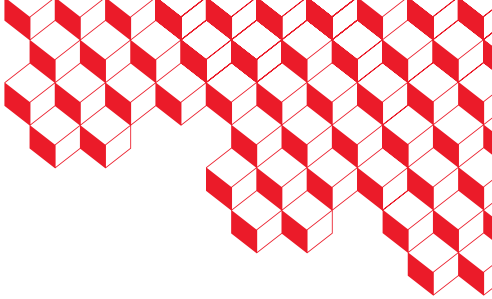
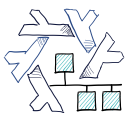


Why  EPNix?

Why do this to ourselves?

Rémi NICOLE

2026-04-20

Disclaimer

This presentation has a lot of opinions.

Disclaimer

This presentation has a lot of opinions.

These (bad) opinions are my own, and not necessarily those of my employer.

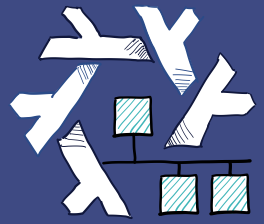
Disclaimer

This presentation has a lot of opinions.

These (bad) opinions are my own, and not necessarily those of my employer.

Or any of my colleagues, really.

1. What's

 EPNix?

EPNix goals

By using the **Nix** package manager and the **NixOS** Linux distribution,

EPNix enables:

EPNix goals

By using the **Nix** package manager and the **NixOS** Linux distribution, **EPNix** enables:

- Building **EPICS** tops & support modules

EPNix goals

By using the **Nix** package manager and the **NixOS** Linux distribution, **EPNix** enables:

- Building **EPICS** tops & support modules
- Manage **EPICS** dependencies

EPNix goals

By using the **Nix** package manager and the **NixOS** Linux distribution, **EPNix** enables:

- Building **EPICS** tops & support modules
- Manage **EPICS** dependencies
- Deploy **EPICS** IOCs

EPNix goals

By using the **Nix** package manager and the **NixOS** Linux distribution, **EPNix** enables:

- Building **EPICS** tops & support modules
- Manage **EPICS** dependencies
- Deploy **EPICS** IOCs
- Deploy other **EPICS**-related software:
  - Archiver Appliance,
  - The Phoebus stack, etc.

What are people saying about EPNix

I should try **NixOS** one day, I fear that I will like it…

— Ralph Lange (ITER)

"**EPNix**" ; at least the name is catchy.

— Simon Rose (ESS)

What are people saying about EPNix

I should try **NixOS** one day, I fear that I will like it…

— Ralph Lange (ITER)

"**EPNix**" ; at least the name is catchy.

— Simon Rose (ESS)

**This is not an official endorsement*

2. Why, then?

Advantages

- Reproducible

Advantages

- Reproducible
- Declarative

Advantages

- Reproducible
- Declarative
- Works for basically any language/ framework

Advantages

- Reproducible
- Declarative
- Works for basically any language/ framework
- Complete dependencies

Advantages

- Reproducible
- Declarative
- Works for basically any language/ framework
- Complete dependencies
- Locked dependencies

Advantages

- Reproducible
- Declarative
- Works for basically any language/ framework
- Complete dependencies
- Traced dependencies
- Lockable dependencies

Advantages

- Reproducible
- Declarative
- Works for basically any language/ framework
- Complete dependencies
- Traced dependencies
- Lockable dependencies
- Provides development shell

Advantages

- Reproducible
- Declarative
- Works for basically any language/ framework
- Complete dependencies
  - Traced dependencies
  - Lockable dependencies
- Provides development shell
- Declarative configurations
  - Integration tests
  - Interactive tests
- Offline deployments
- Vendor-agnostic continuous integration
  - VM image building
  - ISO building
  - Docker container building
- Appliance-style image building
- Works for embedded
- Concerns separation
- Configuration rollbacks
- Atomic upgrades
- Multi-machine declaration
- Apache server
- Cache dependencies
- Archive build dependencies
- Build isolation
- Install multiple concurrent versions
- Infrastructure as code
- Single configuration repository
- SBOM generation
- Immutable package store
- Specialization support
- Configuration checking before application
- Sub-container declaration
- Per-user package installation
- Declarative disk partitioning with disko
- No dependency on the host Linux distribution
- Parallel builds
- Configure **everything**

