Nix: A new way to manage packages

Jack Garner

What is Nix?

Nix is an ecosystem

- Nix (the daemon)

Nix is an ecosystem

- Nix (the daemon)
- Nix (the language)

Nix is an ecosystem

- Nix (the daemon)
- Nix (the language)
- Nix-* commands (official and unofficial)

Nix is an ecosystem

- Nix (the daemon)
- Nix (the language)
- Nix-* commands (official and unofficial)
- Nixpkgs

Nix is an ecosystem

- Nix (the daemon)
- Nix (the language)
- Nix-* commands (official and unofficial)
- Nixpkgs
- **Nixos**

But why?

What makes Nix* cool

- Atomic upgrades and installs
- Partial upgrades through isolated packages
- Easy rollbacks
- Rootless package operations
- Source based with a binary cache
- Cross platform (theoretically)
- Version pinning of specific packages
- Reproducible, sandboxed build environments
- Can output docker files, VM's, file systems, etc.
- Supports cross compilation

What makes Nix* less cool

- Documentation

What makes Nix* less cool

- Documentation
- The language has a learning curve

What makes Nix* less cool

- Documentation
- The language has a learning curve
- Documentation

What makes Nix* less cool

- Documentation
- The language has a learning curve
- Documentation
- Documentation

What makes Nix* less cool

- Documentation
- The language has a learning curve
- Documentation
- Documentation

Hopefully this talk will help you understand Nix

Let's make a package!

Nix (the daemon)

- Accepts derivations and builds them
- Gives every package (derivation) a hash based on the hashes of inputs
- Stores built packages in /nix/store/hash-packageName/{bin/share/lib/include/*}

Nix (the daemon)

- Accepts derivations and builds them
- Gives every package (derivation) a hash based on the hashes of inputs
- Stores built packages in /nix/store/hash-packageName/{bin/share/lib/include/*}

What about the Linux Filesystem Hierarchy?

Nix uses a combination of symlinks, environment variables, and patched low level tools to make software run in this environment. If those fail for a package, Nix can create a fake hierarchy which only that package sees.

What is a derivation?

A derivation is a key value pair with the following keys:

- system = “x86_64-linux” (or darwin or . . .)
- name = “Package name”
- builder = “command to build package”
- args = [“Args” “To pass” “to the builder”]
- A few really uncommon ones. . .

Any other keys become environment variables in the sandbox.

What makes derivations cool?

- Builders can't depend on any files or variables not mentioned in the derivation
- Any files mentioned in the derivation become packages
- Any packages needed by the derivation get built first
- Derivations are reproducible, complete build instructions

What makes derivations cool?

- Builders can't depend on any files or variables not mentioned in the derivation
- Any files mentioned in the derivation become packages
- Any packages needed by the derivation get built first

- Derivations are reproducible, complete build instructions
- Derivations can be copied between machines

./default.nix

```
with (import <nixpkgs> {});
derivation {
  name = "hello";
  builder = "${bash}/bin/bash";
  args = [ ./hello_builder.sh ];
  inherit gnumake gcc coreutils gawk gnused gnugrep;
  binutils = binutils-unwrapped;
  src = fetchTarball {
    url = https://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz;
    sha256 = "...Not Important...";
  };
  system = builtins.currentSystem;
}
```

./hello_builder.sh

```
export PATH="$gcc/bin:$gnumake/bin:$coreutils/bin:..."
$src/configure --prefix=$out
make
make install
```

Stdenv

```
with (import <nixpkgs> {});
stdenv.mkDerivation {
  pname = "hello";
  version = "2.10";
  src = fetchTarball {
    url = https://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz;
    sha256 = "...Not Important...";
  };
}
```

Install it

- Run `nix build -f default.nix`
- Take a look in `result` to makes sure it built correctly
- Run `nix-env -f default.nix -i hello`

Install it

- Run `nix build -f default.nix`
- Take a look in `result` to makes sure it built correctly
- Run `nix-env -f default.nix -i hello`

What just happened?

Symlinks!

Removing a package!

Easy right?

```
nix-env --uninstall hello
```

Garbage Collection

```
nix-collect-garbage
```

Garbage Collection

`nix-collect-garbage`

`nix-collect-garbage -d` Once you're certain your new profile works

But why?

Nixpkgs

What is it?

- A (git) repository of packages used by Nix/Nixos by default

What is it?

- A (git) repository of packages used by Nix/Nixos by default
- Has versioned releases and an unstable rolling branch

What is it?

- A (git) repository of packages used by Nix/Nixos by default
- Has versioned releases and an unstable rolling branch
- Contains packages you would expect from a package manager

What is it?

- A (git) repository of packages used by Nix/Nixos by default
- Has versioned releases and an unstable rolling branch
- Contains packages you would expect from a package manager
- A collection of modules for Nixos

What is it?

- A (git) repository of packages used by Nix/Nixos by default
- Has versioned releases and an unstable rolling branch
- Contains packages you would expect from a package manager
- A collection of modules for Nixos
- Also a TON of library functions for packaging and modules

Nixos

What is it?

Nix has a pretty cool way of packaging. . .

What is it?

Nix has a pretty cool way of packaging. . .

What if your entire OS were a package!

What is it?

Nix has a pretty cool way of packaging. . .

What if your entire OS were a package!

Nixos is some bootstrapping scripts + nixos-rebuild + Nixpkgs

Low level primitives

A few low level nix functions + scripts dynamically create the right directory structure

- system.build.toplevel = The contents of the system profile
- system.build.installBootLoader = A script run on `sudo nixos-rebuild boot|switch`
- environment.etc = dictionary of files to symlink in `/etc`
- environment.pathsToLink = directories to symlink in `$systemProfile/sw`

What might a config look like?

Nix-shell

Why you should care

Since nix runs on Linux/OSX, you can easily share development/build environments

Given a shell.nix file, you can be sure a user will have all dependencies and/or dev dependencies.