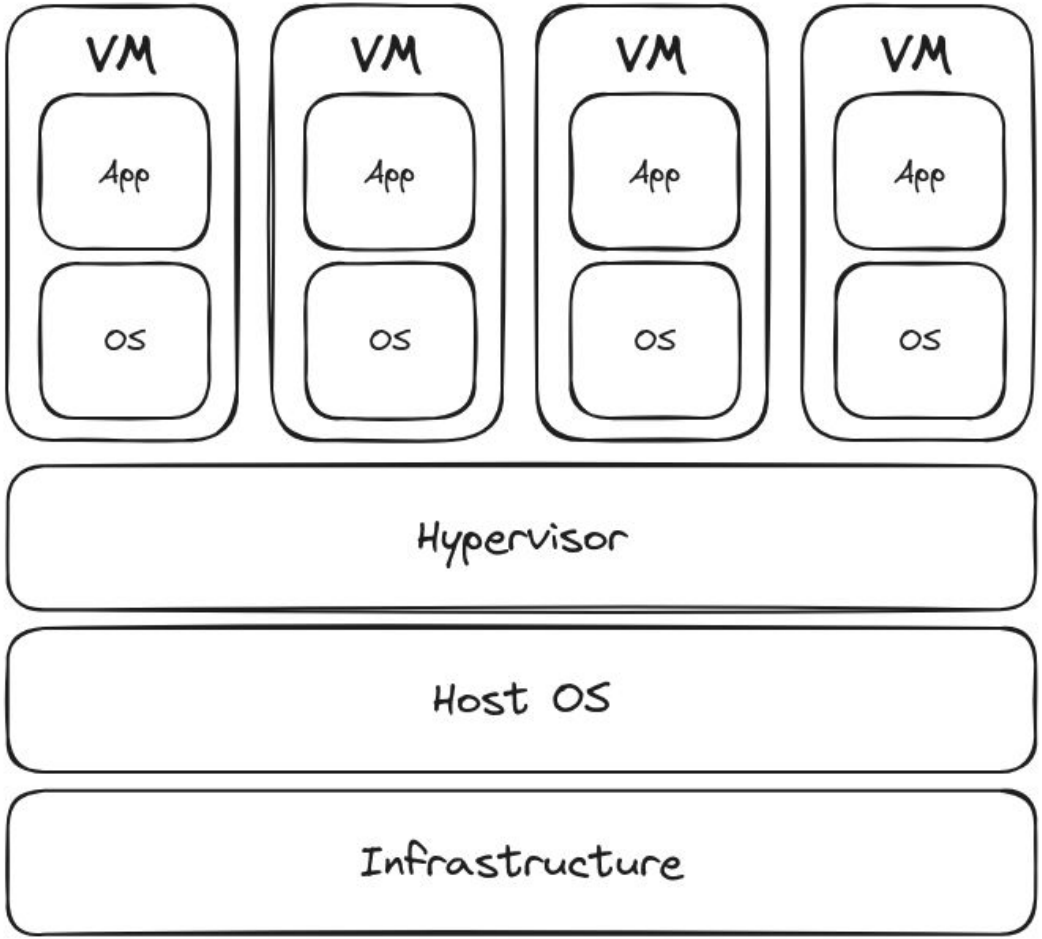
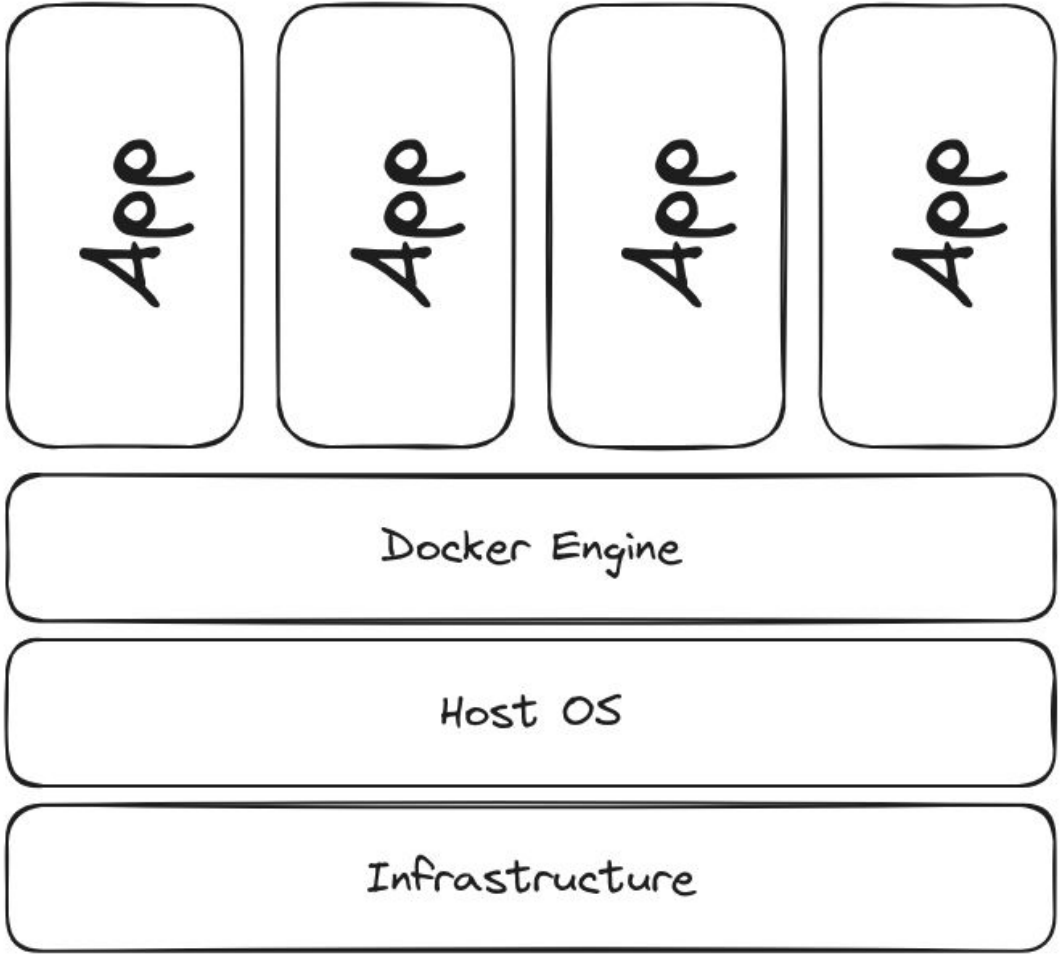
Einstieg in Containerdeployments mit Docker