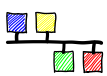
Advantages

- Reproducible
- Declarative
- Works for basically any language/ framework
- Complete dependencies
- Traced dependencies
- Lockable dependencies
- Provides development shells
- Declarative configurations
- Integration tests
- Interactive tests
- Offline deployments
- Vendor-agnostic continuous integration
- VM image building
- ISO building
- Docker container building
- Appliance image building
- Customizable framework
- Composable configuration
- Atomic upgrades
- Multi-machine declaration
- Apache server
- Caching dependencies
- Archive build dependencies
- Build isolation
- Multiple configurations
- Infrastructure as code
- Single configuration repository
- SBOM generation
- Immutable package store
- Specialization support
- Configuration checking before application
- Sub-configuration declaration
- Per-user package installation
- Declarative disk partitioning with disko
- No dependency on the host Linux distribution
- Parallel builds
- Configure **everything**

Who cares?

Human drawbacks

If you think about it, we adopted a technology:

- Not well known in the sysadmin community
- Completely unknown to the  **EPICS** community
- Completely unknown to our team
- Created and maintained by ~1 person

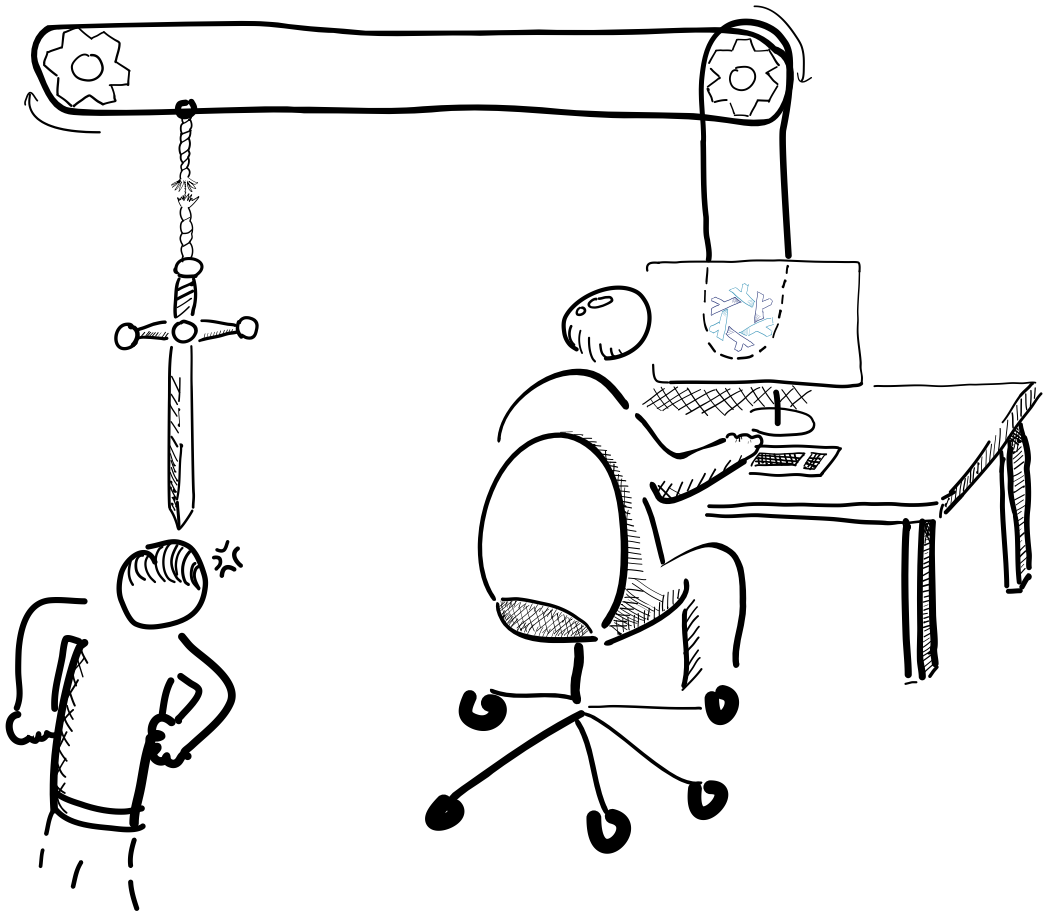
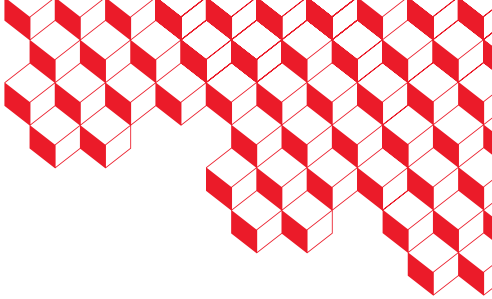


Figure 1: My manager watching me deploying something that only I understand

Who allowed  EPNix to exist in the first place?



