

SonarQube

Static code analysis is the analysis of applications performed without executing them, in contrast with dynamic analysis, which is performed on applications during their execution.

[Documentation](#)

Whats SonarQube about ?

SonarQube is an open-source platform that is developed by [SonarSource](#). It continuously inspects code quality by performing automatic reviews with static analysis of code. It detects bugs, code smells, security hotspots, and vulnerabilities.

What is Static Code Testing?

The term is usually applied to analysis performed by an automated tool, with human analysis typically being called "program understanding", program comprehension, or code review. The last of these, software inspection and walkthroughs are also used. In most cases the analysis is performed on some version of a program's source code, and, in other cases, on some form of its object code.

Jest vs SonarQube?

Testing with jest is called Dynamic Testing. We are expecting a predictable outcome with the code we wrote. For example, expect a Title on the screen when landing on the Home page.

With SonarQube, we are testing static code like functions, variables, structures, etc. It goes through all the code and checks for security risks, vulnerabilities, bugs, etc. How it differs from JEST is, even with bugs and security risks we can expect a successful execution of our code without knowing the harm it may cause in the future. SonarQube can help developers tackle the issues before it goes to production.

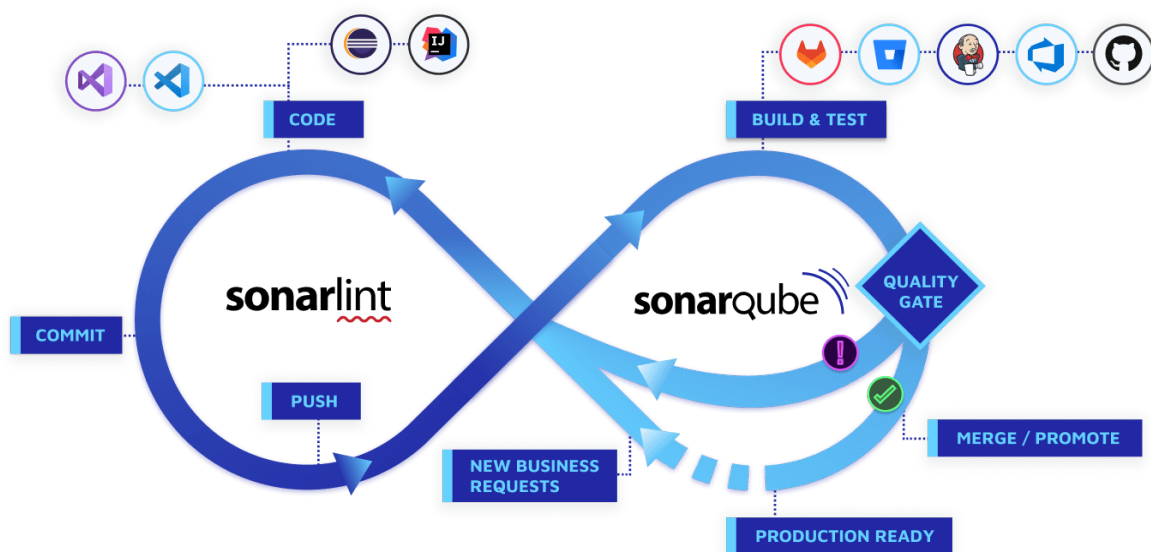
Issues SonarQube can prevent?

- **Bugs:** An issue that represents something wrong in the code.
- **Code smell:** A maintainability-related issue in the code. It makes further changes harder and possibly will introduce additional errors with future changes.
- **Security hotspot:** Security-sensitive pieces of code that need to be manually reviewed.
- **Vulnerability:** A security-related issue that represents a backdoor for attackers.

But why SonarQube?

SonarQube has 6 million developers. It helps to build high-quality code quickly and systematically.

- It helps to write clean code and keeps maintenance time and costs to a minimum.
- Clean code makes the work environment enjoyable and satisfying.
- Clean code empowers developers to focus on solving business problems.
- It ensures software is robust and secure with the right checks at the right place and time. It minimizes risks and maximizes the product's reputation.



Prerequisite Testing Tools in React

Well for React Testing SonarQube uses the Jest Coverage report to examine all the covered code. React App comes with a JEST preinstall, so no need to worry about installing anything as in your project. Next, I will go through how to use SonarQube for your project.

