

MINI-PROJET

1- La création du Dockerfile

```
mar. juil. 11 20:02:57
Dockerfile - vagrant [SSH: 192.168.56.12] - Visual Studio Code

Dockerfile M X docker-compose.yml M index.php M

student-list > simple_api > Dockerfile > ...
1 FROM python:2.7-buster AS base
2
3 LABEL MAINTAINER="kamgaingrodrigue3@gmail.com"
4 VOLUME [ "/data" ]
5 EXPOSE 5000
6 COPY student_age.py /
7
8 FROM base AS build
9 RUN apt update -y && apt install python-dev python3-dev libsasl2-dev python-dev libldap2-dev libssl-dev -y
10
11 RUN pip install flask==1.1.2 flask_httpauth==4.1.0 flask_simpleldap python-dotenv==0.14.0
12
13 #ADD git clone https://github.com/diranetafen/student-list.git /
14
15 #RUN cd student-list/
16 |
17 FROM build AS final
18 CMD ["python", "./student_age.py"]
```

2- Création de l'image de l'api

docker build -t api-image:1.0 .

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
bash - simple_api + v [ ] ... ^ X

[vagrant@docker1 simple_api]$ docker build -t api-image:1.0 .
[+] Building 0.8s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 593B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:2.7-buster
=> [internal] load build context
=> => transferring context: 95B
=> [base 1/2] FROM docker.io/library/python:2.7-buster@sha256:d8fac68ebdc45b8d66d53f1ed6c1532da81109a8f5532a6ca0c951ed31107d70
=> CACHED [base 2/2] COPY student_age.py /
=> CACHED [build 1/2] RUN apt update -y && apt install python-dev python3-dev libsasl2-dev python-dev libldap2-dev libssl-dev -y
=> CACHED [build 2/2] RUN pip install flask==1.1.2 flask_httpauth==4.1.0 flask_simpleldap python-dotenv==0.14.0
=> exporting to image
=> => exporting layers
=> => writing image sha256:4ac3ce5d2f2d155855909598e489b866438a1f5ee1b6e6e52dc8c458a2ed766f
=> => naming to docker.io/library/api-image:1.0
[vagrant@docker1 simple_api]$ docker build -t api-image:1.0 .

Ln 16, Col 1 Spaces: 4 UTF-8 LF Dockerfile
```



```
Unable to find image 'api-image:latest' locally
docker: Error response from daemon: pull access denied for api-image, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.

[vagrant@docker1 simple_api]$ docker build -t api-image:1.0 .
[+] Building 0.8s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 593B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:2.7-buster
=> [internal] load build context
=> => transferring context: 95B
=> [base 1/2] FROM docker.io/library/python:2.7-buster@sha256:d8fac68ebdc45b8d66d53f1ed6c1532da81109a8f5532a6ca0c951ed31107d70
=> CACHED [base 2/2] COPY student_age.py /
=> CACHED [build 1/2] RUN apt update -y && apt install python-dev python3-dev libssl2-dev python-dev libldap2-dev libssl-dev -y
=> CACHED [build 2/2] RUN pip install flask==1.1.2 flask_httpauth==4.1.0 flask_simpleldap python-dotenv==0.14.0
=> exporting to image
=> => exporting layers
=> => writing image sha256:4ac3ce5d2f2d155855909598e489b866438a1f5ee1b6e6e52dc8c458a2ed766f
=> => naming to docker.io/library/api-image:1.0

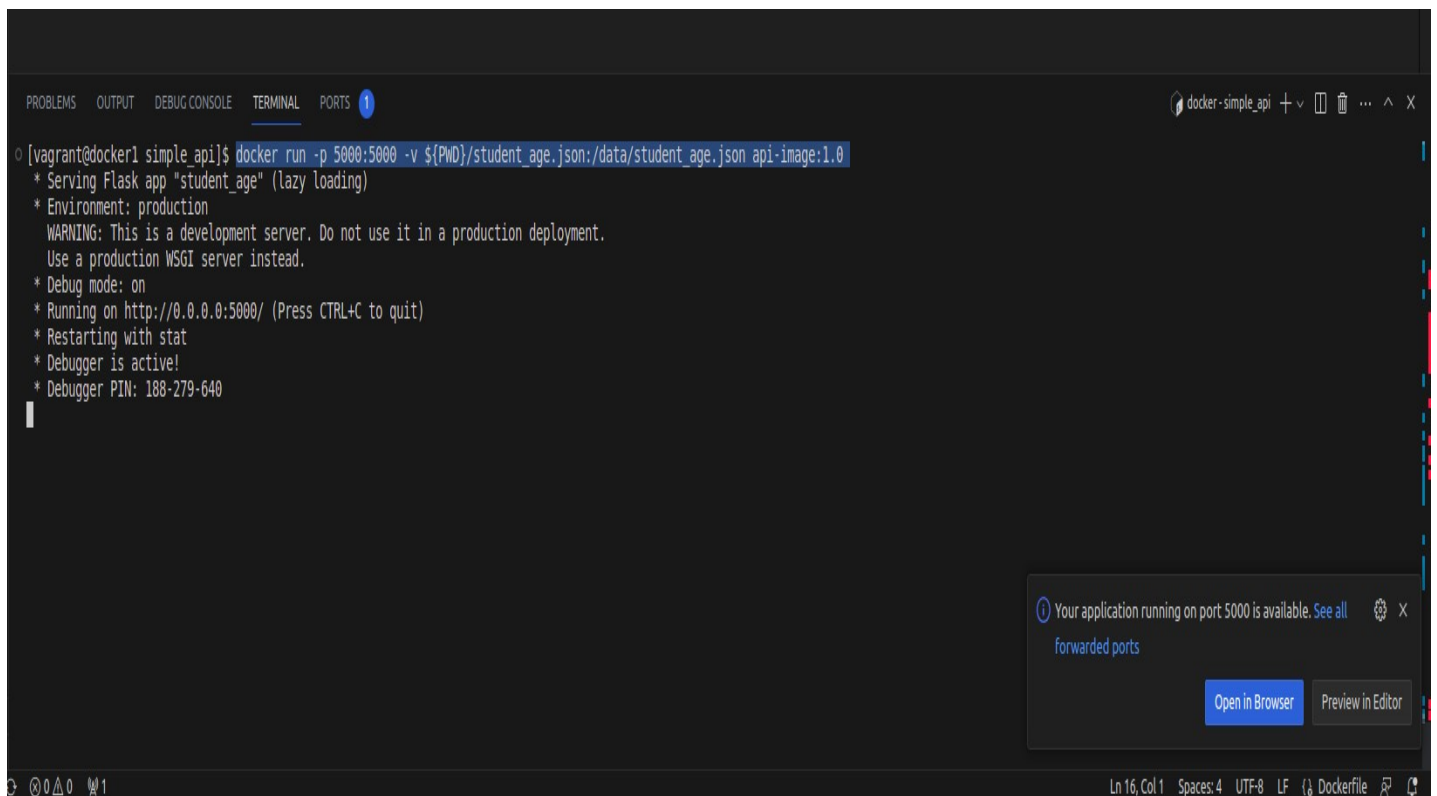
[vagrant@docker1 simple_api]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
api-image 1.0 4ac3ce5d2f2d 7 hours ago 1.14GB

