TP4 - Docker

Charger les fichiers partagé des images docker:
docker load -i nginx.tar
docker load -i alpine.tar
docker load -i python.tar
docker load -i busybox.tar

Vérifier les images:
docker images

Exercise 1:

- Lancer un premier conteneur

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

- Tester dans le navigateur sur http://localhost:8080 :
- Vérifier que le conteneur s'exécute

docker ps

- Voir les logs

docker logs web

- Arrêter et supprimer

docker stop web
docker rm web

Exercise 2:

- Entrer dans un conteneur

docker run -it busybox sh

- Inspecter un conteneur

docker inspect busybox

- Créer un volume:

docker volume create data-tp

Creer une application web flask app.py :

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Bonjour, ceci est une application Docker TP !"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Un ficheir Dockerfile simple:

```
FROM python:3.11-slim
WORKDIR /app
COPY app.py .
RUN pip install flask
EXPOSE 5000
CMD ["python", "app.py"]
```

Si vous n'avez pas de connexion stable mettez le dossier partagé packages dans le même dossier du projet et remplacer pip install flask par:
`RUN pip install --no-index --find-links=packages flask`

- Construire l'image

`docker build -t hello-tp .`

- Vérifier qu'elle existe

`docker images`

- Lancer l'application

`docker run -d -p 5000:5000 --name hello hello-tp`

- Tester sur http://localhost:5000
- Arrêter

`docker stop hello`

- Supprimer le conteneur et l'image

`docker rm hello`
`docker rmi hello-tp`

Exercice 3:

- Créer un Dcoekrfile et mettez le projet sur git out utiliser l'ancien projet git avec dcoekr