Immutable Linux Desktops

Produktiv Arbeiten mit Fedora Silverblue, Chezmoi und Distrobox

Christoph Stoettner

2025-06-19

Christoph Stoettner (stoeps)

stoeps@stoeps.de
christophstoettner
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.

Disclaimer

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

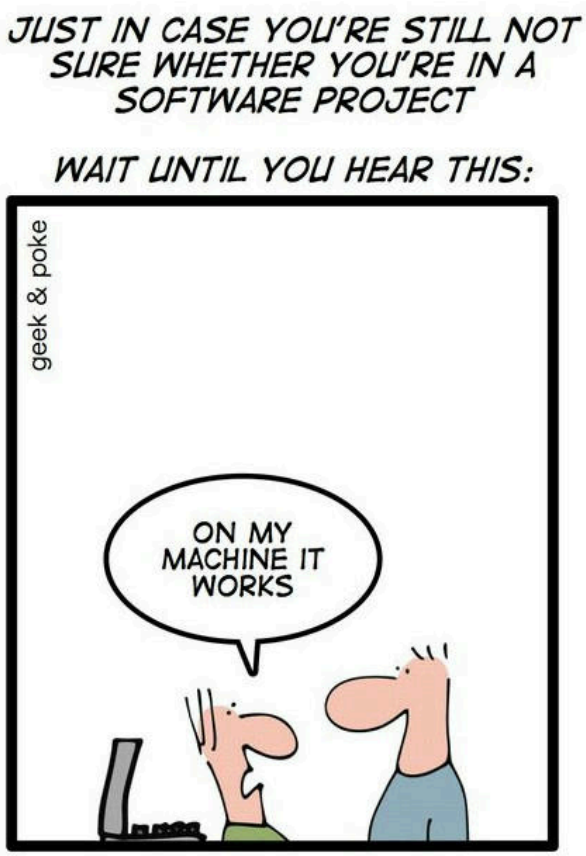


Figure 1: Geek & Poke [1]

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

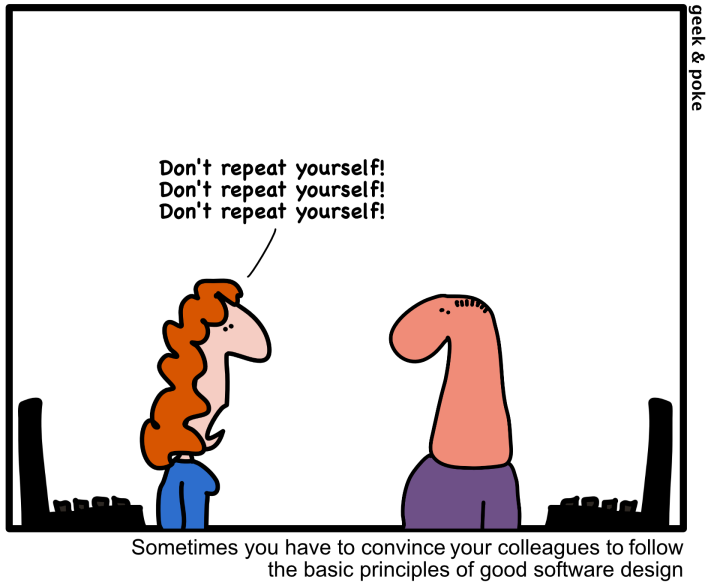


Figure 2: Geek & Poke [1]

- Versuch mit Ansible
  - siehe auch Froscon 2023 - Dotfiles verwalten [2]

Ankündigung Bluefin

- YT videos von Jorge Castro (CNCF) machten neugierig
- Blogpost mit Ankündigung [3]
- Project Bluefin [4]
- GitHub Repository von Bluefin [5]

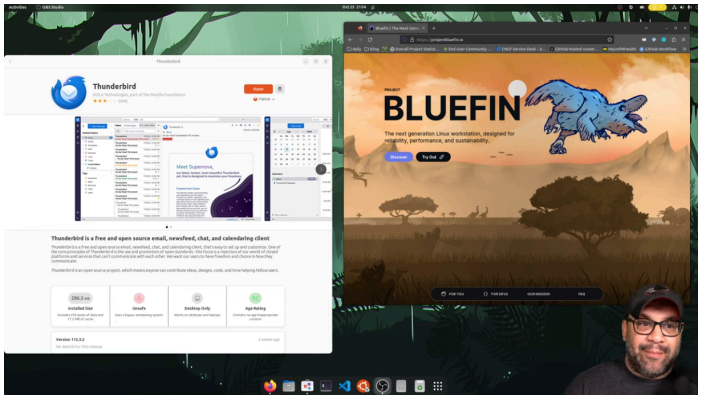


Figure 3: Announcement of Project Bluefin [6]

Fedora Atomic Desktops

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

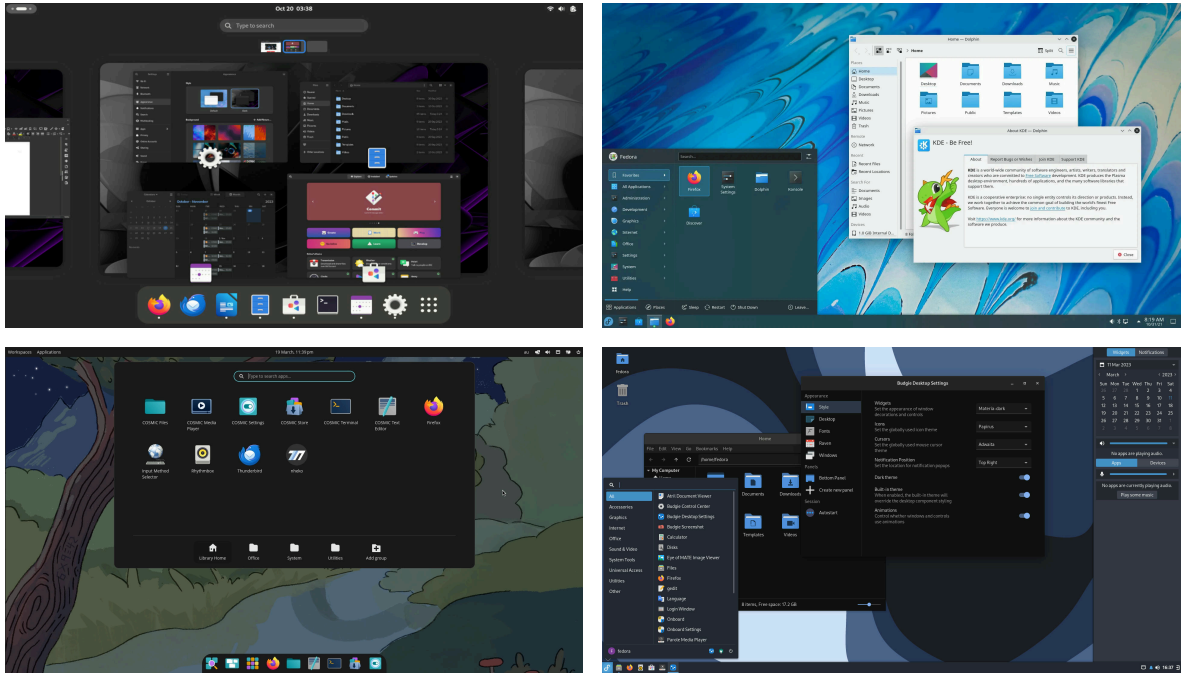


Figure 4: Screenshots: Silverblue, Kinoite, Cosmic & Budgie [7]

