

Industrialisation IA

Zacharie BUISSON - Mario DEFRENCE

Introduction



Agenda



Séance 1 Zacharie BUISSON

- Contexte et enjeux de l'industrialisation IA
- Rappels sur les bonnes pratiques de programmation
- Spécificités liées au déploiement d'IA
- Lancement des projets

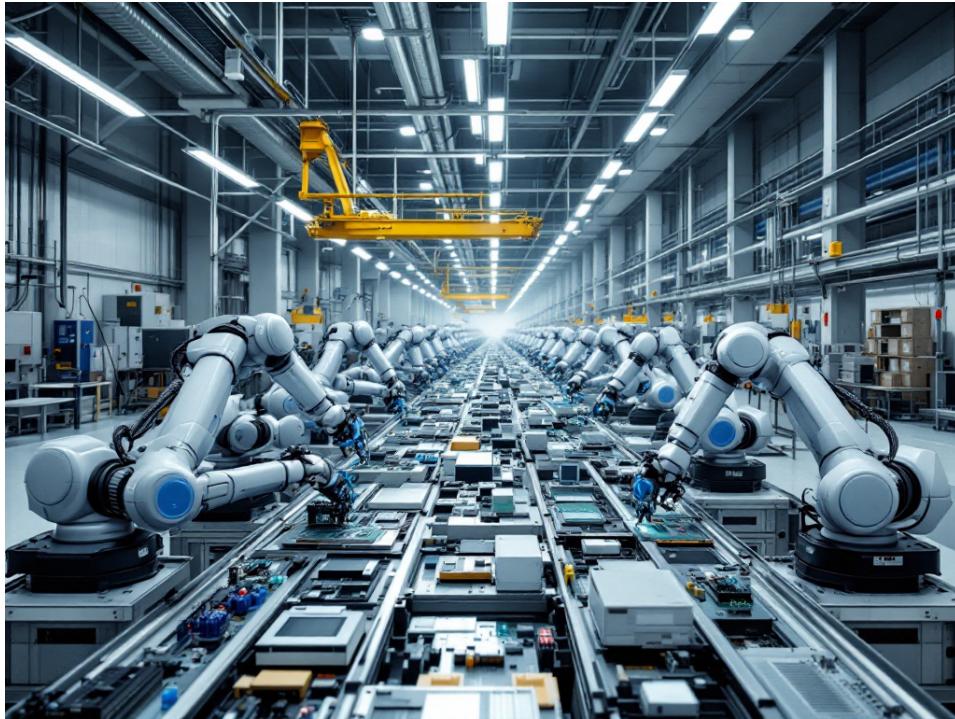
Séance 2 Mario DEFRENCE

- Continuation des projets
- Présentation du rendu attendu

Séance 3 Zacharie BUISSON & Mario DEFRENCE

- Finalisation des projets
- Soutenance des projets (15 min présentation + 10 min questions)

Contexte



- Vous avez construit votre modèle de Machine Learning / IA
➤ **Il ne sert à rien !**

Pour le valoriser, vous allez :

- le rendre disponible aux utilisateurs à travers une API
- mettre en place une chaîne d'industrialisation automatisée



Livraison d'un projet IA : 2 mois à 3 ans

Pour une même fonctionnalité, plus le temps de livraison est long:

- Plus les utilisateurs se tourneront vers un service concurrent :
 - ChatGPT interne vs ChatGPT externe
 - Problèmes de sécurité des données
 - Perte de clients
 - Déception des collègues qui préfèrent se tourner vers des prestataires externes, start-ups...
 - Plus il faut payer le data scientist longtemps...
 - ... plus le ROI du projet diminue
 - Plus il faudra de temps pour itérer et corriger les erreurs en production et répondre aux nouveaux besoins métiers...
- Nécessité d'automatiser l'industrialisation pour livrer le plus vite possible



L'intérêt de la solution reste à démontrer en production !

- Besoin de monitoring des performances
 - Est-ce que la solution est utilisée ?
 - Est-ce que la solution fonctionne ?
 - Est-ce que la solution est performante ?
- Besoin de feedback utilisateurs (pas forcément évident en IA...)
 - Biais de sélection: comment savoir si un prospect qui n'a pas pris un prêt chez nous l'aurait payé ?
 - Comment savoir si un utilisateur à utiliser la prédition de l'IA ? sans ajouter de surcharge ?

