



## 3. Mise en production

### Déploiement de la VM [↗](#)

La machine virtuelle utilisée pour la mise en production du projet dispose des ressources suivantes :

- RAM : 4G
- CPU : 4 (2 Sockets, 2 Cores)
- Stockage : SSD 128G
- OS : Ubuntu Server 24.04
- Nom de domaine : **parivision.heuzef.com**

---

### Connexion SSH [↗](#)

La connexion s'effectue en SSH avec votre clé privée.

```
ssh -i .ssh/cle.priv parivision@parivision.heuzef.com
```

Java

### Installation [↗](#)

La VM est déployée sur un Hyperviseur Proxmox, l'installation d'Ubuntu Server est effectuée sans partitionnement LVM, avec les drivers des applications tierces, OpenSSH et sans ajout logiciels supplémentaire.

Un pare-feu et un reverse proxy est actuellement en place sur l'infra de Heuzef pour autoriser publiquement les ports suivants :

- 22 (SSH)
- 80 (HTTP)
- 443 (HTTPS)

### Initialisation [↗](#)

```
# Mise à jour du système
sudo apt update && sudo apt upgrade

# Désactiver l'authentification par mot de passe
sudo vim /etc/ssh/sshd_config # PasswordAuthentication no
sudo systemctl restart ssh

# Ajouter les clefs SSH pour l'authentification
vim /home/parivision/.ssh/authorized_keys
```

Java

Une sauvegarde de bas niveau est effectuée à ce stade.

## Configuration de la VM

```
# Configurer sur l'heure de Paris
sudo timedatectl set-timezone "Europe/Paris"

# Ajout de quelques outils
sudo apt install -y curl wget git tmux htop vim nano tree unzip smartmontools bmon
```

Bash

## Github

Le dépôt GIT du projet est hébergé ici : [https://github.com/DataScientest-Studio/DEC24\\_MLOPS\\_PARIS\\_SPORTIFS](https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS)

## Cloner le dépôt

```
# Création d'une nouvelle clef SSH
ssh-keygen -t ed25519

# La clef doit être ajoutée sur le repo github :
# https://github.com/DataScientest-Studio/DEC24\_MLOPS\_PARIS\_SPORTIFS/settings/keys
cat .ssh/id_ed25519.pub

# Cloner le repo GIT :
cd
git clone git@github.com:DataScientest-Studio/DEC24\_MLOPS\_PARIS\_SPORTIFS.git
tree DEC24_MLOPS_PARIS_SPORTIFS/
```

Bash

## Actualiser le dépôt

```
cd /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/ ; git pull
```

Bash

## Mise en place d'un Workflow Github Actions

Le workflow suivant est ajouté dans notre dépôt GIT dans [.github/workflows/workflows.yaml](#)

```
name: Workflow GitHub Action

# Run Workflow when push on main
on:
  push:
    branches:
      - master
jobs:
  execute:
    name: Update the production VM
```

```

runs-on: ubuntu-latest
steps:
- name: Connecting on remote VM
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.SSH_HOST }}
    username: ${ secrets.SSH_USER }}
    key: ${ secrets.SSH_KEY }}
    script: |
      cd /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/
      git fetch
      git pull

```

## Java

Une clé SSH est créée pour l'occasion, dédiée à Github et autorisée sur notre VM de production et ajoutée sur Github.com.

The screenshot shows the GitHub repository settings for 'DEC24\_MLOPS\_PARIS\_SPORTIFS'. The 'Settings' tab is selected, and the 'Secrets and variables' section is expanded. The 'Repository secrets' table shows three secrets: SSH\_HOST, SSH\_KEY, and SSH\_USER, all updated within the last 27 minutes. A red arrow points to the 'Settings' tab, and another red arrow points to the 'Secrets and variables' section in the left sidebar.

