Preparing a 30 year-long project with Nix and NixOS

Rémi NICOLE

Who am I?

Figure 1: Minijackson

1. Context

A particle accelerator

A particle accelerator generating neutrons.

Composed of lots of hardware, like:

- Power supplies
- Magnets
- Beam diagnostics
- Cryogenics
- Plasma chamber
- ...

A need of control

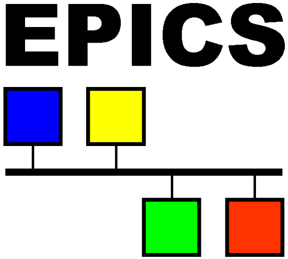

Figure 2: Old logo

Packaging the control

Figure 3: EPNix Logo

A need of isolation

Network as isolated as possible

A need of resilience

Lots of assumptions to rethink.

We could have to modify a project 10 years after development.

2. Nix and NixOS and resilience

Development resilience

With flake-style development, projects are *pinned*.

It should be easy to pick them up again years after deployment.

Source code resilience

Some software might not be available in the future.

Source code resilience

Some software might not be available in the future.

Solution:

- CI
- cache server
- cache *everything*
  - runtime dependencies
  - build-time dependencies (*everything*)
  - flake inputs
  - Nix itself (done by default for NixOS)

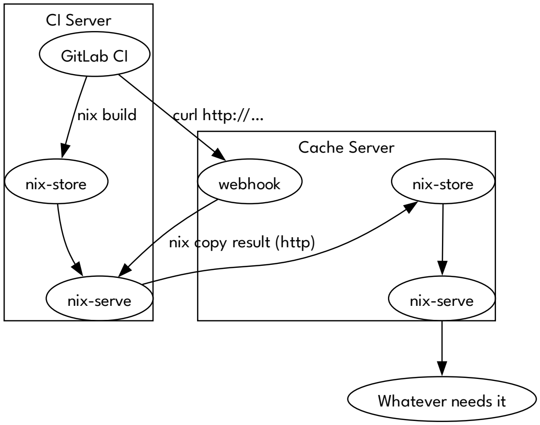

Figure 4: Cache continuous integration

Profiles

Usage of nix profiles to have a deletion policy.

We can differentiate between old versions and new versions of garbage collection roots.

```
system-583-link -> /nix/store/ ...
system-584-link -> /nix/store/ ...
system-585-link -> /nix/store/ ...
system-586-link -> /nix/store/ ...
system-587-link -> /nix/store/ ...
```

Conclusion

I have high hopes that Nix can be useful for building reliable and resilient systems.

Building some parts of a resilient infrastructure are still manual and need documentation.

Links

- https://github.com/epics-extensions/EPNix/
- https://github.com/NixOS/bundlers (the `toBuildDerivation` bundler)