Introduction à Docker

1. QUESTIONS

Qu’est ce que « docker » ?

Docker est un outils pour faire des conteneurs, il permet le déploiement des applications, on peut le composé en trois parties principale :

- docker-cli: ligne de commande
- dockerserver
- dockerhub : service en ligne de docker

Qu’est ce qu’une image ?

Une image est une représentation des instructions présentent dans le un fichier nommé « Dockerfile ».
C'est grâce aux images qu'on peut créer des conteneur.

Il existe trois manière pour lancer un conteneur:

1) Récupérer une image depuis un serveur en ligne: docker run nom_image
2) A partir d'une image : docker build -t nom_image .
3) Récupérer l'id du conteneur avec docker exec id_conteneur

Qu’est ce qu’un « Dockerfile » ?

Un fichier contenant l’ensemble des instructions dont en a besoin pour lancer notre application.

Comment le structurer ?
Il faut se poser les question : Quoi? Pourquoi? Comment?

Quoi ? from image
Pourquoi? se renseigner sur les besoins
Comment? avec les instructions =>

FROM, RUN, COPY src dest (recupere les fichiers/dossier depuis notre machine local vers le conteneur) , WORKDIR (permet de nous déplacer dans le dossier du conteneur), VOLUME (permet la persistance des données) Expose -p(avoir accès à un port depuis la machine local), NETWORK name(relier/communiquer 2 conteneur), ENV (definir une variable d'environnement)

lancer un conteneur :
-it : interagir avec un conteneur lancé
- d: lance un conteneur en arriere plan
- p: portLocale:portConteneur
-e nomdelavariable=value

A quoi sert la directive FROM dans un docker file ?

L'instruction FROM sert à spécifier l'image de base qu'on va utiliser

Quel est la différence entre une machine virtuel et un conteneur ?

- Une machine virtuel recrée une OS complète et utilise plus de ressources .
- Un conteneur utilise uniquement les ressource nécessaire pour que ça fonctionne.

Quel est la différence entre docker et image:

Les images sont créées à partir de fichiers de configuration, nommés "Dockerfile".
Un conteneur est l'exécution d'une image : il possède la copie du système de fichiers de l'image, ainsi que la capacité de lancer des processus.

A quoi sert le mot clé CMD dans un Dockerfile?

CMD est utilisé pour définir la commande de démarrage par défaut du conteneur

Quel est la différence entre CMD et Entrypoint ?

Ces deux commandes sont très similaires ; cependant, leur différence est que la commande CMD peut être remplacée lors de l'exécution alors que la commande ENTRYPOINT ne le peut pas.

Quelle est la différence entre un réseau bridge et un réseau «host» ?

Le conteneur est connecté par défaut au réseau bridge, ce réseau permet à plusieurs conteneurs de pouvoir communiquer entre eux mais aussi peuvent communiquer avec l'extérieur via l'interface de réseau hôte, le réseau bridge est lent car le trafic doit passer par la couche de virtualisation de Docker .

En revanche, un réseau "host" permet au conteneur d'utiliser directement l'interface réseau de l'hôte et il est donc rapide. Le conteneur partage alors la même adresse IP et les mêmes interfaces réseau que l'hôte.

Quel est la différence entre un réseau bridge par défaut et un réseau «user-defined bridge» ?

Un réseau bridge est créer lors du lancement du conteneur, les conteneurs sont isolés des autres réseaux du système hote et Utilise des adresses IP privées du réseau 172.17.0.0/16.

Un réseau user-defined bridge est créer par l'utilisateur, il permet une meilleur isolation et un contrôle sur les adresses IP et ports utilisés par les conteneurs.

