

Nix(OS)

Declarative builds and deployments

haennes

2024-03-27

Ansible? Nein Danke!

Inhaltsverzeichnis

1. Nix
2. Nix shells
3. Nun zum eigentlichen Thema **NixOS**
4. Flakes
5. Wie nutzt man Nix

Nix vs NixOS

- Nix:
- NixOS:

Nix vs NixOS

- Nix:
 - funktionale Programmiersprache
 - package manager (wie: apt, pip, appstores, ...)
- NixOS:

Nix vs NixOS

- Nix:
 - funktionale Programmiersprache
 - package manager (wie: apt, pip, appstores, ...)
- NixOS:
 - Linux distribution
 - nutzt Nix als:
 - package manager
 - configuration manager

Nix



Features von Nix

- alle Programme liegen in `/nix/store`
- vollständige dependencies von Paketen / pureness
- Nutzer können ohne Adminrechte Pakete installieren
- upgrades und rollbacks

Features von Nix


- alle Programme liegen in `/nix/store`
 - weniger suchen
 - mehrere Versionen
- vollständige dependencies von Paketen / pureness
- Nutzer können ohne Adminrechte Pakete installieren
- upgrades und rollbacks

Features von Nix

- alle Programme liegen in `/nix/store`
- vollständige dependencies von Paketen / pureness
 - immer die richtige Version einer Bibliothek, dank Hash nach Pfad
 - immer alle dependencies (sonst funktioniert die Software nicht)
- Nutzer können ohne Adminrechte Pakete installieren
- upgrades und rollbacks

Features von Nix

- alle Programme liegen in `/nix/store`
 - weniger suchen
 - mehrere Versionen
- vollständige dependencies von Paketen / pureness

▸ 
`/nix/store/avg4nfsj9shm81g0v142dfdlifq8j149-pdfpc-0.7.1-tex.drv`

- Nutzer können ohne Adminrechte Pakete installieren
- upgrades und rollbacks

Features von Nix

- alle Programme liegen in `/nix/store`
 - weniger suchen
 - mehrere Versionen
- vollständige dependencies von Paketen / pureness
 - immer die richtige Version einer Bibliothek, dank Hash nach Pfad
 - immer alle dependencies (sonst funktioniert die Software nicht)
- Nutzer können ohne Adminrechte Pakete installieren
- upgrades und rollbacks

Features von Nix

- alle Programme liegen in `/nix/store`
 - weniger suchen
 - mehrere Versionen
- vollständige dependencies von Paketen / pureness
 - immer die richtige Version einer Bibliothek, dank Hash nach Pfad
 - immer alle dependencies (sonst funktioniert die Software nicht)
- Nutzer können ohne Adminrechte Pakete installieren
- upgrades und rollbacks **später: Demo... also wenn genügend Zeit ist**

Nix shells

Warum brauch ich das?

Warum brauch ich das?

Hey, cooles Projekt, kann ich da mitprogrammieren?

Warum brauch ich das?

Hey, cooles Projekt, kann ich da mitprogrammieren?

Ja klar gerne, du musst dir nur
build-essential, *git*, *cmake* und *libsdl2-dev* installieren
dann 2x nen cmake-script laufen lassen
nen Ordner erstellen und dann das Compilat ausführen
~ [osp-magnum: A spaceship game](#)

WTF, Nein

Wie nutze ich sowas?

1. **EINER** schreibt eine “flake.nix”

Wie nutze ich sowas?

1. **EINER** schreibt eine “flake.nix” oder “shell.nix”

2.



```
$ nix develop
```

oder




```
$ nix-shell
```

3. Bereit zum Entwickeln

Wie nutze ich sowas?

1. **EINER** schreibt eine “flake.nix”

2.



```
$ nix run .#
```

3. das Programm läuft

Mega



Warum nicht Alles so konfigurieren?

Nun zum eigentlichen Thema
NixOS

Features

- alle Programme liegen in /nix/store
- vollständige dependencies von Paketen / pureness
- **nicht-Admins** können Pakete installieren
- upgrades und rollbacks **SYSTEMWIDE**
- konfiguriere dein gesamtes System in **einer Datei**

- Ein Dateiformat um Alles zu konfigurieren
- 100% deklarativ und reproducible

- Ein Dateiformat um Alles zu konfigurieren
vom Bootloader über die installierte Software bis hin zu deiner VPN
- 100% deklarativ und reproducible

- Ein Dateiformat um Alles zu konfigurieren
- Teile deine Konfiguration mit anderen Systemen
- 100% deklarativ und reproducible

