TSBE Bern
Höhere Fachschule

Virtuelle Systeme – Docker

FS-2018

Christoph Bühlmann

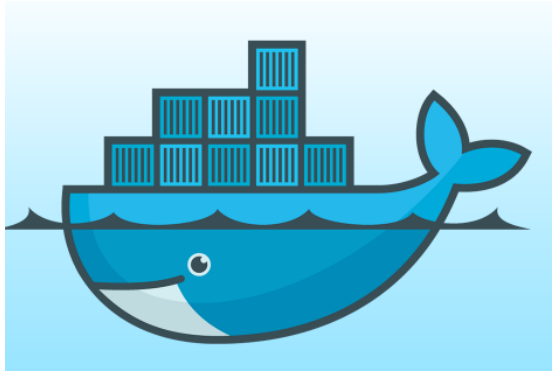
Einleitung – Was ist Docker

Docker ist eine Open-Source-Software, die dazu verwendet werden kann, Anwendungen mithilfe von Betriebssystemvirtualisierung in Containern zu isolieren. Dies vereinfacht einerseits die Bereitstellung von Anwendungen, weil sich Container, die alle nötigen Pakete enthalten, leicht als Dateien transportieren und installieren lassen. Andererseits gewährleisten Container die Trennung der auf einem Rechner genutzten Ressourcen, sodass ein Container keinen Zugriff auf Ressourcen anderer Container hat. [wikipedia.org]

Einleitung – Was ist Docker

docker [naut.]:

der Dockarbeiter, der Hafenarbeiter [dict.leo.org]



- Docker stellt einen einheitlichen Wrapper um SW-Pakete bereit «Build, Ship and Run Any App, Anywhere» [www.docker.com]

  Ganz im Sinne von Schiffscontainern: Der Container ist immer der gleiche, wobei der Inhalt variiert.

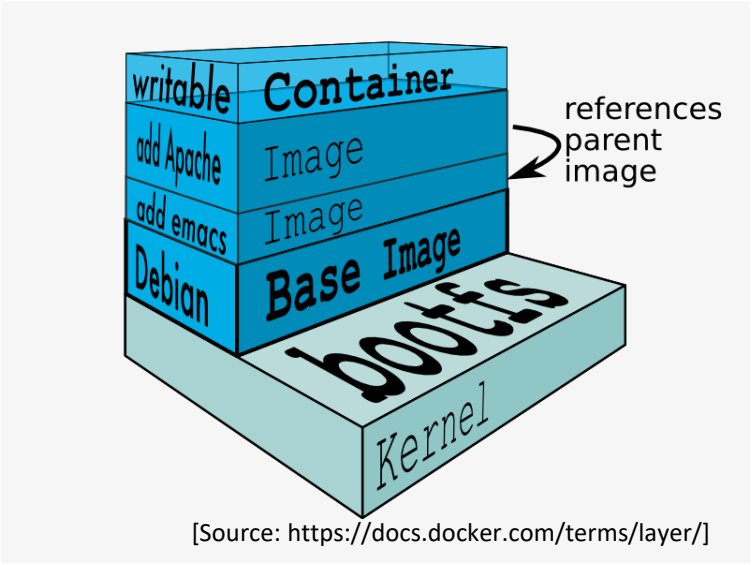
Technologie

Laufzeit

- Plattform Virtualisierung mit libvirt
- LXC (LinuX Containers) als Framework um mehrere isolierte Linux-Instanzen (Container) auf dem selben Host laufen zu lassen.

Filesystem

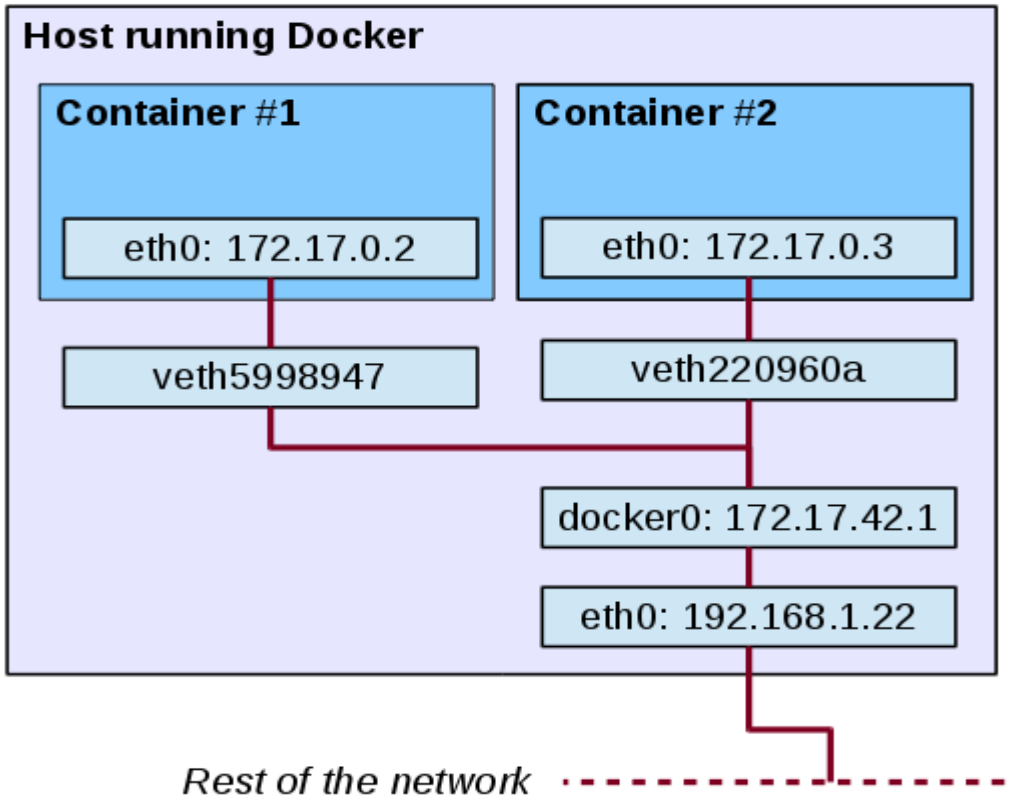
- Hierarchisch organisiert
- Images leiten von anderen Images ab, der Container ist die Laufzeit-Repräsentation der app.



