Data Engineering: Docker & Kubernetes

GitHub Repository

Technische Hochschule Rosenheim

https://github.com/

NickStudRepo/Docker-Kubernetes

Agenda

Technische Hochschule Rosenheim

Docker

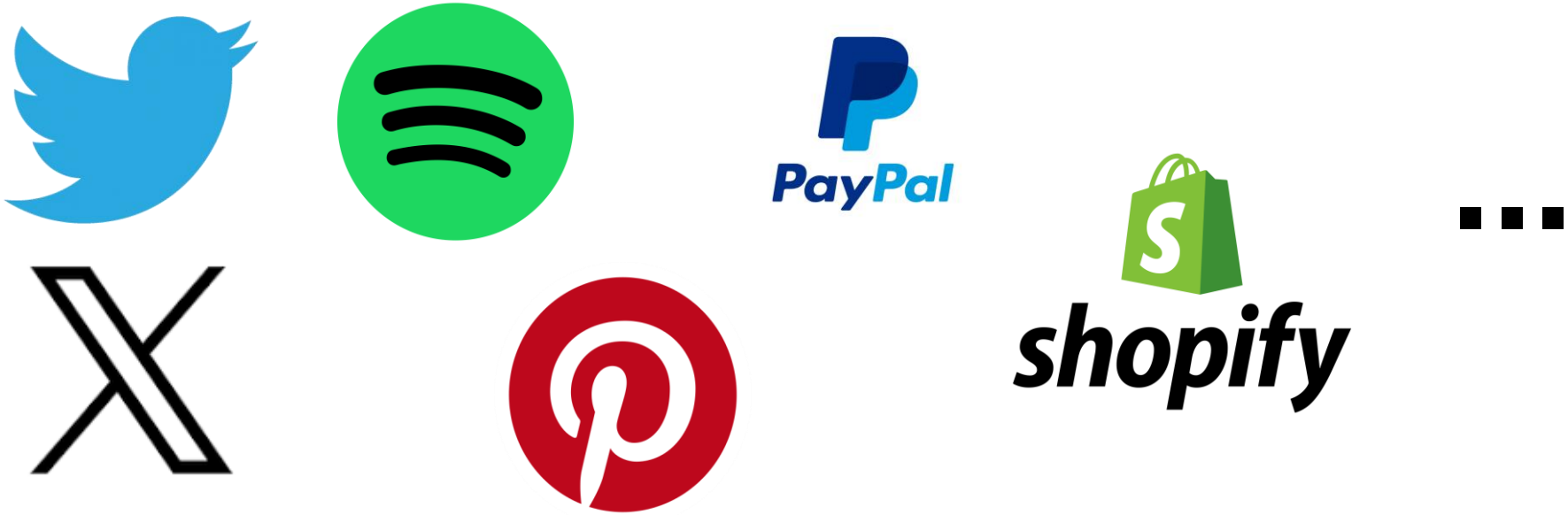
- Überblick
- Container vs. VM
- Architektur
- Dockerfile
- Docker Compose
- Dev Container
- Portainer (Dashboard)

Kubernetes

- Überblick
- Architektur
- Konfiguration
- Helm (Paketmanager)
- Virtual IP
- Persistent Storage
- Portainer (Dashboard)

Motivation

- Unternehmen die Docker/ Kubernetes verwenden



https://stackshare.io/docker

Docker

Docker Überblick

Technische Hochschule Rosenheim

- Software zur Isolierung von Anwendungen durch **Containervirtualisierung**

Ein Docker Container Image
ist ein **leichtgewichtiges**, **eigenständiges**, **ausführbares Softwarepaket**, das alles enthält, was zum Ausführen einer Anwendung erforderlich ist

- Docker ist eine Open-Source (Go)

Probleme die Docker löst

Technische Hochschule Rosenheim

Dependency Hell:

Versionskonflikte, **Komplexität der Abhängigkeitskette**, **Inkonsistenz**, **Aktualisierungen**, **Build Bereitstellung**

Docker:

- **Portabilität:** Docker Container verpackt Anwendung und Abhängigkeiten
- **Isolation:** Container ist in sich geschlossen
- **Konsistenz:** unabhängig von der Umgebung
- **Skalierbarkeit:** horizontale Skalierung durch mehrere Container
- **Schnelle Bereitstellung:** leichtgewichtig und notwendige/minimale Abhängigkeiten
- **Entwicklungsumgebungen**
- **Ressourcenoptimierung**

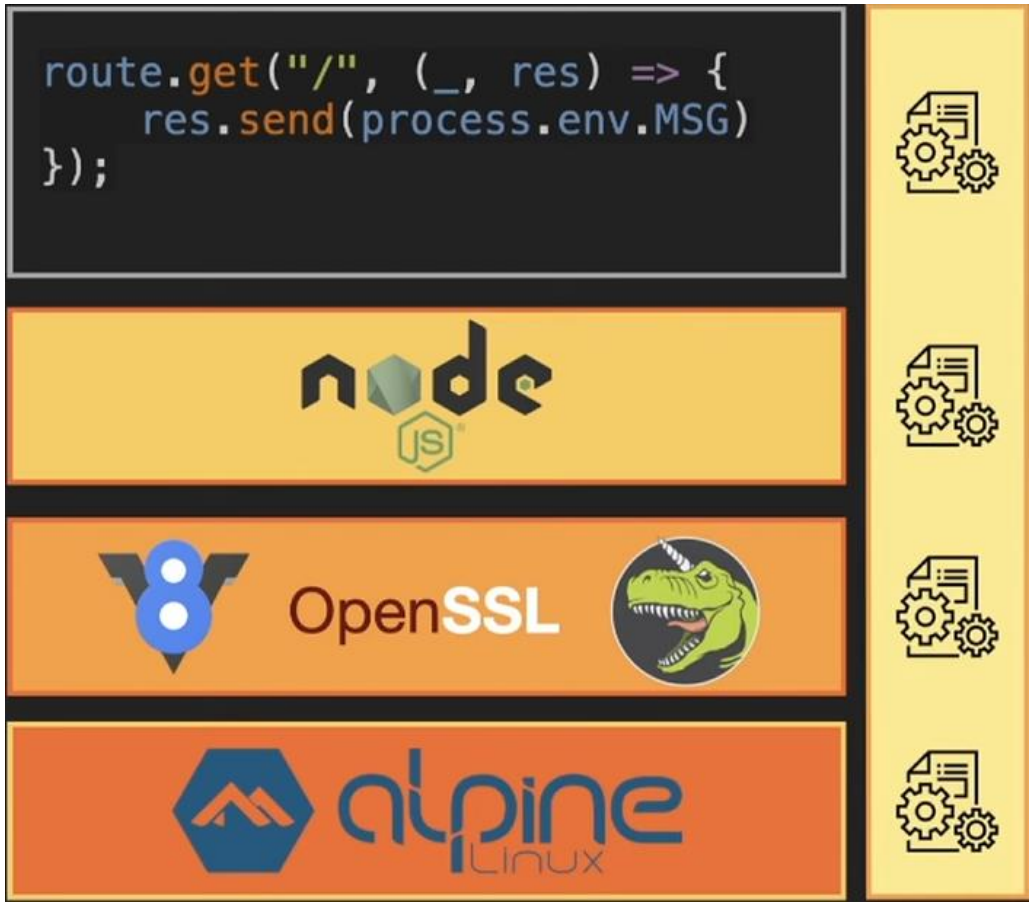
Container

Technische Hochschule Rosenheim

Ein Container ist ein **leichtgewichtiges**, **eigenständiges**, **ausführbares Softwarepaket**, das alles enthält, was zur Ausführung einer Anwendung erforderlich ist:

Code, **Laufzeit**, **Systemtools**, **Systembibliotheken** und **Einstellungen**.

Ist das eine VM?



Klassischer Ansatz (ohne VM oder Container)

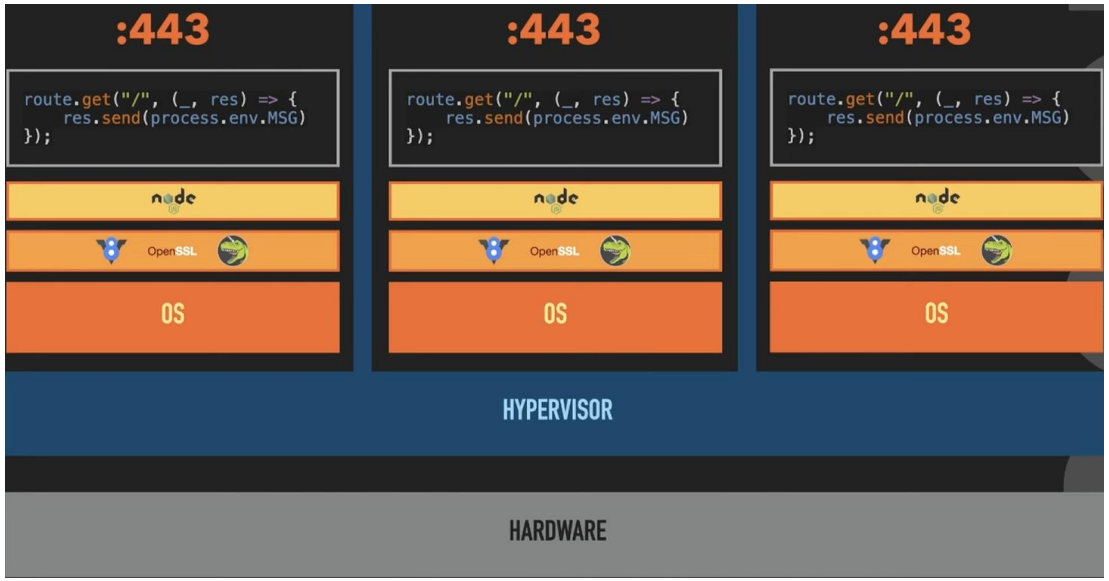
Technische Hochschule Rosenheim

- Ressourcennutzung
- Gegenseitige Beeinflussung!