Was sind VMs?

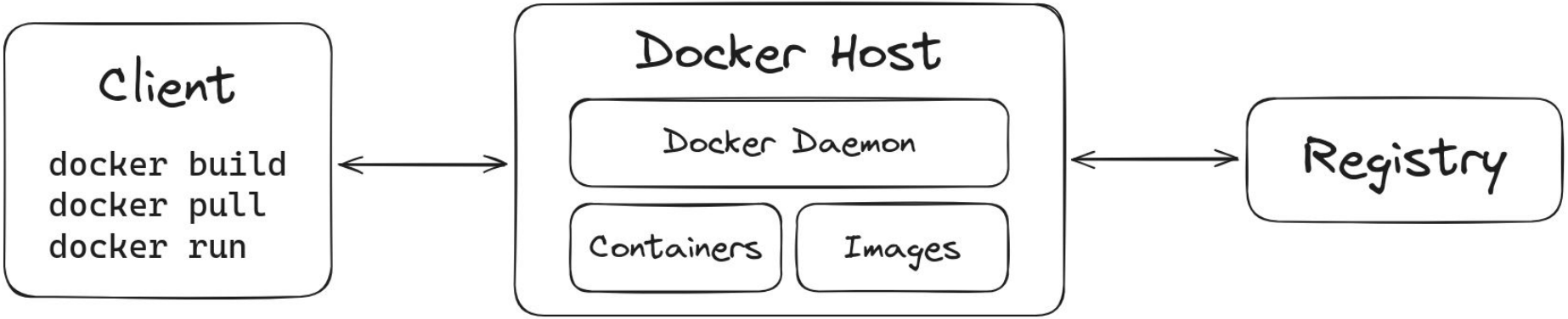
- Virtual Machines
- Virtuelle Maschinen



Was ist Docker?



