# Heart Disease Prediction System – MLOps

**Course:** MLOps
**Assignment:** End-to-End MLOps Pipeline
**Dataset:** UCI Heart Disease Dataset
**Group:** Group 41
**Repository:** https://github.com/rahulvg/MLOPS-Assignment-Group-41-

## Problem Statement

The objective of this project is to design, develop, and deploy a scalable, reproducible, and production-ready machine learning system to predict the presence of heart disease based on patient health attributes.

The solution follows modern MLOps best practices, including experiment tracking, CI/CD automation, containerization, Kubernetes deployment, and monitoring.

## 1. Setup and Installation Instructions

### 1.1 Local Environment

**Python Version:** 3.10

**Install Dependencies**

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

**Run Training File(Runs experiment with different parameter on Logistics Regression and Random forest)**

```
python train/train_experiment.py
```

**Run all unit tests using Pytest**

```
pytest
```

Create report using pytest

```
pytest --html=pytest_report.html
```

**Launch MLflow UI (Local SQLite DB)**

```
mlflow ui --backend-store-uri sqlite:///mlflow.db
```

Access MLflow at:

```
http://localhost:5000
```

## 1.2 Verification of Docker Build and Execution via GitHub Actions

Due to organizational restrictions that prevent local installation of Docker Desktop, the Docker image build and container execution were verified using **GitHub Actions**, which provides a Docker-enabled runner environment.

This ensures that containerization and execution are **reproducible, verifiable, and independent of local system constraints**.

1. Navigate to the GitHub repository:
   https://github.com/rahulvg/MLOPS-Assignment-Group-41-

2. Click on the **Actions** tab in the repository.

3. Select the most recent workflow run under the **CI pipeline**.

4. Open the workflow run and inspect the following steps:

   - **Build Docker image**
     This step executes the Docker build command using the project's `Dockerfile` .

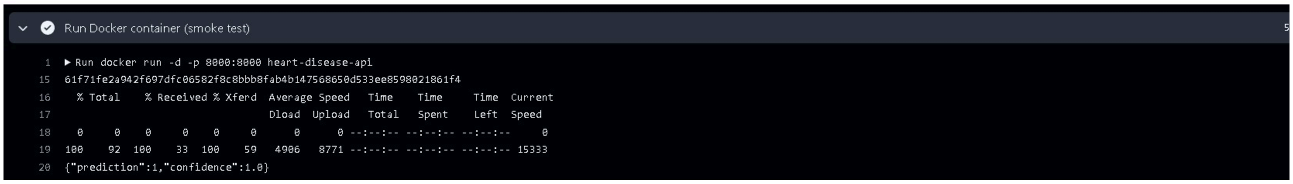
   - **Run Docker container and test API**
     This step starts the container and invokes the `/predict` endpoint using a sample JSON request.

## Evidence of Successful Docker Execution

Within the GitHub Actions workflow logs, the following evidence can be observed:

- Docker build logs confirming successful image creation
- Container startup logs indicating the FastAPI service is running
- Successful HTTP response from the `/predict` endpoint returning a prediction and confidence score

## Screenshot of successfull Docker run event



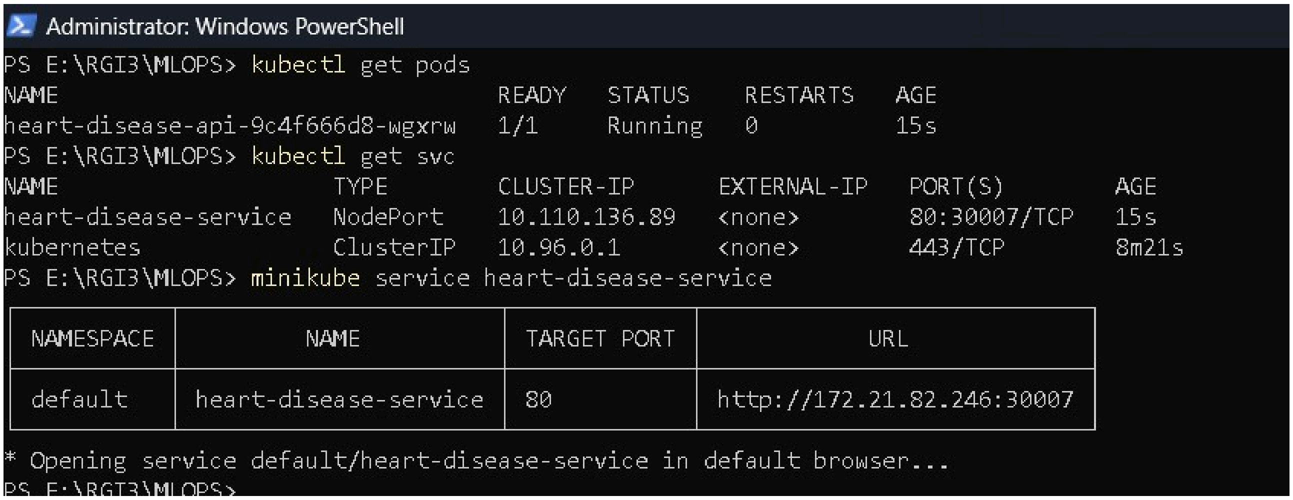
Run Docker container (smoke test)

```
▶ Run docker run -d -p 8000:8000 heart-disease-api
61f71fe2a942f697dfc06582f8c8bbb8fab4b147568650d533ee8598021861f4
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    92  100    33  100    59   4906   8771 --:--:-- --:--:-- --:--:-- 15333
{"prediction":1,"confidence":1.0}
```

