launch_code

John Woolbright, Technical Curriculum Developer

June, 15, 2022

Package Manager

Introduction

A key aspect of a Linux distribution is the Package Manager.

A distribution's package manager is the preferred tool for managing software on the computer.

Package

A Package is software and additional metadata.

The additional metadata includes: dependencies, version, origin, essential to operating system, conflicts, source, and more.

The package contains the executable software and additional data for managing the software.

Under the Hood

A package is a collection of compressed files that can easily be made into executable files. Many packages are configured to work directly with a package manager. However, it is possible to manually build packages to access the executable software.

For Debian based (like Ubuntu) distributions you will regularly find .deb packages.

For RedHat based (like CentOS) distributions you will regularly find .rpm packages.

Manual packages are commonly bundled together as a .tar file.

Package Repository

All packages that are managed directly by a package manager have an associated repository. This is the web location in which the package files can be downloaded.

Part of the Package Manager's responsibilities is in maintaining these package repository lists.

For the major linux distributions there are thousands of repositories. Many of which are maintained by the linux distribution themselves.

Package Repo’s in Ubuntu

Checkout the list of Packages managed by Ubuntu for Ubuntu 22.04 (jammy): https://packages.ubuntu.com/jammy/

This is the list of official Ubuntu Packages. You can add new package repositories from parties outside of Ubuntu, which would give you access to even more packages.

Common Package Managers

The majority of Debian based distributions use APT (Advanced Package Tool).

The majority of RedHat based distributions use RPM (RPM Package Manager).

These are the predominate underlying package managers in the Linux world. However, they are low level tools used directly by the operating system. End users interface with these tools with various text-based or graphical clients.

Package Manager Clients

For APT we have the CLI's named apt and apt-get / apt-cache.

For RPM there are the CLI's named rpm and yum.

Most linux distributions that come with a graphical windows system also provide a graphical client to access the underlying package manager (Ubuntu Software in Ubuntu 22.04).

We will only be using the CLI options from our Bash shell.

Git Overview

Review

Version Control Software created by Linus Torvalds

Similar to the Linux Kernel (also created by Torvalds) it is open source software

Basic Workflow

Adding changes to staging

Committing changes to local git repo

Pushing changes from local repo to remote

Creating a Local Repo

git init (within project directory) will initialize a git repository in your current directory

This allows you to begin controlling "versions" of your project

Branches

Branches allow you to diverge from your master or main branch of development

You are able to create a new branch in multiple ways:

git branch new-branch-name

git checkout -b new-branch-name

Merging

This course will cover two different types of merging strategies

Traditional merge with git merge

merging with git rebase

Traditional Merge

Using the traditional merge strategy allows you to keep the original history of commits in-tact

It also means that you will be working directly on your main or master branch of development

Git Rebase

git rebase will reapply commits from your current branch on top of the target branch

You are working with a feature branch and conflicts are handled prior to merging directly into your main or master branch of development

A rebase does alter the history of the feature branch

launch_code

launchcode.org