

# Was sind Linux Container und wie funktionieren sie?

---

Anian Ziegler

24. September 2018

cioplenu

# Was sind Container?

---

# Was sind Container?

- Virtualisierung auf Betriebssystemebene

# Was sind Container?

- Virtualisierung auf Betriebssystemebene
- Basiert auf mehreren Features des Linux Kernels

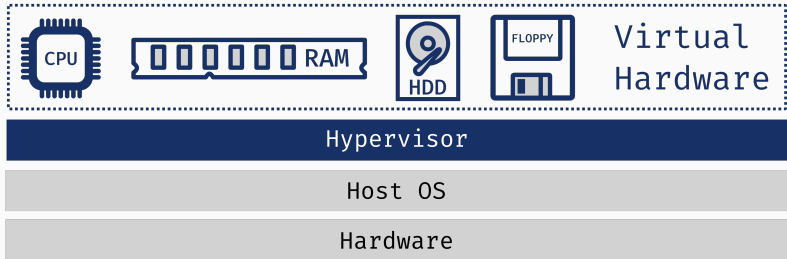
# Was sind Container?

- Virtualisierung auf Betriebssystemebene
- Basiert auf mehreren Features des Linux Kernels
- Isoliert und verpackt Anwendungen

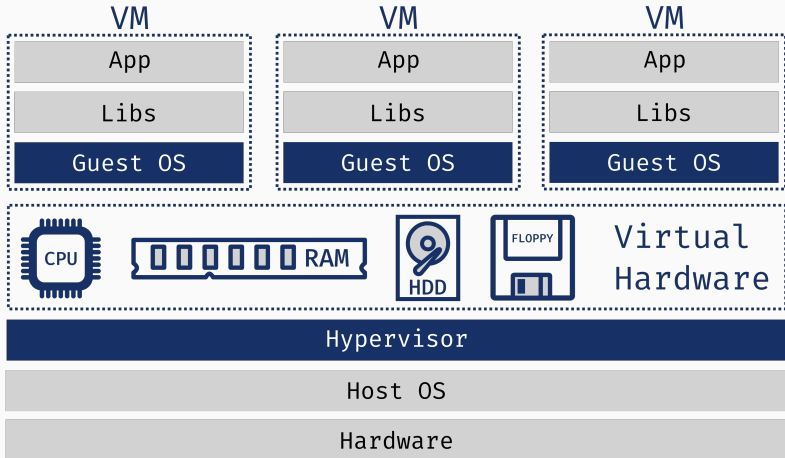
# Was sind Container?

- Virtualisierung auf Betriebssystemebene
- Basiert auf mehreren Features des Linux Kernels
- Isoliert und verpackt Anwendungen

# Exkurs: Hardware-Virtualisierung

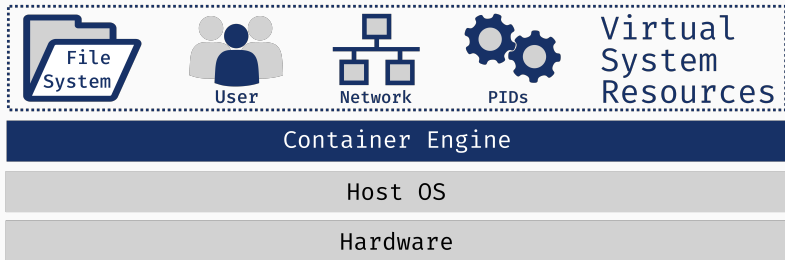


# Exkurs: Hardware-Virtualisierung

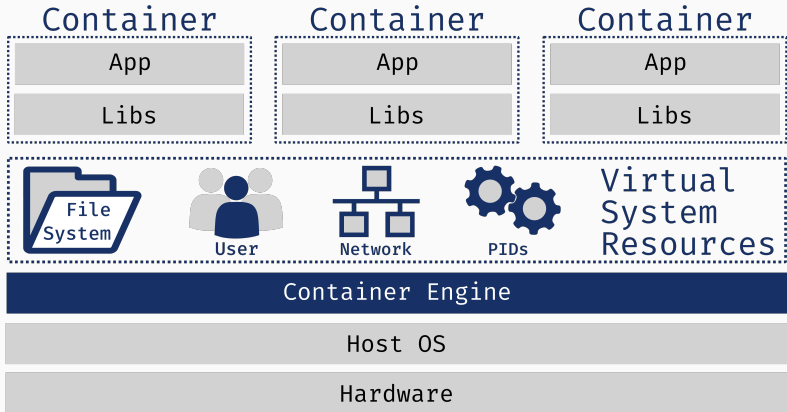




# Virtualisierung auf Betriebssystemebene



# Virtualisierung auf Betriebssystemebene



Demo

# Implementierung in Linux

---

# chroot

```
/
├── /bin
├── /dev
├── /etc
├── /home
│   └── /hans
├── /mnt
├── /proc
├── /root
├── /var
│   ├── /log
│   └── /lib
│       ├── /docker
│       └── /python
├── /tmp
└── /new-root
...

```

# chroot

```
/
├── /bin
├── /dev
├── /etc
├── /home
│   └── /hans
├── /mnt
├── /proc
├── /root
├── /var
│   ├── /log
│   ├── /lib
│   │   ├── /docker
│   │   └── /python
├── /tmp
└── /new-root
...