- Ein Dateiformat um Alles zu konfigurieren
- Teile deine Konfiguration mit anderen Systemen oder auch Menschen
- 100% deklarativ und reproducible

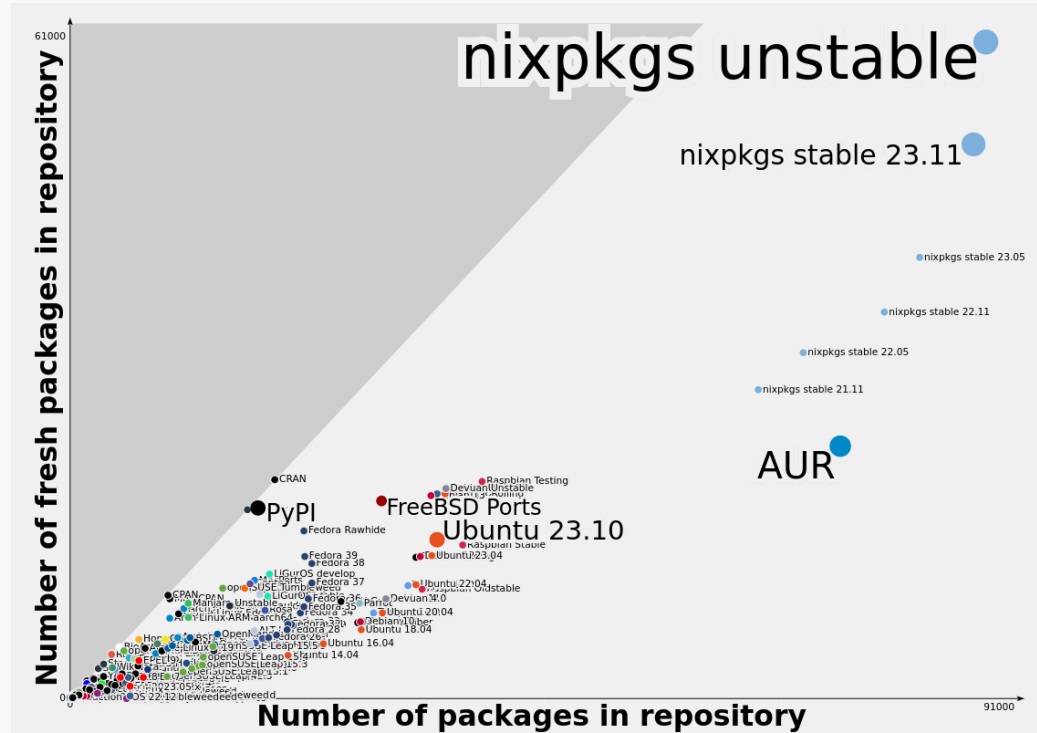
- Ein Dateiformat um Alles zu konfigurieren
- Teile deine Konfiguration mit anderen Systemen oder auch Menschen
- 100% deklarativ und reproducible
- System kaputt?

Vorteile

- Ein Dateiformat um Alles zu konfigurieren
- Teile deine Konfiguration mit anderen Systemen oder auch Menschen
- 100% deklarativ und reproducible
- System kaputt? rollbacks

