SonarQube Integration

**Overview**

| **Component** | **Purpose** |
| --- | --- |
| SonarQube Server | Analyzes code for bugs, code smells, vulnerabilities |
| Jenkins | Automates build and triggers Sonar scans |
| ArgoCD | Deploys only after code passes quality gates |
| Project Structure | backend/ (Spring Boot) + frontend/ (HTML/JS/CSS) |

**🛠️ Step-by-Step Implementation**

**1. Install Docker and Docker Compose on EC2**

Target EC2: *Jenkins Agent + Kubeadm Master*

sudo apt-get update

sudo apt-get install docker.io -y

sudo systemctl start docker

sudo systemctl enable docker

Create Docker volumes (optional but recommended for persistence):

docker volume create sonarqube\_data

docker volume create sonarqube\_logs

docker volume create sonarqube\_extensions

**2. Run SonarQube Container**

docker run -d --name sonarqube \

-p 9000:9000 \

-v sonarqube\_data:/opt/sonarqube/data \

-v sonarqube\_logs:/opt/sonarqube/logs \

-v sonarqube\_extensions:/opt/sonarqube/extensions \

sonarqube:community

Access SonarQube via browser:  
http://13.233.150.66:9000

**3. Configure Jenkins Integration**

* Go to Jenkins → **Manage Jenkins → Configure System**
* Under **SonarQube Servers**, add:
  + Name: sonarqube
  + Server URL: http://13.233.150.66:9000
  + Authentication token: Generated under SonarQube UI (My Account → Security)

**4. Add SonarScanner Tool in Jenkins**

Go to Jenkins → **Global Tool Configuration**

* Under **SonarQube Scanner installations**:
  + Add new scanner: SonarScanner-CLI
  + Install automatically (version: 7.1.0.4889)
  + Save configuration

**5. Update Jenkinsfile**

You added the following stage:

stage("SonarQube Analysis") {

steps {

withSonarQubeEnv('sonarqube') {

script {

def scannerHome = tool 'SonarScanner-CLI'

sh "${scannerHome}/bin/sonar-scanner " +

"-Dsonar.projectKey=quantumsoft-app " +

"-Dsonar.sources=. " +

"-Dsonar.host.url=http://13.233.150.66:9000"

}

}

}

}

You later improved it by splitting frontend/backend analysis separately.

**6. Frontend Configuration (Static HTML/CSS/JS)**

Created a file: frontend/sonar-project.properties

sonar.projectKey=quantumsoft-frontend

sonar.projectName=Frontend App

sonar.sources=.

sonar.inclusions=\*\*/\*.html,\*\*/\*.css,\*\*/\*.js

sonar.language=js

Pipeline stage:

stage("SonarQube Frontend Analysis") {

steps {

withSonarQubeEnv('sonarqube') {

script {

def scannerHome = tool 'SonarScanner-CLI'

sh "${scannerHome}/bin/sonar-scanner -Dproject.settings=frontend/sonar-project.properties"

}

}

}

}

**7. Backend Configuration (Spring Boot)**

Used Maven with the embedded SonarQube plugin:

stage("SonarQube Backend Analysis") {

steps {

withSonarQubeEnv('sonarqube') {

sh '''

cd backend

mvn sonar:sonar \

-Dsonar.projectKey=quantumsoft-backend \

-Dsonar.host.url=http://13.233.150.66:9000

'''

}

}

}

**8. Enforce Quality Gates (Recommended)**

stage("Quality Gate") {

steps {

timeout(time: 1, unit: 'MINUTES') {

waitForQualityGate abortPipeline: true

}

}

}

This prevents Docker image builds or ArgoCD deployments if code doesn’t meet standards.

**Results & Next Steps**

* Scans are running and results showing in SonarQube UI
* You handled agent path issues and corrected CLI invocation
* Logs confirm successful scan engine and indexed files
* Future: you can enable PR-level scanning, create custom quality profiles, or integrate Sonar badges in your GitHub repo