Bloc 4
Projet final

David **RAMBEAU**

Full **S**tack in **D**ata **S**cience

Promotion 2024 -2025

- Sommaire -

1. **Contexte.** **3**
2. **Objectif.** **4**
3. **Vue d'ensemble.** **4**
4. **Architecture CI/CD** **5**
5. **Les tests critiques** **6**
6. **Monitoring** **7**
7. **Conclusion** **8**

- Projet final -

1. Contexte.

Le déploiement d'un modèle de machine learning en production ne marque pas la fin d'un projet, mais le début d'un cycle de vie de la donnée.

Sa pérennité dépend de trois piliers que sont le maintien des performances, le respect des contraintes réglementaires, et l'automatisation des déploiements.

En effet, les modèles ML sont vulnérables à la dérive des données qui dégradent leurs performances au fil du temps. Par exemple, notre modèle de détection de fraude peut perdre en efficacité si les comportements clients évoluent.

Parallèlement, des réglementations strictes comme le RGPD, l'IA Act ou le PCI-DSS imposent une traçabilité rigoureuse des données et des décisions, ainsi qu'une explicabilité des prédictions qui ont été faites.

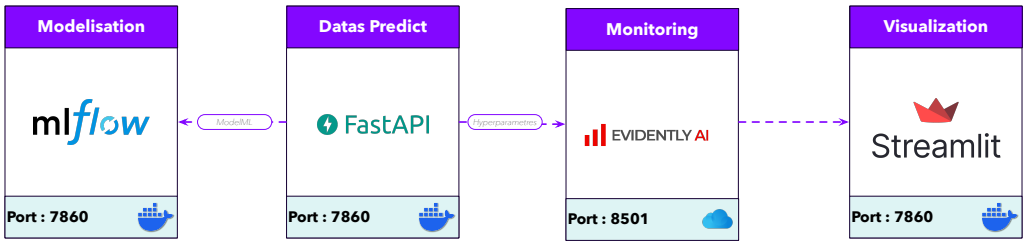
La problématique centrale réside donc dans la capacité à garantir la fiabilité du modèle dans la durée, tout en respectant les exigences légales tout en intégrant les évolutions au fil de l'eau.

Pour y répondre, il est nécessaire de mettre en place des mécanismes de monitoring continu pour détecter les dérives et déclencher des re-entrainements du modèle de façon automatisés.

Elles doivent aussi intégrer des pipelines CI/CD spécifiques au ML, incluant des tests de performance, de robustesse et de conformité avant chaque déploiement.

Enfin, une gouvernance renforcée est indispensable, incluant la documentation des versions, la traçabilité des données, et des audits réguliers.

Sans cette approche structurée, les modèles risquent de devenir obsolètes, inefficaces, voire non conformes, compromettant ainsi leur valeur opérationnelle.



2. Objectif.

L'objectif principal est d'opérationnaliser le système de détection de fraude en adoptant une approche industrielle, garantissant sa fiabilité, sa scalabilité et sa conformité en environnement de production. Pour ce faire, nous déployons un pipeline complet couvrant l'intégralité du cycle de vie du modèle, depuis le déploiement jusqu'au monitoring continu.

Ce pipeline repose sur trois axes majeurs :

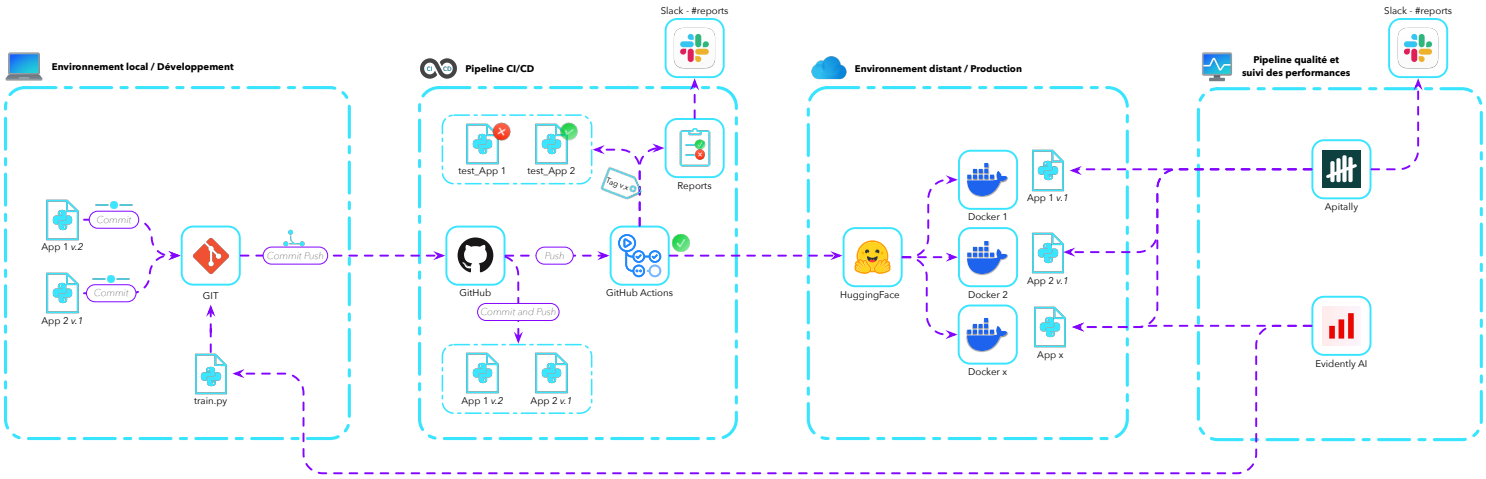
- *L'intégration de tests unitaires et d'intégration pour valider à la fois le code et les performances du modèle.*
- *L'automatisation des déploiements via des pipelines CI/CD, incluant des étapes de validation qualité avant toute mise en production. Cela permet de réduire les risques d'erreurs et d'accélérer les itérations.*
- *La surveillance en continu des indicateurs clés pour détecter rapidement les dérives ou anomalies. Des outils sont alors utilisés pour alerter les équipes en cas de dégradation des performances ou de comportements suspects.*

