**SPECIALITE : DEVOPS**

**Compte Rendu**

**PROJET DOCKER**

**Préparé par**: JEAN THIBAUT

**Formateur**: TAFEN

**PROJET DOCKER**

**L’objectif de ce mini-projet docker est de nous permettre d’implémenter tout ce que nous avons appris sur l’écosystème micro-service, conteneurisation, docker-compose.**

**C’est un projet très intéressant car issu des besoins d’entreprise.**

**On nous demande dans ce projet de conteneuriser une application écrite en python, et ensuite de la déployer sur docker avec un frontend en PHP et un backend qui est fait en python et qui tourne sur flask.**

**Dans un premier temps on nous demande de Builder et tester l’appli (en écrivant le DOCKERFILE).**

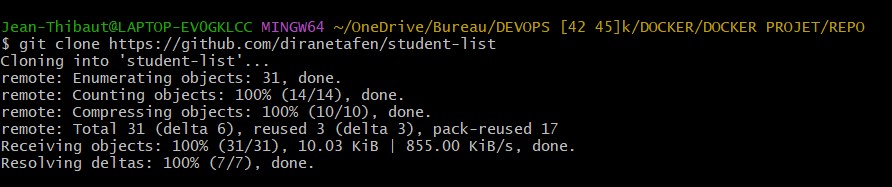
**Ensuite après avoir écrit le DOCKERFILE pour conteneuriser l’application, on va tester l’appli avec un Curl.**

**Ensuite on va mettre en place la partie Infrastructure As Code pour de déploiement facile de notre application et la reproduire ce déploiement sur diverses autres infrastructures. Dans ce cas on va modifier un certain nombre de fichiers.**

**Par la suite on va créer un docker registry qui va nous permettre d’héberger l’image du client que nous avons produite en occurrence l’image de student list.**

**CLONER LE REPO DISTANT OU SE TROUVE LE PROJET**

**$ git clone** **<https://github.com/diranetafen/student-list>**

****

**IHM : interface homme-machine**

**PARTIE 1 : BUILD / TEST.**

**Tout va partir d’une machine qui contient docker, nous avons une application qui est un script développé en flask et stockée sur un GITHUB.**

**Du coup on devra télécharger l’application en local, par la suite on pourra créer un dockerfile qui lui va Builder une image pour pouvoir instancier un container de cette application (student-list).**

**PARTIE 2 : INFRASTRUCTURE AS CODE.**

**Ici nous allons écrire un docker-compose qui va nous permettre d’automatiser la création de l’application.**

**A cela rajouter une deuxième appli qui sera l’IHM (interface HOMME-MACHINE) qui va taper le container student-list afin de pouvoir visualiser le rendu via un navigateur.**

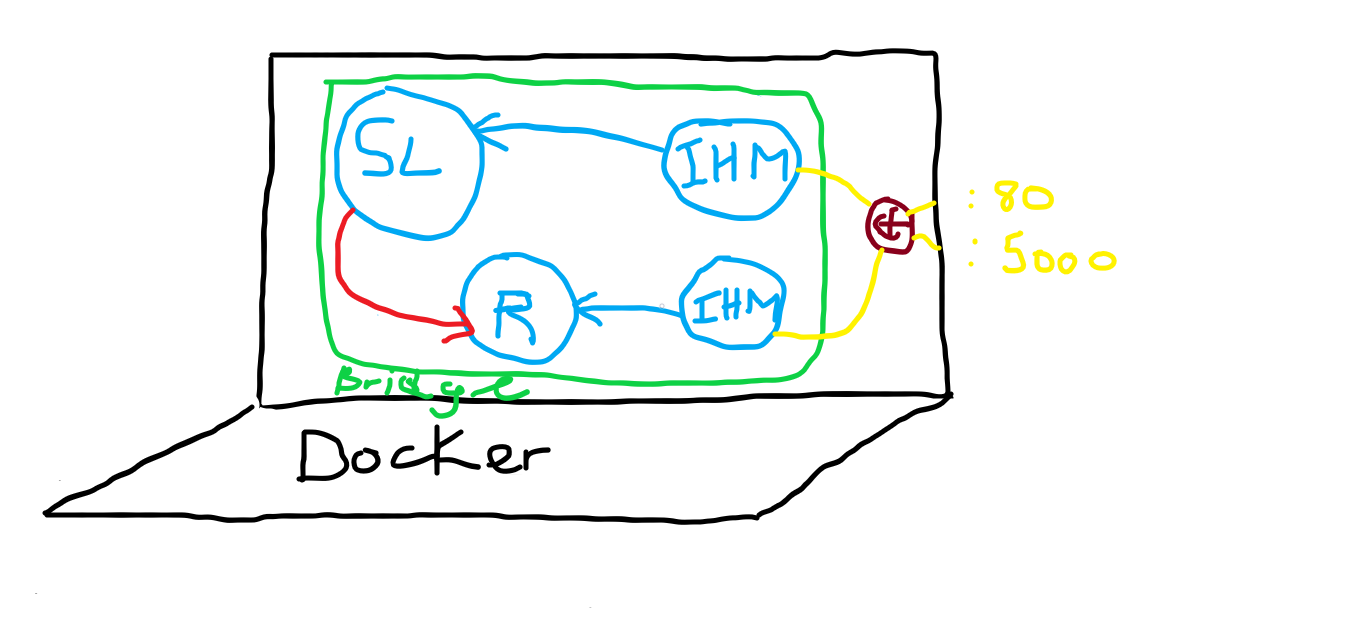
**PARTIE 3 : DEPLOYER UN REGISTRE PRIVE.**

