Nix

Fabio Lenherr

07.07.2024

Table of Contents

What is Nix?

- just a programming language
- functional
  - lazy
  - everything is an expression
- turing complete
- made to configure environments
  - native paths
  - tooling for environments -> nixos etc

Basics

- haskell interactive shell
  - nix repl
- evaluate code in file
  - nix-instantiate
- comments: --
- function: argument:body
  - x: 5 + x
  - one argument only
  - more arguments == currying
- native types
  - lists
  - attribute set
  - functions
  - numbers, strings, **paths**, etc
- inbuilt functionality
  - import
  - lib.toString
  - std.mkDerivation

Tricks

let in

```nix
let
  func = x: x * 2;
in {
  func 24
}
-- 48
```

- **only in this scope**

With

```nix
let
  attrs = {
    x = 10;
    b = 55;
    c = 34;
  };
in with attrs; [
  x
  b
  c
];
```

inherit

```nix
let
  x = 1;
  y = 2;
in
{
  inherit x y;
}
```

- copies the values in let to the scope in the curly brackets

Packages

```
{ stdenv
, lib
, pkgs
, fetchFromGitLab
}:

pkgs.python3.pkgs.buildPythonApplication rec {
  pname = "pingpang";
  version = "0.0.1";

  src = fetchFromGitLab {
    domain = "gitlab.globi.org";
    owner = "globi";
    repo = pname;
```

```
15      rev = version;
16      sha256 = "sha.something";
17    };
18
19    nativeBuildInputs = with pkgs; [
20      cargo
21    ];
22
23    buildInputs = with pkgs; [
24      gtk
25    ];
26
27    meta = with lib; {
28      -- homepage, architecture support, help page etc
29    };
30  }
```

NixOS

- a GNU/Linux distribution
- fundamentally different file system design
  - nix store
  - otherwise just like any penguin variant
- only configures and installs system wide programs
  - use home-manager for user-based configuration

Home-Manager

- extension to nixos/nix-darwin
- user level variant of nixos
- available on other distributions
- configures user configuration files

Flakes

- extension to nixos/homemanager
- removes the issue of hashes
  - enables easy updates of inputs
  - lock file
- technically an “experimental” feature

Example

```
{
  description = "dots";
  inputs = {
      nixpkgs.url = "github:NixOs/nixpkgs/nixos-unstable";
      home-manager = {
        url = "github:nix-community/home-manager";
        inputs.nixpkgs.follows = "nixpkgs";
      };
  };

  outputs = { ... }@inputs:
    let
      pkgs = import inputs.nixpkgs {
        system = "x86_64-linux";
```

```
15          config = { allowUnfree = true; };
16        };
17      in {
          nixosConfigurations."something" = inputs.nixpkgs.lib.nixosSystem
18  {
19          specialArgs = {
20            inherit inputs pkgs;
21            mod = ./hardware/something/base_config.nix;
22          };
23          modules = [ ./hardware/hardware.nix ]
24        };
25      };
26  }
```

Overrides

```
new: old:
{
  packages = old.packages.override {
    package = new.newpackage;
  };
}
```

What else is Nix good for?

- servers
  - declarative/reproducible server instantiation
  - ansible on steroids
  - easily convertible to and from docker files/images
- CI/CD
  - declarative and reproducible pipelines
  - no version mismatch due to nix
  - available as a github runner -> nix-run
- dev environment
  - even on multiple operating systems