```

Demo

- Erlaubt es das root Verzeichnis / neu zu setzen
- Ermöglicht verschiedene Versionen eines Tools auf einem System zu installieren
- Sehr Hilfreich zum Debugging oder bei der Installation von Linux
- **Aber:** Noch keine Isolierung der Prozesse, User etc. von einander



- API des Linux Kernel um **virtuelle System Ressourcen** wie Netzwerk Interfaces, Mount points, UserIDs und weitere System Ressourcen zu erstellen

- API des Linux Kernel um **virtuelle System Ressourcen** wie Netzwerk Interfaces, Mount points, UserIDs und weitere System Ressourcen zu erstellen
- Diese Ressourcen können einzelnen Prozessen zugewiesen werden

- API des Linux Kernel um **virtuelle System Ressourcen** wie Netzwerk Interfaces, Mount points, UserIDs und weitere System Ressourcen zu erstellen
- Diese Ressourcen können einzelnen Prozessen zugewiesen werden
- Können auch für sich genommen verwendet werden.  
Beispiel: Auf seinem eigenen Rechner mit Network-Namespaces ein Netzwerk simulieren

- Management von CPU Zyklen, Arbeitsspeicher oder Netzwerk Bandbreite für Gruppen von Prozessen

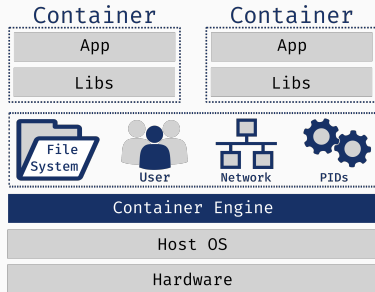
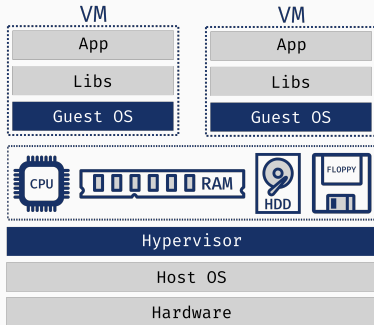
- Management von CPU Zyklen, Arbeitsspeicher oder Netzwerk Bandbreite für Gruppen von Prozessen
- Prozesse können in ihrem Ressourcenverbrauch eingeschränkt werden

- Management von CPU Zyklen, Arbeitsspeicher oder Netzwerk Bandbreite für Gruppen von Prozessen
- Prozesse können in ihrem Ressourcenverbrauch eingeschränkt werden
- Auch separat Nutzbar

## Vergleich mit VMs

---

# Vergleich mit VMs



**Unterschied:** Bei VMs wird im Kernel/Hypervisor Hardware virtualisiert und darauf laufen andere Kernels. Container teilen sich einen Kernel, der OS-Ressourcen virtualisiert.



- Voller virtueller Computer mit allen Features
- Egal welcher Kernel: Linux, BSD, Windows NT, x86, x64
- Starke Isolierung

- Effizienter: Weniger Ressourcenverbrauch und kein Boot-Vorgang
- Einfacher zu managen
- Auch einzelne Module verwendbar

# Praktische Anwendung

---

- **Konsolidierung** mehrerer Anwendungen ohne ineffiziente VMs

- **Konsolidierung** mehrerer Anwendungen ohne ineffiziente VMs
- Weg aus der **Dependency Hell**

- **Konsolidierung** mehrerer Anwendungen ohne ineffiziente VMs
- Weg aus der **Dependency Hell**
- Portable **Development Environments**. Verhindert "Works on My Machine" weil alle die gleiche Version haben

- **Konsolidierung** mehrerer Anwendungen ohne ineffiziente VMs
- Weg aus der **Dependency Hell**
- Portable **Development Environments**. Verhindert "Works on My Machine" weil alle die gleiche Version haben
- **Isolierung** unsicherer Prozesse von einander

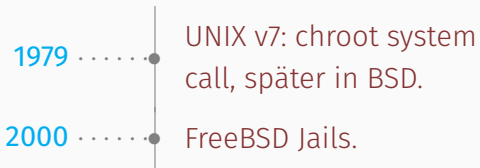
- **Konsolidierung** mehrerer Anwendungen ohne ineffiziente VMs
- Weg aus der **Dependency Hell**
- Portable **Development Environments**. Verhindert "Works on My Machine" weil alle die gleiche Version haben
- **Isolierung** unsicherer Prozesse von einander
- Ermöglicht weitreichende **Orchestrierung** auf großen Rechner-Clustern mit Failover und großer Skalierung durch Lösungen wie Kubernetes



Container sind keine neue Erfindung

---

1979 .....| UNIX v7: chroot system  
call, später in BSD.











2013

## Docker

- Entwickler Tooling
  - Daemon für Container Management
  - Standardisierung
  - Packaging in Images
  - Docker Hub
- Container werden für viele zugänglich und interessant für Entwickler.



Docker ist weit nicht die einzige Container-Software auf Linux:

- LXD
- Rocket
- systemd nspawn
- Flatpak
- Snappy



cioplenu.de  
@AnianZ

Wir suchen Entwickler und Sysadmins!

Vielen Dank! Fragen?