统信服务器操作系统 V20（1060）

管理手册

版本号	B04	**文件编号**	FWQ20230530T_RL02
文档密级	C 级商密	**编制部门**	服务器操作系统与云计算产线
编制人	庞庆	**日期**	2023 年 05 月 29 日
审核人	杨晓旋	**日期**	2023 年 05 月 29 日
批准人	薛皓琳	**日期**	2023 年 05 月 29 日

修订记录

序号	版本号	修订内容描述	修订日期	修订人	审核人	批准人
1	A01	创建。	2020/06/22	王策	潘晨博	叶青龙
2	A02	修改 repo 描述、简称和 ssh 登录信息。	2020/11/02	王策	潘晨博	叶青龙
3	A03	增加“升级操作”章节以及修改格式。	2021/02/24	刘刚	潘晨博	叶青龙
4	A04	修改特殊字符和错别字。	2021/05/27	刘刚	潘晨博	叶青龙
5	A05	增加虚拟化、nmcli 章节。	2021/06/23	刘刚	潘晨博	叶青龙
6	A06	增加迁移助手、A-Tune、AWX 工具章节。	2021/07/08	刘刚	潘晨博	叶青龙
7	A07	增加模块化安装问题的 FAQ 章节。	2021/12/29	何仁贵	潘晨博	叶青龙
8	A08	增加 AppStream 使用章节。	2022/01/29	郝咪咪	潘晨博	薛皓琳
9	A09	修改 AppStream 测试样例。	2022/01/29	吕耕耕	潘晨博	薛皓琳
10	A10	添加模块化源配置方法。	2022/02/08	吕耕耕	潘晨博	薛皓琳

统信软件技术有限公司 © 版权所有

序号	版本号	修订内容描述	修订日期	修订人	审核人	批准人
11	A11	添加飞腾环境使用 vi 查看/var/log/messages 不能正确显示中文字符的 FAQ。	2022/02/08	何仁贵	潘晨博	薛皓琳
12	A12	添加 perl 模块使能导致依赖 perl 的非模块化包安装失败 FAQ。	2022/03/30	吕耕耕	潘晨博	薛皓琳
13	A13	修改封面文本字体 ;修改“虚拟机 XML 文件”章节中示例镜像名称，由 1060e 变更为 1060u1e。	2022/05/17	何仁贵	潘晨博	薛皓琳
14	A14	按照最新文档模板刷新文档格式。	2022/06/01	何仁贵	潘晨博	薛皓琳
15	A15	增加“启用额外软件源”章节。	2022/06/24	孔立栋	潘晨博	薛皓琳
16	A16	增加“系统更新提示”章节。	2022/07/01	赵爽	潘晨博	薛皓琳
17	A17	更新 X86 虚拟机 XML 文件示例中的 machine 参数为“q35”，增加 cpu mode 配置项。	2022/07/07	何仁贵	潘晨博	薛皓琳

序号	版本号	修订内容描述	修订日期	修订人	审核人	批准人
18	A18	“密码管理”添加“认证模块”；“常用模块介绍”添加“pam_pwquality.so”和“pam_deepin_pw_check.so”。	2022/07/20	何仁贵	潘晨博	薛皓琳
19	A19	修改错别字，虚拟机 xml 配置文件，密码策略相关描述。	2022/07/27	吕耕耕	潘晨博	薛皓琳
20	A20	修正部分图和表的描述信息，增加几个国家法律文件名称限定描述。	2022/08/19	何仁贵	潘晨博	薛皓琳
21	A21	删除“升级操作”章节。	2022/08/23	孔立栋	潘晨博	薛皓琳
22	A22	根据评审意见优化全文。	2022/11/1	孔立栋	潘晨博	薛皓琳
23	B01	将 1060a 和 1060e 的文档合并为一本。	2023/01/12	张柯	潘晨博	薛皓琳
24	B02	修改部分描述和文档格式	2023/2/14	孔立栋	潘晨博	薛皓琳
25	B03	修改 ARM 架构虚拟机 XML 文件	2023/3/15	庞庆	杨晓旋	薛皓琳

序号	版本号	修订内容描述	修订日期	修订人	审核人	批准人
26	B04	修改标点符号，修正 top 命令	2023/5/29	庞庆	杨晓旋	薛皓琳

目录

修改 ARM 架构虚拟机 XML 文件 III

修改标点符号，修正 top 命令 IV

1 欢迎使用 1

1.1 概要 1

1.2 GNU/Linux 1

1.3 统信服务器操作系统 2

1.4 登录 2

1.5 注销 3

1.6 关机 3

1.7 重启 4

1.8 显示用户名 5

1.9 切换用户 5

2 系统配置 6

2.1 时间调整 6

2.1.1 hwclock 6

2.1.2 date 6

2.1.3 chrony 7

2.1.4 tzselect 7

2.2 用户管理 8

2.2.1 添加用户 8

2.2.2 修改用户 9

2.2.3 删除用户 10
2.2.4 密码管理 10
2.2.5 添加用户组 11
2.2.6 修改用户组 11
2.2.7 删除用户组 12
2.3 文件管理 12
2.3.1 cd 命令 12
2.3.2 pwd 命令 13
2.3.3 id 命令 13
2.3.4 ls 命令 14
2.3.5 mkdir 命令 14
2.3.6 rm 命令 15
2.3.7 cp 命令 15
2.3.8 mv 命令 16
2.3.9 cat 命令 16
2.3.10 more 命令 17
2.3.11 less 命令 18
2.3.12 head 命令 19
2.3.13 tail 命令 19
2.3.14 file 命令 20
2.3.15 chmod 命令 20
2.3.16 find 命令 21

2.3.17 stat 命令 23
2.3.18 tar 命令 23
2.3.19 gzip 命令 24
2.4 磁盘管理 25
2.4.1 df 命令 25
2.4.2 du 命令 26
2.4.3 fdisk 命令 27
2.5 任务管理 28
2.5.1 ps 命令 28
2.5.2 kill 命令 29
2.5.3 fg/bg 命令 29
2.5.4 nohup 命令 30
2.5.5 crontab 命令 30
2.5.6 重定向命令 31
2.6 环境变量 32
2.6.1 env 命令 32
2.6.2 export 命令 33
2.6.3 配置/etc/profile 33
2.7 网络管理 34
2.7.1 配置网络 34
2.7.2 ifconfig 命令 34
2.7.3 ifup 命令 34

2.7.4 ifdown 命令 35
2.7.5 nmcli 命令 35
2.7.6 管理 IP 地址 36
2.7.7 网络诊断 39
2.7.8 域名管理 40
2.7.9 netstat 命令 41
2.7.10 网络工具 42
2.7.11 网络下载 43
2.8 系统和服务管理 45
2.8.1 Unit 分类 45
2.8.2 Unit 基础操作 46
2.8.3 Unit 启动耗时 46
2.8.4 Unit 查看 47
2.8.5 Unit 状态 47
2.8.6 Unit 管理 47
2.9 帮助手册 48
3 软件管理 48
3.1 dnf & yum 48
3.2 rpm 49
3.3 which 50
3.4 Update & Upgrade 50
3.4.1 配置软件源 50

3.4.2 安装软件包 52
3.4.3 系统更新提示 53
4 服务器软件 54
4.1 远程连接服务器 54
4.1.1 概述 54
4.1.2 环境准备 54
4.1.3 SSH 安装 54
4.1.4 配置文件 55
4.1.5 功能测试 55
4.2 远程连接桌面环境 55
4.2.1 概述 55
4.2.2 统信 UOS 远程连接 Windows 55
4.2.3 其他 Linux 远程连接统信 UOS 57
4.2.4 Windows 远程连接统信 UOS 59
4.3 邮件服务器 60
4.3.1 概述 60
4.3.2 安装 61
4.3.3 配置 Postfix 邮件服务器 61
4.3.4 功能测试 62
4.4 域名服务器 63
4.4.1 概述 63
4.4.2 环境准备 64

4.4.3 配置过程 64
4.4.4 功能测试 65
4.5 DHCP 服务器 66
4.5.1 概述 66
4.5.2 环境准备 66
4.5.3 安装过程 67
4.5.4 配置过程 67
4.5.5 功能测试 68
4.6 Web 服务器-Apache 68
4.6.1 概述 68
4.6.2 安装过程 68
4.6.3 配置过程 69
4.6.4 功能测试 69
4.7 Web 服务器-Nginx 70
4.7.1 概述 70
4.7.2 安装过程 70
4.7.3 配置过程 70
4.7.4 功能测试 70
4.8 关系型数据库服务器-MySQL 71
4.8.1 概述 71
4.8.2 配置环境 71
4.8.3 安装过程 72

4.8.4 功能测试 72
4.9 关系型数据库服务器-PostgreSQL 74
4.9.1 概述 74
4.9.2 安装过程 74
4.9.3 功能测试 75
4.10 关系型数据库服务器-MariaDB 76
4.10.1 概述 77
4.10.2 配置环境 77
4.10.3 安装过程 78
4.10.4 功能测试 78
4.11 非关系型数据库服务器-Redis 80
4.11.1 概述 80
4.11.2 Redis 安装 80
4.11.3 Redis 启动 81
4.11.4 Redis 配置 81
4.11.5 Redis 连接 83
4.11.6 Redis 连接命令 84
4.11.7 Redis 键（key） 84
4.11.8 Redis keys 命令 85
4.11.9 Redis 字符串（String） 86
4.11.10 Redis 哈希（Hash） 88
4.11.11 Redis 列表（List） 89

4.11.12 Redis 集合（Set） 91
4.11.13 Redis 事务 92
4.11.14 Redis 脚本 94
4.11.15 Redis 服务器 94
4.11.16 参考 97
4.12 国密算法 97
4.12.1 概述 97
4.12.2 目的 97
4.12.3 术语说明 97
4.12.4 使用方法 99
4.13 关系型数据库服务器-OpenGauss 101
4.13.1 简介 101
4.13.2 安装 101
4.13.3 基本用法 103
4.13.4 数据库连接 105
4.13.5 创建和管理表 108
4.13.6 其他操作 114
4.13.7 注意事项 118
4.13.8 参考 118
4.14 分布式数据库-MongoDB 118
4.14.1 简介 118
4.14.2 安装 118

4.14.3 启动 119
4.14.4 配置实例 120
4.14.5 连接实例 120
4.14.6 数据库基本操作 121
4.15 Haproxy 负载均衡 130
4.15.1 简介 130
4.15.2 安装 131
4.15.3 配置 131
4.15.4 功能使用 133
4.16 大数据-Flink 139
4.16.1 简介 139
4.16.2 部署 139
4.16.3 访问 web 页面 140
4.16.4 查看日志 140
4.16.5 运行实例 140
4.16.6 停止集群 144
4.17 运维工具 AWX 144
4.17.1 概述 144
4.17.2 安装 AWX 144
4.17.3 使用 AWX 146
4.18 系统智能性能调优工具 A-Tune 153
4.18.1 概述 153

4.18.2 支持业务模型 153
4.18.3 使用限制 153
4.18.4 安装 A-Tune 154
4.18.5 配置 A-Tune 154
4.18.6 启动 A-Tune 160
4.18.7 启动 A-Tune engine 160
4.18.8 使用方法 161
4.18.9 常见问题 179
5 服务器安全管理 180
5.1 PAM 180
5.1.1 配置文件 180
5.1.2 模块类型 180
5.1.3 控制标记 181
5.1.4 模块路径 181
5.1.5 常用模块介绍 181
5.2 iptables 188
5.2.1 概述 188
5.2.2 功能测试 188
5.2.3 安全规则 189
5.2.4 相关命令 190
5.3 Nmap 190
5.3.1 概述 190

5.3.2 安装 191
5.3.3 功能测试 191
5.4 Auditd 191
5.4.1 概述 191
5.4.2 启动 191
5.4.3 配置文件 191
5.5 sysctl 192
5.5.1 概述 192
5.5.2 功能测试 192
6 服务器运维工具 193
6.1 sysstat 193
6.1.1 概述 193
6.1.2 安装 sysstat 工具 193
6.1.3 配置文件 193
6.1.4 命令介绍 193
6.1.5 功能测试 196
6.2 top 197
6.2.1 概述 197
6.2.2 命令介绍 198
6.2.3 功能测试 198
6.3 iotop 200
6.3.1 概述 200

6.3.2 命令介绍 200
6.3.3 功能测试 201
6.4 lsof 201
6.4.1 概述 201
6.4.2 命令介绍 202
6.4.3 功能测试 202
6.5 psmisc 202
6.5.1 概述 202
6.5.2 命令介绍 203
6.5.3 功能测试 203
6.6 procps 203
6.6.1 概述 203
6.6.2 命令介绍 204
6.6.3 功能测试 204
6.7 e2fsprogs 205
6.7.1 概述 205
6.7.2 命令介绍 205
6.7.3 功能测试 206
6.8 strace 206
6.8.1 概述 206
6.8.2 命令介绍 206
6.8.3 功能测试 207

6.9 ltrace 207
6.9.1 概述 207
6.9.2 命令介绍 207
6.9.3 功能测试 208
6.10 dmidecode 208
6.10.1 概述 208
6.10.2 安装过程 209
6.10.3 功能测试 209
6.11 cockpit 209
6.11.1 概述 209
6.11.2 安装 Cockpit 209
6.11.3 使用 Cockpit 209
6.11.4 功能模块 211
6.12 ansible 219
6.12.1 概述 219
6.12.2 安装部署的前提条件 219
6.12.3 安装主控端 220
6.12.4 安装受控端 220
6.12.5 Ansible 配置 220
6.12.6 Ansible 的使用 220
6.12.7 Modules 的使用 221
6.12.8 附录 226

6.13 x11vnc 229
6.13.1 概述 229
6.13.2 安装部署 229
7 AppStream 使用指南 231
7.1 AppStream 介绍 231
7.1.1 基本概念 232
7.1.2 模块和流三者之间的关系 232
7.2 AppStream 常用命令 232
7.2.1 相关命令 232
8 虚拟化管理 237
8.1 概述 237
8.2 KVM+Libvirt+QEMU 虚拟化 237
8.2.1 前提条件 237
8.2.2 安装虚拟化组件 237
8.2.3 虚拟机 XML 配置 239
8.2.4 环境准备 244
8.2.5 虚拟机管理 245
8.2.6 登录虚拟机 246
8.2.7 虚拟机其他操作 248
8.3 KVM+Virt-Manager+QEMU 虚拟化 259
8.3.1 前提条件 259
8.3.2 安装虚拟化组件 259

8.3.3 上传镜像 259
8.3.4 添加连接 259
8.3.5 （可选）创建虚拟网络 262
8.3.6 创建镜像存储文件 265
8.3.7 创建虚拟机 268
9 启用额外软件源 276
9.1 启用 OpenStack 源（1060e） 277
9.1.1 安装 OpenStack 源 277
9.1.2 刷新元数据缓存 277
9.2 启用 CDH 源（1060e） 277
9.2.1 安装 CDH 源 277
9.2.2 刷新元数据缓存 277
9.3 启用国密源 277
9.3.1 安装 GM 源 278
9.3.2 刷新元数据缓存 278
10 FAQ 278
10.1 输入法常见问题 278
10.2 虚拟机 XML 文件 278
10.2.1 AMD 架构的虚拟机 XML 文件（Legacy） 278
10.2.2 AMD 架构的虚拟机 XML 文件（UEFI） 280
10.2.3 ARM 架构的虚拟机 XML 文件（UEFI） 283
10.3 A-Tune 常见问题 284

10.3.1 train 命令训练模型出错，提示“training data faild” 284
10.3.2 atune-adm 无法连接 atuned 服务 285
10.3.3 atuned 服务无法启动，提示“Job for atuned.service failed because a timeout was exceeded.” 285
10.4 模块化安装常见问题 285
10.4.1 llvm-toolset 无法安装 285
10.4.2 rust-toolset 无法安装 287
10.5 时间同步服务器使用问题 288
10.6 无密码帐户锁定问题 288
10.7 重启 dbus 服务导致桌面系统无法使用问题 289
10.8 在飞腾环境中使用 vi 命令查看/var/log/messages 时，出现中文字符乱码问题 289
10.9 系统常见问题 289
10.9.1 firebird 用户登录 shell 问题 289
10.9.2 如何正确设置 selinux 模式 290
10.10 perl 模块使能导致依赖 perl 的非模块化包安装失败问题 291

1 欢迎使用

本章将简要介绍统信服务器操作系统（简称“统信 UOS”）和基础操作。

1.1 概要

统信服务器操作系统是一个基于 Linux 内核的完整而紧密的服务器操作系统。统信服务器操作系统开发人员所做的工作包括：Web 和 FTP 站点管理、图形设计、软件许可协议的法律分析、编写文档、bugfix、软件生态扩充、维护软件包等。统信服务器操作系统的开发人员还参与了许多其它计划。有些是专注于统信服务器操作系统的，还有些则是面向 Linux 社区。

1.2 GNU/Linux

Linux 是一种计算机操作系统，是一系列能让您与计算机进行交互操作并运行其它程序的程序。

操作系统由多种基础程序构成。它们使计算机可以与用户进行交流并接受指令，读取数据或将其写入硬盘、磁带或打印机，控制内存的使用，以及运行其它软件。操作系统最重要的组成部分是内核。在 GNU/Linux 系统中，Linux 就是内核组件，而该系统的其余部分主要是由 GNU 工程编写和提供的程序组成。因为单独的 Linux 内核并不能成为一个可以正常工作的操作系统，所以我们更倾向使用“GNU/Linux”一词来表达人们通常所说的“Linux”。

Linux 是以 Unix 为原型创造的。自诞生之日起，它就被设计成一种多任务、多用户的系统。这些特点使 Linux 完全不同于其它著名操作系统。事实上，Linux 比您能想象到的更特别，后来演变为 GNU/Linux 系统的开发工作始于 1984 年。当时，自由软件基金会开始研发被称为 GNU 的自由的类 Unix 操作系统。

GNU 工程开发了大量用于 Unix™的自由软件工具和类 Unix 操作系统，例如 Linux。这些工具使用户能执行从日常俗事（例如复制和删除文件）到神秘操作（例如书写和编译程序或对多种文档格式进行熟练的编辑工作）的各种任务。

虽然有许多组织和个人都对 Linux 的发展作出了帮助，但是自由软件基金会依然是最大的单个贡献者。他不仅仅创造了绝大部分在 Linux 中使用的工具，还为 Linux 的存在提供了理论和社会基础。

Linux 内核的首次面世是在 1991 年。当时，名为 Linus Torvalds 的芬兰计算机科学系学生在 Usenet 新闻组 comp.os.minix 上发布了一种 Minix 替代内核的早期版本。（请自行查阅 Linux International 的相关历史）。

Linux 很少会崩溃，适合在同一时间运行多个程序，而且比大多数操作系统更为安全。有了这些优势，Linux 成为在服务器市场上增长最快的操作系统。近来，它还开始在家庭和商业用户中变得越来越流行。

1.3 统信服务器操作系统

将统信服务器操作系统的哲学与方法论、GNU 工具集、Linux 内核，以及其他重要的自由软件结合在一起，构成的独特的软件发行版称为统信服务器操作系统。该发行版由大量的软件包组成，发行版中的每个软件包都包含了执行文件、脚本、文档和配置信息。

统信服务器操作系统注重高质、稳定和灵活。可以方便地使之应用到各种场合，从精简的防火墙到服务器科学工作站，甚至高端网络服务器都可以轻松胜任。

统信服务器操作系统拥有自主的软件包管理系统。这些工具给予统信服务器操作系统的系统管理员对安装到系统上的软件包的完全控制，包括安装单个软件包和自动升级整个操作系统。个别软件包也可以被保护不被升级，甚至可以告诉包管理系统哪些软件是自己编译的以及它们所需要的依赖关系。

为了提防“特洛伊木马”和其他恶意软件，更好地保护您的系统，统信服务器操作系统会校验统信服务器操作系统的注册维护人员所上传的软件包。统信服务器操作系统的开发人员也会特别注意以安全的方式配置软件包。依靠统信服务器操作系统的简易更新选项，安全更新可以通过互联网自动下载和安装。

可以通过统信软件技术有限公司（简称“统信软件”）官方网站的“**联系我们**”页面获取技术人员联系方式，以寻求技术支持。

1.4 登录

统信服务器操作系统启动后，支持在终端使用用户名密码登录；也支持远程登录，系统默认支持通过 SSH 方式使用远程客户端登录到服务器。如远程服务器为统信服务器操作系统（以 IP 地址为 192.168.100.20 为例进行介绍，具体值请以实际为准），本地客户端为统信服务器操作系统，执行远程登录的命令如下：

```
[uos-server@localhost ~]# ssh uos-server@192.168.100.20
UOS Server 20
uos-server@192.168.100.20's password:

Authorized users only. All activities may be monitored and reported.
Activate the web console with: systemctl enable --now cockpit.socket
Last login: Mon Nov  2 09:45:29 2020 from 192.168.100.6

Welcome to 4.19.140-2009.4.0.0048.up1.uel20.aarch64
System information as of time:  Sat Jul  4 16:52:20 CST 2020

System load:       0.01
```

```
Processes:          176
Memory used:         15.0%
Swap used:           0.5%
Usage On:            44%
IP address:         192.168.100.20
Users online:       4

[uos-server@localhost ~]#
```

说明：

- *uos-server 为远程服务器上的管理员用户，远程登录也需要输入登录用户的密码，验证通过后，可登录远程服务器。*
- *本文中 192.168.100.20 和 192.168.100.6 地址只作为示例（下同）。*
- *如果统信服务器操作系统未安装 SSH（默认已安装 SSH），需要先安装 SSH。打开终端命令窗口，输入以下命令即可完成 SSH 的安装。*

```
[uos-server@localhost ~]$ sudo dnf install openssh

我们信任您已经从系统管理员那里了解了日常注意事项。
总结起来无外乎这三点：

    #1) 尊重别人的隐私。
    #2) 输入前要先考虑（后果和风险）。
    #3) 权力越大，责任越大。

[sudo] uos-server  的密码：
```

1.5 注销

logout 命令可让用户退出系统，其功能和 login 命令相互对应。打开终端命令窗口，执行 **logout** 命令，即可注销当前用户。

1.6 关机

- 命令格式：shutdown [选项] [参数] [参数]
- 常用选项

表 1.1 shutdown 常用选项

选项	描述
-k	并不会真的关机，只是将警告讯息传送给所有使用者。

选项	描述
-r	关机后重新开机。
-h	等效于--poweroff，除非指定了--halt。
-P	--poweroff：关闭机器电源。
-H	--halt：停机。
-c	取消目前已经进行中的关机动作。
--help	显示帮助信息。
--no-wall	停机/关机/重启前不发送任何消息。

■ 执行示例

- ◆ **shutdown** -h *now*　　//立即关机。
- ◆ **shutdown** -h *10*　　//指定 10 分钟后关机。
- ◆ **shutdown** -r *now*　　//立即重启计算机。
- ◆ **shutdown** -r *+3*　　//指定三分钟后重启计算机，并提示消息。
- ◆ **shutdown** -c　　//取消正在进行的关机操作。

1.7 重启

■ 命令格式：reboot [OPTIONS...] [ARG]

■ 常用选项

表 1.2 reboot 常用选项

选项	描述
-p	关机。
-w	不要停机/关机/重启，仅把记录写入“**/var/log/wtmp**”文件。
-d	不把记录写入“**/var/log/wtmp**”文件。
-f	强制立即停机/关机/重启。
--halt	停止机器。
--no-wall	停机/关机/重启前不发送任何消息。

说明：停机是指关闭正在运行的操作系统，而不关闭硬件。

■ 执行示例：终端执行命令 **sudo reboot** 重启系统。

```
[uos-server@localhost ~]$ sudo reboot
我们信任您已经从系统管理员那里了解了日常注意事项。
总结起来无外乎这三点：

    #1) 尊重别人的隐私。
    #2) 输入前要先考虑(后果和风险)。
```

```
    #3) 权力越大，责任越大。

[sudo] uos-server 的密码：
```

1.8 显示用户名

whoami 命令用于显示当前用户名称，相当于执行 **id** -un 指令。

■ 命令格式：whoami [参数]

■ 常用选项

表 1.3 whoami 常用选项

选项	描述
--help	显示帮助信息。
--version	显示版本信息。

■ 执行示例

```
[uos-server@localhost ~]$ whoami
uos-server
```

1.9 切换用户

su 命令用于切换用户，除当前使用用户为 root 外，其他用户切换时都需要键入当前用户的密码。

■ 命令格式：su [选项] [-] [<用户> [<参数>...]]

■ 常用选项

表 1.4 su 常用选项

选项	描述
-g	指定主要组，此参数仅对 root 用户可用。
-h	显示帮助信息。
-V	显示版本信息。
-m	变更身份时，不要变更环境变量。
-w	不重置指定的变量。
-,-l	使用户登录本身 shell 环境。
-c	向 shell 传递命令。

■ 执行示例

◆ root 切换至 uos-server。

```
[root@localhost ~]# whoami              //显示当前用户
root
[root@localhost ~]# su - uos-server     //切换到 uos-server
```

```
[uos-server@localhost ~]$ whoami     //查询当前用户，确认切换成功
uos-server
```

◆ uos-server 切换至 root。

```
[uos-server@localhost ~]$ su - root
密码：
[root@localhost ~]# whoami
root
```

2 系统配置

本章节所有命令均以 root 用户身份登录系统进行操作。

2.1 时间调整

统信服务器操作系统中存在系统时间与硬件时间两个概念。硬件时间是指 BIOS 中的时间，称为 RTC 时间，可以执行 **hwclock** 命令来查看和设置硬件时间。系统时间是 Linux 内核的一部分，由定时器中断驱动。是自 1970 年 1 月 1 日 00:00:00 UTC 以来的秒数（或更简洁地说，是自 1969 UTC 以来的秒数）。可以执行 **date** 命令查看和修改，如果需要同步可以执行 **chrony** 命令同步远端服务器的时间。

2.1.1 hwclock

■ 命令格式：hwclock [功能] [选项...]

■ 常用选项

表 2.1 hwclock 常用选项

选项	描述
-r	显示 RTC 时间。
-h	显示帮助信息。
-v	显示版本信息。
-w	将系统系统时间设置为 RTC 时间。
-f	使用/dev/rtc0 备用文件设置 rtc 时间
-s	将 RTC 时间设置为系统时间。

■ 执行示例

```
[root@localhost ~]# hwclock -s
[root@localhost ~]# hwclock -w
```

2.1.2 date

■ 命令格式：date [选项]... [+格式]

■ 常用选项

表 2.2 date 常用选项

选项	描述
--help	显示帮助信息。
--version	显示版本信息。
-u	按照 UTC（协调世界时）显示或设置时间。
-s	按照给定字符串描述的时间来设置时间。
-r	显示指定文件的最后修改时间。

■ 执行示例

```
[root@localhost home]# touch test.file
[root@localhost home]# date
2020 年  07 月  04 日  星期六  19:10:15 CST
[root@localhost home]# date -r test.file
2020 年  07 月  04 日  星期六  19:10:12 CST
[root@localhost home]#
```

2.1.3 chrony

1 执行如下命令，修改“**/etc/chrony.conf**”文件添加时间同步服务器列表。

```
echo "server ntp.ntsc.ac.cn iburst" >> /etc/chrony.conf
```

2 执行如下命令，重启 chrony 服务。

```
systemctl restart chronyd
```

3 查看 NTP 服务工作是否正常。

```
chronyc sources
```

```
[root@localhost ~]# chronyc sources
210 Number of sources = 4
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? time.cloudflare.com           3   6     3     1  +2415us[+2415us] +/-   97ms
^? ntp8.flashdance.cx            2   6     3     0   +882us[ +882us] +/-  115ms
^? 119.28.206.193                2   6     1     2   +544us[ +544us] +/-   44ms
^? ntp1.flashdance.cx            2   6     1     2   +892us[ +892us] +/-   98ms
[root@localhost ~]# chronyc --help
Usage: chronyc [-h HOST] [-p PORT] [-n] [-c] [-d] [-4|-6] [-m] [COMMAND]
```

图 2.2 可访问的 NTP 服务器列表

2.1.4 tzselect

以 root 用户身份登录系统，在终端界面，执行 **tzselect** 命令调整时区。

■ 命令格式：tzselect [选项]

■ 常用选项

表 2.3 tzselect 常用选项

选项	描述
-c	从最大城市最接近地理坐标 COORD 位置的时区中进行选择，COORD 应使用 ISO 6709 表示法。例如巴黎的'-c +4852+00220'（以度和分表示，北和东），或'-c -35-058' 表示布宜诺斯艾利斯（度数、南部和西部）。
-n	使用-c 选项时显示最多 LIMIT 个位置（LIMIT 默认值为 10）。
--version	显示版本信息。
--help	显示帮助信息。

■ 执行示例

tzselect

tzselect -c *+4852+00220*

*说明：执行完 **tzselect** 命令，通过对应的数字选择城市。确认要选择的城市后，会生成这个城市对应的时区。通过 export 添加到环境变量使环境变量生效，需要永久生效，需要将时区变量添加到 profile 中。有关如何添加环境变量的方法请自行查阅，此处不做赘述*

2.2 用户管理

用户管理常用的功能包括：添加用户、修改用户、删除用户、密码管理、添加用户组、修改用户组、删除用户组。

2.2.1 添加用户

useradd 命令用来创建用户信息。

■ 命令格式：useradd [选项] <参数>

■ 常用选项

表 2.4 useradd 常用选项

选项	描述
-c	comment：给新用户添加备注。
-d	home_dir：为 HOME 目录指定一个名字（如果不想用登录名作为主目录名的话）。
-m	创建用户的 HOME 目录。
-M	不创建用户的 HOME 目录（当默认设置里指定创建时，才用到）。
-N	不创建与用户同名的组。
-r	创建系统帐户（由系统使用，区别于个人账户）。
-p	passwd：为用户帐户指定默认密码。

选项	描述
-s	shell：指定默认登录的 shell。
-u	uid：为帐户指定一个唯一的 UID。
-U	--user-group：创建与用户同名的组，并将用户添加到组中。

■ 执行示例

◆ 执行 **useradd** *testuser1* 命令，添加用户 testuser1。

◆ 执行 **useradd** -m *testuser2* 命令，创建用户主目录。

◆ 执行 **useradd** -s */bin/bash testuser1* 命令，指定用户登录后使用的 shell。

◆ 执行 **useradd** -U *testuser2* 命令，创建一个和用户同名的组，并将用户添加到组中。

说明：

■ 执行 **less** */etc/passwd* 命令，可以查看新增加用户的信息。

■ *执行* ***cat*** */etc/shells 命令，可以查看系统安装了哪些 shell。*

2.2.2 修改用户

usermod 命令可用来修改用户帐号的各项设定。

■ 命令格式：usermod [选项] <参数>

■ 常用选项

表 2.5 usermod 常用选项

选项	描述
-a	--append：把用户追加到某些组中，仅与-G 一起使用。
-d	--home：用户的新 HOME 目录。如果同时使用-m 选项，则会将当前 HOME 目录的内容移动到新 HOME 目录中，如果新 HOME 目录不存在，则创建。
-l	--login：用户的名称将会从原用户名称修改为新用户名称。
-G	--groups GROUP1[,GROUP2,...[,GROUPN]]]：用户属于的附加组列表。组之间使用逗号隔开，没有空格。
-u	--uid UID：设置用户 ID 的新数值。

■ 执行示例

◆ 执行“**usermod** -aG *testgrp，testuser*”命令，且多个组之间逗号格隔开，把 testuser 用户加入 testgrp 组。可以执行 **id** *testuser 命令，*查看是否成功追加到组。

◆ 执行 **usermod** -l *testuser testuser1* 命令，修改 testuser1 的用户名为 testuser。

- 执行 usermod -d /home/testuser4 -m testuser 命令，修改 testuser 用户的 HOME 目录为/home/testuser*4* 。
- 执行 **usermod** -u *10002 testuser* 命令，修改用户 testuser 的 UID 为新值。

2.2.3 删除用户

■ 命令格式：userdel [选项] <参数>

■ 常用选项

表 2.6 userdel 常用选项

参数	描述
-h	--help：显示帮助信息。
-r	--remove：用户 HOME 目录中的文件将随用户 HOME 目录和用户邮箱一起删除，在其它文件系统中的文件需要手动搜索并删除。
-Z	--selinux-user：为用户删除所有的 SELinux 用户映射。

■ 执行示例

- 执行 **userdel** *testuser1* 命令，将删除 testuser1 相关的用户数据文件，如/etc/passwd、/etc/group、/etc/shadow 等都将被删除，但不删除 home 目录及文件。
- 执行 **userdel** -r *testuser1* 命令，删除用户 HOME 目录及其它相关文件，home 目录及文件也将被删除。

⚠ *注意：请不要轻易用-r 参数，它会在删除用户的同时删除用户所有的文件和目录。如果用户目录下有重要的文件，切记在删除前请备份。*

2.2.4 密码管理

■ 命令格式：passwd [选项] <参数>

■ 常用选项

表 2.7 passwd 常用选项

选项	描述
-l	锁定密码，即禁用帐号。
-u	解锁指定用户的密码。
-d	删除用户密码，这是禁用一个用户密码的快速方法。

■ 执行示例

```
[root@localhost ~]# passwd test                    #修改 test 这个用户的密码。
更改用户 test 的密码 。
新的 密码：
```

```
重新输入新的 密码：
passwd：所有的身份验证令牌已经成功更新。
[root@localhost ~]# passwd -l test          #锁定 test 这个用户。
锁定用户 test 的密码 。
passwd: 操作成功
[root@localhost ~]# passwd -u test          #解锁 test 这个用户。
解锁用户 test 的密码。
passwd: 操作成功
```

*说明：passwd 调用的认证模块与“**/etc/pam.d/system-auth**”和“**/etc/pam.d/password-auth**”中的相关配置有关，配置详情请参见“常用模块介绍”章节中的“**pam_pwquality.so**”和“**pam_deepin_pw_check.so**”。*

2.2.5 添加用户组

- 命令格式：groupadd [选项] <参数>
- 常用选项

表 2.8 groupadd 常用选项

选项	描述
-o	--non-unique：允许创建有重复 GID 的组。
-g	GID：指定新用户组的组标识号。
-r	--system：创建一个系统用户组。

- 执行示例
  - 执行 **groupadd** *testgrp* 命令，添加用户组 testgrp。
  - 执行 **groupadd** -r *testgrp2* 命令，创建一个系统用户组。
  - 执行 **groupadd** -g *1110 testgrp2*命令 ,向系统中增加一个新组 testgrp2，同时指定新组的组标识号是 1110。
  - 执行 **useradd** -m -g **testgrp testuser1** 命令，创建用户 testuser1 并添加到 testgrp 组。

说明：

- *创建用户并添加到组中时，必须保证组已存在。*
- *执行 **less** /etc/group 命令，可查看用户组是否添加成功。*

2.2.6 修改用户组

- 命令格式：groupmod [选项] <参数>
- 常用选项

表 2.9 groupmod 常用选项

选项	描述
-g	GID：为用户组指定新的组标识号。
-n	修改用户组的名称。

■ 执行示例

◆ 执行 **groupmod** -n *newgrp testgrp* 命令，将用户组 testgrp 的名称修改成 newgrp。

◆ 执行 **groupmod** -g *111111 newgrp* 命令，修改用户组 newgrp 的用户组 ID 为 111111。

2.2.7 删除用户组

■ 命令格式：groupdel [选项] 参数

■ 常用选项

选项	描述
-R	chroot 到的目录。
-P	/etc/* 文件所在的前缀目录。
-f	强制删除。

■ 执行示例：执行 **groupdel** *testgrp* 命令，删除用户组 testgrp。

说明：当没有用户属于该组或用户组不是任何用户的主组的情况下，才可以删除该用户组，其它情况下均不能删除该用户组。

2.3 文件管理

统信服务器操作系统中磁盘文件系统的功能包括文件与目录管理、磁盘与文件系统管理、文件权限管理、文件的压缩与打包管理等。本节主要介绍基本文件管理相关的命令。

2.3.1 cd 命令

■ 命令格式：cd [选项] [参数]

■ 常用选项

表 2.10 cd 常用选项

选项	描述
-P	显示出确实的路径，而非使用链接（link）路径。
-L	使用物理目录结构而不跟随符号链接：在处理`..'之前解析 DIR 中的符号链接。
-e	如果使用了-P 参数，但不能成功确定当前工作目录时，返回非零的返回值。

选项	描述
-@	在支持拓展属性的系统上，将一个有这些属性的文件当作有文件属性的目录。

■ 执行示例

- ◆ 执行 **cd** */*命令，切换到根目录。
- ◆ 执行 **cd** *..*命令，切换到上级目录。
- ◆ 在任意路径下，执行 **cd** */root/testdir* 命令，使用绝对路径切换到 testdir 目录。
- ◆ 在“**/root**”路径下，执行 **cd** *./testdir/*命令，使用相对路径切换到 testdir 目录。
- ◆ 执行 **cd** ~命令，切换到当前用户的默认工作目录。

说明：路径可以是绝对路径或相对路径，绝对路径从/(根目录)开始，然后指定到所需的目录；相对路径从当前目录开始。

2.3.2 pwd 命令

pwd 命令将当前目录的全路径名称（从根目录）写入标准输出。全部目录使用/（斜线）分隔。第一个/表示根目录，最后一个目录是当前目录。

■ 命令格式：pwd [选项]

■ 常用选项

表 2.11 pwd 常用选项

选项	描述
-P	打印当前的物理路径，不带有任何的符号链接。

说明：在终端界面，执行 ***man*** *pwd 命令，可查看其他更多参数及其含义。*

■ 执行示例：执行 **pwd** 命令，查看目录的完整路径。

2.3.3 id 命令

id 命令可以显示真实有效的用户 ID（UID）和组 ID（GID）。UID 是对一个用户的单一身份标识，GID 则对应多个 UID。

■ 命令格式：id [选项] [参数]

■ 常用选项

表 2.12 id 常用选项

选项	描述
-g	--group：显示用户所属群组的 ID。
-G	--groups：显示用户所属附加群组的 ID。
-u	--user：显示用户 ID。

■ 执行示例

- ◆ 执行 **id** 命令，显示当前用户的用户名、UID 和该用户所属的所有组。
- ◆ 执行 **id** *testuser1* 命令，显示用户 testuser1 的用户名、UID 和该用户所属的所有组。
- ◆ 执行 **id** -g *testuser1* 命令，显示用户 testuser1 所属的组 ID。

2.3.4 ls 命令

ls 命令是最常用的命令之一。ls 命令是 list 的缩写，用来列出文件和目录，以及它们相关的信息，例如文件大小、所有权和修改时间。如果没有使用选项，执行 **ls** 命令会列出当前目录中包含的文件，并按字母顺序对它们进行排序。

■ 命令格式：ls [选项] [参数]

■ 常用选项

表 2.13 ls 常用选项

选项	描述
-a	显示所有文件及目录（也会列出以.开头的隐藏文件）。
-l	将文件名、文件型态、权限、拥有者、文件大小等资讯详细列出。
-p	将“/”指示符附加到目录名称后。
-t	按照修改时间排序，最新的在最前面。
-r	以当前排序方式进行倒序排序。
-S	按照文件大小排序，最大的在最前面。
-h	以易于阅读的格式输出文件大小（例如 1K、 234M、 2G 等）

■ 执行示例

- ◆ 执行 **ls** -lrt 命令，查看当前目录下所有文件的所有信息，并按照修改时间倒序排列。
- ◆ 执行 **ls** -p 命令，给输出的目录添加“/”。
- ◆ 执行 **ls** -l --color=auto 命令，给执行 ls 命令后输出的文件名添加颜色。
- ◆ 执行 **ls** -lrt */etc* 命令，查看“**/etc**”目录下所有文件的所有信息，并按照修改时间倒序排列。

2.3.5 mkdir 命令

mkdir 命令，是 make directories 的缩写，用于创建新目录，所有用户都可以使用此命令。

■ 命令格式：mkdir [选项] [参数]

■ 常用选项

表 2.14 mkdir 常用选项

选项	描述
-m	创建目录时直接配置目录的权限。
-p	递归创建目录。

■ 执行示例

◆ 执行 **mkdir** *testdir1* 命令，创建一个空目录 testdir1。

◆ 执行 **mkdir** -p *testdir2/dir2* 命令，创建多层目录 testdir2/dir2。

◆ 执行 **mkdir** -m *777 testdir3* 命令，创建目录 testdir3 并配置权限为 777。

2.3.6 rm 命令

rm 命令用于删除文件或目录。用户可以用 rm 命令删除不需要的文件或目录。该命令的功能为删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录均删除。对于链接文件（Linux 中类似 windows 下的快捷方式，详细信息请自行查询），只是断开了链接，原文件保持不变。

■ 命令格式：rm [选项] 文件或目录

■ 常用选项

表 2.15 rm 常用选项

选项	描述
-f	--force：就是 force 的意思，忽略不存在的文件，不会出现警告信息。
-i	--interactive：进行交互式删除，在删除前会询问使用者是否删除。
-r	-R，--recursive：递归删除列出的全部目录及子目录。

■ 执行示例

◆ 执行 **rm** -i **.log* 命令，删除当前目录下以.log 结尾的所有文件，并在删除前询问是否删除。

◆ 执行 **rm** -f *file*，强制删除文件。

◆ 执行 **rm** -rf *test1*，删除 test1 目录及子目录所有文件。

2.3.7 cp 命令

cp 命令用于将一个或多个源文件或目录复制到指定的文件或目录。它可以将单个源文件复制成一个指定文件名的具体的文件或一个已经存在的目录下。

■ 命令格式：cp [选项] [参数] [参数]

■ 常用选项

表 2.16 cp 常用选项

选项	描述
-f	强行复制文件或目录，不论目标文件或目录是否已存在。

选项	描述
-i	若目标文件（destination）已经存在时，在覆盖目标文件之前给出提示，要求用户确认是否覆盖，回答 y 时目标文件将被覆盖。
-l	不复制文件，只是生成链接文件。
-p	连同文件的属性一起复制过去，而非使用默认属性。
-r	递归持续复制，用于目录的复制行为。

■ 执行示例

◆ 执行 **cp** */root/testfile* .命令，复制文件 testfile 到当前目录。

◆ 执行 **cp** */root/testfile /home* 命令，复制文件 testfile 到目标目录。

◆ 执行 **cp** -r */root/testdir1/* /root/testdir2/*命令，复制 testdir1 下所有文件及文件夹到 testdir2 目录。

◆ 执行 **cp** -r -i */root/testdir1/* /root/testdir2/*命令，复制 testdir1 下所有文件及文件夹到 testdir2 目录 如果文件或目录存在 会先询问是否覆盖。

2.3.8 mv 命令

mv 命令是 move 的缩写，可以用来移动文件或者重命名文件。

■ 命令格式：mv [选项] [参数] [参数]

■ 常用选项

表 2.17 mv 常用选项

选项	描述
-f	强制的意思，如果目标文件已经存在，不会询问而直接覆盖。
-i	若目标文件已经存在时，就会询问是否覆盖。
-u	仅在源文件比目标文件新或目标文件不存在时，才执行移动操作。

■ 执行示例

◆ 执行 **mv** *testfile1 testfile2* 命令，修改 testfile1 的名称为 testfile2。

◆ 执行 **mv** -i *testfile /root/testdir/*命令，移动 testfile 到目标目录，如果文件已存在，会提示是否覆盖。

2.3.9 cat 命令

cat 命令主要用来查看文件内容、创建文件、合并文件和追加文件内容等，查看文件时默认从文件第一行开始显示内容。

■ 命令格式：cat [选项][参数]

■ 常用选项

表 2.18 cat 常用选项

选项	描述

选项	描述
-b	–number-nonblank：和-n 相似，只不过对于空白行不编号。
-n	–number：由 1 开始对所有输出的行编号。
-s	–squeeze-blank：当遇到有连续两行以上的空白行，就代换为一行的空白行。
-v	列出一些看不出来的特殊字符。

■ 执行示例

◆ 执行 **cat** */etc/host.conf* 命令，查看 host.conf 的内容。

◆ 执行 **cat** -n */etc/host.conf* 命令 ，从 1 开始对 host.conf 所有输出行编号。

2.3.10 more 命令

more 命令类似于 cat 命令，却比 cat 命令功能强大。它以全屏幕的方式按页显示文本文件的内容，支持 vi 中的关键字定位操作。

■ 命令格式：more [选项] [参数]

■ 常用选项

表 2.19 more 常用选项

选项	描述
-d	显示帮助，而不是响铃。
-f	统计逻辑行数而不是屏幕行数。
-p	不滚屏，清屏并显示文本。
-c	不滚屏，显示文本并清理行尾。
-u	屏蔽下划线。
-s	将多个空行压缩为一行。
-<数字>	指定每屏显示的行数。
+<数字>	从指定行开始显示文件。
+/<字符串>	从匹配搜索字符串的位置开始显示文件。

■ 快捷键

表 2.20 more 常用快捷键

键值	描述
space 或者 z	向下翻页。
b	向上翻页。
Enter	向下滚动 1 行。
=	显示当前行号。
v	用 vi 编辑器打开当前内容。

键值	描述
d	向下翻 K 行，默认 K=11。
q	退出 more。
h	显示快捷键帮助。

■ 执行示例：执行 **more** +100 *testfile* 命令，从 100 行开始显示 testfile 内容。

2.3.11 less 命令

less 工具也是对文件或其它输出进行分页显示的工具，是查看文件内容的工具，功能极其强大。less 的用法比起 more 更加的有弹性。在使用 more 的时候，我们并没有办法向前面翻，只能往后面看，但若使用 less，就可以使用“**pageup**”和“**pagedown**”等按键往前或往后翻看文件，更容易查看文件的内容。除此之外，在 less 里可以拥有更多的搜索功能，不止可以向下搜，也可以向上搜。

■ 命令格式：less [选项][参数]

■ 常用选项

表 2.21 less 常用选项

选项	描述
-e	当文件显示结束后，自动离开。
-f	强迫打开特殊文件，例如外围设备代号、目录和二进制文件。
-i	忽略搜索时的大小写。
-m	显示类似 more 命令的百分比。
-N	显示每行的行号。

■ 快捷键

表 2.22 less 常用快捷键

键值	描述
/字符串	向下搜索“字符串”的功能。
?字符串	向上搜索“字符串”的功能。
n	重复前一个搜索（与/或?有关）。
N	反向重复前一个搜索（与/或?有关）。
b	向后翻一页
d	向后翻半页。
h	显示帮助界面。
q	退出 less 命令。
u	向前滚动半页。
y	向前滚动一行。

键值	描述
空格键	滚动一页。
回车键	滚动一行。
pagedown	向下翻动一页。
pageup	向上翻动一页。

■ 执行示例

◆ 执行 **less** -N *testfile* 命令，查看 testfile，并显示行号。

◆ 执行 **less** *testfile* 命令，查看 testfile 文件内容。

◆ 执行 **ps** -ef | **less** 命令，ps 查看进程信息并通过 less 分页显示。

2.3.12 head 命令

head 命令用于查看文档的开始指定数量的字符块，默认显示文档的前 10 行，如果给定的文件不止一个，则在显示的每个文件前面加一个文件名标题。

■ 命令格式：head [选项] [参数]

■ 常用选项

表 2.23 head 常用选项

选项	描述
-c	--bytes=[-]NUM：显示每个文件的前 NUM 个字节数。
-n	--lines=[-]NUM：打印前 NUM 行。
-q	--quiet，--silent：不显示包含给定文件名的文件头。
-v	--verbose：总是显示包含给定文件名的文件头。

■ 执行示例

◆ 执行 **head** -n *5 testfile* 命令，显示 testfile 的前 5 行。

◆ 执行 **head** -c *20 testfile* 命令，显示 testfile 的前 20 个字节。

◆ 执行 **head** -n *-6 testfile* 命令，显示 testfile 除了最后 6 行的全部内容。

◆ 执行 **head** -v -n *3 testfile* 命令，显示 testfile 的前 3 行，并显示文件名。

2.3.13 tail 命令

tail 命令用于按照要求将指定文件的最后部分输出到标准设备，通常是终端。通俗讲来，就是把某个档案文件的最后几行显示到终端上。假设该档案有更新，tail 会自己主动刷新，确保您看到最新的档案内容。

■ 命令格式：tail [选项][参数]

■ 常用选项

表 2.24 tail 常用选项

选项	描述

选项	描述
-f	随文件增长即时输出新增数据。
-c	输出文件最后 K 个字节的内容。如果文件中有中文，可能会被截断。
-n	显示最后 k 行的内容。

■ 执行示例

- ◆ 执行 **tail** -f *testfile* 命令，监视文件 testfile 的内容。
- ◆ 执行 **tail** -n *15 testfile* 命令，查看文件 testfile 最后 15 行的内容。
- ◆ 执行 **tail** -c *100 testfile* 命令，查看文件 testfile 最后 100 个字节的内容。

2.3.14 file 命令

该命令用来识别文件类型，也可用来辨别一些文件的编码格式。它是通过查看文件的头部信息来获取文件类型，而非扩展名。命令所在路径：/usr/bin/file。

■ 命令格式：file [选项] [参数]

■ 常用选项

表 2.25 file 常用选项

选项	描述
-b	输出文件辨识结果时，不输出文件名称。
-c	详细显示指令执行过程，便于排错或分析程序执行的情形。
-f	列出文件列表中所列举文件的类型。
-i	输出 MIME 类型的字符串。
-z	查看压缩文件类型。
-v	--version，显示版本信息。

■ 执行示例

- ◆ 执行 **file** *testfile* 命令，查看 testfile 文件的类型。
- ◆ 执行 **file** -i *bbb.doc* 命令，查看 bbb.doc 文件 MIME 类型的字符串。
- ◆ 执行 **file** -z *test.tar.gz* 命令，查看压缩文件的类型。

2.3.15 chmod 命令

文件调用权限分为三级：文件拥有者、群组、其他。使用 chmod 命令可以控制文件如何被他人所调用。chmod 命令有两种用法，一种是包含字母和操作符表达式的文字设定法；另一种是包含数字的数字设定法。

文件或目录的访问权限有三组，可用字符-表示无权限、r 表示可读、w 表示可写、x 表示可执行。还可用数字赋予权限，0 表示无权限、1 表示可执行、2 表示可写、4 表示可读，然后将其所有权限的代表数字相加。数字属性的格式是 3 个从 0 到 7 的八进制数，其顺序是文件拥有者（u）、群组（g）和其他（o）。

■ 命令格式：chmod [选项] [模式] [参数]

■ 常用选项

表 2.26 chmod 常用选项

选项	描述
c	类似-v 选项，但仅在做出修改时进行报告。
f	不显示大多数错误消息。
v	输出各个处理的文件的诊断信息。
R	递归修改文件和目录。
u	user：表示该文件的拥有者。
g	group：表示与该文件的拥有者属于同一群体（group）的人。
o	other：表示其他人。
a	表示包括 ugo 三者。
s	在执行命令时，设置用户或组 ID。
t	限制删除标志或粘性位(linux 特殊权限)。
+	表示增加权限。
-	表示取消权限。
=	表示唯一设定权限。
r	表示可读。
w	表示可写。
x	表示可执行。
X	仅当文件是目录或已具有某些用户的执行权限时才执行/搜索。

■ 执行示例

◆ 执行 **chmod** a+x *testfile* 命令，将文件 testfilet 设为所有人皆可执行。

◆ 执行 **chmod** ug+w,o-w **testfile testfile1** 命令，将文件 testfile 与 testfile1 设为该文件拥有者、与其所属同一群体者可写，其他人不可写。

◆ 执行 **chmod** *777 testfile* 命令，将文件 testfile 设为所有人可读、可写、可执行。

◆ 执行 **chmod** ug-wx,o-rwx *testfile* 命令，取消文件 testfile 的文件拥有者、所属同一群体者可写和可执行的权限，取消其他人可读、可写、可执行的权限。

2.3.16 find 命令

find 命令是个使用频率比较高的命令，通常在系统特定目录下，查找具有某种特征的文件。该命令中查询条件使用逻辑运算符 and、or、not 来表示：

and：逻辑与，在命令中用“-a”表示。是系统缺省的选项，表示只有当所

给的条件都满足时，寻找条件才算满足。

or：逻辑或，在命令中用“-o”表示。该运算符表示所给的条件中有一个满足时，寻找条件才算满足。

not：逻辑非，在命令中用“!”表示。该运算符表示查找不满足所给条件的文件。

■ 命令格式：find [选项] [参数][参数]

■ 常用选项

表 2.27 find 常用选项

选项	描述
-print	将查找到的文件输出到标准输出。
-amin	n：在过去 n 分钟内被读取过的文件。
-anewer	file：比文件 file 更晚被读取过的文件。
-atime	n：在过去 n 天内被读取过的文件。
-cmin	n：在过去 n 分钟内被修改过的文件。
-cnewer	file：比文件 file 更新的文件。
-ctime	n：在过去 n 天内被修改过的文件。
-prune	忽略某个目录。
-perm	按执行权限来查找。
-empty	文件为空并且是常规文件或目录
-ipath	p，-path p：路径名称符合 p 的文件，ipath 会忽略大小写。
-name	name，-iname name：文件名称符合 name 的文件，iname 会忽略大小写。
-size	n：文件大小是 n 单位的文件。b 代表 512 位元组的区块，c 表示字元数，k 表示 kilo bytes，w 是二个位元组。
-type	c：文件类型是 c 的文件。如 d 表示目录、c 表示字型装置文件、b 表示区块装置文件、p 表示具名贮列、f 表示一般文件、l 表示符号连结、s 表示 socket、-pid n 表示 process id 是 n 的文件。

■ 执行示例

◆ 执行 **find** */etc* -name *'host*'* -print 命令，查找/etc 目录下前缀为 host 的文件。

◆ 执行 **find** */usr* -name *"out*"* -prune -o -name *"*.txt"* -print 命令，在 usr 目录及子目录中，查找不是 out 开头的.txt 结尾文件。

◆ 执行 **find** . -perm *755* -print 命令，在当前目录及子目录中，查找属主具有读写执行、其他具有读执行权限的文件。

- 执行 **find** / -size *+100M* -type f -print 命令，查找超过 100M、文件类型为一般文件的文件。
- 执行 **find** . -mtime *+2* -type f -name *'*tra*'* -print 命令 ，查找 2 天前被修改过的，并且含有 tra 字符的文件。

2.3.17 stat 命令

stat 命令用于以文字的格式显示 inode 的内容。

■ 命令格式：stat [选项][参数]

■ 常用选项

表 2.28 stat 常用选项

选项	描述
-f	不显示文件本身的信息，显示文件所在文件系统的信息。
-L	显示符号链接。
-c	文件权限。
-t	简洁模式，只显示摘要信息。

■ 执行示例

- 执行 **stat** -f */tmp/testfile* 命令，显示文件所在文件系统的信息。
- 执行 **stat** */tmp/testfile* 命令，显示文件的详细信息。
- 执行 **stat** -c *"%a" /etc/passwd* 命令，显示文件权限，%a 代表八进制格式的文件访问权限。

2.3.18 tar 命令

统信服务器操作系统下最常用的打包程序就是 tar 了，使用 tar 程序打出来的包我们常称为 tar 包，通常是以.tar 结尾的。生成 tar 包后，就可以用其它的程序来进行压缩。

■ 命令格式：tar [选项] [参数]

■ 常用选项

表 2.29 tar 常用选项

选项	描述
-A	新增压缩文件到已存在的压缩包。
-B	设置区块大小。
-c	建立新的压缩文件。
-d	记录文件的差别。
-r	向压缩归档文件末尾追加文件。
-u	更新原压缩包中的文件。

选项	描述
-x	从压缩的文件中提取文件。
-t	显示压缩文件的内容。
-z	支持 gzip 文件。
-j	支持 bzip2 文件。
-Z	支持 compress 文件。
-v	显示操作过程。
-l	只要不是所有链接都被输出就打印信息。
-k	保留原有文件不覆盖。
-m	不改变解压文件的修改时间。
-w	确认压缩文件的正确性。
-f	使用的归档文件名称。

■ 执行示例

- ◆ 执行 **tar** -cvf *homebak.tar /home* 命令，将 home 目录及子目录压缩成 tar 格式。
- ◆ 执行 **tar** -xvf *homebak.tar* 命令，将 homebak.tar 文件解压。
- ◆ 执行 **tar** -zcvf *homebak.tar.gz /home* 命令，将 home 目录及子目录压缩成 gzip 格式。
- ◆ 执行 **tar** -xzvf *homebak.tar.gz* 命令，将 homebak.tar.gz 解压。

2.3.19 gzip 命令

gzip 是个使用广泛的压缩程序，文件经它压缩过后，其名称后面会多出“**.gz**”的扩展名。

■ 命令格式：gzip [选项] [参数]

■ 常用选项

表 2.30 gzip 常用选项

选项	描述
-c	--stdout 或--to-stdout：把压缩后的文件输出到标准输出设备，不去变动原始文件。
-d	--decompress 或----uncompress：解开压缩文件。
-r	--recursive：递归处理，将指定目录下的所有文件及子目录一并处理。
-f	--force：强行压缩文件。不理会文件名称或硬连接是否存在以及该文件是否为符号连接。
-l	--list：列出压缩文件的相关信息。

选项	描述
-n	--no-name：压缩文件时，不保存原来的文件名称及时间戳记。
-N	--name：压缩文件时，保存原来的文件名称及时间戳记。
-q	--quiet：不显示警告信息。
-S	----suffix：更改压缩字尾字符串。
-t	--test：测试压缩文件是否正确无误。
-v	--verbose：显示指令执行过程。
-V	--version：显示版本信息。
-L	--license：显示版本与版权信息。
-num	用指定的数字 num 调整压缩的速度。-1 或--fast 表示最快压缩（低压缩比），-9 或--best 表示最慢压缩（高压缩比），系统缺省值为 6。

■ 执行示例

- ◆ 执行 **gzip** */tmp/testfile* 命令，将 testfile 压缩成 testfile.gz。
- ◆ 执行 **gzip** -dv *testfile.gz* 命令，解压 testfile.gz 文件。
- ◆ 执行 **gzip** -rv *test* 命令，递归压缩目录。
- ◆ 执行 **gzip** -drv *test* 命令，递归解压目录。

2.4 磁盘管理

2.4.1 df 命令

df 命令用于统计目前操作系统上文件系统磁盘空间的使用情况。

■ 命令格式：df [选项] [参数]

■ 常用选项

表 2.31 df 常用选项

选项	描述
-a	--all：包括伪、重复、不可访问的文件系统。
-B	--block-size={SIZE}：使用{SIZE}大小的 Blocks。
-h	--human-readable：使用人类可读的格式（预设值是不加这个参数的）。
-H	--si：很像-h，但是用 1000 为单位而不是用 1024。
-i	--inodes：显示 inodes 信息，而非块使用量。
-k	--kilobytes：就像是--block-size=1024。
-l	--local：只显示本机的文件系统。
-P	--portability：使用 POSIX 输出格式。
--sync	在获取使用信息之前调用 sync 命令，从内存中同步数据至磁盘。

选项	描述
-t	--type=TYPE：只显示 TYPE 类型的文件系统。
-T	--print-type：显示文件系统的形式。

■ 执行示例

- ◆ 执行 **df** -h 命令，显示目前磁盘空间和使用情况。
- ◆ 执行 **df** -ia 命令，列出各文件系统的 i 节点使用情况。
- ◆ 执行 **df** -t ext4 命令，显示 ext4 类型磁盘。

2.4.2 du 命令

du 命令用于显示目录或文件的大小。du 会显示指定的目录或文件所占用的磁盘空间。

■ 命令格式：du [选项][参数]

■ 常用选项

表 2.32 du 常用选项

选项	描述
-a	-all：显示目录中个别文件的大小。
-b	-bytes：显示目录或文件的大小时，以 byte 为单位。
-c	--total：除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和。
-D	仅取消引用命令行中列出的符号链接。
-h	--human-readable：以 K、M、G 为单位，提高信息的可读性。
-H	等效于--dereference-args。
-k	等效于--block-size=1k。
-l	--count-links：重复计算硬件连接的文件。
-L	解引用所有符号链接。
-m	等效于--block-size=1M。
-s	--summarize：仅显示总计大小。
-S	--separate-dirs：显示个别目录的大小时，并不包含其子目录的大小。
-x	--one-file-xystem：以一开始处理时的文件系统的目录为准，若遇上其它不同的文件系统的目录则略过。
-X	--exclude-from=<文件>：在<文件>指定目录或文件。
--exclude	--exclude=<目录或文件>：略过指定的目录或文件。

■ 执行示例

- 执行 **du** */tmp/testfile1 /tmp/testfile2* 命令，显示多个文件的大小。
- 执行 **du** -h */tmp/testfile1* 命令，显示 testfile 文件的大小。
- 执行 **du** -s 命令，显示当前目录空间的总大小。

2.4.3 fdisk 命令

fdisk 是一个创建和维护分区表的程序，用于磁盘添加、删除、转换分区等。

■ 命令格式：fdisk [选项] [参数]

■ 常用选项

表 2.33 fdisk 常用选项

选项	描述
-l	列出所有分区表。
-u	显示单位，“cylinders”柱面或“sectors”扇区。
-s	指定扇区显示的设备大小。

■ 命令说明

表 2.34 fdisk 操作命令

命令	描述
m	显示菜单和帮助信息。
a	活动分区标记/引导分区。
d	删除分区。
l	显示分区类型。
n	新建分区。
p	显示分区信息。
q	退出不保存。
t	设置分区号。
v	进行分区检查。
w	保存修改。
x	扩展应用，高级功能。

■ 执行示例

- 执行 **fdisk** -l 命令，查看当前分区情况。
- 执行 **fdisk** */dev/sda5* 命令，根据菜单操作说明进行操作。如输入 m，就会提示帮助信息；输入 d，就会删除分区。

⚠ *注意：删除分区时要小心，请看好分区的序号。如果您删除了扩展分区，扩展分区之下的逻辑分区都会删除，所以操作时一定要小心。如果知道自己操作错了，请不要惊慌，输入 q 不保存退出，千万不要输入 w 保存退出。*

2.5 任务管理

操作系统最重要的任务之一是使用户充分有效的利用系统资源，当系统资源在一定或有限的情况下，需要同时执行更多程序，高效地完成更多任务。

进程、作业、任务调度是操作系统的重要任务之一，本节主要介绍管理相关的常用命令，包括 ps、kill、fg/bg、nohup、crontab、重定向等。

2.5.1 ps 命令

要对进程进行监测和控制，首先必须要了解当前进程的情况，也就是需要查看当前进程，ps 命令就是最基本的进程查看命令。使用该命令可以确定有哪些进程正在运行以及运行的状态、进程是否结束、进程是否处于僵尸态、哪些进程占用了过多的资源等。总之，大部分信息都是可以通过执行该命令得到。ps 是显示进程的瞬间状态，并不动态实时显示；如果想对进程进行实时监控应该用 top 命令。统信服务器操作系统中进程有如下 5 种状态以及状态码：

1 运行（R）：正在运行或在运行队列中等待。

2 中断（S）：休眠中，受阻并在等待某个条件的形成或接受到信号。

3 不可中断（D）：收到信号不唤醒和不可运行，进程必须等待直到有中断发生。

4 停止（T）：收到 SIGSTOP、SIGSTP、SIGTIN、SIGTOU 信号后停止运行。

5 僵死（Z）：进程已终止，但进程描述符依然存在，直到父进程调用 wait4() 系统，调用后释放。

■ 命令格式：ps [选项]

■ 常用选项

表 2.35 ps 常用选项

选项	描述
-A	所有的进程均显示出来，与-e 具有同样的效用。
-a	显示现行终端机下的所有进程，包括其他用户的进程。
-u	以用户为主的进程状态。
-e	选择所有进程，与-A 相同。
x	通常与-a 这个参数一起使用，可列出较完整信息。
l	较长、较详细的将所查询 PID 的信息列出。
j	工作的格式（jobs format）。
f	以完整格式输出，包括命令行。

■ 执行示例

◆ 执行 **ps** -ef 命令，显示所有进程的信息以及执行的命令行。

◆ 执行 **ps** -u *root* 命令，显示用户 root 的进程信息。

◆ 执行 **ps** -ef | grep *java* 命令，查找并显示 java 进程的信息。

2.5.2 kill 命令

kill 是向进程发送信号的命令。如果不指定信号，则发送 SIGTERM(15)终止指定进程；如果无法正常终止该进程，可以向进程号发送 SKILL(9)信号，该信号将强制结束进程。

■ 命令格式：kill [选项] [参数]

■ 常用信号

表 2.36 kill 参数说明

选项	信号名	说明
0	EXIT	程序退出时收到该信息。
1	HUP	挂掉电话线或终端连接的挂起信号，这个信号也会造成某些进程在没有终止的情况下重新初始化。
2	INT	表示结束进程，但并不是强制性的，常用的“**Ctrl+C**”快捷键发出的就是该信号。
3	QUIT	退出。
9	KILL	杀死进程，即强制结束进程。
11	SEGV	段错误。
15	TERM	正常结束进程，是 kill 命令默认发送的信号。

■ 执行示例

◆ 执行 **kill** -9 *12345* 命令，强制杀死进程号为 12345 的进程。

◆ 执行 **kill** -KILL *12345* 命令，强制杀死进程号为 12345 的进程。

◆ 执行 **kill** -l 命令，显示信号列表。

2.5.3 fg/bg 命令

统信服务器操作系统提供了 fg 和 bg 命令 ,让管理员轻松调度正在运行的任务。fg 命令可以将后台中的命令调至前台继续运行，如果后台中存在多个命令，可以用 fg %jobnumber 将选中的命令调出。bg 命令可以将一个后台暂停的命令换成继续执行，如果后台中存在多个命令，可以用 bg %jobnumber 将选中的命令调出。

假设前台运行的一个程序需要很长的时间，但是需要处理其他的事情，可以按下键盘上的“**Ctrl+z**”快捷键，将此程序挂起，然后可以看到系统提示。

■ 命令格式：fg <作业号>；bg <作业号>

■ 执行示例

1 执行 jobs 命令，查看当前正在进行的任务。

```
[root@localhost ~]#jobs
[1]  运行中  tar cvf homebak.tar /home &
```

2 执行 **bg** *1* 命令，将作业号为 1 且在后台暂停的命令，变成继续执行。

3 执行 **fg** *1* 命令，将作业号为 1 调回到前台，继续执行。

*说明：执行 tar cvf homebak.tar /home 任务时，如果想要暂停，可以使用“**Ctrl+z**”快捷键。*

2.5.4 nohup 命令

nohup 命令用于运行由 Command 参数和任何相关的 Arg 参数指定的命令，忽略所有挂断（SIGHUP）信号。在注销后，要使用 nohup 命令运行后台中的程序，需添加&（表示“and”的符号）到命令的尾部。

■ 命令格式：nohup Command [Arg …] [&]

■ 执行示例：

1 执行 **nohup** *start.sh* &命令，将脚本程序在后台保持一直运行状态。

2 注销当前用户后，再次登录系统。

3 执行 **ps** -ef | grep *start.sh* 命令，显示脚本进程一直在运行中。

2.5.5 crontab 命令

crontab 命令用于定期执行程序。

■ 命令格式：crontab [选项] [参数]

■ 常用选项

表 2.37 crontab 常用选项

选项	说明
-u	user：用来设定某个用户的 crontab 服务，例如"-u demo"表示设备 demo 用户的 crontab 服务，此参数一般由 root 用户来运行。
-e	编辑某个用户的 crontab 文件内容。如果不指定用户，则表示编辑当前用户的 crontab 文件。
-l	显示某用户的 crontab 文件内容。如果不指定用户，则表示显示当前用户的 crontab 文件内容。
-r	从“**/var/spool/cron**”删除某用户的 crontab 文件。如果不指定用户，则默认删除当前用户的 crontab 文件。
-i	在删除用户的 crontab 文件时，给确认提示。

■ 常用时间对照表

表 2.38 crontab 时间定义说明

位置	说明	数值范围
第一个*	一小时当中的第几分钟（minute）	0~59
第二个*	一天当中的第几小时（hour）	0~23
第三个*	一个月当中的第几天（day）	1~31

位置	说明	数值范围
第四个*	一年当中的第几个月（month）	1~12
第五个*	一周当中的星期几（week）	0~7（0 和 7 都代表星期日）

■ 特殊符号

表 2.39 crontab 特殊符号说明

符号	说明
*（星号）	代表任何时间。比如第一个"*"就代表一小时中每分钟都执行一次的意思。
,（逗号）	代表不连续的时间。比如"0 8,12,16***命令"就代表在每天的 8 点 0 分、12 点 0 分、16 点 0 分都执行一次命令。
-（中杠）	代表连续的时间范围。比如"0 5 ** 1-6 命令"，代表在周一到周六的凌晨 5 点 0 分执行命令。
/（正斜线）	代表每隔多久执行一次。比如"*/10****命令"，代表每隔 10 分钟就执行一次命令。

■ 执行示例

- 执行 **crontab** -e 命令后，输入 5 5 * * 2 /sbin/shutdown -r now，系统会在每周二的凌晨 5 点 05 分重启一次。
- 执行 **crontab** -e 命令后，输入 30 3 1,10,15 * * /root/sh/autobak.sh，在每月 1 日、10 日、15 日的凌晨 3 点 30 分都定时执行日志备份脚本 autobak.sh。
- 执行 **crontab** -l 命令，查看定时任务。
- 执行 **crontab** -r 命令，删除定时任务。

2.5.6 重定向命令

■ 命令格式

表 2.40 重定向格式说明

重定向格式	说明
cmd > file	把标准输出重定向到一个新文件中。
cmd >> file	把标准输出重定向到一个文件中（追加）。
cmd 1 > file	把标准输出重定向到一个文件中。
cmd > file 2 > & 1	把标准输出和标准错误一起重定向到一个文件中。
cmd 2 > file	把标准错误重定向到一个文件中。
cmd 2 >> file	把标准错误输出重定向到一个文件中（追加）。
cmd >> file 2 > & 1	把标准输出和标准错误一起重定向到一个文件中(追加)。
cmd < file > file2	cmd 命令以 file 文件作为输入，以 file2 文件作为输出。

重定向格式	说明
cmd < file	cmd 命令以 file 文件作为输入。
cmd & m	把标准输出重定向到文件描述符 m 中。

■ 执行示例

◆ 执行 **ls** -l */usr/*> *ls.txt* 命令，将某个目录下所有文件及目录的列表保存在一个文本文件中。

◆ 执行 **ls** -l */usr/*>> *ls.txt* 命令，将某个目录下所有文件及目录的列表追加到一个文本文件中。

说明：使用>>可以在实现重定向时不覆盖原有内容，而是在文件末尾追加内容。

◆ 执行 **ls** */usr/notexist* 2 > *lserr.txt* 命令，将标准错误信息重定向到文件 lserr.txt 中。

◆ 执行 ls 1 > */dev/null* 2 > */dev/null* 命令，正常输出和错误信息都不显示，将把标准输出和标准错误都重定向到/dev/null。

2.6 环境变量

统信服务器操作系统是一个多用户的操作系统。当每个用户登录到系统后，都会有一个专用的运行环境。通常每个用户默认的环境变量都是相同的，用户可以对自己的运行环境进行自定义，其方法就是修改相应的系统环境变量。按变量的生存周期来划分，统信服务器操作系统变量可分为两类：

■ 永久的：需要修改配置文件，变量永久生效。

■ 临时的：使用 export 命令声明即可，变量在关闭 shell 时失效。

统信服务器操作系统中常用的环境变量有 PATH、HOME、LOGNAME 等。

■ PATH：指定的是命令的搜索路径。

■ HOME：指定的是当前用户的主目录。

■ LOGNAME：指定的是当前用户的登录名。

说明：除了以上常见的环境变量，部分应用程序在安装时也需要增加环境变量才能生效，比如 Java 使用的环境变量有 JAVA_HOME 和 CLASSPATH 等。设置环境变量可以参见“env 命令、export 命令、配置/etc/profile”3 种方式。

2.6.1 env 命令

统信服务器操作系统中，执行 **env** 命令可以显示当前用户的环境变量，还可以在指定环境变量下执行其他命令。

■ 命令格式：env [选项][参数]

■ 常用选项

表 2.41 env 常用选项

选项	描述
-0	以空字符而非新行符结束每一输出行。
-i	开始一个新的空的环境变量。
-u	从当前环境中撤销一个变量。

■ 执行示例

◆ 执行 **env** 命令，显示所有环境变量。

◆ 执行 **env** -i PATH=/root/testdir *./test.sh* 命令，设置执行 test.sh 脚本时的环境变量。

◆ 执行 **env** | egrep -i "user|mail|pwd|loginname"命令，查询变量。

*说明：终端执行 **echo** $HOME 命令，可以打印环境变量。*

2.6.2 export 命令

export 命令用于设置或显示环境变量，在 shell 中执行程序时，提供一组环境变量。export 可以新增、修改或删除环境变量，供后续执行的程序使用。

执行 export 命令定义变量后，该变量只在当前的 shell(BASH)或其子 shell(BASH)下是有效的。shell 关闭了，变量也就失效了，再打开新 shell 时就没有这个变量，需要使用的话还需要重新定义。

■ 命令格式：export [选项] [参数]

■ 常用选项

表 2.42 export 常用选项

选项	描述
-f	代表变量名称为函数名称。
-n	删除指定的变量。变量实际上并未删除，只是不会输出到后续指令的执行环境中。
-p	列出所有的 shell 赋予程序的环境变量。

■ 执行示例

◆ 执行 **export** -p 命令，查询所有的环境变量。

◆ 执行 **export** PATH=/usr/local/nginx/bin:$PATH 命令，修改环境变量。修改完成后，执行 **echo** $PATH 命令，查看环境变量是否设置成功。

2.6.3 配置/etc/profile

执行 **vi** 命令在文件“**/etc/profile**”文件中增加变量，该变量将会对统信服务器操作系统的所有用户有效，并且是“**永久的**”。操作步骤如下：

1 在终端执行如下命令。

vi */etc/profile*

2 在文件的最末段设置要配置的环境变量，如配置 java 环境变量。

export
PATH=$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH:$HOMR/bin

3 执行 **source** */etc/profile* 命令，立即生效。否则，需要重启操作系统。

*说明：如果配置的环境变量只想对当前用户有效，可以执行 **vi** 命令编辑当前用户主目录下的“**.bash_profile**”文件。*

2.7 网络管理

2.7.1 配置网络

在统信服务器操作系统中，一般用于配置网络接口的命令行工具包括：ifconfig、ifup、ifdown。

2.7.2 ifconfig 命令

管理员用户可以使用 ifconfig 命令查看网络设备状态信息，以及对网络接口进行配置。

- 命令格式：ifconfig [网络设备] [参数]
- 常用选项

表 2.43 ifconfig 常用选项

选项	描述
add <地址>	设置网络设备的 IP 地址。
del <地址>	删除网络设备的 IP 地址。
down	关闭指定的网络设备。
netmask <子网掩码>	设置网络设备的子网掩码。

- 执行示例
  - 执行 **ifconfig** 命令，显示网络设备信息。
  - 执行 **ifconfig** *eth1* down 命令，关闭网络接口 eth1。
  - 执行 **ifconfig** -a 命令，显示当前有效的全部接口信息。
  - 执行 **ifconfig** *eth1* add *192.168.1.88* netmask *255.255.255.0* 命令，设置网络接口 eth1 的 IP 地址和子网掩码。

2.7.3 ifup 命令

管理员用户可以使用 ifup 命令激活一个指定的网络接口。例如，执行 **ifup** *eth1* 命令，可激活网络接口 eth1。

2.7.4 ifdown 命令

管理员用户可以使用 ifdown 命令关闭一个指定网络接口。例如，执行 **ifdown** *eth1* 命令，可关闭网络接口 eth1。

2.7.5 nmcli 命令

- 显示网络管理器的整体状态。

  nmcli general status

- 查看主机名。

  nmcli general hostname

- 设置主机名。

  nmcli general hostname *new_name*

- 查看所有网络信息。

  nmcli connection show

- 查看指定接口信息。

  nmcli connection show *ens33*

- 显示所有活动连接。

  nmcli connection show --active

- 查看网络设备状态。

  nmcli device status

- 创建动态 IP 地址的网卡连接。

  nmcli connection add type *ethernet* con-name *ens37* ifname *ens37*

  说明：

  - *connection add：添加新的连接。*
  - *con-name：连接名。*
  - *type：设备类型。*
  - *ifname：接口名。*

- 创建固定 IP 地址的网卡连接。

  nmcli connection add type *ethernet* con-name *ens37* ifname *ens37* ipv4.addresses *192.168.38.142/24* ipv4.gateway *192.168.38.2* ipv4.method *manual*

- 重载配置文件使配置生效。

  nmcli connection reload

- 启动 ens37 网卡。

  nmcli connection up *ens37*

- 停止 ens37 网卡。

nmcli device disconnect *ens37*

- 给 ens37 网卡添加 DNS。

  nmcli connection modify *"static"* ipv4.dns *"202.131.124.4 202.131.124.5"*

  说明：static 为连接名。

- 删除网卡。

  nmcli connection show
  nmcli connection delete *网卡名称*

- 修改网卡的 IP 地址。

  nmcli connection modify *ens37* ip4.addresses *192.168.38.161/24*

  说明：ens37 为网卡名称

- 查看生成的配置文件。

  cat */etc/sysconfig/network-scripts/ifcfg-ethernet-ens37*

- 修改连接名称。

  nmcli connection modify *ethernet-ens37* con-name *ens37*

- 允许/禁止开机自动启动。

  nmcli connection modify *enp2s0* connection.autoconnect yes
  nmcli connection modify *enp2s0* connection.autoconnect no

- 给网卡增加 IP 地址或者删除额外的 IP 地址。

  nmcli connection modify *ens37* +ipv4.addresses *192.168.38.161/24*
  nmcli connection modify *ens37* -ipv4.addresses *192.168.38.161/24*

- 重启网络。

  nmcli connection reload
  systemctl restart NetworkManager

2.7.6 管理 IP 地址

统信服务器操作系统中，一般用于管理 IP 地址的命令行工具包括：ip、dhclient、route。

ip 命令

ip 命令用来显示或操作 Linux 主机的路由、网络设备、策略路由和隧道，是 Linux 下较新的功能强大的网络配置工具。

使用 ip 命令，只需一个命令，您就能很轻松地执行一些网络管理任务。相对于 ifconfig，ip 命令功能更强大。

统信服务器操作系统已经预装了 iproute 软件包（提供 ip 命令），可以轻松的管理 IP 地址。

■ 命令格式：ip [选项] [参数]...

