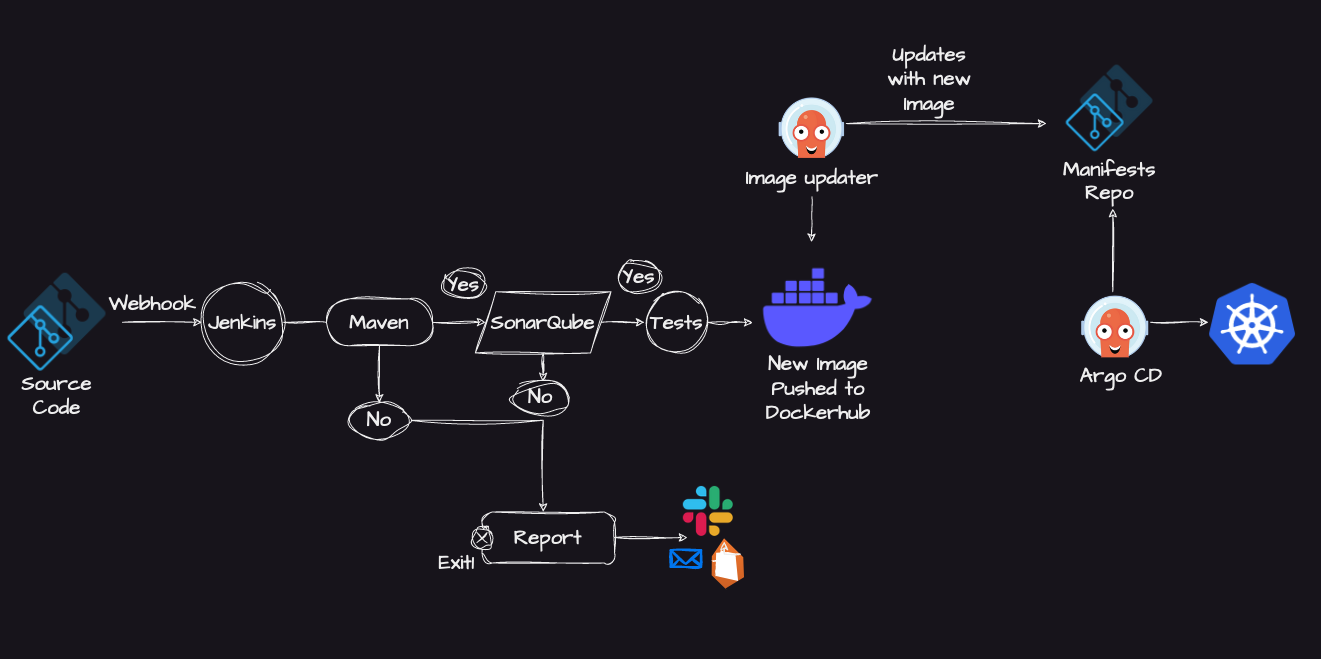
# **Jenkins Pipeline for Java based application using Maven, SonarQube, Argo CD and Kubernete**[**s**](https://github.com/vignesh020919997/Jenkins-Zero-To-Hero/tree/main/java-maven-sonar-argocd-helm-k8s#jenkins-pipeline-for-java-based-application-using-maven-sonarqube-argo-cd-helm-and-kubernetes)

**ARCHITECTURE**

**Source code link:**

[**https://github.com/vignesh020919997/Jenkins-Zero-To-Hero**](https://github.com/vignesh020919997/Jenkins-Zero-To-Hero)

