

Learn Nix the Fun Way

Why I love Nix, and you should too

Farid Zakaria · farid.m.zakaria@gmail.com

fzakaria.github.io/learn-nix-the-fun-way

github.com/fzakaria/learn-nix-the-fun-way · [02108e5](https://github.com/fzakaria/learn-nix-the-fun-way/commit/02108e5)
/nix/store/3x9v5nfy2x31cx9c5n5hnjlds9r48si7-source

Hello 🖐️

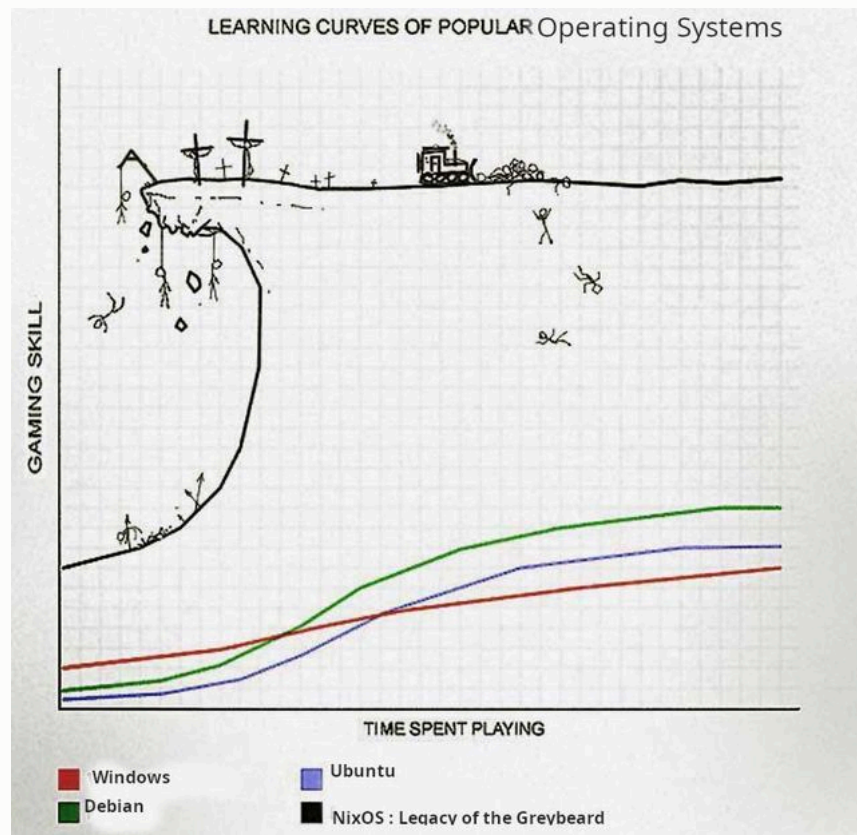
My name is Farid Zakaria, and I'm a software engineer at [Meta](#), with a PhD from UC Santa Cruz.

- Long-time Nix user (circa 2018)
- Introduced Nix at many companies, including Confluent & Google, and an active member of their internal communities
- Written a lot of Nix software and projects, such as [nixmultiverse.com](#), [omnibin.dev](#), [trynix.dev](#) and [omniflake.com](#), to name a few
- Daily drive NixOS on my wonderful [frame.work](#) laptop