2. EXERCICES

Mon premier docker file

0.1) sudo service docker start
0.2) docker build -t nom_image . => crée une image à partir du dockerfile
1) docker run -d nom_image
2) docker run -d nom_image ls -la
3) docker run nom_image bash
4) docker run -it nom_image bash - exit pour quitter
5) docker images : liste les images
6) docker ps -a: liste les conteneurs

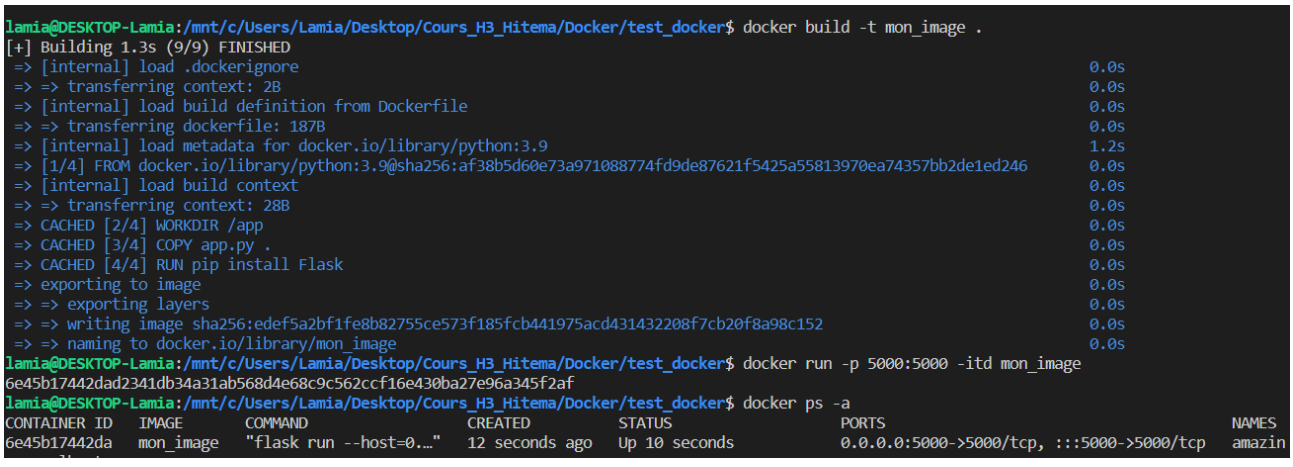
7) docker rmi -f <id-image> => supprime une image
8) docker rmi $(docker images -a -q) => supprime toutes les images
9) docker stop $(docker ps -a -q) => stop tout les containeurs

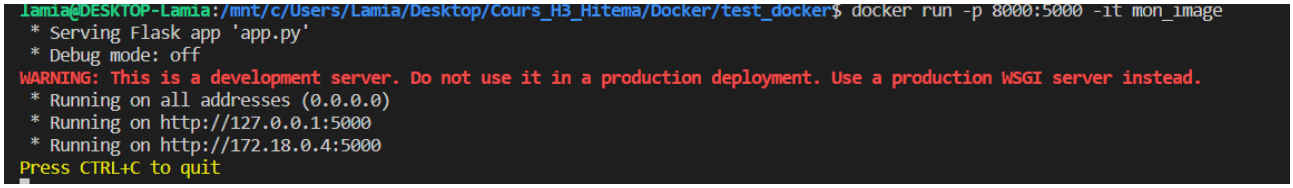
Interagir avec le serveur mysql du conteneur - utilisation du volume:

1) docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=password -d mysql:5.7
2) docker exec -it 87343780f0b4 bash
3) mysql -p
4) create DATABASE test;
5) use test ;
6) CREATE TABLE test (id INT(10), name VARCHAR(120)) ;
7) insert into test value(1, 'lamia');
8) docker stop id_conteneur
la database n'existe plus…

