Capstone Project - 1 Report

Name: Shubham Saini

Registration No: 12215854

Section: KO024

Introduction

This Capstone Project focuses on automating the deployment process of a Next.js web application using DevOps tools and practices. The aim is to establish a complete CI/CD pipeline that automatically builds, tests, containers, and deploys the application to an AWS EC2 instance using infrastructure as code and monitoring solutions.

Objective of the project

The main objective is to design a fully automated CI/CD pipeline that ensures continuous integration, delivery, and deployment of a Next.js application using GitHub Actions, Docker, AWS ECR, Terraform, Ansible, Prometheus, and Grafana.

Description of the project

The project demonstrates a real-world DevOps pipeline implementation. The workflow starts when a developer pushes code to a GitHub repository. GitHub Actions triggers the CI/CD process, building a Docker image, pushing it to AWS ECR, and deploying it on an EC2 instance created through Terraform. Ansible is used to configure the EC2 environment, while Prometheus and Grafana provide monitoring and alerting for application and server health.

Scope of the project

This project covers the complete DevOps lifecycle - from infrastructure provisioning to monitoring. It includes:

1. Building a Docker image of the Next.js app.
2. Creating and configuring AWS resources using Terraform and Ansible.
3. Implementing CI/CD pipelines via GitHub Actions.

4. Monitoring infrastructure and application health with Prometheus and Grafana.

The scope also extends to scalability and fault tolerance through automation.

Use Case Model (If applicable)

The main use case is for software developers and DevOps engineers who want to deploy Next.js applications in a production environment with automation, monitoring, and infrastructure as code.

System Description

The system integrates several DevOps tools:

- GitHub Actions for automation.
- Docker for containerization.
- AWS ECR for storing container images.
- Terraform for infrastructure provisioning.
- Ansible for configuration management.
- EC2 for hosting the app.
- Prometheus and Grafana for monitoring performance and uptime.

Customer/User Profiles

Primary users include:

1. Developers - who commit code and trigger CI/CD workflows.
2. DevOps Engineers - who maintain the pipeline, infrastructure, and monitoring.
3. System Administrators - who observe performance metrics and alerts.

Assumptions and Dependencies (If applicable)

Assumptions:

- Users have access to AWS, Docker Hub, and GitHub.
- Proper IAM permissions are configured for automation.

Dependencies:

- Internet connectivity.

Capstone Project - 1 Report

- AWS CLI and Terraform installed.
- Prometheus and Grafana setup configured correctly.

Functional Requirements

1. Code commit should trigger an automatic build process.
2. Docker image should be created and pushed to ECR.
3. Terraform should provision EC2 instances.
4. Ansible should configure EC2 and deploy the app.
5. Prometheus should collect metrics.
6. Grafana should visualize system and app health.

Non-Functional Requirements

1. Scalability - should support multiple deployments.
2. Reliability - automatic rollback on failure.
3. Security - credentials managed securely in GitHub Secrets.
4. Performance - quick build and deployment times.
5. Maintainability - reusable Terraform and Ansible code.

Design

The system design integrates CI/CD automation, infrastructure provisioning, and monitoring. The architecture ensures minimal manual intervention and improved reliability.

System design

The system consists of the following workflow:

1. Developer pushes code -> GitHub Actions triggers build.
2. Docker image is created -> pushed to AWS ECR.
3. Terraform provisions EC2.
4. Ansible installs dependencies and deploys the app.
5. Prometheus and Grafana monitor metrics and display dashboards.

Capstone Project - 1 Report

E-R diagram

The project does not involve a database-centric entity relationship model since it focuses on deployment infrastructure.

DFD?s

Data Flow involves the automation process:

- Source Code -> GitHub Actions -> Docker Build -> ECR -> Terraform -> EC2 -> Prometheus & Grafana.

Database Design

The Next.js application can use any external database (like MongoDB or PostgreSQL), but it?s not part of this core DevOps automation project.

Scheduling and Estimates

The estimated project timeline:

1. Week 1-2: Research and environment setup.
2. Week 3: Create Next.js app and Dockerize it.
3. Week 4: Implement GitHub Actions for CI/CD.
4. Week 5: Integrate Terraform and Ansible.
5. Week 6: Add Prometheus and Grafana monitoring.
6. Week 7: Testing, optimization, and documentation.

Total Duration: 7 weeks.