表 2.44 ip 对象说明

对象	说明
link	网络设备。
address	设备上的协议（IP 或 IPv6）地址。
addrlabel	协议地址选择的标签配置。
route	路由表条目。
rule	路由策略数据库中的规则。

■ 常用选项说明

表 2.45 ip 常用选项说明

选项	说明
-V	-Version：显示版本信息。
-s	-stats，statistics：输出详细信息。
-h	-human，-human-readable：输出人类可读的统计信息和后缀。
-o	-oneline：将每条记录输出到一行，用 ‘\’ 字符代替换行符。

■ 执行示例

- 执行 **ip** addr show 或 **ip** address show 命令，显示网卡及配置的地址。
- 执行 **ip** addr add *192.168.100.20/24* dev *eth0* 命令，设置 IP 地址。24 表示子网掩码的长度是 24 位，换算成十进制是 255.255.255.0。
- 执行 **ip** addr del *192.168.100.20/24* dev *eth0* 命令，删除 IP 地址。
- 执行 **ip** link set *eth0* up 命令，启用被禁用的网卡 eth0。
- 执行 **ip** link set *eth0* down 命令，禁用网卡 eth0。
- 执行 **ip** -s link list 命令，显示详细的设备信息。
- 执行 **ip** route list 命令，显示核心路由表。

dhclient 命令

dhclient 命令用于动态获取或释放 IP 地址。

■ 命令格式：dhclient [选项][参数]

■ 常用选项

表 2.46 dhclient 常用选项

选项	说明
-p	指定 DHCP 客户端监听的端口号。
-d	总是以前台方式运行程序。
-r	释放 IP 地址。
-4/-6	支持 IPv4 或者 IPv6。

■ 执行示例

◆ 执行 **dhclient** -4 *eth0* 命令，指定支持 IPv4 协议。

◆ 执行 **dhclient** -r 命令，释放当前 IP 地址。

route 命令

route 命令用于显示和操作 IP 路由表。要实现两个不同的子网之间的通信，需要一台连接两个网络的路由器，或者同时位于两个网络的网关来实现。

*说明：直接执行 route 命令来添加路由，不会永久保存，当网卡重启或者机器重启之后，该路由就失效了。可以在“**/etc/rc.local**”中添加 route 命令来设置路由永久有效。*

■ 命令格式：route [选项] [参数]

■ 常用选项

表 2.47 route 常用选项

参数	描述
-n	--numeric：不解析名字。
-v	--verbose：显示详细的处理信息。
-F	--fib：显示发送信息。
-C	--cache：显示路由缓存。
add	添加一条新路由。
del	删除一条路由。
-net	目标地址是一个网络。
-host	目标地址是一个主机。
netmask	当添加一个网络路由时，需要使用网络掩码。
metric	设置路由跳数。
gw	路由数据包通过网关。注意，您指定的网关必须能够达到。

■ 执行示例

◆ 执行 **route** add -net *192.168.100.0* netmask *255.255.255.0* dev *eth0* 命令，添加一条到达 192.168.100.0 网络的路由，指定网络掩码为 255.255.255.0，数据包通过网络接口 eth0。

◆ 执行 **route** add -net *192.168.100.0* netmask *255.255.255.0* gw *192.168.100.1* 命令，添加一条到达 192.168.100.0 网络的路由，指定网络掩码为 255.255.255.0，数据包通过网关 192.168.100.1。

◆ 执行 **route** add -host *192.168.100.20* gw *192.168.100.1* 命令，所有去往 192.168.100.20 主机的数据包发往网关 192.168.100.1。

◆ 执行 **route** add default gw *192.168.100.1* 命令，添加一条默认网关，所有的数据包将被转发到 192.168.100.1。

- 执行 **route** add -net *224.0.0.0* netmask *240.0.0.0* dev *eth0* 命令，添加或设置网关。
- 执行 **route** del -net *192.168.100.0* netmask *255.255.255.0* 命令，删除路由记录。

2.7.7 网络诊断

ping 命令

ping 命令是常用的网络命令，也属于一个通信协议，是 TCP/IP 协议的一部分。使用 ping 命令可以检查网络是否连通，能帮助管理员分析和判定网络故障。

执行 ping 命令会使用 ICMP 传输协议发出要求回应的信息，若远端主机的网络功能没问题，就会回应该信息，因而得知该主机运作正常。ICMP 协议是 Internet 控制报文协议，是 TCP/IP 协议簇的一个子协议，用于在 IP 主机和路由器之间传递控制消息。

■ 命令格式：ping [选项] [参数]

■ 常用选项

表 2.48 ping 常用选项

选项	描述
-d	使用 Socket 的 SO_DEBUG 功能。
-c	<完成次数>：设置完成要求回应的次数。
-f	极限检测。
-i	<间隔秒数>：指定收发信息的间隔时间。
-I	<网络界面>：使用指定的网络接口送出数据包。
-l	<前置载入>：设置在送出要求信息之前，先行发出的数据包。
-n	只输出数值。
-p	<范本样式>：设置填满数据包的范本样式。
-q	不显示指令执行过程，开头和结尾的相关信息除外。
-r	忽略普通的 Routing Table，直接将数据包发送到远端主机上。
-R	记录路由过程。
-s	<数据包大小>：设置数据包的大小。
-t	<存活数值>：设置存活数值 TTL 的大小。

■ 执行示例

- 执行 **ping** *192.168.100.20* 命令，查看本机的网络状态。
- 执行 **ping** -i *3* -s *1024 www.baidu.com* 命令，设置发送周期和包大小。
- 执行 **ping** -c *2 www.baidu.com* 命令，收到两次包后自动退出。

◆ 执行 **ping** -i *0.5* -c *10 www.baidu.com* 命令 ，每 0.5 秒发送一次网络封装包并限制发送 10 次。

traceroute 命令

通过 traceroute 命令，我们可以知道信息从您的计算机到互联网另一端的主机是走的什么路径。当然，每次数据包由某一同样的出发点（source）到达某一同样的目的地（destination）走的路径可能会不一样，但大多数时候所走的路由是相同的。

traceroute 主要用于追踪网络数据包的路由途径，默认的数据包大小是 40Bytes，用户可自定义大小。

⚠ *注意：在统信服务器操作系统中，默认未集成 traceroute，需执行* ***dnf install*** *traceroute 命令安装。*

■ 命令格式：traceroute [选项] [参数]

■ 常用选项

表 2.49 traceroute 常用选项

选项	描述
-d	使用 Socket 层级的排错功能。
-f	设置第一个检测数据包的存活数值 TTL 的大小。
-F	设置勿离断位。
-g	设置来源路由网关，最多可设置 8 个。
-i	使用指定的网络界面送出数据包。
-I	使用 ICMP 回应取代 UDP 资料信息。
-m	设置检测数据包的最大存活数值 TTL 的大小。
-n	直接使用 IP 地址而非主机名称。
-p	设置 UDP 传输协议的通信端口。
-r	忽略普通的 Routing Table，直接将数据包送到远端主机上。
-s	设置本地主机送出数据包的 IP 地址。
-t	设置检测数据包的 TOS 数值。
-w	设置等待远端主机回报的时间。

■ 执行示例

◆ 执行 **traceroute** *www.baidu.com* 命令，显示到达目的地的数据包路由。

◆ 执行 **traceroute** -m *4 www.baidu.com* 命令，显示到达目的地的数据包路由，设置数据存活数值为 4。

2.7.8 域名管理

在统信服务器操作系统中，一般用于管理域名的命令行工具包括：host、

nslookup。

host 命令

host 是一个常用的 DNS 工具，经常用来查询域名、检查域名解析是否正确。

- 命令格式：host [选项] [参数] [参数]
- 常用选项

表 2.50 host 常用选项

选项	描述
-a	查询所有的信息。
-c	设置查询类型。
-C	查询完整的 SOA 记录。
-d | -v	显示详细过程。
-l	列表模式。
-t	选择查询类型：CNAME NS SOA SIG KEY AXFR。
-w	永久等待。
-W	设置等待超时。

- 执行示例：执行 **host** -a *www.baidu.com*，查看域名地址的详细信息。

nslookup 命令

nslookup 是一个用于查询 Internet 域名信息或诊断 DNS 服务器问题的工具，主要用来诊断域名系统的基础信息。

- 命令格式：nslookup [选项] [参数]
- 执行示例
  - 执行 **nslookup** *www.baidu.com* 命令，查询网站域名信息。
  - 执行 **nslookup** -q=mx *mail.qq.com* 命令，查看邮件服务器的信息。

2.7.9 netstat 命令

在统信服务器操作系统中，一般通过 netstat 命令来查看网络连接状态。当设置不同命令参数时，可以显示网络连接状态、路由表、接口状态、无效连接和多播成员，也可以用于查看某个服务是否运行。

- 命令格式：netstat [选项] [参数]
- 常用选项

表 2.51 netstat 常用选项

选项	描述
-a	all：显示所有选项，netstat 默认不显示 LISTEN 相关。
-t	tcp：仅显示 tcp 相关选项。

选项	描述
-u	udp：仅显示 udp 相关选项。
-n	拒绝显示别名，能显示数字的全部转化成数字。（重要）
-l	仅列出有在 Listen（监听）的服务状态。
-p	显示建立相关链接的程序名。
-r	显示路由信息、路由表。
-e	显示扩展信息，例如 UID 等。
-s	按各个协议进行统计。（重要）
-c	连续打印所选信息。

■ 执行示例

- ◆ 执行 **netstat** -a 命令，列出所有网路端口。
- ◆ 执行 **netstat** -i 命令，显示网卡列表。
- ◆ 执行 **netstat** -apu 命令，查看 udp 相关的程序。
- ◆ 执行 **netstat** -s 命令，显示网络统计信息。

2.7.10 网络工具

NetCat（简称 nc）是一个可读写 TCP 或 UDP 网络连接的工具，它可以读取和从命令行跨网络写入数据。它被设计成一个可靠的后端工具，能被其它的程序或脚本直接调用，也是一个功能丰富的网络调试和开发工具。

数据传输

netcat 提供了这样一种方法，您只需要创建一个 Chat 服务器，一个预先确定好的端口，便可以和客户端进行信息传输。具体步骤如下：

1 服务端输入：**nc** -l *8888* # 8888 为服务端设定的端口。

2 客户端输入：**nc** *192.168.100.20 8888* # 客户端连接需要输入服务端的 IP 地址及端口号。

3 客户端输入任何消息，服务端的 shell 终端会展示数据。

4 服务端回复任何消息，客户端的 shell 终端会展示数据。

文件传输

netcat 工具还可用来传输文件。假设，您想要将一个文件 file.txt 从 A 传到 B，A 或 B 都可以根据需要作为服务器或者客户端。以下示例中，以 A 作为服务器端，以 B 作为客户端。

1 服务端（A）输入：**nc** -l *8888* < */home/file.txt*

⚠ *注意：服务端需要指定需要传输的文件。*

2 客户端（B）输入：**nc** -n *192.168.100.20 8888* > */root/file.txt*

⚠ *注意：客户端需要指定保存文件的地址。*

目录传输

netcat 也可以传输目录。假设，在服务器 A 上，我们创建一个 tar 归档包并且通过-在控制台重定向它；然后使用管道，重定向给 netcat，netcat 可以通过网络发送它；在客户端 B 我们下载该压缩包通过 netcat 管道然后打开文件。

1 服务端（A）输入：**tar** -cvf - */tmp* | **nc** -l *8888*　　# 传送 tmp 目录。

2 客户端（B）输入：**nc** -n *192.168.100.20 8888* | **tar** -xvf -　# 接收 tmp 目录。

2.7.11 网络下载

在统信服务器操作系统中，一般用于网络下载的命令行工具包括：wget、curl。

wget 命令

wget 是一个非交互式的网络文件下载工具，需要在终端命令行下使用。它支持从网站下载软件或从远程服务器恢复/备份数据到本地服务器。wget 支持 HTTP、HTTPS 和 FTP 协议，还可以使用 HTTP 代理。

■ 命令格式：wget [选项][参数]

■ 常用选项

表 2.52 wget 常用选项

选项	描述
-a	在指定的日志文件中记录执行过程。
-A	指定要下载文件的后缀名，多个后缀名之间使用逗号隔开。
-b	以后台的方式运行 wget。
-B	设置参考的连接地址的基地地址。
-c	继续执行上次终端的任务。
-d	以调试模式运行指令。
-D	--domains=列表：逗号分隔的可接受的域名列表。
-e	作为文件“.wgetrc”中的一部分执行指定的指令。
-h	显示帮助信息。
-i	从指定文件获取要下载的 URL 地址。
-l	指定递归最大深度级别，默认最大深度级别为 5。
-L	仅关注相关链接。用于在没有任何干扰的情况下检索特定主页，即使是来自同一主机的主页。
-r	递归下载方式。
-nc	文件存在时，下载文件不覆盖原有文件。

选项	描述
-nv	下载时只显示更新和出错信息，不显示详细执行过程。
-q	不显示执行过程。
-v	显示详细执行过程。

■ 执行示例

◆ 下载网页。

wget *https://www.uniontech.com/*

◆ 限速下载。

wget --limit-rate=1K *https://www.uniontech.com/*

◆ 后台下载。

wget -b *https://www.uniontech.com/*

curl 命令

curl 是文件传输工具，它支持文件的上传和下载。

■ 命令格式：curl [选项] [参数]

■ 常用选项

表 2.53 curl 常用选项

选项	描述
-A	设置用户代理发送给服务器。
-b	cookie 字符串或文件读取位置。
-c	操作结束后把 cookie 写入到这个文件中。
-C	断点续转。
-D	把 header 信息写入到该文件中。
-e	来源网址。
-f	连接失败时不显示 http 错误。
-o	把输出写到该文件中。
-O	把输出写到该文件中，保留远程文件的文件名。
-r	检索来自 HTTP/1.1 或 FTP 服务器的字节范围。
-s	静默模式，只执行命令不输出任何东西。
-T	上传文件。
-u	设置服务器的用户和密码。
-#	进度条显示当前的传送状态。
-x	--proxy <host[:port]>：在给定的端口上使用 HTTP 代理。

■ 执行示例

- 执行 **curl** *https://www.uniontech.com* 命令，将网站内容显示在终端屏幕上。
- 执行 **curl** *https://www.uniontech.com* >> *test.html* 命令，重定向保存为 test.html。
- 执行 **curl** -o *test1.html https://www.uniontech.com* 命令，抓取网站页面内容并保存在文件中。
- 执行 **curl** -D *cookied.txt http://www.linux.com* 命令，保存 http 的 response 里面的 header 信息。

2.8 系统和服务管理

在统信服务器操作系统中默认使用 systemd 作为系统守护进程管理工具。systemd 并不是一个命令，而是一组命令，涉及到系统管理的方方面面。systemctl 是 systemd 的主命令，主要负责控制 systemd 系统和服务管理器，可用于查看系统状态和管理系统及服务。

2.8.1 Unit 分类

systemd 可以管理所有系统资源。不同的资源统称为 unit（单元），unit 一共分成 12 种。

表 2.54 systemd 单元分类

Unit 分类	描述
Service	系统服务，守护进程的启动、停止、重启和重载是此类 unit 中最为明显的几个类型。
Target	多个 unit 构成的一个组。
Device	硬件设备，此类 unit 封装一个存在于 Linux 设备树中的设备。每一个使用 udev 规则标记的设备都将会在 systemd 中作为一个设备 unit 出现。udev 的属性设置可以作为配置设备 unit 依赖关系的配置源。
Mount	文件系统挂载点，此类 unit 封装系统结构层次中的一个挂载点。
Automount	自动挂载点，此类 unit 封装系统结构层次中的一个自挂载点。每一个自挂载 unit 对应一个已挂载的挂载 unit。
Path	文件或路径，用于监控指定文件或目录变化，并触发其他 unit 运行。
Scope	不是由 Systemd 启动的外部进程。
Slice	进程组。
Snapshot	systemd 快照，可以切回 unit 的某个运行状态。

Unit 分类	描述
Socket	进程间通信的 socket，此类 unit 封装系统和互联网中的一个 socket。当下，systemd 支持流式、数据报和连续包的 AF_INET、AF_INET6、AF_UNIX socket，也支持传统的 FIFOs 传输模式。
Swap	swap 文件。
Timer	定时器。

*注意：每个配置单元都有一个对应的配置文件，系统管理员的任务就是编写和维护这些不同的配置文件，比如一个 MySQL 服务对应一个“**mysql.service**”文件。*

2.8.2 Unit 基础操作

- 执行 **systemctl** --help 命令，查看其他更多参数及其含义。
- 执行 **systemctl** --version 命令，检查您的系统中是否安装有 systemd 并查询当前安装的版本。
- 执行 **systemctl reboot** 命令，重启系统。
- 执行 **systemctl poweroff** 命令，关闭系统、切断电源。
- 执行 **systemctl halt** 命令，CPU 停止工作。
- 执行 **systemctl suspend** 命令，暂停系统。
- 执行 **systemctl hibernate** 命令，让系统进入冬眠状态。
- 执行 **systemctl hybrid-sleep** 命令，让系统进入交互式休眠状态。
- 执行 **systemctl rescue** 命令，启动进入救援状态（单用户状态）。

2.8.3 Unit 启动耗时

- 执行 **systemd-analyze** 命令，用于查看启动耗时。
- 执行 **systemd-analyze blame** 命令，查看每个服务的启动耗时。
- 执行 **systemd-analyze critical-chain** 命令，显示瀑布状的启动过程流。

```
[root@localhost ~]# systemd-analyze critical-chain
The time after the unit is active or started is printed after the "@" character.
The time the unit takes to start is printed after the "+" character.

multi-user.target @8.432s
└─rsyslog.service @8.386s +45ms
  └─network-online.target @8.384s
    └─NetworkManager-wait-online.service @5.142s +3.242s
      └─NetworkManager.service @5.020s +85ms
        └─network-pre.target @5.017s
          └─firewalld.service @3.711s +1.305s
            └─polkit.service @3.057s +651ms
              └─basic.target @3.055s
                └─sockets.target @3.055s
                  └─sssd-kcm.socket @3.054s
                    └─sysinit.target @3.052s
                      └─systemd-update-utmp.service @3.043s +8ms
                        └─auditd.service @2.997s +45ms
                          └─systemd-tmpfiles-setup.service @2.961s +34ms
                            └─import-state.service @2.910s +50ms
                              └─local-fs.target @2.909s
                                └─boot.mount @2.835s +73ms
                                  └─systemd-fsck@dev-disk-by\x2duuid-10a11199\x2d7c7e\x2d40ae\x2d871a\x2db46fd1123a61.se
                                    └─local-fs-pre.target @2.808s
                                      └─lvm2-monitor.service @688ms +1.182s
                                        └─dm-event.socket @594ms
                                          └─system.slice
                                            └─-.slice
```

图 2.3 systemd 启动过程流

2.8.4 Unit 查看

systemctl list-units 命令可以查看当前系统的所有 unit。

- 执行 **systemctl list-unit-files** 命令，列出所有可用单元。
- 执行 **systemctl list-units** 命令，列出所有运行中单元。
- 执行 **systemctl** --failed 命令，列出所有失败单元。
- 执行 **systemctl list-units** --all --state=inactive 命令，列出所有没有运行的单元。
- 执行 **systemctl list-units** --type=service 命令，列出所有正在运行的、类型为 service 的单元。

```
[root@localhost ~]# systemctl list-units --type=service
UNIT                                                                    LOAD   ACTIVE SUB     DESCRIPTION
atd.service                                                             loaded active running Job spooling tools
auditd.service                                                          loaded active running Security Auditing Service
chronyd.service                                                         loaded active running NTP client/server
crond.service                                                           loaded active running Command Scheduler
dbus.service                                                            loaded active running D-Bus System Message Bus
dracut-shutdown.service                                                 loaded active exited  Restore /run/initramfs on
getty@tty1.service                                                      loaded active running Getty on tty1
import-state.service                                                    loaded active exited  Import network configurat
iscsi-shutdown.service                                                  loaded active exited  Logout off all iSCSI sess
kdump.service                                                           loaded active exited  Crash recovery kernel arm
kmod-static-nodes.service                                               loaded active exited  Create list of required s
libstoragemgmt.service                                                  loaded active running libstoragemgmt plug-in se
lvm2-monitor.service                                                    loaded active exited  Monitoring of LVM2 mirror
lvm2-pvscan@8:2.service                                                 loaded active exited  LVM event activation on d
mcelog.service                                                          loaded active running Machine Check Exception L
NetworkManager-wait-online.service                                      loaded active exited  Network Manager Wait Onli
NetworkManager.service                                                  loaded active running Network Manager
nis-domainname.service                                                  loaded active exited  Read and set NIS domainna
polkit.service                                                          loaded active running Authorization Manager
rngd.service                                                            loaded active running Hardware RNG Entropy Gath
rsyslog.service                                                         loaded active running System Logging Service
smartd.service                                                          loaded active running Self Monitoring and Repor
sshd.service                                                            loaded active running OpenSSH server daemon
sssd.service                                                            loaded active running System Security Services
systemd-fsck@dev-disk-by\x2duuid-10a11199\x2d7c7e\x2d40ae\x2d871a\x2db46fd1123a61.service loaded active exited  File Sys
systemd-journal-flush.service                                           loaded active exited  Flush Journal to Persiste
systemd-journald.service                                                loaded active running Journal Service
systemd-logind.service                                                  loaded active running Login Service
systemd-random-seed.service                                             loaded active exited  Load/Save Random Seed
```

图 2.4 正在运行的类型是 service 的服务单元

2.8.5 Unit 状态

systemctl status 命令用于查看系统状态和单个 unit 的状态。

- 执行 **systemctl status** 命令，显示系统状态。
- 执行 **systemctl status** *dbus.service* 命令，显示单个 unit 的状态。
- 执行 **systemctl is-active** *dbus.service* 命令，查看某个 unit 是否运行。

```
[root@localhost ~]# systemctl is-active dbus.service
active
[root@localhost ~]# systemctl is-active application.service
inactive
```

2.8.6 Unit 管理

对于您来说，最常用的是或许是以下这些命令，用于启动或停止 unit（主要针对 service 类型的 unit）。

- 执行 **systemctl start** *apache.service* 命令，立即启动一个服务。

- 执行 **systemctl stop** *apache.service* 命令，立即停止一个服务。
- 执行 **systemctl restart** *apache.service* 命令，重启一个服务。
- 执行 **systemctl kill** *apache.service* 命令，杀死一个服务的所有子进程。
- 执行 **systemctl reload** *apache.service* 命令，重新加载一个服务的配置文件。
- 执行 **systemctl daemon-reload** 命令，重载所有修改过的配置文件。
- 执行 **systemctl enable** *httpd.service* 命令，使某服务自动启动。
- 执行 **systemctl disable** *httpd.service* 命令，使某服务不自动启动。

2.9 帮助手册

在统信服务器操作系统中自带一本联机使用的手册，以供用户在终端上查找。使用 man 命令可以调阅其中的帮助信息，非常方便和实用。

- 命令格式：man [选项] [参数]
- 示例：**man** *systemctl*

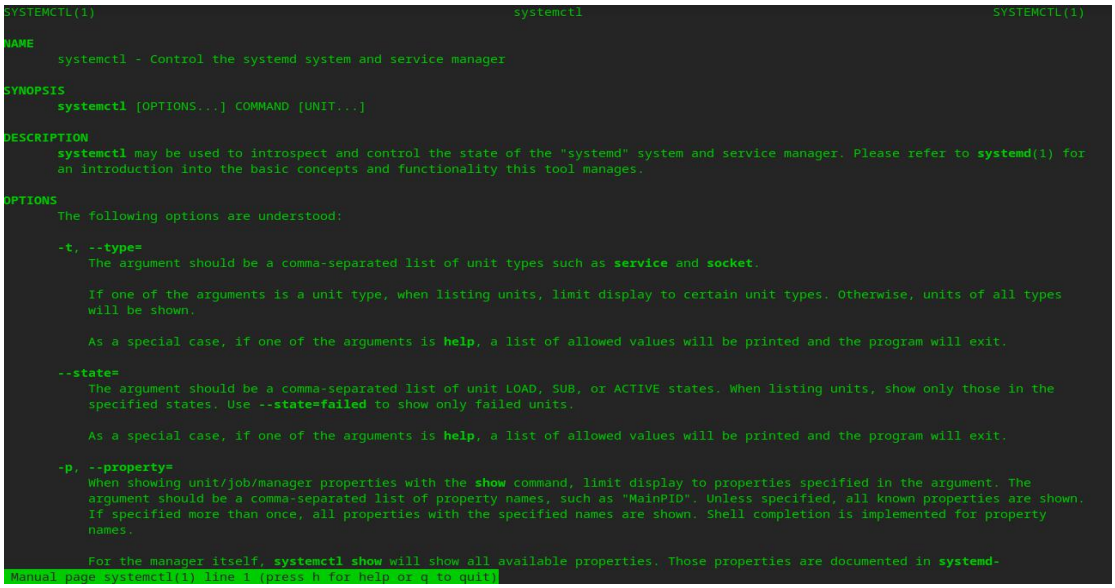


图 2.5 man 查看帮助手册示例

3 软件管理

在统信服务器操作系统中，一般用于管理软件包的命令行工具包括：dnf & yum 和 rpm。

3.1 dnf & yum

dnf 是一款 Linux 软件包管理工具，用于管理 rpm 软件包。dnf 可以查询软件包信息，从指定软件库获取软件包，自动处理依赖关系以安装或卸载软件包，以及更新系统到最新可用版本。dnf 与 yum 完全兼容，提供了 yum 兼容的命令行以及为扩展和插件提供的 API，以下为常见的命令。

表 3.1 dnf 常见命令及参数

命令及参数	说明
dnf install <package>	安装软件包。
dnf update <package>	更新已安装的软件包。
dnf remove <package>	删除软件包。
dnf check-update	检查更新。
dnf download <package>	下载软件包。
dnf info <package>	显示 RPM 包信息。
dnf list all	列出软件包清单。
dnf search	搜索软件包。
dnf repolist	显示相应软件源的配置。

说明：在终端界面，执行 ***dnf*** *--help 命令，可查看其他更多参数及其含义。*

3.2 rpm

rpm 是包安装、删除、编译和管理的工具，它可以用来安装、删除、构建和管理软件包。

- 命令格式：rpm [选项] [安装包]
- 常用选项

表 3.2 rpm 常用选项

选项	说明
-i	安装软件包。
-e	删除软件包。
-U	升级软件包。
-l	列出包中的文件。
-d	列出所有文本文件。
--nodeps	忽略软件包的依赖关系强行安装。
-f	查询/验证拥有文件的包。
--force	忽略软件包及文件的冲突。
-v	提供更详细的输出。
-h	在安装包时打印哈希标记（与-v 一起使用效果更好）。

- 执行示例
  - 执行 **rpm** -ivh *firewalld* 命令，安装 firewalld 包并显示安装进度。
  - 执行 **rpm** -Uvh *firewalld-*.rpm* 命令，升级 firewalld 版本。
  - 执行 **rpm** -e *firewalld* 命令，删除 firewalld 包。

*说明：在终端界面，执行 **rpm** --help 命令，可查看其他更多参数及其含义。*

3.3 which

在统信服务器操作系统中，使用 which 命令可以查看可执行文件的位置。

- 命令格式：which [选项] [参数]
- 执行示例
  - 执行 **which** *pwd* 命令，查看 pwd 的绝对路径。
  - 执行 **which** -a *ifconfig* 命令，查找所有 ifconfig 可执行文件的位置。

3.4 Update & Upgrade

3.4.1 配置软件源

在完成统信服务器操作系统的安装后，首先对软件源配置文件进行配置并升级。简单来说就是您安装软件时，程序从哪个仓库地址获取软件包，就对软件源配置文件进行相应的修改。

软件源的配置一般有两种方式，一种是直接配置“**/etc/dnf/dnf.conf**”文件，另一种是在“**/etc/yum.repos.d**”中增加 repo 文件，任选一种即可。

- 配置“**/etc/dnf/dnf.conf**”文件的“**main**”部分，配置示例如下：

```
[main]
gpgcheck=0
installonly_limit=3
clean_requirements_on_remove=True
skip_if_unavailable=False
best=True
```

表 3.3 常用参数说明

参数	说明
cachedir	缓存目录，该目录用于存储 RPM 包和数据库文件。
keepcache	可选值是 1 和 0，表示是否要缓存已安装成功的那些 RPM 包及头文件。默认值为 0，即不缓存。
debuglevel	设置 dnf 生成的 debug 信息。取值范围是[0-10]，数值越大就会输出越详细的 debug 信息。默认值为 2，设置为 0 表示不输出 debug 信息。
clean_requirements_on_remove	删除在 dnf remove 期间不再使用的依赖项。如果软件包是通过 dnf 安装的，而不是通过显式用户请求安装的，则只能通过 clean_requirements_on_remove 删除软件包，即它是作为依赖项引入的。默认值为 True。

参数	说明
best	升级包时，总是尝试安装最高版本，如果最高版本无法安装，则提示无法安装的原因并停止安装。默认值为 True。
obsoletes	可选值为 1 和 0，设置是否允许更新陈旧的 RPM 包。默认值为 1，表示允许更新。
gpgcheck	可选值为 1 和 0，设置是否进行 gpg 校验。默认值为 1，表示需要进行校验。
plugins	可选值为 1 和 0，表示启用或禁用 dnf 插件。默认值为 1，表示启用 dnf 插件。
installonly_limit	设置可以同时安装 “installonlypkgs” 指令列出包的数量。默认值为 3，不建议降低此值。

- 在 “**/etc/yum.repos.d**” 中增加 repo 文件：repository 部分允许您定义定制化的软件源仓库，各个仓库的名称不能相同，否则会引起冲突。下面是 repository 部分的一个最小配置示例：

```
[repository]
name=repository_name
baseurl=repository_url
```

表 3.4 repository 参数说明

参数	说明
name=repository_name	软件仓库（repository）描述的字符串。
baseurl=repository_url	软件仓库（repository）的地址。 ■ 使用 http 协议的网络位置，例如：http://path/to/repo。 ■ 使用 ftp 协议的网络位置，例如：ftp://path/to/repo。 ■ 本地位置，例如：file:///path/to/local/repo。

  - 统信服务器操作系统（版本为 10xxa 时）中有默认的 repo 文件 UniontechOS.repo，包含 AppStream、BaseOS、PowerTools、Plus、Update 等源 UnionTechOS.repo 文件中的 AppStream 配置示例如下：

```
[UniontechOS-$releasever-AppStream]
name = UniontechOS $releasever AppStream
baseurl=
https://enterprise-c-packages.chinauos.com/server-enterprise-c/kongzi/1060/AppStream/$basearch
```

```
enabled = 1
username=$auth_u
password=$auth_p
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-uos-release
gpgcheck = 0
skip_if_unavailable = 1
```

- 统信服务器操作系统（版本为 10xxe 时）中有 4 个默认的 repo 文件（以 ARM 架构为例），包含 UnionTechOS-aarch64.repo（标准源）、UnionTechOS-everything-aarch64.repo（全量源，包含所有软件包）、UnionTechOS-update-aarch64.repo（升级源）、UnionTechOS-modular-aarch64.repo（模块化源）。其中升级源和模块化源在系统初始安装后默认是关闭（enabled=0）状态。当有下一个版本发布后会同步标准源中的升级包到升级源路径，此时管理员只需要将升级源配置文件中的 enabled 值修改为 1，就可以进行软件包升级操作。当需要开启模块化源时，需要管理员手动将模块化源中的 enabled 值修改为 1。以 UnionTechOS-aarch64.repo 文件为例，配置示例如下：

```
[UnionTechOS-Server-20]
name=UnionTechOS-Server-20-$releasever
baseurl=https://euler-packages.chinauos.com/server-euler/fuyu/$releasever/OS/$basearch
gpgkey=https://euler-packages.chinauos.com/server-euler/fuyu/$releasever/OS/$basearch/RPM-GPG-KEY-UnionTech
enabled=1
gpgcheck=1
username=$auth_u
password=$auth_p
```

说明：

- *UnionTechOS 默认已经配置好了镜像源仓库，但需要激活授权后才能使用。*
- *UnionTechOS 默认配置中有认证相关的配置，请勿修改 repo 默认配置文件，否则会导致软件包更新失败，username 和 password 为认证所需要的用户名和密码。*

3.4.2 安装软件包

1 执行 **dnf install** *PACKAGE_NAME* 命令安装指定的软件包。以 wget 为例，执行 **dnf install** *wget* 命令，然后输入“**y**”确认，完成安装。

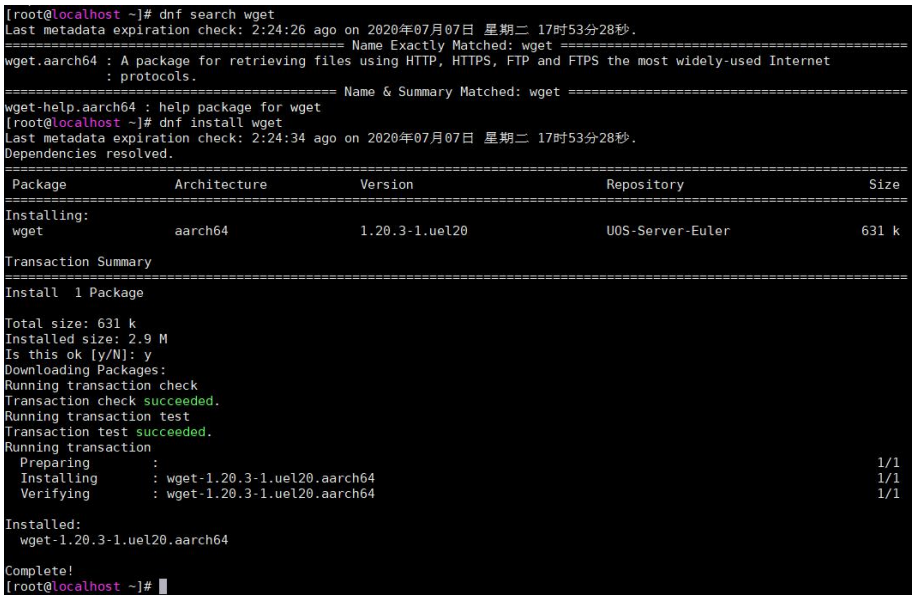


图 3.2 安装 wget 示例

2 执行如下命令，更新所有已安装的软件包。

```
dnf upgrade
```

*说明：更新软件包时，软件包将被下载到“**/var/cache/dnf/**”目录下缓存，然后再进行安装。升级的记录会写在日志文件/var/log/dnf.log、/var/log/dnf.librepo.log、/var/log/dnf.rpm.log 中。*

3.4.3 系统更新提示

通过 ssh 登录系统时，会给出用户更新提示，包括可更新软件包个数和更新命令，如图 3.3 所示。如果提示有软件包可更新，则可执行提示的命令进行更新。

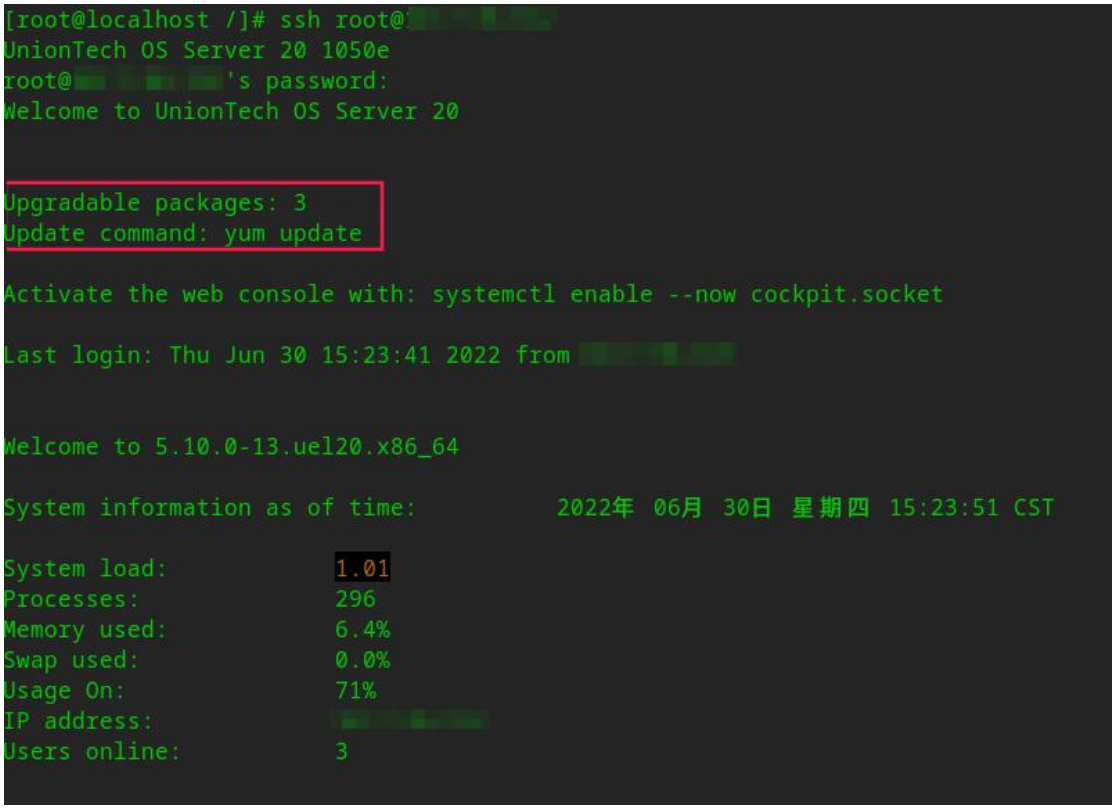


图 3.3 通过 ssh 登录时的系统更新提示

4 服务器软件

4.1 远程连接服务器

4.1.1 概述

OpenSSH 是 SSH（Secure Shell）协议的免费开源实现。SSH 协议族可以用来进行远程控制，或在计算机之间传送文件。OpenSSH 提供了服务端后台程序和客户端工具，用来加密远程控件和文件传输过程中的数据。

4.1.2 环境准备

- 测试机：测试机 1（ssh 服务器 192.168.100.20）和测试机 2（ssh 客户端），两台测试机需保持在同一网段，并且能够相互之间进行通讯。
- 网络测试：执行 **ping** *192.168.100.20* 命令，测试两台机器是否可以通讯。

4.1.3 SSH 安装

统信服务器操作系统默认支持通过 ssh 方式在客户端远程登录到服务器。客户端如果没有安装 OpenSSH，则需要先安装。

1 执行 **dnf install** *openssh-server* 命令，安装 openssh。

2 安装成功后，执行 **systemctl status** *sshd* 命令，查看 sshd 服务的状态。

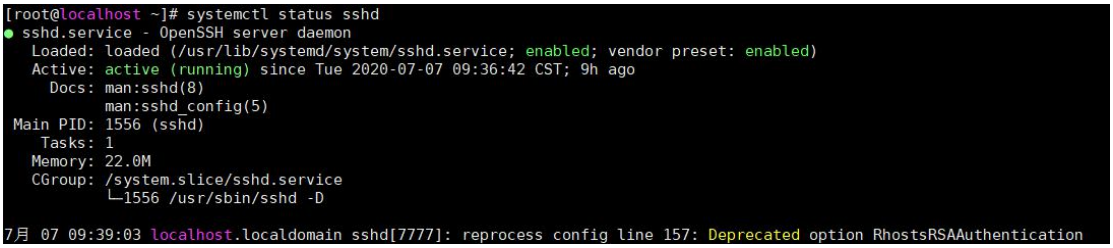


图 4.1 查看 sshd 服务的状态

4.1.4 配置文件

SSH 远程连接采用客户端/服务器模式，您可以根据个人需要配置 SSH 配置文件，如配置监听端口号等。

- SSH 服务器端配置文件为“**/etc/ssh/sshd_config**”。
- SSH 客户端配置文件为“**/etc/ssh/ssh_config**”。

4.1.5 功能测试

1 在客户端执行 **ssh** *uos@192.168.100.20* 命令，并输入密码，连接成功。

2 测试机 1：执行 **systemctl stop** *sshd* 命令，停止 ssh 服务。

3 测试机 2 执行 **ssh** *root@192.168.100.20* 命令 回显“**ssh: connect to host 192.168.100.20 port 22: Connection refused**”，说明服务断开。

4.2 远程连接桌面环境

4.2.1 概述

统信服务器操作系统集成了用于远程连接 Linux/Windows 的工具 FreeRDP 和 xrdp，可以很方便的连接 Linux/Windows 的桌面环境。FreeRDP 是一个免费开源实现的一个远程桌面协议（RDP）工具，用于从 Linux 远程连接到 Windows 的远程桌面。Xrdp 是一个微软远程桌面协议（RDP）的开源实现，它允许我们通过图形化界面控制远程系统。

4.2.2 统信 UOS 远程连接 Windows

1 执行如下命令，安装 freerdp 包。（在统信 UOS 端操作。）

yum install *freerdp* -y

2 按照下图中的提示，启用远程连接。（在 Windows 端操作。）

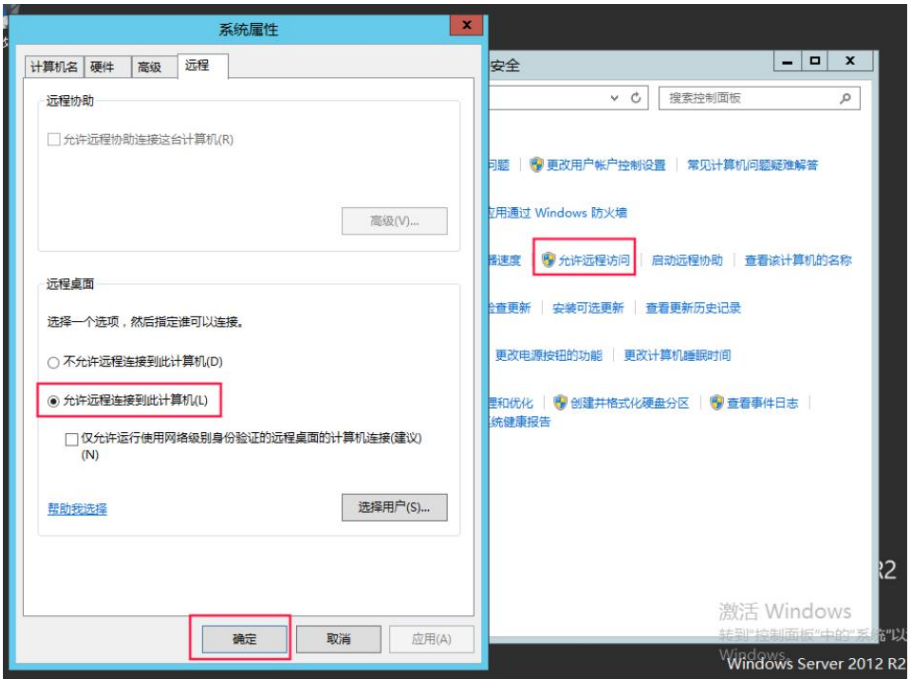


图 4.2 Windows 端启用允许远程连接

3 获取 Windows 的 IP 地址。（在 Windows 端操作。）

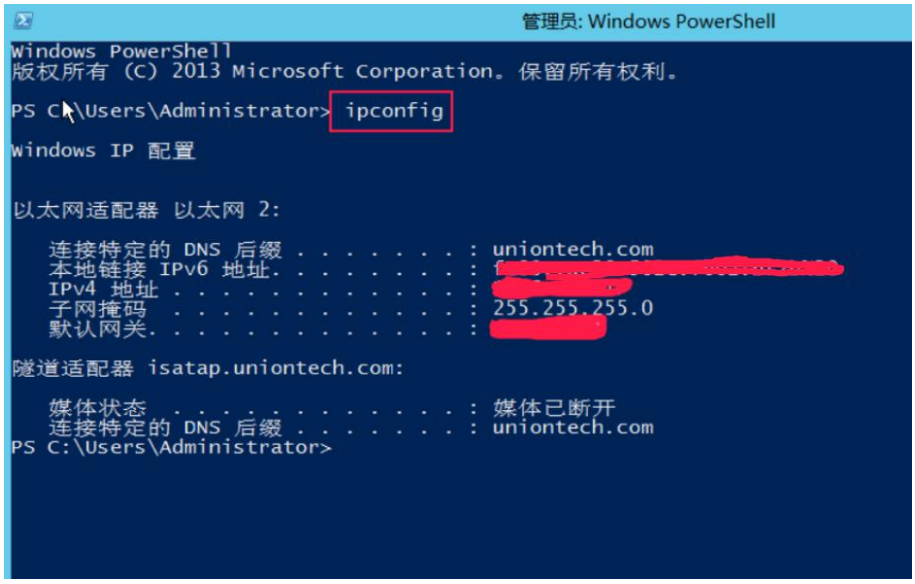


图 4.3 Windows 端查看 IP 地址

4 执行如下命令，远程连接到 Windows 桌面。（在统信 UOS 端操作。）

xfreerdp /v:Windows_IP /u:Account /p:Passwd

说明：

- *Windows_IP：Windows 的 IP 地址。*
- *Account：Windows 的登录用户名。*
- *Passwd：Windows 的登录密码。*

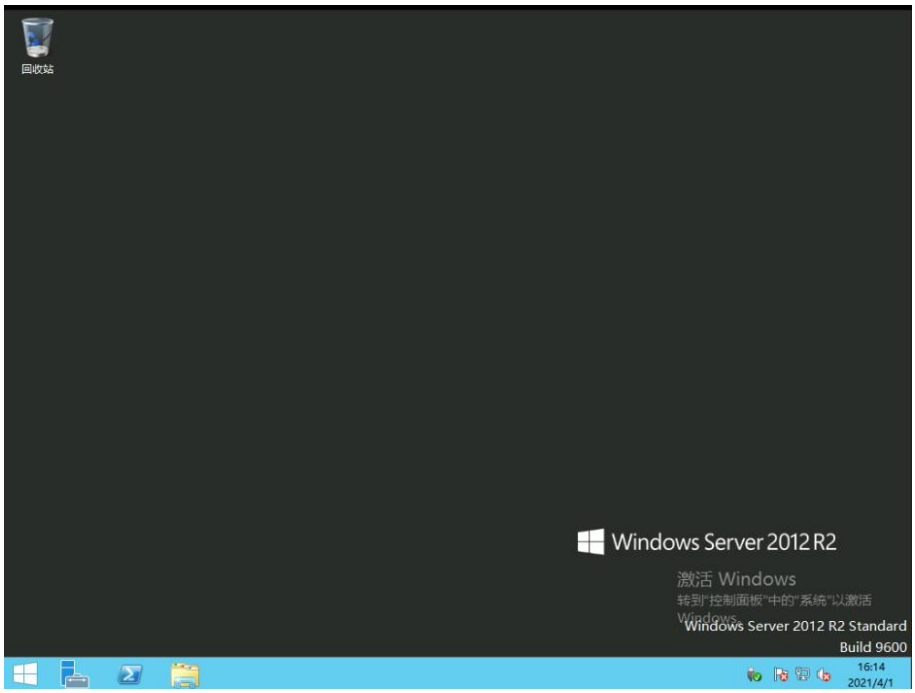


图 4.4 freerdp 远程连接 Windows 桌面成功

4.2.3 其他 Linux 远程连接统信 UOS

1 执行如下命令，安装 xrdp。（在统信 UOS 端操作。）

yum install *xrdp* -y

2 执行如下命令，启动服务。（在统信 UOS 端操作。）

systemctl start *xrdp*
systemctl start *x11vnc*

3 执行如下命令，关闭防火墙。（在统信 UOS 端操作。）

systemctl stop *firewalld*

4 执行如下命令，连接远程 Linux 桌面。（在其他 Linux 端操作。）

xrdp /v:Linux_Server_IP /u:Account

说明：

- *Linux_Server_IP：其他 Linux 的 IP 地址*
- *Account：其他 Linux 的登录用户名。*

5 执行成功后，会打开 vnc 连接配置界面，如图 4.5 所示。

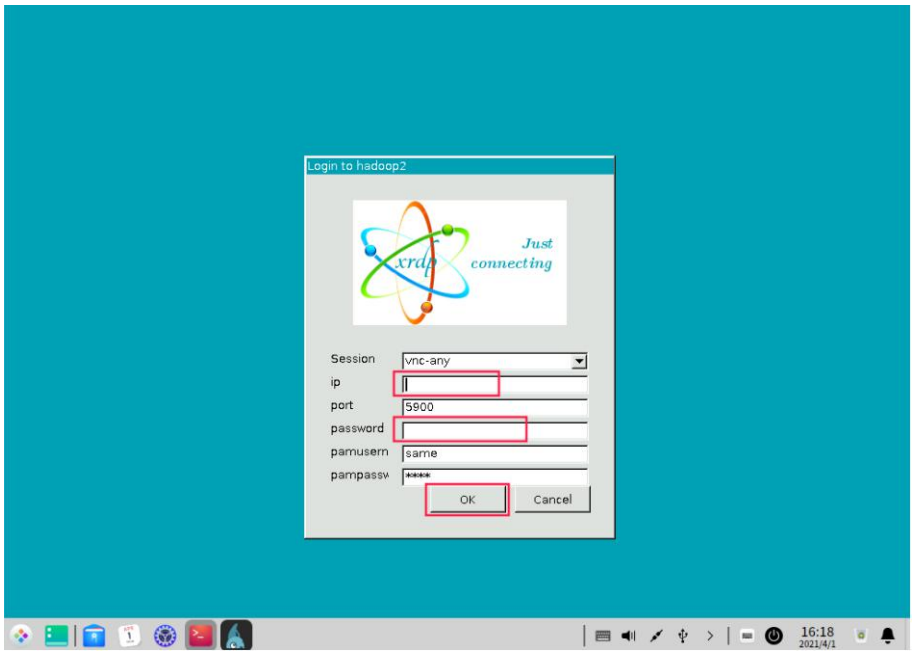


图 4.5 配置远程统信 UOS 的连接信息

说明：

- *Session：选择“**vnc-any**”。*
- *ip：服务端（统信 UOS）的 IP 地址。*
- *端口：服务端（统信 UOS）的实际端口，默认值为 5900。*
- *passwd：服务端（统信 UOS）登录用户的密码。*
- *pamusern：保持默认。*
- *pampassw：保持默认。*

6 按照说明进行配置后，可成功进行远程连接，如图 4.6 所示。

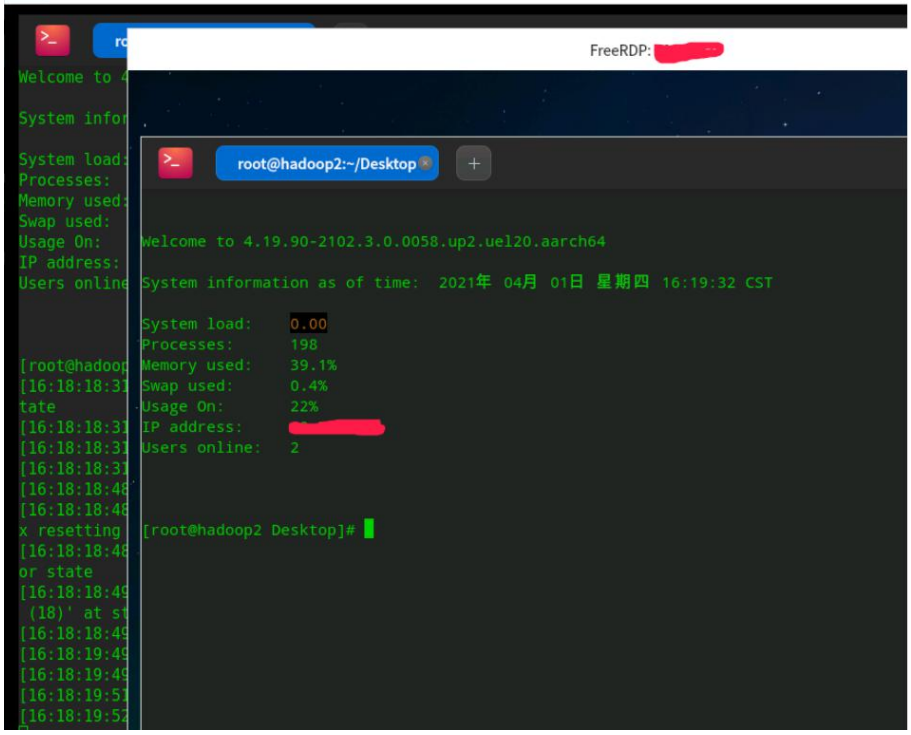


图 4.6 成功远程连接统信 UOS

4.2.4 Windows 远程连接统信 UOS

1 执行如下命令，安装 xrdp。（在统信 UOS 端操作。）

```
yum install xrdp -y
```

2 执行如下命令，启动服务。（在统信 UOS 端操作。）

```
systemctl start xrdp
systemctl start x11vnc
```

3 执行如下命令，关闭防火墙。（在统信 UOS 端操作。）

```
systemctl stop firewalld
```

4 按“**win+r**”快捷键，在弹出的窗口中输入 mstsc。（在 Windows 端操作。）

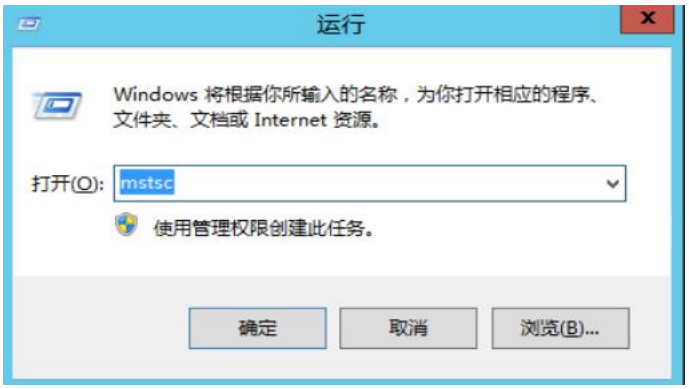


图 4.7 启动 windows 远程桌面程序

5 输入统信 UOS 的 IP 地址，单击“**连接**”。

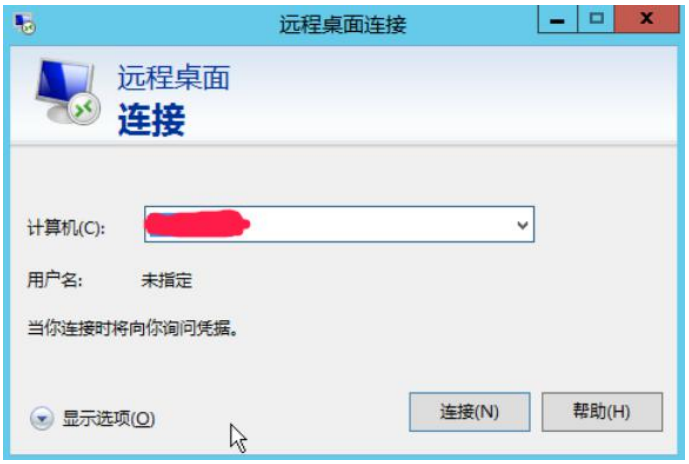


图 4.8 输入 linux 服务器 IP 地址

6 输入统信 UOS 的 IP 地址，其他参数保持默认，单击“**连接**”，即可远程连接到统信 UOS 的桌面环境。连接成功如图 4.9 所示 。

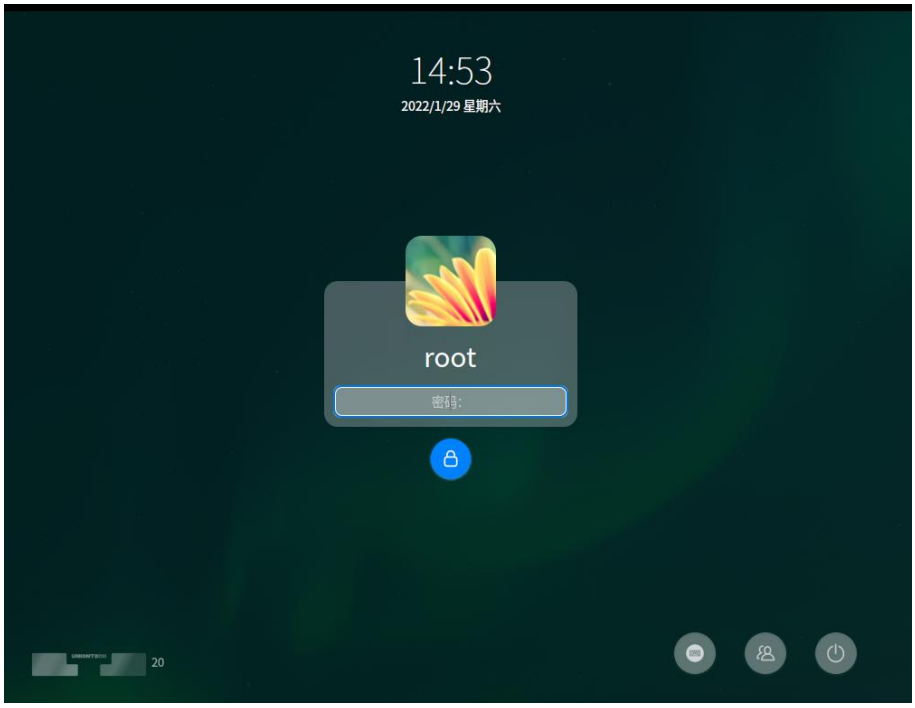


图 4.9 成功远程连接 UOS

4.3 邮件服务器

4.3.1 概述

postfix 是一个 MTA（Mail Transfer Agent，邮件传输代理）服务器软件，主要负责邮件的路由、转发和投递。一个完整的邮件系统应该包含三部分内容：

- 邮件用户代理（MUA，Mail User Agent）：负责查看邮件、编写邮件和向 MTA 发送邮件。
- 邮件传送代理（MTA，Mail Transport Agent）：负责邮件在 Internet 中的传递。

■ 邮件分发代理（MDA，Mail Deliver Agent）：负责从 MTA 中收取邮件，并根据用户名保存到用户邮箱。

4.3.2 安装

如果服务器未安装 postfix 和 mailx，则需要安装。

1 在终端界面，以 root 用户执行 **dnf install** *postfix* 命令，安装 postfix。

```
[root@localhost ~]# dnf install postfix
Last metadata expiration check: 2:05:25 ago on 2020年06月23日 星期二 09时16分02秒.
Dependencies resolved.
================================================================================
 Package             Architecture        Version                  Repository                  Size
================================================================================
Installing:
 postfix             aarch64             2:3.3.1-10.uel20         UOS-Server-Euler            787 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 787 k
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
postfix-3.3.1-10.uel20.aarch64.rpm                               77 kB/s | 787 kB     00:10
--------------------------------------------------------------------------------
Total                                                            77 kB/s | 787 kB     00:10
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                              1/1
  Running scriptlet: postfix-2:3.3.1-10.uel20.aarch64                             1/1
  Installing       : postfix-2:3.3.1-10.uel20.aarch64                             1/1
  Running scriptlet: postfix-2:3.3.1-10.uel20.aarch64                             1/1
  Verifying        : postfix-2:3.3.1-10.uel20.aarch64                             1/1

Installed:
  postfix-2:3.3.1-10.uel20.aarch64

Complete!
[root@localhost ~]#
```

图 4.10 安装 postfix 软件包

2 执行 **dnf install** *mailx* 命令，安装 mailx。

```
[root@localhost ~]# dnf install mailx
Last metadata expiration check: 2:06:11 ago on 2020年06月23日 星期二 09时16分02秒.
Dependencies resolved.
================================================================================
 Package             Architecture        Version                  Repository                  Size
================================================================================
Installing:
 mailx               aarch64             12.5-32.uel20            UOS-Server-Euler            188 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 188 k
Installed size: 486 k
Is this ok [y/N]: y
Downloading Packages:
mailx-12.5-32.uel20.aarch64.rpm                                  36 kB/s | 188 kB     00:05
--------------------------------------------------------------------------------
Total                                                            36 kB/s | 188 kB     00:05
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                              1/1
  Installing       : mailx-12.5-32.uel20.aarch64                                  1/1
  Verifying        : mailx-12.5-32.uel20.aarch64                                  1/1

Installed:
  mailx-12.5-32.uel20.aarch64

Complete!
[root@localhost ~]#
```

图 4.11 安装 mailx 软件包

4.3.3 配置 Postfix 邮件服务器

Postfix 的配置文件位于“**/etc/postfix/main.cf**”中。

1 执行如下命令，打开 vi 工具用于修改配置文件。

vi */etc/postfix/main.cf*

2 修改配置文件，修改后的配置文件示例如下。

```
myhostname = server1.uos.info
mydomain = uos.info
myorigin = $mydomain
inet_interfaces = all
inet_protocols = all
#mydestination = $myhostname, localhost.$mydomain, localhost
mydestination = $myhostname, localhost.$mydomain, localhost,
$mydomain
mynetworks = 192.168.100.0/24, 127.0.0.0/8
home_mailbox = Maildir/
```

📖 *说明：请替换配置文件中的 IP 网段地址为实际使用的网段，此处仅为示例。*

3 执行如下命令，启动邮件服务器。

```
systemctl start postfix
systemctl enable postfix
```

4.3.4 功能测试

1 执行如下命令，创建用户。

```
useradd postfixuser
passwd postfixuser
```

2 执行 **telnet** *localhost 25* 命令连接邮件服务器。

```
[root@localhost ~]# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 server1.uos.info ESMTP Postfix
ehlo localhost
250-server1.uos.info
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
```

```
250 SMTPUTF8
Mail from:<pkumar> 250 2.1.0 Ok
rcpt to:<postfixuser>
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Hello,Welcome to UnionTech OS (postfix)
.
250 2.0.0 Ok: queued as 2B56737D10BB
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

*说明：终端阻塞时，输入“**quit**”可退出上一命令的执行。*

3 在服务器的“**/home/postfixuser/Maildir/new**”目录下查看邮件。

```
[root@localhost postfixuser]# cd /home/postfixuser/Maildir/new
[root@localhost new]# cat *
Return-Path: <root@uos.info>
X-Original-To: postfixuser@uos.info
Delivered-To: postfixuser@uos.info
Received: by server1.uos.info (Postfix, from userid 0)
        id 8D16E217E131; Sat, 29 Oct 2022 20:19:59 +0800 (CST)
Date: Sat, 29 Oct 2022 20:19:59 +0800
To: postfixuser@uos.info
Subject: Test mail
User-Agent: Heirloom mailx 12.5 6/20/10
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Message-Id: <20221029121959.8D16E217E131@server1.uos.info>
From: root <root@uos.info>

hello,this is a postfix test
[root@localhost new]#
```

图 4.12 postfix 接收邮件测试

4.4 域名服务器

4.4.1 概述

BIND 是一种开源的 DNS（Domain Name System）协议的实现，包含对域名的查询和响应所需的所有软件，它也是使用最为广泛的 DNS 服务器软件。

网络通讯大部分是基于 TCP/IP 的，而 TCP/IP 是基于 IP 地址的，所以计算机在网络上进行通讯时只能识别如“**192.168.100.20**”之类的 IP 地址，而不能认识域名。我们很难记住 10 个以上 IP 地址的网站，所以我们访问网站时，更多的是在浏览器地址栏中输入全域名，就能看到所需要的页面，这是因为有一个叫

“DNS 服务器”的计算机自动把我们的全域名“翻译”成了相应的 IP 地址，然后调出 IP 地址所对应的网页。DNS 服务器的类型常见三种类型如下：

- Primary DNS Server(Master)：一个域的主服务器，保存着该域的 ZONE 配置文件，该域所有的配置、更改在该服务器上进行操作。
- Secondary DNS Server(Slave)：一个域的从服务器，一般作为冗余负载使用，从该域的主服务器上抓取 ZONE 配置文件，从服务器不会进行任何信息的更改，必须在主 DNS 服务器上进行 ZONE 配置文件的修改，所有的修改都将会主服务器上进行同步。
- Caching only Server：DNS 缓存服务器，不存在任何的 ZONE 配置文件，仅依靠缓存来为客户端提供服务，它通常用于负载均衡及加速访问操作。

4.4.2 环境准备

1 通过交叉线直接连接测试机 1 （DNS 服务端：192.168.100.20）和测试机 2（客户端：192.168.100.6）。

2 在测试机 1 上，执行如下命令，安装软件包 bind。

```
dnf install bind
```

3 执行如下命令，启动 named 服务，并设置 named 服务开机自启动。

```
systemctl start named.service
systemctl enable named.service
```

4.4.3 配置过程

1 以 root 用户身份登录测试机 1 ，执行 **vi** */etc/named.rfc1912.zones* 命令，设置正向解析和反向解析的配置文件路径，在配置文件中添加内容如下，保存并退出。

```
zone "uos.com" IN {
        type master;
        file "uos.com.zone";
        allow-update { none; };
};
```

2 执行如下命令，生成正向解析配置文件/反向解析配置文件。

```
cp -af /var/named/named.localhost /var/named/uos.com.zone
```

3 执行 **vi** */var/named/uos.com.zone* 命令，在正向解析配置文件中添加如下内容。

```
@       IN SOA dns1.uos.com. zg.xu.uos.com. (
                                        0       ; serial
                                        1D      ; refresh
```

```
                                        1H          ; retry
                                        1W          ; expire
                                        3H )        ; minimum
            NS          dns1
            AAAA        ::1
dns1        A           192.168.100.20
websrv      A           192.168.100.20
ftpsrv      A           192.168.100.20
www         CNAME       websrv
@           A           192.168.100.20
```

4 修改“**/etc/named.conf**”文件，内容如下：

```
options {
listen-on port 53 { 192.168.100.20; };
...
allow-query { any; };
```

5 执行如下命令，修改“**/etc/resolv.conf**”文件。（服务端和客户端）

```
vi /etc/resolv.conf
# Generated by NetworkManager
search uos.com
nameserver 192.168.100.20
```

*说明：配置域名解析服务器前将网络配置为静态 IP，否则会导致重启网络后，“**/etc/resolv.conf**”文件还原，配置无法生效。*

6 执行如下命令，关闭防火墙。

```
systemctl stop firewalld
systemctl disable firewalld
```

4.4.4 功能测试

1 以 root 用户身份登录测试机 2，并执行如下命令。

```
systemctl restart NetworkManager        # 重启网络服务
systemctl restart named                 # 重启 named 服务
```

2 执行 **cat** */var/log/messages* | grep --color *named* 命令，查看系统登录文件里显示的 named 相关信息。

3 执行 **nslookup** *uos.com* 命令，观察域名解析是否成功。（客户端）

4 执行 **dig** *@192.168.100.20 uos.com* 命令，用 dig 工具进行测试。（客户端）

5 执行 **dig** *@192.168.100.20* -x *192.168.100.20* 命令，用 dig 工具进行反向查

询测试。（客户端）

6 执行**dig** *@192.168.100.20 www.uos.com*命令，使用dig工具进行域名测试。（客户端）

7 执行 **dig** *@192.168.100.20 ftp.uos.com* 命令，使用 dig 工具进行域名测试。（客户端）

```
[root@localhost ~]# dig @192.168.100.20 www.uos.com

; <<>> DiG 9.11.4-P2-9.11.4-13.uel20 <<>> @192.168.100.20 www.uos.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38519
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 05723a49ebaf5a692f05f87b5f046e0305d6c5df7b9bbf2f (good)
;; QUESTION SECTION:
;www.uos.com.                   IN      A

;; ANSWER SECTION:
www.uos.com.            10800   IN      CNAME   websrv.uos.com.
websrv.uos.com.         10800   IN      A       192.1.50.4

;; AUTHORITY SECTION:
uos.com.                10800   IN      NS      dns1.uos.com.

;; ADDITIONAL SECTION:
dns1.uos.com.           10800   IN      A       192.168.100.20

;; Query time: 0 msec
;; SERVER: 192.168.100.20#53(192.168.100.20)
;; WHEN: 二 7月 07 20:43:47 CST 2020
;; MSG SIZE  rcvd: 140

[root@localhost ~]#
```

图 4.13 域名测试示例

4.5 DHCP 服务器

4.5.1 概述

DHCP（Dynamic Host Configuration Protocol，动态主机配置协议）是一个局域网的网络协议，使用 UDP 协议工作，主要给内部网络或网络服务供应商自动分配 IP 地址，给用户或内部网络管理员作为对所有计算机进行管理的手段。

- DHCP 客户端：DHCP 客户端是通过 DHCP 来获得网络配置参数的 Internet 主机，通常是普通的用户工作站。
- DHCP 服务器：DHCP 服务器是提供网络参数给 DHCP 客户端的 Internet 主机。
- DHCP 中继代理：在 DHCP 客户端和服务器之间转发 DHCP 消息的主机或者路由器。DHCP 是基于客户端/服务器模型设计的，DHCP 客户端和 DHCP 服务器之间通过收发 DHCP 消息进行通讯。

4.5.2 环境准备

- 测试机 1 （DHCP 服务器：192.168.100.20）和测试机 2 （DHCP 客户端）通过交叉线直连。

■ 测试机 1 具有双网口，默认使用网口 ens33。

■ 测试机 1 上已经安装软件包 dhcp-server。

4.5.3 安装过程

以 root 用户身份登录测试机 1，并执行 **dnf install** *dhcp-server* 命令。

```
[root@localhost named]# yum install dhcp-server
Last metadata expiration check: 3:25:51 ago on 2020年06月23日 星期二 18时21分38秒.
Dependencies resolved.
================================================================================
 Package                Architecture    Version                  Repository              Size
================================================================================
Installing:
 dhcp                   aarch64         12:4.3.6-37.uel20        UOS-Server-Euler        625 k
Installing dependencies:
 bind-export-libs       aarch64         32:9.11.4-13.uel20       UOS-Server-Euler        968 k
 ipcalc                 aarch64         0.2.5-1.uel20            UOS-Server-Euler         29 k

Transaction Summary
================================================================================
Install  3 Packages

Total download size: 1.6 M
Installed size: 5.4 M
Is this ok [y/N]: y
Downloading Packages:
(1/3): ipcalc-0.2.5-1.uel20.aarch64.rpm                  324 kB/s |  29 kB     00:00
(2/3): dhcp-4.3.6-37.uel20.aarch64.rpm                   3.1 MB/s | 625 kB     00:00
(3/3): bind-export-libs-9.11.4-13.uel20.aarch64.rpm      4.2 MB/s | 968 kB     00:00
--------------------------------------------------------------------------------
Total                                                    6.9 MB/s | 1.6 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : ipcalc-0.2.5-1.uel20.aarch64                           1/3
  Installing       : bind-export-libs-32:9.11.4-13.uel20.aarch64            2/3
  Running scriptlet: bind-export-libs-32:9.11.4-13.uel20.aarch64            2/3
  Running scriptlet: dhcp-12:4.3.6-37.uel20.aarch64                         3/3
  Installing       : dhcp-12:4.3.6-37.uel20.aarch64                         3/3
  Running scriptlet: dhcp-12:4.3.6-37.uel20.aarch64                         3/3
  Verifying        : bind-export-libs-32:9.11.4-13.uel20.aarch64            1/3
  Verifying        : dhcp-12:4.3.6-37.uel20.aarch64                         2/3
  Verifying        : ipcalc-0.2.5-1.uel20.aarch64                           3/3

Installed:
  dhcp-12:4.3.6-37.uel20.aarch64            bind-export-libs-32:9.11.4-13.uel20.aarch64
  ipcalc-0.2.5-1.uel20.aarch64

Complete!
[root@localhost named]#
```

图 4.14 安装 dhcp-server 软件包

4.5.4 配置过程

1 执行 **vi** */etc/dhcp/dhcpd.conf* 命令，修改 dhcp 服务的配置文件。

```
#默认租期时间(秒)
default-lease-time 21600;
#最大租约时间
max-lease-time 43200;
ddns-update-style none;
#为所分配的域分配域名
option domain-name "testdhcp.com";
#DNS 服务器地址(多个地址用","隔开)
```

```
option domain-name-servers 192.168.100.254;
#subnet 后跟子网网段，netmask 后跟子网掩码
subnet 192.168.100.0 netmask 255.255.255.0 {
    #地址池
    range 192.168.100.100 192.168.100.200;
    #分发默认网关
    option routers 192.168.100.1;
}
```

2 执行 **dhcpd** -t -cf */etc/dhcp/dhcpd.conf* 命令，查看修改语法是否正确。

4.5.5 功能测试

1 登录测试机 2。

2 执行 **systemctl restart** *NetworkManager* 命令，重启网络服务。

3 执行 **systemctl restart** *dhcpd* 命令，重启 dhcpd 服务。

4 执行 **netstat** -uap 命令，查看服务列表里是否有 dhcpd 服务。

5 执行 **route** -n 命令，查看路由信息。

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         [illegible]     0.0.0.0         UG    100    0        0 enp3s0
[illegible] 0   0.0.0.0         255.255.224.0   U     100    0        0 enp3s0
[illegible] 0   0.0.0.0         255.255.0.0     U     0      0        0 docker0
```

图 4.15 路由信息

4.6 Web 服务器-Apache

4.6.1 概述

Apache HTTP Server（简称 Apache）是 Apache 软件基金会的一个开放源代码的网页服务器软件，可以在大多数电脑操作系统中运行。由于其跨平台和安全性，是最流行的 Web 服务器软件之一。它快速、可靠，并且可通过简单的 API 扩充，将 Perl／Python 等解释器编译到服务器中。

4.6.2 安装过程

1 以 root 用户执行如下命令，安装 Apache。

```
dnf install httpd
```

2 执行如下命令，启动 httpd 服务。

```
systemctl start httpd
```

4.6.3 配置过程

Httpd 配置文件的详情示例如下：

```
# Further relax access to the default document root:
<Directory "/var/www/html">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.4/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
    Require all granted
</Directory>

#
"/etc/httpd/conf/httpd.conf" 356 lines --35%--
```

图 4.16 httpd 配置文件示例

创建“**/var/www/html/index.html**”文件，内容示例如下，其它配置无需修改，采用默认的配置即可。

```
<html>
<body>
    <h1>Hello,how are you!</h1>
    <p>这是一个 apache 测试程序!<br>
        今天是 2020/6/23<br>
    </p>
</body>
</html>
```

4.6.4 功能测试

1 执行如下命令，重启 apache 服务。

```
systemctl restart httpd.service
```

2 在浏览器地址栏输入 http://192.168.100.20/，按下键盘上的“**Enter**”键，显示内容如下：

图 4.17 apache 首页示例

4.7 Web 服务器-Nginx

4.7.1 概述

Nginx 是一个轻量级的网页服务器、反向代理服务器以及电子邮件（IMAP/POP3/SMTP）代理服务器。因稳定性强、具有丰富的功能集和示例配置文件，以及低消耗系统资源而闻名。

4.7.2 安装过程

1 以 root 用户执行如下命令，安装 nginx。

dnf install *nginx*

2 执行如下命令，启动 ngnix 服务。

systemctl start *nginx*

4.7.3 配置过程

执行 **vi** */etc/nginx/nginx.conf* 命令，增加如下所示的配置。

```
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
"/etc/nginx/nginx.conf" 90 lines --43%--
```

图 4.18 nginx.conf 文件内容示例

4.7.4 功能测试

1 在终端界面，切换到“**/usr/share/nginx/html**”目录，并编写 html 测试程序后保存为“**index.html**”文件，内容示例如下：

```
<html>
<body>
        <h1>Hello,how are you!</h1>
        <p>这是一个 nginx 测试程序!<br>
        今天是 2020/6/23<br>
         </p>
</body>
```

```
</html>
```

2 打开浏览器，访问 http://192.168.100.20/，显示如下信息：

图 4.19 nginx 首页示例

4.8 关系型数据库服务器-MySQL

说明：由于 MariaDB 数据库是 MySQL 数据库的另一个分支，因此不能将 MySQL 和 MariaDB 安装在同一操作系统中，如果这样做了，会导致数据库无法正常启动。

4.8.1 概述

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗下公司。在 WEB 应用方面，MySQL 是最好的 RDBMS（Relational Database Management System，关系数据库管理系统）应用软件之一。MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样增加了速度并提高了灵活性。

4.8.2 配置环境

1 执行如下命令，在 root 权限下停止并关闭防火墙。

```
systemctl stop firewalld
systemctl disable firewalld
```

2 执行如下命令，修改 SELINUX 为 disabled。

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/sysconfig/selinux
```

3 执行如下命令，创建组、用户和密码。

```
groupadd mysql
useradd -g mysql mysql
passwd mysql
```

4 创建数据盘。

（1）执行如下命令，创建分区（以“**/dev/vdb**”为例，请根据实际情况创建）。

```
fdisk /dev/vdb
输入 n，按回车确认。
输入 p，按回车确认。
输入 1，按回车确认。
采用默认配置，按回车确认。
```

输入 w，按回车保存。

（2）执行如下命令，创建文件系统（以 xfs 为例，请根据实际情况创建）。

mkfs.xfs */dev/vdb1*

（3）执行如下命令，挂载分区到“**/data**”以供操作系统使用。

mkdir */data*

mount */dev/vdb1 /data*

（4）执行如下命令，在文件中添加如图 4.20 所示的最后一行内容（其中，“**/dev/vdb1**”为示例，具体名称以实际情况为准）。

vi */etc/fstab*

```
[mysql@localhost ~]$ cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Tue Jun 23 08:56:13 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/use-root    /                       ext4    defaults        1 1
UUID=ff5d51ad-8d78-41ad-960a-d6f0c88961c1 /boot                   ext4    defaults        1 2
UUID=329A-28D1          /boot/efi               vfat    umask=0077,shortname=winnt 0 2
/dev/mapper/use-swap    swap                    swap    defaults        0 0
/dev/vdb1              /data xfs defaults 1 2
[mysql@localhost ~]$
```

图 4.20 系统默认挂载配置

5 执行如下命令，创建数据库目录并且授权。

mkdir -p */data/mysql*

cd */data/mysql*

mkdir *data tmp run log*

chown -R *mysql:mysql /data*

4.8.3 安装过程

以 root 用户执行如下命令进行安装。

dnf install *mysql*

4.8.4 功能测试

1 修改“**/etc/my.cnf**”配置文件，修改后的文件内容示例如下：

```
[mysqld_safe]
log-error=/data/mysql/log/mysql.log
pid-file=/data/mysql/run/mysqld.pid
[mysqldump]
quick
[mysql]
no-auto-rehash
[client]
```

```
default-character-set=utf8
[mysqld]
basedir=/usr/local/mysql
socket=/data/mysql/run/mysql.sock
tmpdir=/data/mysql/tmp
datadir=/data/mysql/data
default_authentication_plugin=mysql_native_password
port=3306
user=mysql
```

2 在 root 权限下执行如下命令，修改“**/etc/my.cnf**”文件的组和用户为 mysql:mysql。

```
chown -R mysql:mysql /etc/my.cnf
```

3 执行如下命令，配置环境变量。

```
echo export PATH=$PATH:/usr/local/mysql/bin >> /etc/profile
source /etc/profile
```

4 在 root 权限下执行如下命令，初始化数据库。

```
mysqld --defaults-file=/etc/my.cnf --initialize
2020-03-18T03:27:13.702385Z 0 [System] [MY-013169] [Server] /usr/local/mysql/bin/mysqld (mysqld 8.0.17) initializing of server in progress as process 34014
2020-03-18T03:27:24.112453Z 5 [Note] [MY-010454] [Server] A temporary password is generated for root@localhost: iNat=)#V2tZu
2020-03-18T03:27:28.576003Z 0 [System] [MY-013170] [Server] /usr/local/mysql/bin/mysqld (mysqld 8.0.17) initializing of server has completed
```

说明：本步骤会生成初始密码，请注意保存，登录数据库时需要使用。

5 在 root 权限下执行如下命令，修改文件权限，并先以 root 用户检测 mysql 配置，然后用 mysql 用户启动数据库。

```
chmod 777 /usr/local/mysql/support-files/mysql.server
cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql
chkconfig mysql on
su - mysql
#启动数据库
service mysql start
```

6 执行如下命令，登录数据库。

/usr/local/mysql/bin/mysql -uroot -p -S */data/mysql/run/mysql.sock*

```
[mysql@localhost ~]$ /usr/local/mysql/bin/mysql -uroot -p  -S /data/mysql/run/mysql.sock
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.17