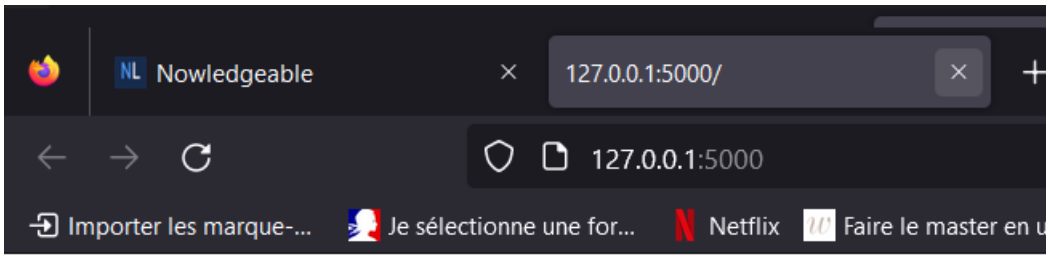
9) docker run -v /var/lib/mysql --name mysql-container2 -e MYSQL_ROOT_PASSWORD=password -d mysql:5.7
10) memes instruction que 3) 4) 5) 6) 7) ensuite docker stop id_container
11) On relance un container identique, on s'y connecte, la database existe encore !

Comment lancer un petit serveur web python/flask dans un conteneur





vérification d’accès à la page d’accueil :



Hello World!

Affichage des logs en mode deamon :

1) docker logs container-id

```
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/test_docker$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                       NAMES
6e45b17442da   mon_image   "flask run --host=0.…"   13 minutes ago   Up 13 minutes   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   amazing_engelbart
f50cc94337b3   mysql:5.7   "docker-entrypoint.s…"   30 minutes ago   Up 30 minutes   3306/tcp, 33060/tcp                         mysql-container2
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/test_docker$ docker logs 6e45b17442da
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.18.0.3:5000
Press CTRL+C to quit
172.18.0.1 - - [29/Mar/2023 12:22:37] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [29/Mar/2023 12:31:15] "GET /page-inconnue HTTP/1.1" 404 -
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/test_docker$
```

Publication de port et base de données

1)docker run --name mysql -e MYSQL_ROOT_PASSWORD=password -d mysql
2)mysql -u root -p -h localhost
On a un message d’erreur car le port n’est pas ouvert…
3) docker stop mysql
4) docker rm mysql
5)On relance le conteneur en ouvrant le port 3306 :
docker run --name mysql -e MYSQL_ROOT_PASSWORD=password -p 3306:3306 -d mysql
6)On peut désormais se connecter à la BDD avec le client mysql :

mysl -u root -p -h localhost -P 3306 –protocol tcp

```
lamia@DESKTOP-Lamia:~$ mysql -u root -p -h localhost -P 3306 --protocol tcp
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database test
    -> ;
Query OK, 1 row affected (0,02 sec)
```

Docker et les réseaux

```
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/exo_mysql_conteneurs$ docker run --name mysql-test-cont
ainers -e MYSQL_ROOT_PASSWORD=password -d mysql
ec07de4a1ba6869b1989cbd49e347ab4d68482a758ddc2be7ba708cfd9c5b552
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/exo_mysql_conteneurs$ docker build -t image_mysql .
[+] Building 28.9s (6/6) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 110B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                                        1.6s
 => CACHED [1/2] FROM docker.io/library/ubuntu:latest@sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea  0.0s
 => [2/2] RUN apt-get update && apt-get install -y mysql-client                                        26.5s
 => exporting to image                                                                                  0.7s
 => => exporting layers                                                                                 0.7s
 => => writing image sha256:f6deb8fd7cd315a7c660b11d7218df548162fc302b76f88fc6d205035a05d81d            0.0s
 => => naming to docker.io/library/image_mysql                                                          0.0s
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/exo_mysql_conteneurs$ docker run --name my-ubuntu-conta
iner -it image_mysql bash
root@a504cd310bbc:/# mysql -h mysql-test-container -u root -p
Enter password:
ERROR 2005 (HY000): Unknown MySQL server host 'mysql-test-container' (-2)
root@a504cd310bbc:/#
```

