Development Environments as Code

Johannes Hüttinger

johannes.huettinger@aformatik.de

aformatik – Training & Consulting
www.aformatik.de

Wozu Entwicklungsumgebungen?

Wozu Entwicklungsumgebungen?

1. Wartung

Tools zur Entwicklung eines Projektes bereitstellen

#Erstellung #Updates
#Rollbacks #Automatisierung
#Reproduzierbarkeit

Wozu Entwicklungsumgebungen?

1. Wartung

Tools zur Entwicklung eines Projektes bereitstellen

#Erstellung #Updates
#Rollbacks #Automatisierung
#Reproduzierbarkeit

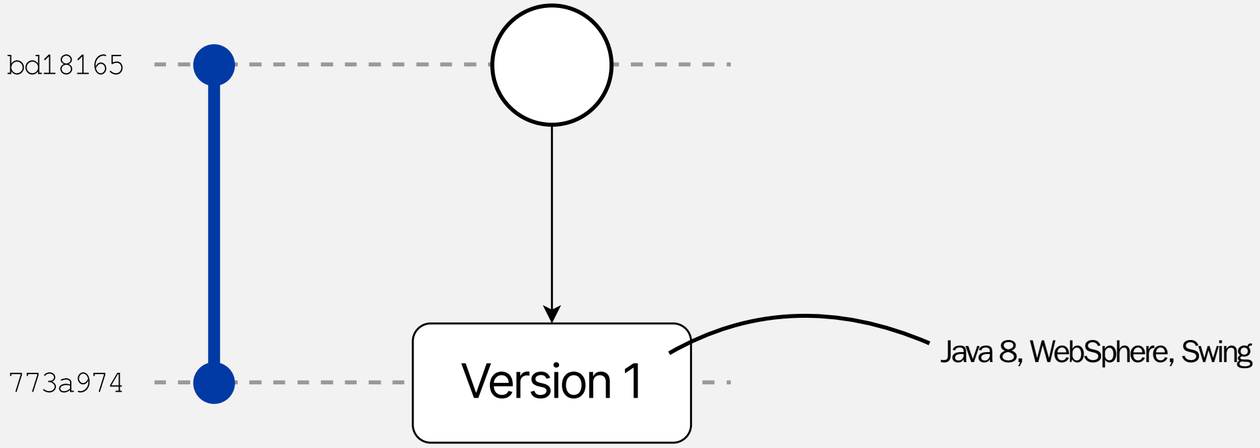
2. Developer Experience

Entwickler entwickeln lassen

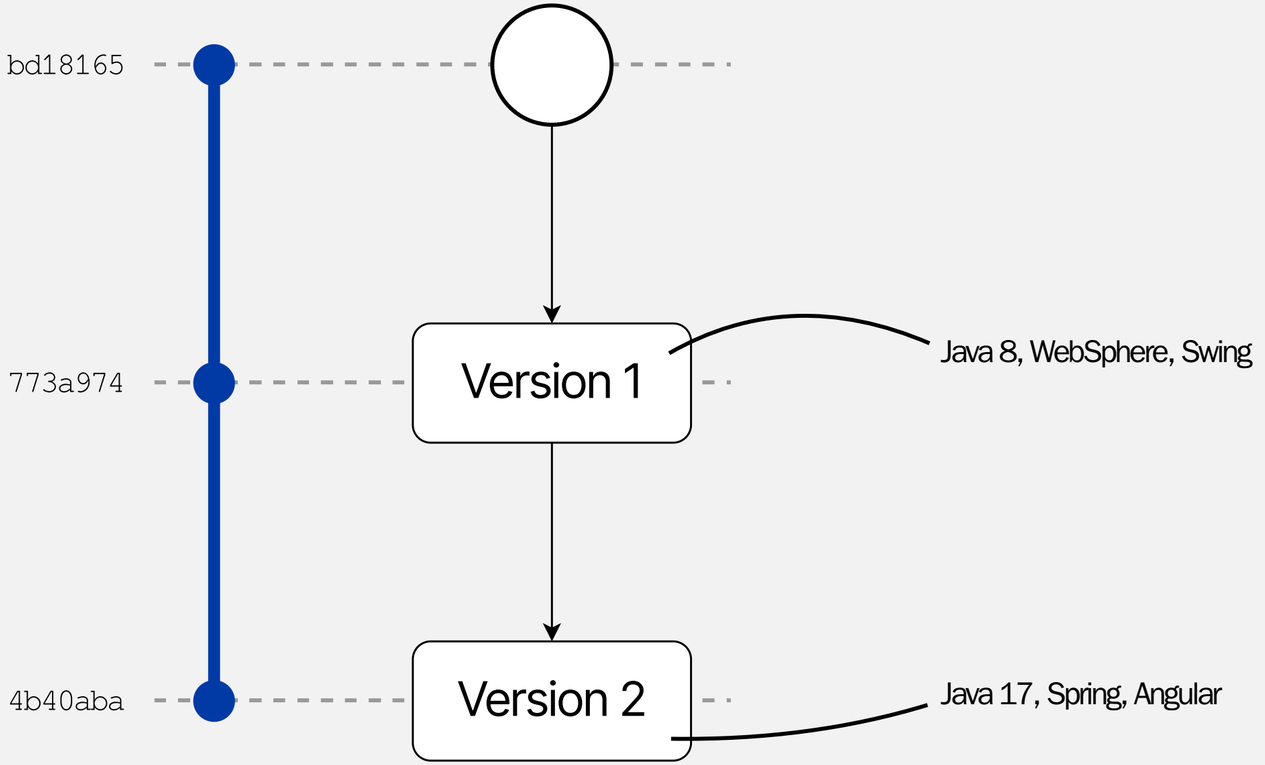
#Integration #Performance
#Individualisierung