EPNix: The HR factor (not the music band)

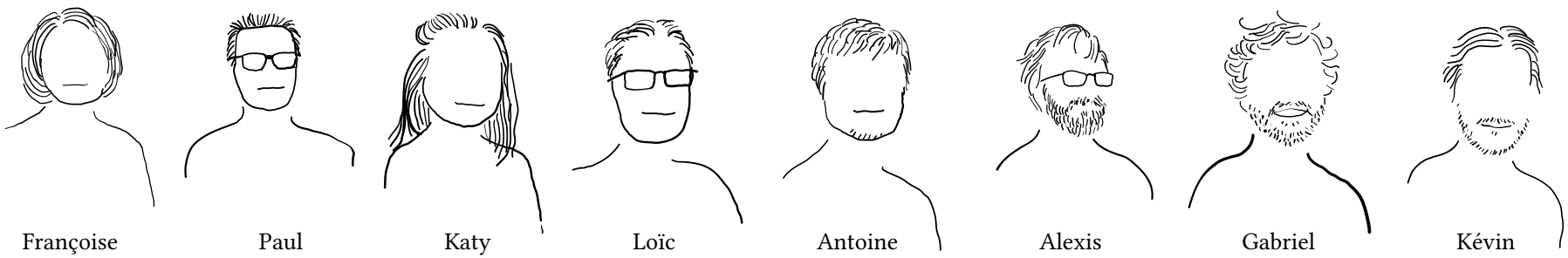
The **real**  **EPNix** presentation

---

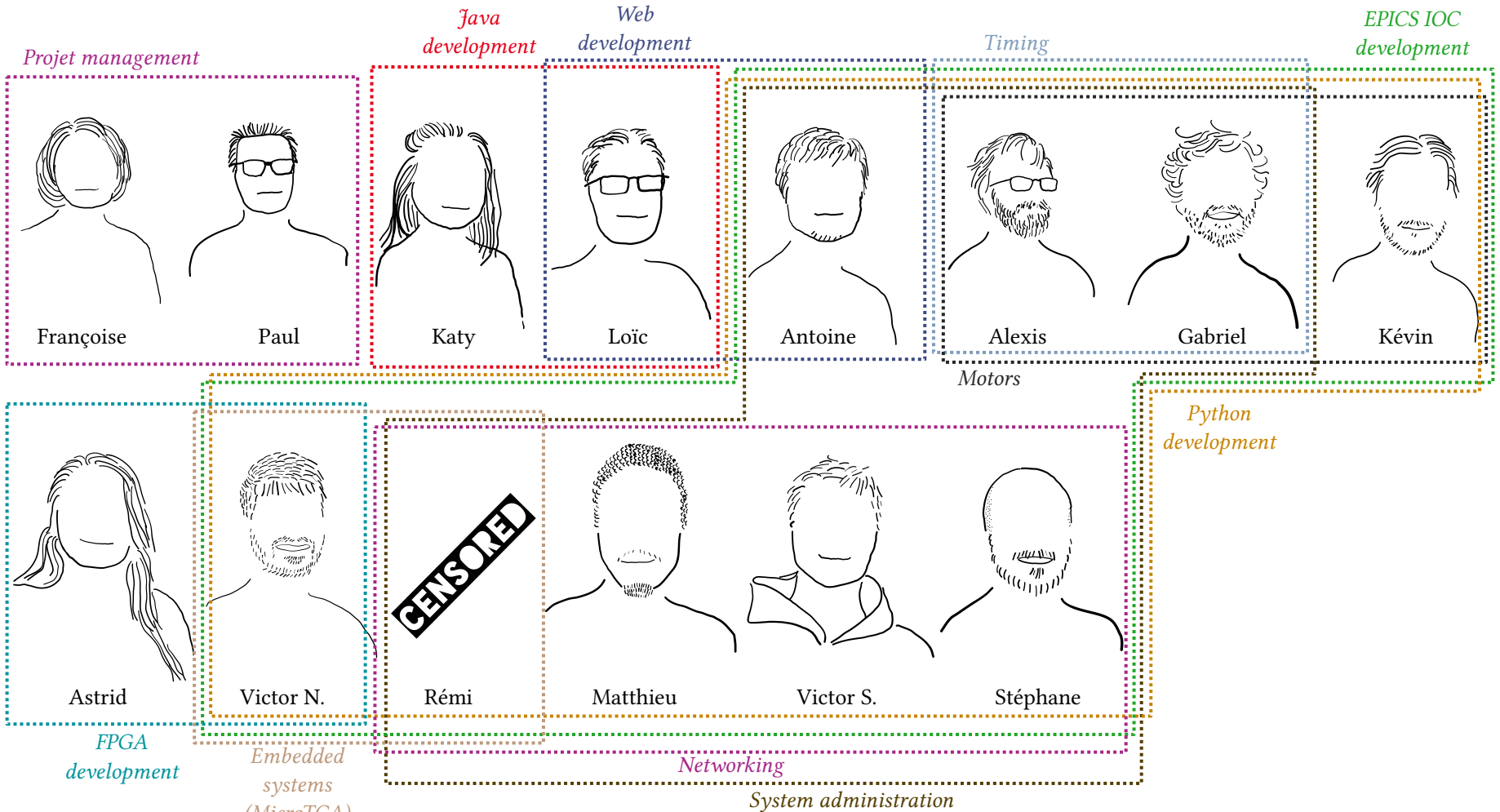
Rémi NICOLE (again)

