NixOS

Befehle

Apps suchen unter https://search.nixos.org/packages

nix-shell -p bitwarden	zum installieren von Bitwarden
nix-env --install bitwarden	installiert Bitwarden im System permanent
sudo nixos-rebuild switch --upgrade	Update des Systems
sudo nixos-rebuild switch --rollback	Vorherigen Stand wiederherstellen.

Programme Dauerhaft installieren

```
sudo nano /etc/nixos/configuration.nix
```

Füge das Programm hinzu

```
environment.systemPackages = with pkgs; {
  htop
  # andere Pakete hier hinzufügen
}
```

Aktualisiere die NixOs-Konfiguration

```
sudo nixos-rebuild switch
```

Nix Konfiguration im Homeordner

```
mkdir ~/nixos
sudo cp -r /etc/nixos/* ~/nixos/
```

kopiert die einstellungen in den Homeordner. Der kann leichter gesichert werden.

Git erstellen

```
cd ~/nixos
git init
git add .
git commit -m "Erster Stand"
```

2. Online-Repo erstellen (GitHub, GitLab, Gitea, Codeberg …)

3. Verknüpfen:

```
git remote add origin <URL>
git push -u origin main
```

Ab jetzt sicherst du Änderungen mit:

```
git add .
git commit -m "Update"
git push
```

🔌 Weg 2: Backup auf USB-Stick

```
git clone ~/nixos /media/usb/nixos-backup
```

☁️ Weg 3: Backup in einer Cloud (OneDrive, Dropbox, Nextcloud)

Den ganzen Ordner `~/nixos` einfach in einen Sync-Ordner legen:

```
mv ~/nixos ~/OneDrive/nixos
```

→ Git + Cloud = doppelte Sicherheit.

🔄 Wie stelle ich das System später wieder her?

Neue Maschine? Festplatte neu?

Du installierst NixOS minimal und machst danach:

```
git clone <repo-url> ~/nixos
cd ~/nixos
sudo nixos-rebuild switch --flake .
```

□□ Boom — dein ganzes System ist zurück.

□□ Was passiert mit deinen eigenen Skripten?

Wenn du deine eigenen Skripte in dein Git-Repo legst, z. B.:

```
~/nixos/scripts/meinskript.sh
```

und in der Nix-Konfiguration referenzierst:

- werden sie **im Repo versioniert**
- beim Rebuild **in den Nix Store kopiert**
- bei Neuinstallation **automatisch wiederhergestellt**

Damit ist das Problem „/home/hermann/bin existiert nicht“ sauber gelöst.

Du kannst deinen Konfigurationsordner z. B. so anlegen:

```
~/Nextcloud/nixos/
```

Und dann:

```
cd ~/Nextcloud/nixos
git init
git add .
git commit -m "Initial commit"
```

✔ Wiederherstellung extrem einfach

Neue Maschine oder Neuinstallation?

```
cd ~/Nextcloud/nixos
sudo nixos-rebuild switch --flake .
```

□ Git + Nextcloud = doppelte Sicherheit

Der beste Setup ist:

1. Ordner in Nextcloud:

   ```
   ~/Nextcloud/nixos
   ```

2. Git-Repo in diesem Ordner:

   ```
   cd ~/Nextcloud/nixos
   git init
   ```

Wichtiges Detail:

Du solltest den Ordner nicht gleichzeitig in `/etc/nixos` haben

Stattdessen:

1. **Arbeite und speichere in:**

   ```
   ~/Nextcloud/nixos
   ```

2. **Und verknüpfe `/etc/nixos` mit deinem Ordner**

z. B. via Symlink:

```
sudo rm -r /etc/nixos
sudo ln -s ~/Nextcloud/nixos /etc/nixos
```

Dann ist `/etc/nixos` nur ein Zeiger – und du bearbeitest alles dort, wo Nextcloud es sichert.

Nixpaket selber machen

```
{ stdenv }:

stdenv.mkDerivation {
  name = "hello";
  src = ./hello.sh;
  installPhase = ''
    mkdir -p $out/bin
    cp $src $out/bin/hello
  '';
```

```
}
```

Ein Skript selber einbinden über Flake

```
environment.systemPackages = [ self.packages.${system}.meinProgramm ];
```

Flakes

```
{
  description = "Meine NixOS-Konfiguration";

  inputs = {
    nixpkgs.url = "github:nixos/nixpkgs/nixos-unstable";
  };

  outputs = { self, nixpkgs }: {
    packages.x86_64-linux.defaultPackage =
      nixpkgs.legacyPackages.x86_64-linux.hello;
  };
}
```

Konfiguration von einstellungen

über den Home-Manger

```
programs.bash.enable = true;

programs.bash.shellAliases = {
  ll = "ls -la";
};

programs.bash.initExtra = ''
  export EDITOR=nvim
'';
```

Ja, Hermann — **du kannst die Fish-Shell absolut mit Home Manager verwenden** und das funktioniert sogar **sehr komfortabel**, viel besser als manuelles Dotfile-Gebastel.

