

SonarQube Tutorials – Step by Step Guide - devopsuniversity.org

Thursday, May 1, 2025 3:29 PM

Clipped from: <https://devopsuniversity.org/sonarqube-setup-guide/>

Code Quality

Code quality is an extremely important parameter for software quality deliverables and affects the overall success of a software organization. It is critical to ensure utmost quality of delivered software as code quality defines how safe, secure and reliable the software is.

Ensuring high quality of software should definitely be the goal throughout any development process. It can surely be achieved if everyone in the team is determined to deliver quality and holds himself accountable for it.

Code quality depends on a number of different attributes and requirements, primarily determined and prioritized by the business needs.

Essentially, a good quality code

- Does what it should.
- Follows a consistent style.
- Is easy to understand.
- Has been well-documented.
- Can be tested.

Testing alone cannot ensure good code quality as testing will never find every error in the code.

The quality of code is important because if the code is of low quality, it might introduce safety or security risks. The results can be catastrophic or fatal if the software fails due to security violations or safety flaws.

Key Aspects to Measure

The key traits to measure for higher quality are:

- **Reliability**
Reliability measures the system's probability to run without failure over a specific period of operation. It relates to the number of defects and availability of the software.
- **Maintainability**
Maintainability measures how easy is the system maintenance. It relates to the size, consistency, structure, and complexity of the codebase. Ensuring that source code is maintainable depends on a number of factors, such as testability and understandability.
- **Testability**
Testability measures how well the software can be tested. It relies on the extent to which you can control, observe, isolate, and automate testing.
- **Portability**
Portability measures how usable or portable the same software is

across different environments. Platform independence is the key factor.

- **Reusability**

Reusability measures whether existing code can be used again or not. Modular and loosely coupled code, is easy to reuse.

How to Improve Code Quality

Once, the code quality is measured, we can take steps to improve it.

Let us discuss some ways you can improve code quality:

- **Coding Standards**
using a static code analyzer helps.
- **Code Analysis before Code Reviews**
Quality should be a priority from the very start of development and that's why it's important to analyze code before code reviews begin. In fact, it's best to analyze code as soon as it's written. The earlier the defects are detected, the easier and faster it is to resolve them.
- **Code Review Best Practices**
Good code reviews help improve overall software quality. We will discuss Code reviews in detail shortly.
- **Refactor Legacy Code**
One way to improve the quality of an existing codebase is through refactoring.

Code Reviews

Code Review is a phase of the software development process in which the authors of the code use a system of review to examine the source code. The code is usually reviewed by another programmer who inspects the code for mistakes, irregular formatting, or inconsistencies with the system requirements that may create larger issues during the integration process.

Code review is the most commonly used procedure for validating the design and implementation of features.

Code Review is an instrumental component of an efficient software solution. It creates consistent code and enables better QA testing and seamless software integration. By diminishing mistakes and fixing errors earlier at the review stage, programmers become more efficient in designing customized solutions in line with the specifications to the client.

Code Reviews primarily check for four things:

1. Code issues
2. Consistency in formatting with the overall software design
3. Quality of documentation
4. Consistency with coding standards and project requirements

The major purpose of the review process is to create a collective responsibility of the software's quality through knowledge transfer by examining the source code collectively as a team.

Another advantage of maintaining consistent code standards is that it is easier for specialists and testers to understand the source code.

Source code that is constantly under review allows developers to learn more reliable techniques and best practices leading to robust software for seamless integration and functionality.

Code review can be performed in two levels:

Peer review

It is mainly focused on functionality, design, and the implementation and usefulness of proposed fixes for stated problems.

Peer reviewers look for the following in the code:

- Feature Completion
- Potential Side Effects
- Readability and Maintenance
- Consistency
- Performance
- Exception Handling
- Simplicity
- Reuse of Existing Code
- Test Cases

External review

It is different than peer reviews as it addresses different kinds of issues. Specifically, external reviews focuses on how to enhance code quality, promote best practices, and remove “code smells.”

