

Software und Plattform Architektur

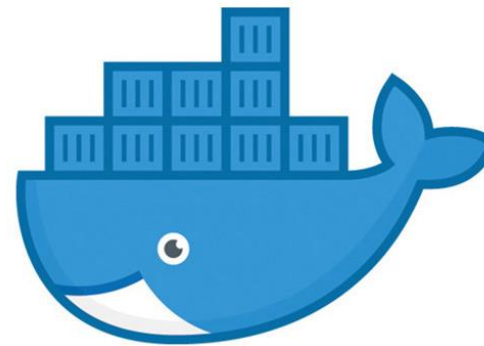
Einführung Container/Docker

Definition von Container

⚠ Es existiert keine standardisierte Definition von Containern am besten werden sie durch ihre Funktion beschrieben.

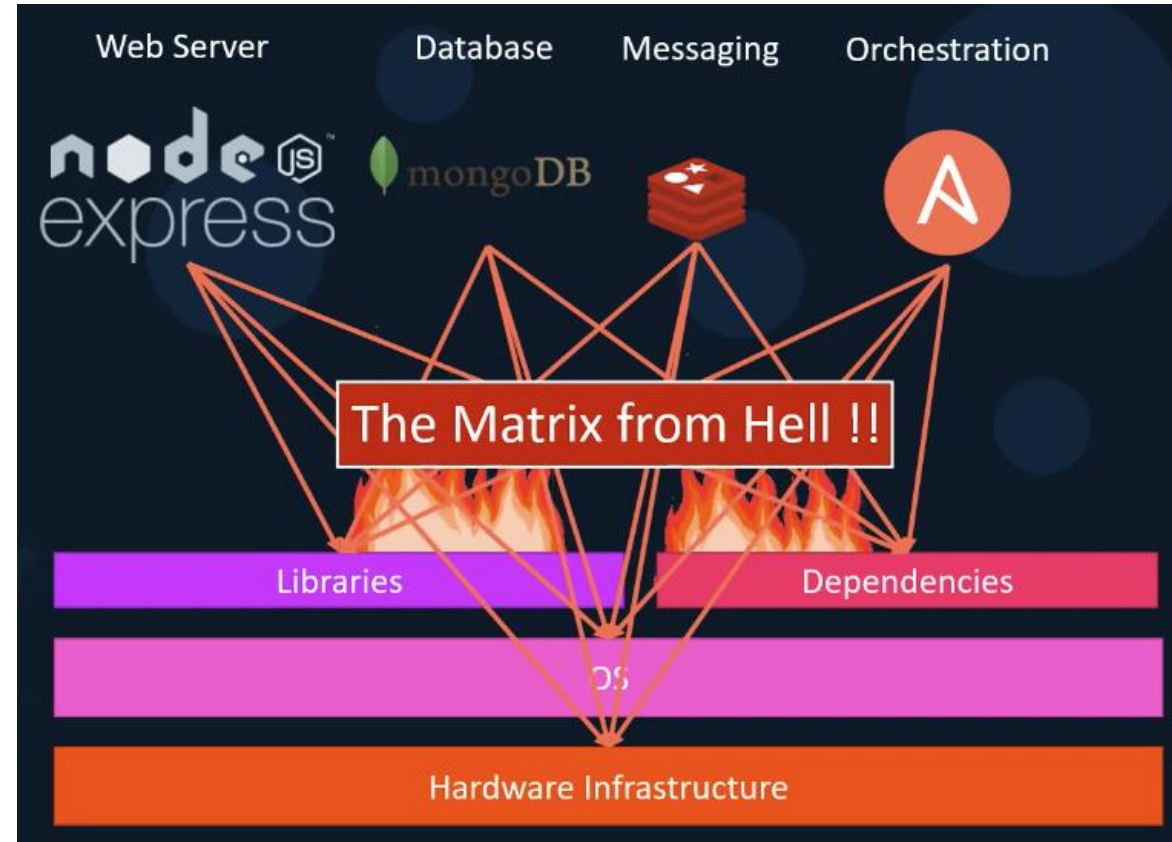
Analogie

Container für den Transport in der Logistik ermöglichen es, Waren jeglicher Art in einer standardisierten “Verpackung” flexibel und schnell mit unterschiedlichen Verkehrsmitteln zu verschicken und zu verteilen – ob Schiff, Flugzeug, Bahn oder LKW. Ähnlich operieren IT-Container. Sie verpacken eine Anwendung und alle zu ihrer Ausführung erforderlichen Dateien und Abhängigkeiten in ein handliches Paket.



