4.5

1. 部署LVS-DR集群
2. 问题描述

· 使用LVS实现DR模式的集群调度服务器，为用户提供Web服务：

\* 客户端IP地址为192.168.4.10

\* LVS调度器VIP地址为192.168.4.15

\* LVS调度器DIP地址设置为192.168.4.5

\* 真实Web服务器地址分别为192.168.4.100、192.168.4.200

\* 使用加权轮询调度算法，web1的权重为1，web2的权重为2

\* 说明：

\* 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；

1. 方案

· 使用4台虚拟机,1台作为Director调度器,2台作为Real Server

\* client eth0(192.168.4.10/24)

\* proxy eth0(192.168.4.5/24) eth0:0(192.168.4.15/24)

\* web1 eth0(192.168.4.100/24) lo:0(192.168.4.15/32)

\* web2 eth0(192.168.4.200/24) lo:0(192.168.4.15/32)

/\*\*

\* 注意:

\* 1.web1/web2的子网掩码必须是32位

\* 2.VIP必须配在虚拟网卡上

\*\*/

\* 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）。

· proxy配置:

# cp ifcfg-eth0 ifcfg-eth0:0

# 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

# systemctl restart network

· web1配置:

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

# vim /etc/sysconfig/network-scripts/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。

# 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

\* 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

# sysctl -p

/\*\* 重启网络服务，设置防火墙与SELinux \*\*/

· proxy部署调度器:

# yum -y install ipvsadm

# ipvsadm -C

/\*\* 1.清空规则 \*\*/

# ipvsadm -A -t 192.168.4.15:80 -s wrr

/\*\* 2.添加集群服务器 \*\*/

# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100 -g -w 1

# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200 -g -w 2

/\*\* 3.添加Real Server \*\*/

# ipvsadm -Ln

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

/\*\* 4.查看规则并保存 \*\*/

· 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)

# 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/null

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

1. Day 03 Keepalived热备 Keepalived+LVS 、 HAProxy服务器
2. nginx与LVS调度器区别
3. nginx代理(我帮你去访问)
4. LVS调度(转发数据包) LVS-NAT LVS-DR LVS-TUN

# yum install ipvsadm -y

# ipvsadm -A -t|u VIP:port -s wrr

# ipvsadm -a -t|u VIP:port -r RIP [-g|m|i]

# ipvsadm -E|D|C /\*\* VIP \*\*/

# ipvsadm -e|d /\*\* RIP \*\*/

# ipvsadm -Ln /\*\* 查看 \*\*/

\* NAT转发一定要有网关,且RIP不能和client在同网段中

/\*\*

\* Keepalived是给LVS写的

\* 1.自动配置LVS规则,做健康检查

\* 2.Keepalived学习了路由器上的功能VRRP HSRP路由热备

\*\*/

1. Keepalived热备
2. Keepalived概述
3. Keepalived概述

· 调度器出现单点故障,如何解决?

· Keepalived实现了高可用集群

· Keepalived最初是为LVS设计的,专门监控各服务器节点的状态

· Keepalived后来加入了VRRP功能,防止单点故障

1. Keepalived运行原理

· Keepalived检测每个服务器节点状态

· 服务器节点异常或工作出现故障,Keepalived将故障节点从集群系统中剔除

· 故障节点恢复后,Keepalived再将其加入到集群系统中

· 所有工作自动完成,无需人工干预

1. Keepalived服务

/\*\*

\* 准备三台 Linux服务器,两台做Web服务器,并部署Keepalived高可用软件,一台作为客户端

\* 使用Keepalived实现web服务器的高可用

\* proxy服务器的IP地址为192.168.4.5

\* web服务器IP地址分别为192.168.4.100和192.168.4.200

\* web服务器的浮动VIP为192.168.4.80

\* 客户端通过访问VIP地址访问web页面

\*\*/

1. Keepalived安装

/\*\* web1和web2 \*\*/

# yum install keepalived

/\*\* 装包 \*\*/

# 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 50 //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（实验需要修改）

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主备服务器密码必须一致

}

virtual\_ipaddress { 192.168.4.80 } //谁是主服务器谁获得该VIP（实验需要修改）

}

/\*\* 配置 \*\*/

# systemctl restart keepalived && iptables -F

/\*\* 起服务 \*\*/

\* 当web1的keepalived服务停止时,192.168.4.80会自动切换到web2上

/\*\* 注意

\* keepalived只监控服务本身,当网卡接口停止时,需要停止keepalived服务VIP才会切换

\*\*/

1. Keepalived+LVS服务器
2. 问题

1) 使用Keepalived为LVS调度器提供高可用功能,防止调度器单点故障,为用户提供Web服务

· LVS1调度器真实IP地址为192.168.4.5

· LVS2调度器真实IP地址为192.168.4.6

· 服务器VIP地址设置为192.168.4.100,192.168.4.200

· 使用加权轮询调度算法,真实iweb服务器权重不同

2) 机器配备:

client eth0(192.168.4.10/24)

proxy1 eth0(192.168.4.5/24)

proxy2 eth0(192.168.4.6/24)

web1 eth0(192.168.4.100/24)

web2 eth0(192.168.4.200/24)

1. 操作
2. 设置web1服务器的网络参数
3. 为web1配置VIP地址lo:0(子网掩码必须是32)
4. 重启网络服务,设置防火墙与SELinux
5. 给web2重复上面操作
6. 配置proxy1与proxy2网络参数(VIP由keepalived自动配置)
7. 调度器安装Keepalived和ipvsadm
8. 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

/\*\* 上接VIP \*\*/

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

}

}

}

/\*\* 健康检查种类 \*\*/

real server {

TCP\_CHECK{

connect\_timeout 3 /\*\* 超时时间 \*\*/

nb\_get\_retry 3 /\*\* 尝试3次 \*\*/

delay\_before\_retry 3 /\*\* 每3秒重试一次 \*\*/

} /\*\* 以端口可用性为依据 \*\*/

HTTP\_CHECK{} /\*\* 以网页为依据 \*\*/

SSL\_GET{

path /

digest md5sum /\*\* 以md5码为验证 \*\*/

} /\*\* 以网页为依据 \*\*/

}

1. 配置HAProxy负载均衡集群
2. 集群调度软件对比
3. Nginx分析

· 优点：

- 工作在7层，可以针对http做分流策略

- 正则表达式比HAProxy强大

- 安装、配置、测试简单，通过日志可以解决多数问题

- 并发量可以达到几万次

- Nginx还可以作为Web服务器使用

· 缺点：

- 仅支持http、https、mail协议，应用面小

- 监控检查仅通过端口，无法使用url检查

1. LVS分析

· 优点：

- 负载能力强，工作在4层，堆内存、CPU消耗低

- 配置性低，没有太多可配置性，减少人为错误

- 应用面广，几乎可以为所有应用提供负载均衡

· 缺点：

- 不支持正则表达式，不能实现动静分离

- 如果网站架构庞大，LVS-DR配置比较繁琐

1. HAProxy分析

· 优点：

- 支持session、cookie功能

- 可以通过url进行健康检查

- 效率、负载均衡速度高于Nginx，低于LVS

- HAProxy支持TCP，可以对MySQL进行负载均衡

- 调度算法丰富

· 缺点：

- 正则弱于Nginx

- 日志依赖于syslogd，不支持apache日志

1. 其他调度器

· F5 big-ip

速度:F5 > LVS > HAProxy > Nginx

/\*

\* 总结：

\* LVS 4层调度,不支持7层

\* HAProxy 有一定的功能,速度还不错,对正则的支持不如Nginx

\* HAProxy既能做4层调度,也能做7层调度

\* Nginx 4,7层调度器，4.9以后支持4层调度,但还不够成熟

\*/

1. 准备4台Linux服务器,两台做web服务器,1台安装HAproxy,1台做客户端,实现以下功能

1) 客户端访问HAProxy,HAProxy分发请求到后端Real Server

2) 开启HAProxy监控页面,及时查看调度器状态

3) 设置HAProxy为开机启动

1. 清除前面的实验环境

· web1/web2:

# rm -f network/ifcfg-lo:0

# vim /etc/sysctl.conf

# ifdown eth0

# systemctl restart network

· proxy1/proxy2

# systemctl stop keepalived.service

1. 配置后端服务器

· 配置web1/2的httpd

· 在proxy1上安装HAProxy

# yum install haproxy -y

· 修改配置文件

# 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

bind 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 #统计页面密码框上提示文本

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

#对后端服务器的健康状况检查间隔为2000毫秒，连续2次健康检查成功，则认为是有效的，连续3次健康检查失败，则认为服务器宕机

参数解析

global /\*\* 全局设置 \*\*/

maxconn=2000 [全局并发量]

defaults /\*\* 默认设置 \*\*/

maxconn=1000 [默认并发量]

listen集群 /\*\* 单集群设置 \*\*/

maxconn=500 [单集群并发]

1. 两种方式定义集群:

frontend+backend

listen

listen xxx \*:80

server ip1

server ip2

server web1 192.168.2.100:80 check inter 2000 rise 2 fall 5 /\*\* 健康检查:5次算失败,每20s再次检查,成功两次后放回集群 \*\*/

listen status 0.0.0.0:1080

stats refresh 30s

stats uri /stats

stats realm HAProxy Manager

stats auth admin:admin

帮助文档

# ls /usr/share/doc/haproxy-1.5.18/configuration.txt

1. Day 04 Ceph概述 部署Ceph集群 Ceph块存储
2. Ceph概述
3. 基础知识
4. 什么是分布式文件系统

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

· 分布式文件系统的设计基于客户机/服务器模式

/\*\*

\* 常见分布式网站架构

\* 1.用户访问调度器

\* 2.后台有一组应用服务器，调度器将请求转发给服务器

\* 3.服务器下有各自的页面文件/目录

\* 4.可以在后台做一台共享存储NFS/Sumba，服务器mount此目录

\* 5.当文件过大，NFS无法支持的时候，网站会没法继续做下去

\* 6.还有一台数据库为应用服务器提供数据

\* 7.数据库和共享存储统称为数据端

\*\*/

1. 常用分布式文件系统

· Lustre

· Hadoop

· FastDES

· Ceph

· GlusterFS

1. 什么是Ceph

· Ceph是一个分布式文件系统

· 具有高扩展、高可用、高性能的特点

· Ceph可以提供对象存储、块存储、文件系统存储

/\*\*

\* 对象存储，也叫做基于对象的存储，是用来描述解决和处理离散单元的方法的通用术语，这些离散单元被称作为对象。

\* 就像文件一样，对象包含数据，但是和文件不同的是，对象在一个层结构中不会再有层级结构。每个对象都在一个被称作存储池的扁平地址空间的同一级别里，一个对象不会属于另一个对象的下一级。

\* 文件和对象都有与它们所包含的数据相关的元数据，但是对象是以扩展元数据为特征的。每个对象都被分配一个唯一的标识符，允许一个服务器或者最终用户来检索对象，而不必知道数据的物理地址。这种方法对于在云计算环境中自动化和简化数据存储有帮助。

\*\*/

· Ceph可以提供PB级别的存储空间(PB>TB>GB)

- 1024G\*1024G=1048576G

· 软件定义存储(Software Defined Storage)作为存储行业的一大发展趋势，已经越来越上受到市场的认可

· 帮助文档：<http://docs.ceph.org/start/intro>

1. Ceph组件

· OSDs

- 存储设备

· Monitors

- 集群监控组件

- 至少三台启动，否则集群报错

· RGW

- 对象存储网关

· MDSs

- 存放文件系统的元数据(对象存储和块存储不需要该组件)

· Client

- ceph客户端

iscsi[块]

NFS[文件系统]

· Ceph提供：块，文件系统，对象存储(百度云)

- ceph-osd ceph-mon

\* 必装，用户只能以块的方式访问

- ceph-mds

\* 可以以文件系统方式访问，mount挂载

- ceph-radosgw (RGW，radosgateway)

\* 可以以对象存储的方式访问

/\*\*

\* 分布式存储的算法

\* 以取余算法为例：

\* node1:10T node2:10T node3:10T node4:10T 4台计算机为后端存储OSD，提供真正的存储磁盘

\* Monitor[监控/管理]：算法的执行者，用户访问数据前需要先访问Monitor

\* 用户访问a.txt|md5sum=16进制数%4=1 ==> 存入node2

\* b.txt|md5sum=16进制数%4=0 ==> 存入node1

\* Ceph用的是CRUSH算法

\*\*/

/\*\*

\* 分布式存储的特点

\* 1.数据被分开存储

\* 2.数据的读写是并行的

\* 3.数据会像raid一样，单个文件会被打散

\* 4.Ceph所有数据是3副本

\*\*/

1. 实验环境准备
2. 安装前准备
3. 物理机为所有节点配置yum源服务器

[root@room9pc01 ~]# yum -y install vsftpd

[root@room9pc01 ~]# mkdir /var/ftp/ceph

[root@room9pc01 ~]# mount -o loop \

rhcs2.0-rhosp9-20161113-x86\_64.iso /var/ftp/ceph

[root@room9pc01 ~]# systemctl restart vsftpd

1. 修改所有节点都需要配置YUM源（这里仅以node1为例）

[root@node1 ~]# cat /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/MON

gpgcheck=0

[osd]

name=osd

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

gpgcheck=0

[tools]

name=tools

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

gpgcheck=0

1. 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

1. 配置无密码连接(包括自己远程自己也不需要密码)

[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

1. 配置NTP时间同步
2. 真实物理机创建NTP服务器

[root@room9pc01 ~]# yum -y install chrony

[root@client ~]# cat /etc/chrony.conf

server 0.centos.pool.ntp.org iburst

allow 192.168.4.0/24

local stratum 10

[root@room9pc01 ~]# systemctl restart chronyd

如果有防火墙规则，需要清空所有规则

[root@room9pc01 ~]# iptables -F

1. 其他所有节点与NTP服务器同步时间（以node1为例）

[root@node1 ~]# cat /etc/chrony.conf

server 192.168.4.254 iburst

[root@node1 ~]# systemctl restart chronyd

1. 准备存储磁盘
2. 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

1. 部署Ceph集群
2. 准备部署环境
3. 安装部署软件

· 使用node1作为部署主机

# yum -y install ceph-deploy

· ceph-deploy命令与子命令都支持--help查看帮助

# ceph-deploy --help

1. 创建目录

· 为部署工具创建目录，存放秘钥与配置文件

# mkdir ceph-cluster

# cd ceph-cluster/

1. 部署存储集群
2. 创建Ceph集群
3. 创建Ceph集群配置(所有节点都为mon)

[root@node1 ceph-cluster]# ceph-deploy new node1 node2 node3

1. 给所有节点安装Ceph软件包

[root@node1 ceph-cluster]# ceph-deploy install node1 node2 node3

1. 初始化所有节点的mon服务(主机名解析必须对)

[root@node1 ceph-cluster]# ceph-deploy mon create-initial

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

/\*\*

\* 常见错误和解决方法：

\* [node1][ERROR ] admin\_socket: exception getting command descriptions: [Error 2] No such file or directory

\* 先检查自己的命令是否是在ceph-cluster目录下执行的！！！！如果时确认是在该目录下执行的create-initial命令，依然保存，可以使用如下方式修复

\* [root@node1 ceph-cluster]# vim ceph.conf #文件最后追加以下内容

public\_network = 192.168.4.0/24

\* 修改后重新推送配置文件:

\* [root@node1 ceph-cluster]# ceph-deploy --overwrite-conf config push node1 node2 node3

\*\*/

1. 初始化完成后，使用ceph -s与systemctl查看状态

# systemctl status ceph-<tab>

[ceph-create-keys@node1.service](mailto:ceph-create-keys@node1.service) ceph-mon.target

ceph-mds.target ceph-osd.target

ceph-mon@node1.service ceph-radosgw.target

1. 创建OSD
2. 给所有节点准备磁盘分区

[root@node1 ~]# parted /dev/vdb mklabel gpt

[root@node1 ~]# parted /dev/vdb mkpart primary 1M 50%

[root@node1 ~]# parted /dev/vdb mkpart primary 50% 100%

[root@node1 ~]# lsblk

[root@node1 ~]# chown ceph.ceph /dev/vdb1

[root@node1 ~]# chown ceph.ceph /dev/vdb2

[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"

/\*\*

\* 我们都爱for循环

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

ssh $i 'parted /dev/vdb mklabel gpt;

parted /dev/vdb mkpart primary 1M 50%;

parted /dev/vdb mkpart primary 50% 100%;

chown ceph.ceph /dev/vdb1;

chown ceph.ceph /dev/vdb2';

ssh $i '

cat > /etc/udev/rules.d/70-vdb.rules << EOF

ENV{DEVNAME}=="/dev/vdb1",OWNER="ceph",GROUP="ceph"

ENV{DEVNAME}=="/dev/vdb2",OWNER="ceph",GROUP="ceph"

EOF'; done

\*\*/

1. 初始化清空磁盘数据

[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

/\*\*

\* 我们都爱for循环

\* [root@node1 ceph-cluster]# for i in node{1..3};do ceph-deploy disk zap $i:vd{c,d};done

\*\*/

1. 创建OSD存储空间

[root@node1 ceph-cluster]# ceph-deploy osd create \

> node1:vdc:/dev/vdb1 node1:vdd:/dev/vdb2

[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

//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]# for i in node{1..3};do ceph-deploy osd create $i:vdc:/dev/vdb1 $i:vdd:/dev/vdb2;done

\*\*/

/\*\*

\* vdb1和vdb2用来做存储服务器的日志journal盘

\* vdb1做vdc的缓存，vdb2做vdd缓存

\* 缓存盘对速度的要求较高，因此最好是用固态硬盘，而ceph做在传统磁盘上

\*\*/

1. 验证

· 查看集群状态

[root@node1 ~]# ceph -s

/\*\*

\* 常见错误：

health: HEALTH\_WARN

clock skew detected on node2, node3…

表示时间不同步，将所有主机时间同步即可

Ceph要求所有主机时间差不能超过0.05s,否则就会提示WARN，如果使用NTP还不能精确同步时间，可以手动修改所有主机的ceph.conf，在[MON]下面添加如下一行：

mon clock drift allowed = 1

如果状态还是失败，可以尝试执行如下命令，重启ceph服务：

[root@node1 ~]# systemctl restart ceph\\*.service ceph\\*.target

\* 错误2：

权限配置完毕而ceph -s 显示EOR，可能是因为磁盘信息没有刷新

# partprobe

状态根据ceph主机数量和存储容量大小来确定

HEALTH状态会有延迟

\*\*/

1. Ceph块存储
2. 概述
3. 什么是块存储

· 单机块设备

- 光盘

- 磁盘

· 分布式块存储

- Ceph

- Cinder

· Ceph块设备也叫做RADOS块设备

- RADOS block device：RBD

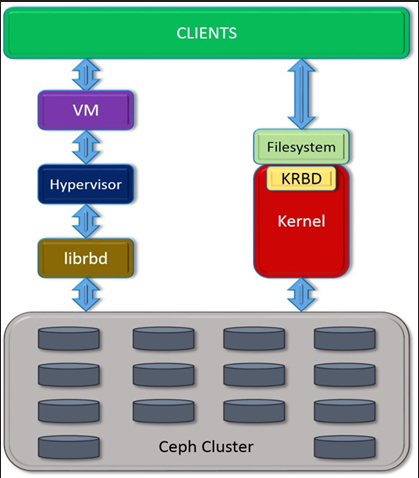
· RDB驱动已经很好的集成在了Linux内核中

· RBD提供了企业功能，如快照、COW克隆等

· RBD还支持内存缓存，从而能够大大提高性能

· Linux内核可用直接访问Ceph块存储

· KVM可用借助于librbd访问



图例1

1. 块存储集群
2. 创建镜像

· 查看存储池(默认有一个rbd池)

[root@node1 ~]# ceph osd lspools

· 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

/\*\* image-feature支持哪些功能; layering 支持分层快照功能 \*\*/

/\*\* 帮助文档 rbd help create \*\*/

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info demo-image

1. 动态调整大小

· 缩小容量

[root@node1 ~]# rbd resize --size 7G image --allow-shrink

[root@node1 ~]# rbd info image

· 扩容容量

[root@node1 ~]# rbd resize --size 15G image

[root@node1 ~]# rbd info image

/\*\*

\* 三台ceph-mon 三台ceph-osd

\* rbd create image 创建镜像

\*\*/

1. 集群内通过KRBD访问

· 集群内将镜像映射为本地磁盘

[root@node1 ~]# rbd map demo-image

/dev/rbd0

[root@node1 ~]# lsblk

… …

rbd0 251:0 0 10G 0 disk

[root@node1 ~]# mkfs.xfs /dev/rbd0

[root@node1 ~]# mount /dev/rbd0 /mnt

1. 客户端通过KRBD访问
2. 客户端需要安装ceph-common软件包
3. 拷贝配置文件(否则不知道集群在哪)
4. 拷贝连接密钥(否则无连接权限)

[root@client ~]# yum install ceph-common -y

[root@node1 ceph]# scp ceph.conf client:/etc/ceph/

[root@node1 ceph]# scp ceph.client.admin.keyring client:/etc/ceph/

1. 将镜像映射为本地磁盘

[root@client ~]# rbd map image

[root@client ~]# lsblk

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

1. 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

1. 创建镜像快照

/\*\* 快照 COW(Copy on Write)写时复制 \*\*/

/\*\* 快照使用写时复制技术，对大数据快照速度会很快

\* 原始盘100G

\* a.txt:111--XXX 1G

\* b.txt:222

\* 快照备份50G

\* a.txt:111

\* b.txt:222

\* 10T数据库 先快照 --> 后期再cp和tar

\* Ceph支持快照，但不支持在线快照，在线快照会使Ceph盘崩溃

\*\*/

1. 查看镜像快照

[root@node1 ~]# rbd snap ls image

1. 创建镜像快照

[root@node1 ~]# rbd snap create image --snap image-snap1

[root@node1 ~]# rbd snap ls image

SNAPID NAME SIZE

4 image-snap1 15360 MB

1. 删除客户端写入的测试文件

[root@client ~]# rm -rf /mnt/test.txt

1. 使用快照恢复数据
2. 还原快照

[root@node1 ~]# rbd snap rollback image --snap image-snap1

1. 客户端重新挂载分区

[root@client ~]# umount /mnt

[root@client ~]# mount /dev/rbd0 /mnt/

/\*\* 在线还原会导致坏块，此时挂载会报错，需要重新回滚一次 \*\*/

[root@node1 ~]# rbd snap rollback image --snap image-snap1

Rolling back to snapshot: 100% complete...done.

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# ls /mnt

1. 快照克隆

· 如果想从快照恢复出来一个新的镜像，则可以使用克隆

· 注意，克隆前，需要对快照进行<保护>操作

· 被保护的快照无法删除，取消保护(unprotect)

[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image-clone镜像

· 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.d36a3d1b58ba

format: 2

features: layering

flags:

parent: rbd/image@image-snap1

overlap: 15360 MB

· 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，比较耗时

[root@node1 ~]# rbd flatten image-clone

Image flatten: 100% complete...done.

[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.d36a3d1b58ba

format: 2

features: layering

flags:

//注意，父快照信息没了

1. 客户端撤销磁盘映射
2. umount挂载点

[root@client /]# umount /mnt/

1. 取消RBD磁盘映射

[root@client /]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

/\*\*

\* 语法格式：

\* rbd unmap /dev/rbd/{poolname}/{imagename}

\*\*/

[root@client /]# rbd unmap /dev/rbd/rbd/image

[root@client /]# rbd showmapped

1. 删除快照与镜像

· 删除快照

[root@node1 ~]# rbd snap rm image --snap image-snap1

rbd: snapshot 'image-snap1' is protected from removal.

2019-03-08 09:48:09.492277 7fd031781d80 -1 librbd::Operations: snapshot is protected

[root@node1 ~]# rbd snap unprotect image --snap image-snap1

[root@node1 ~]# rbd snap rm image --snap image-snap1

· 删除镜像

[root@node1 ~]# rbd list

demo-image

image

image-clone

[root@node1 ~]# rbd rm image

Removing image: 100% complete...done.

[root@node1 ~]# rbd list

1. Day 05 块存储应用案例 分布式文件系统 对象存储
2. 块存储应用案例
3. 准备实验环境
4. 创建磁盘镜像

KVM:

/var/lib/libvirt/images/镜像文件

/etc/libvirt/qemu/XML文件

cp xx.qcow2 new.qcow2

cp xx.xml new.xml

# cd /var/lib/libvirt/images/

# qemu-img create -f qcow2 -b .rh7\_template.img mycp.img 10G

# cp .rhel7.xml /etc/libvirt/qemu/rhel7.xml

# vim /etc/libvirt/qemu/rhel7.xml

<name>mycp</name>

<source file='/var/lib/libvirt/images/mycp.img'/>

# virsh define /etc/libvirt/qemu/rhel7.xml

# virsh start mycp

1. 为虚拟机创建磁盘镜像

[root@node1 ~]# rbd create vm1-image --image-feature layering --size 10G

1. 查看镜像

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info vm1-image

[root@node1 ~]# qemu-img info rbd:rbd/vm1-image

image: rbd:rbd/vm1-image

file format: raw

virtual size: 10G (10737418240 bytes)

disk size: unavailable

1. Ceph认证账户

· Ceph默认开启用户认证，客户端需要账户才可以访问

- 默认账户名称为client.admin，key是账户的密钥

- 可以使用ceph auth添加新账户(案例中使用默认账户)

[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==

1. 部署客户端环境

· 注意：这里使用物理机当客户端

· 客户端需要安装ceph-common软件包

· 拷贝配置文件，否则不知道集群在哪

· 拷贝连接密钥，否则无连接权限

[root@yyzh\_pc ~]# yum install ceph-common -y

[root@yyzh\_pc ~]# ssh node1

[root@node1 ~]# scp /etc/ceph/ceph.conf /etc/ceph/ceph.client.admin.keyring 192.168.4.254:/etc/ceph/

[root@node1 ~]# exit

1. 创建KVM虚拟机
2. 创建初始化虚拟机

· 使用virt-manager创建2台普通的KVM虚拟机

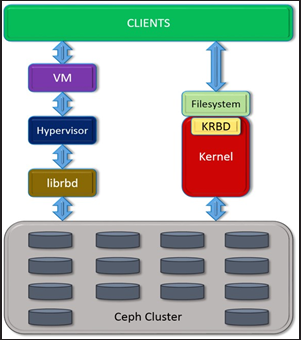
- 这里以1个虚拟机为例

1. 配置libvirt secret

· KVm虚拟机需要使用librbd才可以访问ceph集群

· Librbd访问ceph需要账户认证

· 所以这里需要给libvirt设置账户信息



原理图示

1. 编写账户信息文件(物理机操作)

[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>

[root@yyzh\_pc ~]# ceph auth get-key client.admin

AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

[root@yyzh\_pc ~]# cat /etc/ceph/ceph.client.admin.keyring

[client.admin]

key = AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

1. 使用XML配置文件创建secret

[root@room9pc01 ~]# virsh secret-define --file secret.xml

生成 secret 9877ff83-2ca6-48c1-982b-50a1def1bd45

//随机的UUID，这个UUID对应的有账户信息

1. 设置secret，添加账户密钥

[root@yyzh\_pc ~]# virsh secret-set-value \

> --secret 9877ff83-2ca6-48c1-982b-50a1def1bd45 \

> --base64 AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

secret 值设定

//这里secret后面是之前创建的secret的UUID

//base64后面是client.admin账户的密码

//现在secret中既有账户信息又有密钥信息

1. 虚拟机的XML配置文件

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2'/>

<source file='/var/lib/libvirt/images/mycp.img'/>

<target dev='vda' bus='virtio'/>

<boot order='1'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>

</disk>

1. 修改XML配置文件

# virsh edit mycp

<disk type='network' device='disk'>

<driver name='qemu' type='raw'/>

<auth username='admin'>

<secret type='ceph' uuid='9877ff83-2ca6-48c1-982b-50a1def1bd45'/>

</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='0x00' slot='0x07' function='0x0'/>

</disk>

/\*\* 硬盘接口可以是virtio,ide,sata盘中的任意一种，不可以是SCSI和USB \*\*/

· 关键词说明

<secret type='ceph' uuid='9877ff83-2ca6-48c1-982b-50a1def1bd45'/>

//这里的uuid就是secret的uuid，有client.admin账户和密语额信息

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

<host name='192.168.4.11' port='6789'/>

</source>

//这里说明使用账户连接哪台ceph主机和端口，访问哪个池和镜像

<target dev='vda' bus='virtio' />

//这里将获取的镜像设置为虚拟机的vda磁盘

1. 分布式文件系统
2. 基础知识
3. 什么是CephFS

/\*\* 什么是文件系统？NTFS,fat32,ext4,xfs文件系统 \*\*/

/\*\*

\* 文件系统格式化过程：

\* 1.inode索引+block

\* 2.4K对齐

\*

\* 索引中记录了权限，大小等以及对应文件的指针

\*\*/

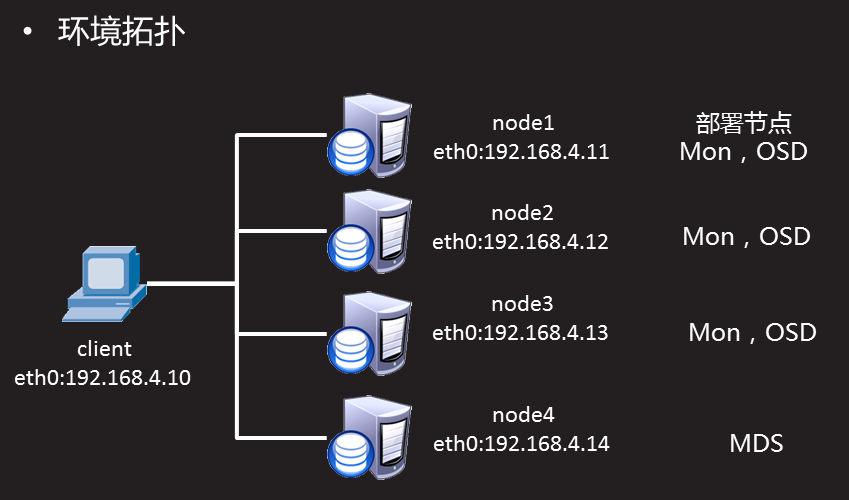
· 分布式文件系统(Distributed Fiile System)是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连

· CephFS使用Ceph集群提供与POSIX兼容的文件系统

· 允许Linux直接将Ceph存储mount到本地

1. 环境准备

· 环境拓扑



拓扑图例

1. 准备一台新的虚拟机，作为GW服务器
2. 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

- 修改node1的/etc/hosts，并同步到所有node主机

1. 元数据服务器
2. 什么是元数据

· 元数据(Metadata)

- 任何文件系统中的数据分为数据和元数据

- 数据是指普通文件中的实际依据

- 而元数据指用来描述一个文件的特征的系统数据

- 比如：访问权限、文件拥有者以及文件数据块的分步信息(inode...)等

- 所有CephFS必须有MDSs节点

1. 部署元数据服务器
2. 登录node4，安装ceph-mds软件包

[root@node4 ~]# yum install ceph-mds -y

1. 登录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node4拷贝配置文件，启动mds服务

1. 同步配置文件和key

[root@node1 ceph-cluster]# ceph-deploy admin node4

1. 实际操作

实际操作：

[root@node1 ceph-cluster]# ceph-deploy mds create node3

/\*\*

\* 实际生产环境中推荐新建一台计算机提高性能

\* 实验中mds做在了node3上

\*\*/

1. 文件系统服务器
2. 创建存储池

[root@node3 ~]# ceph osd pool create cephfs\_data 128

pool 'cephfs\_data' created

[root@node3 ~]# ceph osd pool create cephfs\_metadata 128

pool 'cephfs\_metadata' created

//创建两个存储池，对应128个PG

1. 创建Ceph文件系统

[root@node3 ~]# ceph fs new myfs1 cephfs\_metadata cephfs\_data

new fs with metadata pool 2 and data pool 1

//metadata作inode，data作block

//注意先写metadata再写data

//ceph默认只能创建1个文件系统，多余的会报错

1. 查看创建状态

[root@node3 ~]# ceph fs ls

name: myfs1, metadata pool: cephfs\_metadata, data pools: [cephfs\_data ]

[root@node3 ~]# ceph mds stat

e5: 1/1/1 up {0=node3=up:active}

1. 客户端挂载

[root@client ~]# mount -t ceph 192.168.4.11:6789:/ /media/ -o name=admin,secret=AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

//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

1. 对象存储
2. 概述

/\*\*

\* ceph 支持的存储方案：

\* 1.块(磁盘) osd

\* 2.文件系统 mds

\* 3.对象存储 rgw(radosgateway)

\*

\* 对象存储必须开发人员写程序

\*\*/

1. 什么是对象存储

· 对象存储

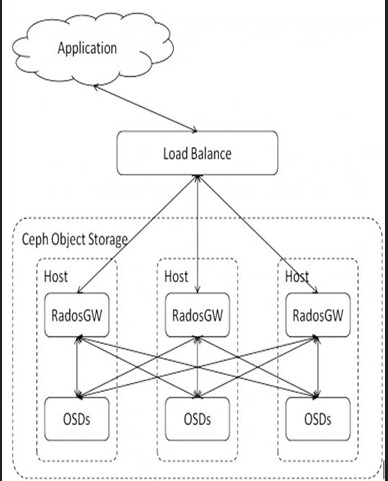
- 也就是键值存储，通过其接口指令，也就是简单的GET、PUT、DEL和其他扩展，向存储服务器上传下载数据

- 对象存储中所有数据都被认为是一个对象，所以，任何数据都可以存入对象存储服务器，如图片、视频、音频

· RGW全程是Rados Gateway

· RGW是Ceph对象存储网关，用于向客户端应用呈现存储界面，提供RESTful API访问接口

· RGW可以部署多台，拓扑如下：



RGW拓扑

1. 环境准备

· 准备一台新的虚拟机，作为元数据服务器

· 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

1. 对象存储
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

1. 新建网关实例

· 启动一个rgw服务

[root@node1 ~]# ceph-deploy rgw create node5

· 登录node3验证服务是否启动

[root@node3 ~]# ps aux |grep radosgw

root 8848 0.4 2.3 1073560 11788 ? Ssl 17:07 0:00 /usr/bin/radosgw -f --cluster ceph --name client.rgw.node3 --setuser ceph --setgroup ceph

[root@node5 ~]# systemctl status ceph-radosgw@\\*

1. 修改服务端口

· 登陆node3，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

1. 客户端测试
2. 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>

1. 使用第三方软件访问

· 登录node3(RGW)创建账户

[root@node3 ~]# radosgw-admin user create \

> --uid="testuser" --display-name="First User"

{

"user\_id": "testuser",

"display\_name": "First User",

"email": "",

"suspended": 0,

"max\_buckets": 1000,

"auid": 0,

"subusers": [],

"keys": [

{

"user": "testuser",

"access\_key": "JUB7AVZB2JHS5YHO67LF",

"secret\_key": "lRW5PJSPaVRekA9CH9W8Fg1gezIYsOi3yvE5rQBB"

}

],

"swift\_keys": [],

"caps": [],

"op\_mask": "read, write, delete",

"default\_placement": "",

"placement\_tags": [],

"bucket\_quota": {

"enabled": false,

"max\_size\_kb": -1,

"max\_objects": -1

},

"user\_quota": {

"enabled": false,

"max\_size\_kb": -1,

"max\_objects": -1

},

"temp\_url\_keys": []

}

[root@node5 ~]# radosgw-admin user info --uid=testuser

//testuser为用户，key是账户访问密钥

1. 使用第三方软件访问
2. 客户端安装软件

[root@client ~]# yum install s3cmd-2.0.1-1.el7.noarch.rpm

1. 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: JUB7AVZB2JHS5YHO67LF  
 Secret Key: lRW5PJSPaVRekA9CH9W8Fg1gezIYsOi3yvE5rQBB

S3 Endpoint [s3.amazonaws.com]: 192.168.4.13:8000

[%(bucket)s.s3.amazonaws.com]: %(bucket)s.192.168.4.13:8000

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

1. 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

2019-03-08 10:56 s3://my\_bucket

[root@client rpm]# s3cmd ls s3://my\_bucket/log

DIR s3://my\_bucket/log/

[root@client ~]# s3cmd ls s3://my\_bucket/log/

2019-03-08 10:59 1318215 s3://my\_bucket/log/messages

1. 测试下载功能

[root@client ~]# s3cmd get s3://my\_bucket/log/messages /tmp/

[root@client rpm]# ls /tmp/messages

/tmp/messages

1. 测试删除功能

[root@client ~]# s3cmd del s3://my\_bucket/log/messages

[root@client rpm]# s3cmd ls s3://my\_bucket/log/messages

[root@client rpm]# s3cmd ls s3://my\_bucket/log/

[root@client rpm]# s3cmd ls s3://my\_bucket/

1. 总结和答疑
2. DBA1
3. Day 01 数据库服务概述 构建MySQL服务器 、 数据库基本管理 MySQL数据类型
4. 数据库服务概述
   * 1. 什么是数据库
5. 什么是数据库

· DB,DataBase

- 数据库:依照某种数据模型进行组织并存放到存储器的数据集合

· DBMS,DataBase Management System

- 数据库管理系统:用来操纵和管理数据库的大型服务软件

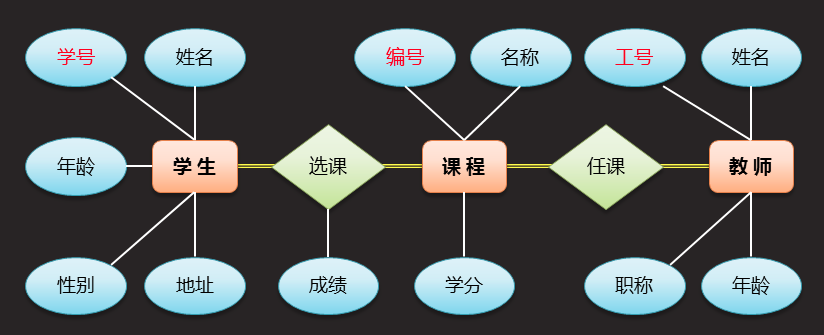
· DBS,Database System

- 数据库系统:即DB+DBMS,指带有数据库并整合了数据库管理软件的计算机系统

1. E-R数据模型

· 实体-关系 模型

- Entity-Relationship Model



1. 常见的数据库服务软件

Oracle(甲骨文)：Oracle Database，MySQL/MariaDB

Microsoft(微软)：SQL Server，Access

IBM：DB2

Sybase：Sybase

加州大学伯克利分校：PostgreSQL

* + 1. MySQL数据库介绍

1. MySQL的起源与发展过程

· 最为著名、应用最广泛的开源数据库软件

- 最早隶属于瑞典的MySQL AB公司

- 2008年1月，MySQL AB被Sun收购

- 2009年4月，SUN被Oracle收购

· 崭新的开源分支MariaDB

- 为应付MySQL可能会闭源的风险而诞生

- 由MySQL原作者Widenius主导开发

- 与MySQL保持做大程度兼容

1. MySQL的特点及应用

· 主要特点

- 适用于中小规模、关系型数据库系统

- 支持Linux/Unix、Windows等多种操作系统

- 使用C和C++编写，可移植性强

- 通过API支持Python/Java/Perl/PHP等语言

· 典型应用环境

- LAMP平台，与Apache HTTP Server组合

- LNMP平台，与Nginx组合

* 1. 构建MySQL服务器
     1. 准备工作(非必须操作)：

1. 关闭防火墙
2. 关闭SELinux
3. 如果之前有mariadb，需要先卸载并删除对应的配置和数据：

# systemctl stop mariadb

# rm -rf /etc/my.cnf

# rm -rf /var/lib/mysql/\*

# rpm -e --nodeps mariadb mariadb-server mariadb-dev

* + 1. 安装部署MySQL

[root@mysql50 ~]# tar -xf mysql-5.7.17.tar

[root@mysql50 ~]# yum -y install mysql-communit\*

* + 1. 启动服务

[root@mysql50 ~]# systemctl start mysqld

//提示：第一次启动，需要初始化数据，会比较慢

[root@mysql50 ~]# systemctl status mysqld

[root@mysql50 ~]# systemctl enable mysq

* + 1. 配置MySQL管理员密码(默认数据库管理员账户为root)

第一次启动时，mysql会自动为root账户配置随机密码，我们需要通过日志查看该密码

[root@mysql50 ~]# grep password /var/log/mysqld.log

2018-12-25T12:43:41.164573Z 1 [Note] A temporary password is generated for root@localhost: rMx89XRTPi+h

[root@mysql50 ~]# mysql -uroot -p'cvAd3af8a<j?'

mysql> show databases;

ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.

/\*\* 注意：用该密码登录到服务端后，必须马上修改密码，不然会报上面的错误 \*\*/

密码策略

validate\_password\_policy

0或者LOW[只匹配长度]

1或者MEDIUM(默认)[长度;数字、大小写;特殊符号]

2或者STRONG[长度;数字、大小写;特殊符号;字典文件]

mysql> set global validate\_password\_policy=0;

//只验证长度

mysql> set global validate\_password\_length=6;

//修改密码长度，默认值为8

mysql> alter user user() identified by "123456";

//修改登录密码

* 1. 数据库基本管理
     1. 数据库基本操作流程

· 连接登录数据库

· 创建数据库

· 创建数据表

· 插入数据记录

· 断开连接

连接MySQL数据库的命令语法格式：

[root@mysql50 ~]# mysql [ -h 服务器 IP 或域名 -u 用户名 -p 密码 数据库名称 ]

quit 或者 exit 退出数据库

[root@mysql50 ~]# mysql -h 127.0.0.1 -uroot -p123456

注意事项：

操作指令不区分大小写（密码和变量除外）

每条 SQL 语句都以分号;结束

默认不支持Tab键补齐（可以自行下载安装https://github.com/dbcli/mycli 工具实现自动补齐功能）

\c 可以取消书写错误的命令

常用的SQL命令分类：

DDL数据定义语言(create,alter,drop)

DML数据操作语言(insert,update,delete)

DCL数据控制语言(grant,revoke)

DTL数据事务语言(commit,rollback,savepoint)

* + 1. 数据库相关指令练习

mysql> show databases;

//查看数据库

mysql> use mysql;

//切换数据库

mysql> select database();

//查看当前数据库

+------------+

| database() |

+------------+

| mysql |

+------------+

1 row in set (0.00 sec);

mysql> create database tts character set utf8;

//创建数据库(支持中文)

mysql> drop database tts;

//删除数据库

提示：数据库命名规则

（数字、字母、下划线，不能纯数字；区分大小写；不能使用关键词或特殊符号）

* + 1. 数据表相关指令练习

1. 创建数据表基本语法格式如下：

create table 数据库名称，数据表名称(

字段名1 数据类型(宽度) 约束条件，

字段名2 数据类型(宽度) 约束条件，

... ...

);

创建如下图所示的数据表，数据库名称为school，数据表名称为student



mysql> show character set;

//查看所有可用编码

mysql> create database school character set utf8;

//创建数据库

mysql> create table school.student(

学号 char(20),

姓名 char(20),

性别 char(5),

手机号 int(11),

通信地址 char(50)

);

//创建数据表

1. 插入数据的格式：

insert into 数据库名称.数据表名称 values(值列表);

mysql> insert into school.student values('NSD181001','葫芦娃','男',1388888888,'北京');

//一次插入1条数据

mysql> insert into school.student values('NSD181002','蛇精','女',1389999999,'上海'),

('NSD181003','爷爷','男',1387777777,'长白山');

//一次插入多条数据（多条数据使用逗号分隔)

1. 查看数据：

mysql> select \* from school.student;

1. 更新数据语法格式：

update 数据库名称.数据表名称 set 字段=值 [where 条件]

mysql> update school.student set 性别='女';

//更新所有数据

mysql> update school.student set 性别='男' where 姓名='葫芦娃';

//更新满足条件的数据

1. 删除数据：

mysql> delete from school.student where 学号='NSD181003';

//删除表中满足条件的数据

mysql> delete from school.student;

//删除表中所有数据

1. 删除数据表：

mysql> drop table school.student;

//删除整个数据表

* 1. MySQL数据类型
     1. 数字类型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 大小 | 范围(有符号) | 范围(无符号) | 用途 |
| tinyint | 1字节 | -128~127 | 0~255 | 微小整数 |
| smallint | 2字节 | -32768~32767 | 0~65535 | 小整数 |
| mediumint | 3字节 | -223~223-1 | 0~224-1 | 中整数 |
| int | 4字节 | -231~232-1 | 0~232-1 | 大整数 |
| bigint | 8字节 | -263~263-1 | 0~264-1 | 极大整数 |
| float | 4字节 |  | | 单精度浮点数 |
| double | 8字节 |  | | 双精度浮点数 |
| decimal | Decima(M,D),其中M为有效位数，D为小数位数，M应大于D，占用M+2字节 | | | |
| unsigned | 标记使用无符号存储 | | | |

数字类型表

mysql> create table school.num(

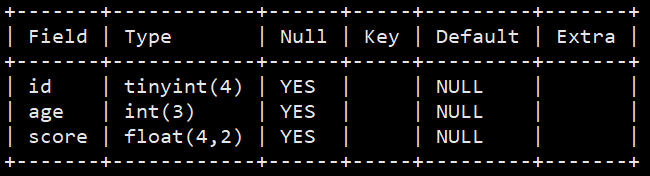
id tinyint,

age int(3),

score float(4,2));

Query OK, 0 rows affected (0.16 sec)

mysql> desc school.num;

3 rows in set (0.01 sec)

mysql> insert into school.num values(1111,22,11.2);

ERROR 1264 (22003): Out of range value for column 'id' at row 1

#提示值超出范围（tinyint 只能存-128~127 或者 0~255 之间的值）

mysql> insert into school.num values(130,22,11.2);

ERROR 1264 (22003): Out of range value for column 'id' at row 1

#130 也提示错误，因为默认使用的是有符号的存储，如果需要无符号需要添加 unsigned 标记

mysql> insert into school.num values(-125,22,11.2);

Query OK, 1 row affected (0.06 sec)

#正确

mysql> insert into school.num values(-125,22,143.434);

ERROR 1264 (22003): Out of range value for column 'score' at row 1

提示：错误，小数的总长度为 4 位，也就是整数为 2 位，小数为 2 位。当整数写 3 位就报错

mysql> insert into school.num values(-125,22,14.43433333);

ERROR 1264 (22003): Out of range value for column 'score' at row 1

提示：不报错误，整数位合法，小数位超出，系统会自动把多余的删除，进行四舍五入

* + 1. 字符类型

|  |  |
| --- | --- |
| 类型 | 描述 |
| char(字符数) | 固定长度，最大长度255字符，不够指定的字符数时自动在右边填补空格、超出指定字符数则无法写入 |
| varchar(字符数) | 可变长度，根据实际数据大小分配存储空间，超出指定字符数则无法写入 |
| text/blob | 字符数大于65535时使用 |

mysql> create table school.info(

name char(4),

email varchar(30));

mysql> desc school.info;

mysql> insert into school.info values('tom', 'tom@163.com');

Query OK, 1 row affected (0.06 sec)

#注意：字符串需要使用引号！

mysql> select \* from school.info;

* + 1. 日期时间类型

datetime 日期时间类型，占8个字符

范围1000-01-01 00:00:00.000000~9999-12-31 23:59:59.999999

如果不给该类型的数据赋值，则默认为NULL

timestamp日期时间类型，占4个字节

范围1970-01-01 00:00:00:000000~2038-01-19 03:14:07.999999

如果不给该类型的数据赋值，则mysql自动为其分配当前的系统时间

/\*

\* 注意：

\* 1.当未给timestamp字段赋值时，自动以当前系统时间赋值，而datetime字段默认赋值为NULL

\*/

时间格式：YYYYmmddhhmmss

date日期类型，占用4个字节

范围 0001-01-01~9999-12-31

默认使用4位数字表示，当只用两位数字赋值时，01~69自动识别为2001~2069，70~99 自动识别为 1970~1999

year 年份类型，占用 1 个字节，范围 1901-2155

time 时间类型，占用 3 个字节，范围 HH:MM:SS

· 时间函数

|  |  |
| --- | --- |
| 类型 | 用途 |
| now() | 获取系统当前日期和时间 |
| year() | 获取指定时间中的年份 |
| day() | 获取指定时间中的日期 |
| sleep(N) | 休眠N秒 |
| curdate() | 获取当前的系统日期 |
| cuetime() | 获取当前的系统时刻 |
| month() | 获取指定时间中的月份 |
| date() | 获取指定时间中的日期 |
| time() | 获取指定时间中的时刻 |

创建学员信息表：姓名、出生日期、入学年份、上课时间、下课时间

mysql> create table school.stuinfo(

name char(5),

birth datetime,

start year,

begin time,

end time);

mysql> insert into school.stuinfo values ('tom',20011010121200,2018,0800,1800);

mysql> select \* from school.stuinfo;

mysql> insert into school.stuinfo values ( 'lucy',20011010121200,2555,0800,1800);

ERROR 1264 (22003): Out of range value for column 'start' at row 1 #year 年份的有效范围是 1901-2155

* + 1. 枚举类型(选择类型)

enum(值 1，值 2，值 3...) #单选项

set(值 1，值 2，值 3...) #多选项

mysql> create table school.tea(

name char(5),

gender enum('boy','girl'),

interest set('book','film','music','football')

);

mysql> desc school.tea;

mysql> insert into school.tea values ('tom', 'man', 'it,boot')

//错误，超出了可选择的范围

mysql> insert into school.tea values ('tom', 'boy', 'book')

//正确输入

* 1. 总结和答疑

1. Day 02 约束条件 修改表结构 MySQL键值
   1. 约束条件
      1. 约束条件

|  |  |
| --- | --- |
| NULL | 允许为空，默认设置 |
| NOT NULL | 不允许为空 |
| Key | 索引类型 |
| Default | 设置默认值，缺省为NULL |

mysql> create table school.restrict(

name char(5) not null,

gender enum("male","female") not null default "male",

age int(3) not null default 21,

interest set("book","movie","eat")

);

mysql> desc school.restrict;

mysql> insert into school.restrict values ("tom","male",18,"book");

mysql> insert into school.restrict(name) values ("tom");

Query OK, 1 row affected (0.06 sec)

#仅插入姓名可以成功

mysql> select \* from school.restrict;

mysql> insert into school.restrict(age) values (11);

ERROR 1364 (HY000): Field 'name' doesn't have a default value

#name 不可以为空

1. 修改表结构
2. 语法结构

· 基本用法

- alter table 表名 执行动作;

|  |  |
| --- | --- |
| 执行动作 | 功能描述 |
| Add | 添加字段 |
| Modify | 修改字段类型 |
| Change | 修改字段名 |
| Drop | 删除字段 |
| Rename | 修改表名 |

* + 1. 添加新字段

· 基本用法

alter table 表名 add 字段名 类型(宽度) 约束条件;

mysql> alter table school.restrict

add

email varchar(30);

Query OK, 0 rows affected (0.33 sec)

Records: 0 Duplicates: 0 Warnings: 0

通过 after 字段名可以将新添加的字段放到某个字段后面，或者 first 直接放到第一列。

mysql> alter table school.restrict

add

phone varchar(30) not null after name;

#添加新字段 phone，放到 name 字段的后面。

mysql> alter table school.restrict

add

addr varchar(30) not null default 'beijing' first;

#添加新字段 addr，放到所有字段前面。

* + 1. 修改字段类型

· 基本用法

alter table 表名 modify 字段名 类型(宽度) 约束条件;

通过 after 字段名可以将字段放到某个字段后面，或者 first 直接放到第一列。

mysql> alter table school.restrict

modify

addr varchar(50) default 'shanghai' after name;

#修改 addr 字段的类型长度为 50，修改默认值为’shanghai’,将 addr 调整到 name 字段的后面。

mysql> desc school.restrict;

* + 1. 修改字段名

· 基本用法

alter table 表名 change 源字段名 新字段名 类型(宽度) 约束条件;