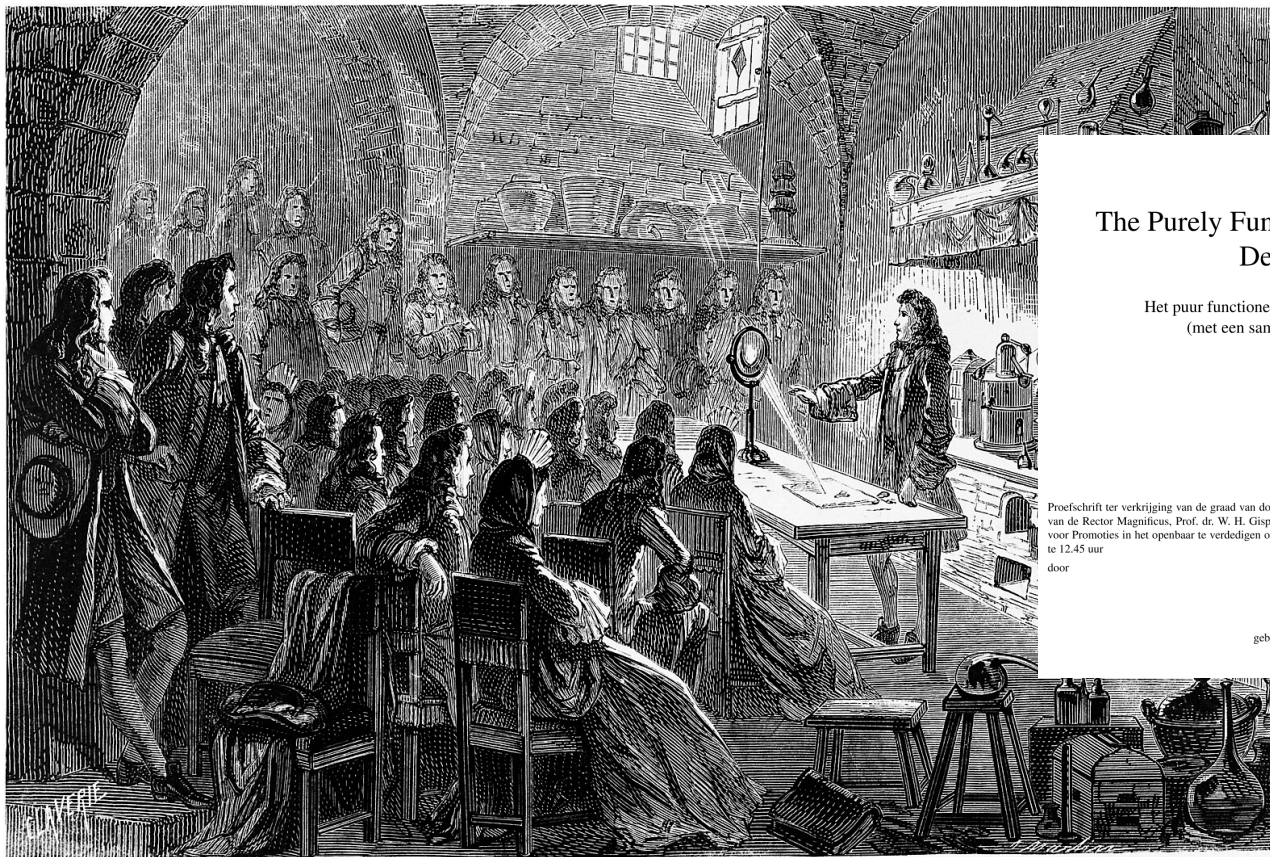


farid.m.zakaria@gmail.com
linkedin.com/in/fmzakari
x.com/fmzakari

Disclaimer



Why I love Nix, and you should too! 💖



The Purely Functional Software Deployment Model

Het puur functionele softwaredeploymentmodel
(met een samenvatting in het Nederlands)

Proefschrift ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de Rector Magnificus, Prof. dr. W. H. Gispen, ingevolge het besluit van het College voor Promoties in het openbaar te verdedigen op woensdag 18 januari 2006 des middags te 12.45 uur door

Eelco Dolstra

geboren op 18 augustus 1978, te Wageningen



Let's do it different this time.

Let's learn Nix the fun way.

What is my IP?

```
#!/usr/bin/env bash  
> curl -s http://httpbin.org/get | \  
    jq --raw-output .origin
```

73.231.52.39

The same script, in Nix

```
{
  system ? builtins.currentSystem,
  pkgs ?
    let
      # NixOS 24.11
      rev = "5ef6c425980847c78a80d759abc476e941a9bf42";
      url = "https://github.com/NixOS/nixpkgs/archive/${rev}.tar.gz";
    in
      import (fetchTarball url) {inherit system;},
}:
derivation {
  name = "what-is-my-ip";
  builder = "/bin/sh";
  args = [
    "-c"
    ''
      ${pkgs.coreutils}/bin/mkdir -p $out/bin
      echo '#!/bin/sh' > $out/bin/what-is-my-ip
      echo '${pkgs.curl}/bin/curl -s http://httpbin.org/get | \
        ${pkgs.jq}/bin/jq --raw-output .origin' \
        >> $out/bin/what-is-my-ip
      ${pkgs.coreutils}/bin/chmod +x $out/bin/what-is-my-ip
    ''
  ];
  system = builtins.currentSystem;
  outputs = ["out"];
}
```

The same script, in idiomatic Nix

```
{
  system ? builtins.currentSystem,
  pkgs ?
    let
      rev = "5ef6c425980847c78a80d759abc476e941a9bf42";
      url = "https://github.com/NixOS/nixpkgs/archive/${rev}.tar.gz";
    in
      import (fetchTarball url) {inherit system;},
}:
pkgs.writeShellScriptBin "what-is-my-ip" ''
  ${pkgs.curl}/bin/curl -s http://httpbin.org/get | \
  ${pkgs.jq}/bin/jq --raw-output .origin
''
```

Let's build it

```
> nix build -f what-is-my-ip.nix --print-out-paths
```

```
/nix/store/y2g0ijqqiyi9vxr9xgmvvgbllxqflqzav-what-is-my-ip
```

```
> /nix/store/y2g0ijqqiyi9vxr9xgmvvgbllxqflqzav-what-is-my-ip/bin/what-is-my-ip
```

```
73.231.52.39
```

