Application simple pour lister les étudiants avec un serveur web (PHP) et une API (Flask)

Nom : Mamadou Bamba Dieye

Introduction

Ce projet a pour objectif de mettre en place une infrastructure Docker permettant de déployer une application web et une API Flask pour la gestion d'une liste d'étudiants. L'application est destinée à la société POZOS et vise à démontrer les avantages de Docker en termes de déploiement et d'automatisation.

I. Construire et tester

Nous allons d'abord telecharger le projet avec la commande git clone :

git clone https://github.com/guissepm/student-list.git

1. **API Flask**

L'API Flask est construite à partir de l'image python:2.7-buster.

Dockerfile de l'API

```
 GNU nano 2.3.1                          File: Dockerfile

FROM python:2.7-buster
LABEL maintainer="bamba99" email="dieyebamba605@gmail.com"
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
COPY student_age.py /
VOLUME /data
EXPOSE 5000
CMD [ "python", "./student_age.py" ]
```

Ce fichier Dockerfile va définir l'image Docker pour l'API Flask.

Le fichier *simple_api/requirements.txt* qui contient les dépendances Python nécessaires pour l'API Flask.

Le fichier *simple_api/student_age.json* contient la liste des étudiants avec leur âge.

Le fichier *simple_api/student_age.py* contient le code source de l'API Flask.

Le fichier *website/index.php* contient le code PHP pour l'interface web.

Pour construire l'image Docker de l'API, nous avons utilisé la commande suivante :

docker build -t api-pozos:1 .

```
[vagrant@docker simple_api]$ docker build -t api-pozos:1 .
[+] Building 3.6s (9/9) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                         0.1s
 => => transferring dockerfile: 461B                                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:2.7-buster                                                         2.7s
 => [internal] load .dockerignore                                                                                            0.1s
 => => transferring context: 2B                                                                                              0.0s
 => [1/4] FROM docker.io/library/python:2.7-buster@sha256:d8fac68ebdc45b8d66d53f1ed6c1532da81109a8f5532a6ca0c951ed31107d70    0.0s
 => [internal] load build context                                                                                            0.1s
 => => transferring context: 95B                                                                                             0.0s
 => CACHED [2/4] RUN apt update -y && apt install python-dev python3-dev libsasl2-dev python-dev libldap2-dev libssl-dev -y  0.0s
 => CACHED [3/4] RUN pip install flask==1.1.2 flask_httpauth==4.1.0 flask_simpleldap python-dotenv==0.14.0                   0.0s
 => CACHED [4/4] COPY student_age.py /                                                                                       0.0s
 => exporting to image                                                                                                       0.1s
 => => exporting layers                                                                                                      0.0s
 => => writing image sha256:7720b189665ccfc94aa6ef49a59c77def10ee5f3eec2c6a5296e127cfa5e91f3                                 0.0s
 => => naming to docker.io/library/api-pozos:1                                                                               0.0s
[vagrant@docker simple_api]$
```

Après l'exécution de cette commande, l'image Docker de l'API est créée.

```
[vagrant@docker simple_api]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
api-pozos    1         7720b189665c   3 minutes ago    1.14GB
[vagrant@docker simple_api]$
```

Nous pouvons voir ici que l'image a été créée.

Nous allons exécutez la commande suivante pour nous assurer que l'API répond correctement :

curl -u toto:python -X GET http://192.168.56.5:4000/pozos/api/v1.0/get_student_ages

```
[vagrant@docker-vm student-list]$ curl -u toto:python -X GET http://192.168.56.5:4000/pozos/api/v1.0/get_student_ages
{
  "student_ages": {
    "alice": "12",
    "bob": "13"
  }
}
[vagrant@docker-vm student-list]$
```

Nous constatons que l'API répond correctement.

II. Infrastructure As Code

Après avoir testé notre image API, nous devons tout assembler et le déployer à l'aide de docker-compose.