当更新类型和约束条件时，也可修改字段类型

mysql> alter table school.restrict change name myname varchar(10);

#把数据表中的 name 字段重命名为 myname，同时修改了数据类型的长度。

* + 1. 删除字段

· 基本用法

alter table 表名 drop 字段名;

表中有多条记录时，所有列的此字段的值都会被删除

mysql> alter table school.restrict drop interest;

#删除数据表中的 interest 字段。

* + 1. 修改表名

· 基本用法

alter table 表名 rename 新表名;

表对应的文件名也会改变

mysql> alter table school.restrict rename school.rest;

mysql> use school;

mysql> show tables;

1. MySQL键值
2. MySQL索引概述
3. 索引介绍

· 什么是索引：就是对数据表中的若干字段进行排序的方法，类似于对一本书做目录，有了目录就可以快速定位数据的具体位置

- 索引是对记录集的多个字段进行排序的方法

- 类似于书的目录

- 索引类型包括：Btree、B+tree、hash

1. 索引优缺点

· 优点：

- 通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性

- 可以加快数据的检索速度

· 缺点：

- 当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，会降低数据库的写速度

- 索引需要占额外的物理空间

1. 键值类型

|  |  |
| --- | --- |
| index | 普通索引 |
| unique | 唯一索引 |
| fulltext | 全文索引 |
| primary key | 主键 |
| foreign key | 外键 |

1. MySQL键值类型
2. index普通索引

使用说明

· 一个表中可以有多个index字段

· 允许重复，且可以赋值为空

· 经常吧做查询条件的字段设置为index字段

· index字段的key标志是mul

1. 建表过程中创建索引

mysql> create table school.info(

id int(6) not null,

name varchar(5),

sex enum('male','female'),

age int(3) default 1,

index(id),index(name));

Query OK, 0 rows affected (0.13 sec)

1. 在已有的数据表中创建或删除索引

/\*

\* 1.创建索引：

\* create index 索引名称 on 数据表(字段名称)

\*

mysql> create index age on school.info(age);

Query OK, 0 rows affected (0.18 sec)

mysql> create index nianling on school.info(age);

Query OK, 0 rows affected, 1 warning (0.09 sec)

提示：可以创建多个索引，索引与字段名称也可以不一样

\*

\* 2.删除索引：

\* drop index 索引名称 on 数据表

\*

mysql> drop index name on school.info;

查看索引信息：

mysql> show index from school.info\G;

\*

\*/

1. primary key主键

注意事项

· 一个表中只能有一个primary key字段

· 对应的字段值不允许有重复，且不允许赋NULL值

· 如果有多个字段都作为primary key，成为复合主键，必须一起创建

· 主键字段的key标志是pri

· 通常与AUTO\_INCREMENT连用

· 经常把表中能够唯一表示记录的字段设置为主键字段[记录编号字段]

1. 创建主键

mysql> create table student(

stu\_id char(9),

name char(5),

primary key(stu\_id)

);

Query OK, 0 rows affected (0.23 sec)

mysql> desc school.student;

提示：stu\_id 字段自动被设置为不能为 Null，Key 下面有 Pri 标记（主键标记）。

mysql> insert into school.student values (null,'tom'); #写入 null 数据会报错。

ERROR 1048 (23000): Column 'stu\_id' cannot be null

mysql> insert into school.student values ('NSD181001','卡哇伊');

Query OK, 1 row affected (0.04 sec)

mysql> insert into school.student values ('NSD181001','小帕克');

ERROR 1062 (23000): Duplicate entry 'NSD181001' for key 'PRIMARY'

#报错：主键不能重复！

mysql> insert into school.student values ('NSD181002','诸葛波波');

Query OK, 1 row affected (0.03 sec)

1. 创建主键索引的另一种方式：

mysql> create table school.student2(

id char(9) primary key,

name char(5));

Query OK, 0 rows affected (0.15 sec)

mysql> desc school.student2;

1. 对已经存在的数据表创建主键索引

mysql> create table school.student3(

id char(9),

name char(5));

Query OK, 0 rows affected (0.68 sec)

语法：alter table 数据表名称 add primary key(字段名称);

mysql> alter table school.student3 add primary key(id);

Query OK, 0 rows affected (0.34 sec)

提示：在已经存在的表中创建主键索引，一定要确保做为主键的字段数据中没有 null 的值。

如果已经有null值，可以delete删除数据或者update更新数据为非null。

1. 删除数据表中的主键索引

mysql> alter table school.student3 drop primary key;

Query OK, 0 rows affected (0.52 sec)

#提示:删除主键后，对应的字段数据就可以出现重复的数据了。

1. 复合主键索引(多个字段做主键)

· 不做主键(可能同一个人技能不同)

|  |  |  |
| --- | --- | --- |
| 姓名 | 单位 | 技能 |
| 孙悟空 | 西游记 | 100 |
| 孙悟空 | 西游记 | 80 |

· 以姓名为主键(不同书中的任务技能水平不同，相同名称时无法写入，因为姓名不能重复)

|  |  |  |
| --- | --- | --- |
| 姓名 | 单位 | 技能 |
| 孙悟空 | 西游记 | 100 |
| 孙悟空 | 沉香救母 | 80 |

· 创建复合主键索引(姓名和单位，不能重复，但是单独的姓名或单位可以重复)

mysql> create table school.book(

姓名 char(20),

单位 char(20),

技能 int(100),

primary key(姓名,单位));

Query OK, 0 rows affected (0.17 sec)

mysql> insert into school.book values ('孙悟空','西游记',100);

Query OK, 1 row affected (0.03 sec)

mysql> insert into school.book values ('孙悟空','沉香救母',80);

Query OK, 1 row affected (0.02 sec)

#提示：单独的姓名重复是可以的

mysql> insert into school.book values ('孙悟空','沉香救母',100);

ERROR 1062 (23000): Duplicate entry '孙悟空-沉香救母' for key 'PRIMARY'

#提示：姓名和单位都重复会报错。

· 自动添加属性（可以自动将数据自动加 1）

mysql> create table school.demo(

id int(100) auto\_increment primary key,

name char(10));

Query OK, 0 rows affected (0.16 sec)

mysql> insert into school.demo(name) values ('tom');

mysql> insert into school.demo(name) values ('jerry');

mysql> select \* from school.demo;

提示：id 自动不写，默认也会自动加 1.

mysql> insert into school.demo values (10,'jerry');

mysql> insert into school.demo(name) values ('xyz');

mysql> select \* from school.demo;

提示：当同时有自增长和主键时，一定要通过 alter 将自增长删除后才可以删除主键无法直接删除主键。

/\* 通过change取消自增属性

\* mysql> alter table school.demo change id id int(5);

\*/

1. foreign key外键

· 一个作者信息表

· 一个图书信息表

· 要求图书表中的图书作者必须是作者表中的作者。

· 什么是外键？

· 让当前表字段的值在另一个表中某个字段值的范围内选择。

· 使用外键的条件：

· 表存储引擎必须是 innodb（默认就是）

· 字段类型必须一致

· 被参照字段必须是索引类型中的一种（primary key）

1. 创建外键的语法：

foreign key(表 A 的字段名称)

references 表 B(字段名称)

on update casecade #同步更新

on delete casecade #同步删除

1. 创建数据库：

mysql> create database press character set utf8;

1. 创建包含主键的数据表（作者信息表）：

mysql> create table press.author(

姓名 char(10) primary key,

地址 char(10));

1. 往作者信息表中插入数据:

mysql> insert into press.author values

('施耐庵','苏州'),

('曹雪芹','辽宁'),

('罗贯中','山西');

1. 创建图书信息表并创建外键：

mysql> create table press.book(

书名 char(20),

作者 char(10),

foreign key(作者) references press.author(姓名) on update cascade on delete cascade);

mysql> insert into press.book values ('红楼','无');

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails(`press`.`book`, CONSTRAINT`book\_ibfk\_1` FOREIGN KEY (`作者`) REFERENCES `author` (`姓名`) ON DELETE CASCADE ON UPDATE CASCADE)

#当作者不在 author 表中就报错

mysql> insert into press.book values ('红楼','曹雪芹');

Query OK, 1 row affected (0.21 sec)

mysql> delete from press.author where 姓名='曹雪芹';

Query OK, 1 row affected (0.03 sec)

#提示：当作者表中的作者被删除后，图书信息表中对应的数据也会被删除。

1. 删除外键，需要先查看外键的名称:

mysql> show create table press.book\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: book

Create Table: CREATE TABLE `book` (

`书名` char(20) DEFAULT NULL,

`作者` char(10) DEFAULT NULL,

KEY `作者` (`作者`),

CONSTRAINT `book\_ibfk\_1` FOREIGN KEY (`作者`) REFERENCES `author` (`姓名`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8

1 row in set (0.00 sec)

1. 删除外键：

mysql> alter table press.book drop foreign key book\_ibfk\_1;

mysql> drop table press.book;

1. 在现有的数据表中创建外键:

mysql> create table press.book(

书名 char(20),

作者 char(10));

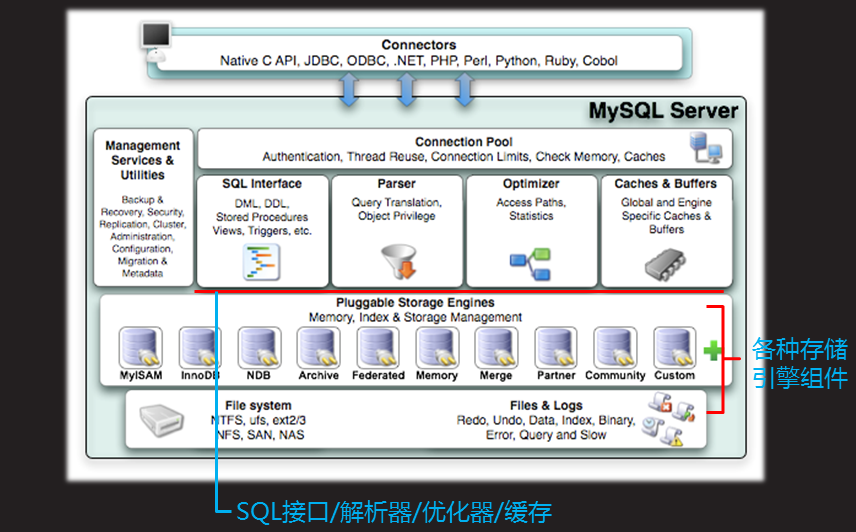
mysql> alter table press.book

add

foreign key(作者) references press.author(姓名)

on update cascade on delete cascade;

1. Day 03 MySQL存储引擎 、 数据导入导出 管理表记录 匹配条件
   1. MySQL存储引擎
      1. MySQL工作原理
2. MySQL体系结构



API接口，连接池，SQL接口，解析器，优化器，缓存，存储引擎

1. 管理工具：MySQL自带的命令
2. 连接池：验证用户连接和检查系统资源
3. SQL接口：把sql命令传给mysqld进程处理
4. 分析器：检查sql命令语法
5. 优化器：对执行的sql命令优化
6. 查询缓存：存储查找过的数据(8MB 从物理内存划分出来的)
7. 存储引擎：处理表的处理器，不同的存储引擎有不同的功能和数据存储方式
8. 文件系统：机器的硬盘
9. MySQL存储引擎

· 作为可插拔式的组件提供

- MySQL服务软件自带的功能程序，处理表的处理器

- 不同的存储引擎有不同的功能和数据存储方式

· 默认的存储引擎

- MySQL 5.0/5.1 --> MyISAM

- MySQL 5.5/5.6 --> InnoDB

· 列出可用的存储引擎类型

mysql> show engines;

mysql> show engines\G

* + 1. 存储引擎的配置

1. 修改表的存储引擎

· 建表时手动指定

· 未指定时，使用默认存储引擎

show create table 表名\G;

1. 设置默认存储引擎

· 修改/etc/my.cnf配置文件

[mysqld]

default-storage-engine=存储引擎名称

* + 1. 存储引擎特点

1. MyISAM存储引擎
2. 主要特点

支持表级锁

不支持外键、事务、事务回滚

表文件个数：

t3.frm 表结构

t3.MYD 数据

t3.MYI index信息

1. InnoDB存储引擎
2. 主要特点

支持外键

支持行级锁

支持事务和事务回滚

表文件个数：

t6.frm 表结构

t6.ibd 数据+index

1. MySQL锁机制
2. 锁粒度

· 表级锁：一次直接对整张表进行加锁

· 行级锁：只锁定某一行

· 页级锁：对整个页面(MySQL管理数据的基本存储单位进行加锁)

1. 锁类型

· 读锁(共享锁)：支持并发读

· 写锁(互斥锁、排它锁)：是独占锁，上锁期间其他线程不能读表或写表

1. 查看当前的锁状态

· 检查Table\_lock开头的变量，%作通配符

show status like 'Table\_lock%';

mysql> show status like 'Table\_lock%';

1. 事务的介绍

· 事务：一次数据访问从开始到结束的过程

/\* 登录 数据操作 断开连接 \*/

· 事务回滚：在访问过程中，任意一步操作失败，恢复之前的所有操作

· 事务日志：记录对所有innodb存储引擎表的操作

/\* 事务日志文件

\* 记录没有提交的命令

\* ibdata1

\* 记录已经提交的命令

\* ib\_logfile0

\* ib\_logfile1

\*/

· 事务特性(ACID）

1. 原子性[Atmic]

- 事务的整个操作是一个整体，不可分割，要么全部成功，要么全部失败

1. 一致性[Consistency]

- 事务操作的前后，表中的记录不违背数据完整性约束

1. 隔离性[Isolation]

- 事务操作是相互隔离不受影响的

1. 持久性[Durability]

- 数据一旦提交，不可改变，永久改变表数据

· 查看自动提交是否开启

mysql> show variables like 'autocommit';

· 关闭自动提交

mysql> set autocommit=off;

· 数据回滚

mysql> rollback;

· 提交数据

mysql> commit;

1. 工作中建表时，如何决定表使用哪种存储引擎？

查询操作多的表，适合使用myisam存储引擎

优点：节省系统资源

修改操作多的表，适合使用innodb存储引擎

优点：并发访问量大

* 1. 数据导入导出
     1. 设置搜索路径

1. 查看默认使用目录及目录是否存在

mysql> show variables like "secure\_file\_priv";

secure\_file\_priv | /var/lib/mysql-files/

[root@mysql50 ~]# ls -ld /var/lib/mysql-files/

1. 修改目录及查看修改结果

[root@mysql50 ~]# mkdir /myload;chown mysql /myload

[root@mysql50 ~]# vim /etc/my.cnf

[mysqld]

secure\_file\_priv="/myload"

[root@mysql50 ~]# systemctl restart mysqld

* + 1. 数据导入

1. 命令格式

load data infile "目录名/文件名" into table 表名 fields terminated by "分隔符" lines terminated by "\n";

· 注意事项

- 字段分隔符要与文件内的一致

- 指定导入文件的绝对路径

- 导入数据的表字段类型要与文件字段匹配

- 禁用selinux保护机制

1. 步骤
2. 把系统文件拷贝到检索目录下

[root@mysql50 ~]# cp /etc/passwd /myload/

[root@mysql50 ~]# head -1 /myload/passwd

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

1. 根据导入文件的内容创建表

mysql> create table userdb.user(

username char(50),

password char(1),

uid int,

gid int,

comment char(150),

homedir char(180),

shell char(50)

);

Query OK, 0 rows affected (0.18 sec)

1. 执行导入命令

mysql> load data infile "/myload/passwd" into table userdb.user fiellds terminated by ":" lines terminated by "\n";

Query OK, 41 rows affected (0.20 sec)

Records: 41 Deleted: 0 Skipped: 0 Warnings: 0

1. 添加一个主键id

mysql> alter table userdb.user

add id int primary key auto\_increment first;

Query OK, 0 rows affected (0.51 sec)

Records: 0 Duplicates: 0 Warnings: 0

* + 1. 数据导出

1. 命令格式

SQL查询 into outfile "目录名/文件名";

SQL查询 into outfile "目录名/文件名" fields terminated by "分隔符" lines terminated by "\n";

· 注意事项

- 导出的内容由SQL查询语句决定

- 导出的是表中的记录，不包括字段名

- 禁用selinux

1. 例子

mysql> select \* from userdb.user into outfile "/myload/a1.txt" fiel

ds terminated by ":" lines terminated by "\n";

Query OK, 41 rows affected (0.00 sec)

mysql> select \* from userdb.user into outfile "/myload/a2.txt";

Query OK, 41 rows affected (0.01 sec)

mysql> select \* from userdb.user where id <= 3 into outfile "/myload/a3.txt" fields terminated by ":" lines terminated by "???";

Query OK, 3 rows affected (0.01 sec)

* 1. 管理表记录
     1. 增加表记录

1. 插入单条记录，单个字段赋值

insert into 库名.表名(字段名) values 值

mysql> insert into userdb.user(username,shell) values("lily","/bin/bash");

Query OK, 1 row affected (0.05 sec)

1. 插入多条记录，所有字段赋值

insert into 库名.表名 values (值1，值2... );

mysql> insert into userdb.user values(43,"bob","x",5000,5000,"bob","/home/bob","bin/bash");

Query OK, 1 row affected (0.03 sec)

* + 1. 查询表记录

select 字段 from 表名 [where 条件表达式];

* + 1. 更新表记录

update 库名.表名 set 字段名=字段值 [where 条件表达式];

* + 1. 删除表记录

delete from 表名 库名.表名 [where 条件];

* 1. 匹配条件

1. 基本匹配条件
2. 数值比较

|  |  |
| --- | --- |
| 类型 | 用途 |
| = | 等于 |
| >、>= | 大于、大于等于 |
| <、<= | 小于、小于等于 |
| != | 不等于 |

where 字段名 符号 数字

mysql> select username,uid from userdb.user where uid <= 10;

1. 字符比较/匹配空/非空

|  |  |
| --- | --- |
| 类型 | 用途 |
| = | 等于 |
| != | 不相等 |
| IS NULL | 匹配空 |
| IS NOT NULL | 非空 |

where 字段名 符号 "字符串"

mysql> select username,shell from userdb.user where shell!="/bin/bash";

where 字段名 is null

where 字段名 is not null

mysql> select username,gid from userdb.user where gid is null;

1. 逻辑匹配(多个判断条件)

|  |  |
| --- | --- |
| 类型 | 用途 |
| OR | 逻辑或 |
| AND | 逻辑与 |
| ! | 逻辑非 |
| () | 提高优先级 |

mysql> select \* from userdb.user where username="bin" and uid=3 and shell="/bin/bash";

Empty set (0.01 sec)

mysql> select \* from userdb.user where username="bin" or uid=3 or shell="/bin/bash";

... ...

5 rows in set (0.00 sec)

1. 范围匹配/去重显示

|  |  |
| --- | --- |
| 类型 | 用途 |
| in | 在...里 |
| not in | 不在...里 |
| between 数字1 and 数字2 | 在...之间 |
| distinct 字段名 | 去重显示 |

mysql> select \* from userdb.user where id between 1 and 10;

mysql> select shell from userdb.user;

mysql> select distinct shell from userdb.user;

1. 高级匹配条件
2. 模糊查询

where 字段名 like '通配符'

· 通配符

|  |  |
| --- | --- |
| 类型 | 用途 |
| \_ | 匹配单个字符 |
| % | 匹配0~N个字符 |

· 查找user表中长度为4的username的值

mysql> select username from user where username like '\_\_\_\_';

· 查找user表中username包含a的值

mysql> select username from user where username like '%a%';

· 查找user表中username字段以y结尾的值

mysql> select username from user where username like '%y';

· 查找user表中长度大于4的username的值

mysql> select username from user where username like '\_\_%\_\_';

1. 正则表达式

where 字段名 regexp '正则表达式'

正则元字符：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ^ | $ | . | [] | \* | | |

· 查找以r开头以a结尾的username

mysql> select username from user where username regexp '^r.\*a$';

· 查找username中包含数字的值

mysql> select username from user where username regexp '[0-9]';

1. 四则运算

计算符号

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | - | \* | / | % |

· 求和

mysql> select username,uid+gid result from user where username='bin';

· 求平均值

mysql> select username,uid,gid,(uid+gid)/2 average from user where username='bin';

1. 操作查询结果
2. 聚集函数

|  |  |
| --- | --- |
| avg() | 统计字段平均值 |
| sum() | 统计字段之和 |
| min() | 统计字段最小值 |
| max() | 统计字段最大值 |
| count() | 统计字段值个数 |

· 求uid平均值

· 求id<10的uid的平均值

· 求uid的和

mysql> select avg(uid) from user;

mysql> select avg(uid) from user where id <= 10;

mysql> select sum(uid) from user;

mysql> select min(uid) from userdb.user where shell!="/bin/bash";

mysql> select max(uid) from userdb.user where shell!="/bin/bash";

mysql> select count(username) from userdb.user where shell!="/bin/bash";

mysql> select count(username) from userdb.user;

mysql> select count(\*) from userdb.user;

1. 查询结果排序

order by 字段名 [asc|desc];

//通常是数值类型字段

mysql> select \* from user order by uid\G

1. 查询结果分组

group by 字段名;

//通常是字符类型字段

mysql> select shell from user group by shell;

1. 查询结果过滤

sql查询 having条件表达式;

sql查询 where 条件 having 条件表达式;

sql 查询group by 字段名 having 条件表达式;

1. 限制查询结果显示行数

SQL查询 limit N;

//显示查询结果前N条记录

SQL查询 limit N,M;

//显示指定范围内的查询记录

SQL查询 where 条件查询 limit 3;

//显示查询结果前三条记录

SQL查询where条件查询 limit 3,3;

//从第四条开始，共显示3条

mysql> select \* from userdb.user where uid >= 10 and uid <= 500 limit 3;

mysql> select id,username,homedir from userdb.user where uid >= 10 and uid <= 500 limit 3,3;

mysql> select username,uid,homedir from userdb.user where uid > 1000 order by uid desc limit 1;

* 1. 总结和答疑

1. Day 04 多表查询 MySQL管理工具 、 用户授权及撤销
2. 多表查询
3. 复制表

· 命令格式

create table 库名.表名 SQL查询;

/\* 此方法不继承索引 \*/

mysql> create table user\_bak select \* from userdb.user;

· 作用1：备份表

mysql> create table user2 select \* from userdb.user;

· 作用2：快速建表

mysql> use db4;

mysql> select \* from userdb where 11<10;

1. 多表查询
2. 多表查询概述

· 多表查询，也称为连接查询

- 将两个或两个以上的表，按某个条件连接起来，从中选取需要的数据

- 当多个表中存在相同意义的字段(字段名可以不同)时，可以通过该字段连接多个表

1. 多表查询

· 格式1

- select 字段名列表from 表a,表b;

以上格式的查询结果叫笛卡尔集

查询结果的总条目数 = (表a的记录数x表b的记录数)

· 格式2

- select 字段名列表 from 表a,表b where 条件;

mysql> create table t1 select username,uid,password from userdb.user limit 3;

mysql> create table t2 select username,uid,shell,homedir from userdb.user limit 3;

mysql> select \* from db4.t1,db4.t2;

mysql> select \* from db4.t1,db4.t2 where db4.t1.uid=db4.t2.uid;

1. where嵌套查询

· 把内层查询结果作为外层查询条件

· 格式：

- select 字段名列表 from 表名 where 条件(select 字段名列表 from 表名 where 条件)

mysql> select username,age from userdb.user

where age < (select avg(age) from userdb.user);

mysql> select username from userdb.user

where username in(select username from db4.t1 where uid <= 2);

1. 左连接查询

左连接查询条件成立时，以左边的表为主显示查询结果

select 字段名列表 from 表a left join 表b on 条件表达式;

mysql> create table t3 select username,uid,homedir,password from userdb.user limit 3;

mysql> create table db4.t4 select username,uid,homedir,password from userdb.user limit 6;

mysql> select \* from db4.t3 left join db4.t4 on t3.uid=t4.uid;

1. 右连接查询

右连接查询条件成立时，以右边的表为主显示查询结果

select 字段名列表 from 表a right join 表b on 条件表达式;

mysql> select \* from db4.t3 right join db4.t4 on t3.uid=t4.uid;

1. MySQL管理工具
   * 1. MySQL管理方式
2. MySQL管理方式
3. 常见的MySQL管理工具

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 界面 | 操作系统 | 说明 |
| mysql | 命令行 | 跨平台 | MySQL官方bundle包自带 |
| MySQL-Workbench | 图形 | 跨平台 | MySQL官方提供 |
| MySQL-Front | 图形 | Windows | 开源，轻量级客户端软件 |
| phpMyAdmin | 浏览器 | 跨平台 | 开源，需LAMP |
| Navicat | 图形 | Windows | 专业、功能强大、商业版 |

* + 1. MySQL管理工具

1. MySQL Workbench
2. MySQL front
3. phpMyAdmin

· 基本思路

1. 安装httpd、mysql、php-mysql及相关包

[root@mysql50 phpAdmin]# yum install httpd php php-mysql -y

1. 启动httpd服务程序

[root@mysql50 day04]# systemctl restart httpd

1. 解压phpMyAdmin包，部署到网站目录

[root@mysql50 day04]# tar xf phpMyAdmin-2.11.11-all-languages.tar.gz

[root@mysql50 day04]# mv phpMyAdmin-2.11.11-all-languages /var/www/html/phpmyadmin

1. 配置config.inc.php指定mysql主机地址

[root@mysql50 phpmyadmin]# cp config.sample.inc.php config.inc.php

[root@mysql50 phpmyadmin]# vim config.sample.inc.php

$cfg['Servers'][$i]['host'] = 'localhost';

$cfg['blowfish\_secret'] = 'zqy999'; /\* YOU MUST FILL IN THIS FOR COOKIE AUTH! \*/

[root@mysql50 ~]# chown -R apache:apache /var/www/html/phpmyadmin/

1. 创建授权用户
2. 浏览器访问、登录使用

[root@yyzh\_pc ~]# firefox http://mysql50/phpmyadmin

1. 用户授权及撤销
2. 密码恢复及设置
3. 恢复MySQL管理员密码/修改数据库管理员root用户本机登录密码

· 密码忘了怎么办？

1. 停止MySQL服务程序
2. 跳过授权启动MySQL服务程序

[root@mysql50 ~]# vim /etc/my.cnf

[root@mysql50 ~]# systemctl restart mysqld

1. 重设root密码

mysql> select host,user,authentication\_string from mysql.user;

mysql> update mysql.user set authent\_string=password("888888")

where

user="root" and host="localhost";

mysql> flush privileges;

//这一步可以换成重启mysqld服务

[root@mysql50 ~]# mysqladmin -hlocalhost -uroot -p password

Enter password:

New password:

Confirm new password:

1. 以正常方式重启MySQL服务程序
2. 设置管理员密码
3. 查看模块、修改配置文件、重启服务
4. 在Shell命令行修改登录密码

- 需要验证旧密码

- 不适用于跳过授权表启动的情况

1. 创建授权

· 用户授权：在数据库服务器上 添加链接用户及设置访问权限

· 默认哪个用户有授权权限：数据库管理员root用户本机登录才有授权权限

1. MySQL授权库和表信息

· 授权记录在哪里：

授权库 mysql

授权表

|  |  |
| --- | --- |
| user | 已有的用户及访问权限 |
| db | 记录已有的用户对数据库的访问权限 |
| tables\_priv | 记录已有的用户对表的访问权限 |
| columns\_priv | 记录已有的用户对字段的访问权限 |

1. Grant配置授权

mysql> grant 权限列表 on 数据库名 to 用户名@'客户端地址'

identified by '密码' [ with grant option ];

/\* 添加一个拥有连接权限的用户[ 拥有授权权限的用户 ] \*/

· 注意事项

- 当库名.表名为\*.\*时，匹配所有库所有表

- 授权设置存放在mysql库的user表

· 权限列表

- all：匹配所有权限

- select，update,insert .. ..

- select,update(字段1，.. ..，字段N)

· 客户端地址

- %：匹配所有主机

- 192.168.1.%：匹配指定的一个网段

- 192.168.1.1：匹配指定IP地址的单个主机

- %.tarena.com：匹配一个DNS区域

- svr.tarena.com：匹配指定域名的单个主机

mysql> grant all on \*.\* to mydba@"%"

-> identified by "123456" with grant option;

Query OK, 0 rows affected, 1 warning (0.00 sec)

1. 查看用户授权

客户端测试：

[root@mysql51 ~]# mysql -hmysql50 -umydba -p

Enter password:

mysql> select user();

mysql> show grants;

mysql> grant select on \*.\* to mysel@'%'

-> identified by '123456';

Query OK, 0 rows affected, 1 warning (0.00 sec)

[root@mysql51 ~]# mysql -hmysql50 -umysel -p

Enter password:

1. 重设用户密码
2. 撤销授权
3. 撤销用户权限

revoke 权限列表 on 库名.表名 from 用户名@"客户端地址"

mysql> revoke insert,drop on \*.\* from mydba@'%';

Query OK, 0 rows affected (0.00 sec)

1. 删除授权用户

drop user 用户名@"客户端地址";

mysql> drop user mydba@'%';

Query OK, 0 rows affected (0.00 sec)

1. 总结和答疑
2. Day 05 mysqldump 实时增量备份 、 innobackupex
3. mysqldump
4. 数据库备份概述
5. 数据备份方式

· 物理备份：直接拷贝库或表对应系统文件

- 冷备：cp、tar、.. ..

· 逻辑备份：使用备份命令或软件对数据做备份，生成对应的备份文件

- mysqldump

- mysql

1. 数据备份策略：

· 完全备份

- 备份所有数据(一张表、一个库、一台数据库服务器)

/\* 备份新产生的数据，两者参照不同 \*/

· 增量备份

- 备份上次备份后，所有新产生的数据

· 差异备份

- 备份完全备份后，所有新产生的数据

1. 物理备份
2. 备份操作

[root@mysql50 ~]# mkdir /mybak

[root@mysql50 ~]# cp -rp /var/lib/mysql/userdb /mybak/

[root@mysql50 mybak]# tar czvf userdb.tar.gz userdb/

1. 恢复操作

[root@mysql50 mybak]# scp userdb.tar.gz mysql51:~

[root@mysql51 ~]# tar xf userdb.tar.gz -C /var/lib/mysql/

1. 物理备份的优点和缺点

· 优点

- 简单粗暴

· 缺点

- 具有过程性(考虑备份中写操作)

- 兼容性差(跨平台)

- 恢复需要停止服务

- innodb引擎不适合这样备份(备份不了事务日志)

总结：点一下，玩一年，炸服不花一分钱

1. 逻辑备份
2. 数据完全备份

mysqldump -uroot -p密码 库名 > 路径/xxx.sql

[root@mysql50 mybak]# mysqldump -uroot -p --all-databases > /mybak/mysql-all.sql

/\*等价于mysqldump -uroot -p -A > /mybak/mysql-all.sql \*/

[root@mysql50 mybak]# mysqldump -uroot -p userdb user > /mybak/user.sql

[root@mysql50 mybak]# mysqldump -uroot -p -B userdb db4 > /mybak/2b.sql

[root@mysql50 mybak]# mysqldump -uroot -p db4 t1 t2 > /mybak/2t.sql

1. 数据完全恢复

mysql -uroot -p密码 库名 < 路径/xxx.sql

恢复库

[root@mysql51 mybak]# mysql -uroot -p < 2b.sql

· 恢复单表之前需要先建立数据库

缺点

· 在备份和恢复期间会锁表

· 备份后新增的数据无法恢复

1. 实时增量备份/恢复
2. binlog简介
3. binlog日志概述

· 二进制日志，用于记录所有更改数据的操作

配置：

log\_bin[=dir/name]

server\_id=数字

max\_binlog\_size=数字m

1. 采用binlog日志的好处

- 记录除查询外的所有SQL命令

- 可用于数据恢复

- 配置mysql主从同步的必要条件

1. 单个binlog日志大小：

- 新版本最大值为1G

- 旧版本最大值为500M

1. 使用binlog日志
2. 启用binlog日志

[mysqld]

.. ..

log-bin

//启用binlog日志

server\_id=100

//指定id值

[root@mysql50 ~]# vim /etc/my.cnf

[mysqld]

server\_id=50

log\_bin=/logdir/db50

[root@mysql50 ~]# mkdir /logdir

[root@mysql50 ~]# chown mysql /logdir/

[root@mysql50 ~]# systemctl restart mysqld

1. 手动创建新的日志文件(4种方式)

# systemctl restart mysqld

# mysqldump -uroot -p --flush-logs db5 > /mybak/db5.sql

# mysql -uroot -p -e "flush logs"

mysql> flush logs;

mysql> reset master;

1. 清理binlog日志

show master status; //显示正在使用日志文件信息

· 删除早于指定版本的binlog日志

- purge master logs to "binlog文件名";

· 删除所有binlog日志，重建新日志

- reset master;

1. 分析binlog日志

· 查看日志当前记录格式

mysql> show variables like 'binlog\_format';

/\* 三种记录格式

\* 1.statement：每一条修改数据的sql命令都会记录在binlog日志中 报表格式

\* 2.row：不记录sql语句上下文相关信息，仅保存那条记录被修改 行模式

\* 3.mixed：以上两种格式混合使用 混合模式

\*/

· 修改日志记录格式

[root@mysql50 ~]# vim /etc/my.cnf

[mysqld]

server\_id=50

log\_bin=/logdir/db50

binlog\_format=mixed

[root@mysql50 ~]# mysql -uroot -p -e 'show variables like "binlog\_format"'

· 使用mysqlbinlog工具

- 格式：mysqlbinlog [选项] binlog日志文件名

· 常用选项 偏移量 - position 日期 - datetime

--start-datetime="yyyy-mm-dd hh:mm:ss"

--stop-datetime="yyyy-mm-dd hh:mm:ss"

--start-position=数字

--stop-position=数字

mysql> create database bbsdb;

Query OK, 1 row affected (0.05 sec)

mysql> create table bbsdb.user(

-> name char(10),

-> password char(6)

-> );

Query OK, 0 rows affected (0.36 sec)

mysql> insert into bbsdb.user values("bob","abc123");

Query OK, 1 row affected (0.15 sec)

mysql> insert into bbsdb.user values("tom","123456");

Query OK, 1 row affected (0.11 sec)

mysql> insert into bbsdb.user values("xiaohong","654321");

Query OK, 1 row affected (0.23 sec)

1. 利用binlog恢复数据
2. 基本思路

· 使用mysqlbinlog提取历史SQL操作

· 通过管道交给mysql命令执行

mysqlbinlog 日志文件 | mysql -u$user -p$password

1. 应用实例

[root@mysql50 logdir]# scp db50.000003 mysql51:~

[root@mysql51 ~]# mysqlbinlog db50.000003 | mysql -uroot -p

[root@mysql51 ~]# mysql -uroot -p -e "show databases"

bbsdb

mysql> insert into user(name) values("jack"),("alice");

mysql> delete from user where name="jack";

mysql> delete from user where name="alice";

mysql> select \* from bbsdb.user;

[root@mysql50 logdir]# mysqlbinlog db50.000003 | less

[root@mysql50 ~]# mysqlbinlog /logdir/db50.000003 --start-position=1525 --stop-position=1648 | mysql -uroot -p

mysql> select \* from bbsdb.user;

1. innocackupex
   * 1. MySQL备份工具
2. 常用的MySQL备份工具

· 物理备份缺点

- 跨平台性差

- 备份时间长、冗余备份、浪费存储空间

· mysqldump备份缺点

- 效率较低，备份和还原速度慢

- 备份过程中，数据插入和更新操作会被挂起

1. XtraBackup工具

· 一款强大的在线热备份工具

- 备份过程中不锁库表，适合生产环境

- 由专业组织Percona提供(改进MySQL分支)

· 主要含两个组件

- xtrabackup：C程序，支持InnoDB/XtraDb

- innobackupex：以Perl脚本封装xtrabackup，支持MyISAM和InnoDB

* + 1. 安装percona

1. 安装percona

· 下载适配的ROM包并安装

- percona-xtrabackup

[root@mysql50 day05]# ls

libev-4.15-1.el6.rf.x86\_64.rpm

percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

[root@mysql50 day05]# yum -y install ./\*.rpm

1. innobackupex基本使用

· 基本选项

|  |  |
| --- | --- |
| 常用选项 | 含义 |
| --host | 主机名 |
| --user | 用户名 |
| --port | 端口号 |
| --password | 密码 |
| --databases | 数据库名  --databases="库名";--databases="库1 库2";--databases="库.表" |
| --no-timestamp | 不用日期命名备份文件存储的子目录名 |
| --redo-only | 日志合并 |
| --apply-log | 准备还原(回滚日志) |
| --copy-back | 恢复数据 |
| --incremental 目录名 | 增量备份 |
| --incremental-basedir=目录名 | 增量备份时，指定上一次备份数据存储的目录名 |
| --incremental-dir=目录名 | 准备恢复数据时，指定增量备份数据存储的目录名 |
| --export | 导出表信息 |
| import | 导入表空间 |

* + 1. innobk应用案例

1. 完全备份与恢复
2. 备份数据

[root@mysql50 ~]# innobackupex --user root --password 123456 /allbak --no-timestamp

[root@mysql50 ~]# ls /allbak

1. 恢复数据

//清空数据

[root@mysql50 ~]# rm -rf /var/lib/mysql

//准备恢复数据

[root@mysql50 ~]# innobackupex --apply-log /allbak/

//把备份文件拷贝到数据库目录下

[root@mysql50 ~]# innobackupex --copy-back /allbak/

//修改目录的所有者和组为mysql用户和组

[root@mysql50 allbak]# chown -R mysql.mysql /var/lib/mysql

//启动服务

[root@mysql50 allbak]# systemctl restart mysqld

//管理员登录查看数据

1. 增量备份与恢复

/\* 注意：增量备份只适用于InnoDB存储引擎 \*/

· 应用实例

1. 增量备份时，必须先有一次备份，通常是完全备份

[root@mysql50 mysql]# innobackupex --user root --password 123456 /fullbak --no-timestamp

1. 第一次增量备份

[root@mysql50 ~]# innobackupex --user root -password 123456 \

--databases ="库名列表" --incremental /new1 \

--incremental-basedir="/fullbak" --no-timestamp

1. 第二次增量备份

[root@mysql50 ~]# innobackupex --user root --password 123456 \

--databases="库名列表" --incremental /new2 \

--incremental-basedir="/new1" --no-timestamp

/\* 准备恢复数据 \*/

1. 完整恢复

[root@mysql50 mysql]# rm -rf /var/lib/mysql

[root@mysql50 mysql]# innobackupex --apply-log --redo-only /fullbak

1. 合并增量

[root@mysql50 ~]# innobackupex --user root --password 123456 \

--databases = "库名列表" --apply-log --redo-only /fullbak \

--incremental-dir="/new1"

/\* 合并增量1 \*/

[root@mysql50 ~]# innobackupex --user root --password 123456 \

--databases = "库名列表" --apply-log --redo-only /fullbak \

--incremental-dir="/new2"

/\* 合并增量2 \*/

1. 恢复数据

[root@mysql50 mysql]# innobackupex --copy-back /fullbak

* + 1. 在完全备份文件中恢复单个表

1. 环境准备

mysql> create table db5.b(

-> name char(10))engine=innodb;

Query OK, 0 rows affected (0.56 sec)

mysql> insert into db5.b values("bob"),("bob"),("bob"),("bob"),("bob");

Query OK, 5 rows affected (0.16 sec)

1. 备份库db5

[root@mysql50 ~]# innobackupex --user=root --password=123456 --databases="db5" /db5\_bak --no-timestamp

1. 恢复b表
2. 创建表b

mysql> create table db5.b( name char(10));

1. 删除b表 表空间文件

[root@mysql50 db5]# cd /var/lib/mysql/db5;ls

/\* 此时直接rm删除会导致事务日志不一致 \*/

mysql> alter table db5.b discard tablespace;

1. 在备份数据里导出表信息

[root@mysql50 db5]# innobackupex -u=root -p=123456 --apply-log --export /db5\_bak

[root@mysql50 db5]# ls /db5\_bak/db5/b\*

/db5\_bak/db5/b.cfg /db5\_bak/db5/b.frm

/db5\_bak/db5/b.exp /db5\_bak/db5/b.ibd

/\*

在mysql界面中使用系统命令

mysql> system ls /

bin dev home logdir mybak proc sbin tmp

boot etc lib media myload root srv usr

db5\_bak fullbak lib64 mnt opt run sys var

\*/

1. 把导出的表信息文件拷贝到数据库目录下

[root@mysql50 db5]# cp /db5\_bak/db5/b.\* .

1. 修改表信息文件的所有者和组用户为mysql

[root@mysql50 db5]# chown mysql.mysql b.\*

1. 导入表信息

mysql> alter table db5.b import tablespace;

1. 查看表记录

mysql> select \* from db5.b;

1. 总结和答疑
2. innobackupex增量备份总结：
3. 完全备份、增量备份、数据恢复之间的关系
4. 增量备份之前需要有一次全量备份
5. 每一次增量备份需要指定一个参照，即incremental-basedir，该次备份是在参照的基础之上做的备份
6. 数据恢复时，需要经历三个阶段：1.增量合并; 2.拷贝数据; 3.权限修改; 其中有几点需要注意：
7. 合并过程中需要加redo-only选项，主要合并各类日志，将增量内容合并到基础包中,合并结束后，增量包可以直接删除
8. 拷贝数据只拷贝基础包
9. innobackupex日志合并报错

xtrabackup: error: The transaction log file is corrupted.

xtrabackup: error: The log was not applied to the intended LSN!

xtrabackup: Log applied to lsn 2871364

xtrabackup: The intended lsn is 2883808

1. 此报错出现在同一批备份多次还原的情况下。
2. 意思为日志已经合并，可以直接进行数据恢复
3. 多次恢复后mysqld服务无法启动的原因
4. 出现在一次备份多次实验的情况下
5. 特点是无确切报错
6. 原因分析：
   1. 出现的原因应该是设定了自定义binlog目录为/logdir
   2. 在前面实验中，还原后再启动mysqld服务，系统会自动新建一个binlog日志，在排查第二个错误的时候，尝试着删除了第一次实验中创建的binlog5和6，但没有修改索引文件，binlog索引中依然存在5和6，索引和文件不匹配导致服务无法启动
7. DBA2
8. Day 01 MySQL主从同步 、 主从同步模式
9. MySQL主从同步
10. 主从同步概述

1) MySQL主从同步

· 对指定库的异地同步

· MySQL主-->从复制架构的实现

· MySQL服务器的只读控制

1. 一主、一从

- 单项复制时,建议将从库设为只读

client: mysql50

master: mysql51

slave: mysql52

1. 主从同步原理

· Master,记录数据更改操作

- 启用binlog日志

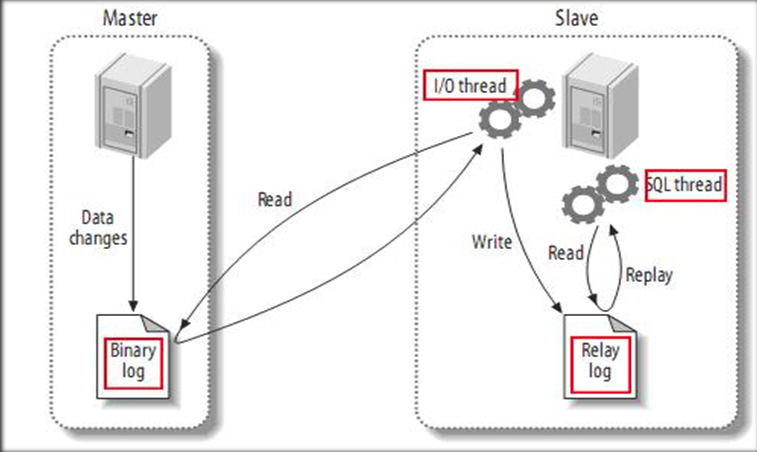
- 设置binliog日志格式

- 设置server\_id

· Slave运行两个线程

- Slave IO: 复制master主机binlog日志文件里的SQL到本机的relay-log文件里

- Slave\_SQL: 执行本机relay-log文件里的SQL语句,重现Master的数据操作



主从同步原理

1. 构建组从同步
2. 基本构建思路

· 确保数据相同

- 从库必须要有主库上的数据

· 配置主服务器

- 启用binlog日志、授权用户、查看当前正使用的日志

[root@mysql51 ~]# cat /etc/my.cnf

[mysqld]

log\_bin

server\_id=51

mysql> set global validate\_password\_policy=0;

mysql> set global validate\_password\_length=6;

mysql> grant replication slave on \*.\* to repluser@"%" identified by '123456';

· 配置从服务器

- 设置server\_id,指定主库信息

[root@mysql52 ~]# cat /etc/my.cnf

[mysqld]

log\_bin

server\_id=52

mysql> change master to

master\_host='192.168.4.51',

master\_user='repluser',

master\_password='123456',

master\_log\_file='mysql51-bin.000001',

master\_log\_pos=441;

Query OK, 0 rows affected, 2 warnings (0.29 sec)

mysql> start slave;

Query OK, 0 rows affected (0.07 sec)

/\* 修改配置之前需要先stop slave \*/

· 测试配置

- 客户端连接主库写入数据,在从库上也能查询到

mysql51:

mysql> grant all on \*.\* to root@'%' identified by '123456';

Query OK, 0 rows affected, 1 warning (0.04 sec)

mysql50:

[root@mysql50 ~]# mysql -h192.168.4.51 -uroot -p

Enter password:

mysql> create database testdb1;

Query OK, 1 row affected (0.04 sec)

mysql> create table testdb1.test(

-> name char(10));

Query OK, 0 rows affected (0.39 sec)

mysql> insert into testdb1.test values("zqy");

Query OK, 1 row affected (0.37 sec)

mysql52:

mysql> select \* from testdb1.test;

· 把从库还原成独立的数据库服务器

1. 相关文件

|  |  |
| --- | --- |
| 文件名 | 说明 |
| master.info | 主库信息 |
| relay-log.info | 中继日志信息 |
| 主机名-relay-bin.xxxxxx | zhong日志 |
| 主机名-relay-bin.index | 索引文件 |

b. 还原步骤

[root@mysql52 mysql]# cd /var/lib/mysql

[root@mysql52 mysql]# rm {\*relay\*,\*master\*}

rm：是否删除普通文件 "mysql52-relay-bin.000001"？y

rm：是否删除普通文件 "mysql52-relay-bin.000002"？y

rm：是否删除普通文件 "mysql52-relay-bin.index"？y

rm：是否删除普通文件 "relay-log.info"？y

rm：是否删除普通文件 "master.info"？y

1. 常用配置选项
2. 主库配置选项

· 适用于Master服务器

|  |  |
| --- | --- |
| 选项 | 用途 |
| binlog\_do\_db=name | 设置Master对哪些库记日志 |
| binlog\_ignore\_db=name | 设置Master对哪些库不记日志 |

1. 从库配置选项

· 适用于Slave服务器

|  |  |
| --- | --- |
| 选项 | 用途 |
| log\_slave\_updates | 记录从库更新,允许链式复制 |
| relay\_log=dbsvr2-relay-bin | 指定中继日志文件名 |
| replicate\_do\_db=mysql | 仅复制指定库,其他库将被忽略,此选项可设置多条(省略时复制所有库) |
| replicate\_ignore\_db=test | 不复制哪些库,其他库将被忽略,ignore-db与do-db只需选用其中一种 |

51 52 53(主-从-从)

配置53仅同步52的teadb库数据

53:

[root@mysql53 ~]# vim /etc/my.cnf

[mysqld]

replicate\_do\_db=teadb

[root@mysql53 ~]# systemctl restart mysqld

mysql> show slave status\G

配置主库51不允许同步本机的gamedb数据

51:

[root@mysql51 ~]# vim /etc/my.cnf

[mysqld]

binlog\_ignore\_db=gamedb

[root@mysql51 ~]# systemctl restart mysqld

mysql> show master status;

1. 主从同步模式
2. 结构类型
3. 主从同步结构

· 基本应用

- 单项复制: 主 --> 从

· 扩展应用

- 链式复制: 主 --> 从 --> 从

- 互为主从: 主 <--> 主

- 一主多从: 从<-- 主 --> 从

1. 配置mysql一主多从同步

· 给主机53配置两个从库,分别是54和55

配置主库53

1. 启用binlog日志

[root@mysql53 ~]# vim /etc/my.cnf

[mysqld]

log\_bin=master53

server\_id=53

validate\_password\_policy=0

validate\_password\_length=6

1. 用户授权

mysql> grant replication slave on \*.\* to repluser@'%' identified by '123456';

Query OK, 0 rows affected, 1 warning (0.05 sec)

1. 查看binlog日志信息

mysql> show master status;

配置从库54/55

1. 指定server\_id

[root@mysql54 ~]# vim /etc/my.cnf

[mysqld]

log\_bin

server\_id=54

validate\_password\_policy=0

validate\_password\_length=6

1. 指定主库信息

mysql> change master to

master\_host='192.168.4.53',

master\_user='repluser',

master\_password='123456',

master\_log\_file='master53.000001',

master\_log\_pos=701;

1. 启动slave

mysql> start slave;

Query OK, 0 rows affected (0.07 sec)

1. 查看状态信息

mysql> show slave status\G

1. 配置主从从结构

主从从结构中,做主库的从库需要设置级联复制:

[mysqld]

log\_slave\_updates

1. 复制模式
2. 复制模式介绍

· 异步复制

- 主库执行完一次事务后,立即将结果返给客户端,不关心从库是否已经接受并处理

· 全同步复制

- 当主库执行完一次事务,切所有从库都执行了该事务后才返回给客户端

· 半同步复制

- 介于异步复制和全同步复制之间

- 主库在执行完一次事务后,等待至少一个从库接收到并写到relay log中才返回给客户端

1. 模式配置

· 启用半同步复制模式

1. 查看当前的数据库服务器是否允许动态加载模块

mysql> show variables like 'have\_dynamic\_loading';

1. 加载模块

//主库

mysql> install plugin rpl\_semi\_sync\_master soname "semisync\_master.so";

//从库

mysql> install plugin rpl\_semi\_sync\_slave soname "semisync\_slave.so";

//查看

mysql> show variables like 'rpl\_semi%enabled';

1. 启用半同步复制模式

//主库

mysql> set global rpl\_semi\_sync\_master\_enabled=1;

//从库

mysql> set global rpl\_semi\_sync\_slave\_enabled=1;

· 永久启用半同步复制

- 需要修改到主配置文件/etc/my.cnf

- 添加相关设置到[mysqld]部分

//主库

pllugin-lodad=rpl\_semi\_sync\_master=semisync\_master.so

rpl\_semi\_sync\_master\_enabled=1

//从库

plugin-load=rpl\_semi\_slave=semisync\_slave.so

rpl\_semi\_sync\_slave\_enabled=1

/\* 如果一台数据库服务器既做主库又做从库,两个模块都要启用 \*/

1. Day 02 MySQL读写分离 MySQL多实例 、 MySQL性能调优
2. MySQL读写分离
3. 读写分离概述
4. MySQL读写分离

· 数据读写分离: 把客户端的访问数据的查询请求和写数据的请求分别给不同的数据库服务器处理

· 数据读写分离实现方式:

1. 通过程序实现: 明确告诉程序员查询操作和写数据的操作必须连接固定的数据库服务器
2. 架设服务结构实现:使数据读写分离服务的软件(统称为mysql中间件),比如: maxscale mysql-proxy mycat