Infrastructure d'entreprise



Choix 1 : Architecture Cloud

Une architecture distante

+

 Avantages

- Utilisation à la demande (*pay what you use*)
- Services pré existants (*délégation software*)
- Pas de serveurs (*délégation hardware*)

-

 Inconvénients

- Dépendance à services externes : obligation de montée de version, suppression de service
- Protection des données, notamment extra UE



🇺🇸 Azure (Microsoft)



🇺🇸 AWS (Amazon)



🇫🇷 OVH



Choix 2 : Architecture on premise

Une architecture à la maison

+

 Avantages

- Flexibilité hardware & software : sur mesure aux besoins
- Indépendance aux organismes externes
- Sécurité gérée en interne, pas de problèmes de transit de données

-

 Inconvénients

- Mise à jour hardware et software à la main de l'entreprise : ajout de puissance de calcul manuel...
- Recrutement de personnel pour maintenir le système

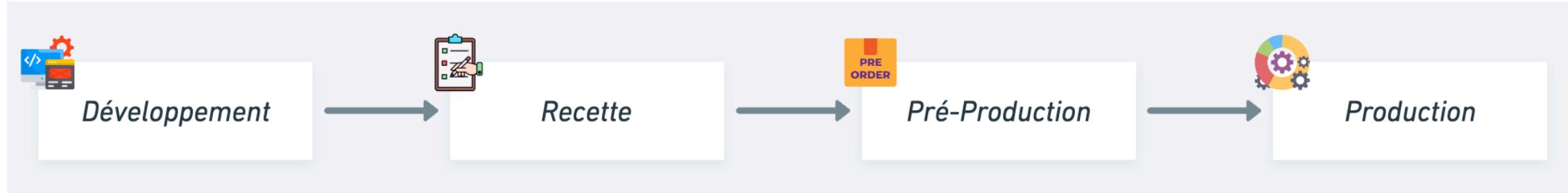


Docker

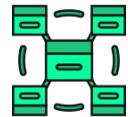


Kubernetes

Le cycle de vie d'un projet IT en production

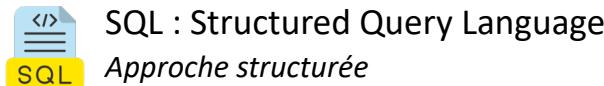


Data Scientist	Data Scientist, Dev Ops	Equipe Infrastructure (App Ops, Dev Ops)	Equipe Infrastructure (App Ops, Dev Ops)
Livraison d'une première version dans un environnement industriel	Livraison de l'ensemble des briques composants le système final	Vérifier que l'infrastructure répond correctement	Mise à disposition des utilisateurs
Instable	Moyennement stable	Stable	Stable
Tests d'intégration Tests de masse	Tests de masse Tests fonctionnel	Tests de charge Tests d'intrusion Tests fonctionnels	Tests d'intrusion Smoke tests



Bases de données relationnelles

Modèle historique dans les banques et assurances



SQL : Structured Query Language

Approche structurée

- Système polyvalent - rapide en lecture et écriture ;
- Pas d'unicité dans les observations ;
- Optimisé pour le brassage de données : jointure, calculs de variables en fonction de caractéristiques d'intérêt.



Oracle DataBase



PostgreSQ_L



NoSQL : Not only Structured Query Language

Approche non structurée

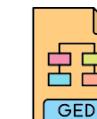
- Optimiser en lecture - pas écriture ;
- Accès aux données par clé (rapide), unicité des clés ;
- Utilisation sur de la donnée brute, en production.



Bases de données non relationnelles

Modèle novateur dans les banques et assurances

- Stockage documentaire : note de procédure interne, contrats client...
- Vidéo : caméra de surveillance, pièces de sinistres ;
- Voix : call center.