### 1.3 Kubernetes (Local Deployment with Minikube)

**Start Minikube**

```
minikube start --container-runtime=containerd
```



Administrator: Windows PowerShell

```
PS E:\RGI3\MLOPS> kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
heart-disease-api-9c4f666d8-wgxrw   1/1     Running   0          15s
PS E:\RGI3\MLOPS> kubectl get svc
NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
heart-disease-service   NodePort    10.110.136.89   <none>        80:30007/TCP   15s
kubernetes              ClusterIP   10.96.0.1       <none>        443/TCP        8m21s
PS E:\RGI3\MLOPS> minikube service heart-disease-service
|-----------|-----------------------|-------------|----------------------------|
| NAMESPACE |         NAME          | TARGET PORT |            URL             |
|-----------|-----------------------|-------------|----------------------------|
| default   | heart-disease-service | 80          | http://172.21.82.246:30007 |
|-----------|-----------------------|-------------|----------------------------|
* Opening service default/heart-disease-service in default browser...
PS E:\RGI3\MLOPS>
```

### Build Docker Image Inside Minikube

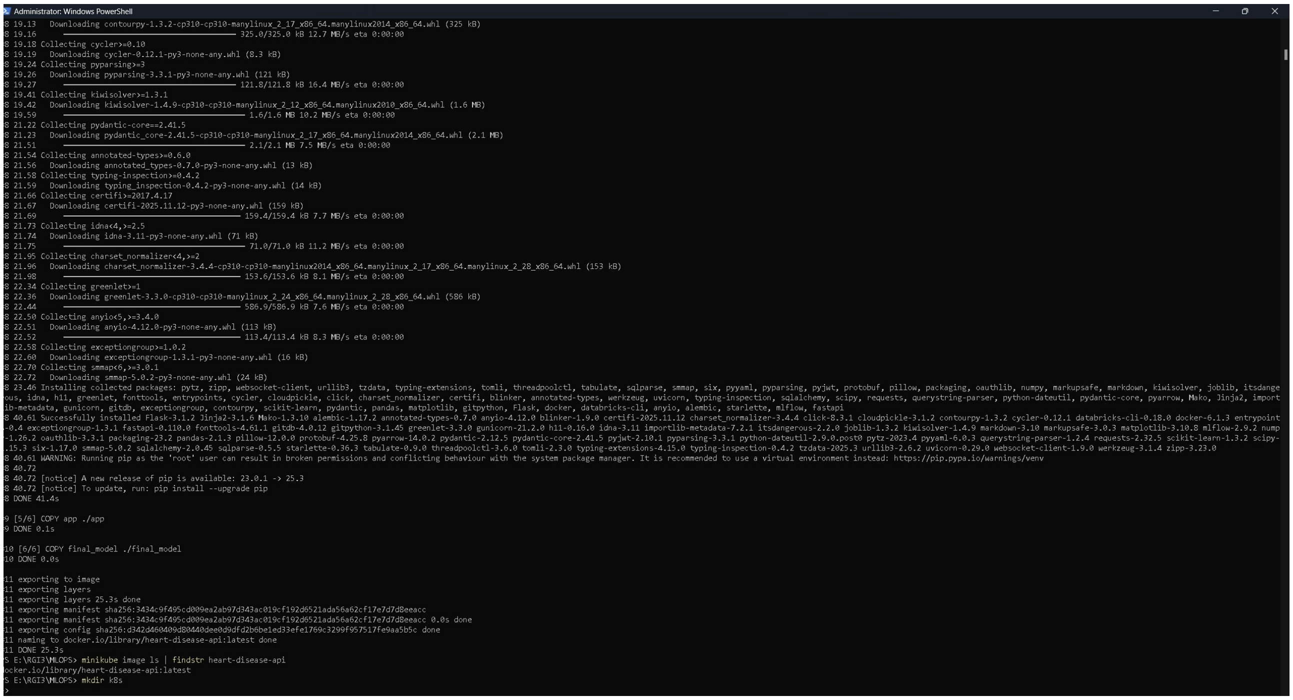
```
minikube image build -t heart-disease-api .
```

