[1. 云计算介绍 1](#_Toc23848)

[1.1. 典型服务模式 1](#_Toc10090)

[服务器: 1](#_Toc21619)

[1.2. 修改一台计算机名称并加入指定工作组 1](#_Toc13022)

[1.2.1. 步骤一：修改计算机名 1](#_Toc3128)

[1.2.2. 步骤二：计算机加入指定工作组 2](#_Toc29388)

[2. TCP/IP协议及配置 2](#_Toc13780)

[2.1. TCP/IP 协议 2](#_Toc23647)

[2.1.1. TCP/IP 协议简介 2](#_Toc24068)

[2.1.2. IP地址的概述 2](#_Toc27827)

[2.2. 基本环境配置 5](#_Toc27519)

[2.2.1. 更改计算机名和工作组名 5](#_Toc32135)

[2.2.2. TCP/IP参数设置 5](#_Toc12125)

[2.2.3. 配置IP地址 5](#_Toc26749)

[2.2.4. 查看IP有效配置 5](#_Toc32228)

[2.2.5. 用命令查看IP有效配置 5](#_Toc13646)

[2.2.6. 使用ping命令测试网络连通性 5](#_Toc14947)

[3. Linux系统简介 6](#_Toc12953)

[3.1. 什么是Linux? 6](#_Toc22344)

[3.1.1. Linux是一种操作系统 6](#_Toc5689)

[3.1.2. 超级计算机TOP500 6](#_Toc5018)

[3.2. Unix/Linux发展史 6](#_Toc26953)

[3.2.1. Unix系统发展 6](#_Toc18806)

[3.2.2. Linux的诞生 6](#_Toc25913)

[3.2.3. Linux系统内核 6](#_Toc32065)

[3.3. Linux版本及应用 7](#_Toc21157)

[3.3.1. Linux发行版本 7](#_Toc7089)

[3.3.2. Red Hat系统版本 7](#_Toc16336)

[3.3.3. Linux在企业中的应用 8](#_Toc1247)

[4. 安装RHEL7系统 8](#_Toc21476)

[4.1. 预备知识 8](#_Toc13550)

[4.1.1. 如何使用硬盘? 8](#_Toc28021)

[4.1.2. 磁盘与分区表示 9](#_Toc7966)

[4.1.3. Linux文件系统 9](#_Toc28119)

[4.2. 安装RHEL7系统 10](#_Toc13888)

[4.2.1. 准备系统光盘 10](#_Toc24634)

[4.2.2. 安装过程 10](#_Toc13788)

[4.2.3. 初始化过程 10](#_Toc25270)

[4.2.4. 使用虚拟机软件 10](#_Toc6548)

[5. RHEL7基本操作 10](#_Toc10412)

[5.1. 使用图形桌面 10](#_Toc23680)

[5.1.1. 登录到图形桌面 10](#_Toc26911)

[5.1.2. 常见的桌面管理操作 10](#_Toc10398)

[5.2. 命令行基本操作 10](#_Toc24826)

[5.2.1. 获取命令行界面 10](#_Toc7818)

[5.2.2. 查看及切换目录 11](#_Toc4188)

[5.2.3. 查看系统版本 11](#_Toc24595)

[5.2.4. 查看CPU和内存 11](#_Toc21338)

[5.2.5. 查看主机名和IP信息 12](#_Toc7279)

[5.2.6. 关机及重启操作 13](#_Toc14821)

[5.2.7. 创建文档 13](#_Toc4889)

[5.2.8. 文本内容操作 14](#_Toc24489)

[6. 总结和答疑 16](#_Toc27299)

[6.1. RHEL7安装异常 16](#_Toc19493)

[6.1.1. 问题现象 16](#_Toc25340)

[6.1.2. 故障分析及排除 16](#_Toc25681)

[6.2. 图形桌面使用 16](#_Toc26359)

[6.2.1. 问题现象 16](#_Toc641)

[6.2.2. 故障分析及排除 16](#_Toc24354)

[6.3. 命令行操作故障 16](#_Toc11888)

[6.3.1. 问题现象 16](#_Toc30757)

[6.3.2. 故障分析及排除 16](#_Toc32407)

[7. 命令行基础 17](#_Toc24897)

[7.1. 如何编写命令行 17](#_Toc13623)

[7.1.1. 什么是命令? 17](#_Toc27561)

[7.1.2. 命令行的一般格式 17](#_Toc32451)

[7.1.3. 快速编辑技巧 17](#_Toc31183)

[7.2. Mount挂载操作 18](#_Toc9360)

[7.2.1. 什么是挂载? 18](#_Toc26067)

[7.2.2. 挂载光盘或分区 18](#_Toc24040)

[7.2.3. 卸载已挂载的设备或分区 19](#_Toc186)

[8. 目录和文件管理 20](#_Toc31491)

[8.1. 查看及切换目录 20](#_Toc6171)

[8.1.1. 使用pwd、cd 20](#_Toc16493)

[8.1.2. Ls列出文档及属性 20](#_Toc7770)

[8.1.3. 使用通配符 21](#_Toc16090)

[8.1.4. 别名的定义 22](#_Toc12724)

[8.2. 新建文档 22](#_Toc4741)

[8.2.1. Mkdir创建目录 22](#_Toc24740)

[8.2.2. 使用vim创建/修改文件 23](#_Toc3715)

[8.3. 复制/删除/移动 24](#_Toc15972)

[8.3.1. Cp复制 24](#_Toc15284)

[8.3.2. Rm删除 25](#_Toc12448)

[8.3.3. Mv移动/重命名 25](#_Toc4151)

[9. 教学环境介绍 26](#_Toc26514)

[9.1. Linux技能等级 26](#_Toc11151)

[9.1.1. Linux系统管理员 26](#_Toc15389)

[9.1.2. Linux系统工程师 26](#_Toc1241)

[9.2. 使用教学虚拟机 28](#_Toc19361)

[9.2.1. 预装虚拟机说明 28](#_Toc9155)

[9.2.2. 访问练习用虚拟机 28](#_Toc10147)

[9.2.3. 使用rht-vmctl辅助工具 29](#_Toc545)

[9.2.4. 真机远程管理 29](#_Toc24444)

[10. 总结和答疑 30](#_Toc25432)

[10.1. Mkdir建目录失败 30](#_Toc7337)

[10.1.1. 问题现象 30](#_Toc6137)

[10.1.2. 故障分析及排除 30](#_Toc3698)

[10.2. Vim不能保存文件 30](#_Toc5943)

[10.2.1. 问题现象 30](#_Toc2056)

[10.2.2. 故障分析及排除 30](#_Toc25895)

[11. 软件包管理 31](#_Toc3141)

[11.1. 零散软件管理 31](#_Toc13290)

[11.1.1. 使用rpm命令管理软件 31](#_Toc7344)

[11.2. Yum配置及使用 32](#_Toc24365)

[11.2.1. 指定yum软件源 32](#_Toc3453)

[11.2.2. 使用yum命令管理软件 33](#_Toc7332)

[11.2.3. 下载软件包 33](#_Toc28313)

[11.2.4. 升级Linux内核 33](#_Toc20845)

[12. 配置网络 34](#_Toc23042)

[12.1. 临时配置及测试 34](#_Toc15233)

[12.1.1. 查看/设置IP地址 34](#_Toc14845)

[12.1.2. 查看/设置主机名 34](#_Toc22417)

[12.1.3. DNS测试 34](#_Toc28138)

[12.2. 永久配置 34](#_Toc28430)

[12.2.1. 配置静态主机名 34](#_Toc10949)

[12.2.2. 为本机指定DNS服务器 35](#_Toc22314)

[12.3. Nmcli连接管理 35](#_Toc7312)

[12.3.1. 查看网络连接 35](#_Toc28940)

[12.3.2. 修改网络连接配置 35](#_Toc14401)

[12.3.3. 激活/禁用网络连接 36](#_Toc15511)

[13. 文本/文件查找 37](#_Toc27110)

[13.1. 查找文本内容 37](#_Toc6769)

[13.1.1. Grep过滤操作 37](#_Toc26907)

[13.1.2. 重定向输出 37](#_Toc32553)

[13.2. 查找文件 37](#_Toc3596)

[13.2.1. Find按条件查找文件 37](#_Toc10416)

[13.2.2. Find结果处理 37](#_Toc5006)

[14. 总结和答疑 37](#_Toc12421)

[14.1. 配置网络参数 37](#_Toc22305)

[14.1.1. 问题现象 37](#_Toc26682)

[14.1.2. 故障分析及排除 37](#_Toc28115)

[14.2. Classroom异常 37](#_Toc6985)

[14.2.1. 问题现象 37](#_Toc23897)

[14.2.2. 故障分析及排除 37](#_Toc26693)

[15. 管理用户和组 38](#_Toc732)

[15.1. 管理用户账号 38](#_Toc13554)

[15.1.1. 添加用户 38](#_Toc15695)

[15.1.2. 设置登录密码 40](#_Toc7116)

[15.1.3. 修改用户属性 42](#_Toc24389)

[15.1.4. 检查用户ID信息 42](#_Toc26365)

[15.1.5. 删除用户 42](#_Toc19613)

[15.2. 管理组账号 43](#_Toc3031)

[15.2.1. 添加组 43](#_Toc2696)

[15.2.2. 管理组成员 43](#_Toc29516)

[15.2.3. 修改组属性(了解) 44](#_Toc31915)

[15.2.4. 删除组 44](#_Toc32508)

[16. Tar备份与恢复 44](#_Toc8304)

[16.1. Tarball备份包 44](#_Toc30668)

[16.1.1. 归档和压缩 44](#_Toc24516)

[16.1.2. Tar工具的常用选项 46](#_Toc14886)

[16.2. 备份与恢复操作 46](#_Toc7888)

[16.2.1. 制作tar备份包 46](#_Toc21845)

[16.2.2. 查看tar备份包内容 46](#_Toc1135)

[16.2.3. 从tar备份包恢复文档 47](#_Toc16021)

[17. Ntp时间同步 47](#_Toc15220)

[17.1. 校对系统时间 47](#_Toc32209)

[17.1.1. Ntp网络时间协议 47](#_Toc6908)

[17.1.2. 配置chronyd校时服务 47](#_Toc10534)

[17.1.3. 测试校时服务 48](#_Toc22104)

[18. Cron计划任务 48](#_Toc28065)

[18.1. 周期性任务 48](#_Toc6493)

[18.1.1. Cron任务概述 48](#_Toc5845)

[18.1.2. 管理计划任务策略 49](#_Toc23576)

[18.1.3. 如何编写crontab任务记录 49](#_Toc21564)

[19. 总结和答疑 50](#_Toc20791)

[19.1. NTP校时失败 50](#_Toc12928)

[19.1.1. 问题现象 50](#_Toc11432)

[19.1.2. 故障分析及排除 50](#_Toc12313)

[20. 权限和归属 51](#_Toc11321)

[20.1. 基本权限 51](#_Toc20462)

[20.1.1. 基本权限的类别 51](#_Toc23109)

[20.1.2. 查看权限 51](#_Toc25040)

[20.1.3. 设置基本权限 52](#_Toc685)

[20.1.4. 设置文档归属 53](#_Toc14351)

[20.2. 附加权限(特殊权限) 54](#_Toc21675)

[20.2.1. Set GID 54](#_Toc31356)

[20.2.2. 设置附加权限 54](#_Toc16460)

[20.3. Acl访问控制列表 54](#_Toc2693)

[20.3.1. Acl策略的作用 54](#_Toc21641)

[20.3.2. 设置acl访问控制策略 55](#_Toc18787)

[21. 使用LDAP认证 56](#_Toc19287)

[21.1. LDAP目录服务 56](#_Toc22628)

[21.1.1. 什么是LDAP? 56](#_Toc6962)

[21.1.2. 典型的LDAP工作模式 57](#_Toc27865)

[21.2. 如何加入LDAP域 57](#_Toc25059)

[21.2.1. 加入LDAP需要的条件 57](#_Toc23606)

[21.2.2. 配置LDAP认证 57](#_Toc23189)

[21.2.3. 验证LDAP用户登录 58](#_Toc27233)

[22. 家目录漫游 58](#_Toc24949)

[22.1. 访问NFS共享 58](#_Toc29507)

[22.1.1. 什么是NFS共享 58](#_Toc22895)

[22.1.2. 如何访问NFS共享目录 58](#_Toc4630)

[22.2. 挂载LDAP家目录 59](#_Toc11470)

[22.2.1. 漫游家目录的条件 59](#_Toc15375)

[22.2.2. 客户机配置及验证 59](#_Toc10175)

[23. 总结和答疑 59](#_Toc12617)

[23.1. 对目录的w权限 59](#_Toc9224)

[23.1.1. 问题现象 59](#_Toc32291)

[23.1.2. 故障分析及排除 59](#_Toc30587)

[23.2. LDAP家目录漫游 59](#_Toc22636)

[23.2.1. 问题现象 59](#_Toc25314)

[23.2.2. 故障分析及排除 59](#_Toc16144)

[24. 综合串讲 60](#_Toc28033)

[24.1. 附加权限 60](#_Toc14382)

[24.1.1. Set UID 60](#_Toc14681)

[24.1.2. Sticky Bit 60](#_Toc10418)

[24.2. 查找文件 61](#_Toc32464)

[24.2.1. Find扩展 62](#_Toc23016)

[24.3. 提示及指导 65](#_Toc4728)

[24.3.1. 练习注意事项 65](#_Toc9394)

[24.3.2. 指导练习及疑难解答 65](#_Toc11768)

[25. 综合练习 65](#_Toc14136)

[25.1. Linux管理员测试 65](#_Toc15728)

[25.1.1. 环境准备及说明 65](#_Toc20814)

[26. 分区规划及使用 66](#_Toc15567)

[26.1. 硬盘分区管理 66](#_Toc29931)

[26.1.1. 一块硬盘的”艺术”之旅 66](#_Toc19691)

[26.1.2. 使用fdisk分区工具 66](#_Toc13149)

[26.1.3. 识别新分区表 67](#_Toc9830)

[26.2. 格式化分区 67](#_Toc17697)

[26.2.1. 常用的格式化工具 67](#_Toc158)

[26.2.2. 创建文件系统 67](#_Toc18829)

[26.3. 访问文件系统 67](#_Toc30011)

[26.3.1. 访问已格式化的分区 67](#_Toc13483)

[26.3.2. 实现开机自动挂载 68](#_Toc27933)

[27. LVM逻辑卷 70](#_Toc25840)

[27.1. 新建逻辑卷 70](#_Toc18864)

[27.1.1. LVM工作方式 71](#_Toc26552)

[27.1.2. LVM管理工具集 71](#_Toc22004)

[27.1.3. LVM快速部署及使用 71](#_Toc27319)

[27.2. 综合分区规划 71](#_Toc9404)

[27.2.1. 应用需求 71](#_Toc2481)

[27.2.2. 调整现有磁盘分区 71](#_Toc28147)

[27.2.3. LVM卷组的PE大小 71](#_Toc25405)

[27.3. 扩展逻辑卷大小 73](#_Toc19169)

[27.3.1. 检查现有逻辑卷大小 73](#_Toc29793)

[27.3.2. 扩展卷组 73](#_Toc29129)

[27.3.3. 扩展逻辑卷 73](#_Toc31303)

[27.3.4. 更新文件系统大小 73](#_Toc21342)

[28. 总结和答疑 74](#_Toc21081)

[28.1. 逻辑卷扩展 74](#_Toc20693)

[28.1.1. 问题现象 74](#_Toc21642)

[28.1.2. 故障分析及排除 75](#_Toc14245)

[28.1.3. 75](#_Toc16531)

[29. Shell脚本基础 76](#_Toc22584)

[29.1. 认识Shell环境 76](#_Toc24173)

[29.1.1. Bash shell的使用方式 76](#_Toc30479)

[29.1.2. 什么是shell脚本 76](#_Toc13141)

[29.1.3. 规范shell脚本的一般组成 76](#_Toc5167)

[29.2. 第一个shell脚本 76](#_Toc30737)

[29.2.1. 脚本的创建过程 76](#_Toc8911)

[29.2.2. 编写一个批处理shell脚本 77](#_Toc8792)

[29.2.3. 脚本运行及调试 77](#_Toc23456)

[29.3. 简单脚本技巧 77](#_Toc8846)

[29.3.1. 管道传递 77](#_Toc26149)

[29.3.2. 免交互及输出处理 78](#_Toc16426)

[29.3.3. 重定向输出 78](#_Toc14834)

[30. 使用变量 79](#_Toc4736)

[30.1. 变量的定义/引用 79](#_Toc24276)

[30.1.1. 什么是变量 79](#_Toc16593)

[30.1.2. 定义/赋值变量 79](#_Toc11042)

[30.1.3. 查看/引用变量 80](#_Toc19395)

[30.2. 变量的种类 80](#_Toc22997)

[30.2.1. 运维角度的变量区分 80](#_Toc29617)

[30.2.2. 环境变量 80](#_Toc24083)

[30.2.3. 位置变量 81](#_Toc10874)

[30.2.4. 预定义变量 82](#_Toc18509)

[31. 条件测试及选择 83](#_Toc13488)

[31.1. 条件测试 83](#_Toc28966)

[31.1.1. Shell测试的依据 83](#_Toc32086)

[31.1.2. 测试操作 83](#_Toc189)

[31.1.3. 常用的测试选项 83](#_Toc30871)

[31.2. If选择结构 84](#_Toc23819)

[31.2.1. If单分支处理 84](#_Toc5397)

[31.2.2. If双分支处理 84](#_Toc8312)

[31.2.3. If多分支处理 86](#_Toc30907)

[32. 列表式循环 89](#_Toc471)

[32.1. For循环结构 89](#_Toc27976)

[32.1.1. 列表式循环场景 89](#_Toc8539)

[32.1.2. For循环处理 89](#_Toc16862)

[32.1.3. 利用命令替换取值 89](#_Toc2646)

[33. 总结和答疑 91](#_Toc21082)

[33.1. 脚本语法错误 91](#_Toc30648)

[33.1.1. 问题现象 91](#_Toc10832)

[33.1.2. 故障分析及排除 91](#_Toc6562)

[33.1.3. 91](#_Toc10446)

[34. 系统安全保护 92](#_Toc19603)

[34.1. Selinux安全机制 92](#_Toc29529)

[34.1.1. Selinux概述 92](#_Toc30452)

[34.1.2. Selinux运行模式的切换 92](#_Toc14792)

[35. 配置用户环境 93](#_Toc10257)

[35.1. 自定义命令 93](#_Toc6620)

[35.1.1. Linux命令字的来源 93](#_Toc11195)

[35.1.2. Alias别名设置 93](#_Toc10447)

[35.2. 用户初始化文件 93](#_Toc8237)

[35.2.1. 用户个性化配置文件 93](#_Toc6708)

[35.2.2. 全局环境配置 93](#_Toc18411)

[36. 配置高级连接 93](#_Toc30417)

[36.1. 配置IPv6地址 93](#_Toc24493)

[36.1.1. IPv6地址的组成 93](#_Toc11106)

[36.1.2. Nmcli命令行配置 94](#_Toc16536)

[36.2. 配置聚合连接 94](#_Toc13927)

[36.2.1. 链路聚合的优势 94](#_Toc6484)

[36.2.2. 实现链路聚合的条件 94](#_Toc13674)

[36.2.3. Nmcli命令行配置 94](#_Toc7363)

[36.2.4. 激活聚合连接 95](#_Toc21851)

[37. 防火墙策略管理 96](#_Toc15194)

[37.1. Firewalld服务基础 96](#_Toc1786)

[37.1.1. RHEL7的防火墙体系 97](#_Toc25716)

[37.1.2. 预设安全区域 97](#_Toc8040)

[37.1.3. 查看防火墙规则列表 98](#_Toc30489)

[37.2. 配置防火墙 98](#_Toc3661)

[37.2.1. 指定默认的安全区域 98](#_Toc1545)

[37.2.2. 封网段、开服务 102](#_Toc24128)

[37.2.3. 实现本机的端口映射 102](#_Toc28937)

[38. 总结和答疑 103](#_Toc6479)

[38.1. 配置IPv6地址 103](#_Toc9677)

[38.1.1. 问题现象 103](#_Toc15060)

[38.1.2. 故障分析及排除 103](#_Toc11502)

[38.2. 配置聚合连接 103](#_Toc24834)

[38.2.1. 问题现象 103](#_Toc3020)

[38.2.2. 故障分析及排除 103](#_Toc17359)

[39. 配置SMB共享 104](#_Toc2640)

[39.1. Samba服务基础 104](#_Toc25632)

[39.1.1. Samba服务基础 104](#_Toc9190)

[39.1.2. 管理共享账号 105](#_Toc15715)

[39.1.3. 配置文件及参数 105](#_Toc25900)

[39.1.4. Selinux对SMB共享的保护 106](#_Toc21762)

[39.2. 访问共享文件夹 106](#_Toc17443)

[39.2.1. 使用smbclient测试 106](#_Toc7753)

[39.2.2. 使用mount挂载 107](#_Toc12009)

[39.3. Multiuser机制 109](#_Toc7872)

[39.3.1. 共享文件夹的多用户访问 109](#_Toc11590)

[39.3.2. 挂载参数调整 110](#_Toc24067)

[39.3.3. 切换共享用户身份 110](#_Toc5005)

[40. 配置NFS共享 110](#_Toc17818)

[40.1. 普通NFS服务 110](#_Toc31141)

[40.1.1. NFS共享概述 110](#_Toc30821)

[40.1.2. Exports配置文件解析 112](#_Toc32561)

[40.1.3. 访问NFS共享文件夹 112](#_Toc9878)

[41. 总结和答疑 112](#_Toc28215)

[41.1. SMB访问失败 112](#_Toc19027)

[41.1.1. 问题现象 112](#_Toc6067)

[41.1.2. 故障分析及排除 113](#_Toc23147)

[41.2. Selinux策略故障 113](#_Toc25071)

[41.2.1. 问题现象 113](#_Toc11212)

[41.2.2. 故障分析及排除 113](#_Toc32426)

[41.2.3. 114](#_Toc16286)

[42. Iscsi网络磁盘 115](#_Toc12104)

[42.1. Iscsi服务基础 115](#_Toc27432)

[42.1.1. Iscsi磁盘的工作模式 115](#_Toc28575)

[42.1.2. Iscsi磁盘的构成 115](#_Toc28651)

[42.2. 发布iscsi磁盘 116](#_Toc25634)

[42.2.1. 准备空闲存储设备 116](#_Toc3485)

[42.2.2. 使用targetcli建立配置 116](#_Toc3706)

[42.2.3. 确认发布结果 117](#_Toc9446)

[42.3. 访问iscsi磁盘 117](#_Toc9164)

[42.3.1. 客户机配置IQN名称 117](#_Toc30984)

[42.3.2. 连接、发现iscsi磁盘 118](#_Toc31290)

[42.3.3. 开机启动配置 119](#_Toc20450)

[42.3.4. Iscsi设备的挂载设备 119](#_Toc28514)

[43. 数据库服务基础 119](#_Toc27370)

[43.1. 构建数据库系统 119](#_Toc30697)

[43.1.1. 什么是数据库 119](#_Toc26898)

[43.1.2. 部署mariadb数据库服务器 120](#_Toc18521)

[43.1.3. 访问mariadb数据库 120](#_Toc24753)

[43.2. 基本管理操作 120](#_Toc18895)

[43.2.1. Mariadb服务端配置调整 120](#_Toc14830)

[43.2.2. 数据库的增删查 120](#_Toc24920)

[43.2.3. 数据库的导出/导入操作 121](#_Toc21818)

[43.2.4. 用户授权设置 122](#_Toc1767)

[44. 管理表数据 123](#_Toc9944)

[44.1. 数据表的增删查 123](#_Toc20809)

[44.1.1. 创建新的数据表 123](#_Toc30611)

[44.1.2. 查看表结构 123](#_Toc1679)

[44.1.3. 删除数据表 123](#_Toc15360)

[44.2. 表记录基本操作 123](#_Toc31921)

[44.2.1. Insert插入表记录 123](#_Toc16133)

[44.2.2. Select查询表记录 123](#_Toc13443)

[44.2.3. Update修改表记录 124](#_Toc389)

[44.2.4. Delete删除表记录 124](#_Toc10556)

[44.2.5. 多表组合查询 125](#_Toc82)

[45. 总结和答疑 126](#_Toc24844)

[45.1. Mariadb数据库 127](#_Toc23261)

[45.1.1. 问题现象 127](#_Toc28573)

[45.1.2. 故障分析及排除 127](#_Toc26683)

[45.1.3. 127](#_Toc30070)

[46. http服务基础 128](#_Toc32660)

[46.1. 独立web主机 128](#_Toc32581)

[46.1.1. Web通信基本概念 128](#_Toc5311)

[46.1.2. RHEL7中的web服务 128](#_Toc2717)

[46.1.3. 独立web站点的快速部署 130](#_Toc27216)

[46.1.4. 访问web站点 130](#_Toc6845)

[46.2. 虚拟web主机 130](#_Toc8342)

[46.2.1. 虚拟主机的含义及类型 130](#_Toc30880)

[46.2.2. 配置一个虚拟站点 131](#_Toc26360)

[46.2.3. 对默认web站点的影响 132](#_Toc24314)

[47. 网页内容访问 132](#_Toc31363)

[47.1. 配置目录访问 132](#_Toc30511)

[47.1.1. 文件夹权限 132](#_Toc17797)

[47.1.2. 客户机地址限制 135](#_Toc14050)

[47.2. Selinux策略保护 135](#_Toc4450)

[47.2.1. 标准web目录 135](#_Toc27400)

[47.2.2. 增加新的web目录 135](#_Toc978)

[48. 部署动态网站 135](#_Toc25776)

[48.1. 动态网站概述 135](#_Toc11031)

[48.1.1. 静态网站的运行 135](#_Toc7380)

[48.1.2. 动态网站的运行 135](#_Toc14582)

[48.2. 部署WSGI应用 136](#_Toc4084)

[48.2.1. 快速构建WSGI动态网站 136](#_Toc19069)

[48.2.2. 测试WSGI网页 137](#_Toc22347)

[48.3. Selinux策略保护 137](#_Toc19153)

[48.3.1. 标配web端口 137](#_Toc29914)

[48.3.2. 允许开放其他web端口 137](#_Toc12703)

[49. 总结和答疑 138](#_Toc12447)

[49.1. Httpd启动失败 138](#_Toc8677)

[49.1.1. 问题现象 138](#_Toc6320)

[49.1.2. 故障分析及排除 138](#_Toc21547)

[49.2. Selinux策略故障 138](#_Toc792)

[49.2.1. 问题现象 138](#_Toc20327)

[49.2.2. 故障分析及排除 138](#_Toc20003)

[50. 综合串讲 139](#_Toc22779)

[50.1. 安全web服务 139](#_Toc22934)

[50.1.1. PKI公钥基础设施 139](#_Toc24521)

[50.1.2. 实现HTTPS加密的条件 139](#_Toc13014)

[50.1.3. 部署证书、密钥相关文件 139](#_Toc20401)

[50.1.4. 调整web服务配置 140](#_Toc8780)

[50.1.5. 访问HTTPS加密的站点 140](#_Toc11621)

[50.2. 基础邮件服务 140](#_Toc18447)

[50.2.1. 电子邮件通信 140](#_Toc13861)

[50.2.2. 快速部署postfix邮件服务器 141](#_Toc7123)

[50.2.3. 使用mail命令发信/收信 142](#_Toc23654)

[50.3. 准备交换分区 142](#_Toc396)

[50.3.1. Parted分区工具 142](#_Toc27246)

[50.3.2. 什么是交换空间 143](#_Toc16377)

[50.3.3. 如何建立一个交换分区 143](#_Toc816)

[50.3.4. 交换空间的启用控制 144](#_Toc20657)

[51. 综合练习 144](#_Toc15022)

[51.1. Linux工程师测试 144](#_Toc32103)

[51.1.1. 环境准备及说明 144](#_Toc2719)

[52. 扩展的几个应用 145](#_Toc7294)

[52.1. 补充技巧 145](#_Toc19770)

[52.1.1. 目录结构 145](#_Toc3886)

[52.1.2. 权限的数值表示 145](#_Toc30537)

[52.1.3. 历史命令 146](#_Toc30143)

[52.1.4. 实用命令小工具 147](#_Toc21499)

[52.2. 获取命令帮助 148](#_Toc29347)

[52.2.1. 查看命令的简要说明 148](#_Toc22532)

[52.2.2. 实用man手册 149](#_Toc23426)

[52.3. zip归档工具(跨平台的压缩工具) 149](#_Toc3860)

[52.3.1. 制作.zip压缩包 149](#_Toc28525)

[52.3.2. 释放.zip压缩包 149](#_Toc32169)

[53. 发布网络YUM源 149](#_Toc9166)

[53.1. 准备YUM源 149](#_Toc25284)

[53.1.1. yum仓库特点 149](#_Toc26076)

[53.1.2. 实用RHEL7光盘库 150](#_Toc24599)

[53.1.3. 实用第三方RPM包建库 151](#_Toc23698)

[53.2. 源的HTTP发布 152](#_Toc1166)

[53.2.1. 快速构建HTTP服务器 152](#_Toc22397)

[53.2.2. 部署HTTP资源 152](#_Toc21626)

[53.2.3. 测试HTTP方式的yum源 152](#_Toc14556)

[54. vim编辑技巧 152](#_Toc920)

[54.1. 命令模式操作 153](#_Toc24043)

[54.1.1. 光标跳转 153](#_Toc30925)

[54.1.2. 复制/粘贴/删除 153](#_Toc24139)

[54.1.3. 查找/撤销/保存 153](#_Toc12370)

[54.2. 末行模式操作 154](#_Toc999)

[54.2.1. 保存/退出/文件操作 154](#_Toc30877)

[54.2.2. 字符串替换 154](#_Toc9927)

[54.2.3. 开关参数的控制 155](#_Toc10071)

[55. 源码编译安装 155](#_Toc3856)

[55.1. 编译安装介绍 155](#_Toc10863)

[55.1.1. 源码编译安装的优势 155](#_Toc13692)

[55.1.2. 准备编译环境 156](#_Toc27175)

[55.1.3. 基本实现过程 156](#_Toc9683)

[55.2. 分布解析及示例 156](#_Toc31671)

[55.2.1. 下载及解包 156](#_Toc23935)

[55.2.2. 配置 156](#_Toc21312)

[55.2.3. 编译及安装 156](#_Toc4196)

[55.2.4. 结果验证 156](#_Toc28337)

[56. systemctl控制 156](#_Toc32717)

[56.1. systemd介绍 156](#_Toc16607)

[56.1.1. init程序的作用 156](#_Toc30505)

[56.1.2. systemd 156](#_Toc9152)

[56.1.3. unit配置单元 159](#_Toc15305)

[56.2. 管理系统服务 159](#_Toc13741)

[56.2.1. 列出服务 159](#_Toc28595)

[56.2.2. 启动/停止/重启/看状态 159](#_Toc16768)

[56.2.3. 配置开机自启 159](#_Toc10162)

[56.3. 管理运行级别 159](#_Toc4289)

[56.3.1. 切换级别 159](#_Toc14162)

[56.3.2. 设置默认级别 159](#_Toc16464)

[57. 总结和答疑 159](#_Toc2420)

[57.1. vim编辑器异常 159](#_Toc13087)

[57.1.1. 问题现象 159](#_Toc21129)

[57.1.2. 故障分析及排除 159](#_Toc23547)

[57.2. 编码编译安装 159](#_Toc4548)

[57.2.1. 问题现象 159](#_Toc7806)

[57.2.2. 故障分析及排除 159](#_Toc16576)

[57.2.3. systemctl控制 问题现象 160](#_Toc18584)

[57.2.4. 故障分析及排除 160](#_Toc19819)

[57.2.5. 160](#_Toc29007)

[58. DNS服务基础 161](#_Toc6201)

[58.1. DNS工作原理 161](#_Toc11950)

[58.1.1. DNS解析的作用 161](#_Toc16610)

[58.1.2. DNS的分布式结构 161](#_Toc25594)

[58.1.3. DNS域名管理 161](#_Toc7749)

[58.2. BIND服务分析 162](#_Toc12956)

[58.2.1. BIND域名服务 162](#_Toc25120)

[58.2.2. named.conf配置文件 162](#_Toc11115)

[58.2.3. 地址库文件 163](#_Toc12216)

[58.3. 单区域DNS服务 164](#_Toc2872)

[58.3.1. 快速搭建DNS服务器 164](#_Toc25805)

[58.3.2. DNS查询测试 165](#_Toc3482)

[59. 特殊解析 165](#_Toc27575)

[59.1. 地址记录应用 165](#_Toc3078)

[59.1.1. DNS轮询 165](#_Toc28309)

[59.1.2. 泛域名解析 165](#_Toc25119)

[60. DNS子域授权 166](#_Toc17613)

[60.1. 配置子域授权 166](#_Toc30739)

[60.1.1. 子域授权的作用 166](#_Toc21187)

[60.1.2. 案例环境及要点 167](#_Toc4569)

[60.1.3. 为父DNS启用子域授权 167](#_Toc8468)

[60.1.4. 测试子域FQDN查询 168](#_Toc8215)

[60.2. 递归/迭代查询 168](#_Toc9828)

[60.2.1. 递归/迭代的作用 168](#_Toc11565)

[60.2.2. DNS查询的工作方式 169](#_Toc8370)

[60.2.3. 如何开启/禁用递归 169](#_Toc28599)

[60.2.4. 验证迭代查询 169](#_Toc604)

[61. 缓存DNS 169](#_Toc5636)

[61.1. 缓存DNS概述 169](#_Toc22583)

[61.1.1. 互联网DNS角色 169](#_Toc14927)

[61.1.2. 缓存DNS的适用场景 169](#_Toc31074)

[61.1.3. 解析记录来源 170](#_Toc11714)

[61.2. 全局转发式缓存 171](#_Toc5828)

[61.2.1. 配置要点 171](#_Toc30857)

[61.2.2. 测试查询效果 171](#_Toc12108)

[62. 总结和答疑 171](#_Toc8060)

[62.1. NS资源记录 171](#_Toc22386)

[62.1.1. 问题现象 171](#_Toc1071)

[62.1.2. 故障分析及排除 171](#_Toc30093)

[62.2. 地址库文件权限 172](#_Toc6758)

[62.2.1. 问题现象 172](#_Toc25073)

[62.2.2. 故障分析及排除 172](#_Toc17981)

[62.2.3. 172](#_Toc4348)

[63. Split分离解析 173](#_Toc4234)

[63.1. 分离解析概述 173](#_Toc24176)

[63.1.1. 什么是分离解析 173](#_Toc2875)

[63.1.2. 典型适用场景 173](#_Toc3287)

[63.1.3. BIND的view视图 173](#_Toc19904)

[63.1.4. acl地址列表 175](#_Toc24765)

[63.2. 配置分离解析 176](#_Toc16135)

[63.2.1. 案例需求及要点 176](#_Toc31450)

[63.2.2. 配置Split分离解析 176](#_Toc10108)

[63.2.3. 测试分离解析 176](#_Toc24997)

[64. RAID磁盘阵列 176](#_Toc24863)

[64.1. RAID磁盘阵列 176](#_Toc17855)

[64.1.1. RAID阵列概述 176](#_Toc26600)

[64.1.2. RAID0/1/10 176](#_Toc26046)

[64.1.3. RAID5/6 177](#_Toc507)

[64.1.4. RAID各级别特点对比 177](#_Toc25596)

[64.1.5. RAID阵列实现方式 177](#_Toc29182)

[65. 进程管理 178](#_Toc25427)

[65.1. 查看进程 178](#_Toc22260)

[65.1.1. 查看进程树 178](#_Toc6178)

[65.1.2. 查看进程快照 179](#_Toc26521)

[65.1.3. 进程动态排名 179](#_Toc19108)

[65.1.4. 检索进程 180](#_Toc19936)

[65.2. 控制进程 180](#_Toc7425)

[65.2.1. 进程的前后台调度 180](#_Toc19283)

[65.2.2. 杀死进程 181](#_Toc15023)

[66. 日志管理 182](#_Toc1060)

[66.1. 日志概述 182](#_Toc4253)

[66.1.1. 日志的功能 182](#_Toc8435)

[66.1.2. 内核及系统日志 183](#_Toc3426)

[66.1.3. 用户日志 183](#_Toc29620)

[66.2. 日志分析 183](#_Toc29199)

[66.2.1. 查看文本日志消息 183](#_Toc1657)

[66.2.2. 用户登录分析 183](#_Toc28796)

[66.2.3. 日志消息的优先级 184](#_Toc31642)

[66.2.4. 使用journalctl工具 185](#_Toc508)

[67. 总结和答疑 185](#_Toc11797)

[67.1. 进程管理 185](#_Toc2235)

[67.1.1. 问题现象 185](#_Toc17892)

[67.1.2. 故障分析及排除 185](#_Toc10570)

[67.1.3. 185](#_Toc26602)

[68. 批量装机环境 186](#_Toc1351)

[68.1. 部署DHCP服务器 186](#_Toc9834)

[68.1.1. DHCP概述及原理 186](#_Toc2176)

[68.1.2. 配置dhcpd地址分配服务 186](#_Toc15916)

[68.1.3. dhclient测试 187](#_Toc17421)

[68.2. 网络装机概述 187](#_Toc19889)

[68.2.1. 网络装机的优势 187](#_Toc28449)

[68.2.2. 什么是PXE网络 187](#_Toc24362)

[68.2.3. PXE组件及过程分析 188](#_Toc9142)

[68.3. 基础条件 188](#_Toc8975)

[68.3.1. 整体思路 188](#_Toc23865)

[68.3.2. 提供软件安装源 188](#_Toc26763)

[69. 配置PXE引导 188](#_Toc12761)

[69.1. 部署TFTP服务 188](#_Toc2576)

[69.1.1. 启用TFTP服务端 188](#_Toc17205)

[69.1.2. 提供PXE启动程序 189](#_Toc30679)

[69.1.3. 提供装机内核及初始文件 189](#_Toc2415)

[69.2. 配置启动菜单 189](#_Toc5913)

[69.2.1. 拷贝模板文件 189](#_Toc29687)

[69.2.2. default配置文件调整 190](#_Toc9274)

[69.2.3. 确认发布结果 191](#_Toc30088)

[69.3. PXE装机测试 191](#_Toc30539)

[69.3.1. 新建一台PXE虚拟机 191](#_Toc32042)

[69.3.2. PXE引导及验证 192](#_Toc11325)

[70. kickstart自动应答 192](#_Toc4010)

[70.1. kickstart概述 192](#_Toc22995)

[70.1.1. 什么是kickstart技术 192](#_Toc11382)

[70.1.2. 应答文件从哪来? 192](#_Toc13798)

[70.2. 使用应答文件 192](#_Toc6015)

[70.2.1. 创建应答文件 192](#_Toc13097)

[70.2.2. 启用应答文件 193](#_Toc12345)

[70.3. 无人值守装机 193](#_Toc21645)

[70.3.1. PXE+kickstart测试 194](#_Toc12795)

[71. 总结和答疑 194](#_Toc9127)

[71.1. TFTP连接失败 194](#_Toc13614)

[71.1.1. 问题现象 194](#_Toc32674)

[71.1.2. 故障分析及排除 194](#_Toc19228)

[71.2. DHCP服务冲突 194](#_Toc12383)

[71.2.1. 问题现象 194](#_Toc4333)

[71.2.2. 故障分析及排除 194](#_Toc18415)

[71.3. PXE应答不完整 194](#_Toc15317)

[71.3.1. 问题现象 194](#_Toc27793)

[71.3.2. 故障分析及排除 194](#_Toc28552)

[71.3.3. 194](#_Toc14287)

[72. rsync同步操作 195](#_Toc31081)

[72.1. rsync基本使用 195](#_Toc31612)

[72.1.1. rsync概述 195](#_Toc7696)

[72.1.2. rsync同步操作 195](#_Toc923)

[72.1.3. 同步控制 195](#_Toc21528)

[72.2. rsync+SSH同步 196](#_Toc24154)

[72.2.1. 用法及服务端要求 196](#_Toc7782)

[72.2.2. 下行同步示例 197](#_Toc2301)

[72.2.3. 上行同步示例 197](#_Toc27957)

[73. inotify实时同步 197](#_Toc10242)

[73.1. 部署监控环境 197](#_Toc1770)

[73.1.1. 同步的实时性 197](#_Toc11063)

[73.1.2. 安装inotify-tools工具 198](#_Toc28279)

[73.1.3. inotifywait监控 199](#_Toc31409)

[73.2. 配置实时同步 200](#_Toc24769)

[73.2.1. inotify与rsync的结合 200](#_Toc10311)

[73.2.2. 编写同步脚本 200](#_Toc22211)

[73.2.3. 验证实时同步效果 201](#_Toc9628)

[74. Cobbler装机平台 201](#_Toc11737)

[74.1. Cobbler概述 201](#_Toc27577)

[74.1.1. Cobbler简介 201](#_Toc20356)

[74.2. Cobbler平台部署 203](#_Toc10317)

[74.2.1. 安装Cobbler组件 203](#_Toc15394)

[74.2.2. 配置Cobbler 203](#_Toc26119)

[75. 总结和答疑 205](#_Toc14433)

[75.1. rsync+inotify同步 205](#_Toc4262)

[75.1.1. 问题现象 205](#_Toc7049)

[75.1.2. 故障分析及排除 205](#_Toc31418)

[76. 计算机网络 206](#_Toc27850)

[76.1. 计算机网络概述 206](#_Toc31267)

[76.1.1. 什么是计算机网络 206](#_Toc12330)

[76.1.2. 计算机网络的功能 206](#_Toc6610)

[76.1.3. 计算机网络发展阶段 206](#_Toc2001)

[76.1.4. 网络标准 207](#_Toc4703)

[76.1.5. WAN与LAN 207](#_Toc13505)

[76.2. 网络设备及拓扑 208](#_Toc2152)

[76.2.1. 网络设备生产厂商 208](#_Toc6716)

[76.2.2. 路由交换设备 208](#_Toc18138)

[76.2.3. 网络拓扑结构 208](#_Toc16738)

[77. 网络通信参考模型 210](#_Toc7314)

[77.1. OSI参考模型 210](#_Toc1752)

[77.1.1. 计算机网络的功能 210](#_Toc32761)

[77.1.2. 网络分层 210](#_Toc10859)

[77.1.3. 邮局实例 210](#_Toc28807)

[77.1.4. OSI协议模型 210](#_Toc30992)

[77.2. TCP/IP模型 211](#_Toc3406)

[77.2.1. OSI模型、TCP/IP模型 211](#_Toc19382)

[77.2.2. TCP/IP协议族的组成 212](#_Toc20739)

[77.2.3. 什么是协议 212](#_Toc10685)

[77.2.4. 协议数据单元(PDU) 212](#_Toc10010)

[77.2.5. 设备与层的对应关系 212](#_Toc5139)

[78. 物理层解析 213](#_Toc27373)

[78.1. 物理层 213](#_Toc26176)

[78.1.1. 物理层-网络的基础 213](#_Toc10512)

[78.1.2. 以太网接口 213](#_Toc25845)

[78.1.3. 双绞线 213](#_Toc31800)

[78.1.4. 双绞线的标准 214](#_Toc3535)

[78.1.5. 线缆的连接 214](#_Toc27228)

[78.1.6. 物理层的设备 214](#_Toc6763)

[78.2. Packet Tracer软件 215](#_Toc3079)

[78.2.1. Packet Tracer软件简介 215](#_Toc30988)

[78.2.2. Packet Tracer软件使用 216](#_Toc22244)

[79. 交换机命令行 216](#_Toc20306)

[79.1. 交换命令行模式 216](#_Toc28180)

[79.1.1. 交换机的命令行模式 216](#_Toc32732)

[79.1.2. 模式间的转换 216](#_Toc11726)

[79.2. 交换机命令行配置 217](#_Toc12746)

[79.2.1. 配置主机名 217](#_Toc5458)

[79.2.2. 查看交换机的配置 217](#_Toc4470)

[79.2.3. 配置enable明文口令 217](#_Toc2370)

[79.2.4. 保存交换机的配置 218](#_Toc31718)

[79.2.5. 恢复设备出厂默认值 218](#_Toc31185)

[80. 总结和答疑 218](#_Toc4794)

[80.1. 自动退出配置界面 218](#_Toc4977)

[80.1.1. 问题现象 218](#_Toc25768)

[80.1.2. 原因分析 218](#_Toc5602)

[80.2. 打断命令输入 218](#_Toc30405)

[80.2.1. 问题现象 219](#_Toc3286)

[80.2.2. 原因分析 219](#_Toc7468)

[80.3. 禁用DNS查询 219](#_Toc15997)

[80.3.1. 问题现象 219](#_Toc17858)

[80.3.2. 原因分析 219](#_Toc1979)

[80.3.3. 219](#_Toc11811)

[81. 数据链路层解析 220](#_Toc23263)

[81.1. 数据链路层 220](#_Toc1538)

[81.1.1. 以太网MAC地址 220](#_Toc32000)

[81.1.2. 以太网帧格式 220](#_Toc18819)

[81.2. 以太网交换机 221](#_Toc16247)

[81.2.1. 什么是交换机 221](#_Toc15184)

[81.2.2. 交换机的工作原理 221](#_Toc7704)

[81.2.3. 查看MAC地址表 222](#_Toc25725)

[81.2.4. 交换机的工作原理案例 222](#_Toc12291)

[81.2.5. 广播域 222](#_Toc28798)

[82. VLAN技术及应用 222](#_Toc2132)

[82.1. VLAN 222](#_Toc11696)

[82.1.1. VLAN概述 222](#_Toc19703)

[82.2. VLAN配置 223](#_Toc32580)

[82.2.1. 静态VLAN的配置 223](#_Toc15716)

[82.2.2. 将接口加入VLAN 224](#_Toc13212)

[82.2.3. 验证VLAN的配置 224](#_Toc28702)

[83. TRUNK 224](#_Toc2635)

[83.1. Trunk原理 224](#_Toc1307)

[83.1.1. 交换机之间的VLAN通信 224](#_Toc20313)

[83.1.2. VLAN标识 224](#_Toc15491)

[83.1.3. VLAN标识的种类 224](#_Toc23940)

[83.2. Trunk配置 225](#_Toc7282)

[83.2.1. 配置接口为Trunk模式 225](#_Toc10120)

[83.2.2. 恢复接口默认模式 225](#_Toc26002)

[83.2.3. 查看接口模式 226](#_Toc10430)

[83.2.4. 配置VLAN Trunk实例 226](#_Toc21968)

[83.3. 以太通道 226](#_Toc24459)

[83.3.1. 以太通道概述 226](#_Toc29051)

[83.3.2. 配置以太网通道 226](#_Toc18552)

[83.3.3. 以太通道配置指导原则 227](#_Toc3720)

[84. 网络层解析 228](#_Toc18161)

[84.1. 网络层 228](#_Toc31447)

[84.1.1. 网络层的功能 228](#_Toc4339)

[84.1.2. ICMP协议 228](#_Toc7587)

[84.2. 路由原理及配置 229](#_Toc7476)

[84.2.1. 什么是路由 229](#_Toc13710)

[84.2.2. 路由器怎么工作 229](#_Toc20364)

[84.2.3. 如何获得路由表 229](#_Toc8161)

[84.2.4. 静态路由 229](#_Toc15783)

[84.2.5. 配置静态路由 230](#_Toc15096)

[84.2.6. 默认路由 231](#_Toc30281)

[85. VLAN间通讯 231](#_Toc17658)

[85.1. 三层交换概述 231](#_Toc18346)

[85.1.1. 三层交换技术 231](#_Toc24898)

[85.1.2. 虚接口概述 231](#_Toc9650)

[85.1.3. 三层交换机的配置 231](#_Toc553)

[85.2. 动态路由 233](#_Toc5920)

[85.2.1. 认识动态路由 233](#_Toc8724)

[85.2.2. 动态路由协议OSPF 233](#_Toc2274)

[85.2.3. OSPF基本配置 233](#_Toc19947)

[86. 传输层 235](#_Toc8483)

[86.1. 传输层概述 235](#_Toc21616)

[86.1.1. 传输层的作用 235](#_Toc14301)

[86.1.2. 传输层的协议 235](#_Toc18693)

[86.2. TCP协议 235](#_Toc2294)

[86.2.1. TCP的封装格式 235](#_Toc15838)

[86.2.2. TCP的连接与断开 235](#_Toc5156)

[86.2.3. TCP的应用 236](#_Toc30916)

[86.3. UDP协议 236](#_Toc314)

[86.3.1. UDP的封装格式 236](#_Toc16070)

[86.3.2. UDP的应用 236](#_Toc21383)

[86.3.3. UDP的流控和差错控制 237](#_Toc30344)

[87. ACL 237](#_Toc19813)

[87.1. 访问控制列表概述 237](#_Toc7709)

[87.1.1. 访问控制列表作用 237](#_Toc11579)

[87.1.2. 访问控制列表的工作原理 237](#_Toc14468)

[87.1.3. 访问控制列表的类型 237](#_Toc30577)

[87.2. 标准ACL配置 238](#_Toc14403)

[87.2.1. 标准访问控制列表的配置 238](#_Toc23852)

[87.2.2. 扩展访问控制列表的配置 239](#_Toc22463)

[88. NAT 239](#_Toc3537)

[88.1. NAT概述 239](#_Toc5420)

[88.1.1. NAT的作用 239](#_Toc29076)

[88.1.2. 私有ip地址分类 240](#_Toc30558)

[88.1.3. NAT的特性 240](#_Toc10687)

[88.1.4. NAT实现方式 240](#_Toc5837)

[88.1.5. NAT的工作过程 240](#_Toc12582)

[88.2. 静态转换 241](#_Toc23870)

[88.2.1. 静态NAT 241](#_Toc18555)

[88.2.2. 静态NAT的配置 241](#_Toc21860)

[88.2.3. NAT端口映射配置 241](#_Toc11069)

[88.3. 端口多路复用(PAT) 242](#_Toc15020)

[88.3.1. PAT的作用 242](#_Toc1376)

[88.3.2. PAT的配置 242](#_Toc30318)

[88.3.3. 跟踪NAT 242](#_Toc15152)

[88.3.4. 243](#_Toc17111)

[89. STP 244](#_Toc4365)

[89.1. STP生成树算法 244](#_Toc543)

[89.1.1. 广播风暴的产生 244](#_Toc14906)

[89.1.2. STP概述 244](#_Toc21667)

[89.1.3. 选择根网桥 244](#_Toc20262)

[89.2. STP配置 245](#_Toc3224)

[89.2.1. PVST+的配置命令 245](#_Toc28356)

[89.2.2. PVST+配置的意义 245](#_Toc21608)

[90. HSRP 246](#_Toc11256)

[90.1. HSRP概述 246](#_Toc17665)

[90.1.1. HSRP的相关概念 246](#_Toc12072)

[90.2. HSRP配置 247](#_Toc21408)

[90.2.1. HSRP的配置命令 247](#_Toc30048)

[90.2.2. HSRP的配置案例 247](#_Toc31448)

[90.2.3. 247](#_Toc8634)

[91. 综合项目 248](#_Toc24803)

[91.1. 目前网络环境介绍 248](#_Toc15080)

[91.1.1. 目前网络环境 248](#_Toc3464)

[91.1.2. 目前网络技术 248](#_Toc15881)

[91.2. 目前网络拓扑结构 248](#_Toc5142)

[91.2.1. 目前网络拓扑 248](#_Toc31028)

[92. 网络升级 248](#_Toc31294)

[92.1. 项目规划 248](#_Toc29538)

[92.1.1. 问题分析 248](#_Toc10975)

[92.1.2. 项目需求 248](#_Toc5277)

[92.1.3. 项目技术 248](#_Toc31736)

[92.2. 项目规划后拓扑 248](#_Toc6294)

[92.2.1. 项目拓扑 248](#_Toc18779)

[92.2.2. 248](#_Toc26395)

[93. 网络部分 249](#_Toc29151)

[93.1. 任务介绍 249](#_Toc13076)

[93.1.1. 搭建教学办公网络 249](#_Toc25960)

[93.1.2. 企业网络升级 249](#_Toc32726)

[93.2. 网络设备选项 249](#_Toc4795)

[93.2.1. 品牌与型号 249](#_Toc4141)

[93.2.2. 技术指标 249](#_Toc8034)

[94. 服务器部分 249](#_Toc7507)

[94.1. 任务目标 249](#_Toc6722)

[94.1.1. 企业内网DNS 249](#_Toc4832)

[94.1.2. PXE一键装机平台 249](#_Toc8114)

[94.1.3. 搭建邮件平台 249](#_Toc1645)

[94.1.4. Discuz!技术服务论坛 249](#_Toc4395)

[94.2. 服务器选型 249](#_Toc28331)

[94.2.1. 什么是服务器? 249](#_Toc17207)

[94.2.2. 服务器分类 250](#_Toc19931)

[94.2.3. 服务器选型参数 250](#_Toc26309)

[94.2.4. 选项原则 250](#_Toc15476)

[95. Shell概述 251](#_Toc14669)

[95.1. Shell环境及特性 251](#_Toc27522)

[95.1.1. 什么是Shell? 251](#_Toc13143)

[95.1.2. Shell的使用方式 251](#_Toc30314)

[95.1.3. 常见的Shell程序种类 251](#_Toc31660)

[95.1.4. Bash基本特性 252](#_Toc26085)

[95.2. 历史命令 252](#_Toc18665)

[95.2.1. 存放位置及数量 252](#_Toc19392)

[95.2.2. 操作历史命令 253](#_Toc19655)

[95.3. I/O设备与重定向 253](#_Toc17792)

[95.3.1. 交互式硬件设备 253](#_Toc24218)

[95.3.2. 重定向操作 254](#_Toc1751)

[96. 编写及执行脚本 254](#_Toc23445)

[96.1. 第一个Shell脚本 254](#_Toc21815)

[96.1.1. 什么是Shell脚本? 254](#_Toc24178)

[96.1.2. Shell版HelloWorld的诞生 255](#_Toc11982)

[96.2. 脚本构成及执行 255](#_Toc20032)

[96.2.1. 规范的脚本构成 255](#_Toc68)

[96.2.2. 脚本的执行方式 255](#_Toc13151)

[96.2.3. 调试Shell脚本 255](#_Toc4108)

[96.3. 简单脚本应用 256](#_Toc25527)

[96.3.1. 示例1:快速配置YUM 256](#_Toc2193)

[96.3.2. 示例2:快速搭建FTP服务 256](#_Toc25513)

[97. Shell变量 256](#_Toc14814)

[97.1. 变量的设置和取消 256](#_Toc29560)

[97.1.1. 什么是变量? 256](#_Toc31213)

[97.1.2. 定义/赋值/查看变量 256](#_Toc21746)

[97.1.3. 取消变量 257](#_Toc8233)

[97.2. 变量的种类 257](#_Toc29536)

[97.2.1. Shell变量的分类角度 257](#_Toc15896)

[97.2.2. 环境变量 257](#_Toc10250)

[97.2.3. 预定义变量 258](#_Toc1617)

[97.2.4. 位置变量 259](#_Toc26582)

