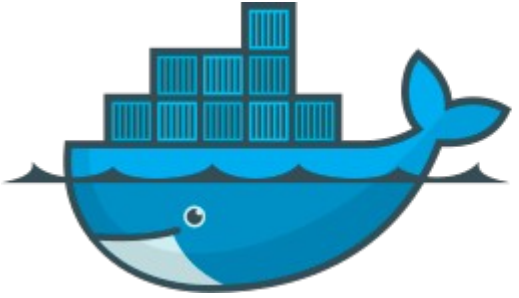
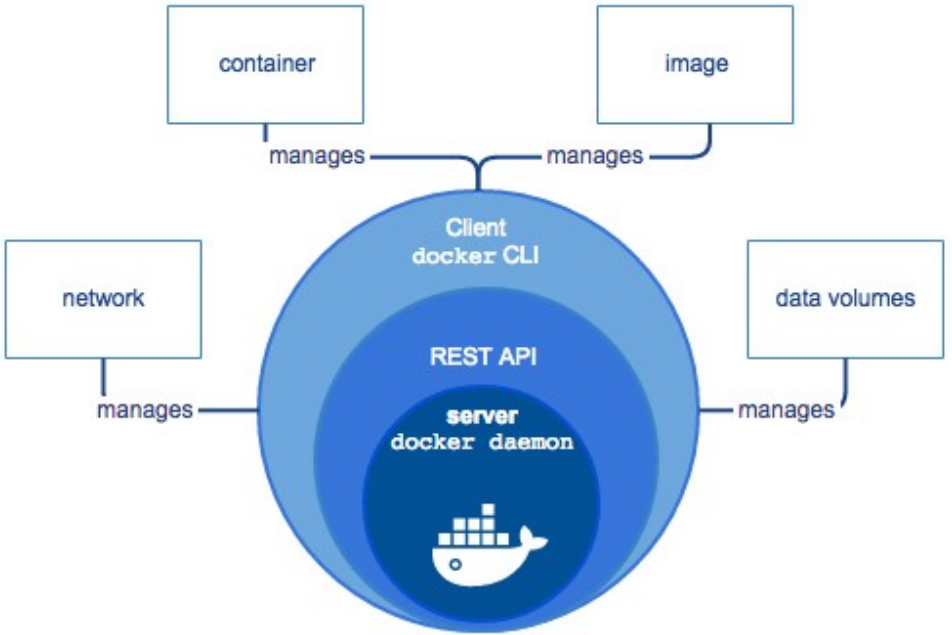
BGD2

