

# Immutable Linux Desktops

Produktiv Arbeiten mit Fedora Silverblue, Chezmoi und  
Distrobox

Christoph Stoettner

2025-06-19

# Christoph Stoettner (stoeps)



✉ stoeps@stoeps.de  
in christophstoettner  
RSS stoeps.de  
@stoeps@infosec.exchange

- seit 30 Jahren was mit Computern
  - Amiga, OS/2, Linux
  - Beruflich auch Windows (wenn es sein muss)
- Linux / OSS seit 1994/1995
  - Linux Kernel < 1.0
  - Slackware
- mag `vi`, `vim`, `neovim`
  - Zu doof für `emacs`
  - Was ist `nano`?

Wenn ein Prozess Excel enthält, ist der Prozess kaputt.

# Agenda

Immutable OS .....	9
Eigene Images bauen .....	20
Boot Prozess .....	23
Distrobox .....	24
Chezmoi .....	31
Praktische Beispiele .....	39
Tips & Tricks .....	44
Vielen Dank! .....	53
Fragen .....	54

# Disclaimer

- Ich will niemanden missionieren
- Wenn Ihr glücklich mit eurer Distribution seid, bleibt dabei!
- Man kann das auch mit NixOS machen

*JUST IN CASE YOU'RE STILL NOT  
SURE WHETHER YOU'RE IN A  
SOFTWARE PROJECT*

*WAIT UNTIL YOU HEAR THIS:*

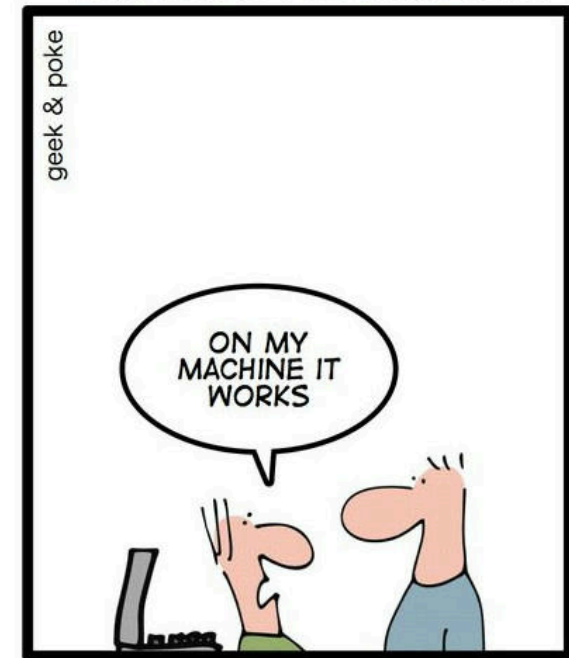


Figure 1: Geek & Poke<sup>1</sup>

# Ausgangslage

- Mehrere Rechner
  - Linux
  - Windows (mit WSL)
- Dotfiles / Konfiguration in Git
- Oft fehlt das **eine** Tool, das am anderen Rechner installiert ist
- Programmreste durch testweise installierte Tools
- Versuch mit Ansible
  - siehe auch Froscon 2023 - Dotfiles verwalten<sup>2</sup>

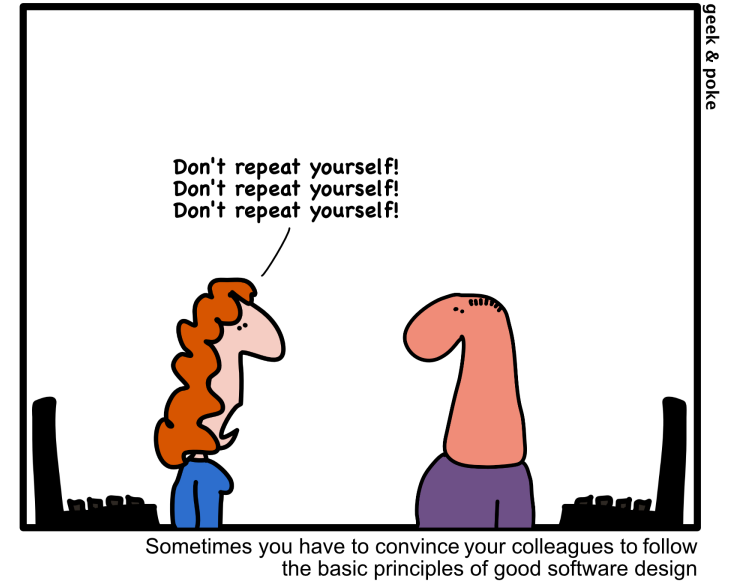


Figure 2: Geek & Poke<sup>1</sup>

# Ankündigung Bluefin

- YT videos von Jorge Castro (CNCF) machten neugierig
- Blogpost mit Ankündigung<sup>3</sup>
- Project Bluefin<sup>4</sup>
- GitHub Repository von Bluefin<sup>5</sup>