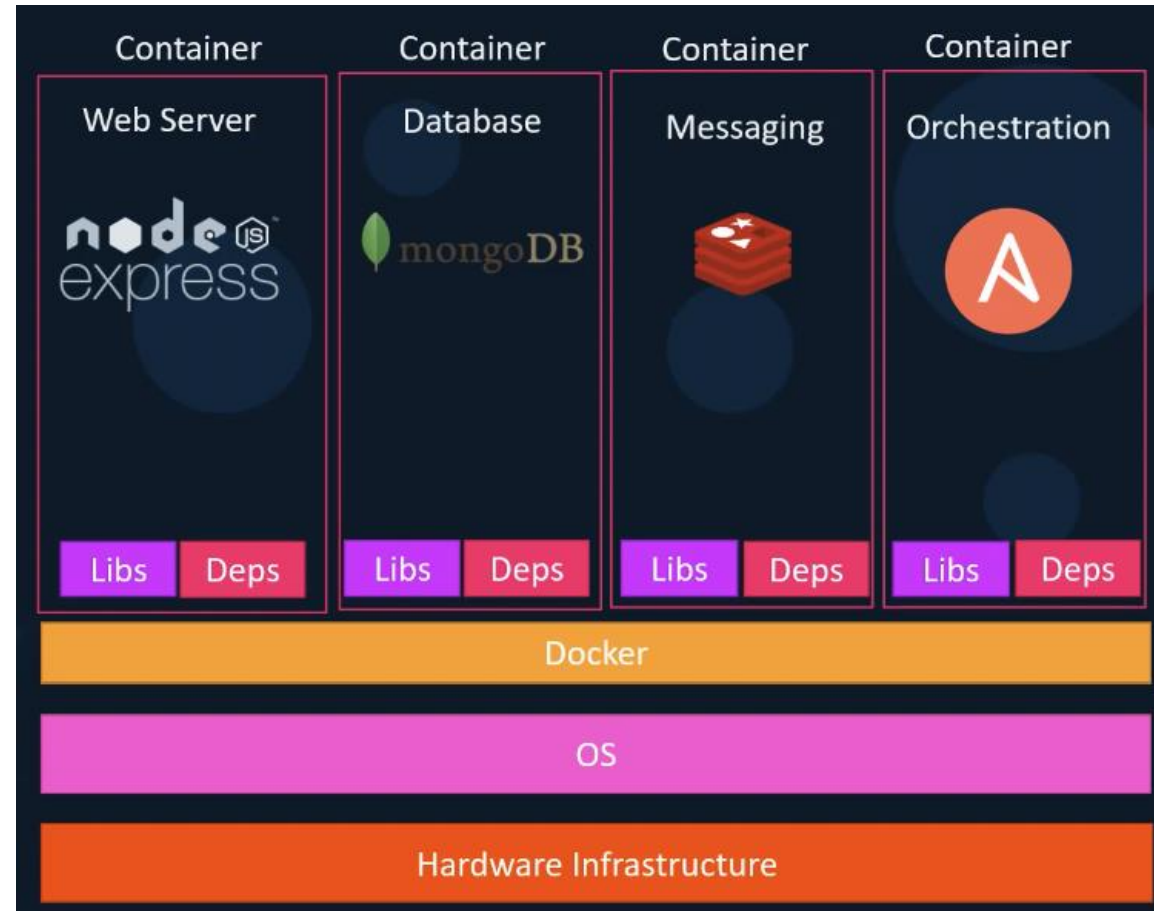
Warum brauchen wir Container? Das Problem

- Keine Isolation von Prozessen
- Kompatibilitätskonflikte da Applikationen jeweils andere Abhängigkeiten haben
- Aktualisierung von einzelnen Services kann anderen Service beeinflussen



