Projets Intégrateurs

DevOps & MLOps

12 projets en suggestion

Module DevOps & MLOps — 45h
28 septembre 2025

Table des matières

1 Introduction

Ce document présente **12 projets intégrateurs** conçus pour permettre aux étudiants de mobiliser l'ensemble des compétences acquises dans le module DevOps & MLOps. Chaque projet combine plusieurs outils étudiés : **Git/GitHub**, **Docker**, **CML**, **DVC** et **MLflow**, dans des contextes variés et progressifs.

Information

Objectifs généraux des projets

- ▷ **Intégration** : Combiner plusieurs outils MLOps dans un workflow cohérent
- ▷ **Reproductibilité** : Garantir la reproductibilité des expériences et déploiements
- ▷ **Collaboration** : Mettre en place des workflows Git efficaces pour le travail en équipe
- ▷ **Production** : Déployer des solutions ML en production avec monitoring
- ▷ **Bonnes pratiques** : Appliquer les standards industriels MLOps

1.1 Modalités de réalisation

- **Durée** : 2-3 semaines par projet
- **Format** : Individuel ou binôme
- **Environnement** : Docker Desktop sur Windows, GitHub, cloud optionnel
- **Évaluation** : Démonstration + code + rapport technique

1.2 Progression pédagogique

Les projets sont organisés pour permettre une montée en compétences progressive, en commençant par la maîtrise d'outils individuels puis en évoluant vers des intégrations plus complexes. Chaque projet peut être adapté selon les besoins et le temps disponible.

2 Projet 1 : Classification d'Images avec Pipeline DVC

Contexte

Une startup de e-commerce souhaite automatiser la classification de ses produits à partir d'images. Vous devez créer un pipeline ML reproductible pour entraîner et évaluer différents modèles de classification d'images.

Outils mobilisés

Outils principaux : Git/GitHub, Docker, DVC
Outils secondaires : Python, scikit-learn, TensorFlow/Keras, matplotlib

Objectifs pédagogiques

- Maîtriser le versioning de datasets avec DVC
- Créer des pipelines reproductibles avec `dvc.yaml`
- Containeriser un environnement ML avec Docker
- Gérer les artefacts et métriques avec DVC

2.1 Description détaillée

Vous travaillez avec un dataset d'images de produits (vêtements, électronique, etc.) que vous devez classifier automatiquement. Le projet doit inclure :

1. **Préparation des données** : Téléchargement, nettoyage, augmentation
2. **Entraînement** : Modèles CNN simples et transfer learning
3. **Évaluation** : Métriques, matrices de confusion, exemples d'erreurs
4. **Pipeline DVC** : Automatisation complète du workflow

Livrables attendus

✓ Repository Git avec structure DVC propre
✓ `dvc.yaml` avec stages : prepare, train, evaluate
✓ Dockerfile pour environnement reproductible
✓ Métriques et visualisations versionnées
✓ Documentation complète (README + rapport)

Conseils de réalisation

→ Utilisez un dataset public (CIFAR-10, Fashion-MNIST) pour commencer
→ Implémentez d'abord un modèle simple avant le transfer learning
→ Configurez un remote DVC (local ou cloud) dès le début
→ Documentez chaque étape du pipeline dans le README

Critères d'évaluation

Critères principaux (100 points)

☐ Pipeline DVC fonctionnel et reproductible (30 pts)
☐ Containerisation Docker optimisée (20 pts)
☐ Qualité du code et documentation (20 pts)
☐ Métriques et visualisations pertinentes (20 pts)
☐ Workflow Git propre (10 pts)

Extensions possibles

⋆ Intégration avec MLflow pour le tracking
⋆ Déploiement du modèle avec FastAPI
⋆ Pipeline d'augmentation de données avancé
⋆ Comparaison automatique de modèles

3 Projet 2 : API de Prédiction avec MLflow et Docker

Contexte

Une banque veut déployer un modèle de scoring crédit en production. Vous devez créer une API REST sécurisée qui sert le modèle, avec tracking MLflow et containerisation Docker pour un déploiement fiable.