Docker: Technische Grundlagen



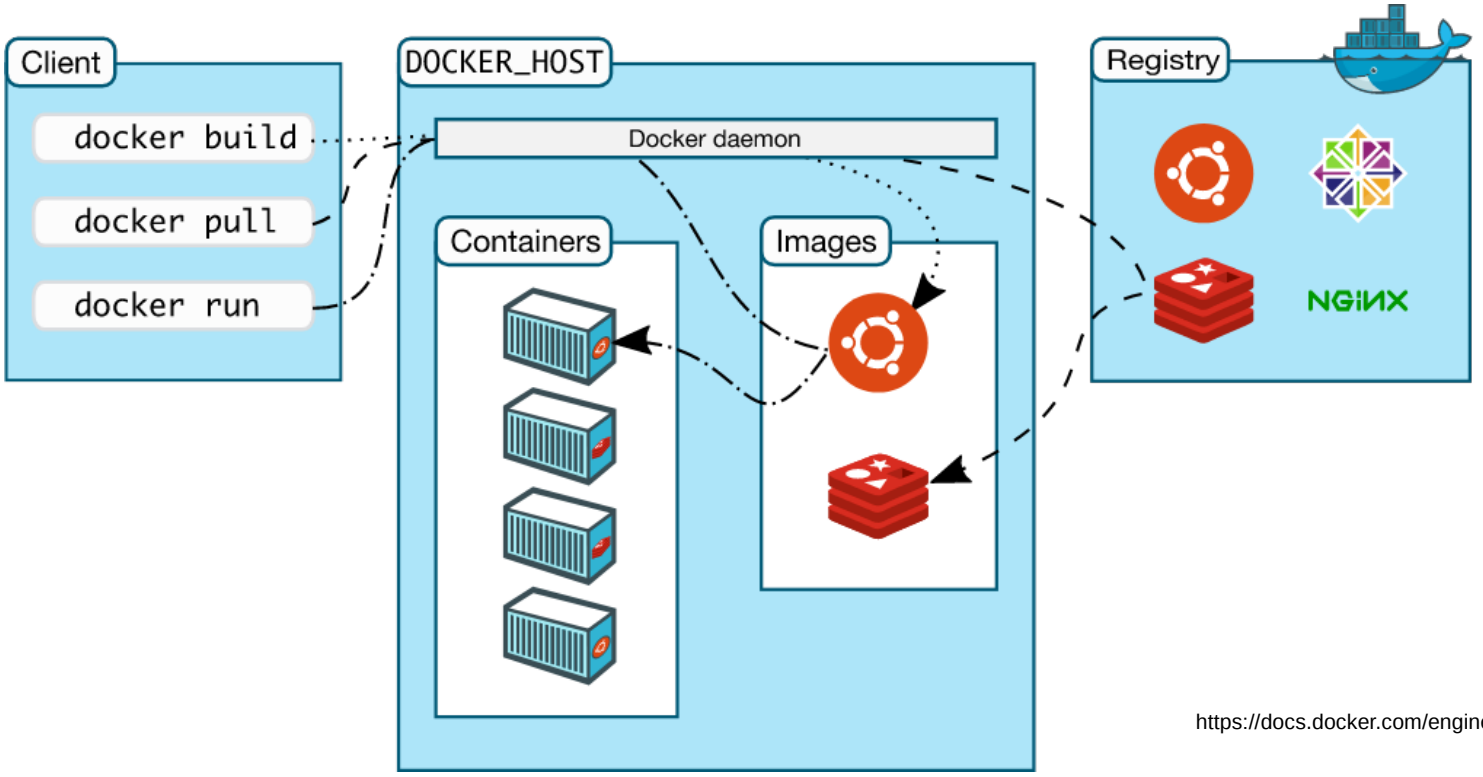
Andreas Scheibenpflug

Docker Engine



https://docs.docker.com/engine/docker-overview/

Docker Architektur



https://docs.docker.com/engine/docker-overview/

Docker Daemon

- Docker Engine
  - Erstellt und verwaltet Container und Images
  - Verwendet `runC` um Images auszuführen
  - Stellt eine REST Schnittstelle zur Verfügung
    - Programmatisches Ansprechen von Docker
    - Wird vom Docker Client verwendet

Docker Client

- Verwendet REST um mit dem Docker Daemon zu kommunizieren
- Frontend für Benutzer
  - Starten von Containern
  - Bauen von Dockerfiles
  - Herunterladen von Images aus der Registry
  - ...
  - `docker --help`

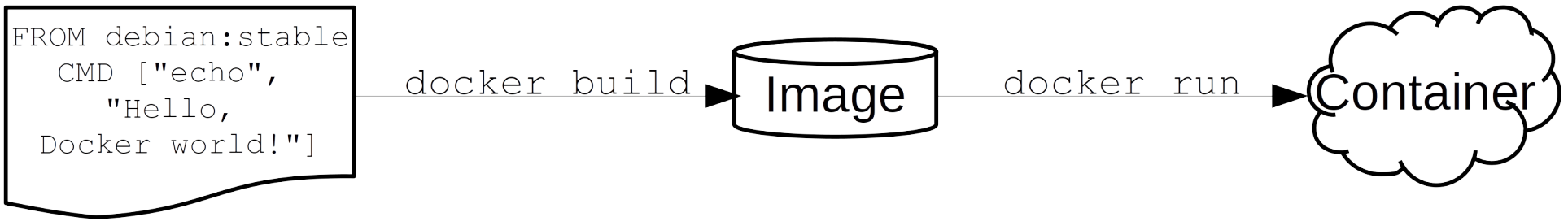
- `docker run debian /bin/echo "Hello, Docker world!"`

