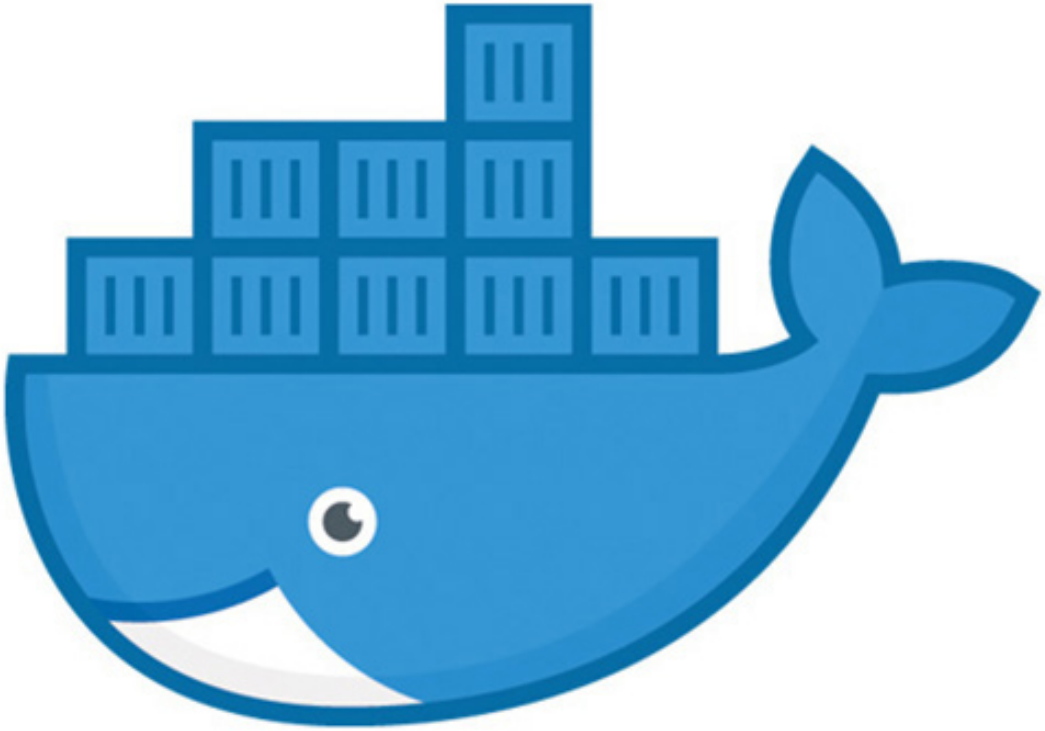
Docker



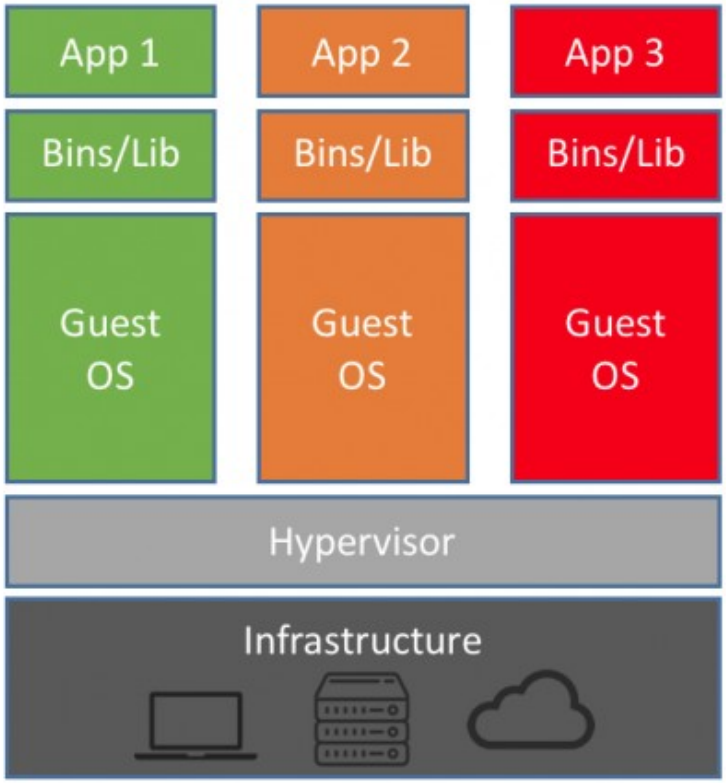
Was ist Virtualisierung?

