Reproducible distributed environments with NixOS Compose

ACM Conference on Reproducibility and Replicability 2024

Dorian GOEPP[1], Fernando AYATS LLAMAS[1], Olivier RICHARD[1], Quentin GUILLOTEAU[2]

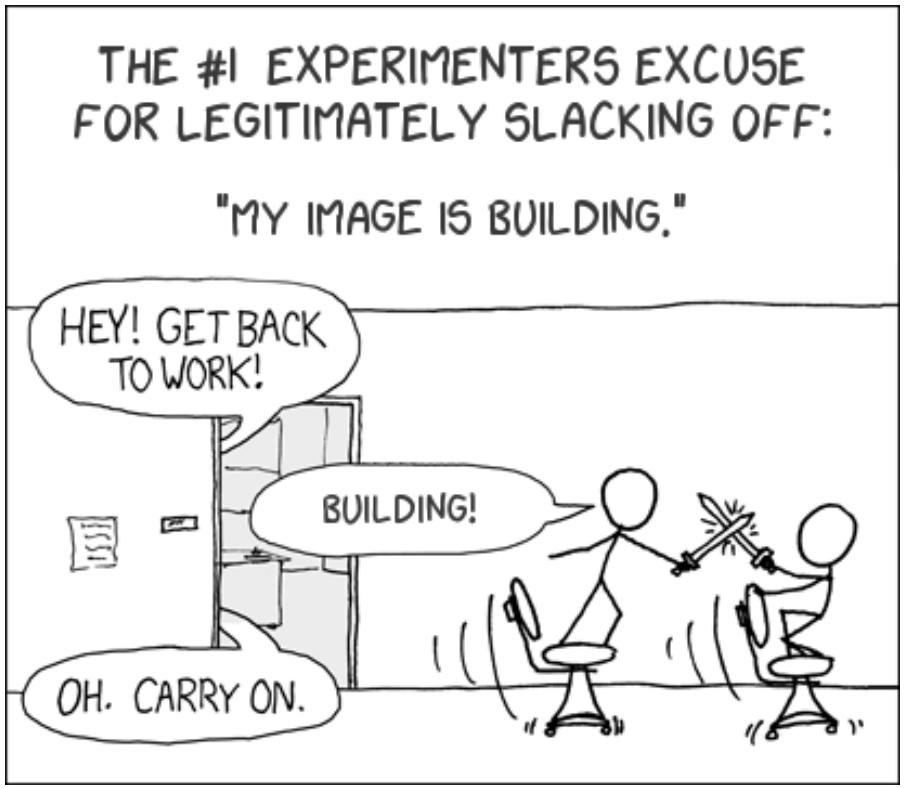
[1] Université Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG

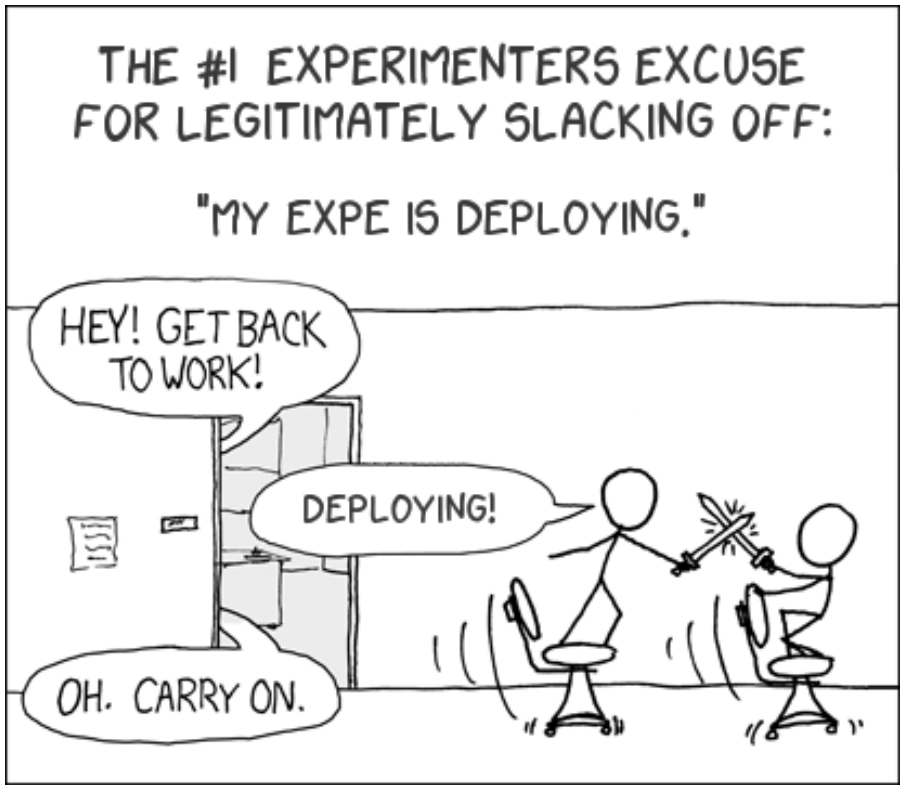
[2] University of Basel, Switzerland

2024-06-18

Motivation

Setting up Distributed Environments for Distributed Experiments
↪ **Difficult**, **Time-consuming** and **Iterative** process





$\Longrightarrow$ **Does not encourage good reproducibility practices**

The Reproducibility Problem

Different Levels of Reproducibility

1. **Repetition**: Run exact same experiment
2. **Replication**: Run experiment with different parameters
3. **Variation**: Run experiment with different environment

↪ **Share the experimental environment and how to build/modify it**

How to share a Software Environment in HPC?

- Containers? ⇝ need `Dockerfile` to rebuild/modify. might not be repo (e.g., `apt update`, `curl`, commit)
- Modules? ⇝ cluster dependent. how to modify?
- Spack? ⇝ share through modules...

Nix and NixOS

The Nix Package Manager

- Functional Package Manager
- Packages are functions
  - Inputs = dependencies
  - Body of function = how to build
- No side-effect
- ($\simeq$) Solves Dependencies Hell
- Reproducible by design

The NixOS Linux Distribution

- Based on Nix
- Declarative approach
- Complete description of the system (kernel, services, pkgs)

How to store the packages?

Usual approach: **Merge them all**

- Conflicts
- `PATH=/usr/bin`

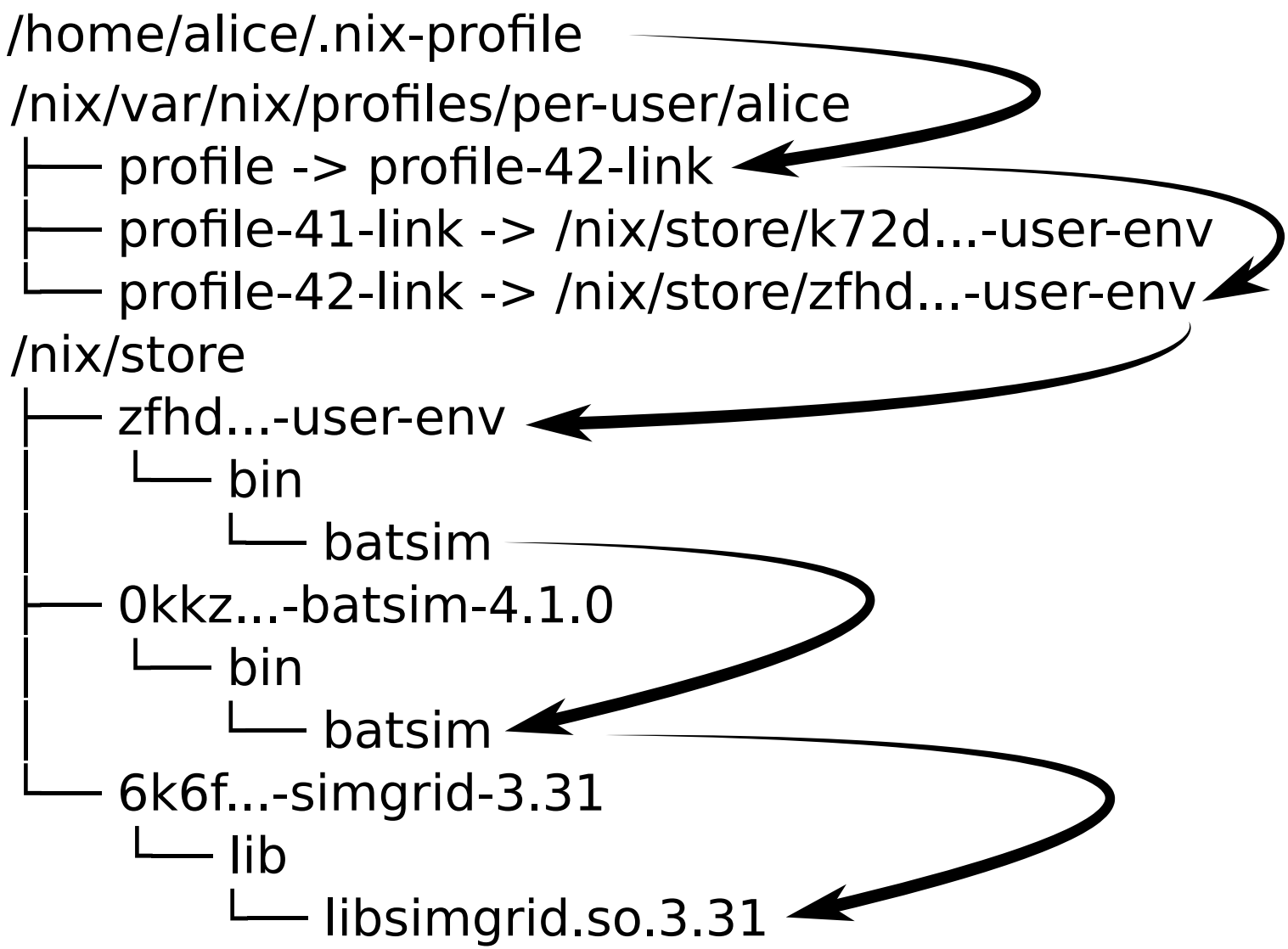
```
/usr
├── bin
│   └── myprogram
└── lib
    ├── libc.so
    └── libmylib.so
```

