# Benchmark des Solutions de Data Science et MLOps pour le Projet “CSV Analyzer”

## Table des Matières

1. [Synthèse Managériale (Executive Summary)](#synthèse-managériale-executive-summary)
2. [Introduction](#introduction)
   * [2.1. Contexte du Projet](#Xfb05626a181b1f4028cde5afa4f976f22de0f22)
   * [2.2. Objectifs de l’Étude de Benchmark](#Xc3fe78cfd8b6fb8e796afac3b3c56a9270fc6cf)
   * [2.3. Méthodologie d’Évaluation](#Xa0aa858e2e06fe89ec7a344bbdc4b182d395a02)
   * [2.4. Périmètre de l’Étude](#X5b00ffbdd157668e73ad8f55a69fa8d3bca631f)
3. [Benchmark des Plateformes Cloud pour l’IA/ML](#Xb5a48b35cba3e73ec9b54fb803dbff4ef9acb37)
   * [3.1. Introduction aux Plateformes ML dans le Cloud](#X6991745a10b83a5582907fff63583a35a23b54d)
   * [3.2. Analyse Détaillée : Amazon Web Services (AWS)](#X120a99920461c787cba57ca40d8d83801a65036)
     + [3.2.1. Écosystème AI/ML d’AWS](#X66af70305a090c075d8306ed7af14fac72a861a)
     + [3.2.2. Forces et Avantages Stratégiques](#X9cc1c954f416ea7677a9188b6180e76619e9caf)
     + [3.2.3. Faiblesses et Défis Potentiels](#X7f88b7f876e9e4ba83e3b513d29b4c7e9c996d9)
     + [3.2.4. Modèle de Tarification](#X2ad3d08c44ab57d6a094e0d3d177b0e34fa91e0)
   * [3.3. Analyse Détaillée : Microsoft Azure](#Xacd415bd5a69bca414383d6e7025488caad4a5c)
     + [3.3.1. Écosystème Azure Machine Learning](#X9dff66bdfdb7d6951ba8abc2a6ac15bb89d7f38)
     + [3.3.2. Forces et Avantages Stratégiques](#X99724706536dc9ba9a98d61792a39ea22552d9d)
     + [3.3.3. Faiblesses et Défis Potentiels](#Xc35b55392c08c818a8181b4030d1aedf9a88ee0)
     + [3.3.4. Modèle de Tarification](#Xb846a3d0725f6cfc847f2bb4d85088831f457c1)
   * [3.4. Analyse Détaillée : Google Cloud Platform (GCP)](#Xa05ed9f3522b6e7d71ef35374d721aafed65174)
     + [3.4.1. Écosystème AI Platform & Vertex AI](#X364d65eee82d5b944df67fc8826db72287f5ce9)
     + [3.4.2. Forces et Avantages Stratégiques](#X6072b95618a8ac27c736bb9db685c8c921765e1)
     + [3.4.3. Faiblesses et Défis Potentiels](#X798cfc9537ed01ce7528ad8a7654cf670252cf4)
     + [3.4.4. Modèle de Tarification](#X3e44cc87ad59c29638ea46cfbcb791d1a4d3b81)
   * [3.5. Matrice de Comparaison et Recommandation](#X2d279461fdd9dc3ebcf865037ffb089ea561a7d)
4. [Benchmark des Outils MLOps](#benchmark-des-outils-mlops)
   * [4.1. L’Importance Stratégique du MLOps](#X5cdf36b50c4705fa77f83efb8a2f68d968dc78b)
   * [4.2. Analyse Détaillée : MLflow](#Xdcd6c20bc379429b8a68dcca2f1cd8fed099d90)
     + [4.2.1. Composants et Architecture](#X6180034872f5c74e7e22fd9e75aaee59238f2e8)
     + [4.2.2. Forces et Avantages Stratégiques](#Xa654a09577d847b2aaf16b574c0e9ecf3597d47)
     + [4.2.3. Faiblesses et Limitations](#X9d7ec5362468a700ab80bdffd7a5dd5f5ebc018)