[97.3. 变量值及范围控制 259](#_Toc7209)

[97.3.1. 扩展赋值操作 259](#_Toc19136)

[97.3.2. read标准输入取值 259](#_Toc6584)

[97.3.3. 变量的作用范围 260](#_Toc9286)

[98. 总结和答疑 260](#_Toc18221)

[98.1. 变量定义 260](#_Toc23544)

[98.1.1. 问题现象 260](#_Toc5506)

[98.1.2. 原因分析 260](#_Toc28625)

[98.2. 变量调用 261](#_Toc16401)

[98.2.1. 问题现象 261](#_Toc5630)

[98.2.2. 原因分析 261](#_Toc21404)

[98.3. read命令 261](#_Toc25154)

[98.3.1. 问题现象 261](#_Toc3778)

[98.3.2. 原因分析 261](#_Toc13082)

[98.3.3. 262](#_Toc28412)

[99. 数值运算 263](#_Toc3396)

[99.1. 整数运算 263](#_Toc5778)

[99.1.1. 基本运算类别 263](#_Toc101)

[99.1.2. expr运算工具 263](#_Toc23338)

[99.1.3. $[]算式替换 264](#_Toc17875)

[99.1.4. 变量的自增/减等操作 264](#_Toc2526)

[99.2. 小数运算 265](#_Toc4826)

[99.2.1. 整数运算的局限性 265](#_Toc5375)

[99.2.2. 使用bc实现小数运算 265](#_Toc8689)

[99.2.3. 小数值的比较 266](#_Toc16621)

[100. 条件测试 266](#_Toc21168)

[100.1. 测试操作概述 266](#_Toc20652)

[100.1.1. Shell脚本的智能化 266](#_Toc32410)

[100.1.2. test测试操作 266](#_Toc30748)

[100.2. 常见的测试操作 267](#_Toc20712)

[100.2.1. 整数值比较 267](#_Toc18687)

[100.2.2. 字符串比较 268](#_Toc8697)

[100.2.3. 文件状态测试 268](#_Toc14648)

[100.3. 组合多个条件 269](#_Toc6946)

[100.3.1. 逻辑分隔操作 269](#_Toc5090)

[100.3.2. 条件组合应用 270](#_Toc19932)

[101. if选择结构 270](#_Toc29740)

[101.1. 单分支结构 270](#_Toc436)

[101.1.1. 语法格式及特点 270](#_Toc8229)

[101.1.2. if单分支的执行流程 270](#_Toc22118)

[101.1.3. 脚本应用示例 270](#_Toc2780)

[101.2. 双分支结构 270](#_Toc1021)

[101.2.1. 语法结构及特点 270](#_Toc16509)

[101.2.2. if双分支的执行流程 271](#_Toc4355)

[101.2.3. 脚本应用示例 271](#_Toc21584)

[101.3. 多分支结构 272](#_Toc22263)

[101.3.1. 语法格式及特点 272](#_Toc26572)

[101.3.2. if多分支的执行流程 272](#_Toc12193)

[101.3.3. 脚本应用示例 272](#_Toc23317)

[102. 总结和答疑 273](#_Toc14860)

[102.1. 算术运算 273](#_Toc31394)

[102.1.1. 问题现象 273](#_Toc23582)

[102.1.2. 原因分析 273](#_Toc23264)

[102.2. 条件判断 273](#_Toc3627)

[102.2.1. 问题现象 273](#_Toc23044)

[102.2.2. 原因分析 274](#_Toc10236)

[102.2.3. 274](#_Toc16757)

[103. 循环结构 275](#_Toc25370)

[103.1. for循环 275](#_Toc2925)

[103.1.1. 语法结构及特点 275](#_Toc9308)

[103.1.2. for循环的执行流程 275](#_Toc14023)

[103.1.3. 脚本应用示例 275](#_Toc8723)

[103.2. while循环 275](#_Toc13866)

[103.2.1. 语法结构及特点 275](#_Toc28769)

[103.2.2. while循环的执行流程 276](#_Toc9337)

[103.2.3. 脚本应用示例 276](#_Toc31381)

[104. case语句 278](#_Toc21508)

[104.1. case分支结构 278](#_Toc23075)

[104.1.1. 语法结构及特点 278](#_Toc12527)

[104.1.2. case分支的执行流程 278](#_Toc20418)

[104.1.3. 脚本应用示例 278](#_Toc11666)

[105. 函数及中断控制 280](#_Toc14716)

[105.1. Shell函数 280](#_Toc21953)

[105.1.1. 关于函数的认识 280](#_Toc31786)

[105.1.2. 函数的定义与调用 280](#_Toc26417)

[105.1.3. 函数应用示例 281](#_Toc25030)

[105.2. 中断及退出 282](#_Toc31680)

[105.2.1. 中断/退出及相关指令 282](#_Toc6805)

[105.2.2. 脚本中断示例 283](#_Toc6653)

[105.2.3. 脚本继续示例 283](#_Toc24958)

[105.2.4. 脚本退出示例 283](#_Toc28030)

[106. 总结和答疑 283](#_Toc18145)

[106.1. for循环 283](#_Toc5058)

[106.1.1. 问题现象 283](#_Toc28278)

[106.1.2. 原因分析 284](#_Toc29955)

[106.2. case语句 284](#_Toc19787)

[106.2.1. 问题现象 284](#_Toc12504)

[106.2.2. 原因分析 284](#_Toc28787)

[106.3. 启动脚本 285](#_Toc9359)

[106.3.1. 问题现象 285](#_Toc23139)

[106.3.2. 原因分析 285](#_Toc420)

[106.3.3. 285](#_Toc13333)

[107. 字符串处理 286](#_Toc23760)

[107.1. 子串截取 286](#_Toc14442)

[107.1.1. 子串截取的方法 286](#_Toc8813)

[107.1.2. 子串截取示例 286](#_Toc13148)

[107.2. 子串替换 287](#_Toc855)

[107.2.1. 子串替换的方法 287](#_Toc31743)

[107.2.2. 子串替换示例 287](#_Toc17772)

[107.3. 按条件掐头去尾 288](#_Toc18282)

[107.3.1. 字符串掐头 288](#_Toc32077)

[107.3.2. 字符串去尾 288](#_Toc25595)

[107.3.3. 脚本应用示例 289](#_Toc8994)

[107.4. 变量初始值处理 289](#_Toc10123)

[107.4.1. 初值的检测及设置 289](#_Toc16994)

[107.4.2. 脚本应用示例 290](#_Toc28825)

[108. 扩展的脚本技巧 290](#_Toc17647)

[108.1. Shell- 290](#_Toc22898)

[108.1.1. 关于变量的类型 290](#_Toc7175)

[108.1.2. 定义/赋值数组 290](#_Toc30170)

[108.1.3. 输出数组元素 291](#_Toc24031)

[108.2. expect预期交互 291](#_Toc30406)

[108.2.1. expect简介 291](#_Toc25821)

[108.2.2. expect应用示例 292](#_Toc11559)

[109. 正则表达式 292](#_Toc9671)

[109.1. 正则表达式概述 292](#_Toc27613)

[109.1.1. 什么是正则表达式? 292](#_Toc23661)

[109.1.2. egrep过滤工具 293](#_Toc2616)

[109.2. 基本元字符 293](#_Toc27281)

[109.2.1. 行首尾及单字匹配 293](#_Toc5063)

[109.2.2. 未定匹配次数 294](#_Toc25369)

[109.2.3. {}限定次数 295](#_Toc22326)

[109.3. 其他元字符g 295](#_Toc19412)

[109.3.1. []范围内单字匹配 295](#_Toc21902)

[109.3.2. 整体及边界匹配 296](#_Toc12882)

[110. sed基本用法 297](#_Toc23580)

[110.1. sed命令解析 297](#_Toc21672)

[110.1.1. sed工具概述 297](#_Toc9916)

[110.1.2. 命令格式解析 297](#_Toc25868)

[110.1.3. 基本的处理动作 298](#_Toc5475)

[110.2. 常见处理操作示例 298](#_Toc5918)

[110.2.1. 输出文本 298](#_Toc25711)

[110.2.2. 删除文本 299](#_Toc12696)

[110.2.3. 替换文本 299](#_Toc18888)

[110.3. sed应用案例 300](#_Toc3280)

[110.3.1. 修改IP配置 300](#_Toc12259)

[110.3.2. 修改网站根目录 300](#_Toc5465)

[110.3.3. 使用sed脚本 300](#_Toc32601)

[111. sed文本块处理 301](#_Toc7903)

[111.1. 文本块概述 301](#_Toc25003)

[111.1.1. 文本块的应用场景 301](#_Toc18267)

[111.1.2. sed的文本块处理动作 301](#_Toc5504)

[111.2. 文本块处理的应用 302](#_Toc20770)

[111.2.1. 处理单行文本 302](#_Toc251)

[111.2.2. 处理多行文本 302](#_Toc2659)

[112. sed高级应用 302](#_Toc8127)

[112.1. 文件导入导出 302](#_Toc5538)

[112.1.1. 导入导出操作 302](#_Toc29188)

[112.1.2. sed复制剪切 303](#_Toc17133)

[112.1.3. 复制剪切操作示例 303](#_Toc32111)

[112.2. sed流控制 303](#_Toc26596)

[112.2.1. !取反操作 303](#_Toc13705)

[112.3. sed综合脚本示例 304](#_Toc21700)

[112.3.1. 任务需求 304](#_Toc4357)

[112.3.2. 思路讲解及分析 304](#_Toc14106)

[113. 总结和答疑 305](#_Toc31295)

[113.1. sed替换 305](#_Toc32463)

[113.1.1. 问题现象 305](#_Toc31891)

[113.1.2. 原因分析 305](#_Toc16539)

[113.1.3. 305](#_Toc23929)

[114. awk基本用法 306](#_Toc10542)

[114.1. awk命令解析 306](#_Toc1938)

[114.1.1. awk工具概述 306](#_Toc28305)

[114.1.2. 命令格式解析 306](#_Toc7940)

[114.1.3. awk内置变量 306](#_Toc22918)

[114.1.4. awk过滤的时机 308](#_Toc4574)

[114.2. awk处理条件 309](#_Toc3825)

[114.2.1. 处理条件概述 309](#_Toc19536)

[114.2.2. 条件设置示例 309](#_Toc16792)

[114.2.3. 多个条件的组合 310](#_Toc14338)

[114.2.4. 变量的运算 311](#_Toc30385)

[114.3. awk综合脚本示例 311](#_Toc9937)

[114.3.1. 任务需求 311](#_Toc7193)

[114.3.2. 思路讲解及分析 311](#_Toc3254)

[115. awk高级应用 311](#_Toc25909)

[115.1. awk流程控制 311](#_Toc18853)

[115.1.1. 分支结构 311](#_Toc19389)

[115.1.2. 循环结构 312](#_Toc2982)

[115.1.3. 其他控制语句 312](#_Toc15370)

[115.2. awk数组 312](#_Toc21111)

[115.2.1. 数组的定义及使用 312](#_Toc28822)

[115.2.2. awk数组的经典应用 313](#_Toc10613)

[115.3. awk案例分析 313](#_Toc24278)

[115.3.1. 任务需求 313](#_Toc4515)

[115.3.2. 思路讲解及分析 313](#_Toc26524)

[116. 总结和答疑 314](#_Toc9474)

[116.1. awk分隔符 314](#_Toc20367)

[116.1.1. 问题现象 314](#_Toc1435)

[116.1.2. 原因分析 314](#_Toc26765)

[116.2. awk引号 315](#_Toc32234)

[116.2.1. 问题现象 315](#_Toc23276)

[116.2.2. 原因分析 315](#_Toc9609)

[117. 项目实战:脚本 316](#_Toc13815)

[117.1. 一键部署软件脚本 316](#_Toc13390)

[117.1.1. 任务需求 316](#_Toc26905)

[117.1.2. 思路讲解及分析 316](#_Toc16182)

[117.2. 启动脚本 317](#_Toc5010)

[117.2.1. 任务需求 317](#_Toc13782)

[117.2.2. 思路讲解及分析 317](#_Toc18095)

[117.3. 监控脚本 317](#_Toc27690)

[117.3.1. 任务需求 317](#_Toc30420)

[117.3.2. 思路讲解及分析 318](#_Toc7157)

[117.4. 安全检测脚本 319](#_Toc31203)

[117.4.1. 任务需求 319](#_Toc25109)

[117.4.2. 思路讲解及分析 319](#_Toc9664)

[117.5. 进度显示脚本 320](#_Toc8047)

[117.5.1. 任务需求 320](#_Toc30718)

[117.5.2. 思路讲解及分析 320](#_Toc14335)

[118. Nginx安装与升级 323](#_Toc25881)

[118.1. Nginx安装 323](#_Toc20237)

[118.1.1. Web服务器对比 323](#_Toc19801)

[118.1.2. Nginx简介 323](#_Toc11825)

[118.1.3. Nginx进程管理 324](#_Toc2016)

[118.2. Nginx升级 325](#_Toc6105)

[118.2.1. 平滑升级Nginx 325](#_Toc7118)

[119. Nginx服务器 326](#_Toc4997)

[119.1. Nginx配置解析 326](#_Toc1037)

[119.1.1. 配置文件结构 326](#_Toc17433)

[119.1.2. 全局选项 326](#_Toc17655)

[119.1.3. 配置容器 327](#_Toc23943)

[119.1.4. 用户认证 327](#_Toc26568)

[120. Nginx虚拟主机 328](#_Toc7399)

[120.1. 三种模式虚拟主机 328](#_Toc21211)

[120.1.1. 基于域名的虚拟主机 328](#_Toc32314)

[120.1.2. 基于端口的虚拟主机 328](#_Toc27866)

[120.1.3. 基于IP的虚拟主机 329](#_Toc1393)

[121. HTTPS加密网站 329](#_Toc10810)

[121.1. 密钥 329](#_Toc23655)

[121.1.1. 对称密钥 330](#_Toc24503)

[121.1.2. 非对称密钥 330](#_Toc22001)

[121.1.3. Hash值 330](#_Toc18871)

[121.2. SSL虚拟主机 330](#_Toc9780)

[121.2.1. 生成密钥 330](#_Toc30651)

[121.2.2. SSL加密网站 331](#_Toc25311)

[122. 总结和答疑 331](#_Toc1085)

[122.1. 软件依赖 331](#_Toc12792)

[122.1.1. 问题现象 331](#_Toc1204)

[122.1.2. 原因分析 331](#_Toc1124)

[122.2. 配置文件错误 332](#_Toc15003)

[122.2.1. 问题现象 332](#_Toc24910)

[122.2.2. 原因分析 332](#_Toc5537)

[122.2.3. 333](#_Toc12819)

[123. 部署LNMP 334](#_Toc22641)

[123.1. LNMP基础知识 334](#_Toc14953)

[123.1.1. 什么是LNMP 334](#_Toc31929)

[123.2. LNMP安装 334](#_Toc1748)

[123.2.1. 安装Nginx 334](#_Toc6224)

[123.2.2. 安装MairaDB 334](#_Toc11175)

[123.2.3. 安装PHP 334](#_Toc21679)

[123.2.4. 安装PHP扩展 334](#_Toc16672)

[123.2.5. 启动服务 335](#_Toc21277)

[124. Nginx+FastCGI 335](#_Toc11624)

[124.1. FastCGI 335](#_Toc27445)

[124.1.1. FastCGI工作原理 335](#_Toc4980)

[124.1.2. FastCGI简介 336](#_Toc24246)

[124.1.3. FastCGI缺点 336](#_Toc13856)

[124.1.4. 配置FastCGI 336](#_Toc13503)

[124.2. Nginx 338](#_Toc22783)

[124.2.1. 编辑nginx.conf 338](#_Toc16344)

[124.2.2. 测试 338](#_Toc11657)

[125. Nginx高级技术 339](#_Toc670)

[125.1. 地址重写 339](#_Toc13121)

[125.1.1. 基础知识 339](#_Toc1249)

[125.1.2. rewrite语法 339](#_Toc19419)

[125.1.3. 应用案例 340](#_Toc20491)

[125.1.4. 正则表达式 340](#_Toc21009)

[125.1.5. rewrite选项 340](#_Toc4715)

[126. 总结和答疑 341](#_Toc26508)

[126.1. 动态页面故障 341](#_Toc10578)

[126.1.1. 问题现象 341](#_Toc21878)

[126.1.2. 原因分析 341](#_Toc7637)

[126.1.3. 342](#_Toc15045)

[127. Nginx调度器 343](#_Toc11396)

[127.1. HTTP调度 343](#_Toc15752)

[127.1.1. Nginx反向代理架构 343](#_Toc22499)

[127.1.2. Nginx反向代理语法格式 343](#_Toc24583)

[127.1.3. Nginx调度算法 344](#_Toc17326)

[127.1.4. 服务器组主机状态 344](#_Toc24434)

[127.1.5. 案例应用 345](#_Toc18217)

[127.2. TCP/UDP调度 345](#_Toc29039)

[127.2.1. 模块 345](#_Toc22284)

[127.2.2. 语法格式 345](#_Toc7891)

[127.2.3. 案例应用 346](#_Toc11222)

[128. Nginx优化 346](#_Toc11577)

[128.1. HTTP错误代码 346](#_Toc28539)

[128.1.1. HTTP常见错误代码列表 346](#_Toc22116)

[128.1.2. Nginx如何返回错误页面 347](#_Toc21969)

[128.2. Nginx状态页面 347](#_Toc17284)

[128.2.1. status模块 347](#_Toc23853)

[128.2.2. status页面 348](#_Toc13274)

[128.2.3. 状态信息 348](#_Toc8455)

[128.3. 配置优化 349](#_Toc12628)

[128.3.1. 常用压力测试工具 349](#_Toc13984)

[128.3.2. 全局配置优化 349](#_Toc2830)

[128.3.3. EVENT模块优化 349](#_Toc3075)

[128.3.4. HTTP模块优化 350](#_Toc19677)

[128.3.5. 日志切割 352](#_Toc17361)

[129. 总结和答疑 353](#_Toc25691)

[129.1. 调度器故障 353](#_Toc7359)

[129.1.1. 问题现象 353](#_Toc26367)

[129.1.2. 原因分析 353](#_Toc10669)

[129.1.3. 353](#_Toc19015)

[130. memcached原理 354](#_Toc1977)

[130.1. 基础知识 354](#_Toc9741)

[130.1.1. 传统Web架构的问题 354](#_Toc27080)

[130.1.2. 数据存储位置对比 354](#_Toc19786)

[130.2. memcached概述 354](#_Toc32210)

[130.2.1. memcached简介 354](#_Toc2641)

[130.2.2. 内存管理机制 355](#_Toc23932)

[131. 部署memcached 356](#_Toc16691)

[131.1. 安装软件 356](#_Toc26838)

[131.1.1. 安装memcached 356](#_Toc31302)

[131.1.2. 启动memcached 356](#_Toc27362)

[131.2. 测试memcached 357](#_Toc12409)

[131.2.1. 查看memcached的内部状态 357](#_Toc18141)

[131.2.2. telnet测试memcached 357](#_Toc1508)

[131.3. PHP+Memcached 358](#_Toc23339)

[131.3.1. PHP+memcached介绍 358](#_Toc6516)

[131.3.2. 安装PHP的memcached扩展 358](#_Toc1201)

[131.3.3. 编写PHP页面 359](#_Toc3590)

[132. Session共享 359](#_Toc10786)

[132.1. 基本概念 359](#_Toc28961)

[132.1.1. Session&cookies 359](#_Toc32734)

[132.1.2. 案例拓扑 360](#_Toc18278)

[132.2. 本地Session 360](#_Toc30913)

[132.2.1. 部署Nginx调度器 360](#_Toc25418)

[132.2.2. 部署后端LNMP主机 360](#_Toc9368)

[132.2.3. 部署测试页面 361](#_Toc10209)

[132.2.4. 客户端访问 361](#_Toc10908)

[132.2.5. 验证Session 361](#_Toc5653)

[132.3. 实现Session共享 362](#_Toc27742)

[132.3.1. 修改PHP配置文件 362](#_Toc1110)

[132.3.2. 重启服务 362](#_Toc32037)

[132.3.3. 测试Session共享 362](#_Toc23809)

[133. 总结和答疑 362](#_Toc31047)

[133.1. PHP连接memcached 362](#_Toc24787)

[133.1.1. 问题现象 362](#_Toc25924)

[133.1.2. 原因分析 363](#_Toc5667)

[133.2. 无法共享session 363](#_Toc15012)

[133.2.1. 问题现象 363](#_Toc4790)

[133.2.2. 原因分析 363](#_Toc14575)

[134. Tomcat服务器 365](#_Toc32107)

[134.1. 概述 365](#_Toc820)

[134.1.1. Java简介 365](#_Toc19986)

[134.1.2. JDK简介 365](#_Toc5883)

[134.1.3. Java Servlet 366](#_Toc17704)

[134.2. 安装Tomcat 366](#_Toc20375)

[134.2.1. 安装部署JDK 366](#_Toc8788)

[134.2.2. 安装部署Tomcat 366](#_Toc305)

[134.2.3. 测试服务器 367](#_Toc6094)

[135. Tomcat应用案例 367](#_Toc20741)

[135.1. Tomcat虚拟主机 367](#_Toc10790)

[135.1.1. 主配置文件解析 367](#_Toc23717)

[135.1.2. 基于域名的虚拟主机 368](#_Toc4160)

[135.1.3. Tomcat工作原理 369](#_Toc29331)

[135.1.4. SSL加密站点 369](#_Toc19523)

[135.1.5. 日志 369](#_Toc11177)

[135.2. Tomcat集群 370](#_Toc15559)

[135.2.1. Nginx反向代理tomcat 370](#_Toc23916)

[136. Varnish代理服务器 370](#_Toc6571)

[136.1. 代理服务器概述 370](#_Toc31703)

[136.1.1. 基本概念 370](#_Toc31821)

[136.1.2. Varnish服务器 370](#_Toc747)

[136.2. 部署Varnish 371](#_Toc8255)

[136.2.1. 编译安装 371](#_Toc6442)

[136.2.2. Varnish服务器 371](#_Toc20073)

[136.3. 加速服务器 371](#_Toc22167)

[136.3.1. 快速部署Varnish 371](#_Toc28281)

[136.3.2. 缓存管理 372](#_Toc22671)

[136.3.3. Varnish日志 372](#_Toc4967)

[136.3.4. Varnish状态 372](#_Toc4483)

[137. 总结和答疑 373](#_Toc9654)

[137.1. Java环境变量 373](#_Toc11839)

[137.1.1. 问题现象 373](#_Toc13098)

[137.1.2. 原因分析 373](#_Toc16129)

[137.2. Tomcat配置文件错误 373](#_Toc10531)

[137.2.1. 问题现象 373](#_Toc27004)

[137.2.2. 原因分析 374](#_Toc29990)

[137.3. 端口冲突 374](#_Toc20785)

[137.3.1. 问题现象 374](#_Toc20580)

[137.3.2. 原因分析 374](#_Toc10112)

[137.3.3. 375](#_Toc14336)

[138. 版本控制 376](#_Toc18743)

[138.1. Subversion 376](#_Toc223)

[138.1.1. 什么是Subversion 376](#_Toc15790)

[138.1.2. Subversion架构 376](#_Toc10972)

[138.2. 基本概念 377](#_Toc19378)

[138.2.1. 版本库 377](#_Toc30338)

[138.2.2. 版本模型 377](#_Toc15048)

[139. SVN基础 378](#_Toc6596)

[139.1. 服务器设置 378](#_Toc19347)

[139.1.1. 安装Subversion 378](#_Toc4940)

[139.1.2. 创建版本库 378](#_Toc2146)

[139.1.3. 导入项目代码 379](#_Toc22862)

[139.1.4. 认证与授权 379](#_Toc22319)

[139.1.5. svnserve服务 380](#_Toc16535)

[139.2. 客户端操作 380](#_Toc5497)

[139.2.1. 访问方式 382](#_Toc15251)

[139.2.2. 客户端命令行工具 382](#_Toc31348)

[139.2.3. Windows客户端软件 385](#_Toc3066)

[140. 实战案例 385](#_Toc2009)

[140.1. 多人协同案例 385](#_Toc29207)

[140.1.1. 创建项目 385](#_Toc3135)

[140.1.2. 初始化数据 385](#_Toc7119)

[140.1.3. 修改副本 386](#_Toc22772)

[140.1.4. 解决冲突 386](#_Toc3126)

[140.2. 备份与还原 387](#_Toc26580)

[140.2.1. 备份版本库 387](#_Toc5244)

[140.2.2. 还原版本库 387](#_Toc12523)

[140.3. GitHub 387](#_Toc4508)

[140.3.1. 简介 387](#_Toc13818)

[140.3.2. 使用GitHub 388](#_Toc31802)

[141. RPM打包 388](#_Toc15197)

[141.1. 基本概念 388](#_Toc1153)

[141.1.1. 应用场景 388](#_Toc8416)

[141.2. 软件打包 389](#_Toc528)

[141.2.1. 打包流程 389](#_Toc8840)

[141.2.2. 安装rpm-build 389](#_Toc54)

[141.2.3. 准备源码软件 389](#_Toc18526)

[141.2.4. 编写编译配置文件 389](#_Toc12546)

[141.2.5. 编译RPM包 391](#_Toc17841)

[142. 总结和答疑 391](#_Toc9725)

[142.1. 无法检出数据 391](#_Toc19324)

[142.1.1. 问题现象 391](#_Toc25478)

[142.1.2. 原因分析 392](#_Toc11974)

[142.2. 创建或删除文件 392](#_Toc19486)

[142.2.1. 问题现象 392](#_Toc2871)

[142.2.2. 原因分析 392](#_Toc2883)

[142.2.3. 393](#_Toc9015)

[143. VPN服务器 394](#_Toc2692)

[143.1. GRE VPN 394](#_Toc8397)

[143.1.1. VPN概述 394](#_Toc11346)

[143.1.2. 拓扑图 394](#_Toc17876)

[143.1.3. GRE模块 395](#_Toc3983)

[143.1.4. 创建VPN隧道 395](#_Toc20557)

[143.1.5. 测试连通性 396](#_Toc21046)

[143.2. PPTP VPN 396](#_Toc26175)

[143.2.1. 概述 396](#_Toc27205)

[143.2.2. 拓扑图 397](#_Toc13693)

[143.2.3. 部署VPN服务器 397](#_Toc18153)

[143.2.4. Windows客户端 398](#_Toc3738)

[143.3. L2TP+IPSec VPN 398](#_Toc1049)

[143.3.1. 概述 398](#_Toc15515)

[143.3.2. 拓扑图 398](#_Toc3076)

[143.3.3. 部署L2TP+IPSec服务器 399](#_Toc10026)

[143.3.4. Windows客户端 400](#_Toc7426)

[144. NTP时间同步 401](#_Toc31566)

[144.1. 概述 401](#_Toc8160)

[144.1.1. NTP协议概述 401](#_Toc14070)

[144.1.2. 软件包 402](#_Toc22979)

[144.2. NTP服务器 402](#_Toc5597)

[144.2.1. 修改配置文件 402](#_Toc7675)

[144.2.2. 部署服务器 402](#_Toc4753)

[144.3. NTP客户端 403](#_Toc1904)

[144.3.1. 配置文件 403](#_Toc1183)

[144.3.2. 验证时间 403](#_Toc16029)

[145. PSSH远程工具 403](#_Toc27956)

[145.1. 概述 403](#_Toc19891)

[145.1.1. PSSH简介 403](#_Toc31163)

[145.2. 应用案例 404](#_Toc15)

[145.2.1. 准备工作 404](#_Toc7482)

[145.2.2. pssh远程连接 404](#_Toc1776)

[145.2.3. pscp.pssh远程拷贝 405](#_Toc20763)

[145.2.4. pslurp远程下载数据 406](#_Toc15329)

[145.2.5. pnuke远程杀死进程 406](#_Toc16941)

[146. 总结和答疑 406](#_Toc13343)

[146.1. VPN连接错误 406](#_Toc1465)

[146.1.1. 问题现象 406](#_Toc14535)

[146.1.2. 原因分析 407](#_Toc3454)

[146.2. 时间不同步 407](#_Toc21864)

[146.2.1. 问题现象 407](#_Toc15466)

[146.2.2. 原因分析 407](#_Toc22609)

[147. 存储技术与应用 408](#_Toc10824)

[147.1. 存储概述 408](#_Toc14859)

[147.1.1. 存储的目标 408](#_Toc21951)

[147.1.2. 存储技术分类 408](#_Toc16205)

[147.2. 常见存储技术 409](#_Toc27473)

[147.2.1. SCSI技术 409](#_Toc32044)

[147.2.2. DAS技术 409](#_Toc283)

[147.2.3. NAS技术 410](#_Toc5948)

[147.2.4. SAN技术 410](#_Toc17345)

[147.2.5. ISCSI技术 411](#_Toc13161)

[148. iSCSI技术应用 412](#_Toc32502)

[148.1. 基础知识 412](#_Toc7727)

[148.1.1. iSCSI操作规范 412](#_Toc1686)

[148.1.2. iSCSI命名规范 413](#_Toc22553)

[148.2. 部署iSCSI服务 413](#_Toc27800)

[148.2.1. 安装target软件 413](#_Toc22648)

[148.2.2. 配置iSCSI Target 414](#_Toc2929)

[148.2.3. 服务管理 414](#_Toc30378)

[148.2.4. 安装客户端软件 414](#_Toc12483)

[148.2.5. 客户端访问并验证 415](#_Toc13981)

[149. Multipath多路径 415](#_Toc21068)

[149.1. 基础知识 415](#_Toc23164)

[149.1.1. 多路径概述 415](#_Toc18279)

[149.1.2. 多路径主要功能 416](#_Toc27297)

[149.2. 配置案例 417](#_Toc5039)

[149.2.1. 拓扑规划 417](#_Toc11845)

[149.2.2. 准备共享存储 417](#_Toc29556)

[149.2.3. 访问共享存储 417](#_Toc8837)

[149.2.4. 多路径设备 417](#_Toc8486)

[149.2.5. 多路径配置概述 418](#_Toc10932)

[149.2.6. 多路径设备识别符 418](#_Toc17132)

[149.2.7. 获取WWID 418](#_Toc19471)

[149.2.8. 指定获取WWID的方法 419](#_Toc10738)

[149.2.9. 为多路径设备配置别名 419](#_Toc6228)

[149.2.10. 启动服务并验证 419](#_Toc6460)

[150. NFS网络文件系统 420](#_Toc17866)

[150.1. NFS服务基础 420](#_Toc10420)

[150.1.1. 文件系统的类型 420](#_Toc13260)

[150.1.2. NFS共享协议 420](#_Toc8526)

[150.2. 配置并访问NFS共享 421](#_Toc30998)

[150.2.1. 配置NFS服务器 421](#_Toc27333)

[150.2.2. 使用NFS客户端 421](#_Toc23765)

[151. udev配置 423](#_Toc23233)

[151.1. 基础知识 423](#_Toc10324)

[151.1.1. 设备文件管理方法 423](#_Toc24195)

[151.1.2. 接入设备事件链 423](#_Toc16499)

[151.2. udev的作用 424](#_Toc10654)

[151.2.1. udev应用 424](#_Toc11227)

[151.2.2. udev时间监控 424](#_Toc27607)

[151.2.3. 配置udev 424](#_Toc6139)

[151.2.4. udev操作符 425](#_Toc20149)

[151.2.5. udev变量 426](#_Toc15047)

[151.2.6. udev案例分析 426](#_Toc16572)

[152. 总结和答疑 426](#_Toc8078)

[152.1. 客户端检测不到磁盘 426](#_Toc884)

[152.1.1. 问题现象 426](#_Toc8036)

[152.1.2. 故障分析及排除 426](#_Toc23202)

[152.2. NFS无法写入数据 427](#_Toc2542)

[152.2.1. 问题现象 427](#_Toc28260)

[152.2.2. 故障分析及排除 427](#_Toc27038)

[152.2.3. 427](#_Toc16143)

[153. 集群及LVS简介 428](#_Toc26257)

[153.1. 集群简介 428](#_Toc16630)

[153.1.1. 什么是集群 428](#_Toc9656)

[153.1.2. 集群目的 428](#_Toc10124)

[153.1.3. 集群分类 428](#_Toc11764)

[153.2. LVS概述 429](#_Toc25216)

[153.2.1. LVS项目介绍 429](#_Toc27457)

[153.2.2. LVS集群组成 429](#_Toc9407)

[153.2.3. LVS术语 429](#_Toc26674)

[153.2.4. LVS工作模式 430](#_Toc19587)

[153.2.5. 负载均衡调度算法 431](#_Toc22596)

[154. LVS-NAT集群 432](#_Toc31957)

[154.1. 软件安装 432](#_Toc8661)

[154.1.1. 安装前准备 432](#_Toc31791)

[154.1.2. 安装ipvsadm 432](#_Toc19648)

[154.1.3. ipvsadm用法 432](#_Toc18723)

[154.2. LVS-NAT案例 434](#_Toc8940)

[154.2.1. LVS-NAT集群拓扑 434](#_Toc7465)

[154.2.2. 操作流程 434](#_Toc31319)

[154.2.3. 部署LVS调度器 434](#_Toc5900)

[154.2.4. Client端验证 435](#_Toc7262)

[155. LVS-DR集群 436](#_Toc20875)

[155.1. 案例说明 436](#_Toc7242)

[155.1.1. LVS-DR集群拓扑 436](#_Toc4133)

[155.1.2. 操作流程 436](#_Toc16253)

[155.1.3. ARP广播的问题 437](#_Toc15149)

[155.1.4. 内核参数说明 437](#_Toc27557)

[155.1.5. ARP防火墙 438](#_Toc28055)

[155.2. LVS-DR案例实施 438](#_Toc29662)

[155.2.1. 配置后端Web服务器 438](#_Toc10787)

[155.2.2. 部署LVS调度器 439](#_Toc9143)

[155.2.3. Client端验证 440](#_Toc8041)

[156. 总结和答疑 441](#_Toc5254)

[156.1. 加权轮询不生效 441](#_Toc11055)

[156.1.1. 问题现象 441](#_Toc13484)

[156.1.2. 故障分析及排除 441](#_Toc15224)

[156.2. 服务无法启动 441](#_Toc13056)

[156.2.1. 问题现象 441](#_Toc1942)

[156.2.2. 故障分析及排除 442](#_Toc31604)

[156.3. 无法访问网页 442](#_Toc8546)

[156.3.1. 问题现象 442](#_Toc26547)

[156.3.2. 故障分析及排除 442](#_Toc30306)

[156.3.3. 442](#_Toc18335)

[157. Keepalived热备 443](#_Toc13696)

[157.1. Keepalived概述 443](#_Toc31425)

[157.1.1. Keepalived概述 443](#_Toc9024)

[157.1.2. Keepalived运行原理 443](#_Toc4652)

[157.2. Keepalived服务 443](#_Toc22688)

[157.2.1. Keepalived安装 443](#_Toc9807)

[157.2.2. 配置文件解析 444](#_Toc2575)

[157.2.3. 高可用Web拓扑 445](#_Toc10388)

[157.2.4. 高可用Web案例 445](#_Toc7974)

[158. Keepalived+LVS 445](#_Toc30398)

[158.1. 高可用调度器 445](#_Toc26877)

[158.1.1. Keepalived+LVS拓扑 445](#_Toc26205)

[158.1.2. Keepalived配置说明 446](#_Toc29780)

[158.1.3. Real Server配置 446](#_Toc28892)

[158.2. 集群调度软件对比 446](#_Toc7661)

[158.2.1. Nginx分析 446](#_Toc11500)

[158.2.2. LVS分析 447](#_Toc712)

[158.2.3. HAProxy分析 447](#_Toc26715)

[159. HAProxy服务器 448](#_Toc29469)

[159.1. HAProxy概述 448](#_Toc3483)

[159.1.1. HAProxy简介 448](#_Toc1606)

[159.1.2. 衡量负责均衡器性能的因素 449](#_Toc6786)

[159.1.3. HAProxy工作模式 449](#_Toc7941)

[159.2. HAProxy配置示例 450](#_Toc23051)

[159.2.1. 项目拓扑图 450](#_Toc27320)

[159.2.2. HAProxy安装 450](#_Toc24866)

[159.2.3. 配置文件说明 450](#_Toc31618)

[159.2.4. 管理服务 451](#_Toc3772)

[159.2.5. 监控HAProxy状态 451](#_Toc23667)

[160. 总结和答疑 451](#_Toc4556)

[160.1. Keepalived产生大量日志 451](#_Toc14387)

[160.1.1. 问题现象 451](#_Toc19418)

[160.1.2. 故障分析及排除 452](#_Toc11708)

[160.2. LVS规则不完整 452](#_Toc23873)

[160.2.1. 问题现象 452](#_Toc3107)

[160.2.2. 故障分析及排除 452](#_Toc20570)

[161. Ceph概述 453](#_Toc13707)

[161.1. 基础知识 453](#_Toc17194)

[161.1.1. 什么是分布式文件系统 453](#_Toc10921)

[161.1.2. 常用分布式文件系统 453](#_Toc11078)

[161.1.3. 什么是Ceph 453](#_Toc2959)

[161.1.4. Ceph组件 453](#_Toc28169)

[161.2. 实验环境准备 453](#_Toc32653)

[161.2.1. 实验拓扑图 453](#_Toc5973)

[161.2.2. 配置YUM 453](#_Toc17777)

[161.2.3. 配置SSH无密钥连接 453](#_Toc2928)

[161.2.4. NTP时间同步 453](#_Toc30316)

[161.2.5. 准备存储磁盘 453](#_Toc29809)

[162. 部署Ceph集群 453](#_Toc20830)

[162.1. 准备部署环境 453](#_Toc5455)

[162.1.1. 安装部署软件 453](#_Toc2495)

[162.1.2. 创建目录 454](#_Toc14933)

[162.2. 部署存储集群 454](#_Toc21205)

[162.2.1. 创建Ceph集群 454](#_Toc20154)

[162.2.2. 创建OSD 454](#_Toc32330)

[162.2.3. 验证 454](#_Toc27536)

[163. Ceph块存储 454](#_Toc16073)

[163.1. 概述 454](#_Toc1839)

[163.1.1. 什么是块存储 454](#_Toc28223)

[163.2. 块存储集群 454](#_Toc9461)

[163.2.1. 创建镜像 454](#_Toc30482)

[163.2.2. 动态调整大小 454](#_Toc12335)

[163.2.3. 集群内通过KRBD访问 454](#_Toc32748)

[163.2.4. 客户端通过KRBD访问 454](#_Toc17191)

[163.2.5. 创建镜像快照 454](#_Toc14101)

[163.2.6. 使用快照恢复数据 454](#_Toc25503)

[163.2.7. 快照克隆 455](#_Toc16755)

[163.2.8. 客户端撤销磁盘映射 455](#_Toc18512)

[163.2.9. 删除快照与镜像 455](#_Toc12962)

[164. 总结和答疑 455](#_Toc5916)

[164.1. 还原快照后无法挂载 455](#_Toc24268)

[164.1.1. 问题现象 455](#_Toc18965)

[164.1.2. 故障分析及排除 455](#_Toc7197)

[164.1.3. 455](#_Toc6542)

[165. 块存储应用案例 456](#_Toc30989)

[165.1. 准备实验环境 456](#_Toc31689)

[165.1.1. 创建磁盘镜像 456](#_Toc21980)

[165.1.2. Ceph认证账户 456](#_Toc3840)

[165.1.3. 部署客户端环境 456](#_Toc745)

[165.2. 创建KVM虚拟机 456](#_Toc10464)

[165.2.1. 创建初始化虚拟机 456](#_Toc21201)

[165.2.2. 配置libvirt secret 456](#_Toc24239)

[165.2.3. 虚拟机的XML配置文件 456](#_Toc18039)

[165.2.4. 修改XML配置文件 456](#_Toc13223)

[166. 分布式文件系统 456](#_Toc27241)

[166.1.1. 基础知识 456](#_Toc18272)

[166.1.2. 什么是CephFS 456](#_Toc30773)

[166.1.3. 环境准备 456](#_Toc27699)

[166.2. 元数据服务器 456](#_Toc9139)

[166.2.1. 什么是元数据 457](#_Toc3575)

[166.2.2. 部署元数据服务器 457](#_Toc31565)

[166.3. 文件系统服务器 457](#_Toc21399)

[166.3.1. 创建存储池 457](#_Toc15301)

[166.3.2. 创建Ceph文件系统 457](#_Toc15127)

[166.3.3. 客户端挂载 457](#_Toc8027)

[167. Ceph对象存储 457](#_Toc27484)

[167.1. 概述 457](#_Toc4393)

[167.1.1. 什么是对象存储 457](#_Toc12226)

[167.1.2. 环境准备 457](#_Toc25277)

[167.2. 对象存储 457](#_Toc21735)

[167.2.1. 部署RGW软件包 457](#_Toc30408)

[167.2.2. 新建网关示例 457](#_Toc6901)

[167.2.3. 修改服务端口 457](#_Toc30768)

[167.2.4. 客户端测试 457](#_Toc3503)

[167.2.5. 使用第三方软件访问 458](#_Toc26191)

[168. 总结和答疑 458](#_Toc28421)

[168.1. ceph-deploy无法完成部署 458](#_Toc17238)

[168.1.1. 问题现象 458](#_Toc18549)

[168.1.2. 故障分析及排除 458](#_Toc2706)

[168.1.3. 458](#_Toc2238)

[169. 块存储应用案例 459](#_Toc21984)

[169.1. 准备实验环境 459](#_Toc6136)

[169.1.1. 创建磁盘镜像 459](#_Toc5089)

[169.1.2. Ceph认证账户 459](#_Toc31910)

[169.1.3. 部署客户端环境 459](#_Toc20098)

[169.2. 创建KVM虚拟机 459](#_Toc8434)

[169.2.1. 创建初始化虚拟机 459](#_Toc4156)

[169.2.2. 配置libvirt secret 459](#_Toc23589)

[169.2.3. 虚拟机的XML配置文件 459](#_Toc13339)

[169.2.4. 修改XML配置文件 459](#_Toc25365)

[170. 分布式文件系统 459](#_Toc13254)

[170.1. 基础知识 459](#_Toc25157)

[170.1.1. 什么是CephFS 459](#_Toc2244)

[170.1.2. 环境准备 459](#_Toc1088)

[170.2. 元数据服务器 459](#_Toc30221)

[170.2.1. 什么是元数据 460](#_Toc8279)

[170.2.2. 部署元数据服务器 460](#_Toc23837)

[170.3. 文件系统服务器 460](#_Toc977)

[170.3.1. 创建存储池 460](#_Toc2258)

[170.3.2. 创建Ceph文件系统 460](#_Toc7655)

[170.3.3. 客户端挂载 460](#_Toc1036)

[171. Ceph对象存储 460](#_Toc27740)

[171.1. 概述 460](#_Toc610)

[171.1.1. 什么是对象存储 460](#_Toc13792)

[171.1.2. 环境准备 460](#_Toc21354)

[171.2. 对象存储 460](#_Toc18607)

[171.2.1. 部署RGW软件包 460](#_Toc23914)

[171.2.2. 新建网关实例 460](#_Toc26376)

[171.2.3. 修改服务端口 460](#_Toc17006)

[171.2.4. 客户端测试 460](#_Toc6253)

[171.2.5. 使用第三方软件访问 461](#_Toc11021)

[172. 总结和答疑 461](#_Toc3530)

[172.1. ceph-deploy无法完成部署 461](#_Toc19561)

[172.1.1. 问题现象 461](#_Toc30712)

[172.1.2. 故障分析及排除 461](#_Toc11060)

# 云计算介绍

## 典型服务模式

C/s client/Server架构 请求/提供资源.功能

服务器:能够为其他计算机提供服务的更高级的电脑

机架式 塔式 机柜式 刀片式

互联网数据中心 IPC

**虚拟化软件:**虚拟计算机硬件,用真的虚拟出假的,最终假的能够当真的来用

1. Windows:虚拟化软件VMware
2. Linux:虚拟化软件KVM

关闭Windows2008进行克隆

Windows2008修改桌面背景:开始-搜索-desktop

## 修改一台计算机名称并加入指定工作组

### 步骤一：修改计算机名

启动计算机后桌面右击计算机图标—属性—高级系统设置—计算机名及工作组名—更改—计算机名（输入计算机名称）—重启（计算机名在网络上必须是唯一的，即使位于不同的工作组，也不能同名。更改计算机名称后需要重新启动计算机生效。）

### 步骤二：计算机加入指定工作组

启动计算机后桌面右击计算机图标—属性—高级系统设置—计算机名及工作组名—更改—选择工作组输入工作组名）—重启（计算机所属工作组习惯按照计算机操作者所属的部门划分）

# TCP/IP协议及配置

## TCP/IP 协议

### TCP/IP 协议简介

最广泛支持的通信协议集合:包括大量Internet应用中的标准协议,支持跨网络架构,跨操作系统平台的通信

主机与主机之间通信的三个要素:IP地址,子关掩码(subnet mask),IP路由(IP router)

### IP地址的概述

作用:唯一标识网络中一台主机

作用:用来标识一个节点的网络地址

地址组成(点分十进制):

-一共32个二进制位,划分了4部分

-表示为4个十进制数,以.隔开

IP地址的分类

用于一般计算机网络

-A类:1~126(127本机) 网+主+主+主

-B类:128~191 网+网+主+主

-C类:192~223 网+网+网+主

组播及科研专用

-D类:224~239 组播

-E类:240~254 科研

IP地址的组成:网络位+主机位

192.168.1.1 IP地址类别(C) 网+网+网+主

网络:192.168.1 主机位:1

网络位:标识一个网络

主机位:网络中的编号

**-子网掩码:用来标识IP地址网络位与主机位**

192.168.1.1

11000000.10101000.00000001.00000001

子网掩码 11111111.11111111.11111111.00000000

24个网络位.8个主机位

192.168.1.1 255.255.255.0

192.168.1.1/24 24个网络位

**默认子网掩码**

-A类地址,255.0.0.0

-B类地址,255.255.0.0

-C类地址,255.255.255.0

什么是**网关**?

-从一个网络连接到另一个网络的”关口”

-通常是一台路由器,或者防火墙/接入服务器

公有地址(public address,也称公网地址)

-由Internet NIC(互联网信息中心)负责分配

-必须向NIC申请/注册并获得批准方可使用

-适用于Internet,属于广域网地址

私有地址(private address,也称专网地址)

-预留给企业内部网络,使用无需申请

-适用于局域网,通过网关才能访问Internet

预留给企业的私有网络使用

-不在Internet中使用

A类:10.0.0.1~10.255.255.254(10网段)

B类:172.16.0.1~172.31.255.254

C类:192.168.0.1~192.168.255.254

## 基本环境配置

### 更改计算机名和工作组名

### TCP/IP参数设置

### 配置IP地址

IP地址配置方式是:

1.手动配置

2.DHCP自动获取IP地址(前提网络里必须要有DHCP服务器)

DNS服务器:将域名解析为对应IP地址

为了方便记忆网站,网站的推广

网关地址:一个网络到另一个网络的关口(路由器)

### 查看IP有效配置

### 用命令查看IP有效配置

Win+r键:ipconfig

### 使用ping命令测试网络连通性

测试通信命令ping,一次ping操作是双向的,有来有回

关闭Windows2008防火墙

Windows键+R键----->ping 192.168.1.2

# Linux系统简介

## 什么是Linux?

### Linux是一种操作系统

操作系统:一堆软件的集合,计算机硬件正常合理工作

### 超级计算机TOP500

## Unix/Linux发展史

### Unix系统发展

Unix诞生,1970-1-1

Linux诞生,Linus Torwalds

### Linux的诞生

### Linux系统内核

版本号:主版本.次版本.修订号

内核:调配所有的硬件

用户------>内核------->硬件

是一套公开发布的完整Linux系统

-Linux内核+各种应用程序

## Linux版本及应用

### Linux发行版本

发行版的名称/版本由发行方决定

-Red Hat Enterprise Linux 5/6/7

-Suse Linux Enterprise 12

-Debian Linux 7.8

-Ubuntu Linux 14.10/15.04

### Red Hat系统版本

红帽Linux企业版

-简称RHEL(Red Hat Enterprise Linux)

服务费 自主研发的软件

centOS,社区企业操作系统

-Community Enterprise Operating System

如何选择 Linux 操作系统版本?

一般来讲,桌面用户首选 Ubuntu;服务器首选 RHEL 或 CentOS,两者中首选 CentOS。

根据具体要求:

1安全性要求较高,则选择 Debian 或者 FreeBSD。

2需要使用数据库高级服务和电子邮件网络应用的用户可以选择 SUSE。

3想要新技术新功能功能可以选择 Feddora,Feddora 是 RHEL 和 CentOS 的一个测试版和预发布版本。

4根据现有状况,绝大多数互联网公司选择 CentOS。现在比较常用的是 6 系列,现在市场占有大概一半左右。另外的原因是 CentOS 更侧重服务器领域,并且无版权约束。

### Linux在企业中的应用

基于Linux的企业服务器

嵌入式系统

高性能大型运算

# 安装RHEL7系统

虚拟机安装RHEL7.4操作系统

1. 虚拟计算的硬件

Ctrl+Alt:鼠标回到真机

软件包选择:带GUI的服务器

## 预备知识

### 如何使用硬盘?

-物理硬盘==>分区规划==>格式化==>读/写文档

-毛坯楼层==>打隔断==>装修==>入驻

格式化:数据在空间,存储的规则(文件系统)

文件系统:

ext4(RHEL6),第四代扩展文件系统

xfs(RHEL7),高级日志文件系统

SWAP,交换空间(虚拟内存):利用磁盘空间,缓解物理真实内存不足

NTFS(Windows)

### 磁盘与分区表示

一切皆文件

### Linux文件系统

Linux目录结构:树型结构

最顶层为根目录(/)

根目录(/):Linux所有数据都在此目录下(Linux系统的起点)

路径:/root

/dev(device):所有设备相关的文件(例如:键盘、鼠标、硬盘、光驱....)

Hd,表示IDE设备

/dev/hda /dev/dhb /dev/dhc

Sd,表示SCSI设备

/dev/sda1 /dev/sdb2 /dev/sdc3

/dev/sda5:表示SCSI设备,第一块磁盘,第5个分区

vd,表示虚拟化设备

virtual:虚拟化

## 安装RHEL7系统

### 准备系统光盘

### 安装过程

### 初始化过程

### 使用虚拟机软件

# RHEL7基本操作

## 使用图形桌面

### 登录到图形桌面

### 常见的桌面管理操作

## 命令行基本操作

### 获取命令行界面

虚拟控制台切换(ctrl+alt+fn组合键)

-tty1:图形桌面

-tty2~tty6:字符控制台

桌面空白处,右键”打开终端”

应用程序-->工具-->终端

命令行提示标识的含义

-[当前用户@主机名 工作目录]$

-若当前用户是root,则最后一个字符为#,否则为$

### 查看及切换目录

Pwd-print working directory

-用途:查看当前工作目录

Cd-change directory

-用途:切换工作目录

-格式:cd [目标文件夹位置]

Ls-list

-格式:ls[选项]...[目录或文件名]...(支持多个参数)

选项: -l 以长格式显示,显示目录内容的详细属性

### 查看系统版本

存放当前系统版本信息配置文件/etc/redhat-release

### 查看CPU和内存

列出内核版本

[root@svr7 桌面]# uname -r

3.10.0-327.el7.x86\_64 #主版本.次版本.修订号

列出CPU处理器信息

[root@svr7 桌面]# lscpu

......

CPU(s):4 #4个核心

On-line CPU(s) list:0-3

......

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

......

检查内存大小、空闲情况

[root@svr7 桌面]# cat /proc/meminfo

Memtotal:16166612KB

......

第一步:查看系统版本

第二步:查看系统的主机名

第三步:查看系统的IP地址 (远程管理)

### 查看主机名和IP信息

列出当前系统的主机名称

[root@localhost ~]# hostname

[root@localhost ~]# hostname #设置主机名

[root@localhost ~]# hostname

新开一个终端进行验证提示符的变化

列出网卡连接信息

[root@localhost 桌面]# ifconfig

127.0.0.1 永远代表本机

[root@localhost ~]# ping 127.0.0.1

结束正在运行的命令:Ctrl+c

[root@localhost ~]# ifconfig eth0 192.168.1.1 #为网卡eth0 配置IP地址

[root@localhost ~]# ifconfig

[root@localhost ~]# ping 192.168.1.1

### 关机及重启操作

关机:poweroff

重启:reboot

### 创建文档

Mkdir-make directory

-格式:mkdir [/路径/]目录名...

[root@localhost ~]# mkdir test01 #当前目录创建

[root@localhost ~]# ls #查看当前路径下内容

[root@localhost ~]# mkdir /student #在根目录下创建student目录

[root@localhost ~]# ls / #查看根目录下内容

[root@localhost ~]# mkdir /opt/test02 #在/opt目录创建test02目录

[root@localhost ~]# ls /opt #查看/opt目录内容

Touch命令

-用途:新建空文件

-格式:touch 文件名...

绝对路径:以根(/)开始的路径

相对路径:不以根(/)开始的路径,以当前路径为参照

..表示父目录

### 文本内容操作

Cat

用途:查看文本文件的内容(支持多个参数)

选项: -n ,显示行号

颜色方面:

黑色:文本文件

蓝色:目录

完整的命令一般格式

-命令字 [选项]...[参数1][参数2]...

Less分屏阅读工具

格式:less [选项] 文件名...

-优势:支持前后翻页

基本交互操作

-按/键查找(n丶N切换)

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

输入 /a 全文查找a

输入 q 退出

Head、tail命令

-格式:

Head -n 数字 文件名

Tail -n 数字 文件名

[root@localhost ~]# head -3 /etc/passwd #显示文件内容前3行

[root@localhost ~]# tail -3 /etc/passwd #显示文件内容后3行

[root@localhost ~]# head /etc/passwd #默认显示文件内容前10行

[root@localhost ~]# tail /etc/passwd #默认显示文件内容后10行

Grep工具

-用途:输出包含指定字符串的行

-格式:grep [选项]... “查找条件” 目标文件

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

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

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

正则表达式

# 总结和答疑

## RHEL7安装异常

### 问题现象

### 故障分析及排除

## 图形桌面使用

### 问题现象

### 故障分析及排除

## 命令行操作故障

### 问题现象

### 故障分析及排除

# 命令行基础

## 如何编写命令行

### 什么是命令?

Linux命令

-用来实现某一类功能的指令或程序

Linux执行命令,都要找到对应的程序

-命令的执行依赖于解释器(默认解释器:/bin/bash)

用户------>解释器----->内核------>硬件

绿色:可以运行的程序

Linux命令的分类

-内部命令:属于解释器的一部分

-外部命令:解释器之外的其他程序

### 命令行的一般格式

基本用法

-命令字 [选项]... [参数1][参数2]...

