SONARQUBE LAB – Node.js

Version 1.0 DEC 2025

PREPARED BY
Kushar Raj Kashyap

Contents

Essential Prerequisites

Before starting the lab, ensure you have the following installed:

- **Node.js and npm**: Required for your project and the SonarScanner.
- **Docker** (Recommended): For an easy local setup of the SonarQube server.
- **A Node.js Project**: An existing project (or a lab project like **NodeGoat**) with a package.json file.
- **Jest/Other Test Runner** (Optional but Recommended): To generate code coverage reports.

Lab Setup Steps

Follow these steps for a complete SonarQube lab setup:

1. Set up the SonarQube Server

The easiest way to get the server running locally is with Docker:

1. Run SonarQube with Docker:

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

   This pulls and runs the Long-Term Support (LTS) version of SonarQube, accessible at http://localhost:9000.

2. Log In:

Open http://localhost:9000 in your browser. The default credentials are **Username: admin**, **Password: admin**. You will be prompted to change your password upon first login.

2. Create a Project and Generate a Token

1. Create a Project:

   Log into the SonarQube dashboard, navigate to **Projects** (or use the **+** button in recent versions), and select **Create Project**.

2. Enter Details:

1. Provide a **Project Key** (e.g., my-node-app) and a **Display Name**.
2. **Generate a Token**: When prompted, choose **Locally** for the analysis, then **Generate a Token**. Give the token a name (e.g., sonar-lab-token) and **save the generated value**. This token is used for authenticating the analysis.

3. Configure the Node.js Project

Navigate to your Node.js project's root directory.

1. **Install SonarQube Scanner**: You can use the sonar-scanner CLI tool or the recommended npm package for Node.js projects:

   Bash code

```
npm install --save-dev sonarqube-scanner
```

2. **Create sonar-project.properties**: Create a file named sonar-project.properties in your project's root directory and add the necessary configuration. Replace the placeholder values with your own:

```
# Project details
sonar.projectKey=my-node-app
sonar.projectName=My Node.js Application
sonar.projectVersion=1.0

# Source code configuration
sonar.sources=src # Directory containing your source code
sonar.exclusions=**/node_modules/**,**/*.test.js,**/*.spec.js

# LCOV report for code coverage (generated in the next step)
sonar.javascript.lcov.reportPaths=coverage/lcov.info

# SonarQube server details (use token for production)
sonar.host.url=http://localhost:9000
sonar.login=YOUR_GENERATED_TOKEN
```

Alternatively, you can configure the scanner using a JavaScript file and the sonarqube-scanner npm package.

3. **Configure Code Coverage (Optional but recommended)**: If you are using Jest, ensure it is configured to output an **LCOV** format report.
    1. Install dependencies:

    ```
    npm install --save-dev jest
    ```

    2. Add a jest.config.js file (or configure in package.json) to include the LCOV reporter:

```
// jest.config.js
module.exports = {
  collectCoverage: true,
  coverageDirectory: "coverage",
  coverageReporters: ["lcov", "text"], // lcov is key
};
```

3. Run tests to generate the report:
   npm test -- --coverage

4. **Run the Sonar Scanner**
   1. **Add an npm script**: For convenience, add a script to your package.json file. If you are using the installed npm package sonarqube-scanner (aliased as @sonar/scan), the script would look like this:

```
"scripts": {
  "sonar": "sonar-scanner"
}
```

   (Note: If you installed the package via npm install -g @sonar/scan, you can run sonar directly. If you installed the CLI, ensure it's in your PATH).
   2. **Execute the Analysis**: Run the scan from your project's root directory:

Bash Code

```
npm run sonar
```

Or

Bash Code

```
sonar-scanner
```

5. **View the Analysis Results**

After the scan completes successfully, go back to your SonarQube dashboard (http://localhost:9000).

Your project should now appear, and you can click on it to see the static code analysis report, including bugs, vulnerabilities, code smells, and coverage data.