



Automation of Fitness Static Website Deployment Using DevOps

AUTOMATIC DEPLOYMENT PROJECT

DEPI2-DEVOPs-Group3-Team

SUPERVISOR NAME | AHMED ABD EL-FATTAH

Contents

Abstract.....	2
Introduction.....	2
Technology Stack	3
System Architecture	4
Implementation Steps.....	4
Pipeline Flow	Error! Bookmark not defined.
Monitoring and Alerts	5
Challenges Faced	6
Conclusion and Future Scope.....	6
References	6

Abstract

This project presents the development of an automated deployment pipeline for a static fitness website using a range of DevOps tools and practices. The solution integrates Jenkins, GitHub, Ansible, Docker, AWS EC2, VMware, Slack, Prometheus, and Grafana to create a robust, scalable, and efficient continuous integration and continuous deployment (CI/CD) pipeline. The objective is to enable seamless code integration, testing, containerization, deployment, monitoring, and notifications in a fully automated manner.











Introduction

DevOps has revolutionized the software development lifecycle by combining development and operations into a unified process. In this project, a static fitness website is used as a case study to demonstrate the implementation of a complete CI/CD pipeline that automates every stage from code commit to deployment and monitoring.

Problem Statement	Manual deployment processes are time-consuming, error-prone, and inefficient. Automating these tasks ensures reliability, consistency, and faster time-to-market.
Objective	To develop an automated pipeline using DevOps tools that deploys a static fitness website to a virtualized or cloud environment, while also implementing monitoring and alerting systems.
Scope	The project includes CI/CD pipeline creation, deployment automation, integration of monitoring and alerting systems, and real-time notifications.

Technology Stack

To achieve end-to-end automation and observability in the deployment pipeline, a diverse set of DevOps tools and platforms were selected. Each tool plays a specific role in ensuring smooth code integration, infrastructure automation, containerization, monitoring, and communication. The following technologies collectively form the backbone of the pipeline used in this project.

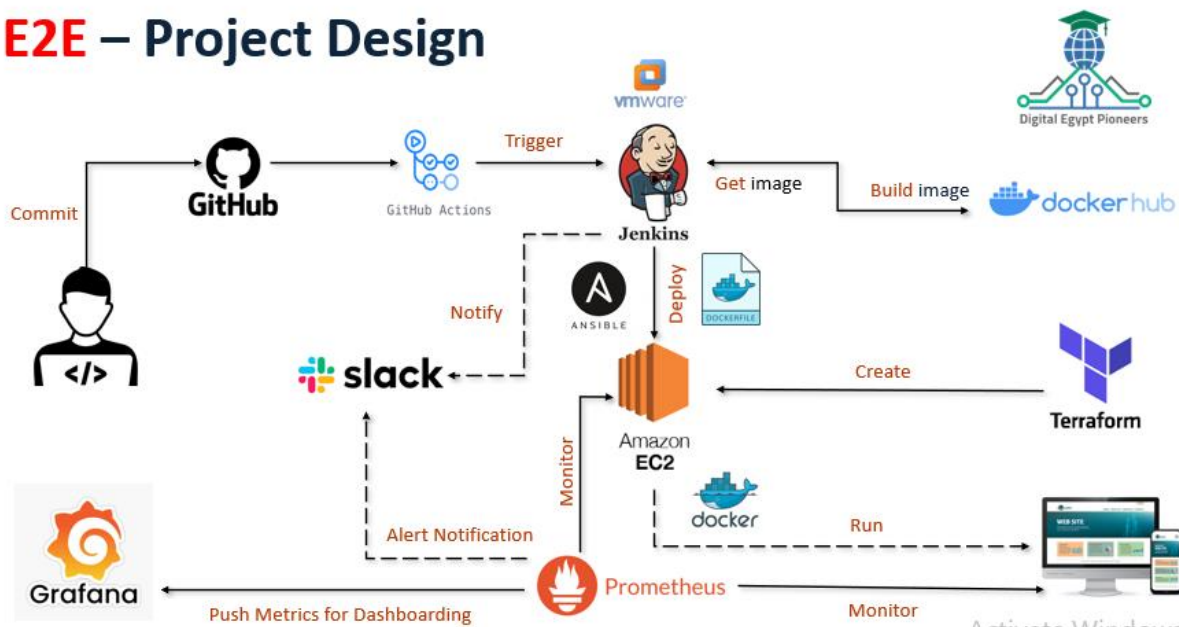
 Jenkins	Jenkins	Automates the build, test, and deployment process.
 GitHub	GitHub	Source code management and version control.
 docker	Docker	Containerization of the application.
 ANSIBLE	Ansible	Infrastructure automation and configuration management.
 aws	AWS EC2	Cloud hosting of the deployed website.
 vmware	VMware	Alternative on-premise hosting solution
 slack	Slack	Real-time notifications of pipeline status.
 Prometheus	Prometheus	Monitoring system and time-series database.
 Grafana	Grafana	Visualization and alerting for monitored metrics.
 Terraform	Terraform	Infrastructure provisioning and management as code.

