Kanton St.Gallen
Gewerbliches Berufs- und Weiterbildungszentrum St.Gallen

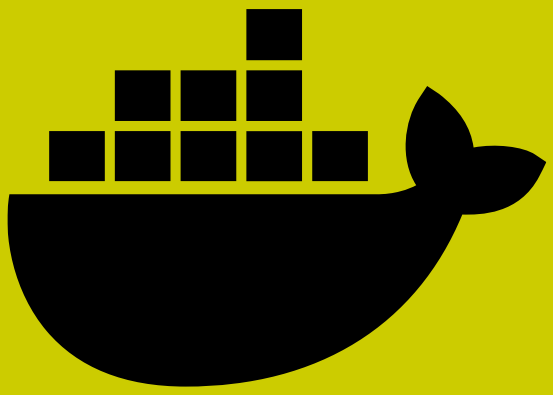
Grundbildung

gbssg.ch

Informatik – Modul 169.

Services mit Containern bereitstellen

Docker



Inhaltsverzeichnis.

1 Docker

Um mit Docker vertraut zu werden, werden im folgenden Kapitel einige Grundkonzepte erläutert, mit welchen Sie im folgenden experimentieren können. Die folgenden Abschnitte geben Ihnen Anregungen dazu. Wichtig ist, dass das Ziel der Beispiele nicht der Start eines echten Entwicklungssystems oder gar der produktive Einsatz von Docker ist. An dieser Stelle geht es vielmehr darum, die Konzepte von Docker anhand von Beispielen kennen zu lernen – Learning by Doing also.

Versuchen Sie parallel dazu die wichtigsten Dinge für sich zu dokumentieren.

1.1 Container ausführen

Um einen Container in Docker auszuführen, können Sie in der Kommandozeile den Befehl `docker run` verwenden. Falls der gewünschte Container lokal noch nicht verfügbar ist, wird er zuerst von Docker Hub heruntergeladen und danach gestartet. Der zweite Start wird danach blitzschnell ausgeführt. Mit `docker ps -a` kann der Status eingesehen werden und mit `docker image rm` kann der Container lokal wieder gelöscht werden.

Praxisaufgabe: Hello-World Container

Führen Sie den Hello-World Container aus, lassen Sie sich Statusinformationen anzeigen und löschen Sie ihn am Ende wieder.

1.2 Container interaktiv verwenden

Der Hello-World Container ist insofern speziell, da sich der Container direkt wieder schliesst. Viele andere Container können interaktiv genutzt werden. Um dies zu testen eignen sich die gängigen Base-Images von Linux-Distributionen, z.B. von Alpine, CentOS, Debian oder Ubuntu.

Base-Images dienen als Startpunkt für eigene Entwicklungen und können ergänzt werden. Wird jeweils keine Version angegeben, dann verwendet Docker die Version mit dem Attribut *latest*. In der Praxis empfiehlt sich die Version jeweils anzugeben, da ansonsten allenfalls ungewünschte Effekte entstehen.

Mit den Optionen `-i` und `-t` können Sie Container interaktiv verwenden und Docker stellt einen Pseudo-Terminaleditor über die Standardeingabe zur Verfügung.

Praxisaufgabe: Container interaktiv verwenden

Verwenden Sie ein gängiges Linux interaktiv und versuchen Sie dabei die folgenden Fragen zu beantworten:

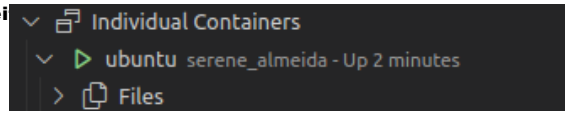
1. Sie setzt sich der Image-Namen zusammen? imageding:version https://hub.docker.com/search?q=
2. Wie können Sie das lokale Image wieder auffrischen?
3. Warum funktionieren die Kommandos `ip addr` oder `ping` nicht? net tool nicht installiert
4. Welchen Namen hat Ihr Container und wie können Sie diesen erneut ausführen?
5. Wie können Sie eigene Container-Namen vergeben? docker rename alt neu

docker start container_id

```
heike@ubuntu:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
677076032cca: Pull complete
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
heike@ubuntu:~$
```



```
heike@ubuntu:~$ docker run -ti ubuntu:22.04
root@367a49f55a18:/#
```



