Pour mettre en place une chaîne complète MLOps, vous devez procéder étape par étape, en veillant à respecter les bonnes pratiques de développement et d'automatisation. Voici un plan d'action structuré et détaillé pour réaliser votre projet en plusieurs étapes, en intégrant les pratiques DevOps et les outils spécifiques au Machine Learning (ML).

### 1. \*\*Préparation de l'infrastructure et des outils\*\*

L'infrastructure est la première étape clé, car elle soutient l'ensemble du projet. Avant d'entamer le développement du modèle ou de l'application, vous devez vous assurer que l'environnement dans lequel votre application tournera est bien configuré.

#### a. \*\*Mise en place de l'infrastructure cloud via Terraform\*\*

- \*\*Objectif\*\* : Créer l'infrastructure cloud (ex. AWS, GCP) en utilisant \*\*Terraform\*\* pour l'IaC (Infrastructure as Code).

- Créez des instances EC2 (ou machines virtuelles sur un autre fournisseur) pour exécuter votre modèle et l'API.

- Configurez le stockage nécessaire, comme un \*\*bucket S3\*\* pour stocker les modèles et les données.

- Utilisez \*\*Terraform\*\* pour gérer la création et la configuration de l'infrastructure, y compris la gestion de l'accès (utilisation d'\*\*IAM\*\* si nécessaire).

#### b. \*\*Configuration des serveurs via Ansible\*\*

- \*\*Objectif\*\* : Déployer la configuration des serveurs avec \*\*Ansible\*\* ou une alternative comme \*\*Pulumi\*\*.

- Utilisez Ansible pour installer les dépendances nécessaires (Python, bibliothèques ML, Docker, etc.).

- Configurez les serveurs pour qu'ils puissent exécuter votre modèle de Machine Learning en production.

### 2. \*\*Développement du modèle de Machine Learning\*\*

Une fois l'infrastructure mise en place, vous pouvez développer l'application ML.

#### a. \*\*Développement du modèle\*\*

- \*\*Objectif\*\* : Créez ou utilisez un modèle de Machine Learning pour faire des prédictions (vous avez déjà un exemple de modèle avec un fichier `.pkl` et un script Python).

- Si vous développez votre propre modèle (option 1), suivez les étapes suivantes :

- Préparez les données.

- Entraînez le modèle (par exemple un modèle de classification comme RandomForest).

- Sauvegardez votre modèle en utilisant \*\*joblib\*\* (comme déjà fait dans votre code).

- Si vous utilisez un modèle préexistant via une API (option 2), l’API doit être configurée pour accepter les requêtes de prédiction.

#### b. \*\*Versioning des modèles avec MLflow\*\*

- \*\*Objectif\*\* : Gérer le versioning des modèles pour garder une trace des différentes versions de votre modèle.

- Installez et configurez \*\*MLflow\*\* pour le suivi des modèles.

- Intégrez MLflow dans votre pipeline d'entraînement pour suivre les différentes itérations du modèle (hyperparamètres, performance, etc.).

### 3. \*\*Développement de l'application\*\*

Vous avez déjà développé une partie de l’application en Python avec \*\*Tkinter\*\* pour l’interface utilisateur et une fonction de prédiction. Vous devrez toutefois l’adapter pour l’intégrer dans un cadre MLOps complet.

#### a. \*\*Création de l’API ML\*\*

- \*\*Objectif\*\* : Développer une API RESTful pour exposer votre modèle ML et faire des prédictions via HTTP.

- Utilisez \*\*FastAPI\*\* ou \*\*Flask\*\* pour créer une API simple qui acceptera des requêtes pour effectuer des prédictions.

- L’API doit charger le modèle ML et accepter les entrées sous forme de paramètres ou d’un payload JSON.

#### b. \*\*Containerisation avec Docker\*\*

- \*\*Objectif\*\* : Créez une image Docker pour déployer facilement votre modèle ML et API dans n’importe quel environnement.