Ich zeige dir genau:

- ✔ ob es geht (JA)
- ✔ wie du Fish aktivierst
- ✔ wie du deine eigene Konfiguration einbringst
- ✔ wie du Plugins und Aliases verwaltest
- ✔ warum Home Manager das perfekt unterstützt

✅ Ja, Home Manager unterstützt die Fish-Shell vollständig

Home Manager hat ein eigenes Modul:

programs.fish.enable = true;

Damit erzeugt Home Manager automatisch:

- deine `config.fish`
- dein Fish-Init-Skript
- deine Aliases
- deine Funktionen
- deine Plugins (über `oh-my-fish` oder Fisher)

Du musst nicht mehr manuell in `~/.config/fish/config.fish` arbeiten.

🛠️ Beispiel: Fish in Home Manager aktivieren

In deiner Home-Manager-Config (z. B. `home/hermann.nix`):

```
programs.fish = {
  enable = true;
  interactiveShellInit = ''
    echo "Hallo Herrmann, Welcome to Fish!"
  '';
  shellAliases = {
    ll = "ls -alh";
    gs = "git status";
  };
};
```

Damit erledigt Home Manager:

- Shell aktivieren
- Konfiguration erzeugen
- Aliases setzen
- Init-Code einfügen

Fish als Standardshell setzen

Zusätzlich in NixOS selbst:

users.users.hermann.shell = pkgs.fish;

Dann ist Fish auch beim Login (TTY oder GUI) aktiv.

Fish Plugins einbinden

Du kannst direkt Plugins definieren:

```
programs.fish.plugins = [
  {
    name = "fzf-fish";
    src = pkgs.fzf-fish;
  }
];
```

Oder mit Fisher:

```
programs.fish.fisherPlugins = [
  "jorgebucaran/fisher"
  "jethrokuan/z"
  "PatrickF1/fzf.fish"
];
```

Home Manager installiert alles sauber und reproduzierbar.

Eigene Konfiguration einfügen

Wenn du eigene Fish-Funktionen oder eigene Konfigdateien hast:

Variante A — Inline-Konfiguration:

```
programs.fish.interactiveShellInit = ''
  set -g EDITOR nvim
'';
```

Variante B — ganze Dateien einbinden:

home.file.".config/fish/functions/myfun.fish".source = ./functions/myfun.fish;
``

Home Manager legt diese Dateien automatisch an.

Warum Fish + Home Manager so gut zusammenpassen

- Keine Dotfiles mehr kopieren
- Keine Konflikte mehr
- Identische Shell überall (Laptop, PC, Server)
- Rollbacks funktionieren
- Plugins sauber versioniert
- Backups über Git + Nextcloud automatisch gesichert

Viele NixOS-User nutzen Fish **genau wegen Home Manager**, weil das Zusammenspiel perfekt ist.

Wichtiger Hinweis

Wenn Home Manager aktiv ist, solltest du **niemals** die Datei:

```
~/.config/fish/config.fish
```

manuell bearbeiten.

Home Manager überschreibt sie bei jedem „Rebuild“.
Alle Änderungen gehören in die Home-Manager-Config.

Kontrollfrage

Wenn du Fish über Home Manager aktivierst — **wo sollten deine Aliases, Plugins und Konfigurationen gespeichert werden?**

a) Direkt in `~/.config/fish/config.fish`
b) In der Home-Manager-Konfiguration (`home/hermann.nix`)
c) In `/etc/nixos/configuration.nix`
d) In `/home/hermann/bin`

Welcher Buchstabe ist richtig?

Alles klar, Hermann — ich gebe dir jetzt die **sauberste, stabilste und wirklich praxistaugliche Best-Practice-Methode**, um ein vollständig funktionierendes NixOS-System **auf einem neuen Rechner oder nach einer Neuinstallation** wiederherzustellen.

Du bekommst:

- **den genauen Ablauf**, Schritt für Schritt
- **was du vorher sichern musst**
- **was NixOS rekonstruiert**
- **wie du Nextcloud + Git effektiv nutzt**
- **den exakten Wiederherstellungsprozess** (minimal, klar, idiotensicher)

Das ist die Methode, die 99 % der erfahrenen NixOS-User nutzen.

