

## Build and test

### Création du DockerFile :

1. Lire attentivement le readMe pour connaître toutes les versions qu'il nous faudra.

2. 

```
C: > Users > 33664 > Desktop > Ensup > M2 > Docker > student-list-master > simple_api > Dockerfile
1 FROM python:2.7-stretch
```

FROM permet de définir notre image de base, vous pouvez l'utiliser seulement une fois dans un Dockerfile. Notre image sera basée sur python:2.7-stretch selon le readMe.

- 3.

```
LABEL Maintainer="Flavien Annaix <flavien.annaix@gmail.com>"
```

Puis dire qui est le créateur de ce dockerFile pour s'il est partagé pouvoir le contacter pour faire des mises à jour.

- 4.

```
RUN apt-get update -y && \
    apt-get install python-dev python3-dev libsass12-dev python-dev libldap2-dev libssl-dev -y
RUN pip install flask flask_httpauth flask_simpleldap python-dotenv
```

Définir une commande RUN qui permet de lancer une commande et cette commande est donnée dans le readMe.

- 5.

```
COPY ./student_age.py /
```

Cette ligne sert à copier le fichier student\_age.py dans le répertoire / .

- 6.

```
VOLUME [ "/data/" ]
```

La commande VOLUME sert à monter un volume pour la persistance des données et donc partager les données, notre volume sera nommé data.

- 7.

```
EXPOSE 5000
```

La commande EXPOSE sert à savoir quel est le port de l'api, ici on l'expose sur le port 5000.

8.

```
CMD [ "python", "/student_age.py" ]
```

CMD est une instruction qui va être exécutée lors du lancement du conteneur. Et cette commande lance l'api python.

## Build de l'image avec le DockerFile :

1. Allez dans le répertoire où se situe le dockerFile.
2. lancez la commande "docker build -t simple-api:v1 ."

```
C:\Users\33664\Desktop\Ensup\M2\Docker\student-list-master\simple_api>docker build -t simple-api:v1 .
[+] Building 1.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 423B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/python:2.7-stretch            1.2s
=> [auth] library/python:pull token for registry-1.docker.io                    0.0s
=> [internal] load build context                                                 0.0s
=> => transferring context: 36B                                                  0.0s
=> [1/4] FROM docker.io/library/python:2.7-stretch@sha256:548e680020444b0f6ddc4c7b0c24964d1af5f47cd2e2b3b44d7428 0.0s
=> CACHED [2/4] RUN apt-get update -y && apt-get install python-dev python3-dev libsasl2-dev python-dev libl 0.0s
=> CACHED [3/4] RUN pip install flask flask_httpauth flask_simpleldap python-dotenv 0.0s
=> CACHED [4/4] COPY ./student_age.py /                                          0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:a34bd3b35969c76c573c544117976295fa0c41f956ca799430bb6ca6d08f4e08 0.0s
=> => naming to docker.io/library/simple-api:v1                                0.0s
```

3. Pour run "docker run --name simple-api-app -d -v  
"C:/Users/33664/Desktop/Ensup/M2/Docker/student-list-master/simpl  
e\_api/student\_age.json:/data/student\_age.json" -p 5000:5000  
simple-api:v1"
4. Le résultat de la commande curl

```
C:\Users\33664\Desktop\Ensup\M2\Docker\student-list-master\simple_api>curl -u toto:python -X GET http://127.0.0.1:5000/p
ozos/api/v1.0/get_student_ages
{
  "student_ages": {
    "alice": "12",
    "bob": "13"
  }
}
```