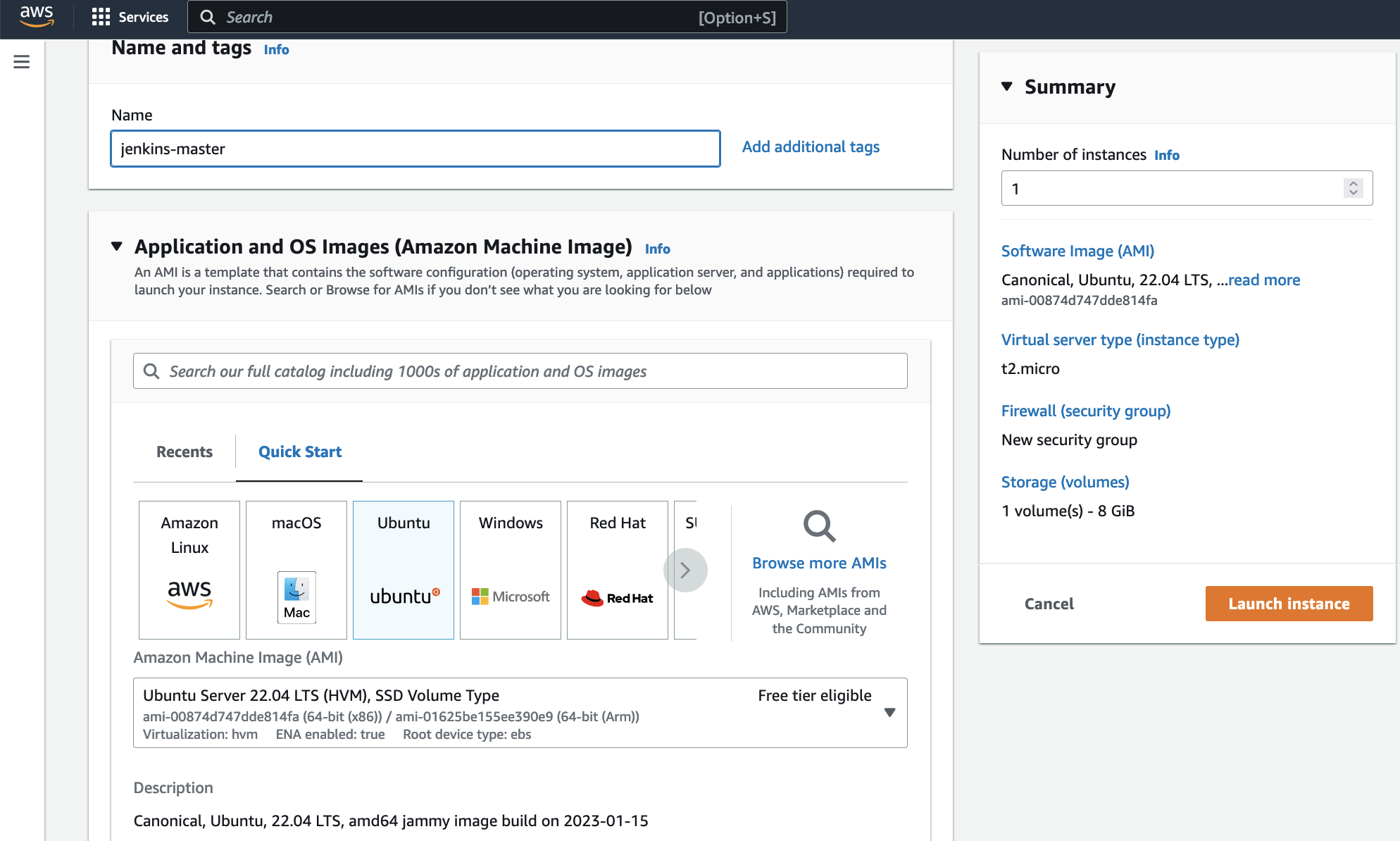
****

**Prerequisites:**

* **Java application code hosted on a Git repository**
* **Jenkins server**
* **sonarQube**
* **Kubernetes cluster**
* **Maven**
* **Argo CD**

## [AWS EC2 Instance](https://github.com/vignesh020919997/Jenkins-Zero-To-Hero/tree/main#aws-ec2-instance)

* Go to AWS Console
* Instances(running)
* Launch instances



### 2. Install Jenkins

Pre-Requisites:

* Java (JDK)

### 3 . Run the below commands to install Java and Jenkins

* Install Java

sudo apt update

sudo apt install openjdk-11-jre

* Verify Java is Installed

java -version

* Now, you can proceed with installing Jenkins

sudo apt-get install jenkins

curl -fsSL https://pkg.jenkins.io/debian/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 binary/ | sudo tee \