· 主从复制的应用局限性

· 分离MySQL读写流量

· 在客户端区分还是在服务端区分

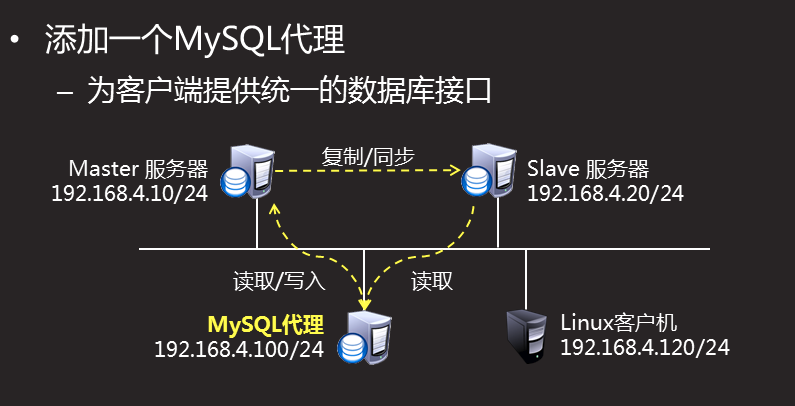
1. 案例拓扑

client: mysql50

server\_master: mysql51

server\_slave: mysql52

proxy: mysql57



1. 读写分离原理

· 多台MySQL服务器

- 分别提供读写服务,均衡流量

- 通过主从复制保持数据一致性

· 由MySQL代理面向客户端

- 收到SQL写请求时交给服务器A处理

- 收到SQL读请求时交给服务器B处理

- 具体区分策略由服务设置

1. 构建读写分离
2. 构建思路
3. 已经搭建好MySQL主从复制

- 基于上一个实验结果

- 其中Slave为只读

b. 添加一台MySQL代理服务器

(1) 配置MySQL代理服务器192.168.4.57

[root@mysql57 ~]# systemctl stop mysqld

[root@mysql57 ~]# systemctl disable mysqld

c. 客户端通过代理主机访问MySQL数据库

- 访问代理服务器

1. 部署maxscale服务

· MaxScale代理软件

- 由MySQL的兄弟公司MariaDB开发

- 下载地址

<https://downloads.mariadb.com/files/Maxscale>

[root@mysql57 ~]# rpm -ivh maxscale-2.1.2-1.rhel.7.x86\_64.rpm

· 修改配置/etc/maxscale.cnf

[maxscale]

threads=auto

[server1]

//定义数据库服务器主机名

type=server

address=192.168.4.51

//master主机IP地址

port=3306

protocol=MySQLBackend

[server2]

//定义数据库服务器

type=server

address=192.168.4.52

//slave主机IP地址

port=3306

protocol=MySQLBackend

[MySQL Monitor]

//定义要监视的数据库服务器

type=monitor

module=mysqlmon

servers=server1,server2

user=monitoruser

//监控用户名

passwd=123456

//密码

monitor\_interval=10000

#[Read-Only Service]

//不定义只读服务

#type=service

#router=readconnroute

#servers=server1

#user=myuser

#passwd=mypwd

#router\_options=slave

[Read-Write Service]

type=service

router=readwritesplit

servers=server1,server2

user=scaluser

//定义路由用户

passwd=123456

//定义路由密码

max\_slave\_connections=100%

[MaxAdmin Service]

//定义管理服务

type=service

router=cli

#[Read-Only Listener]

#type=listener

#service=Read-Only Service

#protocol=MySQLClient

#port=4008

[Read-Write Listener]

//定义读写分离服务使用的端口

type=listener

service=Read-Write Service

protocol=MySQLClient

port=4006

[MaxAdmin Listener]

//定义管理服务使用的端口

type=listener

service=MaxAdmin Service

protocol=maxscaled

socket=default

port=4016

根据配置文件做相应的配置

1. 在两台数据库服务器上创建监控用户monitoruser

mysql> grant replication slave,replication client on \*.\* to monitoruser@'%' identified by '123456';

Query OK, 0 rows affected, 1 warning (0.02 sec)

1. 在两台数据服务器上创建路由用户scaluser

mysql> grant select on \*.\* to scaluser@'%' identified by '123456';

Query OK, 0 rows affected, 1 warning (0.03 sec)

1. 启动服务

[root@mysql57 ~]# maxscale -f /etc/maxscale.cnf

[root@mysql57 ~]# maxscale --config=/etc/maxscale.cnf

[root@mysql57 ~]# ps -C maxscale

[root@mysql57 ~]# ss -antulp | grep maxscale

/\* 关闭服务 \*/

# killall -9 maxscale

/\* 查看状态 \*/

[root@mysql57 ~]# maxadmin -uadmin -pmariadb -P4016

MaxScale> list servers

1. 测试配置

· 在2台数据库服务器上添加客户端访问数据的连接用户

1. 在主库授权访问数据的连接用户jim，密码为123456

mysql> grant select,insert on \*.\* to jim@'%' identified by '123456';

Query OK, 0 rows affected, 1 warning (0.06 sec)

1. 在主库51上创建db2\_2库和t1表

mysql> create database db2\_2;

mysql> create table db2\_2.t1( name char(10));

Query OK, 0 rows affected (0.74 sec)

1. 在客户端50上插入数据测试

[root@mysql50 ~]# mysql -hmysql57 -ujim -p123456 -P4006

mysql> insert into db2\_2.t1 values("Jim");

Query OK, 1 row affected (1.01 sec)

1. 在51，52上查看是否同步

mysql> select \* from db2\_2.t1;

1. 验证select是否在从库上进行

mysql52:

mysql> insert into db2\_2.t1 values("mysql52");

Query OK, 1 row affected (0.74 sec)

mysql50:

mysql> select \* from db2\_2.t1;

1. MySQL多实例
2. 多实例概述
3. 什么是多实例

· 在一台物理主机上运行多个数据库服务

1. 为什么要使用多实例

· 节省运维成本

· 提高硬件利用率

1. 配置多实例
2. 配置步骤
3. 安装支持多实例服务的软件包

[root@mysql50 ~]# tar xf mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

[root@mysql50 ~]# mv mysql-5.7.20-linux-glibc2.12-x86\_64 /usr/local/mysql

[root@mysql50 ~]# vim /etc/profile

[root@mysql50 ~]# source /etc/profile

1. 修改主配置文件

[root@mysql50 ~]# vim /etc/my.cnf

[mysqld\_multi]

mysqld = /usr/local/mysql/bin/mysqld\_safe

mysqladmin = /usr/local/mysql/bin/mysqladmin

user=root

[mysqld1]

datadir=/dir1

log-error=/dir1/mysqld.log

socket=/dir1/mysqld.sock

pid-file=/dir1/mysqld.pid

port=3307

[mysqld2]

datadir=/dir2

log-error=/dir2/mysqld.log

socket=/dir2/mysqld.sock

pid-file=/dir2/mysqld.pid

port=3308

1. 初始化授权库

/\* 初始化授权库 \*/

./mysqld --user=mysql --basedir=软件安装目录 --datadir=数据库目录 --initalize

1. 启动服务

/\* 直接启动服务会自动初始化 \*/

[root@mysql50 ~]# mysqld\_multi start 1

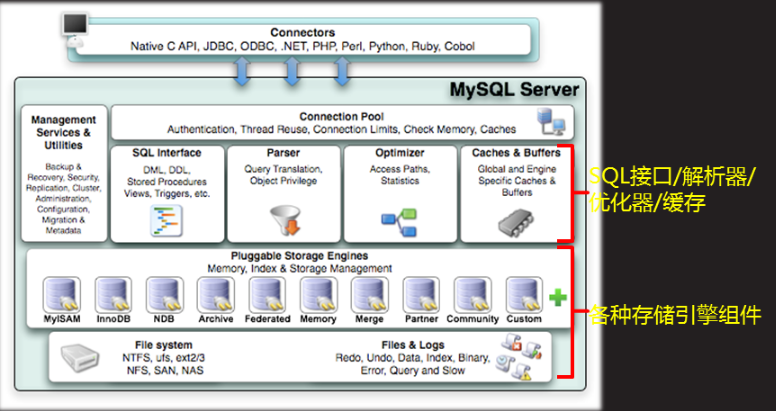
1. 客户端访问测试

[root@mysql50 ~]# mysql -uroot -p'8(=-/ia\*3ioP' -S /dir1/mysqld.sock

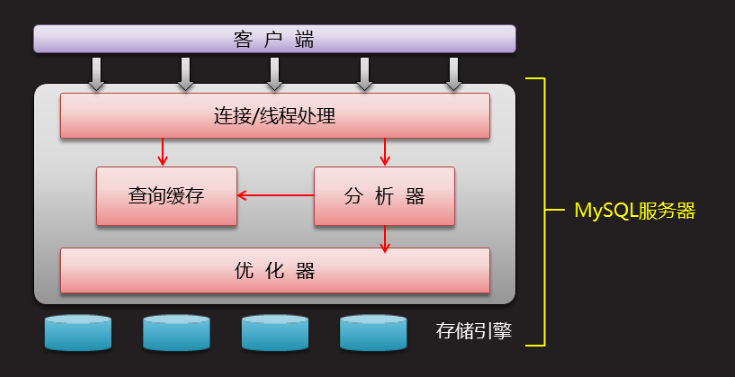
//mysql初始密码在服务启动(初始化时会提示)

1. MySQL性能调优
2. MySQL工作原理
3. MySQL体系结构

· MySQL体系结构



1. MySQL查询过程



1. MySQL性能调优
2. MySQL性能调优

· 提高MySQL系统的性能、响应速度的方法：

- 替换有问题的硬件(CPU/磁盘/内存等)

- 服务程序的运行参数调整

- 对SQL查询进行优化

1. 并发及连接控制

· 连接数、连接超时

|  |  |
| --- | --- |
| 选项 | 含义 |
| max\_connections | 允许的最大并发连接数 |
| connect\_timeout | 等待连接超时，默认10秒，仅登录时有效 |
| wait\_timeout | 等待关闭连接的不活动超时秒数，默认28800秒(8小时) |

· 查看当前已使用的连接数

mysql> show global status like "max\_used\_connections";

· 查看默认的最大连接数

mysql> show variables like "max\_connections";

/\* 已使用/最大 理想比率<=85% \*/

1. 缓存参数控制

· 缓存区、线程数量、开表数量

|  |  |
| --- | --- |
| 选项 | 含义 |
| key\_buffer\_size | 用于MyISAM引擎的关键索引缓存大小 |
| sort\_buffer\_size | 为每个要排序的线程分配此大小的缓存空间 |
| read\_buffer\_size | 为顺序读取表记录保留的缓存大小 |
| thread\_cache\_size | 允许保存在缓存中被重用的线程数量 |
| table\_open\_cache | 为所有线程缓存的打开的表的数量 |

· key\_buffer\_size=8M

- 当Key\_reads / Key\_read\_requests较低时

- 可适当加大此缓存值

mysql> show global status like "key\_read%";

mysql> show variables like "key\_buffer\_size";

· sort\_buffer\_size=256K

- 增大此值可提高order和group的速度

mysql> show variables like "sort\_buffer\_size";

· 查看表记录读取缓存

- 此缓存值影响SQL查询的响应速度

mysql> show variables like "read\_%\_size";

· 查看可重用线程数

mysql> show variables like "thread\_%\_size";

· 查看当前的线程重用状态

mysql> show global status like "threads\_%";

· 查看已打开、打开过多少个表

mysql> show global status like "open%tables";

· 查看可缓存多少个打开的表

mysql> show variables like "table\_open\_cache";

/\* 已缓存/可缓存理想比率<=95% \*/

1. SQL查询优化
2. MySQL日志类型

· 常用日志种类及选项

|  |  |  |
| --- | --- | --- |
| 类型 | 用途 | 配置 |
| 错误日志 | 记录启动/运行/停止过程中的错误消息 | log-error[=name] |
| 查询日志 | 记录客户端连接和查询操作 | general-log  general-log-file= |
| 慢查询日志 | 记录耗时较长或不适用索引的查询操作 | slow-query-log  slow-query-log-file=  long-query-time= |

1. 优化SQL查询

· 记录慢查询

|  |  |
| --- | --- |
| 选项 | 含义 |
| slow-query-log | 启用慢查询(查询超时的记录会被记录到慢查询日志中) |
| slow-query-log-file | 指定慢查询日志文件 |
| log-query-time | 超过时间(默认10秒) |
| log-queries-not-using-indexes | 记录未使用索引的查询 |

1. 关于查询的缓存

· 查看缓存的大小

mysql> show variables like "query\_cache%";

· 查看当前的查询缓存统计

mysql> show global status like "qcache%";

1. 优化数据存储架构(查看是否具有数据传输瓶颈和单店故障)
2. 检测网络流量(带宽)
3. Day 03 MHA集群概述 、 部署MHA集群 测试配置
4. MHA集群概述
5. MHA介绍
6. 集群的定义：

· 多台服务器一起提供相同的服务

LB 多台一起提供服务 LVS Haproxy Nginx

HA 主备方式提供服务 keepalived

1. MHA简介

· 由日本DeNA公司youshimaton使用perl语言开发

· 是一套优秀的实现MySQL高可用的解决方案

· 数据库的自动故障切换操作能做到在0~30s之内

· MHA能确保在故障切换过程中保证数据的一致性，以达到真正意义上的高可用

1. MHA组成

· MHA Manager(管理节点)

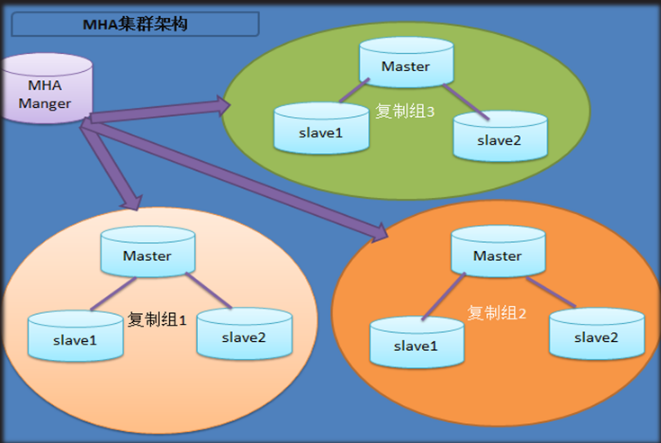
- 可以单独部署在一台独立的机器上

- 也可以部署在一台slave节点上

· MHA Node(数据节点)

- 运行在每台MySQL服务器上

1. MHA工作过程
2. MHA集群架构



MHA集群架构

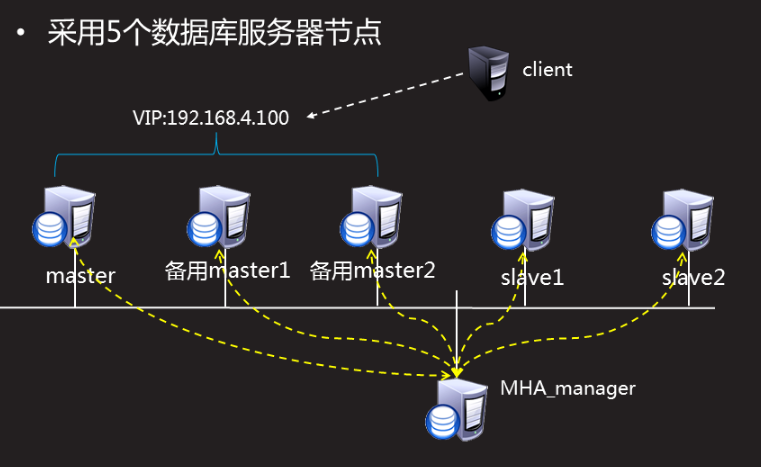
1. MHA工作过程

· 由Manager定时探测集群中的master节点

· 当master故障时，Manager自动将拥有最新数据的slave提升为新的master

· 关键点

1. 从宕机崩溃的master保存二进制日志事件
2. 识别含有最新更新的slave
3. 应用差异的中继日志( relay log )到其他的slave
4. 应用从master保存的二进制日志事件
5. 提升一个slave为新的master
6. 使其他的slave连接新的master进行复制
7. 部署MHA集群
8. 拓扑结构
9. 拓扑图



1. IP规划

mysql51：Master主节点服务器

mysql52：备用1主节点服务器

mysql53：备用2主节点服务器

mysql54：第一台slave服务器

mysql55：第二台slave服务器

mysql56：MHA\_manager服务器

VIP地址：192.168.4.100

1. 准备集群环境
2. 安装软件包

· 所有主机上安装Perl依赖包

# cd mha;yum -y install ./perl-\*

· 在所有数据节点上授权监控用户

· 在所有数据库服务器上安装mha-node包

· 在管理主机上安装mha\_node和mha-manager包

1. 配置ssh密钥对认证登录

· 所有节点之间互相以root密钥配对认证登录

· 管理主机以root密钥对认证登录所有数据节点主机

1. 相关命令
2. 配置MHA集群
3. 配置master主节点

· master51数据库服务器配置文件

mysql> set global relay\_log\_purge=off;

mysql> install plugin rpl\_semi\_sync\_master soname "semisync\_master.so";

mysql> install plugin rpl\_semi\_sync\_slave soname "semisync\_slave.so";

mysql> set global rpl\_semi\_sync\_master\_enabled=1;

mysql> set global rpl\_semi\_sync\_slave\_enabled=1;

[root@mysql51 ~]# vim /etc/my.cnf

[mysqld]

log\_bin

server\_id=51

relay\_log\_purge=off

plugin-load="rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so"

rpl\_semi\_sync\_master\_enabled=1;

rpl\_semi\_sync\_slave\_enabled=1;

1. 配置备用1主节点

mysql> set global relay\_log\_purge=off;

mysql> install plugin rpl\_semi\_sync\_master soname "semisync\_master.so";

mysql> install plugin rpl\_semi\_sync\_slave soname "semisync\_slave.so;";

mysql> set global rpl\_semi\_sync\_master\_enabled=1;

mysql> set global rpl\_semi\_sync\_slave\_enabled=1;

[root@mysql52 ~]# vim /etc/my.cnf

[mysqld]

log\_bin

server\_id=52

relay\_log\_purge=off

plugin-load="rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so"

rpl\_semi\_sync\_master\_enabled=1;

rpl\_semi\_sync\_slave\_enabled=1;

1. 配置备用2主节点

mysql> set global relay\_log\_purge=off;

mysql> install plugin rpl\_semi\_sync\_master soname "semisync\_master.so";

mysql> install plugin rpl\_semi\_sync\_slave soname "semisync\_slave.so;";

mysql> set global rpl\_semi\_sync\_master\_enabled=1;

mysql> set global rpl\_semi\_sync\_slave\_enabled=1;

[root@mysql53 ~]# vim /etc/my.cnf

[mysqld]

log\_bin

server\_id=53

relay\_log\_purge=off

plugin-load="rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so"

rpl\_semi\_sync\_master\_enabled=1;

rpl\_semi\_sync\_slave\_enabled=1;

1. 配置2台slave服务器

mysql> set global relay\_log\_purge=off;

mysql> install plugin rpl\_semi\_sync\_slave soname "semisync\_slave.so";

mysql> set global rpl\_semi\_sync\_slave\_enabled=1;

1. 配置管理主机
2. 装perl依赖

# cd;yum -y install perl-ExtUtils-\* perl-CPAN-\*

1. 装包

[root@mysql56 mha4mysql-manager-0.56]# perl Makefile.PL;make && make install

[root@mysql56 mha\_manager]# mkdir /etc/mha\_manager;cp samples/conf/app1.cnf /etc/mha\_manager/

1. 改配置文件

[root@mysql56 mha\_manager]# vim app1.cnf

bash

[server default]

manager\_workdir=/etc/mha\_manager

manager\_log=/etc/mha\_manager/manager.log

master\_ip\_failover\_script=/etc/mha\_manager/master\_ip\_failover

ssh\_user=root

ssh\_port=22

repl\_user=repluser

repl\_password=123456

user=root

password=123456

[server1]

hostname=192.168.4.51

candidate\_master=1

[server2]

hostname=192.168.4.52

candidate\_master=1

[server3]

hostname=192.168.4.53

candidate\_master=1

[server4]

hostname=192.168.4.54

no\_master=1

[server5]

hostname=192.168.4.55

no\_master=1

1. 在数据库上创建root@'%'授权

mysql> grant all on \*.\* to root@'%' identified by '123456';

1. 在备用主库服务器52~53添加 添加同步数据用户repluser

mysql> grant replication slave on \*.\* to repluser@'%' identified by '123456';

1. 测试配置
2. 测试集群配置
3. 测试ssh密钥对认证登录

[root@mysql56 mha\_manager]# masterha\_check\_ssh --conf=/etc/mha\_manager/app1.cnf

1. 测试主从同步状态

[root@mysql56 mha\_manager]# masterha\_check\_repl --conf=/etc/mha\_manager/app1.cnf

1. 启动MHA\_Manager

· 使用masterha\_manager工具

--remove\_dead\_master\_conf //删除宕机主库配置

--ignore\_last\_failover //忽略xxx.health文件

[root@mysql56 ~]# masterha\_manager --conf=/etc/mha\_manager/app1.cnf \

--remove\_dead\_master\_conf --ignore\_last\_failover

· 客户端50连接vip地址100访问数据库服务

[root@mysql50 ~]# mysql -h192.168.4.100 -uroot -p123456

mysql> create database gamedb;

mysql> create table gamedb.t1(id int);

mysql> insert into gamedb.t1 values(1551);

mysql> select \* from gamedb.t1;

· 查看状态：masterha\_check\_staus

# masterha\_check\_status --conf=/etc/mha\_manager/app1.cnf

· 停止服务：masterha\_stop

# masterha\_stop --conf=/etc/mha\_manager/app1.cnf

1. 测试高可用配置
2. 修改故障切换脚本

[root@mysql56 ~]# vim /etc/mha\_manager/master\_ip\_failover

my $vip = '192.168.4.100/24'; # Virtual IP

my $key = "1";

my $ssh\_start\_vip = "/sbin/ifconfig eth0:$key $vip";

my $ssh\_stop\_vip = "/sbin/ifconfig eth0:$key down";

1. 查看VIP地址

· 当主库服务器宕机后，在备用1主节点查看VIP地址

1. 停止当前主库51的数据库服务

[root@mysql51 ~]# systemctl stop mysqld

1. 查看从库53的状态

mysql> show slave status\G

Master\_Host: 192.168.4.52

1. 在新主库上查看vip地址

[root@mysql52 ~]# ip a s | grep 4.100

inet 192.168.4.100/24 brd 192.168.4.255 scope global secondary eth0:1

1. Day 04 MySQL视图 MySQL存储过程
2. MySQL视图
3. 视图概述
4. 视图介绍

· 什么是视图

- 虚拟表

- 内容与真实的表相似，有字段有记录

- 视图并不在数据库中以存储的数据形式存在

- 行和列的数据来自定义视图时查询所引用的基表，并且在具体引用视图时动态生成

- 更新视图的数据，就是更新基表的数据

- 更新基表的数据，视图的数据也会跟着改变

1. 视图优点

· 简单

- 用户不需关心视图中的数据如何查询获得

- 视图中的数据已经是过滤好的复合条件的结果集

· 安全

- 用户只能看到视图中的数据

· 数据独立

- 一旦视图结构确定，可以屏蔽表结构对用户的影响

1. 视图使用限制

· 不能在视图上创建索引

· 在视图的FROM子句中不能使用子查询

· 以下情形中的视图是不可更新的

- 包含以下关键字的SQL语句：聚合函数(sum,min,max,count等)、distinct,group by,having,union或union all

- 常量视图，join，from一个不能更新的视图

- where子句的子查询引用了from子句中的表

- 使用了临时表

1. 视图基本使用
2. 创建视图

create view 视图名称 as SQL查询;

create view 视图名称(字段名列表) as SQL查询;

1. 查看视图

show table status;

show table status where comment="view"\G

1. 使用视图

· 查询记录

select 字段名列表 from 视图名 where 条件

· 插入记录

insert into 视图名(字段名列表) values(字段值列表)

· 更新记录

update 视图名 set 字段名=值 where 条件;

· 删除记录

delete from 视图名 where 条件

1. 删除视图

drop view 视图名;

1. 视图进阶
2. 创建视图的完全格式

create [or replace] [algorithm={undefined|merge|temptable}] [definer={user|current\_user}] [sql security{definer|invoker}] view view\_name [(column\_list)] as select\_statement [with [cascaded|local ] check option]

1. 设置字段别名

create view 视图名

as

select 表别名.源字段名 as 字段别名

from 源表名 表别名 left join 源表名 表别名

on 条件;

1. 重要选项说明

· or replace

- create or replace view 视图名 as select 查询;

- 创建时，若视图已存在，会替换已有的视图

· local和cascaded关键字决定检查范围

- local仅检查当前视图的限制

- cascaded同时要满足基表的限制

1. MySQL存储过程
2. 存储过程概述
3. 存储过程介绍

· 存储过程，相当于是MySQL语句组成的脚本

- 指的是数据库中保存的一系列SQL命令的集合

- 可以在存储过程中使用变量、条件判断、流程控制等

1. 存储过程优点

· 提高性能

· 减轻网络负担

· 防止对表的直接访问

· 避免重复编写SQL操作

1. 基本使用
2. 创建存储过程

delimiter //

create procedure 名称()

begin

功能代码

end

//

delimiter ;

1. 查看存储过程

//方法1

mysql> show procedure status;

//方法2

mysql> select db,name,typpe from mysql.proc

where name="存储过程名";

1. 调用/删除存储过程

· 调用存储过程

call 存储过程名();

· 删除存储过程

drop procedure 存储过程名;

1. 存储过程进阶
2. 变量类型

|  |  |
| --- | --- |
| 名称 | 描述 |
| 会话变量 | 会话变量和全局变量叫系统变量，使用set定义  select @@hostname |
| 全局变量 | 全局变量的修改会影响到整个服务器，但是对话变量的修改，只会影响到当前的会话。 |
| 用户变量 | 在客户端连接到数据库服务的整个过程中都是有效的。当当前连接断开后所有用户变量失效。  set @变量名=值; //定义  select @变量名; //输出 |
| 局部变量 | 存储过程中的begin/end。其有效范围仅限于该语句块中，语句块执行完毕后，变量失效  declare 定义局部变量 |
| 注意：局部变量和参数变量调用时，变量名前不需要加@ | |

1. 参数类型

· 调用参数时，名称前也不需要加@

create procedure 名称(

类型 参数名 数据类型，

类型 参数名 数据类型

)

|  |  |  |
| --- | --- | --- |
| 关键字 | 名称 | 描述 |
| in | 输入参数 | 作用是给存储过程传值，必须在调用存储过程时赋值，在存储过程中该参数的值不允许修改;默认值类型值是in |
| out | 输出参数 | 该值可在存储过程内部被改变，并可返回 |
| inout | 输入/输出参数 | 调用时指定，并且可被改变和返回 |

1. 运算符号及用法示例

|  |  |  |
| --- | --- | --- |
| 符号 | 描述 | 示例 |
| + | 加法运算 | set @var1=2+2; 4 |
| - | 减法运算 | set @var2=3-2; 1 |
| \* | 乘法运算 | set @var3=3\*2; 6 |
| / | 除法运算 | set @var4=10/3; 3.33333 |
| DIV | 整除运算 | set @var5=10 DIV 3; 3 |
| % | 取余运算 | set @var6=10%3; 1 |

1. 流程控制
2. 条件测试
3. 数值比较

|  |  |
| --- | --- |
| 类型 | 用途 |
| = | 等于 |
| >、>= | 大于、大于或等于 |
| <、<= | 小于、小于或等于 |
| != | 不等于 |
| between...and... | 在..与..之间 |

1. 逻辑比较、范围、空、非空、模糊、正则

|  |  |
| --- | --- |
| 类型 | 用途 |
| or、and、! | 逻辑或、逻辑与、逻辑非 |
| in..、not in.. | 在..范围内、不在..范围内 |
| is null | 字段的值为空 |
| is not null | 字段的值不为空 |
| like | 模糊匹配 |
| regexp | 正则 |

1. 选择结构
2. 单分支选择结构

· 当条件成立时执行命令序列

1. 双分支选择结构

· 单条件成立时执行代码1，否则执行代码2

1. 循环结构
2. while条件式循环

· 反复测试条件，只要成立就执行命令序列

while 条件判断 do

循环体

..

end while;

1. loop死循环

· 无条件、反复执行某一段代码

loop

循环体

.. ..

end loop;

1. repeat条件式循环

· 当条件成立时结束循环

repeat

循环体

.. ..

until 条件判断

end repeat;

delimiter //

create procedure p4(in y int)

begin

declare x int;

set x =1;

repeat

select x;

set x = x + 1;

until x > y

end repeat;

end

//

delimiter ;

1. 控制循环的执行

- LEAVE 标签名 //跳出循环

- ITERATE 标签名 //放弃本次循环，执行下一次循环

mysql> delimiter //

mysql> create procedure p7()

-> begin

-> declare x int;

-> set x = 1;

-> abc:while x <= 10 do

-> LEAVE abc;

-> select x;

-> set x=x+1;

-> end while;

-> end

-> //

mysql> delimiter ;

mysql> create procedure p9() begin declare x int; set x=1; bcd:while x <= 10 do if x=5 or x=6 then set x=x+1; iterate bcd; end if; seleect x; set x=x+1; end while; end//

mysql> delimiter ;

1. 案例：

使用循环结构满足以下要求：

· 定义名称为p3的存储过程

· 用户可以自定义显示user表记录的行数

· 若调用时用户没有输入行数，默认显示第1条记录

delimiter //

create procedure p3(in num int)

begin

if num is not null then

select \* from db9.user where id <=num

else

select \* from db9.user where id=1;

end if;

end

//

delimiter;

call p3(5);

select@x;

call p3(@x);

1. Day 05 分库分表概述 配置mycat
2. 分库分表概述
3. 相关概念
4. 分库/分表

· 解决单数据库服务器的访问压力和存储压力

· 解决单表过大

· 什么是分库分表

- 将存放在一个数据库(主机)中的数据，按照特定方式进行差分，分散存放到多个数据库(主机)中，以达到分散单台设备负载的效果

1. 水平分割(横向分割)

· 按照表中某个字段的某种规则，把表中的许多纪录按行切分，分散到多个数据库中

1. 垂直分割(纵向分割)

· 将单个表，拆分成多个表，分散到不同的数据库

· 将单个数据库的多个表进行分类，按业务类别分散到不同的数据库上

1. mycat介绍
2. 软件介绍

· mycat是基于Java的分布式数据库系统中间层，为高并发数环境的分布式访问提供解决方案

· mycat是一个中间件

- 支持JDBC形式连接

- 支持MySQL、Oracle、Sqlserver、Mongodb

- 提供数据读写分离服务

- 可以实现数据库服务器的高可用

- 提供数据分片服务

- 基于阿里巴巴Cobar进行研发的开源软件

- 适合数据大量写入的存储需求

1. 分片规则

· mycat支持提供10种分片规则

1. 枚举法 sharding-by-intfile
2. 固定分片rule1
3. 范围约定auto-sharding-long
4. 求模法 mod-long
5. 日期列分区法 sharding-by-date
6. 通配取模 sharding-by-pattern
7. ASCII码求模通配 sharding-by-prefixpattern
8. 编程指定 sharding-by-substring
9. 字符串拆分hash解析 sharding-by-stringhash
10. 一致性hash sharding-by-murmur
11. 工作过程

· 当mycat收到一个SQl查询时

1. 先解析这个SQL查找涉及到的表
2. 然后看此表的定义，如果有分片规则，则获取SQL里分片字段的值，并匹配分片函数，获得分片列表
3. 然后将SQL发往这些分片去执行



1. 配置mycat
2. 装包

[root@mysql53 ~]# yum install java-1.8.0-openjdk.x86\_64 -y

[root@mysql53 ~]# yum install java-1.8.0-openjdk-headless -y

[root@mysql53 ~]# tar xvf Mycat-server-1.4-beta-20150604171601-linux.tar.gz

1. 修改配置文件
2. 重要配置文件说明

- server.xml 设置了解mycat服务的账号、密码等

- schema.xml 配置mycat使用的真实数据库和表

- rule.xml 定义mycat分片规则

1. 配置标签说明

- <user>...</user> 定义连接mycat服务时使用的用户名和密码及逻辑库的名字

- <datanode>...</datanode> 指定数据节点(物理库的主机名和存储分片数据的数据库名)

- <datahost>...</datahost> 指定数据库服务器的IP地址及连接数据库时使用的授权用户名及密码

1. 定义客户端连接mycat服务使用的用户及密码和虚拟数据库名

[root@mysql53 conf]# vim /usr/local/mycat/conf/server.xml

<user name="user">

<property name="password">user</property>

<property name="schemas">TESTDB</property>

<property name="readOnly">true</property>

</user>

1. 配置数据分片(对哪些表做数据分片及使用的分片规则)

../conf/schema.xml

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100">

<table name="物理表名" dataNode="数据库服务器名称" rule="auto-sharding-long" />

</schema>

//定义分片信息

<dataNode name="dn1" dataHost="localhost1" database="db1" />

//定义数据库节点与要进行分库的真实数据库

[root@mysql53 conf]# cp schema.xml{,.bak}

[root@mysql53 conf]# sed -i '56,77d' schema.xml

[root@mysql53 conf]# sed -i '39,42d' schema.xml

<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100">

<!-- auto sharding by id (long) -->

<table name="travelrecord" dataNode="dn1,dn2" rule="auto-sharding-long" />

</schema>

//去掉默认的所有dn3

<dataNode name="dn1" dataHost="c1" database="db1" />

<dataNode name="dn2" dataHost="c2" database="db2" />

<dataHost name="c1" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="192.168.1.51:3306" user="root"

password="123456">

<!-- can have multi read hosts -->

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

<dataHost name="c2" maxCon="1000" minCon="10" balance="0"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<!-- can have multi write hosts -->

<writeHost host="hostM1" url="192.168.1.52:3306" user="root"

password="123456">

<!-- can have multi read hosts -->

</writeHost>

<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->

</dataHost>

1. 修改数据库服务器配置文件

- 添加对应设置后重启mysqld服务

- 添加授权用户

# vim /etc/my.cnf

[mysqld]

...

...

lower\_case\_table\_names = 1 #表名区分字母大小写

# systemctl restart mysqld

//mycat连接数据库服务器用户

> grant all on \*.\* to root@'%' identified by '123456';

1. 启动服务

[root@mysql53 conf]# /usr/local/mycat/bin/mycat start

1. 查看服务信息

[root@mysql53 conf]# /usr/local/mycat/bin/mycat status

Mycat-server is running (2273).

1. 客户端测试配置

# mysql -utest -ptest -h192.168.1.53 -P8066

//注意用户和密码是server.xml中定义的

mysql> show databases;

TESTDB

mysql> use TESTDB

mysql> show tables;

+------------------+

| Tables in TESTDB |

+------------------+

| company |

| customer |

| customer\_addr |

| employee |

| goods |

| hotnews |

| orders |

| order\_items |

| travelrecord |

+------------------+

9 rows in set (0.00 sec)

mysql> desc orders;

ERROR 1146 (42S02): Table 'db1.orders' doesn't exist

mysql> use TESTDB;

mysql> show create table employee;

CREATE TABLE `employee` (

`id` int(11) NOT NULL,

`name` varchar(100) DEFAULT NULL,

`age` int(2) DEFAULT NULL,

`sharding\_id` int(11) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

mysql> insert into employee values(1,'bob',21,10000),(2,'lucy',10010);

ERROR 1064 (HY000): partition table, insert must provide ColumnList

mysql> insert into employee(id,name,age,sharding\_id) values(1,'bob',21,10000),(2,'lucy',18,10010);

Query OK, 2 rows affected (0.01 sec)

# mysql -uroot -p123456

mysql> select \* from db2.employee;

+----+------+------+-------------+

| id | name | age | sharding\_id |

+----+------+------+-------------+

| 2 | lucy | 18 | 10010 |

+----+------+------+-------------+

1 row in set (0.00 sec)

1. NOSQL
2. Day 01 NoSQL概述、部署Redis服务、部署LNMP+Redis
3. NoSQL概述
4. 数据库类型
5. RDBMS
6. 关系数据库管理系统

- Relational Database Management System

- 按照预先设置的组织结构，将数据存储在物理介质上

- 数据之间可以做关联操作

1. RDBMS服务软件
2. 主流的RDBMS软件

- Oracle

- DB2

- Micrasoft SQL Server

- MySQL、MariaDB

1. NoSQL
2. NoSQL(NoSQL=Not Only SQL)

- 意思是"不仅仅是SQL"

- 泛指非关系型数据库

- 不需要预先定义数据存储结构

- 表的每条记录都可以有不同的类型和结构

1. NoSQL服务软件
2. 主流软件

- Redis

- MongoDB

- Memcached

- CouchDB

- Neo4j

- FlockDB

1. 部署Redis服务
2. 部署Redis服务
3. Redis介绍
4. Redis

- Remote Dictionary Server(远程字典服务器)

- 是一款高性能的(Key/Values)分布式内存数据库

- 支持数据持久化，可以把内存里的数据保存到硬盘中

- 也支持list、hash、set、zset数据类型

- 支持master-slave模式数据备份

- 中文网站www.redis.com

1. 装包
2. 从源码包编译安装

# tar xf redis-4.0.8.tar.gz

# cd redis-4.0.8

# make

# make install

//需要gcc编译器

/\*

装包时可能会出现问题，在执行初始化的过程中会出现报错提示"没有可执行的redis文件"系统中没有redis相关命令，make 与 make install执行失败，报错如下：

[root@mysql51 redis-4.0.8]# make

cd src && make all

make[1]: Entering directory `/root/redis-4.0.8/src'

CC adlist.o

In file included from adlist.c:34:0:

zmalloc.h:50:31: fatal error: jemalloc/jemalloc.h: No such file or directory

#include <jemalloc/jemalloc.h>

^

compilation terminated.

make[1]: \*\*\* [adlist.o] Error 1

make[1]: Leaving directory `/root/redis-4.0.8/src'

make: \*\*\* [all] Error 2

此时删掉redis安装目录，重新解压安装

\*/

1. 初始配置

[root@mysql51 redis-4.0.8]# cd utils/

[root@mysql51 utils]# ./install\_server.sh

(默认配置：一路回车)

1. 启动/停止服务

# /etc/init.d/redis\_6379 status | start | stop

# netstat -antulp | grep :6379

tcp LISTEN 0 128 127.0.0.1:6379 \*:\* users:(("redis-server",pid=4490,fd=6))

# ps -C redis-server

1. 连接Redis数据库服务

# redis-cli

//连接本地redis服务(127.0.0.1)不需要指定地址，且因没有配置所以不需要使用用户和密码

1. 常用操作指令

- set keyname keyvalue //存储

- get keyname //获取

- select 数据库编号0-15 //切换库

- keys \* //打印所有变量

- keys a? //打印指定变量

- EXISTS keyname //测试是否存在

- ttl keyname //查看生存时间

- type keyname //查看类型

- move keyname dbname //移动变量

- expire keyname 10 //设置有效时间

- del keyname //删除变量

- flushall //删除所有变量

- save //保存所有变量

- shutdown //关闭redis服务

1. 配置文件解析

· 配置文件位置

/etc/redis/6379.conf

1. 数据单位

1k => 1000 bytes

1kb => 1024 bytes

1m => 1000000 bytes

1mb => 1024\*1024 bytes

1g => 1000000000 bytes

1gb => 1024\*1024\*1024 bytes

1. 常用配置选项

- port 6379 //端口

- bind 127.0.0.1 //IP地址

- tcp-backlog 511 //tcp连接总数

- timeout 0 //连接超时时间

- tcp-keepalive 300 //长连接时间

- daemonize yes //守护进程方式运行

- databases 16 //数据库个数

- logfile /var/log/redis\_6379.log //pid文件

- maxclients 10000 //并发连接数量

- dir /varlib/redis/6379 //数据库目录

[root@mysql51 ~]# vim /etc/redis/6379.conf

bind 192.168.1.51

port 6050

[root@mysql51 ~]# /etc/init.d/redis\_6379 stop

[root@mysql51 ~]# /etc/init.d/redis\_6379 start

[root@mysql51 ~]# redis-cli

Could not connect to Redis at 127.0.0.1:6379: Connection refused

Could not connect to Redis at 127.0.0.1:6379: Connection refused

[root@mysql51 ~]# /etc/init.d/redis\_6379 stop

Stopping ...

Could not connect to Redis at 127.0.0.1:6379: Connection refused

Waiting for Redis to shutdown ...

[root@mysql51 ~]# redis-cli -h 192.168.1.51 -p 6050 shutdown

[root@mysql51 ~]# redis-cli -h 192.168.1.51 -p 6050

192.168.1.51:6050> ping

PONG

1. 内存管理
2. 内存清除策略

- volatile-lru //最近最少使用(针对设置了TTL的key)

- allkeys-lru //删除最少使用的key

- volatite-random //在设置了TTL的key里随机移除

- allkeys-random //随机移除key

- volatile-ttl(minor TTL) //移除最近过期的key

- noeviction //不删除，写满时报错

1. 选项默认设置

- maxmemory <bytes> // 最大内存

- maxmemory-policy noeviction //定义使用策略

- maxmemory-samples 5 //选取模板数据的个数(针对lru和ttl策略)

1. 设置连接密码

[root@mysql52 ~]# grep -n requirepass /etc/redis/6379.conf

284:# If the master is password protected (using the "requirepass" configuration

501:# requirepass foobared

[root@mysql52 ~]# sed -i '501s/^#//;501s/foobared/123456/' /etc/redis/6379.conf

//去掉501行的注释，requirepass后接密码字段

[root@mysql52 ~]# /etc/init.d/redis\_6379 stop

[root@mysql52 ~]# /etc/init.d/redis\_6379 start

[root@mysql52 ~]# redis-cli

127.0.0.1:6379> ping

(error) NOAUTH Authentication required.

127.0.0.1:6379> auth 123456

OK

127.0.0.1:6379> ping

PONG

//或者在连接redis服务时使用-a选项

[root@mysql52 ~]# redis-cli -a 123456

127.0.0.1:6379> ping

PONG

[root@mysql52 ~]# /etc/init.d/redis\_6379 stop

Stopping ...

(error) NOAUTH Authentication required.

Waiting for Redis to shutdown ...

Waiting for Redis to shutdown ...

//此时用脚本停停不下来，需要自行修改脚本

* 1. 部署LNMP+Redis

1. 部署LNMP
2. 安装软件包

[root@mysql52 ~]# yum install -y zlib-devel

[root@mysql52 ~]# cd nginx-1.12.2/

[root@mysql52 ~]# ./configure

[root@mysql52 ~]# make && make install

//装nginx

[root@mysql52 ~]# rpm -qa | grep php

php-fpm-5.4.16-42.el7.x86\_64

php-common-5.4.16-42.el7.x86\_64

php-devel-5.4.16-42.el7.x86\_64

php-mysql-5.4.16-42.el7.x86\_64

php-cli-5.4.16-42.el7.x86\_64

php-pdo-5.4.16-42.el7.x86\_64

//装php

[root@mysql52 ~]# yum install mysql-community-\* -y

//装mysql

1. 修改配置文件

[root@mysql52 ~]# vim /usr/local/nginx/conf/nginx.conf

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

include fastcgi.conf;

}

1. 启动服务

[root@mysql52 ~]# /usr/local/nginx/sbin/nginx

[root@mysql52 ~]# systemctl start php-fpm

[root@mysql52 ~]# systemctl enable php-fpm

1. 测试配置

[root@mysql52 ~]# vim /usr/local/nginx/html/test.php

<?php

echo "hello boy";

?>

[root@mysql52 ~]# curl http://mysql52/test.php

1. PHP+Redis支持
2. 运行Redis服务

[root@mysql51 ~]# /etc/init.d/redis\_6379 start

1. 配置php支持Redis
2. 安装php-redis模块

[root@mysql51 ~]# tar xf php-redis-2.2.4.tar.gz

[root@mysql51 ~]# cd phpredis-2.2.4/

[root@mysql51 phpredis-2.2.4]# phpize

Configuring for:

PHP Api Version: 20100412

Zend Module Api No: 20100525

Zend Extension Api No: 220100525

//注意必须要在phpredis-2.2.4目录下才能执行成功看到以上信息

[root@mysql51 phpredis-2.2.4]# ls /usr/bin/php-config

/usr/bin/php-config

[root@mysql51 phpredis-2.2.4]# ./configure --with-php-config=/usr/bin/php-config

[root@mysql51 phpredis-2.2.4]# make && make install

...

Installing shared extensions: /usr/lib64/php/modules/

//php连接redis的模块位置/usr/lib64/php/modules/

1. 修改php主配置文件

[root@mysql51 phpredis-2.2.4]# vim /etc/php.ini

728 extension\_dir = "/usr/lib64/php/modules/"

730 extension = "redis.so"

1. 启动服务，写测试页面

[root@mysql51 phpredis-2.2.4]# systemctl restart php-fpm.service

[root@mysql51 phpredis-2.2.4]# php -m | grep redis

redis

[root@mysql51 phpredis-2.2.4]# vim /usr/local/nginx/html/linkRedis.php

<?php

$redis = new redis();

$redis->connect('127.0.0.1',6379);

$redis->set('redistest','666666');

echo $redis->get('redistest');

?>

[root@mysql51 phpredis-2.2.4]# ss -antulp | grep :80

[root@mysql51 phpredis-2.2.4]# nginx

1. 测试配置

[root@mysql51 phpredis-2.2.4]# redis-cli

127.0.0.1:6379> get redistest

(nil)

[root@mysql51 phpredis-2.2.4]# curl <http://192.168.1.51/linkRedis.php>

666666

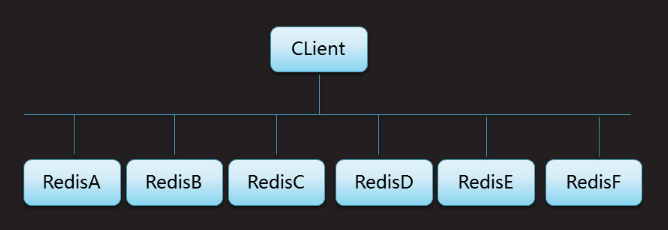
[root@mysql51 phpredis-2.2.4]# redis-cli

127.0.0.1:6379> get redistest

"666666"

1. Day 02 创建Redis集群、管理集群
2. 创建Redis集群
3. 准备集群环境
4. 拓扑结构

· 六台Redis服务器



1. IP地址规划
2. IP规划

|  |  |  |
| --- | --- | --- |
| 主机 | 地址 | 端口 |
| redisA | 192.168.1.51 | 6351 |
| redisB | 192.168.1.52 | 6352 |
| redisC | 192.168.1.53 | 6353 |
| redisD | 192.168.1.54 | 6354 |
| redisE | 192.168.1.55 | 6355 |
| redisF | 192.168.1.56 | 6356 |

1. 修改配置文件

/etc/redis/redis.conf

//配置文件路径

bind IP地址 (70行)

//只写物理接口IP地址

daemonize yes

//守护进程方式运行

port xxxx(93行)

//修改对应的端口号

cluster-enabled yes

//开启集群(815行)

cluster-config-file nodes-xx.conf

//集群的配置文件不要使用默认的(823行)

cluster-node-timeout 5000

//请求超时 5秒(829行)

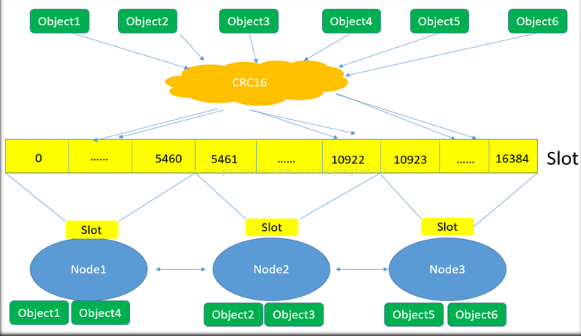
1. 默认情况下，每个库都认为自己是主库

[root@mysql51 ~]# cat /var/lib/redis/6379/nodes-51.conf

47ac560118b9fbec72b07cabaa897a546613f5f7 :0@0 myself,master - 0 0 0 connected

vars currentEpoch 0 lastVoteEpoch 0

1. 集群工作原理



1. 分布式存储
2. key/CRC16 值%16384 所得值在哪个hash值范围内就存在哪一组redis服务器上
3. 创建Redis集群
4. 装包
5. 创建集群
6. 创建集群的工具

· ruby脚本

- 脚本位置：../redis-4.0.8/src/redis-trib.rb

1. 将ruby脚本放入系统PATH

[root@mysql51 src]# cp redis-trib.rb /usr/local/sbin/

//此时没有安装ruby环境，不能直接执行

[root@mysql51 src]# redis-trib.rb

/usr/bin/env: ruby: No such file or directory

[root@mysql51 src]# rpm -q ruby

package ruby is not installed

1. 安装ruby脚本的运行环境包

[root@mysql51 src]# yum install ruby rubygems -y

[root@mysql51 redis]# rpm -ivh --nodeps ruby-devel-2.0.0.648-30.el7.x86\_64.rpm

//忽略依赖安装ruby-devel包

[root@mysql51 redis]# gem install redis-3.2.1.gem

//用gem工具安装

1. 查看ruby工具帮助信息

[root@mysql51 ~]# redis-trib.rb help

//如果运行环境没安装好，此命令会报错

1. 创建集群

· ruby脚本管理集群：创建 查看 删除 添加

· 创建集群时要保证redis库中没有数据

[root@mysql51 ~]# redis-trib.rb create --replicas 1 \

192.168.1.51:6351 \

192.168.1.52:6352 \

192.168.1.53:6353 \

192.168.1.54:6354 \

192.168.1.55:6355 \

192.168.1.56:6356

--replicas 1 表示自动为每一个master节点分配一个slave节点

>>> Creating cluster

>>> Performing hash slots allocation on 6 nodes...

Using 3 masters:

192.168.1.51:6351

192.168.1.52:6352

192.168.1.53:6353

Adding replica 192.168.1.55:6355 to 192.168.1.51:6351

Adding replica 192.168.1.56:6356 to 192.168.1.52:6352

Adding replica 192.168.1.54:6354 to 192.168.1.53:6353

M: 47ac560118b9fbec72b07cabaa897a546613f5f7 192.168.1.51:6351

slots:0-5460 (5461 slots) master

M: e0dca01f74b8b9bed378fb11d709f42f8e05d67c 192.168.1.52:6352

slots:5461-10922 (5462 slots) master

M: 7ab29546773cab637450187d65f7d014aebb8461 192.168.1.53:6353

slots:10923-16383 (5461 slots) master

S: bc5434ff3adadbab9d9dea816f0f420a40355a3d 192.168.1.54:6354

replicates 7ab29546773cab637450187d65f7d014aebb8461

S: 0f0139fb7d9ff20ecf52b3afdc42960a2833ed21 192.168.1.55:6355

replicates 47ac560118b9fbec72b07cabaa897a546613f5f7

S: 8004b8422a20dcfb254a1a8cb3ae202d212e4b2a 192.168.1.56:6356

replicates e0dca01f74b8b9bed378fb11d709f42f8e05d67c

Can I set the above configuration? (type 'yes' to accept): yes

>>> Nodes configuration updated

>>> Assign a different config epoch to each node

>>> Sending CLUSTER MEET messages to join the cluster

Waiting for the cluster to join...

>>> Performing Cluster Check (using node 192.168.1.51:6351)

M: 47ac560118b9fbec72b07cabaa897a546613f5f7 192.168.1.51:6351

slots:0-5460 (5461 slots) master

1 additional replica(s)

S: 0f0139fb7d9ff20ecf52b3afdc42960a2833ed21 192.168.1.55:6355

slots: (0 slots) slave

replicates 47ac560118b9fbec72b07cabaa897a546613f5f7

M: e0dca01f74b8b9bed378fb11d709f42f8e05d67c 192.168.1.52:6352

slots:5461-10922 (5462 slots) master

1 additional replica(s)

S: bc5434ff3adadbab9d9dea816f0f420a40355a3d 192.168.1.54:6354

slots: (0 slots) slave

replicates 7ab29546773cab637450187d65f7d014aebb8461

S: 8004b8422a20dcfb254a1a8cb3ae202d212e4b2a 192.168.1.56:6356

slots: (0 slots) slave

replicates e0dca01f74b8b9bed378fb11d709f42f8e05d67c

M: 7ab29546773cab637450187d65f7d014aebb8461 192.168.1.53:6353

slots:10923-16383 (5461 slots) master

1 additional replica(s)

[OK] All nodes agree about slots configuration.