Wie arbeitet Docker?



Images und Container

Container starten

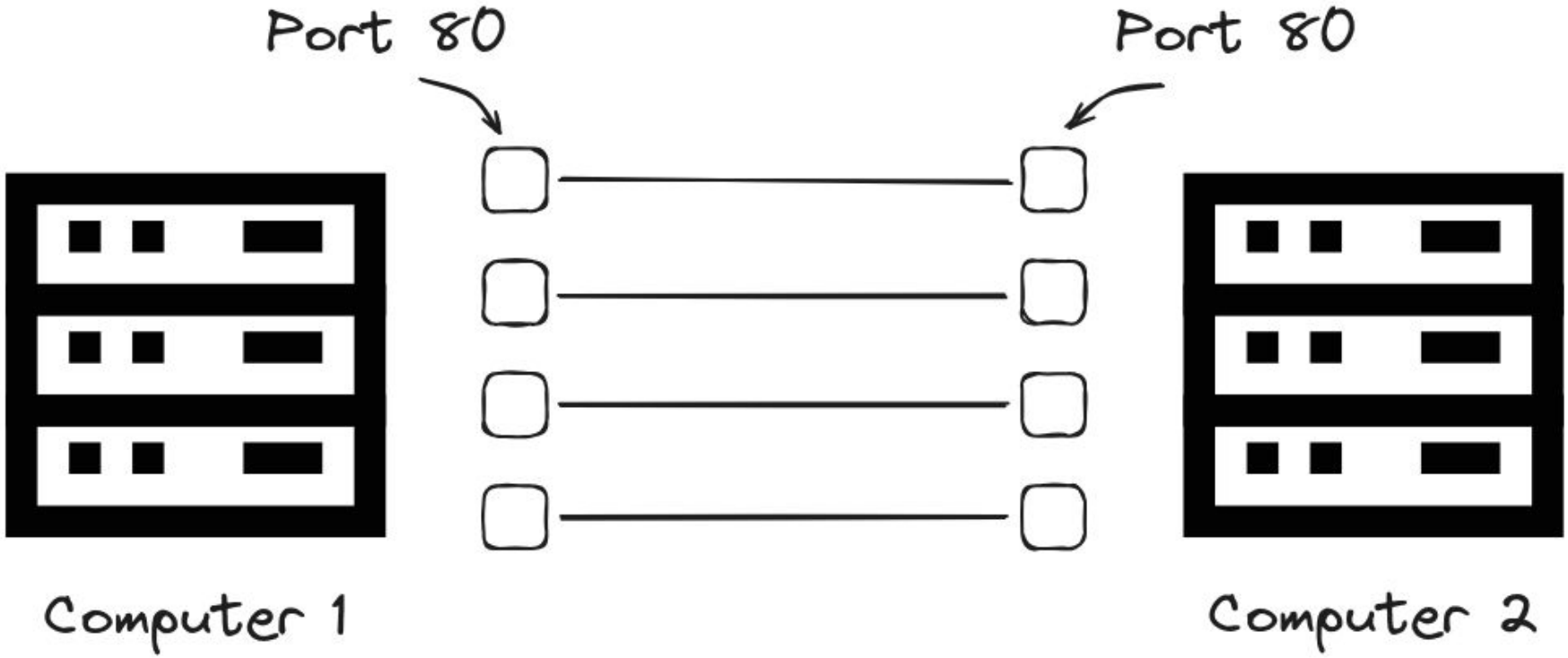
```
docker run nginx
```

1. Image herunterladen
2. Container erstellen
3. Container starten

Offizielle Dokumentation

Intermezzo: HTTP und Webserver

Intermezzo: TCP ports



Ports in Docker

- Container benötigen Portweiterleitungen
  - “publish port”
- Um auf nginx zugreifen zu können:

```
docker run -p 8080:80 nginx
```

http://localhost:8080

Offizielle Dokumentation

Tags

```
docker run nginx:stable
```

- verschiedene Versionen können mit Tags markiert werden
- vgl. mit Git Tags

nginx im Docker Hub

Containerbefehle

```
docker run -d nginx          # Container detached starten

docker ps                    # Laufende Container auflisten

docker logs <container-id># Log eines Containers anzeigen
```

