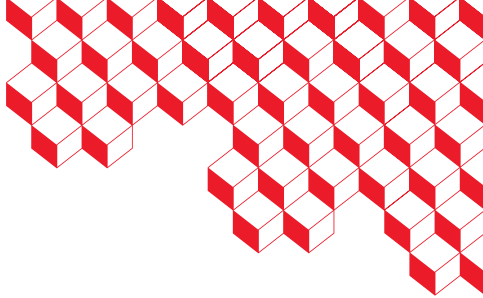


Packaging and system administration with Nix and NixOS

Rémi NICOLE

CEA/DRF/IRFU/DIS/LDISC

Inspiration

Matthiew Croughan, and his SCaLE talks

Demos?

They are here:[1]

You need to install Nix on your machine to run them.

[1] https://github.com/minijackson/2024-04-hepix-nix

Note on experimental features

They simplify my life, they are *not* needed.

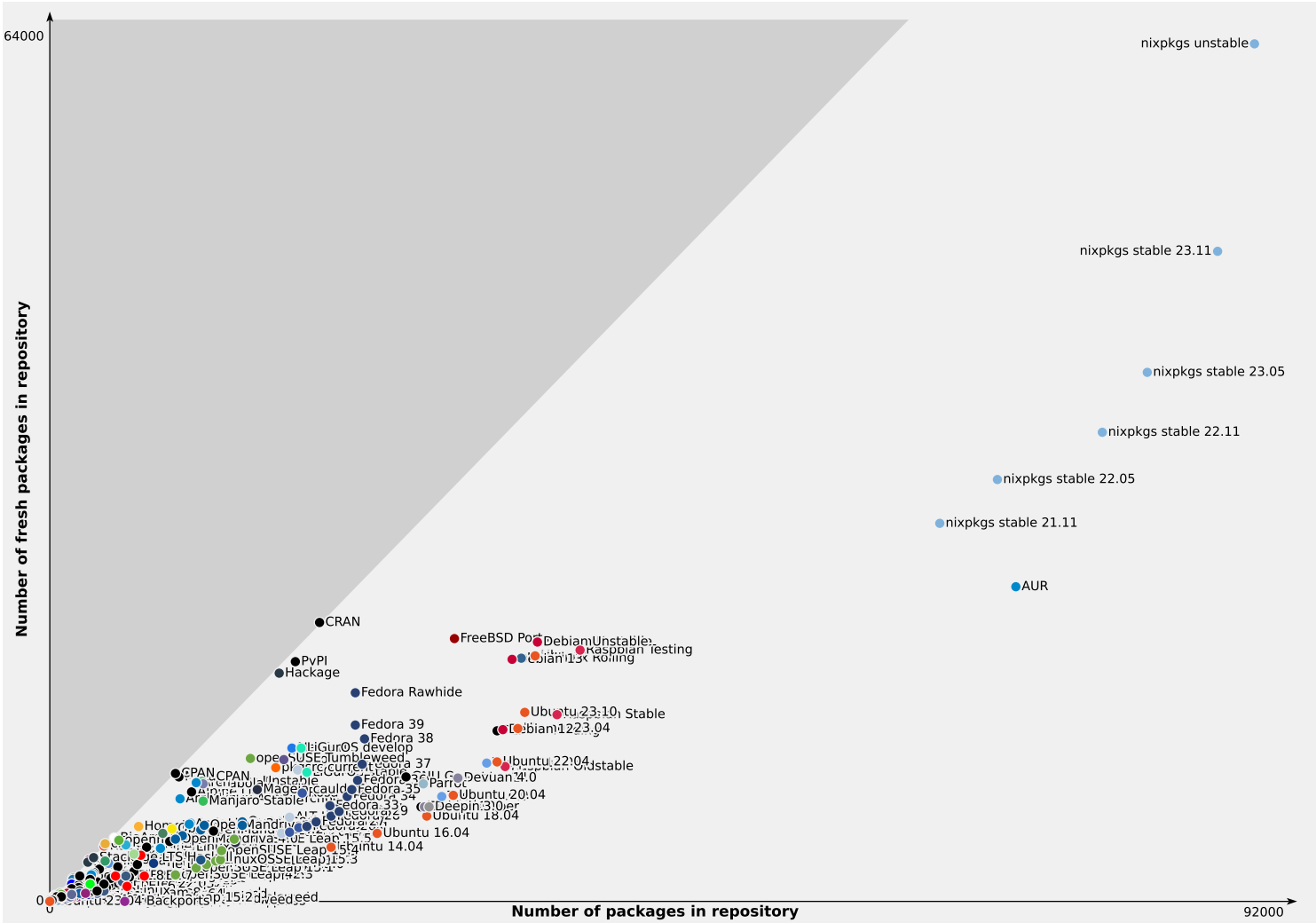
If you want to test the demos, enable the `flakes` experimental feature[2]