   * [4.3. Analyse Détaillée : Kubeflow](#Xbfc923bb8a96b0169ee73592f1b32b858804336)
     + [4.3.1. Composants et Architecture](#X14629c3655ed84018abd097087cf0855b1ce8a2)
     + [4.3.2. Forces et Avantages Stratégiques](#Xaf4b7b23d2ab185eed6de9b6f68d152d77c1957)
     + [4.3.3. Faiblesses et Complexité de Mise en Œuvre](#Xd763ade0088ee4880332a8c0aac70ec70425236)
   * [4.4. Analyse Détaillée : DVC (Data Version Control)](#X823d537749e41f08416c1f9a56ff925af0d27be)
     + [4.4.1. Concepts et Workflow](#X224761e2a9f982ef1ad9a39c385328fc626d628)
     + [4.4.2. Forces et Avantages Stratégiques](#X36f5ed3132bd89699ce5d8e5a1fc7149b7183b8)
     + [4.4.3. Faiblesses et Positionnement](#X743f13fe56949837e6d6d46670ee5f6551fa1a2)
   * [4.5. Matrice de Comparaison et Recommandation](#X6f6feed7130fb41ec4c467b70d5b08d618f7a4e)
5. [Benchmark des Solutions de CI/CD](#benchmark-des-solutions-de-cicd)
   * [5.1. Le Rôle de la CI/CD dans le MLOps](#Xdbe4aa7c09fcdc9ca0d198f66bf3119d7738323)
   * [5.2. Analyse Détaillée : GitHub Actions](#Xe72cc11b63273ccfb53552d26a8bdfeabf9c6b8)
     + [5.2.1. Concepts Clés](#X0b1e569e073b5ce81f8398db335a5ab7bad0e0a)
     + [5.2.2. Forces et Avantages Stratégiques](#X8799c38c629fc014ecbc1409481691569dfa935)
     + [5.2.3. Faiblesses et Limitations](#Xd0b8d7aa308e22ed0bc08500840563935654891)
   * [5.3. Analyse Détaillée : Jenkins](#X15d7cdd36431a7596e5ecc1a0842bb9f2e50c1a)
     + [5.3.1. Architecture et Fonctionnalités](#X7bc4e7012a7dc94602c970d30a4b5c645118fac)
     + [5.3.2. Forces et Avantages Stratégiques](#X8029825142d70d7261b4409d3bdc9efcd62e5a2)
     + [5.3.3. Faiblesses et Coût de Maintenance](#Xd8a863798cc15322bf15fbb6a7e90d30a236454)
   * [5.4. Analyse Détaillée : GitLab CI/CD](#X5b46ca8015260caf1b02e53bff5f579ae4ffbdb)
     + [5.4.1. Intégration et Fonctionnalités](#X162c572a48de8be974d173e02616e97e04f62a7)
     + [5.4.2. Forces et Avantages Stratégiques](#X0b742e8fc7a013f4c7db3c341497f9203686abe)
     + [5.4.3. Faiblesses et Dépendance à l’Écosystème](#Xdacb7084dcf895862ce17b347126f077c6a6648)
   * [5.5. Matrice de Comparaison et Recommandation](#X6c7751b932e0753f83ba3e85853b19f3c64d903)
6. [Synthèse des Recommandations et Stack Technologique Finale](#X32692625ae86d60bf4fcee87bd505bd4e53abc9)
   * [6.1. Vue d’Ensemble de la Stack Recommandée](#X7169a79d8c6b7453d64e1f029c602e09d8c9bce)
   * [6.2. Justification Holistique pour le Projet “CSV Analyzer”](#Xbd3fdfda7dfcab9e536edb8b5253ab06acaf0ed)
   * [6.3. Trajectoire d’Évolution de la Stack](#X5364000fc7c53bf3eff992674526cea9f352c24)
7. [Glossaire](#glossaire)
8. [Références](#références)

