End-to-End Jenkins Pipeline with SonarQube for addressbook-cicd-project

This guide helps you set up Jenkins + SonarQube on an Ubuntu EC2, run a CI pipeline for the GitHub repo https://github.com/akshu20791/addressbook-cicd-project, and deploy the app as a Docker container. It includes a production-ready Jenkinsfile and a checklist of screenshots to capture.

# Prerequisites (Ubuntu 22.04 EC2 recommended)

- Security Group: open inbound 22 (SSH), 8080 (Jenkins UI), 9000 (SonarQube), and 80/443 if needed.

- Instance type: t3.medium or better (SonarQube needs >= 4GB RAM).

- Login as ubuntu user with sudo privileges.

# 1) Install Java, Docker, Jenkins, and Maven

sudo apt-get update -y  
  
# Java 17  
sudo apt-get install -y fontconfig openjdk-17-jdk  
  
# Docker  
sudo apt-get install -y ca-certificates curl gnupg lsb-release  
sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update -y  
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin  
  
# Jenkins LTS  
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 -y  
sudo apt-get install -y jenkins  
  
# Add Jenkins to docker group  
sudo usermod -aG docker jenkins  
sudo systemctl restart docker  
sudo systemctl enable --now jenkins  
  
# Maven  
sudo apt-get install -y maven

Open Jenkins at http://<EC2-Public-IP>:8080. Unlock with the initial admin password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

# 2) Install Jenkins Plugins and Tools

Manage Jenkins → Plugins → Available: install "Pipeline", "Git", "SonarQube Scanner for Jenkins", "Docker Pipeline".

Manage Jenkins → Tools:

\* JDK: Name "JDK17", set JAVA\_HOME to /usr/lib/jvm/java-17-openjdk-amd64 (or let Jenkins auto-install).

\* Maven: Name "Maven3" (auto-install OK).

# 3) Run SonarQube with PostgreSQL (Docker)

# Network  
sudo docker network create sonarnet || true  
  
# PostgreSQL  
sudo docker run -d --name sonarqube\_db --network sonarnet -e POSTGRES\_USER=sonar -e POSTGRES\_PASSWORD=sonar -e POSTGRES\_DB=sonarqube -v sonar\_db:/var/lib/postgresql/data -p 5432:5432 postgres:13  
  
# SonarQube Community LTS  
sudo docker run -d --name sonarqube --network sonarnet -e SONAR\_JDBC\_URL=jdbc:postgresql://sonarqube\_db:5432/sonarqube -e SONAR\_JDBC\_USERNAME=sonar -e SONAR\_JDBC\_PASSWORD=sonar -p 9000:9000 sonarqube:lts-community

Open SonarQube at http://<EC2-Public-IP>:9000 (default login: admin / admin → force new password).

Generate a token: Your avatar → My Account → Security → Generate Token (name it "jenkins-token"). Copy it.

# 4) Configure SonarQube in Jenkins

Manage Jenkins → System → "SonarQube servers": Add → Name "MySonarQube", Server URL http://<EC2-IP>:9000, "Server authentication token": add as a Secret Text credential using the token you generated.

Manage Jenkins → Credentials (Global):

\* Secret Text: ID "sonar-token" → paste your SonarQube token.

\* Username/Password: ID "dockerhub-creds" for Docker Hub (if you will push images).

SonarQube Webhook (to enable Quality Gate wait): In SonarQube UI → Administration → Configuration → Webhooks → Create:

\* Name: Jenkins

\* URL: http://<JENKINS\_URL>/sonarqube-webhook/ (e.g., http://<EC2-IP>:8080/sonarqube-webhook/).

# 5) Create a Jenkins Pipeline Job

In Jenkins: New Item → Pipeline → "addressbook-cicd-pipeline" → Pipeline script → paste Jenkinsfile below → Save → Build Now.

