DEVOPS

DevOps CI/CD Pipeline Implementation

Submitted by

AJAY VETRI NJ - 711022205002

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

SEVENTH SEMESTER

INFO INSTITUTE OF ENGINEERING, COIMBATORE – 641107

NOVEMBER/DECEMBER – 2025

<u>BONAFIDE CERTIFICATE</u>

Certified that this project “DevOps CI/CD Pipeline Implementation” is the Bonafide work of AJAY VETRI N J (711022205002) who carried out the project work under any supervision.

SIGNATURE	**SIGNATURE**
STAFF COORDINATOR	**HEAD OF THE DEPARTMENT**
Mr. K. MADESWARAN M.E.,	**Mr. A. AROCKIYASELVARAJ M.E., (Ph.D.,)**
ASSISTANT PROFESSOR	**HEAD OF THE DEPARTMENT**
DEPT. INFORMATION TECHNOLOGY	DEPT. INFORMATION TECHNOLOGY
INFO INSTITUTE OF ENGINEERING,	INFO INSTITUTE OF ENGINEERING,
KOVILPALAYAM COIMBATORE - 641107	KOVILPALAYAM COIMBATORE - 641107

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely thank to Tamil Nadu Skill Development Corporation (TNSDC), Naan Mudhalvan Platform and DEVOPS for encouragement towards our project work for providing necessary skill training.

We sincerely thank our Principal Dr. N. KOTTISWARAN, M.E., Ph.D., and Head of the Department Mr. A. AROCKIYASELVARAJ, M.E., (Ph.D.,) and also Staff Coordinator Mr. M. MADESWARAN M.E., for his encouragement towards our project works.

We also thank our project guide and our parents for the complete and whole hearted support, motivation guidance and help in making our project activities.

ABSTRACT

DevOps CI/CD Pipeline Implementation

This project outlines the comprehensive implementation of a DevOps Continuous Integration and Continuous Delivery (CI/CD) pipeline using an open-source toolstack designed to automate software development workflows. The project encompasses the complete lifecycle of application development, from source code management to deployment, ensuring quality assurance at every stage.

Core Pipeline Components: The project integrates multiple industry-standard tools to create a seamless automated pipeline. It utilizes Git for source code management with EGit plugin integration in Eclipse IDE, enabling developers to perform essential version control operations including cloning repositories, committing changes, pushing updates, and managing branches. Jenkins serves as the central continuous integration server, orchestrating the entire pipeline through interconnected jobs that automate compilation, testing, code quality analysis, and deployment processes.

Quality Assurance Framework: A robust quality management system is established through SonarQube for static code analysis, enforcing coding standards and identifying potential bugs, vulnerabilities, and code smells. The pipeline implements JaCoCo for comprehensive code coverage reporting, ensuring adequate test coverage through JUnit unit testing. Maven handles build automation, managing project dependencies and executing various lifecycle goals that drive the CI/CD process.

Automated Deployment Architecture: The pipeline features a multi-stage job structure in Jenkins, beginning with setup and source code checkout, progressing through compilation, static analysis, unit testing, and code coverage analysis, culminating in WAR file generation and deployment to Artifactory repository management.

The application aims to demonstrate practical DevOps implementation by establishing an automated, continuous delivery environment that minimizes manual intervention, reduces deployment risks, and accelerates software release cycles. By leveraging industry-standard tools and best practices, this project provides hands-on experience in building enterprise-grade CI/CD pipelines that enhance development efficiency, maintain code quality, and ensure reliable application deployment.

