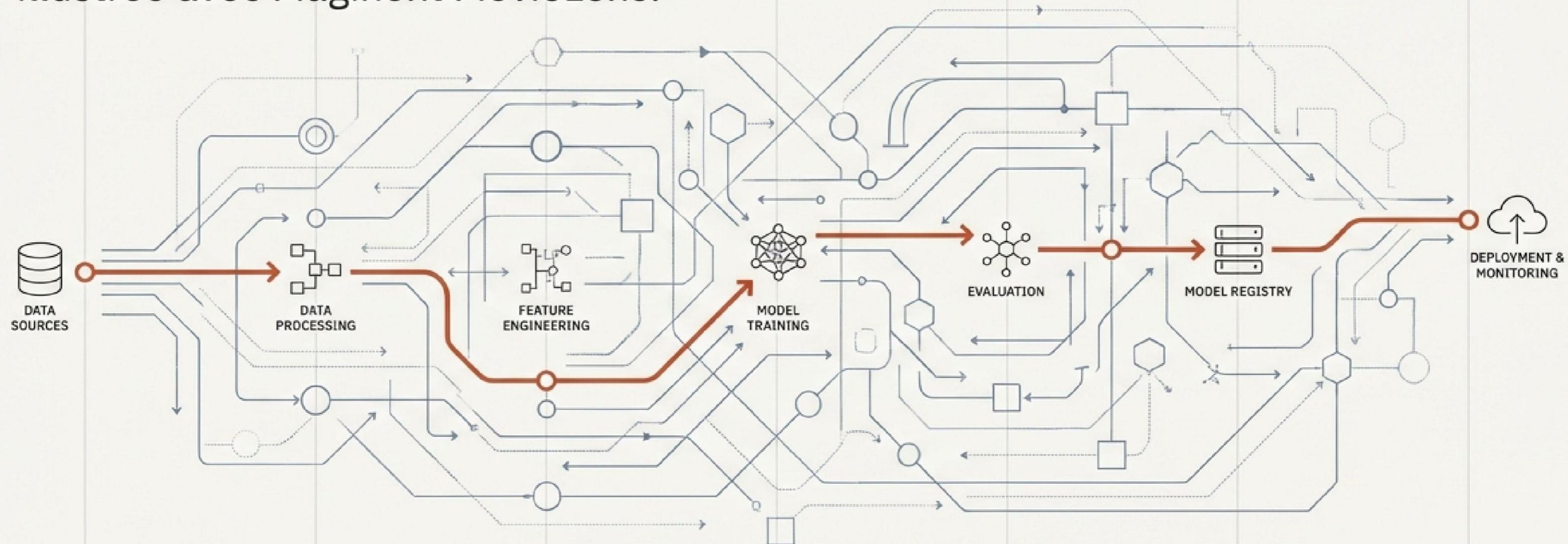


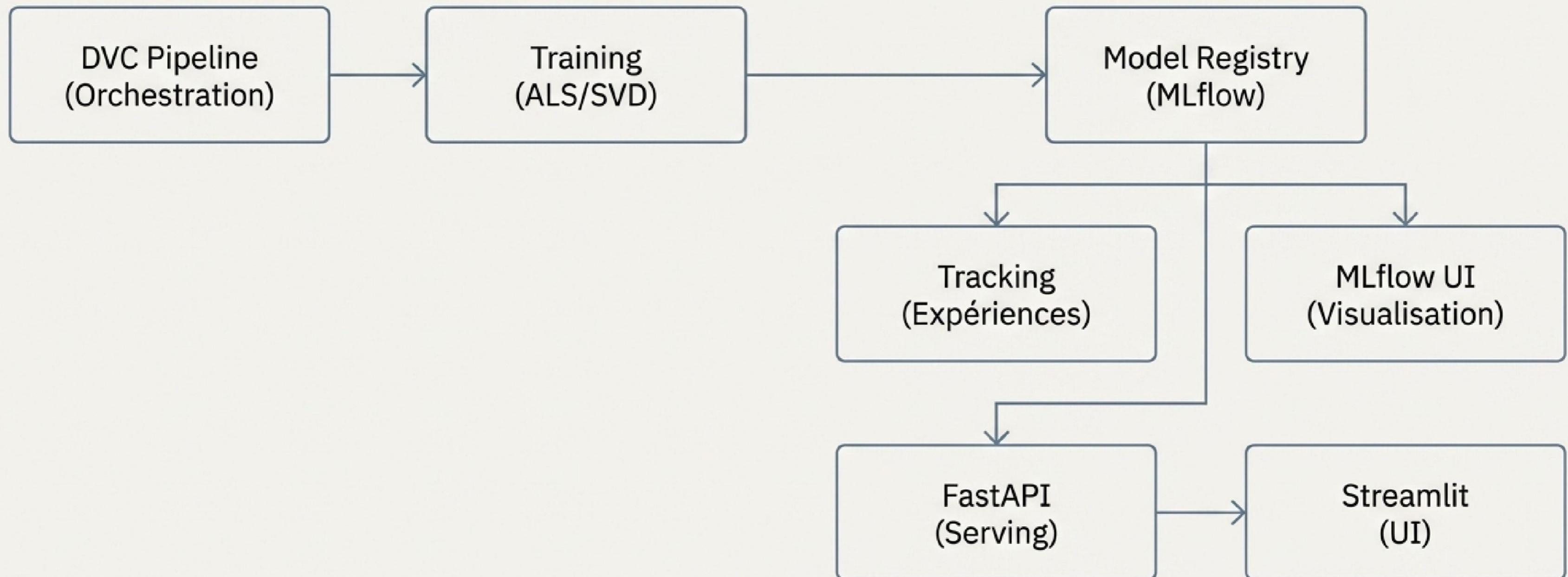
Un Blueprint MLOps pour les Systèmes de Recommandation

