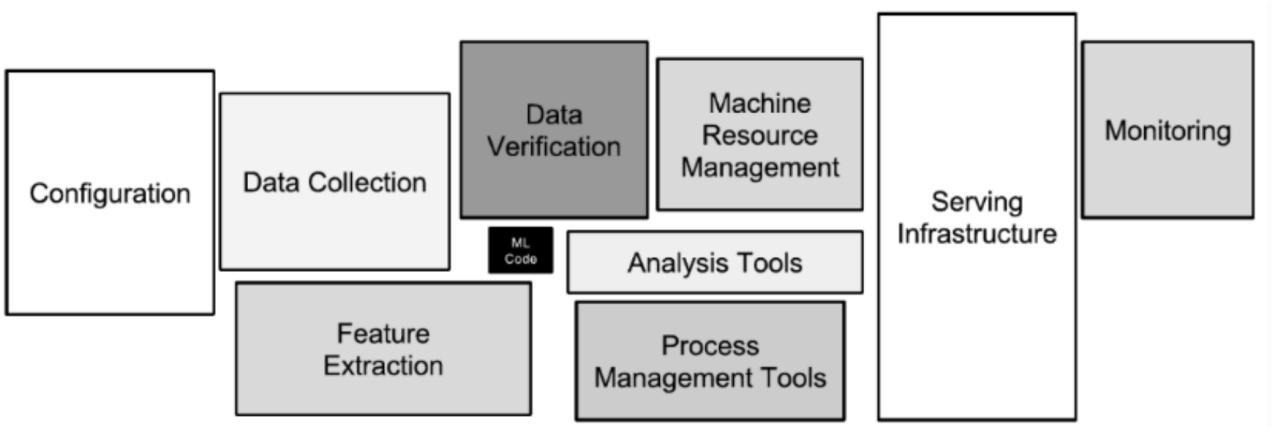
MLOps : pipelines d'automatisation

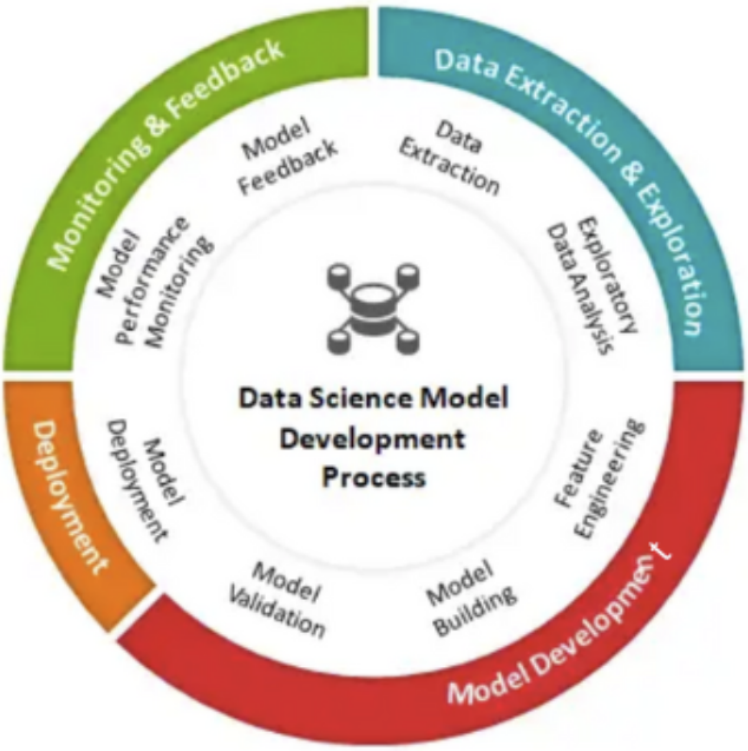
Dans cette note, on va aborder les différents concepts et définitions afin de comprendre le besoin de MLOps, son état et ses principaux composants. De plus, on aura un aperçu du cycle de vie de l'apprentissage automatique et des défis possibles. A la fin de la note, on propose un exemple d'architecture MLOps qu'on va développer dans le cadre de notre formation.

1. Cycle de vie d'un système ML

Le cycle de vie ML n'implique pas seulement la construction de modèles d'apprentissage automatique, l'infrastructure d'un système d'apprentissage automatique est vaste et complexe et seule une petite fraction de celle-ci est composée du code du modèle (le petit composant dans la figure ci-dessous).



C'est pourquoi il est crucial d'avoir un pipeline robuste. Dans le domaine de la science des données, le cycle de vie du système peut être présenté sous différentes formes en fonction du domaine d'activité, mais toutes ces étapes sont communes. La figure suivante montre les étapes du cycle de vie de l'apprentissage automatique.



Extraction et exploration des données L'extraction des données est généralement la première étape de tout projet ML : elle consiste à collecter les données à partir de sources multiples, qu'il s'agisse d'un fichier csv par exemple ou d'un big data. Une fois qu'on a les données, on devra les explorer et les traiter. On peut vérifier s'il y a des valeurs nulles ou des doublons et mettre à l'échelle certaines colonnes si nécessaire.

