Abteilung für INFORMATIK

HTL ST. PÖLTEN

Titel: Labor 11 - Docker

Klasse: 3BHIF

Name: Haiden

Gruppe: 01

Aufgabe: 01.04.2020 Abgabe: 29.04.2020

Inhaltsverzeichnis

1 Labor 11 – Docker 1
1.1 Was bringt Docker? 1
1.2 Wie arbeitet Docker? 1
1.3 Installation von Docker unter Debian 1
1.4 Prüfen der Installation 2
1.5 Was ist der Unterschied zwischen einem Container und einem Image? 3
1.6 Wie und von wo können Images heruntergeladen werden? 3
1.7 Wie können verfügbare Images angezeigt werden? 3
1.8 Was ist eine BusyBox? Probiere die beiden folgenden Varianten zum Start einer Busybox. Worin unterscheiden sich diese? 3
1.9 Wie können laufende/beendete Container angezeigt werden? 4
1.10 Wie können beendete Container „aufgeräumt werden“ 4
1.11 Lade und starte einen nginx Webserver. Der Webserver soll unter http://localhost:5000 erreichbar sein. 4
1.12 Was ist der Detached-Mode? Wozu wird/kann er verwendet werden? 5
1.13 Starte einen neuen Container und zeige eine personalisierte Index Seite an. 5
1.14 Installation von Docker-Compose (https://github.com/docker/compose/releases/) 6
1.15 Wozu dient Docker-Compose? 6

1 Labor 11 – Docker

1.1 Was bringt Docker?

Da Docker mit dem Prinzip von Containern arbeitet und diese nur die essenziellen Dateien für eine Applikation, z.B. nur die essenziellen Dateien für einen Datenbank-Server beinhalten und kein ganzes OS wie in einer virtuellen Maschine „mitgeschleppt“ wird, sind Docker Container extrem leichtgewichtig im Vergleich zu virtuellen Maschinen. Es lassen sich in Sekunden Bruchteilen Container starten sowie stoppen. Die geringere Auslastung durch Container auf einem Host-System bedeutet, dass man mehr Container auf dichteren Raum packen kann und so die Auslastung eines Host-System deutlich dichter ist und man so kosten einspart.

1.2 Wie arbeitet Docker?

Docker arbeitet nach dem Prinzip von Containern. Container sind kleine abgeschottete Bereiche, in denen nur die nötigsten Dateien und Programme für die Ausführung eines ganz bestimmen Services liegen. Docker Container setzen auf dem Betriebssystem auf und müssen so nicht wie in virtuellen Maschinen ein eigenes OS mitbringen, was bedeutet dass Container um einiges leichtgewichtiger an Ressourcen sind und so kosten einsparen. Die Container werden von dem Docker Daemon verwaltet. Er dient zum Starten, Stoppen, Herunterladen von neuen Images vom Docker Hub usw…. Er ist das zentrale Gehirn von Docker. Im Docker Hub, einem „Appstore“ für Docker-Apps, findet man allerhand Programme, die sich mit einem simplen Kommando in einem Container starten lassen.

Im Gegensatz zu VMs muss man bei Docker die Abhängigkeiten, die ein Service haben kann, nicht extra installieren, da ein Docker Image, welches vom Docker Hub gezogen wurde, alle Abhängigkeiten bzw. Dependencies in einem zusammengepackten Image mitbringt. So lassen sich Docker-Container auf mehreren, unterschiedlichen Maschinen problemlos ausführen ohne sich über Abhängigkeiten oder ähnlichem scheren zu müssen.

1.3 Installation von Docker unter Debian

Um Docker zu installieren, installiert man ein paar nötige Pakete, die man braucht, um Repos hinzuzufügen:

```
root@debian10VM:/home/schueler# sudo apt -y install apt-transport-
https ca-certificates curl gnupg2 software-properties-common

Paketlisten werden gelesen... Fertig

Abhängigkeitsbaum wird aufgebaut.

Statusinformationen werden eingelesen.... Fertig

ca-certificates ist schon die neueste Version (20190110).

ca-certificates wurde als manuell installiert festgelegt.
```

Danach wird der Docker GPG Key hinzugefügt:

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-
key add -
```

Danach das Repo:

```
sudo add-apt-repository \ "deb [arch=amd64]
https://download.docker.com/linux/debian \ $(lsb_release -cs) \
stable"
```

Updaten der Paketquellen mit apt update und dann installieren der Pakete:

```
sudo apt -y install docker-ce docker-ce-cli containerd.io
```

1.4 Prüfen der Installation

```
root@debian10VM:/home/schueler# docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 1
 Server Version: 19.03.8
 Storage Driver: overlay2
  Backing Filesystem: <unknown>
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local
logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
 init version: fec3683
```