Nix approach: **Keep them separated**

+ Pkg variation
+ Isolated
+ Well def. `PATH`
+ Read-only

```
/nix/store
├── y9zg6ryffgc5c9y67fcmfdkyyiivjzpj-glibc-2.27
│   └── lib
│       └── libc.so
└── nc5qbagm3wqfg2lv1gwj3r3bn88dpqr8-mypkg-0.1.0
    ├── bin
    │   └── myprogram
    └── lib
        └── libmylib.so
```

Nix Profiles



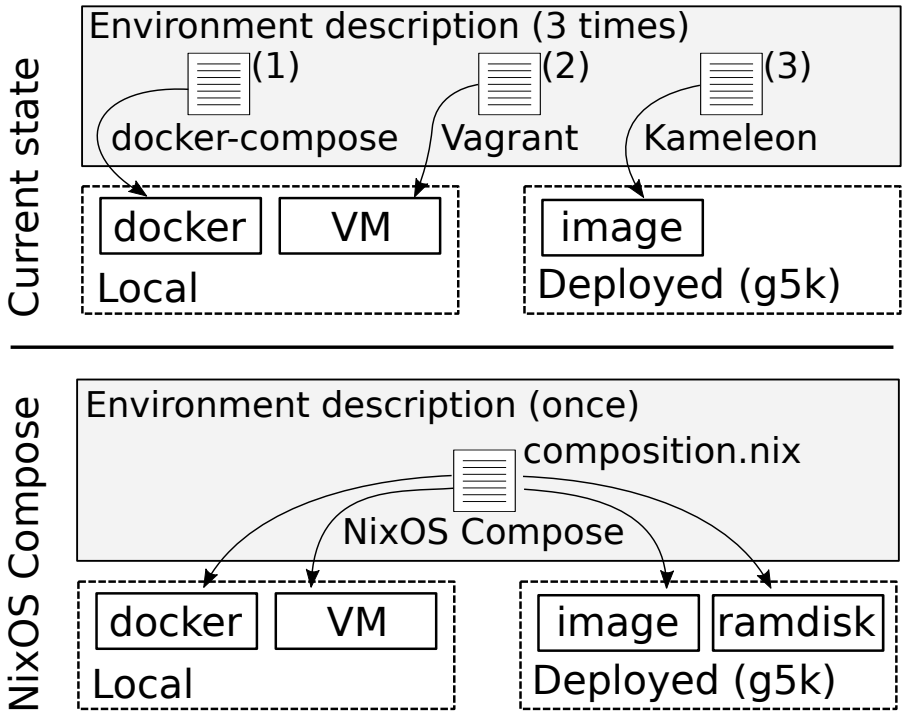
NixOS Compose - Introduction

Goal

Use Nix(OS) to reduce friction for the development of **reproducible distributed environments**

The Tool

- Python + Nix ($\simeq$ 4000 l.o.c.)
- One Definition
  $\hookrightarrow$ Multiple Platforms
- Build and Deploy
- **Reproducible by design**



NixOS Compose - Terminology

Transposition

Capacity to deploy a **uniquely defined environment** on several platforms of different natures (flavours, see later).