Développement du modèle La première étape de la phase de développement du modèle consiste à appliquer les techniques d'ingénierie des caractéristiques et de sélection des caractéristiques dans le but de sélectionner ou de créer les caractéristiques les plus informatives pour effectuer les prédictions ultérieurement.

Une fois les caractéristiques préparées, on peut construire et entraîner notre modèle sur une partie des données qu'on appele les données d'entraînement. On peut évaluer la qualité du modèle en vérifiant s'il renvoie de bonnes prédictions sur la partie restante des données qu'on appelle l'ensemble de test. En fonction des résultats, on peut raffiner le modèle jusqu'à ce qu'on trouve les meilleurs hyperparamètres et l'architecture la plus performante du modèle.

Déploiement L'étape de déploiement permet de rendre opérationnel notre modèle ML. On va le prendre de l'environnement de développement et l'intégrer dans un environnement de production afin qu'il soit accessible aux utilisateurs finaux. Par exemple, il peut être un point de terminaison de REST API qui répond aux demandes en temps réel.

Surveillance et retour d'information Lorsque le modèle est déployé en production, il est crucial qu'il continue à être performant. Il doit donc être surveillé pour s'assurer que les performances ne se dégradent pas avec le temps. En fonction des résultats de la surveillance de notre système ML, on peut décider de réentraîner le modèle si nécessaire.

2. Besoin du MLOps

2.1. Dette technique dans les systèmes ML

"La dette technique (technical Debt)" est un terme lié à un code immature, incomplet ou inadéquat (en raison de défauts de conception, de faible qualité ou d'autres problèmes), qui nécessitera un travail supplémentaire pour être corrigé. Les choses se compliquent pour les systèmes ML, ces systèmes sont constitués non seulement du code typique des systèmes logiciels traditionnels, mais aussi de composants supplémentaires tels que le modèle et les données. La présence de cette couche supplémentaire de complexité va probablement ajouter des problèmes supplémentaires. Certaines dettes techniques peuvent être liées aux dépendances des données, à la complexité du modèle, à la reproductibilité, aux tests et à la gestion des changements dans le monde réel.

2.2. Défis

Plus de la moitié des modèles d'analyse et de ML créés par les entreprises aujourd'hui ne sont pas mis en production. Certains des défis auxquels ils sont confrontés sont d'ordre technique ou organisationnel, comme on l'a vu dans la section sur la dette technique. Les projets ML sont beaucoup plus délicats que les systèmes logiciels traditionnels, car il faut tenir compte d'un grand nombre de points lors de la construction d'un système ML :

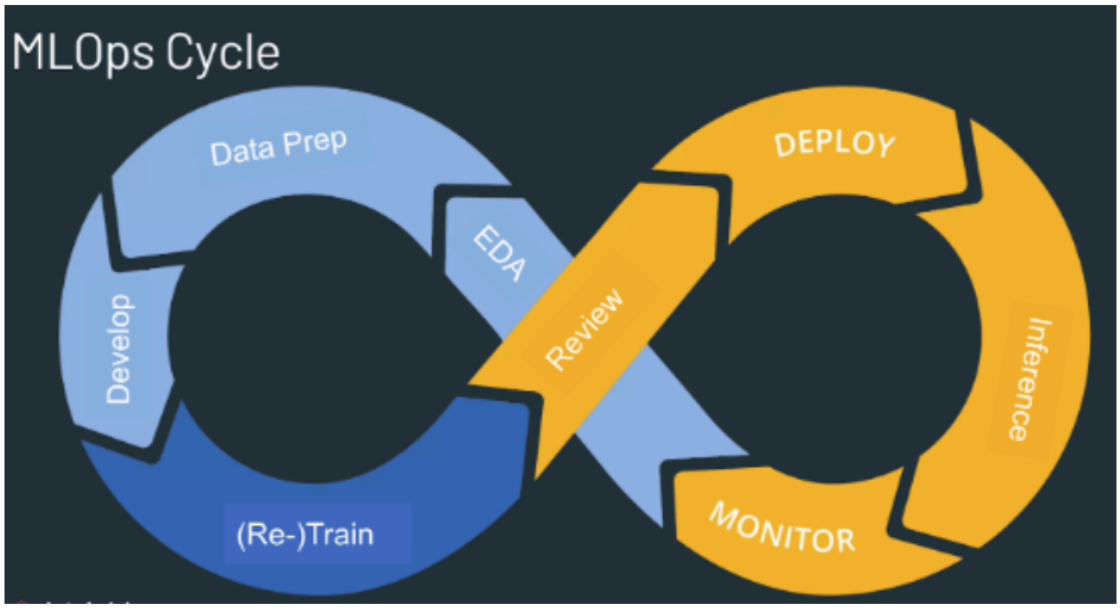
- Augmenter la collaboration entre les différents membres de l'équipe.
- Choisir les outils et les frameworks adéquats.
- Garder la trace du code et des données utilisés à chaque itération pour pouvoir reproduire le travail à tout moment.
- Garder la trace des différentes expériences et modèles en suivant les métriques et paramètres utilisés pour construire ces modèles.
- Automatiser autant que possible le pipeline ML.
- Valider la qualité des données avant de les utiliser dans la phase de train.
- Tester le modèle (les systèmes ML ont leurs propres tests spécifiques , les types traditionnels de tests logiciels ne sont pas suffisants).
- Manque de modularité et de personnalisation : En raison du manque de personnalisation, la réutilisation d'un modèle pour différentes tâches ou le traitement de différents formats d'entrée nécessite un effort important.
- Dégradation progressives des modèles ML : Les systèmes ML présentent une dégradation prévisible des performances au fil du temps si les modèles ne sont pas mis à jour avec de nouvelles données.
- Surveillance de la fiabilité en raison du manque de généralisabilité du modèle : Un modèle ML déployé nécessite une surveillance constante pour évaluer la sortie par rapport à de nouvelles entrées.

