



Fuzail Ahmed · [Follow](#)

8 min read · May 19



Listen



Share



More

CI/CD Pipeline: Jenkins Pipeline for Java Application with Maven, SonarQube, Argo CD, Helm, and Kubernetes

Introduction:

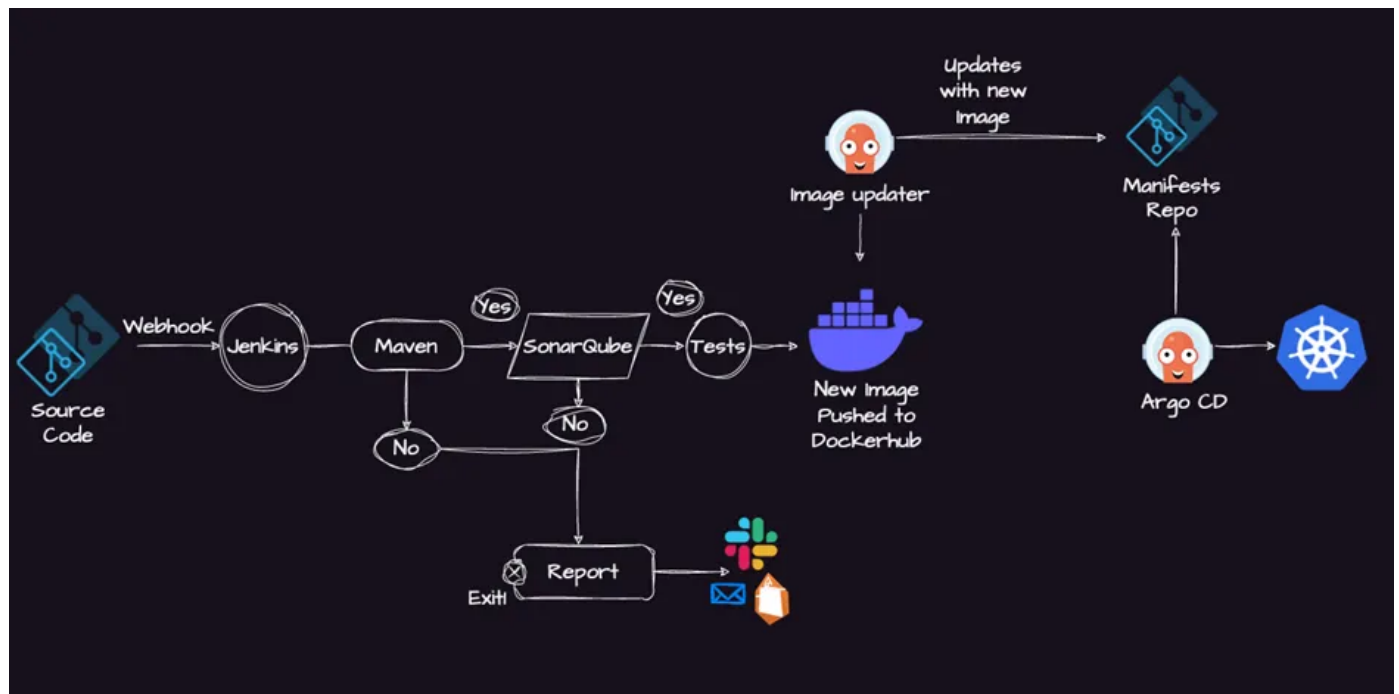
- The purpose of this documentation is to guide you through setting up a CI/CD (Continuous Integration/Continuous Deployment) pipeline for a Java-based application using Jenkins Pipeline, Maven, SonarQube, Argo CD, Helm, and Kubernetes.
- In today's software development practices, CI/CD has become an essential part of delivering high-quality software quickly and efficiently. By automating the build, test, and deployment processes, CI/CD pipelines enable teams to achieve faster release cycles, improve code quality, and increase overall productivity.
- This documentation will provide step-by-step instructions and best practices for configuring each component of the CI/CD pipeline. You'll learn how to leverage Jenkins Pipeline to define and execute your entire build and deployment process, utilize Maven for building and packaging your Java application, integrate SonarQube for code quality analysis, set up Argo CD to manage your deployments, and leverage Helm for Kubernetes deployment management.
- Whether you're starting a new project or enhancing an existing one, implementing a CI/CD pipeline using these tools will help you streamline your development workflow, increase collaboration among team members, and ensure the stability and reliability of your software releases.

