Adv.DevOps Exp 08

Name- Ishika Devare RollNo- 14 Batch- A

Adv. DevOps

Experiment No. 8

Aim - create a Jenkins CICD pipeline with sonarQube / GitLab integration to perform a static analysis of the code to detect bugs, code smells and security vulnerabilities on a sample Web / Java / Python application.

Theory -

What is SAST.?

Static Application security Testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to next phase of software development lifecycle (SDLC).

What is CI/CD pipeline ?

CI/CD pipeline refers to continuous Integration/ continuous Development pipeline.

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases.

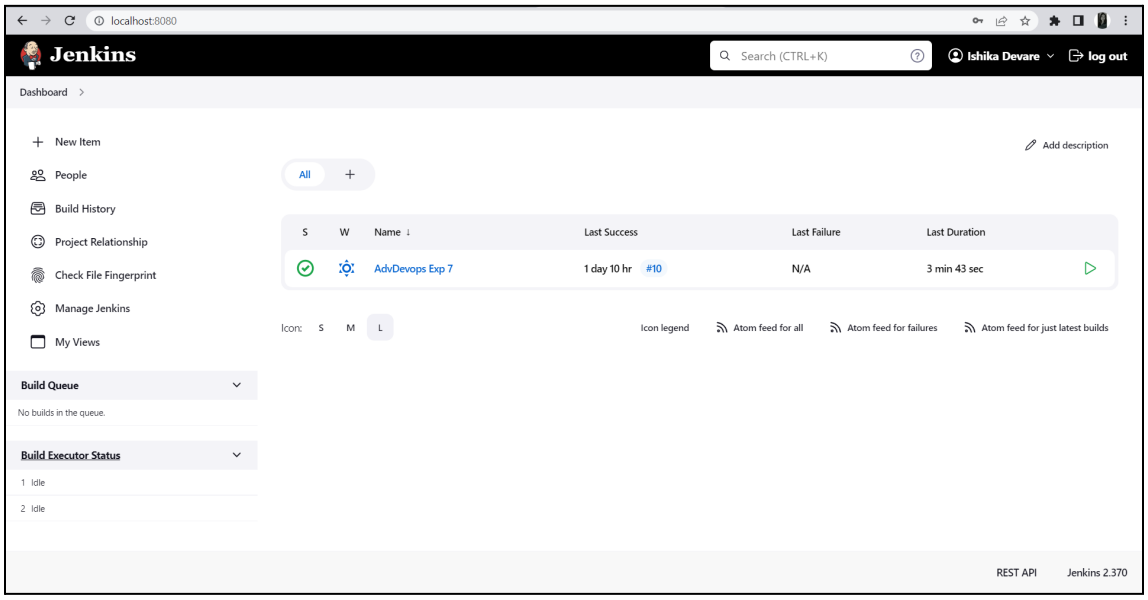
The pipeline is a backbone of DevOps approach. The pipeline is responsible for building codes, running tests and deploying new software versions.

What is sonarQube?
SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality.

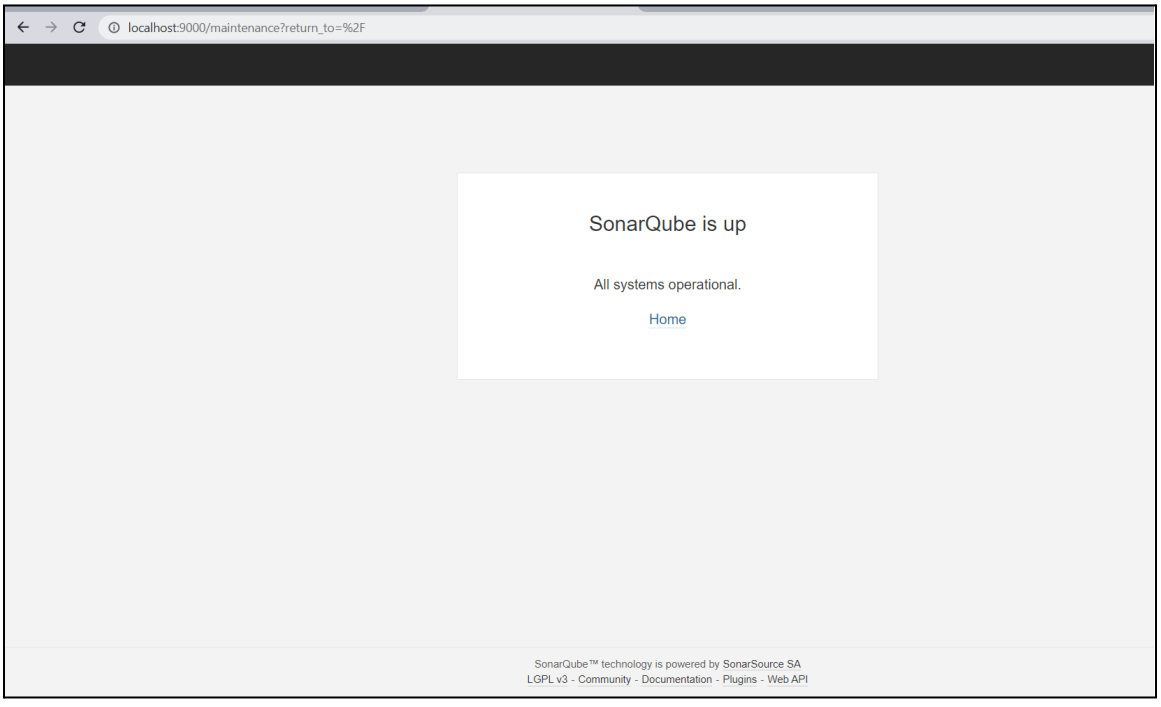
It supports 25+ major programming languages through built-in rulesets and also be extended with various plugins.

Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST:

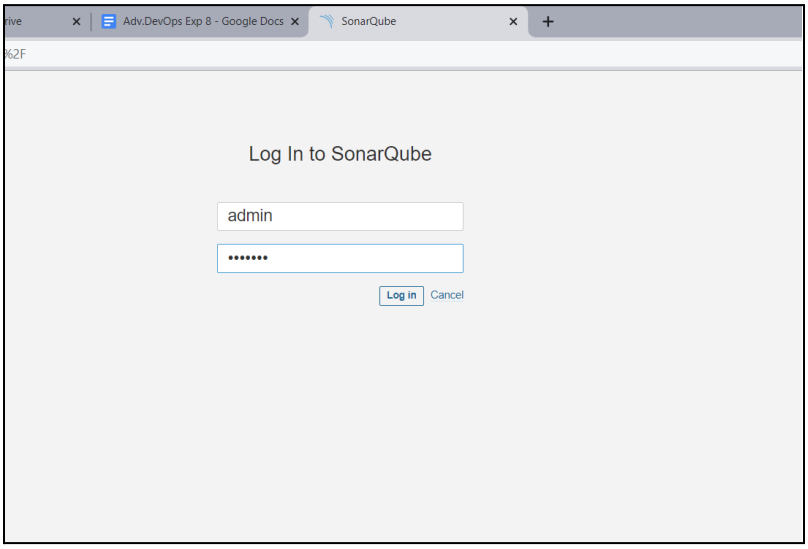
Step 1: Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



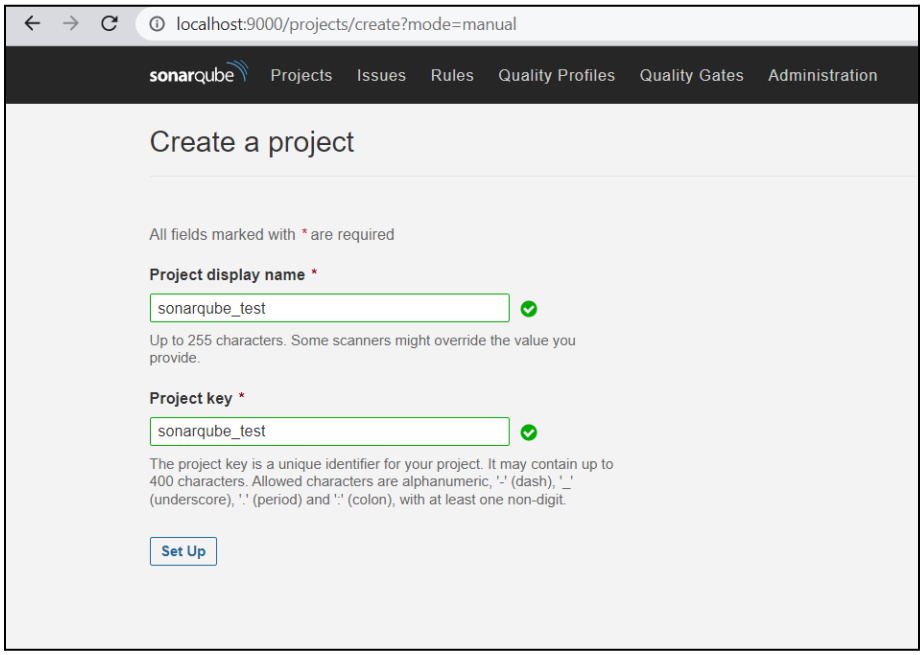
Step 2: Open SonarQube dashboard on localhost, port 9000 or whichever port it is at for you.



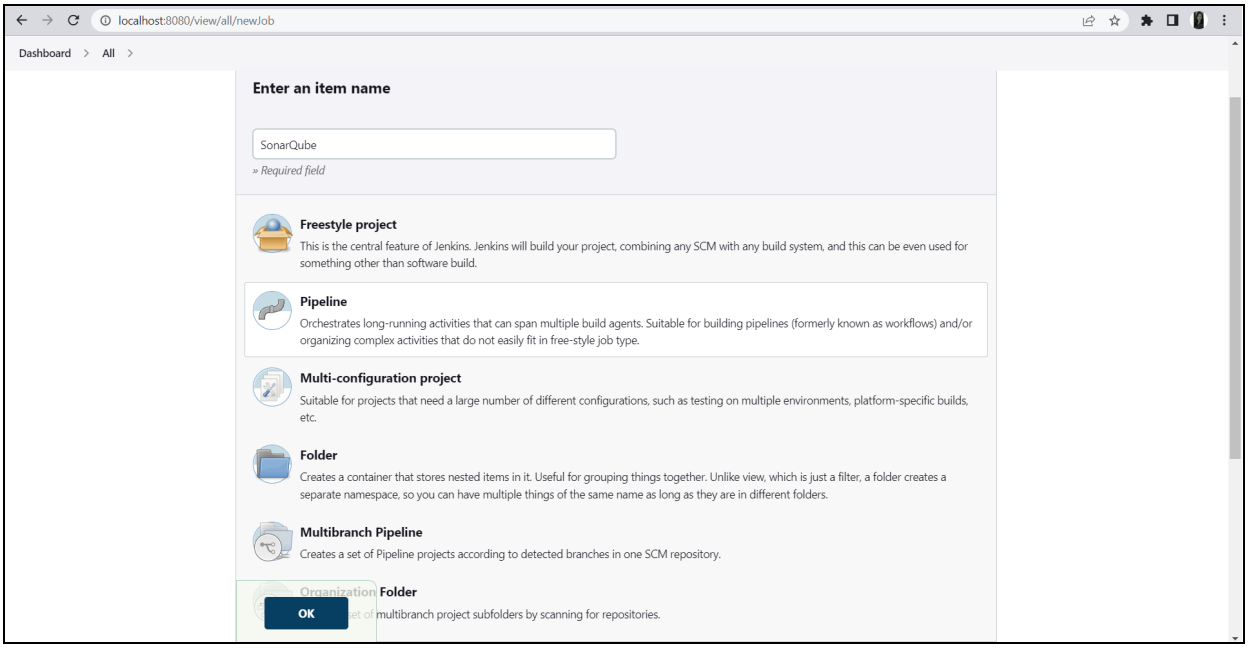
Step 3: Login to SonarQube using username *admin* and password *which you have set.*



Step 4: Create a manual project in SonarQube with the name sonarqube-test. Setup the project and come back to Jenkins Dashboard.

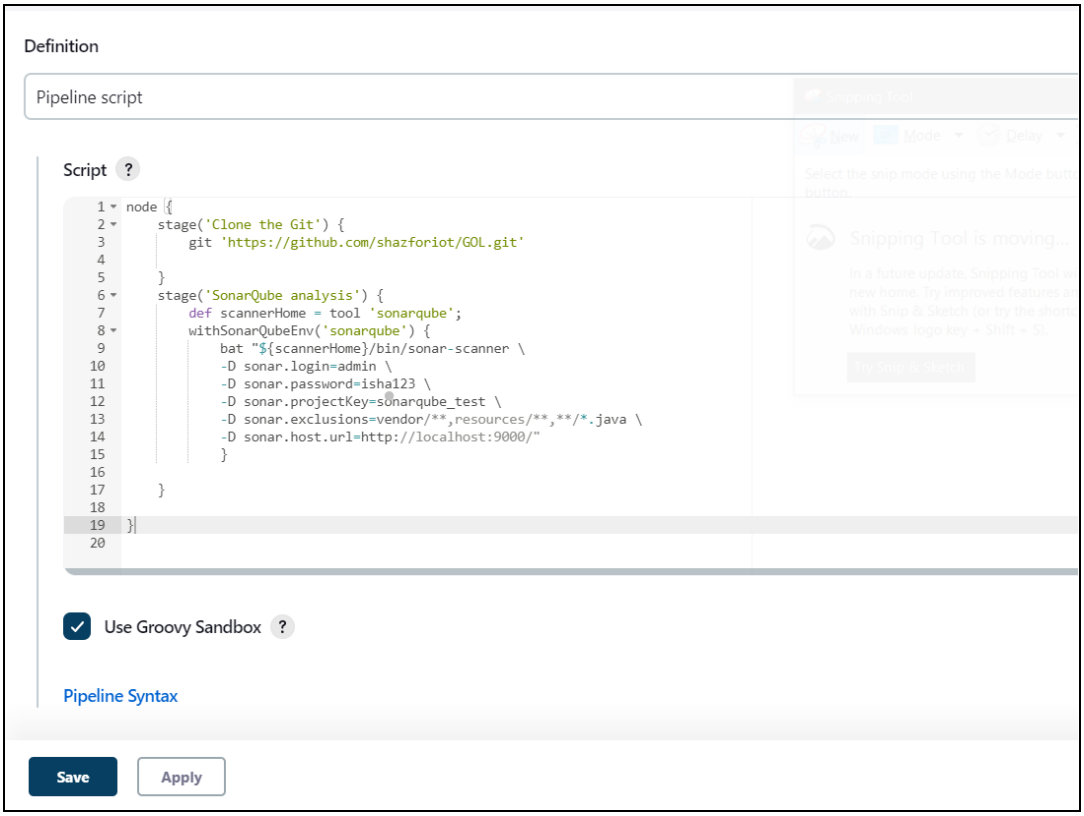


Step 5: Create a New Item in Jenkins, choose **Pipeline**



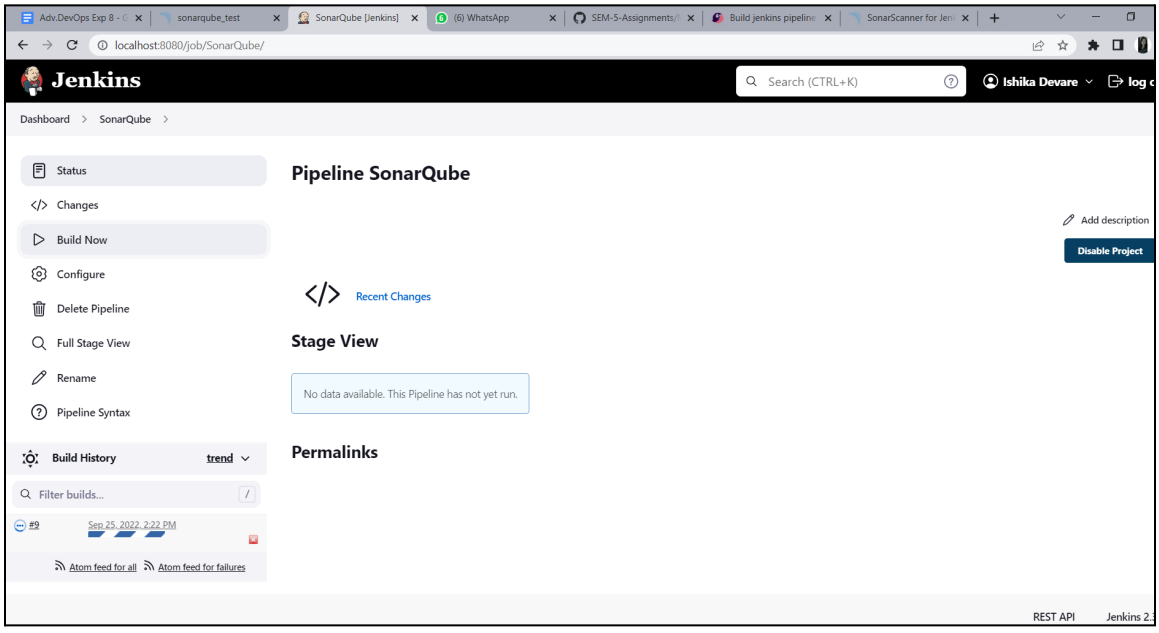
Step 6: Under Pipeline Script, enter the following -

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }
    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            sh "<PATH_TO_SONARQUBE_FOLDER//bin//sonar-scanner \
            -D sonar.login=<SonarQube_USERNAME> \
            -D sonar.password=<SonarQube_PASSWORD> \
            -D sonar.projectKey=<Project_KEY> \
            -D sonar.exclusions=vendor/**,resources/**,**/*.java \
            -D sonar.host.url=http://127.0.0.1:9000/"
        }
    }
}
```

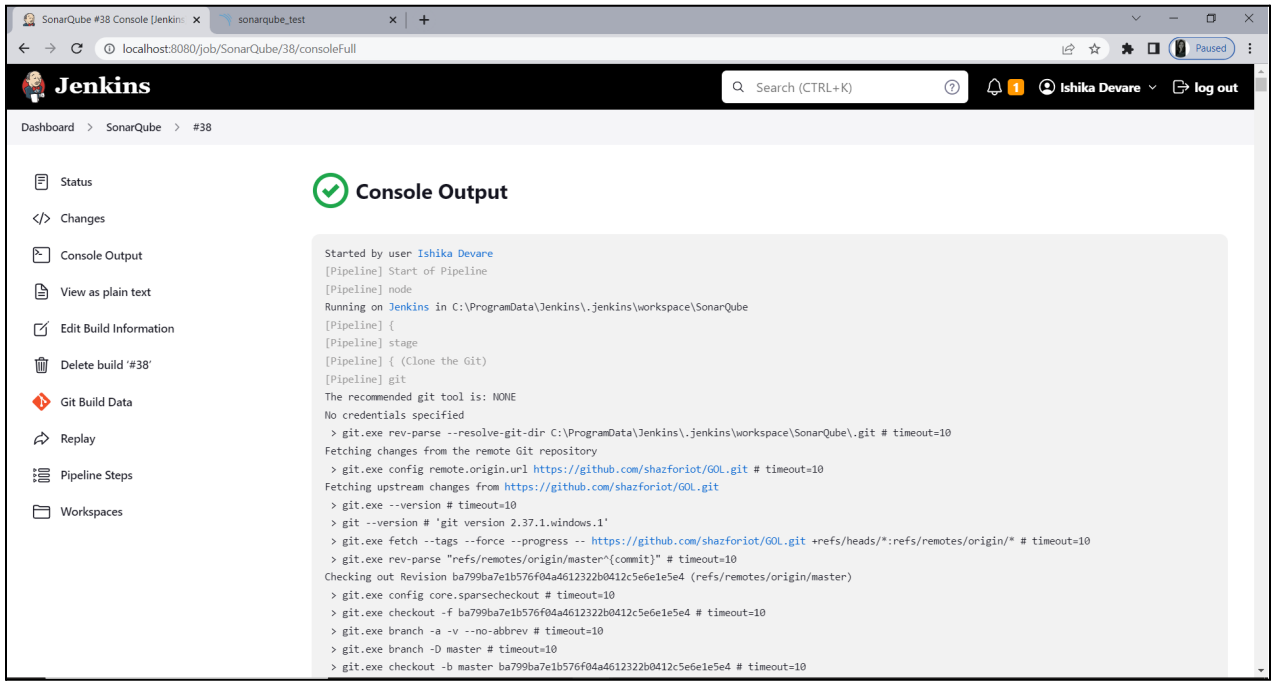


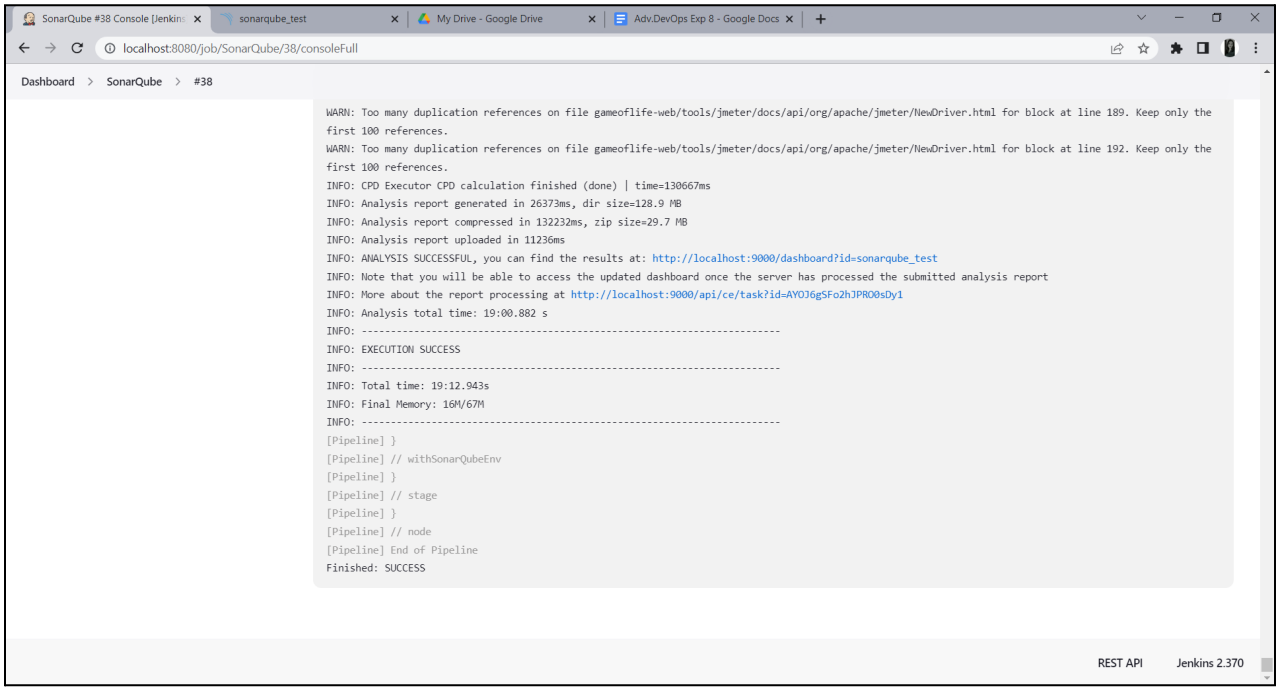
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

Step 7: Run the build

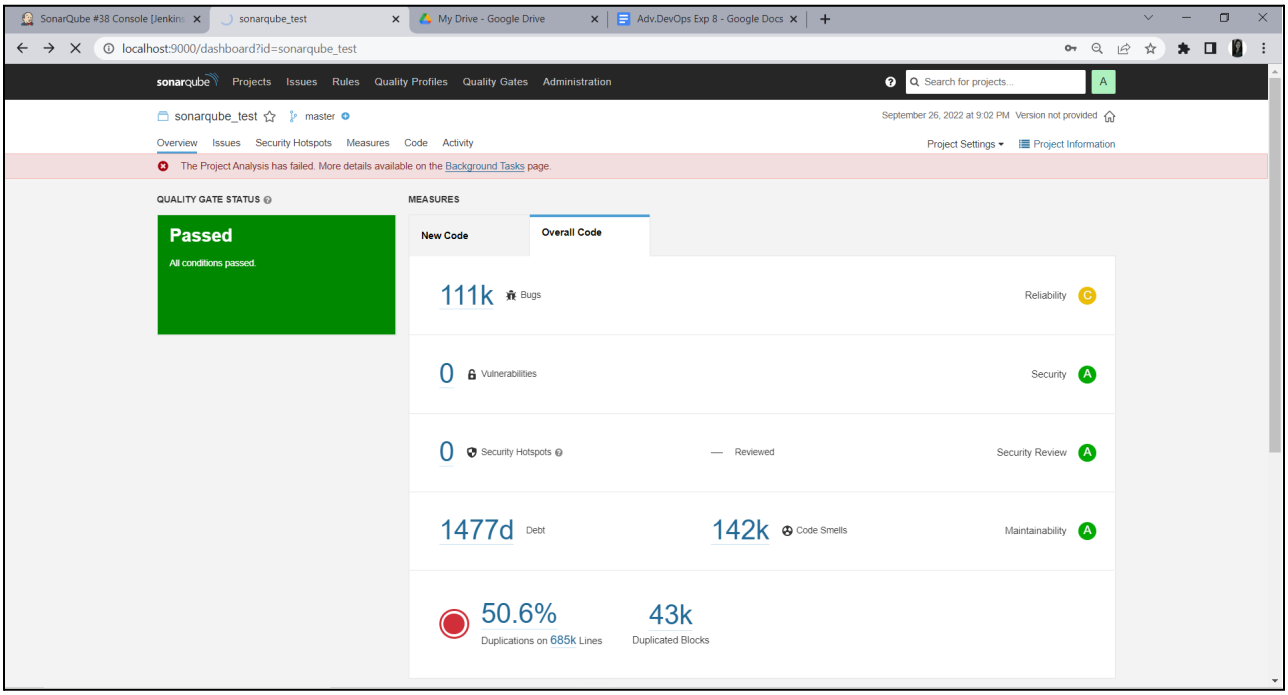


Step 8: Check the console output once the build is complete.





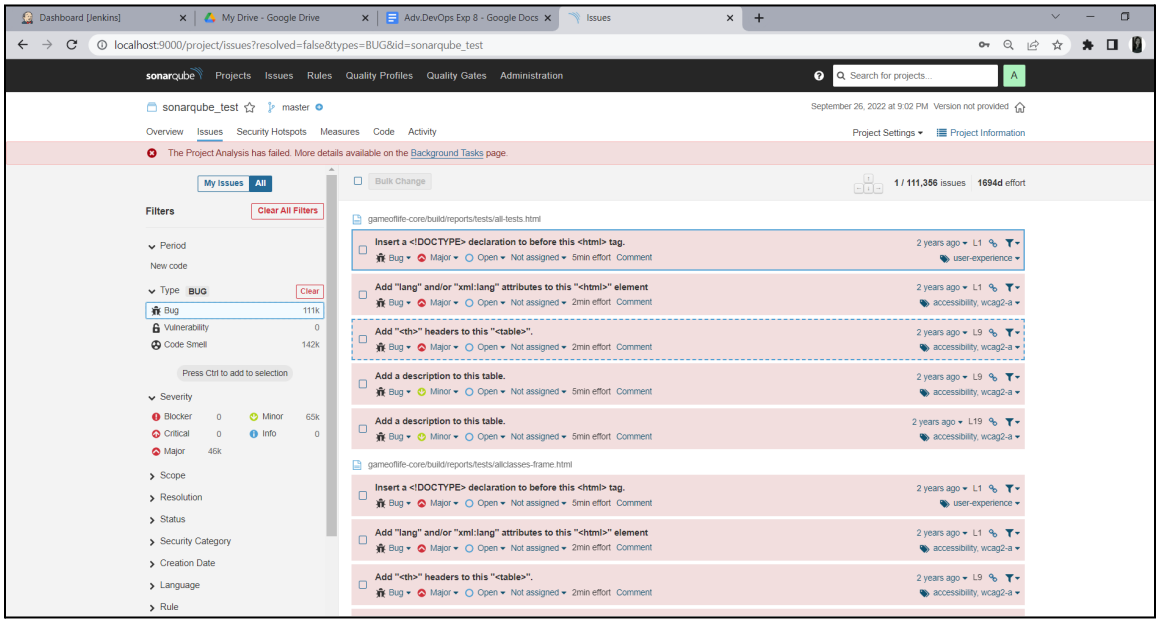
Step 9: After that, check the project in SonarQube.



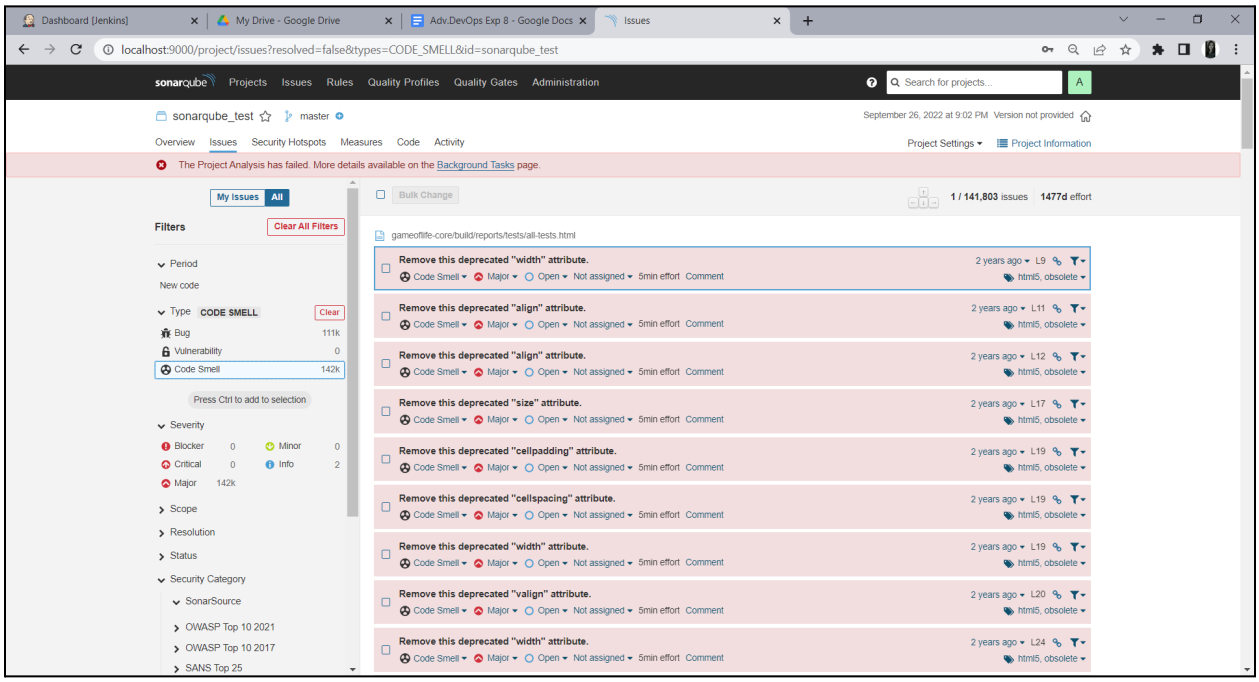
Under different tabs, check all different issues with the code.

Step 10: Code Problems

Bugs



Code smell



In this way, we have created a CI/CD Pipeline with Jenkins and integrated it with SonarQube to find issues in the code like bugs, code smells, duplicates, cyclomatic complexities, etc.

* Conclusion - We already integrated jenkins previously, with this experiment we will see how to build pipeline that will aid in SAST analysis. We discovered bugs, code smells in sample code after executing pipeline.