School of Computer Science and Engineering

MLOps with Cloud Computing Laboratory Manual

B.Tech – VI Semester
Course Duration: 30 Hours

1. Course Information

Course Name: MLOps with Cloud Computing Lab
Semester: VI
Program: B.Tech
Credits: As per scheme

2. Course Objectives

This laboratory course is designed to: - Provide hands-on experience with cloud-based development environments - Enable students to build, deploy, and manage ML models using MLOps practices - Introduce Infrastructure as Code (IaC), CI/CD, and container orchestration - Familiarize students with cloud AI services and AutoML workflows - Develop an end-to-end MLOps pipeline using industry-standard tools

3. Course Outcomes (COs)

After completing this laboratory course, students will be able to:

- **CO1:** Set up and use cloud-based development environments for ML workflows
- **CO2:** Develop and deploy ML microservices using cloud platforms
- **CO3:** Apply Infrastructure as Code and CI/CD principles for ML systems
- **CO4:** Perform model, data, and experiment versioning using MLOps tools
- **CO5:** Design and deploy scalable ML pipelines and AutoML solutions

4. Mapping of Experiments to Course Outcomes

Exp. No	Experiment Title	COs
1	Setup and Explore Cloud Developer Workspaces	CO1
2	Create and Deploy a Simple ML Microservice API	CO1, CO2
3	Implement Infrastructure as Code (IaC) using Terraform	CO3
4	Build a CI/CD Pipeline for ML Model Deployment	CO3
5	Automate Model and Data Versioning with DVC and MLflow	CO4
6	Deploy ML Model using AWS App Runner with Flask Application	CO2, CO5
7	Create AutoML Workflow Using Cloud	CO5
8	Consume Cloud AI APIs for NLP and Computer Vision	CO5
9	Build and Deploy Kubeflow Pipelines on Kubernetes	CO3, CO5
10	Capstone Mini-Project: End-to-End MLOps Pipeline	CO1–CO5

5. Software and Hardware Requirements

Hardware Requirements

- Laptop/Desktop with minimum 8 GB RAM
- Stable Internet connection

Software / Cloud Requirements

- Google Cloud Platform / AWS Account (Free Tier)
- Python 3.8 or above
- Git & GitHub
- Docker
- Terraform
- Kubernetes (Minikube / GKE / EKS)
- MLflow
- DVC
- Flask
- VS Code / Cloud Developer Workspace

6. General Instructions

- Students must complete each experiment within the allotted lab hours
- Maintain a proper lab record with code, outputs, and screenshots
- Follow academic integrity and avoid plagiarism
- Demonstrate each experiment during evaluation

Experiment 1

Setup and Explore Cloud Developer Workspaces

Aim

To set up and explore a cloud-based developer workspace for machine learning and MLOps development.

Learning Outcome

Configure and use a cloud-hosted IDE for ML projects.

Tools Required

- Google Cloud Developer Workspace / AWS Cloud9
- GitHub account
- Python 3.8+

Theory

Cloud developer workspaces provide browser-based development environments with pre-installed tools, eliminating local setup issues and enabling scalable ML development.

Procedure

1. Login to Google Cloud / AWS Console
2. Create a new Developer Workspace / Cloud9 environment
3. Open terminal and verify Python installation
4. Clone a GitHub repository
5. Execute a sample ML script

Sample Code (test_ml.py)

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = RandomForestClassifier()
model.fit(X_train, y_train)
print("Accuracy:", model.score(X_test, y_test))
```

Execution Command

```
python test_ml.py
```

Expected Output (Screenshot Description)

- Terminal showing accuracy value (e.g., Accuracy: 0.96)
- Cloud IDE interface visible

Result

Cloud developer workspace successfully configured and tested.

Viva Questions

1. What is a cloud IDE?
2. Advantages over local setup?

Experiment 2

Create and Deploy a Simple ML Microservice API

Aim

To develop and deploy an ML model as a REST API.

Tools Required

Python, Flask, scikit-learn

Sample Code (app.py)

```
from flask import Flask, request, jsonify
import pickle
from sklearn.linear_model import LinearRegression
import numpy as np

app = Flask(__name__)
model = LinearRegression()
X = np.array([[1],[2],[3],[4]])
y = np.array([2,4,6,8])
model.fit(X, y)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json['value']
    prediction = model.predict([[data]])
    return jsonify({'prediction': prediction[0]})

if __name__ == '__main__':
    app.run(debug=True)
```

Execution

```
python app.py
```

Expected Output (Screenshot Description)

- Browser/Postman showing JSON response: {“prediction”: 10}

Result

ML microservice API executed successfully.

Experiment 3

Implement Infrastructure as Code (IaC) using Terraform

Aim

To provision cloud infrastructure using Terraform.

Sample Code (main.tf)

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "example" {
  ami           = "ami-0abcdef"
  instance_type = "t2.micro"
}
```

Commands

```
terraform init
terraform apply
```

Expected Output (Screenshot Description)

- Terminal showing “Apply complete!”
- EC2 instance visible in AWS console

Result

Infrastructure created using IaC.

Experiment 4

Build a CI/CD Pipeline for ML Model Deployment

Aim

To automate ML deployment using CI/CD.

Sample Code (.github/workflows/ml.yml)

```yaml
name: ML Pipeline
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Install dependencies
        run: pip install scikit-learn
      - name: Run script
        run: python test_ml.py
```

Expected Output (Screenshot Description)

- GitHub Actions workflow showing green check

Result

CI/CD pipeline executed successfully.

Experiment 5

Automate Model and Data Versioning with DVC and MLflow

Sample Commands

```
dvc init
dvc add data.csv
git add data.csv.dvc
git commit -m "Track data"
```

MLflow Code Snippet

```
import mlflow
mlflow.start_run()
mlflow.log_metric("accuracy", 0.95)
mlflow.end_run()
```

Expected Output

- MLflow UI showing logged experiment

Experiment 6

Deploy ML Model using AWS App Runner with Flask Application

Sample Code

Reuse Experiment 2 Flask app

Expected Output

- Public URL displaying prediction results

Experiment 7

Create AutoML Workflow Using Cloud

Procedure

- Upload dataset
- Run AutoML job

Expected Output

- Dashboard showing best model accuracy

Experiment 8

Consume Cloud AI APIs for NLP and Computer Vision

Sample Code

```
from google.cloud import vision
client = vision.ImageAnnotatorClient()
```

Expected Output

- JSON response with detected labels

Experiment 9

Build and Deploy Kubeflow Pipelines on Kubernetes

Sample Code (pipeline.py)

```
import kfp
```

Expected Output

- Kubeflow UI showing pipeline graph

Experiment 10

Capstone Mini-Project: End-to-End MLOps Pipeline

Sample Architecture

- GitHub → CI/CD → Cloud → Monitoring

Expected Output

- End-to-end deployed ML system