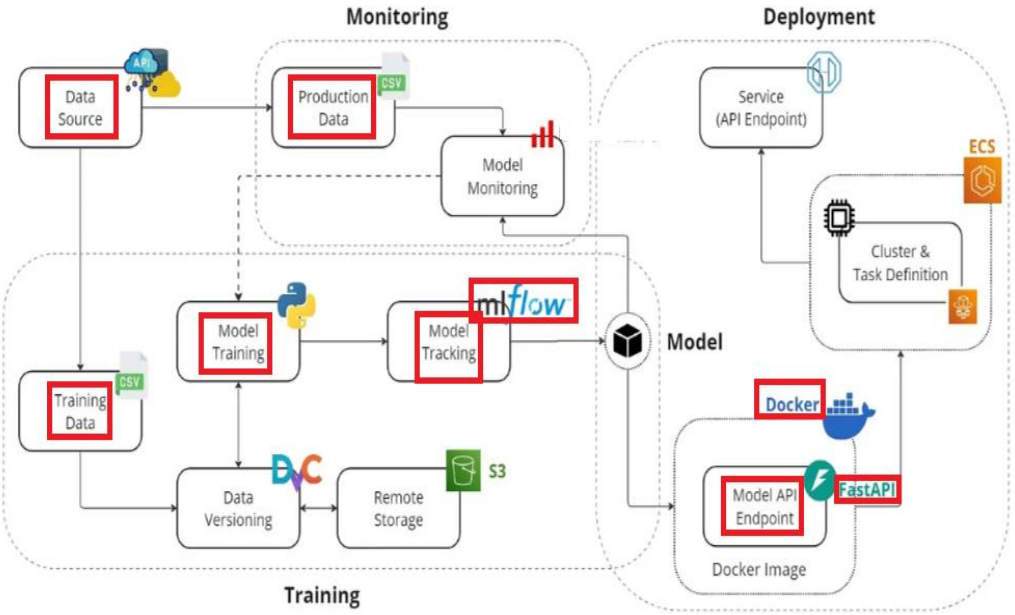
A.U : 2024-2025

Projet MLOps

Architecture de la solution finale :

★ Préparer en avance l'environnement de développement et tous les outils nécessaires.



Déroulement de la validation du projet MLOps :

★ La validation du projet MLOps se déroule en mode présentiel. Il s'agit d'une validation **technique** du projet MLOps (projet individuel) et une validation individuelle des acquis **théoriques** (QCM ou Q/R).

★ La validation se passera en deux phases comme suit :

- Semaine 6 [Validation des acquis théoriques + Validation de la **première** phase du projet (étapes 1, 2, et 3 : Modularisation, Automatisation et MLFlow)]

- Semaine 7 [Validation de la **deuxième** phase du projet (étapes 4, 5, et 6 : FastAPI, Docker et Monitoring)]

NB : Aucune partie abordée dans la semaine 6 ne sera validée de nouveau. Cette validation est consacrée exclusivement à la deuxième phase du projet.

★ Il est important de noter que la **note** obtenue dans le cadre de ce projet représente **60 %** de la **moyenne** du module.

Validation des acquis théoriques :

★ Des questions théoriques/techniques, sur les différents notions de cours, ateliers et outils utilisés, seront posées à chaque étudiant (QCM ou Q/R).

Validation technique du projet MLOps :

★ Chaque étudiant devra disposer de sa **propre solution** sur sa propre machine. La solution demandée **doit** inclure les étapes suivantes :

1. **Modularisation** du code depuis un projet Machine Learning (un script Jupyter fonctionnel (votre projet ML du premier semestre)).
2. **Automatisation** des étapes CI/CD avec l'orchestrateur Makefile (**déclenchement automatique des étapes CI/CD** suite à la modification de n'importe quelle fichier de votre projet et réalisation des étapes suivantes: qualité du code (utilisation des outils tels que Pylint, Flake8, MyPy, SonarQube), format du code (utilisation de l'outil Black), sécurité du code, etc.).
3. **Intégration** du projet avec MLflow (suivi des expériences et enregistrement des modèles : tracking et gestion de versions de modèles).
4. **Déploiement** du modèle (transformation du modèle en une API REST en utilisant FastAPI) et documentation (objet JSON)).
5. **Conteneurisation** avec Docker (construction de l'image Docker (pour l'application FastAPI) à partir du fichier Dockerfile, publication de l'image créée sur DockerHub et déploiement de l'application).

   Lancement du conteneur contenant le projet exposé avec FastAPI et test de la méthode predict avec des paramètres bien documentés et des valeurs concrètes
6. **Surveillance** des performances des modèles (monitoring continu avec MLflow et Elasticsearch et affichage d'un graphique de surveillance de l'évolution des métriques des modèles avec Kibana).

Pistes d'excellences :

★ La note attribuée à l'excellence sera décernée à ceux qui présentent des améliorations possibles dans chacune des étapes mentionnées ci-dessus, comme :

1. **Modularisation** du code :

- L'indépendance entre les fonctions (modules)

2. **Automatisation** des étapes CI/CD :
    - L'utilisation d'autres outils d'orchestration, tels que Jenkins, GitHub Actions, etc. (voir cours), pour l'automatisation du pipeline
    - La création des pipelines distinctes pour les différentes parties (data, training, code, etc.)
    - La réalisation et l'exécution des tests unitaires, des tests fonctionnels, etc.
    - Notification système (à la fin de l'exécution d'une tâche) ou notification par email (à la fin du pipeline)
3. **Déploiement** du modèle :
    - Utilisation d'autres formes de déploiement (plugin ou WebSocket)
    - Utilisation d'autres outils pour exposer et consommer le service (flask, django ou autres)
    - Exposition de la fonction retrain() comme un service REST
4. **Intégration** du projet avec MLflow :
    - Tracking (enregistrer une trace/courbe de training)
    - Gestion de versions de modèles (Registry Models)
    - Définition d'un backend de stockage
    - Automatisation avec Docker
    - Intégration avec un système de gestion de versions de modèles
5. **Déploiement** avec Docker :
    - Enregistrement des résultats de prédictions dans une base de données (deux images : BD et FastAPI) →Multi-conteneurs (Docker compose, Docker swarm, Minikube)
6. **Surveillance** des performances des modèles :
    - Vous pouvez proposer le monitoring de n'importe quelle partie du projet. Par exemple, on peut suivre le comportement de notre machine ubuntu (RAM, CPU, Espace disque, etc..), les conteneurs docker, les différents outils utilisés, les données, etc …..

Bon courage à tous 😀