>>> Check for open slots...

>>> Check slots coverage...

[OK] All 16384 slots covered.

1. 查看集群信息
2. 通过文件查看

[root@mysql51 ~]# cat /var/lib/redis/6379/nodes-51.conf

1. 通过脚本查看

[root@mysql51 ~]# redis-cli -h 192.168.1.51 -p 6351

192.168.1.51:6351> CLUSTER nodes

//查看集群节点信息

192.168.1.51:6351> CLUSTER info

//查看集群信息

1. 测试集群

· 在客户端访问任意一台master主机存取数据

- redis-cli -c -h ip地址 -p 端口

> set key

//存数据

[root@mysql57 ~]# redis-cli -c -h 192.168.1.51 -p 6351

192.168.1.51:6351> set name bob

-> Redirected to slot [5798] located at 192.168.1.52:6352

OK

//可以看到虽然访问的是51，但数据存在52上

192.168.1.52:6352> set age 19

-> Redirected to slot [741] located at 192.168.1.51:6351

OK

//age在51上

//从库默认不允许查询

> get key

//取数据

[root@mysql52 ~]# redis-cli -h 192.168.1.52 -p 6352

192.168.1.52:6352> keys \*

1. "name"

192.168.1.51:6351> get name

-> Redirected to slot [5798] located at 192.168.1.52:6352

"bob"

* 1. 管理集群

1. 管理命令
2. redis-cli命令

· 查看命令帮助

- redis-cli -h

· 常用选项

-h IP地址

-p 端口

-c 集群模式

1. redis-trib.rb脚本

· 语法格式

- redis-trib.rb 选项 参数

· 选项

- add-node 添加master主机

- check 检测集群

- reshard 重新分片

- add-node --slave 添加slave主机

- del-node 删除主机

1. master选举测试
2. 选举master主机

· 检测集群：

[root@mysql51 ~]# redis-trib.rb check 192.168.1.51:6351

注意：当集群中的主库宕机，对应的从库会自动升级为主库，而当一组主从全部宕机时，整个集群会坏掉。因此在构建集群时通常部署1主2从，当主库宕机时需要及时将机器修复。

将mysql52宕机，从库变为主库，重新启动redis

1. 添加新节点
2. 运行redis服务，做启用集群配置后，重启服务

将新主机添加到集群中：redis-trib.rb add-node [--slave] [--master-id] <arg> new\_host:new\_port xisting\_host:existing\_port

1. 添加master主机

· 注意：添加主库需要重做分片

[root@mysql51 ~]# redis-trib.rb add-node 192.168.1.57:6357 192.168.1.

51:6351

[root@mysql51 ~]# redis-trib.rb reshard 192.168.1.51:6351

How many slots do you want to move (from 1 to 16384)? 4096

//从当前的集群中拿出多少个hash slot

What is the receiving node ID? 2c8b2cba92c6169dc82dc39657a770f53ef9f750

Source node #1:all

Do you want to proceed with the proposed reshard plan (yes/no)? yes

...

M: 2c8b2cba92c6169dc82dc39657a770f53ef9f750 192.168.1.57:6357

slots:0-1364,5461-6826,10923-12287 (4096 slots) master

0 additional replica(s)

1. 添加slave主机

· 不指定默认添加到从库最少的主库下，若从库数量都一致则随机添加。可以指定添加从库的主库。

1. 添加从库54，随机成为到57从库。

[root@mysql51 ~]# redis-trib.rb add-node --slave 192.168.1.54:6354 192.168.1.51:6351

...

>>> Configure node as replica of 192.168.1.57:6357.

1. 将54添加为53的从库

[root@mysql51 ~]# redis-trib.rb add-node --slave --master-id 7ab29546773cab637450187d65f7d014aebb8461 192.168.1.54:6354 192.168.1.51:6351

...

>>> Configure node as replica of 192.168.1.53:6353.

1. 将58添加为57从库

[root@mysql51 ~]# redis-trib.rb add-node --slave 192.168.1.58:6358 192.168.1.51:6351

...

>>> Configure node as replica of 192.168.1.57:6357.

1. 移除节点
2. 移除master主机

· 移除多少个hash slots

· 接受hash slots主机的ID

· 移出hash slots的主机的ID

· yes

· 移除集群中的mysql57

[root@mysql51 ~]# redis-trib.rb reshard 192.168.1.51:6351

4096

52ID

57ID

done

yes

[root@mysql51 ~]# redis-trib.rb del-node 192.168.1.51:6351 2c8b2cba92c6169dc82dc39657a770f53ef9f750

1. 移除slave主机

· 从集群中删除54，删除中需要指定具体的ID

[root@mysql51 ~]# redis-trib.rb del-node 192.168.1.51:6351 b6c7e177d1e441a8c8eed1855a64111f464a5687

1. Day 03: redis主从同步 存储持久化 redis数据类型
2. Redis主从复制
3. 主从复制概述
4. 主从复制结构模式

· 结构模式

- 一主一从、一主多从、主从从

1. 主从复制工作原理

· 工作原理

- slave向master发送sync命令

- master启动后台存盘进程，并收集所有修改数据命令

- master完成后台存盘后，传送整个数据文件到slave

- slave接受数据文件，加载到内存中完成首次完全同步

- 后续有新数据产生时，master继续将新的数据收集到的修改命令依次传给slave，完成同步

1. 主从复制缺点

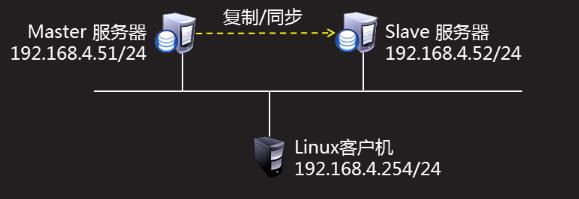
· 缺点：

- 网络繁忙会产生数据同步延时问题

- 系统繁忙会产生数据同步掩饰问题

1. 配置主从复制
2. 拓扑结构

· 以主服务器为基准，将数据自动同步到从服务器



1. 配置从库

· 配置从库192.168.1.55/24

- redis服务运行后，默认都是master服务器

redis-cli -h 192.168.1.55 -p 6355

> info replication

//查看主从配置信息

> SLAVEOF 192.168.1.55 6355

//手动设置从库

OK

> info replication

1. 反客为主

· 主库宕机后，手动将从库设置为主库

redis-cli -h 192.168.1.52

> SLAVEOF no one

//手动设为主库

OK

> info replication

1. 哨兵模式

- 主库宕机后，从库自动升级为主库

- 在slave主机编辑sentinel.conf文件

- 在slave主机运行哨兵程序

[root@mysql55 ~]# vim /etc/sentinel.conf

sentinel monitor redis51 192.168.1.55 6355 1

sentinel monitor 主机名 ip地址 端口 票数

其中主机名为自定义，主库宕机后，票数大于1的主机被升级为主库。

[root@mysql55 ~]# redis-sentinel /etc/sentinel.conf

1. 配置永久主从同步结构且带验证的主从复制
2. 配置主库密码
3. 配置从库验证

282行与289行

[root@mysql55 ~]# vim /etc/redis/6379.conf

282 slaveof 192.168.1.54 6354

289 masterauth 123456

设置密码后，sentinel配置文件中需要配置密码字段

sentinel auth-pass mymaster $password

1. RDB/AOF持久化
2. 持久化之RDB
3. RDB介绍

· Redis数据库文件，全称Redis DataBase，又叫snap快照

- 数据持久化方式之一

- 按照指定时间间隔，将内存中的数据集快照写入硬盘

- 术语叫Snapshot快照

- 恢复时，将快照文件直接读入内存

1. 相关配置参数

· 文件名

- dafilename "dump.rbd"

//文件名

- save ""

//禁用RDB

· 数据从内存保存到硬盘的频率

- save 900 1

//900秒内且有一次修改

- save 300 10

//300秒内且有10次修改

- save 60 10000

//60秒内且有10000次修改

· 手动立刻存盘

- save

//阻塞写存盘

- bgsave

//不阻塞写存盘

· 压缩

- rdbcompression yes|no

· 在存储快照后，使用crc16算法做数据校验

- rdbchecksum yes|no

· bgsave出错时停止写操作

- stop-writes-on-bgsave-error yes|no

1. 使用RDB文件恢复数据

· 备份数据

- 备份dump.rdb文件到其他位置

# cp 数据库目录/dump.rdb

//备份目录

· 恢复数据

- 拷贝备份文件到数据库目录，重启redis服务

1. RDB优点与缺点

· RDB优点

- 高性能的持久化实现——创建一个子进程来执行持久化，先将数据写入临时文件，持久化过程结束后，再用这个临时文件替换上次持久化号的文件;过程中进程不做任何IO操作。

- 比较适合大规模数据恢复，且对数据完整性要求不是特别高的场合

· RDB的缺点

- 意外宕机时，最后一次持久化的数据会丢失

1. 持久化之AOF
2. AOF介绍

· 类似于MySQL服务的binlog日志文件

· 只做追加操作的文件，Append Only File，追加写操作

- 记录redis服务所有写操作

- 不断的将新的写操作追加到文件的末尾

- 使用cat命令可以查看文件的内容

1. 相关配置参数

· 文件名

- appendfilename "appendonly.aof"

//指定文件名

- appendonly yes

//启用AOF，默认no

· AOF文件记录写操作的方式

- appendfsync always

//有新写操作立即记录

- appendfsync everysec

//每秒记录一次

- appendfsync no

//从不记录

· 日志文件会不断增大，何时触发日志重写？

- redis会记录上次重写时AOF文件的大小

- 默认配置当aof文件是上次rewrite后大小的1倍且文件大于64M时触发

> auto-aof-rewrite-percentage 100

> auto-aof-rewrite-min-size 64mb

· 修复AOF文件

- 把文件恢复到最后一次正确的操作

# redis-check-aof --fix appendonly.aof

1. 使用AOF文件恢复数据

· 备份数据

- 备份appendonly.aof文件到其他位置

# cp 数据库目录/appendonly.aof 备份目录

· 恢复数据

- 拷贝备份文件到数据库目录，重启redis服务

# cp 备份目录/appendonly.aof 数据库目录

# /etc/redis/redis\_端口 start

1. AOF优点与缺点

· AOF优点

- 可以灵活设置持久化方式，同步持久化appendfsync always或者异步持久化appendfsync everysec

- 出现意外宕机时，尽可能丢失1s的数据

· AOF缺点

- 持久计划文件的体积通常会大于RDB方式

- 执行fsync策略时的速度可能会比RDB方式慢

1. 数据类型
2. String字符串
3. 字符串操作

· set key value [ex seconds] [px milliseconds] [nx|xx]

- 设置key及值，过期时间可以使用秒或者毫秒为单位

· setrange key offset value

- 从偏移量开始复写key的特定位的值

> set first "hello world"

> setrange first 6 "Redis"

//改写hello Redis

> set tel 187010108888

> get tel

> SETRANGE tel 3 \*\*\*\*\*

> get tel

"187\*\*\*\*\*8888"

>STRLEN tel

· strlen key，统计字串长度

> strlen first

· append key value

- 存在则追加，不存在则创建key及value，返回key长度

> append myname jacob

· setbit key offset value

- 对key所存储字串，设置或清除特定偏移量上的位(bit)

- value值可以为1或0，offset为0~2^32之间

- key不存在，则创建新key

> setbit bit 0 1

//设置bit第0位为1

> setbit bit 1 0

//设置bit第1位为0

· bitcount key

- 统计字串中被设置为1的比特位数量

> setbit bits 0 1

//0001

> setbit bits 3 1

> bit count bits

//结果为2

> setbit peter 100 1

//网站上线100天用户登陆了一次

> setbit peter 105 1

//网站上线105天用户登陆了一次

· decr key

- 将key中的值-1，key不存在则先初始化为0，再减1

> set test 10

> decr test

· decrby key decrement

- 将key中的值减去decrement

· get key

- 返回key存储的字符串值，若key不存在则返回null

- 若key的值不是字符串，则返回错误。get只能处理字串

· 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，-2表示倒数第二个字符

> set first "hello,the world"

> getrange first -5 -1

> getrange first 0 4

· incrkey

- 将key的值加1，如果key不存在，则初始化为0后加1

- 主要应用为计数器

>set page 20

> incr page

· incrby key increment

- 将key的值增加increment

· incrbyfloat key increment

- 为key中所存储的值加上浮点数增量increment

> set num 16.1

> incrbyfloat num 1.1

· mget key [key...]

- 获取一个或多个key的值，空格分割，具有原子性

· mset key value [key value ...]

- 设置多个key及值，空格分割，具有原子性

1. List列表
2. List列表简介

· Redis的list是一个字符队列

· 先进后出

· 一个key可以有多个值

1. List列表操作

· lpush key value [value...]

- 讲一个或多个值value插入到列表key的表头

- Key不存在，则创建key

> lpush list a b c

//list值依次为c、b、a

· lrange key start stop

- 从开始位置读取key的值到stop结束

> lrange liust 0 2

//从0位开始，读到2位为止

> lrange list 0 -1

//从开始读到结束为止

> lrange list 0 -2

//从开始读到倒数第二位为止

· lpop key

- 移除并返回列表头元素数据，key不存在则返回nil

> lpop list

//删除表头元素，可以多次执行

· llen key

- 返回列表key的长度

· lindex key index

- 返回列表中第index个值

> lindex key 0; lindex key 2; lindex key -2

· lset key index value

- 将key中index位置的值修改为value

> lset list 3 test

//将list中第3个值修改为test

· rpush key value [value...]

- 将value插入到key的末尾

> rpush list3 a b c

//list3值为a b c

> rpush list3 d

//末尾插入d

· rpop key

- 删除并返回key末尾的值

> rpush list4 a b c

//list4的值为a b c

> rpop list4

//删除末尾的c，并返回删除的值

1. Hash表
2. Hash表简介

· 让定义的变量可以存储多个key/values

· Redis hash

- 是一个string类型的field和value的映射表

- 一个key可对应多个field，一个field对应一个value

- 将一个对象存储为hash类型，较每个字段都存储成string类型更能节省内存

1. Hash表操作

· hset key field value

- 将hash表中field值设置为value

> hset site google 'www.h.cn'

> hset site baidu 'www.baidu.com'

· hget key field

- 获取hash表中field的值

> hget site google

· hmset key field value [field value...]

- 同时给hash表中的多个field赋值

> hmset site google [www.g.cn](http://www.g.cn) baidu [www.baidu.com](http://www.baidu.com)

· hmget key field [field...]

- 返回hash表中多个field的值

> hmget site google baidu

· hkeys key

- 返回hash表中所有field名称

> hmset site google [www.g.cn](http://www.g.cn) baidu [www.baidu.com](http://www.baidu.com)

> hkeys site

· hgetall key

- 返回hash表中所有field的值

· hvals key

- 返回hash表中所有field的值

> hvals key

· hdel key field [field...]

- 删除hash表中多个field的值，不存在则忽略

> hdel site google baidu

1. Day 04 MongoDB安装、MongoDB命令、MongoDB数据类型、MongoDB导入导出、MongoDB备份还原
2. 部署MongoDB服务
3. MongoDB概述
4. MongoDB介绍

· 介于关系数据库和非关系数据库之间的产品

- 一款基于分布式文件存储的数据库，旨在为WEB应用提供可扩展的高性能数据存储解决方案

- 将数据存储为一个文档(类似于JSON对象)，数据结构由键值对(key => value)组成

- 支持丰富的查询表达，可以设置任何属性的索引

- 支持副本集，分片

1. 相关概念



1. 搭建MDB服务器
2. 安装软件
3. 装包

· 免安装，解压后即可使用

[root@mysql51 ~]# mkdir -p /usr/local/mongodb

[root@mysql51 ~]# mv mongodb-linux-x86\_64-rhel70-3.6.3/bin/ /usr/local/mongodb

[root@mysql51 mongodb]# mkdir -p etc log data/db

1. 创建配置文件

[root@mysql51 ~]# vim /usr/local/mongodb/etc/mongodb.conf

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

# 追加的方式记录日志信息

logappend=true

# 数据库目录

dbpath=/usr/local/mongodb/data/db

# 守护进程方式运行

fork=true

1. 启动服务/查看进程/查看端口

[root@mysql51 ~]# mongod -f /usr/local/mongodb/etc/mongodb.conf

about to fork child process, waiting until server is ready for connections.

forked process: 1225

child process started successfully, parent exiting

[root@mysql51 ~]# ps aux | grep mongod

[root@mysql51 ~]# ss -antulp | grep 27017

tcp LISTEN 0 128 127.0.0.1:27017 \*:\* users:(("mongod",pid=1225,fd=11))

默认端口为27017，地址为回环

· 修改默认地址和端口

[root@mysql51 ~]# vim /usr/local/mongodb/etc/mongodb.conf

...

# 修改服务IP地址和端口号

bind\_ip=192.168.1.51

port=27051

相应的字段可以通过mongod --help查看

[root@mysql51 ~]# mongod -f /usr/local/mongodb/etc/mongodb.conf

1. 连接服务

[root@mysql51 ~]# mongo

//默认连接方式

[root@mysql51 ~]# mongo --host 192.168.1.51 --port 27051

//指定地址和端口的连接方式

1. MongoDB基本使用
2. 常用管理命令
3. 数据库管理

· 查看、创建、切换、删除库

- show dbs

//查看已有的库

- db

//显示当前所在的库

- use 库名

//切换库，若库不存在延时创建库

- show collections或show tables

//查看库下已有集合

- db.dropDatabase()

//删除当前所在的库

· 数据库名称规范

- 不能是空字符串(" ")

- 不得含有空格,.,$,/,\,和\0空字符

- 应全部小写

- 最多64字节

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> use studb

switched to db studb

1. 集合管理

· 查看、创建、删除集合

- show collections或show tables

//查看集合

- db.集合名.drop()

//删除集合

- db.集合名.save({","})

//创建集合，集合不存在时，创建并添加文档

· 集合名命名规范

- 不能是空字符串" "

- 不能含有\0字符(空字符)

- 不能以"system."开头，这是为系统集合保留的前缀

- 用户创建的集合名字不能含有保留字符

> show tables

> db.c1.save({name:"Lee JW",age:22,sex:"girl"})

WriteResult({ "nInserted" : 1 })

> db.c1.save({name:"Lee JW",class:"XD YJY"})

WriteResult({ "nInserted" : 1 })

> show tables

c1

> db.c1.find()

{ "\_id" : ObjectId("5ceacfb010ec5ee786410008"), "name" : "Lee JW", "age" : 22, "sex" : "girl" }

{ "\_id" : ObjectId("5ceacfe210ec5ee786410009"), "name" : "Lee JW", "class" : "XD YJY" }

1. 文档基本管理

· 文档：类似于MySQL表里的记录

· 查看、统计、添加、删除文档

- db.集合名.find()

- db.集合名.count()

- db.集合名.insert({"name":"jim"})

- db.集合名.find({条件})

- db.集合名.findOne()

//返回一条文档

- db.集合名.remove({})

//删除所有文档

- db.集合名.remove({条件})

//删除匹配的所有文档

> db.c1.count()

> db.c1.insert({"name":"jim"})

> db.c1.findOne()

1. 基本数据类型
2. 字符string/布尔bool/空null

· 字符串string

- UTF-8字符串都可以表示为字符串类型的数据

- {name:"张三"}或{school:"tarena"}

· 布尔bool

- 布尔类型有两个值true和false，{x:true}

· 空null

- 用于标识空值或者不存在的字段，{x:null}

1. 数值/数组array

· 数值

- shell默认使用64位浮点型数值。{x:3.14}或{x:3}

- NumberInt(4字节整数){x:NumberInt(3)}

- NumberLong(8字节整数){x:NumberLong(3)}

· 数组array

- 数据列表或数据集可以表示为数组

- {x:["a","b","c"]}

1. 代码/日期/对象

· 代码

- 查询和文档汇总可以包括任何JS代码

- {x:function(){/\* 代码 \*/}}

> db.c1.save({ |name:"php",codeformat:function(){/\* <?php echo "hello world" ?> \*/ } })

· 日期

- 日期被存储为自新纪元以来经过的毫秒数，不含时区

- {x:new Date()}

· 对象

- 对象id是一个12字节的字符串，是文档的唯一标识

- {x:ObjectId()}

1. 内嵌/正则表达式

· 内嵌

- 文档可以嵌套其他文档，被嵌套的文档作为值来处理

- {tarena:

{address:"Beijing",tel:"888888",person:"hansy" }}

> db.c1.save({ bob:{age:19,sex:"boy"} })

· 正则表达式

- 查询时，使用正则表达式作为限定条件

- {x:/正则表达式/}

> db.c1.save({

name:"hanmm",match:/^a/

})

1. 数据导入导出
2. 数据导出

# mongoexport --help

· 语法格式1

# mongoexport [ --host IP地址 --port 端口 ] \

-d 库名 -c 集合名 -f 字段名1,字段名2 \

-- type=csv > 目录名/文件名.csv

· 语法格式2

# mongoexprt [ --host IP地址 --port 端口 ] \

-d 库名 -c 集合名 -f 字段名1,字段名2 \

--type-csv > 目录名/文件名.csv

注意：导出为csv格式必须使用-f指定字段名列表

· 语法格式3

# mongoexport [ --host IP地址 --port 端口 ] \

-d 库名 -c 集合名 [ -q '{条件}' -f 字段列表 ] \

--type=json > 目录名/文件名.json

[root@mysql51 ~]# mkdir /mongodbdir

[root@mysql51 ~]# mongoexport --host 192.168.1.51 --port 27051 -d studb -c c1 --type=json > /mongodbdir/ci.json

2019-05-27T22:52:38.099+0800 connected to: 192.168.1.51:27051

2019-05-27T22:52:38.101+0800 exported 2 records

1. 数据导入

# mongoimport --help

· 语法格式1

# mongoimport --host IP地址 --port 端口 \

-d 库名 -c 集合名 \

--type=json 目录名/文件名.json

· 语法格式2

# mongoimport --host IP地址 --port 端口 \

-d 库名 -c 集合名 \

-f 字段名1,字段名2 --type=csv [--headerline] [--drop] 目录名/文件名.csv

1. 导入数据时，若库和集合不存在，则先创建库和集合后再导入数据;
2. 若库和集合已存在，则以追加的方式导入数据到集合里
3. 使用--drop选项可以删除原数据后导入新数据，--headerline，忽略标题

[root@mysql52 ~]# mongoimport --host 192.168.1.52 --port 27052 -d testdb -c t1 -f \_id,name --type=csv /root/c1.csv

2019-05-28T00:20:02.381+0800 connected to: 192.168.1.52:27052

2019-05-28T00:20:02.420+0800 imported 3 documents

-f与--headerline两个选项只能用一个

1. 数据备份与恢复
2. 数据备份

· 备份所有库到当前目录下的dump目录下

# mongodump [ --host ip地址 --port 端口 ]

· 备份时指定备份的库和备份目录

# mongodump [ --host ip地址 --port 端口 ] -d 数据库名 -c 集合名 -o 目录

· 查看bson文件内容

# bsondump ./dump/bbs/t1.bson

1. 数据恢复

· 语法格式

# mongorestore --host IP地址 --port 端口 -d 数据库名 [ -c 集合名 ] 备份目录名

1. Day 05 MongoDB副本集、MongoDB文档管理
2. MongoDB副本集
3. 环境准备
4. 副本集介绍

· 也称为MongoDB复制

- 指在多个服务器上存储数据副本，并实现数据同步

- 提高数据可用性、安全性，方便数据故障恢复

1. MongoDB复制原理

· 副本集工作过程

- 至少需要两个节点。其中一个是主节点，负责处理客户端请求，其余是从节点，负责复制主节点数据

- 常见搭配方式：一主一从、一主多从

- 主节点记录所有操作oplog，从节点定期轮询主节点获取这些操作，然后对自己的数据副本执行这些操作，从而保证从节点的数据与主节点一致

1. 副本集实现方式

· Master-Slave主从复制

- 启动一台服务器时加上"-master"参数，作为主节点

- 启动其他服务器时加上"-salve"和"-source"参数，作为从节点

· 主从复制的优点

- 从节点可以提供数据查询，降低主节点的访问压力

- 由从节点执行备份，避免锁定主节点数据

- 当主节点故障时，可快速切换到从节点，实现高可用

· Replica Sets副本集

- 从1.6版本开始支持，优于之前的replication

- 支持孤战挂自动切换、自动修复成员节点，降低运维成本

- Replica Sets副本集的结构类似高可用集群

1. 配置Replica Sets
2. 运行MongoDB服务

· 启动服务时，指定主机所在副本集名称

- 所有副本集成员使用相同的副本集名称

- --replSet rs1

//指定副本集名称

[root@mysql53 ~]# vim /usr/local/mongodb/etc/mongodb.conf

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

logappend=true

dbpath=/usr/local/mongodb/data/db

fork=true

bind\_ip=192.168.1.53

port=27053

replSet=rs1

[root@mysql53 ~]# mongod -f /usr/local/mongodb/etc/mongodb.conf

1. 配置节点信息

[root@mysql51 ~]# mongo --host 192.168.1.51 --port 27051

· 创建集群

config = {

\_id:"rs1",

members:[

{\_id:0,host:"192.168.1.51:27051"},

{\_id:1,host:"192.168.1.52:27052"},

{\_id:2,host:"192.168.1.53:27053"}

]

};

1. 初始化Replica Sets环境

> rs.initiate(config)

//此处可Tab

1. 查看副本集信息

· 查看状态信息

rs1:OTHER> rs.status()

· 查看是否是master库

rs1:PRIMARY> rs.isMaster()

1. 验证副本集配置

· 同步数据验证，允许从库查看数据

rs1:PRIMARY> db.getMongo().setSlaveOk()

从库默认不允许任何操作

· 自动切换主库验证

rs1:PRIMARY> rs.isMaster()

· 定义优先级

config = {

\_id:"rs1",

members:[

{\_id:0,host:"192.168.1.51:27051",priority:9},

{\_id:1,host:"192.168.1.52:27052",priority:10},

{\_id:2,host:"192.168.1.53:27053",priority:9}

]

};

1. 文档管理
2. 插入文档
3. save()

· 格式

> db.集合名save({ key:"值" , key:"值" })

· 注意

- 集合不存在时创建集合，然后再插入记录

- \_id字段值已存在时，修改文档字段值

- \_id字段值不存在时，插入文档

1. insert()

· 格式

> db.集合民=名.insert({key:"值" , key:"值" })

· 注意

- 集合不存在时创建集合

- \_id字段值已存在时，放弃插入

- \_od字段值不存在时，插入文档

· 插入多条记录

> db.集合名.insertMany(

[

{name:"xiaoju" ,age:19},

{name:"laoshi" ,email:"yaya@tedu.cn"}

]

)

1. 查询文档
2. 查询语法

· 显示所有行( 默认输出20行，驶入it可显示后续行 )

> db.集合名.find()

· 显示第1行

> db.集合名.findOne()

· 指定查询条件并指定显示的字段

> db.集合名.find( {条件},{定义显示的字段} )

> db.user.find( {},{\_id:0,name:1,shell:1} )

//0不显示，1显示

1. 行数显示限制

· limit(数字)

//显示前几行

> db.集合名.find().limit(3)

· skip(数字)

//跳过前几行

> db.集合名.find().skip(2)

· sort(字段名)

//1升序，-1降序

> db.集合名.find().sort(age:1|-1)

> db.user.find({shell:"/sbin/nologin"},{\_id:0,name:1,uid:1,shell:1}).skip(2).limit(2)

1. 匹配条件

· 简单条件

> db.集合名.find({key:"值"})

> db.集合名.find({shell:"/bin/bash" ,name:"root"})

> db.user.find({shell:"/bin/bash"})

> db.user.find({shell:"/bin/bash",name:"root"})

· 范围比较

- $in 在…里

- $nin 不在…里

- $or 或

> db.user.find({uid:{$in:[1,6,9]}})

> db.user.find({uid:{$nin:[1,6,9]}})

> db.user.find({$or:[{name:"root"},{uid:1} ]})

· 正则匹配

> db.user.find({name:/^a/})

· 数值比较

- $It(<) $lte(<=) $gt(>) $gte(>=) $ne(!=)

> db.user.find({ uid:{$gte:10,$lte:40}},{\_id:0,name:1,uid:1})

> db.user.find({uid:{$lte:5,}})

· 匹配null，也可以匹配没有的字段

> db.user.save({name:null,uid:null})

> db.user.find({name:null})

> db.user.find({"\_id":

ObjectId("..."),"name":null,"uid":null})

1. 更新文档
2. update()

· 语法格式

> db.集合名.update({条件},{修改的字段})

1. 多文档更新

· 语法格式：默认只更新与条件匹配的第1行

> db.user.update({条件},{$set:{修改的字段}},false,true)

> db.user.update({name:"bin"},{$set:{password:"abc12123"}},false,true)

//修改全部匹配的值

1. $set/$unset

· $set 条件匹配时，修改指定字段的值

> db.user.update( {条件},$set:{修改的字段} )

> db.user3.update( {name:"bin"},{$unset:{password:"A"}} )

· $unset 删除与条件匹配文档的字段

> db.集合名.update( {条件},{$unset:{key:values}} )

> db.user3.update( {name:"bin"},{$unset:{password:"A"}} )

1. $inc

· $inc 条件匹配时，字段值自加或自减

> db.集合名.update( {条件},{$inc:{字段名:数字}} )

> db.user.update( {name:"bin"},{$inc:{uid:2}} )

//字段值自加2

> db.user.update( {name:"bin"},{$inc:{uid:-1}} )

//字段值自减1

1. $push/$addToSet

· $push 像数组中添加新元素

> db.集合名.update( {条件}，{$push:{数组名:"值"}} )

> db.user.insert( {name:"bob",likes:["a","b","c","d","e","f"]} )

> db.user.update( {name:"bob"},{$push:{likes:"w"}} )

· $addToSet 避免重复添加

> db.集合名.update( {条件},{$addToSet:{数组名:"值"}} )

> db.user.update( {name:"bob"},{$addToSet:{likes:"f"}} )

1. $pop/$pull

· $pop 从数组头部删除一个元素

> db.集合名.update( {条件},{$pop:{数组名:数字}} )

> db.user.update( {name:"bob"},{$pop:{likes:1}} )

> db.user.update( {name:"bob"},{$pop:{likes:-1}} )

· $pull 删除数组指定元素

> db.集合名.update( {条件},{$pull:{数组名:值}} )

> db.user.update( {name:"bob"},{$pull:{lpikes:"b"}} )

1. 删除文档
2. drop()/remove()

· $drop() 删除集合的同时删除索引

> db.集合名.drop()

> db.user.drop()

· $remove() 删除文档时不删除索引

> db.集合名.remove({})

//删除所有文档

> db.集合名.remove( {条件} )

//删除与条件匹配的文档

> db.user.remove( {uid:{$lte:10}} )

> db.user.remove({})

1. SECURITY
2. Day 01 Linux基本防护 用户切换与提权 SSH访问控制 SELinux安全 、 SSH访问控制 SELinux安全

一.Linux基本防护

· 用户账号安全

1.设置账号有效日期

· 使用chage工具

-d 0 强制修改密码

-l 查看账户信息

-E yyyy-mm-dd 指定失效日期(-1取消) 失效的用户将无法登录

· 定义默认有效期（强制定期修改密码）

# cat /etc/login.defs

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 //UID最小值

UID\_MAX 60000 //UID最大值

2.账号的锁定/解锁

· 使用passwd命令

-l 锁定

-u 解锁

-S 查看状态

3.修改tty登录的提示信息，隐藏系统版本(伪装登录提示)

· vim /etc/issue 本地登录

· vim /etc/issue.net 远程登录

4.文件系统安全

· 程序和服务控制

1) 禁用非必要的系统服务

- 使用systemctl,chkconfig工具

2) 关闭执行权限

# mount -o remount,rw,noexec /dev/vda1 /boot/ //临时修改,永久修改需写入fstab中

# mount | grep boot

3) 关闭更新atime

# # mount -o remount,rw,atime /dev/vda1 /boot/ //临时修改,永久修改需写入fstab中

5.锁定/解锁保护文件

· EXT3/EXT4的文件属性控制

- chattr,lsattr

· +,-,=控制方式

-i 不可变

-a 仅可追加

· 用户切换与提权

1.su切换用户身份

· su - [目标用户]

· su [-] -c "命令" [目标用户]

2.sudu提升执行权限

· 管理员预先设置执行许可

· 被授权用户有执行授权的命令,验证自己的口令

· 命令格式

# sudo 特权命令

# sudo [-u 目标用户] 特权命令

· 授权

# vim /etc/sudoers

或

# visudo

· 允许用户useradm通过sudo方式添加/删除/修改除root以外的用户账号

zhangsan ALL=(ALL) /usr/bin/systemctl

useradm ALL=(root) /usr/bin/passwd,!/usr/bin/passwd root,/usr/sbin/user\*,!/usr/sbin/user\* root

· 允许wheel组成员以特权执行所有命令

%wheel ALL=(ALL) ALL

# usermod -a -G wheel zhangsan

· 为sudo机制启用日志记录，以便跟踪sudo执行操作

# visudo

Defaults logfile="/var/log/sudo"

· 提高SSH服务安全

1.配置基本安全策略

# vim /etc/ssh/sshd\_config

.. ..

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

MaxAuthTries 3 //每连接最多认证次数

.. ..

# systemctl restart sshd

2.针对SSH访问采用仅允许的策略，未明确列出的用户一概拒绝登录

# vim /etc/ssh/sshd\_config

.. ..

AllowUsers zhangsan tom useradm@192.168.4.0/24 //定义账户白名单

##DenyUsers USER1 USER2 //定义账户黑名单

##DenyGroups GROUP1 GROUP2 //定义组黑名单

##AllowGroups GROUP1 GROUP2 //定义组白名单

# systemctl restart sshd

· 验证SSH访问控制，未授权的用户将拒绝登录

· SELinux安全防护

1.SELinux概述

· 什么是SELinux

- 一套强化Linux安全的扩展模块

- 美国国家安全局主导开发

· SELinux的运作机制

- 集成到Linux内核(2.6及以上)

- 操作系统可定制的策略,管理工具

· SELinux策略集

SELINUXTYPE=targeted

- 推荐,仅保护最常见/关键的网络服务,其他不限制

- 主要软件包:

selinux-policy

selinux-policy-targeted

libselinux-utils

coreutils

policycoreutuilks

2.SELinux模式控制

1）固定配置：修改/etc/selinux/config文件

· 修改kernel启动参数

- 添加selinux=0以禁用

- 添加enforcing=0 设置SELinux为允许模式

· 修改文件/etc/selinux/config

- 设置SELINUX=disabled以禁用

- 设置SELINUX=permissive宽松/允许模式

- 设置SELINUX=enforcing强制模式

==============================================================================

Day 02 加密与解密 AIDE入侵检测系统 扫描与抓包

一.加密与解密

· 加/解密概述

1.信息传递中的风险

· 电脑黑客

· 出差人员

· 离职人员

· 合作伙伴

· 商业间谍

· 高管习惯

· 开发人员

· 流程失控

2.什么是加密/解密

· 发送方 : 明文 -> 密文

· 接收方 : 密文 -> 明文

3.加密目的及方式

· 确保数据的机密性

- 对称加密:加密/解密用同一个密钥

- 非对称加密:加密/解密用不同密钥

· 保护信息的完整性

- 信息摘要:基于输入的信息生成长度较短,数位固定的散列值

4.常见的加密算法

· 对称加密

- DES

- AES

· 非对称加密

- RSA

- DSA

· Hash散列技术,用于信息摘要

- MD5

- SHA(256/512)

5.MD5完整性教研

· 使用md5sum校验工具

- 生成MD5校验值

- 与软件官方提供的校验值比对

vimdiff a b 比对a,b两个文件

· GPG加/解密工具

1.使用GPG对称加密方式保护文件

1) 确保已经安装了相关软件（默认已经安装好了）

# yum -y install gnupg2 //安装软件

# gpg --version //查看版本

gpg (GnuPG) 2.0.22

2) gpg使用对称加密算法加密数据的操作

# echo 233 > 1.txt //创建1.txt

# gpg -c 1.txt //对称加密

# cat 1.txt.gpg //显示为乱码

�ʐik�����­����n-r6\_x0010\_X�E@��2��MJ9C�L����

3) 使用gpg对加密文件进行解密操作

# gpg -d 1.txt //对称解密

# gpg -d 1.txt.gpg | cat

gpg: 钥匙环‘/root/.gnupg/secring.gpg’已建立

gpg: CAST5 加密过的数据

gpg: 以 1 个密码加密

gpg: 警告：报文未受到完整的保护

233

2.使用GPG非对称加密方式保护文件

非对称加密/解密文件时，UserA（192.168.4.100）生成私钥与公钥，并把公钥发送给UserB（192.168.4.5），UserB使用公钥加密数据，并把加密后的数据传给UserA，UserA最后使用自己的私钥解密数据

1) 接收方UserA创建自己的公钥,私钥对(192.168.4.100)

# gpg --gen-key //创建密钥对

请选择您要使用的密钥种类：

(1) RSA and RSA (default)

(2) DSA and Elgamal

(3) DSA (仅用于签名)

(4) RSA (仅用于签名)

您的选择？

RSA 密钥长度应在 1024 位与 4096 位之间。

您想要用多大的密钥尺寸？(2048)

您所要求的密钥尺寸是 2048 位

请设定这把密钥的有效期限。

0 = 密钥永不过期

<n> = 密钥在 n 天后过期

<n>w = 密钥在 n 周后过期

<n>m = 密钥在 n 月后过期

<n>y = 密钥在 n 年后过期

密钥的有效期限是？(0)

密钥永远不会过期

以上正确吗？(y/n)y

You need a user ID to identify your key; the software constructs the user ID

from the Real Name, Comment and Email Address in this form:

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

真实姓名：tommy

电子邮件地址：tom@163.com

注释：abc

您选定了这个用户标识：

“tommy (abc) <tom@163.com>”

更改姓名(N)、注释(C)、电子邮件地址(E)或确定(O)/退出(Q)？o

您需要一个密码来保护您的私钥。

您不想要有密码――这个想法实在是遭透了！

不过，我仍然会照您想的去做。您任何时候都可以变更您的密码，仅需要

再次执行这个程序，并且使用“--edit-key”选项即可。

我们需要生成大量的随机字节。这个时候您可以多做些琐事(像是敲打键盘、移动

鼠标、读写硬盘之类的)，这会让随机数字发生器有更好的机会获得足够的熵数。

我们需要生成大量的随机字节。这个时候您可以多做些琐事(像是敲打键盘、移动

鼠标、读写硬盘之类的)，这会让随机数字发生器有更好的机会获得足够的熵数。

gpg: 密钥 C600BC96 被标记为绝对信任

公钥和私钥已经生成并经签名。

gpg: 正在检查信任度数据库

gpg: 需要 3 份勉强信任和 1 份完全信任，PGP 信任模型

gpg: 深度：0 有效性： 1 已签名： 0 信任度：0-，0q，0n，0m，0f，1u

pub 2048R/C600BC96 2018-08-30

密钥指纹 = 9E41 785B BBF8 3558 E71C D197 DA72 E794 C600 BC96

uid tommy (abc) <tom@163.com>

sub 2048R/3EF61A55 2018-08-30

2) UserA导出自己的公钥文件

· 用户的公钥,私钥信息分别保存在pubring.gpg和secring.gpg文件内

· 使用gpg命令结合--export选项将其中的公钥文本导出

# gpg -a --export UserA > UserA.pub //--export的作用是导出密钥，-a的作用是导出的密钥存储为ASCII格式

# scp UserA.pub 192.168.4.5:/tmp/ //将密钥传给Proxy

3) UserB使用公钥加密数据，并把加密后的数据传给UserA（在192.168.4.5操作）

# echo "I love you ." > love.txt

# gpg -e -r UserA love.txt

无论如何还是使用这把密钥吗？(y/N)y //确认使用此密钥加密文件

//-e选项是使用密钥加密数据

//-r选项后面跟的是密钥，说明使用哪个密钥对文件加密

# scp love.txt.gpg 192.168.4.100:/root //加密的数据传给UserA

4) UserA以自己的私钥解密文件（在192.168.4.100操作）

# gpg -d love.txt.gpg > love.txt

您需要输入密码，才能解开这个用户的私钥：“UserA (UserA) <UserA@tarena.com>”

2048 位的 RSA 密钥，钥匙号 9FA3AD25，建立于 2017-08-16 (主钥匙号 421C9354)

//验证私钥口令

gpg: 由 2048 位的 RSA 密钥加密，钥匙号为 9FA3AD25、生成于 2017-08-16

“UserA (UserA) <UserA@tarena.com>”

# cat love.txt //获得解密后的文件内容

I love you.

3.使用GPG的签名机制，检查数据来源的正确性

1) 在client(192.168.4.100)上，UserA为软件包创建分离式签名

# tar zcf log.tar /var/log //建立测试软件包

# gpg -b log.tar //创建分离式数字签名

# ls -lh log.tar\*

-rw-rw-r--. 1 root root 170 8月 17 21:18 log.tar

-rw-rw-r--. 1 root root 287 8月 17 21:22 log.tar.sig

# scp log.tar\* 192.168.4.5:/root //将签名文件与签名传给UserB

2) 在192.168.4.5上验证签名

# gpg --verify log.tar.sig log.tar

gpg:于2028年06月07日 星期六 23时23分23秒 CST 创建的签名，使用 RSA，钥匙号 421C9354

gpg: 完好的签名，来自于“UserA (UserA) <UserA@tarena.com>”

.. ..

二.使用AIDE做入侵检测

· 初始化系统

1.安装软件包

# yum -y install aide

2.修改配置文件

# vim /etc/aide.conf

@@define DBDIR /var/lib/aide //数据库目录

@@define LOGDIR /var/log/aide //日志目录

database\_out=file:@@{DBDIR}/aide.db.new.gz //数据库文件名

//一下内容为可以检查的项目（权限，用户，组，大小，哈希值等）

#p: permissions

#i: inode:

#n: number of links

#u: user

#g: group

#s: size

#md5: md5 checksum

#sha1: sha1 checksum

#sha256: sha256 checksum

DATAONLY = p+n+u+g+s+acl+selinux+xattrs+sha256

//以下内容设置需要对哪些数据进行入侵校验检查

//注意：为了校验的效率，这里将所有默认的校验目录与文件都注释

//仅保留/root目录，其他目录都注释掉

/root DATAONLY

#/boot NORMAL //对哪些目录进行什么校验

#/bin NORMAL

#/sbin NORMAL

#/lib NORMAL

#/lib64 NORMAL

#/opt NORMAL

#/usr NORMAL

#!/usr/src //使用[!]，设置不校验的目录

#!/usr/tmp

· 初始化数据库，入侵后检测

1.入侵前对数据进行校验，生成初始化数据库

[root@proxy ~]# aide --init

AIDE, version 0.15.1

AIDE database at /var/lib/aide/aide.db.new.gz initialized.

//生成校验数据库，数据保存在/var/lib/aide/aide.db.new.gz

# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz

//校验时是和不带new的文件进行比对

2.备份数据库，将数据库文件拷贝到U盘（非必须的操作）

[root@proxy ~]# cp /var/lib/aide/aide.db.new.gz /media/

3.入侵后检测

[root@proxy ~]# cd /var/lib/aide/

[root@proxy ~]# mv aide.db.new.gz aide.db.gz

[root@proxy ~]# aide --check //检查哪些数据发生了变化

三.扫描与抓包分析

· 使用NMAP扫描来获取指定主机/网段的相关信息

1.安装软件

# yum -y install nmap

//基本用法：

# nmap [扫描类型] [选项] <扫描目标 ...>

//常用的扫描类型

// -sS，TCP SYN扫描（半开）

// -sT，TCP 连接扫描（全开）

// -sU，UDP扫描

// -sP，ICMP扫描

// -A，目标系统全面分析

2.检查192.168.4.100主机是否可以ping通

# nmap -sP 192.168.4.100

Starting Nmap 6.40 ( http://nmap.org ) at 2018-06-06 21:59 CST

mass\_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers

Nmap scan report for host3 (192.168.4.100)

Host is up (0.00036s latency).

MAC Address: 52:54:00:71:07:76 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds

· 使用-n选项可以不执行DNS解析

· 可以批量检查网段(192.168.2.0/24)

3.检查目标主机所开启的TCP服务

[root@proxy ~]# nmap -sT 192.168.4.100

.. ..

PORT STATE SERVICE

21/tcp open ftp

22/tcp open ssh

25/tcp open smtp

80/tcp open http

110/tcp open pop3

111/tcp open rpcbind

143/tcp open imap

443/tcp open https

993/tcp open imaps

995/tcp open pop3s

MAC Address: 00:0C:29:74:BE:21 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.2 seconds

· 检查192.168.4.0/24网段内哪些主机开启了FTP、SSH服务

[root@proxy ~]# nmap -p 21-22 192.168.4.0/24

[root@proxy ~]# nmap -n -sT -p 22-80 192.168.2.0/24

· nmap -sT 全开扫描

· nmap -sS 半开扫描

4.检查目标主机所开启的UDP服务

[root@proxy ~]# nmap -sU 192.168.4.100 //指定-sU扫描UDP

53/udp open domain

111/udp open rpcbind

5.全面分析目标主机192.168.4.100和192.168.4.5的操作系统信息

[root@proxy ~]# nmap -A 192.168.4.100,5

· 中间人攻击

IP地址欺骗

MAC地址欺骗

IP不同(域名类似)

www.icbc.coe[955xx发短信]

dhcp(ip,netmask,网关)

\* 中间人攻击!=抓包

· 使用tcpdump分析FTP访问中的明文交换信息

1.准备vsftpd服务器

# yum -y install vsftpd

# systemctl restart vsftpd

2.启用tcpdump命令行抓包

# tcpdump

//监控选项如下：

// -i，指定监控的网络接口（默认监听第一个网卡）

// -A，转换为 ACSII 码，以方便阅读

// -w，将数据包信息保存到指定文件

// -r，从指定文件读取数据包信息

//tcpdump的过滤条件：

// 类型：host、net、port、portrange

// 方向：src、dst

// 协议：tcp、udp、ip、wlan、arp、……

// 多个条件组合：and、or、not

========================================================

Day 03 系统审计 服务安全 Linux安全之打补丁

一.系统审计

· 什么是审计

- 基于事先配置的规则生成日志,记录可能发生在系统上的事件

- 审计不会为系统提供额外的安全保护,但他会发现并记录违反安全策略的人及其对应的行为

- 审计能够记录的日志内容:

a.日期与事件,事件结果

b.触发事件的用户

c.所有认证机制的使用都可以被记录,如ssh等

d.对关键数据文件的修改行为等

· 审计的案例

- 监控文件访问

- 监控系统调用

- 记录用户运行的命令

- 审计可以监控网络访问行为

- ausearch工具,可以根据条件过滤审计日志

- aureport工具,可以生成审计报告

1.部署audit监控文件

1) 安装软件包，查看配置文件（确定审计日志的位置）

# yum -y install audit //安装软件包

# cat /etc/audit/auditd.conf //查看配置文件，确定日志位置

log\_file = /var/log/audit/audit.log //日志文件路径

# systemctl start auditd //启动服务

# systemctl enable auditd //设置开机自启

2) 配置审计规则

· 可以使用auditctl命令控制审计系统并设置规则

# auditctl -s //查询状态

# auditctl -l //查看规则

# auditctl -D //删除所有规则

· 定义临时文件系统规则：

#语法格式：auditctl -w path -p permission -k key\_name

# path为需要审计的文件或目录

# 权限可以是r,w,x,a(文件或目录的属性发生变化)

# Key\_name为可选项，方便识别哪些规则生成特定的日志项

· example:

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

//设置规则所有对passwd文件的写、属性修改操作都会被记录审计日志

# auditctl -w /etc/selinux/ -p wa -k selinux\_change

//设置规则，监控/etc/selinux目录

# auditctl -w /usr/sbin/fdisk -p x -k disk\_partition

//设置规则，监控fdisk程序

# auditclt -w /etc/ssh/sshd\_conf -p warx -k sshd\_config

//设置规则，监控sshd\_conf文件

· 如果需要创建永久审计规则，则需要修改规则配置文件：

# vim /etc/audit/rules.d/audit.rules

-w /etc/passwd -p wa -k passwd\_changes

-w /usr/sbin/fdisk -p x -k partition\_disks

2.查看并分析日志

1）手动查看日志

查看SSH的主配置文件/etc/ssh/sshd\_conf，查看audit日志信息：

[root@proxy ~]# tailf /var/log/audit/audit.log

type=SYSCALL msg=audit(1517557590.644:229228): arch=c000003e

syscall=2 success=yes exit=3

a0=7fff71721839 a1=0 a2=1fffffffffff0000 a3=7fff717204c0

items=1 ppid=7654 pid=7808 auid=0 uid=0 gid=0 euid=0 suid=0

fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=3 comm="cat"

exe="/usr/bin/cat"

subj=unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023 key="sshd\_config"

.. ..

