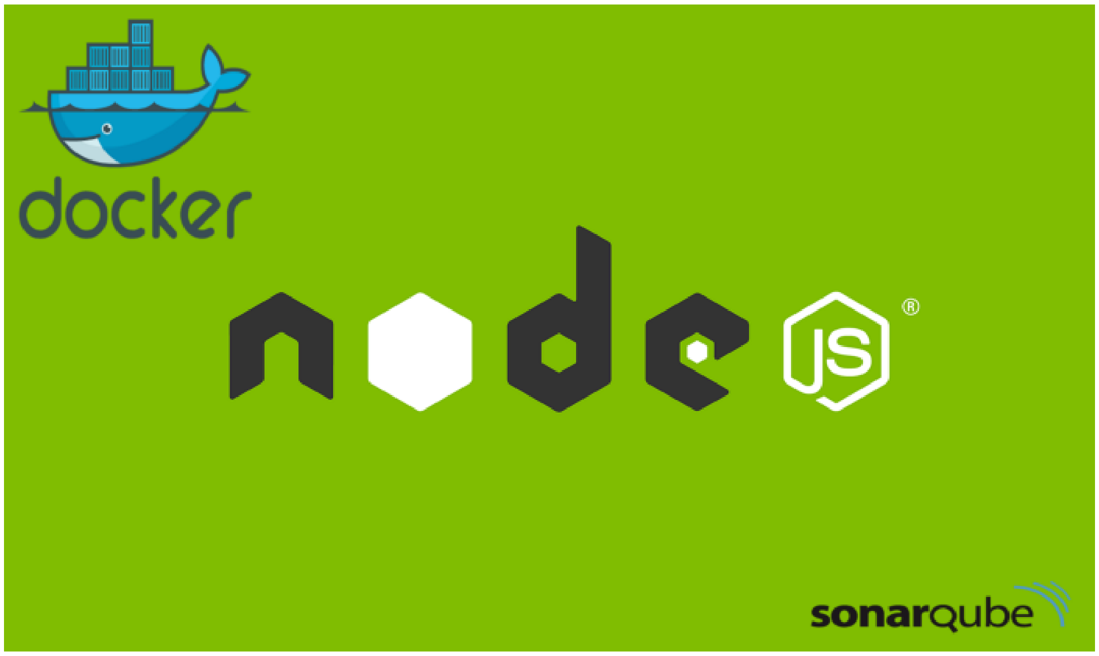
Nodejs Code Evaluation Using Jest, SonarQube and Docker

Rafael Dias Follow
Jun 18, 2020 · 4 min read ★



In this article, we talk about a basic example using Nodejs, Express, Docker, Jest and Sonarqube.
Using the wikipedia explanation "**SonarQube is an open source platform developed by SonarSource for continuous code quality inspection, to perform automatic reviews with static code analysis to detect bugs, code odors and security vulnerabilities in over 20 languages. programming.**"
For this tutorial, we'll need:

- Node/npm
- Docker

With node and docker installed, let's start the project

Starting the project

Creating project folder and browsing

```
mkdir NodeSonarExample
cd ./NodeSonarExample
```

Starting the project

```
npm init -y
```

Installing dependencies

In this session, we will install the dependencies and development dependencies for the project.

1. **Express** which allows http requests, widely used in MVC and Restfull applications.
2. **Jest** is used to perform unit testing.

```
npm install — save express jest
```

1. **sonarqube-scanner** is necessary to scan JS code very simply, without needing to install any specific tool or (Java) runtime.
2. **jest-sonar-reporter** is a custom results processor for Jest. The processor converts Jest's output into Sonar's generic test data format.
3. **supertest** we can test http requests for express routes

```
npm install -D sonarqube-scanner jest-sonar-reporter supertest
```

Docker Image SonarQube

Let's start sonarqube by creating the **docker-compose.sonar.yml** file.

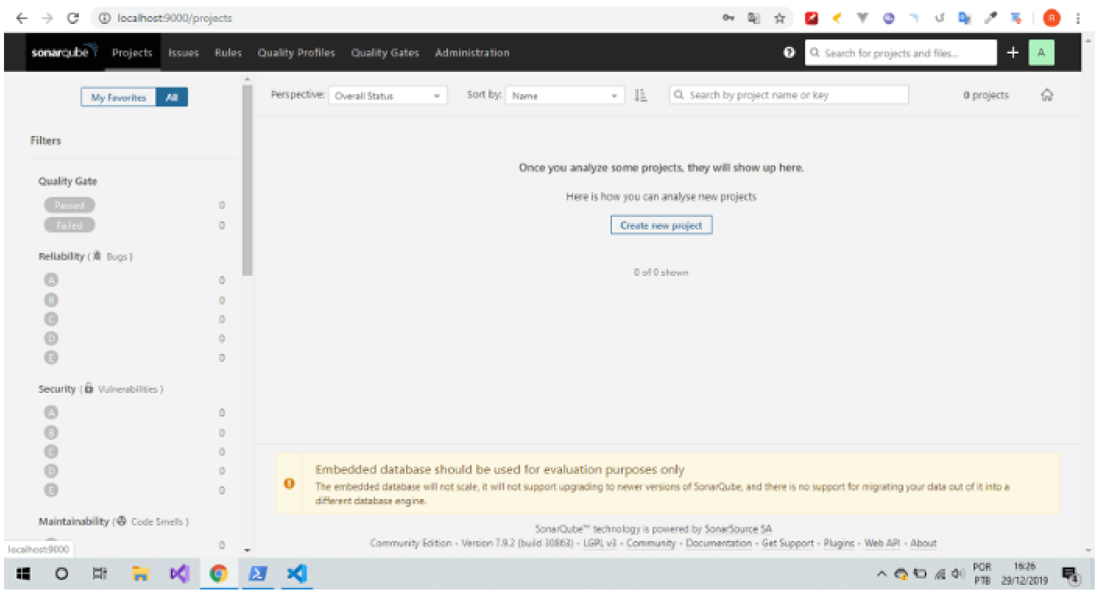
```
version:  '3'
services:
    sonarqube:
        container_name:  sonarqube
        image:  sonarqube:latest
        ports:
            -  "9000:9000"
            -  "9092:9092"
```

