

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.

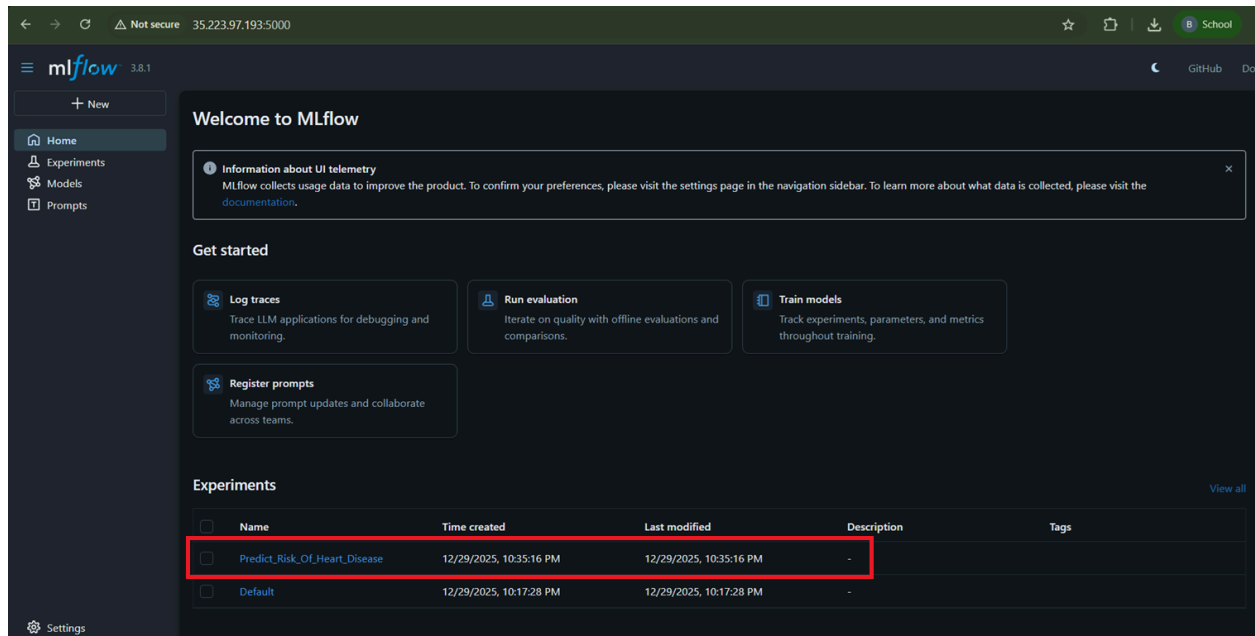
Experiment timestamp	Model Name	Run ID	Accuracy	Precision	Recall	F1 score	Roc_auc
01/04/2026, 02:25:11 PM	Logistic Regression	dc068f133a2a43f1904649f284acb570	85.24%	82.76%	85.71%	84.21%	95.23%
01/04/2026, 02:25:11 PM	Random forest	7602dfbb8f04058b8d1a7b918bf4522	86.88%	81.25%	92.85%	86.66%	94.20%

MLflow Dashboard url:

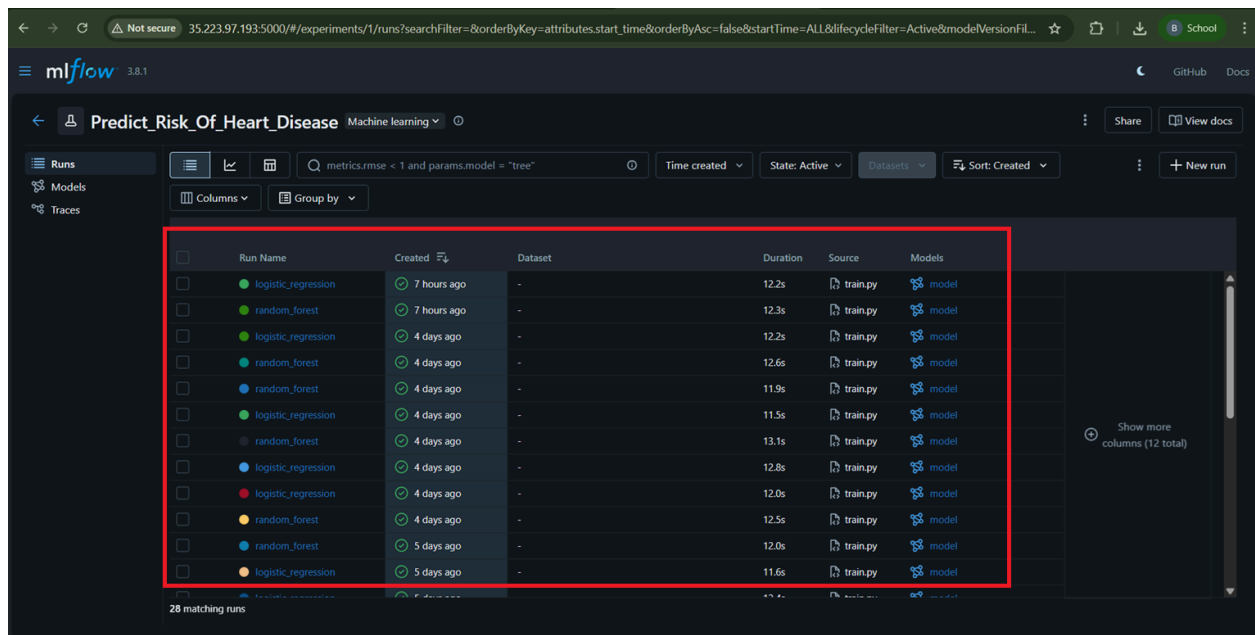
http://35.223.97.193:5000/#/experiments/1/runs?searchFilter=&orderByKey=attributes.start_time&orderByAsc=false&startTime=ALL&lifecycleFilter=Active&modelVersionFilter=All+Runs&datasetsFilter=W10%3D

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”



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



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

The screenshot displays the mlflow web interface for a specific run. The breadcrumb navigation shows 'Predict_Risk_Of_Heart_Disease > Runs > logistic_regression'. The 'logistic_regression' run is highlighted in the left sidebar. The main content area is divided into two panels. The left panel, titled 'Overview', contains a 'Description' section with 'No description', a 'Metrics (5)' table, and a 'Parameters (2)' table. The right panel, titled 'About this run', provides metadata about the run.

Metric	Value	Models
accuracy	0.8524590163934426	model
precision	0.8275862068965517	model
recall	0.8571428571428571	model
f1	0.8421052631578947	model
roc_auc	0.9523809523809524	model

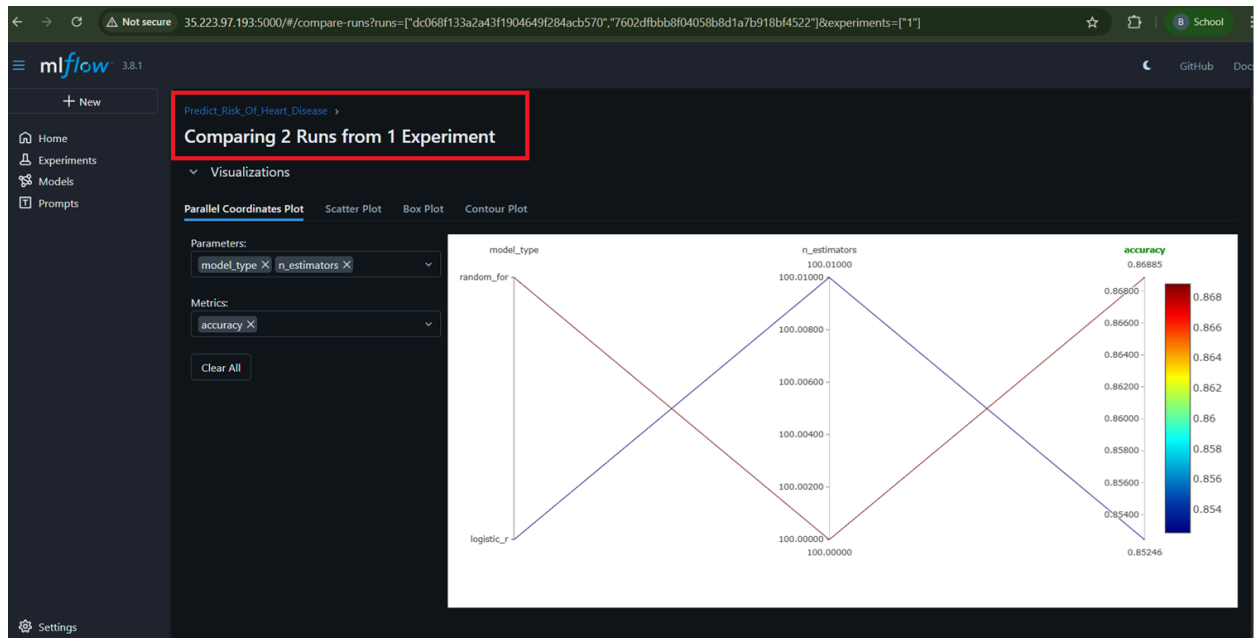
Parameter	Value
model type	logistic regression

