

Docker Grundlagen Training (2 days)

Agenda

1. Grundlagen

- [Übersicht Architektur](#)
- [Was ist ein Container](#)
- [Container vs. Virtuelle Maschine](#)

2. Installation

- [Installation Docker unter Ubuntu mit snap](#)

3. Docker-Befehle

- [Die wichtigsten Befehle](#)
- [docker run](#)
- [Docker container/image stoppen/löschen](#)
- [Docker containerliste anzeigen](#)
- [Docker container analysieren](#)
- [Nginx mit portfreigabe laufen lassen](#)

4. Dockerfile - Examples

- [Ubuntu mit ping](#)
- [Nginx mit content aus html-ordner](#)
- [ssh server](#)

5. Daten persistent machen / Shared Volumes

- [Überblick](#)
- [Volumes](#)

6. Netzwerk

- [Netzwerk](#)

7. Docker Compose

- [Example with Wordpress / Nginx / MariadB](#)

8. Tipps & Tricks

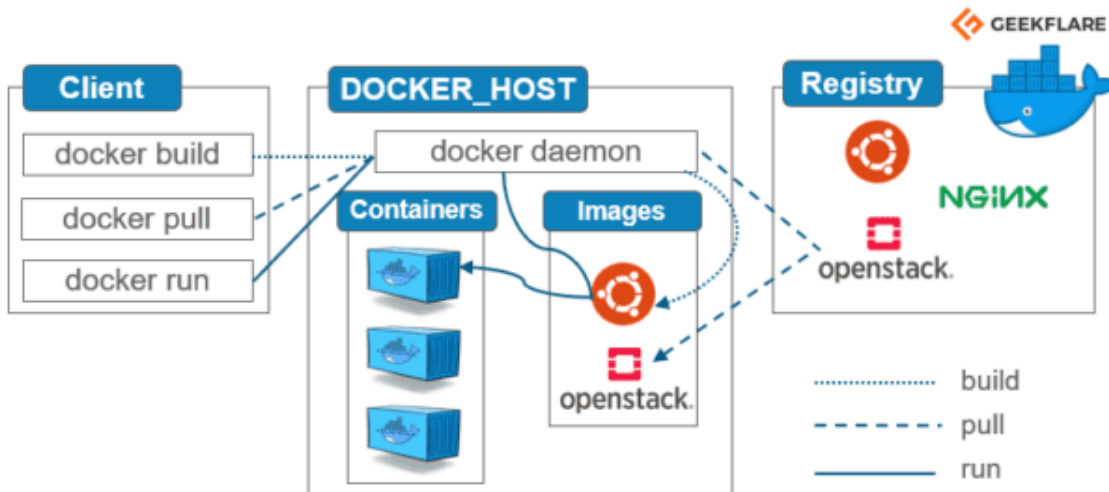
- [Auf ubuntu root-benutzer werden](#)
- [IP - Adresse abfragen](#)
- [Hostname setzen](#)
- [Proxy für Docker setzen](#)

9. Documentation

- [Vulnerability Scanner with docker](#)
- [Vulnerability Scanner mit snyk](#)

Grundlagen

Übersicht Architektur



Was ist ein Container

- vereint in sich Software
 - Bibliotheken
 - Tools
 - Konfigurationsdateien
 - keinen eigenen Kernel
 - gut zum Ausführen von Anwendungen auf verschiedenen Umgebungen
-
- Container sind entkoppelt
 - Container sind voneinander unabhängig
 - Können über wohldefinierte Kommunikationskanäle untereinander Informationen austauschen
-
- Durch Entkopplung von Containern:
 - o Unverträglichkeiten von Bibliotheken, Tools oder Datenbank können umgangen werden, wenn diese von den Applikationen in unterschiedlichen Versionen benötigt werden.