### 快速编辑技巧

Tab键自动补全

-可补齐命令字、选项、参数、文件路径、软件名、服务名

[root@localhost ~]# ls /etc/sysconfig/network-scripts/ifcfg-eth0

# ls /et(tab)c/sysco(tab)nfig/netw(tab)ork-(tab)scripts/ifcfg-eth0

# if 连续按两次tab

# ifco(tab)

快捷键

-Ctrl+l:清空整个屏幕

-Ctrl+c:废弃当前编辑的命令行(结束正在运行的命令)

-ESC+.或者alt+.:粘贴上一个命令的参数

-Ctrl+u:清空至行首

-Ctrl+w:往回删除一个单词(以空格界定)

## Mount挂载操作

### 什么是挂载?

### 挂载光盘或分区

从光盘文件中获得众多的软件包

Windows:

光盘文件-------->光驱设备--------->CD驱动器(图标)访问点

Linux:

光盘文件------>光驱设备----------->目录(访问点)

/dev/hdc

/dev/sr0

在命令行访问设备,必须通过访问点进行访问

1. 通过图形将光盘镜像文件,放入虚拟光驱设备
2. 查看光驱设备

青色:快捷方式

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

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

[root@localhost ~]# ls /dev/csr0

1. 进行挂载:让目录成为设备的访问点

[root@localhost ~]# mkdir /dvd #创建目录dvd

[root@localhost ~]# ls /dvd

使用mount命令

-格式:mount 设备路径 挂载点目录

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

Mount: /dev/sr0 写保护,将以只读方式挂载

让/dvd成为/dev/cdrom的访问点

[root@localhost ~]# ls /dvd

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

### 卸载已挂载的设备或分区

1. 卸载

[root@localhost ~]# umount /dvd/

[root@localhost ~]# ls /dvd/

注意事项:

1.卸载时,当前不要在访问点内

[root@localhost dvd]# umount /dvd/

Umount: /dvd:目标忙.

2.挂载设备时,访问点最好是自己创建的目录

# 目录和文件管理

## 查看及切换目录

### 使用pwd、cd

~:代表用户的家目录

~user代表用户user的家目录

/root:管理员的家目录

/home:存放所有普通用户的家目录

[root@localhost /]# cd ~root #去往root用户家目录

[root@localhost ~]# pwd

[root@localhost ~]# useradd dc #创建用户dc

[root@localhost ~]# cd ~dc #去往dc用户家目录

.表示当前目录,经常复制与移动

..表示父目录

### Ls列出文档及属性

Ls-list

-格式:ls[选项]...[目录或文件名]

常用命令选项

-l:以长格式显示,显示详细属性

-h:提供易读的容量单位(K、M等)

-A:包括名称以.开头的隐藏文档

-d:显示目录本身(而不是内容)的属性

[root@localhost ~]# ls --help

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

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

[root@localhost ~]# man passwd #查看权威帮助信息

[root@localhost ~]# man ls #查看权威帮助信息

/example #全文搜索example

### 使用通配符

针对不确定的文档名称,以特殊字符表示

-\*:任意多个任意字符

-?:单个字符

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

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

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

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

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

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

-[a-z]:多个字符或连续范围中的**一个**,若无则忽略

-{a,min,xy}:多组不同的字符串,全匹配

[root@localhost /]#ls /dev/tty[3-6]

[root@localhost /]#ls /dev/tty[1-3]

[root@localhost ~]# ls /dev/tty{1,3,5,7,11,S0}

[root@localhost ~]# ls /dev/tty{2[0-9],30}

### 别名的定义

别名的定义:简化复杂的命令

查看已设置的别名

-alias[别名名称]

定义新的别名

-alias 别名名称=’实际执行的命令行’

取消已设置的别名

-unalias[别名名称]

为真机设置永久的别名,修改配置文件

设置永久的别名配置文件:/root/.bashrc 系统级配置文件

type也可以查看别名

## 新建文档

### Mkdir创建目录

Mkdir-make directory

-格式:mkdir [-p] [/路径/]目录名...(支持多个参数)

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

//-p表示连父目录一起创建

[root@localhost ~]# ls -R /vod/movie #-R表示递归显示

递归:目录本身,以及目录下所有都算上

### 使用vim创建/修改文件

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

-若目标文件已存在,则打开此文件并编辑

三个模式:命令模式 插入模式(编辑模式) 末行模式

[root@localhost ~]# vim

I键

命------------------->输入模式

令 esc键

模 esc键

式-------------------->末行模式

:键

末行模式:wq 保存并退出

末行模式:q! 强制不保存并退出

[root@localhost ~]# vim /mnt/haixing.txt

Vim技巧:

-命令模式下复制、粘贴:

复制1行(yy)、复制3行(3yy)

粘贴到当前行之后(小写p)

-命令模式下删除

删除单个字符(x)

删除到行首(d^)、删除到行尾(d$)

删除1行(dd)、删除3行(3dd)

-命令模式下查找关键词:

搜索(/word)切换结果(n、N)

-补充:在命令模式下大写的C,可以删除光标之后,并且进入输入模式

## 复制/删除/移动

### Cp复制

Cp-copy

-格式:cp [选项]...原文件... 目标路径

可以支持两个以上的参数,永远把最后一个参数作为目标,其他的所有参数都作为源

常用命令选项

-r:递归,复制目录时必须有此选项

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

[root@localhost ~]# cp -r /boot /mnt

处理复制时出现重名,强制覆盖

在本次操作临时取消别名的操作

[root@localhost ~]# \cp -r /boot /mnt

复制与点的结合应用:将原文档复制到当前路径下

.表示当前路径

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

复制时候,可以重新命名目标路径文档的名字

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

[root@localhost /]# cp -r /home /mnt/test

[root@localhost /]# cp -r /home /mnt/test

请描述两次复制的不同?

1. 将/home目录复制到/mnt下重命名为test
2. 将/home目录复制到/mnt/test/目录下

### Rm删除

Rm-remove

-格式:rm [选项]...文件或目录...

常用命令选项

-r、-f:递归删除(含目录)、强制删除

[root@localhost ~]# rm -rf /opt/abc/ /mnt/test05

### Mv移动/重命名

Mv-move

-格式:mv 原文件... 目标路径

[root@localhost ~]# mv /mnt/a.txt /mnt/nsd01/

重命名:路径不变的移动

[root@localhost ~]# mv /mnt/nsd01/ /mnt/student

查看时间date

[root@localhost /]# date

2018年 12月 05日 星期三 16:37:45 CST

[root@localhost /]# date -s “年-月-日 时:分:秒”

[root@localhost /]# date -s “2018-12-22 15:33:53”

交互式计算器:bc

运算法则: 加+ 减- 乘\* 除/ 取余数 %

# 教学环境介绍

## Linux技能等级

### Linux系统管理员

### Linux系统工程师

要求具备从事Linux行业中的中级技能

-侧重于多个服务器的应用部署及管理

-既能对Linux主机进行基础管理工作,还可以配置Web、邮件、文件等服务器,并实现安全运行

-比如SELinux、防火墙、各种网站部署、网络磁盘、资源共享、多网卡聚合、用户环境定制、基础Shell脚本、基础数据库运维等

初学者在 Linux 系统的开机启动项如何选择

建议选择五个开机启动项:

1.crond: 该服务用于周期地执行系统及用户配置的计划任务。有要周期性执行的任务计划需要开启,此服务是生产场景必须要用的一个软件。

2.iptables: iptables 包过滤防火墙,有外网 IP 时,考虑开启。

3.network: 启动系统时,若想激活/关闭启动时的各个网络接口,则应(必须)考虑开启。

4.sshd: 远程连接 Linux 服务器时需要用到这个服务程序,所以必须要开启,否则将无法远程连接到 Linux 服务器。

5.rsyslog: 是操作系统提供的一种机制,系统的守护程序通常会使用 rsyslog 将各种信息收集写入到系统日志文件中,CentOS6 以前此服务的名字为 syslog。

6.sysstat: 是一个软件包,包含监测系统性能及效率的一组工具,这些工具对于 Linux 系统性能数据很有帮助,比如 CPU 使用率、硬盘和网络吞吐数据等,这些数据的分析,有利于判断系统运行是否正常,所以它是提高系统运行效率、安全运行服务的助手。

请描述 Linux 系统优化的 12 个步骤

(1)登录系统:不使用 root 登录,通过 sudo 授权管理,使用普通用户登录。

(2)禁止 SSH 远程:更改默认的远程连接 SSH 服务及禁止 root 远程连接。

(3)时间同步:定时自动更新服务器时间。

(4)配置 yum 更新源,从国内更新下载安装 rpm 包。

(5)关闭 selinux 及 iptables(iptables 工作场景如有 wan ip,一般要打开,高并发除外)

(6)调整文件描述符数量,进程及文件的打开都会消耗文件描述符。

(7)定时自动清理/var/spool/clientmquene/目录垃圾文件,防止节点被占满(c6.4 默认没有 sendmail,因此可以不配。)

(8)精简开机启动服务(crond、sshd、network、rsyslog)

(9)Linux 内核参数优化/etc/sysctl.conf,执行 sysct -p 生效。更改字符集,支持中文,但是还是建议使用英文,防止乱码问题出现。

1. 锁定关键系统文件(chattr +i /etc/passwd /etc/shadow /etc/group /etc/gshadow /etc/inittab 处理以上内容后,把 chatter 改名,就更安全了。
2. 清空/etc/issue,去除系统及内核版本登陆前的屏幕显示。

描述 Linux 系统从开机到登陆界面的启动过程

(1)开机 BIOS 自检,加载硬盘。

(2)读取 MBR,MBR 引导。

(3)grub 引导菜单(Boot Loader)。

(4)加载内核 kernel。

(5)启动 init 进程,依据 inittab 文件设定运行级别

(6)init 进程,执行 rc.sysinit 文件。

(7)启动内核模块,执行不同级别的脚本程序。

(8)执行/etc/rc.d/rc.local

(9)启动 mingetty,进入系统登陆界面。

## 使用教学虚拟机

### 预装虚拟机说明

每个学院机上有三台预先配置好的虚拟机

-server-作为练习用服务器

-desktop -作为练习用客户机

-classroom -提供网关/DNS/软件素材/DHCP服务等资源

优先开机classroom,再去开启Server、desktop

### 访问练习用虚拟机

进入虚拟机server与虚拟机desktop

用户名:root 密码为:redhat

虚拟机server

主机名:server0.example.com

系统版本:RHEL7.0

Eth0网卡IP地址:172.25.0.11

虚拟机desktop

主机名:主机名:server0.example.com

系统版本:RHEL7.0

Eth0网卡IP地址:172.25.0.10

系统版本:RHEL7.0

Eth0网卡IP地址:172.25.0.11

### 使用rht-vmctl辅助工具

真机还原命令:

[root@localhost ~]# rht-vmctl reset classroom

[root@localhost ~]# rht-vmctl reset server

[root@localhost ~]# rht-vmctl reset desktop

### 真机远程管理

[root@localhost ~]# ping 172.25.0.11

[root@localhost ~]# ping 172.25.0.10

远程管理的命令:ssh 登录对方的用户名@对方的IP地址

[root@localhost ~]# ssh root@172.25.0.11

Last login: Thu Dec 6 15:22:38 2018

[root@server0 ~]# exit

登出

Connection to 172.25.0.11 closed.

[root@localhost ~]#

Ctrl+shift+t:新开一个终端

Ssh -X:在远程管理时候,本地运行对方的图形程序

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

[root@server0 ~]# firewall-config #防火墙图形工具

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

alias gos='ssh -X root@172.25.0.11'

alias god='ssh -X root@172.25.0.10'

新开一个全新的终端

# 总结和答疑

## Mkdir建目录失败

### 问题现象

### 故障分析及排除

## Vim不能保存文件

### 问题现象

### 故障分析及排除

# 软件包管理

1. 由classroom提供光盘RHEL7.0所有内容,具备软件包

http:超文本传输协议

虚拟机classroom搭建web服务器,共享光盘RHEL7.0所有内容

虚拟机Server-------->classroom提供光盘RHEL7.0

真机打开浏览器:http://classroom.example.com/

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

## 零散软件管理

### 使用rpm命令管理软件

默认不允许用户做任何选择

RPM Package Manager,RPM包管理器

-rpm -q 软件名... #查询当前系统软件包是否安装

-rpm -i 软件名-版本信息.rpm... #安装软件包

-rpm -ivh 软件名-版本信息.rpm... #带进度条的安装软件包

-rpm -e 软件名... #卸载

-rpm -ql 软件名... #列出安装清单

了解:导入红帽官方信息

[root@server0 ~]# wget

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

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

[root@server0 opt]# rpm -i vsftpd-3.0.2-9.el7.x86\_64.rpm

[root@server0 opt]# rpm -ivh /opt/vsftpd-3.0.2-9.el7.x86\_64.rpm

[root@server0 opt]# rpm -q vsftpd-3.0.2-9.el7.x86\_64

[root@server0 opt]# rpm -ql vsftpd-3.0.2-9.el7.x86\_64

[root@server0 opt]# rpm -e vsftpd-3.0.2-9.el7.x86\_64

vsftpd-3.0.2-9.el7.x86\_64.rpm(包名)

Vsftpd(软件名)

常见的错误信息:依赖关系错误

错误:依赖检测失败:

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

## Yum配置及使用

### 指定yum软件源

Yum软件包仓库,自动解决依赖关系

服务:为客户端自动解决依赖关系,安装软件包

服务端:虚拟机classroom.example.com

1.众多的软件包 2.仓库清单文件(repodata) 3.构建web服务或FTP服务传递数据

客户端:虚拟机server0.example.com

客户端配置文件:/etc/yum.repos.d/\*.repo

错误的配置文件会影响正确的配置文件

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

[rhel7] #仓库标识

name=rhel7.0 #仓库描述信息

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

#指定服务端位置

enabled=1 #是否启用该文件

gpgcheck=0 #是否检测红帽签名

### 使用yum命令管理软件

[root@server0 ~]# yum repolist #列出仓库信息

[root@server0 ~]# yum install httpd #安装软件包

[root@server0 ~]# yum remove #卸载软件

yum clean all #清空缓存

快速建立repo配置文件

-yum-config-manager --add-repo 软件仓库地址

### 下载软件包

Wget 下载工具

# cd /opt

#wget http://classroom.example.com/content/rhel7.0/x86\_64/dvd/Packages/vsftpd-3.0.2-9.el7.x86\_64.rpm

#ls

### 升级Linux内核

虚拟机Server:升级内核

1. 下载内核软件包

#cd /opt

#wget [root@server0 opt]# wget

http://classroom.example.com/content/rhel7.0/x86\_64/errata/Packages/kernel-3.10.0-123.1.2.el7.x86\_64.rpm

#ls

1. 升级内核

[root@server0 /]# uname -r

3.10.0-123.el7.x86\_64

[root@server0 /]# rpm -ivh /opt/kernel-3.10.0-123.1.2.el7.x86\_64.rpm

[root@server0 /]# reboot

[root@localhost~]# gos

[root@server0 ~]# uname -r

3.10.0-123.1.2.el7.x86\_64

# 配置网络

## 临时配置及测试

### 查看/设置IP地址

### 查看/设置主机名

### DNS测试

## 永久配置

### 配置静态主机名

配置永久的主机名

# vim /etc/hostname

# cat /etc/hostname

A.tedu.cn

# exit

登出

Connection to 172.25.0.11 closed.

[root@localhost ~]# gos

[root@A ~]# hostname

### 为本机指定DNS服务器

配置DNS服务器地址

# vim /etc/resolv.conf

nameserver 172.25.254.254

## Nmcli连接管理

### 查看网络连接

### 修改网络连接配置

网卡配置文件/etc/sysconfig/network-scripts/ifcfg-eth0

方式1:通过命令 nmcli connection show #可以修改网卡配置文件

查看命令识别网卡的名称

[root@A ~]# nmcli connection show #可以修改网卡配置文件

进行配置永久的IP地址、子网掩码、网关地址

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

ipv4.method manual

ipv4.addresses '172.25.0.110/24 172.25.0.254'

connection.autoconnect yes

# nmcli connection 修改 ‘识别的网卡名’

ipv4.方法 手工配置

ipv4.addresses ‘IP地址/子网掩码 网关地址’

connection.autoconnect yes 每次开机自动激活配置

方式2:nmtui 修改IP地址为172.25.0.120/24

1.edit a connection (回车) #编辑连接

1. system eth0(回车) #选择网卡
2. Require IPv4 addressing for this connection #利用Ipv4进行激活
3. Automatically connect #每次开机自动激活配置

查看网关命令:

# route -n #以数字的方式显示

Gateway

172.25.0.254

### 激活/禁用网络连接

激活配置

# nmcli connection up 'System eth0'

# 文本/文件查找

## 查找文本内容

### Grep过滤操作

### 重定向输出

## 查找文件

### Find按条件查找文件

### Find结果处理

# 总结和答疑

## 配置网络参数

### 问题现象

### 故障分析及排除

## Classroom异常

### 问题现象

### 故障分析及排除

# 管理用户和组

用户账号作用:

1. 访问控制的设置,不同的用户具备不同权限
2. 提高安全可靠性

组账号作用:方便管理用户账号

用户账号标识:UID

管理员UID为0

所有普通用户UID默认从1000起始

用户账号标识:GID

组账号分类: 基本组 附加组(从属组)

所有用户至少属于一个组

基本组:由Linux系统创建,由Linux将用户加入,与用户同名

附加组(从属组):由root用户创建

## 管理用户账号

### 添加用户

/etc/passwd:用户基本信息存放在/etc/passwd文件

[root@server0 /]# head -1 /etc/passwd

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

用户名:密码占位符:UID:基本组GID:用户描述信息:家目录:解释器

使用useradd命令

-useradd [选项]...用户名

常用命令选项

-u 用户id、-d 家目录路径、-s 登录解释器、-G 附加组

[root@server0 ~]# useradd nsd01

[root@server0 ~]# id nsd01 #查看基本信息

[root@server0 ~]# grep nsd01 /etc/passwd

[root@server0 ~]# useradd nsd02

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

[root@server0 ~]# useradd -u 1600 nsd05 #指定UID

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

[root@server0 ~]# id nsd05

# useradd -d /opt/nsd07 nsd07 #指定用户家目录创建

# grep nsd07 /etc/passwd

# ls /opt/

-s 登录解释器

/sbin/nologin:禁止用户登录系统

[root@server0 ~]# useradd -s /sbin/nologin nsd08

[root@server0 ~]# grep nsd08 /etc/passwd

[root@server0 ~]# groupadd tarena

[root@server0 ~]# useradd -G tarena harry #指定附加组

[root@server0 ~]# id harry

重定向输出:将前面命令的输出结果,写入文本文件

>:覆盖重定向

>>:追加重定向

[root@server0 /]# ls --help

[root@server0 /]# ls --help > /opt/ls.txt

[root@server0 /]# vim /opt/ls.txt

[root@server0 /]# less /opt/ls.txt

[root@server0 /]# hostname > /opt/ls.txt

[root@server0 /]# cat /opt/ls.txt

[root@server0 /]# hostname >> /opt/ls.txt

[root@server0 /]# cat /opt/ls.txt

[root@server0 /]# ifconfig eth0 >> /opt/ls.txt

[root@server0 ~]# echo hello world

[root@server0 ~]# echo hello world > /opt/ls.txt

[root@server0 ~]# cat /opt/ls.txt

[root@server0 ~]# echo A.tedu.cn > /etc/hostname

[root@server0 ~]# cat /etc/hostname

[root@server0 ~]# hostname

管道 |:将前面命令的输出结果,交由后面命令处理

[root@server0 ~]# head -12 /etc/passwd

[root@server0 ~]# head -12 /etc/passwd | tail -5

[root@server0 ~]# cat -n /etc/passwd | head -12 |tail -5

[root@server0 ~]# echo 1+2 | bc

### 设置登录密码

使用passwd命令

-passwd [用户名]

[root@server0 ~]# passwd nsd01

更改用户 nsd01 的密码 。

新的 密码： #输入密码

无效的密码： 密码少于 8 个字符

重新输入新的 密码： #输入密码

passwd：所有的身份验证令牌已经成功更新。

临时切换用户身份:su - 用户名

[root@server0 ~]# su - nsd01

[nsd01@A ~]$ passwd

Changing password for user nsd01.

Changing password for nsd01.

(current) UNIX password: 123456

New password: zxc123456

BAD PASSWORD: The password is too similar to the old one

New password: zxc987654

BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic

New password:987654

BAD PASSWORD: The password is shorter than 8 characters

passwd: Have exhausted maximum number of retries for service

New password: zqy330722

Retype new password: zqy330722

passwd: all authentication tokens updated successfully.

[nsd01@A ~]$ exit

logout

非交互式设置用户密码

-echo ‘密码’ | passwd -- stdin 用户名

[root@server0 ~]# echo 123 | passwd --stdin nsd01

更改用户 nsd01 的密码 。

passwd：所有的身份验证令牌已经成功更新。

用户密码信息存放在 /etc/shadow 文件

用户名:密码加密字符串:上一次修改密码的时间

自1970-1-1到上一次修改密码时间经过的天数

### 修改用户属性

使用usermod 命令

-usermod [选项]... 用户名

常用命令选项

-u 用户id、-d家目录路径、-s登录解释器

-G 附加组

[root@server0 ~]# useradd nsd10

[root@server0 ~]# grep nsd10 /etc/passwd

nsd10:x:1603:1604::/home/nsd10:/bin/bash

[root@server0 ~]# usermod -u 1700 -s /sbin/nologin -d /opt/abc -G tarena nsd10

[root@server0 ~]# grep nsd10 /etc/passwd

nsd10:x:1700:1604::/opt/abc:/sbin/nologin

[root@server0 ~]# id nsd10

uid=1700(nsd10) gid=1604(nsd10) 组=1604(nsd10),1602(tarena)

密码加密字符串为!!时是锁定状态

### 检查用户ID信息

### 删除用户

使用userdel命令

-userdel [-r] 用户名 #-r:连同用户家目录一并删除

[root@server0 ~]# userdel -r nsd11

[root@server0 ~]# id nsd11

id: nsd11: no such user

## 管理组账号

组基本信息存放在/etc/group文件

[root@server0 ~]# grep stugrp /etc/group

stugrp:x:2002:

组名:密码占位符:组的GID:组的成员列表

### 添加组

使用groupadd命令

-groupadd [-g 组ID] 组名

[root@server0 ~]# groupadd stugrp

[root@server0 ~]# grep stugrp /etc/group #查看组的信息

stugrp:x:1605:

[root@server0 ~]# useradd kenji

[root@server0 ~]# useradd chihiro

[root@server0 ~]# useradd jack

[root@server0 ~]# useradd natasha

### 管理组成员

使用gpasswd命令

-gpasswd -a 用户名 组名

-gpasswd -d 用户名 组名

[root@server0 ~]# gpasswd -a kenji stugrp

[root@server0 ~]# grep stugrp /etc/group

[root@server0 ~]# gpasswd -a natasha stugrp

[root@server0 ~]# grep natasha /etc/group

[root@server0 ~]# gpasswd -d kenji stugrp

[root@server0 ~]# grep stugrp /etc/group

### 修改组属性(了解)

使用groupmod命令

-groupmod [-g 组ID] [-n 新组名] 组名

### 删除组

使用groupdel命令

-groupdel 组名

仅删除组不删除用户

创建一个用户Linux有那些文件会改变?

[root@server0 ~]# grep nsd20 /etc/passwd

[root@server0 ~]# ls /home/

[root@server0 ~]# grep nsd20 /etc/shadow

[root@server0 ~]# grep nsd20 /etc/group

# Tar备份与恢复

## Tarball备份包

### 归档和压缩

归档及压缩:1.零散的文件归成一个 2.占用硬盘空间变小

归档的含义

-将许多零散的文件整理为一个文件

-文件总的大小基本不变

压缩的含义

-按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

常见的压缩格式及命令工具:

gz--->gzip、gunzip

bz2--->bzip2、bunzip2

xz--->xz、unxz

制作归档及压缩(打包---->tar包)

格式:tar 选项 /路径/tar包名字 /路径/源文件 /路径/源文件

解包

格式:tar 选项 /路径/tar包名字 /路径/释放的位置

打包:

tar -zcf /路径/tar包名字 /路径/源文件

tar -jcf /路径/tar包名字 /路径/源文件

tar -Jcf /路径/tar包名字 /路径/源文件

解包:

tar -xf /路径/tar包名字 -C /路径/指定释放位置

红色:压缩包

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

[root@A ~]# rm -rf /mnt/\*

[root@A ~]# tar -zcf /opt/nsd02.tar.gz /home/ /boot/

[root@A ~]# ls /opt

[root@A ~]# tar -jcf /opt/test01.tar.gz /home/ /boot/

[root@A ~]# tar -Jcf /opt/test01.tar.gz /home/ /boot/

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

[root@A ~]# ls /mnt

### Tar工具的常用选项

-c:创建归档

-x:释放归档

-f:指定归档文件名称

-z、-j、-J:调用.gz、.bz2、.xz格式的工具进行处理

-t:显示归档中的文件清单

-C(大写):指定释放位置

## 备份与恢复操作

### 制作tar备份包

创建一个备份包

使用tar工具完成以下备份任务:

-创建一个名为/root/backup.tar.bz2的归档文件

-其中包含/usr/local目录中的内容

-tar归档必须使用bzip2进行压缩

### 查看tar备份包内容

查看tar包内的内容

tar -tf

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

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

### 从tar备份包恢复文档

# Ntp时间同步

## 校对系统时间

### 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

### 配置chronyd校时服务

服务端:虚拟机classroom

常见的Web:Apache(httpd) Nginx Tomcat

客户端:虚拟机Server

1. 安装chrony软件包(客户端软件:专用于访问服务的软件)

# yum -y install chrony

# rpm -q chrony

1. 修改程序主配置文件,指定服务端位置

在Linux大多数配置文件中,以#开头的行为注释行

[root@A ~]# vim /etc/chrony.conf

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

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

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

server classroom.example.com iburst

1. 重启程序

# systemctl restart chronyd #重启chronyd

# systemctl enable chronyd #设置为开机自启

Daemon:超级守护进程

### 测试校时服务

# date -s "2008-10-1 12:00"

# date

# systemctl restart chronyd

# date

# Cron计划任务

## 周期性任务

### Cron任务概述

用途:按照设置的时间间隔为用户反复执行某一项固定的系统任务

软件包:cronie、crontabs

系统服务:crond

日志文件:/var/log/crond 排除故障

配置格式可参考 /etc/crontab文件

-分 时 日 月 周 任务命令行(绝对路径)

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

30 23 \* \* \*

\*:匹配范围内任意时间

,:分隔多个不连续的时间点

-:指定连续时间范围

/n:指定时间频率,每n

### 管理计划任务策略

### 如何编写crontab任务记录

如何编写crontab任务记录

分 时 日 月 周 任务命令行(绝对路径)

30 23 \* \* \* poweroff

30 \*/2 \* \* \*

34 8 \* \* 1-5

34 8 1 \* 1 每月1号与每周一都会运行

• 使用 crontab 命令

– 编辑:crontab -e [-u 用户名]

– 查看:crontab -l [-u 用户名]

– 清除:crontab -r [-u 用户名]

每分钟记录当前系统的时间,写入/opt/time.txt

[root@server0 ~]# date

2019年 01月 04日 星期五 17:29:25 CST

[root@server0 ~]# date >> /opt/time.txt

[root@server0 ~]# cat /opt/time.txt

[root@server0 ~]# crontab -e -u root #编写root用户计划任务

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

[root@server0 ~]# crontab -l -u root #查看root用户计划任务

[root@server0 ~]# ls /var/spool/cron/

[root@server0 ~]# cat /var/spool/cron/root #生成的任务文件

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

[root@server0 ~]# cat /opt/time.txt

# 总结和答疑

## NTP校时失败

### 问题现象

### 故障分析及排除

# 权限和归属

## 基本权限

### 基本权限的类别

访问方式(权限)

-读取:允许查看内容-read r

-写入:允许修改内容-write w

-可执行:允许运行和切换-execute x

对于文本文件作用:

r:查看文件内容 cat less head tail

w:修改文件内容 vim 保存

x:双击运行该文件

权限适用对象(归属)

-所有者:拥有此文件/目录的用户-user u

-所属组:拥有此文件/目录的组-group g

-其他用户:除所有者、所属组以外的用户-other o

### 查看权限

使用ls -l命令

-ls -ld 文件或目录...

以-开头:代表文本文件

以d开头:代表目录

以l开头:快捷方式

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

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

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

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

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

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

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

### 设置基本权限

使用chmod命令

-chmod [-R] 归属关系+-=权限类别 文档...

[-R]:递归修改权限

[root@A ~]# mkdir /nsd01

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

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

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

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

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

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

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

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

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

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

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

Linux判断用户具备的权限: 匹配及停止

1.查看用户对于文档处于什么身份 所有者>所属组>其他人

2.查看相应权限位置的权限

Permission denied:权限不足

目录的r权限:能够ls浏览此目录内容

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

目录的x权限:能够cd切换到此目录

### 设置文档归属

设置文档归属

使用chown命令

-chown [-R] 属主 文档...

-chown [-R] :属组 文档...

-chown [-R] 属主:属组 文档...

[-R]:递归修改归属关系

[root@A ~]# useradd zhangsan

[root@A ~]# groupadd tarena

[root@A ~]# mkdir /nsd03

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

[root@A ~]# chown zhangsan:tarena /nsd03

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

[root@A ~]# chown lisi /nsd03 #修改所有者

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

[root@A ~]# chown :root /nsd03 #修改所属组

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

## 附加权限(特殊权限)

### Set GID

附加在属组的x位上

-属组的权限标识会变为s

-适用于目录,Set GID可以使目录下新增的文档自动设置与父目录相同的属组

-让新增的子文档,自动继承父目录的所属组

[root@A ~]# mkdir /nsd09

[root@A ~]# chown :tarena /nsd09 #修改所属组权限

[root@A ~]# ls -ld /nsd09

[root@A ~]# mkdir /nsd09/test01

[root@A ~]# ls -ld /nsd09/test01

### 设置附加**权限**

[root@A ~]# chmod g+s /nsd09 #添加Set GID权限

[root@A ~]# ls -ld /nsd09

## Acl访问控制列表

### Acl策略的作用

文档归属的局限性

-任何人只属于三种角色:属主、属组、其他人

-无法实现更精细的控制

Acl访问策略

-能够对个别用户、个别组设置独立的权限

-大多数挂载的EXT3/4、XFS文件系统默认已支持

### 设置acl访问控制策略

[root@A ~]# mkdir /nsd11

[root@A ~]# chmod o=--- /nsd11

[root@A ~]# su - zhangsan

[zhangsan@A ~]$ cd /nsd11 #切换失败

[zhangsan@A ~]$ exit

[root@A ~]# setfacl -m u:zhangsan:rx /nsd11

[root@A ~]# getfacl /nsd11

[root@A ~]# su - zhangsan

[zhangsan@A ~]$ cd /nsd11 #切换成功

[zhangsan@A nsd11]$ pwd

[zhangsan@A nsd11]$ exit

使用getfacl、setfacl命令

-getfacl文档...

-setfacl -m u:用户名:权限类别 文档...

-setfacl -m g:组名:权限类别 文档...

-setfacl -x u:用户名 文档... #删除指定的ACL

-setfacl -b 文档... #删除所有的ACL

[-R]:递归设置ACL权限

[root@A ~]# mkdir /nsd12

[root@A ~]# setfacl -m u:zhangsan:rwx /nsd12 #设置用户ACL

[root@A ~]# setfacl -m u:lisi:rx /nsd12 #设置用户ACL

[root@A ~]# setfacl -m u:dc:rwx /nsd12 #设置用户ACL

[root@A ~]# setfacl -x u:zhangsan /nsd12 #删除zhangsan的ACL

[root@A ~]# getfacl /nsd12

[root@A ~]# setfacl -b /nsd12 #删除所有的ACL

[root@A ~]# getfacl /nsd12

[root@A ~]# mkdir /public

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

[root@A ~]# getfacl /public

# 使用LDAP认证

## LDAP目录服务

### 什么是LDAP?

轻量级目录访问协议

主要实现网络用户认证,达到用户的集中管理

本地用户:由本地创建,用于本地登录 本地/etc/passwd

网路用户:由LDAP服务器提供用户名、密码信息

服务端:LDAP服务器 虚拟机classroom

客户端:虚拟机server

### 典型的LDAP工作模式

## 如何加入LDAP域

### 加入LDAP需要的条件

### 配置LDAP认证

1. 安装sssd软件包,与LDAP沟通的软件

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

1. 安装authconfig-gtk图形的工具,配置sssd软件包

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

1. 运行authconfig-gtk图形的工具

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

[root@A ~]# authconfig-gtk

选择LDAP

dc=example,dc=com #指定服务端域名

classroom.example.com #指定服务端主机名

勾选TLS加密

使用证书加密:http://classroom.example.com/pub/example-ca.crt

选择LDAP密码

最后选择”应用” dc为域控制器

1. 重启服务,验证

[root@A ~]# systemctl restart sssd

[root@A ~]# systemctl enable sssd

[root@A ~]# grep ldapuser0 /etc/passwd

[root@A ~]# id ldapuser0

uid=1700(nsd10) gid=1604(nsd10) 组=1604(nsd10),1602(tarena)

### 验证LDAP用户登录

# 家目录漫游

## 访问NFS共享

### 什么是NFS共享

Network File System,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

Classroom-->NFS共享----->家目录

### 如何访问NFS共享目录

NFS共享服务端:虚拟机classroom

客户端:虚拟机Server 访问虚拟机classroom的共享

1. 查看NFS资源

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

Export list for classroom.example.com:

/home/guests 172.25.0.0/255.255.0.0

共享路径 共享给哪些客户端

## 挂载LDAP家目录

### 漫游家目录的条件

1. 访问虚拟机classroom

[root@A ~]# mount classroom.example.com:/home/guests /mnt/

[root@A ~]# ls /mnt

[root@A ~]# umount /mnt

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

[root@A ~]# mount classroom.example.com:/home/guests /home/guests

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

#mount 服务端:服务端共享路径 本地挂载点

### 客户机配置及验证

# 总结和答疑

## 对目录的w权限

### 问题现象

### 故障分析及排除

## LDAP家目录漫游

### 问题现象

### 故障分析及排除

# 综合串讲

## 附加权限

### Set UID

附加在属主的x位上

-属主的权限标识会变为s

-适用于可执行文件,Set UID可以让使用者具有文件属主的身份及部分权限

[root@A ~]# cp /usr/bin/touch /usr/bin/xixich

[root@A ~]# chmod u+s /usr/bin/xixich

[root@A ~]# ls -ld /usr/bin/xixich

[root@A ~]# ls -ld /usr/bin/touch

[root@A ~]# su - student

[student@A ~]$ /usr/bin/touch 1.txt

[student@A ~]$ /usr/bin/xixich 2.txt

[student@A ~]$ ls -ld 1.txt

[student@A ~]$ ls -ld 2.txt

[student@A ~]$ exit

### Sticky Bit

附加在其他人的x位上

-其他人的权限标识会变为t

-适用于开放w权限的目录,可以阻止用户滥用w写入权限(禁止操作别人的文档)

[root@A ~]# mkdir /home/public

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

[root@A ~]# ls -ld /home/public

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

[root@A ~]# ls -ld /home/public

## 查找文件

根据字符串模式提取文本行

-grep [选项] ‘匹配模式’ 文本文件...

-命令行 | grep [选项] ‘匹配模式’

常用命令选项

-v,取反匹配

-i,忽略大小写

-p,屏蔽过滤结果,静默过滤,但可以通过$?查看结果

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@A ~]# grep root /etc/passwd

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

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

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

[root@A ~]# grep ^UUID /etc/fstab

[root@A ~]# grep ^127 /etc/hosts

显示/etc/login.defs配置文件有效信息(去除空行 去除注释行)

[root@A ~]# grep -v ^# /etc/login.defs

[root@A ~]# grep -v ^# /etc/login.defs | grep -v ^$

[root@A ~]# grep -v ^# /etc/login.defs | grep -v ^$ > /opt/1.txt

[root@A ~]# cat /opt/1.txt

### Find扩展

查找文件

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

[root@A ~]# find /boot/ -type d #查找/boot目录下是目录的

[root@A ~]# find /boot/ -type f #查找/boot目录下是文件的

[root@A ~]# find /boot/ -type l #查找/boot目录下是快捷方式

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

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

统计命令

[root@A ~]# wc -l /etc/passwd #统计行数

统计/etc目录下以.conf结尾的有多少个?

[root@A ~]# find /etc/ -name "\*.conf"

[root@A ~]# find /etc/ -name "\*.conf" | wc -l

[root@A ~]# find /etc/ -name "\*tab"

[root@A ~]# find /etc/ -name "\*tab" | wc -l

[root@A ~]# find /etc/ -name "\*tty\*"

[root@A ~]# find /etc/ -name "tty\*" | wc -l

[root@A ~]# mkdir /root/nsd1808

[root@A ~]# touch /root/nsd01.txt

[root@A ~]# touch /root/nsd02.txt

[root@A ~]# find /root/ -name "nsd\*"

[root@A ~]# find /root/ -name "nsd\*" -type f #默认两个条件都满足

[root@A ~]# find /root/ -name "nsd\*" -type d

-size +10M #大于10M

-size -10M #小于10M

k(小写) G(大写) M(大写)

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

[root@A ~]# ls -lh /boot/initr\*

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

-user 用户名 #按照文档的所有者

/proc/:所占用的空间不是硬盘,而是内存

[root@A ~]# find / -user student

[root@A ~]# find / -user student -type d

[root@A ~]# find / -user student -type f

根据名称查找,忽略大小写

-iname

[root@A ~]# find /etc -iname "PASSWD"

[root@A ~]# find /etc -name "PASSWD"

根据账号名称或所属组

-user、-group

[root@A ~]# find /home -group lisi

限制目录查找的深度(最大层数)

-maxdepth

[root@A ~]# find /etc -maxdepth 1 -name "\*.conf"

[root@A ~]# find /etc -maxdepth 2 -name "\*.conf"

[root@A ~]# find /etc -maxdepth 3 -name "\*.conf"

使用find命令的-exec操作

-find .. .. -exec 处理命令 {} \;

-优势:以{}代替每一个结果,逐个处理,遇 \;结束

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

[root@A ~]# find /etc -name "\*tab"

[root@A ~]# find /etc -name "\*tab" -exec cp {} /mnt/ \;

[root@A ~]# ls /mnt

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

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

[root@A ~]# ls /opt

使用find命令完成以下任务

-找出所有用户student拥有的文件

-把它们拷贝到/root/findfiles/文件夹中

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

[root@A ~]# find / -user student -type f -exec cp {} /root/findfiles/ \;

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

根据文件修改时间.所有的时间都是过去时间

-mtime +10 #过去的10天之前修改和创建文档

-mtime -10 #过去的10天之内修改和创建文档

var:存放经常变化的数据

[root@A ~]# find /var/log -mtime -10

[root@A ~]# find /var/log -mtime +10

[root@A ~]# find /var/log -mtime -1000

## 提示及指导

### 练习注意事项

### 指导练习及疑难解答

分 时 日 月 周 执行动作(提高执行的可靠性,书写绝对路径)

[root@A ~]# which find #查询命令所对应的程序

首先激活网卡,再去修改DNS服务器配置文件

# 综合练习

## Linux管理员测试

### 环境准备及说明

# 分区规划及使用

## 硬盘分区管理

### 一块硬盘的”艺术”之旅

### 使用fdisk分区工具

识别硬盘

[root@A ~]# lsblk

NAME SIZE TYPE MOUNTPOINT

vda 10G disk

└─vda1 10G part /

vdb 10G disk

分区规划

MBR/msdos分区模式

-分区类型:主分区、扩展分区、逻辑分区

-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2TB的磁盘

-扩展分区不能格式化

/dev/sda5:SCSI接口设备第一块硬盘的第一个逻辑分区

GPT分区模式,最大到18EB

1EB=1000PB

1PB=1000TB

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

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

p 查看分区表

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

w 保存并退出

[root@A ~]# lsblk

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

### 识别新分区表

## 格式化分区

### 常用的格式化工具

格式化,赋予空间文件系统(规则)

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

[root@A ~]# blkid /dev/vdb1 #查看文件系统类型

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

[root@A ~]# blkid /dev/vdb2 #查看文件系统类型

### 创建文件系统

## 访问文件系统

### 访问已格式化的分区

挂载使用分区

[root@A ~]# mkdir /mypart1 /mypart2

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

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

[root@A ~]# df -h #显示已经挂载的设备使用情况

### 实现开机自动挂载

配置/etc/fstab

设备路径 挂载点 类型 参数 备份标记 检测顺序

[root@A ~]# tail -2 /etc/fstab

/dev/vdb1 /mypart1 ext4 defaults,ro 0 0

/dev/vdb2 /mypart2 xfs defaults,ro 0 0

[root@A ~]# umount /mypart1

[root@A ~]# umount /mypart2

[root@A ~]# df -h

[root@A ~]# mount -a

检测/etc/fstab开机自动挂载配置文件,格式是否正确

检测/etc/fstab中,书写完成,但当前没有挂载的设备,进行挂载

[root@A ~]# df -h

继续分区

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

p 查看分区表

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

p 查看分区表

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

p 查看分区表

n 创建逻辑分区---->起始回车---->结束+1G

n 创建逻辑分区---->起始回车---->结束+1G

p 查看分区表

w 保存并退出

[root@A ~]# lsblk

[root@A ~]# partprobe #刷新 新的分区表

[root@A ~]# lsblk

**总结:**

1.识别硬盘 lsblk

2.划分分区 fdisk

3.刷新分区表 partprobe

4.格式化 mkfs.ext4 mkfs.xfs mkfs.vfat -F 32---->blkid查看文件系统

5.利用mount挂载使用 ---->/etc/fstab完成开机自动挂载

为虚拟机添加新的硬盘60G

1. 关闭虚拟机Server
2. 利用图形为虚拟机Server添加60G硬盘

开机之后:

[root@localhost ~]# gos

[root@A ~]# lsblk

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

新加的60G硬盘进行分区

三个主分区10G,划分扩展分区

两个逻辑分区10G

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

p 查看分区表

n 创建新的分区---->回车---->回车---->回车---->在last结束时+10G

连续创建3个10G主分区

......

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

p 查看分区表

n 创建逻辑分区---->起始回车---->结束+10G

n 创建逻辑分区---->起始回车---->结束+10G

p 查看分区表

w 保存并退出

[root@A ~]# lsblk

# LVM逻辑卷

LVM逻辑卷的管理

作用:1.可以整合分散的空间 2.容量大小可以扩大

-零散空间存储---->整合的虚拟磁盘---->虚拟的分区

物理卷PV 卷组VG 逻辑卷LV

将众多的物理卷(PV),组成卷组(VG),再从卷组中划分出逻辑卷(LV)

## 新建逻辑卷

Successfully:成功

1. 直接创建卷组

格式:vgcreate 卷组的名 设备路径1 设备路径2 ......

[root@A ~]# vgcreate systemvg /dev/vdc1 /dev/vdc2

[root@A ~]# vgs #查看卷组信息

[root@A ~]# pvs #查看物理卷信息

1. 通过卷组划分逻辑卷

格式:lvcreate -n 逻辑卷名 -L 逻辑卷大小 卷组名

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

[root@A ~]# lvs #查看逻辑卷信息

[root@A ~]# vgs

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

[root@A ~]# mkfs.ext4 /dev/systemvg/mylv #格式化ext4

[root@A ~]# blkid /dev/systemvg/mylv #查看设备文件系统

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

/dev/systemvg/mylv /lvm ext4 defaults 0 0

[root@A ~]# mkdir /lvm

[root@A ~]# mount -a #检测是否书写正确

[root@A ~]# df -h #查看挂载情况

### LVM工作方式

### LVM管理工具集

### LVM快速部署及使用

## 综合分区规划

### 应用需求

### 调整现有磁盘分区

### LVM卷组的PE大小

卷组划分空间的单位:PE 默认4M

1. 查看卷组PE大小

[root@A ~]# vgdisplay systemvg

PE Size 4.00 MiB

1. 修改方式

卷组已经存在,修改PE的大小

vgchange -s PE大小 卷组名

创建卷组的时候设置PE的大小

vgcreate -s PE大小 卷组名 设备路径

创建逻辑卷使用PE的个数创建

lvcreate -l PE的个数 -n 逻辑卷 卷组名

1. 基于刚建立的2000MiB分区构建新的LVM存储

-新的逻辑卷命名为database,大小为50个物理扩展单元(Physical Extent),属于datastore卷组

-在datastore卷组中的所有逻辑卷,其物理扩展单元(Physical Extent)的大小为16MiB

-使用EXT3文件系统对逻辑卷database格式化,此逻辑卷应该在开机时自动挂载到/mnt/database目录

vgchange -s 16M systemvg

lvcreate -L 800M -n database systemvg

lvcreate -l 50 -n lvtest02 systemvg

lvs

1. 查看卷组PE大小

[root@A ~]# vgdisplay systemvg

[root@A ~]# vgchange -s 1M systemvg

[root@A ~]# vgdisplay systemvg

1. 利用PE的个数创建逻辑卷

[root@A ~]# lvcreate -l 80 -n lvtest systemvg

[root@A ~]# lvs

## 扩展逻辑卷大小

### 检查现有逻辑卷大小

### 扩展卷组

卷组没有足够的剩余空间

[root@A ~]# vgextend systemvg /dev/vdc3

[root@A ~]# vgs

### 扩展逻辑卷

卷组有足够的剩余空间

1. 扩展空间的大小

[root@A ~]# vgs

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

[root@A ~]# lvs

### 更新文件系统大小

1. 扩展文件系统的大小

resize2fs:扩展ext4文件系统

xfs\_growfs:扩展xfs文件系统

[root@A ~]# df -h

[root@A ~]# resize2fs /dev/systemvg/mylv

[root@A ~]# df -h

了解内容:缩减空间

ext4文件系统支持缩减

xfs文件系统不支持缩减

**逻辑卷的删除**

1. 先删除逻辑卷本身(先卸载)
2. 再删除卷组
3. 最后删除物理卷(可选)

[root@A ~]# lvremove /dev/systemvg/mylv #删除

[root@A ~]# umount /lvm

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

[root@A ~]# lvs

[root@A ~]# lremove /dev/systemvg/lvtest

[root@A ~]# lvs

[root@A ~]# vgremove systemvg #删除卷组

[root@A ~]# vgs

[root@A ~]# pvremove /dev/vdc[1-3] #删除物理卷

[root@A ~]# pvs

# 总结和答疑

## 逻辑卷扩展

### 问题现象

利用lvextend命令

-逻辑卷扩展后,df命令信息未变化

### 故障分析及排除

原因分析

-没刷新文件系统

解决办法

-刷新ext4文件系统:resize2fs

-刷新xfs文件系统:xfs\_growfs

### 

# Shell脚本基础

## 认识Shell环境

### Bash shell的使用方式

### 什么是shell脚本

脚本:一个可以运行的文件,可以实现某种功能

初级脚本,就是进行命令的堆积

常见的脚本语言:Bash Shell

Python/Perl/Ruby

JSP/PHP/ASP/CGI

JavaScript

### 规范shell脚本的一般组成

#!环境声明(以下的代码有那个程序进行翻译)

#注释文本

可执行代码

## 第一个shell脚本

### 脚本的创建过程

案例:

请利用shell语言实现,运行一个文件,可以在命令行

输出:hello world

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

echo hello world

[root@A ~]# chmod +x /root/hello.sh

[root@A ~]# ls -l /root/hello.sh

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

### 编写一个批处理shell脚本

案例:

1. 输出当前红帽系统的版本信息
2. 输出当前使用的内核版本
3. 输出当前系统的主机名

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

#! /bin/bash

echo hello world

hostname

cat /etc/redhat-release

uname -r

ifconfig | head -2 | tail -1

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

### 脚本运行及调试

## 简单脚本技巧

### 管道传递

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

-cmd1|cmd2[|cmd3]......

### 免交互及输出处理

### 重定向输出

>:只收集前面命令的正确输出信息,写入文本文件中

2>:只收集前面命令的错误输出信息,写入文本文件中

&>:收集前面命令的正确与错误输出信息,写入文本文件中

[root@A ~]# echo 123 > /opt/1.txt

[root@A ~]# cat /opt/1.txt /2.txt #有正确信息与错误信息的输出

[root@A ~]# cat /opt/1.txt /2.txt > /mnt/a.txt

[root@A ~]# cat /mnt/a.txt

[root@A ~]# cat /opt/1.txt /2.txt 2> /mnt/a.txt

[root@A ~]# cat /mnt/a.txt

[root@A ~]# cat /opt/1.txt /2.txt &> /mnt/a.txt

[root@A ~]# cat /mnt/a.txt

案例:

书写创建用户zhangsan,并设置密码

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

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

#! /bin/bash

useradd zhangsan

echo 123 | passwd --stdin zhangsan

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

黑洞设备:/dev/null 专用于收集无用的信息

# 使用变量

## 变量的定义/引用

### 什么是变量

为了增加脚本的灵活度,适用多变的环境------>变量

变量:类似于容器,以不变的名称,存储可以变化的值

以不变的名称存放的可能会变化的值

-变量名=变量值

-方便以固定名称重复使用某个值

-提高对任务需求、运行环境变化的适应能力

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

-等号两边不要有空格

-变量名只能字母/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

### 定义/赋值变量

变量名=储存的值 #为变量赋值

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

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

#! /bin/bash

read -p ‘请输入您要创建的用户:’ abc

useradd $abc &> /dev/null

echo $abc创建成功q`

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

echo $abc密码设置成功

read:

1. 产生交互
2. 记录用户在键盘上的输入
3. 将记录的信息赋值给变量

-p: 屏幕的提示信息

### 查看/引用变量

基本格式

-引用变量值:$变量名

-查看变量值:echo $变量名、echo ${变量名}

[root@server0 ~]# a=rhel

[root@server0 ~]# echo $a

[root@server0 ~]# echo $a7

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

## 变量的种类

### 运维角度的变量区分

### 环境变量

变量名一般都大写,由系统定义完成,用户直接使用

USER=永远储存当前系统登录的用户名

[root@server0 ~]# echo $USER

[root@server0 ~]# su - dc

[dc@server0 ~]# echo $USER

[dc@server0 ~]$ exit

[root@server0 ~]# su - student

[student@server0 ~]$ echo $USER

[student@server0 ~]$ exit

### 位置变量

由系统定义完成,用户直接使用,非交互式

在执行脚本时,提供的命令行参数

-表示为$n,n为序号

-$1、$2、.. ..${10}、${11}、.. ..

案例:利用位置变量实现

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

#! /bin/bash

useradd $1 &> /dev/null

echo $1创建成功

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

echo $1密码设置成功

[root@server0 ~]# /root/user.sh nsd20

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

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

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

#!/bin/bash

cat -n $1 | head -$2

[root@server0 ~]# chmod +x /root/1.sh

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

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

2 bin:x:1:1:bin:/bin:/sbin/nologin

3 daemon:x:2:2:daemon:/sbin:/sbin/nologin

### 预定义变量

预定义变量:由系统定义完成,用户直接使用

用来保存脚本程序的执行信息

-直接使用这些变量

-不能直接为这些变量赋值

$0 已运行的脚本名

$# 已加载的位置变量的个数(计算用户输入位置变量的个数)

$\* 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

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

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

#!/bin/bash

echo $1

echo $2

echo $3

echo $# #输出一共有多少个参数(求和)

echo $\* #输出所有位置变量的值

自定义变量:用户自主设置、修改及使用

单引号’’:取消所有特殊字符的含义

# 条件测试及选择

## 条件测试

### Shell测试的依据

### 测试操作

### 常用的测试选项

检查文档状态

-e:文档存在为真

-d:文档存在,且为目录才为真

-f:文档存在,且为文件才为真

-r:文档存在,且具备读取权限才为真

-w:文档存在,且具备写入权限才为真

-x:文档存在,且具备执行权限才为真

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

[root@server0 ~]# echo $?

[root@server0 ~]# [ -d /home ]

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

[root@server0 ~]# [ -f /home ]

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

比较整数大小

-gt:大于

-ge:大于等于

-eq:等于

-ne:不等于

-lt:小于

-le:小于等于

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

[root@server0 ~]# echo $?

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

[root@server0 ~]# echo $?

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

[root@server0 ~]# echo $?

字符串比对

==:两个字符串相等为真

!=:两个字符串不相等为真

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

[root@server0 ~]# echo $?

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

[root@server0 ~]# echo $?

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

[root@server0 ~]# echo $?

## If选择结构

### If单分支处理

### If双分支处理

If [条件测试];then

命令序列xx

else

命令序列yy

Fi

案例:

利用read实现交互

用户输入一个用户名,脚本判断用户所输入的用户,是否存在

如果存在,输出用户存在

如果不存在,输入用户不存在

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

#!/bin/bash

read -p '请输入您要测试的用户名:' user

id $user &> /dev/null

if [ $? -eq 0 ];then #判断上一条命令是否成功

echo '用户已存在'

else

echo '用户不存在'

fi

案例:

利用位置变量实现命令行传递参数

参数为IP地址,脚本判断用户所输入IP地址,是否本机可以ping通

如果可以ping,输出 可以通信

如果不可以ping,输出 不可以通信

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

#!/bin/bash

ping -c 2 $1 &> /dev/null

if [ $? -eq 0 ];then

echo '本机可以通信'

else

echo '本机不可以通信'

fi

### If多分支处理

If [条件测试1];then

命令序列xx

elif [条件测试2];then

命令序列yy

elif [条件测试3];then

命令序列cc

......

else

命令序列zz

fi

案例:

利用read交互,让用户输入考试成绩

进行如下判断

如果成绩大于90 则 输出 优秀

如果成绩大于80 则 输出 良好

如果成绩大于70 则 输出 一般

如果成绩大于60 则 输出 合格

以上条件均不满足 , 则 输出 不合格

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

#!/bin/bash

read -p '请输入考试成绩:' num

if [ $num -gt 90 ];then

echo '优秀'

elif [ $num -gt 80 ];then

echo '良好'

elif [ $num -gt 70 ];then

echo '一般'

elif [ $num -gt 60 ];then

echo '合格'

else

echo '不合格'

fi

[root@server0 ~]# /root/if02.sh 192.168.1.1

优化:

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

#!/bin/bash

if [ $# -eq 0 ];then

echo '格式错误,请在脚本运行时添加一个参数'

echo '参数为IP地址'

else

ping -c 2 $1 &> /dev/null

if [ $? -eq 0 ];then

echo '本机可以通信'

else

echo '本机不可以通信'

fi

fi

案例:Shell脚本

为系统server0书写脚本/root/user.sh

运行脚本,可以判断用户输入的用户是否存在

如果存在,输出用户基本信息(id 用户名)

如果用户,不存在则创建用户,并输出用户创建成功.