De la Donnée au Déploiement : une architecture de production complète illustrée avec Maginvent MovieLens.



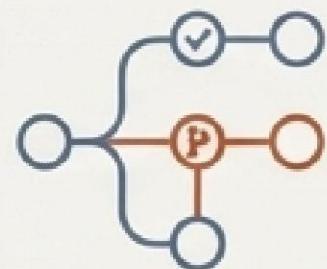
Notre Itinéraire : L'Architecture du Système

Ce document est une visite guidée d'un système de recommandation de bout en bout. L'architecture ci-dessous est notre carte. Nous explorerons chaque étape, des données brutes à l'interface utilisateur interactive.



La Trousse à Outils MLOps du Projet

Reproductibilité & Orchestration



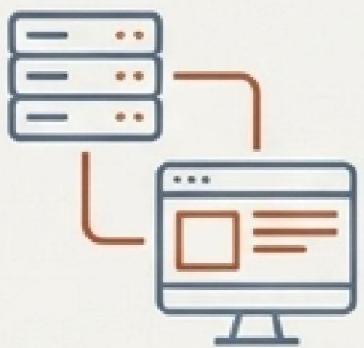
- **DVC:** Pour garantir la reproductibilité complète du pipeline de données et des modèles.

Expérimentation & Suivi



- **MLflow:** Pour le suivi des expériences et la gestion centralisée des modèles (Model Registry).
- **Optuna:** Pour l'optimisation systématique des hyperparamètres.

Déploiement & Interface



- **FastAPI:** Pour servir le modèle via une API REST performante.
- **Streamlit:** Pour créer une interface utilisateur interactive et explorer les recommandations.