2026-04-20

Our team


Françoise
Paul
Katy
Loïc
Antoine
Alexis
Gabriel
Kévin
Astrid
Victor N.
CENSORED
Matthieu
Victor S.
Stéphane

Our team


Projet management
Java development
Web development
Timing
EPICS IOC development
Françoise
Paul
Katy
Loïc
Antoine
Alexis
Gabriel
Kévin
Motors
Python development
Astrid
Victor N.
CENSORED
Rémi
Matthieu
Victor S.
Stéphane
FPGA development
Embedded systems (MicroTCA)
Networking
System administration

Our #1 goal:
Reduce maintenance

What really happened

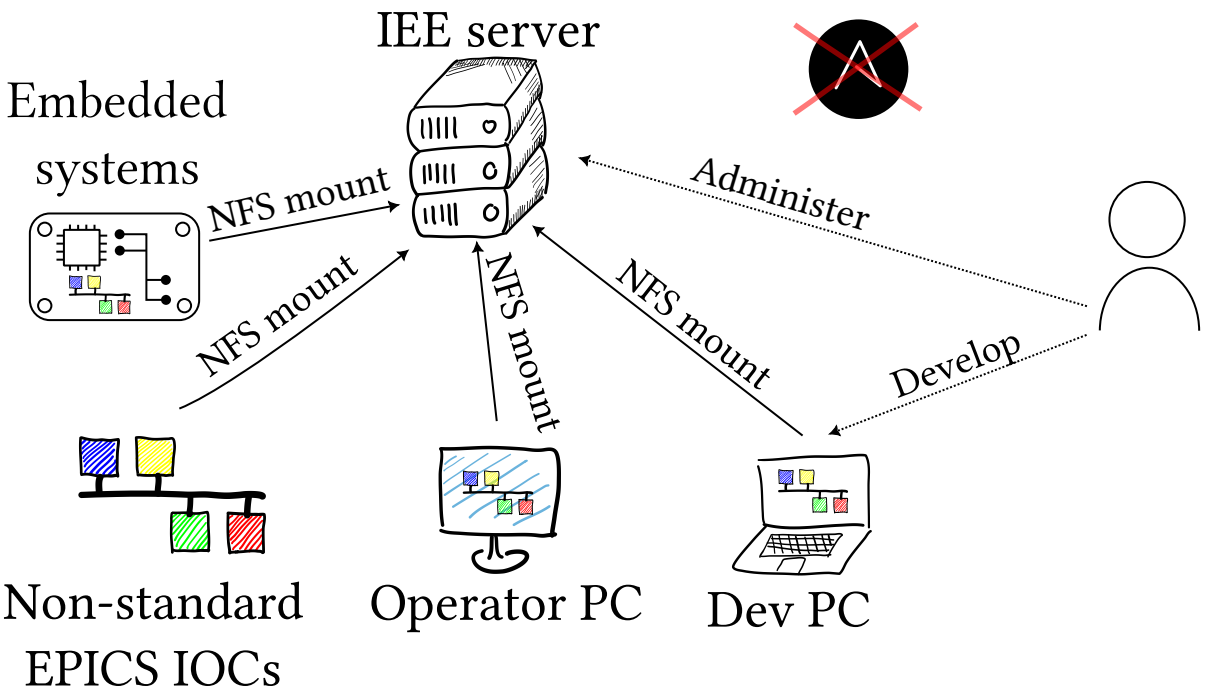


