PROJET SUR DOCKER

Dans ce projet nous allons utiliser centos7 puis crée une machine virtuel qui portera le nom de docker et utiliser visual studio pour les codes.

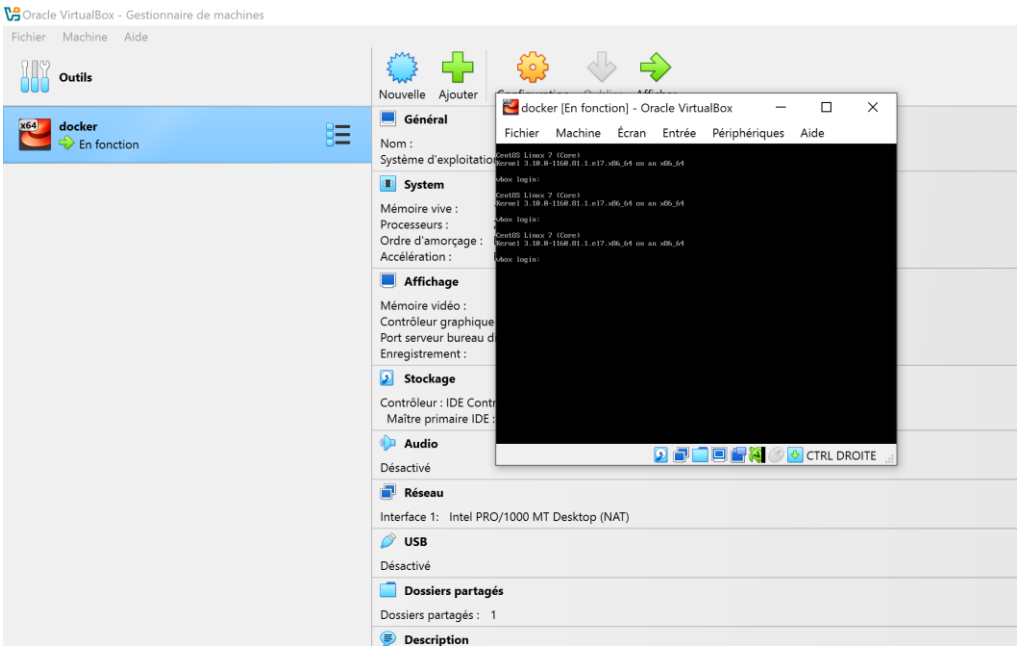
```
Vagrantfile
# -*- mode: ruby -*-
# vi: set ft=ruby :
# To enable zsh, please set ENABLE_ZSH env var to "true" before launching vagrant up
#   + On windows => $env:ENABLE_ZSH="true"
#   + On Linux  => export ENABLE_ZSH="true"

Vagrant.configure("2") do |config|
  config.vm.define "docker" do |docker|
    docker.vm.box = "eazytrainingfr/centos7"
    docker.vm.box_version = "1.0"
    docker.vm.network "private_network", type: "dhcp"
    docker.vm.hostname = "docker"
    docker.vm.provider "virtualbox" do |v|
      v.name = "docker"
      v.memory = 1024
      v.cpus = 2
    end
    docker.vm.provision :shell do |shell|
      shell.path = "install_docker.sh"
      shell.env = { 'ENABLE_ZSH' => ENV['ENABLE_ZSH'] }
    end
  end
end
```

Voici notre code vagrant qui contient toutes les information du machine docker a virtualiser sur virtuel box.

```
PS C:\Users\monji\Desktop\TP temps\mini projet docker>
PS C:\Users\monji\Desktop\TP temps\mini projet docker> vagrant  up --provision
==> vagrant: A new version of Vagrant is available: 2.2.19 (installed version: 2.2.18)!
==> vagrant: To upgrade visit: https://www.vagrantup.com/downloads.html
```

- ***Vagrant –up –provision*** : cette commande nous permet de lancer la virtualisation du docker sur virtuel box.



Voici notre machine docker virtualiser.

```
    docker:
    docker: Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/s
ystem/docker.service.
    docker: For this Stack, you will use 172.28.128.11 IP Address
PS C:\Users\monji\Desktop\TP temps\mini projet docker> vagrant ssh
[vagrant@docker ~]$
[vagrant@docker ~]$
```

Connection via ssh pour accéder a notre machine virtuel.

NOTRE DOCKERFILE

On va clone les codes source de l'appli sur le github via la commande :
git clone https://github.com/guissepm/student-list.git

```
-rw-r--r--. 1 root    root    18617 Jan 16 12:40 get-docker.sh
drwxrwxr-x. 5 vagrant vagrant    94 Jan 16 12:43 student-list
[vagrant@docker ~]$ cd student-list/
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$ ll
total 8
-rw-rw-r--. 1 vagrant vagrant    0 Jan 16 12:43 docker-compose.yml
-rw-rw-r--. 1 vagrant vagrant 7133 Jan 16 12:43 README.md
drwxrwxr-x. 2 vagrant vagrant   70 Jan 16 12:43 simple_api
drwxrwxr-x. 2 vagrant vagrant   23 Jan 16 12:43 website
[vagrant@docker student-list]$ cd simple_api/
```

L'appli a bien été clone.