External reviewers look for following in the code:

- Readability and Maintenance
- Coding Style
- Code Smells

Why Reviewing Code is important?

There are several reasons why performing a code review is a necessary part of an efficient and complete development process.

- **It limits risks**
This is probably the foremost reason of all. We can limit the risk due to unnoticed mistakes and manual errors, by someone double-checking our code. Trust me, double-checking is always a great idea.
- **It dramatically improves code quality**
It's all about making code more efficient. While working together on code examination, each team member can come up with smarter solutions, applicable design patterns and improvisation techniques to reduce complexity as well as enhance the performance of the project, resulting in high quality deliverables.
- **It makes everyone better**
Everyone can grow and learn together. Through code review feedback, developers become aware of possible issues and areas

for improvement. The reviewers also gain as they learn new things from code, and can find out new solutions, applicable to their own work also.

- **It helps being familiar with the project**

When a team works on a project, it's highly unlikely that every developer is aware of what other developers are working on. Doing code reviews helps team members become familiar with code pieces that they haven't written but might fix a defect on or maintain or test in the future. It definitely promotes knowledge of the codebase across the team, and is likely to speed up future development.

In the DevOps world where so much code lives on GitHub, code reviewing typically goes hand-in-hand with "pull requests". A pull request is a request to introduce changes to a code repository using a distributed version control system. It works by "pulling" the original code, applying changes, then submitting a request to merge the changes in.

Having a regular and efficient code reviewing process in place is extremely useful and important as it helps to maintain high-quality coding standards and foster growth and knowledge sharing.


It is important to understand that asking for a code review is not a sign of weakness or something negative and one should always be open and positive feedback.

Static Code Analysis

Static code analysis is a set of algorithms and techniques used to analyze source code in order to automatically find potential errors or poor coding practices.

Static code analysis, also commonly known as "white-box" testing, tests applications in non-runtime environments. It is a proven technique which covers entire codebase and identifies all the vulnerabilities.

The tasks performed in the static code analysis can be divided into 3 categories:

- Detecting errors in programs
 - Recommendations on code formatting
 - Software Metrics computation
- 

There are many static analysis tools available. However, Checkstyle, PMD, and Sonarqube are well-known and commonly used in the projects.

PMD

PMD is a static code analysis tool. It detects programming errors such as unused variables, empty catch blocks, unnecessary object creation, and so forth. It's majorly concerned with Java and Apex, but supports six other languages.

PMD is a static code analysis tool capable of detecting many potential defects and unsafe or non-optimized code (bad practices). PMD focuses more on preemptive defect detection ensuring good practices are followed.

It comes with a rich and highly configurable set of rules, and you can easily configure which particular rules should be used for a given project. PMD is very useful when it is integrated into the build process. It can then be used as an enforced quality check, to adhere to a coding standard for your codebase.

PMD can be run as:

- A Maven goal
- An Ant task
- A Gradle task
- From cmd prompt

Checkstyle

Checkstyle is an open source tool that can help enforce coding standards and best practices, with a major focus on coding conventions.

Checkstyle is a development tool that aids programmers adhere to coding standards while writing code. It automates the process of checking Java code sparing the human effort for this not so interesting yet an important task. This makes it ideal for projects that want to enforce a coding standard.

Checkstyle is highly configurable and can be made to support almost any coding standard.

It has the ability to check several aspects of your source code. It can detect class design problems, method design problems, check code layout and formatting issues.

SonarQube

SonarQube is a web-based open source platform by SonarSource, used to measure and analyse the source code quality. Code quality analysis makes your code more reliable and more readable.

It is implemented in Java language and can analyze the code of about 20 different programming languages, including c/c++, PL/SQL, Cobol etc through plugins. More than 50 plugins are available which extend the functionality of SonarQube.

SonarQube can inspect and evaluate anything that affects code base, from minor styling details to critical design errors. Hence, it is a great aid to software application developers in identifying the issues and their impact.



