TMF – Technologie- und Methodenplattform
für die vernetzte medizinische Forschung e.V.

Dockerbank

Container-basiertes Deployment von biomedizinischen IT-Lösungen

Block 2: Grundlagen Container-Virtualisierung

Matthias Löbe

Institut für Medizinische Informatik, Statistik und Epidemiologie (IMISE), Universität Leipzig

Motivation

Was sind Docker-Container?

- Container sind mittels Virtualisierungstechniken erstellte modulare Pakete, die lauffähige Anwendungen oder Dienste in einer abgekapselten Umgebung bereitstellen

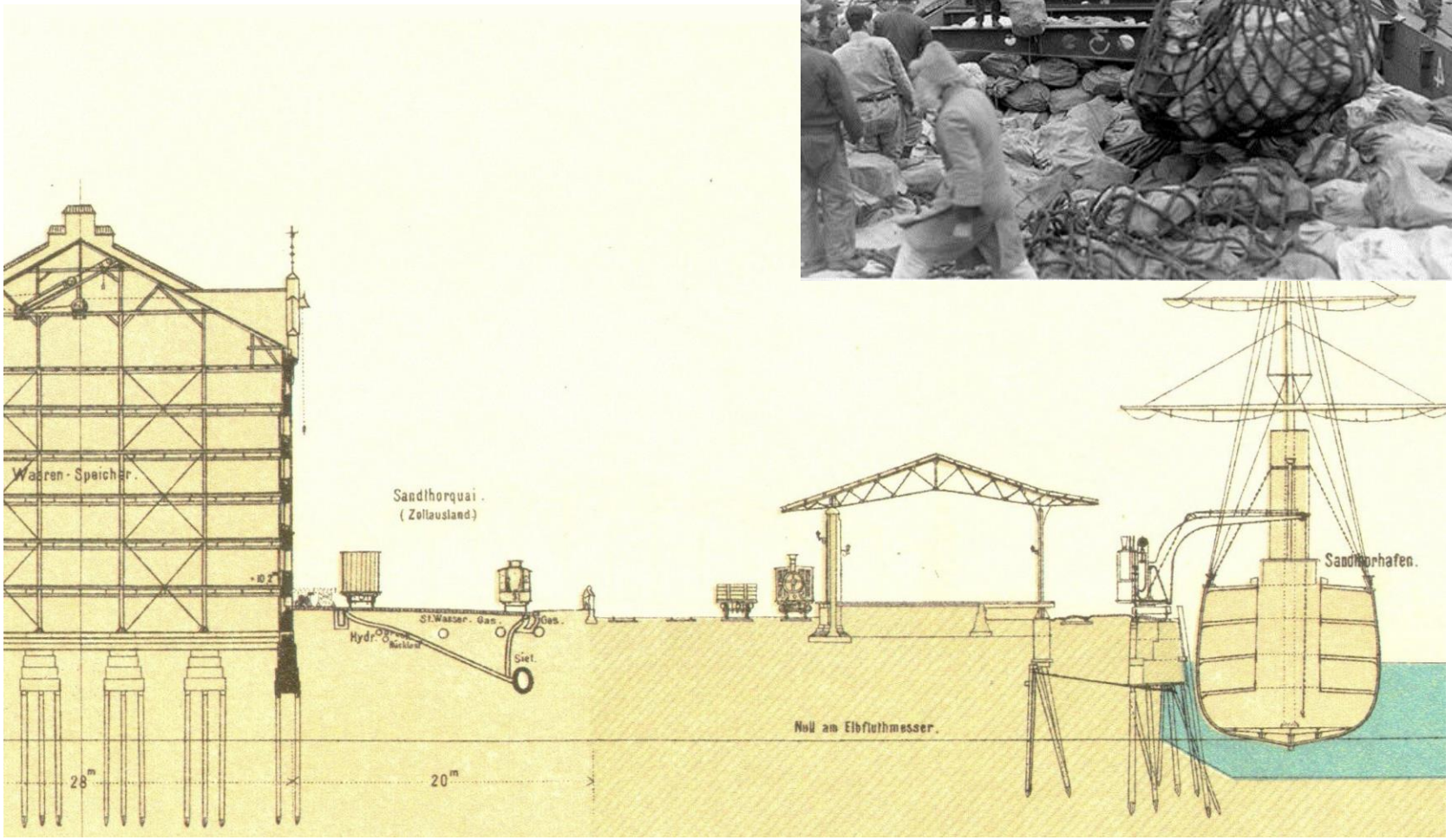
Warum Container? Ist das nicht zusätzlicher Ballast?

