

Windows Docker

Von Sandra Schuart, Sommersemester 2020

Was ist Docker?

Docker ist eine OpenSource Software. Diese stellt Container zur Verfügung, die es ermöglichen, dass Anwendungen einfacher erstellt, bereitgestellt und ausgeführt werden können. Mit Hilfe von Containern, kann der Entwickler Anwendungen mit allen nötigen Bibliotheken und Abhängigkeiten verpacken und diese dann anschließend als ein Paket bereitstellen. Damit hat der Entwickler die Sicherheit, dass die Anwendung auf jedem anderen System ausgeführt werden kann, unabhängig von Betriebssystem und den darauf vorhandenen benutzerdefinierten Einstellungen. Docker ähnelt damit gewissermaßen Virtuellen Maschinen, dennoch gibt es eindeutige Unterschiede zwischen Docker und Virtuellen Maschinen [1]. Der Unterschied zwischen Docker und der Virtuellen Maschine wird im Folgenden erklärt, sowie alles weitere Wichtige.

Docker vs. Virtuelle Maschine

Während Docker Container nutzt, um Anwendungen auf einem Gastrechner auszuführen, nutzt die Virtuelle Maschine die Virtualisierung. Der Unterschied ist, dass mit Hilfe der VM viele Betriebssysteme auf einem einzigen Gastrechner erstellt werden, bei der Nutzung mit Docker werden je nach Bedarf mehrere Container für jeden Anwendungstyp erstellt.

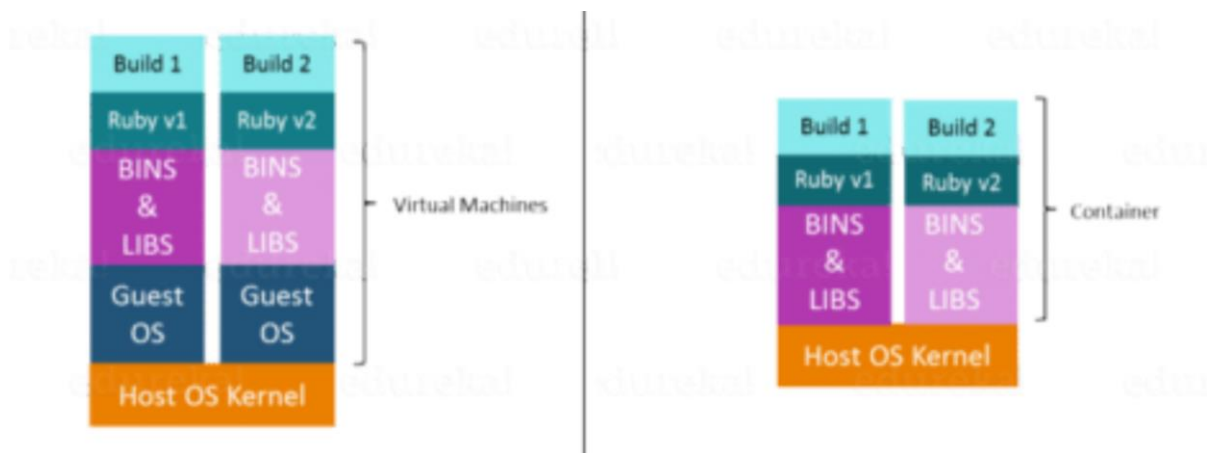


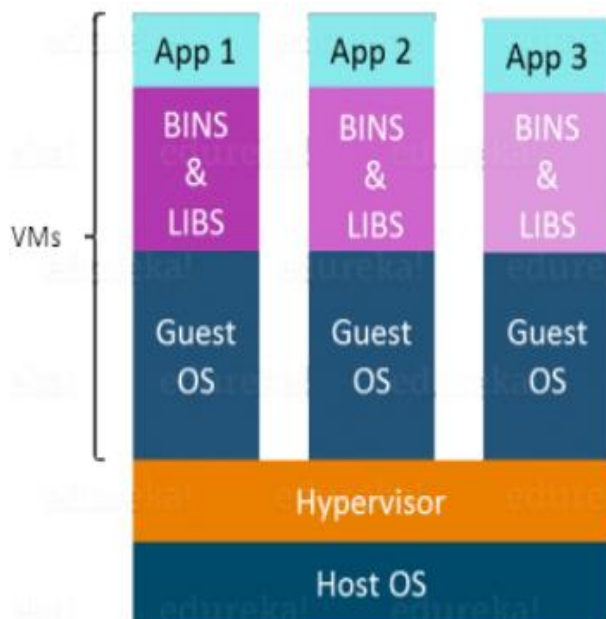
Abbildung 1: Virtuelle Maschine vs. Docker