Afin de relever ces défis, un nouveau concept a été introduit, appelé MLOps, qui signifie Machine Learning Operations. Il est apparu au cours des dernières années dans le but de résoudre les problèmes de déploiement auxquels les équipes ML sont confrontées.

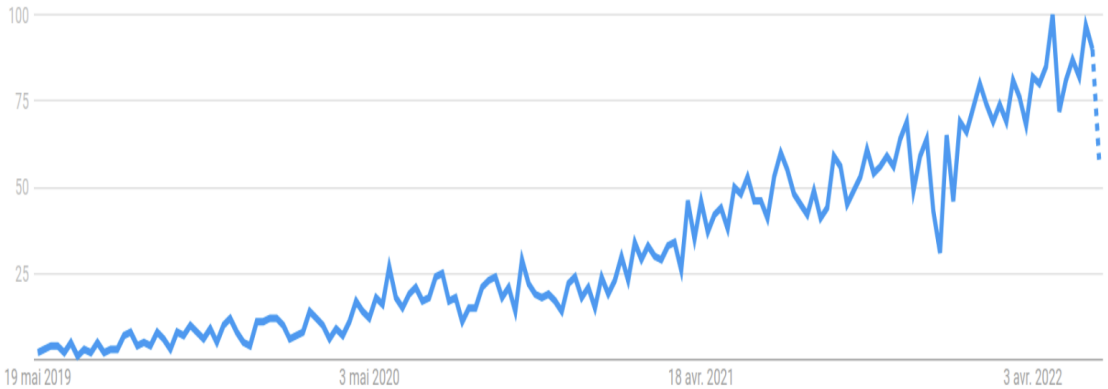
2.3. C'est quoi MLOps ?

MLOps consiste en un ensemble de principes et de pratiques de collaboration et de communication entre les data scientists et les professionnels des opérations. L'application de ces principes **augmentera la qualité** en simplifiant le processus de gestion, **réduira les risques et les coûts**, et **automatisera** autant que possible le pipeline ML.

MLOps est une capacité essentielle de l'ingénierie ML, axée sur l'optimisation du processus, la mise en production des modèles ML, puis leur maintenance et leur surveillance. Il s'agit d'une capacité de collaboration qui implique généralement les data scientists, les ingénieurs devops.



Une recherche sur GoogleTrends montre que le terme MLOps suscite un intérêt croissant, tant d'un point de vue scientifique que pratique, et qu'il s'agit de l'un des concepts les plus prometteurs.



3. Caractéristiques du MLOps

MLOps a beaucoup de principes et de pratiques. Comme on l'a avancé, MLOps vise à améliorer la collaboration entre les différents membres de l'équipe. Ceci peut être atteint par :

3.1. Gestion des versions et suivi des expériences

Les données et les modèles finiront par changer au fil du temps. On peut avoir une nouvelle version du modèle, les données peuvent changer en raison du flux d'informations. De même, les data scientists peuvent tester différents modèles ML et donc apporter certaines modifications à l'ensemble de données. Ainsi, le stockage des anciennes versions des données peut aider les organisations à reproduire l'environnement précédent.

Dans une approche MLOps, il est crucial de conserver des données et des modèles versionnés. Cependant, le versionnement des différents composants du système ML est plus difficile que celui des systèmes logiciels traditionnels. Les données ne peuvent pas être versionnées en utilisant uniquement les systèmes de contrôle de version classiques (par exemple git) en raison de leur taille importante, c'est pourquoi des outils spéciaux comme DVC sont utilisés pour résoudre le problème du stockage.

Le versionnage des modèles nécessite également une configuration différente de celle du code classique, on devra garder la trace de la façon dont le modèle lui-même a été construit : les paramètres utilisés, l'environnement dans lequel le modèle est développé, les artefacts connexes, les métriques obtenues, etc. En apprentissage automatique, pour qu'un modèle soit performant, de nombreuses expérimentations peuvent être nécessaires. Garder la trace de ces paramètres nous aidera à évaluer nos modèles ultérieurement.

