

# Next-Gen Desktops

**Universal Blue immutable desktop**

Christoph Stoettner

[stoeps@vegardit.com](mailto:stoeps@vegardit.com)

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

# Christoph Stoettner (stoeps)



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

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

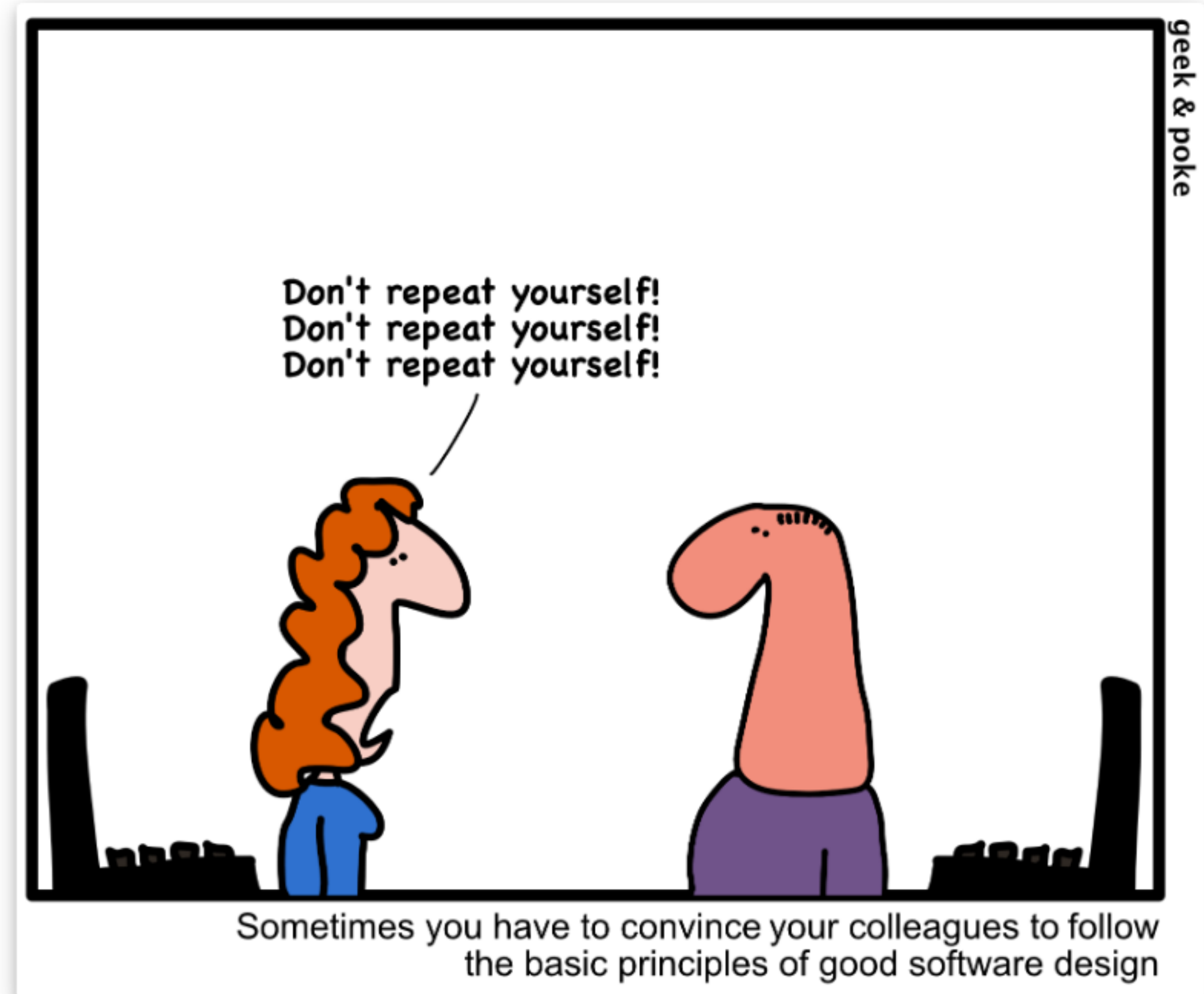
📡 [stoeps.de](https://stoeps.de)

✉ [@stoeps@infosec.exchange](mailto:@stoeps@infosec.exchange)

- Macht seit 30 Jahren was mit Computern
  - Amiga, OS/2, Linux
  - Beruflich auch Windows (wenn es sein muss)
- Started with Linux / OSS around 1994/1995
  - Linux Kernel < 1.0
  - Slackware
- mag vi, vim, neovim
  - zu doof für emacs

# Ausgangslage

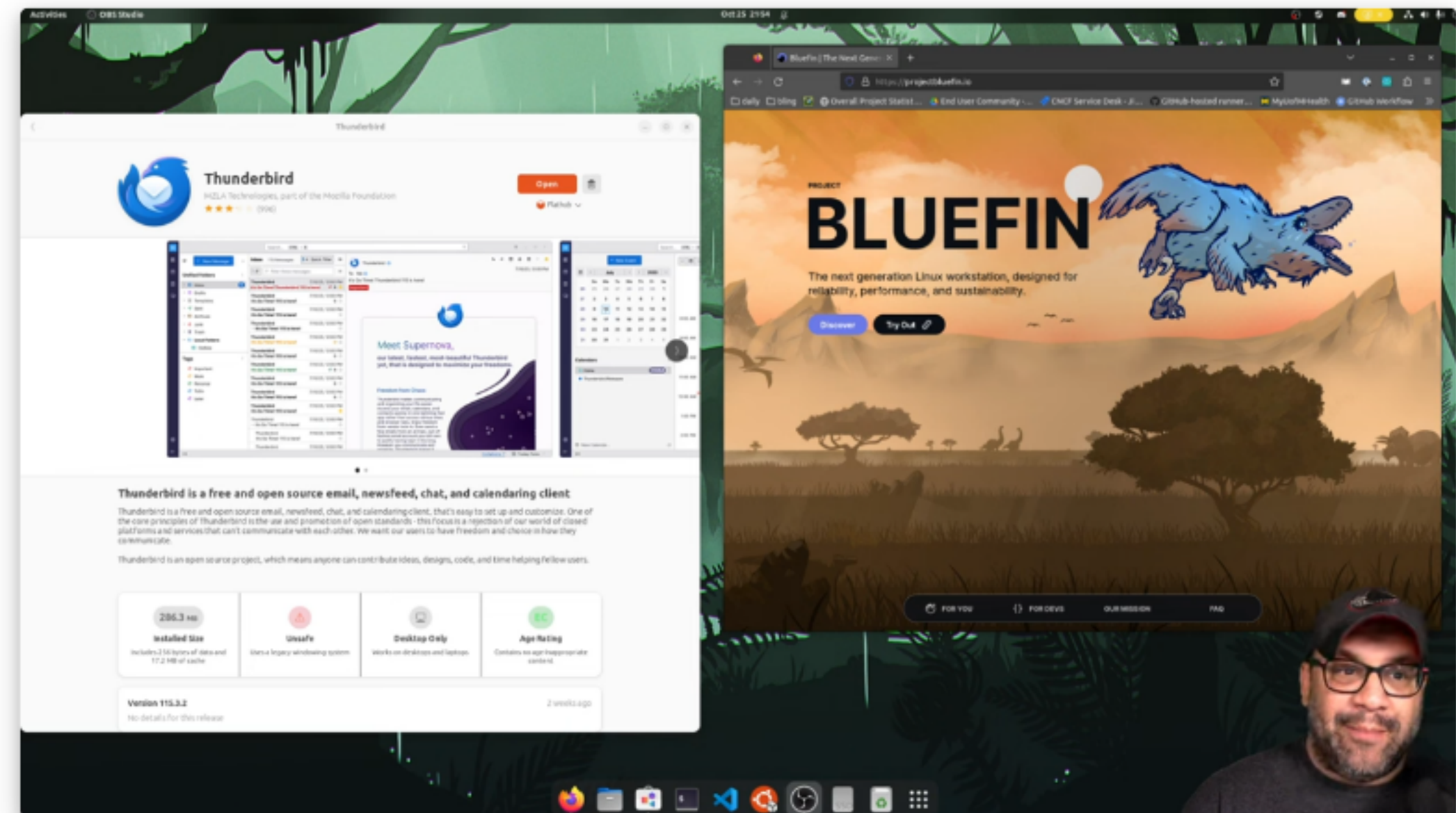
- Mehrere Rechner
  - Linux
  - Windows mit WSL
- Dotfiles in git
- Oft fehlt auf einem Rechner das eine Tool
- 2023 Versuch mit Ansible
  - Sehr aufwändig
  - siehe auch: [Froscon 2023](#)