Universal Blue

- Universal Blue hilft bei automatisierter Erstellung
  - Desktop- und Server-Betriebssysteme
  - mit der “Zuverlässigkeit eines Chromebooks”
  - der Flexibilität traditioneller Linux-Desktops
- Build Skripte um eigene Images zu bauen
  - Cloud Infrastruktur Technik (Yaml und Containerfiles)
  - “Desktop DevOps Team”

Figure 5: Logo Universal Blue

Universal Blue Images

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

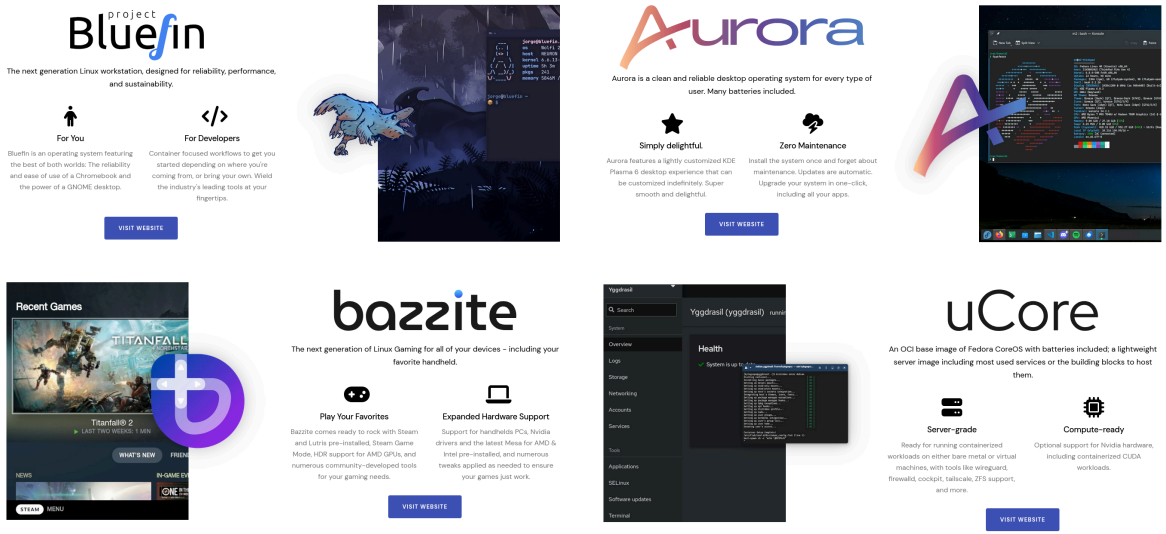


Figure 6: Screenshots: Bluefin, Aurora, Bazzite, uCore [8]

Entwickler Editionen –DX

- Container und DevContainer
- Integriert in VS Code
- Homebrew - installieren wie am Mac
- Pytxis

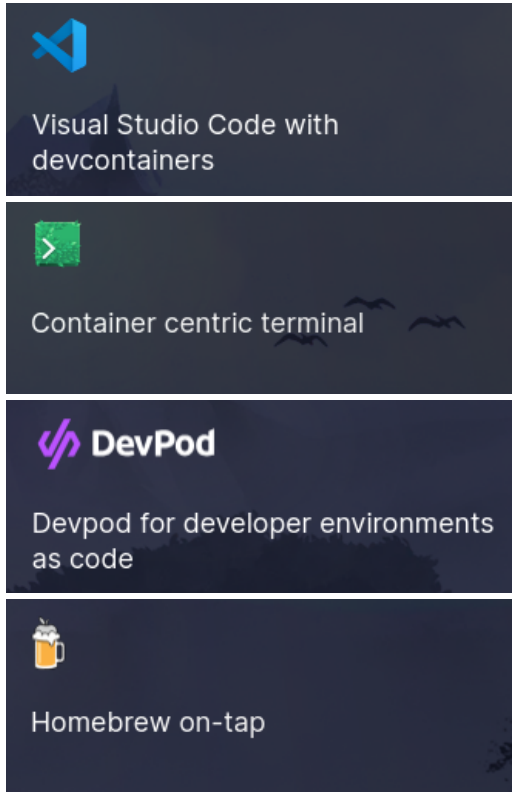


Figure 7: Screenshots VS Code, DevPod, Homebrew, Ptyxis [8]

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)

Technologien

OSTree

- Git-ähnliches Versionierungssystem für Betriebssysteme
- Entwickelt von Red Hat

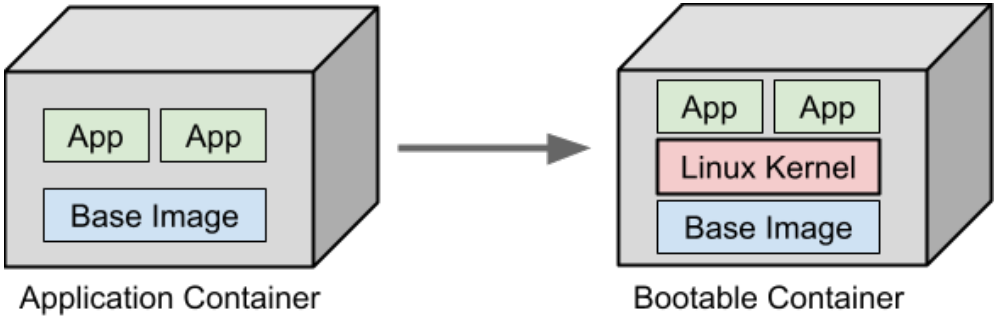
rpm-ostree

- Kombination aus RPM-Paketmanagement und OSTree
- Hybrides Image-/Paketsystem

bootc

- bootc
- Container-native Betriebssystemverwaltung
- OCI-Container als OS-Image-Basis (OCI Container mit Kernel)
- Vereinfacht CI/CD für Betriebssysteme
- Nachfolger der CoreOS-Architektur

Figure 8: Bootc Logo [9]



