DevOps and MLOps Lab Manual

Engineering Department

January 2026

Lab Questions

1. DevOps and MLOps Lab Manual
   Create a simple User Registration Form for an event with fields (Name, Email, Phone). Initialize a Git repository, commit the project files, and push them to GitHub using appropriate Git commands. Create a new branch named `update-form`, modify the registration form by adding a new field (Department), merge the branch into `main`, and push the changes to GitHub. Show the use of: `git branch`, `git checkout`, `git merge`, `git push`.

2. Your college department is developing a small student portal website hosted on GitHub. As a DevOps engineer, you must set up Continuous Integration using Jenkins:
   Task Requirements
   Configure a Jenkins Freestyle Job that automatically pulls the website code from GitHub.
   Enable Jenkins to build the project every time a change is pushed.
   Make a small update to the HTML page, push it to GitHub, and show that Jenkins automatically triggers a new build.

3. Create a simple web application (HTML or Python Flask). Write a Dockerfile, build a Docker image, and run the container so the application is accessible on a browser using port mapping. Using Docker commands, perform the following operations:
   List images and containers
   Stop a running container
   Remove a container and image
   Demonstrate commands like: `docker ps`, `docker stop`, `docker rm`, `docker rmi`.

4. Deploy the previously created Docker image on Kubernetes using a Deployment YAML file. Verify that the pod is running using `kubectl` commands. Expose the application using a NodePort Service, access it using the node's IP and port, and scale the deployment to 3 replicas using `kubectl scale`.

5. Set up a simple ML project environment by creating a `requirements.txt` file, installing packages, and verifying environment setup. Document the steps in a Jupyter Notebook and commit it to the Git repository.

6. To design and deploy a multi-container application using Docker Compose by creating an application service and a dependent database/Redis service, configuring service dependencies, networks, and volumes, and verifying inter-container communication.

7. To implement data ingestion, cleaning, and versioning using Data Version Control (DVC) by tracking raw and processed datasets, creating a reproducible pipeline with `dvc.yaml`, and validating reproducibility via `dvc repro`.

8. To perform experiment tracking using MLflow by training a machine learning model, logging metrics, parameters, and artifacts, and comparing multiple runs to identify the best-performing model.

9. To optimize and standardize model inference using ONNX by exporting a trained ML model to ONNX format, running inference with ONNX Runtime, and benchmarking performance against the native scikit-learn model.

10. To serve a machine learning model using FastAPI by developing a `/predict` REST endpoint with input validation, writing test cases, containerizing the service, and verifying predictions through Postman or `curl`.

Prerequisites

Ensure the following tools are installed and properly configured on your Linux system (Ubuntu-based). Commands below assume `sudo` privileges.

Git Installation and Verification

Download & Install:

```
$ sudo apt update
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease [128 kB]
...
Reading package lists... Done
$ sudo apt install -y git
Reading package lists... Done
Building dependency tree... Done
The following NEW packages will be installed:
  git git-man ...
After this operation, 45.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
...
Setting up git (1:2.43.0-1ubuntu...) ...
```

Verify Version:

```
$ git --version
git version 2.43.0
```

Docker & Docker Compose

Download & Install Docker Engine:

```
$ sudo apt update
$ sudo apt install -y docker.io
Reading package lists... Done
...
Setting up docker.io (24.0.x-0ubuntu...) ...
Adding group 'docker' (GID 998) ...
```

Enable & Start Docker Service:

```
$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script...
$ sudo systemctl start docker
$ sudo systemctl status docker
    docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled)
     Active: active (running) since Tue 2026-01-06 10:15:00 IST; 5s ago
...
```

Install Docker Compose Plugin (if needed):

```
$ sudo apt install -y docker-compose
Reading package lists... Done
...
Setting up docker-compose (1.29.x-...) ...
```

Verify Docker:

```
$ docker --version
Docker version 24.0.x, build abcdefg
$ docker-compose --version
docker-compose version 1.29.x, build hijklmno
```

Jenkins

Download & Install Jenkins (Debian package method summarized):

```
$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key \
  | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | \
  sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
$ sudo apt update
$ sudo apt install -y openjdk-17-jre-headless jenkins
Reading package lists... Done
...
Setting up jenkins (2.xxx) ...
```

Start Jenkins and Verify:

```
$ sudo systemctl enable jenkins
$ sudo systemctl start jenkins
$ sudo systemctl status jenkins
    jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled)
     Active: active (running) since Tue 2026-01-06 10:20:00 IST; 10s ago
...
```

Access Jenkins in a browser at:

```
http://localhost:8080
```

Kubernetes (Minikube Example)

Download Minikube Binary:

```
$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  74.4M  100  74.4M    0     0  8.5M      0  0:00:08  0:00:08 --:--:--  9.0M
$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Start a Local Cluster:

```
$ minikube start
* minikube v1.34.0 on Ubuntu 24.04
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
...
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace
```

Verify kubectl (Minikube bundles it or install via apt):

```
$ kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   5m1s   v1.29.x
```

Python, pip, venv, and Jupyter

Install Python 3 and pip:

```
$ sudo apt update
$ sudo apt install -y python3 python3-pip python3-venv
Reading package lists... Done
...
Setting up python3 (3.12.x-...) ...
```

Create Virtual Environment:

```
$ python3 -m venv ml-env
$ source ml-env/bin/activate
(ml-env) $ python --version
Python 3.12.x
```

Install Jupyter and ML Packages:

```
(ml-env) $ python -m pip install --upgrade pip
(ml-env) $ pip install jupyter pandas scikit-learn mlflow fastapi uvicorn dvc
    onnxruntime
Collecting jupyter
Collecting pandas
Collecting scikit-learn
...
Successfully installed jupyter-1.x pandas-2.x scikit-learn-1.x mlflow-2.x fastapi-0.11x
     ...
```

Launch Jupyter Notebook:

```
(ml-env) $ jupyter notebook
[I 10:30:00.000 NotebookApp] Serving notebooks from local directory: /home/user/project
[I 10:30:00.001 NotebookApp] The Jupyter Notebook is running at:
[I 10:30:00.001 NotebookApp] http://localhost:8888/?token=...
```

---

1 Git Initialization & GitHub Push

Objective: Create a form and push it to a remote repository.

File: index.html

```
<!DOCTYPE html>
<html>
<head><title>Event Registration</title></head>
<body>
    <h2>Registration Form</h2>
    <form>
        Name: <input type="text"><br>
        Email: <input type="email"><br>
        Phone: <input type="text"><br>
        <button>Submit</button>
    </form>
</body>
</html>
```

Terminal Commands & Output

```
$ git init
Initialized empty Git repository in /home/user/lab1/.git/
$ git add index.html
$ git commit -m "Initial commit"
[main (root-commit) 4e2a1b] Initial commit
 1 file changed, 11 insertions(+)
 create mode 100644 index.html
$ git remote add origin https://github.com/username/event-form.git
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 245 bytes | 245.00 KiB/s, done.
To https://github.com/username/event-form.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

2 Git Branching, Merging & Push

Objective: Add a "Department" field using a feature branch.

Terminal Commands & Output

```
$ git branch update-form
$ git checkout update-form
Switched to branch 'update-form'
# (Modify index.html to add <input type="text" name="Dept">)
$ git add index.html
$ git commit -m "Added department field"
[update-form 7a2b3c] Added department field
 1 file changed, 1 insertion(+)
$ git checkout main
Switched to branch 'main'
$ git merge update-form
Updating 4e2a1b..7a2b3c
Fast-forward
 index.html | 1 +
 1 file changed, 1 insertion(+)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 280 bytes | 280.00 KiB/s, done.
To https://github.com/username/event-form.git
   4e2a1b..7a2b3c  main -> main
```

3 Jenkins CI Integration

Objective: Automatically trigger builds on push.

Terminal Commands & Output

```
$ echo " " >> index.html
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  modified:   index.html

$ git commit -am "Trigger Jenkins"
[main 9c1d2e3] Trigger Jenkins
 1 file changed, 1 insertion(+)
$ git push origin main
To https://github.com/username/event-form.git
   7a2b3c..9c1d2e3  main -> main
```

Jenkins Console Log

```
Started by GitHub push by username
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/StudentPortal
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git fetch --tags --progress -- origin +refs/heads/*:refs/remotes/origin/*
 > git checkout -f origin/main
 + echo "Starting Build..."
Starting Build...
+ ls
index.html  Jenkinsfile  ...
Finished: SUCCESS
```

4 Dockerization of Web App

Objective: Containerize a Python Flask application.

File: app.py

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Portal Running!"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

File: Dockerfile

```
FROM python:3.9-slim
RUN pip install flask
COPY app.py .
CMD ["python", "app.py"]
```

Terminal Commands & Output

```
$ docker build -t my-app .
Sending build context to Docker daemon  6.144kB
Step 1/4 : FROM python:3.9-slim
 ---> a1b2c3d4e5f6
Step 2/4 : RUN pip install flask
 ---> Running in 123456789abc
Collecting flask
...
Successfully built 0f1e2d3c4b5a
Successfully tagged my-app:latest

$ docker run -d -p 8080:5000 --name web-container my-app
1a2b3c4d5e6f7g8h9i0j

$ docker ps
CONTAINER ID   IMAGE   COMMAND           STATUS        PORTS
1a2b3c4d5e6f   my-app  "python app.py"   Up 10 seconds 0.0.0.0:8080->5000/tcp

$ curl http://localhost:8080
Portal Running!
```

5 Docker Operations

Objective: Demonstrate image and container management.