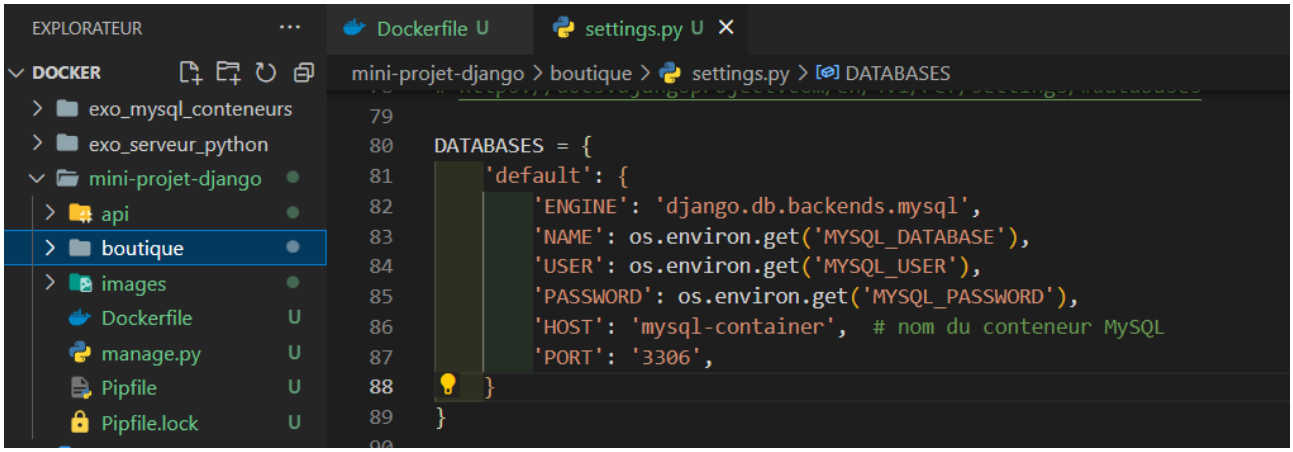
Ne fonctionne pas puisque les deux conteneurs ne sont pas sur le même réseau :

1) docker stop mysql-test-containers my-ubuntu-container
2) docker rm mysql-test-containers my-ubuntu-container
3) Création du réseau : docker network create exonet
4) On recré un conteneur en utilisant le nouveau réseau exonet :

```
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/exo_mysql_conteneurs$ docker network create exonet
dab184313dfdbbc5bb6b57bff7c3008eeca5e93f71c79c48e2d232b66bd2f80e
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/exo_mysql_conteneurs$ docker network ls
NETWORK ID     NAME                                DRIVER    SCOPE
82c1a9413646   abymap-system_abymap                bridge    local
70a10042fbdc   bridge                              bridge    local
dab184313dfd   exonet                              bridge    local
9d01f758b33e   host                                host      local
4167e0cc0c81   none                                null      local
73b00cc12b10   projet_maquette_artiste_default     bridge    local
6d8d72354a8c   projetdockerredis_monreseau         bridge    local
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/exo_mysql_conteneurs$ docker run --name mysql-test-containers
--network exonet -e MYSQL_ROOT_PASSWORD=password -d mysql
314f1e2433d4a9fcfa225e25bd205c3c0b6252a5e83185e4cb95623dfec20697
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/exo_mysql_conteneurs$ docker run --name my-ubuntu-container --
network exonet -it image_mysql bash
root@0deda0d19232:/# mysql -h mysql-test-containers -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Mini projet avec django rest framework

1) Création de l’api et du dockerfile
2) docker build -t image_django .
3) docker run -d -p 5000:8000 image_django
4)Créer un container pour la base de donnée mysql :
docker run --name mysql-container -e MYSQL_USER=myuser -e MYSQL_PASSWORD=mypassword -e MYSQL_DATABASE=mydb -d mysql:latest
5)Pour connecter l’application Django à la base de données Mysql :



6) Pour connecter les deux containers ensemble :