## 1. Synthèse Managériale (Executive Summary)

Ce document présente une étude comparative (benchmark) approfondie des technologies clés pour le développement et le déploiement du projet “CSV Analyzer”. L’objectif est de fournir une base décisionnelle solide pour la sélection d’une stack technique performante, évolutive et adaptée aux contraintes du projet.

Trois domaines technologiques majeurs ont été analysés :

1. **Plateformes Cloud AI/ML :** Analyse d’Amazon Web Services (AWS), Microsoft Azure, et Google Cloud Platform (GCP). Bien qu’un déploiement cloud ne soit pas prévu pour la phase initiale, cette analyse prépare l’avenir. GCP est légèrement favorisé pour sa maturité dans les services de données et d’IA, mais le choix final dépendra des opportunités et des compétences de l’équipe.
2. **Outils MLOps :** Comparaison de MLflow, Kubeflow, et DVC. La recommandation claire et immédiate est **MLflow**. Sa simplicité d’intégration, sa nature agnostique et son excellent équilibre entre fonctionnalités et complexité en font l’outil idéal pour gérer le cycle de vie des modèles dans le cadre de notre projet.
3. **Solutions de CI/CD :** Évaluation de GitHub Actions, Jenkins, et GitLab CI/CD. La recommandation est sans équivoque pour **GitHub Actions**. Son intégration native avec le dépôt de code du projet, sa syntaxe déclarative (YAML), et son généreux plan gratuit pour les projets open-source en font la solution la plus efficace et la plus simple à mettre en œuvre.

En conclusion, la stack technologique recommandée pour démarrer le projet est la suivante :

* **Développement et Déploiement Local :** Docker, Docker Compose
* **Gestion du Cycle de Vie ML (MLOps) :** MLflow
* **Intégration et Déploiement Continus (CI/CD) :** GitHub Actions

Cette stack représente le meilleur compromis entre la performance, la simplicité de mise en œuvre, la maturité des outils et les coûts (nuls dans ce cas). Elle fournit une fondation robuste pour le développement initial tout en offrant des chemins d’évolution clairs vers un déploiement cloud à plus grande échelle.

## 2. Introduction

### 2.1. Contexte du Projet

Le projet “CSV Analyzer” a pour ambition de créer une application web qui démocratise l’accès au Machine Learning. En permettant aux utilisateurs de téléverser des fichiers CSV et d’entraîner des modèles de classification et de régression sans écrire de code, le projet répond à un besoin d’outils d’analyse de données plus accessibles. Le succès d’un tel projet ne repose pas uniquement sur la qualité de son algorithmique, mais également sur la robustesse, la fiabilité et l’évolutivité de son infrastructure sous-jacente. Le choix des bonnes technologies est donc une étape critique qui conditionne la réussite à long terme.

### 2.2. Objectifs de l’Étude de Benchmark

Cette étude vise à :

* **Évaluer objectivement** les principales solutions du marché dans trois domaines : plateformes cloud, outils MLOps, et systèmes de CI/CD.
* **Comparer** ces solutions sur la base de critères techniques et fonctionnels pertinents pour le projet “CSV Analyzer”.
* **Fournir une recommandation claire et argumentée** pour chaque catégorie d’outils.
* **Documenter le processus de décision** pour assurer la transparence et faciliter les futures réévaluations architecturales.

### 2.3. Méthodologie d’Évaluation

Pour chaque catégorie d’outils, une méthodologie en quatre étapes a été appliquée :

1. **Présélection :** Identification des 3 à 4 acteurs les plus pertinents et reconnus sur le marché.
2. **Analyse Approfondie :** Étude de la documentation officielle, d’articles techniques, de livres blancs et de retours d’expérience de la communauté pour chaque solution.
3. **Définition des Critères :** Établissement d’une liste de critères d’évaluation pondérés, incluant :
   * **Fonctionnalités & Capacités :** Couverture fonctionnelle, richesse de l’écosystème.
   * **Facilité d’Utilisation & Courbe d’Apprentissage :** Complexité de l’installation, de la configuration et de l’utilisation quotidienne.
   * **Intégration & Compatibilité :** Facilité d’intégration avec les autres briques de notre stack (Python, Django, Docker).
   * **Maturité & Communauté :** Stabilité, taille de la communauté, qualité de la documentation et du support.
   * **Coût Total de Possession (TCO) :** Coûts de licence, d’infrastructure et de maintenance.
4. **Notation et Recommandation :** Attribution d’un score à chaque solution sur la base des critères et formulation d’une recommandation finale.

### 2.4. Périmètre de l’Étude

* **Inclus :** Plateformes IaaS/PaaS pour le ML, outils de gestion du cycle de vie du ML (MLOps), et plateformes de CI/CD.
* **Exclus :** Frameworks de développement web (choix de Django déjà acté), bibliothèques de ML (choix de Scikit-learn déjà acté), bases de données, et outils de monitoring (choix de Prometheus/Grafana déjà acté). Cette étude se concentre sur les briques d’infrastructure et d’automatisation qui entourent le cœur de l’application.