[vagrant@docker1 simple_api]$
```

Le nom de l'image créée est : **api-image**

3- Lancement du conteneur avec l'image précédente

docker run -p 5000:5000 -v \${PWD}/student_age.json:/data/student_age.json api-image:1.0



```
[vagrant@docker1 simple_api]$ docker run -p 5000:5000 -v ${PWD}/student_age.json:/data/student_age.json api-image:1.0
* 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: 188-279-640
```

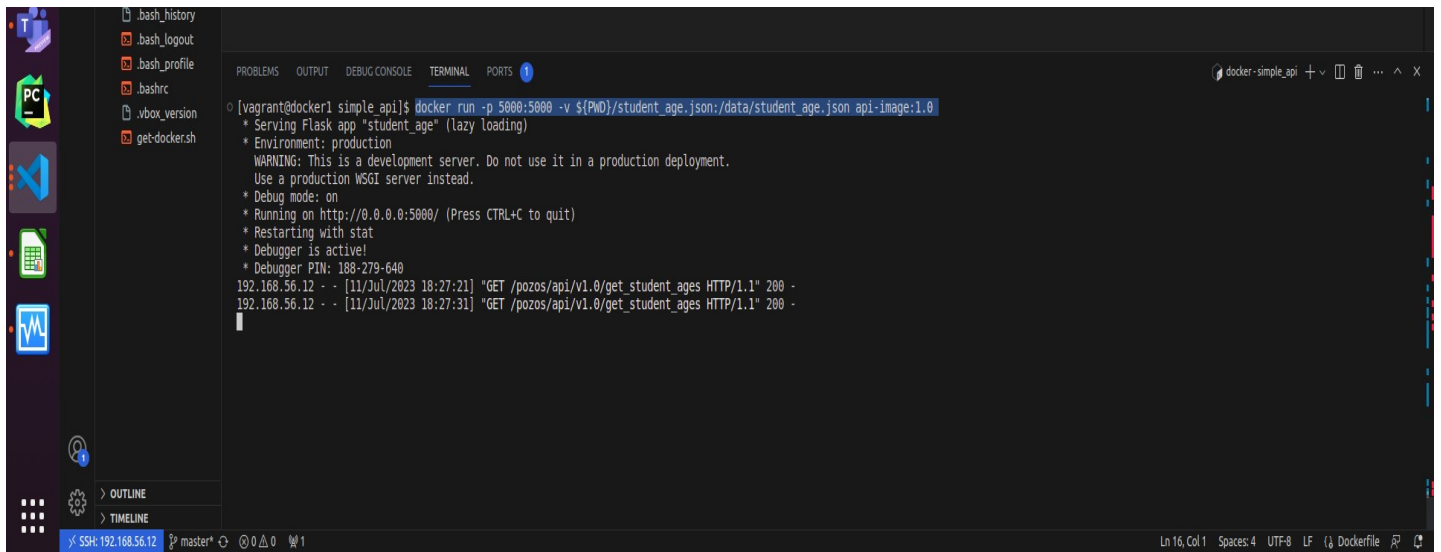
ⓘ Your application running on port 5000 is available. See all forwarded ports

Open in Browser Preview in Editor

4 – Test de l'api

L'adresse sur laquelle le conteneur est déployée : 192.168.56.12 sur le port 5000

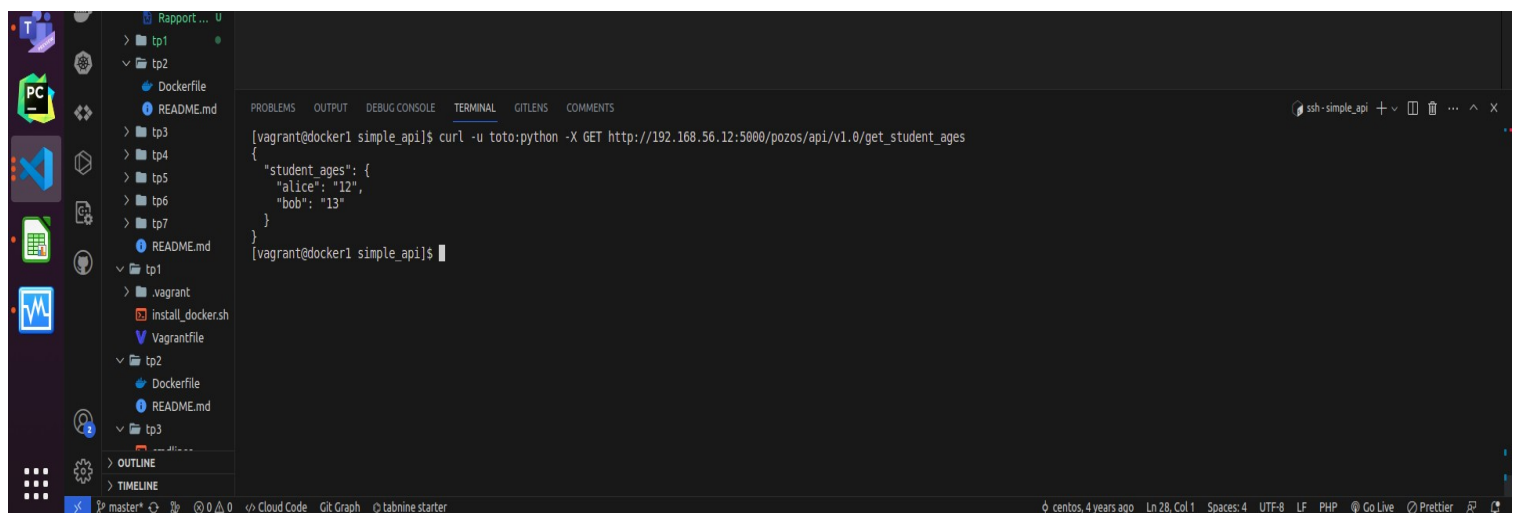
curl -u toto:python -X GET http://192.168.56.12:5000/pozos/api/v1.0/get_student_ages