Created at	01/04/2026, 02:25:11 PM
Created by	root
Experiment ID	1
Status	Finished
Run ID	dc068f133a2a43f1904649284acb570
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 'Runs' page for the 'Predict_Risk_Of_Heart_Disease' experiment. The 'Compare' button is highlighted in the top toolbar. Below the toolbar, a table lists the runs. The first two runs, 'logistic_regression' and 'random_forest', are selected with checkboxes and highlighted by a red box. The table columns include Run Name, Created, Dataset, Duration, Source, and Models. A 'Show more columns (12 total)' button is visible on the right side of the table.

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
random_forest	5 days ago	-	11.9s	train.py	model
logistic_regression	5 days ago	-	11.5s	train.py	model
random_forest	5 days ago	-	13.1s	train.py	model
logistic_regression	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
random_forest	6 days ago	-	12.0s	train.py	model
logistic_regression	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



mlflow 3.8.1

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

Home Experiments Models Prompts Settings

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.

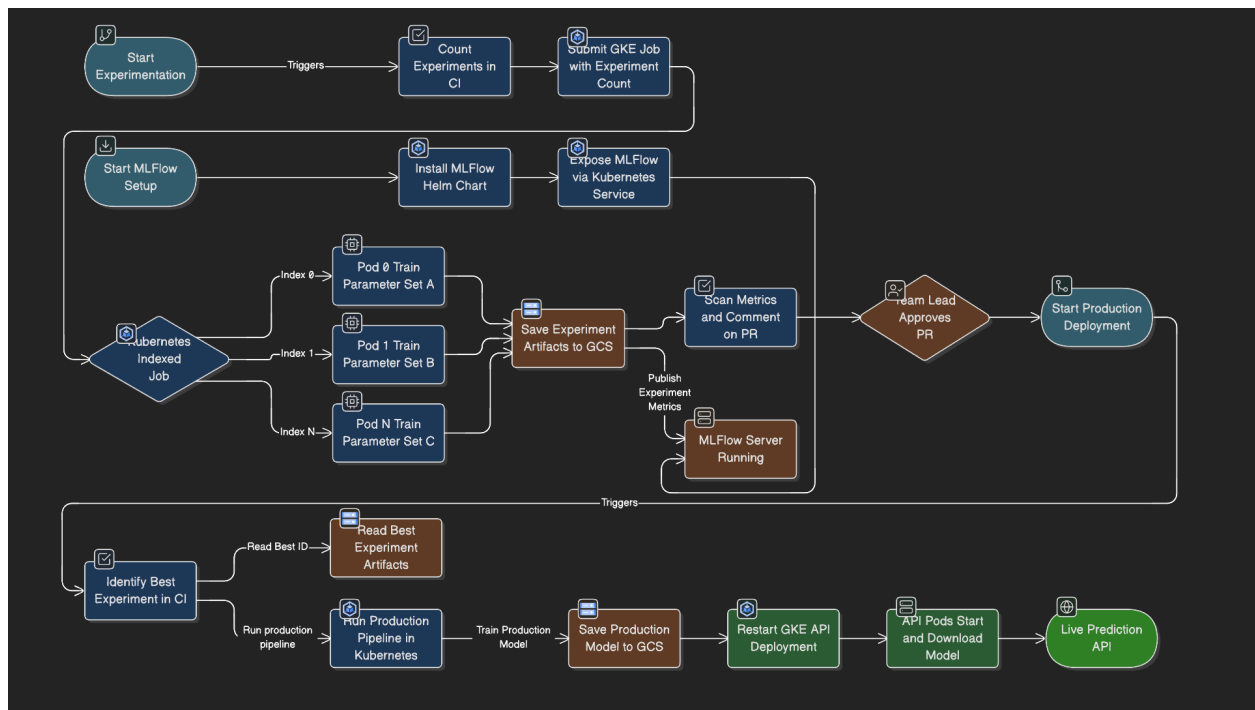
Tags

Show diff only

model	logistic_regression	random_forest
-------	---------------------	---------------

