Welcome to the Nixos introduction

NLLGG/Linux Nijmegen beginners introduction

Dude
NLLGG

14/04/2026


NixOS

Introduction to Nixos

Welcome!

Who is Hans

My name is Hans, previously working in the IT Now in Quality Assurance for the medical device sector

I have 25 years of experience in Debian, Ubuntu, FreeBSD, Arch and Nixos

Member of NLLGG, and yes I have a T-Shirt 👕

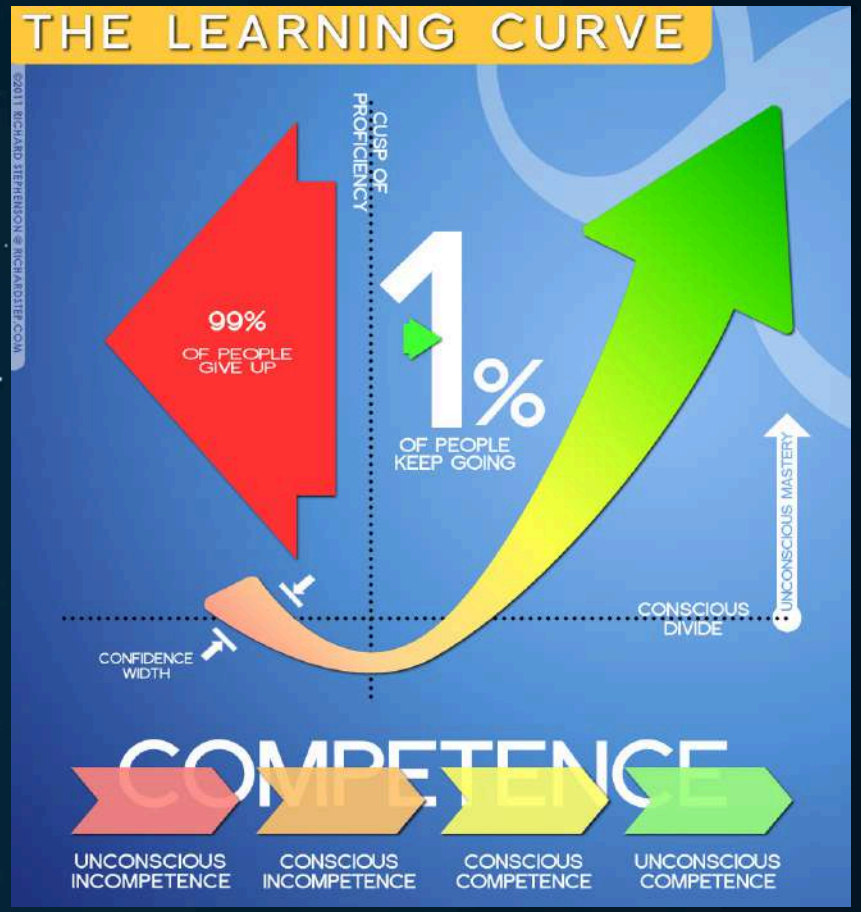
I work 🥊, have 3 kids 🚹 and a wife 🚺 , Brew beer 🍺, code 📄 and solder hardware 🔲.

I am proud that you give me the chance to tell about by experiences with tools and linux/BSD.

