

Setup/install instructions

Prerequisites

- Git
- Python 3.7.x or higher
- pip
- Docker
- Google Cloud Account
- gcloud CLI

Gcloud Configuration

- Verify gcloud installation using:
gcloud version
- Login and set project:
gcloud auth login
gcloud config set project "PROJECT_ID"

Clone the Repository

- git clone https://github.com/rtgarg1991/MLOps_Assignment.git
- cd MLOps_Assignment

Python Environment Setup

- Create virtual environment:
python3 -m venv venv
- Activate environment:
source venv/bin/activate
- Install dependencies:
pip install -r requirements.txt

Run Unit Tests

- Run tests to verify setup:
pytest tests/

Run Lint Tests

- Run Lint steps to identify any lint issues
flake8 src

Kubernetes Deployment Using GKE

- Create cluster:
gcloud container clusters create mlops-cluster --zone us-central1-a
- Connect to cluster:
gcloud container clusters get-credentials mlops-cluster --zone us-central1-a
- Deploy application:
kubectl apply -f k8s/

- Check status:
`kubectl get pods`
`kubectl get services`

EDA and modelling choices

- Performed EDA on the raw UCI Heart Disease dataset to understand data quality and feature behavior before preprocessing and modeling.
- Analyzed missing values across all features using a bar plot to identify incomplete clinical attributes and plan appropriate handling strategies.
- Examined distributions of numeric features (age, trestbps, chol, thalach, oldpeak) using histograms to understand spread, skewness, and potential outliers.
- Generated a correlation heatmap between numeric features and the target variable to identify strong predictors and check for multicollinearity.
- Used the target variable only for analysis purposes, ensuring no data leakage into feature engineering.

Feature Engineering

- Explicitly categorized features into numeric and categorical groups based on domain understanding of the dataset.
- Applied one-hot encoding to categorical features to convert diagnostic and binary variables into machine-readable format without imposing any ordinal relationships.
- Retained all categorical levels during encoding to keep the feature set complete and compatible with different model types.
- Scaling and normalization were performed for numeric features.

Modeling Choices

- The modeling pipeline was designed to support both experimentation and production training, with clear separation between feature preparation, model training, evaluation, and artifact logging.
- The dataset is split into training and testing sets. An 80-20 train-test split is used for standard training to ensure sufficient data for both learning and evaluation.
- Two models were implemented to balance interpretability and performance
 - Logistic Regression
 - Random Forest
- Logistic Regression is treated as a baseline model due to its simplicity and interpretability, which is important for healthcare-related predictions.

- Random Forest is used to capture non-linear relationships and feature interactions that may not be handled well by linear models.
- Multiple evaluation metrics are computed to provide a comprehensive view of model performance
 - Accuracy
 - Precision
 - F1-score
 - ROC-AUC
- These metrics help evaluate not only overall correctness but also class-wise performance, which is critical in medical diagnosis tasks.

Model Training

- Model configuration parameters (such as model type and hyperparameters) are logged to MLflow for experiment tracking.
- Each training run is executed within an MLflow experiment to ensure consistency and reproducibility.

Experiment tracking summary

- We have integrated MLflow for experiment tracking in our project.
- MLflow is used to track experiments by logging parameters, metrics, and artifacts for each model run.
- Each experiment is defined using a logical experiment name:
`mlflow.set_experiment("Predict_Risk_Of_Heart_Disease")`

This experiment groups all related model runs for the heart disease prediction task, enabling easy comparison across different algorithms and configurations.

- Each training execution creates a unique MLflow run under a single experiment.
- Parameters, metrics (accuracy, precision, recall, F1, ROC-AUC) are logged.
- Artifacts such as trained models, confusion matrix, ROC and PR curves are stored.
- Ensures reproducibility, comparison across models, and auditability.

Below table summarizes the final experiment runs, based on which the best-performing model is automatically selected for deployment. Among the evaluated models, the Random Forest model is chosen for deployment.

