DOCKER

Containers

`docker ps` : listing containers

- `-a` : list all, running and stopped

`docker run <image>` : run an image

- `-i` : interactive mode
- `-t` : expose TTY
- `-p` : expose port `ext:int` eg. `3000:80`
- `-d` : detached mode
- `-v <abs-lpath>:<cpath>` : create bind mount
- `-v <abs-lpath>:<cpath>:ro` : create read-only bind mount
- `-v <name>:<cpath>` : create named volume
- `-v <cpath>` : create anonymous volume
- `-rm` : automatically remove when exits
- `--env (-e) KEY=VALUE` : overwriting default env vars
- `--env-file <env-file-path>` : overwriting default env vars with vars from .env file
- `-name <string>` : give container a name
- `--network=host` : Use host networking. Then `-p, -publish, -P`, and `-publish-all` are ignored, since container does not have its own IP address
- `--add-host=host.docker.internal:host-gateway` : Mapp DNS host.docker.internal to host-gateway

`docker rm <container>` : remove container

`docker stop <container>` : stop running container

`docker start <container>` : start stopped container

`docker attach <container>` : attach to the running container

Images

`docker build <src>` : build docker image; `src`(.) - usually the `Dockerfile` is in the root

- `-t <name>:<tag>` : tag an image
- `--build-arg KEY=VALUE` : Overwriting default values of the args

`docker images` : list all images

`docker rmi <image-id>` : remove image

`docker image inspect <image-id>` : details of the image

`docker image prune` : remove image

- `-a` : remove all images

DockerHub & Sharing

`docker push <image-name>` : Push image to DockerHub

`docker pull <image-name>` : Pull image from DockerHub

`docker login` : Login into DockerHub

`docker logout` : Logout from DockerHub

Volumes

`docker volume ls` : List volumes

`docker volume rm` : Remove volume

Other

`docker logs <container>` : fetch the logs of the container

- `-f` : follow

`docker cp <src> <dest>` : copy files to / from the container; `src/dest` (`<container-name>:<path> | <path>`)

Dockerfile

`FROM <base-image>` : eg. `node`

`WORKDIR /app` : Commands will be executed relatively to this dir

`COPY <src> <src>` : `src` (`.`) `dest` (`/app`)

`RUN npm install` : runs when image is created

`ARG NAME=DEFAULT_VAL` : Defining build argument

`ENV PORT 80` : Creates env variable PORT with a default value 80

`EXPOSE $PORT` : exposes port under the PORT variable

`VOLUME ["<path>", ...]` : create anonymous volume

`CMD ["node", "server.js"]` : runs when container is started. Should always be the last

Networking

`host.docker.internal` : Resolves to you localhost IP as seen from container. In order to use that, we must add new host in build command: `--add-host=host.docker.internal:host-gateway`

`docker network create` : Create new docker network

`docker network ls` : List all networks

Volumes & Bind mounts

Anonymous volumes

- Created specifically for a single container
- Survives container shutdown / restart, but <u>NOT removal</u>
- <u>Cannot be shared</u> across containers

Anonymous volumes

- Created in general / not container specific
- Survives container shutdown / restart & removal
- <u>Can be shared</u> across containers
- Can be reused

Bind mounts

- Located at host file system, not tied to any container
- Survives container shutdown / restart & removal
- <u>Can be shared</u> across containers
- Can be reused

DOCKER COMPOSER

Compose

```
services:
  mongodb:
    image:  'mongo'
    container_name:  mongodb
    volumes:
      - data:/data/db
    #1 environment:
      MONGO_INITDB_ROOT_USERNAME: max
    #2 env_file:
      - ./env/mongo.env
    networks:
      - goals-net
  backend:
    #1 build:  ./backend
    #2 build:
      context:  ./backend
      dockerfile:  Dockerfile-name
    ports:
      - '80:80'
volumes:
  data:
```

Compose commands

`docker compose up` : Compose up containers

  `-d` : detached mode

`docker compose down` : Stop containers

  `-v` : remove volumes