软件包的类型

- 源码包
  - 需要编译
- 二进制包
  - 已编译

常见二进制包类型

- RedHat/Centos
  - RPM
  - 工具
    - rpm
    - yum(在线安装，自动解决依赖关系)
    - dnf(yum的升级版本，在RockyLinux中，yum命令是dnf的软连接)
- Ubuntu/Debain
  - DPKG
  - 工具
    - dpkg
    - apt(在线安装，自动解决依赖关系)

注意！不管是源码包，还是二进制包，安装时都可能会有依赖关系

yum/dnf

yum/dnf是红帽系列的发行版本上用来下载和安装软件包的一个工具

我们可以使用`yum install xxx`命令来安装一个软件

但是大家有没有考虑过一个问题？yum工具是去哪里找到这些软件包的呢?

这就不得不提到yum源,所谓源的意思就是指这些软件包的来源，也可以成称之为软件仓库

dnf与yum的对比

YUM (Yellowdog Updater, Modified)：

- 是 RHEL 8 及之前版本使用的包管理工具。
- 提供了一个简单的命令行接口，用来安装、更新、删除和管理 RPM 软件包。
- 基于 Python 2 开发，在性能和依赖性解析方面存在一定的限制。

DNF (Dandified YUM)：

- 是 YUM 的下一代替代工具，首次在 Fedora 18 引入，并从 RHEL 8 开始逐步取代 YUM，到了 RHEL 9 完全淘汰 YUM。
- 采用了更先进的依赖解析器和更高效的代码。
- 基于 Python 3 开发，支持现代特性和技术，性能更高。

RockyLinux中yum命令的本质：

```
[root@localhost bin]# ll /usr/bin/ | grep dnf
lrwxrwxrwx. 1 root root          5 Apr 14  2024 dnf -> dnf-3
-rwxr-xr-x. 1 root root       2094 Apr 14  2024 dnf-3
lrwxrwxrwx. 1 root root          5 Apr 14  2024 yum -> dnf-3
```

由此可以看出，yum命令和dnf命令都链接自dnf-3。实际上dnf-3才是包管理器的真正可执行程序，代表基于python3开发。这样链接是为了我们使用起来更方便。

官方源

一些有名的大厂或者高校都有提供他们整合好的软件仓库

- 阿里云镜像：https://mirrors.aliyun.com/rockylinux/
- 腾讯云镜像：https://mirrors.cloud.tencent.com/rocky/
- 中科大镜像：https://mirrors.ustc.edu.cn/rocky/
- 上海交大镜像：https://mirrors.sjtug.sjtu.edu.cn/rocky/
- 西安交大镜像：https://mirrors.xjtu.edu.cn/rocky/
- 浙江大学镜像：https://mirrors.zju.edu.cn/rocky/
- 南京大学镜像：https://mirrors.nju.edu.cn/rocky/
- 山东大学镜像：https://mirrors.sdu.edu.cn/rocky/
- 兰州大学镜像：https://mirror.lzu.edu.cn/rocky/
- 大连东软镜像：https://mirrors.neusoft.edu.cn/rocky/

```
[root@localhost ~]# yum repolist        # 查看现有的软件仓库
repo id                                         repo name
appstream                                       Rocky Linux 9 - AppStream
baseos                                          Rocky Linux 9 - BaseOS
extras                                          Rocky Linux 9 - Extras
```

软件源相关文件存放的位置/etc/yum.repos.d/

```
[root@localhost ~]# ls /etc/yum.repos.d/
rocky-addons.repo  rocky-devel.repo  rocky-extras.repo  rocky.repo

# 这里面的一个个.repo文件就是一个个软件仓库
```

可以尝试添加中科大软件仓库，下面的是RockyLinux的添加命令

```
# 备份原来的源
[root@localhost ~]# mkdir /repobackup
[root@localhost ~]# cp /etc/yum.repos.d/* /repobackup/
# 使用sed命令，替换原来repo文件中的内容
```

```
[root@localhost ~]# sed -e 's|^mirrorlist=|#mirrorlist=|g' \
    -e
's|^#baseurl=http://dl.rockylinux.org/$contentdir|baseurl=https://mirrors.
ustc.edu.cn/rocky|g' \
    -i.bak \
    /etc/yum.repos.d/rocky-extras.repo \
    /etc/yum.repos.d/rocky.repo

# 更新缓存
[root@localhost ~]# yum makecache
Rocky Linux 9 - BaseOS
4.1 MB/s | 2.3 MB     00:00
Rocky Linux 9 - AppStream
9.2 MB/s | 8.3 MB     00:00
Rocky Linux 9 - Extras
34 kB/s |  15 kB     00:00
Metadata cache created.
[root@localhost ~]# yum repolist
repo id                                             repo name
appstream                                           Rocky Linux 9 - AppStream
baseos                                              Rocky Linux 9 - BaseOS
extras                                              Rocky Linux 9 - Extras
```