Ressourcennutzung

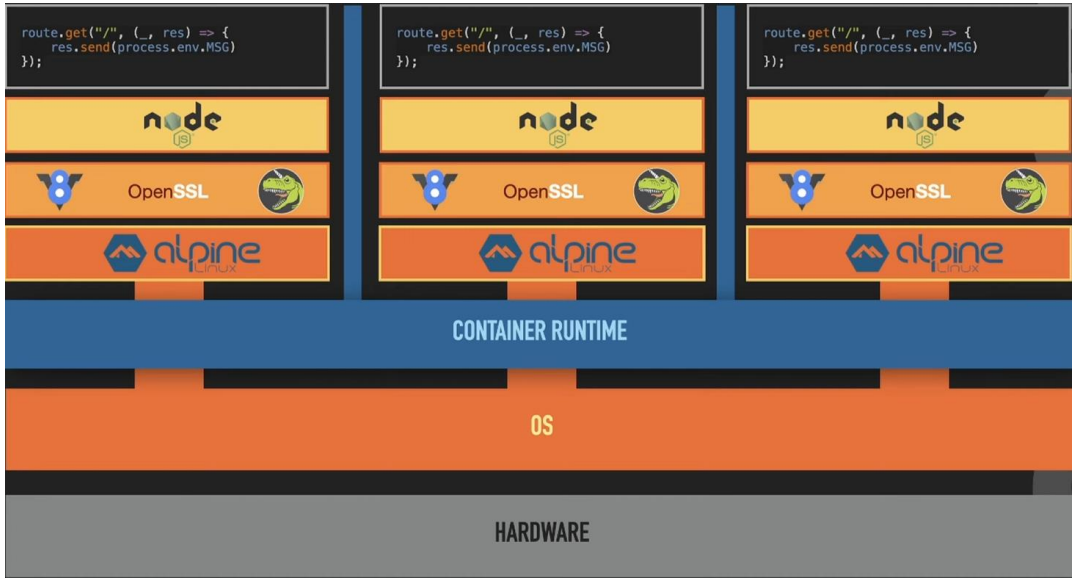
Technische Hochschule Rosenheim

VMs



- Hypervisor verwaltet VMs
- Jede VM braucht eigenes OS
- Viele Ressourcen + Zeit

Container



- Container Runtime isoliert Container
- Nur ein OS benötigt
- OS verwaltet Isolierung der Container
- Weniger Ressourcen + schneller

Fazit Container vs. VM

Technische Hochschule Rosenheim

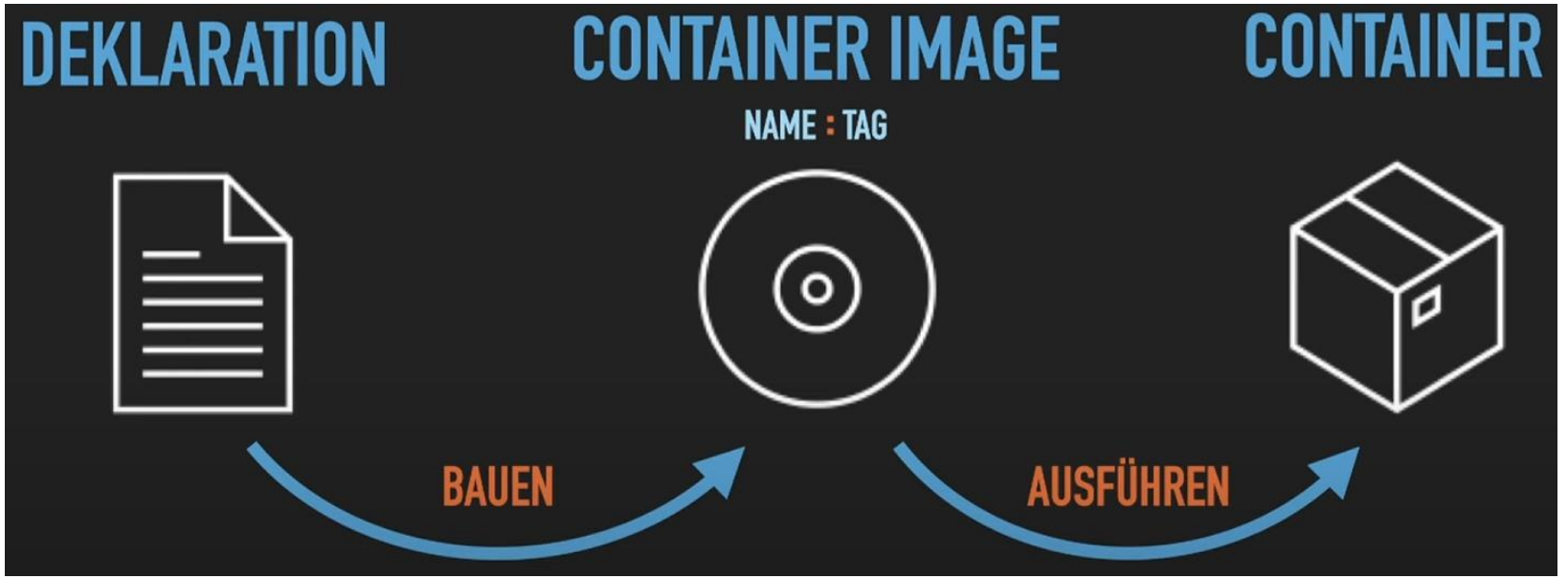
Gemeinsamkeiten:

- Es gibt eine Shell in VM und Container
  (muss nicht zwingend: statisch kompilierter Code)
- Eigener Prozessraum
- Eigenes Netzwerk Interface

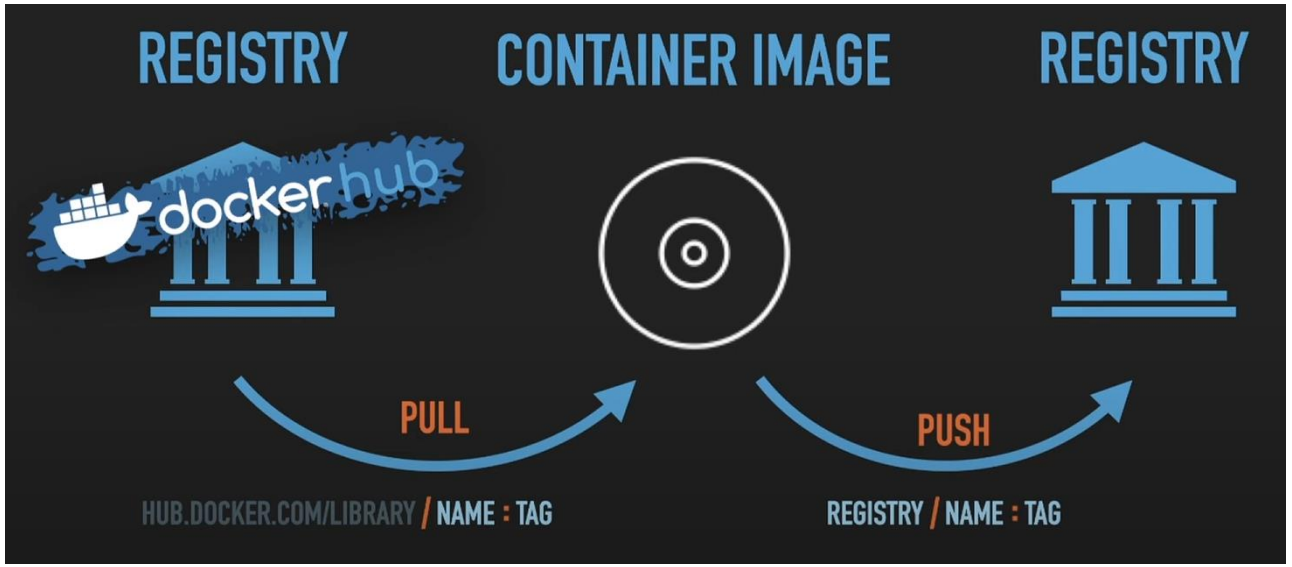
Unterschiede:

- Container benutzt den Host Kernel
- Linux Kernel -> Linux Container; Windows Kernel -> Windows Container
- Container ist normaler Prozess auf dem Host