GED : Gestion Electronique
de Documents

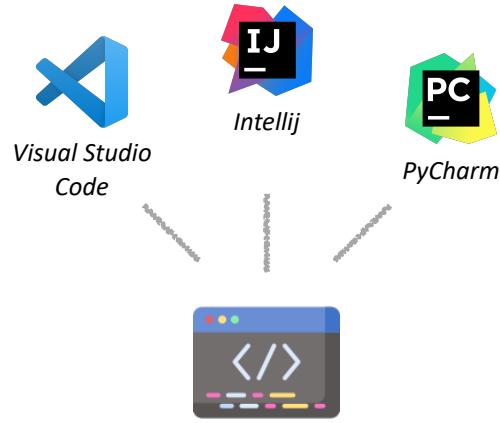


Amazon S3

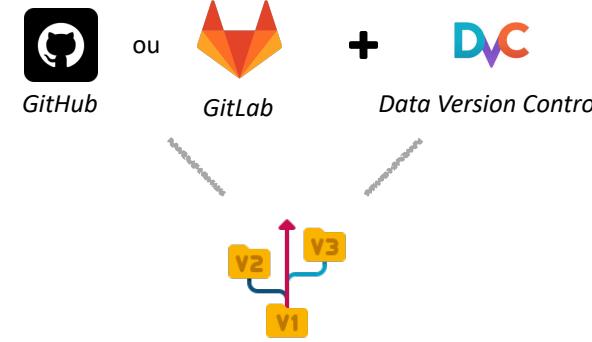


Programmation, bonnes pratiques et comment construire un logiciel utile

Quelques étapes pour la réussite d'un projet IA en collaboration



Utilisation d'un IDE
But : Coder avec des avantages



Versionner son code et ses données
But : Collaborer et retracer l'histoire de ceux-ci



Définition d'un environnement de travail
But : Avoir les mêmes fonctionnalités



Tracer ses expérimentations de ML
But : Historiser les résultats des tests

mlflow™



PEP8 : Python Enhancement Proposal

Norme de programmation Python

Ensemble de pratiques pour faire du code dans les standards Python

- Respecter la `snake_case` ;
- Faire des doctrines ;
- Ne pas laisser d'éléments non utilisés...



But : Normalisation du code, possibilité de le rendre compréhensible de tous...



Programmation Orientée Objet

Synthèse de fonctions pour suivre dans le temps



Exemple d'object : Client de Banque

Attributs : Montant sur le compte, nom, prénom



Objet hérité : Client crédit agricole

Attributs : Nom de l'agence, caisse régionale...



But : Regrouper et normaliser des éléments avec des caractéristiques communes, suivre l'état d'un élément...



Programmation SOLID

Comment construire une bonne fonction ?

Single Responsibility Principle

Chaque fonction doit avoir une seule fonctionnalité

Open / Closed Principle

Ouvert à l'extension, fermé à la modification

Liskov Substitution Principle

Les classes héritées doivent avoir les méthodes de la classe mère fonctionnelles

Interface Segregation Principle

Une classe ne doit avoir que des fonctions utiles à sa classe

Dependency Inversion Principle

Non détaillé, pas le plus important...



But : Avoir des fonctions simples à tester, limiter les erreurs...

3 principaux modes de livraison de code



Packaging de code

Mise à disposition du code en ligne



Code brut livré dans un répertoire git
Construction d'une Wheel
Déploiement de la wheel en ligne :

- PyPI
- Conda Forge
- Artifactory interne



But : Utilisation et installation d'un code à la demande et à la main d'un utilisateur.



Batching & CronJob

Publier une application à lancement auto.



Code brut livré dans un répertoire git
Encapsulation dans un outil

- CronJob
- Lambda + EventBridge sur AWS
- DataBricks
- Linux : Cron
- Windows : Task Scheduler



But : Déploiement / installation en production, lancement automatisé chaque jour, à intervalle régulière, envoi notification succès / échec.



API

Publier une application en temps réel



Code brut livré dans un répertoire git
Construction d'une API
Encapsulation dans un conteneur Docker
Déploiement du docker sur un serveur

- ArgoCD (*on premise*)
- AWS ECS (*cloud*)
- Azure Machine Learning



But : Déploiement / installation en production, application disponible pour un appel en temps réel.

Zoom sur le fonctionnement d'une API



Comment Interagir avec une API ?

API et requêtes..

Quatre types de requêtes principales :



GET

Sert à récupérer des informations directement



POST

Sert à envoyer une information (sensible) à l'application



PUT

Enregistrer une donnée dans une BDD ou la mettre à jour



DELETE

Supprimer une donnée dans une base de données

Documentation : OpenAPI (swagger)

Fichiers .yaml, ou .json qui décrit les façons possibles pour interagir avec un API

health

GET /1.0.0/health_check Checks health of API

claim_extraction

POST /1.0.0/entities/{uom_code}/customers/{customer_id}/contracts/{contract_id}/claim_files_extraction Extracts info from claim files

PUT /1.0.0/entities/{uom_code}/customers/{customer_id}/contracts/{contract_id}/save_extract Saves claim data in AWS database

Schemas

Status >

Invoice >

Address >

BankDetails >

Comment livrer un logiciel



Créer un plat avec une recette de cuisine

Ecrire la recette de cuisine

Décrire les étapes permettant de créer la recette.