### Deploy Application

```
kubectl apply -f k8s/deployment.yaml
kubectl apply -f k8s/service.yaml
```

### Expose Service

```
minikube service heart-disease-service
```



Administrator: Windows PowerShell

```
8 19.13   Downloading contourpy-1.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (325 kB)
8 19.16      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 325.0/325.0 kB 12.7 MB/s eta 0:00:00
8 19.18 Collecting cycler>=0.10
8 19.19   Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
8 19.24 Collecting pyparsing>=3
8 19.26   Downloading pyparsing-3.3.1-py3-none-any.whl (121 kB)
8 19.27      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.8/121.8 kB 16.4 MB/s eta 0:00:00
8 19.41 Collecting kiwisolver>=1.3.1
8 19.42   Downloading kiwisolver-1.4.9-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.6 MB)
8 19.59      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 10.2 MB/s eta 0:00:00
8 21.22 Collecting pydantic-core==2.41.5
8 21.23   Downloading pydantic_core-2.41.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
8 21.51      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 7.5 MB/s eta 0:00:00
8 21.54 Collecting annotated-types>=0.6.0
8 21.56   Downloading annotated_types-0.7.0-py3-none-any.whl (13 kB)
8 21.58 Collecting typing-inspection>=0.4.2
8 21.59   Downloading typing_inspection-0.4.2-py3-none-any.whl (14 kB)
8 21.66 Collecting certifi>=2017.4.17
8 21.67   Downloading certifi-2025.11.12-py3-none-any.whl (159 kB)
8 21.69      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 159.4/159.4 kB 7.7 MB/s eta 0:00:00
8 21.73 Collecting idna<4,>=2.5
8 21.74   Downloading idna-3.11-py3-none-any.whl (71 kB)
8 21.75      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.0/71.0 kB 11.2 MB/s eta 0:00:00
8 21.95 Collecting charset_normalizer<4,>=2
8 21.96   Downloading charset_normalizer-3.4.4-cp310-cp310-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (153 kB)
8 21.98      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 153.6/153.6 kB 8.1 MB/s eta 0:00:00
8 22.34 Collecting greenlet>=1
8 22.36   Downloading greenlet-3.3.0-cp310-cp310-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (586 kB)
8 22.44      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 586.9/586.9 kB 7.6 MB/s eta 0:00:00
8 22.50 Collecting anyio<5,>=3.4.0
8 22.51   Downloading anyio-4.12.0-py3-none-any.whl (113 kB)
8 22.52      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 113.4/113.4 kB 8.3 MB/s eta 0:00:00
8 22.58 Collecting exceptiongroup>=1.0.2
8 22.60   Downloading exceptiongroup-1.3.1-py3-none-any.whl (16 kB)
8 22.70 Collecting smmap<6,>=3.0.1
8 22.72   Downloading smmap-5.0.2-py3-none-any.whl (24 kB)
8 23.46 Installing collected packages: pytz, zipp, websocket-client, urllib3, tzdata, typing-extensions, tomli, threadpoolctl, tabulate, sqlparse, smmap, six, pyyaml, pyparsing, pyjwt, protobuf, pillow, packaging, oauthlib, numpy, markupsafe, markdown, kiwisolver, joblib, itsdange
ous, idna, h11, greenlet, fonttools, entrypoints, cycler, cloudpickle, click, charset_normalizer, certifi, blinker, annotated-types, werkzeug, uvicorn, typing-inspection, sqlalchemy, scipy, requests, querystring-parser, python-dateutil, pydantic-core, pyarrow, Mako, Jinja2, import
ib-metadata, gunicorn, gitdb, exceptiongroup, contourpy, scikit-learn, pydantic, pandas, matplotlib, gitpython, Flask, docker, databricks-cli, anyio, alembic, starlette, mlflow, fastapi
8 40.61 Successfully installed Flask-3.1.2 Jinja2-3.1.6 Mako-1.3.10 alembic-1.17.2 annotated-types-0.7.0 anyio-4.12.0 blinker-1.9.0 certifi-2025.11.12 charset_normalizer-3.4.4 click-8.3.1 cloudpickle-3.1.2 contourpy-1.3.2 cycler-0.12.1 databricks-cli-0.18.0 docker-6.1.3 entrypoint
-0.4 exceptiongroup-1.3.1 fastapi-0.110.0 fonttools-4.61.1 gitdb-4.0.12 gitpython-3.1.45 greenlet-3.3.0 gunicorn-21.2.0 h11-0.16.0 idna-3.11 importlib-metadata-7.2.1 itsdangerous-2.2.0 joblib-1.3.2 kiwisolver-1.4.9 markdown-3.10 markupsafe-3.0.3 matplotlib-3.10.8 mlflow-2.9.2 nump
-1.26.2 oauthlib-3.3.1 packaging-23.2 pandas-2.1.3 pillow-12.0.0 protobuf-4.25.8 pyarrow-14.0.2 pydantic-2.12.5 pydantic-core-2.41.5 pyjwt-2.10.1 pyparsing-3.3.1 python-dateutil-2.9.0.post0 pytz-2023.4 pyyaml-6.0.3 querystring-parser-1.2.4 requests-2.32.5 scikit-learn-1.3.2 scipy-
.15.3 six-1.17.0 smmap-5.0.2 sqlalchemy-2.0.45 sqlparse-0.5.5 starlette-0.36.3 tabulate-0.9.0 threadpoolctl-3.6.0 tomli-2.3.0 typing-extensions-4.15.0 typing-inspection-0.4.2 tzdata-2025.3 urllib3-2.6.2 uvicorn-0.29.0 websocket-client-1.9.0 werkzeug-3.1.4 zipp-3.23.0
8 40.61 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
8 40.72
8 40.72 [notice] A new release of pip is available: 23.0.1 -> 25.3
8 40.72 [notice] To update, run: pip install --upgrade pip
8 DONE 41.4s

9 [5/6] COPY app ./app
9 DONE 0.1s

10 [6/6] COPY final_model ./final_model
10 DONE 0.0s

11 exporting to image
11 exporting layers
11 exporting layers 25.3s done
11 exporting manifest sha256:3434c9f495cd009ea2ab97d343ac019cf192d6521ada56a62cf17e7d7d8eeacc
11 exporting manifest sha256:3434c9f495cd009ea2ab97d343ac019cf192d6521ada56a62cf17e7d7d8eeacc 0.0s done
11 exporting config sha256:d342d460409d80440dee0d9dfd2b6be1ed33efe1769c3299f957517fe9aa5b5c done
11 naming to docker.io/library/heart-disease-api:latest done
11 DONE 25.3s
S E:\RGI3\MLOPS> minikube image ls | findstr heart-disease-api
ocker.io/library/heart-disease-api:latest
S E:\RGI3\MLOPS> mkdir k8s
>
```