```
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/mini-projet-django$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS                                       NAMES
eab2c6219a83   mysql:latest   "docker-entrypoint.s…"   35 seconds ago   Exited (1) 34 seconds ago                                               mysql-container
f6418c654860   image_django   "python3 manage.py r…"   8 minutes ago    Up 8 minutes                0.0.0.0:5000->8000/tcp, :::5000->8000/tcp   pedantic_agnesi
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker/mini-projet-django$ docker network create mynetwork
26a4c4ebb89c65fbc9a141feeb42b7d8c95ff6cba3d57bf8ae5126fc2028f0ab
```

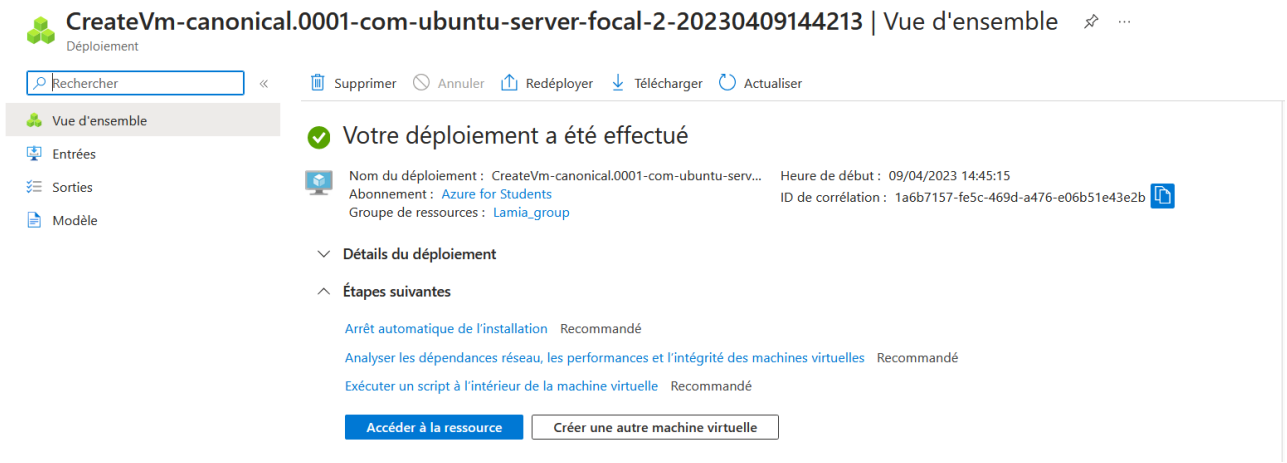
docker network connect mynetwork pedantic_agnesi
docker network connect mynetwork mysql-container

Mini projet web et docker compose

```
lamia@DESKTOP-Lamia:/mnt/c/Users/Lamia/Desktop/Cours_H3_Hitema/Docker$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED              STATUS
       PORTS                                      NAMES
2f5c26a95471   mini-projet-docker-compose_web   "python manage.py ru…"   About a minute ago   Up About a minute
       0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   mini-projet-docker-compose_web_1
4250b40b284a   mysql:latest                   "docker-entrypoint.s…"   About an hour ago    Restarting (1) 9 second
s ago                                          mini-projet-docker-compose_mysql_1
```

3. PROJET

1) Création de la machine virtuel :



2) Connexion à la machine virtuel

```
lamia@DESKTOP-Lamia:~$ ssh -i /home/lamia/Docker/Lamia_key.pem azureuser@51.11.246.140
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1035-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Apr  9 13:27:42 UTC 2023

  System load:  0.11              Processes:             125
  Usage of /:   5.3% of 28.89GB   Users logged in:       0
  Memory usage: 4%                IPv4 address for eth0: 10.0.0.4
  Swap usage:   0%
```

3) Clonage du projet :

chemin : /home/azureuser/projet_docker/Docker/projet