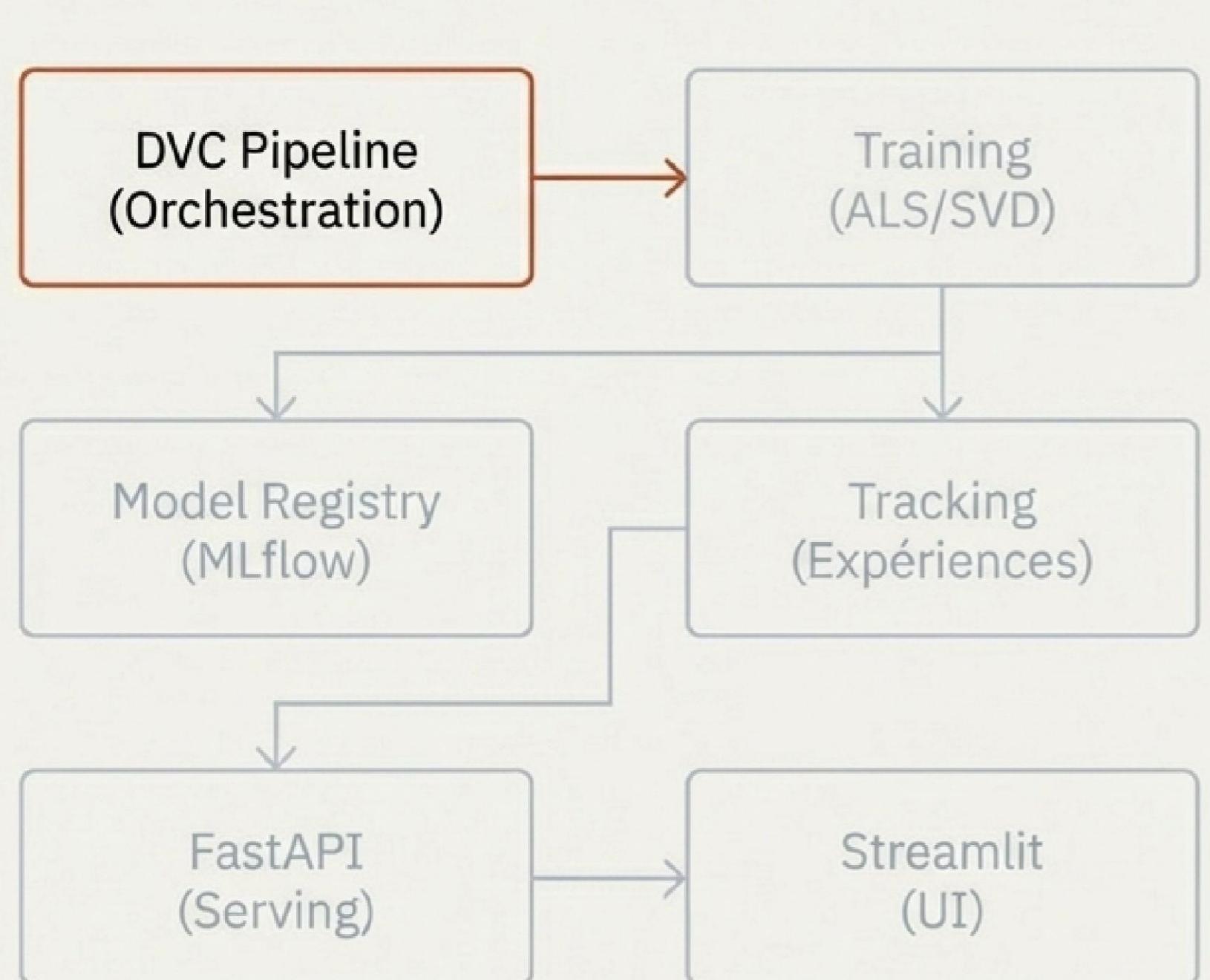
Automatisation & Production



- **GitHub Actions:** Pour l'intégration et le déploiement continus (CI/CD).
- **Docker Compose + Render:** Pour la conteneurisation et le déploiement cloud.

Étape 1 : La Lignée des Données

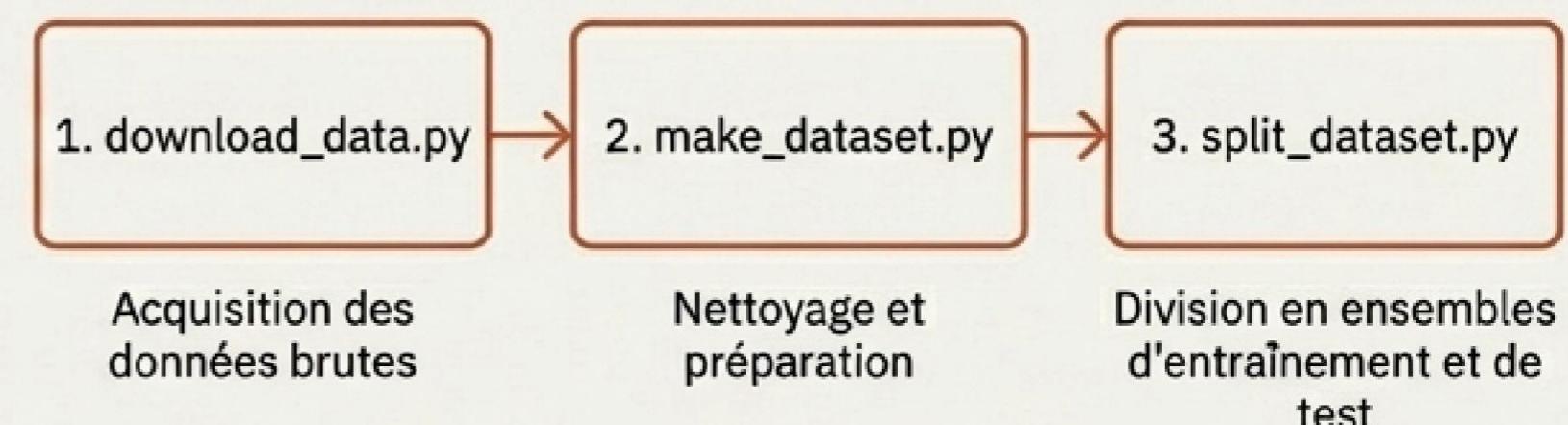
Assurer la reproductibilité avec DVC



Le Rôle de DVC

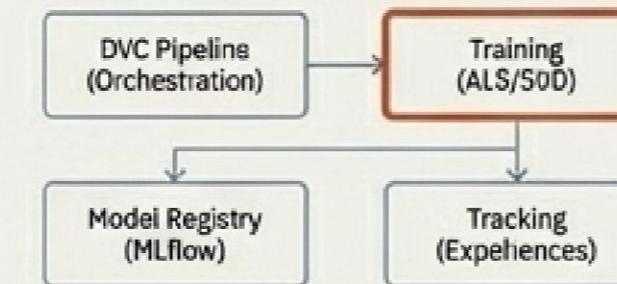
Le pipeline est défini dans `dvc.yaml`. DVC versionne les données et les étapes de traitement, garantissant que chaque exécution, du téléchargement à la division des données, est 100% reproductible.