```
 Security Options:
  apparmor
  seccomp
   Profile: default
 Kernel Version: 4.19.0-8-amd64
 Operating System: Debian GNU/Linux 10 (buster)
 OSType: linux
 Architecture: x86_64
 CPUs: 6
 Total Memory: 3.832GiB
```

1.5 Was ist der Unterschied zwischen einem Container und einem Image?

1.6 Wie und von wo können Images heruntergeladen werden?

Docker Images werden vom so genannten Docker Hub heruntergeladen. Der Docker Hub bündelt alle Images und stellt sie kostenlos den Usern zur Verfügung, die sie herunterladen wollen.

Images können mit einem einfachen `docker pull <IMAGE>` heruntergeladen werden:

```
root@debian10VM:/home/schueler# docker pull debian
Using default tag: latest
latest: Pulling from library/debian
```

1.7 Wie können verfügbare Images angezeigt werden?

Docker Images können mit einem einfachen `docker image ls` aufgelistet werden.

```
root@debian10VM:/home/schueler# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
nginx               latest              ed21b7a8aee9        27 hours
ago         127MB
debian              latest              58075fe9ecce        29 hours
ago         114MB
```

1.8 Was ist eine BusyBox? Probiere die beiden folgenden Varianten zum Start einer Busybox. Worin unterscheiden sich diese?

BusyBox bündelt viele elementare Programme des GNU-Projektes. Sie sind leichtgewichtigere, in der Funktion beschnittene Varianten der GNU-Programme, daher ideal für Embedded-Systeme.

Der erste Befehl sagt Docker, dass er das BusyBox Image in einem Container starten soll und darin, in einer gewöhnlichen Shell, dieses Kommando ausführen soll. Da dies ein Kommand ist, der nur kurz existiert und nur kurz etwas ausgibt und danach wieder beendet wird.

```
root@debian10VM:/home/schueler# sudo docker run busybox echo "Hello
World"
```

```
Hello World
```

```
sudo docker run -it busybox sh
```

Das -i Attribut sagt Docker, dass er den Input offenhalten soll, d.h., dass der Container interaktiv ist.

Das -t Attribut sagt Docker, dass er eine Shell (ein virtuelles TTY) Terminal aufmachen soll.

Das letzte Wort in der Kette ist „`sh`“: Es sagt dem Busybox Image als übergebenes Kommando eine Shell aufzumachen.

```
root@debian10VM:/home/schueler# sudo docker run -it busybox sh
/ #
```

Führt man den Befehl nun aus, befindet man sich in der Shell des BusyBox-Containers.

1.9 Wie können laufende/beendete Container angezeigt werden?

Um alle Container, die laufen oder beendete Container anzuzeigen, kann man das Kommando docker ps -a wobei das Argument -a bedeutet, das alle Container (laufende & gestoppte Container) angezeigt werden.

```
niklas@codeservervm:~$ docker ps -a
CONTAINER ID        IMAGE                  COMMAND
CREATED             STATUS                     PORTS
NAMES
f7ca6188ab13        theiaide/theia:next    "node /home/theia/sr…"   2
weeks ago         Exited (0) 2 weeks ago
vigorous_tesla
```

1.10 Wie können beendete Container „aufgeräumt werden“

Um Container oder Images, die man nicht mehr benötigt, loszuwerden, kann man das Kommando docker image prune bzw. docker container prune ausführen. Dieses Kommando löscht alle Container bzw. Images vom System und gibt somit Speicherplatz frei.

```
niklas@codeservervm:~$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
f7ca6188ab1360c43884c7c46ad7cfcd8229e93044d74815232061f8d867e8c0


Total reclaimed space: 0B
```

1.11 Lade und starte einen nginx Webserver. Der Webserver soll unter http://localhost:5000 erreichbar sein.

Um einen Nginx Server zu starten, der im Detached-Mode läuft, verwendet man folgendes Kommando:

Mit `run` sagt man Docker, dass er von einem bestimmten Image einen Container ausführen soll.

Mit `–name` weißt man einem Container einen Namen zu, in diesem Fall `nginxserver`.

Mit `-d` sagt man Docker, dass er den Container im Detached-Mode starten soll.

Mit `-p` teilt man Docker mit, wohin er den Port 80 (den Standardport von dem Nginx-Container) leiten soll. Dabei wird der Port, der nach außen geht als erstes (5000) und der Port, der auf diesen umgeleitet wird, rechts, als letztes, angegeben.

Als letztes Argument teilt man Docker mit, welches Image, in dem Fall das `Nginx` Image, er benutzen soll.