What did we build?

```
> cat /nix/store/y2g0ijqqiyi9vxr9xgmvvgblxqflqzav-what-is-my-ip/bin/what-is-my-ip
```

```
#!/nix/store/gwgqdl0242ymlikq9s9s62gkp5cvyal3-bash-5.2p37/bin/bash  
/nix/store/7p8814g62bybwa8blmaqr9piv3vjwan5-curl-8.11.1-bin/bin/curl -s http://httpbin.org/get | \  
/nix/store/jhw7aklgrc23125ysglmc72y8zq0lndw-jq-1.7.1-bin/bin/jq --raw-output .origin
```

What is `/nix/store/y2g0ijqqiyi9vxr9xgmvvgblxqflqzav-what-is-my-ip`?

Why do we care about `y2g0ijqqiyi9vxr9xgmvvgblxqflqzav`?

```
> nix derivation show /nix/store/y2g0ijqqiyi9vxr9xgmvvgblxqflqzav-what-is-my-ip
```

Hashed into y2g0ijqqiyi9vvr9xgmvvgblxqflqzav

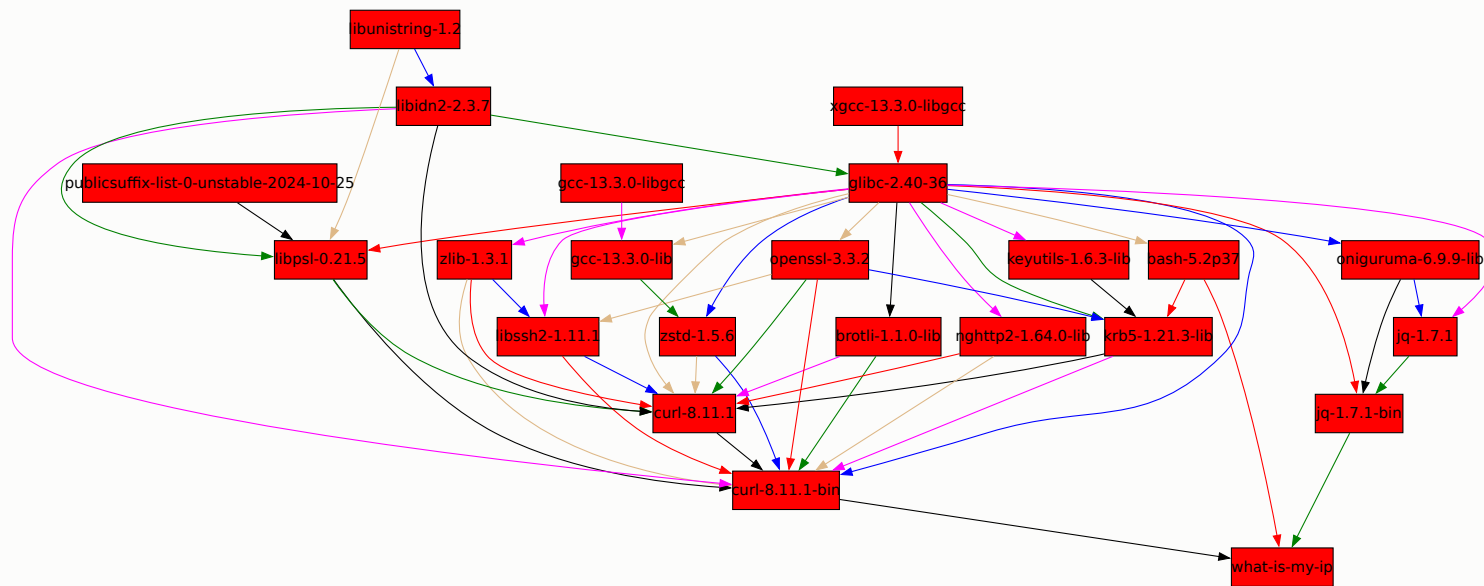
```
{
  "/nix/store/rw89fpljini3yx09v83ds4hyl4945lmk-what-is-my-ip.drv": {
    "args": [
      "-e",
      "/nix/store/v6x3cs394jgqfbi0a42pam708flxaphh-default-builder.sh"
    ],
    "builder": "/nix/store/gwgqdl0242ymlikq9s9s62gkp5cvyal3-bash-5.2p37/bin/bash",
    "env": {
      "buildCommand": "<build command>",
      "builder": "/nix/store/gwgqdl0242ymlikq9s9s62gkp5cvyal3-bash-5.2p37/bin/bash",
      "name": "what-is-my-ip",
      "out": "/nix/store/y2g0ijqqiyi9vvr9xgmvvgblxqflqzav-what-is-my-ip",
      "stdenv": "/nix/store/cf464y4lp2x3lhlqvbg6678lc3f8zbd6-stdenv-linux",
      "system": "x86_64-linux",
      "text": "<contents of script removed for brevity>"
    },
    "inputDrvs": {
      "/nix/store/7k0msqyp2dm021sdj0qjgpkzff8xhqzr-bash-5.2p37.drv": {},
      "/nix/store/a9dyisrdm25k6cprjvypprxs3n8b283-jq-1.7.1.drv": {},
      "/nix/store/crgvallixc0kpg62whvkiv6x38p15nqr-curl-8.11.1.drv": {},
      "/nix/store/ycj0m56p8b0rv9v78mggfa6xhm3lrw3-stdenv-linux.drv": {}
    },
    "inputSrcs": [
      "/nix/store/v6x3cs394jgqfbi0a42pam708flxaphh-default-builder.sh"
    ],
    "name": "what-is-my-ip",
    "outputs": {
      "out": {
        "path": "/nix/store/y2g0ijqqiyi9vvr9xgmvvgblxqflqzav-what-is-my-ip"
      }
    },
    "system": "x86_64-linux"
  }
}
```