and execute the file with the command:

```
docker-compose -f docker-compose.sonar.yml up -d
```

With sonarqube running, navigate to sonarqube address and authenticate using the default account
login: admin
password: admin

Authenticated, you will notice that there is no project pipeline created as shown in the image below



Once you analyze some projects, they will show up here.

Here is how you can analyse new projects

0 of 0 shown

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

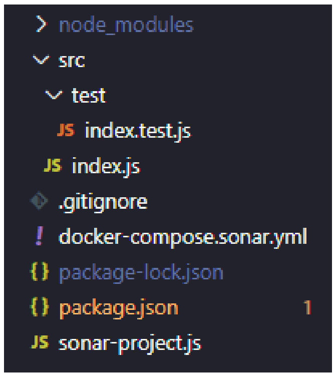
SonarQube™ technology is powered by SonarSource SA

Community Edition - Version 7.9.2 (build 30863) - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API - About

Simple project example

In this session, I will show the project structure, and all the code involved

Project structure



```
> node_modules
v src
  v test
    index.test.js
  index.js
.gitignore
docker-compose.sonar.yml
package-lock.json
package.json                    1
sonar-project.js
```

Code

file: src/index.js

```
const express =  require('express');
const app =  express();
const port =  process.env.PORT  ||  8080

// Route to be tested
app.get('/', (req, res) => {
    return res.status(200).json({ nome:  'Rafael Dias' });
});

// Application running on the door
let server = app.listen(port, () => {
    console.log(`Application running on ${port}`);
});

module.exports  = server;
```

file: sonar-project.js

```
const sonarqubeScanner =  require('sonarqube-scanner');
sonarqubeScanner(
    {
        serverUrl:  'http://localhost:9000',
        options : {
            'sonar.sources':  'src',
            'sonar.tests':  'src',
            'sonar.inclusions'  :  '**', // Entry point of your
code
            'sonar.test.inclusions':
'src/**/*.spec.js,src/**/*.spec.jsx,src/**/*.test.js,src/**/*.test
.jsx',
            'sonar.javascript.lcov.reportPaths':
'coverage/lcov.info',
            'sonar.testExecutionReportPaths':  'coverage/test-
reporter.xml'
        }
    }, () => {});
```

Include these lines in your package.json file

file: package.json

```
{
    .
    .
    .
    "scripts": {
        "sonar":  "node sonar-project.js",
        "test":  "jest --coverage"
    },
    "jest": {
        "testEnvironment":  "node",
        "coveragePathIgnorePatterns": [
            "/node_modules/"
        ],
        "testResultsProcessor":  "jest-sonar-reporter"
    },
    "jestSonar": {
        "reportPath":  "coverage",
        "reportFile":  "test-reporter.xml",
        "indent":  4
    }
    .
    .
    .
}
```

With the project created, just run

```
node src/index.js
```

With the project running, open the browser and navigate to http://localhost:8080/ the expected return is

```
{ name: 'Rafael Dias' }
```

Now let's go to automated testing to perform sonarqube test coverage

Automated test

let's create a test file. So we will import the index.js file and supertest to get the get request for route '/'.
In the end it is necessary to close the open server connection for the test to be terminated

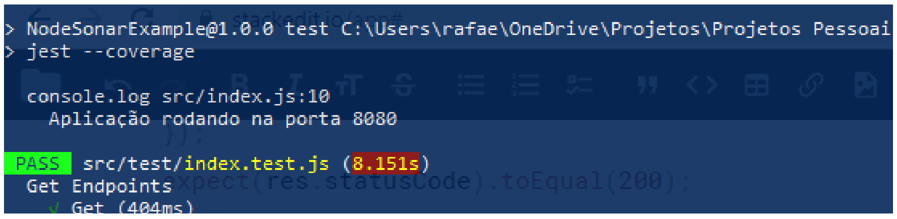
file: src/test/index.test.js

```
const request =  require('supertest')
const server =  require('../index')

describe('Get Endpoints', () => {
    it('Get', async (done) => {
        const res =  await  request(server)
        .get('/')
        .send({
            userId:  1,
            title:  'test is cool',
        });
        expect(res.statusCode).toEqual(200);
        expect(res.body).toHaveProperty('nome');
        done();
    })
})
afterAll(async  done  => {
    // close server conection
    server.close();
    done();
});
```