CC BY 3.0 – Geek & Poke

# Ankündigung von Bluefin

- YT videos von Jorge Castro (CNCf) machten neugierig
  - Bluefin Announcement
  - Bluefin





# Disclaimer

- Der Vortrag soll nicht missionieren
- Werdet bitte glücklich mit eurer Distribution
- NixOS ist super, erschliesst sich mir aber v.a. sprachlich nicht
- Universal Blue ist sehr neu und offiziell nicht stable



Photo by Lukas on Unsplash

# Was zeichnet ein immutable OS aus?

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

# Vorteile

- **Sicherheit / Security**
  - Schadsoftware kann nur begrenzt Änderungen am System vorzunehmen
- **Stabilität / Stability**
  - Systemdateien können nicht versehentlich verändert oder gelöscht werden
  - Atomare Updates sorgen dafür, dass Systemaktualisierungen das System nicht in einem teilweise aktualisierten und potenziell instabilen Zustand hinterlassen.
- **Reproduzierbarkeit**
  - Es einfacher, das System zu testen, zu prüfen und zu verifizieren
  - Troubleshooting am eigenen System möglich und Ergebnis übertragbar
  - Läuft auf meiner Maschine :D
- **Cloud native**
  - Anpassung mittel Containerfiles und Github Actions



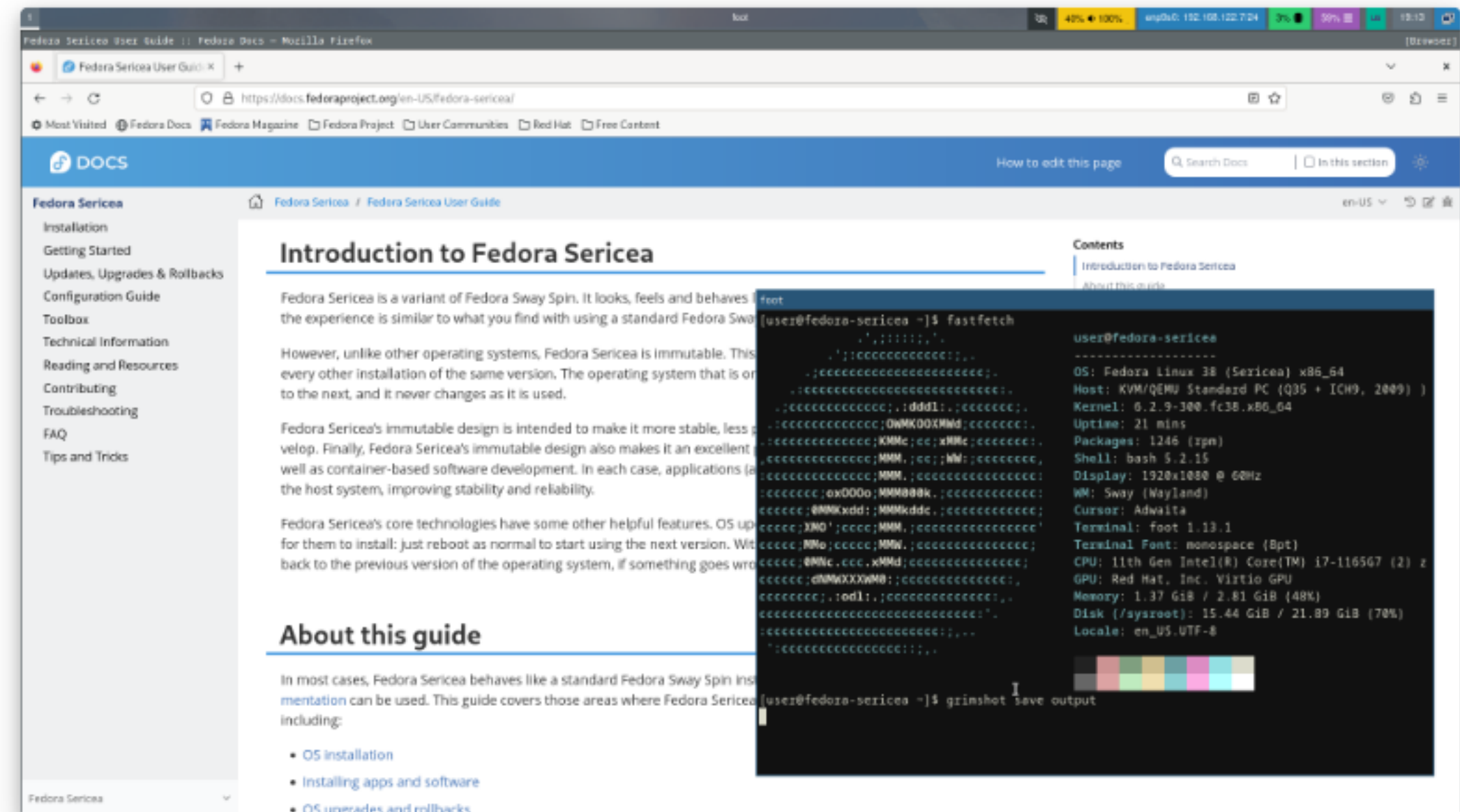
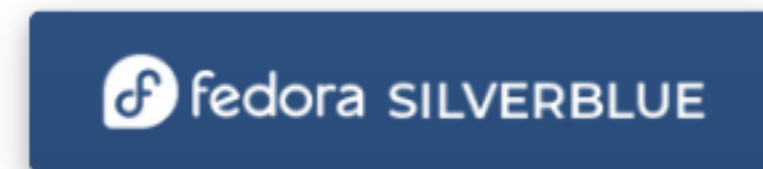
# Nachteile