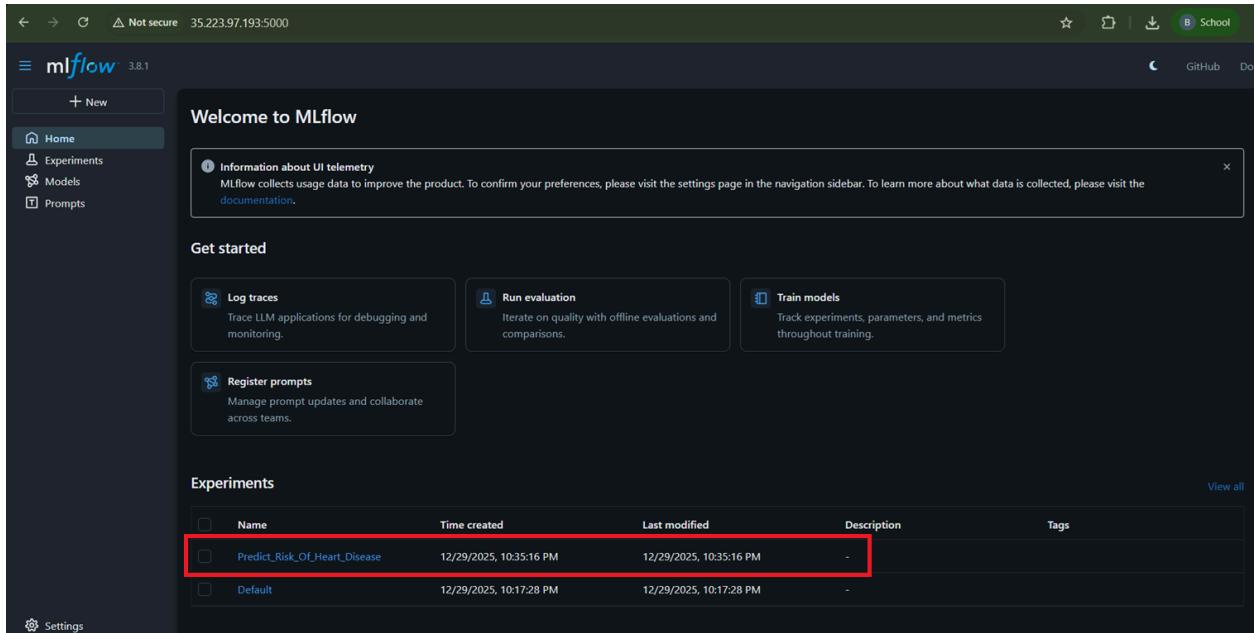
Experiment timestamp	Model Name	Run ID	Accuracy	Precision	Recall	F1 score	Roc_auc
01/04/2026, 02:25:11 PM	Logistic Regression	dc068f13 3a2a43f1 904649f2 84acb57 0	85.24%	82.76%	85.71%	84.21%	95.23%
01/04/2026, 02:25:11 PM	Random forest	7602dfbb b8f04058 b8d1a7b 918bf452 2	86.88%	81.25%	92.85%	86.66%	94.20%

MLflow Dashboard url:

http://35.223.97.193:5000/#/experiments/1/runs?searchFilter=&orderByKey=attributes.start_time&orderByAsc=false&startTime=ALL&lifecycleFilter=Active&modelVersionFilter=All+Runs&datasetsFilter=W10%3D

The dashboard displays all experiment runs conducted in this project, enabling comparison across models, parameters, metrics, and artifacts. Please refer to the dashboard for previous experiment details.

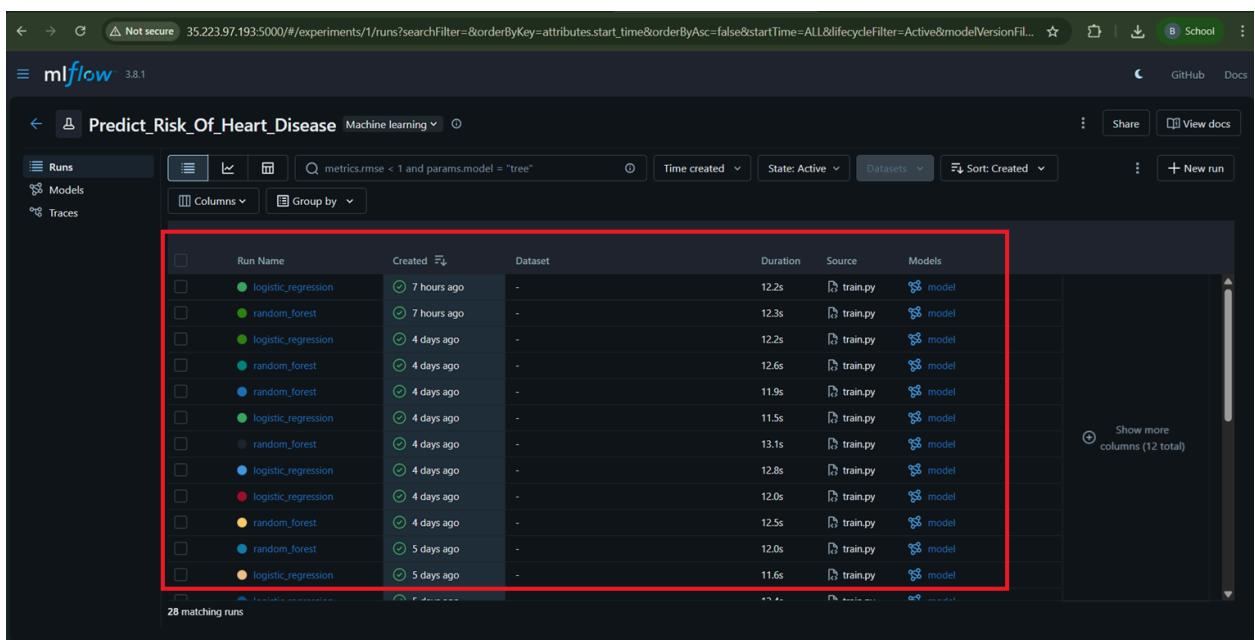
As shown below, experiment group has been created with name-
“Predict_Risk_Of_Heart_Disease”



The screenshot shows the MLflow UI homepage. In the 'Experiments' section, there is a table with two rows. The first row, which is highlighted with a red box, corresponds to the experiment group 'Predict_Risk_Of_Heart_Disease'. The second row is for the 'Default' experiment.

Name	Time created	Last modified	Description	Tags
Predict_Risk_Of_Heart_Disease	12/29/2025, 10:35:16 PM	12/29/2025, 10:35:16 PM	-	-
Default	12/29/2025, 10:17:28 PM	12/29/2025, 10:17:28 PM	-	-

All the different experiments and runs of different models have been maintained under
“Predict_Risk_Of_Heart_Disease”.