Architecture diagram

The architecture diagram is created using Mermaid diagraming 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

The screenshot shows a GitHub pull request interface. At the top, a purple 'Merged' badge is visible. Below it, a commit message reads: 'updating git pipeline to add eda and test cases #21' by 'nayan14', merged into 'main' from 'test_experiment' 16 minutes ago. A comment from 'github-actions' (bot) from 5 days ago is shown, titled 'Model Training Results (GKE Jobs)'. This comment contains a table with the following data:

Model Type	Accuracy	Run ID
random_forest	0.8689	20251230-173839
logistic_regression	0.8525	20251230-173839

Below the comment, a message states: 'Repository owner deleted a comment from github-actions (bot) 5 days ago'. Further down, a merge commit by 'nayan14' is shown, merging commit '1086632' into 'main' 16 minutes ago. A 'Hide details' button is next to this commit. Below the merge commit, a section titled '6 checks passed' lists the following checks, each with a green checkmark and a 'Details' link:

- changes
- unit-tests-and-lint
- build-trainer
- test-api-dry-run
- run-pipeline
- deploy-api

All stages and tasks defined as part of github actions

This screenshot shows the summary page of a GitHub Actions workflow named 'mlops-pipeline.yml'. The workflow was triggered by a push to the 'main' branch by user 'nayan14' and has a status of 'Success' with a total duration of '7m 10s'. The left sidebar lists the jobs: 'changes', 'unit-tests-and-lint', 'build-trainer', 'test-api-dry-run', 'run-pipeline', and 'deploy-api', all of which are marked as successful. The main area displays a visual representation of the workflow graph, showing the sequence of jobs and their durations: 'changes' (4s), 'unit-tests-and-lint' (1m 1s), 'build-trainer' (1m 38s), 'run-pipeline' (2m 5s), and 'deploy-api' (2m 2s). A 'test-api-dry-run' job is also shown, branching off from 'unit-tests-and-lint' and lasting 1m 26s.

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

This screenshot shows the detailed summary for the 'changes' job within the 'mlops-pipeline.yml' workflow. The job is part of a pull request merge for 'rtgarg1991/test_experiment #114' and has a status of 'Success'. The job summary indicates it 'succeeded 7 minutes ago in 4s'. The left sidebar shows the job 'changes' as the selected item. The main area lists the steps of the job, each with a success icon and a duration: 'Set up job' (1s), 'Run actions/checkout@v4' (1s), 'Run dorny/paths-filter@v3' (0s), 'Post Run actions/checkout@v4' (0s), and 'Complete job' (0s). A search bar for logs is visible at the top right of the job details.

Unit test cases and Eslint is executed

The screenshot shows the GitHub Actions interface for a workflow named 'MLOps Pipeline'. The current job is 'unit-tests-and-lint', which has succeeded 7 minutes ago in 1m 1s. The left sidebar shows a list of jobs: changes, unit-tests-and-lint (selected), build-trainer, test-api-dry-run, run-pipeline, and deploy-api. The main panel displays the steps of the 'unit-tests-and-lint' job:

Step	Duration
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
Complete job	0s

Docker images are built for experimentation

The screenshot shows the GitHub Actions interface for a workflow named 'MLOps Pipeline'. The current job is 'build-trainer', which has succeeded 5 minutes ago in 1m 38s. The left sidebar shows a list of jobs: changes, unit-tests-and-lint, build-trainer (selected), test-api-dry-run, run-pipeline, and deploy-api. The main panel displays the steps of the 'build-trainer' job:

Step	Duration
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
Complete job	0s

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

The screenshot shows the GitHub Actions interface for a workflow named 'run-pipeline'. The workflow is triggered by a merge pull request #21 from 'rtgarg1991/test_experiment #114'. The workflow has succeeded 3 minutes ago in 2m 8s. The left sidebar shows a list of jobs: changes, unit-tests-and-lint, build-trainer, test-api-dry-run, run-pipeline (selected), and deploy-api. The main panel displays the steps of the 'run-pipeline' job, which include: 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), and Post Run google-github-actions/auth@v2 (0s).