```
azureuser@Lamia:~$ cd projet_docker
azureuser@Lamia:~/projet_docker$ git clone https://github.com/LamaElkhouk/Docker.git
Cloning into 'Docker'...
remote: Enumerating objects: 412, done.
remote: Counting objects: 100% (412/412), done.
remote: Compressing objects: 100% (130/130), done.
remote: Total 412 (delta 278), reused 405 (delta 271), pack-reused 0
Receiving objects: 100% (412/412), 2.52 MiB | 6.10 MiB/s, done.
Resolving deltas: 100% (278/278), done.
azureuser@Lamia:~/projet_docker$ ls
Docker
azureuser@Lamia:~/projet_docker$ cd Docker/
azureuser@Lamia:~/projet_docker/Docker$ ls
Exercices_Docker.odt  exo_mysql_conteneurs  exo_serveur_python  mini-projet-django  mini-projet-docker-compose  projet
azureuser@Lamia:~/projet_docker/Docker$ cd projet
azureuser@Lamia:~/projet_docker/Docker/projet$ ls
Dockerfile  Pipfile  Pipfile.lock  api  boutique  docker-compose.yml  docker-entrypoint.sh  images  manage.py
azureuser@Lamia:~/projet_docker/Docker/projet$
```

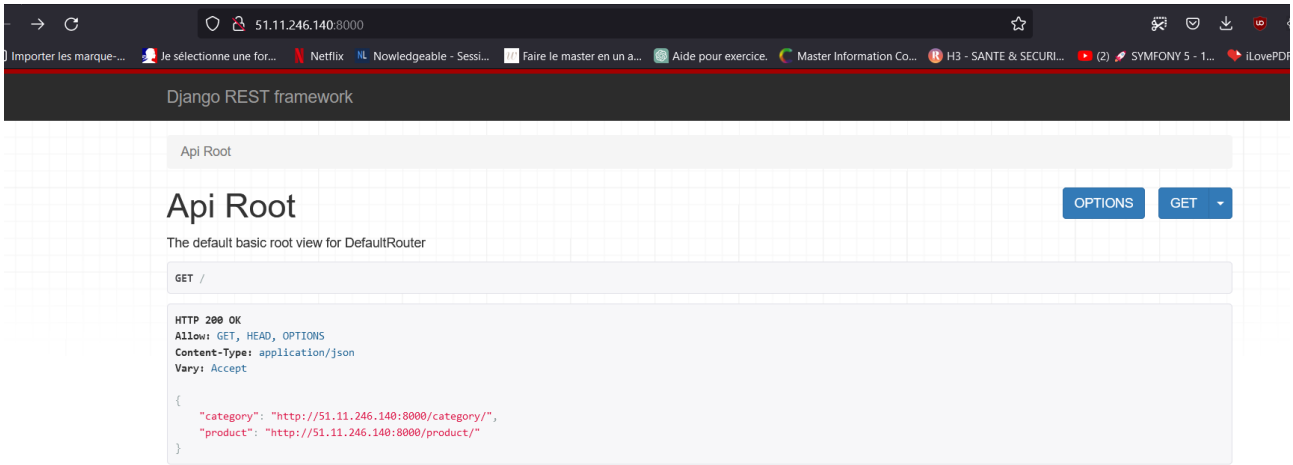
3) installation de docker sur la machine et création de l'image a partir du dockerfile :

=> docker build -t nom_image .

=> nohup docker-compose up &

```
azureuser@Lamia:~/projet_docker/Docker/projet$ docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS                                                  NAM
S
08fa027d9956   projet_web    "/app/docker-entrypo…"   25 minutes ago   Up 3 minutes    0.0.0.0:8000->8000/tcp, :::8000->8000/tcp              pro
et_web_1
3548d526bc9a   mysql:8       "docker-entrypoint.s…"   25 minutes ago   Up 3 minutes    0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   pro
et_mon_app_web_1
```

=> aller sur http://51.11.246.140:8000/



4) Test unitaire simple sur le projet avec cypress

=> npm init -y

=> npm install cypress –save-dev

=> ajout du test dans Cypress/integration

=> docker login

=> creation de l'image a partir du dockerfile :

docker build -t lamia/cypress-docker-tutorial:latest -f DockerfileCypress .

=> docker pull lamia/cypress-docker-tutorial:latest