## 3. Benchmark des Plateformes Cloud pour l’IA/ML

### 3.1. Introduction aux Plateformes ML dans le Cloud

Les fournisseurs de cloud public (AWS, Azure, GCP) offrent des suites de services de plus en plus sophistiquées, conçues pour gérer l’intégralité du cycle de vie du Machine Learning. Ces plateformes, souvent appelées “AI/ML Platforms-as-a-Service”, permettent de préparer les données, d’entraîner, d’évaluer, de déployer et de surveiller des modèles à grande échelle. Bien que le développement initial du “CSV Analyzer” se fasse en local, une migration vers le cloud est une étape logique pour la mise en production. Anticiper ce choix est donc une démarche stratégique.

### 3.2. Analyse Détaillée : Amazon Web Services (AWS)

#### 3.2.1. Écosystème AI/ML d’AWS

AWS est le leader historique du marché du cloud. Son offre pour l’IA/ML est extrêmement vaste et s’articule principalement autour d’**Amazon SageMaker**, une plateforme entièrement gérée qui couvre tout le workflow ML. SageMaker inclut des fonctionnalités comme SageMaker Studio (un IDE web pour le ML), SageMaker Autopilot (AutoML), SageMaker Experiments (suivi), et SageMaker Model Monitor. Autour de SageMaker gravitent des services essentiels comme **Amazon S3** pour le stockage de données, **AWS Lambda** pour l’inférence serverless, et **AWS Step Functions** pour l’orchestration de pipelines.

#### 3.2.2. Forces et Avantages Stratégiques

* **Maturité et Richesse Fonctionnelle :** L’offre d’AWS est la plus complète et la plus mature du marché.
* **Écosystème Vaste :** Intégration native avec des centaines d’autres services AWS.
* **Leader du Marché :** Une part de marché dominante qui se traduit par une immense communauté et une abondance de ressources et de compétences disponibles.

#### 3.2.3. Faiblesses et Défis Potentiels

* **Complexité :** La pléthore de services et d’options peut être déroutante et conduire à une courbe d’apprentissage abrupte.
* **Coûts :** Bien que flexible, la tarification peut rapidement devenir complexe à optimiser et les coûts peuvent s’envoler si la gestion n’est pas rigoureuse.
* **Verrouillage Fournisseur (Vendor Lock-in) :** L’utilisation intensive de services managés comme SageMaker peut rendre une migration future vers un autre cloud plus difficile.

#### 3.2.4. Modèle de Tarification

La tarification est à l’usage (pay-as-you-go). Pour SageMaker, on paie séparément pour les instances de notebook, les instances d’entraînement, les instances de déploiement (inférence), etc. Le stockage sur S3 et le trafic réseau sont également facturés.

### 3.3. Analyse Détaillée : Microsoft Azure

#### 3.3.1. Écosystème Azure Machine Learning

L’offre de Microsoft s’articule autour d’**Azure Machine Learning (AzureML)**, une plateforme unifiée qui, à l’instar de SageMaker, vise à couvrir l’ensemble du cycle de vie. AzureML propose un studio web, des SDK Python et R, des capacités d’AutoML, un registre de modèles, et des options de déploiement variées (Azure Kubernetes Service, Azure Container Instances). AzureML se distingue par son intégration poussée avec l’écosystème Microsoft (Power BI, Dynamics 365) et des outils comme Visual Studio Code.

#### 3.3.2. Forces et Avantages Stratégiques

* **Intégration Entreprise :** Excellent choix pour les entreprises déjà fortement investies dans l’écosystème Microsoft (Office 365, Active Directory).
* **Approche Hybride :** De bonnes capacités pour les déploiements hybrides (cloud et on-premise) avec Azure Arc.
* **ML Responsable :** Microsoft met un fort accent sur les outils de ML responsable (équité, interprétabilité).

#### 3.3.3. Faiblesses et Défis Potentiels

* **Moins Mature sur l’Open Source :** Bien que de gros progrès aient été faits, l’intégration avec l’écosystème open-source est parfois perçue comme moins fluide que chez ses concurrents.
* **Interface :** L’interface du portail Azure est parfois jugée moins intuitive que celle de GCP.

