

Lab 4: Automated Model Deployment Using GitHub Actions (CI/CD)

Objective

This lab automates the **entire model deployment workflow** introduced in Lab 3 using **GitHub Actions**. Students will configure a CI/CD pipeline that automatically:

1. Retrieves the best trained model artifact
2. Builds a containerized FastAPI inference service
3. Pushes the container image to **Docker Hub**
4. Produces a deployable image that can be pulled for inference

This lab demonstrates **Continuous Deployment (CD)** for machine learning inference.

Prerequisites

- Completion of **Lab 1, Lab 2, and Lab 3**
- GitHub account
- Docker Hub account
- Familiarity with:
 - GitHub Actions
 - Docker images and tags
 - FastAPI-based inference services

Context: What Changes from Lab 3

Aspect	Lab 3	Lab 4
Model selection	Manual	Automated via CI artifacts
Docker build	Manual	Automated
Image push	Manual	Automated
Deployment trigger	Local command	GitHub push
Traceability	Ad hoc	CI run + image tags

Task 1: Secrets and Configuration

Students must configure the following **GitHub repository secrets**:

- Docker Hub username
- Docker Hub access token (Account Settings-> Settings-> Personal acces tokens->Generate new token->Access permission(Read,Write,Delete)->Generate)
- GitHub Personal Access Token (Account Settings-> Developer Settings-> Personal acces tokens->fine grade tokens->generate new token-> only selected

repositories(select your repo)->Add permissions (Actions, Variables, Metadata)->Generate Token)

- Add Variables to track the best R2 score and MSE

Task 2: GitHub Actions Deployment Workflow

Students must create a **train and deployment workflow**

Job 1: train (Continuous Integration)

Goal: Verify the code works and train a new model.

1. Environment: Sets up Python 3.11 on an Ubuntu server.
2. Execution: Runs scripts/train.py.
 - The script trains a model and saves metrics.json (e.g., {"accuracy": 0.98}).
3. Capture metrics:
 - Extracts the metrics value and saves it for later use.
4. Upload: Saves the model and metrics files as a GitHub Artifact.

Job 2: deploy (Continuous Deployment)

Goal: If the model is good, release it.

1. Check Success: usage of needs: train ensures this only runs if training passed.
2. Download Artifacts: Gets the model trained in the previous job.
3. Compare both metrics:
 - Checks if Current Metric > BEST_Metric (the variable we set up earlier).
 - If No: It prints "<Roll_no>----Metric did not improve" and stops.
 - If Yes: It proceeds to build the Docker image.
4. Build and Publish Image:
 - Login to Docker Hb using username and token defined in github secrets
 - Builds the image using the Dockerfile.
 - Pushes it to DockerHub so it can be pulled by servers.
5. Update Baseline:
 - Updates the Best Metrics in github variable to the new high score, so future models have a harder target to beat.

Task 3: Post-Deployment Validation

After the CI pipeline completes, students must:

1. Pull the published image from Docker Hub
2. Run the container locally
3. Perform inference using the API
4. Verify:
 - Prediction endpoint response in the following format

```
{ name: <your_name>, roll_no: <your_roll_no>,  
wine_quality: 5}
```

Deliverables

Students must submit:

1. GitHub repository link
2. DockerHub repository Link
3. Screenshot(s) showing:
 - Successful GitHub Actions deployment workflow
 1. Running both train and deploy jobs
 2. Running train job and skipping deploy job
 - Docker Hub image repository
 - Pulled image running locally
 - Successful inference response