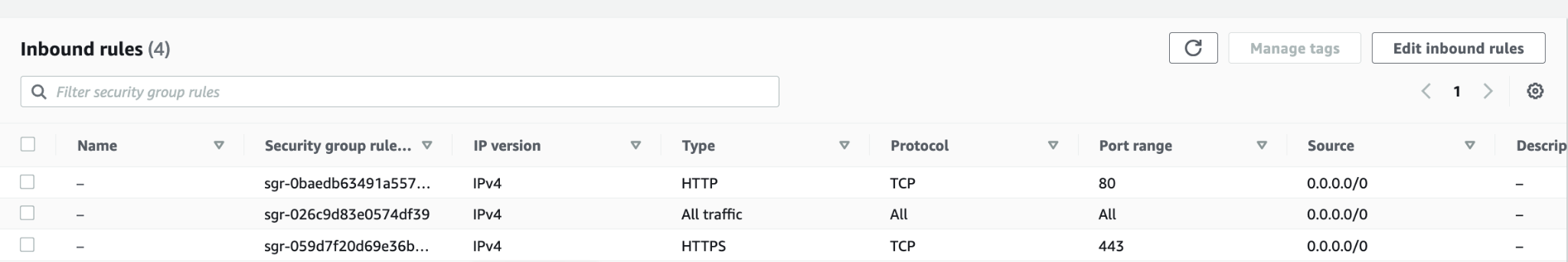
/etc/apt/sources.list.d/jenkins.list > /dev/null