#内容分析

# type为类型

# msg为(time\_stamp:ID)，时间是date +%s（1970-1-1至今的秒数）

# arch=c000003e，代表x86\_64（16进制）

# success=yes/no，事件是否成功

# a0-a3是程序调用时前4个参数，16进制编码了

# ppid父进程ID，如bash，pid进程ID，如cat命令

# auid是审核用户的id，su - test, 依然可以追踪su前的账户

# uid，gid用户与组

# tty:从哪个终端执行的命令

# comm="cat" 用户在命令行执行的指令

# exe="/bin/cat" 实际程序的路径

# key="sshd\_config" 管理员定义的策略关键字key

# type=CWD 用来记录当前工作目录

# cwd="/home/username"

# type=PATH

# ouid(owner's user id） 对象所有者id

# guid(owner's groupid） 对象所有者id

2）通过工具搜索日志

系统提供的ausearch命令可以方便的搜索特定日志，默认该程序会搜索/var/log/audit/audit.log，ausearch options -if file\_name可以指定文件名。

[root@proxy ~]# ausearch -k sshd\_config -i

//根据key搜索日志，-i选项表示以交互式方式操作

二.加固常见服务的安全

1.优化Nginx服务的安全配置

1） 删除不需要的模块

Nignx是模块化设计的软件，需要什么功能与模块以及不需要哪些模块，都可以在编译安装软件时自定义，使用--with参数可以开启某些模块，使用--without可以禁用某些模块。最小化安装永远都是对的方案！

下面是禁用某些模块的案例：

[root@proxy ~]# tar -xf nginx-1.12.tar.gz

[root@proxy ~]# cd nginx-1.12

[root@proxy nginx-1.12]# ./configure \

>--without-http\_autoindex\_module \ //禁用自动索引文件目录模块

>--without-http\_ssi\_module

[root@proxy nginx-1.12]# make

[root@proxy nginx-1.12]# make install

2）修改版本信息，并隐藏具体的版本号

默认Nginx会显示版本信息以及具体的版本号，这些信息给攻击者带来了便利性，便于他们找到具体版本的漏洞。

如果需要屏蔽版本号信息，执行如下操作，可以隐藏版本号。

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

… …

http{

server\_tokens off; //在http下面手动添加这么一行

… …

}

[root@proxy ~]# nginx -s reload

[root@proxy ~]# curl -I http://192.168.4.5 //查看服务器响应的头部信息

· 但服务器还是显示了使用的软件为nginx，通过如下方法可以修改该信息。

[root@proxy nginx-1.12]# vim +48 src/http/ngx\_http\_header\_filter\_module.c

//注意：vim这条命令必须在nginx-1.12源码包目录下执行！！！！！！

//该文件修改前效果如下：

static u\_char ngx\_http\_server\_string[] = "Server: nginx" CRLF;

static u\_char ngx\_http\_server\_full\_string[] = "Server: " NGINX\_VER CRLF;

static u\_char ngx\_http\_server\_build\_string[] = "Server: " NGINX\_VER\_BUILD CRLF;

//下面是我们修改后的效果：

static u\_char ngx\_http\_server\_string[] = "Server: Jacob" CRLF;

static u\_char ngx\_http\_server\_full\_string[] = "Server: Jacob" CRLF;

static u\_char ngx\_http\_server\_build\_string[] = "Server: Jacob" CRLF;

//修改完成后，再去编译安装Nignx，版本信息将不再显示为Nginx，而是Jacob

[root@proxy nginx-1.12]# ./configure

[root@proxy nginx-1.12]# make && make install

[root@proxy nginx-1.12]# killall nginx

[root@proxy nginx-1.12]# /usr/local/nginx/sbin/nginx //启动服务

[root@proxy nginx-1.12]# curl -I http://192.168.4.5 //查看版本信息验证

3） 限制并发量

DDOS攻击者会发送大量的并发连接，占用服务器资源（包括连接数、带宽等），这样会导致正常用户处于等待或无法访问服务器的状态。

Nginx提供了一个ngx\_http\_limit\_req\_module模块，可以有效降低DDOS攻击的风险，操作方法如下：

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

… …

http{

… …

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

server {

listen 80;

server\_name localhost;

limit\_req zone=one burst=5;

}

}

//备注说明：

//limit\_req\_zone语法格式如下：

//limit\_req\_zone key zone=name:size rate=rate;

//上面案例中是将客户端IP信息存储名称为one的共享内存，内存空间为10M

//1M可以存储8千个IP信息，10M可以存储8万个主机连接的状态，容量可以根据需要任意调整

//每秒中仅接受1个请求，多余的放入漏斗

//漏斗超过5个则报错

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

· 客户端使用ab测试软件测试效果：

[root@client ~]# ab -c 100 -n 100 http://192.168.4.5/

4） 拒绝非法的请求

通过如下设置可以让Nginx拒绝非法的请求方法：

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

server {

listen 80;

#这里，!符号表示对正则取反，~符号是正则匹配符号

#如果用户使用非GET或POST方法访问网站，则retrun返回400的错误信息

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

return 400;

}

}

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

修改服务器配置后，客户端使用不同请求方法测试：

[root@client ~]# curl -i -X GET http://192.168.4.5 //正常

[root@client ~]# curl -i -X HEAD http://192.168.4.5 //报错

5） 防止buffer溢出

当客户端连接服务器时，服务器会启用各种缓存，用来存放连接的状态信息。

如果攻击者发送大量的连接请求，而服务器不对缓存做限制的话，内存数据就有可能溢出（空间不足）。

修改Nginx配置文件，调整各种buffer参数，可以有效降低溢出风险。

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

client\_body\_buffer\_size 1K;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 1k;

large\_client\_header\_buffers 2 1k;

… …

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

2.数据库安全

1） 初始化安全脚本

安装完MariaDB或MySQL后，默认root没有密码，并且提供了一个任何人都可以操作的test测试数据库。有一个名称为mysql\_secure\_installation的脚本，该脚本可以帮助我们为root设置密码，并禁止root从远程其他主机登陆数据库，并删除测试性数据库test。

[root@proxy ~]# systemctl status mariadb

//确保服务已启动

[root@proxy ~]# mysql\_secure\_installation

//执行初始化安全脚本

2）密码安全

手动修改MariaDB或MySQL数据库密码的方法：

[root@proxy ~]# mysqladmin -uroot -predhat password 'mysql'

//修改密码，旧密码为redhat，新密码为mysql

[root@proxy ~]# mysql -uroot -pmysql

MariaDB [(none)]>set password for root@'localhost'=password('redhat');

//使用账户登录数据库，修改密码

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

· 修改密码成功，而且密码在数据库中是加密的，有什么问题吗？问题是你的密码被明文记录了，下面来看看明文密码：

[root@proxy ~]# cat .bash\_history

mysqladmin -uroot -pxxx password 'redhat'

//通过命令行修改的密码，bash会自动记录历史，历史记录中记录了明文密码

[root@proxy ~]# cat .mysql\_history

set password for root@'localhost'=password('redhat');

select user,host,password from mysql.user;

flush privileges;

//通过mysql命令修改的密码，mysql也会有所有操作指令的记录，这里也记录了明文密码

管理好自己的历史，不使用明文登录，选择合适的版本5.6以后的版本，日志，行为审计（找到行为人），使用防火墙从TCP层设置ACL（禁止外网接触数据库）。

3）数据备份与还原

首先，备份数据库（注意用户名为root，密码为redhat）：

[root@proxy ~]# mysqldump -uroot -predhat mydb table > table.sql

//备份数据库中的某个数据表

[root@proxy ~]# mysqldump -uroot -predhat mydb > mydb.sql

//备份某个数据库

[root@proxy ~]# mysqldump -uroot -predhat --all-databases > all.sql

//备份所有数据库

接下来，还原数据库（注意用户名为root，密码为redhat）：

[root@proxy ~]# mysql -uroot -predhat mydb < table.sql //还原数据表

[root@proxy ~]# mysql -uroot -predhat mydb < mydb.sql //还原数据库

[root@proxy ~]# mysql -uroot -predhat < all.sql //还原所有数据库

4）数据安全

在服务器上（192.168.4.5），创建一个数据库账户：

[root@proxy ~]# mysql -uroot -predhat

//使用管理员，登陆数据库

MariaDB [(none)]> grant all on \*.\* to tom@'%' identified by '123';

//创建一个新账户tom

使用tcpdump抓包（192.168.4.5）

[root@proxy ~]# tcpdump -w log -i any src or dst port 3306

//抓取源或目标端口是3306的数据包，保存到log文件中

客户端（192.168.4.100）从远程登陆数据库服务器（192.168.4.5）

[root@client ~]# mysql -utom -p123 -h 192.168.4.5

//在192.168.4.100这台主机使用mysql命令登陆远程数据库服务器（192.168.4.5）

//用户名为tom，密码为123

MariaDB [(none)]> select \* from mysql.user;

//登陆数据库后，任意执行一条查询语句

回到服务器查看抓取的数据包

[root@proxy ~]# tcpdump -A -r log

//使用tcpdump查看之前抓取的数据包，很多数据库的数据都明文显示出来

· 可以使用SSH远程连接服务器后，再从本地登陆数据库;

· 或者也可以使用SSL对MySQL服务器进行加密，类似与HTTP+SSL一样，MySQL也支持SSL加密

3.Tomcat安全性

1） 隐藏版本信息、修改tomcat主配置文件（隐藏版本信息）

未修改版本信息前，使用命令查看服务器的版本信息

注意：proxy有192.168.2.5的IP地址，这里使用proxy作为客户端访问192.168.2.100服务器。

[root@proxy ~]# curl -I http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看头部信息

[root@proxy ~]# curl -I http://192.168.2.100:8080

//访问存在的页面文件，查看头部信息

[root@proxy ~]# curl http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看错误信息

· 修改tomcat配置文件，修改版本信息(在192.168.2.100操作)：

[root@web1 tomcat]# yum -y install java-1.8.0-openjdk-devel

[root@web1 tomcat]# cd /usr/local/tomcat/lib/

[root@web1 lib]# jar -xf catalina.jar

[root@web1 lib]# vim org/apache/catalina/util/ServerInfo.properties

//根据自己的需要，修改版本信息的内容

[root@web1 lib]# /usr/local/tomcat/bin/shutdown.sh //关闭服务

[root@web1 lib]# /usr/local/tomcat/bin/startup.sh //启动服务

· 修改后，客户端再次查看版本信息（在192.168.2.5操作）：

[root@proxy ~]# curl -I http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看头部信息

[root@proxy ~]# curl -I http://192.168.2.100:8080

//访问存在的页面文件，查看头部信息

[root@proxy ~]# curl http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看错误信息

· 再次修改tomcat服务器配置文件，修改版本信息，手动添加server参数（在192.168.2.100操作）：

[root@web1 lib]# vim /usr/local/tomcat/conf/server.xml

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

connectionTimeout="20000" redirectPort="8443" server="jacob" />

[root@web1 lib]# /usr/local/tomcat/bin/shutdown.sh //关闭服务

[root@web1 lib]# /usr/local/tomcat/bin/startup.sh //启动服务

· 修改后,客户端再次查看版本信息（在192.168.2.5操作）：

[root@proxy ~]# curl -I http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看头部信息

[root@proxy ~]# curl -I http://192.168.2.100:8080

//访问存在的页面文件，查看头部信息

[root@proxy ~]# curl http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看错误信息

2）降级启动

默认tomcat使用系统高级管理员账户root启动服务，启动服务尽量使用普通用户。

[root@web1 ~]# useradd tomcat

[root@web1 ~]# chown -R tomcat:tomcat /usr/local/tomcat/

//修改tomcat目录的权限，让tomcat账户对该目录有操作权限

[root@web1 ~]# su -c /usr/local/tomcat/bin/startup.sh tomcat

//使用su命令切换为tomcat账户，以tomcat账户的身份启动tomcat服务

[root@web1 ~]# chmod +x /etc/rc.local //该文件为开机启动文件

[root@web1 ~]# vim /etc/rc.local //修改文件，添加如下内容

su -c /usr/local/tomcat/bin/startup.sh tomcat

3）删除默认的测试页面

[root@web1 ~]# rm -rf /usr/local/tomcat/webapps/\*

三.使用diff和patch工具打补丁

1.比单个文件差异

1） 编写两个版本的脚本，一个为v1版本，一个为v2版本。

[root@proxy ~]# cat test1.sh //v1版本脚本

#!/bin/bash

echo "hello wrld"

[root@proxy ~]# cat test2.sh //v2版本脚本

#!/bin/bash

echo "hello the world"

echo "test file"

2） 使用diff命令语法

使用diff命令查看不同版本文件的差异。

[root@proxy ~]# diff test1.sh test2.sh //查看文件差异

@@ -1,3 +1,3 @@

#!/bin/bash

-echo "hello world"

-echo "test"

+echo "hello the world"

+echo "test file"

[root@proxy ~]# diff -u test1.sh test2.sh //查看差异，包含头部信息

--- test1.sh 2018-02-07 22:20:02.723971251 +0800

+++ test2.sh 2018-02-07 22:20:13.358760687 +0800

@@ -1,3 +1,3 @@

#!/bin/bash

-echo "hello world"

-echo "test"

+echo "hello the world"

+echo "test file"

2.使用patch命令对单文件代码打补丁

1）准备实验环境

[root@proxy ~]# cd demo

[root@proxy demo]# vim test1.sh

#!/bin/bash

echo "hello world"

echo "test"

[root@proxy demo]# vim test2.sh

#!/bin/bash

echo "hello the world"

echo "test file"

2) 生成补丁文件

[root@proxy demo]# diff -u test1.sh test2.sh > test.patch

3）使用patch命令打补丁

在代码相同目录下为代码打补丁

[root@proxy demo]# yum -y install patch

[root@proxy demo]# patch -p0 < test.patch //打补丁

patching file test1.sh

//patch -pnum（其中num为数字，指定删除补丁文件中多少层路径前缀）

//如原始路径为/u/howard/src/blurfl/blurfl.c

//-p0则整个路径不变

//-p1则修改路径为u/howard/src/blurfl/blurfl.c

//-p4则修改路径为blurfl/blurfl.c

//-R(reverse)反向修复，-E修复后如果文件为空，则删除该文件

[root@proxy demo]# patch -RE < test.patch //还原旧版本，反向修复

==================================================================

1. Day 04 iptables防火墙 filter表控制 扩展匹配 nat表典型应用
   1. iptables防火墙
      1. 防火墙概述
2. 什么是防火墙

· 一道保护性的安全屏障

- 主要作用: 保护、隔离

1. Linux包过滤防火墙

· Linux包过滤防火墙

- RHEL7默认使用firewalld作为防火墙

- 但firewalld底层还是调用包过滤防火墙iptables

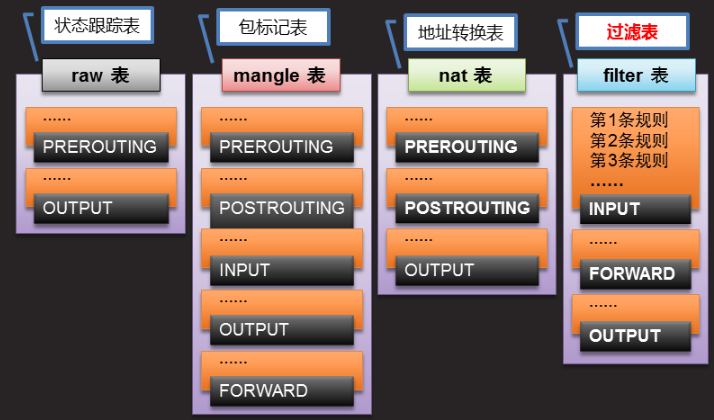
# systemctl stop firewalld

# systemctl disable firewalld

# yum -y install iptables-services

# systemctl start iptables

1. iptables的表、链结构



· iptables的四个表(区分大小写)

- iptables默认有4个表

- nat表(地址转换表)

- filter表(数据过滤表)

- raw表(状态跟踪表)

- mangle表(包标记表)

· iptables的五个链(区分大小写)

- INPUT链(入站规则)

- OUTPUT(出站规则)

- FORWARD(转发规则)

- PREROUTING(路由前规则)

- POSTROUTING(路由后规则)

1. 包过滤匹配流程

· 规则链内的匹配顺序

- 顺序对比, 匹配即停止(LOG除外)

- 若无任何匹配, 则按照该链的默认策略处理

* + 1. iptables用法解析

1. iptables基本用法

· 管理程序位置

1. 基本的目标操作
2. 常用的管理选项
   * 1. 规则管理示例
3. 添加新的规则
4. 查看规则列表
5. 删除、清空规则
6. 设置默认规则
   * 1. iptables命令的基本使用方法

# iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

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

# iptables -t filter -I INPUT -p icmp -j ACCEPT

# iptables -I INPUT -p icmp -j REJECT

1. 注意事项与规律：

- 可以不指定表，默认为filter表

- 可以不指定链，默认为对应表的所有链

- 如果没有找到匹配条件，则执行防火墙默认规则

- 选项/链名/目标操作用大写字母，其余都小写

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

1. 目标操作：

- ACCEPT：允许通过/放行

- DROP：直接丢弃，不给出任何回应

- REJECT：拒绝通过，必要时会给出提示

- LOG：记录日志，然后传给下一条规则

1. iptables常用选项

· 添加规则:

-A 追加一条防火墙规则至链末尾位置

-I 插入一条防火墙规则至链的开头

· 查看规则:

-L 查看iptables 所有规则

-n 以数字形式显示地址,端口等信息

--line-numbers 查看规则时,显示规则的行号

· 删除规则:

-D 删除链内指定序号(或内容)的一条规则

-F 清空所有的规则

-P 为指定的链设置默认规则

# iptables -t filter -A INPUT -p tcp -j ACCEPT

//追加规则至filter表中的INPUT链的末尾，允许任何人使用TCP协议访问本机

# iptables -I INPUT -p udp -j ACCEPT

//插入规则至filter表中的INPUT链的开头，允许任何人使用UDP协议访问本机

# iptables -I INPUT 2 -p icmp -j ACCEPT

//插入规则至filter表中的INPUT链的第2行，允许任何人使用ICMP协议访问本机

· 查看iptables防火墙规则

# iptables -nL INPUT //仅查看INPUT链的规则

· 删除规则，清空所有规则

# iptables -D INPUT 3

//删除filter表中INPUT链的第3条规则

# iptables -nL INPOUT

# iptables -F

//清空filter表中所有链的防火墙规则

· 设置防火墙默认规则

# iptables -t filter -P INPUT DROP

# iptables -nL

Chain INPUT (policy DROP)

1. iptables防火墙规则的条件

通用匹配

协议匹配 -p 协议名称

地址匹配 -s 源地址,-d 目标地址

接口匹配 -i 接受数据的网卡,-o 发送数据的网卡

隐含匹配:

端口匹配 --sport 源端口号,--dport 目标端口号

ICMP 类型匹配 --icmp-type ICMP 类型

# iptables -I INPUT -p tcp --dport 80 -j REJECT

# iptables -I INPUT -s 192.168.2.100 -j REJECT

# iptables -I INPUT -d 192.168.2.5 -p tcp --dport 80 -j REJECT

# iptables -I INPUT -i eth0 -p tcp --dport 80 -j REJECT

# iptables -A INPUT -s 192.168.4.100 -j DROP

//丢弃192.168.4.100发给本机的所有数据包

2.禁ping的相关策略

1) 默认直接禁ping的问题

# iptables -I INPUT -p icmp -j DROP

//设置完上面的规则后，其他主机确实无法ping本机，但本机也无法ping其他主机

//当本机ping其他主机，其他主机回应也是使用icmp，对方的回应被丢弃

2) 禁止其他主机ping本机，允许本机ping其他主机

# iptables -A INPUT -p icmp \

> --icmp-type echo-request -j DROP

//仅禁止入站的ping请求，不拒绝入站的ping回应包

注意：关于ICMP的类型，可以参考help帮助，参考命令如下：

# iptables -p icmp --help

二.防火墙扩展规则

1.根据MAC地址过滤

# ip link show eth0 //查看client的MAC地址

eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP mode DEFAULT qlen 1000

link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff

[root@proxy ~]# iptables -A INPUT -p tcp --dport 22\

> -m mac --mac-source 52:54:00:00:00:0b -j DROP

//拒绝52:54:00:00:00:0b这台主机远程本机

2.基于多端口设置过滤规则

# iptables -A INPUT -p tcp \

> -m multiport --dports 20:22,25,80,110,143,16501:16800 -j ACCEPT

//一次性开启20,21,22,25,80,110,143,16501到16800所有的端口

3.根据IP地址范围设置规则

# iptables -A INPUT -p tcp --dport 22 \

> -m iprange --src-range 192.168.4.10-192.168.4.20 -j ACCEPT

三.配置SNAT实现共享上网

1.确保proxy主机开启了路由转发功能

# echo 1 > /proc/sys/net/ipv4/ip\_forward //开启路由转发

2.设置防火墙规则，实现SNAT地址转换

# iptables -t nat -A POSTROUTING \

> -s 192.168.4.0/24 -p tcp --dport 80 -j SNAT --to-source 192.168.2.5

3.登陆web主机查看日志

# tail /var/log/httpd/access\_log

.. ..

192.168.2.5 - - [12/Aug/2018:17:57:10 +0800] "GET / HTTP/1.1" 200 27 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"

4.扩展知识，对于proxy外网IP不固定的情况可以执行下面的地址伪装，动态伪装IP。

# iptables -t nat -A POSTROUTING \

> -s 192.168.4.0/24 -p tcp --dport 80 -j MASQUERADE

最后，所有iptables规则都是临时规则，如果需要永久保留规则需要执行如下命令:

# service iptables save

===========================================================

Day 05 监控概述 、 Zabbix基础 、 Zabbix监控服务

一.监控概述

· 监控的目的

1.报告系统运行状况

- 每一部分必须同时监控

- 内容暴扣吞吐量,反应时间,使用率等

2.提前发现问题

- 进行服务器性能调整前,知道调整什么

- 找出系统的瓶颈在什么地方

· 监控的资源类别

1.公开数据

- Web,ftp,ssh,数据库等应用服务

- TCP或UDP端口

2.私有数据

- CPU,内存,磁盘,网卡流量等使用信息

- 用户,进程等运行信息

· 被监控主机需安装agent(每一台)

· agent a

· 监控软件

1.系统监控命令

· ps · ifconfig

· uptime · netstat或ss

· free · ping

· swapon -s · traceroute

· df -h · iostat

2.自动化监控系统

· Cacti

- 基于SNMP协议的监控软件,拥有强大的绘图能力

· Nagios

- 基于Agent监控,强大的状态检查与报警机制

- 插件极多,自己写监控脚本潜入到Nagios非常方便

· Zabbix

- 基于多种监控机制,支持分布式监控(树型结构)

3.部署监控服务器ZAbbix Server

1) 源码安装Zabbix Server

· 先安装相关依赖包

# yum -y install net-snmp-devel \

>curl-devel

# yum -y install \

> libevent-devel-2.0.21-4.el7.x86\_64.rpm //在lnmp-soft中

# tar -xf zabbix-3.4.4.tar.gz

# cd zabbix-3.4.4/

# ./configure --enable-server \

> --enable-proxy --enable-agent --with-mysql=/usr/bin/mysql\_config \

> --with-net-snmp --with-libcurl

// --enable-server安装部署zabbix服务器端软件

// --enable-agent安装部署zabbix被监控端软件

// --enable-proxy安装部署zabbix代理相关软件

// --with-mysql配置mysql\_config路径

// --with-net-snmp允许zabbix通过snmp协议监控其他设备

// --with-libcurl安装相关curl库文件，这样zabbix就可以通过curl连接http等服务，测试被监控主机服务的状态

# make && make install

4.初始化Zabbix,配置监控服务器

1) 创建数据库，上线Zabbix的Web页面

# mysql

mysql> create database zabbix character set utf8;

//创建数据库，支持中文字符集

mysql> grant all on zabbix.\* to zabbix@'localhost' identified by 'zabbix';

//创建可以访问数据库的账户与密码

# cd lnmp\_soft/zabbix-3.4.4/database/mysql/

# mysql -uzabbix -pzabbix zabbix < schema.sql

# mysql -uzabbix -pzabbix zabbix < images.sql

# mysql -uzabbix -pzabbix zabbix < data.sql

//刚刚创建是空数据库，zabbix源码包目录下，有提前准备好的数据

//使用mysql导入这些数据即可（注意导入顺序）

2) 上线Zabbix的Web页面

# cd lnmp\_soft/zabbix-3.4.4/frontends/php/

# cp -r \* /usr/local/nginx/html/

# chmod -R 777 /usr/local/nginx/html/\*

3) 修改Zabbix\_server配置文件，设置数据库相关参数，启动Zabbix\_server服务

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

DBHost=localhost

//数据库主机，默认该行被注释

DBName=zabbix

//设置数据库名称

DBUser=zabbix

//设置数据库账户

DBPassword=zabbix

//设置数据库密码，默认该行被注释

LogFile=/tmp/zabbix\_server.log

//设置日志，仅查看以下即可

[root@zabbixserver ~]# useradd -s /sbin/nologin zabbix

//不创建用户无法启动服务

# zabbix\_server //启动服务

4) 确认连接状态，端口10051

# ss -ntulp | grep zabbix\_server

· 如果是因为配置文件不对，导致服务无法启动时，不要重复执行zabbix\_server,一定要先使用killall zabbix\_server关闭服务后，再重新启动一次

5) 浏览器访问Zabbix\_server服务器的Web页面

# firefox http://192.168.2.5/index.php

//第一次访问，初始化PHP页面会检查计算机环境是否满足要求，如果不满足会给出修改建议

//默认会提示PHP的配置不满足环境要求，需要修改PHP配置文件

根据错误提示，修改PHP配置文件，满足Zabbix\_server的Web环境要求,php-bcmath和php-mbstring都在lnmp\_soft目录下有提供软件包

# yum -y install php-gd php-xml

# yum install php-bcmath-5.4.16-42.el7.x86\_64.rpm

# yum install php-mbstring-5.4.16-42.el7.x86\_64.rpm

# vim /etc/php.ini

date.timezone = Asia/Shanghai //设置时区

max\_execution\_time = 300 //最大执行时间，秒

post\_max\_size = 32M //POST数据最大容量

max\_input\_time = 300 //服务器接收数据的时间限制

memory\_limit = 128M //内存容量限制

# systemctl restart php-fpm

修改完PHP配置文件后，再次使用浏览器访问服务器，则会开始初始化

数据库端口:3306

数据库密码:zabbix

· 配置完成后,使用用户admin和密码zabbix登录,设置语言环境为中文

5.配置监控客户端

· 修改Zabbix\_agent配置文件，启动Zabbix\_agent服务,使服务器监控自身

# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5 //允许哪些主机监控本机

ServerActive=127.0.0.1,192.168.2.5 //允许哪些主机通过主动模式监控本机

Hostname=zabbixserver //设置本机主机名

LogFile=/tmp/zabbix\_agentd.log //设置日志文件

UnsafeUserParameters=1 //是否允许自定义key

# zabbix\_agentd //启动监控agent

· 配置web1,web2被监控服务器

1) 源码编译安装zabbbix,修改配置

# cd /root/myTools/lnmp-soft/

# tar -xf zabbix[Tab]

# cd zabbix[Tab]

# ./configure --enable-agent

# make && make install

# vim /usr/local/etc/zabbix\_agentd.conf

Logfile=/tmp/zabbix\_agentd.log

Hostname=zabbixclient\_webx[1-2] //被监控端自己的主机名

Server=127.0.0.1,192.168.2.5 //谁可以监控本机（被动监控模式）

ActiveServer=127.0.0.1,192.168.2.5 //谁可以监控本机（主动监控模式）

EnableRemoteCommands=1 //监控异常后，是否允许服务器远程过来执行命令，如重启某个服务

UnsafeUserParameters=1 //是否允许自定义key监控

# zabbix\_agentd //启动agent服务

2) 拷贝启动脚本（非必须操作，可选做），有启动脚本可以方便管理服务，启动与关闭服务。启动脚本位于zabbix源码目录下。

# cd misc/init.d/fedora/core

# cp zabbix\_agentd /etc/init.d/

# /etc/init.d/zabbix\_agentd start

# /etc/init.d/zabbix\_agentd stop

# /etc/init.d/zabbix\_agentd status

# /etc/init.d/zabbix\_agentd restart

6.自定义Zabbix监控项目

1) 步骤:

· 创建自定义key

· 创建监控项目

· 创建监控图形

· 将监控模板关联到主机

2) 监控主机创建自定义key（在192.168.2.100操作）

# vim /usr/local/etc/zabbix\_agentd.conf

Include=/usr/local/etc/zabbix\_agentd.conf.d/ //加载配置文件目录

# cd /usr/local/etc/zabbix\_agentd.conf.d/

# vim count.line.passwd

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

//自定义key语法格式:

//UserParameter=自定义key名称,命令

3) 测试自定义key是否正常工作

# killall zabbix\_agentd

# zabbix\_agentd //重启agent服务

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

21

如zabbix\_get命令执行错误，提示Check access restrictions in Zabbix agent configuration，则需要检查agent配置文件是否正确：

# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5

ServerActive=127.0.0.1,192.168.2.5

7.创建监控模板

1) 添加监控模板

2) 创建应用

3) 创建监控项目item（监控项）

4) 创建图形

5) 查看监控数据图形

1. CLOUD
2. Day 01 KVM简介、Virsh管理 、自定义虚拟机、虚拟设备管理
3. 阶段简介

· 云计算

· 大型架构

· 重启重装换机器

· 锻炼自己排错的技能

· 学会问问题，看文档

· IT是一个积累的过程，笔记、文档与技术博客

· windows连接Linux常用软件：Xshell、secureCRT、putty

· 企业装包——优先yum，其次编译

1. 配置yum源
2. 安装vsftpd服务

· 物理机：禁用selinux和firewalld

# vim /etc/selinux/config

SELINUX=disabled

# systemctl stop firewalld

# systemctl mask firewalld

# reboot

· 相关知识点：

- 网络yum源

1. 安装FTP

# yum install -y vsftpd

# vim /etc/vsftpd/vsftpd.conf

listen=YES

listen\_ipv6=NO

# systemctl start vsftpd

# cd ~ftp //切换到ftp用户的家目录下

# pwd

/var/ftp

# yum install lftp

# lftp 192.168.1.10

lftp 192.168.1.10:~> ls

lftp 192.168.1.10:~> get //下载ftp中的内容

lftp 192.168.1.10:/> exit

# lftp sftp://192.168.1.254 -u root

Password:(ssh密码)

- 支持Tab

· lftp字段

put 上传

get 下载

exit 退出

1. mount iso

# mkdir -p /var/ftp/CentOS7

# mount -t iso9660 -o ro,loop /var/iso/CentOS7-1708.iso /var/ftp/CentOS7

1. 创建自定义yum源

# mkdir -p /var/ftp/public

# cp xx.rpm /var/ftp/public

# cd /var/ftp/public

# createrepo . 创建新的

# createrepo --update . 更新

· 客户端配置

/etc/yum.repos.d/xxoo.repo

[local\_centos]

name=CentOS 7

baseurl=ftp://ip.xx.xx.xx/CentOS7

enabled=1

gpgcheck=0

[local\_soft]

name=local soft

baseurl=ftp://ip.xx.xx.xx/public

enabled=1

gpgcheck=0

转发dns安装配置

时间服务器 server、client

shell命令 exec 重定向

1. 自定义yum源
2. dns转发+时间同步

物理机配置:

1. 转发dns安装配置

# yum install -y bind-chroot

# vim /etc/named.conf

listen-on port 53 { 192.168.1.10; };

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

...

allow-query { any; };

...

forwarders { 202.106.196.115; };

dnssec-enable no;

dnssec-validation no;

# systemctl strat named

# nslookup

> server 192.168.1.10

defalut server: 192.168.1.10

address: 192.168.1.10#53

> set q=a

> www.baidu.com

# ss -ltun

1. 时间同步服务器,server,client

# yum install chrony -y

server:配置文件

# vim /etc/chrony.conf

server s2b.time.edu.cn iburst //设定上一级时间服务器

bindacqaddress 0.0.0.0

allow 0/0

cmdallow 127.0.0.1

# systemctl restart chronyd

# chronyc sources -v

\* 表正常，? 表失败，+ 表备胎

client:配置文件

server server.ip.xx.xx iburst

检查状态

# chronyc sources -v

\*表正常，?表失败，+表备胎

1. shell命令 exec重定向
2. KVM简介
   * 1. 搭建KVM服务器

1) 虚拟化概念

· 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)

|  |  |
| --- | --- |
| 系列 | PC/服务器代表 |
| VMWare | WMWare Workstation,vSphere |
| Microsoft | VirtualPC,Hyper-V |
| RedHat | KVM,RHEV |
| Citrix | Xen |
| Oracle | Oracle VM VirtualBox |

· VMWare的主要组成部分:

- ESXi -- 红帽系统二次开发

- VCenter -- 管理虚拟化的工具,由VMWare自己开发,需要购买,无操作手册

1. 安装虚拟化服务器平台

· KVM/QEMU/libvirtd

- KVM是linux内核的模块,它需要CPU的支持,采用硬件辅助虚拟化技术Intek-VT,AMD-V,内存的相关如;Intel的EPT和AMD的RVI技术

- QEMU是一个虚拟化的方针工具,通过ioctl与内核kvm交互完成对硬件的虚拟化支持

- libvirtd是一个对虚拟化管理的借接口工具,提供客户端程序virsh,virt-install,virt-manager,virt-view与用户交互

· 必备软件

(1) qemu-kvm

- 为kvm提供底层仿真支持

(2) libvirt-daemon

- libvirtd守护进程,管理虚拟机

(3) libvirt-client

- 客户端软件,提供客户端管理命令

(4) libvirt-daemon-driver-qemu

- libvirtd连接qemu的驱动

· 可选功能

- virt-install

//系统安装工具

- virt-managaer

//图形工具

- virt-v2v

//虚拟机迁移工具

- virt-p2v

//物理机迁移技术

· 虚拟化平台的安装

# yum -y install qemu-kvm \

libvirt-daemon \

libvirt-client \

libvirt-daemon-driver-qemu

# systemctl start libvirtd

1. KVM虚拟机的组成

· 虚拟机的组成

- 内核虚拟化模块(KVM)

- 系统设备仿真(QEMU)

- 虚拟机管理程序(libvirt)

- 一个xml文件(虚拟机配置声明文件)

- 位置 /etc/libvirt/qemu/

- 一个磁盘镜像文件(虚拟机的硬盘)

- 位置 /var/lib/libvirt/images/

* + 1. 管理KVM平台

1) virsh命令工具介绍

· 提供管理各虚拟机的命令接口

- 支持交互模式，查看/创建/停止/关闭

- 格式：virsh 控制指令 [虚拟机名称] [参数]

# virsh

virsh#

· 查看KVM节点信息

- virsh nodeinfo

· 列出虚拟机

# virsh list

//查看正在运行的虚拟机

# virsh list -all

//查看所有的虚拟机

· 列出虚拟网络

- virsh net-list [--all]

· 查看指定虚拟机信息

- virsh dominfo 虚拟机名称

· 运行|重启|关闭指定的虚拟机

- virsh start|reboot|shutdown 虚拟机名称

shutdown关闭 start开启 reboot重启

· 强制关闭指定的虚拟机

- virsh destroy 虚拟机名称

· 虚拟机设置为开机自启

- virsh autostart [--disable] 虚拟机名称

· virsh虚拟机管理

- edit

//修改虚拟机的配置

- autostart

//设置虚拟机自启动

- domiflist

//查看虚拟机网卡信息

- domblklist

//查看虚拟机硬盘信息

- blockresize --path [选对路径] --size 50G open.stack

· LANG变量

LANG=zh\_CN.UTF-8

LANG=en\_US.UTF-8

· 实验1:

1. 创建一台虚拟机

base-vm vm1

1. 启动虚拟机

virsh start vm1

1. 查看虚拟机磁盘

进入虚拟机:

# lsblk

# df -h

在物理机上:

$ virsh domblklist 虚拟机名称

$ qemu-img info 块设备

1. 扩容磁盘

virsh blockresize --path 块设备 --size 50G 虚拟机名称

[student@room9pc01 scripts]$ virsh blockresize --path /var/lib/libvirt/images/vm1.img --size 50G

重新定义 '/var/lib/libvirt/images/vm1.img' 块设备大小

1. 分区扩容

进入虚拟机

LANG=C

growpart /dev/vda 1

[student@room9pc01 scripts]$ sshv vm1

[root@vm1 ~]# growpart /dev/vda 1

CHANGED: partition=1 start=2048 old: size=62912479 end=62914527 new: size=104855519,end=104857567

1. 扩容文件系统

xfs\_growfs /

[root@vm1 ~]# xfs\_growfs /

1. 常用镜像盘类型

· qcow2格式简介

· 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 |

3) qcow2-img命令

· 是虚拟机磁盘管理命令,支持非常多的磁盘格式,如raw,qcow2,vdi,vmdk等等

· 格式

- qemu-img命令 参数 块文件名称 大小

- 常用的有

create 创建一个磁盘

convert 转换磁盘格式

info 查看磁盘信息

snapshot 管理磁盘快照

· 创建新的镜像盘文件

- qemu-img create -f 格式 映像文件名称 大小

# cd /var/lib/libvirt/images

# qemu-img create -f qcow2 node01.img 20G

# qemu-img create -f qcow2 -b rhel7.qcow2 node01.img 20G

# watch -n 1 'll -h'

# dd if=/dev/zere of=./f1 bs=100M count=2

· 查询镜像盘文件的信息

- qemu-img info 磁盘路径

- qemu-img info disk.img

· -b使用后端模板文件

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

4) COW技术原理

· Copy On Write，写时复制

- 直接映射后端盘的数据内容

- 当后端盘的旧数据有修改时，在修改之前自动将旧数据存入前端盘

- 对前端盘的修改不回写到后端盘

* 1. virsh管理
     1. virsh虚拟机管理

1. 连接本地/远程KVM

· 使用virsh客户端工具

- 连接本地

- virsh

- virsh# connect qemu:///system

- 连接远程

- virsh# connect qemu+ssh://user@ip.xx.xx.xx:port/system

# virsh

# virsh -c qemu:///system

# virsh -c qemu+ssh://root@172.40.55.106/system

password:

1. 虚拟机远程管理

· 使用virt-manager客户端工具

- virt-manager也可以通过add connection管理其他机器上的虚拟机，一般通过"remoute wunnel over ssh"就可以了

- 需要注意的是virt-manager需要使用ssh免密码登录，如果没有免密码登录需要单独安装python的ssh相关模块，这里我们可以使用部署key的方法解决

# ssh-keygen -t rsa -b 2048 -N '' -f id\_rsa

# ssh-copy-id -i id\_rsa.pub ip.xx.xx.xx

1. 创建虚拟交换机

· libvirtd网络接口

- 原理：调用dnsmasq提供DNS、DHCP等功能

- 创建配置文件/etc/libvirt/qemu/networks/vbr.xml

<network>

<name>vbr</name>

<bridge name="vbr" />

<forward mode="nat" />

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

<dhcp>

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

</dhcp>

</ip>

</network>

1. 网络管理

· virsh管理虚拟网络

- net-list 查看虚拟网络

- net-define vbr.xml 创建虚拟网络

- net-undefine vbr 删除虚拟网络

- net-start vbr 启动虚拟网络

- net-destroy vbr 停止虚拟网络

- net-edit vbr 修改vbr网络的配置

- net-autostart vbr 设置vbr虚拟网络开机自启动

* + 1. xml管理

1. 导出虚拟机

· xml配置文件

- 定义了一个虚拟机的名称，CPU、内存、虚拟磁盘、网卡等各种参数设置

- 默认位于/etc/libvirt/qemu/虚拟机名.xml

· 导出xml配置文件

- 查看：virsh dumpxml 虚拟机名

- 备份：virsh dumpxml 虚拟机名 > 虚拟机名.xml

1. 编辑虚拟机设置

· 对虚拟机的配置进行调整

- 编辑：virsh edit 虚拟机名

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

1. 导入虚拟机

· 根据修改后的独立xml文件定义新虚拟机

- virsh define XML描述文件

1. 删除虚拟机

· 必要时可以删除多余的xml配置

- 比如虚拟机改名的情况

- 避免出现多个虚拟机的磁盘或MAC地址冲突

- virsh undefine 虚拟机名

* 1. 自定义虚拟机
     1. 自定义虚拟机安装

1. 网络yum源的安装和配置

· 配置网络yum源的思路

1. 配置ftp
2. 修改配置文件 /etc/vsftpd/vsftpd.conf
3. listen=YES
4. listen\_ipv6=NO
5. systemctl enable vsftpd
6. vsftpd默认根目录为 /var/ftp

· 快速配置网络yum源

- 在ftp根目录出那个键文件夹Centos7

- 把CentOS7的光盘挂载到刚刚创建的目录上

- 在客户机里面配置 /etc/yum.repos.d/xxx.repo

[Centos\_repo]

name = CentOS packet

baseurl=ftp://xx.xx.xx.xx/centos7

enabled=1

gpgcheck=0

- yum repolist

1. virt-manager安装虚拟机
   * 1. 虚拟机模板制作
2. 软件包安装及yum源配置

· 命令行安装，网络安装

# virt-install --virt-type=kvm --name=test --vcpus=2 --memory=2048 --location=/ISO/CentOS-7-x86\_64-Everything-1708.iso --disk path=test.qcow2,format=qcow2 --network bridge=private1 --graphics none --extra-args='console=ttyS0' --force

· 常规分盘，只分根

· 把刚刚安装好的系统初始化

1. 禁用selinux /etc/selinux/config

SELINUX=disabled

1. 卸载防火墙与NetworkManager

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

1. 配置yum源

[local\_repo]

name=CentOS-$release-Base

baseurl="ftp://192.168.1.254/ CentOS7/dvd"

enabled=1

gpgcheck=0

·

# yum install -y lftp

1. 安装软件
2. yum源导入公钥验证配置

- gpgcheck=1

1. 导入gpg key

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

1. 常用系统命令安装

# yum install -y net-tools vim-enhanced bridge-utils psmisc

1. 网卡及配置文件设置

· 删除网络配置里的个性化信息

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

DEVICE="eth0"

ONBOOT="yes"

IPV6INIT="no"

TYPE="Ethernet"

BOOTPROTO="dhcp"

- 禁用空路由

- ip route show

169.xxxx

- /etc/sysconfig/network

NOZEROCONF="yes"

- systemctl restart network

1. Console及磁盘分区配置

- /etc/default/grub

# vim /etc/default/grub

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

GRUB\_DISABLE\_LINUX\_UUID="true"

GRUB\_ENABLE\_LINUX\_LABEL="true"

- 重新生成grub.cfg

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

- /etc/fstab 文件中到UUID手工修改成系统设备

- blkid查看UUID对应的磁盘设备,修改fstab文件

1. 去除个性化信息

· 安装分区扩展软件

yum intsall -y cloud-utils-growpart

- 设置第一次开机自动扩容根目录

chmod 755 /etc/rc.local

- 根分区扩容

/usr/bin/growpart /dev/vda 1

/usr/sbin/xfs\_growfs /

- 关闭虚拟机后执行信息清理工作

virt-sysprep -d node

* 1. 虚拟设备管理
     1. xml文件

1. xml详解

· XML配置文件

- 保存node虚拟机配置文件

virsh dumpxml node > demo.xml

- 清除模板配置

virsh undefine node

- xml模板

- <!-- 注释的开始

- --> 注释的结尾

- xml标签必须成对出现

- <keyword> ... ... </keyword>

Example:

# virsh dumpxml node1 > n1.xml

<domain type='kvm' id='4'>

<name>centos7.0</name>

<uuid>...</uuid>

<memory unit='KiB'>2097152</memory>

<currentMemory unit='KiB'>2097152<currentMemory>

<vcpu placement='static'>4</vcpu>

<resource>

<partition>/machine</partition>

</resource>

<os>

<type arch='x86\_64' machine='pc-i440fx-rhel7.0.0'>hvm</type>

<boot dev='hd' />

<bootmenu enable='yes' />

<bios useserial='yes' />

</os>

1. 内存,cpu配置

· XML配置文件

- id=4 个性化设置，去掉

- uuid 去掉

- memory unit 虚拟机最大使用内存，可以手动调整

- curentmemory 创建虚拟机使用内存

- cpu设置，可以调整

<vcpu placement='static'>2</vcpu>

- 资源设置，可以删除

<resource> ... ... </ resource >

- 系统配置及引导设备

<os>

<boot dev='hd' />

</os>

- 电源相关配置无需修改

<features> ... ... </fetures>

- cpu配置，可以修改成使用真机cpu

<cpu ... ... />

<cpu mode='host-passthrough'></cpu> //使用物理机CPU

- 时钟相关配置，可以删除

<clock offset='localtime' />

- 重启，关机，强制关闭对应命令

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

<on\_boot>restart</on\_boot>

<on\_crash>destroy</on\_crash>

1. 磁盘设置

- 内存及硬盘休眠相关设置，可以删除

<pm> ... ... </pm>

- 仿真设备相关配置

<devices> ... ... <devices>

- 其他配置

- 驱动安全标签，可删除

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

1. 硬件仿真设备

· 仿真设备配置

- type='pci' pci总线设备，可删除

- type='virtio-serial' 串口设备需要保留串口终端

- type='network' 网络配置需要修改

- type='pty' 串口终端，需要保留

- type='unix' 虚拟机通讯控制接口

- type='spicevmc' 图形图像相关配置可以删除

- type='tavlet' 数位板，可以删除

- type='mouse' 鼠标，保留

- type='keyboard' 键盘，保留

- graphics、video 图形图像显卡相关配置，可以删除

- redirdev 设备重定向，可以删除

- memballoon 内存气泡，可以动态调整内存

- 总线地址，别名配置，都可以删除

<address ...

<alias ...

- 硬盘配置，需要修改

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

<disk .type='file' device='disk'>

<driver name='qemu' type='qcow2' />

<source file='/var/lib/libvirt/images/node1.img' />

<backingStore type='file' index='1'>

<format type='raw' />

<source file='/var/lib/libvirt/images/rhel7.qcow2' />

<backingStore/>

</backingStore>

<target dev='sda' bus='scsi' />

/\*\*

\* <alias name='scsi0-0-0-0' />

\* <address type='drive' controller='0' bus='0' target='0' unit='0' />

\* 不知道的省去，让计算机自动识别

\*\*/

</disk>

<controller type='scsi' index='0' model='virtio-scsi'>

- usb相关设备配置，可以删除

<controller type='usb'

- 网卡配置文件

<interface type='bridge'>

<mac address='...' />

<source bridge='private1' />

<source bridge='vbr' />

<target dev='vnet0' />

<model type='virtio' />

/\*\* 去掉总线地址

\* <alias name='net0' />

\* <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' />

\*\*/

</interface>

* + 1. 创建访问虚拟机

1. 快速创建虚拟机

· 如何快速创建虚拟机

1. xml文件配置

- 把我们的xml模板文件进行复制

- 修改名称及磁盘文件

1. img文件创建

- 以我们刚刚装系统的模板为后端文件创建虚拟机

qemu-img create -b node.qcow2 node1.img

- 完成虚拟机注册及启动虚拟机

virsh define node1.xml

virsh start node1

virsh console node

1. 创建前端盘

# cd /var/lib/libvirt/images/

# qemu-img vreate -f qcow2 -b node.qcow2 node1.img 30G

1. 创建xml文件
2. 配置:名称,cpu,内存,硬盘,网卡

# sed 's/demo/node1/' demo.xml > /etc/libvirt/qemu/node1.xml

1. 定义虚拟机

# virsh define /etc/libvirt/qemu/node1.xml

# virsh start node1

# virsh console node1

1. 离线访问虚拟机

· 使用guestmount工具

- 支持离线挂载raw,qcow2格式虚拟机磁盘

- 可以在虚拟机关机的情况下直接修改磁盘中的文档

- 方便对虚拟机定制,修复,脚本维护

PS:需要注意SELinux机制的影响

· 基本用法

guestmount -a 虚拟机磁盘路径 -i 挂载点

# ll /var/lib/libvirt/images/node1.img

# guestmount -a /var/lib/ilibvirt/images/node1.img -i /mnt/

# ls /mnt

1. Day 02 Openstack概述、部署安装环境、部署Openstack OpenStack操作基础
2. Openstack概述
   * 1. 云计算简介
3. 什么是云计算

· 对于到底什么是云计算，至少可以找到100种解释

· 现阶段广为接受的是美国国家标准与技术研究院(NIST)定义

- 基于互联网的相关服务的增加,使用和交付模式

- 这种模式提供可用的,便捷的,按需的网络访问,进入可配置的计算资源共享池

- 这些资源能够被快速提供,只需投入很少的管理工作,或与服务供应商进行很少的交互

- 通常涉及通过互联网来提供动态易扩展且经常是虚拟化的资源

1. IaaS

· IaaS( Infrastructure as a Service ),即基础设施服务

· 提供给消费者的服务是对所有计算基础设施的利用,包括处理CPU,内存,存储,网络和其他基本的计算资源,用户能够不熟和运行任意软件,包括操作系统和应用程序

· IaaS通常分为三种用法:公有云,私有云和混合云

- 亚马逊( Elastic Compute Cloud )

- 阿里云,华为云

1. PaaS

· PaaS( Platform-as-a-Service ),意思是平台服务

· 以服务器平台或者开发环境作为服务进行提供就成为了PaaS

· PaaS运营商所需提供的服务,不仅仅是单纯的基础平台,还针对该平台的技术支持服务,甚至针对该平台而进行的应用系统开发,优化等服务

· 简单地说,PaaS平台是指云魂晶中的应用基础设施服务,也可以说是中间件服务

- Openshift

- 淘宝

1. SaaS

· SaaS( Software-as-a-Service )软件服务,是一种通过Internet提供软件的模式,厂商将应用软件统一部署在自己的服务器上,客户可以根据自己实际需求,通过互联网向厂商订购所需的应用软件服务

· 用户不用再购买软件,而是向提供商租用基于Web的软件来管理企业经营活动,不用对软件进行维护,提供商会全权管理和维护软件,同事也提供软件的离线操作和本地数据存储

- 金蝶友商网

- 安卓市场,appStore

* + 1. Openstack简介

1. 什么是Openstack

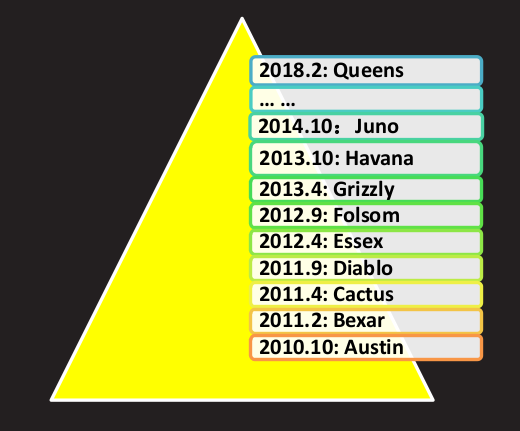
· Openstack是一个由NASA( 美国国家航空航天局 )和Rackspace合作研发并发起的项目

· Openstack是一套IaaS解决方案

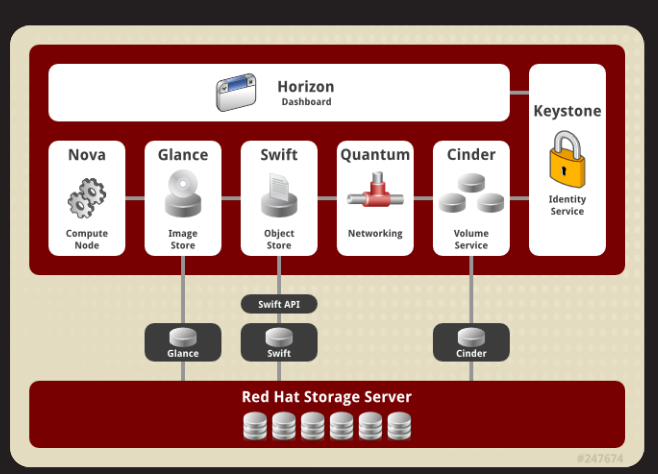
· Openstack是一个开源的云计算管理平台

· 以Apache许可证为授权

1. Openstack版本



1. Openstack结构图



1. Openstack主要组件

· Horizon

- 用于管理Openstack各种服务的,基于web的管理接口

- 通过图形界面实现创建用户,管理网络,启动实例等操作

· Keystone

- 为其他服务提供认证和授权的集中身份管理服务

- 也提供了集中的目录服务

- 支持多种身份认证模式,如密码认证,令牌认证,以及AWS( 亚马逊Web服务 )认证服务

- 为用户和其他服务提供了SSO认证服务

· Neutron

- 一种软件定义网络服务

- 用于创建网络,子网,路由器,管理浮动IP地址

- 可以实现虚拟交换机,虚拟路由器

- 可用于在项目中创建VPN

· Cinder

- 为虚拟机提供管理存储卷的服务

- 为运行在Nova中的实例提供永久的块存储

- 可以通过快照进行数据备份

- 经常应用在实例存储魂晶中,如数据库文件

· Glance

- 扮演虚拟机镜像注册的角色

- 允许用户为直接存储拷贝服务器镜像

- 这些镜像可以用于新建虚拟机的模板

· Nova

- 在节点上用于管理虚拟机的服务

- Nova是一个分布式的服务,能够与Keystone交互实现认证,与Glance交互实现镜像管理

- Nova被设计成在标准硬件上能够进行水平扩展

- 启动实例时,如果有则需要下载镜像

1. Openstack版本
2. Openstack结构图
   1. 部署安装环境
      1. Openstack环境准备
3. 准备虚拟机

· 虚拟机1:Openstack管理主机

- 2CPU,6G内存,50G硬盘

- 静态IP:192.168.1.10

· nova01,nova02计算节点

- 2CPU,4.5G内存,100G硬盘

- 配置静态IP:192.168.1.11(12)



1. 配置yum仓库

· CentOS7-1804.iso 系统软件

· RHEL7-extras.iso 提供Python依赖软件包

· RHEL7OSP-10.iso 光盘拥有众多目录,每个目录都是一个软件仓库,我们配置其中2个软件仓库

- Openstack主要软件仓库

rhel-7-server-openstack-10-rpms

- packstack 软件仓库

rhel-7-server-openstack-10-devtools-rpms

[student@room9pc01 ~]$ cd ~ftp

[student@room9pc01 ftp]$ mkdir openstack extras

[student@room9pc01 ftp]$ mount /linux-soft/04/openstack/RHEL7OSP-10.iso openstack/