# Infrastructure As Code

## Création du Docker-compose.yml :

1. Pour commencer le docker-compose.yml il faut préciser la version.

```
> Users > 33664 > Desktop
1  version: "1"
```

2. Ensuite il faut définir la liste des services

```
version:
services:
```

3. Pour ce projet il nous faut 2 services website et api
4. Le 1er service website et le 2eme api

```
website:
  image: php:apache
  container_name: 'website'
  restart: 'always'
  hostname: website
  depends_on:
    - api
  environment:
    - PASSWORD=python
    - USERNAME=toto
  ports:
    - "80:80"
  volumes:
    - ./website:/var/www/html
  networks:
    - pozos_network

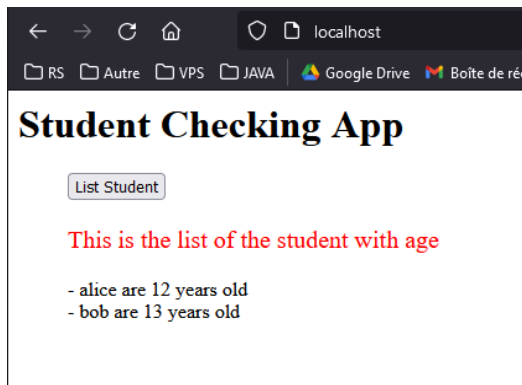
api:
  image: simpleapi:v1
  container_name: 'simple-api'
  build: './simple_api'
  restart: 'always'
  hostname: api
  volumes:
    - ./simple_api/student_age.json:/data/student_age.json
  ports:
    - "5000:5000"
  networks:
    - pozos_network
```

- image ⇒ sert à définir l'image du conteneur
  - container\_name ⇒ permet de nommer le conteneur (équivalent à --name)
  - restart : 'always' ⇒ redémarre tous les services arrêtés et en cours d'exécution.
  - depends\_on ⇒ qui prend en paramètre la liste des services dont dépend un conteneur, et de vérifier si les services qui en dépend soit lancer
  - environment ⇒ sert à définir les variables d'environnement
  - ports ⇒ sert à savoir quel est le port de l'api.
  - volumes ⇒ est la liste des volumes que l'on souhaite monter dans le conteneur
  - networks ⇒ permet de s'attacher à un réseau et de s'y attacher
  - build ⇒ sert a build le dockerfile dans le répertoire en paramètre
5. définition des networks

```
networks:
  pozos_network:
```

## Build et lancer le conteneur :

1. Se rendre dans le répertoire où se situe le fichier Docker-compose.yml
2. Lancement de la commande "docker compose up -d"
3. Se rendre sur l'adresse web <http://localhost:80/>
4. Voici le résultat



## Docker Registry

### Installation d'un registry private :

Avec les commande Docker

1. `docker network create --driver bridge --subnet 192.168.1.0/24 lan0`
2. `docker run --name private_registry -d -e REGISTRY_STORAGE_DELETE_ENABLED=true --network lan0 -p 5001:5000 registry:2`
3. `docker run --name registry-ui -d --network lan0 -e REGISTRY_URL="http://private_registry:5000" -e DELETES_IMAGES=true -e REGISTRY_TITLE="Flavien" -p 8090:80 joxit/docker-registry-ui:static`

Avec un docker-compose.yml

```
private_registry1:
  image : registry:2
  ports:
    - 5001:5000
  environment:
    - REGISTRY_STORAGE_DELETE_ENABLED=true
registry-ui1:
  image: joxit/docker-registry-ui:static
  depends_on:
    - private_registry1
  ports:
    - 8050:80
  environment:
    - REGISTRY_URL=http://private_registry1:5000
    - DELETES_IMAGES=true
    - REGISTRY_TITLE=Flavien
```

## Envoyer une image sur le Registry priver

1. Tager l'image que nous avons build

```
C:\Users\33664\Desktop\Ensup\M2\Docker\student-list-master>docker tag simpleapi:v1 localhost:5001/simpleapiprozo
```