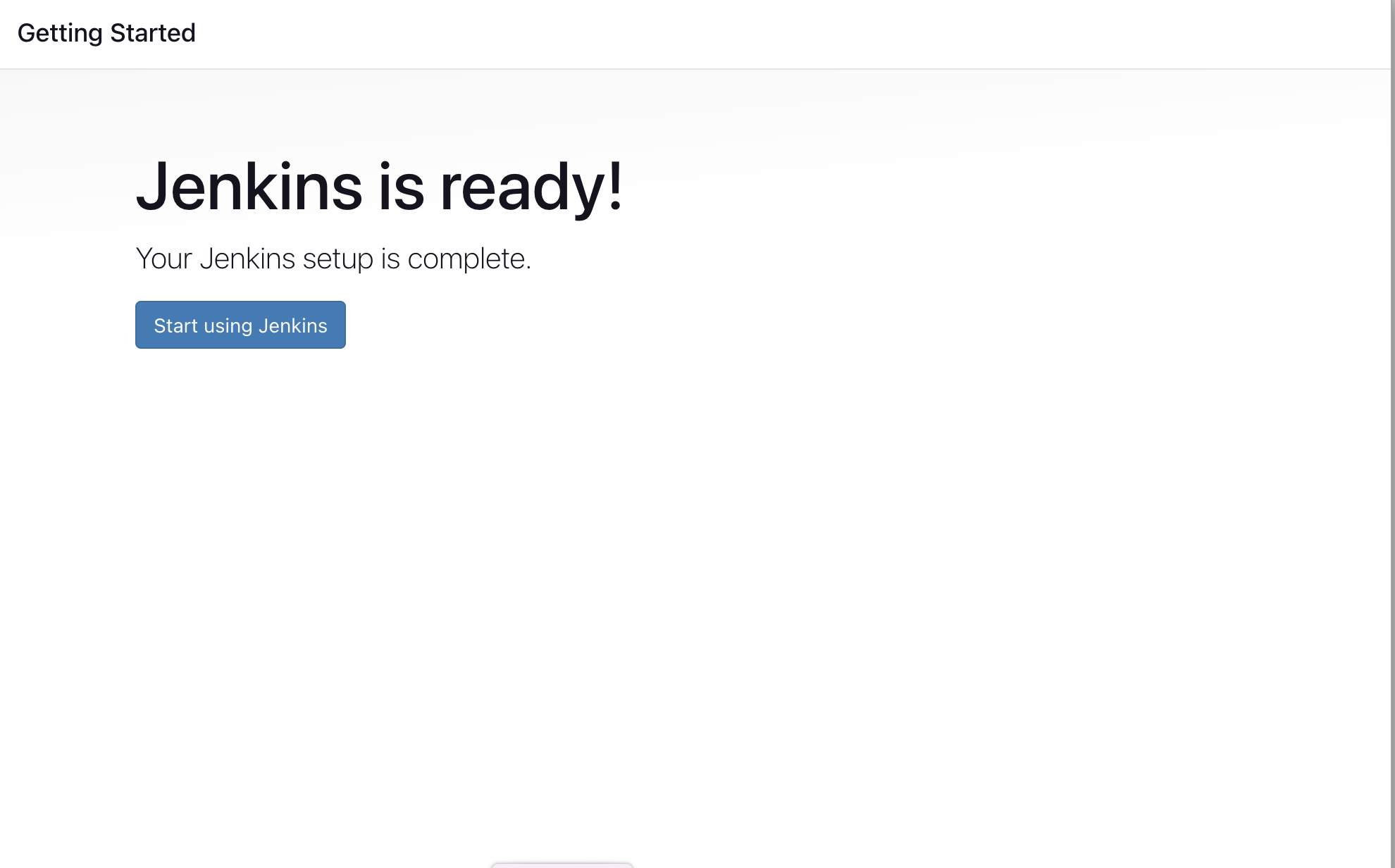
sudo apt-get update

sudo apt-get install jenkins

\*\*Note: \*\* By default, Jenkins will not be accessible to the external world due to the inbound traffic restriction by AWS. Open port 8080 in the inbound traffic rules as show below.

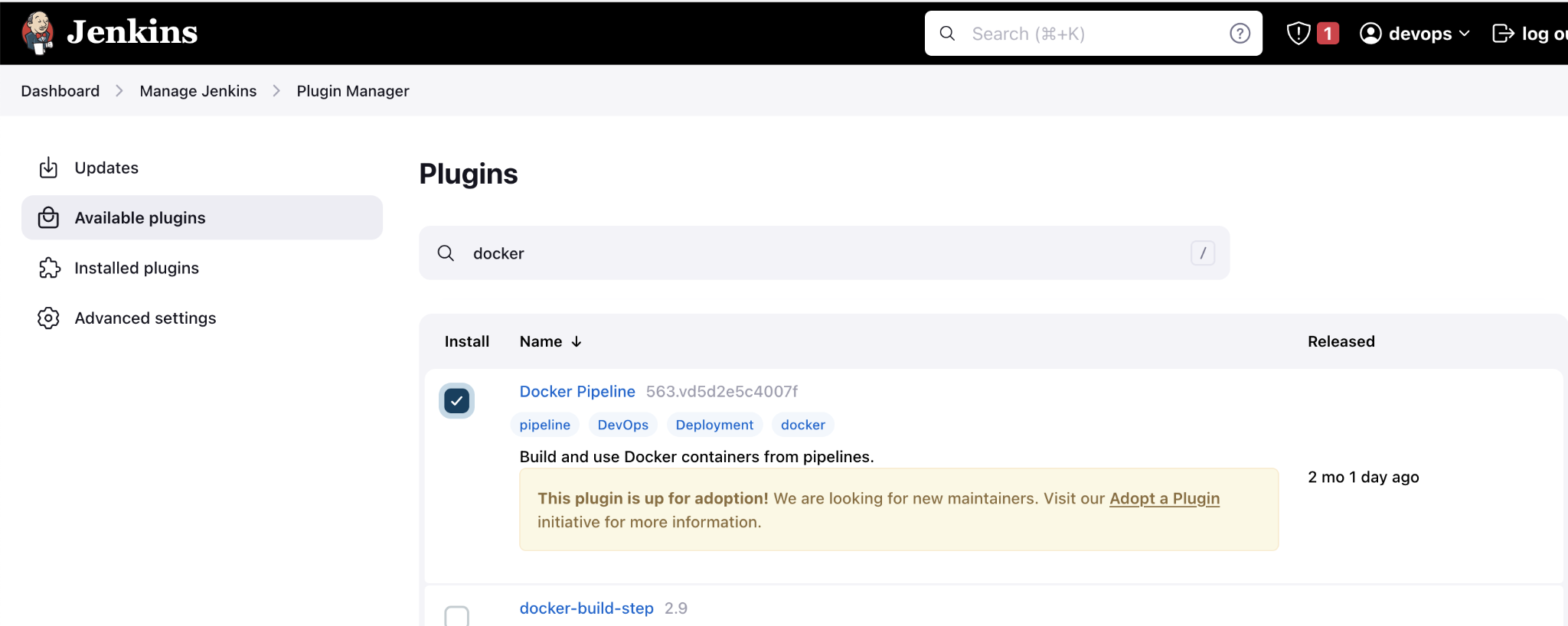
* EC2 > Instances > Click on
* In the bottom tabs -> Click on Security
* Security groups
* Add inbound traffic rules as shown in the image (you can just allow TCP 8080 as well, in my case, I allowed All traffic).