- EPEL源(Extra Packages for Enterprise Linux)为“红帽系”的操作系统提供额外的软件包
  - `yum -y install epel-release`

yum常用命令

查看当前可用仓库

```
[root@localhost ~]# yum clean all       # 清空缓存及其他文件
[root@localhost ~]# yum makecache       # 重建缓存
[root@localhost ~]# yum repolist        # 查询可用的仓库
```

查找软件包

注意：有些命令跟软件包的名字是不同的

```
1. 通过命令的名字查找是属于哪个软件包
[root@localhost ~]# yum provides pstree
Last metadata expiration check: 0:00:20 ago on Fri Nov 22 14:50:43 2024.
psmisc-23.4-3.el9.x86_64 : Utilities for managing processes on your system
Repo        : @System
Matched from:
Filename    : /usr/bin/pstree

psmisc-23.4-3.el9.x86_64 : Utilities for managing processes on your system
Repo        : baseos
Matched from:
```

```
Filename    : /usr/bin/pstree

2. 通过配置文件查找属于哪个软件包提供的
[root@localhost ~]# yum provides /etc/ssh/sshd_config
Last metadata expiration check: 0:00:56 ago on Fri Nov 22 14:50:43 2024.
openssh-server-8.7p1-38.el9.x86_64 : An open source SSH server daemon
Repo        : @System
Matched from:
Filename    : /etc/ssh/sshd_config

openssh-server-8.7p1-43.el9.x86_64 : An open source SSH server daemon
Repo        : baseos
Matched from:
Filename    : /etc/ssh/sshd_config

3. 通过search参数查找软件包
[root@localhost ~]# yum search psmisc
Last metadata expiration check: 0:02:07 ago on Fri Nov 22 14:50:43 2024.
======================================== Name Exactly Matched: psmisc
========================================
psmisc.x86_64 : Utilities for managing processes on your system

4. 查看系统中的安装包
[root@localhost ~]# yum list installed          # 查看已安装的所有软件包
[root@localhost ~]# yum list                    # 查看所有可以安装的软件包
[root@localhost ~]# yum list httpd
```

安装软件包

```
# yum install xxx

- 安装httpd软件包
[root@localhost ~]# yum install -y httpd
-y：一律允许
[root@localhost ~]# yum install -y httpd vim nginx      # 多个软件包之间用空格
隔开
[root@localhost ~]# yum reinstall httpd                 # 重新安装
[root@localhost ~]# yum update httpd                    # 更新软件包的版本-
laster
[root@localhost ~]# yum update                          # 更新所有软件包到软件
仓库的最新版本

# 从本地安装
[root@localhost ~]# yum install -y httpd-2.4.57-11.el9_4.x86_64.rpm
Last metadata expiration check: 0:06:47 ago on Fri Nov 22 14:50:43 2024.
Error:
 Problem: conflicting requests
  - nothing provides httpd-core = 2.4.57-11.el9_4 needed by httpd-2.4.57-
11.el9_4.x86_64 from @commandline
(try to add '--skip-broken' to skip uninstallable packages or '--nobest'
to use not only best candidate packages)
```

```
# 从URL安装
[root@localhost ~]# yum install -y
https://rpmfind.net/linux/almalinux/9.4/AppStream/x86_64/os/Packages/httpd
-2.4.57-11.el9_4.x86_64.rpm
Last metadata expiration check: 0:07:55 ago on Fri Nov 22 14:50:43 2024.
httpd-2.4.57-11.el9_4.x86_64.rpm
27 kB/s |  45 kB     00:01
Error:
 Problem: conflicting requests
  - nothing provides httpd-core = 2.4.57-11.el9_4 needed by httpd-2.4.57-
11.el9_4.x86_64 from @commandline
(try to add '--skip-broken' to skip uninstallable packages or '--nobest'
to use not only best candidate packages)
```