- **Reduzierte Flexibilität**
  - Benutzer können Systemdateien nicht in gleichem Maße ändern oder ihr System anpassen
- **Eingeschränkte Kompatibilität**
  - Nicht alle Anwendungen und Dienste sind mit den containerisierten oder isolierten Umgebungen eines unveränderlichen Betriebssystems kompatibel
- **Speicheranforderungen**
  - Update-Mechanismen erfordern oft die Speicherung von Image-Snapshots
  - Isolierte Anwendungen können zu Redundanzen in der Speicherung von Anwendungsabhängigkeiten führen.
- **Entwicklernerfahrung**
  - Dockerfile / Containerfile (eh nur Yaml)



# Fedora Atomic Desktop

- rpm-ostree
- Applikationen als Flatpak
- Toolbox (OCI Container für CLI und GUI Applikationen)
- Varianten:
  - Fedora Silverblue (Gnome)
  - Fedora Kinoite (KDE)
  - Fedora Sway Atomic (Sway WM)
  - Fedora Budgie Atomic (Budgie)



# Universal Blue

- <https://universal-blue.org/>
- Build Umgebung für eigene Images auf Basis von Fedora Atomic Desktop
- Aurora (KDE)
- Bazzite (Linux Gaming)
- Project Bluefin (Gnome)
- Images für spezielle Hardware
  - Nvidia Support
  - Steamdeck
  - Framework Laptop
  - und vieles mehr (Surface, Asus)
- Mittels rebase kann man von jeder Silverblue Variante diese Images testen





# Bazzite

## bazzite

The next generation of Linux Gaming for all of your devices -  
including your favorite handheld.



### Play Your Favorites

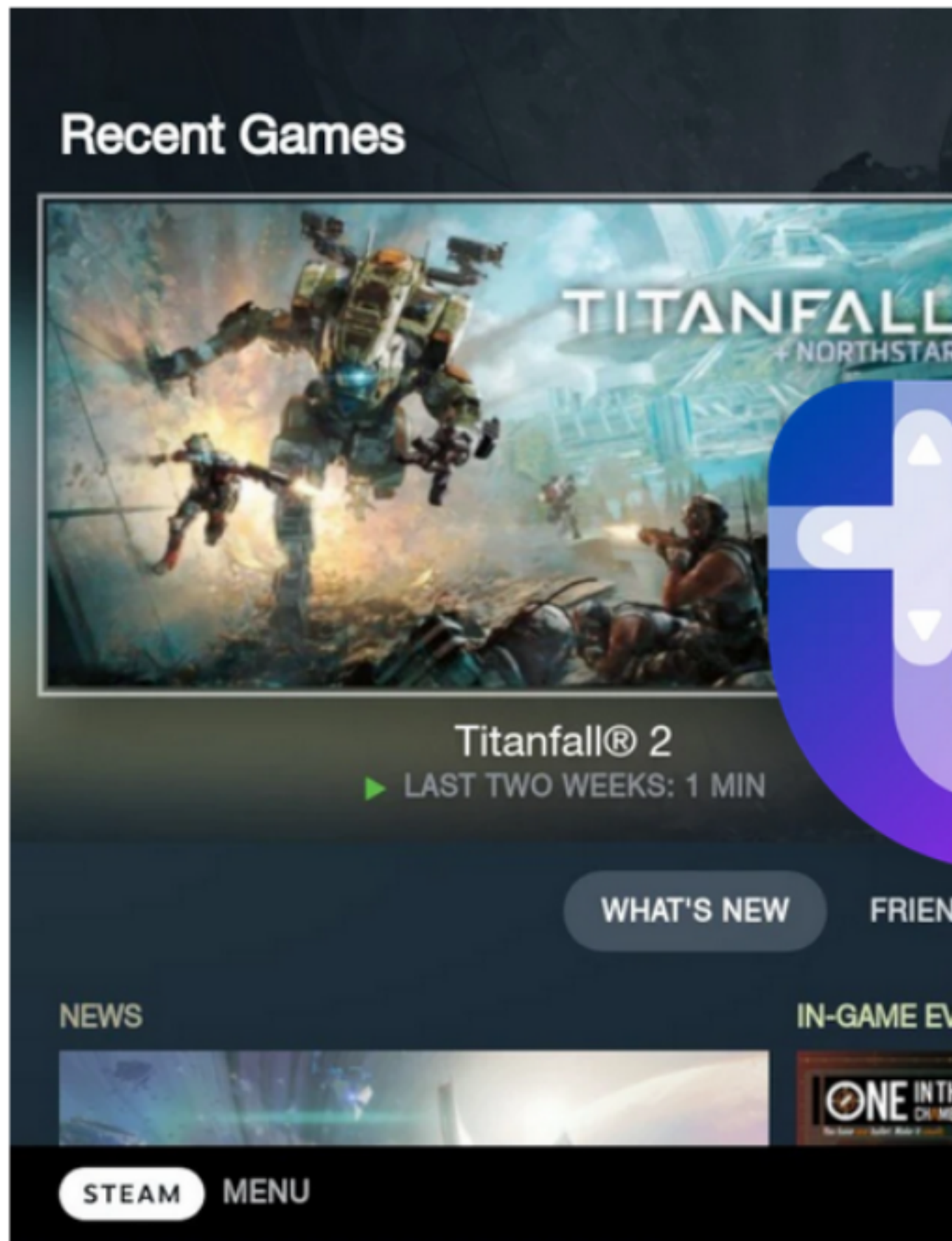
Bazzite comes ready to rock with Steam and Lutris pre-installed, Steam Game Mode, HDR support for AMD GPUs, and numerous community-developed tools for your gaming needs.



### Expanded Hardware Support

Support for handhelds PCs, Nvidia drivers and the latest Mesa for AMD & Intel pre-installed, and numerous tweaks applied as needed to ensure your games just work.

VISIT WEBSITE



# Project Bluefin

project  
**Bluefin**

The next generation Linux workstation, designed for reliability, performance, and sustainability.