Container und Images löschen

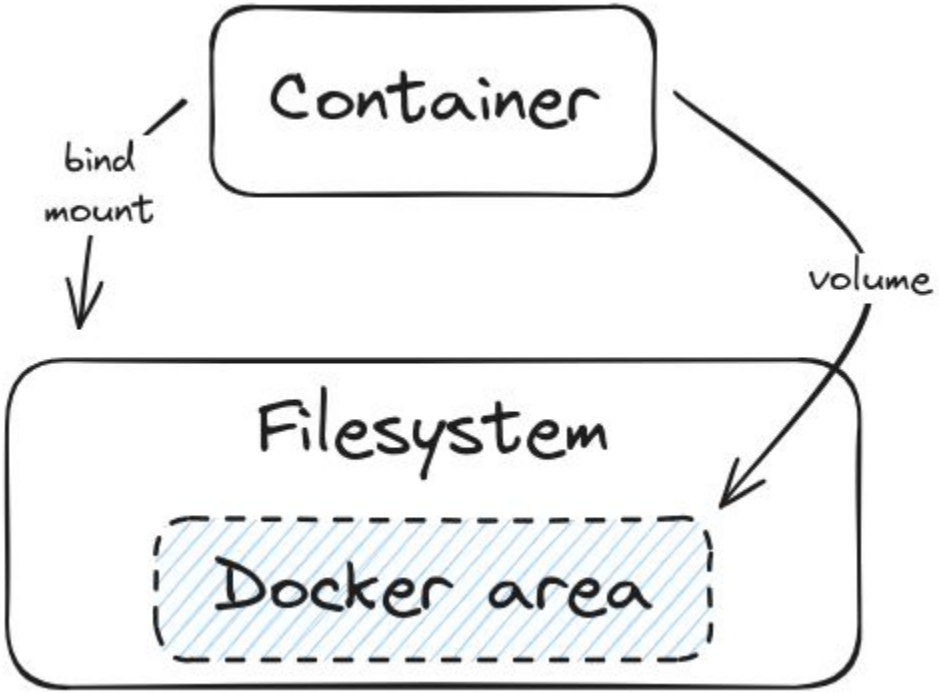
```
docker stop <container-id>    # Container stoppen

docker rm <container-id>      # Container löschen

docker images                 # Images auflisten

docker rmi <image-id>         # Image löschen
```

Bind mounts

- Verzeichnisse in Container mounten

```
mkdir blog && cd blog
touch index.html
docker run -p 8080:80 \
    -v $(PWD)/blog:/usr/share/nginx/html:ro \
    nginx
```