[student@room9pc01 ftp]$ mount /linux-soft/04/openstack/RHEL7-extras.iso extras/

1. 设置DNS

· 系统换将准备

- Openstack安装时需要使用外部DNS来解析域名

去掉本地查询,设置真机DNS

- 将Openstack.tedu.cn域名对应的IP地址解析到安装的Openstack服务器(修改hosts文件)

[root@nova01 ~]# sed -i '/search/d' /etc/resolv.conf

1. NTP服务

· nova服务器之间的时间必须保持一致

· 编辑配置文件 /etc/chrony.conf

- server 192.168.1.254 iburst

- 重启服务

- systemctl restart chronyd

[root@nova01 ~]# sed -i 's/gateway/192.168.1.254/' /etc/chrony.conf

· 测试服务

# chronyc sources -v

//出现\*号代表NTP时间可用

* 1. 部署Openstack
     1. 基础环境准备(旧)

1. 配置IP地址

· 配置eth0为公共网络,网络地址192.168.1.0/24

· 配置eth1为隧道接口,网络地址192.168.2.0/24

· 关闭NetworkManager服务

· 禁用SELinux

· 卸载firewalld

1. 配置yum客户端

· CentOS7-1804.iso 系统软件

· RHEL7-extras.iso 提供Python依赖软件包

· RHEL7OSP-10.iso 光盘拥有众多目录,每个目录都是一个软件仓库,我们配置其中2个软件仓库

1. 配置卷组

· Openstack为虚拟机提供的云硬盘,本质上是本地的逻辑卷

· 逻辑卷创建于名为cinder-volumes的卷组

· 没有物理卷可以使用loop文件替代

- losetup loop0 /var/disk.img

- pvcreate /dev/loop0

- vgcreate cider-volumes /dev/loop0

* + 1. 基础环境准备(新)

1. 安装额外软件包

· 安装Openstack期间,有些软件包所以来的软件包并没有在安装过程中安装,这些软件包需要提前安装

- qemu-kvm

- libvirt-daemon

- libvirt-daemon-driver-qemu

- libvirt-client

- python-setuptools

· 软件包安装

[root@openstack ~]# for i in {openstack,nova01,nova02};do ssh $i "yum install -y qemu-kvm libvirt-client libvirt-daemon libvirt-daemon-driver-qemu python-setuptools";done

1. 检查Openstack基础环境部署

· 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

· 检查磁盘空间是否足够

[root@openstack ~]# grep 'vm' /proc/cpuinfo

[root@openstack ~]# grep 'ssse3' /proc/cpuinfo

//查看虚拟机是否开启支持虚拟化

/\* 没有开启虚拟化的,需要修改物理机配置 /etc/modprobe.d/kvm.conf修改字段options kvm\_intel nested=1,重启物理机后生效 \*/

* + 1. 安装Openstack

1. 配置packstack

· 安装Openstack需要使用packstack

- 首先安装openstack-packstack

[root@openstack ~]# yum install -y openstack-packstack

· 使用packstack创建通用应答文件

[root@openstack ~]# packstack --gen-answer-file=answer.ini

· 修改应答文件

[root@openstack ~]# vim answer.ini

42: CONFIG\_SWIFT\_INSTALL=n

//使用ceph对象存储

45: CONFIG\_CEILOMETER\_INSTALL=n

49: CONFIG\_AODH\_INSTALL=n

53: CONFIG\_GNOCCHI\_INSTALL=n

//上面三行与计费有关,实验中不需要开启

75: CONFIG\_NTP\_SERVERS=192.168.1.254

//NTP时间服务器地址

98: CONFIG\_COMPUTE\_HOSTS=192.168.1.11

//装虚拟机服务

102: CONFIG\_NETWORK\_HOSTS=192.168.1.10,192.168.1.11

//哪些主机安装nova网络

333: CONFIG\_KEYSTONE\_ADMIN\_PW=a

//管理Openstack的管理员密码,用户名默认为admin

840: CONFIG\_NEUTRON\_ML2\_TYPE\_DRIVERS=flat,vxlan

//Openstack运行中支持哪些网络协议

910: CONFIG\_NEUTRON\_OVS\_BRIDGE\_MAPPINGS=physnet1:br-ex

//定义一个虚拟交换机

921: CONFIG\_NEUTRON\_OVS\_BRIDGE\_IFACES=br-ex:eth0

//把物理机的eth0与交换机WAN口绑定

1179: CONFIG\_PROVISION\_DEMO=n

//不安装演示版

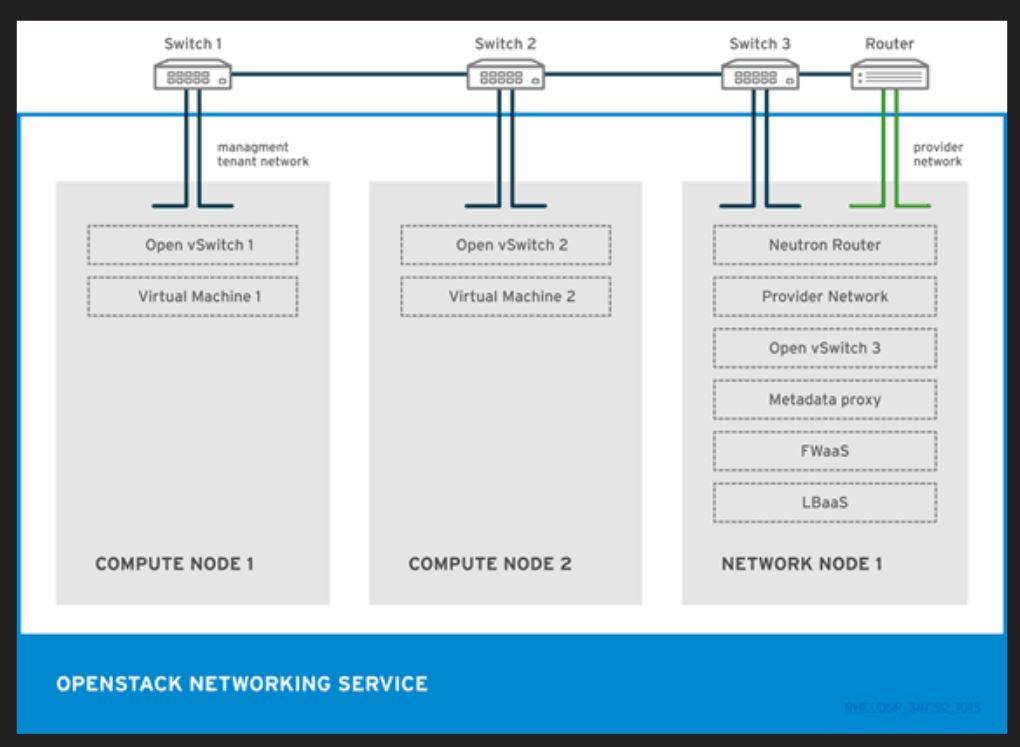
1. 一键部署Openstack

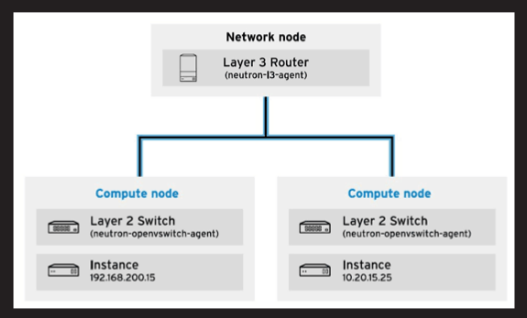
[root@openstack ~]# packstack --answer-file=answer.ini

* + 1. 网络配置

1. 网络拓扑

· 多计算节点的拓扑





1. 查看外部OVS网桥

· br-ex为外部OVS网桥

[root@openstack ~]# cat /etc/sysconfig/network-scripts/ifcfg-br-ex

ONBOOT="yes"

NM\_CONTROLLED="no"

IPADDR="192.168.1.10"

NETMASK="255.255.255.0"

GATEWAY="192.168.1.254"

DEVICE=br-ex

NAME=br-ex

DEVICETYPE=ovs

OVSBOOTPROTO="static"

TYPE=OVSBridge

1. 查看外部OVS网桥端口

· eth0为外部OVS网桥的端口

[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

1. 验证OVS配置

· ovs-vsctl show

[root@openstack ~]# ovs-vsctl show

fa0befc0-78b9-465a-95fb-44c676b39a65

Manager "ptcp:6640:127.0.0.1"

is\_connected: true

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"

Bridge br-int

Controller "tcp:127.0.0.1:6633"

is\_connected: true

fail\_mode: secure

Port int-br-ex

Interface int-br-ex

type: patch

options: {peer=phy-br-ex}

Port br-int

Interface br-int

type: internal

Port patch-tun

Interface patch-tun

type: patch

options: {peer=patch-int}

Bridge br-tun

Controller "tcp:127.0.0.1:6633"

is\_connected: true

fail\_mode: secure

Port patch-int

Interface patch-int

type: patch

options: {peer=patch-tun}

Port "vxlan-c0a8010b"

Interface "vxlan-c0a8010b"

type: vxlan

options: {df\_default="true", in\_key=flow, local\_ip="192.168.1.10", out\_key=flow, remote\_ip="192.168.1.11"}

Port br-tun

Interface br-tun

type: internal

ovs\_version: "2.5.0"

* 1. Openstack操作基础
     1. Horizon配置

1. 概述

· Horizon是一个用以管理、控制Openstack服务的Web控制面板,也称之为Dashboard仪表盘

· 可以管理实例、镜像、创建密钥对,对实例添加卷、操作Swift容器等。除此之外,用户还可以在控制面板中使用终端( console )或VNC直接访问实例

· 基于python的django web框架进行开发

1. 功能与特点

· 实例管理:创建、终止实例,查看终端日志,VNC连接,添加卷等

· 访问与安全管理:创建安全群组,管理密钥对,设置浮动IP等

1. Horizon

· Horizon BUG处理

- 安装虽然没有报错,但默认无法打开Horizon,这是一个BUG,因为少配置

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

- 添加"WSGIApplicationGroup %{GLOBAL}"

ServerAlias localhost

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

WSGIProcessGroup apache

WSGIApplicationGroup %{GLOBAL} <--- 添加

- 重新载入配置文件apachectl graceful( 重启httpd服务 )

1. 基本配置

· 命令行认证

[root@openstack ~]# openstack user list

Missing value auth-url required for auth plugin password

· 需要通过keystonerc\_admin文件验证

[root@openstack ~]# cat keystonerc\_admin

unset OS\_SERVICE\_TOKEN

export OS\_USERNAME=admin

export OS\_PASSWORD=a

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

[root@openstack ~]# source keystonerc\_admin

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

|  |  |
| --- | --- |
| ID | Name |
| 18011d8fe48345bba9428a2659ba8973 | admin |
| 194ad580d5264ebd95fc5ae553c47f19 | nova |
| c9b0fcd91e444f9fbccfafc68156b064 | neutron |
| f3dc576be4e8487689f6bc7adc40f255 | cinder |
| f9024ac42a2b413494d96a11fd50134d | glance |

1. 命令行接口

· 初始化环境变量

[root@openstack ~]# source ~/keystonerc\_admin

[root@openstack ~(keystone\_admin)]# env | grep OS

HOSTNAME=openstack.tedu.cn

OS\_REGION\_NAME=RegionOne

OS\_PASSWORD=admin

OS\_AUTH\_URL=http://192.168.1.10:5000/v2.0

OS\_USERNAME=admin

OS\_TENANT\_NAME=admin

· 使用帮助

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

1. Day 03 用户配额管理、云主机类型管理、镜像管理、网络管理案例和实例管理、安装额外计算节点
   1. 项目配额、用户管理
      1. 项目管理
2. 基本概念

· 项目:一组隔离的资源和对象。由一组关联的用户进行管理

· 在旧版本里,也用租户(tenant)来表示

· 根据配置的需求,项目对应一个组织、一个公司或是一个使用客户等

· 项目中可以有多个用户,项目中的用户可以在该项目创建、管理虚拟资源

· 具有admin角色的用户可以创建项目

· 项目相关信息保存到MariaDB中

· 缺省情况下,packstack安装的openstack中有两个独立的项目

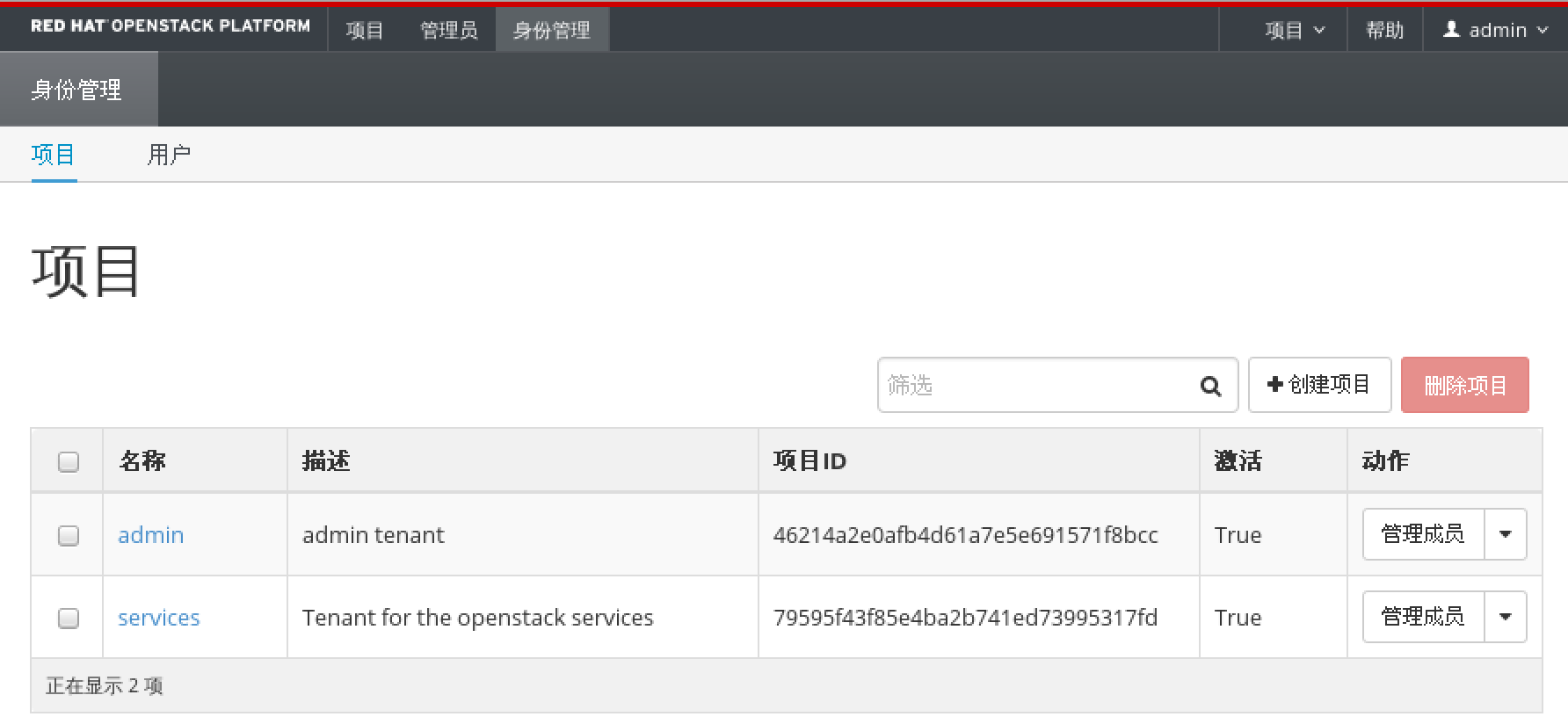
– admin:为admin账户创建的项目

– services:与安装的各个服务相关联

1. 在Horizon中管理项目

· 创建项目

- \* 为必填项



1. 通过命令行管理项目

· 创建名为myproject项目

[root@openstack ~(keystone\_admin)]# systemctl list-unit-files

[root@openstack ~(keystone\_admin)]# openstack project create myproject

· 列出所有项目

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

· 查看myproject详细信息

[root@openstack ~(keystone\_admin)]# openstack project show myproject

· 禁用与激活项目

[root@openstack ~(keystone\_admin)]# openstack project set --disable myproject

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

· 查看项目配额

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

· 更新可用vcpu数目为30

[root@openstack ~(keystone\_admin)]# nova quota-update --cores 30 myproject

· 删除myproject

[root@openstack ~(keystone\_admin)]# openstack project delete myproject

* + 1. 用户管理

1. 用户基本概念

· 用户在Openstack中用于身份认证

· 管理员用户admin一般在packstack安装过程中创建

· 其他用户由管理员用户出纳噶金,并指定可以访问的项目

· 非管理员用户创建后,保存到MariaDB中

· 非管理员用户具有以下权限

- 启动实例

- 创建卷和快照

- 创建镜像

- 分配浮动IP

- 创建网络和路由器

- 创建防火墙以及规则、规则策略

- 查看网络拓扑、项目使用概况等

1. 通过Horizon管理用户
2. 通过命令行管理用户

· 创建user2用户,指定密码为tedu.cn

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

(openstack) user create --password tedu.cn user2

· 设置user2的email地址

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

· 列出所有用户

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

· 查看user2信息

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

· 指定user可以访问myproject,角色为\_member\_

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

· 查看user2在myproject中的角色

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

· 禁用用户

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

· 激活用户

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

· 修改user2的密码为redhat

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

· 将user2从myproject中移除

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

· 删除user2用户

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

* + 1. 配额管理

1. 配额基础

· 管理员可以通过配额限制,防止资源的过度使用

· 配额基本项目,限制每个项目可以使用多少资源

· 这些操作上的功能限制,赋予了管理员对每个项目的精准控制

1. 资源参数

· 安全组规则:指定每个项目可用的规则数

· 核心:指定每个项可用的VCPU核心数

· 固定IP地址:指定每个项目可用的固定IP数

· 浮动IP地址:指定每个项目可用的浮动IP数

· 注入文件大小 :指定每个项目内容大小

· 注入文件路径:指定每个项目注入的文件路径长度

· 注入文件:指定每个项目允许注入的文件数目

· 实例:指定每个项目可创建的虚拟机实例数目

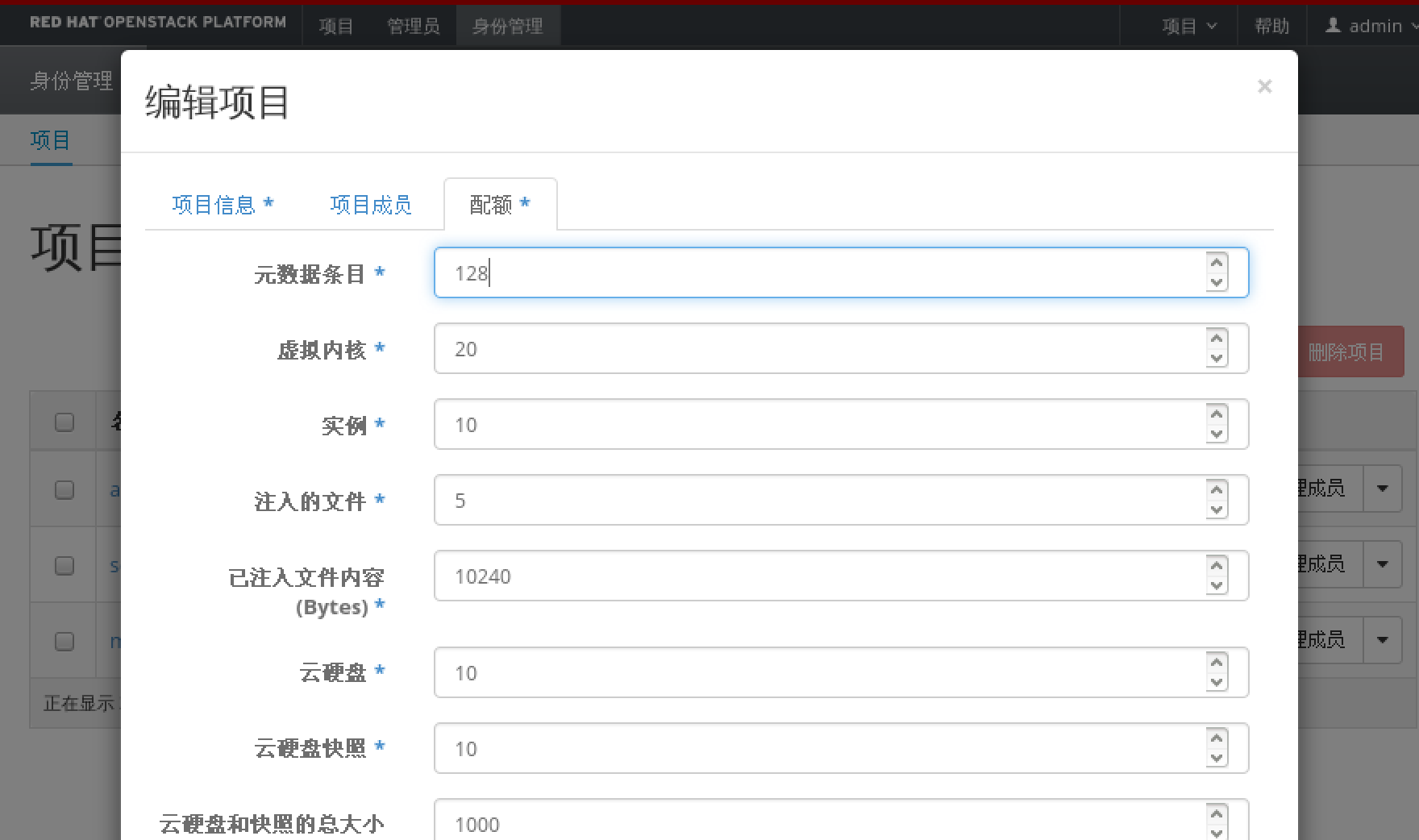
· 密钥对:指定每个项可创建的密钥数

· 元数据:指定每个项目可用的元数据数目

· 内存:指定每个项目可用的最大内存

· 安全组:指定每个项目可创建的安全组数目

1. 通过Horizon管理配额



1. 通过命令行管理配额

· 列出项目的缺省配额

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

· 列出myproject的配额

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

· 修改浮动IP地址配额

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

* 1. 云主机类型管理
     1. 云主机类型

1. 基本概念

· 云主机类型就是资源的模板

· 它定义了一台云主机可以使用的资源,如内存大小、

磁盘容量和CPU核心数等

· Openstack提供了几个默认的云主机类型

· 管理员还可以自定义云主机类型

1. 云主机类型采纳数

· Name:云主机类型名称

· ID:云主机类型ID,系统自动生成一个UUID

· VCPUs:虚拟CPU数目

· RAM(MB):内存大小

· Root disk(GB):外围磁盘大小。如果希望使用本地

磁盘,设置为0

· 临时磁盘:第二个外围磁盘

· swap磁盘:交换磁盘大小

1. 通过Horizon管理云主机类型



1. 通过命令行管理云主机类型

· 列出所有的云主机类型

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

· 创建一个云主机类型

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

· 删除云主机类型

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

* 1. 镜像管理
     1. 镜像基础

1. 基本概念

· 在红帽Openstack平台中,镜像指的是虚拟磁盘文件,磁盘文件中应该已经安装了可启动的操作系统

· 镜像管理功能由Glance服务提供

· 它形成了创建虚拟机实例最底层的块结构

· 镜像可以由用户上传,也可以通过红帽官方站点下载

1. Glance磁盘格式

· raw : 非结构化磁盘镜像格式

· vhd : VMWare、Xen、Microsoft、VirtualBox等均支持的通用磁盘格式

· vmdk : 是VMWare的虚拟磁盘格式

· vdi : VirtualBox虚拟机和QEMU支持磁盘格式

· iso : 光盘数据内容的归档格式

· qcow2 : QEMU支持的磁盘格式,空间自动扩展,并支持写时复制copy-on-write

1. 镜像服务

· 镜像服务提供了服务器镜像的拷贝、快照功能,可以作为模板快速建立、启动服务器

· 镜像服务维护了镜像的一致性

· 当上传镜像时,容器格式必须制定

· 容器格式指示磁盘文件格式是否包含了虚拟机元数据

1. 镜像容器格式

· bare : 镜像中没有容器或元数据封装

· ovf : 一种开源的文件规范,描述了一个开源、安全、有效、可拓展的便携式虚拟打包及软件分布格式

· ova : OVA归档文件

· aki : 亚马逊内核镜像

· ami : 亚马逊主机镜像

* + 1. 镜像应用

1. 通过Horizon管理镜像



1. 通过命令行管理镜像

· 上传镜像

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

· 列出镜像

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

· 查看镜像详情

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

· 修改镜像属性

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

· 另存镜像为本地文件

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

· 删除镜像

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

* 1. 网络管理
     1. 网络和路由

1. Openstack网络工作原理

· 实例被分配到子网中,以实现网络连通性

· 每个项目可以有一到多个子网

· 在红帽的Openstack平台中,OpenStack网络服务是缺省的网络选项,Nova网络服务作为备用

· 管理员能够配置丰富的网络,将其他Openstack服务连接到这些网络的接口上

· 每个项目都能拥有多个私有网络,各个项目的私有网络互相不受干扰

1. 网络类型

· 项目网络:由Neutron提供的项目内部网络,网络间可用VLAN隔离

· 外部网络:可以让虚拟机接入外部网络,但需要配置浮动IP地址

· 提供商网络:将实例连接到现有网络,实现虚拟机实例与外部系统共享同一二层网络

1. 通过Horizon创建网络

· 项目网络由租户在自己的项目中创建

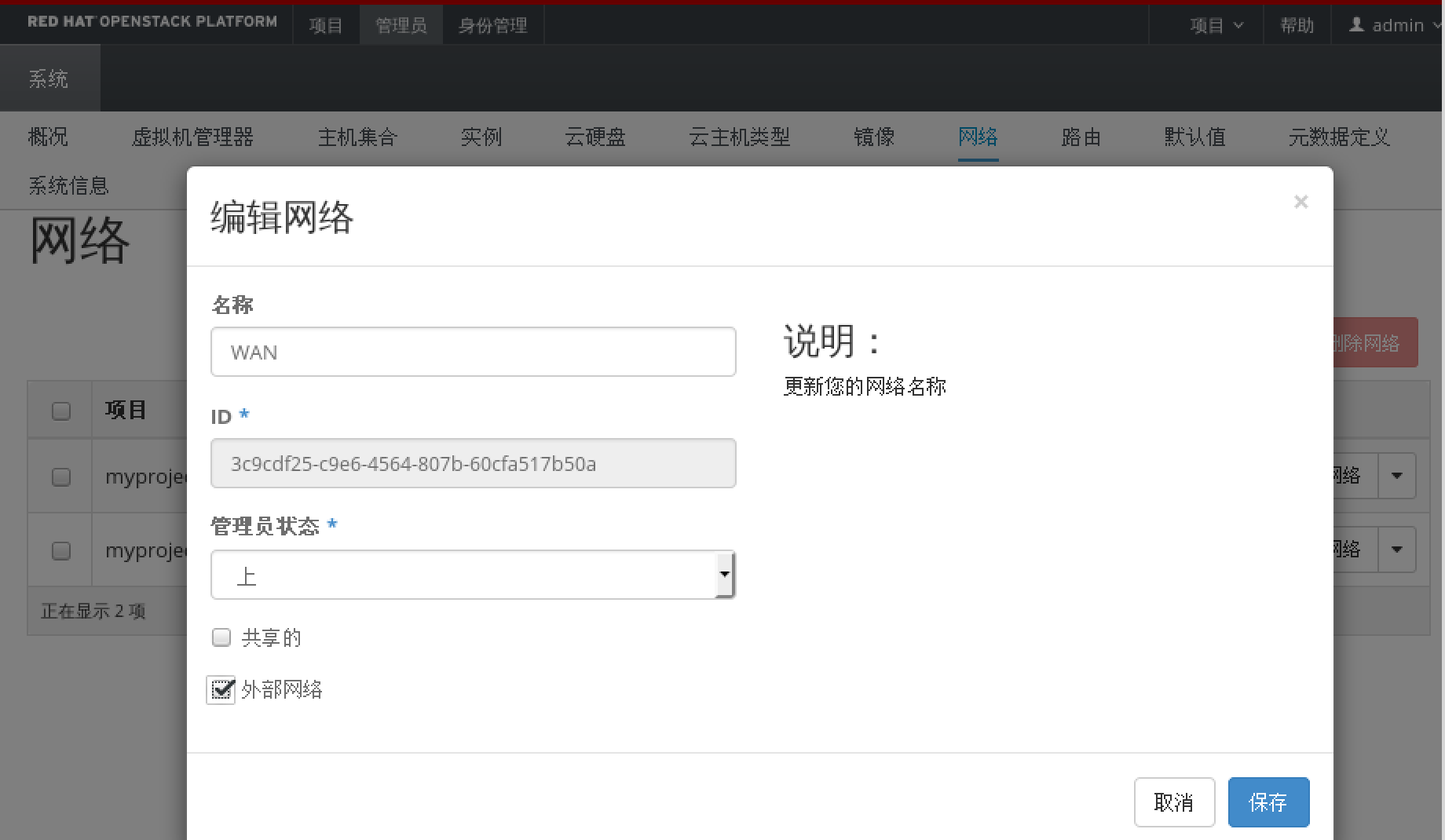






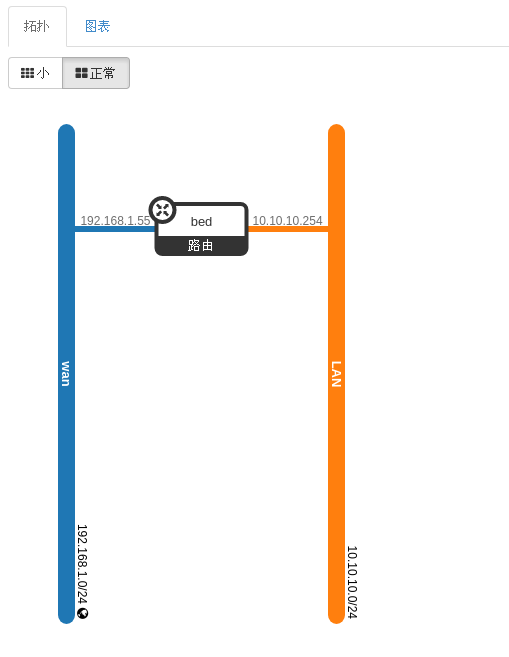
1. 通过Horizon设置外部网络

· 外部网络只有管理员有权限设置

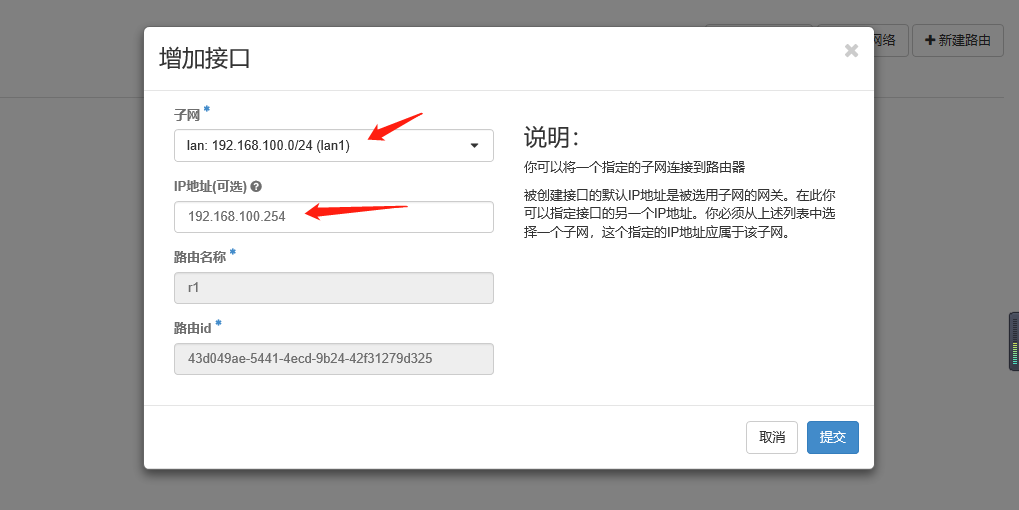


1. 配置路由器

· 内外网通过路由器连接起来



· 创建路由接口,与内网相连



1. 通过命令行管理网络

· 创建网络

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

· 创建子网

[root@openstack ~(keystone\_admin)]# neutron subnet-create --name subnet3 --gateway 192.168.1.254 --allocation-pool

start=192.168.1.100,end=192.168.1.200 --enable-dhcp internal

192.168.1.0/24 --tenant-id ff387162978643f894cdd1c98597160c

· 查看网络详情

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

· 新建路由器

[root@openstack ~(keystone\_admin)]# neutron router-create --tenant-id ff387162978643f894cdd1c98597160c r2

· 删除网络

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

[root@openstack ~(keystone\_admin)]# neutron router-delete r2

* + 1. 浮动IP地址

1. 浮动IP地址的作用

· 浮动IP一般是花钱购买的

· 浮动IP地址用于从外界访问虚拟机实例

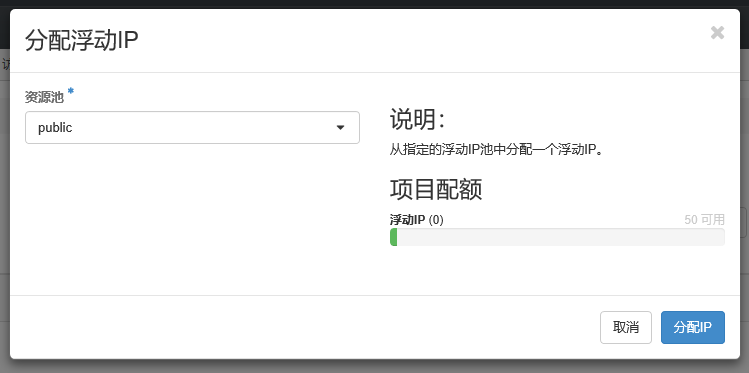
· 浮动IP地址只能从现有IP地址池中分配

· 创建外部网络时,浮动IP地址池被定义

· 虚拟机实例启动后,可以为其关联一个浮动IP地址

· 虚拟机实例也可以解除IP地址绑定

1. 在Horizon中管理浮动IP地址



1. 通过命令行管理浮动IP地址

· 分配地址

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

· 查看地址

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

* 1. 安全和实例管理
     1. 安全管理

1. 安全组

· 安全组用于控制对虚拟机实例的访问

· 安全组在高层定义了哪些网络及哪些协议是被授权可以访问虚拟机实例的

· 每个项目都可以定义自己的安全组

· 项目成员可以编辑默认的安全规则,也可以添加新的安全规则

· 所有的项目都有一个默认的default安全组

1. 安全组规则

· 安全组规则定义了如何处理网络访问

· 规则基于网络或协议定义

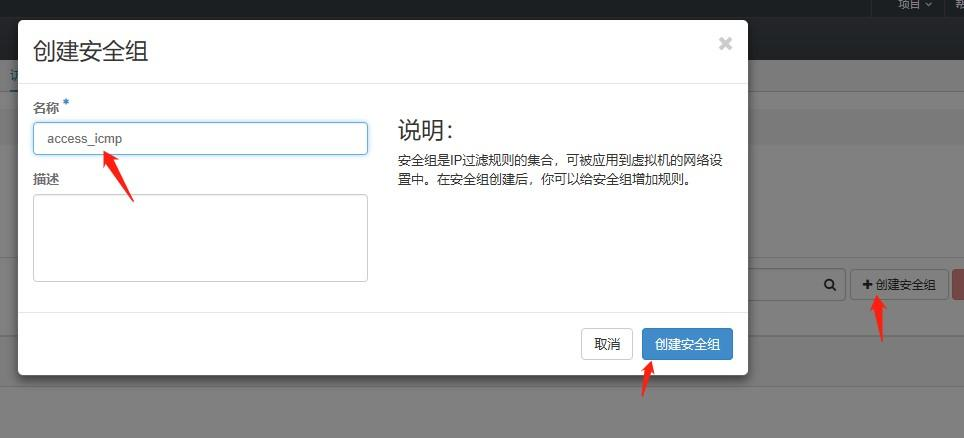
· 每个规则都有出和入两个方向

· 规则也可以指定ip协议版本

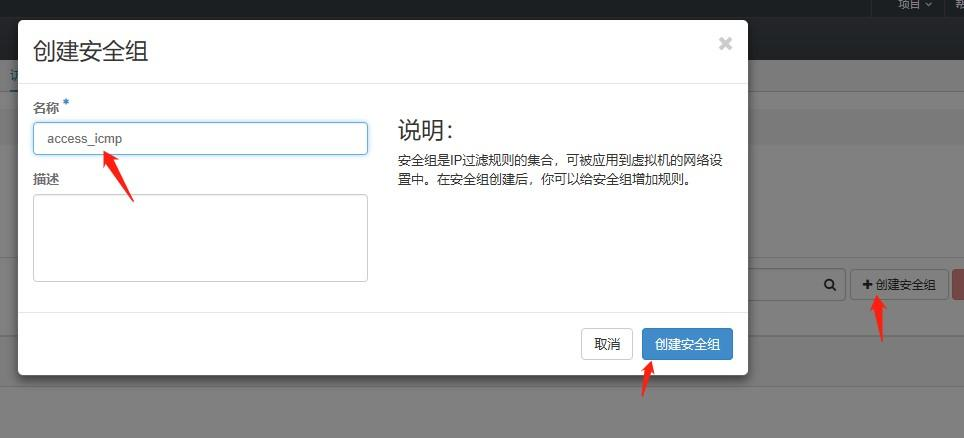
· 默认的安全组规则,允许虚拟机实例对外访问,但是阻止所有对虚拟机实例的访问

1. 创建安全组及规则

· 创建安全组



· 点击新建安全组的管理规则,进行规则定义



* + 1. 实例管理

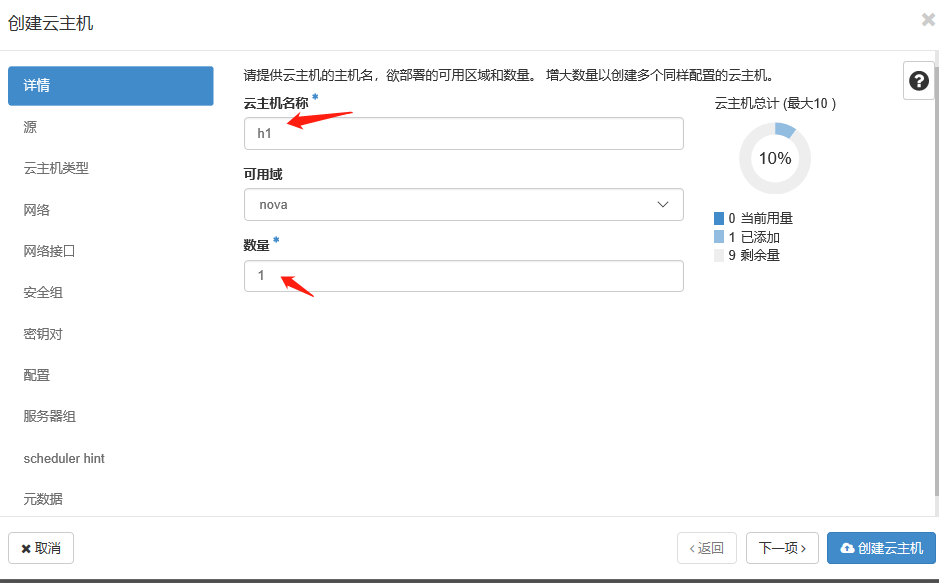
1. 云主机实例要求

· 使用m2.tiny云主机类型

· 云主机处于新建的安全组中

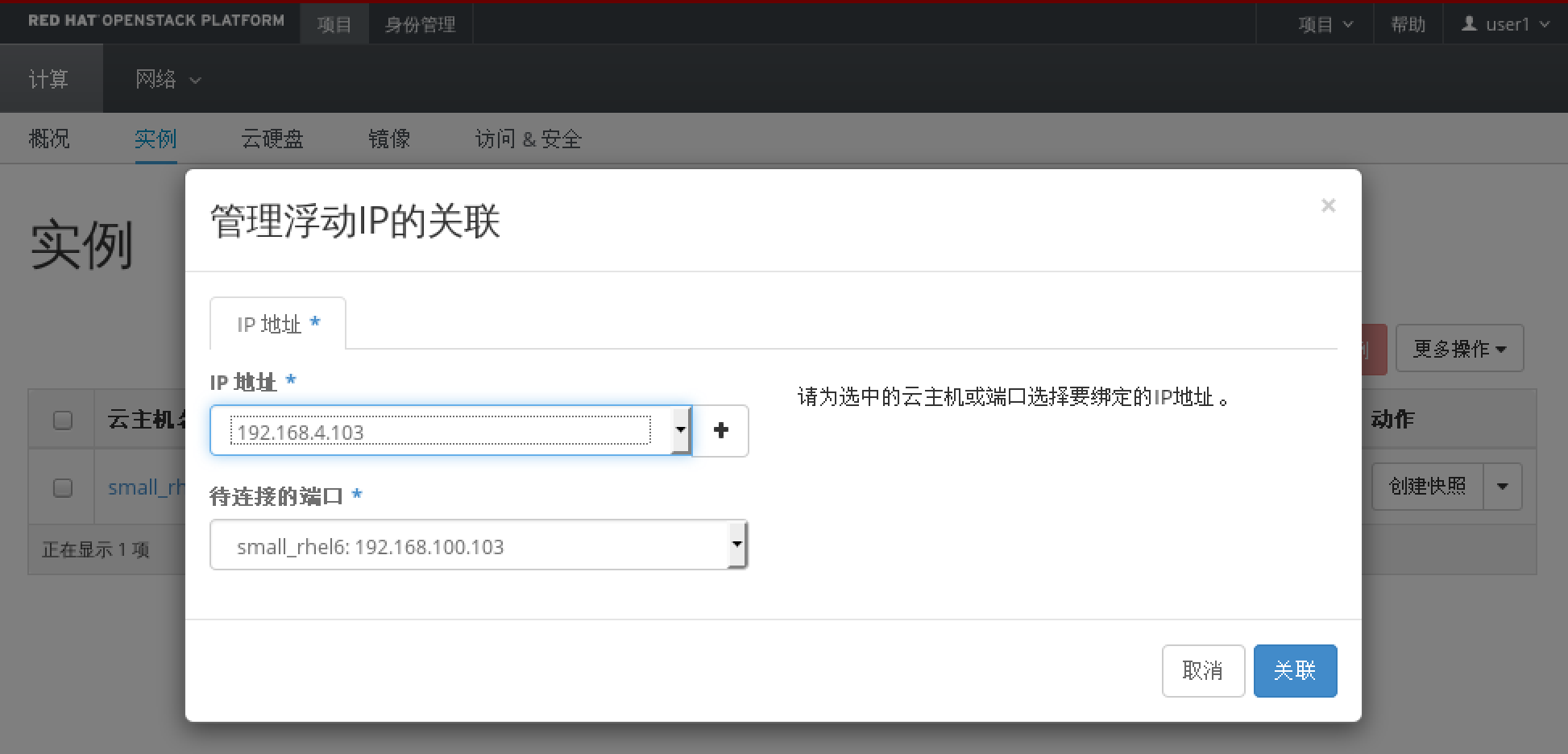
· 将云主机接入到内部网络

1. 在Horizon中创建云主机



1. 绑定浮动IP地址

· 远程主机通过访问浮动IP地址来访问云主机



* 1. 计算节点扩容
     1. 环境准备

1. 准备扩容主机

· 扩容Openstack计算节点

- 参考nova01配置过程

- 配置静态ip : 192.168.1.12与主机名

- 保证与Openstack、nova01能够互ping

- 配置 /etc/resolv.conf , 删除search开头行

- 配置时间同步 /etc/chrony.conf

1. 准备软件环境

· 配置yum源,软仓库一共4个(10670)

· 安装依赖软件包

qemu-kvm,libvirt-client,libvirt-daemon,libvirt-daemon-driver-qemu,python-setuptools

* + 1. 安装计算节点

1. 修改应答文件

· Openstack计算节点采用远程安装

- 即在Openstack上执行指令,远程安装nova02

- 远程安装使用ssh指令,从Openstack上必须能够通过ssh命令登录nova02

· 在Openstack上修改配置文件

[root@openstack ~]# vim answer.ini

98: CONFIG\_COMPUTE\_HOST=192.168.1.11,192.168.1.12

102:CONFIG\_NETWORK\_HOSTS=192.168.1.10,192.168.1.11,192.168.1.12

1. 安装节点

· 在Openstack上重新执行安装命令

[root@openstack ~]# packstack --answer-file answer.ini

· 本机已安装的服务不会被覆盖,只有改动后的选项才需要重新配置

· 安装后,appache配置会被还原,需要重新添加

- WSGIApplicationGroup %{GLOBAL}

1. 查看节点状态

· 安装后的状态如图所示



1. 迁移云主机

· 有多个nova计算节点的时候,我们可以选择性的把某一云主机从某台机器上迁移到另外一台机器上

· 迁移条件

- nova计算节点与Openstack管理节点都能相互ping通,主机名称也要能ping通

- 所有计算节点安装qemu0img-rhev,qemu-kvm-rhev

- 如未安装,在安装以后需要重启libvirt服务

· watch命令

[student@room9pc01 ~]$ watch -n 1 'ls -l'

//每隔一秒执行一次'ls -l'

· screen工具 : 分屏

CaS横切

Ca| 竖切

Cac新建终端

Ca<Tab> 切换窗口

CaX关闭窗口

Ca[0-9] 切换到bash[n]

1. Day 04 阿里云简介、ECS实战、华为云简介、云计算测评报告
2. 阿里云产品
   * 1. 云计算基础
3. 什么是云计算

· 对于到底什么是云计算，至少可以找到100种解释

· 现阶段广为接受的是美国国家标准与技术研究院(NIST)定义

- 基于互联网的相关服务的增加,使用和交付模式

- 这种模式提供可用的,便捷的,按需的网络访问,进入可配置的计算资源共享池

- 这些资源能够被快速提供,只需投入很少的管理工作,或与服务供应商进行很少的交互

- 通常涉及通过互联网来提供动态易扩展且经常是虚拟化的资源

1. IaaS

· IaaS( Infrastructure as a Service ),即基础设施服务

· 提供给消费者的服务是对所有计算基础设施的利用,包括处理CPU,内存,存储,网络和其他基本的计算资源,用户能够不熟和运行任意软件,包括操作系统和应用程序

· IaaS通常分为三种用法:公有云,私有云和混合云

- 亚马逊( Elastic Compute Cloud )

- 阿里云,华为云

1. PaaS

· PaaS( Platform-as-a-Service ),意思是平台服务

· 以服务器平台或者开发环境作为服务进行提供就成为了PaaS

· PaaS运营商所需提供的服务,不仅仅是单纯的基础平台,还针对该平台的技术支持服务,甚至针对该平台而进行的应用系统开发,优化等服务

· 简单地说,PaaS平台是指云魂晶中的应用基础设施服务,也可以说是中间件服务

- Openshift

- 淘宝

1. SaaS

· SaaS( Software-as-a-Service )软件服务,是一种通过Internet提供软件的模式,厂商将应用软件统一部署在自己的服务器上,客户可以根据自己实际需求,通过互联网向厂商订购所需的应用软件服务

· 用户不用再购买软件,而是向提供商租用基于Web的软件来管理企业经营活动,不用对软件进行维护,提供商会全权管理和维护软件,同事也提供软件的离线操作和本地数据存储

- 金蝶友商网

- 安卓市场,appStore

* + 1. 阿里云产品和技术

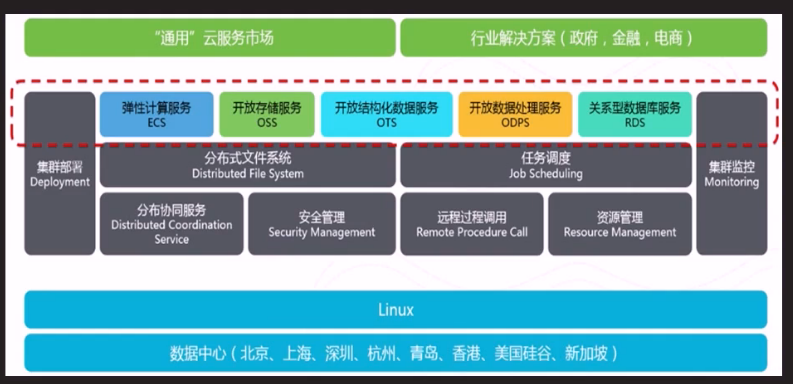
1. 阿里云简介

· 阿里巴巴业务增长的推手

· 为阿里巴巴30个业务提供底层技术支持

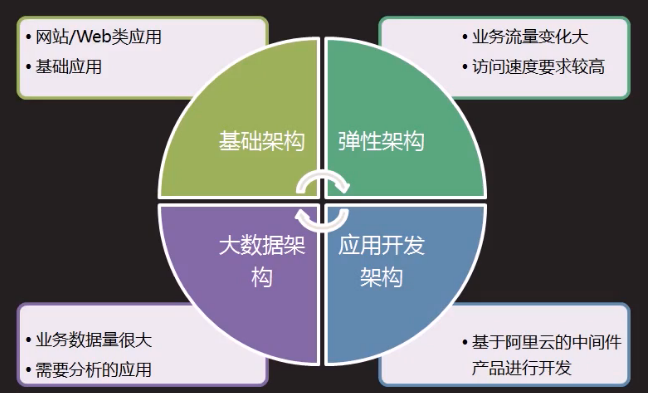


1. 阿里云的体系架构



* + 1. 阿里云应用架构

1. 架构建议总览



1. 基础应用架构

· SLB负载均衡

1. 流量入口
2. ESC应用服务器(ESC运行应用，如CentOS 6.3服务器)
3. OSS对象存储、EDS数据库(文件数据库 [MySQL6.5] )

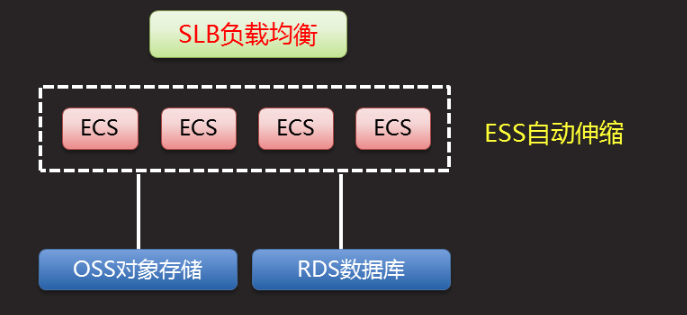
DR,NAT,TUN,FULLNAT

FULLNAT可以跨VLAN

1. 弹性应用架构

· ESS自动伸缩,按也无需要自动增加和删除ECS实例

· 按时间伸缩或按性能伸缩



* 1. 华为云
     1. 简介

1. 华为云简介

· 华为云是领先的云服务品牌,用在线的方式将华为30年ICT基础设施以云服务的方式提供给客户,为企业提供稳定可靠,安全可信,可持续创建新的云服务

· 面向大中型企业,华为云帮助他们解决云转型中的困难,更好地把握未来,引领数字化转型

· 面向中小型企业,华为云帮助他们应对互联网业务云基础设施2.0时代的新挑战,陪伴他们成长

· 到2018年7月,华为云已上线16大类120+云服务,包括制造,电商,游戏,金融,车联网,SAP,HPC,IoT,安全,DevOps等60+解决方案

· 在2017年9月的华为全联接大会上,华为云正式发布RI( Enterprise Intelligence )企业只能,将华为在人工智能领域的技术积累,最佳实践与企业应用场景相结合,为企业客户提供一站式的人工智能平台型服务

* + 1. 产品介绍

1. 华为云产品介绍

· 全面覆盖各应用领域

- 弹性计算ECS,存储,安全,数据库,区块链等...