Nous allons éditer *docker-compose.yml* :

```
  GNU nano 2.3.1                    File: docker-compose.yml

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
```

Nous allons supprimer le conteneur précédemment créé :

```
[vagrant@docker-vm student-list]$ docker rm 2d242ef215d9
2d242ef215d9
[vagrant@docker-vm student-list]$
```

Pour exécuter l'ensemble des services définis dans docker-compose.yml, il suffit de lancer la commande suivante :

docker-compose up –d :

```
[vagrant@docker student-list]$ docker-compose up -d
Creating network "student-list_api-pozos" with the default driver
Pulling web-pozos (php:apache)...
apache: Pulling from library/php
7cf63256a31a: Pull complete
b069e2d43501: Pull complete
9c4ebbec2218: Pull complete
591e2fd0e37e: Pull complete
ea39dbc21557: Pull complete
6827e0405026: Pull complete
ef9823796ab7: Pull complete
f1fe2996e66b: Pull complete
ac59bdfc6c0f: Pull complete
d3160fc0d00e: Pull complete
b365602aea53: Pull complete
0047b9e64465: Pull complete
8809fb74d31b: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:48572bd03e6bb0b08004893afa22fec0cf17ddcd1b95f32f8b7723064e16e975
Status: Downloaded newer image for php:apache
Creating student-list_api-pozos_1 ... done
Creating student-list_web-pozos_1 ... done
[vagrant@docker student-list]$
```

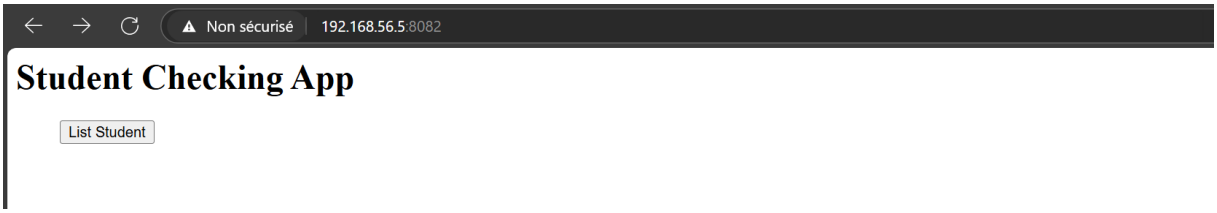
Vérification des conteneurs en cours d'exécution

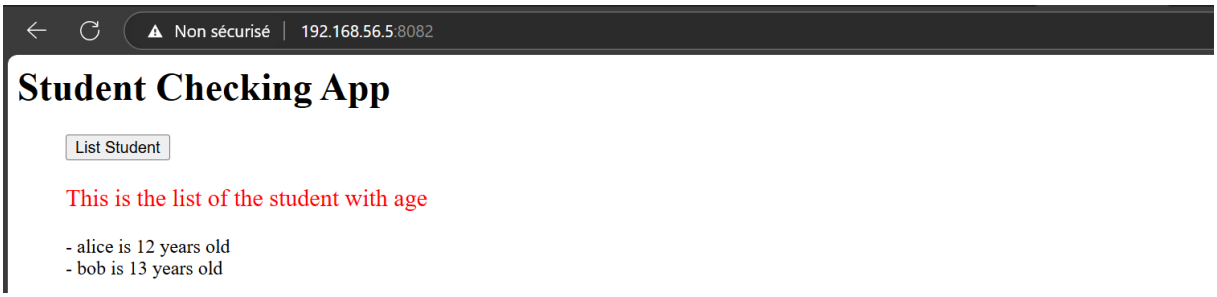
```
[vagrant@docker student-list]$ docker-compose ps
          Name                        Command               State                    Ports
-------------------------------------------------------------------------------------------------------------
student-list_api-pozos_1   python ./student_age.py          Up      0.0.0.0:5000->5000/tcp,:::5000->5000/tcp
student-list_web-pozos_1   docker-php-entrypoint apac ...   Up      0.0.0.0:8082->80/tcp,:::8082->80/tcp
[vagrant@docker student-list]$
```

```
[vagrant@docker student-list]$ docker network ls
NETWORK ID     NAME                          DRIVER    SCOPE
6c2b0126364e   bridge                        bridge    local
3594ff7c06ce   host                          host      local
2cdf5bd2d92d   none                          null      local
e2d1d07426f9   student-list_api-pozos         bridge    local
```

Nous pouvons aussi voir le réseau *student-list_api-pozos*.