The screenshot shows the 'Runs' section for the experiment group 'Predict_Risk_Of_Heart_Disease'. The table lists 28 matching runs, each with details such as Run Name, Created time, Dataset, Duration, Source, and Models. A red box highlights the first few rows of the table.

Run Name	Created	Dataset	Duration	Source	Models
logistic_regression	7 hours ago	-	12.2s	train.py	model
random_forest	7 hours ago	-	12.3s	train.py	model
logistic_regression	4 days ago	-	12.2s	train.py	model
random_forest	4 days ago	-	12.6s	train.py	model
random_forest	4 days ago	-	11.9s	train.py	model
logistic_regression	4 days ago	-	11.5s	train.py	model
random_forest	4 days ago	-	13.1s	train.py	model
logistic_regression	4 days ago	-	12.8s	train.py	model
logistic_regression	4 days ago	-	12.0s	train.py	model
random_forest	4 days ago	-	12.5s	train.py	model
random_forest	5 days ago	-	12.0s	train.py	model
logistic_regression	5 days ago	-	11.6s	train.py	model

Below screenshot shows Metrics and parameters captured when running logistic regression model as experiment.

The screenshot shows the mlflow UI for a single experiment run. The URL is 35.223.97.193:5000/#/experiments/1/runs/dc068f133a2a43f1904649f284acb570. The experiment name is "Predict_Risk_Of_Heart_Disease". The run name is "logistic_regression".

Metrics (5)

Metric	Value	Models
accuracy	0.8524500163934426	model
precision	0.8275862068965517	model
recall	0.8571428571428571	model
f1	0.8421052631578947	model
roc_auc	0.9523809523809524	model

Parameters (2)

Parameter	Value
model type	logistic regression

About this run

Created at	01/04/2026, 02:25:11 PM
Created by	root
Experiment ID	1
Status	Finished
Run ID	dc068f133a2a43f1904649f284acb570
Duration	12.2s
Source	train.py
Registered prompts	—

Datasets

None

Tags

model: logistic_regression

Registered models

None

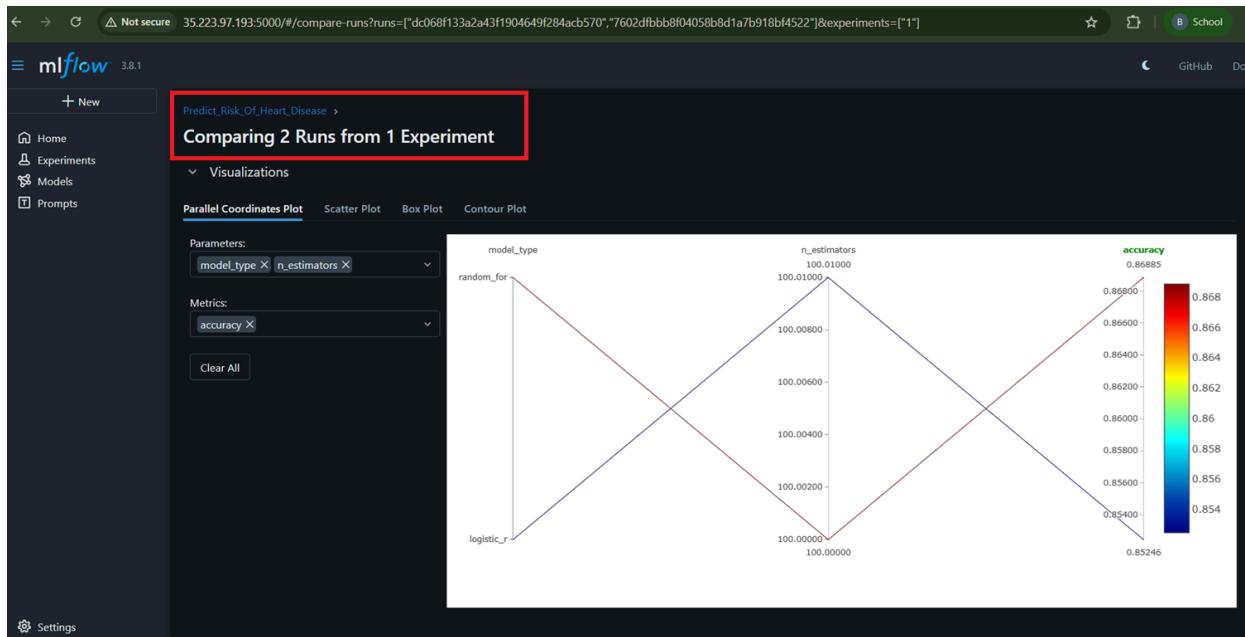
The screenshots below demonstrate a side-by-side comparison of two experiments and their evaluation metrics.

The screenshot shows the mlflow UI comparing multiple runs. The URL is 35.223.97.193:5000/#/experiments/1/runs?searchFilter=&orderByKey=attributes.start_time&orderByAsc=false&startTime=ALL&lifecycleFilter=Active&modelVersionFilter=All... The experiment name is "Predict_Risk_Of_Heart_Disease".

The "Compare" button is highlighted with a red box. Two runs are selected for comparison: "logistic_regression" and "random_forest".

