Project Report on

Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST

Submitted by

Samrajya Pujari	**(240844223037)**
Dhanashree Dhondge	**(240844223011)**
Akshat Sisodiya	**(240844223003)**
Jaspreet Sindhu	**(240844223018)**
Divyansh Baghel	**(240844223013)**

Under the guidance of

Mr. Sandeep Walvekar

In partial fulfillment of the award of Post Graduate Diploma in

IT Infrastructure, Systems and Security

(PG-DITISS)

Sunbeam Institute of Information Technology,

Pune (Maharashtra)

PG-DITISS - 2024

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed**.**

Place: Pune

Date:

Samrajya Pujari	**(240844223037)**
Dhanashree Dhondge	**(240844223011)**
Akshat Sisodiya	**(240844223003)**
Jaspreet Sindhu	**(240844223018)**
Divyansh Baghel	**(240844223013)**

CERTIFICATE

This is to certify that the project report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"**, submitted by **Samrajya Pujari** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar

Guide

Mr. Vishal Salunkhe

Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

CERTIFICATE

This is to certify that the project report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"**, submitted by **Dhanashree Dhondge** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar

Guide

Mr. Vishal Salunkhe

Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

CERTIFICATE

This is to certify that the project report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"**, submitted by **Akshat Sisodiya** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar

Guide

Mr. Vishal Salunkhe

Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

CERTIFICATE

This is to certify that the project report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"**, submitted by **Jaspreet Sindhu** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar

Guide

Mr. Vishal Salunkhe

Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

CERTIFICATE

This is to certify that the project report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"**, submitted by **Divyansh Baghel** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Mr. Sandeep Walvekar

Guide

Mr. Vishal Salunkhe

Course Coordinator

Mr. Nitin Kudale

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

APPROVAL CERTIFICATE

This Project II report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"** by **Samrajya Pujari (240844223037)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: _____

(Signature)

_____

(Name)

APPROVAL CERTIFICATE

This Project II report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"** by **Dhanashree Dhondge (240844223011)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: _____

(Signature)

_____

(Name)

APPROVAL CERTIFICATE

This Project II report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"** by **Akshat Sisodiya (240844223003)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: _____

(Signature)

_____

(Name)

APPROVAL CERTIFICATE

This Project II report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"** by **Jaspreet Sindhu (240844223018)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: _____

(Signature)

_____

(Name)

APPROVAL CERTIFICATE

This Project II report entitled **"Deploying Web App with Docker, GitHub & ELK Monitoring - DevSecOps SAST & DAST"** by **Divyansh Baghel (240844223013)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner: _____

(Signature)

_____

(Name)

CONTENTS

ABSTRACT

In the contemporary software development environment, the integration of security into the DevOps pipeline is essential for creating resilient and secure applications.

This project focuses on the design and implementation of an end-to-end DevSecOps CI/CD pipeline using AWS services, enhanced with Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) tools.

The pipeline aims to automate and streamline the software development lifecycle by incorporating AWS CodePipeline, CodeBuild, and CodeDeploy to manage code integration, builds, and deployments. Security is integrated throughout the process, with SAST tools employed to analyze source code for vulnerabilities early in the development phase, and DAST tools used to assess the runtime behavior of applications to identify potential security issues in real time.

The primary objectives of this project include creating a seamless and automated CI/CD pipeline that embeds rigorous security checks, reducing the risk of vulnerabilities, and ensuring compliance with security best practices. By integrating these tools into the pipeline, the project demonstrates how automated security testing can be effectively embedded into continuous integration and deployment workflows, enhancing both the security posture and efficiency of software development practices. This approach not only accelerates the development cycle but also ensures that security is a fundamental component of the development process.

1. INTRODUCTION

This project focuses on implementing a DevSecOps pipeline to automate security scanning and monitoring during the CI/CD process. The application code is pushed to GitHub, triggering Jenkins to build, test, and deploy the application using Docker. Security scans are performed at various stages, and logs are collected and visualized using ELK.

In the ever-evolving landscape of web applications, the process of deploying, managing, and safeguarding online services has become increasingly intricate and critical. This project introduces a comprehensive approach to web server deployment, employing Amazon Web Services (AWS) Elastic Compute Cloud (EC2) for hosting, integrating continuous integration and delivery (CI/CD) pipelines for efficient updates, implementing Nagios monitoring for proactive oversight, and leveraging the Snort Intrusion Detection System (IDS) for elevated security measures.

Web Server Deployment using AWS EC2:

Amazon EC2 provides a resilient and scalable infrastructure for hosting web applications. This project delves into the fundamentals of deploying web servers on AWS EC2 instances. Through a systematic walkthrough, we explore the process of provisioning EC2 instances, configuring networking settings, and optimizing the environment for seamless web application hosting.

Continuous Integration and Delivery (CI/CD):

The adoption of CI/CD practices has revolutionized the software development lifecycle, enhancing code quality and expediting deployment cycles. This project showcases the integration of CI/CD pipelines into the web server deployment process. By employing AWS Code Pipeline and Code Deploy, we illustrate how developers can automate code testing, deployment, and monitoring, leading to consistent and reliable application updates.