Wie man in der Abbildung erkennen kann, liegt der Hauptunterschied darin, dass bei der Virtualisierung mehrere Gastbetriebssysteme benötigt werden, was bei der Containerisierung nicht der Fall ist. Dadurch sind Container leichtgewichtiger und schneller in der Benutzung.

Um zu verstehen, wie Docker funktioniert sollte man sich erst den genauen Unterschied zwischen Virtualisierung und Containerisierung vor Augen führen. Im nächsten Abschnitt wird beides ausführlich erklärt.

Virtualisierung

Bei der Virtualisierung können Betriebssysteme als Gastbetriebssysteme auf Rechnern laufen. Diese Technik war eine Revolution für die Entwickler, da man nun in der Lage war verschiedene Betriebssysteme in verschiedenen Virtuellen Maschinen laufen zu lassen, ganz ohne zusätzliche Hardware. Daraus ergeben sich verschiedene Vorteile, wie bereits erwähnt können mehrere Betriebssysteme auf einem Rechner laufen. Bei Ausfällen sind die Wartung und Wiederherstellung einfacher. Aber auch die Gesamtbetriebskosten sind geringer, da der Infrastrukturbedarf durch die Virtualisierung sinkt.



In dieser Abbildung sieht man ein Hostbetriebssystem auf dem drei Gastbetriebssysteme ausgeführt werden, dabei handelt es sich um Virtuelle Maschinen. Doch die Virtualisierung bringt auch Probleme mit sich. Denn das Ausführen von mehreren Virtuellen Maschinen auf einem Hostbetriebssystem führt zu einer schwächeren Leistung. Denn das Gastbetriebssystem hat seinen eigenen Kern und eine Reihe an Bibliotheken und Abhängigkeiten, die auf dem Host-Betriebssystem ausgeführt werden. Damit beansprucht die Virtuelle Maschine einen großen Teil der Systemressourcen wie Festplattenspeicherplatz, Prozessoren aber insbesondere RAM-Leistung.

Abbildung 2: Virtuelle Maschinen

Ein weiterer Nachteil ist, dass das Hochfahren von einer virtuellen Maschine fast eine Minute dauert. Dies kann bei Echtzeitanwendungen ein Problem darstellen. Außerdem ist der Hypervisor nicht so effizient wie das eigentliche Host-Betriebssystem [2].

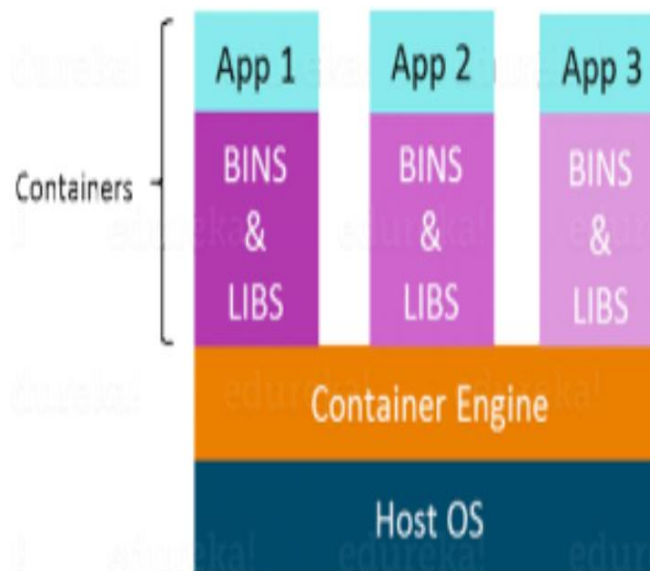
Ein Hypervisor ist eine Software, die zur Virtualisierung von Rechnerressourcen genutzt wird, auch als VMM (Virtual-Machine-Monitor) bekannt. Der VMM hat die Aufgabe den Virtuellen Maschinen Hardware-Ressourcen zuzuweisen, wie Rechenleistung, Arbeitsspeicher, Netzwerkschnittstellen oder auch Festplattenspeicher. Aber auch das Isolieren zwischen den Maschinen ist eine Aufgabe des Hypervisors. Außerdem wird durch den Virtual-Machine-Monitor der parallele Betrieb mehrerer VMs erlaubt [3].

