**CI/CD Deployment for Spring boot Application**

\*\*Project Objective:\*\*

The objective of the CI/CD Deployment for spring boot Application project is to build a continuous integration and continuous deployment (CI/CD) pipeline for a Spring Boot application. The project aims to automate the integration and deployment process, ensuring efficient and reliable delivery of the application. The final goal is to host the application on an AWS EC2 instance, allowing users to access and utilize the application.

\*\*Overview:\*\*

The CI/CD Deployment project focuses on automating the integration and deployment of a Spring Boot application. It involves setting up a development environment using Eclipse as the Integrated Development Environment (IDE), GitHub as the source code repository, Jenkins for CI/CD pipeline configuration, and hosting the application on an AWS EC2 instance or a virtual machine. The project emphasizes the automation of the deployment process to ensure efficient and reliable application delivery.

\*\*Capabilities and Appearance:\*\*

The project focuses on the following capabilities and appearance:

- Continuous Integration: Automate the process of integrating source code changes into a central repository.

- Continuous Deployment: Automate the deployment of the application to the hosting environment.

- Source Code Management: Utilize GitHub as the repository for version control and source code management.

- Jenkins Configuration: Configure Jenkins to orchestrate the CI/CD pipeline and automate the build, test, and deployment processes.

- AWS EC2 Instance: Host the application on an AWS EC2 instance or a virtual machine for accessibility.

- Documentation: Provide detailed documentation of the project, including the step-by-step process and repository link.

\*\*User Interactions:\*\*

As a developer, the user interactions in this project involve setting up and configuring the CI/CD pipeline using the following tools and platforms:

1. Eclipse: Use Eclipse as the IDE to develop and manage the Spring Boot application.

2. GitHub: Create and maintain a GitHub repository to track and manage the source code of the application.

3. Jenkins: Configure Jenkins to automate the CI/CD pipeline, including build, test, and deployment processes.

4. AWS EC2/ Virtual Machine: Host the deployed application on an AWS EC2 instance or a virtual machine for accessibility and usage.

\*\*Sprint Planning:\*\*

The project will be divided into multiple sprints to achieve the objective of building the CI/CD pipeline and hosting the application. The sprints can be planned as follows:

\*\*Sprint 1: Environment Setup\*\*

- Set up the development environment using Eclipse as the IDE.

- Create a GitHub repository to track and manage the source code.

- Configure Jenkins for CI/CD pipeline automation.

\*\*Sprint 2: Source Code Management\*\*

- Track the necessary source code files in the GitHub repository.

- Document the files that should be ignored during the final push to the repository.

\*\*Sprint 3: Build and Test Automation\*\*

- Configure Jenkins to automate the build process for the Spring Boot application.

- Implement automated tests to ensure the application's quality and stability.

\*\*Sprint 4: Deployment Automation\*\*

- Configure Jenkins to automate the deployment process of the application.

- Set up an AWS EC2 instance or a virtual machine to host the deployed application.

\*\*Sprint 5: Documentation and Submission\*\*

- Create detailed documentation of the project, including the step-by-step process involved.

- Share the link to the GitHub repository as a submission for tracking and evaluation.

\*\*Concepts or Technologies Used:\*\*

- IDE: Eclipse

- Source Code Management: GitHub

- CI/CD Pipeline: Jenkins

- Hosting Environment: AWS EC2 or Virtual Machine

\*\*Flowchart:\*\*

Unfortunately, as a text-based AI, I cannot generate a flowchart directly. However, the general flow of the project can be represented as follows:

1. Developer sets up the development environment using Eclipse.

2. Developer creates a GitHub repository for source code management.

3. Developer configures Jenkins to automate the CI/CD pipeline.

4. Developer tracks and manages the necessary source code files in the GitHub repository.

5. Developer documents the files that should be ignored during the final push to the repository.

6. Jenkins automates the build process, triggering a build whenever changes are pushed to the repository.

7. Automated tests are executed to ensure the application's quality and stability.

8. Jenkins automates the deployment process, deploying the application to an AWS EC2 instance or a virtual machine.

9. The deployed application is hosted on the AWS EC2 instance or virtual machine, making it accessible to users.

10. Developer creates detailed documentation of the project, including the step-by-step process and the GitHub repository link.

11. The documentation and the repository link are submitted for tracking and evaluation.