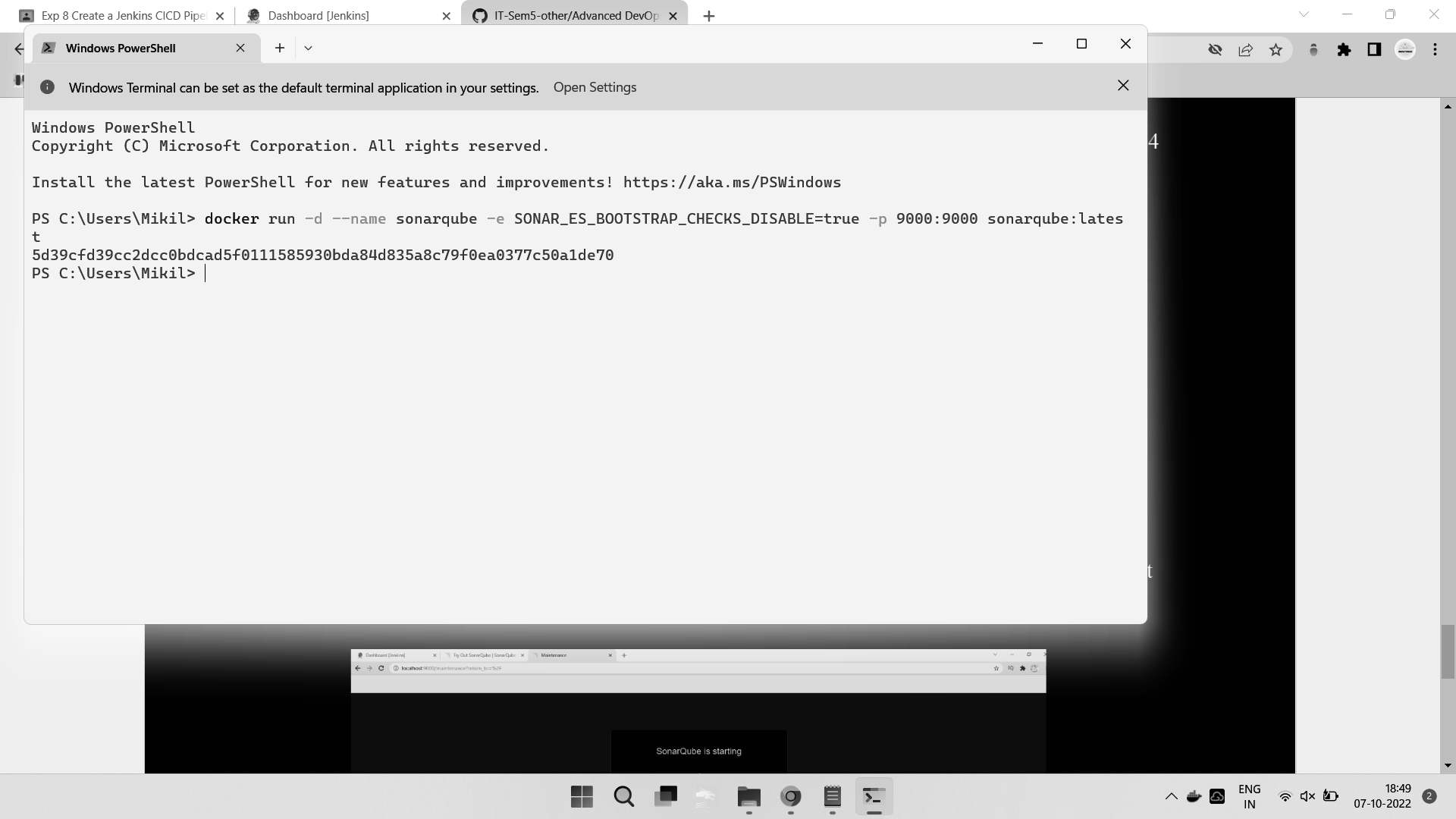
Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST:

