Docker TP to learn some commands

1 $ docker info, docker images or docker image ls, docker container, docker (container, images, volumes, etc) prune, docker rmi <image_name>.

2 $ docker help

3 $ docker search <image_name> : adminer => for this TP

4 $ docker pull Debian

Créer dans un fichier Dockerfile

FROM busybox:latest

CMD[“data”]

The docker build command builds an image from a Dockerfile. To build the image from our above Dockerfile, use this command:

docker build -t dockerfile .
docker run -it --name exemple(nom en sortie) dockerfile

Now, using the docker commit command, we can create a Docker image from the container.
docker ps -a

docker commit exemple docker_file2:latest

Publish an Image to Docker Hub

To be able to publish our Docker images to Docker Hub, there are some steps that we need to follow:

Log into the Docker public registry from your local machine terminal using Docker CLI:

1 $ docker login

Tag the image.

This is a crucial step that is required before we can upload our image to the repository. As we discussed earlier, Docker follows the naming convention to identify unofficial images. What we are creating is an unofficial image. Hence, it should follow that syntax. According to that naming convention, the unofficial image name should be named as
follows: <username>/<image_name>:<tag_name>. In my case, I need to rename it
as loicsanou/dockerfile:latest 2

$ docker tag dockerfile:latest loicsanou/dockerfile:latest

Publish the image

3 $ docker push loicsanou/dockerfile:latest

If you want to test out your image, use the below command and launch a container from it:

1 $ docker pull loicsanou/dockerfile:latest

2 $ docker run -it loicsanou/dockerfile:latest

Our first Container

For this part we will work with nginx image.

1 $ docker pull nginx

2 $ docker run nginx => by default docker will pull if the image is missing.

3 $ docker inspect <container_name> => to access the metadata of our container.

After pulling nginx the distribution can be ecstatic_kowalevski or silly_herman in this case you should type docker inspect ecstatic_kowalevski

4 $ curl ip address seen when typed docker inspect ecstatic_kowalevski

5 $ docker stop ecstatic_kowalevski => to stop the container.

6 $ docker ps -a => to find all docker containers.

7 $ docker start ecstatic_kowalevski => to start this single container.

Our first Bash

$ docker run -d –name c1 -p 8080:80 nginx => to run in detached mode on port 8080 (test on the browser 😊).

$ docker exec -ti c1 bash => to run in interactive mode on the bash (shell).

$ vim => the command will not be recognized.

$ apt update

$ apt install -y vim

$ vim /usr/share/nginx/html/index.html

$ docker stop <c1_name or c1_id>

$ docker rm -f c1

Docker Volume

$ docker run -d --name c1 -v /srv/nginx/:/usr/share/nginx/html/ -p 8080:80 nginx => Run container on a volume

$ nano /srv/nginx.index.html => replace nano by vim if y're using linux system.

$ docker volume ls

$ docker volume create myvolume

$ docker volume inspect myvolume

$ docker run -d --name c1 -v myvolume:/usr/share/nginx/html/ -p 8080:80 nginx

$ curl 127.0.0.1:8080

$ dir (or ls on linux) /var/lib/docker/volumes/myvolume/_data

$ docker volume rm myvolume

$ docker volume COMMAND

$ docker rm -f c1

$ docker volume rm -f myvolume

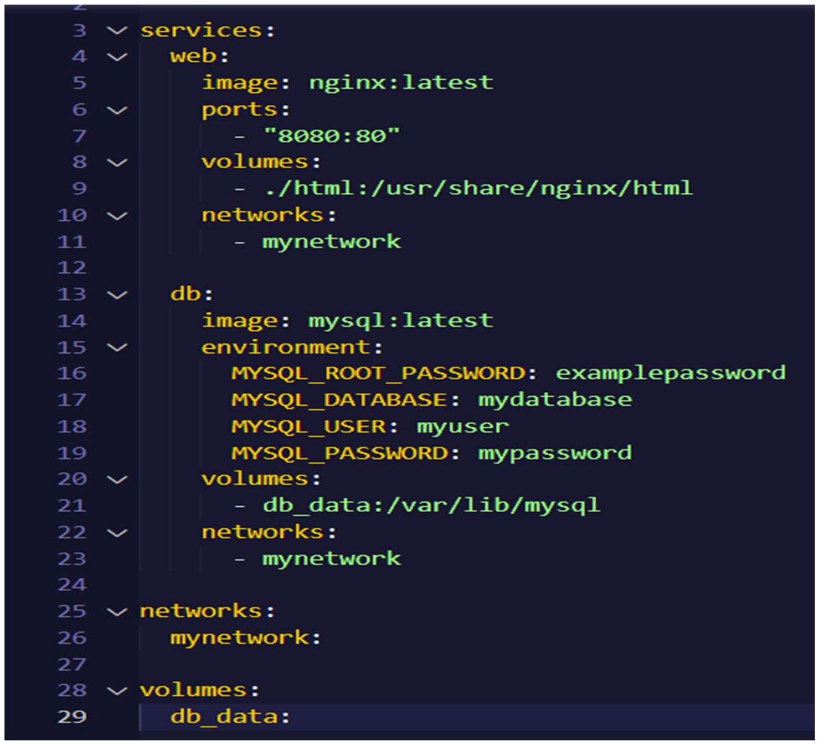
Environment Variables

$ docker run -d –name c1 -e MYVAR="Loic" -e MYENV="Production" ubuntu:latest sleep 3600 (you can use –hostname myhostname to set the hostname).

$ docker exec -ti c1 bash

$ env

Docker compose



```yaml
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/share/nginx/html
    networks:
      - mynetwork

  db:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: examplepassword
      MYSQL_DATABASE: mydatabase
      MYSQL_USER: myuser
      MYSQL_PASSWORD: mypassword
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - mynetwork

networks:
  mynetwork:

volumes:
  db_data:
```

This **docker-compose.yml** file defines two services: **web** using the Nginx image and **db** using the MySQL image. These services are connected to the same network named **mynetwork**. Additionally, the **db** service uses a volume named **db_data** to store the database data.

To run this **docker-compose.yml** file, ensure that Docker and Docker Compose are installed on your system. Then, execute the following command in the directory where your **docker-compose.yml** file is located:

$ docker compose up -d (to run in background).

The TP is just the basics, so it can't be perfect, to master docker follow this playlist, we did not abord Docker Network so we encourage you to do your proper research on this topics 😊. https://www.youtube.com/playlist?list=PLy7NrYWoggjzfAHlUusx2wuDwfCrmJYcs and read the official documentation : https://docs.docker.com/get-started/

Thanks.

Loic Sanou.