Niama Aroubite M2 ILW

Rapport : Mini-Projet Docker

1. déploiement de l'api flask :

Pour déployer l'application API flask, en utilisant Docker on doit développer une image docker
Pour cela, on vas créer un fichier Dockerfile qui contient la description de l'image:

```
# To build API image you must use "python:2.7-stretch"
FROM python:2.7-stretch
# Please don't forget to specify the maintainer information
LABEL org.opencontainers.image.authors="aroubiteniama@gmail.com"
# You need to copy the source code of the API in the container at the root "/" path
COPY ./student_age.py /
# The API is using FLASK engine,  here is a list of the package you need to install
RUN apt-get update -y && apt-get 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
# Persistent data (volume)
VOLUME [ "/data" ]
COPY ./student_age.json /data
# To interact with this API expose 5000 port
EXPOSE 5000
# When container start, it must run the student_age.py
CMD [ "python", "/student_age.py" ]
```

1.1 : image de base :

Dans notre image, nous se basons sur l'image python:2.7-stretch :

```
# To build API image you must use "python:2.7-stretch"
FROM python:2.7-stretch
```

1.2 Maintainer :

En lisant la documentation, nous constatons que l'instruction MAINTAINER est dépréciée, et il est recommandé d'utiliser LABEL comme ceci :

```
# Please don't forget to specify the maintainer information
LABEL org.opencontainers.image.authors="aroubiteniama@gmail.com"
```

1.3 code source :

Notre container a besoin de notre code python pour s'exécuter, donc nous devons copier le code dans le système de fichier de notre container :

```
# You need to copy the source code of the API in the container at the root "/" path
COPY ./student_age.py /
```

1.4. dependence :

Notre application a besoin de plusieurs dépendances système et applicatif.

```
# The API is using FLASK engine,  here is a list of the package you need to install
RUN apt-get update -y && apt-get 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
```

1.4 volume :

Pour la persistance des données, on va créer un répertoire et on va le définir comme un volume :

```
# Persistent data (volume)
RUN mkdir /data
VOLUME [  "/data"]
COPY ./student_age.json /data
```

1.5 port :

Pour que notre application puisse communiquer avec le monde extérieur on doit l'exposer dans le port 5000.

```
# To interact with this API expose 5000 port
EXPOSE 5000
```

1.5 lancement :

Pour lancer l'application on doit exécuter le script concerné avec l'interpréteur python :

```
# When container start, it must run the student_age.py
CMD [ "python", "/student_age.py" ]
```

2.Build :

Après le développement de notre dockerfile, il arrive le temps de créer l'image à partir de Dockerfile :

```
PS C:\data\projets\pycharm\student-list> docker build -t api-flask-niama .\simple_api\.
[+] Building 105.8s (4/9)
 => [1/5] FROM docker.io/library/python:2.7-stretch@sha256:548e680020444b0f6ddc4c7b0c24964d1af5f47cd2e2b3b44d742852b8b09cfc
 => => resolve docker.io/library/python:2.7-stretch@sha256:548e680020444b0f6ddc4c7b0c24964d1af5f47cd2e2b3b44d742852b8b09cfc
 => => sha256:4a9f2207c812b2530bc06352aa57cf50244726c52db6ff775ca9f9a76d6ea880 10.80MB / 10.80MB
 => => sha256:6fe350d2b14088b19ffff7a1c6683f0949fb0161e6ab87011a24e21e5d7fed4e 4.34MB / 4.34MB
[+] Building 106.0s (4/9)
 => [1/5] FROM docker.io/library/python:2.7-stretch@sha256:548e680020444b0f6ddc4c7b0c24964d1af5f47cd2e2b3b44d742852b8b09cfc
 => => sha256:548e680020444b0f6ddc4c7b0c24964d1af5f47cd2e2b3b44d742852b8b09cfc 1.65kB / 1.65kB
 => => sha256:58e0d45ffe8fca0cde7faecb39ff4f9a0329d83c856bb196eda3b385f77ed529 2.22kB / 2.22kB
 => => sha256:e71fc5c0fcb1f3dde916cb124834175cabf8d70196d5b3d093012e1b928a6f11 8.94kB / 8.94kB
 => => sha256:7568c21980bd8003ca9d23a218302c6386aac91e069cc0c0be6bedf45476f056 16.78MB / 45.38MB
 => => sha256:d95a2fdc8b3d1140238f9ea09e64cff45e839a31e5e5d25309af9fa7f0c331eb 5.24MB / 50.09MB
```

3.L'exécution du container :

finalement, on exécute notre container :

```
PS C:\data\projets\pycharm\student-list> docker run -p 5000:5000 api-flask-niama
 * Serving Flask app "student_age" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 173-910-599
172.17.0.1 - - [06/Oct/2021 09:25:22] "GET /pozos/api/v1.0/get_student_ages HTTP/1.1" 401 -
172.17.0.1 - - [06/Oct/2021 09:25:27] "GET /pozos/api/v1.0/get_student_ages HTTP/1.1" 401 -
172.17.0.1 - - [06/Oct/2021 09:25:29] "GET /favicon.ico HTTP/1.1" 404 -
```

4.up docker compose:

Pour lancer les deux application en même temps, on utilise docker-compose.yml :

5.Docker registry :

On doit lancer le registre locale :

```
PS C:\data\projets\pycharm\student-list> docker run -d -p 5001:5001 --name registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
```

Push notre image:

```
PS C:\data\projets\pycharm\student-list> docker image tag api-flask-niama localhost:5000/api-flask-niama
PS C:\data\projets\pycharm\student-list> docker push localhost:5001/api-flask-niama
Using default tag: latest
The push refers to repository [localhost:5001/api-flask-niama]
```