Role

Type of configuration associated with the mission of a node.
Example: One Server and several Clients.

Composition

Nix expression describing the NixOS **configuration of every role** in the environment.

NixOS Compose - Composition Example: K3S

```
{ pkgs, ... }:
let k3sToken = "df54383b5659b9280aa1e73e60ef78fc";
in {
  nodes = {
    server = { pkgs, ... }: {
      environment.systemPackages = with pkgs; [
        k3s gzip
      ];
      networking.firewall.allowedTCPPorts = [
        6443
      ];
      services.k3s = {
        enable = true;
        role = "server";
        package = pkgs.k3s;
        extraFlags = "--agent-token ${k3sToken}";
      };
    };
    agent = { pkgs, ... }: {
      environment.systemPackages = with pkgs; [
        k3s gzip
      ];
      services.k3s = {
        enable = true;
        role = "agent";
        serverAddr = "https://server:6443";
        token = k3sToken;
      };
    };
  };
}
```

Role

Packages

Ports

Services

NixOS Compose - Flavours = Target Platform + Variant

`docker` - local and virtual

Generate a `docker-compose` configuration.

`vm` - local and virtual

In memory QEMU Virtual Machines.

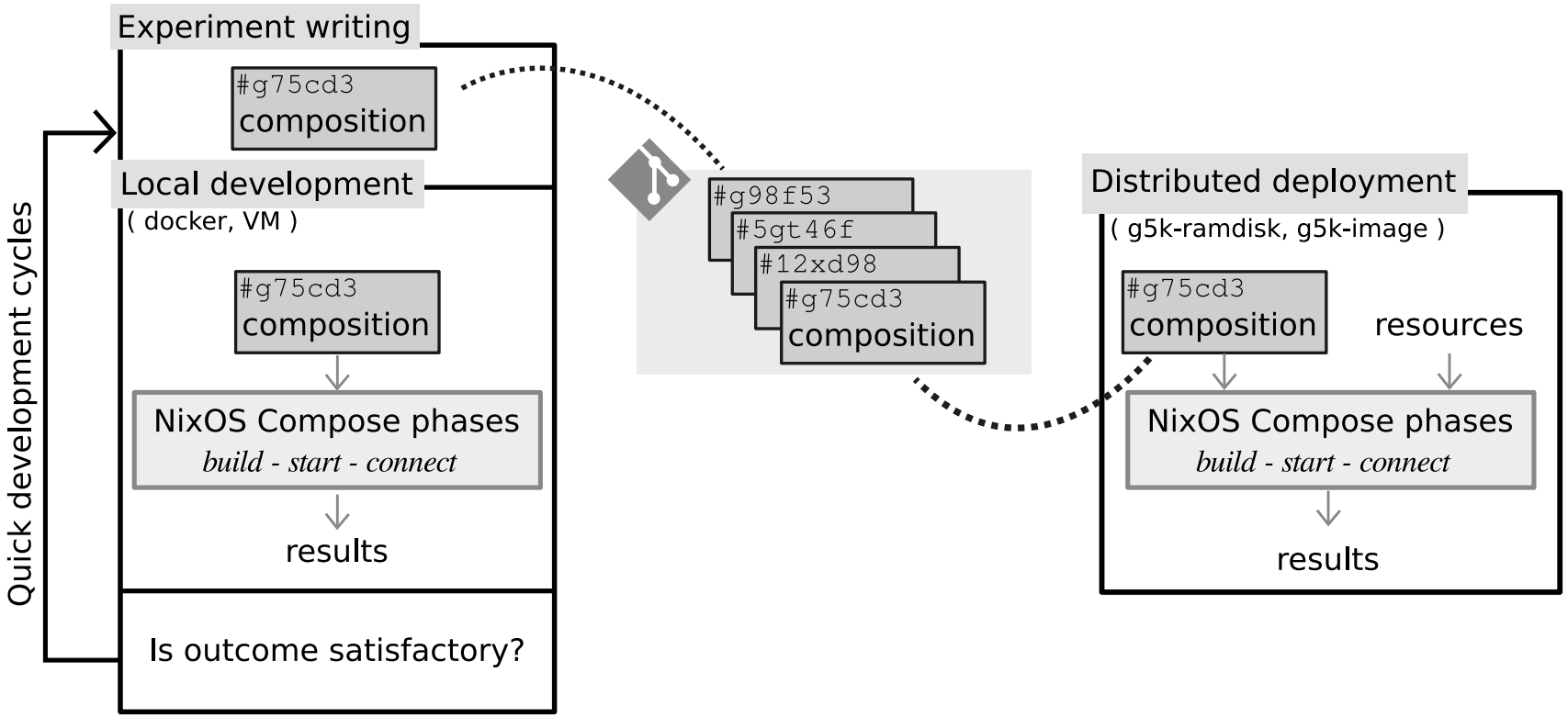
`g5k-nfs-store` - distributed and physical