SonarQube is in no way competing with any of the above static analysis tools, but rather it complements and works very well with these tools. In fact, it ceases to work if these static analysis tools (Checkstyle, PMD, and FindBugs) do not exist.

SonarQube's offerings span what its creators call the Seven Axes of Quality:

1. Architecture and Design
2. Unit tests

3. Duplicated code
4. Potential bugs
5. Complex code
6. Coding standards
7. Comments

We will explore this tool in detail in the further sections.

Key concepts in Static Code Analysis

A code review typically starts with the static analysis of the source code using a static analysis tool. Various key statistics are analyzed on the basis of software metrics.

- **Code Smell:**
It is basically an indicator of something fishy in the code – “a hint that something has gone wrong somewhere in your code. Use the smell to track down the problem.” You can resolve issues by tracking down these indicators in your code.
- **Technical Debt:**
It describes the estimated effort to fix all the maintainability Issues / code smells. It is a programming concept that reflects the extra development work, which has to be done if long-term and best solutions are not sought, and things are implemented keeping the ease in mind, for short term.
- **Code coverage:**
It is a measure which describes the degree of which the source code of the program has been tested. It is one form of white box testing which finds the areas of the program not exercised by a set of test cases.
- **Test coverage:**
It is a Software Testing metric used to measure the amount of testing performed by a particular set of tests. It will gather information related to the parts of a program actually executed when running the test suite in order to determine which branches of conditional statements have executed.
Code Bugs: A problem or an issue that shows that something is wrong in the code. It might not break the code right away, but surely will, and may be at the moment when its impact is huge. It has to be fixed.
- **Vulnerabilities:**
It is a cyber-security term that refers to a flaw or loophole in the system that makes it vulnerable to attack. A vulnerability may also refer to any type of weakness in a set of procedures, or in anything that leaves information security exposed to a threat. Minimizing vulnerabilities leaves bare minimum chances of unwanted access to sensitive and secure data by malicious users.

SonarQube

SonarQube Integration is an open source static code analysis tool that has become quite popular in recent times. It can integrate with your existing workflow to enable continuous code inspection across your project branches and pull requests.

SonarQube provides reports for your projects after thorough analysis.

Everything right from minor styling details to critical design errors, is inspected and evaluated by SonarQube, thereby enabling developers to access and track code analysis data continuously, ranging from styling errors, potential bugs, and code defects to design inefficiencies, code duplication, lack of test coverage, and excess complexity.

The Sonar platform analyzes source code from different aspects and drills down to your code layer by layer, moving from the module level down to the class level, giving metric values and statistics, revealing issues in the source code at each level, that need to be addressed.

Your project home page tells you where you stand in terms of quality in a glimpse of an eye. It you an immediate sense of what you have achieved till now and the project status.

Features:

Write Clean Code

- **Overall Health** – Discovered issues can either be Unreachable source code, a Bug, Vulnerability, Code Smell, Coverage or Duplication. Each category has a corresponding number of issues. Dashboard page shows where you stand in terms of quality in a glimpse of an eye.
- **Enforce Quality gate** – To fully enforce a code quality practice across all teams, you need to set up a Quality Gate which is a set of standard conditions the project must meet before it can be released to production.
- **Issue Resolution** – There are five different severity levels of Issues like blocker, critical, major, minor and info. Also, the issues tab has different filter criteria like category, severity level, tag(s), and the calculated time required for the issue rectification.

Handle Multi-Language Projects

SonarQube automatically detects the project language and run corresponding code analyzer for each language.

Centralize Quality – All projects in one place

SonarQube enables organization to estimate and predict project risks as there is a centralized system of storing the code metrics. It simplifies deployment and helps monitor the project status.

Shared rulesets

SonarQube provides the facility to create your own quality profiles and define Sonar Rules shareable across different projects.

Quality Gates

Quality Gates are a powerful feature by SonarQube.

They are a combination of threshold measures set on your project.

A quality gate is the best way to enforce a quality policy in your company.