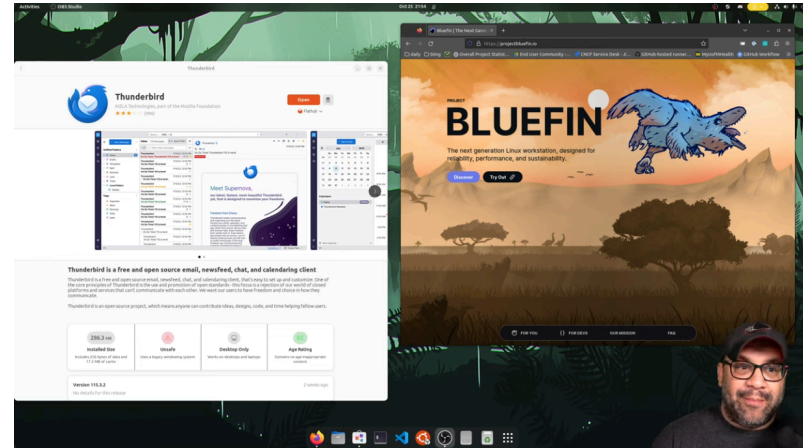


Figure 3: Announcement of Project Bluefin<sup>6</sup>

# Fedora Atomic Desktops

- Silverblue (Gnome)
- Kinoite (KDE)
- Sway Atomic
- Budgie Atomic
- Cosmic Atomic