- Neue Technologie zu lernen und zu schulen
- Neue Verfahrensanweisungen für Erstellung und Betrieb nötig

Ja, aber…

Quelle: Wikipedia, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=1443327

Früher: Transport- und Umschlag-probleme bei Gütern

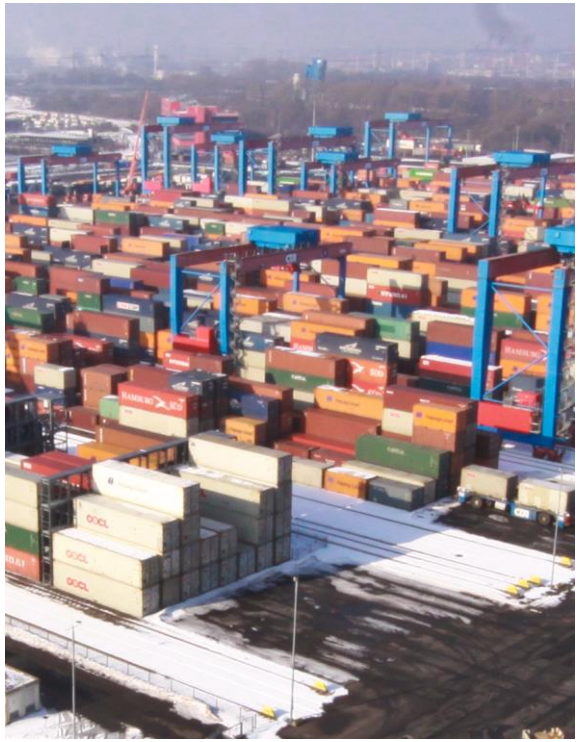


Quellen: Franz Andreas Meyer: Aus dem Hamburger Freihafengebiet, Hamburg 1888, Strumper & Co und Public Domain, https://commons.wikimedia.org/w/index.php?curid=1852088 (rechts oben)

„Containerization“ in den 60er Jahren

- Bessere Raumausnutzung durch genormte Abmaße (Aufstellfläche und Innenraum), stapelbar
- Besser geschützt gegen Beschädigungen und Diebstahl
- Effizient be- und entladbar sowie transportierbar
- Beschleunigung bei Abfertigung (Zoll)
- Geschlossene Transportkette Land-See-Land
- Höhere Transportsicherheit

Heute ~90% aller Schiffstransporte mit Containern



Quelle: Frank Grunwald, original unter https://de.wikipedia.org/wiki/Containertransport#/media/File:HHLA_Container_Terminal_Altenwerder_%28CTA%29_in_Hamburg_-_Winter_2010_-_04.jpg

„Containerization“ in den 2010er Jahren

„Bessere Raumausnutzung“	Verzicht auf Betriebssystem im Container, Ressourcenallokation über cgroups, Overlay-FS
„Schutz gegen Beschädigungen und Diebstahl“	Virtuelle Netzwerke, wenige offene Ports, derzeit keine Signierung
„Effiziente Beladung, Transport, Entladung“	Sowohl einfache Shellkommandos wie auch komplexe Deploymenttols, zentrales Verzeichnis
„Beschleunigte Abfertigung (Sysadmin)“	Nachvollziehbarkeit von Änderungen durch Modifikationsskripte (Dockerfiles)
„Geschlossene Transportkette“	Gewisse Betriebssystemunabhängigkeit, Unterstützung durch Cloud-Provider
„Höhere Transportsicherheit“	Isolierung der Prozesse durch Linux Namespaces

Was ist Docker?