Extremely strong guarantees

Seeing `y2g0ijqqiyi9vxr9xgmvvgbllxqflqzav` means we have *extremely* strong guarantees of the software graph down to the commit and build instructions.



Once you know your full software graph, everything is a permutation on that



```
> nix-store --query --graph /nix/store/y2q0ijqqiyi9vxr9xgmvvqblxqflqzav-what-is-my-ip
```

Permutation #1: a developer shell

```
let
  rev = "5ef6c425980847c78a80d759abc476e941a9bf42";
  url = "https://github.com/NixOS/nixpkgs/archive/${rev}.tar.gz";
  pkgs = import (fetchTarball url) {};
  what-is-my-ip = import ./what-is-my-ip.nix {inherit pkgs;};
in
  pkgs.mkShell {
    packages = [what-is-my-ip];

    shellHook = ''
      echo "Hello, Nix!"
    '';
  }
```

Permutation #1: a developer shell

```
> nix-shell what-is-my-ip-shell.nix
```

```
Hello, Nix!
```

```
[nix-shell]
```

```
> which what-is-my-ip
```

```
/nix/store/y2g0ijqqiyi9vvr9xgmvvgblxqflqzav-what-is-my-ip/bin/what-is-my-ip
```

```
[nix-shell]
```

```
> /nix/store/y2g0ijqqiyi9vvr9xgmvvgblxqflqzav-what-is-my-ip/bin/what-is-my-ip
```

```
73.231.52.39
```

Remember y2g0ijqqiyi9vvr9xgmvvgblxqflqzav

Reproducible developer environments ✓



We actually know everything about our code now.

We can now do binary or source deployments 🚀🔧📦
since we know the full dependency closure of our tool(s).

We simply copy the necessary `/nix/store` paths to another machine ~~with Nix installed.~~

Profit 😊💰

Permutation #2: a deployment

```
> nix copy --to ssh://dennard.soe.ucsc.edu \  
    $(nix build -f what-is-my-ip.nix --print-out-paths)  
  
> ssh dennard.soe.ucsc.edu  
Last login: Mon Mar  3 05:02:57 2025 from 73.231.52.39  
  
[dennard]  
> /nix/store/y2g0ijqqiyi9vxr9xgmvvgbllxqflqzav-what-is-my-ip/bin/what-is-my-ip  
128.114.53.24
```

Remember y2g0ijqqiyi9vxr9xgmvvgbllxqflqzav

Binary (or source) deployments ✓



That's awesome! Unfortunately, our company uses Docker / Kubernetes 🙄

No problem! Again, once we know the graph of our software everything is just a permutation.

```
let
  rev = "5ef6c425980847c78a80d759abc476e941a9bf42";
  url = "https://github.com/NixOS/nixpkgs/archive/${rev}.tar.gz";
  pkgs = import (fetchTarball url) {};
  what-is-my-ip = import ./what-is-my-ip.nix {inherit pkgs;};
in
  pkgs.dockerTools.buildImage {
    name = "what-is-my-ip-docker";
    config = {
      Cmd = ["${what-is-my-ip}/bin/what-is-my-ip"];
    };
  }
```

Permutation #3: a container

```
> nix build -f what-is-my-ip-docker.nix --print-out-paths

/nix/store/vwlbi7m9l23cm6kjylskh4fskh02c5zc-docker-image-what-is-my-ip-docker.tar.gz

> docker load /nix/store/vwlbi7m9l23cm6kjylskh4fskh02c5zc-docker-image-what-is-my-ip-docker.tar.gz
Loaded image: what-is-my-ip-docker:vwlbi7m9l23cm6kjylskh4fskh02c5zc

> docker run -it what-is-my-ip-docker:vwlbi7m9l23cm6kjylskh4fskh02c5zc
73.231.52.39

> docker inspect what-is-my-ip-docker:vwlbi7m9l23cm6kjylskh4fskh02c5zc --format "{{.Config.Cmd}}"
[/nix/store/y2g0ijqqiyi9vvr9xgmvvgbllxqflqzav-what-is-my-ip/bin/what-is-my-ip]
```

Remember y2g0ijqqiyi9vvr9xgmvvgbllxqflqzav

Cool! 🧐

Nix + Docker integrate perfectly. The image produced has only the files exactly necessary to run the tool provided, effectively distroless.