INTRODUCTION

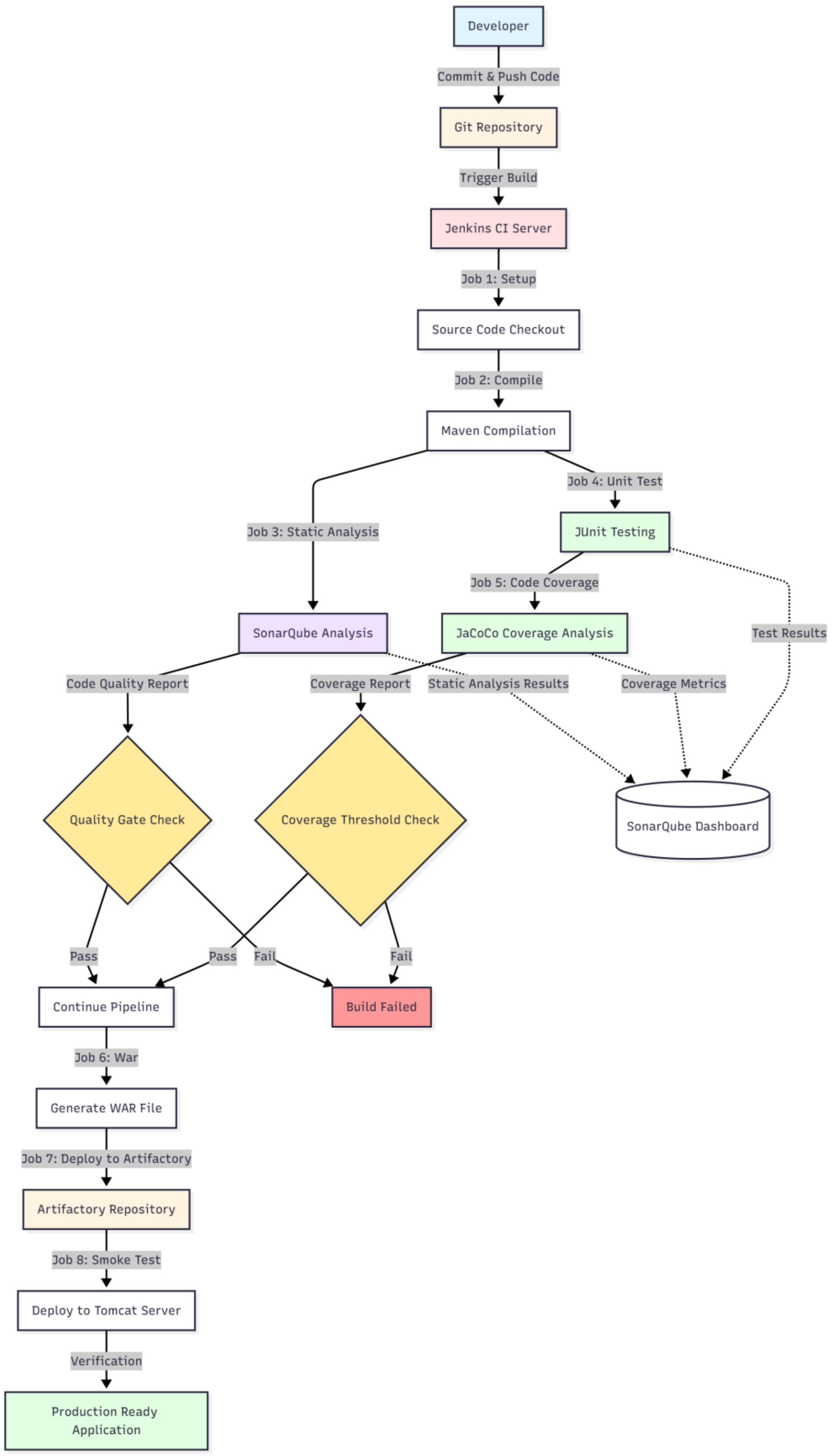
In today's rapidly evolving software development landscape, DevOps practices play a vital role in meeting the demands of organizations seeking efficiency, quality assurance, and faster time-to-market for software products. The DevOps Continuous Integration and Continuous Delivery (CI/CD) Pipeline Implementation is a comprehensive solution designed to automate the entire software development lifecycle, from source code management to production deployment. By integrating industry-standard open-source tools into a unified pipeline, this project addresses the fundamental needs of modern software development teams to deliver reliable, high-quality applications consistently.

The project encompasses multiple interconnected components that work cohesively to create an automated workflow. Git serves as the distributed version control system enabling developers to collaborate effectively through branching, merging, and code versioning strategies. Jenkins acts as the central continuous integration server, orchestrating the entire pipeline through a series of automated jobs that execute compilation, testing, static code analysis, and deployment tasks. SonarQube provides comprehensive code quality management by performing static program analysis, identifying bugs, vulnerabilities, and code smells before they reach production environments. Maven handles build automation, managing project dependencies and executing lifecycle goals that drive the CI/CD process, while Artifactory serves as the repository manager for storing build artifacts.