- Eine Client-Server-Plattform zum Erzeugen, Verteilen und Laufenlassen (vernetzter) Anwendungen (Motto: “Ship, Build, Run”)
- Läuft auf 64bit-Linux Distributionen ab Kernel 3.8
  - Teil von RedHat, für Suse und Debian/Ubuntu im Repository
  - Von allen namhaften Cloud-Plattformen unterstützt (Amazon EC2, Google Compute Engine, Rackspace, ...)
  - Implementierung in Windows Server 2016
- Idee: Ein Dienst je Container
  - Container enthälten alle Abhängigkeiten (Libs, Konfigurationsdateien)
  - Container bauen schichtenartig aufeinander auf
  - „Microservice“-Konzept
- Von Entwicklern geliebt für einfache Test auf multiplen Plattformen
- Open-Source-Lizenz (Apache 2.0)

Architektur

cgroups

- Allokation von Ressourcen (CPU, RAM, IO)

namespaces

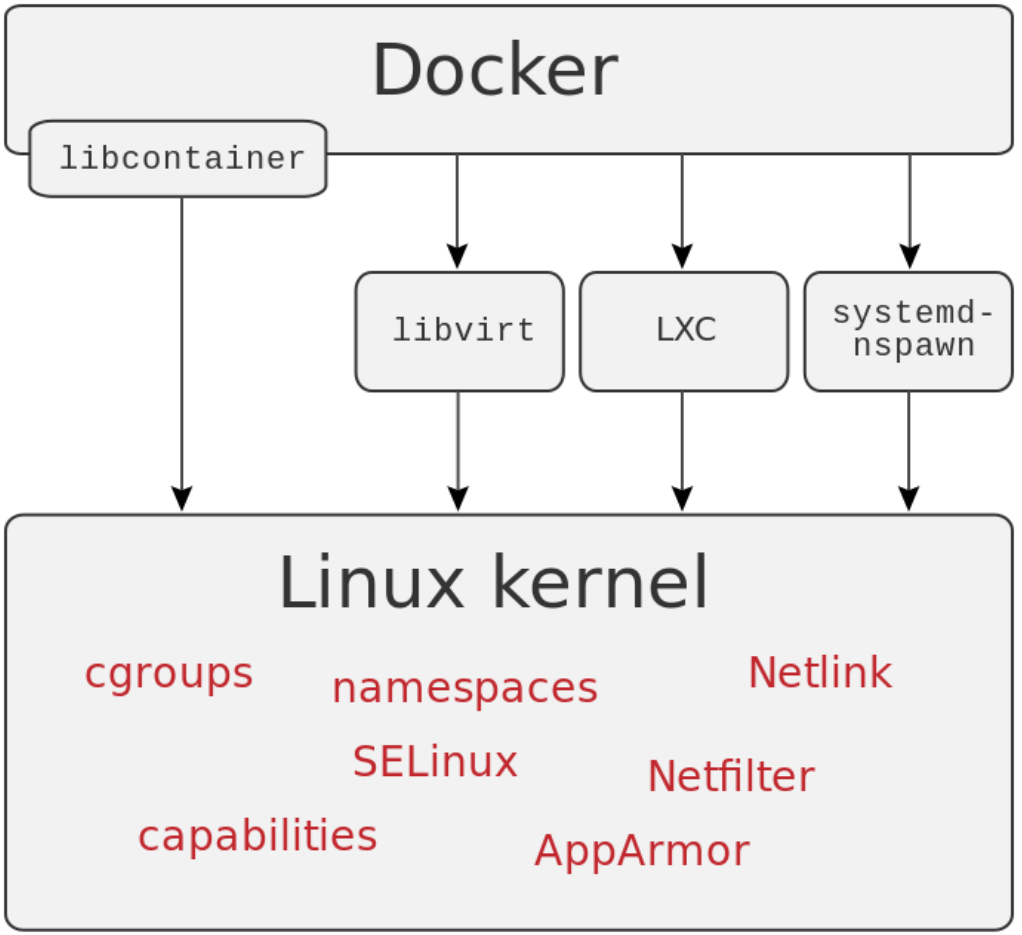
- Prozessisolation

aufs

- Another Unionfs
- Overlay-Dateisystem

SELinux/AppArmor

- Sicherheitserweiterungen



Quelle: Von User:Maklaan - Based on a Docker blog post, Gemeinfrei, https://commons.wikimedia.org/w/index.php?curid=37965701