3.2. Modularité

Dans les MLOps, du point de vue de l'architecture du système, les composants doivent être faiblement couplés. Cette caractéristique architecturale clé permet aux équipes de tester et d'utiliser facilement les composants individuels. En créant un code indépendant pour chaque partie de notre pipeline ML, il sera plus facile d'apporter des modifications supplémentaires ou même de réutiliser le code ultérieurement.

3.3. Testing

Afin d'assurer la qualité et la fiabilité de chaque composant du système ML, il est crucial de les tester. Cependant, le test des systèmes ML n'implique pas seulement le test typique des logiciels. Ci-dessous les principales différences entre les systèmes ML et les logiciels traditionnels :

1. Les logiciels sont composés de code ; toutefois, les systèmes ML présentent des niveaux de complexité supplémentaires puisqu'ils combinent données, modèle et code.
2. Les logiciels sont sujets à des défaillances bruyantes, généralement lorsqu'une fonctionnalité ne s'exécute pas correctement, on obtient des erreurs et des bugs et on peut déterminer la source du problème, alors qu'un système ML est sujet à des défaillances silencieuses, par exemple, on peut soudainement avoir une baisse de performance du modèle sans avoir d'erreurs dans le code, ce qui rend plus difficile l'identification de la cause de la défaillance.

En raison de ces différences, voici quelques erreurs courantes commises lors du test des systèmes ML :

- Tester uniquement le modèle, et non le système ML dans son ensemble.
- Ne pas tester les données
- Ne pas mesurer la relation entre les métriques de performance du modèle et les métriques métier.

- Penser que les tests hors ligne sont suffisants, et donc, ne pas surveiller ou tester en production.

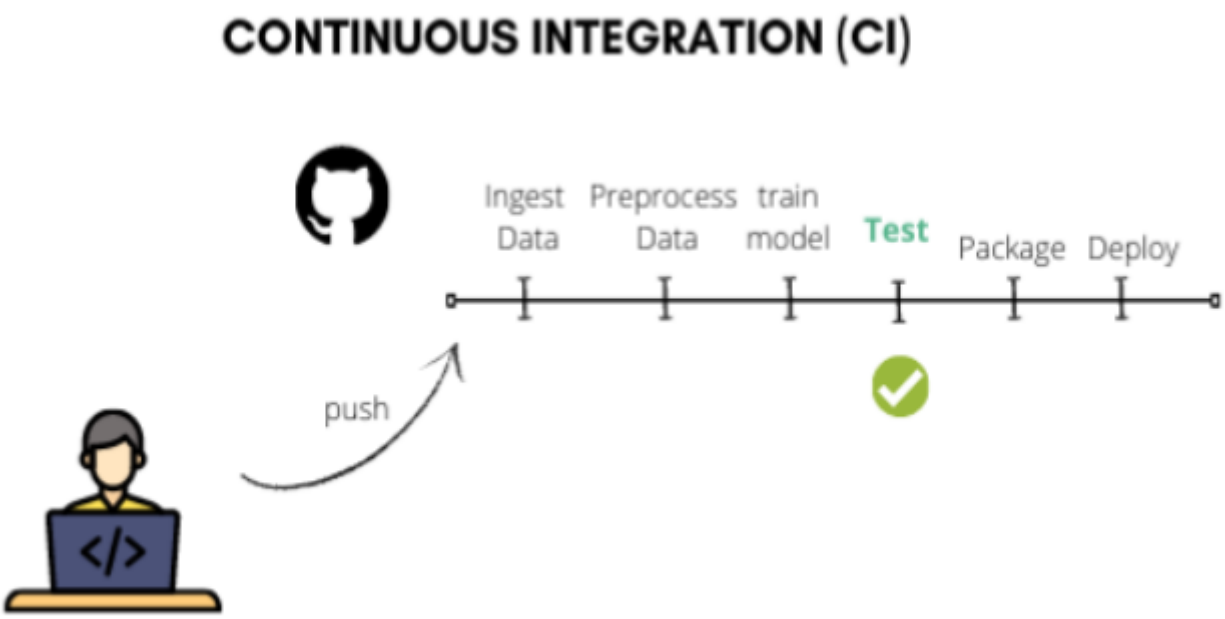
Si certains aspects sont intrinsèquement incertains et difficiles à automatiser, différents types de tests automatisés peuvent apporter une valeur ajoutée et améliorer la qualité globale du système. Les tests comprennent :

- **Validation des données** : Les données d'entrée doivent être validées par rapport au schéma attendu , avec des hypothèses sur leurs valeurs valides.
- **Valider la qualité du modèle** : On devra vérifier que le modèle prend des décisions qui s'alignent avec les objectifs de l'entreprise tout en ayant une bonne performance basée sur des métriques de perte (MSE , Loss , etc.).
- **Valider la distribution des données** : Par exemple,il peut y avoir des données déséquilibrées pour une caractéristique donnée (par exemple le sexe ou la région) par rapport à la distribution réelle.
- **Éviter les modèles périmés** : le système, en production, ne doit pas avoir de modèles périmés, ce qui pourrait affecter la qualité de la prédiction.
- **Tester la reproductibilité** : l'entraînement des modèles ML doit être reproductible, ce qui signifie que l'entraînement du modèle sur les mêmes données doit produire des résultats identiques. Avant de servir le modèle à un environnement de production, on devra vérifier et valider si l'environnement d'entraînement a donné environ le même score que le modèle dans l'environnement de service.