For You

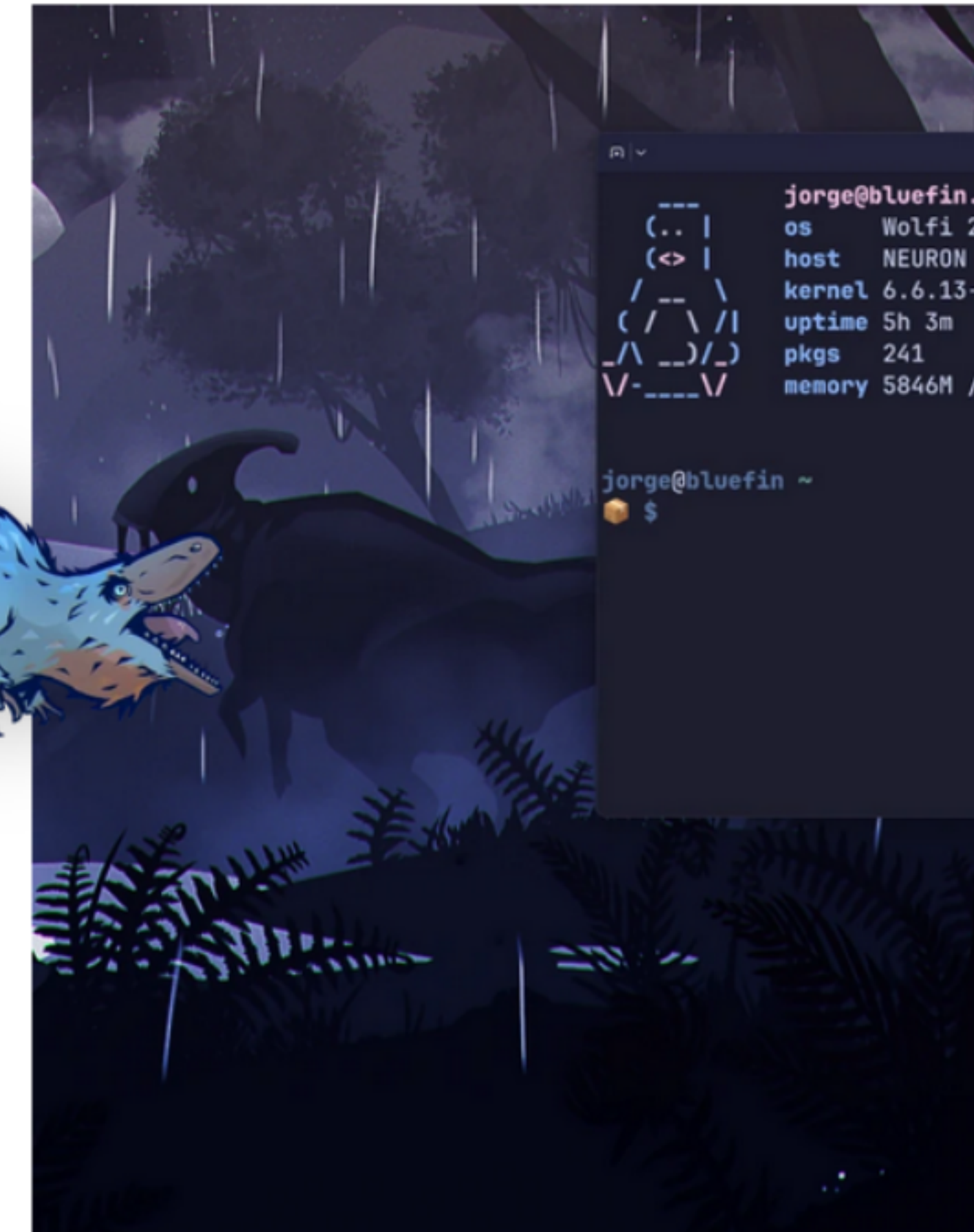
Bluefin is a custom image of Fedora Silverblue offering the best of both worlds: The reliability and ease of use of a Chromebook and the power of a GNOME desktop.



For Developers

Container focused workflows to get you started depending on where you're coming from, or bring your own. Wield the industry's leading tools at your fingertips.

[VISIT WEBSITE](#)



