Containerdeployments mit Docker

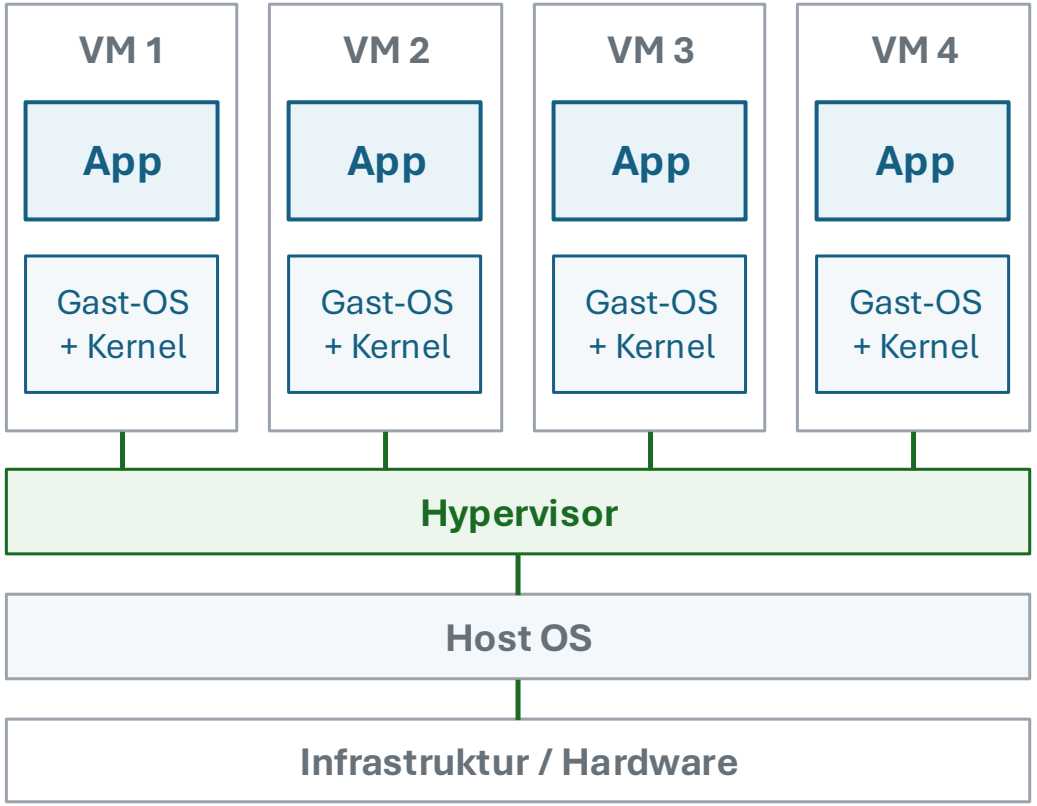
essembly

Vorbereitung

- haben alle Docker (Desktop) installiert?
- läuft Docker (Desktop) bei allen?
- User in docker group?
- Was sagt `docker --version`?

Was sind Virtuelle Machinen?

