Write the Nix package of a Rust project in 2022

an opinionated comparison

@yvan-sraka

Where I started?

Back in 2019, I learned NixOS tools to configure my VPS (understand how to find the right configuration options and how to write custom package overlays), and then after finally read Nix Pills (and understand Nix/Nixpkgs language and concepts).

Nowadays, I want to package stuffs! Googling things make me end up on https://nixos.wiki :)

A bunch of solution ...

- `rustPlatform`
- `naersk`
- `cargo2nix`
- ... and other alternatives (some are fairly outdated ...)

Rust is just a pretext, the same question arise for every other languages and frameworks. A "crate" (a Rust module) is made of a hierarchy of `.rs` source files with a `Cargo.toml` / `Cargo.lock`

Number of derivations?

Just 1 derivation:

- Faster (less Nix overhead... really?)
- Simpler (less Nix code to maintain... can I just rely on some Nix framework?)

More (than 1) derivation:

- `/nix/store` better cache incremental builds (you like to cache in your CI)
- `/nix/store` should help different programs to share same derivations

Did it rely on IFDs? (Import From Derivation)

What?

A handy feature: evaluate Nix code (generate by a derivation) on the fly!

Why?

It's quite hard to implement a custom language dependencies manager custom format parser in Nix (e.g., `.cabal`) while it's easy to adapt existing one to generate Nix expressions (e.g., `cabal2nix`)!

Side note: `nixpkgs` repository don't like that you open a PR of a package that would heavily rely on IFD ...

Flake or not flake?

What is the `nixpkgs` default approach?

It defines `rustPlatform.buildRustPackage`.
https://nixos.org/manual/nixpkgs/stable/#chap-language-support

What does `nix flake init templates#rust` does?

It generates a flake with `naersk` as input.
https://github.com/NixOS/templates

nixpkgs

Basically, it requires you to specify a checksum of all the downloaded dependencies (but also offer you to just set the path to `Cargo.lock`), and will build the project and all its dependencies in one derivation.

naersk

`naersk` dynamically imports and parses `Cargo.lock` from a Nix expression, which means it needs to be present in the same repository.

A project is built into 2 derivations: one for project Rust sources depending on one other for all its dependencies (defined in `Cargo.lock`).

cargo2nix

`cargo2nix` user-generate a `Cargo.nix` from a `Cargo.lock` (that should be committed to source control).

One derivation per crate (dependency)!

What else?

Opening: can I rather have just one big, generic solution? `dream2nix` offer a consistent wrapper around (at user choice) either `nixpkgs` or `crane`.

Reading list:

- https://ipetkov.dev/blog/introducing-crane
- https://determinate.systems/posts/introducing-riff

Q/A