CONQUER CABAL HELL WITH NIX

@steshaw

OVERVIEW OF TALK

1. Purely Functional Package Systems
2. Overview of Nix
3. How to use Nix with Haskell

PURELY FUNCTIONAL PACKAGE SYSTEMS

FEATURES

- multiple variants of a package at the same time (i.e. side by side)
- immutable packages
- atomic upgrades/rollbacks
- mutiple environments
- usable as non-root

EXAMPLES

- Nix
- Zero Install
- Listaller (was Autopackage)

NIX

COMMANDS

Find packages

```
$ nix-env --query --available 'hello*'
$ nix-env -qa 'hello*'
```

Install a package

```
$ nix-env --install hello
$ nix-env -i hello
```

Remove a package

```
$ nix-env --uninstall hello
$ nix-env -e hello
```

COMMANDS (CONTINUED)

Alternatively rollback

```
$ nix-env --rollback
```

Upgrade your packages

```
$ nix-env --upgrade
```

ACCELERATING HASKELL DEVELOPMENT WITH NIX

demo time

NOTES (WORK IN PROGRESS)

NIX FEATURES

- Add more Nix features from http://nixos.org/nix/about.html
- Diagrams!

ARCHITECTURE OF NIX

Under the hood a little bit

- /nix/store
- Mention the crazy linker thing.
- How hashing works. Why binary substiution works.
- A -> B. What are the inputs?
  - What inputs?
  - What outputs?
  - hashing for the store
- Caching/Memoisation.

NIX COMMANDS

- Installing haskell tools like ghc (different versions) and nix2cabal.
- Usage of nix-shell
- Usage of nix2cabal

ACCELERATING HASKELL DEVELOPMENT WITH NIX

WHERE ARE WE NOW?

- Using cabal sandboxes.
- Perhaps some shared sandboxes.
- Waiting for builds is no fun.
- Wasting time building `lens` for each of your projects that uses it is not good.
- Let’s not accept the status quo.
- One option is to use Halcyon – a build cache for Cabal.

WHAT’S NOT GOOD?

- Long build times.
- Building the same dependencies over and over again in different sandboxes.
- These sandboxes could be on your machine or your team members machine.
- Or on the build box.
- There is wastage of time but also of disk space.
- With SSDs, disk space isn’t as cheap as it used to be.

DEMO

- Instant `lens` environment.
- Instant `reflex` environment with tryreflex.
  https://github.com/ryantrinkle/try-reflex
- Work through hutton-razor.

DOWNSIDES OF NIX.

- Still early (but you'd be getting in at a great time)
- Written in C++ and Perl (HT Raahul Kumar for pointing out that it's C++ not C).
- Perhaps there's hope for a Haskell implementation — https://github.com/jwiegley/hnix.

NIX WORKSHOP AT HACK NIGHT

- Get set up with NixOS (and perhaps Nix on Mac/Linux/*BSD if you’re more adventurous).
- Learn how to set up a modern Haskell development environment.
- Hopefully, learn how to work on sources to multiple dependencies in your tree.

REFERENCES

- Ollie Charles
  - How I develop with Nix
  - http://wiki.ocharles.org.uk/Nix
- Peter Simons
  - Nix loves Haskell — slides — slides.md
  - Haskell User Guide for Nixpkgs
- https://nixos.org/nix/manual/