

Docker & Kubernetes

Tagesablauf

- 09:00 ... 10:30 Kurs
- 10:30 ... 10:45 Pause
- 10:45 ... 12:30 Kurs
- 12:30 ... 13:30 Mittagspause
- 13:30 ... 15:00 Kurs
- 15:00 ... 15:15 Pause
- 15:15 ... 17:00 Kurs

Docker & Kubernetes – Ziele?

- Kenntnisse & Erfahrungen des Dozenten
- Ihre Ziele und Wünsche

Docker & Kubernetes - Erkenntnisse

- Erkenntnisse aus dem Kurs
 - Wie funktionieren Docker & Kubernetes?
 - Einfluss von Containern auf Software und Deployment
 - Hands-on
 - Installation von Docker & K8s,
 - Container-Bau,
 - Speicherung,
 - Start, Logging, ...
 - Hintergrundwissen für die eigene Arbeit mit Docker & K8s

Docker & Kubernetes

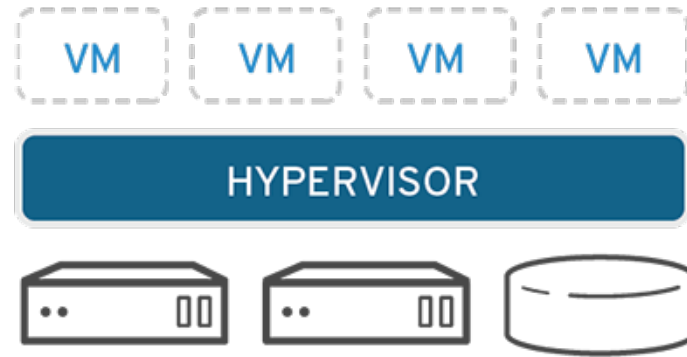
Teil 1

Teil 1 – Grundlagen

- Warum ist die Welt so, wie sie ist?
 - Möglichkeiten der Virtualisierung (Ausschnitt)
 - Cloud-Computing-Umgebungen
 - Software und Virtualisierung
- Docker – Grundlagen 1
- Kaffee-/Tee-Pause 10:30 bis 10:45

Möglichkeiten der Virtualisierung

- *Hypervisor* trennt phys. Ressourcen und Software



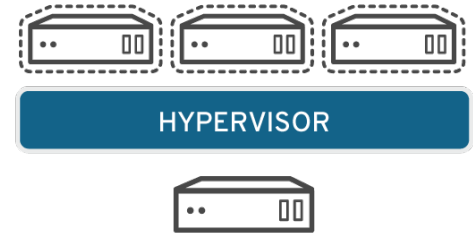
- *Virtuelle Maschine* (Guest) bildet Rechner ab
- Hypervisor regelt die Zuordnung der Ressourcen des Hosts zu Guests

Möglichkeiten der Virtualisierung

- Vorteile
 - Bessere Auslastung der Hardware
 - Einfache Datensicherung
 - Einfaches Skalieren
 - Einfaches Setup/Deployment
- Nachteile
 - Höherer Aufwand in Administration
 - Verlust an Leistung
 - Mehr Wissen notwendig
- Systematisierungen sind nicht eindeutig

Möglichkeiten der Virtualisierung

- 1 – Vollständige Virtualisierung
 - Jede virtuelle Maschine enthält ein vollständiges Betriebssystem
 - Hypervisor ist „transparent“
 - Vollständige Trennung
 - Teilweise Emulation der Hardware
 - Hypervisor „frisst“ bis zu 15% der Ressourcen



Möglichkeiten der Virtualisierung

- 2 – Paravirtualisierung
 - Keine vollständige Trennung der Guests
 - Guests kennen sich und arbeiten zusammen
 - Betriebssysteme müssen angepasst werden
 - Funktions-/API-Aufrufe in den Hypervisor
 - Weniger Ressourcenverbrauch durch Hypervisor

Möglichkeiten der Virtualisierung

- 3 – Virtualisierung auf Betriebssystem-Ebene
 - Container-Virtualisierung
 - Virtualisierung/Hypervisor ist Teil des Host-Betriebssystems
 - Guests sind unabhängig voneinander
 - Guests nutzen Teile des Host-Betriebssystems und teilen sich diese
 - Kaum Ressourcenverbrauch durch Hypervisor & kleine Guest Systeme (nur mit OS-Teilen)