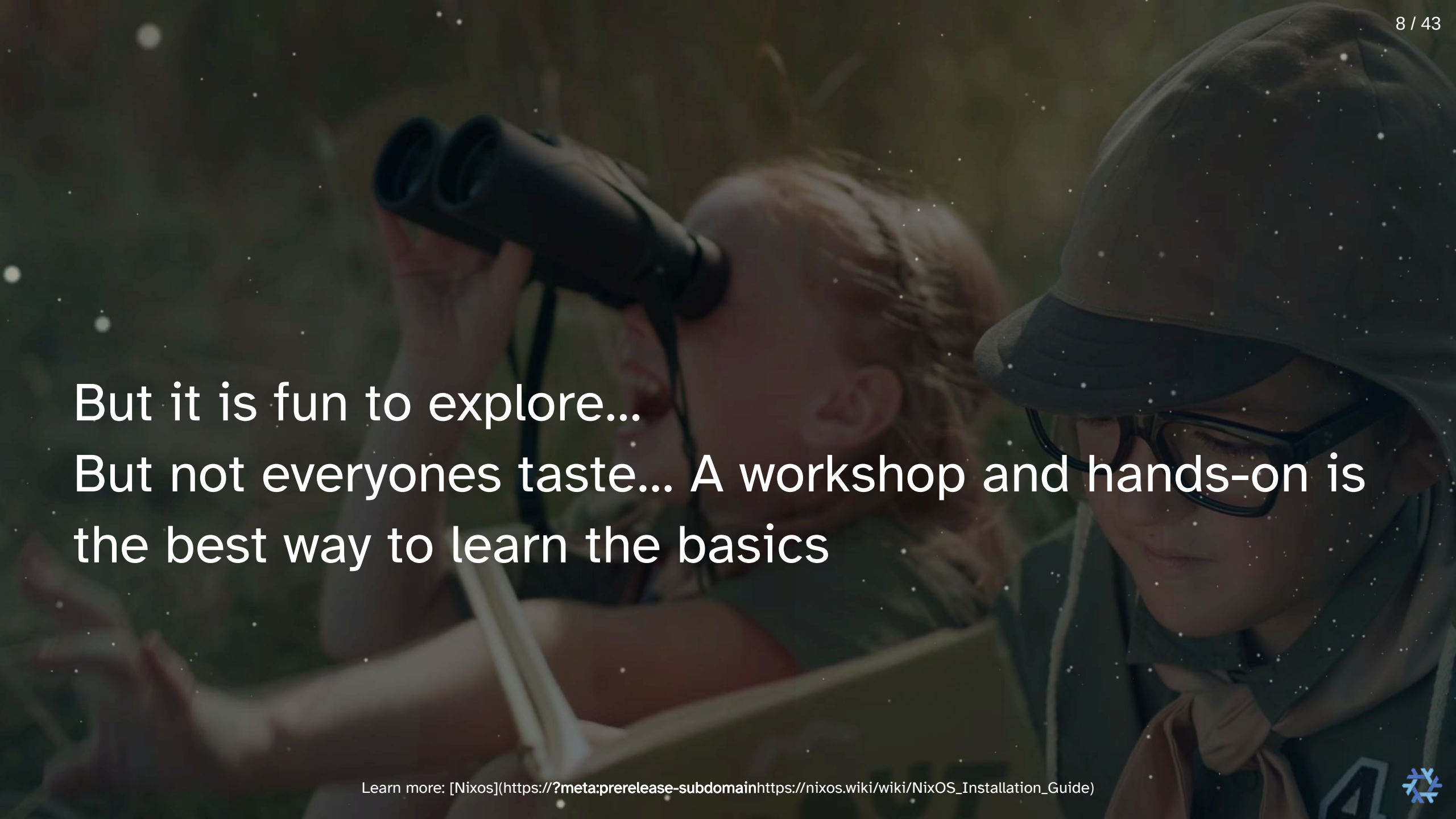
Nixos is a time rabbit hole

Nixos is a time rabbit hole

But after a steep learning curve
it stabilizes and you will find



the zen mode



But it is fun to explore...
But not everyones taste... A workshop and hands-on is the best way to learn the basics

Learn more: [Nixos](https://?meta:prerelease-subdomainhttps://nixos.wiki/wiki/NixOS_Installation_Guide)

1: Introduction to Nixos

Birth of Nixos

> “A Purely Functional NixOS: This paper has implemented the NixOS system configuration model, which means that all static parts of a system are built by pure functions and are immutable, stored in a way analogously to a heap in a purely function language.”
>
> Thesis from Eelco Dijkstra (2006) [1]

1. https://edolstra.github.io/pubs/phd-thesis.pdf

Nixos Architecture and OS variations

- System architectures
- NixOS provides out of the box support for most `x86_64` devices, and `generic ARM64` devices. `32-bit x86 64-bit x86 32-bit ARM 64-bit ARM MIPS RISC-V`
- Filesystems out of the box `Ext4` / `btfrs` and with some tweaks `ZFS`
- There are different flavors of Nix and NixOS Nix can be used as packaging system on Linux Home manager as additional user space config environment
- We will focus on NixOS as OS, not covering in detail Home Manager / Flakes

See https://wiki.nixos.org/wiki/Comparison_of_NixOS_setups

Why Nixos?