Créer une application avec Docker

Ecrire la recette de l'application

Décrire les étapes permettant de reconstruire l'application (DockerFile)

Commande : pas de commande

Il faut juste rédiger un DockerFile

Faire la recette de cuisine

Exécuter les étapes une à une pour construire un plat.

Faire la recette de l'application

Exécuter les étapes une à une pour construire l'application. (Image Docker)

Commande : docker build

Peut être une image stockée en local ou dans un répertoire en ligne : artifactory, Docker Hub, AWS ECR.

Avoir un plat préparé

Avoir un plat à disposition, prêt à être utilisé.

Avoir une application préparée

Avoir une application déployée, dans un espace isolé et performant. (Conteneur)

Commande : docker run

Peut s'utiliser en local ou sur un serveur avec Kubernetes

Points Critiques à surveiller :



Gestion de la charge de l'application
Comment gérer un afflux d'utilisateurs ?



Redémarrer l'application si échec
Cela permet de maintenir une application fonctionnelle



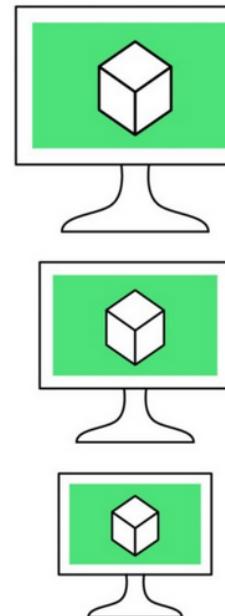
Gérer les mises à jour
Comment mettre à jour une application sans tout couper ?



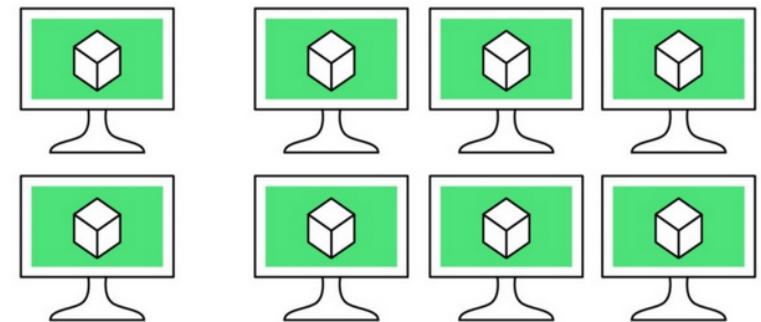
Tester mon application
Comment s'assurer que mon application soit correcte ?

Scalability

Vertical scaling



Horizontal scaling



CI

Intégration continue

Comment intégrer efficacement des modifications dans une application de production ?

Evaluation de la qualité de code



Solutions Open Source

- *Flake8 : Formateur de code*
- *Black : Formateur de code*
- *Ruff : Formateur de code (rapide)*
- *PyLint : Formateur de code*
- *Isort : Formateur des imports*



But : S'assurer que le code de l'application sera compris par tous.



But : S'assurer que le code fonctionne et que l'on ne « casse » rien qui fonctionnait.

Tests unitaires

- Vérifier que chaque fonction de son code marche correctement
- Tester sur des cas classiques et des cas limites.



Construction des artefacts

- Construction de l'image contenant l'API ou d'un .zip ;
- Stockage de l'artefact créé dans un répertoire.