Outils mobilisés

Outils principaux : Git/GitHub, Docker, MLflow
Outils secondaires : FastAPI, scikit-learn, Prometheus, pytest

Objectifs pédagogiques

- Maîtriser MLflow Tracking et Model Registry
- Développer une API REST production-ready
- Containeriser une application ML complète
- Implémenter monitoring et logging

3.1 Description détaillée

Vous développez un système de scoring crédit automatisé avec les composants suivants :

1. **Modèle ML** : Classification binaire (accord/refus crédit)
2. **API FastAPI** : Endpoints de prédiction avec validation
3. **MLflow** : Tracking des expériences et registry des modèles
4. **Monitoring** : Métriques Prometheus et health checks

Livrables attendus

✓ API FastAPI avec documentation automatique
✓ Modèles versionnés dans MLflow Registry
✓ Images Docker optimisées (multi-stage builds)
✓ Tests automatisés (pytest)
✓ Monitoring avec métriques Prometheus
✓ Documentation d'API et guide de déploiement

Conseils de réalisation

→ Utilisez des données synthétiques pour éviter les problèmes RGPD
→ Implémentez la validation des inputs avec Pydantic
→ Configurez MLflow avec backend PostgreSQL
→ Ajoutez des health checks dans le Dockerfile

Critères d'évaluation

Critères principaux (100 points)

☐ API fonctionnelle et documentée (25 pts)
☐ Intégration MLflow complète (25 pts)
☐ Containerisation et déploiement (20 pts)
☐ Tests et monitoring (20 pts)
☐ Sécurité et bonnes pratiques (10 pts)

4 Projet 3 : CI/CD pour Machine Learning avec CML

Contexte

Une équipe de data scientists veut automatiser ses expériences ML. Vous devez mettre en place un système CI/CD qui exécute automatiquement les entraînements et génère des rapports de comparaison dans les Pull Requests.

Outils mobilisés

Outils principaux : Git/GitHub, CML, Docker
Outils secondaires : GitHub Actions, Python, scikit-learn, matplotlib

Objectifs pédagogiques

- Maîtriser CML pour l'automatisation ML
- Configurer des workflows GitHub Actions
- Générer des rapports automatiques dans les PR
- Comparer des expériences de manière systématique

4.1 Description détaillée

Vous automatisez le workflow d'expérimentation pour un projet de prédiction de prix immobiliers :

1. **Expériences automatisées** : Entraînement déclenché par les commits
2. **Rapports CML** : Métriques et visualisations dans les PR
3. **Comparaisons** : Baseline vs nouvelles expériences
4. **Matrices d'expériences** : Tests de plusieurs hyperparamètres

Livrables attendus

✓ Workflows GitHub Actions avec CML
✓ Rapports automatiques dans les Pull Requests
✓ Comparaisons visuelles de modèles
✓ Matrices d'expériences automatisées
✓ Documentation du workflow CI/CD

Conseils de réalisation

→ Commencez par un workflow simple avant d'ajouter la complexité
→ Utilisez des secrets GitHub pour les credentials cloud
→ Optimisez les temps de build avec le cache
→ Testez les workflows sur des branches de développement

Critères d'évaluation

Critères principaux (100 points)

- ☐ Workflows CI/CD fonctionnels (30 pts)
- ☐ Rapports CML dans les PR (25 pts)
- ☐ Comparaisons automatiques (20 pts)
- ☐ Documentation et reproductibilité (15 pts)
- ☐ Optimisation et bonnes pratiques (10 pts)

5 Projet 4 : Analyse de Sentiments avec Workflow Git Avancé

Contexte

Une entreprise de médias sociaux veut analyser les sentiments des commentaires utilisateurs. Vous devez créer un système ML avec des workflows Git avancés permettant la collaboration entre plusieurs développeurs sur différentes approches.

Outils mobilisés

Outils principaux : Git/GitHub, Docker, DVC
Outils secondaires : Python, NLTK, transformers, pandas

Objectifs pédagogiques

- Maîtriser les workflows Git avancés (GitFlow, feature branches)
- Gérer la collaboration sur des projets ML
- Comparer différentes approches de NLP
- Versioner des modèles de traitement de texte

5.1 Description détaillée

Vous développez un système d'analyse de sentiments avec :

1. **Preprocessing** : Nettoyage de texte, tokenisation, vectorisation
2. **Modèles multiples** : Approches classiques (SVM, Naive Bayes) et modernes (BERT)
3. **Branches par approche** : Feature branches pour chaque méthode
4. **Comparaison** : Évaluation et sélection du meilleur modèle

