**CI CD Pipeline**

**Project Report**

Industry Internship (ICS871)

Degree

**BACHELOR OF TECHNOLOGY (CTIS)**

|  |  |
| --- | --- |
| PROJECT GUIDE:  **Mr. Ajay Rastogi**  **Mr. Divyanshu Saxena** | SUBMITTED BY:  **Akshat Sharma (TCA2150001)**  **Prashant Chauhan (TCA2150003)** |

Jan-June, 2025



**FACULTY OF ENGINEERING & COMPUTING SCIENCES**

**TEERTHANKER MAHAVEER UNIVERSITY, MORADABAD**

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to all those who supported me throughout the development of this project.

First and foremost, I extend my heartfelt thanks to Teerthanker Mahaveer University for providing me with this opportunity to work on a project that aligns with current industry practices in DevOps and CI/CD automation.

I am deeply thankful to my project guides , for their continuous guidance, feedback, and motivation during the course of this project.

Special thanks to our faculty members and lab in-charge for ensuring the availability of the required infrastructure and resources in the CCSIT Lab.

*<Akshat Sharma>*

*<Prashant Chauhan>*

**Place: Moradabad**

**Date:**

**DECLARATION**

We hereby declare that this Project Report titled CI CD Pipeline submitted by us and approved by our project guide, the College of Computing Sciences and Information Technology (CCSIT), Teerthanker Mahaveer University, Moradabad, is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

|  |  |  |
| --- | --- | --- |
| **Project Group :** | Project Group Name/Id | |
| **Student Name:** | Akshat Sharma | Signature |
| **Student Name:** | Prashant Chauhan | Signature |
|  |  |  |
| **Project Guide: (External)** | Name | Signature |
| **Project Guide: (Internal)** | Mr. Ajay Rastogi  Mr. Divyanshu Saxena | Signature |

**Brief About the Company**

**Company Overview:**

CodeApto India Private Limited is a Bangalore-based technology consulting and IT services company that was incorporated in February 2023. With a vision to become a leading solutions integrator, CodeApto focuses on helping enterprises achieve digital excellence through innovative and scalable technology solutions. The company aims to bridge the gap between business needs and emerging technologies by providing tailored, end-to-end services.

**Core Services and Offerings:**

CodeApto offers a comprehensive portfolio of services designed to support digital transformation and modern business needs:

1. **Enterprise Application Services:**

CodeApto helps organizations implement, optimize, and manage enterprise-grade applications like ERP systems (SAP, Oracle, etc.). The goal is to enhance operational efficiency, improve risk management, and align IT infrastructure with business strategy.

1. **Modern Infrastructure Solutions:**

The company assists in modernizing legacy IT systems, migrating to cloud environments, and improving IT infrastructure scalability, security, and performance. Services include cloud strategy, virtualization, and hybrid IT solutions.

1. **Data & AI:**

CodeApto empowers clients to leverage the power of data through advanced analytics, artificial intelligence, and machine learning. Their services enable real-time decision-making, customer insights, automation, and predictive intelligence.

1. **Technology Consulting:**

From digital strategy to implementation, CodeApto provides end-to-end consulting services that help businesses stay competitive. Their consulting approach focuses on improving technology alignment, enhancing business agility, and enabling data-driven decision-making.

1. **Managed Services:**

CodeApto’s managed services provide organizations with continuous IT support, proactive monitoring, vendor management, and scalable resources. This ensures operational continuity, reduced downtime, and better resource allocation.

**Industry Focus:**

CodeApto delivers industry-specific solutions tailored to the unique needs of various sectors, including:

**Manufacturing**

**Healthcare & Life Sciences**

**Retail & Consumer Packaged Goods (CPG)**

**Utilities & Energy**

**Logistics & Supply Chain**

**Education**

**Public Sector & Government**

Each industry solution is designed with compliance, scalability, and customer-centricity in mind.

**Mission & Vision:**

**Mission:** To enable organizations to innovate, scale, and lead in their respective markets by leveraging modern technology and strategic insights.

**Vision:** To be a global leader in technology integration and digital transformation by delivering excellence and measurable business outcomes.

