Very brief introduction to Nix

Ryan Watkins

August 31, 2021

Table of contents

Classical package management

- Packaging is difficult
- Made harder by sharing dependencies and transient modus operandi
- Inherent complexity in dependency resolution
- Upgrades can always lead to bad state
- Non-deterministic builds
- Reliance on binary blobs
- No rollbacks (atomicity)
- Lack of configuration

What Nix provides

- Configuration
- Atomic operations on package management
- Lots of packages
- Deterministic builds checked by Hydra (build server)
- Multiple versions of a package (for ex. Python3.5, Python 3.6 etc.)

Nixpkgs

- Packages declared in github
- Unstable, stable etc. similar to debian
- Easily modifiable

NixOS

- Linux distribution using Nix package manager and the system configuration system
- Mostly open source but has easy means of hardware detection and unfree software
- Extremely easy to replicate systems bitwise (i.e. exactly the same)

Example system configuration

```
{ config, pkgs, ... }:
let
  unstable = import
    (builtins.fetchTarball
        https://github.com/nixos/nixpkgs/tarball/master)
    { config = config.nixpkgs.config; };
in
{
  nixpkgs.config.allowUnfree = true;
  environment.systemPackages = with pkgs; [
    unstable.vim
    unstable.wget
    unstable.emacs27
    ...
 ];
  imports =
    [ # Include the results of the hardware scan.
      ./hardware-configuration.nix
    ];
}
```

Example service configuration

```
services.postgresql = {
  enable = true;
  package = pkgs.postgresql_10;
  enableTCPIP = true;
  authentication = pkgs.lib.mkOverride 10 ''
    local all all trust
    host all all ::1/128 trust
  '';
  initialScript = pkgs.writeText "backend-initScript" ''
    CREATE ROLE falcon WITH LOGIN PASSWORD 'falcon'
        CREATEDB;
    CREATE DATABASE falcon;
    GRANT ALL PRIVILEGES ON DATABASE falcon TO falcon;
  '';
};
```

Nix Shell

- Spin up 'virtual environment' ad-hoc
- If you need packages to develop on something or temporarily this is very useful
- Keeps the system from being cluttered with packages that were only temporarily needed.

```
nix-shell -p python39 python39Packages.pip
    python39Packages.virtualenv
```

How is atomicity maintained?

- A read-only section of the filesystem contains all packages
- The nix daemon writes to this via for example:

```
nix-env -iA nixos-unstable.neovide
```

- This would install neovide and in-term write to ‘/nix/store/some-given-hash/neovide‘
- Then; a symlink would be made for the binary towards ‘ /.nix-profile/bin/neovide‘

Talks/papers

- A Purely Functional Linux Distribution - NixOS https://nixos.org/ eelco/pubs/nixos-icfp2008-submitted.pdf
- Tim Steinbach- Sane System Management with NixOS - I Do Declare!- lambdaconf 2019 - https://www.youtube.com/watch?v=_LDzO5_d1a0

End

- Any questions?