Mini projet : Nettoyage, Transformation et Déploiement d'un Modèle de Prédiction des prix de ventes de maisons (partie 2)

Etape 2 : Déploiement avec Kubeflow (Bonus)

Dans cette étape, nous allons construire un **pipeline ML** simple pour automatiser l'exécution des différentes phases du modèle de prédiction. Ce pipeline comprendra les étapes suivantes :

1. Prétraitement des données (preprocessing.py)
2. Entraînement du modèle (train.py)
3. Évaluation du modèle (evaluate.py)
4. Déploiement du modèle (deploy.py)

Chaque étape fonctionne dans un conteneur séparé, orchestré par Kubeflow, pour assurer la modularité et la scalabilité.

Présentation du Kubeflow :

Définition : Kubeflow est une plateforme open source conçue pour simplifier l'exécution des workflows d'apprentissage automatique sur Kubernetes. Son objectif principal est de rendre les projets ML reproductibles, scalables et faciles à déployer dans différents environnements (cloud, local, hybride). En tirant parti de Kubernetes, Kubeflow offre une infrastructure puissante pour l'entraînement, l'optimisation et le déploiement des modèles ML.

Objectifs

- Fournir une plateforme unifiée pour gérer l'ensemble du cycle de vie des projets ML, de la préparation des données au déploiement des modèles.
- Permettre aux équipes d'ingénieurs et de data scientists de collaborer efficacement dans des environnements distribués.
- Automatiser et orchestrer les tâches ML complexes pour gagner en productivité et en fiabilité.

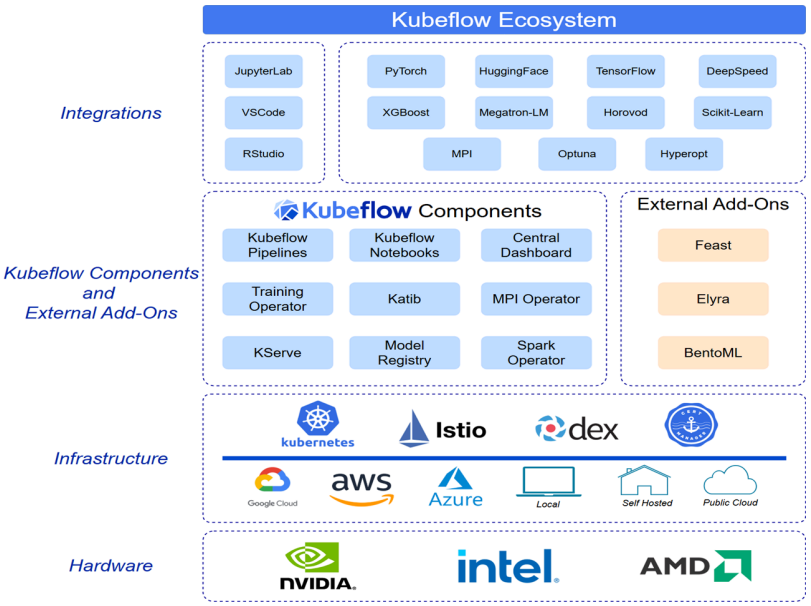
Avantages

- Scalabilité : Kubeflow s'appuie sur Kubernetes, ce qui permet de facilement adapter les ressources pour des charges de travail ML intensives (CPU, GPU, etc.).
- Portabilité : Compatible avec les principales plateformes cloud (Google Cloud, AWS, Azure) et les environnements locaux.
- Automatisation : Intégration de pipelines ML pour automatiser les processus d'entraînement, de validation et de déploiement.
- Reproductibilité : Chaque étape d'un workflow est traçable, ce qui garantit la cohérence des résultats.
- Flexibilité : Prise en charge de nombreux frameworks ML (TensorFlow, PyTorch, Scikit-learn) et d'environnements comme Jupyter Notebooks.
- Collaboration : Favorise le travail d'équipe grâce à ses outils intuitifs et centralisés comme le tableau de bord.