Container vs. Virtuelle Maschine

VM's virtualisieren Hardware

Container virtualisieren Betriebssystem

Installation

Installation Docker unter Ubuntu mit snap

```
snap install docker

## for information retrieval
snap info docker
systemctl list-units
systemctl list-units -t service
systemctl list-units -t service | grep docker

systemctl status snap.docker.dockerd.service
## oder (aber veraltet)
service snap.docker.dockerd status

systemctl stop snap.docker.dockerd.service
systemctl status snap.docker.dockerd.service
systemctl start snap.docker.dockerd.service
```

Docker-Befehle

Die wichtigsten Befehle

```
docker run <image>
## z.b. // Zieht das image aus docker hub
## hub.docker.com
docker run hello-world

## images die lokal vorhanden
docker images

## container (laufende)
docker container ls
## container (vorhanden, aber beendet)
docker container ls -a

## z.b hilfe für docker run
docker help run

## docker hub durchsuchen
docker search hello-world
```

docker run

Beispiel (binden an ein terminal), detached

```
## before that we did
docker pull ubuntu:xenial
docker run -t -d --name my_xenial ubuntu:xenial
## will wollen überprüfen, ob der container läuft
docker container ls
## image vorhanden
docker images

## in den Container reinwechsel
docker exec -it my_xenial bash
docker exec -it my_xenial cat /etc/issue
##
```

Docker container/image stoppen/löschen

```
docker stop ubuntu-container
## Kill it if it cannot be stopped -be careful
docker kill ubuntu-container

docker rm ubuntu-container

## oder alternative
docker rm -f ubuntu-container

## image löschen
docker rmi ubuntu:xenial
```


Docker containerliste anzeigen

```
## besser
docker container ls
docker container ls -a

## deprecated
docker ps
## -a auch solche die nicht mehr laufen
docker ps -a
```

Docker container analysieren

```
docker inspect hello-web # hello-web = container name
```

Nginx mit portfreigabe laufen lassen

```
docker run --name test-nginx -d -p 8080:80 nginx

docker container ls
lsof -i
cat /etc/services | grep 8080
curl http://localhost:8080
docker container ls
## wenn der container gestoppt wird, keine ausgabe mehr, weil kein webserver
docker stop test-nginx
curl http://localhost:8080
```

Dockerfile - Examples

Ubuntu mit ping

```
mkdir myubuntu
cd myubuntu/

## nano Dockerfile
FROM ubuntu:latest
RUN apt-get update; apt-get install -y inetutils-ping
CMD ["/bin/bash"]

docker build -t myubuntu .
docker images
docker run -d -t --name container-ubuntu myubuntu
docker container ls
## in den container reingehen mit dem namen des Containers: myubuntu
docker exec -it myubuntu bash
ls -la
```

Nginx mit content aus html-ordner

```
cd
mkdir nginx-test
cd nginx-test
mkdir html
cd html/
## vi index.html
Text, den du rein haben möchtest

cd ..
vi Dockerfile

FROM nginx:latest
COPY html /usr/share/nginx/html

docker build -t dockertrainereu/jml-hello-web .
docker images
## eventually you are not logged in
docker login
docker push dockertrainereu/jml-hello-web
##aus spass geloescht
docker rmi dockertrainereu/jml-hello-web
## und direkt aus der Registry wieder runterladen
docker run --name hello-web -p 8888:80 -d dockertrainereu/jml-hello-web

##
docker rm -f hello-web
```

ssh server

```
cd
mkdir devubuntu
cd devubuntu
## vi Dockerfile
```

```
FROM ubuntu:latest
```

```
RUN apt-get update && \
    DEBIAN_FRONTEND="noninteractive" apt-get install -y inetutils-ping openssh-server
&& \
    rm -rf /var/lib/apt/lists/*
```

```
RUN mkdir /run/sshd && \
    echo 'root:root' | chpasswd && \
    sed -ri 's/^#?PermitRootLogin\s+.*?PermitRootLogin yes/' /etc/ssh/sshd_config && \
    sed -ri 's/UsePAM yes/#UsePAM yes/g' /etc/ssh/sshd_config && \
    mkdir /root/.ssh
```

```
EXPOSE 22/tcp
```

```
CMD ["/usr/sbin/sshd", "-D"]
```

```
docker build -t devubuntu .
docker run --name=devjoy -p 2222:22 -d -t devubuntu3
```

```
ssh root@localhost -p 2222
## example, if your docker host ist 192.168.56.101 v
ssh root@192.168.56.101 -p 2222
```

Daten persistent machen / Shared Volumes

Überblick

Overview

```
bind-mount  # not recommended
volumes
tmpfs
```

Disadvantages

```
stored only on one node
Does not work well in cluster
```

Alternative for cluster

```
glusterfs
cephfs
nfs

## Stichwort
ReadWriteMany
```