Run Name	Created	Dataset	Duration	Source	Models
logistic_regression	1 day ago	-	12.2s	train.py	model
random_forest	1 day ago	-	12.3s	train.py	model
logistic_regression	5 days ago	-	12.2s	train.py	model
random_forest	5 days ago	-	12.6s	train.py	model
logistic_regression	5 days ago	-	11.9s	train.py	model
random_forest	5 days ago	-	11.5s	train.py	model
logistic_regression	5 days ago	-	13.1s	train.py	model
random_forest	5 days ago	-	12.8s	train.py	model
logistic_regression	6 days ago	-	12.0s	train.py	model
random_forest	6 days ago	-	12.5s	train.py	model
logistic_regression	6 days ago	-	12.0s	train.py	model
random_forest	6 days ago	-	11.6s	train.py	model
logistic_regression	6 days ago	-	12.4s	train.py	model
random_forest	6 days ago	-	12.7s	train.py	model

28 matching runs



mlflow 3.8.1

+ New

Home Experiments Models Prompts Settings

Run details

Run ID:	dc068f133a2a43f1904649f284acb570	7602dfbbb8f04058b8d1a7b918bf4522
Run Name:	logistic_regression	random_forest
Start Time:	01/04/2026, 02:25:11 PM	01/04/2026, 02:25:11 PM
End Time:	01/04/2026, 02:25:23 PM	01/04/2026, 02:25:23 PM
Duration:	12.2s	12.3s

Parameters

Show diff only

isExperiment	False	False
model_type	logistic_regression	random_forest
n_estimators		100

Metrics

Show diff only

mlflow 3.8.1

Metrics

Show diff only

accuracy	0.852	0.869
f1	0.842	0.867
precision	0.828	0.813
recall	0.857	0.929
roc_auc	0.952	0.942

Artifacts

No common artifacts to display.

Tags

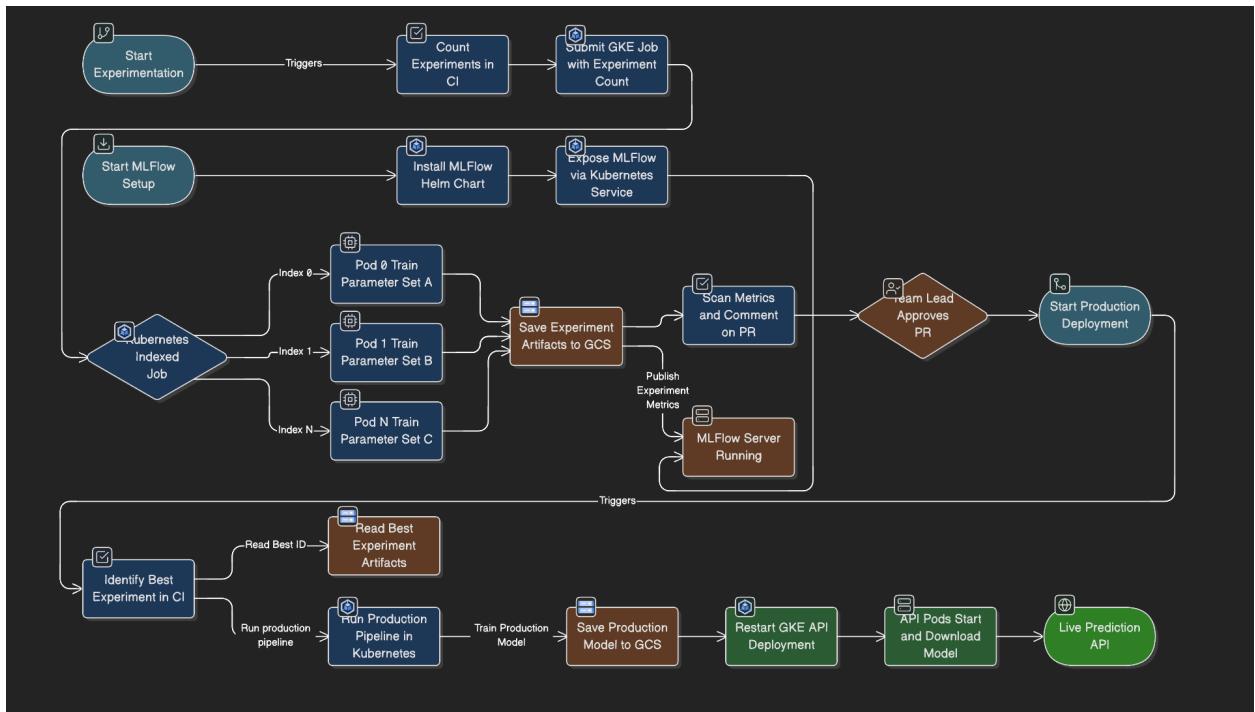
Show diff only

model logistic_regression random_forest

Settings

Architecture diagram

The architecture diagram is created using Mermaid diagramming tool



It has two separated phases:

- **Experimentation phase** : try many model configurations to identify best candidates
- **Production deployment phase** : train and deploy only the best model

Everything is **CI-driven, Kubernetes-based**, and **artifact-centric** (GCS buckets).

Experimentation Phase

This phase is about trying many model variants in parallel and deciding which one is best.

