

Docker Befehle

Praxiserprobte Docker- und Compose-Befehle für den Betrieb – durchsuchbar und mit einem Klick kopierbar. Mit Warnhinweisen bei Befehlen, die Daten löschen.

daniele.michelin //
cheat sheet

Images

Lokale Images auflisten

```
docker images
```

Image mit festem Tag laden

```
docker pull nginx:1.27
```

Image aus dem Dockerfile im aktuellen Ordner bauen

```
docker build -t myapp:1.0 .
```

Image für eine Registry umbenennen

```
docker tag myapp:1.0 registry.example.com/myapp:1.0
```

! Image löschen

```
docker rmi myapp:1.0
```

Container starten & verwalten

Laufende Container auflisten

```
docker ps
```

Alle Container inkl. beendeter auflisten

```
docker ps -a
```

Container im Hintergrund mit Port-Mapping starten

```
docker run -d --name web -p 8080:80 nginx
```

Wegwerf-Container mit interaktiver Shell

```
docker run --rm -it alpine sh
```

Container nach Reboot oder Absturz neu starten

```
docker run -d --restart unless-stopped nginx
```

Container geordnet stoppen (SIGTERM, dann SIGKILL)

```
docker stop web
```

! Container erzwungen stoppen und löschen

```
docker rm -f web
```

Logs, exec & Diagnose

Letzte 100 Logzeilen und live weiterverfolgen

```
docker logs -f --tail 100 web
```

Shell im laufenden Container öffnen

```
docker exec -it web sh
```

Einzelnen Befehl im Container ausführen

```
docker exec web nginx -t
```

CPU- und RAM-Verbrauch aller Container (Momentaufnahme)

```
docker stats --no-stream
```

Volumes & Netzwerke

Benanntes Volume in einen Container einhängen

```
docker run -v data:/var/lib/mysql mariadb
```

Volume als Tar-Archiv sichern

```
docker run --rm -v data:/d -v $(pwd):/b alpine tar czf /b/data.tgz -C /d .
```

! Volume löschen – Daten sind weg

```
docker volume rm data
```

Netzwerke auflisten

```
docker network ls
```

Eigenes Bridge-Netzwerk anlegen (DNS per Containername)

```
docker network create appnet
```

Docker Compose

Stack im Hintergrund starten

```
docker compose up -d
```

Images neu bauen und Stack starten

```
docker compose up -d --build
```

Logs eines Dienstes live verfolgen

```
docker compose logs -f web
```

Neue Images laden, geänderte Dienste neu erstellen

```
docker compose pull && docker compose up -d
```

Shell in einem Dienst-Container

```
docker compose exec web sh
```

Stack stoppen und Container samt Netzwerk entfernen

```
docker compose down
```

! Wie down, löscht aber auch die Volumes

```
docker compose down -v
```

Aufräumen mit prune

Speicherbelegung durch Images, Container, Volumes

```
docker system df
```

! Ungetaggte (dangling) Images löschen

```
docker image prune
```

! Alle nicht genutzten Volumes löschen – Datenverlust möglich

```
docker volume prune
```

! Gestoppte Container, ungenutzte Netze und dangling Images löschen

```
docker system prune
```

! Radikal: alles Ungenutzte inkl. Images und Volumes

```
docker system prune -a --volumes
```

Ressourcen-Limits

Arbeitsspeicher auf 512 MB begrenzen

```
docker run -d --memory 512m nginx
```

Auf 1,5 CPU-Kerne begrenzen

```
docker run -d --cpus 1.5 nginx
```

Limits eines laufenden Containers ändern

```
docker update --memory 1g --memory-swap 1g web
```

Log-Rotation für einen Container festlegen

```
docker run -d --log-opt max-size=10m --log-opt max-file=3 nginx
```

OOM-Kills der letzten Stunde finden

```
docker events --since 1h --filter event=oom
```