```
niklas@codeservervm:~$ docker run --name nginxserver -d -p 5000:80
nginx

7ce805c8d549ad3f9064095e6646b22aa5c6a80dfee0b18c56d2a4bd86cb60a7
```

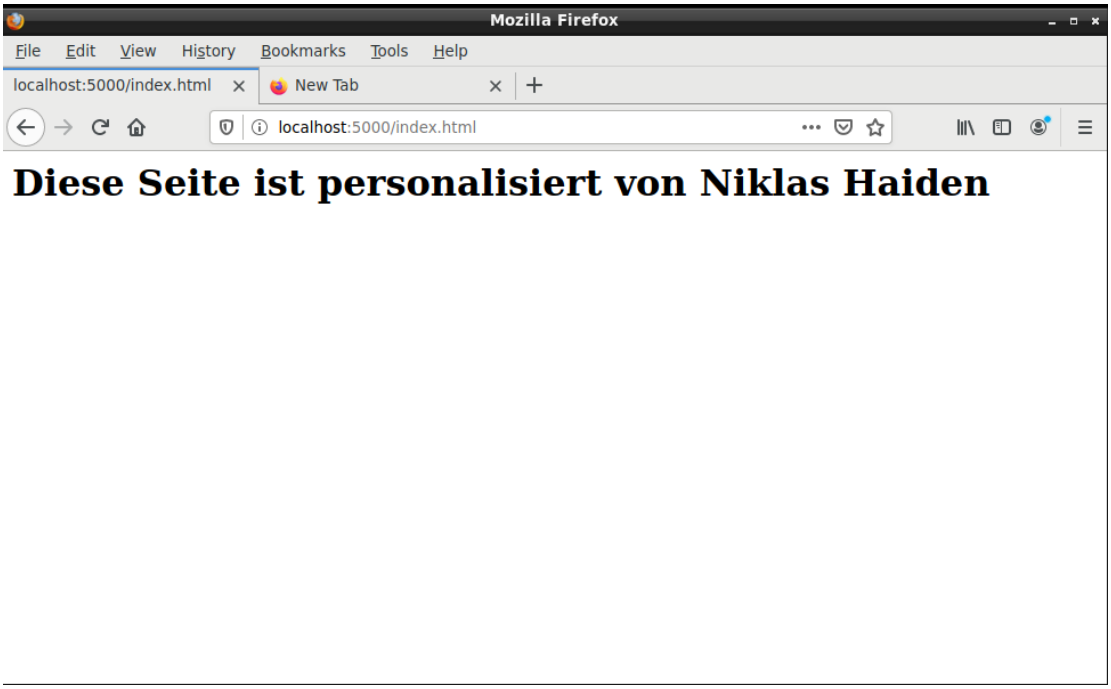
1.12 Was ist der Detached-Mode? Wozu wird/kann er verwendet werden?

Der Detached-Mode bei Docker ist ein Modus, bei dem der Container im Hintergrund auf dem Host-System läuft und keine Display-Outputs anzeigt und keine Eingaben von der Tastatur bzw. Maus erhält. Um einen Detached Container zu starten fügt man an den Run-Command einfach –detach oder -d an. Der Container läuft so nun im Hintergrund, besonders interessant bei Dingen wie einfachen Datenbank-Server-Containern oder Webservern.

1.13 Starte einen neuen Container und zeige eine personalisierte Index Seite an.

Um einen Nginx Container zu starten, der den Ordner mit einer index.html mitnimmt, muss man das Argument -v verwenden. Dieses teilt Docker mit, in von welchem Ordner in welchen Ordner er die Dateien übertragen soll. Dabei werden während der Container startet alle Dateien des Ordner hinüber kopiert und die alten werden dabei ersetzt.

```
niklas@codeservervm:~/html$ sudo docker run --name docker-nginx -p
5000:80 -d -v /home/niklas/html/:/usr/share/nginx/html nginx
```



Labor 11 – Docker

1.14 Installation von Docker-Compose (https://github.com/docker/compose/releases/)

Docker-Compose kann durch einen einfachen Befehl, nämlich `sudo apt install docker-compose` auf einem Ubuntu bzw. Debian-basierten System installiert werden.

```
niklas@codeservervm:~/html$ sudo apt install docker-compose
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
```

1.15 Wozu dient Docker-Compose?

Docker Compose ist ein mächtiges Tool, mit denen man in einer einzelnen Datei Multi-Container Applikationen definieren kann. Definiert werden diese Templates in einer sogenannten YAML-Datei (YAML Ain't Another Markup Language) definiert. Hierbei definiert man die Images, die in den Containern gestartet werden, sondern z.B. auch Docker Volumes, worauf Daten gespeichert werden.