Docker Images und Container

- Vorkonfigurierte Laufzeitumgebung
- Technisch: Archive das
  - ein oder mehrere Dateisysteme mit Dateien und
  - Metadaten
- enthält
- `docker run` führt Images aus
- Ein Image zur Laufzeit ist ein Container
- Container sollten als immutable betrachtet werden

Dockerfiles

- Dockerfiles enthalten Instruktionen zum Erstellen von Docker Images
- Der `docker build` Befehl baut aus einem Dockerfile ein Image



Docker registry

- Service um Docker Images zur Verfügung zu stellen
- Offizielle Docker Registry: https://hub.docker.com/
- Eigene Registries: https://hub.docker.com/_/registry/
- Image: Name des Images (z.B. debian:stable)
- Tag: Identifizierer innerhalb eines Repositories (z.B. debian:stable)

Technische Umsetzung unter Linux

- Control Groups (cgroups)
- Namespaces
- Copy-on-write (Union Dateisysteme)

cgroup - Funktionalität

- Steuert welche Ressourcen ein(e) Prozess(gruppe) verwenden kann
- Gruppierung von Prozessen in hierarchische Struktur
  - Monitoring von verbrauchten Ressourcen einer Gruppe
  - Definition von Limits auf eine Gruppe
  - Priorisierung zwischen Gruppen
  - Kontrolle von Gruppen (Erstellen, Löschen)

cgroups - Ressourcen

- Arbeitsspeicher
- Prozessor(en)
- Geräte
- Netzwerk
- Speichergeräte
- ...

cgroup – Memory (1/2)

- Verwaltet memory pages für Gruppen
  - Dateien
  - Heap und Stack
  - Active and inactive Memory
- Limitierung von
  - Prozessspeicher (`docker run --memory`)
  - Kernelspeicher (`docker run --kernel-memory`)
  - Swapspeicher (`docker run --memory-swap`)

cgroup – Memory (2/2)

- Setzen der Out of Memory Exception (OOM) Priority
  - Definiert welche Prozesse zuerst beendet werden wenn der Arbeitsspeicher ausgeht

    (`docker run --oom-score-adj/--oom-kill-disable`)
- Memory Pages können zwischen Prozessen geteilt sein
  - Wird nur einer Gruppe/Prozess „verrechnet“

cgroup - CPU

- Verwaltet für Gruppen von Prozessen
  - Verbrauchte CPU Zeit insgesamt und pro Prozessor
- Erlaubt das Festlegen von
  - CPU shares: Prioritäten, die nur unter Volllast berücksichtigt werden (`docker run --cpu-shares`)
    - Andernfalls haben Gruppen keine Beschränkung an CPU Zeit
  - Bandwidth: Erlaubt Definition einer Obergrenze an CPU Zeit für eine Gruppe (`docker run --cpus`)
    - Wird immer berücksichtigt

Weitere cgroups

- Cpuset: Setzen des Prozessors, auf der die Gruppe ausgeführt werden soll
- Blkio: Limits auf Block Devices (Speichermedien)
- net_cls: Setzen von classid auf Netzwerkpakete
  - Firewall rules, traffic shaping
- net_prio: Prioritäten für Netzwerkschnittstellen
- Device: Lese/Schreibrechte auf Geräte
- Freezer: Stop, Pause, Resume Gruppen

Kernel Namespaces

- Abstrahiert Hostsystem Ressourcen und stellt (virtuelle) Instanzen davon für Prozesse eines Namespaces zur Verfügung
- Änderungen an diesen Ressourcen sind sichtbar für die Prozesses dieses Namespaces, aber unsichtbar für andere Prozesse
- Kurz: Steuert was ein(e) Prozess(gruppe) sehen kann (Isolation)
- Ein Prozess ist in jedem der folgenden Namespaces:
  - IPC: Kommunikation zwischen Prozessen (Message queues, Semaphoren)
  - Network: Netzwerkgeräte, Ports, Stacks,...
  - Mount: Mount Punkte
  - PID: Prozess IDs
  - User: Benutzer und Gruppen IDs
  - UTS: Hostname und Domainname

