Nix/NixOS/Nixpkgs

Made with Nix, ♡, and LaTeX

Ethan Carter Edwards
ethanedwards@college.harvard.edu

February 21, 2026

Who am I?

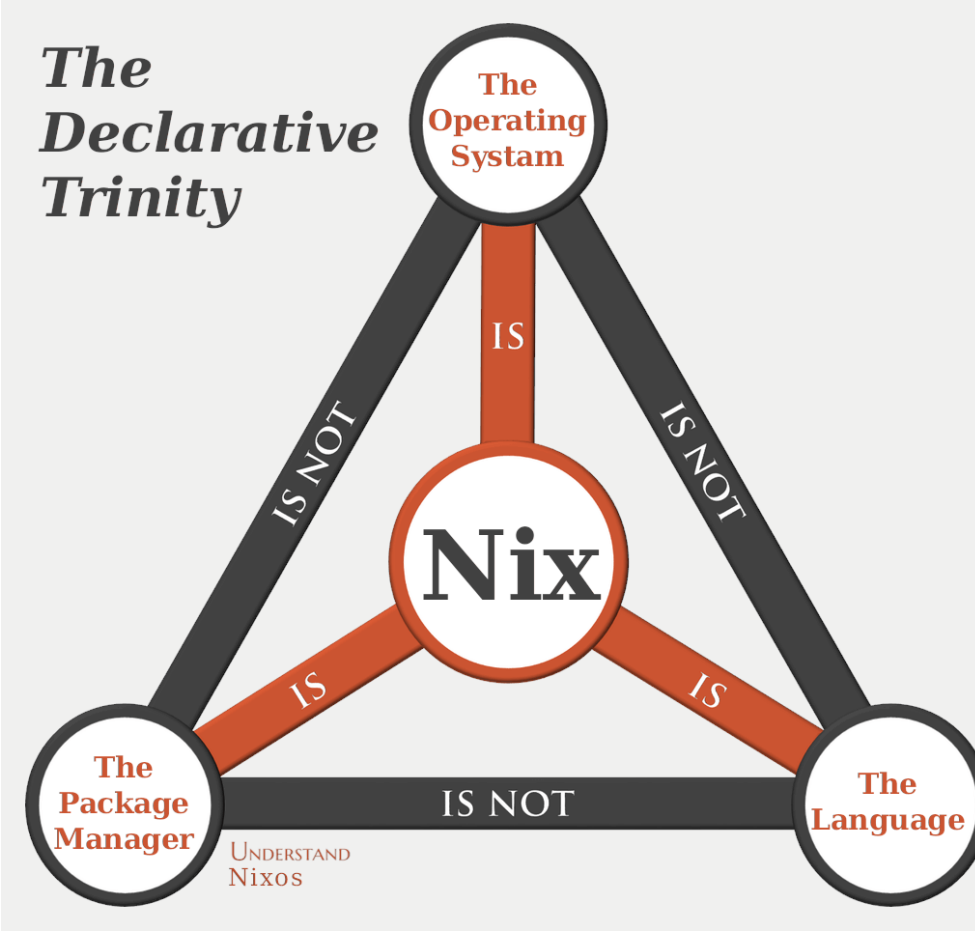
My name is Ethan Edwards. I've been heavily involved with the project for the past few years and interned with the NixOS foundation last summer. Here are some statistics outlining my contributions upstream:

1. 286 Packages maintained
2. 229 PRs merged
3. 30+ modules/integration tests written
4. 1400+ PRs reviewed
5. 216 PRs otherwise commented on
6. 407 PRs otherwise involved with
7. Maintainer for the NGI and Darwin (MacOS) teams
8. Converted countless people to Nix :)

Nix is the brainchild of Eelco Dolstra. He started the project more than 20 years ago as his PhD thesis. While he has remained heavily involved, he is no longer the technical lead.