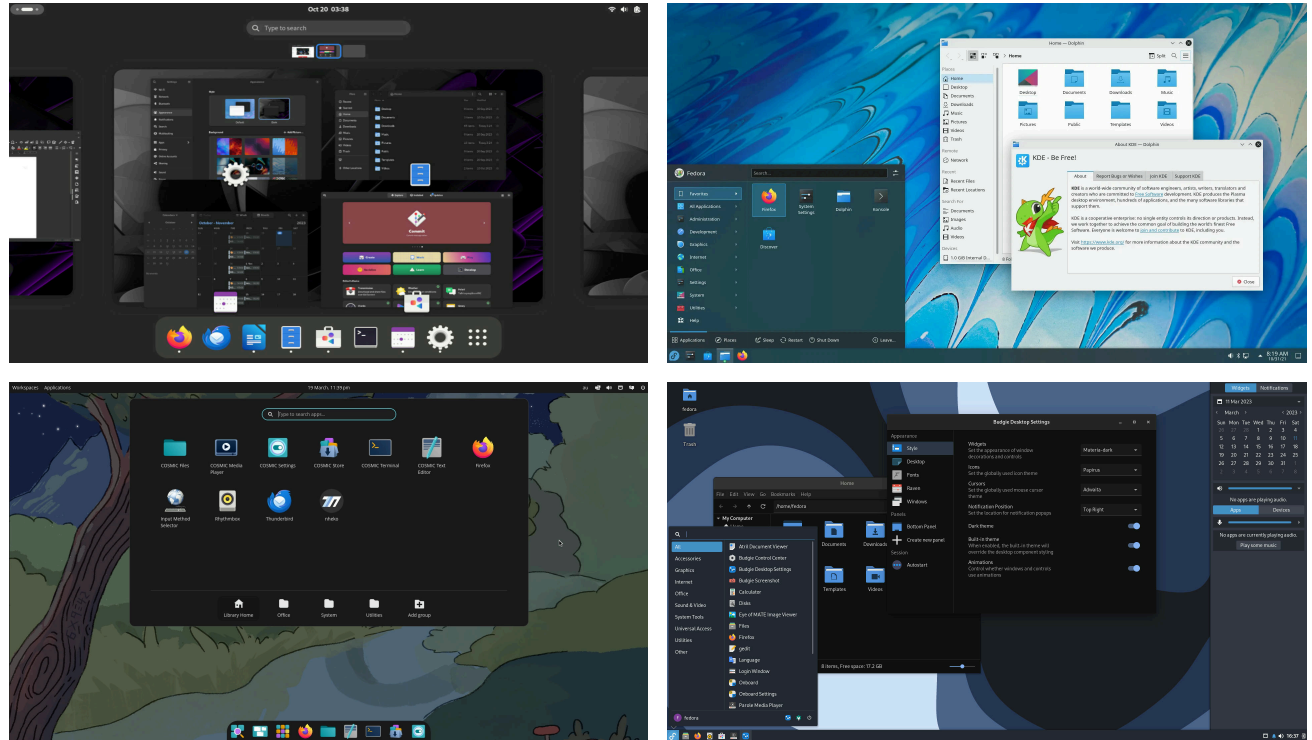


Figure 4: Screenshots: Silverblue, Kinoite, Cosmic & Budgie<sup>7</sup>

# Universal Blue

- Universal Blue
- Bluefin (Gnome)
- Aurora (KDE)
- Bazzite (Gaming)
- Images
  - ▶ Nvidia Support
  - ▶ Steamdeck
  - ▶ Framework Laptop
  - ▶ Asus, Surface uvm.

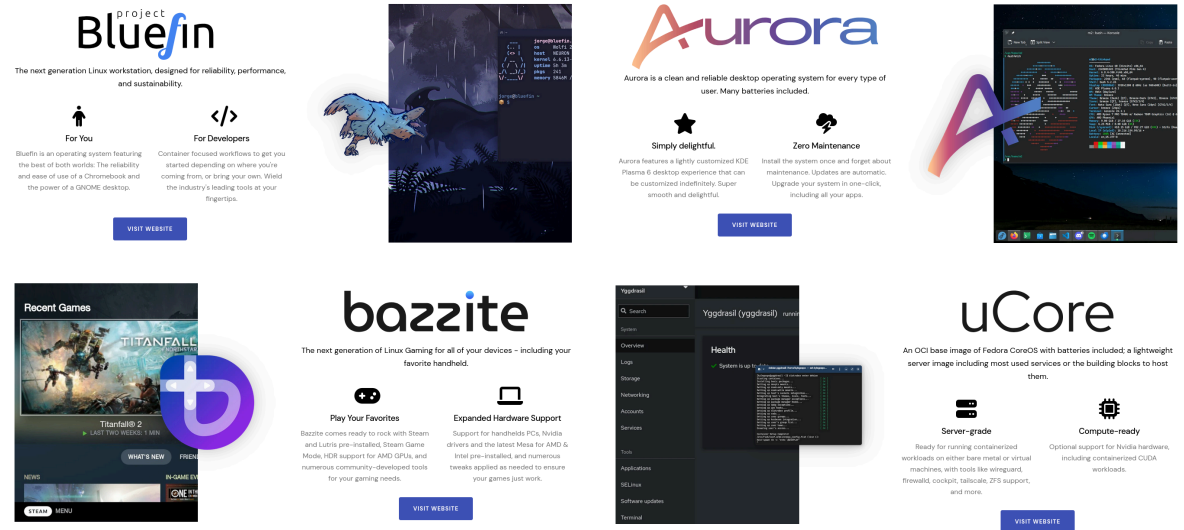


Figure 5: Screenshots: Bluefin, Aurora, Bazzite, uCore<sup>8</sup>

# Universal Blue

- Universal Blue ist der automatisierter Herstellungsprozess für
  - Desktop- und Server-Betriebssysteme
  - mit der Zuverlässigkeit eines Chromebooks
  - der Flexibilität traditioneller Linux-Desktops
- Das Projekt ermöglicht es Cloud-Infrastruktur-Experten
  - ihre Fähigkeiten für Linux-Desktop-Systeme zu nutzen
  - fungiert als “Desktop DevOps Team”
- Build Skripte um eigene Images zu bauen