Containerisierung

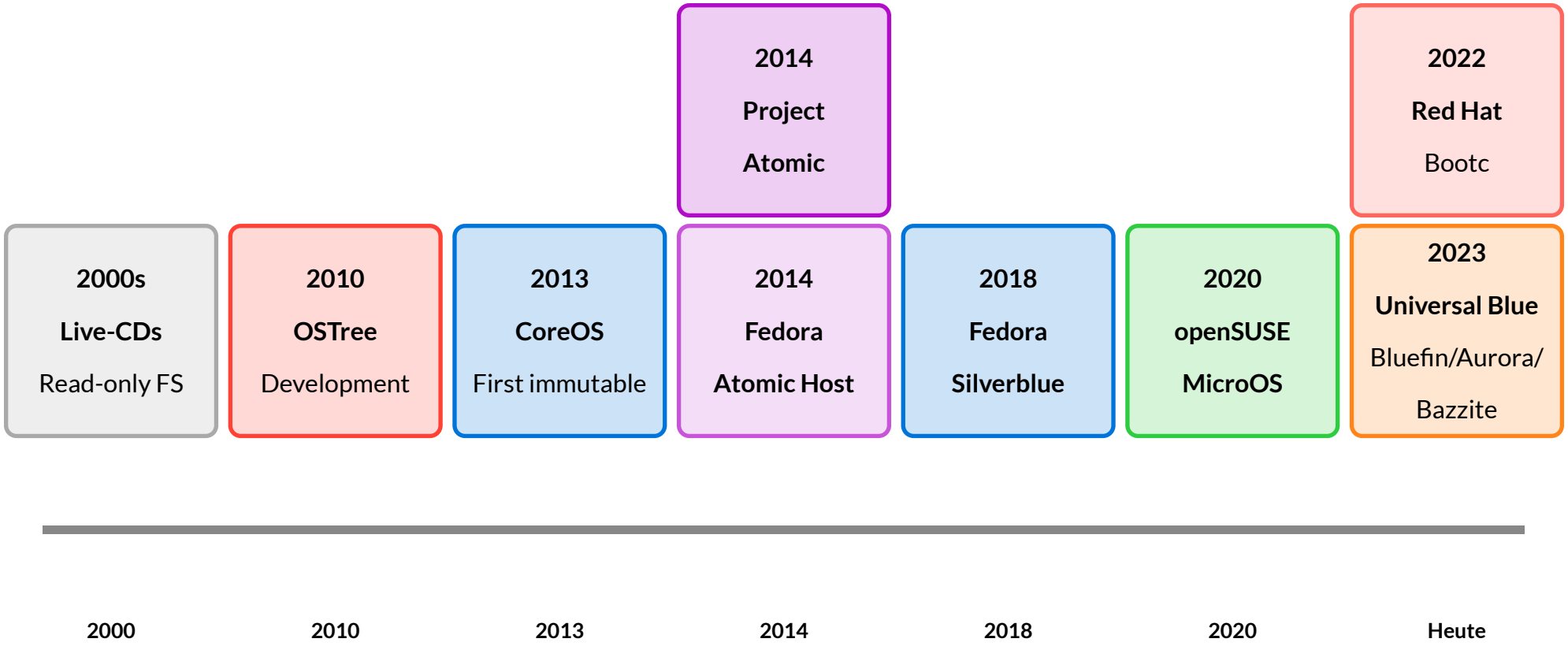
Flatpak

- Sandboxing und Abhängigkeitsisolation
- Desktop-Integration
- Flatseal GitHub
- Flatseal - Manage Flatpak Permissions

Container Runtime

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

Entwicklungsgeschichte



Warum Immutable OS wählen? Admins und Entwickler

Entwickler

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

Admins

- Reduzierter Wartungsaufwand
- Vorhersagbare Update-Zyklen
- Einfache Rollback-Möglichkeiten
- Bessere Compliance und Sicherheit

- 24 Monate Immutable Desktop – Ein Erfahrungsbericht [10]

Warum Immutable OS wählen? Benutzer

Endnutzer

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

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/roothome`
- `/usr` **readonly**
- `/usr/local` -> `/var/usrlocal`
- `/var`

Eigene Images bauen

Blue Build template

- Universal Blue Image Vorlage
- Blue Build Image Vorlage
  - Ublue-Stoeps Repository

```yaml
name: ublue-stoeps-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/default-flatpaks.yml
  - from-file: common_modules/gschema-overrides.yml
  - from-file: common_modules/rpm-ostree.yml
```

Pakete installieren

- Besser als `rpm-ostree` Layer
  - `common-modules/rpm-ostree.yml`

```yaml
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
```

Flatpaks installieren

- Vorinstallierte Anwendungen

`common_modules/default-flatpaks.yaml`

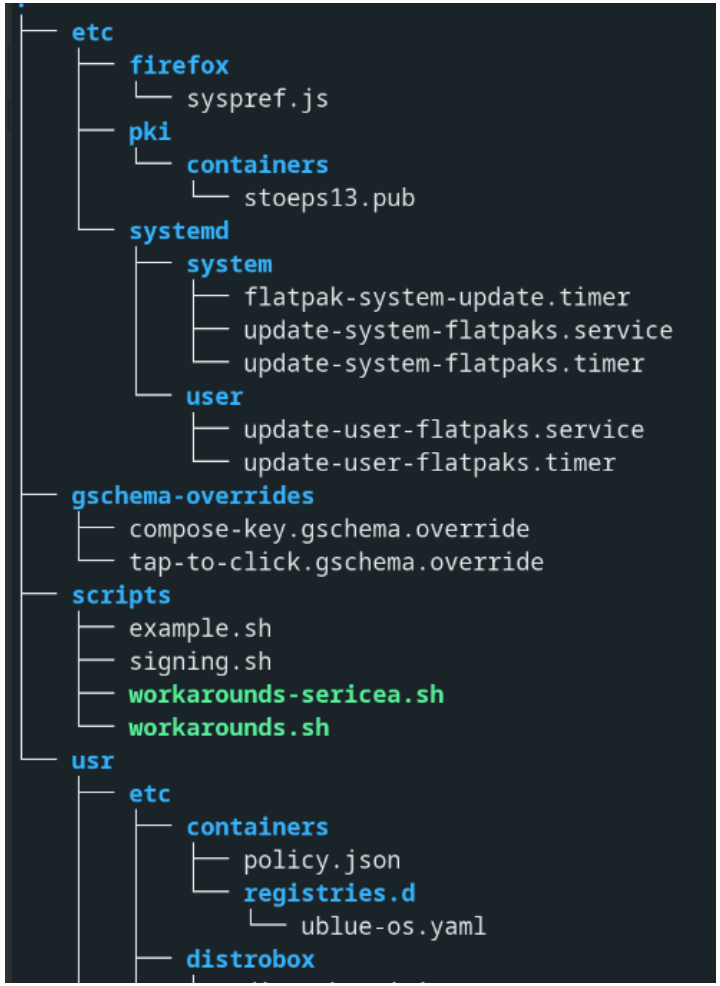
```yaml
type: default-flatpaks
system:
  repo-url: https://dl.flathub.org/repo/flathub.flatpakrepo
  repo-name: flathub
  install:
    - com.github.tchx84.Flatseal
    - io.dbeaver.DBeaverCommunity
    - io.github.java_decompiler.jd-gui
    - org.zaproxy.ZAP
```