Livrables attendus

- ✓ Repository avec branches multiples bien organisées
- ✓ Pipeline DVC pour chaque approche
- ✓ Comparaison systématique des modèles
- ✓ Documentation des workflows Git
- ✓ Rapport de sélection du modèle final

Conseils de réalisation

→ Utilisez des datasets publics (IMDB, Amazon reviews)
→ Créez une branche par approche (classical-ml, deep-learning, transformers)
→ Implémentez des pull requests avec code review
→ Documentez les choix techniques dans les commits

6 Projet 5 : Pipeline MLOps Complet avec DVC + MLflow

Contexte

Une entreprise de retail veut prédire la demande de ses produits. Vous devez créer un pipeline MLOps complet combinant versioning des données (DVC) et tracking des expériences (MLflow) pour un workflow professionnel.

Outils mobilisés

Outils principaux : Git/GitHub, DVC, MLflow, Docker
Outils secondaires : Python, pandas, scikit-learn, FastAPI

Objectifs pédagogiques

- Intégrer DVC et MLflow dans un workflow unique
- Gérer le cycle de vie complet des modèles
- Automatiser le passage de l'expérimentation à la production
- Implémenter un Model Registry professionnel

6.1 Description détaillée

Vous développez un système de prédiction de demande avec :

1. **Pipeline DVC** : Preprocessing, feature engineering, entraînement
2. **MLflow Tracking** : Logging automatique des runs DVC
3. **Model Registry** : Promotion automatique des meilleurs modèles
4. **API de serving** : Déploiement des modèles en production

Livrables attendus

✓ Pipeline DVC intégré avec MLflow
✓ Model Registry avec workflow de promotion
✓ API de serving automatisée
✓ Monitoring des performances en production
✓ Documentation technique complète

Conseils de réalisation

→ Configurez MLflow pour logger automatiquement les runs DVC
→ Implémentez des seuils de performance pour la promotion automatique
→ Utilisez des données de vente simulées réalistes
→ Créez des dashboards MLflow pour le suivi des expériences

7 Projet 6 : Détection de Fraude avec Monitoring Avancé

Contexte

Une fintech veut détecter les transactions frauduleuses en temps réel. Vous devez créer un système ML avec monitoring avancé pour détecter les dérives de performance et les anomalies dans les données.

Outils mobilisés

Outils principaux : Git/GitHub, Docker, MLflow
Outils secondaires : FastAPI, Prometheus, Grafana, scikit-learn

Objectifs pédagogiques

- Implémenter le monitoring de modèles ML
- Détecter la dérive des données (data drift)
- Créer des dashboards de performance
- Automatiser les alertes et notifications

7.1 Description détaillée

Vous créez un système de détection de fraude avec :

1. **Modèle de détection** : Classification binaire fraude/légitime
2. **API temps réel** : Scoring des transactions en live
3. **Monitoring** : Métriques de performance et data drift
4. **Alertes** : Notifications automatiques sur dégradation

Livrables attendus

✓ API de scoring en temps réel
✓ Dashboard Grafana avec métriques ML
✓ Système d'alertes automatiques
✓ Détection de data drift
✓ Documentation du monitoring

8 Projet 7 : Recommandation de Produits avec A/B Testing

Contexte

Une plateforme e-commerce veut améliorer son système de recommandations. Vous devez créer plusieurs modèles de recommandation et mettre en place un système d'A/B testing pour comparer leurs performances en production.

Outils mobilisés

Outils principaux : Git/GitHub, MLflow, CML
Outils secondaires : Docker, FastAPI, pandas, scikit-learn

Objectifs pédagogiques

- Développer plusieurs algorithmes de recommandation
- Implémenter un système d'A/B testing
- Comparer les performances en production
- Automatiser la sélection du meilleur modèle

8.1 Description détaillée

Vous développez un système de recommandations avec :

1. **Modèles multiples** : Collaborative filtering, content-based, hybrid
2. **A/B testing** : Déploiement simultané de plusieurs modèles
3. **Métriques business** : CTR, conversion, satisfaction utilisateur
4. **Sélection automatique** : Promotion du modèle le plus performant

Livrables attendus

✓ Plusieurs modèles de recommandation
✓ Système d'A/B testing fonctionnel
✓ Métriques business et techniques
✓ Rapports CML de comparaison
✓ Workflow de promotion automatique

9 Projet 8 : Prédiction de Séries Temporelles avec Déploiement Multi-Environnements

Contexte

Une entreprise énergétique veut prédire la consommation électrique. Vous devez créer un système de prédiction de séries temporelles avec déploiement sur plusieurs environnements (dev, staging, production).