· 弹性云服务器ECS( Elastic Compute Service )

- 是一种弹性可伸缩的计算服务( 虚拟机 ),助您降低IT成本,提升运维效率,使您更专注于核心业务创新

* 1. ECS云服务器管理
     1. 华为云控制台

1. 注册华为云用户

· 访问官网https://huaweicloud.com/

- 注册一个华为云用户

- 需要一个可用的手机号验证

PS:网站可能会改版,具体操作以实际登录为准

1. 使用华为云Web控制台

· 登陆后选择"控制台"

- 若要购买云计算服务或相关产品

- 账户余额不低于100元,可通过微信/支付宝等充值

1. 完成实名认证

· 通过实名认证才可以购买云服务器

- 绑定一张本人的银行卡

- 或者提交身份证照片,都可以通过实名认证

* + 1. ECS选购及操作

1. 服务器选型要点

· 重点注意事项

- 计费方式( 包年包月,按需付费 )

- 规格( CPU/内存/硬盘 )

- 镜像( Linux系统 )

- 登录方式( 设密码 )

- 计费方式( 按带宽,按流量 )

1. 选购一台ECS云服务器

· 选型 --> 立即购买 --> 接受许可协议

· 提交申请 --> 确认实例

1. ECS基本操作

· 基本管理:开机,关机,停止,重启,远程登录

· 更多操作:更改配置,重装系统,换系统

* + 1. ECS选购及操作

1. 远程控制云服务器

· 远程控制协议

- RDP( 端口号3389 ),适用于Windows服务器

(Remote Desktop)

- SSH( 端口号22 ),适用于Linux服务器,防火墙等设备

· 常用SSH管理工具

- PC端:Xshell,SecureCRT,Putty等

- 手机端:iTerminal,Termius,阿里云APP等

1. 创建网站'

· 在云服务器上启用Web服务

- 安装httpd软件包

- 启动名为httpd的系统服务,并设置开机自动运行

· 特别注意事项

- ECS实例需要开放Web端口

22 ssh远程控制协议

80 普通网站服务

443 加密网站服务

· 向云服务器上传,部署Web网页

- 注意目标位置/var/www/html目录

· 从互联网访问刚建立的网站,查看页面效果

1. Day 06 Docker概述、部署Docker、Docker镜像、Docker基本命令
2. Docker概述
3. 概述
4. 什么是容器

· 容器技术已经成为应用程序封装和交付的核心技术

· 容器技术的核心有以下几个内核技术组成：

- CGroups(Control Groups)-资源管理(限制容器使用的资源)

- NameSpace-进程隔离，通信隔离，网络隔离

- SELinux安全

· 由于是在物理机上实施隔离，启动一个容器，可以像启动一个进程一样快速

1. 什么是Docker

· Docker是完整的一套容器管理系统

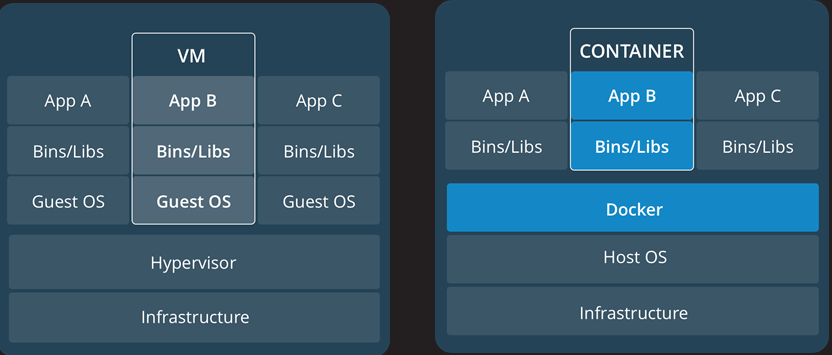
· Docker提供了一组命令，让用户更加方便直接地使用容器技术，而不需要过多关心底层内核技术

1. Docker特性
2. Docker优点

· 相比于传统的虚拟化技术，容器更加简洁高效

· 传统虚拟机需要给每个VM安装操作系统

· 容器使用的共享公共库和程序



1. Docker缺点

· 容器的隔离性没有虚拟化强

· 公用Linux内核，安全性有先天缺陷

· SELinux难以驾驭

· 监控容器和容器排错是挑战

1. 部署Docker
2. 安装前准备

· 需要64位操作系统

· 至少RHEL6.5以上版本，强烈推荐RHEL7

· 关闭防火墙(不是必须)

1. 安装Docker
2. 配置yum源

· docker软件包位于光盘RHEL7OSP-10.iso中

- 在物理机的ftp目录下创建rhelosp

# mkdir /var/frp/rhelosp

- 把光盘挂载到目录下

# mount -t iso9660 -o ro,loop RHEL7OSP-10.iso /var/ftp/rhelosp

· 在虚拟机中配置yum源,指向刚刚创建的目录:

[rhelosp]

name=CentOS rhel osp

baseurl=ftp://192.168.1.254/rhelosp

enabled=1

gpgcheck=0

· 软件包安装

# yum install -y docker

# systemctl restart docker

# systemctl enable docker

1. 软件包列表

· docker-engine

· docker-engine-selinux

1. docker安装(教学环境)

· 物理机

1. 创建一台4G内存的虚拟机
2. 创建虚拟机硬盘

# qemu-img create -b node.qcow2 -f qcow2 node1.img 20G

1. 创建虚拟机配置文件

# sed 's/demo/node1/' demo.xml > /etc/libvirt/qemu/node1.xml

1. 定义虚拟机

# virsh define /etc/libvirt/qemu/node1.xml

1. 编辑虚拟机修改内存

# virsh edit node1

1. 启动虚拟机

# virsh start node1

1. 创建自定义yum源

# cp docker-engine\* /var/ftp/public

# cd /var/ftp/public

# createrepo .

· 虚拟机

1. 修改主机名和静态ip地址

# echo docker01 > /etc/hostname

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

BOOTPROTO="static"

IPADDR="192.168.1.10"

NETMASK="255.255.255.0"

GATEWAY="192.168.1.254"

1. 配置yum，安装docker

# vim /etc/yum.repos.d/docker.repo

[local\_docker]

name=CentOS docker

baseurl="ftp://192.168.1.254/public"

enabled=1

gpgcheck=0

# yum install docker-engine

# systemctl enable docker

# reboot

1. docker安装(vmware)
2. 做一台CentOS7模板机【最简安装，安装常用软件，网卡模式调整为NAT，设置为1网段，配置好yum】
3. 给模板机做快照，通过链接克隆新建虚拟机"docker01"，安装docker软件包

[root@docker01~]# vim /etc/yum.repos.d/CentOS-Base.repo

[docker]

name=centos\_docker

baseurl=file:///software/docker

enabled=1

gpgcheck=0

[root@docker01~]# ls /software/docker/

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

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

docker\_images.zip

repodata

[root@docker01~]# yum install -y docker-engine

[root@docker01 ~]# systemctl start docker

[root@docker01 ~]# systemctl enable docker

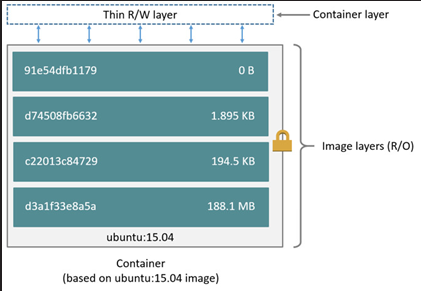
1. Docker镜像
2. 基本概念
3. 什么是镜像

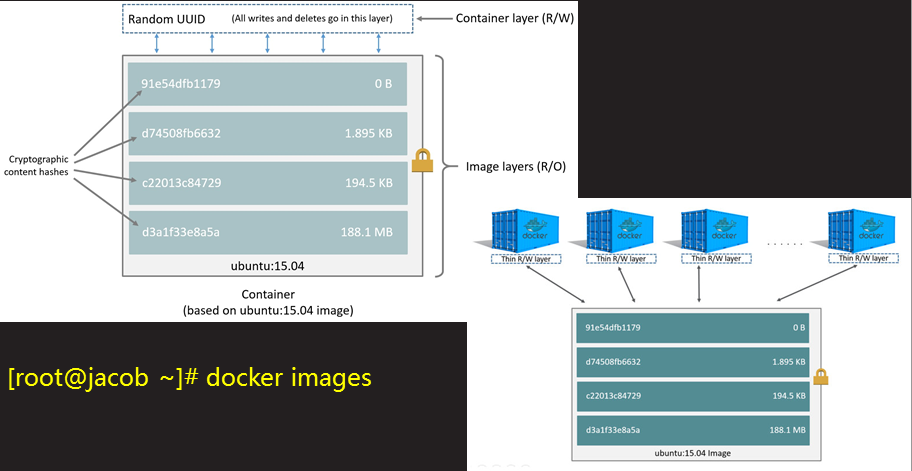
· 在Docker中容器是基于镜像启动的

· 镜像是启动容器的核心

· 镜像采用分层设计

· 使用快照的COW技术，确保底层数据不丢失





1. Docker hub镜像仓库

· <https://hub.docker.com>

· Docker官方提供公共镜像的仓库(Registry)

[root@docker01 ~]# docker version

//查看版本

[root@docker01 ~]# docker images

//查看镜像

[root@docker01 ~]# docker search

//搜索镜像

[root@docker01 ~]# docker pull

//下载镜像

[root@docker01 ~]# docker help

[root@docker01 ~]# docker help push

//查看帮助

1. 镜像操作
2. 下载、上传镜像

· 下载

[root@docker01 ~]# docker pull busybox

[root@docker01 ~]# docker images

· 上传

[root@docker01 ~]# docker help push

[root@docker01 ~]# docker push busybox

1. 导入、导出镜像

· 导入镜像(通过tar包文件导入镜像)

[root@docker01 docker\_images]# ls

centos.tar nginx.tar redis.tar registry.tar ubuntu.tar

[root@docker01 docker\_images]# for i in `ls`;do docker load < $i;done

[root@docker01 docker\_images]# docker images

· 导出镜像(将本地镜像导出为tar文件)

[root@docker01 ~]# docker images

[root@docker01 ~]# docker save image\_name > xx.tar

1. 启动镜像

· 启动centos镜像生成一个容器

[root@docker01 docker]# docker images

[root@docker01 ~]# docker run -it centos bash

· 开启另一个终端(查看容器信息)

[root@docker01 ~]# docker ps

/\*

\* 1.查看容器信息

[root@3574ccd720e7 /]# vi /etc/yum.repos.d/yum.repo

[CentOS]

name=CentOS7\_Base

baseurl=ftp://192.168.1.130/CentOS7

enabled=1

gpgcheck=0

[root@3574ccd720e7 /]# yum -y install net-tools

[root@3574ccd720e7 /]# ifconfig

inet 172.17.0.2 netmask 255.255.0.0 broadcast 0.0.0.0

[root@3574ccd720e7 /]# sleep 123456

[root@docker01 ~]# ps -ef | grep 123456

root 3097 2735 0 19:14 pts/1 00:00:00 sleep 12345

[root@docker01 ~]# kill 3097

容器中运行的进程可以被宿主机看到(并未完全隔离)

当宿主机杀死进程时，容器中的进程被杀死

\* 2.开启路由转发的方式

[root@docker01 ~]# sysctl -w net.ipv4.ip\_forward=1

[root@docker01 ~]# vim /etc/sysctl.conf

net.ipv4.ip\_forward=1

\*/

1. 启动不同镜像中遇到的问题
2. 启动docker时的细微区别

[root@docker01 ~]# docker run -it centos:latest /bin/bash

//完整格式

[root@docker01 ~]# docker run -it centos:latest

//不加程序，启动默认程序

[root@docker01 ~]# docker run -it centos

//不加标签，默认为latest标签

1. 启动nginx和redis

[root@docker01 ~]# docker run -it nginx

//直接宕死，因为启动的是一个服务，监听的是一个端口

/\* 可以启动镜像中的可交互程序 \*/

[root@docker01 ~]# docker run -it nginx /bin/bash

1. Docker基本命令
2. 镜像常用命令
3. docker images //查看镜像列表

· 查看镜像列表

- 镜像仓库名称

- 镜像标签

- 镜像ID

- 创建时间

- 大小

1. docker history //查看镜像制作历史

· 查看镜像历史

- 了解镜像制作过程

- 详细参考后面的dockerfile内容

1. docker inspect //查看镜像底层信息

· 查看镜像底层信息

- 了解镜像环境变量、存储卷、标签等信息

1. docker pull //下载镜像
2. docker push //上传镜像
3. docker rmi //删除本地镜像

· 删除本地镜像

- 注意：启动容器时删除镜像会提示错误

docker rmi 镜像名称：标签

1. docker save|load //镜像另存为tar包|使用tar包导入镜像
2. docker search //搜索镜像
3. docker tag //修改镜像名称和标签

· 重命名镜像名称(复制)

[root@docker01 ~]# docker tag centos:latest abc:ooxx

[root@docker01 ~]# docker images

可以看到centos和abc两个镜像的镜像id相同，相当于一个硬链接

1. 容器常用命令
2. docker run //运行容器
3. 选项

-i 交互式

-t 终端

-d 后台进程

1. 启动一个交互式的终端

[root@docker01 ~]# docker run -it centos /bin/bash

[root@docker01 ~]# docker run -it centos /bin/python

1. 启动一个服务(非交互)

[root@docker01 ~]# docker run -d nginx

20bc026f69eaa74402ed5a5ab96fd831e490ea5698c6004c840bc9538a37f923

1. 启动一个交互式的终端，放在后台运行

[root@docker01 ~]# docker run -itd centos

1. 启动容器时给容器命名

[root@docker01 ~]# docker run --name web01 -d nginx

1. docker ps //查看容器列表
2. 列出容器列表

- docker ps 查看正在运行的容器

- docker ps -a 查看所有容器列表

- docker ps -aq 仅显示容器id

1. docker stop|start|restart //关闭|启动|重启容器
2. docker支持函数重录
3. 停止所有容器

[root@docker01 ~]# docker stop $(docker ps -aq)

1. docker attach|exec //进入容器

attach使用上帝进程连接容器，当退出时，容器关闭

ctrl+pq可以将正在运行的终端挂到后台，适用于使用attach的场合

exec进入容器，相当于是新建一个进程，不影响其他进程，退出容器不关闭

[root@docker01 ~]# docker start 35

[root@docker01 ~]# docker exec -it 35 /bin/bash

[root@3574ccd720e7 /]#

1. docker inspect //查看容器底层信息

· 双括号，以点分隔

[root@docker01 ~]# docker inspect -f "{{.NetworkSettings.IPAddress}}" 35

172.17.0.2

[root@docker01 ~]# docker inspect -f "{{.NetworkSettings.MacAddress}}" 35

02:42:ac:11:00:02

1. docker top //查看容器进程列表
2. docker rm //删除容器

[root@docker01 ~]# docker rm $(docker ps -aq)

1. Day 07 自定义镜像及仓库、持久化存储、Docker网络架构
2. 自定义镜像与仓库
3. 自定义镜像
4. docker commit

· 使用镜像启动容器，在该容器基础上修改

· 另存为另一个镜像

1. Dockerfile

- FROM：基础镜像

- MAINTAINER：镜像创建者信息

- EXPOSE：开放的端口

- ENV：设置变量

- 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

1. 使用Dockerfile工作流程

[root@docker01 ~]# mkdir oo

[root@docker01 ~]# cd oo

[root@docker01 ~]# cp /etc/yum.repos.d/CentOS-Base.repo local.repo

Dockerfile01

[root@docker01 oo]# vim Dockerfile

FROM centos:latest

RUN rm -f /etc/yum.repos.d/\*

ADD local.repo /etc/yum.repos.d/local.repo

RUN yum install -y nettools psmisc lftp iproute vim

[root@docker01 oo]# docker build -t testos:001 .

Dockerfile02

[root@docker01 xx]# vim Dockerfile

FROM myos00:latest

CMD ["/usr/bin/python"]

Dockerfile03

[root@docker01 httpos]# vim Dockerfile

FROM myos00

RUN yum install -y httpd

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

WORKDIR /var/www/html

RUN echo "Hello World!" > index.html

MAINTAINER yyzh

ENV EnvironmentFile=/etc/sysconfig/httpd

EXPOSE 80

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

[root@docker01 xx]# docker build -t myos00 .

[root@docker01 xx]# docker run -it 3f6aa775d919

1. RHEL和Debian之间的一些区别

yum apt-get

rpm deb

管理

rpm dpkg

1. 自定义镜像仓库
2. registry基本概念

· 共享镜像的一台服务器(镜像化的一台服务器)



1. 自定义私有仓库

· 流程：

1. 获取registry镜像

[root@docker01 ~]# docker pull registry

1. 修改配置文件，创建私有仓库

[root@docker01 ~]# vim /etc/docker/daemon.json

{

"insecure-registries" : ["192.168.1.130:5000"]

}

1. 重启服务，配置生效

[root@docker01 ~]# systemctl restart docker

1. 启动私有仓库服务

[root@docker01 ~]# docker run -d -p 5000:5000 registry

1. 打标记

[root@docker01 ~]# docker tag busybox:latest 192.168.1.130:5000/busybox:latest

[root@docker01 ~]# docker tag testos:001 192.168.1.130:5000/testos:001

[root@docker01 ~]# docker tag testos:http 192.168.1.130:5000/testos:http

1. 上传镜像

[root@docker01 ~]# docker push 192.168.1.130:5000/busybox:latest

[root@docker01 ~]# docker push 192.168.1.130:5000/testos:001

[root@docker01 ~]# docker push 192.168.1.130:5000/testos:http

1. 客户端测试

[root@docker02 ~]# vim /etc/docker/daemon.json

{

"insecure-registries":["192.168.1.130:5000"]

}

[root@docker02 ~]# systemctl restart docker

[root@docker02 ~]# docker run -d 192.168.1.130/testos:http

[root@docker02 ~]# curl http://172.17.0.2

Hello World!

1. 查看私有仓库中的镜像

浏览器打开：

192.168.1.130:5000/v2/\_catalog

1. 查看镜像的标签

浏览器打开：

192.168.1.130:5000/v2/myos/tags/list

1. 第二种搭建方式

· 安装私有仓库( 服务端 )

- yum install docker-distribution

· 启动私有仓库, 并设置开机启动

- systemctl start docker-distribution

- systemctl enable docker-distribution

· 仓库配置文件及数据存储路径

- /etc/docker-distribution/registry/config.yml

- /var/lib/regitstry

· 客户端配置

- 修改配置文件 /etc/sysconfig/docker

- 允许非加密方式访问仓库

INSECURE\_REGISTRY='--insecure-registry docker1:5000'

- docker仓库地址

ADD\_REGISTRY='--add-registry docker1:5000'

· 重启docker服务

- systemctl restart docker

· 为镜像创建标签

- 这里地址要写宿主机的IP或主机名

- docker tag 镜像:标签 IP:5000/镜像:latest

· 上传镜像

- 上传镜像的标签内包含地址和端口号

- dockerpush IP:5000/镜像:latest

· 远程启动容器( docker2 )

- 配置 /etc/sysconfig/docekr

INSECURE\_REGISTRY='--insecure-registry docker1:5000'

ADD\_REGISTRY='--add-registry docker1:5000'

· 重启docker

- 重启docker服务

- systemctl restart docker

· 远程启动镜像

- docker run -it [ docker1:5000 ]/myos:latest

· 查看私有镜像仓库中的镜像名称

- curl <http://docker1:5000/v2/_catalog>

· 查看某一仓库的标签

- curl [http://docker1:5000/v2/<repo>/tags/list](http://docker1:5000/v2/<repo>/list/tags)

· 进入registry的配置文件, 进入容器查看

- /etc/docker/registry/config.yml

1. 持久化存储
2. 存储卷
3. 卷的概念
4. docker容器不保持任何数据
5. 重要数据请使用外部卷存储(数据持久化)
6. 容器可以挂载真实机目录或共享存储为卷
7. 主机卷的映射

· 将真实机目录挂载到容器中提供持久化存储

[root@docker02 ~]# docker run -d -v /var/webroot/:/var/www/html 192.168.1.130:5000/testos:http

1. 共享存储
2. 共享存储基本概念
3. 一台共享存储服务器可以提供给所有Docker主机使用
4. 共享存储服务器(NAS、SAN、DAS等)
5. 原理：

· 使用NFS创建共享存储服务器

· 客户端挂载NFS共享，并最终映射到容器中

1. 使用共享存储的案例

· 服务器

[root@docker02 ~]# vim /etc/exports

/var/webroot \*(rw)

[root@docker02 ~]# systemctl start nfs

· Docker主机

1. mount挂载共享

[root@docker01 ~]# yum -y install nfs-utils.x86\_64

[root@docker01 ~]# mount -t nfs 192.168.1.129:/var/webroot/ /var/webroot/

1. 运行容器时，使用-v选项映射磁盘到容器中

[root@docker01 ~]# docker run -it -v /var/webroot:/var/www/html testos:http

1. Docker网络架构
2. Linux网桥
3. 创建虚拟网卡

· 真实网卡配置文件

[root@docker01 ~]# cat network/ifcfg-ens33

· 虚拟网卡配置文件

[root@docker01 ~]# cat network/ifcfg-br0

1. 创建虚拟网桥

· 物理机

[root@docker01 network]# cp ifcfg-{ens33,br0}

[root@docker01 network]# vim ifcfg-br0

TYPE=Bridge

DEVICE=br0

ONBOOT=yes

BOOTPROTO=static

IPADDR=192.168.2.15

PREFIX=24

[root@docker01 network]# ifup br0

1. Docker网络拓扑
2. 查看Docker默认网络模型

[root@docker01 ~]# docker network list

[root@docker01 ~]# docker network

//查看帮助

[root@docker01 ~]# docker network create --help

1. 使用Docker创建网桥

[root@docker01 ~]# docker network create --driver bridge --subnet 192.168.100.0/24 docker1

[root@docker01 ~]# docker network ls

5c5b19232e6f docker1 bridge local

[root@docker01 ~]# ifconfig br-5c5b19232e6f | head -2

br-5c5b19232e6f: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 192.168.100.1 netmask 255.255.255.0 broadcast 0.0.0.0

1. 使用自定义网桥

[root@docker01 ~]# docker run -it --network docker1 testos

[root@docker01 ~]# docker run -it --network docker1 testos:http

/\*

不同交换机里的容器不能互相访问

原因：docker在iptables里写入了规则，保持了隔离性

# iptables -F则可访问

\*/

[root@docker01 ~]# curl http://192.168.100.2

curl: (7) Failed connect to 192.168.100.2:80; Connection refused

[root@105222df1edc html]# /sbin/httpd

[root@docker01 ~]# curl http://192.168.100.2

Hello World!

1. 客户端访问容器内的资源

· 默认容器通过SNAT可以访问外网

· 但外部网络的主机不可以访问容器内的资源

· 端口映射

- 使用端口映射可以实现外部网络访问容器内的资源

docker run -p 物理机端口:虚拟机端口

[root@docker01 ~]# docker run -d -v /var/webroot/:/var/www/html -p 80:80 testos:http

[root@docker02 ~]# curl http://192.168.1.130

Hello!This is docker1!

1. ARCHITECTURE
2. Day 01 ansible概述、ansible基础 、 ad-hoc、批量配置管理
   1. absible基础
      1. ansible简介
3. 什么是ansible

· ansible是2013年推出的一款IT自动化和DevOps软件,2015年被RedHat收购,是基于Python研发,糅合很多老运维工具的优点,实现了批量操作系统配置,批量程序部署,批量运行命令等功能

· ansible可以实现:

- 自动化部署APP

- 自动化管理配置项

- 自动化持续交付

- 自动化(AWS)云服务管理

1. 为什么选择ansible

· 选择一款配置管理软件,无外乎从以下几点来权衡利弊

- 活跃度(社区)

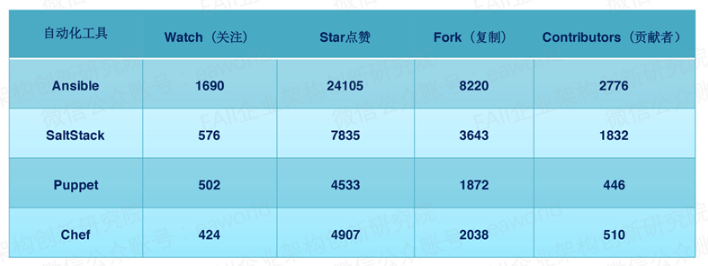
- 学习成本

- 使用成本

- 编码语言

- 性能

- 使用是否广泛



· ansible优点

- 只需要SSH和Python即可使用

- 无客户端

- ansible功能强大,模块丰富

- 上手容易,门槛低

- 基于Python开发,做二次开发更容易

- 使用公司较多,社区活跃

· ansible问题

- 对于几千台、上万台机器的操作,还不清楚性能,效率情况如何,需要进一步了解。

1. ansible特性

· 模块化设计,调用特定的模块完成特定的任务

· 基于Python语言实现

- paramiko

- PyYAML(半结构化语言)

- Jinja2

· 其模块支持JSON等标准输出格式,可以采用任何编程原因重写

· 部署简单

· 主从模式工作

· 支持自定义模块

· 支持playbook

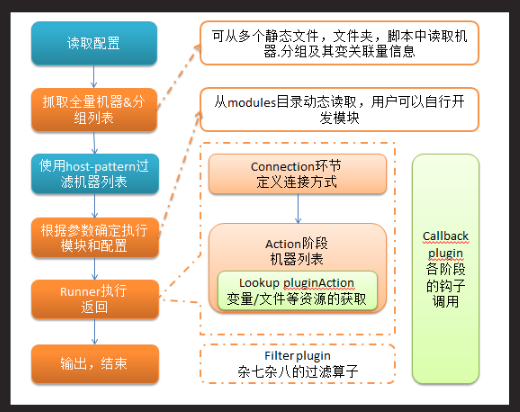
· 易于使用

· 支持多层部署

· 支持异构IT环境

1. 工作流程

· ansible大体执行过程



* + 1. ansible安装

1. 软件依赖关系

· 对管理主机

- 要求Python2.6或Python2.7

· ansible使用以下模块,都需要安装

- paramiko

- PyYAML

- Jinja2

- httplib2

- six

· 对于被托管主机

- ansible默认通过SSH协议管理机器

- 被管理主机要开启ssh服务,允许ansible主机登录

- 在托管节点上也需要安装Python2.5或以上版本

- 如果托管节点上开启了SELinux,需要安装libselinux-python

1. 安装ansible

· ansible可以基于源码运行

· 源码安装

- pip,需要配置扩展软件包源extras

- git

yum install epel-release

yum install git python2-pip

- pip安装依赖模块

pip install paramiko PyYAML Jinja2 httplib2 six

· ansible源码下载

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

- yum install python-setuptools python-devel

- python setup.py build

- python setup.py install

· pip方式安装(Ubuntu等没有yum的系统)

- pip install ansible

· yum扩展源安装简单,自动解决以来关系(推荐)

- <http://mirror.centos.org/.../.../extras/>

- yum install ansible

[root@vm1 ansible\_soft]# ls

ansible-2.4.2.0-2.el7.noarch.rpm

python2-jmespath-0.9.0-3.el7.noarch.rpm

python-httplib2-0.9.2-1.el7.noarch.rpm

python-paramiko-2.1.1-4.el7.noarch.rpm

python-passlib-1.6.5-2.el7.noarch.rpm

sshpass-1.06-2.el7.x86\_64.rpm

· 安装完成后验证

- ansible --version

1. ansible环境部署

· 安装六台虚拟机

|  |  |  |
| --- | --- | --- |
| ansible | 192.168.1.10 | 管理主机 |
| web1 | 192.168.1.11 | 托管机器 |
| web2 | 192.168.1.12 | 托管机器 |
| db1 | 192.168.1.21 | 托管机器 |
| db2 | 192.168.1.22 | 托管机器 |
| cache | 192.168.1.33 | 托管机器 |

ansible : 主机上配置ip与主机名对应关系

* 1. ad-hoc(执行临时操作)
     1. 主机管理

1. 主机定义与分组

· 安装ansible之后可以做一些简单的任务

· ansible配置文件查找顺序

- 首先检测ANSIBLE\_CONFIG变量定义的配置文件

- 其次检查当前目录下的./ansible.cfg文件

- 再次检查当前用户家目录下的~/ansible.cfg文件

- 最后检查 /etc/ansible/ansible.cfg文件

· /etc/ansible/ansible.cfg是ansible的默认配置文件路径

· ansible.cfg配置文件

- inventory定义托管主机地址配置文件

- 先编辑 /etc/ansible/hosts文件,写入远程主机的地址

· 格式

- #表示注释

[组名称]

主机名称或ip地址,登录用户名,密码,端口等信息

· 测试

- ansible [组名称] --list-hosts

· inventory参数说明

- ansible\_ssh\_host

- 将要连接的远程主机名与你想要设定的主机别名不同,可以通过此变量设置

- ansible\_ssh\_port

- ssh端口号:如果不是默认的端口号,通过此变量设置

- ansible\_ssh\_user

- 默认的ssh用户名

- ansible\_ssh\_pass

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

- ansible\_sudo\_pass

- sudo密码(建议使用--ask-sudo-pass)

- ansible\_sudo\_exe( new in version 1.8 )

- sudo命令路径(适用于1.8及以上版本)

- ansible\_connection

- 与主机的连接类型,如:local,ssh或paramiko,1.2以前默认使用paramiko,1.2以后默认使用'smart',它会根据是否支持ControlPersist来判断'ssh'方式是否可行

- ansible\_ssh\_private\_key\_file

- ssh使用的私钥文件,适用于有多个密钥,而你不想使用SSH代理的情况

- ansible\_shell\_type

- 目标系统的;shell类型,默认情况下,命令的执行适用'sh'语法,可设置为'csh'或'fish'

- ansible\_python\_interpreter

- 目标主机的python路径,使用情况:系统中有多个Python,或者命令路径不是"/usr/bin/python"

· 分组定义,范围定义样例

[web]

web1

web2

[db]

db[1:2]

[cache]

192.168.1.16

[app:children]

web

db

[web]

web[1:2]

[web:vars]

ansible\_ssh\_user="root"

ansible\_ssh\_pass="pwd"

ansible\_ssh\_port="22"

//针对一组主机配置

[cache]

c01 ansible\_ssh\_user="root" ansible\_ssh\_pass="pwd"

· 定义并测试测试分组

[root@ansible ~]# vim /etc/ansible/hosts

[web]

web[1:2]

[db]

db[1:2]

[other]

cache

[root@ansible ~]# ansible web --list-host

hosts (2):

web1

web2

[root@ansible ~]# ansible db --list-host

hosts (2):

db1

db2

· 测试连通性

[root@ansible ~]# ansible web -m ping

web2 | SUCCESS => {

"changed": false,

"ping": "pong"

}

web1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

· 删除cache主机上的公钥,进行连通测试

[root@ansible ~]# ansible other -m ping

cache | UNREACHABLE! => {

"changed": false,

"msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).\r\n",

"unreachable": true

}

PS:此时可以通过在/etc/ansible/hosts后面加上对应的用户和密码

[cache]

c01 ansible\_ssh\_user="root" ansible\_ssh\_pass="pwd"

· 自定义配置文件

- 创建文件夹myansible

- 创建配置文件ansible.cfg

[defaults]

inventory = myhost

- 配置主机文件

[nginx]

192.168.1.11

192.168.1.12

192.168.1.13

- ansible nginx --list-hosts

· 自定义配置实例

- 创建自定义目录

[root@ansible ~]# mkdir xxoo

- 自定义配置

[root@ansible xxoo]# vim ansible.cfg

[defaults]

inventory = myhost

[root@ansible xxoo]# vim myhost

[app1]

web1

db1

[app2]

web2

db2

cache

[app:children]

app1

app2

- 测试自定义配置

[root@ansible xxoo]# ansible app1 -m ping

web1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

db1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

1. 动态主机

· 无限可能

- ansible Inventory包含静态和动态的Inventory,静态Inventory指在文件/etc/ansible/hosts中制定的主机和组,动态Inventory指通过外部脚本获取主机列表,按照其要求格式返回给ansible命令

· json

- JSON( JavaScript Object Notation, JavaScript对象表示法 ),一种基于文本,独立语言的轻量级数据交换格式

· 注意事项

- 主机部分必须是列表格式

- Hostdata行,其中的"hosts"部分可以省略,但使用时,必须是"hosts"

· 脚本输出主机列表

#!/usr/bin/python

import json

hostlist = {}

hostlist["bb"] = ["192.168.1.15","192.168.1.16"]

hostlist["192.168.1.13"] = {

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

}

hostlist["aa"] = {

"hosts":["192.168.1.11","192.168.1.12"],

"vars":{

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

}

}

print(json.dumps(hostlist))

· 脚本输出样例

{

"web" : ["web1","web2"],

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

"other" : {

"hosts" : ["cache"],

"vars" : {

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

"ansible\_ssh\_pass" : "a"

}

}

}

* + 1. 批量执行

1. ansible命令基础

· ansible <host-p[attern> [options]

- host-pattern 主机或定义的分组

- -M 制定模块路径

- -m 使用模块,默认command模块

- -a or --args模块参数

- -i inventory文件路径或可执行脚本

- -k 使用交互式登录密码

- -e 定义变量

- -v 详细信息,-vvvv开启debug模式

· 列出要执行的主机

- ansible all --list-hosts

· 批量检测主机

- ansible all -m ping

· 批量执行命令

- ansible all -m command -a 'id' -k

1. 批量部署证书文件

· 每次交互输入密码太麻烦

· 密码写入配置文件安全性太差

· 不同主机密码不一定相同

- 使用key方式认证

· 给所有主机部署公钥

# ansible all -m authorized\_key -a "user=root exclusive=true manage\_dir=true key='$(< /root/.ssh/authorized\_keys)'" -k [-v]

· 遇到的报错:

1. manage\_dir写成了manage-dir

报错:

web2 | FAILED! => {

"changed": false,

"msg": "Unsupported parameters for (authorized\_key) module: manage-dir Supported parameters include: comment,exclusive,key,key\_options,manage\_dir,path,state,unique,user,validate\_certs"

}

1. 没有加入到known\_hosts中导致ssh时需要回答yes/no

web2 | FAILED! => {

"changed": false,

"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known\_hosts file to manage this host."

}

· 解决方法

- 在ansible配置文件(ansible.cfg)中设置host\_key\_checking = False

* 1. 批量配置管理
     1. 模块

1. ansible-doc和ping模块

· ansible-doc

- 模块的手册,相当于shell的man,非常重要

- ansible-doc -l 列出所有模块

- ansible-doc moduleName 查看帮助

· ping模块

- 测试网络连通性,ping模块没有参数

- 注:测试ssh的连通性

# ansible host-pattern -m ping

1. command模块

· command模块

- 默认模块,远程执行命令

- 用法

# ansible host-pattern -m command -a '[args]'

- 查看所有机器负载

# ansible all -m command -a 'uptime'

- 查看日期和时间

# ansible all -m command -a 'date +%F\_%T'

· command模块注意事项:

- 该模块通过-a跟上要执行的命令可以直接执行,若命令里有如下字符则执行不成功

"<" , ">" , "|" , "&"

- 该模块不启动shell直接在ssh进程中执行,所有使用到shell的命令执行都会失败

- 下列命令执行会失败

# ansible all -m command -a 'ps aux | grep ssh'

# ansible all -m command -a 'set'

1. shell | raw模块

· shell | raw模块

- shell模块用法基本和command一样,区别是shell模块通过/bin/sh执行命令,可以执行任意命令

- raw模块,用法和shell一样,可以执行任意命令

- 区别是raw没有chdir,creates,removes参数

- 执行以下命令查看结果

# ansible t1 -m command -a 'chdir=/tmp touch f1'

# ansible t1 -m shell -a 'chdir=/tmp touch f2'

# ansible t1 -m raw -a 'chdir=/tmp touch f3'

命令1,2有Warning警告,文件f1,f2创建成功,命令3没有警告,显示Success,但是文件f3没有创建

1. script模块

· script模块

- 命令太复杂?

- 在本地写脚本,然后使用script模块批量执行

# ansible t1 -m script -a 'urscript'

- 注意 : 该脚本包含但不限于shell脚本,只要制定Sha-bang解释器的脚本都可运行

1. copy模块

· copy模块

- 复制文件到远程主机

- src : 复制远程主机的文件到本地,绝对路径和相对路径都可,路径为目录时会递归复制。若路径以"/"结尾,只复制目录里的内容,若不以"/"结尾,则复制包含目录在内的整个内容,类似于rsync

- dest : 必选项。远程主机的绝对路径,如果源文件是一个目录,那该路径必须是目录

- backup : 覆盖前先备份源文件,备份文件包含时间信息。有两个选项:yes/no

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

- 复制文件

# ansible t1 -m copy -a 'src=/root/alog dest=/root/a.log'

- 复制目录

# ansible t1 -m copy -a 'src=urdir dest=/root/'

1. lineinfile | replace模块

· lineinfile | replace模块

- 类似sed的一种行编辑替换模块

- path目的文件

- regexp正则表达式

- line替换后的结果

# ansible t1 -n lineinfile -a 'path="/etc/selinux/config" regexp="^SELINUX=" line="SELINUX=disabled"'

- 替换指定字符

# ansible t1 -m replace -a 'path="/etc/selinux/config" regexp="^(SELINUX=).\*" replace="\1disabled"'

1. yum模块

· yum模块

- 使用yum包管理器来管理软件包

- config\_file : yum的配置文件

- disable\_gpg\_check : 关闭gpg\_check

- disablerepo : 不启用某个源

- enablerepo : 启用某个源

- name : 要进行操作的软件包名字 ,也可传递一个url或一个本地的rpm包的路径

- state : 状态( present ,absent ,latest )

- 删除软件包

# ansible t1 -m yum -a 'name="lrzsz" state=absent'

- 删除多个软件包

# ansible t1 -m yum -a 'name="lrzsz,lftp" state=absent'

- 安装软件包

# ansible t1 -m yum -a 'name="lrzsz"'

- 安装多个软件包

# ansible t1 -m yum -a 'name="lrzsz,lftp"'

1. service模块

· service模块

- name : 必选项,服务名称

- enable : 是否开机启动yes/no

- sleep : 执行restarted,会在stop和start之间沉睡几秒钟

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

# ansible t1 -m service -a 'name="sshd" enabled="yes" state="started"'

1. srtup模块

· setup

- 主要用于获取主机信息,playbooks里经常会用的另一个参数gather\_facts与该莫苦熬相关,setup模块下经常用的是filter参数

-filter过滤所需信息

# ansible t1 -m setup -a 'filter=ansible\_distribution'

1. ansible综合小实验

· 安装Apache并修改监听端口为8080

[root@ansible ~]# ansible web -m yum -a 'name="httpd"'

[root@ansible ~]# ansible web -m lineinfile -a 'path="/etc/httpd/conf/httpd.conf" regexp="^Listen" line="Listen 8080"'

· 修改ServerName配置,执行apachectl -t命令不报错

· 设置默认主页hello world

· 启动服务并设置开机自启动

[root@ansible ~]# ansible web -m service -a 'name="httpd" enabled=yes state=started'

1. Day 02 playbook基础、playbook进阶
   1. Playbook基础
      1. 基础知识
2. ansible七种武器

· 第一种武器

- ansible命令,用于执行临时性的工作,必须掌握

· 第二种武器

- ansible-doc是ansible模块的文档说明,针对每个模块都有详细的说明及应用案例介绍,功能和Linux系统man命令类似,必须掌握

· 第三种武器

- ansible-console是ansible为用户提供的交互式工具,用户可以在ansible-console虚拟出来的终端上像shell一样使用ansible内置的各种命令,这为习惯使用shell交互方式的用户提供了良好的使用体验

· 第四种武器

- ansible-galaxy从github上下载管理Roles的一款工具,与python的pip类似

· 第五种武器

- ansible-playbook是日常应用中使用频率最高的命令,工作机制 : 通过读取先编写好的playbook文件实现批量管理,可以理解为按一定条件组成的ansible任务集,必须掌握

· 第六种武器

- ansible-vault主要用于配置文件加密,如编写的playbook文件中包含敏感信息,不想其他人随意查看,可以用它加密/解密这个文件

· 第七种武器

- ansible-pull

- ansible有两周工作模式pull/push,默认使用push模式工作,pull和push工作模式机制刚好相反

- 适用场景 : 有大批机器需要配置,即便使用高并发线程依旧要花费很多时间

- 通常在配置大批量机器的场景下使用,灵活性稍有欠缺,但效率几乎可以无限提升,对运维人员的技术水平和前瞻性规划有较高要求

1. JSON简介

· JSON是什么

- JSON是JavaScript对象表示法,它是一种基于文本独立于语言的轻量级数据交换格式

- JSON中搞得分隔符限于单引号、小括号、中括号、大括号、冒号和逗号

· JSON特性

- JSON是纯文本

- JSON具有"自我描述性"( 人类可读 )

- JSON具有层级结构( 值中存在值 )

- JSON可通过JavaScript进行解析

· JSON语法规则

- 数据在名称对/值对中

- 数据由逗号分隔

- 大括号保存对象

- 中括号保存数组

· JSON数据的书写格式是 : 名称/值对

- 名称/值对包括字段名称( 在双引号中 ), 后面写一个冒号, 然后是值,例如 :

"诗仙" : "李白"

· JSON语法规则之数组

{

"诗人":

["李白", "杜甫", "白居易", "李贺"]

}

· 复合复杂型

{

"诗人":

[

{"李白":"诗仙","年代":"唐"},

{"杜甫":"诗圣","年代":"唐"},

{"白居易":"诗魔","年代":"唐"},

{"李贺":"诗鬼","年代":"唐"}

]

}

1. YAML简介

· YAML是什么

- 是一个可读性高, 用来表达数据序列的格式

- YAML( YAML Ain't Markup Language )

- YAML参考了多种语言,如:C,Pyhton,Perl等, 并从XML、电子邮件的数据格式中获得灵感, Clark Evans在2001年首次发表了这种语言, 目前已有数种编程语言或脚本语言支持这种语言

· YAML基础语法

- YAML的结构通过空格来展示

- 数组使用"-"来表示

- 键值对使用":"来表示

- YAML使用一个固定的缩进风格( 空格对齐 )表示数据层级结构关系

- 一般每个缩进级别由两个以上空格组成

- # 表示注释

· 注意 :

- 不要使用tab,缩进是初学者容易出错的地方之一

- 同一层级缩进必须对齐

· YAML的键值表示方法

- 采用冒号分割

- :后面必须有一个空格

- YAML键值对例子

"诗仙": "李白"

- 或

"李白":

"诗仙"

- 复杂YAML的键值对嵌套

"诗人":

"李白": "诗仙"

- 或

"诗人":

"李白":

"诗仙"

- 数组

["李白", "杜甫", "白居易", "李贺"]

· YAML数组表示方法

- 使用一个短横杠加一个空格

- YAML数组例子

- "李白"

- "杜甫"

- "白居易"

- "李贺"

- 哈希数组符合表达式

"诗人":

- "李白"

- "杜甫"

- "白居易"

- "李贺"

- 高级复合表达式

"诗人":

-

"李白": "诗仙"

"唐代": "唐"

-

"杜甫": "诗圣"

"唐代": "唐"

-

"白居易": "诗魔"

"唐代": "唐"

-

"李贺": "诗鬼"

"唐代": "唐"

· yaml高级语法

- | 与 > 表示对应的值为多行字符, > 与 | 的区别是会把\n转化为空格

- ! 可以设置类型, !! 可以强制类型转换

- 为了维持文件的简洁,并且避免数据输入的错误, YAML提供了节点参考( \* )和散列合并( << )参考到其他节点标签的锚点标记( & )。参考会将树状结构加入锚点标记的内容,并可以在所有数据结构中运作, 合并只有散列表可以使用, 可以将键值自锚点标记复制到指定的散列表中

1. Jinja2模板简介

· jinja2是什么

- jinja2是基于Python的模板引擎, 包含变量和表达式两部分, 两者在模板求值时会被替换为值, 模板中还有标签, 控制模板的逻辑

· 为什么要学习jinja2模板

- 因为playbook的模板使用Python的jinja2模块来处理

· jinja2模板基本语法

- 模板的表达式都是包含在分隔符"{{ }}"内的

- 控制语句都是包含在分隔符"{% %}"内的

- 模板支持注释, 都是包含在分隔符"{# #}"内,支持块注释

- 调用变量

{{varname}}

- 计算

{{2+3}}

- 判断

{{1 in [1,2,3]}}

· jinja2模板控制语句

{% if name == '诗仙' %}

李白

{% elif name == '诗圣' %}

杜甫

{% elif name == '诗魔' %}

白居易

{% else %}

李贺

{% endif %}

· jinja2模板控制语句

{% if name == ... ... %}

... ...

{% elif name == '于谦' %}

{% for method in [抽烟, 喝酒, 烫头] %}

{{do method}}

{% endfor %}

... ...