Entwicklung einer Entwicklungsumgebung

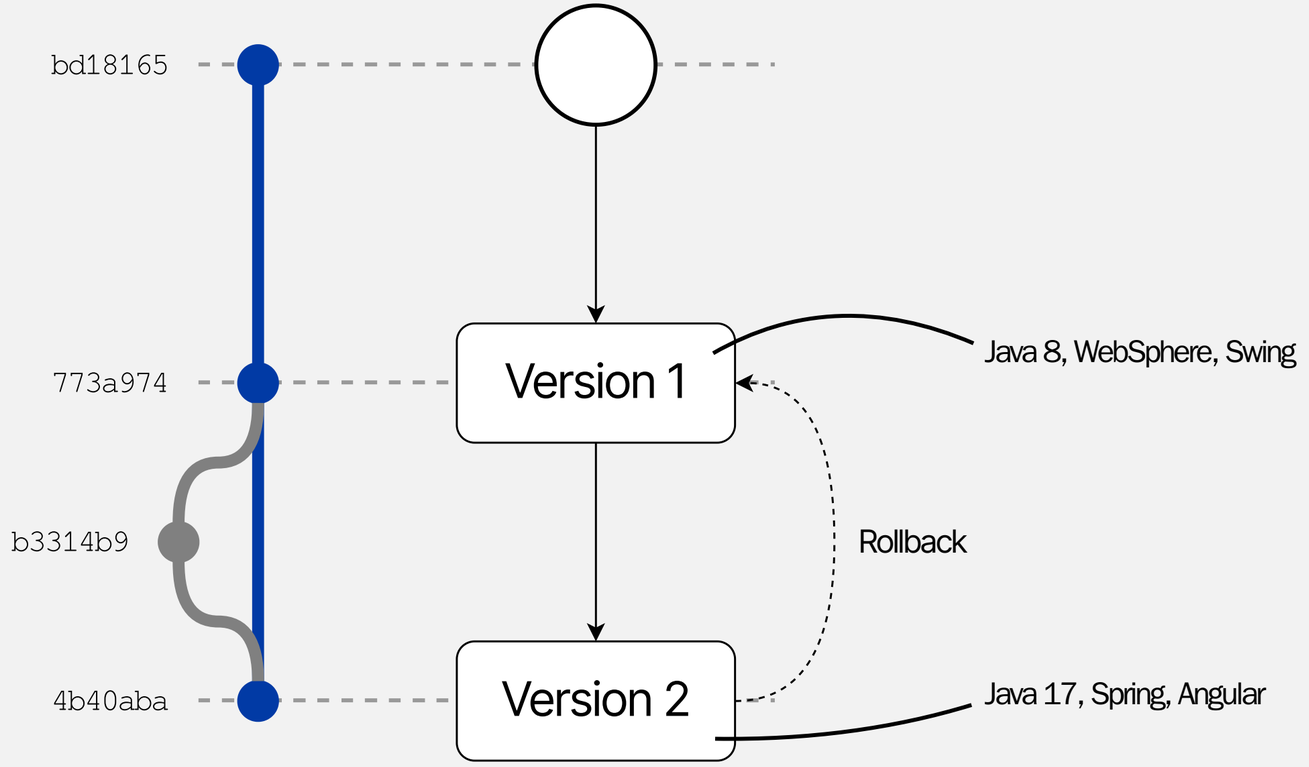
Entwicklung einer Entwicklungsumgebung



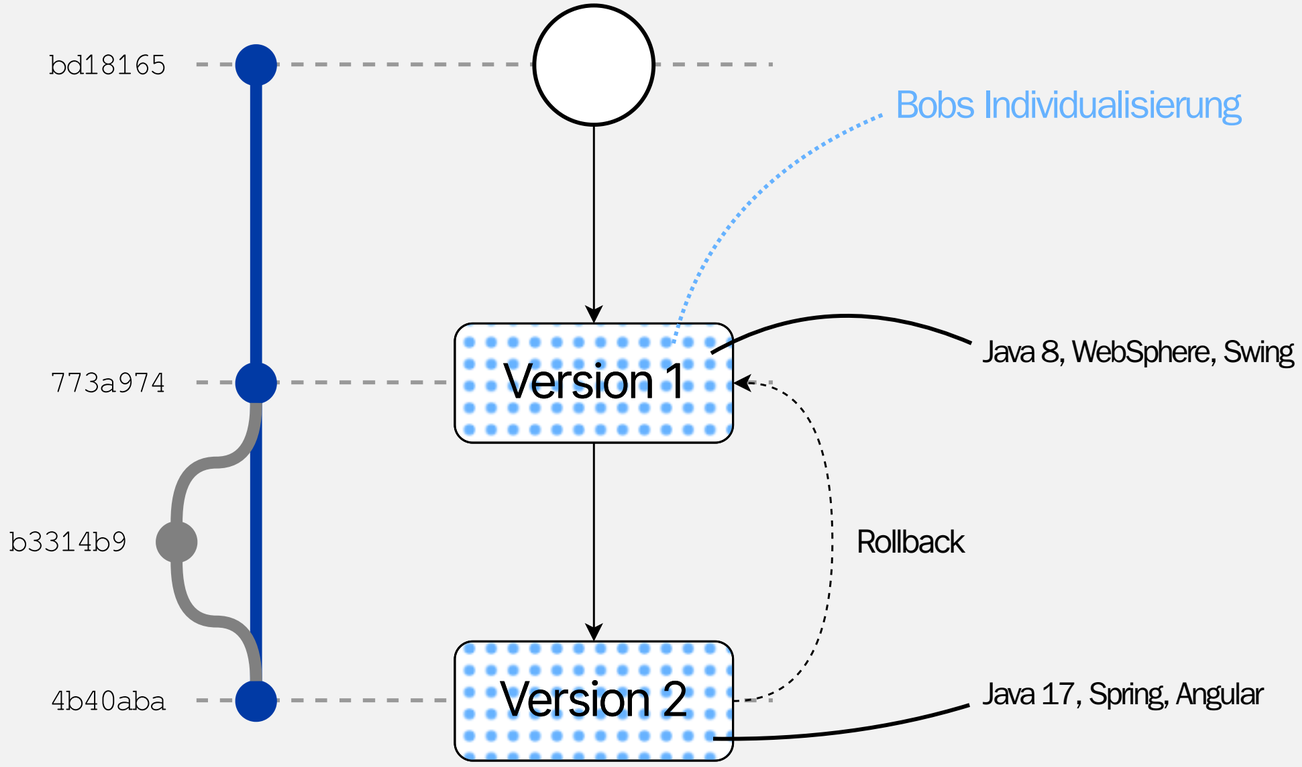
Entwicklung einer Entwicklungsumgebung



Entwicklung einer Entwicklungsumgebung



Entwicklung einer Entwicklungsumgebung



Stand der Dinge

Stand der Dinge

~~❓ Manuell / Bare Metal~~

Stand der Dinge

- ~~❓ Manuell / Bare Metal~~
- 🟧 Virtuelle Maschinen

Stand der Dinge

~~❓ Manuell / Bare Metal~~

🟧 Virtuelle Maschinen

🟩 ⚠️ VS Code + Docker
(Devcontainers / Codespaces)

Stand der Dinge

~~❓ Manuell / Bare Metal~~

🟧 Virtuelle Maschinen

🟩 ⚠️ VS Code + Docker
(Devcontainers / Codespaces)

🟩 Vagrant + Ansible

Stand der Dinge

❓ ~~Manuell / Bare Metal~~

🟧 Virtuelle Maschinen

🟩 ⚠ VS Code + Docker
(Devcontainers / Codespaces)

🟩 Vagrant + Ansible

...

Stand der Dinge

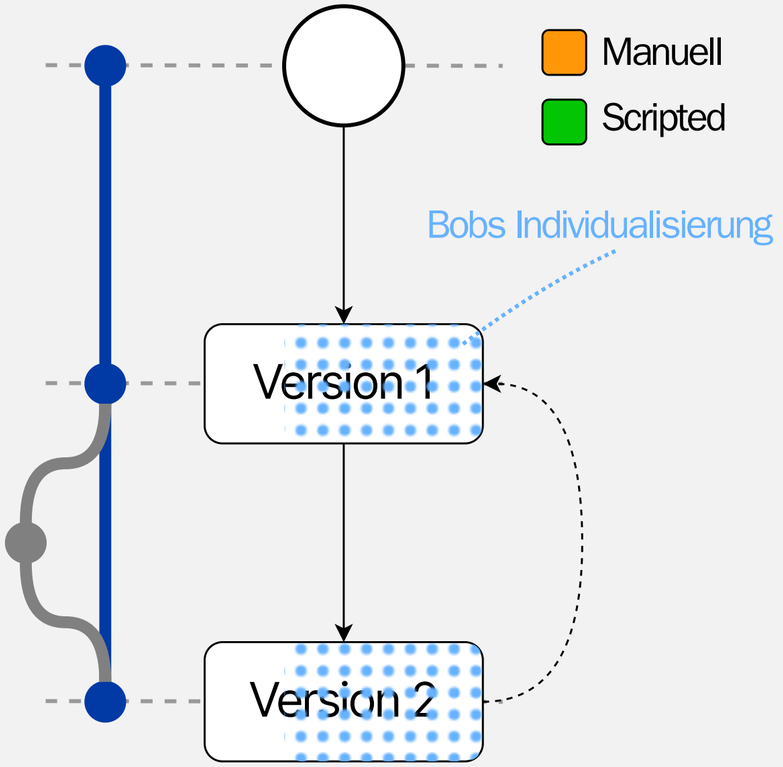
~~❓ Manuell / Bare Metal~~

🟧 Virtuelle Maschinen

🟩 ⚠ VS Code + Docker (Devcontainers / Codespaces)

🟩 Vagrant + Ansible

...



Stand der Dinge

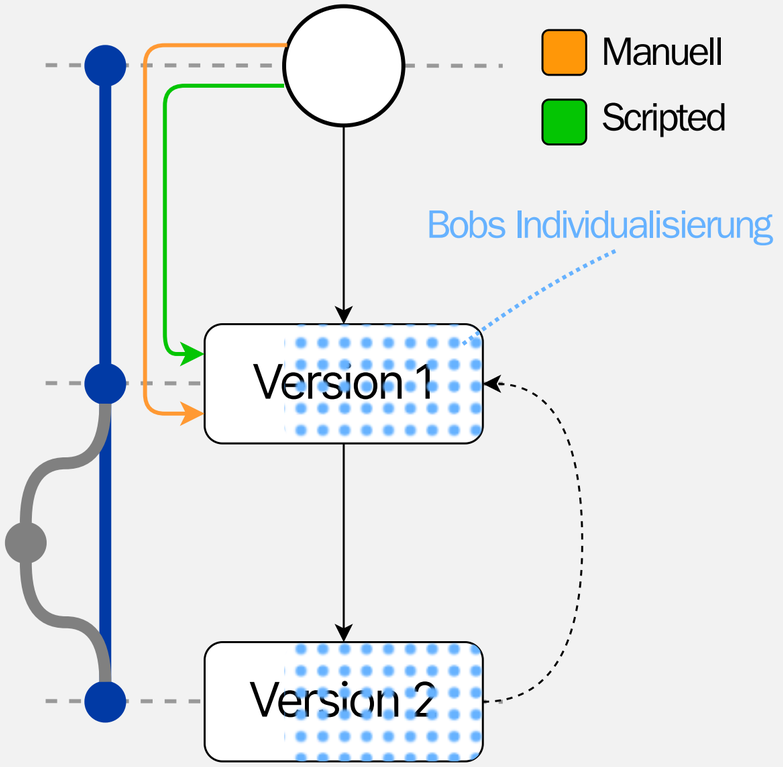
~~❓ Manuell / Bare Metal~~

🟧 Virtuelle Maschinen

🟩 ⚠ VS Code + Docker (Devcontainers / Codespaces)

🟩 Vagrant + Ansible

...



Stand der Dinge

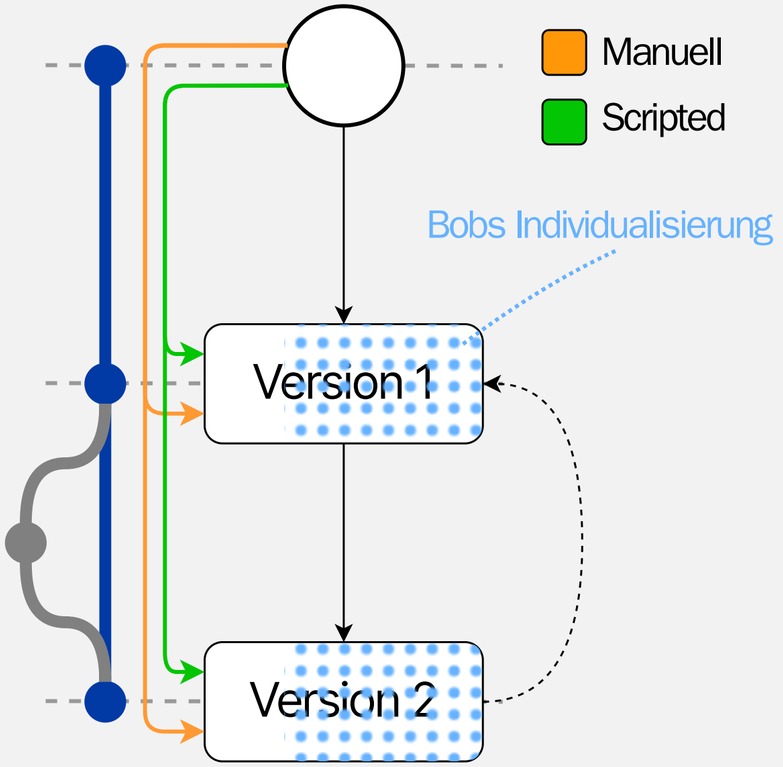
~~❓ Manuell / Bare Metal~~

🟧 Virtuelle Maschinen

🟩 ⚠ VS Code + Docker (Devcontainers / Codespaces)

🟩 Vagrant + Ansible

...



