# Projet

Titre du projet : Déploiement automatisé d'une application web légère avec CI/CD

## Description du projet :

Créer une application web légère avec un backend et un frontend, et mettre en place un processus de déploiement automatisé utilisant Jenkins, Ansible, Kubernetes, Docker, Linux et Git. L'objectif est de faciliter le processus de développement et de déploiement pour les développeurs, tout en assurant une gestion optimale des ressources.

### Etapes du projet :

1. Créez une application web simple :

Backend : Utilisez un framework tel que Flask (Python) ou Express (Node.js) pour développer un serveur web léger avec quelques API de base.

Frontend : Utilisez un framework tel que React ou Vue.js pour développer une interface utilisateur simple qui consomme les API du backend.

1. Dépôt git

Créez un dépôt Git pour le projet et suivez les meilleures pratiques de gestion de version en utilisant des branches, des tags, des commit et des pull requests.

1. Conteneur

Créez des conteneurs Docker pour le frontend et le backend de l'application. Utilisez un fichier Dockerfile pour décrire les dépendances et les configurations nécessaires pour chaque service.

1. Kubernetes

Configurez Kubernetes pour déployer les conteneurs de l'application sur un cluster. Utilisez des fichiers de configuration YAML pour décrire les ressources Kubernetes (par exemple, les déploiements, les services et les volumes).

1. Ansible

Utilisez Ansible pour automatiser la configuration et la gestion du cluster Kubernetes et des ressources associées. Créez des playbooks Ansible pour déployer, mettre à jour et supprimer les ressources Kubernetes.

1. Jenins

Configurez Jenkins pour automatiser le processus de CI/CD (intégration continue et déploiement continu) :

* Créez des pipelines Jenkins pour déclencher des builds et des tests à chaque push vers le dépôt Git.
* Configurez des étapes de pipeline pour déployer les changements sur le cluster Kubernetes à l'aide des playbooks Ansible.

1. Hébergement avec linux

Utilisez Linux pour héberger les outils de CI/CD (Jenkins) et les environnements de développement et de production.

**Avec ce projet, vous mettrez en pratique l'utilisation de Jenkins, Ansible, Kubernetes, Docker, Linux et Git pour déployer une application web légère.**

**Nous : Détails de l’étape 3 :**

**Reponse :**

Étape 3 : Création des conteneurs Docker pour le frontend et le backend de l'application

Dans cette étape, vous allez créer des conteneurs Docker pour le frontend (React) et le backend (Flask) de l'application. Vous allez utiliser des fichiers Dockerfile pour décrire les dépendances et les configurations nécessaires pour chaque service.

1. Backend (Flask) :

a. Créez un fichier **Dockerfile** dans le répertoire du backend avec le contenu suivant :

# Utilisez une image de base Python 3

FROM python:3.9-slim

# Définissez le répertoire de travail

WORKDIR /app

# Copiez les fichiers de dépendance et installez-les

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

# Copiez le fichier app.py dans le répertoire de travail

COPY app.py .

# Exposez le port sur lequel l'application sera accessible

EXPOSE 5000

# Lancez l'application Flask

CMD ["python", "app.py"]

b. Construisez l'image Docker pour le backend en exécutant la commande suivante dans le répertoire où se trouve le fichier **Dockerfile** :

docker build -t my-web-app-backend .

2-Frontend (React) :

a. Créez un fichier **Dockerfile** dans le répertoire du frontend avec le contenu suivant :

# Utilisez une image de base Node.js

FROM node:14

# Définissez le répertoire de travail

WORKDIR /app

# Copiez les fichiers package.json et package-lock.json et installez les dépendances

COPY package\*.json ./

RUN npm install

# Copiez le reste des fichiers de l'application

COPY . .

# Construisez l'application React pour la production

RUN npm run build

# Utilisez une image de base Nginx pour servir l'application React

FROM nginx:1.21

# Copiez les fichiers de build React vers le répertoire de Nginx

COPY --from=0 /app/build /usr/share/nginx/html

# Exposez le port sur lequel l'application sera accessible

EXPOSE 80

# Lancez Nginx

CMD ["nginx", "-g", "daemon off;"]

b. Construisez l'image Docker pour le frontend en exécutant la commande suivante dans le répertoire où se trouve le fichier **Dockerfile** :

docker build -t my-web-app-frontend .

3-Vérifiez que les images Docker ont été créées avec succès en exécutant la commande :

docker images

Vous devriez voir les images **my-web-app-backend** et **my-web-app-frontend** dans la liste des images disponibles.

1. Pour tester les conteneurs localement, vous pouvez les exécuter avec les commandes suivantes :

docker run -d -p 5000:5000 --name backend my-web-app-backend

docker run -d -p 80:80 --name frontend --link backend my-web-app-frontend