**Déployer un 3ème container qui aura pour rôle qui aura pour rôle d’être un registre privé pour notre image que nous aurons Buildé.**

**Pareillement pour pouvoir tester cela on nous demande de rajouter une IHM pour le registre qui elle aura pour rôle de venir taper le registre afin qu’on le consulter via un navigateur.**

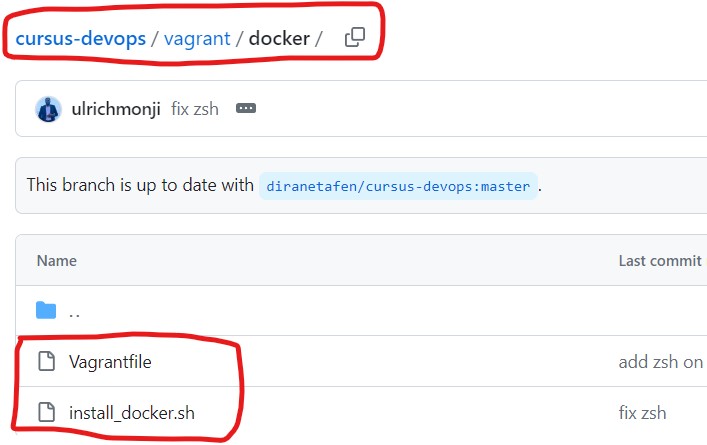
**Logiquement pourque tous ces containers puissent interagir il faudrait qu’ils soient tous dans un même réseau docker. Nous on va partir sur un réseau de type BRIDGE.**

**Pour exposer nos applications vers l’extérieur, il faudra mettre des règles d’expositions. Nos applications doivent être accessibles via l’IHM du coup nous allons exposer que nos containers IHM**

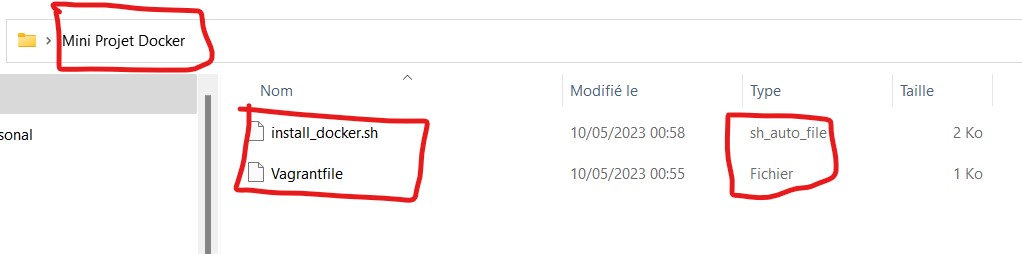
****

**1ère étape : Nous allons déployer notre machine qui contient docker.**

**Nous allons travailler sur VirtualBox et utiliser vagrant pour le déploiement.**

****

**Par la suite nous allons copier ces deux fichiers avec les extensions qui vont avec dans le répertoire de notre projet**

