Nix

Antoine R. Dumont (@ardumont)

07/04/2019

Outline

Topic

Enforces functional approach to package management

Declarative

Let the machines work or complain!

Lazy

Only compute and install what you ask for!

Pure

Idempotency!

Topic

Same as other package managers:

- install programs/libraries (with a *unified* DSL)
- integrated in a distribution: NixOS
- home-manager: Reproduce your home!

Topic

- *Unified* DSL
- Multi-user support
- Source/binary model (binary cache, build-farm, hydra, etc. . .)
- Upgrades/rollback environment
- No versioning clash in dependencies between programs/libraries
- Reproducibility
- Documented (mostly centralized today)
- Community (open discussion on tracker/irc, open PR, etc. . .)
- Tools: nix repl, nix search, nix-build, nix-shell, nix-collect-garbage, . . .

Topic

- Steep learning curve, quite some new notions (nix-channel, derivation, overlays. . .)
- Disk Space ($\longrightarrow$ *nix-collect-garbage*)
- Inconsistency in between environments or within (haskell, python, etc. . .)
- Not unified tool interface (nix build vs nix-build?. . .)

Topic

Program

```
{ stdenv, fetchurl }:

stdenv.mkDerivation rec {
  name = "hello-${version}";
  version = "2.10";
  src = fetchurl {
    url = "mirror://gnu/hello/${name}.tar.gz";
    sha256 =  "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kz...";
  };
  doCheck = true;
  meta = with stdenv.lib; {
    description = "Produces a familiar friendly greeting";
    homepage = https://www.gnu.org/software/hello/manual/;
    license = licenses.gpl3Plus;
    maintainers = [ maintainers.eelco ];
    platforms = platforms.all;
  };
}
```

Service

```
{ config, lib, pkgs, ... }:

with lib;
let cfg = config.services.xserver.windowManager.fluxbox;
in {
  options = {
    services.xserver.windowManager.fluxbox.enable =
      mkEnableOption "fluxbox";
  };
  config = mkIf cfg.enable {
    services.xserver.windowManager.session = singleton {
      name = "fluxbox";
      start = ''
        ${pkgs.fluxbox}/bin/startfluxbox &
        waitPID=$!
      '';
    };
    environment.systemPackages = [ pkgs.fluxbox ];
  };
}
```

Library

TODO

Topic

Imperative

```
$ hello
The program ‘hello’ is currently not installed.
It is provided by several packages. You can install
it by typing one of the following:
  nix-env -iA nixos.hello
...
$ nix-env --install hello
installing ’hello-2.10’
these paths will be fetched (0.04 MiB download, 0.19 MiB
unpacked):
  /nix/store/gdh8165b7rg4y53v64chjys7mbbw89f9-hello-2.10
copying path ’/nix/store/gdh8165b7rg4y53v64chjys7mbbw89f9
-hello-2.10’
from ’https://cache.nixos.org’...
building ’/nix/store/39c7sm1sn97yd783jyw50bdabq69gfjm-user-
environment.drv’...
created 1656 symlinks in user environment
$ hello
Hello, world!
```

Declarative: config.nix

```
{ pkgs }:

{
  packageOverrides = self: {
    helloEnv = pkgs.buildEnv {
      name = "hello-nix";
      paths = [ pkgs.hello ];
    };
}
```

Then:

```
nix-env --install hello-nix
```

Declarative: home-manager

```
home.packages = [ pkgs.hello ];
```

Then:

```
home-manager switch
```

Declarative: nixos

```
environment.systemPackages = [ pkgs.hello ];
```

Then:

```
sudo nixos-rebuild switch
```

Note: All users have now access to that program

Topic

Imperative

Remove explicitely from user environment:

```
$ nix-env --uninstall hello
uninstalling 'hello-2.10'
$ hello
The program ‘hello’ is currently not installed...
```

Declarative

Rollback to previous generation

```
$ nix-env --rollback
switching from generation 91 to 90
$ hello
The program ‘hello’ is currently not installed...
```