Installation and Usage

SonarQube with React-App

Prerequisites

Make sure you have JAVA 11 or higher installed on your preferred device. **SonarQube will not work with Java.**

Click here to [download](#)

Downloading SonarQube zip

1. Go to this [site](#).
2. Choose the SonarQube community edition. (the free one)
3. Unzip the SonarQube file and move it to the preferred location (I will use /opt/ in Linux)

Version: 9.7.1 | Release: November 2022 | [Getting Started](#) | [Release Notes](#) | [Upgrade Notes](#) | [Available From DockerHub](#)

Community
EDITION

Used and loved by 200,000+ companies
FREE & OPEN SOURCE

[Download for free](#)

All the following features:

- ✓ Static code analysis for 17 languages
Java, C#, JavaScript, TypeScript, CloudFormation, Terraform, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML and VB.NET
- ✓ Detect Bugs & Vulnerabilities
- ✓ Review Security Hotspots
- ✓ Track Code Smells & fix your Technical Debt
- ✓ Code Quality Metrics & History
- ✓ CI/CD Integration
- ✓ Extensible, with 50+ community plugins

Developer
EDITION

Built for developers by developers

[Download](#)

Community Edition plus:

- ✓ C, C++, Obj-C, Swift, ABAP, T-SQL, PL/SQL support
- ✓ Detection of Injection Flaws in Java, C#, PHP, Python, JavaScript, TypeScript
- ✓ Analysis of feature and maintenance branches
- ✓ Pull Request decoration for:
 - GitHub
 - Bitbucket
 - Azure DevOps
 - GitLab

[Start Free Trial](#)

Enterprise
EDITION

Designed to meet Enterprise Requirements

[Download](#)

Developer Edition plus:

- ✓ Portfolio Management & PDF Executive Reports
- ✓ Project PDF reports
- ✓ Security Reports
- ✓ Project Transfer
- ✓ Parallel processing of analysis reports
- ✓ Support for Apex, COBOL, PL/I, RPG, VB6

[Start Free Trial](#)

Data Center
EDITION

Designed for High Availability Requirements

[Download](#)

Enterprise Edition plus:

- ✓ Component redundancy
- ✓ Data resiliency
- ✓ Horizontal Scalability

[Discover Now](#)

Below are the steps to run the SonarQube instance.

Running SonarQube Instance

- Go to your terminal and navigate to the directory where you have placed the SonarQube unzipped file.

```
→ linux-x86-64 cd /opt/  
→ /opt
```

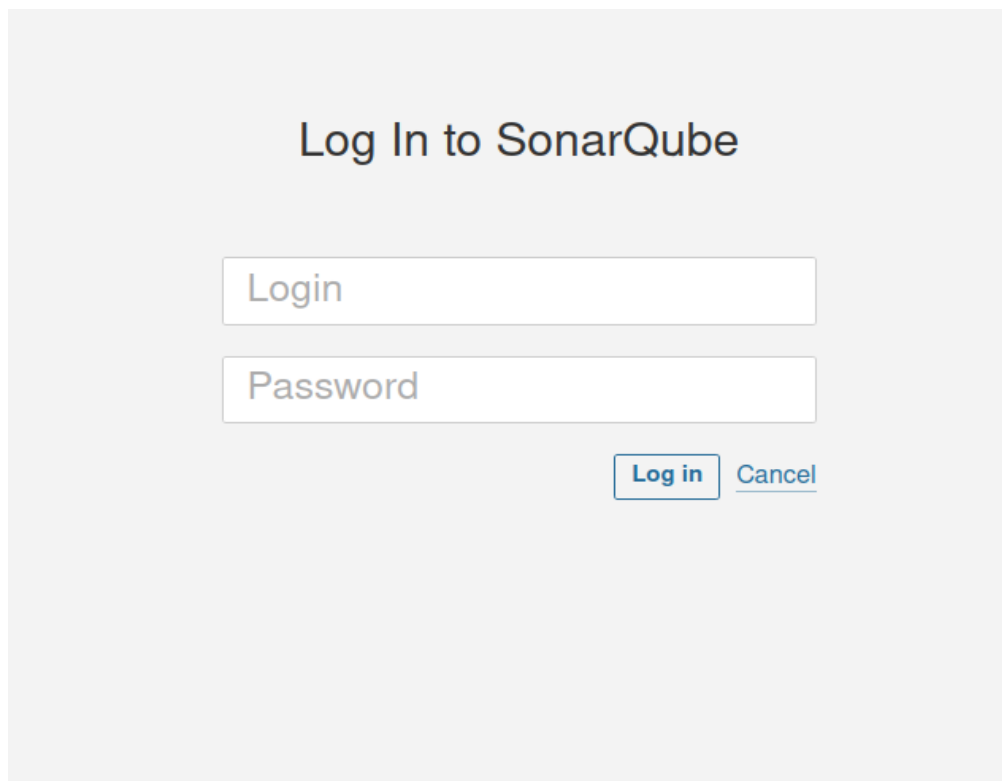
- Note that you should know which operating system you are working on.

```
linux-x86-64  macosx-universal-64  windows-x86-64  
winsw-license  
// I will use linux-x86-64
```