Container: Lebenszyklus

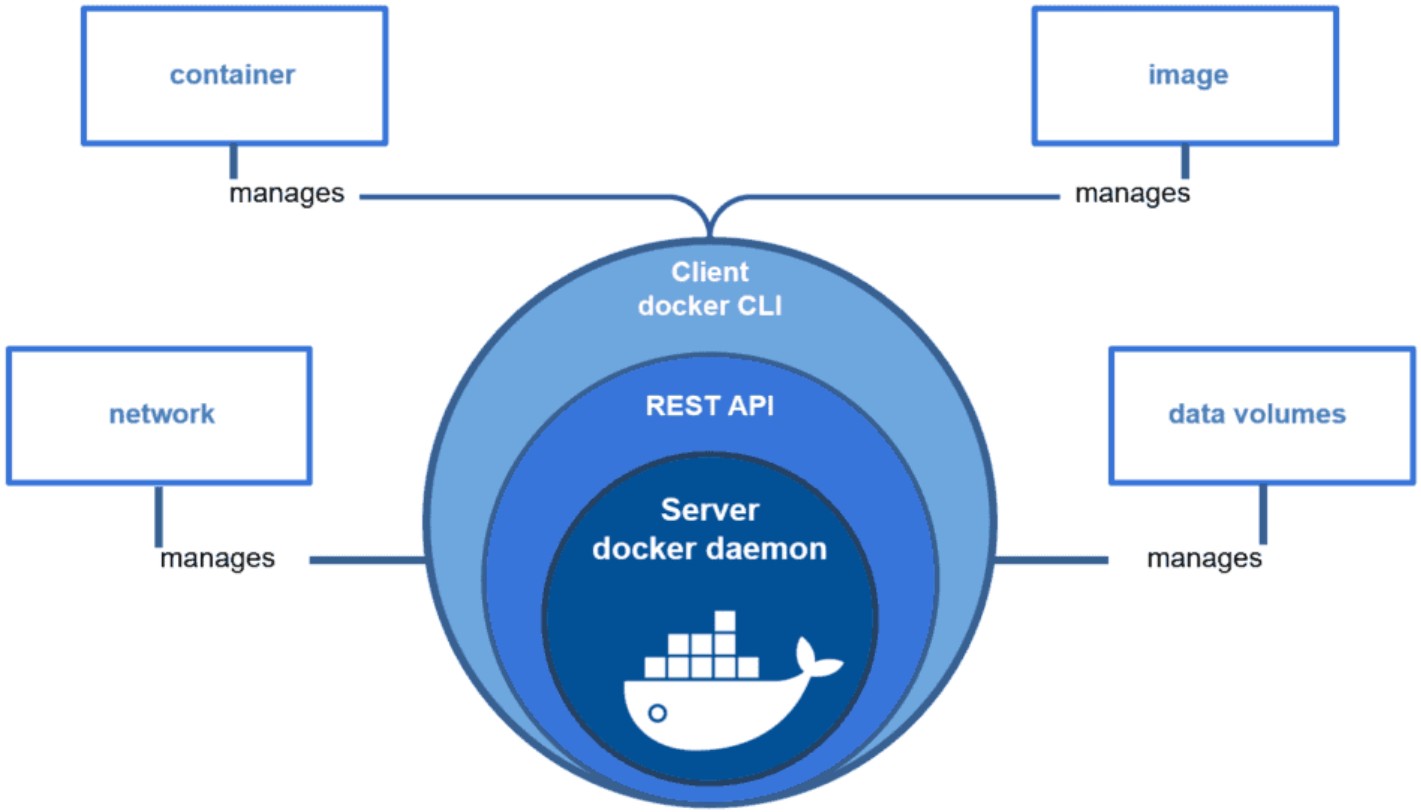


Container Teilen

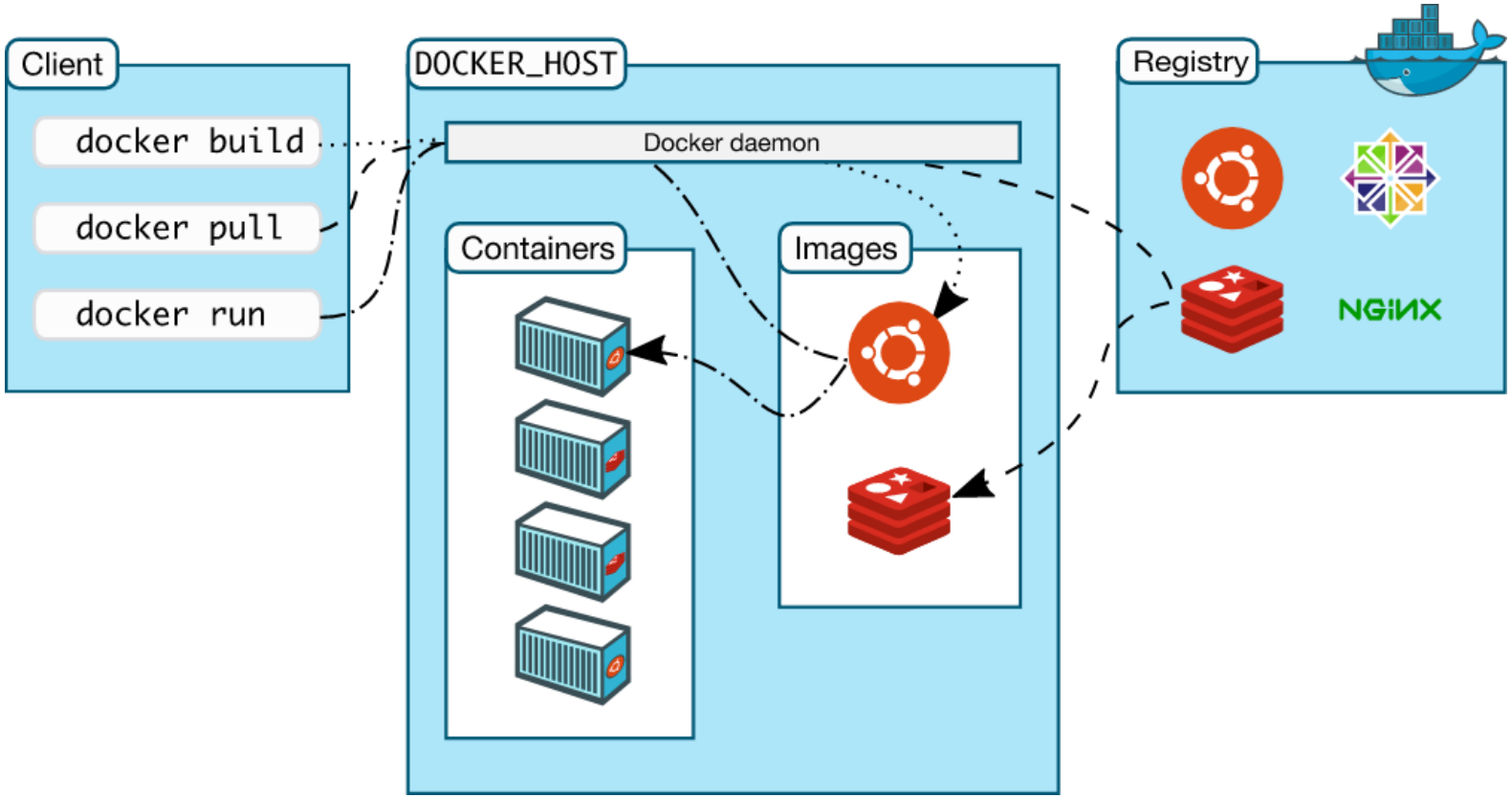


Öffentliche oder Private Registry

Docker Architektur



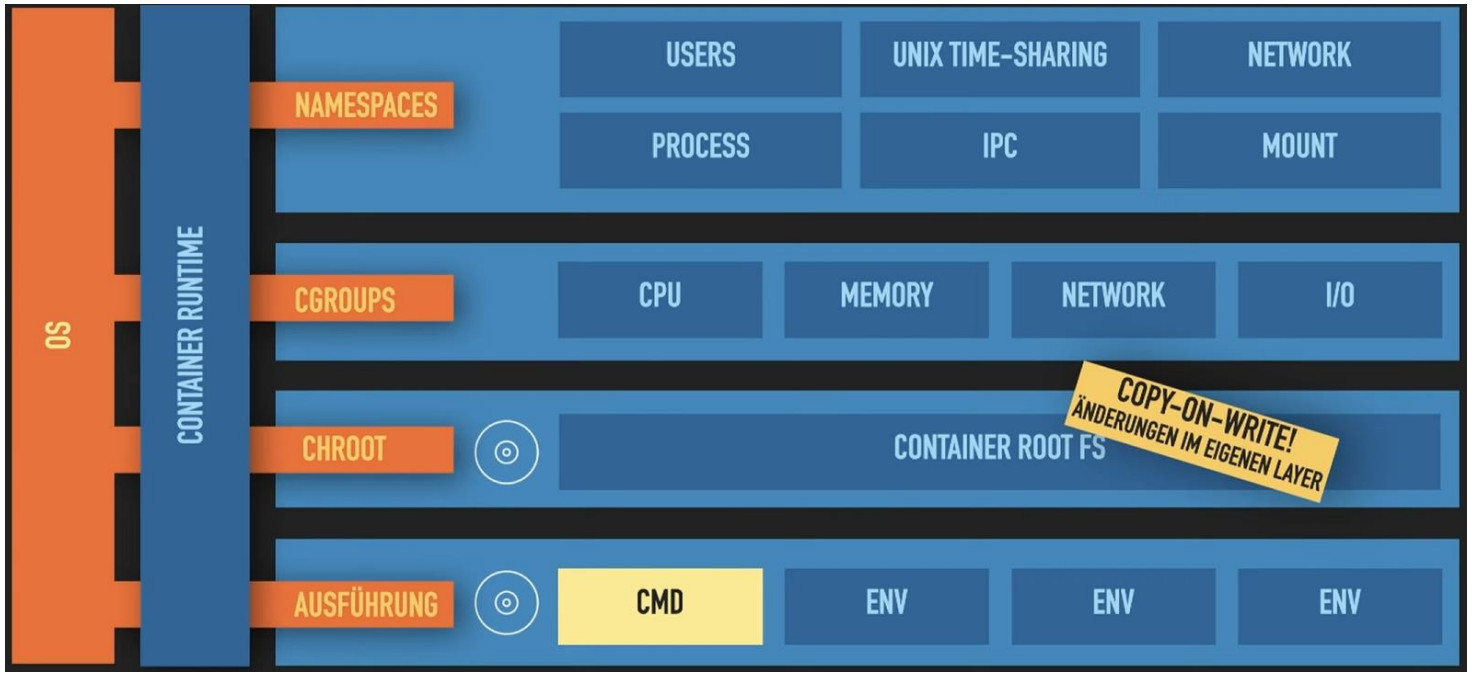
Docker Architektur



Docker Container Runtime

Technische Hochschule Rosenheim

- Host OS
  (Linux Kernel Befehle)
- Container Runtime verwendet Befehle von OS (nichts neues)
- Isolierung über Namespaces, cgroups und chroot



Runtime Komponenten

Technische Hochschule Rosenheim

- **Namespaces**: Was kann ein Container sehen?

  Abgrenzung von Prozessen + kein Zugriff aufs Hostsystem

- **Cgroups** (Control Groups): Was kann ein Container benutzen?