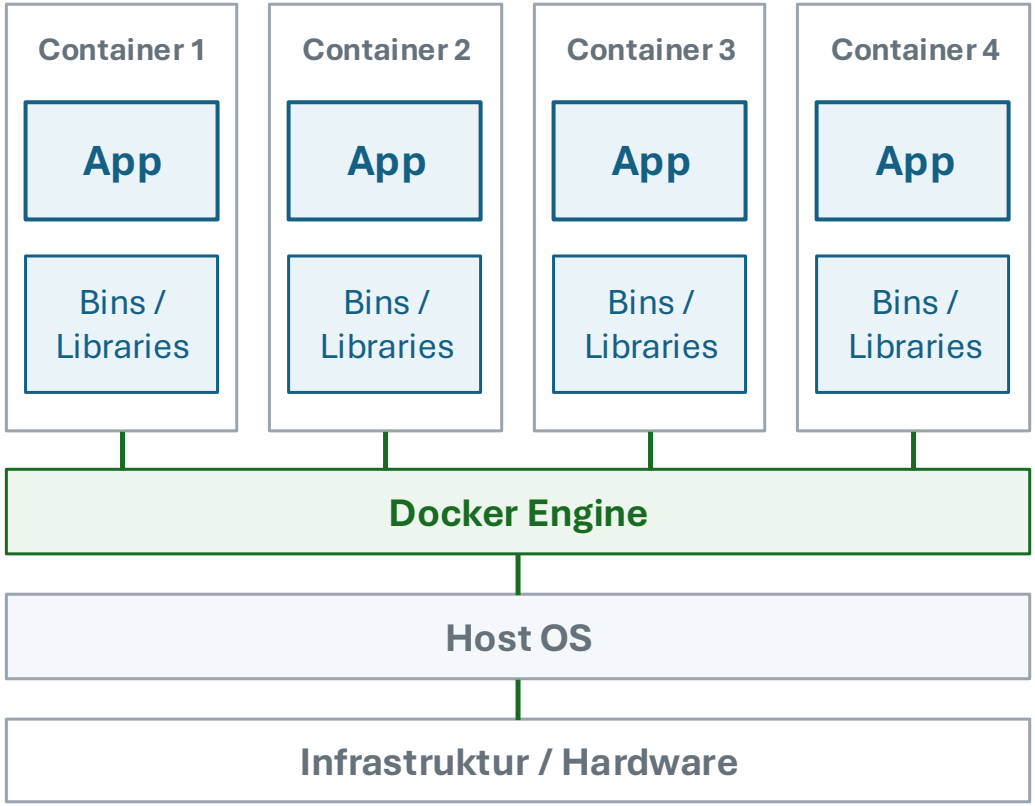
- Eigenes OS pro VM
- Höherer Ressourcen-bedarf
- Starke Isolation der VMs



essembly

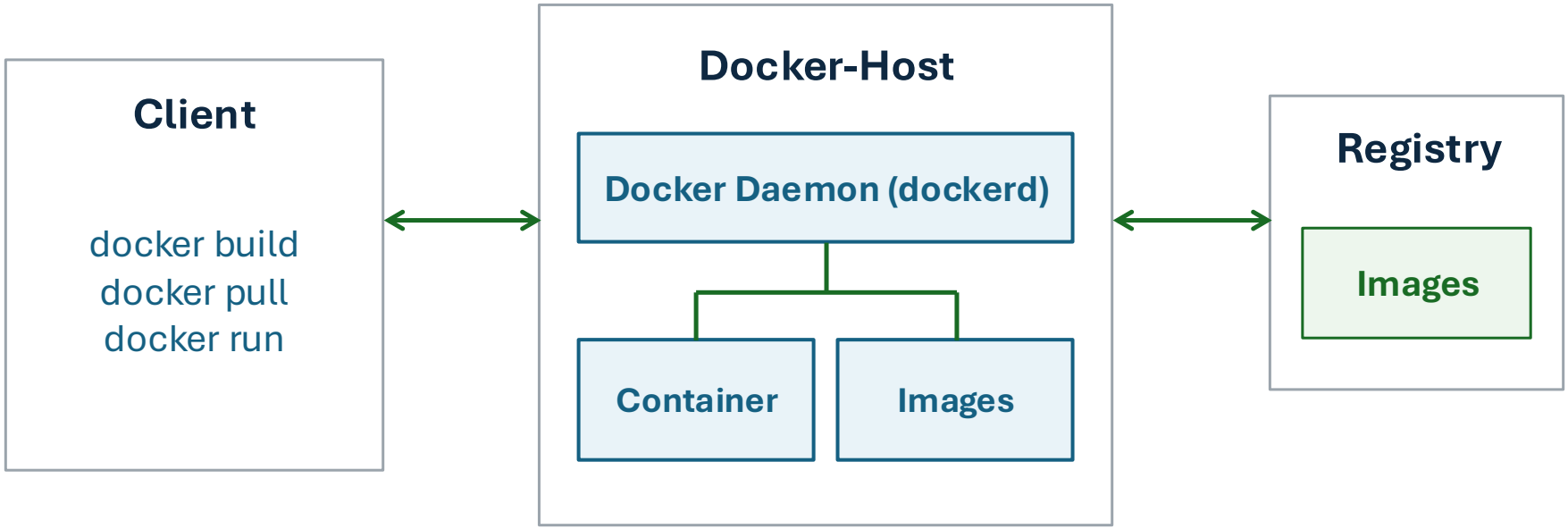
Was macht Docker anders?

