

---

# PROJET FINAL DEVOPS

---

DevOps



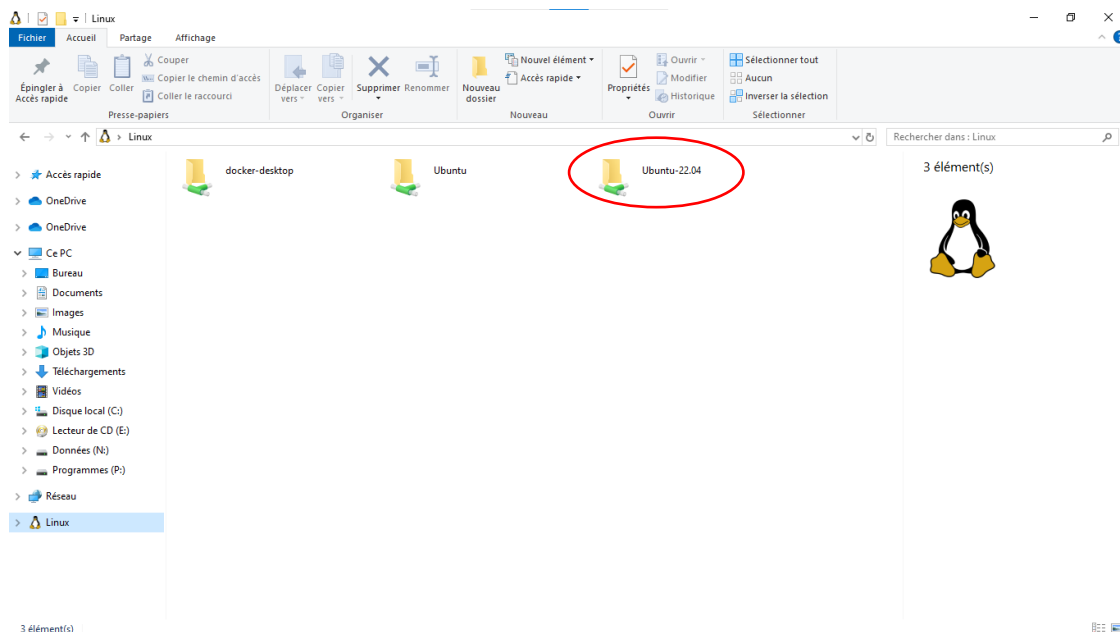
**Description :** Conteneurisation d'une application web dans une image Docker et publication dans le Docker Hub ; création d'un pipeline CI/CD avec Jenkins et Ansible puis déploiement Kubernetes.

TATOU TATOU Josias Nathan

28 DECEMBRE 2024

## Partie 1 : Conteneurisation de l'application Web

La conteneurisation de l'application s'est faite via WSL Ubuntu-22.04.



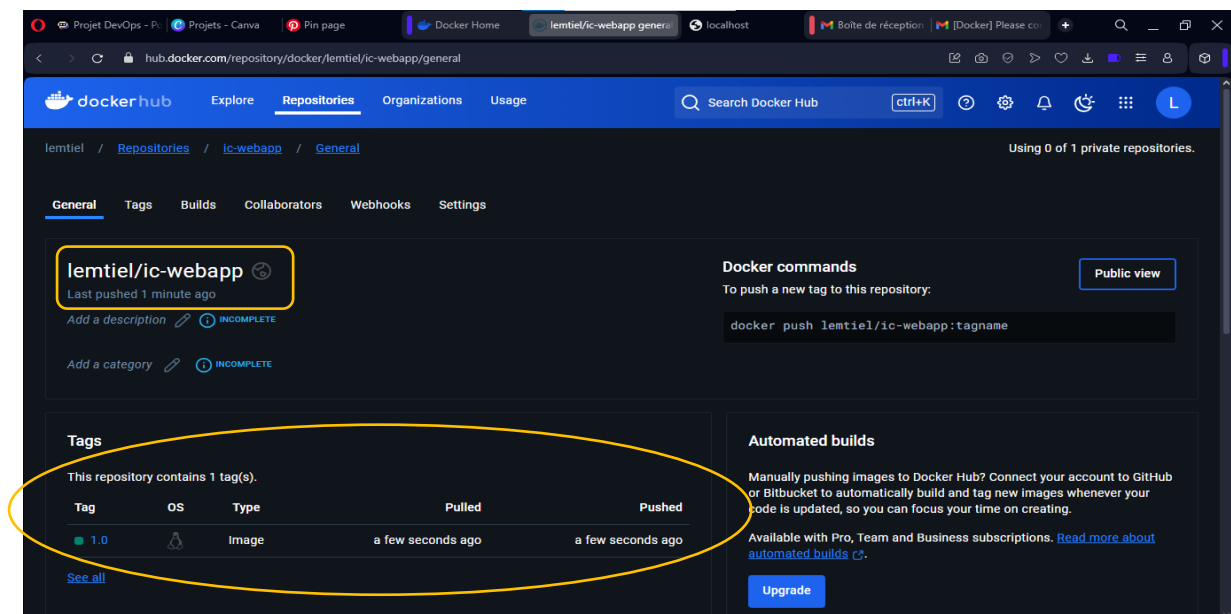
Démarrage de la WSL et création du Docker file. Dans ce fichier, on indique la version de Python à utiliser, le port sur d'écoute de l'application le point d'entrée de l'appli et quelques autres informations nécessaires :

```
1 # Étape 1 : Utiliser une image de base légère
2 FROM python:3.6-alpine
3
4 # Étape 2 : Définir le répertoire de travail
5 WORKDIR /opt
6
7 # Étape 3 : Copier les fichiers du projet dans l'image
8 COPY . /opt
9
10 # Étape 4 : Installer Flask
11 RUN pip install flask
12
13 # Étape 5 : Exposer le port utilisé par l'application
14 EXPOSE 8080
15
16 # Étape 6 : Définir des variables d'environnement dynamiques pour l'application
17 ENV ODOO_URL= https://www.odoo.com/fr_FR
18 ENV PGADMIN_URL= https://www.pgadmin.org/
19
20 # Ajout du script permettant d'extraire les variables d'environnement
21 RUN cat releases.txt | awk '/ODOO_URL/ {print $2}' | xargs -I {} echo {} > /opt/odoo_url.txt
22 RUN cat releases.txt | awk '/PGADMIN_URL/ {print $2}' | xargs -I {} echo {} > /opt/pgadmin_url.txt
23
24 # Étape 7 : Lancer l'application Flask
25 ENTRYPOINT ["python", "app.py"]
```

Une fois créé, le conteneur est testé pour vérifier que les URLs Odoo et pgAdmin configurées dans les variables d'environnement fonctionnent correctement.



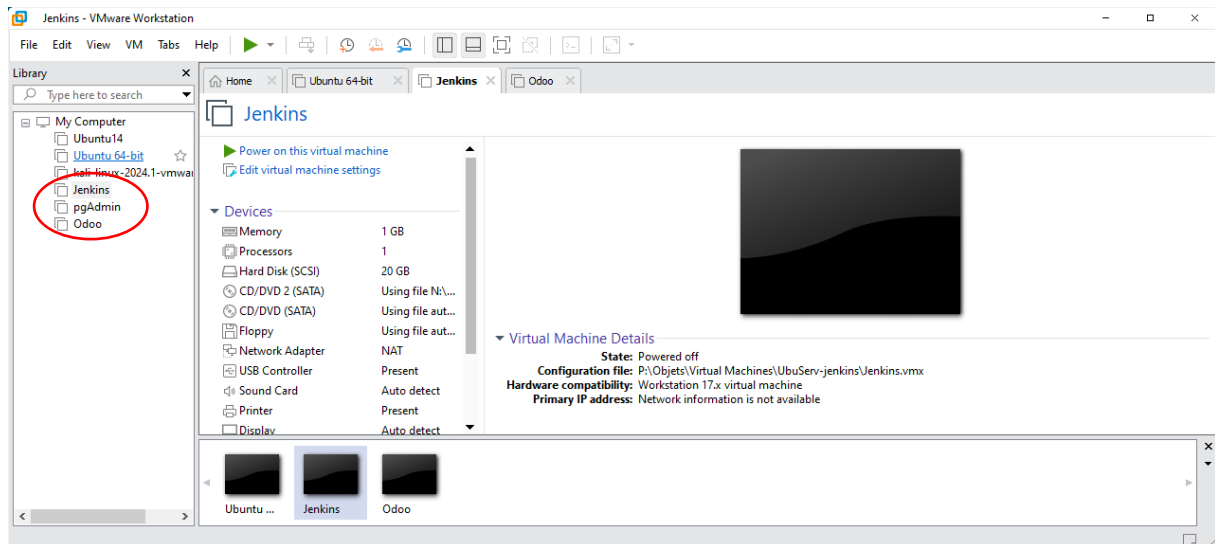
L'image est poussée sur Docker Hub pour faciliter le partage et l'intégration dans les étapes suivantes.



## Partie 2 : Création d'un pipeline CI/CD avec Jenkins et Ansible

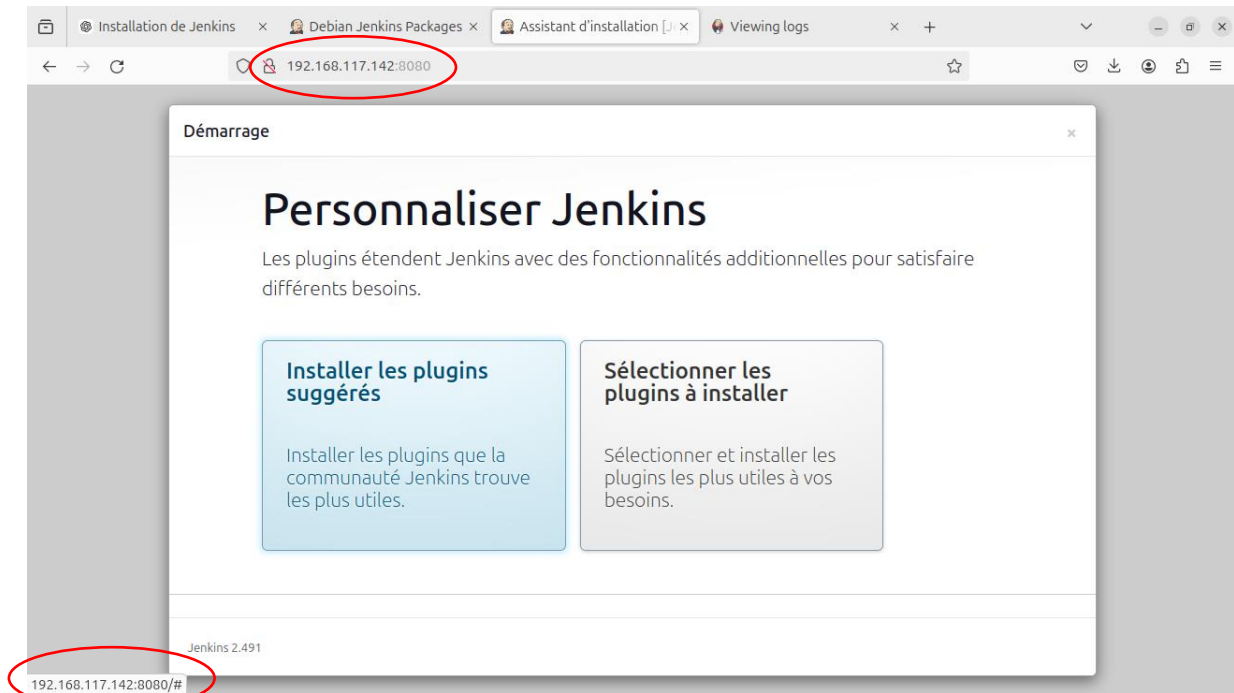
L'objectif principal de cette rubrique est d'automatiser le processus de build, test, et déploiement sur trois serveurs distincts. Pour cela, nous nous servons de notre image Docker qui a été build à la précédente étape, de trois serveurs virtuels où nous installerons respectivement Jenkins, Odoo, pgAdmin et le site vitrine. La gestion et l'automatisation du processus se fera grâce à **Ansible**.

Tout d'abord, créons nos serveurs virtuels ; ceux-ci utiliseront chacun le OS Ubuntu 24.04.



