Week 4 Report – Buildable Fellowship

Topic: GitHub Actions and Security Automation

Introduction

In Week 4 of the Buildables Fellowship, my focus was on strengthening **automation and security** within my DevOps workflow using **GitHub Actions**.

This week, I extended my previous **Flask + React full-stack application** by building an **automated CI/CD pipeline** integrated with **Static Application Security Testing (SAST)** tools. The goal was to ensure that every code change triggers a secure, reliable build, test, and deployment process.

I also explored **Dynamic Application Security Testing (DAST)** tools and learned how **Observability fundamentals** help in maintaining visibility, reliability, and performance in production systems.

This week's work gave me a clearer view of **DevSecOps** — embedding security and observability directly into continuous integration pipelines.

Key Learnings

1. GitHub Actions – CI/CD Automation

- Implemented an **end-to-end GitHub Actions pipeline** for my Flask + React project.
- The workflow automates building, testing, and Docker image deployment on each **push** or **pull request** to the main branch.
- Integrated steps for running unit tests and performing a security scan before pushing images to Docker Hub.

Pipeline Highlights:

- Trigger: VCS events (push/pull request)
- Actions: Checkout → Install dependencies → Run tests → Build Docker image → Push to Docker Hub
- Secret management via GitHub Secrets for DockerHub credentials

Real-World Relevance:
This pipeline structure mirrors how modern organizations ensure automated, secure, and reproducible builds in production environments.

2. SAST (Static Application Security Testing)

- Integrated **Bandit** to perform security scanning on the **Flask backend** source code.
- Configured Bandit to detect insecure imports, unsafe function calls, and potential vulnerabilities.
- The pipeline automatically runs the scan and uploads the **bandit-report.json** as an artifact for tracking.

Why SAST?
SAST ensures vulnerabilities are caught early — before deployment — aligning with **OWASP Top 10** standards.

3. DAST (Dynamic Application Security Testing)

While I didn't perform a live DAST implementation this week, I studied how it complements SAST by testing **running applications** for runtime vulnerabilities.

Common DAST Tools:

- **OWASP ZAP** – for detecting runtime web application issues.
- **Burp Suite** – used in enterprise-level testing.

Key Difference:
SAST analyzes **source code**, while DAST evaluates **deployed applications** in real or staging environments.

4. Observability Fundamentals

- Explored the **three pillars of observability**: Logs, Metrics, and Traces.
- Used **Prometheus** and **Grafana** locally to simulate metrics collection and visualization for the Flask app.
- Understood how observability tools provide insights into system health, performance, and failures.

Example:
Prometheus was used to scrape metrics exposed by Flask endpoints, while Grafana was configured to visualize request latency and error rates.

Real-World Insight:
Observability ensures proactive issue detection — allowing DevOps teams to identify bottlenecks before they affect users.

Hands-on Practice

During this week, I:

- Created a **GitHub Actions CI/CD pipeline** for my Flask + React project.
- Integrated **Bandit SAST scanning** for Python security analysis.
- Configured automated **Docker image builds and pushes** to Docker Hub.
- Experimented with **Prometheus and Grafana** for a basic observability setup.

This practice improved automation, security posture, and monitoring visibility across the application lifecycle.

Conclusion

Week 4 was a key milestone in my DevOps journey. I successfully automated my project's CI/CD process using **GitHub Actions**, introduced **SAST scanning** for secure code analysis, and understood how **observability** ensures transparency in modern deployments.

These enhancements represent practical DevSecOps principles — combining development, security, and operations in one automated workflow.

The knowledge gained this week directly strengthens my ability to build secure, automated, and production-ready pipelines.