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

#!/bin/bash

read -p '请输入用户:' user

id $user &> /dev/null

if [ $? -eq 0 ];then

echo '用户已存在,id:'$user

else

useradd $user

echo '用户已创建成功'

fi

#!/bin/bash

if [ $# -eq 0 ];then #匹配用户没有输入参数

echo '/root/foo.sh redhat|fedora' >&2 #转变为错误输出

exit 3 #程序退出返回值

elif [ $1 = 'redhat' ];then

echo 'fedora'

elif [ $1 = 'fedora' ];then

echo 'redhat'

else

echo '/root/foo.sh redhat|fedora' >&2 #转变为错误输出

exit 4 #程序退出返回值

fi

# 列表式循环

## For循环结构

### 列表式循环场景

### For循环处理

循环:让计算机重复执行一个操作

for 变量名 in 值列表

do

命令序列

done

for a in zhangsan lisi wangwu dc

do

useradd $a

echo 123 | passwd --stdin $a

done

造数: {起始数值..结束数值}

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

#!/bin/bash

for a in {1..20}

do

useradd stu$a &> /dev/null

echo stu$a'创建成功'

done

### 利用命令替换取值

$()或反撇号``:

将命令的输出,作为另一个命令的参数,参与下一个命令的运行

[root@server0 ~]# date +%F

[root@server0 ~]# cd /mnt

[root@server0 mnt]# mkdir nsd-`date +%F`

[root@server0 mnt]# ls

[root@server0 mnt]# mkdir $(hostname)-$(date +%F)

[root@server0 mnt]# ls

案例:编写一个批量添加用户脚本

在server0上创建/root/batchusers脚本

1. 此脚本要求提供用户名列表文件作为参数
2. 如果没有提供参数,此脚本应该给出提示

usage:/root/batchusers,退出并返回相应值

1. 如果提供一个不存在的文件,此脚本应该给出提示Input file not found,退出并返回相应值
2. 新用户的登录shell为/bin/false,无需设置密码
3. 用户列表测试文件:

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

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

#!/bin/bash

if [ $# -eq 0 ];then

echo 'usage:/root/batchusers' >&2 #将输出变为错误输出

exit 17

elif [ -f $1 ];then #判断命令行参数是否为文件

for user in `cat $1` #将命令的输出作为列表

do

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

echo '用户'$user'创建成功'

done

else

echo 'Input file not found' >&2

exit 18

fi

# 总结和答疑

## 脚本语法错误

### 问题现象

### 故障分析及排除

### 

# 系统安全保护

## Selinux安全机制

### 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运行模式的切换

SELinux的运行模式

-enforcing(强制)、permissive(宽松)

-disabled(彻底禁用)

变成disabled,都必须经过修改配置文件,重启才可以

切换运行模式

-临时切换:setenforce 1|0

-固定配置:/etc/selinux/config 文件

虚拟机server和虚拟机desktop

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

Enforcing

[root@server0 ~]# setenforce 0

[root@server0 ~]# getenforce

Permissive

[root@server0 ~]# vim /etc/selinux/config

SELINUX=permissive

# 配置用户环境

## 自定义命令

### Linux命令字的来源

### Alias别名设置

## 用户初始化文件

### 用户个性化配置文件

影响指定用户的bash解释环境

- ~/.bashrc,每次开启bash终端时生效

### 全局环境配置

影响所有用户的bash解释环境

- /etc/bashrc,每次开启bash终端时生效

# 配置高级连接

## 配置IPv6地址

### IPv6地址的组成

IPv4:32个二进制,用 点 分隔 四 部分,转换为4个十进制

IPv6:128个二进制,用 冒号 分隔 八 部分,转换每部分4个16进制数

-每段内连续的前置0可省略、连续的多个:可简化为::

[root@room9pc01 ~]# host www.qq.com

https.qq.com has IPv6 address 240e:e1:8100:28::2:1f

[root@room9pc01 ~]# ping6 240e:e1:8100:28::2:1f

### Nmcli命令行配置

[root@server0 ~]# nmcli connection modify 'System eth0' ipv6.method manual ipv6.addresses 2003:ac18::305/64 connection.autoconnect yes

[root@server0 ~]# nmcli connection up ‘System eth0’

[root@desktop0 ~]# ping6 2003:ac18::305

## 配置聚合连接

### 链路聚合的优势

### 实现链路聚合的条件

配置聚合连接(网卡绑定、链路聚合)

eth1 eth2

虚拟网卡team(组队)

### Nmcli命令行配置

1.创建 虚拟网卡

[root@room9pc01 ~]# man teamd.conf #查看帮助信息

[root@server0 ~]# nmcli connection add type team con-name team0 ifname team0 autoconnect yes config '{"runner": {"name": "activebackup"}}'

[root@server0 ~]# ifconfig #查看是否有 team0网卡

[root@server0 ~]# nmcli connection 添加 类型为 team(绑定类型)

配置文件名 team0 ifname显示网卡名 team0 每次开机自启 配置网卡绑定工作模式 热备份方式

# 如果敲错误

# nmcli connection delete team0

1. 添加成员(添加奴隶)

[root@server0 ~]# nmcli connection add type team-slave con-name team0-1 ifname eth1 master team0

[root@server0 ~]# nmcli connection add type team-slave con-name team0-2 ifname eth2 master team0

[root@server0 ~]# nmcli connection 添加 类型为team-slave配置文件名 team0-1 网卡为eth1 添加到team0中

#如果敲错误 nmcli connection delete team0-1

1. 配置team0的IP地址

[root@server0 ~]# nmcli connection modify team0 ipv4.method manual ipv4.addresses 192.168.1.1/24 connection.autoconnect yes

### 激活聚合连接

1. 激活所有配置

[root@server0 ~]# nmcli connection up team0

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

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

如果激活失败

[root@server0 ~]# nmcli connection delete team0

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

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

终极验证:

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

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

[root@server0 ~]# teamdctl team0 state

# 防火墙策略管理

## Firewalld服务基础

基本web服务

1. 服务器虚拟机server,安装可以提供web服务软件

Apache(httpd) Nginx Tomcat

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

1. 虚拟机server操作,启动httpd服务,设置为开机自启服务

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

[root@server0 ~]# systemctl enable httpd #设置开启自启服务

1. 虚拟机server操作,本机测试访问

[root@server0 ~]# firefox 172.25.0.11

1. 虚拟机server操作,书写页面文件

默认存放页面文件的路径:/var/www/html

默认页面文件:index.html

写网站语言:html

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

<marquee><font color=red><h1>nsd1812

滚动 字体颜色 红色 最大字体

基本FTP服务:文件传输协议

http:超文本传输协议

1. 服务端虚拟机server,安装可以提供FTP服务软件

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

1. 虚拟机server操作,启动vsftpd服务,设置为开机自启服务

[root@server0 ~]# systemctl restart vsftpd

[root@server0 ~]# systemctl enable vsftpd

1. 测试:

默认FTP共享的路径:/var/ftp

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

### RHEL7的防火墙体系

防火墙策略管理

作用:隔离,过滤入站请求,允许出站

硬件防火墙

软件防火墙

系统服务:firewalld

管理工具:firewall-cmd、firewall-config(图形工具)

### 预设安全区域

匹配规则的原则:匹配即停止原则

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

### 查看防火墙规则列表

## 配置防火墙

### 指定默认的安全区域

防火墙决定,客户端请求进入某个区域的规则:

1. 查看客户端请求中源IP地址,再看所有区域中,哪一个区域有该源IP地址的规则,则进入该区域
2. 进入默认区域,public

互联网常见的服务协议

http:超文本传输协议 80

FTP:文件传输协议 21

https:安全的超文本传输协议 443

DNS:域名解析协议 53 TCP UDP

telnet:远程管理协议 23

smtp:邮件协议,用户发邮件的协议 25 TCP

pop3:邮件协议,用户收邮件的协议 110 TCP

tftp:简单文件传输协议 69

snmp:简单的网络管理协议 161

imap 143 TCP

win远程 3389 TCP

dhcp 67 udp服务端 68客户端

mysql 3306 TCP

默认区域的修改

虚拟机server

[root@server0 ~]# firewall-cmd --get-default-zone #查看默认区域

public

虚拟机desktop

[root@desktop0 ~]# ping -c 2 172.25.0.11 #可以通信

虚拟机server

[root@server0 ~]# firewall-cmd --set-default-zone=block #修改默认区域

[root@server0 ~]# firewall-cmd --get-default-zone #查看默认区域

虚拟机desktop

[root@desktop0 ~]# ping -c 2 172.25.0.11 #不可以通信,有回应

虚拟机server

[root@server0 ~]# firewall-cmd --set-default-zone=drop #修改默认区域

[root@server0 ~]# firewall-cmd --get-default-zone #查看默认区域

虚拟机desktop

[root@desktop0 ~]# ping -c 2 172.25.0.11 #不可以通信,没有回应

默认区域服务的添加

虚拟机server

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

[root@server0 ~]# firewall-cmd --zone=public --list-all #查看区域规则

[root@server0 ~]# firewall-cmd --zone=public --add-service=http #添加服务

[root@server0 ~]# firewall-cmd --zone=public --list-all

虚拟机desktop

[root@desktop0 ~]# firefox http://172.25.0.11 #可以访问

[root@desktop0 ~]# firefox ftp://172.25.0.11 #不可以访问

虚拟机server

[root@server0 ~]# firewall-cmd --zone=public --add-service=ftp

[root@server0 ~]# firewall-cmd --zone=public --list-all

虚拟机desktop

[root@desktop0 ~]# firefox http://172.25.0.11 #可以访问

[root@desktop0 ~]# firefox ftp://172.25.0.11 #可以访问

策略的永久配置

-永久(permanent)

[root@server0 ~]# firewall-cmd --reload #重新加载防火墙配置,模拟重启机器

[root@server0 ~]# firewall-cmd --zone=public --list-all

[root@server0 ~]# firewall-cmd --permanent --zone=public --add-service=http

[root@server0 ~]# firewall-cmd --permanent --zone=public --add-service=ftp

[root@server0 ~]# firewall-cmd --zone=public --list-all

[root@server0 ~]# firewall-cmd --reload #重新加载防火墙配置

[root@server0 ~]# firewall-cmd --zone=public --list-all

添加源IP的规则设置

虚拟机server

[root@server0 ~]# firewall-cmd --zone=block --list-all

[root@server0 ~]# firewall-cmd --zone=block --add-source=172.25.0.10

[root@server0 ~]# firewall-cmd --zone=block --list-all

虚拟机desktop

[root@desktop0 ~]# ping -c 2 172.25.0.11 #失败

[root@desktop0 ~]# firefox 172.25.0.11 #失败

[root@desktop0 ~]# firefox ftp://172.25.0.11 #失败

真机访问

[root@desktop0 ~]# ping -c 2 172.25.0.11 #成功

[root@desktop0 ~]# firefox 172.25.0.11 #成功

[root@desktop0 ~]# firefox ftp://172.25.0.11 #成功

工作时防火墙,设置的方式

严格:默认区域为drop,把允许的IP单独放入trusted

宽松:默认区域为trusted,把拒绝的IP单独放入drop

### 封网段、开服务

### 实现本机的端口映射

端口:编号,标识作用,标识每个服务

本地应用的端口重定向(端口1-->端口2)

-从客户机访问端口1的请求,自动映射到本机 端口2

-比如,访问以下两个地址可以看到相同的页面

客户端desktop----->172.25.0.11:5423----->服务端server

服务端server-----172.25.0.11:5423移交-----172.25.0.11:80

虚拟机server

[root@server0 ~]# firewall-cmd --reload

虚拟机desktop

[root@desktop0 ~]# firefox 172.25.0.11 #可以访问

[root@desktop0 ~]# firefox 172.25.0.11:5423 #不可以访问

[root@server0 ~]# firewall-cmd --permanent --zone=public --add-forward-port=port=5423:proto=tcp:toport=80

#添加 - 转发 - 端口 = 将端口5423 协议为tcp 转发到80

[root@server0 ~]# firewall-cmd --reload

[root@server0 ~]# firewall-cmd --zone=public --list-all

虚拟机desktop

[root@desktop0 ~]# firefox 172.25.0.11:5423 #可以访问

# 总结和答疑

## 配置IPv6地址

### 问题现象

### 故障分析及排除

## 配置聚合连接

### 问题现象

### 故障分析及排除

# 配置SMB共享

## Samba服务基础

### Samba服务基础

配置SMB共享,跨平台的共享,Windows与Linux的共享

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. 修改配置文件/etc/samba/smb.conf

[自定共享名]

path=文件夹绝对路径

public = no|yes //默认no

browseable = yes|no //默认yes

read only = yes|no //默认yes

write list = 用户1 .. .. //默认无

valid users = 用户1 .. .. //默认任何用户

hosts allow = 客户机地址 .. ..

hosts deny = 客户机地址 .. ..

### 管理共享账号

Samba共享账号:默认情况下,访问Samba共享必须通过用户验证

专门用于访问Samba共享时验证的用户与密码

与系统用户为同一个用户,但是密码是samba独立密码

服务端,虚拟机server:

1. 安装可以提供smb共享功能的软件

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

[root@server0 ~]# useradd harry

[root@server0 ~]# useradd kenji

[root@server0 ~]# useradd chihiro

2.建立Samba共享验证的用户

[root@server0 ~]# pdbedit -a harry #添加为Samba共享账号

[root@server0 ~]# pdbedit -a kenji #添加为Samba共享账号

[root@server0 ~]# pdbedit -a chihiro #添加为Samba共享账号

[root@server0 ~]# pdbedit -L #查看所有Samba共享账号

### 配置文件及参数

3.创建共享目录与文件

[root@server0 ~]# mkdir /common

[root@server0 ~]# ls /common/

[root@server0 ~]# echo 123 > /common/123.txt

4. 修改配置文件/etc/samba/smb.conf

vim 末行模式 :set nu #添加行号

命令模式 G 到全文的最后

[global]

workgroup = STAFF #指定工作组名

[common] #指定共享名

path = /common #指定共享文件夹的实际绝对路径

1. 重启smb服务,刷新配置

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

[root@server0 ~]# systemctl enable smb #设置为开机自启

### Selinux对SMB共享的保护

6.服务端,虚拟机Server:

-需要加-P选项才能实现永久设置(需要较大的内存)

SELinux策略:布尔值(所服务功能的开关)

1.查看samba布尔值

[root@server0 ~]# getsebool -a | grep samba

2.修改SELinux策略的布尔值

[root@server0 ~]# setsebool samba\_export\_all\_ro on

3.查看samba布尔值

[root@server0 ~]# getsebool -a | grep samba

## 访问共享文件夹

### 使用smbclient测试

客户端虚拟机Desktop:

1. 安装客户端软件,访问samba共享

[root@desktop0 ~]# yum -y install samba-client

2.查看服务端samba共享,目的看 共享名

[root@desktop0 ~]# smbclient -L //172.25.0.11

Enter root's password: #直接敲回车

Sharename

---------

common

1. 以harry身份,访问服务端samba共享

[root@desktop0 ~]# smbclient -U harry //172.25.0.11/common

Enter harry's password: #输入密码

Domain=[STAFF] OS=[Unix] Server=[Samba 4.1.1]

smb: \>

### 使用mount挂载

客户端虚拟机desktop

使用mount挂载访问

所需软件包:cifs-utils #支持cifs协议

[root@desktop0 ~]# yum -y install cifs-utils.x86\_64

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

[root@desktop0 ~]# mount -o user=harry,pass=123 //172.25.0.11/common /mnt/samba

[root@desktop0 ~]#df -h

开机自动挂载

\_netdev:网络设备

先启动网络服务,具备网络参数后,再进行挂载

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

//172.25.0.11/common /mnt/samba cifs

defaults,user=harry,pass=123,\_netdev 0 0

[root@desktop0 ~]# mount -a

[root@desktop0 ~]# df -h #查看是否挂载成功

读写的Samba共享

服务端虚拟机server:

1.部署共享

[root@server0 ~]# mkdir /devops

[root@server0 ~]# echo nsd > /devops/test.txt

[root@server0 ~]# ls /devops

[root@server0 ~]# vim /etc/samba/smb.conf #修改配置文件

追加写入

[devops] #共享名

path = /devops #路径为/devops

write list = chihiro #允许chihiro用户可写

[root@server0 ~]# systemctl restart smb

2.修改SELinux策略

[root@server0 ~]# getsebool -a | grep samba

[root@server0 ~]# setsebool samba\_export\_all\_rw on

[root@server0 ~]# getsebool -a | grep samba

1. 用户本身的本地权限

[root@server0 ~]# setfacl -m u:chihiro:rwx /devops

[root@server0 ~]# getfacl /devops

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

客户端虚拟机desktop

1. 实现开机自动挂载

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

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

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

[root@desktop0 ~]# mount -a

[root@desktop0 ~]# df -h

**总结:**客户端访问服务端资源

1. 服务本身的访问控制
2. 本地目录的权限
3. 防火墙
4. SELinux

## Multiuser机制

### 共享文件夹的多用户访问

multiuser机制,专门为普通用户设计,专为客户端设计

mount.cifs的挂载参数

-multiuser,提供对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

在客户端挂载时,利用权限较小的用户进行验证挂载,必要的时候,任何普通用户都可以通过命令切换成权限较大的用户来临时获取写的权限

使用cifscerds提交新的用户凭据并测试

- cifscerds add|update -u共享用户名 服务器地址

### 挂载参数调整

### 切换共享用户身份

客户端虚拟机Desktop:

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

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

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

[root@desktop0 ~]# mount -a

[root@desktop0 ~]# su -student

[student@desktop0 ~]$ cifscreds add -u chihiro 172.25.0.11

[student@desktop0 ~]$ ls /mnt/pub

[student@desktop0 ~]$ exit

# 配置NFS共享

## 普通NFS服务

### NFS共享概述

配置只读的NFS共享,Linux与Linux的共享

Network Flie 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

虚拟机server

1. 安装软件包

[root@server0 ~]# yum -y install nfs-utils

已加载插件：langpacks

软件包 1:nfs-utils-1.3.0-0.el7.x86\_64 已安装并且是最新版本

无须任何处理

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

nfs-utils-1.3.0-0.el7.x86\_64 //一般默认会安装

1. 修改 /etc/exports

-文件夹路径 客户机地址(权限) 客户机地址(权限) .. ..

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

/public \*(ro)

[root@server0 ~]# mkdir /public

[root@server0 ~]# echo 123 > /public/1.txt

[root@server0 ~]# ls /public/

1.txt

使用exportfs可以重载更新过的配置

-exportfs -r

1. 重启nfs-server服务

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

[root@server0 ~]# systemctl enable nfs-server

ln -s '/usr/lib/systemd/system/nfs-server.service'

'/etc/systemd/system/nfs.target.wants/nfs-server.service'

客户端:虚拟机desktop

1.

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

[root@desktop0 ~]# mount 172.25.0.11:/public /mnt/pub

[root@desktop0 ~]# df -h

1. 开机自动挂载

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

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

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

[root@desktop0 ~]# df -h

[root@desktop0 ~]# mount -a

[root@desktop0 ~]# df -h

### Exports配置文件解析

### 访问NFS共享文件夹

# 总结和答疑

## SMB访问失败

### 问题现象

挂载server0共享的devops资源失败

-报错1:mount:wrong fs type,bad option .. ..

-报错2:mount error(13):Permission denied .. ..

-报错3:Couldn’t chdir to /mnt/dev .. ..

#mount -a

mount:wrong fs type,bad option,bad superblock on .. ..

#mount -a

mount error(13):Permission denied

Refer to the mount.cifs(8) manual page (e.g. man mount.cifs)

#mount -a

Couldn’t chdir to /mnt/dev:No such file or directory

### 故障分析及排除

原因分析

-问题1:未装cifs-utils包,导致不识别CIFS文件系统

-问题2:未创建/未指定或指错共享用户及密码

-问题3:未创建挂载点目录 /mnt/dev

解决办法

-问题1:先安装cifs-utils包

-问题2:检查用户名、密码及挂载参数设置

-问题3:执行mount -a之前,确保挂载点目录都可用

## Selinux策略故障

### 问题现象

配置SMB共享服务时,访问异常

-能成功连入Samba共享资源,但ls列表失败

-执行setsebool -P .. .. 操作启用SELinux开关参数时失败,提示:Killed(或已杀死)

### 故障分析及排除

原因分析

-问题1:对文件夹权限不足,或者存在SELinux限制

-问题2:内存不足,而且交换空间也不足

解决办法

-问题1:调整文件夹的权限,并打开SELinux开关

-问题2:添加交换分区(1GB左右)再重试

### 

# Iscsi网络磁盘

## Iscsi服务基础

环境准备:设置防火墙区域为trusted

虚拟机server

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

虚拟机desktop

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

### Iscsi磁盘的工作模式

Internet SCSI,网络SCSI接口 默认端口3260

-一种基于C/S架构的虚拟磁盘技术

-服务器提供磁盘空间,.客户机连接并当成本地磁盘使用

### Iscsi磁盘的构成

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

ISCSI Qualified Name iqn名称规范

-iqn.yyyy-mm.倒序域名:自定义标识

-用来识别target磁盘组,也用来识别客户机身份

## 发布iscsi磁盘

### 准备空闲存储设备

服务端:虚拟机server

一 划分新分区

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

[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

└─vdb1 253:17 0 5G 0 part

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

### 使用targetcli建立配置

二 安装软件包,进行配置

1. 安装服务端软件targetcli

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

1. 进行配置

[root@server0 ~]# targetcli

/> ls #查看支持的配置

A:建立backstore,后端存储,并且设置后端存储的名字

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

Created block storage object nsd using /dev/vdb1.

/> ls

B:建立target磁盘组(木质的箱子)

/> iscsi/ create iqn.2016-02.com.example:server0

iqn.yyyy-mm.倒序域名:自定义标识

/> ls

/> exit

[root@server0 ~]# targetcli

C:lun进行关联(将后端存储,放入target磁盘组)

/>iscsi/iqn.2016-02.com.example:server0/tpg1/luns

create /backstores/block/nsd

Created LUN 0.

/> ls

D:设置ACL访问控制,设置客户端声称的名字

/>iscsi/iqn.2016-02.com.example:server0/tpg1/acls

create iqn.2019-01.com.example:desktop0

/> ls

E:开启端口,设置在IP地址提供服务

/> iscsi/iqn.2016-02.com.example:server0/tpg1/portals create 172.25.0.11

/> exit

### 确认发布结果

三 重启服务

[root@server0 ~]# systemctl restart target

[root@server0 ~]# systemctl enable target

## 访问iscsi磁盘

### 客户机配置IQN名称

客户端:虚拟机desktop

一 安装软件包

所需软件包:iscsi-initiator-utils

yum 可以补全包名:

前提1.该软件包没有安装

前提2:必须要有yum缓存

yum repolist #生成缓存

yum -y install iscsi-(tab)

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

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

[root@desktop0 ~]# yum search iscsi #查询yum中包含iscsi的包

二 进行配置

1.指定配置文件指定客户端声称的名字

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

InitiatorName=iqn.2019-01.com.example:desktop0

1. 重启iscsid服务,刷新客户端声称的名字

[root@desktop0 ~]# systemctl restart iscsid

[root@desktop0 ~]# systemctl daemon-reload

[root@desktop0 ~]# systemctl restart iscsid

### 连接、发现iscsi磁盘

三 发现与加载共享存储

1. 发现共享存储

Linux查看命令帮助的方法:man iscsiadm

全文查找/example 按n向下跳转匹配

ctrl -:减小字体

ctrl shift + :变大字体

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

172.25.0.11:3260,1 iqn.2016-02.com.example:server0

### 开机启动配置

1. 加载共享存储

[root@desktop0 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

[root@desktop0 ~]# systemctl restart iscsi

[root@desktop0 ~]# systemctl enable iscsi

[root@desktop0 ~]# lsblk

### Iscsi设备的挂载设备

# 数据库服务基础

## 构建数据库系统

### 什么是数据库

数据库:存放数据的仓库

在数据库中有众多的库,在每一个库中会有许多的表格

常见的关系型 数据库管理系统

-微软的SQL Server

-IBM的DB2

-甲骨文的Oracle、MySQL

-社区开源版 MariaDB

### 部署mariadb数据库服务器

一 部署MariaDB数据库服务器

1. 安装软件mariadb-server:提供服务端有关的系统程序

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

1. 重启mariadb服务

[root@server0 ~]# systemctl restart mariadb

[root@server0 ~]# systemctl enable mariadb

### 访问mariadb数据库

二 mariadb数据库简单操作

[root@server0 ~]# mysql

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

## 基本管理操作

### Mariadb服务端配置调整

监听端口3306

### 数据库的增删查

MariaDB [(none)]> create database nsd; #创建nsd库

MariaDB [(none)]> show databases;

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

MariaDB [(none)]> show databases;

MariaDB [(none)]> create database nsd1812; #创建nsd1812库

MariaDB [(none)]> show databases;

MariaDB [(none)]> use nsd1812; #进入nsd1812

MariaDB [nsd1812]> show tables; #查看当前库的所有表格

MariaDB [nsd1812]> show databases; #显示数据库中所有的库

MariaDB [nsd1812]> use mysql; #进入mysql库

MariaDB [mysql]> show tables; #显示当前库的所有表格

### 数据库的导出/导入操作

导入/恢复到数据库

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

1. 下载备份文件

[root@server0 ~]# wget

http://classroom.example.com/pub/materials/users.sql

[root@server0 ~]# ls

2.将数据导入nsd1812库中

[root@server0 ~]# mysql -u root -p123 nsd1812 < users.sql

#将数据导入nsd1812库中

3.进入数据库查看表格是否生成

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

MariaDB [(none)]> use nsd1812; #切换到nsd1812库

MariaDB [mysql]> show tables; #显示所有表格

+-------------------+

| Tables\_in\_nsd1812 |

+-------------------+

| base |

| location |

+-------------------+

### 用户授权设置

为数据库管理员设置密码

数据库管理员:用户名为root,用户信息放在mysql库user表

系统管理员:用户名为root,用户信息放在/etc/passwd

为数据库账号修改密码

-mysqladmin [-u用户名] [-p[旧密码]] password ‘新密码’

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

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

Enter password: 123

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

MariaDB主配置文件:/etc/my.cnf

用户授权设置

MariaDB [(none)]> 交互指令

-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’;

-除了root用户,此nsd1812数据库只能被用户lisi查询,

此用户的密码为123

grant select on nsd1812.\* to lisi@localhost identified by ‘123’;

当lisi从本地localhost进行登录数据库,输入密码123,将会获得nsd1812库中所有表的查询权限

MariaDB [(none)]> grant select on nsd1812.\* to lisi@localhost identified by '123';

# 管理表数据

## 数据表的增删查

表格操作:insert(增) delete(删) update(更改) select(查)

表记录

表字段

### 创建新的数据表

### 查看表结构

### 删除数据表

## 表记录基本操作

### Insert插入表记录

### Select查询表记录

表查询的操作:

select 表字段 from 库名.表名

MariaDB [nsd1812]>show tables; #显示当前库所有表格

MariaDB [nsd1812]> select \* from base;

MariaDB [(none)]> select \* from nsd1812.base;

MariaDB [(none)]> select \* from nsd1812.location;

MariaDB [nsd1812]> select name from base;

MariaDB [nsd1812]> select ID,name from base;

查看表结构:desc 表名

MariaDB [nsd1812]>desc base;

MariaDB [nsd1812]> desc location;

MariaDB [nsd1812]> use mysql ;

MariaDB [mysql]> desc user;

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

查询base表 密码为123的人是谁?

select 表字段 from 库名.表名 where 字段名=’值’

MariaDB [nsd1812]> select \* from base where password='123';

MariaDB [nsd1812]> select \* from base where password='456';

MariaDB [nsd1812]> select \* from base where name='james';

MariaDB [nsd1812]> select id,name from base where password='123';

### Update修改表记录

MariaDB [(none)]>交互指令

-UPDATE [数据库名.]表名 SET 字段名=新值 WHERE 条件语句;

### Delete删除表记录

1. 禁止空密码root用户访问mariadb数据库

MariaDB [(none)]> use mysql;

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

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

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

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

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

MariaDB [mysql]> flush privileges; #刷新user表所有记录

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

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

删除数据表

MariaDB [(none)]>交互指令

-DROP TABLE 表名;

### 多表组合查询

使用数据库查询

1. 在系统server0上使用数据库nsd1812,并使用相应的sql查询回答下列问题;
2. 密码是solicitous的人的名字?

MariaDB [mysql]> use nsd1812;

MariaDB [nsd1812]> select \* from base;

MariaDB [nsd1812]> select \* from base where password='solicitous';

MariaDB [nsd1812]> select id,name from base where password='solicitous';

1. 有多少人的姓名是Barbara同时居住在Sunnyvale?

MariaDB [nsd1812]> select \* from base,location where base.name='Barbara' and location.city='Sunnyvale' and base.id=location.id ;

MariaDB [nsd1812]> select count(\*) from base,location where base.name='Barbara' and location.city='Sunnyvale' and base.id=location.id ;

MariaDB [nsd1812]> insert base values ('6','Barbara','789');

MariaDB [nsd1812]> select \* from base;

MariaDB [nsd1812]> insert location values ('6','Sunnyvale');

MariaDB [nsd1812]> select \* from location;

# 总结和答疑

**总结:搭建ISCSI共享存储**

服务端:

1. 修改防火墙默认区域trusted
2. 划分新的分区
3. 安装软件包targetcli
4. 运行targetcli进行配置
5. 建立后端存储
6. 建立target磁盘组
7. 建立关联
8. 设置ACL访问控制
9. 开启IP地址与端口
10. 重启target服务

客户端:

1.修改防火墙默认区域trusted

2.安装软件包iscsi-initiator-utils

3.修改配置文件,指定客户端声称的名字,重启iscsid服务

4.运行命令进行发现共享存储(参考man iscsiadm)

5.重启iscsi 加载共享存储

## Mariadb数据库

### 问题现象

### 故障分析及排除

### 

# http服务基础

## 独立web主机

### Web通信基本概念

基于B/S(Browser/Server)架构的网页服务

-服务端提供网页

-浏览器下载并显示网页

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

Hyper Text Transfer Protocol(http) 超文本传输协议 80端口

### RHEL7中的web服务

软件包:httpd

系统服务:httpd

提供的默认配置

– Listen:监听地址:端口(80)

– ServerName:本站点注册的DNS名称(空缺)

– DocumentRoot:网页文件的根目录(/var/www/html)

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

搭建基本的Web

虚拟机Server:

1. 安装httpd软件

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

1. 重启httpd服务

[root@server0 ~]# systemctl restart httpd

[root@server0 ~]# systemctl enable httpd

1. 书写一个页面文件

[root@server0 ~]# echo '<h1>My First Web' >/var/www/html/index.html

[root@server0 ~]# firefox 172.25.0.11

虚拟机Desktop

[root@server0 ~]# firefox 172.25.0.11

-DocumentRoot:网页文件的根目录(/var/www/html)

根目录=起始点

修改存放网页文件的路径为/var/www/wyweb

虚拟机Server0:

1. 修改配置文件/etc/httpd/conf/httpd.conf

vim /etc/httpd/conf/httpd.conf

DocumentRoot “/var/www/wyweb”

1. 重启httpd服务

[root@server0 ~]# systemctl restart httpd

[root@server0 ~]# systemctl enable httpd

1. 建立存放网页文件路径,书写新的页面文件用于测试

[root@server0 ~]# mkdir /var/www/myweb

[root@server0 ~]# echo '<h1>wo shi myweb' > /var/www/myweb/index.html

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

<h1>wo shi myweb

[root@server0 ~]# systemctl restart httpd

虚拟机Desktop:

[root@desktop0 ~]# firefox 172.25.0.11

网络路径与实际路径对照

客户端:firefox httpd://172.25.0.11:80 --->服务端 httpd---->DocumentRoot /var/www/myweb----> /var/www/myweb

客户端: firefox 172.25.0.11

服务端: /var/www/myweb

客户端: firefox 172.25.0.11/var/www/myweb/abc

服务端: /var/www/myweb/var/www/myweb/abc

虚拟机Server0:

[root@server0 ~]# mkdir /var/www/myweb/abc

[root@server0 ~]# echo '<h1>wo shi abc' >/var/www/myweb/abc/index.html

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

<h1>wo shi abc

虚拟机Desktop0:

[root@server0 ~]# firefox 172.25.0.11/abc/

### 独立web站点的快速部署

### 访问web站点

## 虚拟web主机

### 虚拟主机的含义及类型

-由同一台服务器提供多个不同的Web站点

区分方式

-基于域名的虚拟主机

-基于端口的虚拟主机

-基于IP地址的虚拟主机

### 配置一个虚拟站点

解决域名:DNS服务器能够解析的域名 classroom.example.com

server0.example.com www0.example.com webapp0.example.com

基于域名的构建虚拟Web主机

配置文件路径

-/etc/httpd/conf/httpd.conf(主配置文件)

-/etc/httpd/conf.d/\*.conf(调用配置文件)

为每个虚拟站点添加配置

<VirtualHost IP地址:端口>

ServerName 此站点的DNS名称

DocumentRoot 此站点的网页根目录

</VirtualHost>

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

<VirtualHost \*:80> ＃启用虚拟Web主机功能　\*：本机任意IP

ServerName www0.example.com #指定访问域名

DocumentRoot /var/www/qq #指定网页文件存放路径

</VirtualHost>

<VirtualHost \*:80>

ServerName webapp0.example.com #指定访问域名

DocumentRoot /var/www/baidu #指定网页文件存放路径

</VirtualHost>

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

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

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

[root@server0 ~]# systemctl restart httpd

一旦使用了虚拟Web主机功能,所有的网站都要用虚拟Web主机来呈现

### 对默认web站点的影响

# 网页内容访问

Apache网页内容访问

针对于网页文件存放路径进行访问控制

## 配置目录访问

### 文件夹权限

使用 <Directory> 配置区段

-每个文件夹自动继承其父目录的ACL访问权限

-除非针对子目录有明确设置

<Directory 目录的绝对路径>

Require all dinied|granted

Require ip IP或网段地址.. ..

</Directory>

<Directory /var/www/myweb>

Require ip 172.25.0.11 #仅允许172.25.0.11访问

</Directory>

<Directory /opt>

Require all denied #拒绝所有人访问

</Directory>

<Directory /opt/abc>

Require all granted #允许所有人访问

</Directory>

不能访问 /opt/bcd 路径下页面

能访问 /opt/abc 路径下页面

案例3:配置网页内容访问

在Web网站 <http://server0.example.com的Document> 目录下创建一个名为 private 的子目录,要求如下:

1. 查看网站 <http://server0.example.com> 的DocumentRoot

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

[root@server0 ~]# mkdir /var/www/myweb/private

[root@server0 ~]# echo '<h1>wo shi private' > /var/www/myweb/private/index.html

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

1. 从 http://classroom/pub/materials/private.html下载一个文件副本到这个目录,重命名为index.html
2. 不要对文件 index.html 的内容作任何修改
3. 从虚拟机server0上,任何人都可以浏览private的内容,但是从其他系统不能访问这个目录的内容.

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

[root@server0 ~]# vim /etc/httpd/conf.d/nsd02.conf

<Directory /var/www/myweb/private>

Require ip 172.25.0.11

</Directory>

[root@server0 ~]# systemctl restart httpd

虚拟机Desktop0:访问测试报错

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

Forbidden

You don't have permission to access /private on this server.

案例4:使用自定Web根目录

调整Web站点 <http://server0.example.com> 的网页目录,要求如下:

1. 新建目录 /webroot,作为此站点新的网页文件目录

[root@server0 ~]# mkdir /webroot

[root@server0 ~]# echo '<h1>wo shi webroot' > /webroot/index.html

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

<VirtualHost \*:80>

ServerName server0.example.com

DocumentRoot /webroot #修改网页文件存放目录

</VirtualHost>

[root@server0 ~]# vim /etc/httpd/conf.d/nsd02.conf

<Directory /webroot> #针对于/webroot设置权限

Require all granted #允许所有人访问

</Directory>

[root@server0 ~]# systemctl restart httpd.service

1. .修改SELinux的安全上下文值(标签)

[root@server0 ~]# ls -Zd /var/www #查看目录安全上下文值

[root@server0 ~]# ls -Zd /webroot #查看目录安全上下文值

方式1:参照标准目录,重设新目录的属性

-chcon [-R] --reference=模板目录 新目录

[root@server0 ~]# chcon -R --reference=/var/www /webroot

### 客户机地址限制

## Selinux策略保护

### 标准web目录

### 增加新的web目录

# 部署动态网站

## 动态网站概述

### 静态网站的运行

服务端的原始网页 = 浏览器访问到的网页

-由Web访问软件处理所有请求

-文本(txt/html)、图片(jpg/png)等静态资源

### 动态网站的运行

服务端的原始网页 ≠ 浏览器访问到的网页

-由Web服务软件接受请求,动态程序转后端模块处理

-PHP网页、Python网页、JSP网页......

## 部署WSGI应用

### 快速构建WSGI动态网站

新建一个动态站点(虚拟主机)

-站点名称为webapp0.example.com

1. 下载Python网页的页面文件,下载到webapp0.example.com的DocumentRoot

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

[root@server0 ~]# cd /var/www/baidu/

[root@server0 baidu]# wget

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

1. 方便客户端用户访问动态页面

页面跳转:当客户端访问网页时,进行跳转到另一个页面

当用户在浏览器输入webapp0.example.com,

进行跳转到/var/www/baidu/webinfo.wsgi

Alias 客户端浏览器的网络路径 服务端本机实际路径

Alias /(网页文件根目录) /var/www/baidu/webinfo.wsgi

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

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/baidu

Alias / /var/www/baidu/webinfo.wsgi

</VirtualHost>

[root@server0 baidu]# systemctl restart httpd

[root@server0 baidu]# firefox webapp0.example.com

1. 安装mod\_wsgi软件进行翻译Python页面代码

[root@server0 /]# yum -y install mod\_wsgi

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

<VirtualHost \*:80>

ServerName webapp0.example.com

DocumentRoot /var/www/baidu

WsgiScriptAlias / /var/www/baidu/webinfo.wsgi

</VirtualHost>

[root@server0 /]# systemctl restart httpd

[root@server0 /]# firefox webapp0.example.com

页面内容:

UNIX时间戳:自1970-1-1 0:0:0到现在时间经过的秒数

1. 此虚拟主机webapp0.example.com 使用的端口8909

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

Listen 8909 #设置httpd需要监听8909

<VirtualHost \*:8909> #此虚拟机web主机在8909提供服务

ServerName webapp0.example.com

DocumentRoot /var/www/baidu

WsgiScriptAlias / /var/www/baidu/webinfo.wsgi

</VirtualHost>

### 测试WSGI网页

## Selinux策略保护

### 标配web端口

### 允许开放其他web端口

1. SELinux非默认端口的开放

[root@server0 /]# semanage port -l | grep http

[root@server0 /]# semanage port -a -t http\_port\_t -p tcp 8909

-a:添加 -t:类型 -p:协议

[root@server0 /]# systemctl restart httpd

[root@server0 /]# firefox webapp0.example.com:8909

默认情况下,端口的优先级最高

写在第一个虚拟web主机优先级最高

# 总结和答疑

## Httpd启动失败

### 问题现象

### 故障分析及排除

## Selinux策略故障

### 问题现象

### 故障分析及排除

# 综合串讲

## 安全web服务

https:安全的超文本传输协议 443

### PKI公钥基础设施

-公钥:主要用来加密数据

-私钥:主要用来解密数据(与相应的公钥匹配)

-数字证书:证明拥有者的合法性/权威性(单位名称、有效期、公钥、颁发机构及签名、......)

-Certificate Authority,数字证书授权中心:负责证书的申请/审核/颁发/鉴定/撤销等管理工作

### 实现HTTPS加密的条件

### 部署证书、密钥相关文件

虚拟机Server0:构建安全的Web信息

1.部署网站证书(营业执照)

[root@server0 ~]# cd /etc/pki/tls/certs/

[root@server0 certs]# wget

http://classroom.example.com/pub/tls/certs/server0.crt

1. 部署根证书(授权中心的信息)

[root@server0 certs]# wget <http://classroom.example.com/pub/example-ca.crt>

1. 部署私钥(用于解密)

[root@server0 certs]# cd /etc/pki/tls/private/

[root@server0 private]# wget

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

1. 安装mod\_ssl软件包,支持加密通信

[root@server0 private]# yum -y install mod\_ssl

[root@server0 private]# rpm -ql mod\_ssl

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

1. 修改配置文件/etc/httpd/conf.d/ssl.conf

补充:vim 末行模式 输入:set nu #开启行号功能

[root@server0 private]# vim /etc/httpd/conf.d/ssl.conf

59 DocumentRoot "/var/www/html" #指定存放网页路径

60 ServerName [www.example.com:443](http://www.example.com:443) #指定网站的域名

指定网站证书

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

指定私钥

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

指定根证书(授权中心的信息)

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

1. 重启httpd服务

[root@server0 private]# systemctl restart httpd

1. 验证:

[root@server0 private]# firefox https://www0.example.com

点击---->”我已了解可能的风险”---->”确认安全例外”

### 调整web服务配置

### 访问HTTPS加密的站点

## 基础邮件服务

### 电子邮件通信

邮件服务器的构建

电子邮件服务器的基本功能

-为用户提供电子邮箱存储空间(用户名@邮件域名)

-处理用户发出的邮件--传递给收件服务器

-处理用户收到的邮件--传递到邮箱

SMTP:用户发出的邮件协议 默认端口25

pop3:用户收到的邮件协议 默认端口110

DNS服务器:虚拟机classroom

server0.example.com

邮箱账号:lisi@server0.example.com

### 快速部署postfix邮件服务器

虚拟机server0:构建邮件服务

1. 安装软件包postfix

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

[root@server0 /]# rpm -q postfix

2.修改配置文件

[root@server0 /]# vim /etc/postfix/main.cf

99 myorigin = server0.example.com #默认补全域名后缀

114 inet\_interfaces = all #本机所有网卡都提供邮件服务

162 mydestination = server0.example.com #此字段的值,判断为本域邮件

1. 重启postfix服务

[root@server0 /]# systemctl restart postfix.service

1. 测试收发邮件

[root@server0 /]# useradd

### 使用mail命令发信/收信

mail发信操作

-mail -s ‘邮件标题’ -r 发件人 收件人

-echo ‘邮件正文’ | mail -s ‘邮件标题’ -r 发件人 收件人

mail 收信操作

-mail [-u 用户名]

[root@server /]# mail -s ‘test01’ -r yg xln

hahaxixihehelele

. #只有一个点代表提交

EOT

[root@server0 /]# mail -u xln

& 1 #输入编号1查看邮件

## 准备交换分区

### Parted分区工具

MBR:最多有4个主分区,最大支持的磁盘2TB

GPT:最多由128个主分区,最大支持的空间18EB

[root@server0 /]# parted /dev/vdb

(parted) mktable gpt #指定分区模式为GPT

(parted) print #输出分区表信息

(parted) mkpart #划分新的分区

分区名称？ []? haha #分区的名称随意起

文件系统类型？ [ext2]? ext4

起始点？ 0

结束点？ 2G

警告: The resulting partition is not properly aligned for best performance.

忽略/Ignore/放弃/Cancel? Ignore #选择忽略,留给分区表一些空间

(parted) unit gb #使用GB作为显示单位

(parted) print

(parted) mkpart #划分新的分区

分区名称？ []? haha

文件系统类型？ [ext2]?

起始点？ 2G

结束点？ 4G

(parted) print

### 什么是交换空间

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

### 如何建立一个交换分区

1. 查看交换空间的信息

[root@server0 /]# swapon -s

1. 格式化交换文件系统

[root@server0 /]# mkswap /dev/vdb1

[root@server0 /]# mkswap /dev/vdb2

[root@server0 /]# blkid /dev/vdb1

[root@server0 /]# blkid /dev/vdb2

### 交换空间的启用控制

1. 启用交换分区

[root@server0 /]# swapon /dev/vdb1

[root@server0 /]# swapon -s

[root@server0 /]# swapon /dev/vdb2

[root@server0 /]# swapon -s

[root@server0 /]# swapoff /dev/vdb2 #停用交换分区

[root@server0 /]# swapon -s

1. 开机自动启用交换分区

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

/dev/vdb1 swap swap defaults 0 0

/dev/vdb2 swap swap defaults 0 0

[root@server0 /]# swapon -a

[root@server0 /]# swapon -s

# 综合练习

## Linux工程师测试

### 环境准备及说明

# 扩展的几个应用

## 补充技巧

### 目录结构

目录结构

-man hier

常见一级目录的用途

/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

### 权限的数值表示

权限的数值化

-基本权限:r=4,w=2,x=1

-附件权限:SUID=4,SGID=2,Sticky Bit=1

采用数值形式设置权限

-chmod [-R] nnn文档...

-chmod [-R] xnnn文档...

[root@svr7 ~]# mkdir /nsd01

[root@svr7 ~]# ls -ld /nsd01

drwxr-xr-x. 2 root root 6 1月 16 14:16 /nsd01

[root@svr7 ~]# chmod 700 /nsd01

[root@svr7 ~]# ls -ld /nsd01

drwx------. 2 root root 6 1月 16 14:16 /nsd01

[root@svr7 ~]# chmod 007 /nsd01

[root@svr7 ~]# ls -ld /nsd01

d------rwx. 2 root root 6 1月 16 14:16 /nsd01

[root@svr7 ~]# chmod 750 /nsd01

[root@svr7 ~]# ls -ld /nsd01

drwxr-x---. 2 root root 6 1月 16 14:16 /nsd01

[root@svr7 ~]# chmod 3755 /nsd01

[root@svr7 ~]# ls -ld /nsd01

drwxr-sr-t. 2 root root 6 1月 16 14:16 /nsd01

### 历史命令

管理/调用曾经执行过的命令

- history:查看历史命令列表

- history -c:清空历史命令

- !n:执行命令历史中的第n条命令

- !str:执行最近一次以str开头的历史命令

调整历史命令的数量

[root@svr7 ~]# vim /etc/profile

HISTSIZE=1000 #默认记录1000条

### 实用命令小工具

du,统计文件的占用空间

-du [选项]... [目录或文件]...

-s:只统计每个参数所占用的总空间大小

-h:提供易读容量单位(K、M等)

[root@svr7 ~]# du -sh /etc /boot /root

[root@svr7 ~]# du -sh /

date,查看/调整系统日期时间

-date +%F、date+%R

-date +”%Y-%m-%d %H:%M:%S”

-date -s “yyyy-mm-dd HH:MM:SS”

[root@svr7 ~]# date

[root@svr7 ~]# date +%F #显示年月日

[root@svr7 ~]# date +%R #显示时与分

[root@svr7 ~]# date +%Y #显示年

[root@svr7 ~]# date +%m #显示月

[root@svr7 ~]# date +%d #显示日

制作快捷方式:

[root@svr7 ~]# cat /etc/redhat-release

Red Hat Enterprise Linux Server release 7.4 (Maipo)

[root@svr7 ~]# ln -s /etc/redhat-release /mnt

[root@svr7 ~]# ls /mnt

redhat-release

[root@svr7 ~]# ln -s /etc/redhat-release /mnt/abc

[root@svr7 ~]# ls /mnt

abc redhat-release

[root@svr7 ~]# ls -l /mnt/abc

lrwxrwxrwx. 1 root root 19 1月 16 15:07 /mnt/abc -> /etc/redhat-release

[root@svr7 ~]# cat /mnt/abc

ln -s,创建软连接

软连接若原始文件或目录被删除,连接文件将失效

软连接可存放在不同分区/文件系统(跨分区)

软连接可以是目录

ln,创建硬连接

制作硬连接(硬链接)

硬链接 若原始文件被删除,连接文件仍可用

硬链接与原始文件必须在同一分区/文件系统

## 获取命令帮助

### 查看命令的简要说明

方式1:命令 --help

方式2:man 命令

[root@svr7 ~]# man 5 passwd #5配置文件的帮助信息

### 实用man手册

## zip归档工具(跨平台的压缩工具)

### 制作.zip压缩包

归档+压缩操作

-zip [-r] 备份文件.zip 被归档的文档...

释放归档+解压操作

-unzip 备份文件.zip [-d 释放目标文件夹]

### 释放.zip压缩包

# 发布网络YUM源

## 准备YUM源

### yum仓库特点

搭建新的教学环境

一 建立两台虚拟机

真机:快速新建虚拟机命令(脚本)

[root@roomp8p1 ~]# clone-vm7

Enter VM number: 5

Creating Virtual Machine disk image...... [OK]

Defining new virtual machine...... [OK]

[root@roomp8p1 ~]# clone-vm7

Enter VM number: 6

Creating Virtual Machine disk image...... [OK]

Defining new virtual machine...... [OK]

二 配置两台虚拟机 利用root用户 密码为123456

虚拟机A

配置永久的主机名:svr7.tedu.cn

配置eth0网卡IP地址:192.168.4.7/24

虚拟机B

配置永久的主机名:pc207.tedu.cn

配置eth0网卡IP地址:192.168.4.7/24

三 真机远程管理虚拟机

1. 真机可以与虚拟机通信
2. 真机进行远程管理
3. 为真机设置别名

alias goa='ssh -X root@192.168.4.7'

alias gob='ssh -X root@192.168.4.207'

### 实用RHEL7光盘库

四 构建Yum仓库

1. 真机构建FTP服务,共享光盘所有内容

[root@roomp8p1 ~]# rpm -q vsftpd

vsftpd-3.0.2-22.el7.x86\_64

[root@roomp8p1 ~]# systemctl status vsftpd #查看服务状态

netstat -antpu 查看监听端口

FTP默认共享的路径:/var/ftp

验证:

[root@roomp8p1 ~]# ls /var/ftp/rhel7/

/etc/fstab不支持快捷方式,所有的路径都必须是确切的路径

[root@roomp8p1 ~]# ls /

[root@roomp8p1 ~]# ls -l /ISO

[root@roomp8p1 ~]# vim /etc/fstab

/ISO/rhel-server-7.4-x86\_64-dvd.iso /var/ftp/rhel7 iso9660 loop,ro 0 0

1. 客户端:虚拟机A与虚拟机B

[root@svr7 ~]# vim /etc/yum.repos.d/rhel7.repo

[rhel7]

name=rhel7

baseurl=ftp://192.168.4.254/rhel7

enabled=1

gpgcheck=0

[root@pc207 ~]# vim /etc/yum.repos.d/rhel7.repo

[rhel7]

name=rhel7

baseurl=ftp://192.168.4.254/rhel7

enabled=1

gpgcheck=0

### 实用第三方RPM包建库

自定义Yum仓库

1. 具备互联网下载的软件包
2. 将tools.tar.gz传输到虚拟机A的/root目录下

scp命令:依赖于ssh

真机上传递:

[root@roomp8p1 ~]# scp /root/桌面/tools.tar.gz [root@192.168.4.7:/root/](mailto:root@192.168.4.7:/root/)

3.进行解包

[root@svr7 ~]# tar -xf /root/tools.tar.gz -C /

[root@svr7 ~]# ls /

bin dev home lib64 mnt opt root sbin sys tools var

boot etc lib media nsd01 proc run srv tmp usr

[root@svr7 ~]# ls /tools/

inotify-tools-3.13.tar.gz other

[root@svr7 ~]# ls /tools/other/

1. 创建仓库数据文件

[root@svr7 ~]# createrepo /tools/other/

[root@svr7 ~]# ls /tools/other/

1. 修改Yum客户端配置文件

[root@svr7 ~]# vim /etc/yum.repos.d/rhel7.repo

[other]

name=my rpm

baseurl=file:///tools/other #指定本地为Yum仓库服务端

enabled=1

gpgcheck=0

[root@svr7 ~]# yum repolist

## 源的HTTP发布

### 快速构建HTTP服务器

### 部署HTTP资源

### 测试HTTP方式的yum源

# vim编辑技巧

[root@svr7 ~]# cp /etc/passwd /opt/1.txt

[root@svr7 ~]# vim /opt/1.txt

## 命令模式操作

### 光标跳转

操作类型 按键指令 用途

光标行内跳转 Home键或^、数字0 跳转到行首

End键或”$”键 跳转到行尾

全文翻页 PgUp键、PgDn键 向上翻页、向下翻页

光标行间跳转 1G或gg 跳转到文件的首行

G 跳转到文件的末尾行

### 复制/粘贴/删除

操作类型 按键指令 用途

复制 yy、#yy 复制光标处的一行、#行

粘贴 p、P 粘贴到光标处之后、之前

删除 x或Delete键 删除光标处的单个字符

dd、#dd 删除光标处的一行、#行

d^ 从光标处删除至行首

d$ 从光标处删除到行尾

### 查找/撤销/保存

操作类型 按键指令 用途

文本查找 /word 向后查找字符串”word”

n、N 跳至后/前一个结果

撤销编辑 u 撤销最近的一次操作

U 撤销对当前行的所有修改

Ctrl+r 取消前一次撤销操作

保存退出 ZZ 保存修改并退出

## 末行模式操作

### 保存/退出/文件操作

操作类型 设置指令 用途

存盘及退出 :w 保存当前文件

:q! 放弃已有更改后强制退出

:wq或:x 保存已有修改后退出

文件操作 :w /root/newfile 另存为其它文件

:r /etc/filesystems 读入其他文件内容

### 字符串替换

操作类型 设置指令 用途

行内替换 :s/old/new 替换当前行第一个”old”

:s/old/new/g 替换当前行所有的”old”

区域内替换 :n,m s/old/new/g 替换第n-m行所有的”old”

:% s/old/new/g 替换文件内所有的”old”

### 开关参数的控制

操作类型 设置指令 用途

编辑器设置 :set nu或nonu 显示/不显示行号

:set ai或noai 启用/关闭自动缩进

# 源码编译安装

## 编译安装介绍

### 源码编译安装的优势

inotify-tools:源码包

rpm包: rpm -ivh yum

源码包-----make gcc----->可以执行的程序--------->运行安装

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

-源码包适用于各种平台

步骤1:安装gcc与make开发工具

步骤2:tar解包,释放源代码至指定目录

步骤3:./configure 配置,指定安装目录/功能模块等选项

步骤4:make编译,生成可执行的二进制程序文件

步骤5:make install 安装,将编译好的文件复制到安装目录

### 准备编译环境

### 基本实现过程

## 分布解析及示例

### 下载及解包

### 配置

### 编译及安装

### 结果验证

# systemctl控制

## systemd介绍

### init程序的作用

Linux系统和服务管理器

-是内核引导之后加载的第一个初始化进程(PID=1)

-负责掌握整个Linux的运行/服务资源组合

### systemd

一个更高效的系统&服务管理器

-开机服务并行启动,各系统服务间的精确依赖

-配置目录:/etc/systemd/system/

-服务目录:/lib/systemd/system/

– 主要管理工具:systemctl

对于服务的管理

systemctl restart 服务名 #重起服务

systemctl start 服务名 #开启服务

systemctl stop 服务名 #停止服务

systemctl status 服务名 #查看服务当前的状态

systemctl enable 服务名 #设置服务开机自启动

systemctl disable 服务名 #设置服务不开机自启动

RHEL6 运行级别 200

0：关机

1：单用户模式（基本功能的实现，破解Linux密码）

2：多用户字符界面（不支持网络）

3：多用户字符界面（支持网络）服务器默认的运行级别

4：未定义

5：图形界面

6：重起

切换运行级别：init 5

RHEL7 运行模式

字符模式：multi-user.target

图形模式：graphical.target

[root@svr7 /]# ls -l /lib/systemd/system/

当前直接切换到图形模式

[root@svr7 /]# systemctl isolate graphical.target

当前直接切换到字符模式

[root@svr7 /]# systemctl isolate multi-user.target

查看每次开机默认进入模式

# systemctl get-default

multi-user.target

设置永久策略，每次开机自动进入graphical.target

# systemctl set-default graphical.target

# reboot

### unit配置单元

## 管理系统服务

### 列出服务

### 启动/停止/重启/看状态

### 配置开机自启

## 管理运行级别

### 切换级别

### 设置默认级别

# 总结和答疑

## vim编辑器异常

### 问题现象

### 故障分析及排除

## 编码编译安装

### 问题现象

### 故障分析及排除

### **systemctl控制** 问题现象

### 故障分析及排除

### 

# DNS服务基础

## DNS工作原理

### DNS解析的作用

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

### DNS的分布式结构

一级DNS服务器 二级DNS服务器 三级DNS服务器

### DNS域名管理

域名体系结构为:树型结构

所有的域名都必须以点作为结尾

根域名

一级域名:.cn .us .hk .tw .kr .jp ......