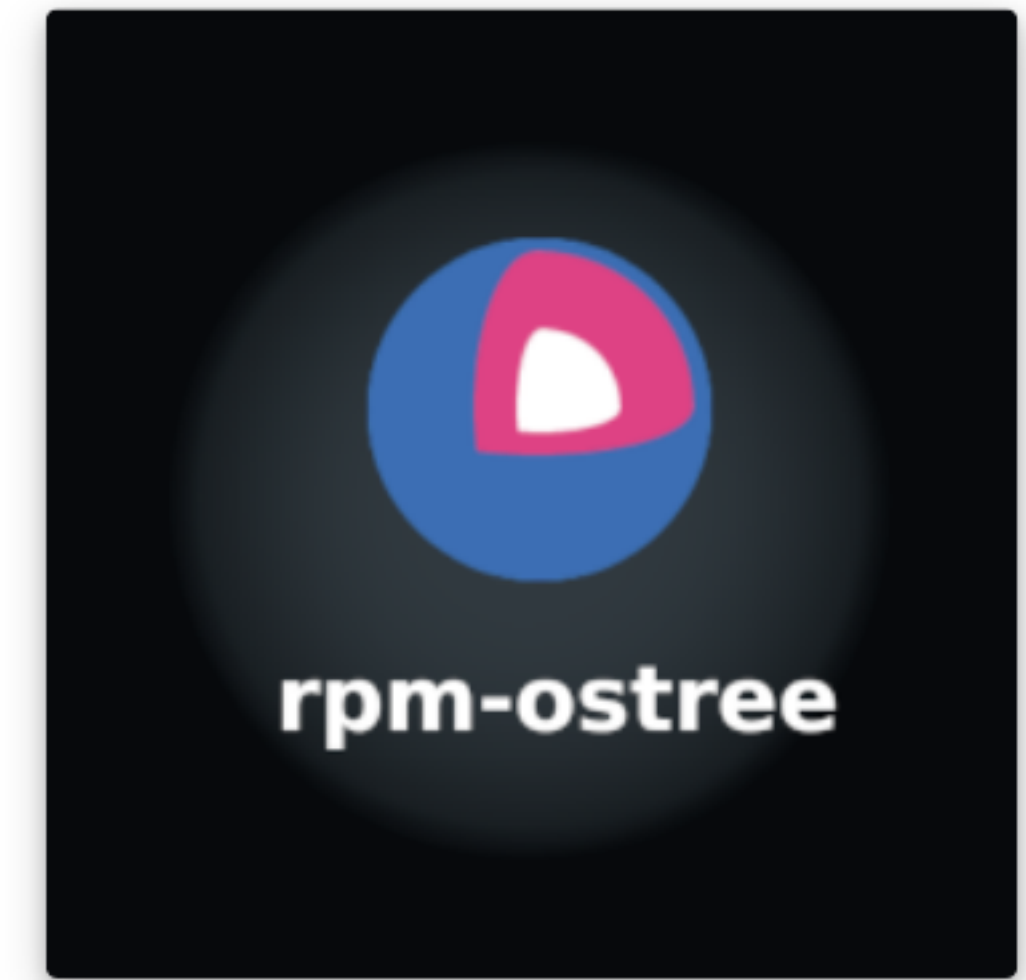


# Technische Grundlagen

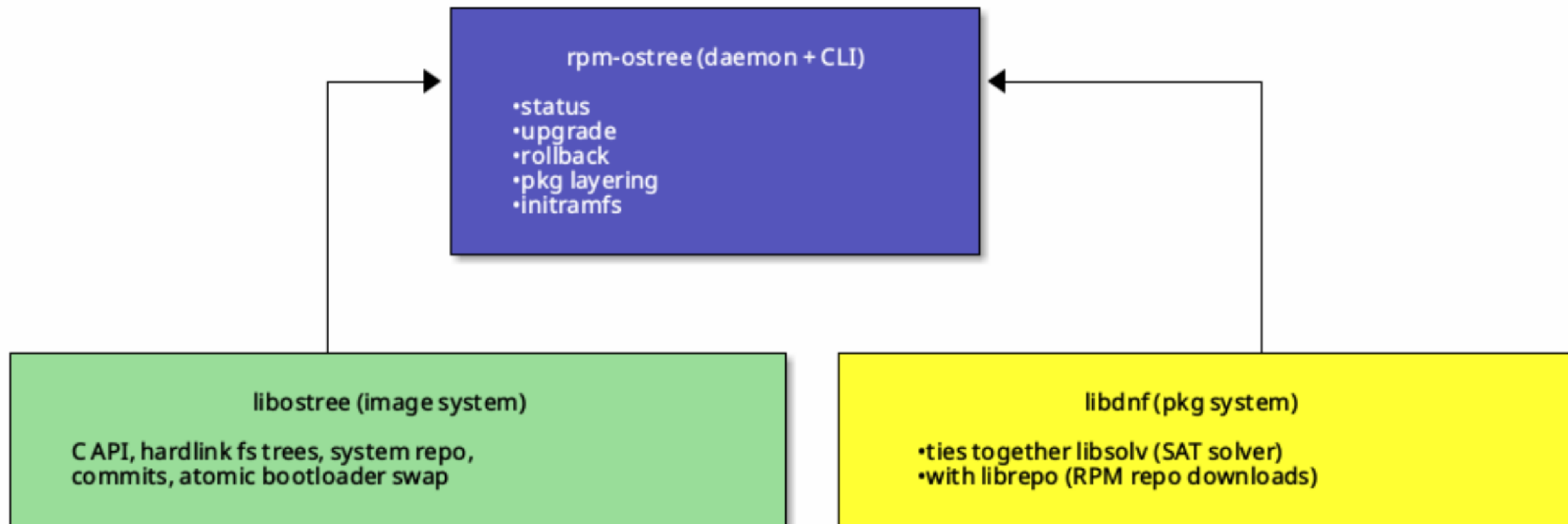
- OSTree-enabled [OCI compliant images](#)
- Hosted auf [ghcr.io](#)
  - 90 Tage Image Archiv
  - Images sind signiert mit cosign
- Kernel Module vorinstalliert (je nach Image Nvidia, Framework etc)
- Daten und Applikationen getrennt
- Rebase zurück zu Fedora jederzeit ohne Neuinstallation möglich
- Keine eigene Distributionen
  - Angepasste Layer, die man auch wieder entfernen kann

# rpm-ostree

- rpm-ostree ist ein hybrides Image-/Paketsystem
  - kombiniert libostree und libdnf
  - atomare und sichere Upgrades mit lokaler RPM-Paketschichtung (Layer)
- Rebase
- Rollback
  - temporary: Reboot and select prior version
  - permanent: `rpm-ostree rollback`
- Pinning von Images
  - Ich pinne unregelmässig, aber damit habe ich immer mindestens ein funktionierendes Image
  - Theoretisch, ich habe in 8 Monaten keinen Rollback durchgeführt



# rpm-ostree, libostree und libdnf



# rpm-ostree

- Weisser Punkt zeigt das aktuell gebootet Image
- Es wird ein weiteres Image vorgehalten (2. in der Liste)
- Drittes Image Pinned: yes bleibt zusätzlich

## Pin Image

```
sudo ostree admin pin 0
```

## Unpin Image

```
sudo ostree admin pin -u 2
```

```
stoeps@lnx-stwsh ~/devel/silverblue/ublu-stoeps main rpm-ostree status
State: idle
Deployments:
  ostree-image-signed:docker://ghcr.io/stoeps13/ublu-stoeps-nvidia:latest
    Digest: sha256:221efd723e4e938888a380c3259e423f4174d0957331d0046ea8d3100134ad97
    Version: 40.20240807.0 (2024-08-07T07:18:28Z)
    Diff: 96 upgraded, 82 removed, 58 added
  • ostree-image-signed:docker://ghcr.io/stoeps13/ublu-stoeps-nvidia:latest
    Digest: sha256:593458874fbaecd17a68392605efd604733192164db68e90627c4f6fe29ac0c8
    Version: 40.20240801.0 (2024-08-01T18:22:41Z)
  ostree-image-signed:docker://ghcr.io/stoeps13/ublu-stoeps-nvidia:latest
    Digest: sha256:3cee80385166b30c7287a05ca32c235f20aa2ed034048e0d3b2324d4fcf78b00
    Version: 40.20240801.0 (2024-08-01T05:53:20Z)
  ostree-image-signed:docker://ghcr.io/stoeps13/ublu-stoeps-nvidia:latest
    Digest: sha256:ccd12183c2e883baa0b627e3bd8c3cbd8ef3af7e6826bfb05f3615b90dff75a0
    Version: 40.20240525.0 (2024-05-27T11:22:27Z)
    Pinned: yes
```

## rpm-ostree status



# Beispiel unpin und pin (1)

```
ostree-image-signed:docker://ghcr.io/stoeps13/ubluue-stoeps-main:latest
  Digest: sha256:ce0027dd2a7624296b0033c1ee4c040805d5268701c96d74842f87171a3c4abf
  Version: 40.20240719.0 (2024-07-19T17:14:14Z)
  Pinned: yes
stoeps@stoeps-t470 ~/.dotfiles  main ±  sudo ostree admin pin -u 2 1
[sudo] password for stoeps:
Deployment 2 is now unpinned
stoeps@stoeps-t470 ~/.dotfiles  main ±  rpm-ostree status
State: idle
Deployments:
● ostree-image-signed:docker://ghcr.io/stoeps13/ubluue-stoeps-sericea-main:latest
  Digest: sha256:d8d4ced786e3d1775a31e578f9399e0cee892fc0b5649382e67a1d576cc08f1c
  Version: 40.20240803.0 (2024-08-04T13:32:20Z)

ostree-image-signed:docker://ghcr.io/stoeps13/ubluue-stoeps-sericea-main:latest
  Digest: sha256:5cf41c504bfba7ae0bc5961c99f8e65f35a00bbd3abaf9666eb33b7a654f2154
  Version: 40.20240801.0 (2024-08-01T18:20:51Z)

ostree-image-signed:docker://ghcr.io/stoeps13/ubluue-stoeps-main:latest
  Digest: sha256:ce0027dd2a7624296b0033c1ee4c040805d5268701c96d74842f87171a3c4abf
  Version: 40.20240719.0 (2024-07-19T17:14:14Z)
stoeps@stoeps-t470 ~/.dotfiles  main ±  sudo ostree admin pin 2 2
```