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

图 4.21 登录 mysql 数据库

说明：提示输入密码时，请输入 3 产生的初始密码。

7 登录数据库以后，修改通过 root 用户登录数据库的密码，创建全域 root 用户并进行授权。

```
mysql> alter user 'root'@'localhost' identified by "123456";
mysql> create user 'root'@'%' identified by '123456';
mysql> grant all privileges on *.* to 'root'@'%';
mysql> flush privileges;
```

```
mysql> alter user 'root'@'localhost' identified by "123456";
Query OK, 0 rows affected (0.00 sec)

mysql> create user 'root'@'%' identified by '123456';
Query OK, 0 rows affected (0.01 sec)

mysql> grant all privileges on *.* to 'root'@'%';
Query OK, 0 rows affected (0.05 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

图 4.22 修改数据库 root 用户密码示例

8 执行 **exit** 命令，退出 MySQL。

4.9 关系型数据库服务器-PostgreSQL

4.9.1 概述

PostgreSQL 是自由的对象-关系型数据库服务器（数据库管理系统），PostgreSQL 支持大部分 SQL 标准并且提供了许多其他现代特性：复杂查询、外键、触发器、视图、事务完整性、MVCC。

4.9.2 安装过程

1 以 root 用户执行如下命令，安装 postgresql-server 包。

yum install *postgresql-server*

2 执行如下命令，初始化数据库。

postgresql-setup --initdb

3 执行如下命令，启动 postgresql 服务。

systemctl start *postgresql*

4 执行如下命令，查看 postgresql 服务状态。

systemctl status *postgresql*

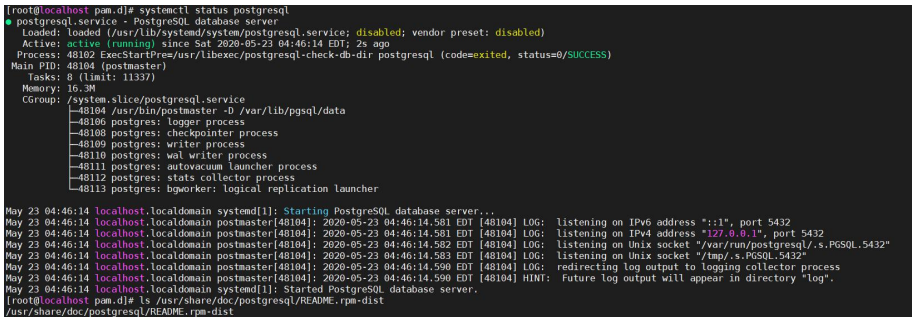


图 4.23 查看 postgresql 数据库服务状态

5 执行如下命令，修改数据库密码。

passwd *postgres*

说明：因为安装完成后，系统会创建一个数据库超级用户 postgres，密码为空。需要执行 ***passwd*** *postgres 命令，修改数据库用户 postgres 的密码。*

4.9.3 功能测试

1 执行 **su** - *postgres* 命令，切换到 postgres 用户。

2 执行 **psql** 命令，连接到 PostgresSQL 数据库。

[postgres@localhost ~]$ **psql**
psql (10.5)
Type "help" for help.
postgres=# help

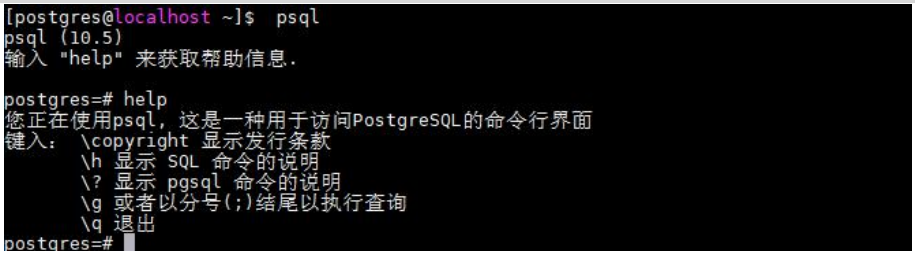


图 4.24 连接 postgresql 数据库

3 执行 **SELECT** version();命令，查看 PostgresSQL 版本信息。

4 执行如下命令，创建测试数据库 pstgdatabase。

postgres=# create database pstgdatabase;

5 执行如下命令，切换到 pstgdatabase 数据库。

postgres=# \c pstgdatabase

说明：选择数据库，可以执行\l 检查可用的数据库列表，利用\c database 选择数据库。

6 执行如下命令，创建表 testtable。

pstgdatabase=# create table testtable(sid integer,sname text,sex char,score integer);

```
CREATE TABLE
pstgdatabase=#
```

7 执行如下命令，向 testtable 表中插入 4 条记录并查询详细信息。

```
pstgdatabase=# insert into testtable(sid,sname,sex,score)
values(001,'zhang','f',75);
INSERT 0 1
pstgdatabase=# insert into testtable(sid,sname,sex,score)
values(002,'li','',75);
INSERT 0 1
pstgdatabase=# select * from testtable;
 sid | sname | sex | score
-----+-------+-----+-------
   1 | zhang | f   |    75
   2 | li    |     |    75
(2 行记录)
```

8 执行如下命令，删除 id 为 001 的记录后再查询其详细信息。

```
pstgdatabase=# delete from testtable where sid='001';
DELETE 1
pstgdatabase=# select * from testtable;
 sid | sname | sex | score
-----+-------+-----+-------
   2 | li    |     |    75
(1 行记录)
```

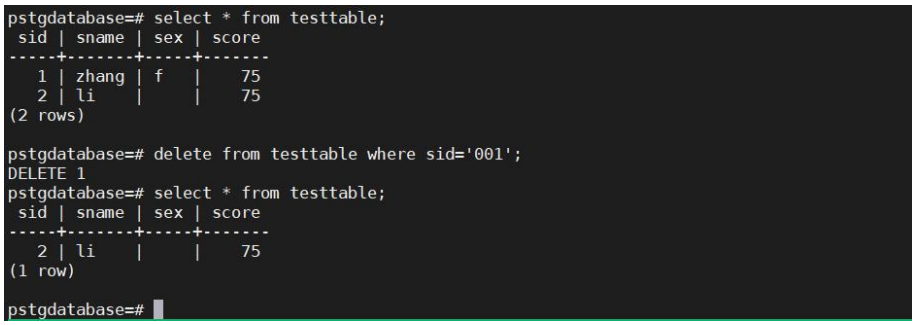


图 4.25 查询数据示例

9 执行**\q** 命令，退出 postgresql。

```
pstgdatabase=#\q
```

4.10 关系型数据库服务器-MariaDB

说明：由于 MariaDB 数据库是 MySQL 数据库的另一个分支，因此不能将 MySQL 和 MariaDB 安装在

同一操作系统中，如果这样做了，会导致数据库无法正常启动。

4.10.1 概述

MariaDB 数据库管理系统是 MySQL 的一个分支，主要由开源社区在维护，采用 GPL 授权许可。在性能、功能、管理、NoSQL 扩展方面，MariaDB 包含了更丰富的特性。比如微秒的支持、线程池、子查询优化、组提交、进度报告等。

4.10.2 配置环境

1 在 root 权限下执行如下命令，停止并关闭防火墙。

systemctl stop *firewalld*
systemctl disable *firewalld*

2 执行如下命令，修改 SELINUX 为 disabled。

sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' */etc/sysconfig/selinux*

3 执行如下命令，创建组、用户和密码。

groupadd *mysql*
useradd -g *mysql mysql*
passwd *mysql*

4 搭建数据盘。

（1）执行如下命令，创建分区（以“**/dev/vdb**”为例，请根据实际情况创建）。

fdisk */dev/vdb*
输入 n，按回车确认；
输入 p，按回车确认；
输入 1，按回车确认；
#采用默认配置，按回车确认；
输入 w，按回车保存。

（2）执行如下命令，创建文件系统（以 xfs 为例，请根据实际情况创建）。

mkfs.xfs */dev/vdb1*

（3）执行如下命令，挂载分区到“**/data**”以供操作系统使用。

mkdir */data*
mount */dev/vdb1 /data*

（4）执行如下命令，编辑“**/etc/fstab**”文件，在文件末尾新增如图 4.26 所示内容，使系统重启后自动挂载数据盘（“**/dev/vdb1**”为示例，具体名称以实际情况为准）。

vi */etc/fstab*

```
[mysql@localhost ~]$ cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Tue Jun 23 08:56:13 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/use-root    /                       ext4    defaults        1 1
UUID=ff5d51ad-8d78-41ad-960a-d6f0c88961c1 /boot                   ext4    defaults        1 2
UUID=329A-28D1          /boot/efi               vfat    umask=0077,shortname=winnt 0 2
/dev/mapper/use-swap    swap                    swap    defaults        0 0
/dev/vdb1              /data xfs defaults 1 2
[mysql@localhost ~]$
```

图 4.26 配置分区自动挂载

5 执行如下命令，创建数据库目录并且授权。

mkdir -p */data/mariadb*

cd */data/mariadb*

mkdir *data tmp run log*

chown -R *mysql:mysql /data*

4.10.3 安装过程

1 以 root 用户执行如下命令，安装 mariadb 包。

dnf install *mariadb-server*

2 执行如下命令，启动 mariadb 服务。

systemctl start *mariadb.service*

3 执行如下命令，查看服务状态。

systemctl status *mariadb.service*

```
[root@localhost pam.d]# systemctl status mariadb.service
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; vendor preset: disabled)
   Active: active (running) since Sat 2020-05-23 04:19:54 EDT; 8min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 46165 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited, status=0/SUCCESS)
  Process: 46030 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mariadb.service (code=exited, status=0/SUCCESS)
  Process: 46006 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited, status=0/SUCCESS)
 Main PID: 46133 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 31 (limit: 11337)
   Memory: 77.1M
   CGroup: /system.slice/mariadb.service
           └─46133 /usr/libexec/mysqld --basedir=/usr

May 23 04:19:54 localhost.localdomain mysql-prepare-db-dir[46030]: Please report any problems at http://mariadb.org/jira
May 23 04:19:54 localhost.localdomain mysql-prepare-db-dir[46030]: The latest information about MariaDB is available at http://mariadb.org/.
May 23 04:19:54 localhost.localdomain mysql-prepare-db-dir[46030]: You can find additional information about the MySQL part at:
May 23 04:19:54 localhost.localdomain mysql-prepare-db-dir[46030]: http://dev.mysql.com
May 23 04:19:54 localhost.localdomain mysql-prepare-db-dir[46030]: Consider joining MariaDB's strong and vibrant community:
May 23 04:19:54 localhost.localdomain mysql-prepare-db-dir[46030]: https://mariadb.org/get-involved/
May 23 04:19:54 localhost.localdomain mysqld[46133]: 2020-05-23  4:19:54 0 [Note] /usr/libexec/mysqld (mysqld 10.3.17-MariaDB) starting as process 46133 ...
May 23 04:19:54 localhost.localdomain mysqld[46133]: 2020-05-23  4:19:54 0 [Warning] Could not increase number of max_open_files to more than 1024 (request: 4183)
May 23 04:19:54 localhost.localdomain mysqld[46133]: 2020-05-23  4:19:54 0 [Warning] Changed limits: max_open_files: 1024  max_connections: 151 (was 151)  table_c
May 23 04:19:54 localhost.localdomain systemd[1]: Started MariaDB 10.3 database server.
lines 1-25/25 (END)
```

图 4.27 查看 mariadb 服务状态示例

4.10.4 功能测试

1 以 root 用户执行如下命令，开启 mariadb 服务。

systemctl start *mariadb*

2 以 root 用户执行如下命令，初始化数据库。

/usr/bin/mysql_secure_installation

*说明：命令执行过程中需要输入数据库的 root 设置的密码，若没有密码则直接按“**Enter**”键，然后根据提示及实际情况进行设置。*

3 执行如下命令，登录数据库，密码为上一步骤中设置的密码。

mysql -u *root* -p

```
[root@localhost log]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.3.9-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

图 4.28 连接 mysql 数据库示例

4 执行如下命令，创建用户。

```
CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';
```

说明：

- *username：用户名。*
- *host：主机名，即用户连接数据库时所在的主机的名字。若是本地用户可用 localhost，若在创建的过程中未指定主机名，则主机名默认为“%”，表示一组主机。*
- *password：用户的登录密码。密码可以为空，如果为空则该用户可以不需要密码登录服务器，但从安全的角度而言，不推荐这种做法。*

```
MariaDB [(none)]> CREATE USER 'userexample1'@'localhost' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]>
```

图 4.29 创建 mariadb 数据库用户示例

5 执行如下命令，查看用户。

```
SELECT USER,HOST,PASSWORD FROM mysql.user;
```

```
MariaDB [(none)]> SELECT USER,HOST,PASSWORD FROM mysql.user;
+--------------+-----------------------+-------------------------------------------+
| USER         | HOST                  | PASSWORD                                  |
+--------------+-----------------------+-------------------------------------------+
| root         | localhost             | *23AE809DDACAF96AF0FD78ED04B6A265E05AA257 |
| root         | localhost.localdomain | *23AE809DDACAF96AF0FD78ED04B6A265E05AA257 |
| root         | 127.0.0.1             | *23AE809DDACAF96AF0FD78ED04B6A265E05AA257 |
| root         | ::1                   | *23AE809DDACAF96AF0FD78ED04B6A265E05AA257 |
| userexample1 | localhost             | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
+--------------+-----------------------+-------------------------------------------+
5 rows in set (0.000 sec)

MariaDB [(none)]>
```

图 4.30 查询 mariadb 数据库中用户信息示例

6 执行如下命令，进行用户授权。

```
GRANT privileges ON databasename.tablename TO
'username'@'hostname';
```

说明：

- *ON 子句：用于指定权限授予的对象和级别。*
- *privileges：用户的操作权限，如 SELECT，INSERT，UPDATE 等，如果要授予所有的权限则使用 ALL。*
- *databasename：数据库名。*
- *tablename：表名。*
- *TO 子句：用来设定用户密码，以及指定被赋予权限的用户。*

- *username：用户名。*
- *hostname：主机名。*
- *如果要授予该用户对所有数据库和表的相应权限，参考如下示例。*

```
MariaDB [(none)]> GRANT SELECT,INSERT ON *.* TO 'userexample1'@'localhost';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]>
```

图 4.31 赋予用户对数据库和表的相应权限示例

7 执行如下命令，创建数据库。

```
CREATE DATABASE databaseexample;
```

```
MariaDB [(none)]> CREATE DATABASE databaseexample;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>
```

图 4.32 创建数据库示例

8 执行如下命令，查看数据库。

```
SHOW DATABASES;
```

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| databaseexample    |
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.000 sec)

MariaDB [(none)]>
```

图 4.33 显示数据库

9 执行 **exit** 命令，退出数据库。

```
MariaDB [(none)]> exit
Bye
[root@localhost log]#
```

图 4.34 退出数据库连接

4.11 非关系型数据库服务器-Redis

统信服务器操作系统中集成的非关系型数据库有 Redis、mongodb。

4.11.1 概述

Redis 是一个高性能 Key-Value 存储系统，使用 ANSI C 编写，它通常被称为数据结构服务器。因为值（value）可以是字符串（String）、哈希（hashes）、列表（list）、集合（sets）和有序集合（sorted sets）等类型，Redis 是最流行的键值对存储数据库。

4.11.2 Redis 安装

1 执行如下命令，安装 redis 包。

sudo yum install *redis* -y

```
[root@localhost ~]# sudo yum install redis.aarch64  -y
上次元数据过期检查：2:10:43 前，执行于 2021年01月22日 星期五 12时52分01秒。
依赖关系解决。
================================================================================
 Package           Architecture        Version                    Repository              Size
================================================================================
安装：
 redis             aarch64             4.0.11-11.uel20            10.7.10.100             775 k

事务概要
================================================================================
安装  1 软件包

总下载：775 k
安装大小：5.4 M
下载软件包：
redis-4.0.11-11.uel20.aarch64.rpm                                  57 MB/s | 775 kB     00:00
--------------------------------------------------------------------------------
总计                                                               53 MB/s | 775 kB     00:00
运行事务检查
事务检查成功。
运行事务测试
```

图 4.35 安装 redis 软件包

2 执行如下命令，查看版本号。

redis-cli -v

```
[root@localhost ~]# redis-cli -v
redis-cli 4.0.11
```

图 4.36 查看 redis-cli 版本

4.11.3 Redis 启动

1 执行如下命令，启动 redis 服务。

systemctl start *redis.service*

2 执行如下命令，查看 redis 服务状态。

systemctl status *redis.service*

```
[root@localhost ~]# systemctl start redis.service
[root@localhost ~]# systemctl status redis.service
● redis.service - Redis persistent key-value database
   Loaded: loaded (/usr/lib/systemd/system/redis.service; disabled; vendor preset: disabled)
   Active: active (running) since Fri 2021-01-22 15:06:13 CST; 4s ago
 Main PID: 5648 (redis-server)
    Tasks: 4
   Memory: 4.1M
   CGroup: /system.slice/redis.service
           └─5648 /usr/bin/redis-server 127.0.0.1:6379

1月 22 15:06:13 localhost.localdomain systemd[1]: Starting Redis persistent key-value database...
1月 22 15:06:13 localhost.localdomain systemd[1]: Started Redis persistent key-value database.
```

图 4.37 查看 redis 服务状态

3 执行如下命令，检查端口监听状态。

lsof -i:*6379*

```
[root@localhost ~]# lsof -i:6379
COMMAND    PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
redis-ser 5648 redis    6u  IPv4 171866      0t0  TCP localhost:redis (LISTEN)
```

图 4.38 检查 redis 端口监听状态

4.11.4 Redis 配置

您可以通过修改“**redis.conf**”文件或使用 CONFIG set 命令来修改配置。

■ 语法：Redis CONFIG 命令格式如下。

```
redis 127.0.0.1:6379> CONFIG GET CONFIG_SETTING_NAME
```

■ 实例

1 执行如下命令。

```
redis 127.0.0.1:6379> CONFIG GET loglevel
1) "loglevel"
2) "notice"
```

```
[root@localhost ~]# redis-cli
127.0.0.1:6379> CONFIG SET loglevel "notice"
OK
127.0.0.1:6379> CONFIG GET loglevel
1) "loglevel"
2) "notice"
127.0.0.1:6379>
```

图 4.39 redis 使用实例

2 使用*号获取所有配置项：

```
127.0.0.1:6379> CONFIG GET *
 1) "dbfilename"
 2) "dump.rdb"
 3) "requirepass"
 4) ""
 5) "masterauth"
 6) ""
 7) "cluster-announce-ip"
 8) ""
 9) "unixsocket"
10) ""
11) "logfile"
12) "/var/log/redis/redis.log"
13) "pidfile"
14) "/var/run/redis_6379.pid"
15) "slave-announce-ip"
16) ""
17) "maxmemory"
```

图 4.40 使用通配符获取所有配置项

3 设置 redis 密码。

```
127.0.0.1:6379> CONFIG SET requirepass admin12#$
OK
```

```
127.0.0.1:6379> AUTH admin12#$
OK
```

图 4.41 设置 redis 临时密码示例

*说明：重启服务后，这样的设置会丢失，永久配置还需要写进“**/etc/redis.conf**”文件中。*

4.11.5 Redis 连接

Redis-cli 连接命令主要是用于连接 redis 服务。

- 语法 1：Redis 客户端的基本语法如下。

```
$ redis-cli
```

- 实例 1：此实例讲解了如何启动 redis 客户端。启动 redis 客户端，并执行 **redis-cli** 命令，连接本地的 redis 服务。

```
$redis-cli
redis 127.0.0.1:6379> PING
PONG
```

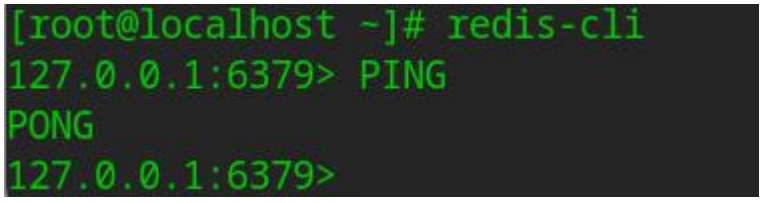


图 4.42 测试 redis 联通性（1）

在以上实例中我们连接到本地的 redis 服务并执行 PING 命令，该命令用于检测 redis 服务是否启动。如果需要在远程 redis 服务上执行命令，我们同样使用的也是 redis-cli 命令。

- 语法 2

```
$ redis-cli -h host -p port -a password
```

- 实例 2：此实例演示了如何连接到主机为 127.0.0.1、端口为 6379、无密码的 redis 服务上。

```
$ redis-cli -h 127.0.0.1 -p 6379 -a ""
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> PING
PONG
```

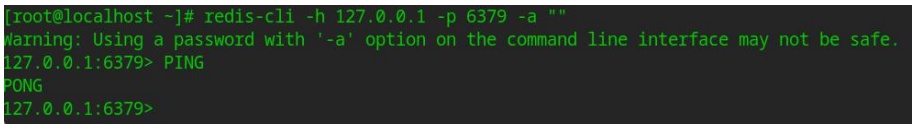


图 4.43 测试 redis 联通性（2）

■ 实例 3：此实例演示了在客户端如何通过验证密码连接到 redis 服务，并检测服务是否在运行。

```
redis-cli -h 127.0.0.1 -p 6379 -a "password"
redis 127.0.0.1:6379> AUTH "password"
OK
redis 127.0.0.1:6379> PING
PONG
```

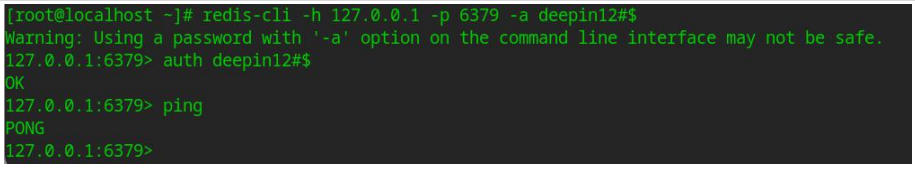


图 4.44 测试 redis 联通性（3）

4.11.6 Redis 连接命令

下表列出了 redis 连接的基本命令。

表 4.1 redis 基本命令

序号	命令及描述
1	AUTH password：验证密码是否正确。
2	ECHO message：打印字符串。
3	PING：查看服务是否运行。
4	QUIT：关闭当前连接。
5	SELECT index：切换到指定的数据库。

4.11.7 Redis 键（key）

Redis 键命令用于管理 redis 的键。

■ 语法：Redis 键命令的基本语法如下。

```
redis 127.0.0.1:6379> COMMAND KEY_NAME
```

■ 实例

```
redis 127.0.0.1:6379> SET w3ckey redis
OK
redis 127.0.0.1:6379> DEL w3ckey
(integer) 1
```

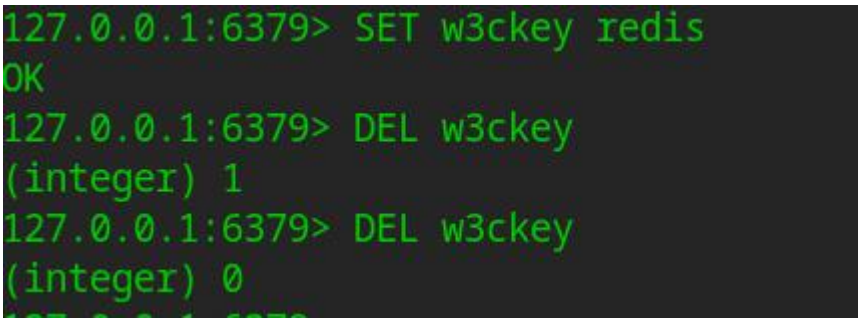


图 4.45 redis 增删键示例

在以上实例中，**DEL** 是一个命令，**w3ckey** 是一个键。如果键被删除成功，执行命令后，会输出**(integer) 1**；否则，将输出**(integer) 0**。

4.11.8 Redis keys 命令

表 4.2 给出了与 Redis 键相关的基本命令。

表 4.2 redis 键操作命令

序号	命令及描述
1	DEL key：该命令用于在 key 存在时删除 key。
2	DUMP key：序列化给定 key，并返回被序列化的值。
3	EXISTS key：检查给定 key 是否存在。
4	EXPIRE key seconds：为给定 key 设置过期时间。
5	EXPIREAT key timestamp EXPIREAT：作用和 EXPIRE 类似，都用于为 key 设置过期时间。不同点在于 EXPIREAT 命令接受的时间参数是 UNIX 时间戳（unix timestamp）。
6	PEXPIRE key milliseconds：设置 key 的过期时间以毫秒计。
7	PEXPIREAT key milliseconds-timestamp：设置 key 过期时间的时间戳（unix timestamp）以毫秒计。
8	KEYS pattern：查找所有符合给定模式（pattern）的 key。
9	MOVE key db :将当前数据库的 key 移动到给定的数据库 db 当中。
10	PERSIST key：移除 key 的过期时间，key 将持久保持。
11	PTTL key：以毫秒为单位返回 key 的剩余的过期时间。
12	TTL key；以秒为单位，返回给定 key 的剩余生存时间（TTL，time to live）。
13	RANDOMKEY：从当前数据库中随机返回一个 key。
14	RENAME key newkey：修改 key 的名称。
15	RENAMENX key newkey：仅当 newkey 不存在时，将 key 改名为 newkey。

序号	命令及描述
16	TYPE key：返回 key 所储存的值的类型。

4.11.9 Redis 字符串（String）

string 是 redis 最基本的类型，您可以理解成和 Memcached 一模一样的类型，一个 key 对应一个 value。

string 类型是二进制安全的，意思是 redis 的 string 可以包含任何数据，比如 jpg 图片或者序列化的对象。

string 类型是 Redis 最基本的数据类型，一个键最大能存储 512MB。

- 语法

```
redis 127.0.0.1:6379> COMMAND KEY_NAME
```

- 实例 1

```
redis 127.0.0.1:6379> SET w3ckey redis
OK
redis 127.0.0.1:6379> GET w3ckey
"redis"
```

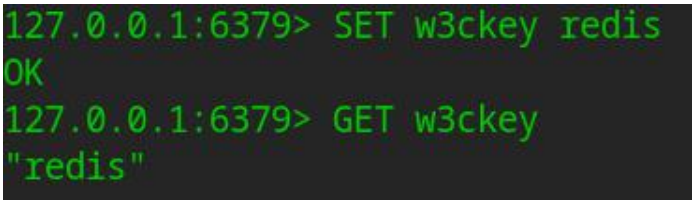


图 4.46 redis 增加、读取示例 1

在以上实例中，我们使用了 **SET** 和 **GET** 命令，键为 w3ckey。

- 实例 2

```
redis 127.0.0.1:6379> SET name "redis.net.cn"
OK
redis 127.0.0.1:6379> GET name
"redis.net.cn"
```

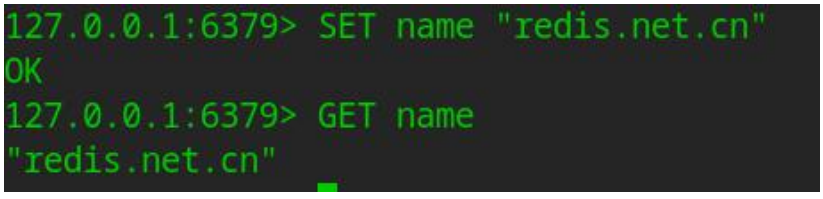


图 4.47 redis 增加、读取示例 2

在以上实例中我们使用了 Redis 的 **SET** 和 **GET** 命令，键为 name，对应的值为 redis.net.cn。

- Redis 字符串命令：下表列出了常用的 redis 字符串命令。

表 4.3 redis 字符串命令

序号	命令和描述
1	SET key value：设置指定 key 的值
2	GET key：获取指定 key 的值。
3	GETRANGE key start end：返回 key 中字符串值的子字符。
4	GETSET key value：将给定 key 的值设为 value，并返回 key 的旧值（old value）。
5	GETBIT key offset：对 key 所储存的字符串值，获取指定偏移量上的位(bit)。
6	MGET key1 [key2..]：获取所有（一个或多个）给定 key 的值。
7	SETBIT key offset value：对 key 所储存的字符串值，设置或清除指定偏移量上的位（bit）。
8	SETEX key seconds value：将值 value 关联到 key，并将 key 的过期时间设为 seconds（以秒为单位）。
9	SETNX key value：只有在 key 不存在时设置 key 的值。
10	SETRANGE key offset value：用 value 参数覆写给定 key 所储存的字符串值，从偏移量 offset 开始。
11	STRLEN key：返回 key 所储存的字符串值的长度。
12	MSET key value [key value ...]：同时设置一个或多个 key-value 对。
13	MSETNX key value [key value ...]：同时设置一个或多个 key-value 对，当且仅当所有给定 key 都不存在时。
14	PSETEX key milliseconds value：这个命令和 SETEX 命令相似，但它以毫秒为单位设置 key 的生存时间，而不是像 SETEX 命令那样以秒为单位。
15	INCR key：将 key 中储存的数字值增一。
16	INCRBY key increment：将 key 所储存的值加上给定的增量值（increment）。
17	INCRBYFLOAT key increment：将 key 所储存的值加上给定的浮点增量值（increment）。
18	DECR key：将 key 中储存的数字值减一。
19	DECRBY key decrement key：所储存的值减去给定的减量值（decrement）。
20	APPEND key value：如果 key 已经存在并且是一个字符串，APPEND 命令将 value 追加到 key 原来的值的末尾。

4.11.10 Redis 哈希（Hash）

Redis hash 是一个键值对集合。Redis hash 是一个 string 类型的 field 和 value 的映射表，hash 特别适合用于存储对象。Redis 中每个 hash 可以存储 2^{32} - 1 个键值对（40 多亿）。

■ 实例

```
redis 127.0.0.1:6379> HMSET user:1 username redis.net.cn password redis.net.cn points 200
OK
redis 127.0.0.1:6379> HGETALL user:1
1) "username"
2) "redis.net.cn"
3) "password"
4) "redis.net.cn"
5) "points"
6) "200"
redis 127.0.0.1:6379>
```

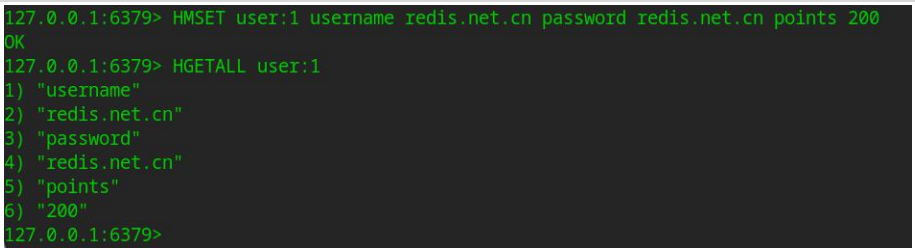


图 4.48 HMSET 和 HGETALL 命令示例

在以上实例中，hash 数据类型存储了包含用户脚本信息的用户对象，使用了 Redis 的 **HMSET** 和 **HGETALL** 命令，**user:1** 为键值。

■ Redis hash 命令：下表列出了 redis hash 基本的相关命令。

表 4.4 redis hash 命令描述

序号	命令及描述
1	HDEL key field2 [field2]：删除一个或多个哈希表字段。
2	HEXISTS key field：查看哈希表 key 中，指定的字段是否存在。
3	HGET key field：获取存储在哈希表中指定字段的值。
4	HGETALL key：获取在哈希表中指定 key 的所有字段和值。
5	HINCRBY key field increment：为哈希表 key 中的指定字段的整数值加上增量 increment。
6	HINCRBYFLOAT key field increment：为哈希表 key 中的指定字段的浮点数值加上增量 increment。

序号	命令及描述
7	HKEYS key：获取所有哈希表中的字段。
8	HLEN key：获取哈希表中字段的数量。
9	HMGET key field1 [field2]：获取所有给定字段的值。
10	HMSET key field1 value1 [field2 value2]：同时将多个 field-value（域-值）对设置到哈希表 key 中。
11	HSET key field value 将哈希表 key 中的字段 field 的值设为 value。
12	HSETNX key field value：只有在字段 field 不存在时，设置哈希表字段的值。
13	HVALS key：获取哈希表中所有值。
14	HSCAN key cursor [MATCH pattern] [COUNT count]：迭代哈希表中的键值对。

4.11.11 Redis 列表（List）

Redis 列表是简单的字符串列表，按照插入顺序排序。您可以添加一个元素导列表的头部（左边）或者尾部（右边）。

一个列表最多可以包含 2^{32}-1 个元素（4294967295，每个列表超过 40 亿个元素）。如下示例：

```
redis 127.0.0.1:6379> lpush redis.net.cn redis
(integer) 1
redis 127.0.0.1:6379> lpush redis.net.cn mongodb
(integer) 2
redis 127.0.0.1:6379> lpush redis.net.cn rabitmq
(integer) 3
redis 127.0.0.1:6379> lrange redis.net.cn 0 10
1) "rabitmq"
2) "mongodb"
) "redis"
redis 127.0.0.1:6379>
```

```
127.0.0.1:6379> lpush redis.net.cn redis
(integer) 1
127.0.0.1:6379> lpush redis.net.cn mongodb
(integer) 2
127.0.0.1:6379> lpush redis.net.cn rabitmq
(integer) 3
127.0.0.1:6379> lrange redis.net.cn 0 10
1) "rabitmq"
2) "mongodb"
3) "redis"
127.0.0.1:6379>
```

图 4.49 redis list 使用示例

下表列出了 Redis 列表相关的基本命令。

表 4.5 redis 列表命令描述

序号	命令及描述
1	BLPOP key1 [key2] timeout：移出并获取列表的第一个元素，如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
2	BRPOP key1 [key2] timeout：移出并获取列表的最后一个元素，如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
3	BRPOPLPUSH source destination timeout：从列表中弹出一个值，将弹出的元素插入到另外一个列表中并返回它，如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
4	LINDEX key index：通过索引获取列表中的元素。
5	LINSERT key BEFORE|AFTER pivot value：在列表的元素前或者后插入元素。
6	LLEN key：获取列表长度。
7	LPOP key：移出并获取列表的第一个元素。
8	LPUSH key value1 [value2]：将一个或多个值插入到列表头部。
9	LPUSHX key value：将一个或多个值插入到已存在的列表头部。
10	LRANGE key start stop：获取列表指定范围内的元素。
11	LREM key count value：移除列表元素。
12	LSET key index value：通过索引设置列表元素的值。
13	LTRIM key start stop：对一个列表进行修剪（trim），就是说，让列表只保留指定区间内的元素，不在指定区间之内的元素都将被删除。
14	RPOP key：移除并获取列表最后一个元素。

序号	命令及描述
15	RPOPLPUSH source destination：移除列表的最后一个元素，并将该元素添加到另一个列表并返回。
16	RPUSH key value1 [value2]：在列表中添加一个或多个值。
17	RPUSHX key value：为已存在的列表添加值。

4.11.12 Redis 集合（Set）

Redis 的 Set 是 string 类型的无序集合。集合成员是唯一的，这就意味着集合中不能出现重复的数据。Redis 中集合是通过哈希表实现的，所以添加、删除、查找的复杂度都是 O(1)。集合中最大的成员数为 2^{32}-1（4294967295，每个集合可存储 40 多亿个成员）。

■ 实例

```
redis 127.0.0.1:6379> SADD w3ckey redis
(integer) 1
redis 127.0.0.1:6379> SADD w3ckey mongodb
(integer) 1
redis 127.0.0.1:6379> SADD w3ckey mysql
(integer) 1
redis 127.0.0.1:6379> SADD w3ckey mysql
(integer) 0
redis 127.0.0.1:6379> SMEMBERS w3ckey
1) "mysql"
2) "mongodb"
3) "redis"
```

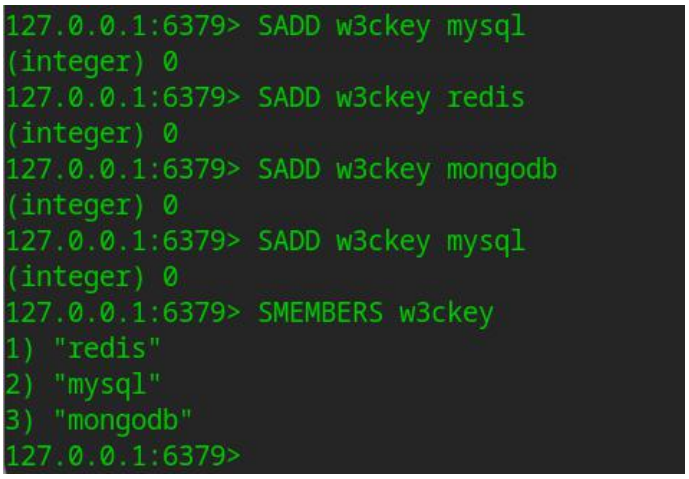


图 4.50 redis 集合使用示例

在以上实例中，我们通过 SADD 命令向名为 w3ckey 的集合插入三个元素。

⚠ *注意：在以上实例中，w3ckey 添加了两次，但根据集合内元素的唯一性，第二次插入的元素将被忽略。*

■ Redis 集合命令：下表列出了 Redis 集合的基本命令。

表 4.6 redis 集合命令描述

序号	命令及描述
1	SADD key member1 [member2]：向集合添加一个或多个成员。
2	SCARD key：获取集合的成员数。
3	SDIFF key1 [key2]：返回给定所有集合的差集。
4	SDIFFSTORE destination key1 [key2] :返回给定所有集合的差集并存储在 destination 中。
5	SINTER key1 [key2]：返回给定所有集合的交集。
6	SINTERSTORE destination key1 [key2]：返回给定所有集合的交集并存储在 destination 中。
7	SISMEMBER key member：判断 member 是否是集合 key 的成员。
8	SMEMBERS key：返回集合中的所有成员。
9	SMOVE source destination member：将 member 元素从 source 集合移动到 destination 集合。
10	SPOP key：移除并返回集合中的一个随机元素。
11	SRANDMEMBER key [count]：返回集合中一个或多个随机数。
12	SREM key member1 [member2]：移除集合中一个或多个成员。
13	SUNION key1 [key2]：返回所有给定集合的并集。
14	SUNIONSTORE destination key1 [key2] :所有给定集合的并集存储在 destination 集合中。
15	SSCAN key cursor [MATCH pattern] [COUNT count]：迭代集合中的元素。

4.11.13 Redis 事务

Redis 事务可以一次执行多个命令，并且带有以下两个重要的保证：

■ 事务是一个单独的隔离操作：事务中的所有命令都会序列化、按顺序地执行。事务在执行的过程中，不会被其他客户端发送来的命令请求所打断。

■ 事务是一个原子操作：事务中的命令要么全部被执行，要么全部都不执行。

一个事务从开始到执行会经历三个阶段，即开始事务、命令入队、执行事务。

以下是一个事务的例子，它先以 **MULTI** 开始一个事务，然后将多个命令入队到事务中，最后由 **EXEC** 命令触发事务，一并执行事务中的所有命令。

```
redis 127.0.0.1:6379> MULTI
```

```
OK
redis 127.0.0.1:6379> SET book-name "Mastering C++ in 21 days"
QUEUED
redis 127.0.0.1:6379> GET book-name
QUEUED
redis 127.0.0.1:6379> SADD tag "C++" "Programming" "Mastering Series"
QUEUED
redis 127.0.0.1:6379> SMEMBERS tag
QUEUED
redis 127.0.0.1:6379> EXEC
1) OK
2) "Mastering C++ in 21 days"
3) (integer) 3
4) 1) "Mastering Series"
   2) "C++"
   3) "Programming"
```

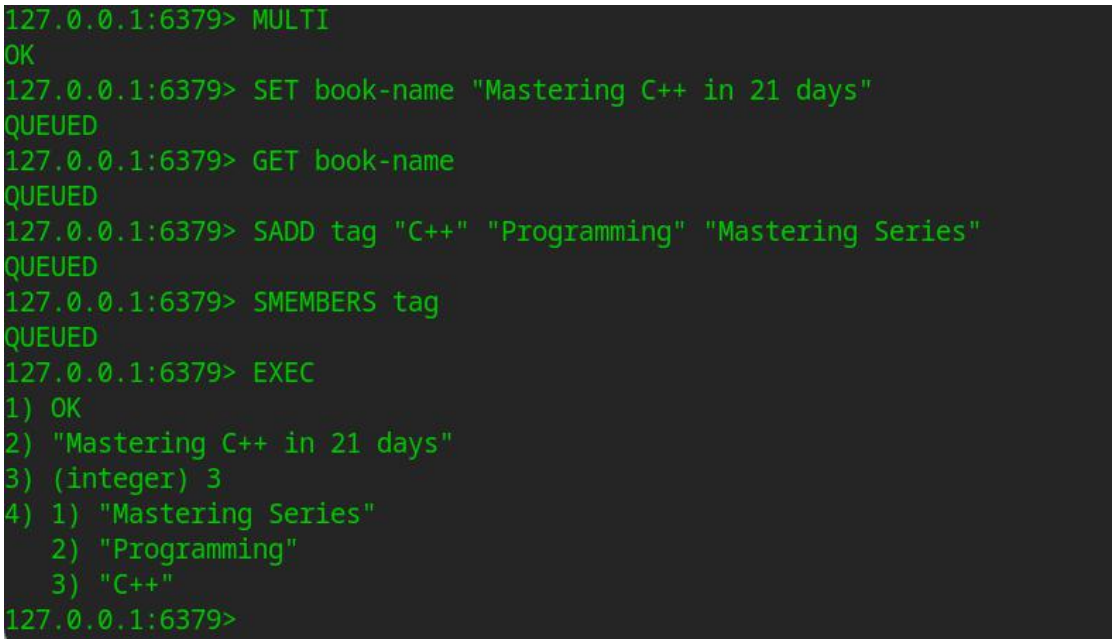


图 4.51 redis 事务示例

表 4.7 列出了 redis 事务的相关命令。

表 4.7 redis 事务命令描述

序号	命令及描述
1	DISCARD：取消事务，放弃执行事务块内的所有命令。
2	EXEC：执行所有事务块内的命令。

序号	命令及描述
3	MULTI：标记一个事务块的开始。
4	UNWATCH：取消 WATCH 命令对所有 key 的监视。
5	WATCH key [key ...]：监视一个或多个 key，如果在事务执行之前这个或这些 key 被其他命令所改动，那么事务将被打断。

4.11.14 Redis 脚本

使用 Lua 解释器来执行 Redis 脚本 ，Reids 2.6 版本通过内嵌支持 Lua 环境。执行脚本的常用命令为 **EVAL**。

- 语法：Eval 命令的基本语法如下。

```
redis 127.0.0.1:6379> EVAL script numkeys key [key ...] arg [arg ...]
```

- 实例：以下实例演示了 redis 脚本的工作过程。

```
redis 127.0.0.1:6379> EVAL "return {KEYS[1],KEYS[2],ARGV[1],ARGV[2]}" 2 key1 key2 first second
1) "key1"
2) "key2"
3) "first"
4) "second"
```

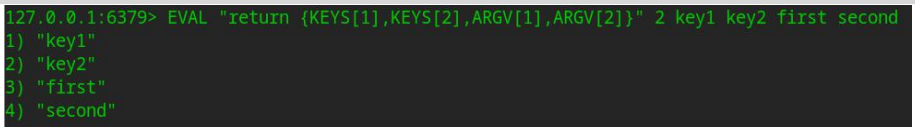


图 4.52 redis 脚本示例

- Redis 脚本命令：下表列出了 redis 脚本常用命令。

表 4.8 redis 脚本命令描述

序号	命令及描述
1	EVAL script numkeys key [key ...] arg [arg ...]：执行 Lua 脚本。
2	EVALSHA sha1 numkeys key [key ...] arg [arg ...]：执行 Lua 脚本。
3	SCRIPT EXISTS script [script ...]：查看指定的脚本是否已经被保存在缓存当中。
4	SCRIPT FLUSH：从脚本缓存中移除所有脚本。
5	SCRIPT KILL：杀死当前正在运行的 Lua 脚本。
6	SCRIPT LOAD script：将脚本 script 添加到脚本缓存中，但并不立即执行这个脚本。

4.11.15 Redis 服务器

Redis 服务器命令主要是用于管理 redis 服务。

■ 实例：以下实例演示了如何获取 redis 服务器的统计信息。

```
redis 127.0.0.1:6379> INFO
# Server
redis_version:2.8.13
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:c2238b38b1edb0e2
redis_mode:standalone
os:Linux 3.5.0-48-generic x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.7.2
process_id:3856
run_id:0e61abd297771de3fe812a3c21027732ac9f41fe
tcp_port:6379
uptime_in_seconds:11554
uptime_in_days:0
hz:10
lru_clock:16651447
*** ***
```

```
127.0.0.1:6379> INFO
# Server
redis_version:4.0.11
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:66bc17d8c6ac4932
redis_mode:standalone
os:Linux 4.19.90-2101.1.0.0055.up1.uel20.aarch64 aarch64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:7.3.0
process_id:6099
run_id:30710bdd1c766cd4441929e843d3935b53dfea11
tcp_port:6379
uptime_in_seconds:318
uptime_in_days:0
```

图 4.53 redis 服务统计信息

■ Redis 服务器命令：下表列出了 redis 服务器的相关命令。

表 4.9 redis 服务器命令描述

序号	命令及描述
1	BGREWRITEAOF：异步执行一个 AOF（AppendOnly File）文件重写操作。
2	BGSAVE：在后台异步保存当前数据库的数据到磁盘。
3	CLIENT KILL [ip:port] [ID client-id]：关闭客户端连接。
4	CLIENT LIST：获取连接到服务器的客户端连接列表。
5	CLIENT GETNAME：获取连接的名称。
6	CLIENT PAUSE timeout：在指定时间内终止运行来自客户端的命令。
7	CLIENT SETNAME connection-name：设置当前连接的名称。
8	CLUSTER SLOTS：获取集群节点的映射数组。
9	COMMAND：获取 Redis 命令详情数组。
10	COMMAND COUNT：获取 Redis 命令总数。
11	COMMAND GETKEYS：获取给定命令的所有键。
12	TIME：返回当前服务器时间。
13	COMMAND INFO command-name [command-name ...]：获取指定 Redis 命令描述的数组。
14	CONFIG GET parameter：获取指定配置参数的值。
15	CONFIG REWRITE：对启动 Redis 服务器时所指定的 redis.conf 配置文件进行改写。
16	CONFIG SET parameter value：修改 redis 配置参数，无需重启。
17	CONFIG RESETSTAT：重置 INFO 命令中的某些统计数据。
18	DBSIZE：返回当前数据库的 key 的数量。
19	DEBUG OBJECT key：获取 key 的调试信息。
20	DEBUG SEGFAULT：让 Redis 服务崩溃。
21	FLUSHALL：删除所有数据库的所有 key。
22	FLUSHDB：删除当前数据库的所有 key。
23	INFO [section]：获取 Redis 服务器的各种信息和统计数值。
24	LASTSAVE：返回最近一次 Redis 成功将数据保存到磁盘上的时间，以 UNIX 时间戳格式表示。
25	MONITOR：实时打印出 Redis 服务器接收到的命令，调试用。
26	ROLE：返回主从实例所属的角色。
27	SAVE：异步保存数据到硬盘。
28	SHUTDOWN [NOSAVE] [SAVE] :异步保存数据到硬盘，并关闭服务器。

序号	命令及描述
29	SLAVEOF host port：将当前服务器转变为指定服务器的从属服务器（slave server）。
30	SLOWLOG subcommand [argument]：管理 redis 的慢日志。
31	SYNC：用于复制功能（replication）的内部命令。

4.11.16 参考

Redis 其他操作请参见 redis 的官网指导：

- https://www.redis.net.cn/tutorial/3501.html
- http://www.redis.cn/documentation.html

4.12 国密算法

4.12.1 概述

国密算法是国家密码局制定标准的一系列算法，包括对称加密算法、椭圆曲线非对称加密算法和杂凑算法，如 SM1、SM2、SM3 等。SM2 为国家密码管理局公布的公钥算法，其加密强度为 256 位，其它几个重要的商用密码算法如下：

- SM1：对称加密算法，加密强度为 128 位，采用硬件实现。
- SM3：密码杂凑算法，杂凑值长度为 32 字节，和 SM2 算法同期公布，详情请参见《国家密码管理局公告（第 22 号）》。
- SMS4：对称加密算法，随 WAPI 标准一起公布，可使用软件实现，加密强度为 128 位。

4.12.2 目的

- openssl 支持国密算法 SM2、SM3、SM4。
- kernel 支持加载 SM4 模块。

4.12.3 术语说明

SM2

SM2 算法就是 ECC 椭圆曲线密码机制，但在签名、密钥交换方面不同于 ECDSA、ECDH 等国际标准，而是采取了更为安全的机制。另外，SM2 推荐了一条 256 位的曲线作为标准曲线。

SM2 算法主要考虑素域 Fp 和 F2m 上的椭圆曲线，分别介绍了这两类域的表示、运算，以及域上的椭圆曲线的点的表示、运算和多倍点计算算法。然后介绍了编程语言中的数据转换，包括整数和字节串，字节串和比特串，域元素和比特串，域元素和整数，点和字节串之间的数据转换规则。详细说明了有限域上椭圆曲线的参数生成以及验证，椭圆曲线的参数包括有限域的选取、椭圆曲线方程

参数、椭圆曲线群基点的选取等，并给出了选取的标准以便于验证。最后给椭圆曲线上密钥对的生成以及公钥的验证，用户的密钥对为（s，sP），其中 s 为用户的私钥，sP 为用户的公钥，由于离散对数问题从 sP 难以得到 s，并针对素域和二元扩域给出了密钥对生成细节和验证方式。

SM3

SM3 密码杂凑（哈希、散列）算法给出了杂凑函数算法的计算方法和计算步骤，并给出了运算示例。此算法适用于商用密码应用中的数字签名和验证，消息认证码的生成与验证以及随机数的生成，可满足多种密码应用的安全需求。此算法对输入长度小于 2 的 64 次方的比特消息，经过填充和迭代压缩，生成长度为 256 比特的杂凑值，其中使用了异或，模，模加，移位，与，或，非运算，由填充，迭代过程，消息扩展和压缩函数所构成。

SM4

此算法是一个分组算法，用于无线局域网产品。该算法的分组长度为 128 比特，密钥长度为 128 比特。加密算法与密钥扩展算法都采用 32 轮非线性迭代结构。解密算法与加密算法的结构相同，只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

此算法采用非线性迭代结构，每次迭代由一个轮函数给出，其中轮函数由一个非线性变换和线性变换复合而成，非线性变换由 S 盒所给出。其中 rki 为轮密钥，合成置换 T 组成轮函数。轮密钥的产生与上图流程类似，由加密密钥作为输入生成，轮函数中的线性变换不同，还有些参数的区别。

参考资料

- GB/T 36441-2018 硬件产品与操作系统兼容性规范。
- GB/T 36465-2018 网络终端操作系统总体技术要求。
- GB/T 25645-2010 信息技术 中文 Linux 服务器操作系统技术要求。
- GB/T 25655-2010 信息技术 中文 Linux 桌面操作系统技术要求。
- GB18030-2005《信息技术中文编码字符集》。
- GB/T 16260-2006 软件工程产品质量。
- GB/T 25645-2010 中文 Linux 服务器操作系统技术要求。
- GB/T 32394-2015 中文 Linux 操作系统运行环境扩充要求。
- GB/T 25000.1-2010《软件工程软件产品质量要求与评价》。

4.12.4 使用方法

kernel

■ 在统信服务器操作系统（版本为 10xxa 时）中

1 安装操作系统的最新版本。

2 查看“**/boot/config**”中如下配置是否为 y。若为 y，则 sm4 已经在内核：CONFIG_CRYPTO_SM4、CONFIG_CRYPTO_SM4_AESNI_AVX_X86_64、CONFIG_CRYPTO_SM4_AESNI_AVX2_X86_64。

```
CONFIG_CRYPTO_SM4=y
CONFIG_CRYPTO_SM4_AESNI_AVX_X86_64=y
CONFIG_CRYPTO_SM4_AESNI_AVX2_X86_64=y
```

图 4.54 sm4 已在内核

■ 在统信服务器操作系统（版本为 10xxe 时）中

1 安装操作系统的最新版本。

2 执行 **modprobe** *sm4_generic* 命令，加载该模块。

3 执行 **lsmod** | grep *sm4* 命令，查看 sm4_generic 模块是否加载成功。

```
[root@localhost ~]# modprobe sm4_generic
[root@localhost ~]# lsmod | grep sm4
sm4_generic            16384  0
[root@localhost ~]#
```

图 4.55 sm4_generic 模块已加载

openssl

■ SM2

1 执行如下命令，检查椭圆曲线是否包含 SM2。

openssl ecparam -list_curves | grep *SM2*
SM2 : SM2 curve over a 256 bit prime field

2 执行如下命令，使用 openssl 生成 SM2 公钥和私钥。

openssl ecparam -genkey -name *SM2* -out *priv.key*
openssl ec -in *priv.key* -pubout -out *pub.key*

```
[root@localhost ~]# lsmod | grep sm4
[root@localhost ~]# modprobe sm4_generic
[root@localhost ~]# openssl ecparam -list_curves | grep SM2
  SM2       : SM2 curve over a 256 bit prime field
[root@localhost ~]# openssl ecparam -genkey -name SM2 -out priv.key
[root@localhost ~]# openssl ec -in priv.key -pubout -out pub.key
read EC key
writing EC key
[root@localhost ~]#
```

图 4.56 使用 openssl 生成 SM2 公私钥

3 查看 SM2 公钥和私钥。

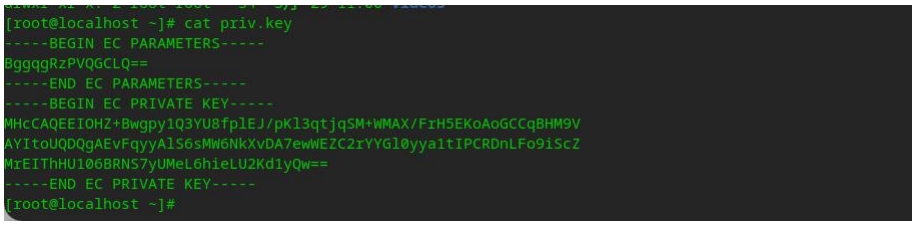


图 4.57 查看 SM2 私钥

图 4.58 查看 SM2 公钥

■ SM3：SM3 是一种利用单向函数构建的密码，只能正向加密，不能逆推解密。

1 执行如下命令，进行 hamc 校验。

echo -n *"abc"* | **openssl dgst** -sm3 -hmac "01234567890123456789012345678901"
(stdin)= ce68f88b05a45a87303a2a3eca942e46d4dce6d06596fa52b3fc0ce43440d5dc

2 执行如下命令，进行非 hamc 校验。

echo -n *"abc"* | **openssl dgst** -SM3
(stdin)= 66c7f0f462eeedd9d1f2d46bdc10e4e24167c4875cf2f7a2297da02b8f4ba8e0

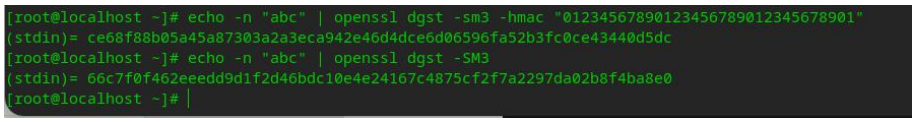


图 4.59 非 hamc 校验

■ SM4

1 执行如下命令，检查 SM4 对称算法。

openssl enc -ciphers | grep *sm4*

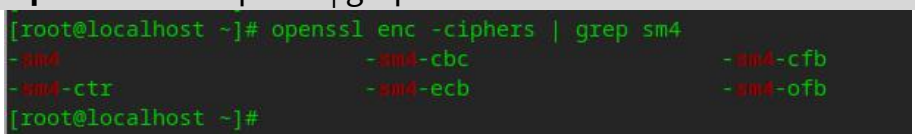


图 4.60 检查 openssl 支持 SM4 算法

2 通过 SM4 国密算法进行数据加密、解密，并验证是否成功。

（1）执行如下命令，创建文件。

echo *UOS> msg.txt*

（2）执行如下命令，进行加密。

openssl enc -e -sm4-cbc -pbkdf2 -k *123456* -in *msg.txt* -out *enc_msg.txt*

（3）执行如下命令，进行解密。

openssl enc -d -sm4-cbc -pbkdf2 -k *123456* -in *enc_msg.txt* -out *dee_msg.txt*

（4）执行如下命令，进行验证。

diff *dee_msg.txt msg.txt*
cat *dee_msg.txt*

4.13 关系型数据库服务器-OpenGauss

4.13.1 简介

openGauss 是关系型数据库，采用客户端/服务器模式和单进程多线程架构；支持单机和一主多备部署方式，备机可读；支持双机高可用和读扩展。

说明：

- *本示例以 openGauss-1.1.0 ARM 版本为例，进行安装介绍。*
- *操作系统升级后，需要将对应版本的 openGauss 数据库也一并进行升级，详情请参见“注意事项”。*
- *物理机或者虚拟机安装 openGauss 数据库时需要物理 CPU 支持 sse4.2 等指令集。*
- *虚拟机安装 openGauss 数据库时，需要在 xml 中配置 host-passthrough 模式，如下所示：*

<cpu mode='host-passthrough' />

```
<os>
  <type arch='x86_64' machine='pc-i440fx-4.0'>hvm</type>
</os>
<features>
  <acpi/>
</features>
<cpu mode='host-passthrough'/>
<clock offset='utc'/>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
```

图 4.61 虚拟机 cpu mode 配置

4.13.2 安装

1 执行如下命令，创建临时目录并修改目录权限。

mkdir -p */opt/software/openGauss*
chmod *755* -R */opt/software*

2 执行如下命令，将上传后的安装包解压。

cd */opt/software/openGauss*

```
tar -xf /opt/software/openGauss/openGauss-1.1.0-UnionTech-64bit-all.tar.gz
tar -xf /opt/software/openGauss/openGauss-1.1.0-UnionTech-64bit-om.tar.gz
```

说明：安装包请联系统信软件售后服务人员获取并上传至对应的目录。

3 执行如下命令，添加用户。

```
useradd omm
groupadd dbgrp
echo "123456 123456" | passwd omm
```

4 执行如下命令，创建安装目录并修改目录权限。

```
mkdir -p /opt/huawei/install/
chmod -R 777 /opt/huawei/
chown omm.dbgrp -R /opt/huawei/
chown omm.dbgrp -R /opt/software/
chmod 755 -R /opt/software/
mkdir -p /opt/huawei/install/
chmod -R 777 /opt/huawei/
userdel -fr omm
cd /opt/software/openGauss/script
sh -x auto_config.sh
```

5 执行如下命令，修改“**/opt/software/openGauss/clusterconfig.xml**”中的 IP 地址为本机 IP 地址。

```
sed -i "s/localhost/xx.xx.xx.xx/g" /opt/software/openGauss/clusterconfig.xml
```

6 执行如下命令，执行预安装。

```
./gs_preinstall -U omm -G dbgrp -X /opt/software/openGauss/clusterconfig.xml
chown omm.dbgrp -R /opt/software/
```

7 执行如下命令，开始安装。

```
su - omm
cd /opt/software/openGauss/script
./gs_install -X /opt/software/openGauss/clusterconfig.xml
```

说明：

- *如果这一步出现*

***"/opt/huawei/install/om/script/local/../gspylib/component/Kernel/Kernel.py", line 87, in start** 等报错，请用 root 权限执行如下命令。*

```
sed -i "s/#enable_numa.*/enable_numa = false/g"
/opt/huawei/install/data/db1/mot.conf
if numactl --hardware |grep "size.*0 MB";then
sed -i "s/#enable_numa.*/enable_numa = false/g"
/opt/huawei/install/data/db1/mot.conf
fi
```

- *如发生不能分配内存片导致的错误，如 **Failed to initialize the memory protect for g_instance.attr.attr_storage.cstore_buffers (1024 Mbytes) or shared memory (1341 Mbytes) is larger.**，说明内存不足，可以执行如下命令。*

```
sync;echo 1 > /proc/sys/vm/drop_caches
```

执行上述命令后，将强制释放 cache。此时，再执行如下命令，减少 shared_buffers，然后去数据目录下找到 postgressql.conf 修改对应参数，如 /opt/huawei/install/data/db1/postgresql.conf。

```
#max_process_memory = 3GB ###之前是 2GB，无法运行
#shared_buffers = 256KB       ###修改后需要重启安装程序
```

8 执行如下命令，启动服务并查看状态。

```
#gs_om -t start
#gs_om -t status --detail
```

4.13.3 基本用法

- 启动 openGauss：以操作系统用户 omm 登录数据库主节点，使用 **gs_om** -t **start** 命令启动 openGauss。

```
[omm@node1 ~]$ gs_om -t start
Starting cluster.
=========================================
=========================================
Successfully started.
```

- 停止 openGauss：以操作系统用户 omm 登录数据库主节点，使用 **gs_om** -t **stop** 命令停止 openGauss。

```
[omm@node1 ~]$ gs_om -t stop
Stopping cluster.
=========================================
Successfully stopped cluster.
```

```
==========================================
End stop cluster.
```

■ 查看 openGauss：以操作系统用户 omm 登录数据库主节点，使用 **gs_om** -t **status** --detail 命令查询 openGauss 各实例情况。

```
[omm@node1 ~]$ gs_om -t status --detail
distName:UnionTech version:20 idNum:
[     Cluster State     ]
cluster_state     : Normal
redistributing    : No
current_az        : AZ_ALL
[    Datanode State     ]
node        node_ip             instance                                state
-----------------------------------------------------------------------
1     node1  192.168.0.128          6001  /opt/huawei/install/data/db1  P
Primary Normal
```

如上部署了数据库主节点实例的服务器 IP 地址为 192.168.0.128，数据库主节点数据路径为“**/opt/huawei/install/data/db1**”。

在查询到的数据库主节点数据路径下的“**postgresql.conf**”文件中查看端口号信息，示例如下：

```
[omm@node1 ~]$ cat /opt/huawei/install/data/db1/postgresql.conf |
grep port
# (change requires restart)
port = 26000
# amount of data between renegotiations, no longer supported
#ssl_renegotiation_limit = 0
# Assigned by installation (change requires restart)
#comm_sctp_port = 1024
# Assigned by installation (change requires restart)
#comm_control_port = 10001
# supported by the operating system:
# The heartbeat thread will not start if not set localheartbeatport and
remoteheartbeatport.
```

说明：26000 为数据库主节点的端口号。

4.13.4 数据库连接

gsql 是 openGauss 提供的在命令行下运行的数据库连接工具。此工具除了具备操作数据库的基本功能，还提供了若干高级特性，便于用户使用。本节只介绍如何使用 gsql 连接数据库，关于 gsql 使用方法的更多信息请参考 gsql。

■ 使用 gsql 本地连接

1 以操作系统用户 omm 登录数据库主节点。

2 使用 gsql 本地连接数据库。

```
gsql -d postgres -p 26000
```

3 连接成功后，系统显示类似如下信息表示数据库连接成功。

```
[omm@localhost ~]$ gsql -d postgres -p 26000
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:06 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
        Type "help" for help.

postgres=#
```

其中，postgres 为 openGauss 安装完成后默认生成的数据库，初始可以连接到此数据库进行新数据库的创建。26000 为数据库主节点的端口号，需根据 openGauss 的实际情况做替换。

4 首次登录需要修改密码，原始密码为安装 openGauss 数据库时手动输入的密码。下方创建了一个用户名为 omm、密码为 uoserpasswd 的用户。

```
ALTER ROLE omm IDENTIFIED BY 'uoserpasswd' REPLACE '123456';
```

■ 使用 gsql 远程连接

◆ 通过 gs_guc 配置白名单

1 以操作系统用户 omm 登录数据库主节点。

2 配置客户端认证方式，允许客户端以“**jack**”用户连接到本机，此处远程连接禁止使用“**omm**”用户（即数据库初始化用户）。例如，下面示例中配置允许 IP 地址为 192.168.0.127 的客户端访问本机，先本地连接数据库，并在数据库中使用如下语句建立“**jack**”用户。

```
postgres=# CREATE USER jack PASSWORD 'Test@123';
```

3 配置白名单。

```
[omm@node1 ~]$ gs_guc set -N all -I all -h "host all jack 192.168.0.127/32 sha256"
```

```
Begin to perform the total nodes: 1.
Popen count is 1, Popen success count is 1, Popen failure count is 0.
Begin to perform gs_guc for datanodes.
Command count is 1, Command success count is 1, Command failure count is 0.

Total instances: 1. Failed instances: 0.
ALL: Success to perform gs_guc!
```

说明：

- *-N：all 表示 openGauss 的所有主机。*
- *-I：all 表示主机的所有实例。*
- *-h：表示指定需要在“**pg_hba.conf**”增加的语句。*
- *all：表示允许客户端连接到任意的数据库。*
- *jack：表示连接数据库的用户。*

4 按照以下格式手动修改本地“**pg_hba.conf**”文件。

```
host DATABASE USER ADDRESS METHOD
```

```
[root@node1 install]# cat /opt/huawei/install/data/db1/pg_hba.conf | grep  "^\s*[^# \t].*$"
local   all             all                                     trust
host    all             all             127.0.0.1/32            trust
host    all             jack            [redacted]/32           sha256
host    all             all             [redacted]/32           trust
host    all             all             ::1/128                 trust
[root@node1 install]#
```

图 4.62 pg_hba.conf 配置示例

192.168.x.x/32 表示只允许 IP 地址为 192.168.x.x 的主机连接。此处的 IP 地址不能为 openGauss 内的 IP 地址，在使用过程中，请根据用户的网络进行配置修改。32 表示子网掩码为 1 的位数，即 255.255.255.255。sha256 表示连接时 jack 用户的密码使用 sha256 算法加密，如果写成 trust，则会报以下错误：

```
gsql: FATAL:  Forbid remote connection with trust method!
FATAL:  Forbid remote connection with trust method!
```

同时，由于 all 代表所有用户，也包括了 jack，所以要将 jack 那行写在 all 上面，否则也会走到 all 的配置里。

说明：修改此配置文件不用重启数据库

◆ 安装 gsql 客户端并连接数据库

1 以 root 登录客户端，上传客户端工具包并配置 gsql 的执行环境变量。

2 执行如下命令，创建“**/tmp/tools**”目录。

```
# mkdir /tmp/tools
```

3 将软件安装包中的“**openGauss-1.1.0-ODBC.tar**”上传到“**/tmp/tools**”路径下。软件包相对位置为安装时所放位置，请根据实际情况填写。不同的操作系统，工具包文件名称会有差异，请根据实际情况选择对应的工具包。

4 执行如下命令，解压文件。

```
# cd /tmp/tools
# tar -zxvf openGauss-1.1.0-ODBC.tar
```

5 执行如下命令，登录数据库主节点所在的服务器，拷贝数据库安装目录下的 bin 目录到客户端“**/tmp/tools**”路径下。随后继续登录客户端执行步骤 6 操作。

```
# scp -r /opt/huawei/install/app/bin
root@192.168.0.127:/tmp/tools
```

其中，“**/opt/huawei/install/app**”为“**clusterconfig.xml**”文件中配置的{gaussdbAppPath}路径，192.168.0.127 为客户端的 IP 地址。

6 执行如下命令，设置环境变量。（在客户端操作）

```
# vi ~/.bashrc
export PATH=/tmp/tools/bin:$PATH
export LD_LIBRARY_PATH=/tmp/tools/lib:$LD_LIBRARY_PATH
```

7 执行如下命令，使环境变量配置生效。（在客户端操作）

```
# source ~/.bashrc
```

8 执行如下命令，连接数据库。（在客户端操作）

```
# gsql -d postgres -h 192.168.0.128 -U jack -p 26000 -W Test@123
```

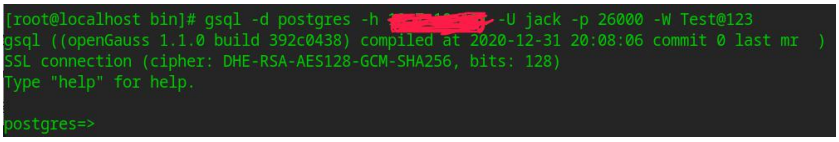


图 4.63 只是指定用户登录 openGauss 数据库

说明：

- *postgres 为需要连接的数据库名称，192.168.0.128 为数据库主节点所在的服务器 IP 地址，jack 为连接数据库的用户，26000 为数据库主节点的端口号，Test@123 为连接数据库的用户 jack 的密码。*
- *连接 openGauss 的机器与 openGauss 不在同一网段时，-h 指定的 IP 地址应为 Manager 界面上所设的 coo.cooListenIp2（应用访问 IP）的取值。*
- *禁止使用 omm 用户进行远程连接数据库。*

■ *数据库安装完成后，默认生成名称为 postgres 的数据库，第一次连接数据库时可以连接到此数据库。*

4.13.5 创建和管理表

表是建立在数据库中的，在不同的数据库中可以存放相同的表，甚至可以通过使用模式在同一个数据库中创建相同名称的表。

■ 创建表：执行如下命令创建表。

```
postgres=# CREATE TABLE customer_t1(
    c_customer_sk             integer,
    c_customer_id             char(5),
    c_first_name              char(6),
    c_last_name               char(8));
```

其中 c_customer_sk、c_customer_id、c_first_name 和 c_last_name 是表的字段名，integer、char(5)、char(6)和 char(8)分别是这些字段名的类型。

■ 向表中插入数据：向表 customer_t1 中插入一行，数据值是按照这些字段在表中出现的顺序列出的，并且用逗号分隔。通常数据值是文本（常量），但也允许使用标量表达式。

```
postgres=# INSERT INTO customer_t1(c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', 'Grace');
```

```
postgres=# INSERT INTO customer_t1(c_customer_sk, c_customer_id, c_first_name) VALUES (3769,
 'hello', 'Grace');
INSERT 0 1
```

图 4.64 openGauss 插入数据到表示例

◆ 如果用户已经知道表中字段的顺序，也可无需列出表中的字段。例如，以下命令与上面的命令效果相同。

```
postgres=# INSERT INTO customer_t1 VALUES (3769, 'hello', 'Grace');
```

◆ 如果用户不知道所有字段的数值，可以忽略其中的一些。没有数值的字段将被填充为字段的缺省值。例如：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_first_name) VALUES (3769, 'Grace');

postgres=# INSERT INTO customer_t1 VALUES (3769, 'hello');
```

◆ 用户也可以对独立的字段或者整个行明确缺省值。例如：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', DEFAULT);
```

```
postgres=# INSERT INTO customer_t1 DEFAULT VALUES;
```

◆ 如果需要在表中插入多行，请使用以下命令：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk,
c_customer_id, c_first_name) VALUES
(6885, 'maps', 'Joes'),
(4321, 'tpcds', 'Lily'),
(9527, 'world', 'James');
```

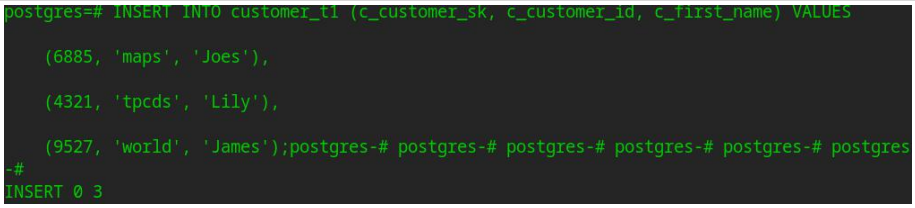


图 4.65 openGauss 插入多条数据示例

如果需要向表中插入多条数据，除此命令外，也可以多次执行插入一行数据命令实现，但是使用此命令可以提升效率。

◆ 如果从指定表插入数据到当前表，例如在数据库中创建了一个表 customer_t1 的备份表 customer_t2，现在需要将表 customer_t1 中的数据插入到表 customer_t2 中，则可以执行如下命令。

```
postgres=# CREATE TABLE customer_t2
(
    c_customer_sk             integer,
    c_customer_id             char(5),
    c_first_name              char(6),
    c_last_name               char(8)
);

postgres=# INSERT INTO customer_t2 SELECT * FROM customer_t1;
```

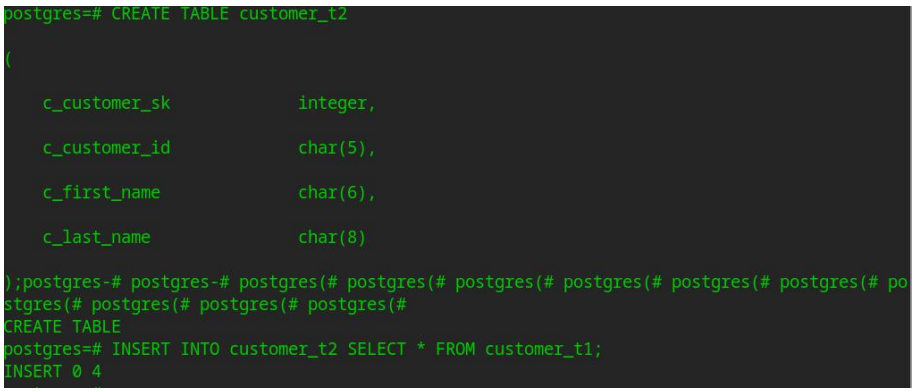


图 4.66 openGauss 创建表示例

- ◆ 从指定表插入数据到当前表时，若指定表与当前表对应的字段数据类型之间不存在隐式转换，则这两种数据类型必须相同。

■ 删除表：执行如下命令删除表。

```
postgres=# DROP TABLE customer_t2 CASCADE;
```

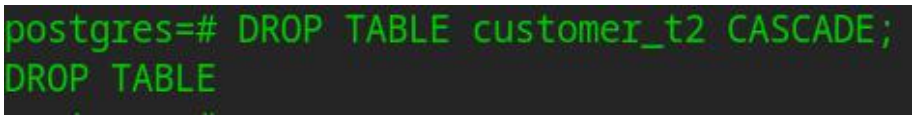


图 4.67 openGauss 删除表示例

在删除表的时候，若当前需删除的表与其他表有依赖关系，需先删除关联的表，然后再删除当前表。

■ 更新表中数据：修改已经存储在数据库中数据的行为叫做更新。用户可以更新单独一行、所有行或者指定的部分行；还可以独立更新每个字段，而其他字段则不受影响。使用 UPDATE 命令更新现有行，需要提供以下三种信息：

- ◆ 表的名称和要更新的字段名。
- ◆ 字段的新值。
- ◆ 要更新哪些行。

SQL 通常不会为数据行提供唯一标识，因此无法直接声明需要更新哪一行。但是可以通过声明一个被更新的行必须满足的条件。只有在表里存在主键的时候，才可以通过主键指定一个独立的行。

建立表和插入数据的步骤请参考创建表和向表中插入数据。

需要将表 customer_t1 中 c_customer_sk 为 9527 的地域重新定义为 9876：

```
postgres=# UPDATE customer_t1 SET c_customer_sk = 9876 WHERE c_customer_sk = 9527;
```

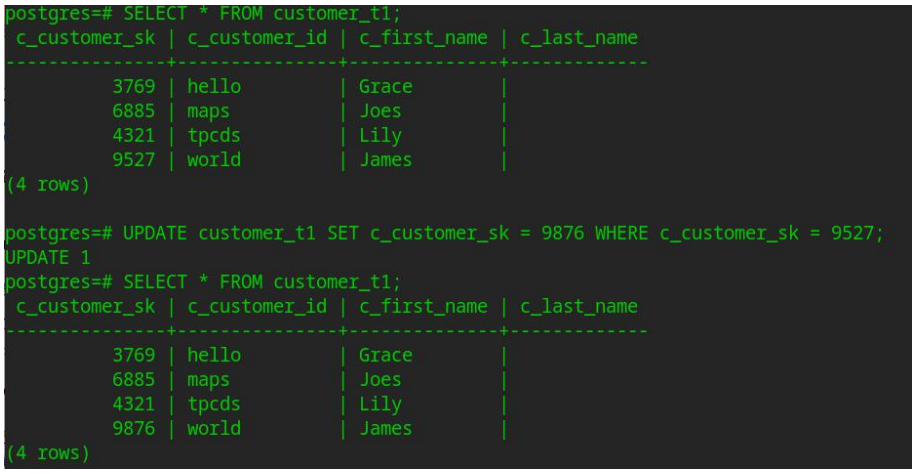


图 4.68 openGauss 更新表数据并查询

这里的表名称也可以使用模式名修饰，否则会从默认的模式路径找到这个表。SET 后面紧跟字段和新的字段值，新的字段值不仅可以是常量，也可以是变

量表达式。例如，把所有 c_customer_sk 的值增加 100：

```
postgres=# UPDATE customer_t1 SET c_customer_sk = c_customer_sk + 100;
```

```
postgres=# UPDATE customer_t1 SET c_customer_sk = c_customer_sk + 100;
UPDATE 4
postgres=# SELECT * FROM customer_t1;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          3869 | hello         | Grace        |
          6985 | maps          | Joes         |
          4421 | tpcds         | Lily         |
          9976 | world         | James        |
(4 rows)
```

图 4.69 openGauss 批量更新记录

在这里省略了 WHERE 子句，表示表中的所有行都要被更新。如果出现了 WHERE 子句，那么只有匹配其条件的行才会被更新。

在 SET 子句中的等号是一个赋值，而在 WHERE 子句中的等号是比较。WHERE 条件不一定是相等测试，许多其他的操作符也可以使用。

用户可以在一个 UPDATE 命令中更新更多的字段，方法是在 SET 子句中列出更多赋值，例如：

```
postgres=# UPDATE customer_t1 SET c_customer_id = 'Admin', c_first_name = 'Local' WHERE c_customer_sk = 4421;
```

```
postgres=# UPDATE customer_t1 SET  c_customer_id = 'Admin', c_first_name = 'Local' WHERE c_customer_sk = 4421;
UPDATE 1
postgres=# SELECT * FROM customer_t1;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          3869 | hello         | Grace        |
          6985 | maps          | Joes         |
          9976 | world         | James        |
          4421 | Admin         | Local        |
(4 rows)
```

图 4.70 openGauss 更新符合条件的记录

批量更新或删除数据后，会在数据文件中产生大量的删除标记，查询过程中标记删除的数据也是需要扫描的。故多次批量更新/删除后，标记删除的数据量过大会严重影响查询的性能。建议在批量更新/删除业务会反复执行的场景下，定期执行 VACUUM FULL 以保持查询性能。

- 查看表中的数据
  - 执行如下命令，使用系统表 pg_tables 查询数据库所有表的信息。

    ```
    postgres=# SELECT * FROM pg_tables;
    ```

  - 执行如下命令，使用 gsql 的\d+命令查询表的属性。

    ```
    postgres=# \d+ customer_t1;
    ```

```
postgres=# \d+ customer_t1;
                        Table "public.customer_t1"
    Column     |     Type     | Modifiers | Storage  | Stats target | Description
---------------+--------------+-----------+----------+--------------+-------------
 c_customer_sk | integer      |           | plain    |              |
 c_customer_id | character(5) |           | extended |              |
 c_first_name  | character(6) |           | extended |              |
 c_last_name   | character(8) |           | extended |              |
Has OIDs: no
Options: orientation=row, compression=no
```

图 4.71 openGauss 查看表属性

◆ 执行如下命令，查询表 customer_t1 的数据量。

```
postgres=# SELECT count(*) FROM customer_t1;
```

```
postgres=# SELECT count(*) FROM customer_t1;
 count
-------
     4
(1 row)
```

图 4.72 openGasuu 查询表记录

◆ 执行如下命令，查询表 customer_t1 的所有数据。

```
postgres=# SELECT * FROM customer_t1;
```

```
postgres=# SELECT * FROM customer_t1;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          3869 | hello         | Grace        |
          6985 | maps          | Joes         |
          9976 | world         | James        |
          4421 | Admin         | Local        |
(4 rows)
```

图 4.73 openGauss 查询表中所有记录

◆ 执行如下命令，只查询字段 c_customer_sk 的数据。

```
postgres=# SELECT c_customer_sk FROM customer_t1;
```

```
postgres=# SELECT c_customer_sk FROM customer_t1;
 c_customer_sk
---------------
          3869
          6985
          9976
          4421
(4 rows)
```

