

# Containerplattform

Lego für DevOps

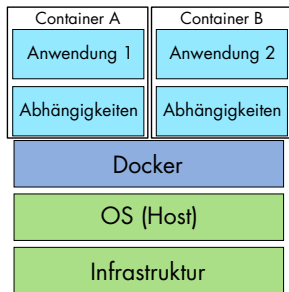


Dr. Jonas Trüstedt

19.10.2017

Virtualisierung in Containern:

- ▶ Minimalisierung der Abhängigkeiten
- ▶ Mit Host geteilter Kernel
  - Wenig Overhead
- ▶ 1 Container pro Funktion/Prozess (Microservices)
  - Anwendungen aus mehreren Containern



Beweggründe für den Einsatz von Containern:

- ▶ portabel
- ▶ schnell
- ▶ skalierbar

⇒ Container = Docker  
(Alternativen: rkt, Cloudfoundry, u.a.)



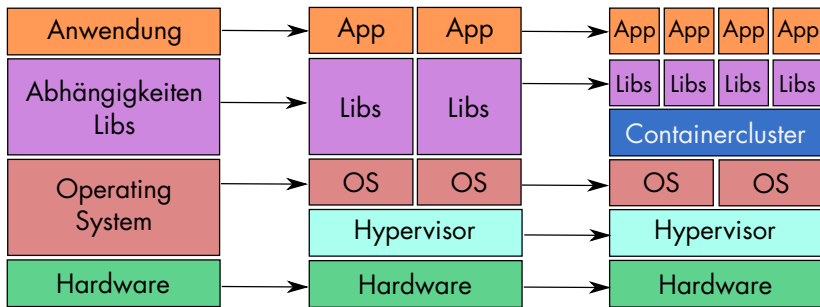
## Docker-Container:

- ▶ OS-unabhängig (sofern Linux)
- ▶ werden nicht gepflegt sondern neuerstellt (*Pets vs Cattle*)
- ▶ Schnell und speichereffizient

## Klassische virtuelle Maschinen (VM):

- ▶ Engverzahnte monolithische Software schwer zu containerisieren
- ▶ Leichter zu überwachen und zu härten

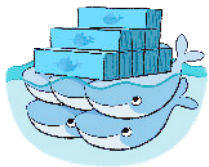
⇒ Kombination von VMs und Containern



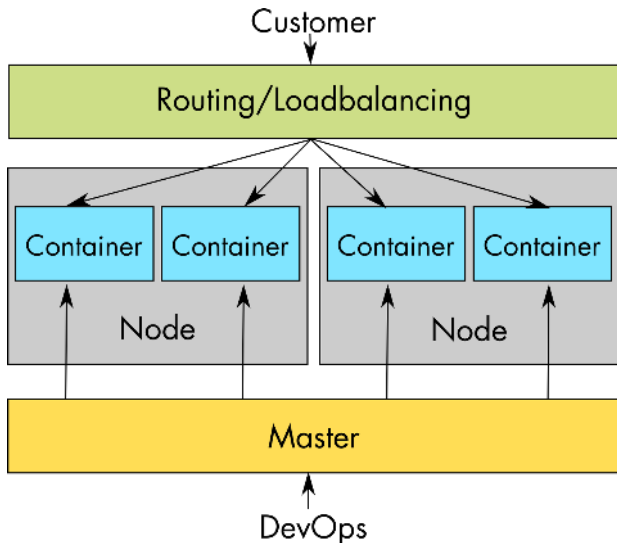
Docker alleine reicht oft nicht:

- ▶ Anzahl der Container
- ▶ Lastverteilung (Skalierung)
- ▶ Erstellung und Starten der Container erfolgt einzeln

→ Erweiterung der Infrastruktur auf einen Cluster mithilfe von z.B.:



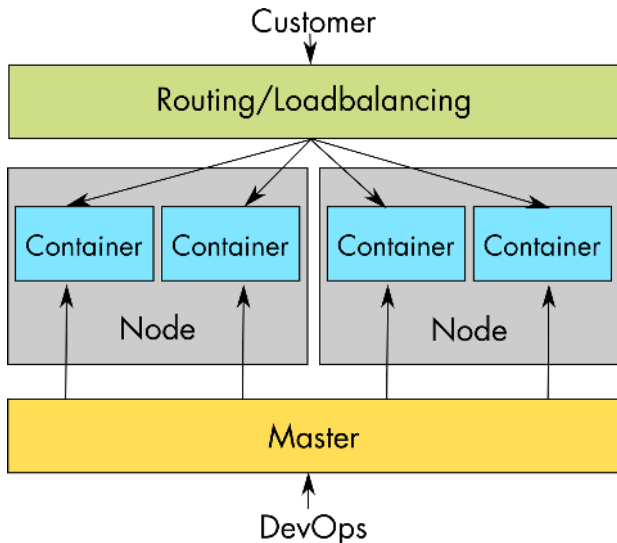
→ Features wie Scheduling, Loadbalancing, Healthchecks, Autoscaling