`initrds` deployed in memory without reboot on G5K (via `kexec`).

`g5k-image` - distributed and physical

Full system tarball images on G5K via Kadeploy.

NixOS Compose - Workflow



NixOS Compose - Workflow for experiment setup

Step 1: Setting up the environment

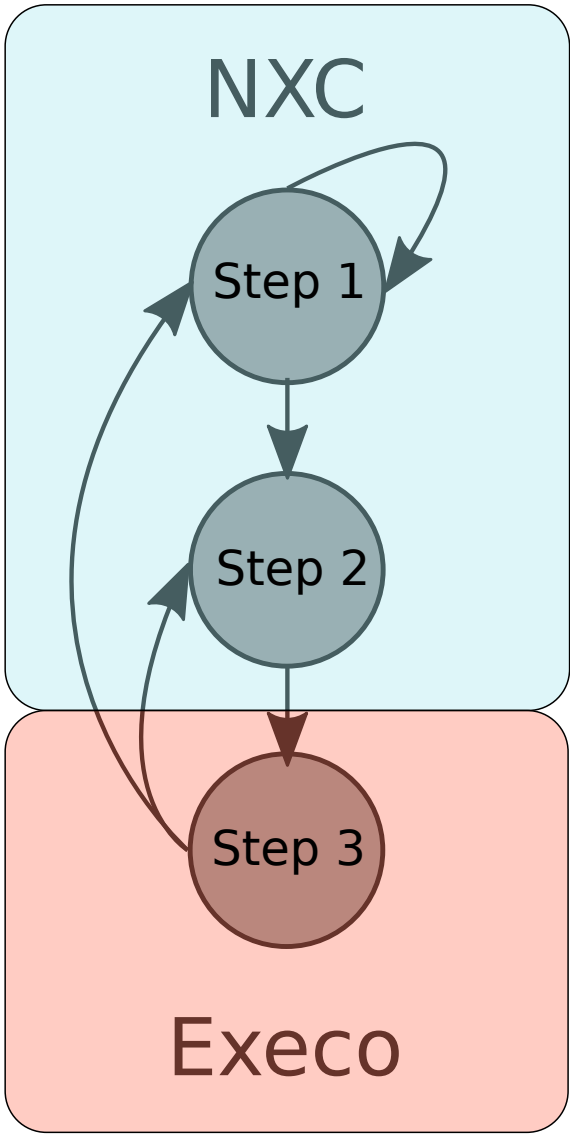
- Add the needed packages
- Define systemd services

Step 2: Setting up experimental bricks

- Gather command lines as scripts
- Package those scripts

Step 3: Script the experiment

- NXC provides an interaction with Execo
- Use Execo to use the "bricks"
- Integrate with Workflow Manager



Your turn

Take Home Message

NXC helps with setting up reproducible distributed experiments

What will you do now?

1. Get a Grid'5000 account
2. Install NixOS Compose
3. Small demonstration by us
4. Get familiar with concepts
5. Setup environment for testing NFS performances
6. Hackathon?

https://tinyurl.com/ACMREP24-NXC