Volumes

Storage volumes verwalten

```
docker volume ls
docker volume create test-vol
docker volume ls
docker volume inspect test-vol
```

Storage volumes in container einhängen

```
docker run -it --name=container-test-vol --mount target=/test_data,source=test-vol
ubuntu bash
1234ad# touch /test_data/README
exit
## stops container

## create new container and check for /test_data/README
docker run -it --name=container-test-vol2 --mount target=/test_data,source=test-vol
ubuntu bash
ab45# ls -la /test_data/README
```

Storage volume löschen

```
## Zunächst container löschen
docker rm container-test-vol
docker rm container-test-vol2
docker volume rm test-vol
```


Netzwerk

Netzwerk

Übersicht

```
3 Typen

o none
o bridge (Standard-Netzwerk)
o host

### Additionally possible to install
o overlay (needed for multi-node)
```

Kommandos

```
## Netzwerk anzeigen
docker network ls

## bridge netzwerk anschauen
## Zeigt auch ip der docker container an
docker inspect bridge

## im container sehen wir es auch
docker inspect ubuntu-container
```

Eigenes Netz erstellen

```
docker network create -d bridge test_net
docker network ls

docker container run -d --name nginx --network test_net nginx
docker container run -d --name nginx_no_net --network none nginx

docker network inspect none
docker network inspect test_net

docker inspect nginx
docker inspect nginx_no_net
```

Netzwerk rausnehmen / hinzufügen

```
docker network disconnect none nginx_no_net
docker network connect test_net nginx_no_net

### löschen erst möglich, wenn es keine Endpoints
### d.h. container die das Netzwerk verwenden
```

Docker Compose

Example with Wordpress / Nginx / MariadB

```
mkdir wordpress-mit-docker-compose
cd wordpress-mit-docker-compose
## nano docker-compose.yml
version: "3.7"

services:
  database:
    image: mysql:5.7
    volumes:
      - database_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: mypassword
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    image: wordpress:latest
    depends_on:
      - database
    ports:
      - 8080:80
    restart: always
    environment:
      WORDPRESS_DB_HOST: database:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
    volumes:
      - wordpress_plugins:/var/www/html/wp-content/plugins
      - wordpress_themes:/var/www/html/wp-content/themes
      - wordpress_uploads:/var/www/html/wp-content/uploads

volumes:
  database_data:
  wordpress_plugins:
  wordpress_themes:
  wordpress_uploads:

### now start the system
docker-compose up -d
### we can do some test if db is reachable
docker exec -it wordpress_compose_wordpress_1 bash
### within shell do
apt update
apt-get install -y telnet
## this should work
```

```
telnet database 3306
```

```
## and we even have logs
```

```
docker-compose logs
```

Tipps & Tricks

Auf ubuntu root-benutzer werden

```
## kurs>  
sudo su -  
## password von kurs eingegeben  
## wenn wir vorher der benutzer kurs waren
```

IP - Adresse abfragen

```
## IP-Adresse abfragen  
ip a
```

Hostname setzen

```
## als root
hostnamectl set-hostname server.training.local
## damit ist auch sichtbar im prompt
su -
```

Proxy für Docker setzen

Walktrough

```
## as root
systemctl list-units -t service | grep docker
systemctl cat snap.docker.dockerd.service
systemctl edit snap.docker.dockerd.service
## in edit folgendes reinschreiben
[Service]
Environment="HTTP_PROXY=http://user01:password@10.10.10.10:8080/"
Environment="HTTPS_PROXY=https://user01:password@10.10.10.10:8080/"
Environment="NO_PROXY= hostname.example.com,172.10.10.10"

systemctl show snap.docker.dockerd.service --property Environment
systemctl restart snap.docker.dockerd.service
systemctl cat snap.docker.dockerd.service
cd /etc/systemd/system/snap.docker.dockerd.service.d/
ls -la
cat override.conf
```

Ref

- <https://www.thegeekdiary.com/how-to-configure-docker-to-use-proxy/>

Documentation

Vulnerability Scanner with docker

- <https://docs.docker.com/engine/scan/#prerequisites>

Vulnerability Scanner mit snyk

- <https://snyk.io/plans/>