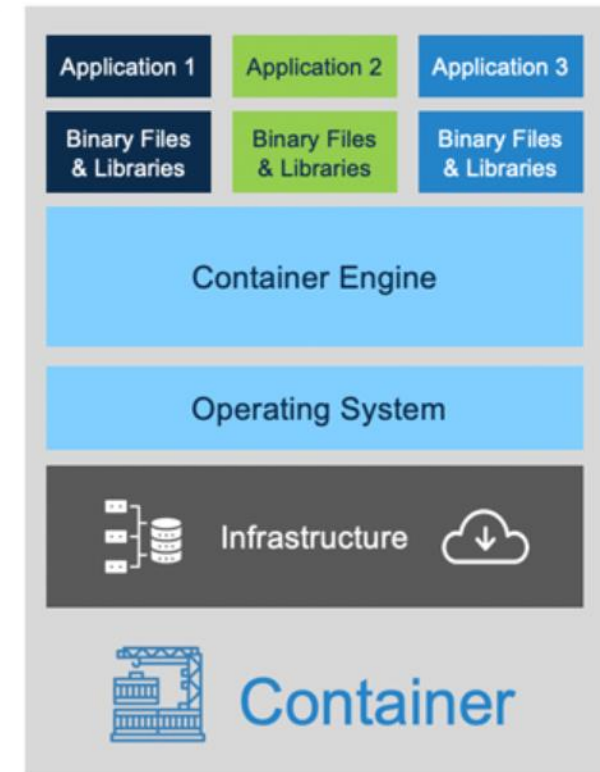
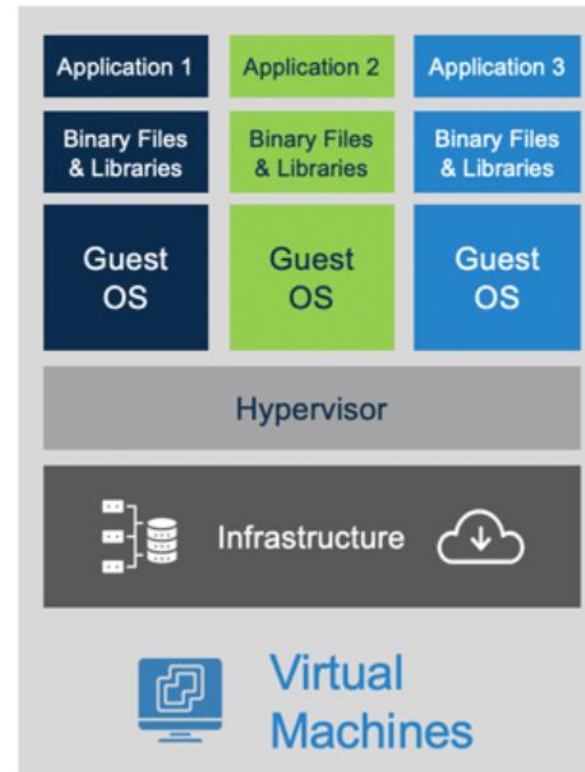
Warum brauchen wir Container? Die Lösung

- Kompatibilitätskonflikte auf ein Minimum reduziert da jeder Container seine eigenen Libraries / Dependencies enthält
- Isolation der Prozesse welche die Sicherheit erheblich erhöht



Unterschied zu virtuellen Maschinen

- Hat keinen eigenen Kernel und verwenden den vom Hostsystem
- Besitzt eigenes isoliertes Dateisystem
- Betreibt in der Regel nur 1 Prozess
- Somit ist ein Container kein eigenes Betriebssystem sondern setzt jeweils auf ein bestehendes Betriebssystem einer VM.



Vorteile von Containern

- geringerer Ressourcenbedarf und Overhead als virtuelle Maschinen
- schnelles Starten und Ausführen der Container
- sichere Trennung der Anwendungen in verschiedenen Containern
- schnelles und einfaches Verschieben kompletter Anwendungen inklusive ihrer Laufzeitumgebungen auf andere Systeme
- gute Skalierbarkeit und einfache Anpassung der Kapazitäten an die Anforderungen der Anwendung
- große, über mehrere Rechner verteilte Container-Setups realisierbar

Nachteile von Containern

- Container auf einem Host-System müssen das gleiche Betriebssystem nutzen
- Kompatibilitätsprobleme der Anwendungen mit dem Betriebssystem sind schwerer zu lösen; Lösungen haben eventuell Einfluss auf andere Container auf dem System
- Softwarefehler oder Sicherheitslücken können sich auf alle Container und das komplette Betriebssystem auswirken
- Know-How und eine entsprechende Plattform welche Container betreiben kann muss vorhanden sein.