# 2. Data Acquisition and Exploratory Data Analysis

## 2.1 Dataset

- Source: UCI Machine Learning Repository
- Format: CSV
- Task: Binary classification (presence or absence of heart disease)

## 2.2 Preprocessing

- Missing values handled
- Numerical features scaled using StandardScaler
- Target variable encoded
- Preprocessing implemented using a scikit-learn Pipeline

## 2.3 Exploratory Data Analysis (EDA) & Modelling choice

- Feature distributions analyzed using histograms
- Correlation heatmap used to study feature relationships
- Class balance verified

The modelling approach was guided by dataset characteristics, interpretability needs, and deployment stability.

Two models were evaluated:

- **Logistic Regression** – chosen as a strong, interpretable baseline for structured medical data
- **Random Forest –** included to capture non-linear relationships and feature interactions

All numerical features were standardized using **StandardScaler**, and preprocessing was implemented through a unified **scikit-learn Pipeline** to ensure reproducibility, prevent data leakage, and enable deployment-safe inference.

### Hyperparameter Tuning

- Logistic Regression: `C ∈ {0.1, 1.0, 10.0}`
- Random Forest:
  - `n_estimators ∈ {100, 200}`
  - `max_depth ∈ {None, 10}`

Each configuration was logged as a separate experiment using **MLflow**.

### Evaluation

Models were evaluated using **5-fold cross-validation** with the following metrics:

- Accuracy
- Precision
- Recall
- ROC-AUC

### Final Model

**Logistic Regression with C = 0.1** was selected due to:

- Consistent cross-validation performance
- Lower variance across folds
- Better generalization
- Simpler and more interpretable behavior

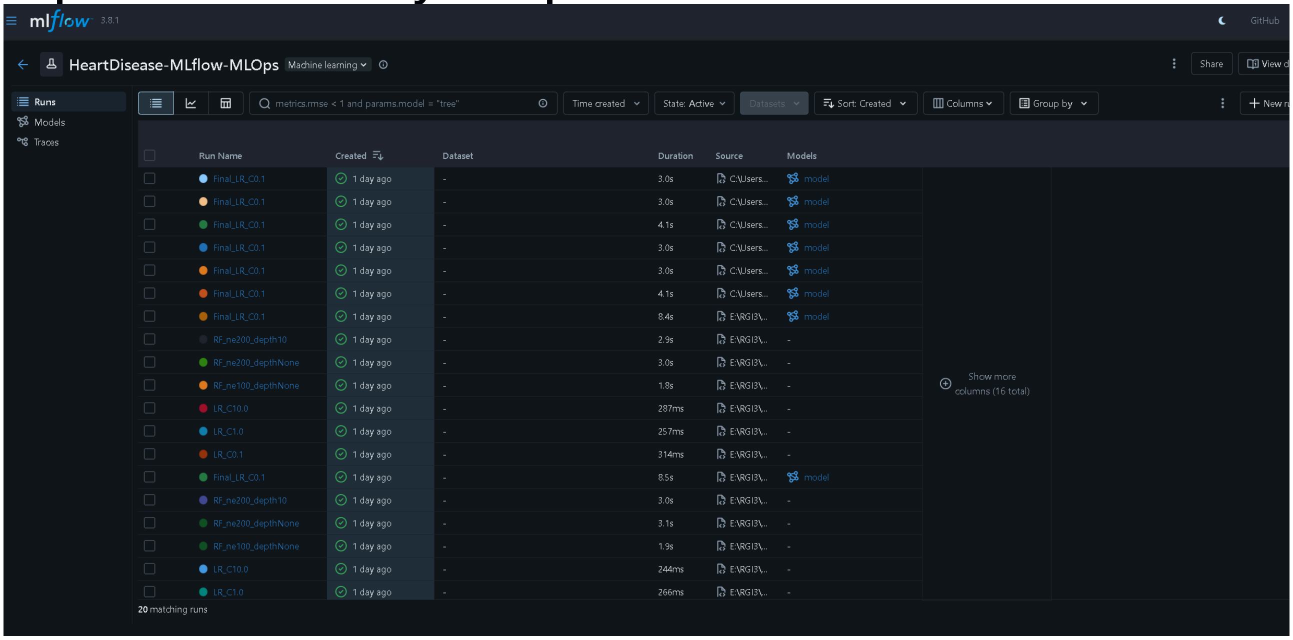
Its stability and ease of monitoring make it well-suited for a production-oriented MLOps pipeline.

