# Exercice : Création d’un pipeline CI/CD avec AWS CodePipeline, CodeDeploy, CodeBuild et GitHub …… Fait par Donald Programmeur

---

### *1. Qu'est-ce que le CI/CD ?*

Le CI/CD (Continuous Integration / Continuous Deployment) est une approche qui permet d'automatiser les phases de développement, de test et de déploiement d'une application. Cette méthode améliore l'efficacité des développeurs et assure une mise en production plus rapide et sécurisée.

---

### *2. Que signifient Code, Build, Test et Deploy ?*

- **Code** : Phase où les développeurs écrivent et gèrent le code source de l'application.
- **Build** : Compilation du code source et génération des artefacts prêts à être exécutés.
- **Test** : Exécution de tests unitaires et fonctionnels pour vérifier la qualité du code.
- **Deploy** : Déploiement du code sur les serveurs ou infrastructures cloud pour le rendre accessible aux utilisateurs.

---

### *3. Pourquoi le CI/CD est important ?*

- Automatisation des tâches répétitives.
- Détection rapide des erreurs grâce aux tests automatisés.
- Déploiement plus rapide et plus fiable des applications.
- Réduction des risques en production grâce aux tests et validations continus.

---

## Mise en place d’un pipeline CI/CD sur AWS

---

## Étapes de l'exercice :

1. **Configuration du compte AWS et création d'un accès limité.**
2. **Création des rôles IAM pour EC2 et CodeDeploy.**
3. **Installation de l’agent CodeDeploy sur le serveur EC2.**
4. **Mise en place de la structure du code et des fichiers de configuration.**
5. **Création du pipeline CI/CD avec CodePipeline et CodeDeploy.**
6. **Test de l'application après le déploiement.**

---

## Étape 1 : Configuration des rôles IAM

### *1.1 Création du rôle IAM pour EC2*

- Accédez à la console AWS IAM.
- Créez un rôle IAM pour EC2 en choisissant la politique : `AmazonEC2RoleforAWSCodeDeploy`.
- Nommez le rôle **ec2-code-deploy-s3**.

- Validez la création du rôle.

### *1.2 Création du rôle IAM pour CodeDeploy*

- Créez un nouveau rôle IAM et sélectionnez **AWS CodeDeploy**.
- Donnez-lui le nom **aws-codedeploy-role**.
- Associez les permissions nécessaires à CodeDeploy.

---

## Étape 2 : Création et configuration d'une instance EC2

- Lancez une nouvelle instance **Ubuntu** sur AWS EC2.
- Nommez l'instance **Django-server**.
- Créez une nouvelle paire de clés pour vous connecter en SSH.
- Configurez les **règles de sécurité** pour autoriser le trafic HTTP depuis Internet.
- Attachez le rôle IAM **ec2-code-deploy-s3** à votre instance.
    - Accédez à "Actions" → "Sécurité" → "Modifier le rôle IAM" et mettez à jour le rôle IAM.
- **Redémarrez l'instance après la configuration.**

---

## Étape 3 : Installation de l'agent CodeDeploy

- Connectez-vous à votre instance EC2 via SSH :

```
bash
CopyEdit
ssh -i "votre-clé.pem" ubuntu@adresse-ip-de-votre-instance
```

- Mettez à jour le système :

```
bash
CopyEdit
sudo apt update
```

- Installez Ruby et Wget :

```
bash
CopyEdit
sudo apt install ruby-full wget -y
```

- Téléchargez l'agent CodeDeploy :

```
bash
CopyEdit
wget https://Bucket-name.s3.Region-
identifier.amazonaws.com/latest/install
```

- Changez les permissions du fichier d'installation :

```
bash
CopyEdit
```

```
chmod +x ./install
```

- Installez l'agent CodeDeploy :

```
bash
CopyEdit
sudo ./install auto > /tmp/logfile
```

- Vérifiez si l'agent CodeDeploy fonctionne :

```
bash
CopyEdit
sudo service codedeploy-agent status
```

    - **Si l'agent ne fonctionne pas**, démarrez-le :

```
bash
CopyEdit
sudo service codedeploy-agent start
```

---

## Étape 4 : Configuration du serveur et des fichiers de déploiement

- Installez et configurez **Gunicorn** et **Nginx** pour servir votre application Django.
- Ajoutez les fichiers de configuration :
    - `appspec.yml` pour CodeDeploy.
    - `buildspec.yml` pour CodeBuild.

📌 **Remarque** : Les fichiers de configuration sont disponibles sur GitHub, il suffit de les copier-coller dans votre projet.

---

## Étape 5 : Création du pipeline CI/CD sur AWS

### *5.1 Création de CodeBuild*

- Accédez à **AWS Console** → Recherchez **CodeBuild**.
- Cliquez sur **Créer un projet de compilation**.
- Nommez le projet **Django-project-build**.
- Sélectionnez **GitHub** comme source et connectez votre dépôt.
- Configurez l'environnement avec **Ubuntu (Amazon Linux 2)** et **Python 3.x**.
- Spécifiez le rôle IAM pour CodeBuild.
- Assurez-vous d'avoir un fichier `buildspec.yml`.
- Validez et créez le projet.

---

### *5.2 Création de CodeDeploy*

- Accédez à **AWS Console** → Recherchez **CodeDeploy**.
- Cliquez sur **Créer une application** et nommez-la **Django-application-1**.
- Créez un **groupe de déploiement** et sélectionnez le rôle IAM **aws-codedeploy-role**.
- Choisissez le type de déploiement **"In-Place"**.

- Validez et créez l’application.

---

### *5.3 Création de CodePipeline*

- Accédez à **AWS Console** → Recherchez **CodePipeline**.
- Cliquez sur **Créer un pipeline** et nommez-le **Django-project-pipeline**.
- Sélectionnez **GitHub** comme source et connectez votre dépôt.
- Configurez **CodeBuild** comme étape de compilation.
- Configurez **CodeDeploy** comme étape de déploiement.
- Spécifiez les rôles IAM requis.
- Validez et créez le pipeline.

---

## Étape 6 : Test de l'application

- Une fois le pipeline CI/CD créé, effectuez un commit sur GitHub et observez le pipeline en action.
- Vérifiez que l'application est bien déployée sur votre serveur EC2 en testant l’URL associée.

---

## Conclusion

À la fin de cet exercice, vous aurez mis en place un pipeline CI/CD complet sur AWS, automatisant l’intégration et le déploiement de votre application Django. 🚀