Stand der Dinge

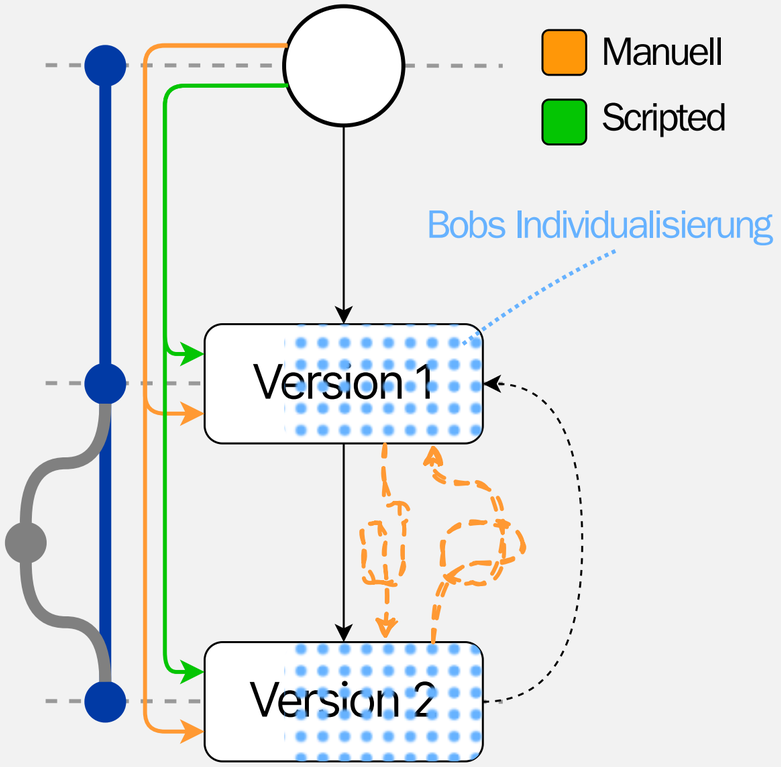
~~❓ Manuell / Bare Metal~~

🟧 Virtuelle Maschinen

🟩 ⚠ VS Code + Docker (Devcontainers / Codespaces)

🟩 Vagrant + Ansible

...



Traditionelles Systemmanagement

```
apt update && apt upgrade
apt install x y z

useradd bob
groupadd postgres
chown -R bob /var/lib/...
vi /etc/ssh/sshd_config
systemd enable sshd.service
```

```
├── etc
│   └── systemd
├── home
├── usr
│   ├── bin
│   ├── lib
│   └── share
└── var/lib
```

Traditionelles Systemmanagement

```
apt update && apt upgrade
apt install x y z

useradd bob
groupadd postgres
chown -R bob /var/lib/...
vi /etc/ssh/sshd_config
systemd enable sshd.service
```

```
├── etc
│   └── systemd
├── home
├── usr
│   ├── bin
│   ├── lib
│   └── share
└── var/lib
```

- Destruktive Inplace-Updates
- Veränderung des globalen Systemzustands
- Nicht-atomar

👉 **Zustandsabhängig**

Traditionelles Systemmanagement

```
apt update && apt upgrade
apt install x y z

useradd bob
groupadd postgres
chown -R bob /var/lib/...
vi /etc/ssh/sshd_config
systemd enable sshd.service
```

- Destruktive Inplace-Updates
- Veränderung des globalen Systemzustands
- Nicht-atomar

👉 **Zustandsabhängig**

```
├── etc
│   └── systemd
├── home
├── usr
│   ├── bin
│   ├── lib
│   └── share
└── var/lib
```

- "Dependency Hell"
- Oft nur *eine* Version einer Software
- Keine unpriviligierte Installation

Imperativ vs. Funktional

Imperativ vs. Funktional

```
static int x;

int add(int a, int b) {
    ...
    Date.now()
    ...
    new Random()
    ...
    x++
    ...
}
```

Imperativ vs. Funktional

```
static int x;

int add(int a, int b) {
    ...
    Date.now()
    ...
    new Random()
    ...
    x++
    ...
}
```

- **Mutable** by default
- Uneingeschränkte Seiteneffekte

Imperativ vs. Funktional

```
static int x;

int add(int a, int b) {
    ...
    Date.now()
    ...
    new Random()
    ...
    x++
    ...
}
```

```
plus :: Int -> Int -> Int
plus = ...
```

- **Mutable** by default
- Uneingeschränkte Seiteneffekte

Imperativ vs. Funktional

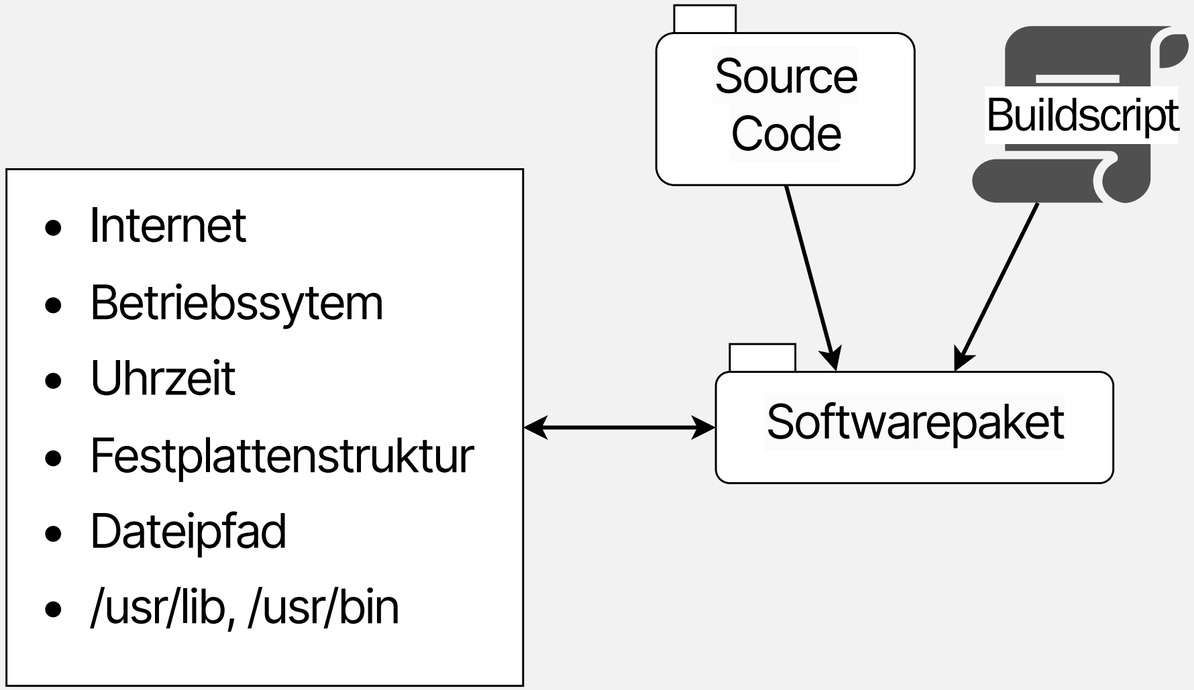
```
static int x;

int add(int a, int b) {
    ...
    Date.now()
    ...
    new Random()
    ...
    x++
    ...
}
```