Docker Volumes



Beispiel: Volumes

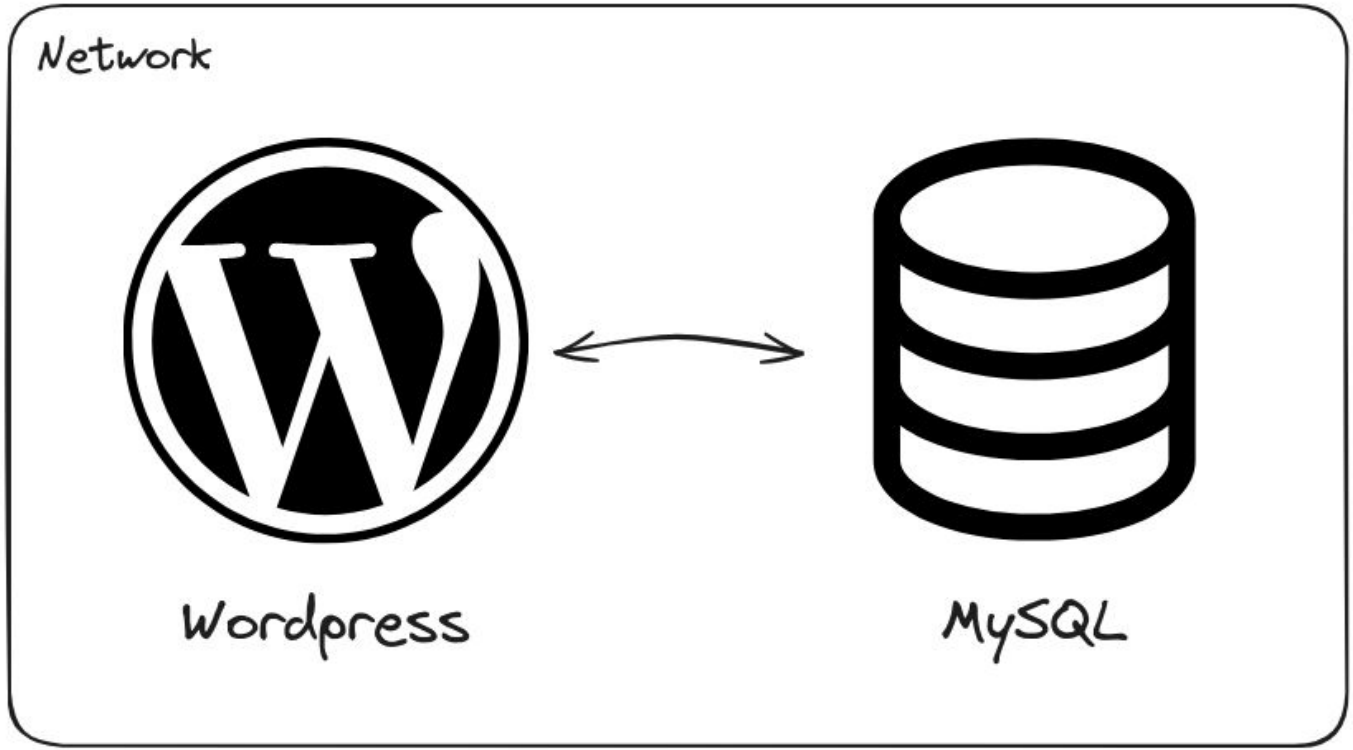
```
docker volume create db-volume
docker run -d \
    --name my-postgres \
    -e POSTGRES_PASSWORD=mysecretpassword \
    -e PGDATA=/var/lib/postgresql/data/pgdata \
    -v db-volume:/var/lib/postgresql/data \
    postgres
```

Offizielle Dokumentation

Postgres testen

```
docker exec -it my-postgres bash

psql -h localhost -U postgres


\list
SELECT usename, passwd from pg_shadow;
```

Wordpress-Deployment



Wordpress mit Docker

```
docker network create wordpress
docker volume create mysql-volume

docker run \
    --network wordpress \
    --network-alias mysql \
    -e MYSQL_DATABASE=wordpress \
    -e MYSQL_ROOT_PASSWORD="super_secret" \
    -v mysql-volume:/var/lib/mysql \
    mysql
```

```
docker run -p 8080:80 \
    --network wordpress \
    -e WORDPRESS_DB_HOST=mysql \
    -e WORDPRESS_DB_USER=root \
    -e WORDPRESS_DB_NAME=wordpress \
    -e WORDPRESS_DB_PASSWORD="super_secret" \
    wordpress
```

Wordpress mit Docker-Compose

```
version: "3"
services:
    wordpress:
        image: wordpress
        platform: linux/amd64
        ports:
            - "8080:80"
        depends_on:
            - mysql
        environment:
            WORDPRESS_DB_HOST: mysql
            WORDPRESS_DB_USER: root
            WORDPRESS_DB_NAME: wordpress
            WORDPRESS_DB_PASSWORD: "super_secret"
    mysql:
        image: "mysql"
        platform: linux/amd64
        environment:
            MYSQL_DATABASE: wordpress
            MYSQL_ROOT_PASSWORD: "super_secret"
        volumes:
          - db-volume:/var/lib/mysql

volumes:
    db-volume:
```

Dockerfiles

```
FROM nginx

COPY blog /usr/share/nginx/html
```

Build docker image

```
docker build -t mein-blog .          # build docker image

docker run -p 8080:80 mein-blog      # run docker container
```

Repositories

Repository anlegen auf https://hub.docker.com

```
docker build -t <repository-name>/<image-name> .

docker login

docker push <repository-name>
```

Offizielle Dokumentation

Reference Material

Docker Documentation

Docker 101 Tutorial

Python mit Docker

Docker Tool Container

How Docker Works - Intro to Namespaces

Deepdive Containers - Kernel Sources and nsenter

Cheat sheet