Architecture et écosystème de Kubeflow

L'architecture de Kubeflow repose sur quatre couches principales :

a) Intégration : Kubeflow s'intègre avec divers outils et frameworks pour offrir une expérience utilisateur complète. Par exemple : JupyterLab, VSCode, TensorFlow, PyTorch, Scikit-learn, etc.
b) Composants principaux et addons externes : Les composants principaux de Kubeflow offrent une large gamme de fonctionnalités pour le développement et le déploiement ML :
   - Pipelines Kubeflow : Permettent de concevoir, exécuter et surveiller des workflows ML. Chaque étape d'un pipeline peut être une tâche indépendante comme le prétraitement des données ou l'entraînement du modèle.
   - Katib : Plateforme pour l'optimisation et la recherche des hyperparamètres, l'exploration des modèles et la comparaison des résultats expérimentaux.
   - Notebooks Kubeflow : Permettent de créer des environnements interactifs pour prototyper et expérimenter avec des notebooks Jupyter. Il est possible de suspendre ou de reprendre les notebooks pour économiser les ressources.
   - KServe : Déploiement scalable des modèles ML pour l'inférence en production, gestion du trafic et mise à l'échelle automatique, prise en charge de plusieurs frameworks (TensorFlow, PyTorch, etc.).
   - Training Operator : Support pour l'entraînement distribué sur des frameworks comme TensorFlow et PyTorch.
   - Central Dashboard : Interface unifiée pour gérer tous les composants Kubeflow.
c) Infrastructure et matériel (hardware) : Ces composants englobent la gestion des ressources (avec Kubernetes), compatibilité matérielle (NVIDIA, Intel, AMD) et des outils supplémentaires permettant le réseautage, l'authentification, la gestion des services, les certificats TLS, etc (Istio, cert-manager, etc.).



Comment construire un pipeline ML simple avec Kubeflow ?

Un pipeline dans Kubeflow est un ensemble d'étapes indépendantes qui s'exécutent dans des conteneurs Docker distincts. Chaque étape est appelée un composant, et elles sont connectées entre elles pour former un pipeline complet.

Étape 1 : Préparer l'environnement

- Déployer Kubeflow sur un cluster Kubernetes préexistant (exemple, minikube en local).
- Assurez-vous que tous les services essentiels (Pipelines, Katib, Notebooks) sont correctement configurés.

Etape 2 : Créer un Notebook Kubeflow

- Accéder au tableau de bord de Kubeflow et créer un notebook.
- Installer les dépendances nécessaires (par ex., TensorFlow, scikit-learn).
- Préparer les données en les nettoyant et en les transformant selon les besoins du modèle.
- Deux options pour exécuter le code ML :
    - Exécuter directement les scripts Python dans le notebook.
    - Créer des images Docker pour chaque étape (prétraitement, entraînement, évaluation) et les héberger sur Docker Hub.

Étape 3 : Concevoir un pipeline Kubeflow

Une fois que nous avons les scripts Python ou les images pour chaque étape de la création du modèle ML, nous allons définir un pipeline Kubeflow. Cette dernière est définie par un script Python décrivant comment exécuter chaque étape sous forme de conteneur. Il faut utiliser le SDK “kfp” pour définir et exécuter des pipelines en Python :

```
import kfp

from kfp.dsl import pipeline, ContainerOp
```

Exemple en python :

```
@dsl.pipeline( //Permet la définition des méta-données du pipeline, notamment le nom, la description, etc.
    name="Exemple de pipeline",
    description="Pipeline ML simple avec Kubeflow"
)

def simple_pipeline(): //Une fonction permettant la création du pipeline en définissant chaque étape

    train = ContainerOp( // ContainerOp représente une étape du pipeline (prétraitement, entraînement, évaluation).
        name="Training",
        image="mydockerhub/training:v1",
        command=["python", "train.py"] //Commande à lancer
    ).after(preprocess) //assure que l’étape de l'entraînement s’exécute après le prétraitement.
```

Étape 4 : Exécuter le Pipeline

Soumettre le pipeline à Kubeflow :

```
import kfp
client = kfp.Client()
client.create_run_from_pipeline_func(simple_pipeline, arguments={})
```

Ensuite, Surveiller l’exécution dans le tableau de bord Kubeflow.