Nichtsdestotrotz führten die Nachteile zu der Entstehung einer neuen Technik, namens Containerisierung. Im nächsten Abschnitt wird diese Technik erläutert.

Containerisierung

Bei der Containerisierung geht es darum die Virtualisierung auf Betriebssystemebene zu bringen. Während bei der Virtualisierung Abstraktion auf die Hardware gebracht wird, wird bei der Containerisierung Abstraktion auf das Betriebssystem gebracht. Aber auch bei Containern handelt es sich um eine Art von Virtualisierung. Jedoch ist die Containerisierung um einiges effizienter als die

Virtualisierung, denn es gibt kein Gastbetriebssystem, welches das Hostbetriebssystem nutzt. Wichtige Bibliotheken und Ressourcen werden erst bei Bedarf geteilt, anders als bei virtuellen Maschinen. Das Hochfahren eines Containers dauert nur den Bruchteil einer Sekunde. Denn Anwendungsspezifische und Binärdateien des Containers werden auf dem Host Kern ausgeführt, dies führt zu einer schnellen Verarbeitung und Ausführung. Außerdem teilen sich alle Container das Hostbetriebssystem. Dies alles führt dazu, dass Container schneller sind als virtuelle Maschinen und weniger Speicherplatz benötigen.



In der Abbildung sieht man deutlich, dass das Hostbetriebssystem gemeinsam von allen Containern genutzt wird. Die Container enthalten nur Anwendungsbezogenen Bibliotheken, die für jeden einzelnen Container separat behandelt werden. Dadurch sind sie schneller und verschwenden keine Ressourcen.

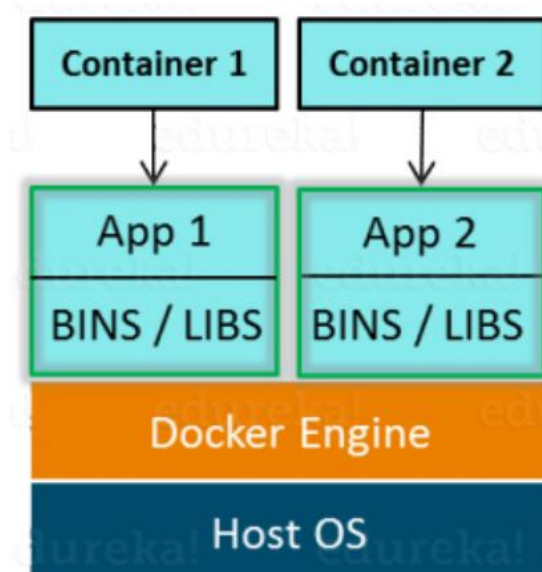
Alle Container werden von der Containerschicht überwacht, doch diese Schicht ist nicht standardmäßig vorhanden, daher wird eine Software benötigt, die das Arbeiten mit Containern ermöglicht.

Abbildung 3: Container

Im Folgenden wird die Software „Docker“ vorgestellt die das Arbeiten mit Containern ermöglicht.

Einführung in die Software Docker

Docker ist eine Software die Container nutzt, um Anwendungen gemeinsam mit allen ihren Abhängigkeiten zu verpacken, um sicher zu stellen, dass die Anwendung in jeder Umgebung Reibungslos funktioniert.



Wie man in der Grafik sehen kann, wird jede Anwendung in einem separaten Container ausgeführt. Darüber hinaus verfügt jeder Container über eigene Bibliotheken und anwendungsbezogene Binärdaten. Damit wird auch sichergestellt, dass jede Anwendung unabhängig von anderen Anwendungen ist. Dadurch können sich verschiedene Anwendungen nicht gegenseitig stören.