Official Website: [www.codeapto.com](http://www.codeapto.com)

Table of Contents

**1 PROJECT TITLE…………………………………………………………………………………………..……………………………7**

**2 PROBLEM STATEMENT…………………………………………………………………………………………..….……………7**

**3 PROJECT DESCRIPTION…………………………………………………………………………………………..…….…………7**

3.1 SCOPE OF THE WORK**…………………………………………………………………………………………..……..**8

3.2 PROJECT MODULES**……………………………………………………………………………..……..………………**8

3.3 CONTEXT DIAGRAM (HIGH LEVEL) **………………………………………………………………………………**9

**4 IMPLEMENTATION METHODOLOGY………………………………………………………………………………………9**

**5 TECHNOLOGIES TO BE USED……………………………………………………………………………….…………..……11**

5.1 SOFTWARE PLATFORM………………………………………………………………………………..………………11

5.2 HARDWARE PLATFORM**……………………………………………………..……………………………………..**11

5.3 TOOLS, IF ANY**…………………………………………………………………………………………..………………**12

**6 ADVANTAGES OF THIS PROJECT…………………………………………………………………………………… 13**

**7 ASSUMPTIONS, IF ANY…………………………………………………………………………………………………………13**

**8 FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT………………………………………….14**

**9 PROJECT REPOSITORY LOCATION………………………………………………………………………………….………15**

**10 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS…………………………………………………………………15**

**11 CONCLUSION……………………………………………………………………………………………………………………….……….16**

**12 REFERENCES……………………………………………………………………………….………………………………………………..17**

**Appendix**

**A: Data Flow Diagram (DFD)**

**B: Entity Relationship Diagram (ERD)**

**C: Use Case Diagram (UCD)**

**D: Data Dictionary (DD)**

**E: Screen Shots**

# Project Title

CI CD Pipeline

# Problem Statement

In the modern era of software development, delivering applications faster, more securely, and more reliably is critical. Traditional software delivery processes are often manual, time-consuming, and error-prone.

Without automation, teams face challenges like inconsistent deployments, lack of testing, and delayed feedback cycles, which impact product quality and user satisfaction.

As cloud-native architectures gain prominence, the need for cloud-based CI/CD solutions has become vital. AWS offers managed services like CodeCommit, CodeBuild, CodeDeploy, and CodePipeline, enabling organizations to automate software delivery securely at scale.

This project aims to build an AWS-powered CI/CD pipeline to automate code integration, testing, and deployment processes, reducing manual overhead and ensuring rapid, reliable, and repeatable application releases.

# Project Description

The goal of this project is to design and implement a cloud-native Continuous Integration and Continuous Deployment (CI/CD) pipeline using Amazon Web Services (AWS).

The pipeline will automate the entire software delivery lifecycle — from source code management to building, testing, deploying, and monitoring applications.

The project utilizes AWS services including:

AWS CodeCommit for hosting source code repositories,

AWS CodeBuild for automating build and test processes,

AWS CodeDeploy for automated deployment to compute resources (EC2/Elastic Beanstalk),

AWS CodePipeline for orchestrating the entire workflow.

Additional services like Amazon S3, Amazon CloudWatch, and Elastic Container Service (ECS) may be used to enhance functionality.

The pipeline will demonstrate how cloud-native DevOps practices can speed up development cycles, enhance software quality, and reduce operational risks.

## Scope of the Work

**In-Scope**:

* Setting up a Git-based repository using AWS CodeCommit
* Automating the build and testing phases using AWS CodeBuild
* Containerizing the application using Docker
* Storing container images on Amazon Elastic Container Registry (ECR)
* Deploying applications using AWS CodeDeploy and Elastic Beanstalk (or ECS)
* Implementing a complete CI/CD pipeline using AWS CodePipeline
* Setting up monitoring and alerts using Amazon CloudWatch

**Out of Scope:**

* Application development or writing complex business logic
* Multi-region deployment or disaster recovery implementation
* Integration with third-party DevOps tools outside AWS ecosystem

## Project Modules

1. **Source Code Management Module**

* AWS CodeCommit repository for version control
* Code push triggers the pipeline automatically

1. **Continuous Integration Module**

* AWS CodeBuild automatically pulls the latest code
* Builds the code, runs unit tests, and generates artifacts

1. **Containerization Module**

* Docker builds container images for the application
* Images are pushed to AWS Elastic Container Registry (ECR)