Möglichkeiten der Virtualisierung

- ... – Kombination von Virtualisierungen
 - Virtuelle Server
 - Container-Virtualisierung
 - Häufig in Kubernetes-Clustern zu finden

Cloud-Computing-Umgebungen

- Vielfältige Bezeichnungen
- Gemeinsamkeiten:
 - Virtualisierte Server
 - Üblich sind kleine/kleinste Einheiten (CPU, Ram, Storage)
 - Cloud-Storage
 - Zusätzliche, managed Services
 - Preiswert nur bei Nutzung der spez. Eigenschaften

Software und Virtualisierung

- Container
 - je kleiner um so passender
 - eine Funktionalität = ein Prozess = ein Container
- Microservices
 - Schnelles und automatisches Skalieren
 - Kapselung und funktionale Trennung
- CI/CD
 - Unterbrechungsfreie Updates
 - Schnelle Reaktion auf Probleme und Kundenwünsche
- 12 Factors ... <https://12factor.net/de/>

Die zwölf Faktoren

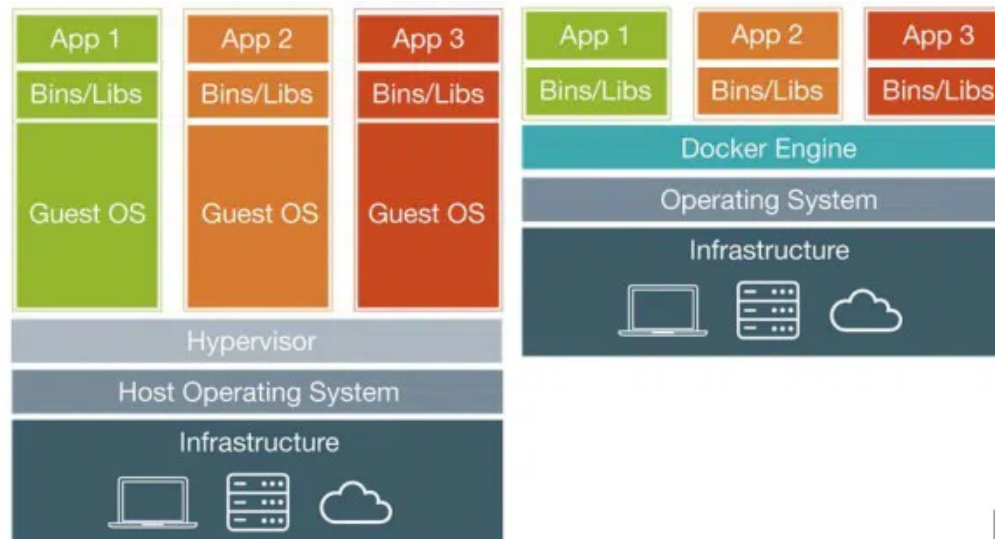
- I. Codebase
 - Eine im Versionsmanagementsystem verwaltete Codebase, viele Deployments
- II. Abhängigkeiten
 - Abhängigkeiten explizit deklarieren und isolieren
- III. Konfiguration
 - Die Konfiguration in Umgebungsvariablen ablegen
- IV. Unterstützende Dienste
 - Unterstützende Dienste als angehängte Ressourcen behandeln
- V. Build, release, run
 - Build- und Run-Phase strikt trennen
- VI. Prozesse
 - Die App als einen oder mehrere Prozesse ausführen
- VII. Bindung an Ports
 - Dienste durch das Binden von Ports exportieren
- VIII. Nebenläufigkeit
 - Mit dem Prozess-Modell skalieren
- IX. Einweggebrauch
 - Robuster mit schnellem Start und problemlosen Stopp
- X. Dev-Prod-Vergleichbarkeit
 - Entwicklung, Staging und Produktion so ähnlich wie möglich halten
- XI. Logs
 - Logs als Strom von Ereignissen behandeln
- XII. Admin-Prozesse
 - Admin/Management-Aufgaben als einmalige Vorgänge behandeln

Software und Virtualisierung

- No-Go's (sollte zumindest vermieden werden)
 - „große“ Container (z.B. im Gbyte-Bereich)
 - Umfangreiche Programme in Containern
 - Mehrere Programme in einem Container
- Ausweg: PouchContainer (von Alibaba)
 - Große Container (ganze Anwendungen)
 - <https://pouchcontainer.io/>