## 4. Install the Docker Pipeline plugin in Jenkins

* Log in to Jenkins.
* Go to Manage Jenkins > Manage Plugins.
* In the Available tab, search for "Docker Pipeline".
* Select the plugin and click the Install button.
* Restart Jenkins after the plugin is installed.



### Grant Jenkins user and Ubuntu user permission to docker deamon.

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.**

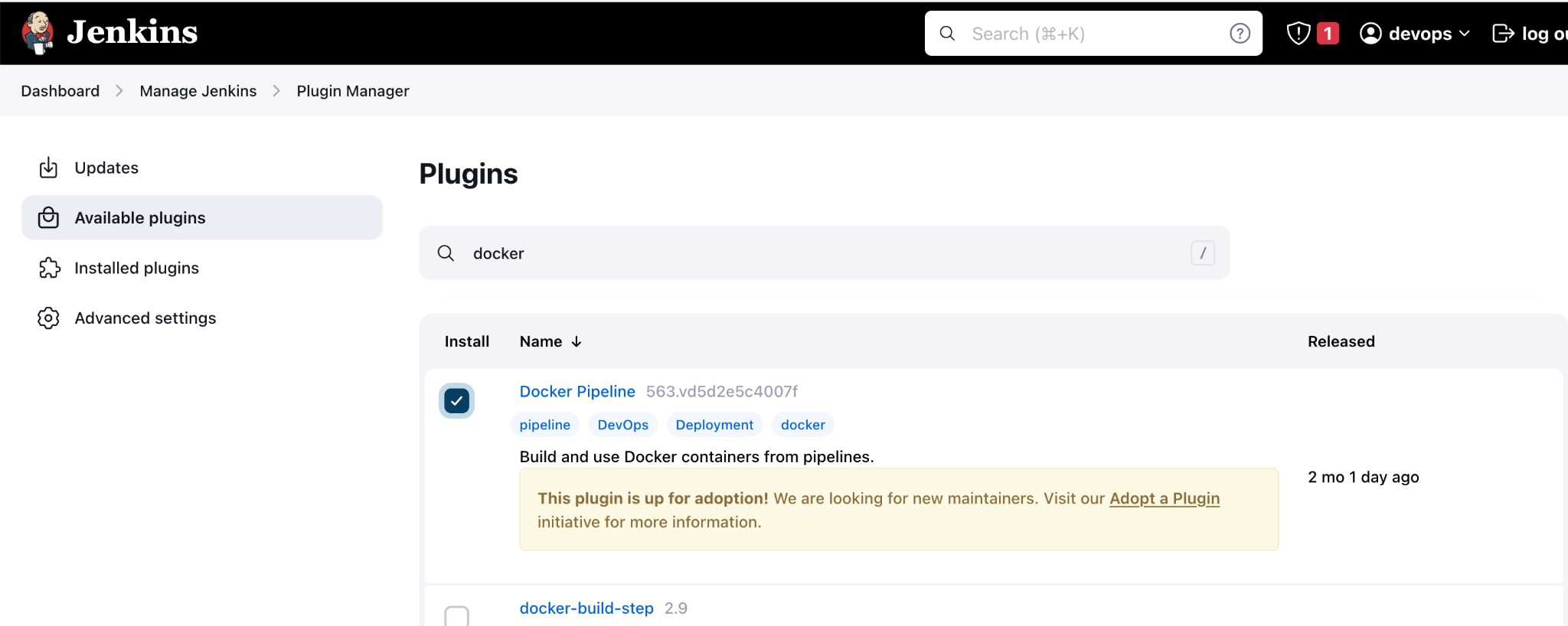
Install the necessary Jenkins plugins:

1.1 Git plugin

1.2 Maven Integration plugin

1.3 Pipeline plugin

1.4 Kubernetes Continuous Deploy plugin



Create a new Jenkins pipeline:

1 In Jenkins, create a new pipeline job and configure it with the Git repository URL for the Java application.

2 Add a Jenkinsfile to the Git repository to define the pipeline stages.

Stage 1: Checkout the source code from Git.

Stage 2: Build the Java application using Maven.

Stage 3: Run SonarQube analysis to check the code quality.

Stage 4: Package the application into a JAR file.

Stage 5: Run user acceptance tests on the deployed application.

Stage 6: Promote the application to a production environment using Argo CD.

**PIPELINE**

**pipeline {**

**agent {**

**docker {**

**image 'abhishekf5/maven-abhishek-docker-agent:v1'**

**args '--user root -v /var/run/docker.sock:/var/run/docker.sock' // mount Docker socket to access the host's Docker daemon**

**}**

**}**

**stages {**

**stage('Checkout') {**

**steps {**

**sh 'echo passed'**

**//git branch: 'main', url: 'https://github.com/vignesh020919997/Jenkins-Zero-To-Hero.git'**

**}**

**}**

**stage('Build and Test') {**

**steps {**

**sh 'ls -ltr'**

**// build the project and create a JAR file**

**sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn clean package'**

**}**

**}**

**stage('Static Code Analysis') {**

**environment {**

**SONAR\_URL = "http://18.218.210.98:9000"**

**}**

**steps {**

**withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR\_AUTH\_TOKEN')]) {**

**sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR\_AUTH\_TOKEN -Dsonar.host.url=${SONAR\_URL}'**

**}**

**}**

**}**

**stage('Build and Push Docker Image') {**

**environment {**

**DOCKER\_IMAGE = "dockervignesh97/ultimate-cicd:${BUILD\_NUMBER}"**

**// DOCKERFILE\_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"**

**REGISTRY\_CREDENTIALS = credentials('docker-cred')**

**}**

**steps {**

**script {**

**sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER\_IMAGE} .'**

**def dockerImage = docker.image("${DOCKER\_IMAGE}")**

**docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {**

**dockerImage.push()**

**}**

**}**

**}**

**}**

**stage('Update Deployment File') {**

**environment {**

**GIT\_REPO\_NAME = "Jenkins-Zero-To-Hero"**

**GIT\_USER\_NAME = "vignesh020919997"**

**}**

**steps {**

**withCredentials([string(credentialsId: 'github', variable: 'GITHUB\_TOKEN')]) {**

**sh '''**

**git config user.email "abhishek.xyz@gmail.com"**

**git config user.name "Abhishek Veeramalla"**

**BUILD\_NUMBER=${BUILD\_NUMBER}**

**sed -i "s/replaceImageTag/${BUILD\_NUMBER}/g" java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yml**

**git add java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yml**

**git commit -m "Update deployment image to version ${BUILD\_NUMBER}"**

**git push https://${GITHUB\_TOKEN}@github.com/${GIT\_USER\_NAME}/${GIT\_REPO\_NAME} HEAD:main**

**'''**

**}**

**}**

**}**

**}**

**}**

Configure Jenkins pipeline stages:

Stage 1: Use the Git plugin to check out the source code from the Git repository.