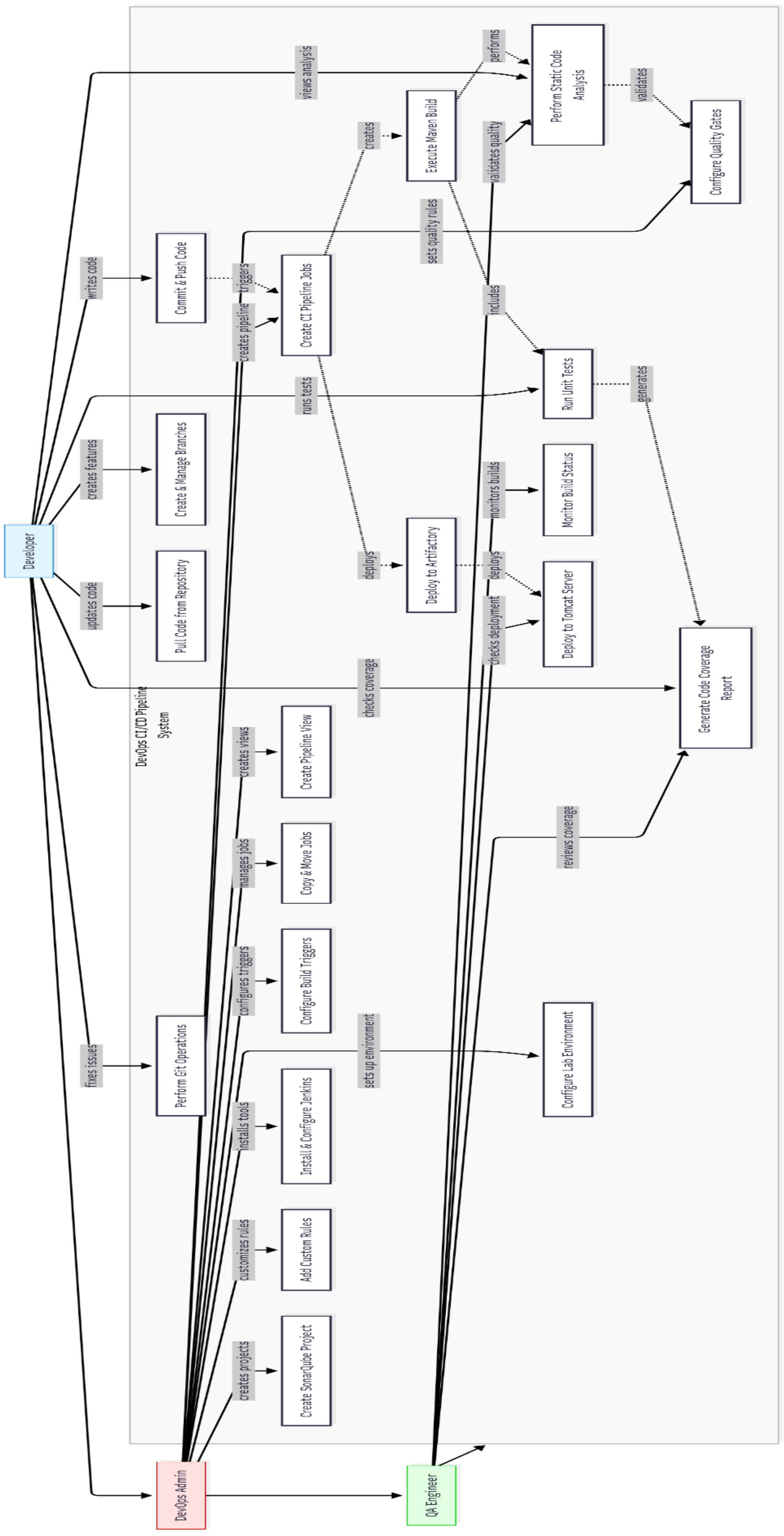
The pipeline implementation features a multi-stage architecture beginning with environment setup and source code checkout, progressing through compilation and unit testing with JUnit, followed by JaCoCo code coverage analysis to ensure adequate test coverage. Quality gates are strategically positioned at critical checkpoints, automatically halting builds that fail to meet predefined thresholds for code quality metrics, technical debt ratios, and test coverage percentages. This ensures that only code meeting organizational quality standards proceeds through the pipeline.

This project combines key elements of DevOps engineering, such as automated testing, secure build processes, quality assurance mechanisms, and deployment automation, to create a reliable and efficient software delivery solution. It serves as a practical demonstration of implementing industry best practices in continuous integration and continuous delivery, exploring the intersection of automation, quality control, and operational efficiency in real-world enterprise application scenarios.