· Let's dive in and get your CI/CD pipeline up and running!

Prerequisites:

- Java application code hosted on a Git repository
- Jenkins server
- Kubernetes cluster
- Helm package manager
- Argo CD

CI/CD Architecture Overview



The CI/CD architecture for the Java-based application using Jenkins Pipeline, Maven, SonarQube, Argo CD, Helm, and Kubernetes follows a streamlined and automated workflow from source code management to deployment. Here's an overview of the architecture and the role of each component:

1. Source Code Management (SCM):

- Git or any other SCM tool is used to manage the source code of the Java-based application.
- Developers push their code changes to the repository, which triggers the CI/CD pipeline.

2. Jenkins:

- Jenkins serves as the central orchestrator of the CI/CD pipeline.
- It receives triggers from the SCM upon code changes and initiates the pipeline.
- Jenkins Pipeline, defined in the Jenkinsfile, provides a declarative approach to define the build and deployment stages.

3. Maven:

- Maven is responsible for building, testing, and packaging the Java application.
- It reads the project's configuration file (pom.xml) and resolves dependencies, compiles source code, runs tests, and generates deployable artifacts.

4. SonarQube:

- SonarQube is integrated into the pipeline to analyze and assess the quality of the code.
- It performs static code analysis, identifies bugs, security vulnerabilities, and code smells.
- SonarQube provides detailed reports and metrics to help improve code quality.

5. Argo CD:

- Argo CD is a declarative continuous delivery tool used for managing Kubernetes deployments.
- It pulls the deployment manifests (Helm charts or YAML files) from the repository and deploys them to Kubernetes clusters.

- Argo CD ensures consistent and reliable deployments, tracks application versions, and supports rollbacks and canary deployments.

6. Helm:

- Helm is a package manager for Kubernetes that simplifies the deployment and management of applications.
- It uses Helm charts, which define the structure, configuration, and dependencies of an application.
- Helm packages the application as a chart, and Argo CD uses Helm to deploy and manage the application on Kubernetes clusters.

7. Kubernetes:

- Kubernetes is an open-source container orchestration platform.
- It provides a scalable and resilient environment for running containerized applications.
- Kubernetes manages the deployment, scaling, and monitoring of application components (pods, services, deployments) across clusters.

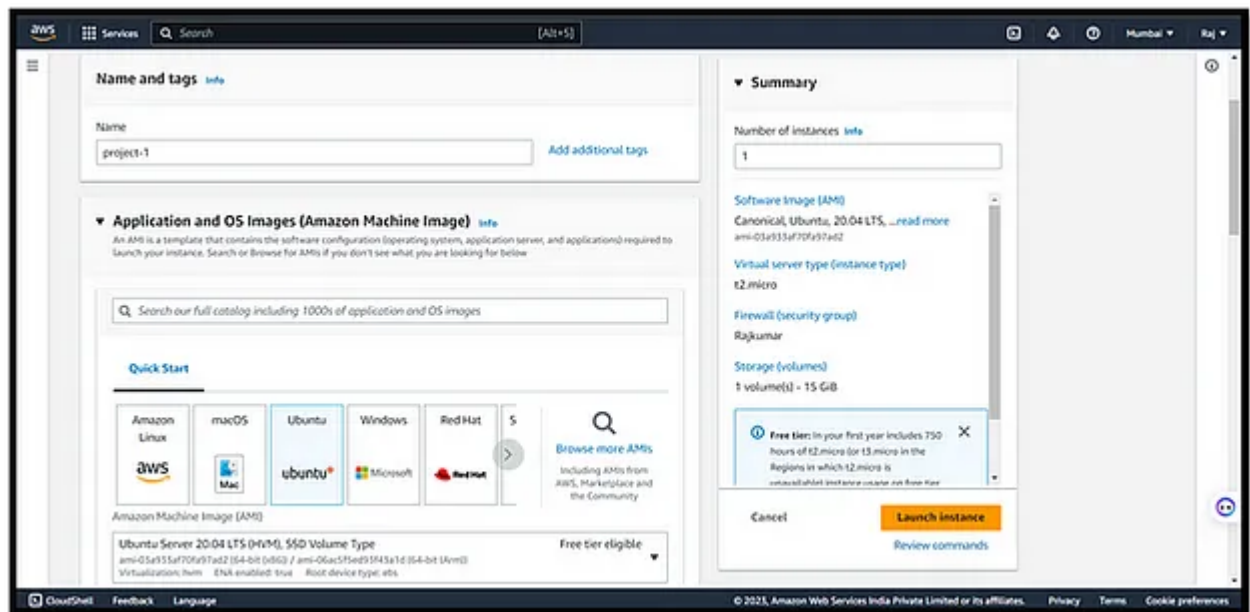
The CI/CD architecture integrates these components to automate the software delivery process, ensuring the build, test, and deployment stages are performed consistently and efficiently. Developers push code changes, triggering Jenkins to initiate the pipeline. Jenkins orchestrates the build process using Maven, performs code quality analysis using SonarQube, and deploys the application using Argo CD and Helm on Kubernetes clusters.