# 3. Experiment Tracking

MLflow was integrated to track:

- Model parameters
- Cross-validation metrics
- Model artifacts

## All experiments are logged under a dedicated MLflow experiment for easy comparison.



| Run Name | Created | Dataset | Duration | Source | Models |
|---|---|---|---|---|---|
| Final_LR_C0.1 | 1 day ago | - | 3.0s | C:\Users... | model |
| Final_LR_C0.1 | 1 day ago | - | 3.0s | C:\Users... | model |
| Final_LR_C0.1 | 1 day ago | - | 4.1s | C:\Users... | model |
| Final_LR_C0.1 | 1 day ago | - | 3.0s | C:\Users... | model |
| Final_LR_C0.1 | 1 day ago | - | 3.0s | C:\Users... | model |
| Final_LR_C0.1 | 1 day ago | - | 4.1s | C:\Users... | model |
| Final_LR_C0.1 | 1 day ago | - | 8.4s | E:\RGI3\... | model |
| RF_ne200_depth10 | 1 day ago | - | 2.9s | E:\RGI3\... | - |
| RF_ne200_depthNone | 1 day ago | - | 3.0s | E:\RGI3\... | - |
| RF_ne100_depthNone | 1 day ago | - | 1.8s | E:\RGI3\... | - |
| LR_C10.0 | 1 day ago | - | 287ms | E:\RGI3\... | - |
| LR_C1.0 | 1 day ago | - | 257ms | E:\RGI3\... | - |
| LR_C0.1 | 1 day ago | - | 314ms | E:\RGI3\... | - |
| Final_LR_C0.1 | 1 day ago | - | 8.5s | E:\RGI3\... | model |
| RF_ne200_depth10 | 1 day ago | - | 3.0s | E:\RGI3\... | - |
| RF_ne200_depthNone | 1 day ago | - | 3.1s | E:\RGI3\... | - |
| RF_ne100_depthNone | 1 day ago | - | 1.9s | E:\RGI3\... | - |
| LR_C10.0 | 1 day ago | - | 244ms | E:\RGI3\... | - |
| LR_C1.0 | 1 day ago | - | 266ms | E:\RGI3\... | - |

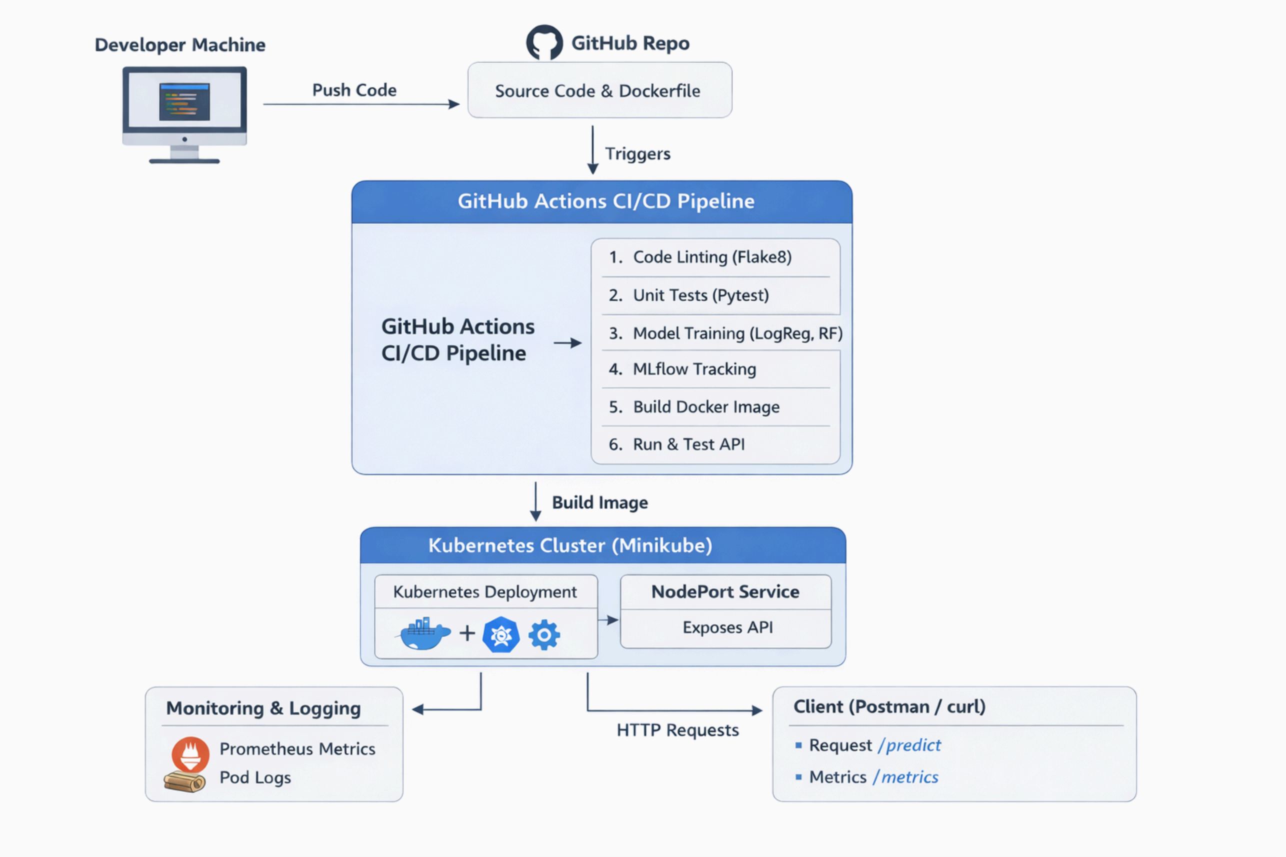
20 matching runs

---

## Model Packaging and Reproducibility

- Final model saved as a serialized scikit-learn Pipeline
- Model can be found at **final_model\heart_disease_lr_c01.pkl** in git repo.
- Preprocessing included within the model
- Reproducible inference guaranteed
- Dependencies listed in requirements.txt
- Artifacts stored and versioned using MLflow check **mlflow_experiment.db** in git repo