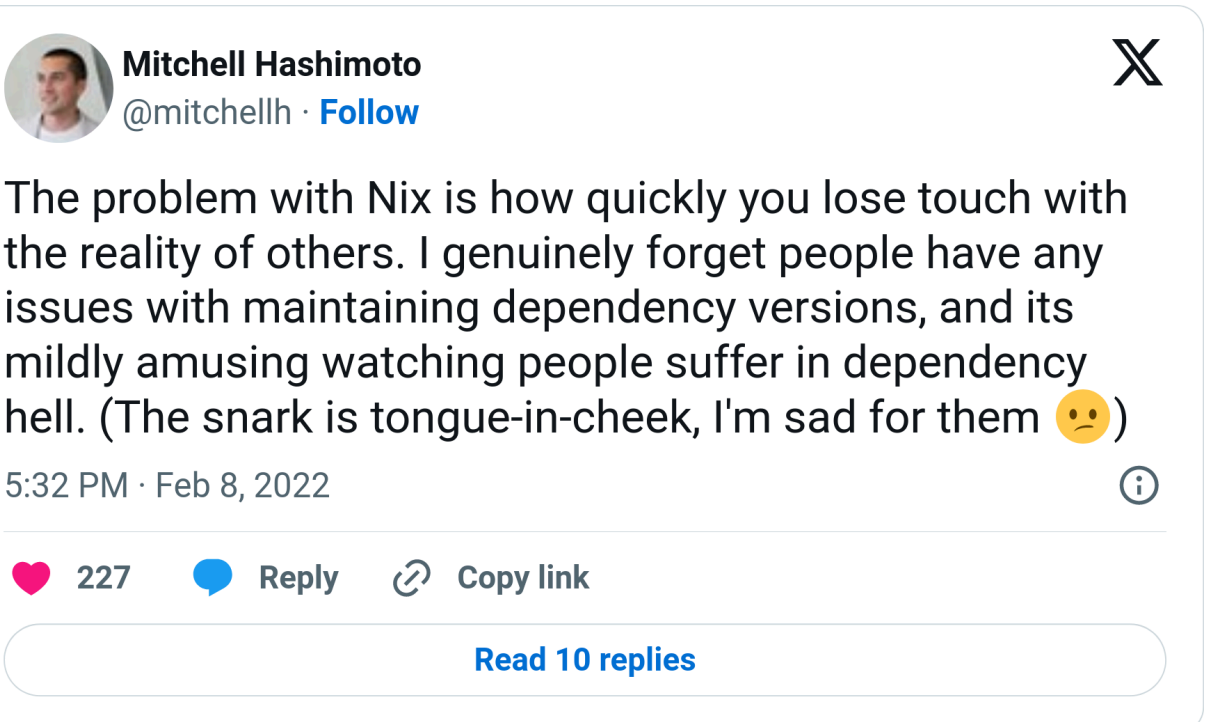
[2] https://wiki.nixos.org/wiki/Flakes#Other_Distros,_without_Home-Manager

1. Convincing you

The buzzwords slide

- Infrastructure as code (part of it)
- Reproducibility
- Software supply chain security (SLSA)
- Software Bill of Materials (SBOM)


