Nixology

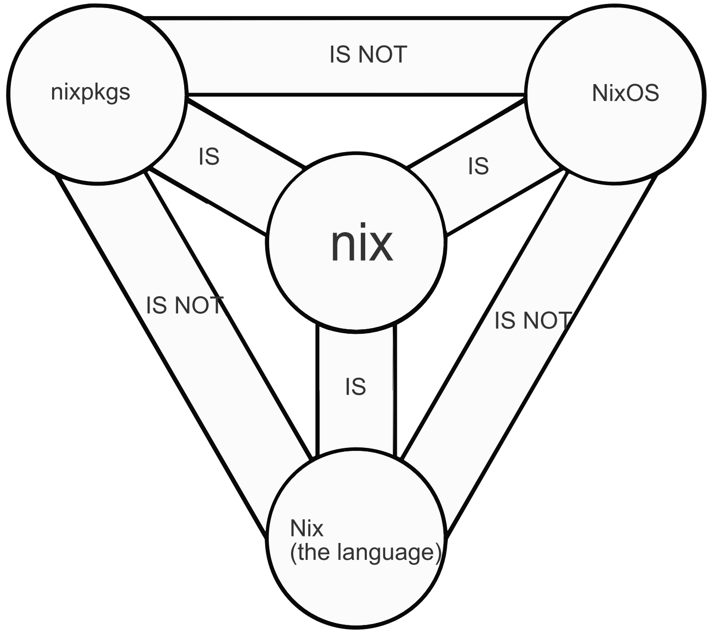
Yifei Sun

February 2, 2024

What's Nix?

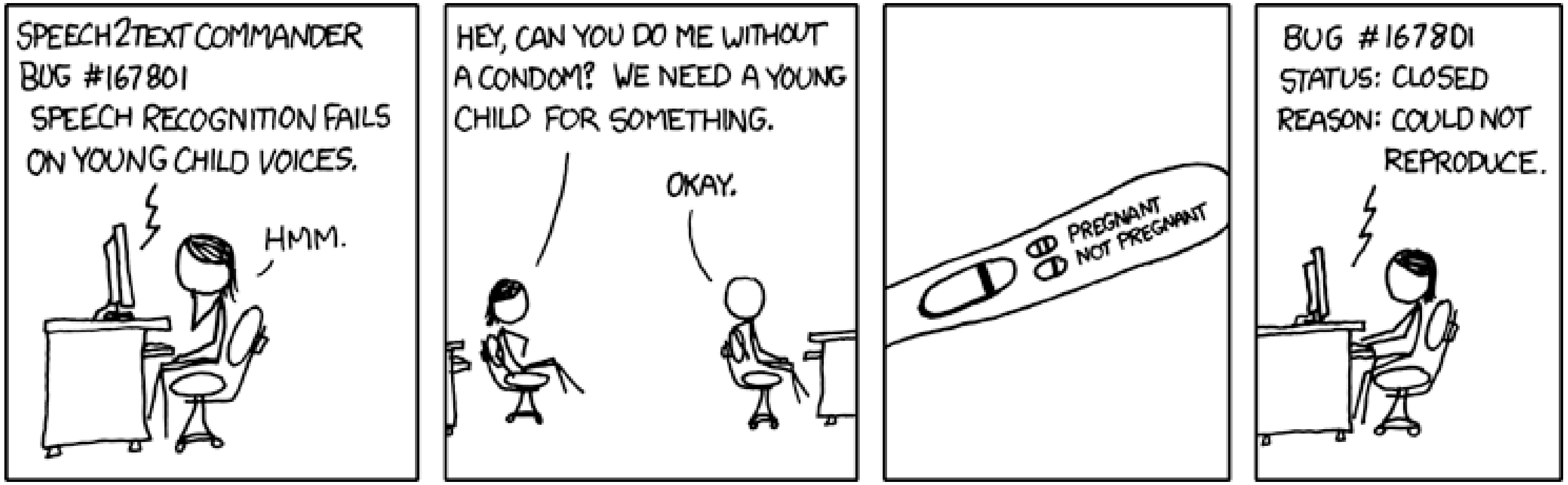
The holy trinity[1]:

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



[1]https://zackerthescar.com/nix-nix-nix

The Problem


SPEECH2TEXT COMMANDER
BUG #167801
SPEECH RECOGNITION FAILS ON YOUNG CHILD VOICES.
HMM.
HEY, CAN YOU DO ME WITHOUT A CONDOM? WE NEED A YOUNG CHILD FOR SOMETHING.
OKAY.
PREGNANT
NOT PREGNANT
BUG #167801
STATUS: CLOSED
REASON: COULD NOT REPRODUCE.
[2]

[2] https://imgs.xkcd.com/comics/cnr.png

“Reproducibility” * ** ***

```
{
  inputs = { ... };
  outputs = { self, ... } @ inputs: { ... };
}
```

*: only in **pure** mode

**: can achieve with `nix-channel`, but painful

***: the example above is with `flakes`, still experiemntal[3]

---

[3] Experiemntal™ since Nov. 2021

Channels v.s. Flakes

Channels
- required by all `nix-*`
- `nix-channel <args>`
- hard to pin

Flakes:
- have inputs and outputs, like a function
- inputs gets “locked” with `flake.lock`
- experiemntal $\neq$ unstable[4]

[4]https://determinate.systems/posts/experimental-does-not-mean-unstable

Derivations and Closures[5]

A *derivation*

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

A *closure*

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

[5]https://zero-to-nix.com/concepts/derivations

Nix Store[6]

```
/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

[6] 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[7]

```
{
  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 |
                ||     ||
```

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

Development[8]

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
'';
```

[8] 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[9]:

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

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

System Configurations[10]

```
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
```

[10] 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