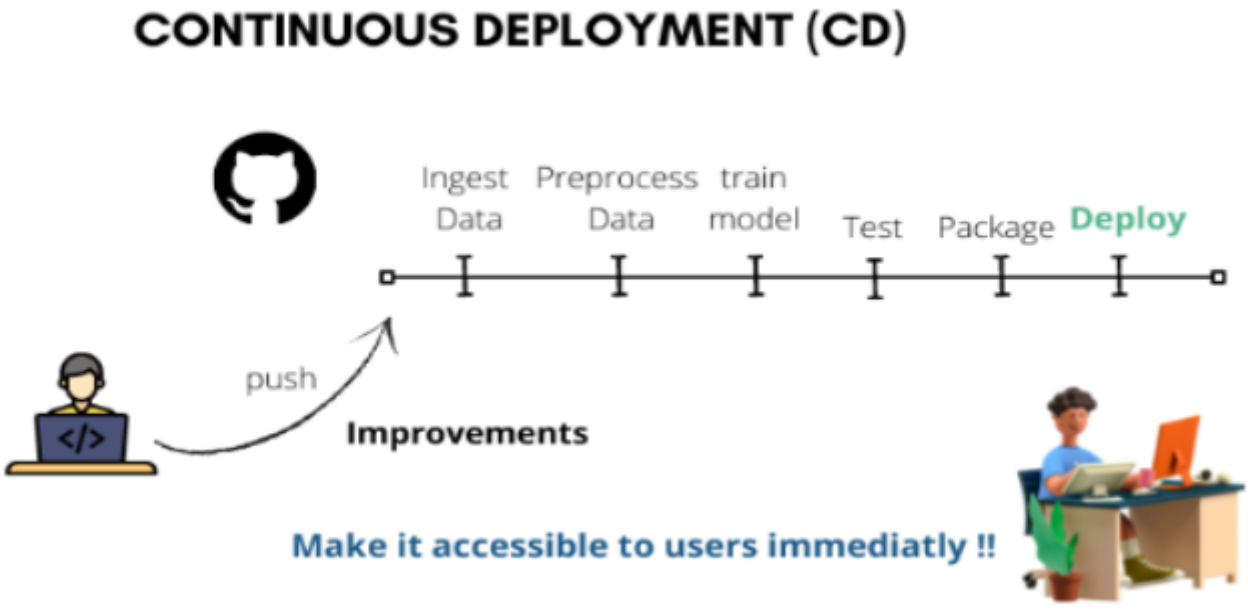
3.4. Continuous

L'un des concepts les plus importants de MLOps est l'intégration continue (CI), le déploiement continu (CD) et la formation continue (CT), qui permettent une livraison continue au service ML.

Intégration continue Dans les logiciels traditionnels, l'intégration continue est la pratique consistant à tester et à valider le code et les composants. Elle fait également référence à l'exécution de tests unitaires lors de la modification du code source. L'objectif de l'intégration continue est de s'assurer rapidement que les nouvelles modifications apportées par les développeurs sont "bonnes" et adaptées à une utilisation ultérieure. Dans le contexte des systèmes ML, l'intégration continue consiste à tester et à valider non seulement le code et les composants, mais aussi les données, les schémas de données et les modèles. Ainsi, chaque fois qu'on apporte des modifications et qu'on les committe sur notre dépôt git, ces tests seront déclenchés automatiquement pour vérifier que tout est correct avant de passer en production.



Déploiement continu La livraison continue est la capacité de mettre en production des changements, y compris de nouvelles fonctionnalités, des modifications de configuration et des corrections de bogues, de manière sûre et rapide et durable. Cela signifie que chaque fois qu'on apporte des améliorations, on veut que l'application soit immédiatement accessible aux utilisateurs.



Entraînement continu consiste à réentraîner notre modèle automatiquement chaque fois que cela est nécessaire. Par exemple, une mise à jour se produit ou de nouvelles données sont nécessaires.