1.3 Portweiterleitungen

Der Container *getting-started* ist im Vergleich mit den bisherigen Beispielen spannend, da er eine Integration in das lokale Netzwerk bringt. Der Container läuft im Hintergrund und über den Browser kann der einfache Webserver über den Port 80 aufgerufen werden.

Aufgabe: Portweiterleitungen

1. Worin könnte eine Problematik bei der Integration in ein lokales Netzwerk auftreten? Denken Sie dabei an den Fall, bei Sie bereits einen lokalen Webserver laufen haben.
2. Notieren Sie sich den Befehl zur Portweiterleitung.

docker run -d -p
8888:80 docker/
getting-started

Ports können schon belegt sein

Praxisaufgabe: Container *getting-started*

1. Starten Sie den Container *getting-started* unter dem Port 8080 und rufen Sie die entsprechende Webseite auf.
2. Lesen Sie das *getting-started* durch.
3. Beenden Sie am Ende den Docker-Container wieder und löschen Sie ihn.

docker stop
pedantic_mahavira

1.4 Datenspeicherung in Volumes

Dieser Abschnitt beschäftigt sich mit der Frage, wie und wo Container Daten speichern. Um dies zu erforschen eignet sich ein Datenbankserver, bspw. MariaDB (https://hub.docker.com/_/mariadb). MariaDB speichert grundsätzlich die Daten im Verzeichnis /var/lib/mysql. Im Dockerfile von MariaDB, also in der Beschreibung des Images, ist /var/lib/mysql aber als Volume gekennzeichnet. Das bedeutet, dass dieses Verzeichnis ausserhalb des Containers angelegt wird. Es wird aber immer noch von Docker verwaltet und somit hat man keinen direkten Zugriff darauf. Diese Volumes ermöglichen es aber, dass andere Container darauf zugreifen können.

Praxisaufgabe: MariaDB

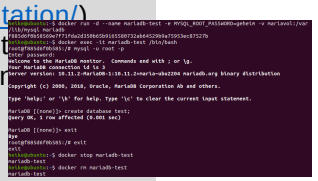
1. Starten Sie den Container *mariadb*. Informieren Sie sich, wie Sie den Container mit Benutzer und Passwort starten. Überlegen Sie sich auch, ob dies so für die Praxis tauglich ist. **docker run -d –name herbert -e MYSQL_ROOT_PASSWORD=geheim mariadb**
2. Analysieren Sie das Volume mit dem `inspect` Befehl.

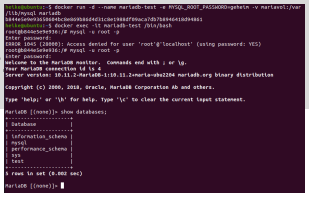
1.4.1 Volumes mit Namen

Standardmässig kümmert sich Docker um die Volumes, allerdings haben die automatisch erzeugten Volumes einen gravierenden Nachteil. Sie bekommen eine zufällige, 64-stellige hexadezimale Zahl. Die Option `-v <vname>:<containerdir>` ordnet einem Verzeichnis des Containers einen Volumen-Namen zu.