Anwendungsgebiete

- Anwendungen die viele und/oder spezielle Abhängigkeiten haben
 - Java Applikation welche eine bestimmte Version der JVM und bestimmter Pakete benötigt
- Anwendungen welche häufigen Updates unterliegen
 - Sicherheitsrelevante Applikationen welche sofort mit Sicherheitskritischen Updates beliefert werden müssen z.B. Passwortmanager
- Microservice-Architekturen
 - Webapplikationen mit Frontend / Backend / Datenbank / Caching
- Applikationen welche horizontal gut skalieren
 - Datenbankcluster
 - NodeJS / Python Applikationen



Mit anderen Worten Container sind heutzutage ÜBERALL einsetzbar!

Container / Docker / Kubernetes

- Container
 - Bezeichnet eine Technik in der IT wie Applikationen isoliert und gepackt werden können. Ist nicht abhängig von einer konkreten Technologie oder einem Produkt.
- Docker
 - Beschreibt ein konkretes Produkt und eine Firma welche aktuell als Standard für die Containerisierung von Anwendungen gesehen werden kann.
- Kubernetes
 - Beschreibt ein konkretes Produkt welches aktuell als Standard für die Container Orchestrierung gesehen werden kann.
-  Somit wichtig -> Container != Docker != Kubernetes

Wiederholung

- Beschreibe den Begriff Container in Ihren eigenen Worten
- Warum brauchen wir Container?
- Was ist der Hauptunterschied zwischen Container & VM's?
- Was sind Die Vorteile?
- Was sind die Nachteile?
- In welchen Bereichen werden Container eingesetzt?
- Unterschied von Container vs. Docker vs. Kubernetes


Was ist Docker? Überblick

Der Begriff „Docker“ kann sich auf verschiedene Dinge beziehen. Das können sein:

- ein Open Source Community-Projekt
- Tools des Open Source-Projekts
- Docker Inc., das Unternehmen, das dieses Projekt hauptsächlich unterstützt und die Tools, die von Docker offiziell unterstützt werden.

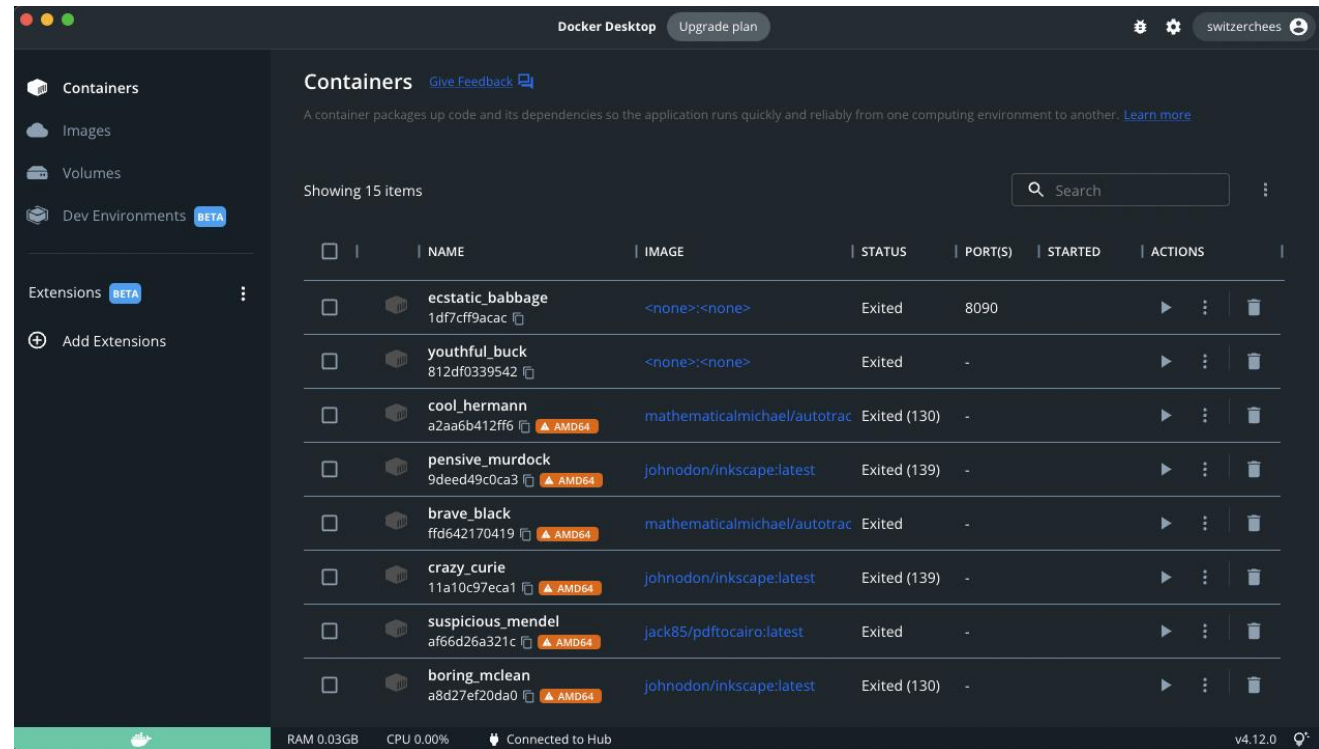
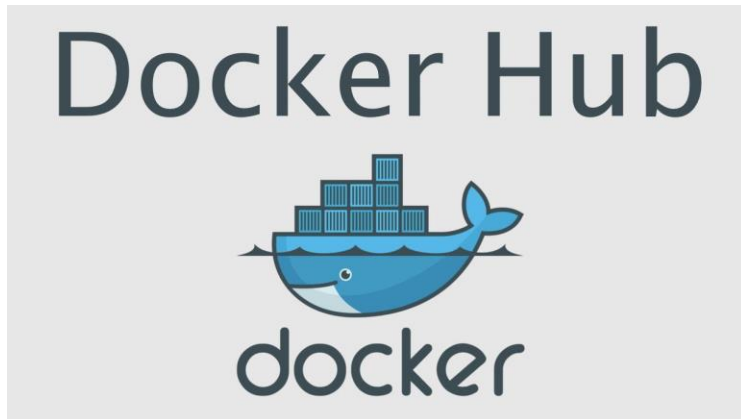
Die Tatsache, dass die Technologien und die Firma den gleichen Namen haben, kann verwirrend sein.

