**How to install docker**

sudo apt update -> update libraries.

sudo apt install docker.io -> download and install docker.

sudo usermod -aG docker $USER -> command for not need to use "sudo" needs restart of server. If not used need to use "sudo"

docker --version -> check the installed version.

docker container run --rm hello-world -> simple test container if docker works.

**docker container run**-it -> interactive terminal.  
-d -> run in background.  
**-**v {volume name}:/{container name} /{path in image} -> links volume to the image.  
**-**v {host path}:/{container name} /{path in image} -> bind host path to the image.  
-m -> limits memory used by docker container.  
--rm {container name} -> remove after stop.  
--name {container name} -> name container.  
--restart always -> restart always when stopped.  
--restart on-failure:INTEGER -> restart on failure INTEGER times.  
--restart unless-stopped -> restart always unless stopped by user.  
--env VARIABLE=VALUE -> set variable.  
--env-file {.env file name} -> set variables from file.  
--publish {host port}:{container port} -> publish on specifed ports.

docker container ls -> list of running containers.  
docker container ls -all -> list of not running containers.

docker container create {container name} -> create container.  
docker container stop {container name} -> stop container.  
docker container start {container name} -> start container.  
docker container attach {container name} -> attach to container.  
docker container prune -> remove all non running containers.  
docker container exec -it {container name} command -> execute command in container.  
docker container logs {container name} -> show logs of container.  
docker container logs --follow {container name} -> show live logs of container.  
  
**docker container inspect {container name} | jq --raw-output .[0].NetworkSettings.Networks.bridge.**

docker container inspect {container name} | jq --raw-output .[0] -> you can always start like that to see whole output, after that you can specifty what you want. As .NetworkSettings and .Networks. and even further .bridge.

**.[0] - needs to be always there**

**- the first dot represent the object being proccesed so docker inspect  
 - the [0] represents the first element of array, since docker inspect return array**

docker container stats -> show container stats  
docker volume ls -> show all volumes.  
docker volume prune -> delete all unused volumes.  
docker volume create {volume name} -> create volume.  
docker volume rm {volume name} -> delete volume.

docker network create -> create network  
docker network disconnect {network name} {container name} -> disconnect container from network.  
docker network connect {network name} {container name} -> connect container to network.  
docker network rm -> remove network.  
docker network prune -> remove unused networks.

docker login -u {user} -> log in your docker hub account.  
docker image tag {image name} marvy936/{image name}:{ tag} -> tag your image.  
docker image push marvy936/{image name} --all-tags -> push all image tags to hub.docker.com.  
docker image push marvy936/{image name} -> push image to hub.docker.com.  
docker image pull marvy936/{image name} -> pull image from hub.docker.com.

**Docker Image**

docker image ls -> show all images.  
docker image pull {image name} -> pull image.  
docker image rm {image name} -> remove image.  
docker image inspect {image name} -> inspect image.  
docker image build –tag {image name} . -> build image from current location, need Dockerfile

**Dockerfile**

**FROM** - Určuje základný obraz, na ktorom budeš stavať svoj obraz.

**LABEL** - Pridáva metadata k obrazu, ako sú autor, verzia a popis.

**RUN** - Vykonáva príkazy na inštaláciu balíkov alebo konfiguráciu prostredia pri vytváraní obrazu.

**COPY** - Kopíruje súbory alebo adresáre z hostiteľského systému do obrazu.

**ADD** - Podobné príkazu `COPY`, ale navyše podporuje rozbaľovanie tar súborov a stiahnutie súborov z URL.

**CMD** - Definuje predvolený príkaz, ktorý sa má vykonať pri spustení kontajnera.

**ENTRYPOINT** - Definuje príkaz, ktorý sa má vykonať vždy, keď sa kontajner spustí. Môže byť kombinovaný s `CMD` na poskytnutie argumentov.

**ENV** - Nastavuje premenné prostredia, ktoré sú dostupné v kontajneri.

**EXPOSE** - Označuje porty, na ktorých aplikácia počúva. Neotvorí ich automaticky, ale dokumentuje, ktoré porty by sa mali mapovať.

**VOLUME** - Vytvára bod pre pripojenie (volume), kde môžu byť trvalé dáta uchovávané mimo

**USER** - Nastavuje používateľa, pod ktorým sa budú príkazy vykonávať.

**WORKDIR** - Nastavuje pracovný adresár pre príkazy `RUN`, `CMD`, `ENTRYPOINT`, `COPY` a `ADD`.

**Example:**

# Stage 1: Build the Python application

FROM python:3.9-slim AS builder

# Set environment variables

ENV PYTHONDONTWRITEBYTECODE=1

ENV PYTHONUNBUFFERED=1

# Create a working directory

WORKDIR /app

# Install build dependencies

RUN apt-get update && \

apt-get install -y --no-install-recommends gcc libpq-dev

# Install Python dependencies

COPY requirements.txt .

