Schulstunde am 15.04.2024 mit indibit zum Thema Docker

Agenda

1. Was ist Docker?
2. Docker vs. Virtuelle Maschinen
3. Starten des Containers
4. Docker-Begriffe
5. Docker-compose
6. Docker Swarm
7. Docker-compose Projekte starten

Voraussetzungen

Internet – zum Beziehen der Images

Docker-Desktop

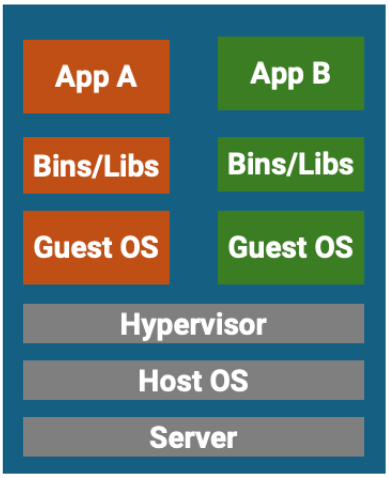
Visual Studio Code

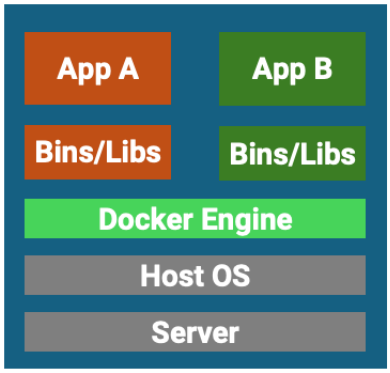
1. Was ist Docker?

- Eine Open-Source-Plattform, die die Erstellung, Bereitstellung und Ausführung von Anwendungen in Containern erleichtert.
- Container sind eine Art Virtualisierung, die es ermöglicht, Anwendungen und ihre Abhängigkeiten in einem isolierten Umfeld auszuführen.
- Docker bietet eine einfache Möglichkeit, Container zu erstellen, zu verwalten und zu skalieren, indem es eine Reihe von Tools und APIs bereitstellt.
- Diese Container sind portabel und können auf jedem System ausgeführt werden, das Docker unterstützt. Es ist keine zusätzliche Konfiguration erforderlich.

2. Docker vs. Virtuelle Maschinen

Vorteile	Nachteile
Leichter einzurichten	Läuft bei Windows „nur“ im Subsystem, nicht wie bei Linux/Mac nativ
Unabhängig von Betriebssystem	Teilweise nicht ausführlich dokumentiert
Container haben bereits alles integriert, was zum Betrieb nötig ist	Bedienung erfolgt meist ohne jegliche UI via CLI
Einfachere Skalierung (starten des Containers x100 problemlos möglich)	
Ressourcenschonender	





3. Starten des Containers

- Starten eines default nginx-containers
  `docker run --name docker-nginx -p 80:80 nginx`

Was ist beim Starten des Containers passiert?

- Image wird lokal nicht gefunden -> wird von Dockerhub bezogen: https://hub.docker.com/_/nginx
- die einzelnen Layer werden heruntergeladen
- der Container wird gestartet
- Wir können den Container erreichen

4. Docker-Begriffe

- Docker-Engine -> kommuniziert mit dem Host-System
- Docker-API -> nimmt API-Anfragen entgegen
- Docker-registry -> verwaltet Docker-Images
- Dockerfile -> definiert den Aufbau eines Containers
- Docker-compose: yaml-Struktur, um einzelne oder mehrere Docker Container einfacher verwalten zu können
- Docker-swarm: Clustering von Servern & Verteilung von Container

- Dockerfile: definiert den Aufbau eines Containers

Bsp:

```
FROM IMAGE:TAG
WORKDIR <ARBEITSVERZEICHNIS>
COPY <VON> <NACH>
RUN <build Befehl>
CMD echo "BLUB"
EXPOSE 80
```

Build und Versions-Tagging:

Docker File:

```
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
COPY ./simple.gif /usr/share/nginx/html/simple.gif
EXPOSE 80
```

Image bauen:

```
docker build -t nginx:wiesau .
```

Alle Images listen:

```
docker image ls
```

```
REPOSITORY       TAG          IMAGE ID       CREATED              SIZE
docker-gs-ping   wiesau       e3fdde09f172   About a minute ago   28.1MB
```

5. Docker-compose

Aufbau einer `docker-compose.yml`:

```
version: '3.8'
services:
 minecraft:
  image: minecraft:1.9
  container_name: minecraftserver
  restart: unless-stopped
  volumes:
   - minecraft-data:/opt/minecraftserver/data
   - ./backupdata:/opt/minecraftserver/backup
 environment:
    SERVERNAME: „lan-server“
    SERVERPASSWORT: „lan_1337“
    EXTERNAL_PORT: „25570
    EULA: „TRUE“
 ports:
  - „25570:„25565
 networks:
  - minecraft-net
volumes:
 minecraft-data:
networks:
 minecraft-net:
  driver: host
```

Bedienung:

- docker compose config: zeigt die ausgefüllte Docker-compose config an
- docker compose up (-d): startet alle Services der Docker-compose inkl. Netzwerke, zieht images falls diese nicht lokal vorhanden sind, „-d“ startet die definierten Services im Hintergrund
- docker compose down: stoppt alle Services/netzwerke, welche in der Docker-compose definiert sind
- docker compose pull: lädt alle Images, welche in der Docker-compose definiert sind#

6. Docker Swarm

- Verbinden von mehreren Servern
- Verteilung von Ressourcen (containern)
- Duplizierung von Services
- Ausfallsicherheit für Services

Einrichtung:

```
indibit@MacBook-Pro ~ % docker swarm init
Swarm initialized: current node (k7ey1t6yugwpnzfm1z4wud96t) is now
a manager.

To add a worker to this swarm, run the following command:

docker swarm join --token SWMTKN-1-
53sctnza8j3wrfrsf884eylfd96yci1cp4hg388rj5d9lc7et9-
4lkom3f3ye054e17l4kufedkp 192.168.65.3:2377
```

```
To add a manager to this swarm, run 'docker swarm join-token
manager' and follow the instructions.
```

Nodetypes:

- Manager-node : verteilt und steuert den swarm (ist gleichzeitig eine Leader-node)
- Leader-node : es gibt jeweils eine Leader-Node, wenn eine ausfällt wird eine andere Manager-node zum leader (falls definiert)
- Worker-node : Nimmt Aufgaben der Leader-node entgegen

Unterschiede zu Docker-Befehlen:

- `Docker service inspect` (inspiziert einen Service -> docker inspect -> lokal)
- `Docker service ls` (listet alle Services)
- `Docker service rm` (entfernt einen Service docker stop -> lokal)
- `Docker service scale` (skaliert einen Services)
- `Docker service ps` (Pendant zu docker ps -> lokal)
- `Docker service update`

7. Docker-compose Projekte starten

Wie starte ich container?

- via `docker run`
- via `docker compose up` (-d)

Wie stoppe ich

- via `docker stop`
- via `docker compose down`

Wie kann ich im Container arbeiten?

- via `docker ps` ermitteln
  - `docker compose exec –it <containerid>`

Containerlogs verfolgen:

- `docker logs <container>`

Viel Spaß beim Ausprobieren.

Schaue auch auf unserer Website indibit.eu/karriere vorbei, wenn du Lust hast Teil des indibit Teams zu werden. ☺