Angular Calculator App — SonarQube Code Analysis (Full Guide)

Purpose:
This document is a complete, step-by-step training guide for performing static code analysis on a JavaScript/TypeScript Angular application using SonarQube. It covers environment setup, preparing the Angular project, running analyses (sonar-scanner and Maven where applicable), interpreting results, integrating with CI (Jenkins), and remediation best practices.

Table of Contents

10. Fixing common issues (examples and code snippets)
11. CI Integration — Jenkins pipeline example
12. Automating Quality Gates & Pull Request analysis
13. Reporting & dashboards
14. Troubleshooting
15. Appendix: Commands & files

---

1. Overview

SonarQube analyzes source code to find bugs, vulnerabilities, code smells, and duplications. For Angular apps (TypeScript + HTML + CSS), SonarQube inspects:

- TypeScript and JavaScript logic
- HTML templates
- CSS files
- Unit test results (coverage)
- Dependency vulnerabilities via SCA plugins (optional)

This guide uses the built-in H2 DB for quick training and SonarScanner CLI to upload results to SonarQube server.

2. Prerequisites

- Ubuntu 20.04+ (steps shown on Ubuntu 24.04 in training)
- Java 17 installed
- Node.js v18 (or LTS) and npm
- Angular CLI installed
- SonarQube server accessible (example: `http://18.218.37.139:9000`)
- Sonar token created (e.g., `squ_...`)

3. SonarQube server (quick recap)

If SonarQube is already installed at `/opt/sonarqube` and running, ensure it's reachable from the analysis machine:

curl -I http://18.218.37.139:9000

If admin UI is needed: login with `admin/admin` (change password after first login).

> For production: use PostgreSQL, tune JVM memory, run behind a reverse proxy, and secure with HTTPS.

4. Project preparation (Angular Calculator)

Project path: `~/Angular-Calc-App`

4.1 Restore/verify `angular.json`

Make sure `angular.json` is compatible with your Angular CLI. Remove deprecated properties (e.g., `extractCss`) if using Angular CLI v12+.

4.2 Install dependencies and build

cd ~/Angular-Calc-App
npm install
ng build --configuration production

If build errors appear (schema issues), edit `angular.json` and remove deprecated flags.

5. Sonar Scanner CLI — install & configure

5.1 Download & install (example for v7.3.0)

cd /opt
sudo wget
https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-7.3.0.5189-linux-x64.zip
sudo apt install unzip -y
sudo unzip sonar-scanner-cli-7.3.0.5189-linux-x64.zip
sudo mv sonar-scanner-7.3.0.5189-linux-x64 sonar-scanner
echo 'export PATH=$PATH:/opt/sonar-scanner/bin' | sudo tee
/etc/profile.d/sonar-scanner.sh
source /etc/profile.d/sonar-scanner.sh
sonar-scanner -v

5.2 Configure `sonar-scanner.properties` (optional global settings)

Edit `/opt/sonar-scanner/conf/sonar-scanner.properties` to set defaults like server URL and authentication (but storing tokens in project file is fine for training).

sonar.host.url=http://18.218.37.139:9000
#sonar.login=YOUR_TOKEN # optional global; prefer per-project sonar.login

6. Sonar project configuration (`sonar-project.properties`)

Create `sonar-project.properties` in the project root with these contents:

sonar.projectKey=AngularCalculator
sonar.projectName=Angular Calculator App
sonar.projectVersion=1.0
sonar.sources=src
sonar.language=js
sonar.sourceEncoding=UTF-8
sonar.exclusions=**/node_modules/**,**/dist/**,**/*.spec.ts
sonar.host.url=http://18.218.37.139:9000
sonar.login=squ_c2e738e7a1dd54a92d75a005513b313fe21e656c

Notes:

- `sonar.language` can be omitted; Sonar detects languages automatically. For TypeScript projects, you may set properties to include `.ts` files.

If you have unit test coverage reports (e.g., `coverage/lcov.info`), add:

sonar.javascript.lcov.reportPaths=coverage/lcov.info
sonar.typescript.lcov.reportPaths=coverage/lcov.info

-

7. Running a local scan

From the project root:

cd ~/Angular-Calc-App
sonar-scanner

What to watch for:

- `ANALYSIS SUCCESSFUL` message
- Link to the project dashboard: `http://18.218.37.139:9000/dashboard?id=AngularCalculator`

If scanner cannot find project root, ensure `sonar-project.properties` exists and `sonar.projectKey` is unique.

8. Interpreting SonarQube results

When you open the project dashboard, key panels are:

- **Overview / Project Summary**: quality gate status, reliability/security/maintainability ratings
- **Bugs**: definite bugs
- **Vulnerabilities**: security-related issues
- **Code Smells**: maintainability/cleanliness issues
- **Coverage**: percentage of lines covered by unit tests
- **Duplications**: duplicate blocks across project

Quality Gate

Quality gates are sets of conditions (e.g., no new critical vulnerabilities, coverage > 80%). Use default gate or create custom ones under **Quality Gates**.

