INTÉGRATION CONTINUE ET ARCHITECTURE À BASE DE CONTENEURS POUR EXPOSER UN MODÈLE D'APPRENTISSAGE PROFOND COMME UN SERVICE

Théophile Tiffet [1], Guillaume Guerdoux [2], Cédric Bousquet [1,3]

[1] Service de santé publique et information médicale, CHU de Saint Etienne, France

[2] Geegz, 18 Pass. du Chantier, 75012, Paris, France

[3] Sorbonne Université, Inserm, université Sorbonne Paris Nord, Laboratoire d'informatique médicale et ingénierie des connaissances en e-santé, LIMICS, F-75006 Paris, France

Journée Santé et IA – Plate-Forme Intelligence Artificielle - Saint-Etienne, France – 28 juin 2022 -

INTRODUCTION

- Performances remarquables de l'apprentissage profond en médecine
- Nombreux modèles issus de travaux de recherche non disponibles en pratique clinique
- Défis liés à la mise en production de solutions d'intelligence artificielle difficiles à évaluer
- Différence entre les problématiques à résoudre pendant
  - La phase de recherche et développement : optimisation du travail collaboratif d'implémentation entre les chercheurs
  - La phase de déploiement : mise à disposition du modèle auprès des utilisateurs finaux, facilitation des mises à jours et de la maintenance

PROBLÉMATIQUE PLUS GÉNÉRALE DU DÉPLOIEMENT D'ALGORITHME

- Hypothèse de Wilson en 2006 sur les difficultés liées au calcul scientifique
  - Méconnaissance des bonnes pratiques dans le développement de code informatique chez les chercheurs
  - Principal exemple : absence de tests unitaires et logiciel de gestion des versions
  - Meilleure efficacité à coder en faisant appel à certains outils largement utilisés dans l'industrie du logiciel
- Même point de vue en 2013 par Zaytsev et coll., dans le cadre du développement de logiciels en neuroinformatique
  - Afin de gérer la taille et la complexité croissante de projets logiciels, recommandation d'une approche d'intégration continue

SOLUTIONS PROPOSÉES

- Architecture à base de microservices
  - Eviter les inconvénients liés à la complexité des grosses applications monolithiques
  - Composants avec une tâche spécifique qui n'interfère pas avec le fonctionnement des autres composants
  - Isolation à l'aide de conteneurs Docker
- Déploiement de correctif ou de mise à jour en minimisant le risque d'introduire des erreurs.
  - Processus d'automatisation de l'intégration de nouveaux segments de code, répétition automatique des tests, de déploiement automatique désignés par l'expression "pipeline CI/CD" (*continuous integration / continuous deployment*)

