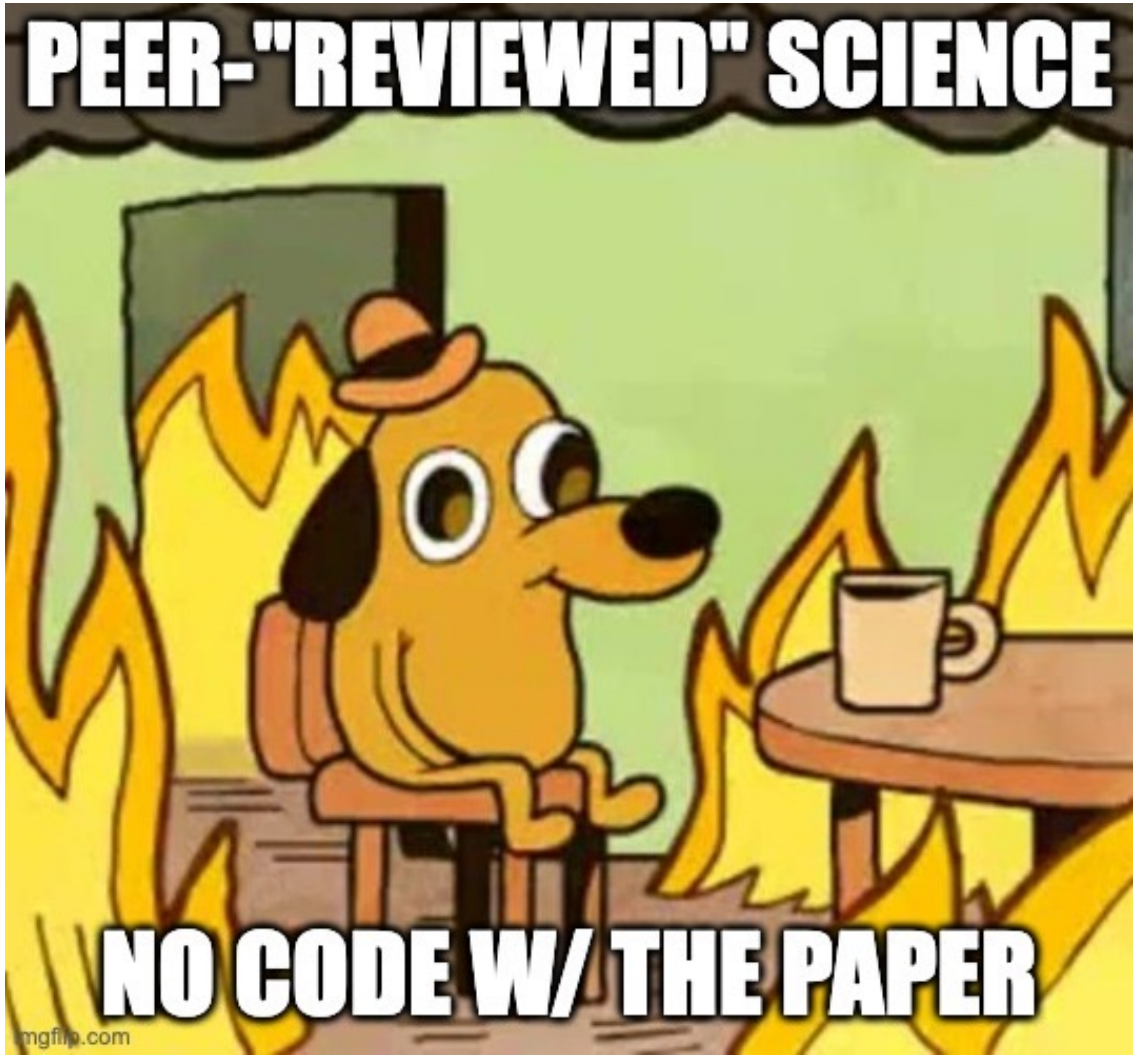
Make reproducible, declarative and reliable systems.

Plan for today

- What is Nix, Nixos ?
- Let’s build something
- What problem does it solves ?
- What compromises are made when using Nix.

