Docker

1 Version

```
docker version                          # installierte Version anzeigen
```

2 Images verwalten

Image-Namen

`<user>/<image>:<tag>`

- Default-User: `library`
- Default-Tag: `latest`

Kommandos

```
docker image pull <image>               # Image herunterladen
docker image build -t <image>:<tag> .   # Image von Dockerfile bauen
docker image ls                         # heruntergeladene Images anzeigen
docker image rm <image>                 # Image löschen
```

3 Container verwalten

```
docker container ls                     # alle laufenden Container anzeigen
docker container ls -a                  # auch Container, die nicht laufen, anzeigen
docker container stop <container>       # laufenden Container stoppen
docker container start <container>      # Container wieder starten
docker container inspect <container>    # Details des Containers anzeigen
docker container rm <container>         # Container löschen
docker exec -it <container> KOMANDO     # KOMMANDO im laufenden Container ausführen
```

4 Volumes verwalten

```
docker volume ls                        # alle Volumes anzeigen
docker volume create <volume>           # Volume erstellen (/var/lib/docker/volumes/...)
docker volume inspect <volume>          # Eigenschaften anschauen
docker volume rm <volume>               # Volume löschen
```

5 Netzwerke verwalten

```
docker network create <network>         # Netzwerk erstellen
docker network rm <network>             # Netzwerk löschen
```

6 Image mit Dockerfile erstellen

Beispieldatei: Dockerfile

```
FROM python
RUN pip install flask
COPY ./static /home/myapp/static/
COPY ./templates /home/myapp/templates/
COPY sample_app.py /home/myapp/
EXPOSE 8080
CMD python /home/myapp/sample_app.py
```

Image im aktuellen Verzeichnis erstellen

```
docker build -t sampleapp .
```

7 Container aus Image erstellen und starten

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

OPTIONS

```
-d                                      # Container im Hintergrund laufen lassen
-e VARIABLE=WERT                        # Umgebungsvariablen setzen
-it                                     # interaktive Shell, beenden mit exit
--name <containername>                  # Name vergeben
--network <network>                     # mit Netzwerk verbinden
--network-alias <alias>                 # Aliasnamen für Container vergeben
-p <hostport>:<containerport>           # Containerport zu Host weiterleiten
--rm                                    # Container nach Beenden löschen
-v </pfad/im/host>:</pfad/im/container> # Verzeichnisse mappen (Bind mount volume)
-v <volume>:</pfad/im/container>        # Volume mappen
```

Beispiele

Hello World

```
docker run hello-world
```

Alpine Linux

```
docker run alpine echo "Hallo!"         # Kommando
docker run -it alpine /bin/sh           # Interaktive Shell, beenden mit exit
```

Webserver

```
docker run -d -p 8080:80 -v /home/gbs/htdocs:/usr/local/apache2/htdocs httpd
docker run -d -p 8080:80 -v /home/gbs/htdocs:/usr/share/nginx/html     nginx
```

MongoDB-Server mit MongoDB-Client

```
docker network create mongonet
docker run -d --network mongonet --name mongoserver mongo
docker run -it --network mongonet --rm mongo mongo --host mongoserver
```

MySQL mit PHPMyAdmin

```
docker network create webnet
docker run -d --network webnet -e MYSQL_ROOT_PASSWORD=123 --name dbserver mysql:8
docker run -d --network webnet -e PMA_HOST=dbserver --name webserver -p 80:80 phpmyadmin/phpmyadmin
```

8 docker-compose

Grundbefehle

```
docker-compose version        # zeigt installierte Version
docker-compose up             # startet Container
docker-compose up -d          # startet Container im Hintergrund
docker-compose down           # Container und Networks entfernen
```

Konfigurationsdatei docker-compose.yml

- Einrückungen: zwei Leerzeichen
- nach Doppelpunkt: ein Leerzeichen

Beispiel

- WordPress with Docker-Compose (c't magazine): gist.github.com/jamct/...