- Container teilen sich einen Kernel
- Geringerer Ressourcenbedarf
- Leichtgewichtigere Isolation



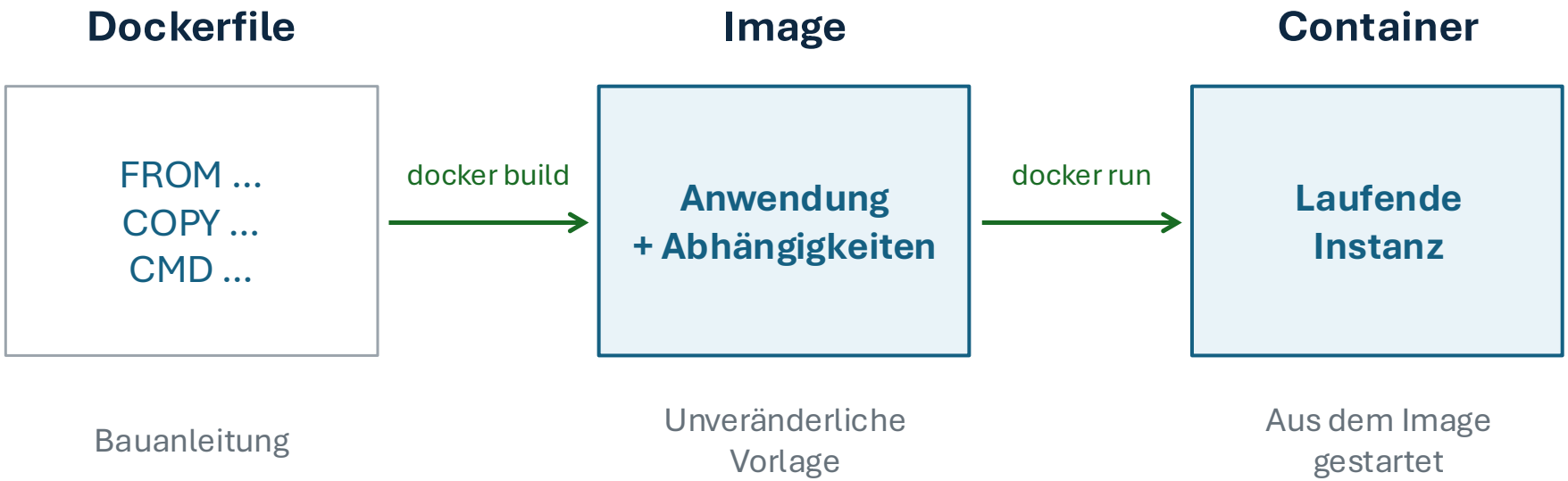
essembly

Wie arbeitet Docker?



essembly

Images und Container



essembly

Container starten

```
docker run nginx # nginx ist das image
```

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

Dokumentation

Inspect and debug 1

```
# Laufende Container auflisten
docker ps

# Alle Container auflisten
docker ps -a
```


essembly

Inspect and debug 2

```
# Logs anzeigen
docker logs <container name or hash>

# Detaillierte Informationen über Container
docker inspect <container name or hash>
```


essembly

Ports

Ports müssen explizit freigegeben werden:

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

Danach ist nginx unter http://localhost:8080 erreichbar

UDP geht auch: `docker run -p 1234:42/udp nginx:latest`

Offizielle Dokumentation

essembly

Container stoppen/ entfernen

```
# Im Hintergrund starten
docker run -d <image>

# Container anhalten
docker stop <container name or hash>

# Container entfernen
docker rm <container name or hash>
```


essembly

Images

```
# Images auflisten
docker images

# Image löschen
docker rmi <image name or hash>
```


essembly

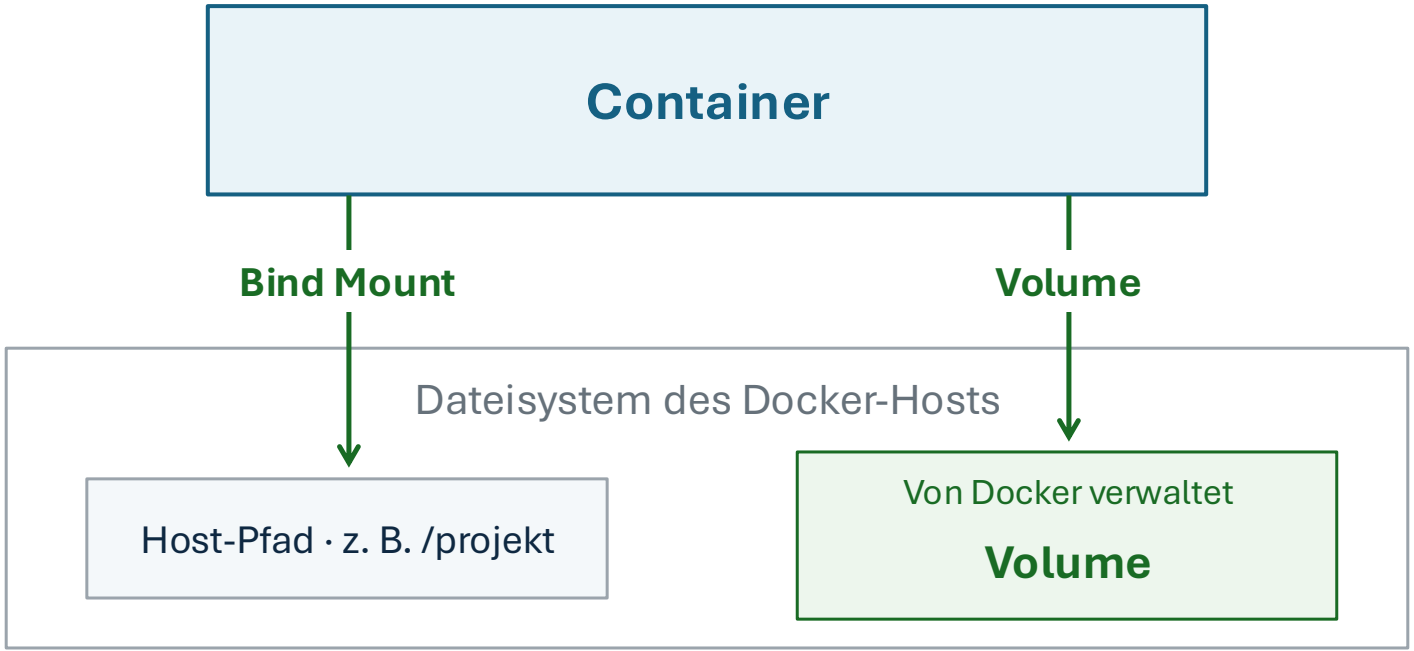
Tags

```
docker run nginx:stable
```

- verschiedene Versionen können mit Tags markiert werden
- vergleichbar mit Git Tags

nginx im Docker Hub

Persistente Daten



essembly

Bind mounts

```
mkdir blog
echo '<h1>Hallo Essembly</h1>' > blog/index.html

docker run -p 8080:80 \
    -v $(pwd)/blog:/usr/share/nginx/html:ro \
    nginx
```


essembly

Named 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
```

Dokumentation für docker volumes

essembly

Volumes

