

ECOLE SUPERIEURE D'INFORMATIQUE SALAMA  
République Démocratique Du Congo  
Province du Haut-Katanga  
Lubumbashi  
[www.esisalama.org](http://www.esisalama.org)

---



---

## TRAVAIL PRATIQUE DU COURS DE DEVOPS

---

RAPPORT : PROJET SUR LE FLASK, DOCKERFILE, JENKINS

Par : **KABUYA KAZADI Gaël**  
**LUKENDO SAMUEL SALUT**

Demandé par : **Prof Blaise ANGOMA**  
Filière : **M2 Réseaux**

**JUIN 2023**

## 1. INTRODUCTION

Ce rapport explique en détails notre projet en se basant aux séances des cours suivie en classe. Nous définissons quelques termes tels que :

- **FLASK** : Etant appelée un micro Framework open-source car il est très léger pour le développement web en python. Il a comme un objectif de garder un noyau simple mais extensible.
- **DOCKERFILE** : Etant un simple fichier texte, contenant les instructions nécessaire afin de construire un (BUILD) une image docker. Les instructions décrivent que les actions que l'image doit s'exécuter une fois quelle est créer.
- **JENKINS** : Etant un serveur d'automatisation gratuit et open source. Il aide à automatiser les parties du développement logiciel liées à la construction, aux tests et au déploiement. Il s'agit d'un système basé sur un serveur qui s'exécute dans des conteneurs de servlet.

## 2. VOICI EN QUELQUES SORTES LES DETAILS DE LA CREATION DE L'APPLICATION :

Nous avons mis en place une application de gestion des étudiants du Master ESIS, en se basant sur les questionnaires :

- a. Notre application contient deux ressources :
  - **la première ressource (Etudiant)** : L'application affiche sous forme d'un tableau les informations de l'étudiant :
    - Nom
    - Post nom
    - Prénom
    - Age
    - Genre
    - Promotion
  - **La deuxième ressource (Cours)** : L'application affiche les informations sur les cours concernant l'étudiant :
    - L'intitulé du cours
    - Pondération
    - Crédit
    - Titulaire

## ❖ Pour l'étudiant

- cette ressource nous a permis à obtenir tous les étudiants :

```
app.route('/etudiants', methods=['GET'])
```

- cette ressource nous permis à obtenir un étudiant spécifique :

```
app.route('/etudiants/<int:etudiant_id>', methods=['GET'])
```

- cette ressource nous permis à ajouter un étudiant :

```
app.route('/etudiants', methods=['POST'])
```

## ❖ pour les cours

- cette ressource nous a permis à ajouter un cours :

```
app.route('/cours', methods=['POST'])
```

- cette commande nous a permis de faire le lancement de l'application

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=9000, debug=True)
```

- b. Pour créer un fichier Dockerfile qui expose qui expose le service au port 9000, nous somme passer par ces 3 étapes qui suivent :

- **Pour la première étape avec la commande (FROM)**, nous avons utilisé l'image Node.js en tant qu'image de base (FROM node : 14). Ensuite il était question de copier de copier les fichiers de l'application dans le conteneur avec cette commande (**COPY. /app**) enfin de définir le répertoire travail a /app (**WORKDIR /app**).
- **Pour la deuxième étape avec la commande (RUN)**, il était question d'installer les dépendances de l'application à l'aide de commande **RUN npm Install** pour nous permettre ensuite d'exposer l'application au port 9000 avec la commande **EXPOSE 9000**.
- **Pour la troisième étape avec la commande (CMD)**, elle nous a aidé a démarrée l'application en utilisant la commande **CMD ["npm", "Start"]**, pour exécuter la commande "npm Start" en fin de lancer l'application.

- c. Pour créer un fichier Jenkinsfile qui génère un conteneur docker et l'exécute, nous avons défini deux étapes dont : "Build docker" et "RUN docker container"

- Pour la première étape avec "Build Docker image" qui construit l'image docker en utilisant le nom de l'image spécifié et l'ID de construction généré par Jenkins.
- Pour la deuxième étape avec "Run Docker" exécute le conteneur Docker en utilisant l'image Docker construite à l'étape précédente. Il expose le port 9000 et donne le nom au conteneur.
- La section "post" contient des instructions qui seront exécutées après la fin de toutes les étapes, quel que soit leur résultat. Nous avons ajouté des instructions pour arrêter et supprimer l'image Docker.

Voici quelques images du test de notre application flask en local :

```

C:\WINDOWS\system32\cmd.exe - docker run -p 9000:80 monappflask

G:\Cours\Master 2\DevOps\Projet\test>docker build -t monappflask .
[+] Building 18.5s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 222B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.9-slim-buster
=> [1/5] FROM docker.io/library/python:3.9-slim-buster@sha256:320a7a4250aba4249f458872adecf92eea88dc6abd2d76dc5c0f01cac9b53990
=> [internal] load build context
=> => transferring context: 494.00kB
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY . /app
=> [4/5] RUN pip3 install --no-cache-dir -r requirements.txt
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:a5830bb1579c6efc44aa6de70652099dda91422f139ae25fa5c8f49225208fc
=> => naming to docker.io/library/monappflask

G:\Cours\Master 2\DevOps\Projet\test>docker run -p 9000:80 monappflask
* Serving Flask app 'appetudiant'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:9000
* Running on http://172.17.0.2:9000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 139-866-778
  
```

Figure 1 Compilation de l'application avec l'invite de commande

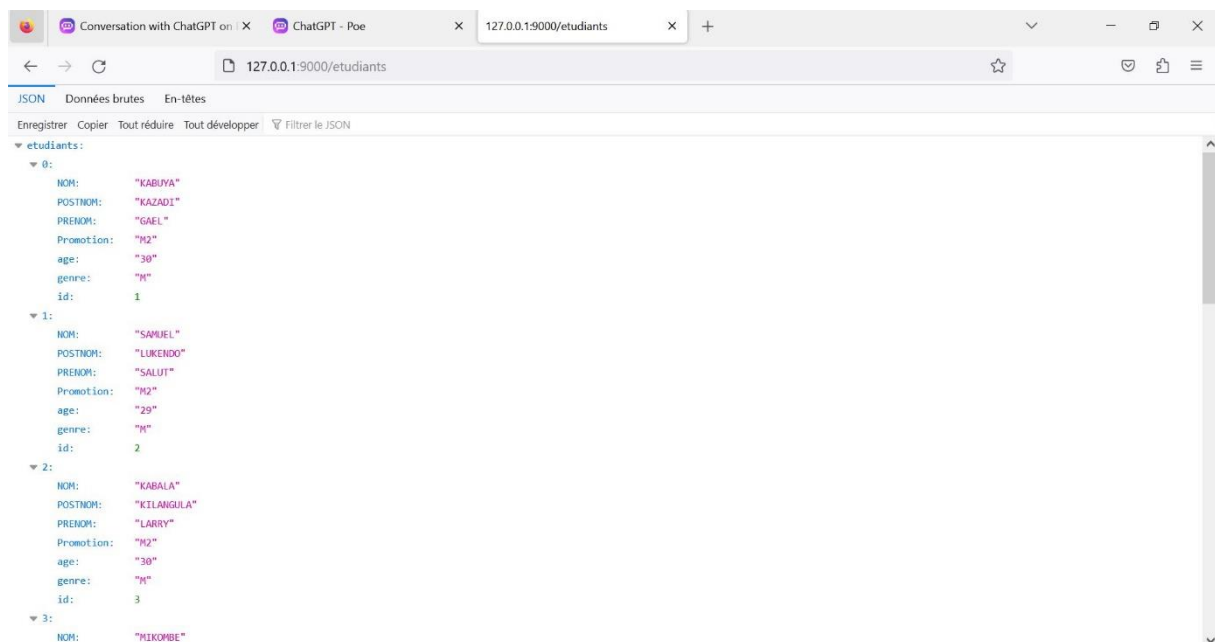
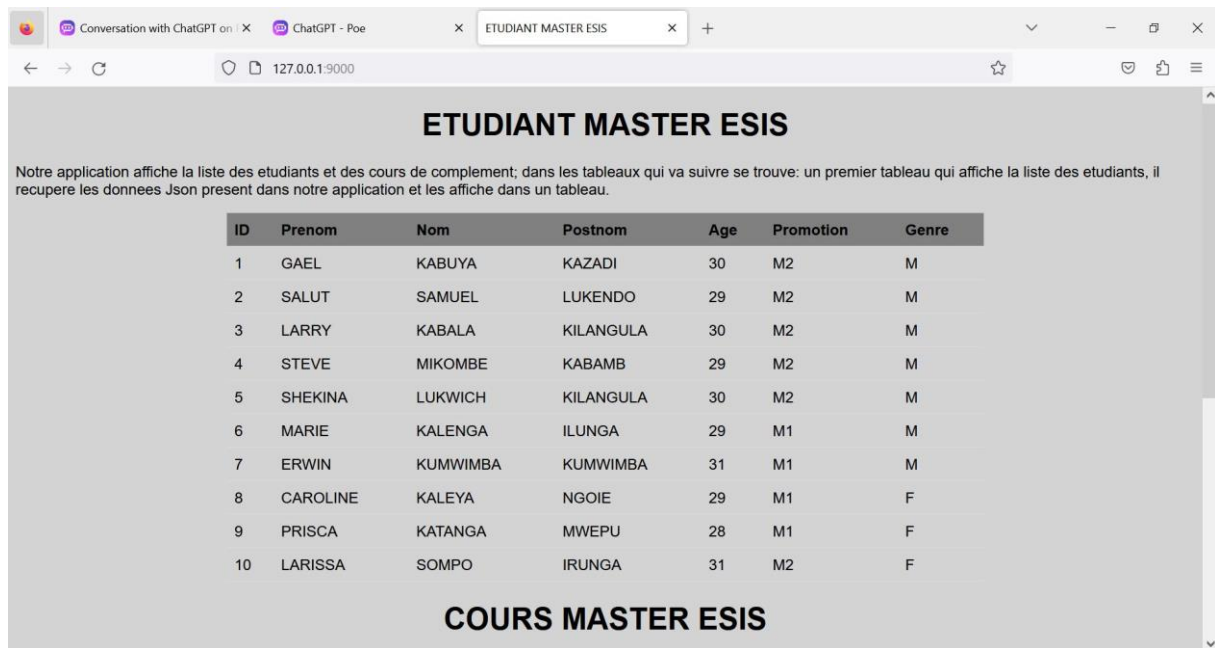


Figure 2 L'API permet d'accéder à la liste JSON



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:9000'. The page title is 'ETUDIANT MASTER ESI'. Below the title, there is a paragraph of text in French: 'Notre application affiche la liste des etudiants et des cours de complement; dans les tableaux qui va suivre se trouve: un premier tableau qui affiche la liste des etudiants, il recupere les donnees Json present dans notre application et les affiche dans un tableau.' Below this text is a table with 7 columns: ID, Prenom, Nom, Postnom, Age, Promotion, and Genre. The table contains 10 rows of student data. Below the table, there is a section titled 'COURS MASTER ESI'.

ID	Prenom	Nom	Postnom	Age	Promotion	Genre
1	GAEL	KABUYA	KAZADI	30	M2	M
2	SALUT	SAMUEL	LUKENDO	29	M2	M
3	LARRY	KABALA	KILANGULA	30	M2	M
4	STEVE	MIKOMBE	KABAMB	29	M2	M
5	SHEKINA	LUKWICH	KILANGULA	30	M2	M
6	MARIE	KALENGA	ILUNGA	29	M1	M
7	ERWIN	KUMWIMBA	KUMWIMBA	31	M1	M
8	CAROLINE	KALEYA	NGOIE	29	M1	F
9	PRISCA	KATANGA	MWEPU	28	M1	F
10	LARISSA	SOMPO	IRUNGA	31	M2	F

**COURS MASTER ESI**

Figure 3 Interface de l'application web

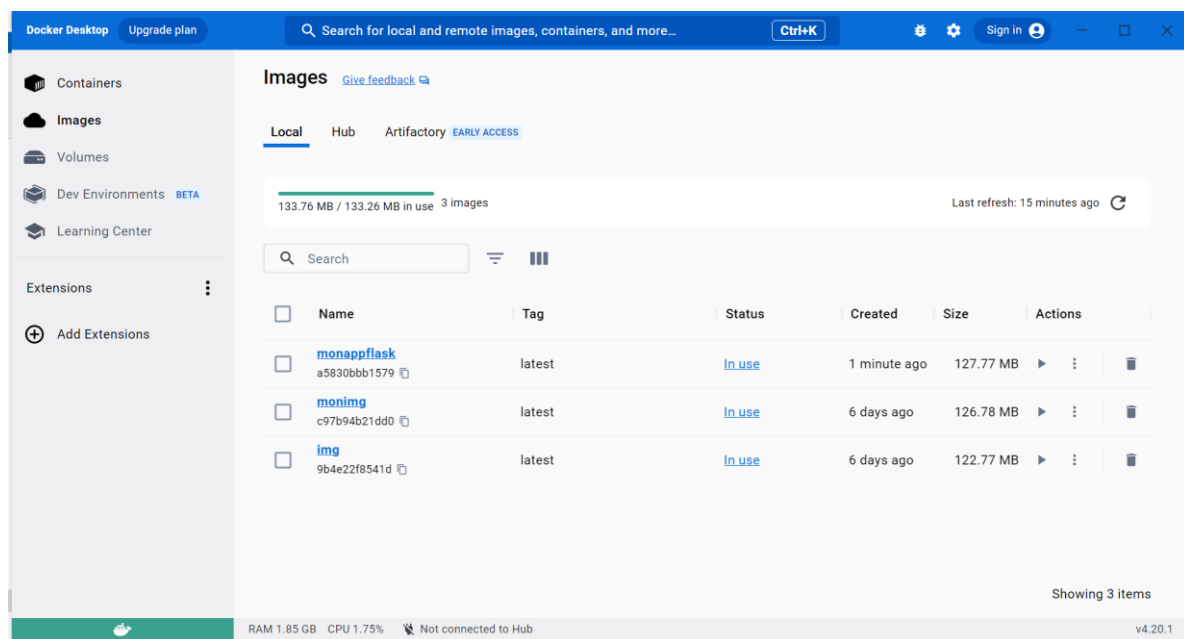


Figure 4 Docker Desktop