1.1 Applications

- Secure web application deployment.
- Continuous monitoring and vulnerability assessment.
- Automated DevSecOps practices for enhanced security.

1.2 Project Plan

- Setup of Jenkins and required security tools.
- Configuration of a CI/CD pipeline for automated security scanning and deployment.
- Monitoring system health and security logs using ELK Stack.

Table: Activities Details

Sr. No.	ACTIVITY	WEEK			
		1	2	3	4
1	Project group formation				
2	Project work to be started in respective labs				
3	First review with PPT presentation				
4	Design Use-Case view as per project				
5	Design Block diagram as per project				
6	Second review with PPT presentation				
7	Selection				
8	Final review with PPT presentation				
9	Implementation coding as per project				
10	Testing, Troubleshooting with different techniques				
11	Created Soft copy of project and then final hard copy				

2. SYSTEM DEVELOPMENT AND DESIGN

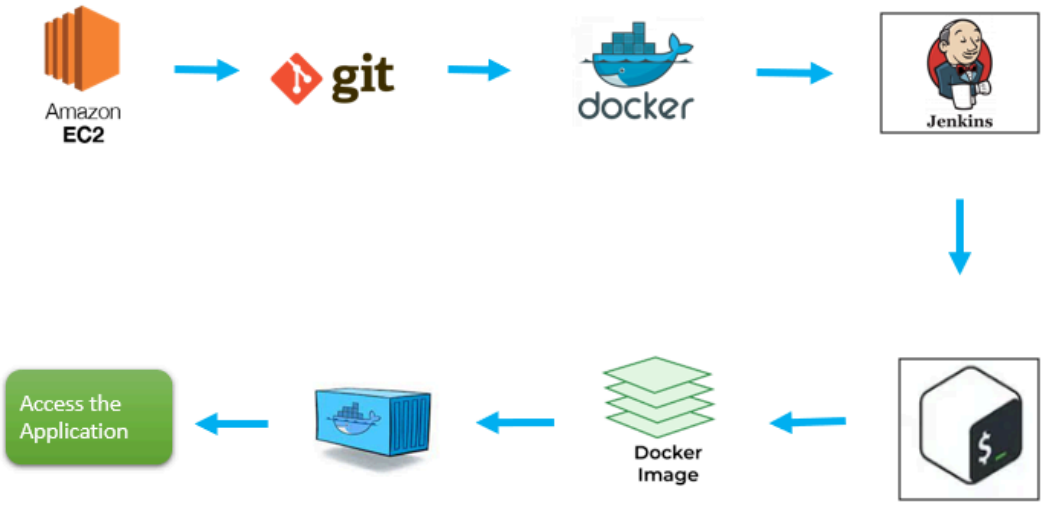
2.1 Proposed System

The proposed system automates the deployment and security assessment of web applications. It ensures that vulnerabilities are detected before deployment and provides real-time monitoring.

We propose a system where we are setting up Amazon EC2 instances to host your web application. Storing our application code in a version-controlled repository (Git-Hub).

Set up a CI/CD pipeline using Jenkins. On code changes, trigger an automated build and deployment process using Jenkins. Setting up another EC2 instance to host the Nagios monitoring server. Install Nagios plugins to monitor various aspects of your infrastructure, such as server health, resource usage, and application responsiveness. Configuring AWS Security Groups to control inbound and outbound traffic to our EC2 instances. And also configuring Snort rules to detect various types of network traffic anomalies and security threats.