二级域名:.com.cn .org.cn .net.cn ......

三级域名:nb.com.cn sina.com.cn ......

完整的主机名(完全合格的域名):

Full Qualified Domian Name,完全合格主机名(FQDN)

-系统服务:named

-默认端口:TCP/UDP 53

主配置文件:/etc/named.conf #本机负责解析的域名 tedu.cn

地址库文件:/var/named/ #记录域名与IP地址对应关系

检查配置语法

-格式:named-checkzone 区域名 配置文件

## BIND服务分析

### BIND域名服务

运行时的虚拟根环境:/var/named/chroot

1. 安装软件包

[root@svr7 ~]# yum -y install bind bind-chroot

bind.x86\_64 32:9.9.4-50.el7 //域名服务包

bind-chroot.x86\_64 32:9.9.4-50.el7 //提供虚拟根支持

### named.conf配置文件

1. 修改主配置文件/etc/named.conf

[root@svr7 ~]# cp /etc/named.conf /etc/name.bak

[root@svr7 ~]# vim /etc/named.conf

options {

directory "/var/named"; #指定地址库文件

};

zone "tedu.cn" IN { #设置本机负责解析的域名

type master; #设置本机为主DNS服务器

file "tedu.cn.zone"; #设置地址库文件名称

};

### 地址库文件

1. 建立地址库文件,必须保证named用户有读取权限

-p:保持权限不变进行复制

[root@svr7 ~]# cd /var/named/

[root@svr7 named]# cp -p named.localhost tedu.cn.zone

[root@svr7 named]# ls -l tedu.cn.zone

所有的域名都要以.结尾

如果没有以.结尾,自动补全本地址库文件负责的域名

[root@svr7 named]# vim tedu.cn.zone

$TTL 1D

@ IN SOA @ rname.invalid. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

tedu.cn. NS svr7.tedu.cn. #指定域名解析服务器为svr7

svr7.tedu.cn. A 192.168.4.7 #解析服务器为svr7的IP地址

www.tedu.cn. A 192.168.4.110 #域名与IP对应记录

ftp.tedu.cn A 192.168.4.120 #域名与IP对应记录

[root@svr7 named]# vim tedu.cn.zone

tedu.cn. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.110

ftp A 192.168.4.120

[root@svr7 named]# systemctl restart named

虚拟机B:进行验证

1. 指定DNS服务器位置

[root@pc207 ~]# echo nameserver 192.168.4.7 >/etc/resolv.conf

[root@pc207 ~]# vim /etc/resolv.conf

1. 测试解析

[root@pc207 ~]# nslookup www.tedu.cn

[root@pc207 ~]# nslookup ftp.tedu.cn

实现多区域的DNS服务器

1. 修改主配置文件/etc/named.conf

[root@svr7 named]# vim /etc/named.conf

zone ”qq.com” IN {

type master;

file “qq.com.zone”;

};

1. 建立地址库文件qq.com.zone

[root@svr7 named]# cd /var/named

[root@svr7 named]# cp -p tedu.cn.zone qq.com.zone

[root@svr7 named]# vim qq.com.zone

qq.com. NS svr7

svr7 A 192.168.4.7

www A 1.2.3.4

ftp A 10.20.30.40

3.重启named服务

[root@svr7 named]# systemctl restart named

## 单区域DNS服务

### 快速搭建DNS服务器

案例:搭建DNS服务器 实现解析www.sina.con结果为192.168.1.1

搭建DNS服务器负责解析sina.com

地址库文件解析记录为 www A 192.168.1.1

### DNS查询测试

# 特殊解析

特殊的解析记录(地址库文件的解析记录)

## 地址记录应用

### DNS轮询

基于NDS的站点负载均衡

-一个域名 ---> 多个不同IP地址

### 泛域名解析

虚拟机A:修改地址库文件

[root@svr7 named]# vim /var/named/tedu.cn.zone

\* A 40.50.60.70

[root@svr7 named]# systemctl restart named

虚拟机B:测试

[root@pc207 ~]# nslookup wwww.tedu.cn

[root@pc207 ~]# nslookup hahaxixi.tedu.cn

有规律的泛域名解析

pc1.tedu.cn ----> 192.168.10.1

pc2.tedu.cn ----> 192.168.10.2

pc3.tedu.cn ----> 192.168.10.3

pc4.tedu.cn ----> 192.168.10.4

......

pc50.tedu.cn ----> 192.168.10.50

内置函数:$GENERATE 制造连续的数字

$GENERATE 起始点-结束点

虚拟机:A修改地址库文件

[root@svr7 named]# vim /var/named/tedu.cn.zone

$GENERATE 1-50 pc$ A 192.168.10.$

[root@svr7 named]# systemctl restart named

虚拟机B:测试

[root@pc207 ~]# nslookup pc3.tedu.cn

[root@pc207 ~]# nslookup pc12.tedu.cn

[root@pc207 ~]# nslookup pc37.tedu.cn

补充:解析记录的别名:

[root@svr7 named]# vim /var/named/tedu.cn.zone

tts CNAME ftp

tts解析结果与ftp的解析结果是一致的

[root@svr7 named]# systemctl restart named

虚拟机B:测试

[root@pc207 ~]# nslookup tts.tedu.cn

# DNS子域授权

## 配置子域授权

### 子域授权的作用

父域:www.qq.com

子域:www.bj.qq.com

虚拟机A构建DNS服务器,负责解析qq.com

虚拟机B构建DNS服务器,负责解析bj.qq.com

1.安装软件包

[root@pc207 ~]# yum -y install bind bind-chroot

2.修改主配置文件/etc/named.conf

[root@svr7 ~]# cp /etc/named.conf /etc/name.bak

[root@svr7 ~]# vim /etc/named.conf

options {

directory "/var/named";

};

zone "bj.qq.com" IN {

type master;

file "bj.qq.com.zone";

};

3.

[root@svr7 ~]# cd /var/named/

[root@svr7 named]# cp -p named.localhost bj.qq.com.zone

[root@svr7 named]# ls -l bj.qq.com.zone

[root@svr7 named]# vim bj.qq.com.zone

bj.qq.com. NS pc207

pc207 A 192.168.4.207

www A 10.20.30.40

[root@svr7 named]# systemctl restart named

### 案例环境及要点

### 为父DNS启用子域授权

实现子域授权

虚拟机A构建 DNS服务器,能够解析子域的bj.qq.com

在虚拟机A上操作:

1. 修改地址库文件

[root@svr7 named]# vim /var/named/qq.com.zone

qq.com. NS svr7

bj.qq.com. NS pc207

svr7 A 192.168.4.7

pc207 A 192.168.4.207

www A 1.2.3.4

ftp A 10.20.30.40

[root@svr7 named]# systemctl restart named

[root@pc207 named]# nslookup www.bj.qq.com 192.168.4.7

Server: 192.168.4.7

Address: 192.168.4.7#53

Non-authoritative answer: #非权威解答

Name: www.bj.qq.com

Address: 10.20.30.40

### 测试子域FQDN查询

## 递归/迭代查询

### 递归/迭代的作用

递归解析:

客户端发送解析请求给DNS服务器,DNS服务器与其他DNS服务器交流,最终将解析结果带回来的过程.

迭代查询:DNS服务器与其他DNS服务器交流的过程.

对于一台DNS服务器来说：

若允许递归，则当客户端请求解析的域名非本DNS管辖时，本DNS会向其他DNS服务器代询；

若不允许递归，则当客户端请求解析的域名非本DNS管辖时，本DNS会放弃代询 —— 但是，如果目标地址位于已知的某个授权子域，本DNS会告知客户端对应的子DNS服务器的地址信息（即迭代）。

### DNS查询的工作方式

客户端解析域名时优先级

1. /etc/hosts
2. /etc/resolv.conf

/etc/hosts(主机映射记录文件,可以直接起到域名解析作用)

只为本机提供解析

[root@pc207 /]# vim /etc/hosts

192.168.4.130 www.360.com

[root@pc207 /]# ping www.360.com

### 如何开启/禁用递归

### 验证迭代查询

# 缓存DNS

## 缓存DNS概述

### 互联网DNS角色

作用:缓存解析记录,加速解析

### 缓存DNS的适用场景

真机上实现缓存DNS服务器

1. 真机搭建Yum仓库

[root@roomp8p1 ~]# cat /etc/redhat-release

[root@roomp8p1 ~]# mkdir /dvd

[root@roomp8p1 ~]# mount /ISO/CentOS-7-x86\_64-

[root@roomp8p1 ~]# mount /ISO/CentOS-7-x86\_64-DVD-1708.iso /dvd

[root@roomp8p1 ~]# cd /etc/yum.repos.d/

[root@roomp8p1 yum.repos.d]# mkdir repo

[root@roomp8p1 yum.repos.d]# mv \*.repo repo

[root@roomp8p1 yum.repos.d]# vim dvd.repo

[dvd]

name=CentOS7.4

baseurl=file:///dvd

enabled=1

gpgcheck=0

1. 安装软件包bind bind-chroot

[root@roomp8p1 /]# yum -y install bind bind-chroot

1. 搭建缓存DNS服务器

[root@roomp8p1 /]# cat /etc/resolv.conf #查看DNS服务器地址

[root@roomp8p1 /]# cp /etc/named.conf /etc/named.bak

[root@roomp8p1 /]# vim /etc/named.conf

options {

directory "/var/named";

forwarders { 176.19.0.26; }; #转发给真正的DNS服务器

};

[root@roomp8p1 /]# systemctl restart named

1. 虚拟机测试

[root@svr7 named]# nslookup www.qq.com 192.168.4.254

### 解析记录来源

NS A CNAME

NS记录为域名服务器记录（本域权威DNS的FQDN）

A记录为正向解析记录（FQDN --> IP地址）

配置缓存DNS服务器时，为客户机提供的解析记录如何获取

方式1，全局转发：将请求转发给指定的公共DNS，请求递归服务；

方式2，根域迭代：依次向根、一级、二级……域的DNS服务器迭代。

## 全局转发式缓存

### 配置要点

### 测试查询效果

# 总结和答疑

## NS资源记录

### 问题现象

启动named服务时失败

-报错:NS ‘.. ..’has no address records (A or .. ..)

-语法检查报同样的错误

### 故障分析及排除

原因分析

-NS记录用来标记本区域DNS服务器的地址(接收地址库更新通知)

-如果该地址正好隶属本区域,则需要增加相应A记录

解决办法

-指定NS记录时,同时设置配套的A记录即可

## 地址库文件权限

### 问题现象

启动named服务成功,但host查询没有结果

-报错1:connection timed out; no servers .. ..

-报错2:Host .. .. not found:2(SERVFAIL).. ..

### 故障分析及排除

原因分析

-问题1:客户机未正确指定DNS,无法查询

-问题2:DNS地址库中不包括要查询域名的A记录,或者named服务对地址库文件没有r权限

解决办法

-为客户机正确设置/etc/resolv.conf文件

-确认存在该站点记录,并调整地址库文件权限 644

### 

# Split分离解析

## 分离解析概述

### 什么是分离解析

当收到客户机的DNS查询请求的时候

-能够区分客户机的来源地址

-为不同类别的客户机提供不同的解析结果(IP地址)

------>网通用户------>网通服务器

[www.qq.com](http://www.qq.com)

------>电信用户------>电信服务器

### 典型适用场景

访问压力大的网站,购买CDN提供的内容分发服务

-在全国各地/不同网络内部署大量镜像服务节点

-针对不同的客户端,提供最近的资源服务器

### BIND的view视图

根据源地址集合将客户机分类

- 不同客户机获得不同结果(待遇有差别)

- 分离解析,由上到下,依次匹配,匹配及停止

- 分类要合理,每一个客户端都要找到自己的分类

- 所有的zone,都要在view中的大括号中

view “a” {

match-clients { 192.168.4.1; .. ..; }; #客户端为192.168.4.1

zone “12306.cn” IN {

...... 12306.cn.zone;

};};

view “b” {

match-clients { 192.168.4.2; .. ..; }; #客户端为192.168.4.2

zone “12306.cn” IN {

...... 12306.cn.other;

};};

view “c” {

match-clients { any; .. ..; }; #客户端为任何人

zone “12306.cn” IN {

...... 12306.cn.other;

};};

案例需求及要求

环境及需求

- 权威DNS:svr7.tedu.cn 192.168.4.7

- 负责区域:tedu.cn

- A记录分离解析 ------以www.tedu.cn

客户机来自解析结果

192.168.4.207、192.168.7.0/24 ----->192.168.4.100

其他地址 ---->1.2.3.4

多区域的分离解析:

每一个view

虚拟机A:构建DNS分离解析

[root@svr7 ~]# vim /etc/named.conf

options {

directory "/var/named";

};

view "a" {

match-clients { 192.168.4.207; };

zone "tedu.cn" IN {

type master;

file "tedu.cn.zone";

};

};

view "b" {

match-clients { any; };

zone "tedu.cn" IN {

type master;

file "tedu.cn.other";

};

};

[root@svr7 ~]# vim /var/named/tedu.cn.other

tedu.cn. NS svr7

svr7 A 192.168.4.7

www A 1.2.3.4

[root@svr7 ~]# vim /var/named/tedu.cn.zone

tedu.cn. NS svr7

svr7 A 192.168.4.7

www A 192.168.4.100

### acl地址列表

补充了解内容:

acl地址列表

为大批量的客户机地址建立列表

-调用时指定列表名即可,列表名any可匹配任意地址

-根据view调用的顺序,”匹配即停止”

acl test { 192.168.4.207; 192.168.4.1; 192.168.1.1; 192.168.7.0/24; };

## 配置分离解析

### 案例需求及要点

### 配置Split分离解析

### 测试分离解析

# RAID磁盘阵列

## RAID磁盘阵列

### RAID阵列概述

廉价冗余磁盘阵列

-Redundant Arrays of Inexpensive Disks

-通过硬件/软件技术,将多个较小/低速的磁盘整合成一个大磁盘

-阵列的价值:提升I/O效率、硬件级别的数据冗余

-不同RAID级别的功能、特性各不相同

### RAID0/1/10

RAID 0,条带模式

-同一个文档分散存放在不同磁盘

-并行写入以提高效率

-至少由两块磁盘组成

RAID 1,镜像模式

-一个文档复制成多份,分别写入不同磁盘

-多份拷贝提高可靠性,效率无提升

-至少由两块磁盘组成

RAID 0+1/RAID 1+0

-整合RAID 0、RAID 1的优势

-并行存取提高效率、镜像写入提高可靠性

-至少由四块磁盘组成

### RAID5/6

RAID 5,高性价比模式

-相当于RAID0和RAID1的折中方案

-需要至少一块磁盘的容量来存放校验数据

-至少由三块磁盘组成

RAID6,高性价比/可靠模式

-相当于扩展的RAID5阵列,提供2份独立校验方案

-需要至少两块磁盘的容量来存放校验数据

-至少由四块磁盘组成

### RAID各级别特点对比

### 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

# 进程管理

## 查看进程

### 查看进程树

进程管理

程序:静态的代码 占用磁盘空间

进程:动态执行的代码 占用CPU与内存

表示:编号 PID

父进程与子进程 树型结构

systemed:上帝 进程(所有进程的父进程)

查看进程

pstree-Processes Tree

-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

[root@svr7 ~]# id lisi

uid=1000(lisi) gid=1000(lisi) 组=1000(lisi)

[root@svr7 ~]# pstree lisi #查看lisi用户开启的进程

bash───vim

[root@svr7 ~]# pstree -a lisi #-a:显示完整命令行

bash

└─vim a.txt

[root@svr7 ~]# pstree -ap lisi #-p:显示进程的PID编号

bash,2929

└─vim,2972 a.txt

### 查看进程快照

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、包括最完整的进程信息(-f)

ps aux操作

-列出正在运行的所有进程

ps -elf操作

-列出正在运行的所有进程,常用于查询进程的父进程

PPID:父进程PID

### 进程动态排名

top交互式工具

-格式:top [-d 刷新秒数] [-U 用户名]

排序:按大写的P,进行CPU使用的排序

按大写的M,进行内存使用的排序

### 检索进程

pgrep-Process Grep

-用途:pgrep [选项]... 查询条件

常用命令选项

-l:输出进程名,而不仅仅是PID

-U:检索指定用户的进程

-t:检索指定终端的进程

-x:精确匹配完整的进程名

[root@svr7 ~]# pgrep -l log

[root@svr7 ~]# pgrep a

[root@svr7 ~]# pgrep -l a

[root@svr7 ~]# pgrep -l cron

[root@svr7 ~]# pgrep -lU lisi

pts:图形命令行终端

## 控制进程

### 进程的前后台调度

后台启动

-在命令行末尾添加”&”符号,不占用当前终端

Ctrl+z组合键

-挂起当前进程(暂停并转入后台)

jobs命令

-查看后台任务列表

fg命令

-将后台任务恢复到前台运行

bg命令

-激活后台被挂起的任务

[root@svr7 ~]# sleep 900& #正在运行放入后台

[root@svr7 ~]# jobs #查看后台进程信息

[root@svr7 ~]# sleep 800

^Z #按Ctrl+z暂停放入后台

[2]+ 已停止 sleep 800

[root@svr7 ~]# jobs

[root@svr7 ~]# bg 2 #将后台编号为2的进程继续运行

[root@svr7 ~]# jobs

[root@svr7 ~]# fg 1 #将后台编号为1的进程,恢复到前台

sleep 900

^C #按Ctrl+c结束该进程

[root@svr7 ~]# jobs

[root@svr7 ~]# fg 2

sleep 900

^C #按Ctrl+c结束该进程

[root@svr7 ~]# jobs

### 杀死进程

干掉进程的不同方法

-Ctrl+c组合键,中断当前命令程序

-kill [-9] PID ...、kill [-9] %后台任务编号

kill 进程号 默认选项是-15

kill -l 查看kill的多种选项,可以跟数字也可以跟字母.

-killall [-9] 进程名...

-pkill 查找条件

[root@svr7 ~]# sleep 800&

[root@svr7 ~]# sleep 900&

[root@svr7 ~]# sleep 1000&

[root@svr7 ~]# jobs -l

[root@svr7 ~]# kill 4155

[root@svr7 ~]# jobs -l

[root@svr7 ~]# killall sleep

[root@svr7 ~]# jobs -l

pkill -9 -U hackli #强制踢出用户

killall -9 -u hackli #强制踢出用户

# 日志管理

## 日志概述

### 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”

### 内核及系统日志

由系统服务rsyslog统一记录/管理

### 用户日志

常见的日志文件

/var/log/messages 记录内核消息、各种服务的公共消息

/var/log/dmesg 记录系统启动过程的各种消息

/var/log/cron 记录与cron计划任务相关的消息

/var/log/maillog 记录邮件收发相关的消息

/var/log/secure 记录与访问限制相关的安全消息

## 日志分析

### 查看文本日志消息

由登录程序负责记录/管理

-日志消息采用二进制格式

-记录登录用户的事件、来源、执行的命令等信息

tailf:实时跟踪日志

Jan 30 17:25:55 localhost dhclient[1304]: All rights reserved.

时间标签 主机名 服务[端口] :消息内容

### 用户登录分析

users、who、w命令

-查看已登录的用户信息,详细度不同

last、lastb命令

-查看最近登录成功/失败的用户信息

[root@svr7 ~]# last -2 #最近两条登录成功用户信息

[root@svr7 ~]# lastb -2 #最近两条登录失败的用户信息

### 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

常见的Linux开机设置文件

/etc/fstab:实现开机自动挂载设备的配置文件

/etc/initab:定义开机进入默认级别的配置文件

/etc/rc.local:定义开机自定义任务的配置文件

GPL,GNU,自由软件

GPL:(通用公共许可证):一种授权,任何人有权取得、修改、重新发布自由软件的权力。GNU:(革奴计划):目标是创建一套完全自由、开放的的操作系统。

自由软件:是一种可以不受限制地自由使用、复制、研究、修改和分发的软件。主要许可证有 GPL 和 BSD 许可证两种。

### 使用journalctl工具

提取由systemd-journal 服务搜集的日志

-主要包括内核/系统日志、服务日志

常见用法/etc/httpd/conf.d/ssl.conf

-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”

# 总结和答疑

## 进程管理

### 问题现象

### 故障分析及排除

### 

# 批量装机环境

## 部署DHCP服务器

### DHCP概述及原理

Dynamic Host Configuration Protocol

-动态主机配置协议,由IETF(Internet网络工程师任务小组)组织制定,用来简化主机地址分配管理

主要分配以下入网参数

-IP地址/子网掩码/广播地址

-默认网关地址、DNS服务器地址

-PXE引导设置(TFTP服务器地址、引导文件名)

DHCP地址分配的四次会话(广播进行,先到先得)

-DISCOVERY --> OFFER --> REQUEST --> ACK

-一个网络中只能有一台DHCP服务器

默认端口号:67

### 配置dhcpd地址分配服务

虚拟机A:构建DHCP服务器

1. 安装软件包

[root@svr7 ~]# yum -y install dhcp

1. 修改配置文件

[root@svr7 ~]# vim /etc/dhcp/dhcpd.conf

:r /usr/share/doc/dhcp\*/dhcpd.conf.example

subnet 192.168.4.0 netmask 255.255.255.0 { #分配的网段

range 192.168.4.100 192.168.4.200; #分配的IP地址范围

option domain-name-servers 8.8.8.8; #分配的DNS服务器

option routers 192.168.4.254; #分配的网关地址

default-lease-time 600; #IP地址的租期时间

max-lease-time 7200;

}

[root@svr7 ~]# systemctl restart dhcpd

### dhclient测试

## 网络装机概述

### 网络装机的优势

规模化:同时装配多台主机

自动化:装系统、配置各种服务

远程实现:不需要光盘、U盘等物理安装介质

### 什么是PXE网络

PXE,Pre-boot eXecution Environment

-预启动执行环境,在操作系统之前运行

-可用于远程安装

工作模式

-PXE client 集成在网卡的启动芯片中

-当计算机引导时,从网卡芯片中把PXE client 调入内存执行,获取PXE server[配置、显示菜单,根据用户选择将远程引导程序下载到本机运行

### PXE组件及过程分析

## 基础条件

### 整体思路

### 提供软件安装源

# 配置PXE引导

## 部署TFTP服务

### 启用TFTP服务端

构建网络装机服务器

租期：允许客户机租用IP地址的时间期限，单位为秒

作用域：分配给客户机的IP地址所在的网段

地址池：用来动态分配的IP地址的范围

一 修改DHCP服务器配置,指定网络装机服务器配置

[root@svr7 ~]# vim /etc/dhcp/dhcpd.conf

......

next-server 192.168.4.7; #指定服务器下一个IP地址

filename "pxelinux.0"; #指定网卡引导文件名字

}

[root@svr7 ~]# systemctl restart dhcpd

pxelinux.0:网卡引导文件 (安装说明书) 二进制文件

安装软件可以自动生成该文件,默认叫pxelinux.0

二 搭建tftp服务,提供众多引导文件

tftp:简单的文件传输协议 默认端口为69

默认共享路径:/var/lib/tftpboot

1. 安装软件包

[root@svr7 ~]# yum -y install tftp-server.x86\_64

1. 重启服务

[root@svr7 ~]# systemctl restart tftp

[root@svr7 ~]# systemctl enable tftp

### 提供PXE启动程序

1. 部署pxelinux.0引导文件

[root@svr7 ~]# yum provides \*/pxelinux.0 #查询仓库中哪个软件产生此文件

[root@svr7 ~]# yum -y install syslinux

[root@svr7 ~]# rpm -ql syslinux | grep pxelinux.0 #查询软件包安装清单

[root@svr7 ~]# cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/

[root@svr7 ~]# ls /var/lib/tftpboot/ #tftp默认共享路径

### 提供装机内核及初始文件

## 配置启动菜单

### 拷贝模板文件

1. 读取菜单文件/var/lib/tftpboot/pxelinux.cfg/default

[root@svr7 ~]# mkdir /var/lib/tftpboot/pxelinux.cfg/

[root@svr7 ~]# mount /dev/cdrom /mnt

[root@svr7 ~]# cp /mnt/isolinux/isolinux.cfg

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

[root@svr7 ~]# ls -l /var/lib/tftpboot/pxelinux.cfg/default

[root@svr7 ~]# chmod u+w /var/lib/tftpboot/pxelinux.cfg/default

[root@svr7 ~]# ls -l /var/lib/tftpboot/pxelinux.cfg/default

1. 部署 图形的模块 与 背景图片

vesamenu.c32(图形的模块:显示的颜色,识别背景图片)

splash.png(背景图片)

[root@svr7 ~]# cp /mnt/isolinux/vesamenu.c32 /mnt/isolinux/splash.png /var/lib/tftpboot/

[root@svr7 ~]# ls /var/lib/tftpboot/

1. 部署 启动内核 与 驱动程序

vmlinuz(启动内核)

initrd.img(驱动程序)

[root@svr7 ~]# cp /mnt/isolinux/vmlinuz /mnt/isolinux/initrd.img /var/lib/tftpboot/

### default配置文件调整

1. 修改菜单文件内容

1 default vesamenu.c32 #默认加载图形模块

2 timeout 600 #默认读秒实现为60秒 1/10秒

10 menu background splash.png #加载背景图片splash.png

11 menu title NSD1812 PXE Server! #菜单标题

61 label linux

62 menu label Install RHEL7.4 #选项显示

menu default #默认选项

63 kernel vmlinuz #加载内核

64 append initrd=initrd.img #加载驱动程序

### 确认发布结果

总结:DHCP-----> IP地址、next-server、filename

tftp -----> pxelinux.0

pxelinux.0------>读取菜单文件 default

default----->vesamenu.c32、splash.png ......

## PXE装机测试

### 新建一台PXE虚拟机

新建一台全新的虚拟机

1.选择”网络引导安装(PXE)”

2.选择”网络类型:private1”

### PXE引导及验证

# kickstart自动应答

## kickstart概述

### 什么是kickstart技术

### 应答文件从哪来?

## 使用应答文件

### 创建应答文件

1. 安装一个图形的工具,生成应答文件

[root@svr7 ~]# yum -y install system-config-kickstart.noarch

1. 运行system-config-kickstart进行生成应答文件

[root@svr7 ~]# system-config-kickstart

首先查看”软件包选择”是否可以使用

需要本地光盘Yum仓库的支持,必须要求 光盘的仓库标识为 [development]

[root@svr7 ~]# vim /etc/yum.repos.d/rhel7.repo

[development]

......

[root@svr7 ~]# system-config-kickstart

查看”软件包选择”是否可以使用

1. 保存应答文件

[root@svr7 ~]# ls /root/ks.cfg

/root/ks.cfg

1. 共享应答文件

[root@svr7 ~]# cp /root/ks.cfg /var/www/html

[root@svr7 ~]# firefox 192.168.4.7/ks.cfg

1. 通过菜单文件,指定应答文件的位置

[root@svr7 ~]# 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

### 启用应答文件

## 无人值守装机

1. 部署无人值守安装
2. 搭建Web服务器,共享光盘所有内容

[root@svr7 ~]# yum -y install httpd

[root@svr7 ~]# mkdir /var/www/html/rhel7

[root@svr7 ~]# mount /dev/cdrom /var/www/html/rhel7

[root@svr7 ~]# systemctl restart httpd

[root@svr7 ~]# systemctl enable httpd

[root@svr7 ~]# firefox 192.168.4.7/rhel7

总结:DHCP-----> IP地址、next-server、filename

tftp -----> pxelinux.0

pxelinux.0------>读取菜单文件 default

default----->vesamenu.c32、splash.png 、ks.cfg......

ks.cfg-----> root密码 分区 时区 .....

url --url=”http://192.168.4.7/rhel7”

课外扩展:

让客户端安装完系统,Yum仓库可用(利用ks应答文件安装后脚本实现)

在PXE服务器上修改ks应答文件：

%post --interpreter=/bin/bash

useradd lisi

echo 123 | passwd --stdin lisi

rm -rf /etc/yum.repos.d/\*

cd /etc/yum.repos.d/

wget http://192.168.4.7/rhel7.repo

%end

### PXE+kickstart测试

# 总结和答疑

## TFTP连接失败

### 问题现象

### 故障分析及排除

## DHCP服务冲突

### 问题现象

### 故障分析及排除

## PXE应答不完整

### 问题现象

### 故障分析及排除

### 

# rsync同步操作

## rsync基本使用

### rsync概述

### rsync同步操作

命令用法

-rsync [选项...] 源目录 目标目录

同步与复制的差异

-复制:完全拷贝源到目标

-同步:增量拷贝,只传输变化过的数据

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yasu

### 同步控制

本地同步

-rsync [选项...] 本地目录1 本地目录2 //同步整个文件夹

-rsync [选项...] 本地目录1/ 本地目录2 //只同步目录下的数据

[root@svr7 ~]# mkdir /nsd

[root@svr7 ~]# mkdir /dir

[root@svr7 ~]# cp /etc/passwd /etc/fstab /nsd

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# ls /dir

[root@svr7 ~]# rsync -avz /nsd/ /dir/

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# ls /dir

[root@svr7 ~]# touch /nsd/1.txt

[root@svr7 ~]# rsync -avz /nsd/ /dir

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# ls /dir

[root@svr7 ~]# touch /dir/{10..15}.txt

[root@svr7 ~]# ls /dir

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# rsync -avz --delete /nsd/ /dir

[root@svr7 ~]# ls /dir

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# mkdir /nsd/test

[root@svr7 ~]# ls /dir

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# rsync -avz --delete /nsd/ /dir

[root@svr7 ~]# ls /dir

[root@svr7 ~]# ls /nsd

## rsync+SSH同步

### 用法及服务端要求

与远程的SSH目录保持同步

-下行:rsync [...] user@host:远程目录 本地目录

-上行:rsync [...] 本地目录 user@host:远程目录

### 下行同步示例

虚拟机A:

[root@svr7 ~]# mkdir /nsd/test

[root@svr7 ~]# ls /dir

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# rsync -avz --delete /nsd/ /dir

[root@svr7 ~]# ls /dir

[root@svr7 ~]# ls /nsd

[root@svr7 ~]# rm -rf /opt/\*

[root@svr7 ~]# cp /etc/resolv.conf /etc/hosts /opt/

[root@svr7 ~]# ls /opt

[root@svr7 ~]# rsync -avz --delete /opt/ root@192.168.4.207:/opt/

### 上行同步示例

# inotify实时同步

## 部署监控环境

### 同步的实时性

一 前提ssh实现无密码验证

1. 生成公钥与私钥

[root@svr7 ~]# ssh-keygen #一路回车

[root@svr7 ~]# ls /root/.ssh

2.传递公钥到对方服务器

[root@svr7 ~]# ssh-copy-id [root@192.168.4.207](mailto:root@192.168.4.207)

3.验证同步无需密码

[root@svr7 ~]# rsync -avz --delete /opt/ [root@192.168.4.207:/opt/](mailto:root@192.168.4.207:/opt/)

/root/.ssh/known\_hosts:记录曾经远程管理过的主机信息

### 安装inotify-tools工具

二 监控目录/opt/变化

Linux内核的inotify机制

-提供事件响应式的文件系统通知机制

-安装inotify-tools控制工具可调用此机制实现监控

inotify-tools:源码包

1. 准备源码包inotify-tools-3.13.tar.gz

[root@svr7 ~]# tar -xf /root/tools.tar.gz -C /

[root@svr7 ~]# ls /tools/

inotify-tools-3.13.tar.gz other

1. 进行tar解包

[root@svr7 ~]# mkdir /rsync

[root@svr7 ~]# tar -xf /tools/inotify-tools-3.13.tar.gz -C /rsync/

[root@svr7 ~]# ls /rsync/

[root@svr7 ~]# ls /rsync/inotify-tools-3.13/

1. 安装开发工具

[root@svr7 ~]# yum -y install make gcc

1. ./configure 配置,指定安装目录/功能模块等选项

--prefix=指定安装目录

检测系统是否安装gcc

[root@svr7 ~]# cd /rsync/inotify-tools-3.13/

[root@svr7 inotify-tools-3.13]# ./configure

常见报错:没有安装gcc

1. 进行make编译,生成可以执行的程序

[root@svr7 inotify-tools-3.13]# make

1. 进行make install 安装

[root@svr7 inotify-tools-3.13]# make install

[root@svr7 inotify-tools-3.13]# ls /usr/local/bin/inotifywait #检测程序的生成

### inotifywait监控

inotifywait:可以监控目录内容的变化

inotifywait监控

-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,指定监控的modify、move、create、delete、attrib事件类别

1）modify：文档的内容变更

2）attrib：文档的权限、归属、修改时间、访问控制列表等变更

3）move：文档被改名、移出或移入

4）create：在被监控的目录中增加了新的子目录或文档

5）delete：在被监控的目录中删除了某个子目录或文档

## 配置实时同步

### inotify与rsync的结合

### 编写同步脚本

书写Shell同步脚本

for循环:处理有次数的重复性操作

while死循环:适合处理没有次数的重复性操作

格式:

for 变量 in 值列表

do

重复执行的语句

done

while [条件]

do

重复执行的语句

done

[root@svr7 ~]# vim /root/rsync.sh

#!/bin/bash

while inotifywait -rqq /opt

do

rsync -az --delete /opt/ [root@192.168.4.207:/opt/](mailto:root@192.168.4.207:/opt/)

done &

[root@svr7 ~]# /root/rsync.sh

[root@svr7 ~]# pgrep -l rsync

10175 rsync.sh

[root@svr7 ~]# kill 10175

### 验证实时同步效果

# Cobbler装机平台

## Cobbler概述

### Cobbler简介

Cobbler装机平台,不同版本的多系统的安装

一 安装一个CentOS虚拟机

真机:

[root@room9pc01 ~]# ls /ISO/

CentOS-7-x86\_64-DVD-1708.iso

虚拟机要求:

1. 硬盘大小50G
2. 网络类型private1
3. 软件包选择”带GUI的服务器”
4. 分区选择”自动分区”
5. 管理员root密码为1
6. 创建普通用户lisi
7. 将CentOS放入光驱设备,搭建本地Yum仓库
8. 配置IP地址:192.168.4.80/24
9. 配置主机名:Cobbler.tedu.cn
10. 虚拟机设置

1.设置防火墙为trusted

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

2.当前及永久设置SELinux状态为permissive

[root@Cobbler ~]# setenforce 0 #当前临时关闭

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

SELINUX=permissive

1. 利用scp真机传递Cobbler.zip包到虚拟机192.168.4.80中

[root@room9pc01 ~]# scp /root/桌面/Cobbler.zip [root@192.168.4.80:/root/桌面/](mailto:root@192.168.4.80:/root/桌面/)

1. 搭建Cobbler装机平台

Cobbler概述软件,管理dhcp、tftp、Web服务

自由的导入镜像与ks应答文件

1. 解压Cobbler.zip包

[root@Cobbler ~]# unzip /root/桌面/Cobbler.zip -d /

[root@Cobbler ~]# cd /Cobbler/

[root@Cobbler Cobbler]# ls

[root@Cobbler Cobbler]# unzip cobbler.zip -d /opt

[root@Cobbler Cobbler]# ls /opt/cobbler/

1. 安装cobbler主程序、工具包等

[root@Cobbler cobbler]# yum -y install \*

## Cobbler平台部署

### 安装Cobbler组件

### 配置Cobbler

cobbler网络装机部署

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

[root@Cobbler cobbler]# vim /etc/cobbler/settings

next\_server: 192.168.4.80 #设置下一个服务器还为本机

server: 192.168.4.80 #设置本机为cobbler服务器

manage\_dhcp: 1 #设置cobbler管理dhcp服务

pxe\_just\_once: 1 #防止客户端重复安装操作系统

开机启动项:匹配及停止

1.硬盘启动 2.光驱设备 3. U盘 4.网络引导

1. 配置cobbler的dhcp

[root@Cobbler ~]# vim /etc/cobbler/dhcp.template

:%s /旧/新/g #全文替换

:%s /192.168.1/192.168.4/g #全文替换

5 次替换，共 4 行

1. 绝对路径解压cobbler\_boot.tar.gz #众多的引导文件

[root@Cobbler cobbler]# tar -tf /Cobbler/cobbler\_boot.tar.gz #查看包里面内容

[root@Cobbler cobbler]# tar -xPf /Cobbler/cobbler\_boot.tar.gz #绝对路径释放

[root@Cobbler cobbler]# ls /var/lib/cobbler/loaders/

1. 启动相关服务

[root@Cobbler cobbler]# systemctl restart cobblerd

[root@Cobbler cobbler]# systemctl enable cobblerd

[root@Cobbler cobbler]# systemctl restart httpd

[root@Cobbler cobbler]# systemctl enable httpd

[root@Cobbler cobbler]# systemctl restart tftp

[root@Cobbler cobbler]# systemctl enable tftp

[root@Cobbler cobbler]# systemctl restart rsyncd #同步服务

[root@Cobbler cobbler]# systemctl enable rsyncd

1. 同步刷新cobbler配置

[root@Cobbler cobbler]# cobbler sync #检测所有配置

......

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

[root@Cobbler cobbler]# firefox https://192.168.4.80/cobbler\_web

导入安装镜像数据

cobbler import --path=/CentOS --name=CentOS7

cobbler导入的镜像放在:/var/www/cobbler/ks\_mirror

[root@Cobbler cobbler]# cobbler list #查看有哪些系统

distros:

CentOS7-x86\_64 #安装客户端至少2G内存

profiles:

CentOS7-x86\_64

# 总结和答疑

## rsync+inotify同步

### 问题现象

### 故障分析及排除

# 计算机网络

## 计算机网络概述

### 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互连

-实现资源共享、信息传递

### 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

### 计算机网络发展阶段

60年代

-分组交换

70-80年代

-TCP/IP

90代年后

-Web技术

### 网络标准

协议

-语法

-语义

-同步

标准

-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)

### 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

## 网络设备及拓扑

### 网络设备生产厂商

-Cisco(思科)

-华为

### 路由交换设备

网络安全设备

-防火墙

-VPN设备 虚拟专用网络

### 网络拓扑结构

线缆连接计算机和网络设备的布局

-点对点

-总线型

-环型

-星型及扩展的星型

-网状

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

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

网络拓扑结构

-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

# 网络通信参考模型

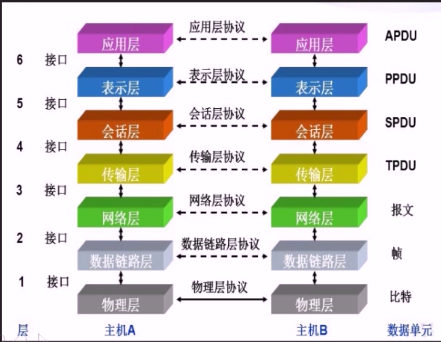
## OSI参考模型

### 计算机网络的功能

### 网络分层

### 邮局实例

### OSI协议模型



OSI的七层框架功能

应用层 网络服务与最终用户的一个接口

表示层 数据的表示、安全、压缩

会话层 建立、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

## TCP/IP模型

### OSI模型、TCP/IP模型



应用层:HTTP、FTP、TFTP、SMTP、SNMP、DNS

传输层:TCP、UDP

网络层:ICMP、IGMP、IP、ARP、RARP

数据链路层、物理层:由底层网络定义的协议

应用层 上层数据

传输层 TCP头部 上层数据 数据段

网络层 IP头部 TCP头部 上层数据 数据包

数据链路层 MAC头部 IP头部 TCP头部 上层数据 数据帧

物理层 比特流

### TCP/IP协议族的组成

### 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”规范的一组规则就是协议

数据通信协议的定义

-决定数据的格式和传输的一组规则或者一组惯例

协议分层

网络通信的过程很复杂

-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

### 协议数据单元(PDU)

### 设备与层的对应关系

应用层 计算机

传输层 防火墙

网络层 路由器

数据链路层 交换机

物理层 网卡

# 物理层解析

## 物理层

### 物理层-网络的基础

### 以太网接口

RJ-45

-RJ是描述公用电信网络的接口,常用的有RJ-11和RJ-45

光纤接口

-用以稳定地但并不是永久地连接两根或多跟光纤的无源组件

FC 圆形带螺纹光纤接头

ST 卡接式圆形光纤接头

SC 方型光纤接头

LC 窄体方形光纤接头

MT-RJ 收发一体的方型光纤接头

### 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,以减少对邻近线对的电气干扰

由若干对双绞线构成的电缆被称为双绞线电缆

非屏蔽双绞线UTP和屏蔽双绞线STP

### 双绞线的标准

类型 传输速率

cat5 100Mbps

cat5e 100Mbps

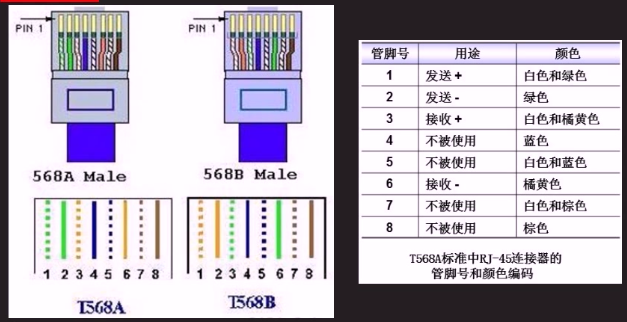
cat6 1000Mbps(1Gbps)

cat7 10000Mbps(10Gbps)

### 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



-标准网线

-交叉网线

-全反线(控制线 console线)

### 物理层的设备

网络接口卡

-连接计算机和网络硬件

-由一个唯一的网络节点地址

-按照速率可分为10M网卡、10/100M自适应网卡和千兆网卡

-按照总线类型可分为ISA网卡、PCI网卡等

-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

-便携式电脑可使用PCMCIA网络接口卡

中继器

-能放大信号

-延长网络传输距离

-只包含由一个输入端口和一个输出端口,所以只能接收和转发数据流

-成本低

## Packet Tracer软件

### Packet Tracer软件简介

Cisco Packet Tracer软件

-Packet Tracer是由Cisco公司发布的一个辅助学习工具,为学习思科网络课程的初学者去设计、配置、排除网络故障提供了网络模拟环境.用户可以在软件的图形用户界面上直接使用拖拽方法建立网络拓扑,并可提供数据包在网络中行进的详细处理过程,观察网络实时运行情况.可以学习IOS的配置、锻炼故障排查能力.

### Packet Tracer软件使用

# 交换机命令行

## 交换命令行模式

### 交换机的命令行模式

Cisco交换机的命令行

用户模式

Switch>

特权模式(一般用于查看配置信息)

Switch>enable

Switch#

全局配置模式(所做的配置对整个设备生效)

Switch#configure terminal

Switch(config)#

接口模式

Switch(config)#interface fastEthernet 0/1

Switch(config-if)#

-interface:关键字

-fastethernet:接口类型

-0/1:”0”表示模块号,”1”表示端口号

### 模式间的转换

模式间转换

-exit命令

返回上一级模式

-end命令

回到特权模式

-快捷键

<ctrl+Z>退出到特权模式

中断操作

ctrl+shift+6

? 相当于Linux系统中的双tab键

## 交换机命令行配置

### 配置主机名

修改交换机主机名,要在全局配置模式

Switch>en

Switch#conf t

Switch(config)#hostname sw01

### 查看交换机的配置

查看交换机当前的运行配置,要在特权模式

sw01#show running-config

### 配置enable明文口令

sw01(config)#enable password cisco

sw01(config)#exit

sw01(config)#show running-config

### 保存交换机的配置

sw01#copy running-config startup-config 或write

### 恢复设备出厂默认值

sw01#erase startup-config

重启 reload

# 总结和答疑

## 自动退出配置界面

### 问题现象

### 原因分析

## 打断命令输入

打断命令输入

-配置输入日志同步

-sw01(config)#line console 0

-sw01(config-line)#logging synchronous

自动退出配置界面

-配置控制台会话时间永不超时

-sw01(config)#line console 0

-sw01(config-line)#exec-timeout 0 0

### 问题现象

### 原因分析

## 禁用DNS查询

禁用DNS查询

-sw01#no ip domain-lookup

### 问题现象

### 原因分析

### 

# 数据链路层解析

## 数据链路层

### 以太网MAC地址

ip 192.168.0.1 32位

11000000.10101000.00000000.00000001

mac 物理地址 硬件地址 48位

用来识别一个以太网上的某个单独的设备或一组设备

d8:9e:f3:23:fa:40 d8-9e-f3-23-fa-40 d89e-f323-fa40

48位中的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)

单播 1对1

组播 1对多

广播 1对所有

### 以太网帧格式

数据链路层的协议数据单元

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

mtu 最大传输单元

## 以太网交换机

### 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的转发数据,因此交换机工作在数据链路层

### 交换机的工作原理

交换机的转发原理

-初始状态

-MAC地址学习

-广播未知数据帧

-接收方回应

-交换机实现单播通信

学习

-MAC地址表是交换机通过学习接收的数据帧的源MAC地址来形成的

广播

-如果目标地址在MAC地址表中没有,交换机就向除接收到该数据帧的接口外的其他接口广播该数据帧

转发

-交换机根据MAC地址表单播转发数据帧

更新

-交换机MAC地址表的老化时间是300秒

-交换机如果发现一个帧的入接口和MAC地址表中的源MAC地址的所在接口不同,交换机将MAC地址重新学习到新的接口

### 查看MAC地址表

ipconfig /all windows中查看所有ip地址信息

physical address 物理地址(mac地址)

Switch#show mac-address-table 在交换机中查看mac地址表

### 交换机的工作原理案例

### 广播域

广播域指接收同样广播消息的节点的集合

交换机的所有接口默认属于同一个广播域

# VLAN技术及应用

## VLAN

### VLAN概述

什么是VLAN

-Virtual LAN(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

-交换机的所有接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,降低了网络的效率

-为了分割广播域,引入了VLAN

VLAN 的作用

-广播控制

-增加安全性

-提高带宽利用

-降低延迟

基于接口划分的静态VLAN

## VLAN配置

### 静态VLAN的配置

创建VLAN 的步骤

-创建VLAN

-将接口加入到相应的VLAN中

-验证

在全局配置模式下创建VLAN

Switch(config)#vlan vlan-id

删除已创建的vlan 创建的前面加no

Switch(config-name)#name \*\*\* //为该vlan命名

### 将接口加入VLAN

Switch(config)#interface f0/1 //进入f0/1接口

Switch(config-if)#switchport access vlan vlan-id //把该接口放入vlan-id

Switch(config-if)#no switchport access vlan vlan-id

也可以同时将多个接口添加到某个VLAN中:

Switch(config)#interface range f0/1-10 //同时进入1-10接口

### 验证VLAN的配置

Switch#show vlan brief //查看vlan摘要信息

# TRUNK

## Trunk原理

### 交换机之间的VLAN通信

### VLAN标识

access 接入链路 承载1个vlan

trunk 中继链路 承载多个vlan

### VLAN标识的种类

以太网上实现中继可用两种封装类型

-ISL(Cisco私有协议)

-IEEE 802.1Q(dot1q)

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

类型 IEEE ISL

标记方式 公有 私有

标记方法 外部标记 内部标记

标记长度 30字节 40字节

RIP的最大跳数是15.

请描述基于CEF的快速转发有哪两个信息表

转发信息库(FIB):FIB类似于路由包,包含路由表中转发信息的镜像.当网络的拓扑发生变化时,路由表将被更新,而FIB也将随之变化,这些信息是根据路由表中的信息得到的.

邻接关系表:存储第2层编址信息,对于每个FIB条目,邻接关系表中都包含相应的第2层地址.

## Trunk配置

### 配置接口为Trunk模式

Switch(config)#interface f0/7 //进入f0/7接口

Switch(config-if)#switchport mode trunk //将接口改为中继链路模式

### 恢复接口默认模式

前面+no

### 查看接口模式

show vlan brief或show running-config

了解:Switch#show interface interface-id switchport

### 配置VLAN Trunk实例

## 以太通道

### 以太通道概述

也称为以太接口捆绑、接口聚或以太链路聚集。英文名EtherChannel

以太通道为交换机提供了接口捆绑的技术,允许两个交换机之间通过两个或多个接口并行连接,同时传输数据,以提供更高的带宽和可靠性

### 配置以太网通道

EtherChannel-以太通道

-多条线路负载均衡,带宽提高

-容错,当一条线路失效时,其他线路通信,不会丢包

配置接口为以太通道模式

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

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

验证以太通道

Switch(config-if)#interface f0/1

Switch(config-if)#shutdown //关闭接口

Switch(config-if)#no shutdown //打开接口

Switch#show etherchannel summary //查看以太通道汇总信息

配置思路:

1. 将7、8口都加入相同vlan,都配置好trunk
2. 将7、8口都组建为以太通道
3. 两台交换机都做相同操作

### 以太通道配置指导原则

参与捆绑的接口必须属于同一个vlan,如果是在中继模式下,要求所有参加捆绑的接口都是在中继模式下

如果接口配置的是中继模式,那么应该在链路的两端将通道中的所有接口配置成中继模式

# 网络层解析

## 网络层

### 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

连接不同范围的主机

### ICMP协议

ICMP通过IP数据报传送,用来发送错误和控制信息

ICMP定义了很多信息类型,例如:

-目的地不可达

-TTL超时

-信息请求

-信息应答

-地址请求

-地址应答

ICMP检测双向通路的连通性

Ping命令使用ICMP协议

-Ping [-t] [-l 字节数] 目标IP或主机名

## 路由原理及配置

### 什么是路由

将数据包从一个网络发送到另一个网络

-需要依靠路由器来完成

-路由器只关心网络的状态,决定最佳路径

### 路由器怎么工作

-识别数据包的目标IP地址

-识别数据包的源IP地址(主要用于策略路由)

-在路由表中发现可能的路径

-选择路由表中到达目标最好的路径

-维护和检查路由信息

### 如何获得路由表

路由表 标识了当前设备能够到达的网段,以及方向

Router#show ip route //查看路由表

### 静态路由

C 直连路由 产生条件是接口配置ip并处于开启状态

S 静态路由 由管理员手工配置,并且设备两端都要相同的配置,否则有去无回,由目标网段

(网络ID)+子网掩码+下一跳地址构成,单向条目

S\* 默认路由 特殊的静态路由,在路由表中可以到达任意地点,其他路由条目无法到达时,将使用默认路由,通常在企业网关设备对外出口使用.

路由环路(避免)

ip route 0.0.0.0 0.0.0.0 下一跳

动态路由 使用某种动态路由协议实现,可以做到自动学习路由条目,根据网络状态自动变化来达成网络互通的目的.

宣告 对外通告本设备能够到达的网段

反掩码

255.255.255.0

0.0.0.255

### 配置静态路由

ip route命令

-指定到达IP目的网络

第一台路由器

Router(config)#ip route 192.168.3.0 255.255.255.0 192.168.2.2 //配置前往3.0网段并且出口方向是192.168.2.2 的静态路由

第二台路由器

Router(config)#ip route 192.168.1.0 255.255.255.0 192.168.2.1 //配置前往3.0网段并且出口方向是192.168.2.1 的静态路由

思路:

1. 考虑所要前往的目标网段
2. 考虑前往的目标网段第一步往哪个ip跳
3. 使用静态路由添加目标网段的网络ID与下一跳

### 默认路由

-默认路由是一种特殊的静态路由

-默认路由的目标网络为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,才会使用默认路由,一般在企业网关出口使用

# VLAN间通讯

## 三层交换概述

### 三层交换技术

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

### 虚接口概述

在三层交换机上配置的VLAN接口为虚接口

使用SVI(交换虚拟接口)实现VLAN间路由

### 三层交换机的配置

SVI 交换机虚拟接口

Router(config)#vlan 2

Router(config)#interface vlan 2

Router(config-if)#ip address 192.168.2.254 255.255.255.0

Router(config)#interface f0/2

Router(config-if)#switchport access vlan 2 //将接口加入vlan2

Router(config)#ip routing //开启路由功能

创建所需VLAN

为每个VLAN配置IP地址

启用SVI接口

启用三层交换机的IP路由功能

如果需要,配置三层交换机的动态或静态路由

三层交换机上的路由接口

-三层交换机的物理接口默认是二层接口

-可以转换为三层接口

-转换为三层接口后,该接口不属于任何VLAN

-可以像路由器接口一样使用

思路:

1. 添加二层交换机一台,创建vlan2 vlan3
2. 将二层交换机f0/2口加入vlan2,f0/3口加入vlan3 f0/24口配置为中继模式
3. 在三层交换机的f0/24口上封装802.1q协议,然后配置为中继模式

三层交换机配置中继链路,先封装802.1q协议,再配置

Switch(config)#interface fastEthernet 0/24

Switch(config-if)#switchport trunk encapsulation dot1q //打封装

Switch(config-if)#switchport mode trunk

三层交换机配置：

Switch(config)#interface fastEthernet 0/23

Switch(config-if)#no switchport //将交换机接口转换为路由器接口

Switch(config-if)#ip address 192.168.4.1 255.255.255.0 //配置ip

## 动态路由

### 认识动态路由

动态路由

-基于某种路由协议实现

动态路由特点

-减少了管理任务

-占用了网络带宽

### 动态路由协议OSPF

全称为Open Shortest Path First (开放式最短路径优先)

适合大中型网络使用

### OSPF基本配置

配置动态路由：

三层交换机配置

Switch(config)#router ospf 1

Switch(config-router)#network 192.168.1.0 0.0.0.255 area 0 //宣告本机所在的直连网段,区域都是0

Switch(config-router)#network 192.168.2.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.2.0 0.0.0.255 area 0

Switch(config-router)#network 192.168.2.0 0.0.0.255 area 0

路由器配置

Router(config)#router ospf 1

Router(config-router)#network 192.168.4.0 0.0.0.255 area　0

什么是TTL,作用并简要描述TTL的原理

TTL是数据生命周期

作用:避免数据在网络中无限循环转发

原理:当网络中的数据包每经过一个路由器TTL值减,当TTL值为0时,数据包丢弃.