管理软件包组

软件包组就是把一些相关的软件包给整合起来，可以打包安装和卸载

```
[root@localhost ~]# export LANG="en-US.utf8"            切换英文环境
[root@localhost ~]# yum grouplist
Failed to set locale, defaulting to C.UTF-8
Last metadata expiration check: 0:08:59 ago on Fri Nov 22 14:50:43 2024.
Available Environment Groups:
   Server with GUI
   Server
   Workstation
   KDE Plasma Workspaces
   Custom Operating System
   Virtualization Host
Installed Environment Groups:
   Minimal Install
Available Groups:
   Fedora Packager
   VideoLAN Client
   Xfce
   Legacy UNIX Compatibility
   Console Internet Tools
   Container Management
   Development Tools
   .NET Development
   Graphical Administration Tools
   Headless Management
   Network Servers
   RPM Development Tools
   Scientific Support
   Security Tools
   Smart Card Support
   System Tools

[root@localhost ~]# yum groupinfo "Server with GUI"
```

```
[root@localhost ~]# yum groupinstall "Server with GUI"
[root@localhost ~]# yum -y groupremove "Server with GUI"
```

切换图形化模式

- 临时切换到图形化模式

```
[root@localhost ~]# systemctl isolate graphical.target
```

- 永久切换（开机进入图形化）

```
[root@localhost ~]# systemctl set-default graphical.target

# 验证启动模式
[root@localhost ~]# systemctl get-default
multi-user.target
```

- 临时切换到命令行模式

```
[root@localhost ~]# systemctl isolate multi-user.target
```

- 永久切换

```
[root@localhost ~]# systemctl set-default multi-user.target
```

软件包组解释

Environment Groups（环境组）

- **Server with GUI** GNOME 桌面环境及相关应用
- **Server** 无图形界面的纯服务器环境，包含基本的服务器组件（SSH、存储管理、网络管理工具等）
- **Workstation** 桌面工作站环境，适合普通用户和开发人员，包含办公软件、多媒体支持和开发工具
- **KDE Plasma Workspaces** KDE Plasma 桌面环境及其相关工具
- **Custom Operating System** 自定义操作系统环境，用户可以手动选择需要的软件包
- **Virtualization Host** 专用于虚拟化的环境，包含 KVM、`libvirt` 和虚拟化管理工具
- **Minimal Install** 最小化安装，仅包含基础系统和命令行工具

Available Groups（可用的软件包组）

- **Fedora Packager** 用于构建 Fedora 或 RHEL/CentOS 软件包的工具（如 `rpmdevtools` 和 `mock`）

- **VideoLAN Client** VLC 媒体播放器及其依赖项
- **Xfce** 轻量级桌面环境，适合资源受限的设备或低性能机器
- **Legacy UNIX Compatibility** 提供传统 UNIX 系统兼容的工具和库（如 `csh` 和 `compat-libstdc++`）
- **Console Internet Tools** 基于命令行的互联网工具（如 `wget`、`curl` 和文本浏览器 `lynx`）
- **Container Management** 用于管理容器的工具（如 Podman 或 Docker）
- **Development Tools** 常用开发工具（如 `gcc` 编译器、`gdb` 调试器和 `git` 版本控制工具）
- **.NET Development** 用于开发和运行 .NET 应用程序的工具和运行时
- **Graphical Administration Tools** 图形化的系统管理工具（如磁盘管理和网络配置工具）
- **Headless Management** 用于远程管理无图形界面服务器的工具
- **Network Servers** 常见网络服务（如 `httpd`、`vsftpd` 和 `bind`）
- **RPM Development Tools** 用于构建和维护 RPM 软件包的工具（如 `rpmbuild` 和 `createrepo`）
- **Scientific Support** 科学计算工具和库（如 `numpy` 和 `scipy`）
- **Security Tools** 安全扫描、入侵检测和分析工具（如 `nmap`、`wireshark` 和 `clamav`）
- **Smart Card Support** 与智能卡相关的工具和库，支持基于智能卡的身份认证
- **System Tools** 常用的系统管理工具（如磁盘管理和日志分析工具）

卸载

```
[root@localhost ~]# yum -y remove httpd                    # 卸载httpd软件包
[root@localhost ~]# yum -y groupremove "Server with GUI"         # 卸载软件包
组
```