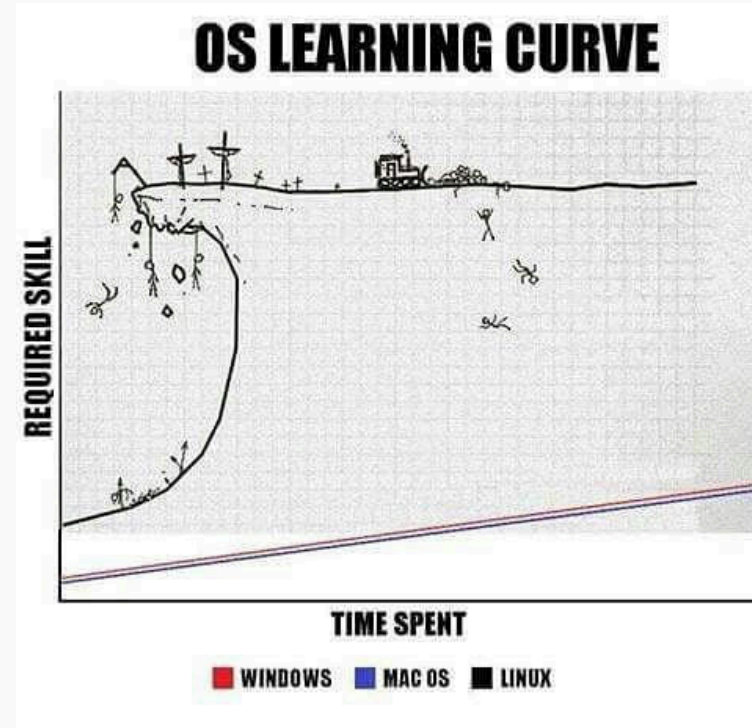
⇒ keine Angst vor updates

Vorteile



Nachteile

- Lernkurve...
- dynamische libraries schwieriger



kleiner Auszug aus einer Konfiguration

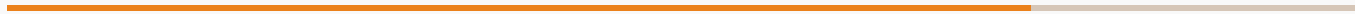
```
{ config, pkgs, ... }: {
  imports = [ ./hardware-configuration.nix ];
  boot.loader.grub.enable = true;
  networking.hostName = "nixos";
  time.timeZone = "Europe/Berlin";
  sound.enable = true;
  users.users.hannses = {
    isNormalUser = true;
    description = "hannses";
    extraGroups = [ "networkmanager" "wheel" ];
    packages = with pkgs; [
      firefox
    ];
  };
  environment.systemPackages = with pkgs; [
    gcc
  ];
  system.stateVersion = "23.11";
}
```

/etc/nixos/configuration.nix

```
{config, lib, ...}: {
  boot.initrd.availableKernelModules = [ "nvme" "xhci_pci" ];
  boot.kernelModules = [ "kvm-amd" "amdgpu" ];
  fileSystems."/ " =
    { device = "/dev/disk/by-uuid/c4a7f0..4b389";
      fsType = "ext4";
    };
  fileSystems."/boot" =
    { device = "/dev/disk/by-uuid/80..77";
      fsType = "vfat";
    };
  networking.useDHCP = lib.mkDefault true;
  nixpkgs.hostPlatform = lib.mkDefault "x86_64-linux";
}
```

/etc/nixos/hardware-configuration.nix

Flakes



Was sind Flakes?

- spezielle Nix expressions
- können folgendes beschreiben:
 - Systeme
 - Pakete / Apps
 - devShells (nix-shell)
 - formatter, templates, checks, overlays, modules, ...

Beispiel

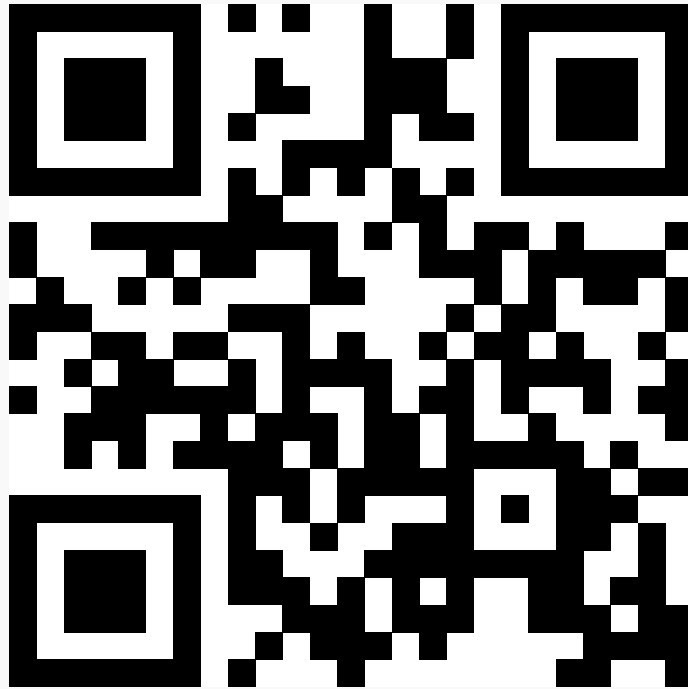
```
{
  inputs.nixpkgs.url = github:NixOS/nixpkgs/nixos-unstable;
  outputs = { self, nixpkgs, ... }@attrs: {
    nixosConfigurations.fnord = nixpkgs.lib.nixosSystem {
      system = "x86_64-linux";
      specialArgs = attrs;
      modules = [ ./configuration.nix ];
    };
  };
}
```

Wie nutzt man Nix

```
sh <(curl -L https://nixos.org/nix/install) --daemon
```

Weiterführende Links

[Nix & NixOS | Declarative builds and deployments](#) [NixOS: Everything Everywhere All At Once - YouTube](#) [Ultimate NixOS Guide | Flakes |](#)
[Home-manager - YouTube](#) [NixOS in 60 seconds - YouTube](#) [NixOS Wiki](#)



PDF zum Download