Flux de Données



Étape 2 : Le Laboratoire de Modélisation

De la Baseline à la Factorisation de Matrice



Popularité (Baseline)



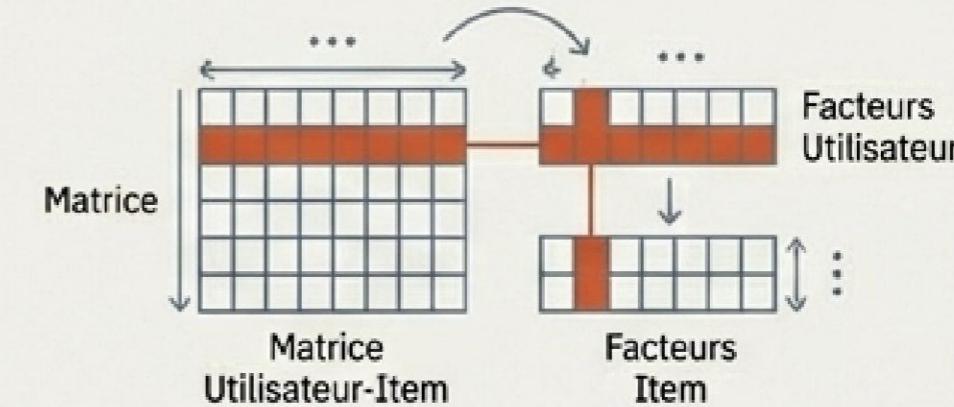
Objectif

Recommande les items les plus populaires à tous les utilisateurs.

Rôle Stratégique

Sert de référence indispensable pour évaluer la performance réelle des modèles plus sophistiqués. Si un modèle complexe ne bat pas cette baseline, il n'apporte pas de valeur.

ALS (Alternating Least Squares)



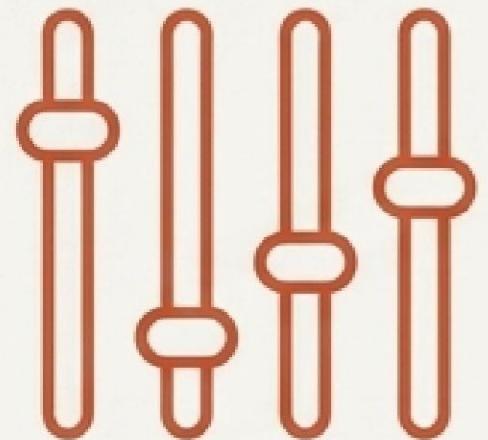
Mécanisme

Algorithme de factorisation matricielle optimisé pour le feedback implicite. Il décompose la matrice utilisateur-item en facteurs latents pour découvrir des préférences cachées.

Avantage

Capture des relations personnalisées au-delà de la simple popularité.

Optimiser ALS avec Rigueur et Automatisation



Les Levier de Performance d'ALS

- **embedding_dim**: Dimension des facteurs latents (vecteurs user/item).
- **regularization**: Régularisation L2 pour prévenir le surapprentissage.
- **iterations**: Nombre de passages pour la convergence de l'optimisation.
- **alpha**: Paramètre de confiance pour le feedback implicite.

L'Optimisation avec Optuna

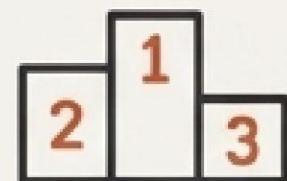
Plutôt que de choisir manuellement, nous utilisons Optuna pour explorer l'espace des hyperparamètres de manière systématique.

```
optuna:  
    enabled: true  
    n_trials: 20  
    metric: ndcg_at_10
```

Le système exécute 20 essais pour maximiser la métrique **ndcg_at_10**, garantissant que nous sélectionnons la meilleure configuration de modèle de manière data-driven.