****

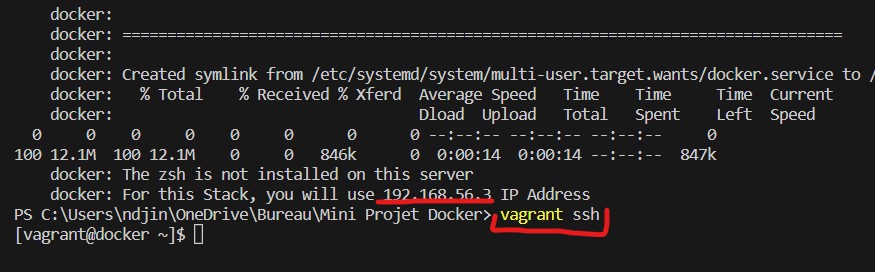
**Et par la suite nous allons l’ouvrir avec Visual studio code**

**Une fois dans visual studio code on fait :**

**Vagrant up –-provision : cette commande va nous permettre de créer notre machine docker dans VirtualBox.**

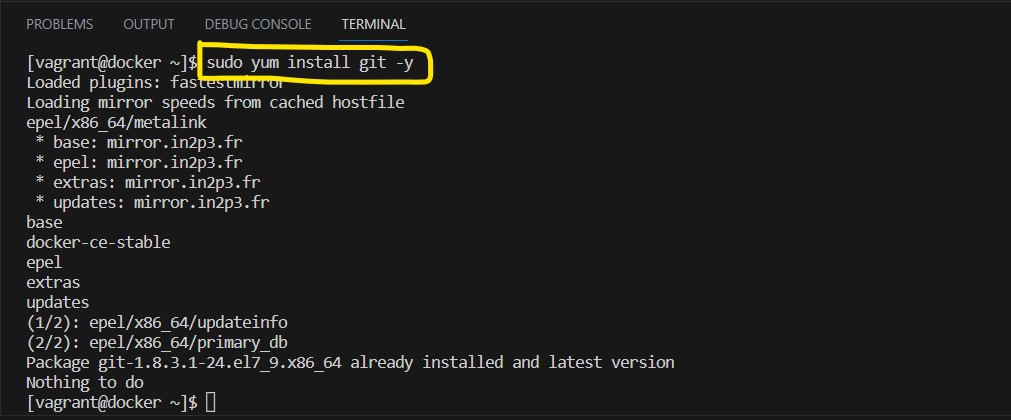
**Une fois la machine créée on fait : uptime pour avoir les infos sur la machine créée.**

**Ensuite pour pouvoir se connecter on fait un Vagrant ssh**

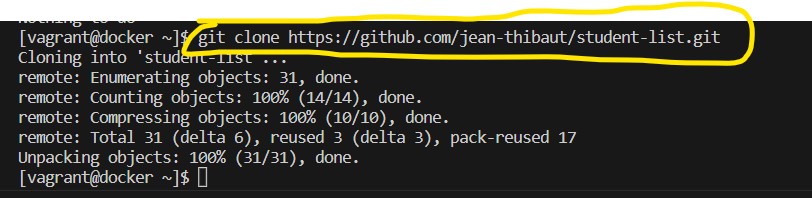
****

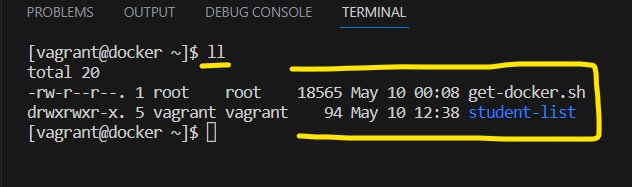
**A cette étape nous allons télécharger le repo GIT du projet dans notre machine DOCKER déployée. Mais avant il faut installer GIT.**

**SUDO YUM INSTALL GIT -Y**

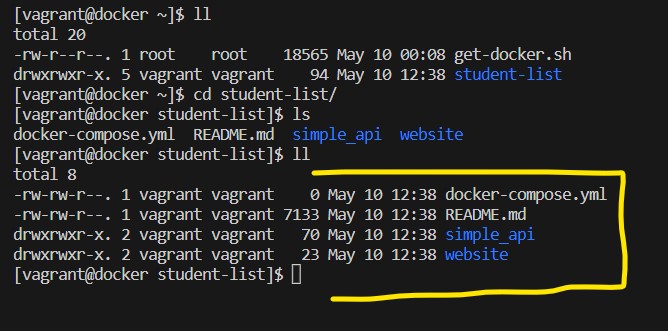
****

**Une fois GIT installé on peut faire un GIT CLONE (link\_repo)**

****

****

**On va se déplacer dans notre répertoire de travail student-list**

****

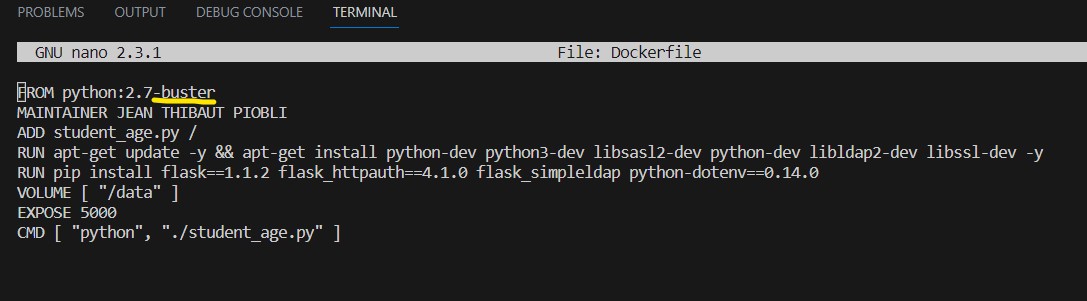
**Nous avons à cette étape mis en place notre environnement de travail en déployant notre machine docker avec Vagrant, une fois cela fait nous nous sommes connectés à notre machine docker (vagrant ssh). Une fois dans notre machine docker, nous avons installé GIT (sudo yum Install GIT -y) pour par la suite faire un :**

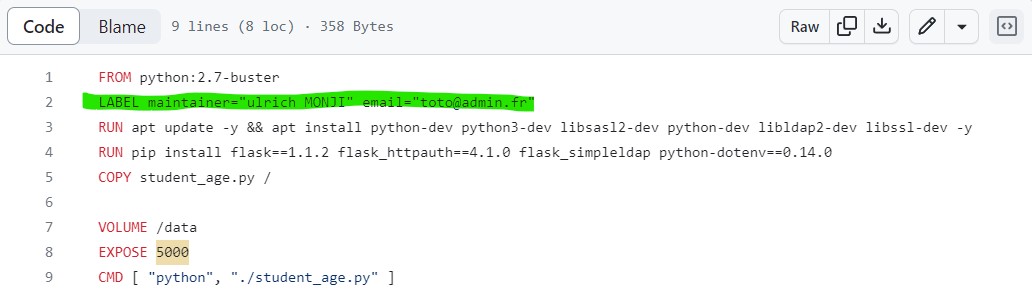
**Git clone (**[**https://github.com/diranetafen/student-list**](https://github.com/diranetafen/student-list)**) pour récupérer le repo du projet.**

**A ce niveau nous allons donc commencer la construction du Dockerfile.**

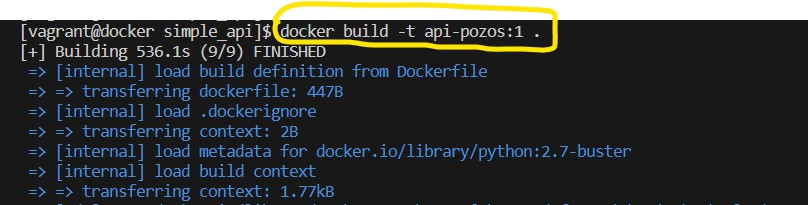
**En suivant les instructions données dans l’énoncé nous avons construit le dockerfile ci-dessous.**