```
# Volumes auflisten
docker volume ls

# Volume löschen
docker volume rm <name or hash>

# Ungenutzte volumes entfernen
docker volume prune
```


essembly

Container starten 2

```
docker run -d \                 # Run container in background
  --name=dokuwiki \             # Assign a name to the container
  -e PUID=1000 \                # Set environment variable
  -e PGID=1000 \
  -e TZ=Etc/UTC \
  -p 80:80 \                    # Publish container’s port(s) to host
  -p 443:443 \
  -v ./config:/config:rw        # Bind mount a volume
  --rm \                        # Remove container and volumes
  --restart unless-stopped \
  lscr.io/linuxserver/dokuwiki:latest    # Image:Tag
```

Dokumentation für linuxserver/dokuwiki

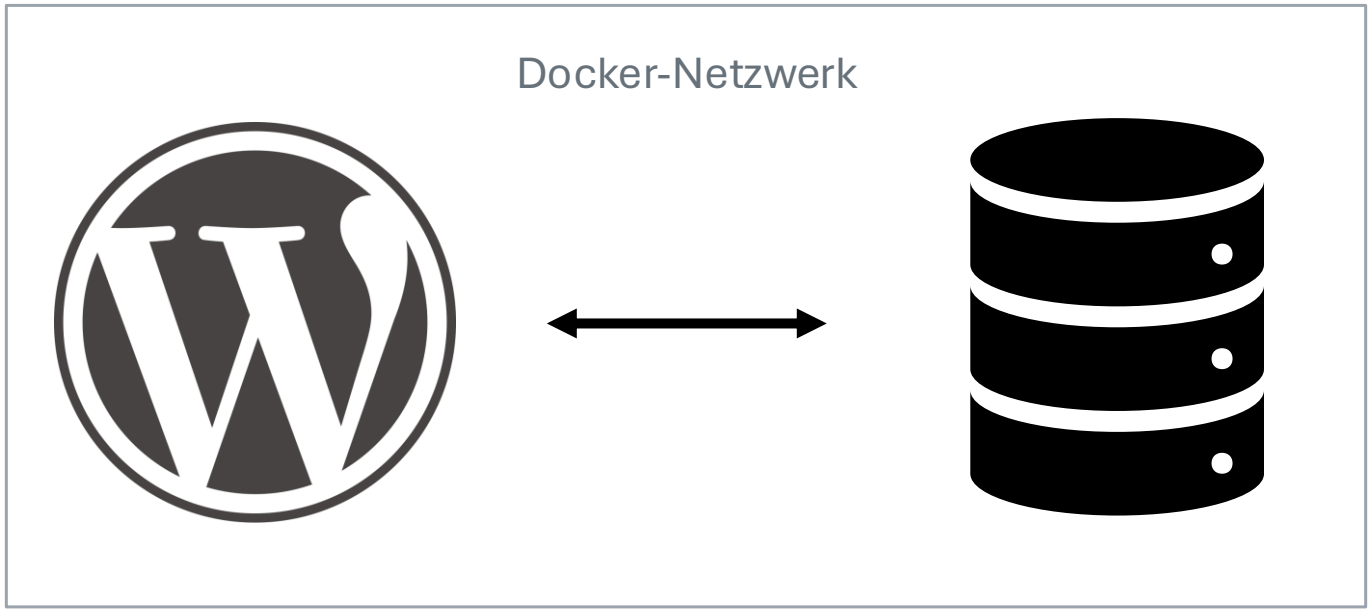
essembly

Restart

Flag	Beschreibung
no	Den Container nicht automatisch neu starten. (Standard)
on-failure[:max-retries]	Den Container neustarten, wenn er aufgrund eines Fehlers beendet wird (Exit-Code ungleich Null).
always	Den Container immer neu starten. Wenn er manuell angehalten wurde, wird er erst dann neu gestartet, wenn der Docker-Daemon neu gestartet wird oder der Container selbst manuell neu gestartet wird.
unless-stopped	Ähnlich wie bei „always“, mit dem Unterschied, dass der Container, wenn er (manuell oder anderweitig) angehalten wird, auch nach einem Neustart des Docker-Daemons nicht neu gestartet wird.

essembly

Wordpress-Deployment



essembly

Networks

Eigenes bridge Network erstellen