Wie funktioniert Docker?

- Die Docker-Technologie verwendet den Linux Kernel und seine Funktionen wie **cgroups** und **namespaces**, um Prozesse zu isolieren, damit diese unabhängig voneinander ausgeführt werden können.
 - Diese Unabhängigkeit ist der Zweck der Container – die Fähigkeit, mehrere Prozesse und Apps getrennt voneinander betreiben zu können.
 - So wird Ihre Infrastruktur besser genutzt und gleichzeitig die Sicherheit bewahrt, die sich aus der Arbeit mit getrennten Systemen ergibt.
-  Man merke ein Container ist kurzgesagt ein eigener namespace im Linux Kernel des Hostsystems

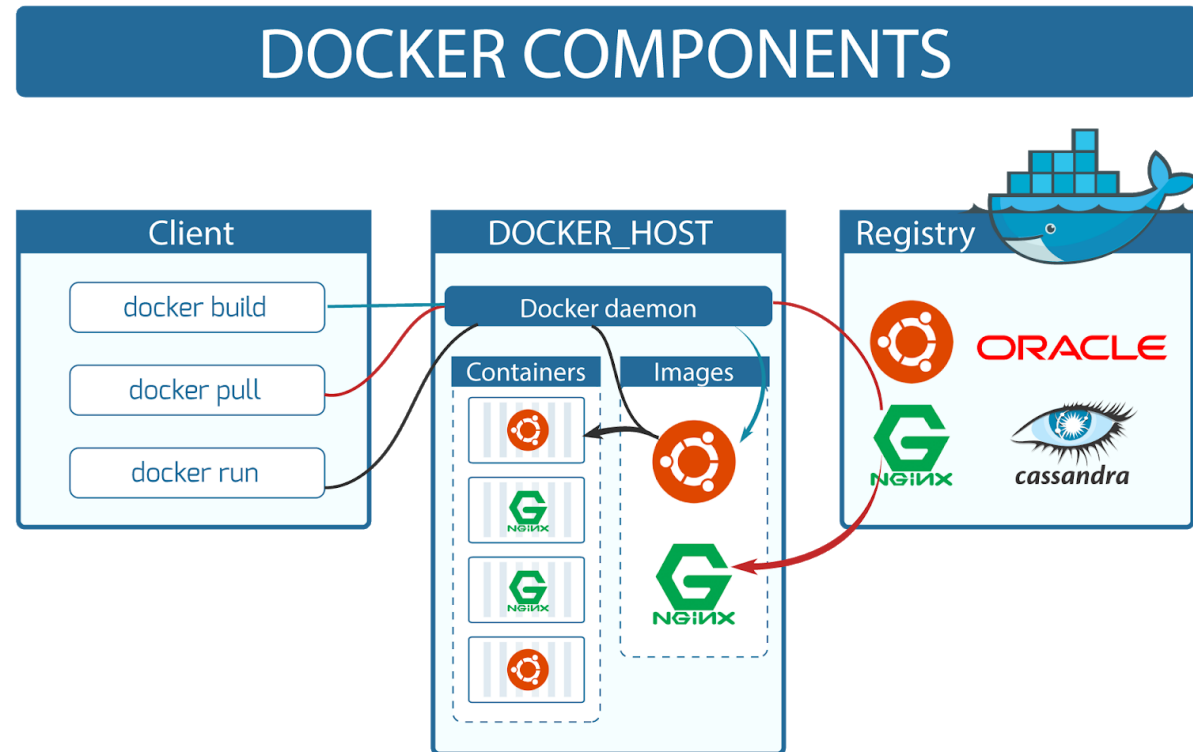
Warum ist Docker so beliebt oder populär?