Imagine You start Python and you get this :

```
import numpy
```

```
>>> import numpy
Segmentation fault core dumped
```


It works on my machine?

Declarative, reproducible development environments. No more “works on my machine.” Create environments that work seamlessly and are easily sharable across platforms.

Why not Nixos?

- Small changes? Rebuild. So in the beginning you do this a lot.
- Steep Learning Curve. Not for the faint hearted
- Pre-compiled binaries for other platforms do not work (although AppImage and FlatPak work fine!)
- You must like coding and git repositories

2: Nixos Basics

Key features

REPRODUCABLE ✈️

DECLARATIVE 🤝

IMMUTABLE 🪂

REPRODUCABLE ✈️

- Rolling release, update upon request
- “Never Crashes”, when it does: recovery is easy
- Does the new config fail? Revert to the previous one
- One config file to maintain for all your systems, from `intel` / `AMD` / `Darwin` to `Raspberry`
- `Git` as repository, backups are easy. New system? Copy/Adjust/Deploy

Key features

REPRODUCABLE ✈️

DECLARATIVE 🤝

IMMUTABLE 🪂

DECLARATIVE 🤝

- 2 files for the (initial) config:
  `configuration.nix` and **`hardware-configuration.nix`**
- Configuration is defined through clear, descriptive specifications rather than step-by-step scripts.
- `Modular` break down of you own machine build, one can `read and comment`
- Test a package? `nix-shell -p vim`
- Temporary shells / upon request: `nix develop` / `nix-shell` using `.nix` files
- Nix `Flakes` add another dimension: pick you pieces of add-ons and extend the system

default hardware-configuration.nix and configuration.nix

Fragments of code....

```nix
# Do not modify this file!  It was generated by 'nixos-generate-config'
# and may be overwritten by future invocations.  Please make changes
# to /etc/nixos/configuration.nix instead.
{ config, lib, pkgs, modulesPath, ... }:

{
  imports =
    [ (modulesPath + "/installer/scan/not-detected.nix")
    ];

  boot.initrd.availableKernelModules = [ "xhci_pci" "nvme" "usb_storage" "sd_mod" ];
  boot.initrd.kernelModules = [ ];
  boot.kernelModules = [ "kvm-intel" ];
  boot.extraModulePackages = [ ];

  # Enables DHCP on each ethernet and wireless interface. In case of scripted networking
  # (the default) this is the recommended approach. When using systemd-networkd it's
  # still possible to use this option, but it's recommended to use it in conjunction
  # with explicit per-interface declarations with `networking.interfaces.<interface>.useDHCP`.
  networking.useDHCP = lib.mkDefault true;
  # networking.interfaces.wlp0s20f3.useDHCP = lib.mkDefault true;

  nixpkgs.hostPlatform = lib.mkDefault "x86_64-linux";
  hardware.cpu.intel.updateMicrocode = lib.mkDefault config.hardware.enableRedistributableFirmware;
}
```

```nix
{ config, pkgs, ... }:

{
  imports =
    [ # Include tweaks from NixOS-Hardware.
      <nixos-hardware/lenovo/thinkpad/x1/7th-gen>
      # Include the results of the hardware scan.
      ./hardware-configuration.nix
    ];

  # Bootloader.
  boot.loader.systemd-boot.enable = true;
  boot.loader.efi.canTouchEfiVariables = true;
  boot.supportedFilesystems = [ "zfs" ];
  boot.initrd.systemd.enable = true;

  fileSystems."/" = {
    device = "tank/root";
    fsType = "zfs";
    };
  fileSystems."/nix" = {
    device = "tank/nix";
    fsType = "zfs";
    };
  fileSystems."/var" = {
    device = "tank/var";
    fsType = "zfs";
    };
```