Dateien ins Image kopieren

- Zusätzliche Dateien in `/usr` oder `/etc`

`common_modules/files.yml`

```yaml
type: files
files:
  - source: usr
    destination: /usr
  - source: etc
    destination: /etc
```



Skripte ausführen beim Image erstellen

- Skript ausführen beim Image bauen

```yaml
type: script
scripts:
  - workarounds.sh
```

- Änderungen in den später schreibgeschützten Ordnern

```
#!/bin/sh
set -oue pipefail
echo 'This is the workaround shell script'
/usr/libexec/gdm-runtime-config set daemon WaylandEnable false
```

Dateien aus anderen Containern kopieren

- Dateien aus anderen Container Images kopieren
- Praktisch bei Binaries die vorher kompiliert werden müssten

```yaml
type: containerfile
snippets:
  - COPY --from=cgr.dev/chainguard/dive:latest /usr/bin/dive /usr/bin/dive
  - COPY --from=cgr.dev/chainguard/helm:latest /usr/bin/helm /usr/bin/helm
  - COPY --from=cgr.dev/chainguard/kubectl:latest /usr/bin/kubectl /usr/bin/kubectl
```

Schriften installieren

- Schriftarten herunterladen und installieren

```yaml
type: fonts
fonts:
  nerd-fonts:
    - FiraCode
    - SourceCodePro
    - CascadiaCode
    - Hack
  google-fonts:
    - Fira Sans
    - Lato
    - Open Sans
```

Rebase zu anderem Image

- Von jeder Silverblue Installation kann man ohne Neuinstallation auf z.B. Bluefin wechseln

Offizielles Bluefin-dx

```
sudo bootc switch ghcr.io/ublue-os/bluefin-dx:stable \
    --enforce-container-sigpolicy
```

Rpm-ostree

```
sudo rpm-ostree rebase \
    ostree-image-signed:docker://ghcr.io/ublue-os/bluefin-dx:latest
```

Image

Pin

- Es sind immer mindestens zwei Images vorhanden
- zusätzliche Images können “gepinnt” werden
  - Fallback bei Bootproblemen

```
1  sudo ostree admin pin 1|0
```

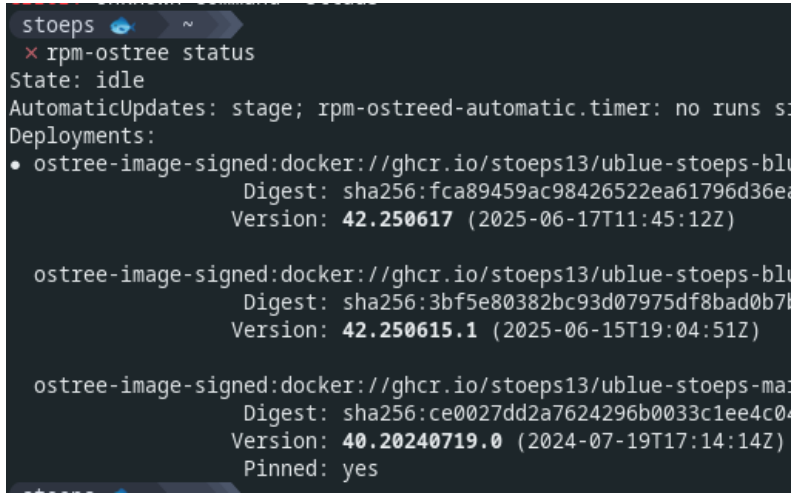


Figure 10: Lokale Images, default 2, zusätzlich ein Gepinntes

Rollback

```
1  rpm-ostree rollback 1
```

Distrobox

Was ist Distrobox?

Distrobox
Distrobox GitHub
Alternative zu Toolbx

- Fedora Atomic Desktop: Toolbx
- Universal Blue: Distrobox

Figure 11: Distrobox Logo [11]

- Container-basierte Umgebung für verschiedene Linux-Distributionen

Distrobox Features

Host-Integration

- Zugriff auf `$HOME`-Verzeichnis
- Netzwerk- und Audio-Integration

Multi-Distribution Support

- Verwendung beliebiger Container-Images

Grafische Anwendungen

- X11 und Wayland Support
- GPU-Zugriff für Grafikanwendungen
- Export von Anwendungen ins Host-System

Distrobox Container starten / erstellen

Distrobox Container erstellen

```
distrobox create -i docker.io/almalinux/8-init --init --name test
```

Automatisiert in `distrobox.ini`

```
[test]
image=docker.io/almalinux/8-init
init=true
pull=true
```

Es spricht absolut nichts gegen einen NixOS Container!

Shell im Container

```
1 distrobox enter test
```

- Container nutzen normal die gleich `$SHELL` wie unser Basis OS
- Shell wechseln mit `chsh` (beim ersten Ausführen der Box)
- Installieren von Apps mit den jeweiligen Paketmanagern
  - nicht persistent!
- `chsh` im Container wechselt die Shell (dauerhaft)

Prompt anpassen

- Container setzen `$CONTAINER_ID=Containername`

```
1 if [ -n $CONTAINER_ID ]; then
2   PS1="🐳 \e[0;34m$CONTAINER_ID\e[0m \w # "
3 fi
```