Voici notre Dockerfile qui contient toutes les informations demandes.

```dockerfile
# Utilisation de l'image python 3.9-buster
FROM python:3.8-buster
# Creation de notre container
LABEL maintainer="ulrich MONJI" email="toto@admin.fr"
# Installation des differentes paquage necessaires au demarrage de notre application
RUN apt update -y && apt install python-dev python3-dev libsasl2-dev python-dev libldap2-dev libssl-dev -y
RUN pip install flask==1.1.2 flask_httpauth==4.1.0 flask_simpleldap python-dotenv==0.14.0
# Copier le code source dans notre conteneur racine
COPY student_age.py /
# creation de la volume Data
VOLUME /data
# exposition du port 500
EXPOSE 5000
# Commande pour le lancement de l'appli Flask
CMD [ "python", "./student_age.py" ]
```

On build notre dockerfile en creant un image : api-pozos :1

```
[vagrant@docker simple_api]$ docker build -t api-pozos:1 .
 ---> 380da30298af
Step 6/8 : VOLUME  [ "/data" ]
 ---> Running in 69fb245f6516
Removing intermediate container 69fb245f6516
 ---> fa1f8070f277
Step 7/8 : EXPOSE 5000
 ---> Running in fbf9fcf4a030
Removing intermediate container fbf9fcf4a030
 ---> 1c0a032172aa
Step 8/8 : CMD [ "python", "./student_age.py" ]
 ---> Running in ce7ae8a41f66
Removing intermediate container ce7ae8a41f66
 ---> c23ebfdd6064
Successfully built c23ebfdd6064
Successfully tagged api-pozos:1
[vagrant@docker simple_api]$
```

Et on voix nos différentes partie des codes Dockerfile être exécuter.

Voici notre image docker :

```
[vagrant@docker simple_api]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
api-pozos    1         c23ebfdd6064   37 seconds ago   1.13GB
```

Création de notre conteneur :

```
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$ docker run -d --network pozos  --name test-api-pozos -v ${PWD}/student_age.json:/data/student_age.json -p 4
000:5000 api-pozos:1
5eb55b76c8d880fc46e15e4671aa4d2cff980e814294280fe09f2a836fee1172
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS                                       NAMES
5eb55b76c8d8   api-pozos:1   "python ./student_ag…"   7 seconds ago   Up 5 seconds   0.0.0.0:4000->5000/tcp, :::4000->5000/tcp   test-ap
i-pozos
[vagrant@docker simple_api]$ curl -u toto:python -X GET http://127.0.0.1:4000/pozos/api/v1.0/get_student_ages
```

Ici on va créer notre conteneur en lui donnant le nom de test-api.
Dans ce cas on utilise le network-pozos et monté le fichier student_age.json dans le volume data cree dans le port 5000.

```
i-pozos
[vagrant@docker simple_api]$ curl -u toto:python -X GET http://127.0.0.1:4000/pozos/api/v1.0/get_student_ages
{
  "student_ages": {
    "alice": "12",
    "bob": "13"
```

On teste notre conteneur en local avec l'adresse donné.

CREATION DE NOTRE DOCKER-COMPOSE

```yaml
version: '2'
services:
  web-pozos:
    image:  php:apache
    depends_on:
      - api-pozos
    ports:
      - "8082:80"
    volumes:
      - ./website:/var/www/html
    environment:
      - USERNAME=toto
      - PASSWORD=python
    networks:
      - api-pozos

  api-pozos:
    image: api-pozos:1
    ports:
      - "4000:5000"
    volumes:
      - ./simple_api/student_age.json:/data/student_age.json
    networks:
      - api-pozos

networks:
  api-pozos:
```

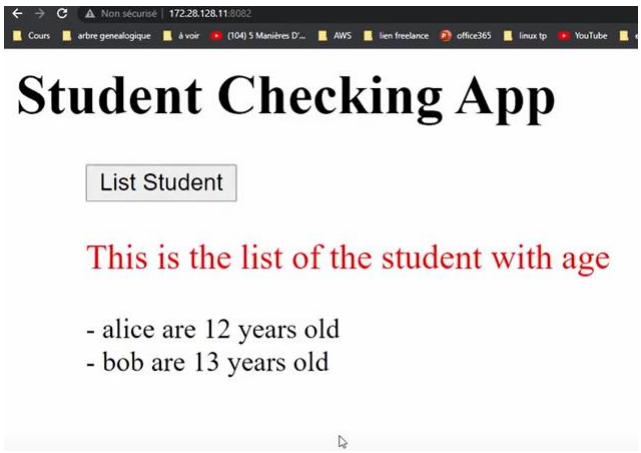
- Ici on a utilisé la version 2 du langage docker compose
- La partie web qui provient de l'image apache qui va dépendre du service api sur le port 80.
- Il est stocker dans le volume /website :/var/www/html
- L'environnement sera déployé dans un réseau api-pozos
- L'api-pozos aussi utilisera l'image api-pozos :1 sur le port 5000 Stocker dans le volume spécifié dans l'image sur le network api-pozos.

```
[vagrant@docker student-list]$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS
5eb55b76c8d8   api-pozos:1   "python ./student_ag…"   7 minutes ago   Up 7 minutes   0.0.0.0:4000->5000/tcp, :::4000->5000/tcp
i-pozos
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$ docker rm -f 5eb55b76c8d8
5eb55b76c8d8
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[vagrant@docker student-list]$
```