Key features

REPRODUCABLE ✈️

DECLARATIVE 🤝

IMMUTABLE 🪂

IMMUTABLE 🤝

- Ensures stability by preventing unintended modifications.
- Changes require explicit updates
- Home manager: control over the `User environment`: `.config=ReadOnly`
- `nix-shell` and `nix develop` to “try-out” without installing permanently

```
impure ~/r/p/demo nix-shell -p slides figlet toilet
these 3 paths will be fetched (6.40 MiB download, 24.39 MiB unpacked):
  /nix/store/1r6710hls28v31lwnrjkqyz92vxay518-figlet-2.2.5
  /nix/store/577dmx72lbdb9br2ix06ssmwx3m4zk9q-slides-0.9.0
  /nix/store/idj0y5lz322s76y2zvxiavlgjxxqjhcc-toilet-0.3
copying path '/nix/store/1r6710hls28v31lwnrjkqyz92vxay518-figlet-2.2.5' from 'https://cache.nixos.org'...
copying path '/nix/store/577dmx72lbdb9br2ix06ssmwx3m4zk9q-slides-0.9.0' from 'https://cache.nixos.org'...
copying path '/nix/store/idj0y5lz322s76y2zvxiavlgjxxqjhcc-toilet-0.3' from 'https://cache.nixos.org'...

[nix-shell:~/repos/presentations/demo]$ figlet demo

    _
  _| | ___ _ __ ___   ___
 / _` |/ _ \ '_ ` _ \ / _ \
| (_| |  __/ | | | | | (_) |
 \__,_|\___|_| |_| |_|\___/


[nix-shell:~/repos/presentations/demo]$
```

```
[nix-shell:~/repos/presentations/demo]$ toilet demo

     #
  mmm#   mmm    mmmmm   mmm
 #" "#  #"  #   # # #  #" "#
 #   #  #""""   # # #  #   #
 "#m##  "#mm"   # # #  "#m#"



[nix-shell:~/repos/presentations/demo]$ exit
exit
impure ~/r/p/demo figlet
fish: Unknown command: figlet
! impure ~/r/p/demo
```


flake.nix

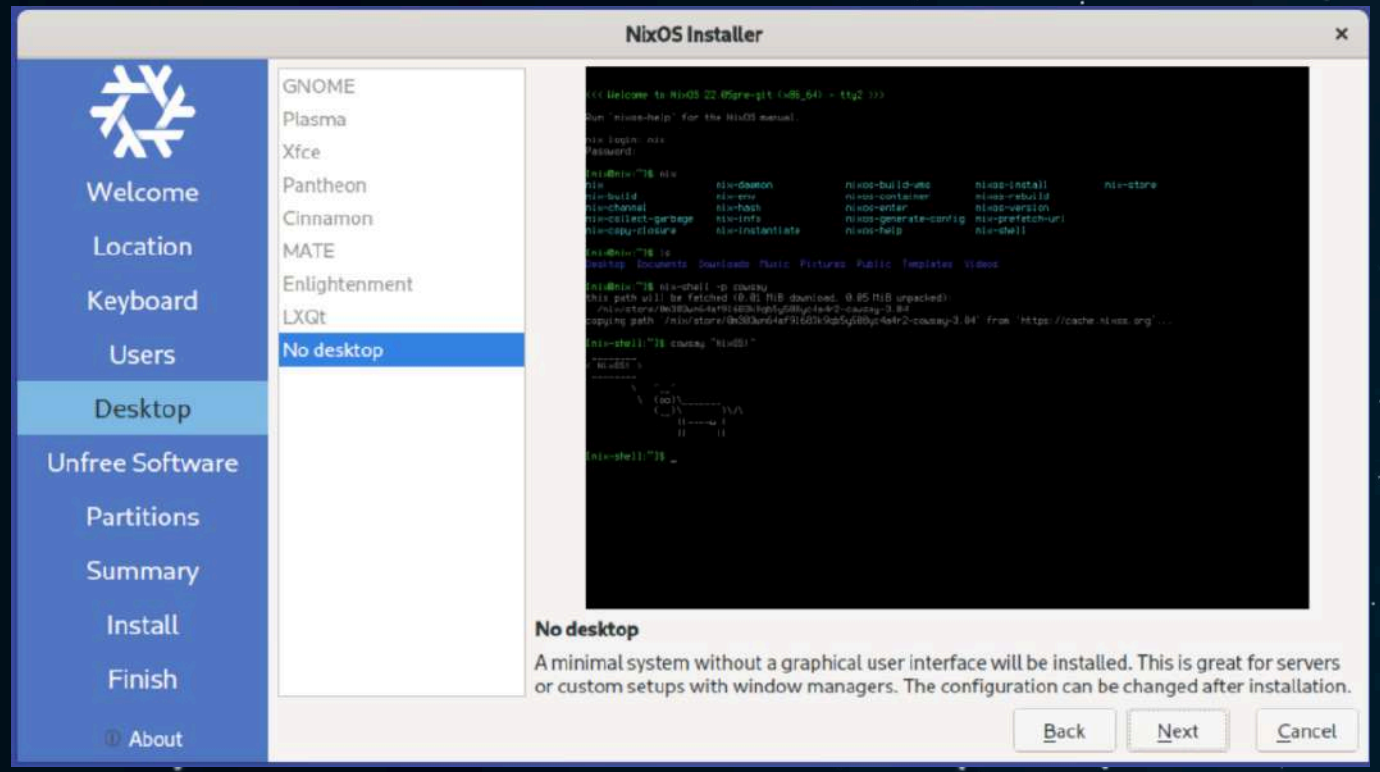
A development shell for Quarto and Python:

```
❯ cat flake.nix
{
  description = "A basic flake with a shell";
  inputs.nixpkgs.url = "github:NixOS/nixpkgs/nixpkgs-unstable";
  inputs.flake-utils.url = "github:numtide/flake-utils";

  outputs = { self, nixpkgs, flake-utils }:
    flake-utils.lib.eachDefaultSystem (system: let
      pkgs = nixpkgs.legacyPackages.${system};
      pythonEnv = pkgs.python3.withPackages (ps: with ps; [
          numpy
          pip
          pipx
          jupyter
```