Figure 12: Simpler Prompt

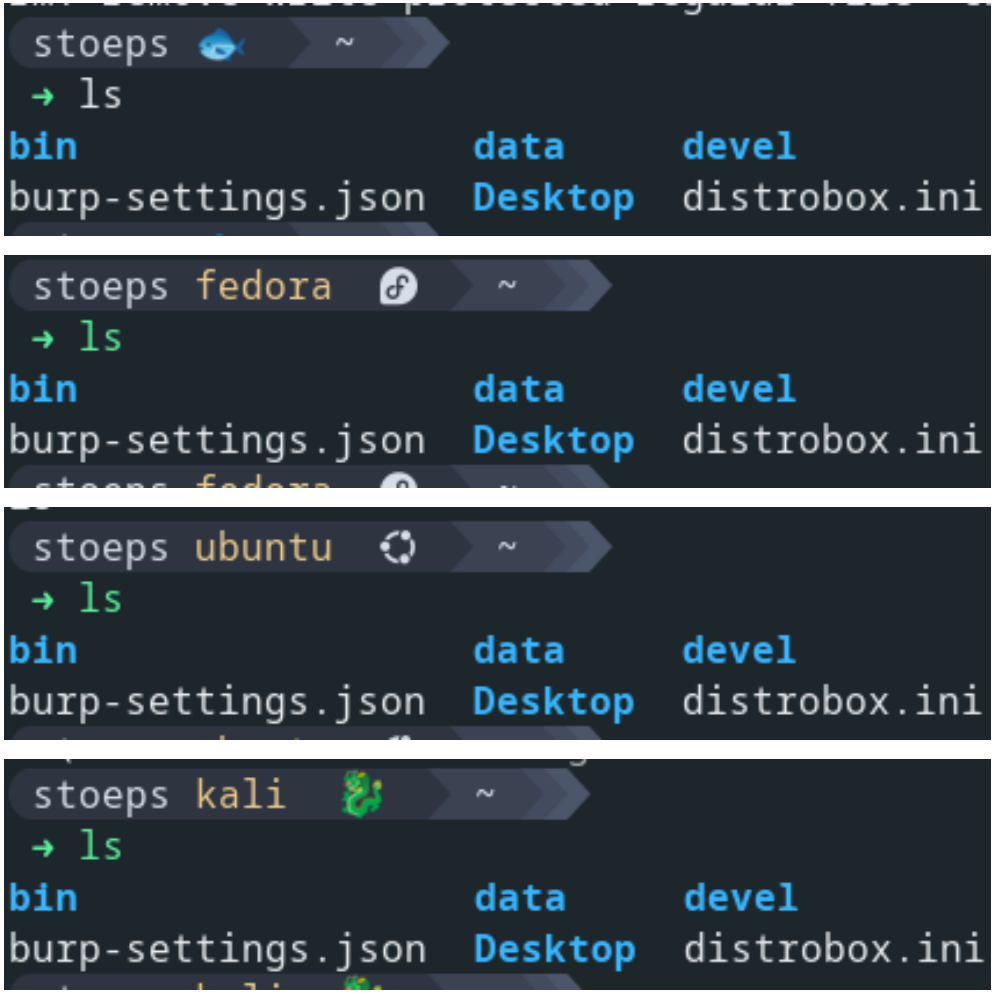
Prompt mit Starship anpassen

```
[hostname]
ssh_only = true
ssh_symbol = ' '
detect_env_vars = ['!CONTAINER_ID']

[container]
format = '\[$name\]'

[os]
format = '[ $symbol ]($style)'

[os.symbols]
Arch = ' '
Debian = ' '
Fedora = ' '
```



Home-Verzeichnis

- Default: Container teilen sich `$HOME`
  - gleiche Einstellungen für Bash, Zsh, Vim etc
  - also auch im Profil gespeicherte Binaries, Python oder Ruby Libs
  - `~/.local/bin` in `$PATH`

Container mit separatem `$HOME` erstellen

```
1 distrobox create --name test --image your-chosen-image:tag \
2       --home /your/custom/home
```

- Zum Testen von Anwendungen und Konfigurationen

Host Applikation im Container starten

- Zur Ausführung von Programmen im Basis OS aus einem Container
- z.B.
  - `podman`: starten von zusätzlichen Containern (nicht Docker in Docker)
  - `top` / `htop`: Auslastung des Host OS
  - `bootc` / `rpm-ostree`: Starten von Updates

Kommandozeile

```
1 /usr/bin/distrobox-host-exec /usr/bin/podman
```

Container Apps vom Host starten

```
1 [kali]
2 additional_packages="burpsuite zaproxy bloodhound neo4j"
3 image=ghcr.io/stoeps13/kali-toolbox:latest
4 exported_apps="burpsuite zaproxy"
```

- `additional_packages`: Installieren im Container
- `exported_apps`: `~/.local/share/applications/kali-burpsuite.desktop`

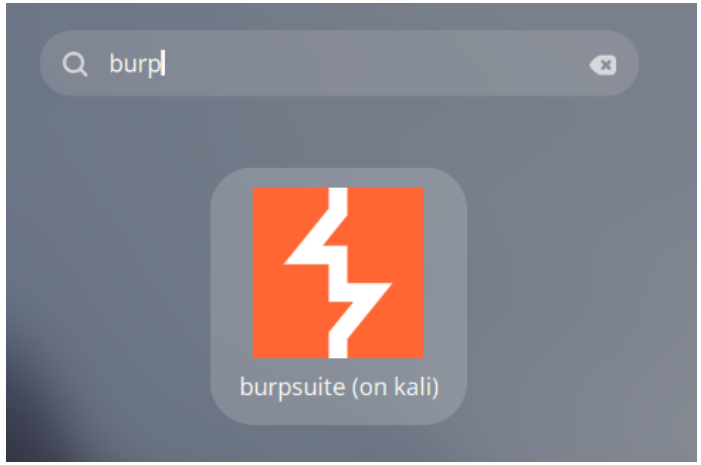


Figure 14: Exportierte App

Distrobox Anwendungsfälle

- Standard-Shell für die tägliche Arbeit
- Software testen
  - in unterschiedlichen Distributionen und Versionen
- Apps die nicht als Flatpak verfügbar sind
  - Beispiel: Davinci Resolve / OBS
- Container für LaTeX, Jupyter usw.

Chezmoi

Mein Hauptanwendungsfall

- Synchronisation meiner Einstellungen
  - Dotfiles und Gnome Settings
  - mehrere Rechner
  - WSL
- Installation von Tools die ich sowohl im Basis OS als auch im Container benutzen möchte
  - Viele Tools sind Single-File Binaries
  - Download als GitHub Release
- Linux speichert viele Benutzereinstellungen in sogenannten Dotfiles

Dotfiles?

- Konfigurationsdateien, die mit einem Punkt (`.`) beginnen
- Oder in `$XDG_CONFIG_HOME` (`~/.config`)
- Standardmäßig in Unix-ähnlichen Systemen versteckt
- Beispiele:
  - `.bashrc`
  - `.vimrc`
  - `.gitconfig`
  - `.tmux.conf`
  - `.zshrc`

Traditionelle Ansätze

- Manuelles Kopieren (fehleranfällig!)
- Einfaches Git-Repository
- Symlink-basierte Lösungen
- Benutzerdefinierte Skripte
- Spezialisierte Tools (GNU Stow, rcm, usw.)