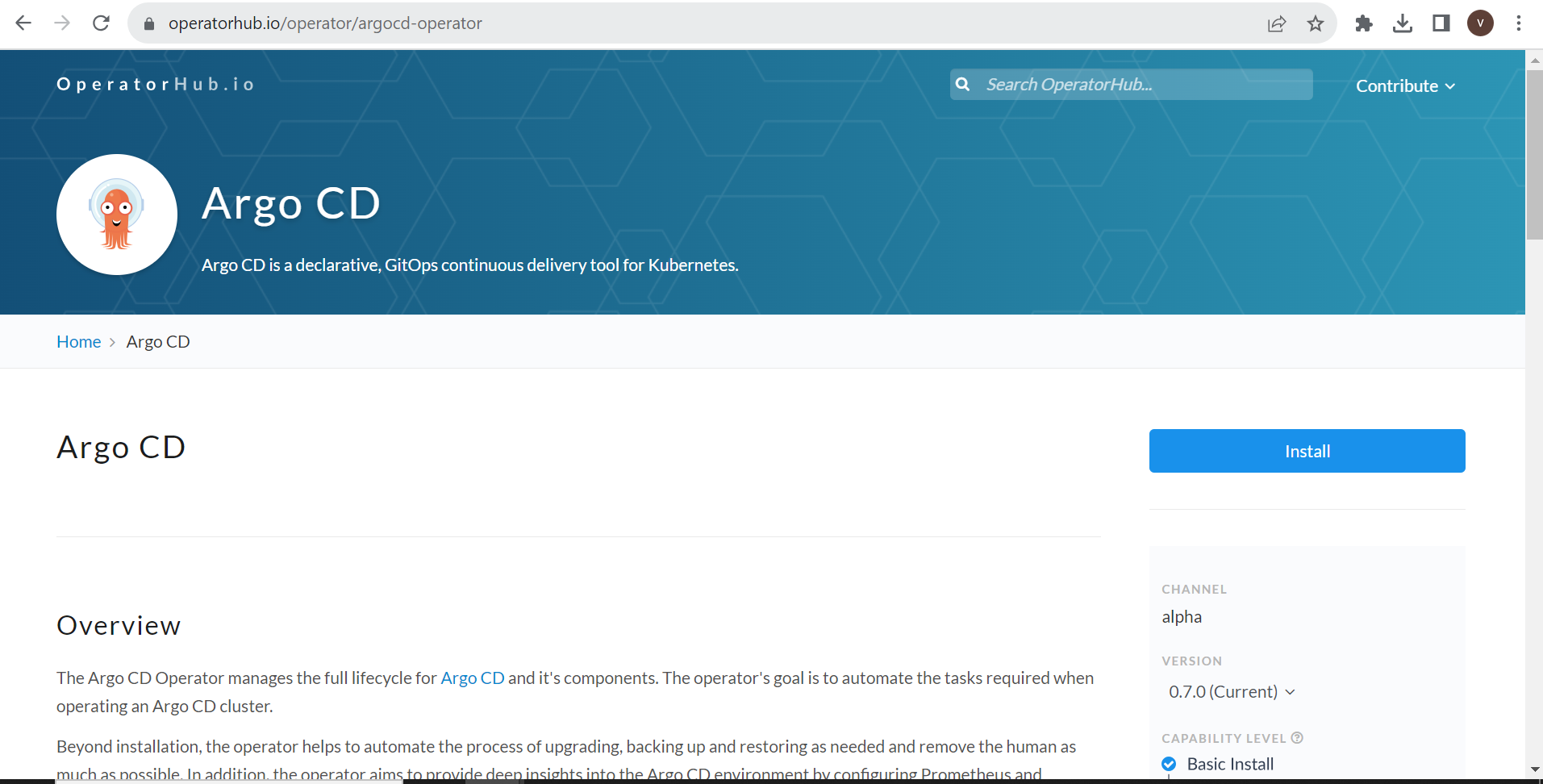
Stage 2: Use the Maven Integration plugin to build the Java application.

Stage 3: Use the SonarQube plugin to analyze the code quality of the Java application.

Stage 4: Use the Maven Integration plugin to package the application into a JAR file.

Stage 5: Use a testing framework like Selenium to run user acceptance tests on the deployed application.

Stage 6: Use Argo CD to promote the application to a production environment.