Comme leurs noms l'indique, chacun d'eux stockera le programme dont il porte le nom :

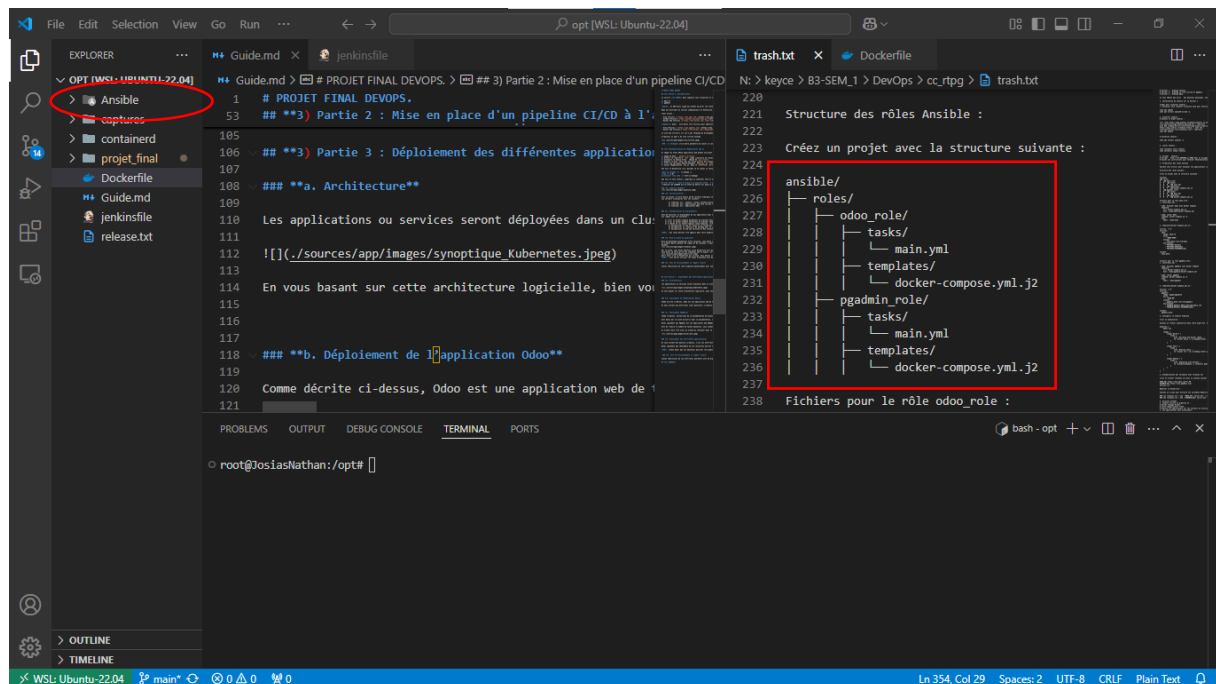
```
jenkins@jenkins-VMware-Virtual-Platform:~$ sudo apt-get install jenkins
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
jenkins est déjà la version la plus récente (2.491).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 309 non mis à jour.
jenkins@jenkins-VMware-Virtual-Platform:~$ java -version
openjdk version "17.0.13" 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Ubuntu-2ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Ubuntu-2ubuntu124.04, mixed mode, sharing)
jenkins@jenkins-VMware-Virtual-Platform:~$ sudo systemctl start jenkins
jenkins@jenkins-VMware-Virtual-Platform:~$ ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:08:8b:01 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.117.142/24 brd 192.168.117.255 scope global dynamic noprefixroute ens33
        valid_lft 1542sec preferred_lft 1542sec
```