Einschränkungen traditioneller Ansätze

- Gerätespezifische Konfiguration
- Token und Secrets
- Dateiberechtigungen
- Manuelle Schritte oder Skripte
- Keine Vorlagen
- Komplexität steigt mit der Zeit

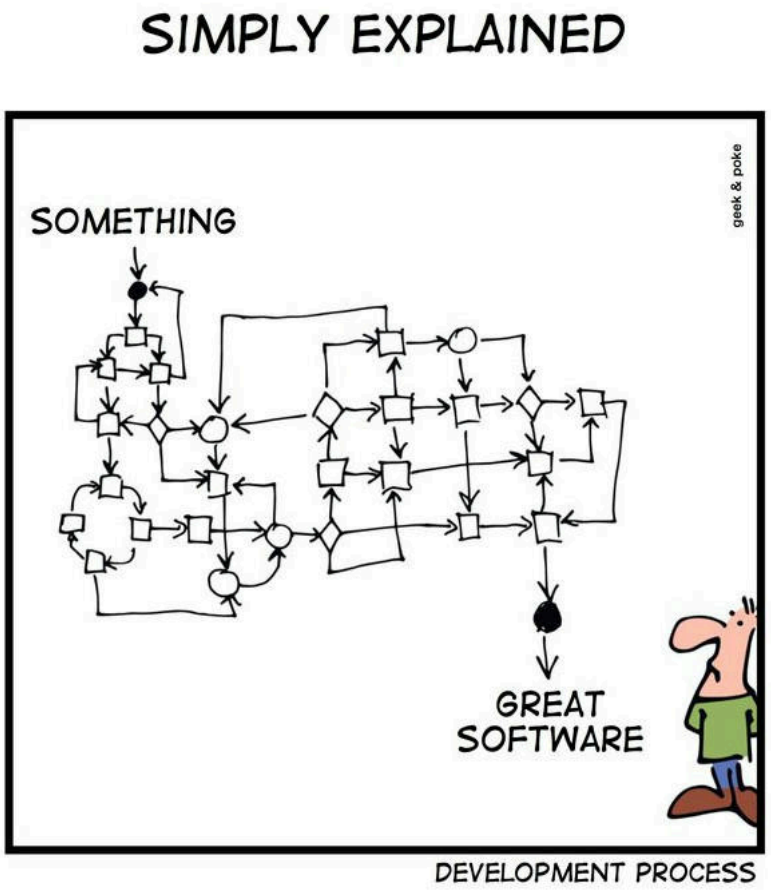


Figure 15: Geek & Poke [1]

Wie hilft Chezmoi

- GitHub Repo twpayne/chezmoi
  - Tom Payne
  - 15.200 Sterne
  - 200+ Contributor

“Verwalten Sie Ihre Dotfiles sicher über mehrere verschiedene Geräte hinweg.”

Installation

- Pakete für viele Linux Distributionen
  - Modul in https://blue-build.org
  - Nix, Snap
- Windows, Mac OS, FreeBSD
  - Homebrew, Chocolatey
- Einzeiler (natürlich erstmal das Skript prüfen)

Neue `chezmoi` Konfiguration

```
1  chezmoi init
2  chezmoi cd
3  git remote add origin https://github.com/$GITHUB_USERNAME/dotfiles.git
4  git push -u origin main
```

Dateien zu Chezmoi hinzufügen

```
1  chezmoi add ~/.bashrc
2  chezmoi add ~/.config/nvim
```

Datei als Template hinzufügen

```
1  chezmoi add --template ~/.bashrc
```

Konfigurationsdateien editieren

1. `chezmoi edit ~/.bashrc`
2. `chezmoi cd; vim dot_bashrc`
3. `chezmoi edit`
4. Direkt editieren und mit `chezmoi add` bzw. `chezmoi re-add` hinzufügen
5. Direkt editieren und mit `chezmoi merge ~/.bashrc`

3-Wege Merge mit `vimdiff`

```
1  chezmoi merge .bashrc
```

- Default `vimdiff`
  - Strg + w + h / j / k / l

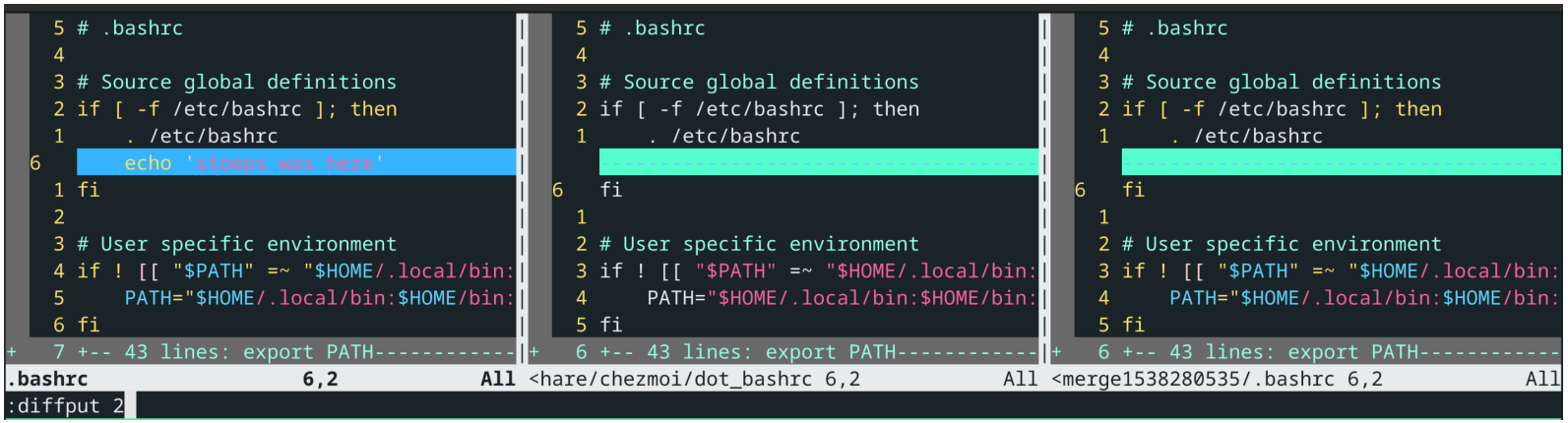


Figure 16: 2025-03-17_21-53-17.png

Dateien finden, die nicht von `chezmoi` gemanaged werden

```
1  chezmoi unmanaged
2  chezmoi unmanaged ~/.config
```