  Verwaltet Ressourcenzugriffe

- **Chroot** (Change Root): Ändert Root Directory für den aktuell laufenden Prozess

  Setzt Root Dir auf Root Dir des Containers, Zugriff nur auf Dateien innerhalb des Root Dirs

  → **Isolation von Containern und Prozessen**

- Host Windows: hat namespaces, cgroups nachgebaut, + Hyper-V (VMs im Hintergrund)

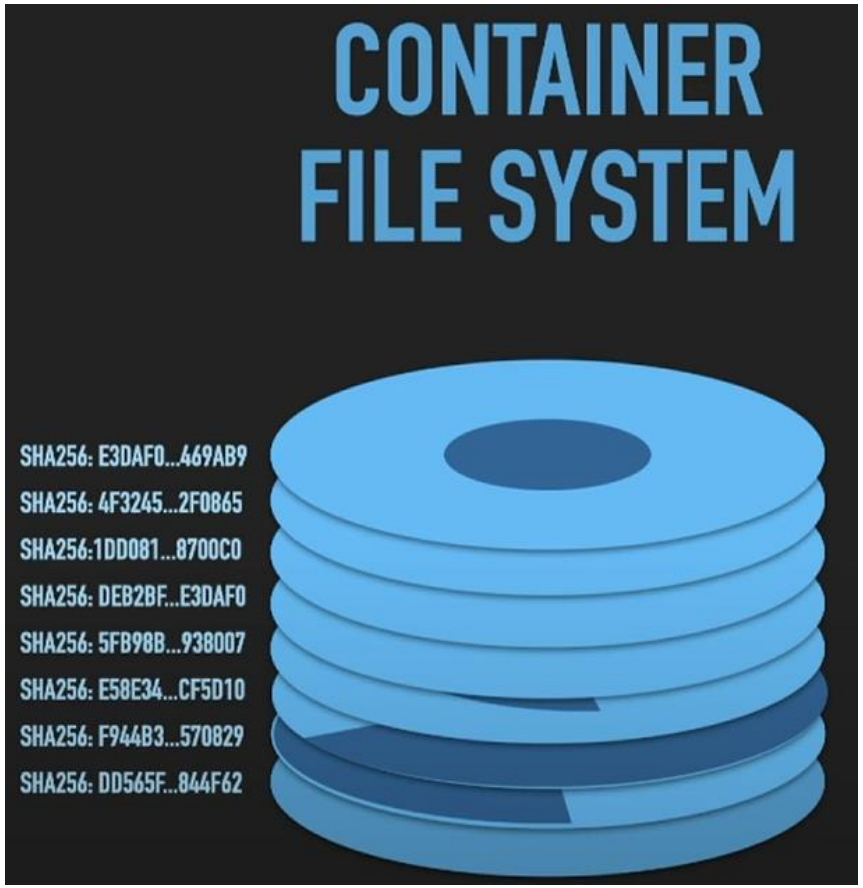
https://learn.microsoft.com/en-us/archive/msdn-magazine/2017/april/containers-bringing-docker-to-windows-developers-with-windows-server-containers

Deklaration: Dockerfile

Technische Hochschule Rosenheim

Dockerfile:

```
# Das Basisimage für unseren Container / oder FROM scratch
FROM node:18-alpine

# Anlegen und Wechsel in das Verzeichnis /app im Container
WORKDIR /app

# Kopieren der Projekt-Metadaten in das Verzeichnis /app..
COPY package.json .
# .. und des lokalen Verzeichnisses ./src in das Verzeichnis /app im Container
COPY src ./

# Installation der Projektabhängigkeiten mit dem Node Package Manager (npm)
RUN npm install

# Umgebungsvariablen für die Applikation setzen
ENV PORT=3000
ENV MSG="Hello World"

# Explizite Dokumentation: Die Applikation lauscht auf Port :3000
EXPOSE 3000

# Ausführen der Applikation als letzter Befehl
CMD ["node", "index.js"]
```

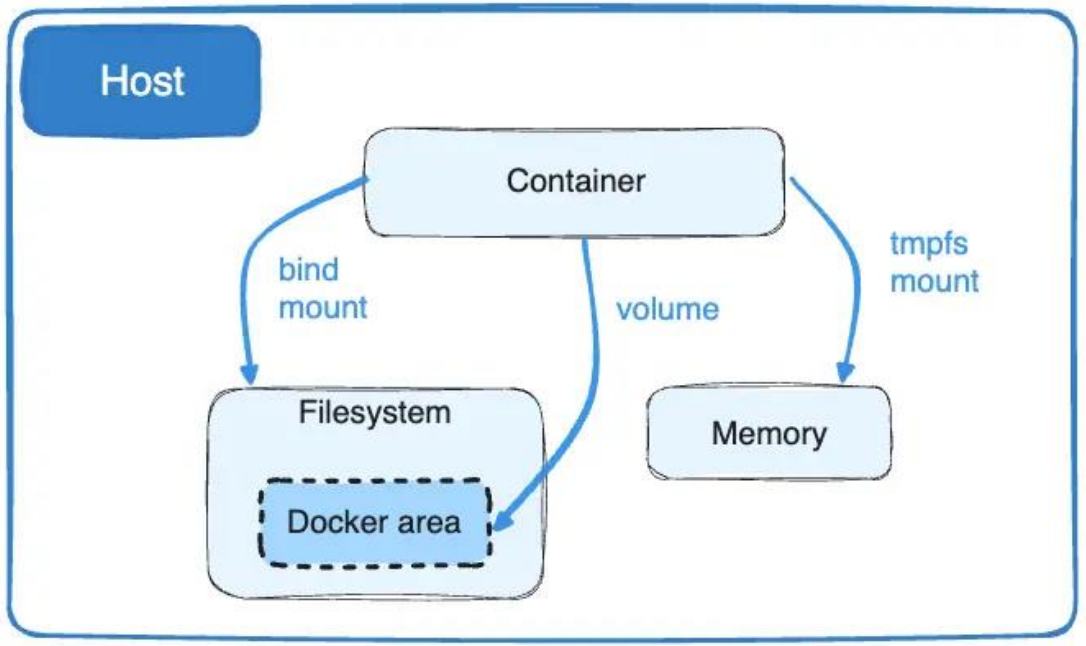


Demo: Dockerfile

Volume

Technische Hochschule Rosenheim

Volumes werden genutzt, da Container standard-mäßig nicht persistent sind



Docker Compose

Technische Hochschule Rosenheim

- Konfigurationsdatei in YAML-Format
- Deklarativer Ansatz
- Kann in Versionierungssystemen verwaltet werden
- Nützlich für System aus mehreren Containern

Demo: Docker Compose

Anwendungsfälle

Technische Hochschule Rosenheim

- Entwicklungs- und Testumgebungen
- Continuous Integration/Continuous Deployment (CI/CD)
- Microservices-Architektur
- Legacy-Anwendungen

Dev Container

Technische Hochschule Rosenheim

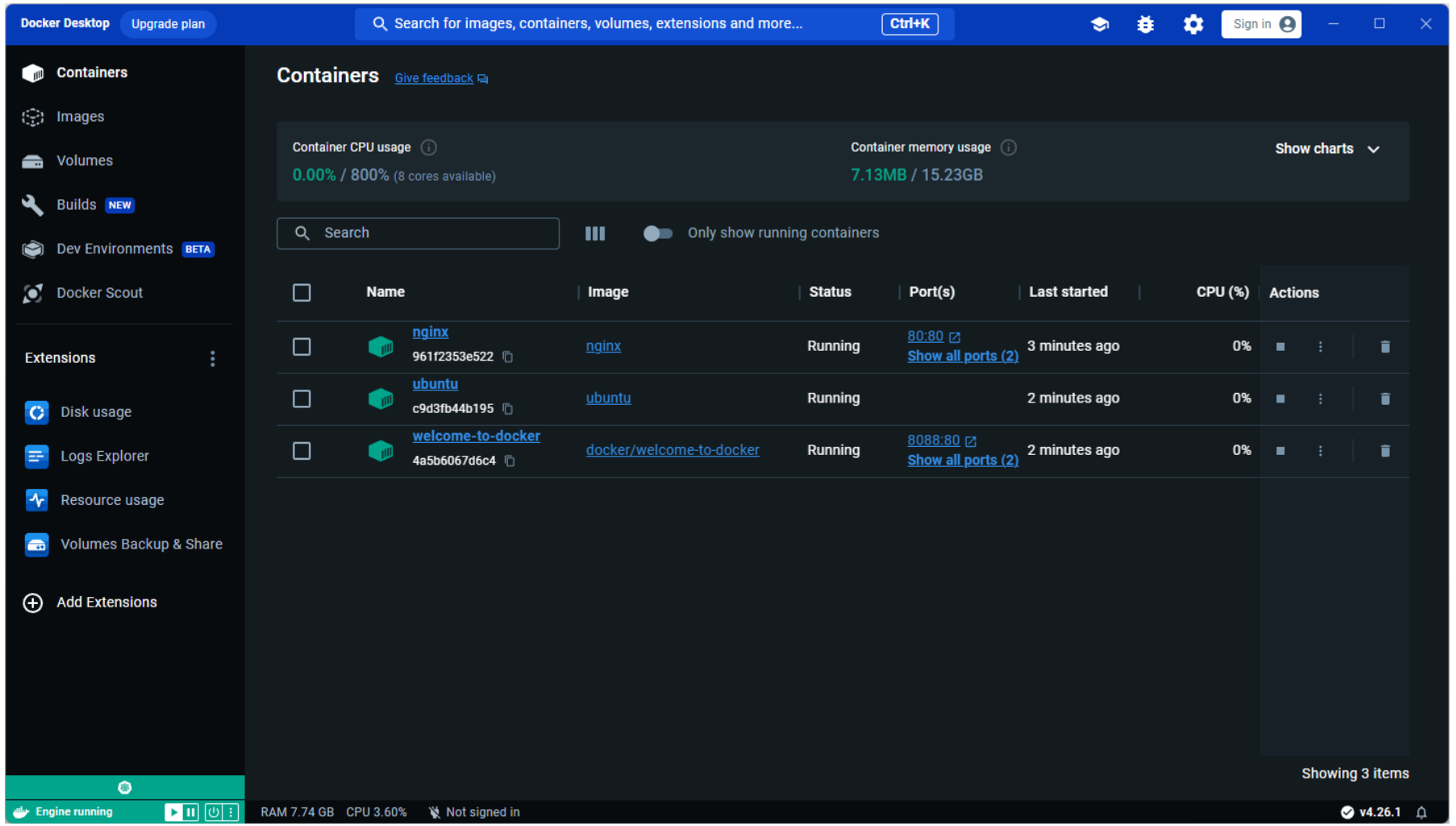
Dev Container werden genutzt, um das Problem „Es funktioniert auf meinem Rechner“ zu lösen