—

Step -1: Create EC2 Instance:

- **No. of Instance — 01**

- AMI — Ubuntu Server 20.04 LTS (HVM), SSD Volume Type
- Instance type — t2.medium
- Security group — ALL TCP
- “Launch Instance”



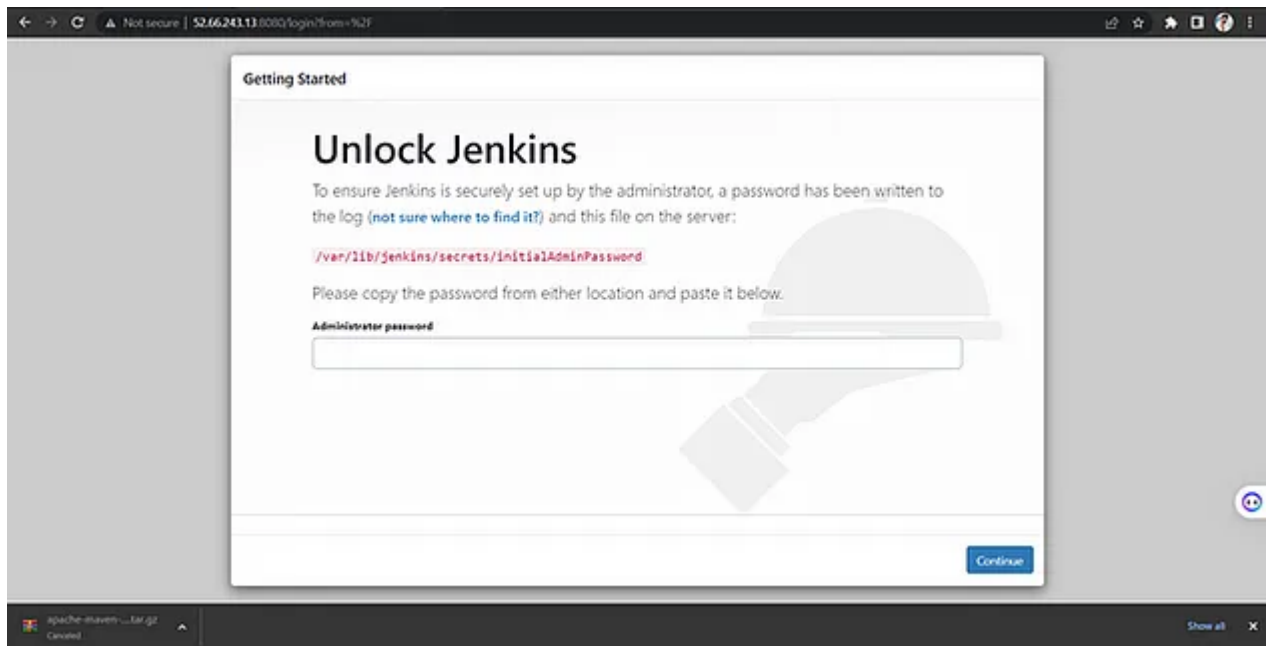
Connect with terminal and login as “Ubuntu” user and switch to root user using “sudo su” command.