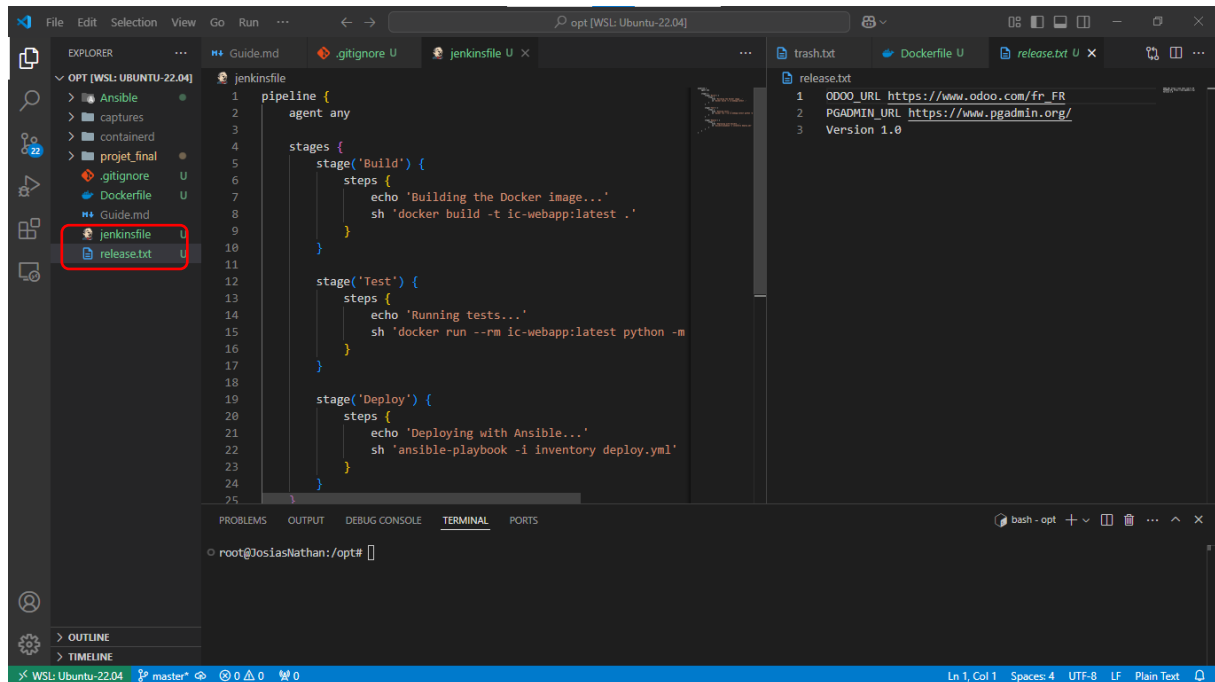
Dans chacun des serveurs, a été installé l'outil qui devait être utilisé. Place à présent à l'installation et la configuration de Ansible et de Jenkins dans la WSL. Pour cela on va créer des rôles pour

- Déployer Odoo avec persistance des données.
- Déployer pgAdmin avec des paramètres personnalisés
- Gérer les configurations dynamiques via des variables (nom des conteneurs, volumes, réseau Docker).

La structure de notre dossier Ansible se présente comme sur le schéma ci-dessous :



Une fois la structure réalisée, on va dans le dossier principal créer un **Jenkins file** pour la gestion du pipeline Jenkins et un fichier **releases.txt** contenant les informations sur les versions et URL ; il sera utilisé pour générer automatiquement les variables d'environnement nécessaires.



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays the file structure of a project in a WSL Ubuntu 22.04 environment. The files 'jenkinsfile' and 'releases.txt' are highlighted with a red box. The main editor area shows the content of 'jenkinsfile' and 'releases.txt'.

```
jenkinsfile
1 pipeline {
2   agent any
3
4   stages {
5     stage('Build') {
6       steps {
7         echo 'Building the Docker image...'
8         sh 'docker build -t ic-webapp:latest .'
9       }
10    }
11
12    stage('Test') {
13      steps {
14        echo 'Running tests...'
15        sh 'docker run --rm ic-webapp:latest python -m'
16      }
17    }
18
19    stage('Deploy') {
20      steps {
21        echo 'Deploying with Ansible...'
22        sh 'ansible-playbook -i inventory deploy.yml'
23      }
24    }
25  }
26 }
```

```
releases.txt
1 ODOO_URL https://www.odoo.com/fr_FR
2 PGADMIN_URL https://www.pgadmin.org/
3 Version 1.0
```

The terminal at the bottom shows the command prompt: `root@JosiasNathan:~/opt#`.

## Partie 3 : Déploiement Kubernetes