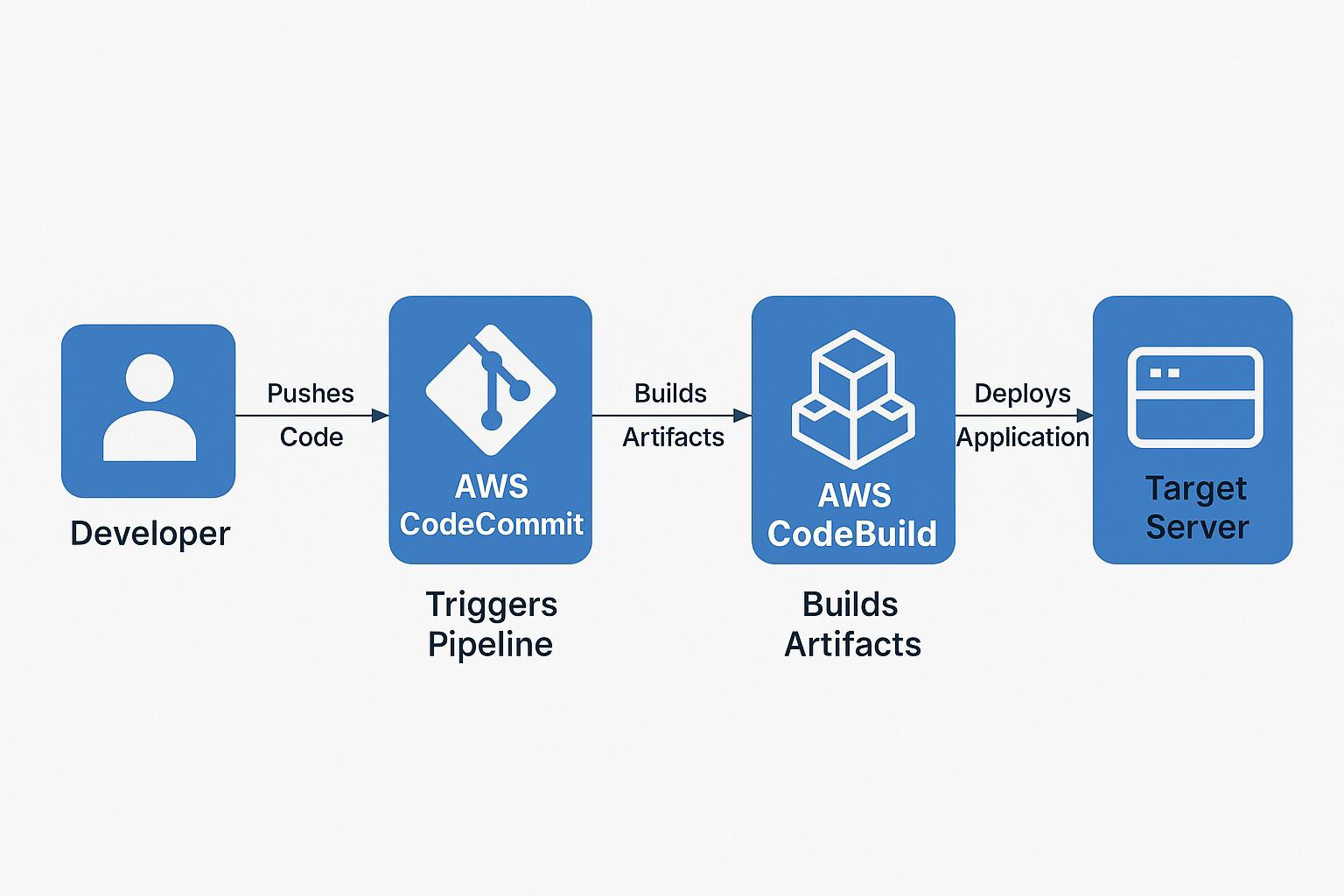
1. **Continuous Deployment Module**

* AWS CodeDeploy or Elastic Beanstalk deploys updated builds
* Auto-scaling and blue/green deployment strategies available

1. **Monitoring and Notification Module**

* Amazon CloudWatch monitors application health and performance
* AWS SNS sends notifications about build/deployment status

## Context Diagram (High Level)

**

# Implementation Methodology

The implementation of the CI/CD pipeline using AWS services follows a structured DevOps approach that automates the process of building, testing, and deploying applications. The methodology is divided into key stages, each leveraging managed AWS services to improve scalability, security, and automation efficiency.

1. **Environment Setup:**

AWS Account Setup: Configure IAM roles, security policies, and billing alerts.

Source Control: Create a repository in AWS CodeCommit to host the source code.

Build Environment: Create a build project in AWS CodeBuild with appropriate buildspec.yml configuration.

Deployment Platform: Choose a target deployment platform (e.g., EC2, Elastic Beanstalk, or ECS).

Pipeline Orchestration: Use AWS CodePipeline to define the end-to-end workflow.

1. **Source Code Management:**

The developer commits code to the CodeCommit repository.

Each code push automatically triggers the pipeline through CloudWatch Events.

1. **Continuous Integration Stage:**

CodePipeline detects changes in the CodeCommit repository.