```
.bash_history
.bash_logout
.bash_profile
.bashrc
.vbox_version
get-docker.sh

[vscode@docker1 simple_api]$ docker run -p 5000:5000 -v ${PWD}/student_age.json:/data/student_age.json api-image:1.0
* 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: 188-279-640
192.168.56.12 - - [11/Jul/2023 18:27:21] "GET /pozos/api/v1.0/get_student_ages HTTP/1.1" 200 -
192.168.56.12 - - [11/Jul/2023 18:27:31] "GET /pozos/api/v1.0/get_student_ages HTTP/1.1" 200 -
```

Ici, on peut remarquer que l’api a reçu des requêtes venant de l’hôte 192.168.56.12 (Même si cela aurait pu aussi marcher avec une VM ayant une autre ip).



```
Rapport... U
> tp1
> tp2
  Dockerfile
  README.md
> tp3
> tp4
> tp5
> tp6
> tp7
  README.md
> tp1
  .vagrant
  install_docker.sh
  Vagrantfile
> tp2
  Dockerfile
  README.md
> tp3

[vscode@docker1 simple_api]$ curl -u toto:python -X GET http://192.168.56.12:5000/pozos/api/v1.0/get_student_ages
{
  "student_ages": {
    "alice": "12",
    "bob": "13"
  }
}
[vscode@docker1 simple_api]$
```

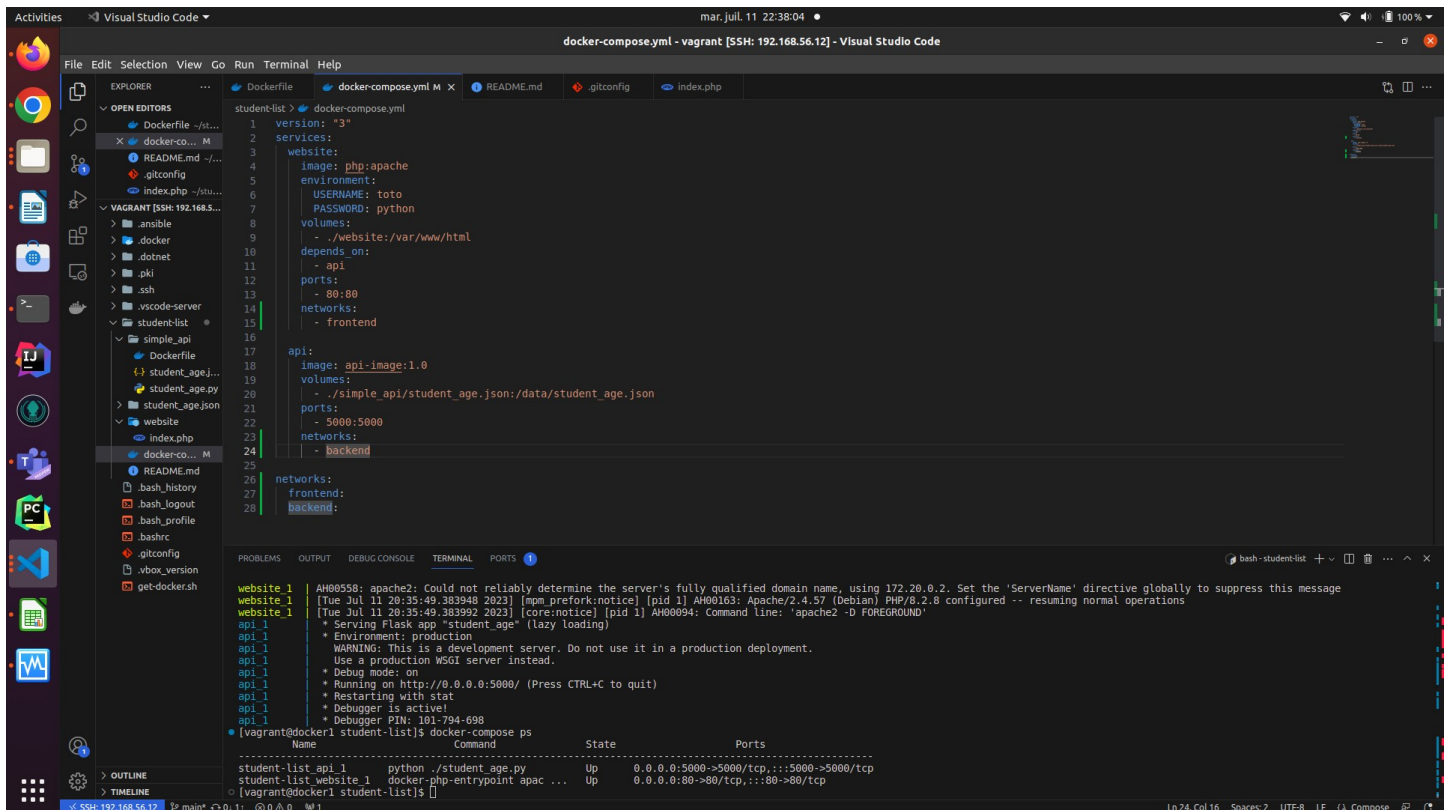
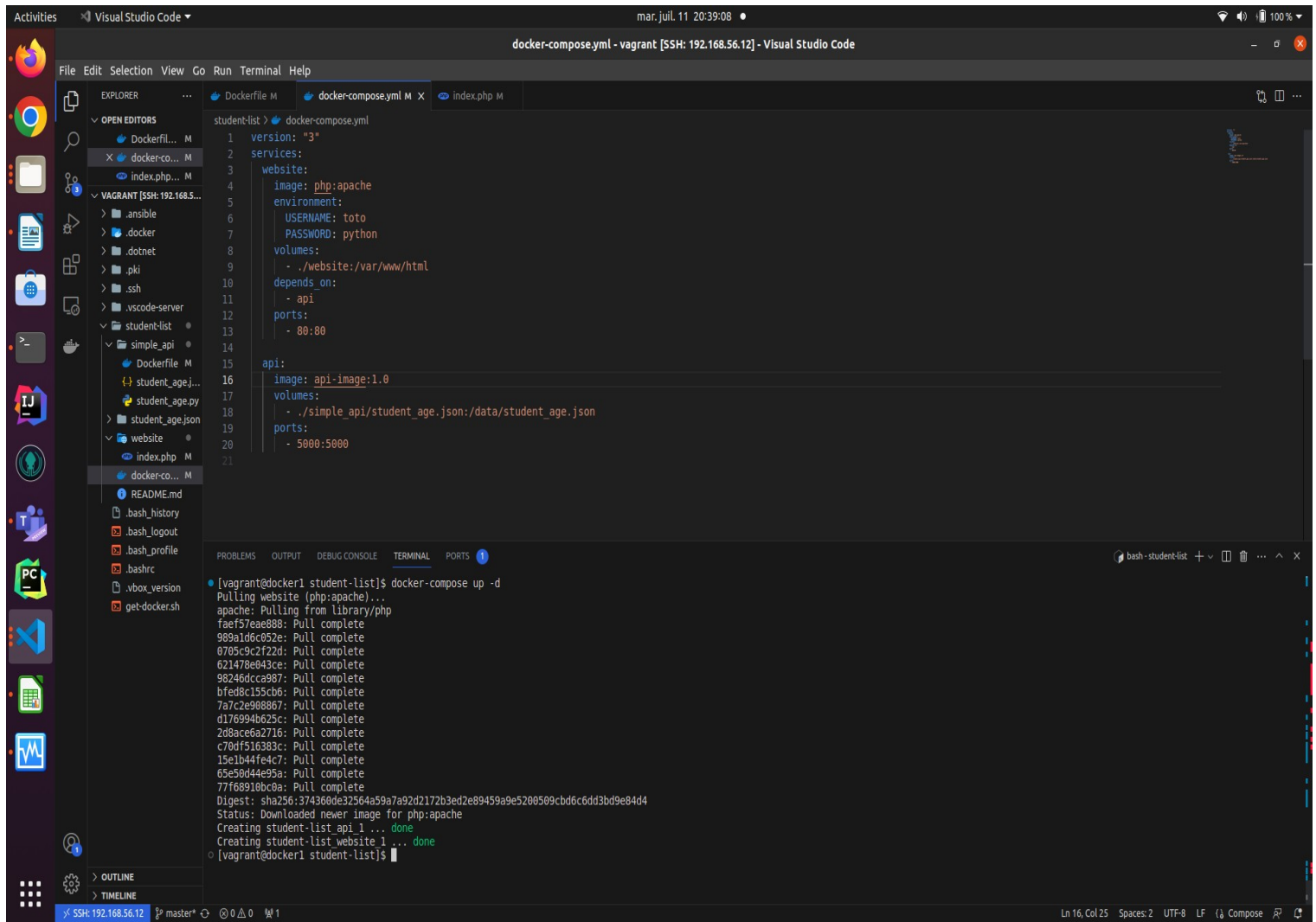
Ici cest la reponse venant de notre base de données : **student_age.json**

5 – Le docker-compose.yml

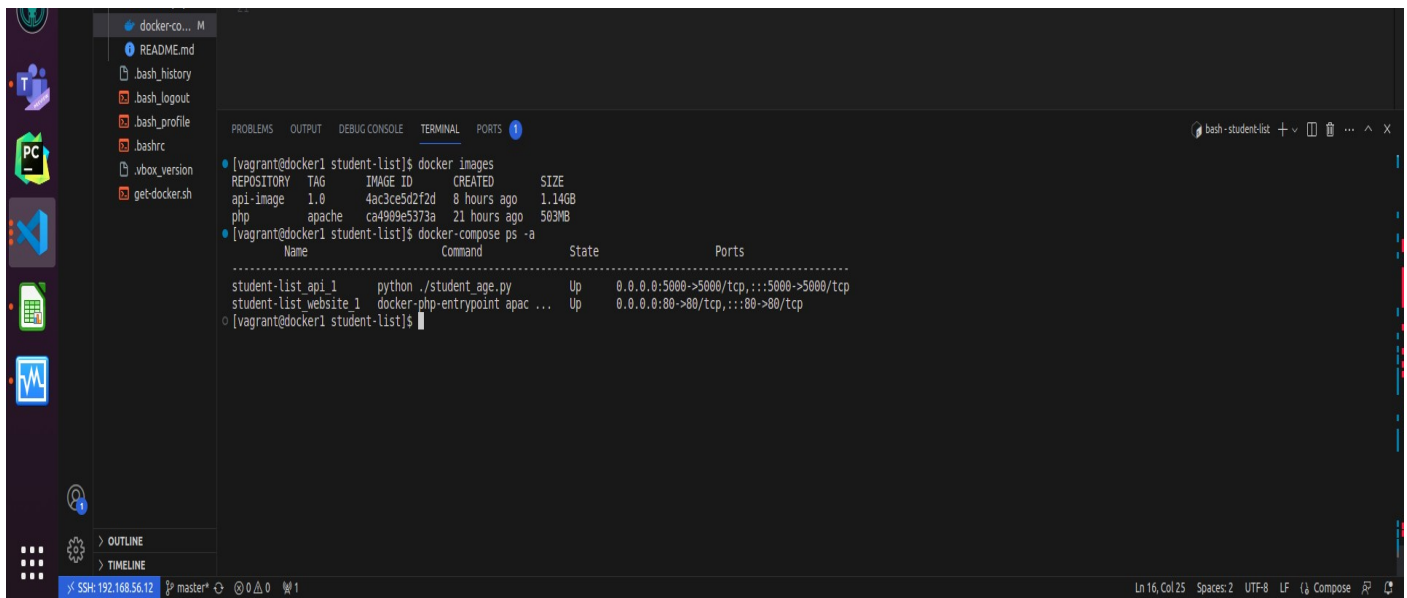
Après avoir supprimé le conteneur de l’image **api-image**, on crée le fichier **docker-compose.yml**.

Voici le contenu et le lancement :

Lancement: **docker-compose up -d**



- les differents services :



The screenshot shows a terminal window with the following commands and output:

```
[vagrant@docker1 student-list]$ docker images
```

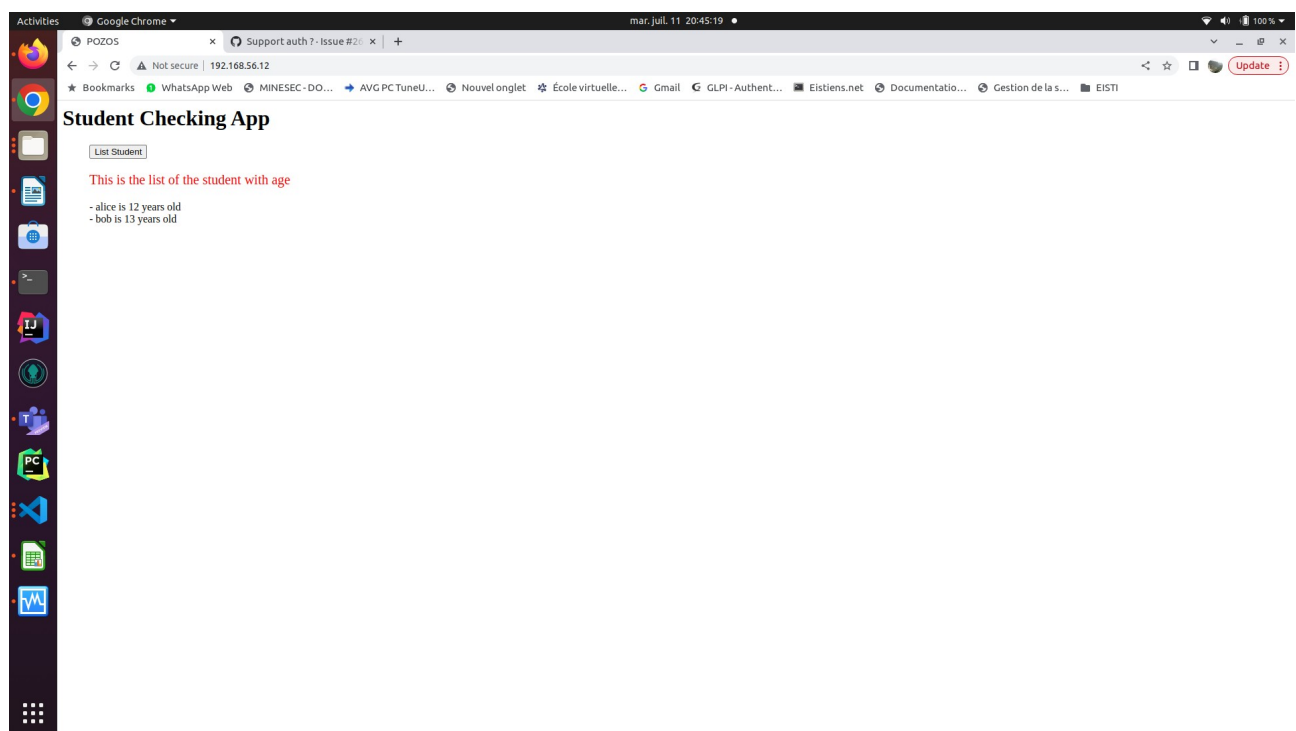
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
api-image	1.0	4ac3ce502f2d	8 hours ago	1.14GB
php	apache	ca4909e5373a	21 hours ago	503MB

```
[vagrant@docker1 student-list]$ docker-compose ps -a
```

Name	Command	State	Ports
student-list api_1	python ./student_age.py	Up	0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
student-list website_1	docker-php-entrypoint apac ...	Up	0.0.0.0:80->80/tcp, :::80->80/tcp