Somit hat man als Entwickler die Möglichkeit einen Container zu erstellen auf den verschiedene Anwendungen konfiguriert werden. Dieser Container kann dann von anderen Entwickler ausgeführt werden, um die Anwendungen zu replizieren.

Abbildung 4: Docker

Hierdurch bringt Docker viele Vorteile mit sich, denn andere Entwickler müssen sich nun nicht mehr die benötigten Anwendungsprogramme und dazugehörige Software installieren, um eine Anwendung testen zu können. Dadurch können viel Zeit und Energie gespart werden. Unter anderem wird sichergestellt, dass die Arbeitsumgebung während allen Schritten für alle Beteiligten beständig bleibt. Aber auch die Anzahl der integrierten Systeme kann ohne weiteres erhöht werden [2].

Dockerfile, Image & Container

Sowohl Dockerfile, Image und auch Container sind wichtige Begriffe im Zusammenhang mit Docker.



Abbildung 5: Container Entstehungsprozess

Sobald das Dockerfile erstellt wird, wird dieses zum Docker Image. Wird dieses Image ausgeführt wird das Image zum Container.

Dockerfile

Bei einem Dockerfile handelt es sich um ein Textdokument, dieses Dokument enthält alle Befehle, die aufgerufen werden können, um ein Image zu erstellen. Dadurch ist Docker in der Lage automatisch aus dem Dockerfile ein Image zu erstellen. Mittels `docker build` können Befehle nacheinander ausgeführt werden.

Image

Einfach gesagt kann das Docker Image mit einer Vorlage verglichen werden, die genutzt werden kann um Container zu erstellen. Mittels `docker run` wird das Image ausgeführt und zu einem Container. Die Docker Images werden schreibgeschützt im Docker Registry gespeichert. Dies kann öffentlich oder lokal geschehen.

Container

Hierbei handelt es sich um eine ausführbare Instanz eines Images, diese Instanz enthält alle Pakete die benötigt werden, um ein Programm auszuführen. Also handelt es sich hier im Grunde genommen um das fertige Produkt von Docker [4].

Installation von Docker unter Windows 10

Get Docker Desktop for Windows

Docker Desktop for Windows is available for free.

Requires Microsoft Windows 10 Professional or Enterprise 64-bit. For previous versions get [Docker Toolbox](#).

By downloading this, you agree to the terms of the [Docker Software End User License Agreement](#) and the [Docker Data Processing Agreement \(DPA\)](#).



Um mit Docker auf Windows zu arbeiten, muss die freiverfügbare Software auf der offiziellen Docker Homepage heruntergeladen und installiert werden [5].

Abbildung 6: Docker Software [5]

Um zu überprüfen ob die Docker Software erfolgreich installiert worden ist kann man den Befehl:

docker run hello-world

in die PowerShell eintippen.

```
PS C:\Users\sandra> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Abbildung 7: Zeigt, dass die Installation erfolgreich ausgeführt worden ist. Das hello-world Image wurde von Docker gepullt und kann als Container ausgeführt werden.

Mit dem Befehl:

docker image -ls

kann man sich alle Images anzeigen lassen die erstellt worden sind, um zu überprüfen ob ein Image auch tatsächlich erstellt worden ist [6].

Erste Schritte mit Docker Tutorial

Nun wird das nächste Image gepullt, um einen Container auszuführen.

```
docker run -d -p 80:80 docker/getting-started
```

Erklärung:

- `-d` -> führt den Container im Hintergrund als Dämon-Prozess aus
- `-p 80:80` -> ordnet den Host Port 80 dem Container Port 80 zu
- `docker/getting-started` -> das Image [7]