PID Namespace

- Lokale PIDs in einem Namespace
- Ein Prozess kann mehrere PIDs haben
  - Prozess hat andere PID in Container als aus Sicht des Hostsystems
- Verschachtelte Namespaces
  - Namespace sieht Prozesse eines Kindnamespaces
  - Namespace sieht keine Prozesse des Elternnamespaces

Network Namespace

- Namespace erhält eigene Netzwerkressourcen
  - Eigene Netzwerkschnittstellen
  - IP Protocol Stack
  - Sockets
  - Routing tables
  - Iptables rules
- Physische Netzwerkgeräte können nur in einem Namespace leben
- Virtuelle Netzwerkgeräte können verwendet werden um
  - Tunnel zwischen Namespaces zu errichten
  - Brücken (Bridge) zu physischen Geräten herzustellen

Mount Namespace

- Isolation von Mount Points für einen Namespace
- Eigenes Dateisystem für einen Namespace (ähnlich chroot)

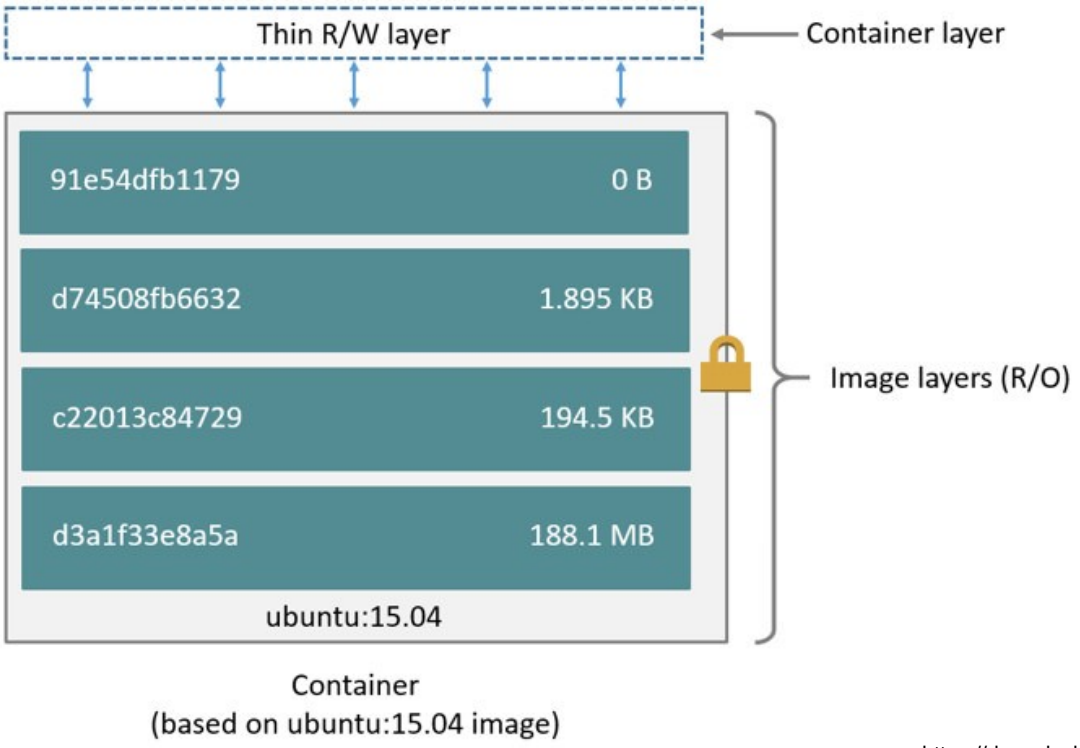
User Namespace

- Isolation von User und Gruppen für Namespaces
- Eigene UID/GUID im Namespace und Übersetzung zum Hostsystem
- Vererbbar auf Kindnamespaces (analog PID)
  - Kindnamespace hat die selben Rechte wie Elternnamespace
- Wenn ein nicht-priviligierter Benutzer einen Namespace anlegt und darin zum root Nutzer wechselt, ist er noch immer ein nicht-priviligierter Nutzer