```
docker network create mein-netz
```

Host-Netzwerk verwenden

```
docker run --network host nginx
```

Netzwerk deaktivieren

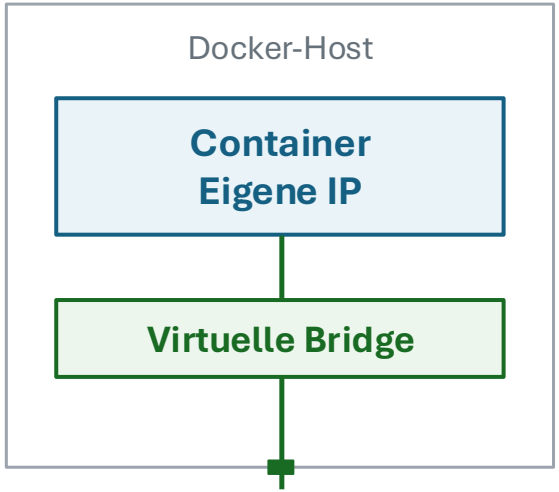
```
docker run --network none alpine
```


essembly

Networks

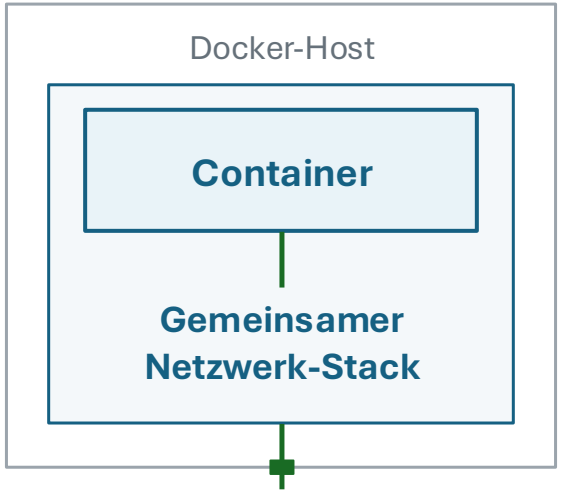
Networking Dokumentation Deepdive

bridge



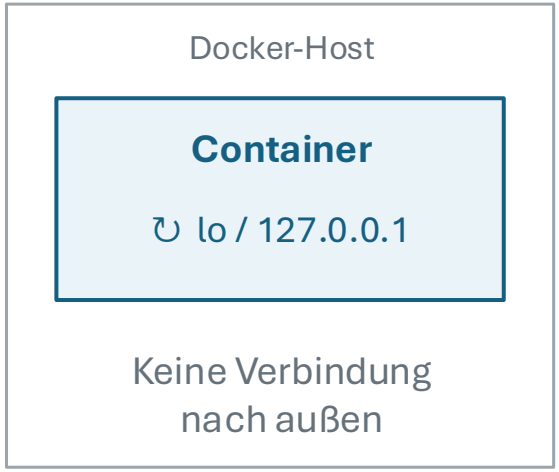
Eigene IP
Ports mit -p freigeben

host



Host-IP und Ports gemeinsam
Kein Port-Mapping

none

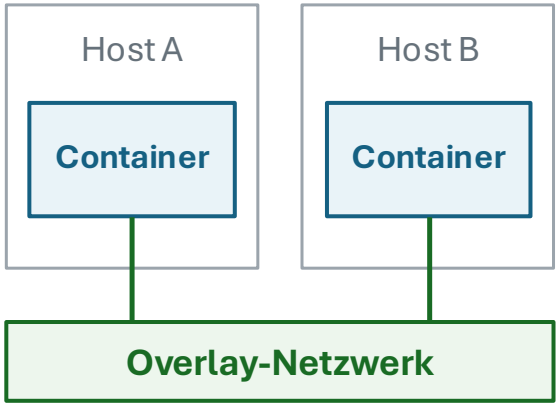


Nur Loopback
Kein Netzwerkzugriff

essembly

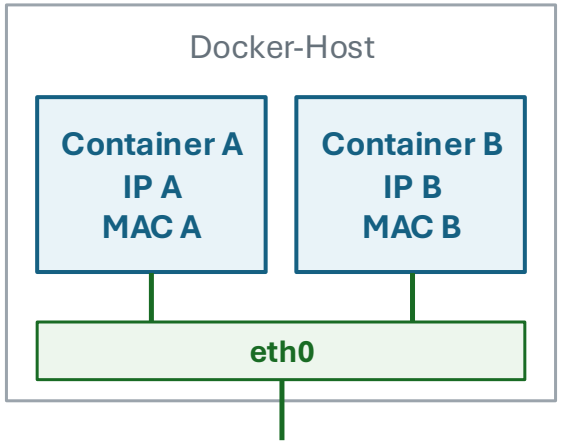
Weitere Netzwerktypen

overlay



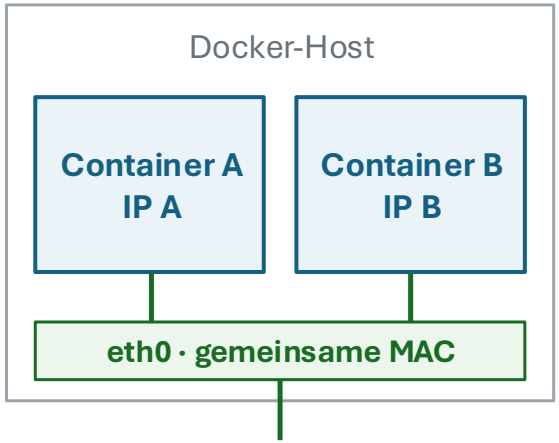
Docker Swarm
über mehrere Hosts

macvlan



Eigene IP und eigene MAC
pro Container

ipvlan



Eigene IP pro Container
MAC des Host-Interfaces

essembly

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
```


