

Sonar Qube:

From SonarSource, we have three possible options:

SonarQube - A self-managed tool/server that needs to be installed/hosted.

SonarCloud - Cloud variant of Sonar analyzers - only registration required.

SonarLint - IDE extension. You can also use it with SonarQube and SonarCloud in Connected Mode to catch issues earlier and implement a shift-left approach.

SonarAnalyzer.CSharp—This code analyzer set is delivered via a NuGet package. You can install them in any .NET project and use them for free, even without an extension or paid subscription for any other products.

SonarJava- SonarJava is designed to analyze Java code for bugs, vulnerabilities, code smells, and other issues that could impact code quality and security. By using SonarJava in your development workflow, you can ensure that your Java code adheres to best practices, follows coding standards, and meets security requirements, ultimately leading to higher-quality and more maintainable software.

SonarQube allows you to customize the standards and requirements for any project in two ways:

Quality Profiles. With this capability, you can specify best practices and standards for every programming language, allowing customization of the rules applied during code analysis. This customization ensures the rules align with the coding guidelines and project quality expectations. Quality Profiles can be specific to different programming languages, meaning different rules can be applied depending on the language used.

Quality Gates in SonarQube are conditions that determine whether a project passes or fails the quality check at the end of an analysis. They act as checkpoints, ensuring code meets predefined quality standards before progressing through the development pipeline. The conditions within a Quality Gate typically include thresholds for various metrics such as bugs, vulnerabilities, code smells, test coverage, and duplications. A Quality Gate might, for instance, require that all newly written code have at least 60% test coverage or that no security flaws be found.

Using SonarQube locally


To use SonarQube locally, we first need to install it from a ZIP file or a Docker image.

When the installation is done, we need to run `StartSonar.bat`, in administration mode, and the SonarQube application will run at <http://localhost:9000>.


Now, we can create a project and create a unique token:

1 of 2


Create a local project

 Local projects won't benefit from the features associated with DevOps Platforms integration unless you configure them in the project settings.


Project display name *

 
Up to 255 characters. Some scanners might override the value you provide.

Project key *

 
The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

The name of your project's default branch [Learn More](#) 

Before analyzing a project, we need to install sonar scanner tools globally or run it as part of your DevOps CI pipeline, such as:

```
dotnet tool install --global dotnet-sonarscanner
```

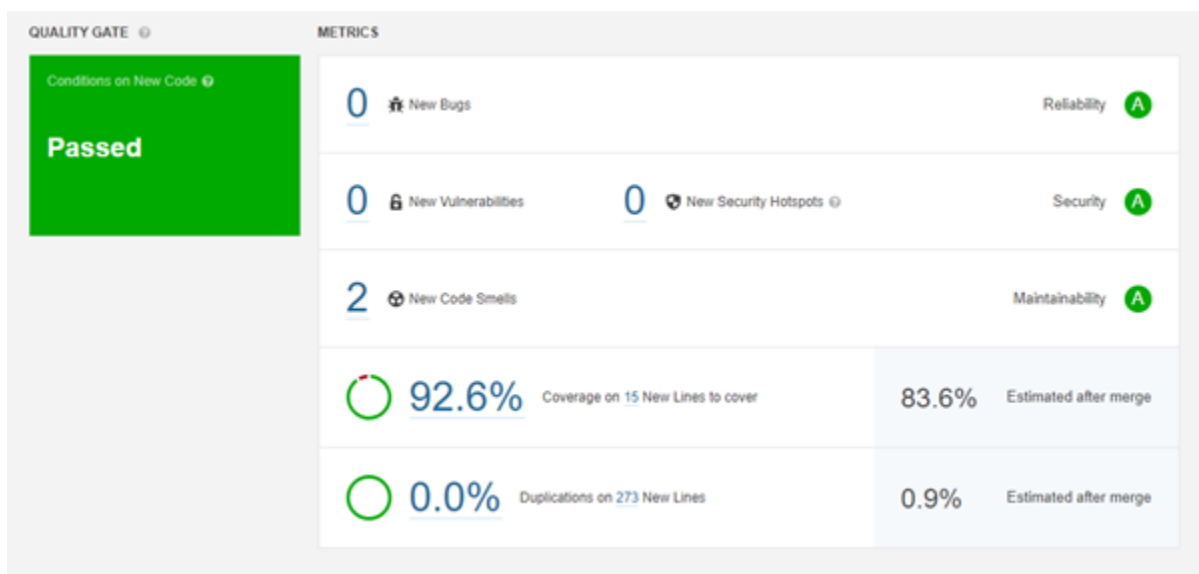
And then, we can execute the scanner:

```
dotnet sonarscanner begin /k:"demo"  
/d:sonar.host.url="http://localhost:9000" /d:sonar.login="token"
```

```
dotnet build
```

```
dotnet sonarscanner end /d:sonar.login="token"
```