3.5. Monitoring

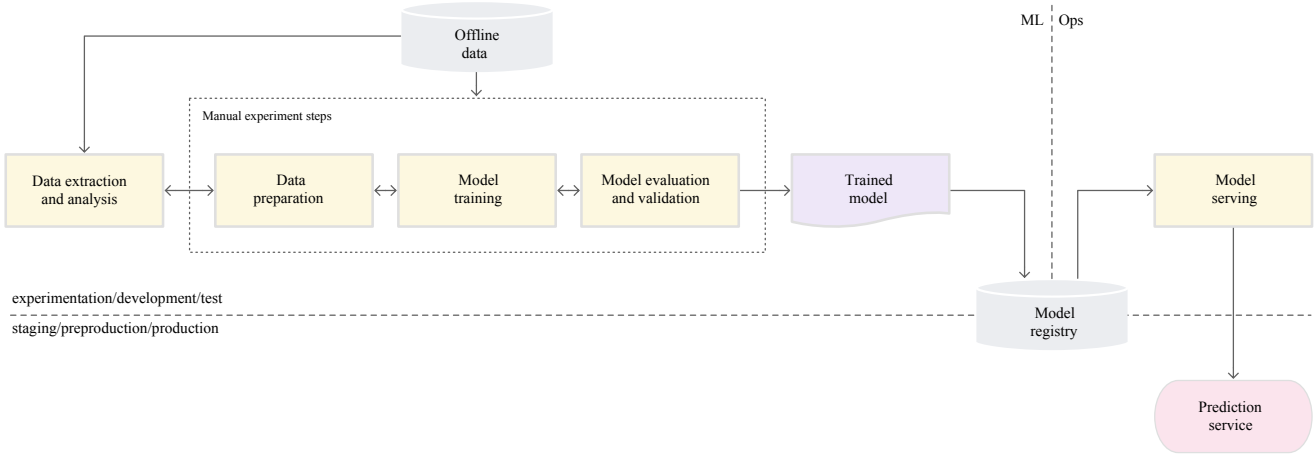
Dans un système ML, les performances du modèle se dégradent avec le temps, les données changent et le modèle lui-même doit être réentraîné. Une fois que le modèle a été déployé, il doit être vérifié pour s'assurer qu'il fonctionne comme prévu en production. On veut avoir une observabilité totale des

fonctionnalités et des performances de notre produit. On peut poser plusieurs questions : Les utilisateurs s'attendent-ils à des problèmes ? Le modèle donne-t-il des prédictions cohérentes ? La distribution des données change-t-elle avec le temps ? L'application prend-elle trop de temps pour répondre ? Toutes ces questions peuvent être répondues par le résultat de la surveillance du système ML.

3.6. Automatisation

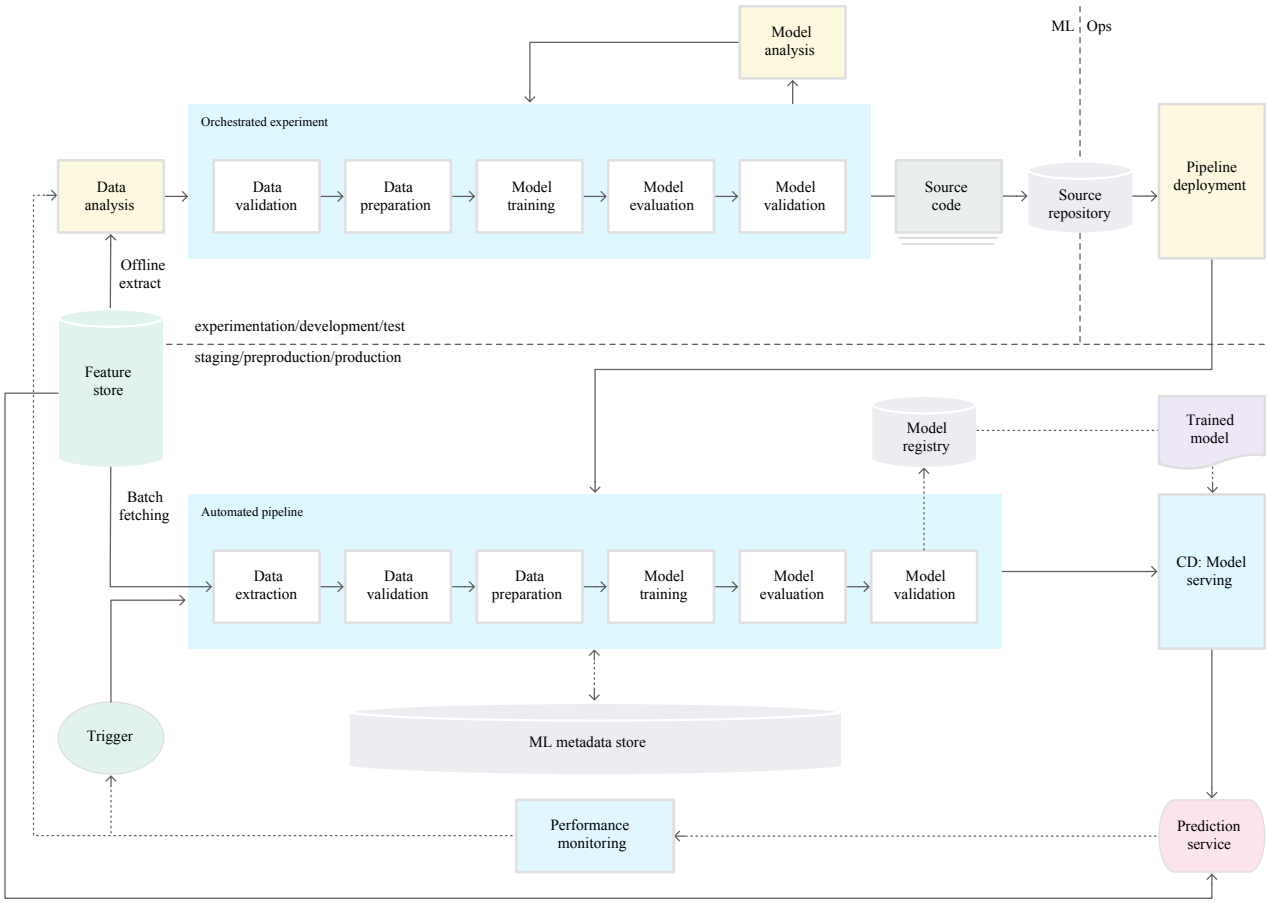
L'automatisation est très importante dans le MLOps car son niveau définit la maturité du système ML, qui reflète la vitesse d'apprentissage des nouveaux modèles. Google identifie trois niveaux d'automatisation pour un système ML (voir [1]) :