Figure 6: Logo Universal Blue

# Immutable OS

---

# Was zeichnet ein immutable OS aus?

- **Schreibgeschützt** / *Read-only File System*
  - das laufende System kann nicht direkt verändert werden
- **Atomare Updates** / *Atomic Updates*
  - Updates werden vollständig angewendet oder gar nicht
- **Vorhersehbar** / *Reproducible*
  - Das Kernbetriebssystem ändert sich nicht
  - Verhalten ist auf verschiedenen Geräten vorhersehbar
- **Isolierte Anwendungen** / *Isolated Applications*
  - Anwendungen sind vom OS und voneinander isoliert

# Vorteile

- **Sicherheit** / *Security*
  - Änderungen stark begrenzt
- **Stabilität** / *Stability*
  - Systemdateien können nicht versehentlich verändert werden
  - Keine teilweise ausgeführten Updates / Instabile Zustände
- **Reproduzierbarkeit**
  - Es einfacher, das System zu testen, zu prüfen und zu verifizieren
  - Troubleshooting am eigenen System möglich
- **Cloud native**
  - Anpassung mittels Containerfiles und Github Actions

# Nachteile

- **Reduzierte Flexibilität**
  - Anpassung am System erfordern Aufwand
  - Müssen im Image erfolgen oder überlagert werden
- **Eingeschränkte Kompatibilität**
  - `/opt` ist z.B. nicht direkt schreibbar
- **Speicheranforderungen**
  - Update-Mechanismen erfordern Speicherung von Snapshots
- **Entwicklererfahrung**
  - Dockerfile / Containerfile (eh nur Yaml)

# Schlüsseltechnologien

## OSTree

- Git-ähnliches Versionierungssystem für Betriebssysteme
- Atomare Updates und Rollbacks
- Entwickelt von Red Hat
- Ermöglicht Verwaltung mehrerer Systemzustände

## rpm-ostree

- Kombination aus RPM-Paketmanagement und OSTree
- Hybrides Image-/Paketsystem
- Lokale Paketschichtung (Layering) möglich

# Schlüsseltechnologien

## Flatpak

- Moderne Anwendungsverteilung
- Sandboxing und Abhängigkeitsisolation
- Desktop-Integration

## Container Runtime

- Docker, Podman für Anwendungsisolierung
- Toolbox/Distrobox für Entwicklungsumgebungen