Étape 3 : Le Centre de Contrôle Qualité

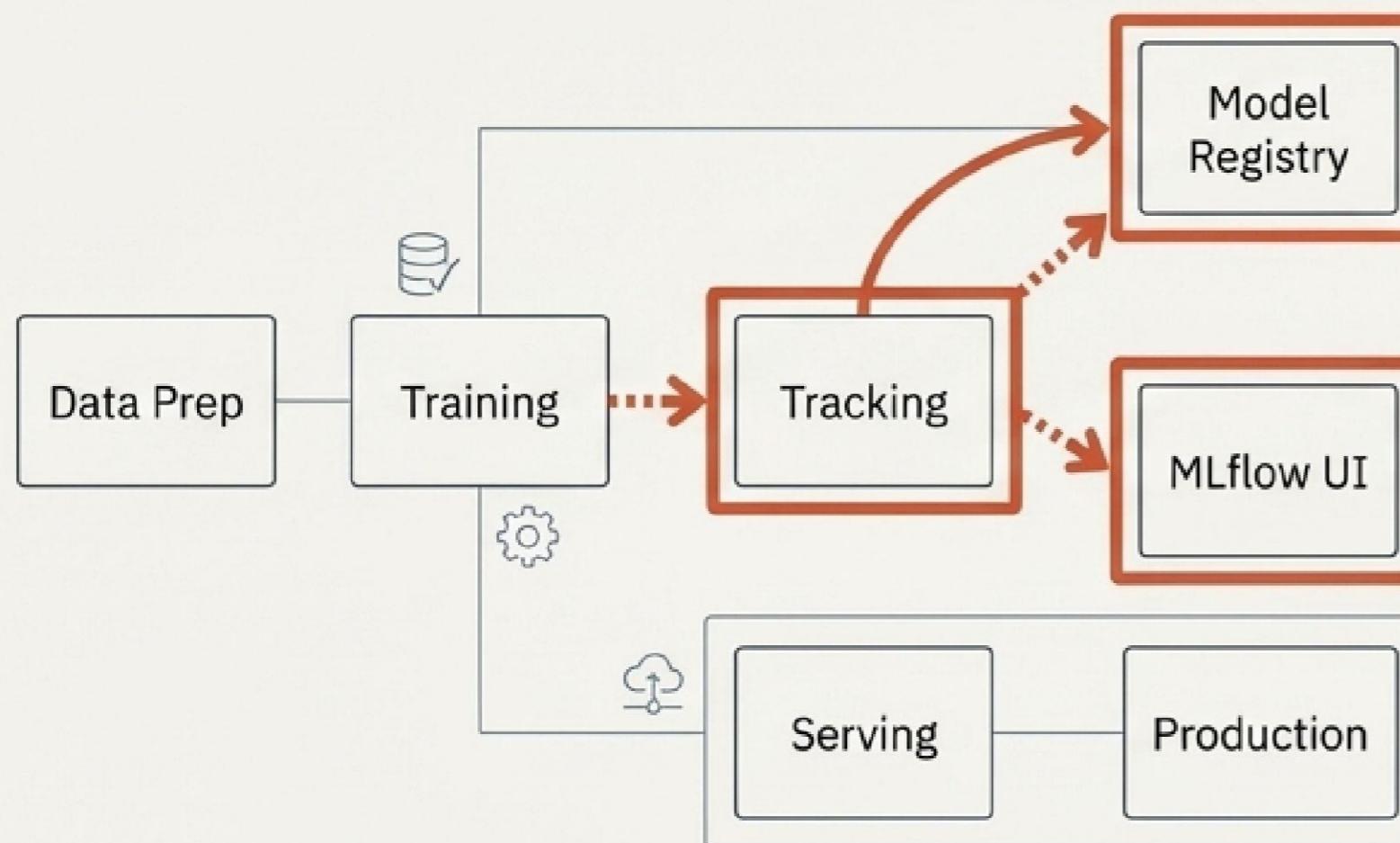
Mesurer la performance des recommandations



Métrique	Description
Precision@K	Proportion d'items pertinents parmi les K premiers recommandés.
Recall@K	Proportion d'items pertinents que le modèle a réussi à retrouver.
NDCG@K	Qualité du classement, pénalisant les bons items mal positionnés.
MAP@K	Précision moyenne, tenant compte de l'ordre des recommandations.
MRR	Mesure la position du premier item pertinent recommandé.

Étape 4 : Archiver et Versionner nos Modèles

MLflow comme Registre Central



Double Fonction de MLflow

1. Suivi d'Expériences

Chaque entraînement est enregistré comme une ‘expérience’ MLflow. Paramètres, métriques, et artefacts (comme le modèle lui-même) sont automatiquement versionnés.