**YUM est le gestionnaire de package de CentOS**

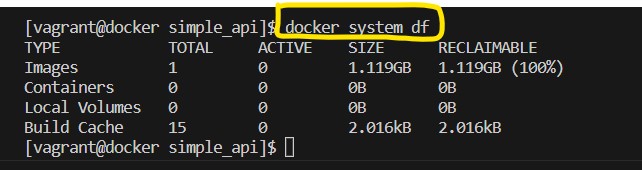
****

****

**Et donc après avoir construit le dockerfile on fait un DOCKER BUILD -T API-POZOS:1 .**

****

**DOCKER SYSTEM DF**

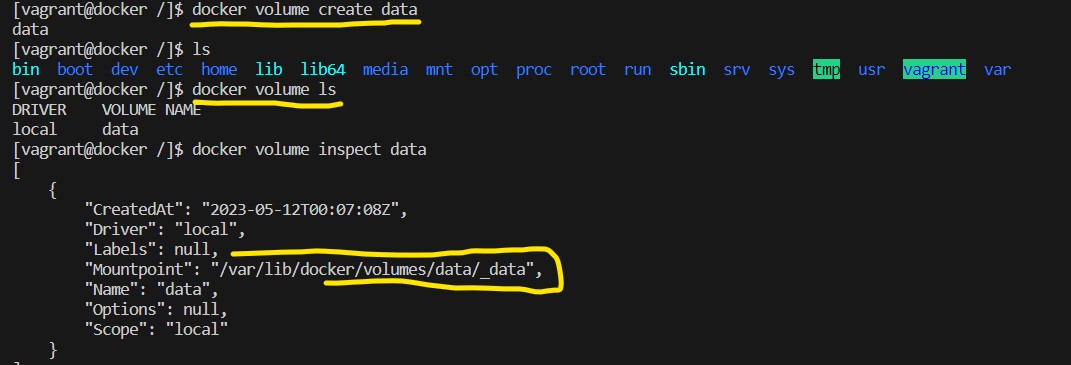
****

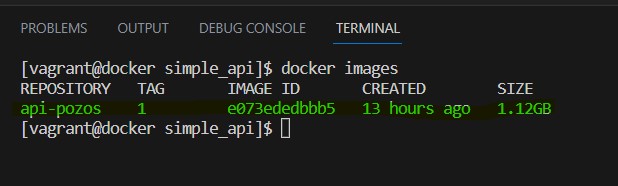
**On peut remarquer dans la capture que notre Build n’a pas pris en compte le VOLUME que nous avons déclaré dans le dockerfile.**

**Il nous était demandé de créer un fichier data à la racine de notre machine docker et la considérer comme volume car c’est la que vont être stockés les données de notre application (student\_age.json)**

**Face à cette situation, nous avons donc de nous même créé ce volume dans la racine.**

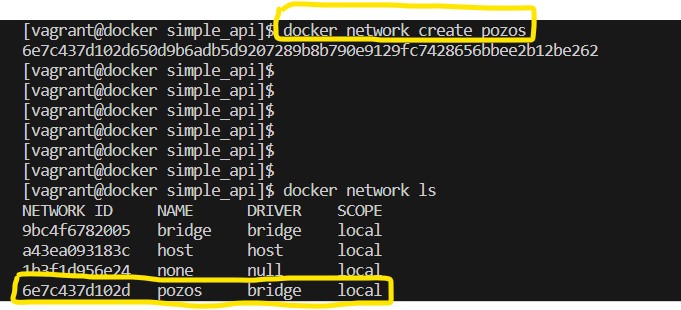
**DOCKER VOLUME CREATE DATA**

****

****

**Maintenant que nous avons notre image il faut la tester mais avant il faut faire un run qui lancer un container.**

**Pour tester nous allons nous créer un réseau, nous allons déjà commencer par travailler dans un réseau. Créons un réseau pozos.**