- Gleiche Umgebung für alle Entwickler

Demo: Dev Container

Dashboard: Docker Desktop

Technische Hochschule Rosenheim

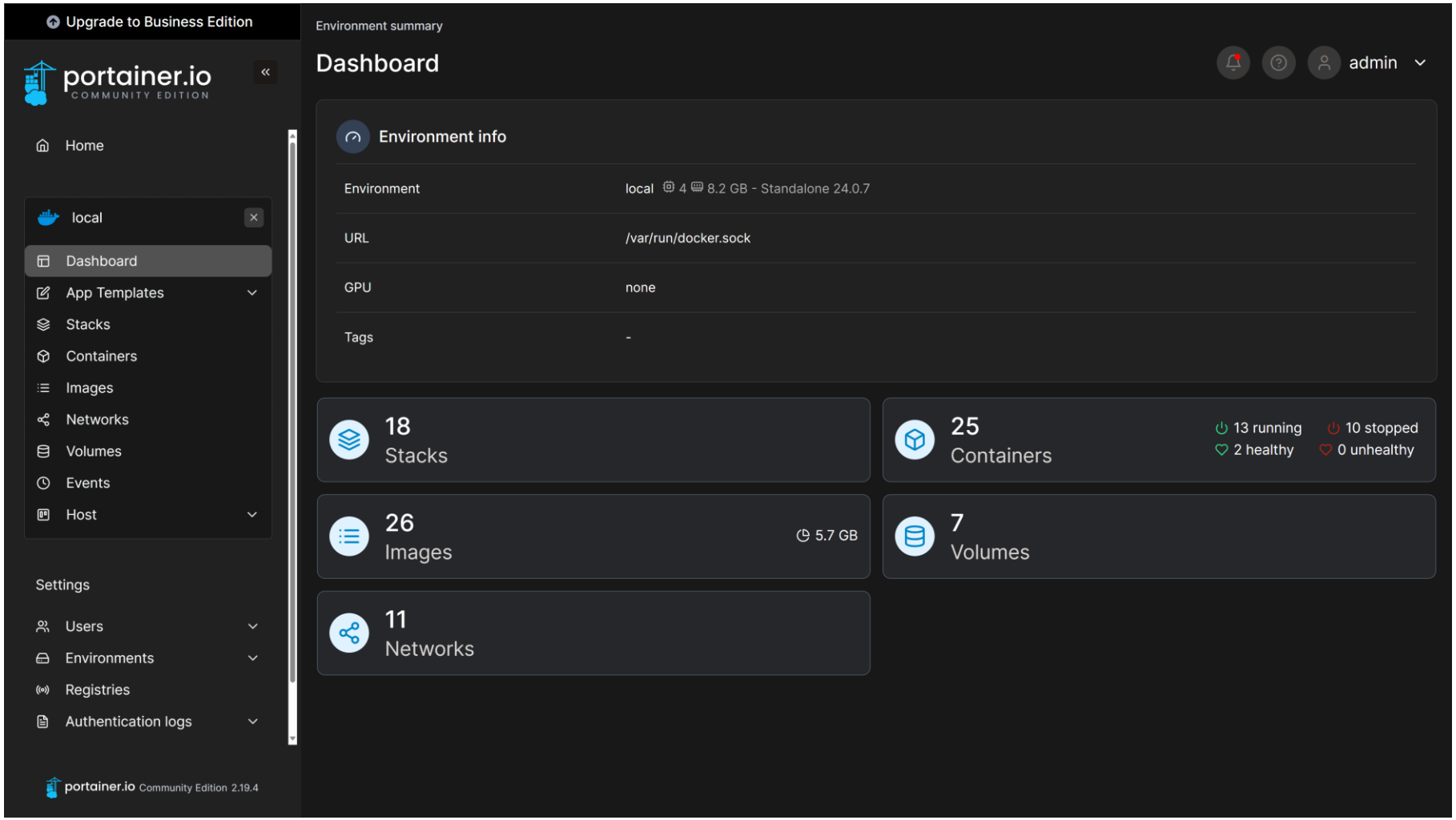


Dashboard: Portainer

- Einfache Container-Verwaltung
- Headless
- Unterstützt externe Docker-Instanzen
- GitOps

Demo: Portainer

Technische Hochschule Rosenheim



Alternative: Podman

Technische Hochschule Rosenheim

- Daemon-freie Architektur
- Vollständig kompatibel zu Docker-CLI
- Rootless Betrieb
- Verbesserte Sicherheit

Alternative: LXC/LXD

Technische Hochschule Rosenheim

- Linux Containers (LXC) und erweiterter Daemon (LXD)
- Näher am traditionellen VM-Modell
- Systemcontainer statt Anwendungscontainer
- Anpassbares Sicherheitsmodell

Zusammenfassung

Technische Hochschule Rosenheim

- Effiziente Containerisierung
- Vergleich mit VMs
- Dockerfile und Docker Compose
- Entwicklungsumgebung mit Dev Container
- Dashboards zur vereinfachten Verwaltung

Übung Docker

Technische Hochschule Rosenheim

Im Rahmen dieser Übung wird Docker inklusive Dockerfile und Docker Compose genutzt, um ein System aus Frontend, Backend und Datenbank zu containerisieren.

https://github.com/NickStudRepo/Docker-Kubernetes

Übung Dev Container

Technische Hochschule Rosenheim

Im Rahmen dieser Übung wird ein Dev Container für NodeJS in Visual Studio Code erstellt.

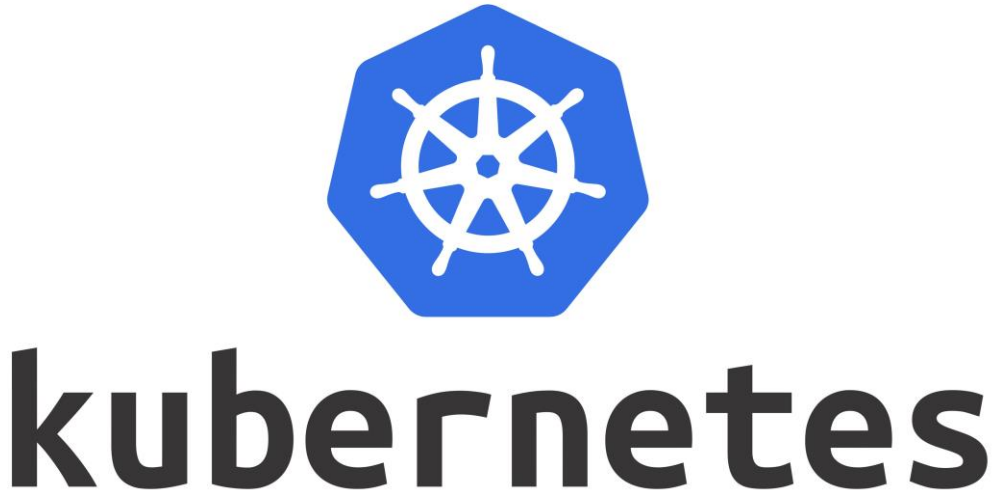
https://github.com/NickStudRepo/Docker-Kubernetes

Kubernetes

Kubernetes: Überblick

Kubernetes (K8s):

- Open Source (von Google entwickelt)
- Orchestrierung von Containern
- Deklarativer Ansatz (Soll zustand)



Kubernetes Features

Technische Hochschule Rosenheim

- **Container-Orchestrierung**: Bereitstellung, Aktualisierung, Ressourcenverwaltung und Skalierung
- **Geräte-** und **Netzwerkübergreifende Orchestrierung**
- Kubernetes oft in **Cloud** oder als **Hybrid** eingesetzt (AWS oder Microsoft Azure)

Features:

- **Load Balancing**: automatisches Verteilen von Lasten
- **Automatisches Skalieren** (hoch und runterfahren von Containern)
- **Selbstheilung** und **Ausfallsicherheit** durch Redundanzen
- **Rolling Updates** und **Rollbacks**

› Ressourcenmanagement

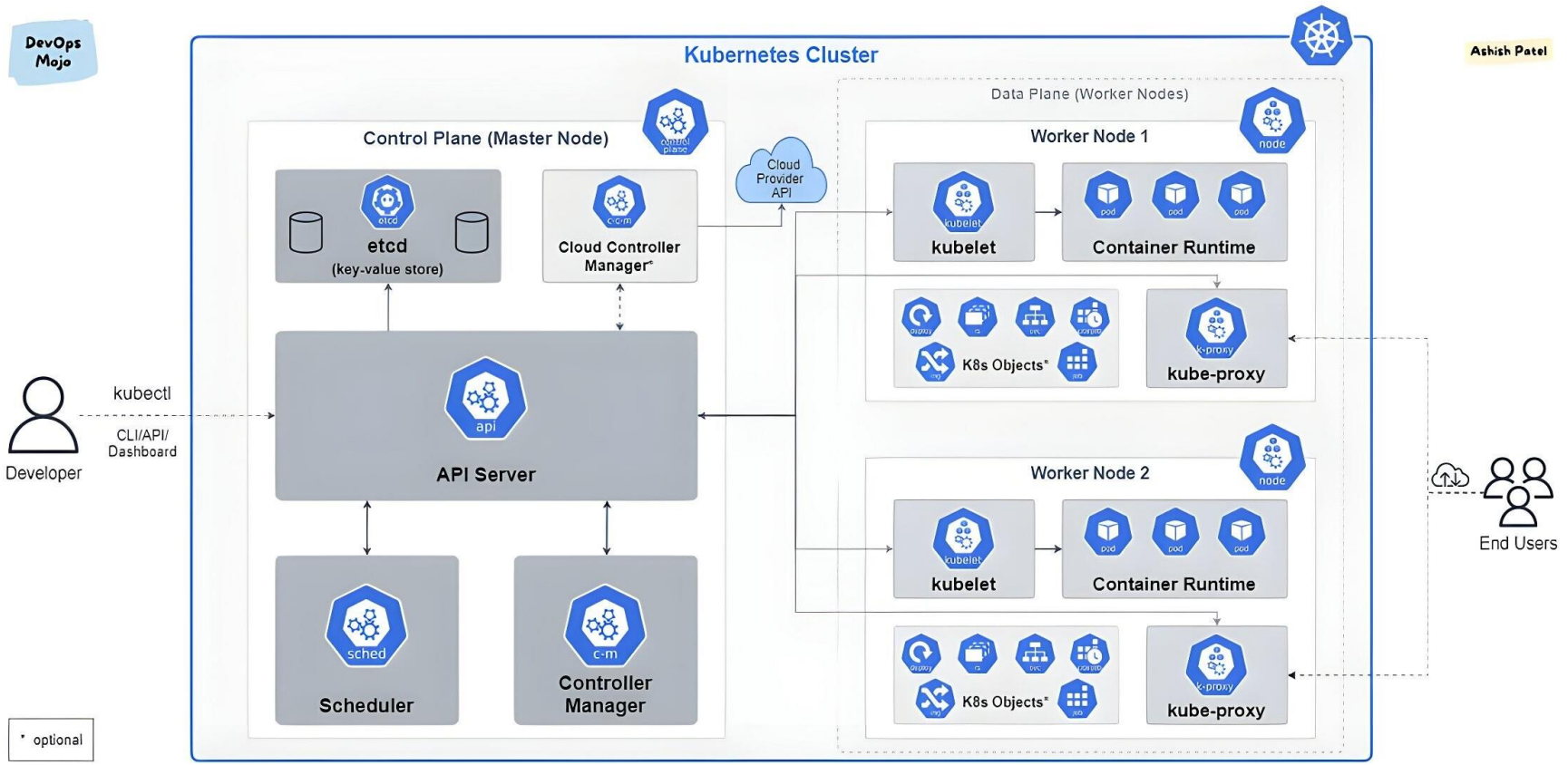
→ Keine Downtime!