2.2 Flow chart



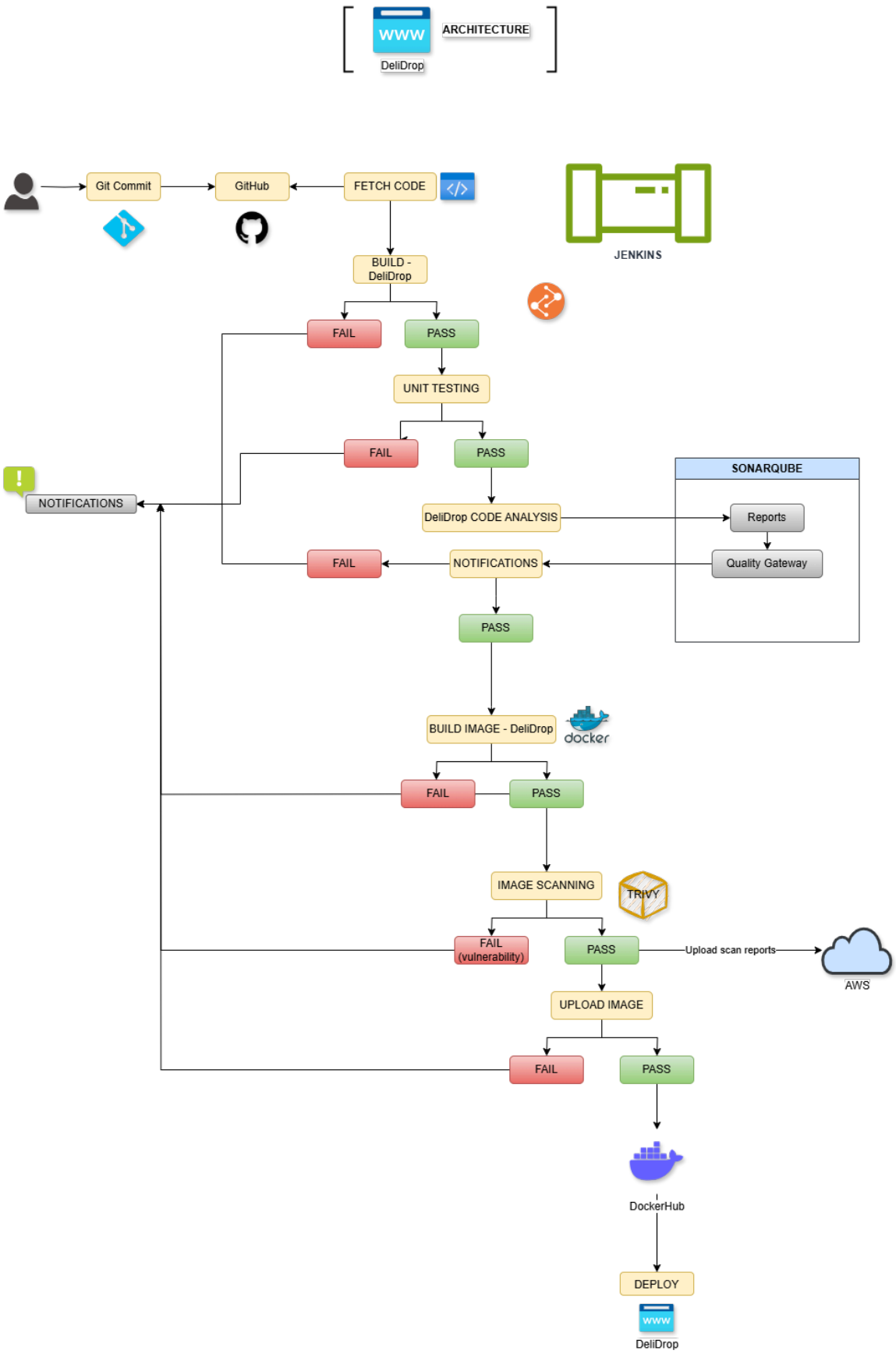

ARCHITECTURE
DeliDrop
Git Commit
GitHub
FETCH CODE
JENKINS
BUILD - DeliDrop
FAIL
PASS
UNIT TESTING
FAIL
PASS
SONARQUBE
NOTIFICATIONS
DeliDrop CODE ANALYSIS
Reports
FAIL
NOTIFICATIONS
Quality Gateway
PASS
BUILD IMAGE - DeliDrop
docker
FAIL
PASS
IMAGE SCANNING
TRIVY
FAIL (vulnerability)
PASS
Upload scan reports
AWS
UPLOAD IMAGE
FAIL
PASS
DockerHub
DEPLOY
DeliDrop

Figure: Flowchart

SCM Code Checkout from GitHub → Jenkins → SAST (SonarQube) → DAST (OWASP ZAP) → Trivy (Container Security) → Deploy (Docker, AWS EC2) → ELK Monitoring.

1. Developer pushes code to GitHub
2. Jenkins triggers the build
3. SonarQube performs SAST analysis
4. OWASP ZAP performs DAST scanning
5. Trivy scans the Docker container for vulnerabilities
6. Application is deployed
7. ELK collects and visualizes logs

2.3 Technology used

- **Security Tools:** SonarQube (SAST), OWASP ZAP (DAST), Trivy (Container Security).
- **CI/CD Tools:** Jenkins, GitHub.
- **Containerization:** Docker.
- **Monitoring:** ELK Stack (ElasticSearch, Logstash, Kibana).
- **Cloud Infrastructure:** AWS EC2.

2.3.1 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon it allows you to rent virtual servers in the cloud, known as instances, to run your applications and workloads. EC2 provides a scalable and flexible infrastructure that enables you to quickly deploy and manage virtual servers without the need to invest in physical hardware.

2.3.2 Git

Git is a distributed version control system (VCS) designed to manage source code history and facilitate collaborative software development.

2.3.3 Docker

Docker is an open-source platform that allows you to automate the deployment, scaling, and management of applications using containerization. Containers are lightweight, portable, and isolated environments that package an application and its dependencies together.

2.3.4 Jenkins

Jenkins is an open-source automation server that facilitates the continuous integration and continuous delivery (CI/CD) of software projects. It helps automate various tasks related to building, testing, and deploying applications, making the development and release process more efficient and reliable.