# 4. Architecture Diagram



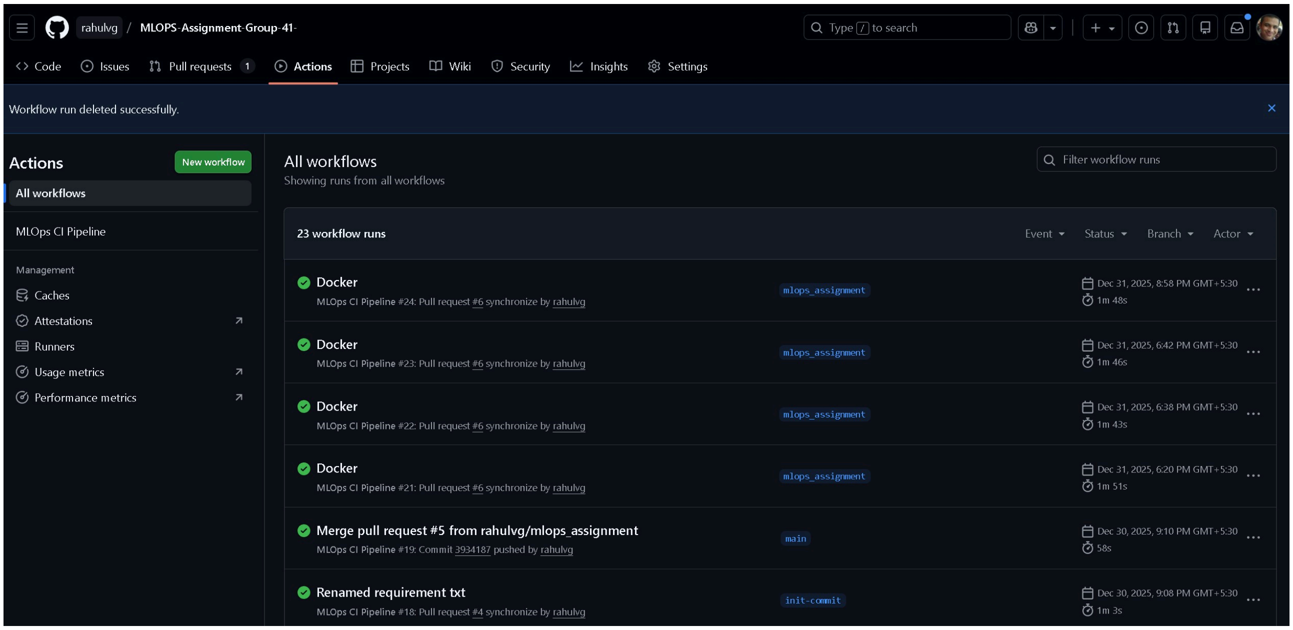
# 5. CI/CD Pipeline

## Tools Used

- GitHub Actions
- Pytest
- Flake8
- Docker

## Pipeline Stages

- Code linting
- Unit testing
- Model training
- Docker image build
- API smoke testing



Workflow run deleted successfully.

All workflows

Showing runs from all workflows

23 workflow runs

Docker
MLOps CI Pipeline #24: Pull request #6 synchronize by rahulvg
mlops_assignment
Dec 31, 2025, 8:58 PM GMT+5:30
1m 48s

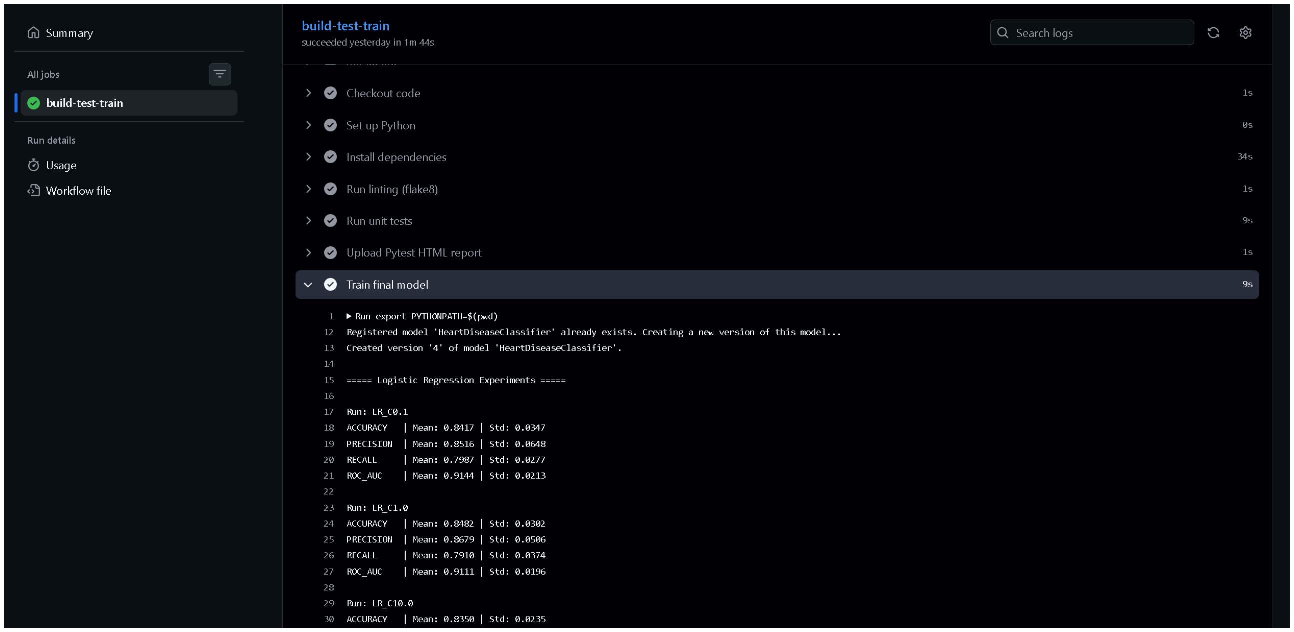
Docker
MLOps CI Pipeline #23: Pull request #6 synchronize by rahulvg
mlops_assignment
Dec 31, 2025, 6:42 PM GMT+5:30
1m 46s