- **Mutable** by default
- Uneingeschränkte Seiteneffekte

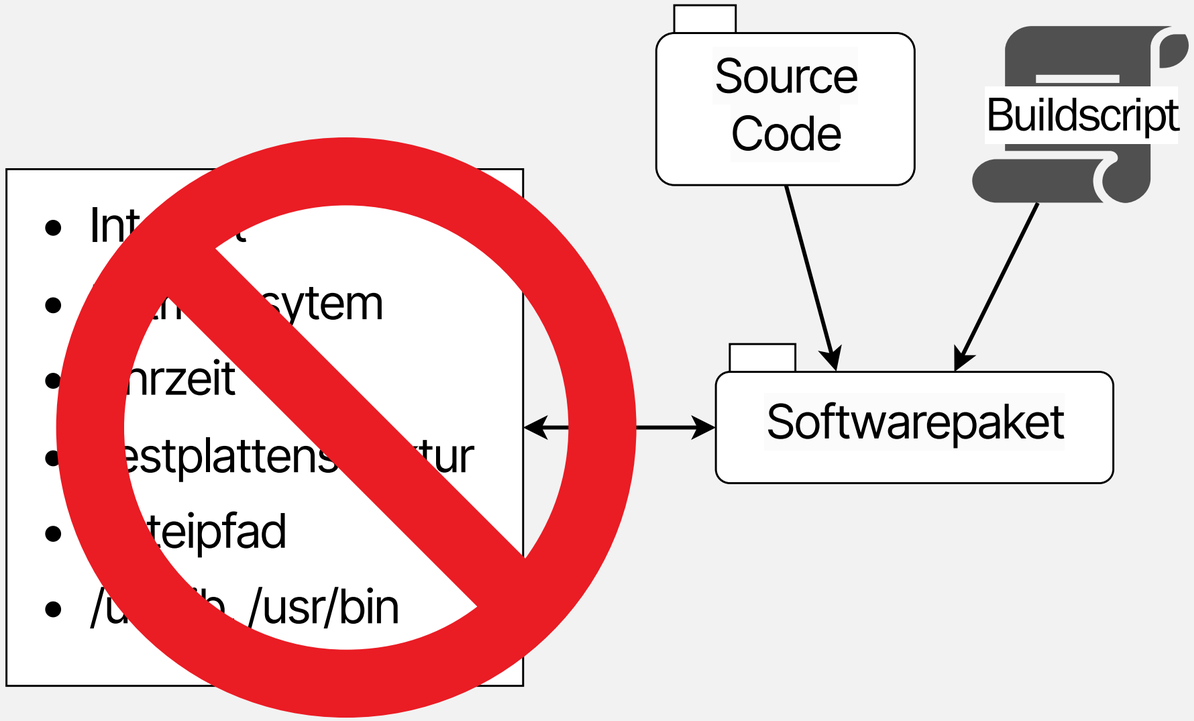
```
plus :: Int -> Int -> Int
plus = ...
```