DATA FLOW DIAGRAM



USE CASE DIAGRAM



SOFTWARE REQUIREMENTS

1. Platform Requirements

Operating System: Windows 8.1 or higher (64-bit) with support for environment variables configuration

Hardware Compatibility: Desktop and server environments with sufficient memory (minimum 4GB RAM recommended for running multiple tools simultaneously)

2. Development Environment Requirements

IDE: Eclipse IDE (Java EE perspective) for integrated development environment

Language: Java (JDK 1.8.0_141 or higher)

Plugins:

- EGit Plugin: For Git integration within Eclipse IDE enabling clone, commit, push, pull, fetch, and merge operations
- SonarLint Plugin: For on-the-fly static code analysis during coding providing immediate feedback on code quality issues

3. Version Control System

Git: Distributed source code management system for version control operations including branching, merging, and repository management

Repository Structure: Support for multiple branches including master, dev-team, and sprint-team branches

4. Build Automation Requirements

Maven: Apache Maven for build automation and dependency management

Configuration:

- M2_HOME environment variable pointing to Maven installation directory
- pom.xml configuration file with project dependencies and build lifecycle goals

Maven Plugins:

- maven-war-plugin (version 2.1.1): For WAR file generation

- maven-surefire-plugin: For executing unit tests
- sonar-maven-plugin (version 3.2): For SonarQube integration
- jacoco-maven-plugin (version 0.7.9): For code coverage analysis
- jmeter-maven-plugin (version 2.4.0): For performance testing

5. Continuous Integration Server

Jenkins: Jenkins CI server for pipeline orchestration and automation

Configuration:

- Jenkins URL: **http://localhost:8064/jenkins**
- JAVA_HOME and M2_HOME environment variables configured in Global Tool Configuration

Jenkins Plugins:

- Copy Artifact Plugin (version 1.45.4): For copying artifacts between jobs
- JaCoCo Plugin: For code coverage reporting
- Deploy to Container Plugin: For Tomcat deployment

6. Code Quality Management

SonarQube: Version 6.2 or higher for static c–ode analysis and quality gate management

Configuration:

- SonarQube URL: **http://localhost:9000**
- Default credentials: admin/admin
- Sonar-Runner (version 2.4) for command-line analysis execution

Quality Metrics:

- Code coverage analysis with JaCoCo
- Static code analysis for bugs, vulnerabilities, and code smells
- Technical debt ratio calculation

7. Testing Framework

JUnit: Unit testing framework for test case execution and validation

JaCoCo: Code coverage tool for generating coverage reports in HTML and XML formats

Coverage Thresholds:

- Minimum complexity coverage ratio: 0.10
- Configurable thresholds for instruction, branch, line, method, and class coverage

8. Artifact Repository Management

Artifactory: Repository manager for storing and managing build artifacts including WAR files

Configuration:

- Artifactory URL: **http://localhost:8081/artifactory**
- Default credentials: admin/Password1!
- Local Maven repository configuration for CalcDevSnapshot

9. Application Server

Apache Tomcat: Servlet container for deploying and running web applications

Configuration:

- Tomcat URL: **http://localhost:8080**
- Deployment credentials: tomcat/s3cret
- Support for WAR file deployment

10. Application Dependencies

Project Type: Simple web application using Maven framework

Core Dependencies:

- javax.servlet-api (version 3.1.0): For servlet functionality
- junit (version 4.11): For unit testing
- selenium-java (version 3.6.0): For automated testing with provided scope

PROGRAM CODE

1. Maven Build Configuration (pom.xml)

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

         http://maven.apache.org/maven-v4_0_0.xsd">


   <modelVersion>4.0.0</modelVersion>

   <groupId>ETA</groupId>

   <artifactId>Calculator</artifactId>

   <packaging>war</packaging>

   <version>0.0.1-SNAPSHOT</version>

   <name>calculator</name>

   <url>http://calculator</url>

   <dependencies>

      <!-- JUnit Testing Framework -->

      <dependency>

         <groupId>junit</groupId>

         <artifactId>junit</artifactId>

         <version>4.11</version>

      </dependency>

      <!-- Java Servlet API -->
```

```
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
    </dependency>
    <!-- Selenium for Automated Testing -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.6.0</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
<build>
    <finalName>calculator</finalName>
    <plugins>
        <!-- Maven WAR Plugin -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.1.1</version>
            <configuration>
                <archive>
                    <manifestEntries>
            <version>${project.version}</version>
          </manifestEntries>
        </archive>
      </configuration>
    </plugin>
    <!-- SonarQube Maven Plugin -->
    <plugin>
      <groupId>org.sonarsource.scanner.maven</groupId>
      <artifactId>sonar-maven-plugin</artifactId>
      <version>3.2</version>
    </plugin>
    <!-- JaCoCo Code Coverage Plugin -->
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.7.9</version>
      <executions>
        <execution>
          <id>default-prepare-agent</id>
          <goals>
            <goal>prepare-agent</goal>
          </goals>
        </execution>
        <execution>
```