图 4.74 openGauss 查询表中字段

◆ 执行如下命令，过滤字段 c_customer_sk 的重复数据。

```
postgres=# SELECT DISTINCT( c_customer_sk ) FROM customer_t1;
```

```
postgres=# SELECT DISTINCT( c_customer_sk ) FROM customer_t1;
 c_customer_sk
---------------
          9976
          6985
          3869
          4421
(4 rows)
```

图 4.75 openGauss 查询表中字段并过滤重复记录

◆ 执行如下命令，查询字段 c_customer_sk 为 3869 的所有数据。

```
postgres=# SELECT * FROM customer_t1 WHERE c_customer_sk = 3869;
```

```
postgres=# SELECT * FROM customer_t1 WHERE c_customer_sk = 3869;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          3869 | hello         | Grace        |
(1 row)
```

图 4.76 openGauss 以条件查询表中记录

◆ 执行如下命令，按照字段 c_customer_sk 进行排序。

```
postgres=# SELECT * FROM customer_t1 ORDER BY c_customer_sk;
```

```
postgres=# SELECT * FROM customer_t1 ORDER BY c_customer_sk;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          3869 | hello         | Grace        |
          4421 | Admin         | Local        |
          6985 | maps          | Joes         |
          9976 | world         | James        |
(4 rows)
```

图 4.77 openGauss 查询表中记录并安装指定字段排序

■ 删除表中数据：在使用表的过程中，可能会需要删除已过期的数据，删除数据必须从表中整行的删除。SQL 不能直接访问独立的行，只能通过声明被删除行匹配的条件进行。如果表中有一个主键，用户可以指定准确的行。用户可以删除匹配条件的一组行或者一次删除表中的所有行。

◆ 使用 DELETE 命令删除行，例如删除表 customer_t1 中所有 c_customer_sk 为 3869 的记录：

```
postgres=# DELETE FROM customer_t1 WHERE c_customer_sk = 3869;
```

```
postgres=# DELETE FROM customer_t1 WHERE c_customer_sk = 3869;
DELETE 1
postgres=# SELECT * FROM customer_t1;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          6985 | maps          | Joes         |
          9976 | world         | James        |
          4421 | Admin         | Local        |
(3 rows)
```

图 4.78 openGauss 根据条件从表中删除记录

◆ 执行如下命令之一，会删除表中所有的行。

```
postgres=# DELETE FROM customer_t1;
```

或

```
postgres=# TRUNCATE TABLE customer_t1;
```

```
postgres=# DELETE FROM customer_t1;
DELETE 3
postgres=# SELECT * FROM customer_t1;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
(0 rows)
```

图 4.79 openGauss 删除表中所有记录

◆ 删除全表的场景下，建议使用 truncate 清空表数据，不建议使用 delete。

```
postgres=# TRUNCATE TABLE customer_t1;
```

```
postgres=# TRUNCATE TABLE customer_t1;
TRUNCATE TABLE
postgres=# SELECT * FROM customer_t1;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
(0 rows)

postgres=#
```

图 4.80 openGauss 清空表中记录

◆ 执行如下命令，删除创建的表。

```
postgres=# DROP TABLE customer_t1;
```

```
postgres=# DROP TABLE customer_t1;
DROP TABLE
```

图 4.81 openGauss 删除表

4.13.6 其他操作

■ 查看系统表：除了创建的表以外，数据库还包含很多系统表。这些系统表包含 openGauss 安装信息以及 openGauss 上运行的各种查询和进程的信息。可以通过查询系统表来收集有关数据库的信息。“**查看系统表和系统视图**”

中每个表的说明指出了表是对所有用户可见还是只对初始化用户可见。必须以初始化用户身份登录才能查询只对初始化用户可见的表。openGauss 提供了以下类型的系统表和视图：

- 继承自 PG 的系统表和视图，这类系统表和视图具有 PG 前缀。
- openGaussl 新增的系统表和视图，这类系统表和视图具有 GS 前缀。
- 查看数据库中包含的表。例如，在 PG_TABLES 系统表中查看 public schema 中包含的所有表。

```
SELECT distinct(tablename) FROM pg_tables WHERE SCHEMANAME = 'public';
```

结果类似如下这样：

```
tablename
------------------
err_hr_staffs
test
err_hr_staffs_ft3
web_returns_p1
mig_seq_table
films4
(6 rows)
```

■ 查看数据库用户：查询 PG_USER 可以查看数据库中所有用户的列表，还可以查看用户 ID（USESYSID）和用户权限。

```
SELECT * FROM pg_user;
```

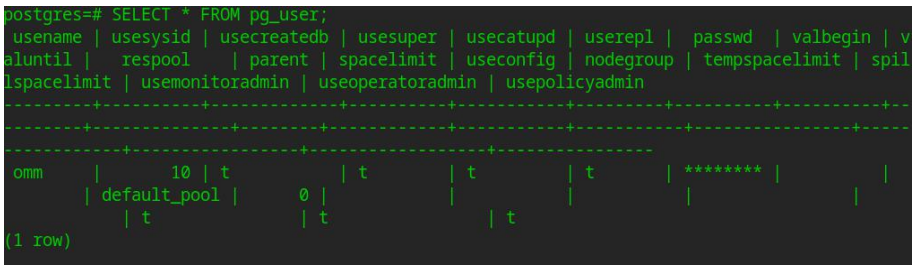


图 4.82 查询所有用户列表

■ 查看和停止正在运行的查询语句：通过视图 PG_STAT_ACTIVITY 可以查看正在运行的查询语句，方法是设置参数 track_activities 为 on。

```
SET track_activities = on;
```

- 当 track_activities 为 on 时，数据库系统才会收集当前活动查询的运行信息，查看正在运行的查询语句。以查看正在运行的查询语句所连接的数据库名、执行查询的用户、查询状态及查询对应的 PID 为例：

```
SELECT datname, usename, state,pid FROM pg_stat_activity;
 datname   | usename | state  |          pid
-----------+---------+--------+-----------------
 postgres | Ruby     | active | 140298793514752
 postgres | Ruby     | active | 140298718004992
 postgres | Ruby     | idle   | 140298650908416
 postgres | Ruby     | idle   | 140298625742592
 postgres | omm | active | 140298575406848
(5 rows)
```

```
postgres=# SELECT datname, usename, state,pid FROM pg_stat_activity;
 datname  | usename | state  |       pid
----------+---------+--------+-----------------
 postgres | omm     | idle   | 281460966187632
 postgres | omm     | idle   | 281460746183280
 postgres | omm     | active | 281460779868784
 postgres | omm     | active | 281460763026032
 postgres | omm     | active | 281460695655024
 postgres | omm     | active | 281461108073072
(6 rows)
```

图 4.83 openGauss 查询数据库运行状态

如果 state 字段显示为 idle ，则表明此连接处于空闲 ，等待用户输入命令。

- 如果仅需要查看非空闲的查询语句，则使用如下命令查看：

```
SELECT datname, usename, state FROM pg_stat_activity WHERE state != 'idle';
```

- 若需要取消运行时间过长的查询 ，通过 PG_TERMINATE_BACKEND 函数，根据线程 ID 结束会话。

```
SELECT PG_TERMINATE_BACKEND(139834759993104);
```

显示类似如下信息，表示结束会话成功。

```
postgres=# SELECT PG_TERMINATE_BACKEND(139834759993104);
WARNING:  PID 139834759993104 is not a gaussdb server thread
CONTEXT:  referenced column: pg_terminate_backend
 pg_terminate_backend
----------------------
 f
(1 row)
```

图 4.84 openGauss 取消回会话

显示类似如下信息，表示用户执行了结束当前会话的操作。

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
```

gsql 客户端使用 PG_TERMINATE_BACKEND 函数结束当前会话后台线

程时，客户端不会退出而是自动重连。即还会返回“**The connection to the server was lost. Attempting reset: Succeeded.**”。

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

- 创建和管理视图：当用户对数据库中的一张或者多张表的某些字段的组合感兴趣，而又不想每次键入这些查询时，用户就可以定义一个视图，以便解决这个问题。视图与基本表不同，不是物理上实际存在的，是一个虚表。数据库中仅存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从视图中查询出的数据也随之改变。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据及变化。视图每次被引用的时候都会运行一次。
  - 创建视图：执行如下命令，创建新视图 MyView。

```
postgres=# CREATE OR REPLACE VIEW MyView AS SELECT * FROM customer_t1 WHERE c_customer_id >1000;
CREATE VIEW
```

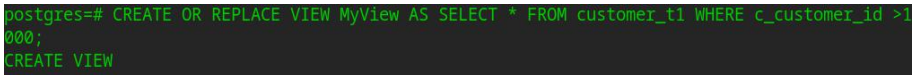


图 4.85 openGauss 创建视图

CREATE VIEW 中的 OR REPLACE 可有可无，当存在 OR REPLACE 时，表示若以前存在该视图就进行替换。

  - 查询视图：执行如下命令，查询 MyView 视图。

```
postgres=# SELECT * FROM MyView;
```

  - 查看某视图的具体信息：执行如下命令，查询 dba_users 的详细信息。

```
postgres=# \d+ dba_users
View "PG_CATALOG.DBA_USERS"
   Column    |                Type                | Modifiers | Storage  | Description
-------------+------------------------------------+-----------+----------+-------------
 USERNAME    | CHARACTER VARYING(64)              |           | extended |
View definition:
 SELECT pg_authid.rolname::character varying(64) AS username
   FROM pg_authid;
```

```
postgres=# SELECT * FROM MyView;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
(0 rows)
```

图 4.86 openGauss 查询视图

◆ 删除视图：执行如下命令，删除 MyView 视图。

```
postgres=# DROP VIEW MyView;
DROP VIEW
```

4.13.7 注意事项

由于版本差异，操作系统升级后，请重新获取对应版本的 openGauss 数据库安装包进行安装。安装前请做好数据备份。

4.13.8 参考

其他操作，如创建 schema、创建分区表、创建索引等，请参考 opengauss 官网的指导：https://opengauss.org/zh/docs/1.1.0/docs/Technicalwhitepaper/Technicalwhitepaper.html

4.14 分布式数据库-MongoDB

4.14.1 简介

MongoDB 是由 C++语言编写的，是一个基于分布式文件存储的开源数据库系统。在高负载的情况下，添加更多的节点，可以保证服务器性能。MongoDB 旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 将数据存储为一个文档，数据结构由键值（key=>value）对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档、数组及文档数组。

4.14.2 安装

```
# sudo yum install mongodb mongodb-server -y
```

```
[root@localhost ~]# sudo yum install mongodb mongodb-server.aarch64
上次元数据过期检查：1:23:54 前，执行于 2021年01月22日 星期五 12时52分01秒。
依赖关系解决。
================================================================================
 Package                 Architecture      Version                  Repository            Size
================================================================================
安装：
 mongodb                 aarch64           4.0.1-6.uel20            10.7.10.100           11 M
 mongodb-server          aarch64           4.0.1-6.uel20            10.7.10.100           21 M
安装依赖关系：
 mongodb-help            noarch            4.0.1-6.uel20            10.7.10.100           32 k

事务概要
================================================================================
安装  3 软件包

总下载：33 M
安装大小：111 M
确定吗？[y/N]： y
下载软件包：
(1/3): mongodb-help-4.0.1-6.uel20.noarch.rpm                 14 MB/s |  32 kB     00:00
(2/3): mongodb-4.0.1-6.uel20.aarch64.rpm                     49 MB/s |  11 MB     00:00
(3/3): mongodb-server-4.0.1-6.uel20.aarch64.rpm              68 MB/s |  21 MB     00:00
--------------------------------------------------------------------------------
总计                                                         102 MB/s |  33 MB     00:00
运行事务检查
事务检查成功。
运行事务测试
事务测试成功。
```

图 4.87 安装 mongodb

4.14.3 启动

systemctl start *mongod.service*

systemctl status *mongod.service*

```
[root@localhost ~]# systemctl start mongod.service
[root@localhost ~]# systemctl status mongod.service
● mongod.service - MongoDB Database Server
   Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2021-01-22 13:12:32 CST; 58min ago
     Docs: https://docs.mongodb.org/manual
  Process: 3944 ExecStartPre=/usr/bin/mkdir -p /var/run/mongodb (code=exited, status=0/SUCCESS)
  Process: 3946 ExecStartPre=/usr/bin/chown mongodb:mongodb /var/run/mongodb (code=exited, status=>
  Process: 3948 ExecStartPre=/usr/bin/chmod 0755 /var/run/mongodb (code=exited, status=0/SUCCESS)
  Process: 3950 ExecStart=/usr/bin/mongod $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 3952 (mongod)
   Memory: 170.6M
   CGroup: /system.slice/mongod.service
           └─3952 /usr/bin/mongod -f /etc/mongod.conf

1月 22 13:12:31 localhost.localdomain systemd[1]: Starting MongoDB Database Server...
1月 22 13:12:31 localhost.localdomain mongod[3950]: about to fork child process, waiting until ser>
1月 22 13:12:31 localhost.localdomain mongod[3950]: forked process: 3952
1月 22 13:12:32 localhost.localdomain mongod[3950]: child process started successfully, parent exi>
1月 22 13:12:32 localhost.localdomain systemd[1]: Started MongoDB Database Server.
lines 1-18/18 (END)
```

图 4.88 启动并查询 mongodb 服务状态

默认情况下，启动 MongoDB 后会初始化以下两个目录：

- 数据存储目录：/var/lib/mongodb。
- 日志文件目录：/var/log/mongodb。

```
[root@localhost ~]# ll /var/lib/mongodb
总用量 64K
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 collection-0-3016583540560261352.wt
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 collection-2-3016583540560261352.wt
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 collection-4-3016583540560261352.wt
drwx------ 2 mongodb mongodb   71  1月  22 14:17 diagnostic.data
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 index-1-3016583540560261352.wt
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 index-3-3016583540560261352.wt
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 index-5-3016583540560261352.wt
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 index-6-3016583540560261352.wt
drwx------ 2 mongodb mongodb  110  1月  22 14:17 journal
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 _mdb_catalog.wt
-rw------- 1 mongodb mongodb    5  1月  22 14:17 mongod.lock
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 sizeStorer.wt
-rw------- 1 mongodb mongodb  114  1月  22 14:17 storage.bson
-rw------- 1 mongodb mongodb   46  1月  22 14:17 WiredTiger
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 WiredTigerLAS.wt
-rw------- 1 mongodb mongodb   21  1月  22 14:17 WiredTiger.lock
-rw------- 1 mongodb mongodb  890  1月  22 14:17 WiredTiger.turtle
-rw------- 1 mongodb mongodb 4.0K  1月  22 14:17 WiredTiger.wt
[root@localhost ~]# ll /var/log/mongodb
总用量 4.0K
-rw-r--r-- 1 mongodb mongodb 3.7K  1月  22 14:17 mongod.log
```

图 4.89 列出 mongodb 数据文件和日志文件

4.14.4 配置实例

配置文件位于“**/etc/mongod.conf**”，使用<option> = <value>的格式，默认配置即可使用大部分功能。

vi */etc/mongod.conf*

```
[root@localhost ~]# grep "^\s*[^# \t].*$" /etc/mongod.conf
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log
storage:
  dbPath: /var/lib/mongodb
  journal:
    enabled: true
processManagement:
  fork: true  # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid  # location of pidfile
net:
  port: 27017
  bindIp: 127.0.0.1  # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses or, alternatively, u
se the net.bindIpAll setting.
```

图 4.90 mongodb 默认配置示例

4.14.5 连接实例

通过 mongo 命令来连接 mongoDB 的命令：

```
mongo [options] [db address] [file names]
```

之前启动实例是在本地 27017 端口，安全模式未被开启，所以不需要输入用户名和密码即可直接连接：

```
mongo 127.0.0.1:27017
```

或者通过**--host** 和**--port** 选项指定主机和端口。一切顺利的话，就进入了

mongoDB shell。shell 会报出一连串权限警告，不过不用担心，这并不会影响之后的操作，在添加授权用户和开启认证后，这些警告会自动消失。

```
[root@localhost ~]# mongo 127.0.0.1:27017
MongoDB shell version v4.0.1
connecting to: mongodb://127.0.0.1:27017/test
MongoDB server version: 4.0.1
Server has startup warnings:
2021-01-22T14:17:39.918+0800 I CONTROL  [initandlisten]
2021-01-22T14:17:39.918+0800 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled f
or the database.
2021-01-22T14:17:39.919+0800 I CONTROL  [initandlisten] **          Read and write access to data a
nd configuration is unrestricted.
2021-01-22T14:17:39.919+0800 I CONTROL  [initandlisten]
2021-01-22T14:17:39.919+0800 I CONTROL  [initandlisten]
2021-01-22T14:17:39.919+0800 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_huge
page/enabled is 'always'.
2021-01-22T14:17:39.919+0800 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2021-01-22T14:17:39.919+0800 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

>
```

图 4.91 连接 mongodb 数据库

4.14.6 数据库基本操作

在进行增查改删操作之前，先了解下常用的 shell 操作：

- db：显示当前所在数据库，默认为 test。
- show dbs：列出可用数据库。
- show tables/show collections：列出数据库中可用集合。
- use <database>：用于切换数据库。

mongoDB 预设有两个数据库，即 admin 和 local。admin 用来存放系统数据，local 用来存放该实例数据。在副本集中，一个实例的 local 数据库对于其它实例是不可见的。

使用 use 命令可切换数据库：

```
> use admin> use local> use newDatabase
```

可以用 use 命令切换至一个不存在的数据库，当您存入新数据时，mongoDB 会创建这个数据库：

```
> use newDatabase
> db.newCollection.insert({x:1})
WriteResult({ "nInserted" : 1 })
```

图 4.92 使用新数据库并插入数据

以上命令向数据库中插入一个文档，返回 1 表示插入成功，mongoDB 自动创建 newCollection 集合和数据库 newDatabase。下面将创建一个 drivers 集合，进行增查改删操作。

■ 创建（Create）：MongoDB 提供 insert 方法创建新文档。

◆ db.collection.inserOne()插入单个文档

WriteResult({ "nInserted" : 1 })

◆ db.collection.inserMany()插入多个文档

◆ db.collection.insert()插入单条或多条文档

这里以 insert 方法为例：

```
> db.drivers.insert({name:"Chen1fa",age:18})
> db.drivers.insert({name:"Xiaose",age:35})
> db.drivers.insert({_id:91,name:"Sun1feng",age:34})
```

要注意，age:18 和 age:"18"是不一样的，前者插入的是数值，后者插入的是字符串。插入新文档如果未指定_id，mongoDB 会自动为插入的文档添加_id 字段。使用 db.drivers.find()命令即可看到刚刚插入的文档：

```
> db.drivers.find()
{ "_id" : ObjectId("598964bd56b8c69ae1e5f36a"), "name" : "Chen1fa", "age" : 18 }
{ "_id" : ObjectId("598964d456b8c69ae1e5f36b"), "name" : "Xiaose", "age" : 35 }
{ "_id" : 91, "name" : "Sun1feng", "age" : 34 }
```

图 4.93 mongodb 插入数据示例

■ 查找（Read）：MongoDB 提供 find 方法查找文档，第一个参数为查询条件。

```
> // 查找所有文档
> db.drivers.find()
{ "_id" : ObjectId("598964bd56b8c69ae1e5f36a"), "name" : "Chen1fa", "age" : 18 }
{ "_id" : ObjectId("598964d456b8c69ae1e5f36b"), "name" : "Xiaose", "age" : 35 }
{ "_id" : 91, "name" : "Sun1feng", "age" : 34 }
> // 查找 name 为 Xiaose 的文档
> db.drivers.find({name: "Xiaose"})
{ "_id" : ObjectId("598964d456b8c69ae1e5f36b"), "name" : "Xiaose", "age" : 35 }
> // 查找 age 大于 20 的文档
> db.drivers.find({age:{$gt:20}})
{ "_id" : ObjectId("598964d456b8c69ae1e5f36b"), "name" : "Xiaose", "age" : 35 }
{ "_id" : 91, "name" : "Sun1feng", "age" : 34 }
```

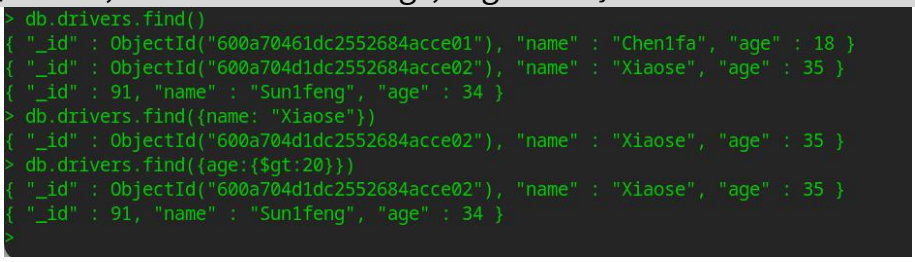


图 4.94 mongodb 查找示例

上述代码中的$gt 对应于大于号>的转义。第二个参数可以传入投影（projection，投影仪中，白色光源通过分光镜、液晶板、投影镜头的光学变换后，投射到反射面上，显示出了彩色图像）文档映射数据：

```
> db.drivers.find({age:{$gt:20}},{name:1})
{ "_id" : ObjectId("598964d456b8c69ae1e5f36b"), "name" : "Xiaose" }
{ "_id" : 91, "name" : "Sun1feng" }
```

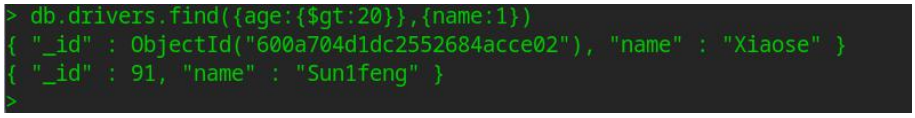


图 4.95 mongodb 查询示例 2

上述命令将查找 age 大于 20 的文档，返回 name 字段，排除其它字段。投影文档中字段为 1 或真值表示包含，0 或假值表示排除，可以设置多个字段

为 1 或 0，但不能混合使用。除此之外，还可以通过 count、skip、limit 等指针（Cursor）方法，改变文档查询的执行方式：

```
> db.drivers.find().count() #统计查询文档数目 3
> db.drivers.find().skip(1).limit(10).sort({age:1})
{ "_id" : 91, "name" : "Sun1feng", "age" : 34 }
{ "_id" : ObjectId("598964d456b8c69ae1e5f36b"), "name" : "Xiaose", "age" : 35 }
```

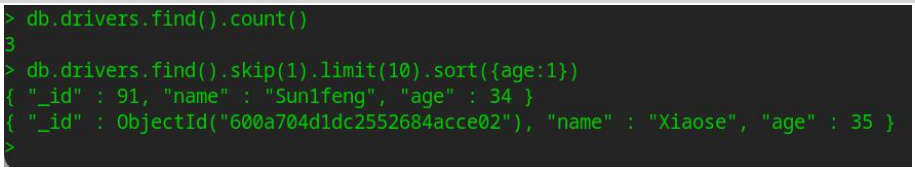


图 4.96 mongodb 查询示例 3

上述查找命令跳过 1 个文档，限制输出 10 个，以 name 子段正序排序（大于 0 为正序，小于 0 位反序）输出结果。可以使用 Cursor 方法中的 pretty 方法，提升查询文档的易读性，特别是在查看嵌套的文档和配置文件的时候：

```
>db.drivers.find().pretty()
```

图 4.97 mongodb 查询示例 4

■ 排序（Sort）：在 MongoDB 中使用 sort()方法对数据进行排序。sort()方法可以通过参数指定排序的字段，并使用 1 和-1 来指定排序的方式，其中 1 为升序排列，而-1 是用于降序排列。

◆ 语法：sort()方法基本语法如下所示。

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

◆ 实例：以下实例演示了 drivers 集合中的数据按字段 age 的降序排列。

```
>db.drivers.find({},{"name":1,"age":1,_id:0}).sort({"age":-1})
{ "name" : "Xiaose", "age" : 35 }
{ "name" : "Sun1feng", "age" : 34 }
```

```
{ "name" : "Chen1fa", "age" : 18 }
```

```
> db.drivers.find({},{"name":1,"age":1,_id:0}).sort({"age":-1})
{ "name" : "Xiaose", "age" : 35 }
{ "name" : "Sun1feng", "age" : 34 }
{ "name" : "Chen1fa", "age" : 18 }
>
```

图 4.98 mongodb 查询示例 5

■ 更新（Update)：MongoDB 提供 updata 方法更新文档。

◆ db.collection.updateOne()：更新最多一个符合条件的文档。

◆ db.collection.updateMany()：更新所有符合条件的文档。

◆ db.collection.replaceOne()：替代最多一个符合条件的文档。

◆ db.collection.update()：默认更新一个文档，可配置 multi 参数后更新多个文档。

以 update()方法为例，其格式如下：

```
> db.collection.update(
<query>,
<update>,
    {
        upsert: <boolean>,
        multi: <boolean>
    }
)
```

各参数意义：

◆ query 为查询条件。

◆ update 为修改的文档。

◆ upsert 为真，查询为空时插入文档。

◆ multi 为真，更新所有符合条件的文档。

下面的命令将 name 字段为 Chen1fa 的文档，更新 age 字段为 30，并添加身高字段：

```
> db.drivers.update({name:"Chen1fa"},{name:"Chen1fa", age:30, height:173})
> db.drivers.update({name:"Sun1feng"},{name:"Sun1feng", age:34, height:171})
> db.drivers.update({name:"Xiaose"},{name:"Xiaose", age:35, height:176})
```

```
> db.drivers.update({name:"Chen1fa"},{name:"Chen1fa", age:30, height:173})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.drivers.update({name:"Sun1feng"},{name:"Sun1feng", age:34, height:171})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.drivers.update({name:"Xiaose"},{name:"Xiaose", age:35, height:176})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

图 4.99 mongodb 更新记录示例 1

要注意的是，如果更新文档只传入 age 字段，那么文档会被更新为{age: 30}，而不是{name:"Chen1fa", age:30}。要避免文档被覆盖，需要用到 set 指令，set 仅替换或添加指定字段：

```
> db.drivers.update({name:"Chen1fa"},{$set:{age:30}})
```

```
> db.drivers.update({name:"Chen1fa"},{$set:{age:30}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
```

图 4.100 mongodb 更新记录示例 2

如果要在查询的文档不存在的时候插入文档，要把 upsert 参数设置真值：

```
> db.drivers.update({name:"Alen"},{age:24},{upsert: true})
```

```
> db.drivers.update({name:"Alen"},{age:24},{upsert: true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("600a720ea0acf8f2cd3755a1")
})
```

图 4.101 mongodb 更新记录示例 3

update 方法默认情况只更新一个文档，如果要更新符合条件的所有文档，要把 multi 设为真值，并使用$set 指令：

```
> db.drivers.update({age:{$gt:25}},{$set:{license:'A'}},{multi: true})
> db.drivers.update({age:{$lt:25}},{$set:{license:'C'}},{multi: true})
```

```
> db.drivers.update({age:{$gt:25}},{$set:{license:'A'}},{multi: true})
WriteResult({ "nMatched" : 3, "nUpserted" : 0, "nModified" : 3 })
> db.drivers.update({age:{$lt:25}},{$set:{license:'C'}},{multi: true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

图 4.102 mongodb 更新记录示例 4

最终结果：

```
> db.drivers.find()
{ "_id" : ObjectId("598964bd56b8c69ae1e5f36a"), "name" : "Chen1fa",
"age" : 30, "height" : 173, "license" : "A" }
{ "_id" : ObjectId("598964d456b8c69ae1e5f36b"), "name" : "Xiaose",
"age" : 35, "height" : 176, "license" : "A" }
{ "_id" : 91, "name" : "Sun1feng", "age" : 34, "license" : "A" }
{ "_id" : ObjectId("598968b3ed1eccef17e79abe"), "age" : 24, "license" :
```

```
"C" }
>
```

```
> db.drivers.find();
{ "_id" : 91, "name" : "Sun1feng", "age" : 34, "height" : 171, "license" : "A" }
{ "_id" : ObjectId("600e24336be39012403a00ee"), "name" : "Chen1fa", "age" : 30, "height" : 173, "li
cense" : "A" }
{ "_id" : ObjectId("600e244a6be39012403a00ef"), "name" : "Xiaose", "age" : 35, "height" : 176, "lic
ense" : "A" }
{ "_id" : ObjectId("600e288f842a57fc628bca91"), "age" : 24, "license" : "C" }
>
```

图 4.103 mongodb 检查已更新记录

■ 索引（Index）：索引通常能够极大的提高查询的效率，如果没有索引，MongoDB 在读取数据时必须扫描集合中的每个文件并选取那些符合查询条件的记录。这种扫描全集合的查询效率是非常低的，特别在处理大量的数据时，查询可以要花费几十秒甚至几分钟，这对网站的性能是非常致命的。索引是特殊的数据结构，索引存储在一个易于遍历读取的数据集合中，索引是对数据库表中一列或多列的值进行排序的一种结构。

◆ createIndex()方法：MongoDB 使用 createIndex()方法来创建索引。

⚠ *注意：在 3.0.0 版本前创建索引方法为 db.collection.ensureIndex()，之后的版本使用 db.collection.createIndex()方法，ensureIndex()还能用，只是 createIndex()的别名。*

语法：createIndex()方法基本语法格式如下所示。

```
>db.collection.createIndex(keys, options)
```

语法中 Key 值为您要创建的索引字段，1 为指定按升序创建索引，如果您想按降序来创建索引指定为-1 即可。

实例：

```
> db.drivers.createIndex({"age":1})
```

```
> db.drivers.createIndex({"age":1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
>
```

图 4.104 mongodb 创建索引示例 1

createIndex()方法中您也可以设置使用多个字段创建索引（关系型数据库中称作复合索引）。

```
> db.drivers.createIndex({"age":1,"height":-1})
```

```
> db.drivers.createIndex({"age":1,"height":-1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

图 4.105 mongodb 创建索引示例 2

在后台创建索引：

```
db.drivers.createIndex({"age":1,"height":-1}, {background: true})
```

通过在创建索引时加 background:true 的选项，让创建工作在后台执行。

- 其他操作

查看集合索引：

```
db.drivers.getIndexes()
```

```
> db.drivers.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "newDatabase.drivers"
        },
        {
                "v" : 2,
                "key" : {
                        "age" : 1
                },
                "name" : "age_1",
                "ns" : "newDatabase.drivers"
        },
        {
                "v" : 2,
                "key" : {
                        "age" : 1,
                        "height" : -1
                },
                "name" : "age_1_height_-1",
                "ns" : "newDatabase.drivers"
        }
]
```

图 4.106 mongodb 查询索引

查看集合索引大小：

```
db.drivers.totalIndexSize()
```

```
> db.drivers.totalIndexSize()
69632
```

图 4.107 mongodb 查看集合索引大小

删除集合指定索引：

```
db.drivers.dropIndex("索引名称")
```

```
> db.drivers.dropIndex("age_1")
{ "nIndexesWas" : 3, "ok" : 1 }
```

图 4.108 mongodb 删除指定索引

删除集合所有索引：

```
db.drivers.dropIndexes()
```

```
> db.drivers.dropIndexes()
{
        "nIndexesWas" : 2,
        "msg" : "non-_id indexes dropped for collection",
        "ok" : 1
}
```

图 4.109 mongodb 删除所有索引

■ 删除（Delete）：MongoDB 提供了 delete 方法删除文档。

◆ db.collection.deleteOne()：删除最多一个符合条件的文档。

◆ db.collection.deleteMany()：删除所有符合条件的文档。

◆ db.collection.remove() 删除一个或多个文档。

以 remove 方法为例：

```
> db.drivers.remove({name:"Xiaose"}) #删除所有 name 为 Xiaose 的文档
> db.drivers.remove({age:{$gt:30}},{justOne:true}) #删除所有 age 大于 30 的文档
> db.drivers.find()
{ "_id" : ObjectId("598964bd56b8c69ae1e5f36a"), "name" : "Chen1fa", "age" : 30, "license" : "A" }
{ "_id" : ObjectId("598968b3ed1eccef17e79abe"), "age" : 24, "license" : "C" }
```

```
> db.drivers.remove({name:"Xiaose"})
WriteResult({ "nRemoved" : 1 })
> db.drivers.remove({age:{$gt:30}},{justOne:true})
WriteResult({ "nRemoved" : 1 })
> db.drivers.find()
{ "_id" : ObjectId("600a70461dc2552684acce01"), "name" : "Chen1fa", "age" : 30, "license" : "A" }
{ "_id" : ObjectId("600a720ea0acf8f2cd3755a1"), "age" : 24, "license" : "C" }
>
```

图 4.110 mongodb 删除记录

MongoDB 提供了 drop 方法删除集合，返回 true 表面删除集合成功：

```
>db.drivers.drop()
```

图 4.111 mongodb 删除集合

■ 关闭实例：关闭 mongoDB 服务。

```
> use admin
> db.shutdownServer()
```

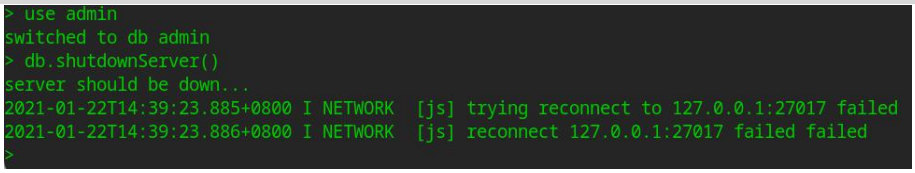


图 4.112 关闭 mongodb 服务

■ 使用 exit 或 Ctrl + C 断开连接。

```
> exit
```

■ 其他操作：其他功能使用请参考 mongodb 官网指导。

- ◆ https://docs.mongoing.com/
- ◆ https://www.mongodb.org.cn/tutorial/

4.15 Haproxy 负载均衡

4.15.1 简介

HAProxy 是由 C 语言编写基于事件驱动模型的一款高效稳定、功能强大的负载均衡软件，其性能可媲美商业负载均衡软件，不过在最新的版本中 HAProxy 已经分为社区版本和企业版，社区版完全免费，企业版有丰富的额外功能。

■ **优点**

- ◆ 支持虚拟主机，通过 frontend 指令来实现。
- ◆ 能弥补 Nginx 的一些缺点比如 Session 的保持、Cookie 的引导等工作。
- ◆ 支持后端服务器健康检查，自动实现故障转移。
- ◆ 它跟 LVS 一样，本身仅仅就只是一款负载均衡软件，单纯从效率上来讲 HAProxy 更会比 Nginx 有更出色的负载均衡速度，在并发处理上也是优于 Nginx。
- ◆ 同时支持四层和七层代理模式。
- ◆ 能对请求的 url 和 header 中的信息做匹配，更细粒度的七层负载均衡。
- ◆ 自带监控管理 web 页面。
- ◆ 能从进出两个方向修改 http 头信息。

- 同样支持多种负载均衡算法。

■ **缺点**

- 重载配置的功能需要重启进程，虽然也是 soft restart，但没有 Nginx 的 reaload 更为平滑和友好。
- 不支持 HTTP cache 功能。

4.15.2 安装

```
# yum install haproxy -y
# systemctl stop firewalld
# systemctl disable firewalld
# setenforce 0
```

4.15.3 配置

配置文件：/etc/haproxy/haproxy.cfg。

说明：haproxy 配置文件可分为全局配置（globalsettings）和代理配置（proxies），而代理配置包含 defaults、frontend、backend、listen。

- *global settings：全局参数配置，用于定义 haproxy 进程管理安全及性能相关参数。*
- *defaults <name>：默认配置参数，下面的段继承该配置，名称是可选的，默认配置参数可由下一个"defaults"重新设定。*
- *frontend <name>：前端配置，定义一系列监听的套接字，这些套接字可接受客户端请求并与之建立连接，可以监听多个端口。*
- *backend <name>：后端配置，定义后台服务器，前端代理服务器将会把客户端的请求调度至这些服务器，类似 nginx 中的 upstream。*
- *listen <name>：定义一组前端和后端的完整代理，可理解为 frontend+backend，通常用于 TCP 流量代理。*
- *name 名称必须由大写和小写字母、数字、'-'、'_'、'.'组成。*

以下是一个配置样例：

```
#---------------------------------------------------------------------
# Example configuration for a possible web application.  See the
# full configuration options online.
#   https://www.haproxy.org/download/1.8/doc/configuration.txt
#---------------------------------------------------------------------
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
pidfile     /var/run/haproxy.pid
```

```
    stats socket     /var/run/haproxy.stat mode 600 level admin
    user            haproxy
    group           haproxy
    daemon
    maxconn         4000

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    retries                 3
    timeout http-request    5s
    timeout queue           1m
    timeout connect         5s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 5s
    timeout check           5s
maxconn                 3000
stats uri               /stats

frontend main
    bind *:80
    default_backend             http_back

backend http_back
    balance       roundrobin
    server    node1 127.0.0.1:5001 check
    server    node2 127.0.0.1:5002 check
    server    node3 127.0.0.1:5003 check
    server    node4 127.0.0.1:5004 check
```

4.15.4 功能使用

http 负载均衡集群

1 环境配置。

表 4.10 负载均衡环境配置

节点类型	IP 地址	安装软件
分发节点	192.168.0.10	haproxy
服务节点	192.168.0.20	httpd
服务节点	192.168.0.30	httpd

2 HAProxy 配置：在“**/etc/haproxy/haproxy.cfg**”中，写入如下内容。

```
global
(略)
defaults
(略)
frontend http
bind 0.0.0.0:80
default_backend http_back
backend http_back
server    s1 192.168.0.20:80    weight 3 check
server    s2 192.168.0.30:80    weight 3 check
```

3 执行如下命令，重启 HAProxy 服务。

```
# systemctl restart haproxy.service
```

4 服务节点配置：分别在两个服务节点执行如下命令，安装 httpd，并启动 httpd。

```
# sudo yum install httpd php -y
# sudo systemctl stop firewalld
# sudo systemctl disable firewalld
# setenforce 0
# sudo echo    “${ip}”    > /var/www/html/index.html
# sudo systemctl start httpd
```

5 访问 HAProxy：访问 http://192.168.0.10 时，HAProxy 会将请求依次分发给 http://192.168.0.20 和 http://192.168.0.30。

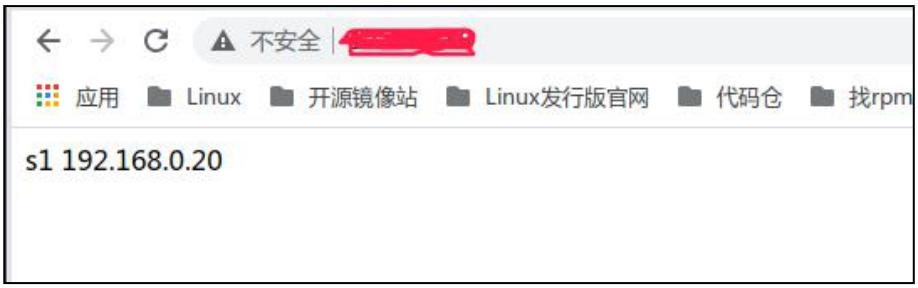


图 4.113 访问 HAProxy 示例 1

图 4.114 访问 HAProxy 示例 2

6 访问 http://192.168.0.10/stats，查看服务器状态。

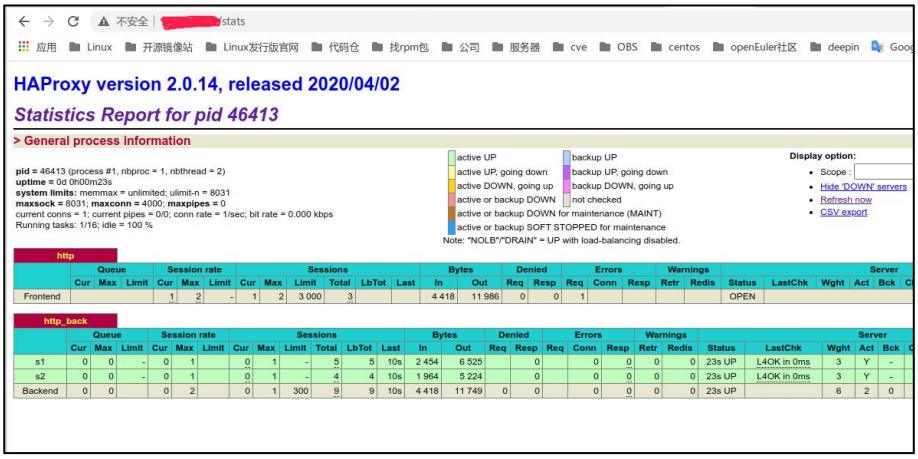


图 4.115 HAProxy 服务示例

mysql 双主高可用

1 环境配置。

表 4.11 mysql 双主高可用环境配置

节点类型	IP 地址	安装软件
分发节点	192.168.0.10	haproxy
服务节点	192.168.0.20	mariadb
服务节点	192.168.0.30	mariadb

2 HAProxy 配置：在“**/etc/haproxy/haproxy.cfg**”中，写入如下内容。

```
global
(略)
```

```
defaults
(略)
listen    mysql
          bind 0.0.0.0:7306
          mode tcp
          balance roundrobin
          server mysql1 192.168.0.20:3306
          server mysql2 192.168.0.30:3306

listen stats
          bind 0.0.0.0:80
```

3 执行如下命令，重启 HAProxy 服务。

```
# systemctl restart haproxy.service
# netstat -plantu | grep 7306
```

4 服务节点配置：分别在两个服务节点执行如下命令。

```
# yum install mariadb mariadb-server -y
# systemctl stop firewalld
# systemctl disable firewalld
# setenforce 0
# systemctl start mariadb
# mysql -uroot -p
MariaDB [mysql]> GRANT ALL ON *.* TO 'haproxy'@'%' IDENTIFIED BY
'123456';
MariaDB [mysql]> FLUSH PRIVILEGES;
```

5 执行如下命令，在 haproxy 远程登录到 mysql 服务器。

```
yum install -y mysql （若没有 mysql 客户端，则需要安装）
mysql -uhaproxy -p123456 -h 192.168.0.20
mysql -uhaproxy -p123456 -h 192.168.0.30
```

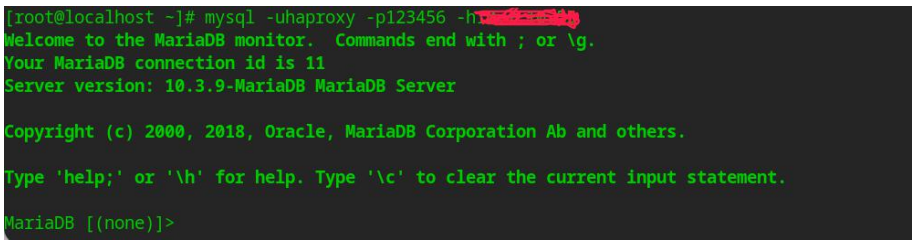


图 4.116 在 HAProxy 服务器远程登录 mysql 服务器

6 在其他服务器通过 haproxy 登录 mysql：在其他的服务器上输入 haproxy 的

服务器地址 192.168.0.10，看是否能连接到数据库。7306 是在配置文件中设置的端口，通过 haproxy 的 7306 端口访问 mysql 的 3306 端口。

```
[root@localhost ~]# mysql -uhaproxy -p123456 -h [redacted] -P 7306
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.3.9-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> quit
Bye
```

图 4.117 通过 HAProxy 访问 mysql 数据库

7 浏览器访问 http://192.168.0.10/stats，出现下图所示信息。

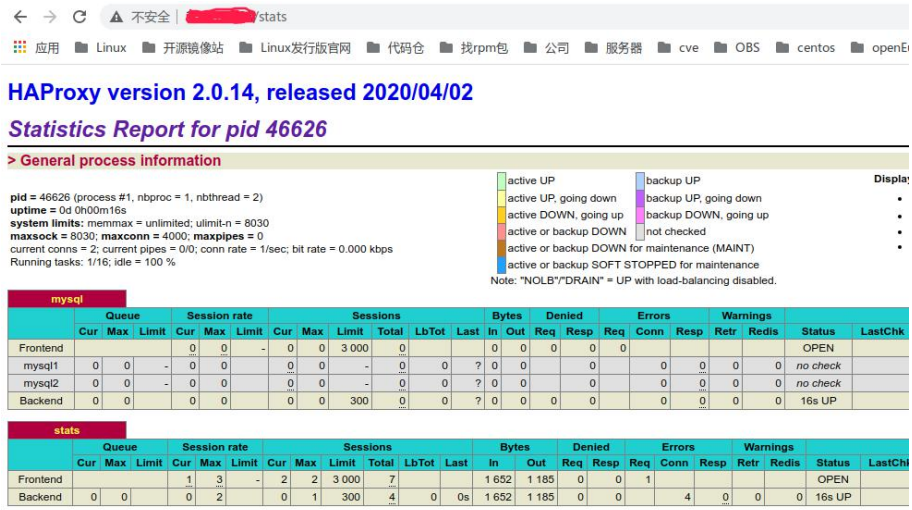


图 4.118 HAProxy 代理 mysql 高可用服务状态示例

HAProxy 动静分离

动静分离原理：将保存静态页面的服务器和动态页面的服务器分成 2 类，HAProxy 根据用户的 URL 当中的后缀来区分是静态页面还是动态页面，HAProxy 工作在 OSI7 层下。

动静分离过程：首先用户会先去请求首页，首页会返回整个页面的框架，用户浏览器会解析到还有很多资源需要再次向 WEB 发起请求，例如<img src=xxx/xxx.jpg>等 html 资源标签。此时，HAProxy 收到请求为 xxx/xxx.jpg 的请求后会判断为.JPG 后缀的资源是静态页面，会向后端的静态页面发起请求，最后交给用户；如果请求的是.php 结尾的则会调度到动态页面的节点中去，继而实现了动静分离效果。

1 环境配置

表 4.12 HAProxy 动静分离环境配置

节点类型	IP 地址	安装软件

节点类型	IP 地址	安装软件
分发节点	192.168.0.10	haproxy
静态节点	192.168.0.20	httpd
动态节点	192.168.0.30	php、httpd

2 haproxy 节点配置：在“**etc/haproxy/haproxy.cfg**”中，写入如下内容。

```
global
(略)
defaults
(略)
frontend public
        bind                192.168.0.10:80 name clear
        use_backend         dynamic if { path_end .php }
        default_backend static

backend static
        balance             roundrobin
        server              web1 192.168.0.20:80 check inter 1000

backend dynamic
        balance             roundrobin
        server              web2 192.168.0.30:80 check inter 1000
```

3 配置服务节点。

■ 静态节点：

```
[root@localhost html]# yum install httpd -y
[root@localhost html]# pwd
/var/www/html
[root@localhost html]# cat index.html
<p align="center">
        <img src="/icons/123.jpg"/>
</p>
[root@localhost html]# cp 123.jpg /usr/share/httpd/icons/
[root@localhost html]# systemctl restart httpd
```

■ 动态节点：

```
[root@localhost html]# yum install php httpd -y
```

```
[root@localhost html]# pwd
/var/www/html
[root@localhost html]# cat index.php
<?php phpinfo();?>
[root@localhost html]# systemctl restart httpd.service
```

4 执行如下命令，重启 HAProxy 服务。

```
[root@localhost html]# systemctl restart haproxy.service
```

5 访问 HAProxy：关闭防火墙，浏览器访问 192.168.0.10，默认打开静态节点的界面。

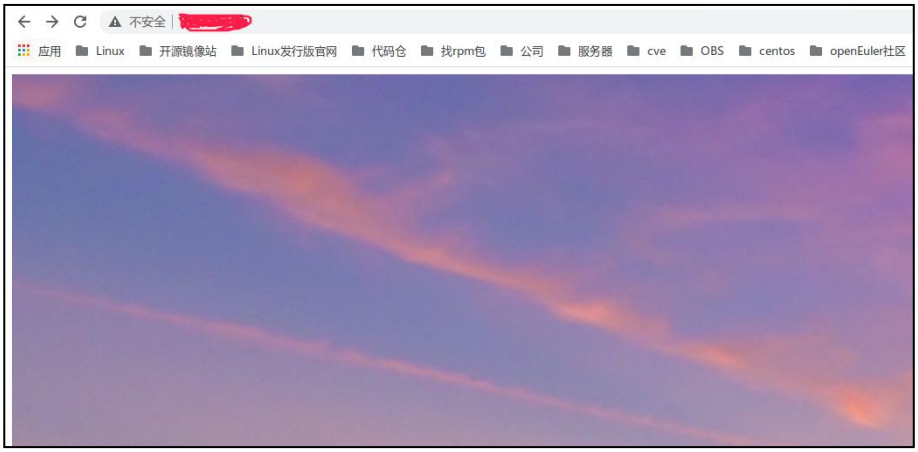
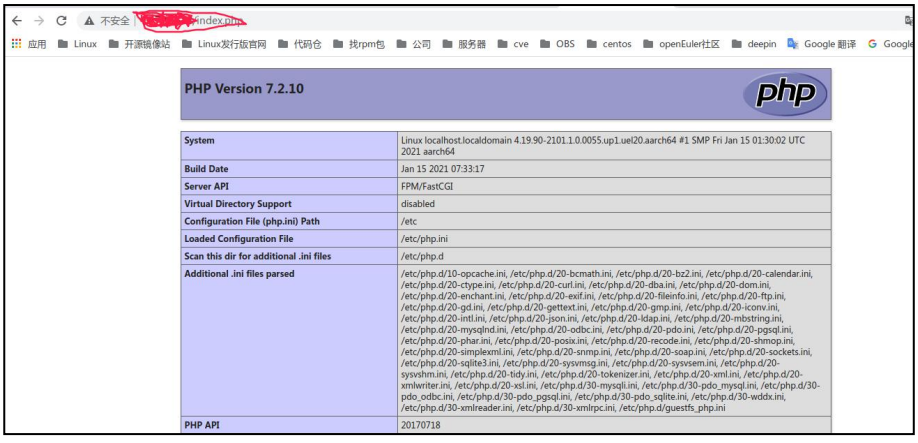


图 4.119 HAProxy 静态节点页面

6 输入 192.168.0.10/index.php，则打开的是动态节点的 php 界面。



PHP Version 7.2.10

System	Linux localhost.localdomain 4.19.90-2101.1.0.0055.up1.uel20.aarch64 #1 SMP Fri Jan 15 01:30:02 UTC 2021 aarch64
Build Date	Jan 15 2021 07:33:17
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/10-opcache.ini, /etc/php.d/20-bcmath.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dba.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-enchant.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-gmp.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-intl.ini, /etc/php.d/20-json.ini, /etc/php.d/20-ldap.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-odbc.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-pgsql.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-posix.ini, /etc/php.d/20-recode.ini, /etc/php.d/20-shmop.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-snmp.ini, /etc/php.d/20-soap.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-sysvmsg.ini, /etc/php.d/20-sysvsem.ini, /etc/php.d/20-sysvshm.ini, /etc/php.d/20-tidy.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_odbc.ini, /etc/php.d/30-pdo_pgsql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-wddx.ini, /etc/php.d/30-xmlreader.ini, /etc/php.d/30-xmlrpc.ini, /etc/php.d/guestfs_php.ini
PHP API	20170718

图 4.120 HAProxy 动态节点页面

7 查看服务节点状态：浏览器访问 http://192.168.0.10/stats，显示如下图所示的信息。

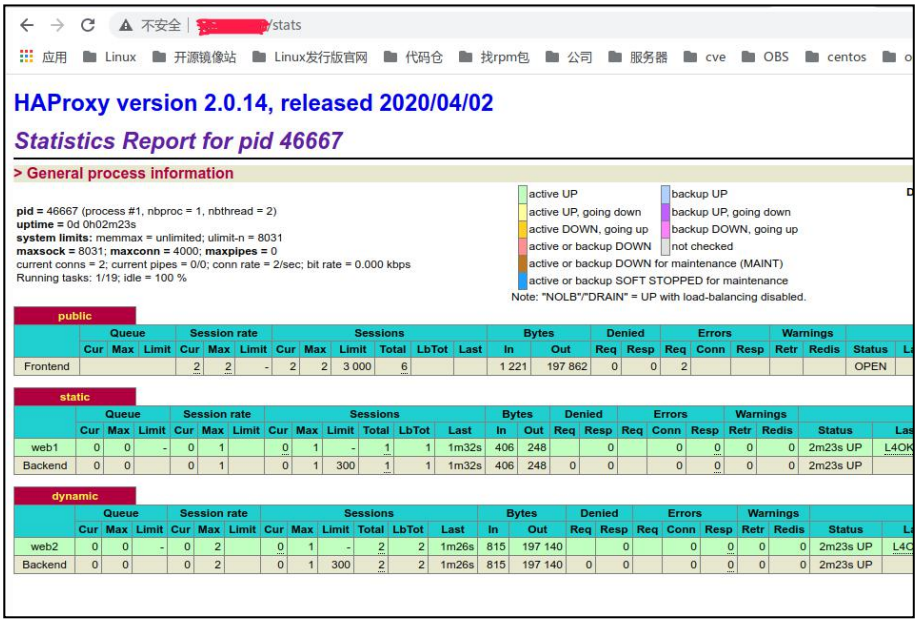


图 4.121 查看 HAProxy 服务状态

4.16 大数据-Flink

4.16.1 简介

Apache Flink 是一个框架和分布式处理引擎，用于在无界和有界数据流上进行有状态计算。Flink 被设计为在所有常见的集群环境中运行，以内存中的速度和任何规模执行计算。以下操作指导以 1.12.0 版本为例。

4.16.2 部署

1 执行如下命令，安装 flink 包。

```
# yum install flink -y
# yum install java-1.8.0* -y
```

2 执行如下命令，配置 JAVA_HOME。

```
# tail -n 4 /etc/profile
export
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-7.uel20.aarch64
export JRE_HOME=${JAVA_HOME}/jre
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
export PATH=${JAVA_HOME}/bin:${JAVA_HOME}/jre/bin:$PATH
```

3 执行如下命令，启动集群。

```
# cd /opt/apache-flink-1.12.0
# ./bin/start-cluster.sh
```

4.16.3 访问 web 页面

打开浏览器，访问 http://${localhost}:8081 检查分派器的 web 前端，并确保一切正常，从而启动一个只有一个 JobManager 和 TaskManager 的本地 Flink 集群。

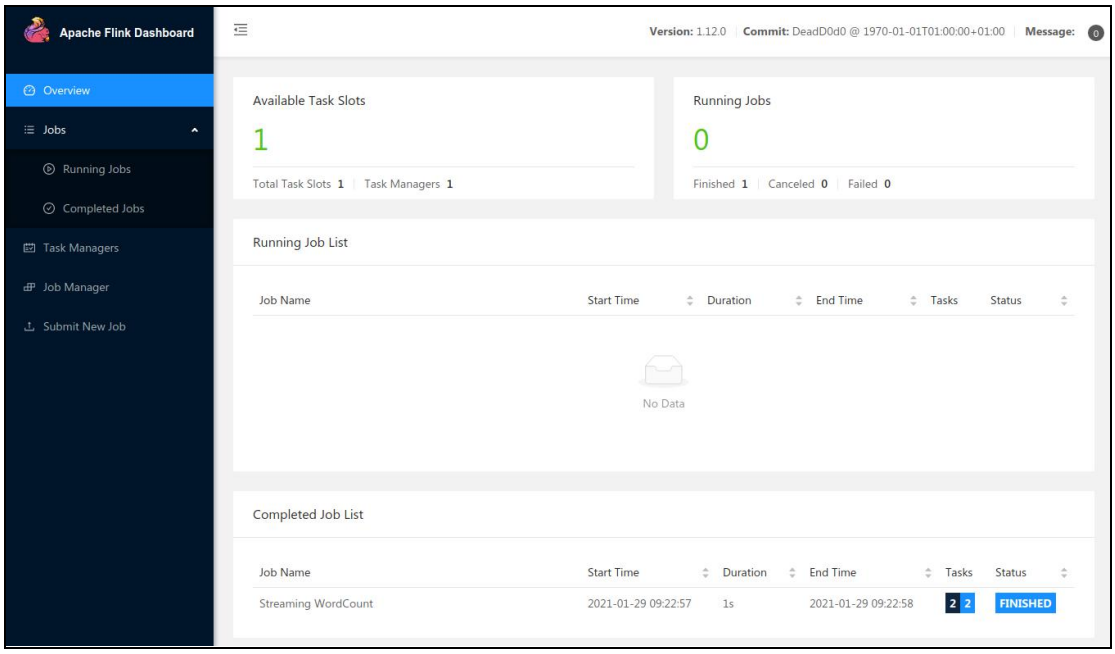


图 4.122 Flink 集群示例

4.16.4 查看日志

您还可以执行如下命令，通过检查 logs 目录中的日志文件来验证系统是否正在运行：

```
# tail log/flink-*-standalonesession-*.log
```

4.16.5 运行实例

每个 Flink 的 binary release 都会包含一个 examples（示例）目录，在其中可以找到这个页面上每个示例的 jar 包文件。

```
[root@localhost examples]# ls ./*
./batch:
ConnectedComponents.jar DistCp.jar EnumTriangles.jar KMeans.jar
PageRank.jar TransitiveClosure.jar WebLogAnalysis.jar
WordCount.jar

./gelly:
```

```
flink-gelly-examples_2.11-1.12.0.jar

./python:
table

./streaming:
IncrementalLearning.jar    SessionWindowing.jar
StateMachineExample.jar    Twitter.jar         WordCount.jar
Iteration.jar                    SocketWindowWordCount.jar
TopSpeedWindowing.jar       WindowJoin.jar

./table:
ChangelogSocketExample.jar    StreamSQLExample.jar
StreamWindowSQLExample.jar    WordCountSQL.jar
GettingStartedExample.jar      StreamTableExample.jar
TPCHQuery3Table.jar               WordCountTable.jar
```

实例 1 WordCount

WordCount 是大数据系统中的“**Hello World**”，他可以计算一个文本集合中不同单词的出现频次。可以通过执行以下命令来运行 WordCount 示例：

```
./bin/flink run ./examples/batch/WordCount.jar
```

说明：其他示例也可通过类似方式执行。注意很多示例在不传递执行参数的情况下都会使用内置数据。

如果需要利用 WordCount 程序计算真实数据，您需要传递存储数据的文件路径。例如：

注意：非本地文件系统需要一个对应前缀，例如 hdfs://。

```
# ./bin/flink run ./examples/batch/WordCount.jar --input ~/data --output ~/result
Job has been submitted with JobID e8630f0b63120d501b92d85e733e51d6
Program execution finished
Job with JobID e8630f0b63120d501b92d85e733e51d6 has finished.
Job Runtime: 462 ms
```

执行上述操作后，您可以检查 Web 界面以验证作业是否按预期运行：

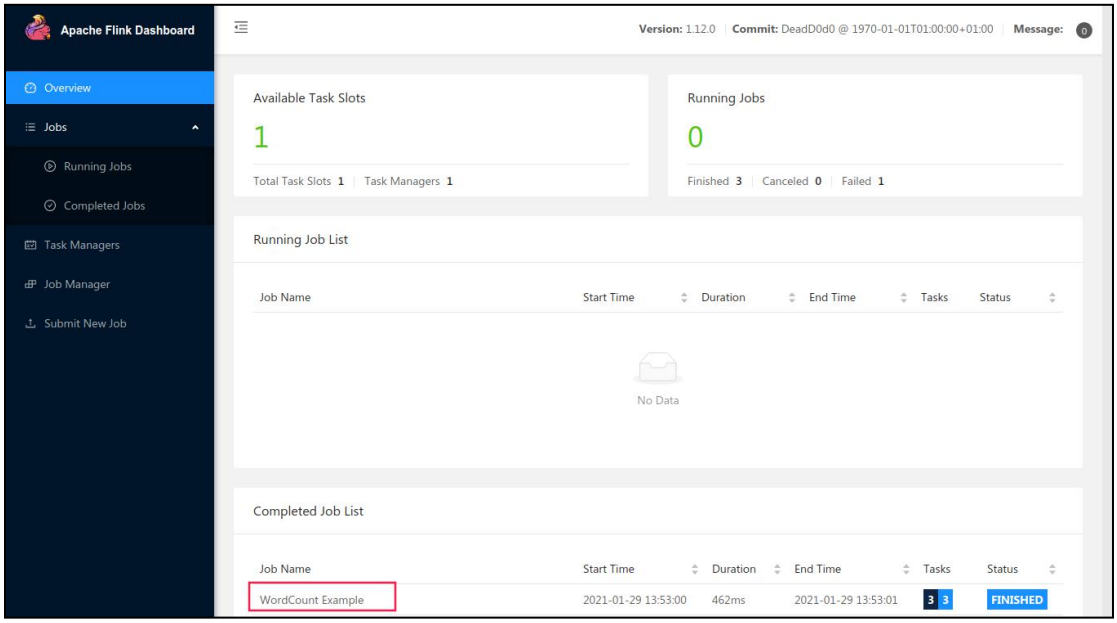


图 4.123 从 Web 端检查作业状态

您还可以执行如下命令，查看运行结果。

■ 输入文件

```
# cat ~/data
hello
how are you
fine
thank you
and you
I am
fine too
```

■ 输出结果

```
# cat ~/result
am 1
and 1
are 1
fine 2
hello 1
how 1
i 1
thank 1
too 1
you 3
```

实例 2 SocketWindowWordCount

现在，我们将运行此 Flink 应用程序。它将从套接字读取文本，每 5 秒钟打印一次在前 5 秒钟内每个单词出现的次数，即只要单词漂浮在其中，处理时间就会翻腾。

1 执行如下命令，启动本地服务器。

```
$ nc -l 9000
```

2 重新打开一个终端，提交 Flink 程序：

```
$ ./bin/flink run examples/streaming/SocketWindowWordCount.jar
--port 9000
Job has been submitted with JobID
2b2b0bf4ea55e45c17d0de52fb3a202b
```

3 回到刚才的 netcat 界面，输入一些字符，例如：

```
lorem ipsum
ipsum ipsum ipsum
bye
```

4 按“**Ctrl+c**”快捷键结束输入，可以看到第二个界面的程序执行成功。

```
[root@localhost apache-flink-1.12.0]# ./bin/flink run
examples/streaming/SocketWindowWordCount.jar --port 9000
Job has been submitted with JobID
2b2b0bf4ea55e45c17d0de52fb3a202b
Program execution finished
Job with JobID 2b2b0bf4ea55e45c17d0de52fb3a202b has finished.
Job Runtime: 123983 ms
```

5 检查 Web 界面，以验证作业是否按预期运行。

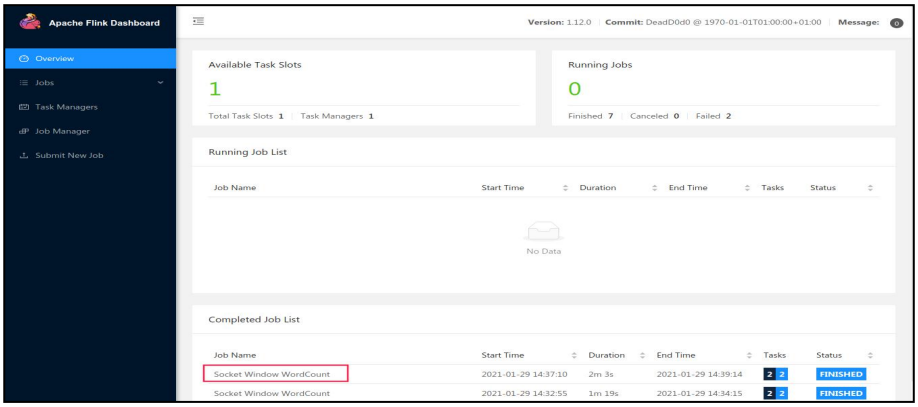


图 4.124 Web 端检查作业状态

6 执行如下命令，查看统计结果。

```
# tail log/flink-*-taskexecutor-*.out
```

```
lorem : 1
ipsum : 1
ipsum : 3
bye : 1
```

4.16.6 停止集群

```
# ./bin/stop-cluster.sh
Stopping taskexecutor daemon (pid: 18466) on host
localhost.localdomain.
Stopping standalonesession daemon (pid: 18209) on host
localhost.localdomain.
```

4.17 运维工具 AWX

4.17.1 概述

AWX 是一个开源社区项目，提供用于管理 Ansible 项目的软件。AWX 托管在 GitHub 上，并提供基于 Web 的用户界面、REST API 和适用于 Ansible 的任务引擎。

Ansible 是一款开发运营工具，可自动执行预置、配置管理、应用程序部署、内部服务编排、持续交付和许多其他 IT 流程。直观的 AWX 控制面板让您能够安排和部署 Ansible Playbook，并提供集中的日志记录、审计和系统跟踪。

4.17.2 安装 AWX

1 执行如下命令，安装 awx。

```
yum install awx -y
```

2 执行如下命令，配置 psotgresql 数据库。

```
postgresql-setup initdb
cp -f /var/lib/awx/setup/pg_hba.conf /var/lib/pgsql/data/
chmod a+r /var/lib/pgsql/data/pg_hba.conf
systemctl start postgresql
cd /var/lib/pgsql/data
su postgres
```

3 执行如下命令，创建一个初始数据库，数据库名称为 awx。

```
createdb awx
```

4 执行如下命令，创建超级用户 awx。创建时需要交互式输入密码。

```
createuser -s -P awx
exit
```

5 执行如下命令，配置 redis。

```
cp -f /var/lib/awx/setup/redis.conf /etc/redis.conf
chmod a+r /etc/redis.conf
```

6 执行如下命令，修改“**credentials.py**”配置文件。

```
vi /etc/tower/conf.d/credentials.py
DATABASES = {
    'default': {
        'ATOMIC_REQUESTS': True,
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': "awx",
        'USER': "awx",
        'PASSWORD': "awx",
        'HOST': "127.0.0.1",
        'PORT': "5432",
    }
}
BROADCAST_WEBSOCKET_SECRET =
"VXZiRklYMFdzdTFFdE5VSDg1UjJFUTBrZ3NGZEMwZklOQm1JUGxmanQ
5TE96QW5FSzM1Mlk0bVhodndUQW9KeXdOOXFaNUlMMVRVZXRXMzZ6
UVJKcHdqd0E5bWdIbHJKd0FxUWdZYTJDcjFWbjRnQW9vTzRMbjFEN2d
yZnN0eGo="
```

说明：

- *NAME：为步骤 2 中创建的数据库名。*
- *USER：为步骤 2 中创建的数据库用户名。*
- *PASSWORD：为步骤 2 中创建的数据库用户密码。*
- *BROADCAST_WEBSOCKET_SECRET：执行 **tr** -dc A-Za-z0-9 </dev/urandom | head -c 128 | base64 -w 0 命令生成的安全值。*

7 执行如下命令，初始化 awx。

```
awx-config
```

```
[root@localhost data]# awx-config
Operations to perform:
  Apply all migrations: auth, conf, contenttypes, main, oauth2_provider, sessions, sit
es, social_django, sso, taggit
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying taggit.0001_initial... OK
  Applying taggit.0002_auto_20150616_2121... OK
```

图 4.125 初始化 awx 测试

```
Superuser created successfully.
Password updated
Successfully registered instance awx
(changed: True)
Creating instance group tower
(changed: True)
Default organization added.
Demo Credential, Inventory, and Job Template added.
(changed: True)
Awx data initialization is complete
[root@localhost data]#
```

图 4.126 初始化 awx 成功

8 执行如下命令，启动 awx 服务。

ansible-awx-service start

说明：

- ***ansible-awx-service stop***：*停止 awx 服务。*
- ***ansible-awx-service status***：*查看 awx 服务。*

9 执行如下命令，关闭防火墙。

systemctl stop *firewalld*

10 打开浏览器，访问 http://destip:8052。用户名和密码均为“**admin**”。

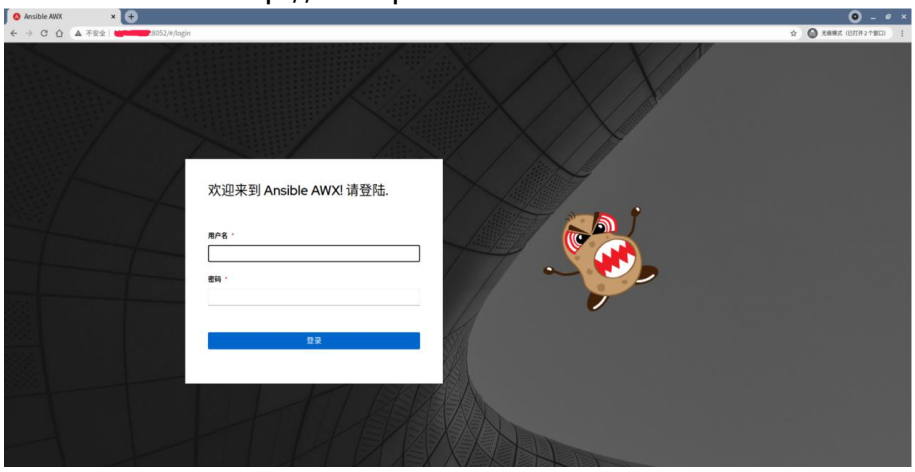


图 4.127 awx 登录页面示例

4.17.3 使用 AWX

配置邮箱通知

1 访问主页，在主菜单中，选择“**管理** > **通知**”后，单击“**添加**”。

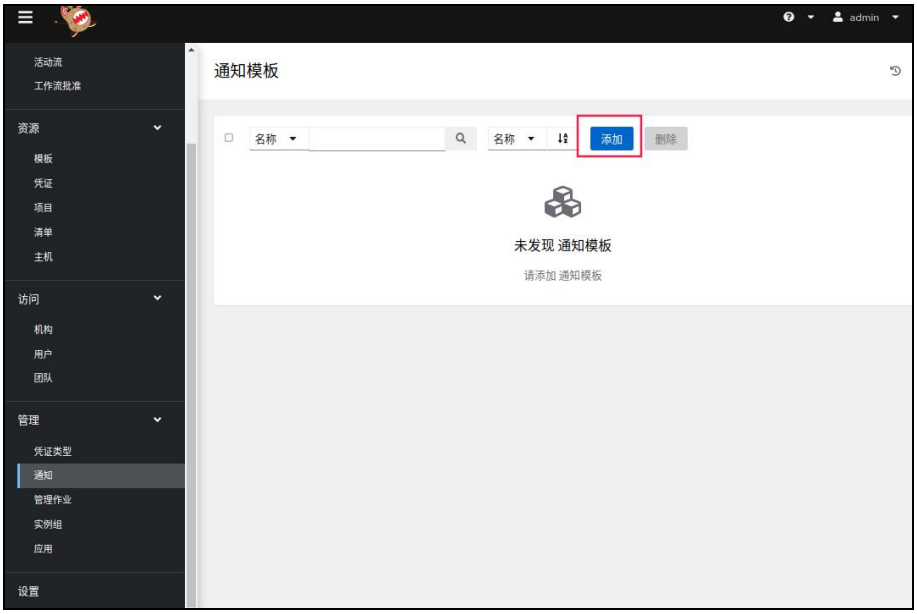


图 4.128 awx 添加通知示例

2 在“**类型**”下拉菜单中，选择“**电子邮件**”。填写相关信息后，单击“**保存**”。

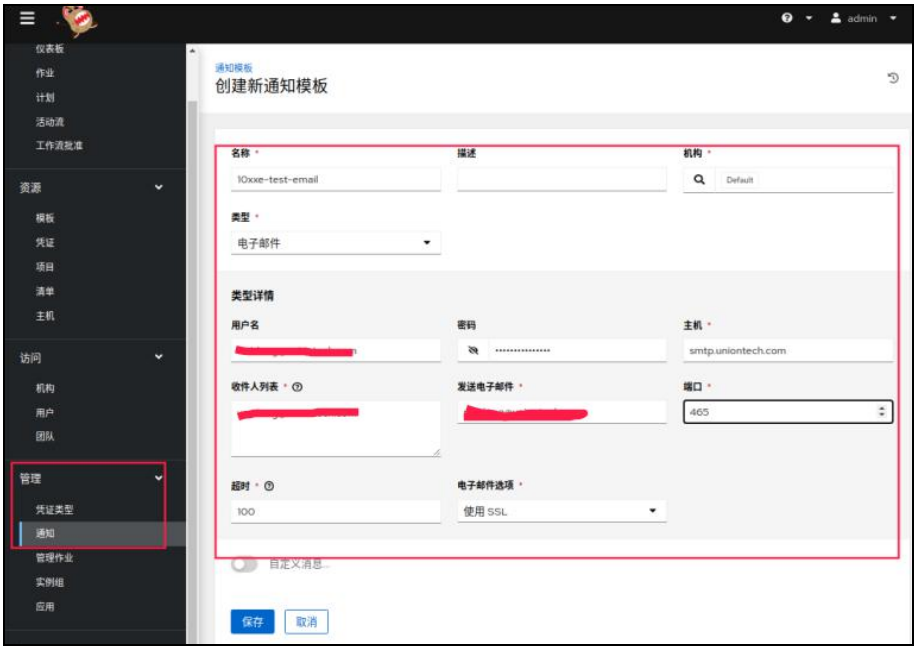


图 4.129 awx 创建通知模板

3 单击测试发送按钮，此时邮箱就可以收到测试信息。

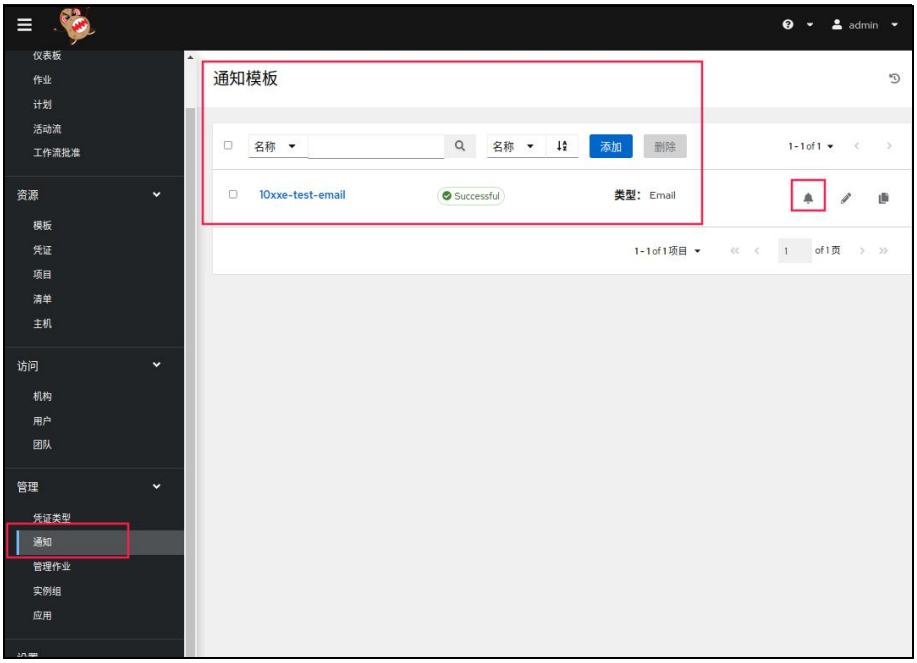


图 4.130 awx 测试发送通知到邮箱

配置任务运行

1 访问主页，在主菜单中，选择“**资源** > **清单**”后，单击“**添加**”。

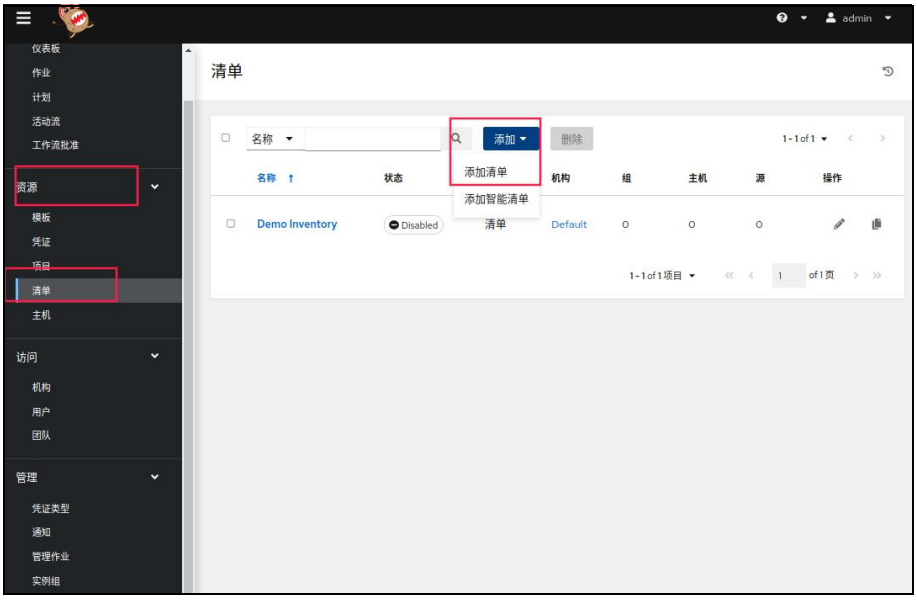


图 4.131 awx 添加资源清单

2 填写相关信息后，单击右下角“**保存**”。

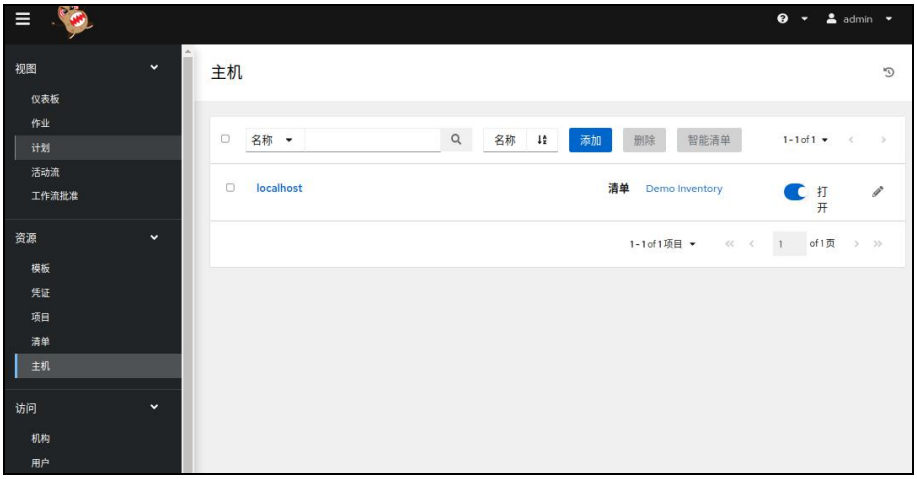


图 4.132 awx 保存资源清单

3 填写主机信息（“**名称**”：请填写主机 IP 地址）后，选择已创建的清单，并单击“**保存**”。

图 4.133 填写主机信息

4 访问主页，在主菜单中，选择“**资源** > **凭证**”后，单击“**添加**”。

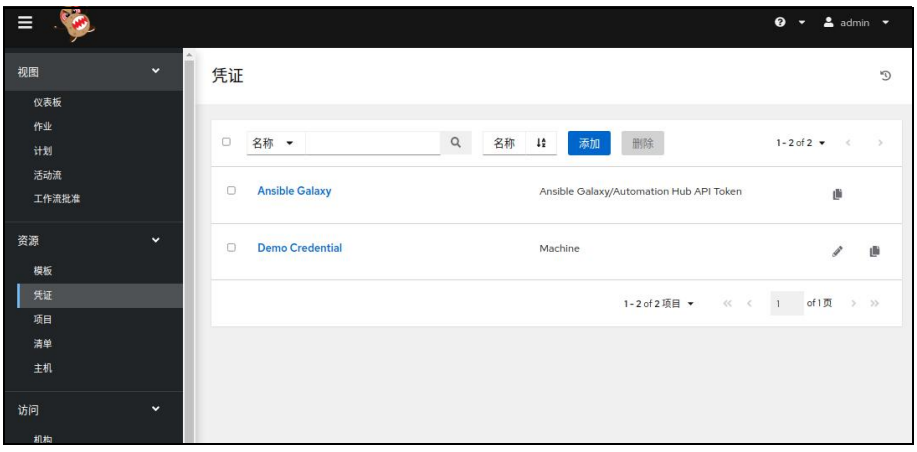


图 4.134 添加资源凭证

5 添加“**凭证类型**”为“**机器**”的凭证，单击“**保存**”。

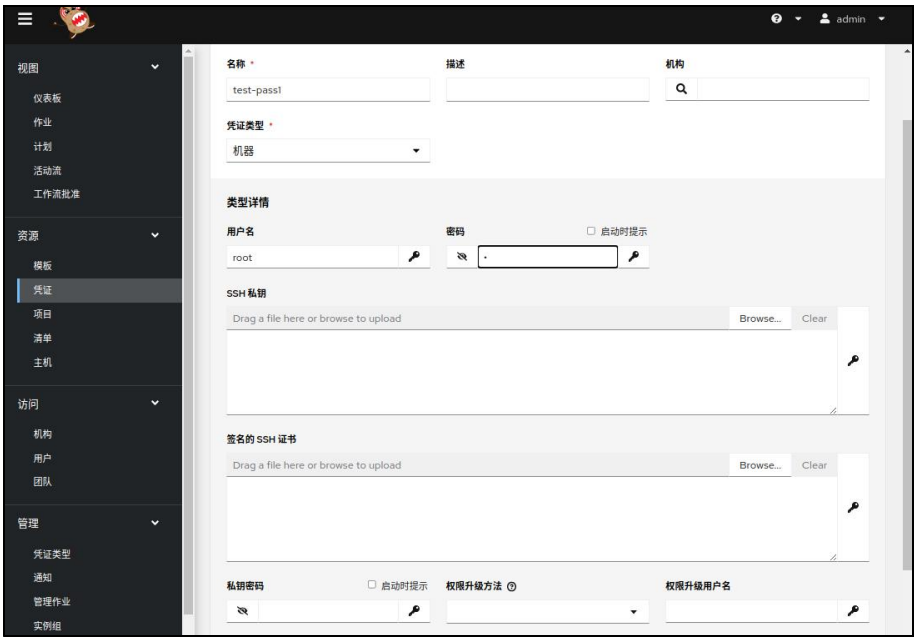


图 4.135 保存资源凭证

6 访问主页，在主菜单中，选择“**资源** > **项目**”后，单击“**添加**”，用于添加一个新的 playbook 项目。

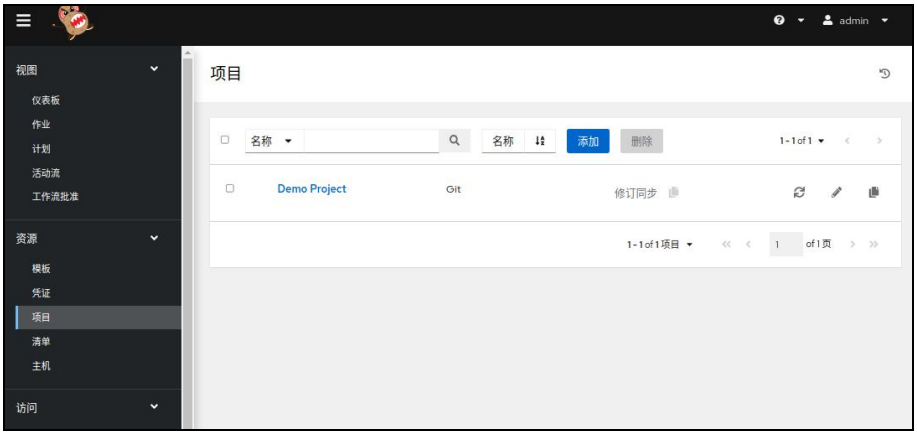


图 4.136 添加资源项目

7 在“**创建新项目**”界面中，将“**源控制凭据类型**”设置为“**Git**”，并配置测试项目是托管在 gitee 上的 test-ansible-project。完成配置后，单击“**保存**”。