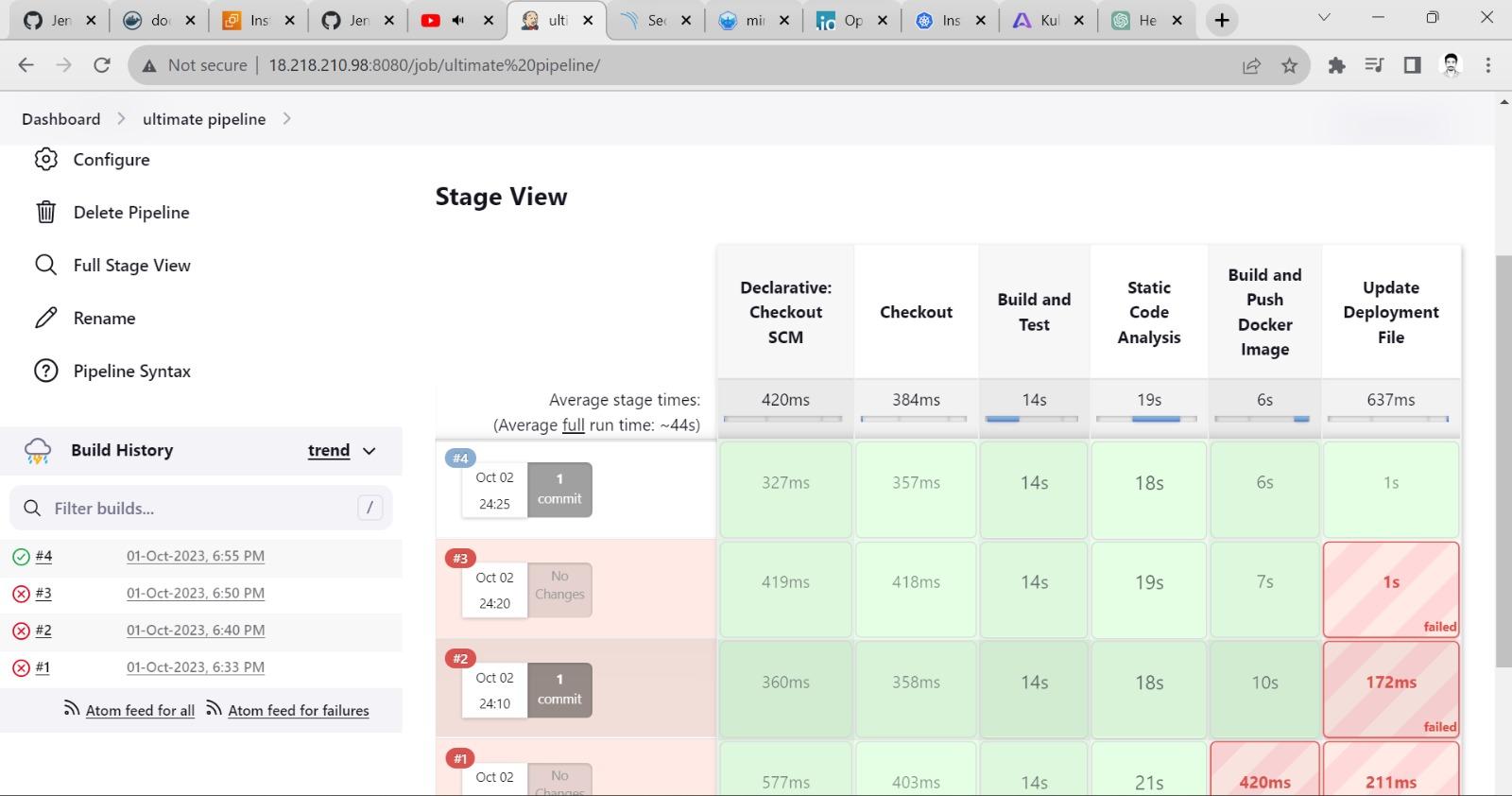
Set up Argo CD:

Install Argo CD on the Kubernetes cluster.

Set up a Git repository for Argo CD to track the changes in the Helm charts and Kubernetes manifests.

Create a Helm chart for the Java application that includes the Kubernetes manifests and Helm values.

Add the Helm chart to the Git repository that Argo CD is tracking



Run the Jenkins pipeline:

1 Trigger the Jenkins pipeline to start the CI/CD process for the Java application.

2 Monitor the pipeline stages and fix any issues that arise.