You can do the same thing for any deployment format!

- OCI images
- tarballs
- self-extracting ZIP
- rpm
- deb
- Flatpak / AppImage

A Linux distribution is simply a collection of packages.

Permutation #4: an operating system

```
let
  rev = "5ef6c425980847c78a80d759abc476e941a9bf42";
  url = "https://github.com/NixOS/nixpkgs/archive/${rev}.tar.gz";
  nixpkgs = fetchTarball url;
  pkgs = import nixpkgs {};
  what-is-my-ip = import ./what-is-my-ip.nix {inherit pkgs;};
  nixos = import "${nixpkgs}/nixos" {
    configuration = {
      users.users.alice = {
        isNormalUser = true;
        # enable sudo
        extraGroups = ["wheel"];
        packages = [
          what-is-my-ip
        ];
        initialPassword = "swordfish";
      };

      system.stateVersion = "24.05";
    };
  };
in
  nixos.vm
```

Permutation #4: an operating system

```
> nix build -f what-is-my-ip-vm.nix --print-out-paths

/nix/store/k6vqp2hn0gh7r5dmc61f5knqgnqn53wz-nixos-vm

> QEMU_KERNEL_PARAMS=console=ttyS0 \
  /nix/store/k6vqp2hn0gh7r5dmc61f5knqgnqn53wz-nixos-vm/bin/run-nixos-vm -nographic; reset

... long boot sequence ...

<<< Welcome to NixOS 24.11pre-git (x86_64) - ttyS0 >>>

nixos login: alice
Password:

[alice@nixos:~]$ which what-is-my-ip
/etc/profiles/per-user/alice/bin/what-is-my-ip

[alice@nixos:~]$ realpath $(which what-is-my-ip)
/nix/store/y2g0ijqqiyi9vxr9xgmvvqblxqflqzav-what-is-my-ip/bin/what-is-my-ip

[alice@nixos:~]$ what-is-my-ip
73.231.52.39
```

Remember y2g0ijqqiyi9vxr9xgmvvqblxqflqzav

So... what is Nix?

A language, and a command-line tool

```
# hello-world.nix
derivation {
  name = "hello-world";
  # POSIX requires this file to be present
  # so we can depend on it but not on /bin/bash
  builder = "/bin/sh";
  args = [ "-c" "echo 'hello world' > $out" ];
  system = builtins.currentSystem;
}
```

```
# hello-world-writeText.nix
let pkgs = import <nixpkgs> {};
in
pkgs.writeText "hello-world" "hello world"
```

```
> nix build --file hello-world.nix --json | jq -r '.[].outputs.out'
/nix/store/dg2rpsw16yfiw3h2ma14j5qh7a6m08kk-hello-world

> cat /nix/store/dg2rpsw16yfiw3h2ma14j5qh7a6m08kk-hello-world
hello world
```

Repository size/freshness map



```
fzakaria/learn-nix-the-fun-way@02108e5 · /nix/store/3x9v5nfy2x31cx9c5n5hnjlds9r48si7-source
```

One recipe, many permutations

```
> nix run nixpkgs#hello
```

```
Hello, world!
```

```
> nix build nixpkgs#hello --no-link --print-out-paths
```

```
/nix/store/mlr53pnm6hnjwyjmxska24y8amvlpjp-hello-2.12.1
```

```
> nix build nixpkgs#pkgsStatic.hello --no-link --print-out-paths
```

```
/nix/store/d3scdlvhyflvjgjjwgp2ydbhdlss7zpcr-hello-static-x86_64-unknown-linux-musl-2.12.1
```

```
> nix build nixpkgs#pkgsCross.riscv64.hello --no-link --print-out-paths
```

```
/nix/store/p2fhzrphxmd8if2mf7b28biawj7walf1-hello-riscv64-unknown-linux-gnu-2.12.1
```

```
> /nix/store/p2fhzrphxmd8if2mf7b28biawj7walf1-hello-riscv64-unknown-linux-gnu-2.12.1/bin/hello
```

```
bash: ../bin/hello: cannot execute binary file
```