2. Registre de Modèles

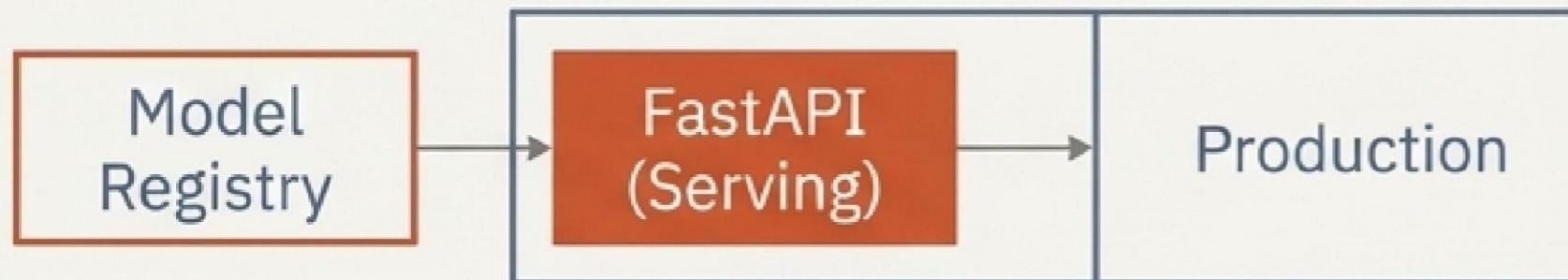
Les modèles les plus performants sont ‘promus’ dans le Registre. Cela crée un catalogue versionné des modèles prêts pour la production, avec des étapes claires (Staging, Production).

Bénéfice

Cette approche garantit une traçabilité complète et permet des déploiements et des rollbacks fiables.

Étape 5 : L'Usine de Déploiement

Servir le modèle via une API REST avec FastAPI



Méthode	Endpoint	Description
GET	/health	Vérifie l'état de santé et la disponibilité du service.
POST	/recommend	Génère des recommandations pour un utilisateur donné.
POST	/similar-items	Trouve les items les plus similaires à un item donné.
GET	/users	Retourne la liste des utilisateurs disponibles.
GET	/items	Retourne la liste des items disponibles.

Interagir avec le Modèle en Production

Exemple de requête à l'endpoint `/recommend`

```
# Demande les 10 meilleures recommandations pour l'utilisateur avec l'ID 1
curl -X POST http://localhost:8000/recommend \
-H "Content-Type: application/json"
-d '{
  "user_id": 1,
  "k": 10
}'
```

L' API est conçue pour être simple et stateless. Une simple requête POST avec l'ID de l'utilisateur et le nombre de recommandations souhaitées (`k`) suffit pour obtenir une réponse.

Étape 6 : La Vitrine Utilisateur

Exploration interactive avec Streamlit

Rôle de l'UI

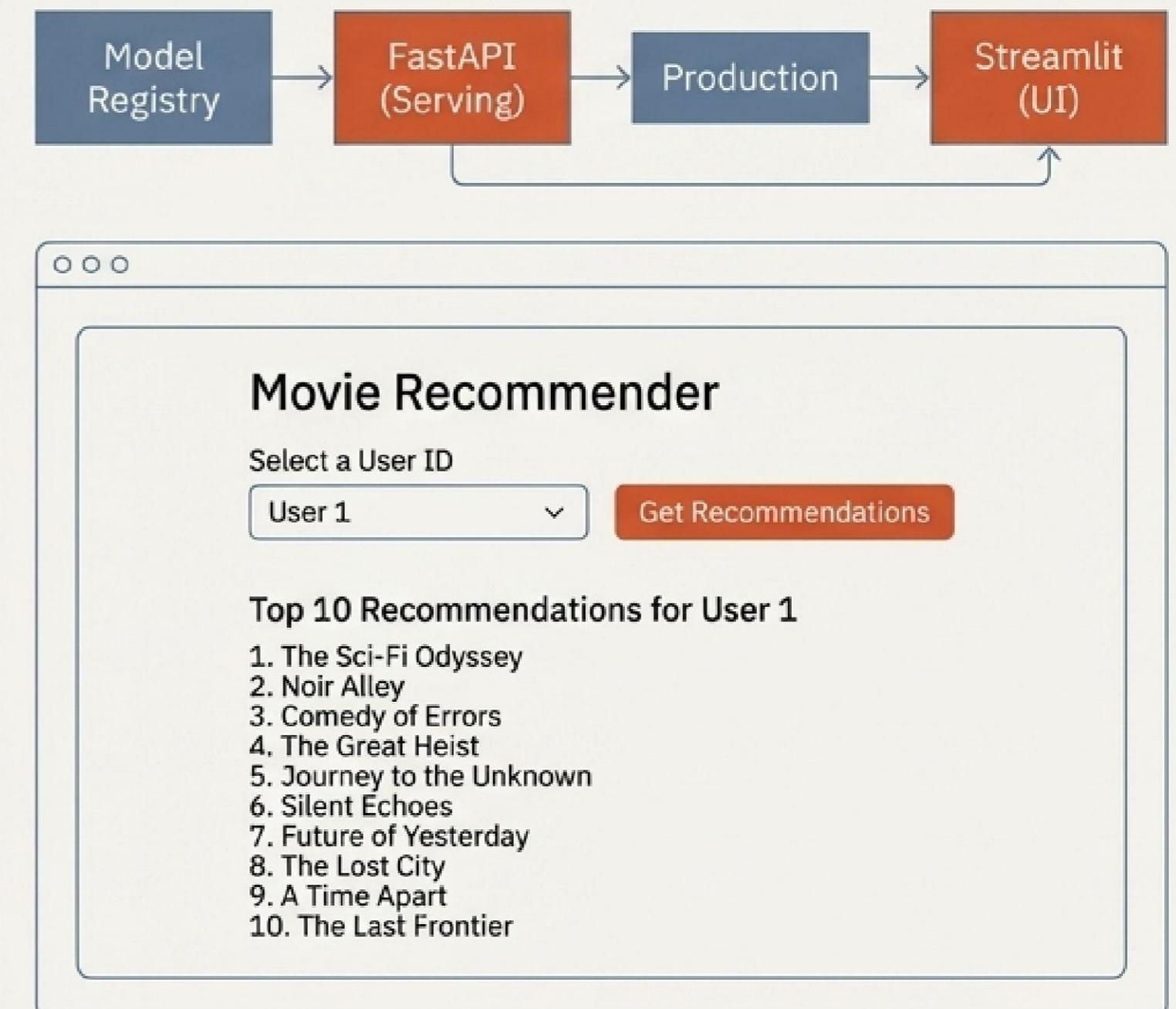
Une interface Streamlit est fournie pour permettre l'exploration et la démonstration du système de recommandation.