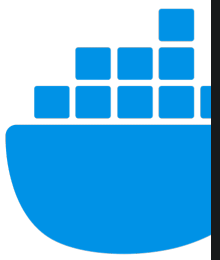
The problem of reproducibility



Sure my code is in my paper
The code

Requirements.txt

Sure my code is in my paper
The c



Dependencies

C++

- C++11 capable compiler (tested with GCC/G++ 4.8.4)
- SimGrid built with graphviz support (build script provided, tested v3.13)
- CMake (for building SimGrid)

Python

- python 2.7 or 3.4+
- setuptools
- Cython
- numpy
- networkx

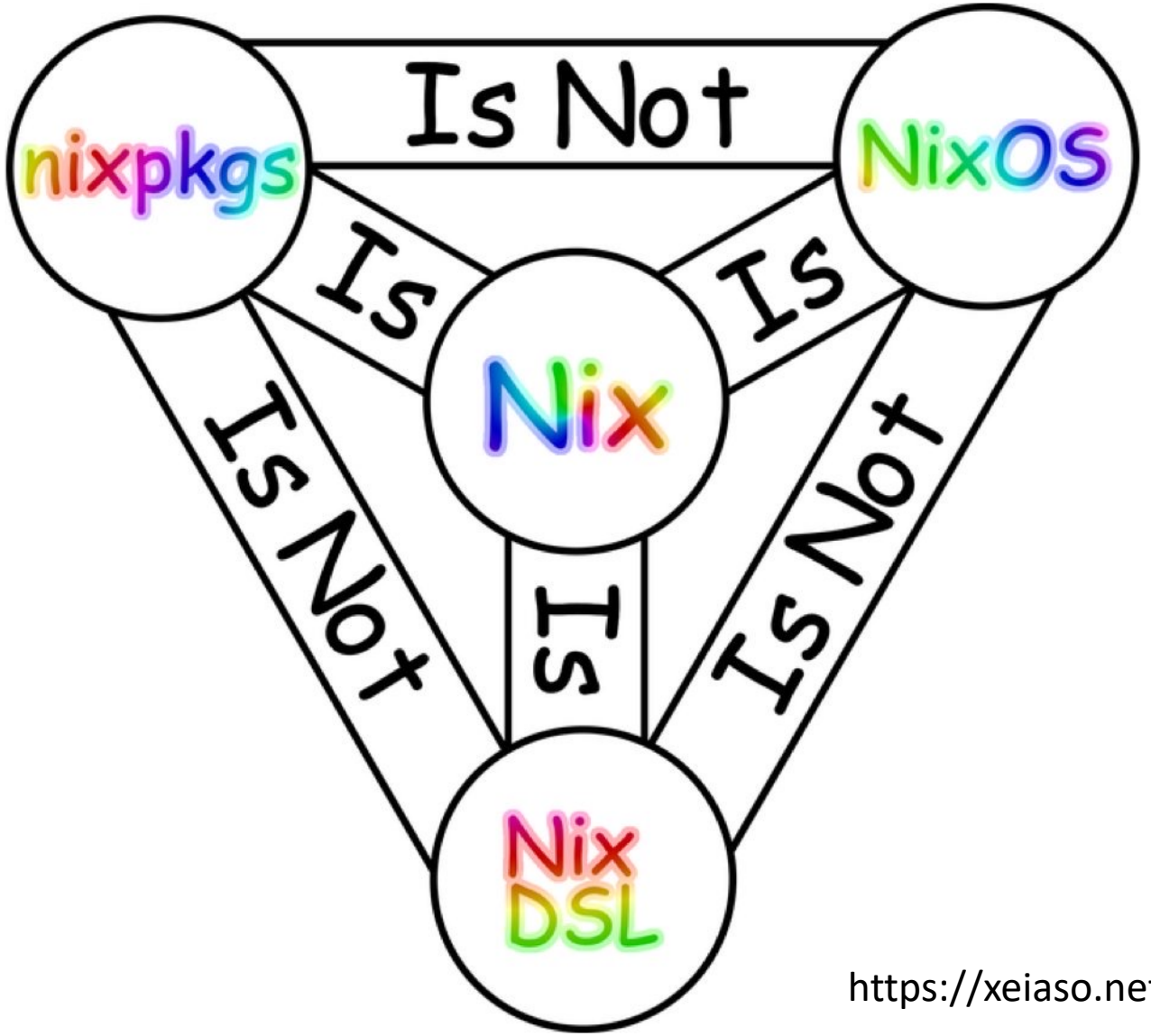
Build instructions

Ubuntu 14.04+

Install system dependencies (list is not full):

```
sudo apt-get install libboost-context-dev libboost-program-options-dev libboost-filesystem-dev
```

@juvelo@cohost.com



https://xeiaso.net/talks/2024/nix-docker-build/

Nix is a functional language, that builds packages and systems from deterministic inputs.