Cette industrialisation vise à donc transformer un prototype fonctionnel en une solution robuste et évolutive, capable de s'adapter aux évolutions de l'environnement du projet.

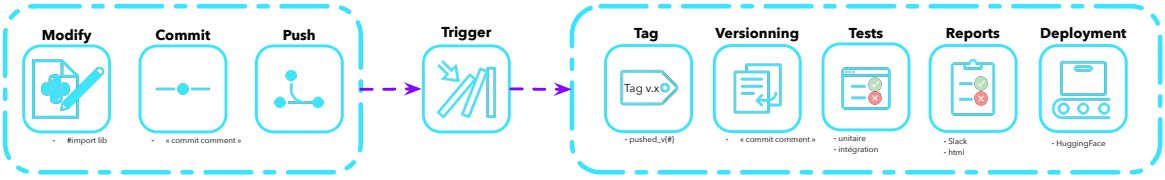
3. Vue d'ensemble.

Le schéma ci-après illustre l'architecture technique cible conçue pour industrialiser le processus de détection de fraude en temps réel.

Cette architecture vise à garantir une intégration des modifications et d'un niveau de qualité constant tout en conservant une forte réactivité face aux transactions suspectes.



4. Architecture CI/CD



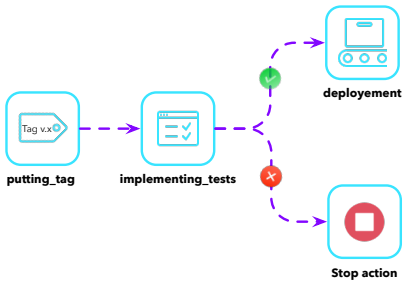
L'architecture CI/CD mise en place s'appuie sur GitHub Actions©, une solution intégrée au système de versionnage GitHub©, offrant ainsi une cohérence technologique et une intégration optimale avec l'environnement de développement existant.

Son mode de fonctionnement est entièrement automatisé dès lors qu'un « push » est effectué sur la branche principale. L'ensemble des tâches, des événements et des configurations sont concentrés dans un fichier unique qui permet d'opérer l'automatisation.
GitHub Actions© déclenche alors une série de tests via pytest, couvrant l'ensemble des fichiers éligibles. Un rapport détaillé est alors généré en fonction des résultats obtenus.

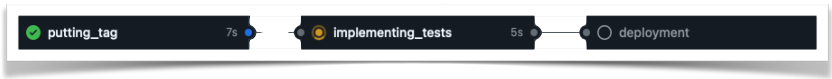
Ce processus d'intégration des modifications débute lorsqu'un « push » est effectuée sur la branche master, ce qui déclenche automatiquement le workflow GitHub Actions.
La première étape crée un tag de version (*pushed-v{numéro}*) assurant la traçabilité des déploiements.

Le pipeline exécute ensuite les deux portes de contrôle qualité séquentielles : les tests unitaires valident la logique métier et les endpoints API, les tests d'intégration vérifient la cohérence du système complet.

Deux cas de figure se posent alors, soit les tests sont validés et le pipeline procède automatiquement au déploiement de la solution mise à jour sur la plateforme HuggingFace©, assurant que la version en production reste toujours à jour.
Soit les tests échouent, le processus de déploiement s'arrête pour éviter d'exposer l'erreur en production. Alors, une alerte est transmise aux développeurs, accompagnée d'un rapport d'erreur, permettant une correction ciblée.



Parallèlement, un événement de log est généré en base de données pour garantir une traçabilité complète et une conformité aux exigences réglementaires, telles que le RGPD ou le PCI-DSS.



