**1. Deploy SonarQube Server**

* Run **SonarQube in a Docker container** on the **same Ubuntu VM** (or a different host if you prefer).
* Ensure the SonarQube web UI is accessible (e.g., on port 9000).

**2. Create a Project in SonarQube**

* Access the SonarQube UI (http://<your-vm-ip>:9000).
* Create a **new project** for your app.
* Generate a **project token** (you’ll use this token in your pipeline for authentication).

**3. Add the Token to GitLab**

* In GitLab, go to your **project settings → CI/CD → Variables**.
* Add a variable like:
  + SONAR\_TOKEN: your SonarQube token

**4. Update Your Maven Project**

* Make sure your project has the **SonarQube Maven plugin** configured.
* This enables static code analysis when the Maven build runs.

**5. Add a SonarQube Scan Job to Your .gitlab-ci.yml**

* In your existing pipeline, insert a **new job** (in the right stage, like test or analyze) that:
  + Runs your Maven build.
  + Includes the sonar:sonar command.
  + Uses the SONAR\_TOKEN and the SonarQube server URL.

**6. Run the Pipeline**

* Commit and push your code.
* GitLab will trigger the pipeline:
  + Your Docker-based runner will execute the SonarQube scan via Maven.
  + The results will be uploaded to the SonarQube server.

**7. View and Share Results**

* Open the SonarQube dashboard.
* See detailed **vulnerability reports**, **code smells**, and **bugs** for your Java code.
* Use these reports to build your own dashboard or export data.

squ\_4853d36dd8d2fe229ba46e6fb9de41f6cc463495

[todolist - Issues - SonarQube Community Build](http://10.97.7.109:9000/project/issues?id=th.ab.demo%3Atodolist&issueStatuses=OPEN%2CCONFIRMED)