Goals for this talk:

1. Introduce the Nix DSL
2. Introduce the Nixpkgs repository
3. Introduce the Linux Distro NixOS
4. Show my use cases/Nix in action
5. Convert you all to Nix! (world domination)



What is Nix (the DSL)?

Follow along: https://glot.io/new/nix

docker run -it ethancedwards8/nix

JSON like syntax :()

```
{
    string = "hello";
    integer = 1;
    float = 3.141;
    bool = true;
    nullval = null;
    list = [ 1 "two" false ];
    attribute-set = {
        a = { e = "hello"; };
        b = 2;
        c = 2.718;
        d = false;
    }; # comments are supported
}
```

rec keyword/let blocks

```
rec {
  one = 1;
  two = one + 1;
  three = two + 1;
}
# evals to:
{ one = 1; three = 3; two = 2; }


let
  b = a + 1;
  a = 1;
in
a + b
# evals to:
3
```

with keyword

```
let
  a = {
    x = 1;
    y = 2;
    z = 3;
  };
in
with a; [ x y z ]
# evals to
[ 1 2 3 ]
# with a; [ x y z ]
# is equivalent to
# [ a.x a.y a.z ]
```

Functions!

Nix has strong Functional Programming roots:

1. Immutable variables
2. Single argument functions (currying!)
3. Lazy evaluation
4. Anonymous, first-class functions

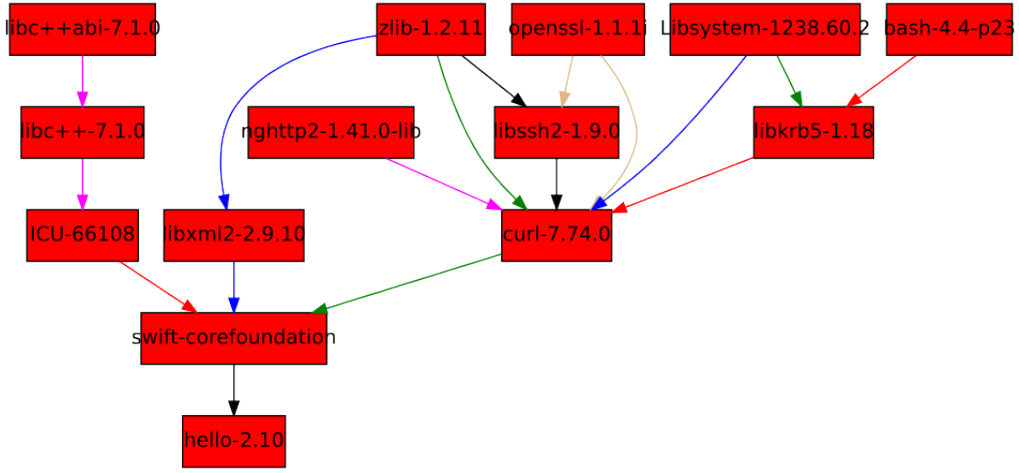
```
# single argument functions
f = x: x + 1 # f 3 == 4


# single argument functions *with a set*
let
  f = {a, b}: a + b;
in
f { a = 1; b = 2; } # == 3
```

Learn more at nix.dev

Nix the package manager

Now that you know the language, you can understand the package manager. The nix language is used to create "derivations," which are the basic building blocks of packages. A derivation essentially includes the instructions for building/compiling a program. (I'll show some code later)
Packages can depend on other packages (like libraries), creating dependency trees.



So, what's the point?