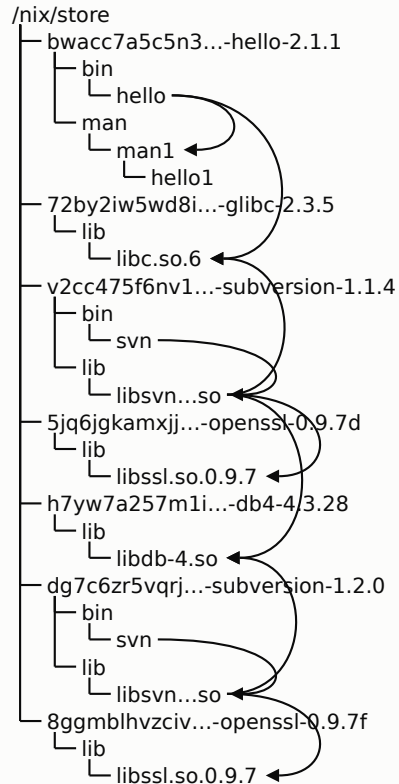
```
> nix build nixpkgs#pkgsCross.mingw64.hello --no-link --print-out-paths
```

```
/nix/store/b78fyx3z62pdww8a3nk8vfa3cb2api6h-hello-x86_64-w64-mingw32-2.12.1
```

```
> wine64 /nix/store/b78fyx3z62pdww8a3nk8vfa3cb2api6h-hello-x86_64-w64-mingw32-2.12.1/bin/hello.exe
```

```
Hello, world!
```

A store



Source: [Eelco Dolstra, The Purely Functional Software Deployment Model \(2006\)](#), redrawn

An operating system

```
eudoxia@sextant:~$ neofetch
```



```
eudoxia@sextant
```

```
OS: NixOS 22.11.3681.2b1bba76a13 (Raccoon) x86_64
Host: LENOVO 20Y5002MAU
Kernel: 5.15.106
Uptime: 2 hours, 7 mins
Packages: 934 (nix-system), 1431 (nix-user)
Shell: bash 5.1.16
Resolution: 2560x1600
DE: none+spectrum
WM: spectrum
Theme: Adwaita [GTK3]
Icons: Adwaita [GTK3]
Terminal: urxvt
Terminal Font: terminus
CPU: 11th Gen Intel i7-11800H (16) @ 4.600GHz
GPU: NVIDIA GeForce RTX 3050 Ti Mobile
GPU: Intel TigerLake-H GT1 [UHD Graphics]
Memory: 3218MiB / 15727MiB
```



Nix Language

Nix Operating System

Nix Package Manager

Modeling software as a directed acyclic graph.

Nix Ecosystem

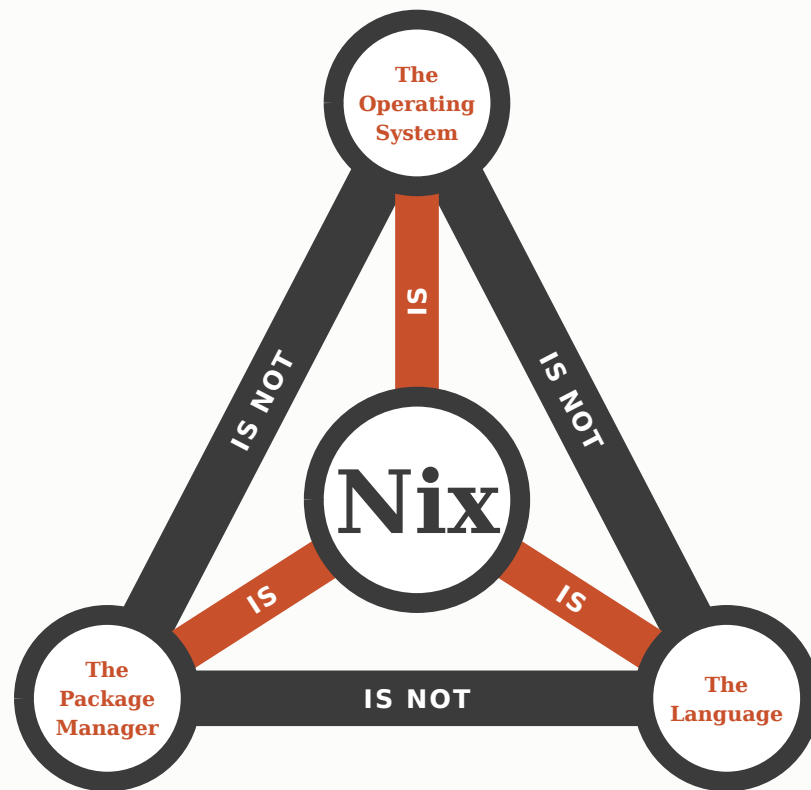
Nix Build System

Nix Command Line Tools

Nix Stores

Developer Environments

The declarative trinity



Source: Understand NixOS, redrawn

Under the hood

How does any of this actually work?