OBJECTIF

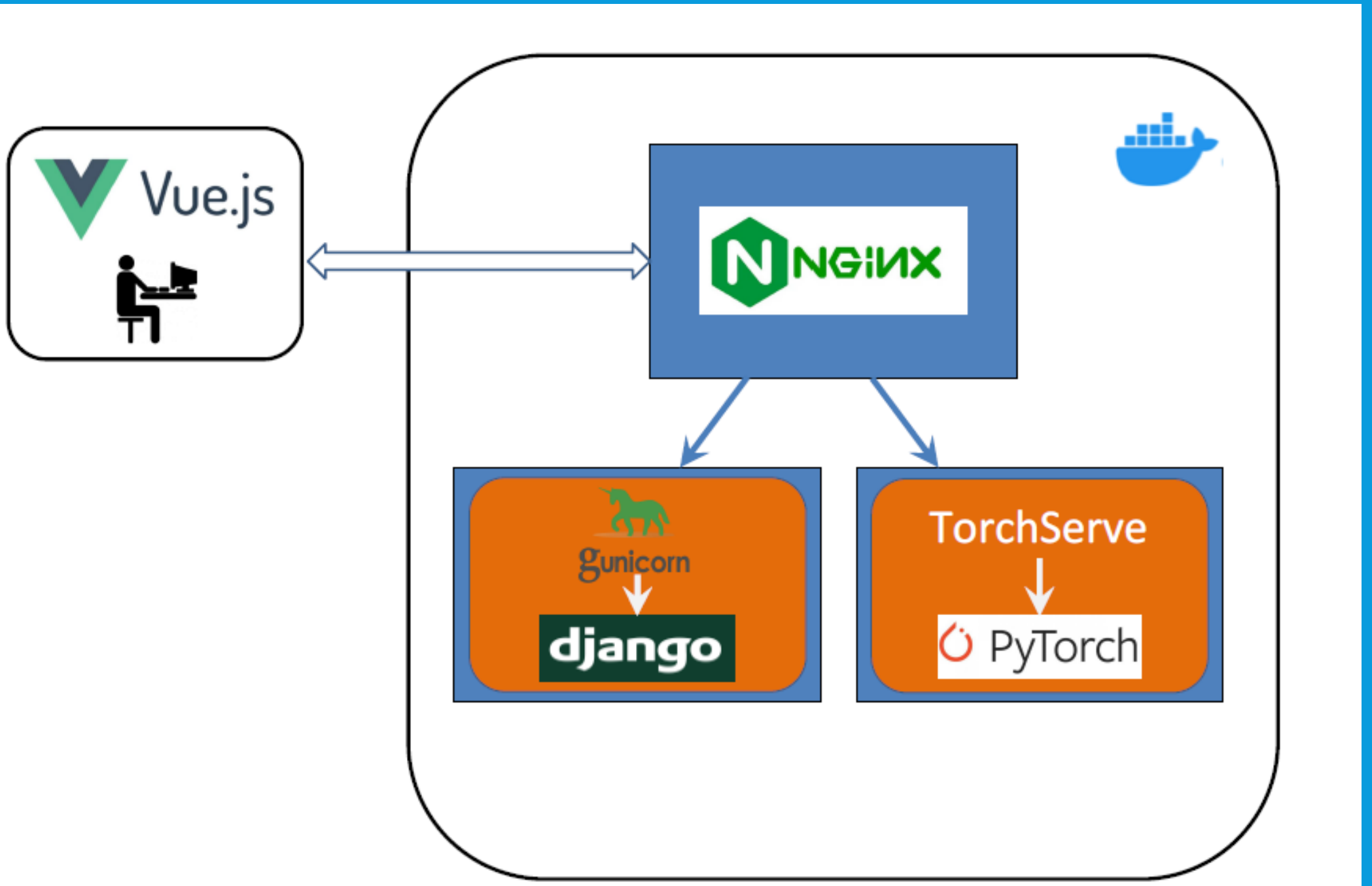
- Proposer un processus de déploiement de bout en bout d'un algorithme d'apprentissage profond, facilitant la mise à disposition d'un modèle précédemment entraîné
- Définir la meilleure manière de procéder pour
  - Mettre à disposition de manière efficace et générique un modèle d'apprentissage profond
  - Intégrer des développements successifs du code source et déployer automatiquement ces modifications

MATÉRIEL

- A titre d'exemple, modèle de classification d'opinion d'utilisateurs de Twitter sur la vaccination, entrainé en 2020 avant le début de la campagne de vaccination contre le COVID- 19 en France
  - Réseau de neurones profond implémenté avec Pytorch et CamemBERT (modèle disponible dans la librairie Transformers d'Hugging Faces)
  - 4548 tweets écrits en français en lien avec le COVID-19, identifiés par le PanaceaLab et mis à disposition sur son Github, annotés manuellement