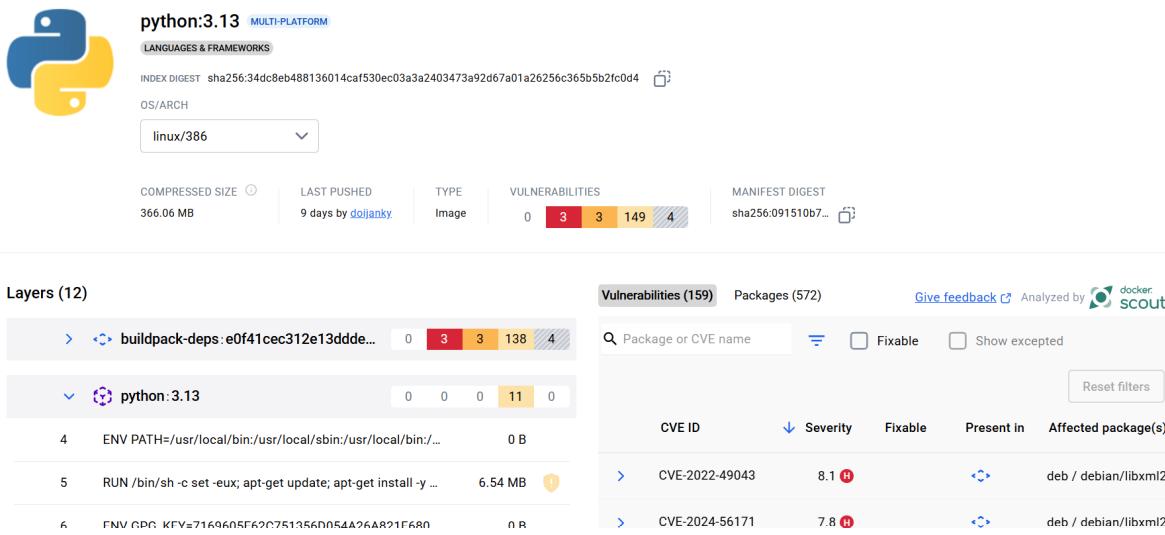


DOCKER



But : Faciliter et normaliser la livraison du code.

1. tests de sécurité Xray CVE : vulnérabilités de l'application + test d'injection de code dans les
2. déploiement de l'application dans un environnement
3. promotion vers environnements ultérieurs de DEV à PROD



O'REILLY®

Deploy First, Pray Later

god abandoned this
pipeline long ago.

Some Random guy on X
@observer_ofyou



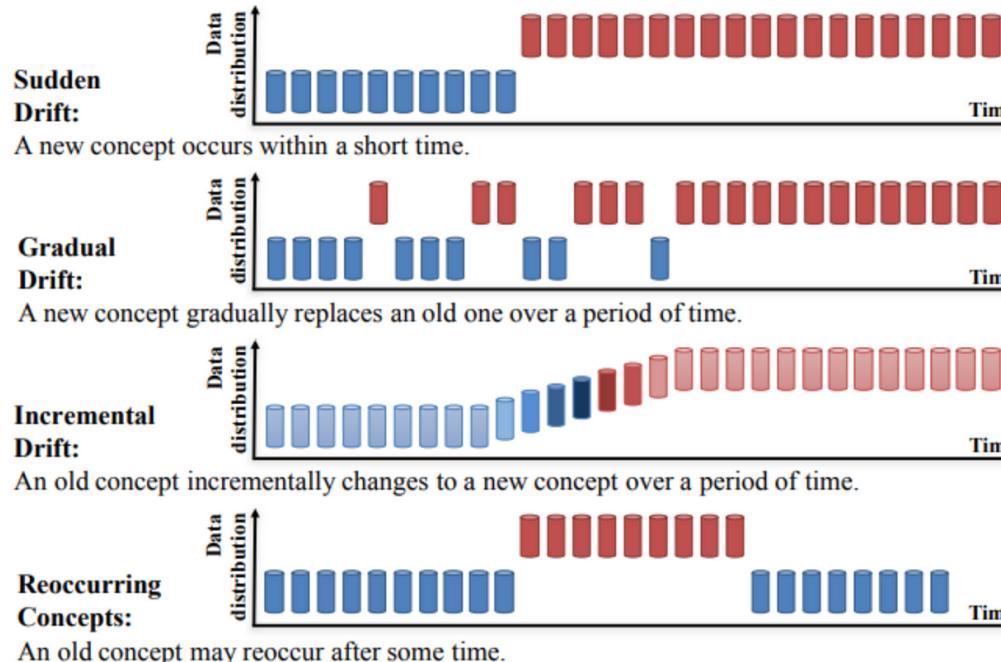
Application à l'IA

MLOps : suivre les performances de l'IA



Suivi et monitoring du drift en production

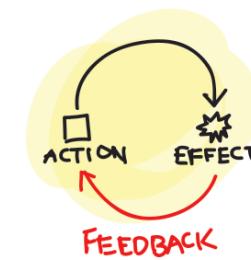
Ne pas laisser son application dépérir en production...