```
PS C:\Users\sandra> docker run -dp 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
aad63a933944: Pull complete
b14da7a62044: Pull complete
343784d40d66: Pull complete
6f617e610986: Pull complete
Digest: sha256:d2c4fb0641519ea208f20ab03dc40ec2a5a53fdfbcccc90bef14f870158ed577
Status: Downloaded newer image for docker/getting-started:latest
1b829fae889d22a82e59d55703ec3b281a4be5f6453cfdfb3e25f67099dfbd6a
```

Abbildung 8: PowerShell die zeigt, dass das Image erfolgreich gepullt worden ist

In der folgenden Grafik dem Docker Dashbord, kann man alle Container sehen die momentan auf dem Computer ausgeführt werden könnten. Dadurch erhält man Schnellzugriff auf die Containerprotokolle. Man kann eine Shell mit einbinden oder die vorhandenen Container verwalten. Bei den Containernamen handelt es sich um zufällig erstellte Namen.

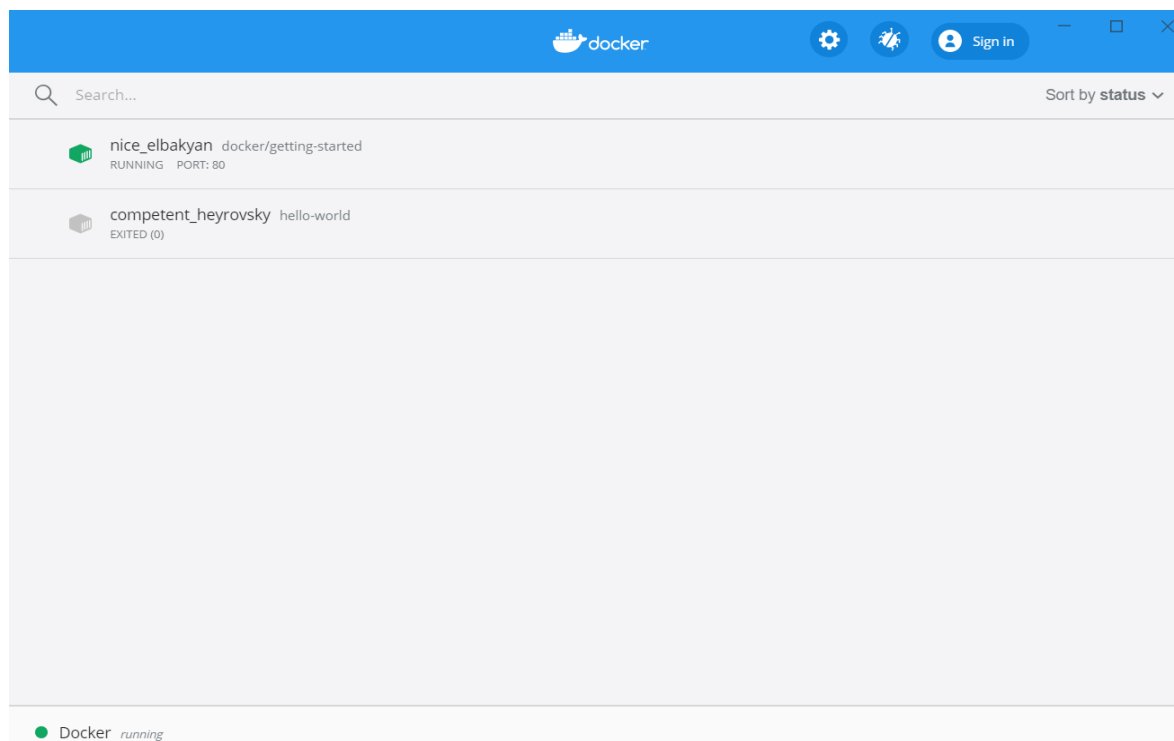
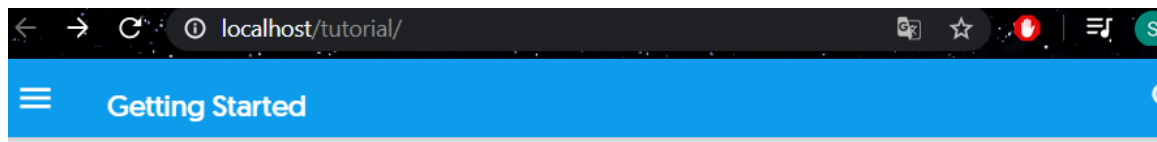


Abbildung 9: Docker Dashbord

Wenn man nun den Dockercontainer mittels eines Webbrowser auf localhost:80 aufruft sieht man, dass man Zugriff auf den Default Webserver von Docker hat.



Getting Started

The command you just ran

Congratulations! You have started the container for this tutorial! Let's first explain the command that you just ran. In case you forgot, here's the command:

Abbildung 10: Container erfolgreich gestartet [7].

Erstellen eines Apache - HTTP Servers

Umgangssprachlich Apache genannt, spielt dieser Webserver eine Schlüsselrolle bei der Erstellung des World Wide Webs. Heute ist er einer der beliebtesten HTTP-Server. Vor 1995 basierte er auf dem NCSA-HTTPd-Server.

Im Folgenden wird ein Apache – HTTP Server mit einem einfachen HTML-Code unter der Benutzung von Docker ausgeführt.

Zunächst erstellt man einen neuen Ordner in welchem sich anschließend der Quellcode für den Webserver befinden soll. Danach wechselt man in den erstellten Ordner, um Zugriff auf diesen Quellcode zu haben.

