Reproducable build and dev environments with Nix

Pin everything everywhere

Mark Dickie

Introduction

Mark Dickie

- Background in data network, telecoms, infrastructure and build systems.
- Passionate about best practice, repeatability and knowledge sharing.
- Generally spends waking hours working with some kind of tech
- Gamer and private pilot also

Overview

- Define Nix
- Show a small project
- Demonstrate a dev shell using a Nix flake
- Demonstrate a container image using Nix flakes
- Talk about build pipelines with Nix
- Demonstrate a Nix package
- Answer questions

Definition

Nix is an overloaded term

Nix Language

Declarative, purely functional, lazyly evaluated, dynamically typed, domain specific language for package development.

Nix Package Manager

Cross platform package manager for UNIX like systems (Linux, Darwin) with some support for Windows. Packages are source based with binary caching.

NixOS Linux Distro

Linux distribution built using Nix for package management and definition of the system configuration declaratively. It is immutable and has an atomic update model.

The traditional way

The project

Small C application which needs to be built and the artifacts deployed.

Challenges

We need to have all the correct compilers, libraries and other dependencies on every developer workstation and every build runner for the build and deploy pipelines.

Dev Shell with Nix

Dependencies

- Nix package manager
- direnv

Get started

https://github.com/the-nix-way/dev-templates/tree/main

```
nix flake init -t github:the-nix-way/dev-templates#c-cpp
```

Dev Shell with Nix

- Demo

```
~/Documents/nix-presentation  main ≢
```

Dev Shell with Nix

Benefits

- Dependencies are part of the repository
- Versions of all dependencies are pinned
- Onboarding of developers becomes easy
- Makes environments modular and re-usable

Generate a container

- Demo

```
~/Documents/nix-presentation  main ≢
```

Generate a container

Benefits

- Can be used for development or build pipelines
- In webapp and similar contexts this could also be used as a deployable production environment
- Multiple container definitions could be contained in the flake with shared code

Build pipelines

Multiple approaches

- Build inside the generated container
- Run the devShell from the flake to build
- Use nix to build the package directly

Build a Nix package

- Demo

```
~/Documents/nix-presentation  main ≢ |
```

Nix Details

- Nix Store
- Binary Caching
- Garbage Collection

Questions?

Ask away!

Links

Presentation

https://github.com/deephack1982/nix-presentation

More information about Nix

https://www.nixos.org

More information about direnv

https://direnv.net/

My BlueSky

https://bsky.app/profile/deephackontech.bsky.social