- Now use the below command to run SonarQube Instance

```
→ /opt sonarqube/bin/linux-86-64/sonar.sh console
```

- An instance will start running on `http://localhost:9000`
 - Note - do not close the terminal where you have run the instance.
- Go to `http://localhost:9000` and you will be greeted with this screen.



The image shows the SonarQube login interface. It has a light gray background. At the top, the text "Log In to SonarQube" is centered in a dark gray font. Below this, there are two white input fields with gray borders. The first field is labeled "Login" and the second is labeled "Password". Both labels are in a light gray font. At the bottom right of the form, there are two buttons: a blue button labeled "Log in" and a blue link labeled "Cancel".

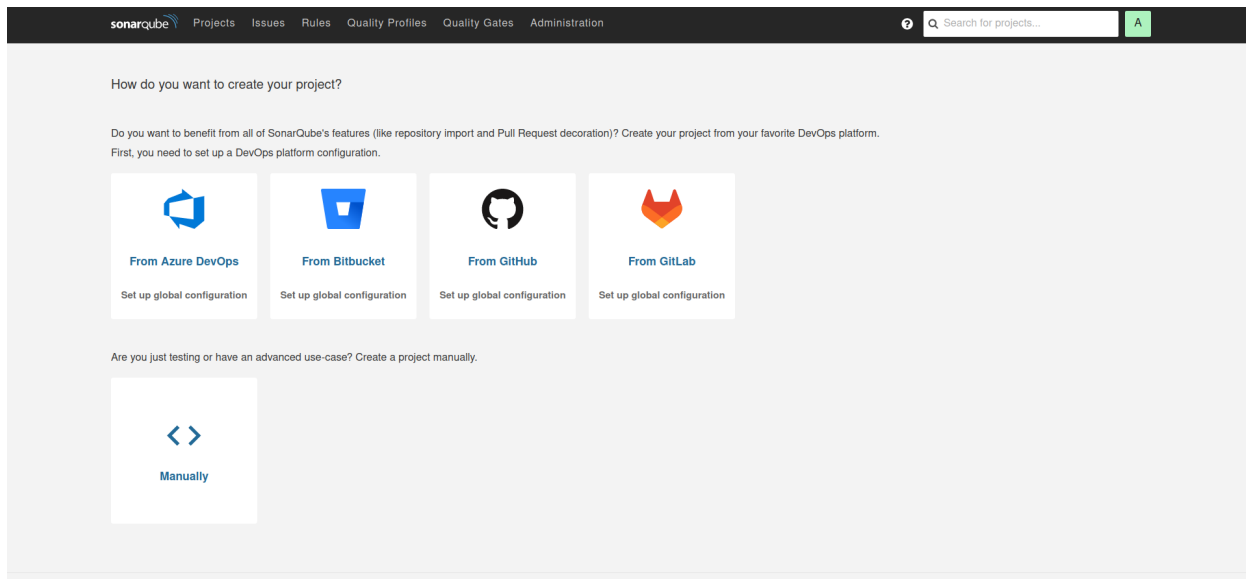
Setting up SonarQube for the project

Login to SonarQube

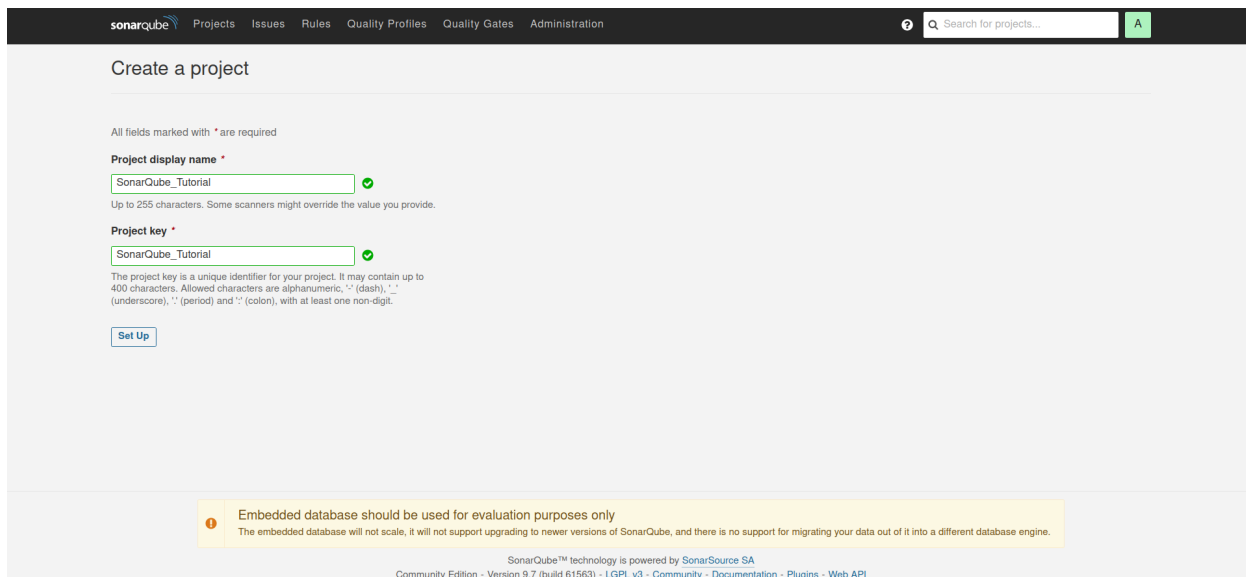
If it is your first time logging into SonarQube use Login as **admin** and Password as **admin**
After logging in you will be first greeted with a reset password screen, choose the password you want and reset it.

Adding Project to SonarQube

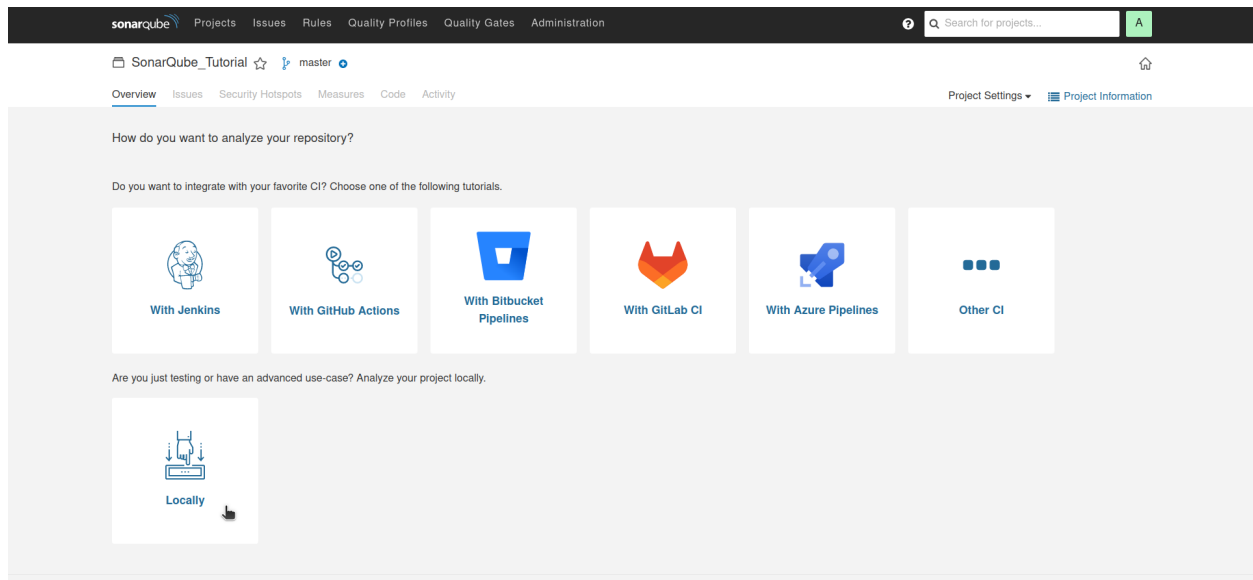
- Choose **Manually** option on the screen