Terminal Commands & Output

```
$ docker ps
CONTAINER ID   IMAGE   STATUS       PORTS                     NAMES
1a2b3c4d5e6f   my-app  Up 2 minutes 0.0.0.0:8080->5000/tcp    web-container

$ docker stop web-container
web-container

$ docker rm web-container
```

```
web-container

$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
my-app        latest    0f1e2d3c4b5a   1 minute ago     125MB

$ docker rmi my-app
Untagged: my-app:latest
Deleted: sha256:0f1e2d3c4b5a...
```

6 Kubernetes Deployment & Scaling

Objective: Deploy to Kubernetes and scale to 3 replicas.

File: deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ml
  template:
    metadata:
      labels:
        app: ml
    spec:
      containers:
      - name: ml-container
        image: my-app
        ports:
        - containerPort: 5000
```

Terminal Commands & Output

```
$ kubectl apply -f deployment.yaml
deployment.apps/ml-deploy created

$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
ml-deploy-7f8d9-abc1         1/1     Running   0          20s

$ kubectl expose deployment ml-deploy --type=NodePort --port=5000
service/ml-deploy exposed

$ kubectl get svc ml-deploy
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
ml-deploy   NodePort   10.96.123.45   <none>        5000:31456/TCP 30s

# Access via http://<node-ip>:31456

$ kubectl scale deployment ml-deploy --replicas=3
deployment.apps/ml-deploy scaled

$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
ml-deploy-7f8d9-abc1         1/1     Running   0          1m
ml-deploy-7f8d9-abc2         1/1     Running   0          10s
ml-deploy-7f8d9-abc3         1/1     Running   0          10s
```

7 ML Environment Setup

Objective: Set up environment and document via Jupyter.

File: requirements.txt

```
pandas
scikit-learn
mlflow
jupyter
```

Terminal Commands & Output

```
(ml-env) $ pip install -r requirements.txt
Collecting pandas
Collecting scikit-learn
Collecting mlflow
Collecting jupyter
...
Successfully installed jupyter-1.x mlflow-2.x pandas-2.x scikit-learn-1.x

(ml-env) $ jupyter notebook
[I 10:30:00.000 NotebookApp] Serving notebooks from local directory: /home/user/project
[I 10:30:00.001 NotebookApp] The Jupyter Notebook is running at:
[I 10:30:00.001 NotebookApp] http://localhost:8888/?token=...
```

8 Docker Compose (Multi-Container)

Objective: Launch web and Redis services.

File: docker-compose.yml

```
services:
  app:
    build: .
    ports: ["5000:5000"]
  redis:
    image: "redis:alpine"
```

Terminal Commands & Output

```
$ docker-compose up -d
Creating network "lab_default" with the default driver
Creating lab_redis_1 ... done
Creating lab_app_1   ... done

$ docker-compose ps
NAME         COMMAND               STATUS   PORTS
lab_app_1    "python app.py"       running  0.0.0.0:5000->5000/tcp
lab_redis_1  "docker-entrypoint "  running  6379/tcp
```

9 DVC & MLflow Tracking

Objective: Track data changes and log experiments.

Terminal Commands & Output

```
(ml-env) $ dvc init
Initialized DVC repository.

(ml-env) $ dvc add raw_data.csv
Adding...
100% Add|

    |1/1 [00:00,  2.00file/s]
To track the changes with git, run:

    git add raw_data.csv.dvc .gitignore

(ml-env) $ git add raw_data.csv.dvc .gitignore
(ml-env) $ git commit -m "Track data v1"
[main 1a2b3c4] Track data v1
 2 files changed, 5 insertions(+)
 create mode 100644 raw_data.csv.dvc

(ml-env) $ dvc repro
Data is up to date.

(ml-env) $ python train.py
Model trained. Accuracy: 0.92
Logging to MLflow...

(ml-env) $ mlflow ui
INFO [mlflow.server] Starting gunicorn 20.x
INFO [mlflow.server] Listening at: http://127.0.0.1:5000 (pid=1234)
```

10 FastAPI Serving & ONNX Optimization

Objective: REST API for inference and model optimization.

Terminal Commands & Output

```
(ml-env) $ uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['/home/user/project']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)

$ curl -X POST "http://127.0.0.1:8000/predict" \
  -H "Content-Type: application/json" \
  -d '{"feature1":5, "feature2":10}'
{"prediction": 15}

(ml-env) $ python convert_onnx.py
Converting model to ONNX...
Model converted successfully. Saved as 'model.onnx'.

(ml-env) $ ls -lh model.onnx
-rw-r--r-- 1 user user 45K Jan  6 11:00 model.onnx
```