Cal State Fullerton

CPSC 254

Software Development With Open Source Systems - Package Management

Instructor: Tejaas Mukunda Reddy

Package Management

A package manager or package management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner.

A package manager deals with packages, distributions of software and data in archive files. Packages contain metadata, such as the software's name, description of its purpose, version number, vendor, checksum, and a list of dependencies necessary for the software to run properly. Upon installation, metadata is stored in a local package database. Package managers typically maintain a database of software dependencies and version information to prevent software mismatches and missing prerequisites. They work closely with software repositories, binary repository managers, and app stores.

Package managers are designed to eliminate the need for manual installs and updates. This can be particularly useful for large enterprises whose operating systems are based on Linux and other Unix-like systems, typically consisting of hundreds or even tens of thousands of distinct software packages.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

Functions

A software package is an archive file containing a computer program as well as necessary metadata for its deployment. The computer program can be in source code that has to be compiled and built first. Package metadata include package description, package version, and dependencies (other packages that need to be installed beforehand).

Package managers are charged with the task of finding, installing, maintaining or uninstalling software packages upon the user's command. Typical functions of a package management system include:

- Working with file archivers to extract package archives
- Ensuring the integrity and authenticity of the package by verifying their digital certificates and checksums
- Looking up, downloading, installing or updating existing software from a software repository or app store
- Grouping packages by function to reduce user confusion
- Managing dependencies to ensure a package is installed with all packages it requires, thus avoiding "dependency hell"

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

APT Tool

APT tool Debian, Ubuntu, other derivatives

Advanced Packaging Tool (APT)Permalink

You may already be familiar with apt-get, a command which uses the advanced packaging tool to interact with the operating system's package system. The most relevant and useful commands are (to be run with root privileges):

- apt-get install package-name(s) - Installs the package(s) specified, along with any dependencies.
- apt-get remove package-name(s) - Removes the package(s) specified, but does not remove dependencies.
- apt-get autoremove - Removes any orphaned dependencies, meaning those that remain installed but are no longer required.
- apt-get clean - Removes downloaded package files (.deb) for software that is already installed.
- apt-get purge package-name(s) - Combines the functions of remove and clean for a specific package, as well as configuration files.
- apt-get update - Reads the /etc/apt/sources.list file and updates the system's database of packages available for installation. Run this after changing sources.list.
- apt-get upgrade - Upgrades all packages if there are updates available. Run this after running apt-get update.

APT Cache

While apt-get provides the most often-used functionality, APT provides additional information in the apt-cache command.

- apt-cache search package-name(s) - If you know the name of a piece of software but apt-get install fails or points to the wrong software, this looks for other possible names.
- apt-cache show package-name(s) - Shows dependency information, version numbers and a basic description of the package.
- apt-cache depends package-name(s) - Lists the packages that the specified packages depends upon in a tree. These are the packages that will be installed with the apt-get install command.
- apt-cache rdepends package-name(s) - Outputs a list of packages that depend upon the specified package. This list can often be rather long, so it is best to pipe its output through a command, like less.
- apt-cache pkgnames - Generates a list of the currently installed packages on your system. This list is often rather long, so it is best to pipe its output through a program, like less, or direct the output to a text file.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

Repositories

Combining most of these commands with apt-cache show can provide you with a lot of useful information about your system, the software that you might want to install, and the software that you have already installed. If you're overwhelmed by apt-cache check out the following resources for easy-to-read lists of available packages:

- The Debian Package Directory
- The Ubuntu Package Directory

Repositories

To give users more control over the kinds of software that they are allowing to be installed on their system (and sometimes due to legal or convenience reasons on the distributors' side), software is often downloaded from a number of software repositories.[4] https://en.wikipedia.org/wiki/Package_manager

Software Repository

What is a Software Repository?

Repositories

A Linux distribution provides a command, and usually a graphical interface to that command, that pulls the software from the server and installs it onto your computer. It's such a simple concept that it has served as the model for all major cellphone operating systems and, more recently, the "app stores" of the two major closed source computer operating systems. - https://opensource.com/article/18/1/how-install-apps-linux

Repositories store container images which represent executable packages containing everything required to run a discrete application including the code, system tools, system libraries, and the runtime and settings of the application.

Public repositories securely store, publish, and freely share open-source software. Organizations use private repositories to manage their proprietary software resources. They can publish that software and charge fees through licensing arrangements. Code repositories support the discovery of software assets and promote code reuse.

Development teams often rely on both open-source packages and those built within their organization. Some repositories only support package managers for specific development platforms such as Java (npm), Python (pip), Ruby (RubyGems), or .Net (NuGet). Integrated software repository solutions support multiple package types from both public and private sources. The repositories identify potential licensing and security issues.

When you install Linux, your Software Centre will come pre-configured with a default Repository. Normally, this is perfectly acceptable, but there are times when the applications you need are not listed in this repository and it becomes necessary to add additional Repositories to access the additional functionality. - http://www.linuceum.com/Desktop/linuxAppRepos.php

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

Software Repository

There are exactly two types of repositories: local and remote: the local repository is a directory on the computer where Maven runs. It caches remote downloads and contains temporary build artifacts that you have not yet released.

Software Repositories Features

- Build, maintain, store, or source software packages and containers from public and private feeds
- Package management
- Deployment tracking
- Version control
- Access controls
- Encrypted storage and backup
- Asset discovery And Pipeline workflow tools
- Release management tools
- Static code analysis And Vulnerability testing
- Dashboards, statistics, and reporting

Where is the Debian Repository?

/etc/apt/source.list contains the url that points to the repository.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn