TMF – Technologie- und Methodenplattform
für die vernetzte medizinische Forschung e.V.

Dockerbank

Container-basiertes Deployment von biomedizinischen IT-Lösungen

Praktische Übung 2: Erstellung eigener Container, Orchestrierung von Containern

Benjamin Baum[1], Sebastian Stäubert[2]

[1] Institut für Medizinische Informatik, Universitätsmedizin Göttingen

[2] Institut für Medizinische Informatik, Statistik und Epidemiologie (IMISE), Universität Leipzig

Docker TMF VM

Docker TMF

- Voraussetzungen:
- Oracle VirtualBox
- Docker TMF VM
- PgAdmin3 (Optional) https://www.pgadmin.org/
- Putty (Optional) http://www.putty.org/
- Webbrowser

TMF Docker VM

- Login: **docker/root**
- Password: **docker**
- Root werden: **sudo -s**
- SSH (putty): **localhost:5022**

- Vorbereitete Übungen unter: **/docker/excercise (Achtung, TYPO!)**
- Lösungen unter: **/docker/solution**
- Cheatsheet **/docker/dockercheatsheet.txt**

Praktische Übung 2

Erstellung eigener Container

Dockerfile

Dockerfile

- **FROM** <baseimage> – Auswahl eines Basisimages
- **MAINTAINER** – Information über Betreiber
- **RUN** <command> – Ausführen eines Befehls
- **ENV** – Environment Variablen
- **COPY** – Kopiert Dateien vom Host
- **VOLUME** – Erstellt einen Einhängepunkt
- **EXPOSE** – Öffnet Ports
- **CMD** – Was beim Starten ausgeführt werden soll

- Achtung: Läuft kein aktiver Prozess, beendet sich ein Container!

Apache Webserver

1. Apache Container

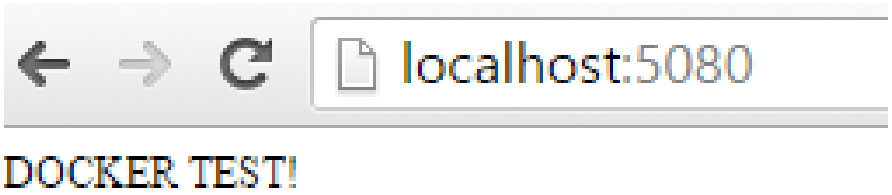
1. Navigieren Sie in den Ordner /docker/excercise/apache
2. Editieren Sie das Dockerfile:
    1. Ausgehend von ubuntu:16.04 (FROM)
    2. Update der Installierten Pakete (RUN)
       (apt-get update && apt-get -yy upgrade)
    3. Installieren von Apache (RUN)
       apt-get install -yy apache2
    4. index.html erstellen und nach /var/www/html/ im Container kopieren (COPY)
    5. Port 80 öffnen (EXPOSE)
    6. Apache2 starten und Logs ausgeben (CMD)
       (service apache2 start && tail -f /var/log/apache2/access.log)

1. Apache Container

3. Image kompilieren
   1. `docker build -t tutorial_apache .`
4. Container ausführen
   1. `docker run -it -p 80:80 --name tutorial_apache tutorial_apache`
   2. http://localhost:5080 im Webbrowser aufrufen
5. Lokale index.html einbinden, anstatt COPY
   1. `docker rm -f tutorial_apache`
   2. `docker run -it -v html/index.html:/var/www/html -p 80:80 --name tutorial_apache tutorial_apache`
   3. localhost:5080 im Webbrowser aufrufen
   4. Lokale index.html ändern, Seite refreshen

1. Apache Container Solution

- cd /docker/solution/apache
- `docker build -t tmf_apache .`
-
```
docker run
-it
-v /docker/solution/apache/html/:/var/www/html
-p 80:80
--name tmf_apache
tmf_apache
```



2. PostgreSQL-9.5 Datenbank

2. PostgreSQL Container

1. Navigieren Sie in den Ordner /docker/excercise/postgres
2. Editieren Sie das Dockerfile:
    1. Ausgehend von Ubuntu:16.04 (FROM)
    2. Update der installierten Pakete (RUN)
       (apt-get update && apt-get -yy upgrade)
    3. Installieren von PostgreSQL-9.5 (RUN)
       apt-get install -yy postgresql-9.5
    4. Datenbank-Dateien nach Außen öffnen (VOLUME)
        1. /var/lib/postgresql/
    5. Remote Verbindungen erlauben (RUN)
        1. echo "host all all 0.0.0.0/0 md5" >>
           /etc/postgresql/9.5/main/pg_hba.conf
        2. echo "listen_addresses='*'" >>
           /etc/postgresql/9.5/main/postgresql.conf

2. PostgreSQL Container

6. Port 5432 öffnen (EXPOSE)
7. PostgreSQL starten und Logs ausgeben (CMD)
   (service postgresql start && tail -f /var/log/postgresql/postgresql-9.5-main.log)

3. Image kompilieren
   1. `docker build -t tutorial_apache .`
4. Container ausführen
   1. `docker run -it -p 80:80 --name tutorial_postgres tutorial_postgres`

2. PostgreSQL Container

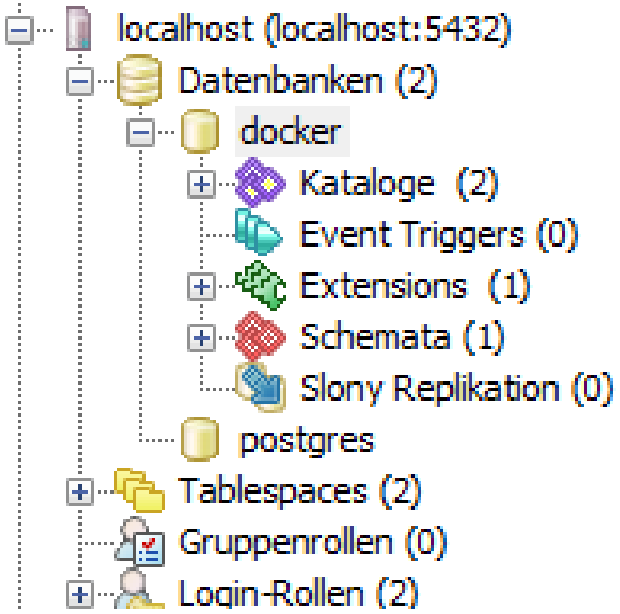
Test

5. PgAdmin3:
    1. localhost:5432
    2. user/pass: docker/docker

    Oder: psql -h localhost -U docker -d docker

2. PostgreSQL Container Solution

- cd /docker/solution/postgres
- `docker build -t tmf_postgres`
- ```
  docker run
  -it
  -p 5432:5432
  --name tmf_postgres
  tmf_postgres
  ```



Test

- psql -h localhost -U docker -d docker
- \list
  ```

Praktische Übung 2

Orchestrierung von Containern

Docker Compose

- Wofür?
- Viele Container haben Abhängigkeiten
- Wordpress braucht z.B. MySQL
- Container gleichzeitig starten/stoppen

- docker-compose.yml — Datei für Instruktionen
- docker-compose up — Startet Container

Docker Compose

- Registrierung eigener Services
- Abhängigkeiten zu anderen Services
- Übergabe von Umgebungsvariablen
- Verlinkung von Containern
- Öffnen von Ports

3. Wordpress

3. Wordpress

```
version: '2'
services:
  db:
    image: mysql:5.7
    volumes:
      - "./.data/db:/var/lib/mysql"
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: wordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

wordpress:
    depends_on:
      - db
    image: wordpress:latest
    links:
      - db
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_PASSWORD: wordpress
```

3. Wordpress Container

1. Navigieren Sie in den Ordner /docker/excercise/wordpress_compose
2. Starten sie Wordpress
3. Öffnen Sie die Webapplikation in Ihrem Browser http://localhost:5800
4. Beenden Sie die Wordpress-Installation im Browser

3. Wordpress Solution

- cd /docker/solution/wordpress_compose
- `docker-compose up`