4: Nixos Installation and configuration

Installation (1) TUI/GUI install

Load the USB stick with an image and from the Live environment
the installation is very basic and easy: - Choose your desktop as usual



Installation (2) default config

Packages and configs Let's take a look at the `/etc/nixos` folder first.

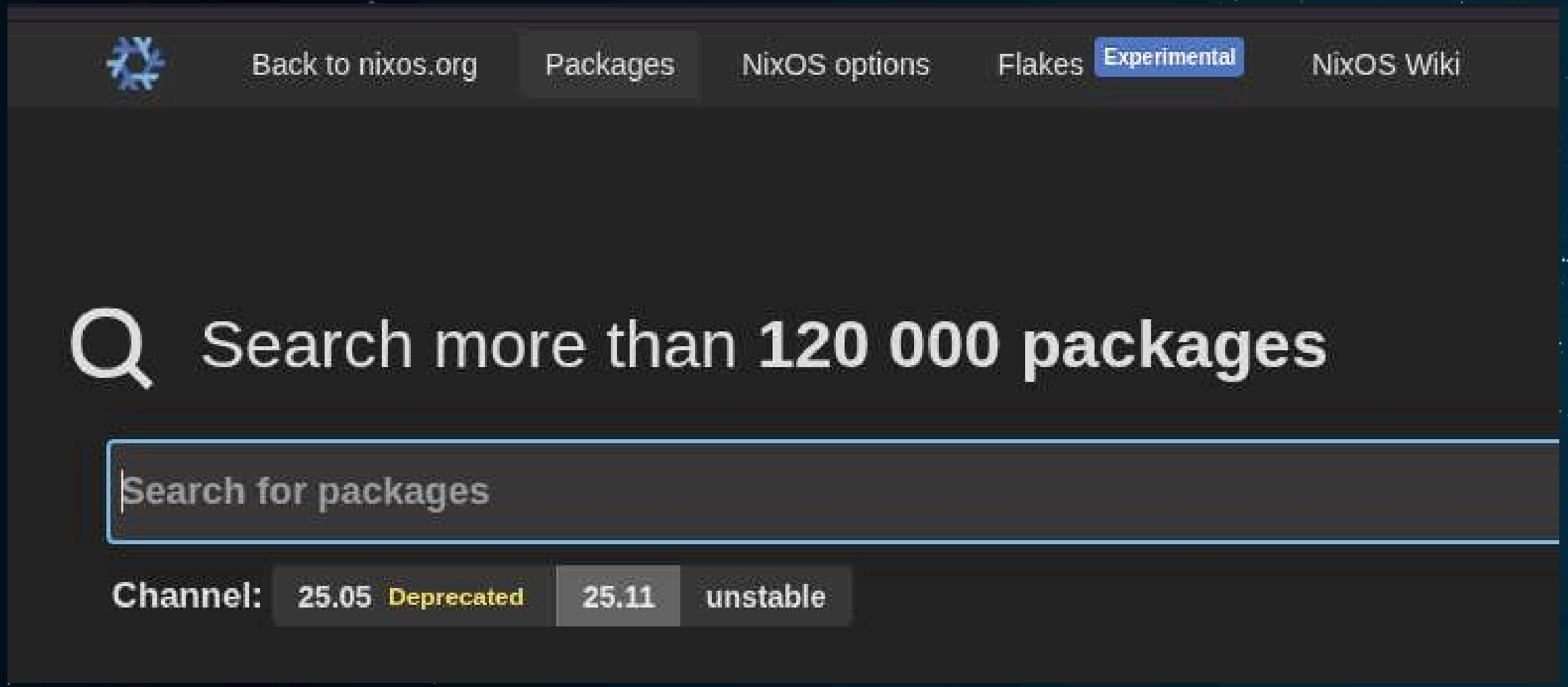
```
.
├── configuration.nix
└── hardware-configuration.nix
```

