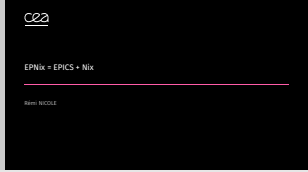
EPNix = EPICS + Nix

Rémi NICOLE

EPNix = EPICS + Nix



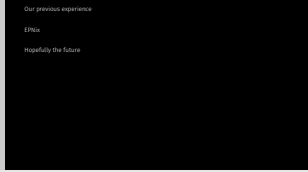
Our previous experience

EPNix

Hopefully the future

EPNix = EPICS + Nix

2022-09-16



Our previous experience

Our previous experience

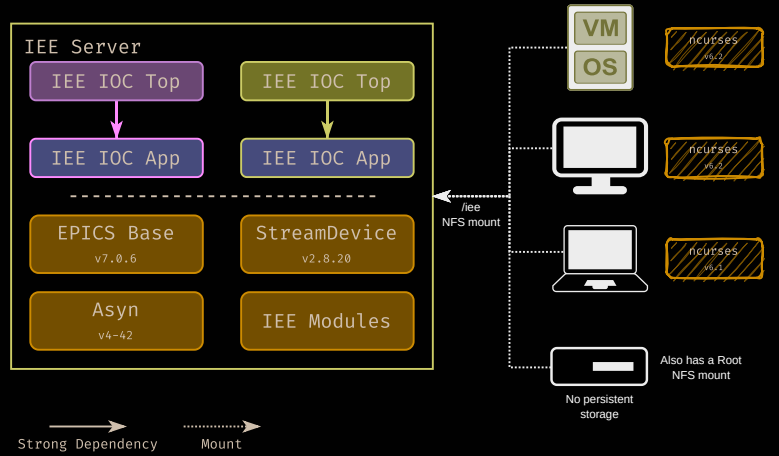
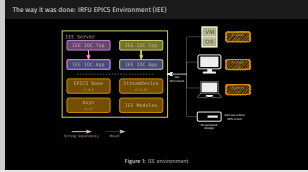


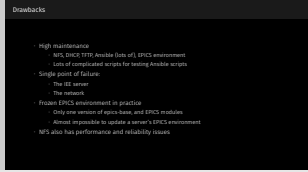
Figure 1: IEE environment



- IEE server, usually virtualised in a Proxmox cluster
- Central NFS server, containing EPICS IOCs, apps, support, etc.
  - For development and production
  - For developers and target machines
- Everybody's `/iee` folder is a NFS mount
- For boards without persistent storage, everybody's root folder is also a NFS mount
- Ansible to automate installation

Drawbacks

- High maintenance
  - NFS, DHCP, TFTP, Ansible (lots of), EPICS environment
  - Lots of complicated scripts for testing Ansible scripts
- Single point of failure:
  - The IEE server
  - The network
- Frozen EPICS environment in practice
  - Only one version of epics-base, and EPICS modules
  - Almost impossible to update a server's EPICS environment
- NFS also has performance and reliability issues



If network goes down temporarily, what happens with NFS is pretty unpredictable

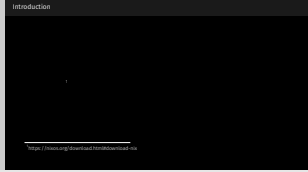
EPNix

EPNix

Introduction

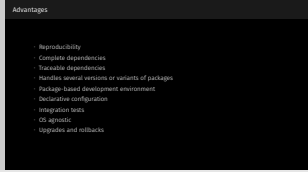
Goal: using Nix[1] to package EPICS IOCs

[1] https://nixos.org/download.html#download-nix



Advantages

- Reproducibility
- Complete dependencies
- Traceable dependencies
- Handles several versions or variants of packages
- Package-based development environment
- Declarative configuration
- Integration tests
- OS agnostic
- Upgrades and rollbacks



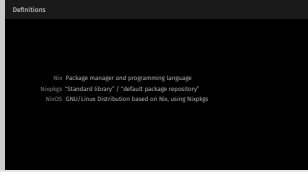
Dans le cadre de EPNix, un top est un paquet

Definitions

Nix Package manager *and* programming language

Nixpkgs “Standard library” / “default package repository”

NixOS GNU/Linux Distribution based on Nix, using Nixpkgs



Nix particularities

- Source based
- Purely functional
- Every package installed in its own path under `/nix/store/`

Nix particularities
- Source based
- Purely functional
- Every package installed in its own path under /nix/store/

Démo

```
nix flake new \
  --template 'epnix#top' \
  my-top
```


Binary cache

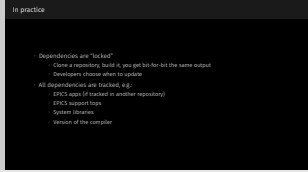
```nix
{
  # ...

  nixConfig = {
    extra-substituters = ["http://nix-cache.extra.cea.fr"];
    extra-trusted-public-keys =
      ↪ ["nix-cache.extra.cea.fr-1:......"];
  };

  # ...
}
```


In practice

- Dependencies are “locked”
  - Clone a repository, build it, you get bit-for-bit the same output
  - Developers choose when to update
- All dependencies are tracked, e.g.:
  - EPICS apps (if tracked in another repository)
  - EPICS support tops
  - System libraries
  - Version of the compiler



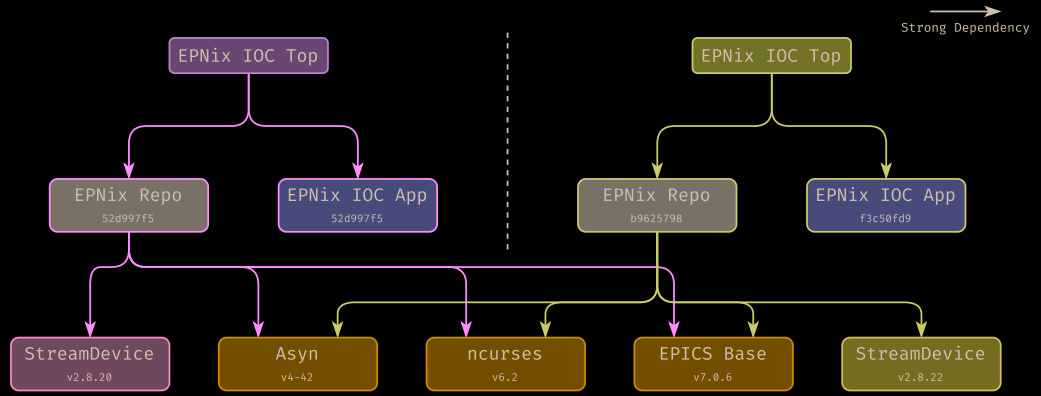
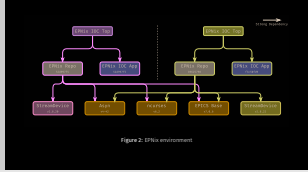


Figure 2: EPNix environment

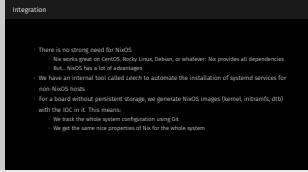


By strong dependency, I mean that it depends on this specific version

As a developer no need to care about the installation of the base / streamdevice, etc.

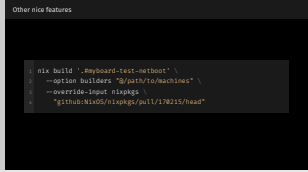
Integration

- There is no strong need for NixOS
  - Nix works great on CentOS, Rocky Linux, Debian, or whatever: Nix provides all dependencies
  - But… NixOS has a lot of advantages
- We have an internal tool called *Leech* to automate the installation of systemd services for non-NixOS hosts
- For a board without persistent storage, we generate NixOS images (kernel, initramfs, dtb) with the IOC in it. This means:
  - We track the whole system configuration using Git
  - We get the same nice properties of Nix for the whole system



Other nice features

```
nix build '.#myboard-test-netboot' \
  --option builders "@/path/to/machines" \
  --override-input nixpkgs \
    "github:NixOS/nixpkgs/pull/170215/head"
```

Drawbacks

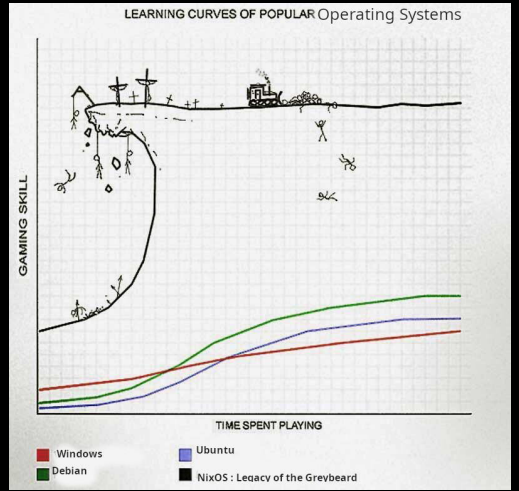
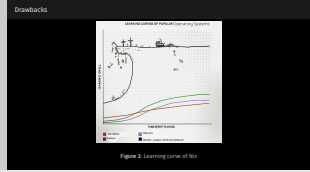


Figure 3: Learning curve of Nix



Due to mainly two things

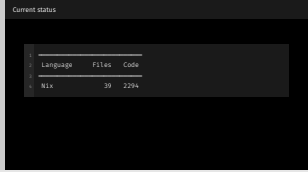
- Nix does things differently
- Documentation is often lacking, or too technical

Current status

```
═══════════════════════════════
 Language      Files   Code
═══════════════════════════════
 Nix              39   2294
```

Not open-source yet, but planned

Planned for production on a particle accelerator

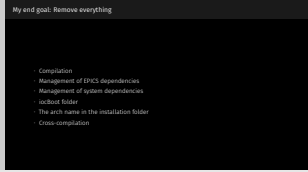


Hopefully the future

My end goal: Remove everything

A lot of EPNix is implemented to accommodate for EPICS' weirdness, in terms of:

- Compilation
- Management of EPICS dependencies
- Management of system dependencies
- iocBoot folder
- The arch name in the installation folder
- Cross-compilation



Hoping for the most normal EPICS v7 variant.

Having a PV access EPICS library, that's "just a library" like any other.

Since the properties of EPNix comes from Nix itself, it would allow us to have the same advantages, with a minimal amount of development.

It helps for other efforts too, like for packaging for Debian and other distributions, CI would be simpler, etc.

A “normal” IOC

```
stdenv.mkDerivation {
  pname = "epics-pv-access";
  version = "7.1.6";

  nativeBuildInputs = [ meson ninja pkg-config ];
  buildInputs = [ epics-com epics-pv-data epics-ca ];

  src = ./modules/pvAccess;

  doCheck = true;
};
```


Extract of my Nix packaging of PV libraries on top of the PR #260:
https://github.com/epics-base/epics-base/pull/260

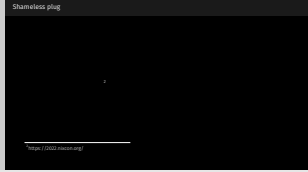
My hopes

I think a transition is possible right now.

Shameless plug

I'll be in NixCon 2022 in Paris[2], probably doing a talk on cross-compilation using Nix.

[2] https://2022.nixcon.org/



I will be talking about doing a network boot for a board without persistent storage, which starts an IOC compiled using EPNix.

The board runs NixOS, from an image which was entirely cross-compiled using Nix.

This will hopefully be in production in a particle accelerator.

Thank you!

For any question, contact, or complaint:

- Email: remi.nicole@cea.fr
- Matrix: @Minijackson:matrix.org