Praxisaufgabe: MariaDB Update

1. Starten Sie den Container *mariadb* und geben Sie einen eigenen Volumen-Namen an. Erstellen Sie in der Datenbank Daten (https://mariadb.org/documentation/)
2. Stoppen und löschen Sie den Container wieder, laden ihn erneut herunt
   starten ihn wieder mit dem Volumen-Namen (Simulation Update Versio
   MariaDB).
3. Prüfen Sie Ihre Daten.
4. Löschen Sie am Ende Ihre Volumes wieder



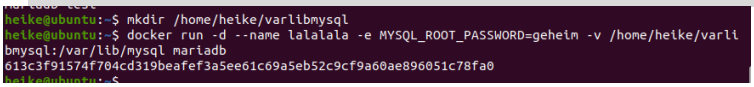


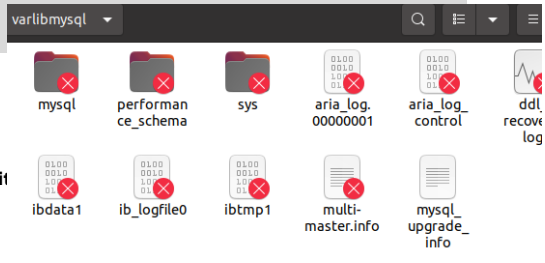
1.4.2 Volumes in eigenen Verzeichnissen

Docker ermöglicht es auch, Volumes in einem eigenen Verzeichnis im lokalen Dateisystem zu speichern. Dies hat den Vorteil, dass losgelöst von Docker darauf zugegriffen werden kann.

Praxisaufgabe: MariaDB eigenes Verzeichnis

1. Erstellen Sie einen Ordner, bspw. `mkdir /home/<username>/varlibmysql`.
2. Starten Sie den Container *mariadb* und geben das lokale Volume an mit `-v /home/<username>/varlibmysql/:/var/lib/mysql`
3. Versuchen Sie von Ihrem lokalen Dateisystem darauf zuzugreifen.
4. Recherchieren Sie, ob Sie auch ein Volume in Windows erstellen können, d.h. unter C:\Users\name\dir…
   Hinweis: In Dockerforen gibt es eine Menge Threads von Benutzern, die Probleme mit Volume-Verzeichnissen in Windows haben. Verzichten Sie am besten darauf.
5. Löschen Sie am Ende Ihre Volumes wieder.





1.5 Kommunikation zwischen Container

Für viele Docker-Anwendungen gilt: Ein Container kommt selten alleine. Sobald mehrere Container gleichzeitig ausgeführt werden, müssen diese untereinander kommunizieren können. Am einfachsten gelingt dies über ein internes Netzwerk, das Sie mit `docker network` einrichten können.

Praxisaufgabe: Wordpress in Betrieb nehmen

1. Erstellen Sie ein eigenes Netzwerk.
2. Nehmen Sie MariaDB, phpMyAdmin und Wordpress als eigene Container in Betrieb und verbinden Sie diese Container über ihr eigenes Netzwerk.
3. Starten Sie über einen Browser die Basiskonfiguration von Wordpress.

1.6 Docker administrieren

Aufgabe: Docker administrieren

Informieren Sie sich über die folgenden Themen und notieren Sie sich die entsprechenden Befehle dazu:

- Platzbedarf von Images und Containern ermitteln
- Container und Images löschen
- Volumes verwalten
- Gesamtüberblick erhalten
- Ungenutzten Speicher freigeben

docker rm [containername]

docker image rm [imagename]

docker volume inspect my-vol

docker Images
docker container ls
docker volume ls

docker Image prune
docker container prune
docker volume prune
docker network prune

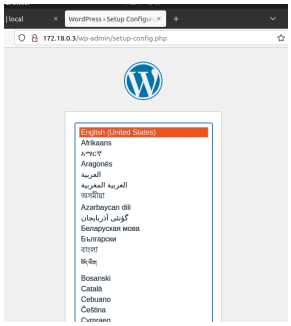
```
heike@ubuntu:~$ docker network create -d bridge test-netzwerk
6042578fbfdb13cedf4d485836afa067e7b2528be8d212afcb69f3f85059642c
```

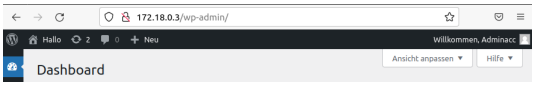
docker run -d --name Maria -e MYSQL_ROOT_PASSWORD=geheim mariadb

docker run --name phpmyadm -d --link maria:db -p 8081:80 phpmyadmin

docker run -d --name word -d -p 8088:80 wordpress

```
$ docker network connect test-netzwerk maria
$ docker network connect test-netzwerk php
$ docker network connect test-netzwerk word
$
```





Source: Ihttps://migueldoctor.medium.com/run-mysql-phpmyadmin-locally-in-3-steps-using-docker-74eb735fa1fc#:~:text=You%20only%20need%20to%20open,tutorial%20the%20password%20is%20mypass123).

LUX WEG:

docker network create tibornetwork
docker run -d --name tiborwordpress --network tibornetwork -h tiborwordpress -v tiborhtmlvolume:/var/www/html/wp-content -p 8081:80 -e WORDPRESS_DB_HOST=tibormariadb -e WORDPRESS_DB_USER=tibooor -e WORDPRESS_DB_NAME=tiborwordpress -e WORDPRESS_DB_PASSWORD=gibbiX12345 wordpress
docker run -d --name tiborphpmyadmin --network tibornetwork -p 8080:80 -e PMA_HOST=tibormariadb phpmyadmin/phpmyadmin
docker run -d --name tibormariadb --network tibornetwork -e MYSQL_RANDOM_ROOT_PASSWORD=1 -e MYSQL_DATABASE=tiborwordpress -e MYSQL_USER=tibooor -e MYSQL_PASSWORD=gibbiX12345 -v tiborvolume:/var/lib/mysql mariadb

2 Eigene Docker-Images (Dockerfiles).

Die Stärke von Docker liegt im Erstellen von eigenen Containern, die genau den gewünschten Anforderungen entsprechen. Natürlich könnte jedes Mal, wenn ein Container eingerichtet wird, die erforderlichen Zusatzarbeiten erledigt werden. Effizienter ist es aber, in solchen Fällen ein eigenes Image zu erstellen. Dazu können die Änderungen in eine Datei namens Dockerfile eingetragen und mittels `docker build` ein neues lokales Image erzeugt werden.

Eigene erstellte Images können auch mit anderen Benutzern geteilt werden, indem diese in den Docker Hub hochgeladen werden.

Freiwilliger Leseauftrag: Dockerfile – Best Practice

Unter https://docs.docker.com/develop/develop-images/dockerfile_best-practices/ finden Sie hilfreiche Informationen im Umgang mit Dockerfiles.

2.1 Dockerfile Syntax

Dockerfiles sind Textdateien, die in einer spezifischen Syntax und mit Schlüsselwörtern aufgebaut sind. Die wichtigsten Schlüsselwörter sind:

Schlüsselwort	Bedeutung
ADD	Kopiert Dateien in das Dateisystem des Images.
CMD	Führt das angegebene Kommando beim Container-Start aus.
COPY	Kopiert Dateien aus dem Projektverzeichnis in das Image.
ENTRYPOINT	Führt das angegebene Kommando immer beim Container-Start aus.
ENV	Setzt eine Umgebungsvariable
EXPOSE	Gibt die aktiven Ports des Containers an.
FROM	Gibt das Base-Image an.
LABEL	Legt eine Zeichenkette fest.
RUN	Führt das angegebene Kommando aus.
USER	Gibt den Account für RUN, CMD und ENTRYPOINT an.
VOLUME	Gibt Volume-Verzeichnisse an.
WORKDIR	Legt das Arbeitsverzeichnis für RUN, CMD, COPY etc. fest.

Aufgabe: Einstieg Dockerfile

Schauen Sie sich die Einführung unter https://www.youtube.com/watch?v=WmcdMiyqfZs an und machen Sie sich dazu Notizen.