```
   <id>default-report</id>
   <phase>prepare-package</phase>
   <goals>
      <goal>report</goal>
   </goals>
</execution>
<execution>
   <id>default-check</id>
   <goals>
      <goal>check</goal>
   </goals>
   <configuration>
      <rules>
         <rule>
            <element>BUNDLE</element>
            <limit>
               <counter>COMPLEXITY</counter>
               <value>COVEREDRATIO</value>
               <minimum>0.10</minimum>
            </limit>
         </rule>
      </rules>
   </configuration>
</execution>
```

```
    </executions>
</plugin>
<!-- Maven Surefire Plugin for Unit Testing -->
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
        <includes>
            <include>Calculator.java</include>
        </includes>
    </configuration>
</plugin>
<!-- JMeter Maven Plugin for Performance Testing -->
<plugin>
    <groupId>com.lazerycode.jmeter</groupId>
    <artifactId>jmeter-maven-plugin</artifactId>
    <version>2.4.0</version>
    <executions>
        <execution>
            <id>jmeter-tests</id>
            <phase>test</phase>
            <goals>
                <goal>jmeter</goal>
            </goals>
```

```
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
<!-- Distribution Management for Artifactory -->
<distributionManagement>
    <repository>
        <id>artifactory</id>
        <name>CalcDevSnapshot</name>
        <url>http://localhost:8081/artifactory/CalcDevSnapshot</url>
    </repository>
</distributionManagement>

<!-- Build Profiles -->
<profiles>
    <profile>
        <id>ut</id>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-surefire-plugin</artifactId>
                    <configuration>
                <includes>
                    <include>Calculatorut.java</include>
                </includes>
            </configuration>
        </plugin>
    </plugins>
  </build>
</profile>
<profile>
  <id>it</id>
  <build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <configuration>
                <includes>
                    <include>CalculatorIT.java</include>
                </includes>
            </configuration>
        </plugin>
    </plugins>
  </build>
</profile>
```

```
    </profiles>

</project>
```

2. SonarQube Configuration (sonar-runner.properties)

```
# Configure general information about the environment

# Default SonarQube server

sonar.host.url=http://localhost:9000

# Default source code encoding

sonar.sourceEncoding=UTF-8

# Security when sonar.forceAuthentication is set to true

sonar.login=admin

sonar.password=admin

# Project Configuration

sonar.projectKey=calculator

sonar.projectName=calculator

sonar.projectVersion=0.0.1-SNAPSHOT

sonar.sources=src/main/java

sonar.binaries=target/classes

sonar.log.level=WARN

sonar.language=java
```

3. Jenkins Pipeline Jobs Configuration

Job 1: Setup (Source Code Checkout):

```
// Git Repository Configuration

Repository URL: https://github.com/Ajay-vetri-NJ/DevOpsInfoSys.git

Branch Specifier: */dev-team1
```

// Build Configuration

Goals: clean

Job 2: Compile:

// Copy Artifacts from Setup Job

Project name: Setup

Which build: Latest successful build

Target directory: .

// Maven Build Goals

Goals: compile

Job 3: Static Analysis:

// Copy Artifacts from Compile Job

Project name: Compile

Which build: Latest successful build

Target directory: .

// Maven Build Goals

Goals: sonar:sonar

// Post-build Actions

Trigger projects: UnitTest

Job 4: Unit Test:

// Copy Artifacts from StaticAnalysis Job

Project name: StaticAnalysis

Which build: Latest successful build

Target directory: .

// Maven Build Goals

Goals: test -Put

// Post-build Actions

Publish JUnit test result report

Test report XMLs: **/surefire-reports/*.xml

Trigger projects: CodeCoverage

Job 5: Code Coverage:

// Copy Artifacts from UnitTest Job

Project name: UnitTest

Which build: Latest successful build

Target directory: .

// Maven Build Goals

Goals: jacoco:report

// Post-build Actions

Record JaCoCo coverage report

Path to exec files: **/jacoco.exec

Path to class directories: **/classes

Path to source directories: **/src/main/java

Trigger projects: War

Job 6: War

// Copy Artifacts from CodeCoverage Job

Project name: CodeCoverage

Which build: Latest successful build

Target directory: .

// Maven Build Goals

Goals: war:war

// Post-build Actions

Archive the artifacts: **/*.war

Trigger projects: ToArtifactory

Job 7: ToArtifactory:

// Copy Artifacts from War Job

Project name: War

Which build: Latest successful build

Target directory: .

// Maven Build Goals

Goals: deploy

// Artifactory Configuration

Artifactory URL: http://localhost:8081/artifactory

Repository Key: CalcDevSnapshot

Credentials: admin/Password1!

// Post-build Actions

Trigger projects: SmokeTest

Job 8: Smoke Test (Deploy to Tomcat)

// Copy Artifacts from War Job

Project name: War

Which build: Latest successful build

Target directory: .

// Deploy to Container

WAR/EAR files: **/*.war

Context path: calculator

Tomcat URL: http://localhost:8080

Credentials: tomcat/s3cret

4. Quality Gate Configuration (SonarQube)