Docker
MLOps CI Pipeline #22: Pull request #6 synchronize by rahulvg
mlops_assignment
Dec 31, 2025, 6:38 PM GMT+5:30
1m 43s

Docker
MLOps CI Pipeline #21: Pull request #6 synchronize by rahulvg
mlops_assignment
Dec 31, 2025, 6:20 PM GMT+5:30
1m 51s

Merge pull request #5 from rahulvg/mlops_assignment
MLOps CI Pipeline #19: Commit 3934187 pushed by rahulvg
main
Dec 30, 2025, 9:10 PM GMT+5:30
58s

Renamed requirement txt
MLOps CI Pipeline #18: Pull request #4 synchronize by rahulvg
init-commit
Dec 30, 2025, 9:08 PM GMT+5:30
1m 3s



build-test-train
succeeded yesterday in 1m 44s

Checkout code 1s
Set up Python 0s
Install dependencies 34s
Run linting (flake8) 1s
Run unit tests 9s
Upload Pytest HTML report 1s
Train final model 9s

```
▶ Run export PYTHONPATH=$(pwd)
Registered model 'HeartDiseaseClassifier' already exists. Creating a new version of this model...
Created version '4' of model 'HeartDiseaseClassifier'.

===== Logistic Regression Experiments =====

Run: LR_C0.1
ACCURACY   | Mean: 0.8417 | Std: 0.0347
PRECISION  | Mean: 0.8516 | Std: 0.0648
RECALL     | Mean: 0.7987 | Std: 0.0277
ROC_AUC    | Mean: 0.9144 | Std: 0.0213

Run: LR_C1.0
ACCURACY   | Mean: 0.8482 | Std: 0.0302
PRECISION  | Mean: 0.8679 | Std: 0.0506
RECALL     | Mean: 0.7910 | Std: 0.0374
ROC_AUC    | Mean: 0.9111 | Std: 0.0196

Run: LR_C10.0
ACCURACY   | Mean: 0.8350 | Std: 0.0235
```

# 6. Code Repository

https://github.com/rahulvg/MLOPS-Assignment-Group-41-

# 7. Containerization and Deployment

## 7.1 Dockerized API

- FastAPI-based service
- /predict endpoint
- Accepts JSON input

- Returns prediction and confidence score

### 7.2 Kubernetes Deployment

- Local Kubernetes using Minikube
- Deployment and NodePort Service manifests
- API tested using curl and Postman

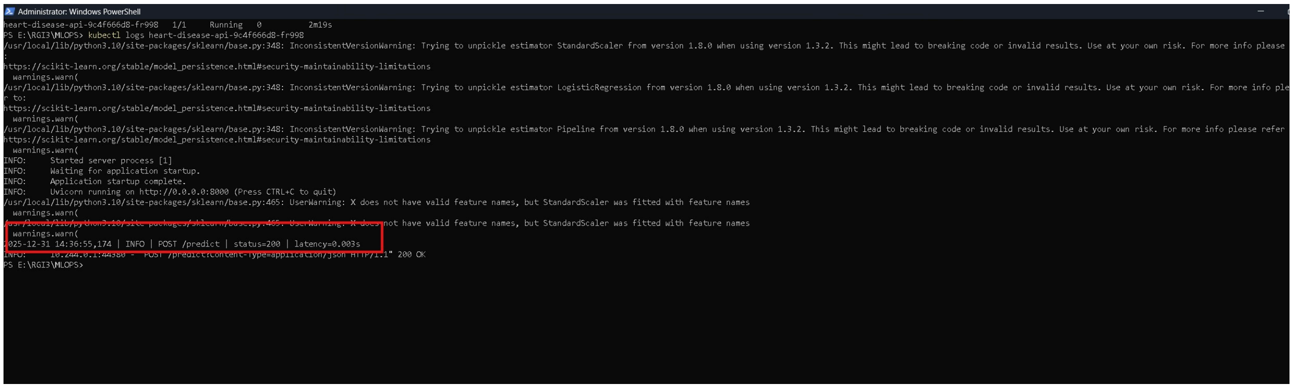
## 7. Monitoring and Logging

### 7.1 Logging

- Request-level logging implemented via FastAPI middleware
- Logs include endpoint, HTTP status, and latency
- Logs accessible via Kubernetes pod logs