Step	Duration
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 the GitHub Actions interface for a workflow named 'deploy-api'. The workflow is triggered by a merge pull request #21 from 'rtgarg1991/test_experiment #114'. The workflow has succeeded 2 minutes ago in 2m 3s. The left sidebar shows a list of jobs: changes, unit-tests-and-lint, build-trainer, test-api-dry-run, run-pipeline, and deploy-api (selected). The main panel displays the steps of the 'deploy-api' job, which include: 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), and Complete Job (0s).

Step	Duration
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

https://github.com/rtgarg1991/ML_Ops_Assignment/actions/runs/20690395861/job/59397786190

Experimentation results are shown on PR before merging

The screenshot shows a GitHub Pull Request interface. At the top, a commit titled 'tests' is shown with a green checkmark and the commit hash '9f925a3'. Below this, a comment from the 'github-actions' bot is displayed, titled 'Model Training Results (GKE Jobs)'. This comment contains a table with the following data:

Model Type	Accuracy	Run ID
random_forest	0.8689	20251230-173839
logistic_regression	0.8525	20251230-173839

Below the comment, a message indicates that the repository owner deleted a comment from 'github-actions' bot 5 days ago. Further down, a merge commit by 'nayan14' is shown, merging commit '1086632' into 'main' 25 minutes ago. At the bottom, a section titled '6 checks passed' lists the following checks, each with a green checkmark and a 'Details' link:

- changes
- unit-tests-and-lint
- build-trainer
- test-api-dry-run
- run-pipeline

GKS cluster

The screenshot shows the Google Cloud Kubernetes Engine Clusters dashboard. The top navigation bar includes the Google Cloud logo, a search bar, and various icons. The left sidebar contains a navigation menu with options like 'All Fleets', 'Resource Management', 'Workloads', 'AI/ML', 'Teams', 'Applications', 'Secrets & ConfigMaps', 'Storage', 'Object Browser', 'Upgrades', 'Backup for GKE', 'Posture Management', 'Security', 'Marketplace', and 'Release Notes'. The main content area is titled 'Kubernetes clusters' and includes buttons for 'Create', 'Deploy', 'Refresh', 'Attach cluster', and 'New'. Below this, there are tabs for 'Overview', 'Utilization', 'Observability', and 'Cost Optimization'. The 'Overview' tab is selected, showing a summary of the cluster's health and upgrade status. The 'Health' section indicates '100% healthy' with a green progress bar. The 'Upgrade' section indicates '100% up to date' with a green progress bar. The 'Estimated monthly cost' section shows '₹0.00 / month · 0%' with 'No recommendations'. Below this, a table lists the clusters with columns for 'Status', 'Name', 'Location', 'Number of nodes', 'Total vCPUs', 'Total memory', 'Notifications', and 'Labels'. The table contains one entry for the 'mlops-cluster' in the 'us-central1-a' location, with 1 node and 2 vCPUs. The 'Notifications' column for this cluster includes links for 'Create a backup plan' and 'Set maintenance window'.

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	mlops-cluster	us-central1-a	1	2	8 GB	Create a backup plan Set maintenance window	—

Cluster Nodes

The screenshot shows the Google Cloud console interface for the 'mlops-final' project. The main navigation menu on the left includes 'All Fleets', 'Resource Management', 'Workloads', 'AI/ML', 'Teams', 'Applications', 'Secrets & ConfigMaps', 'Storage', 'Object Browser', 'Upgrades', 'Backup for GKE', 'Posture Management', 'Security', 'Marketplace', and 'Release Notes'. The 'Clusters' link under 'Resource Management' is selected.

The main content area displays the 'Cluster details' page for the 'mlops-cluster'. The 'Nodes' tab is active, showing a table of node pools and a table of individual nodes.

Node pools table:

Name	Status	Version	Number of nodes	Machine type	Image type	Autoscaling	Default IPv4 Pod IP address range
default-pool	Ok	1.33.5-gke.1308000	1	e2-standard-2	Container-Optimized OS with containerd (cos_containerd)	Off	10.108.0.0/14

Nodes table:

Name	Status	Node type	CPU requested	CPU allocatable	Memory requested	Memory allocatable
gke-mlops-cluster-default-pool-64a92117-b835	Ready	User-managed	926 mCPU	1.93 CPU	1.47 GB	6.32 GB

The URL at the bottom of the page is: <https://console.cloud.google.com/kubernetes/clusters/details/us-central1-a/mlops-cluster/nodes?project=mlops-final-001>

Model Artifacts stored through CI/CD pipeline

The screenshot shows the Google Cloud console interface for the 'mlops-final' project. The main navigation menu on the left includes 'Cloud Storage', 'Overview', 'Monitoring', 'Settings', 'Storage Intelligence', 'Insights datasets', 'Configuration', 'Marketplace', and 'Release Notes'. The 'Cloud Storage' link is selected.

