**Choosed SAST Tool**

**Why SonarQube is the Best SAST Choice:**

After evaluating project’s specific needs and constraints, I will provide a detailed analysis on why SonarQube is the best SAST tool for our Java Maven Spring Boot project, which will run in a Docker container on an Ubuntu VM and within a GitLab CI/CD pipeline.

**Criteria and Key Considerations**

1. **Project Setup**:
   * **Java Maven Spring Boot Project**
   * **Docker Container**: the project runs as a docker image in the docker container.
   * **Ubuntu VM**: The CI/CD pipeline will run in an **Ubuntu Virtual Machine** (VM), which can run both the code analysis tool and Docker containers.
   * **GitLab CI/CD**: The project is hosted on **GitLab**, and the pipeline is used for building, testing, and deploying the application.

**Why SonarQube is most suitable:**

**1. SAST for Java/Maven Projects:**

* **SonarQube** is a well-established and widely-used tool for **Static Application Security Testing (SAST)** that focuses on **source code security**. It is an ideal tool for projects like **Java** and **Spring Boot**, which are known to have security issues rooted in the application’s **codebase**.
* **Key Features for Java/Maven Spring Boot Projects**:
  + SonarQube offers specific rules and plugins for **Java** code, making it **highly relevant** for analyzing **Spring Boot** applications.
  + **Spring Boot** applications often have complex configurations and interactions with various dependencies, such as **Spring Security** or **Hibernate**, which SonarQube can help secure by detecting security vulnerabilities like **SQL injection**, **XSS**, **insecure data handling**, and **hardcoded credentials**.
  + **Maven Integration**: Since we are using **Maven** to manage our dependencies, SonarQube integrates **seamlessly with Maven**. The SonarQube Maven plugin can be used to analyze the code with minimal configuration, so after building the project using Maven, you can automatically trigger the code analysis.
* **Key SAST Focus Areas for SonarQube**:
  + **Security Vulnerabilities**: SonarQube detects security flaws in Java code such as **SQL injection**, **XSS**, **broken authentication**, and more.
  + **Code Quality**: SonarQube goes beyond security to also provide insights into **code smells**, **bugs**, and **complexity**. It helps improve code maintainability and readability, which is important for long-term project health.
  + **Code Duplication & Maintainability**: SonarQube will help you spot code duplication, which is crucial in Java-based systems like **Spring Boot**, where reusing configurations is common.

**2. Optimized for Continuous Integration and GitLab:**

* **GitLab Integration**: SonarQube integrates **seamlessly** into **GitLab CI/CD pipelines**. You can trigger the SonarQube analysis directly within your **GitLab pipeline**, making it an excellent choice for continuous code analysis during development and deployment stages.
* **How SonarQube Integrates**:
  + define a **SonarQube job** in your .gitlab-ci.yml file to trigger code analysis automatically as part of pipeline.
  + You can use SonarQube’s **scanner for Maven** or **SonarQube scanner CLI** for integration with the build process to analyze the **source code** once your application is built by Maven.
  + After each code push or merge, SonarQube will run in the CI/CD pipeline and provide reports that are accessible directly in the **SonarQube dashboard**.

**3. Compatibility with Docker and Ubuntu VM:**

* **Docker Compatibility**: Although SonarQube does not perform direct **Docker image security scanning**, it can run within a **Docker container** itself. This means that you can deploy SonarQube in a **Docker container** on your **Ubuntu VM** and easily integrate it into your CI/CD pipeline.
  + **SonarQube Docker Container**: we can run SonarQube in a **Docker container** on the Ubuntu VM to provide centralized code analysis and monitoring. This setup is quite common in CI/CD pipelines, and it works well with both **Maven-based Java projects** and **containerized environments**.
  + **Dockerized Pipeline**: Since **Docker** is used for our Spring Boot application, SonarQube can run its own Docker container, allowing us to keep your analysis environment separate from your application environment. This also means you can scale or replicate the analysis environment easily.

**4. Actionable Reports and Dashboards:**

* **SonarQube Dashboards**: The **SonarQube dashboard** provides **detailed reports** that are very helpful for both developers and project managers. After each analysis, SonarQube provides a comprehensive overview of:
  + **Security vulnerabilities**
  + **Code quality issues**
  + **Test coverage**
  + **Duplication and complexity**

This will give the necessary visibility into the health of the project, whether it’s about **code security** or **overall code quality**.

**5. Community and Documentation:**

* **Wide Adoption and Documentation**: SonarQube has been widely adopted across many organizations for **SAST** due to its reliability and robustness. It has an extensive documentation base, active community, and a lot of **tutorials and best practices** available, which will help you during integration and troubleshooting.

**SonarQube** is the **best choice** for **SAST** because:

1. It integrates smoothly with **Maven-based Spring Boot project**.
2. It provides **detailed static analysis** of **Java code**.
3. It is well-supported for **GitLab CI/CD pipelines**.
4. It integrates well in a **Dockerized environment** (both for containerizing SonarQube and running it alongside your project).

**Tldr;**

Given project’s **tech stack** (Java, Spring Boot, Maven), **Docker**-based setup, and use of **GitLab CI/CD**, **SonarQube** is the **best fit for SAST**. It provides comprehensive analysis for **Java code** and integrates well into **Ubuntu VM** and **GitLab pipeline**. It will allow us to focus on **code security** and **code quality**, which is exactly what you need for static application security testing in your project. As well as a vulnerability report from their website.