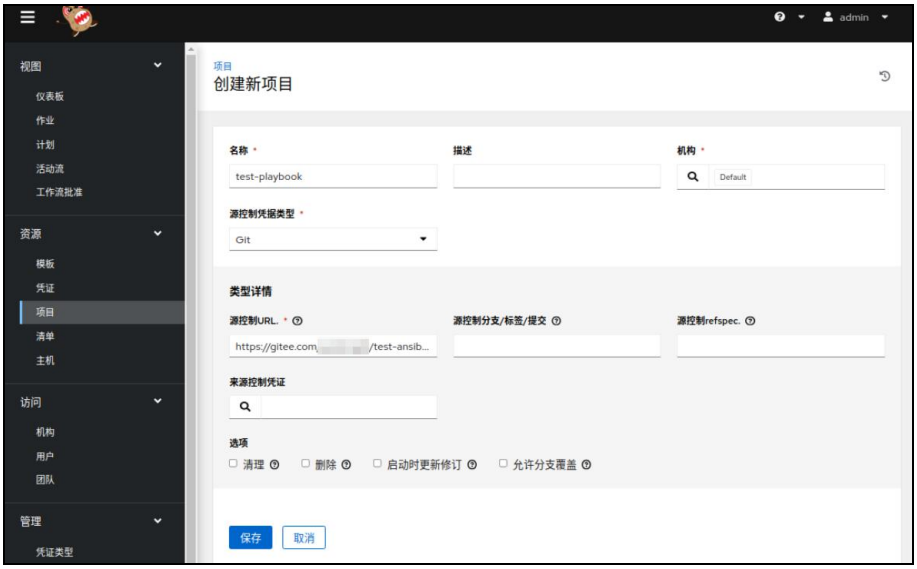


图 4.137 创建新项目

8 访问主页，在主菜单中，选择“**资源** > **模板**”后，单击“**添加** > **添加作业模板**”。

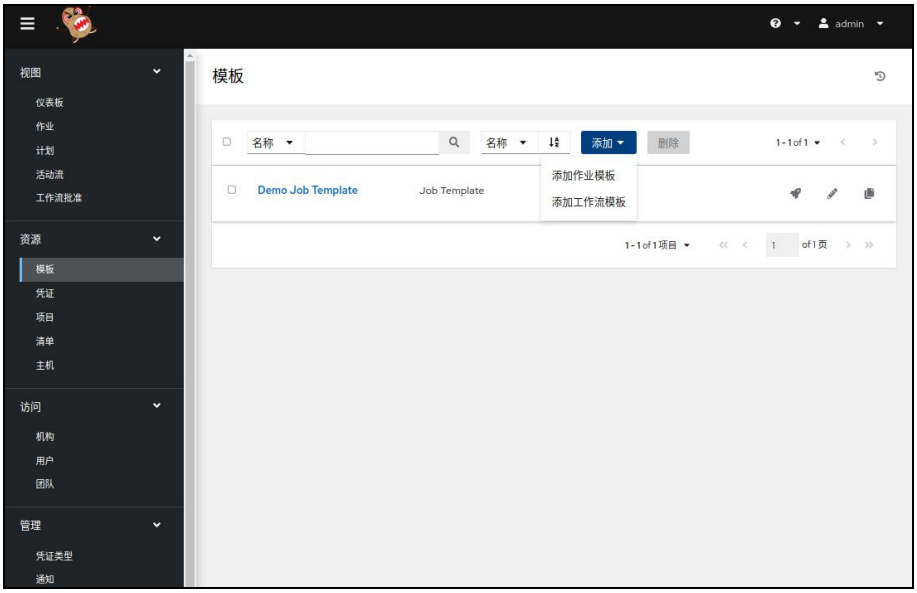


图 4.138 添加资源模板

9 在作业模板中，将“**清单**”、“**项目**”、“**凭证**”和“**playbook**”配置成已创建的相关信息后，单击“**保存**”。

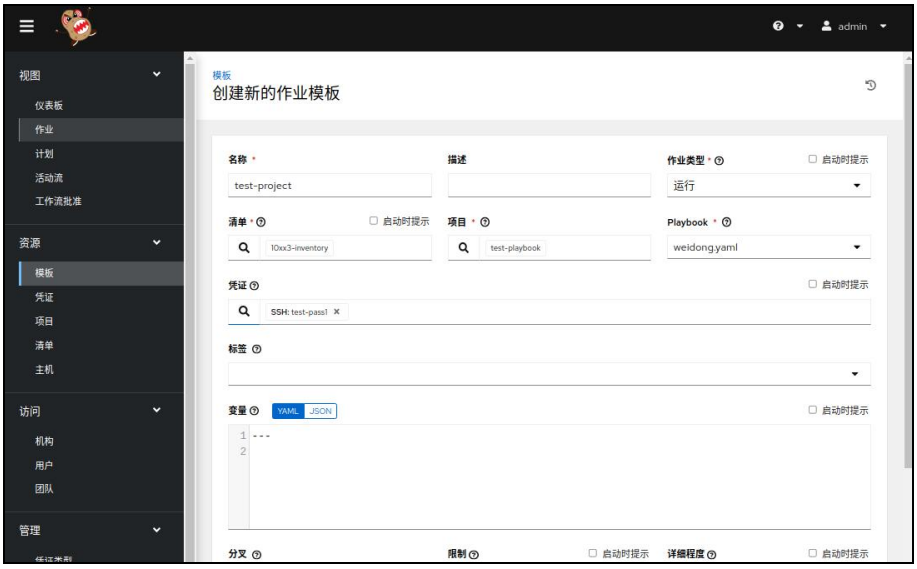


图 4.139 保存资源模板

10 访问主页，在主菜单中，选择“**资源** > **模板**”后，单击模板执行按钮。

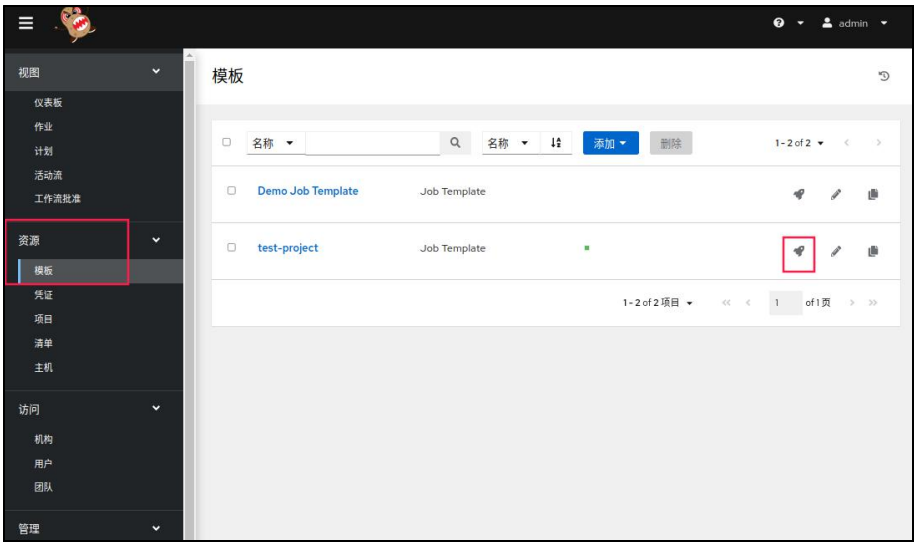


图 4.140 执行模板

11 访问主页，在主菜单中，选择“**视图** > **作业**”后，可看到任务已经在执行。

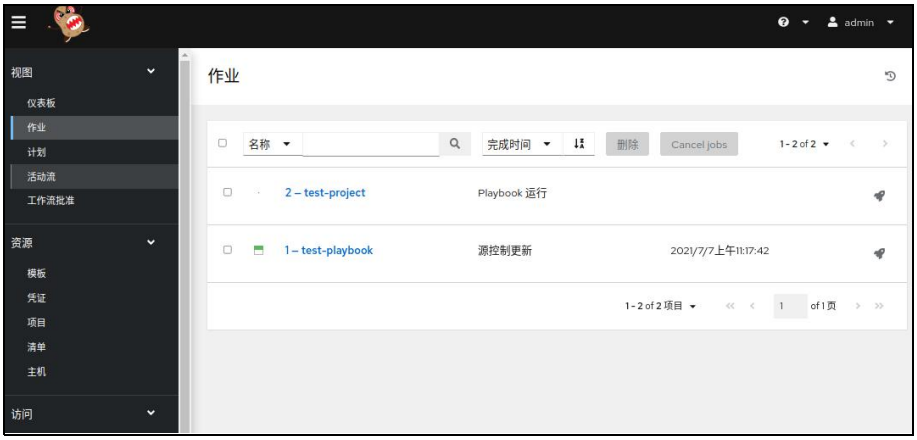


图 4.141 查看任务状态

4.18 系统智能性能调优工具 A-Tune

4.18.1 概述

A-Tune 是一款基于 AI 开发的系统性能优化引擎。它利用人工智能技术，对业务场景建立精准的系统画像，感知并推理出业务特征，进而做出智能决策，匹配并推荐最佳的系统参数配置组合，使业务处于最佳运行状态。

4.18.2 支持业务模型

表 4.13 A-tune 支持的业务模型

业务大类	业务类型	瓶颈点	支持的应用
default	默认类型	算力、内存、网络、IO 各维度资源使用率都不高	N/A
webserver	web 应用	算力瓶颈、网络瓶颈	Nginx、Apache Traffic Server
database	数据库	算力瓶颈、内存瓶颈、IO 瓶颈	Mongodb、Mysql、Postgresql、Mariadb
big-data	大数据	算力瓶颈、内存瓶颈	Hadoop-hdfs、Hadoop-spark
middleware	中间件框架	算力瓶颈、网络瓶颈	Dubbo
in-memory-database	内存数据库	内存瓶颈、IO 瓶颈	Redis
basic-test-suite	基础测试套	算力瓶颈、内存瓶颈	SPECCPU2006、SPECjbb2015
hpc	人类基因组	算力瓶颈、内存瓶颈、IO 瓶颈	Gatk4
storage	存储	网络瓶颈、IO 瓶颈	Ceph
virtualization	虚拟化	算力瓶颈、内存瓶颈、IO 瓶颈	Consumer-cloud、Mariadb
docker	容器	算力瓶颈、内存瓶颈、IO 瓶颈	Mariadb

4.18.3 使用限制

- A-Tune 不能和 tuned 工具一起使用，两者存在部分调优场景配置冲突。
- 只有 root 用户可以修改 A-Tune 场景静态识别功能配置文件“**/etc/atune/tune_rule.conf**”。

4.18.4 安装 A-Tune

安装模式介绍

A-Tune 支持单机模式和分布式模式安装：

- 单机模式：client 和 server 安装到同一台机器上。
- 分布式模式：client 和 server 分别安装在不同的机器上。

安装操作

1 执行如下命令，安装 A-Tune 服务端。

```
# yum install atune atune-engine -y
```

说明：本步骤会同时安装服务端和客户端软件包。

2 若为分布式部署，请执行如下命令，安装 A-Tune 客户端。若为单机部署，请跳过该步骤。

```
# yum install atune-client -y
```

3 执行如下命令，验证是否安装成功。若回显信息如下所示，表示安装成功。

```
# rpm -qa | grep atune
atune-client-xxx
atune-db-xxx
atune-xxx
atune-engine-xxx
```

4.18.5 配置 A-Tune

配置介绍

A-Tune 配置文件“**/etc/atuned/atuned.cnf**”的配置项说明如下所示：

- A-Tune 服务启动配置，可根据需要进行修改。
  - protocol：系统 grpc 服务使用的协议，unix 或 tcp。unix 为本地 socket 通信方式，tcp 为 socket 监听端口方式。默认为 unix。
  - address：系统 grpc 服务的侦听地址，默认为 unix socket。若为分布式部署，需修改为侦听的 IP 地址。
  - port：系统 grpc 服务的侦听端口，范围为 0~65535 未使用的端口。如果 protocol 配置是 unix，则不需要配置。
  - connect：若为集群部署时，atune 所在节点的 IP 列表，IP 地址以逗号隔开。
  - rest_host：系统 rest service 的侦听地址，默认为 localhost。
  - rest_port：系统 rest service 的侦听端口，范围为 0~65535 未使用的端口，默认为 8383。

- engine_host：与系统 atune engine service 链接的地址。
- engine_port：与系统 atune engine service 链接的端口。
- sample_num：系统执行 analysis 流程时采集样本的数量，默认为 20。
- interval：系统执行 analysis 流程时采集样本的间隔时间，默认为 5s。
- grpc_tls：系统 grpc 的 SSL/TLS 证书校验开关，默认不开启。开启 grpc_tls 后，atune-adm 命令在使用前需要设置以下环境变量方可与服务端进行通讯：

```
export ATUNE_TLS=yes
export ATUNED_CACERT=<客户端 CA 证书路径>
export ATUNED_CLIENTCERT=<客户端证书路径>
export ATUNED_CLIENTKEY=<客户端秘钥路径>
export ATUNED_SERVERCN=server
```

- tlsservercafile：gPRC 服务端 CA 证书路径。
- tlsservercertfile：gPRC 服务端证书路径。
- tlsserverkeyfile：gPRC 服务端秘钥路径。
- rest_tls：系统 rest service 的 SSL/TLS 证书校验开关，默认开启。
- tlsrestcacertfile：系统 rest service 的服务端 CA 证书路径。
- tlsrestservercertfile：系统 rest service 的服务端证书路径
- tlsrestserverkeyfile：系统 rest service 的服务端秘钥路径。
- engine_tls：系统 atune engine service 的 SSL/TLS 证书校验开关，默认开启。
- tlsenginecacertfile：系统 atune engine service 的客户端 CA 证书路径。
- tlsengineclientcertfile：系统 atune engine service 的客户端证书路径。
- tlsengineclientkeyfile：系统 atune engine service 的客户端秘钥路径。

■ system 信息：system 为系统执行相关的优化需要用到的参数信息，必须根据系统实际情况进行修改。

- disk：执行 analysis 流程时需要采集的对应磁盘的信息或执行磁盘相关优化时需要指定的磁盘。
- network：执行 analysis 时需要采集的对应的网卡的信息或执行网卡相关优化时需要指定的网卡。
- user：执行 ulimit 相关优化时用到的用户名。目前只支持 root 用户。

■ 日志信息：根据情况修改日志的级别，默认为 info 级别，日志信息打印在“**/var/log/messages**”中。

■ monitor 信息：为系统启动时默认采集的系统硬件信息。

- tuning 信息：tuning 为系统进行离线调优时需要用到的参数信息。
  - noise：高斯噪声的评估值。
  - sel_feature：控制离线调优参数重要性排名输出的开关，默认关闭。

配置示例 1

```
############## server ##############
# atuned config
[server]
# the protocol grpc server running on
# ranges: unix or tcp
protocol = unix

# the address that the grpc server to bind to
# default is unix socket /var/run/atuned/atuned.sock
# ranges: /var/run/atuned/atuned.sock or ip address
address = /var/run/atuned/atuned.sock

# the atune nodes in cluster mode, separated by commas
# it is valid when protocol is tcp
# connect = ip01,ip02,ip03

# the atuned grpc listening port
# the port can be set between 0 to 65535 which not be used
# port = 60001

# the rest service listening port, default is 8383
# the port can be set between 0 to 65535 which not be used
rest_host = localhost
rest_port = 8383

# the tuning optimizer host and port, start by engine.service
# if engine_host is same as rest_host, two ports cannot be same
# the port can be set between 0 to 65535 which not be used
engine_host = localhost
engine_port = 3838
```

```
# when run analysis command, the numbers of collected data.
# default is 20
sample_num = 20

# interval for collecting data, default is 5s
interval = 5

# enable gRPC authentication SSL/TLS
# default is false
# grpc_tls = false
# tlsservercafile = /etc/atuned/grpc_certs/ca.crt
# tlsservercertfile = /etc/atuned/grpc_certs/server.crt
# tlsserverkeyfile = /etc/atuned/grpc_certs/server.key

# enable rest server authentication SSL/TLS
# default is true
rest_tls = true
tlsrestcacertfile = /etc/atuned/rest_certs/ca.crt
tlsrestservercertfile = /etc/atuned/rest_certs/server.crt
tlsrestserverkeyfile = /etc/atuned/rest_certs/server.key

# enable engine server authentication SSL/TLS
# default is true
engine_tls = true
tlsenginecacertfile = /etc/atuned/engine_certs/ca.crt
tlsengineclientcertfile = /etc/atuned/engine_certs/client.crt
tlsengineclientkeyfile = /etc/atuned/engine_certs/client.key


############## log ##############
[log]
# either "debug", "info", "warn", "error", "critical", default is "info"
level = info
```

```
############## monitor ##############
[monitor]
# with the module and format of the MPI, the format is {module}_{purpose}
# the module is Either "mem", "net", "cpu", "storage"
# the purpose is "topo"
module = mem_topo, cpu_topo

############## system ##############
# you can add arbitrary key-value here, just like key = value
# you can use the key in the profile
[system]
# the disk to be analysis
disk = sda

# the network to be analysis
network = enp189s0f0

user = root

############## tuning ##############
# tuning configs
[tuning]
noise = 0.000000001
sel_feature = false
```

可根据需要修改 A-Tune engine 配置文件“**/etc/atuned/engine.cnf**”，可用配置项说明如下所示：

- engine_host：系统 atune engine service 的侦听地址，默认为 localhost。
- engine_port：系统 atune engine service 的侦听端口，范围为 0~65535 未使用的端口，默认为 3838。
- engine_tls：系统 atune engine service 的 SSL/TLS 证书校验开关，默认开启。
- tlsenginecacertfile：系统 atune engine service 的服务端 CA 证书路径。

- tlsengineservercertfile：系统 atune engine service 的服务端证书路径
- tlsengineserverkeyfile：系统 atune engine service 的服务端秘钥路径。
- 日志信息：根据情况修改日志的级别，默认为 info 级别，日志信息打印在 /var/log/messages 中。

配置示例 2

```
############## engine ##############
[server]
# the tuning optimizer host and port, start by engine.service
# if engine_host is same as rest_host, two ports cannot be same
# the port can be set between 0 to 65535 which not be used
engine_host = localhost
engine_port = 3838

# enable engine server authentication SSL/TLS
# default is true
engine_tls = true
tlsenginecacertfile = /etc/atuned/engine_certs/ca.crt
tlsengineservercertfile = /etc/atuned/engine_certs/server.crt
tlsengineserverkeyfile = /etc/atuned/engine_certs/server.key

############## log ##############
[log]
# either "debug", "info", "warn", "error", "critical", default is "info"
level = info
```

A-Tune 静态场景识别规则配置：配置文件以"[ ]"为章节，从上到下一次执行匹配该章节配置的正则式。正则式返回非空，则输出对应的场景名称；如果匹配到多个场景，则以第一个场景为准；如果没有配置到任何规则，则输出默认场景“**default-default**”。

配置示例 3

```
[oracle]
process=.*oracle.*

[mssql]
process=.*sqlservr.*
```

```
[atomic-host]
virt=
system=.*atomic.*

[atomic-guest]
virt=.+
system=.*atomic.*

[throughput-performance]
virt=
system=.*(computenode|server).*

[virtual-guest]
virt=.+

[balanced]
```

4.18.6 启动 A-Tune

A-Tune 安装完成后，需要启动 A-Tune 服务才能使用。

1 执行如下命令，启动 atuned 服务。

```
# systemctl start atuned
```

2 执行如下命令，查询 atuned 服务状态。

```
# systemctl status atuned
```

3 若回显信息如下图所示，表示服务启动成功。

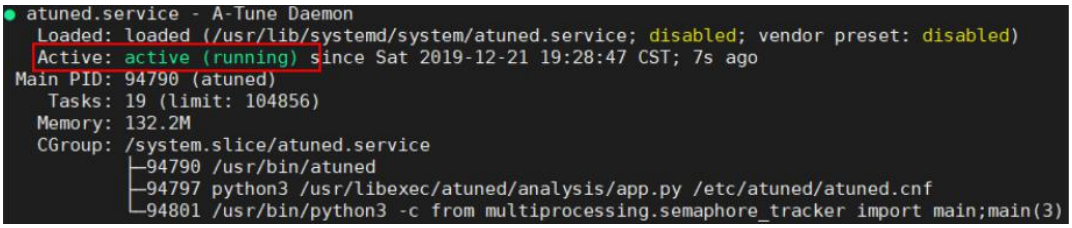


图 4.142 检查 atuned 服务状态

4.18.7 启动 A-Tune engine

若需要使用 AI 相关的功能，需要启动 A-Tune engine 服务才能使用。

1 执行如下命令，启动 atune-engine 服务。

```
# systemctl start atune-engine
```

2 执行如下命令，查询 atune-engine 服务状态。

```
# systemctl status atune-engine
```

3 若回显信息如下图所示，表示服务启动成功。

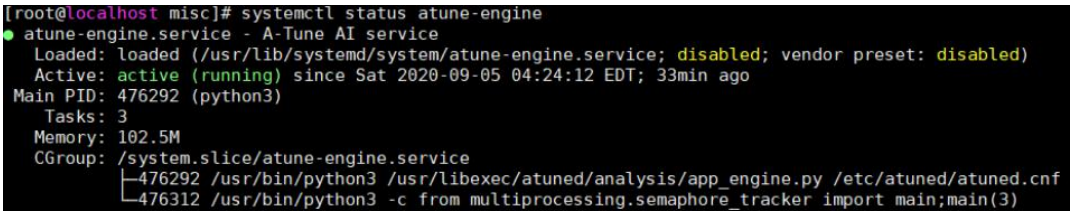


图 4.143 检查 atune-engine 服务状态

4.18.8 使用方法

用户可以通过命令行客户端 atune-adm 使用 A-Tune 提供的功能。本章介绍 A-Tune 客户端包含的功能和使用方法。

总体说明

- 使用 A-Tune 需要使用 root 权限。
- atune-adm 支持的命令可通过执行 **atune-adm** help/–help/-h 命令查询。
- 使用方法中所有命令的使用举例都是在单机部署模式下，如果是在分布式部署模式下，需要指定服务器 IP 地址和端口号，例如：

  ```
  # atune-adm -a 192.168.3.196 -p 60001 list
  ```

- define、update、undefine、collection、train、upgrade 不支持远程执行。
- 命令格式中，[ ]表示参数可选，<>表示参数必选，具体参数由实际情况确定。

查询负载类型

- 命令字：list
- 功能描述：查询系统当前支持的 profile，以及当前处于 active 状态的 profile。
- 命令格式：atune-adm list
- 使用示例：

```
# atune-adm list

Support profiles:
+-----------------------------------------------+----------+-------------------+
| ProfileName                                   | Active   | ProfileAttr       |
+===============================================+==========+===================+
| arm-native-android-container-robox            | false    |
```

```
model_analysis      |
+-------------------------------------------+----------+-------------------+
| atomic-guest-tuned                        | false    |
static_analysis     |
+-------------------------------------------+----------+-------------------+
| atomic-host-tuned                         | false    |
static_analysis     |
+-------------------------------------------+----------+-------------------+
| balanced-tuned                            | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-fio             | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-lmbench         | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-netperf         | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-stream          | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-unixbench       | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-fio             | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-lmbench         | false    |
manual_active       |
+-------------------------------------------+----------+-------------------+
| basic-test-suite-baseline-netperf         | false    |
manual_active       |
```

```
+-----------------------------------------------+-----------+-------------------+
| basic-test-suite-baseline-stream              | false     |
manual_active       |
+-----------------------------------------------+-----------+-------------------+
| basic-test-suite-baseline-unixbench           | false     |
manual_active       |
+-----------------------------------------------+-----------+-------------------+
| basic-test-suite-speccpu-speccpu2006          | false     |
model_analysis      |
+-----------------------------------------------+-----------+-------------------+
| basic-test-suite-specjbb-specjbb2015          | false     |
model_analysis      |
+-----------------------------------------------+-----------+-------------------+
| big-data-hadoop-hdfs-dfsio-hdd                | false     |
model_analysis      |
+-----------------------------------------------+-----------+-------------------+
| big-data-hadoop-hdfs-dfsio-ssd                | false     |
model_analysis      |
+-----------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-bayesian                | false     |
manual_active       |
+-----------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-kmeans                  | false     |
manual_active       |
+-----------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql1                    | false     |
manual_active       |
+-----------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql10                   | false     |
manual_active       |
+-----------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql2                    | false     |
manual_active       |
+-----------------------------------------------+-----------+-------------------+
```

```
| big-data-hadoop-spark-sql3                          | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql4                          | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql5                          | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql6                          | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql7                          | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql8                          | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-sql9                          | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-tersort                       | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| big-data-hadoop-spark-wordcount                     | false     |
manual_active      |
+-----------------------------------------------------+-----------+-------------------+
| cloud-compute-kvm-host                              | false     |
model_analysis     |
+-----------------------------------------------------+-----------+-------------------+
| database-mariadb-2p-tpcc-c3                         | false     |
model_analysis     |
+-----------------------------------------------------+-----------+-------------------+
| database-mariadb-4p-tpcc-c3                         | false     |
```

```
manual_active       |
+-----------------------------------------+----------+------------------+
| database-mongodb-2p-sysbench                | false    |
model_analysis      |
+-----------------------------------------+----------+------------------+
| database-mysql-2p-sysbench-hdd              | false    |
model_analysis      |
+-----------------------------------------+----------+------------------+
| database-mysql-2p-sysbench-ssd              | false    |
manual_active       |
+-----------------------------------------+----------+------------------+
| database-postgresql-2p-sysbench-hdd         | false    |
model_analysis      |
+-----------------------------------------+----------+------------------+
| database-postgresql-2p-sysbench-ssd         | false    |
manual_active       |
+-----------------------------------------+----------+------------------+
| default-default                             | false    |
model_analysis      |
+-----------------------------------------+----------+------------------+
| docker-mariadb-2p-tpcc-c3                   | false    |
model_analysis      |
+-----------------------------------------+----------+------------------+
| docker-mariadb-4p-tpcc-c3                   | false    |
manual_active       |
+-----------------------------------------+----------+------------------+
| fileserver-vsftpd-download                  | false    |
manual_active       |
+-----------------------------------------+----------+------------------+
| hpc-gatk4-human-genome                      | false    |
model_analysis      |
+-----------------------------------------+----------+------------------+
| in-memory-database-redis-redis-benchmark    | false    |
model_analysis      |
```

```
+-------------------------------------------+----------+-------------------+
| latency-performance-tuned                 | false    |
manual_active      |
+-------------------------------------------+----------+-------------------+
| middleware-dubbo-dubbo-benchmark          | false    |
model_analysis     |
+-------------------------------------------+----------+-------------------+
| mssql-tuned                               | false    |
static_analysis    |
+-------------------------------------------+----------+-------------------+
| network-latency-tuned                     | false    |
manual_active      |
+-------------------------------------------+----------+-------------------+
| network-throughput-tuned                  | false    |
manual_active      |
+-------------------------------------------+----------+-------------------+
| oracle-tuned                              | false    |
static_analysis    |
+-------------------------------------------+----------+-------------------+
| sap-hana-tuned                            | false    |
manual_active      |
+-------------------------------------------+----------+-------------------+
| sap-hana-vmware-tuned                     | false    |
manual_active      |
+-------------------------------------------+----------+-------------------+
| sap-netweaver-tuned                       | false    |
manual_active      |
+-------------------------------------------+----------+-------------------+
| storage-ceph-vdbench-hdd                  | false    |
model_analysis     |
+-------------------------------------------+----------+-------------------+
| storage-ceph-vdbench-ssd                  | false    |
manual_active      |
+-------------------------------------------+----------+-------------------+
```

```
| test_service-test_app-test_scenario                  | false    |
model_analysis     |
+------------------------------------------------------+----------+-------------------+
| throughput-performance-tuned                         | false    |
static_analysis    |
+------------------------------------------------------+----------+-------------------+
| virtual-guest-tuned                                  | true     |
static_analysis    |
+------------------------------------------------------+----------+-------------------+
| virtual-host-tuned                                   | false    |
manual_active      |
+------------------------------------------------------+----------+-------------------+
| virtualization-consumer-cloud-olc                    | false    |
model_analysis     |
+------------------------------------------------------+----------+-------------------+
| virtualization-mariadb-2p-tpcc-c3                    | false    |
model_analysis     |
+------------------------------------------------------+----------+-------------------+
| virtualization-mariadb-4p-tpcc-c3                    | false    |
manual_active      |
+------------------------------------------------------+----------+-------------------+
| web-apache-traffic-server-spirent-pingpo             | false    |
model_analysis     |
+------------------------------------------------------+----------+-------------------+
| web-nginx-http-long-connection                       | false    |
model_analysis     |
+------------------------------------------------------+----------+-------------------+
| web-nginx-https-short-connection                     | false    |
model_analysis     |
+------------------------------------------------------+----------+-------------------+
```

说明：Active 为 true 表示当前激活的 profile，例如，上述示例中的当前激活的 profile 是 virtual-guest-tuned，ProfileAttr 为 manual_active，表示需手动激活，为 model_analysis，表示可 AI 识别；为 static_analysis，表示可用 analysis-static 自动识别。

动态分析负载类型并自优化

- 命令字：analysis
- 功能描述：采集系统的实时统计数据进行负载类型识别，并进行自动优化。
- 命令格式：atune-adm analysis [选项]
- 参数说明：

表 4.14 analysis 参数说明

参数	描述
--model/-m	用户自训练产生的新模型。
--characterization/-c	使用默认的模型进行应用识别，不进行自动优化。

- 使用示例
  - 执行如下命令，使用默认的模型进行应用识别。

    # **atune-adm analysis** --characterization
  - 执行如下命令，使用默认的模型进行应用识别，并进行自动优化。

    # **atune-adm analysis**
  - 执行如下命令，使用自训练的模型进行应用识别。

    # **atune-adm analysis** --model */usr/libexec/atuned/analysis/models/new-model.m*

静态识别场景并自优化

- 命令字：analysis-static
- 功能描述：静态分析系统类型，并进行自动优化。
- 命令格式：atune-adm analysis-static [选项]
- 参数说明

表 4.15 analysis-static 参数说明

参数	描述
--characterization/-c	只静态分析系统负载，不进行调优。

- 使用示例
  - 执行如下命令，使用静态场景识别来识别系统当前场景。

    # **atune-adm analysis-static** --characterization
  - 执行如下命令，使用静态场景识别来识别系统当前场景，并进行自动优化。

    # **atune-adm analysis-static**

自定义模型

A-Tune 支持用户定义并学习新模型，定义新模型的操作流程如下：

1 用 define 命令定义一个新应用的 profile。

2 用 collection 命令收集应用对应的系统数据。

3 用 train 命令训练得到模型。

命令字：define

■ 功能描述：添加用户自定义的应用场景，及对应的 profile 优化项。

■ 命令格式：atune-adm define

■ 使用示例

1 执行如下命令，新增一个 profile。其中，service_type 的名称为 test_service，application_name 的名称为 test_app，scenario_name 的名称为 test_scenario，优化项的配置文件为 example.conf。

```
# atune-adm define test_service test_app
test_scenario ./example.conf
```

2 参考如下内容编写（以下各优化项非必填，仅供参考）“**example.conf**”，也可执行 **atune-adm info** 命令后参考已有的 profile 是如何书写的。

```
[main]
# list its parent profile
[kernel_config]
# to change the kernel config
[bios]
# to change the bios config
[bootloader.grub2]
# to change the grub2 config
[sysfs]
# to change the /sys/* config
[systemctl]
# to change the system service status
[sysctl]
# to change the /proc/sys/* config
[script]
# the script extention of cpi
[ulimit]
# to change the resources limit of user
[schedule_policy]
# to change the schedule policy
[check]
```

```
# check the environment
[tip]
# the recommended optimization, which should be performed manunaly
```

命令字：collection

- 功能描述：采集业务运行时系统的全局资源使用情况以及操作系统的各项状态信息，并将收集的结果保存到 csv 格式的输出文件中，作为模型训练的输入数据集。

  说明：

  - *本命令依赖采样工具 perf、mpstat、vmstat、iostat、sar。*
  - *CPU 型号目前仅支持鲲鹏 920，可执行* ***dmidecode*** *-t processor 命令检查 CPU 型号。*

- 命令格式：atune-adm collection
- 参数说明

表 4.16 collection 参数说明

参数	描述
–filename/-f	生成的用于训练的 csv 文件名 ,格式为 :名称-时间戳.csv。
–output_path/-o	生成的 csv 文件的存放路径，需提供绝对路径。
–disk/-b	业务运行时实际使用的磁盘，如/dev/sda。
–network/-n	业务运行时使用的网络接口，如 eth0。
–app_type/-t	标记业务的应用类型，作为训练时使用的标签。
–duration/-d	业务运行时采集数据的时间，单位秒，默认值为 1200 秒。
–interval/-i	采集数据的时间间隔，单位秒，默认值为 5 秒。

- 使用示例

  # **atune-adm collection** --filename *name* --interval *5* --duration *1200* --output_path */home/data* --disk *sda* --network *eth0* --app_type *test_type*

命令字：train

- 功能描述：使用采集的数据进行模型的训练。训练时至少采集两种应用类型的数据，否则训练会出错。
- 命令格式：atune-adm train
- 参数说明

表 4.17 train 参数说明

参数	描述
–data_path/-d	存放模型训练所需的 csv 文件的目录。

参数	描述
–output_file/-o	训练生成的新模型。

- 使用示例：执行如下命令，使用 data 目录下的 csv 文件作为训练输入，将生成的新模型 new-model.m 存放在 model 目录下。

```
# atune-adm train --data_path /home/data --output_file
/usr/libexec/atuned/analysis/models/new-model.m
```

命令字：undefine

- 功能描述：删除用户自定义的 profile。
- 命令格式：atune-adm undefine
- 使用示例：执行如下命令，删除自定义的 profile。

```
# atune-adm undefine test_service-test_app-test_scenario
```

查询 profile

- 命令字：info
- 功能描述：查看对应的 profile 内容。
- 命令格式：atune-adm info
- 使用示例：执行如下命令，查看 web-nginx-http-long-connection 的 profile 内容。

```
# atune-adm info web-nginx-http-long-connection

*** web-nginx-http-long-connection:

#
# nginx http long connection A-Tune configuration
#
[main]
include = default-default

[kernel_config]
#TODO CONFIG

[bios]
#TODO CONFIG

[bootloader.grub2]
```

```
iommu.passthrough = 1

[sysfs]
#TODO CONFIG

[systemctl]
sysmonitor = stop
irqbalance = stop

[sysctl]
fs.file-max = 6553600
fs.suid_dumpable = 1
fs.aio-max-nr = 1048576
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
kernel.shmmni = 4096
kernel.sem = 250 32000 100 128
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.ip_local_port_range = 1024       65500
net.ipv4.tcp_max_tw_buckets = 5000
net.core.somaxconn = 65535
net.core.netdev_max_backlog = 262144
net.ipv4.tcp_max_orphans = 262144
net.ipv4.tcp_max_syn_backlog = 262144
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 1
net.ipv4.tcp_syn_retries = 1
net.ipv4.tcp_fin_timeout = 1
net.ipv4.tcp_keepalive_time = 60
net.ipv4.tcp_mem = 362619       483495    725238
net.ipv4.tcp_rmem = 4096          87380    6291456
net.ipv4.tcp_wmem = 4096          16384    4194304
net.core.wmem_default = 8388608
```

```
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

[script]
prefetch = off
ethtool = -X {network} hfunc toeplitz

[ulimit]
{user}.hard.nofile = 102400
{user}.soft.nofile = 102400

[schedule_policy]
#TODO CONFIG

[check]
#TODO CONFIG

[tip]
SELinux provides extra control and security features to linux kernel.
Disabling SELinux will improve the performance but may cause
security risks. = kernel
disable the nginx log = application
```

更新 profile

用户可根据需要更新已有 profile。

- 命令字：update
- 功能描述：将已有 profile 中原来的优化项更新为 new.conf 中的内容。
- 命令格式：atune-adm update
- 使用示例：执行如下命令，更新名为 test_service-test_app-test_scenario 的 profile 优化项为 new.conf。

```
# atune-adm update test_service-test_app-test_scenario ./new.conf
```

激活 profile

- 命令字：profile
- 功能描述：手动激活 profile，使其处于 active 状态。

- 命令格式：atune-adm profile
- 参数说明：profile 名参考 list 命令查询结果。
- 使用示例：执行如下命令，激活 web-nginx-http-long-connection 对应的 profile 配置。

  # **atune-adm profile** *web-nginx-http-long-connection*

回滚 profile

- 命令字：rollback
- 功能描述：回退当前的配置到系统的初始配置。
- 命令格式：atune-adm rollback
- 使用示例

  # **atune-adm rollback**

更新数据库

- 命令字：upgrade
- 功能描述：更新系统的数据库。
- 命令格式：atune-adm upgrade
- 参数说明：DB_FILE 表示新的数据库文件路径。
- 使用示例：执行如下命令，数据库更新为 new_sqlite.db。

  # **atune-adm upgrade** *./new_sqlite.db*

系统信息查询

- 命令字：check
- 功能描述：检查系统当前的 cpu、bios、os、网卡等信息。
- 命令格式：atune-adm check
- 使用示例：

```
# atune-adm check
cpu information:
cpu:0 version: pc-i440fx-4.0 speed: 2000000000 HZ cores: 1
cpu:1 version: pc-i440fx-4.0 speed: 2000000000 HZ cores: 1
system information:
DMIBIOSVersion: rel-1.12.1-0-ga5cab58e9a3f-prebuilt.qemu.org
OSRelease: 4.19.90-2106.3.0.0095.up2.uel20.x86_64
network information:
name: eth0 product: Virtio network device
name: eth1 product: RTL-8100/8101L/8139 PCI Fast Ethernet Adapter
```

```
name: docker0 product:
```

参数自调优

A-Tune 提供了最佳配置的自动搜索能力，免去人工反复做参数调整、性能评价的调优过程，极大地提升最优配置的搜寻效率。

- 命令字：tuning
- 功能描述：使用指定的项目文件对参数进行动态空间的搜索，找到当前环境配置下的最优解。
- 命令格式：atune-adm tuning [选项]

说明：在运行命令前，需要满足如下条件。

- *服务端的 yaml 配置文件已经编辑完成并放置于 atuned 服务下的 “**/etc/atuned/tuning/**” 目录中。*
- *客户端的 yaml 配置文件已经编辑完成并放置于 atuned 客户端任意目录下。*

- 参数说明

表 4.18 tuning 参数描述

参数	描述
--restore/-r	恢复 tuning 优化前的初始配置。
--project/-p	指定需要恢复的 yaml 文件中的项目名称。
--restart/-c	基于历史调优结果进行调优。
--detail/-d	打印 tuning 过程的详细信息。

说明：当使用参数时，-p 参数后需要跟具体的项目名称且必须指定该项目 yaml 文件。PROJECT_YAML 表示客户端 yaml 配置文件。

- 配置说明

表 4.19 服务端 yaml 文件

配置名称	配置说明	参数类型	取值范围
project	项目名称。	字符串	-
startworkload	待调优服务的启动脚本。	字符串	-
stopworkload	待调优服务的停止脚本。	字符串	-
maxiterations	最大调优迭代次数，用于限制客户端的迭代次数。一般来说，调优迭代次数越多，优化效果越好，但所需时间越长。用户必须根据实际的业务场景进行配置。	整型	>10
object	需要调节的参数项及信息。object 配置项请参见下表。	-	-

表 4.20 object 项配置说明

配置名称	配置说明	参数类型	取值范围
name	待调参数名称。	字符串	-
desc	待调参数描述。	字符串	-
get	查询参数值的脚本。	-	-
set	设置参数值的脚本。	-	-
needrestart	参数生效是否需要重启业务。	枚举	"true" ,"false"
type	参数的类型，目前支持 discrete、continuous 两种类型，对应离散型、连续型参数。	枚举	"discrete"，"continuous"
dtype	该参数仅在 type 为 discrete 类型时配置，目前支持 int、float、string 类型。	枚举	int，float，string
scope	参数设置范围，仅在 type 为 discrete 且 dtype 为 int、float，或 type 为 continuous 时生效。	整型/浮点型	用户自定义 ,取值在该参数的合法范围
step	参数值步长，dtype 为 int 或 float 时使用。	整型/浮点型	用户自定义
items	参数值在 scope 定义范围之外的枚举值，dtype 为 int 或 float 时使用。	整型/浮点型	用户自定义 ,取值在该参数的合法范围
options	参数值的枚举范围，dtype 为 string 时使用。	字符串	用户自定义 ,取值在该参数的合法范围

表 4.21 客户端 yaml 文件配置说明

配置名称	配置说明	参数类型	取值范围
project	项目名称，需要与服务端对应配置文件中的 project 匹配。	字符串	-
engine	调优算法。	字符串	"random"，"forest"，"gbrt"，"bayes"，"extraTrees"

配置名称	配置说明	参数类型	取值范围
iterations	调优迭代次数。	整型	>=10
random_starts	随机迭代次数。	整型	<iterations
feature_filter_engine	参数搜索算法，用于重要参数选择，该参数可选。	字符串	"lhs"
feature_filter_cycle	参数搜索轮数，用于重要参数选择，该参数配合 feature_filter_engine 使用。	整型	-
feature_filter_iters	每轮参数搜索的迭代次数，用于重要参数选择，该参数配合 feature_filter_engine 使用。	整型	-
split_count	调优参数取值范围中均匀选取的参数个数，用于重要参数选择，该参数配合 feature_filter_engine 使用。	整型	-
benchmark	性能测试脚本。	-	-
evaluations	性能测试评估指标 evaluations 配置项请参见下表。	-	-

表 4.22 evaluations 项配置说明

配置名称	配置说明	参数类型	取值范围
name	评价指标名称。	字符串	-
get	获取性能评估结果的脚本。	-	-
type	评估结果的正负类型，positive 代表最小化性能值，negative 代表最大化对应性能值。	枚举	"positive", "negative"
weight	该指标的权重百分比。	整型	0-100
threshold	该指标的最低性能要求。	整型	用户指定

■ 配置示例

◆ 服务端 yaml 文件配置示例：

```
project: "compress"
maxiterations: 500
startworkload: ""
stopworkload: ""
object :
-
name : "compressLevel"
info :
desc : "The compresslevel parameter is an integer from 1 to 9 controlling the level of compression"
get : "cat /root/A-Tune/examples/tuning/compress/compress.py | grep 'compressLevel=' | awk -F '=' '{print $2}'"
set : "sed -i 's/compressLevel=\\s*[0-9]*/compressLevel=$value/g' /root/A-Tune/examples/tuning/compress/compress.py"
needrestart : "false"
type : "continuous"
scope :
- 1
- 9
dtype : "int"
-
name : "compressMethod"
info :
desc : "The compressMethod parameter is a string controlling the compression method"
get : "cat /root/A-Tune/examples/tuning/compress/compress.py | grep 'compressMethod=' | awk -F '=' '{print $2}' | sed 's/\"//g'"
set : "sed -i 's/compressMethod=\\s*[0-9,a-z,\"]*/compressMethod=\"$value\"/g' /root/A-Tune/examples/tuning/compress/compress.py"
needrestart : "false"
type : "discrete"
options :
```

```
- "bz2"
- "zlib"
- "gzip"
dtype : "string"
```

◆ 客户端 yaml 文件配置示例：

```
project: "compress"
engine : "gbrt"
iterations : 20
random_starts : 10

benchmark : "python3
/root/A-Tune/examples/tuning/compress/compress.py"
evaluations :
-
name: "time"
info:
get: "echo '$out' | grep 'time' | awk '{print $3}'"
type: "positive"
weight: 20
-
name: "compress_ratio"
info:
get: "echo '$out' | grep 'compress_ratio' | awk '{print $3}'"
type: "negative"
weight: 80
```

■ 使用示例

◆ 执行如下命令，进行 tuning 调优。

```
# atune-adm tuning --project compress --detail
compress_client.yaml
```

◆ 执行如下命令，恢复 tuning 调优前的初始配置。其中，compress 为 yaml 文件中的项目名称。

```
# atune-adm tuning --restore --project compress
```

4.18.9 常见问题

请参见“**A-Tune 常见问题**”章节。

5 服务器安全管理

5.1 PAM

PAM 即可插拔认证模块（Pluggable Authentication Module--PAM）。它采用模块化设计和插件功能，使用户可以轻易地在应用程序中插入新的认证模块或替换原先的组件，同时不需要对应用程序做任何修改，从而使软件的定制、维持和升级更加轻松。因为认证和鉴别机制与应用程序之间相对独立，所以应用程序可以通过 PAM API，来方便地使用 PAM 提供的各种鉴别功能，而不需要了解太多的底层细节。

5.1.1 配置文件

PAM 配置文件放到“**/etc/pam.d/**”目录下。如下图所示，第一列代表模块类型、第二列代表控制标记、第三列代表模块路径、第四列代表模块参数。

```
[root@localhost mysql]# cat /etc/pam.d/system-auth
#%PAM-1.0
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        required      pam_faillock.so preauth audit deny=3 even_deny_root unlock_time=60
-auth        sufficient    pam_fprintd.so
auth        sufficient    pam_unix.so nullok try_first_pass
-auth        sufficient    pam_sss.so use_first_pass
auth        [default=die] pam_faillock.so authfail audit deny=3 even_deny_root unlock_time=60
auth        sufficient    pam_faillock.so authsucc audit deny=3 even_deny_root unlock_time=60
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required      pam_deny.so

account     required      pam_unix.so
account     required      pam_faillock.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
-account     [default=bad success=ok user_unknown=ignore] pam_sss.so
account     required      pam_permit.so

password    requisite     pam_pwquality.so try_first_pass local_users_only
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_authtok
-password    sufficient    pam_sss.so use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
-session     optional      pam_systemd.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session     required      pam_unix.so
-session     optional      pam_sss.so
```

图 5.1 PAM 文件内容示例

5.1.2 模块类型

Linux-PAM 有四种模块类型，分别代表四种不同的任务，它们是：认证管理（auth）、帐号管理（account）、会话管理（session）和密码（password）管理。一个类型可能有多行，它们按顺序依次由 PAM 模块调用，管理方式如下：

- auth：用来对用户的身份进行识别。如：提示用户输入密码，或判断用户是否为 root 等。
- account：对帐号的各项属性进行检查。如：是否允许登录、是否达到最大用户数，或是 root 用户是否允许在这个终端登录等。
- session：用来定义用户登录前的，及用户退出后所要进行的操作。如：登录连接信息、用户数据的打开与关闭、挂载文件系统等。

■ password：使用用户信息来更新。如：修改用户密码。

5.1.3 控制标记

PAM 使用控制标记来处理和判断各个模块的返回值。

■ required：表示即使某个模块对用户的验证失败，也要等所有的模块都执行完毕后，PAM 才返回错误信息。这样做是为了不让用户知道被哪个模块拒绝。如果对用户验证成功，所有的模块都会返回成功信息。

■ requisite：与 required 相似，但是如果这个模块返回失败，则立刻向应用程序返回失败，表示此类型失败，而不再进行同类型后面的操作。

■ sufficient：表示如果一个用户通过这个模块的验证，PAM 结构就立刻返回验证成功信息（即使前面有模块 fail 了，也会把 fail 结果忽略掉），把控制权交回应用程序。后面的层叠模块即使使用 requisite 或者 required 控制标志，也不再执行。如果验证失败，sufficient 的作用和 optional 相同。

■ optional：表示即使本行指定的模块验证失败，也允许用户接受应用程序提供的服务，一般返回 PAM_IGNORE(忽略)。

5.1.4 模块路径

模块路径即要调用模块的位置。统信服务器操作系统保存在“**/etc/security**”，可以出现在不同的类型中。它在不同的类型中所执行的操作都不相同。这是由于每个模块针对不同的模块类型，编制了不同的执行函数。

5.1.5 常用模块介绍

pam_limits.so

■ 概述：统信服务器操作系统提供了 pam_limits.so 用户资源限制模块。该模块的主要功能是限制用户会话过程中对各种系统资源的使用情况。系统支持通过配置 limits.conf 实现对用户、用户组可使用的系统资源进行分配和限制，从而确保用户和主体不会独占某种受控的资源。系统中的资源分配按最大额的要求进行分配。

■ 配置文件：统信服务器操作系统中用户资源限制的配置文件是“**/etc/security/limits.conf**”。

■ 功能测试

1 执行如下命令，创建测试用户。

```
[root@localhost pam.d]# useradd test1
[root@localhost pam.d]# useradd test2
[root@localhost pam.d]# passwd test1
Changing password for user test1.
```

```
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost pam.d]# passwd test2
Changing password for user test2.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

2 执行如下命令，在配置文件“**/etc/security/limits.conf**”中添加如下所示的内容，限制用户 test1 登录到 SSH 服务器时的最大连接数。

```
[root@localhost pam.d]# vim /etc/security/limits.conf
test1    hard    maxlogins 2
```

3 在另外一台测试机上，使用 SSH 连接本机服务器。如下图所示，可以正常开启两个终端连接到服务器。当登录第三个时，提示“**Too many logins for ‘test1’**”信息，无法登录 ssh 服务器。

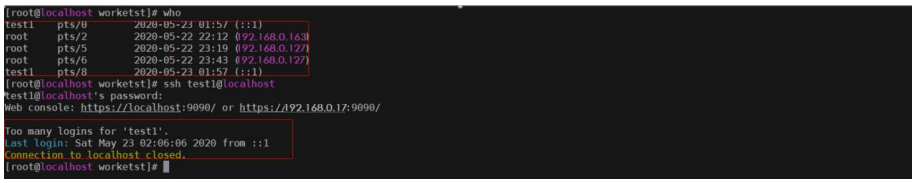


图 5.2 ssh 连接测试

pam_cracklib.so

■ 概述：统信服务器操作系统提供了 pam_cracklib.so 模块对用户登录系统的密码进行了安全加固。该模块主要的作用是对用户密码的强健性进行检测，即检查和限制用户自定义密码的长度、复杂度和历史等。用户第一次登录系统后被要求强制修改登录密码，并且对用户密码进行强度校验，即采用强化密码管理，并在每次用户登录系统时进行鉴别。通过对 pam 模块相关配置文件的修改，就可以实现一定的用户密码复杂度。

■ 常用选项

表 5.1 参数描述

选项	说明
retry=N	定义登录/修改密码失败时，可以重试的次数。
difok=N	定义新密码中必须至少有几个字符要与旧密码不同。但是如果新

选项	说明
	密码中有 1/2 以上的字符与旧密码不同时，该新密码将被接受。
minlen=N	定义用户密码的最小长度。
dcredit=N	定义用户密码中必须至少包含多少个数字。
ucredit=N	定义用户密码中必须至少包含多少个大写字母。
lcredit=N	定义用户密码中必须至少包含多少个小些字母。
ocredit=N	定义用户密码中必须至少包含多少个特殊字符（除数字 、字母外）。

说明：参数 credit=-1，表示至少有一个的意思。

■ 功能测试

1 执行如下命令，在配置文件的最后增加 pam_cracklib.so 的配置。

```
[root@localhost pam.d]# cat passwd
#%PAM-1.0
# This tool only uses the password stack.
password     substack      system-auth
-password    optional      pam_gnome_keyring.so use_authtok
password     substack      postlogin
password     requisite pam_cracklib.so retry=3 minlen=8 difok=3
[root@localhost pam.d]#
```

2 修改用户密码，使用 12345678 与 123456 进行密码测试。

图 5.3 修改密码示例

pam_access.so

■ 概述：统信服务器操作系统中提供了 pam_access.so 模块实现用户登录 IP 限制 。该模块主要的功能和作用是根据主机名（包括普通主机名或者 FQDN）、IP 地址和用户实现全面的访问控制。此模块的具体工作行为根据该模块的相应配置文件来决定。

■ 配置文件：统信服务器操作系统中用户登录时间限制的配置文件为“**/etc/security/access.conf**”。该配置文件包含了三个字段，即权限、用户和访问发起方，用“:”隔开。

- ◆ 第一个字段：权限（permission）。“+”表示授予权限，“-”表示禁止权限。
- ◆ 第二个字段：用户（user）。定义了用户、组，以及使用“@”表示的在不同主机上的相同用户和同一主机上的不同用户。
- ◆ 第三个字段：访问发起方（origins）。定义了发起访问的主机名称、域名称、终端名称。

■ 功能测试

1 执行如下命令，创建测试用户 testuser1。

```
[root@localhost pam.d]# useradd testuser1
```

2 执行 **sudo vi** */etc/pam.d/sshd* 命令，在配置文件中添加如下行。

```
account required pam_access.so
```

3 执行 **vi** */etc/security/access.conf* 命令，在文件中添加内容如下：

```
#允许用户 testuser1 从 IP 地址为 192.168.100.6 的主机远程登录本机
+:testuser1:192.168.100.6
#拒绝用户 testuser1 从 IP 地址为 192.168.100.20 的主机远程登录本机
-:testuser1:192.168.100.20
```

4 在 192.168.100.6 的终端，使用用户 testuser1、以 SSH 方式登录远程主机，显示结果为用户 testuser1 远程登录成功。

5 在 192.168.100.20 的终端，使用用户 testuser1、以 SSH 方式登录远程主机，显示结果为登录失败。

pam_time.so

■ 概述：统信服务器操作系统中提供了 pam_time.so 模块实现用户登录时间的限制，该模块主要的功能和作用是能控制用户在指定的时间段登录系统。

■ 配置文件：登录统信服务器操作系统的方式有三种，包括 tty 方式、ssh 登录方式和图形登录方式。根据登录方式的不同，对应的配置文件如下。

- ◆ /etc/pam.d/login：tty 登录方式的配置文件。
- ◆ /etc/pam.d/sshd：ssh 登录方式的配置文件。
- ◆ /etc/pam.d/lightdm：图形登录方式的配置文件。

其中，限制用户登录时间的配置文件为：/etc/security/time.conf。

■ 功能测试

1 执行如下命令，创建测试用户 testuser1。

```
useradd -m -s /bin/bash testuser1
```

2 在终端界面，切换到/etc/pam.d 目录下。

3 在配置文件 login 和 sshd 中在找到“**account requisite pam_time.so**”，

确定该行前没有注释符号“#”。如果没有此行内容，则增加该内容。

4 执行 **sudo vi** */etc/security/time.conf* 命令，在文件中添加如下内容。

```
#限制 testuser1 在 18 点～22 点不能以 login 方式登录系统。
login;*;testuser1;!Al1800-2200
#限制 testuser1 在 18 点～22 点不能以 SSH 方式登录系统。
sshd;*;testuser1;!Al1800-2200
```

*说明：执行 **date** 命令，确认系统时间在 18:00-22:00 时间段。若不在该时间段，可以执行 **date** 命令，修改为该时间段。*

5 在另外一台设备上，在 18:00-22:00 时间段内，通过 ssh 连接。提示远程服务连接关闭。

pam_pwquality.so

■ 概述：该模块由 libpwquality 提供，用于检测密码是否符合设定的密码策略。

■ 配置文件：如需在验证密码时使用该模块，需在“**/etc/pam.d/system-auth**”和“**/etc/pam.d/password-auth**”的配置文件中，配置如下内容。

```
password requisite pam_pwquality.so try_first_pass local_users_only retry=1 enforce_for_root
```

说明：retry 表示设置密码时，第二次重复输入密码错误，可以重复尝试的次数。enforce_for_root 表示 root 用户设置密码也需符合密码策略的配置。

该模块使用的密码策略的配置文件为“**/etc/security/pwquality.conf**”，该模块的常用配置如表 5.2 所示。

表 5.2 参数描述

参数	描述
minlen = 8	密码长度不小于 8 位。
minclass = 3	至少需要数字、大写字母、小写字母和其他字符这四类字符中的三类。
dictcheck = 1	开启字典检查。
usercheck = 1	检查到密码中包含用户名，则不能设置。
dictpath =	字典检查所使用字典的配置，默认使用“**/usr/share/cracklib/pw_dict**”。
enforcing = 1	密码必须通过所有检测才能设置成功。

*说明：虽然没有参数控制回文的检测，但是 pam_pwquality.so 会进行密码是否为回文的检测。可以执行 **man pam_pwquality** 命令，查询所有可用的配置参数。*

■ 功能测试

1 执行如下命令，创建测试用户 test。

sudo useradd *test*

2 在终端界面，切换到目录/etc/pam.d 下。

3 在配置文件 system-auth 和 password-auth 中在找到“**password requisite pam_pwquality.so**”，确认该行前没有注释符号“#”。如果没有此行内容，则加入该内容。

◆ 需执行的命令如下：

sudo vi */etc/pam.d/system-auth*
sudo vi */etc/pam.d/password-auth*

◆ 需增加或修改的内容如下：

password requisite pam_pwquality.so try_first_pass local_users_only retry=1 enforce_for_root

4 执行 **sudo vi** */etc/security/pwquality.conf* 命令，添加如下内容：

#修改密码最小长度为 9
minlen = 9

5 执行 **sudo passwd** *test* 命令，验证 passwd 命令结果。

更改用户 test 的密码 。
新的 密码：
无效的密码： 密码少于 9 个字符
passwd: 鉴定令牌操作错误

说明：“union12#”不符合 minlen=9 的密码策略的要求，不能成功设置密码。

更改用户 test 的密码 。
新的 密码：
重新输入新的 密码：
passwd：所有的身份验证令牌已经成功更新。

说明：“union12#$”符合 minlen=9 的密码策略的要求，可以成功设置密码。

pam_deepin_pw_check.so

■ 概述：该模块由 deepin-pw-check 提供，用于检测密码是否符合设定的密码策略。

■ 配置文件：如需在验证密码时使用该模块，需在“**/etc/pam.d/system-auth**”和“**/etc/pam.d/password-auth**”的配置文件中，配置如下内容。

password requisite pam_deepin_pw_check.so try_first_pass local_users_only retry=1 enforce_for_root

说明：retry 表示设置密码时，第二次重复输入密码错误,可以重复尝试的次数。enforce_for_root 表示 root 用户设置密码也需符合密码策略的配置。

该模块所使用的密码策略的配置文件为“**/etc/deepin/dde.conf**”，该配置可使用的所有配置如表 5.3 所示。

表 5.3 参数描述

参数	描述
STRONG_PASSWORD = true	启用密码检测。
PASSWORD_MIN_LENGTH = 8	密码长度不小于 8 位。
PASSWORD_MAX_LENGTH = 511	密码长度不大于 511 位。
VALIDATE_POLICY = 1234567890;abcdefghijklmnopqrstuvwxyz;ABCDEFGHIJKLMNOPQRSTUVWXYZ;!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~	以“;”区分类型，依次为数字、小写字母、大写字母、符号四种类型。
VALIDATE_REQUIRED = 3	包含 VALIDATE_POLICY 中的 3 种类型。
PALINDROME_NUM = 4	密码中不得包含大于等于 4 位的回文，如包含 abcddcba，则不能设置成功。
WORD_CHECK = 1	启用字典检查，密码不得包含字典词汇。
MONOTONE_CHARACTER_NUM = 0	不检查连续字符的个数。
CONSECUTIVE_SAME_CHARACTER_NUM = 0	不检查相同字符的个数。
DICT_PATH =	字典检查所使用字典的配置，默认使用“**/usr/share/cracklib/pw_dict**”。
FIRST_LETTER_UPPERCASE = false	不检测首字母大写。

■ 功能测试

1 执行如下命令，创建测试用户 test。

sudo useradd *test*

2 在终端界面，切换到目录/etc/pam.d 下。

3 在配置文件 system-auth 和 password-auth 中在找到“**password requisite pam_deepin_pw_check.so**”，确认该行前没有注释符号“#”。如果没有此行内容，则加入该内容。

◆ 需要执行的命令如下：

sudo vi */etc/pam.d/system-auth*

sudo vi */etc/pam.d/password-auth*

◆ 需要增加或修改的内容如下：

```
password requisite pam_deepin_pw_check.so try_first_pass
local_users_only retry=1 enforce_for_root
```

4 执行 **sudo vi** */etc/deepin/dde.conf* 命令，在文件中修改或添加如下内容：

```
#修改密码最小长度为 9
PASSWORD_MIN_LENGTH = 9
```

5 执行 **sudo passwd** *test* 命令，验证 passwd 命令结果。

```
更改用户 test 的密码 。
新的 密码：
重新输入新的 密码：
无效的密码： 密码长度不能少于 9 个字符
passwd: 鉴定令牌操作错误
```

说明：“union12#”不符合 PASSWORD_MIN_LENGTH = 9 的密码策略的要求，不能成功设置密码。

```
更改用户 test 的密码 。
新的 密码：
重新输入新的 密码：
passwd：所有的身份验证令牌已经成功更新。
```

说明：“union12#$”符合 PASSWORD_MIN_LENGTH = 9 的密码策略的要求，可以成功设置密码。

5.2 iptables

5.2.1 概述

iptables 是 Linux 管理员用来设置 IPv4/IPv6 数据包过滤条件和 NAT 的命令行工具，是 Linux 防火墙系统的重要组成部分。iptables 工具运行在用户态，主要是设置各种规则。

5.2.2 功能测试

1 加载以下两个模块。

```
modprobe ip_tables
modprobe iptable_filter
```

2 执行如下命令，禁止从测试机（192.168.100.20）ping 服务器。

iptables -A INPUT -p icmp --icmp-type *8* -s *192.168.100.20/24* -j *DROP*

3 执行如下命令，禁止外网对某个端口（22）的访问。

iptables -A INPUT -i *eth0* -p *tcp* --dport *22* -j *DROP*

5.2.3 安全规则

统信服务器操作系统默认集成了网络防火墙软件 iptables。

1 以 root 用户身份登录系统。

2 在终端界面，执行 **iptables-save** 命令，显示如下系统默认的 iptables 规则。

```
[root@localhost pam.d]# iptables-save
# Generated by xtables-save v1.8.1 on Sat May 23 03:16:15 2020
*filter
:INPUT ACCEPT [23498:16150128]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [15403:3768018]
COMMIT
# Completed on Sat May 23 03:16:15 2020
# Generated by xtables-save v1.8.2 on Sat May 23 03:16:15 2020
*security
:INPUT ACCEPT [23498:16150128]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [15403:3768018]
COMMIT
# Completed on Sat May 23 03:16:15 2020
# Generated by xtables-save v1.8.2 on Sat May 23 03:16:15 2020
*raw
:PREROUTING ACCEPT [23498:16150128]
:OUTPUT ACCEPT [15403:3768018]
COMMIT
# Completed on Sat May 23 03:16:15 2020
# Generated by xtables-save v1.8.2 on Sat May 23 03:16:15 2020
*mangle
:PREROUTING ACCEPT [23498:16150128]
:INPUT ACCEPT [23498:16150128]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [15403:3768018]
:POSTROUTING ACCEPT [15403:3768018]
COMMIT
# Completed on Sat May 23 03:16:15 2020
```

