SonarQube Notes (DevOps Friendly)

**📌 What is SonarQube?**

**SonarQube** is an open-source platform used for **continuous inspection** of code quality. It performs:

* **Static code analysis**
* **Bug detection**
* **Code smells identification**
* **Security vulnerability analysis**
* **Technical debt estimation**

Supports 25+ programming languages like Java, Python, JavaScript, C#, etc.

**🚀 Key Features**

* **Code Quality Reports**
* **Security Vulnerability Detection (SAST)**
* **Customizable Quality Gates**
* **Integration with CI/CD tools (Jenkins, GitHub Actions)**
* **Code coverage support via external tools (JaCoCo, etc.)**

**⚙️ SonarQube Architecture**

1. **SonarQube Server** – Hosts web interface and database.
2. **SonarScanner** – CLI tool that analyzes source code and sends results to SonarQube server.
3. **Database** – Stores project analysis results.
4. **Web Interface** – For viewing reports, dashboards, and managing projects.

**🔄 Workflow of SonarQube in DevOps**

1. Developer writes code and pushes to Git.
2. Jenkins triggers a build and runs tests.
3. SonarScanner runs during the CI pipeline.
4. Code analysis results sent to SonarQube server.
5. Reports are viewed via the SonarQube web UI.
6. Build may pass/fail based on **Quality.**

**🧠 Key Terms**

| **Term** | **Meaning** |
| --- | --- |
| **Code Smells** | Maintainability issues |
| **Bugs** | Logic/functional issues |
| **Vulnerabilities** | Security issues |
| **Quality Gate** | Set of rules to determine pass/fail |
| **SonarScanner** | CLI tool to run analysis |
| **Technical Debt** | Time to fix code quality issues |

**💡 Best Practices**

* Run analysis early in CI pipeline.
* Set meaningful **Quality Gates**.
* Use with **code coverage** tools like JaCoCo.
* Review **hotspots** and **duplicated code** regularly.
* Automate checks with Jenkins .

**🧪 Integrating SonarQube with Jenkins**

SonarQube

-first we have to create an instance and connect

-now install the maven for java/ npm for nodejs

: sudo apt install maven -y / : sudo apt install npm -y

-then install jenkins and run

-then open the jenkins application and login

-now we have to install the required plugins

: Satge View, sonarQube Scanner, NodeJS

-then create another instance with sonarQube as name

-select the t2.large

-and also add Metadata version

-add below data in user data

: #!/bin/bash

git clone https://github.com/dinesh-4136/installations.git

cd installations

sh docker.sh

-then launch the instance and connect

-then check the docker is running or not

: systemctl status docker

-then add the non-root user to the docker for accessing docker

: usermod -aG docker ubuntu

: newgrp docker

-then check the user got the permissions for docker

: docker images

-now run the docker command to create container for sonarQube

: docker run --name sonarqube -d -p 9000:9000 sonarqube:lts

-check the sonarqube container created or not

: docker ps

-now launch the sonaraqube and login

-copy the url fo the sonarQube

-now open the jenkins

-select the 'manage jenkins' and select 'system'

-add the sonarqube installations

-give the name: sonar-server

-paste the copied sonarqube url: http://18.21.205.125:9000/

-server authentication token

: add new token

: go to the sonarqube, click on Administration and select security

: in the security select users and add token

: enter token name and click on generate

: copy the token code

-Domain: Global

-kind: secret file

-scope: global

-secret: paste the copied token (squ\_2ccb5872f0d51d8dd40c0971fd7d27f471ec8c83)

-id: sonar-token

-description: sonar-token

-then click add

: select the added token

-then apply and save

-then again open the 'manage jenkins' and select 'tools'

-add the sonarqube scanner installations

-Name: sonar-scanner

-version: select required version (5.0.1.3006)

-then apply and save

-add the NodeJS installations

-Name: Nodejs18

-version: select the required version (18.20.8)

-then apply and save

-Now create a pipeline with the name SonarQube

-update the configuration

-write the jenkins file

pipeline {

agent any

stages {

stage('Checkout SCM') {

steps {

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[credentialsId: 'my-github', url: 'https://github.com/dinesh-4136/JavaWebCalculator.git']])

}

}

stage('Maven Compile') {

steps {

sh 'mvn clean compile'

}

}

stage('Maven Test') {

steps {

sh 'mvn test'

}

}

stage('SonarQube Analysis') {

environment {

scannerHome = tool 'sonar-scanner'

}

steps {

withSonarQubeEnv('sonar-server') {

sh '''

${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=JavaWeb \

-Dsonar.projectName=JavaWebCalculator \

-Dsonar.projectVersion=1.0 \

-Dsonar.sources=src/ \

-Dsonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/ \

-Dsonar.junit.reportsPath=target/surefire-reports/ \

-Dsonar.jacoco.reportsPath=target/jacoco.exec \

-Dsonar.java.checkstyle.reportPaths=target/checkstyle-result.xml

'''

}

timeout(time: 10, unit: 'MINUTES') {

waitForQualityGate abortPipeline: true

}

}

stage('Quality Gate') {

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'

}

}

}

}

}

pipeline {

agent any

stages {

stage('Checkout Code') {

steps {

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[credentialsId: 'github', url: 'https://github.com/dinesh-4136/vprofile-project.git']])

}

}

stage('Maven Compile') {

steps {

sh 'mvn clean compile'

}

}

stage('Maven Test') {

steps {

sh 'mvn test'

}

}

stage('CODE ANALYSIS with SONARQUBE') {

environment {

scannerHome = tool 'sonar-scanner'

}

steps {

withSonarQubeEnv('sonar-server') {

sh '''${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=vprofile \

-Dsonar.projectName=vprofile-repo \

-Dsonar.projectVersion=1.0 \

-Dsonar.sources=src/ \

-Dsonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/ \

-Dsonar.junit.reportsPath=target/surefire-reports/ \

-Dsonar.jacoco.reportsPath=target/jacoco.exec \

-Dsonar.java.checkstyle.reportPaths=target/checkstyle-result.xml'''

}

timeout(time: 10, unit: 'MINUTES') {

waitForQualityGate abortPipeline: true

}

}

}

}

}

pipeline {

agent any

tools {

nodejs 'nodejs-18'

}

environment {

SONAR\_SCANNER = tool 'sonar-scanner'

SONARQUBE\_SERVER = 'sonar-server'

}

stages {

stage('Checkout SCM') {

steps {

"github clone url"

}

}

stage('install dependencies') {

steps {

sh 'npm install'

}

}

stage('Code Analysis with SonarQube') {

steps {

withSonarQubeEnv("${SONARQUBE\_SERVER}") {

sh '''

${SONAR\_SCANNER}/bin/sonar-scanner \

-Dsonar.projectKey=nginxwebdb \

-Dsonar.projectName=nginxwebdb-node \

-Dsonar.projectVersion=1.0 \

-Dsonar.sources=. \

-Dsonar.exclusions=node\_modules/\*\* \

'''

}

}

}

stage('Quality Gate') {

steps {

timeout(time: 5, unit: 'MINUTES') {

waitForQualityGate abortPipeline: true

}

}

}

}

}

-now open the sonarqube and select the configuration and select the webhook

-now create a webhook

-name: jenkins-sonar

-url: copy and paste the jenkins url with sonarqube-webhook/ (http://54.92.196.222:8080/sonarqube-webhook/)

-and click on create

-then build the sonarqube pipeline

-after build is completed, then check the sonarqube for the scan results