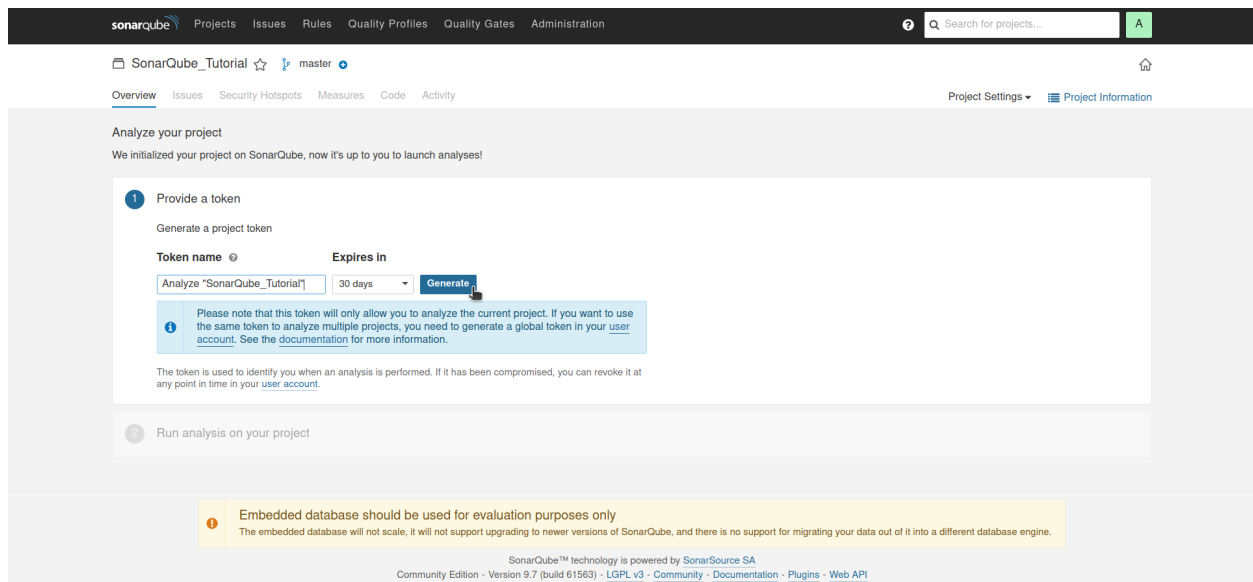
- Write the project name and click on setup



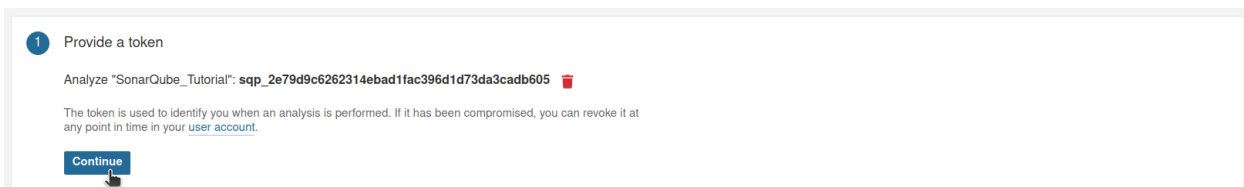
- Click on **Locally** to continue.
 - This is just a basic tutorial with basic testing, that's why we are choosing Locally