They are basically a set of conditions that the project must meet before it can be delivered to production.

You can define as many quality gates as you wish to, depending on the project need.

Each Quality Gate condition is typically a combination of:

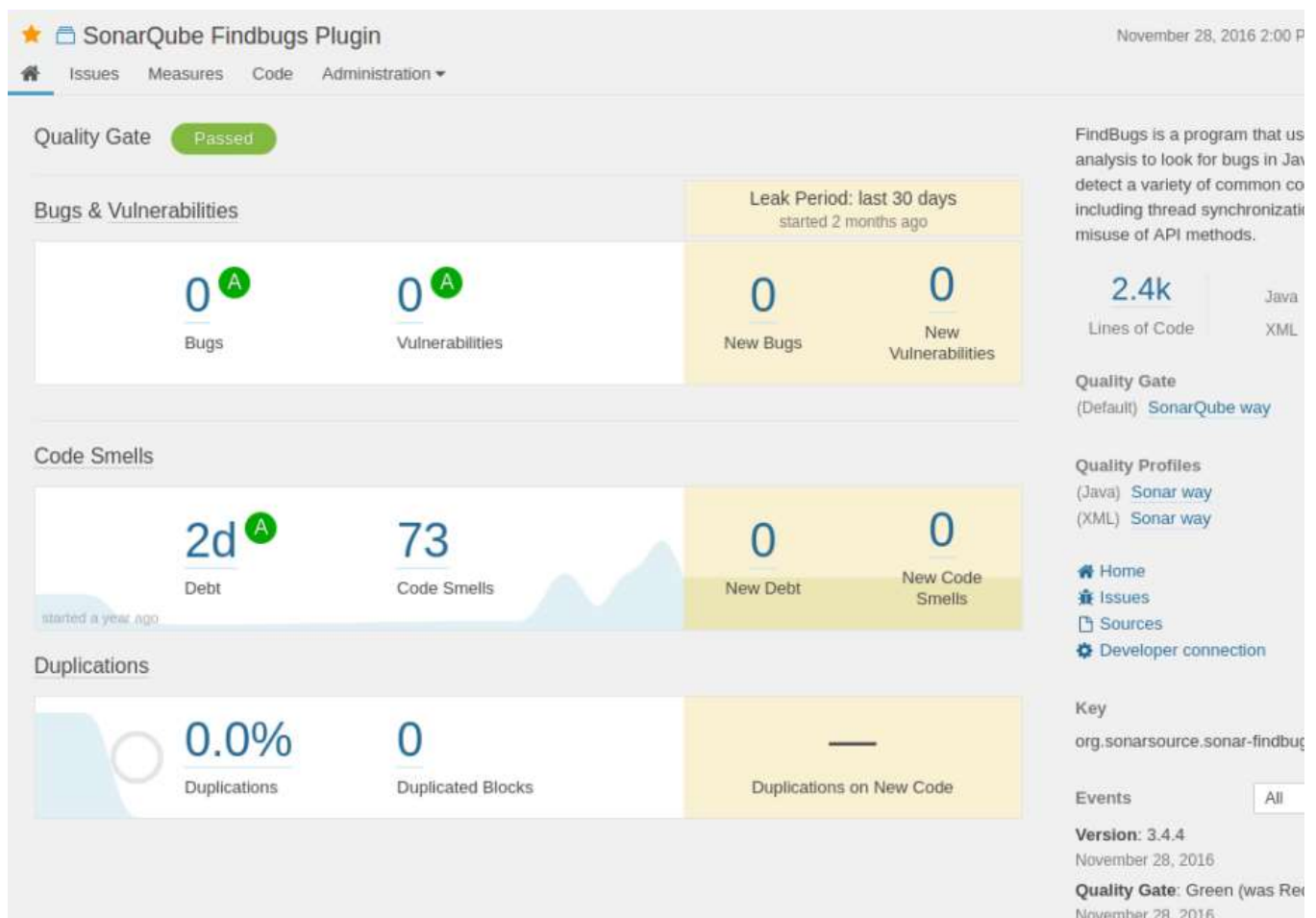
- measure
- period: Value (to date) or New Code (differential value over the New Code period)
- comparison operator
- warning value (optional)
- error value (optional)

For instance, a condition might be:

- measure: Blocker issue
- period: Value
- comparison operator: >
- error value: 0

All projects may not be verified against the same quality gate, as it's not practical that the threshold measures would be the same for all projects.