```
[vagrant@docker1 student-list]$
```

Test sur le navigateur: 192.168.56.12:80



Les sources sont disponibles ici : <https://github.com/Rodriguez001/dockerLabs.git/Mini-projet>

6- Sauvegarde des images sur le registre privé

creation d'une private registry, on commence par le reseau qui va heberger notre registry

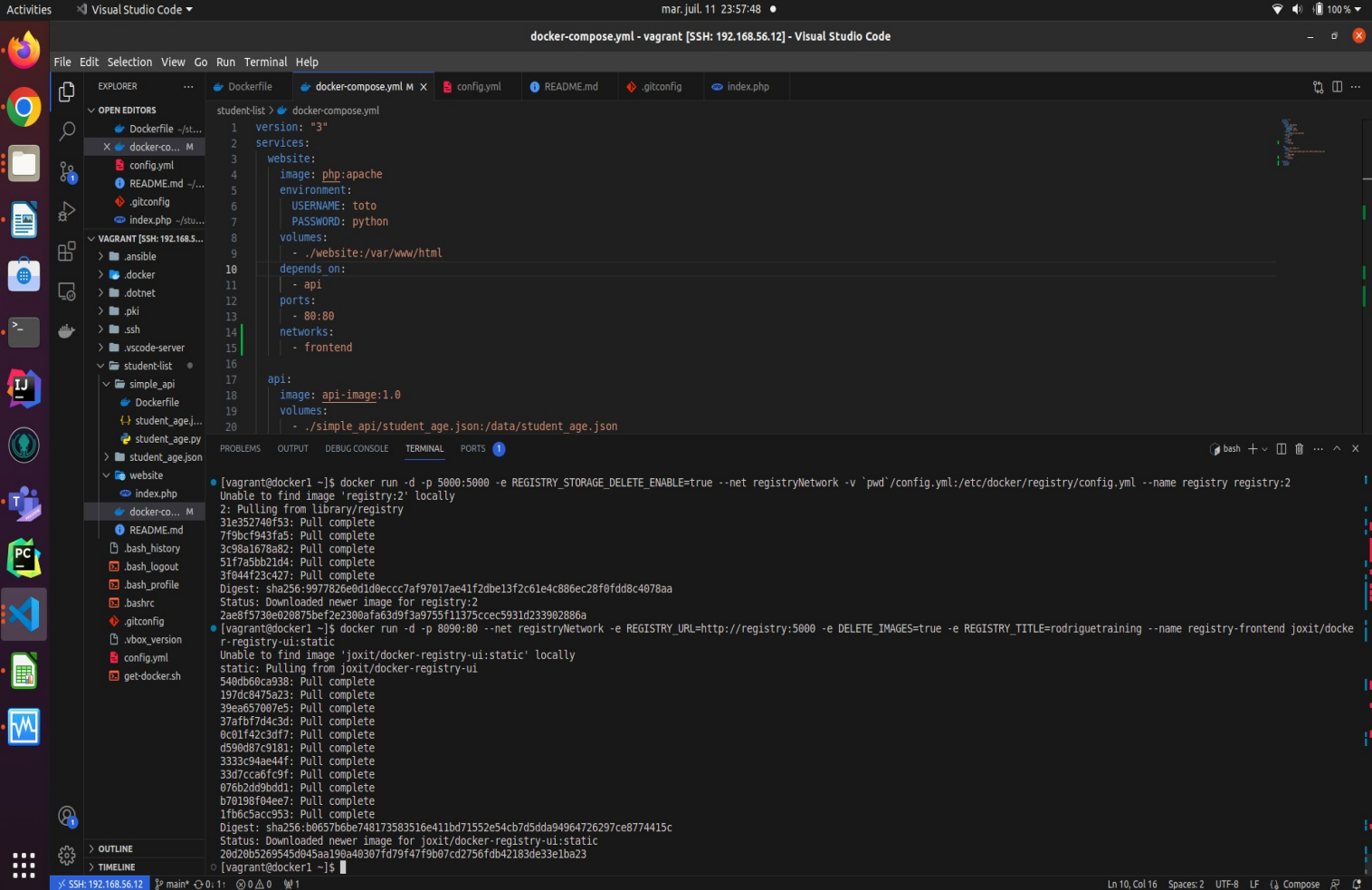
docker network create registryNetwork

a – Creation de l'image serveur (pour le registre privé) avec la commande:

docker run -d -p 5000:5000 -e REGISTRY_STORAGE_DELETE_ENABLE=true --net registryNetwork -v `pwd`/config.yml:/etc/docker/registry/config.yml --name registry registry:2

b – Pour le frontend qui va permettre la visualisation de nos images

docker run -d -p 8090:80 --net registryNetwork -e REGISTRY_URL=http://registry:5000 -e DELETE_IMAGES=true -e REGISTRY_TITLE=rodriguetraining --name registry-frontend joxit/docker-registry-ui:static



The screenshot shows the Visual Studio Code interface with a Docker Compose file open in the editor. The file is named `docker-compose.yml` and is located in a directory named `vagrant`. The file contains the following content:

```
1 version: "3"
2 services:
3   website:
4     image: php:apache
5     environment:
6       USERNAME: toto
7       PASSWORD: python
8     volumes:
9       - ./website:/var/www/html
10    depends on:
11      - api
12    ports:
13      - 80:80
14    networks:
15      - frontend
16
17  api:
18    image: api-image:1.0
19    volumes:
20      - ./simple_api/student_age.json:/data/student_age.json
```

The terminal output shows the following commands and results:

```
[vagrant@docker1 ~]$ docker run -d -p 5000:5000 -e REGISTRY_STORAGE_DELETE_ENABLE=true --net registryNetwork -v `pwd`/config.yml:/etc/docker/registry/config.yml --name registry registry:2
2: Pulling from library/registry
31e352740f53: Pull complete
7f9bcf943fa5: Pull complete
3c98a1678a82: Pull complete
51f7a5bb21d4: Pull complete
3f044f23c427: Pull complete
Digest: sha256:9977826e01d0eccc7af97017ae41f2db13f2c61e4c886ec28f0fdd8c4078aa
Status: Downloaded newer image for registry:2
2ae8f5730e020875be72e2300afa3d9f3a9755f11375ccec5931d233902886a
[vagrant@docker1 ~]$ docker run -d -p 8090:80 --net registryNetwork -e REGISTRY_URL=http://registry:5000 -e DELETE_IMAGES=true -e REGISTRY_TITLE=rodriguetraining --name registry-frontend 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
39ea65700e5: Pull complete
37afb7d4c3d: Pull complete
0c01f42c3df7: Pull complete
d590d87c9181: Pull complete
3333c94ae44f: Pull complete
33d7cca0f9f: Pull complete
076b2a090dd1: Pull complete
b70198f04ee7: Pull complete
1fb6c5acc953: Pull complete
Digest: sha256:b0657b6be748173583516e411bd71552e54cb7d5dda94964726297ce8774415c
Status: Downloaded newer image for joxit/docker-registry-ui:static
20d20b5269545d045aa190a40307fd79f47f9b07cd2756fdb42183de33e1ba23
[vagrant@docker1 ~]$
```

c- On va taguer nos images locales

- **docker tag ca4909e5373a localhost:5000/student-list-front:local**
- **docker tag 4ac3ce5d2f2d localhost:5000/student-list-back:local**

d- On fait le push sur le registre

- **docker push localhost:5000/student-list-front:local**
- **docker push localhost:5000/student-list-back:local**

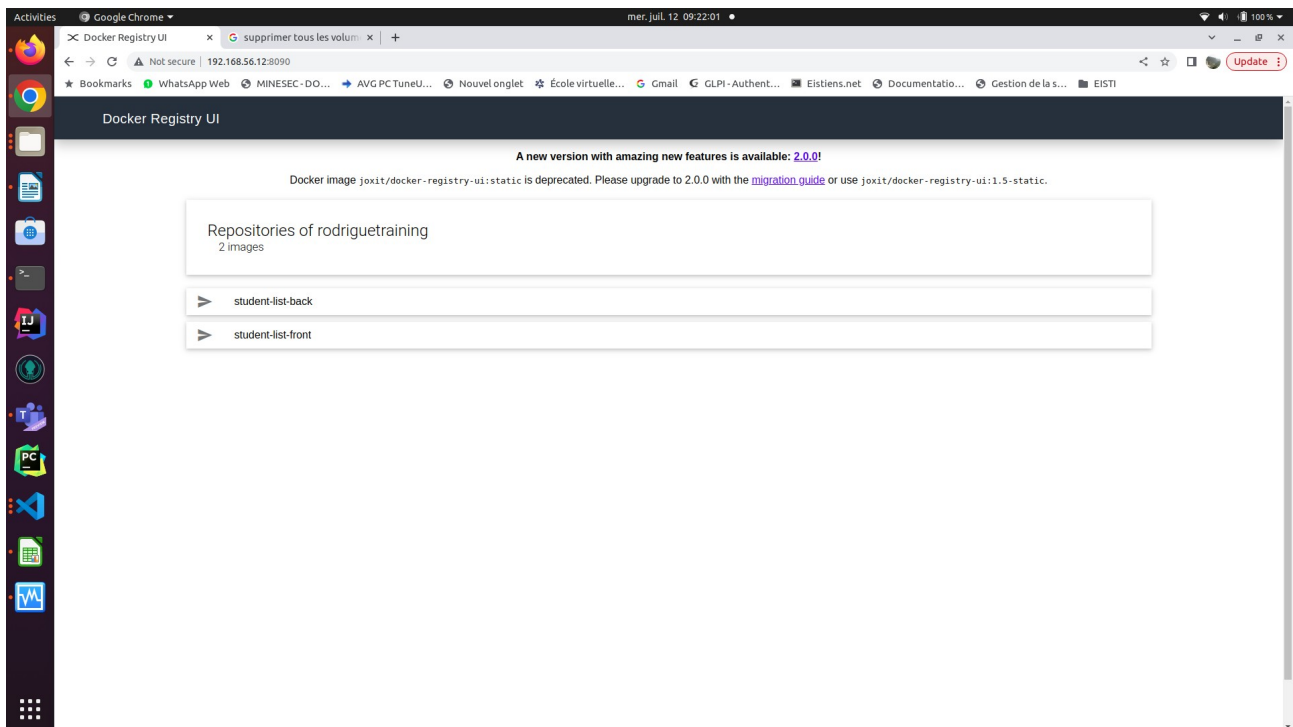
The screenshot shows the Visual Studio Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with files like Dockerfile, docker-compose.yml, config.yml, and README.md. The terminal displays the output of Docker commands, including tagging and pushing images to a local registry.

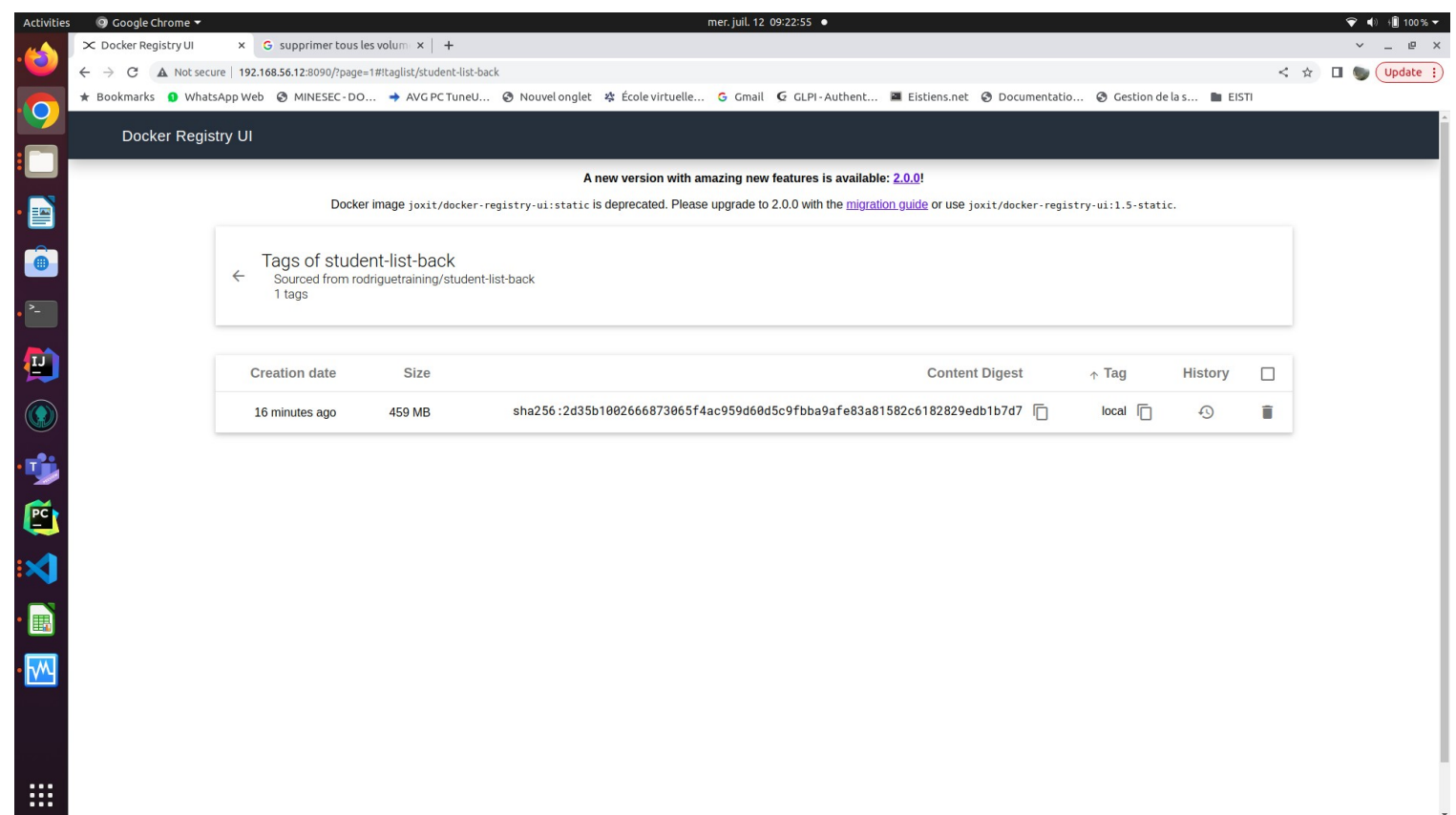
```
config.yml - vagrant [SSH: 192.168.56.12] - Visual Studio Code