历史记录

用来查看 yum 软件包管理器的历史记录的

```
[root@localhost ~]# yum history
[root@localhost ~]# yum history info 4                     # 查看具体某个yum记录
的信息
```

yum自建源

除了使用上述提到的官方源之外，我们也可以自己搭建一个自建源用来为系统提供软件仓库

实验一：本地yum镜像仓库

可以通过安装系统时下载的iso中的软件包来提供本地软件仓库

RockyLinux ISO下载地址：https://dl.rockylinux.org/vault/rocky/9.4/isos/x86_64/Rocky-9.4-x86_64-dvd.iso

iso文件中具有一些基础的软件包

```
- 挂载本地ISO光盘：
[root@localhost ~]# ls
Rocky-9.4-x86_64-dvd.iso  anaconda-ks.cfg
[root@localhost ~]# mkdir /media/cdrom
```

```
[root@localhost ~]# mount Rocky-9.4-x86_64-dvd.iso /media/cdrom/
mount: /media/cdrom: WARNING: source write-protected, mounted read-only.
[root@localhost ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
devtmpfs             4.0M     0  4.0M   0% /dev
tmpfs                872M     0  872M   0% /dev/shm
tmpfs                349M  5.2M  344M   2% /run
/dev/mapper/rl-root   17G   12G  5.1G  71% /
/dev/nvme0n1p1       960M  261M  700M  28% /boot
tmpfs                175M     0  175M   0% /run/user/0
/dev/loop0            11G   11G     0 100% /media/cdrom

- 清空原有的软件仓库
[root@localhost ~]# rm -f /etc/yum.repos.d/*

- 配置本地yum
[root@localhost ~]# vim /etc/yum.repos.d/local.repo
[BaseOS]
name=RockyLinux 9.4 BaseOS
baseurl=file:///media/cdrom/BaseOS
enabled=1
gpgcheck=0

[AppStream]
name=RockyLinux 9.4 AppStream
baseurl=file:///media/cdrom/AppStream
enabled=1
gpgcheck=0

=======================字段解释=========================
-name 软件仓库名字
-baseurl 软件包的存放的地址
-enabled 是否启用这个仓库
-gpgcheck 是否校验软件包

# 安装软件验证
[root@localhost cdrom]# yum install -y httpd
```

实验二：提供内网服务器软件包更新

一台server1服务器可以访问公网，一台server2服务器无法访问公网

我们可以在这台能够访问公网的服务器上下载好软件包，通过http或者ftp提供给内网中的server2机器安装软件包

server1：

第一步：下载软件包，可以使用`--downloadonly`选项仅下载不安装

第二步：将软件包放到合适的位置上，并且创建软件索引数据

```
[root@localhost ~]# yum clean all                          # 清理缓存
[root@localhost ~]# yum update -y --downloadonly --downloaddir=/packages
# 仅下载更新包
[root@localhost ~]# ls /packages/   # 查看下载的包

[root@localhost ~]# mkdir /var/ftp/update
[root@localhost ~]# find /packages -iname "*.rpm" -exec cp -rf {}
/var/ftp/update/ \;
[root@localhost ~]# yum -y install createrepo           # createrepo用来创建
软件包的索引
[root@localhost ~]# createrepo /var/ftp/update/         # 创建软件包的索引
```

第三步：部署ftp文件服务器，将软件包通过ftp协议共享给sever2

```
# 部署ftp服务并且启动
[root@localhost ~]# yum -y install vsftpd
[root@localhost ~]# vim /etc/vsftpd/vsftpd.conf     # 更改ftp的配置文件，允许
匿名登录
anonymous_enable=yes

[root@localhost ~]# systemctl start vsftpd

# 关闭防火墙和selinux
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# setenforce 0
```

server2：

在server2上我们编写repo文件，软件的来源是有server1通过ftp服务提供

```
[root@localhost ~]# rm -rf /etc/yum.repos.d/*.repo  #干掉所有的仓库
[root@localhost ~]# vi /etc/yum.repos.d/update.repo
[update]
name=RockyLinux update
baseurl=ftp://192.168.88.104/update
gpgcheck=0
enabled=1

# 关闭防火墙和selinux
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# setenforce 0

[root@localhost ~]# yum clean all
[root@localhost ~]# yum update -y       # 更新测试
```

RPM

在RHEL系列的系统上所使用的软件包为rpm包，上面使用的yum为在线安装软件包。除了在线的安装方法，我们还可以将rpm包下载到本地，然后使用rpm工具来安装这个软件包

- rpm工具使用分为安装、查询、验证、更新、删除等操作
- rpm包的格式说明：

```
[root@atopos ~]# rpm -ivh httpd-2.4.6-93.el7.centos.x86_64.rpm
- httpd：包名
- 2：主版本号
- 4：次版本号
- 6：修订次数，指的是第几次修改bug
- 93：release（第几次发布，指的是简单的修改参数）
- e17：操作系统版本
- x86_64：64位操作系统
```