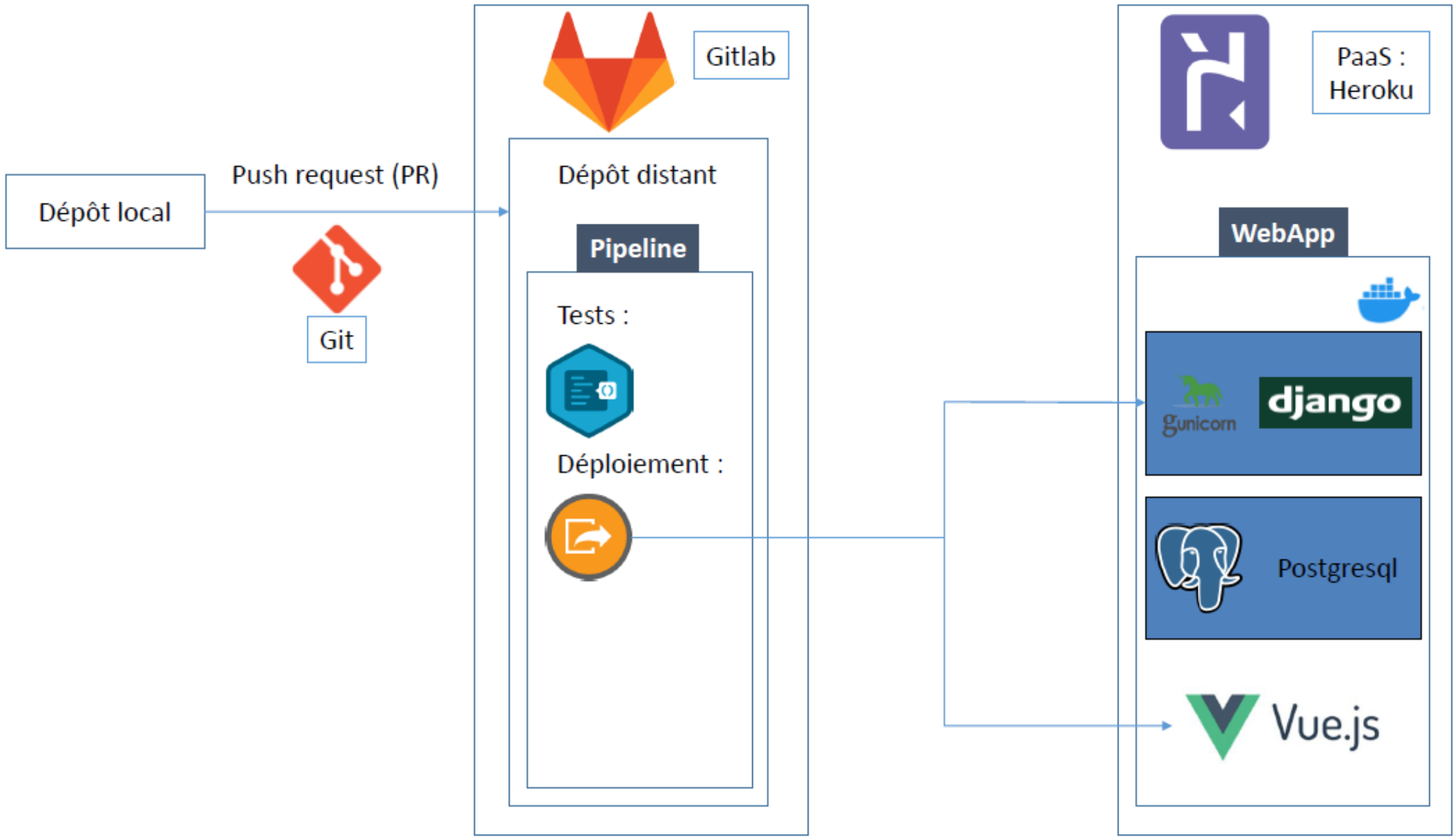
TORCHSERVE POUR LE DÉPLOIEMENT DES MODÈLES PYTORCH

- Librairie issue d'une collaboration entre Amazon Web Services et Facebook
- Latences faibles permettant d'obtenir des inférences à haute performance
- Gestionnaires par défaut pour les applications les plus courantes : détection d'objets et classification de textes
  - Fichier, appelé *handler*, décrivant à TorchServe le modèle et la manière dont il convient de l'utiliser
- *Application programming interface* pour communiquer avec d'autres applications


Vue.js
NGINX
gunicorn
django
TorchServe
PyTorch

INTÉGRATION ET LIVRAISON CONTINUES

- Architecture précédente adaptée à un contexte de recherche ou de développement, mais assez limitée une fois l'application déployée
  - Compliqué de publier des mises à jours et des correctifs
  - Nécessité de valider et tester toutes les modifications apportées avant de publier
- Avec le CI/CD, possibilité de détecter de façon plus précoce les erreurs en testant automatiquement et systématiquement le code
  - Outil de gestion de version comme GitLab, qui permet aux chercheurs de développer et tester en local avant d'envoyer ces modifications sur un dépôt distant
  - Meilleure productivité et réduction des temps de mise à disposition de nouvelles versions de l'application