Outils mobilisés

Outils principaux : Git/GitHub, Docker, DVC, MLflow
Outils secondaires : Docker Compose, FastAPI, pandas, statsmodels

Objectifs pédagogiques

- Maîtriser la prédiction de séries temporelles
- Gérer plusieurs environnements avec Docker Compose
- Implémenter des pipelines de déploiement
- Monitorer les prédictions en temps réel

9.1 Description détaillée

Vous créez un système de prédiction avec :

1. **Modèles temporels** : ARIMA, Prophet, LSTM
2. **Multi-environnements** : Configurations dev/staging/prod
3. **API de prédiction** : Endpoints pour prédictions à court/long terme
4. **Monitoring** : Suivi des erreurs de prédiction

Livrables attendus

✓ Pipeline DVC pour séries temporelles
✓ Docker Compose multi-environnements
✓ API de prédiction temps réel
✓ Monitoring des performances
✓ Documentation de déploiement

10 Projet 9 : Classification de Texte avec Pipeline CI/CD Complet

Contexte

Un journal en ligne veut automatiser la catégorisation de ses articles. Vous devez créer un système de classification de texte avec un pipeline CI/CD complet incluant tests automatisés et déploiement continu.

Outils mobilisés

Outils principaux : Git/GitHub, CML, MLflow, Docker
Outils secondaires : GitHub Actions, FastAPI, NLTK, transformers

Objectifs pédagogiques

- Créer des pipelines CI/CD pour NLP
- Implémenter des tests automatisés pour ML
- Automatiser les déploiements avec validation
- Gérer les versions de modèles de texte

10.1 Description détaillée

Vous développez un système de classification avec :

1. **Preprocessing NLP** : Nettoyage, tokenisation, vectorisation
2. **Tests automatisés** : Unit tests, integration tests, model validation
3. **Déploiement automatique** : Staging puis production après validation
4. **Rollback** : Retour automatique en cas de problème

Livrables attendus

✓ Pipeline CI/CD complet avec tests
✓ Déploiement automatique multi-environnements
✓ Tests de validation de modèles
✓ Système de rollback automatique
✓ Monitoring des performances NLP

11 Projet 10 : Vision par Ordinateur avec Edge Deployment

Contexte

Une entreprise de sécurité veut déployer des modèles de détection d'objets sur des caméras edge. Vous devez créer un système de vision par ordinateur optimisé pour le déploiement sur des dispositifs à ressources limitées.

Outils mobilisés

Outils principaux : Git/GitHub, Docker, MLflow
Outils secondaires : TensorFlow Lite, OpenCV, FastAPI, Raspberry Pi (simulation)

Objectifs pédagogiques

- Optimiser des modèles pour l'edge computing
- Gérer les contraintes de ressources
- Implémenter le déploiement sur dispositifs edge
- Monitorer les performances en edge

11.1 Description détaillée

Vous créez un système de vision avec :

1. **Modèle optimisé** : Quantification, pruning, distillation
2. **Edge deployment** : Conteneurs légers pour dispositifs edge
3. **Monitoring edge** : Métriques adaptées aux contraintes
4. **Mise à jour OTA** : Over-the-air model updates

Livrables attendus

- ✓ Modèles optimisés pour l'edge
- ✓ Conteneurs Docker légers
- ✓ Simulation de déploiement edge
- ✓ Système de mise à jour des modèles
- ✓ Monitoring adapté à l'edge

12 Projet 11 : Analyse Prédictive avec Feature Store

Contexte

Une banque veut centraliser ses features ML pour plusieurs cas d'usage (scoring, marketing, risque). Vous devez créer un feature store simple et l'intégrer dans des pipelines ML pour différents modèles.