Kubernetes Begriffe

Technische Hochschule Rosenheim

- **Kubernetes Cluster**: Gruppe von verbundenen Computern/ Nodes
- **Master Node** (Control Plane): zentraler Steuerungsknoten
- **Worker Node**: ausführenden Knoten, haben Container Runtime, enthalten Pods
- **Pod**: ist Wrapper um Container, enthält mind. 1 Container
  Grundlegende/ kleinste Einheit die verwaltet wird

Kubernetes: Architektur



Komponenten

Technische Hochschule Rosenheim

- **etcd**: verteilte Datenbank (Key, Value Store)
  Speichert Zustand des Clusters, Konfigurationen
- **API Server**: Rest Schnittstelle (Interaktion mit Devs + Auth), Node Kommunikation, Überwachung ob Nodes im gewünschten Zustand sind
- **Scheduler**: Verteilung von Pods auf Nodes (Lastenausgleich), Ressourcenmanagement auf Nodes
- **Controller Manager**: Überwacht Controller (Replica Controller, Namespace Controller), Reagiert im Fehlerfall
- **Cloud Controller Manager**: ermöglicht Integration mit Cloud (Auth)
- **kubelet**: Master Kommunikation, Pod Management (Start, Stopp), Pod Networking (innerhalb eines Pods)
- **kube-proxy**: Netzwerkmanagement (Pod- und Node-übergreifend), Load Balancing (Verteilung des Netzwerkverkehrs auf die Nodes), ermöglicht jedem Pod stabilen Endpunkt (Service)
- **K8s objects**: ConfigMap und Secrets: wer darf mit Pod kommunizieren
- **Useranfragen an kube-proxy**: welcher Node antwortet ist Requestabhängig + Load Balancing Algorithmus (Round Robin ist default)