Docker - Grundlagen

- Das große Bild ... vollständige Virtualisierung vs. Container-Visualisierung mit Docker

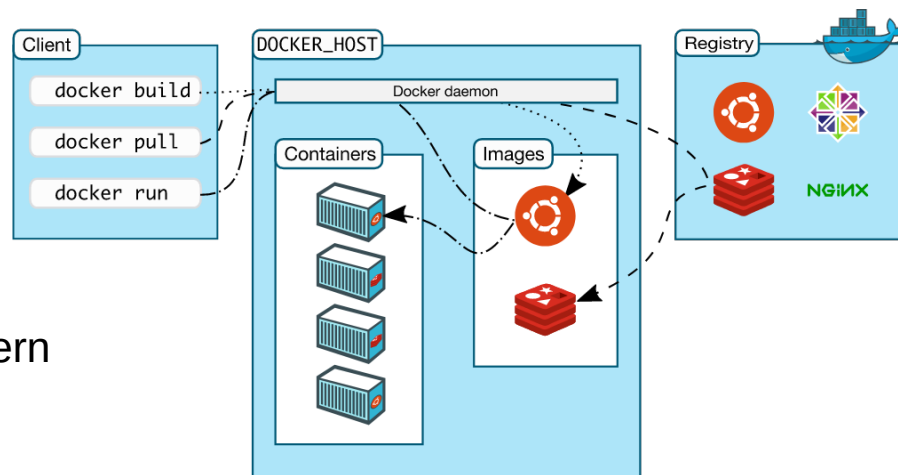


© IDG

Docker - Grundlagen

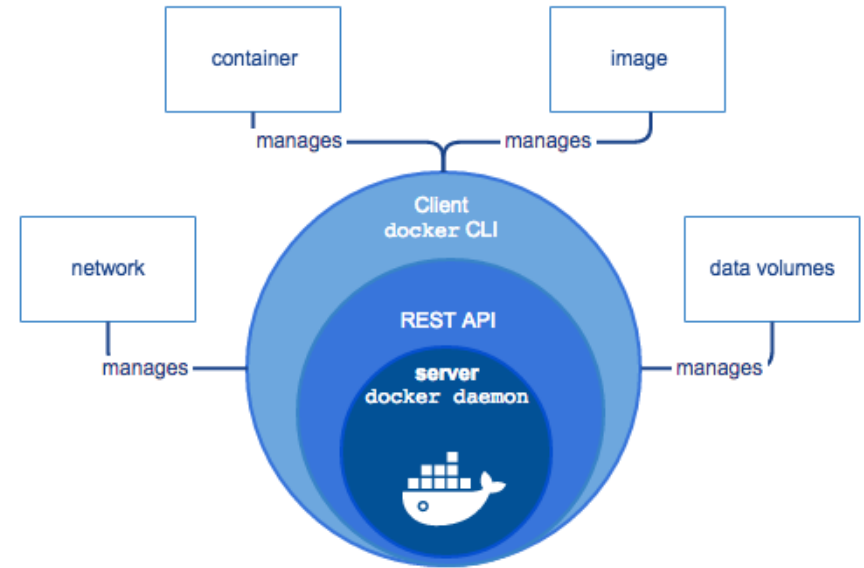
- Architektur

- Docker daemon (dockerd)
 - API-Anfragen verarbeiten
 - Objekte & Services verwalten
 - Container runtime (containerd) steuern
- Docker client (docker)
 - Interaktion mit Docker daemon über API
- Container registries
 - Verwaltung und Speicherung von images



Docker - Grundlagen

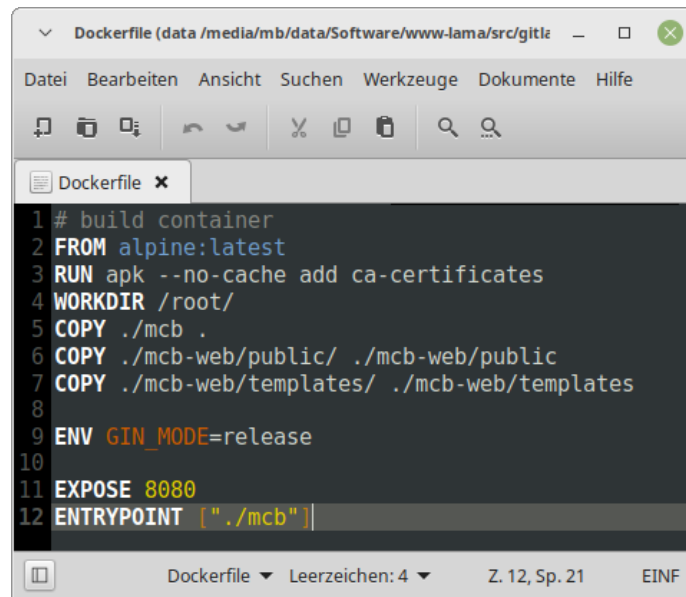
- Architektur – Docker objects
 - Images
 - Containers
 - Networks
 - Volumes



