**MLOps Assessment Document (Docker Swarm Deployment)**

**1. Introduction and Artefact Description**

This project delivers a complete MLOps pipeline using GitHub Actions for CI/CD and Docker Swarm for orchestrated deployment. The core artefact is a simple machine learning model deployed via a Flask API. The model predicts salary based on a user's age using a linear regression model trained on sample tabular data. The application is containerized using Docker and automatically deployed to a Swarm cluster as a scalable service with three replicas.

**Key components:**

* **Model**: Scikit-learn Linear Regression (Age ➔ Salary)
* **API**: Flask app handling POST requests with JSON input
* **CI/CD**: GitHub Actions workflows for testing, image building, and Swarm deployment
* **Deployment**: Docker Swarm with 3 replicas across nodes
* **Retraining**: Workflow-triggered model retraining and redeployment

**2. Branching Strategy**

A GitHub-based branching strategy was adopted to separate development, integration, and production workflows:

* main: Protected production branch for stable, deployable releases
* dev: Integration branch for combined features and testing
* feature/\*: Individual branches for model, API, and pipeline components

This strategy enforces pull requests and approvals before any merge to main, ensuring code quality and CI validation.

**3. Testing Strategy and Use-Cases**

Two main test cases validate the integrity of the pipeline:

* **Unit Test 1**: test\_api.py — checks Flask route for proper status code and response format
* **Unit Test 2**: test\_preprocess.py — verifies preprocessing logic correctly strips and standardizes input

These tests are run on each push to dev and main branches using GitHub Actions.

**4. CI/CD, CT, and CM Workflows**

**CI Workflow (ci.yaml)**

* Triggered on pushes to dev and PRs to main
* Steps:
  + Set up Python environment
  + Install dependencies
  + Run unit tests
  + Build Docker image
  + Push image to Docker Hub

**CD Workflow (deploy.yaml)**

* Triggered on successful CI or manually via workflow\_dispatch
* Steps:
  + SSH into Docker Swarm manager node
  + Pull latest Docker image
  + Deploy/Update service with 3 replicas using docker stack deploy

**CT Workflow (ct.yaml)**

* Triggered manually or on a schedule
* Checks for model drift by comparing statistics from new input data (e.g., mean age)
* Alerts via GitHub Actions log or messaging system if drift exceeds threshold
* Recommends retraining if drift detected

**CM Workflow (retrain.yaml)**

* Manually triggered retraining pipeline
* Steps:
  + Pull latest cleaned data
  + Retrain model and save .pkl
  + Commit updated model to repo or upload to Docker image
  + Rebuild and redeploy

**5. Deployment Architecture**

The architecture uses **Docker Swarm Mode**:

* 1 Manager Node
* 2+ Worker Nodes
* Flask API runs as a **service with 3 replicas**
* An **ingress load balancer** routes traffic across replicas

[Client] ---> [Ingress Load Balancer (Swarm)] ---> [Flask Container 1]

---> [Flask Container 2]

---> [Flask Container 3]

All nodes are VMs configured on a cloud provider (e.g., Google Cloud). The manager orchestrates deployments and maintains service state. The GitHub Action pushes the updated image and triggers the deployment via docker stack deploy.

**6. References**

* [Docker Swarm Stack Deploy](https://docs.docker.com/engine/swarm/stack-deploy/)
* [GitHub Actions](https://docs.github.com/en/actions)
* Flask Documentation
* Scikit-learn Linear Regression
* Week 7 & 8 Tutorials on GitHub Actions, Docker, and Swarm (Course Materials)

**Repo Link**: [[**l00188628\_Mlops\_Assesment**](https://github.com/l00188628/l00188628_Mlops_Assesment)