## Mise en place d'un Workflow de tests unitaires [🔗](#)

Le workflow suivant est ajouté dans notre dépôt GIT dans [.github/workflows/pytest.yaml](https://github.com/parivision/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/.github/workflows/pytest.yaml)

```

name: Run units tests
# Run Units tests when push on master
on:
  push:
    branches:
      - master
jobs:
  execute:
    name: Run Pytest
    runs-on: ubuntu-latest
    steps:

```

```

- uses: actions/checkout@v4
- name: Set up Python
  uses: actions/setup-python@v5
  with:
    python-version: '3.x'
- name: Install dependencies
  run: |
    python -m pip install --upgrade pip
    pip install -r requirements.txt
- name: Test with pytest
  run: |
    pip install pytest pytest-cov
    pytest tests/*.py --doctest-modules --junitxml=junit/test-results.xml --
cov=com --cov-report=xml --cov-report=html

```

## Java

Ce dernier va effectuer une série de test unitaires, avec le module Pytest, l'ensemble des scripts `tests/*.py` qui seront ajoutée au fur et à mesure du développement, offrant une évolution TDD possible.

```

(.venv) heuzef@PGMR:~/GIT/DEC24_MLOPS_PARIS_SPORTIFS$ pytest tests/*.py
===== test session starts =====
platform linux -- Python 3.12.8, pytest-8.3.4, pluggy-1.5.0
rootdir: /home/heuzef/GIT/DEC24_MLOPS_PARIS_SPORTIFS
plugins: Faker-33.3.1, anyio-4.9.0
collected 8 items

tests/test_process_data.py ..... [ 75%]
tests/tests.py .. [100%]

===== 8 passed in 2.43s =====

```


## Livraison continue (CD) [↗](#)

L'actualisation du dépôt est maintenant effectuée automatiquement à chaque push sur Github :


All checks have passed

2 successful checks

✓


Run units tests / Run Pytest (push)
Successful in 1m
Details

✓


Workflows GitHub Action / Update the production VM (push)
Successful in 8s
Details

## Python [↗](#)

## Mise à jour et installation des dépendances [↗](#)

```

sudo apt upgrade python3
sudo apt install python3-pip python3-venv

```

## Bash

## Création de l'environnement virtuel [↗](#)

```

cd /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS
python3 -m venv .venv
source .venv/bin/activate

```

Bash

## Installation des librairies

```
pip install -r requirements.txt
```

Python

## Fichier d'environnement

Un fichier d'environnement présent sur la VM de production est ignoré à la racine du dépôt GIT, ce dernier contient les différents secrets nécessaire au projet.

```
/home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/.env
```

Bash

Les scripts python du projet utilisent la librairie **dotenv** pour charger les secrets :

```
import os
from dotenv import load_dotenv
load_dotenv(".env")

secret = os.getenv('secret')
```

Python

Une sauvegarde de bas niveau est effectuée à ce stade.

## Docker

Docker sert de composante principale pour l'infrastructure du projet.

## Installation de Docker sur la VM

Réf : <https://docs.docker.com/engine/install/ubuntu/>

```
# Add Docker's official GPG key:
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
```

```
# Install the latest version:
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin docker-compose
```

Java

## Configuration

Réf: <https://docs.docker.com/engine/install/linux-postinstall/>

```
# Post-install:
sudo usermod -aG docker parivision
newgrp docker
docker run hello-world

# Auto-start:
sudo systemctl enable docker.service
sudo systemctl enable containerd.service
```

Java

## Docker-Compose

La configuration complète de Docker repose sur Docker-Compose.

Pour cela, le fichier **docker-compose.yml** qui définit la configuration de PariVision est intégré au processus de démarrage. : [https://github.com/DataScientest-Studio/DEC24\\_MLOPS\\_PARIS\\_SPORTIFS/blob/master/docker-compose.yml](https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/docker-compose.yml)

Pour tester, exécuter manuellement le lancement de l'infrastructure Docker ainsi :

```
docker-compose up /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/docker-compose.yml
```

Bash

L'ensemble des services devraient être accessibles.

Le service PORTAINER permet d'administrer sur une interface web toute la configuration Docker, accessible sur le port 9443. Par exemple, pour lister tous les containers :

<https://parivision.heuzef.com:9443/#/2/docker/containers>

Une sauvegarde de bas niveau est effectuée à ce stade.

## Homer

## Description

Homer est un portail statique personnalisable servant de point d'entrée du projet pour lister et accéder rapidement aux différents services du projet.

Dépôt du projet <https://github.com/bastienwartz/homer>

L'adresse d'accès est le domaine principale du projet : <https://parivision.heuzef.com>

La configuration du Dashboard de Homer s'effectue directement sur le dépôt GIT, dossier **"homer"** :  
[https://github.com/DataScientest-Studio/DEC24\\_MLOPS\\_PARIS\\_SPORTIFS/tree/master/homer](https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/tree/master/homer)

Le portail est configuré via un simple fichier YAML.