```
<!-- Quality Gate: CALQG -->

<qualityGate name="CALQG">

  <condition>

    <metric>Critical Issues</metric>

    <operator>Greater than</operator>

    <value>0</value>

  </condition>

  <condition>

    <metric>Duplicated lines</metric>

    <operator>Greater than</operator>

    <value>10</value>

  </condition>

  <condition>

    <metric>Major issues</metric>

    <operator>Greater than</operator>

    <value>5</value>

  </condition>

  <condition>

    <metric>Technical Debt Ratio</metric>

    <operator>Greater than</operator>

    <value>5</value>
    </condition>

    <condition>

        <metric>Complexity</metric>

        <operator>Greater than</operator>

        <value>7</value>

    </condition>

</qualityGate>
```

5. Maven Build Execution Commands

```
# Complete CI/CD Pipeline Execution

mvn clean compile test jacoco:report sonar:sonar war:war deploy

# Individual Goal Execution

mvn clean          # Clean previous builds

mvn compile        # Compile source code

mvn test           # Run unit tests

mvn jacoco:report  # Generate code coverage report

mvn sonar:sonar    # Perform static code analysis

mvn deploy         # Deploy to Artifactory
```

OUTPUT

```
[INFO] BUILD SUCCESS

[INFO] Total time: 48.728 s

[INFO] Finished at: 30-10-2025 10:27:32 AM

[INFO] Final Memory: 40M/351M
```

FUTURE ENHANCEMENT

1. Container Orchestration with Docker and Kubernetes:

Integrate Docker containerization to package applications and their dependencies into lightweight, portable containers that can run consistently across different environments. This would require creating Dockerfile configurations for the application, Jenkins, SonarQube, and other tools in the pipeline. Additionally, implement Kubernetes for container orchestration to enable automated deployment, scaling, and management of containerized applications across clusters. Use Helm charts for managing Kubernetes deployments and implement rolling updates with zero-downtime deployment strategies.

2. Cloud-Native CI/CD Pipeline:

Migrate the entire CI/CD infrastructure to cloud platforms such as AWS, Azure, or Google Cloud Platform to leverage cloud-native services. Replace on-premises Jenkins with managed services like AWS CodePipeline, Azure DevOps, or Google Cloud Build. Implement Infrastructure as Code (IaC) using Terraform or AWS CloudFormation to provision and manage cloud resources programmatically. This would enhance scalability, reduce infrastructure maintenance overhead, and provide better disaster recovery capabilities.

