Package Management System and systemctl

Package Management System in Linux

A ***package management system or package manager*** is a group of software tools. It automates the installation process, upgrading process, configuration process, and removing process of the computer programs for an operating system of the computer in an efficient manner. A ***package manager*** works with packages, data within archive files, and software distributions.

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.
- ✓ **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.
- ✓ **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.
- ✓ **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 update

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

Types 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.

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 Manger (RPM). It has largely been replaced by its successor Dandified YUM, also called DNF, on most new RPM-based distributions.

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

Zypper

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

Install docker and Jenkins using package managers

1. **To install Docker on Ubuntu, follow these steps:**

   Install & Update the package index:

   sudo apt install

   sudo apt update

2. **Install Docker:**

   sudo apt install docker.io

Install Jenkins on Ubuntu

Since Jenkins is written in Java, the first step is to install Java.

1. Update the package index:

   sudo apt update

2. Install Jenkins:

   sudo apt install Jenkins

3. Start the Jenkins daemon:

   sudo systemctl start jenkins

4. To enable the Jenkins daemon to start on boot:

   sudo systemctl enable jenkins

systemctl and systemd

The *systemctl* command is a systemd utility used to manage services, get information about service unit files and service states, and therefore a useful utility to know for managing services on the server while *systemd* is an array of components for Linux OS.

The *systemctl* command in a Linux utility is used to manage the systemd service and service manager.
systemd is an init system and system manager, It runs with PID 1 and it is the one responsible for starting the rest of the system.

Different systemctl services

```
$ sudo systemctl start mariadb

$ sudo systemctl stop mariadb

$ sudo systemctl restart mariadb

$ sudo systemctl reload mariadb

$ sudo systemctl status mariadb

$ sudo systemctl enable mariadb

$ sudo systemctl disable mariadb
```