## Jenkins

### Installation

L'installation de Jenkins ne sera pas containerisé car le projet est sur une VM unique et le nœud maître doit avoir la possibilité de gérer les instances docker du projet. La méthode DooD n'est pas favorisée ici.

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update
sudo apt install -y fontconfig openjdk-17-jre jenkins

sudo usermod -aG docker jenkins
sudo systemctl restart docker

sudo systemctl enable jenkins
sudo systemctl stop jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins

sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Bash

Se rendre sur l'instance, port **8080**, pour déverrouiller l'installation avec le `initialAdminPassword`.

### Configuration

- Démarrer l'installation des plugins communautaires recommandés.
- Créer le compte administrateur.

### Plugins

Installer les plugins suivants :

- GitHub Integration
- Blue Ocean
- AnsiColor

### Créer un agent parivision

L'utilisateur Jenkins ne dispose pas des mêmes permissions que l'utilisateur parivision. Nous pouvons remédier à cela en créant un agent "parivision" qui se connectera sur l'hôte local avec l'utilisateur parivision.

The screenshot shows the Jenkins configuration page for a node named 'parivision'. The breadcrumb navigation at the top is 'Tableau de bord > Nœuds > parivision > Configurer'. On the left sidebar, there are links for 'Déconnexion', 'Open Blue Ocean', and 'État du lanceur de compilations 0/1'. The main configuration area includes the following fields:

- Number of executors**: Set to 1.
- Répertoire de travail du système distant**: Set to /home/parivision/.
- Étiquettes**: Set to parivision.
- Utilisation**: Set to 'Utiliser ce nœud autant que possible'.
- Méthode de lancement**: Set to 'Launch agents via SSH'.
- Host**: Set to localhost.
- Credentials**: Set to parivision/\*\*\*\*\* (parivision).
- Host Key Verification Strategy**: Set to 'Non verifying Verification Strategy'.
- Disponibilité**: Set to 'Maintenir l'agent activé le plus longtemps possible'.

At the bottom of the configuration area, there are buttons for 'Avancé' (with a dropdown arrow) and 'Edited' (with a pencil icon).

Finalement, le nœud maître principal doit être désactivé, en ajustant son nombre d'exécuteurs à zéro.

The screenshot shows the Jenkins configuration page for the 'contrôleur' node. The breadcrumb navigation at the top is 'Tableau de bord > Nœuds > contrôleur > Configurer'. On the left sidebar, there are links for 'Statut' and 'Configurer'. The main configuration area includes the following fields:

- Nombre d'exécuteurs**: Set to 0, which is circled in red.

Finalement, les différents Jobs du projets peuvent être mis en place.

## Pipeline

Concernant la Pipeline ML du projet PariVision, celle-ci sera synchronisé sur le JenkinsFile disponible dans le dépôt GIT : [https://github.com/DataScientest-Studio/DEC24\\_MLOPS\\_PARIS\\_SPORTIFS/blob/master/Jenkinsfile](https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/Jenkinsfile)

La création de l'objet Pipeline s'effectue avec les paramètres suivants :



## Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

### Definition

Pipeline script from SCM

SCM ?  
Git

Repositories ?  
Repository URL ?  
/home/parivision/DEC24\_MLOPS\_PARIS\_SPORTIFS  
Credentials ?  
parivision/\*\*\*\*\*\* (parivision)  
+ Ajouter  
Avancé ▾  
Add Repository

Branches to build ?  
Branch Specifier (blank for 'any') ?  
\*/master  
Add Branch

Navigateur de la base de code ?  
(Auto)

Additional Behaviours  
Ajouter ▾

Script Path ?  
Jenkinsfile

Afin d'autoriser l'accès au repo à Jenkins, il faut désactiver la vérification de propriété depuis le compte jenkins :

```
su jenkins
git config --global --add safe.directory
/home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/.git
```

## Java

Une sauvegarde de bas niveau est effectuée à ce stade.

## MLFlow

Accès à l'instance déployé via Docker : <http://parivision.heuzef.com:5000>

## Objectif

La mise en place de MLflow a pour but de tracer, comparer et reproduire toutes les expériences de modélisation menées dans le projet de prédiction des tirs réussis NBA. MLflow permet de centraliser les informations relatives aux modèles testés, d'optimiser le processus de sélection et d'assurer la transparence du pipeline data science.