- Ici on vient de supprimer le conteneur de teste que nous avions créés avec la commande **rm .**

```
[vagrant@docker student-list]$ docker-compose up -d
[+] Running 14/14
 ⠿ web-pozos Pulled
   ⠿ a2abf6c4d29d Pull complete
   ⠿ c5608244554d Pull complete
   ⠿ 2d07066487a0 Pull complete
   ⠿ 1b6dfaf1958c Pull complete
   ⠿ 32c5e6a60073 Pull complete
   ⠿ 90cf855b27cc Pull complete
   ⠿ 8b0f1068c586 Pull complete
   ⠿ ec575205efc1 Pull complete
   ⠿ ce32dd59dc22 Pull complete
   ⠿ 3a063a1854de Pull complete
   ⠿ 2edfe9f6edd9 Pull complete
   ⠿ a54a54bbb211 Pull complete
   ⠿ 6b5874b920d5 Pull complete
[+] Running 3/3
 ⠿ Network student-list_api-pozos      Created
 ⠿ Container student-list-api-pozos-1  Started
 ⠿ Container student-list-web-pozos-1  Started
[vagrant@docker student-list]$ docker-compose ps
NAME                      COMMAND                  SERVICE             STATUS              PORTS
student-list-api-pozos-1  "python ./student_ag…"   api-pozos           running             0.0.0.0:4000->5000/tcp, :::4000
student-list-web-pozos-1  "docker-php-entrypoi…"   web-pozos           running             0.0.0.0:8082->80/tcp, :::8082->
[vagrant@docker student-list]$
```

- Lancement de notre docker-compose et on voit la création de nos deux conteneurs.



- On vient sur la barre de recherche de notre machine et on tape l’adresse qui nous a été affecté durant l’installation de notre machine virtuelle (docker) et notre docker-compose marche.

DOCKER REGISTRE

```
[vagrant@docker student-list]$ docker run -d -p 5000:5000 --name registry-pozos  --network student-list_api-pozos registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
79e9f2f55bf5: Pull complete
0d96da54f60b: Pull complete
5b27040df4a2: Pull complete
e2ead8259a04: Pull complete
3790aef225b9: Pull complete
Digest: sha256:169211e20e2f2d5d115674681eb79d21a217b296b43374b8e39f97fcf866b375
Status: Downloaded newer image for registry:2
c818d19a8245d582300d302dcec9f96ca830b5142f347f1da1495e8992aba341
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS                                       NAMES
c818d19a8245   registry:2    "/entrypoint.sh /etc…"   4 seconds ago   Up 3 seconds   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   registr
y-pozos
```

- On lance notre registre sur le port 5000 dans notre réseau network bridge et il va s'appelé registry-pozos.

On lance notre registre Privé avec ces commandes.

```
[vagrant@docker student-list]$
[vagrant@docker student-list]$ docker run -d  --name registry-pozos_UI --network student-list_api-pozos -p 4002:80 -e REGISTRY_
ZOS REGISTRY" -e REGISTRY_URL=http://registry-pozos:5000 -e DELETE_IMAGES=true joxit/docker-registry-ui:static
a36914cbfaae139f22de067800eef0c7c21f21914c49b0bd9320bfeb94221b2f
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$
[vagrant@docker student-list]$ docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS
               NAMES
a36914cbfaae   joxit/docker-registry-ui:static   "/docker-entrypoint.…"   3 seconds ago    Up 2 seconds    0.0.0.0:4002->80/tcp
->80/tcp       registry-pozos_UI
c818d19a8245   registry:2                     "/entrypoint.sh /etc…"   21 minutes ago   Up 21 minutes   0.0.0.0:5000->5000/
00->5000/tcp   registry-pozos
46ee8a290756   php:apache                     "docker-php-entrypoi…"   29 minutes ago   Up 29 minutes   0.0.0.0:8082->80/tcp
->80/tcp       student-list-web-pozos-1
```

Et voici l'interface de notre registre prive avec le port 4002 sur notre adresse 172.28.128.11