Repo containing ALL packages

Gnu parallel will be available in the dev environment

```
{
  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-unstable";
    flake-utils.url = "github:numtide/flake-utils";
  };

  outputs = inputs: (inputs.flake-utils.lib.eachDefaultSystem (
    system: let
      pkgs = import inputs.nixpkgs {
        inherit system;
      };
    in {
      devShells.default = pkgs.mkShell {
        packages = with pkgs; [
          parallel
        ];
      };
    }
  ));
}
```

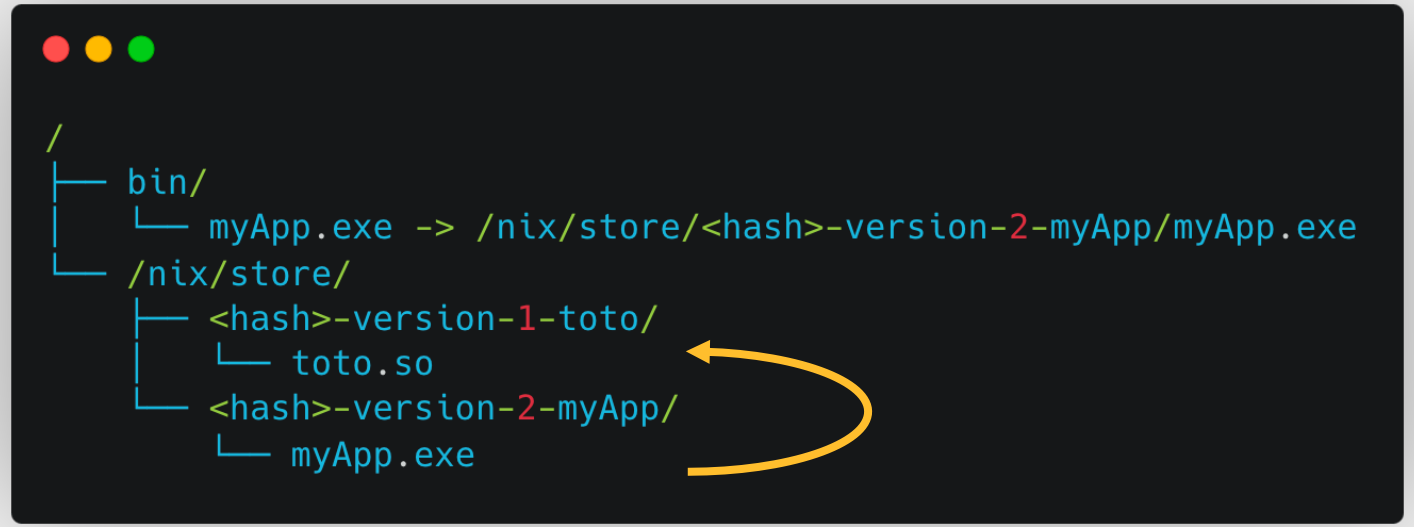
For x86, arm, etc.

Here is what a normal application, on a normal Linux looks like

myApp will look for “/lib” to find its dependency toto.so

Nix patches and/or wraps program so they find their dependencies from a single location

Standard Linux



NixOS

This approache uses hashes of packages derived from inputs and build steps to create unique entries

And now, if all packages have their inputs at a unique location, anyone can use different version of the same package without issues

Here, whatever happens in e.g. a local devenv, you will not have to modify the state of the whole system

Demo time ☺

- We will make a dev env
- Then we will package an application
- Then we will make it a docker image
- Then we will make a VM that runs Kubernetes and launches our image on startup
- https://github.com/VolodiaPG/tutoNixSED

For this, we will use Flakes, a mechanism allowing simple manipulation and containment of packages

Think NPM with package.json that lists the dependencies to install and the LOCK file to freeze versions

Nix is used for a variety of tasks from packaging to remote deployments.

- Reproducible development environments.
- Easy installation of software over URLs.
- Easy transfer of software environments between computers.
- Declarative specification of Linux machines.
- Reproducible integration testing using virtual machines.
- Avoidance of version conflicts with already installed software.
- Installing software from source code.
- Transparent build caching using binary caches.
- Strong support for software auditability.
- First-class cross compilation support.
- Remote builds.
- Remote deployments.
- Atomic upgrades and rollbacks.
- + others (*ask me at the end of the presentation*)

Nix, as any tool, makes compromises

- It is an opinionated wrapper: there will always be applications that do not deliver in its format (or are not present)
- Its functional language makes sense, however it requires experience and can create a knowledge gap in a team
- There is boilerplate to know
- By default, you have to enable some commands for flakes, and for parallelism

The biggest one is the documentation is scarce… because code is documentation

Some resources to get started:

- https://nixos.org/guides/nix-pills/
- https://nix.dev/
- https://search.nixos.org/packages (all packages + link to sources)
- https://search.nix.gsc.io (search existing code)
- https://github.com/ (filter with Nix as language)
- https://dataswamp.org/~solene/ (good blog)
- https://lantian.pub/ (that guy is just crazy good)
- https://xeiaso.net/blog (blog)
- https://www.phind.com (AI is actually pretty useful there)
- https://github.com/VolodiaPG/faas_fog (my repo)

Hopefully that will motivate you to go deeper into the reproducible way or even Nix

Feel free to ask questions

The name Nix is derived from the Dutch word *niks*, meaning nothing; build actions do not see anything that has not been explicitly declared as an input.
- Nix: A Safe and Policy-Free System for Software Deployment