3. INSTALLATION & CONFIGURATION

3.1 JDK Installation

sudo apt update
sudo apt install fontconfig openjdk-17-jre
java -version

3.2 Jenkins Installation

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list >
/dev/null
sudo apt-get update
sudo apt-get install jenkins

3.3 Docker Installation

sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/debian $(. /etc/os-release && echo
"$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin

3.4 SonarQube Installation

docker run -itd --name sonarqube-server -p 9000:9000 sonarqube:lts-community

3.5 Trivy Installation

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb generic main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy

3.6 OWASP ZAP Configuration

docker run --memory=4g --rm zaproxy/zap-stable zap-baseline.py -t http://192.168.80.171:9080 --fail-on WARN-INPROG,INFO || true
docker run --memory=4g --rm zaproxy/zap-stable zap-full-scan.py -t http://192.168.80.171:9080 --fail-on WARN-INPROG,INFO || true

3.7 ELK Setup

Elasticsearch:

Install JDK: sudo apt-get install -y default-jdk curl
Download and install Elasticsearch:

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.17.1-amd64.deb

sudo dpkg -i elasticsearch-8.17.1-amd64.deb

Configure Elasticsearch & then start Elasticsearch:

sudo vim /etc/elasticsearch/elasticsearch.yml
cluster.name: my-application
node.name: node-1
network.host: 0.0.0.0
xpack.security.enabled: false

sudo systemctl daemon-reload
sudo systemctl enable --now elasticsearch

Logstash:

Install Logstash:

wget https://artifacts.elastic.co/downloads/logstash/logstash-8.17.1-amd64.deb
sudo dpkg -i logstash-8.17.1-amd64.deb

Kibana:

Install Kibana:

wget https://artifacts.elastic.co/downloads/kibana/kibana-8.17.1-amd64.deb
sudo dpkg -i kibana-8.17.1-amd64.deb
sudo systemctl enable --now kibana

Metricbeat Setup

Install Metricbeat:

curl -L -O https://artifacts.elastic.co/downloads/beats/metricbeat/metricbeat-8.17.1-amd64.deb

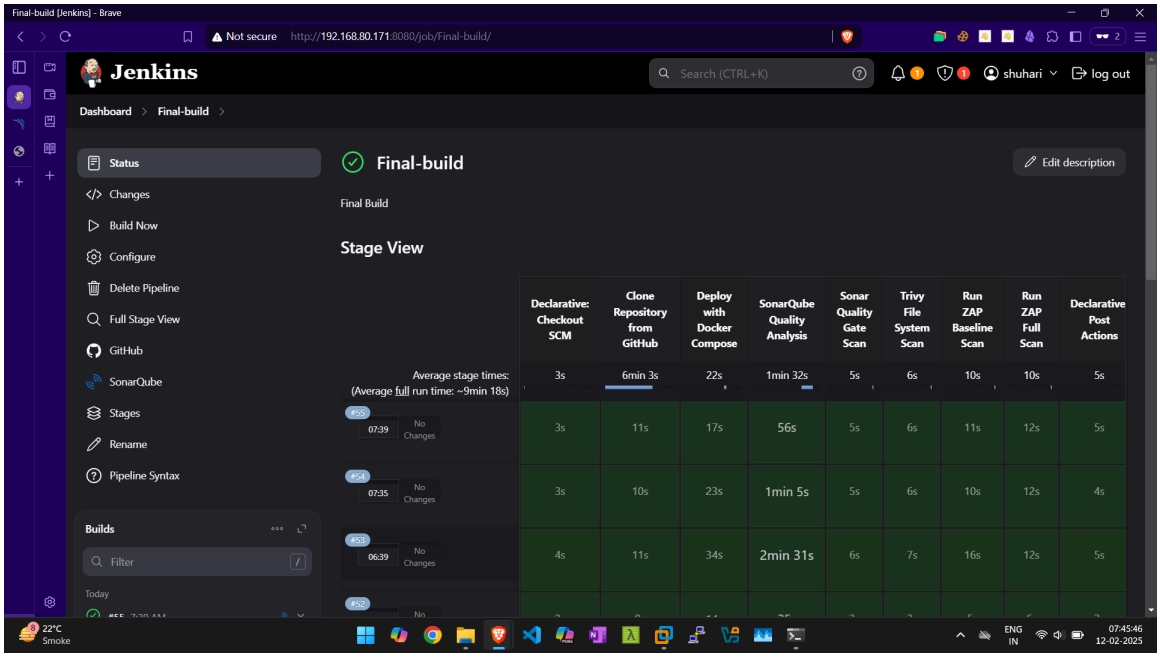
sudo dpkg -i metricbeat-8.17.1-amd64.deb

Configure Metricbeat & Start Metricbeat:

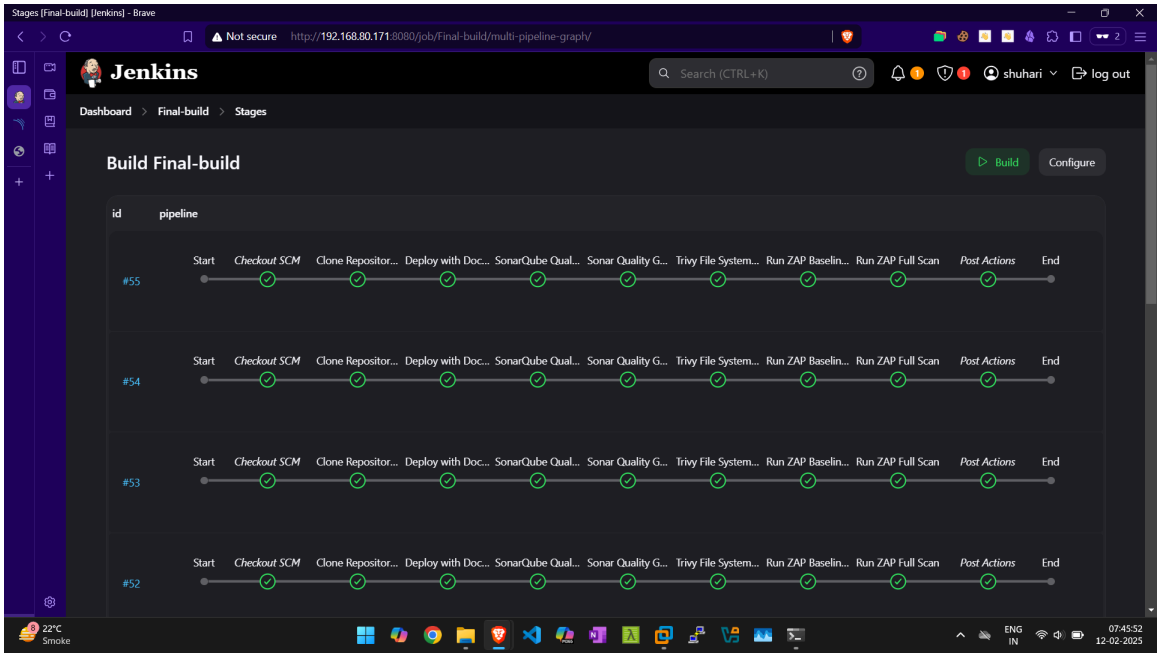
sudo systemctl enable --now metricbeat

4. Project Output

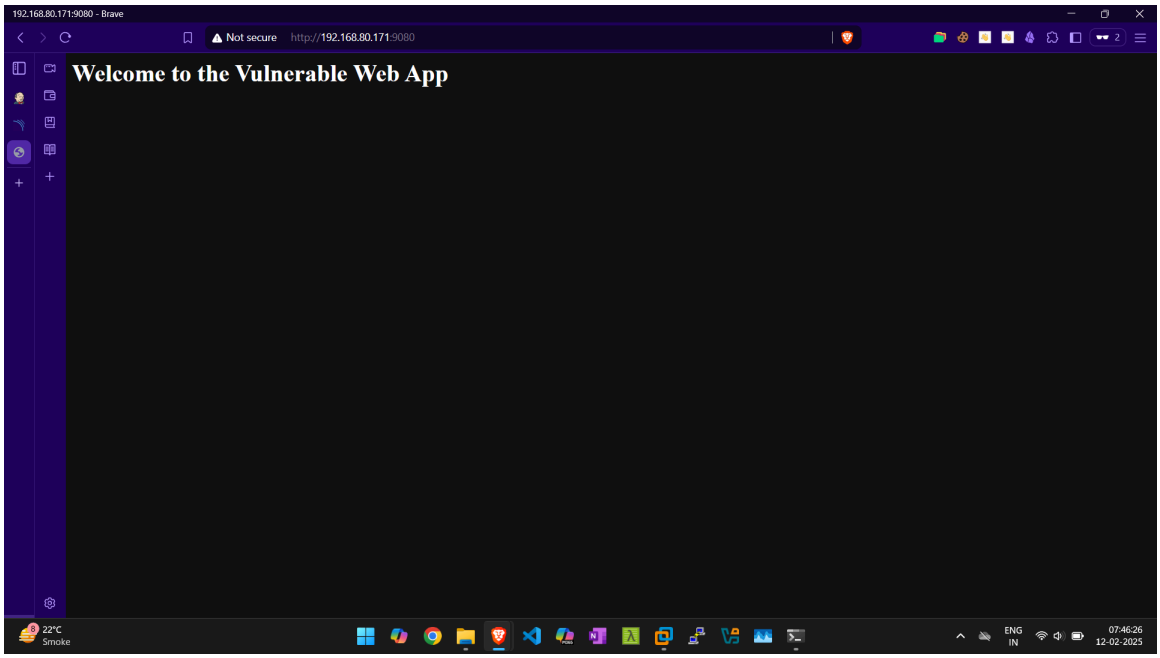
4.1 Jenkins CICD Pipeline



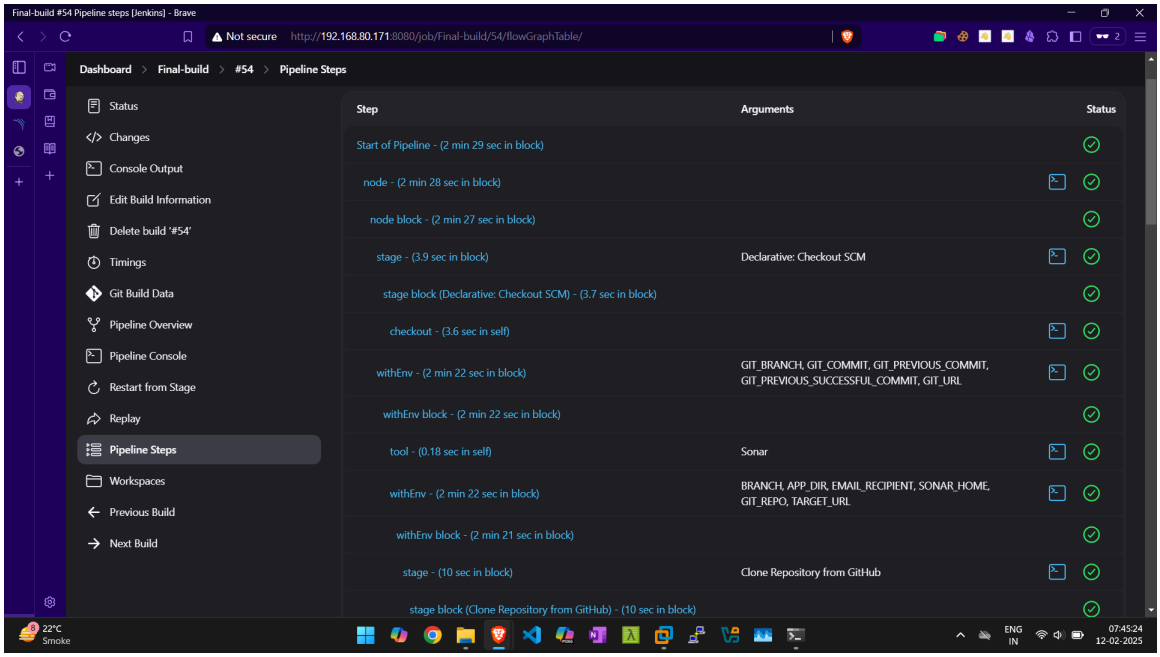
4.2 Jenkins pipeline stages

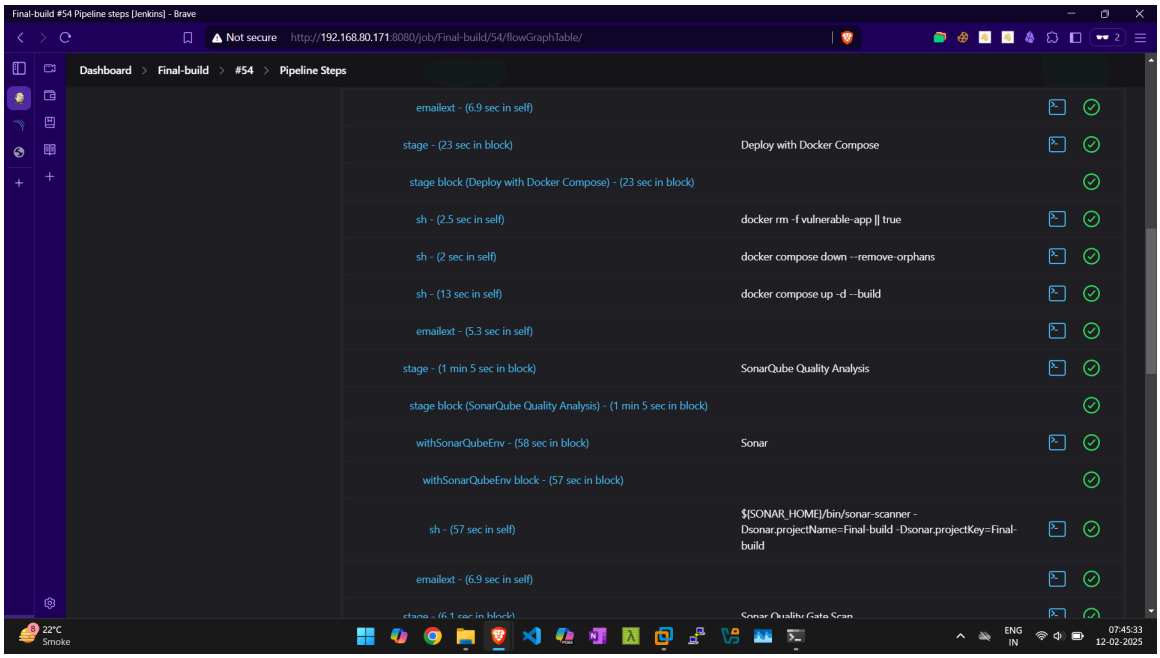


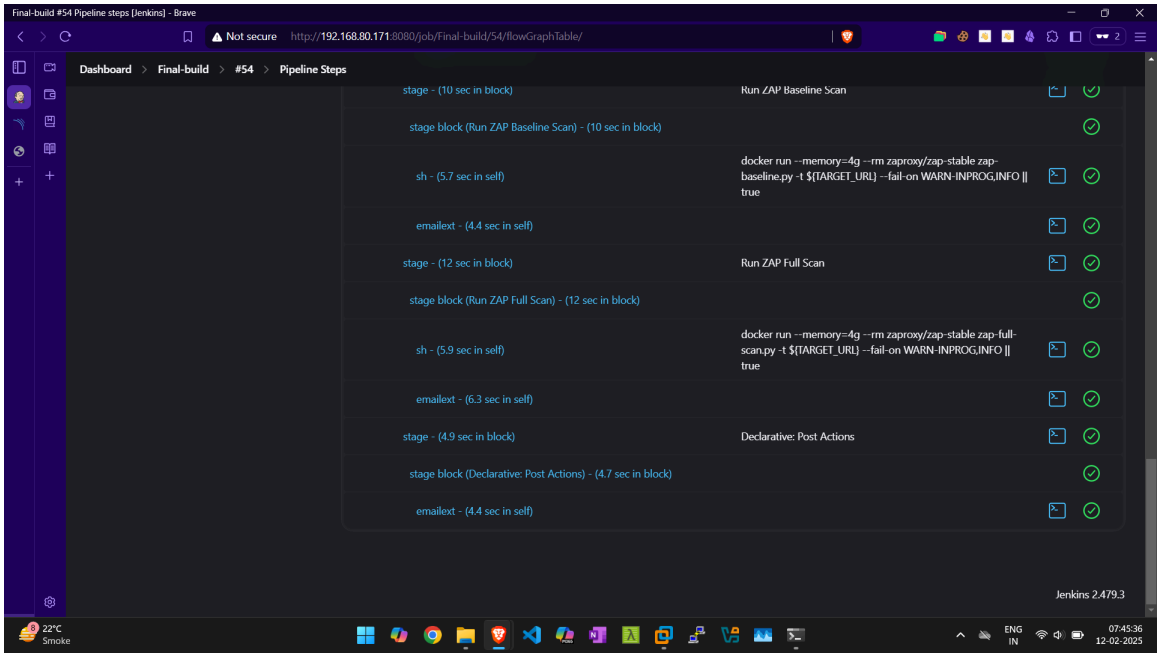
4.3 Web Application



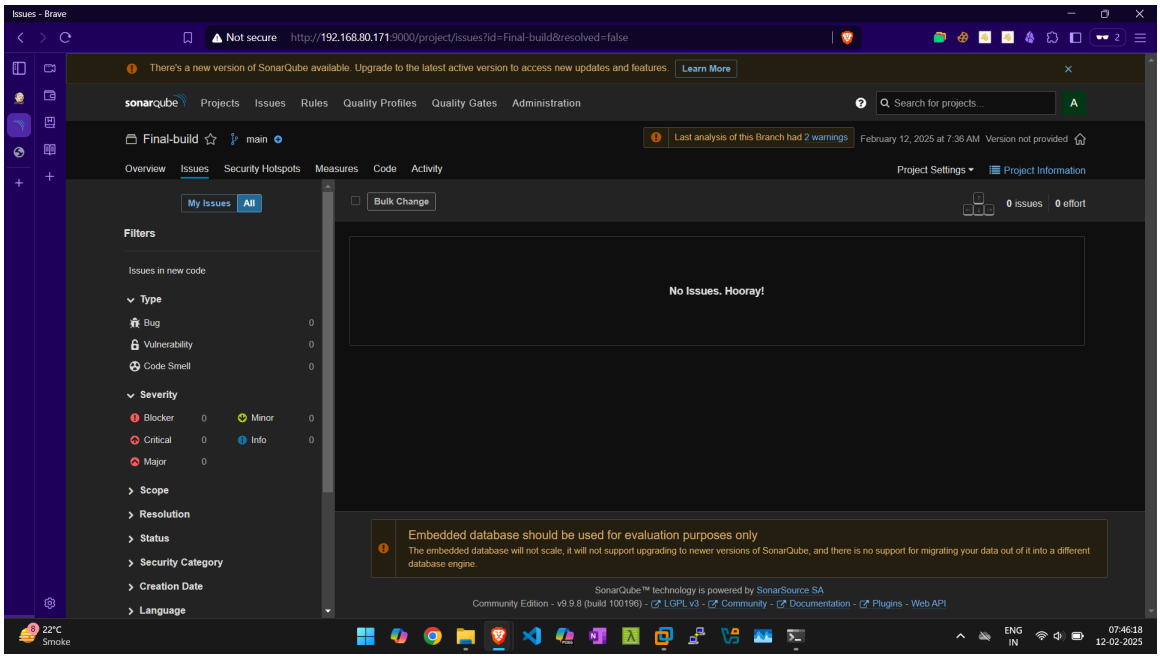
4.4 Pipeline Steps & stages



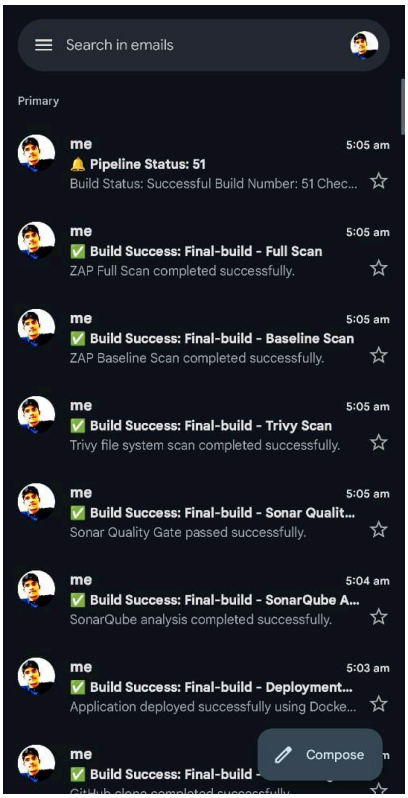

Final-build #54 Pipeline steps [Jenkins] - Brave
Not secure http://192.168.80.171:8080/job/Final-build/54/flowGraphTable/
Dashboard > Final-build > #54 > Pipeline Steps
emailext - (6.9 sec in self)
stage - (23 sec in block) Deploy with Docker Compose
stage block (Deploy with Docker Compose) - (23 sec in block)
sh - (2.5 sec in self) docker rm -f vulnerable-app || true
sh - (2 sec in self) docker compose down --remove-orphans