Nix provides some guarantees that are pretty attractive to users: (buzzword warning)

1. Determinism
2. Reproducibility
3. Build environments
4. Rollbacks
5. Declarative configurations!!!

Nix managing packages

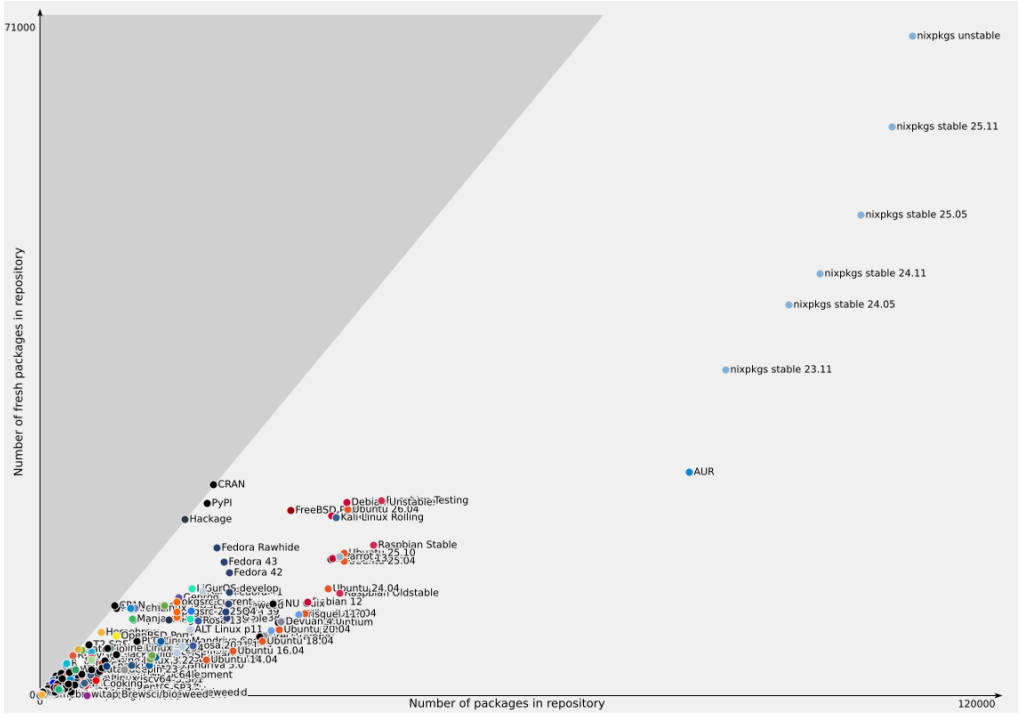
Nix can be used to imperatively (bad) install packages or to declaratively (good) install them:

```
# flakes (new, more reproducible)
$ nix profile add nixpkgs#neovim
# channel (older)
$ nix-env -iA nixpkgs.hello

# declaratively in configuration.nix
environment.systemPackages = with pkgs; [
    neovim
];
```