Docker-Historie

Timeline

03/2013: Docker 0.1 veröffentlicht

11/2013: Docker 0.7 ohne Kernelpatches

06/2014: Docker 1.0 erste Produktivversion

Start von Docker Hub und des Docker-Hypes

02/2015: Docker 1.5 großes Update, IPV6, Statistiken

07/2016: Google veröffentlicht Kubernetes

Open-Source-Framework zum Verwalten von Docker-Containern

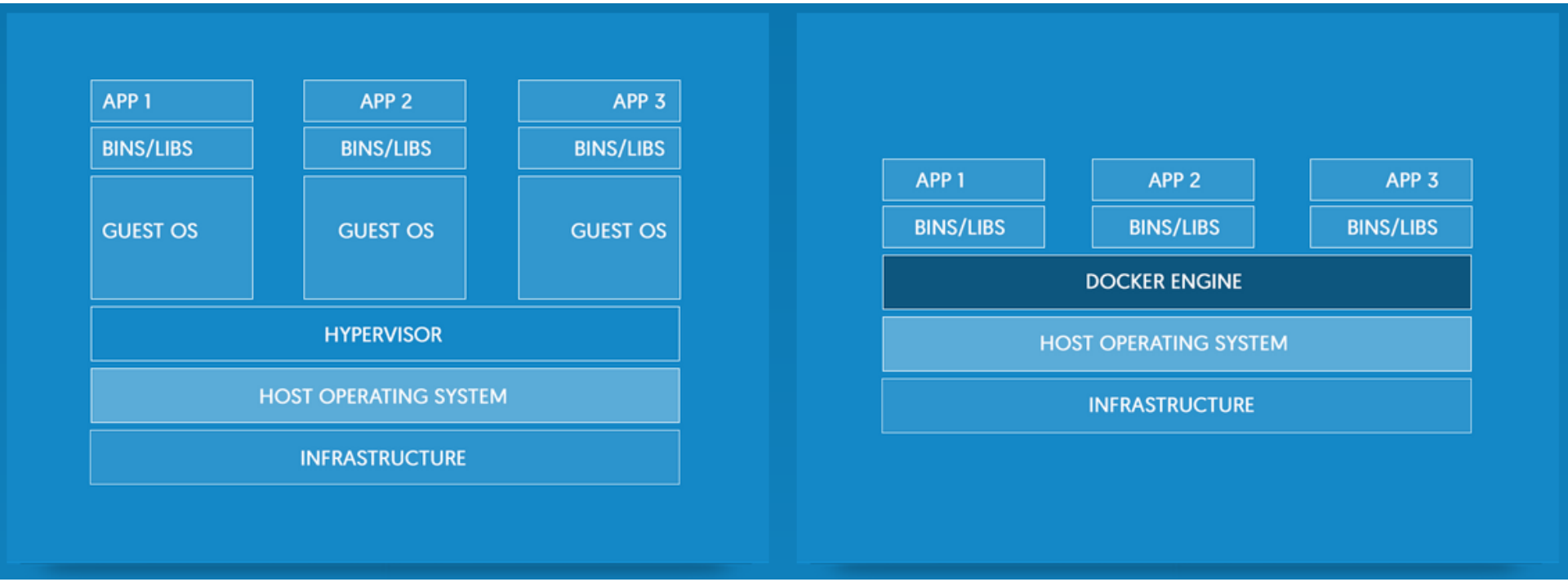
11/2015: Docker 1.9 Networking, Orchestrierung

07/2016: Docker 1.12 aktuelle Version

Releases regelmäßig aller 2-3 Monate

- Kompatibilität nicht immer gewährleistet

Vergleich Virtualisierung und Containerisierung



Quelle: Docker, Inc. [https://www.docker.com/what-docker]

- Ähnliches Konzept: Abstraktion der Hardware, verschiebbar, umfasst alle Abhängigkeiten
- Aber deutlich schlanker (nur ein Kernel), schneller und weniger abhängig von einer bestimmten Infrastruktur

Docker vs. VMware?