- Nachbildung eines Hard- oder Software-Objekts durch ein ähnliches Objekt vom selben Typ
- Dadurch lassen sich virtuelle (d. h. nicht-physische) Geräte oder Dienste erzeugen
- Drei Arten:
  - Paravirtualisierung
  - Vollvirtualisierung
  - Containervirtualisierung

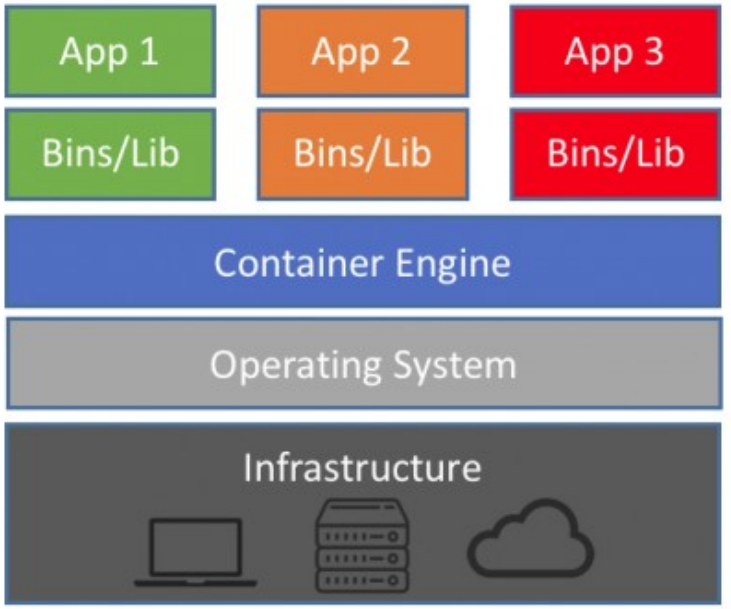
Virtualisierungsarten

- Paravirtualisierung: es wird nur ein Teil virtualisiert. Dazu muss allerdings das Hostsystem angepasst werden
- Vollvirtualisierung: das ganze System (inkl. Kernel) wird virtualisiert
- Containervirtualisierung: Methode um mehrere Instanzen eines System voneinander abgetrennt zu halten

Virtuale Maschine vs Container



Machine Virtualization



Containers

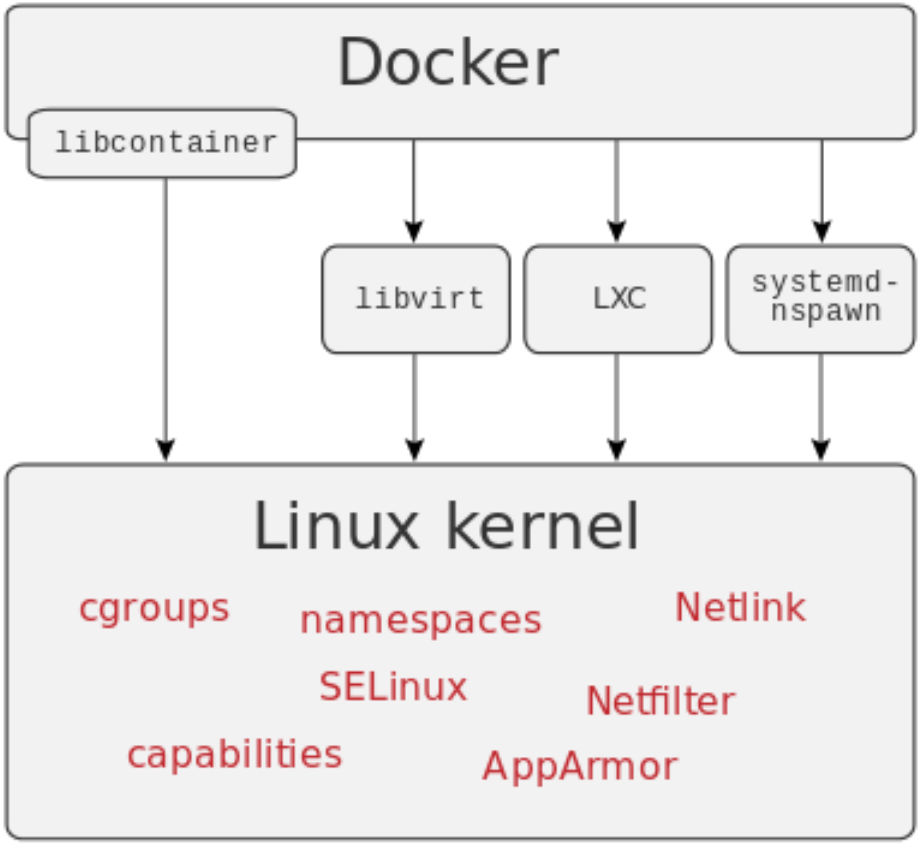
Warum Docker?

- Docker ist ein Containersystem
- Im Gegensatz zu anderen Virtualisierungsarten läuft nur ein System
- Ressoursenschonend (inkl. Speicher)
- Aufteilung – sicherer, überschaubarer, leichter zu managen
- Docker ist die populärste Containerimplementierung

Wie funktioniert Docker?