- result is a working basic config, booting to the choosen desktop
- the hardware-configuration is very esssential so keep it safe
- but everything can be re-generated

```
nixos-generate-config
```

NIXOS wiki: https://wiki.nixos.org/wiki/NixOS_Wiki

Installation (3) > 120.000 packages



Installation (4) add the packages

Add a service `openssh` and package `vscodium` for example to `configuration.nix`:

```
...
# Service
  services.openssh = {
    enable = true;
    # Disable SSH password log in
    passwordAuthentication = false;
  };
...
```

Note: the normal /etc/ssh/sshd.config is not needed

```
...
  # Install vscodium systemwide
  environment.systemPackages = with pkgs; [ vscodium ];
...
```

```
...
{
  # Install fzf for the user
  programs.fzf = {
    enable = true;
    enableFishIntegration = true;
  };
}
...
```


Installation (5) rebuild

```
# Edit your configuration
sudo nano /etc/nixos/configuration.nix
# Rebuild your system
sudo nixos-rebuild switch
reboot
```

- the configs can be named to identify them easily
- clean up and retention can be configured

Installation (6) roll back

```
# Edit your configuration
nix-env --rollback                # roll back a user environment
nixos-rebuild switch --rollback   # roll back a system environment

# git is used to store the config files ;)
git revert <hash>
```


5: Advanced topics and other snippets

My current setup

```
Shell:    fish, zsh
DM:       greetd / tiugreet / tty1
WM:       niri / added DankMaterialShell
Editor:   vi / doom-emacs / vscodium
Terminal: wezterm
Launcher: tofi / fuzzel
Browser:  vivalvi / waterfox / qutebrowser / zen
Theme:    stylix
ZFS:      ZFS supported
```


My config split in parts (1)

Root

```
❯ ls -la
drwxr-xr-x    - dude  8 Apr 07:42  .git
drwxr-xr-x    - dude  8 Apr 07:37  apps
drwxr-xr-x    - dude 19 Mar 09:41  backlog
drwxr-xr-x    - dude  8 Apr 07:40  hosts
drwxr-xr-x    - dude 19 Mar 09:41  imgs
drwxr-xr-x    - dude  2 Apr 11:42  modules
drwxr-xr-x    - dude 19 Mar 09:41  secrets
.rw-r--r--   22 dude 19 Mar 09:41  .gitignore
.rw-r--r-- 6.3k dude 19 Mar 09:41  README.md
.rw-r--r--  44k dude  2 Apr 10:49  flake.lock
.rw-r--r-- 5.9k dude 20 Mar 09:11  flake.nix
.rw-r--r-- 1.9k dude 19 Mar 09:41  justfile
.rw-r--r--    0 dude 19 Mar 09:41  test_branch_txt
```

