COMPOSE

```
version: '3.0'
services:
  web:
    image: effectivetrainings/blog
    networks:
    - web
    - data
    deploy:
       placement:
          constraints: [node.role==manager]
       mode: replicated
       labels: [key=value]
       resources:
          limits:
             cpus: '0.5'
             memory: 125M
       update_config:
          parallelism: 1
          delay: 10s
          monitor: 60s
          max_failure_ratio: 0.1
  db:
    image: effectivetrainings/derby-db:blog
    volumes:
      - .:/db-data
networks:
    data:
      driver: bridge
    web:
      driver: bridge
```

docker-compose up	Stack starten
docker-compose logs -f –tail=20	Zeigt die Logs aller Services, angefangen bei den letzten 20 Zeilen
docker-compose pull	Neueste Images aller Services laden
docker-compose down	Stoppt den Stack / Cleanup
docker-compose ps	Zeigt laufende Prozesse mit Compose gestartet

SWARM

docker swarm init –advertise-addr 10.xx	initialisiert den Schwarm
docker swarm join token <manager-token> 10.1..:2377	Swarm beitreten
docker swarm leave	Swarm verlassen
docker node update --availability drain / active <node>	Node abschalten / einschalten
docker node ls	Nodes im Swarm
docker node promote / denote	node zum Manager befördern / zum Worker degradieren
docker service create –name <name> -p <port-mapping> <image>	service starten
docker service ls	Services im Swarm anzeigen
docker service ps <service-name>	service tasks anzeigen
docker service scale <name>=5	Service Update - 5 Replicas
docker network create –driver overlay <name>	Overlay Netzwerk erstellen

EFFECTIVE TRAININGS & CONSULTING
Dietersheimerstraße 23, 80805 München
0176 / 63134391
http://www.effectivetrainings.de

EFFECTIVE DOCKER

docker run -p 8080:8080 -v /data:/data <image>	Starte Container. Port 8080. Mount Host /data - Container /data.
docker ps	Laufende Container und Tasks
docker images	Zeigt die aktuell lokal verfügbaren Images an
docker exec -ti <container> bash	Neuen Bash-Prozess erzeugen im Container
docker volume create <name>	Named Volume erzeugen
docker system df	Zeige den von Docker verwendeten Speicher
docker system prune	Lösche alles überflüssige (cleanup)
docker system events **docker system events -f {{json .}}**	Docker Events / Docker Events als JSON
docker inspect <container>	Container Details ausgeben
docker logs <container>	Container Logs
docker logs –tail=20 -f <container>	Container Logs folgen, angefangen bei den letzetn 20 Zeilen
docker rm -v -f <container>	Container mit Volumes entfernen

YOURS...

EFFECTIVE DOCKER TRAINING
http://effective-docker.effectivetrainings.de/

DOCKERFILE

```
FROM openjdk:jre-alpine
MAINTAINER Martin <martin@effectivetrainings.de>

ADD file /file
# Datei container hinzufügen
ENV foo=bar
# Environment Variable definieren
EXPOSE 8080
# Port freigeben / deklarieren
LABEL <key>=<value>
# Label definieren
ARG user
USER $user
#Argumente für den Build
ENTRYPOINT ["/entrypoint.sh", "arg1", "arg2"]
# Entry Point definieren
VOLUME /data
# Volume deklarieren
USER <username>
# User festlegen
ONBUILD ADD . /files
# Instruktion die beim Build durchgeführt wird
HEALTHCHECK --interval=5m CMD curl -f http://localhost:8080/
health/ || exit 1
# Health Check definieren
CMD    ["-param1", "-param2", "-param3"]
# Parameter
```

BEST PRACTICES

- [x] EIN PROZESS PRO CONTAINER
- [x] MINIMIERE LAYERS
- [x] NICHT ALS ROOT BETREIBEN
- [x] MINIMIERE BUILD CONTEXT
- [x] KEEP IT SIMPLE

DOCKER WORKFLOW

BUILD ONCE, RUN ANYWHERE...

docker pull <image>:<tag>	Image laden
docker build -t <tag> .	docker Image bauen
docker push <registry>/<repo>	Image pushen

NÜTZLICHE KOMMANDOS

docker cp <container>:<src path> <target-path>	Dateien aus Container kopieren
docker port <container>	Port Mappings
docker info --format "{{json .}}"	Docker Engine Info

DOCKER UI

Portainer	http://www.portainer.io
Swarm Visualizer	https://github.com/ManoMarks/docker-swarm-visualizer
Kitematic	https://kitematic.com/

EFFECTIVE TRAININGS & CONSULTING - BLOG
http://effective-docker.github.io