Declarative Package Management

Freya Murphy 🐈

February 20, 2026

RITlug

contents, Table of

What is a package?

le package 📦

A package is could contain either both its output contents (commonly known as its binaries), and its inputs (dependencies) and build scripts.

1. Dependencies / Inputs (Other Packages)
2. Some source files
3. A build script
4. And its outputs

Depending on your package manager, you will get either just the package source, or its binaries, or both.

Common package formats

There are many common package formats used in linux distributions today...

1. .deb - Debian based distributions - binary tarball
2. .apk - Alpine - binary tarball
3. .pkg - Pacman (arch) - binary tarball
4. .rpm - REHL based - binary idk its weird
5. ebuild - Gentoo - Build script

Ignoring gentoo... you may see a common format

Everything is just a binary

Unless you hate your computer, *or* are Chris, compiling all your programs is annoying.

So...

Most linux distributions distribute binary packages. These are just containers or prebuilt files to place on your system, and maybe some activation scripts.

The good

If everyone is doing this, there has to be a good reason right?

1. Its simple (it just works™)
2. You don't have to compile things
3. The files go in a **consistent** spot
    3.1 I sure love how */bin/bash* is standard but also not standard
4. That's pretty much it

The bad?

There are some obvious downsides though

1. You cannot easily downgrade
    1.1 Have fun compiling everything your self...
2. How can we guarantee the validity of the package contents?
3. What if I require multiple versions of a package?
    3.1 What if two packages have conflicting files?

What is state?

”State” is just untracked manual modification.

1. Edited config file (/etc or .config)
2. List of installed packages (/etc/apk/world on alpine)
3. Firefox profile

State isn’t inherently bad, some state is even needed, but state can cause issues that are not reproducible, or not even known (good luck root causing a random config file).

Declarative Package Management

What does declarative mean?

A **declaration** is something that is made known formally, officially, or explicitly.

- Gentoo ebuild files!

To get rid of state, package builds must be **declarative**, and must be **reproducible**.

- Inputs must be reproducible
- Package source must be reproducible (hash)
- Build script must be declarative
- Sandboxed

Why sandboxing?

Outside state can modify the package build results if the build script depends on it.

```
#!/bin/sh

# let $out be the path to store package contents
cp /etc/resolv.conf $out/etc/resolv.conf

# super not reproducable
# /etc/resolv.conf is not gurenteed to be
# anything specific
```

So what is a package now?

1. **Sandboxed** and **functional** package build script
2. Package source gurenteed with a hash
3. Set of input packages (not binaries) that are also reproducable

But wait? Isn't this just gentoo with extra steps? I want binaries, not scripts!!!

Yes... and also no!

Derivations and substitutes

A **derivation** is a finalized source and hash of a package along with its source and inputs.

1. If any input to the derivation changes, the derivation itself changes
2. If the package source changes, the derivation itself changes
3. If the package script changes, the derivation itself changes

A derivation is the full ”look” into a reproducible package. If two derivations are the same, they ARE the same package.

Derivations and substitutes cont.

A **substitute server** hosts prebuilt packages with their derivations as their keys in a ”hash map” storage system.

Given a derivation, a substitute server may return a prebuilt package binary, such that you don’t have to build it yourself. (yay no gentoo)

1. You do have to *trust* these servers though.
2. They could serve malicious packages...

Package collisions, or lack there of

The last issue we have to talk about is package collisions. What if two packages want to serve the same file at a given location?

Solution: don't let anyone put any files anywhere!

1. Have a "store" that holds the outputs of packages
2. Symlink its outputs to areas on the filesystem in the currently active "profile"

A **profile** is an **atomic** set of "active" (in use) packages.

Benefit: Profiles can be swapped out (woah we can now rollback!)

Examples

Examples

The two most pouplar declarative package managers are **Nix** and **Guix**.

See nixos.org and guix.gnu.org