Copy-on-write storage

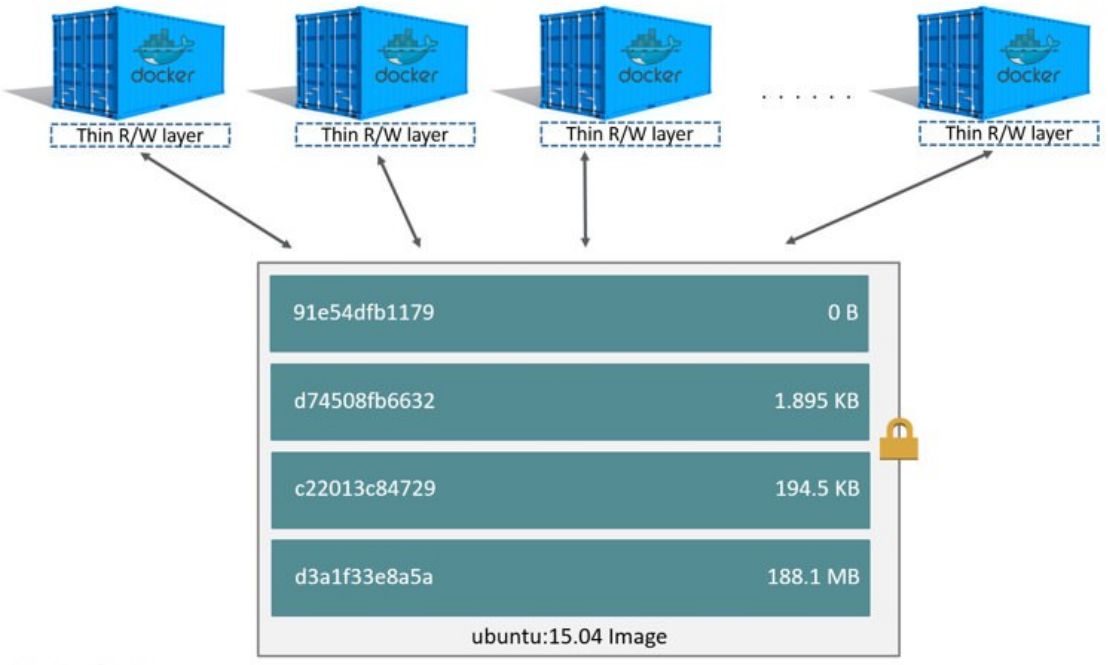
- Union Dateisystem: Ermöglicht das Mounten und Übereinanderlegen mehrerer Dateisysteme
- Docker: Überlagern mehrerer Layer eines Docker Images
- Leseoperationen werden auf den bestehenden Layern durchgeführt
- Schreiboperationen werden auf einem neuen Layer ausgeführt (zuvor wird die zu schreibende Datei kopiert → Copy-on-write)
- Docker bietet unterschiedliche Storagedriver
  - OverlayFS ist der Standard unter Ubuntu

Layer bei Images



https://docs.docker.com/storage/storagedriver/

Layer bei Containern



https://docs.docker.com/storage/storagedriver/

`docker build` und Caching

- `docker build` erzeugt für jeden Befehl im Dockerfile (z.B. `RUN`) einen neuen Layer
- Bei Dateioperationen (z.B. `COPY`) wird die Checksumme der Dateien gespeichert
- Alle Layer werden gecacht
  - Beschleunigt Build Prozess
  - Ermöglicht das Teilen von Layern zwischen unterschiedlichen Images
  - Beschleunigt Image Downloads
  - Reduziert die Größe von Containern und Images

Literatur

- Cgroups, namespaces, and beyond: what are containers made from?
  - https://www.youtube.com/watch?v=sK5i-N34im8
- Man page zu Namespaces
  - http://man7.org/linux/man-pages/man7/namespaces.7.html
- Man page zu cgroups
  - http://man7.org/linux/man-pages/man7/cgroups.7.html
- LWN article: Namespaces in operation
  - https://lwn.net/Articles/531114/
- LWN article: Control Groups
  - https://lwn.net/Articles/603762/
- Docker documentation
  - https://docs.docker.com/engine/docker-overview
- Adrian Mouat, Using Docker, O‘Reilly