Note: secrets stored in yaml using age and sops

Multiple host definitions

(some not in use)

```
❯ ll hosts/
Permissions Size User Date Modified    Name
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ latitude
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ letsnote
drwxr-xr-x     - dude 2026-03-20 09:22 ▪ ltdude
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ nixos-01
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ p14s
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ sg13
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ vaio
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ x1yoga
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ x13
drwxr-xr-x     - dude 2026-03-19 09:41 ▪ x61s
.rw-r--r--   13k dude 2026-04-08 07:45 ✲ default.nix
.rw-r--r--  1.7k dude 2026-03-25 09:44 ✲ home-manager-common-config.nix
```


My config split in parts (2)

My main laptop config

```
❯ cd hosts/ltdude/
❯ ll
Permissions Size User Date Modified    Name
.rw-r--r--  3.2k dude 2026-03-19 09:41 ❄ hardware-configuration.nix
.rw-r--r--   432 dude 2026-03-20 09:22 ❄ home.nix
```

The hardware config:

```nix
# Do not modify this file!  It was generated by ‘nixos-generate-config’
# and may be overwritten by future invocations.  Please make changes
# to /etc/nixos/configuration.nix instead.
{ config, lib, pkgs, modulesPath, ... }:
{
  imports =
    [ (modulesPath + "/installer/scan/not-detected.nix")
    ];


  boot.initrd.availableKernelModules = [ "xhci_pci" "nvme" "usb_storage" "sd_mod" ];
  boot.initrd.systemd.enable = true;
  boot.kernelModules = [ "kvm-intel" ];
  boot.extraModulePackages = [ ];
```


My config split in parts (3)

```
❯ cd apps/
❯ ls -la
drwxr-xr-x - dude 19 Mar 09:41 ▪ alacritty
drwxr-xr-x - dude 19 Mar 09:41 ▪ android
drwxr-xr-x - dude 19 Mar 09:41 ▪ appimage
drwxr-xr-x - dude 19 Mar 09:41 ▪ autoscreen
drwxr-xr-x - dude 19 Mar 09:41 ▪ autoscreen-gaming
drwxr-xr-x - dude 19 Mar 09:41 ▪ autoscreen-gnome
drwxr-xr-x - dude 19 Mar 09:41 ▪ backrest
drwxr-xr-x - dude 19 Mar 09:41 ▪ bat
drwxr-xr-x - dude  8 Apr 07:37 ▪ beeper
drwxr-xr-x - dude 19 Mar 09:41 ▪ boinc
drwxr-xr-x - dude 19 Mar 09:41 ▪ codex
drwxr-xr-x - dude 19 Mar 09:41 ▪ cursor
```

Apps to add when needed with specific configs

My config split in parts (4)

```
ltdude = mkHost {
    hostName = "ltdude";
    stateVersion = "25.11";
    profiles = [
      "laptop"
      "bootloader-systemd-boot"
      "niri"
      "dankMaterialShell"
<...>
    ];
    extraModules = [
      inputs.nixos-hardware.nixosModules.common-cpu-intel
      inputs.nixos-hardware.nixosModules.common-pc-laptop-ssd
      inputs.nixos-hardware.nixosModules.lenovo-thinkpad-x1-7th-gen
```

Combining it all to a specific config

Justfile for easy command reference

```
switch name:
  @echo "Rebuilding config for host {{thishost}} named: {{name}}"
  sudo nixos-rebuild switch --flake .#{{thishost}} --max-jobs auto  -p {{name}}

switch-remote-host host target:
  @echo "Rebuilding config for host {{host}} on target {{target}}"
  sudo -E nixos-rebuild switch --flake .#{{host}} --target-host $USER@{{target}} --sudo --ask-sudo-password

debug:
  ##echo "Rebuilding debug config for host $hostname"
  sudo nixos-rebuild switch --flake . --max-jobs auto  --show-trace --verbose

boot:
  @echo "Rebuilding config for host $hostname (available at next boot)"
```