- Einfaches CLI-Interface
- Image-basiertes Bereitstellungsmodell
- Tools und Ökosystem, welches das Arbeiten mit Docker Containern sehr benutzerfreundlich macht dazu gehören Tools wie
 - Docker Desktop
 - Docker Hub



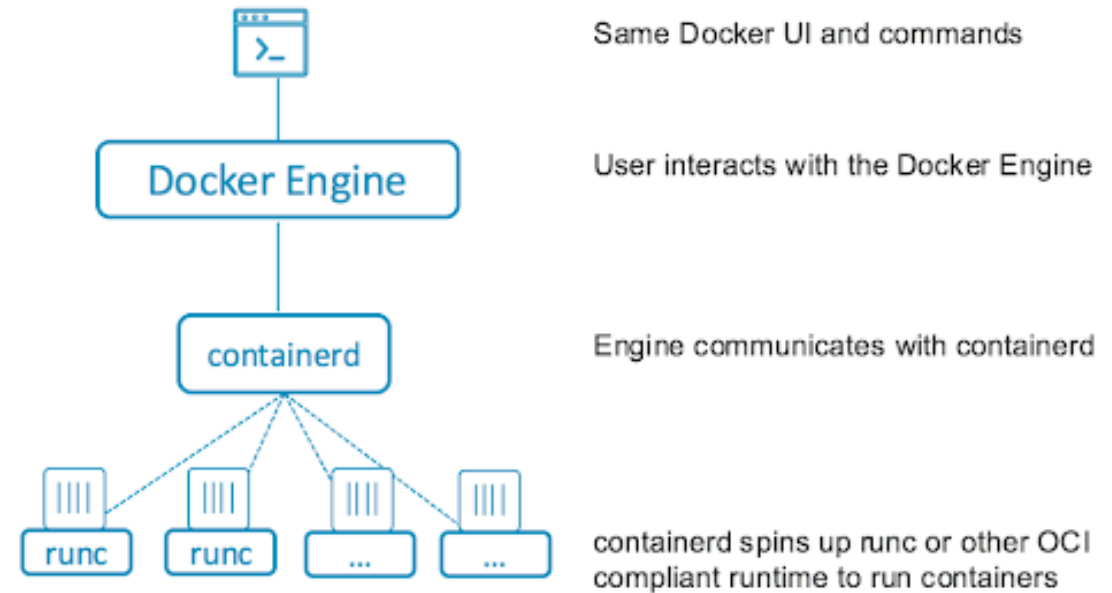
Docker Komponenten

- Client = CLI
- Docker Engine/Daemon
 - Systemprozess welcher Container erstellt, beinhaltet die Container Runtime containerd
- Registry
 - Ablage für Container Images ähnlich wie **git** für Code



Docker Engine vs. containerd

- Docker Engine
 - Interagiert direkt mit dem Endbenutzer und interpretiert deren Eingaben
- containerd
 - Hintergrundprozess welcher die Kommandos der Engine ausführt aber nicht direkt mit dem Endbenutzer interagiert




Docker Editionen

- Docker Community Edition (CE)
 - Open Source Projekt
 - Kein offizieller Support
 - Gratis
 - Läuft auf CentOS, Debian, Ubuntu usw.