Step 2: Installing — Jenkins

```
# install java11
# Jenkins requires Java to run. If you haven't installed Java on your system, you can
apt install openjdk-11-jre
java --version
# install jenkins
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
jenkins -version
```

```
# Once the installation is complete, you can start Jenkins using the following command
systemctl start jenkins
# You can check the status of Jenkins using the following command:
sudo systemctl status jenkins
```

- By default, Jenkins runs on port 8080. You can access Jenkins by opening your web browser and entering the URL `http://<your-server-IP>:8080`.



Your-Server-IP:8080

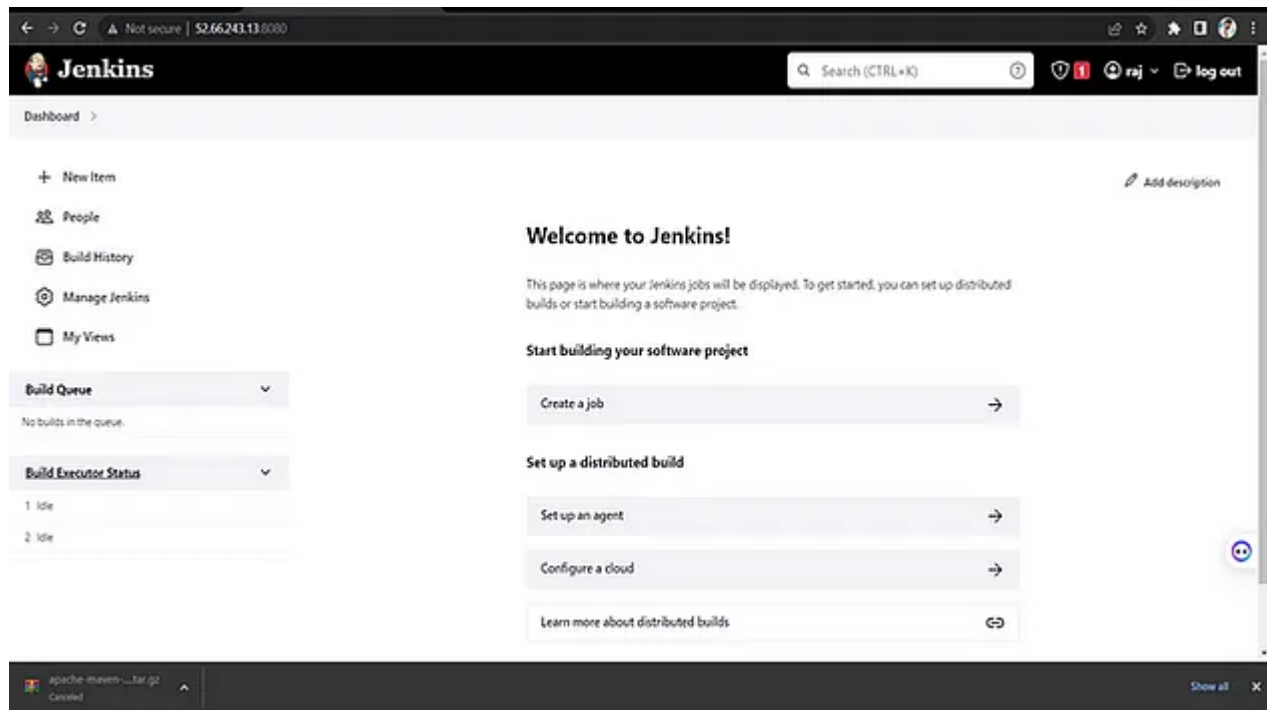
Step 3: Configure Jenkins

- When you first access Jenkins, you will be prompted to unlock Jenkins. To do this, you need to retrieve the initial administrator password from the Jenkins server using the following command:

```
root@ip-172-31-1-79: /opt # cat /var/lib/jenkins/secrets/initialAdminPassword
259761ac8ce7458986599387b41063d6
root@ip-172-31-1-79: /opt #
```

- Copy the password and paste it into the “Administrator password” field on the Jenkins login screen.