- Anfänglich wurde die LXC-Schnittstelle des Linuxkernels verwendet
- LXC erzeugt eine virtuelle Umgebung, die zwar ihre eigenen Prozesse besitzt, doch für diese gemeinschaftlich den Kernel des Hostsystems nutzt
- Es wurde eine Programmierschnittstelle namens libcontainer entwickelt (eine Schnittstelle zu den Grundfunktionen von Docker)
- Als Speicher-Backend verwendet Docker das Overlay-Dateisystem AuFS (advanced multi layered unification filesystem)
- AuFS ist ein Overlay-Dateisystem, das zum Schreiben von Daten auf einem nicht beschreibbaren Datenträger verwendet wird
- Neuerdings kann auch Docker btrfs benutzen

Wie funktioniert Docker?



Verwendung von Docker

- Get Docker: https://www.docker.com/community-edition
- Um Docker zu benutzen muss man den entsprechenden Service starten:
  root@host:~$ systemctl start docker
- Um den Service an jedem Systemstart zu starten:
  root@host:~$ systemctl enable docker

Docker-Images

- Sobald der Service läuft müssen wir erstmal mindestens ein Image runterladen:
  root@host:~$ docker pull <image>
- z.B.: *root@host:~$ docker pull debian*
  lädt Debian-Image runter
- Docker holt die Images standartmäßig von dockerhub-Repository:
  https://hub.docker.com/explore/
- Um alle runtergeladene Images aufzulisten verwendet man: *root@host:~$ docker images*

Docker-Container

- Um ein Container zu starten verwendet man:
  root@host:~$ docker run <image>
- Alle laufenden Container auflisten:
  root@host:~$ docker ps
- Alle Container auflisten (inkl. gestoppte):
  root@host:~$ docker ps -a
- Container stoppen:
  root@host:~$ docker stop <container id>
- Container starten:
  root@host:~$ docker start <container id>

Docker-Container

- Zusätzliche Optionen (einige davon):

-t	Pseudo-tty
-d	Detached mode (abgetrennter Modus)
-i	Interactive mode (interaktiver Modus)
-h	Hostname (-h <name>)
-p	Ports zu veröffentlichen (-p <hostport>:<containerqport>)
-v	Use volumes (-v <source on host>:<destination on container>)
--expose	Ports freizugeben (--expose <startport>-<endport>)

Docker-Container

- Um auf einem Container Befehl(e) auszuführen:
  root@host:~$ docker exec <command>
- Container löschen:
  root@host:~$ docker rm <container id>
- Image löschen:
  root@host:~$ docker rmi <image>
- Alle gestoppte Container und hängende Images zu löschen:
  root@host:~$ docker system prune
  (Option -a löscht alles)

Dockerfile

- Dockerfile ist ein Textdokument, das alle Befehle enthält, die ein Benutzer in der Befehlszeile aufrufen kann, um ein Image zu erstellen
- Benutzung: *root@host:~$ docker build .*
  root@host:~$ docker build -f <Pfad> .
  root@host:~$ docker build -t <name> .

spline

Freie Universität Berlin

Dockerfile Reference

FROM <image>	Basisimage
RUN <command>	Befehl ausführen
EXPOSE <port> [<port>/<protocol>]	Ports freigeben
COPY <src> <dest>	Kopiert Datei(en) ins Zielordner im Container
ADD <src> <dest>	Das gleiche wie COPY, aber ermöglicht URL als src anzugeben
ENTRYPOINT <command> <params>	Ermöglicht Ihnen, einen Container zu konfigurieren, der als ausführbare Datei ausgeführt wird.
CMD <command> <params>	Der Hauptzweck eines CMD besteht darin, Standardwerte für einen ausgeführten Container anzugeben. Wird meistens für Entrypoint verwendet. Falls man mehrere CMD-Einträge hat, wird nut der letzte ausgeführt
VOLUME </data>	erstellt einen Einhängepunkt mit dem angegebenen Namen

Dockerfile-Beispiel

```
FROM debian
RUN apt-get update -y && apt-get install
apache2 apache2-utils -y
EXPOSE 80
VOLUME /var/www/html
COPY index.html /var/www/html/
ENTRYPOINT [ "/usr/sbin/apache2ctl" ]
CMD [ "-D", "FOREGROUND" ]
```

- Startet einen Apache2-Container

Image erstellen

- *root@host:~$ docker build -t apache2 .*
  Erstellt ein Image mit dem Namen apache2
- *root@host:~$ docker run -h apache2 -p 80:80 -dti apache2*
  Startet ein Container. Hostname: apache2, Containerport 80 wird nach Hostport 80 weitergeleitet. Abgetrennter Modus
- *root@host:~$ docker inspect $(docker ps | grep apache2 | awk '{print $1}')*
  Gibt detalierte Information über das Container aus