Figure 6: Our IEE architecture

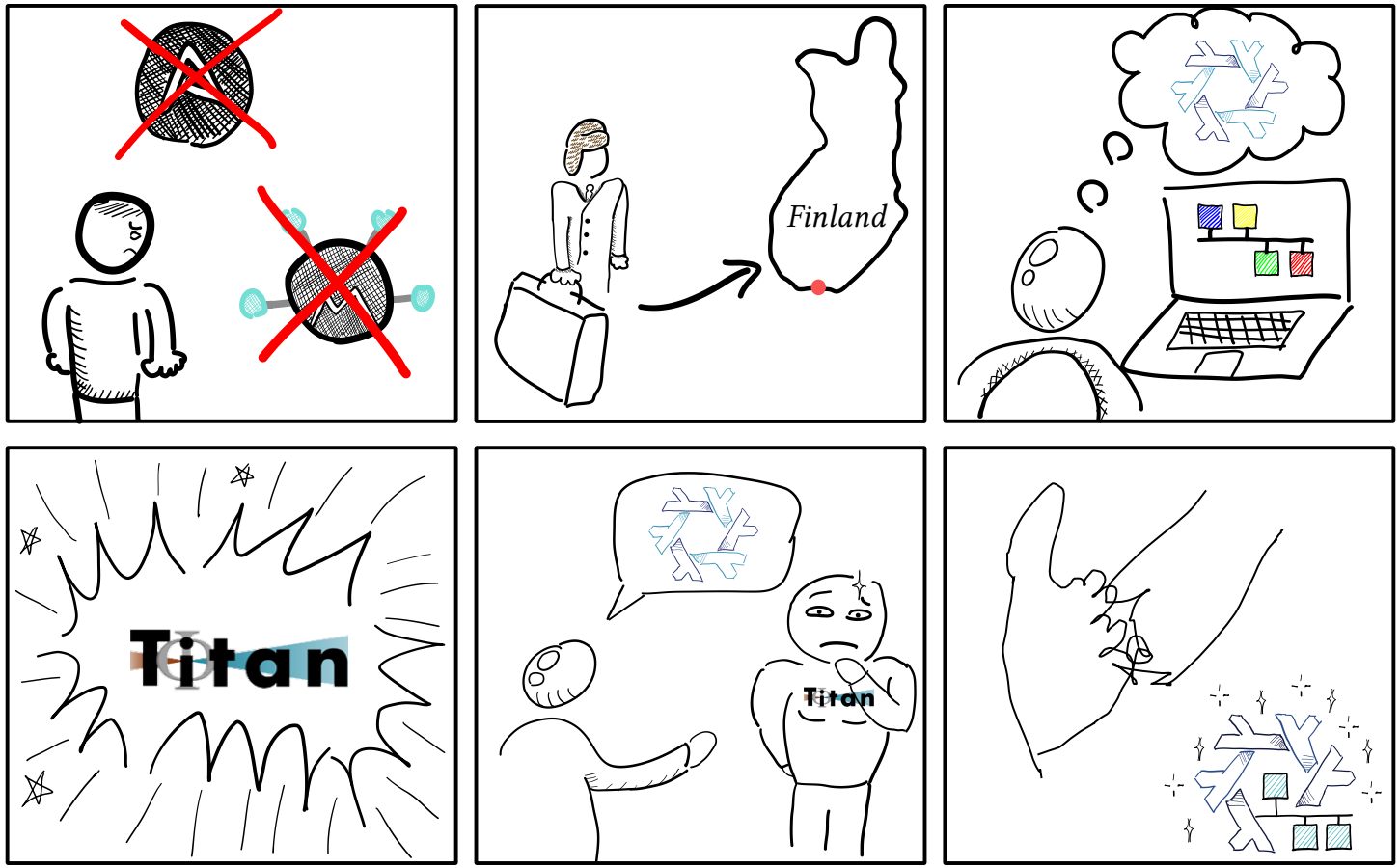


Figure 7: the origin story of **EPNix**

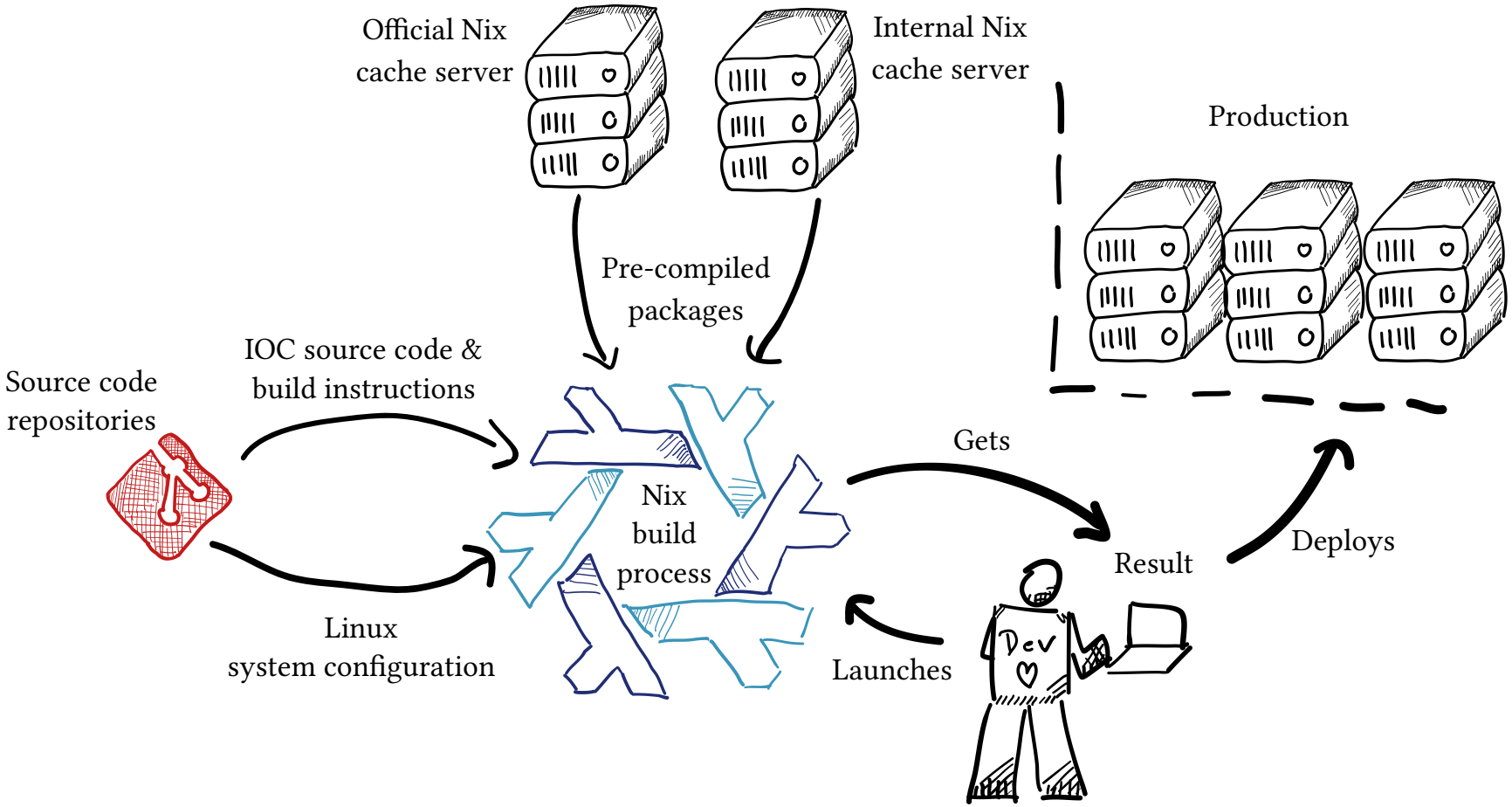


Figure 8: **EPNix**, briefly

3. Consequences

Onboarding colleagues

Takes much more time than expected.