- Developer & CI trigger
 - A developer pushes code to a feature branch
 - This automatically triggers CI
 - Count how many experiments need to be run. Example: different hyperparameters, models, seeds, etc.
- Submitting experiments to Kubernetes
 - CI submits a GKE job with the experiment count
 - Kubernetes runs an indexed job
 - Each index = one experiment. This allows parallel training
- Parallel model training (indexed jobs)
 - Kubernetes spawns multiple pods:
 - **Pod 0** : trains with parameter set A
 - **Pod 1** : trains with parameter set B
- Storing experiment results
 - Each training pod uploads training metrics to GCP and MLFlow
- Automated evaluation & PR feedback
 - Read metrics from the experiment bucket. Comments results directly on the Pull Request
- Human approval gate
 - A team lead reviews and if satisfied and the Pull Request is approved

Production Deployment Phase

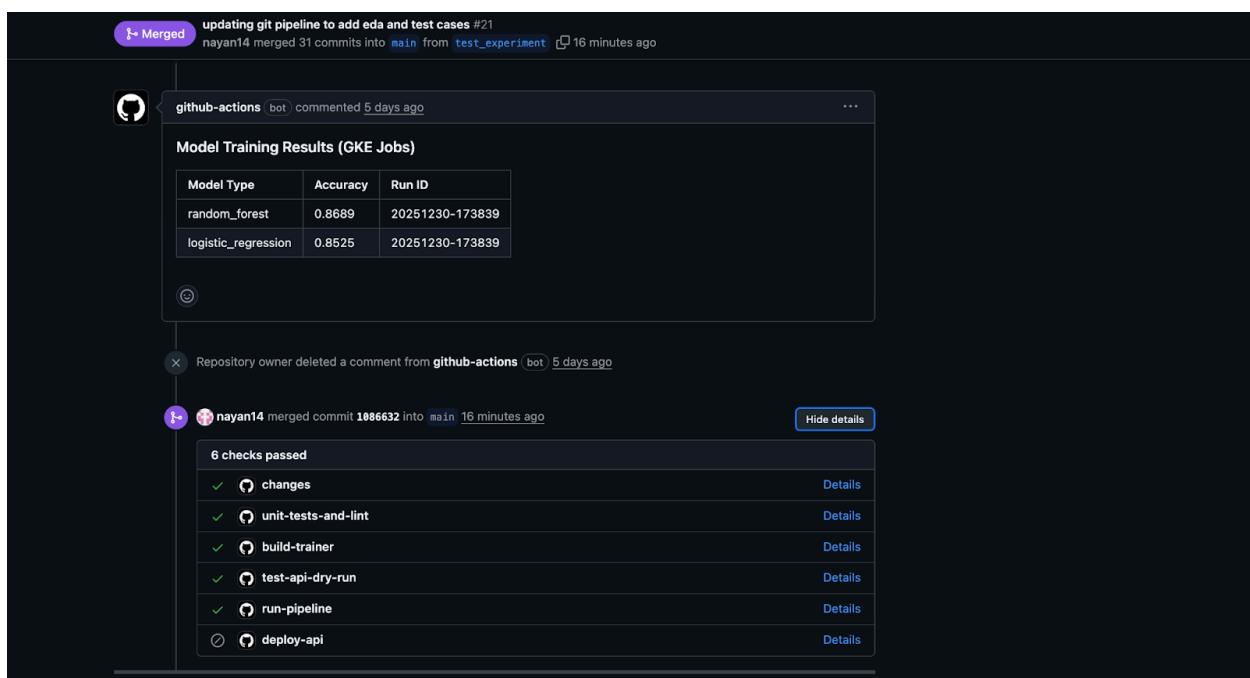
This phase starts only after PR approval and merge.

- Selecting the best experiment

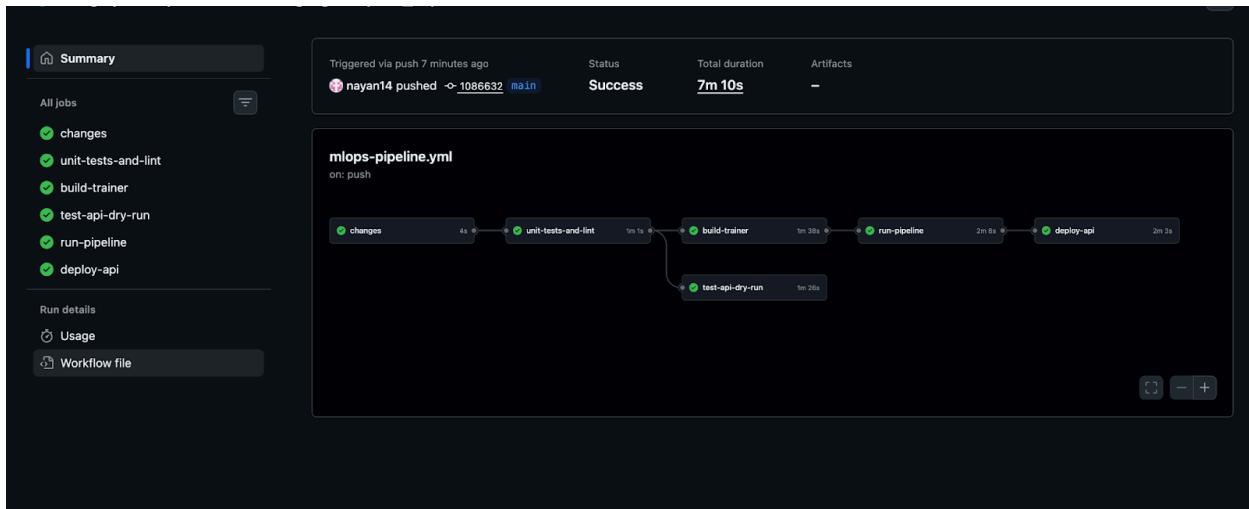
- Read all experiment results. Identifies the best experiment ID
- Production training job
 - CI launches a Kubernetes indexed job for production
- Storing the production model
 - The trained production model is saved to GCS bucket for production models
- Deploying the new model
 - CI restarts the GKE API deployment
 - API pods:
 - Download the latest production model from GCS
- Live inference
 - API pods are now serving Live prediction requests