- **Immutable** by default
- Referentielle Transparenz

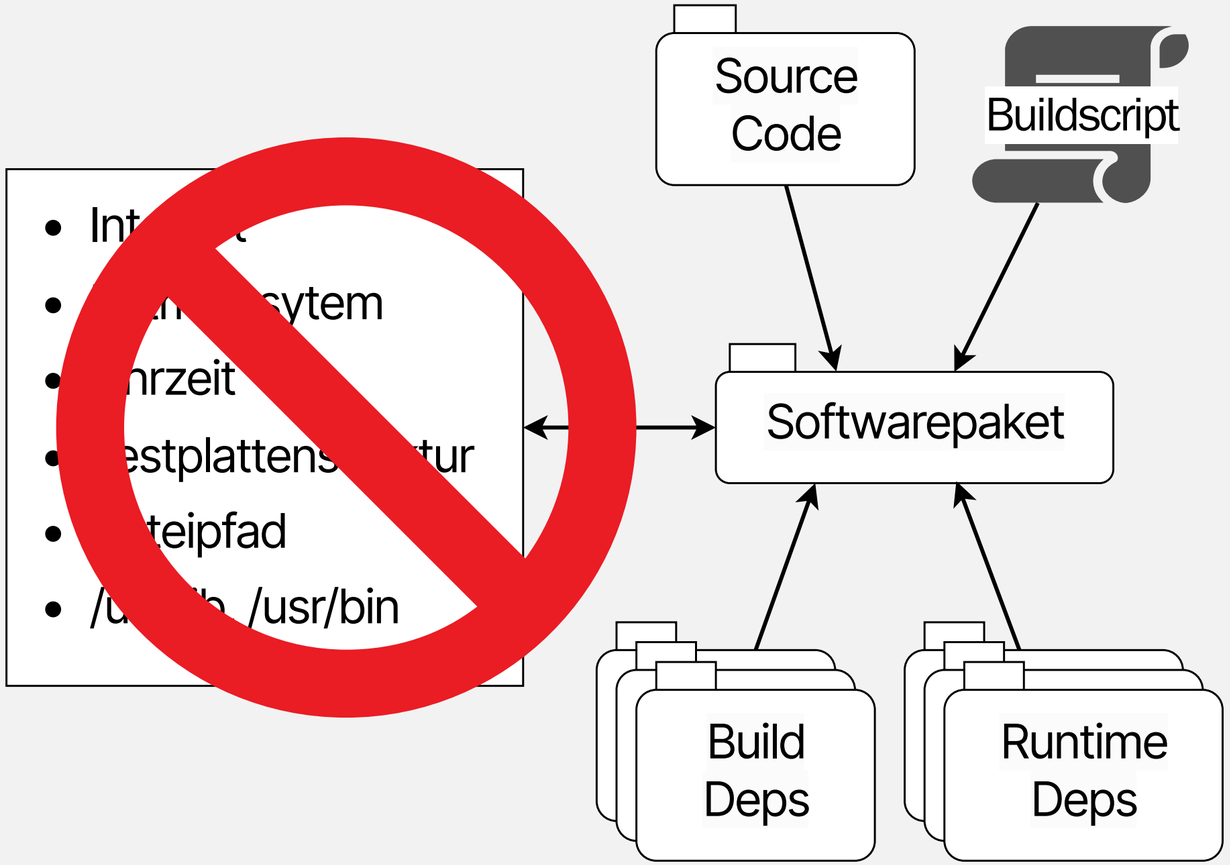
Funktionales Paketmanagement



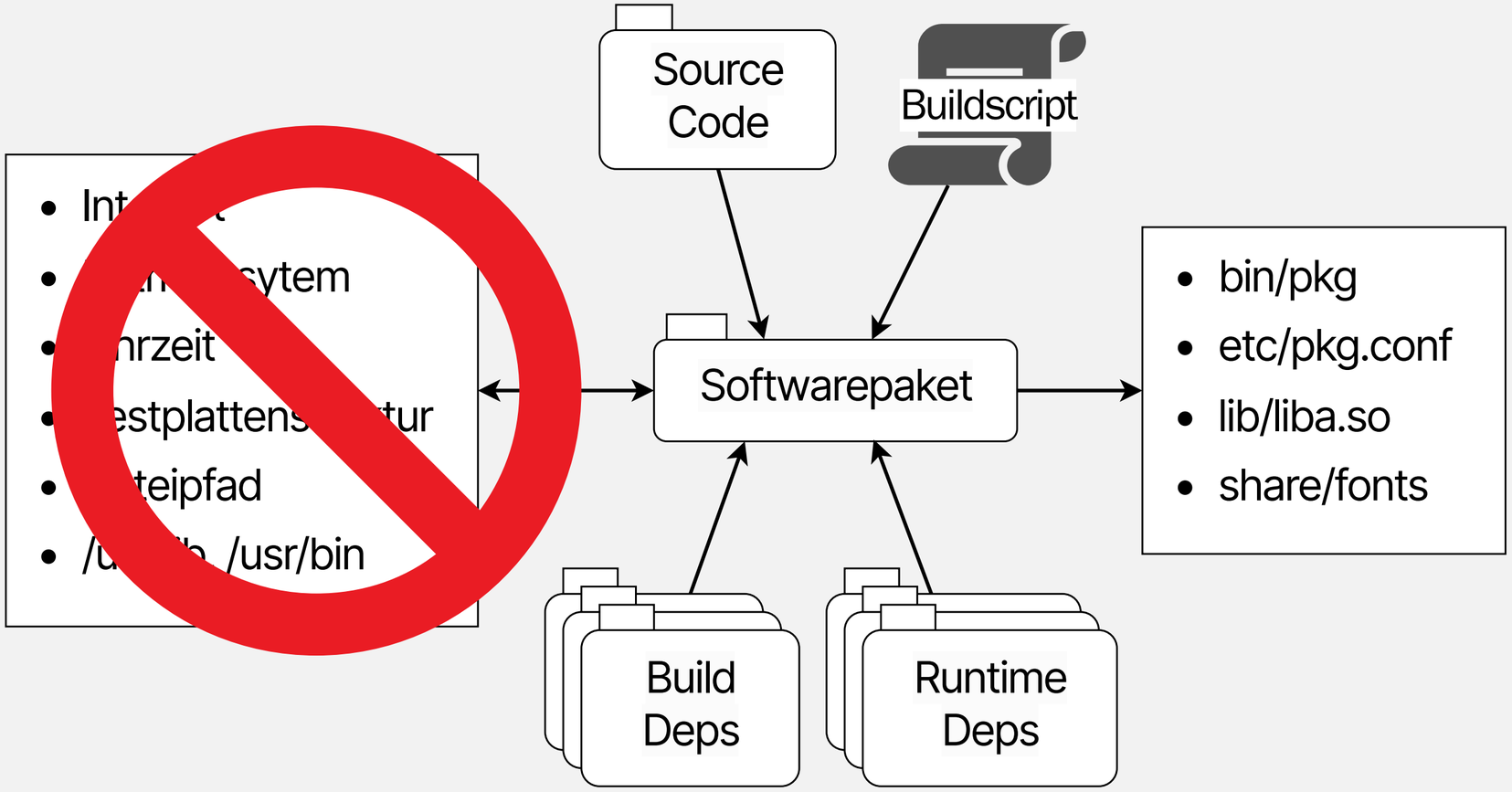
Funktionales Paketmanagement



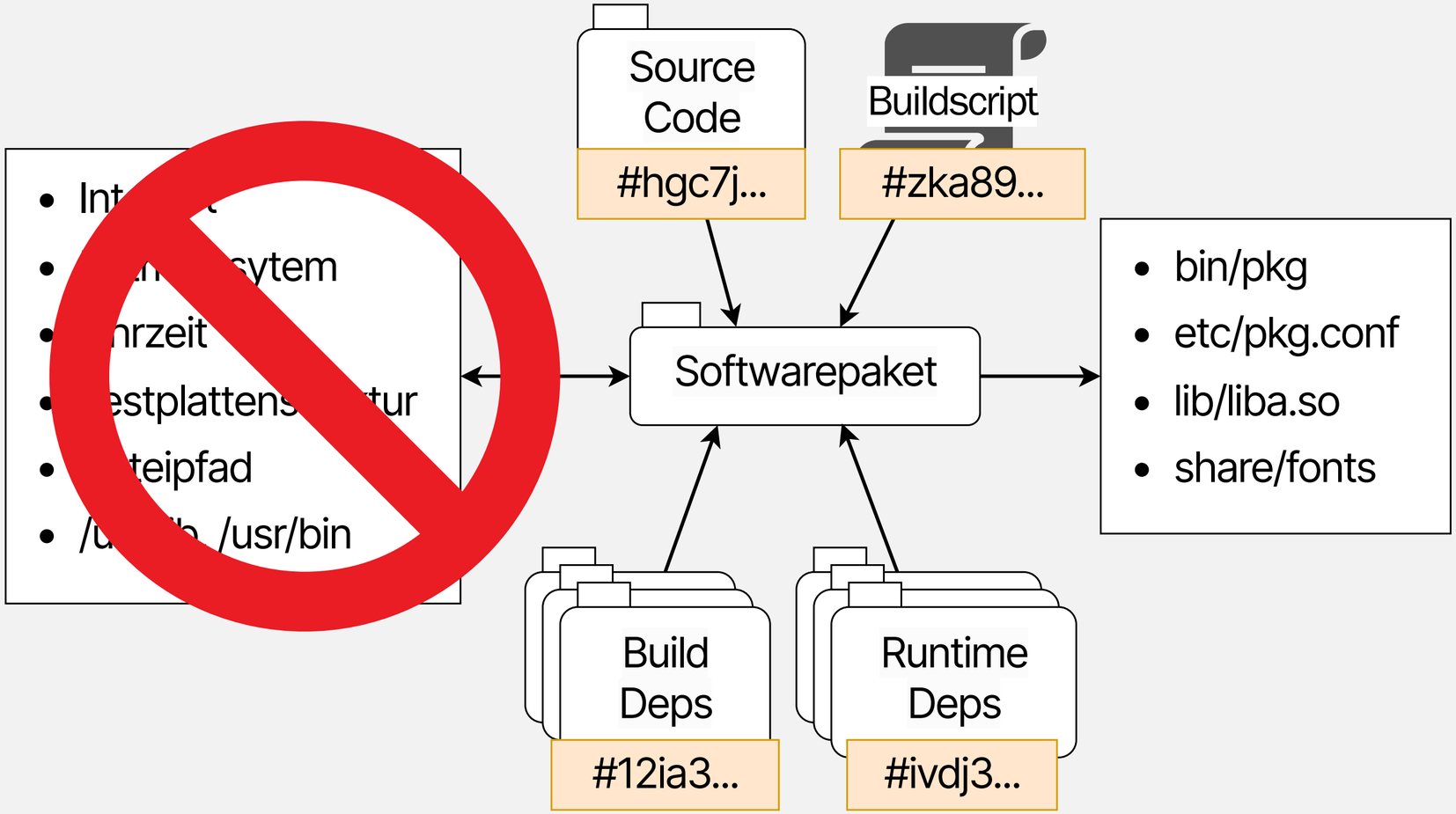
Funktionales Paketmanagement



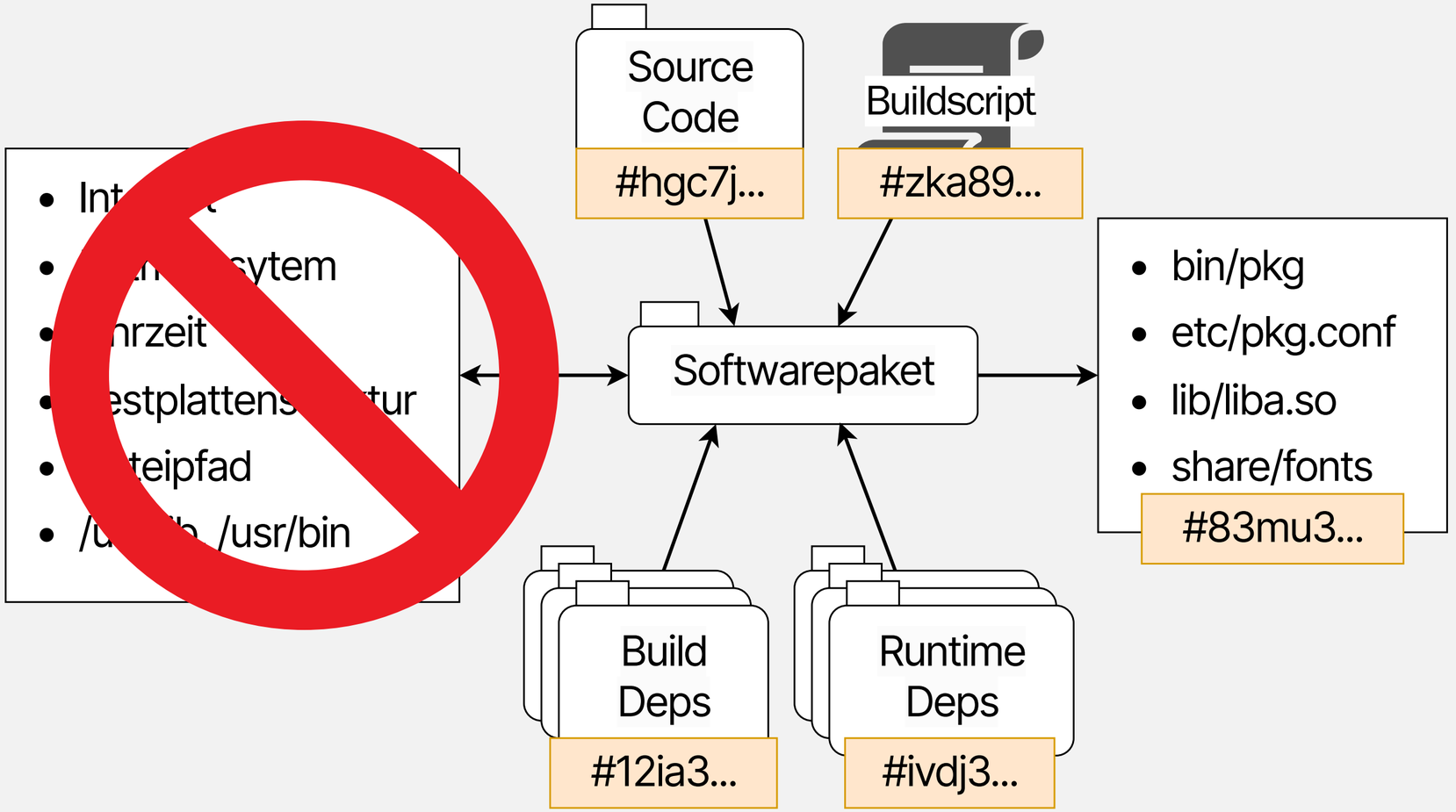
Funktionales Paketmanagement



Funktionales Paketmanagement



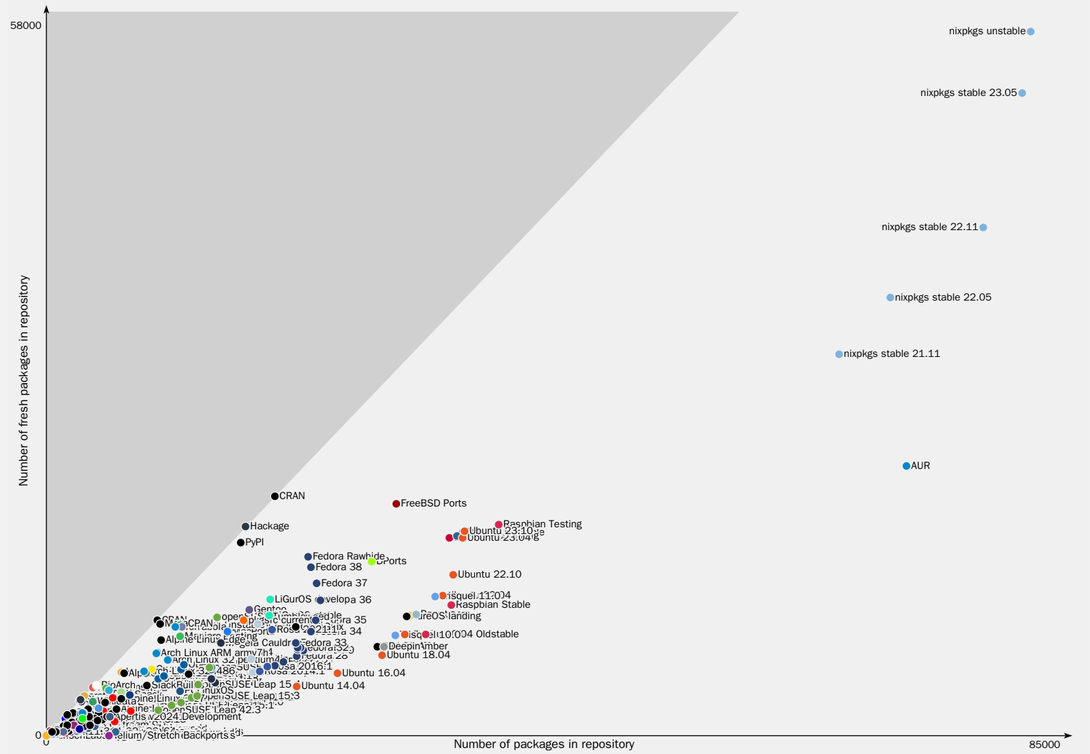
Funktionales Paketmanagement



Nix

Stars 13k

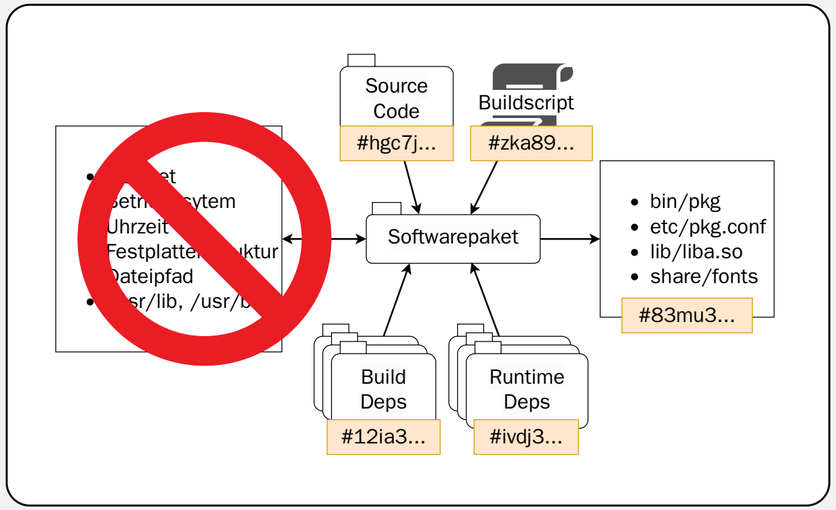
“Nix is a purely functional package manager and deployment system for POSIX”



Quelle: Repology

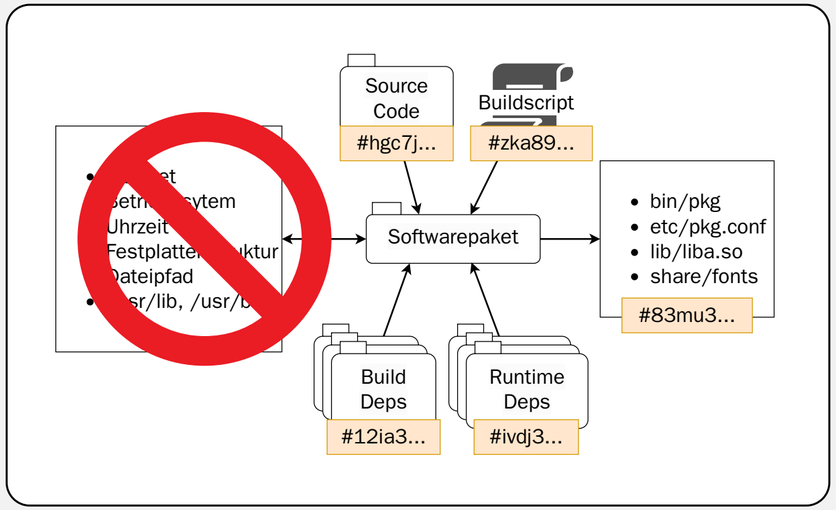
Nix Store

Nix Store


Source Code
#hgc7j...
Buildscript
#zka89...
Softwarepaket
bin/pkg
etc/pkg.conf
lib/liba.so
share/fonts
#83mu3...
Build Deps
#12ia3...
Runtime Deps
#ivdj3...

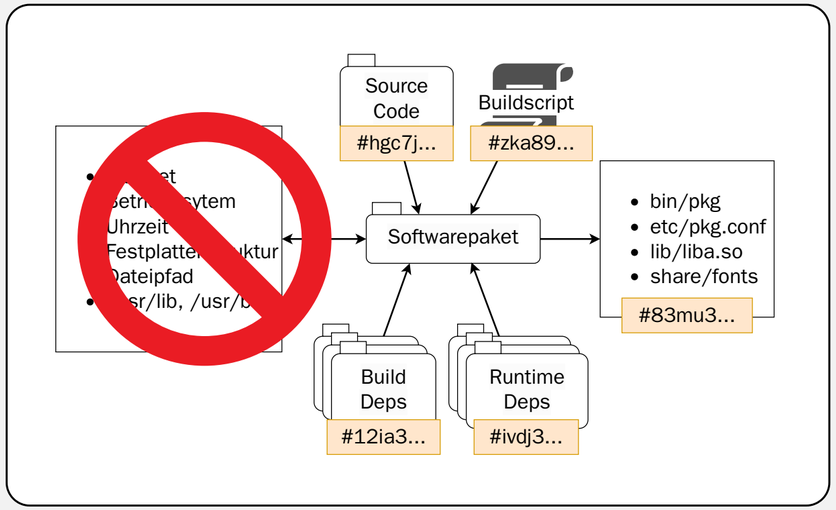
Nix Store

```
/nix/store
├── aa82aeb2d6...jdk17
│   └── bin
│       └── javac
├── c3e8d7680c...jre17
│   └── bin
│       └── java
├── 12c9a5ced0...hello-world
│   ├── bin
│   │   └── hello-world
│   └── lib
│       └── HelloWorld.class
└── 9cc608415e...glibc2.37
    └── lib
        └── libc.so
```



Nix Store

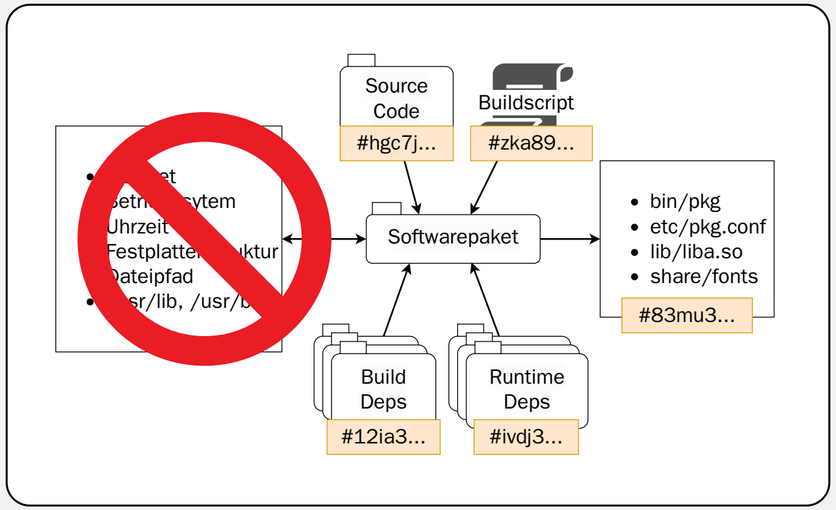
```
/nix/store