## Initialisation

Une fois les dépendance pour l'authentification MLFlow déployés, il faut s'authentifier dans le container pour modifier le mot de passe administrateur par défaut puis créer un nouvel utilisateur. Enfin, créer l'expérience avec les permissions appropriés :

```
docker exec -it mlflow bash
export MLFLOW_TRACKING_USERNAME=admin
```

```

export MLFLOW_TRACKING_PASSWORD=password
python3
>>> import mlflow
>>> from mlflow.server.auth.client import AuthServiceClient
>>> client = AuthServiceClient("http://localhost:5000")
>>> client.create_user(username="parivision", password="*****")
>>> client.update_user_admin(username="parivision", is_admin=True)
>>> client.get_user("parivision").is_admin
>>>
mlflow.MlflowClient(tracking_uri="http://localhost:5000").create_experiment(name="parivision")

>>> client.update_user_password("admin", "*****")
>>> exit()
exit

```

Python

## Utilisation [↗](#)

Se référer au Notebook tutoriel pour obtenir un exemple d'utilisation :

[https://github.com/DataScientest-Studio/DEC24\\_MLOPS\\_PARIS\\_SPORTIFS/blob/master/notebooks/heuzef\\_mlflow.ipynb](https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/notebooks/heuzef_mlflow.ipynb)

## Configurer l'expérience PariVision [↗](#)

Afin de permettre le bon fonctionnement du Workflow, une expérience nommée **“parivision”** contenant au moins une run avec un alias **“champion”** est requis.

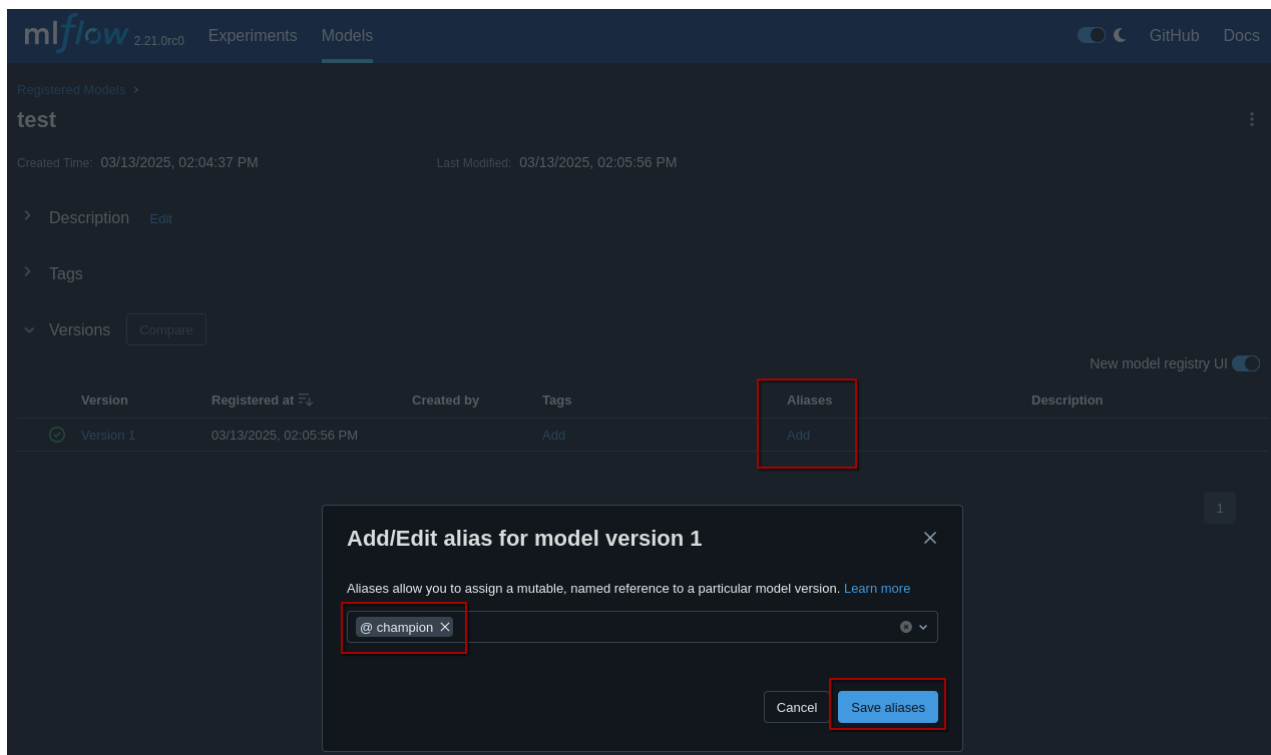
Une fois que la première run est remontée dans l'expérience **“parivision”**, il suffit de consulter cette dernière sur l'UI d'MLFlow, et d'effectuer ce premier enregistrement :