CodeBuild compiles the application, runs unit tests, and builds the artifacts.

If build and tests pass, the artifact is stored in S3 or passed directly to the deployment stage.

1. **Build and Artifact Management:**

Buildspec.yml defines how the build artifacts are structured and packaged.

Artifacts can be stored in Amazon S3 or integrated with Amazon ECR for Docker-based applications.

1. **Continuous Deployment Stage:**

After a successful build, CodeDeploy is triggered.

It automatically deploys the application to EC2 instances, an Elastic Beanstalk environment, or a containerized ECS service.

Supports rolling updates, blue/green deployment, and rollback on failure.

1. **Monitoring and Notification:**

Amazon CloudWatch monitors the build and deployment processes.

Logs and metrics are stored and visualized via CloudWatch Dashboards.

Amazon SNS sends notifications to developers about build success/failure, deployment status, and alerts.

1. **Defect Tracking and Maintenance:**

Any failed builds or deployments are logged via CloudWatch.

Root cause analysis is performed using build logs and CloudWatch Insights.

Regular updates to IAM policies, build configurations, and environment parameters are performed for stability and security.

# Technologies to be used

## Software Platform

1. **Front-end**

Not applicable — the project focuses on infrastructure automation, not UI development.

1. **Back-end**

A lightweight sample web application (e.g., Node.js, Python Flask, or Java Spring Boot) is used to demonstrate the pipeline.

## Hardware Platform

The project is hosted on the cloud, but local setup was also used for development/testing.

**Cloud Platform:** Amazon Web Services (AWS)

**Operating System:** Amazon Linux 2 / Ubuntu 20.04 LTS

**Instance Type:** AWS EC2 t2.medium or t3.medium for build/deployment servers

**Local System for Development:**

**OS:** Ubuntu/Windows

**Processor:** Intel i5 or higher

**RAM:** Minimum 8 GB

**Storage:** At least 100 GB free

## Tools, if any

|  |  |
| --- | --- |
| **Tool/Service** | **Description** |
| AWS CodeCommit | Git-based source control system for hosting code repositories |
| AWS CodeBuild | Fully managed build service that compiles source code and runs tests |
| AWS CodeDeploy | Automates software deployments to EC2, Lambda, or on-premise servers |
| AWS CodePipeline | Visual workflow service that automates CI/CD pipeline orchestration |
| Amazon EC2 | Virtual servers for application deployment |
| Amazon ECR | Container image registry integrated with AWS |
| Amazon S3 | Object storage for storing artifacts and logs |
| Amazon CloudWatch | Logging and monitoring service for AWS resources and custom metrics |
| Amazon SNS | Notification system to send pipeline status updates to stakeholders |
| IAM (Identity & Access Management) | Secures access control for all AWS services used |
| Docker | Used locally for containerizing the sample application before pushing to ECR |

# Advantages of this Project

Implementing a CI/CD pipeline using AWS offers several key benefits:

1. Fully Managed Services:

AWS CodePipeline, CodeBuild, and CodeDeploy are fully managed, reducing the need for manual server management.

2. Faster Release Cycles:

Automated integration and deployment processes lead to quicker application releases, reducing time-to-market.

3. Improved Software Quality:

Continuous testing and validation catch bugs early in the development cycle, enhancing software reliability.

4. High Scalability and Availability:

AWS services scale automatically to handle varying loads, ensuring performance and availability.

5. Centralized Monitoring and Alerts:

CloudWatch enables real-time monitoring and automatic alerting, improving response to issues.

6. Security and Compliance:

Fine-grained IAM roles and encryption policies provide secure access to pipeline resources and data.

7. Pay-as-You-Go Model:

Costs are incurred only for the resources used, making it affordable for both small and large teams.

8. DevOps Best Practices:

The project enforces DevOps principles such as automation, continuous feedback, and version-controlled infrastructure.

# Assumptions, if any

The application is lightweight and doesn’t require complex database or stateful configurations.

AWS account has necessary permissions (IAM roles and policies) to access services like CodeBuild and CodeDeploy.

The student has basic familiarity with Git, Docker, and cloud concepts.

Internet access is available to fetch build dependencies and push to repositories.

The project is executed in a controlled academic environment without cost restrictions.

If these do not apply, this section can be marked as:

***Assumptions: NONE***

# Future Scope and further enhancement of the Project

The AWS-based CI/CD pipeline can be enhanced in several ways:

* 1. DevSecOps Integration:

Integrate tools like AWS Inspector, SonarQube, or Trivy to perform automated security scans during the build phase.