- Once you have unlocked Jenkins, you will be prompted to install recommended plugins. You can either install the recommended plugins or select the plugins that you want to install manually.



Jenkins Dashboard

- New Item → Pipeline → Ok
- Pipeline → Definition — Pipeline Script from SCM → SCM — Git → Git URL

Open in app ↗



Search Medium



Dashboard > ultimate-demo > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

Script Path ?

java-maven-sonar-argocd-helm-k8s/spring-boot-app/JenkinsFile

☒ Lightweight checkout ?

Pipeline Syntax

Save Apply

Apply & Save

Step4: Install Docker & SonarQube Plugin in Jenkins

Dahboard → Manage Jenkins → Available Plugin → “Docker Pipeline”

Dashboard > Manage Jenkins > Plugins

Plugins

Search: docker pipeline

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<div>Docker Pipeline 563.vd5d2e5c4007f</div> <div>pipeline DevOps Deployment docker</div> <div>Build and use Docker containers from pipelines.</div> <div>This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.</div>	4 mo 2 days ago

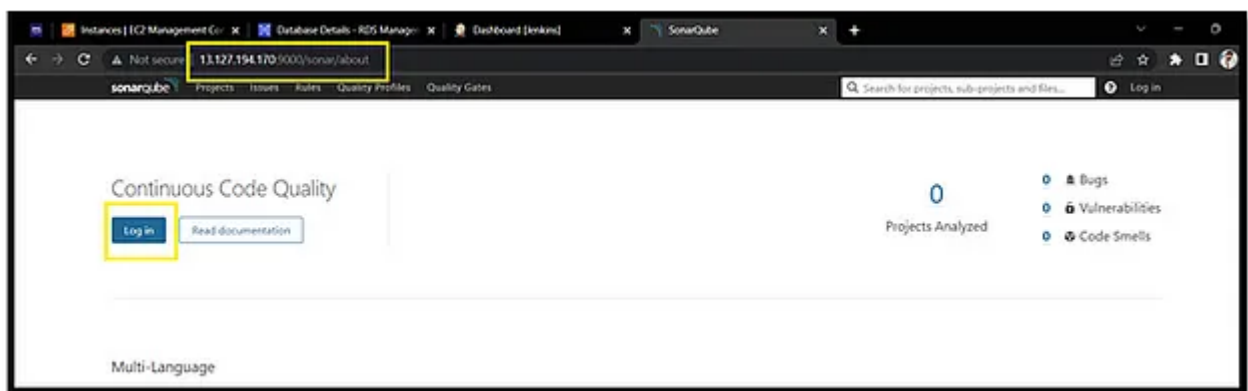
Install without restart



Step 5: Installing Sonar in EC2 Server:

```
adduser sonarqube
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
apt uninstall unzip
unzip *
chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424
chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424
cd sonarqube-9.4.0.54424/bin/linux-x86-64/
./sonar.sh start
```

- You can access SonarQube by opening your web browser and entering the URL `http://<your-server-ip>:9000`



(login — Username — admin; Password — admin)

- my account → security → generate token → name(Jenkins) → generate

(generate and copy the token)

(Add credential for sonar and paste token on secret)

Step 6: Installing Docker in EC2 Server:

```
sudo apt update
sudo apt install docker.io

#Grant Jenkins user and Ubuntu user permission to docker daemon.
sudo su -
usermod -aG docker jenkins
usermod -aG docker ubuntu
systemctl restart docker
```

Once you are done with the above steps, it is better to restart Jenkins.

```
http://<ec2-instance-public-ip>:8080/restart
```

The docker agent configuration is now successful.