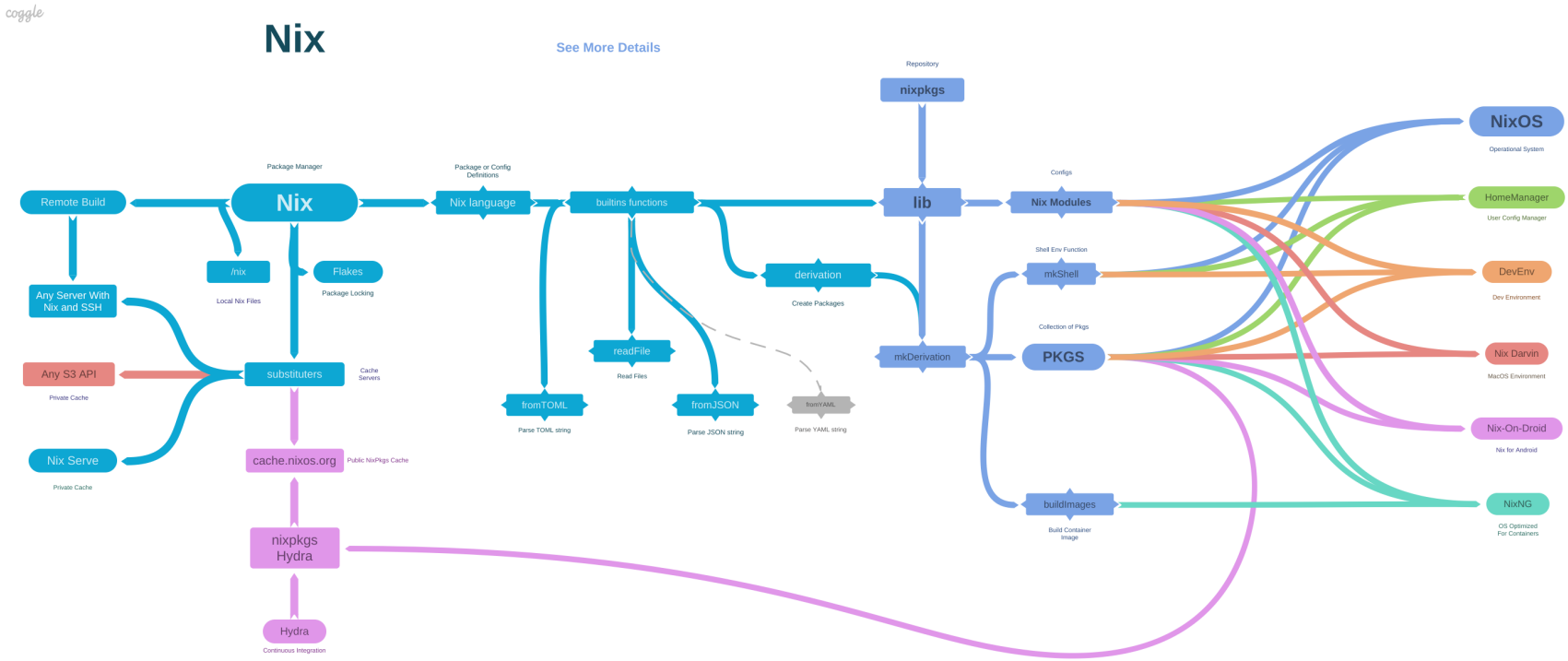
Nixpkgs

Nixpkgs is a collection of derivations that make up the Nixpkgs package set. We are the most up-to-date and largest package set in the world.



This is achieved through insane levels of automation, testing, developer tooling, and more.

