# Nixology

Yifei Sun

June 28, 2024

## Problem

## Solution

Functions:

```
{ inputs = { ... }; }
```

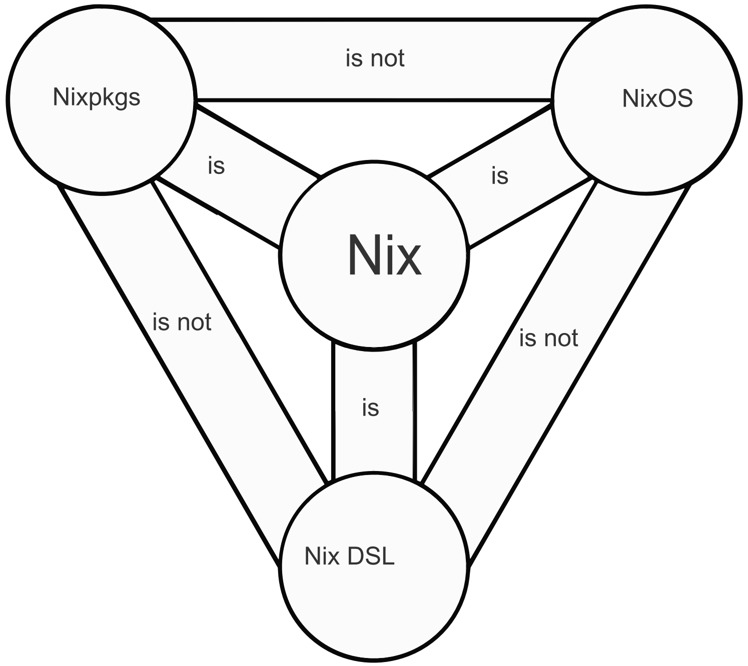
- Dependencies are inputs
- Usually tarballs or git repos
- Pinned and hashed

```
{ outputs = inputs: { ... }; }
```

- Outputs are functions of inputs
- Can be anything
- Lazily evaluated

## Trinity

- Nix - the package manager
- Nix - the DSL
- Nixpkgs - the package collection
- NixOS - the operating system



## Language Basics

Integers:

```
> x = 1 + 1
> x
2
```

Floats:

```
> y = 1.0 + 1.0
> y
2.0
```

Strings:

```
> z = "world"
> "hello ${z}"
"hello world"
```

Attribute sets:

```
> s = { a = { b = 1; }; }
> s.a.b
1
```

## Language Basics

Lists:

```
> [ 1 "2" (_: 3) ]
[ 1 "2" <thunk> ]
```

Recursive attrsets:

```
> rec { x = 1; y = x; }
{ x = 1; y = 1; }
```

Bindings:

```
> let x = 1; in x + 1
2
```

Inherits:

```
> let x = 1; y = x; in
    { inherit x y; }
{ x = 1; y = 1; }
```

## Language Basics

Functions 1:

```
> f = x: x + 1
> f 2
3
> g = g': x: g' x + 1
> g f 2
4
```

Functions 2:

```
> h = { x ? 1 }: x + 1
> h
<function>
> h { }
2
> h { x = 2; }
3
```

## Derivation

A *derivation*

- can depend on any number of other derivation
- can produce one or more outputs

## Closure

A *closure*

- encapsulates all of the packages required to build or run it
- has two types, build-time closure and runtime closure

## Nix Store[1]

```
/nix/store/ffkg7rz4zxfsdix6xxmhk2v3nx76r141-nix-2.18.1
|---------|--------------------------------|---------|
 store     hash                             name
 prefix
```

- store prefix can be local or remote (binary cache)
- hash either derived from input (default) or output (CA derivation)
- `*.drv` for derivation files

[1]https://zero-to-nix.com/concepts/nix-store

## Packaging

Nix expressions ⇒ derivation(s)

- `builtins.derivation`
- `stdenv.mkDerivation` (from `nixpkgs`)
- `pkgs.buildGoApplication` (from `nixpkgs`)
- `crane.lib.x86_64-linux.buildPackage` (from `crane`)
- …

## Packaging[2]

```
{
  inputs = { ... };

  outputs = { self, nixpkgs, flake-utils }:
    flake-utils.lib.eachDefaultSystem (system:
    let
      pkgs = import nixpkgs { inherit system; };
    in
    {
      packages.default = pkgs.writeShellApplication {
        name = "cheese";
        runtimeInputs = [ pkgs.cowsay ];
        text = "cowsay cheese";
      };
    });
}
```

```
 ________
< cheese >
 --------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

[2] https://nixolo.gy/example1

## Development[3]

**Shell**:

- `nix develop`
- `direnv`

```
devShells.default = pkgs.mkShell {
  packages = with pkgs; [
    cargo
    rustc
    rustfmt
  ];
};
```

**Formatter**:

- `nix fmt`
- a single package, or ↓

```
formatter = pkgs.writeShellScriptBin "formatter" ''
  set -eoux pipefail
  shopt -s globstar
  ${pkgs.nixpkgs-fmt}/bin/nixpkgs-fmt .
  ${pkgs.rustfmt}/bin/rustfmt **/*.rs
'';
```

[3] https://nixolo.gy/example2

## Development

**Pinning**:

w/ builtin versions:

```
nix-repl> pkgs.coq_8_
pkgs.coq_8_10  pkgs.coq_8_12
pkgs.coq_8_14  pkgs.coq_8_16
pkgs.coq_8_18  pkgs.coq_8_5
pkgs.coq_8_7   pkgs.coq_8_9
...
```

w/ `nix shell`:

```
nix shell nixpkgs/<hash>#{pkg1,...}
```

or DIY!

w/ flakes:

```
inputs = {
  nixpkgsForA.url = "github:nixos/nixpkgs/<branch or hash>";
  nixpkgsForB.url = "github:nixos/nixpkgs/<branch or hash>";
  ...
};

outputs = { self, ... }: {
  ...
  pkgsA.<some pkg>;
  pkgsB.<some pkg>;
  ...
};
```

## System Configurations

**Modules**[4]**:**

```nix
{ ... }:
{
  networking.firewall.allowedTCPPorts = [ 80 443 ];
  services.caddy = {
    virtualHosts."nixolo.gy" = {
      extraConfig = "redir https://github.com/stepbrobd/nixology/tree/master{uri}";
      serverAliases = [ "*.nixolo.gy" ];
    };
  };
}
```

[4]https://mynixos.com/nixpkgs/options/services.caddy

## System Configurations[5]

```
outputs = { self, nixpkgs, ... }: {
  nixosConfigurations.example3 = nixpkgs.lib.nixosSystem {
    modules = [ ./hardware.nix ./service.nix ];
  };
};
```

**System Closure**:

```
nix build .#nixosConfigurations.example3.config.system.build.toplevel
```

**Rebuild**:

```
nixos-rebuild <switch|boot|...> --flake .#example3
```

[5] https://nixolo.gy/example3

## Resources

- https://github.com/determinatesystems/nix-installer
- https://zero-to-nix.com
- https://nixos.org/manual/nix/unstable/
- https://discourse.nixos.org
- https://mynixos.com
- REPL
- source code
  - https://github.com/features/code-search
  - https://sourcegraph.com