64000
Number of fresh packages in repository
0
Number of packages in repository
92000
nixpkgs unstable
nixpkgs stable 23.11
nixpkgs stable 23.05
nixpkgs stable 22.11
nixpkgs stable 22.05
nixpkgs stable 21.11
AUR
CRAN
FreeBSD Port
PyPI
Hackage
Fedora Rawhide
Fedora 39
Fedora 38
Fedora 37
Ubuntu 20.04
Ubuntu 18.04
Ubuntu 16.04
Ubuntu 14.04
Manjaro Stable
Deepin 20er


Mitchell Hashimoto
@mitchellh · Follow
The problem with Nix is how quickly you lose touch with the reality of others. I genuinely forget people have any issues with maintaining dependency versions, and its mildly amusing watching people suffer in dependency hell. (The snark is tongue-in-cheek, I'm sad for them 😕)
5:32 PM · Feb 8, 2022
227
Reply
Copy link
Read 10 replies

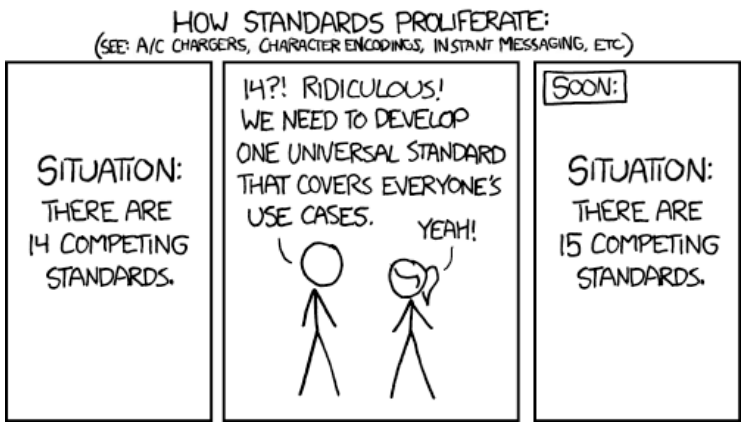


Figure 3: xkcd #97 — Standards

2. Concepts

Definitions

Nix — the package manager
Nix — the programming language
Nixpkgs — the Repository
NixOS — the Linux Distribution

Definitions

Nix — the package manager
Nix — the programming language
Nixpkgs — the Repository
NixOS — the Linux Distribution

Definitions

Nix — the package manager
Nix — the programming language
Nixpkgs — the Repository
NixOS — the Linux Distribution

Definitions

Nix — the package manager
Nix — the programming language
Nixpkgs — the Repository
NixOS — the Linux Distribution

How it works

1. Evaluate the Nix code ⟶ Package
2. Fetch / Compile dependencies
3. Fetch / Build the package in a sandbox
4. Scan the output for runtime dependencies

How it works

1. Evaluate the Nix code ⟶ Package
2. Fetch / Compile dependencies
3. Fetch / Build the package in a sandbox
4. Scan the output for runtime dependencies

How it works

1. Evaluate the Nix code ⟶ Package
2. Fetch / Compile dependencies
3. Fetch / Build the package in a sandbox
4. Scan the output for runtime dependencies

How it works

1. Evaluate the Nix code ⟶ Package
2. Fetch / Compile dependencies
3. Fetch / Build the package in a sandbox
4. Scan the output for runtime dependencies

How it works

1. Evaluate the Nix code ⟶ Package
2. Fetch / Compile dependencies
3. Fetch / Build the package in a sandbox
4. Scan the output for runtime dependencies

Each package is installed in its own directory

```
/nix/store/{hash}-coreutils-full-9.3/
```

3. Demos

Nix

- Reproducible development environment
  - They can have any package inside it
- Reproducible packages
  - You can figure out what its dependencies are
  - You can figure out what its *build* dependencies are
    - You can produce SBOM files from that
  - You can copy the package on any system that has Nix installed
  - It works on any Linux distribution
- Reproducible Docker images
  - Since you know all your runtime dependencies, you can produce minimal container images, without having to install a Linux distribution inside it

NixOS

- Declarative configuration
  - Say what you want in your Linux machine
  - Any setting removed from the configuration, gets removed from the system
- Rollbacks
  - You can go back to any previous configuration, completely offline
- Build VMs, Docker images, and more
  - Since the configuration *is* the system, you can produce VMs, Docker images, etc. to test your configuration before deploying
- Offline machines
  - You can build the configuration, such as including new software, on a separate machine
  - Then you can deploy it on an offline machine