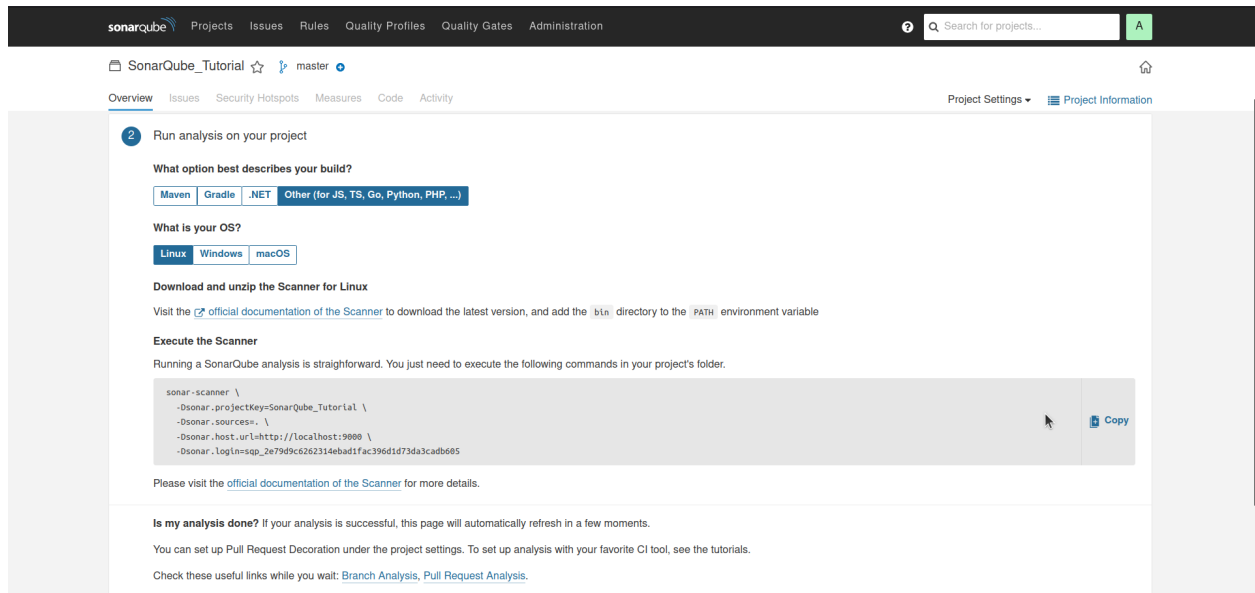
- Now, we will create a Project token. This is used to identify your project from others. We can also set up a Global token to be used in multiple projects but it is not covered in this tutorial.



- Set up how long you want this token to work.
- Click on Generate after that.
- Below is a screenshot of the token generated, click on continue after that



- Now choose the language/technology your project is based on. I have gone with TS or Typescript. Then choose your operating system, I choose Linux as my primary OS for this project.



- Copy the commands given, we will be needing those when running SonarQube on our project. Yours will be different, the codes below are just for reference. Don't use it.

```
npx sonar-scanner \
-Dsonar.projectKey=SonarQube_Tutorial \
-Dsonar.sources=. \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.login=sqp_2e79d9c6262314ebad1fac396d1d73da3cadb605
```

Cool so, we now have set up sonarqube for our project.

Setting up project to SonarQube

Now we are pretty much done with the SonarQube setup. These are the final steps to configure it to our project locally.

First, open your project directory and open in your favorite IDE or code editor, I'm using VS code in this tutorial.

Follow the steps below for an errorless first-time run.

- Open the internal terminal in VScode or cd to your project.
- Before doing anything we need two packages as dev dependencies for our project. Copy and paste the code given below on your terminal and hit enter.

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

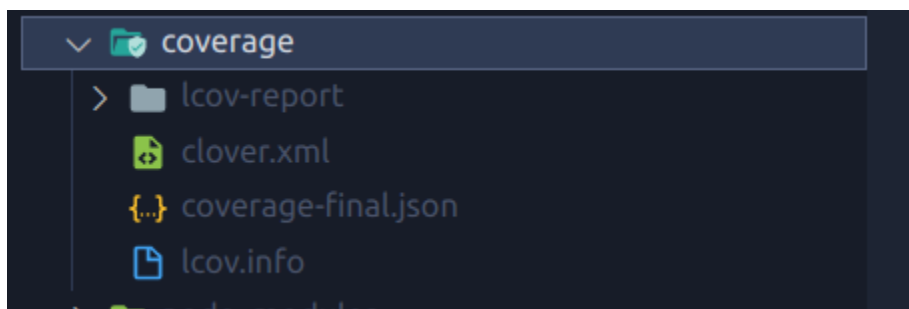
- Check for these dev dependencies on the package. json file

```
"devDependencies": {  
  "jest-sonar-reporter": "^2.0.0",  
  "sonarqube-scanner": "^2.8.2"  
}
```

- We need to run a coverage report using jest. The coverage report will allow SonarQube to check all the files. JEST comes with every react app. So, all we have to do is run this simple command on the terminal.

```
npm run test -- --coverage . --watchAll=false
```

- After the run will be able to see a file called coverage on the project root directory.