### 7.2 Monitoring

- Prometheus-compatible /metrics endpoint exposed
- Metrics include request count and request latency
- Ready for Prometheus and Grafana integration



```
heart-disease-api-9c4f666d8-fr998   1/1     Running   0          2m19s
PS E:\RGI3\MLOPS> kubectl logs heart-disease-api-9c4f666d8-fr998
/usr/local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator StandardScaler from version 1.8.0 when using version 1.3.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please
:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.8.0 when using version 1.3.2. This might lead to breaking code or invalid results. Use at your own risk. For more info ple
r to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator Pipeline from version 1.8.0 when using version 1.3.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
/usr/local/lib/python3.10/site-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/site-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
2025-12-31 14:36:55,174 | INFO | POST /predict | status=200 | latency=0.003s
INFO:     10.244.0.1:44380 - "POST /predict?content-type=application/json HTTP/1.1" 200 OK
PS E:\RGI3\MLOPS>
```

## 8. Architecture Overview

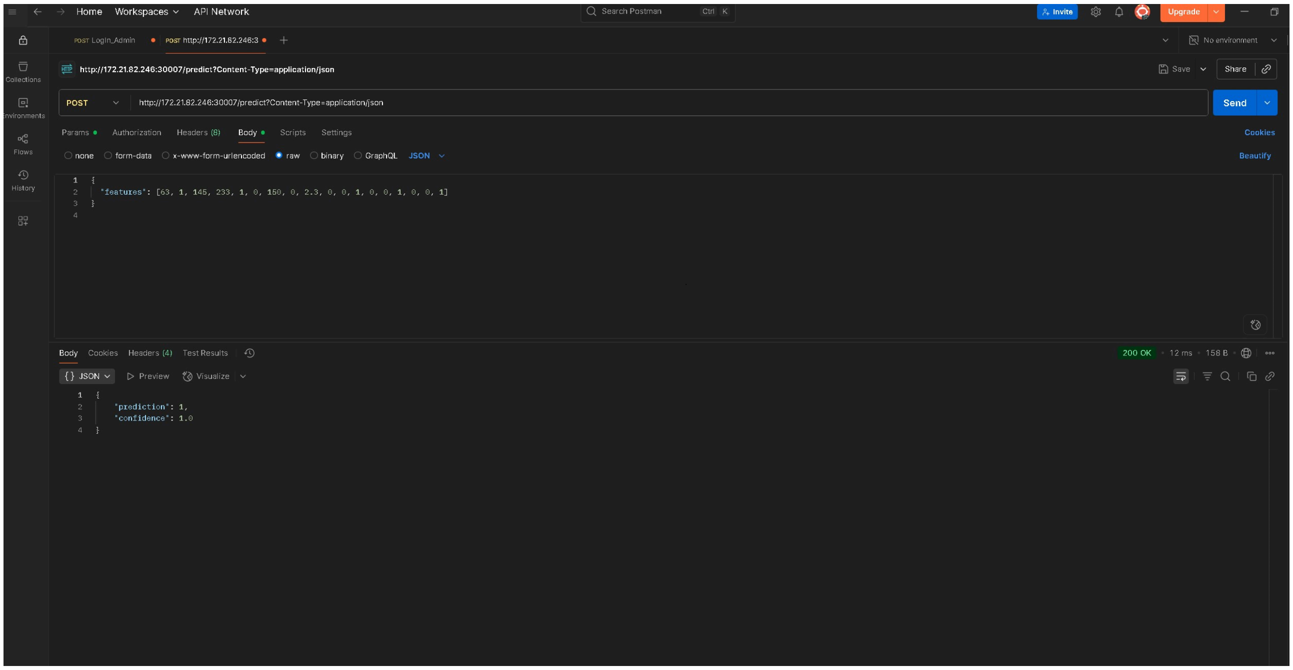
Client (Postman / curl)
→ FastAPI API (/predict, /metrics)
→ Scikit-learn Pipeline
→ Kubernetes Pod
→ NodePort Service

CI/CD is handled using GitHub Actions, and experiment tracking is handled using MLflow.

## 9 Run using CURL/Postman API

Curl Command

```
"curl -X POST http://127.0.0.1:30007/predict \
-H "Content-Type: application/json" \
-d '{"features":[63,1,145,233,1,0,150,0,2.3,0,0,1,0,0,1,0,0,1]}'"
```



http://172.21.82.246:30007/predict?Content-Type=application/json

POST http://172.21.82.246:30007/predict?Content-Type=application/json

```
{
  "features": [63, 1, 145, 233, 1, 0, 150, 0, 2.3, 0, 0, 1, 0, 0, 1, 0, 0, 1]
}
```

200 OK • 12 ms • 158 B

```
{
    "prediction": 1,
    "confidence": 1.0
}
```

## Conclusion

This project demonstrates a complete, production-grade MLOps workflow covering data analysis, model development, experiment tracking, CI/CD automation, containerization, Kubernetes deployment, and monitoring.

The system is scalable, reproducible, and aligned with real-world MLOps practices.

## Demo video

[https://github.com/rahulvg/MLOPS-Assignment-Group-41-/blob/main/Demo%20Video.mp4](https://github.com/rahulvg/MLOPS-Assignment-Group-41-/blob/main/Demo%20Video.mp4) https://drive.google.com/drive/folders/1hGcu4oyM3TusMy8vnackoOLpug7LhJdP?usp=sharing

## Appendix: Useful Commands

### Launch MLflow with Custom Local DB

```
mlflow ui --backend-store-uri sqlite:///E:/RGI3/MLOPS/mlflow.db
```

### Rebuild and Redeploy on Minikube

```
minikube image build -t heart-disease-api .
kubectl delete deployment heart-disease-api
kubectl apply -f k8s/deployment.yaml
```