- Erzeugt eine lange Liste Dateien

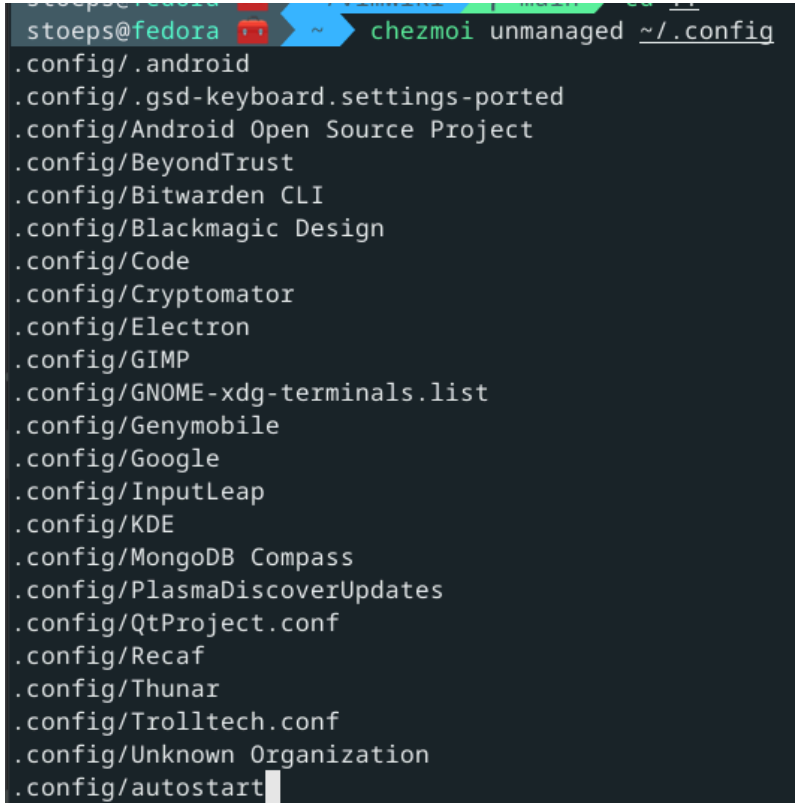


Figure 17: 20250320-174502.png

Installation auf weiteren Rechnern

- Wenn das Git Repository `dotfiles` heißt

```
sh -c "$(curl -fsLS get.chezmoi.io)" --init --apply $GITHUB_USERNAME
```

- GitHub Repos können mit User/Repo abgekürzt werden

```
sh -c "$(curl -fsLS get.chezmoi.io/lb)" --init \
    --apply stoeps13/chezmoi-dotfiles
```

```
sh -c "$(curl -fsLS get.chezmoi.io/lb)" --init --apply \
    --ssh stoeps13/clt-dotfiles
```

```
sh -c "$(curl -fsLS get.chezmoi.io/lb)" --init --apply \
    https://your-git-server/name/repo.git
```

Namenskonventionen Dateien und Ordner

- Dateiname beeinflusst Dateieigenschaften

`~/.local/share/chezmoi`

```
1 drwxr-xr-x. 1 stoeps stoeps    10 Sep 27 15:35 dot_asciidoctor
2 -rw-r--r--. 1 stoeps stoeps  1449 Mar 17 22:00 dot_bashrc
3 drwxr-xr-x. 1 stoeps stoeps   220 Nov 28 12:08 private_dot_config
```

- Erstellt Verzeichnis `.asciidoctor` in `$HOME` mit 755 Mode
- Datei `.bashrc` in `$HOME` mit 644
- Ordner `.config` mit 700

Skripte ausführen

```
1  -rwxr-xr-x. 1    387 Mar  2 22:27 run_after_gsettings.sh
2  -rwxr-xr-x. 1   1188 Mar 19 18:35 run_before_bw_install.sh
3  -rwxr-xr-x. 1   1188 Mar 19 18:35 run_onchange_before_nvim_install.sh
4  -rwxr-xr-x. 1   1345 Mar  2 22:26 run_onchange_install_flatpak.sh
```

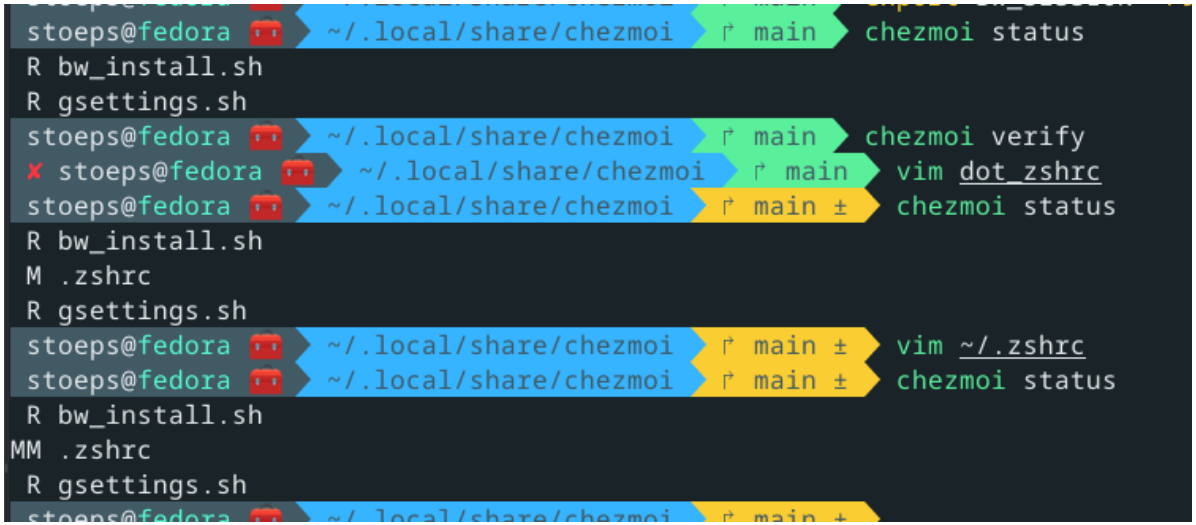
- `run_` scripts: werden immer bei `chezmoi apply` ausgeführt
- `run_onchange_` scripts: Werden nur ausgeführt, wenn sich der Inhalt seit der letzten Ausführung geändert hat
- Ausführung erfolgt alphabetisch
- `\_before_`, `\_after_`
  - Skripte werden vor bzw. nach dem Update der Konfigurationsdateien ausgeführt

Spezielle Dateien und Ordner

- `.chezmoi.$FORMAT.tmpl` (JSON|YAML|INI)
  - Erstellt die Konfigurationsdatei für `chezmoi`
- `.chezmoidata.$FORMAT`
  - Zur zentralen Definition von Variablen
- `.chezmoiexternal`
  - Download git Repositories oder Archive
- `.chezmoiignore[.tmpl]`
  - z.B. README, oder Dateien und Verzeichnisse die maschinenabhängig gesetzt werden
- `.chezmoiremove`
  - Um Dateien aus `$HOME` zu entfernen

Chezmoi status

Character	Meaning	1. Spalte	2. Spalte
A	Added	was created	will be created
D	Deleted	was deleted	will be deleted
M	Modified	was modified	will be modified
R	Run	Not applicable	will be run



Überblick über grundlegende Befehle (Fortsetzung)