# Beispiel unpin und pin (2)

```
stoeps@stoeps-t470 > ~/.dotfiles > ⌘ main ± > sudo ostree admin pin 2
Deployment 2 is now pinned
stoeps@stoeps-t470 > ~/.dotfiles > ⌘ main ± > rpm-ostree status
State: idle
Deployments:
● ostree-image-signed:docker://ghcr.io/stoeps13/ubblue-stoeps-sericea-main:latest
  Digest: sha256:d8d4ced786e3d1775a31e578f9399e0cee892fc0b5649382e67a1d576cc08f1c
  Version: 40.20240803.0 (2024-08-04T13:32:20Z)

  ostree-image-signed:docker://ghcr.io/stoeps13/ubblue-stoeps-sericea-main:latest
    Digest: sha256:5cf41c504bfba7ae0bc5961c99f8e65f35a00bbd3abaf9666eb33b7a654f2154
    Version: 40.20240801.0 (2024-08-01T18:20:51Z)

  ostree-image-signed:docker://ghcr.io/stoeps13/ubblue-stoeps-main:latest
    Digest: sha256:ce0027dd2a7624296b0033c1ee4c040805d5268701c96d74842f87171a3c4abf
    Version: 40.20240719.0 (2024-07-19T17:14:14Z)
    Pinned: yes
```



# Rebase des Systems

*# rebase to special tag*

```
rpm-ostree rebase ostree-image-signed:docker://ghcr.io/stoeps13/bluefin-dx:39-20231217
```

*# rebase Bluefin-dx (Developer edition) latest*

```
rpm-ostree rebase ostree-image-signed:docker://ghcr.io/stoeps13/bluestoeps-dx:latest
```

*# rebase auf Ublue-os main image*

```
rpm-ostree rebase ostree-image-signed:docker://ghcr.io/ubblue-os/sericea-main:latest
```

*# rebase auf custom bluefin image latest*

```
rpm-ostree rebase ostree-image-signed:docker://ghcr.io/stoeps13/bluestoeps-dx:latest
```

*# rebase auf custom image mit universal-blue template*

```
rpm-ostree rebase ostree-image-signed:docker://ghcr.io/stoeps13/ubblue-stoeps-main:latest
```

*# rebase auf sway wm image (custom, latest)*

```
rpm-ostree rebase ostree-image-signed:docker://ghcr.io/stoeps13/ubblue-stoeps-sericea-main:latest
```

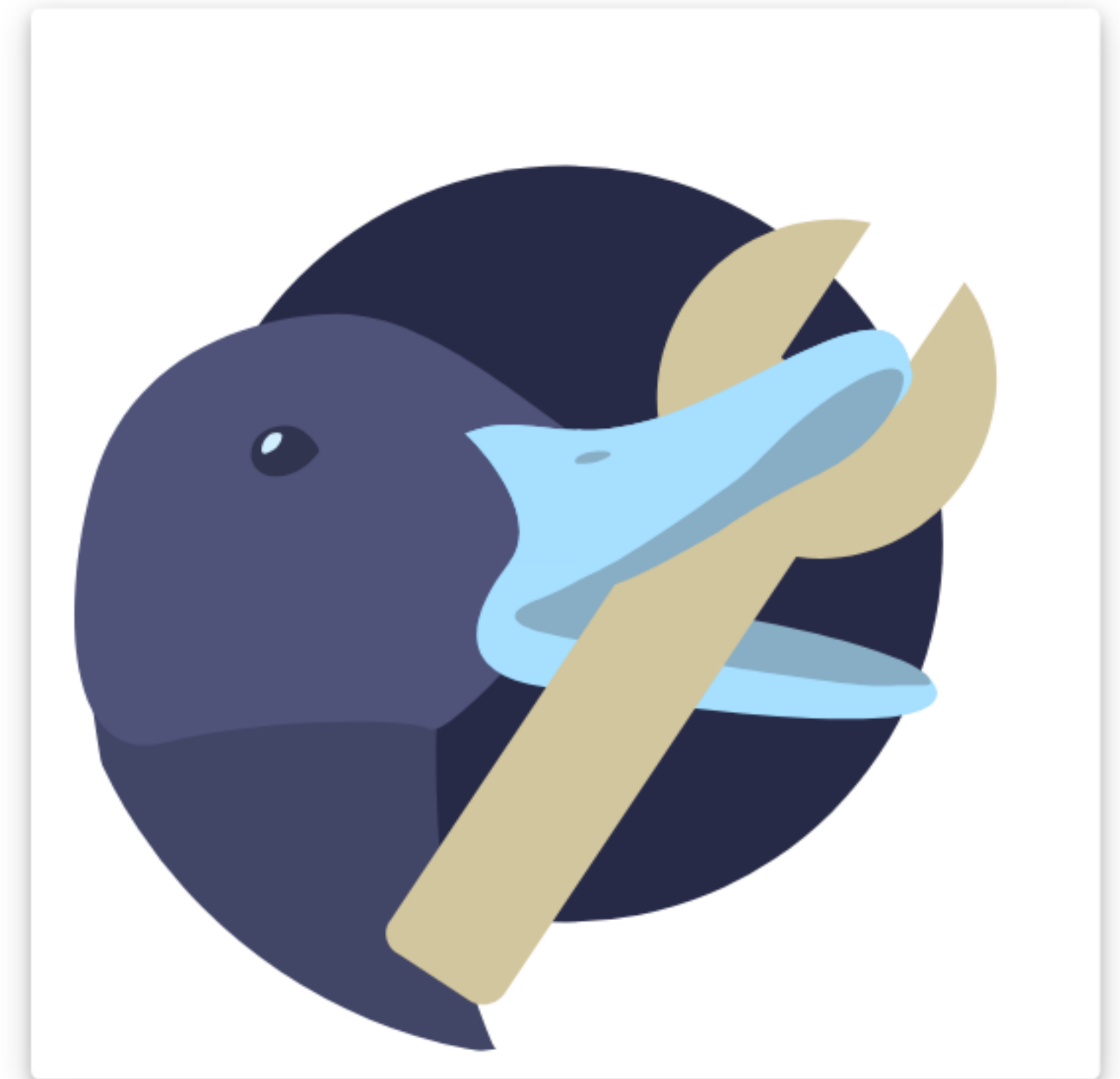
# Layer mit rpm-ostree

- rpm-ostree kann auch geänderte lokale Images per commit in neue Images verwandeln (analog Container)
- Dateisystem
  - /etc mutable Konfigurationsverzeichnis
  - /usr Read-only (Binaries und Data)
  - /var All anderen Stati
  - /opt muss man beim Image build erstellen, oder nach /var/opt umziehen



# Build your own Image

- <https://github.com/ubblue-os/image-template>
  - Erstellt images auf Basis von Containerfile
- Blue build: <https://blue-build.org>
  - Build über yaml Datei(en), modular erweiterbar
    - Recipes / Rezepte werden in Containerfile übersetzt
  - <https://github.com/blue-build/template>
- Die folgenden Beispiele nutzen Blue Build



# Neues Git Repository

- <https://blue-build.org/how-to/setup/>
  - Automatic setup using the BlueBuild Workshop
    - Web Service (Login with Github)
    - Erstellt Repository with Default Konfiguration und cosign
  - Manual setup with git and cosign
    - Selbst ist der Mensch
- Prozess ist gut dokumentiert, daher direkt weiter mit unserem Rezept