To manage Quality Gates, go to Quality Gates (top menu bar) in sonarqube. Quality Gates are defined and managed in the Quality Gates page found in the top menu. If your project passes the Quality Gate, this is how it will be displayed.



A failed Quality Gate would look like the one below, highlighting all the bugs and issues found in the project.



How it helps different users in Organization

- **Developers**
Regularly usage of SonarQube helps developers identify the coding standard violations at the time of coding and they learn to adhere to coding standards from the beginning.
- **Technical management**
SonarQube supports easy integration with version control system to track down the details regarding each code update. This also helps to identify a particular developer's performance and his coding practices.
- **Non-technical management**
Non-Technical management will be keen to know and measure code quality. They may not understand the complexity and duplications etc but with the matrix and total statistics about the project health, it becomes easy for them to make key decisions about projects.

SonarCloud : SonarQube as a Cloud Service

SonarCloud is the leading online product for Continuous Code Quality inspection, totally free for open-source projects. It supports all major programming languages.

Clean Code Rockstar Status

Eliminate bugs and vulnerabilities.
Champion quality code in your projects.

Go ahead! Analyze your repo:



GitHub



Bitbucket



Azure DevOps

Free for Open-Source Projects



For projects with closed source code, SonarCloud offers a paid plan to run private analysis.

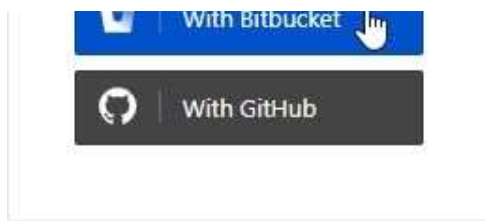
SonarCloud can perform analysis on 20+ different languages. Some of the features are:

- **Ready to use** – You need not worry about anything and just enjoy the service, everything else will be taken care of SonarCloud.
- **Work with latest features** – It always provides the most recent and greatest features of SonarQube with pre-selected plugins.
- **Get started in minutes** – Simply sign up, create an organization for your team, and you are ready to run your builds to get your projects analyzed in minutes.
- **Designed to scale** – Sizing or planning for your future needs is SonarCloud's responsibility and its ensured that the service is able to handle the code analysis of your code at the desired pace.

Depending on which cloud service you use, you can choose from these dedicated integrations:

- GitHub
- Bitbucket Cloud
- Azure DevOps (formerly VSTS)





Using SonarCloud with GitHub:

The SonarCloud GitHub application makes it very simple to pull in new code repositories in SonarCloud, get your pull requests detected with quality issues, and also in team collaboration.

Steps to login for your project using GitHub:

1. On the SonarCloud login page <https://sonarcloud.io/about>, click on the Login button on the extreme right and choose GitHub from the next page.
2. Then, use your GitHub credentials to login with GitHub and connect to SonarCloud using your GitHub account.
3. Click on "Analyze your code" and follow the path to set up a first project
4. You will be asked to install the SonarCloud application on your organization, which will allow you to pick the repository you want to analyze.

Summary

In this document, we have tried to cover why code quality is important and how code reviews help us achieve the same. We also discussed tools available for static code analysis, which is an important part of code reviews.

We explored a very popular static code analyzer, SonarQube in detail and also learnt how to integrate SonarQube with Jenkins – all the required configuration etc.

We also saw how sonarCloud.io, the option available through cloud services, can be used for quality check of our projects.

Hope you enjoyed reading this document and found it helpful. Happy learning!