* 1. Blue/Green or Canary Deployments:

Use CodeDeploy or ECS features to reduce downtime and safely test new releases.

* 1. Infrastructure as Code (IaC):

Use AWS CloudFormation or Terraform to manage infrastructure configuration declaratively.

* 1. Multi-Environment Support:

Extend the pipeline to support multiple environments (dev, staging, production) with approval workflows.

* 1. Lambda-Based Serverless Deployment:

Adapt the pipeline to deploy serverless applications for better scalability and reduced cost.

* 1. Slack/MS Teams Integration:

Add real-time notifications for pipeline events using Amazon SNS integrations with messaging apps.

* 1. Centralized Logging:

Use Amazon OpenSearch (formerly ELK) or CloudWatch Logs Insights for advanced log analysis.

By implementing these enhancements, the pipeline can evolve into a robust, enterprise-grade DevOps system.

# Project Repository Location

The latest source code and all relevant project documents, including the synopsis, requirement specifications, test cases, and the final report, are securely stored in the university lab infrastructure for easy access and version control. The repository location follows the CCSIT lab configuration and has been verified by the concerned faculty members.

| **S#** | **Project Artifacts (softcopy)** | **Location** (Mention Lab-ID, Server ID, Folder Name etc.) | **Verified by Project Guide** | **Verified by Lab In-Charge** |
| --- | --- | --- | --- | --- |
|  | Project Synopsis Report (Final Version) | CCSIT- DevOps-Lab, Server:: 192.168.10.24,Folder:/CI-CD-Pipeline. |  |  |
|  | Project Progress updates | CCSIT- DevOps-Lab, Server:: 192.168.10.24,Folder:/CI-CD-Pipeline. |  |  |
|  | Project Requirement specifications | CCSIT- DevOps-Lab, Server:: 192.168.10.24,Folder:/CI-CD- Pipeline. |  |  |
|  | Project Report (Final Version) | CCSIT- DevOps-Lab, Server:: 192.168.10.24,Folder:/CI-CD- Pipeline . |  |  |
|  | Test Repository | CCSIT- DevOps-Lab, Server:: 192.168.10.24,Folder:/CI-CD- Pipeline. |  |  |
|  | Any other document, give details | CCSIT- DevOps-Lab, Server:: 192.168.10.24,Folder:/CI-CD- Pipeline. |  |  |

# Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| CI/CD | Continuous Integration /Continuous Deployment a DevOps practice where code changes are automatically built, tested, and deployed. |
| DevOps | A combination of development and operations practices aimed at automating and improving the software delivery process. |
| IAM | Identity and Access Management in AWS |
| CodeCommit | AWS-hosted Git repository |
| CodeBuild | AWS-managed build and test service |
| CodeDeploy | AWS deployment automation service |
| CodePipeline | AWS service to automate release workflows |
| EC2 | Elastic Compute Cloud (virtual server in AWS) |
| ECR | Elastic Container Registry |
| S3 | Simple Storage Service (object storage) |
| CloudWatch | Monitoring and alerting service |
| SNS | Simple Notification Service for email, SMS, or application alerts |

# Conclusion

This project successfully demonstrates the design and implementation of a cloud-native CI/CD pipeline using Amazon Web Services. The solution leverages AWS-managed tools such as CodeCommit, CodeBuild, CodeDeploy, and CodePipeline to automate the complete software delivery lifecycle—from code integration to deployment and monitoring.

The approach not only enhances release velocity but also ensures high reliability, scalability, and real-time feedback throughout the process. It mirrors real-world enterprise practices and supports modern DevOps principles, including continuous testing, feedback, and secure deployments.

In conclusion, adopting a cloud-based CI/CD model empowers development teams to deliver software faster, respond quickly to changes, and maintain operational stability across environments. This project provides a strong foundation for future enhancements such as DevSecOps, blue/green deployments, serverless integration, and IaC automation.