[Source: https://docs.docker.com/terms/layer/]

Netzwerk (minimal)

- Nur gelinkte Ports werden an den Host weitergeroutet
- Container-Links haben keine Restriktionen
- Für PRO's: https://docs.docker.com/engine/userguide/networking/



Docker – was wird benötigt

Laufzeit

- Installierte Docker-Plattform

Entwicklung

- Installierte Docker-Plattform
- Dockerfile als Beschreibung des entwickelten Images
- Zur Verteilung allenfalls private Docker-Registry oder Account im Docker Hub

Dockerfile - FROM

- FROM leitet von einem definierten Image ab

```
FROM <image>
```

Or

```
FROM <image>:<tag>
```

Or

```
FROM <image>@<digest>
```

```
Ex.
FROM stackbrew/debian:jessie
```

- Stackbrew sind offizielle Images von Docker, einsehbar in https://hub.docker.com/u/stackbrew/

Dockerfile - RUN

- RUN führt eine Instruktion zu Build-Time aus

- `RUN <command>` (*shell* form, the command is run in a shell, which by default is `/bin/sh -c` on Linux or `cmd /S /C` on Windows)
- `RUN ["executable", "param1", "param2"]` (*exec* form)

```
Ex.
RUN apt-get -q update
```

- Build bedeutet, jedes RUN-Statement wird ausgeführt
- Und das resultierende File-System als Image abgelegt

Dockerfile - ADD

- ADD kopiert ein lokales File in das Image

ADD has two forms:

- `ADD <src>... <dest>`
- `ADD ["<src>",... "<dest>"]` (this form is required for paths containing whitespace)

Ex.

```
ADD index.html /var/www/html/
```

Dockerfile - EXPOSE

- EXPOSE informiert Docker, dass von diesem Image ein Port exponiert wird

```
EXPOSE <port> [<port>...]
```

```
Ex.
EXPOSE 80
```

- Mit EXPOSE wird auf dem Hostsystem noch nichts gemacht
- Der Port muss schlussendlich mit der –p Option geroutet werden

```
Docker run –p <port auf Host>:<port in Docker>
```

Dockerfile - CMD

- CMD wird bei docker run ausgeführt

The `CMD` instruction has three forms:

- `CMD ["executable","param1","param2"]` (*exec* form, this is the preferred form)
- `CMD ["param1","param2"]` (as *default parameters to ENTRYPOINT*)
- `CMD command param1 param2` (*shell* form)

Ex.

```
CMD ["nginx", "-g", "daemon off;" ]
```

- ACHTUNG ein CMD pro Dockerfile (Konvention)
- Es wird nur das LETZTE CMD im Dockerfile ausgeführt

Dockerfile Beispiel

```
FROM stackbrew/debian:jessie
MAINTAINER Christoph Bühlmann
<christoph.buehlmann@gibb.ch>;
RUN apt-get -q update
RUN apt-get -qy install nginx
ADD index.html /var/www/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;" ]
```

- git clone https://github.com/chbuehlmann/docker_uebung.git
- git checkout develop
- cd nginx
- docker build -t meindockerimage .
- docker run -d --name=irgendeinname -p 8080:80 meindockerimage

Fragen

Hands On – Docker

Installation

- Ubuntu-VM
- Docker installieren: wget -qO- https://get.docker.com/ | sh oder apt-get install docker.io
- User berechtigen: sudo usermod -aG docker [deinUser]

Dockers

- Guacamole vorbereiten http://guac-dev.org/doc/gug/guacamole-docker.html
  - guacd zum Rendern: docker run --name some-guacd -d glyptodon/guacd
  - SQL-Scripts erstellen: docker run --rm glyptodon/guacamole /opt/guacamole/bin/initdb.sh --mysql > initdb.sql
- MySQL für die Userverwaltung:
  - docker run --name mysqldb -e MYSQL_USER=guacamole_user -e MYSQL_PASSWORD=Start1234 -e MYSQL_DATABASE=guacamole_db -e MYSQL_ROOT_PASSWORD=Start1234 -d mysql
  - PHPMyadmin: docker run -d --link mysqldb:mysql -e MYSQL_USERNAME=root --name phpmyadmin -p 8000:80 corbinu/docker-phpmyadmin
  - db-sql Scripts einspielen (via php-myadmin)
- Guacamole starten:
  - docker run --name guacamole --link some-guacd:guacd \
    --link mysqldb:mysql \
    -e MYSQL_DATABASE=guacamole_db \
    -e MYSQL_USER=guacamole_user \
    -e MYSQL_PASSWORD=Start1234 \
    -d -p 8080:8080 glyptodon/guacamole .
  - http://[IhreIP]:8080/guacamole -> user guacadmin pw guacadmin
  - Verbindung hinzufügen in Einstellungen -> Protokoll rdp, port 3389, jedes Sicherheitsverfahren, Server Zertifikat ignorieren
- Schreiben Sie ihren eigenen Container