├── aa82aeb2d6...jdk17
│   └── bin
│       └── javac
├── c3e8d7680c...jre17
│   └── bin
│       └── java
├── 12c9a5ced0...hello-world
│   ├── bin
│   │   └── hello-world
│   └── lib
│       └── HelloWorld.class
└── 9cc608415e...glibc2.37
    └── lib
        └── libc.so
```



- ✅ Immutable / Append-only Store
- ✅ ~~Dependency Hell~~
- ✅ Mehrere Versionen
- ✅ Multi User Installations

Nix Store

```
/nix/store
├── aa82aeb2d6...jdk17
│   └── bin
│       └── javac
├── c3e8d7680c...jre17
│   └── bin
│       └── java
├── 12c9a5ced0...hello-world
│   ├── bin
│   │   └── hello-world
│   └── lib
│       └── HelloWorld.class
└── 9cc608415e...glibc2.37
    └── lib
        └── libc.so
```



- ✅ Immutable / Append-only Store
- ✅ ~~Dependency Hell~~
- ✅ Mehrere Versionen
- ✅ Multi User Installations

```
$ /nix/store/12c9a5ced0...hello-world/bin/hello-world
Hello World!
```

NixOS

NixOS


├── /etc
│   └── profile
├── /home
└── /nix
    ├── var/nix/profiles
    │   └── system
    └── store
        ├── 12c9a5ced0...hello-world-v1
        ├── 52kaj3kf9e...system-1
        ├── a7g8k2lf9o...hello-world-v2
        └── bfeaj8415e...system-2
            ├── /bin/hello-world
            └── /etc/profile
export PATH="/nix/store/a7g8k2lf9o...hello-world/bin:..."