CI/CD and deployment workflow screenshots

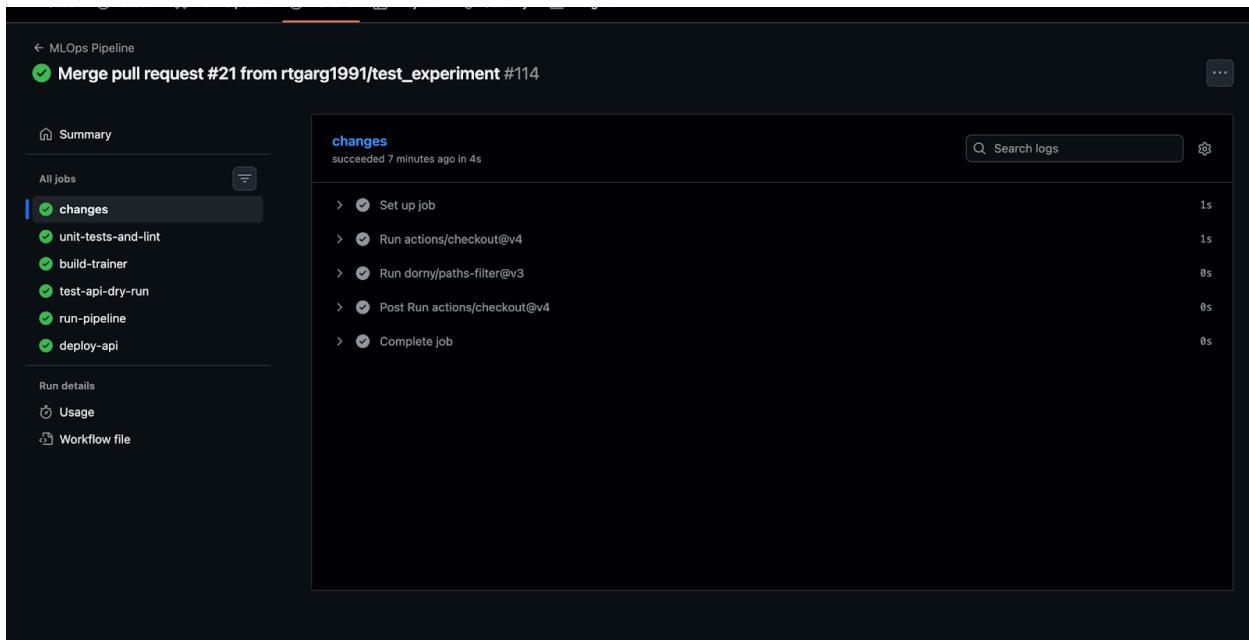
PR is created on Github. Triggers github actions



All stages and tasks defined as part of github actions



Github actions dynamically find out the stages/tasks that need to be triggered.



Unit test cases and Eslint is executed

The screenshot shows a CI pipeline interface. On the left, a sidebar lists several jobs: changes, unit-tests-and-lint, build-trainer, test-api-dry-run, run-pipeline, and deploy-api. The 'unit-tests-and-lint' job is currently selected and highlighted with a blue border. The main panel displays the details for this job. The title is 'unit-tests-and-lint' and it is noted as having succeeded 7 minutes ago in 1m 1s. Below the title is a list of steps, each with a checkmark indicating success and a timestamp: Set up job (1s), Run actions/checkout@v4 (0s), Set up Python (0s), Install Dependencies (51s), Run Lint (0s), Run UnitTests (6s), Post Set up Python (0s), Post Run actions/checkout@v4 (0s), and Complete job (0s). A search bar labeled 'Search logs' and a gear icon for settings are also visible.

Docker images are built for experimentation

The screenshot shows a CI pipeline interface, similar to the previous one. The sidebar lists the same set of jobs: changes, unit-tests-and-lint, build-trainer, test-api-dry-run, run-pipeline, and deploy-api. The 'build-trainer' job is selected and highlighted with a blue border. The main panel shows the details for the 'build-trainer' job. The title is 'build-trainer' and it succeeded 5 minutes ago in 1m 38s. The steps listed are: Set up job (1s), Run actions/checkout@v4 (1s), Run google-github-actions/auth@v2 (0s), Configure Docker (4s), Build and Push Trainer (1m 28s), Post Run google-github-actions/auth@v2 (0s), Post Run actions/checkout@v4 (0s), and Complete job (0s). The interface includes a search bar for logs and a settings gear icon.

