Container starten & verwalten

- docker run [OPTIONS] IMAGE [COMMAND] ➤ Startet einen neuen Container aus einem Image.

Beispiele mit Optionen:

  - -it → Interaktiv + Terminal (docker run -it ubuntu bash)
  - -d → Im Hintergrund (detached) (docker run -d nginx)
  - --name → Container benennen (--name webapp)
  - --rm → Nach dem Stoppen löschen (--rm)
  - -p → Portweiterleitung (-p 8080:80)
  - -v → Volume mounten (-v meinvolume:/app/data)
  - -e → Umgebungsvariable setzen (-e MODE=prod)
  - --network → Netzwerk zuweisen (--network=my_net)
  - --ip → Statische IP im benutzerdefinierten Netzwerk (--ip=10.10.10.10)

- docker start <container>
  ➤ Startet einen gestoppten Container wieder.
- docker stop <container>
  ➤ Beendet einen laufenden Container.
- docker restart <container>
  ➤ Stoppt und startet den Container neu.
- docker kill <container>
  ➤ Erzwingt das Beenden (SIGKILL).
- docker rm <container>
  ➤ Löscht einen gestoppten Container.
- docker ps
  ➤ Zeigt laufende Container.
- docker ps -a
  ➤ Zeigt **alle** Container (auch gestoppte).

Images verwalten

- docker pull <image>
  ➤ Lädt ein Image von Docker Hub herunter.
- docker images
  ➤ Listet alle lokal gespeicherten Images.

- docker rmi <image>
  ➤ Löscht ein Image vom System.
- docker tag <image> <repo>:<tag>
  ➤ Gibt einem Image einen neuen Namen + Version (z. B. meinimage:v1).

Image selbst bauen (Dockerfile)

- docker build -t <name> .
  ➤ Erstellt ein Image aus dem Dockerfile im aktuellen Verzeichnis.

Option:

  - -t → Gibt dem Image einen Namen (-t meine_app)

Dateien & Volumes

- docker cp <container>:/pfad/zur/datei .
  ➤ Kopiert Dateien **aus dem Container** auf den Host.
- docker run -v volume_name:/app/data
  ➤ Bindet ein **benanntes Volume** in den Container ein.
- docker run -v /host/pfad:/container/pfad
  ➤ Bindet ein **hostbasiertes Verzeichnis** in den Container ein.

Container-Inspektion & Debugging

- docker exec -it <container> bash
  ➤ Führt einen Befehl **im laufenden Container** aus (z. B. Shell öffnen).
- docker logs <container>
  ➤ Zeigt die Log-Ausgabe eines Containers.
- docker inspect <container oder image>
  ➤ Zeigt alle Metadaten als JSON (z. B. IP, Volumes etc.).
- docker container ls
  ➤ Alias für docker ps.

Netzwerke

- docker network ls
  ➤ Listet alle Docker-Netzwerke.
- docker network create --subnet=... --gateway=... <name>
  ➤ Erstellt ein benutzerdefiniertes Netzwerk.

- docker network inspect <name>
  ➤ Zeigt Details zum Netzwerk (z. B. IPs aller Container).
- docker network connect <netz> <container>
  ➤ Verbindet einen Container mit einem Netzwerk.
- docker network disconnect <netz> <container>
  ➤ Trennt einen Container vom Netzwerk.
- docker network rm <name>
  ➤ Löscht ein benutzerdefiniertes Netzwerk.

Dateien aus dem Video

app.js

```
const express = require('express');

const app = express();

const PORT = process.env.PORT || 3000;


app.get('/', (req, res) => {
  res.send('Hello, World!');
});


app.listen(PORT, () => {
  console.log(` Server is running on port ${PORT}` );
});
```

package.json

```
{
  "name": "docker-node-app",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "express": "^4.18.2"
  }
```

}

.dockerignore

node_modules

npm-debug.log

Dockerfile

FROM node:14

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "app.js"]

Start & Build aus dem Video

npm init -y

npm install express

docker build -t node-app .

docker run -p 3000:3000 node-app

docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:2.21.5