# Jenkinsfile (copy-paste)

pipeline {  
 agent any  
  
 tools {  
 jdk 'JDK17'  
 maven 'Maven3'  
 }  
  
 environment {  
 GIT\_REPO = 'https://github.com/akshu20791/addressbook-cicd-project'  
 APP\_NAME = 'addressbook'  
 REGISTRY = 'docker.io/<your-dockerhub-username>'  
 IMAGE = "${REGISTRY}/${APP\_NAME}:${BUILD\_NUMBER}"  
  
 // Jenkins Global Configuration -> SonarQube servers -> Name must match  
 SONARQUBE\_SERVER = 'MySonarQube'  
  
 // Jenkins Credentials (add in "Manage Jenkins" -> Credentials)  
 SONAR\_TOKEN = credentials('sonar-token')  
 DOCKERHUB = credentials('dockerhub-creds')  
 }  
  
 stages {  
 stage('Checkout') {  
 steps {  
 git url: "${GIT\_REPO}", branch: 'master'  
 }  
 }  
  
 stage('Build & Unit Tests (Maven)') {  
 steps {  
 sh 'mvn -B -DskipTests=false clean verify'  
 }  
 post {  
 always {  
 junit '\*\*/target/surefire-reports/\*.xml'  
 }  
 }  
 }  
  
 stage('SonarQube Analysis') {  
 steps {  
 withSonarQubeEnv("${SONARQUBE\_SERVER}") {  
 // Uses the configured SonarQube server URL and exposes SONAR\_HOST\_URL  
 sh '''  
 mvn -B sonar:sonar -Dsonar.projectKey=${APP\_NAME} -Dsonar.projectName=${APP\_NAME} -Dsonar.token=${SONAR\_TOKEN} -Dsonar.host.url=$SONAR\_HOST\_URL  
 '''  
 }  
 }  
 }  
  
 stage('Quality Gate') {  
 steps {  
 timeout(time: 10, unit: 'MINUTES') {  
 waitForQualityGate abortPipeline: true  
 }  
 }  
 }  
  
 stage('Build Docker Image') {  
 steps {  
 sh '''  
 docker build -t ${IMAGE} .  
 '''  
 }  
 }  
  
 stage('Push Image') {  
 steps {  
 sh '''  
 echo "${DOCKERHUB\_PSW}" | docker login -u "${DOCKERHUB\_USR}" --password-stdin  
 docker push ${IMAGE}  
 docker logout  
 '''  
 }  
 }  
  
 stage('Deploy (Docker Run on Jenkins Host)') {  
 steps {  
 sh '''  
 docker rm -f ${APP\_NAME} || true  
 # Expose 8080; change if your app listens on a different port  
 docker run -d --name ${APP\_NAME} -p 8080:8080 ${IMAGE}  
 '''  
 }  
 }  
 }  
  
 post {  
 success {  
 echo "Deployed ${APP\_NAME}. Try: http://<EC2-Public-IP>:8080"  
 }  
 always {  
 archiveArtifacts allowEmptyArchive: true, artifacts: 'target/\*.jar'  
 }  
 }  
}

## If the project is not Maven-based (fallback)

If the repository lacks pom.xml, use a CLI scanner instead. Install "SonarScanner" on Jenkins agent and run sonar-scanner with sonar-project.properties.

# 6) Verify Deployment

On the EC2 where Jenkins runs:

\* Check container: sudo docker ps

\* Logs: sudo docker logs -f addressbook

In a browser: http://<EC2-Public-IP>:8080 should load the AddressBook app.

In SonarQube: Projects → addressbook → see Bugs, Vulnerabilities, Code Smells, Duplications, and the Quality Gate status.

# Troubleshooting Tips

- If "waitForQualityGate" hangs: verify SonarQube webhook URL points to http://<EC2-IP>:8080/sonarqube-webhook/ and security group allows inbound 8080.

- If Docker permission denied: ensure "sudo usermod -aG docker jenkins" then restart Jenkins, or run docker with sudo (update Jenkinsfile accordingly).

- If build fails on Java version: confirm Jenkins Tools → JDK is 17 and Maven uses the same JDK.

- If app listens on a different port: update the Dockerfile EXPOSE and the run command (-p host:container) to match.

# Screenshot Checklist (Capture and Save)

- Jenkins → Manage Jenkins → Tools showing JDK17 and Maven3.

- Jenkins → Manage Jenkins → System → SonarQube servers with "MySonarQube".

- SonarQube → Webhooks page showing Jenkins webhook.

- Jenkins pipeline run: stages view (green) and console log portion with Quality Gate passed.

- SonarQube project dashboard with metrics.

- docker ps output with the running container.

- Browser screenshot of http://<EC2-Public-IP>:8080.

Happy shipping!