A story about Ruby

```
> ls /usr/bin/ruby*  
/usr/bin/ruby  /usr/bin/ruby3.0  /usr/bin/ruby3.1  
  
> /usr/bin/ruby3.1 --version  
ruby 3.1.2p20 (2022-04-12 revision 4491bb740a) [x86_64-linux-gnu]
```

Building Ruby

1. Install the prerequisite dependencies for building the CRuby interpreter:

- C compiler

For RubyGems, you will also need:

- OpenSSL 1.1.x or 3.0.x / LibreSSL
- libyaml 0.1.7 or later
- zlib

If you want to build from the git repository, you will also need:

- autoconf - 2.67 or later
- gperf - 3.1 or later
 - Usually unneeded; only if you edit some source files using gperf
- ruby - 2.5 or later
 - We can upgrade this version to system ruby version of the latest Ubuntu LTS.

2. Install optional, recommended dependencies:

- libffi (to build fiddle)
- gmp (if you wish to accelerate Bignum operations)
- libexecinfo (FreeBSD)
- rustc - 1.58.0 or later, if you wish to build YJIT.

From Ruby's [building_ruby.md](#)

What does it actually load?

```
> libtree /usr/bin/ruby3.1 -p
/usr/bin/ruby3.1
├─ /lib/x86_64-linux-gnu/libruby-3.1.so.3.1
│   ├─ /lib/x86_64-linux-gnu/libz.so.1
│   ├─ /lib/x86_64-linux-gnu/libcrypt.so.1
│   └─ /lib/x86_64-linux-gnu/libgmp.so.10
├─ /lib/x86_64-linux-gnu/libcrypt.so.1
├─ /lib/x86_64-linux-gnu/libgmp.so.10
└─ /lib/x86_64-linux-gnu/libz.so.1
```

...and everything it loads, loads

```
> libtree /usr/bin/ruby3.1 -vvv
```

```
/usr/bin/ruby3.1
```

```
├─ libruby-3.1.so.3.1 [ld.so.conf]
│   └─ libz.so.1 [ld.so.conf]
│       └─ libc.so.6 [ld.so.conf]
│           └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libc.so.6 [ld.so.conf]
│   └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libm.so.6 [ld.so.conf]
│   └─ libc.so.6 [ld.so.conf]
│       └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libcrypt.so.1 [ld.so.conf]
│   └─ libc.so.6 [ld.so.conf]
│       └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libgmp.so.10 [ld.so.conf]
│   └─ libc.so.6 [ld.so.conf]
│       └─ ld-linux-x86-64.so.2 [ld.so.conf]
```

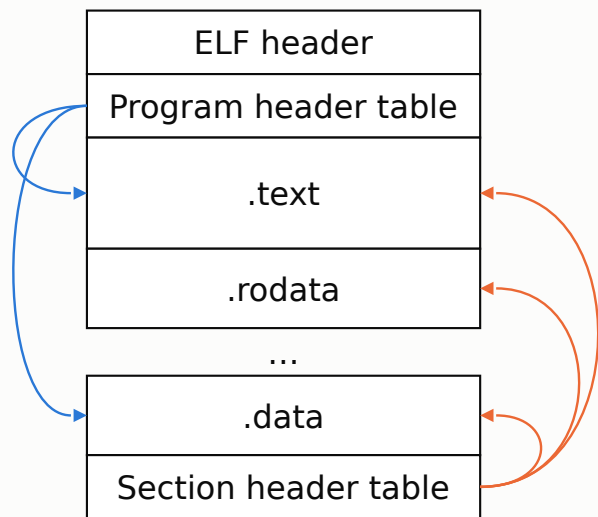
```
├─ libc.so.6 [ld.so.conf]
│   └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libm.so.6 [ld.so.conf]
│   └─ libc.so.6 [ld.so.conf]
│       └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libcrypt.so.1 [ld.so.conf]
│   └─ libc.so.6 [ld.so.conf]
│       └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libgmp.so.10 [ld.so.conf]
│   └─ libc.so.6 [ld.so.conf]
│       └─ ld-linux-x86-64.so.2 [ld.so.conf]
├─ libz.so.1 [ld.so.conf]
│   └─ libc.so.6 [ld.so.conf]
│       └─ ld-linux-x86-64.so.2 [ld.so.conf]
```

Ruby, as Nix sees it



```
> nix-store --query --include-outputs --graph \  
$(nix-store --query --deriver $(nix eval nixpkgs#ruby.outPath --raw))
```