# 传输层

## 传输层概述

### 传输层的作用

定义端口号 65535个 0号保留 1~1023是知名服务端口号

为了实现端到端的传输

### 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

## TCP协议

### TCP的封装格式

0-15源端口号

16-31目标端口号

### TCP的连接与断开

syn:准备与对方建立连接

ack:确认信息

fin:与对方断开连接

TCP的连接-三次握手

1. 发送SYN,请求建立连接
2. 发送SYN、ACK
3. 发送ACK

TCP 的断开-四次断开

1. 发送FIN,请求断开连接
2. 发送ACK
3. 发送FIN,请求断开连接
4. 发送ACK

### TCP的应用

FTP、Telnet、SMTP、DNS、HTTP

## UDP协议

### UDP的封装格式

### UDP的应用

TFTP、DNS、NTP(端口123)

### 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:例如TFTP协议

# ACL

## 访问控制列表概述

### 访问控制列表作用

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

### 访问控制列表的工作原理

### 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访问控制列表号是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

命名访问控制列表

## 标准ACL配置

### 标准访问控制列表的配置

permit 允许

deny 拒绝

反掩码

192.168.2.1

255.255.255.0

0.0.0.0

0匹配

1不匹配

使用标准acl限制pc2

Router(config)#access-list 1 deny 192.168.2.1 0.0.0.0

或

Router(config)#access-list 1 deny host 192.168.2.1

以上两条配置其中一条即可,效果相同

Router(config)#access-list 1 permit any //放行其他数据

在接口中应用acl

Router(config)#interface gigabitEthernet 0/1

Router(config-if)#ip access-group 1 in

### 扩展访问控制列表的配置

Router(config)#access-list 1 permit host 192.168.2.1 //允许192.168.2.1通过

使用扩展ACL过滤数据

Router(config)#access-list 100 deny tcp host 192.168.2.1 host 192.168.1.1 eq 21 //禁止2.1访问1.1的ftp服务

Router(config)#access-list 100 deny tcp host 192.168.2.2 host 192.168.1.1 eq 80 //禁止2.2访问1.1的web服务

Router(config)#access-list 100 permit ip any any //放行其他业务

Router(config)#interface gigabitEthernet 0/1

Router(config-if)#ip access-group 100 in //在接口中应用acl

# NAT

## NAT概述

### NAT的作用

Network Address Translation,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互联网等外部网络上。

### 私有ip地址分类

A类 10.0.0.0~10.255.255.255

B类 172.16.0.0~172.31.255.255

C类 192.168.0.0~192.168.255.255

### NAT的特性

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

不支持某些应用,可以通过静态NAT映射来避免

### NAT实现方式

-静态转换

-端口多路复用

### NAT的工作过程

静态 一般用于服务器发布服务到公网,一台服务器使用一个公网ip

静态端口映射 只发布某个服务到公网时使用

PAT 一般用于仅访问外网时使用,比如员工办公室

## 静态转换

### 静态NAT

100M 500～800 动态公网ip 故障率

50M 40000 固定公网ip

静态 一般用于服务器发布服务到公网，一台服务器使用一个公网ip

静态端口映射 只发布某个服务到公网时使用

PAT 一般用于仅访问外网时使用，比如员工办公室

### 静态NAT的配置

Router(config)#ip nat inside source static 192.168.1.1 100.0.0.2 //将1.1转换为100.0.0.2

Router(config)#interface g0/1

Router(config-if)#ip nat outside //外网区域设置外边

Router(config)#interface g0/0

Router(config-if)#ip nat inside //外网区域设置里边

### NAT端口映射配置

静态端口映射

Router(config)#ip nat inside source static tcp 192.168.1.1 80 100.0.0.2 80 //只发布80端口的服务到公网

## 端口多路复用(PAT)

### PAT的作用

### PAT的配置

使192.168.1.2也能发布服务到外网，新购买公网地址100.0.0.3

Router(config)#ip nat inside source static 192.168.1.2 100.0.0.3

静态端口映射

Router(config)#ip nat inside source static tcp 192.168.1.1 80 100.0.0.2 80 //只发布80端口的服务到公网

PAT

Router(config)#access-list 1 permit 192.168.1.0 0.0.0.255 //限定可以访问外网的主机范围是192.168.1.0网段

Router(config)#ip nat inside source list 1 interface g0/1 overload //使用pat复用外网接口地址，使192.168.1.0网段所有主机可以访问外网（外网设备无法访问内网设备）

### 跟踪NAT

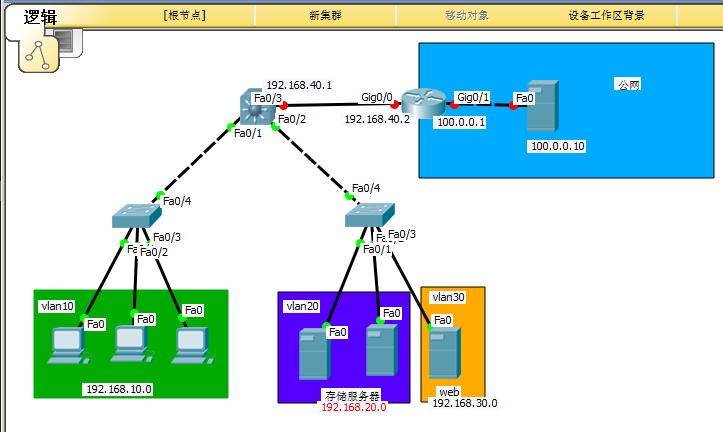
案例:

需求:

1. vlan10区域的员工可以访问外网
2. 所有服务器可以访问外网
3. web服务要发布到外网,使客户可以从外网看到公司主页

提示:

1. 三层交换机要创建相关vlan并配置ip作为网关
2. 需要配置静态路由或者动态路由
3. 第二台二层交换机需要配置中继链路



### 

# STP

## STP生成树算法

### 广播风暴的产生

### STP概述

-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.

交换机端口由哪5种STP状态

转发(Forwarding)

学习(Learning)

侦听(Listening)

阻塞(Blocking)

禁用(Disabled)

### 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

取值范围:0~65535

缺省值:32768=4096\*8

## STP配置

### PVST+的配置命令

启用生成树命令

Switch(config)#spanning-tree vlan 1

指定根网桥

Switch(config)#spanning-tree vlan 1 priority 4096n //修改生成树优先级为28672

Switch(config)#spanning-tree vlan 1 priority root { primary | secondary } //修改主根|次根

查看生成树的配置

Switch#show spanning-tree

查看生成树的配置

Switch#show spanning-tree vlan vlan-id

FWD 转发状态

BLK 阻塞状态

### PVST+配置的意义

-配置网络中比较稳定的交换机为根网桥

-利用PVST+实现网络的负载分担,PVST+ 每vlan生成树

每台三层交换机配置:

Switch(config)#vlan 2 //创建vlan2

Switch(config)#interface range fastEthernet 0/1-3

Switch(config)#switchport trunk encapsulation dot1q //打封装

Switch(config-if-range)#switchport mode trunk //中继链路

MS1

vlan1 主根 vlan2 次根

Switch(config)#spanning-tree vlan 1 priority 24576 //修改生成树优先级为24576

Switch(config)#spanning-tree vlan 2 root secondary //指定当前交换机为vlan2的次根

MS2

vlan1 次根 vlan2 主根

Switch(config)#spnning-tree vlan 1 priority 28672 //修改生成树优先级为28672 (vlan1的次根)

Switch(config)#spanning-tree vlan 2 root parimary //指定当前交换机为vlan2的主根

# HSRP

## HSRP概述

### HSRP的相关概念

热备份路由选择协议

-HSRP(Hot Standby Routing Protocol)

-Cisco私有协议

作用:确保了当网络边缘设备或接入链路出现故障时,用户通信能迅速并透明地恢复,以此为 IP 网络提供冗余性。

通过使用同一个虚拟 IP 地址和虚拟 MAC 地址, LAN 网段上的两台或者多台路由器可以作为一台虚拟路由器对外提供服务。

## HSRP配置

### HSRP的配置命令

MS1

Switch(config)#interface vlan 1

Switch(config-if)#standby 1 ip 192.168.1.254 //开启热备份,组号为1,虚拟路由器ip是192.168.1.254

MS2

Switch(config)#interface vlan 1

Switch(config-if)#standby 1 ip 192.168.1.254

占先权 当本机优先级大于其他主机时,立刻称为活跃路由器

### HSRP的配置案例

### 

# 综合项目

## 目前网络环境介绍

### 目前网络环境

### 目前网络技术

## 目前网络拓扑结构

### 目前网络拓扑

# 网络升级

## 项目规划

### 问题分析

### 项目需求

### 项目技术

## 项目规划后拓扑

### 项目拓扑

### 

# 网络部分

## 任务介绍

### 搭建教学办公网络

### 企业网络升级

## 网络设备选项

### 品牌与型号

### 技术指标

# 服务器部分

## 任务目标

### 企业内网DNS

### PXE一键装机平台

### 搭建邮件平台

### Discuz!技术服务论坛

## 服务器选型

### 什么是服务器?

### 服务器分类

按应用规模

-入门级、工作组级、部门级、企业级

按设备外形

-机架式、塔式、刀片、机柜式

### 服务器选型参数

服务器尺寸

-1U=1.75英寸=4.445厘米

机柜容量、深度

-22U、37U、42U

-网络机柜,深600或800mm

-服务器机柜,深900或1000mm

支持的CPU颗数(x路)、CPU核心数

内存插槽数、最大支持容量

硬盘接口(SAS/SATA)、盘架个数、最大支持容量

RAID卡支持级别、RAID卡电池

电源数量(单/双电)、服务器整机功率

### 选项原则

什么是DOS,有什么特征?

拒绝服务攻击,DDOS

# Shell概述

## Shell环境及特性

### 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”

cat /etc/shells //查看本机可用解释器

### Shell的使用方式

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

-方便在后台静悄悄地运行

### 常见的Shell程序种类

如何切换Shell环境

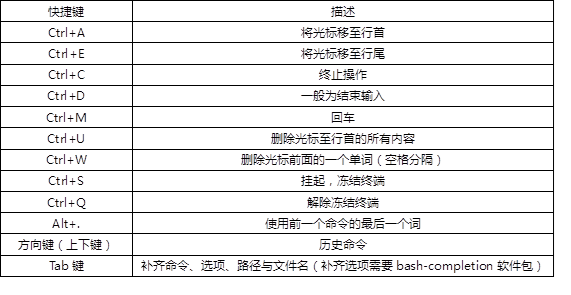
-通过usermod、chsh更改登录Shell

-手动执行目标Shell程序

### Bash基本特性

命令行环境回顾

-快捷键、Tab键补齐



-命令历史

-命令别名

-标准输入输出

-重定向

-管道操作

## 历史命令

### 存放位置及数量

默认记录1000条

HISTSIZE=1000

-保存位置:~/.bash\_history

-控制历史命令的数量:/etc/profile

### 操作历史命令

history工具

-history:查看历史命令列表

查看已为当前用户记录的历史命令条数:

#history | wc -l

-history -c:清空历史命令

> .bash\_history //清空历史记录文件

调用历史命令

-!78:执行历史记录中的第78条命令

-!str:执行最近一次以str开头的历史命令

## I/O设备与重定向

### 交互式硬件设备

标准输入:从该设备接收用户输入的数据

标准输出:通过该设备向用户输出数据

标准错误:通过该设备报告执行中的出错信息

类型 设备文件 文件描述号 默认设备

标准输入 /dev/stdin 0 键盘

标准输出 /dev/stdout 1 显示器

标准错误输出 /dev/stderr 2 显示器

### 重定向操作

改变标准输入/输出/错误输出的方向

类型 操作符 用途

重定向输入 < 将文本输入来源由键盘改为指定的文件

<< 标准输入来自命令行的一对分隔号的中间内容.

重定向输出 > 将命令行的正常执行输出保存到文件,而不是直接 显示在屏幕上

>> 与>类似,但操作是追加而不是覆盖

重定向错误 2> 将命令行的执行出错信息保存到文件,而不是直接 显示在屏幕上

2>> 与2>类似,但操作是追加而不是覆盖

混合重定向 &> 相当于>和2>,覆盖到同一个文件

# 编写及执行脚本

## 第一个Shell脚本

### 什么是Shell脚本?

提前写好可执行语句,能够完成特定任务的文件

-顺序、批量化处理

-解释型程序

常见的脚本语言

Bash Shell

Python/Perl/Ruby

JSP/PHP/ASP/CGI

JavaScript .. ..

### Shell版HelloWorld的诞生

## 脚本构成及执行

### 规范的脚本构成

#!脚本声明(使用哪种解释器)

#注释信息(步骤、思路、用途、变量含义等)

可执行的语句(脚本代码)

### 脚本的执行方式

方法一,作为”命令字”

-指定脚本文件的路径,前提是有x权限

方法二,作为”参数”(不要求x权限)

- sh 脚本文件路径 //开启子进程

- bash 脚本文件路径 //开启子进程

- source 脚本文件路径 //不开启子进程

- . 脚本文件路径=source 脚本文件路径

### 调试Shell脚本

主要途径

-直接观察执行中的输出、报错信息

-通过sh -x开启调试模式

-在可能出错的地方设置echo

## 简单脚本应用

### 示例1:快速配置YUM

### 示例2:快速搭建FTP服务

# Shell变量

## 变量的设置和取消

### 什么是变量?

大小写字母,不以数字开头,不使用特殊符号,等号两边无空格

以固定名称存放,可能会变化的值

-提高脚本对任务需求、运行环境变化的适应能力

-方便在脚本中重复使用

### 定义/赋值/查看变量

-变量名=变量值

查看变量

-引用变量值:$变量名

-查看变量值:echo $变量名、echo ${变量名}

### 取消变量

变量的失效

-退出定义变量的Shell环境时,变量会自动失效

-也可手动取消:unset 变量名.. ..

## 变量的种类

### Shell变量的分类角度

存储类型

-整数型、浮点型、双精度浮点型、字符型、......

-Shell脚本语言对存储类型要求较松散

使用类型

类型 说明

环境变量 变量名通常都大写,由系统维护,用来设置工作环境,只有个别变量用 户可以直接更改

位置变量 bash内置,存储执行脚本时提供的参数

预定义变量 bash内置,一类有特殊用途的变量,可直接调用,但不能直接赋值或修 改

自定义变量 由用户自主设置、修改及使用

### 环境变量

配置文件

- /etc/profile、~/.bash\_profile

相关操作

-env:列出所有的环境变量

-set:列出所有变量

常见的环境变量

-PWD、PATH、USER、LOGNAME、UID、HOSTNAME

-SHELL、HOME、PS1、PS2、......

环境变量PS1表示Shell环境的一级提示符，即命令行提示符（\u 用户名、\h 主机名、\W 工作目录、\$ 权限标识）

[\u@\h \W]\$

环境变量PS2表示二级提示符，出现在强制换行、at任务编辑等场合

[root@svr5 ~]# echo $PS2                     //查看默认的二级提示

>

### 预定义变量

用来保存脚本程序的执行信息

-直接使用这些变量

-不能直接为这些变量赋值

变量名 含义

$0 当前所在的进程或脚本名

$$ 当前运行进程的PID号

$? 命令执行后的返回状态,0表示正常,1或其他值表示异常

$# 已加载的位置变量的个数

$\* 所有位置变量的值

### 位置变量

在执行脚本时提供的命令行参数

-表示为$n,n为序号

-$1、$2、.. .. ${10}、${11}、.. ..

## 变量值及范围控制

### 扩展赋值操作

区分三种定界符

-双引号””:允许扩展,以$引用其他变量

-单引号’’:禁用扩展,即便$也视为普通字符,屏蔽特殊符号的作用

-反撇号``:将命令的执行输出作为变量值

### read标准输入取值

read从键盘读入变量值完成赋值

-格式:read [ -p “提示信息” ]变量名

- -p可选,-t可指定超时秒数

[root@desktop0 ~]#read -p "请输入密码:" password -t 10

stty -echo

终端显示控制

- stty -echo:关闭终端输出(无显示)

- stty echo:恢复终端输出(显示)

### 变量的作用范围

局部变量

-新定义的变量默认只在当前Shell环境中有效

-无法在子Shell环境中使用

全局变量

-全局变量在当前Shell及子Shell环境中均有效

-使用export可将局部变量声明为全局变量

# 总结和答疑

## 变量定义

### 问题现象

# 2test=123

-bash:2test=123:command not found

### 原因分析

分析故障

-报错信息:my#test=123:command not found

分析故障原因

-变量名称不可以使用数字开头

-变量名不可以为特殊字符

-变量名可以为字母、数字、下划线

## 变量调用

### 问题现象

#test=12

#echo test

test

#echo $testyuan

### 原因分析

分析故障

调用变量未获得预期的值

分析故障原因

-调用变量时需要使用$加变量名

-调用变量时,为了防止歧义需要使用{}

## read命令

### 问题现象

# read -p “Please input pass:”pass

Please input pass:pass

#echo $pass

#

### 原因分析

分析故障

-read提示符和变量之间缺少空格

-导致读取变量内容为空

分析故障原因

-因为没有空格,read将提示符和变量识别为一个整体

### 

# 数值运算

## 整数运算

### 基本运算类别

四则运算

-加法:num1+num2

-减法:num1-num2

-乘法:num1\*num2

-整除:num1/num2

取余数运算

-求模:num1%num2

### expr运算工具

计算指定的表达式,并输出结果

-格式:expr整数1 运算符 整数2

-乘法操作应采用\\*转义,避免被作为Shell通配符

类型 运算符 示例

加法 + expr 43 + 21、expr $X + $Y

减法 - expr 43 - 21、expr $X - $Y

乘法 \\* expr 43 \\* 21、expr $X \\* $Y

除法 / expr 43 / 21、expr $X / $Y

取余数 % expr 43 % 21、expr $X % $Y

### $[]算式替换

使用$[]或$(())表达式

-格式:$[整数1 运算符 整数2 .. ..]

-乘法操作\*无需转义,运算符两侧可以无空格

-引用变量可省略$符号

-计算结果替换表达式本身,可结合echo命令输出

### 变量的自增/减等操作

使用$[]替换,或者let命令来完成

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

let运算操作并不显示结果,但是可以结合echo命令查看

简写表达式 完整表达式

i++ i=i+1

i-- i=i-1

i+=2 i=i+2

i-=2 i=i-2

i\*=2 i=i\*2

i/=2 i=i/2

i%=2 i=i%2

let a+a //结果不显示,单独使用意义不大,常用来作变量定义

let c=a+a //定义新变量,结果要使用echo $c查看

let a=a+10 //修改原有变量,结果要使用echo $a查看

let a+=10 //主流写法

## 小数运算

### 整数运算的局限性

Bash内建机制仅支持整数值运算

-expr命令、$[]算式替换 不支持有小数的运算

### 使用bc实现小数运算

多数Linux系统默认安装此工具

-支持高精度的数值运算

-直接运行bc可进入交互式运算界面,quit退出

-设置scale=n可约束小数位

结合管道向bc发送表达式

-多个表达式以分号分隔

-通过echo命令+管道传递要计算的表达式

[root@svr7 ~]# echo 'scale=3;10/3;1+2.3' |bc

3.333

3.3

编写脚本,通过 read 读取用户输入的一个圆半径,脚本输出圆的面积,面积公式如下:

圆面积=πr 2 ,提示 r 2 就是 r 乘 r,π是 3.14

#!/bin/bash

read -p "请输入圆的半径:" r

echo "scale=2;3.14\*$r\*$r" |bc

### 小数值的比较

基本用法

-echo “数值1 比较符 数值2” | bc

-如果表达式成立,则返回的计算结果为1,否则返回0

-常见比较操作:>、>=、<、<=、==、!=

# 条件测试

## 测试操作概述

### Shell脚本的智能化

为命令的执行提供最直接的识别依据

-文件或目录的读/写等状态

-数值的大小

-字符串是否匹配

-多个条件/操作的逻辑组合

### test测试操作

语法格式

-test 选项 参数

- [ 选项 参数 ]

## 常见的测试操作

### 整数值比较

[ 整数值1 操作符 整数值2 ]

操作符 含义

-eq 等于(Equal)

-ne 不等于(Not Equal)

-ge 大于或等于(Greater or Equal)

-le 小于或等于(Lesser or Equal)

-gt 大于(Greater Than)

-lt 小于(Lesser Than)

[root@svr7 ~]# [ $A -gt 10 ]                 //不支持小数比较

-bash: [: 20.4: integer expression expected

[root@svr7 ~]# who | wc -l

1

[root@svr7 ~]# N=$(who | wc -l)

[root@svr7 ~]# [ $N -ge 3 ] && echo "超过了" || echo "没超过"

没超过

[root@svr7 ~]# [ $(who | wc -l) -ge 3 ] && echo "超过了" || echo "没超过"

没超过

需要每分钟检查系统登录的账户总数,如果超过3人,则发邮件给管理员报警

#!/bin/bash

[root@svr7 ~]# [ $(who | wc -l) -ge 3 ] && echo 123 | mail -s test root

\* \* \* \* \* /root/user.sh //使用计划任务每分钟执行一次刚才的脚本,脚本要给x权限

测试前可以把/var/spool/mail/root文件删除,清空管理员的邮箱

### 字符串比较

[ 操作符 字符串 ]

操作符 含义

-z 字符串的值为空

-n 字符串的值不为空(相当于!-z)

[ 字符串1 操作符 字符串2 ]

操作符 含义

== 两个字符串相同

!= 两个字符串不相同

### 文件状态测试

[ 操作符 文件或目录 ]

操作符 含义

-e 判断对象是否存在(Exist),若存在则结果为真

-d 判断对象是否为目录(Directory),是则为真

-f 判断对象是否为一般文件(File),是则为真

-r 判断对象是否有可读(Read)权限,是则为真

-w 判断对象是否有可写(Write)权限,是则为真

-x 判断是否有可执行(eXcute)权限,是则为真

编写一个创建目录的脚本,如果目录存在则返回信息”目录已存在”,如果不存在,则创建.

[ -e $1 ] && echo "$1已存在" || mkdir $1 2>/dev/null

[ ! -e $1 ] && mkdir $1 2>/dev/null || echo "$1已存在"

## 组合多个条件

### 逻辑分隔操作

主要用法:

-命令1 操作符 命令2 .. ..

-[条件1] 操作符 [条件2] .. ..

操作符 含义

&& 给定条件必须都成立,整个测试结果才为真

|| 只要其中一个条件成立,则整个测试结果为真

命令1 && 命令2 //逻辑测试,仅当命令1运行成功才执行命令2

命令1 || 命令2 //逻辑测试,仅当命令1运行失败才执行命令2

命令1 ; 命令2 //前后命令无逻辑关系,运行命令1后继续运行命令2

ls && ls

ls && ls && ls

ls || ls

ls || ls || ls

ls && ls || ls

ls || ls && ls

### 条件组合应用

# if选择结构

## 单分支结构

当”条件成立”时执行命令序列

否则,不执行任何操作

### 语法格式及特点

### if单分支的执行流程

### 脚本应用示例

检测/media/cdrom目录，若不存在则创建

[root@svr5 ~]# vim mountdir.sh

#!/bin/bash

dir="/media/cdrom/"

if [ ! -d $dir ]

then

mkdir -p $dir

fi

[root@svr5 ~]# chmod +x mountdir.sh //添加可执行权限

## 双分支结构

### 语法结构及特点

当”条件成立”时执行命令序列1

否则,执行命令序列2

### if双分支的执行流程

### 脚本应用示例

检测并判断指定的主机是否可ping通

为了节省ping测试时间，可以只发送3个测试包（-c 3）、缩短发送测试包的间隔秒数（-i 0.2）、等待反馈的超时秒数（-W 1）

[root@svr7 ~]# ping -c 3 -i 0.2 -W 1 192.168.4.254

PING 192.168.4.254 (192.168.4.254) 56(84) bytes of data.

64 bytes from 192.168.4.254: icmp\_seq=1 ttl=64 time=0.099 ms

64 bytes from 192.168.4.254: icmp\_seq=2 ttl=64 time=0.185 ms

64 bytes from 192.168.4.254: icmp\_seq=3 ttl=64 time=0.153 ms

--- 192.168.4.254 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 400ms

rtt min/avg/max/mdev = 0.099/0.145/0.185/0.038 ms

[root@svr7 ~]# echo $?

0

#!/bin/bash

ping -c 3 -i 0.2 -W 1 172.25.0.11 &>/dev/null

if [ $? -eq 0 ];then

echo “通了”

else

echo “不通”

fi

#!/bin/bash

x=0

y=0

for i in {1..10}

do

ping -c 3 -i 0.2 -W 1 192.168.4.$i &>/dev/null

if [ $? -eq 0 ];then

echo "172.25.0.$i 通了" >> /opt/ok.txt

let x++

else

echo "172.25.0.$i 不通" >> /opt/ok.txt

let y++

fi

done

echo "${x}台通了,${y}台不通"

## 多分支结构

### 语法格式及特点

相当于if语句嵌套

针对多个条件分别执行不同的操作

### if多分支的执行流程

### 脚本应用示例

使用多分支编写猜数脚本,RANDOM为随机数

#!/bin/bash

x=$[RANDOM%10]

read -p "请输入一个数字(0-9)" t

if [ $x -eq $t ] ;then

echo "猜对了"

exit

elif [ $x -lt $t ];then

echo "猜大了"

else

echo "猜小了"

fi

# 总结和答疑

## 算术运算

### 问题现象

# # expr 1+2

1+2

# expr 10\*5

expr:syntax error

# echo $[3.14\*2]

3.14\*2:syntax error:invalid arithmetic operator (error token is “.14\*2”)

### 原因分析

分析故障

-报错信息:无正确运算结果

-报错信息:syntax error

-报错信息::invalid arithmetic operator

分析故障原因

-使用expr运算时,运算符号两边需要有空格

-expr进行乘法运算时,需要使用\屏蔽\*

-expr无法进行小数运算

## 条件判断

### 问题现象

# [1 -gt 2]

-bash:[1:command not found

# [1 -gt2]

# echo $?

0

# cat test.sh

#!/bin/bash

if [ a = a ];then

echo a

# bash test.sh

syntax error: unexpected end of file

# cat test.sh

#!/bin/bash

if [ $a -eq 2];then

echo a

fi

# bash test.sh

a.sh:line 2:[: -eq:unary operator expected]

### 原因分析

分析故障

-报错信息:command not found

-报错信息:判断结果有误

-报错信息:没有合理的结束符

-报错信息[:-eq:unary operator expected]

分析故障原因

- [的右边,]的左边,都需要空格

- 判断符合两边需要有空格

- if语句开始,需要配合使用fi作为判断结尾符号

- 无法对空值进行判断

### 

# 循环结构

## for循环

### 语法结构及特点

遍历/列表式循环

-根据变量的不同取值,重复执行命令序列

### for循环的执行流程

### 脚本应用示例

C语言风格的for循环

-通过变量控制,条件成立时循环

-步长可控次数

[root@svr7 ~]# vim cfor.sh

#!/bin/bash

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

do

echo $i

done

## while循环

### 语法结构及特点

条件式循环

-反复测试条件,只要成立就执行命令序列

### while循环的执行流程

### 脚本应用示例

使用while循环编写猜数脚本,猜对才停止

#!/bin/bash

x=$[RANDOM%100]

a=10

b=0

while [ $a == 10 ]

do

read -p "请输入一个数字(0-99)" t

let b++

if [ $x -eq $t ] ;then

echo "猜对了,一共猜了${b}次"

let a++

elif [ $x -lt $t ];then

echo "猜大了"

else

echo "猜小了"

fi

done

或者

#!/bin/bash

x=$[RANDOM%10]

b=0

while :

do

read -p "请输入一个数字(0-9):" t

if [ $x -eq $t ] ;then

let b++

echo "猜对了,一共猜了${b}次"

exit

elif [ $x -lt $t ];then

echo "猜大了"

else

echo "猜小了"

fi

let b++

done

使用循环打印 5\*5 的星星,脚本输出如下形状:(5 行,5 列),备注:echo -n 可以不换行输出。

#!/bin/bash

a=0

b=0

while [ $b -le 4 ]

do

let a++

if [ $a -le 5 ] ;then

echo -n "\*"

else

let b++

let a=0

echo ""

fi

done

九九乘法表

#!/bin/bash

a=0

b=0

while [ $b -le 9 ]

do

let a++

if [ $a -le $b ] ;then

echo -n "$b\*$a = "$[$a\*$b]" "

else

let b++

let a=0

echo ""

fi

done

# case语句

## case分支结构

### 语法结构及特点

检查变量的实际取值

-如果与预设的值相匹配,则执行对应的操作

### case分支的执行流程

### 脚本应用示例

1. 将nginx-1.12.2.tar.gz从真机发送到虚拟机

[root@room9pc01 ~]# scp '/root/桌面/nginx-1.12.2.tar.gz'

root@192.168.4.7:/root

1. 在虚拟机中释放该tar包

tar -xf nginx-1.12.2.tar.gz

1. 源码安装Nginx需要提前安装依赖包软件

yum -y install gcc openssl-devel.x86\_64 pcre-devel.x86\_64

1. 源码编译安装,依次执行以下命令

cd /root/nginx-1.12.2/ //进入刚刚释放的目录中

./configure

make

make install

1. 执行程序,开启程序

/usr/local/nginx/sbin/nginx //开启服务

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

报错提示:nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)

报错原因:80端口被占用(httpd)

解决办法:systemctl stop httpd

#!/bin/bash

case $1 in

sta)

/usr/local/nginx/sbin/nginx ;;

stop)

/usr/local/nginx/sbin/nginx -s stop ;;

re)

/usr/local/nginx/sbin/nginx -s stop

/usr/local/nginx/sbin/nginx ;;

stat)

netstat -antpu | grep -q nginx

[ $? -eq 0 ] && echo "nginx开了" || echo "nginx没开" ;;

\*)

echo "请输入sta | stop | re | stat"

esac

netstat -antpu 查看监听端口(缩写 ss -antpu)

netstat 命令可以查看系统中启动的端口信息,

常用选项如下:

-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

-p显示服务正在监听的端口信息(服务名称)(也就是程序名称)

# 函数及中断控制

## Shell函数

### 关于函数的认识

什么是函数?

-在Shell环境中,将一些需要重复使用的操作,定义为公共的语句块,即可称为函数

使用函数的好处?

-使脚本代码更简洁,增强易读性

-提高Shell脚本的执行效率

服务脚本中的函数应用

-适用于比较复杂的启动/终止控制操作

-方便在需要时多次调用

### 函数的定义与调用

调用已定义的函数

-格式:函数名

-先定义了才能调用,就好比脚本的”内部命令”

函数传值

-格式:函数名 值1 值2 .. ..

-传递的值作为函数的”位置参数”

### 函数应用示例

#!/bin/bash

abc(){ #定义函数

mkdir /root/xyz

cd /root/xyz

}

abc #在脚本上执行上面定义好的函数

#使用bash和source执行得到两种效果

echo -e "\033[32mABC\033[0m"

字体颜色30m-37m

背景颜色40m-47m

#!/bin/bash

cecho() {

echo -e "\033[$1m$2\033[0m"

}

cecho 31 123456

cecho 32 123456

cecho 33 123456

cecho 34 123456

cecho 35 123456

cecho 36 123456

cecho 37 123456

测试192.168.4.1~192.168.4.7

通了的用绿色显示

不通的用红色显示

#!/bin/bash

cecho() {

echo -e "\033[$1m$2\033[0m"

}

for a in `seq 1 7`

do

ping -c 4 -i 0.2 -W 1 192.168.4.$a &>/dev/null

if [ $? -eq 0 ];then

cecho 32 "192.168.4.$a通了"

else

cecho 31 "192.168.4.$a不通"

fi

done

Shell版fork炸弹

-仅13个字符: .(){.|.&};.

-递归死循环,可迅速耗尽系统资源

代码解析

.() //定义一个名为.的函数

{ //函数块开始

.|.& //在后台递归调用函数

} //函数块结束

; /与下一条执行语句的分隔

. //再次调用函数

## 中断及退出

### 中断/退出及相关指令

中断、继续、退出

类型 含义

break 跳出当前所在的循环体,执行循环体后的语句块

continue 跳过循环体内余下的语句,重新判断条件以决定是否需要执行下一次循环

exit 退出脚本,默认的返回值是0

### 脚本中断示例

#!/bin/bash

x=0

while :

do

read -p "请输入数字 (0是结束) " n

[ $n -eq 0 ] && break

x=$[x+n]

done

echo $x

### 脚本继续示例

编写脚本文件，找出1-20内6的倍数，并打印它的平方值

[root@svr7 ~]# vim test.sh

#!/bin/bash

for i in {1..20}

do

[ $[i%6] -ne 0 ] && continue

echo $[i\*i]

done

[root@svr7 ~]# chmod +x test.sh

[root@svr7 ~]# ./test.sh

### 脚本退出示例

# 总结和答疑

## for循环

### 问题现象

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

-bash:syntax error near unexpected token `(‘

# for i in 1 2

>echo $i

-bash:syntax error near unexpected token

### 原因分析

分析故障

-报错信息:-bash:syntax error near unexpected token `(‘

分析故障原因

-基本语法错误

-for循环的执行体,需要嵌入到do和done中间

## case语句

### 问题现象

# cat test.sh

#!/bin/bash

case $1 in

start)

echo start

stop)

echo stop

esac

# bash test.sh

test.sh:line 5:syntax error unexpected token `(‘

test.sh:line 5:`stop)’

### 原因分析

分析故障

-报错信息:-bash:syntax error near unexpected token `(‘

分析故障原因

-基本语法错误

-使用case语句时,命令序列需要使用;;作为结束符

## 启动脚本

### 问题现象

# cat test

#!/bin/bash

#chkconfig:- 120 110

case $1 in

start)

echo start;;

stop)

echo stop;;

esac

# chkconfig --add test

service a does not support chkconfig

### 原因分析

分析故障

-报错信息:service a does not support chkconfig

分析故障原因

-chkconfig启动和关闭的数字不能大于99

### 

# 字符串处理

## 子串截取

### 子串截取的方法

方法一,使用${}表达式

-格式:${var:起始位置:长度}

编号从0开始,可省略

可以使用echo${#变量}进行字符的统计

方法二,使用expr substr

-格式:expr substr “$var” 起始位置 长度

方法三,使用cut工具 编号均从1开始

-格式:echo $var | cut -b 起始位置-结束位置

### 子串截取示例

获取8位随机字符

#!/bin/bash

a=abcdefghijklnmopqrstuvwxyzABCDEFGHIJKLNMOPQRSTUVWXYZ1234567890

for i in {1..8}

do

x=$[RANDOM%62]

z=${a:x:1}

pass=$pass$z

done

echo $pass

创建100个用户,并设置6位随机密码,同时输出密码到文本中.

#!/bin/bash

a=abcdefghijklnmopqrstuvwxyzABCDEFGHIJKLNMOPQRSTUVWXYZ1234567890

for n in {1..100}

do

useradd nsd$n &>/dev/null

for i in {1..6}

do

x=$[RANDOM%62]

z=${a:x:1}

pass=$pass$z

done

echo $pass | passwd --stdin nsd$n &>/dev/null

echo "nsd$n $pass" >>/root/password

unset pass #或者pass= ,目的:清空变量

done

## 子串替换

### 子串替换的方法

只替换第1个匹配结果

-格式:${var/old/new}

替换全部匹配结果

-格式:${var//old/new}

### 子串替换示例

a=abcde

echo ${a/a/1} /将a替换成1

1bcde

## 按条件掐头去尾

### 字符串掐头

从左向右,最短匹配删除

-格式:${变量名#\*关键词}

从左向右,最长匹配删除 #用来删除头部,\*通配

-格式:${变量名##\*关键词}

a=`head -1 /etc/passwd`

echo ${变量名#删除的内容} //掐头

echo ${a#root} //将头部的root删除

echo ${a#\*:} //从左至右删,将第一个冒号以及冒号之前的内容删除

echo ${a##\*:} //从左至右删,将最后一个冒号以及冒号之前的内容删除

### 字符串去尾

从右向左,最短匹配删除

-格式:${变量名%关键词\*}

从右向左,最长匹配删除 %用来删除头部,\*通配

-格式:${变量名%%关键词\*}

echo ${变量名%删除的内容} //去尾

echo ${a%bash} //从右至左删,将尾部的bash删除

echo ${a%:\*} //从右至左删,将第一个冒号以及冒号之后的内容删除

echo ${a%%:\*} //从右至左删,将最后一个冒号以及冒号后前的内容删除

只留下/root

c=${a%:\*} ; c=${c##\*:}

### 脚本应用示例

利用字串删除功能,编写批量修改文件扩展名的脚本

#!/bin/bash

for i in $(ls /root/\*.$1) //查看所有以$1结尾的文件

do

u=${i%.$1} //将原文件的扩展名去掉

mv $i $u.$2 //都修改为.$2

done

## 变量初始值处理

### 初值的检测及设置

取值,${var:-word}

-若变量var已存在且非Null,则返回$var的值

-否则返回子串”word”,变量var值不变

设置变量初值(备用值)

echo ${abc:-123} //变量无值时使用初值

123

abc=666

echo ${abc:-123} //变量有值时使用本身的值

666

unset abc //取消变量

echo $abc

echo ${abc:-123} //无值后恢复使用初值

123

### 脚本应用示例

编写脚本,在修改用户密码时使用初值,如果用户定义了密码则使用定义的密码,否则则使用初值

#!/bin/bash

read -p "请输入用户名:" u

read -p "请输入修改的密码:" p

echo ${p:-123} | passwd --stdin $u

# 扩展的脚本技巧

## Shell-

### 关于变量的类型

Shell对变量类型的管理比较松散

-变量的值默认均视为文本

-用在数学运算中时,自动将其转换为整数

### 定义/赋值数组

方法一,整体赋值:

-格式:数组名=(值1 值2 .. .. 值n)

-示例:SVRS=(www ftp mail club)

方法二,为单个元素赋值:

-格式:数组名[下标]=值

-示例:FQDNS[0]=www.tarena.com 下标从0开始

FQDNS[1]=mail.tarena.com

FQDNS[2]=club.tarena.com

### 输出数组元素

获取单个数组元素

-格式:${数组名[下标]}

获取所有数组元素

-格式:${数组名[@]}

获取数组元素个数

-格式:${#数组名[@]}

获取连续的多个数组元素

-格式:${数组名[@]:起始下标:元素个数}

## expect预期交互

### expect简介

基于TCL编写的自动交互式程序

-可以用在Shell脚本中,为交互式过程自动输送预先准备的文本或指令,而无需人工干预

-触发的依据是预期会出现的特征提示文本

[root@svr7 ~]# yum -y install expect.x86\_64

### expect应用示例

expect << EOF

spawn ssh -o StrictHostKeyChecking=no 192.168.4.207 //spawn后面输入要执行的任务

expect “password:” { send “1\r” } //当看到password:时,输入1并回车

expect “#” { send “mkdir /opt/abcd\r” } //当看到#时,输入创建目录的命令回车

expect “#” { send “exit\r” } //最后一条指令不执行

EOF

/root/.ssh/known\_hosts //删除已经连接过的主机记录文件,会导致再次连接该主机时提示yes/no,使用ssh -o StrictHostKeyChecking=no 192.168.4.207 选项可以取消该提示

注意事项：

expect脚本的最后一行默认不执行

如果不希望ssh时出现yes/no的提示，远程时使用如下选项:

# ssh -o StrictHostKeyChecking=no server0

# 正则表达式

## 正则表达式概述

### 什么是正则表达式?

Regular Express ?

-使用”一串符号”来描述有共同属性的数据

### egrep过滤工具

文本处理顺序

-以行为单位,逐行进行处理

-默认只输出与表达式相匹配的文本行

基本用法

-格式1:egrep [选项] ‘正则表达式’ 文件.. ..

-格式2:前置命令 | egrep [选项] ‘正则表达式’

等同于grep -E,表示允许使用扩展的正则表达式

常用命令选项

-i:忽略字母大小写

-v:条件取反

-c:统计匹配的行数

-q:静默、无任何输出,一般用于检测

-n:显示除匹配结果所在的行号

--color:标红显示匹配子串

--看$?返回值,如果为0,说明由匹配,否则无匹配

## 基本元字符

### 行首尾及单字匹配

类型 含义 示例 说明

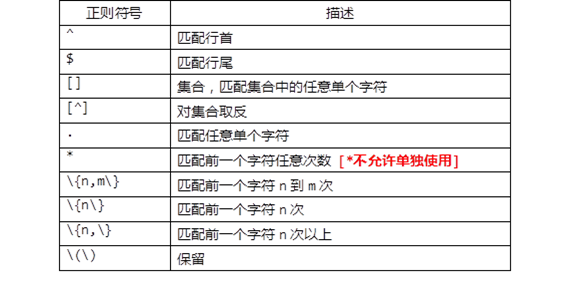
^ 匹配行首 ^abc 以abc开头的行

^# 以#号开头的行(比如注释行)

$ 匹配行尾 abc$ 以abc结尾的行

^$ 空行

. 单个字符 . 除换行符(\n)以外的任意单个字符



基本正则列表

### 未定匹配次数

类型 含义 示例 说明

+ 最少匹配一次 a+ 一个或多个连续的a

(abc)+ 一个或多个连续的abc

? 最多匹配一次 a? 0个或1个a

(abc)? 0个或1个abc

\* 匹配任意次数 a\* 0个或多个连续的a

(abc)\* 0个或多个连续的abc

.\* 任意长度的任意字符串

### {}限定次数

限定表达式的匹配次数

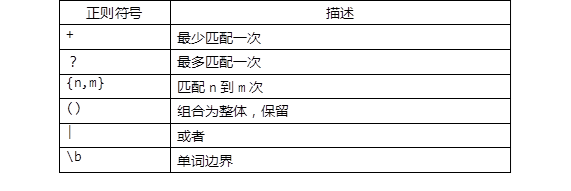
-{n}、{n,m}、{n,}

类型 含义 示例 说明

{n} 匹配n次 (ab){3} 匹配ababab

{n,m} 匹配n-m次 (ab){1,3} 匹配ab、ab、ababab

{n,} 匹配至少n次 (ab){2,} 匹配2个及以上连续的ab



扩展正则列表

## 其他元字符g

### []范围内单字匹配

匹配指定字符集合内的任何一个字符

-[]内加^可取反

示例 说明

[alc45\_?] 匹配a、l、c、4、5、\_、?

[a-z] 匹配任意小写字母

[A-Z] 匹配任意大写字母

[0-9] 匹配任意数字

[a-Z0-9] 匹配任意字母或数字

[^A-Z] 匹配包括非大写字母的行

^[^a-z] 匹配不以小写字母开头的行

### 整体及边界匹配

类型 含义 类型 说明

() 组合为整体 ab{1,3} 匹配ab、abb、abbb

(ab){1,3} 匹配ab、abab、ababab

| 或者 root|bin 匹配root、bin

\b 单词边界 \broot\b 匹配单词root,不匹配keroot、 rooty、brooty等字符串

\< 单词的开头 \<th 匹配以th开头的单词

\> 单词的结束 \<root\> 作用与\broot\b相同

\为转义符号,可以为一些普通字符赋予特殊含义,或者将一些特殊字符变为普通字符.

# sed基本用法

## sed命令解析

### sed工具概述

Stream EDitor,流式编辑器

-非交互,基于模式匹配过滤及修改文本

-逐行处理,并将结果输出到屏幕

-可实现对文本的输出、删除、替换、复制、剪切、导入、导出等各种操作

### 命令格式解析

主要用法:

-格式1:前置命令 | sed [选项] ‘编辑指令’

-格式2:sed [选项] ‘编辑指令’ 文件.. ..

编辑指令(定址符执行指令)

常见命令选项

-n:屏蔽默认输出(全部文本)

-i:直接修改文件内容

-r:启用扩展的正则表达式,若与其他选项一起使用,应作为首个选项

条件,可以是行号或/正则/

-行号可以使用单个数字表示单行

-或者3,5表示连续的多行

-省略条件,则默认逐行处理全部文本

-匹配正则时,需要使用//

### 基本的处理动作

常用动作指令

操作符 用途 指令示例

p 打印行 2,4p 输出第2、3、4行

2p;4p 输出第2行、4行

d 删除行 2,4d 删除第2、3、4行

s 字符串替换 s/old/new/ 将每行的第一个old替换为 new

s/old/new/3 将每行的第3个old替换为 new

s/old/new/g 将所有的old都替换为 new

替换操作的分隔”/”可改用其他字符,如#、&等,便于修改文件路径

## 常见处理操作示例

### 输出文本

示例 含义解析

sed -n ‘p’ a.txt 输出所有行,等同于cat a.txt

sed -n ‘4p’ a.txt 输出第4行

sed -n ‘4,7p’ a.txt 输出第4~7行

sed -n ‘4,+10p’ a.txt 输出第4行及其后的10行内容

sed -n ‘/^bin/p’ a.txt 输出以bin开头的行

sed -n ‘$=’ a.txt 输出文件的行数

### 删除文本

示例 含义解析

sed ‘3,5d’ a.txt 删除第3~5行

sed ‘/xml/d’ a.txt 删除所有包含xml的行

sed ‘/xml/!d’ a.txt 删除不包含xml的行,!符号表示取反

sed ‘/^install/d’ a.txt 删除以install开头的行

sed ‘$d’ a.txt 删除文件的最后一行

sed ‘/^$/d’ a.txt 删除所有空行

此例中只作输出,不更改原文件(若需要更改,应添加选项 -i)

### 替换文本

示例 含义解析

sed ‘s/xml/XML/’ a.txt 将每行中第一个xml替换为XML

sed ‘s/xml/XML/3’ a.txt 将每行中的第3个xml替换为XML

sed ‘s/xml/XML/g’ a.txt 将所有的xml都替换为XML

sed ‘s/xml//g’ a.txt 将所有的xml都删除(替换为空串)

sed ‘4,7s/^/#/’ a.txt 将第4~7行注释掉(行首加#号)

sed ‘s/^#an/an/’ a.txt 解除以#an开头的行的注释(去除行首的#号)

此例中只作输出,不更改原文件(若需要更改,应添加选项 -i)

[root@svr7 ~]# vim test.txt //新建素材

2017 2011 2018

2017 2017 2024

2017 2017 2017

# sed 's/2017/xxxx/' test.txt //将所有行的第1个2017替换成xxxx

# sed '1s/2017/xxxx/' test.txt //将第1行的第1个2017替换成xxxx

# sed '3s/2017/xxxx/g' test.txt //将第3行的所有2017替换成xxxx

# sed '3s/2017xxxx/2' test.txt //将第3行的第2个2017替换成xxxx

# sed '3s/2017//' test.txt //将第3行的2017替换成空

# sed -n ‘3s/2017/xxxx/p’ shu.txt //替换后显示被替换的行

()或\(\) 保留

使用保留功能,将文档每行第一个和最后一个字符互换

sed -r 's/^(.)(.\*)(.)$/\3\2\1/' nssw.txt //按文档结构复制,^(.)是\1 (.\*)是\2 (.)$是\3

sed 's/^\(.\)\(.\*\)\(.\)$/\3\2\1/' nssw.txt

sed -r 's/[0-9]//g' nssw.txt //将每行的所有的数字删除

sed -r 's/^ +//' nssw.txt //将每行的行首空格都删除

sed -r 's/([A-Z])/(\1)/g' nssw.txt //将每行的所有大写字母加上括号

## sed应用案例

### 修改IP配置

### 修改网站根目录

### 使用sed脚本

编写开启ftp访问并支持匿名上传文件的脚本

思路:

1. 增加pub目录写权限
2. vsftpd服务开启上传权限,使用sed修改配置文件

vim /etc/vsftpd/vsftpd.conf

#anon\_uoload\_enable=YES

1. 开启服务

#!/bin/bash

yum -y install vsftpd

chmod o=rwx /var/ftp/pub

sed -i "s/^#anon\_u/anon\_u/" /etc/vsftpd/vsftpd.conf

systemctl restart vsftpd

systemctl enable vsftpd

# sed文本块处理

## 文本块概述

### 文本块的应用场景

### sed的文本块处理动作

操作符 用途 指令示例

i 行前插入文本 2iYY 在第2行之前添加文本航”YY”

4,7iYY 在第4-7行的每一行前添加文本行

a 行后插入文本 2aYY 在第2行之后添加文本

/^XX/aYY 在以XX开头的行之后添加文本

c 替换当前行 2cYY 将第2行的内容修改为”YY”

## 文本块处理的应用

### 处理单行文本

sed ‘1i XXXX’ test.txt //将第1行前插入XXXX

sed ‘1a XXXX’ test.txt //将第1行后追加XXXX

sed ‘1c XXXX’ test.txt //将第1行替换为XXXX

### 处理多行文本

修改后的文本有多行时

-以换行符 \n 分隔

-或者,使用\强制换行

# sed高级应用

## 文件导入导出

### 导入导出操作

基本动作

-r 动作应结合-i选项才会存入,否则只输出

-w 动作以覆盖的方式另存为新文件

操作符 用途 指令示例

r 读取文件 3r b.txt 在第3行下方插入文件b.txt

4,7r b.txt 在第4-7每一行后插入文件b.txt

w 保存到文件 3w c.txt 将第3行另存为文件c.txt

4,7w c.txt 将第4-7行另存为文件c.txt

### sed复制剪切

模式空间

-存放当前处理的行,将处理结果输出

-若当前行不符合处理条件,则原样处理

-处理完当前行再读入下一行来处理

保持空间

-作用类似于”剪贴板”

-默认存放一个空行(换行符 \n)

基本动作

-H:模式空间 ---[追加]--->保持空间 复制

-h:模式空间 ---[覆盖]--->保持空间 复制

-G:保持空间 ---[追加]--->模式空间 粘贴

-g:保持空间 ---[覆盖]--->模式空间 粘贴

### 复制剪切操作示例

## sed流控制

### !取反操作

用途

-根据定址条件取反

示例

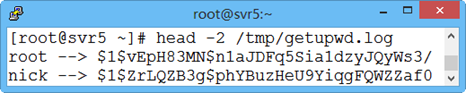
-列出不使用bash的用户账号记录

## sed综合脚本示例

### 任务需求

本案例要求编写脚本getupwd.sh，实现以下需求：

* 找到使用bash作登录Shell的本地用户
* 列出这些用户的shadow密码记录
* 按每行“用户名 --> 密码记录”保存到getupwd.log，如下图所示



### 思路讲解及分析

#!/bin/bash

for a in `sed -n '/bash$/p' /etc/passwd`

do

b=${a%%:\*}

c=`grep $b /etc/shadow`

d=${c#\*:};d=${d%%:\*}

echo "$b -- > $d"

done

`sed -n '/bash$/p' /etc/passwd`=`grep "bash$" /etc/passwd`

a=`sed -n '/bash$/p' /etc/passwd`;b=${a%%:\*}等价于A=$(sed -n ‘/bash$/s/:.\*//p’ /etc/passwd) //使用sed替换第一个冒号以及后面所有内容为空,仅使用bash的用户名

# 总结和答疑

## sed替换

### 问题现象

#sed “s/a/b” tesh.txt

sed:-e expression #1,char 5:unterminated `s’ command

### 原因分析

分析故障

-报错信息:unterminated `s’ command

分析故障原因

-s替换没找到合法的结束符

-s/旧/新/,使用s替换时,最后需要使用/结尾

### 

# awk基本用法

## awk命令解析

### 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

### 命令格式解析

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.. ..

多条语句可以分号分隔

print是最常用的指令

常用命令选项

-F:指定分隔符,可省略(默认空格或Tab位)

### awk内置变量

有特殊含义,可直接使用

变量 用途

FS 保存或设置字段分隔符,例如FS= ”:” ,与-F功能一样

$n 指定分隔的第n个字段,如$1、$3分别表示第1、第3列

$0 当前读入的整行文本内容

NF 记录当前处理行的字段个数(列数)

NR 记录当前已读入行的数量(行数)

awk ‘{print}’ a.txt //输出a.txt文档每行

awk ‘/root/{print}’ a.txt //输出a.txt文档有root的行

awk ‘{print $1}’ a.txt //输出a.txt文档每行的第1列,默认以空格为列的分隔

free -m | awk '/^M/{print $4}'

awk -F: '/^root/{print $6}' a.txt //使用冒号作为分隔符,输出以root开头的行的第6列

awk -F[:/] ‘{print $10}’ a.txt //使用冒号和/作为分隔符,输出m每行的第10列

ifconfig eth0 | awk '/RX p/{print $6,$7}'

awk -F: ‘{print “当前行有”NF”列”}’ u //输出常量与变量,常量用引号

使用awk获取服务器接收的网络流量

要求在屏幕输出:当前主机接收的流量是xxx

RX 接收的流量 TX 发送的流量

ifconfig eth0 | awk -F[\(\)] '/RX p/{print "当前主机接收的流量是"$2}'

要求在屏幕输出:根分区的容量是xxx

df -h | awk '/\/$/{print "根分区的容量是"$2}'

awk '/Failed/{print $11}' /var/log/secure //通过安全日志文件使用awk筛选访问失败的ip地址

### awk过滤的时机

在所有行前处理,BEGIN{} 可单独使用,也可以同时一起使用

-读入第一行文本之前执行

-一般用来初始化操作

逐行处理,{} 可单独使用,也可以同时一起使用

-逐行读入文本执行相应的处理

-是最常见的编辑指令块

在所有行后处理,END{} 可单独使用,也可以同时一起使用

-处理完最后一行文本之后执行

-一般用来输出处理结果

BEGIN{}行前处理,读取文件内容前执行,指令执行1次

{}逐行处理,读取文件过程中执行,指令执行n次

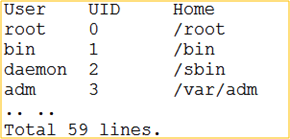
END{}行后处理,读取文件结束后执行,指令执行1次

awk流程化输出 BEGIN{指令} //逐行任务之前的任务

END{指令} //逐行任务之后的任务

awk 'BEGIN{x=0}/nologin$/{x++}END{print x}' /etc/passwd //在逐行任务之前先定义变量x,然后执行逐行任务,每看到一个以nologin结尾的行就把x加1,逐行任务之后输出x的值,统计多少账户用了nologin

要求: 格式化输出passwd文件内容时，要求第一行为列表标题，中间打印用户的名称、UID、家目录信息，最后一行提示一共已处理文本的总行数



awk -F: 'BEGIN{print "User\t\tUID\t\tHome"}{print $1"\t\t"$3"\t\t"$6}END{print "Total "NR" lines."}' /etc/passwd //按一定格式输出用户名、UID、家目录,注意\t代表tab键,属于常量,需要在双引号内

## awk处理条件

### 处理条件概述

条件的表现形式?