5. Les tests critiques

Pour qu'une mise à jour d'un composant soit validée et déployée, elle doit impérativement réussir une série de tests critiques, structurés sur deux niveaux, les tests unitaires et les tests d'intégration.

- Les **tests unitaires** visent à valider le fonctionnement isolé de chaque module ou fonction du code. Ils permettent de vérifier que chaque partie du logiciel, notamment les classes et les fonctions, se comporte comme attendu.

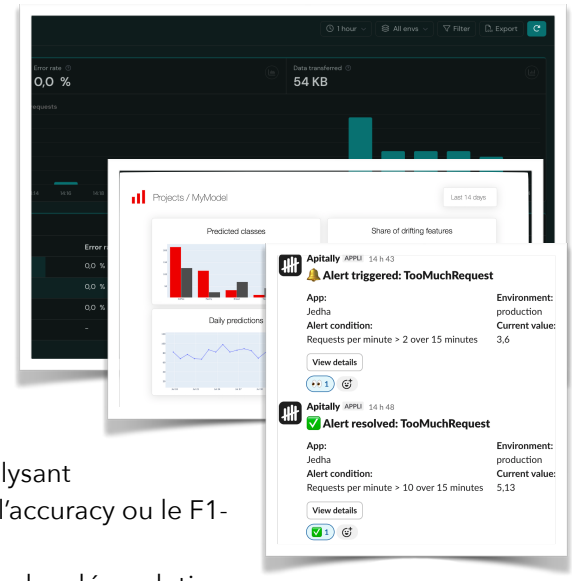
Par exemple, un des tests unitaires réalisés ici est de vérifier la connexion vers le Bucket S3© (créé dans le Bloc3) et de récupérer le modèle.
Ces tests sont généralement rapides à exécuter et couvrent les cas standards, mais également les scénarios d'erreur prévisibles.

- Les **tests d'intégrations**, quant à eux, évaluent les interactions entre plusieurs modules ou services, afin de s'assurer du fonctionnement global. Ils permettent de détecter des incompatibilités ou des dysfonctionnements lorsque les composants sont assemblés.

À l'issue de ces tests, des rapports détaillés sont générés et consignés dans la base de données OLTP NeonDB©.
Ces rapports incluent les résultats des tests, de leurs états de réussite (succès/échec), les temps d'exécution, ainsi que les éventuelles erreurs ou avertissements rencontrés.

Cette traçabilité permet non seulement de documenter la qualité du code à chaque itération, mais aussi de faciliter les audits et de répondre aux exigences réglementaires en matière de transparence et de reproductibilité, à savoir, comment est opéré l'industrialisation du modèle de prédiction.

6. Monitoring



La surveillance continue de la plateforme repose sur Apitally©, qui monitore les endpoints critiques de l'API FastAPI© et envoie des alertes Slack en cas d'anomalie.

Ce dispositif améliore la qualité de service en détectant les défaillances, assure une traçabilité complète des usages et permet d'anticiper les montées en charge en identifiant les pics de trafic avant saturation.

En complément, Evidently© (*en cours de développement à la rédaction de ces lignes*) va permettre de surveiller la dérive du modèle en analysant l'évolution des indicateurs de performance, comme l'accuracy ou le F1-score.

Cette surveillance permettra de détecter la dégradation progressive des performances liée au vieillissement des données d'entraînement et déclenche des alertes pour réentraînement préventif.

Evidently facilite également les phases d'apprentissage en comparant automatiquement les versions de modèles et en générant des rapports de performance.

La mise en place de cet outil permet d'établir une boucle de rétroaction qui engage donc le processus d'amélioration continue.

7. Conclusion

La construction d'un pipeline de distribution de solution logicielle continue est un atout majeur dans une infrastructure métier. Il garantit la qualité des livrables et directement la satisfaction des clients.

Ce bloc a permis de construire une infrastructure complète de production pour un système de détection de fraude bancaire.

Le pipeline CI/CD automatisé, orchestré via GitHub Actions©, intègre des tests de contrôle qualité garantissant et tests d'intégration afin de valider l'ensemble des flux .

L'architecture déployée sur Hugging Face Spaces© comprend une API FastAPI pour le scoring temps réel, une interface Streamlit© pour la supervision, et un système de tracking MLflow©.

Le monitoring Apitally© assure la traçabilité complète des prédictions en conformité RGPD et PCI-DSS.

Il est à noter que la mise en place d'un pipeline de données complet nécessite une expertise dans de nombreux domaines, tels que développement et exploitation, cloud et ingénierie ML.

Cette complexité peut venir générer une dette technique potentielle et nécessite des équipes formées capables de maintenir et faire évoluer l'infrastructure.

En conclusion, cette automatisation élimine les déploiements manuels et réduit drastiquement les risques d'erreur humaine, tout en accélérant les cycles de mise en production en conformité avec les réglementations !