# Blue Build - root Verzeichnis

- <https://github.com/stoeps13/ubblue-stoeps>

```
|— boot_menu.yml  
|— Containerfile  
|— cosign.key  
|— cosign.pub  
|— files  
|— LICENSE  
|— README.md  
|— recipes
```

# Blue Build - recipes Verzeichnis

```
— recipes
  — common_modules
    — akmods.yml
    — bling.yml
    — containerfile.yml
    — default-flatpaks.yml
    — files.yml
    — fonts.yml
    — rpm-ostree-gnome.yml
    — rpm-ostree-sway.yml
    — rpm-ostree.yml
    — script-sericea.yml
    — script.yml
    — systemd.yml
    — yafti.yml
  — README.md
  — recipe-main.yml
  — recipe-nvidia.yml
  — recipe-sericea-main.yml
  — recipe-sericea-nvidia.yml
```



# Blue Build - files Verzeichnis

```
files
├── etc
│   ├── systemd
│   │   └── system
│   │       └── flatpak-system-update.timer
├── scripts
│   ├── example.sh
│   ├── signing.sh
│   ├── workarounds-sericea.sh
│   └── workarounds.sh
└── usr
    ├── etc
    │   ├── containers
    │   │   ├── registries.d
    │   │   └── ublue-os.yaml
    │   └── distrobox
    │       └── distrobox.ini
    └── share
        ├── backgrounds
        │   ├── default.png
        │   ├── iceland.png
        │   └── README.md
```

# Blue-Build Rezept

- Yaml Datei

```
# image will be published to ghcr.io/<user>/<name>
name: ublue-stoepe-main
# description will be included in the image's metadata
description: My ublue image for my personal and working devices

base-image: ghcr.io/ublue-os/silverblue-main
image-version: latest # latest is also supported if you want new updates ASAP

modules:
- from-file: common_modules/files.yml
#- from-file: common_modules/akmods.yml
- from-file: common_modules/bling.yml
- from-file: common_modules/default-flatpaks.yml
- from-file: common_modules/fonts.yml
- from-file: common_modules/rpm-ostree.yml
- from-file: common_modules/rpm-ostree-gnome.yml
- from-file: common_modules/yafti.yml
- from-file: common_modules/systemd.yml
- from-file: common_modules/script.yml
- from-file: common_modules/containerfile.yml
- type: signing
```

# Installation Fonts

`common_modules/fonts.yml`

```
type: fonts
fonts:
  nerd-fonts:
    - FiraCode # don't add "Nerd Font" suffix.
    - SourceCodePro
  google-fonts:
    - Roboto
    - Open Sans
```



# Installation und Entfernen von Paketen

common\_modules/rpm-ostree.yml

```
type: rpm-ostree
install:
  - bat
  - eza
  - fd-find
  - firewall-config
  - flameshot
  - fzf
  - stow
  - tmux
  - vdirsyncer
  - vim
  - virt-install
  - virt-manager
  - virt-viewer
  - zsh
remove:
  - firefox
  - firefox-langpacks
```

# Flatpaks vorinstallieren

`.common_modules/default_flatpaks.yaml`

```
type: default-flatpaks
notify: true
system:
  repo-url: https://dl.flathub.org/repo/flathub.flatpakrepo
  repo-name: flathub
  install:
    - com.mattjakeman.ExtensionManager
    - org.davmail.DavMail
    - org.gnome.font-viewer
    - org.localsend.localsend_app
user:
  repo-url: https://dl.flathub.org/repo/flathub.flatpakrepo
  repo-name: flathub
  install:
    - com.bitwarden.desktop
    - com.github.tchx84.Flatseal
    - com.logseq.Logseq
    - io.github.java_decompiler.jd-gui
```

- Alternative Liste in `yaftei.yaml` bereitstellen
  - Wird beim 1. Boot aufgerufen und BenutzerInnen können Flatpaks zur Installation auswählen

# Binaries aus anderen Containern

- Werden als zusätzliche Layer ins Image kopiert

```
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



# Workarounds per Shell Skript

## common\_modules/scripts.sh

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

## files/scripts/workaround.sh

```
#!/bin/sh

set -oue pipefail
echo 'This is the workaround shell script'

# Disable wayland on Lenovo T470 (prevents OBS crashes)
/usr/libexec/gdm-runtime-config set daemon WaylandEnable false

# Edit vdirsyncer google.py to make it work with gmail
sed -i 's!urn:ietf:wg:oauth:2.0:oob!http://127.0.0.1:8088!g' \
    $(fd google.py /usr/lib | grep vdirsyncer)

# Clean up repos, everything is on the image so we don't need them
for i in $(ls /etc/yum.repos.d/ | grep -v '^fedora' | grep -v rpmfusion); do
    rm -f /etc/yum.repos.de/${i}
done
```

# Zusätzliche Kernelmodulen

- Liste verfügbarer Module

`common_modules/akmods.yml`

```
type: akmods
install:
  - v4l2loopback
```

# Lokaler Build

- Images die auf blue-build basieren bringen bereits die bluebuild cli mit
- Ausgabe des Rezepts:

```
bluebuild generate -d recipes/recipe-main.yml
```

- Ausgabe des Containerfiles

```
bluebuild generate recipes/recipe-main.yml
```

- Lokaler Build des Images

```
bluebuild build recipes/recipe-main.yml
```

 Ich habe keinen Weg gefunden auf ein lokales Images rebase auszuführen.



# Distrobox / toolbox

- Verschiedene Linux Distributionen im Terminal
- Container basiert
  - Integriert in den Host
  - Teilt sich \$HOME mit dem Host
  - Zugriff auf externe Disks
  - Graphische Apps mit X11 und Wayland
  - GPU Zugriff
- Keine speziellen Container notwendig
  - jegliches Container Image von Docker Hub, Quay.io oder Custom Registry funktioniert
- Unterstützt Podman, Docker oder LiliPod

# Distrobox Container starten / erstellen

## Distrobox Container erstellen

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

## Automatisiert in `distrobox.ini`

```
[test]
additional_packages="vim tmux"
image=docker.io/almalinux/8-init
init=true
nvidia=false
pull=true
root=false
replace=true
```

# Shell im Container

- Container nutzen normal die gleich \$SHELL wie unser host
- chsh im Container wechselt die Shell
- Prompt im Container anpassen

```
if [ -n "$CONTAINER_ID" ]; then
    # show container name inside distrobox containers
    psvar[1]="@${(%) :- $CONTAINER_ID 🧳}"
else
    # show hostname inside podman containers
    psvar[1]="@${(%) :- $(hostname)}"
fi
```



# Home-Verzeichnis

- Default container teilen sich \$HOME

## Container mit separatem home erstellen

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

# root Zugriff aus dem Container

- Normale Container verlangen sudo für die Installation von Paketen
- Manchmal braucht man aber `root` z.B. für Netzwerkanalyse
- Mit podman hat ein Container nur den Zugriff den ein normaler User auch hat
  - `sudo` gibt hier keinen vollständigen `root` Zugriff
- `sudo distrobox` ist nicht supportet!
- Container mit `-r` oder `--root` erstellen

