# DevOps Graduation Project Plan: High-Availability Shopping Website

This document outlines the project plan for a simple shopping website developed as a DevOps graduation project.   
The project demonstrates the integration of modern DevOps practices, including containerization, CI/CD pipelines,   
infrastructure automation, and high availability deployment on AWS.

## 1. Project Overview

The goal of this project is to design and deploy a simple e-commerce web application that allows users to browse products,   
purchase items, and manage their profiles. The system will consist of a frontend, backend microservices, a MySQL database,   
and a web server (Nginx) configured as a reverse proxy. The solution will be containerized using Docker, orchestrated   
using Kubernetes, and deployed on AWS infrastructure with GitHub Actions for continuous integration and delivery.

## 2. Team Structure

Team Members: 5–6 trainees  
- Frontend Developers (1): Build user interface using React.js or Node.js frontend.  
- Backend Developers (2): Implement RESTful APIs using FastAPI (Python) with MySQL integration.  
- DevOps Engineers (2): Handle CI/CD pipelines, containerization, deployment, and monitoring.  
- Project Manager (1): Coordinate progress, ensure timely delivery, and maintain documentation.

## 3. System Architecture

Components:  
- Frontend: Built with React.js and served via Nginx.  
- Backend: Microservices developed using FastAPI (Auth, Products, Orders).  
- Database: MySQL (hosted on AWS RDS or local Docker container).  
- Storage: AWS S3 for static content and images.  
- Web Server: Nginx as reverse proxy and load balancer.  
- CI/CD: GitHub Actions for automated build, test, and deploy pipelines.  
- Deployment: AWS EC2 instances or Kubernetes cluster (EKS/Minikube for demo).

## 4. Project Timeline (1 Month)

|  |  |
| --- | --- |
| Week | Tasks |
| Week 1 | Setup environment, initialize GitHub repo, define architecture, assign tasks. |
| Week 2 | Develop backend microservices and MySQL schema; prepare Dockerfiles. |
| Week 3 | Implement frontend and integrate APIs; test locally with Docker Compose. |
| Week 4 | Configure GitHub Actions CI/CD, deploy on AWS (EKS/EC2), final testing and documentation. |

## 5. Tools and Technologies

- Programming Languages: Python (FastAPI), JavaScript (React.js)  
- Database: MySQL  
- Web Server: Nginx  
- Containerization: Docker, Docker Compose  
- Orchestration: Kubernetes (Minikube / AWS EKS)  
- Cloud Provider: AWS (EC2, S3, RDS)  
- Version Control: Git, GitHub  
- CI/CD: GitHub Actions  
- IDEs: VS Code, PyCharm, or WebStorm

## 6. Expected Deliverables

- Functional e-commerce website accessible via browser.  
- Source code hosted on GitHub with CI/CD workflows.  
- Docker and Kubernetes deployment manifests.  
- Project documentation and architecture diagram.  
- Final presentation and demonstration of deployment pipeline.

## 7. Success Criteria

- Backend APIs fully functional and tested.  
- Frontend integrated and responsive.  
- CI/CD pipeline runs successfully on each commit.  
- Application deployed and accessible on AWS.  
- High availability achieved via Nginx load balancing and Kubernetes pods.