3. Advanced Security Integration (DevSecOps):

Implement comprehensive security scanning throughout the CI/CD pipeline using tools like OWASP Dependency Check for vulnerability analysis of third-party libraries, Snyk for open-source security monitoring, and Trivy for container image scanning. Add Static Application Security Testing (SAST) tools beyond SonarQube, such as Checkmarx or Fortify, and incorporate Dynamic Application Security Testing (DAST) tools like OWASP ZAP for runtime security testing. Implement secret management using HashiCorp Vault or AWS Secrets Manager to securely handle sensitive credentials and API keys.

4. Enhanced Monitoring and Observability:

Integrate comprehensive monitoring solutions using the ELK Stack (Elasticsearch, Logstash, Kibana) or Splunk for centralized log aggregation and analysis. Implement Prometheus and Grafana for real-time metrics collection, visualization, and alerting on application performance, build success rates, deployment frequency, and system health. Add distributed tracing capabilities using Jaeger or Zipkin to monitor microservices communication patterns. Configure automated alerting mechanisms through PagerDuty, Slack, or Microsoft Teams to notify teams of build failures, quality gate violations, or deployment issues.

5. Automated Performance and Load Testing:

Expand the JMeter integration mentioned in the pom.xml configuration to include comprehensive automated performance testing as part of the pipeline. Implement load testing scenarios that simulate production-like traffic patterns, stress testing to identify breaking points, and endurance testing for long-running stability verification. Use tools like Gatling or Apache JMeter with Jenkins Performance Plugin to generate detailed performance reports, trend analysis, and automatic build failure on performance regression. Integrate APM (Application Performance Monitoring) tools like New Relic, Dynatrace, or AppDynamics for deep performance insights.

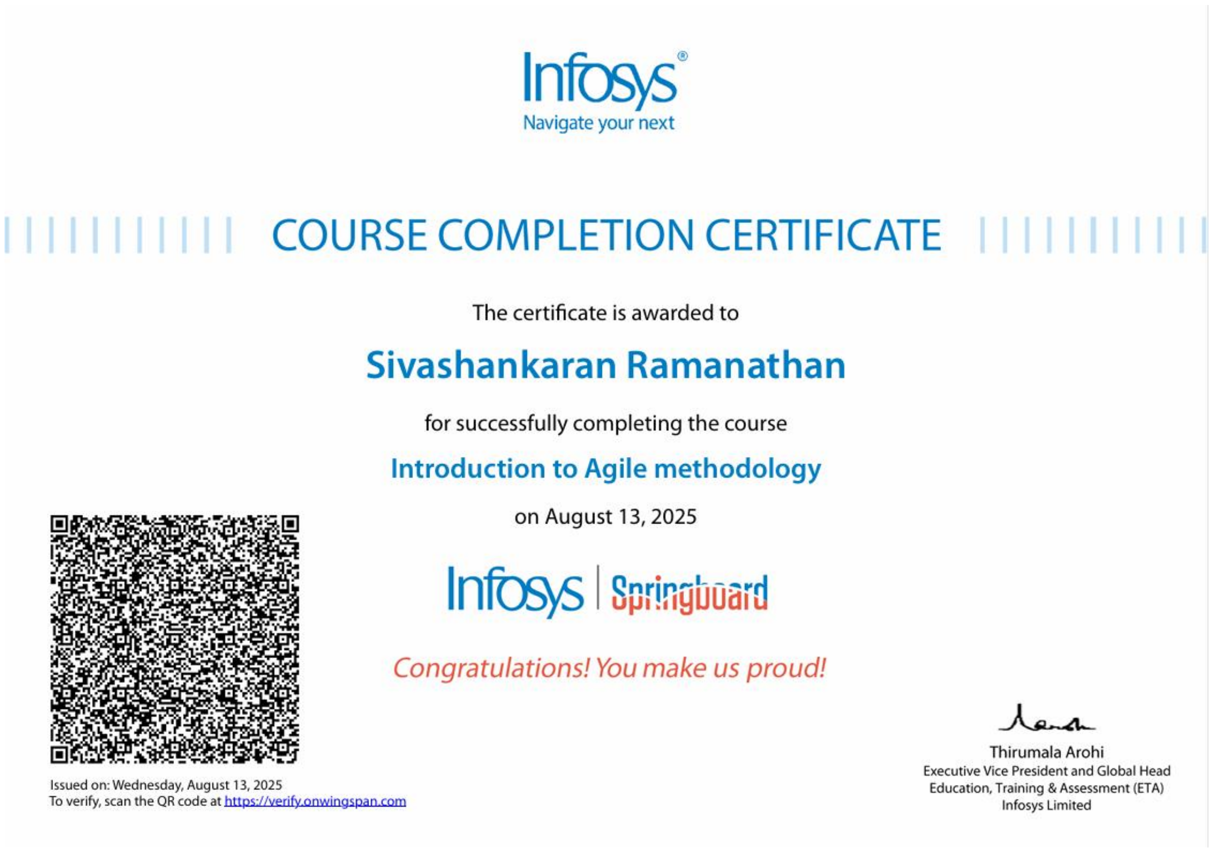
CONCLUSION