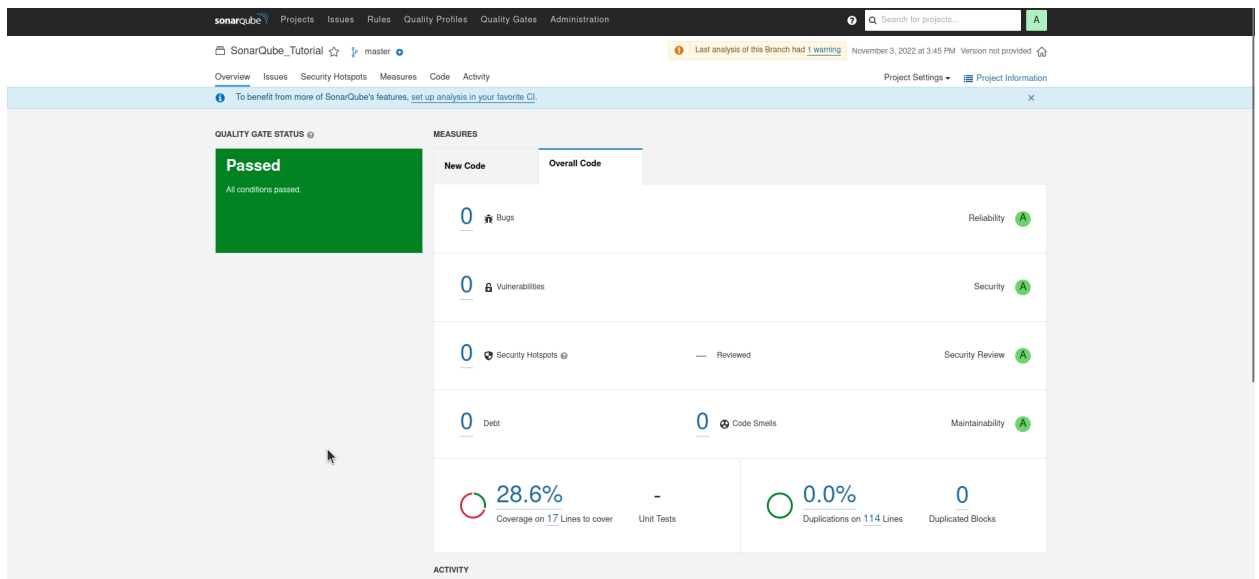
- Now run the SonarQube command that we copied before
 - Make you use **npm** and the rest of the command
 - Check for your project.key and .login which is your project to be accurate


```

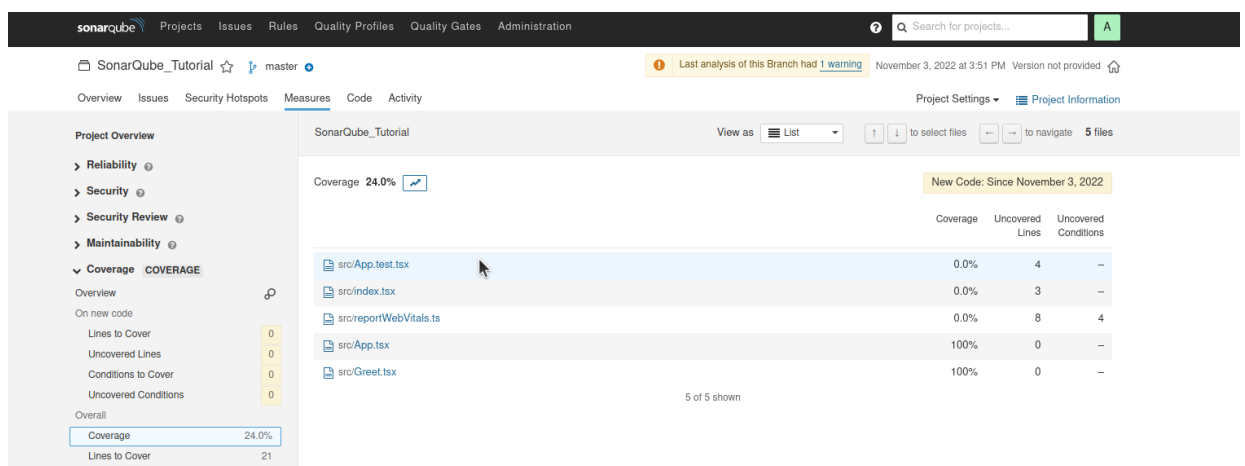
npx sonar-scanner \
  -Dsonar.projectKey="your project key" \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login="your project token"