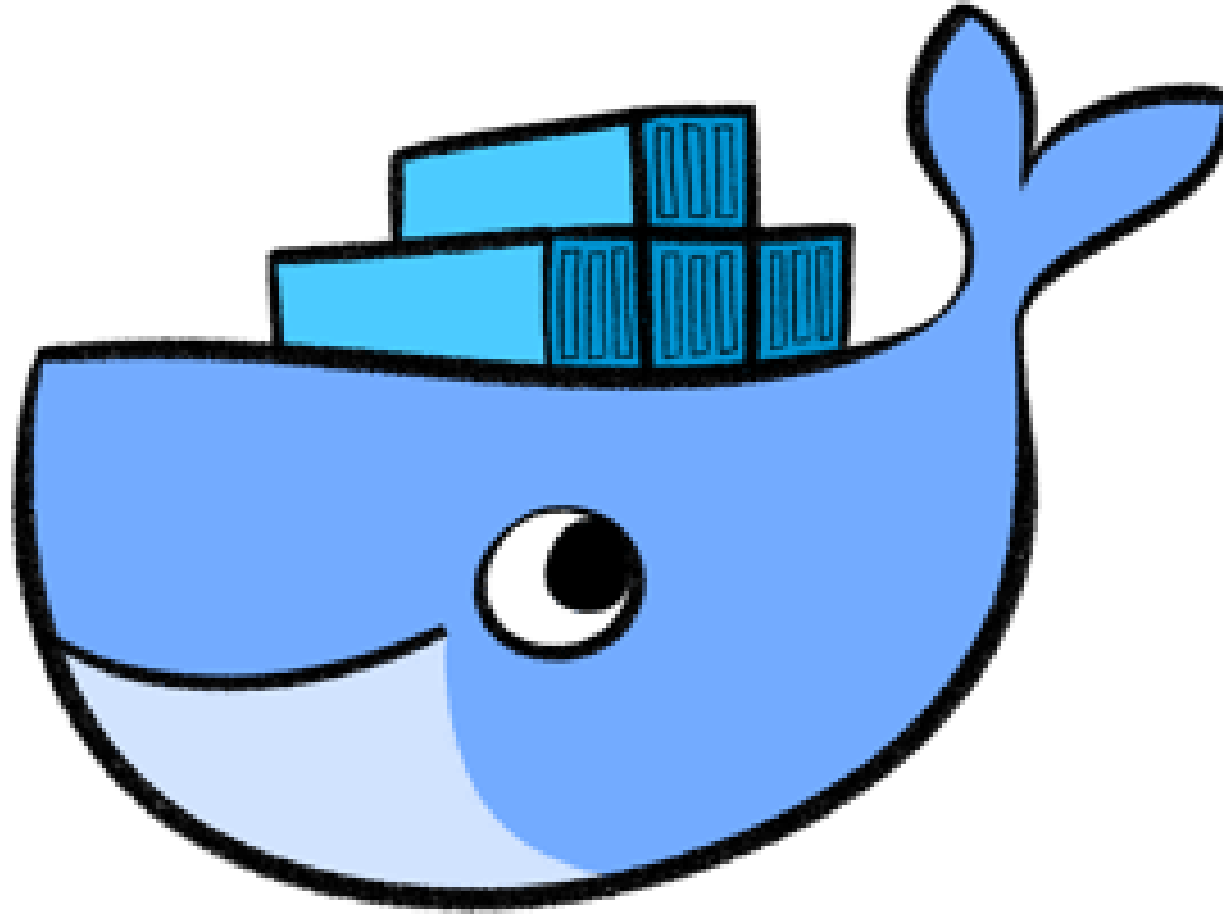


- Docker Enterprise Edition (EE)
 - Zertifiziert und offiziell supportet von Docker Inc.
 - Kostenpflichtig
 - Läuft auf Red Hat Enterprise Linux (RHEL), Suse usw.

Docker Wichtigste Befehle


- `docker ps`
 - Gibt alle laufenden Container zurück
 - `docker run <image>`
 - Startet das angegebene Image als Container
 - `docker exec -it <container-id> sh/bash`
 - Öffnet ein interaktives Terminal zum angegebenen Container
-  Für weitere Befehle ist unter «04_Data» ein Cheatsheet abgelegt.

Docker kurze Demo



Übung (120 Minuten)

- Aufgabe
 - Bearbeiten Sie das Arbeitsblatt «01_Einführung_Docker.docx»

 Machen Sie sich aus den Befehlen ein eigenes kleines Cheat Sheet, das kann später einmal nützlich werden