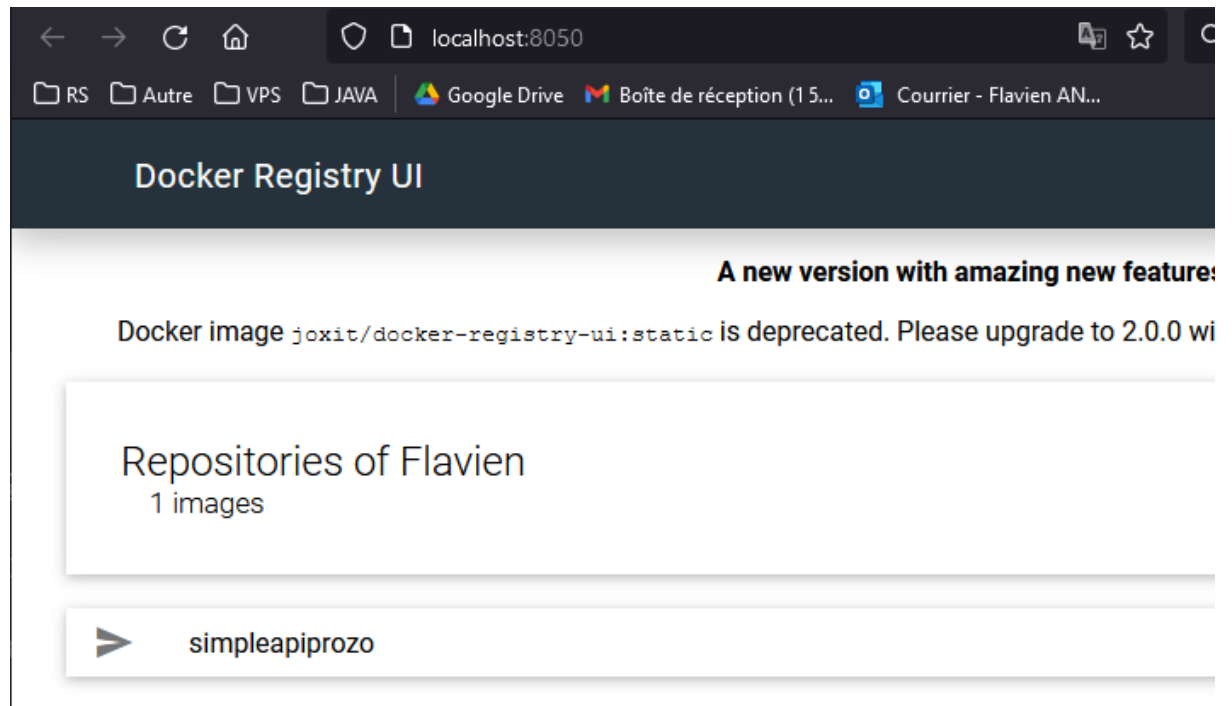
2. Puis push l'image sur le registry private

```
C:\Users\33664\Desktop\Ensup\M2\Docker\student-list-master>docker push localhost:5001/simpleapiprozo
```

3. L'image se met sur le registry

```
C:\Users\33664\Desktop\Ensup\M2\Docker\student-list-master>docker push localhost:5001/simpleapiprozo
Using default tag: latest
The push refers to repository [localhost:5001/simpleapiprozo]
ced2b86b3936: Pushed
07135634780b: Pushed
3de0ac6bcee4: Pushed
811b6c5694d4: Mounted from sadofrazer/api-student-list
1855932b077c: Mounted from sadofrazer/api-student-list
fa28e7fcadc2: Mounted from sadofrazer/api-student-list
4427a3d9a321: Mounted from sadofrazer/api-student-list
4a03ae8d3bee: Mounted from sadofrazer/api-student-list
a9286fedbd63: Mounted from sadofrazer/api-student-list
d50e7be1e737: Mounted from sadofrazer/api-student-list
6b114a2dd6de: Mounted from sadofrazer/api-student-list
bb9315db9240: Mounted from sadofrazer/api-student-list
latest: digest: sha256:c1adb1fc5c256a74e85f548896af42e14794bf0310730eb1de39d26ae86e8d93 size: 2853
```

4. Vérifier sur le site web



# SWARM

1. Initialisation du swarm

```
docker swarm init --advertise-addr 10.0.1.3
```

avec l'adresse ip du manager

2. Dans une autre machine taper la commande

```
docker swarm join --token SWMTKN-1-26grjp6itpmvntgkvab4yknllp93o3hlp6ndq7btyfrpemqr-1625wzyyxnfoae4c7si25c3pe 10.0.1.3:2377
```

Pour être rajouter au swarm et pouvoir accéder au conteneur

3. Dans le docker-compose dans le service api

```
deploy:
  mode: replicated
  replicas: 2
  placement:
    constraints: [node.role == worker]
```

On définit le mode ici replicated le nombre de réplicas ici 2 et le placement qu'on définit ici que sur les machines worker

4. Dans le docker-compose dans le service website

```
deploy:
  mode: replicated
  replicas: 1
  placement:
    constraints: [node.role == manager]
```

On définit le mode ici replicated le nombre de réplicas ici 1 et le placement qu'on définit ici que sur la machine manager.

Lien du github :

<https://github.com/Flav1-ann/Docker-student-list>