Docker will VMs nicht ersetzen! Paradigma „Haustier“ vs. „Herde“

„Haustier“ » Virtuelle Maschine

- Geringe Stückzahl
- Liebevoll manuell gepflegt
- Lange Haltedauer

„Herde“ » Dockercontainer

- Große Stückzahl
- Automatisiert gezüchtet und verwaltet
- Kurze Haltedauer

Docker-Terminologie (1)

Image: statische Vorlage einer Anwendungsumgebung

- Basiert meistens auf anderen Images

Container: lauffähiges virtuelles OS; Instanz eines Images

- Beliebig viele parallele Container eines Images möglich

Registry: Speicherort für Images

- Standard ist Docker Hub, aber auch firmeneigene Registry möglich

Tag: Attribut zur Versionierung von Images

- Aktuellste Version *latest*
- Sonst freie Konvention, z.B. *2.4*

Docker-Terminologie (2)

Docker Engine: Client + Server + API

- Client kommuniziert über ReST-API mit Server

Docker Daemon: Server

- Führt vom Client übergebene Befehle aus
- Verwaltet alle Docker-Objekte, z.B. erstellt er Container aus Images und führt sie aus

Docker CLI: Client

- CLI: Command Line Interface, sozusagen die „Shell“

Repository:

- Lokaler Speicherort für Images aus Registries auf dem Server

Docker ReST-API

GET /containers (über Standard-UNIX-Socket)

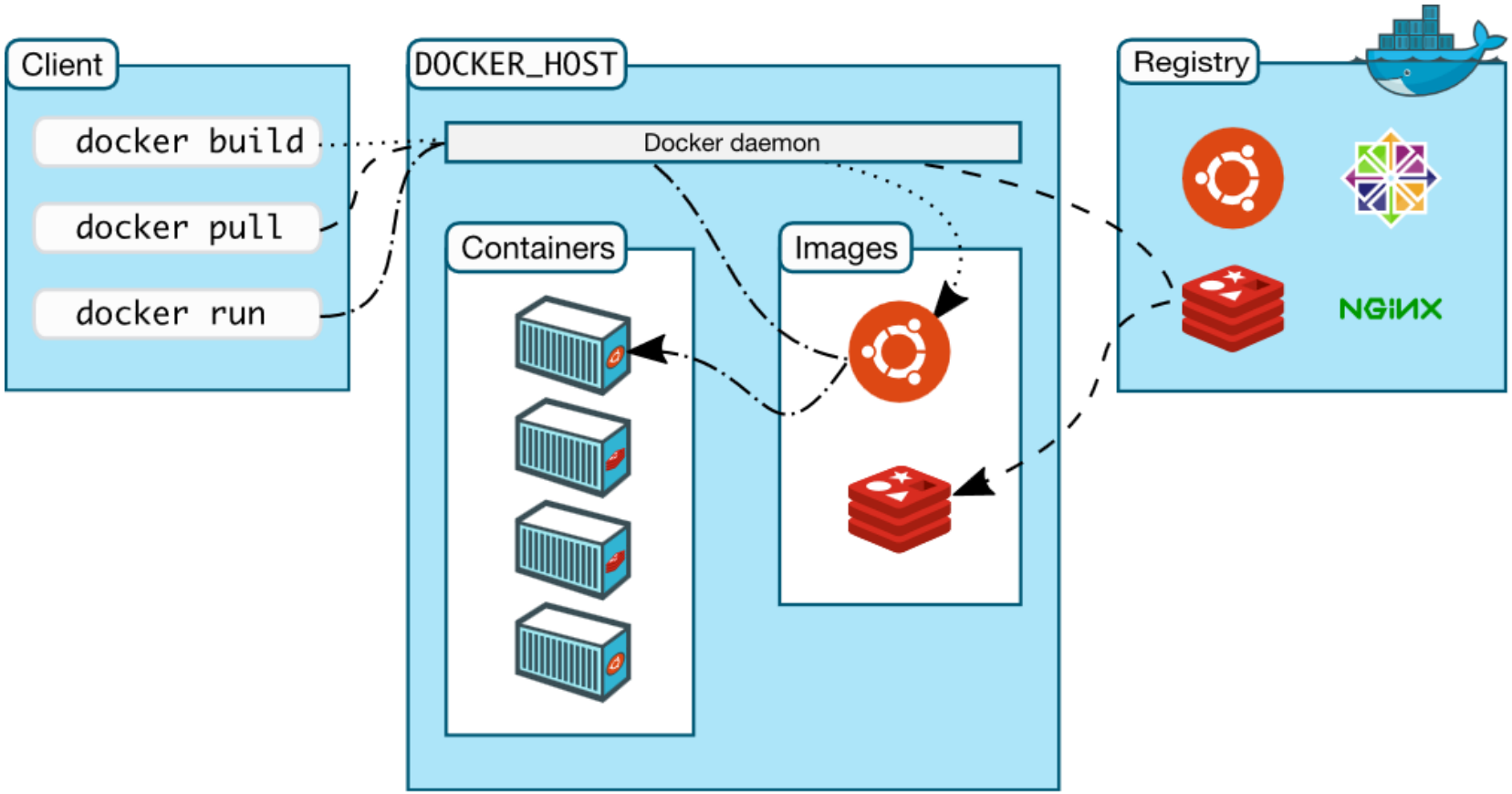
- curl --unix-socket /var/run/docker.sock http:/containers/json

