Advanced DevOps CI/CD Project

(GitHub Actions–Based Continuous Integration Pipeline)

Overview

The **Advanced DevOps CI/CD Project** is a core component of your DevOps assessment and a **key addition to your DevOps portfolio**.

This project evaluates your ability to **design, implement, and reason about a real-world CI pipeline**, incorporating **security, quality gates, automation, and containerization**, rather than merely running builds.

> The focus is **not** on writing the longest GitHub Actions YAML file,
> but on **why each stage exists**, **what risks it mitigates**, and **how it improves software delivery**.

Project Objective

Important

The objective is **NOT** to:

- Just "make the build green"
- Copy-paste a CI/CD pipeline from the internet
- Focus only on tools without reasoning

You are expected to:

- Identify a **real-world application** built by the learner (Java preferred, but flexible)
- Design a **production-grade CI/CD pipeline** using **GitHub Actions**
- Integrate **quality checks, security scans, and container validation**
- Clearly explain **why each CI/CD stage is required**
- Demonstrate understanding of **shift-left security and DevSecOps principles**

What You Are Building

You will implement a **complete CI/CD pipeline** that:

1. Builds an application
2. Ensures code quality
3. Performs static and dependency security scans
4. Packages the application into a Docker image
5. Scans the container for vulnerabilities
6. Performs basic runtime validation
7. Pushes a trusted image to DockerHub

Project Scope & CI/CD Domains

Your CI pipeline must cover **at least ONE** but preferably **multiple** of the following DevOps domains:

1. Continuous Integration (CI)

Examples:

- Automated builds on every push
- Unit test execution
- Dependency caching
- Fail-fast pipelines

2. Code Quality & Linting

Examples:

- Java Checkstyle
- Enforcing coding standards
- Identifying maintainability issues early

3. DevSecOps (Security in CI)

Examples:

- Static Application Security Testing (SAST)
- Software Composition Analysis (SCA)
- Container vulnerability scanning
- Security findings surfaced in GitHub Security tab

4. Containerization

Examples:

- Docker image build
- Container smoke testing
- Image publishing to DockerHub

Mandatory CI/CD Stages (Reference Implementation)

Your GitHub Actions workflow **must be conceptually similar** to the following stages:

CI Trigger

- Trigger on:
  - Push to `master`
  - Manual execution (`workflow_dispatch`)

Required CI Stages

Stage	Purpose
Checkout	Retrieve source code
Setup Runtime	Install Java / language runtime
Linting	Enforce coding standards
SAST	Detect code-level vulnerabilities
SCA	Detect vulnerable dependencies
Unit Tests	Validate business logic
Build	Package application
Docker Build	Create container image
Image Scan	Detect OS & library vulnerabilities
Runtime Test	Validate container behavior
Registry Push	Publish trusted image

Reference CI Pipeline (Skeleton)

https://github.com/vilasvarghesescaler/docker-k8s/blob/master/.github/workflows/ci.yml

(Pipeline stages continue exactly as implemented in your provided workflow)

Why Each CI Stage Exists (Critical for Evaluation)

Stage	Why It Matters
Linting	Prevents technical debt
Unit Tests	Prevents regressions
CodeQL	Detects OWASP Top 10 issues
Dependency Check	Identifies supply-chain risks
Trivy	Prevents vulnerable images from shipping
Container Test	Ensures image is runnable
Docker Push	Enables downstream CD

CD stage

This should be a separate pipeline. You should be deploying to a kubernetes cluster in this pipeline and preferably you can do a dummy DAST.

Project Timeline & Process

1. Individual / Team Formation

- Team Size: **1–5 members** (recommended)
- All members must independently implement a CI/CD pipeline
- During the demo the trainer can ask any individual to demo (or all of them to demo as well)

---

2. Proposal Submission

Note:

All members should submit their Individual proposal documents.

You must submit:

- Project Title
- GitHub Repository URL
- Application Description
- CI/CD Problem Statement
- Chosen CI/CD stages and justification
- Expected outcomes

Proposal file format:

`IndividualName_ScalerStudentID_DevOps_CI_Proposal.pdf`

Submit your Project Proposal here - Google Form

---

3. Proposal Review

- Approval based on:
  - Practical relevance
  - CI/CD completeness
  - Security inclusion

---

4. Final Submission

Deadline: Strict (No late submissions) 18th Jan, 2026 (Changed to 20th Jan, 2026)

Submit your Final Project here - [Google Form]

Submission Requirements

1. Project Report (Max 10 Pages)

Must include:

1. Problem Background & Motivation
2. Application Overview
3. CI/CD Architecture Diagram
4. CI/CD Pipeline Design & Stages
5. Security & Quality Controls
6. Results & Observations
7. Limitations & Improvements

---

2. Additional Files

- **GitHub Actions Workflow**
- **Application Source Code**
- **Dockerfile**
- **README**
  - How to run locally
  - Secrets configuration
  - CI explanation

---

3. Secrets Configuration (Mandatory)

Learners must configure the following **GitHub Secrets**:

Secret Name	Purpose
DOCKERHUB_USERNAME	Docker registry user
DOCKERHUB_TOKEN	Secure registry access

Marks deducted if secrets are hardcoded, issues found during final execution, trace of execution missing etc.

File Organization (indicative)

```
project-root/
├── .github/workflows/ci.yml
├── .github/workflows/cd.yml
├── src/
├── Dockerfile
├── pom.xml
├── README.md
```

Marking Scheme (Total: 100 Marks)

Component	Weightage
Problem Statement	10%
Pipeline Design & Logic	20%
Security Integration	15%
Insights, Reasoning and VIVA	40%
Code & YAML Quality	15%

Key Evaluation Philosophy

> **DevOps is not about tools alone.**
> It is about **automation, reliability, security, and repeatability**.

Pipelines that demonstrate:

- Thoughtful stage ordering
- Clear security gates
- Proper failure handling

- Clean documentation

will score significantly higher than complex but poorly explained workflows.