NixOS


```
$ hello-world
Hello World!
```

NixOS

- Systemkonfigurationen als Nix-Derivations

```
$ hello-world
Hello World!
```

NixOS

- Systemkonfigurationen als Nix-Derivations
- Atomare Updates und Rollbacks

```
$ hello-world
Hello World!
```

NixOS

- Systemkonfigurationen als Nix-Derivations
- Atomare Updates und Rollbacks
- Granulare Kontrolle des Systemzustands

```
$ hello-world
Hello World!
```

NixOS

- Systemkonfigurationen als Nix-Derivations
- Atomare Updates und Rollbacks
- Granulare Kontrolle des Systemzustands
- Build-Targets: Bare Metal, ISOs, VM-Images, Docker-Images, ...

```
$ hello-world
Hello World!
```

Systemkonfiguration mit NixOS

github.com/htngr/jfs-23-deac/configuration.nix

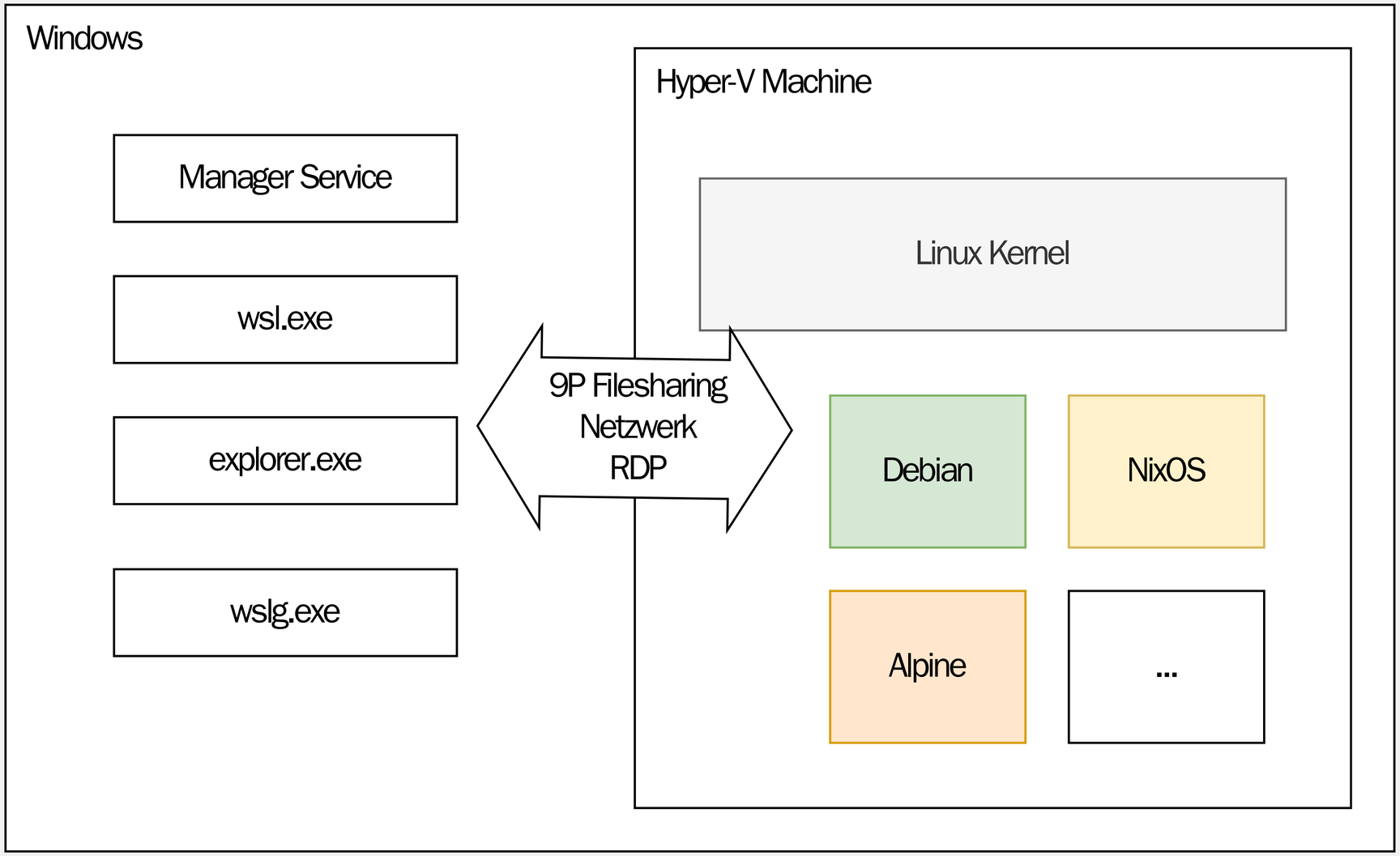
NixOS Search (Packages/Options)

NixOS - Driver

github.com/NixOS/nixpkgs/nixos/modules/virtualisation

```
# configuration.nix
{ pkgs, modulesPath, ... }: {

  imports = [
    # VMware derivation: config.system.build.vmwareImage
    "${modulesPath}/virtualisation/vmware-image.nix"

    # docker **container** derivation: config.system.build.tarball
    "${modulesPath}/virtualisation/docker-image.nix"
  ];

  ...
}
```