# References

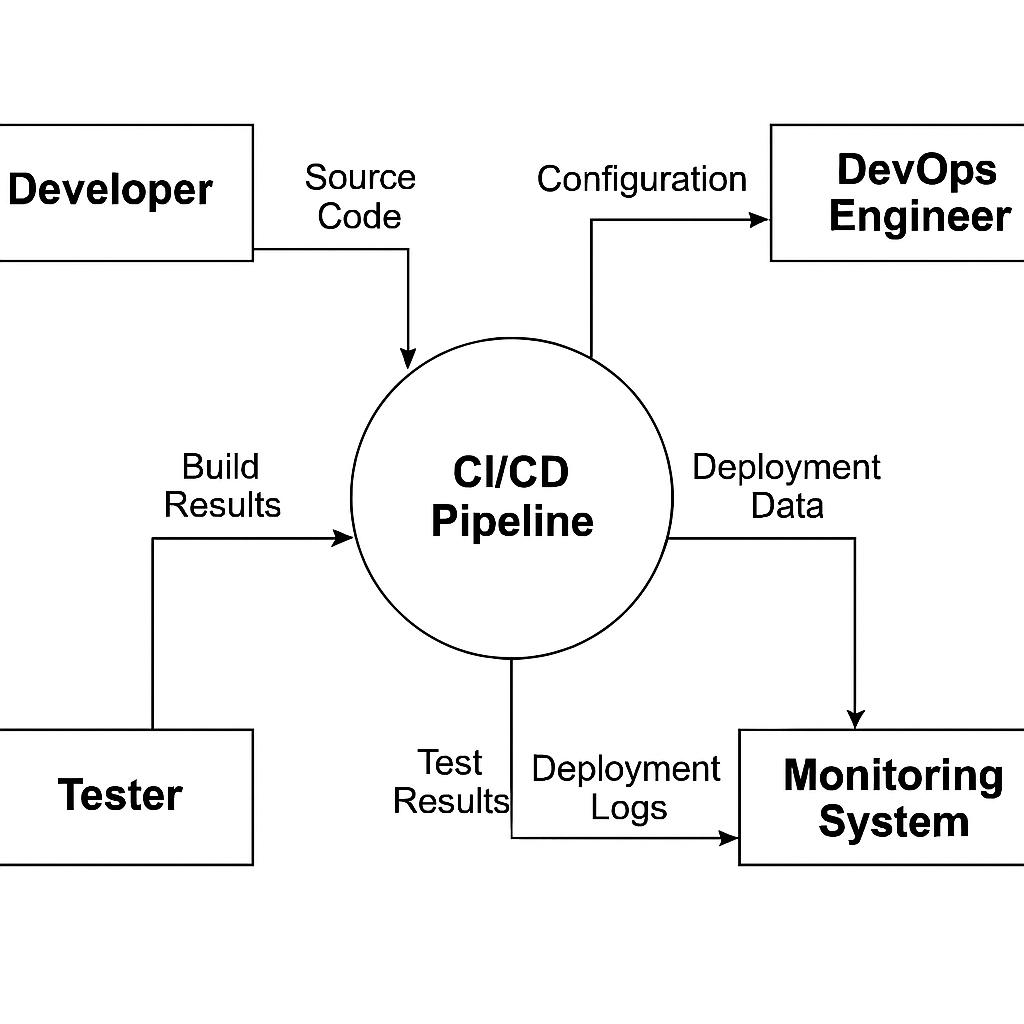
This section lists all the sources referred to during the planning, development, and documentation of the CI/CD pipeline project. The resources include official documentation, open-source tools, and technical articles used to understand and implement various components of the project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S#** | **Reference Details** | **Owner** | **Version** | **Date** |
|  | AWS Code Pipeline Documentation | Amazon Web Services | Latest | 2025 |
|  | AWS Code Commit Documentation | Amazon Web Services | Latest | 2025 |
|  | AWS Code Build Documentation | Amazon Web Services | Latest | 2025 |
|  | AWS Code deploy Documentation | Amazon Web Services | Latest | 2025 |
|  | AWS Cloud Watch Documentation | Amazon Web Services | Latest | 2025 |
|  | Docker Documentation | Docker Inc. | 24.0 | 2025 |
|  | Git Documentation | Git SCM | 2.42 | 2025 |
|  | AWS Training & Certification (Free Tier Labs) | Amazon Web Services | N/A | 2025 |