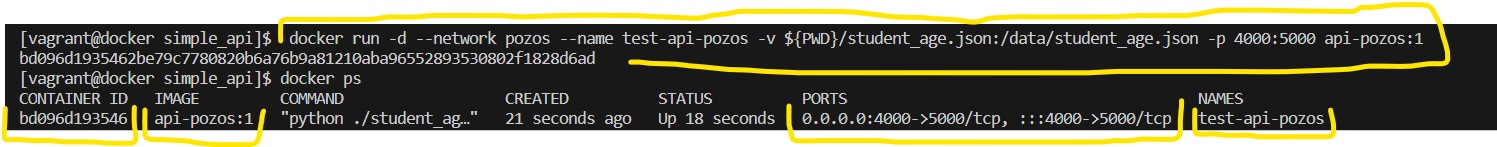
****

**Avec la commande ci-dessous nous allons pouvoir déployer le container.**

**Docker run -d –-network pozos –-name test-api-pozos**

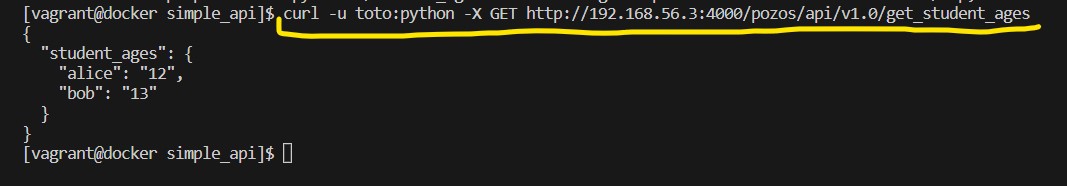
**-v ${PWD}/student-age.json:/data/student-age.json**

**-p 4000:5000 api-pozos:1**

****

**Une fois le container qui tourne, il faut le tester.**

**Curl -u toto:python -X GET http://<host IP>:<API exposed port>/pozos/api/v1.0/get\_student\_ages**

****

**On a notre application qui nous répond, le script**

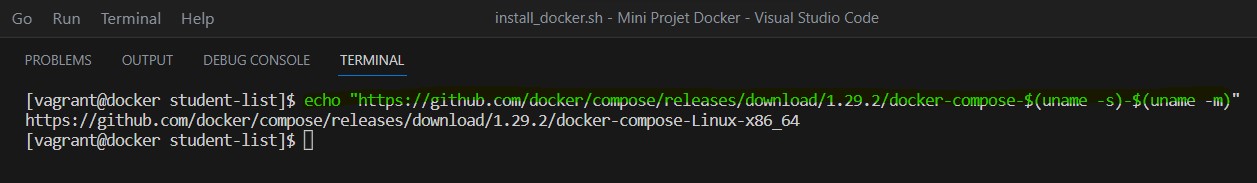
**student-age.json**

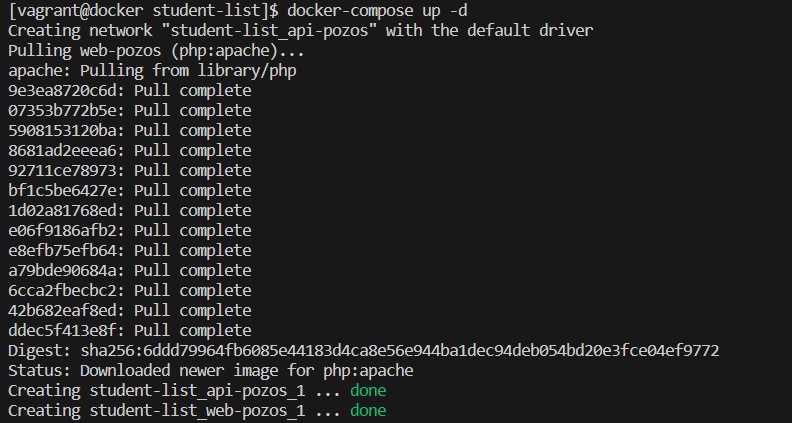
**PARTIE 2 : INFRASTRUCTURE AS CODE**

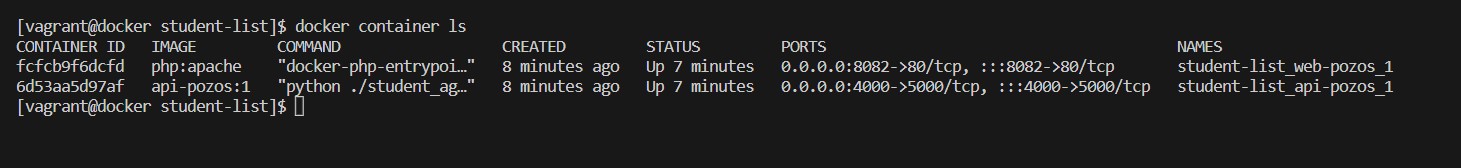
**Il est question dans cette partie de pouvoir mettre en place un docker-compose qui va nous permettre de pouvoir automatiser le déploiement de notre infrastructure.**