- Créez un \*\*Dockerfile\*\* pour emballer votre code et les dépendances nécessaires.

- Utilisez Docker Compose pour gérer les dépendances si nécessaire (par exemple pour l'API et la base de données).

### 4. \*\*Pipeline CI/CD\*\*

Automatisez l'ensemble du processus de développement, de tests et de déploiement à l’aide d’un pipeline CI/CD.

#### a. \*\*Mise en place avec GitHub Actions\*\*

- \*\*Objectif\*\* : Mettre en place un pipeline CI/CD pour automatiser le processus de build, de test et de déploiement.

- Créez des workflows GitHub Actions pour exécuter :

- Des tests unitaires sur le code (tests de l’API, de la fonction de prédiction, etc.).

- La construction de l’image Docker.

- Le déploiement automatique sur le cloud (utilisez \*\*Terraform\*\* pour le déploiement).

#### b. \*\*Tests automatisés\*\*

- \*\*Objectif\*\* : Implémentez des tests pour garantir la qualité du code.

- Testez les API en utilisant des frameworks comme \*\*pytest\*\* ou \*\*unittest\*\* pour Python.

- Testez le modèle avec des ensembles de données de validation.

### 5. \*\*Surveillance et monitoring\*\*

Il est crucial de suivre les performances et la santé de l’infrastructure et du modèle en production.

#### a. \*\*Monitoring de l’infrastructure avec Prometheus\*\*

- \*\*Objectif\*\* : Collecter des métriques sur les serveurs (utilisation du CPU, mémoire, etc.).

- Installez \*\*Prometheus\*\* pour surveiller les métriques des instances cloud (serveurs EC2 ou autres).

#### b. \*\*Visualisation avec Grafana\*\*

- \*\*Objectif\*\* : Créer des dashboards Grafana pour visualiser les métriques collectées par Prometheus.

- Configurez Grafana pour afficher les métriques pertinentes (utilisation des ressources, temps de réponse de l’API, etc.).

#### c. \*\*Surveillance des performances du modèle\*\*

- \*\*Objectif\*\* : Surveiller les performances du modèle en production.

- Intégrez \*\*MLflow\*\* pour suivre la performance du modèle au fil du temps (précision, recall, etc.).

### 6. \*\*Documentation\*\*

L'un des aspects les plus importants du projet est la documentation pour garantir que le code soit compréhensible, reproductible et facile à déployer.

#### a. \*\*README et guide d'installation\*\*

- Documentez clairement l'architecture de votre projet.

- Fournissez un guide détaillé sur l'installation et l'exécution de l'application, y compris comment configurer l'infrastructure cloud.

#### b. \*\*Documentation de l’API\*\*

- Décrivez les points d'API et leurs paramètres, ainsi que les réponses attendues. Utilisez des outils comme \*\*Swagger\*\* pour générer automatiquement la documentation de l'API.

### 7. \*\*Nettoyage et Optimisation\*\*

Une fois toutes les étapes réalisées, vous devrez vérifier et nettoyer les ressources non utilisées dans le cloud, optimiser les coûts, et tester l'ensemble du pipeline.

---

### En résumé, voici l’ordre des étapes :

1. \*\*Mise en place de l'infrastructure cloud via Terraform\*\*

2. \*\*Configuration des serveurs via Ansible\*\*

3. \*\*Développement du modèle de Machine Learning et gestion via MLflow\*\*

4. \*\*Création de l'API RESTful pour les prédictions\*\*

5. \*\*Containerisation de l'application avec Docker\*\*

6. \*\*Création du pipeline CI/CD avec GitHub Actions\*\*

7. \*\*Surveillance de l'infrastructure et des performances du modèle\*\*

8. \*\*Documentation complète du projet\*\*

9. \*\*Optimisation des coûts et nettoyage\*\*

En suivant cette approche, vous pourrez développer et déployer un système complet et fonctionnel en respectant les meilleures pratiques DevOps et MLOps.