RESPONSE (application/json)


```
[{
        "Id" : "68c759aed4765256d60dc7e6515a040163bd58117bf51f311ef565af8338b069",
        "Names" : ["/pg_tut5"],
        "Image" : "tutorial_pg",
        "ImageID" : "sha256:6b80eef7317b1252abb66ca77aa10a7671aae784b02d27654e7bd6d7486f9c01",
        "Command" : "/bin/sh -c 'service postgresql start \u0026\u0026 tail -f /var/log/postgresql/postgresql-9.5-main.log'",
        "Created" : 1473360753,
        "Ports" : [{
                "PrivatePort" : 5432,
                "Type" : "tcp"
            }, {
                "IP" : "0.0.0.0",
                "PrivatePort" : 80,
                "PublicPort" : 80,
                "Type" : "tcp"
            }
        ],
        "Labels" : {},
        "State" : "running",
        "Status" : "Up 55 minutes"
```


- POST /containers/create?name=rocker erzeugt neuen Container

Beispiel: Kommunikation innerhalb Docker Engine



Quelle: Docker, Inc. [https://docs.docker.com/engine/understanding-docker/]

Docker-Terminologie (3): Namen und IDs

Image ID/ Container ID:

- Ein Hashwert aus 64 alphanumerischen Zeichen (Git)
- Meist wird nur der 12 Zeichen lange *digest* angezeigt)

Image Name:

- Ein Name der Form *registry/repository:tag*
- Identifiziert ein Image eindeutig (wie eine URI)

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	5.7	4b3b6b994512	2 weeks ago	384.5 MB

Container Name:

