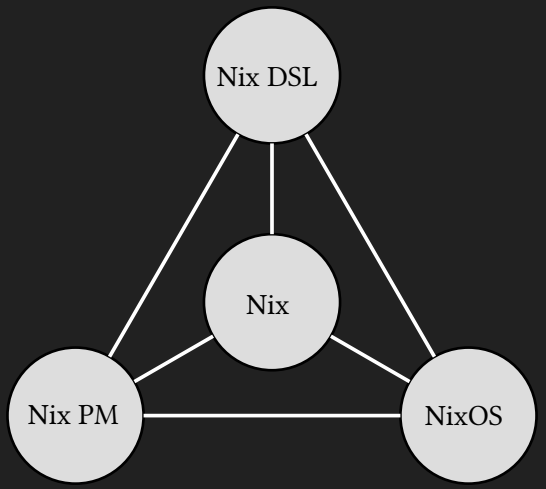
What is Nix?

OtaNix ry

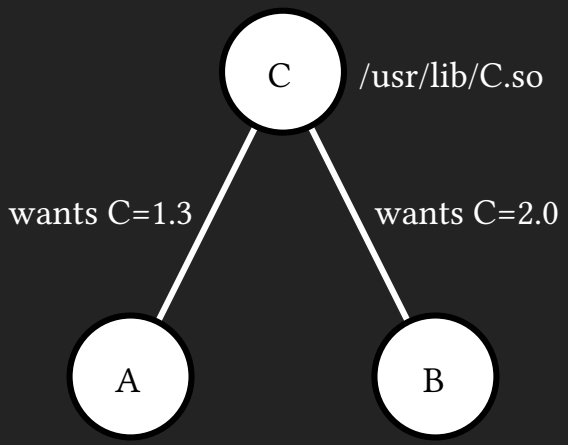
Matias Zwinger

2025-01-22

Aalto University


Nix DSL
Nix
Nix PM
NixOS

Namespace problem



This is called dependency hell

How Nix solves this

- Both versions of C share the same path
- Add version to file path?
  - Example: Conda
  - They could still be different
- **(name + version + source code + dependencies)** ➡ hash
- `pwnbnq2wlxx31fn1s1388pbwgll5kk12-C-1.3`

Nix store

- All packages reside in `/nix/store`
  - FHS violation (or augmentation)
- Immutable for users
- Can be modified using Nix commands
- A package stays in the store until it is **garbage collected**
  - Manually or automatically (cron, systemd timer)

Nix commands

- Enter new shell with software available: `nix shell -f "<nixpkgs>" foo`
  - To expose the package, nix uses symlinks and $PATH modification
- Run `foo` directly: `nix run -f "<nixpkgs>" foo -- --arg1 --arg2`
- Collect garbage: `nix-collect-garbage`

nixpkgs

- The largest collection of Nix packages
  - Hosted on GitHub
  - “Wikipedia approach”: anyone can submit a package
  - Maintained by volunteers (like me)

Nix language

- Every package manager has a way to describe packages
- Nix has its own language
  - Declarative
  - Functional
  - Lazy
  - Dynamically typed
  - “JSON with functions”

An example

GNU hello, stripped down version of the nixpkgs package

```
stdenv.mkDerivation (finalAttrs: {
  pname = "hello";
  version = "2.12.1";

  src = fetchurl {
    url = "mirror://gnu/hello/hello-${finalAttrs.version}.tar.gz";
    hash = "sha256-jZkUKv2SV28wsM18tCqNxoCZmLxdYH2Idh9RLibH2yA=";
  };
})
```

- `stdenv` can build packages that use the UNIX standard of `./configure; make; make install;`

Two birds with one package manager

Normal package managers:

- Serve **already compiled** programs (binaries)
- Only tells you how to install them

Nix

- Also specifies how to **build** software
- You can build everything on your own computer, like on Gentoo
- Binaries can still be downloaded from public cache
  - Nix checks what hash the output *would* have, and downloads it from the cache if available

Putting packages in the store is nice

What if I put my whole operating system there?

NixOS

- All OS components are in the store
  - Programs
  - Config files
  - Systemd services
  - Kernel
  - Bootloader (partly)

```
{
  programs.git = {
    enable = true;
    config.init.defaultBranch = "main";
  };
  services.openssh.enable = true;
  boot.kernelPackages = linuxPackages_5_62;
  boot.lanzaboote.enable = true;
}
```

- Everything in the system is specified using Nix expressions
  - Infrastructure as code
  - Modify system config and rebuild system to install software
    - `sudo nixos-rebuild switch`

Corollary benefits

- Creating a symlink is atomic in UNIX
  - Power outage ⟶ either symlink exists or not
  - There is no in-between state
  - Switching NixOS generations = creating one symlink to `/run/current-system`
    - NixOS updates are **atomic**
    - **No invalid states**
- The Nix store contains previous generations
  - If the update breaks something, reboot and roll back

Any questions?