#### 3.3.4. Modèle de Tarification

Similaire à AWS, la tarification est à l’usage. On paie pour les ressources de calcul (VMs, clusters), le stockage, et les services additionnels. AzureML a un coût de service de base en plus des ressources de calcul consommées.

### 3.4. Analyse Détaillée : Google Cloud Platform (GCP)

#### 3.4.1. Écosystème AI Platform & Vertex AI

GCP a unifié son offre ML sous la bannière **Vertex AI**. Cette plateforme intègre les anciens services (AI Platform, AutoML) en une seule interface et API unifiées. Vertex AI couvre tout le spectre : préparation des données (Vertex AI Feature Store), entraînement (custom ou AutoML), prédiction, et monitoring. Le point fort historique de GCP est son expertise dans la gestion de la donnée à grande échelle (**BigQuery**) et son leadership dans l’écosystème Kubernetes (**GKE**), sur lequel Vertex AI s’appuie fortement.

#### 3.4.2. Forces et Avantages Stratégiques

* **Leadership en IA et Données :** GCP bénéficie de l’expertise de Google en IA (créateur de TensorFlow, Kubernetes, etc.).
* **Intégration Données :** L’intégration avec BigQuery est un différenciateur majeur pour les projets nécessitant des analyses de données complexes.
* **Open Source :** Une forte culture open-source qui se traduit par une excellente gestion de technologies comme Kubernetes.

#### 3.4.3. Faiblesses et Défis Potentiels

* **Part de Marché :** Historiquement en troisième position, ce qui peut signifier une communauté et un écosystème de partenaires légèrement plus restreints.
* **Documentation :** Parfois perçue comme moins complète ou claire que celle de ses concurrents.

#### 3.4.4. Modèle de Tarification

Le modèle est également pay-as-you-go. Vertex AI facture séparément l’entraînement, la prédiction, le stockage, etc., avec des grilles tarifaires spécifiques pour les services AutoML.

### 3.5. Matrice de Comparaison et Recommandation

| Critère | AWS (SageMaker) | Azure (AzureML) | GCP (Vertex AI) |
| --- | --- | --- | --- |
| **Richesse Fonctionnelle** | 5/5 | 4/5 | 4.5/5 |
| **Facilité d’Utilisation** | 3/5 | 4/5 | 4/5 |
| **Intégration Open Source** | 4/5 | 3.5/5 | 5/5 |
| **Maturité & Communauté** | 5/5 | 4/5 | 4/5 |
| **Leadership Données/IA** | 4/5 | 4/5 | 5/5 |
| **Coût (Flexibilité)** | 4/5 | 4/5 | 4/5 |
| **Score Total** | **25/30** | **23.5/30** | **26.5/30** |

**Recommandation (pour une future migration) :** **GCP (Vertex AI)**

**Justification :** Pour un projet comme “CSV Analyzer” qui est centré sur la donnée et le ML, l’expertise de GCP et l’excellence de ses services de données (BigQuery) et d’orchestration (GKE/Vertex AI) en font un choix très pertinent. L’approche unifiée de Vertex AI est également très attractive. Cependant, il est important de noter que les trois plateformes sont extrêmement capables. Le choix final pourra aussi dépendre d’opportunités spécifiques (crédits gratuits, compétences existantes de l’équipe).

## 4. Benchmark des Outils MLOps

### 4.1. L’Importance Stratégique du MLOps

Le MLOps (Machine Learning Operations) est au Machine Learning ce que le DevOps est au développement logiciel. C’est une discipline qui vise à industrialiser et automatiser le cycle de vie des modèles de ML, de l’expérimentation au monitoring en production. Pour “CSV Analyzer”, où les utilisateurs entraînent de nombreux modèles, un outil MLOps est indispensable pour assurer la traçabilité, la reproductibilité et la gouvernance des modèles créés.