Niveau 0 : processus manuel Le niveau 0 correspond à un processus de construction et de déploiement de modèles ML, entièrement manuel. Il est considéré comme le niveau de maturité de base :

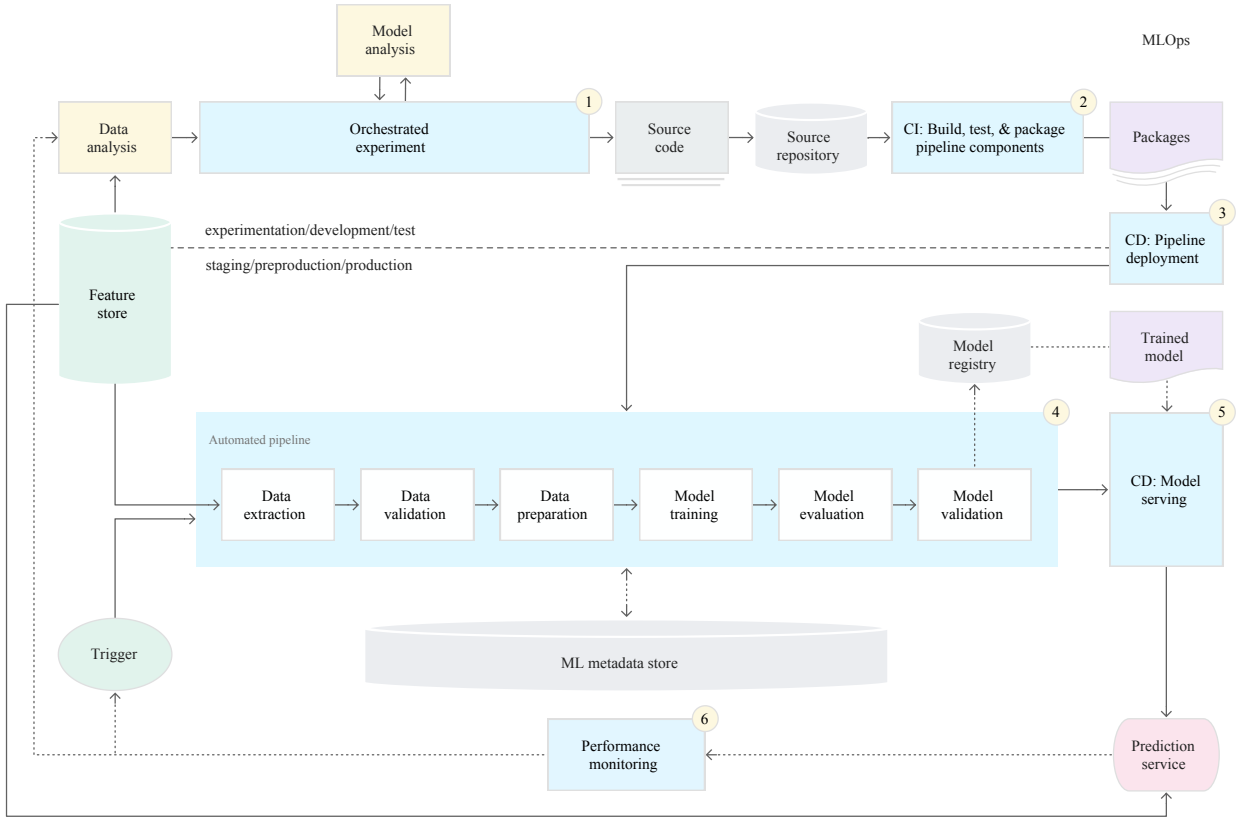


Comme le montre le diagramme ci-dessus, toutes les étapes allant de l'extraction des données et de l'analyse jusqu'à la production sont effectuées manuellement. Cette approche manuelle peut être suffisante lorsque les modèles ne changent pas ou ne sont pas formés fréquemment, cependant, dans le monde réel, les modèles ne cessent pas d'évoluer lorsqu'ils sont déployés et doivent être formés à nouveau. De plus, en adoptant cette stratégie, peu de versions seront réalisées en raison du temps nécessaire pour mettre le modèle en production.

