# SonarQube use report

## Introduction

This document describes the research and implementation details for utilizing SonarQube for Oulu University’s Advanced Software Quality and Security Course for Group X.

## Background

This section details information about the SonarQube itself.

### Sonarqube in general

SonarQube is a software solution for static code analysis. It provides issue detection by analyzing the code base, resulting into identification of code smells, low code coverage and security issues.

SonarQube is developed by Sonar (SonarSource, n.d.-a).

### SonarQube licensing

SonarQube offers multiple licensing plans:

|  |  |
| --- | --- |
| **Edition** | **Key Features (relevant to licensing/integration)** |
| **Community** | Static code analysis, basic reporting, integration with CI/CD pipelines, support for major languages, open-source usage. Limited security vulnerability detection. |
| **Developer** | Adds integration with IDEs, additional languages, pull request decoration, branch analysis, improved security analysis. |
| **Enterprise** | Advanced reporting, governance, security audits, project portfolios, role-based permissions, SAML, high availability clustering. |
| **Data Center** | Designed for very large-scale deployments, distributed architecture, load balancing, horizontal scaling. |

### SonarQube in use

SonarQube can be used in the CI/CD pipeline to detect issues with builds. These analyses can be used as supplementary material in pull requests and in other processes.

## Methods

### Requirements & SonarQube plan selection

For our project, our requirements towards SonarQube are the following:

* Produce SonarQube analysis (Assignment task 2, Case 1)
  + No additional requirements in terms of comprehensiveness
* No cost, or very limited cost

Based on these requirements, we will select Community edition, since it should allow us to produce a basic analysis of the code at no cost.

### Containerization & Use

In line with our project’s containerization direction, we will utilize docker image provided by Sonar to run the service (SonarSource, n.d.-b). To this end, we incorporate SonarQube as a service to our main docker compose file.

Regarding the credentials for the admin access: we will merely hardcode them into the docker compose file to facilitate ease of setup. It goes without saying, that these should be stored elsewhere in a production system. Funnily, we might get code analysis report of this specific issue, but that remains to be seen.

Considering the limited scope of the project, the setup of the project is manually handled into SonarQube, as well as running of the code analysis. Due to persisting of the SonarQube data, the project needs to be initialized only once per machine.

Normally we would consider integrating the running of the analysis into the continuous integration pipeline, but here we determined it to be not relevant.

### Project setup

The project setup needs to be in SonarQube. Github integration might be a good idea otherwise, except for the requirement of setting up the repository as a GitHub app (GitHub, n.d.), which we thought to not be warranted. Thus, we selected to configure the project as a “local project”. See Figure 1.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1. Local project setup selection in SonarQube

### Running an analysis

First step in running an analysis is in provisioning a token. For our static analysis, it was the following:

**sqp\_8c2871782652c677dbbe25f940382ad215798ffb**

The scanner for running the analysis is not provided by the SonarQube container, so it needs to be acquired for the analysis. The following command is used:

docker run \

--rm \

--network llm-message-dispatch-tool\_sonar-net \

-e SONAR\_HOST\_URL="http://sonarqube:9000" \

-e SONAR\_TOKEN="sqp\_8c2871782652c677dbbe25f940382ad215798ffb" \

-v "$(pwd):/usr/src" \

sonarsource/sonar-scanner-cli

This command is run at the base of the repository. It uses a downloaded sonar-scanner-cli image with docker. The container is automatically removed once it exits, and the results are stored within the SonarQube service.

### Exporting textual report

The community edition does not include the possibility for exporting textual report of the issues. However, with SonarQube, a Rest API is provided which can be used to this end. We used the following command to get issues, as instructed by assignment instructions:

curl -u <token\_generated\_via\_sonarqube\_ui>: <https://localhost:9000/api/issues/search?componentKeys=llm-message-dispatch>

This command is executed via bash script available in GitHub repository and path: Scripts/export\_sonarqube\_issues.sh. It outputs the report automatically to path: Deliverables/code\_analysis\_reports/.

## References

​SonarSource. (n.d.-a). *About us*. Sonar. Retrieved March 15, 2025, from <https://www.sonarsource.com/company/about/>​

SonarSource. (n.d.-b). *SonarQube Community Edition Docker Hub.* Retrieved March 15, 2025, from <https://hub.docker.com/_/sonarqube>

GitHub. (n.d.). *About creating GitHub Apps*. Retrieved March 15, 2025, from <https://docs.github.com/en/apps/creating-github-apps/about-creating-github-apps/about-creating-github-apps>