{% endif %}

· Jinja2过滤器

- 变量可以通过过滤器修改。过滤器与变量用管道符号( | )分隔, 也可以用圆括号传递可选参数, 多个过滤器可以链式调用, 前一个过滤器的输出会被作为后一个过滤器的输入

- 例如

- 把一个列表用逗号连接起来 : {{ list | join(',') }}

- 过滤器这里不再一一列举, 可以查询在线文档

<http://docs.jinkan.org/docs/jinja2/templates.html>

#builtin-filters

* + 1. playbook

1. playbook是什么

· playbook是什么

- playbook是ansible用于配置,部署和管理托管主机剧本, 通过playbook的详细描述, 执行其中的一系列tasks, 可以让远端主机达到预期状态

- playbook按照字面意思为"剧本",在ansible中计算机按照playbook进行安装、部署应用, 提供对外服务, 以及组织计算机处理各种各样的事情

· 为什么要使用playbook

1. playbook语法基础

· playbook语法格式

- playbook由YAML语言编写, 遵循YAML标准

- 在同一行中, #之后的内容表示注释

- 同一个列表中的元素应该保持相同的缩进

- playbook由一个或多个play组成

- play中hosts, variables, roles, tasks等对象的表示方法都是键值中间以":"分隔表示

- YAML还有一个小的怪癖, 它的文件开始行都应该是---, 这是YAML格式的一部分, 表明一个文件的开始

· playbook构成

- Target : 定义将要执行playbook的远程主机组

- Variable : 定义playbook运行时需要使用的变量

- Tasks : 定义将要在远程主机上执行的任务列表

- Handler : 定义task执行完成以后需要调用的任务

· playbook执行结果

· 使用ansible-playbook运行playbook文件, 输出内容为JSON格式, 由不同颜色组成便于识别

- 绿色代表执行成功

- \*\*\*代表系统状态发生改变

- 红色代表执行失败

· 第一个playbook

--- #第一行, 表示开始

- hosts: all

remote\_user: root

tasks:

- ping:

# ansible-playbook myping.yml -f 5

- -f 并发进程数量,默认是5

- hosts行 内容更是一个( 多个 )组或主机的patterns, 以逗号为分隔符

- remote\_user 账户名

· tasks

- 每一个play包含了一个task列表( 任务列表 )

- 一个task在其所对应的所有主机上( 通过 host pattern匹配的所有主机 )执行完毕后, 下一个task才会执行

- 有一点很重要, 在一个play值中, 所有hosts会回去相同的任务指令, 这是play的一个目的所在, 即将一组选出的hosts映射到tasks, 执行相同的操作

· playbook执行命令

- 给所有主机添加用户plj, 设置默认密码123456

- 要求第一次登录修改密码

---

- hosts: all

remote\_user: root

tasks:

- name: create user plj

user: group=wheel uid=1000 name=plj

- shell: echo 123456 | passwd --stdin plj

- shell: chage -d 0 plj

· playbook实验

- 使用playbook

- 使用yum模块为web群主机安装apache httpd

- 修改httpd服务端口为8080

- 给httpd配置ServerName

- 设置开机自启动

[root@ansible ~]# vim apache.yml

---

- hosts: web

remote\_user: root

tasks:

- name: install httpd

yum:

name: httpd

state: installed

- name: change configures

lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen'

line: 'Listen 8080'

- lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^#ServerName'

line: 'ServerName www.myweb.com'

- name: create index

shell:

echo 'Hello World!' > /var/www/html/index.html

- name: start service

service:

name: httpd

enabled: yes

state: started

PS :多查ansible-doc

* 1. Playbook进阶
     1. 语法进阶

1. 变量

· 添加用户

- 给所有主机添加用户plj,设置默认密码123456

- 要求第一次登录修改密码( 使用变量 )

---

- hosts: 192.168.1.16

remote \_user: root

vars:

username: plj

tasks:

- name: create user "{{username}}"

user: group=wheel uid=1000 name={{username}}

- shell: echo 123456 | passwd --stdin plj

- shell: chage -d 0 {{username}}

· 设密码

- 解决密码明文问题

- user模块的password为什么不能设置密码

因为password是把字符串直接写入shadow, 并没有改变, 而Linux的shadow密码是经过加密的, 所以不能使用

· 解决方案

- 变量过滤器password\_hash

{{'urpassword' | password\_hash('sha512')}}

· 变量过滤器

- 给所有主机添加用户plj, 设置默认密码123456

- 要求第一次登录修改密码( 使用变量 )

---

- hosts: 192.168.1.16

remote\_user: root

vars:

username: plj

tasks:

- name: create user "{{username}}"

user: group=wheel uid=1000 password="{{'123456' | password\_hash('sha512')}} name={{username}}"

- shell: chage -d 0 {{username}}

1. error

· ansible-playbook对错误处理

- 默认情况判断$?, 如果值不为0就停止执行

- 但某些情况我们需要忽略错误继续执行

---

- hosts: 192.168.1.16

remote\_user: root

vars:

username: plj

tasks:

- name: create user "{{username}}"

user: group=wheel uid=1000

password={{'123456' | password\_hash('sha512')}}

name={{username}}

- shell: setenforce 0

- shell: chage -d 0 {{username}}

· 错误处理方法

- 关闭selinux, 如果selinux已经关闭, 返回1, 若之前已经关闭则不算错误, 可以忽略错误继续运行, 忽略错误有两种方法

- 第一种方式

shell: /usr/bin/somecommand || /bin/true

- 第二种方式

- name: run some command

shell: /usr/bin/somecommand

ignore\_errors: True

1. handlers

· 当关注的资源发生变化时采取的操作

· notify这个action可用于在每个play的最后被触发, 这样可以避免有多次改变发生时每次次都执行指定的操作, 取而代之仅在所有的变化发生完成后一次性地执行指定操作

· notify中列出的操作成为handler, 即notify调用handler中定义的操作

· 前面安装了Apache, 修改httpd的配置文件, 重新载入配置文件让服务生效

· 使用handlers来实现

handlers:

- name: restart apache

service:

name: httpd

state: restarted

enabled: yes

· 注意事项:

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

- 多个task触发同一个notify的时候, 同一个服务只会触发一次

- notify可以触发多个条件, 在生产环境中往往涉及到某一个配置文件的改变要重启若干服务的场景, handler用在这里非常合适

- 结合vars可以写出具有普遍适应性的服务器管理脚本

---

- hosts: web

remote\_user: root

tasks:

- name: install httpd

yum:

name: httpd

state: installed

- name: change configures

lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen'

line: 'Listen 8080'

notify:

- restart\_httpd

- lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^#ServerName'

line: 'ServerName www.myweb.com'

notify:

- restart\_httpd

- name: create index

shell:

echo 'Hello World!' > /var/www/html/index.html

handlers:

- name: restart\_httpd

service:

name: httpd

enabled: yes

state: started

1. when

· 某些时候需要在满足特定的条件后再触发某一项操作, 或在特定的条件下终止某个行为, 这个时候需要进行条件判断, when正式解决这个问题的最佳选项, 远程中的系统变量facts作为when的条件, 可以通过setup模块查看

· when的样例

---

- name: install VIM

hosts: all

tasks:

- name:Install VIM via yum

yum: name=vim-enhanced state=instead

when: ansible\_os\_family == "Redhat"

- name: Install VIM via apt

apt: name=vim state=installed

when: ansible\_os\_family == "Debian"

1. register

· register

- 有时候我们还需要更复杂的例子, 如判断前一个命令的执行结果去处理后面的操作, 这时候就需要在register模块来保存前一个命令的返回状态, 在后面进行调用

- command: test command

register: result

- command: run command

when: result

· 变量注册

- 例如需要判断plj这个用户是否存在

- 如果存在就修改密码, 如果不存在就跳过

tasks:

- shell: id {{username}}

registry: result

- name: change "{{username}}" password

user: password={{'12345678' | password\_hash('sha512')}} name={{username}}

when: result

· 变量注册进阶

- 针对运行命令结果的返回值做判定

- 当系统负载超过一定值的时候做特殊处理

---

- hosts: 192.168.1.16

remote\_user: root

tasks:

- shell: uptime | awk '{printf("%f\n",$(NF-2))}'

register: result

- service: name=httpd state=stopped

when: result.stdout | float > 0.7

1. with\_items

· with\_items是playbook标准循环, 可以用于迭代一个列表或字典, 通过{{item}}获取每次迭代的值

- 例如创建多个用户

---

- hosts: cache

remote\_user: root

tasks:

- name: add users

user: group=wheel password={{'123456' | password\_hash('sha512')}} name={{item}}

with\_items: ["nb","dd","plj","lx"]

· with\_items进阶

- 为不同用户定义不同组

---

- hosts: cache

remote\_user: root

tasks:

- name: add users

user:

name: '{{item.name}}'

group: '{{item.group}}'

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

with\_items:

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

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

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

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

1. with\_nested

· 嵌套循环

---

- hosts: cache

remote\_user: root

vars:

un: [a,b,c]

id: [1,2,3]

tasks:

- name: add users

shell: echo {{item}}

with\_nested:

- "{{un}}"

- "{{id}}"

1. tags

· tags :给指定的任务定义一个调用标识

· 使用格式:

- name: NAME

module: arguments

tags: TAG\_ID

· playbook调用方式

- -t TAGS, --tags=TAGS

- --skip-tags=SKIP\_TAGS

- --start-at-task=START\_AT

· tags样例:

vars:

soft: httpd

tasks:

- name: install {{soft}}

yum: name={{soft}}

- name: config httpd.conf

copy: src=/root/playbook/httpd.conf

dest=/etc/httpd/conf/httpd.conf

- name: config services

service: enabled=yes state=restarted name={{soft}}

tags: restartweb

· 调用方式

ansible-playbook i.yml --tags=restartweb

1. include and roles

· 在编写playbook的时候随着项目越来越大, playbook越来越复杂, 修改也很麻烦。这时可以把一些play、task或handler放到其他文件中, 通过include指令包含进来

---

...

tasks:

- include: tasks/setup.yml

- include: tasks/users.yml user=plj #users.yml可以通过{{user}}来使>用这些变量

handlers:

- include: handlers/handlers.yml

· roles像是加强版include, 它可以引入一个项目的文件和目录

- 一般所需的目录层级有

- vars: 变量层

- tasks: 任务层

- handlers: 触发条件

- files: 文件

- template: 模板

- default: 默认, 优先级最低

· 假如有一个play包含了一个叫"x"的role, 则

---

- hosts: host\_group

roles:

-x

- x/tasks/main.yml

- x/vars/main.yml

- x/handler/main.yml

- x/... .../main.yml

- 都会自动添加进这个play

* + 1. 调试

1. debug

· 对于Python语法不熟悉的同学, playbook书写起来容易出错, 且排错困难, 这里介绍几种简单的排错调试方法

- 检测语法

# ansible-playbook --syntax-check playbook.yaml

- 测试运行

# ansible-playbook -C playbook.yaml

- 显示受到影响的主机 --list-hosts

- 显示工作的tasks --list-tasks

- 显示将要运行的tag --list-tags

· debug模块可以在运行时输出更为详细的信息, 帮助我们排错

· debug使用样例

---

- hosts: cache

remote\_user: root

tasks:

- shell: uptime | awk '{printf("%f\n",$(NF-2))}'

register:result

- shell: touch/tmp/isreboot

when:result.stdout | float > 0.5

- name: Show debug info

debug: var=result #测试result变量

1. Day 03 分布式ELK平台、ES安装、扩展插件
   1. 分布式ELK平台
      1. 重定向
2. 重定向分类

· 重定向

< 读重定向

> 写重定向

>> 追加重定向

<< 终止符

stdin 0 标准输入(读)

stdout 1 标准输出(写)

stderr 2 标准错误(写)

exec 修改文件描述符

1. 给对应终端输出信息
2. echo $$ 查看对应pid
3. 在内存中找到对应进程 /proc/$$/fd
4. 在对应进程输出 echo 'xxx' > / proc/$$/fd/1
5. 主机192.168.1.100：# echo "Hello" > /dev/tcp/192.168.1.10/80
6. 主机192.168.1.10：# tcpdump -i eth0 -n -xx -v host 192.168.1.100
7. exec命令
8. 执行一个命令后退出/修改文件描述符
9. # touch ff ; exec 1 > ff ;
10. # exec 6 > ff ; echo 12345 >& 6 ; exec >&
11. 访问百度首页：

#!/bin/bash

exec 9 <> /dev/tcp/www.baidu.com/80

echo -ne "GET /index.html HTTP/1.1\r\n" >&9

echo -ne "Host: [www.baidu.com\r\n\r\n"](http://www.baidu.com/r/n/r/n\") >&9'

cat <&9

此脚本等价于：

# curl www.baidu.com

* + 1. ELK简介

1. 什么是ELK

· ELK是一整套解决方案, 是三个软件产品的首字母缩写, 很多公司都在使用, 如: Sina、携程、华为、美团等

· ELK分别代表

- Elasticsearch: 负责日志检索和存储

- Logstash: 负责日志的手机和分析、处理

- Kibana: 负责日志的可视化

· 这三款软件都是开源软件, 通常是配合使用, 而且又先后归于Elastic.co公司名下, 故被简称为ELK

1. ELK能做什么

· ELK组件在海量日志的运维中, 可用于解决

- 分布式日志数据集中式查询和管理

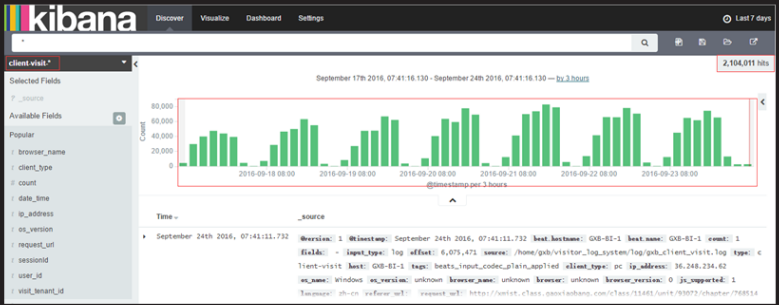
- 系统监控, 包含系统硬件和应用各个组件的监控

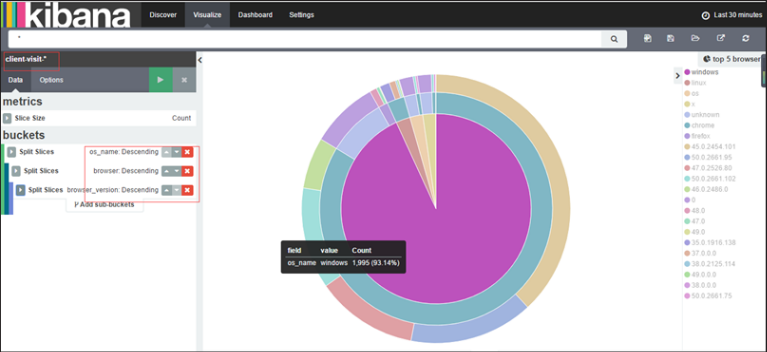
- 故障排查

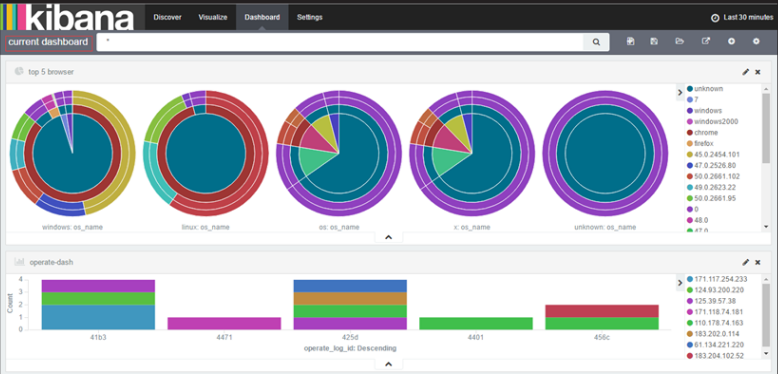
- 安全信息和事件管理

- 报表功能

1. ELK是什么样子







* + 1. Elasticsearch

1. Elasticsearch部分

· ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful API的Web接口

· Elasticsearch使用Java开发，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便

· 主要特点

- 实时分析

- 分布式实时文件存储，并将每一个字段都编入索引

- 文档导向，所有的对象全部都是文档

- 高可用性，易扩展，支持集群(Cluster)、分片和复制(Shards和Replicas)

- 接口友好，支持JSON

· ES没有什么

- ELasticsearch没有典型意义的事务

- Elasticsearch是一种面向文档的数据库

- Elasticsearch没有提供授权和认证特性

· 相关概念

- Node：装有一个ES服务器的节点

- Cluster：有多个Node组成的集群

- Document：一个可被搜索的基础信息单元

- Index：拥有相似特征的文档的集合

- Type：一个索引中可以定义一种或多种类型

- Filed：是ES的最小单位，相当于数据的某一列

- Shards：索引的分片，每一个分片就是一个Shard

- Replicas：索引的拷贝

1. SQL与NoSQL

· ES与关系型数据库的对比

- 在ES中，文档归属于一种类型(type)，而这些类型存在于索引(index)中，类比传统关系型数据库

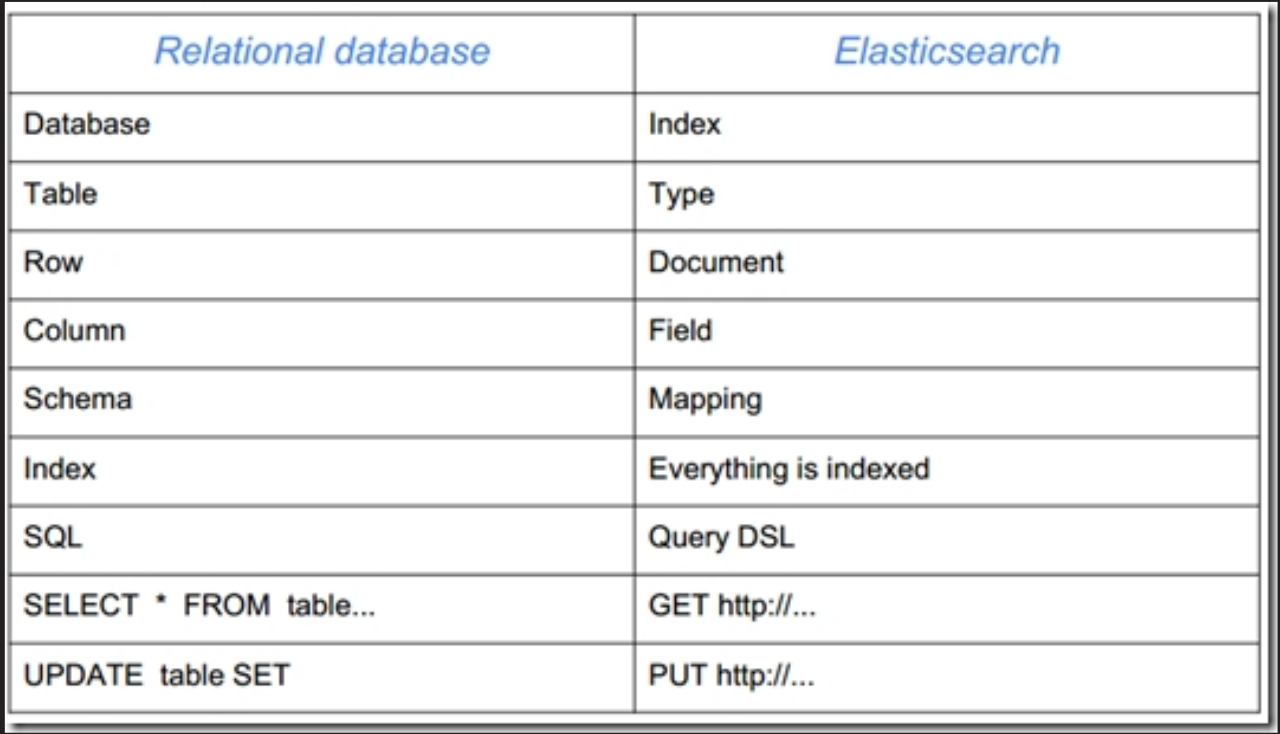
- DB -> Databases -> Tables -> Rows -> Columns

关系型 数据库 表 行 列

- ES -> Indices -> Types -> Documents -> Fields

ES 索引 类型 文档 域(字段)

· ES与关系型数据库的对比



1. Elasticsearch架构图



* 1. ES集群安装
     1. ES基础

1. ES集群安装

· 安装第一台ES服务器

- 设置主机名称和IP对应关系

- 解决依赖关系

- 安装软件包

- 修改配置文件

- 启动服务

- 检查服务

· 设置ip与主机名称对应关系

- 配置/etc/hosts

192.168.1.51 node1

# eip 51

# echo node1 > /etc/hostname

# reboot

# echo '192.168.1.51 node1' >> /etc/hosts

· 安装JDK

- Elasticsearch要求至少Java 7

- 一般推荐使用OpenJDK 1.8

- 配置好安装源以后，我们先解决依赖关系

[root@node1 ~]# rpm -qa | grep -i openjdk

[root@node1 ~]# yum install -y java-1.8.0-openjdk

· 安装ES

[root@node1 ~]# yum install -y elasticsearch

· 修改配置文件

- elasticsearch.yml

network.host: 0.0.0.0

[root@node1 ~]# vim /etc/elasticsearch/elasticsearch.yml

network.host: 192.168.1.51

· 启动服务

- 启动服务并设置开机自启

[root@node1 ~]# systemctl start elasticsearch.service

[root@node1 ~]# systemctl enable elasticsearch.service

- 验证

[root@node1 ~]# ss -nlp | grep 9200

- 能够看到端口9200，9300被监听

· 通过浏览器或curl访问9200端口

# curl <http://192.168.4.11:9200>

{

"name" : "node1",

"cluster\_name" : "my-es",

"version" : {

"number" : "2.3.4",

... ... ...

"build\_snapshot" : false,

"lucene\_version" : "5.5.0"

},

"tagline" : "You Know, for Search"

}

1. ES集群配置

· ES集群配置

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

cluster.name:my-es

node.name: node1

network.host: 0.0.0.0

discovery.zen.ping.unicast.host:["node1","node2","node3"]

- 集群中的所有节点要相互能够ping通，要在所有集群机器上配置/etc.hosts中的主机名与ip对应关系

- 集群中所有机器都要安装Java环境

- cluster.name集群名称配置要求完全一致

- node.name为当前节点标识，应配置本机的主机名

- discovery为集群节点机器，不需要全部配置

- 配置完成以后启动所有节点服务

- 验证集群，使用ES内置字段\_cluster/health

# curl <http://192.168.4.11:9200/cluster/health?pretty>

{

"cluster\_name" : "my-es",

"status" : "green",

... ...

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

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

}

· 通过ansible-playbook的方式配置集群

- 配置ansible.cfg文件,指定主机名文件

[root@node1 ~]# vim ansible.cfg

[defaults]

inventory = elk

- 配置主机名文件elk

[elasticsearch]

node[1:5]

[elasticsearch:vars]

ansible\_ssh\_user="root"

ansible\_ssh\_pass="a"

- 写playbook

[root@node1 ~]# vim elasticsearch.yml

---

- hosts: elasticsearch

remote\_user: root

tasks:

- name: hosts

copy:

src: /etc/hosts

dest: /etc/hosts

- name: yum

copy:

src: /etc/yum.repos.d/

dest: /etc/yum.repos.d/

- yum:

name: java-1.8.0-openjdk,elasticsearch

- name: edit configure

shell: |

sed -i '17s/.\*/cluster.name: my-es/' /etc/elasticsearch/elasticsearch.yml

sed -i "23s/.\*/node.name: ${HOSTNAME}/" /etc/elasticsearch/elasticsearch.yml

sed -i "54s/.\*/network.host: 0.0.0.0/" /etc/elasticsearch/elasticsearch.yml

sed -i "68s/.\*/discovery.zen.ping.unicast.hosts: [\"node1\", \"node2\", \"node3\"]/" /etc/elasticsearch/elasticsearch.yml

- service:

name: elasticsearch

enabled: yes

state: restarted

· 注意: 在playbook中使用shell字段连用时需要写作"shell: | ",否则会报错

· ES集群验证

- 返回字段解析

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

- "number\_of\_nodes" : 5，表示集群中节点的数量

[root@node1 ~]# curl http://node1:9200/\_cluster/health?pretty

{

"cluster\_name" : "my-es",

"status" : "green",

"timed\_out" : false,

"number\_of\_nodes" : 5,

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

"active\_primary\_shards" : 0,

"active\_shards" : 0,

"relocating\_shards" : 0,

"initializing\_shards" : 0,

"unassigned\_shards" : 0,

"delayed\_unassigned\_shards" : 0,

"number\_of\_pending\_tasks" : 0,

"number\_of\_in\_flight\_fetch" : 0,

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

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

}

1. 实验过程
2. elasticsearch集群安装

一共安装5台ES数据库节点

· 配置所有主机的 /etc/hosts

192.168.1.51 node1

192.168.1.52 node2

192.168.1.53 node3

192.168.1.54 node4

192.168.1.55 node5

· 在所有节点安装

# yum install -y java-1.8.0-openjdk elasticsearch

· 修改配置文件

cluster.name:nsd

node.name: 本机主机名

network.host: 0.0.0.0

discovery.zen.ping.unicast.hosts: ["es1", "es2", "es3"]

1. 启动服务

# systemctl start elasticsearch

1. 浏览器访问验证

# curl http://192.168.1.11:9200

* + 1. ES插件的安装与使用

1. ES插件的使用

· ES常用插件

- head插件

- kopf插件

- bigdesk插件

· head插件

- 它展现ES集群的拓扑结构，并且可以通过它来进行索引(Index)和节点(Node)级别的操作

- 它提供一组针对就请你的查询API，并将结果以json和表格形式返回

- 它提供一些快捷菜单，用以展现集群的各种状态

· kopf插件

- 是一个ElasticSearch的管理工具

- 它提供了对ES操作的API

- 是elasticsearch的一个集群监控工具

- 可以通过它来查看es集群的各种状态，如：cpu、内存使用情况、索引数据、搜索情况，http连接数等

· ES插件安装、查看

- 查看安装的插件

/usr/share/elasticsearch/bin/plugin list

- 安装插件

/usr/share/elasticsearch/bin/plugin install <ftp://192.168.1.254/head.zip>

/usr/share/elasticsreach/bin/plugin install [file:///tmp/kopf.zip](file://tmp\\kopf.zip)

- 这里必须使用url的方式进行安装，如果文件在本地，我们也需要使用file://方式指定路径，例如文件在/tmp/xxx下面，我们要写成file:///tmp/xxx，删除使用remove命令

[root@node1 ~]# vim plugins.yml

---

- hosts: elasticsearch

remote\_user: root

tasks:

- shell: |

mycmd=/usr/share/elasticsearch/bin/plugin

$mycmd install ftp://192.168.1.254/elk/elasticsearch-head-master.zip

$mycmd install ftp://192.168.1.254/elk/elasticsearch-kopf-master.zip

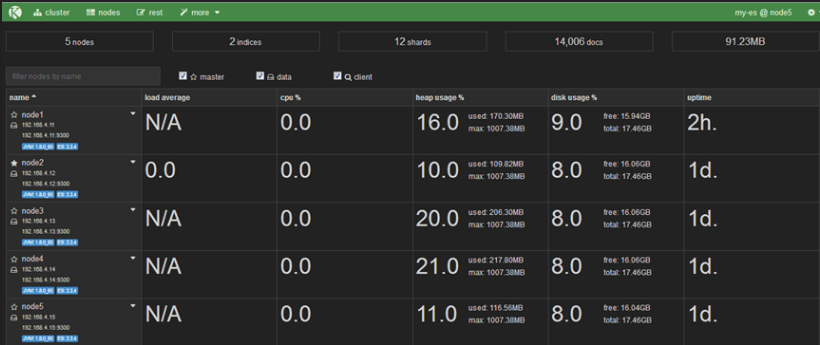
$mycmd install ftp://192.168.1.254/elk/bigdesk-master.zip

[root@node1 ~]# ansible-playbook plugins.yml

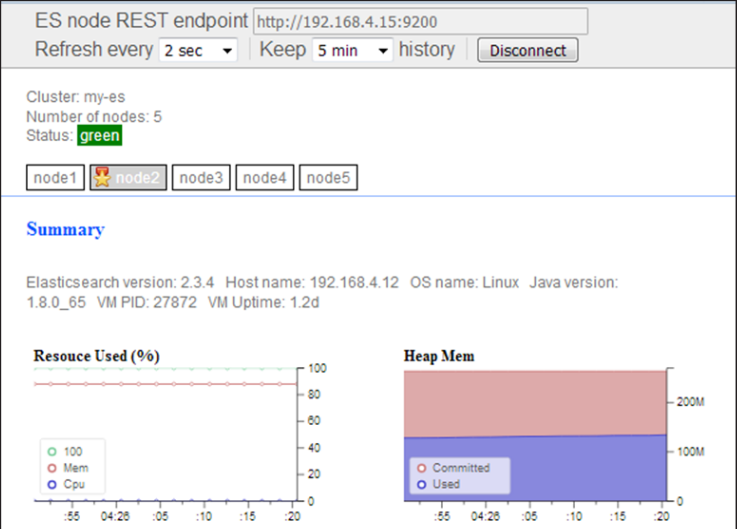
1. ES HEAD插件



1. ES KOPF插件



1. ES bigdesk插件



* 1. 扩展插件
     1. RESTful API

1. HTTP协议简介

· HTTP请求由三部分组成

- 分别是：请求行、消息报头、请求正文

- 请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本，格式如下：

Method Request-URI HTTP-Version CRLF

· http请求方法

- 常用方法GET，POST，HEAD

- 其他方法OPTIONS，PUT，DELET，TRACE和CONNECT

· ES常用

- PUT --- 增

- DELET --- 删

- POST --- 改

- GET --- 查

1. 系统命令curl

· 在Linux中curl是一个利用URL规则在命令行下工作的文件传输工具，可以说是一款很强大的http命令行工具。它支持多种请求模式，自定义请求头等强大功能，是一款综合工具

· curl常用参数介绍

- -A 修改请求agent

# curl -A "Windows/IE6" http://...

- -X 设置请求方法

- -i 显示返回头信息(查看服务器信息)

# curl -i <http://...>

# curl -L -i http://...

1. RESTful API调用

· Elasticsearch提供了一系列RESTful的API

- 检查集群、节点、索引的健康度、状态和统计

- 管理集群、节点、索引的数据及原数据

- 对索引进行CRUD操作及查询操作

- 执行其他高级操作如分页、排序、过滤等

· POST或PUT数据使用json格式

· JSON

- JSON(JavaScript Object Notation), 意思是JavaScript对象表示法, 它是一种基于文本独立于语言的轻量级数据交换格式。

- JSON传输的就是一个字符串

- Python中对应的字符串, 列表, 字典都可以转换成对应的JSON格式

· RESTful API的简单使用

- \_cat API查询集群状态, 节点信息

- v参数显示详细信息

[http://192.168.4.15:9200/\_cat/health?v](http://192.168.1.15:9200/_cat/health?v)

- help显示帮助信息

<http://192.168.4.15:9200/_cat/health?help>

· RESTful API的简单使用

- nodes查询节点状态信息

<http://192.168.4.15:9200/_cat/nodes?v>

- 索引信息

[http://192.168.1.51:9200/\_cat/indices?v](http://192.168.4.15:9200/_cat/indices?v)

· RESTful API增加

- 创建一个索引,并设置分片数量与副本数量

[root@node1 ~]# curl -XPUT 'http://192.168.1.51:9200/tarena/' -d '{

"settings":{

"index":{

"number\_of\_shards": 5,

"number\_of\_replicas": 1

}

}

}'

· RESTful API插入数据

- 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@node1 ~]# curl -XPUT http://192.168.1.51:9200/tarena/n1/1 -d '{

"姓名":"犇犇",

"年龄":"29",

"爱好":"做头发"

}'

[root@node1 ~]# curl -XPUT http://192.168.1.51:9200/tarena/n1/2 -d '{

"姓名":"凯子",

"年龄":"39",

"爱好":"吃香蕉"

}'

[root@node1 ~]# curl -XPUT http://192.168.1.51:9200/tarena/n1/3 -d '{

"姓名":"丁总",

"年龄":"49",

"爱好":"黑丝"

}'

[root@node1 ~]# curl -XPUT http://192.168.1.51:9200/tarena/n1/4 -d '{

"姓名":"静静",

"年龄":"20",

"爱好":"败家"

}'

· POST修改

- 修(改)数据, 使用POST方法

- 在修改数据的时候必须调用\_update关键字

- 调用方式: 数据库地址/索引/类型/id值/\_update

[root@node1 ~]# curl -XPOST \

'http://192.168.1.55:9200/tedu/teacher/3/\_update' -d '{

"doc": {

"年龄": "18"

}

}'

[root@node1 ~]# ansible node1 -m shell -a 'curl -XPOST http://192.168.1.51:9200/tarena/n1/4/\_update -d "{\"doc\":{\"年龄\":\"18\"}}"'

· 查询

- 查询使用GET方法, GET为默认方法

- 查询显示结果时可以用pretty规范显示格式

- 多条查询需要使用\_mget关键字配合json调用

[root@node1 ~]# curl -XGET 'http://192.168.1.55:9200/tedu/teacher/1'

· 删除

-删除的时候可以是文档, 也可以是库, 但不能是类型

[root@node1 ~]# curl -XDELETE 'http://192.168.1.55:9200/tedu/teacher/1'

[root@node1 ~]# curl -XDELETE 'http://192.168.1.55:9200/tedu'

· 批量导入数据

- 使用\_bulk关键字, POST方法, data-binary编码格式

- 导入含有index配置的json文件

# curl -XPOST 'http://192.168.1.52:9200/\_bulk' --data-binary @file

1. Day 04 Kibana安装、Kibana使用、Logstash配置扩展插件
   1. Kibana安装与使用
      1. Kibana
2. Kibana安装与配置

· Kibana是什么

- 数据可视化平台工具

· 特点

- 灵活的分析和可视化平台

- 实时总结流量和数据的图表

- 为不同的用户显示直观的界面

- 即时分享和嵌入的仪表板

· Kibana安装

- Kibana的安装非常简单, 我们使用rpm安装

- Kibana默认安装在/opt/kibana下, 配置文件在/opt/kibana/config/kibana.yml

- 我们只需要修改少量的配置就可以启动

· kibana.yml的配置

- server.port: 5601

- server.host: "0.0.0.0"

- elastcisearch.url: "http://192.168.1.51:9200"

- kibana.index: ".kibana"

– kibana.defaultAppId: "discover"

– elasticsearch.pingTimeout: 1500

– elasticsearch.requestTimeout: 30000

– elasticsearch.startupTimeout: 5000

- 除elasticsearch.url需要配置为我们ES集群的地址外, 其他保持默认值

– 设置开机启动

systemctl enable kibana

– 启动服务

systemctl start kibana

– web访问kibana

[http://192.168.1.50:5601/](http://192.168.1.56:5601/)

[root@kibana ~]# ss -nlp | grep 5601

tcp LISTEN 0 128 \*:5601 \*:\* users:(("node",pid=819,fd=11))

1. map映射

· mapping

- 映射: 创建索引的时候,可以预先定义字段类型及相关属性

- 作用: 这样会让索引建立得更加细致和完善

- 分类: 静态映射和动态映射

- 动态映射: 自动根据数据进行相应的映射

- 静态映射: 自定义字段映射数据类型

· 数据导入以后查看logs是否导入成功

· 修改Kibana的配置文件后启动Kibana,然后查看ES集群, 如果出现.kibana Index表示Kibana与ES集群连接成功

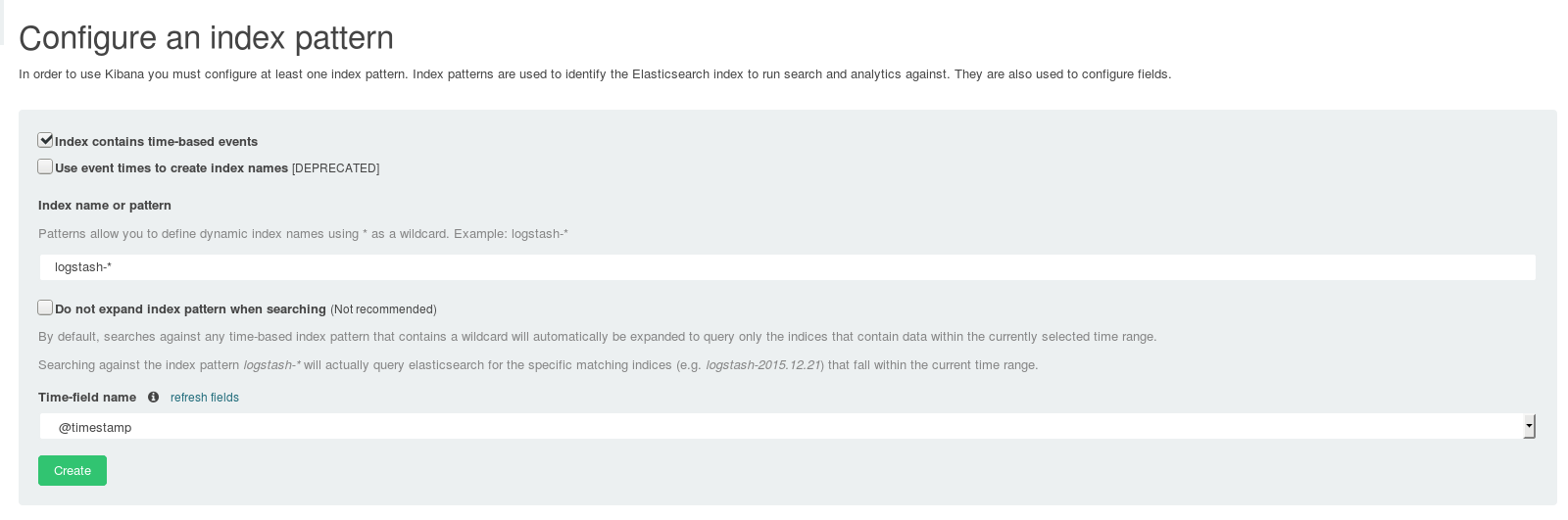
· Kibana里选择日志

- 支持通配符 \*

- 我们在这里选择logstash-\*

- 在下面的Time-field选择@timestamp作为索引

- 然后点create按钮

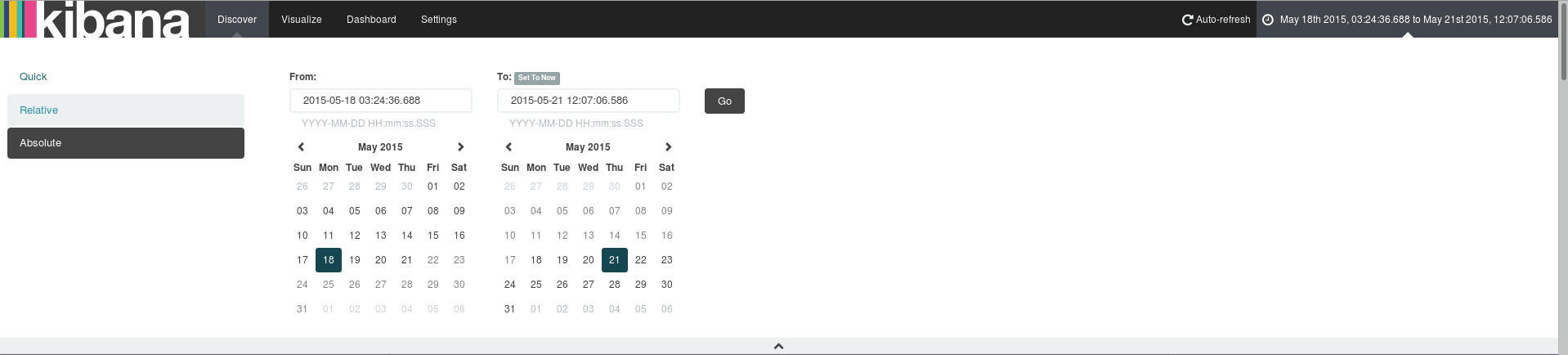


1. Kubana柱状图

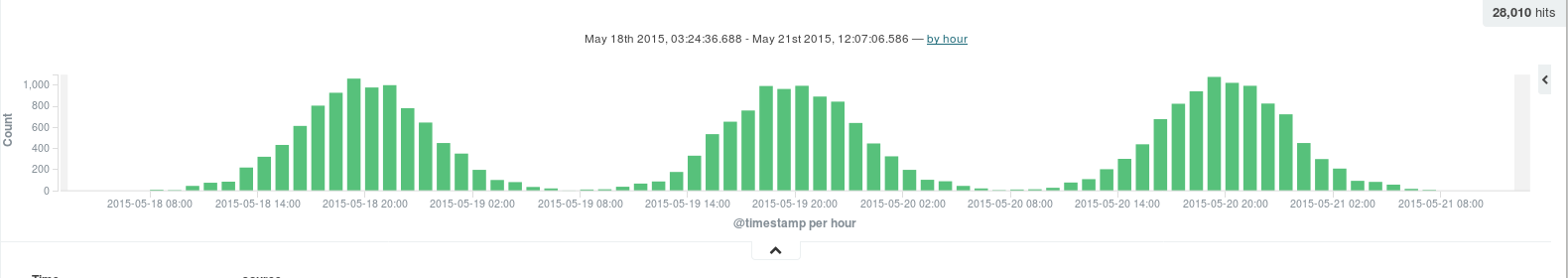
· 导入成功后, 可以选择Discover

· 一开始没有数据, 原因是我们导入的日志是2015-05-10至2015-05-20的时间段, 默认配置是最近15分钟, 在这可以修改时间来显示

· 修改时间: 右上角



· 可以框选放大, 也可以选择"面条"宽度

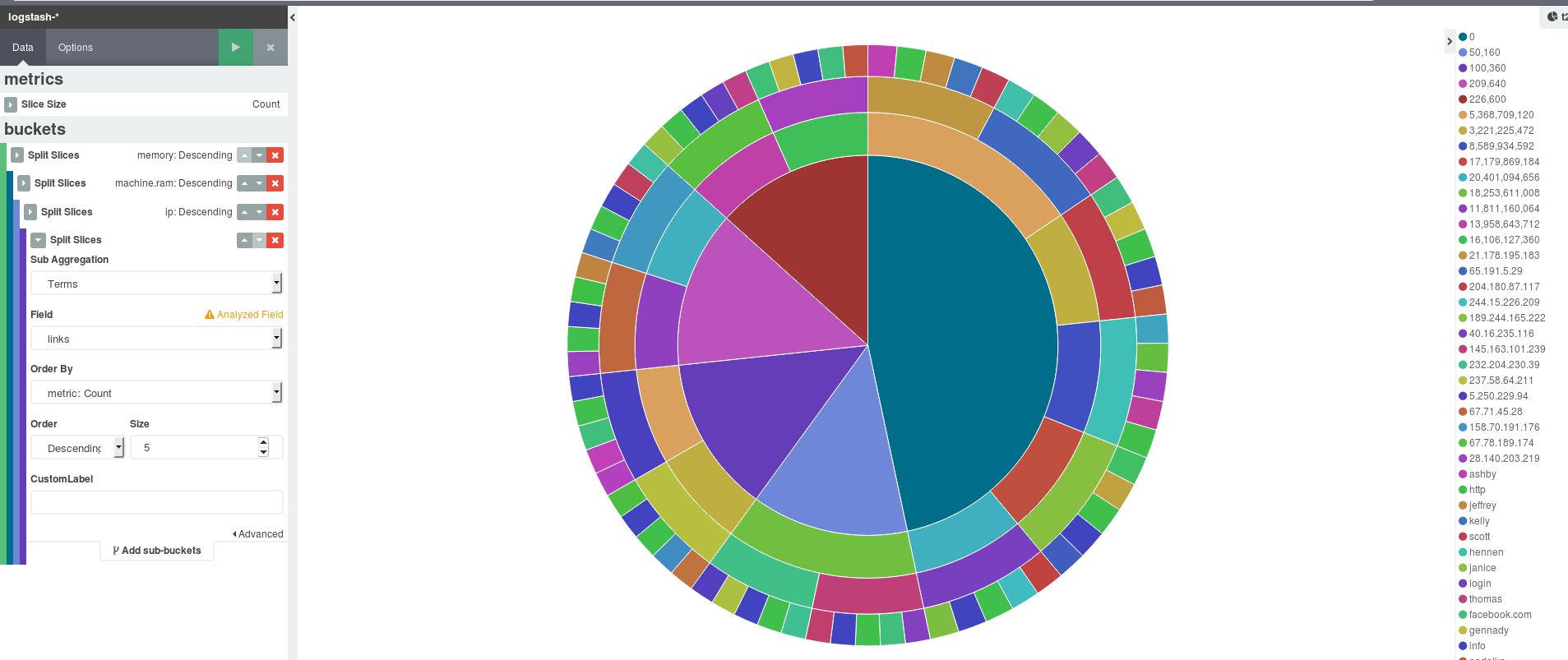


1. Kibana饼图

· 选择Discover后面的Visualize

· 选择Pie chart开始"烙"饼图

· Split Slices(分片)中选择按组分(Terms), fileds定义不同区域的内容, Order定义排序

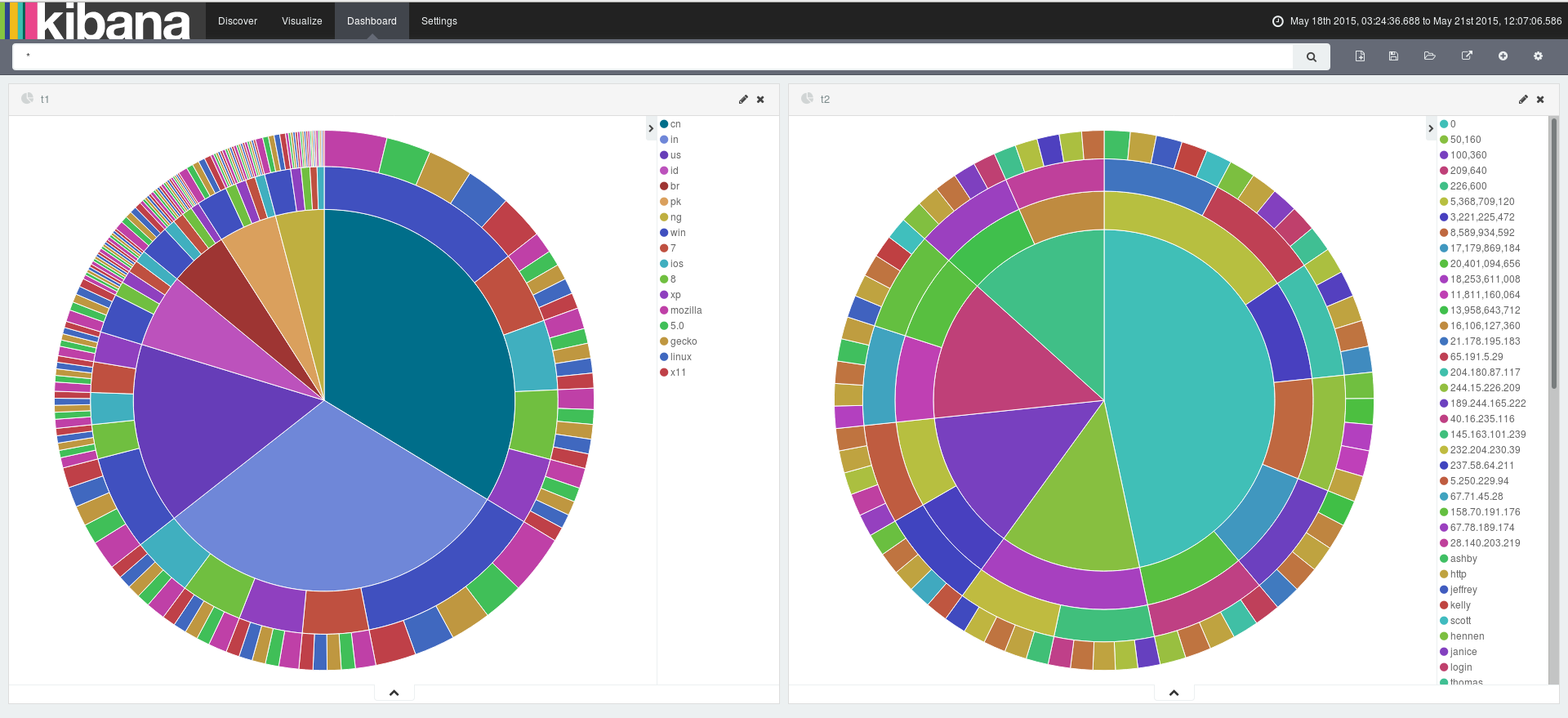


1. Kibana数据展示

· 选择Visualize后面的Dashboard

· 选择加号

· 将做好的饼图添加到展示栏



* 1. Logstash配置扩展插件
     1. Logstash

1. Logstash是什么

· 是一个数据采集、加工处理以及传输的工具

· 特点

- 所有类型的数据集中处理

- 不同模式和格式数据的正常化

- 自定义日志格式的迅速扩展

- 为自定义数据源轻松添加插件

1. Logstash安装

· Logstash安装

- Logstash以来Java环境, 需要安装java-1.8.0-openjdk

- Logstash没有默认的配置文件, 需要手动配置

- Logstash安装在/opt/logstash目录下

yum install -y logstash

· Logstash工作结构

{数据源} ==>

input{ } ==>

filter { } ==>

output{ } ==>

{ES}

数据源 --> 输入 --> 过滤 --> 输出

1. Logstash类型及条件判断

· Logstash里面的类型

- 布尔值类型: ssl\_enable => true

- 字节类型: bytes => "1MiB"

- 字符串类型: name => "xkops"

- 数值类型: port => 22

- 数组: match => ["datetime","UNIX"]

- 哈希: options => {k => "v",k2 => "v2"}

- 编码解码: codec => "json"

- 路径: file\_path => "/tmp/filename"

- 注释: #

· Logstash条件判断

- 等于: ==

- 不等于: !=

- 小于: <

- 大于: >

- 小于等于: <=

- 大于等于: >=

- 匹配正则: =~

- 不匹配正则: !~

- 包含: in

- 不包含: not in

- 与: and

- 或: or

- 非与: nand

- 非或: xor

- 符合表达式: ()

- 取反复合: !()

1. Logstash配置文件

· Logstash的第一个配置文件

- /etc/logstash/logstash.conf

input{

stdin{}

}

filter{ }

output{

stdout{}

}

· 启动并验证

# logstash -f logstash.conf

1. 插件

· Logstash插件

- 上页的配置文件使用了logstash-input-stdin和logstash-output-stdout两个插件, Logstash还有filter和codec类插件, 查看插件的方式是

# /opt/logstash/bin/logstash-plugin list

- 插件及文档地址

<https://github.com/logstash-plugins>

· 练习

- Logstash配置从标准输入读取输入源,然后将结果输出到屏幕

[root@logstash ~]# logstash -f /etc/logstash/logstash.conf

Settings: Default pipeline workers: 2

Pipeline main started

This is a test

2019-06-18T06:30:23.142Z logstash This is a test

The girl who lives in my heart.

2019-06-18T06:31:23.493Z logstash The girl who lives in my heart.

· codec类插件

- plain、json、json\_lines、rubydebug、multiline等

- 使用刚刚的例子, 这次输入json数据

- 设置输入源的codec是json, 在输出的时候选择rubydebug

[root@logstash ~]# vim /etc/logstash/logstash.conf

input{

stdin{ codec => "json" }

}

filter{}

output{

stdout{ codec => "rubydebug" }

}

· input file插件

– sincedb\_path记录读取文件的位置

– start\_position配置第一次读取文件从什么地方开始

[root@logstash ~]# vim /etc/logstash/logstash.conf

input{

file{

path => ["/tmp/a.log", "/var/tmp/b.log"]

}

}

filter{}

output{

stdout{ codec => "rubydebug" }

}

file{

start\_position => "beginning"

//指针起始点

sincedb\_path => "/var/lib/logstash/sincedb-access"

//指针文件位置

path => [“/tmp/a.log”, “/tmp/b.log”]

type => 'filelog'

//定义类型

}

· filter grok插件

- 解析各种非结构化的日志数据插件

- grok使用正则表达式把非结构化的数据结构化

- 在分组匹配, 正则表达式需要根据具体数据结构编写

- 虽然编写困难, 但适用性极广

- 几乎可以应用于各类数据

grok{

match => ["message","%{IP:ip}, (?<key>reg)"]

}

- 宏保存的路径:

/opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-2.0.5/patterns/grok-patterns

· grok正则分组匹配

– 匹配ip时间戳和请求方法

"(?<ip>(\d+\.){3}\d+) \S+

\S+(?<time>.\*\])\s+\"(?<method>[A-Z]+)"]

– 使用正则宏

%{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth}\[%{HTTPDATE:timestamp}\] \"%{WORD:verb}

– 最终版本

%{COMMONAPACHELOG} \"(?<referer>[^\"]+)\"\"(?<UA>[^\"]+)\"

· output ES插件

output{

stdout{ codec => "rubydebug" }

if [type] == "testlog"{

elasticsearch{

hosts => ["node1:9200", "node2:9200"]

index => "testlog"

flush\_size => 2000

idle\_flush\_time => 10

}

}

}

- 调试成功后, 把数据写入ES集群

· input filebeats插件

beats{

port => 5044

}

- 这个插件主要用来接受beats类软件发送过来的数据, 由于logstash以来JAVA环境且占用资源非常大, 因此会使用更轻量的filebeat替代

1. filebeat安装配置

- 使用rpm安装filebeat

# yum –y install filebeat

- 修改配置文件 /etc/filebeat/filebeat.yml

paths:

- /root/logs.jsonl

document\_type: weblog

... ...

paths:

- /root/accounts.json

document\_type: account

output:

logstash:

hosts: ["192.168.1.57:5044"]

- 设置开机运行

# systemctl enable filebeat

- 开启服务

# systemctl start filebeat

1. ELK总结

· 由4个部分组成(5个点)

- 5台Elasticsearch服务器(数据库)

- 1台Kibana服务器(日志管理界面)

- 1台Logstash服务器(日志分析写入, 充当中间件角色)

- 1台搭载filebeat文件传输工具的web服务器(为用户提供服务)

· 部署Elasticsearch

1. 通过rpm安装

# yum install -y java-1.8.0-openjdk elasticsearch

1. 修改配置文件, 配置数据存储集群

cluster.name:my-es

node.name: node1

network.host: 0.0.0.0

discovery.zen.ping.unicast.host:["node1","node2","node3"]

1. 启动服务

# systemctl start elasticsearch

# systemctl enable elasticsearch

1. 此时elasticsearch已经启动, 但想要使用, 需要安装对应的插件

· 安装插件

# mycmd=/usr/share/elasticsearch/bin/plugin

# $mycmd install <ftp://192.168.1.254/elk/elasticsearch-head-master.zip>

# $mycmd install <ftp://192.168.1.254/elk/elasticsearch-kopf-master.zip>

# $mycmd install ftp://192.168.1.254/elk/bigdesk-master.zip

· 查看已安装插件

[root@node1 ~]# /usr/share/elasticsearch/bin/plugin list

Installed plugins in /usr/share/elasticsearch/plugins:

- head

- bigdesk

- kopf

· 通过插件访问数据库

<http://192.168.1.51:9200/_plugin/head>

[http://192.168.1.51:9200/\_plugin/k](http://192.168.1.51:9200/_plugin/head)opf

http://192.168.1.51:9200/\_plugin/bigdesk

1. 除图形外, 还可以通过RESTful API访问Elasticsearch

· 查看索引信息

http://192.168.1.51:9200/\_cat/indices?v

· 数据修改

- 建索引

[root@node1 ~]# curl -XPUT 'http://192.168.1.51:9200/tarena/' -d '{

"settings":{

"index":{

"number\_of\_shards": 5,

"number\_of\_replicas": 1

}

}

}'

- 插数据

[root@node1 ~]# curl -XPUT http://192.168.1.51:9200/tarena/n1/3 -d \

'{

"姓名":"丁总",

"年龄":"49",

"爱好":"黑丝"

}'

- 改数据

[root@node1 ~]# curl -XPOST \

'http://192.168.1.55:9200/tedu/teacher/3/\_update' -d '{

"doc": {

"年龄": "18"

}

}'

- 删数据

[root@node1 ~]# curl -XDELETE \

'http://192.168.1.55:9200/tedu/teacher/1'

[root@node1 ~]# curl -XDELETE 'http://192.168.1.55:9200/tedu'

- 查数据

[root@node1 ~]# curl -XGET 'http://192.168.1.55:9200/tedu/teacher/1'

1. 导入含有index配置的json文件

# curl -XPOST 'http://192.168.1.52:9200/\_bulk' --data-binary @file

· 部署Kibana

1. 通过rpm安装

# yum -y install kibana

1. 修改配置

# vim /opt/kibana/config/kibana.yml

elastcisearch.url: "http://192.168.1.51:9200"

1. 启动服务

# systemctl start kibana

# systemctl enable kibana

1. 映射elasticsearch中的索引
2. 根据数据和分片做图

· 部署logstash

1. 通过rpm安装, 需要Java环境

# yum -y install java-1.8.0-openjdk logstash

1. 配置logstash

· Logstash配置文件由3部分组成

- 输入input, 由此接口将需要进行分片的数据输入

- 过滤器filter, 通过此工具将数据进行分片

- 输出output, 由此将经过分片(分析)后的数据输出(到elasticsearch)

· Logstash十分占用系统资源

· 生产架构中的配置文件 /etc/logstash/logstash.conf

- 因为Logstash十分占用资源, 因此需要用filebeat这样的轻量级工具将需要分析的数据通过5044端口传输至Logstash服务器上进行分片处理

- 分片后的数据传输到Elasticsearch服务器上, 放在索引weblog中

[root@logstash ~]# vim /etc/logstash/logstash.conf

input{

beats{

port => 5044

type => apache\_log

}

}

filter{

if [type] == "apache\_log" {

grok {

match => {"message" => "%{COMBINEDAPACHELOG}" }

}}

}

output{

elasticsearch{

hosts => ["node1:9200", "node2:9200"]

index => "weblog"

flush\_size => 2000

idle\_flush\_time => 10

}

}

1. 启动logstash

# /usr/share/logstash/bin/logstash -f /etc/logstash/logstash.conf

· Web服务器上的filebeat

1. 通过rpm安装

# yum install -y filebeat

1. 修改配置文件

[root@web ~]# grep -Ev '^\s\*(#|$)' /etc/filebeat/filebeat.yml

filebeat:

prospectors:

-

paths:

- /var/log/httpd/access\_log

input\_type: log

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

output:

logstash:

hosts: ["192.168.1.56:5044"]

shipper:

logging:

files:

rotateeverybytes: 10485760 # = 10MB

1. 启动服务

# systemctl start filebeat

# systemctl enable filebeat

· 此时, 使用浏览器访问web服务, 一段时间后可在Kibana页面中查看到对应数据