Niveau 1 : Automatisation du pipeline ML Ce niveau comprend l'exécution automatique de l'entraînement du modèle. On introduit un nouveau concept : la formation continue, qui peut être réalisée en automatisant le pipeline ML. Cette approche permet des expériences rapides puisque la transition entre les différentes étapes est maintenant automatisée. En adoptant la formation continue, le modèle est automatiquement formé en production chaque fois que de nouvelles données sont disponibles. Ce niveau d'automatisation comprend également les étapes de validation des données et du modèle.

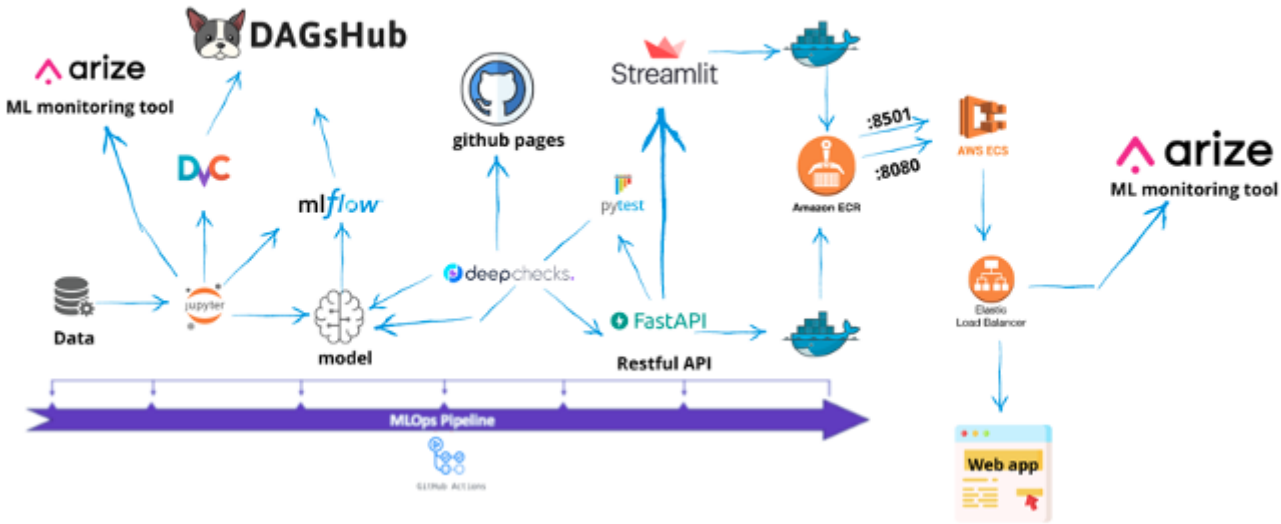


Niveau 2 : Automatisation du pipeline CI/CD Ce niveau implique une automatisation complète du système CI/CD qui met en oeuvre le pipeline d'apprentissage automatique. La principale différence par rapport à l'étape précédente est qu'on construit, teste et déploie maintenant automatiquement les données, le modèle ML et les composants du pipeline de formation ML.



4. Exemple d'architecture d'un système ML

Exemple d'arhcitecture MLOps qu'on va développer dans le cadre de notre bootcamp.



- **DagsHub** Outil aidant à héberger des expériences, à annoter des données, à reproduire des résultats d'expériences, à partager des fichiers, etc.
- **DVC** (Data Version Control) Data versioning
- **MLflow** Plateforme Open ML de Databricks
- **Pytest** Bibliothèque pour les tests
- **Deepcheks** Outil pour valider des données et les modèles (testing)

- **FastAPI** est un framework web Python hautement performant, open-source, utilisé pour développer des APIs Web
- **Streamlit** framework python permettant de créer des applications web qui pourront intégrer aisément des modèles de machine learning et des outils de visualisation de données
- **Docker** Plateforme de conteneurisation
- **Amazon ECR** (Elastic Container Registry) Service de registre d'images de conteneur
- **Amazon ECS** (Elastic Container Service) Service de gestion de conteneurs
- **Amazon ELB** (Elastic Load Balancer) Répartiteur de charge sur le cloud
- **Arize AI** plateforme d'observabilité ML pour surveiller, dépanner et expliquer les modèles

Références

[1] Google (2021) MLOps: Continuous Delivery and automation pipelines in Machine Learning) Google MLOps

[2] V. Lakshmanan, S. Robinson, M. Munn (2021) Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Building, and MLOps

[3] S. Alla, S.K. Adari (2021) Beginning MLOps with MLFlow: Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure

[4] N. Gift, A. Deza (2021) Practical MLOps: Operationalizing Machine Learning Models