Examples

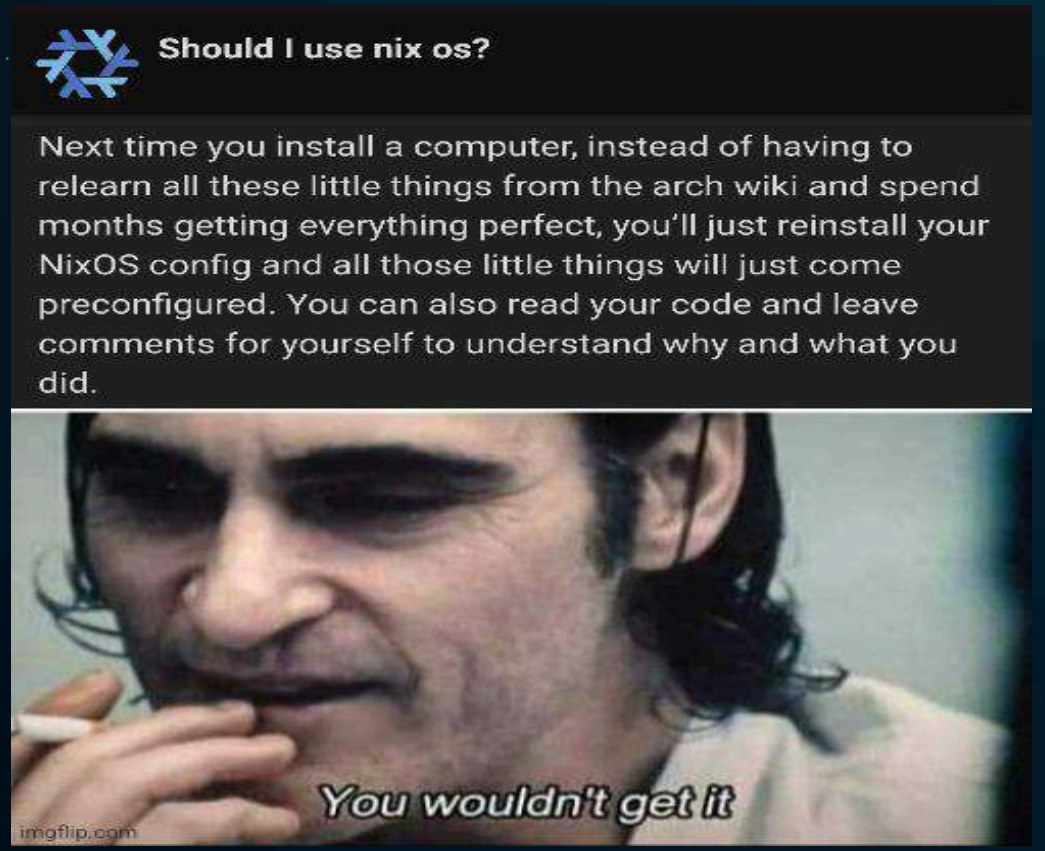
- The flake.nix file defines the flake for your project.
- The flake.lock pins all of the flake inputs—essentially the Nix dependencies—in your flake.nix file to specific Git revisions.

```
# get a basic envriroment

nix develop "github:DeterminateSystems/zero-to-nix#cpp"
Welcome to a Nix development environment for C++!

# get the flake.nix template
[dude@nixos:~/repos/nixos/ttt]$ nix flake init --template "github:DeterminateSystems/zero-to-nix#cpp-dev"
wrote: "/home/dude/repos/nixos/ttt/flake.lock"
wrote: "/home/dude/repos/nixos/ttt/flake.nix"

# activate it
[dude@nixos:~/repos/nixos/ttt]$ nix develop
[1/0/1 copied (176.6/176.6 MiB), 29.1 MiB DL] fetching source from https://cache.nixos.org
```


We made it to the end! 😄

Thanks for listening 👂

Questions ❓


Should I use nix os?
Next time you install a computer, instead of having to relearn all these little things from the arch wiki and spend months getting everything perfect, you'll just reinstall your NixOS config and all those little things will just come preconfigured. You can also read your code and leave comments for yourself to understand why and what you did.
You wouldn't get it
imgflip.com

Resources

- Awesome Nix: Awesome list of resources
- Offical Nixos Wiki: Nixos Wiki
- Learn Nix: Learn Nix
- NixBookOS, like Linux Mint: Nixbook OS
- From Zero to Nix: Zero2Nix
- Nixos forum: Forum
- Age Sops: Unmoved Centre