NixOS tests

- Test of OpenArena, a first person shooter game
  1. Start 3 VMs: a game server, and 2 clients
  2. Wait for the machines to boot, and for the game server to start
  3. Start the game on the 2 clients, and connect them to the server
  4. Disconnect the “ethernet cable” of client1, wait for 10 seconds, and reconnect
  - Make screenshots throughout the process
- BitTorrent test
  1. Start 4 VMs: a tracker in VLAN 1, a router in VLAN 1 & 2, a client in VLAN 1, and a client in VLAN 2
  2. Make client1 download a torrent from the tracker
  3. We stop the tracker
  4. Make client2 download a torrent from client1, through the router

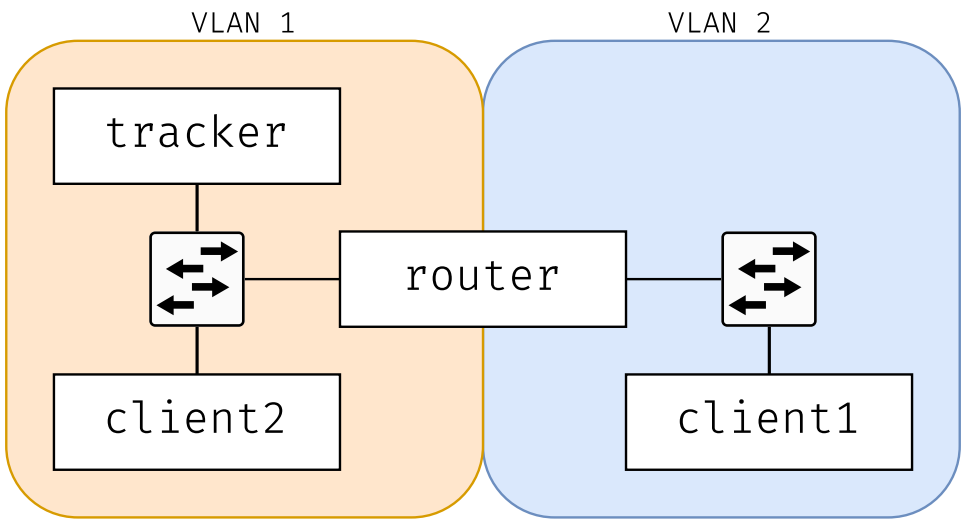


Figure 4: BitTorrent test setup

4. Software supply chain

Advantages of Nix/NixOS

- Know all your dependencies
- Know all your *build* dependencies
- Know metadata of all your dependencies
  - Licenses
  - Source provenance and patches
- SLSA level 4 “for free”
- Fetch the source and patches of all your dependencies
- Reproducible (science!)
- Cache everything
- Mix and match old / new software

Nice projects

- nixos-hardware[3]
- EPNix: EPICS + Nix[4]
- PPC64BE-ELFv2 network boot images, all in RAM
- robotnix: Building Android images[5]
- nix-darwin[6]

[3] https://github.com/NixOS/nixos-hardware
[4] https://github.com/epics-extensions/EPNix/
[5] https://github.com/danielfullmer/robotnix
[6] https://github.com/LnL7/nix-darwin

More reading

- Nix The Planet - SCaLE 21x — Matthew Croughan[7]
- What Nix Can Do (Docker Can't) - SCaLE 20x — Matthew Croughan[8]
- Nix is a better Docker image builder than Docker's image builder — Xe Iaso[9]
- SLSA demo — Tomberek[10]
- Own your CI with Nix — Théophane Hufschmitt[11]

[7] https://www.youtube.com/watch?v=6iviTZfiLGU
[8] https://www.youtube.com/watch?v=6Le0IbPRzOE
[9] https://xeiaso.net/talks/2024/nix-docker-build/
[10] https://www.youtube.com/watch?v=dT0DGVbD-5M&t=985s
[11] https://fosdem.org/2024/schedule/event/fosdem-2024-2282-own-your-ci-with-nix/