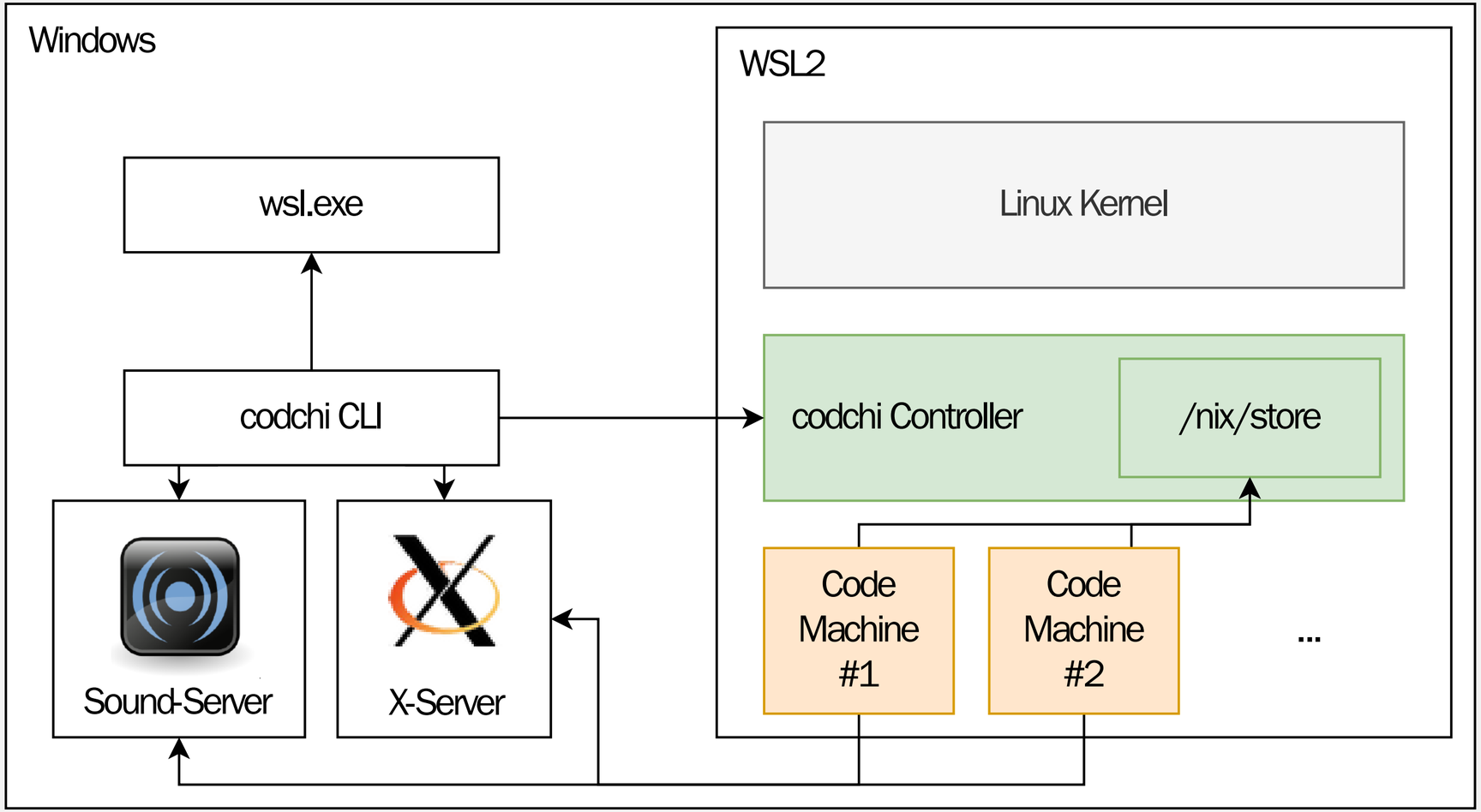
Windows Subsystem for Linux



NixOS-WSL

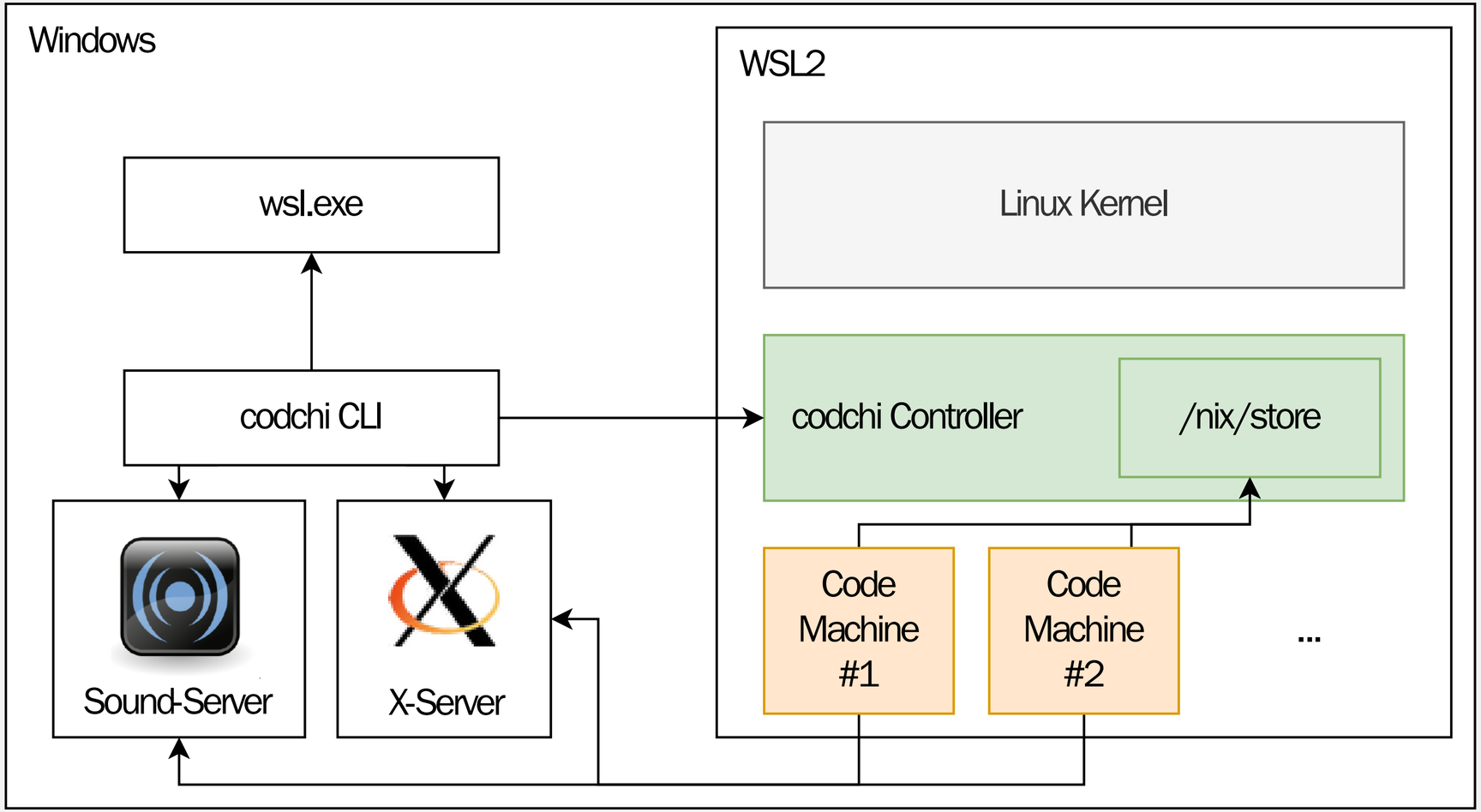
github.com/htngr/jfs-23-deac/wsl.nix

NixOS-WSL


Windows
WSL2
wsl.exe
Linux Kernel
codchi CLI
codchi Controller
/nix/store
Code Machine #1
Code Machine #2
...
Sound-Server
X-Server

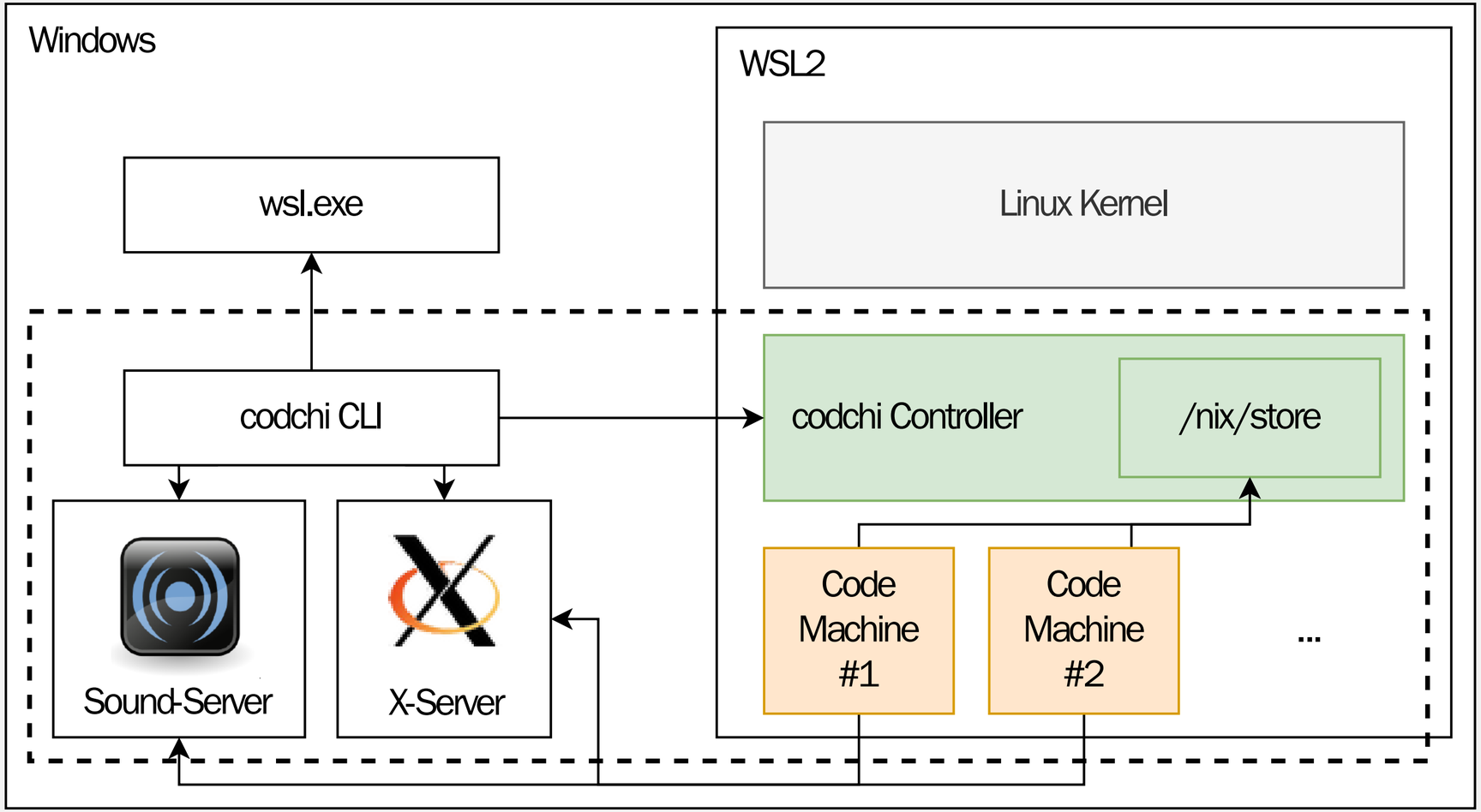
codchi – Code Machines

codchi.dev github.com/aformatik/codchi



codchi – Code Machines

codchi.dev github.com/aformatik/codchi



codchi auf Windows

github.com/aformatik/codchi/README.md

DEaC mit NixOS - Fazit

DEaC mit NixOS - Fazit

✅ Deklarative Systemkonfiguration statt Scripting

DEaC mit NixOS - Fazit

✅ Deklarative Systemkonfiguration statt Scripting

✅ Entwicklungsumgebung entwickelt sich mit Projekt

DEaC mit NixOS - Fazit

✅ Deklarative Systemkonfiguration statt Scripting

✅ Entwicklungsumgebung entwickelt sich mit Projekt

✅ Automatisierte und Reproduzierbare Installation, Updates und Rollbacks

DEaC mit NixOS - Fazit

- ✅ Deklarative Systemkonfiguration statt Scripting
- ✅ Entwicklungsumgebung entwickelt sich mit Projekt
- ✅ Automatisierte und Reproduzierbare Installation, Updates und Rollbacks
- ✅ Individualisierbare Entwicklungsumgebungen

DEaC mit NixOS - Fazit

- ✅ Deklarative Systemkonfiguration statt Scripting
- ✅ Entwicklungsumgebung entwickelt sich mit Projekt
- ✅ Automatisierte und Reproduzierbare Installation, Updates und Rollbacks
- ✅ Individualisierbare Entwicklungsumgebungen
- ✅ Gute Developer-Experience möglich – Tools wie *codchi* helfen

DEaC mit NixOS - Fazit

✅ Deklarative Systemkonfiguration statt Scripting
✅ Entwicklungsumgebung entwickelt sich mit Projekt
✅ Automatisierte und Reproduzierbare Installation, Updates und Rollbacks
✅ Individualisierbare Entwicklungsumgebungen
✅ Gute Developer-Experience möglich – Tools wie *codchi* helfen

❓ Anderes Konzept von Systemmanagement

DEaC mit NixOS - Fazit

✅ Deklarative Systemkonfiguration statt Scripting
✅ Entwicklungsumgebung entwickelt sich mit Projekt
✅ Automatisierte und Reproduzierbare Installation, Updates und Rollbacks
✅ Individualisierbare Entwicklungsumgebungen
✅ Gute Developer-Experience möglich – Tools wie *codchi* helfen

❓ Anderes Konzept von Systemmanagement
❓ Lernkurve

DEaC mit NixOS - Fazit

✅ Deklarative Systemkonfiguration statt Scripting
✅ Entwicklungsumgebung entwickelt sich mit Projekt
✅ Automatisierte und Reproduzierbare Installation, Updates und Rollbacks
✅ Individualisierbare Entwicklungsumgebungen
✅ Gute Developer-Experience möglich – Tools wie *codchi* helfen

❓ Anderes Konzept von Systemmanagement
❓ Lernkurve
❓ (noch) wenig Dokumentation

Development Environments as Code

Johannes Hüttinger

johannes.huettinger@aformatik.de

aformatik – Training & Consulting
www.aformatik.de

nixos.org

codchi.dev