DevOps setup guide for PORTFOLIO website

INDEX

1. Introduction
2. Tools
3. Flowchart
4. Configure on GITHUB
5. Install Jenkins
6. Install Docker and Sonarqube
7. CI\CD Pipeline
8. Monitering and Alerts
9. Troubleshooting
10. Configuring Nginx and Docker
11. Deploying in CloudFlare
12. Summary

Introdution

This project is a Personal Portfolio Website developed using Django. It is designed to showcase my skills, projects, and achievements, while also acting as a learning platform to implement DevOps best practices such as CI/CD, containerization, and cloud deployment.

TOOLS

GITHUB --> It is a Cloud based platform used to store and share and write code.

JENKINS --> It is a open source automation server that help with the software delivery process. It is used to control and manage build, test, documentation etc. It is used to write the pipeline for the project.

DOCKER --> It is a opensource centralised platform used to containerize the application.

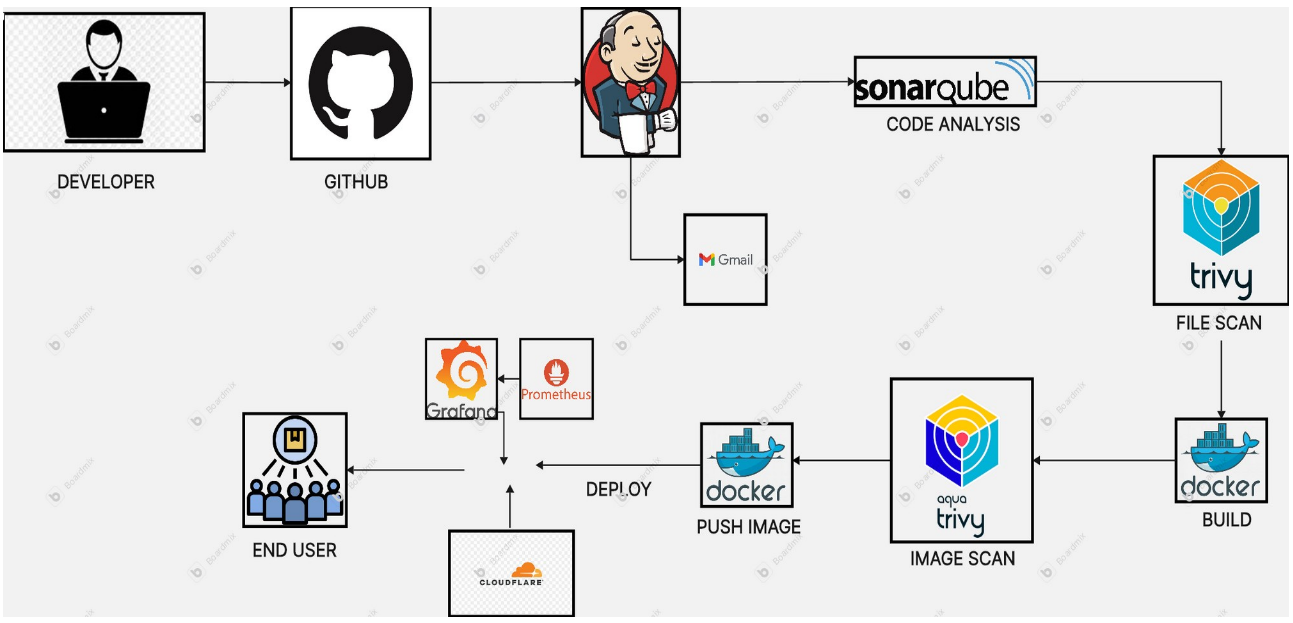
SONARQUBE --> It is a opensource platform that analyze the code quality.

TRIVY --> an open-source security scanner that finds security vulnerabilities,misconfigurations, and exposed secrets in software, containers, and infrastructure code.

PROMETHEUS --> It is an open sourse system monitering. It is a server that is used to pull metrics from different server or data sources.

GRAFANA--> It is used for visualizing and monitoring data from various sources to create interactive dashboards with charts, graphs, and alerts.

FLOWCHART


DEVELOPER
GITHUB
sonarqube
CODE ANALYSIS
Gmail
trivy
FILE SCAN
docker
BUILD
aqua trivy
IMAGE SCAN
docker
PUSH IMAGE
DEPLOY
Grafana
Prometheus
CLOUDFLARE
END USER

Configure on GITHUB

Create a GitHub repository for the web application and set it to private. Generate a personal access token to enable secure access to the repository. This token can be used for cloning, pushing, and pulling code from the repository without using a password. Ensure that the token is stored securely, as it grants access to your private repository.

Install Jenkins

Create a instance on north. Virginia (us-east-1) , give name as Portfolio-jenkins , Select Ubuntu on AMI , Select t3.medium as instance type, Select the key pair, Create a new security group or Select the existing security group that has permisson for SSH, All Traffic, All TCP, Provide the storage as 25.

Connect the instance and Install jenkins. On jenkins install the necessary plugins, Configure the sonarqube , Add Credentials , Setup Email Notification , Write the pipeline .

Install Docker and SonarQube

Create a instance on Ohio (us-east-2) , give name as Portfolio-sonar , Select Ubuntu on AMI , Select t3.medium as instance type, Select the key pair, Create a new security group or Select the existing security group that has permisson for SSH, All Traffic, All TCP, Provide the storage as 25.

Connect the instance and install Docker . Using docker install sonarqube and setup the sonarqube and add webhook

CI/CD Pipeline

- Code Commit → GitHub
- Build & Test → Jenkins
- Image Build & Scan → Docker + Trivy
- Deploy → Docker + Cloudflare
- Moniter → Prometheus + Grafana

Monitoring & Alerts

- Prometheus scrapes application and infrastructure metrics.
- Grafana visualizes metrics in dashboards.
- Alerts can be configured to send notifications via Gmail/Slack.

Troubleshooting

- Jenkins build fails → Check Jenkinsfile syntax and logs.
- Docker push fails → Check Docker login credentials and repository name

Configuring Nginx and Docker

Install Nginx outside the Docker container. Create a new configuration file in the sites-available directory, specifying the domain name and the port number of the Django project. Save the file, then create a symbolic link from sites-available to sites-enabled. Finally, restart Nginx to apply the changes.

Deploying in CloudFlare

Log in to Cloudflare and select the relevant domain. Add an A record with the IP address of the Docker instance and specify the target name, then save the configuration. This ensures that traffic to your domain is directed to your Docker-hosted application. After updating the DNS settings, it may take a few minutes for the changes to propagate globally.

Summary

The complete deployment workflow begins with setting up the application inside a Docker container and configuring Nginx to route incoming traffic to the correct Django port. The domain is then linked to the Docker instance via Cloudflare by adding the appropriate A record, ensuring that all requests reach the application. The project code is maintained in a private GitHub repository, with a personal access token enabling secure interactions with the repository. Following these steps ensures a secure, scalable, and fully functional deployment of the web application, accessible through the configured domain.