**Annexure A**

**Data Flow Diagram (DFD)**

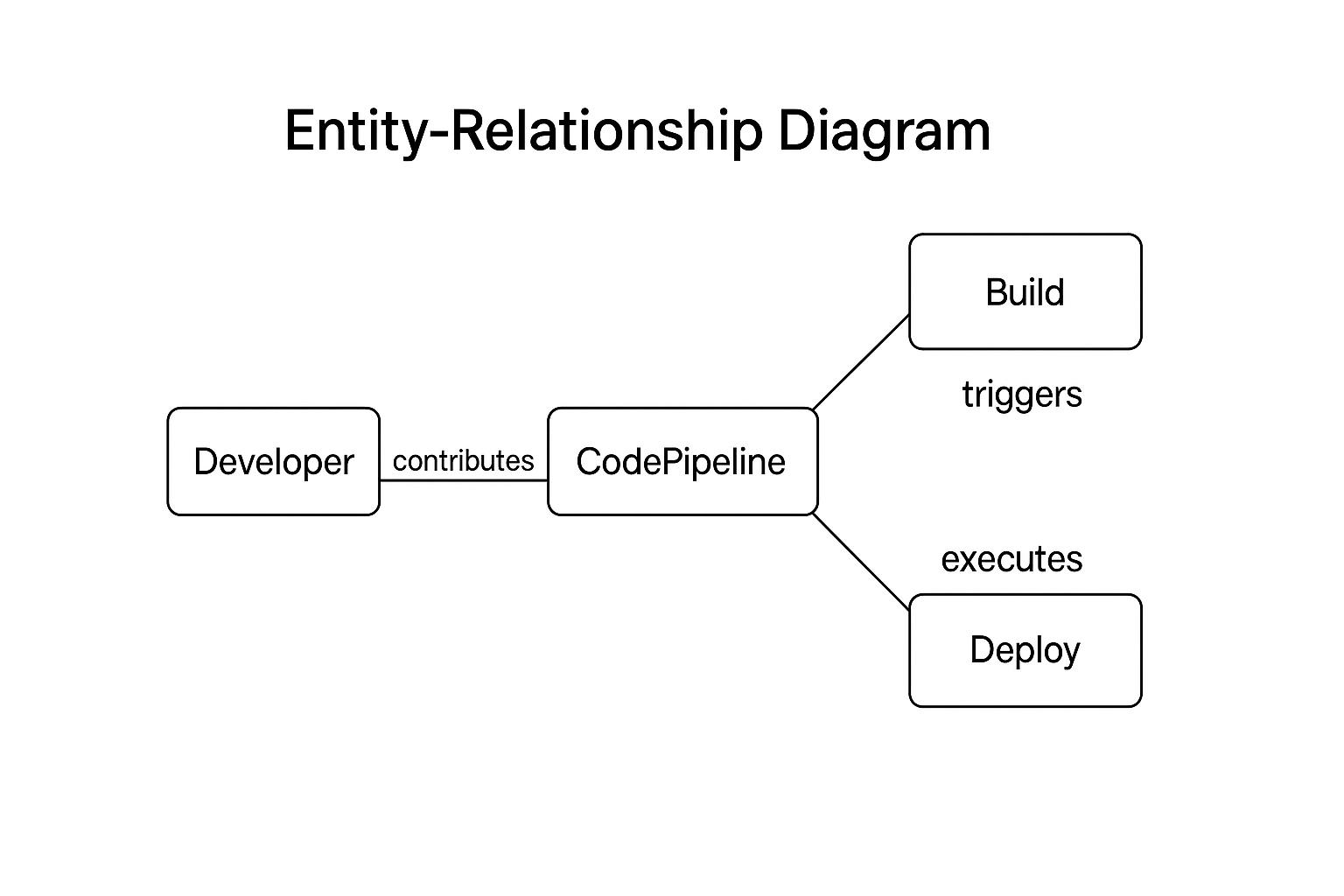
**(Mandatory)**



**Annexure B**

**Entity-Relationship Diagram (ERD)**

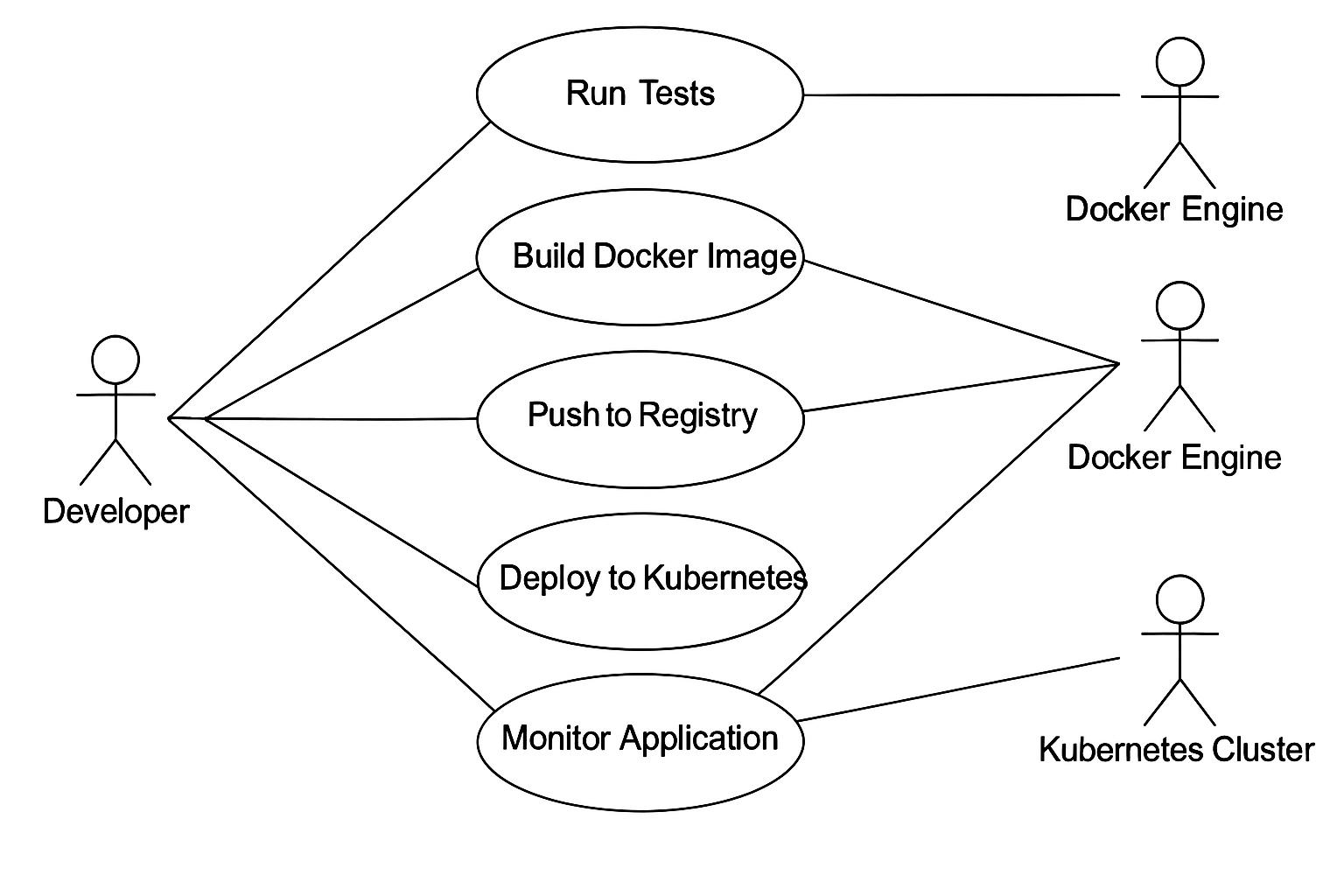
**(Mandatory)**

****

**Annexure C**

**Use-Case Diagram (UCD)**

**(Optional)**

****

The Use Case Diagram (UCD) for the CI/CD pipeline project represents the interaction between the main actors and the key actions (use cases) in the system.

The Developer is the primary actor who initiates the workflow by pushing code, building Docker images, running tests, pushing images to a registry, deploying the application, and monitoring the deployed application.

The Docker Engine acts as a secondary actor involved in building Docker images and pushing them to the Docker registry.

The Kubernetes Cluster is responsible for deploying the Dockerized applications and enabling their monitoring after deployment.

The major use cases identified are:

Run Tests: Initiated by the Developer, it checks the correctness of the application code.

Build Docker Image: The Developer, through Jenkins, triggers the Docker Engine to build an image.

Push to Registry: After building, the Docker image is pushed to a container registry for storage.

Deploy to Kubernetes: The application is deployed to the Kubernetes cluster for orchestration and management.

Monitor Application: Once deployed, the application’s health and performance are monitored using tools like Prometheus and Grafana.

The UCD provides a simple and clear visual representation of the system functionalities and how users (human or system actors) interact with those functionalities, ensuring clarity in understanding the CI/CD workflow.

**Annexure D**

**Data Dictionary (DD)**

**(Mandatory)**

**Build Table (BUILD)**

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data type** | **Description** |
| BUILD-ID | Number | Build ID |
| STATUS | Text | Success /Failed |
| TIME | Date Time | Build time |

**Docker Image Table (DOCKER)**

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data type** | **Description** |

|  |  |  |
| --- | --- | --- |
| IMAGE-ID | Number | Docker image ID |
| NAME | Text | Image name |
| VERSION | Text | Tag version |

**Deployment Table (DEPLOY)**

|  |  |  |
| --- | --- | --- |
| **Fields** | **Data type** | **Description** |

|  |  |  |
| --- | --- | --- |
| DEPLOY-ID | Number | Deployment ID |
| STATUS | Text | Success /Failed |
| TIME | Date Time | Deployment time |

**Annexure E**

**Screen Shots**

**Home Page:**