Docker - Grundlagen

- Architektur – Images

- Binäre Templates für Erzeugung von Containern
 - Programme
 - Bibliotheken
 - Metadaten und Anforderungen
 - Konfigurationen
- Sind read-only
- Können aufeinander aufbauen
- *Dockerfile* => Beschreibung der Konstruktion
- Jede Anweisung im *Dockerfile* erzeugt ein layer im Image
- Bei Veränderungen im *Dockerfile* wird nur das betreffende layer angepasst
- Speicherung in Registern und/oder als Archivdateien
- Werden mittels des Docker Daemon erzeugt



```
1 # build container
2 FROM alpine:latest
3 RUN apk --no-cache add ca-certificates
4 WORKDIR /root/
5 COPY ./mcb .
6 COPY ./mcb-web/public/ ./mcb-web/public
7 COPY ./mcb-web/templates/ ./mcb-web/templates
8
9 ENV GIN_MODE=release
10
11 EXPOSE 8080
12 ENTRYPOINT ["./mcb"]
```

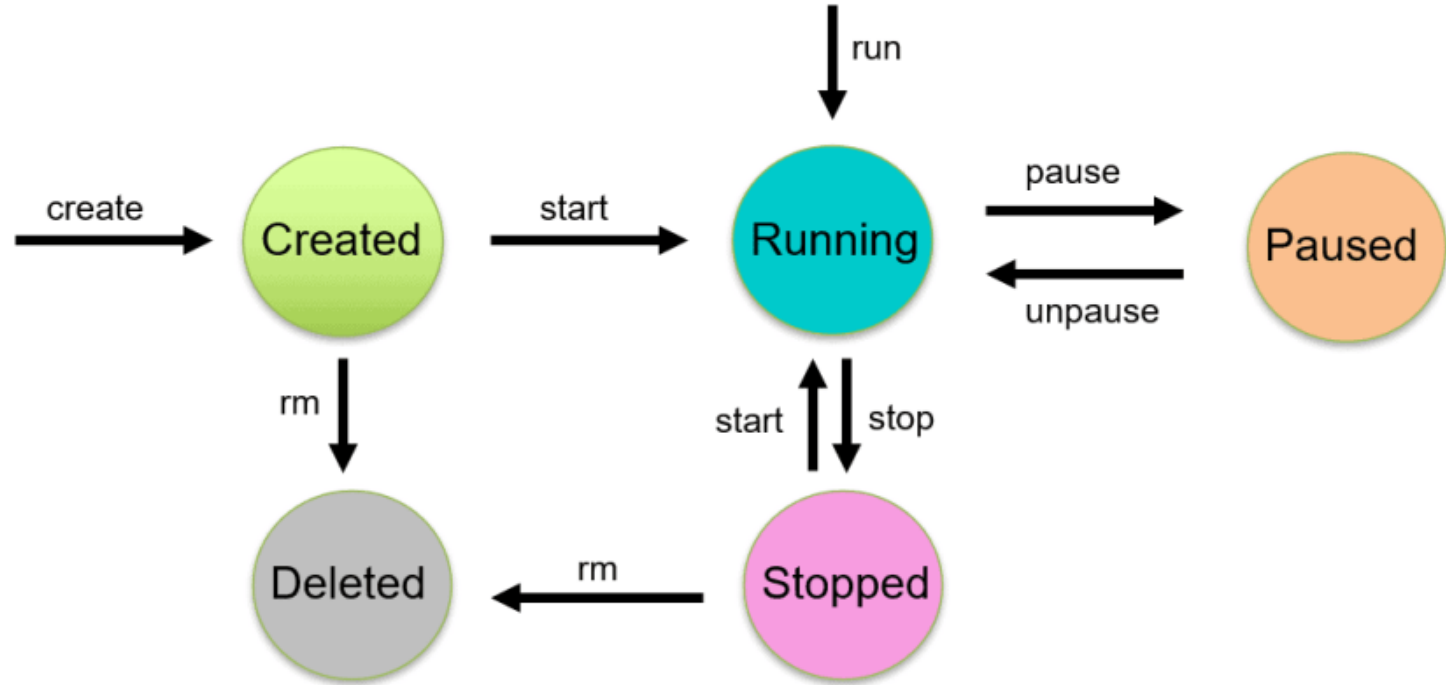
The screenshot shows a text editor window titled 'Dockerfile (data /media/mb/data/Software/www-lama/src/gitl...)'. The editor has a menu bar with 'Datei', 'Bearbeiten', 'Ansicht', 'Suchen', 'Werkzeuge', 'Dokumente', and 'Hilfe'. Below the menu is a toolbar with icons for file operations. The main text area contains a Dockerfile with 12 lines of code. The status bar at the bottom indicates 'Dockerfile', 'Leerzeichen: 4', 'Z. 12, Sp. 21', and 'EINF'.

