Package Management in Linux

On Linux, software is typically built as a package, distributed through repositories, and managed on the end-user's system through package managers. Each Linux system typically contains thousands of packages, many of which are required dependencies for other packages.

Core Concepts for Package Management

Packages

Most software applications designed for Linux or Unix systems are distributed as packages, which are archives that contain the pre-compiled binary software files, installation scripts, configuration files, dependency requirements, and other details about the software. These packages are typically specific to a particular distribution and formatted in that distribution's preferred package format, such as .deb for Debian/Ubuntu and .rpm for CentOS/RHEL.

While it's relatively simple for a user to install a package file, there are other complexities to consider. These complexities include obtaining (downloading) the package, ensuring packages are upgraded with security and bug fixes, and maintaining all the dependencies for the software.

Repositories

While a user can obtain package files through any method of file transfer, it is typical to use software repositories (also called repos) to obtain packages. Repositories are simply the location where the packages are stored, commonly accessible via the internet. A repository can contain a single package or thousands of packages. Most Linux distributions have their own unique repositories, sometimes separating out their cores packages into one while additional features are in others.

Dependencies

Its very common in almost any operating system for software to require other software to run. In Linux, each package contains metadata detailing the additional packages that are required. These additional packages are called dependencies. A single package can sometimes have hundreds of dependencies. When installing, upgrading, or removing packages, these dependencies may also need to installed, upgraded, and optionally removed.

Package Managers

A package manager reduces the complexity for the end-user by automating the process of obtaining, installing, upgrading, and removing packages and their dependencies. This dramatically improves the user experience and the ability to properly and efficiently manage the software on your Linux system. Today, package managers can be a defining feature for Linux distributions and many system administrators prefer to use a particular distribution based on its package management system (among other considerations).

Benefits of Package Management Systems

- Easily obtain the correct, trusted, and stable package for your Linux distribution. Since most distributions maintain their own repositories, using a package manager can ensure you only install packages that have been thoroughly vetted, are stable, are trusted, and work with your system. The subjective judgments and community standards that guide package management also guide the “feel” and “stability” of a given system.

- Automatically manage all dependencies when taking action on a package. While dependencies can be distributed inside the original package to ensure the correct versions of the required software are always present, this increases disk usage and package file size. In most cases, package managers will manage dependencies as separate packages, allowing them to be shared with other software and reducing disk usage and file size.

- Follow the unique conventions for each Linux distribution. Linux distributions often have conventions for how applications are configured and stored in the /etc/ and /etc/init.d/ directories. By using packages, distributions are able to enforce a single standard.

- Stay up-to-date on security patches and software updates. Most package managers provide a single command to automatically update all packages to the latest versions stored on the configured repositories. Provided those repositories are consistently maintained, this enables you to quickly get important security updates.

Comparison of Package Managers

There are lots of package managers in Linux, each working a bit differently. Here is a list of common package managers, along with their supported distributions, package file formats, and a description.

APT

Using APT to Manage Packages in Debian and Ubuntu

Distributions: Ubuntu, Debian, and Kali Linux
Commands: apt, apt-get, apt-cache
Underlying package management tool: dpkg
Package file format: .deb

Advanced Package Tool, more commonly known as APT , is a package management system for Debian, Ubuntu, and other similar Linux distributions. It acts as a front-end to the lower-level dpkg package manager, which is used for installing, managing, and providing information on .deb packages. Most distributions that use APT also include a collection of command-line tools that can be used to interface with APT. These tools include apt-get, apt-cache, and the newer apt, which essentially combines both of the previous tools with some modified functionality.

DNF

Using DNF to Manage Packages in CentOS/RHEL 8 and Fedora

Distributions: RHEL/CentOS 8, Fedora 22, and later versions of both distributions
Commands: dnf, yum
Underlying package management tool: RPM (RPM Package Manager)
Package file format: .rpm

Dandified YUM, or simply DNF , is the successor to YUM. Just like YUM, DNF provides a user-friendly interface to the RPM Package Manager (RPM) that comes with CentOS, RHEL, Fedora, and many other Linux distributions. As the successor to YUM, DNF has several enhancements including increased performance, faster dependency resolution, and more complete documentation for its API. Most distributions still link the yum command to the DNF software and, since DNF maintains compatibility with much of YUM's CLI, most commands using yum still function as intended.

YUM

Using YUM to Manage Packages in CentOS/RHEL 7 and Earlier

Distributions: RHEL/CentOS 7, Fedora 21, and earlier versions of both distributions
Command: yum
Underlying package management tool: RPM (RPM Package Manager)
Package file format: .rpm

Yellowdog Updater, Modified, more commonly known as YUM , is a package management tool for a variety of older RHEL-based distributions (such as CentOS 7) and older versions of Fedora. It provides an easy-to-use interface on top of the low-level functions available in the RPM Package Manager (RPM). It has largely been replaced by it successor Dandified YUM, also called DNF, on most newer RPM-based distributions.

Zypper

Use Zypper to Manage Packages in openSUSE

Distributions: openSUSE
Command: zypper
Underlying package management tool: ZYpp (also called libzypp)
Package file format: .rpm

Zypper is the CLI tool used for managing packages on openSUSE Linux distributions. Just like DNF and YUM, packages are stored in the .rpm file format, though Zypper interfaces with the ZYpp (libzypp) library instead of RPM. Some users report that Zypper is faster than other package managers and, unlike many others, has the ability to add repositories directly from its own CLI. See the Zypper manual for more usage details.

Pacman

Using Pacman to Manage Packages in Arch

Distributions: Arch-based, including Arch and Manjaro
Command: pacman
Package file format: .tar.xz (and other compressed tar formats)

Arch Linux and other similar distributions (like the popular Manjaro desktop distro) use the pacman package manager. Packages are stored as compressed tarballs and, as such, are generally smaller than other package formats. Pacman is unique in that it comes with a system to build packages, not just manage them. This system is called the ABS (Arch Build System).

Portage

Using Portage to Manage Packages in Gentoo

Distributions: Gentoo
Command: emerge
Package file format: ebuild shell script or .tbz2 (compressed tar archive)

Portage , the package manager for Gentoo, is quite a bit different than other solutions. Instead of using pre-compiled binary packages, Portage packages are typically shell scripts called ebuilds. The emerge CLI tool can run these shell scripts to install packages, and is also responsible for managing dependencies and a database of installed packages.

Slackware Package Management

Managing Packages in Slackware

Distributions: Slackware
Commands: slackpkg, pkgtool, installpkg, upgradepkg, removepkg
Package file format: .tgz and .txz (compressed tar archive)

Slackware comes bundled with a few package management tools. The pkgtool is a TUI (menu-driven text interface) for managing packages and installing local packages. To install packages located on the internet, the slackpkg tool can be used. For more advanced tasks, use specialized tools like installpkg, upgradepkg, and removepkg.