essembly

Wordpress mit Docker-Compose

```yaml
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"
    networks:
      - wordpress

  mysql:
    image: "mysql"
    platform: linux/amd64
    environment:
      MYSQL_DATABASE: wordpress
      MYSQL_ROOT_PASSWORD: "super_secret"
    volumes:
      - db-volume:/var/lib/mysql
    networks:
      - wordpress

volumes:
  db-volume:

networks:
  wordpress:
```


essembly

Dockerfile

```
FROM nginx
COPY index.html /usr/share/nginx/html/
```

Build docker image

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

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

Dokumentation

essembly

Keywords in Dockerfiles 1

FROM	Legt das Basis-Image fest.
WORKDIR	Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile
COPY	Kopiert Dateien und Verzeichnisse ins Image.
ADD	Kopiert Dateien und kann Archive automatisch entpacken oder URLs abrufen. Dokumentation der Unterschiede ADD / COPY
ENV	Setzt Umgebungsvariablen im Image bzw. Container.
ARG	Definiert Variablen für den Image-Build. Dokumentation für Build Arguments

Dokumentation

essembly

Keywords in Dockerfiles 2

RUN	Führt Befehle beim Erstellen des Images aus.
EXPOSE	Dokumentiert den vorgesehenen Container-Port.
USER	Legt den Benutzer für nachfolgende Befehle und den Container fest.
LABEL	Fügt dem Image Metadaten hinzu.
CMD	Legt den Standardbefehl beim Containerstart fest.
ENTRYPOINT	Legt das ausführbare Hauptprogramm des Containers fest.