-正则表达式

-数值/字符串比较

-逻辑比较

-运算符

### 条件设置示例

正则表达式

-/正则表达式/

-~匹配、!~不匹配

数值比较

-==等于、!=不等于

->大于、>=大于或等于

-<小于、<=小于或等于

awk -F: '/^(root|adm)/{print $1,$3}' /etc/passwd //查找以root或者adm开头的行,并输出第1列,第3列

awk -F: '$1~/root/' /etc/passwd //查找第一列是root的行,并输出整行,输出整行时可以省略{print}

awk -F: '$7!~/nologin$/{print $1,$7}' /etc/passwd //查找第7列不是nologin的行,并输出第1列,第7列

### 多个条件的组合

逻辑比较测试

-&&逻辑与:期望多个条件都成立

-||逻辑或:只要有一个条件成立即满足要求

awk -F: 'NR==3{print}' /etc/passwd //输出第3行

awk -F: '$3>=1000{print $1,$3}' /etc/passwd //找到第3列大于等于1000的行,并输出第1列,第3列

awk -F: '$3<10{print $1,$3}' /etc/passwd //找到第3列小于10的行,并输出第1列,第3列

awk -F: '$1=="root"' /etc/passwd //输出第1列是root的行

awk -F: '$3>10 && $3<20' /etc/passwd //输出第3列大于10并且小于20的行

awk -F: '$3>1000 || $3<10' /etc/passwd //输出第3列大于100或者第三列小于10的行

编写一个脚本:

使用awk找出passwd文档中使用bash的账户,对应shadow文件以用户名 --> 密码的格式输出

/etc/passwd

/etc/shadow

#!/bin/bash

u=`awk -F: ‘/bash$/{print $1}’ /etc/passwd`

for i in $u

do

grep ^$i: /etc/shadow | awk -F: ‘{print $1,”-->”,$2}’

done

### 变量的运算

运算符

- +、-、\*、/、%

- ++、--、+=、-=、\*=、/=

## awk综合脚本示例

### 任务需求

### 思路讲解及分析

# awk高级应用

## awk流程控制

### 分支结构

单分支

-if(条件){编辑指令}

双分支

-if(条件){编辑指令1}else{编辑指令2}

多分支

-if(条件){编辑指令1}else if(条件){编辑指令2}.. ..

else{编辑指令N}

### 循环结构

### 其他控制语句

## awk数组

### 数组的定义及使用

定义数组

-格式:数组名[下标]=元素值

调用数组

-格式:数组名[下标]

遍历数组

-用法:for(变量 in 数组名){print 数组名[变量]}

### awk数组的经典应用

## awk案例分析

### 任务需求

### 思路讲解及分析

利用awk提取客户机IP地址、计算访问次数

-以$1做下标,定义数组ip

-最后利用for循环输出数组下标、对应数组元素的值

使用awk数组+for循环实现文档的复杂过滤

先准备素材文档,内容为:

192.168.0.1

192.168.0.2

192.168.0.2

192.168.0.3

192.168.0.3

192.168.0.3

192.168.0.3

然后使用awk获取该文档中所有出现的ip地址,以及出现的次数

awk ‘{ip[$1]++}END{for(i in ip){print i ,ip[i]}}’ a.txt

利用sort对提取结果排序

-n:按数字升序排列

-k:针对指定的列进行排序

-r:反向排序

使用awk检查httpd服务的访问者以及访问次数

1. 安装httpd yum install -y httpd
2. 开启访问 systemctl strat httpd
3. 关闭防火墙和selinux
4. 用真机或其他虚拟机访问该web网站,可以用不同主机多访问几次,访问记录被记录在http的日志中 /var/log/httpd/access\_log 每行就是一次访问记录,其中包含访问者的ip
5. awk ‘{ip[$1]++}END{for(i in ip){print i,ip[i]}}’ /var/log/httpd/access\_log

# 总结和答疑

## awk分隔符

### 问题现象

故障错误信息

#awk ‘{print $1,$3}’ /etc/passwd |head -2

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

bin:x:1:1:bin:/bin:/sbin/nologin

### 原因分析

分析故障

-报错信息:输出的信息不是希望的数据

分析故障原因

-默认awk分隔符为空格和tab键

-通过-F指定分隔符

## awk引号

### 问题现象

故障错误信息

#awk -F:”{print $1,$3}” /etc/passwd | head -2

awk:cmd.line:1{print,}

awk:cmd.line:1 ^syntax error

### 原因分析

分析故障

-报错信息:awk:cmd.line:1: ^syntax error

分析故障原因

-awk的条件和指令需要使用单引号

syntax error语法错误

# 项目实战:脚本

## 一键部署软件脚本

### 任务需求

一键安装Nginx

-一键源码安装Nginx软件

-脚本自动安装相关软件的依赖包

-脚本自动判断yum是否可用

### 思路讲解及分析

思路:

-使用数据过滤以及if语句测试判断yum源是否可用

-语句根据安装的软件,自动安装依赖包

-源码编译安装Nginx软件

#!/bin/bash

r=`yum repolist | awk '/^repolist/{print $2}' | sed 's/,//g'`

[ $r -eq 0 ] && exit

yum -y install gcc openssl-devel pcre-devel &>/dev/null

tar -xf '/root/桌面/nginx-1.12.2.tar.gz' -C /root

cd /root/nginx-1.12.2

./configure

make

make install

/usr/local/nginx/sbin/nginx //脚本执行完毕后,手工开启服务

netstat -antpu | grep nginx //检测服务状态,出现监听80端口则成功

## 启动脚本

### 任务需求

编写Nginx启动脚本

-脚本支持start、stop、restart、status

-脚本支持报错提示

-脚本具有判断是否已经开启或关闭的功能

### 思路讲解及分析

思路:

-使用case语句实现主体功能

-case语句中可以调用函数实现具体启动、关闭等操作

-脚本使用if测试判断程序是否已经开启

## 监控脚本

### 任务需求

编写脚本监控本机各项数据指标:

-CPU负载

-网卡流量

-内存剩余容量

free -m (以M为单位)

-磁盘剩余容量

-计算机账户数量

-当前登录账户数量

-计算机当前开启的进程数量

-本机已安装的软件包数量

### 思路讲解及分析

思路:

-uptime

-ifconfig

-free

-df

-cat /etc/passwd

-ps aux

-rpm -qa

#!/bin/bash

while :

do

clear

uptime | awk -F: '{print "CPU的负载是"$5}'

ifconfig eth0 | awk -F[\(\)] '/TX p/{print "eth0的发送流量是"$2}'

free -m | awk '/Mem/{print "内存剩余容量是"$4"M"}'

df -h / | awk '/\/$/{print "根分区剩余容量是"$4}'

z=`cat /etc/passwd | wc -l`

echo "计算机账户数量是$z个"

d=`who | wc -l`

echo "当前登录账户数量$d个"

j=`ps aux | wc -l`

echo "当前开启进程数量是$j个"

b=`rpm -qa | wc -l`

echo "已安装的软件包数量是$b个"

sleep 2

done

## 安全检测脚本

### 任务需求

防止远程ssh暴力破解密码

-检测ssh登录日志,如果远程登录账号名错误3次,则屏蔽远程主机的IP

-检测ssh登录日志,如果远程登录密码错误3次,则屏蔽远程主机的IP

### 思路讲解及分析

思路:

-ssh登录日志为/var/log/secure

-分析日志文件格式

-找出用户名以及密码错误的规律,并提取有效数据

-对有效数据进行汇总统计,实现黑名单过滤功能

#!/bin/bash

ip=`awk '/Failed/{ip[$11]++}END{for(i in ip){print i","ip[i]}}' /var/log/secure` //找到所有访问本机失败的ip,与失败的次数

for a in $ip

do

i=`echo $a | awk -F, '{print $1}'` //定义变量i,内容是访问失败的ip

s=`echo $a | awk -F, '{print $2}'` //定义变量s,内容是访问失败的访问次数

[ $s -ge 3 ]&&firewall-cmd --zone=block --add-source=$i //将失败次数大于3的ip扔到防火墙的block区域

done

## 进度显示脚本

### 任务需求

实现带进程显示的复制脚本

-默认Linux的cp命令不具有进度显示

-我们需要自己编写脚本实现进度显示

-可以使用进度条的方式,或者显示百分比的方式

### 思路讲解及分析

思路:

-使用#或者.作为进度条

-无限死循环显示进度,并设置为后台运行

-当任务结束后kill杀死后台运行的进度条显示

真实机使用 virt-manager 给任意一台虚拟机添加两块 10G 磁盘,在该虚拟机中编写脚本,自动给每块磁盘分2 个分区(总共 4 个分区),脚本自动对 4 个分区格式化,并分别挂载到任意目录。

#!/bin/bash

fenqu(){

fdisk $1 << EOF

n

+5G

n

w

EOF

partprobe

/sbin/mkfs.ext4 "$1"1 && /bin/mkdir -p /data"$1"1 && /bin/mount "$1"1 /data"$1"1

echo "$1"1 /data"$1" ext4 defaults 0 0 >> /etc/fstab

/sbin/mkfs.ext4 "$1"2 && /bin/mkdir -p /data"$1"2 && /bin/mount "$1"2 /data"$1"2

echo "$1"2 /data"$1" ext4 defaults 0 0 >> /etc/fstab

}

fenqu /dev/vdb

fenqu /dev/vdc

EOF

EOF是END Of File的缩写,表示自定义终止符.既然自定义,那么EOF就不是固定的,可以随意设置别名,在linux按ctrl-d就代表EOF.

EOF一般会配合cat能够多行文本输出.

其用法如下:

<<EOF //开始

....

EOF //结束

还可以自定义，比如自定义：

<<BBB //开始

....

BBB //结束

通过cat配合重定向能够生成文件并追加操作,在它之前先熟悉几个特殊符号:

< :输入重定向

> :输出重定向

>> :输出重定向,进行追加,不会覆盖之前内容

<< :标准输入来自命令行的一对分隔号的中间内容.

# Nginx安装与升级

## Nginx安装

### Web服务器对比

Unix和Linux平台下

-Apache、Nginx、Tengine、Lighttpd

-Tomcat、IBM WebSphere、Jboss

Windows平台下

-微软公司的IIS(Internet Information Server)

web服务器

业务展现的形式都是web

电商,游戏,交易,论坛,邮箱......

app

web服务器

硬件+软件

文件和目录权限:用户

### Nginx简介

Nginx(“engine x”)

-是俄罗斯人编写的十分轻量级的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

-官方网站:http://nginx.org/

Nginx配置文件及目录

-/usr/local/nginx/ //安装目录

-/conf/nginx.conf //主配置文件

-html //网页目录

-logs //日志文件

-sbin/nginx //启动脚本

[root@proxy ~]# yum -y install gcc pcre-devel openssl-devel //安装依赖包

[root@proxy ~]# useradd -s /sbin/nologin nginx

[root@proxy ~]# tar -xf nginx-1.10.3.tar.gz

[root@proxy ~]# cd nginx-1.10.3

[root@proxy nginx-1.10.3]# ./configure \

> --prefix=/usr/local/nginx \ //指定安装路径

> --user=nginx \ //指定用户

> --group=nginx \ //指定组

> --with-http\_ssl\_module //开启SSL加密功能

[root@proxy nginx-1.10.3]# make && make install //编译并安装

### Nginx进程管理

启动Nginx服务

[root@localhost nginx]# /usr/local/nginx/sbin/nginx

常用选项

-V:查看编译参数

-c:指定配置文件,启动服务

## Nginx升级

老版本有问题------>新版本

老版本功能少------>新版本

在不停止服务的情况下升级软件版本

[root@localhost ~ ]#tar -zxvf nginx-1.12.2.tar.gz

[root@localhost ~ ]# cd nginx-1.12.2

[root@localhost nginx-1.12.2 ]# ./configure \

> --prefix=/usr/local/nginx \

> --user=nginx \

> --group=nginx \

> --with-http\_ssl\_module

[root@localhost nginx-1.12.2 ]# make

### 平滑升级Nginx

[root@localhost ~]# cd /usr/local/nginx/sbin/

[root@localhost sbin]# mv nginx nginxold //改名旧脚本

[root@localhost ~ ]#cd /root/lnmp\_soft/nginx-1.12.2/objs/

[root@localhost objs]# cp nginx /usr/local/nginx/sbin/ //拷贝新脚本

[root@localhost nginx-1.12.2 ]# make upgrade //升级软件

[root@localhost sbin]# ./nginx -v //查看软件版本

nginx:nginx version:nginx/1.12.2

# Nginx服务器

## Nginx配置解析

### 配置文件结构

全局配置

http{

... ...

server{

... ...

location / {

... ...

}

}

}

全局配置（用户名,日志,进程)

http{

server{

listen 80;

server\_name localhost;

root html;

}

server{

listen 80;

server\_name www.xyz.com;

root www;

}

}

### 全局选项

user nginx; //进程所有者

worker\_processes 1; //启动进程数量

error\_log /var/log/nginx/error.log; //日志文件

pid /var/run/nginx.pid; //PID文件

events {

worker\_connections 1024; //单个进程最大并发量

}

### 配置容器

http{

......

server { //定义虚拟主机

listen 80;

server\_name localhost;

location / { //发布目录

root html;

index index.html index.htm;

}

}

}

### 用户认证

location /{

root html;

index index.html;

auth\_basic “anythings”; //认证提示符

auth\_basic\_user\_file “/usr/local/nginx/pass”; //认证密码文件

}

[root@localhost ~]# yum -y install httpd-tools

[root@localhost ~]# htpasswd -c /usr/local/nginx/pass admin //创建认证用户

htpasswd /usr/local/nginx/pass jerry //追加用户.不使用-c选项

# Nginx虚拟主机

## 三种模式虚拟主机

门户网站,电商网站,论坛,企业邮箱

### 基于域名的虚拟主机

server {

listen 80;

server\_name web1.plj.com; //域名1

.. ..

}

server {

listen 80;

server\_name web2.plj.com; //域名2

.. ..

}

}

server {

listen 80; //端口

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

charset utf-8; //使网站支持中文

location / {

root web; //指定网站根路径

index index.html index.htm;

}

}

### 基于端口的虚拟主机

server {

listen 8080; //端口

server\_name web1.plj.com; //域名

.. ..

}

server {

listen 8000;

server\_name web1.plj.com

.. ..

}

### 基于IP的虚拟主机

server {

listen 192.168.0.1:80; //端口

server\_name web1.plj.com; //域名

.. ..

}

server {

listen 192.168.0.2:80;

server\_name web1.plj.com

.. ..

}

# HTTPS加密网站

## 密钥

人:自然语言 机器:二进制

编码:

ASCII

a 010101

b 010111

c 011100

GB2312

apple+公司

大编码[utf-8]万国编码

A--->

### 对称密钥

AES

DES

应用案例:RAR、ZIP压缩加密(单机加密)

111---111

### 非对称密钥

RSA

DSA

应用案例:网络加密(https、ssh)

123---sjdknhfklandkonfdas

### Hash值

MD5

SHA256

SHA512

应用案例:数据完整性校验、文件秒传

## SSL虚拟主机

### 生成密钥

SSL加密网站的核心技术是非对称生成密钥

-公钥、私钥、证书

# 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

### SSL加密网站

server { //定义虚拟主机

listen 443 ssl; //开启SSL

ssl\_certificate cert .pem; //指定证书文件

ssl\_certificate\_key cert .key; //指定私钥文件

ssl\_session\_timeout 5m;

ssl\_protocols SSLv2 SSLv3 TLSv1;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

location / {

root html;

index index.html index.htm;

}

}

# 总结和答疑

## 软件依赖

### 问题现象

故障错误信息

[root@svr7 nginx-1.10.3]# ./configure

./configure:error:the HTTPrewrite module requires the PCRE library.

### 原因分析

分析故障

报错信息:缺少PCRE库

分析故障原因

-安装PCRE库(pcre-devel)

-禁用nginx地址重写功能(--without-http\_rewrite\_module)

## 配置文件错误

### 问题现象

故障错误信息

[root@svr7 ~]# nginx -t

directive “worker\_processes” is not terminated by “;”

“server” directive is not allowed here in

unexpected end of file,expecting “}”

the “ssl” parameter requires ngx\_http\_ssl\_module

unknown directive “server-name”

### 原因分析

分析故障信息

-directive “worker\_processes” is not terminated by “;”

-“server” directive is not allowed here in

-unexpected end of file,expecting “}”

-the “ssl” parameter requires ngx\_http\_ssl\_module

-unknown directive “server-name”

分析故障原因

-缺少;符号

-关键词server的位置不对

-{}不匹配

-实现加密网站,但编译nginx时没有指定ssl模块

-未知的指令(server-name)

### 

# 部署LNMP

## LNMP基础知识

### 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)

动态

用户-------------------->访问服务器[doc,jpg,mp4,mp3,html]

静态-------------------->服务器[a.php,a.py,a.prel,a.jsp]

## LNMP安装

### 安装Nginx

### 安装MairaDB

### 安装PHP

### 安装PHP扩展

现代化软件都采用模块化设计(100个功能)独立模块1G

[root@localhost ~ ]# yum -y install php-mysql

//给PHP安装扩展库文件,使得PHP可以连接MariaDB数据库

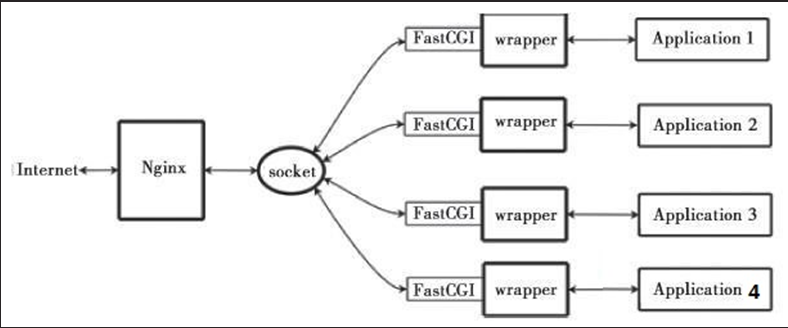
### 启动服务

PHP服务 端口9000

# Nginx+FastCGI

## FastCGI

### FastCGI工作原理



工作流程:

1. Web Server启动时载入FastCGI进程管理器
2. FastCGI进程管理器初始化,启动多个解释器进程
3. 当客户端请求到达Web Server时,FastCGI进程管理器选择并连接到一个解释器
4. FastCGI子进程完成处理后返回结果,将标准输出和错误信息从同一连接返回Web Server

### FastCGI简介

FastCGI技术目前支持语言有PHP、C/C++、Java、Perl、Python、Ruby等

### 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至25M内存,将整个数字乘以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)开64个php-cgi进程消耗1280M内存(20M\*64)

### 配置FastCGI

[root@localhost ~]# vim /etc/php-fpm.d/www.conf

[www]

listen = 127.0.0.1:9000

listen.allowed\_clients = 127.0.0.1

user = apache

group = apache

pm = dynamic

pm.max\_children = 50

pm.start\_servers = 5

pm.min\_spare\_servers = 5

pm.max\_spare\_servers = 35

配置文件

如果用户访问的是静态数据

那么,nginx直接找到文件,返回给用户

如果用户访问的是脚本

那么,nginx找到文件,转发给9000,php执行代码

程序,进程,线程

下载到硬盘中安装到硬盘的是程序

双击程序--->内存(进程)多进程

程序(硬盘)

进程(内存)

线程

程序---进程(20M)

线程(共享)

线程(共享)

线程(共享)

修改nginx,实现:

如果是静态,则nginx直接返回给用户

如果是php,则转发给9000端口,PHP执行,返回location匹配用户的地址栏

http{

server {

listen 80;

server\_name www.baidu.com;

location / {

allow all;

}

location /test {

deny all;

}

location /abc {

allow 1.1.1.1;

deny all;

}

}

firefox 192.168.4.5

## Nginx

### 编辑nginx.conf

[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf

location / {

root html;

index index.php index.html index.htm;

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000; //php-fpm的IP与端口

fastcgi\_index index.php;

include fastcgi.conf; //加载Fast-CGI参数文件

}

[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf

location匹配用户的地址栏(域名后面开始),支持正则表达式

http{

server {

listen 80;

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

location / { 找到页面,直接返回 }

location ~ \.php$ { 找到页面,转发给9000 }

}

}

firefox http://www.baidu.com

### 测试

1. 安装LNMP,启动服务
2. 修改nginx配置(nginx------>PHP)

动静分离

1. 通过编写PHP代码,连接数据库(读写操作)

mysql.php[命令]

# Nginx高级技术

## 地址重写

### 基础知识

什么是地址重写

-获得一个来访的URL请求,然后改写成服务器可以处理的另一个URL的过程

地址栏被重写改写了

www.360buy.com------>www.jd.com

地址重写的好处

-缩短URL,隐藏实际路径提高安全性

-易于用户记忆和键入

-易于被搜索引擎收录

### rewrite语法

rewrite基本语句

-rewrite regex replacement flag

rewrite 旧地址 新地址 [选项]

旧地址可用正则表达式

-if(条件){...}

地址重写格式【总结】

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

### 应用案例

### 正则表达式

正则表达式匹配模式如下:

-区分大小写匹配:~

-不区分大小写匹配:~\*

-区分大小写不匹配:!~

-不区分大小写不匹配:!~\*

rewrite /(.\*) http://www.jd.com/$1;

### rewrite选项

rewrite语句

-rewrite regex replacement flag

-flag:break、last、redirect、permanent

-last:停止执行其他重写规则,根据URI继续搜索其他location,地址栏不改变

-break:停止执行其他的重写规则,完成本次请求

-redirect:302临时重定向,地址栏改变,爬虫不更新URI(标识)

-permanent:301永久重定向,地址栏改变,爬虫更新URI(标识)

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

#'$status $body\_bytes\_sent "$http\_referer" '

#'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

网站服务器,可以判断客户端的浏览器是什么

电脑:访问(宽)

手机:访问(窄)

# 总结和答疑

## 动态页面故障

### 问题现象

[root@svr7 nginx-1.8.0]#firefox <http://192.168.2.100/test.php>

访问动态页面后,无法查看页面内容,浏览器显示下载窗口

[root@svr7 nginx-1.8.0]#firefox <http://192.168.2.100/test.php>

访问动态页面后,显示空白页

### 原因分析

分析故障

-默认nginx无法解析PHP页面

-nginx对自己无法识别的页面,则直接交给用户下载

-访问动态页面时,nginx返回空白页

分析故障原因

-nginx没有配置fastcgi\_pass

-返回空白页时,请检查PHP页面文件的语法格式是否正确

### 

# Nginx调度器

## HTTP调度

### Nginx反向代理架构

nginx代理(web,代理,正则,4层代理新功能)

haproxy代理(快,功能全,正则支持差)

### Nginx反向代理语法格式

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

http {

.. ..

#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 { //定义源服务器组

server 192.168.2.100 weight=1 max\_fails=1 fail\_timeout=30;

server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=30;

server 192.168.2.101 down;

}

#weight设置服务器权重值，默认值为1

#max\_fails设置最大失败次数

#fail\_timeout设置失败超时时间，单位为秒

#down标记服务器已关机，不参与集群调度

.. ..

server {

listen 80;

server\_name localhost; //Web主机名

location / {

#通过proxy\_pass将用户的请求转发给webserver集群

proxy\_pass <http://webserver;> //调用服务组

}

}

### Nginx调度算法

nginx目前支持的调度算法

-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

nginx:相同客户端访问相同Web服务器

7层http代理 [dhcp,ftp,ssh,http,dns...]

4层tcp/udp代理

web1()

proxy

web2()

ip------------------->ip (23.35.11.87)hash%2 =0

ip------------------->ip (23.35.11.87)hash%2 =0

ip------------------->ip (1.1.1.1)hash%2 =1

ip------------------->ip (1.1.1.1)hash%2 =1

### 服务器组主机状态

状态类型

-down:表示当前server暂时不参与负载

-max\_fails:允许请求失败的次数(默认为1)

-fail\_timeout:max\_fails次失败后,暂停提供服务的时间

### 案例应用

upstream sergrp {

#ip\_hash;

#server 192.168.8.5:80 weight=2;

server 192.168.8.5:80 down;

server 192.168.8.4:80;

server 192.168.8.6:80 backup;

server 192.168.8.3:80 max\_fails=2 fail\_timeout=30;

}

## TCP/UDP调度

### 模块

ngx\_stream\_core\_module模块

使用--with-stream开启该模块

注意:

-nginx从1.9版本才开始支持该功能

killall nginx

rm -rf /usr/local/nginx

[root@proxy ~]# yum –y install gcc pcre-devel openssl-devel //安装依赖包

[root@proxy ~]# tar -xf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

[root@proxy nginx-1.12.2]# ./configure \

> --with-http\_ssl\_module //开启SSL加密功能

> --with-stream //开启4层反向代理功能

[root@proxy nginx-1.12.2]# make && make install //编译并安装

### 语法格式

stream {

upstream backend {

server 192.168.2.100:22; //后端SSH服务器的IP和端口

server 192.168.2.200:22;

}

server {

listen 12345; //Nginx监听的端口

proxy\_connect\_timeout 1s;

proxy\_timeout 3s;

proxy\_pass backend;

}

}http{

... ...

}

### 案例应用

# Nginx优化

## HTTP错误代码

### HTTP常见错误代码列表

返回码 描述

200 一切正常

400 请求语法错误

401 访问被拒绝(账户或密码错误)

403 资源不可用,通常由于服务器上文件或目录的权限设置导致

403 禁止访问:客户端的IP地址被拒绝

404 无法找到指定位置的资源(Not Found)

414 请求URI头部太长

500 服务器内部错误

502 服务器作为网关或者代理时,为了完成请求访问下一个服务器,但该服务器返回了非法的应答(Bad Gateway)



### Nginx如何返回错误页面

http {

fastcgi\_intercept\_errors on; //错误页面重定向

server {

error\_page 404 /40x.html; //自定义错误页面

location = /40x.html {

root html;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

}

## Nginx状态页面

### status模块

status

--with-http\_stub\_status\_module开启模块功能

可以查看Nginx连接数等信息

[[root@nginx-1.12.2]#](mailto:[root@nginx-1.12.2]#) ./configure --with-http\_stub\_status\_module ... ...

注意:这里省略了其他选项...

### status页面

修改配置文件nginx.conf

[root@~ ]# cat /usr/local/nginx/conf/nginx.conf

... ...

location /status {

stub\_status on;

}

... ...

浏览器访问状态页面

[root@ ~]# curl <http://192.168.4.5/status>

Active connections:1

server accepts handled requests

10 10 3

Reading:0 Writing:1 Waiting:0

### 状态信息

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

## 配置优化

### 常用压力测试工具

ab

ab -c 并发数 -n 总请求数 URL

其他常见压力测试软件(需要额外下载)

-http\_load、webbench、siege

### 全局配置优化

修改Nginx配置文件，增加并发量

调整进程数量

#user nobody;

worker\_processes 2; //与CPU核心数量一致

events {

worker\_connections 65535; //每个worker最大并发连接数

use epoll;

}

error\_log /var/log/nginx.error\_log info; //定义日志级别

#[debug | info | notice | warn | error | crit ]

### EVENT模块优化

优化Linux内核参数（最大文件数量）

[root@proxy ~]# ulimit -a //查看所有属性值

[root@proxy ~]# ulimit -Hn 100000 //设置硬限制（临时规则）

[root@proxy ~]# ulimit -Sn 100000 //设置软限制（临时规则）

[root@proxy ~]# vim /etc/security/limits.conf

.. ..

\* soft nofile 100000

\* hard nofile 100000

#该配置文件分4列，分别如下：

#用户或组 硬限制或软限制 需要限制的项目 限制的值

Linux内核

max\_clients=worker\_processes \* worker\_connecttions

注意修改系统ulimit限制/etc/security/limits.conf

events{

worker\_connections 10000; //每个worker最大并发连接数

use epoll;

}

### HTTP模块优化

优化前，使用脚本测试长头部请求是否能获得响应

[root@proxy ~]# cat lnmp\_soft/buffer.sh

#!/bin/bash

URL=http://192.168.4.5/index.html?

for i in {1..5000}

do

URL=${URL}v$i=$i

done

curl $URL //经过5000次循环后，生成一个长的URL地址栏

[root@proxy ~]# ./buffer.sh

.. ..

<center><h1>414 Request-URI Too Large</h1></center> //提示头部信息过大

修改Nginx配置文件，增加数据包头部缓存大小

http{

keepalive\_timeout 10; //保持连接的超时时间

对页面进行压缩处理

修改Nginx配置文件

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

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

//先根据client\_header\_buffer分配,如果不够,再根据large值分配

}

修改Nginx配置文件，定义对静态页面的缓存时间

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ { //~代表模糊匹配.\*代表不区分大小写

expires 30d; //定义客户端缓存时间为30天

}

}

[root@proxy ~]# cp /usr/share/backgrounds/day.jpg /usr/local/nginx/html

[root@proxy ~]# 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)

服务器内存缓存

如果需要处理大量静态文件,需要保持这些文件句柄为打开状态,避免后续再次打开

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

}

客户端浏览器缓存数据

http{

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

access\_log off;

expires 30d;

}

}

### 日志切割

日志文件越来越大怎么办?单个文件10G?如何切割？（非常常见的面试题）

-把旧的日志重命名

-kill USR1 PID(nginx的进程号)

kill:给某个进程发送信号

/usr/local/nginx/logs/nginx.pid文件中存放的是nginx的进程PID号。

怎么自动完成,周期性日志切割

-脚本与计划任务!

# 总结和答疑

## 调度器故障

### 问题现象

[root@svr7 nginx-1.8.0]#firefox <http://192.168.2.100>

访问调度器后失败,提示50X错误或An error occured

### 原因分析

分析故障

-查看nginx日志,看看本机失败还是连接后端主机失败

分析故障原因

-Connection refused,while connecting to upstream

-原因:提示连接后端服务器被拒绝

### 

# memcached原理

## 基础知识

### 传统Web架构的问题

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、数据库相应恶化、网站显示延迟等重大影响

### 数据存储位置对比

性能

-CPU缓存>内存>磁盘>数据库

价格

-CPU缓存>内存>磁盘>数据库

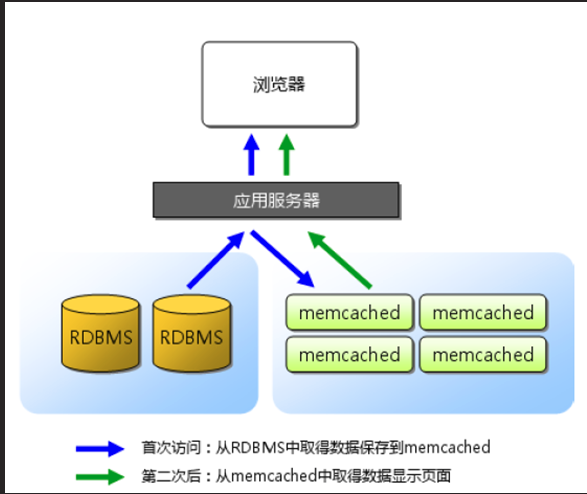
## memcached概述

### memcached简介

memcached是高性能的分布式缓存服务器

-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应用的相应速度

-官方网站:http://memcached.org/



### 内存管理机制

传统内存分配机制

-使用完通过分配的内存后回收内存,这种方式容易产生内存碎片并降低操作系统对内存的管理效率

Slab Allocation机制

-memcached使用这种内存管理机制,可以提高读写速度

# 部署memcached

## 安装软件

### 安装memcached

[root@localhost ~]# yum install memcached

[root@localhost ~]# rpm -qa memcached

memcached-1.4.15-10.el7\_3.1.x86\_64

### 启动memcached

[root@localhost ~]# cat /etc/sysconfig/memcached

PORT=”11211”

USER=”memcached”

MAXCONN=”1024”

CACHESIZE=”64”

[root@localhost memcached]# /usr/lib/systemd/system/memcached.service

[Service]

Type=simple

EnvironmentFile=-/etc/sysconfig/memcached

ExecStart=/usr/bin/memcached -u $USER -p $PORT -m $CACHESIZE -c

$MAXCONN $OPTIONS

[Install]

WantedBy=multi-user.target

[root@localhost memcached]# systemctl start memcached

[root@localhost memcached]# systemctl status memcached

选项

-p 指定memcached监听的端口号,默认11211

-u memcached程序运行时使用的用户身份必须是root用户

-m 指定使用本机的多少物理内存存数据,默认64M

-c memcached服务的最大连接数

-n chunk size的最小空间是多少,单位为字节

-f chunk size大小增长的倍数,默认1.25倍

-d 在后台启动

## 测试memcached

### 查看memcached的内部状态

[root@localhost ~]# telnet localhost 11211

Trying ::1...

Connected to localhost.

Escape character is ‘^]’.

stats

STAT pid 481

STAT uptime 16574

STAT time 1213687612

STAT version 1.2.5

STAT pointer\_size 32

### telnet测试memcached

参考memcached常用指令

[root@localhost bin]# telnet 127.0.0.1 11211

Trying 127.0.0.1...

......

set name 0 180 3 //定义变量

plj //输入变量的值

STORED

get name //获取变量的值

VALUE name 0 3 //输出结果

plj

END

quit //退出登录

memcached常用指令

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

## PHP+Memcached

### PHP+memcached介绍

部署LNMP

-使用php页面,测试对memcached的读写操作

-具体操作参考第二章节Nginx+FastCGI

PHP无法直接操作memcached

-需要安装memcached扩展

### 安装PHP的memcached扩展

[root@localhost ~]# yum install php-pecl-memcache

[root@localhost ~]# rpm -ql php-pecl-memcache

/etc/php.d/memcache.ini

/usr/lib64/php/modules/memcache.so

... ...

[root@localhost ~]# systemctl restart php-fpm

### 编写PHP页面

[root@localhost html]# cat /usr/local/nginx/html/test.php

<?php

$memcache=new Memcache; //创建memcache对象

$memcache->connect(‘localhost’,11211)or die (‘could not connect!!’);

$memcache->set(‘key’,’test’); //定义变量

$get\_values=$memcache->get(‘key’); //获取变量值

echo $get\_values;

?>

[root@localhost html]# elinks --dump http://localhost/mem.php

# Session共享

## 基本概念

### Session&cookies

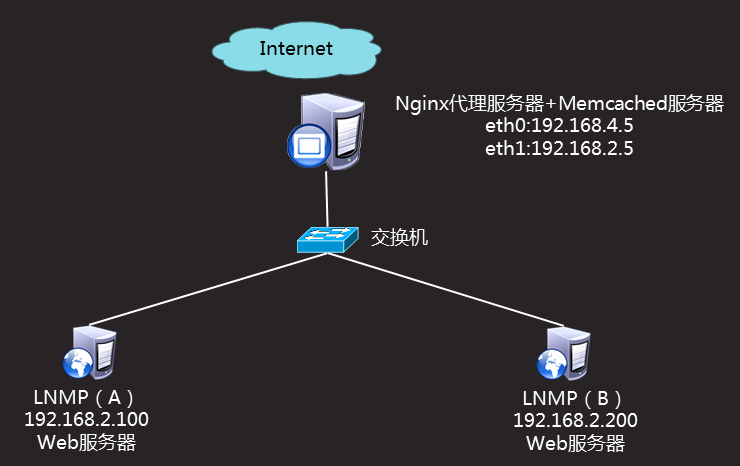
Session?

-存储在服务器端,保存用户名、密码等信息

Cookies?

-由服务器下发给客户端,保存在客户端的一个文件里.保存的内容主要包括:SessionID、账号名,过期时间,路径和域

### 案例拓扑



## 本地Session

### 部署Nginx调度器

安装Nginx软件

安装memcached软件

启动服务

修改配置文件

### 部署后端LNMP主机

Nginx

源码编译安装

MariaDB

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

PHP

[root@localhost ~]# yum -y install php php-mysql php-pecl-memcache

[root@localhost ~]# rpm -ivh php-fpm

启动服务

注意:以上安装LNMP和启动服务需要在后端两台主机都操作

在后台LNMP主机都操作

cd lnmp-soft/php\_scripts/session/

tar -xf php-memcached-demo.tar.gz

cp \* /usr/local/nginx/html

### 部署测试页面

注意:

以上安装LNMP和启动服务需要在后端两台主机都操作

在后端两台LNMP主机都操作

cd lnmp-soft/php\_scripts/session/

tar -xf php-memcached-demo.tar.gz

cd php-memcached-demo/

cp \* /usr/local/nginx/html

### 客户端访问

浏览器直接访问后端的LNMP主机2.100

firefox <http://192.168.2.100>

浏览器直接访问就断的LNMP主机2.200

firefox <http://192.168.2.200>

浏览器访问调度器,刷新页面后需要登录两次

firefox http://192.168.4.5

### 验证Session

登录后端两台LNMP主机分别查看

ls /var/lib/php/session

## 实现Session共享

### 修改PHP配置文件

[root@localhost session]# vim /etc/php-fpm.d/www.conf

修改前:

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

php\_value[session.save\_path]=/var/lib/php/session

修改后

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

php\_value[session.save\_path]=”tcp://192.168.2.5:11211”

### 重启服务

### 测试Session共享

浏览器访问条调度器,刷新页面后,登录账户会被记录在memcached服务器

刷新页面,调度器切换服务器后,账户信息还在

两台后端服务器使用的是同一个账户

[root@localhost ~]# firefox http://192.168.4.5

# 总结和答疑

## PHP连接memcached

### 问题现象

故障错误信息

[root@svr5 nginx-1.8.0]# firefox <http://192.168.2.100>

编写PHP代码,连接测试memcached数据库服务器,返回空白页或者直接出现下载窗口

### 原因分析

分析故障

-是否正确配置nginx

-是否正确配置PHP

-是否正确配置memcached

分析故障原因

-nginx没有配置fastcgi\_pass,将返回下载窗口

-php-fpm没有启动,或nginx没有设置正确的FPM端口

-PHP没安装memcache扩展,或没有在配置文件中加载该扩展

-Memcached服务没有启动,或者没有监听正确的IP和端口

## 无法共享session

### 问题现象

故障错误信息

[root@svr5 nginx-1.8.0]# firefox <http://192.168.4.5>

访问调度器后,刷新页面session(账户信息)依然有变化

### 原因分析

分析故障

-PHP是否添加了memcached扩展库

-PHP-fpm是否修改了www.comf配置

分析故障原因

-未安装php-pecl-memcache的话,无法连接memcached

-没有修改配置文件的话,默认php会在本地存储session

# Tomcat服务器

## 概述

### Java简介

Java

-java是一种跨平台的、面向对象的程序设计语言,Java技术具有卓越的通用性、高效性、平台移植性和安全性.

Java体系

-Java SE(标准版)

-Java EE(企业版)

-Java ME(移动版)

### JDK简介

JDK(Java Development Kit)是Sun针对Java开发者推出的Java语言的软件开发工具包

JDK是整个Java的核心

-包括了Java运行环境

-Java工具(如编译、排错、打包等工具)

-Java基础的类库

JRE(Java Runtime Environment,Java运行环境),JRE是JDK的子集

JRE包括

-Java虚拟机(jvm)

-Java核心类库和支持文件

-不包含开发工具(JDK)--编译器、调试器和其它工具

### Java Servlet

Servlet是java扩展Web服务器功能的组件规范

常见Servlet容器

-IBM websphere

-Oracle weblogic

-Apache tomcat

-RedHat Jboss

nginx+php,Python

Tomcat+Java

Oracle公司(SUN:java)

open-jdk

## 安装Tomcat

### 安装部署JDK

安装jdk1.8

[root@svr5 ~]# yum -y install java-1.8.0-openjdk

[root@svr5 ~]# yum -y install java-1.8.0-openjdk-headless

### 安装部署Tomcat

安装Tomcat

[root@svr5 ~]# tar -xzf apache-tomcat-8.0.30.tar.gz

[root@svr5 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat

[root@svr5 ~]# ls /usr/local/tomcat

bin/ //主程序目录

lib/ //库文件目录

logs/ //日志目录

temp/ //临时目录

work/ //自动编译目录jsp代码转换servlet

conf/ //配置文件目录

webapps/ //页面目录

启动Tomcat

[root@svr5 ~ ]# /usr/lcoal/tomcat/bin/startup.sh

在后台启动了一台基于Java的JVM虚拟机

### 测试服务器

生成测试页面

[root@svr5 ~ ]# # vim /usr/local/tomcat/webapps/ROOT/test.jsp

<html>

<body>

<center>

Now time is: <%=new java.util.Date()%>

</center>

</body>

</html>

客户端测试

-firefox http://localhost:8080

-firefox http://localhost:8080/test.jsp

# Tomcat应用案例

## Tomcat虚拟主机

### 主配置文件解析

server.xml配置文件框架

<Server>

<Service>

<Connector port=8080 /> //内容较少直接在同一行用/结尾

<Connector port=8009 />

<Engine name="Catalina" defaultHost="localhost">

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

</Host>

</Service> //内容较多在另外一行结尾

</Server>

### 基于域名的虚拟主机

修改server.xml文档,添加host虚拟主机

修改server.xml文档,给host添加context

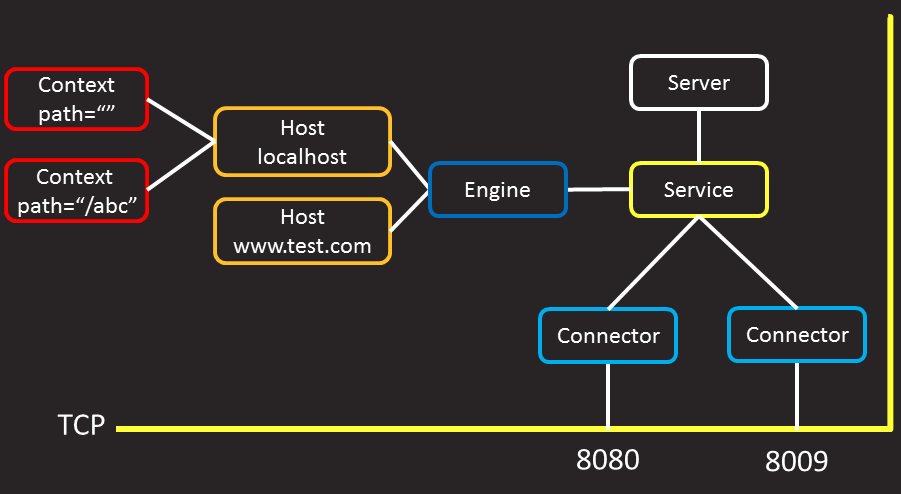
//appBase定义基础目录.基础目录下可以有很多项目,默认项目ROOT

//docBase定义首页路径,默认为ROOT

//path指定用户访问的URL,docBase指定页面存储的位置

验证:http://IP:8080/test/

### Tomcat工作原理



### SSL加密站点

生产私钥证书文件

[root@web1 ~]# keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore //提示输入密码为:123456

//-genkeypair 生成密钥对

//-alias tomcat 密钥别名

//-keyalg RSA 定义密钥算法为RSA算法

//-keystore 定义密钥文件存储在:/usr/local/tomcat/keystore

修改server.xml配置文件

客户端测试

### 日志

虚拟主机创建独立日志文件

## Tomcat集群

### Nginx反向代理tomcat

修改nginx配置

# Varnish代理服务器

## 代理服务器概述

### 基本概念

Proxy,代理人、经纪人、中介

代替用户处理事务

-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/反向)

-IP伪装、”翻墙”

代理服务器工作机制

-Cache池的复用

-线路转移

### Varnish服务器

Varnish一款高性能且开源的反向代理服务器.

Varnish具有性能高、速度更快、管理更方便等诸多优点.

## 部署Varnish

### 编译安装

[root@proxy ~]# yum -y install gcc readline-devel //安装软件依赖包

[root@proxy ~]# yum -y install ncurses-devel //安装软件依赖包

[root@proxy ~]# yum -y install pcre-devel //安装软件依赖包

[root@proxy ~]# yum -y install \

python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm //安装软件依赖包

[root@proxy ~]# useradd -s /sbin/nologin varnish //创建账户

[root@proxy ~]# tar -xf varnish-5.2.1.tar.gz

[root@proxy ~]# cd varnish-5.2.1

[root@proxy varnish-5.2.1]# ./configure

[root@proxy varnish-5.2.1]# make && make install

cpni lnmp\_soft/varnish-5.2.1/etc/example.vcl /usr/local/etc/default.vcl

### Varnish服务器

服务器软件

## 加速服务器

### 快速部署Varnish

修改配置文件

启动服务

[root@proxy ~]# varnishd -f /usr/local/etc/default.vcl

//varnishd命令的其他选项说明如下：

//varnishd –s malloc,128M 定义varnish使用内存作为缓存，空间为128M

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

### 缓存管理

清除缓存内容的命令格式:

varnishadm ban req.url ~.\*

### Varnish日志

varnishlog [-w file] //共享内存的日志

varnishncsa [-w file] //类Apache日志

### Varnish状态

varnishstat

-Client connections accepted:

表示客户端成功发送连接总数量

-Client requests received:

客户端发送HTTP请求的总数

-Cache hits:

!重要:命中缓存的次数

-Cache misses

!重要:缓存非命中的个数.

# 总结和答疑

## Java环境变量

### 问题现象

故障错误信息

[root@svr5 ~]# /usr/local/tomcat/bin/catalina.sh start

Neither the JAVA\_HOME nor the JRE\_HOME environment variable is defined

### 原因分析

分析故障信息

-Tomcat启动时无法找到Java

分析故障原因

-未安装jdk

-或者安装jdk后没有设置正确的环境变量

-使用命令java -version查看版本

## Tomcat配置文件错误

### 问题现象

故障错误信息

[root@svr5 nginx-1.8.0]# vim /usr/local/tomcat/conf/server.xml

该文件语法严格,容易出错

### 原因分析

分析故障

-大小写错误,如<host></Host>

-关键词不匹配,如<Host>无结束的</Host>

-位置错误,如将<Host>写到<Ｅngine＞的上面

-默认的localhost站点名称被修改

分析故障原因

-Tomcat严格区分大小写

-<Host>和</Host>为一对,不可缺少

-每个容器仅可以包含有效的信息,注意位置问题

-修改localhost站点名称后,访问服务时会找不到默认站点

## 端口冲突

### 问题现象

故障错误信息

[root@svr5 ~]# varnishd

bind():Address already in use

### 原因分析

分析故障

故障信息:bind():Address already in use

分析故障原因

-端口已经被其他服务所占用

-找到占用80端口的应用并关闭即可

### 

# 版本控制

## Subversion

### 什么是Subversion

Subversion 是一个自由/开源的版本控制系统

-在Subversion管理下,文件和目录开源超越时空

-Subversion允许你数据恢复到早期版本

-或者是检查数据修改的历史

-许多人将版本控制系统当作一种神奇的”时间机器”

-允许你和别人协作文档并跟踪所做的修改

### Subversion架构

客户端

-命令行

-图形

通信方式

-本地访问

-SVN服务器

-Web服务

## 基本概念

### 版本库

典型的客户/服务器系统

-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

Subversion特点

-记录每一次改变

### 版本模型

需要避免的问题

锁定-修改-解锁的方案

锁定-修改-解锁的问题

-锁定可能导致管理问题

A锁定文件后忘记解锁等问题

-锁定导致不必要的串行开发

A想修改一个文件的开始,B想修改一个文件的结尾

如果能进行正确的合并,则可以更轻松的工作,没必要轮流工作

拷贝-修改-合并的方案

Subversion控制系统使用拷贝-修改-合并模型

-每个客户连接版本库,并建立个人工作副本

-用户并行工作,修改自己的副本

-最终,合并版本!

-个别冲突问题,需要人为手动解决

流程

-Harry拷贝副本,Sally拷贝副本

-Harry修改A1,Sally修改A2

-Harry上传A1后,Sally上传A2,会提示Sally的文件已过期

-Sally更新文件(合并)后上传新的A3

# SVN基础

## 服务器设置

### 安装Subversion

安装Subversion

### 创建版本库

利用svnadmin命令可创建服务器版本库

-subversion版本库管理工具,帮助:svnadmin help

-用法:svnadmin 命令/版本库路径 [选项]

-命令:create 创建一个新的版本库

-subversion版本库管理工具,帮助:svnadmin help

-用法:svnadmin 命令 /版本库路径 [选项]

-命令:create 创建一个新的版本库

### 导入项目代码

使用svn命令将项目代码导入到版本库中

-import指令执行导入操作

- -m选项设置注释性的字符串

### 认证与授权

使用SVN内置的认证机制可以有效地增强客户端访问版本库的安全性

-当客户端访问版本库服务器时,服务器会根据版本库目录下的conf/svnserver.conf文件中定义的认证与授权策略实现权限的控制

使用SVN内置的认证机制可以有效地增强客户端访问版本库的安全性

[root@svr5 ~]# cat /var/svn/web\_project/conf/svnserve.conf

[general]

anon-access=none

//设置拒绝匿名账户访问,此处可以设置为none、read、write

auth-access=write

//经过认证的账户权限为可写权限

password-db=passwd

//账户名称与密码的存放文件名,该文件在conf目录下

authz-db=authz

//基于路径的访问控制文件名(可以对文件或目录设置权限)

在passwd文件中需要设置账户信息

在authz文件中需要设置访问控制权限

[root@svr5 ~]# cat /var/svn/web\_project/conf/passwd

[users]

harry=harryssecret //用户名为harry,密码为harryssecret

sally=sallysecret //用户名为sally,密码为sallyssecret

[root@svr5 ~]# cat /var/svn/web\_project/conf/authz

[groups]

admins=harry,sally //定义组,组成员有harry与sally

[/] //根路径设置权限,也可以设置其他路径

@admins=rw //admins组中的用户可读、可写权限

\*=r //其他所有的人只读

//权限列别:只读(‘r’)、读写(‘rw’)、无权限(‘’)

### svnserve服务

svnserve命令即可启动SVN服务进程

-d //以守护进程方式运行svnserve

--listen-port=port //监听的端口,默认端口号为3690

-r root //设置一个虚拟路径,默认客户端要指定绝对路径访问库

[root@svr5 ~]# svnserve -d -r /var/svn/project1 //共享单个仓库

或者

[root@svr5 ~]# svnserve -d //共享所有仓库

## 客户端操作

[root@web2 ~]# cd /tmp

[root@web2 ~]# svn --username harry --password 123456 \

co svn://192.168.2.100/ code //co=checkout

//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

Store password unencrypted (yes/no)? yes //提示是否保存密码

[root@web2 ~]# cd /tmp/code

[root@web2 code]# ls

[root@web2 code]# vim user.slice //挑选任意文件修改其内容

[root@web2 code]# svn ci -m "modify user" //将本地修改的数据同步到服务器 //ci=commit 上传

[root@web2 code]# svn update //将服务器上新的数据同步到本地

[root@web2 code]# svn info svn://192.168.2.100 //查看版本仓库基本信息

[root@web2 code]# svn log svn://192.168.2.100 //查看版本仓库的日志

[root@web2 code]# echo "test" > test.sh //本地新建一个文件

[root@web2 code]# svn ci -m "new file" //提交失败，该文件不被svn管理

[root@web2 code]# svn add test.sh //将文件或目录加入版本控制

[root@web2 code]# svn ci -m "new file" //再次提交，成功

[root@web2 code]# svn mkdir subdir //创建子目录

[root@web2 code]# svn rm timers.target //使用svn删除文件

[root@web2 code]# svn ci -m "xxx" //提交一次代码

[root@web2 code]# vim umount.target //任意修改本地的一个文件

[root@web2 code]# svn diff //查看所有文件的差异

[root@web2 code]# svn diff umount.target //仅查看某一个文件的差异

[root@web2 code]# svn cat svn://192.168.2.100/reboot.target //查看服务器文件的内容

[root@web2 code]# sed -i 'd' tmp.mount

//删除文件所有内容，但未提交

[root@web2 code]# svn revert tmp.mount

//还原tmp.mount文件

[root@web2 code]# rm -rf \*.target

//任意删除若干文件

[root@web2 code]# svn update

//还原

[root@web2 code]# sed -i '1a #test###' tuned.service

//修改本地副本中的代码文件

[root@web2 code]# svn ci -m "xxx"

//提交代码

[root@web2 code]# svn merge -r7:2 tuned.service

//将文件从版本7还原到版本2

### 访问方式

本地磁盘

SVN

Web

URL格式 含义

[file:///](file://) 直接访问本地磁盘上的版本库(客户端与服务器端在一台机器上)

http:// 配置Apache的WebDAV协议,通过网页访问版本库

https:// 与http://相似,但使用了SSL进行数据加密

svn:// 通过svnserve定义的协议访问版本库

svn+ssh:// 与svn://相似,但使用了SSH封装加密数据

### 客户端命令行工具

svn命令 [选项]

--password 密码

--username 用户名

--revision(-r) 指定要检查的特定版本

checkout命令(初始化检出)

-checkout URL [PATH]

-从服务器版本库中复制一份副本至本地

-URL定位版本库

-PATH为本地工作副本路径

commit命令(提交修改)

-在本地修改本地副本中的代码后,commit可以提交该修改,原子事务提交

update命令

-将服务器上其他人的修改的代码更新到本地

查看版本库信息

-info命令

查看版本仓库信息

-log命令

查看版本修改历史

add命令(本地版本库添加新文件)

-在本地版本库副本,添加新的文件

-注意,add不会自动提交版本库服务器

-需要使用commit命令提交服务器

del、rm命令(本地版本库删除文件)

-删除本地版本库副本文件

-注意,del不会自动提交版本库服务器

-需要使用commit命令提交服务器

mkdir命令(本地版本库创建目录)

-在本地版本库副本,添加新目录

diff命令(数据对比)

-对比本地副本与服务器数据

svn diff killall //对比单个文件

svn diff //对比所有文件

cat命令

-查看版本库数据的内容

-本地副本可以直接cat查看

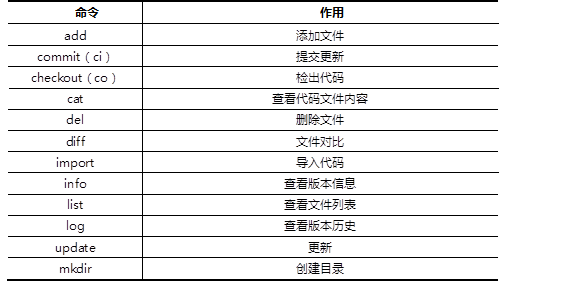
版本回滚

-revert命令

本地副本修改后,但未commit提交修改时回滚数据

-merge命令

本地副本修改commit提交后,使用该命令回滚



### Windows客户端软件

# 实战案例

## 多人协同案例

### 创建项目

使用svn客户端工具连接subversion服务器并测试多人协同工作以及如何手动解决冲突问题，账户名称分别为harry和tom，最后使用svnadmin dump指令对版本库进行备份工作。

### 初始化数据

[root@web1 ~]# cd /tmp

[root@web1 ~]# svn --username tom --password 123456 \

> co svn://192.168.2.100/ code

[root@web2 ~]# cd /tmp

[root@web2 ~]# svn --username harry --password 123456 \

> co svn://192.168.2.100/ code

[root@web1 ~]# cd code

[root@web2 ~]# cd code

### 修改副本

harry和tom修改不同的文件

[root@web1 mycode]# sed -i "3a ###tom modify#####" tmp.mount

[root@web1 mycode]# svn ci -m "has modified"

[root@web2 mycode]# sed -i "3a ###harry modify#####" umount.target

[root@web2 mycode]# svn ci -m "has modified"

[root@web2 mycode]# svn update

[root@web1 mycode]# svn update

harry和tom修改相同文件的不同行

[root@srv5 ~]# cd harry

[root@web1 mycode]# sed -i "3a ###tom modify#####" user.slice

[root@web1 mycode]# svn ci -m "modified"

[root@web2 mycode]# sed -i "6a ###harry 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 .