Source: <https://arxiv.org/pdf/2004.05785.pdf>

Recueillir des retours des utilisateurs :

*Pas possible : un fraudeur ne va pas faire un feedback
A prendre en compte dès la conception du projet*



Surveiller la cohérence statistique d'un modèle d'IA / ML :

*Attention aux données plus représentatives de la réalité
Nécessite un ré-entraînement ou des correctifs.*

MLOps : pipeline d'IA



Pipelines d'IA

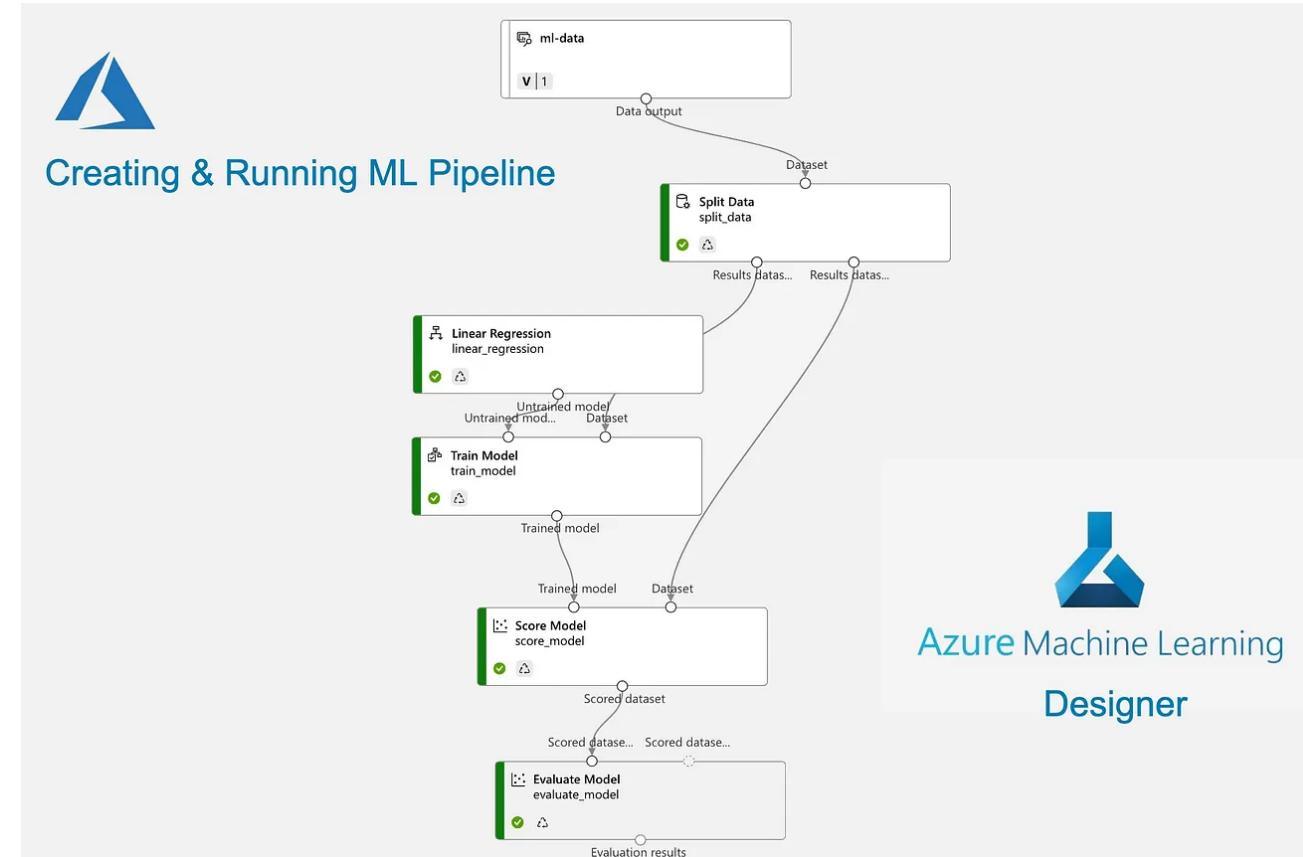
Entrainement automatique de modèles

- Script d'entraînement d'un modèle d'IA, prévu de A à Z
- Peut être déclenché automatiquement ou à dates régulières
- Permet d'itérer rapidement et de créer de nouveaux cas d'usage similaires



Script d'entraînement

Création de blocs thématiques réutilisables au besoin



Azure Machine Learning
Designer