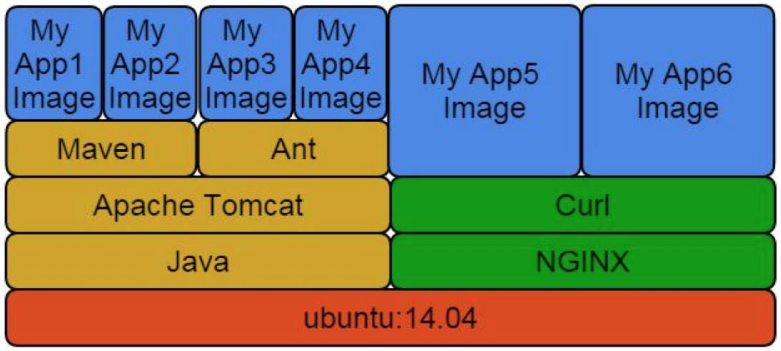
- Ein beim Start des Containers spezifizierbarer String
- Andernfalls ein zufällig generiertes sprechendes Wort („WilliePete“)

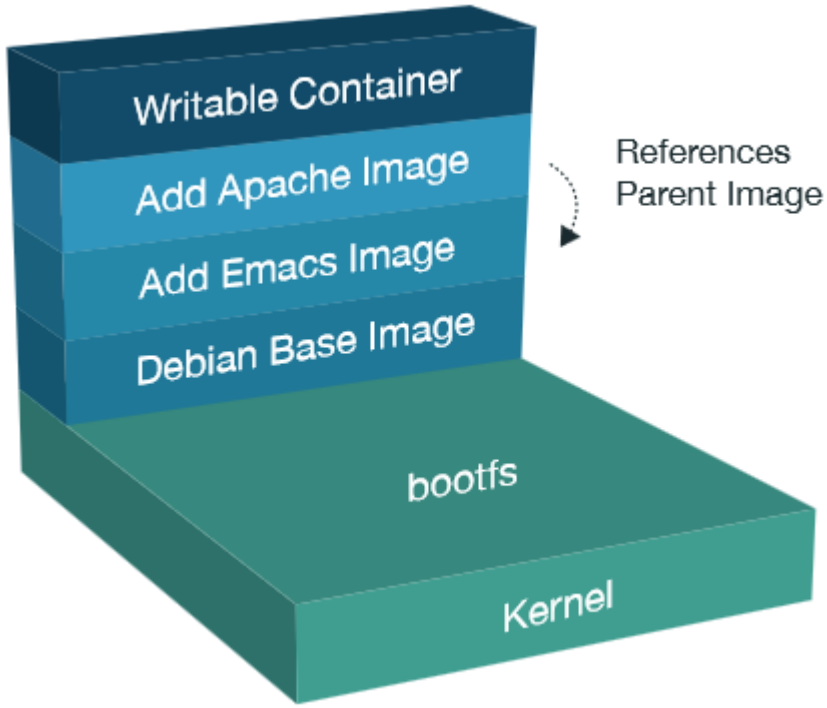
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
68c759aed476	tutorial_pg	/bin/sh -c 'service	5 seconds ago	Up 4 seconds	0.0.0.0:80->80/tcp, 5432/tcp	pg_tut5

Docker Images und Container (1)

Images

- Sind im Wesentlichen Ausschnitte des Dateisystems
- Bauen logisch gesehen aufeinander auf („Stapel“)
- Overlay File System: Dateisystem welches über das Bilden von Schichten funktioniert (sehr leichtgewichtig, keine Redundanz)





Quelle: Docker, Inc. [https://www.docker.com/sites/default/files/WP-%20Definitive%20Guide%20To%20Containers.pdf]