Dokumentation

essembly

CMD und ENTRYPOINT

```
FROM alpine
CMD [”echo”, “Hallo”]


docker run demo
# echo Hallo

docker run demo echo Ade
# echo Ade
```

```
FROM alpine
ENTRYPOINT [“echo”]
CMD [”Hallo”]


docker run demo
# echo Hallo

docker run demo Ade
# echo Ade
```


essembly

Layers 1

```
FROM python:3.12-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .
CMD ["python", "app.py"]
```

Layers 2

```
docker build -t meine-app .
docker image history meine-app
```

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
f726f749579f	47 seconds ago	CMD ["python" "app.py"]	0B	buildkit.dockerfile.v0
<missing>	47 seconds ago	COPY app.py . # buildkit	8.19kB	buildkit.dockerfile.v0
<missing>	47 seconds ago	RUN /bin/sh -c pip install --no-cache-dir -r…	9.63MB	buildkit.dockerfile.v0
<missing>	48 seconds ago	COPY requirements.txt . # buildkit	8.19kB	buildkit.dockerfile.v0
<missing>	48 seconds ago	WORKDIR /app	8.19kB	buildkit.dockerfile.v0
<missing>	2 days ago	CMD ["python3"]	0B	buildkit.dockerfile.v0
<missing>	2 days ago	RUN /bin/sh -c set -eux; for src in idle3 p…	16.4kB	buildkit.dockerfile.v0
<missing>	2 days ago	RUN /bin/sh -c set -eux; savedAptMark="$(a…	44.6MB	buildkit.dockerfile.v0
<missing>	2 days ago	ENV PYTHON_SHA256=c2c4321961fab0fb999d66e0ce…	0B	buildkit.dockerfile.v0
<missing>	2 days ago	ENV PYTHON_VERSION=3.12.15	0B	buildkit.dockerfile.v0
<missing>	2 days ago	ENV GPG_KEY=7169605F62C751356D054A26A821E680…	0B	buildkit.dockerfile.v0
<missing>	2 days ago	RUN /bin/sh -c set -eux; apt-get update; a…	4.97MB	buildkit.dockerfile.v0
<missing>	2 days ago	ENV LANG=C.UTF-8	0B	buildkit.dockerfile.v0
<missing>	2 days ago	ENV PATH=/usr/local/bin:/usr/local/sbin:/usr…	0B	buildkit.dockerfile.v0
<missing>	2 days ago	# debian.sh --arch 'arm64' out/ 'trixie' '@1…	110MB	debuerreotype 0.17

essembly

Multi-stage build

```
# Stage 1: Anwendung bauen

FROM maven AS build
WORKDIR /app

COPY pom.xml .
COPY src ./src

RUN mvn package -DskipTests
```

```
# Stage 2: App ausführen

FROM eclipse-temurin:21-jre
WORKDIR /app

COPY --from=build
/app/target/*.jar app.jar

CMD ["java", "-jar",
"app.jar"]
```


essembly

Warum?

Ohne Multi-Stage Build

- Build-Tools im finalen Image
- Größeres Image
- Mehr unnötige Dateien
- Größere potenzielle Angriffsfläche

Mit Multi-Stage Build

- Build-Tools bleiben zurück
- Meist kleineres Image
- Weniger unnötige Dateien
- Kleinere potenzielle Angriffsfläche

essembly

Dockerfile ohne parent image

```
FROM scratch
ADD app /
CMD ["/app"]
```


essembly

Repositories

Repository anlegen (bspw. auf https://hub.docker.com)

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

Offizielle Dokumentation

essembly

Interessante Links

- Docker 101 Tutorial
- Heise über Docker Image Security
- Hack The Clown on Docker Security
- Docker for CTFs
- Docker für lokale Entwicklung
- Docker CLI Cheat Sheet

essembly