Docker - Grundlagen

- Architektur – Containers
 - Lauffähige Instanzen von Images
 - Laufen in einer gekapselten/isolierten Umgebung
 - Initial Zugriff ausschließlich auf eigene Ressourcen
 - Beim Start können zusätzliche Parameter übergeben werden ...
z.B. Netzwerk, Zugriff zu Volumes
 - Können sehr schnell starten
 - Management durch den Docker Daemon über API und Clients
 - Laufen „stateless“ ... keine persistente Speicherung von Daten

Docker - Grundlagen

- Lebenszyklus von Containern



Docker - Grundlagen

- Architektur – Networks
 - Konfiguration der Kommunikation von Containern
 - Netzwerk-Treiber in Docker:
 - Bridge (default)
 - Host – keine Isolierung zwischen Host und Container
 - Overlay – für Kommunikation in Clustern (multi-host)
 - None
 - Macvlan – Container bekommen eigene MAC-Adressen

Docker - Grundlagen

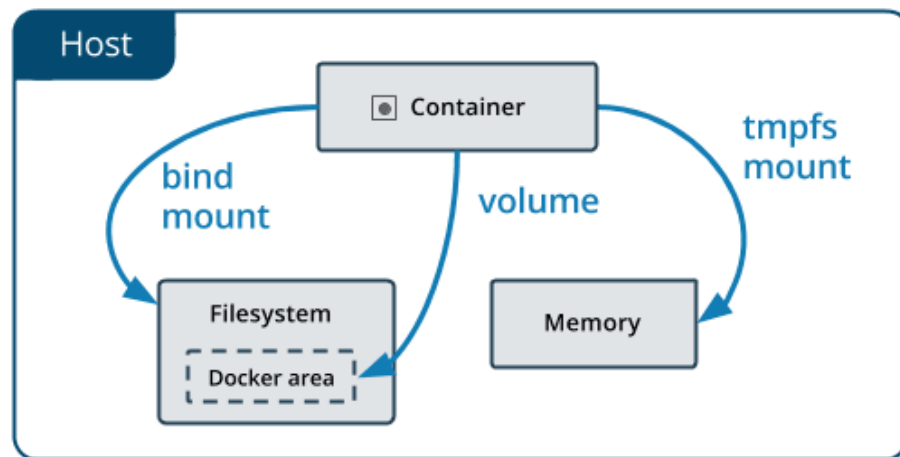
- Architektur – Volumes
 - Persistente Speicherung von Daten
 - Verwaltung über Client und API
 - Außerhalb des Lebenszyklus von Containern
 - Zuweisung von Volumes beim Start von Containern

Docker - Grundlagen

- Architektur – Auswahl des Datenspeichers
 - *Temporäre Daten* im writeable Teil von Containern
... vergrößert den Speicherbedarf
 - *Persistente Daten* (und temporäre Daten):
 - Data Volume – Volume im Filesystem des Hosts
 - Volume Container – Volume eines Containers
 - Directory Mounts – Verzeichnisse auf dem Host
 - Storage Plugins – Verbindung zu externen Storages