```
# Generated by xtables-save v1.8.2 on Sat May 23 03:16:15 2020
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
# Completed on Sat May 23 03:16:15 2020
```

5.2.4 相关命令

- 以 root 用户身份登录系统，在终端界面，执行 **iptables-restore** 命令，从一个已存在的文件中恢复 iptables。
- 执行 **iptables-apply** 命令，更新 iptables。
- 执行 **iptables-save** 命令，显示系统的 iptables 规则并输出到屏幕显示。

5.3 Nmap

5.3.1 概述

NMap（Network Mapper）是 Linux 下的网络扫描和嗅探工具包。它的设计目标是快速地扫描大型网络，以新颖的方式使用原始 IP 报文来发现网络上有哪些主机，以及那些主机提供什么服务（包括应用程序名和版本信息）、那些服务运行在什么操作系统（包括版本信息）、它们使用什么类型的报文过滤器/防火墙。它还可以作为管理员处理日常工作的工具，例如，查看整个网络的信息、管理服务升级计划，以及监视主机和服务的运行。NMap 核心功能如下：

- 主机发现（Host Discovery）:用于发现目标主机是否处于活动状态（Active）。
- 端口扫描（Port Scanning）：用于扫描主机上的端口状态，主要状态有开放（Open）、关闭（Closed）、过滤（Filtered）、未过滤（Unfiltered）、开放|过滤（Open|Filtered）、关闭|过滤（Closed|Filtered）。
- 版本侦测（Version Detection）：用于识别端口上运行的应用程序和版本。
- 操作系统侦测（OS detection）：用于识别目标机的操作系统类型、版本编号及设备类型。
- 防火墙/IDS 规避（Firewall/IDS evasion）：nmap 提供多种机制来规避防火墙、IDS 的的屏蔽和检查，便于秘密地探查目标机的状况。规避方式包括分片（Fragment）、IP 诱骗（IP decoys）、IP 伪装（IP spoofing）、MAC 地址伪装（MAC spoofing）等。
- NSE 脚本引擎（Nmap Scripting Engine）：用于增强主机发现、端口扫描、

版本侦测、操作系统侦测等功能，还可以用来扩展高级的功能，如 web 扫描、漏洞发现、漏洞利用等。

5.3.2 安装

以 root 用户执行 **dnf install** *nmap* 命令，安装 nmap。

5.3.3 功能测试

1 以 root 用户身份登录系统，在终端界面，执行如下命令，扫描本机所有保留的 TCP 端口。

nmap -v *localhost.localdomain*

2 执行如下命令，进行秘密 SYN 扫描，同时尝试确定每台工作主机的操作系统类型。

nmap -sS -O *scanme.nmap.org/24*

3 执行如下命令，随机选择 100 台主机扫描是否运行 Web 服务器（端口：80），由起始阶段发送探测报文来确定主机是否工作。

nmap -v -iR *100* -P0 -p *80*

4 执行如下命令，扫描 4096 个 IP 地址并查找 Web 服务器，将扫描结果以 Grep 和 XML 格式保存。

nmap -P0 -p80 -oX */tmp/pb-port80scan.xml* -oG */tmp/pb-port80scan.gnmap 192.168.100.20/20*

5.4 Auditd

5.4.1 概述

Auditd 是一款 Linux 服务器下安全审计系统工具。审计系统是统信服务器操作系统安全体系的重要组成部分，安全审计能够对文件、目录、系统资源、系统调用进行监控和记录。系统管理员通过创建完善的监控和审计规则，使违反规则的行为被审计和记录，以便进行安全追踪和定位。

5.4.2 启动

Audit 服务启动方式：执行 **systemctl start** *auditd.service* 命令。

5.4.3 配置文件

审计守护进程：在统信服务器操作系统中，守护进程负责把审计信息写入磁盘审计日志。该守护进程具有两个配置文件，分别为环境配置和规则配置。

- 审计环境配置文件：在统信服务器操作系统中，环境配置文件的路径为“**/etc/audit/auditd.conf**”。
- 审计规则配置文件：在统信服务器操作系统中，规则配置文件的路径为

“**/etc/audit/audit.rules**”，用户可根据自己的需求在配置文件中制定相应规则。在统信服务器操作系统中，审计规则设置程序 auditctl 主要用于控制审计系统产生何种审计信息。例如：需要添加对/etc/crontab 进行监控，并且设置 key 为 CFG_crontab，方便对日志进行检索，具体命令如下所示。

- 设置命令：**auditctl** -w */etc/crontab* -p wa -k *CFG_crontab*。
- 移除命令：**auditctl** -W */etc/crontab* -p wa -k *CFG_crontab*。

5.5 sysctl

5.5.1 概述

“**/proc/sys**”目录下存放着大多数内核参数，并且可以在系统运行时进行更改，但是重启机器就会失效。“**/etc/sysctl.conf**”是一个允许改变正在运行中的 Linux 系统的接口，它包含一些 TCP/IP 堆栈和虚拟内存系统的高级选项，修改内核参数永久生效。也就是说，“**/proc/sys**”下内核文件与配置文件“**sysctl.conf**”中变量存在着对应关系。

5.5.2 功能测试

1 以 root 用户身份登录系统，在终端界面，执行 **sysctl** -a 命令，显示所有当前有效的内核参数值。

2 执行“**sysctl** -w *参数*”命令，修改内核参数值，具体的参数名及其作用如下：

- kernel.randomize_va_space：用来设置内存地址随机化的行为。
- net.ipv4.tcp_syncookies：用来防止 SyncFlood 攻击。
- net.ipv4.tcp_fin_timeout：表示当服务器主动关闭连接时，socket 保持在 FIN-WAIT-2 状态的最大时间。
- net.ipv4.tcp_max_syn_backlog：表示 TCP 三次握手建立阶段接受 SYN 请求队列的最大长度。
- net.core.somaxconn：定义系统中每一个端口最大的监听队列的长度。
- kernel.kptr_restrict：是否限制从/proc/kallsym 和其他接口显示（打印）内核符号表的地址。
- vm.mmap_min_addr：指定用户进程通过 mmap 可使用的最小虚拟内存地址，以避免其在低地址空间产生映射导致安全问题。

说明：对不同的参数值，执行 ***cat*** */proc/kallsyms 命令，可对内核符号地址的显示进行测试。*

6 服务器运维工具

6.1 sysstat

6.1.1 概述

sysstat 是一个监测系统性能及效率的工具，比如 CPU 使用率、硬盘和网络吞吐数据。收集和分析这些系统性能数据，有利于判断系统是否正常运行，是提高系统运行效率、安全运行服务器的得力助手。sysstat 软件包集成的工具如下：

- iostat 工具：提供 CPU 使用率及硬盘吞吐效率的数据。
- mpstat 工具：提供处理器相关数据。
- sar 工具：负责收集、报告并存储系统活跃的信息。
- sa1 工具：负责收集并存储每天系统动态信息到一个二进制的文件中。它是通过计划任务工具 cron 来运行。
- sa2 工具：负责把每天的系统活跃性息写入总结性的报告中。它是为 sar 所设计的前端，要通过 cron 来调用。
- sadc 工具：负责收集系统动态数据，将收集的数据写入一个二进制的文件中，它被用作 sar 工具的后端。
- sadf 工具：显示被 sar 通过多种格式收集的数据。

6.1.2 安装 sysstat 工具

yum install *sysstat* -y

6.1.3 配置文件

- 修改的配置文件为“**/etc/sysconfig/sysstat**”。
- sa 和 sar 收集的数据默认保存路径为“**/var/log/sa**”，需修改时，可以参见“**/etc/sysconfig/sysstat**”配置文件的 SA_DIR 配置项。

6.1.4 命令介绍

sadc 工具

sadc 工具存放在“**/usr/lib64/sa/**”目录中。该工具把数据写在一个二进制的文件中，如需要查看数据内容，则使用 sadf 工具来显示。在终端界面，执行 **man** 8 *sadc* 命令，可查看其他更多参数及其含义。

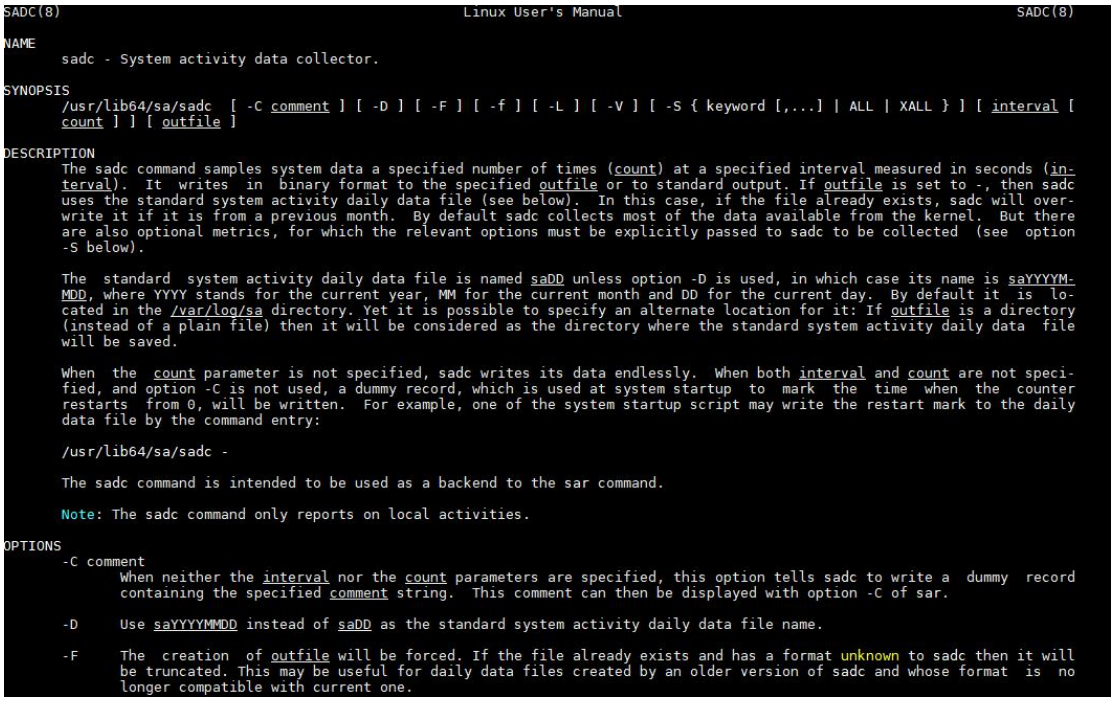


图 6.1 man 8 sadc 显示示例

sar 工具

sar 工具主要收集系统 CPU、硬盘动态数据，还可以收集动态显示。在终端界面，执行 **man** 1 *sar* 命令，可查看其他更多参数及其含义。

图 6.2 man 1 sar 显示示例

iostat 工具

iostat 工具用于显示系统 CPU 使用状态、硬盘设备的吞吐率。在终端界面，执行 **man** *iostat* 命令，可查看其他更多参数及其含义。

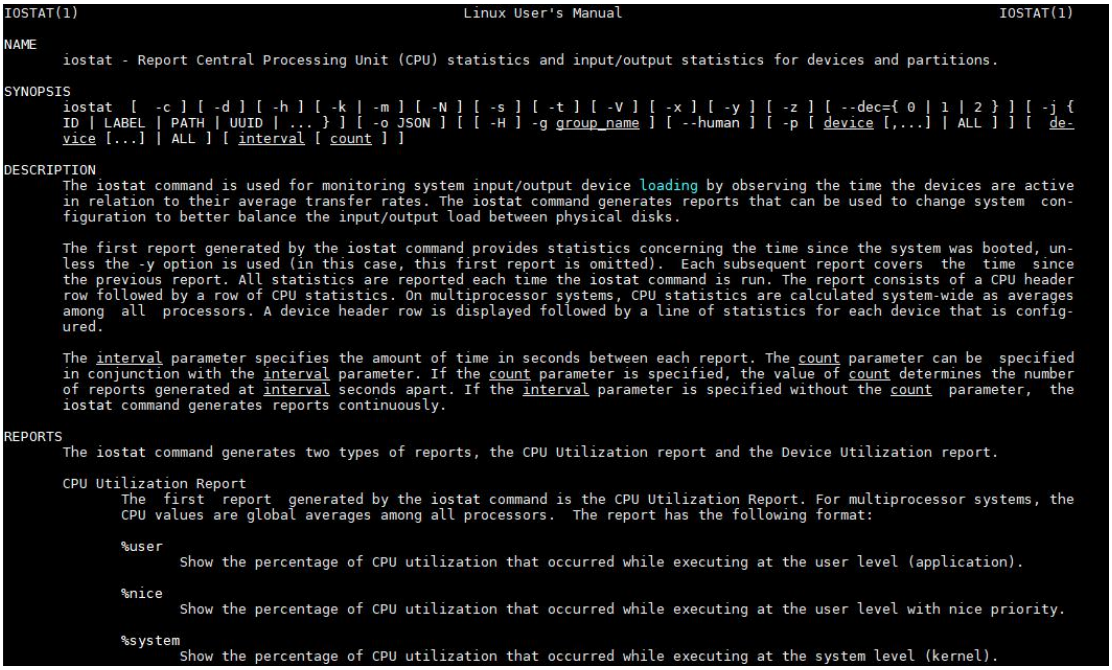


图 6.3 man iostat 显示示例

mpstat 工具

mpstat 工具主要是用于多处理器系统中的 CPU 的利用率的统计。mpstat 也可以加参数，用-P 来指定哪个 CPU，且处理器的 ID 是从 0 开始的。PROC 文件系统必须先挂载，默认情况下，系统已经挂在了 proc 文件系统。在终端界面，执行 **man** 1 *mpstat* 命令，可查看其他更多参数及其含义。

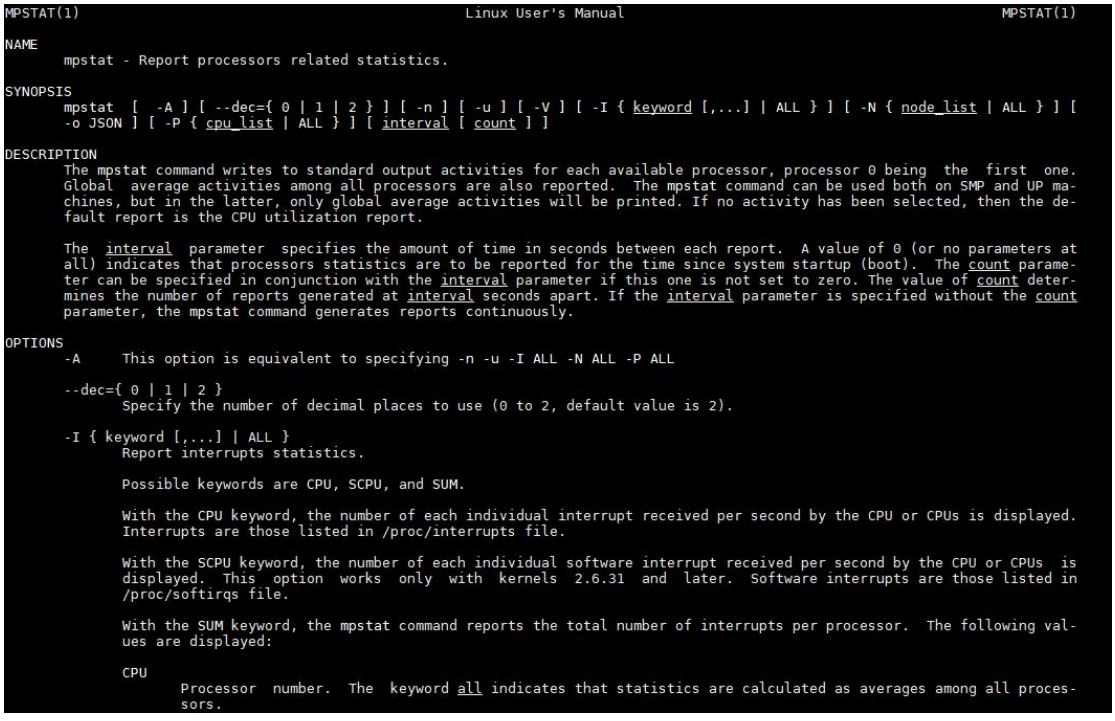


图 6.4 man 1 mpstat 显示示例

sadf 工具

sdaf 工具主要是从二进制文件中提取 sar 所收集的数据，并以多种格式进行显示。在终端界面，执行 **man** 1 *sadf* 命令，可查看其他更多参数及其含义。

图 6.5 man 1 sadf 显示示例

6.1.5 功能测试

1 使用 sadc 收集系统动态数据，让它在每秒收集一次，共收集 5 次动态信息，并写到一个指定的文件。再通过 sar 工具来查看系统的状态。

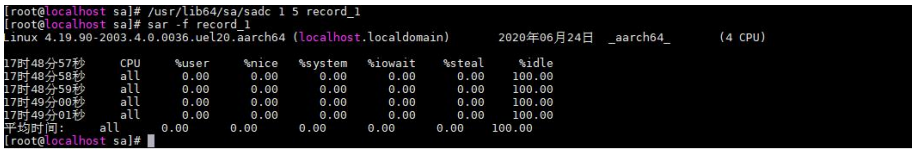


图 6.6 sadc 收集系统数据，并通过 sar 查看系统状态

2 CPU 利用率动态更新，每秒获取一次数据，共获取 5 次。

图 6.7 sar 获取 CPU 利用率

3 查看网络设备的吞吐情况，让数据每 1 秒获取一次，共获取 5 次。

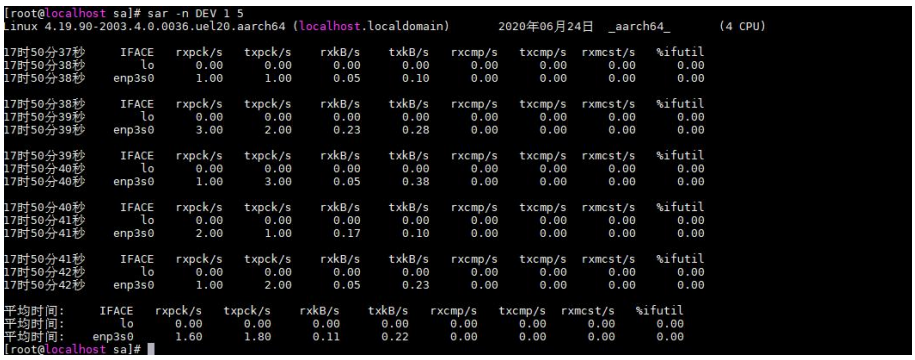


图 6.8 sar 获取网络设备吞吐情况

4 显示 CPU 使用率。

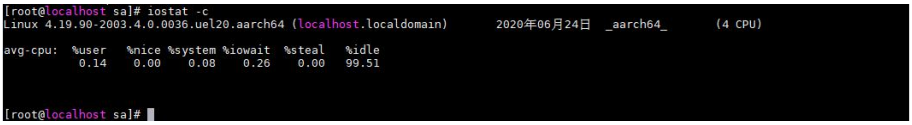


图 6.9 iostat 显示 CPU 利用率

5 统计多处理器系统中的 CPU0 利用率。

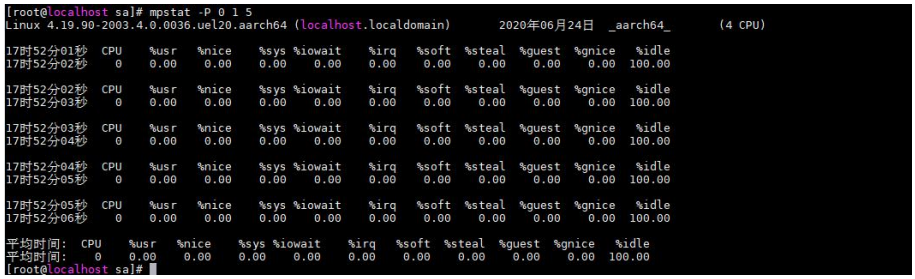


图 6.10 mpstat 统计 CPU0 的利用率

6 从当前的日常数据文件中提取 cpu1 的统计数据。

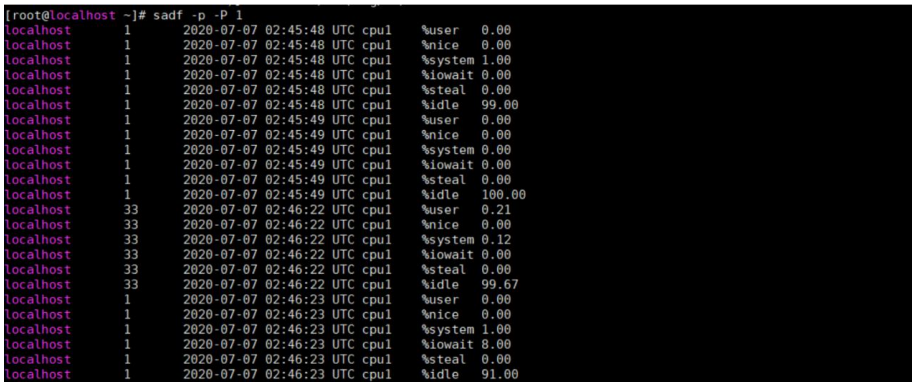


图 6.11 sadf 提取 cpu1 的统计数据

6.2 top

6.2.1 概述

top 是 Linux 系统中的一个互动的进程查看器，是一个文本模式的应用程序。它可让用户进行交互式操作，可横向或纵向滚动浏览进程列表，可通过方向键控

制视图移动。

6.2.2 命令介绍

在终端界面，通过 root 用户执行 **dnf install** *procps-ng procps-ng-help* 命令安装 procps-ng 和 procps-ng-help 软件包。完成后，执行 **man** 1 *top* 命令，可查看其他更多参数及其含义。

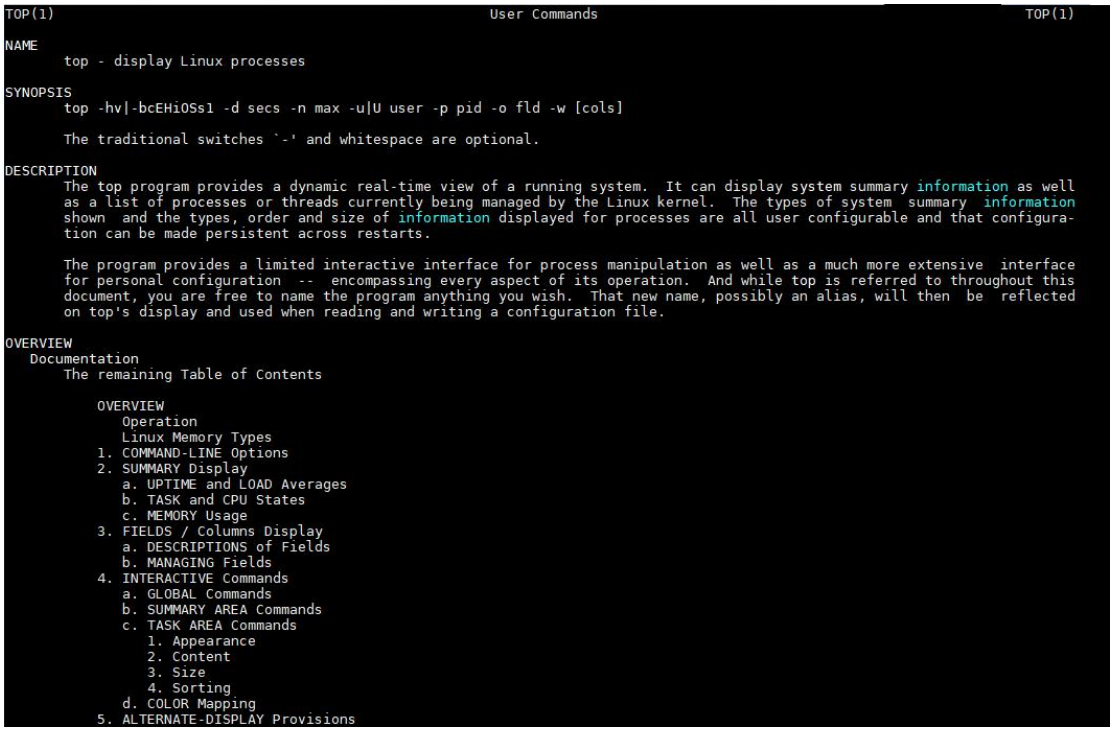


图 6.12 man 1 top 显示示例

交互式命令如下：

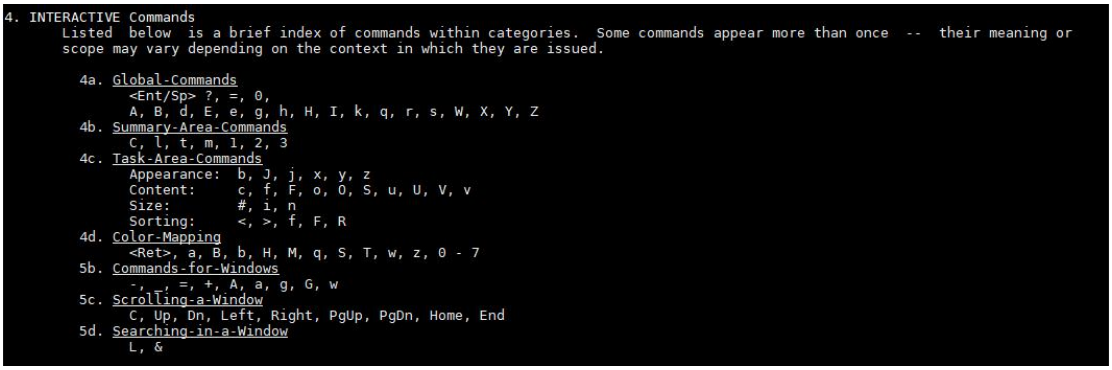


图 6.13 top 交互式命令说明

6.2.3 功能测试

以 root 用户身份登录系统，在终端界面，执行 **top** 命令，显示进程的实时状态。

- 按下 z 键，输出上色；再次按下 z 键，退出上色。
- 按下 N 键，以 PID 的大小的顺序排列表示进程列表。

- 按下 P 键，以 CPU 占用率大小的顺序排列进程列表。
- 按下 m 键，以内存占用率大小的顺序排列进程列表。
- 按下 f 键，选择默认显示的列。
- 示例如下：
  - 显示内存情况。

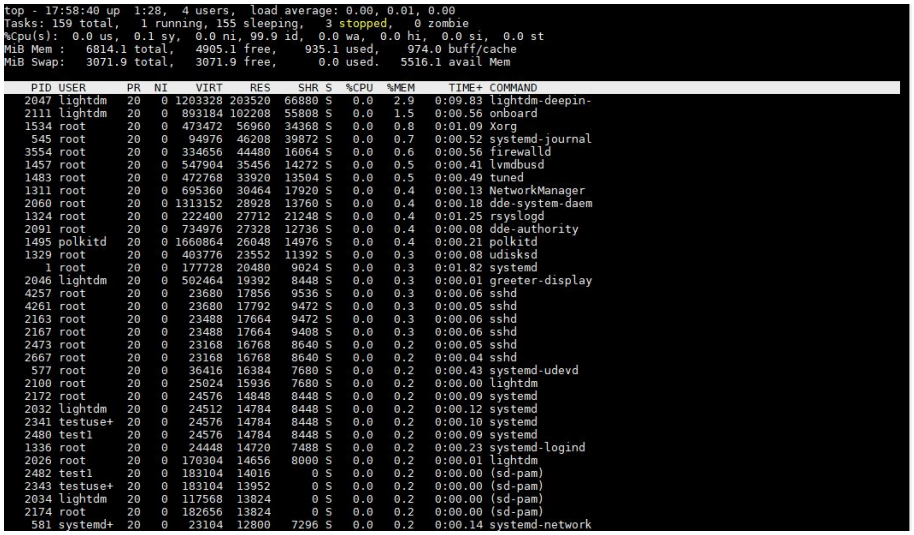


图 6.14 top 显示内存情况

  - 显示上色。

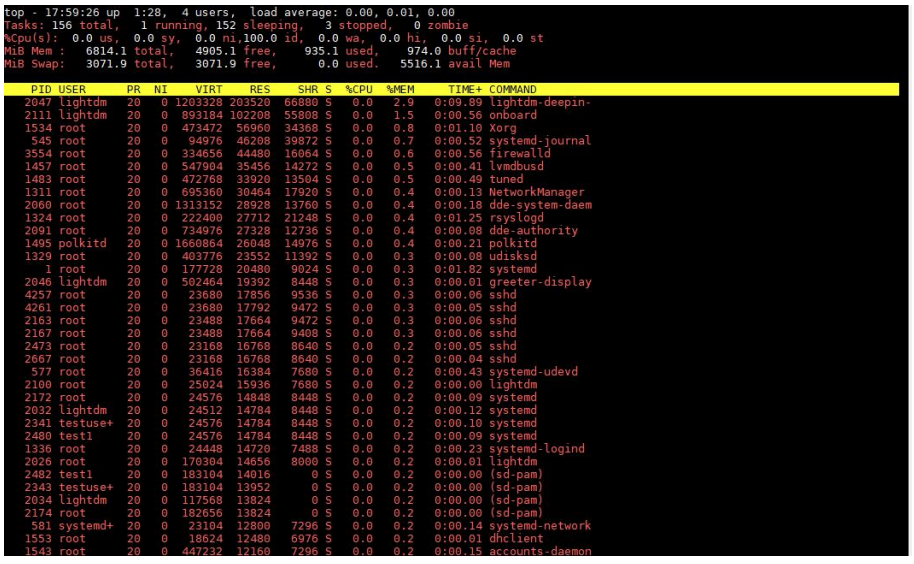


图 6.15 top 显示上色

  - 按照 cpu 使用大小排序。

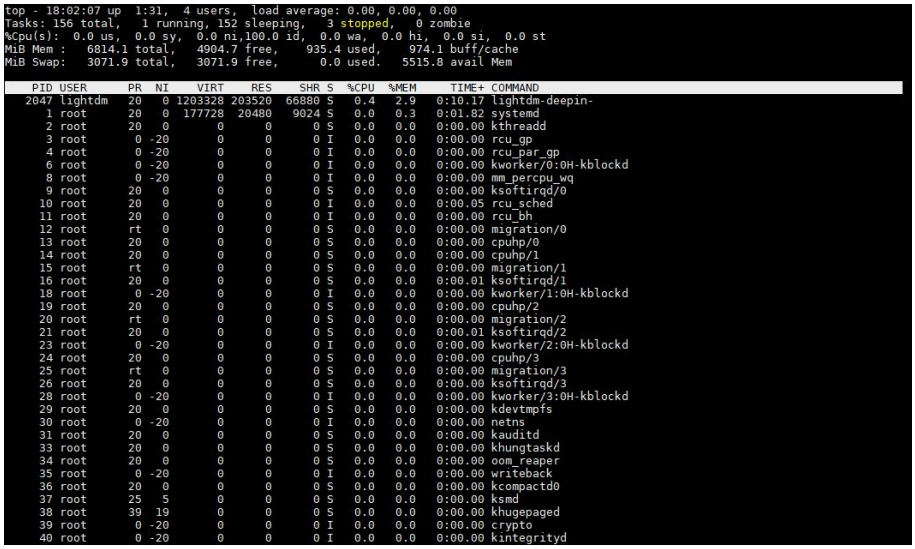


图 6.16 top 显示按照 CPU 使用排序

6.3 iotop

在统信服务器操作系统中，使用类似 top 的 i/o 监视器。

6.3.1 概述

iotop 命令是一个用来监视磁盘 I/O 使用状况的 top 类工具。iotop 具有与 top 相似的用户界面接口，其中包括 PID、用户、I/O、进程等相关信息。Linux 下的 IO 统计工具如 iostat、nmon 等大多数是只能统计到每个设备的读写情况，如果需要知道每个进程是如何使用 IO 的，可以执行 **iotop** 命令进行查看。

6.3.2 命令介绍

在终端界面，通过 root 用户执行 **dnf install** *iotop iotop-help* 命令安装 iotop 和 iotop-help 软件包。完成后，执行 **man** 8 *iotop* 命令，可查看其他更多参数及其含义。

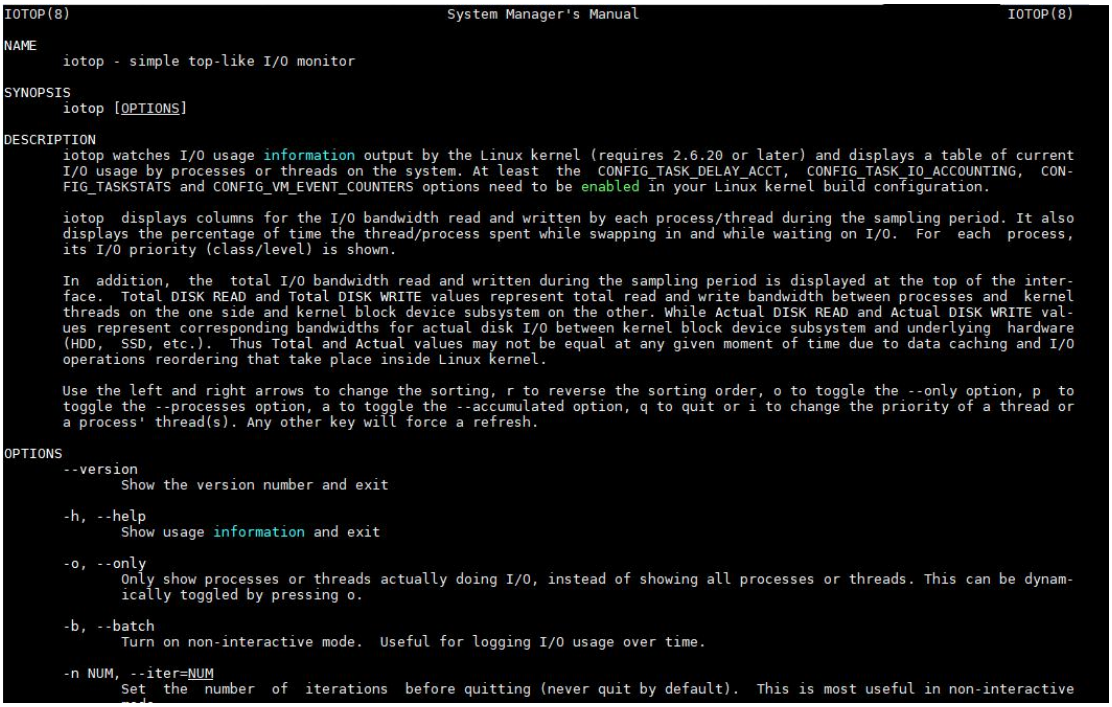


图 6.17 man 8 iotop 显示示例

6.3.3 功能测试

1 以 root 用户身份登录系统，在终端 1 界面，执行 **iotop** 命令，显示系统中操作磁盘 I/O 的进程的实时状态。

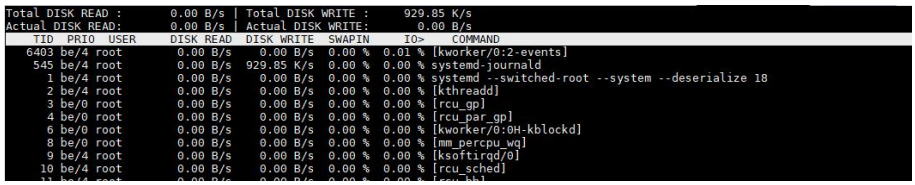


图 6.18 iotop 显示磁盘 I/O 实时状态

2 切换到终端 2 界面，执行如下命令，对磁盘进行写操作，终端 1 中 iotop 状态实时显示监控磁盘的写速度。

cd */home*
dd if=*/dev/zero* of=*bigfile* bs=*1M* count=*5000*

3 切换到终端 2 界面，执行如下命令，对磁盘进行读操作，终端 1 中 iotop 状态实时显示监控磁盘的读速度。

cd */home*
dd if=*bigfile* of=*/dev/null* bs=*1M*

6.4 lsof

6.4.1 概述

lsof（list open files）是一个列出当前系统已打开文件的工具。通过文件不

仅可以访问常规数据，还可以访问网络连接和硬件，如传输控制协议（TCP）和用户数据报协议（UDP）套接字等。系统在后台都为该应用程序分配了一个文件描述符，该文件描述符为应用程序与基础操作系统之间的交互提供了通用接口。

lsof 打开的文件如下：普通文件、目录、网络文件系统的文件、字符或设备文件、(函数)共享库、管道/命名管道、符号链接、网络文件等。

6.4.2 命令介绍

执行 **lsof | less** 命令，各列信息的意义如下。

- COMMAND：进程的名称。
- PID：进程标识符。
- USER：进程所有者。
- FD：文件描述符，应用程序通过文件描述符识别该文件。
- TYPE：文件类型。
- DEVICE：指定磁盘的名称。
- SIZE：文件的大小。
- NODE：索引节点（文件在磁盘上的标识）。
- NAME：打开文件的确切名称。

6.4.3 功能测试

- 执行 **lsof** -u *username* 命令，列出某个用户打开的文件信息。

  说明：-u 选项中的 u 其实是 user 的缩写。

- 执行 **lsof** -c *apache* 命令，列出某个程序进程所打开的文件信息。
- 执行 **lsof** -c *mysql* -c *apache* 命令，列出多个进程打开的多个文件信息。
- 执行 **lsof** -u *test* -c *mysql* 命令 ，列出某个用户以及某个进程所打开的文件信息。
- 执行 **lsof** -u *^root* 命令，列出除了某个用户外的被打开的文件信息。

  说明：^这个符号在用户名之前，表示不显示该用户打开的进程。

- 执行 **lsof** -p *1* 命令，列出某个进程号所打开的文件信息。
- 执行 **lsof** -p *1,2,3* 命令，列出多个进程号对应的文件信息。
- 执行 **lsof** -p *^1* 命令，列出除了某个进程号，其他进程号所打开的文件信息。
- 执行 **lsof** -i 命令，列出所有的网络连接。
- 执行 **lsof** -i *tcp* 命令，列出所有 tcp 网络连接信息。

6.5 psmisc

6.5.1 概述

psmisc 是一个使用 proc 文件系统的软件包，而不是一个单独的应用程序。

安装此软件包后，提供的程序如下表所示。

表 6.1 psmisc 提供程序说明

程序	说明
pstree	显示一个进程状态树。
pstree.x11	显示一个进程树，它是连接到 pstree 的一个符号。
fuser	使用文件或套接字来识别进程。
prtstat	打印一个进程的统计信息。
killall	通过进程名来 kill 进程，给所有运行的进程发一个信号。
pslog	报告进程的当前日志路径。

6.5.2 命令介绍

在终端界面，执行 **man** *pstree/prtstat/...*命令，可查看其他更多参数及其含义。

6.5.3 功能测试

执行 **pstree** -n -p 命令，对同父进程按 pid 大小排序后，显示进程树。

```
[root@localhost man]# pstree -n -p
systemd(1)─┬─systemd-journal(545)
           ├─systemd-udevd(577)
           ├─systemd-network(581)
           ├─rpcbind(1282)
           ├─mdadm(1283)
           ├─auditd(1290)───{auditd}(1291)
           ├─dbus-daemon(1310)───{dbus-daemon}(1408)
           ├─NetworkManager(1311)─┬─{NetworkManager}(1395)
           │                      ├─{NetworkManager}(1426)
           │                      └─dhclient(1553)
           ├─lsmd(1318)
           ├─rasdaemon(1320)
           ├─rngd(1323)
           ├─rsyslogd(1324)─┬─{rsyslogd}(1394)
           │                └─{rsyslogd}(1397)
           ├─rtkit-daemon(1325)─┬─{rtkit-daemon}(1428)
           │                    └─{rtkit-daemon}(1429)
           ├─smartd(1326)
           ├─udisksd(1329)─┬─{udisksd}(1364)
           │               ├─{udisksd}(1431)
           │               ├─{udisksd}(1549)
           │               └─{udisksd}(1561)
           ├─systemd-logind(1336)
           ├─chronyd(1338)
           ├─irqbalance(1339)───{irqbalance}(1340)
           ├─restorecond(1343)
           ├─lvmdbusd(1457)─┬─{lvmdbusd}(1564)
           │                ├─{lvmdbusd}(1568)
           │                └─{lvmdbusd}(1569)
           ├─tuned(1483)─┬─{tuned}(1899)
           │             ├─{tuned}(1947)
           │             └─{tuned}(1990)
           ├─gssproxy(1491)─┬─{gssproxy}(1500)
           │                ├─{gssproxy}(1501)
           │                ├─{gssproxy}(1502)
           │                ├─{gssproxy}(1503)
           │                └─{gssproxy}(1504)
           ├─polkitd(1495)─┬─{polkitd}(1497)
           │               ├─{polkitd}(1498)
           │               ├─{polkitd}(1506)
           │               ├─{polkitd}(1507)
           │               ├─{polkitd}(1508)
           │               ├─{polkitd}(1509)
```

图 6.19 pstree 显示进程树

6.6 procps

6.6.1 概述

procps-ng 是一个使用 proc 文件系统的软件包，而不是一个单独的应用程序。安装此软件包后，提供的程序如下表所示。

表 6.2 procps-ng 软件包提供软件说明

程序	说明
slabtop	实时显示内核 slab cache 信息。
skill	发送一个信号给进程。
vmstat	报告关于进程、内存、页、块 io、陷阱、磁盘和 cpu 的活动信息。
watch	周期性的执行某个程序，并全屏显示其运行。
free	显示系统中可用内存和已使用内存。
pwdx	报告一个进程的当前工作目录。
snice	发送一个信号并报告进程状态。
sysctl	运行时修改内核参数。
pgrep	按照匹配的选项标准查找并列出当前运行的进程。
top	动态实时的显示系统中进程的状态。
uptime	显示一行信息，包括系统当前时间、系统已经运行了多长时间、当前系统登录了多少个用户等。
w	显示谁登录了系统，以及正在做什么。
tload	在指定终端图形化显示系统的平均负载。
pmap	报告一个进程的内存映射。
ps	系统中当前活动进程的快照。
pkill	按照匹配的选项标准查找并列出当前运行的进程。

6.6.2 命令介绍

在终端界面，执行 **dnf install** *procps-ng-help* 命令，安装 procps-ng-help 包。执行 **man** *slabtop/pgrep/...*命令，可查看其他更多参数及其含义。

6.6.3 功能测试

执行 **pgrep** *1* 命令，显示 1 号某个进程的内存映射。

```
[root@localhost man]# pgrep 1
14
15
16
18
342
433
439
513
6367
6443
6704
6940
7405
[root@localhost man]#
```

图 6.20 pgrep 搜索进程示例

6.7 e2fsprogs

6.7.1 概述

e2fsprogs 是一个使用 proc 文件系统的软件包，而不是一个单独的应用程序。安装此软件包后，提供的程序如下表所示。

表 6.3 e2fsprogs 软件包提供软件说明

程序	说明
e2undo	将重放 ext2/3/4 文件系统的 undo log。
dumpe2fs	把 super block 上的信息 dump 到 stdin。
resize2fs	调整 ext2/3/4 文件系统尺寸。
e2fsck	ext2/ext3/ext4 文件系统检查程序。
mke2fs	创建 ext2/ext3/ext4 文件系统。
e4defrag	ext4 文件系统碎片整理程序。
filefrag	报告某指定文件碎片情况。
lsattr	对 ext2/3/4 文件系统特有的属性进行查阅。
fsck.ext4	ext4 文件系统检查一般性的命令。
fsck.ext3	检查 Linux 的 ext3 文件系统。
fsck.ext2	ext2 文件系统检查一般性的命令。
e2image	将 ext2/3/4 系统文件还原到分区中。
badblocks	对指定的设备检查坏掉的块。
debugfs	ext2 文件系统交互式除错工具。
tune2fs	调整和查看 ext2/ext3 文件系统的文件系统参数程序。
logsave	保存一个命令的输出到一个日志文件。
e2freefrag	在 ext2/ext3/ext4 文件系统报告碎片剩余空间选项。
chattr	对 ext2/3/4 文件系统特有的属性进行更变。
mkfs.ext4	创建一个 ext4 文件系统。
e2label	设定 LABEL。
mkfs.ext3	创建一个 ext3 文件系统。
mkfs.ext2	创建一个 ext2 文件系统。

6.7.2 命令介绍

在终端界面，执行 **dnf install** *e2fsprogs e2fsprogs-help* 命令，安装 e2fsprogs 和 e2fsprogs-help 包。执行 **man** *e2undo/e2image/...*命令，可查看其他更多参数及其含义。

6.7.3 功能测试

执行 **tune2fs** -c *30 /dev/sda1* 命令，设置系统开机后文件系统强制自检的挂载次数。

```
[root@localhost yum.repos.d]# tune2fs -c 30 /dev/sda1
tune2fs 1.44.6 (5-Mar-2019)
Setting maximal mount count to 30
[root@localhost yum.repos.d]#
```

6.8 strace

6.8.1 概述

strace 是一个集诊断、调试、统计于一体的工具。使用该工具可以跟踪进程执行时的系统调用和所接收的信号。当进程需要访问硬件设备时，必须由用户态模式切换至内核态模式，通过系统调用访问硬件设备。

strace 的最简单的用法就是执行一个指定的命令，在指定的命令执行结束后即退出。在命令执行的过程中，strace 会记录和解析命令进程的所有系统调用，以及该进程所接收到的所有的信号值。

6.8.2 命令介绍

在终端界面，执行 **man** 1 *strace* 命令，可查看其他更多参数及其含义

```
STRACE(1)                                   General Commands Manual                                   STRACE(1)

NAME
       strace - trace system calls and signals

SYNOPSIS
       strace [-ACdffhikqrtttTvVxxy] [-I n] [-b execve] [-e expr]... [-a column] [-o file] [-s strsize] [-X format] [-P path]...
              [-p pid]... { -p pid | [-D] [-E var[=val]]... [-u username] command [args] }

       strace -c [-df] [-I n] [-b execve] [-e expr]... [-O overhead] [-S sortby] [-P path]... [-p pid]... { -p pid | [-D]
              [-E var[=val]]... [-u username] command [args] }

DESCRIPTION
       In  the simplest case strace runs the specified command until it exits.  It intercepts and records the system calls which
       are called by a process and the signals which are received by a process.  The name of each system call, its arguments and
       its return value are printed on standard error or to the file specified with the -o option.

       strace  is  a  useful  diagnostic, instructional, and debugging tool.  System administrators, diagnosticians and trouble-
       shooters will find it invaluable for solving problems with programs for which the source is not readily  available  since
       they  do  not  need  to  be recompiled in order to trace them.  Students, hackers and the overly-curious will find that a
       great deal can be learned about a system and its system calls by tracing even ordinary programs.   And  programmers  will
       find that since system calls and signals are events that happen at the user/kernel interface, a close examination of this
       boundary is very useful for bug isolation, sanity checking and attempting to capture race conditions.

       Each line in the trace contains the system call name, followed by its arguments in parentheses and its return value.   An
       example from stracing the command "cat /dev/null" is:

           open("/dev/null", O_RDONLY) = 3

       Errors (typically a return value of -1) have the errno symbol and error string appended.

           open("/foo/bar", O_RDONLY) = -1 ENOENT (No such file or directory)

       Signals  are  printed as signal symbol and decoded siginfo structure.  An excerpt from stracing and interrupting the com-
       mand "sleep 666" is:

           sigsuspend([] <unfinished ...>
           --- SIGINT {si_signo=SIGINT, si_code=SI_USER, si_pid=...} ---
           +++ killed by SIGINT +++

       If a system call is being executed and meanwhile another one is being called from a different thread/process then  strace
       will  try  to preserve the order of those events and mark the ongoing call as being unfinished.  When the call returns it
       will be marked as resumed.
```

图 6.21 man 1 strace 显示示例

6.8.3 功能测试

跟踪 **ls** 命令的系统调用情况。

```
[root@localhost man]# strace /bin/ls
execve("/bin/ls", ["/bin/ls"], 0xffffff9ce3530 /* 31 vars */) = 0
brk(NULL)                               = 0xaaaaecac0000
faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (没有那个文件或目录)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=78271, ...}) = 0
mmap(NULL, 78271, PROT_READ, MAP_PRIVATE, 3, 0) = 0xfffde43a0000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0@m\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=198960, ...}) = 0
mmap(NULL, 271752, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xfffde4350000
mmap(0xfffde4380000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20000) = 0xfffde4380000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libcap.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0P\26\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=67880, ...}) = 0
mmap(NULL, 131392, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xfffde4320000
mmap(0xfffde4330000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xfffde4330000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\240@\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1531736, ...}) = 0
mmap(NULL, 1596696, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xfffde4190000
mmap(0xfffde4300000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x160000) = 0xfffde4300000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libpcre2-8.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\200#\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=526768, ...}) = 0
mmap(NULL, 590280, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xfffde40f0000
mmap(0xfffde4170000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x70000) = 0xfffde4170000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\0\21\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=67800, ...}) = 0
mmap(NULL, 131168, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xfffde40c0000
mmap(0xfffde40d0000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xfffde40d0000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\08t\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=136544, ...}) = 0
mmap(NULL, 213336, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xfffde4080000
mmap(0xfffde40a0000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10000) = 0xfffde40a0000
```

图 6.22 strace 跟踪 ls 命令示例

6.9 ltrace

6.9.1 概述

ltrace 是一个调试程序，可在它退出之前执行指定命令来拦截和记录动态库调用和信号接收过程。它还可拦截并打印系统调用及直接在二进制文件上使用。

6.9.2 命令介绍

ltrace 是一个仅运行指定命令后直到退出的程序。它截获并记录由执行的过程调用的动态库调用以及该过程接收的信号，它还可以拦截并打印程序执行的系统调用，它和 strace 的使用十分相似。

在终端界面，执行 **dnf install** *ltrace ltrace-help* 命令，安装 ltrace 和 ltrace-help 包。执行 **man** *ltrace* 命令，可查看其他更多参数及其含义。

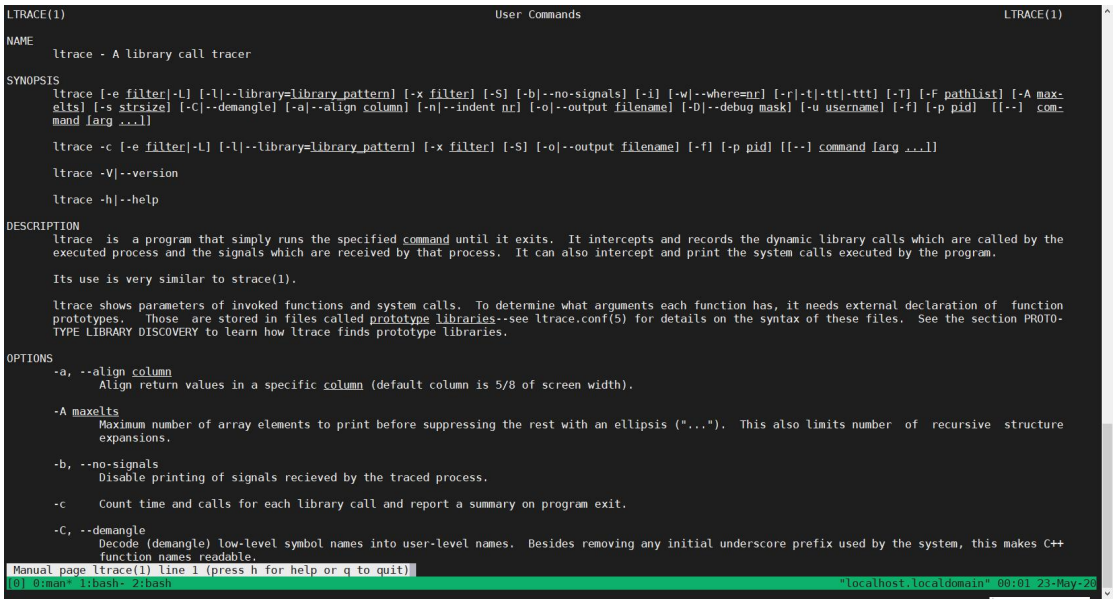


图 6.23 man ltrace 显示示例

6.9.3 功能测试

执行 **ltrace** *ls* 命令，显示跟踪 ls 系统调用的执行情况。

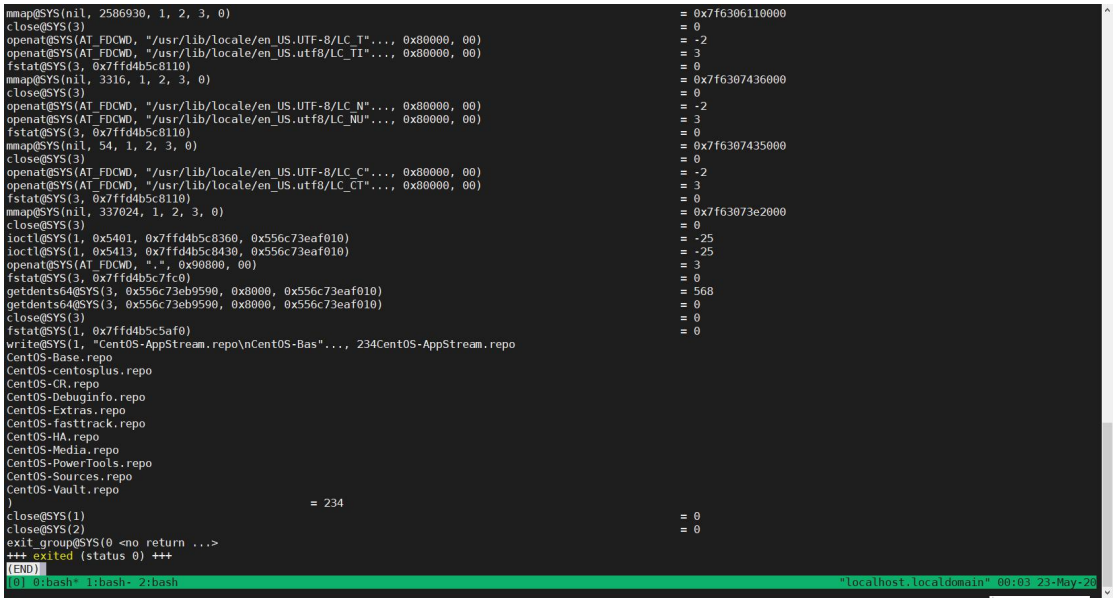


图 6.24 ltrace 跟踪 ls 命令示例

6.10 dmidecode

6.10.1 概述

dmidecode 是在 Linux 下获取有关硬件信息的工具。dmidecode 遵循 SMBIOS/DMI 标准，其输出的信息包括 BIOS、系统、主板、处理器、内存、缓存等等。

■ DMI（Desktop Management Interface，DMI）就是帮助收集电脑系统信息

的管理系统，DMI 信息的收集必须在严格遵照 SMBIOS 规范的前提下进行。

- SMBIOS（System Management BIOS）是主板或系统制造商以标准格式显示产品管理信息所需遵循的统一规范。SMBIOS 和 DMI 是由行业指导机构 Desktop Management Task Force (DMTF)起草的开放性的技术标准，其中 DMI 设计适用于任何的平台和操作系统。

6.10.2 安装过程

在统信服务器操作系统中，默认已经安装 dmidecode。若没安装，请以 root 用户登录系统，执行 **dnf install** *dmidecode* 命令，按照提示输入“**y**”。

6.10.3 功能测试

- 以 root 用户身份登录系统，在终端界面，执行 **dmidecode** -t *bios* 命令，即可显示系统 bios 相关信息。
- 执行 **dmidecode** -t *processor* 命令，即可显示系统 cpu 相关信息。
- 执行 **dmidecode** --type *7* 命令，即可显示系统缓存信息。

6.11 cockpit

6.11.1 概述

Cockpit 是一个免费且开源的基于 web 的管理工具，系统管理员可以执行诸如存储管理、网络配置、检查日志、管理容器等任务。通过 Cockpit 提供的友好的 Web 前端界面，可以轻松地管理我们的 GNU/Linux 服务器，是一个非常轻量级且简单易用的系统管理工具，更重要的是用 Cockpit 可实现集中式管理。

6.11.2 安装 Cockpit

1 执行如下命令，安装 Cockpit 包。

yum install *cockpit* -y

2 执行如下命令，启动服务并设置开机自启动。

systemctl enable *cockpit.socket* --now

3 执行如下命令，关闭防火墙。

systemctl stop *firewalld*

6.11.3 使用 Cockpit

登录

1 打开浏览器，在地址栏中输入 https://IP:9090/后，按“**Enter**”键。

2 输入 root 用户的用户名和密码。

说明：cockpit 管理工具默认监听 9090 端口。

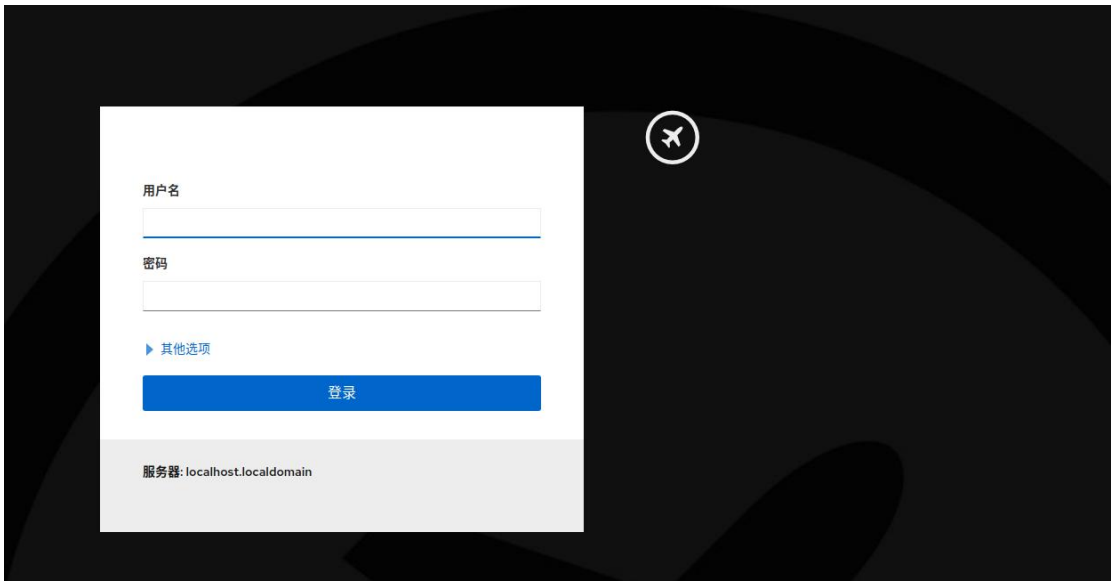


图 6.25 cockpit 登录界面示例

主界面

主界面显示当前主机的系统信息，可以看到 CPU、内存、磁盘，以及网络的使用情况。

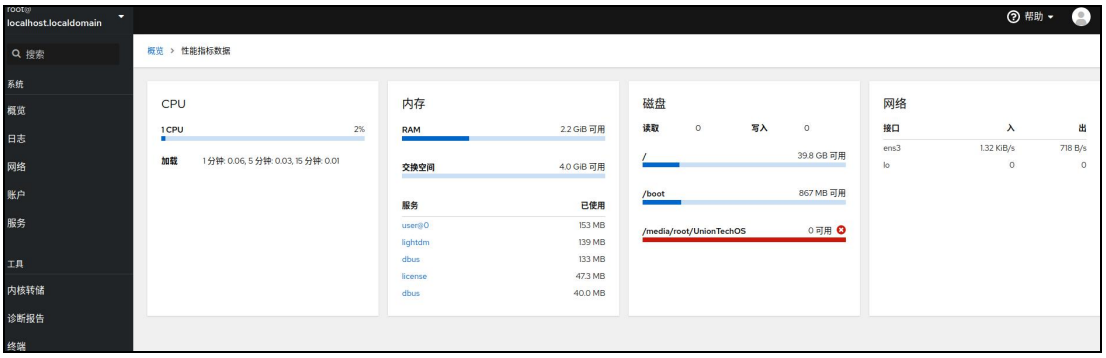


图 6.26 cockpit 主界面显示主机系统信息

添加需要监控的主机

1 单击如下图所示的左侧上方的位置。

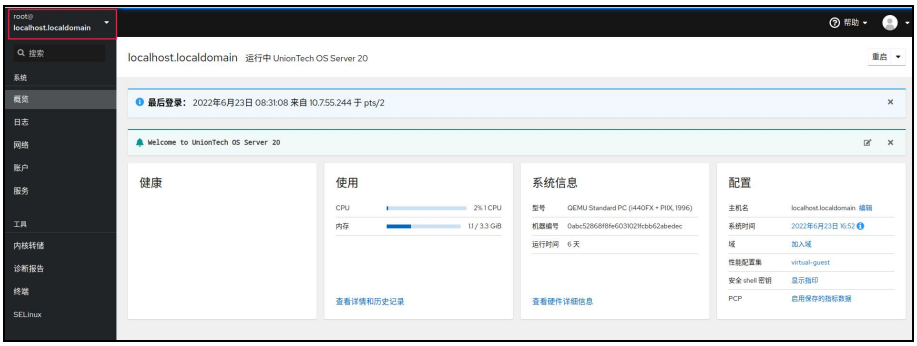


图 6.27 单击左上角主机名按钮

2 单击“**添加新主机**”。

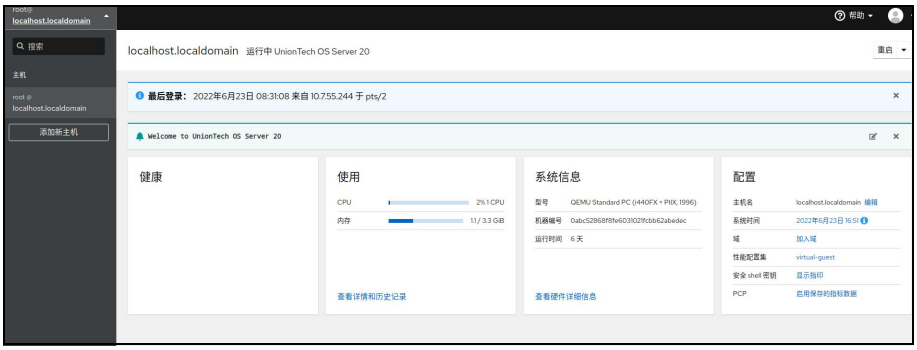


图 6.28 添加被监控主机（1）

3 输入被监控主机的 IP 地址或域名，单击“**添加**”。

图 6.29 添加被监控主机（2）

6.11.4 功能模块

系统

1 切换至新增的主机，可在“**系统**”页面查看主机的硬件信息、负载概览等。

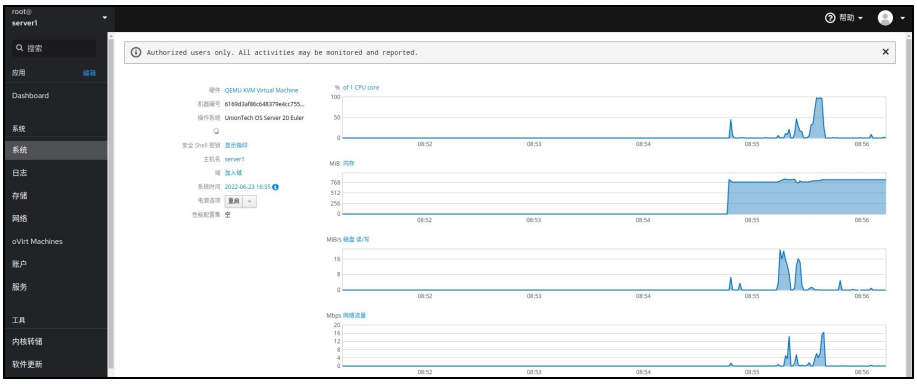


图 6.30 cockpit 系统页面

2 单击硬件名称，可以查看系统的硬件信息和 PCI 设备信息。

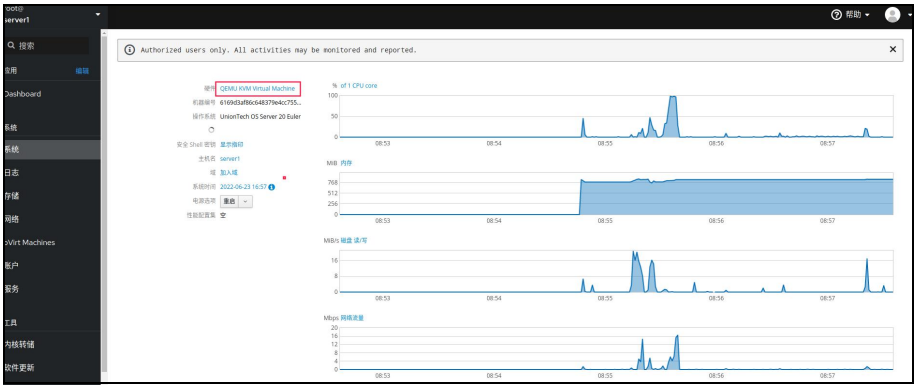


图 6.31 查看硬件信息

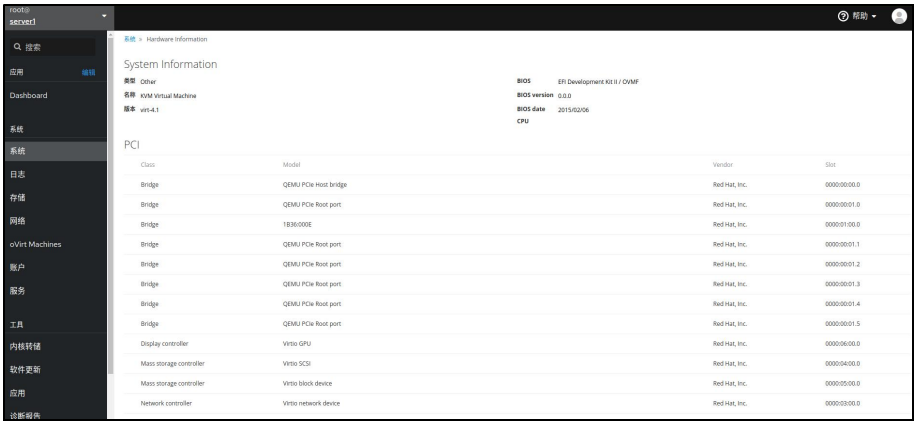


图 6.32 显示系统信息

日志

Cockpit 将日志信息分为错误、警告、注意等不同的分类，可以看出这是按事件的级别进行分类的。

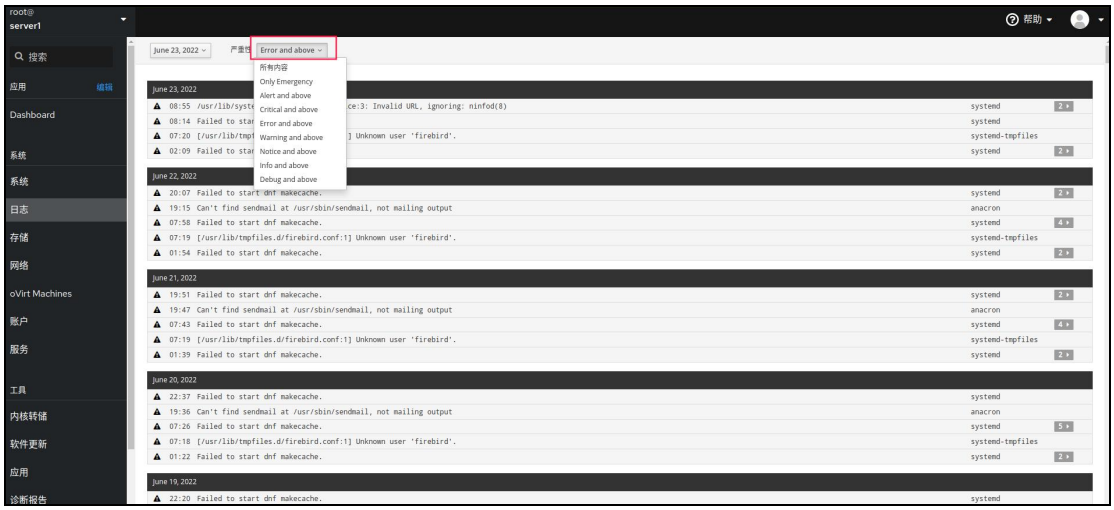


图 6.33 cockpit 日志分类

存储

Cockpit 可以方便地查看硬盘的读写速度。我们可以查看存储的 Journal

日志，以便进行故障排除和修复。在页面中还有一个已用空间的可视化图。在存储页面可以卸载、格式化、删除一块硬盘的某个分区。它还有类似创建 RAID 设备、卷组等功能。

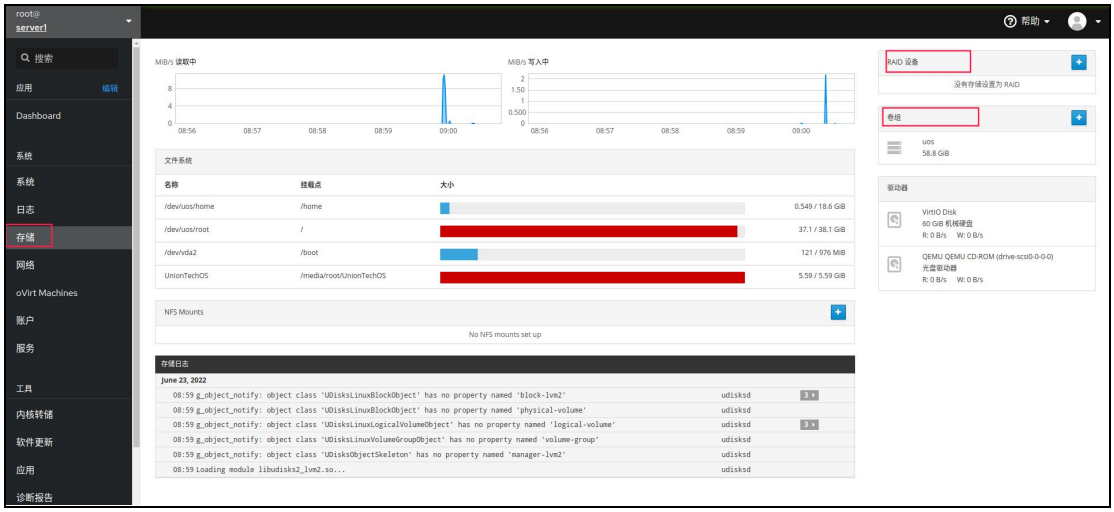


图 6.34 cockpit 存储管理

网络

在“**网络**”页面，可以看到两个可视化发送和接收速度的图。还可以看到可用网卡的列表，可以对网卡进行绑定设置、桥接，以及配置 VLAN。单击网卡，就可以进行编辑操作。最下面的是网络的 Journal 日志信息。

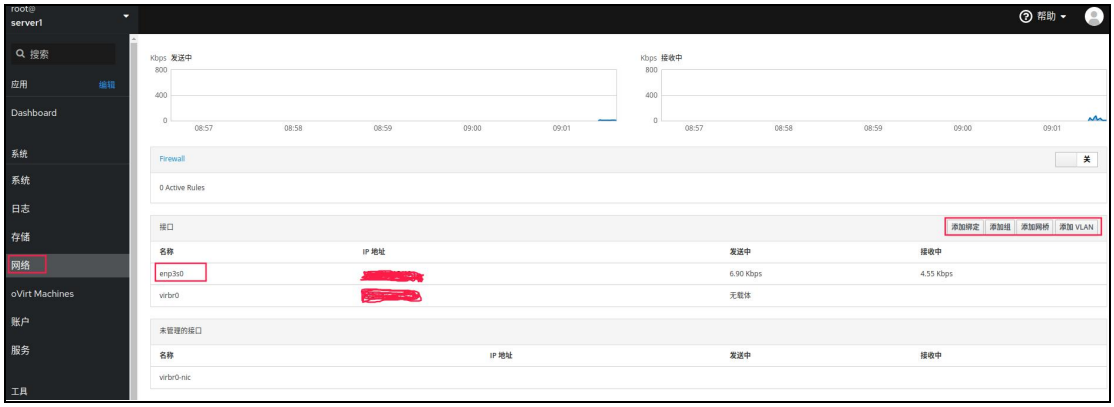


图 6.35 cockpit 管理网络

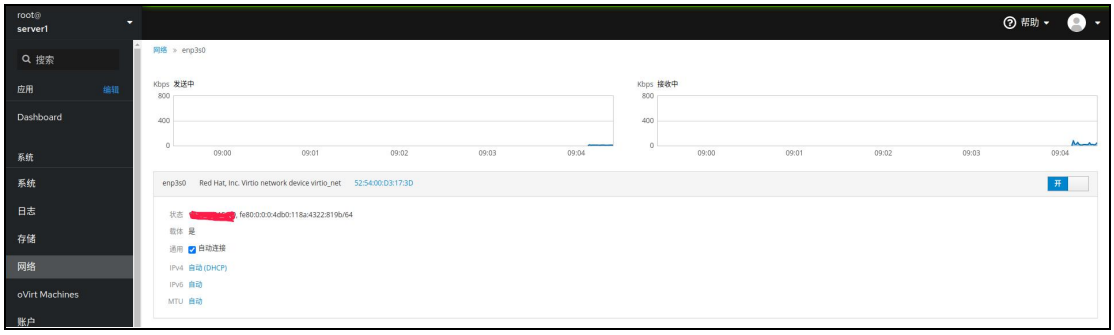


图 6.36 cockpit 配置网络

虚拟机

如果被监控的主机为 KVM 主机，那么其中运行的虚拟机也可以在 cockpit 中进行管理操作。在虚拟机页面中会列出当前主机中的所有 KVM 虚拟机。

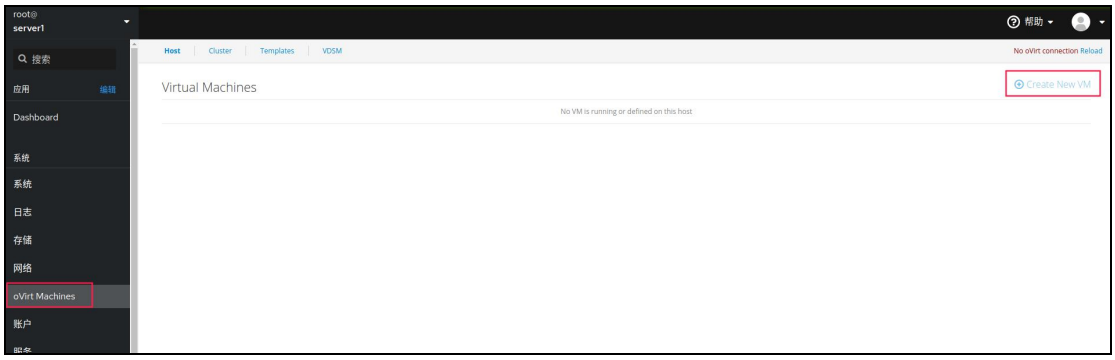


图 6.37 使用 cockpit 管理虚拟机

帐户

在“**帐号**”页面中，可以添加、修改、删除系统帐户。

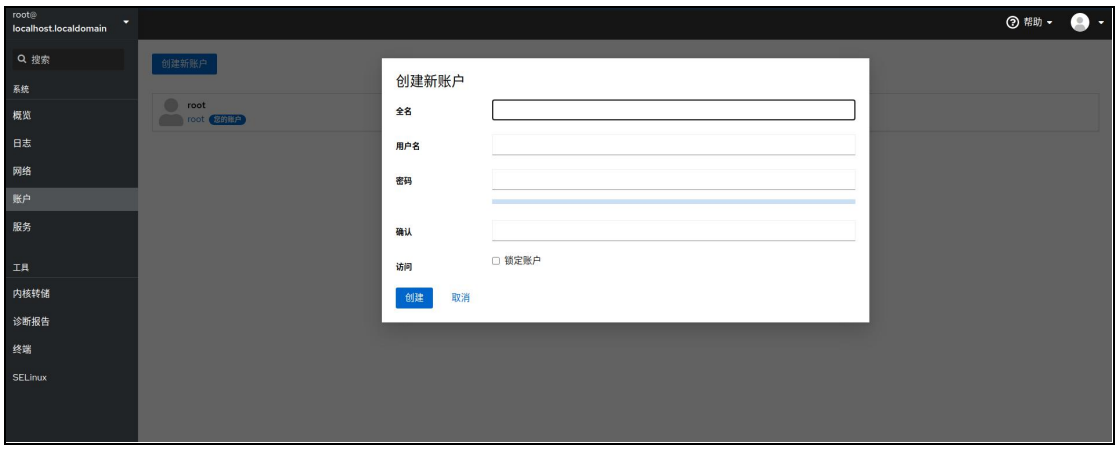


图 6.38 使用 cockpit 管理系统帐户

图 6.39 使用 cockpit 管理用户

服务

服务被分成了 5 个类别：目标、系统服务、套接字、计时器和路径。

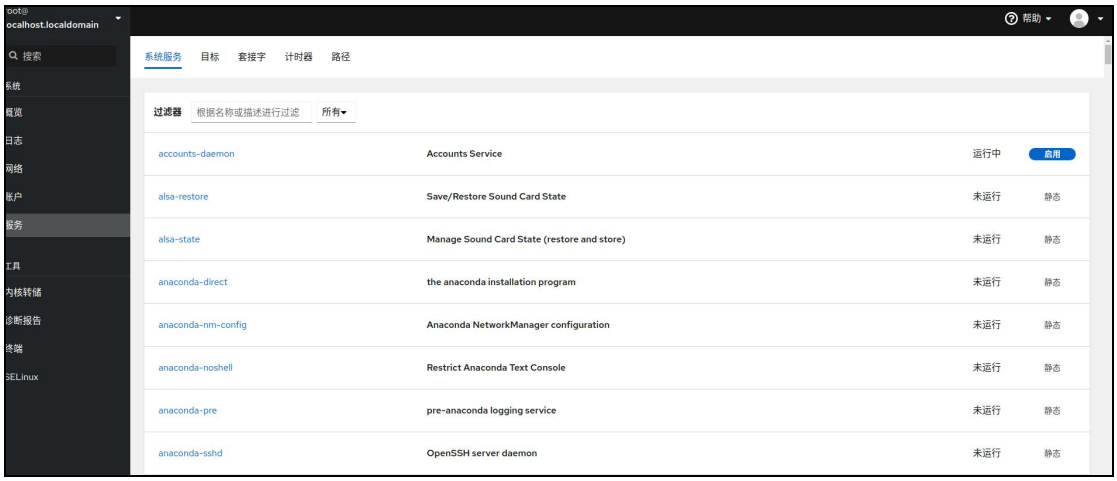


图 6.40 使用 cockpit 管理服务

单击服务名称，可以对其进行更多的管理操作，例如重启、关闭、禁用等。

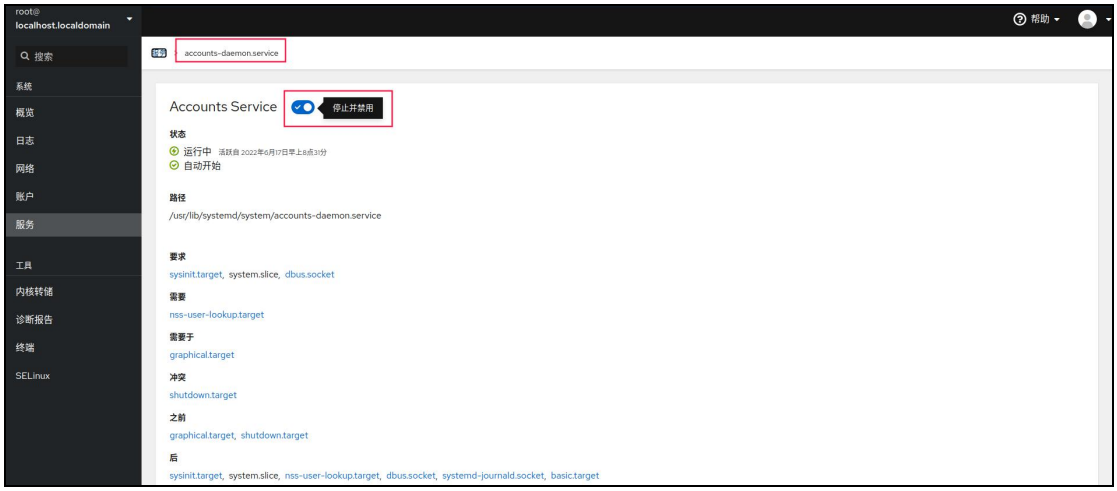


图 6.41 使用 cockpit 管理服务启停

内核转储

可以配置 kdump 服务打开或者关闭，也可以设置故障转储日志的位置。

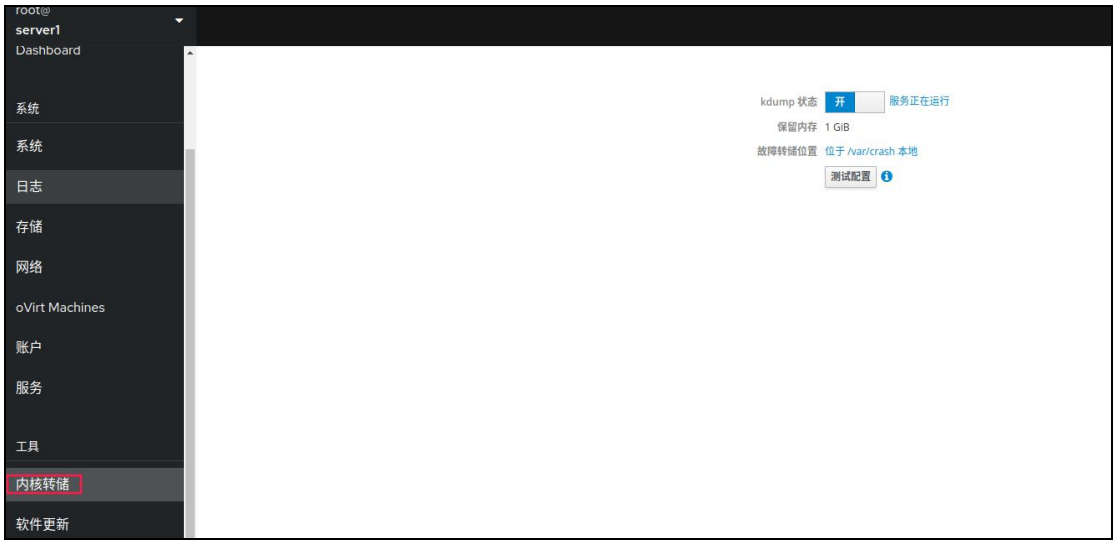


图 6.42 使用 cockpit 管理 kdump

软件更新

Cockpit 可根据更新重要程度给出相应提示和说明，进而进行软件的更新。

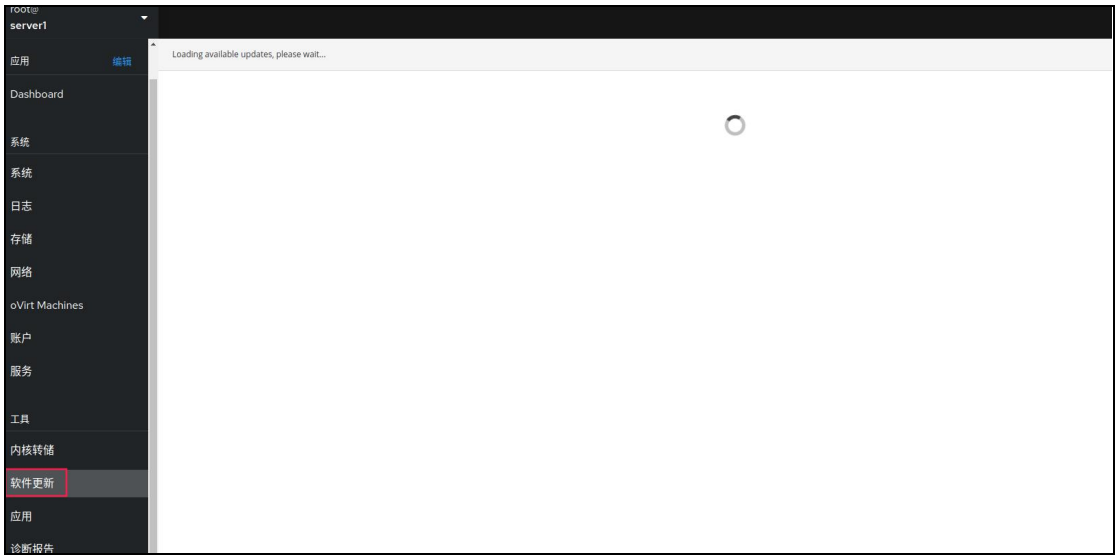


图 6.43 使用 cockpit 管理软件更新

应用

应用程序屏幕显示可用于 Cockpit 的加载项列表 ,这使查找和安装用户所需的插件变得容易。

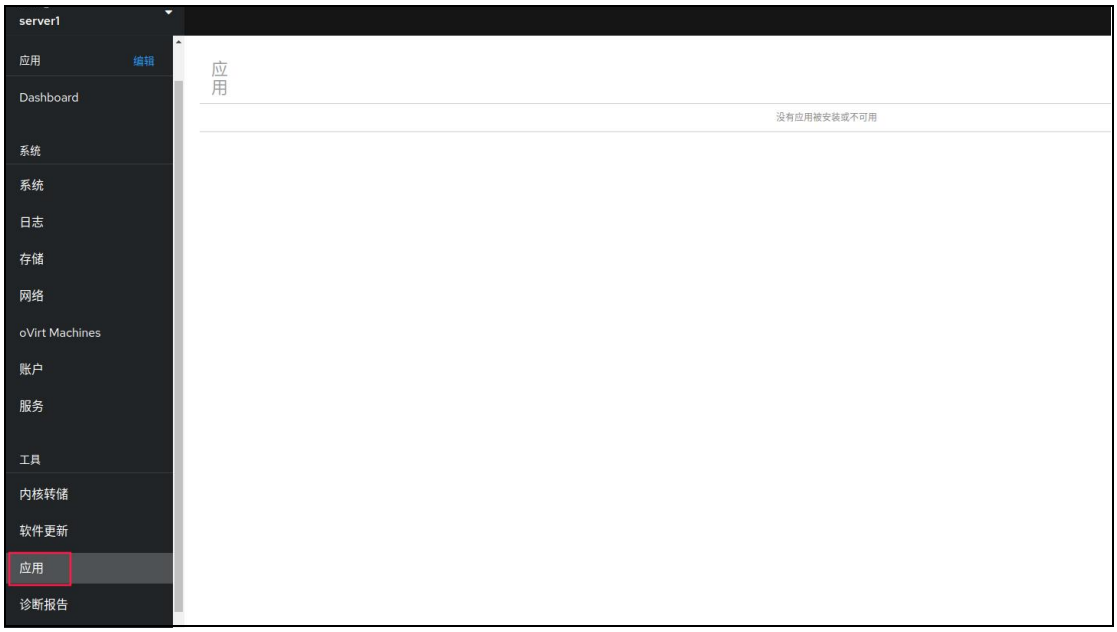


图 6.44 管理 cockpit 加载项

终端

Cockpit 界面提供实时终端执行任务，这使我们可以根据需求在 Web 界面和终端之间自由切换，可以快速执行任务，操作非常方便。

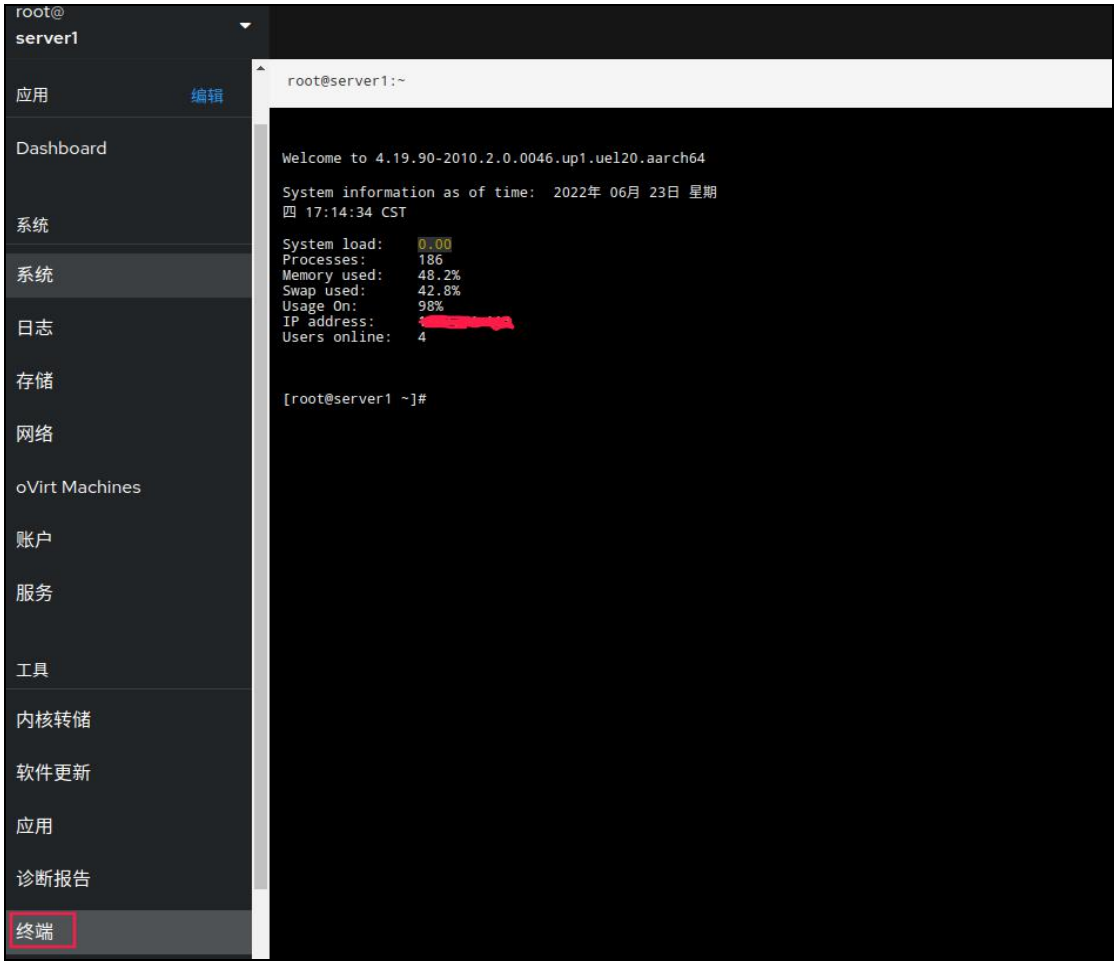


图 6.45 使用 cockpit 连接主机终端

Diagnostic Report

通过 cockpit 的诊断报告功能，可以快速生成系统的 sosreport，并且可以下载到本地。

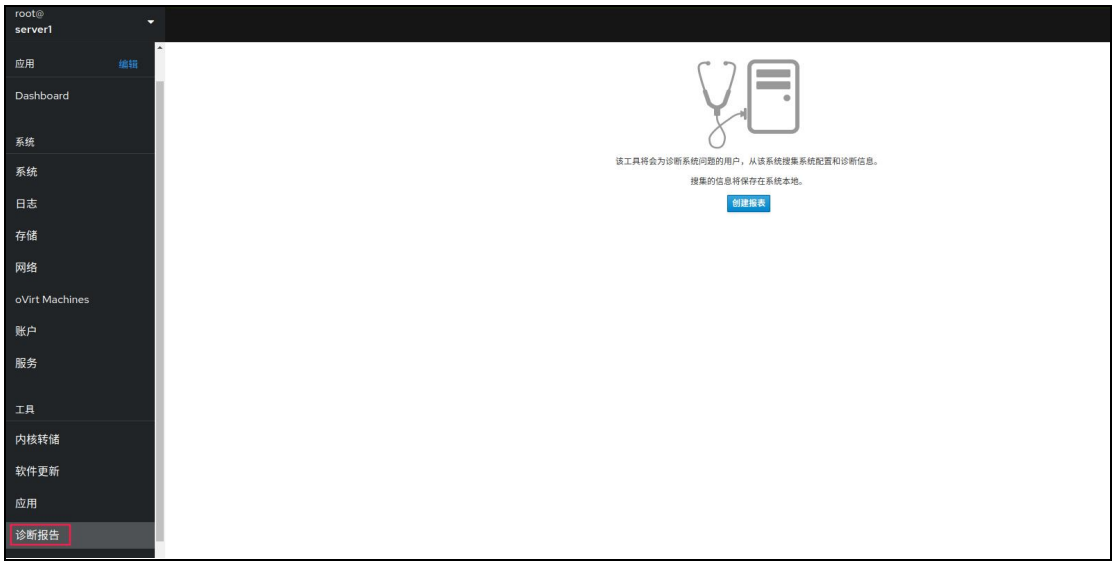


图 6.46 使用 cockpit 创建系统诊断报告

SELinux

可以监控 Selinux 的开启和关闭状态。

图 6.47 使用 cockpit 监控系统 selinux 启用状态

6.12 ansible

6.12.1 概述

Ansible 是一款开源运维自动化工具，通过 Ansible 可以实现运维自动化、提高运维工程师的工作效率、减少人为失误，实现了批量系统配置、批量程序部署、批量运行命令等功能。ansible 是基于模块工作的，ansible 本身没有批量部署的能力，ansible 只提供一个框架，通过 ansbile 所运行的模块实现批量部署的功能。

- 知识点：
  - 基于 Python 开发，默认基于 ssh 协议控制所有节点，支持多并发。
  - 基于模块化工作，内置了丰富的模块。
  - 受控端不需要安装客户端，不启动任何服务，但是需要安装 python3-simplejson。
  - 受控端如果要开启 SElinux，需要安装 python3-libselinux。
- 官网参考指导：http://www.ansible.com.cn/index.html

6.12.2 安装部署的前提条件

- 系统可以访问外网或者已配置本地 yum 源。
- 已安装统信服务器操作系统。
- 本示例不开启 SElinux。

6.12.3 安装主控端

执行如下命令，安装 ansible 包。

yum install *ansible* -y

6.12.4 安装受控端

执行如下命令，安装 python3-simplejson 包。

yum install *python3-simplejson* -y

6.12.5 Ansible 配置

1 在主控端，执行如下命令，生成 SSH 免密密钥对。

[root@localhost ~]# **ssh-keygen** -t *rsa*

2 执行如下命令，拷贝公钥到受控端。

[root@localhost ~]# **ssh-copy-id** -i *.ssh/id_rsa.pub root@受控端 IP 地址*

6.12.6 Ansible 的使用

Inventory 使用方法

*说明：Inventory 是 Ansible 管理主机信息的配置文件，默认存放在“**/etc/ansible/hosts**”中。在该文件中，可以定义主机，也可以定义主机组，实现对主机进行管理。同时，还可以定义 hosts 文件，对其进行管理，详情请参见“**Inventory 使用**”。*

■ 配置规则如下：

- ◆ 以[]包含部分代表组名，设备列表支持主机名和 IP 地址。
- ◆ 默认使用 22 端口作为管理设备端口。如果使用了非默认端口，则需要在主机名称或者 IP 地址之后使用冒号加端口进行标明。
- ◆ hosts 文件支持通配符操作。

■ 配置举例：