- ▶ Docker Swarm Mode:
  - ▶ Erweiterung auf mehrere Hosts von Docker selbst
- ▶ Rancher:
  - ▶ Unterstützung von Docker, Kubernetes, Mesos, Docker Swarm
  - ▶ Ab Version 2.0 nur mit Kubernetes
- ▶ Kubernetes:
  - ▶ ursprünglich entwickelt von Google
  - ▶ Fokus auf hohe Skalierbarkeit

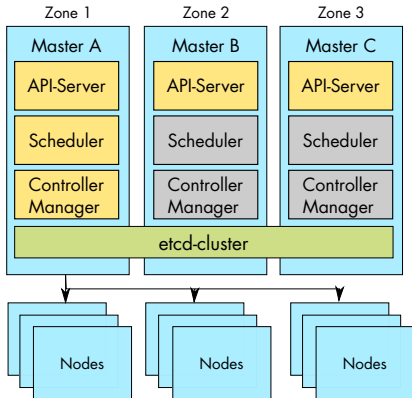
Kubernetes momentan als Favorit was Orchestrierung angeht:

- ▶ Rancher:  
Wechsel von Docker zu Kubernetes als Basiskomponente
- ▶ Red Hat:  
Openshift mit RHEL Atomic Host
- ▶ Canonical:  
Kubernetes mit Ubuntu Core
- ▶ SUSE:  
SUSE CaaS mit SLE MicroOS



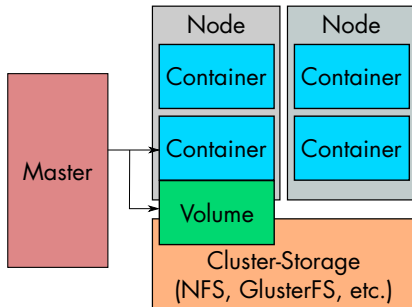
HA-Umsetzung durch:

- ▶ 3 Master (oder mehr)
- ▶ Synchronisation der Konfiguration
- ▶ Node-Ausfälle automatisch abfangen
- ▶ Container-Ausfälle durch Healthchecks abfangen



Daten-Container:

- ▶ Cluster-Storage für alle Nodes erreichbar
- ▶ Bereitstellung von Volumes für Container
- ▶ Wahl des Storage bei Planung der Infrastruktur berücksichtigen!

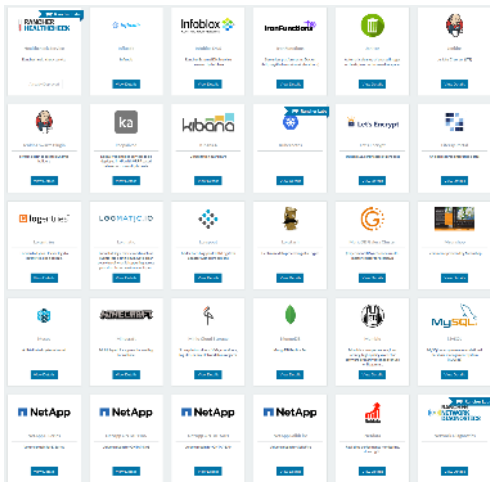


Zentrale Sammelstelle für Containerimages  
(z.B. Docker Registry, Quay, Docker Hub)

- ▶ Bereitstellung der Container-Images mit Versionskontrolle/Tags
- ▶ Intern gehostet oder öffentliche Varianten
- ▶ Docker Hub als zentrale öffentliche Registry
  - offizielle Images von Entwicklern
  - potentielles Sicherheitsrisiko
- ▶ Zertifizierte Registries wie z.B. Red Hat Container Catalog

## Vorkonfigurierte Vorlagen:

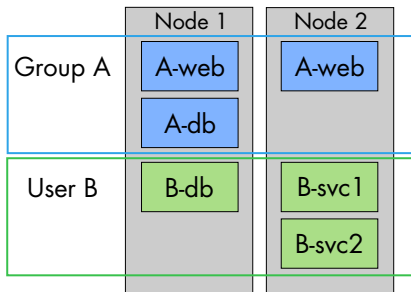
- ▶ Häufig verwendete Setups schneller zu starten
- ▶ Verfügbarkeit für Nicht-Admins (Selbstbedienungsportal)



Ausschnitt des Rancher-Katalogs

## Rollenbasierte Zugriffsrechte:

- ▶ Rechteverwaltung für Benutzer, Gruppen, Umgebungen, Projekte
- ▶ Grafische Oberfläche (WebGUI)
- ▶ Zuweisung von Ressourcen an Projekte



Mögliche Monitoring Ansätze:

- ▶ Verfügbarkeit der Dienste (Health-Checks)
- ▶ Metrics = RAM-, CPU- und Netzwerkauslastung  
→ Monitoring & Alerting-Tools z.B. mit Prometheus (+ Grafana) oder sysdig

Logging:

- ▶ Fluentd als Logging-Agent
- ▶ Meistens mit Elasticsearch und Kibana (ELK/EFK-Stack)

Automatisiertes Bauen, Testen und Ausrollen:

- ▶ Build über Webhooks
- ▶ Automatische Testläufe
- ▶ (Teil-)Automatisierung der Lifecycles
- ▶ Tools: z.B. Jenkins, Drone