### 解决冲突

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(冲突) 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

## 备份与还原

### 备份版本库

使用dump指令备份

[root@web1 ~]# svnadmin dump /var/svn/project > project.bak //备份

### 还原版本库

使用load指令还原数据

[root@web1 ~]# svnadmin load /var/svn/project2 < project.bak //还原

## GitHub

### 简介

GitHub是一个面向开源及私有软件项目的托管平台,因为只支持git作为唯一的版本库格式进行托管,故名GitHub.

GitHub于2008年4月10日正式上线

### 使用GitHub

注册、上传代码

下载代码

登录网址(https://github.com/)

搜索项目

# RPM打包

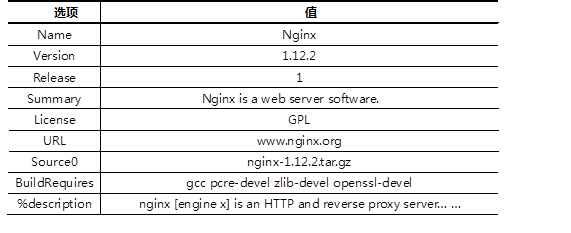
## 基本概念

### 应用场景

官方未提供RPM包

官方RPM无法自定义

大量源码包,希望提供统一的软件管理机制



## 软件打包

### 打包流程

准备源码软件

先把源码编译安装,如安装目录/usr/local/nginx

安装rpm-build

[root@web1 ~]# yum -y install rpm-build

生成rpmbuild目录结构

[root@web1 ~]# rpmbuild -ba nginx.spec //会报错，没有文件或目录

[root@web1 ~]# ls /root/rpmbuild //自动生成的目录结构

BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

编写编译配置文件

编译RPM包

### 安装rpm-build

编译打包RPM的命令工具

### 准备源码软件

将源码包复制到rpmbuild子目录

### 编写编译配置文件

新建SPEC文件

[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec

Name:hello //软件名称

Version: //软件版本

Release:1 //RPM版本

Summary: //描述

Group: //软件组

License: //协议

URL: //网址

Source0: //源码文件

BuildRoot: %(mktemp -ud %{\_tmppath}/%{name}-%{version}-%{release})

新建SPEC文件

[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec

... ...

BuildRequires: //编译时依赖包

Requires: //安装时依赖包

%description //详细描述

%prep //安装前准备,解压

%setup -q //系统使用setup自动解压,安静模式

%build //编译需要执行的命令

make

%configure //配置时需要执行的指令

make %{?\_smp\_mflags}

%install //安装时需要执行的指令

rm -rf %{buildroot}

make install DESTDIR=%{buildroot}

%post //安装结束后需要执行的脚本

新建SPEC文件

[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec

... ...

%clean //清理时需要执行的指令

rm -rf %{buildroot}

%files //定义打包文件列表

%defattr(-,root,root,-)

%doc

%changelog //软件修改历史

### 编译RPM包

使用spec文件编译RPM包

安装测试RPM包

# 总结和答疑

## 无法检出数据

### 问题现象

故障错误信息

[root@svr5 ~]# svn --username harry --password pass \

> checkout svn://127.0.0.1/harry

No repository found in ‘svn://127.0.0.1’

[root@svr5 ~]# svn --username harry --password pass \

> checkout svn://127.0.0.1/harry

No access allowed to this repository

[root@svr5 ~]# svn --username harry --password pass \

> checkout svn://127.0.0.1/harry

Authorization failed

### 原因分析

分析故障信息

-No repository found in ‘svn://127.0.0.1’

-No access allowed to this repository

-Authorization failed

分析故障原因

-无法找到代码库,启动服务器时没有-r,客户端连接需要指定绝对路径

-账户无权限访问代码库

-authz文件没有正确定义账户对项目的权限

-svn配置文件顶头不可以有空格

## 创建或删除文件

### 问题现象

故障错误信息

[root@svr5 ~]# rm httpd

[root@svr5 ~]# mkdir test;touch test.txt

[root@svr5 ~]# svn commit -m “aa”

### 原因分析

分析故障信息

-对版本库中的代码不可以直接使用命令删除或创建

分析故障原因

-svn add可以创建文件

-svn mkdir可以创建目录

-svn del可以删除文件

### 

# VPN服务器

## GRE VPN

### 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

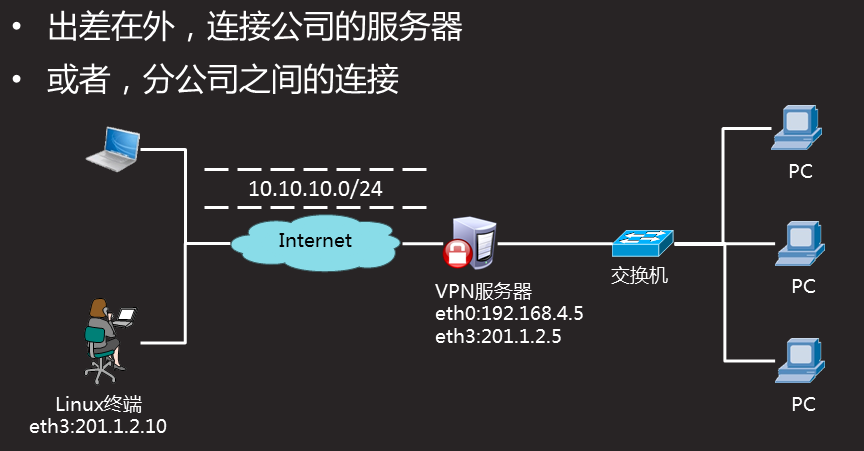
-连接完成后,各个地区的子公司可以像局域网一样通讯

-在企业网络中有广泛应用

-偶尔可以用于翻墙

-目前主流的VPN技术(GRE,PPTP,L2TP+IPSec,SSL)

### 拓扑图



### GRE模块

Linux内核模块

-ip\_gre

加载模块

-lsmod | grep ip\_gre //显示模块列表

-modprobe ip\_gre //加载模板

-modinfo ip\_gre //查看模块信息

缺点:缺少加密机制

### 创建VPN隧道

Client

创建隧道

[root@client ~]# ip tunnel add tun0 mode gre remote 201.1.2.5 local 201.1.2.10

//ip tunnel add创建隧道（隧道名称为tun0），ip tunnel help可以查看帮助

//mode设置隧道使用gre模式

//local后面跟本机的IP地址，remote后面是与其他主机建立隧道的对方IP地址

启用该隧道（类似与设置网卡up）

[root@client ~]# ip link show

[root@client ~]# ip link set tun0 up //设置UP

[root@client ~]# ip link show

为VPN配置隧道IP地址

[root@client ~]# ip addr add 10.10.10.10/24 peer 10.10.10.5/24 dev tun0

//为隧道tun0设置本地IP地址（10.10.10.10.10/24）

//隧道对面的主机IP的隧道IP为10.10.10.5/24

[root@client ~]# ip a s //查看IP地址

开启路由转发(软路由)

echo "1" > /proc/sys/net/ipv4/ip\_forward

或者

sysctl -w net.ipv4.ip\_forward=1

VPN服务器

Client

Proxy

### 测试连通性

缺点:基于Linux内核,排错困难,无加密,安全性较差

优点:支持的内容较多

## PPTP VPN

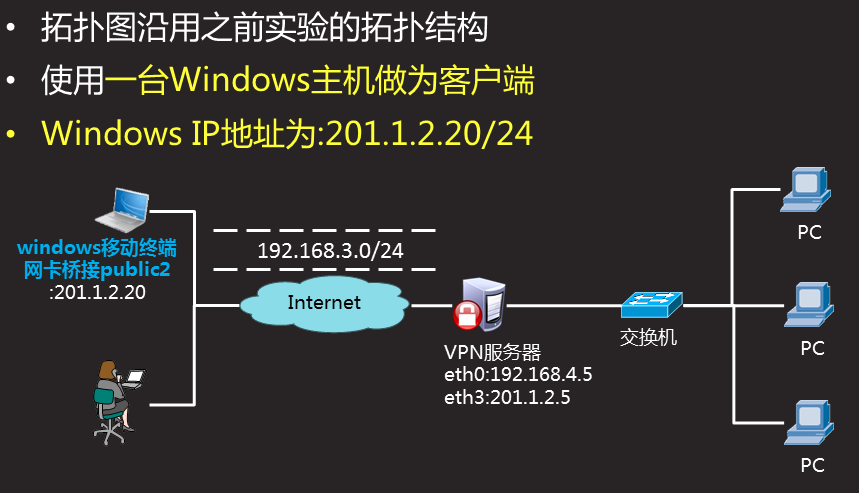
### 概述

PPTP(Point to Point Tunneling Protocol)

支持密码身份验证

支持MPPE(Microsoft Point-to-Point Encryption)加密

### 拓扑图



### 部署VPN服务器

安装软件

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

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

启动服务

[root@proxy ~]# systemctl start pptpd

[root@proxy ~]# systemctl enable pptpd

翻墙设置

iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -j SNAT --to-source 201.1.2.5

### Windows客户端

配置IP地址:201.1.2.20

建立VPN连接

## L2TP+IPSec VPN

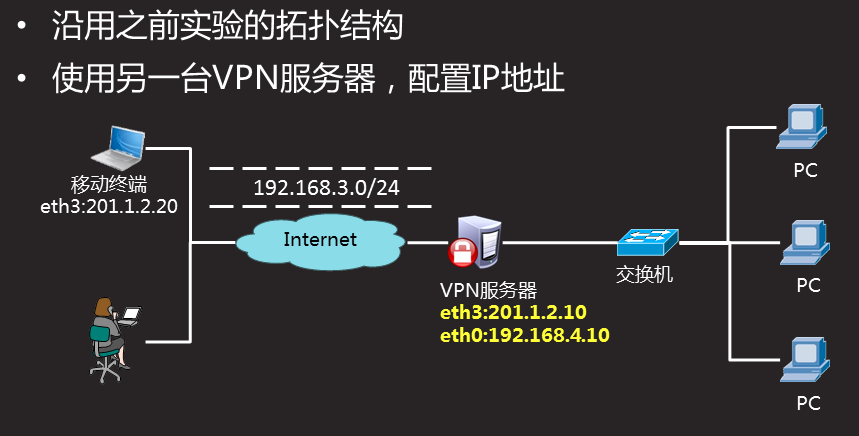
### 概述

Layer Two Tunneling Protocol (L2TP)

L2TP建立主机之间的VPN隧道,压缩、验证

IPSec提供数据加密、数据校验、访问控制的功能

### 拓扑图



### 部署L2TP+IPSec服务器

安装软件

创建IPSec加密配置文件

[root@client ~]# cat /etc/ipsec.conf //仅查看一下该主配置文件

.. ..

include /etc/ipsec.d/\*.conf //加载该目录下的所有配置文件

[root@client ~]# vim /etc/ipsec.d/myipsec.conf

//新建该文件，参考lnmp\_soft/vpn/myipsec.conf

conn IDC-PSK-NAT

rightsubnet=vhost:%priv //允许建立的VPN虚拟网络

also=IDC-PSK-noNAT

conn IDC-PSK-noNAT

authby=secret //加密认证

ike=3des-sha1;modp1024 //算法

phase2alg=aes256-sha1;modp2048 //算法

pfs=no

auto=add

keyingtries=3

rekey=no

ikelifetime=8h

keylife=3h

type=transport

left=201.1.2.10 //重要，服务器本机的外网IP

leftprotoport=17/1701

right=%any //允许任何客户端连接

rightprotoport=17/%any

新建IPSec预定义共享密钥

[root@vpn ~]# cat /etc/ipsec.secrets //仅查看,不要修改该文件

include /etc/ipsec.d/\*.secrets

[root@vpn ~]# vim /etc/ipsec.d/mypass.secrets //新建文件

201.1.2.10 %any: PSK ”randpass” //randpass为密钥

//201.1.2.10是VPN服务器的IP

启动IPSec服务

修改xl2tp配置文件

[root@client ~]# vim /etc/xl2tpd/xl2tpd.conf //修改主配置文件

[global]

.. ..

[lns default]

.. ..

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

local ip = 201.1.2.10 //VPN服务器的IP地址

[root@client ~]# vim /etc/ppp/options.xl2tpd //认证配置

require-mschap-v2 //添加一行，强制要求认证

#crtscts //注释或删除该行

#lock //注释或删除该行

root@client ~]# vim /etc/ppp/chap-secrets //修改密码文件

jacob \* 123456 \*

//账户名称 服务器标记 密码 客户端IP

启动XL2TP服务

翻墙

### Windows客户端

默认拨号会报错:789,需要做如下设置:

-单击”开始”,单击”运行”,键入”regedit”,然后单击”确定”

-找到下面的注册表子项,然后单击它:

-HKEY\_LOCAL\_MACHINE\

System\CurrentControlSet\Services\Rasman\Parameters

-在”编辑”菜单上,单击”新建”->”DWORD值”

-在”名称”框中,键入”ProhibitIpSec”

-在”数值数据”框中,键入”1”,然后单击”确定”

-退出注册表编辑器,然后重新启动计算机

# NTP时间同步

## 概述

### NTP协议概述

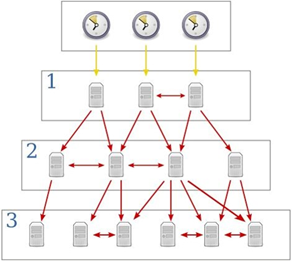
Network Time Protocol (网络地址协议)

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)



### 软件包

ntp和chrony

## NTP服务器

### 修改配置文件

主配置文件/etc/chrony.conf

[root@proxy ~]# cat /etc/chrony.conf

server 0.centos.pool.ntp.org iburst //server用户客户端指向上层NTP服务器

allow 192.168.4.0/24 //允许那个IP或网络访问NTP

deny 192.168.4.1 //拒绝那个IP或网络访问NTP

local stratum 10 //设置NTP服务器的层数量

### 部署服务器

启动服务器

防火墙设置

## NTP客户端

### 配置文件

主配置文件/etc/chrony.conf

[root@proxy ~]#cat /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

### 验证时间

客户端将时间修改错误

重启服务

验证时间是否同步

# PSSH远程工具

## 概述

### PSSH简介

PSSH提供了一套并发openssh工具

-PSSH provides parallel versions of the openssh tools

PSSH使用Python编写

计算机需要安装有Python2.4或更新版本才可以使用

## 应用案例

### 准备工作

安装软件

设置本地域名解析方便远程

[root@proxy ~]# cat /etc/hosts

… …

192.168.2.100 host1

192.168.2.200 host2

192.168.4.10 host3

创建主机列表文件

[root@proxy ~]# cat /root/host.txt //每行一个用户名、IP或域名

… …

root@host1

host2 //默认用户为root

host3

### pssh远程连接

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

案例

[root@proxy ~]# pssh -i -A -H ‘host1 host2 host3’ \

> -x ‘-o StrictHostKeyChecking=no’ echo hello

//使用密码远程多台主机执行命令,屏幕显示标准和错误输出信息

[root@proxy ~]# pssh -A -h host.txt \

> -x ‘-o StrictHostKeyChecking=no’ echo hello

//使用密码远程多台主机执行命令,不显示标准和错误输出信息

//通过读取host.txt读取主机信息

[root@proxy ~]# ssh-keygen -N “ -f /root/.ssh/id\_rsa” //非交互生成密钥文件

[root@proxy ~]# ssh-copy-id -i /root/.ssh/id\_rsa.pub host1

[root@proxy ~]# ssh-copy-id -i /root/.ssh/id\_rsa.pub host2

[root@proxy ~]# ssh-copy-id -i /root/.ssh/id\_rsa.pub host3

[root@proxy ~]# pssh -h host.txt -x ‘-o StrictHostKeyChecking=no’ echo hello

//使用密钥远程其他主机

[root@proxy ~]# pssh -h host.txt -o /tmp/ -x ‘-o StrictHostKeyChecking=no’ echo hello //使用密钥远程其他主机,将标准输出信息写入到/tmp目录

### pscp.pssh远程拷贝

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

[root@proxy ~]# pscp.pssh -h host.txt -x ‘-o StrictHostKeyChecking=no’ etc.tar.gz / //将本地的etc.tar.gz拷贝到远程主机的根目录下

[root@proxy ~]# pscp.pssh -r -h host.txt \

> -x ‘-o StrictHostKeyChecking=no’ /etc /tmp

//递归将本地的/etc目录拷贝到远程主机的/tmp目录下

### pslurp远程下载数据

pslurp提供远程下载功能

-选项与pscp.pssh基本一致

[root@proxy ~]# pslurp -r -h host.txt -x ‘-o StrictHostKeyChecking=no’ /etc /tmp

[root@proxy ~]# pslurp -r -h host.txt -x ‘-o StrictHostKeyChecking=no’ > -L /media /etc /tmp

//将远程主机的/etc/目录,拷贝到media 下,存放在对应IP下的tmp目录

### pnuke远程杀死进程

pnuke提供远程杀死进程功能

-选项与pssh基本一致

[root@proxy ~]# pnuke -h host.txt -x ‘-o StrictHostKeyChecking=no’ sleep

//将远程主机上的sleep进程杀死

[root@proxy ~]# pnuke -h host.txt -x ‘-o StrictHostKeyChecking=no’ test

//将远程主机上的test相关脚本都杀死(如:test1,testtt,test2等等)

[root@proxy ~]# pnuke -h host.txt -x ‘-o StrictHostKeyChecking=no’ test.sh

//将远程主机上的test.sh脚本杀死

# 总结和答疑

## VPN连接错误

### 问题现象

故障错误信息

-Windows某些版本的系统连接VPN时,报错789

### 原因分析

分析故障信息

-连接错误代码:789

分析故障原因

-需要修改注册表,并且重新启动计算机

## 时间不同步

### 问题现象

故障错误信息

-重启chronyd后时间不变

### 原因分析

分析故障信息

-重启服务后时间不同步

分析故障原因

-服务器防火墙是否关闭

-服务器是否配置了正确的allow与deny规则

-服务器是否已经启动了服务

# 存储技术与应用

## 存储概述

### 存储的目标

存储是分局不同的应用环境通过采取合理、安全、有效的方式将数据保存到某些介质上并能保证有效的访问

一方面它是数据临时或长期驻留的物理媒介

另一方面,它是保证数据完整安全存放的方式或行为

存储就是把这两个方面结合起来,向客户提供一套数据存放解决方案

### 存储技术分类

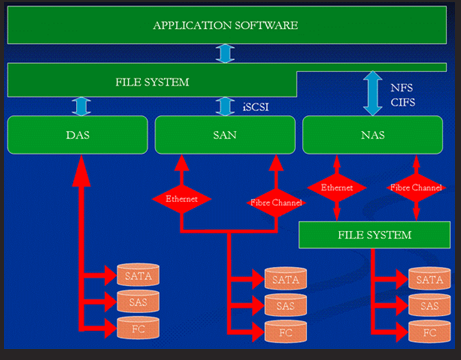
SCSI小型计算机系统接口

DAS直连式存储

NAS网络技术存储

SAN存储区域网络

FC光纤通道



## 常见存储技术

### SCSI技术

Small Computer System Interface的简称

作为输入/输出接口

主要用于硬盘、光盘、磁盘带等设备

### DAS技术

Direct-Attached Storage的简称

将存储设备通过SCSI接口或光纤通道直接连接到计算机上

不能实现数据与其他主机的共享

占用服务器操作系统资源,如CPU、IO等

数据量越大,性能越差

### NAS技术

Network-Attached Storage的简称

一种专用数据存储服务器,以数据为中心,将存储设备与服务器彻底分离,集中管理数据,从而释放带宽、提高性能、降低总拥有成本、保护投资

用户通过TCP/IP协议访问数据

-采用标准的NFS/HTTTP/CIFS等

### SAN技术

Storage Area Network的简称

通过光纤交换机、光纤路由器、光纤集线器等设备将磁盘阵列、磁带等存储设备与相关服务器连接起来,形成高速专网网络

组成部分

-如路由器、光纤交换机、网卡

接口:如SCSI、FC

通信协议:如IP、SCSI

Fibre Channel

一种适合于千兆数据传输的、成熟而安全解决方案

与传统的SCSI相比,FC提供更高的数据传输速率、更远的传输距离、更多的设备连接支持以及更稳定的性能、更简易的安装

FC主要组件

光纤

HBA(主机总线适配置器)

FC交换机

分层交换机交换拓扑

点到点:point-to-point

简单将两个设备互连

已裁定的环路:arbitrated loop

可多达126个设备共享一段信道或环路

交换式拓扑:switched fabric

所有设备通过光纤交换机互连

### ISCSI技术

Internet SCSI

IETF制定的标准,将SCSI数据块映射为以太网数据包

是一种基于IP Storage理论的新型存储技术

将存储行业广泛应用的SCSI接口技术与IP网络相结合

可以在IP网络上构建SAN

最初由Cisco和IBM开发

优势

基于IP协议技术的标准

允许网络在TCP/IP协议上传输SCSI命令

相对FC SAN,ISCSI实现的IP SAN投资更低

解决了传输效率、存储容量、兼容性、开放性、安全性等方面的问题

没有距离限制

客户端

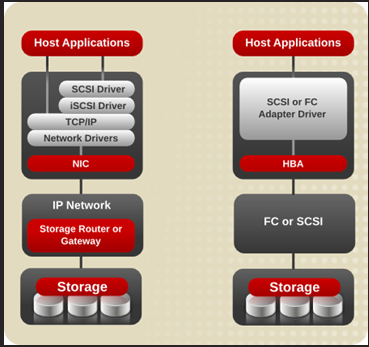
iSCSI Initiator:软件实现,成本低、性能较低

iSCSI HBA:硬件实现,性能好,成本较高

存储设备端

iSCSI Target

以太网交换机



# iSCSI技术应用

## 基础知识

### iSCSI操作规范

Target端

选择target名称

安装iSCSI target

准备用于target的存储

配置target

启用服务

Initiator端

安装initiator

配置initiator并启动服务

### iSCSI命名规范

建议采用IQN(iSCSI限定名称)

全称必须全局唯一

IQN格式:

iqn.<date\_code>.<reversed\_domain>.<string>[:<substring>]

命名示例:

iqn.2013-01.com.tarena.tech:sata.rack2.disk1

## 部署iSCSI服务

### 安装target软件

查询yum仓库

# yum list | grep target

安装

# yum -y install targetcli

查看iSCSI target信息

# yum info targetcli

### 配置iSCSI Target

定义后端存储

# targetcli

/> ls

/> backstores/block create store /dev/vdb1

创建iqn对象

/> /iscsi create iqn.2018-01.cn.tedu:server1

授权客户机访问

/> iscsi/iqn.2018-01.cn.tedu:serever1/tpg1/acls create

iqn.2018-01.cn.tedu:client1

绑定存储

/> iscsi/iqn.2018-01.cn.tedu:serever1/tpg1/luns create

/backstores/block/iscsi\_store

绑定监听地址

/> iscsi/iqn.2018-01.cn.tedu:serever1/tpg1/portals/create 0.0.0.0

保存配置

/> saveconfig

/> exit

### 服务管理

控制服务

# systemctl {start|restart|stop|status} target

设置服务开机运行

# systemctl enable target

查看端口

# netstat -tlnp | grep:3260

### 安装客户端软件

查询yum仓库

# yum list | grep initator

安装

# yum -y install iscsi-initiator-utils

查看iSCSI target信息

# yum info targetcli iscsi-initiator-utils

### 客户端访问并验证

启动服务

# service iscsi start

设置本机的iqn名称

vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2018-01.cn.tedu:client1

发现远程存储

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

登录target

# systemctl restart iscsi

# lsblk

# systemctl enable iscsi

# systemctl enable iscsid

iscsi用于自动login远程存储，iscsid是守护进程。

# Multipath多路径

## 基础知识

### 多路径概述

当服务器到某一存储设备有多条路径时,每条路径都会识别为一个单独的设备

多路径允许您将服务器节点和储存阵列间的多个/IO路径配置为一个单一设备

这些I/O路径是可包含独立电缆、交换器和控制器的实体SAN链接

多路径集合了I/O路径,并生成由这些集合路径组成的新设备

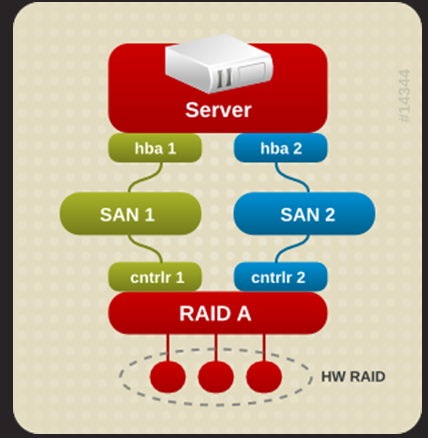
### 多路径主要功能

冗余

-主备模式,高可用

改性的性能

-主主模式,负载均衡



## 配置案例

### 拓扑规划

利用iSCSI实现多路径

应用服务器 IP地址

eth0 192.168.1.10/24

eth1 192.168.2.10/24

存储节点 IP地址

eth0 192.168.1.20/24

eth1 192.168.2.20/24

### 准备共享存储

配置iSCSI服务端

-准备共享介质(分区、LV或磁盘镜像)

-安装iscsi-target-utils

-准备规划iqn名称

-修改target.conf配置文件,以提供存储

-启动tdtd服务

### 访问共享存储

因为到达共享存储有两条路径,所以需要在两条路径上都执行发现命令

设置开机自启

### 多路径设备

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

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

### 多路径配置概述

安装软件包

# yum -y install -y device-mapper-multipath

使用mpathconf命令创建配置文件并启用多路径

# mapathconf --user\_friendly\_name n

若无需编辑该配置文件,可使用此命令启动多路径守护程序

### 多路径设备识别符

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

默认情况下会将多路径设备的名称设定为它的WWID

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

也可以自定义存储设备名称

### 获取WWID

登陆共享存储后，系统多了两块硬盘，这两块硬盘实际上是同一个存储设备。应用服务器使用哪个都可以，但是如果使用sdb时，sdb对应的链路出现故障，它不会自动切换到sda。

为了能够实现系统自动选择使用哪条链路，需要将这两块磁盘绑定为一个名称。

通过磁盘的wwid来判定哪些磁盘是相同的。

假如共享存储在本地被识别为/dev/sdb和/dev/sdc,那么获取它WWID的方法是:

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

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

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

### 指定获取WWID的方法

在配置文件中声明获取WWID的方法

# vim /etc/multipath.conf

defaults {

user\_friendly\_names no

find\_multipath yes

}

### 为多路径设备配置别名

根据得到的WWID,为多路径设备配置别名

# vim /etc/multipath.conf

#在尾部添加以下内容:

multipaths {

multipath {

wwid “1lET 00010001”

alias mpatha

}

}

### 启动服务并验证

启动服务

# systemctl start multipathd

# systemctl enable multipathd

验证

# ls /dev/mapper #mpatha即为多路径设备

# multipath -rr #重新加载多径信息

# multipath -ll #查看多径信息

分区

-为/dev/mapper/mpatha 分区,得到的第一个分取名

为/dev/mapper/mpathap1

# NFS网络文件系统

## NFS服务基础

### 文件系统的类型

本地文件系统

-EXT3/4、SWAP、NTFS、...... ------本地磁盘

伪文件系统

-/proc、/sys、...... ------内存空间

网络文件系统

NFS(Network File System) ------网络存储空间

### NFS共享协议

Unix/Linux最基本的文件共享机制

-1980年由SUN公司开发

-依赖于RPC(远程过程调用)映射机制

-存取位于远程磁盘中的文档数据,对应用程序是透明的,就好像访问本地的文件一样

## 配置并访问NFS共享

### 配置NFS服务器

主要软件包

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

-rpcbind-0.2.0-42.el7.x86\_64

-系统服务脚本

-nfs、rpcbind

主配置文件

-/etc/exports

/etc/exports配置解析

-共享目录 客户机地址(参数,参数,.. ..) .. ..

vim /etc/exports

/root 192.168.4.20(rw) pc110(rw,no\_root\_squash)

IP地址:192.168.4.20 rw、ro:可读可写、只读

网段地址:172.0.0.0/24或172.0.0.\* sync、async:同步写、异步写入

所有主机:\* no\_root\_squash:保留来自客户端的root权限

单个域:\*.tarena.com all\_squash:客户端权限都降为nfsnobody

主机名:pc110.tarena.com

默认情况下，来自NFS客户端的root用户会被自动降权为普通用户，若要保留其root权限，注意应添加no\_root\_squash控制参数(没有该参数，默认root会被自动降级为普通账户)

### 使用NFS客户端

设置2个共享文件夹

-将/root共享给192.168.4.205,可写、保留客户端的root权限

-将/usr/src共享给192.18.4.0/24网段,只读

# vim /etc/exports

/root 192.168.4.205(rw,no\_root\_squash)

/usr/src 192.168.4.0/24(ro)

# systemctl restart rpcbind //启动RPC机制

# systemctl restart nfs //启动nfs服务

rpcbind:111端口

注册端口

nfs:2489(nfs为随机端口)

查看NFS共享列表

-showmount -e [服务器地址]

# showmount -e 192.168.4.5

挂载NFS共享目录

-mount [-t nfs] 服务器地址:共享目录 挂载点

# mount 192.168.4.5:/root /mnt/root5/

# ls /mnt/root5/

anaconda-ks.cfg install.log install.log.syslog

注意事项:

1. 从未授权的客户机将无法挂载此共享
2. 若未启用 no\_root\_squash,挂载后会无权限浏览(750)

# udev配置

## 基础知识

### 设备文件管理方法

devfs

-Linux早起采用的静态管理方法

-/dev目录下有大量静态文件

-内核版本2.6.13开始被完全取代

udev动态设备管理

-只有连到系统上来的设备才在/dev下创建设备文件

-与主、次设备编号无关

-为设备提供持久、一致的名字

### 接入设备事件链

内核发现设备并导入设备状态到sysfs

udev接到事件通知

udev创建设备节点或是运行指定程序

udev通知hald守护进程

HAL探测设备信息

HAL创建设备对象结构

HAL通过系统消息总线广播该事件

用户程序也可以监控该事件

## udev的作用

### udev应用

从内核收到添加/移除硬件事件时,udev将会分析:

- /sys目录下信息

- /etc/udev/rules.d目录中的规则

基于分析结果,udev会:

-处理设备命名

-决定要创建哪些设备文件或链接

-决定如何设置属性

-决定触发哪些事件

### udev时间监控

udevadm monitor --property

### 配置udev

主配置文件/etc/udev/udev.conf

-udev\_root:创建设备文件位置,默认为/dev

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

-udev\_log:syslog优先级,缺省为err

文件位置及格式

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

- 例:75-custom.rules

规则格式

-<match-key><op><value>[,...]<assignmet-key><op>value[,...]

-BUS=”usb”,SYSFS{serial}==”20043512321411d34721”,NAME=”udisk”

### udev操作符

- ==:表示匹配

- != :表示不匹配

匹配示例

- ACTION==”add”

- KERNEL==”sd[a-z]1”

- BUS==”scsi”

- DRIVER!=”ide-cdrom”

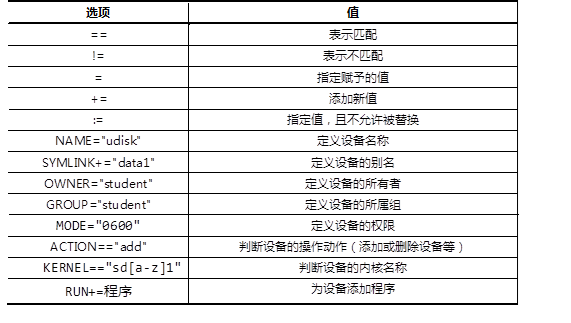
- PROGRAM==”myapp.pl”,RESULT==”test”

操作符

- =:指定赋予的值

- +=:添加新值

- :=:指定值,且不允许被替换



示例

- NAME=”udisk”

- SYMLINK+=”data1”

- OWNER=”student”

- MODE=”0600”

### udev变量

可以简化或缩写规则

KERNEL==”sda\*”,SYMLINK+=”iscsi%n”

常用替代变量

- %k:内核所识别出来的设备名,如sdb1

- %n:设备的内核编号,如sda3中的3

- %p:设备路径,如/sys/block/sdb/sdb1

- %%:%符号本身

### udev案例分析

# 总结和答疑

## 客户端检测不到磁盘

### 问题现象

iSCSI客户端可以发现target

登录之后,本地并没有出现新的磁盘设备

### 故障分析及排除

原因分析

-target端没有出现LUN1

-对target端进行修改,重启服务后,target没有LUN1

解决办法

-只能重启操作系统

## NFS无法写入数据

### 问题现象

NFS服务器端已配置共享输出目录为读写权限

客户端可以正常挂载服务器的输出目录

客户端向服务器输出目录写入数据失败

### 故障分析及排除

原因分析

-服务器没有配置本地的写权限

-输出目录权限的最终权限,由本地权限和配置文件内的权限共同决定

解决办法

-将目录的本地权限改为777

### 

# 集群及LVS简介

## 集群简介

### 什么是集群

一组通过高速网络互联的计算组,并以单一系统的模式加以管理

将很多服务器集中起来一起,提供同一种访问,在客户端看来就像是只有一个服务器

可以在付出较低成本的情况下获得在性能、可靠性、灵活性方面的相对较高的收益

任务调度是集群系统中的核心技术

### 集群目的

提供性能

- 如计算密集型应用,如:天气预报、核试验模拟

降低成本

- 相对百万美元级的超级计算机,价格便宜

提高可扩展性

- 只要增加集群节点即可

增强可靠性

- 多个节点完成相同功能,避免单点失败

### 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

## LVS概述

### LVS项目介绍

Linux虚拟服务器(LVS)是章文嵩在国防科技大学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

最终目标是利用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集群组成

前段: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

### 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

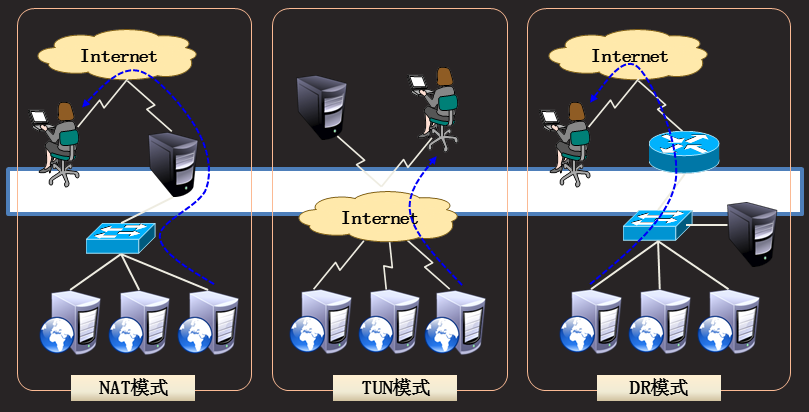
VIP:虚拟IP地址

-公布给用户访问的虚拟IP地址

RIP:真实IP地址

-集群节点上使用的IP地址

DIP:调度器连接节点服务器的IP地址



### LVS工作模式

VS/NAT

适合数据量小的环境使用

-通过网络地址转换实现的虚拟服务器

-大并发访问时,调度器的性能成为瓶颈

VS/DR(直连路由模式)

适合数据量大的环境使用

-直接使用路由技术实现虚拟服务器

-节点服务器需要配置VIP,注意MAC地址广播

VS/TUN(隧道模式)

-通过隧道方式实现虚拟服务器

### 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)

-加权轮询(Weighted Round Robin)

-最少连接(Least Connections)

-加权最少连接(Weighted Least Connections)

轮询(Round Robin)

-将客户端请求平均分发到Real Server

加权轮询(Weighted Round Robin)

-根据Real Server权重值进行轮询调度

最少连接(Least Connections)

-选择连接数最少的服务器

加权最少连接

-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

-基于局部性的最少链接

-带复制的基于局部性最少链接

-目标地址散列(Destination Hashing)

-最短的期望的延迟

-最少的队列调度

# LVS-NAT集群

## 软件安装

### 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

### 安装ipvsadm

使用rpm命令安装ipvsadm

### ipvsadm用法

创建虚拟服务器

-A 添加虚拟服务器

-t 设置群集地址(VIP,Virtual IP)

-s 指定负载调度算法

添加、删除服务器节点

-a 添加真实服务器

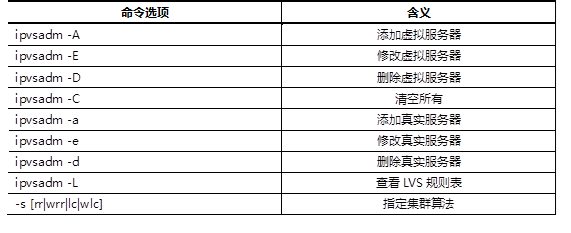
-d 删除真实服务器

-r 指定真实服务器(Real Server)的地址

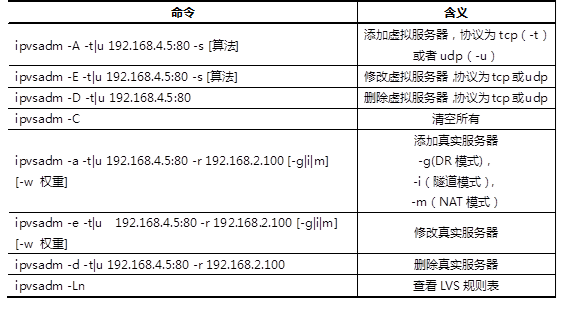
-m 使用NAT模式;-g、-i分别对应DR、TUN模式

-w 为节点服务器设置权重,默认为1

查看IPVS



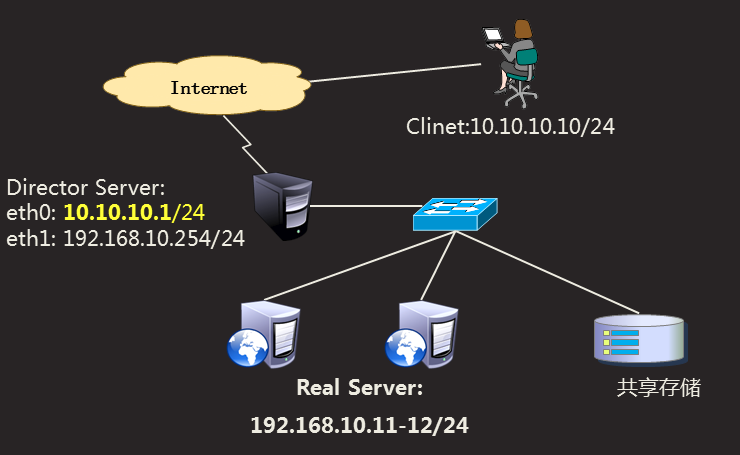
ipvsadm命令选项



ipvsadm语法案例

## LVS-NAT案例

### LVS-NAT集群拓扑



### 操作流程

Real Server:

-配置WEB服务器

Director Server:

-在上安装并启用ipvsadm

-创建虚拟服务器

-向虚拟服务器中加入节点

Client:

-连接虚拟服务器测试

### 部署LVS调度器

打开ip\_forward

# vim /etc/sysctl.conf

net.ipv4.ip\_forward = 1

# sysctl -p

启动ipvsadm

# systemctl start ipvsadm

# systemctl enable ipvsadm

创建虚拟服务器,VIP为10.10.10.1,采用的调度算法为Round Robin

# ipvsadm -A -t 10.10.10.1:80 -s rr

向虚拟服务器中加入节点,并指定权重分别为1和2,目前权重不起作用

# ipvsadm -a -t 10.10.10.1:80 -r 192.168.10.11 -m -w 1

# ipvsadm -a -t 10.10.10.1:80 -r 192.168.10.12 -m -w 2

查看配置

# ipvsadm -L -n //注意:这两个选项不写反

保存配置

# ipvsadm-save > /etc/sysconfig/ipvsadm

修改Director调度算法为WRR

# ipvsadm -E -t 10.10.10.1:80 -s wrr

### Client端验证

通过web浏览器访问

使用ab进行大并发测试

# ab -c 10 -n 1000 <http://10.10.10.1/index>.html

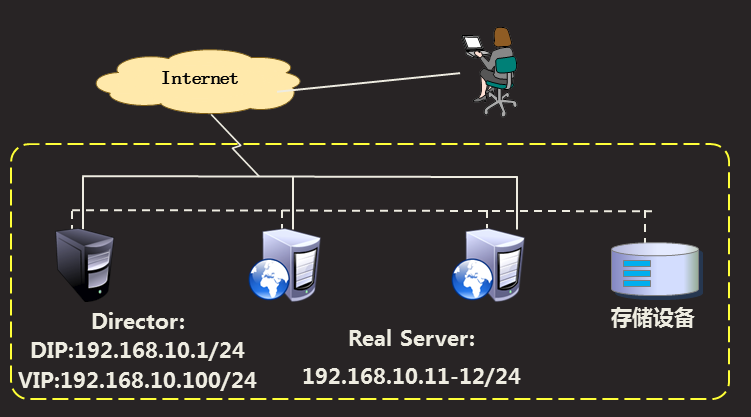
在Director上查看连接数

# ipvsadm -L -n

# LVS-DR集群

## 案例说明

### LVS-DR集群拓扑



### 操作流程

Real Server:

-配置WEB服务器

-配置辅助IP地址、调整内核参数

Director Server:

-在上安装并启用ipvsadm

-配置辅助IP地址

-创建虚拟服务器、向虚拟服务器中加入节点

Client:

-连接虚拟服务器测试

CIP是客户端的IP地址；

VIP是对客户端提供服务的IP地址；

RIP是后端服务器的真实IP地址；

DIP是调度器与后端服务器通信的IP地址（VIP必须配置在虚拟接口）。

### ARP广播的问题

当客户端发起访问VIP对应的域名的请求时,根据网络通信原理会产生ARP广播

因为负载均衡器和真实的服务器在同一网络并且VIP设置在集群中的每个节点上

此时集群内的真实服务器会尝试回答来自客户端的ARP广播,这就会产生问题,大家都说我是”VIP”

### 内核参数说明

arp\_ignore(定义回复ARP广播的方式)

-0(默认值)

回应所有的本地地址ARP广播,本地地址可以配置在任意网络接口

-1

只回应配置在入站网卡接口上的任意IP地址的ARP广播

arp\_announce

-0(默认)

使用配置在任意网卡接口上的本地IP地址

-2

对查询目标使用最适当的本地地址.在此模式下将忽略这个IP数据包的源地址并尝试选择与能与该地址通信的本地地址.首要是选择所有的网络接口的子网中外出访问子网中包含该目标IP地址的本地地址.如果没有合适的地址被发现,将选择当前的发送网络接口或其他的有可能接受到该ARP回应的网络接口来进行发送

### ARP防火墙

使用ARP防火墙也可以禁止对VIP的ARP请求

## LVS-DR案例实施

### 配置后端Web服务器

配置辅助VIP地址

为了防止冲突，VIP必须要配置在网卡的虚拟接口

在web1和web2上做以下配置修改

这里的子网掩码必须是32（也就是全255），网络地址与IP地址一样，广播地址与IP地址也一样。

[root@web1 ~]# cd /etc/sysconfig/network-scripts/

[root@web1 ~]# cp ifcfg-lo{,:0}

[root@web1 ~]# vim ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

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。

调整内核参数

[root@web1 ~]# vim /etc/sysctl.conf

#手动写入如下4行内容

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

net.ipv4.conf.all.arp\_announce = 2

#当有arp广播问谁是192.168.4.15时，本机忽略该ARP广播，不做任何回应

#本机不要向外宣告自己的lo回环地址是192.168.4.15

[root@web1 ~]# sysctl -p

### 部署LVS调度器

配置辅助IP地址

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

[root@proxy ~]# cp ifcfg-eth0{,:0}

[root@proxy ~]# vim ifcfg-eth0

TYPE=Ethernet

BOOTPROTO=none

NAME=eth0

DEVICE=eth0

ONBOOT=yes

IPADDR=192.168.4.5

PREFIX=24

[root@proxy ~]# vim ifcfg-eth0:0

TYPE=Ethernet

BOOTPROTO=none

DEFROUTE=yes

NAME=eth0:0

DEVICE=eth0:0

ONBOOT=yes

IPADDR=192.168.4.15

PREFIX=24

[root@proxy ~]# systemctl restart network

创建虚拟服务器

向虚拟服务器中加入节点

清理之前实验的规则，创建新的集群服务器规则

[root@proxy ~]# ipvsadm -C #清空所有规则

[root@proxy ~]# ipvsadm -A -t 192.168.4.15:80 -s wrr

添加真实服务器(-g参数设置LVS工作模式为DR模式，-w设置权重)

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100 -g -w 1

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200 -g -w 1

查看规则列表，并保存规则

[root@proxy ~]# ipvsadm -Ln

TCP 192.168.4.15:80 wrr

-> 192.168.4.100:80 Route 1 0 0

-> 192.168.4.200:80 Route 2 0 0

[root@proxy ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

### Client端验证

通过web浏览器访问

使用ab进行大并发测试

在Director上查看连接数

客户端使用curl命令反复连接http://192.168.4.15，查看访问的页面是否会轮询到不同的后端真实服务器。

扩展知识：默认LVS不带健康检查功能，需要自己手动编写动态检测脚本，实现该功能：(参考脚本如下，仅供参考)

[root@proxy ~]# vim check.sh

#!/bin/bash

VIP=192.168.4.15:80

RIP1=192.168.4.100

RIP2=192.168.4.200

while :

do

for IP in $RIP1 $RIP2

do

curl -s http://$IP &>/dev/vnull

if [ $? -eq 0 ];then

ipvsadm -Ln |grep -q $IP || ipvsadm -a -t $VIP -r $IP

else

ipvsadm -Ln |grep -q $IP && ipvsadm -d -t $VIP -r $IP

fi

done

sleep 1

done

# 总结和答疑

## 加权轮询不生效

### 问题现象

在LVS调度器上配置的real server权重不一样

客户机访问服务器时,real server的权重不起作用,只是简单的轮询

### 故障分析及排除

原因分析

-LVS调度器虽然为real server设置了权重,但是调度算法仍然是轮询

-轮询算法根本不检查权重值

解决办法

-修改调度算法为wrr

## 服务无法启动

### 问题现象

启动LVS服务时,启动失败

尝试以restart方式启动服务,仍然失败

### 故障分析及排除

原因分析

-检查服务状态和日志,提示没有/etc/sysconfig/ipvsadm文件

-该文件是调度规则文件

解决办法

-先将规则存盘,再重启服务

## 无法访问网页

### 问题现象

当客户端使用浏览器访问网页的时候无法加载

### 故障分析及排除

原因分析

-在服务器配好网关之后没有重新激活网卡

-配置NAT集群的时候,web1的eth0没有关闭,导致LVS调度器和Web服务器处于相同网段

解决办法

-重新激活网卡即可

-关闭web1的eth0

### 

# Keepalived热备

## Keepalived概述

### 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最初是为了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

Keepalived功能:

1. 自动配置LVS规则,做健康检查
2. 学习了路由器上的功能VRRP, HSRP路由热备

### 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

## Keepalived服务

### Keepalived安装

RHEL7的光盘中已经包含Keepalived软件包,只要配置好yum,指向光盘源即可安装

### 配置文件解析

/etc/keepalived/keepalived.conf

[root@web1 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

admin@tarena.com.cn //设置报警收件人邮箱

}

notification\_email\_from ka@localhost //设置发件人

smtp\_server 127.0.0.1 //定义邮件服务器

smtp\_connect\_timeout 30

router\_id web1 //设置路由ID号（实验需要修改）

}

vrrp\_instance VI\_1 {

state MASTER //主服务器为MASTER（备服务器需要修改为BACKUP）

interface eth0 //定义网络接口

virtual\_router\_id 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）

}

### 高可用Web拓扑

### 高可用Web案例

配置web服务器

使用Keepalived为服务器提供VIP

启动keepalived会自动添加一个drop的防火墙规则，需要清空

客户端验证的时候,要使用iptables -F清空防火墙规则

Web服务器查看VIP信息

# ip address show

# Keepalived+LVS

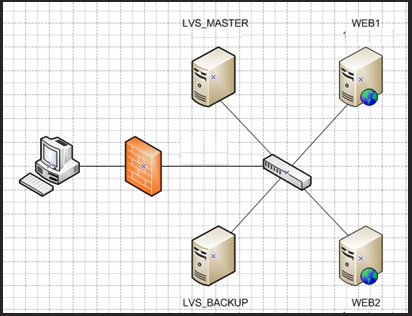
## 高可用调度器

### Keepalived+LVS拓扑

使用Keepalived高可用解决调度器单点失败问题

主、备调度器上配置LVS

主调度器异常时,Keepalived启用备用调度器调度



### Keepalived配置说明

LVS相关信息通过Keepalived配置即可

主要配置文件说明如下

VRRP实例设置

### Real Server配置

真实服务器运行在DR模式下

修改内核参数,并附加VIP

详细配置参见LVS相关章节

## 集群调度软件对比

### 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

### LVS分析

优点

-负载能力强,工作在4层,对内存、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

### 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

# HAProxy服务器

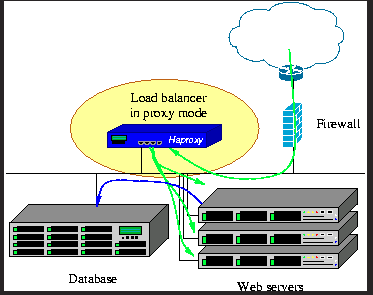
## HAProxy概述

### HAProxy简介

它是免费、快速并且可靠的一种解决方案

适用于那些负载特大的web站点,这些站点通常又需要会话保持或七层处理

提供高可用性、负载均衡以及基于TCP和HTTP应用的代理



### 衡量负责均衡器性能的因素

Session rate会话率

-每秒钟产生的会话数

Session concurrency并发会话数

-服务器处理会话的时间越长,并发会话数越多

Data rate数据速率

-以MB/s或Mbps衡量

-大的对象导致并发会话数增加

-高会话数、高数据速率要求更多的内存

### HAProxy工作模式

mode http

-客户端请求被深度分析后再发往服务器

mode tcp

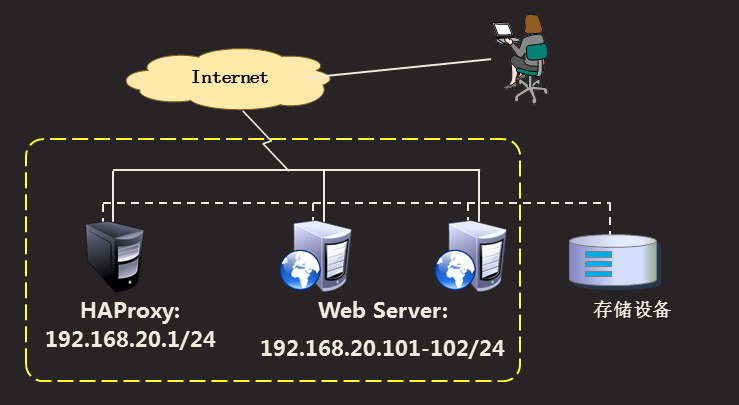
-4层调度,不检查第七层信息

mode health

-仅做健康状态检查,已经不建议使用

## HAProxy配置示例

### 项目拓扑图



### HAProxy安装

RHEL7光盘中内置了HAProxy,只要配置好yum,可以直接安装

### 配置文件说明

HAProxy配置参数来源

-命令行:总是具有最高优先级

-global部分:全局设置进程级别参数

-代理声明部分

来自于default、listen、frontend和backend

配置文件可由如下部分构成:

-default

为后续的其他部分设置缺省参数

缺省参数可以被后续部分重置

-frontend

描述接收客户端侦听套接字(socket)集

-backend

描述转发链接的服务器集

-listen

把frontend和backend结合到一起的完整声明

/etc/haproxy/haproxy.cfg

### 管理服务

启动服务

停止服务

查看状态

### 监控HAProxy状态

# 总结和答疑

## Keepalived产生大量日志

### 问题现象

当观察/var/log/messages日志时,发现该文件每秒钟都产生了很多条日志记录

如果不及时解决,该文件会迅速增长

### 故障分析及排除

原因分析

-Keepalived的工作原理与VRRP相同

-VRRP相同组要求有相同的密码、VIP和组号,如果不一致就会产生日志通知

解决办法

-检查两台Keepalived配置,将虚拟IP、虚拟路由器ID和密码修改成一样的

## LVS规则不完整

### 问题现象

通过Keepalived配置LVS规则,查看LVS规则时,只有一台real server

经检查real server工作都未出现异常

### 故障分析及排除

原因分析

-直接访问real server没有异常

-问题应该出现在Keepalived配置文件

解决办法

-经检查,发现配置文件中,TCP\_CHECK与后面的花括号少了一个空格

# Ceph概述

## 基础知识

### 什么是分布式文件系统

### 常用分布式文件系统

### 什么是Ceph

### Ceph组件

## 实验环境准备

### 实验拓扑图

### 配置YUM

### 配置SSH无密钥连接

### NTP时间同步

### 准备存储磁盘

# 部署Ceph集群

## 准备部署环境

### 安装部署软件

### 创建目录

## 部署存储集群

### 创建Ceph集群

### 创建OSD

### 验证

# Ceph块存储

## 概述

### 什么是块存储

## 块存储集群

### 创建镜像

### 动态调整大小

### 集群内通过KRBD访问

### 客户端通过KRBD访问

### 创建镜像快照

### 使用快照恢复数据

### 快照克隆

### 客户端撤销磁盘映射

### 删除快照与镜像

# 总结和答疑

## 还原快照后无法挂载

### 问题现象

### 故障分析及排除

### 

# 块存储应用案例

## 准备实验环境

### 创建磁盘镜像

### Ceph认证账户

### 部署客户端环境

## 创建KVM虚拟机

### 创建初始化虚拟机

### 配置libvirt secret

### 虚拟机的XML配置文件

### 修改XML配置文件

# 分布式文件系统

### 基础知识

### 什么是CephFS

### 环境准备

## 元数据服务器

### 什么是元数据

### 部署元数据服务器

## 文件系统服务器

### 创建存储池

### 创建Ceph文件系统

### 客户端挂载

# Ceph对象存储

## 概述

### 什么是对象存储

### 环境准备

## 对象存储

### 部署RGW软件包

### 新建网关示例

### 修改服务端口

### 客户端测试

### 使用第三方软件访问

# 总结和答疑

## ceph-deploy无法完成部署

### 问题现象

### 故障分析及排除

### 

# 块存储应用案例

## 准备实验环境

### 创建磁盘镜像

### Ceph认证账户

### 部署客户端环境

## 创建KVM虚拟机

### 创建初始化虚拟机

### 配置libvirt secret

### 虚拟机的XML配置文件

### 修改XML配置文件

# 分布式文件系统

## 基础知识

### 什么是CephFS

### 环境准备

## 元数据服务器

### 什么是元数据

### 部署元数据服务器

## 文件系统服务器

### 创建存储池

### 创建Ceph文件系统

### 客户端挂载

# Ceph对象存储

## 概述

### 什么是对象存储

### 环境准备

## 对象存储

### 部署RGW软件包

### 新建网关实例

### 修改服务端口

### 客户端测试

### 使用第三方软件访问

# 总结和答疑

## ceph-deploy无法完成部署

### 问题现象

### 故障分析及排除