Outils mobilisés

Outils principaux : Git/GitHub, DVC, MLflow, Docker
Outils secondaires : FastAPI, PostgreSQL, pandas, scikit-learn

Objectifs pédagogiques

- Créer un feature store centralisé
- Réutiliser des features entre projets
- Gérer les versions de features
- Optimiser les pipelines de feature engineering

12.1 Description détaillée

Vous développez un système avec :

1. **Feature store** : Base centralisée de features calculées
2. **API de features** : Service pour récupérer les features
3. **Pipelines multiples** : Plusieurs modèles utilisant les mêmes features
4. **Versioning** : Gestion des versions de features avec DVC

Livrables attendus

- ✓ Feature store fonctionnel
- ✓ API de récupération de features
- ✓ Plusieurs modèles utilisant le feature store
- ✓ Versioning des features avec DVC
- ✓ Documentation du feature store

13 Projet 12 : Plateforme ML Multi-Projets avec Governance

Contexte

Une entreprise tech veut standardiser ses pratiques ML entre équipes. Vous devez créer une plateforme MLOps permettant la collaboration entre plusieurs projets avec governance, standards et bonnes pratiques partagées.

Outils mobilisés

Outils principaux : Git/GitHub, CML, MLflow, DVC, Docker
Outils secondaires : GitHub Actions, Prometheus, Grafana, FastAPI

Objectifs pédagogiques

- Intégrer tous les outils MLOps dans une plateforme cohérente
- Implémenter des standards et governance
- Créer des workflows standardisés
- Faciliter la collaboration inter-équipes

13.1 Description détaillée

Vous créez une plateforme avec :

1. **Templates standardisés** : Structures de projets ML uniformes
2. **Workflows partagés** : CI/CD réutilisables entre projets
3. **Monitoring centralisé** : Dashboard multi-projets
4. **Governance** : Processus d'approbation et standards qualité

Livrables attendus

✓ Plateforme MLOps intégrée
✓ Templates de projets standardisés
✓ Workflows CI/CD réutilisables
✓ Dashboard de monitoring centralisé
✓ Documentation de governance
✓ Démonstration avec plusieurs projets

Conseils de réalisation

→ Créez 2-3 projets ML différents pour démontrer la plateforme
→ Implémentez des templates GitHub pour standardiser
→ Utilisez GitHub Organizations pour simuler l'entreprise
→ Créez des workflows réutilisables avec GitHub Actions

Critères d'évaluation

Critères principaux (100 points)

- ☐ Intégration des outils MLOps (30 pts)
- ☐ Standards et templates (25 pts)
- ☐ Workflows et automation (20 pts)
- ☐ Monitoring et governance (15 pts)
- ☐ Documentation et démonstration (10 pts)

14 Guide de sélection des projets

14.1 Par focus technique

- **Computer Vision** : Projets 1, 10
- **NLP/Text Mining** : Projets 4, 9
- **Données tabulaires** : Projets 2, 5, 6, 11
- **Séries temporelles** : Projet 8
- **Systèmes de recommandation** : Projet 7
- **Plateforme/Infrastructure** : Projets 12

14.2 Par durée disponible

- **2-3 semaines** : Projets 1, 2, 3, 4
- **3-4 semaines** : Projets 5, 6, 7, 8, 9
- **4-5 semaines** : Projets 10, 11, 12

14.3 Par combinaison d'outils

- **Git + Docker + DVC** : Projets 1, 4, 8, 11
- **Git + Docker + MLflow** : Projets 2, 6, 10
- **Git + CML + Docker** : Projets 3, 9
- **Intégration multiple** : Projets 5, 7, 12

15 Ressources et support

Information

Ressources techniques

- ▷ Documentation officielle de chaque outil
- ▷ Templates de projets sur GitHub
- ▷ Datasets publics recommandés
- ▷ Exemples de code et configurations

15.1 Évaluation finale

Chaque projet sera évalué sur :

1. **Démonstration technique** (40%) : Fonctionnalité et reproductibilité
2. **Qualité du code** (30%) : Structure, documentation, tests
3. **Rapport technique** (20%) : Architecture, choix techniques, limites
4. **Présentation** (10%) : Clarté, pédagogie, questions/réponses

15.2 Datasets recommandés

— **Vision** : CIFAR-10, Fashion-MNIST, COCO (subset)
— **NLP** : IMDB Reviews, 20 Newsgroups, Reuters
— **Tabulaire** : Titanic, Boston Housing, Wine Quality
— **Séries temporelles** : Stock prices, Weather data, Energy consumption
— **Recommandation** : MovieLens, Amazon Products