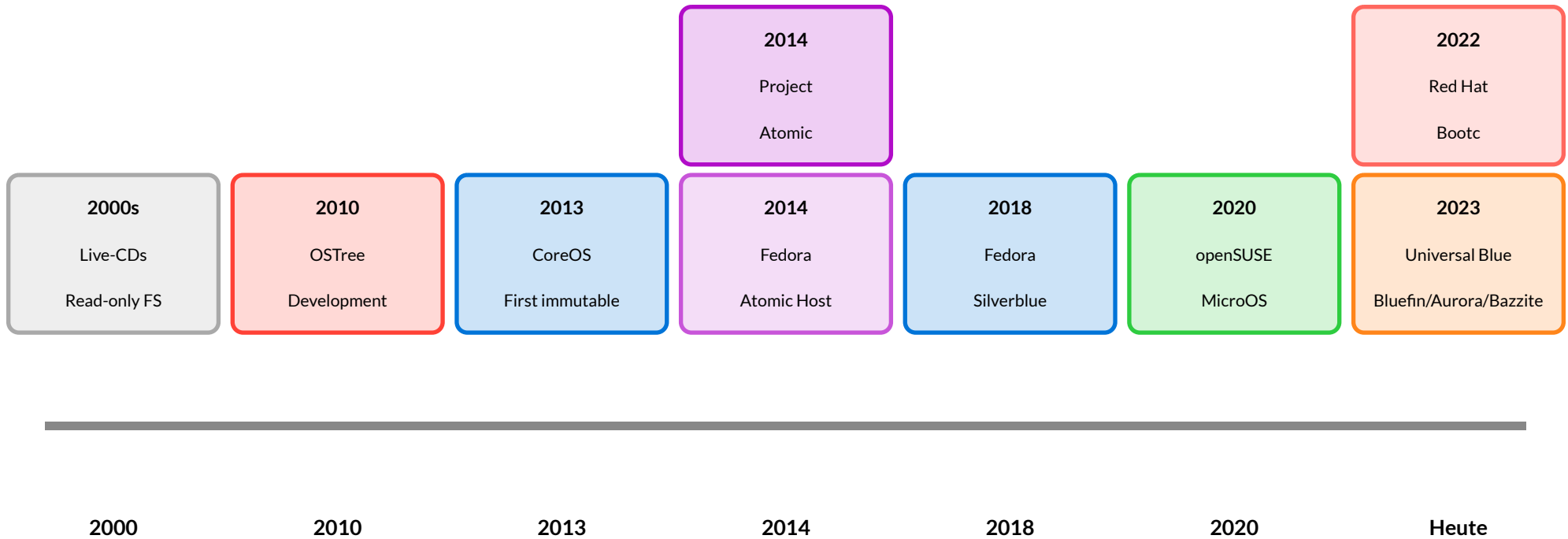
## Bootc

- Container-native Betriebssystemverwaltung

# Schlüsseltechnologien

- OCI-Container als OS-Image-Basis
- Vereinfacht CI/CD für Betriebssysteme
- Nachfolger der CoreOS-Architektur

# Entwicklungsgeschichte



# Warum Immutable OS wählen?

## Für Entwickler

- Konsistente Entwicklungsumgebungen
- Einfache Reproduzierbarkeit zwischen Systemen
- Sichere Experimente ohne Systemgefährdung
- Container-native Workflows

## Für IT-Administratoren

- Reduzierter Wartungsaufwand

## Für Endnutzer

- Stabileres System mit weniger Problemen
- Moderne Anwendungen via Flatpak
- Weniger Systembrüche nach Updates
- “Es funktioniert einfach”

# Warum Immutable OS wählen?

- Vorhersagbare Update-Zyklen
- Einfache Rollback-Möglichkeiten
- Bessere Compliance und Sicherheit
- spans both columns

# Verzeichnisse

- / immutable / **readonly**
- /etc
  - schreibbar
  - überschreibt Einstellungen aus /usr/etc
  - Einstellungen aus Installation
  - **Backup!**
- /home
  - Home-Verzeichnisse in /var/home
- /opt
  - nicht schreibbar
  - erfordert manchmal etwas Aufwand
  - /var/opt
- /root -> /var/root/home
- /usr **readonly**
- /usr/local -> /var/usrlocal
- /var

# Eigene Images bauen

---

# Blue Build template

- universal blue image template
- blue build image template
  - ublue-stoepe repository

```
---
name: ublue-stoepe-bluefin-rs
description: My ublue image for my personal and
working devices
base-image: ghcr.io/brickman240/tuxedo-bluefin-dx
image-version: latest

modules:
- from-file: common_modules/files.yml
- from-file: common_modules/chezmoi.yml
- from-file: common_modules/default-flatpaks.yml
- from-file: common_modules/fonts.yml
- from-file: common_modules/gschema-overrides.yml
- from-file: common_modules/rpm-ostree.yml
- from-file: common_modules/rpm-ostree-gnome.yml
- from-file: common_modules/systemd.yml
- from-file: common_modules/script.yml
- from-file: common_modules/containerfile.yml
- type: signing
```

# Pakete installieren

- Pakete im Haupt-OS
- Besser als `rpm-ostree` Layer
  - kostet Zeit bei Updates

```
common-modules/rpm-ostree.yml
```

```
type: rpm-ostree
```

```
repos:
```

```
- https://copr.fedorainfracloud.org/coprs/pgdev/  
ghostty/repo/fedora-42/pgdev-ghostty-fedora-42.repo
```

```
install:
```

```
- firewall-config  
- ghostty  
- guestfs-tools  
- libvirt-daemon  
- lm_sensors  
- pass  
- pipewire  
- playerctl  
- qemu-kvm  
- realtime-setup  
- tcpdump  
- wireguard-tools  
- wireshark  
- wl-clipboard
```

# Boot Prozess

---

# Distrobox

---

# Was ist Distrobox?

- Container-basierte Umgebung für verschiedene Linux-Distributionen
- Integration in den Host ohne separate VMs
- Zugriff auf Host-Dateisystem und GUI-Anwendungen
- Alternative zu Toolbox mit erweiterten Features

# Distrobox Features

- **Host-Integration**
  - Zugriff auf `$HOME`-Verzeichnis
  - GPU-Zugriff für Grafikanwendungen
  - Netzwerk- und Audio-Integration
- **Multi-Distribution Support**
  - Ubuntu, Debian, Arch, Alpine, etc.
  - Verwendung beliebiger Container-Images
- **Grafische Anwendungen**
  - X11 und Wayland Support
  - Export von Anwendungen ins Host-System