✅ Best Practice: Ein neues NixOS wiederherstellen, wenn die Konfiguration in Nextcloud liegt

🗂️ Vorher (auf deinem Hauptsystem): Was du in Nextcloud brauchst

In `~/Nextcloud/nixos` liegt idealerweise:

```
nixos/
  flake.nix
  flake.lock
  nixos/
    configuration.nix
    hardware-configuration.nix
  home/
    hermann.nix
  scripts/
    eigene Skripte
  modules/
    optionale Module
```

UND:
Das Ganze als **Git-Repository**, damit du Versionen hast.

□ Schritt-für-Schritt Wiederherstellung auf einem frisch installierten NixOS

□□ Schritt 0 – Minimal NixOS installieren (ohne Konfiguration)

Während der grafische Installer fragt:

- Partitionierung → egal
- User erstellen → ja
- Desktop → egal
- Konfiguration erzeugen lassen → JA (wegen hardware-configuration.nix)

Nach der Installation hast du ein absolut minimales Grundsystem.

□□ Schritt 1 – Bei deiner Nextcloud anmelden

Nach dem ersten Boot:

1. Starte dein System
2. Melde dich in Nextcloud an
3. Warte, bis der Ordner `~/Nextcloud/nixos` synchronisiert ist

> **Du brauchst KEINE NixOS-Konfig im System — die kommt jetzt komplett aus Nextcloud.**

□□ Schritt 2 – /etc/nixos durch deine Nextcloud-Version ersetzen

1. Alte config entfernen:

sudo rm -rf /etc/nixos

2. Symlink auf deine Nextcloud-Konfig setzen:

sudo ln -s ~/Nextcloud/nixos /etc/nixos

Ab jetzt ist `/etc/nixos` nur noch ein Zeiger auf deine Nextcloud-Config.

□□ Schritt 3 – NixOS rebuild

Wenn du Flakes nutzt (empfohlen):

sudo nixos-rebuild switch --flake /etc/nixos

UND DU BIST FERTIG.

□□ **Dein komplettes System ist wiederhergestellt.**

Was wird automatisch gesetzt?

- alle Systempakete
- alle Dienste
- alle Netzwerkeinstellungen
- alle Nutzer
- deine Shell (z. B. Fish)
- deine Home-Manager-Konfiguration
- deine Fonts
- deine Window-Manager- oder Desktop-Konfiguration
- deine Skripte (wenn sie als Pakete eingebunden sind)
- deine Dotfiles (generiert von Home Manager)
- etc.

Alles, was DU definiert hast.

❌ Was wird NICHT automatisch wiederhergestellt?

Nur:

- private Dateien (`~/Documents`, `~/Pictures`, etc.)
- Downloads
- Medien
- zufällige Dateien im Home-Verzeichnis

Aber die liegen **in deiner Nextcloud**, also sind sie sowieso wieder da.

Der Rest ist komplett durch NixOS und Home Manager reproduziert.

🎉 Schritt 4 – Fertig. Wirklich.

Kein Rumkopieren.
Kein händisches Nachkonfigurieren.
Keine Dotfiles anpassen.
Keine Programme neu installieren.
Kein Chaos.

□ Warum ist diese Methode „Best Practice“?

Weil:

- **Nextcloud** deine Daten sichert
- **Git** deine komplette Versionierung speichert
- **NixOS** dein System reproduzierbar macht
- **Home Manager** dein Home reproduzierbar macht
- **Flakes** alles sauber einfrieren

Diese Kombination ist:

□□ **das stabilste Desktop-Setup, das du in Linux bekommen kannst**
□□ **komplett unzerstörbar**
□□ **100 % portabel auf jeden Rechner**

□□ Beispiel: Neuinstallation in der Praxis

- Neuer PC
- NixOS minimal installieren (egal wie)
- Nextcloud anmelden
- Warten bis `~/Nextcloud/nixos` synchronisiert ist
- Symlink setzen
- Rebuild
- Fertig

Dauer: **5–10 Minuten**.
Dein ganzes System ist identisch wie vorher.

Kontrollfrage

Warum setzt man auf einem neuen NixOS nach der Installation **einen Symlink** von:

```
/etc/nixos → ~/Nextcloud/nixos
```

statt die Dateien einfach zu kopieren?

(Ein Satz reicht.)

Revision #6
Created 19 May 2025 20:41:44 by Hermann
Updated 23 February 2026 16:43:39 by Hermann