Enfin, nous allons accéder à notre site web et cliquez sur le bouton « List Student »





Nous voyons que la liste de l'étudiant apparaît, nous avons dockeriser avec succès l'application POZOS.

III. Registre Docker

Exécutons la commande suivante pour démarrer un registre privé sur votre machine :

docker run -d -p 5000:5000 --name registry-pozos --network student-list_api-pozos registry:2

```
[vagrant@docker-vm student-list]$ docker run -d -p 5000:5000 --name registry-pozos --network student-list_api-pozos regi
stry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
44cf07d57ee4: Pull complete
bbbdd6c6894b: Pull complete
8e82f80af0de: Pull complete
3493bf46cdec: Pull complete
6d464ea18732: Pull complete
Digest: sha256:a3d8aaa63ed8681a604f1dea0aa03f100d5895b6a58ace528858a7b332415373
Status: Downloaded newer image for registry:2
d5e6efcfdd540fbf419624f47ef3f027e9a1e6b25902f7f8be1b897211c3bc9d
```

Cela démarre un registre privé sur le port 5000.

```
[vagrant@docker-vm student-list]$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS
          NAMES
d5e6efcfdd54   registry:2    "/entrypoint.sh /etc…"   12 seconds ago   Up 10 seconds   0.0.0.0:5000->5000/tcp, :::5000->
5000/tcp   registry-pozos
e64a8ff53284   php:apache    "docker-php-entrypoi…"   8 minutes ago    Up 8 minutes    0.0.0.0:8082->80/tcp, :::8082->80
/tcp       student-list_web-pozos_1
2d242ef215d9   api-pozos:1   "python ./student_ag…"   8 minutes ago    Up 8 minutes    0.0.0.0:4000->5000/tcp, :::4000->
5000/tcp   student-list_api-pozos_1
[vagrant@docker-vm student-list]$
```

Nous pouvons voir ici que le registre privé est créé.

Nous allons taguer l'image pour notre registre privé

```
[vagrant@docker-vm student-list]$ docker image tag api-pozos:1 localhost:5000/api-pozos:1
[vagrant@docker-vm student-list]$ docker images
REPOSITORY                 TAG       IMAGE ID       CREATED             SIZE
api-pozos                  1         0da858bece51   About an hour ago   1.14GB
localhost:5000/api-pozos   1         0da858bece51   About an hour ago   1.14GB
php                        apache    a32aa62d4c04   5 days ago          514MB
registry                   2         26b2eb03618e   17 months ago       25.4MB
[vagrant@docker-vm student-list]$
```

Cela associe l'image api-pozos au registre privé.

Nous allons pousser l'image vers le registre

```
[vagrant@docker-vm student-list]$ docker push localhost:5000/api-pozos:1
The push refers to repository [localhost:5000/api-pozos]
72e517f63d5c: Pushed
8c8eb42e6131: Pushed
42f1d2fa12d8: Pushed
e571d2d3c73c: Pushed
da7b0a80a4f2: Pushed
ceee8816bb96: Pushed
47458fb45d99: Pushed
46829331b1e4: Pushed
d35c5bda4793: Pushed
a3c1026c6bcc: Pushed
f1d420c2af1a: Pushed
461719022993: Pushed
1: digest: sha256:eff7be2a67042d3b6bca986601ff0549b010d512aa2781438df4b480f98e5719 size: 2852
[vagrant@docker-vm student-list]$
```

L'image sera stockée dans le registre privé.