Fonctionnalité

Elle interagit directement avec l'API FastAPI, offrant un moyen simple de visualiser les recommandations pour n'importe quel utilisateur sans avoir à écrire de code.

Cible

Idéal pour le prototypage, la validation par les parties prenantes et le débogage.



Un Workflow Entièrement Automatisé

Partie 1: Intégration Continue (CI/CD)



outil

GitHub Actions (`.github/workflows/`)

Processus

Chaque `push` vers le repository déclenche automatiquement une suite d'actions : exécution des tests unitaires, linting du code, et vérification de la qualité. Cela garantit que la base de code reste saine et stable.

Partie 2: Commandes de Productivité Locale



outil

‘Makefile’

Description

Un ‘Makefile’ fournit des raccourcis pour toutes les tâches courantes, simplifiant le cycle de développement.

Exemples de Commandes

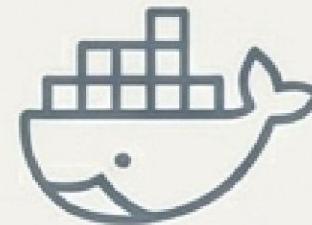
```
make install: Installation des dépendances.  
make train: Lancement du pipeline d'entraînement complet.  
make serve: Démarrage de l'API FastAPI.  
make ui: Lancement de l'interface Streamlit.  
make test: Exécution des tests.
```

La Structure d'un Projet MLOps Robuste

📁 mlops-recommender-system/	
└── 📂 .github/workflows/	CI/CD
└── 📂 configs/	Configuration Hydra
└── 📂 docker/	Dockerfiles
└── 📂 src/	Code source principal
└── 📂 data/	Pipelines de données
└── 📂 features/	Ingénierie des features
└── 📂 models/	Entraînement et évaluation
└── 📂 serving/	API de service
└── 📂 ui/	Interface Streamlit
└── 📂 tests/	Tests unitaires
└── 📄 dvc.yaml	Définition du pipeline DVC
└── 📄 params.yaml	Hyperparamètres
└── 📄 compose.yaml	Docker Compose
└── 📄 Makefile	Commandes rapides

Du Local au Live : Déploiement sur Render

1



Conteneurisation

Le projet est entièrement conteneurisé à l'aide de Docker et orchestré localement avec `docker-compose`.

```
make docker-up
```

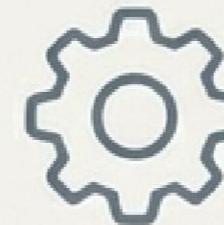
2



Connexion au Git

Connectez votre repository GitHub à un nouveau 'Web Service' sur la plateforme Render.