The screenshot shows the MLFlow web interface. At the top, there's a navigation bar with tabs: Overview, Model metrics, System metrics, Traces, and Artifacts. The 'Overview' tab is selected. Below the tabs, there's a 'Description' section with a link icon and the text 'No description'. Underneath is a 'Details' section with a table of experiment information:

Created at	03/13/2025, 01:57:26 PM
Created by	parivision
Experiment ID	155739054227684976
Status	Finished
Run ID	84d49be5342c46bb898534e9a1d0c682
Duration	8.2s
Datasets used	dataset (1e95)
Tags	estimator_class, estimator_name
Source	ipykernel_launcher
Logged models	sklearn
Registered models	—

Overlaid on the bottom right of the details table is a 'Register model' dialog box. It has a title bar with a close button (X). Inside, there's a 'Model' label and a dropdown menu currently showing 'parivision'. At the bottom of the dialog are two buttons: 'Cancel' and 'Register'.

Finalement, lui attribuer l'alias **“champion”** :



Nous avons défini un premier modèle Champion, qui est maintenant prêt à être challengé par les nouvelles runs.

## Mise en œuvre

### Enregistrement des expériences

À chaque entraînement d'un modèle (Régression Logistique, Random Forest, Gradient Boosting, XGBoost, LightGBM), un nouveau run MLflow est lancé. Pour chaque run, MLflow enregistre automatiquement :

- Les hyperparamètres du modèle (ex : nombre d'estimateurs, learning\_rate, max\_depth...)
- Les métriques de performance (accuracy, F1-score, ROC-AUC, etc.)
- Les artefacts produits (matrices de confusion, graphiques d'importance des variables, rapports HTML)
- Le modèle entraîné, sérialisé et versionné
- Comparaison et sélection :  
L'interface web de MLflow permet de visualiser et comparer les runs sur la base des métriques enregistrées. Cette fonctionnalité facilite l'identification du meilleur modèle et l'analyse de l'impact des hyperparamètres.

### Bénéfices

- Reproductibilité : Chaque expérience peut être rejouée à l'identique.
- Transparence : Toutes les étapes et résultats sont tracés et disponibles pour l'équipe.
- Collaboration : L'interface facilite le partage et la revue des expériences entre data scientists.

### Conclusion

MLflow s'est avéré un outil essentiel pour la gestion du cycle de vie des modèles dans ce projet. Il a permis de structurer le processus d'expérimentation, d'optimiser la sélection du modèle final, et de garantir la robustesse et la traçabilité du pipeline de modélisation.

# MLFLOW-CHAMPION

Dans le cadre de PariVision, un script sur-mesure est développé afin de permettre une interaction avec MLFlow. Ce script Python est conçu pour interagir avec un serveur MLFlow dans le cadre d'une pipeline automatisée de gestion de modèles de machine learning.

Voici une synthèse de ses actions :

1. **Configuration du Client MLFlow** : Initialise un client MLFlow pour interagir avec le serveur.
2. **Récupération des Runs** : Récupère les runs (valides) terminées du modèle spécifique PariVision.
3. **Identification du Champion actuel** : Charge le modèle actuel marqué comme "Tenant au titre" pour obtenir son identifiant de run.
4. **Détermination du nouveau Champion** : Compare les performances des runs terminées pour identifier si un nouveau "champion" (meilleur modèle) est présent.
5. **Mise à Jour du Champion** : Si un nouveau champion est identifié, il enregistre (incrémente) une nouvelle version du modèle sur MLFlow et lui attribue l'alias "champion".

En résumé, ce script automatise le processus de comparaison des performances des modèles, identifie le meilleur modèle (champion), puis met à jour le serveur MLFlow en conséquence.

Cela permet de maintenir un modèle optimal en production de manière automatisée.