Vérifions que l'image est bien poussée :

```
[vagrant@docker-vm student-list]$ curl -X GET http://localhost:5000/v2/_catalog
{"repositories":["api-pozos"]}
[vagrant@docker-vm student-list]$ docker ps -a
```

L'image est bien poussée.

Interface Web pour gérer le registre

Nous allons maintenant ajouter une interface web pour visualiser les images.

Exécutons la commande suivante pour déployer une interface web (Docker Registry UI):

docker run –d –p –name registry-pozos_UI –network student-list_api-pozos –p 4002 :80 -e REGISTRY_TITLE= »POZOS REGISTRY» -e REGISTRY_URL=http://registry-pozos :5000 -e CATALOG_ELEMENTS_LIMIT=1000 joxit/docker-registry-ui :static

```
[vagrant@docker-vm student-list]$ docker run -d --name registry-pozos_UI --network student-list_api-pozos -p 4002:80 -e
REGISTRY_TITLE="POZOS REGISTRY" -e REGISTRY_URL=http://registry-pozos:5000 -e DELETE_IMAGES=true -e CATALOG_ELEMENTS_LIM
IT=1000 joxit/docker-registry-ui:static
```

```
[vagrant@docker-vm student-list]$ docker run -d  --name registry-pozos_UI --network student-list_api-pozos -p 4002:80 -e
 REGISTRY_TITLE="POZOS REGISTRY" -e REGISTRY_URL=http://registry-pozos:5000 -e DELETE_IMAGES=true joxit/docker-registry-
ui:static
Unable to find image 'joxit/docker-registry-ui:static' locally
static: Pulling from joxit/docker-registry-ui
540db60ca938: Pull complete
197dc8475a23: Pull complete
39ea657007e5: Pull complete
37afbf7d4c3d: Pull complete
0c01f42c3df7: Pull complete
d590d87c9181: Pull complete
3333c94ae44f: Pull complete
33d7cca6fc9f: Pull complete
076b2dd9bdd1: Pull complete
b70198f04ee7: Pull complete
1fb6c5acc953: Pull complete
Digest: sha256:b0657b6be748173583516e411bd71552e54cb7d5dda94964726297ce8774415c
Status: Downloaded newer image for joxit/docker-registry-ui:static
6ea7e18528be435c83be12b89ee77fc85486cea5290db16924f692b4b1fa23ac
[vagrant@docker-vm student-list]$
```

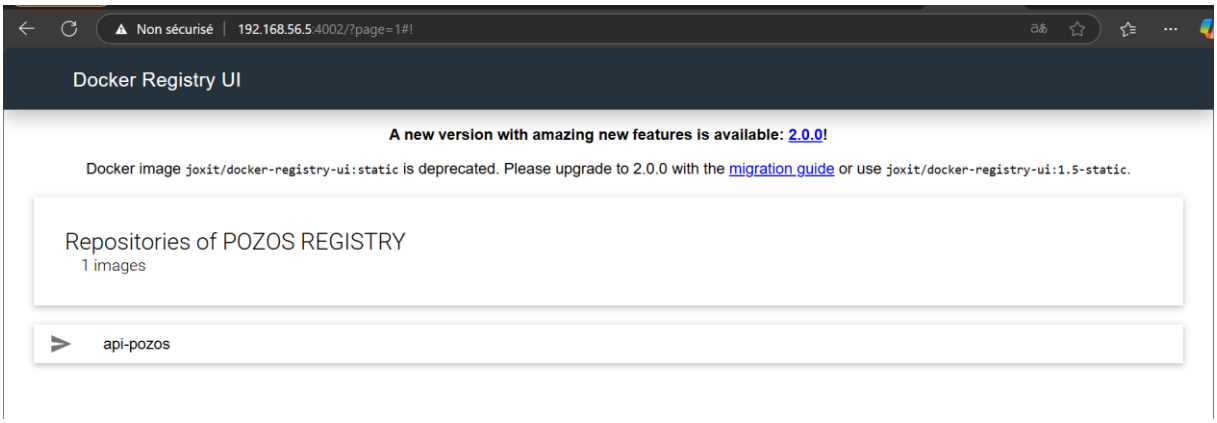
Cela démarre une interface web accessible sur http://192.168.56.5:4002