3



Configuration du Déploiement

Build Command

```
pip install poetry &&  
poetry install
```

Start Command

```
poetry run uvicorn  
src.serving.api:app --  
host 0.0.0.0 --port $PORT
```

4

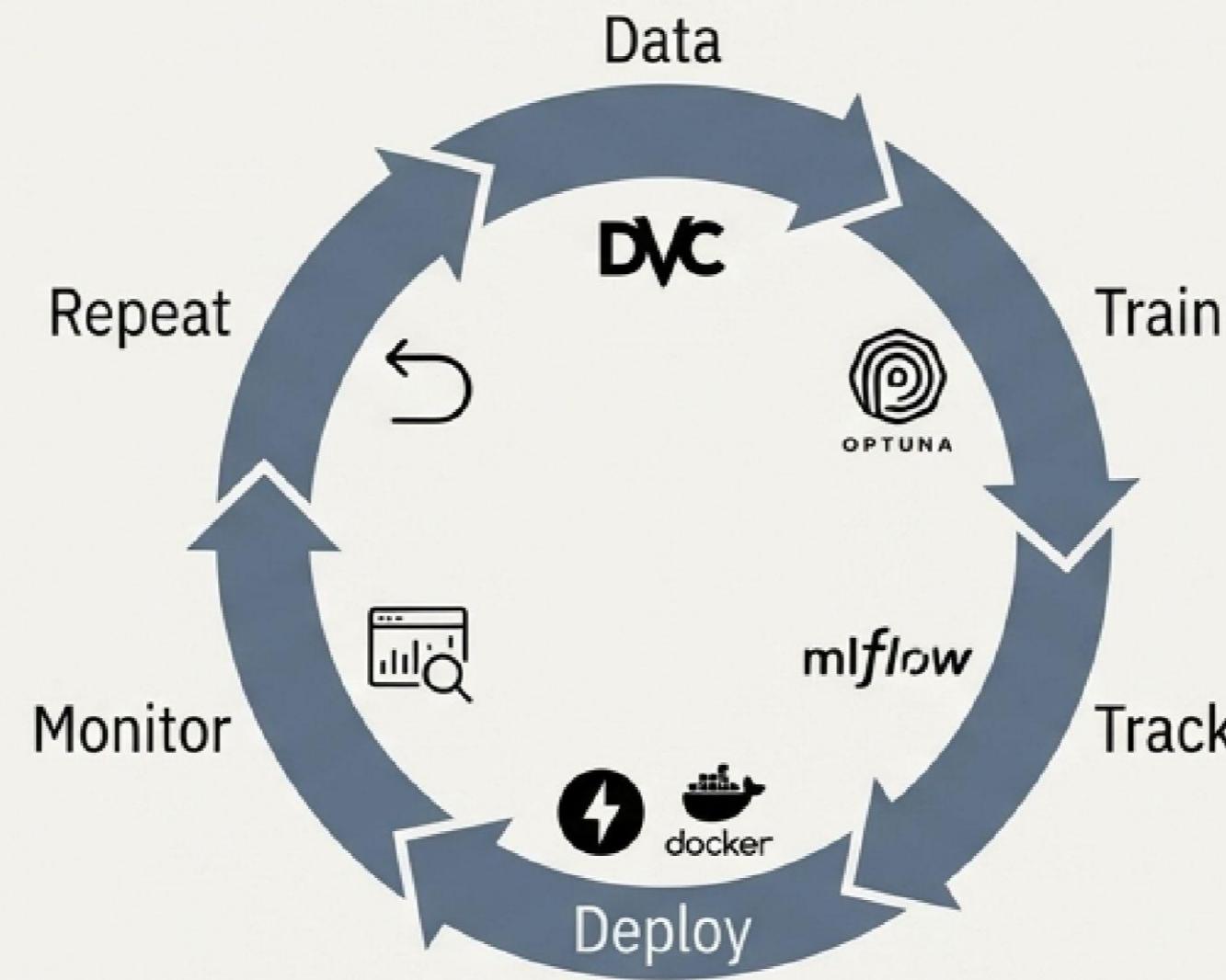


Finalisation

Ajoutez les variables d'environnement nécessaires (ex: `MLFLOW_TRACKING_URI`) et déployez. Render gère automatiquement la construction de l'image et le déploiement.

Pour des instructions détaillées, consultez le fichier 'SETUP.md' du repository.

Plus qu'un Projet : Un Blueprint MLOps Réutilisable



Nous avons parcouru un cycle MLOps complet : d'un pipeline de données versionné à un modèle optimisé et suivi, servi via une API robuste, et déployé en continu. Ce projet est une feuille de route pratique pour la mise en production de systèmes de machine learning.



Explorez le code, lancez le pipeline, et adaptez ce blueprint à vos propres défis.

github.com/Souley225/mlops-recommender-system