Gitlab
PaaS :
Heroku
Push request (PR)
Dépôt local
Dépôt distant
Git
Pipeline
Tests :
Déploiement :
WebApp
gunicorn
django
Postgresql
Vue.js

COMPARAISON ENTRE L'APPROCHE MICROSERVICES ET LE CI/CD

Fonctionnalité	Microservices	Approche CI/CD
Authentification	Oui	Oui
Installation en local	Oui	Non
Intégration continue	Non	Oui
Livraison continue	Non	Oui
Déploiement de l'application	Import des sources sur le serveur (git clone), déploiement avec Docker compose	Déploiement automatique de l'application en cas de succès des tests unitaires
Déploiement du modèle	TorchServe déployé avec Docker compose, modèle mis à jour manuellement	TorchServe et modèle déployé manuellement

DISCUSSION

- Résultats préliminaires et limités à l'implémentation des deux approches
  - Faisabilité de l'implémentation d'une architecture à base de microservices pour le déploiement d'un modèle d'apprentissage profond
  - Résultats insuffisante pour conclure sur la pertinence et la possibilité de généralisation de la démarche proposée dans un contexte de recherche
- Architecture cependant assez flexible pour être modifiée et répondre à diverses exigences
  - Ex : modification du microservice Django pour une authentification via le protocole LDAP (Lightweight Directory Access Protocol)
- Peu de retour d'expérience et aucune recommandation pour les meilleures pratiques

DIFFICULTÉS TECHNIQUES ASSOCIÉES AU DÉPLOIEMENT DE MODÈLES D'APPRENTISSAGE PROFOND DANS LA LITTÉRATURE

- Manque d'intérêt des intervenants sur le sujet
  - Cliniciens peu intéressés par les aspects techniques du déploiement
  - Chercheurs en intelligence artificielle plus intéressés par l'optimisation du modèle qu'à sa mise en production
- Autres difficultés potentiellement plus grandes
  - Manque de confiance des médecins dans les algorithmes, difficulté à les intégrer dans la pratique clinique ou à les expliquer
  - Questions juridiques et éthiques liées à l'utilisation des données
  - Manque de généralisation du modèle
- Cependant, sous-estimation probable de la difficulté technique du déploiement du modèle et son exposition sous la forme d'un service

LIMITES

- Pas de prise en compte des contraintes liées à la production telles que la mise à l'échelle, la gestion des versions du modèle, et la vérification de la stabilité des prédictions dans le temps
  - Par exemple, défi des tweets qui peuvent être produits en quantité en nombre variable au cours du temps
- Nécessité d'une évaluation par des indicateurs objectifs pour mesurer sa performance
  - Exposition de plusieurs versions de déploiement de l'algorithme au flux de données de Twitter afin de vérifier quel déploiement fonctionne le mieux

DEVOPS ET MLOPS

- Architecture à base de microservices partie de l'approche DevOps qui vise à fusionner les activités de déploiement et d'opérationnalisation
  - Machine Learning Operations (MLOps) dans le cas de l'intelligence artificielle
  - Gestion des modèles possible avec MLflow
- Possibilité d'utiliser Kubernetes, un orchestrateur de conteneurs
  - Nombreux avantages par rapport à l'utilisation de conteneurs Docker sans orchestration, au prix d'une plus grande complexité
  - Intégré dans Kubeflow, un environnement pour le MLOps
- Comparaison des temps nécessaires à la mise à jour de l'application avec et sans les outils de DevOps

CONCLUSION

- Première étape pour un programme de recherche sur les meilleures pratiques liées au déploiement d'algorithmes d'apprentissage profond utilisant DevOps et MLOps, leurs avantages et leurs inconvénients
  - Base pour de futures comparaisons avec d'autres types d'architectures
  - Travaux supplémentaires nécessaires pour émettre des recommandations
- Intérêt de développer une solution "clés en main"
  - Outil facilitant et automatisant l'utilisation d'une architecture en microservices dans un projet
  - Permettant une transition vers le déploiement une fois le modèle prêt à être utilisé

MERCI POUR VOTRE ATTENTION

Théophile Tiffet Theophile.Tiffet@chu-st-etienne.fr

Guillaume Guerdoux guillaume.guerdoux@geegz.fr

Cedric Bousquet cedric.bousquet@chu-st-etienne.fr