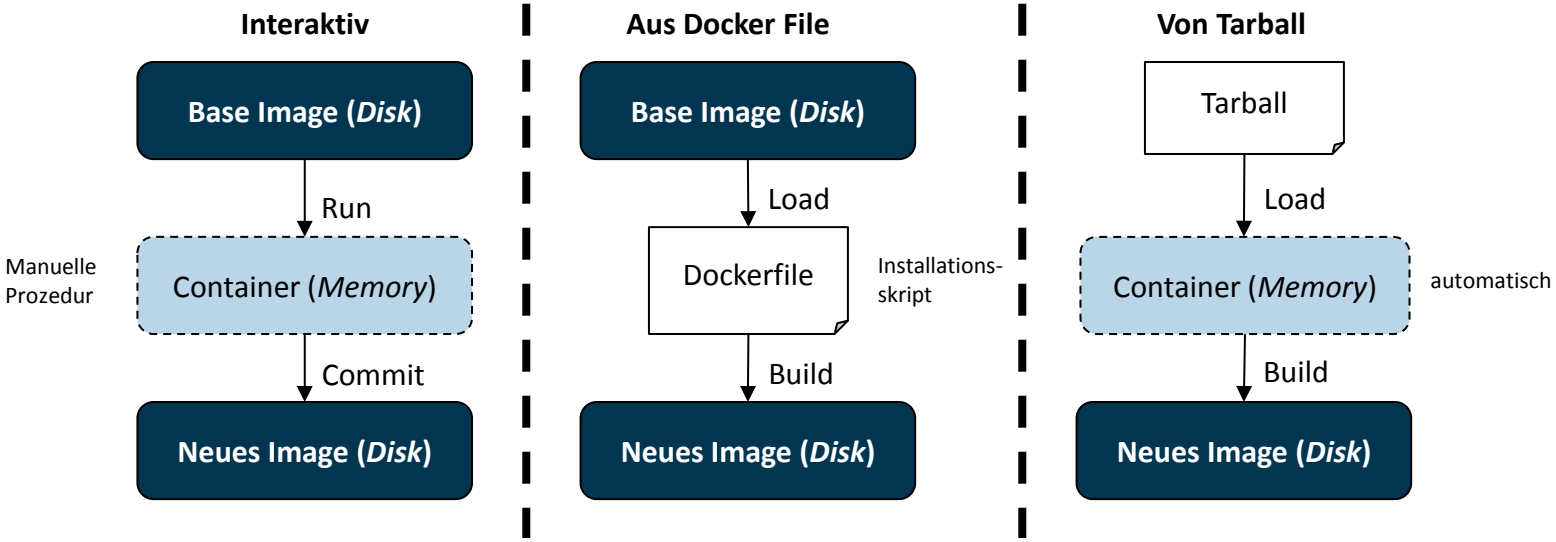
Docker Images und Container (2)

Methoden zum Erzeugen von neuen Images

(1) Änderungen aus einem laufenden Container committen

(2) Mithilfe einer Textdatei (Dockerfile) bauen

(3) Import eines Tarballs in Docker als einzelnes Basisimage



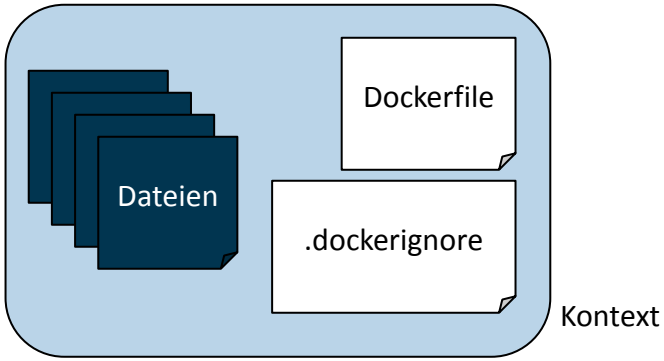
Docker Images und Container (3)

Dockerfile:

- Textdatei mit Befehlen zur Anpassung eines Images an eigene Anforderungen zur automatischen Erstellung eines Images
- „Bauanleitung“ basierend auf domänenspezifische Spracher (DSL)
- Erstellungsprozess ist gut nachvollziehbar

Kontext:

- Menge von Dateien des Verzeichnisses, in welchem das Dockerfile liegt



Ausblick: Docker Tooling und Ökosystem

Docker Compose (Beispiel in der Übung!)

- Komplexe Applikationen, die sich aus mehreren Containern zusammensetzen

Docker Swarm

- Clustert mehrere Docker Host zu einem virtuellen Host

Docker Trusted Registry

- Private (firmeninterne) Docker-Registry

Docker Machine

- Lokales Verwaltungswerkzeug für remote Docker Hosts

Docker Cloud

- Container in der Cloud

Vielen Dank für Ihre Aufmerksamkeit!