# Distrobox Setup

```
# Container erstellen
distrobox create --name ubuntu --image ubuntu:22.04

# Container betreten
distrobox enter ubuntu

# Im Container arbeiten
sudo apt update && sudo apt install -y neovim

# Container mit eigener Konfiguration
distrobox create --name dev \
  --image fedora:39 \
  --additional-packages "vim tmux git"
```

# Distrobox Konfiguration

```
# ~/.config/distrobox/distrobox.ini
[dev-container]
image=fedora:39
additional_packages="neovim tmux git nodejs npm"
init_hooks="npm install -g typescript-language-server"
exported_apps="code"
pull=true
replace=true
```

- Automatische Container-Erstellung
- Paket-Installation beim Start
- Export von Anwendungen ins Host-Menü

# Distrobox Anwendungsfälle

## Entwicklung

- Language-specific Umgebungen (Node.js, Python, Go)
- Legacy-Software testen ohne Host-System zu belasten
- Cross-Distribution Development
- Isolierte Testumgebungen

## System-Administration & Tools

- Sicherheits-Tools (Kali Linux Container)
- Paketmanager verschiedener Distributionen
- Spezielle CLI-Programme
- Testing und Debugging

# Distrobox Anwendungsfälle

## GUI-Anwendungen

- Distribution-spezifische Software
- Ältere Anwendungsversionen
- Proprietäre Tools in sauberer Umgebung
- Apps die nicht als Flatpak verfügbar sind

# Chezmoi

---

# Was ist Chezmoi?

- **Dotfiles-Manager** für Konfigurationsdateien
- **Template-System** für maschinenspezifische Konfiguration
- **Verschlüsselung** sensibler Daten
- **Cross-Platform** (Linux, macOS, Windows)

```
# Installation  
rpm-ostree install chezmoi  
# oder als Flatpak/Container
```

# Chezmoi Grundlagen

# Initialisierung

```
chezmoi init
```

# Datei zur Verwaltung hinzufügen

```
chezmoi add ~/.bashrc
```

# Änderungen anzeigen

```
chezmoi diff
```

# Änderungen anwenden

```
chezmoi apply
```

# Mit Git-Repository

```
chezmoi init --apply https://github.com/user/dotfiles.git
```

# Chezmoi Templates

```
# ~/.local/share/chezmoi/dot_gitconfig.tpl
[user]
    name = {{ .name }}
    email = {{ .email }}
{{- if eq .chezmoi.hostname "workstation" }}
[core]
    sshCommand = "ssh -i ~/.ssh/work_key"
{{- end }}

# ~/.config/chezmoi/chezmoi.toml
[data]
    name = "Christoph Stoettner"
    email = "stoeps@stoeps.de"
```

# Chezmoi Verschlüsselung

```
# Verschlüsselte Datei hinzufügen
chezmoi add --encrypt ~/.ssh/id_rsa

# Age-Verschlüsselung konfigurieren
# ~/.config/chezmoi/chezmoi.toml
[encryption]
  command = "age"
  args = ["-d", "-i", "/home/user/.config/age/key.txt"]
```

- Unterstützt **age**, **gpg**, **1Password**
- Sichere Verwaltung von SSH-Keys
- API-Token und Passwörter verschlüsselt

# Chezmoi Scripts

```
# ~/.local/share/chezmoi/run_once_install_packages.sh
#!/bin/bash

{{- if eq .chezmoi.osRelease.id "fedora" }}
rpm-ostree install --apply-live \
    neovim tmux git
{{- else if eq .chezmoi.osRelease.id "ubuntu" }}
sudo apt update
sudo apt install -y neovim tmux git
{{- end }}
```

- **run\_once\_** Scripts laufen einmalig
- **run\_before\_** und **run\_after\_** für Hooks
- Systemspezifische Installation

# Chezmoi Best Practices

## Struktur & Organisation

- Logische Gruppierung von Konfigurationsdateien
- Template-Variablen für maschinenspezifische Werte nutzen
- `.chezmoidata` für lokale Anpassungen
- Klare Verzeichnisstruktur beibehalten