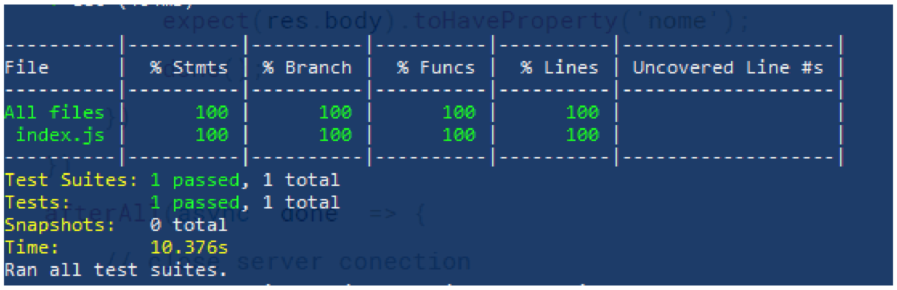
To perform the tests, it is necessary to execute the command

```
npm run test
```

The test results should be successful as in the image below:

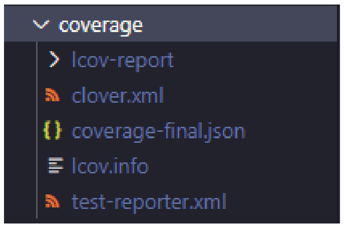


```
> NodeSonarExample@1.0.0 test C:\Users\rafae\OneDrive\Projetos\Projetos Pessoai
> jest --coverage

  console.log src/index.js:10
    Aplicação rodando na porta 8080

 PASS  src/test/index.test.js (8.151s)
  Get Endpoints
    √ Get (404ms)
```



```
----------|----------|----------|----------|----------|-------------------|
File      |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
----------|----------|----------|----------|----------|-------------------|
All files |      100 |      100 |      100 |      100 |                   |
 index.js |      100 |      100 |      100 |      100 |                   |
----------|----------|----------|----------|----------|-------------------|
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        10.376s
Ran all test suites.
```

After all tests are successfully executed, a folder named “**coverage**” will be generated.

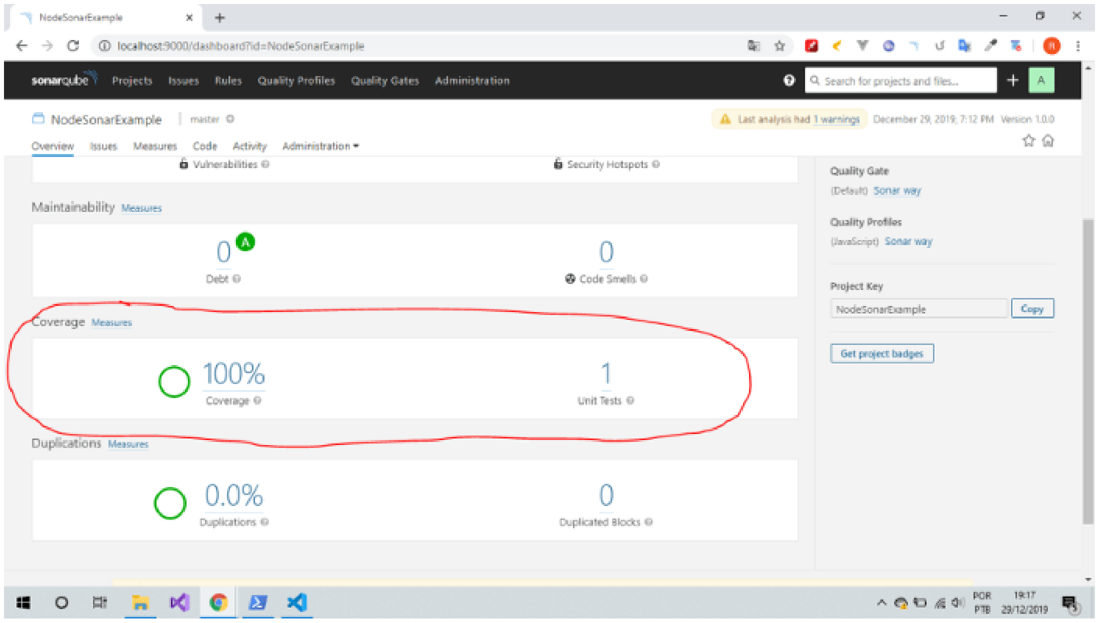


```
coverage
  lcov-report
  clover.xml
  coverage-final.json
  lcov.info
  test-reporter.xml
```

Coverage folder files are being referenced in **sonar-project.js**
finally, the command must be executed

> npm run sonar

This command is responsible for executing the pipeline and committing SonarQube
As a result, you have 100% of your code covered per test, by default you need at least 80%



Node Nodejs Docker Sonarqube Jest

About Help Legal