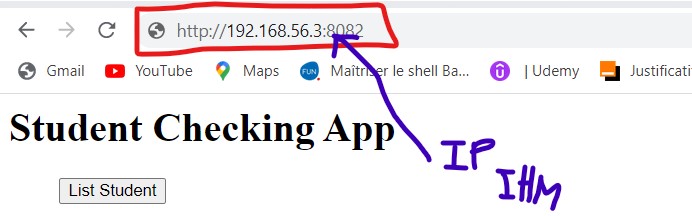
****

**DOCKER-COMPOSE UP -D**

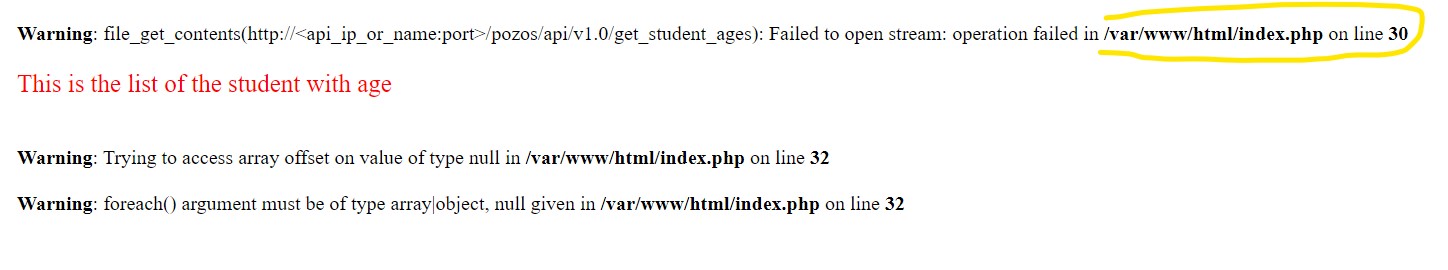
****

****

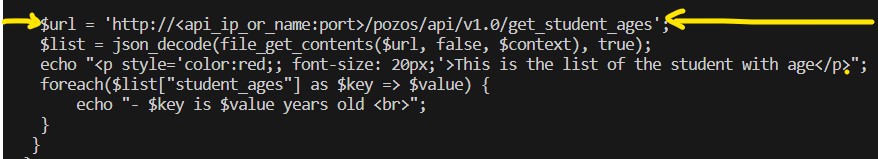
****

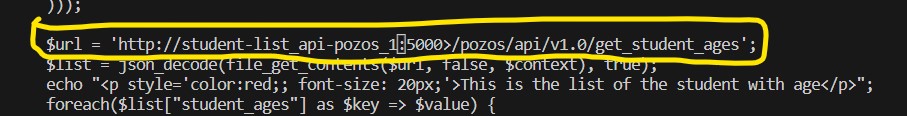
****

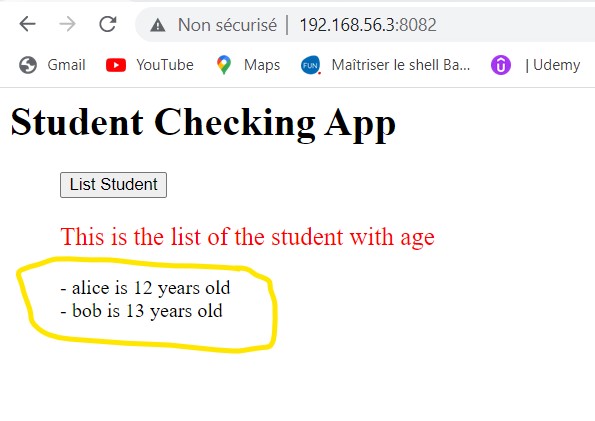
**Notre IHM écoutes sur le port :8082 de l’extérieur redirigé vers le port 80 de notre container.**

****

**NANO WEBSITE/INDEX.PHP**







**PARTIE 3 : DOCKER REGISTRY**

**Commençons par déployer le registre**