sh - (13 sec in self) docker compose up -d --build
emailext - (5.3 sec in self)
stage - (1 min 5 sec in block) SonarQube Quality Analysis
stage block (SonarQube Quality Analysis) - (1 min 5 sec in block)
withSonarQubeEnv - (58 sec in block) Sonar
withSonarQubeEnv block - (57 sec in block)
sh - (57 sec in self) ${SONAR_HOME}/bin/sonar-scanner -Dsonar.projectName=Final-build -Dsonar.projectKey=Final-build
emailext - (6.9 sec in self)
22°C Smoke
ENG IN
07:45:33 12-02-2025


Final-build #54 Pipeline steps [Jenkins] - Brave
Not secure http://192.168.80.171:8080/job/Final-build/54/flowGraphTable/
Dashboard > Final-build > #54 > Pipeline Steps
stage - (10 sec in block) Run ZAP Baseline Scan
stage block (Run ZAP Baseline Scan) - (10 sec in block)
sh - (5.7 sec in self) docker run --memory=4g --rm zaproxy/zap-stable zap-baseline.py -t ${TARGET_URL} --fail-on WARN-INPROG,INFO || true
emailext - (4.4 sec in self)
stage - (12 sec in block) Run ZAP Full Scan
stage block (Run ZAP Full Scan) - (12 sec in block)
sh - (5.9 sec in self) docker run --memory=4g --rm zaproxy/zap-stable zap-full-scan.py -t ${TARGET_URL} --fail-on WARN-INPROG,INFO || true
emailext - (6.3 sec in self)
stage - (4.9 sec in block) Declarative: Post Actions
stage block (Declarative: Post Actions) - (4.7 sec in block)
emailext - (4.4 sec in self)
Jenkins 2.479.3
22°C Smoke
ENG IN
07:45:36 12-02-2025

4.5 Test Reports



4.6 Email Notifications



5. CONCLUSION

5.1 Conclusion

This project is the successful implementation of a DevSecOps pipeline for deploying a secure web application. The integration of SAST, DAST, and container security tools ensures vulnerabilities are detected early in the deployment process. ELK Stack enhances monitoring capabilities, providing real-time security insights. Hence, we have successfully deployed a highly available and secure web server environment on Amazon Web Services (AWS). And ensured the reliability, performance, and security of the web application while maintaining efficient development and operational processes.

5.2 Future Scope

1. Expansion of security checks to include Infrastructure as Code (IaC) scanning.
2. Integration with cloud-native security solutions such as AWS Security Hub.
3. Implementation of AI-driven security analytics for proactive threat detection.

REFERENCES

1. https://aws.amazon.com/
2. https://github.com/ketansonwane1/Port-Scanner
3. https://github.com/krishnaacharyaa/wanderlust.git
4. https://www.youtube.com/watch?v=CoU38rJIjRY&t=2804s
5. https://medium.com/@pardhikhush/devsecops-end-to-end-cicd-project-devops-engineer-github-sonarqube-owasp-trivy-docker-8fe72265f7ea
6. https://github.com/praveensirvi1212/DevSecOps-project
7. https://www.youtube.com/watch?v=AaVO1Mvr3q4