Kubernetes Konfiguration

Technische Hochschule Rosenheim

- .yaml (Deklarativ)
- Deployment
- Service
- Secret
- ConfigMap

webapp.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp-deployment
  labels:
    app: webapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: node:21-bullseye
        ports:
        - containerPort: 3000
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
spec:
  type: NodePort
  selector:
    app: webapp
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
      nodePort: 30100

```

Kubernetes Konfiguration

Technische Hochschule Rosenheim

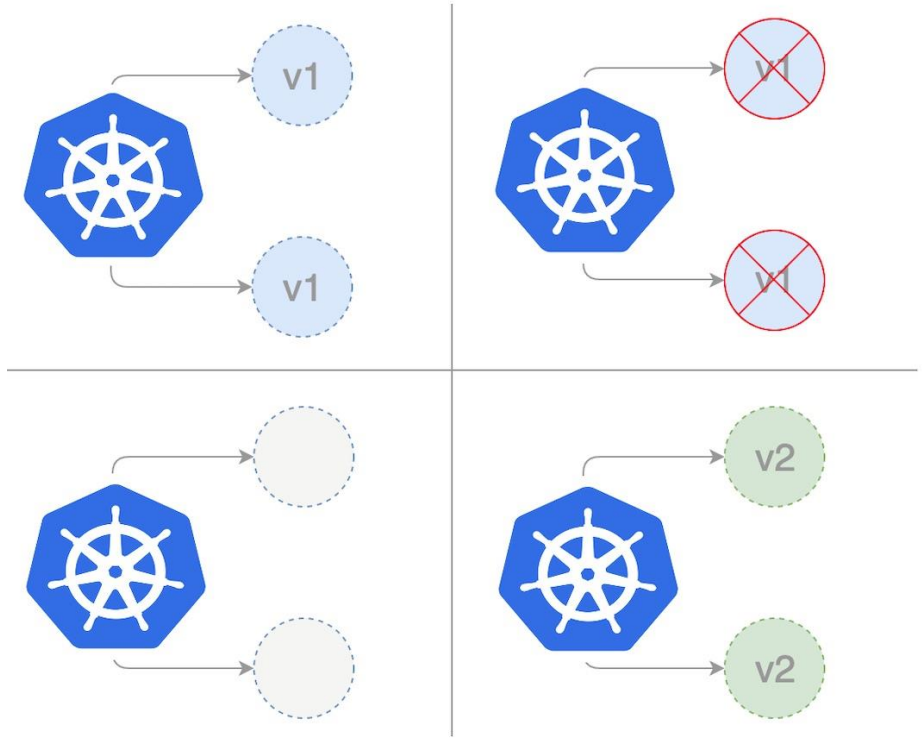
- **Deployment**: Definiert wie Container bereitgestellt werden + Skalierung (Replika)
- **Service**: Definiert stabilen Endpunkt für Gruppe von Pods (Netzwerkschnittstelle)
- **Secret**: Verwaltet sensible Informationen (Auth)
- **ConfigMap**: Key Value Paare an Konfigurationen

Demo: Kubernetes

Konfiguration eines einfachen Clusters

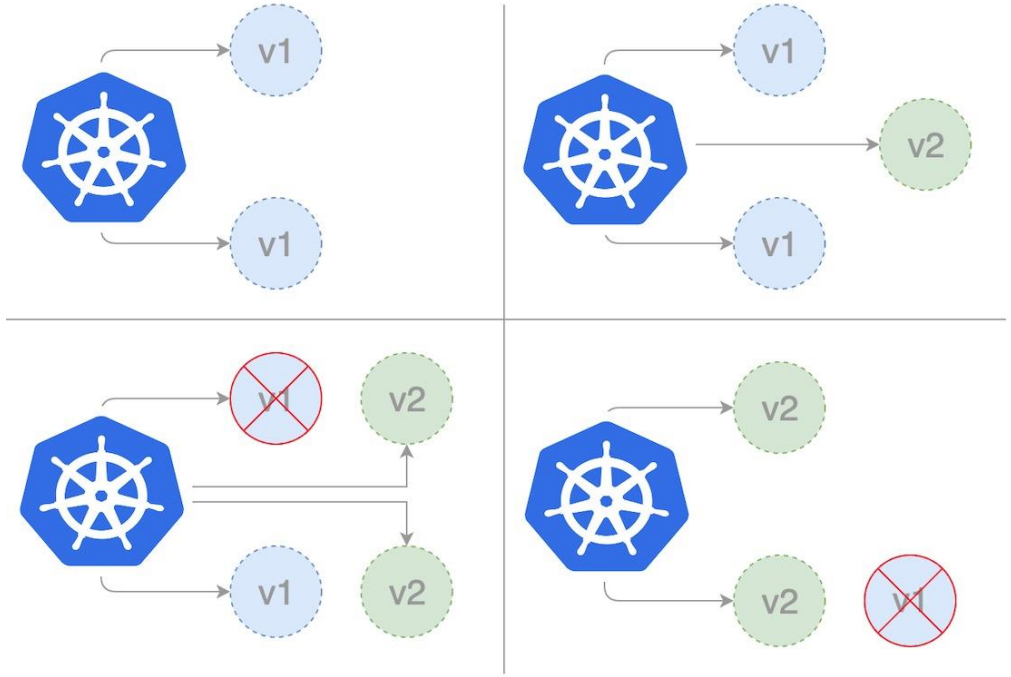
Deployment Strategie: Recreate

- Downtime während Update
- Manuelles Zurücksetzen bei Fehlern



Deployment Strategie: Rolling Update und Rollback

- Update ohne Downtime
- ReplicaSet
- Historie für Rollback



Helm

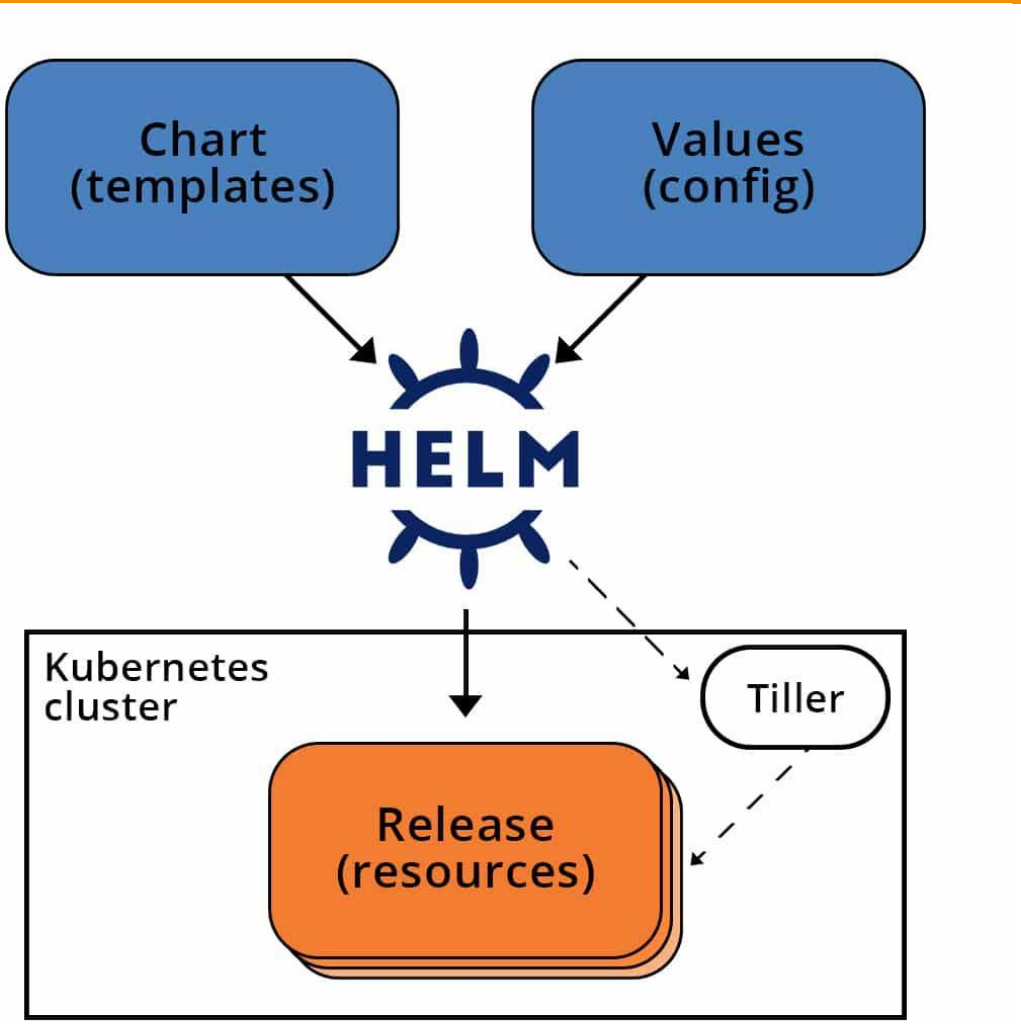
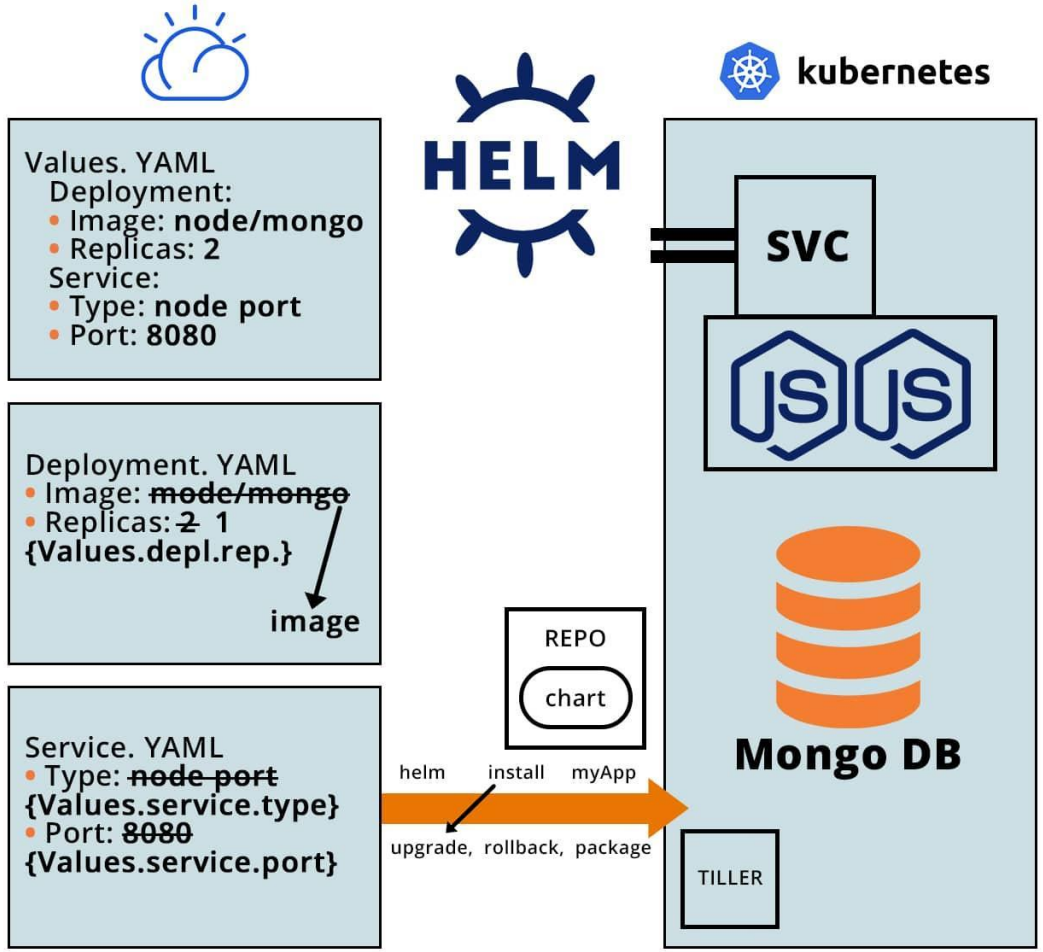


Chart
(templates)
Values
(config)
HELM
Kubernetes
cluster
Tiller
Release
(resources)

Helm

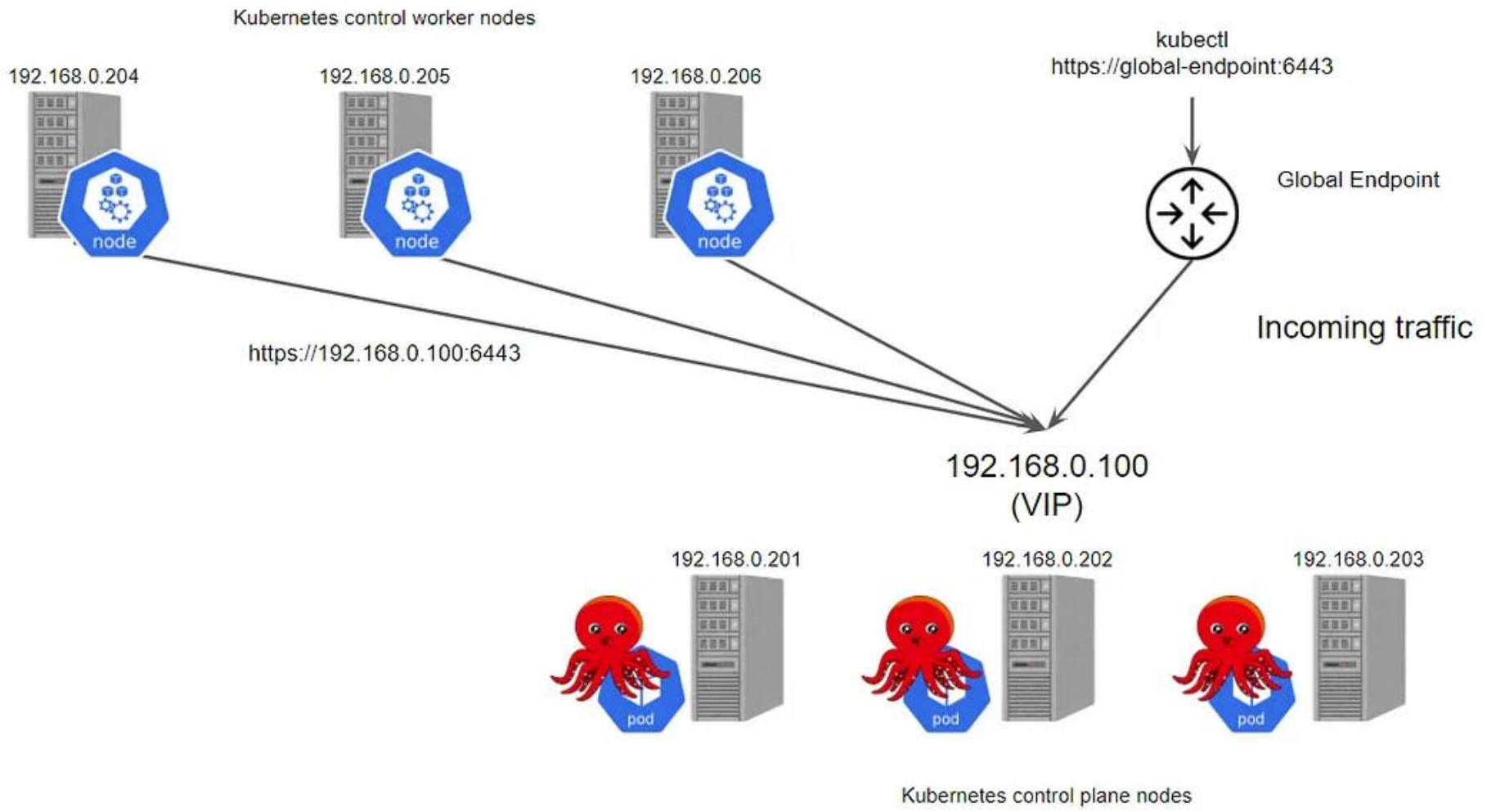

Values. YAML
Deployment:
• Image: node/mongo
• Replicas: 2
Service:
• Type: node port
• Port: 8080
Deployment. YAML
• Image: ~~node/mongo~~
• Replicas: ~~2~~ 1
{Values.depl.rep.}
image
Service. YAML
• Type: ~~node port~~
{Values.service.type}
• Port: ~~8080~~
{Values.service.port}
HELM
kubernetes
SVC
REPO
chart
helm install myApp
upgrade, rollback, package
Mongo DB
TILLER

Virtual IP: kube-vip

kube-vip stellt eine stabile und zuverlässige (externe) Virtual IP (VIP) für hochverfügbare Kubernetes Cluster bereit