Step 6: Install minikube (you can do this follow step in your localhost or vm server)

1. Update and upgrade the system:

```
sudo apt-get update -y  
sudo apt-get upgrade -y
```

2. Restart the computer to apply any system updates.

3. Install required packages:

```
sudo apt-get install curl wget apt-transport-https -y
```

4. Install the VirtualBox Hypervisor (if you prefer to use it):

```
sudo apt-get install virtualbox virtualbox-ext-pack
```

5. Download the latest version of Minikube

```
wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

6. Copy the downloaded file to `/usr/local/bin`.

```
sudo cp minikube-linux-amd64 /usr/local/bin/minikube
```

7. Provide execution permissions to Minikube:

```
sudo chmod 755 /usr/local/bin/minikube
```

8. Check the version of Minikube

```
minikube version
```

Now install the Agro CD using operators (mainy peoples don't have a idea but once go through it you will get to know how simple the Kubernetes will install.)

Go to the website <https://operatorhub.io/operator/argocd-operator>

Step7: DockerHub & Github Credential in Jenkins

Docker Hub Username & Password

- Goto Github → setting → developer setting → token classic → generate token

paste the Github token in secret

- Restart the Jenkins — <server-ip:8080/restart/>

Build Now

Check the SonarQube

```
vi argocd-basic.yml
```

```
kubectl apply -f argocd-basic.yml
```

change cluster ip type nodeport

- above highlighted link — copied this link & hit in the browser
- username — admin
- password — goto EC2 server
- kubectl get secret — you will find admin.password — copy
- echo <paste admin.password> | base64 -d
- you will the password & login

username — admin

- Create
- App details → edit → namespace

- Sync

This end-to-end Jenkins pipeline will automate the entire CI/CD process for a Java application, from code checkout to production deployment, using popular tools like SonarQube, Argo CD, Helm, and Kubernetes.

Thank you for your time!



Follow



Written by Fuzail Ahmed

27 Followers

AWS DevOps

More from Fuzail Ahmed



Fuzail Ahmed

Building a CI/CD Pipeline with Github, Docker, SonarQube, Jenkins, Maven, and Nexus

1. Introduction to CI/CD Pipeline

8 min read · May 14



10



Fuzail Ahmed

Create and Host a Wordpress Website on AWS EC2 with your own domain name

Step 1: Launch an instance

4 min read · Jun 14



27





Fuzail Ahmed

Day 1—Introduction to Terraform

♦ Infrastructure as Code (IaC)

6 min read · Jun '7





Fuzail Ahmed

Day 2 - Terraform Configuration Language (HCL)

HCL (HashiCorp Configuration Language) is a declarative language used in Terraform to define infrastructure configurations. It provides a...

8 min read · Jun 7



1



See all from Fuzail Ahmed

Recommended from Medium



Cumhuri Mesut Akkaya

Working with Microservices-14: Creating Amazon RDS MySQL database for Kubernetes cluster in the...

We continue our Production Pipeline. In this section, we will use Amazon RDS instead of MySQL pod and service during the Production stage...

7 min read · Aug 16



jabir ahammed

Complete Jenkins CI/CD Project

6 min read · Jul 27



6



Lists



Staff Picks

423 stories · 253 saves

Stories to Help You Level-Up at Work


19 stories · 194 saves

Self-Improvement 101

20 stories · 491 saves

Productivity 101

20 stories · 474 saves

 Aman Pathak in DevOps.dev

Deployed Java application using Maven, Sonarqube, Jfrog Artifactor, Jenkins, ArgoCD and finally...

How to build, test, Code Analysis, Upload build Artifacts, build docker Image, Push docker Image and Deploy the Java-based Application?

9 min read · May 19

 1 



Ghazanfar Ali

Kubernetes End To End Project:

Introduction:

9 min read · Aug 9



6





Fuzail Ahmed

Create and Host a Wordpress Website on AWS EC2 with your own domain name

Step 1: Launch an instance

4 min read · Jun 14



27



Mudit Mathur

Day-22 Getting Started with Jenkins

Table of Contents

7 min read · Aug 18



17



[See more recommendations](#)