DevOps CI/CD Project Proposal

Name:	Kushi Varadaraj
Roll Number:	10035
Date:	January 2026

1. Project Title

Automated CI/CD Pipeline with DevSecOps Practices using GitHub Actions

2. GitHub Repository URL

https://github.com/kushivaradaraj/devops-project

3. Application Description

For this project, I am creating a Java-based REST API application using Spring Boot framework. The application exposes simple HTTP endpoints that can be used to check if the service is healthy and running properly.

The main components of my application are: a health endpoint (/health) that returns "OK" when the app is running, a greeting endpoint (/hello) that welcomes users, and a Calculator service that performs basic arithmetic operations. I included the Calculator to have testable business logic for demonstrating unit tests in the pipeline.

I will containerize this application using Docker. The container will use a multi-stage build approach where one stage compiles the code and another stage runs it. This makes the final image smaller and more secure.

4. CI/CD Problem Statement

Without automation, software development faces several challenges:

- Code quality is inconsistent because manual reviews miss things
- Bugs reach production because developers skip running tests locally
- Security issues in the code or libraries are discovered too late
- Deployments fail because of environment differences (works locally, fails in production)
- Release cycles are slow due to manual testing and deployment steps

To address these issues, I am implementing a CI/CD pipeline using GitHub Actions. The pipeline will automatically trigger whenever I push code to GitHub. It will build, test, scan for security issues, containerize, and publish my application without any manual steps. This approach is called "shift-left" because we catch problems early in the development process rather than later in production.

5. Chosen CI/CD Stages and Justification

Stage	Tool Used	Justification
Code Checkout	actions/checkout	Downloads my code from GitHub to the runner
Java Setup	actions/setup-java	Installs Java 17 which is needed to build Spring Boot
Code Linting	Maven Checkstyle	Checks coding standards to maintain code quality
Unit Testing	JUnit 5	Runs automated tests to verify business logic works
Application Build	Maven	Compiles the code and packages it as a JAR file
SAST Scan	GitHub CodeQL	Scans source code for security flaws like injection attacks
Dependency Scan	OWASP Dependency Check	Finds vulnerabilities in third-party libraries
Docker Build	Docker Buildx	Creates a container image with multi-stage build
Container Scan	Trivy	Scans Docker image for OS and library vulnerabilities
Smoke Test	curl commands	Verifies container starts and responds correctly
Registry Push	Docker Hub	Publishes the verified image for deployment

The stages are arranged so that quick checks run first (fail-fast principle). Linting takes seconds while security scans take minutes, so running linting first saves time if there are style issues. Security stages run before Docker push so that vulnerable images never reach the registry.

6. Expected Outcomes

After completing this project, I expect:

- An automated CI/CD pipeline that triggers on every git push
- Code quality enforcement through automated linting checks
- Early bug detection through automated unit tests
- Security integrated at multiple stages (SAST, SCA, container scanning)

- A containerized application that runs consistently everywhere
- Practical understanding of DevSecOps and why each pipeline stage matters

Through this project, I aim to learn how industry-standard CI/CD pipelines are built and understand the reasoning behind each stage, not just the implementation.