- 命令格式

```
[root@atopos ~]# rpm [选项] 软件包
```

选项

安装：

- -i：install的意思，安装软件包
- -v：显示附加信息，提供更多详细信息
- -V：校验，对已安装的软件进行校验

查询：

- -q：查询，一般跟下面的参数配合使用
- -a：查询所有已安装的软件包
- -f：系统文件名（查询系统文件属于哪个安装包）
- -i：显示已安装的rpm软件包信息
- -l：查询软件包文件的安装位置
- -p：查询未安装软件包的相关信息
- -R：查询软件包的依赖性

卸载：

- -e：erase
- --nodeps：忽略依赖

升级：

- -U：一般配合vh使用

实例：安装

```
[root@localhost ~]# rpm -ivh
https://rpmfind.net/linux/almalinux/9.4/AppStream/x86_64/os/Packages/httpd
-2.4.57-11.el9_4.x86_64.rpm

# 或者
[root@localhost ~]# wget
https://rpmfind.net/linux/almalinux/9.4/AppStream/x86_64/os/Packages/httpd
-2.4.57-11.el9_4.x86_64.rpm
[root@localhost ~]# rpm -ivh httpd-2.4.57-11.el9_4.x86_64.rpm
错误：依赖检测失败：
        /etc/mime.types 被 httpd-2.4.6-95.el7.centos.x86_64 需要
        httpd-tools = 2.4.6-95.el7.centos 被 httpd-2.4.6-
95.el7.centos.x86_64 需要
        libapr-1.so.0()(64bit) 被 httpd-2.4.6-95.el7.centos.x86_64 需要
        libaprutil-1.so.0()(64bit) 被 httpd-2.4.6-95.el7.centos.x86_64 需要

# 需要自己手动解决依赖关系

# 额外安装选项
# --nosignature 不检验软件包的签名
# --force 强制安装软件包
# --nodeps 忽略依赖关系
```

实例：查询

```
[root@atopos ~]# rpm -q vim-enhanced
vim-enhanced-7.4.629-8.el7_9.x86_64
[root@atopos ~]# rpm -qa |grep vim-enhanced
vim-enhanced-7.4.629-8.el7_9.x86_64
[root@atopos ~]# rpm -ql vim-enhanced
/etc/profile.d/vim.csh
/etc/profile.d/vim.sh
/usr/bin/rvim
/usr/bin/vim
/usr/bin/vimdiff
/usr/bin/vimtutor
[root@atopos ~]# rpm -qc openssh-server
/etc/pam.d/sshd
/etc/ssh/sshd_config
/etc/sysconfig/sshd
[root@atopos ~]#
```

实例：卸载

```
[root@atopos ~]# rpm -e vim-enhanced
```

源码包管理

源码包获得途径：

- 官方网站
    - apache: www.apache.org
    - nginx: www.nginx.org
- github
    - www.github.com

编译源码包所需要的环境

这取决于源码所使用的编程语言和框架等，但是Linux基础软件包一般都是由C语言编写的，所以我们使用较多的编译环境为gcc、make等等

```
yum install -y ncurses* gcc gcc-c++ make
```

编译

源码包编译三部曲

```
- 第一步：
./configure
#可以指定安装路径，启用或者禁用功能等，最终生成makefile

- 第二步
make
#按Makefile文件编译。可以加 -j 2 参数使用两个cpu核心进行编译

- 第三步
make install
#按Makefile定义的文件路径安装

make clean
#清除上一次make命令所产生的object文件，要重新执行
configure时，需要执行make clean。
```

案例：编译炫酷代码雨

补充：压缩与解压缩

打包一个文件或者目录：

```
tar -cvf archive.tar /path/to/directory

-c 表示创建一个新的归档文件。
```

```
-v 表示在压缩过程中显示详细信息（verbose 详细模式）。
-f 指定归档文件的名称。

# 如果要创建gzip压缩包
tar -zcvf archive.tar.gz /path/to/directory
```

解压缩：

```
tar -xvf archive.tar

-x 表示解压缩。
-v 表示显示详细信息。
-f 指定要解压缩的归档文件。

# 如果归档文件是使用 gzip 压缩的，比如 archive.tar.gz，则需要添加 -z 选项：

tar -zxvf archive.tar.gz
```

开始编译

```
[root@localhost ~]# yum -y install ncurses* gcc gcc-c++
[root@localhost ~]# tar -zxvf cmatrix-1.2a.tar.gz
[root@localhost ~]# cd cmatrix-1.2a
[root@localhost cmatrix-1.2a]# ./configure
[root@localhost cmatrix-1.2a]# make
[root@localhost cmatrix-1.2a]# make install
[root@localhost cmatrix-1.2a]# cmatrix
```