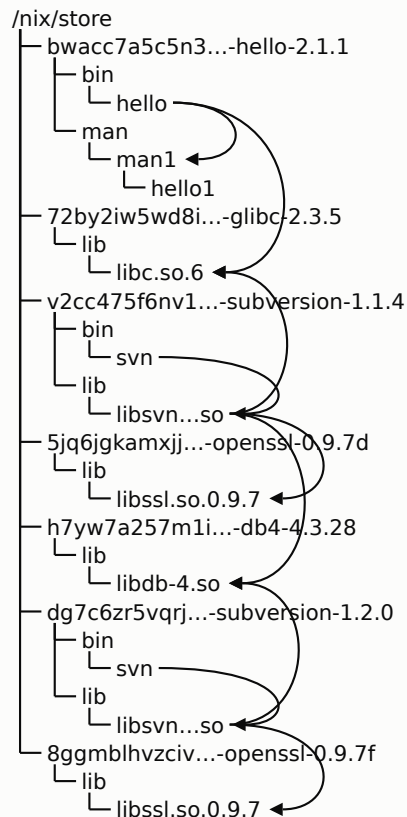
Quick primer: what's a binary?



```
$ readelf -a $(which ruby) | grep NEEDED
0x0000000000000001 (NEEDED) Shared library: [libruby-2.7.so.2.7]
0x0000000000000001 (NEEDED) Shared library: [libc.so.6]
$ readelf -a libruby-2.7.so.2.7 | grep NEEDED
0x0000000000000001 (NEEDED) Shared library: [libpthread.so.0]
0x0000000000000001 (NEEDED) Shared library: [librt.so.1]
0x0000000000000001 (NEEDED) Shared library: [libgmp.so.10]
0x0000000000000001 (NEEDED) Shared library: [libdl.so.2]
0x0000000000000001 (NEEDED) Shared library: [libcrypt.so.1]
0x0000000000000001 (NEEDED) Shared library: [libm.so.6]
0x0000000000000001 (NEEDED) Shared library: [libc.so.6]
0x0000000000000001 (NEEDED) Shared library: [ld-linux-x86-64.so.2]
```

Source: [Surueña, CC BY-SA 3.0](#), redrawn

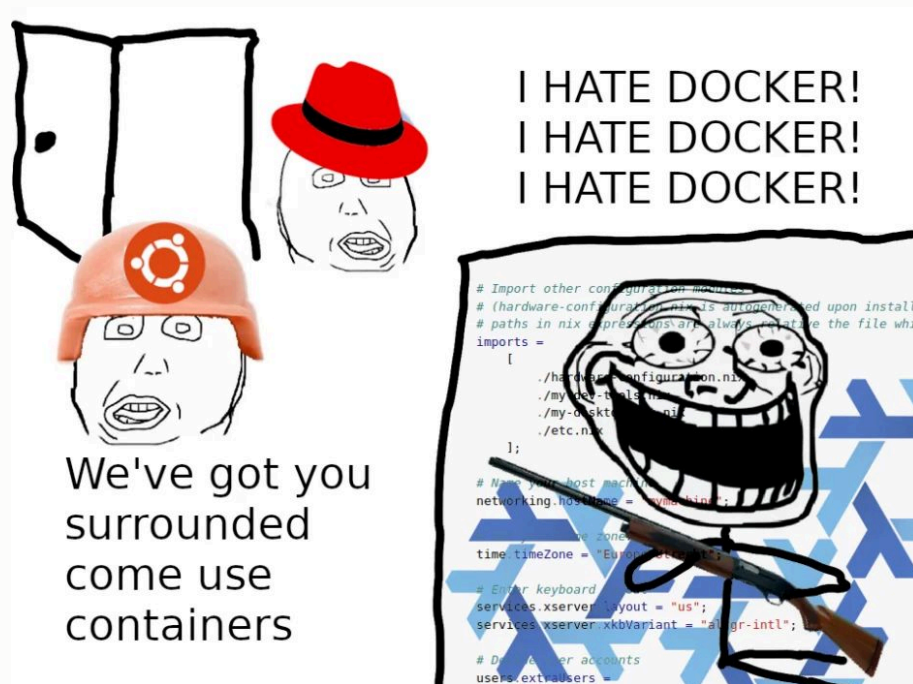
An alternative to the FHS model



```
$ patchelf --print-rpath ruby | tr ':' '\n'
/nix/store/yyhykbfxj6jfc62x4f9a2awfbch605fd-ruby-2.6.6/lib
/nix/store/6hhqb94ilrdmh0cx5bhdqb92m6bqkvdj-zlib-1.2.11/lib
/nix/store/90illc73xfs933d06daq6d41njs8yh66-glibc-2.32-37/
```

Source: [Eelco Dolstra, The Purely Functional Software Deployment Model \(2006\)](#), redrawn

When you discover NixOS



https://www.reddit.com/r/NixOS/comments/p132mz/when_you_discover_nixos/

Fin.

Thank you!

fzakaria.github.io/learn-nix-the-fun-way

fzakaria.com/2024/07/05/learn-nix-the-fun-way.html