Project - High Level Design

on

ProjectTitle

Course Name: DevOps

Institution Name: Medicaps University – Datagami Skill Based Course

Sr no	Student Name	Enrolment Number
1	SUHANI MOGRA	EN22CS301989
2	VAIBHAV SHRIVASTAVA	EN22CS3011051
3	TAMANNA KATARIYA	EN22CS3011013
4	NAMAN PATIDAR	EN22ME303013
5	SNEHA ARAS	EN22CS301964

Group Name : 05D9

Project Number : 05

Industry Mentor Name: Mr. VAIBHAV

University Mentor Name:Dr. Ritesh Joshis

Academic Year : 2025-26

Table of Contents

1. Introduction

1.1 Scope of the document

This document describes the high-level architecture of a Spring Boot Logistics application integrated with GitHub CI/CD, Docker, and deployed on AWS EC2 using Tomcat.

1.2 Intended Audience

- Faculty / Evaluators
- Project Team Members
- DevOps Learners

1.3 System overview

The system automates the complete software lifecycle:

- Code written in VS Code
- Stored in GitHub
- Automatically built & tested using GitHub Actions
- Docker image created and pushed to Docker Hub
- Pulled on AWS EC2
- Deployed inside Tomcat container

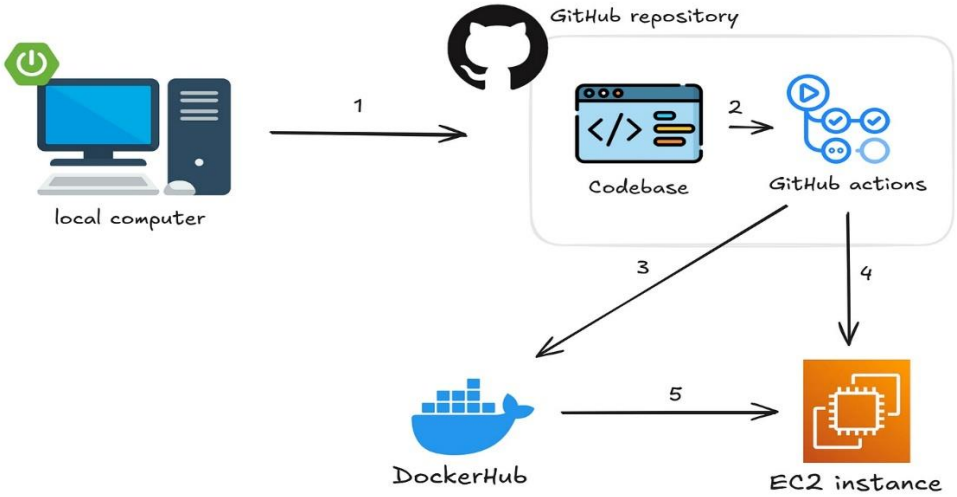
Users access the application using EC2 Public IP.

2. System Design

2.1 Application Design

- Java Spring Boot REST application
- WAR deployed on Apache Tomcat
- Docker used for containerization

Architectural Design



2.2 Process Flow

1. Developer pushes code to GitHub

2. GitHub Actions runs Maven build + tests
3. WAR file generated
4. Docker image built
5. Image pushed to Docker Hub
6. AWS EC2 pulls image
7. Container runs Tomcat + application

2.3 Information Flow

User → EC2 Public IP → Docker Container → Tomcat → Spring Controller → Response

2.4 Components Design

- VS Code – Development
- GitHub – Source control
- GitHub Actions – CI/CD
- Docker Hub – Image storage
- Docker – Container runtime
- Tomcat – Web server
- AWS EC2 – Cloud host

2.5 Key Design Considerations

- Automated deployment
- Portable containers
- Cloud hosting
- Secure credentials (GitHub Secrets)
- Stateless application

2.6 API Catalogue

Endpoint	Method	Purpose
/	GET	App status
/ health	GET	Health check

3. Data Access

REST APIs : REST APIs expose backend functionality to clients using HTTP methods. They follow REST principles such as statelessness, resource-based URLs, and standard HTTP verbs.

4. Interfaces

- Browser

- REST APIs
- Docker CLI
- GitHub UI
- AWS Console

5. State & Session Management

State and Session Management define how user data and application context are maintained across multiple requests.
In this Logistics Application, the system is designed as a **stateless application**, meaning:

- The server does **not store client session data**
- Each request contains all necessary authentication and context information
- No in-memory session tracking is maintained on the server

This design improves scalability, fault tolerance, and cloud compatibility.

6 Non-Functional Requirements

6.1 Security

- EC2 Security Group (22, 8080)
- GitHub Secrets
- Docker isolation

6.2 Performance

- Fast startup (~3 sec)
- Lightweight containers
- Automated builds

7. References

Spring Boot, Docker, GitHub Actions, AWS EC2, Tomcat Docs