```

- If everything goes well, you now have a new file called `.scannerwork` on the root directory. Put it in a `.gitignore` file.
- Check your `http://localhost:9000`, it will now show something like this



- Click on that 28.6% link (you might have something else). And this will show up.



- The above coverage report also contains a .test.tsx file, which is not required in Static code testing. Let's fix this by creating a sonar-project.properties file in the root directory.

- Create a new file called **sonar-project.properties** in the root directory.
- Copy n paste the command given below.

```
sonar.projectKey="your project key"
sonar.sources=.
sonar.host.url=http://localhost:9000
sonar.test.inclusions=*.test.tsx,*.test.ts
sonar.exclusions=**/__tests__/**,**.test.tsx**,**.test.ts**
sonar.typescript.lcov.reportPaths=coverage/lcov.info
```

- This property file will exclude any file with the extension .test.ts, .test.tsx, and folders with the name __tests__.
- These properties can be set up on the SonarQube instance also.
- Make you use your project key

- Now let's again run the SonarQube with the command below and check for results now.

```
npx sonar-scanner \
-Dsonar.projectKey="your project key" \
-Dsonar.login="your project token"
```

Here you can see, I didn't use as many lines of code as before when running on the first try.

The screenshot shows the SonarQube web interface for a project named 'SonarQube_Tutorial'. The 'Coverage' section is active, showing an overall coverage of 28.6%. A table lists the coverage for individual files:

File	Coverage	Uncovered Lines	Uncovered Conditions
src/index.tsx	0.0%	3	—
src/reportWebVitals.ts	0.0%	8	4
src/App.tsx	100%	0	—
src/Greet.tsx	100%	0	—

The sidebar on the left shows the 'Coverage' section expanded, with 'Overall' coverage of 28.6% and 'On new code' coverage of 0.0%.

It now does not contain any test files.

- You can see how SonarQube has gone through all the files from the coverage report and will sniff out all the potential issues and bugs mentioned on page 1.

The screenshot displays the SonarQube web interface for a project named 'SonarQube_Tutorial'. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. A search bar is present on the right. The main header shows the project name, a star icon, a branch selector set to 'master', and a warning message: 'Last analysis of this Branch had 1 warning' dated November 3, 2022 at 4:07 PM. Below this, tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity are visible. The left sidebar contains a 'Project Overview' section with expandable categories: Reliability, Security, Security Review, Maintainability, Coverage (selected), Duplications, Size, Complexity, and Issues. Under the 'Coverage' section, a table shows metrics for 'On new code': Lines to Cover (0), Uncovered Lines (0), Conditions to Cover (0), and Uncovered Conditions (0). An 'Overall' section shows Coverage at 100%, Lines to Cover at 5, Uncovered Lines at 0, Line Coverage at 100%, Tests at 0, Errors at 0, Failures at 0, Skipped at 0, and Success at 100%. The main content area shows the 'src/Greet.tsx' file with a code editor displaying a JavaScript function 'Greet()' and its corresponding HTML output. A warning message at the bottom states: '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.' The footer mentions 'SonarQube™ technology is powered by SonarSource SA' and provides links for Community Edition, Version 9.7, and various documentation and API links.

- Whenever you create a new file and want to see it in SonarQube be sure to run coverage before. Otherwise, it will not show them on its instance.
- You also don't have to run the long command every time you want to test and check the SonarQube, just create a script to run on the package.json file.

Moving On

This covers the very basics of the SonarQube, it is a very powerful and good tool to use for code quality and to find potential issues with the code and it's highly scalable for industrial standards. SonarQube also provides feature for rules, so we can have a standard pattern on the project. It also checks for issues in different branches simultaneously. All in all, it's a good tool to use but requires good learning to use it at its full potential.