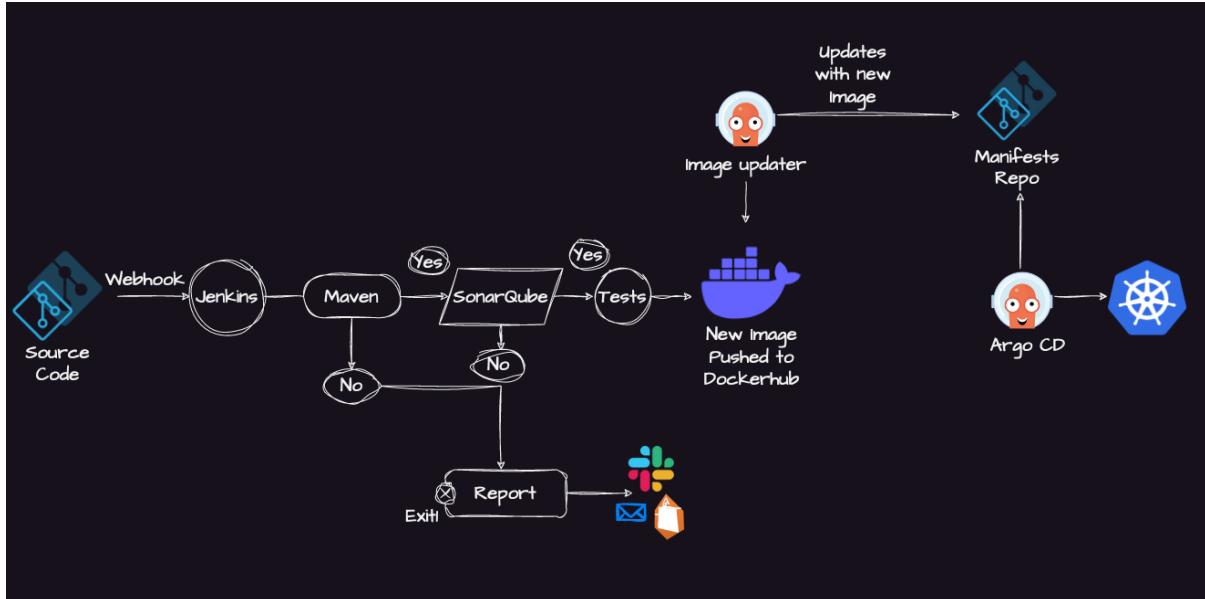


“End to End DevOps Pipeline with GitOps using ArgoCD on Kubernetes”



CI/CD Pipeline Workflow

Stage-wise Description:

➤ Continuous Integration (CI)

- **Checkout**

Fetches the source code from the main branch of the GitHub repository.

- **Build and Test**

Navigates into the spring-boot-app directory and runs mvn clean package, which:

- Compiles the code.
- Runs unit tests.

- **Static Code Analysis**

Uses SonarQube to analyze code quality.

- Authenticates using a SonarQube token from Jenkins credentials.

- **Docker Image Build and Push**

- Builds a Docker image from the Dockerfile located in spring-boot-app.
- Tags the image using the current Jenkins BUILD_NUMBER.
- Pushes the image to Docker Hub using stored credentials.

-  This marks the end of the CI phase — a tested, quality-checked, and packaged Docker image is now ready for deployment.

➤ Continuous Deployment (CD)

- **Update Deployment File**
 - Replaces the placeholder <replaceImageTag> in the Kubernetes **deployment.yml** with the actual BUILD_NUMBER.
 - Commits and pushes this change back to the GitHub main branch using a GitHub token.
- **Argo CD Auto-Sync (Automated Deployment)**
 - Argo CD continuously watches the GitHub repository for changes in the YAML manifest (deployment.yml).
 - When it detects the updated image tag:
 - It automatically pulls the change.
 - Applies the manifest to the Kubernetes cluster.
 - Deploys the new version of your application as a pod.

Tools Used in the CI/CD Pipeline

CI (Continuous Integration) Tools

Tool	Purpose
GitHub	Source code hosting and triggering the pipeline via webhooks
Jenkins	Orchestrates the CI/CD pipeline using declarative pipelines
Maven	Builds the Java project, runs unit tests, and packages the artifact
SonarQube	Conducts static code analysis to check for code quality and security
Docker	Builds application images for containerized deployment

Containerization & Registry

Tool	Purpose
Docker	Builds Docker images from the application
DockerHub	Stores Docker images and makes them accessible for deployment