Graphic

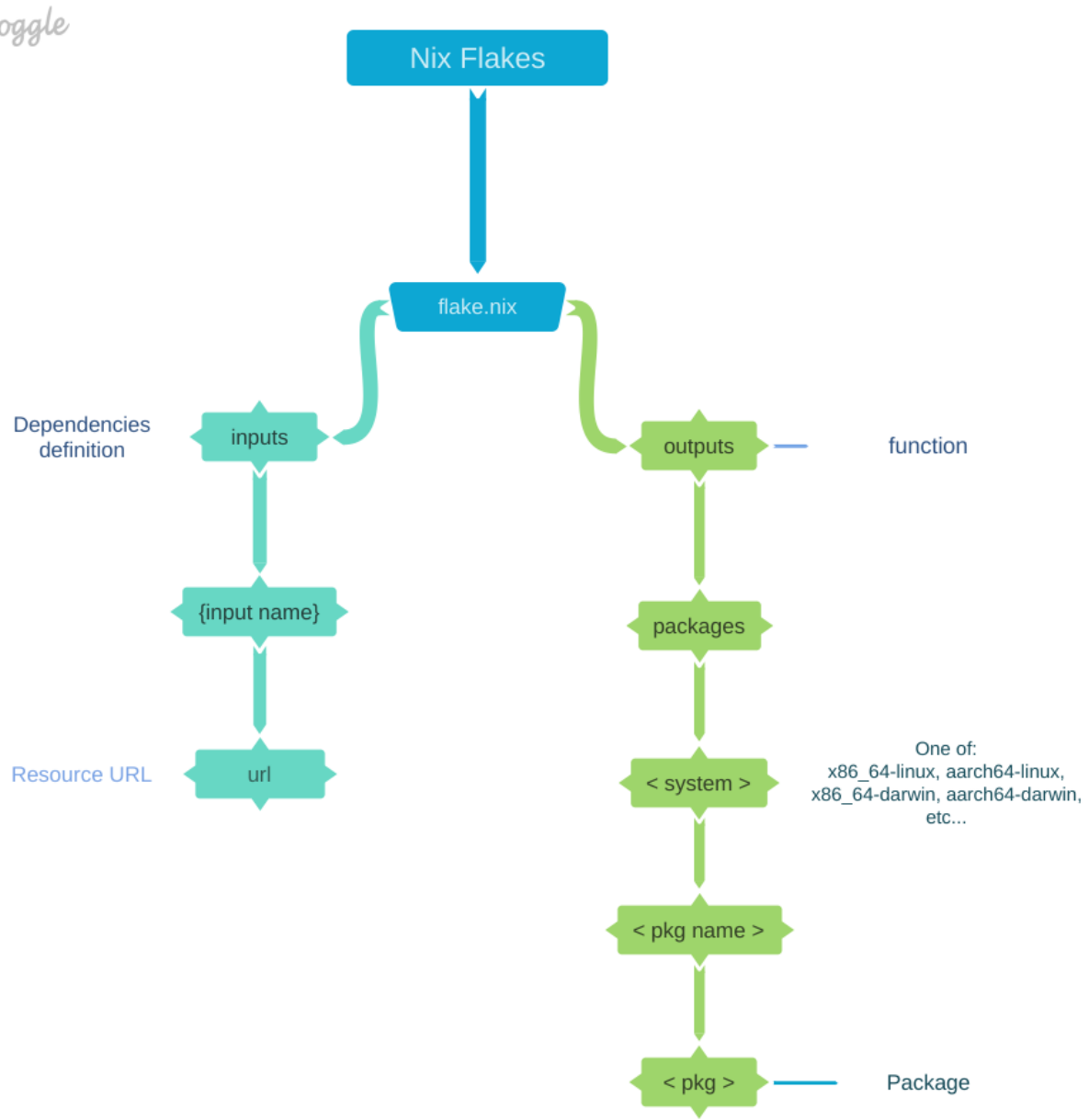

coggle
Nix
See More Details
Repository
nixpkgs
Package Manager
Package or Config Definitions
Configs
Remote Build
Nix
Nix language
builtins functions
lib
Nix Modules
NixOS
Operational System
/nix
Flakes
Local Nix Files
Package Locking
derivation
Create Packages
Shell Env Function
mkShell
HomeManager
User Config Manager
DevEnv
Dev Environment
Any Server With Nix and SSH
readFile
Read Files
mkDerivation
Collection of Pkgs
PKGS
Nix Darvin
MacOS Environment
Any S3 API
Private Cache
substituters
Cache Servers
fromTOML
Parse TOML string
fromJSON
Parse JSON string
fromYAML
Parse YAML string
Nix-On-Droid
Nix for Android
Nix Serve
Private Cache
cache.nixos.org
Public NixPkgs Cache
buildImages
Build Container Image
NixNG
OS Optimized For Containers
nixpkgs Hydra
Hydra
Continuous Integration

NixOS

NixOS is a Linux distribution built around Nix the package manager as well as Nixpkgs. It allows users to declaratively install packages like neovim, firefox, ghostty, etc. but also allows for users to install services like docker, nginx, nextcloud, and so much more. This is a bit hard to show in slides, let's do a demo!

Flakes

Flakes are the epitome of determinism!



Other builders: Docker

Nix can even be used to build Docker images:

```
pkgs.dockerTools.buildImage {
  name = "nix-hello-image";
  tag = "latest";

  contents = pkgs.hello;

  config = {
    Cmd = [ "${pkgs.hello}/bin/hello" ];
    ExposedPorts = {};
  };
}
```

Other builders: Cross Compilation

Nix can also be used to cross compile packages to other platforms/architectures.

```
nix build nixpkgs#pkgsCross.aarch64-multiplatform.hello
```