- Failover und Redundanz
- Load Balancing

Virtual IP: kube-vip



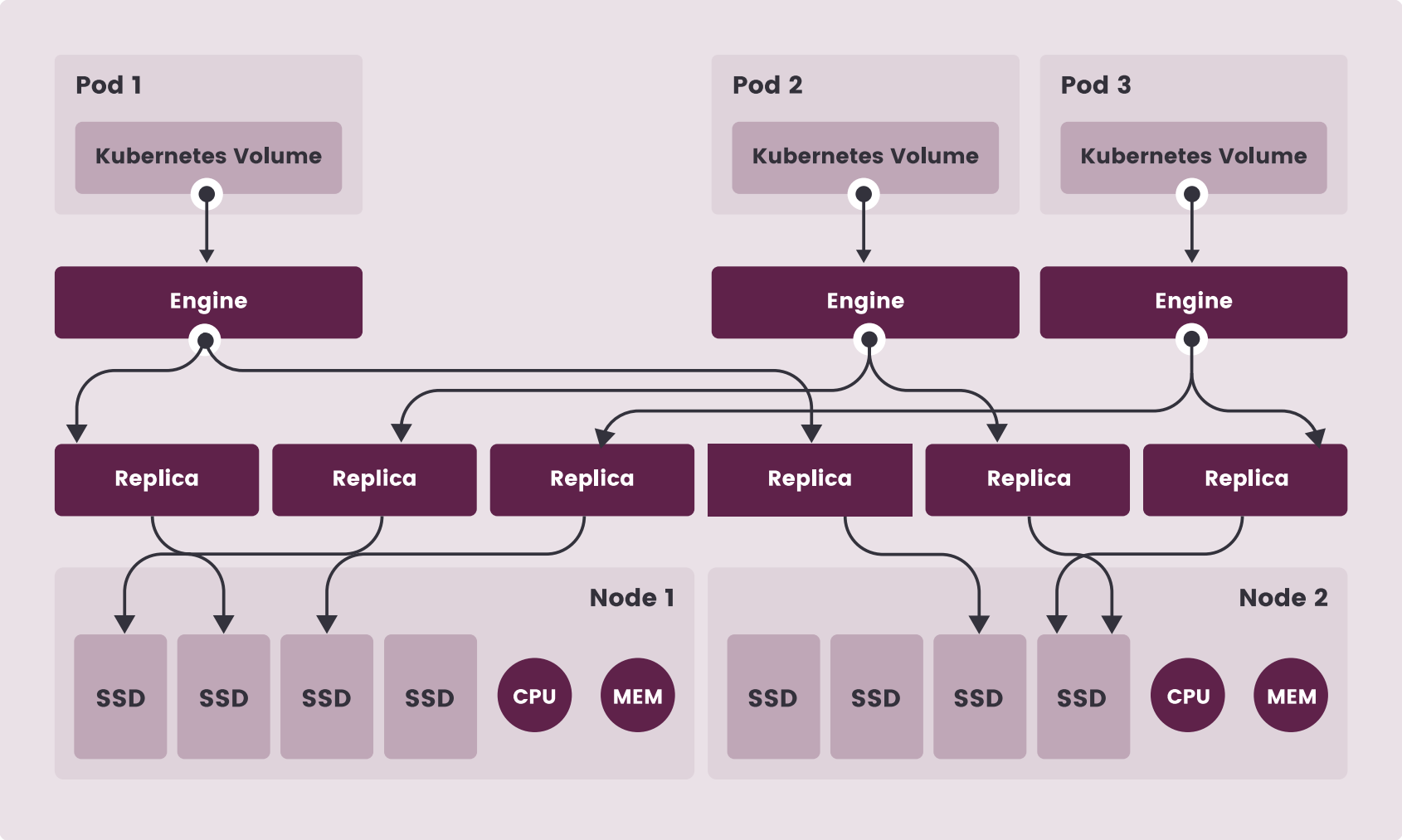
Persistent Storage: Longhorn

Longhorn stellt persistenten und hochverfügbaren

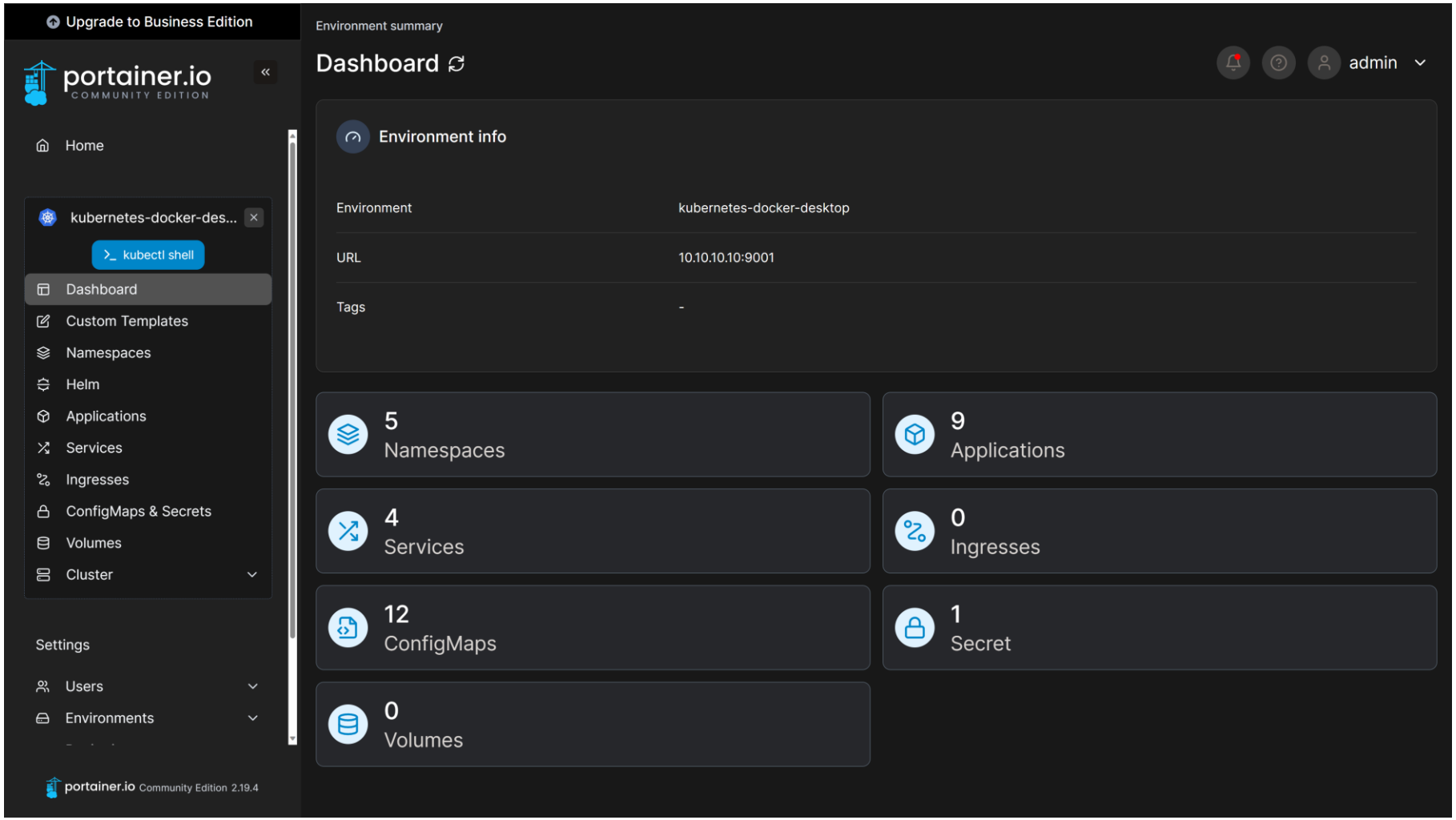
Speicher für containerisierte Anwendungen bereit

- Backup und Wiederherstellung
- Skalierbarkeit und Flexibilität

Persistent Storage: Longhorn



Demo: Portainer



Alternative: Docker Swarm

- Integriert in Docker
- Geringerer Overhead
- Einfachheit
- Unterstützt Deklarativ und Imperativ

Zusammenfassung

Technische Hochschule Rosenheim

- Kubernetes als Orchestrierungsplattform
- Modulare und robuste Architektur
- Helm als Paketmanager
- High Availability durch Persistent Storage und Virtual IP
- Dashboards zur vereinfachten Verwaltung

Übung: Kubernetes

Technische Hochschule Rosenheim

Im Rahmen dieser Übung werden mit kubectl und Helm

Anwendungen deployed.

https://github.com/NickStudRepo/Docker-Kubernetes

Fazit

- Synergie von Docker und Kubernetes
- Kubernetes eignet sich für komplexe Anwendungen
- Vereinfachung und Automatisierung
- Lernkurve und Community-Unterstützung

Reflexion

Technische Hochschule Rosenheim

Was nehmt ihr aus der heutigen Vorstellung zu Docker

und Kubernetes mit?

Bildquellen

Technische Hochschule Rosenheim

- https://k21academy.com/docker-kubernetes/docker-architecture-docker-engine-components-container-lifecycle/
- https://docs.docker.com/storage/volumes/
- https://www.portainer.io/hubfs/portainer-logo-black.svg
- https://rancher.com/docs/one-point-x/img/rancher-logo-nopadding.svg
- https://github.com/rancher/dashboard/blob/b7bae81b4161b65075e2a12cedadca2593e7afd9/docusaurus/docs/getting-started/screenshots/home.png
- https://inductor.medium.com/say-good-bye-to-haproxy-and-keepalived-with-kube-vip-on-your-ha-k8s-control-plane-bb7237eca9fc
- https://longhorn.io/img/logos/longhorn-horizontal-color.png
- https://kube-vip.io/images/kube-vip.png