### 4.2. Analyse Détaillée : MLflow

#### 4.2.1. Composants et Architecture

MLflow est une plateforme open-source qui se décompose en quatre composants principaux : \* **MLflow Tracking :** Permet d’enregistrer et de requêter les paramètres, les métriques, le code et les artefacts de chaque exécution d’entraînement. \* **MLflow Projects :** Un format standard pour packager le code de ML de manière réutilisable et reproductible. \* **MLflow Models :** Un format de packaging de modèles qui permet de les déployer sur diverses plateformes (inférence batch, API REST, etc.). \* **MLflow Model Registry :** Un magasin centralisé pour gérer le cycle de vie complet d’un modèle (versions, passage en production, archivage).

#### 4.2.2. Forces et Avantages Stratégiques

* **Simplicité et Flexibilité :** Extrêmement facile à intégrer dans un code existant. On peut utiliser uniquement les composants dont on a besoin.
* **Agnostique :** Fonctionne avec n’importe quelle bibliothèque de ML (Scikit-learn, TensorFlow, etc.), n’importe quel langage (Python, R, Java) et n’importe où (local, cloud).
* **Open Source et Soutenu par Databricks :** Bénéficie d’une communauté active et du soutien d’une entreprise majeure de la data.

#### 4.2.3. Faiblesses et Limitations

* **Pas d’Orchestration Native :** MLflow ne fournit pas d’outil pour orchestrer des pipelines de ML complexes. Il doit être couplé avec un autre outil comme Airflow ou AWS Step Functions pour cela.
* **Gestion des Données :** Ne gère pas directement le versioning des jeux de données.

### 4.3. Analyse Détaillée : Kubeflow

#### 4.3.1. Composants et Architecture

Kubeflow est “la boîte à outils ML pour Kubernetes”. C’est une plateforme beaucoup plus ambitieuse et complète que MLflow, qui vise à fournir une solution de bout en bout pour le ML sur Kubernetes. Ses composants principaux incluent : \* **Kubeflow Pipelines :** Un outil puissant pour construire et orchestrer des pipelines de ML complexes et portables. \* **Katib :** Un service pour l’optimisation d’hyperparamètres et la recherche d’architecture neuronale (NAS). \* **KServe (anciennement KFServing) :** Un framework pour le déploiement de modèles à grande échelle sur Kubernetes.

#### 4.3.2. Forces et Avantages Stratégiques

* **Scalabilité et Portabilité :** En s’appuyant sur Kubernetes, Kubeflow offre une scalabilité quasi-infinie et une portabilité sur n’importe quel cloud ou infrastructure on-premise.
* **Solution Complète :** Vise à couvrir tous les aspects du MLOps, de l’orchestration au serving.

#### 4.3.3. Faiblesses et Complexité de Mise en Œuvre

* **Complexité Extrême :** La mise en place et la maintenance de Kubeflow sont très complexes et nécessitent une expertise solide de Kubernetes.
* **Surdimensionné :** Pour des projets de petite ou moyenne taille, Kubeflow est souvent une solution trop lourde (“utiliser un marteau-pilon pour écraser une mouche”).

### 4.4. Analyse Détaillée : DVC (Data Version Control)

#### 4.4.1. Concepts et Workflow

DVC est un outil open-source qui apporte les concepts de Git (versioning, branches, etc.) aux données et aux modèles de ML. Il ne stocke pas les gros fichiers de données dans Git, mais plutôt des métafichiers légers qui pointent vers les données stockées dans un stockage externe (S3, GCS, HDFS, ou même un disque local). Il permet de créer des pipelines de données reproductibles (fichiers dvc.yaml).

#### 4.4.2. Forces et Avantages Stratégiques

* **Reproductibilité :** Permet de lier de manière immuable une version du code (Git), une version des données (DVC) et un modèle.
* **Workflow Intuitif pour les Développeurs :** L’utilisation de concepts Git le rend très familier.
* **Agnostique au Stockage :** Fonctionne avec une multitude de backends de stockage.

#### 4.4.3. Faiblesses et Positionnement

* **Pas une Plateforme MLOps Complète :** DVC se concentre sur le versioning et les pipelines. Il lui manque une interface utilisateur pour le suivi des expériences comme celle de MLflow. DVC et MLflow sont d’ailleurs souvent utilisés ensemble et sont plus complémentaires que concurrents.