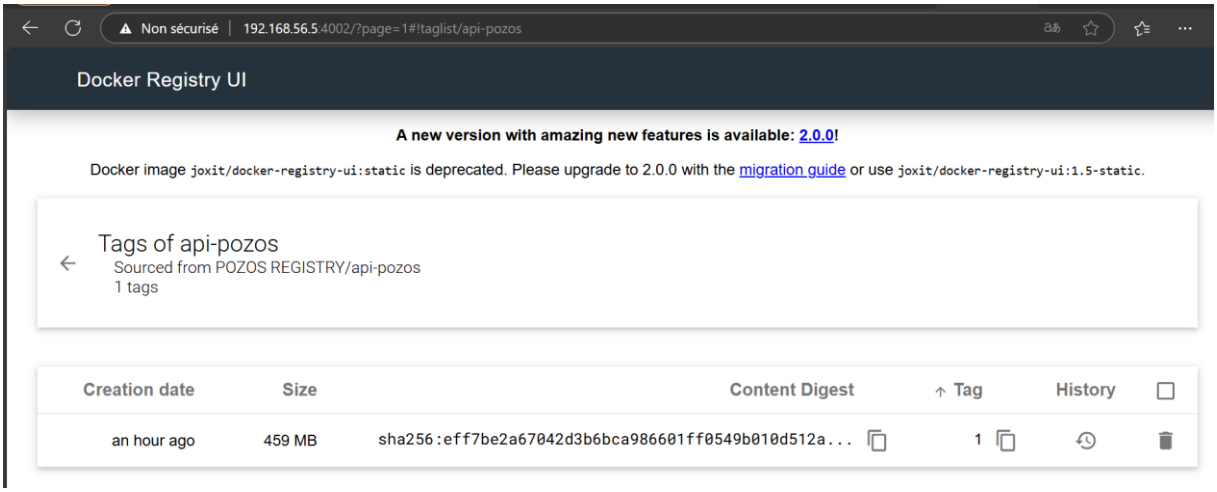
```
[vagrant@docker-vm student-list]$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS
                         NAMES
6ea7e18528be   joxit/docker-registry-ui:static   "/docker-entrypoint.…"   28 seconds ago   Up 25 seconds   0.0.0.0:4002-
>80/tcp, :::4002->80/tcp       registry-pozos_UI
d5e6efcfdd54   registry:2                           "/entrypoint.sh /etc…"   10 minutes ago   Up 10 minutes   0.0.0.0:5000-
>5000/tcp, :::5000->5000/tcp   registry-pozos
e64a8ff53284   php:apache                           "docker-php-entrypoi…"   19 minutes ago   Up 19 minutes   0.0.0.0:8082-
>80/tcp, :::8082->80/tcp       student-list_web-pozos_1
2d242ef215d9   api-pozos:1                          "python ./student_ag…"   19 minutes ago   Up 19 minutes   0.0.0.0:4000-
>5000/tcp, :::4000->5000/tcp   student-list_api-pozos_1
[vagrant@docker-vm student-list]$
```

Vérifions que l'image image est bien présente dans notre registre en listant les images disponibles :

Accéder à l'interface :

Ouvrons un navigateur et taper l'adresse : *http://192.168.56.5:4002*





Nous pouvons voir *api-pozos* dans la liste des images.

Conclusion

Ce projet a permis de démontrer l'efficacité de Docker pour déployer une application découplée, composée d'une API Flask et d'une interface web PHP, tout en garantissant une gestion simplifiée et scalable de l'infrastructure. Grâce à l'utilisation de docker-compose, les services ont été orchestrés de manière automatisée, assurant une interaction fluide entre l'API et le site web. Enfin, la mise en

place d'un registre Docker privé a renforcé la gestion des versions et la reproductibilité du déploiement, répondant ainsi aux besoins d'agilité et de sécurité de POZOS.