The main content area displays the 'Buckets' page. A table lists the buckets in the project.

Name	Created	Location type	Location	Default storage class	Last modified	Public
mlops-final-001-mlops-artifacts	Dec 25, 2025, 6:50:10 PM	Region	us-central1	Standard	Dec 25, 2025, 6:50:10 PM	Subject

Different buckets to store relevant data

The screenshot shows the Google Cloud Storage interface for a bucket named `mlops-final-001-mlops-artifacts`. The bucket is located in `us-central1 (Iowa)` with a `Standard` storage class. It is subject to object ACLs and has `Soft Delete` protection enabled. The `Objects` tab is selected, showing a folder browser on the left and a table of objects on the right.

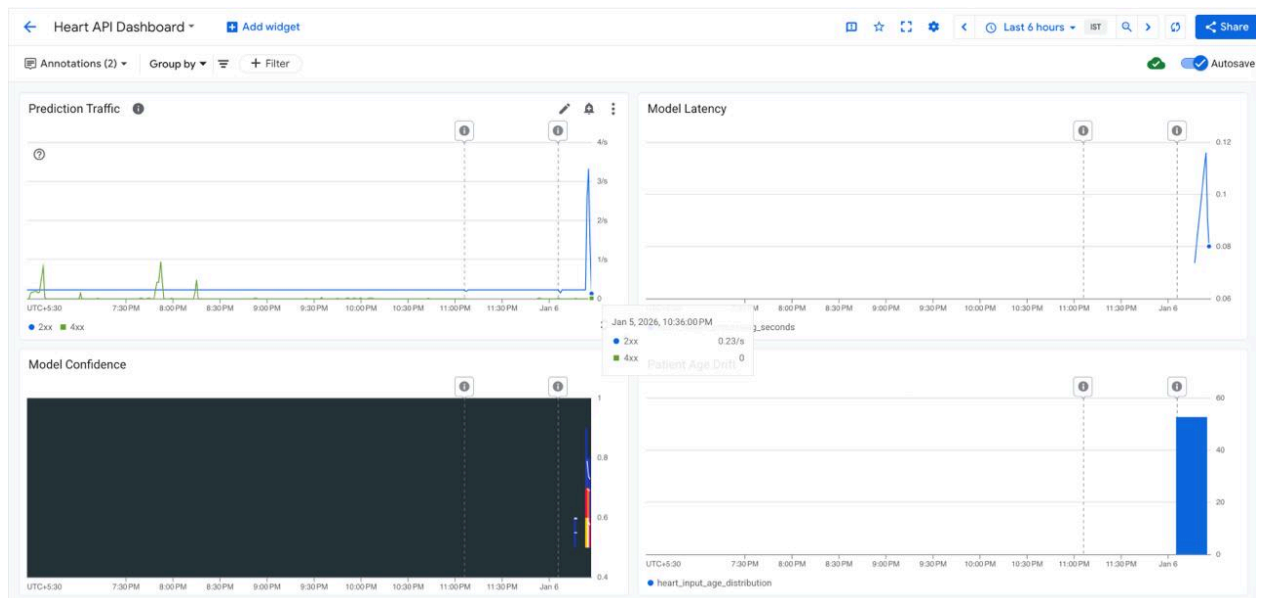
Folder browser:

- `mlops-final-001-mlops-artifacts`
 - `data/`
 - `experiments/`
 - `latest/`
 - `mlflow/`
 - `production/`

Objects table:

Name	Size	Type	Created	Storage class	Last modified
<code>data/</code>	—	Folder	—	—	—
<code>experiments/</code>	—	Folder	—	—	—
<code>latest/</code>	—	Folder	—	—	—
<code>mlflow/</code>	—	Folder	—	—	—
<code>production/</code>	—	Folder	—	—	—

Observability





[Link to code repository](#)

https://github.com/rtgarg1991/MLOps_Assignment