```
[web]
192.168.1.2
192.168.1.3
[test]
www.zabbix_test.com:8090
[database]
oracle1.com
oracle[2:5].com     ##[2:5]表示 2~5 之间的所有数字 即表示 oracle1.com、oracle2.com…的所有主机
```

Inventory 使用示例

- 执行如下命令，只对 web 组中的 192.168.1.2 主机操作，通过--limit 参数限定主机的变更。

  ansible *web* -m *command* -a *"systemctl status sshd"* --limit *"192.168.1.2"*

- 执行如下命令，只对 192.168.1.3 主机操作，通过 IP 地址限定主机的变更。

  ansible *192.168.1.3* -m *command* -a *"systemctl status sshd"*

- 执行如下命令，只对 192.168.1.0 网段主机操作。

  ansible *192.168.1.** -m *command* -a *"systemctl status sshd"*

- 执行如下命令，对所有默认 hosts 文件内的所有主机进行操作。

  ansible all -f *5* -m *ping*

  说明：ping 在这里表示 ping 模块，并不是指操作系统自带的 ping 命令。

- 执行如下命令，列出所有 web 组内的所有主机。

  ansible *web* --list

6.12.7 Modules 的使用

正如概述中所提到的，Ansible 采用模块进行工作，在管理受控端时，Ansible 封装了多个模块进行命令下发、执行来管理受控端。常用模块如下所示：

command 模块

command 模块在远程主机执行命令，不支持管道、重定向等 shell 的特性。

表 6.4 command 模块常用选项描述

参数	描述
chdir	在执行命令之前，先切换到该目录。
creates	一个文件名，当这个文件存在，则该命令不执行。
executable	切换 shell 来执行命令，需要使用命令的绝对路径。
free_form=	要执行的 Linux 指令，一般使用 Ansible 的-a 参数代替。
removes	一个文件名，当这个文件不存在，则该命令不执行。

*说明：在执行的命令中，有管道或者变量时，就需要使用 shell 模块，例如：[root@localhost ~]# **ansible** all -m command -a "mkdir /opt/uos"。*

shell 模块

shell 模块在远程主机执行命令，相当于调用远程主机的 Shell 进程，然后在该 Shell 下打开一个子 Shell 运行命令。和 command 模块的区别是，它支持 Shell 特性，如管道、重定向等。例如：

[root@localhost ~]# **ansible** *web* -m *shell* -a *"echo uos"*

```
[root@localhost ~]# ansible web -m shell -a "echo uoser > ./uostest.txt"
```

copy 模块

copy 模块用于复制指定主机文件到远程主机的指定位置。

表 6.5 copy 模块常用参数如下

参数	描述
backup	在覆盖之前，将源文件备份，备份文件包含时间信息。有两个选项：yes 和 no。
content	用于替代“**src**”，可以直接设定指定文件的值。
dest	必选项。将源文件复制到的远程主机的绝对路径，如果源文件是一个目录，那么该路径也必须是个目录。
directory_mode	递归设定目录的权限，默认为系统默认权限。
force	如果目标主机包含该文件，但内容不同，如果设置为 yes，则强制覆盖；如果为 no，则只有当目标主机的目标位置不存在该文件时，才复制。默认为 yes。
group	应该拥有文件/目录的组的名称，将被馈送到'chown'。
mode	应该是模式的文件或目录，如 0644 将被馈送到'chmod'。
owner	应该拥有文件/目录的用户的名称，将被馈送到'chown'。
recurse	复制源目录中的所有内容。
others	所有的 file 模块里的选项都可以在这里使用。
src	被复制到远程主机的本地文件，可以是绝对路径，也可以是相对路径。如果路径是一个目录，它将递归复制。在这种情况下，如果路径使用“/”来结尾，则只复制目录里的内容；如果没有使用“/”来结尾，则包含目录在内的整个内容全部复制，类似于 rsync。

例如：

```
[root@localhost ~]# ansible all -m copy -a 'src=/root/test.txt dest=/opt/uos'
[root@master opt]# ansible 192.168.1.2 -m copy -a "src=/root/test.txt dest=/opt/uos/ mode=0440 force=yes"
```

hostname 模块

hostname 模块用于管理远程主机上的主机名。

表 6.6 hostname 模块参数说明

参数	描述

参数	描述
name	指定主机名。

例如：

```
[root@localhost ~]# ansible 192.168.1.2 -m hostname -a "name=uos"
```

yum 模块

yum 模块基于 yum 机制，对远程主机管理程序包。

表 6.7 yum 模块参数说明

参数	描述
name	程序包名称，可以带上版本号。若不指明版本，则默认为最新版本。
state=present|atest|absent	指明对程序包执行的操作，present 表明安装程序包，latest 表示安装最新版本的程序包，absent 表示卸载程序包。
disablerepo	在用 yum 安装时，临时禁用某个仓库的 ID。
enablerepo	在用 yum 安装时，临时启用某个仓库的 ID。
conf_file	yum 运行时的配置文件，而不是使用默认的配置文件。
disable_gpg_check=yes|no	是否启用完整性校验功能。

例如：

```
[root@localhost ~]# ansible web -m shell -a "/usr/bin/rm -rf /root/test.txt"
[root@localhost ~]# ansible web -m yum -a "name=httpd state=present"
```

service 模块

service 模块用来管理远程主机上的服务。

表 6.8 service 模块参数说明

参数	描述
name	被管理的服务名称。
state=started|stopped|restarted	动作包含启动、关闭或重启。
enable=yes|no	表示是否设置该服务开机自启动。
runlevel	如果设定了 enabled 开机自启动，则要定义在哪些运行目标下自动启动。

例如：

```
[root@localhost ~]# ansible web -m service -a "name=httpd enabled=yes state=started"
```

user 模块

user 模块主要用于管理远程主机上的用户帐号。

表 6.9 user 模块参数说明

参数	描述
append	如果是 yes，就是给这个用户添加一个组。
comment	可选择地设置用户帐户的描述（也称为“GECOS”）。
createhome	除非设置为 no，否则当创建帐户或主目录不存在时，将为用户创建主目录。
force	当与“state = absent”一起使用时，行为与“userdel --force”一样。
generate_ssh_key	为 yes 时，生成秘钥对。生成密钥时，只会生成公钥文件和私钥文件。和直接使用 ssh-keygen 指令效果相同，不会生成 authorized_keys 文件。
group	设置用户主组，后面跟用户组名称。
groups	设置多个组，当设置为'groups='的时候，用户将从主组以外的所有组移除。
home	可以选择设置用户的主目录。
move_home	如果与“home =”一起使用时设置为“yes”，则尝试将用户的主目录移动到指定的目录（如果尚未存在）。
name=	创建、删除或修改用户的名称。
non_unique	（可选）当与-u 选项一起使用时，该选项允许将用户 ID 更改为非唯一值。
password	（可选）将用户密码设置为此加密值。
remove	当与“state = absent”一起使用时，行为与“userdel --remove”一样。
shell	可选择设置用户的 shell。
ssh_key_bits	可选择指定要创建的 SSH 密钥中的位数。
ssh_key_comment	（可选）定义 SSH 密钥的注释。
ssh_key_file	（可选）指定 SSH 密钥文件名。
ssh_key_passphrase	设置 SSH 密钥的密码。如果没有提供密码，SSH 密钥将默认没有密码。

参数	描述
ssh_key_type	（可选）指定要生成的 SSH 密钥的类型。可用的 SSH 密钥类型将取决于目标主机上的实现。
state	帐户是否应该存在。当“absent”时，删除用户帐户。
system	创建帐户时，将其设置为“yes”后，将使用户成为系统帐户。现有用户无法更改此设置。
uid	可选择地设置用户的“UID”。
update_password	如果不同，“always”会更新密码。'on_create'只会为新创建的用户设置密码。

例如：

```
[root@host01 ~]# ansible web -m user -a "name=uoser system=yes uid=1000 group=uoser groups=uoser shell=/bin/bash home=/home/uoser password=DuPy_9ton"
```

cron 模块

cron 模块被用来定时执行计划任务。

表 6.10 cron 模块参数说明

参数	描述
backup	如果为 yes，那么在修改之后会进行备份，备份的路径在“**backup_file**”中。
cron_file	如果指定，请在 cron.d 中使用此文件，而不是单个用户的 crontab。
day	工作应该运行的月份的第几天（1-31，*，*/2 等）。
hour	工作应该运行的小时（0-23，*，* / 2 等）。
job	执行命令。如果 state =present 则为必需。
minute	工作应该运行一分钟（0-59，*，*/2 等）。
month	工作应该运行的一个月（1-12，*，*/2 等）。
name	crontab 条目的说明。
state	present（不存在就添加；存在，且如果是注释状态就取消状态启动此定时任务），absent 为删除定时任务。
user	crontab 应该修改的具体用户。
weekday	工作应该运行的星期几（周六至周六为 0-6）。

例如：

```
[root@localhost~]# ansible all -m cron -a 'name="modify time" minute=*/3 user=uoser job="/usr/sbin/ntpdate 0.uos.pool.ntp.org"'
```

说明：

- *name：为定时任务的名称，也就是定时任务的注释。*
- *minute：为分钟。hour 为小时、day 为天、month 为月、weekday 为星期，它们默认都为*。*
- *user：将定时任务添加到哪个用户的定时任务中。*
- *job：里面的内容为定时任务的内容。*

script 模块

script 模块用来在受控端执行本地脚本。

表 6.11 script 模块参数说明

参数	描述
creates	一个文件名，当这个文件存在，则该命令不执行。
free_form=	本地脚本路径。
removes	远程节点上的文件名，当它不存在时，不会运行此步骤。

例如：

```
[root@localhost~]# ansible all -m script -a "/root/test.sh"
```

6.12.8 附录

Ansible 主要核心部件

- Inventory 使用：Inventory 是 Ansible 管理主机信息的配置文件，相当于系统 Hosts 文件的功能，默认存放在“**/etc/ansible/hosts**”。在 hosts 文件中，通过分组来组织设备。Ansible 通过 Inventory 来定义主机和分组，也可以自定义“**hosts**”文件来管理主机，通过在 ansible 命令中使用选项-i 或--inventory-file 来指定“**hosts**”文件。

表 6.12 Inventory 配置参数

参数	说明
ansible_ssh_host	SSH 目标主机名或 IP 地址。
ansible_ssh_port	SSH 目的端口（默认是 22）。
ansible_ssh_user	SSH 登录使用的用户名（root）。
ansible_ssh_pass	SSH 认证所使用的密码（默认是 none，这是不安全的，ansible 极力推荐使用--ask-pass 选项或使用 SSH keys）。
ansible_sudo_pass	SSH 认证的 sudo 用户的密码。
ansible_connection	ansible 支持多种 transport 机制，ansible 使用何种连接模式连接到主机，连接类型有

参数	说明
	local、ssh、docker、paramiko（默认是 smart，也就是智能选择）。
ansible_ssh_private_key_file	SSH 认证所使用的私钥（默认是 none）。
ansible_shell_type	命令所使用目标系统的 shell 类型（默认是 sh），除非您设置的 ansible_shell_executable 与默认的 sh 不兼容，否则您不应该设置这个参数。默认情况下，命令是在 sh shell 环境风格下运行的，此处可以设置为 csh 或 fish shell。
ansible_python_interpreter	主机上的 python 解释器（默认是“**/usr/bin/python**”）。
ansible_*_interpreter	如果不是用 python 编写的自定义模块，可以使用这个参数来指定解释器的路径（如“**/usr/bin/ruby**”），默认是 none。

例如：执行 **cat** */etc/ansible/custom_host* 命令后，显示如下内容。

```
[uoser]
master_uos      ansible_ssh_host=192.168.1.2      ansible_ssh_port=22
ansible_ssh_user=uos                              ansible_ssh_pass=123456
ansible_shell_type=sh
#master_uos:   主机名
#ansible_ssh_host：   主机 IP 地址
#ansible_ssh_port：   SSH 监控端口
#ansible_ssh_user：   SSH 登录使用的用户名
#ansible_ssh_pass：   SSH 登录使用的用户名密码
#ansible_shell_type：   命令所使用的 shell
```

■ 使用方法。

ansible -i */etc/ansible/custom_host all* -m *shell* -a *"whoami"*

Ansible 常用命令

■ Ansible-doc：Ansible-doc 用来查询 ansible 模块文档的说明，类似于 man 命令。针对每个模块都有详细的用法说明及应用案例介绍，语法如下。

```
ansible-doc [options] [module……]
```

◆ 列出支持的模块。

```
[root@localhost ansible]# ansible-doc -l
```

◆ 查询具体模块的信息。

```
[root@localhost ansible]# ansible-doc ping
```

■ Ansible-playbook：Playbook 是 ansible 用于配置、部署和管理被控节点的工具。其工作机制是通过读取预先编写好的 playbook 文件实现集中处理任务。playbook 由 YMAL 语言编写，YAML 文件的扩展名通常为.yaml 或.yml。

◆ 创建 playbook.yml 文件

```
mkdir -p /etc/ansible/playbooks/
cd etc/ansible/playbooks/
vi playbook.yml
```

◆ 语法以及内容编写参考指导：

https://github.com/ansible/ansible-examples

◆ 使用 playbook.yml

```
ansible-playbook playbook.yml
```

■ Ansible-console：Ansible-console 是 Ansible 为用户提供的一款交互式工具，用户可在 ansible-console 虚拟出来的终端上用 Ansible 内置的命令。

```
[root@localhost ansible]# ansible-console
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
Welcome to the ansible console.
Type help or ? to list commands.
root@all (0)[f:5]$ help
Documented commands (type help <topic>):
========================================
EOF
a10
a10_server
a10_server_axapi3
a10_service_group
a10_virtual_server
accelerate
aci
aci_aaa_user
aci_aaa_user_certificate
aci_access_port_block_to_access_port
```

```
aci_access_port_to_interface_policy_leaf_profile
aci_access_sub_port_block_to_access_port
```

6.13 x11vnc

6.13.1 概述

x11vnc 是虚拟网络计算（VNC）服务器程序。它允许从远程客户端到承载 X Window 会话和 x11vnc 软件的计算机的远程访问，并持续轮询 X 服务器的帧缓冲区以查找更改。这使用户可以从用户自己的网络上的远程计算机或通过 Internet 来控制自己的 X11 桌面（KDE、GNOME、Xfce 等）。x11vnc 还可以轮询非 X11 帧缓冲设备，例如网络摄像头或电视调谐器卡、iPAQ、Neuros OSD、Linux 控制台和 Mac OS X 图形显示。

x11vnc 具有安全性功能，允许用户设置访问密码或使用 Unix 用户名和密码。它还具有用于通过安全 SSL 链接进行连接的选项。它还提供 SSL Java VNC 查看器小程序，该小程序启用了来自 Web 浏览器的安全连接，并支持 VeNCrypt SSL / TLS VNC 安全类型。

6.13.2 安装部署

前提条件

- 系统已经安装了 DDE 桌面。
- 系统可以访问外网或者已配置本地 yum 源。

安装 Server 端

1 执行如下命令，安装 x11vnc。

```
yum install x11vnc -y
```

2 执行如下命令，创建密码。（可选）

```
x11vnc -storepasswd
```

```
[root@localhost ~]# x11vnc -storepasswd
Enter VNC password:
Verify password:
Write password to /root/.vnc/passwd?  [y]/n y
Password written to: /root/.vnc/passwd
[root@localhost ~]#
```

图 6.48 创建连接 x11vnc 的密码

3 执行如下命令，修改默认监听端口。（可选）

```
vi /usr/lib/systemd/system/x11vnc.service
```

说明：x11vnc 默认端口为 5900，根据需要可以将其修改为其他端口。

图 6.49 x11vnc 服务单元文件中默认端口 5900

4 执行如下命令，设置开机启动服务。

```
systemctl daemon-reload
systemctl start x11vnc.service
systemctl enable x11vnc.service
```

5 执行如下命令，关闭防火墙。

```
systemctl stop firewalld
```

安装远程访问客户端

说明

- *本示例以远程工具 Remmina 为例进行说明。*
- *本地系统为统信 UOS 专业版。*
- *在 Windows 系统中，请下载并安装对应的版本进行验证（例如：tigervnc、vncviewer 等）。*

1 执行如下命令，安装 Remmina 包。

```
apt install remmina -y
```

2 远程连接服务器。

3 在“**启动器**”中，打开客户端“**remmina**”后，单击左上角的“**+**”。

图 6.50 remmina 中添加远程连接环境

4 在“**协议**”下拉菜单中，选择“**Remmina VNC 插件**”，并在输入服务器的

“**用户名**”和“**用户密码**”后，单击“**连接**”。

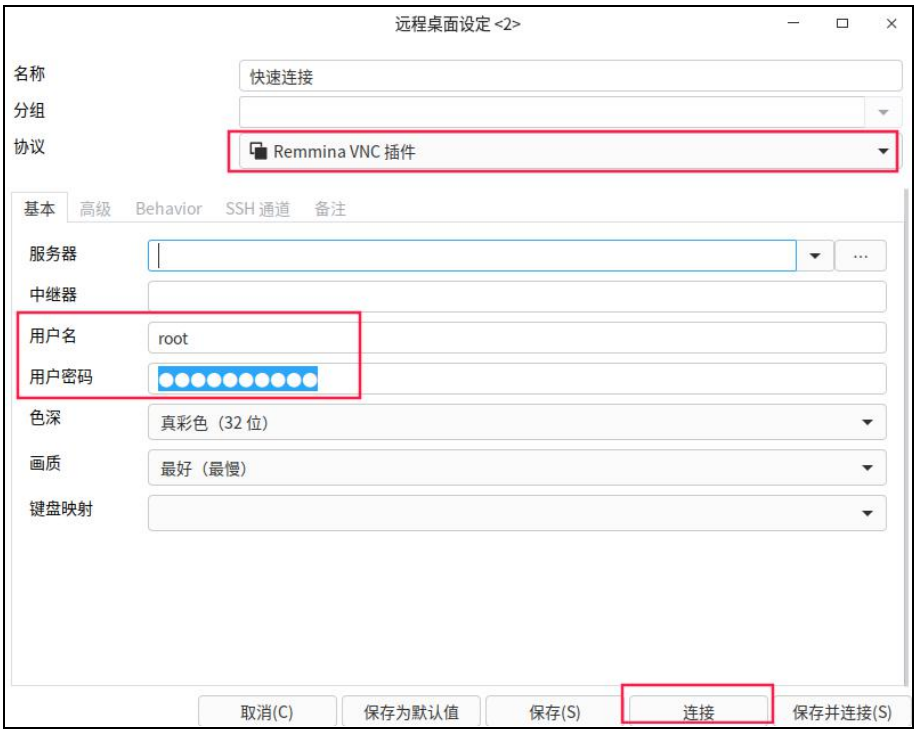


图 6.51 remmina 配置远程连接详细信息

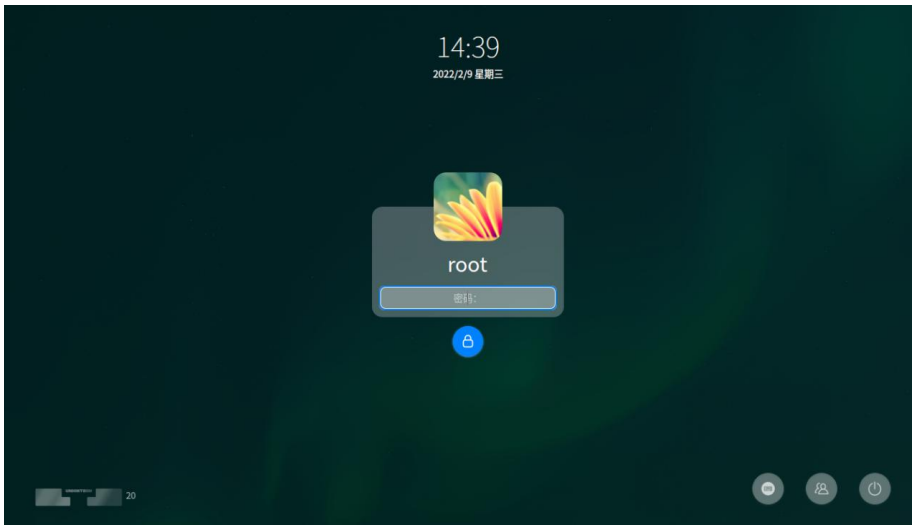


图 6.52 remmina 成功连接远程桌面

7 AppStream 使用指南

7.1 AppStream 介绍

通常来说，每个系统版本只能包含每个包的一个主要版本。同时，每个包需要在其所属系统版本的整个生命周期中保持一定的 API/ABI 稳定性。

开发人员通常倾向于选择最新版本的软件，而服务器管理员则希望 API/ABI 在较长时间内保持稳定。这导致很难在太快和太慢的发行周期中找到平衡点，而

这一问题可以通过 AppStream 这种软件分发方式解决。

传统方式中，分发的软件被放在 BaseOS、Update 和 Everything 三个主要的存储仓中。AppStream 是在这个基础上引入 modular 仓库，这个模块化仓库中用于保存编程语言栈、数据库以及 web 服务器和一系列相关软件工具包，并为这些软件包提供多个可用版本。

7.1.1 基本概念

- 包：包是软件发行的最小单位。
- 模块：模块是包的集合，代表一个逻辑单元。模块中的所有包被一起构建、测试和发行。
- 流：流是模块的一个版本，由一系列有特定目标的软件包版本组成。例如提供 API/ABI 兼容、软件的最新或者稳定版本。
- 档案：模块中的包以不同的使用目的划分成的集合。

7.1.2 模块和流三者之间的关系

- 同一模块可以为同一发行版提供多个流。
- 同一流可以用于多个发行版。

7.2 AppStream 常用命令

在 AppStream 中，模块使用 module-spec 进行描述。module-spec 由三部分组成，格式为 name:stream/profile（模块名:流名/档案名）。在没有指定 stream 和 profile 的情况下，命令会自动选择默认的。需要注意的是，并不是所有的模块都有默认的 stream 和 profile。

7.2.1 相关命令

- 命令格式。

```
dnf module <command> [ 选项 ] <参数>...
```

- 常用命令。

表 7.1 dnf module 命令参数说明

命令及参数	说明
dnf module install <module-spec>...	安装模块中的包。
dnf module update <module-spec>...	更新已启用的模块的流中的包。
dnf module remove <module-spec>...	移除模块 profile 中不被依赖的包。当未指定 profile 时，移除所有 profile 中不被依赖的包。
dnf module remove --all <module-spec>...	在 remove 的基础上，移除所有流提供的包名，被依赖的包和其

命令及参数	说明
	他模块中提供同名的包不会被移除。
dnf module enable <module-spec>...	使模块指定 profile 的包可用。
dnf module disable <module-name>...	使模块中所有流不可用。
dnf module reset <module-name>...	使模块恢复默认状态。
dnf module provides <package-name>...	列举包含指定包名的所有模块，包含已被禁用的模块。
dnf module list [--all] [module_name...]	列举模块的所有流状态。
dnf module list --enabled [module_name...]	列举模块中被启用的流。
dnf module list --disabled [module_name...]	列举模块中被禁用的流。
dnf module list --installed [module_name...]	列举模块中已安装的流。
dnf module info <module-spec>...	打印模块的详细信息。
dnf module info [--profile] <module-spec>...	只打印模块的详细 profile 信息。
dnf module repoquery <module-spec>...	列举模块中已启用流的所有可用包。
dnf module repoquery --available <module-spec>...	列举模块中的所有可用包。
dnf module repoquery --installed <module-spec>...	列举模块中已安装的包。

■ 执行示例。

◆ 执行 **dnf module list** 命令，列举所有模块。

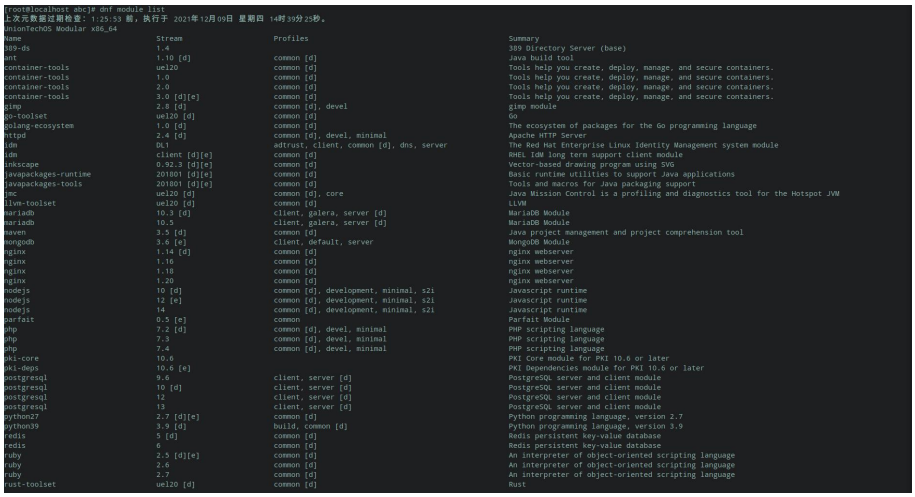


图 7.2 列出所有可用模块化

◆ 执行 **dnf module provides** *php-common* 命令，查看有 php-common 包的模块。

图 7.3 查询提供 php-common 包的模块

◆ 执行 **dnf module list** *php* 命令，列举 php 模块的所有流。

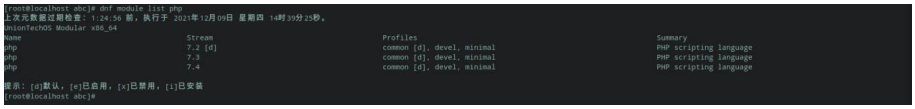


图 7.4 列出所有 php 模块流

◆ 执行 **dnf module info** *php* 命令，查看 php 模块的详细信息。

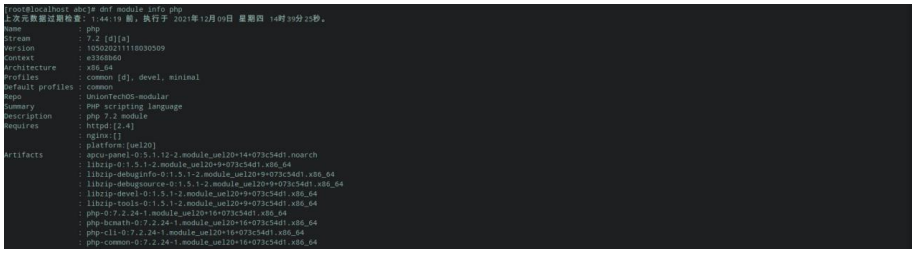


图 7.5 查看 php 模块详细信息

◆ 执行 **dnf module info** --profile *php* 命令 查看 php 模块的 profile 信息。

图 7.6 查看 php 模块的 profile 信息

◆ 执行 **dnf module enable** *php:7.2* 命令，使能 php 模块 7.2 流。

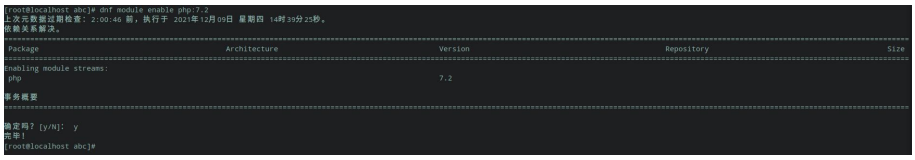


图 7.7 使用 php:7.2 模块流

◆ 执行 **dnf module repoquery** *php* 命令，查看 php 模块已启用流中的包。

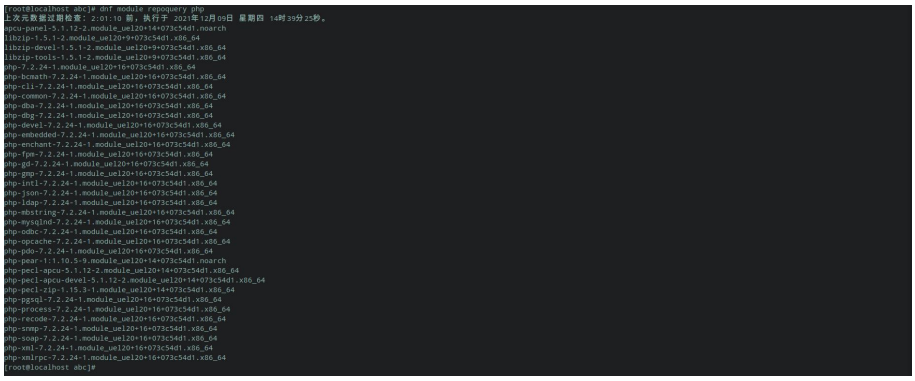


图 7.8 查看 php 模块已启用流中的包

◆ 执行 **dnf module list** --enabled *php* 命令，查看 php 模块中已启用的流。

图 7.9 查看 php 模块已启用的流

◆ 执行 **dnf module install** *php:7.2/devel* 命令，安装 php 模块、流为 7.2 的 devel 档案中包含的所有包。

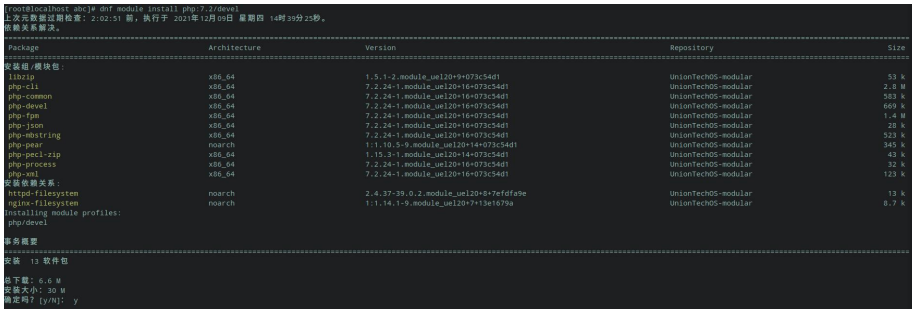


图 7.10 安装 php:7.2/devel 中所有的包

- 执行 **dnf module list** --installed *php* 命令，查看 php 模块中已安装的流。

图 7.11 查看 php 模块已安装的流

- 执行 **dnf module reset** *php* 命令，使 php 模块恢复默认状态。

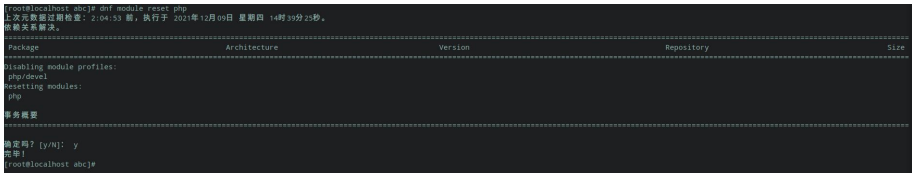


图 7.12 恢复 php 模块的默认状态

- 执行 **dnf module remove** *php* 命令，移除 php 模块中的包。

图 7.13 卸载已安装的 php 模块中的包

- 执行 **dnf module remove** --all *php* 命令，移除 php 模块中的包。

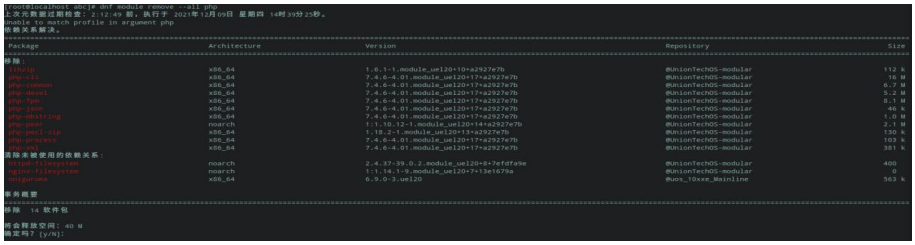


图 7.14 卸载已安装的 php 模块中的所有包

- 执行 **dnf module disable** *php* 命令，禁用 php 模块中的所有流。

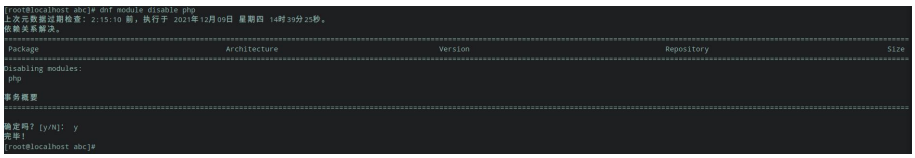


图 7.15 禁用 php 模块中的所有流

8 虚拟化管理

8.1 概述

虚拟化就是在一台物理服务器上可以运行多台虚拟机，且每台虚拟机可以运行不同的操作系统。所有虚拟机共享物理机的 CPU、MEM、I/O 资源等，虚拟机之间是互相隔离的。统信服务器操作系统支持 AMD64 和 ARM64 架构的 KVM 虚拟化组件，同时支持 virsh 命令和 virt-manager 图形化界面管理虚拟机。

8.2 KVM+Libvirt+QEMU 虚拟化

8.2.1 前提条件

- 宿主机必须为物理机。
- 物理网口上的 IP 地址采用 DHCP 方式获取。
- 本示例采用桥接网络，将虚拟机网卡桥接到 br0 上面。

8.2.2 安装虚拟化组件

1 执行如下命令，安装 kvm 虚拟化组件。

```
yum install qemu libvirt virt-manager -y
```

2 执行如下命令，安装 edk2 组件。（不同的架构，安装方法不一样。）

```
# AMD64 架构安装 edk2-ovmf
yum install edk2-ovmf -y
# ARM64 架构安装 edk2-aarch64.noarch
yum install edk2-aarch64.noarch -y
```

3 执行如下命令，启动 libvirt 服务。

```
systemctl start libvirtd
```

4 执行如下命令，启动 cockpit 服务。

```
systemctl start cockpit
```

5 执行如下命令，关闭防火墙。

```
systemctl stop firewalld
```

6 配置网桥 br0。

（1）打开浏览器，在地址栏中输入 https://IP 地址:9090，并在按“**Enter**”键后，输入 root 用户的用户名和密码。

说明：请将链接里面的 IP 地址替换为实际的本机 IP 地址。

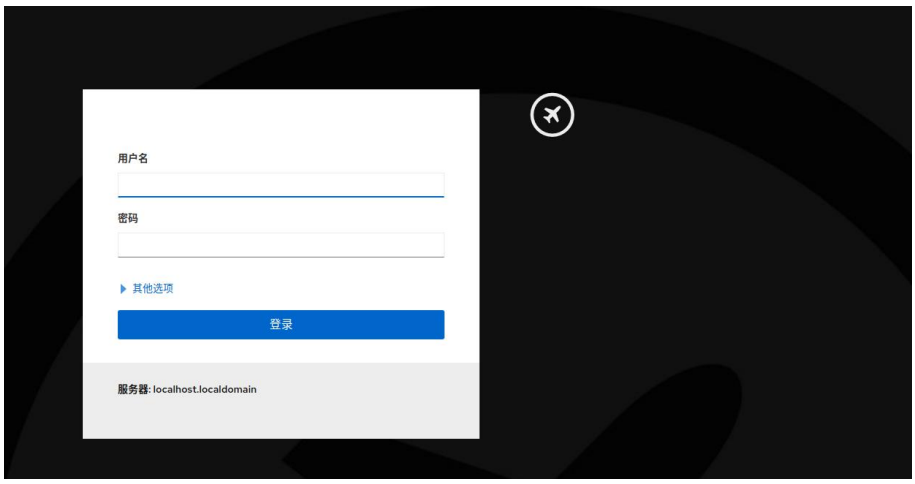


图 8.1 cockpit 的登录页面

（2）登录成功后左侧菜单栏选择“**网络**”标签，查看网络管理功能

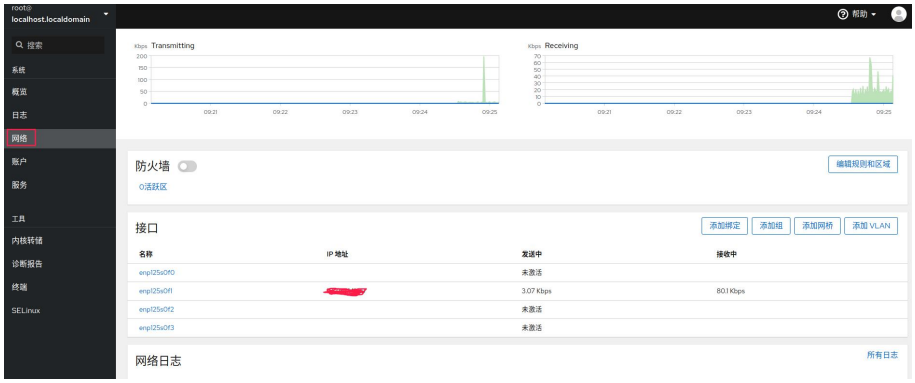


图 8.2 cockpit 的网络管理功能

（3）在“**名称**”文本框中，输入“**br0**”。在“**端口**”区域框中，勾选物理机的网口，选择完毕后单击左下角“**应用**”。

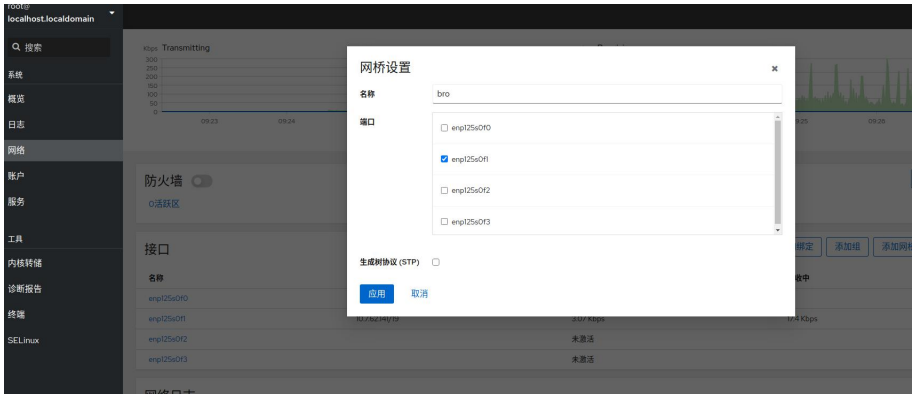


图 8.3 添加网桥 br0,绑定到 enp125s0f0 网卡

（4）网桥配置成功后可以在**网络管理**界面查看到配置结果。

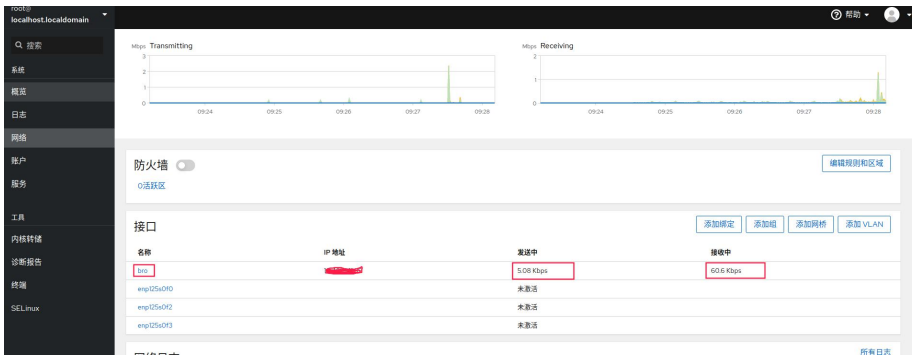


图 8.4 已配置的网桥 br0

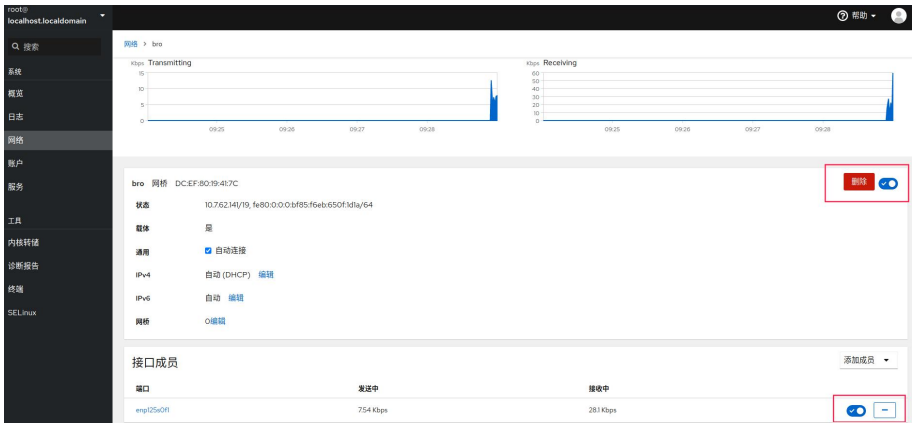


图 8.5 启用网桥 br0，启用物理端口

8.2.3 虚拟机 XML 配置

Libvirt 工具采用 XML 格式的文件描述一个虚拟机特征，包括虚拟机名称、CPU、内存、磁盘、网卡、鼠标、键盘等。用户可通过修改配置文件，对虚拟机进行管理。本章介绍 XML 配置文件各个元素的含义，指导用户完成虚拟机配置。

domain 和 name 说明

- domain：虚拟机 XML 配置文件的根元素，用于配置运行此虚拟机的 hypervisor 的类型。属性 type 表示虚拟化中 domain 的类型，虚拟化中属性值为 kvm。
- name：虚拟机名称，虚拟机名称为一个字符串。同一个主机上的虚拟机名称不能重复。虚拟机名称必须由数字、字母、“_”、“—”、“:”组成，但不支持全数字的字符串，且虚拟机名称不超过 64 个字符。

vcpu 和 memory 说明

- vcpu：虚拟处理器的个数。
- memory：虚拟内存的大小。属性 unit 用于指定内存单位，属性值支持 KiB（210 字节）、MiB（220 字节）、GiB（230 字节）和 TiB（240 字节）等。
- cpu：虚拟处理器模式。

- 属性 mode：表示虚拟 CPU 的模式。host-passthrough 表示虚拟 CPU 的架构和特性与主机保持一致，custom 表示虚拟 CPU 的架构和特性由此 cpu 元素控制。
- 子元素 topology：元素 cpu 的子元素，用于描述虚拟 CPU 模式的拓扑结构。子元素 topology 的属性 socket、cores、threads 分别描述了虚拟机具有多少个 cpu socket、每个 cpu socket 中包含多少个处理核心（core）、每个处理器核心具有多少个超线程（thread），属性值为正整数，且三者的乘积等于虚拟 CPU 的个数。
- 子元素 model：元素 cpu 的子元素，当 mode 为 custom 时用于描述 CPU 的模型。
- 子元素 feature：元素 cpu 的子元素，当 mode 为 custom 时用于描述某一特性的使能情况。其中，属性 name 表示特性的名称，属性 policy 表示这一特性的使能控制策略。

  force：表示强制使能该特性，无论主机 CPU 是否支持该特性。

  require：表示使能该特性，当主机 CPU 不支持该特性并且 hypervisor 不支持模拟该特性时，创建虚拟机失败。

  optional：表示该特性的使能情况与主机上该特性的使能情况保持一致。

  disable：禁用该特性。

  forbid：禁用该特性，当主机支持该特性时创建虚拟机失败。

devices 说明

虚拟机 XML 配置文件使用 devices 元素配置虚拟设备，包括存储设备、网络设备、总线、鼠标等，本节介绍如何配置常用的虚拟设备。

XML 配置文件使用 disk 元素配置存储设备，disk 常见的属性如表 8.1 所示，常见子元素及子元素属性如表 8.2 所示。

表 8.1 元素 disk 的常用属性

元素	属性	含义	属性值及其含义
disk	type	指定后端存储介质类型。	■ block：块设备。 ■ file：文件设备。 ■ dir：目录路径。
	device	指定呈现给虚拟机的存储介质。	■ disk：磁盘。（默认） ■ floppy：软盘。 ■ cdrom：光盘。

表 8.2 元素 disk 的常用子元素及属性说明

子元素	子元素含义	属性说明

子元素	子元素含义	属性说明
source	指定后端存储介质，与 disk 元素的属性“type”指定类型相对应。	■ file：对应 file 类型，值为对应文件的完全限定路径。 ■ dev：对应 block 类型，值为对应主机设备的完全限定路径。 ■ dir：对应 dir 类型，值为用作磁盘目录的完全限定路径。 ■ protocol：使用的协议。 ■ name：rbd 磁盘名称，格式为“**$pool/$volume**”。 ■ host name：mon 地址。 ■ port：mon 地址的端口。
driver	指定后端驱动的详细信息。	■ type：磁盘格式的类型，常用有“**raw**”和“**qcow2**”，需要与 source 的格式一致。 ■ io：磁盘 IO 模式，支持“**native**”和“**threads**”选项。 ■ cache：磁盘的 cache 模式，可选项有“**none**”、“**writethrough**”、“**writeback**”、“**directsync**”等。 ■ iothread：指定为磁盘分配的 IO 线程。
target	指磁盘呈现给虚拟机的总线和设备。	■ dev：指定磁盘的逻辑设备名称，如 SCSI、SATA、USB 类型总线常用命令习惯为 sd[a-p]，IDE 类型设备磁盘常用命名习惯为 hd[a-d]。 ■ bus：指定磁盘设备的类型，常见的有“**scsi**”、“**usb**”、“**sata**”、“**virtio**”等类型。
boot	表示此磁盘可以作为启动盘使用。	■ order：指定磁盘的启动顺序。
readonly	表示磁盘具有只读属性，磁盘内容不可以被虚拟机修改，通常与光驱结合使用。	-

interface 说明

虚拟机中的网络设备描述。其属性“**type**”表示虚拟网卡的模式，可选的值有“**ethernet**”、“**bridge**”、“**vhostuser**”等。下面以“**bridge**”模式虚拟网卡为例介绍其子元素以及对应的属性。

表 8.3 inteface 子元素及说明

子元素	子元素含义	属性及含义
mac	虚拟网卡的 mac 地址。	address：指定 mac 地址，若不配置，会自动生成。
target	后端虚拟网卡名。	dev：创建的后端 tap 设备的名称。
source	指定虚拟网卡后端。	bridge：与 bridge 模式联合使用，值为网桥名称。
boot	表示此网卡可以作为远程启动。	order：指定网卡的启动顺序。
model	表示虚拟网卡的类型。	type：bridge 模式网卡通常使用 virtio。
virtualport	端口类型。	type：若使用 OVS 网桥，需要配置为 openvswitch。
driver	后端驱动类型。	name：驱动名称，通常取值为 vhost。 queues：网卡设备队列数。

controller 说明

虚拟机 xml 中通过 controller 表示一个总线，一个控制器上通常可以挂载一个或多个控制器或设备

- 属性 type：控制器必选属性，表示总线类型。常用取值有“**pci**”、“**usb**”、“**scsi**”、“**virtio-serial**”、“**fdc**”、“**ccid**”。
- 属性 index：控制器必选属性，表示控制器的总线“**bus**”编号（编号从 0 开始），可以在地址元素“**address**”元素中使用。
- 属性 model：控制器必选属性，表示控制器的具体型号，其可选择的值与控制器类型“**type**”的值相关，对应关系及含义请参见表 8.4。
- 子元素 address：为设备或控制器指定其在总线网络中的挂载位置。属性 type 表示设备地址类型，常用取值有“**pci**”、“**usb**”、“**drive**”。address 的 type 类型不同，对应的属性也不同，常用 type 属性值及其该取值下 address 的属性请参见表 8.5。
- 子元素 model：控制器具体型号的名称。属性 name 用于指定控制器具体型号的名称，和父元素 controller 中的属性 model 对应。

表 8.4 controller 属性 type 常用取值和 model 取值对应关系

type 属性值	model 属性值	简介
pci	pcie-root	PCIe 根节点，可挂载 PCIe 设备或控制器。
	pcie-root-port	只有一个 slot，可以挂载 PCIe 设备或控制器。
	pcie-to-pci-bridge	PCIe 转 PCI 桥控制器，可挂载 PCI 设备。
usb	ehci	USB 2.0 控制器，可挂载 USB 2.0 设备。
	nec-xhci	USB 3.0 控制器，可挂载 USB 3.0 设备。
scsi	virtio-scsi	virtio 类型 SCSI 控制器，可以挂载块设备，如磁盘、光盘等。
virtio-serial	virtio-serial	virtio 类型串口控制器，可挂载串口设备，如 pty 串口。

表 8.5 address 元素不同设备类型下的属性说明

类型 type 属性值	含义	对应地址属性
pci	地址类型为 PCI 地址，表示该设备在 PCI 总线网络中的挂载位置。	■ domain：PCI 设备的域号。 ■ bus：PCI 设备的 bus 号。 ■ slot：PCI 设备的 device 号。 ■ function：PCI 设备的 function 号。 ■ multifunction：controller 元素可选，是否开启 multifunction 功能。
usb	地址类型为 USB 地址，表示该设备在 USB 总线中的位置。	■ bus：USB 设备的 bus 号。 ■ port：USB 设备的 port 号。
drive	地址类型为存储设备地址，表示所属的磁盘控制器，及其在总线中的位置。	■ controller：指定所属控制器号。 ■ bus：设备输出的 channel 号。 ■ target：存储设备 target 号。 ■ unit：存储设备 lun 号。

其它常用设备

除存储设备、网络设备外，XML 配置文件中还需要指定一些其他外部设备，本节介绍这些元素的配置方法。

■ serial：串口设备。属性 type 用于指定串口类型，常用属性值为 pty、tcp、pipe、file。

- video：媒体设备。属性 type 表示媒体设备类型，ARM64 架构常用属性值为 virtio，AMD64 架构通常使用属性值为 vga 或 cirrus。子元素 model 表示 video 的子元素，用于指定媒体设备类型。在 model 元素中，type 属性为 vga 表示配置 VGA 类型显卡；vram 属性代表显存大小，单位默认为 KB。例如：给 AMD64 架构虚拟机配置 16MB 的 VGA 类型的显卡，XML 示例如下，其中 vram 属性代表显存大小，单位默认为 KB。

```
<video>
    <model type='vga' vram='16384' heads='1' primary='yes'/>
</video>
```

- input：输出设备。
  - 属性 type：指定输出设备类型。常用属性值为 tabe、keyboard，分别表示输出设备为写字板、键盘。
  - 属性 bus：指定挂载的总线。常用属性值为 USB。
- emulator：模拟器应用路径。
- graphics：图形设备。
  - 属性 type：指定图形设备类型。常用属性值为 vnc。
  - 属性 listen：指定侦听的 IP 地址。

8.2.4 环境准备

1 执行如下命令，创建块设备文件。

qemu-img create -f *<imgFormat>* -o *<fileOption> <fileName> <diskSize>*

示例：**qemu-img create** -f *qcow2 uos-image.qcow2 100G*

说明：

- *imgFormat：镜像格式，取值为 raw, qcow2 等。*
- *fileOption：文件选项，用于设置镜像文件的特性。*
- *fileName：文件名称。*
- *diskSize：磁盘大小*

2 获取 xml 文件：请参见“**虚拟机 XML 文件**”获取创建虚拟机的 XML 文件。

3 获取 ISO 镜像后，请将 ISO 镜像文件和 XML 文件放在同一目录，方便管理。

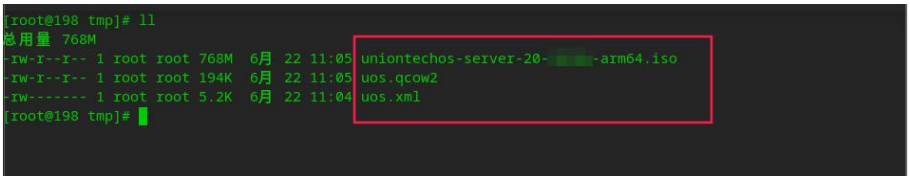


图 8.6 创建虚拟机所需文件示例

4 管理虚拟机，详情请参见“虚拟机管理”。

8.2.5 虚拟机管理

libvrit 采用 virsh 命令进行虚拟机的管理，本章介绍相关命令指导用户使用。

创建虚拟机

- 执行如下命令，定义持久化虚拟机。完成定义后，虚拟机处于关闭状态，并被看作一个 domian 实例，且不会启动。uos.xml 文件的内容，请参见“10.2 虚拟机 XML 文件”。

  virsh define *uos.xml*

- 执行如下命令，创建一个临时性虚拟机。完成创建后，虚拟机处于运行状态。

  virsh create *uos.xml*

启动虚拟机

执行如下命令，启动虚拟机。其中，vm_name 为 uos.xml 中定义的虚拟机名称或者 ID。（当采用 virsh define 命令定义持久化虚拟机时，需要执行此步骤。）

virsh start *vm_name*

关闭虚拟机

执行如下命令，启动虚拟机关机流程，若关机失败可使用强制关闭。

virsh shutdown *vm_name*

强制关闭虚拟机

virsh destroy *vm_name*

重启虚拟机

virsh reboot *vm_name*

转储虚拟机的运行状态

执行如下命令，将虚拟机的运行状态转储到文件中。

virsh save *vm_name vm_file*

恢复虚拟机

执行如下命令，从虚拟机转储文件恢复虚拟机。

virsh restore *vm_file*

暂停虚拟机

virsh suspend *vm_name*

唤醒虚拟机

virsh resume *vm_name*

永久销毁虚拟机

virsh undefine *vm_name*

查看运行的虚拟机列表

virsh list

查看所有的虚拟机

virsh list --all

查看虚拟机的配置信息

virsh dominfo *vm_name*

查看虚拟机网卡配置信息

virsh domiflist *vm_name*

查看虚拟机磁盘文件位置

virsh domblklist *vm_name*

查看 vcpu 个数

virsh vcpucount *vm_name*

查看虚拟机块设备状态

virsh domblkstat *vm_name*

查询虚拟机 I/O 线程及其 CPU 亲和性信息

virsh iothreadinfo *vm_name*

在线编辑虚拟机配置

创建虚拟机后，用户可以修改虚拟机的配置信息，称为在线修改虚拟机配置。在线修改配置以后，新的虚拟机配置文件会被持久化，并在虚拟机关闭、重启后生效。需要执行的命令如下：

virsh edit *vm_name*

8.2.6 登录虚拟机

在 Host 主机上创建完虚拟机后，用户可以通过 VNC 协议远程登录到虚拟机，并对其进行具体操作。

1 安装 VNC 客户端。（在本机上执行）

■ 统信 UOS 桌面操作系统：请执行如下命令进行安装。

sudo apt install *tigervnc-viewer*

■ Windows 操作系统：请参见如下链接的指导进行安装。

https://www.realvnc.com/en/connect/download/viewer/windows/

2 执行如下命令，查看虚拟机使用的 VNC 端口。（在 Host 上执行）

virsh vncdisplay *vm_name*

图 8.7 显示通过主机连接虚拟机所使用的 vnc 端口号

3 打开 vnc 客户端软件，输入主机 IP 地址和端口号，格式为“**主机 IP:端口号**”。单击“**Connect**”。

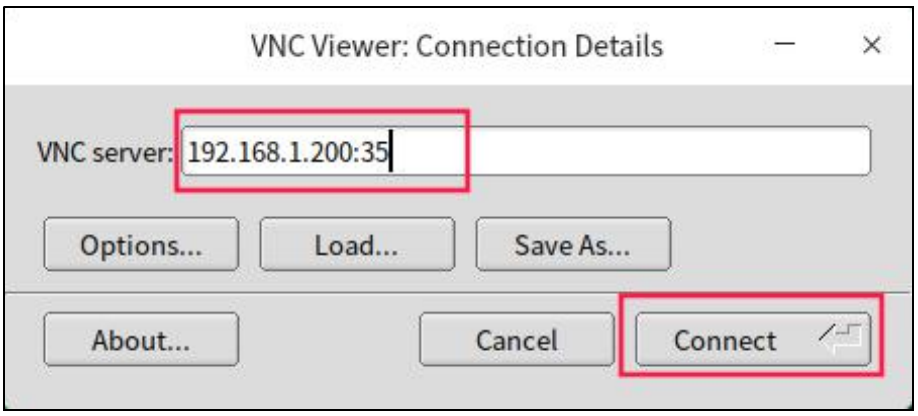


图 8.8 使用 vnc 查看器连接虚拟机

4 执行如下命令，开放端口。

firewall-cmd --zone=*public* --add-port=*5935/tcp*

说明：使用 vnc 客户端登录需要关闭虚拟机的防火墙，或者在防火墙中增加对该端口的放行规则。端口规则是“5900+X”，其中的 X 为步骤 2 中显示的数值，例如本示例的端口为 5935。

5 输入虚拟机的用户名和密码，登录到虚拟机 VNC 进行操作。

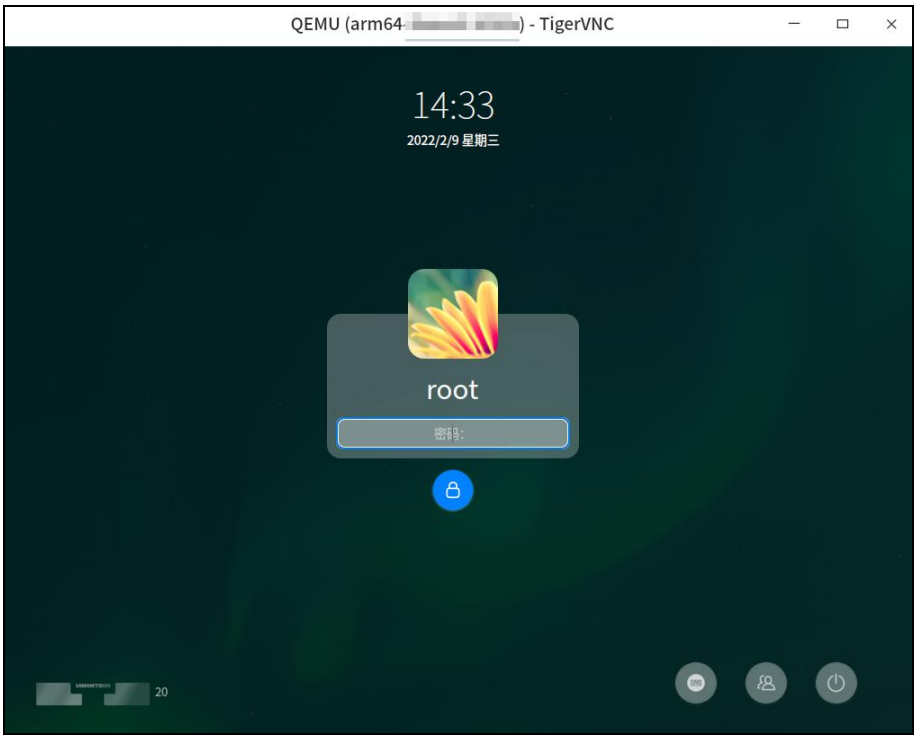


图 8.9 成功通过 vnc 连接到虚拟机

8.2.7 虚拟机其他操作

自定义 CPU Model

1 修改虚拟机 XML 文件，例如，下方示例中，将 CPU 处理模式修改为了 custom、将 model 设置为了 Kunpeng-920、禁用了 pmull 特性。

```
<domain type='kvm'>
    ...
    <vcpu>4</vcpu>
    <memory unit='GiB'>8</memory>
    <cpu mode='custom'>
        <model>Kunpeng-920</model>
        <feature policy='disable' name='pmull'/>
    </cpu>
    ...
</domain>
```

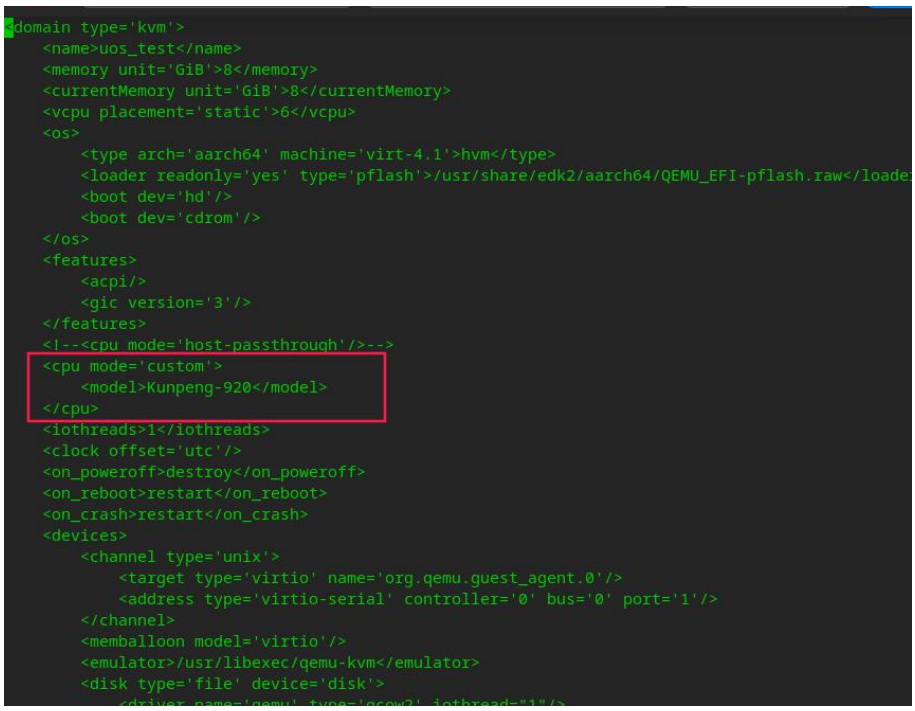


图 8.10 自定义 cpu mode

2 执行如下命令，启动虚拟机，然后观察修改后 CPU MODEL。

virsh create *uos.xml*

3 使用 vnc 或者 sssh 登录到虚拟机后，执行 **lscpu** 命令查看 CPU 型号。

lscpu

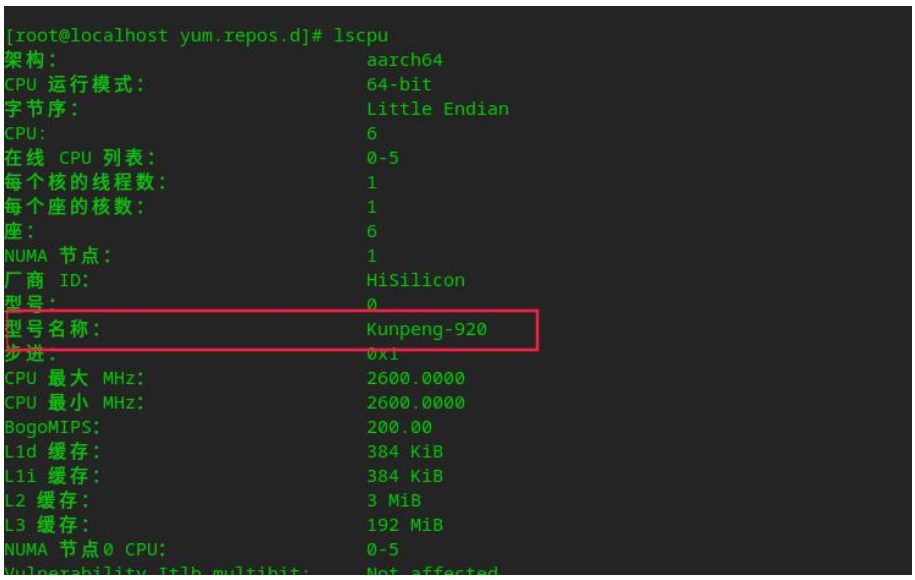


图 8.11 显示当前虚拟机中配置的 CPU 信息

CPU 热插拔

1 配置 uos.xml 文件，文件内容示例如下：

```
<vcpu placement='static' current='4'>8</vcpu>
<cpu mode='host-passthrough' check='none'>
```

```
<topology sockets="2" cores="4" threads="1" />
…
</cpu>
```

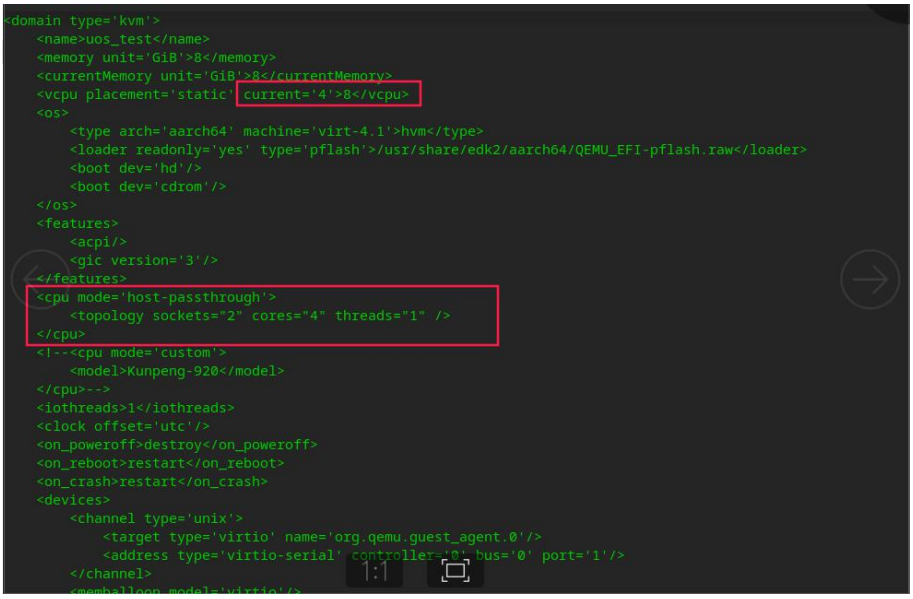


图 8.12 修改虚拟机 CPU 配置

2 执行如下命令，启动虚拟机。

```
virsh define uos.xml
virsh start uos
```

3 执行如下命令，执行 CPU 热插拔。

```
virsh setvcpus vm_name vcpu_counts --config --live
示例：virsh setvcpus uos 6 --config --live
```

说明：

- *vm_name：指的是虚拟机名称。*
- *vcpu_counts：代表最终要扩到的 vCPU 个数。*

4 查看 CPU 数量。

- VCPU 扩容前

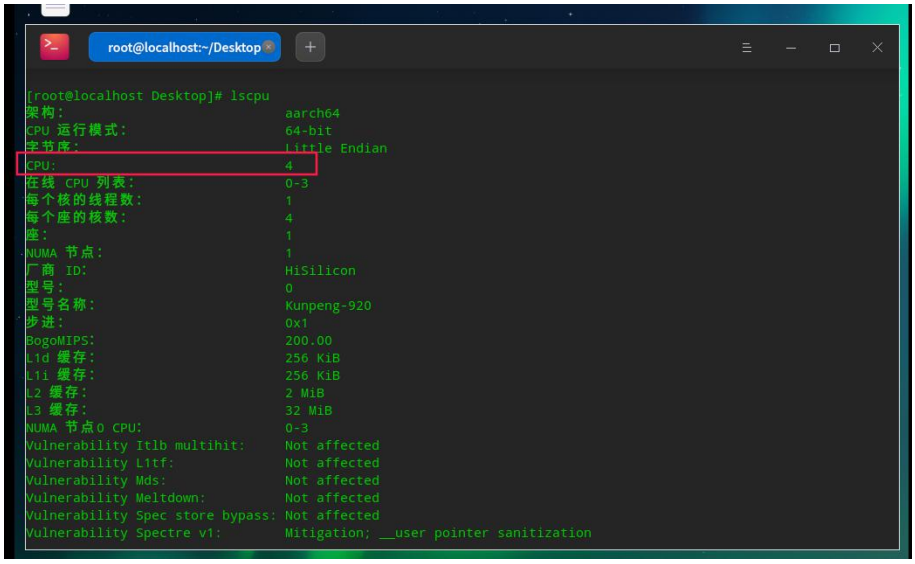


图 8.13 启动虚拟机后查看 CPU 数量

■ VCPU 扩容后

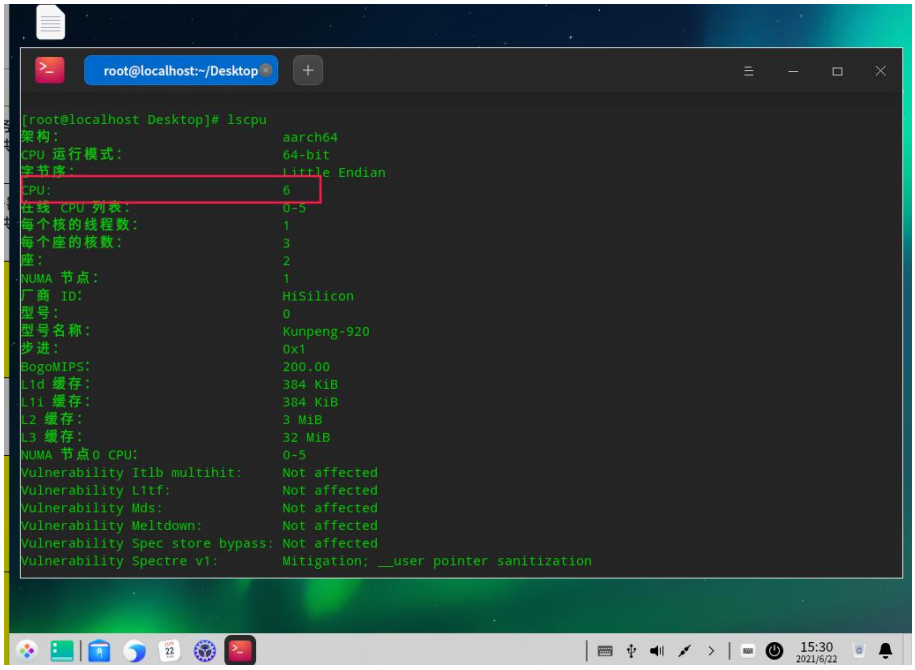


图 8.14 扩容后查看虚拟机的 CPU 数量

MEM 热插拔

1 在虚拟机的“**uos.xml**”文件中，增加如下参数。

```
.....
<memory unit='GiB'>4</memory>
    <!--<currentMemory unit='GiB'>4</currentMemory>-->
    <maxMemory slots='4' unit='GiB'>16</maxMemory>
    <vcpu placement='static' current='4'>8</vcpu>
    <cputune>
```

```
        <vcpupin vcpu='0' cpuset='0-3'/>
    </cputune>
    <numatune>
        <memnode cellid='0' mode='strict' nodeset='0'/>
    </numatune>
.....
    <cpu mode='host-passthrough' check='none'>
        <numa>
            <cell id='0' cpus='0-3' memory='4' unit='GiB'/>
        </numa>
    </cpu>
```

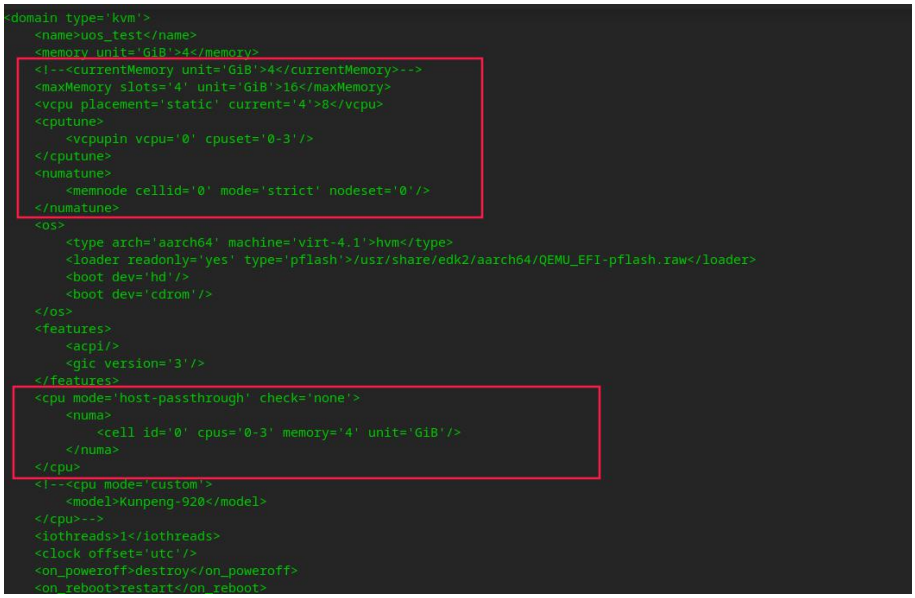


图 8.15 配置虚拟机内存热插拔

2 执行如下命令，创建扩容内存的 XML 文件。

```
# vi mem.xml
<memory model='dimm'>
    <target>
        <size unit='GiB'>2</size>
        <node>0</node>
    </target>
</memory>
```

3 执行如下命令，启动虚拟机。

```
virsh create uos.xml
```

4 执行如下命令，进行内存扩容。

virsh attach-device *vm_name memfile*

示例：**virsh attach-device** *uos mem.xml*

说明：

- *vm_name：虚拟机名称。*
- *memfile：需要扩容的内存 xml 文件。*

5 使用 vnc 或者 ssh 登录到虚拟机后，执行如下命令。

free -g

6 查看内存大小。

- MEM 扩容前

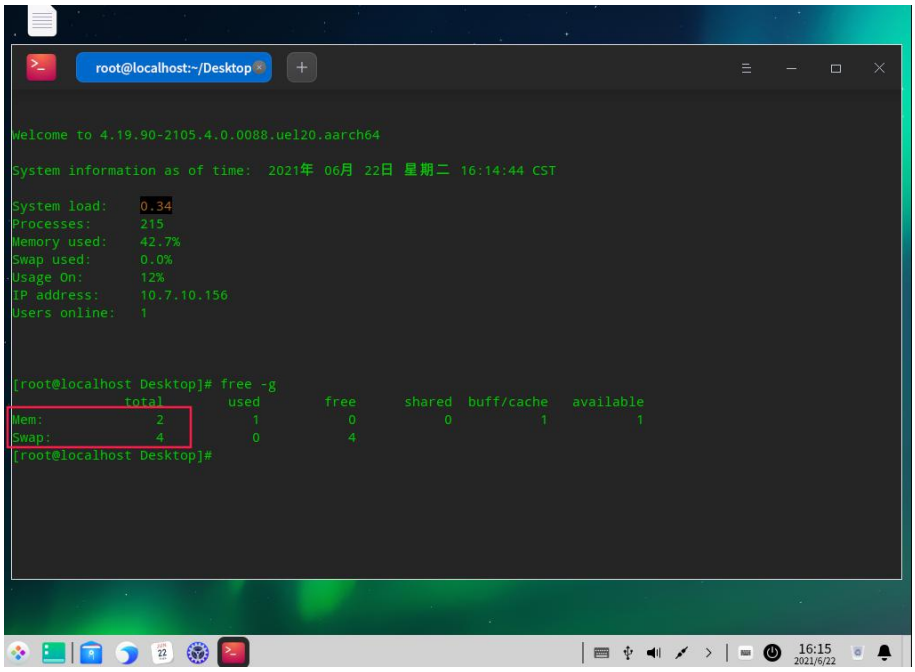


图 8.16 扩容前内存大小

- MEM 扩容后

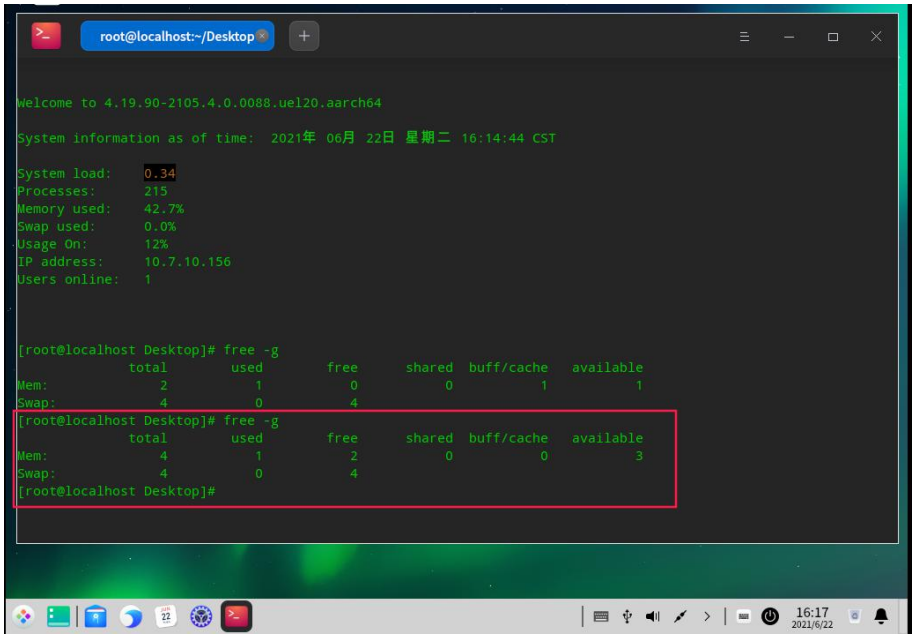


图 8.17 扩容后内存大小

DISK 热插拔

执行 DISK 热插拔的前提条件是虚拟机需正常运行。

1 执行如下命令，创建外挂磁盘。

qemu-img create -f *qcow2 extra_disk.qcow2 20G*

2 执行如下命令，创建磁盘 XML 文件。