```
PS C:\Users\sandra> cd C:/Users/sandra/desktop
PS C:\Users\sandra\Desktop> mkdir webserver

Verzeichnis: C:\Users\sandra\Desktop

Mode                LastWriteTime         Length Name
----                -
d-----          15.06.2020    08:53             webserver

PS C:\Users\sandra\Desktop> cd C:/Users/sandra/desktop/webserver
PS C:\Users\sandra\Desktop\webserver>
```

Abbildung 11: Die ersten Schritte zum erstellen eines Webservers

```
PS C:\Users\sandra\Desktop\webserver> mkdir public-html

Verzeichnis: C:\Users\sandra\Desktop\webserver

Mode                LastWriteTime         Length Name
----                -
d-----          15.06.2020    09:02             public-html
```

Nun wird ein Unterordner mit dem Namen public-html erstellt, in welchem sich anschließend der HTML-Code namens index.html befinden wird. Anschließend wechselt man in den neu erstellten Ordner.

Abbildung 12: public-html Ordner wird erstellt

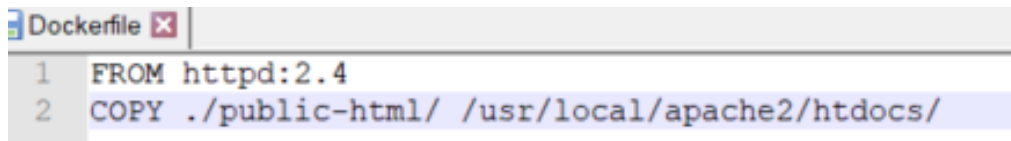
Um mit Notepad++ den HTML-Quellcode zu erstellen gibt man folgenden Befehl in die Powershell ein:

start Notepad++

```
1 <html>
2   <head>
3     <HTTP Apache Webserver>
4     <title> Dies ist ein Webserver </title>
5   </head>
6   <body>
7     Hallo Welt!
8   </body>
9 </html>
```

Abbildung 13: Einfacher HTML-Quellcode dessen Inhalt später beim Aufrufen des Webserver angezeigt wird.

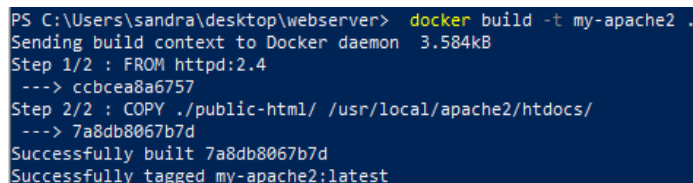
Als nächstes wechselt man wieder in den Ordner webserver, in diesem wird später das Dockerfile erstellt.



