For this project, I am going to design a CICD pipeline for a Java project.

My choice of CICD tool is Gitlab CI because of the following reasons:

1. **Integration with GitLab**: GitLab CI is tightly integrated with the GitLab version control system, making it seamless to use. This integration allows for a smooth workflow from code development to testing to deployment, all within the GitLab ecosystem.
2. **Comprehensive CI/CD pipeline:** GitLab CI provides a comprehensive CI/CD pipeline that supports various stages of the development lifecycle. It allows users to automate the building, testing, and deployment processes, which helps in achieving faster delivery and higher quality of software.
3. **Infrastructure as code (IaC) support**: GitLab CI supports Infrastructure as Code, enabling teams to define and manage their infrastructure using code. This allows for easier management, versioning, and tracking of changes to the infrastructure.
4. **Docker container integration:** GitLab CI has native support for Docker containers, which allows for easier and more efficient management of application dependencies and environments. This helps in creating consistent and reproducible builds, ensuring that the application behaves consistently across different environments.
5. **Scalability:** GitLab CI is known for its scalability, allowing it to handle projects of various sizes and complexities. It can easily scale to accommodate the needs of small teams as well as large enterprises with complex CI/CD requirements.
6. **Flexibility and customization:** GitLab CI provides a high degree of flexibility and customization, allowing users to configure their CI/CD pipelines according to their specific requirements. This flexibility enables teams to tailor the CI/CD processes to their unique development workflows.
7. **Open-source community:** GitLab CI is part of the GitLab platform, which has a thriving open-source community. This community contributes to the continuous improvement and enhancement of the tool, providing valuable feedback and support for users.

**STEPS**

**GitLab Setup**

* Sign up for GitLab SaaS account.
* Create a repository for the Java project within GitLab.

**Configure CI/CD Pipeline**

I would create the .gitlab-ci.yml file in the root directory of the repository to define the stages and actions of the pipeline. The file will look like this:

stages:

- build

- test

- deploy

build\_job:

stage: build

script:

- mvn clean package

test\_job:

stage: test

script:

- mvn test

deploy\_job:

stage: deploy

script:

- docker build -t the\_image\_name .

- docker run -d -p 8080:8080 the\_image\_name

script:

- ansible-playbook -i inventory playbook.yml

artifacts:

paths:

- target/\*.jar

notifications:

email:

recipients:

- [jeruto@email.com](mailto:jeruto@email.com)

**Build**

**Tool:** Maven

**Reason:** Maven is a popular build automation tool used primarily for Java projects. It simplifies the build process and manages project dependencies. It also provides a standard project structure, making it easy to manage Java projects.

**Testing**

**Tool:** JUnit

**Reason:** JUnit is a widely used testing framework for Java. It provides annotations to identify test methods, assertions for testing expected results, and test runners to execute tests. It is well-documented and has a strong community, making it a reliable choice for Java testing.

**Deploy**

**Tool:** Docker

**Reason:**

* Docker simplifies the deployment process by packaging the application and its dependencies into containers, ensuring consistency across different environments.
* Consistent Environments: Docker ensures that the Java application runs in the same environment in which it was built, regardless of the underlying host system. This consistency minimizes the risk of potential deployment issues due to differences between development and production environments.
* Version Control and Rollbacks: Docker allows me to version control the containers, making it easy to roll back to a previous version if necessary. This feature is valuable for managing updates and ensuring that any issues introduced with new deployments can be quickly addressed by reverting to a known working state
* Isolation: Docker containers provide process isolation, allowing me to run multiple instances of the Java application on the same host without conflicts. This isolation enhances security and stability, as issues within one container are less likely to affect others.

**Tool:** Ansible

**Reason:**

* Ansible for orchestration and configuration management tasks: With Ansible, I can define the desired state of the deployment environment, manage configurations, and automate complex deployment tasks. Ansible can also handle various aspects of the deployment process, such as setting up servers, managing configurations, and coordinating multiple Docker containers if necessary.
* Configuration Management: Ansible is well-suited for configuration management tasks. It allows me to define the desired state of the infrastructure, ensuring that configurations remain consistent across different servers or environments. This capability is crucial for maintaining the proper configuration of the Java application's deployment environment.
* Orchestration Capabilities: Ansible facilitates the orchestration of complex deployment tasks and workflows. It enables me to automate multi-step deployment processes, coordinate activities across multiple servers, and manage dependencies between different components of the Java application.

**Configure Notifications via Email**

GitLab also allows devops engineers to configure email notifications for pipeline status updates. By adding the email notification configuration to the .gitlab-ci.yml file, it ensures that I receive timely updates on pipeline statuses and any failures.

**Adding Artifacts**

By specifying the artifact paths in the .gitlab-ci.yml file, I can archive the built JAR files and other relevant artifacts. Archiving artifacts ensures that I can access and utilize the build outputs for future reference or for deployment purposes.