```
distrobox create --name test --image your-chosen-image:tag --root
```

# Host Applikation im Container starten

- docker
- podman
- neovim (Beispiel Kali)

## Kommandozeile

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

## distrobox.ini

```
...  
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/podman;
```



# Container Apps vom Host starten

/usr/etc/distrobox/distrobox.ini

```
[kali]
additional_packages="burpsuite zaproxy bloodhound neo4j"
image=ghcr.io/stoeps13/kali-toolbox:latest
icon=/home/stoeps/Pictures/Kali-dragon-icon.svg.png
exported_apps="burpsuite zaproxy"
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/podman;
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/podman-compose;
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/xdg-open;
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/buildah;
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/rpm-ostree;
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/flatpak;
init_hooks=ln -sf /usr/bin/distrobox-host-exec /usr/bin/nvim;
init=false
nvidia=false
pull=true
root=true
replace=true
```

# Copy text from distrobox to host

- über tmux oder terminal per Maus
- Direkt aus der Console
  - wlroots (wayland) oder xsel (X11)

```
wl-copy < file-to-copy  
wl-copy text to copy
```

- Oder man verwendet die Maus und `Ctrl` + `Shift` + `c` im Terminal

# Toolbx / Distrobox container anpassen / selbst bauen

- in der `distrobox.ini` können zusätzliche Apps beim Laden auf den Client hinzugefügt werden
  - Containererstellung dauert dann länger
  - Falls etwas schief geht, muss man den alten Container wiederherstellen
- Besser die wichtigsten Tools automatisiert hinzufügen



# Image für distrobox bauen

- <https://github.com/ubblue-os/toolbox>
- Containerfile
- packages

```
toolboxes/fedora-toolbox  
├─ Containerfile.fedora  
└─ packages.fedora
```

## packages.fedora – Liste mit Paketen

```
ImageMagick  
ansifilter  
automake  
bat  
chromium  
curl
```

# Containerfile

```
FROM quay.io/fedora/fedora-toolbox:40
LABEL com.github.containers.toolbox="true" \
      usage="This image is meant to be used with the toolbox or distrobox command" \
      summary="A cloud-native terminal experience powered by Fedora"
COPY ./toolboxes/fedora-toolbox/packages.fedora /toolbox-packages
RUN dnf -y upgrade && dnf -y install $(<toolbox-packages) && rm /toolbox-packages
# Install lazygit
RUN dnf copr enable atim/lazygit -y && dnf install -y lazygit && dnf clean all
# Install language server for neovim
RUN npm install -g dockerfile-language-server-nodejs && npm install -g vscode-json-languageserver &&
  npm i -g vscode-langservers-extracted && npm install -g yaml-language-server
# Install mkdocs
RUN pip install mkdocs mkdocs-material mkdocs-section-index ansible-core<2.17
# Install AsciiDoctor
RUN gem install asciidoctor-diagram asciidoctor-pdf asciidoctor-revealjs asciidoctor rouge
# Fix vdirsyncer
RUN sed -i 's!urn:ietf:wg:oauth:2.0:oob!http://127.0.0.1:8088!g' $(fd google.py /usr/lib | grep vdi:
COPY --from=cgr.dev/chainguard/helm:latest /usr/bin/helm /usr/bin/helm
RUN wget https://raw.githubusercontent.com/ahmetb/kubectx/master/kubectx -O /usr/bin/kubectx && \
  chmod +x /usr/bin/kubectx
```

# Dotfiles

- Meine dotfiles liegen in einem privaten git-Repository
  - Noch keine Lösung um einen OAuth Token nicht einchecken zu müssen
  - Passwörter mit pass s abgespeichert und in den dotfiles ausgeführt
- Ins jeweilige Home clonen

```
git clone https://.../mydotfiles.git ~/.dotfiles
```

```
/home/stoeps/.dotfiles
├── alacritty
│   ├── .config
│   │   └── alacritty
│   │       ├── alacritty.toml
│   │       ├── alacritty.yml
│   │       └── catppuccin-mocha.toml
├── asciidoctor
│   ├── .asciidoctor
│   │   └── theme
│   │       ├── images
│   │       │   ├── .gitignore
│   │       │   ├── logo_200.png
│   │       │   ├── logo.png
│   │       │   └── vegardit-logo.png
│   │       ├── LICENSE
│   │       ├── README.adoc
│   │       └── stoeps-theme.yml
└── bash
    ├── .bash_history
    ├── .bash_logout
    ├── .bash_profile
    └── .bashrc
```



# Stow

- <https://www.gnu.org/software/stow>
- Ursprünglich entstand stow um Dateien in unabhängigen Softwarepaketen (perl) zu verwalten
  - Sehr gut für die Verwaltung von Konfigurationsdateien im Home-Verzeichnis geeignet
  - Einfach versionierbar
- Verlinkt die Dateien aus ~/ .dotfiles

```
cd ~/.dotfiles
stow neovim
```

```
nvim → ../.dotfiles/neovim/.config/nvim
```

```
/home/stoeps/.dotfiles/neovim
├── .config
│   └── nvim
│       ├── init.lua
│       ├── lua
│       │   └── core
│       │       ├── keymaps.lua
│       │       ├── plugin_config
│       │       │   ├── gitsigns.lua
│       │       │   ├── init.lua
│       │       │   ├── lsp_config.lua
│       │       │   ├── lua_line.lua
│       │       │   ├── nvim-tree.lua
│       │       │   ├── telescope.lua
│       │       │   ├── theme.lua
│       │       │   └── treesitter.lua
│       │       └── plugins.lua
│       └── plugin
│           └── packer_compiled.lua
```

# Chezmoi

- Bluebuild hat ein Chezmoi Module
- Systemd service der Dotfiles aus Repo abgleicht
- Habe ich aber noch nicht getestet
- <https://www.chezmoi.io>
- <https://blue-build.org/reference/modules/chezmoi/>

# Just

- <https://just.systems/>
- Ersatz / Ergänzung zu make / Makefile

`/usr/share/ubblue-os/just/00-default.just`

```
...  
# Show all messages from this boot  
logs-this-boot:  
    sudo journalctl -b 0  
  
# Show all messages from last boot  
logs-last-boot:  
    sudo journalctl -b -1  
  
# Regenerate GRUB config, useful in dual-boot scenarios where a second operating system isn't listed  
regenerate-grub:  
    #!/usr/bin/bash  
    if [ -d /sys/firmware/efi ]; then  
        sudo grub2-mkconfig -o /etc/grub2-efi.cfg  
    else  
        sudo grub2-mkconfig -o /etc/grub2.cfg  
    fi  
...
```

# Custom Just

- Eigene Shortcuts für Kommandos erstellen

`/usr/share/ubblue-os/just/60-custom.just`

```
# vim: set ft=make :  
# This file can be modified downstream to add custom just commands
```



Answers 1km →