```
1 FROM httpd:2.4
2 COPY ../public-html/ /usr/local/apache2/htdocs/
```

Abbildung 14: Dockerfile

Wie bereits erwähnt handelt es sich bei einem Dockerfile um ein Dokument in welchem man definieren kann welche Befehle bei der Erstellung eines Images ausgeführt werden sollen. In Zeile eins wird die neuste Version des Apache HTTP Servers geladen. In der zweiten Zeile wird der Quellcode aus dem public-html Ordner in den Apache htdocs Ordner geladen, um diesen dort auszuführen. Bei der Namensgebung des Dockerfiles ist wichtig zu beachten, dass der Dateiname „Dockerfile“ groß geschrieben wird, da Docker diese ansonsten nicht findet. Nun kann man das Image erstellt werden.

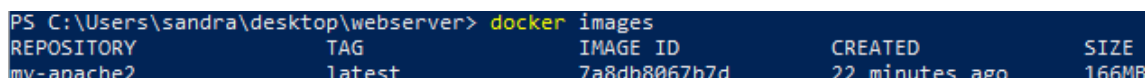


```
PS C:\Users\sandra\desktop\webserver> docker build -t my-apache2 .
Sending build context to Docker daemon 3.584kB
Step 1/2 : FROM httpd:2.4
--> ccbcea8a6757
Step 2/2 : COPY ../public-html/ /usr/local/apache2/htdocs/
--> 7a8db8067b7d
Successfully built 7a8db8067b7d
Successfully tagged my-apache2:latest
```

Abbildung 15: Erstellen eines Docker-Images

Durch den Befehl: „*docker build -t my-apache.*“ wird ein Image erstellt namens my-apache. Durch den punkt wird der nächste Ordner angesprochen in welchem sich der index-html Code befindet. Parameter -t referenziert dabei auf den Quellcode.

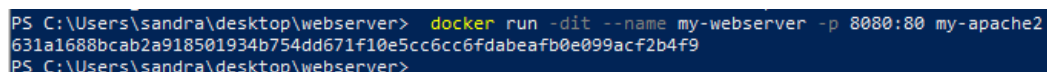
Um sicherzustellen, dass das Image wirklich erstellt worden ist benötigt man den Befehl:



```
PS C:\Users\sandra\desktop\webserver> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
my-apache2          latest          7a8db8067b7d   22 minutes ago 166MB
```

Abbildung 16: Zeigt, dass das Image erfolgreich erstellt worden ist.

Im Folgenden wird der Container nun mit dem run Befehl gestartet.



```
PS C:\Users\sandra\desktop\webserver> docker run -dit --name my-webserver -p 8080:80 my-apache2
631a1688bcb2a918501934b754dd671f10e5cc6cc6fdabeafb0e099acf2b4f9
PS C:\Users\sandra\desktop\webserver>
```


Abbildung 17: Befehl der den Container startet

Der Container läuft nun mit dem Namen my-webserver auf dem Hostport 8080 und wird dabei dem Containerport 80 zugeordnet.

Wenn man nun in den Webbrowser <http://localhost:8080> eingibt sieht man der Webserver wurde erfolgreich eingerichtet.



Abbildung 18: Erfolgreich aufgesetzter Webserver auf der Basis eines einfachen HTML-Dokuments

Quellen

[1] What is Docker?, Red Hat, Inc 2019

URL: <https://opensource.com/resources/what-docker>

Letzte Einsicht: 24.05.2020

[2] Docker Tutorial – Introduction To Docker & Containerization, Vineet Chaturvedi, 22.05.2019

URL: <https://www.edureka.co/blog/docker-tutorial>

Letzte Einsicht: 24.05.2020

[3] Was ist ein Hypervisor?, Stefan Luber/ Dr. Jürgen Ehneß, 13.08.2019

URL: <https://www.storage-insider.de/was-ist-ein-hypervisor-a-842084/>

Letzte Einsicht: 25.05.2020

[4] Docker Explained – An Introductory Guide To Docker, Sahiti Kappagantula, 27.11.2019

URL: <https://www.edureka.co/blog/docker-explained/>

Letzte Einsicht: 26.05.2019

[5] Software Docker Windows

URL: <https://hub.docker.com/editions/community/docker-ce-desktop-windows>

Letzte Einsicht: 10.06.2020

[6] Get started with Docker for Windows

URL: <https://docs.docker.com/docker-for-windows/>

Letzte Einsicht: 10.06.2020

[7] Getting started

URL: <http://localhost/tutorial/>

Letzte Einsicht: 10.06.2020

[8] httpd (Docker Official Image)

URL: https://hub.docker.com/_/httpd?tab=description

Letzte Einsicht: 15.06.2020