## Sicherheit

- Sensible Daten immer verschlüsseln (SSH-Keys, API-Token)
- `.chezmoiignore` für temporäre/lokale Dateien
- Separate Repositories für Arbeit und Privates
- Regelmäßige Backups der Schlüssel

# Chezmoi Best Practices

## Workflow-Optimierung

- `chezmoi edit` für direkte Bearbeitung verwenden
- `chezmoi status` vor jedem apply prüfen
- Git-Hooks für automatische Commits
- Systemd-Timer für regelmäßige Updates

# Praktische Beispiele

---

# Setup-Workflow: Neuer Rechner

```
# 1. Fedora Silverblue/Universal Blue installieren
# 2. Chezmoi Setup
chezmoi init --apply https://github.com/stoeps13/dotfiles.git

# 3. Distrobox Container erstellen
distrobox-assemble create --file ~/.config/distrobox/distrobox.ini

# 4. Flatpaks installieren
flatpak install flathub com.bitwarden.desktop \
  org.gnome.Extensions com.mattjakeman.ExtensionManager

# 5. System-Updates
rpm-ostree upgrade
flatpak update
```

# Development Environment

```
# Python Development Container
distrobox create --name python-dev \
  --image python:3.11 \
  --additional-packages "poetry black pytest"

# Node.js Container mit spezifischer Version
distrobox create --name node18 \
  --image node:18-alpine \
  --additional-packages "npm yarn"

# Container für verschiedene Projekte
distrobox enter python-dev
# vs
distrobox enter node18
```

# Security Tooling

```
# Kali Linux Container für Security Tools
distrobox create --name security \
  --image kalilinux/kali-rolling \
  --additional-packages "nmap wireshark burpsuite"

# Export GUI-Tools ins Host-Menu
distrobox enter security
sudo apt update && apt install -y burpsuite
distrobox-export --app burpsuite

# Jetzt verfügbar im Anwendungsmenü
```

# Legacy Software

```
# Alte CentOS 7 Umgebung für Legacy Apps
distrobox create --name legacy \
  --image centos:7 \
  --additional-packages "epel-release"

# Spezifische Softwareversionen
distrobox enter legacy
yum install -y python2 legacy-app

# Weiterhin moderne Tools auf Host
flatpak run org.mozilla.firefox
```

# Tips & Tricks

---

# Monitoring & Wartung

```
# System-Status überprüfen
```

```
rpm-ostree status
```

```
flatpak list --updates
```

```
distrobox list
```

```
# Updates anwenden
```

```
rpm-ostree upgrade
```

```
flatpak update
```

```
distrobox upgrade --all
```

```
# Rollback bei Problemen
```

```
rpm-ostree rollback
```

- **Systemd-Timer** für automatische Updates
- **Monitoring** mit `journalctl`
- **Backup** wichtiger Container

# Troubleshooting

## Häufige Probleme

- **Flatpak-Permissions:** Apps haben nicht die benötigten Berechtigungen
- **Container-Netzwerk:** Distrobox kann nicht auf Internet zugreifen
- **Hardware-Treiber:** NVIDIA oder andere proprietäre Treiber
- **Dateisystem-Rechte:** Zugriff auf bestimmte Verzeichnisse

## Debugging-Tools & Lösungen

- `rpm-ostree status` für System-Zustand
- `journalctl -b` für aktuelle Boot-Logs
- `distrobox list` für Container-Übersicht

# Troubleshooting

- `flatpak ps` für laufende Flatpak-Apps
- **Flatseal** für Flatpak-Berechtigungen
- Container neu erstellen bei persistenten Problemen

## Backup-Strategien

- Dotfiles regelmäßig in Git committen
- Wichtige Container exportieren: `distrobox-export`
- `/etc` Konfiguration sichern
- Flatpak-Liste exportieren: `flatpak list --app`

# Migration von traditionellen Distros

## Vorbereitung

- Inventar der installierten Software
- Identifikation von Flatpak-Alternativen
- Backup der wichtigsten Konfigurationen

## Migrationsstrategie

```
# 1. Software-Analyse
rpm -qa > installed_packages.txt # Fedora
dpkg --get-selections > packages.txt # Debian/Ubuntu

# 2. Flatpak-Recherche
flatpak search <package-name>

# 3. Distrobox für nicht verfügbare Software
# 4. Schrittweise Migration
```