System Architecture

The architecture consists of the following components:

1. Developers push code to GitHub.
2. Jenkins is triggered via webhooks and executes the pipeline.
3. Code is built and Dockerized.
4. Ansible deploys the Docker container to AWS EC2 or VMware.
5. Prometheus collects system and application metrics.
6. Grafana visualizes the metrics.
7. Slack receives pipeline status updates and alerts.

E2E – Project Design



Implementation Steps

The implementation of this project involved integrating and configuring multiple DevOps tools to establish a fully automated deployment pipeline. Each step was carefully planned to ensure seamless workflow from code commit to deployment, monitoring, and alerting. The following outlines the key phases involved in building and executing the pipeline.

- ✓ GitHub Repository Setup.
- ✓ Created a repository with the static website source code.

- ✓ Integrated GitHub webhook with Jenkins.
- ✓ Jenkins Pipeline Configuration.
- ✓ Created a declarative Jenkinsfile.
- ✓ Configured Jenkins to pull from GitHub.
- ✓ Dockerization.
- ✓ Prepared Dockerfile to containerize the static website.
- ✓ Pushed Docker images to DockerHub.
- ✓ Ansible Deployment.
- ✓ Prepared Ansible playbooks to pull Docker image and run containers.
- ✓ Configured inventory for both AWS EC2 and VMware targets.
- ✓ Hosting.
- ✓ Configured security groups and firewalls on AWS EC2.
- ✓ Deployed containers using Ansible.
- ✓ Slack Integration.
- ✓ Configured Jenkins and alert rules to send notifications.
- ✓ Monitoring with Prometheus and Grafana.
- ✓ Installed Prometheus node exporter.
- ✓ Set up Prometheus to scrape metrics.
- ✓ Created Grafana dashboards to visualize health and usage data.

Monitoring and Alerts

Monitoring plays a crucial role in maintaining the health and performance of the deployed application. To ensure visibility into system metrics and receive timely alerts, Prometheus and Grafana were integrated into the pipeline. This section highlights the setup, configuration, and functionality of the monitoring and alerting components used in the project.

- **Prometheus** monitors CPU, memory, and container health.
- **Grafana** visualizes metrics with custom dashboards.
- **Alerts** configured for high CPU/memory usage.
- Notifications sent to **Slack** channel.

Challenges Faced

While building the automated pipeline, several challenges emerged during the integration and configuration of various tools and platforms. These issues required troubleshooting, research, and adjustments to ensure a smooth and functional workflow. The following are some of the key challenges encountered during the project.

Ensuring seamless integration between multiple tools.

Conclusion and Future Scope

This project successfully implemented an automated CI/CD pipeline for a static fitness website. The integration of DevOps tools enhanced deployment speed, consistency, and observability. Future enhancements could include:

- Adding unit and integration tests.
- Integrating Terraform for infrastructure provisioning.
- Extending to dynamic web applications.

References

- ✓ Official documentation of Jenkins, Docker, Ansible, Prometheus, Grafana
- ✓ AWS EC2 and VMware documentation
- ✓ Slack API documentation
- ✓ GitHub repositories and tutorials