RUN pip install --no-cache-dir --upgrade pip && \

**pip install --no-cache-dir -r requirements.txt**

# Copy the application source code

COPY . .

# Stage 2: Create the final runtime image

FROM python:3.9-slim

# Set environment variables for production

ENV PYTHONDONTWRITEBYTECODE=1

ENV PYTHONUNBUFFERED=1

# Create a user to run the application

RUN adduser --disabled-password appuser

# Set working directory

WORKDIR /app

# Copy only necessary files from the builder stage

COPY --from=builder /usr/local/lib/python3.9/site-packages /usr/local/lib/python3.9/site-packages

COPY --from=builder /usr/local/bin /usr/local/bin

COPY --from=builder /app /app

# Change ownership to the appuser

RUN chown -R appuser:appuser /app

# Switch to the appuser

USER appuser

# Expose the application port

EXPOSE 8000

# Run the application

CMD ["python", "app.py"].dockerignore -> create this file within Dockerfile directory, it ignores files for command COPY . .

**vi requriements.txt**

Flask==2.0.3  
gunicorn==20.1.0  
requests==2.26.0  
psycopg2==2.9.3

**Docker Context**

docker context ls -> shows all context, used is marked with \*.

docker context inspect {context name} -> shows information about context.

docker context create -> create new context.  
 --description "{description}" -> description of context.  
 --docker "host=ssh://{remote user}@{remote host} " {context name} -> docker endpoint  
  
docker context use {context name} -> switch to context.

ssh-copy-id -i ~/.ssh/id\_rsa [{user}@{remote host}](mailto:ubuntu@18.185.48.5)  -> copy key to remote host

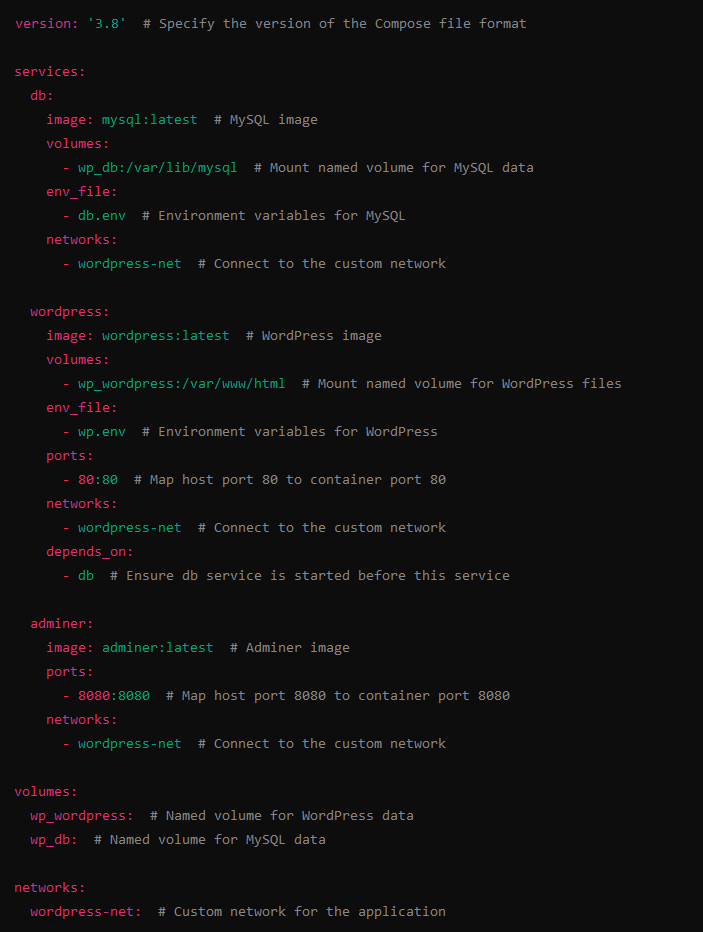
docker context rm {context name} -> remove context.

**Docker Compose**

docker compose up -d

docker compose down

docker compose ls

**docker-compose.yml -> create in custom dir specifed for compose like „wordpress“**

**depends on:  
 wp\_db:** # wait for wp\_db to proceed

.env -> you can create this file for variables, this file is automaticaly imported to compose

file.env -> you can specify different variables in separate files and load them with **env\_file:**

**PORTAINER**