Pipeline is triggered to run various steps of EDA, pre-processing, feature engineering and model training

The screenshot shows a GitHub Actions pipeline run for a merge request. The pipeline consists of several jobs: changes, unit-tests-and-lint, build-trainer, test-api-dry-run, run-pipeline, and deploy-api. The 'run-pipeline' job is currently selected. It has succeeded 3 minutes ago in 2m 8s. The job log details the following steps:

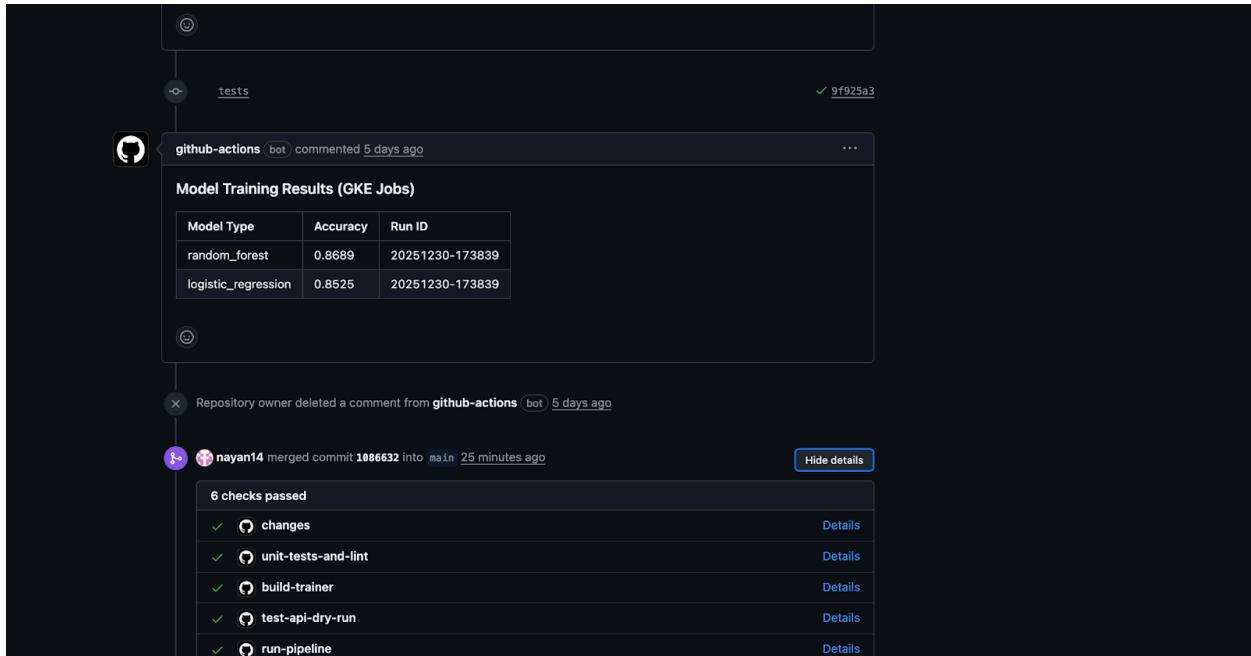
Step	Description	Time
> ✓ Set up job		2s
> ✓ Run actions/checkout@v4		0s
> ✓ Run google-github-actions/auth@v2		1s
> ✓ Run google-github-actions/setup-gcloud@v2		20s
> ✓ Run google-github-actions/get-gke-credentials@v2		1s
> ✓ Install envsubst		2s
> ✓ Set Variables		0s
> ✓ Run Ingest		35s
> ✓ Run EDA		16s
> ✓ Run Pre-processing		7s
> ✓ Run Feature Engineering		7s
> ✓ Promote Artifacts to Production		6s
> ✓ Run Training		29s
⌚ Comment Results on PR		0s
> ✓ Post Run google-github-actions/auth@v2		0s

Best candidate model is deployed in GKS (Kubernetes cluster)

The screenshot shows a GitHub Actions pipeline run for a merge request. The pipeline consists of several jobs: changes, unit-tests-and-lint, build-trainer, test-api-dry-run, run-pipeline, and deploy-api. The 'deploy-api' job is currently selected. It has succeeded 2 minutes ago in 2m 3s. The job log details the following steps:

Step	Description	Time
> ✓ Set up job		1s
> ✓ Run actions/checkout@v4		1s
> ✓ Run google-github-actions/auth@v2		0s
> ✓ Run google-github-actions/setup-gcloud@v2		22s
> ✓ Run gcloud auth configure-docker \$REGION-docker.pkg.dev		1s
> ✓ Run google-github-actions/get-gke-credentials@v2		0s
> ✓ Download Model (Smart Fallback)		4s
> ✓ Build & Push API		1m 26s
> ✓ Deploy Service		5s
> ✓ Post Run google-github-actions/auth@v2		0s
> ✓ Post Run actions/checkout@v4		0s
> ✓ Complete job		0s

Experimentation results are shown on PR before merging



GKS cluster

The screenshot shows the Google Cloud Kubernetes Engine clusters dashboard. The left sidebar has sections for Resource Management (Clusters selected), Workloads, AI/ML, Teams, Applications, Secrets & ConfigMaps, Storage, Object Browser, Upgrades (New), and Backup for GKE. Posture Management sections include Security and Marketplace. The main dashboard shows an 'Overview' tab with cluster statistics: Health (100% healthy), Upgrade (100% up to date), and Estimated monthly cost (₹0.00 / month - 0%). The 'Utilization' tab shows a table with one row for the 'mlops-cluster' cluster:

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
Green checkmark	mlops-cluster	us-central1-a	1	2	8 GB	Create a backup plan Set maintenance window	-

Cluster Nodes

The screenshot shows the 'Cluster details' page for the 'mllops-cluster' in the 'mllops-final' project. The left sidebar has 'Clusters' selected under 'Resource Management'. The main area shows 'Node pools' with one entry: 'default-pool' (Status: Ok, Version: 1.33.5-gke.1308000, 1 node, e2-standard-2 machine type, Container-Optimized OS with containedr image type, Off autoscaling, Default IPv4 Pod IP address range: 10.108.0.0/14). Below it is a 'Nodes' section with one node listed: 'gke-mllops-cluster-default-pool-64a92117-b835' (Status: Ready, Node type: User-managed, CPU requested: 926 mCPU, CPU allocatable: 1.93 CPU, Memory requested: 1.47 GB, Memory allocatable: 6.32 GB).

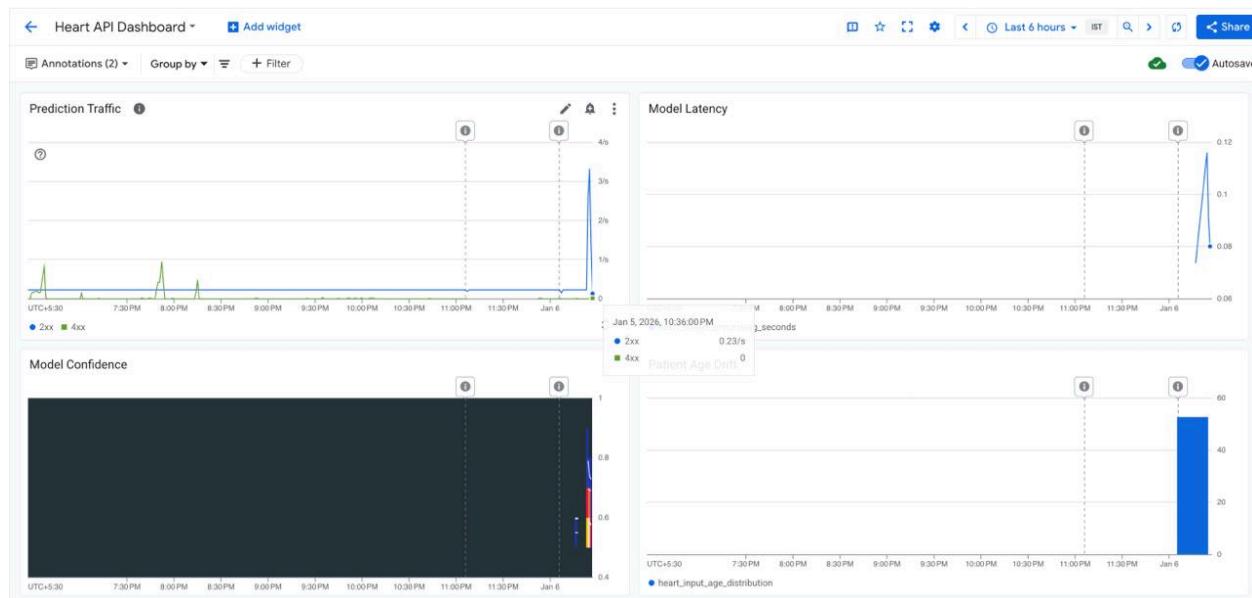
Model Artifacts stored through CI/CD pipeline

The screenshot shows the 'Buckets' page in Cloud Storage. The left sidebar has 'Buckets' selected under 'Cloud Storage'. The main area lists one bucket: 'mllops-final-001-mllops-artifacts' (Created: Dec 25, 2025, 6:50:10PM, Location type: Region, Location: us-central1, Default storage class: Standard, Last modified: Dec 25, 2025, 6:50:10PM, Public). The 'Storage Intelligence' section below is collapsed.

Different buckets to store relevant data

The screenshot shows the Google Cloud Storage interface. On the left, a sidebar navigation includes 'Cloud Storage', 'Overview', 'Buckets' (which is selected), 'Monitoring', and 'Settings'. Below these are sections for 'Storage Intelligence', 'Insights datasets', and 'Configuration'. The main content area is titled 'mllops-final-001-mllops-artifacts'. It displays bucket metadata: Location (us-central1 (Iowa)), Storage class (Standard), Public access (Subject to object ACLs), and Protection (Soft Delete). A tab bar at the top of the main content includes 'Objects' (selected), 'Configuration', 'Permissions', 'Protection', 'Lifecycle', 'Observability', 'New', 'Inventory Reports', and 'Operations'. The 'Objects' section shows a folder browser for 'mllops-final-001-mllops-artifacts'. It lists subfolders: 'data/' (2 items), 'experiments/' (2 items), 'latest/' (2 items), 'miflow/' (2 items), and 'production/' (2 items). Each folder entry includes columns for Name, Size, Type, Created, Storage class, and Last modified.

Observability





Link to code repository

https://github.com/rtgarg1991/MLOps_Assignment