



## **Projet de Virtualisation**

# Sommaire

<b>Introduction</b>	<b>2</b>
<b>Architecture &amp; son évolution</b>	<b>3</b>
1. Dockerfile	3
2. Dossier config	4
3. Scripts	5
<b>Rendu final</b>	<b>6</b>
1. Login	6
2. Register	6
3. Homepage	7

## Introduction

Nous vous exposerons dans ce rapport le déploiement de notre application Python Flask, que nous avons ensuite transférée vers un environnement Kubernetes.

Notre premier projet consistait en l'app.py qui était liée aux répertoires de base, à savoir le répertoire "templates" et le répertoire "static". Au sein du dossier "templates", nous disposons de nos différentes interfaces visuelles, tandis que dans le dossier "static", nous conservons les fichiers tels que les images, les feuilles de style CSS et les scripts JavaScript. En outre, nous avons instauré une base de données MySQL afin d'assurer le bon déroulement de l'interface utilisateur. Son but consistait à offrir à l'utilisateur la possibilité de s'inscrire, puis de se connecter afin d'accéder à la page d'accueil.

Pour transférer notre projet vers cet environnement, nous avons dû ajuster divers éléments. Cela a nécessité la création de divers fichiers, allant du Dockerfile aux divers fichiers YAML.

## Architecture & son évolution

### 1. Dockerfile

Comme indiqué précédemment, l'architecture de base consistait en notre fichier `app.py` couplé aux dossiers "templates" et "static". Comme demandé au sein de la 1ère étape, après avoir mis en place notre mini-app, nous devions créer une image docker et son dockerfile. Ainsi, nous avons ajouté un fichier **Dockerfile**, étant le suivant :

```
1  # Utilisez une image de base qui contient Python
2  FROM python:3.12
3
4  # Définissez le répertoire de travail dans le conteneur
5  WORKDIR /app
6
7  # Copiez les fichiers de votre application dans le conteneur
8  COPY . /app
9
10 # Installez les dépendances Python
11 RUN pip install -r requirements.txt
12
13 # Exposez le port sur lequel votre application Flask fonctionne
14 EXPOSE 5000
15
16 # Commande pour exécuter l'application Flask
17 CMD python ./app.py
18
```

Suite à la mise en place de ce fichier, ils nous suffisaient de faire usage des commandes de docker pour build notre image, la lancer pour la tester et enfin la publier sur le hub grâce aux commandes suivante :

- build de l'image

```
docker build -t itrixpro/app:0.0.2.RELEASE .
```

- run de l'image, pour vérifier son bon fonctionnement

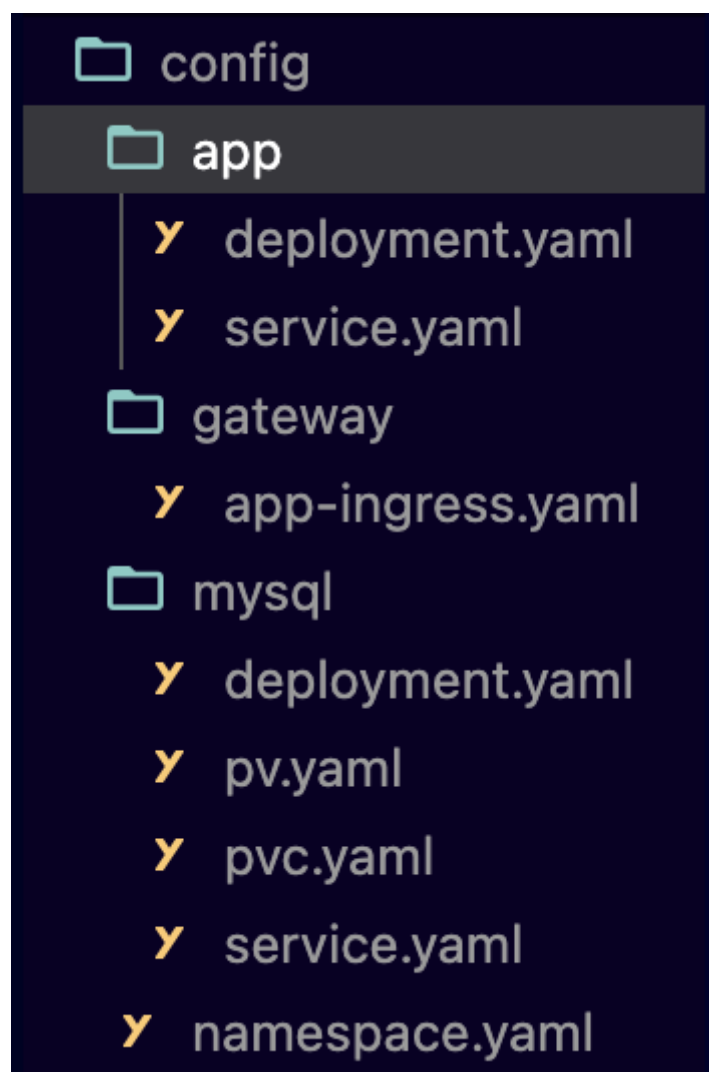
```
docker container run -d -p 5000:5000 itrixpro/app:0.0.2.RELEASE
```

- push l'image sur le hub

```
docker push itrixpro/app:0.0.2.RELEASE
```

## 2. Dossier config

Ayant plusieurs services à mettre en place, nous avons décidé de mettre en place un dossier config contenant les différents fichiers de déploiements, de services et autres. Tout en organisant le tout en sous dossier. Nous avons en premier lieu le dossier app, contenant les fichiers liés à la mise en place du service de notre application. Nous avons ensuite celui lié au fichier nécessaire pour la mise en place de notre gateway et enfin le dossier nommé mysql contenant tous les fichiers nécessaire pour la mise en place du service de notre base de données.



### 3. Scripts

Afin de faciliter la tâche tant pour l'utilisateur que pour nous-mêmes, nous avons développé un script appelé "setup.sh", qui exécutera les différentes commandes Kubernetes requises. Afin d'accomplir cela, il fera appel à des sous-scripts disponibles dans le répertoire "tools". De la même manière, nous avons créé un fichier nommé "clean.sh" pour nettoyer le setup.


```
➤ setup.sh
1  # set namespace
2  ./tools/namespace-setup.sh
3
4  # take care of the app
5  ./tools/app-setup.sh
6
7  # put in place the gateway
8  ./tools/gateway.sh
9
10 # setup the database
11 ./tools/database-setup.sh
12
13
```

```
tools
➤ app-setup.sh
➤ database-setup.sh
➤ gateway.sh
➤ namespace-setup.sh
```

```
➤ clean.sh
1  # clean up
2  kubectl delete --all persistentvolume --namespace=authentication
3  kubectl delete --all PersistentVolumeClaim --namespace=authentication
4
5  kubectl delete --all deployments --namespace=authentication
6  kubectl delete --all services --namespace=authentication
7
8  kubectl delete --all pods --namespace=authentication
9
10 kubectl delete namespace authentication
```

## Rendu final

### 1. Login



### Welcome

Sign into your account

Username

Password

Login

Don't have an account? [Register here](#)

### 2. Register

### CREATE AN ACCOUNT

Username

Password

Register

Have already an account? [Login here](#)

### 3. Homepage

#### Home Page

Hello, admin ! You are logged in !

[Log out](#)