EXPLORER
  student-list > config.yml
    19
    20 auth:
    21   httpasswd:
    22     realm: basic-realm
    23     path: /etc/registry
    24   health:
    25     storagedriver:
    26       enabled: true
    27       interval: 10s
    28       threshold: 3
    29   catalog:
    30     maxentries: 100000
    31

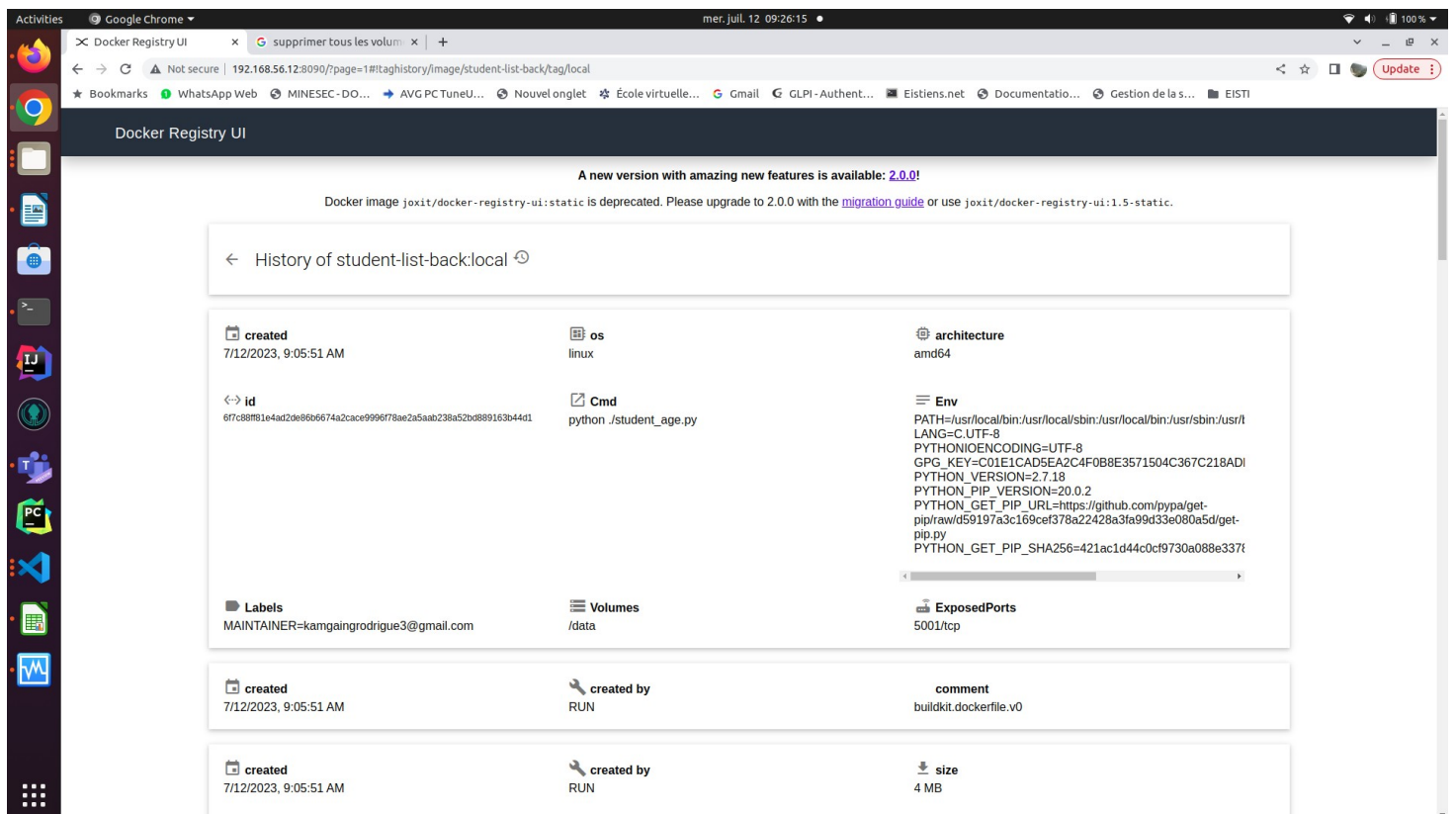
TERMINAL
  joxit/docker-registry-ui static c97caf4d3877 2 years ago 24.5MB
  [vagrant@docker1 student-list]$ docker tag ca4909e5373a localhost:5000/student-list-front:local
  [vagrant@docker1 student-list]$ docker tag 6f7c88ff81e4 localhost:5000/student-list-back:local
  [vagrant@docker1 student-list]$ docker push localhost:5000/student-list-front:local
  The push refers to repository [localhost:5000/student-list-front]
  937b931f3d83: Pushed
  3c6a08069f4e6: Pushed
  b33cdd1a9bfc: Pushed
  c41bbaf795ec: Pushed
  d25256087733: Pushed
  3641a610984b: Pushed
  c6c385ceab87: Pushed
  af169763ac2b: Pushed
  abf6306dfff5: Pushed
  1dfeaa791054: Pushed
  bd5b7b7ab346: Pushed
  90ba21fb5089: Pushed
  24839045ca45: Pushed
  local: digest: sha256:242ea18485e8a62a5adedc3f2ccca6f2415a801c66a8eead4ceec8742d192c2d size: 3036
  [vagrant@docker1 student-list]$ docker push localhost:5000/student-list-back:local
  The push refers to repository [localhost:5000/student-list-back]
  9488893dd052: Pushed
  e6522cab010e: Pushed
  761ec95c9a6c: Pushed
  e571d2d3c73c: Pushed
  da7b6a80a4f2: Pushed
  ceee8816bb96: Pushed
  47458fb45d99: Pushed
  46029331b1e4: Pushing ] 64.69MB/509.8MB
  d95c5bda4793: Pushing ] 125.1MB/145.5MB
  a3c1826c0bcc: Pushed
  f1d420c2af1a: Pushed
  461719022993: Pushing [=====] 92.9MB/114.1MB
```

Confirmation en ligne: 192.168.56.12:8090





L'historique de student-list-back



Remarque: Avant affichage, il faut résoudre un petit bug

- Se connecter au conteneur du registre privé:

docker exec -it [nom_du_registre]

- Ensuite modifier le fichier **etc/docker/registry/config.yml** en y ajoutant les lignes suivantes

```
catalog:
```

```
  maxentries: 100000
```