- Änderungen anwenden

```
1  chezmoi apply
```

- Änderungen auf eine bestimmte Datei anwenden

```
1  chezmoi apply ~/.vimrc
```

- Entfernen Sie eine Datei aus der Verwaltung

```
1  chezmoi forget ~/.vimrc
```

Vorlagen

~/.config/chezmoi/config.toml

```
[data]
    name = "Max Mustermann"
    email = "max@beispiel.de"
    work = true
```

dot_gitconfig.tmpl

```
1 [user]
2     name = {{ .name }}
3     email = {{ .email }}
```

```
1 {{- if .work }} #
2   export HTTP_PROXY="http://proxy.beispiel.de:8080"
3 {{- end }}
```

Gerätespezifische Konfigurationen - OS / Hostname

dot_bashrc.tmpl

```
1  {{- if eq .chezmoi.os "darwin" }}
2  export PATH="$(brew --prefix)/bin:$PATH"
3  {{- end }}
4  {{- if eq .chezmoi.hostname "arbeits-laptop" }}
5  export HTTP_PROXY="http://proxy.beispiel.de:8080"
6  {{- end }}
```

Verwaltung von Token, Passwörtern

- Integration mit Passwortmanagern:
  - Bitwarden
  - 1Password
  - LastPass
  - pass
  - Vault
  - gopass
  - KeePassXC
  - Weitere…

`~/.config/chezmoi/config.toml`

```
1 encryption = "age"
2 [age]
3     identity = "~/.config/age/key.txt"
4     recipient = "age1 ... "
```

Felder oder Dateien verschlüsseln

```
# Vorlage mit Passwortmanager-Integration
{{ (bitwarden "item" "example.com").password }}

# Oder mit age-verschlüsselten Dateien
chezmoi add --encrypt ~/.ssh/id_rsa
```

dot_gitconfig.tmpl

```
[user]
{{- if .work }} #
    signingkey = {{ (bitwarden "item" "work-gpg-key").notes }}
{{- else }}
    signingkey = {{ (bitwarden "item" "personal-gpg-key").notes }}
{{- end }}
```

Externe Quellen - Git Repositories

- Archive und Git Repositories
- `~/.local/share/chezmoi/chezmoiexternal.toml`
- Komplette Git Repositories clonen

```
1  [".oh-my-zsh"]
2  type = "git-repo"
3  url = "https://github.com/ohmyzsh/ohmyzsh.git"
4  refreshPeriod = "168h"
5
6  [".oh-my-zsh-custom/plugins/zsh-syntax-highlighting"]
7  type = "git-repo"
8  url = "https://github.com/zsh-users/zsh-syntax-highlighting.git"
9  refreshPeriod = "168h"
```

Externe Quellen - Einzelne Datei

- z.B. zum Herunterladen von Git Releases
- `gitHubLatestReleaseAssetURL` um die aktuellste Version herunterzuladen

```
[".local/bin/talosctl"]
type = "file"
url = {{
        gitHubLatestReleaseAssetURL
        "siderolabs/talos"
        "talosctl-linux-amd64" | quote
}}
refreshPeriod = "168h"
executable = true
```

Externe Quellen - Einzelne Datei aus Archiv

- Nur die Binary Datei
- Herausforderung:
  - Betriebssystem und Architektur im Namen
  - Wildcard für Versionsnummer

```
[".local/bin/hugo"]
type = "archive-file"
url = {{ gitHubLatestReleaseAssetURL
         "gohugoio/hugo"
         (printf "hugo_extended_*_%s-%s.tar.gz" .chezmoi.os .chezmoi.arch)
         | quote }}
executable = true
refreshPeriod = "168h"
path = "hugo"
```

Externe Quellen - Spezial mit Variable

- Letzte Versionsnummer von GitHub, aber andere Download URL

```
{{- $mitmVersion := ((gitHubLatestRelease "mitmproxy/mitmproxy").TagName) |
replaceAllRegex "v" "" }}

[".local/bin"]
type = "archive"
refreshPeriod = "168h"
url = "https://downloads.mitmproxy.org/{{ $mitmVersion }}/mitmproxy-
{{ $mitmVersion }}-linux-x86_64.tar.gz"
```

Externe Quellen - Optionen

- `include` / `exclude` (nur eine Teilmenge der Dateien entpacken)
- `stripComponents` (Archive mit Unterordnern)

```
[".local/share/fonts"]
type = "archive"
url = {{ gitHubLatestReleaseAssetURL "FortAwesome/Font-Awesome" (printf
"fontawesome-free-*-desktop.zip") | quote }}
exact = true
stripComponents = 2
refreshPeriod = "168h"
include = ["**/otfs/Font*.otf"]
```

Links:

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

Fragen

stoeps@stoeps.de
@stoeps@infosec.exchange
stoeps.de
christophstoettner

Quellen und Links

[1] Oliver Widder, “Geek & Poke.” [Online]. Available: https://geek-and-poke.com/

[2] Christoph Stoettner, *Dotfiles verwalten*, (2023). [Online Video]. Available: https://media.ccc.de/v/froscon2023-2907-dotfiles_verwalten

[3] Jorge Castro, “Announcing Project Bluefin.” [Online]. Available: https://www.ypsidanger.com/announcing-project-bluefin/

[4] Jan Dolanský and Kyle Gospodnetich, [Online]. Available: https://projectbluefin.io/

[5] Jorge Castro and 120 Contributor, [Online]. Available: https://github.com/ublue-os/bluefin

[6] Jorge Castro, *(Re)Announcing Project Bluefin*, (Nov. 28, 2023). [Online Video]. Available: https://www.youtube.com/watch?v=YFXufAVdrw4

[7] Fedoraproject - CC BY-SA 4.0, [Online]. Available: https://fedoraproject.org/atomic-desktops/

[8] Universal blue, [Online]. Available: https://universal-blue.org/#images

[9] bootc-dev, [Online]. Available: https://github.com/bootc-dev/bootc

[10] Jan Wenzel, (2025). [Online Video]. Available: https://media.ccc.de/v/clt25-206-24-monate-immutable-desktop-ein-erfahrungsbericht

[11] David Lapshin (https://github.com/daudix), 2025. [Online]. Available: https://github.com/daudix/distrobox/blob/main/docs/assets/page-logo.svg

Chezmoi - Community und Ressourcen

- GitHub: https://github.com/twpayne/chezmoi
- Dokumentation: https://www.chezmoi.io/
- Benutzerhandbuch: https://www.chezmoi.io/user-guide/command-overview/
- Beispiel-Dotfiles-Repositories
  - https://github.com/zhaohongxuan/dotfiles
  - https://github.com/twpayne/dotfiles
  - https://github.com/xvzc/chezmoi.nvim
  - https://github.com/logandonley/dotfiles