9. Common rule types and examples (JS/TS)

Reliability (bugs)

- Null dereference
- Wrong usage of `Array.prototype` methods

Security (vulnerabilities)

- Use of `eval()` or unsafe `innerHTML`
- Insecure HTTP calls with sensitive data

Maintainability (code smells)

- Long functions (> N lines)
- Complex conditional logic (Cyclomatic complexity)

Example Sonar rule IDs (examples)

- `javascript:S1172` — Unused function parameters
- `typescript:S131` — Track complex code constructs

10. Fixing common issues (examples)

Example 1 — Remove unused imports (TypeScript)

Before:

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router'; // unused
```

```
@Component({ ... })
export class CalcComponent implements OnInit { ... }
```

After: remove `Router` import, or use it.

Example 2 — Avoid `any` types

Before:

```
calculate(input: any) {
  // ...
}
```

After:

```
calculate(input: number): number {
  // ...
}
```

Example 3 — Template XSS

Avoid binding untrusted HTML directly: use Angular sanitization or avoid `innerHTML`.

11. CI Integration — Jenkins pipeline example

Jenkins Preparations

- Install **SonarQube Scanner** plugin
- Add SonarQube server under **Manage Jenkins** → **Configure System** with credentials (token)

Example `Jenkinsfile` for Angular

```
pipeline {
  agent any
  environment {
    SONAR_HOST = 'http://18.218.37.139:9000'
    SONAR_TOKEN = credentials('sonar-token-id')
  }
  stages {
    stage('Checkout') {
      steps { checkout scm }
    }
    stage('Install') {
      steps {
        sh 'npm ci'
      }
    }
    stage('Build') {
      steps {
        sh 'ng build --configuration production'
      }
    }
    stage('Test & Coverage') {
      steps {
        sh 'npm test -- --watch=false --code-coverage'
        // ensure coverage lcov is generated at coverage/lcov.info
      }
    }
    stage('SonarQube Analysis') {
      steps {
        withSonarQubeEnv('SonarQubeServer') {
          sh 'sonar-scanner -Dsonar.projectKey=AngularCalculator
-Dsonar.login=$SONAR_TOKEN'
        }
      }
```

```
    }
    stage('Quality Gate') {
      steps {
        timeout(time: 5, unit: 'MINUTES') {
          waitForQualityGate abortPipeline: true
        }
      }
    }
  }
}
```

Notes:

- For Jenkins pipelines, prefer using `withSonarQubeEnv` and a globally configured SonarQube installation to avoid hardcoding server URL.

- Store tokens in Jenkins credentials for security.

12. Automating Quality Gates & Pull Request analysis

- Enable **Pull Request Decoration** on SonarQube and configure your SCM (GitHub/GitLab) integration to display comments and checks on PRs.

- Use **sonar.pullrequest.key, sonar.pullrequest.branch** and related properties when running scanners for PRs.

Example PR scan properties (in Jenkins or scanner command):

```
-Dsonar.pullrequest.key=${CHANGE_ID} \
-Dsonar.pullrequest.branch=${BRANCH_NAME} \
-Dsonar.pullrequest.base=${TARGET_BRANCH}
```

13. Reporting & dashboards

- Sonar provides project dashboards and widgets; you can create custom portlets or export issues via the REST API (`/api/issues/search`).

- Use `api/measures/component` to fetch metrics programmatically for reporting.

Example to fetch code smell count:

curl -s -u admin:MYTOKEN 'http://18.218.37.139:9000/api/measures/component?component=AngularCalculator&metricKeys=code_smells'

14. Troubleshooting

- **Scanner shows `Project root configuration file: NONE`**: Ensure `sonar-project.properties` exists in project root and you run `sonar-scanner` from that directory.

- **403/Forbidden when downloading binaries**: Try alternate mirrors or download locally and SCP the file to server.

- **Analysis fails with parser errors**: Ensure TypeScript configuration paths are correct and source encoding is UTF-8.

- **No coverage shown**: Verify the coverage lcov file path and property `sonar.javascript.lcov.reportPaths`.

15. Appendix: Commands & files

Key commands

```
# build angular
cd ~/Angular-Calc-App
npm install
ng build --configuration production

# run sonar scanner
sonar-scanner

# verify sonar server
curl -I http://18.218.37.139:9000

# install sonar scanner (example)
cd /opt
sudo wget <scanner-zip>
sudo unzip <zip>
sudo mv <dir> sonar-scanner
echo 'export PATH=$PATH:/opt/sonar-scanner/bin' | sudo tee
/etc/profile.d/sonar-scanner.sh
source /etc/profile.d/sonar-scanner.sh
```

Example files

- `angular.json` (ensure no deprecated flags like `extractCss`)
- `sonar-project.properties` (sample included above)
- `Jenkinsfile` (pipeline snippet above)