- Especially because everyone is overbooked
- Creating tutorials, organizing workshops, etc., takes lots of time

Advantages we actually care about

Advantages we actually care about

- **People can upgrade their development when they want**

Advantages we actually care about

- **People can upgrade their development when they want**
- Linux system configuration is declared  **Git** repositories

Advantages we actually care about

- **People can upgrade their development when they want**
- Linux system configuration is declared  **Git** repositories
- We can import commonly used configuration

Advantages we actually care about

- **People can upgrade their development when they want**
- Linux system configuration is declared  **Git** repositories
- We can import commonly used configuration
- Reproducibility:
  - Production is the same as in development
  - We can pick up projects that are a few years old

Advantages we actually care about

- **People can upgrade their development when they want**
- Linux system configuration is declared  **Git** repositories
- We can import commonly used configuration
- Reproducibility:
  - Production is the same as in development
  - We can pick up projects that are a few years old
- No need to deploy supporting infrastructure

Advantages we actually care about

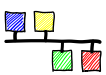
- **People can upgrade their development when they want**
- Linux system configuration is declared **Git** repositories
- We can import commonly used configuration
- Reproducibility:
  - Production is the same as in development
  - We can pick up projects that are a few years old
- No need to deploy supporting infrastructure
- Works for physical machines, virtual machines, and embedded

Advantages we actually care about

- **People can upgrade their development when they want**
- Linux system configuration is declared  **Git** repositories
- We can import commonly used configuration
- Reproducibility:
  - Production is the same as in development
  - We can pick up projects that are a few years old
- No need to deploy supporting infrastructure
- Works for physical machines, virtual machines, and embedded
- **We can deploy to machines disconnected from the Internet**

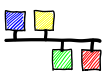
Maintenance cost

Honestly pretty low:

- the  **EPICS** ecosystem moves pretty slowly
-
-
-
-

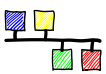
Maintenance cost

Honestly pretty low:

- the  **EPICS** ecosystem moves pretty slowly
- I can upgrade and make breaking changes as I want
-
-
-

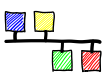
Maintenance cost

Honestly pretty low:

- the  **EPICS** ecosystem moves pretty slowly
- I can upgrade and make breaking changes as I want
- I can leave  **EPNix** untouched if I don't have the time
-
-

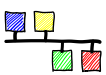
Maintenance cost

Honestly pretty low:

- the  **EPICS** ecosystem moves pretty slowly
- I can upgrade and make breaking changes as I want
- I can leave  **EPNix** untouched if I don't have the time
- Integration tests are easy to write, which makes updates less fearful
-

Maintenance cost

Honestly pretty low:

- the  **EPICS** ecosystem moves pretty slowly
- I can upgrade and make breaking changes as I want
- I can leave  **EPNix** untouched if I don't have the time
- Integration tests are easy to write, which makes updates less fearful
- I'm still basically the only maintainer

Projects using it

- IPHI / TITAN
- SARAF (only for some test bench MCS services)
- SEASON
- AGATA
- PIP2 (test bench)
- Lique4008

- High-Voltage-Engineering
- Dortmund University (old fork)

Some outside NixOS usage

- Some colleagues use or plan to use **NixOS** for their personal home servers.
- The French *interministerial digital department* (DINUM) plan on using **NixOS**

The Ralph slide

Only deploy  **EPNix** in critical parts of your project if you have substantial experience with  **Nix** and  **NixOS**.

The Ralph slide

Only deploy **EPNix** in critical parts of your project if you have substantial experience with **Nix** and **NixOS**.

But you can use **EPNix** in non-critical parts, to try it out and get some experience.

The Ralph slide

Only deploy **EPNix** in critical parts of your project if you have substantial experience with **Nix** and **NixOS**.

But you can use **EPNix** in non-critical parts, to try it out and get some experience.

- Declare developer environment
- Deploy services for test benches
- Do integration tests
- Create Docker images


User
NixOps
Hydra
NixServe
mkDerivation
NixOS
SystemManager
or nix-darwin
HomeManager
DevEnv or
DevShell or
nix shell
nix run
install