The analysis's results can also be seen. This dashboard shows bugs, vulnerabilities, code smells, code coverage, security issues, and more.



Using SonarAnalyzer

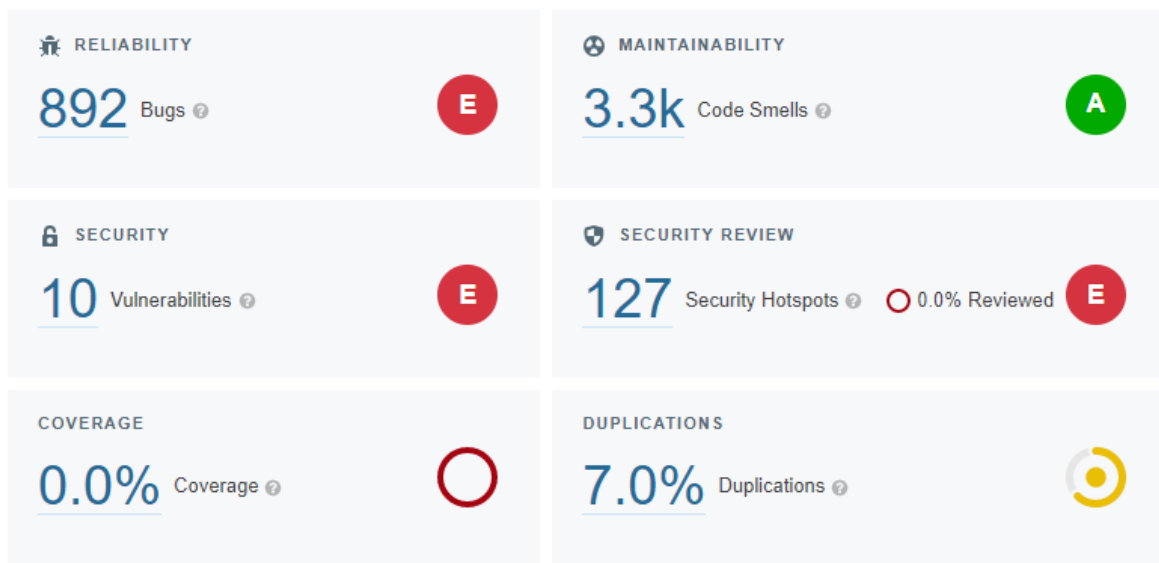
One additional tool that we can use is **SonarAnalyzer.CSharp** is the analyzer's package. This package enables us to integrate IDE rules so that they are applied when building applications locally, just as they were built in the CI/CD process with SonarQube rules applied.

All the rules have documentation that clearly explains the problem, and code samples show good and bad code ([see example](#)).

Using SonarCloud in the cloud

Another possibility is to use **SonarCloud** integration with your repository from GitHub, Bitbucket, Azure DevOps, or GitLab. Here, we can create a straightforward configuration to make a SonarCloud account/project and connect it to our online repository, which will be analyzed. A free version of SonarCloud offers unlimited analyzers and lines of code for all languages, but results are open to anyone. Analysis for private projects is paid from 11e per month.

In the example here, a popular [NopCommerce open-source project](#) is analyzed with a connection to SonarCloud as an open project. In the image below, we can see the result of the SonarQube analysis on the dashboard.



From the results of scanning 337k of code, we have **892 bugs**, **3.3k code smells**, **10 security vulnerabilities**,