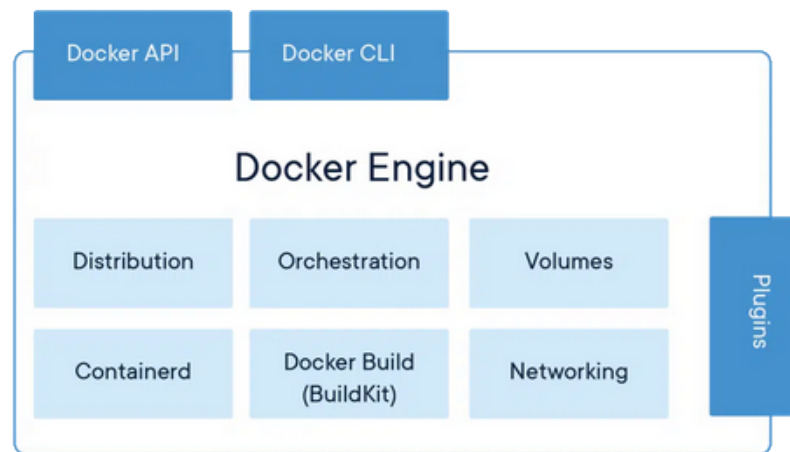
Docker - Grundlagen

- Architektur – Auswahl des Datenspeichers
 - Volume
 - Im Host-Filesystem
 - Zugriff nur aus Containern
 - Verwaltung durch Docker Daemon
 - Bind mount
 - Im Host-Filesystem
 - Zugriff auch vom Host aus
 - Tmpfs mount
 - Keine persistente Speicherung
 - Nur im RAM gehalten



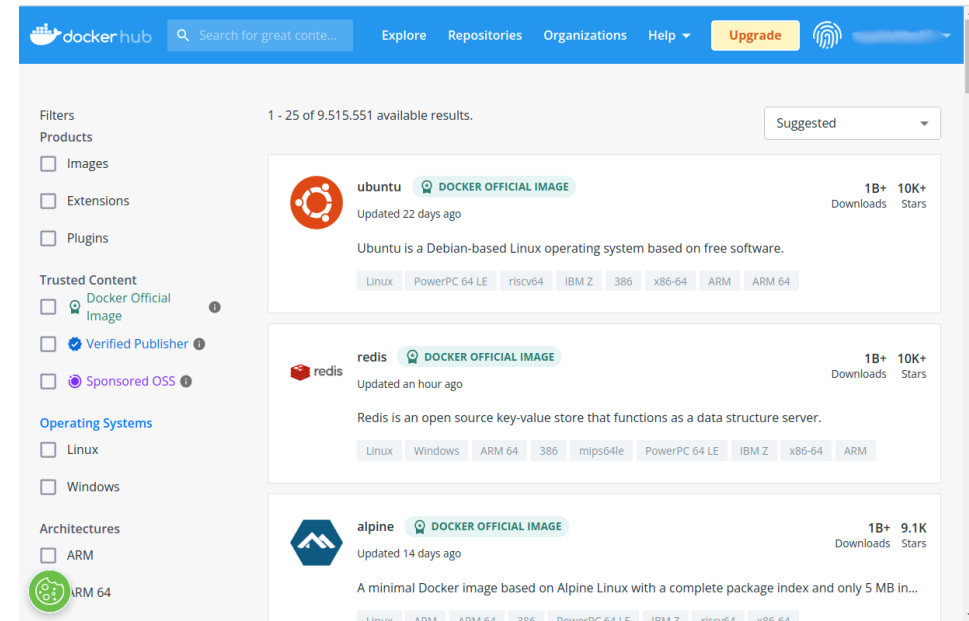
Docker - Grundlagen

- Produkte – Docker Engine
 - Paket aus
 - Docker Daemon (dockerd)
 - Runtime (containerd)
 - Client (docker)
 - Plattformspezifische Treiber
 - Erweiterungen
 - Unterschiedliche Lizenzen



Docker - Grundlagen

- Produkte – Docker Hub
 - Cloud-basierter Registry-Service
 - <https://hub.docker.com/>
 - Verwaltung aus Browser
 - Verzeichnis von images
 - Eigene Repositories
 - Build-Pipeline



Docker - Grundlagen

- Produkte – Docker Desktop
 - Desktop Anwendung für Windows, Linux, Mac
 - Electron-App für Verwaltung der Docker Engine
 - Einfache Installation
 - Enthält zusätzlich
 - Docker Compose
 - Kubernetes
 - Management einer Bibliothek von Extensions