# Performance & Ressourcen

## Speicherverbrauch

- **OSTree** Images: 2-3GB pro Version
- **Flatpaks**: Abhängigkeiten werden geteilt
- **Container**: Basis-Images werden geteilt
- **Gesamt**: Vergleichbar mit traditionellen Systemen

## Performance

- **Boot-Zeit**: Minimal langsamer
- **Anwendungen**: Native Performance
- **Updates**: Schneller (atomare Operationen)
- **Storage**: Intelligent dedupliziert

# Warum immutable OS?

## Für Entwickler

- Konsistente Entwicklungsumgebungen zwischen Projekten
- Einfache Reproduzierbarkeit auf verschiedenen Maschinen
- Sichere Experimente ohne Systemgefährdung
- Container-nativer, professioneller Workflow

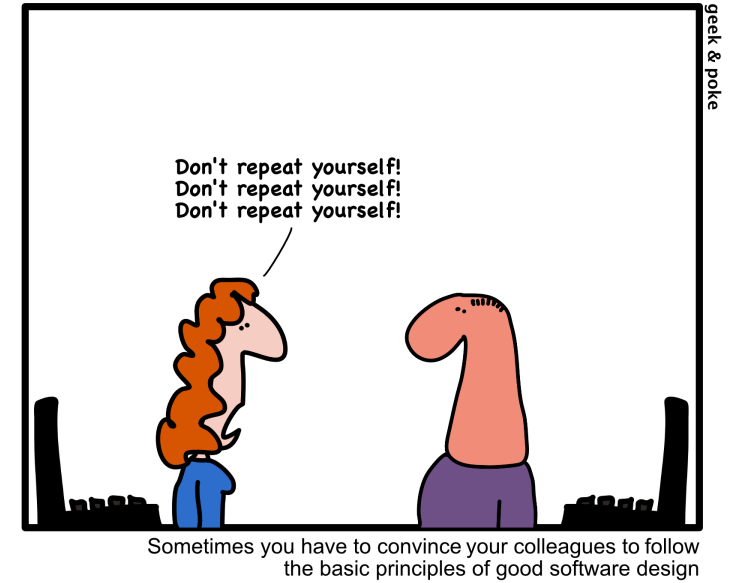


Figure 7: Geek & Poke<sup>1</sup>

# Warum immutable OS?

## Für IT-Administratoren

- Deutlich reduzierter Wartungsaufwand
- Vorhersagbare und testbare Update-Zyklen
- Einfache Rollback-Möglichkeiten bei Problemen
- Verbesserte Sicherheit durch Read-only System

# Warum immutable OS?

## Für Endnutzer

- Stabileres System mit weniger unerwarteten Problemen
- Weniger Systembrüche nach Updates
- Einfache Software-Installation via Flatpak
- Immer aktuelle, moderne Anwendungen

**Vielen Dank!**

---

## Links:

- Fedora Silverblue
- Project Bluefin
- Universal Blue
- Blue Build
- Distrobox
- Chezmoi

# Fragen



✉ [stoeps@stoeps.de](mailto:stoeps@stoeps.de)

🐙 [@stoeps@infosec.exchange](https://infosec.exchange/@stoeps)

🌐 [stoeps.de](https://stoeps.de)

🌐 [christophstoettner](https://www.linkedin.com/in/christophstoettner)

## Quellen und Links

1. Oliver Widder. Geek & Poke. <https://geek-and-poke.com/> (2024).
2. Christoph Stoettner. Dotfiles verwalten. [https://media.ccc.de/v/froscon2023-2907-dotfiles\\_verwalten](https://media.ccc.de/v/froscon2023-2907-dotfiles_verwalten) (2023).
3. Jorge Castro. Announcing Project Bluefin. <https://www.ypsidanger.com/announcing-project-bluefin/> (2023).
4. Jan Dolanský & Kyle Gospodnetich. <https://projectbluefin.io/> (2025).
5. Jorge Castro & 120 Contributor. <https://github.com/ublue-os/bluefin> (2025).
6. Jorge Castro. (Re)Announcing Project Bluefin. <https://www.youtube.com/watch?v=YFXufAVdrw4> (2023).
7. Fedora project - CC BY-SA 4.0. <https://fedoraproject.org/atomic-desktops/> (2025).
8. <https://universal-blue.org/#images> (2025).