This project presents a comprehensive DevOps Continuous Integration and Continuous Delivery (CI/CD) pipeline that automates the entire software development lifecycle using leading open-source tools such as Jenkins, Git, Maven, SonarQube, JaCoCo, and Artifactory. It integrates key DevOps processes including source code management, automated builds, static code analysis, unit testing, code coverage, artifact management, and deployment to Tomcat servers through a well-structured eight-stage pipeline.

The pipeline enforces quality assurance through configurable quality gates, build breakers, and code analysis metrics, ensuring that only reliable and maintainable code progresses through each stage. Automated testing and deployment minimize manual intervention, reduce risks, and accelerate software delivery, while real-time dashboards in SonarQube provide insights into code quality, vulnerabilities, and technical debt.

By implementing these practices, the project bridges the gap between development and operations, fostering collaboration, consistency, and continuous improvement. It also builds practical expertise in configuring integrations, managing Jenkins pipelines, and establishing robust CI/CD workflows. This implementation lays a strong foundation for future enhancements such as containerization with Docker and Kubernetes, DevSecOps integration, cloud-native pipeline deployment, and AI-driven optimization to further improve scalability, security, and efficiency.

CERTIFICATES



Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Sivashankaran Ramanathan

for successfully completing the course

Introduction to Agile methodology

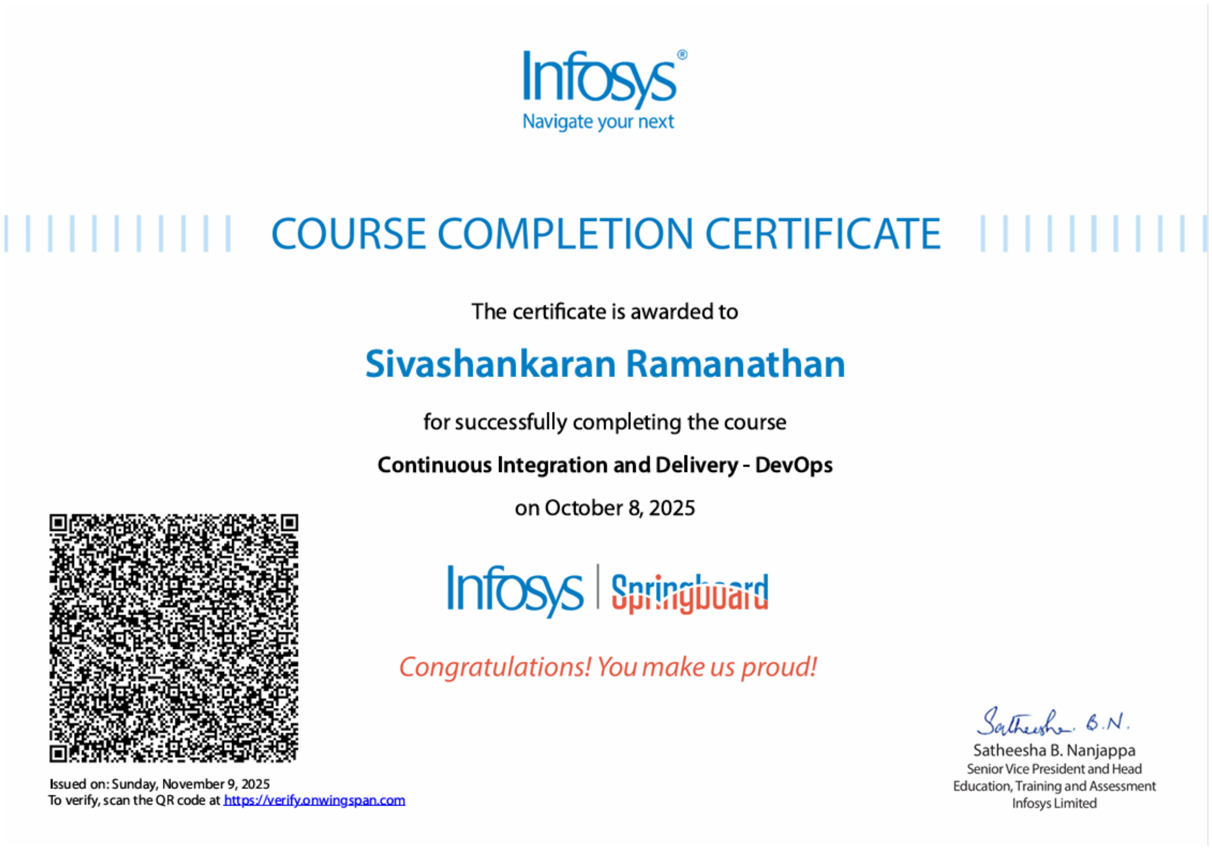
on August 13, 2025

Infosys | Springboard

Congratulations! You make us proud!

Issued on: Wednesday, August 13, 2025
To verify, scan the QR code at https://verify.onwingspan.com

[illegible]
Thirumala Arohi
Executive Vice President and Global Head
Education, Training & Assessment (ETA)
Infosys Limited



Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Sivashankaran Ramanathan

for successfully completing the course

Continuous Integration and Delivery - DevOps

on October 8, 2025

Infosys | Springboard

Congratulations! You make us proud!

Issued on: Sunday, November 9, 2025
To verify, scan the QR code at https://verify.onwingspan.com

Satheesha. B.N.
Satheesha B. Nanjappa
Senior Vice President and Head
Education, Training and Assessment
Infosys Limited