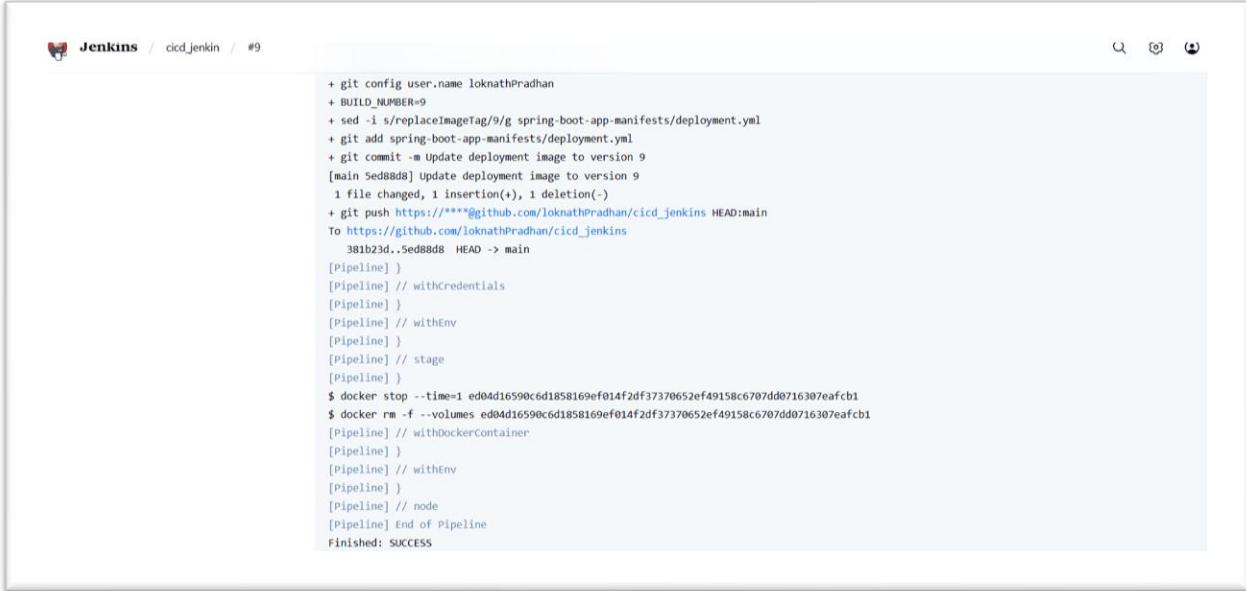
CD (Continuous Deployment) & GitOps

Tool	Purpose
Argo CD	GitOps tool that syncs Kubernetes state with the Git repository
Kubernetes	Container orchestration platform where applications are deployed

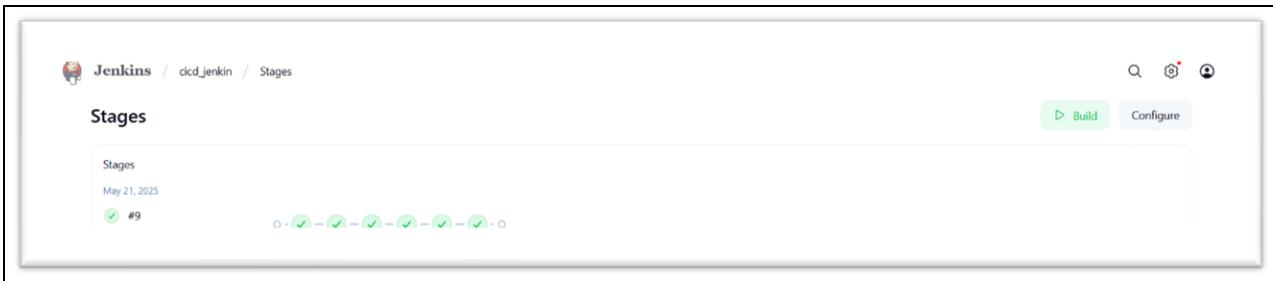
Security & Credentials

Tool/Method	Purpose
Jenkins Credentials Store	Stores secrets like SonarQube tokens, GitHub, and Docker credentials

Pipeline Execution Result



```
+ git config user.name loknathPradhan
+ BUILD_NUMBER=9
+ sed -i s/replaceImageTag/9/g spring-boot-app-manifests/deployment.yml
+ git add spring-boot-app-manifests/deployment.yml
+ git commit -m Update deployment image to version 9
[main Sed88d8] Update deployment image to version 9
1 file changed, 1 insertion(+), 1 deletion(-)
+ git push https://****@github.com/loknathPradhan/cicd_jenkins HEAD:main
To https://github.com/loknathPradhan/cicd_jenkins
 381b23d..5ed88d8 HEAD -> main
[Pipeline]
[Pipeline] // withCredentials
[Pipeline] // withEnv
[Pipeline] // stage
[Pipeline]
$ docker stop --time=1 ed04d16590c6d1858169ef014f2df37370652ef49158c6707dd0716307eafcb1
$ docker rm -f --volumes ed04d16590c6d1858169ef014f2df37370652ef49158c6707dd0716307eafcb1
[Pipeline] // withDockerContainer
[Pipeline] // withEnv
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



ArgoCD

The screenshot shows the ArgoCD interface for the 'test' application. The left sidebar includes links for Applications, Settings, User Info, and Documentation. The main area displays the application health as 'Healthy'. A 'SYNC STATUS' section indicates it is 'Synced' to HEAD (6069de8) with a green checkmark. Below this, a 'LAST SYNC' section shows a successful sync 9 minutes ago by Loknath Pradhan. The central part of the screen is a deployment graph for the 'spring-boot-app' service. It shows a 'test' deployment (green icon) branching into two replicas ('spring-boot-app-service' and 'spring-boot-app'). Each replica has its own set of endpoints and pods. The graph also includes a 'spring-boot-app-b7d9174f' pod under the second replica. The overall status of the deployment is shown as 90% healthy.

SonarQube - result

The screenshot shows the SonarQube results for the 'spring-boot-demo' project. The left sidebar contains filters for Quality Gate (Passed 1, Failed 0), Reliability (A rating 1, B 0, C 0, D 0, E 0), Security (A rating 1, B 0, C 0, D 0, E 0), and Security Review (Security Hotspots 0). The main panel shows the project summary for 'spring-boot-demo' with a 'Passed' status. Key metrics displayed include 0 Bugs (A), 0 Vulnerabilities (A), 0 Hotspots Reviewed (A), 0 Code Smells (A), 0.0% Coverage (red circle), 0.0% Duplications (green circle), and 79 Lines (XML, Java). A note at the bottom states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine." The footer indicates SonarQube™ technology is powered by SonarSource SA.

Folder structure

