DEVOPS ENGINEERING – PRACTICAL ASSIGNMENT

Assignment Title

End-to-End DevOps CI/CD Pipeline Implementation for a Web Application

---

Objective (READ THIS CAREFULLY)

The objective of this assignment is to **build a complete DevOps pipeline** that automatically:

- Builds an application
- Packages it into a container
- Deploys it to a cloud environment
- Manages it using Kubernetes
- Monitors its health

This assignment is designed for **beginners with no real project experience** but who want **job-relevant DevOps exposure**.

Rules (NON-NEGOTIABLE)

1. **Every task must include screenshots**
2. Screenshots without explanation = ZERO value
3. Copy-paste without understanding = project failure
4. Each step must work before moving to the next
5. Final submission must be a **GitHub repository + PDF report**

Tools You MUST Use

- Git & GitHub
- Python Flask (or Node.js – choose ONE)
- Docker
- Docker Hub
- Jenkins
- AWS (EC2)

- Terraform
- Kubernetes (Minikube or EKS)
- Prometheus & Grafana

Skipping any tool = assignment incomplete.

□ TASK BREAKDOWN (STEP-BY-STEP)

TASK 1: Application Creation

Goal

Create a simple web application that can run locally.

Instructions

- Create a simple backend application
- App must have:
  - `/` endpoint → returns "Hello DevOps"
  - `/health` endpoint → returns "OK"
- Run the app locally

Mandatory Screenshots

- Application source code
- App running in browser
- Terminal showing app start command

Explanation Required

- What does this application do?
- Why is a health endpoint important in DevOps?

TASK 2: Version Control Using GitHub

Goal

Store source code in a remote repository and enable automation trigger.

Instructions

- Initialize Git repository
- Create GitHub repository
- Push source code to GitHub

Mandatory Screenshots

- GitHub repository page
- Commit history
- Terminal showing `git push`

Explanation Required

- Why is GitHub required in a CI/CD pipeline?
- What happens when code is pushed?

TASK 3: Dockerization of Application

Goal

Package the application into a container.

Instructions

- Create a `Dockerfile`
- Build Docker image
- Run container locally

Mandatory Screenshots

- Dockerfile
- Docker image build command
- Container running output
- App accessible through container

Explanation Required

- What problem does Docker solve?
- Difference between image and container

TASK 4: Push Image to Docker Hub

Goal

Store Docker image in a remote container registry.

Instructions

- Create Docker Hub account
- Tag Docker image
- Push image to Docker Hub

Mandatory Screenshots

- Docker Hub repository
- Image tags
- Terminal showing `docker push`

Explanation Required

- Why can’t Kubernetes use images from local machine?
- What is a container registry?

TASK 5: Jenkins CI Pipeline

Goal

Automate build and image creation.

Instructions

- Install Jenkins
- Create Jenkins pipeline using `Jenkinsfile`
- Pipeline must:
    - Pull code from GitHub
    - Build Docker image
    - Push image to Docker Hub

Mandatory Screenshots

- Jenkins dashboard
- Jenkinsfile

- Successful pipeline run
- Console output

Explanation Required

- What is CI?
- Why Jenkins is used instead of manual commands?

TASK 6: Cloud Setup on AWS

Goal

Run DevOps tools on real cloud infrastructure.

Instructions

- Create AWS EC2 instance
- Configure security groups
- Install Docker & Jenkins on EC2

Mandatory Screenshots

- AWS EC2 dashboard
- Instance running status
- SSH login terminal

Explanation Required

- Why cloud deployment is important?
- Difference between local and cloud environments

TASK 7: Infrastructure as Code Using Terraform

Goal

Automate infrastructure creation.

Instructions

- Write Terraform code to create:

    - EC2 instance
    - Security group
- Run `terraform apply`

Mandatory Screenshots

- Terraform files
- `terraform apply` output
- AWS resources created by Terraform

Explanation Required

- Why Infrastructure as Code is needed?
- What problem Terraform solves?

TASK 8: Kubernetes Deployment

Goal

Deploy containerized application using Kubernetes.

Instructions

- Install Minikube or configure EKS
- Create:
    - Deployment YAML
    - Service YAML
- Deploy application

Mandatory Screenshots

- Kubernetes YAML files
- Pods running
- Service exposed
- App accessible via Kubernetes

Explanation Required

- Difference between Docker and Kubernetes
- What is a Pod and Deployment?

TASK 9: Continuous Deployment (CD)

Goal

Automate deployment after CI.

Instructions

- Update Jenkins pipeline to deploy to Kubernetes
- New code push should update live application

Mandatory Screenshots

- Updated Jenkinsfile
- Pipeline with deployment stage
- Kubernetes rollout output

Explanation Required

- Difference between CI and CD
- How automation improves reliability

TASK 10: Monitoring Using Prometheus & Grafana

Goal

Monitor application and system health.

Instructions

- Install Prometheus and Grafana
- Configure monitoring for Kubernetes
- Create dashboard

Mandatory Screenshots

- Prometheus targets
- Grafana dashboard
- CPU & memory graphs

Explanation Required

- Why monitoring is critical in production
- What happens if monitoring is missing?

🗁 FINAL SUBMISSION REQUIREMENTS

GitHub Repository Structure

```
devops-ci-cd-project/
├── app/
├── docker/
├── jenkins/
├── terraform/
├── kubernetes/
├── monitoring/
└── README.md
```

Report (PDF or DOCX)

Must include:

- Architecture diagram
- Step-by-step explanation
- Screenshots for every task
- Problems faced & solutions
- What you learned