```
# vi disk.xml
<disk type='file' device='disk'>
  <driver name='qemu' type=qcow2' cache='none'/>
  <source file='/uos/disk.img'/>
  <target dev='sdb' bus='scsi'/>
  <backingStore/>
</disk>
```

3 执行如下命令，对磁盘进行扩容。

virsh attach-device v*m_name extra_disk_xml_file*

示例：**virsh attach-device** *uos disk.xml*

说明：

- *vm_name：虚拟机名称。*
- *extra_disk_xml_file：磁盘扩容的 xml 文件*

4 查看磁盘信息。

- 磁盘扩容前

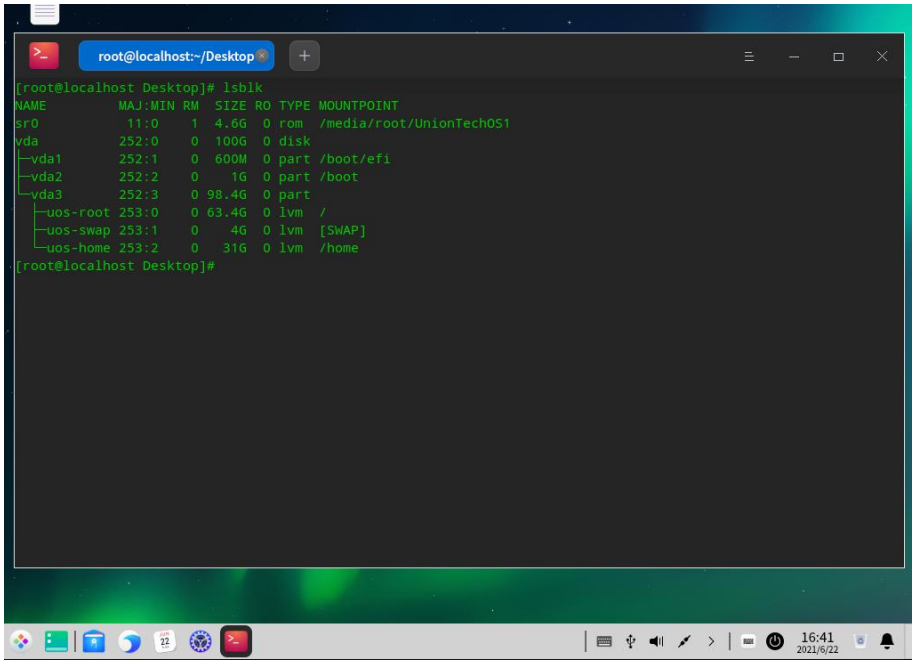


图 8.18 扩容前磁盘信息

■ 磁盘扩容后

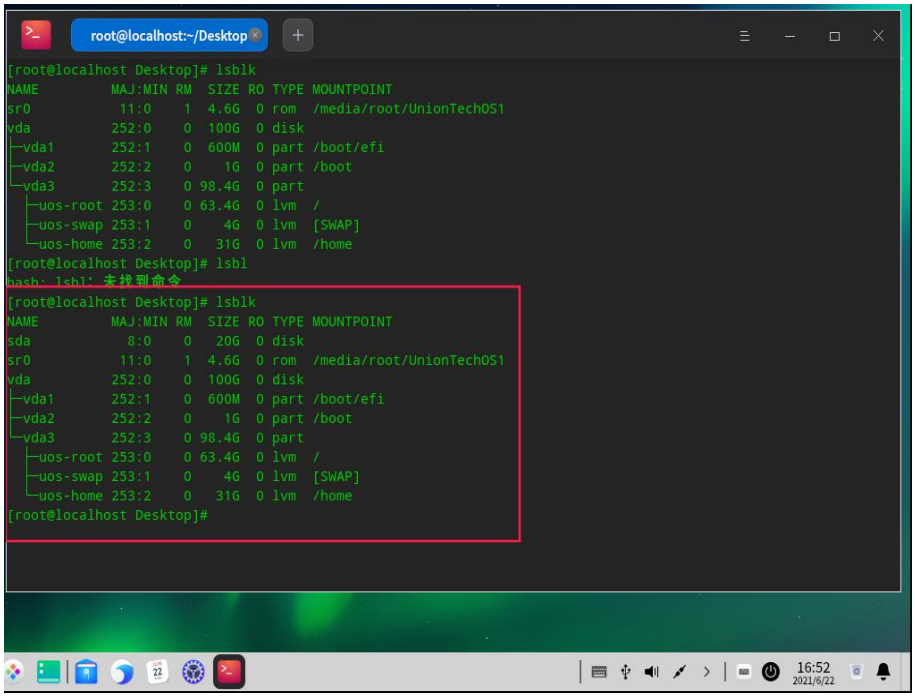


图 8.19 扩容后磁盘信息

网卡热插拔

执行网卡热插拔的前提条件是虚拟机需正常运行。

1 执行如下命令，创建网卡扩容 XML 文件。

```
# vi vnic.xml
<interface type="bridge">
```

```
    <mac address="22:49:27:44:3F:16" />
    <source bridge="br0" />
    <model type="virtio" />
</interface>
```

2 执行如下命令，对网卡进行扩容。

```
virsh attach-device uos_test vnic.xml
```

3 登录虚拟机后，执行如下命令，查看网卡信息。

```
ifconfig -a
```

■ 网卡扩容前

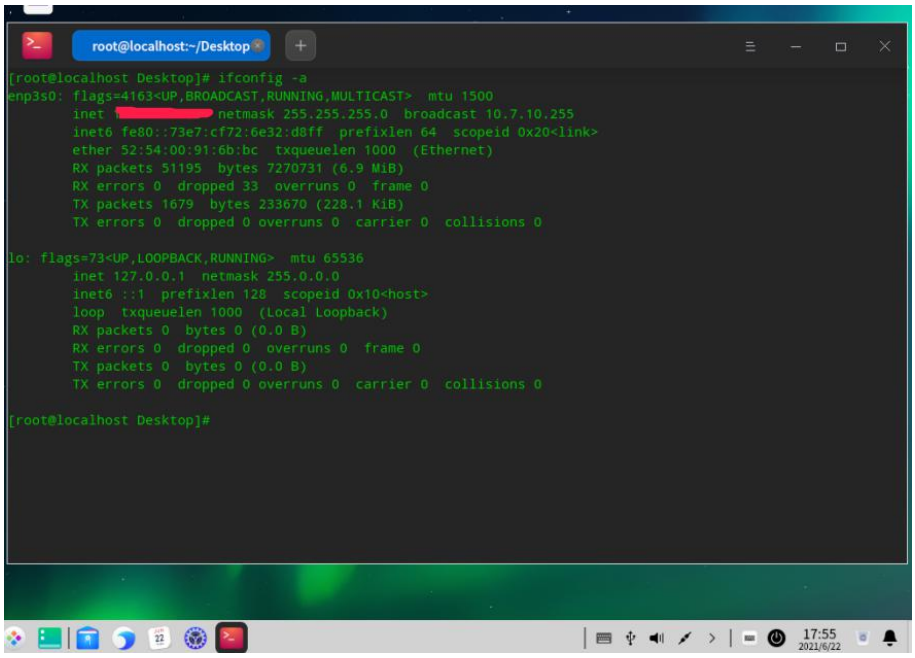


图 8.20 网卡扩容前信息

■ 网卡扩容后

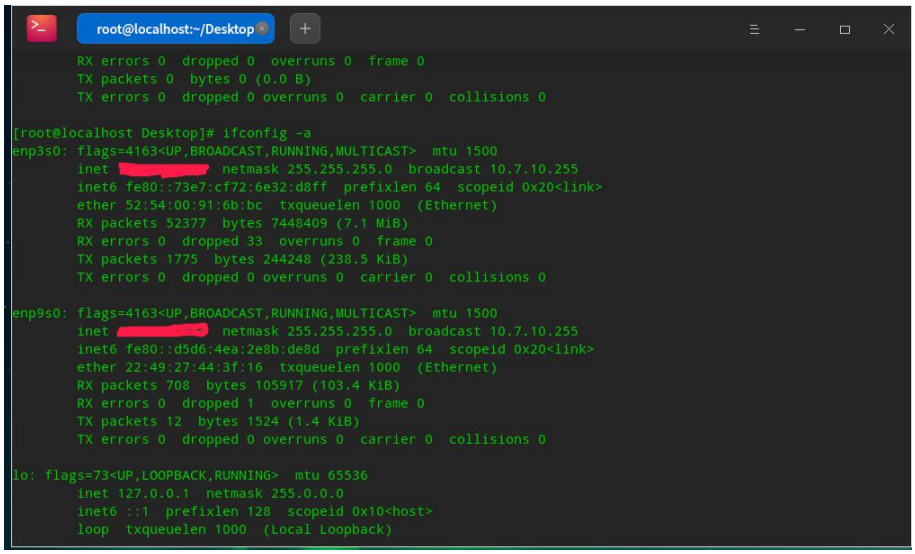


图 8.21 网卡扩容后信息

Numa 配置

1 在虚拟机模板文件 uos.xml 中，增加如下图所示的红框中的参数。

```
<vcpu placement='static' cpuset='0-3,64-67'>8</vcpu>
<features>
    <acpi/>
    <gic version='3'/>
</features>
<cputune>
    <vcpupin vcpu='0' cpuset='0-3'/>
    <vcpupin vcpu='1' cpuset='0-3'/>
    <vcpupin vcpu='2' cpuset='0-3'/>
    <vcpupin vcpu='3' cpuset='0-3'/>
    <vcpupin vcpu='4' cpuset='64-67'/>
    <vcpupin vcpu='5' cpuset='64-67'/>
    <vcpupin vcpu='6' cpuset='64-67'/>
    <vcpupin vcpu='7' cpuset='64-67'/>
</cputune>
<numatune>
    <memnode cellid='0' mode='strict' nodeset='0'/>
    <memnode cellid='1' mode='strict' nodeset='1'/>
</numatune>
<cpu mode='host-passthrough' check='none'>
```

```
        <topology sockets='2' cores='4' threads='1'/>
        <numa>
            <cell id='0' cpus='0-3' memory='4' unit='GiB'/>
            <cell id='1' cpus='4-7' memory='4' unit='GiB'/>
        </numa>
</cpu>
```

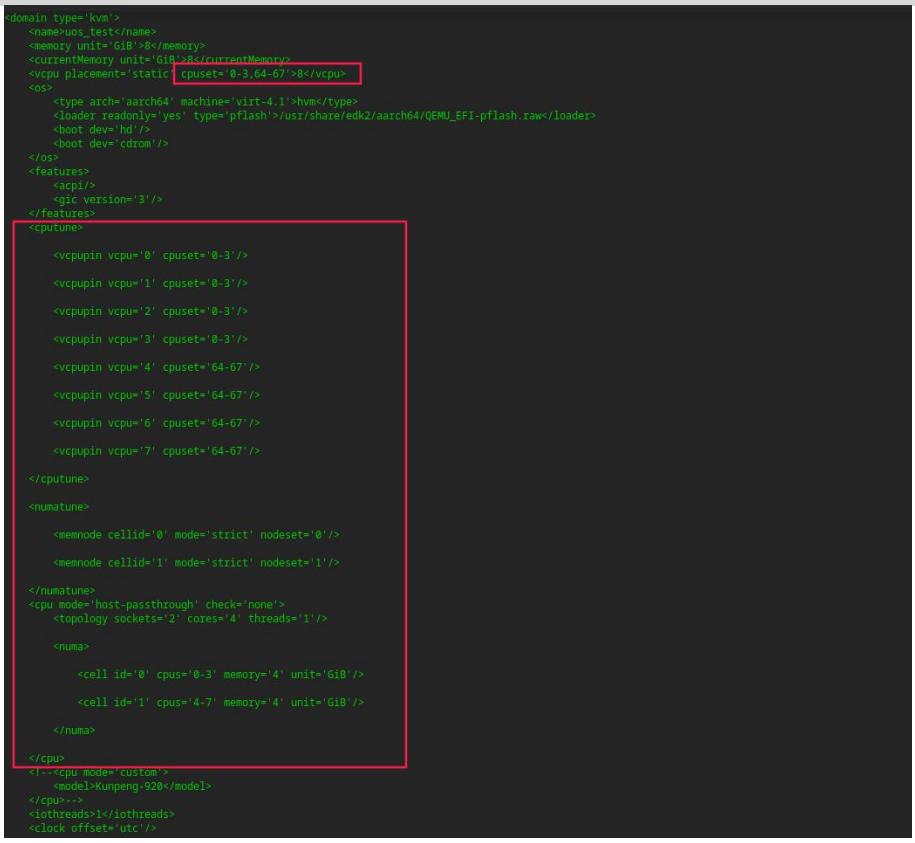


图 8.22 虚拟机 numa 配置

2 执行如下命令，启动虚拟机。

```
virsh create uos.xml
```

3 登录虚拟机后，执行如下命令，查看 numa 绑定结果。

```
lscpu
```

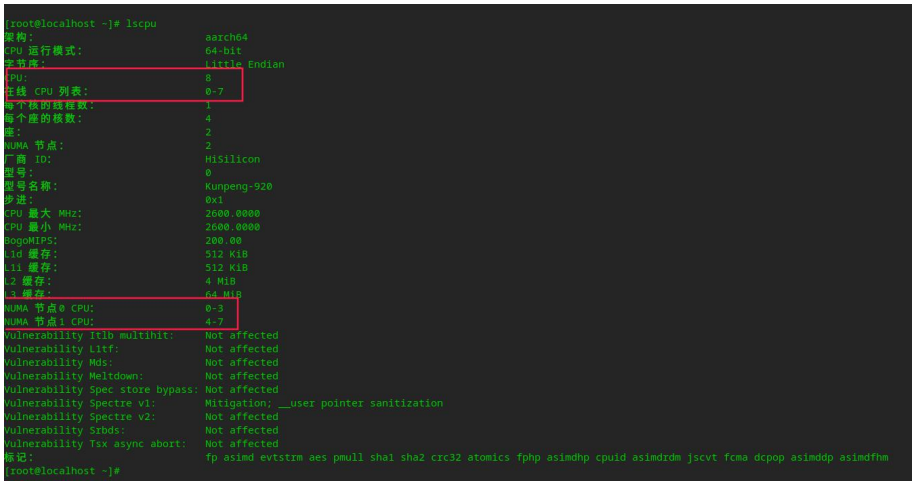


图 8.23 虚拟机 numa 绑定结果

8.3 KVM+Virt-Manager+QEMU 虚拟化

8.3.1 前提条件

- 宿主机必须为物理机。
- 物理网口上的 IP 地址采用 DHCP 方式获取。
- 本示例采用桥接网络，将虚拟机网卡桥接到 br0 上面。

8.3.2 安装虚拟化组件

请参见“8.2.2 安装虚拟化组件”章节。

8.3.3 上传镜像

1 执行如下命令，创建镜像目录。

mkdir -p */uos_images*

2 执行如下命令，上传本地镜像到镜像目录。

scp *uos-server-20*.iso root@IP:/uos_images/*

示例：**scp** *uos-server-20*.iso root@192.168.1.200:/uos_images/*

说明：IP 为服务器 IP 地址。

8.3.4 添加连接

Virt-Manager 图形化界面支持对多个 Host 主机上的虚拟机进行统一管理。如果用户在多个 Host 上安装了 KVM 虚拟化 那么可以在本地通过 Virt-Manager 进行管理，具体步骤如下。

1 执行如下命令，创建密钥。（无须输入密码，一直按“**Enter**”键即可。）

```
[root@localhost ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

```
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ydhzsfKcVujHaEORVOvrg1YxLim1unlx2Z0dPWnqwM8
root@localhost.localdomain
The key's randomart image is:
+---[RSA 3072]----+
|                 ...       |
|                . . .      |
|                 + .      o|
|            + ..*o    =.|
|           . S.*+o=o.=|
|              .O=B=o..o|
|               oX*O        |
|              .+++.E       |
|              o+    ..     |
+----[SHA256]-----+
[root@localhost ~]#
```

2 执行如下命令，拷贝公钥到其他 Host 主机上。（需要输入 root 用户的密码。）

```
[root@localhost ~]# ssh-copy-id root@192.168.1.200
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/root/.ssh/id_rsa.pub"
The authenticity of host '192.1368.1.200 (192.168.1.200)' can't be
established.
ECDSA key fingerprint is
SHA256:61oVHxlUxTTmo3m0Seld3DfZOdCMW7bY9wE3FGvYmss.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
```

```
prompted now it is to install the new keys
UOS Server 20 1060e
root@192.168.1.200's password:

Number of key(s) added: 1

Now try logging into the machine, with:    "ssh 'root@192.168.1.200'"
and check to make sure that only the key(s) you wanted were added.
```

3 打开 virt-manager，在主菜单中，选择“**文件** > **添加连接**”。

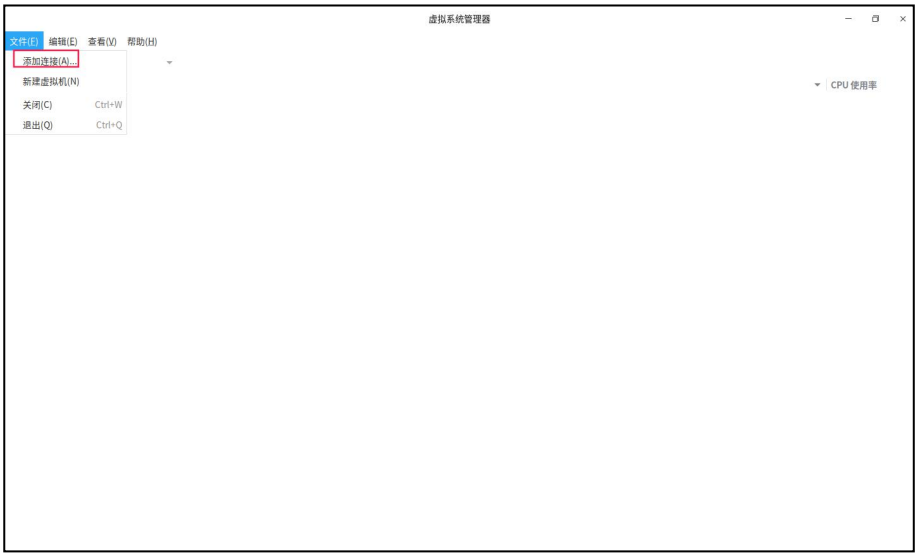


图 8.24 virt-manager 中连接物理主机

4 勾选“**Connect to remote host over SSH**”，并在“**用户名**”文本框中输入“**root**”、在“**主机名**”文本框中输入主机的 IP 地址后，单击“**连接**”。

图 8.25 配置物理主机详细信息

5 单击新创建的连接名，可以看到该主机上有很多虚拟机。

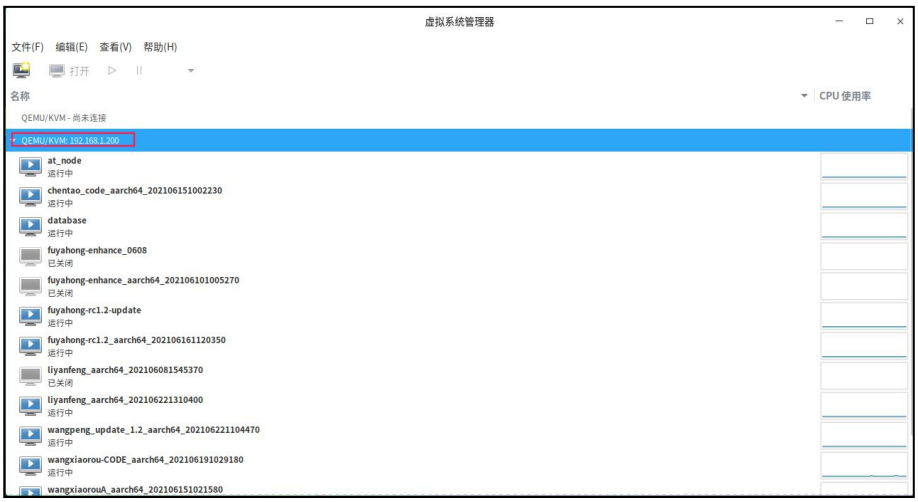


图 8.26 已连接的物理主机

6 单击任一虚拟机名，可以进入该虚拟机的登录界面。

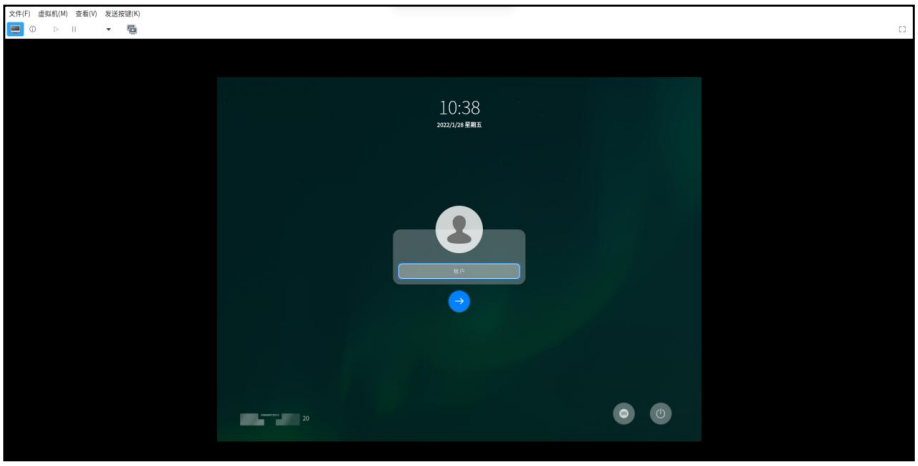


图 8.27 virt-manager 连接虚拟机示例

8.3.5 （可选）创建虚拟网络

1 在主菜单中，选择“**编辑** > **连接详情**”。

图 8.28 选择连接详情菜单

2 在“**虚拟网络**”页签中，单击“**+**”。

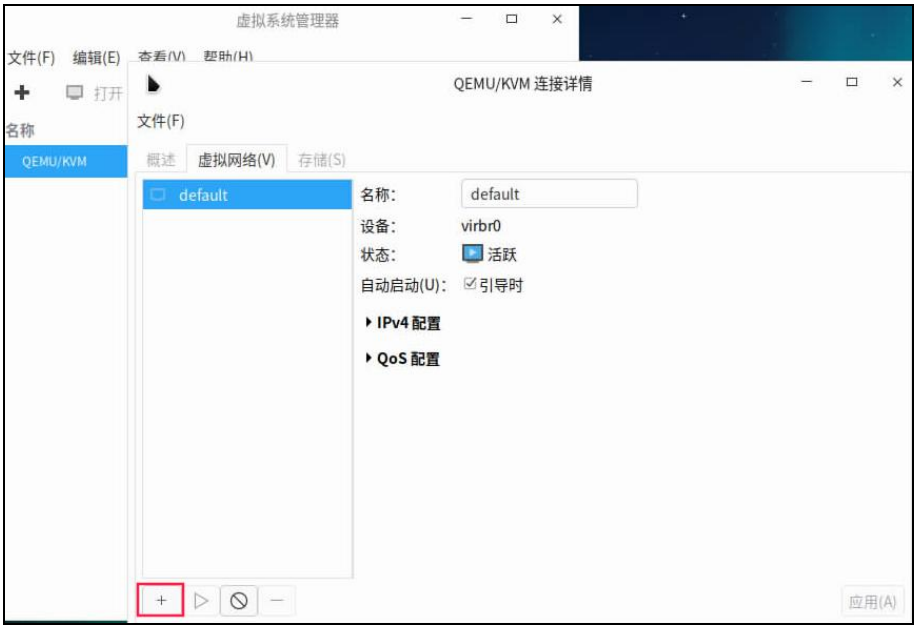


图 8.29 添加虚拟网络

3 设置“**网络名称**”后，单击“**前进**”。

图 8.30 为虚拟网络命名

4 设置 IP 地址范围后，单击“**前进**”。

图 8.31 为虚拟网络配置地址空间

图 8.32 为虚拟网络配置 IPv6 地址空间

5 完成配置后，出现如下图所示的界面。

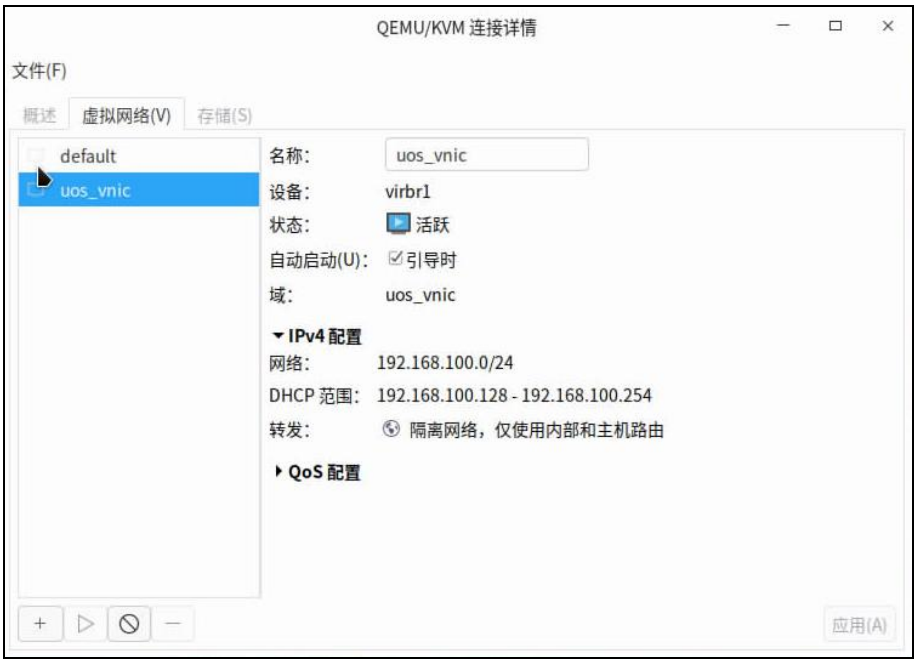


图 8.33 配置完成的虚拟网络

8.3.6 创建镜像存储文件

1 在主菜单中，选择“**编辑** > **连接详情**”。

图 8.34 选择连接详情菜单

2 在“**存储**”页签中，单击“**+**”。

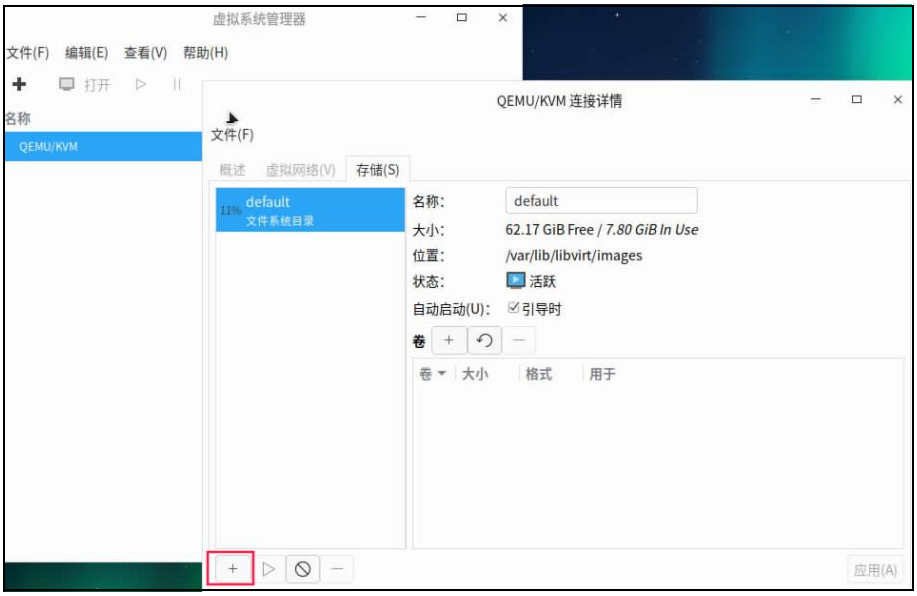


图 8.35 新建连接详情

3 设置存储池的名称后，单击“**前进**”。

图 8.36 输入存储池命令

4 选择存储池的目标路径后，单击“**完成**”。

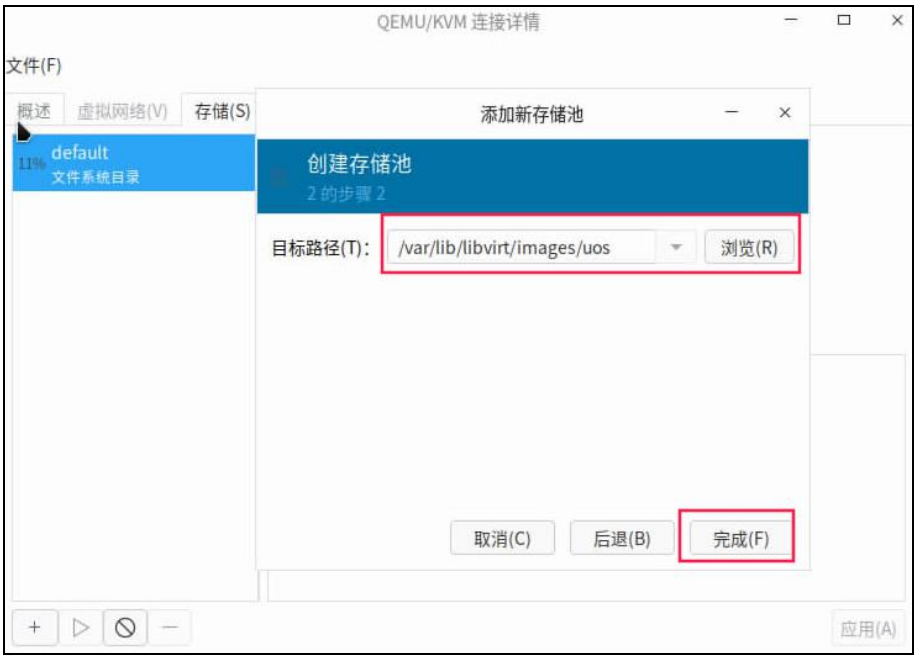


图 8.37 完成存储配置

5 单击已创建的存储池名称后，单击右侧中的“**+**”。

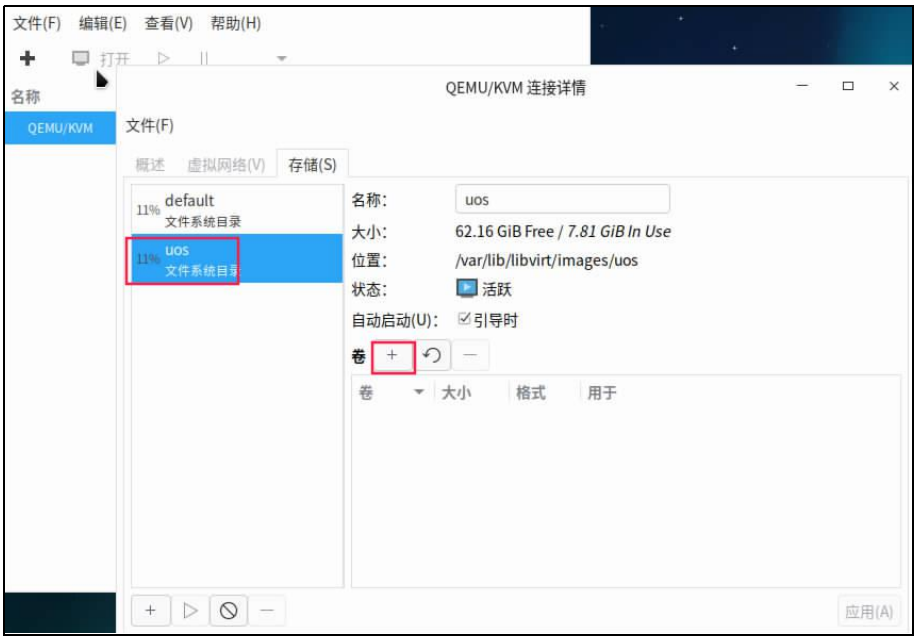


图 8.38 为存储池创建镜像存储文件

6 设置镜像文件的各项参数后，单击“**完成**”。

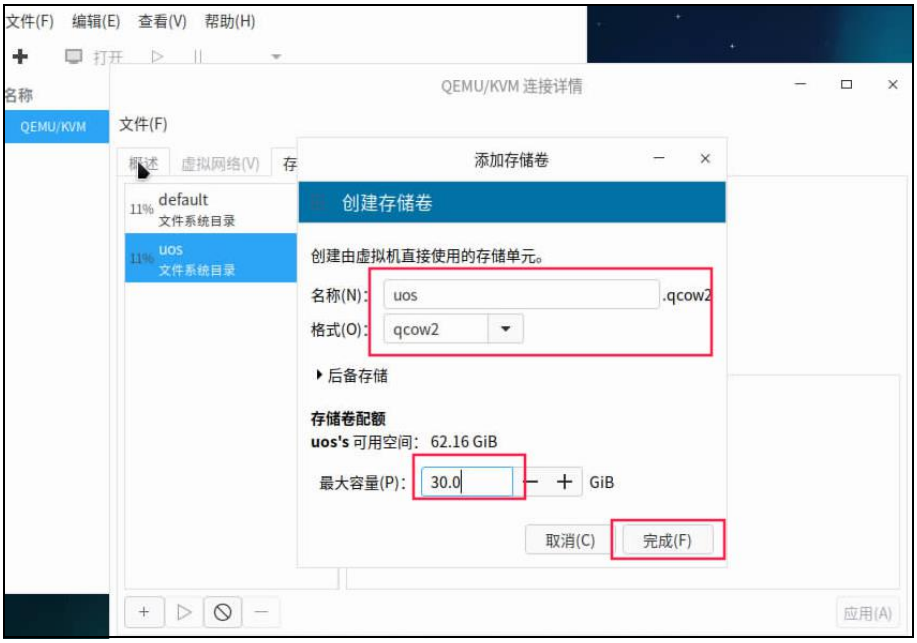


图 8.39 设置镜像文件参数

7 创建成功后，出现如下图所示的界面。

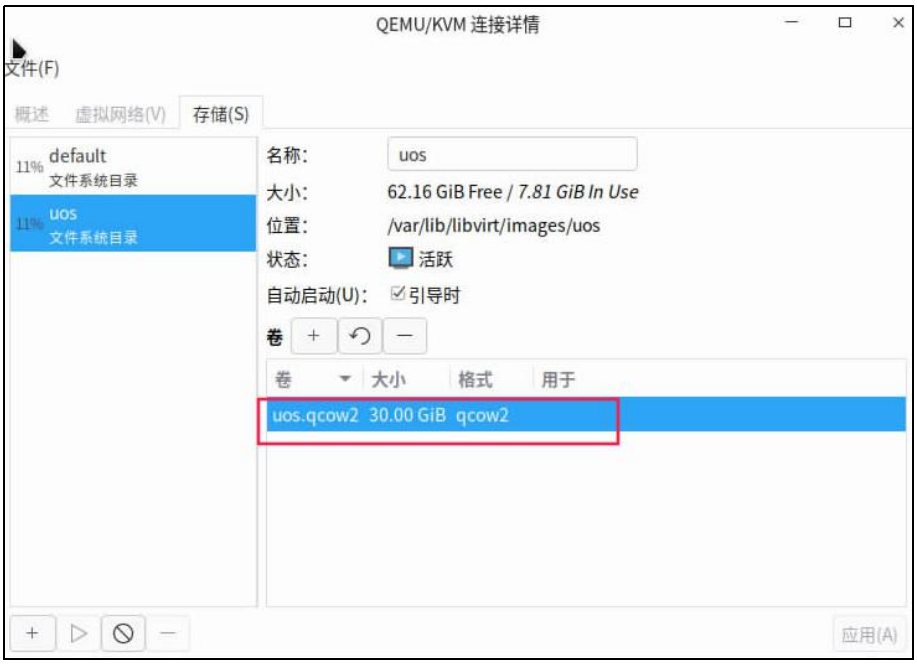


图 8.40 创建完成镜像存储文件

8.3.7 创建虚拟机

1 在主菜单中，选择“**文件** > **新建虚拟机**”。

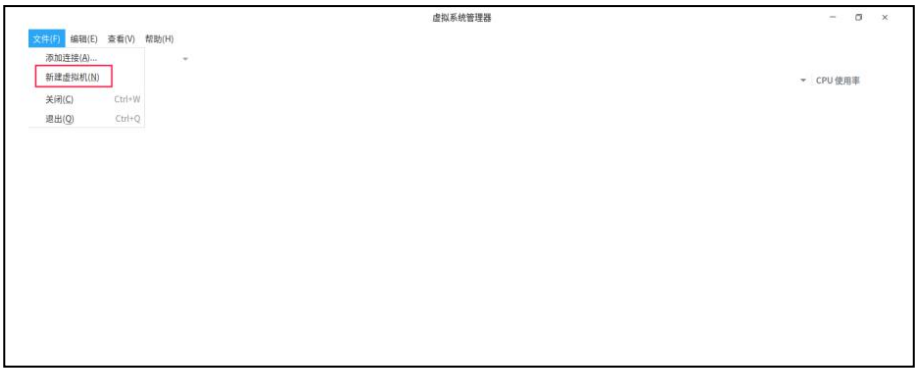


图 8.41 选择新建虚拟机菜单

2 选择“**本地安装介质(ISO 映像或者光驱)**并选择架构的相关信息后，单击“**前进**”。

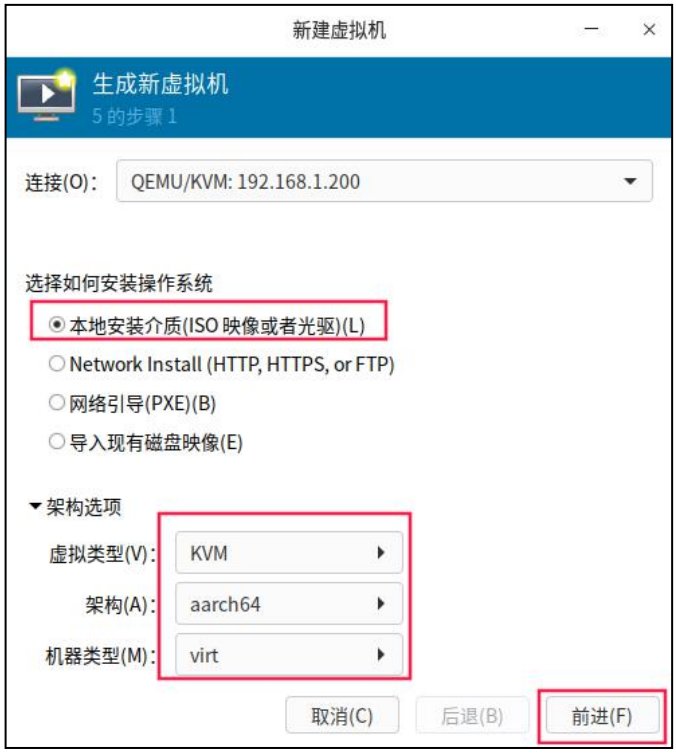


图 8.42 选择本地安装介质

*说明：如果有 kvm 或者 qemu 已经制作好的 vm 镜像，请选择“**导入现有磁盘映像**”。*

3 选择要安装的 ISO 镜像文件和操作系统类型（本示例选择默认值）后，单击“**前进**”。

图 8.43 准备选择本地 ISO 文件

图 8.44 选择本地 ISO 文件

图 8.45 选择要安装的操作系统类型

图 8.46 完成 ISO 镜像选择

4 设置 CPU 和内存后，单击“**前进**”。

图 8.47 设置虚拟机内存大小

5 创建或者选择镜像存储文件后，单击“**前进**”。

图 8.48 选择已创建的存储

6 设置虚拟机名称和网络，并在勾选“**在安装前自定义配置**”后，单击“**完成**”。

图 8.49 设置虚拟机名称，配置网络

7 添加键盘、图形、视频、数位板硬件。

（1）添加键盘：类型选择“**通用 USB Keyboard**”后，单击“**完成**”。

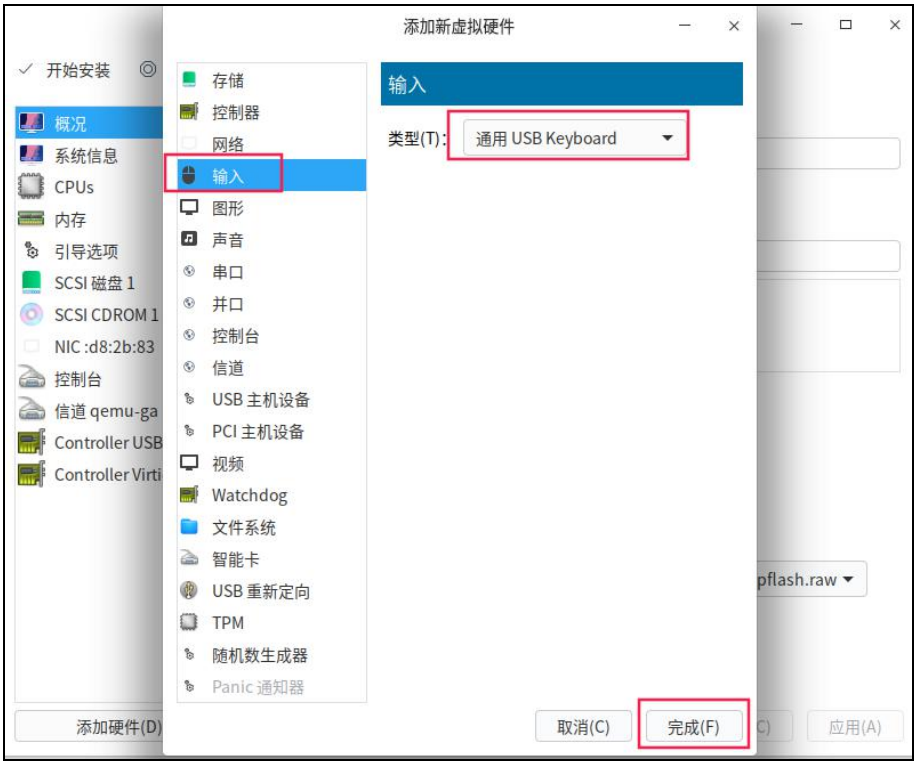


图 8.50 选择键盘类型

（2）添加图形：类型选择“**VNC 服务器**”后，单击“**完成**”。

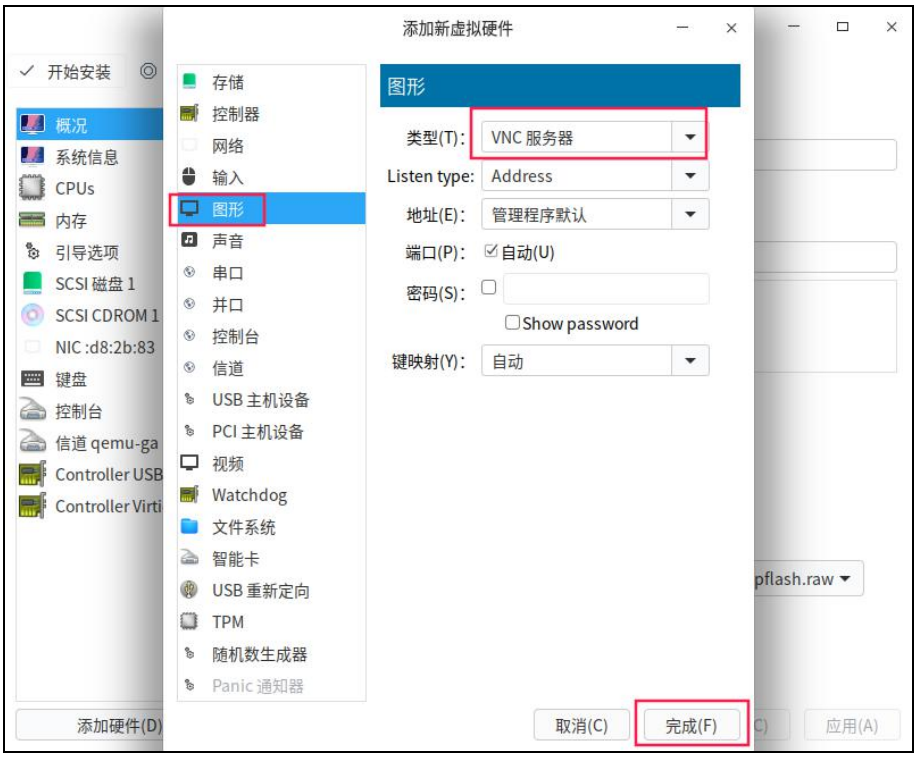


图 8.51 选择图形化显示类型

（3）添加视频：型号选择“**Virto**”后，单击“**完成**”。

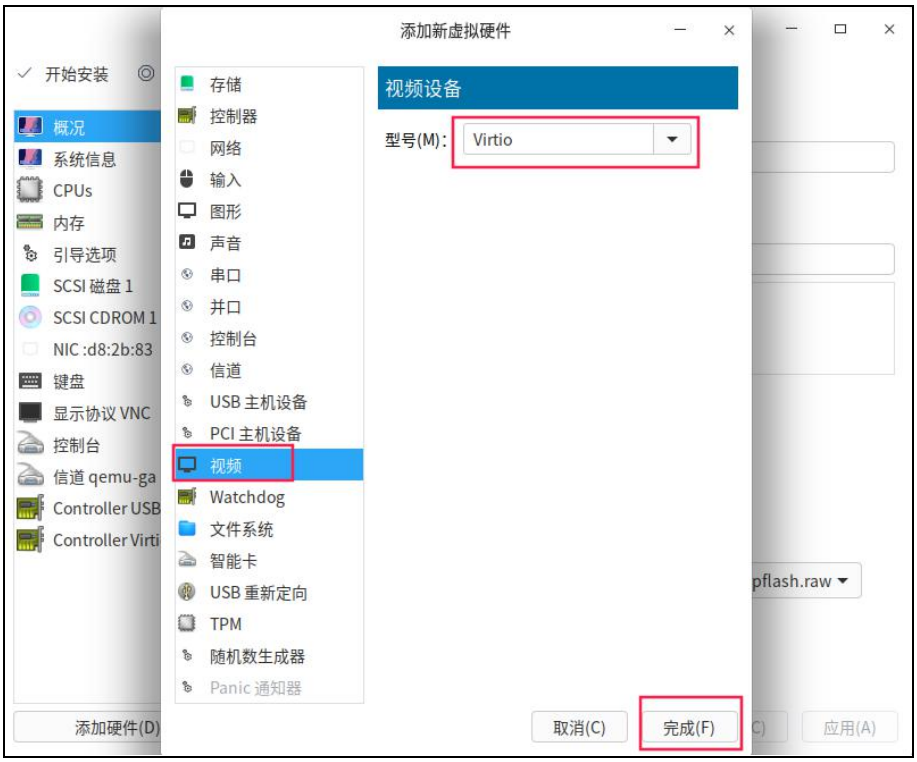


图 8.52 添加视频显示设备

（4）添加数位板：类型选择“**EvTouch USB 图形数位板**”后，单击“**完成**”。

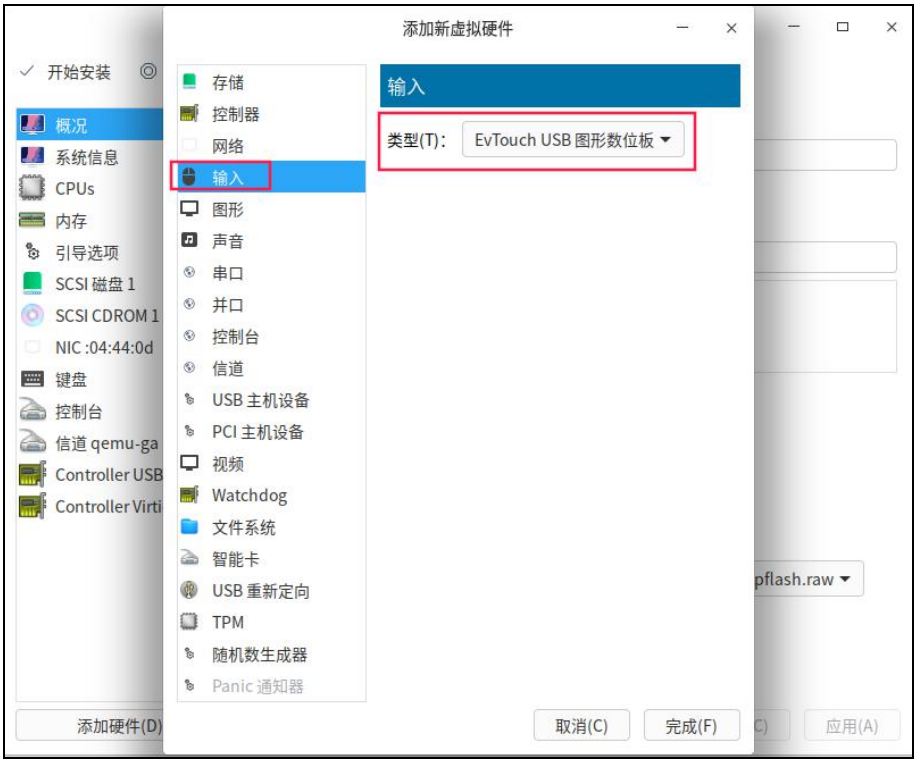


图 8.53 添加鼠标设备

（5）单击 USB 转发器的名称后，单击“**删除**”，删除 USB 转发器。（在 AMD64 架构中需要执行此步骤。）

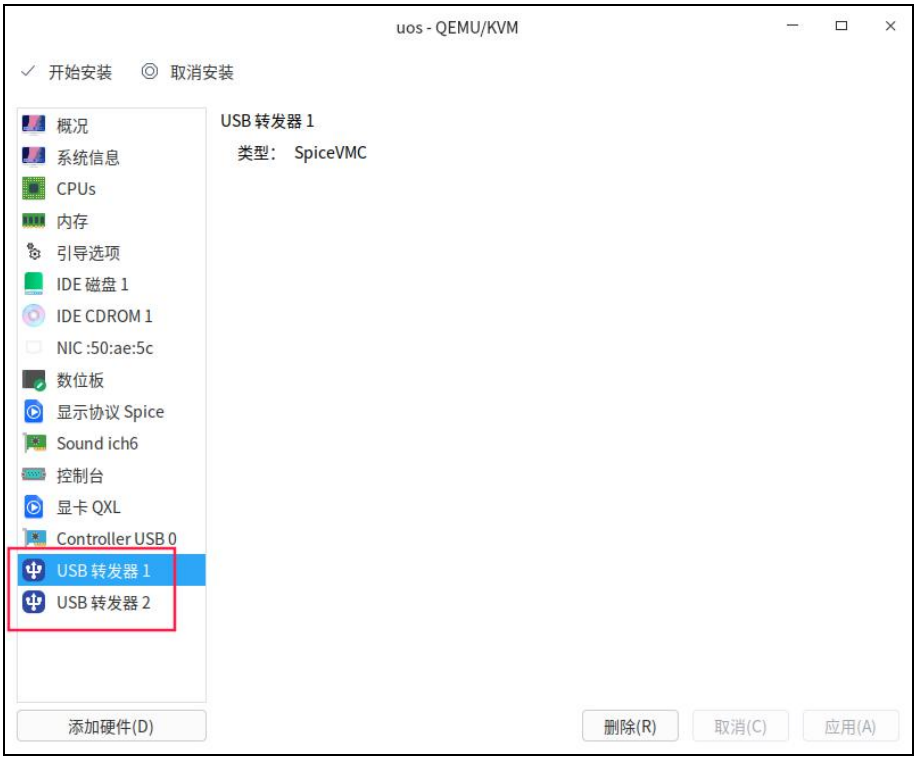


图 8.54 在 AMD64 架构中删除 USB 转发器

8 单击“**开始安装**”，虚拟机会自动重启，进入安装引导界面。

图 8.55 开始安装虚拟机

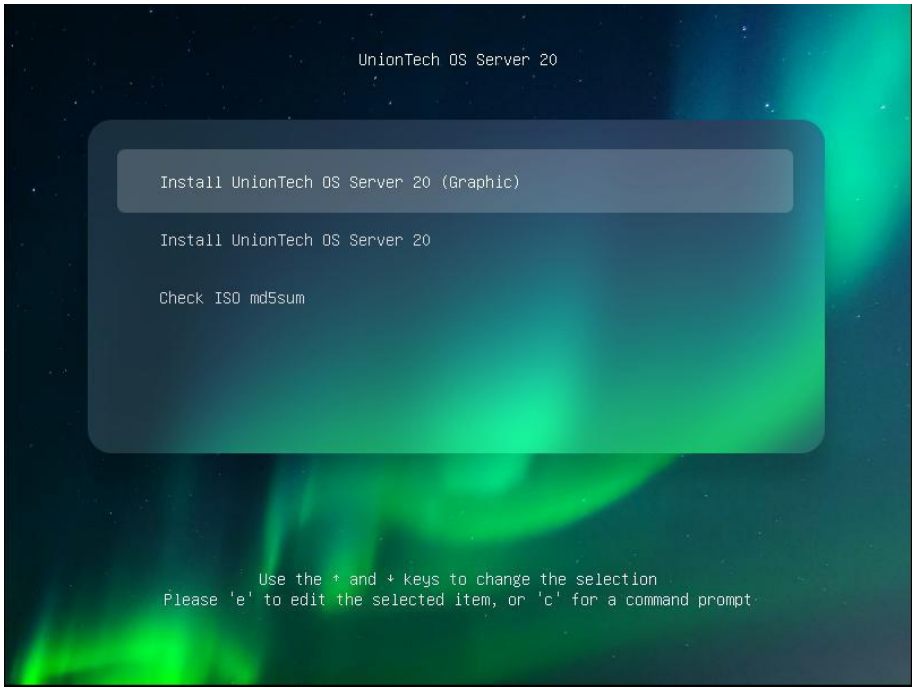


图 8.56 安装引导界面示例

9 操作系统安装界面加载成功后，请参见《统信服务器操作系统 V20（1060）安装手册》文档进行系统安装。

9 启用额外软件源

已安装的统信服务器操作系统 V20（1060e），默认配置了常用的 dnf 软件源。为了实现软件生态扩充、提高系统安全性等，还适配了 CDH、OpenStack

和国密等相关的软件包。可以通过使用 dnf 安装特定软件包的方式，启用额外的 dnf 软件源。目前，额外软件源包含以下三个：

- OpenStack 源
- CDH 源
- 国密源

说明：OpenStack 和 CDH 相关软件包目前仅适配了 1050e 及其 1050e 之后的版本，1xxxa 暂不支持。

9.1 启用 OpenStack 源（1060e）

OpenStack 是一个开源的云计算管理平台，为私有云和公有云提供可扩展的弹性云计算服务。统信服务器操作系统 V20（1060e）适配了 OpenStack（Train）。

9.1.1 安装 OpenStack 源

dnf install *UnionTech-repos-openstack* -y

9.1.2 刷新元数据缓存

```
dnf clean all
dnf makecache
```

9.2 启用 CDH 源（1060e）

CDH 大数据平台是 Hadoop 众多分支中的一种，由 Cloudera 维护。它基于稳定版本的 Apache Hadoop 构建，提供了 Hadoop 的核心、可扩展存储、分布式计算、基于 web 的用户界面等。统信服务器操作系统 V20（1060e）适配了 CDH-6.3.2 的核心软件。

9.2.1 安装 CDH 源

dnf install *UnionTech-repos-cdh6* -y

9.2.2 刷新元数据缓存

```
dnf clean all
dnf makecache
```

9.3 启用国密源

伴随着《中华人民共和国密码法》、《中华人民共和国网络安全法》和《中华人民共和国数据安全法》等法律法规的颁布实施，密码应用新需求不断被提出，国密算法在基础软、硬件中的应用直接关系到国家网络空间安全。为此，统信服务器操作系统 V20（1060）完成部分开源软件国密改造，以提升系统安全性、完善统信 UOS 全链条国密栈。

9.3.1 安装 GM 源

```
dnf install UnionTech-repos-GM -y
```

9.3.2 刷新元数据缓存

```
dnf clean all
dnf makecache
```

10 FAQ

10.1 输入法常见问题

统信服务器操作系统默认会安装 fcitx 输入法 ,如果您同时也安装了 ibus 输入法，则会导致两种输入法冲突。此时，需要执行如下步骤切换输入法。

1 执行如下命令，查看当前已安装的输入法。

```
imsettings-list
- 1: IBus[ibus.conf] (recommended)
  2: FCITX[fcitx.conf]
  3: X compose table[xcompose.conf]
```

2 执行如下命令，切换 ibus 或者 fcitx 输入法。

```
imsettings-switch xxx
```

⚠*注意：xxx 代表 ibus 或者 fcitx。*

3 注销用户后，重新登录桌面环境，输入法切换即可生效。

10.2 虚拟机 XML 文件

10.2.1 AMD 架构的虚拟机 XML 文件（Legacy）

```
<domain type='kvm'>
  <name>uos</name>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <iothreads>1</iothreads>
  <os>
    <type arch='x86_64' machine='q35'>hvm</type>
  </os>
  <features>
    <acpi/>
  </features>
```

```
  <cpu mode='host-passthrough' />
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2' iothread='1'/>
      <source file='/uos/uos.qcow2'/>
      <target dev='vda' bus='virtio'/>
      <boot order='1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08'
function='0x0'/>
    </disk>
    <disk type='file' device='cdrom'>
    <driver name='qemu' type='raw'/>
    <source file='/uos/uos-server-20-1060e-amd64.iso'/>
    <readonly/>
    <target dev='sdb' bus='scsi'/>
    <boot order='2'/>
</disk>
    <controller type='scsi' index='0' model='virtio-scsi'>
    </controller>
    <controller type='virtio-serial' index='0'>
    </controller>
    <controller type='usb' index='0' model='ehci'>
    </controller>
    <controller type='sata' index='0'>
    </controller>
    <controller type='pci' index='0' model='pci-root'/>
    <interface type='bridge'>
      <source bridge='br0'/>
      <model type='virtio'/>
```

```
    </interface>
    <serial type='pty'>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <input type='tablet' bus='usb'>
      <address type='usb' bus='0' port='1'/>
    </input>
    <input type='keyboard' bus='usb'>
      <address type='usb' bus='0' port='2'/>
    </input>
    <input type='mouse' bus='ps2'/>
    <input type='keyboard' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'>
      <listen type='address' address='0.0.0.0'/>
    </graphics>
    <video>
      <model type='vga' vram='16384' heads='1' primary='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
function='0x0'/>
    </video>
    <memballoon model='virtio'>
    </memballoon>
  </devices>
</domain>
```

10.2.2 AMD 架构的虚拟机 XML 文件（UEFI）

```
<domain type='kvm'>
  <name>uos</name>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
```

```
  <vcpu placement='static'>1</vcpu>
  <iothreads>1</iothreads>
  <os>
    <type arch='x86_64' machine='q35'>hvm</type>
     <loader type='rom'>/usr/share/edk2/ovmf/OVMF.fd</loader>
  </os>
  <features>
    <acpi/>
  </features>
  <cpu mode='host-passthrough' />
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2' iothread='1'/>
      <source file='/uos/uos.qcow2'/>
      <target dev='vda' bus='virtio'/>
      <boot order='1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08'
function='0x0'/>
    </disk>
     <disk type='file' device='cdrom'>
          <driver name='qemu' type='raw'/>
          <source file='/uos/uos-server-20-1060e-amd64.iso'/>
          <readonly/>
          <target dev='sdb' bus='scsi'/>
          <boot order='2'/>
     </disk>
    <controller type='scsi' index='0' model='virtio-scsi'>
    </controller>
    <controller type='virtio-serial' index='0'>
```

```
    </controller>
    <controller type='usb' index='0' model='ehci'>
    </controller>
    <controller type='sata' index='0'>
    </controller>
    <controller type='pci' index='0' model='pci-root'/>
    <interface type='bridge'>
      <source bridge='br0'/>
      <model type='virtio'/>
    </interface>
    <serial type='pty'>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <input type='tablet' bus='usb'>
      <address type='usb' bus='0' port='1'/>
    </input>
    <input type='keyboard' bus='usb'>
      <address type='usb' bus='0' port='2'/>
    </input>
    <input type='mouse' bus='ps2'/>
    <input type='keyboard' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'>
      <listen type='address' address='0.0.0.0'/>
    </graphics>
    <video>
      <model type='vga' vram='16384' heads='1' primary='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
function='0x0'/>
    </video>
```

```
    <memballoon model='virtio'>
    </memballoon>
  </devices>
</domain>
```

10.2.3 ARM 架构的虚拟机 XML 文件（UEFI）

```
<domain type='kvm'>
    <name>uos</name>
    <memory unit='GiB'>8</memory>
    <currentMemory unit='GiB'>8</currentMemory>
    <vcpu placement='static'>6</vcpu>
    <os>
        <type arch='aarch64' machine='virt-4.1'>hvm</type>
        <loader readonly='yes'
type='pflash'>/usr/share/edk2/aarch64/QEMU_EFI-pflash.raw</loader>
        <boot dev='hd'/>
        <boot dev='cdrom'/>
    </os>
    <features>
        <acpi/>
        <gic version='3'/>
    </features>
    <cpu mode='host-passthrough'/>
    <iothreads>1</iothreads>
    <clock offset='utc'/>
    <on_poweroff>destroy</on_poweroff>
    <on_reboot>restart</on_reboot>
    <on_crash>restart</on_crash>
    <devices>
        <channel type='unix'>
            <target type='virtio' name='org.qemu.guest_agent.0'/>
            <address type='virtio-serial' controller='0' bus='0' port='1'/>
        </channel>
        <memballoon model='virtio'/>
        <emulator>/usr/libexec/qemu-kvm</emulator>
```

```
        <disk type='file' device='disk'>
            <driver name='qemu' type='qcow2' iothread="1"/>
            <source file='/uos/uos.qcow2'/>
            <target dev='vda' bus='virtio'/>
        </disk>
        <disk type='file' device='cdrom'>
            <driver name='qemu' type='raw'/>
            <source file='/uos/uos-server-20-1060e-arm64.iso'/>
            <readonly/>
            <target dev='sdb' bus='scsi'/>
        </disk>
        <interface type='bridge'>
            <source bridge='br0'/>
            <model type='virtio'/>
            <address type='pci' domain='0x0000' bus='0x01' slot='0x00'
function='0x0'/>
        </interface>
        <console type='pty'/>
        <controller type='scsi' index='0' model='virtio-scsi'/>
        <controller type='usb' model='ehci'/>
        <input type='keyboard' bus='usb'/>
        <input type='tablet' bus='usb'/>
        <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'/>
    </devices>
</domain>
```

10.3 A-Tune 常见问题

10.3.1 train 命令训练模型出错，提示“training data faild”

可能原因

collection 命令只采集了一种类型的数据。

解决方法

至少采集两种数据类型的数据后，再进行训练。

10.3.2 atune-adm 无法连接 atuned 服务

可能原因

- atuned 服务未启动，或 atuned 侦听地址异常。您可执行如下命令，检查 atuned 服务是否启动，并检查 atuned 侦听地址。

```
# systemctl status atuned
# netstat -nap | atuned
```

- 防火墙阻止了 atuned 的侦听端口。
- 系统配置了 http 代理，导致无法连接。

解决方法

1 如果 atuned 服务未启动，执行如下命令，启动该服务。

```
# systemctl start atuned
```

2 分别在 atuned 和 atune-adm 的服务器上，执行如下命令，允许侦听端口接收网络包。其中，60001 为 atuned 的侦听端口号。

```
# iptables -I INPUT -p tcp --dport 60001 -j ACCEPT
# iptables -I INPUT -p tcp --sport 60001 -j ACCEPT
```

3 在不影响业务的前提下，执行如下命令，删除 http 代理，或对侦听的 IP 地址不进行 http 代理。

```
# no_proxy=$no_proxy,侦听地址
```

10.3.3 atuned 服务无法启动，提示“Job for atuned.service failed because a timeout was exceeded.”

可能原因

hosts 文件中缺少 localhost 配置。

解决方法

在“**/etc/hosts**”文件中的 127.0.0.1 这一行添加 localhost。

```
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4
```

10.4 模块化安装常见问题

10.4.1 llvm-toolset 无法安装

问题现象

1 执行命令，安装 llvm-toolset。

```
dnf module install llvm-toolset:uel20/common
```

2 显示类似的如下所示的错误信息。

```
Error:
Problem: problem with installed package
mesa-vulkan-drivers-20.1.4-2.uel20.x86_64
  - package mesa-vulkan-drivers-20.1.4-2.uel20.x86_64 requires
libLLVM-10.so()(64bit), but none of the providers can be installed
  - package mesa-vulkan-drivers-20.1.4-2.uel20.x86_64 requires
libLLVM-10.so(LLVM_10)(64bit), but none of the providers can be
installed
  - cannot install both
llvm-libs-11.0.0-2.up1.module_uel20+13+1b98fcdf.x86_64 and
llvm-libs-10.0.1-2.uel20.x86_64
  - package clang-11.0.0-1.02.module_uel20+12+1b98fcdf.x86_64
requires libLLVM-11.so()(64bit), but none of the providers can be
installed
  - package clang-11.0.0-1.02.module_uel20+12+1b98fcdf.x86_64
requires libLLVM-11.so(LLVM_11)(64bit), but none of the providers can
be installed
  - package llvm-toolset-11.0.0-1.module_uel20+7+1b98fcdf.x86_64
requires clang = 11.0.0, but none of the providers can be installed
  - conflicting requests
  - package llvm-libs-10.0.1-2.uel20.x86_64 is filtered out by modular
filtering
(try to add &apos;--allowerasing&apos; to command line to replace
conflicting packages or &apos;--skip-broken&apos; to skip uninstallable
packages or &apos;--nobest&apos; to use not only best candidate
packages)
```

可能原因

操作系统预装的 mesa-vulkan-drivers 或者 mesa-dri-drivers 依赖了 llvm-libs-10，而 llvm-toolset 模块提供了 llvm-libs-11，这两个版本不能共存，并且 llvm-toolset 安装的 llvm-libs 不满足 mesa-vulkan-drivers 或者 mesa-dri-drivers 的安装依赖。

解决方法

使用 dnf 的--allowerasing 选项，允许删除冲突的软件，即可正常安装。即

执行如下命令，来安装 llvm-toolset。

dnf module install *llvm-toolset:uel20/common* --allowerasing

备注

卸载 llvm-toolset 模块的时候，需要回退被升级的 llvm-libs，这需要使基础镜像或在线源提供的 llvm-libs-10 可见。此时，请先禁用 llvm-toolset 模块，然后执行 **dnf history** 命令查询安装 llvm-toolset 模块的事务编号，最后执行“**dnf history undo**”命令来卸载 llvm-toolset。即应该执行如下命令：

dnf module disable *llvm-toolset*
dnf history # 查询 transno
dnf history undo *[transno]*

10.4.2 rust-toolset 无法安装

问题现象

1 执行命令，安装 rust-toolset。

dnf module install *rust-toolset:uel20/common*

2 显示类似的如下所示的错误信息。

```
Error:
 Problem: problem with installed package
mesa-vulkan-drivers-20.1.4-2.uel20.x86_64
  - package mesa-vulkan-drivers-20.1.4-2.uel20.x86_64 requires
libLLVM-10.so()(64bit), but none of the providers can be installed
  - package mesa-vulkan-drivers-20.1.4-2.uel20.x86_64 requires
libLLVM-10.so(LLVM_10)(64bit), but none of the providers can be
installed
  - cannot install both
llvm-libs-11.0.0-2.up1.module_uel20+13+1b98fcdf.x86_64 and
llvm-libs-10.0.1-2.uel20.x86_64
  - package rust-1.49.0-1.module_uel20+7+ed636893.x86_64 requires
libLLVM-11.so()(64bit), but none of the providers can be installed
  - package rust-1.49.0-1.module_uel20+7+ed636893.x86_64 requires
libLLVM-11.so(LLVM_11)(64bit), but none of the providers can be
installed
  - package rust-toolset-1.49.0-1.module_uel20+3+ed636893.x86_64
requires rust = 1.49.0, but none of the providers can be installed
```

```
 - conflicting requests
 - package llvm-libs-10.0.1-2.uel20.x86_64 is filtered out by modular
filtering
(try to add '--allowerasing' to command line to replace conflicting
packages or '--skip-broken' to skip uninstallable packages or '--nobest'
to use not only best candidate packages)
```

可能原因

rust-toolset 模块依赖 llvm-toolset 模块，报错信息实际是因需要安装 llvm-toolset 模块中的软件包引起的。

解决方法

使用 dnf 的--allowerasing 选项，允许删除冲突的软件，即可正常安装。即执行如下命令，来安装 rust-toolset。

```
dnf module install rust-toolset:uel20/common --allowerasing
```

备注

请先禁用 llvm-toolset 和 rust-toolset 模块，然后查找安装 rust-toolset 的事务编号，最后执行“**dnf history undo**”命令来卸载 rust-toolset。即应该执行如下命令：

```
dnf module disable llvm-toolset rust-toolset
dnf history # 查询 transno
dnf history undo [transno]
```

10.5 时间同步服务器使用问题

服务器操作系统的 DDE 桌面环境使用 systemd-timesyncd.service 作为时间同步服务。由于该服务与 chronyd.service 和 ntpd.service 存在冲突关系，如果您需要在安装有 DDE 桌面环境的服务器内使用 chronyd.service 或 ntpd.service，请提前在“**控制中心** > **时间日期** > **时间设置**”中关闭“**自动同步配置**”开关，避免对 chronyd.service 或 ntpd.service 造成影响。非 DDE 桌面环境不涉及此问题。

10.6 无密码帐户锁定问题

当使用无密码帐户登录，如果 DDE 登录界面遇到无法输入密码的问题，请按“**Ctrl+Alt+F2**”快捷键切换到 tty2，然后使用已设置密码的管理员帐户登录后，执行 **sudo systemctl restart** *lightdm.service* 命令即可。

10.7 重启 dbus 服务导致桌面系统无法使用问题

当重启 dbus 服务时，DDE 桌面可能会出现异常，无法继续使用。此时，请按“**Ctrl+Alt+F2**”快捷键切换到 tty2 后，执行 **sudo init** *5* 命令即可。

10.8 在飞腾环境中使用 vi 命令查看/var/log/messages 时，出现中文字符乱码问题

问题现象

采用飞腾 CPU 的“**擎天 EF860**”服务器，内核将主板信息打印到“**/var/log/messages**”文件中后，可能显示为“**\x93\x8e 天 EF860**”。这时，使用 vi(m)命令打开“**/var/log/messages**”文件时，vi(m)命令会按照默认的“**fileencodings**”列表依次尝试解析“**/var/log/messages**”文件中的字符编码。然而，受主板信息日志行异常编码影响，最终解析的文件编码将是“latin1”，此时 vi(m)窗口将无法正确显示 messages 文件中的中文字符。

解决方法

在用户主目录的“**.virc**”文件或系统的“**/etc/virc**”文件中追加一个文件编码集的配置，不要包含 latin1。然后使用 vi 命令查看“**/var/log/messages**”文件，即可正常显示中文字符。若使用的是 vim 命令，请修改“**.vimrc**”文件。

- 用户级配置：请执行如下命令。

```
echo "set fileencodings=ucs-bom,utf-8,default">> ${HOME}/.virc
```

- 全局配置：请执行如下命令。

```
sudo echo "set fileencodings=ucs-bom,utf-8,default">> /etc/virc
```

⚠*注意：修改文件编码集只能保证打开“**/var/log/messages**”后，可以正确解析部分进程打印的中文日志，但无法解析“\x93\x8e 天 EF860”这段异常编码。*

10.9 系统常见问题

10.9.1 firebird 用户登录 shell 问题

如果您安装的是 1021e 版本，则带有 DDE 桌面，系统默认会安装 firebird，同时也会创建 firebird 用户。该用户的默认登录 shell 为“**/bin/nologin**”。如果您的应用在运行过程中有使用到该用户，那么需要将该用户的登录 shell 修改为“**/sbin/nologin**”，可以执行如下命令使用 vi 编辑器进行修改。

```
# vi /etc/passwd
```

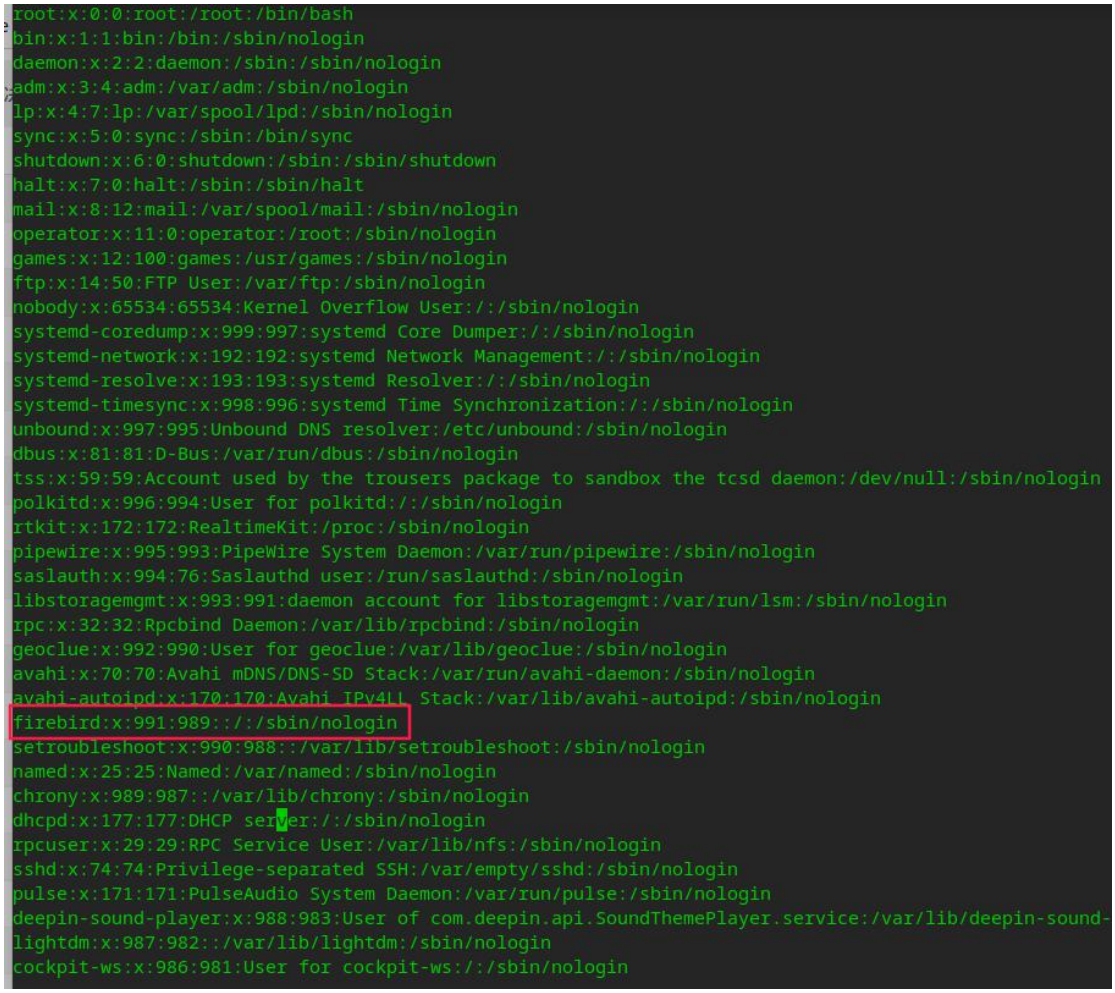


图 10.1 使用 vi 编辑器直接修改指定用户的登录 shell

10.9.2 如何正确设置 selinux 模式

1 执行如下命令，设置 selinux 为 permissive 模式，且默认 disabled。

```
# sed -i 's/SELINUX=disabled/SELINUX=permissive/g' /etc/selinux/config
```

2 执行如下命令，重启操作系统。

```
# reboot
```

3 完成重启后，执行如下命令，修改为 enforcing 模式。

```
# sed -i 's/SELINUX=permissive/SELINUX=enforcing/g' /etc/selinux/config
```

4 执行如下命令，再次重启操作系统。

```
# reboot
```

5 执行如下命令，检查 selinux 模式是否已经设置为 enforcing 模式。

```
# getenforce
```

10.10 perl 模块使能导致依赖 perl 的非模块化包安装失败问题

问题现象

启用模块化源并且使能了 perl 模块后，安装 git 等直接或者间接依赖 perl 的包时提示此包已被模块化过滤掉。

```
[root@localhost ~]# dnf install git
Last metadata expiration check: 0:00:09 ago on 2022年03月30日 星期三 16时45分26秒.
Error:
 Problem: package git-2.27.0-6.uel20.x86_64 requires perl(Term::ReadKey), but none of the providers can be instal
led
  - cannot install the best candidate for the job
  - package perl-TermReadKey-2.38-2.uel20.x86_64 is filtered out by modular filtering
  - package perl-TermReadKey-2.38-2.module_uel20+41+adb00439.x86_64 is filtered out by modular filtering
(try to add '--skip-broken' to skip uninstallable packages or '--nobest' to use not only best candidate packages)
[root@localhost ~]#
```

图 10.2 perl 模块过滤了默认软件包，导致依赖解析问题

解决方法

先执行 **dnf module disable** 命令禁用掉 perl 相关的模块化流（这些流直接导致将要安装的包被过滤掉了），并在安装时使用--allowerasing 选项确保安装后依赖的是正确的 perl 版本。

```
[root@localhost ~]# dnf install git --allowerasing
Last metadata expiration check: 0:20:26 ago on 2022年03月30日 星期三 16时51分25秒.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                          Repository          Size
================================================================================
Installing:
 git                     x86_64   2.27.0-6.uel20                   modular            5.9 M
Upgrading:
 perl                    x86_64   4:5.28.3-7.up2.uel20             UnionTechOS-Server-20   3.0 M
     replacing  perl-Attribute-Handlers.noarch 0.99-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Errno.x86_64 1.28-419.04.up1.module_uel20+109+57f75040
     replacing  perl-ExtUtils-Embed.noarch 1.34-419.04.up1.module_uel20+109+57f75040
     replacing  perl-ExtUtils-Miniperl.noarch 1.06-419.04.up1.module_uel20+109+57f75040
     replacing  perl-IO.x86_64 1.38-419.04.up1.module_uel20+109+57f75040
     replacing  perl-IO-Zlib.noarch 1:1.10-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Locale-Maketext-Simple.noarch 1:0.21-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Math-Complex.noarch 1.59-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Memoize.noarch 1.03-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Module-Loaded.noarch 1:0.08-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Net-Ping.noarch 2.55-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Pod-Html.noarch 1.22.02-419.04.up1.module_uel20+109+57f75040
     replacing  perl-SelfLoader.noarch 1.23-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Test.noarch 1.30-419.04.up1.module_uel20+109+57f75040
     replacing  perl-Time-Piece.x86_64 1.31-419.04.up1.module_uel20+109+57f75040
     replacing  perl-interpreter.x86_64 4:5.26.3-419.04.up1.module_uel20+109+57f75040
     replacing  perl-libnetcfg.noarch 4:5.26.3-419.04.up1.module_uel20+109+57f75040
     replacing  perl-open.noarch 1.11-419.04.up1.module_uel20+109+57f75040
     replacing  perl-utils.noarch 5.26.3-419.04.up1.module_uel20+109+57f75040
 perl-Algorithm-Diff     noarch   1.1903-14.uel20                 UnionTechOS-Server-20    34 k
```

图 10.3 使用--allowerasing 参数安装 git