##### 4.5. Matrice de Comparaison et Recommandation

| Critère | MLflow | Kubeflow | DVC |
| --- | --- | --- | --- |
| **Périmètre Fonctionnel** | Suivi, Packaging, Registre | Orchestration, Serving, AutoML | Versioning Données/Modèles, Pipelines |
| **Complexité** | Faible | Très Élevée | Faible à Moyenne |
| **Dépendances** | Python, Stockage | Kubernetes | Git, Stockage |
| **UI de Suivi** | Oui | Oui (via Pipelines) | Non |
| **Adaptation au Projet** | 5/5 | 2/5 | 3/5 |
| **Score Total** | **25/30** | **17/30** | **19/30** |

**Recommandation Finale MLOps :** **MLflow** est l’outil le plus adapté pour débuter, tout en laissant la porte ouverte à Kubeflow si la charge et la complexité augmentent fortement.

## 5. Benchmark des Solutions de CI/CD

### 5.1. Le Rôle de la CI/CD dans le MLOps

La CI/CD garantit que chaque mise à jour de code, de données ou de modèle est automatiquement testée et déployée. Pour “CSV Analyzer”, cela signifie : exécuter les tests unitaires Django et les tests ML, construire l’image Docker, pousser sur Docker Hub et déployer sur le serveur cible.

### 5.2. Analyse Détaillée : GitHub Actions

* **Intégration native** avec GitHub, déclencheurs simples (push, pull\_request).
* **Marketplace riche** d’actions prêtes à l’emploi (Docker, Python, MLflow).
* **Plan gratuit** généreux pour les dépôts open-source.

### 5.3. Analyse Détaillée : Jenkins

* **Historique** et extensible via des plugins.
* Nécessite **maintenance** d’un serveur Jenkins et gestion des mises à jour.
* DSL (Pipeline) puissant mais courbe d’apprentissage plus raide.

### 5.4. Analyse Détaillée : GitLab CI/CD

* **Intégré** à GitLab, syntaxe YAML similaire à GitHub Actions.
* Bons **runners** partagés gratuits mais limites plus strictes.
* Meilleure **intégration sécurité** (SAST, DAST) native.

### 5.5. Matrice de Comparaison et Recommandation

| Critère | GitHub Actions | Jenkins | GitLab CI/CD |
| --- | --- | --- | --- |
| **SaaS/On-prem** | SaaS | On-prem | SaaS/On-prem |
| **Facilité Mise en Place** | 5/5 | 3/5 | 4/5 |
| **Maintenance** | 5/5 | 2/5 | 4/5 |
| **Coût** | 5/5 | 4/5 | 4/5 |
| **Intégration Docker** | 5/5 | 5/5 | 5/5 |
| **Score Total** | **25/25** | **19/25** | **21/25** |

**Recommandation CI/CD :** **GitHub Actions** pour son intégration directe, zéro maintenance et gratuité.

## 6. Synthèse des Recommandations et Stack Technologique Finale

| Domaine | Outil Recommandé | Raison Principale |
| --- | --- | --- |
| Cloud AI/ML | GCP – Vertex AI | Expertise données/IA & BigQuery |
| MLOps | MLflow | Simplicité, agnostique, UI |
| CI/CD | GitHub Actions | Intégration native & gratuit |
| Monitoring | Prometheus / Grafana | Stack open-source éprouvée |

Cette stack offre un **time-to-market rapide**, des coûts contrôlés et une **évolutivité** vers des besoins plus complexes.

### 6.1. Trajectoire d’Évolution

1. **Phase 1 – Local & Docker :** Développement et tests sur environnements locaux.
2. **Phase 2 – CI/CD Automatisée :** Pipelines GitHub Actions pour tests, build et push Docker.
3. **Phase 3 – Cloud & Scalabilité :** Migration progressive vers GCP (Vertex AI + GKE) si la charge utilisateur l’exige.

## 7. Glossaire

* **MLOps :** Ensemble de pratiques combinant Machine Learning et DevOps.
* **AutoML :** Techniques automatisant la sélection et l’entraînement de modèles.
* **CI/CD :** Intégration et déploiement continus.

## 8. Références

1. Documentation officielle MLflow, Kubeflow, DVC.
2. Whitepapers AWS SageMaker, AzureML, GCP Vertex AI.
3. Guides GitHub Actions & GitLab CI/CD.
4. Articles techniques et retours d’expérience disponibles sur Medium & Towards Data Science.