Docker - Grundlagen

- Docker EE – Enterprise Edition
 - Zertifizierte Version für viele Plattformen
 - *Personal* ... nur persönliche Nutzung, wenig Rechte
 - *Pro, Team, Business* ... kommerzielle Nutzung mit Features und Support in Abstufungen
 - Viele Erweiterungen
 - Langfristige Sicherheitsupdates

Docker - Grundlagen

- Docker CE – open source
 - Docker Engine als „Community Edition“
 - Apache License 2.0
 - „Support“ ausschließlich über Community Foren
 - *ohne* Docker Desktop und ohne Support
 - *Edge* ... jeden Monat neu
 - *Stable* ... Sicherheitsupdates nur für ein Quartal

Docker - Grundlagen

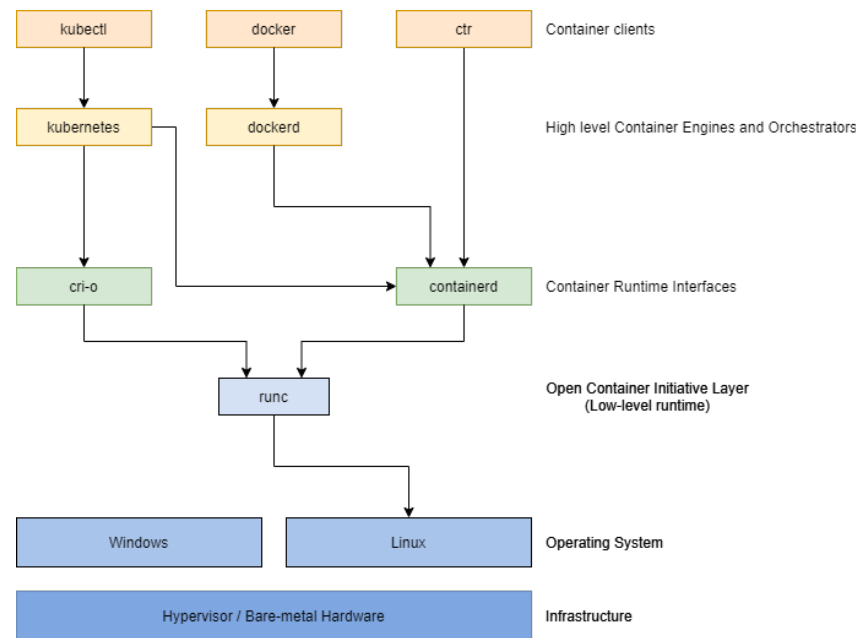
- Produkte ... open source

- containerd

- <https://containerd.io/>
 - Container runtime
 - Teil der Docker Engine
 - Teil des CNCF-Stacks

- containerd CTL

- <https://github.com/containerd/nerdctl>
 - containerd kompatibles CLI



Docker – Grundlagen

- Windows-Container

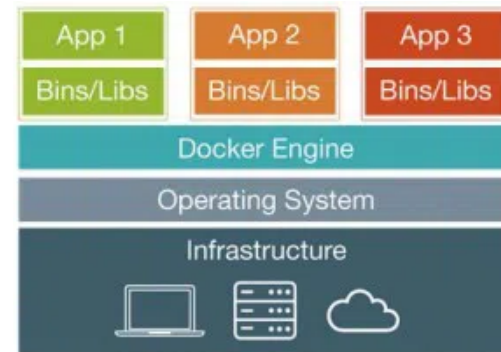
- Container nutzen das Host-Betriebssystem

=> Windows-Container sind nur auf einem Windows-Host lauffähig

- Sind „größer“ als Linux-Container
 - Sind „langsamer“ als Linux-Container (Startup-Zeit)
 - Besitzen kein graphisches Nutzerinterface
 - Wenige (und beschränkte) Basis-Images

<https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/container-base-images>

https://docs.microsoft.com/en-us/virtualization/windowscontainers/?WT.mc_id=AZ-MVP-5003649



Docker – Grundlagen

- Windows-Host
 - Mit Docker Desktop und WSL (Windows Subsystem for Linux)
 - Mit Container Runtime auf Windows-Servern
 - Kann Linux- & Windows-Container managen
 - ... nicht gleichzeitig
 - ... unter *Azure Kubernetes Service* fast gleichzeitig (auf getrennten Linux-Nodes und Windows-Nodes)

... Bildquellen

- RedHat
- IDG
- Docker
- Earthly
- Aquasec
- dev.to