2.2 Einführungsbeispiel

Ein minimales `Dockerfile`, das das Ubuntu-Base-Image um das Paket des Editors `joe` erweitert, sieht folgendermassen aus:

```
# Datei Dockerfile
FROM ubuntu:20.04
LABEL maintainer "name@somehost.com"
RUN   apt-get update && \
      apt-get install -y joe && \
      apt-get clean && \
      rm -rf /var/lib/apt/lists/*
CMD ["/bin/bash"]
```

Aufgabe: Einführungsbeispiel Dockerfile

1. Versuchen Sie die einzelnen Kommandos beim Einführungsbeispiel nachzuvollziehen und zu beschriften
2. Recherchieren Sie den Unterschied zwischen ADD und COPY.
3. Recherchieren Sie den Unterschied zwischen CMD und ENTRYPOINT (Hinweis: Bei vielen Base-Images ist der ENTRYPOINT nicht definiert)
4. Warum werden bei RUN die Befehle mit einem `&&` verknüpft und nicht einzeln abgesetzt?

Praxisaufgabe: Einführungsbeispiel Image erzeugen

1. Erstellen Sie das Dockerfile vom Einführungsbeispiel.
2. Erzeugen Sie das Image. Sie können den Accountnamen noch weglassen oder sich bei Docker Hub (https://hub.docker.com/) registrieren und diesen Usernamen verwenden.
3. Starten Sie danach das eigene erstellte Image.
4. Wenn alles geklappt hat, räumen Sie danach wieder auf.

2.3 Mini-Projekt

Praxisaufgabe: Mini-Projekt

Erstellen Sie ein eigenes Webserver-Image (entweder mit Apache oder NGINX) und lassen Sie darin eine einfache Webseite laufen (Applikationsentwickler können die Webseite aus Modul 293 verwenden).
Ziel ist es, dass Ihre Webseite über den Browser unter Port 8080 aufgerufen werden kann.
Starten Sie den Container am Ende so, dass sich die Dateien zur Webseite (HTML, CSS, etc.) und die Logdateien in lokalen Verzeichnissen befinden.

Dokumentieren Sie Ihr Mini-Projekt auf Github/Gitlab sinnvoll (inkl. Dockerfile) und geben Sie die URL dazu der Lehrperson ab.

3 Docker compose

Mit Docker compose können Sie ganze Container-Umgebungen verwalten.

Aufgabe: Einstieg Docker compose

Zum Einstieg in Docker compose schauen Sie sich den folgenden Video an und versuchen Sie für sich eine Zusammenfassung zu erstellen.
https://www.youtube.com/watch?v=HG6yIjZapSA

Aufgabe: YAML

1. Die Datei `docker-compose.yml` ist mit YAML aufgebaut. Vertiefen Sie sich in die Struktur von YAML und machen Sie sich hierzu Notizen.

2. Wo sehen Sie Vor- und Nachteile von YAML gegenüber JSON oder anderen Beschreibungssprachen?

Praxisaufgabe: Wordpress in Betrieb nehmen

1. Erstellen Sie mittels Docker compose verschiedene Container für Wordpress (siehe auch Praxisaufgabe: Wordpress in Betrieb nehmen)
2. Ziel ist es, dass Sie mittels `docker compose up` die Wordpress-Infrastruktur in Containern starten können.
3. Beschreiben Sie die einzelnen Zeilen aus Ihrer `docker-compose.yml` Datei.

Aufgabe: Geheimnisse mit Docker

Bisher wurden Passwörter als Umgebungsvariablen an die Container übergeben. Besser ist es, Passwörtger, SSH-Schlüssel, SSL-Zertifikate und vergleichbare Dateien als sogenannte Geheimnisse zwischen Docker-Containern zu übertragen.

Informieren Sie sich unter https://secrethub.io/docs/guides/docker/ zum Einsatz von Secrets und testen Sie diese aus.

4 Literaturverzeichnis.

Bernd Öggl, M. K. (2021). *Docker: Das Praxisbuch für Entwicler und DevOps-Teams.* Bonn: Rheinwerk Computing.

Docker Dokumentation. (kein Datum). Von Docker Dokumentation: https://docs.docker.com/ abgerufen