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

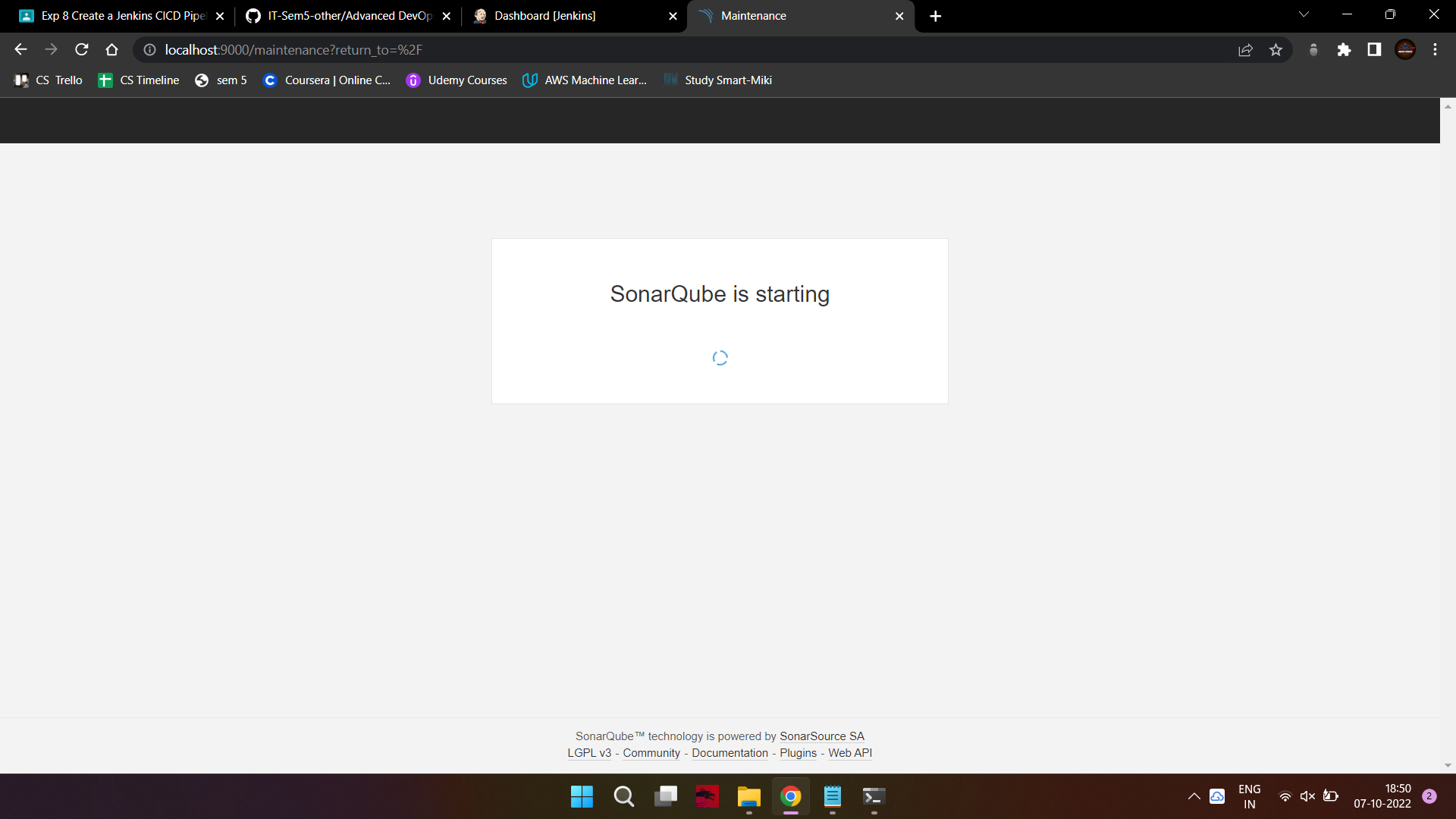
2. Run SonarQube in a Docker container using this command -

docker run -d --name sonarqube -e SONAR\_ES\_BOOTSTRAP\_CHECKS\_DISABLE=true -p 9000:9000 sonarqube:latest



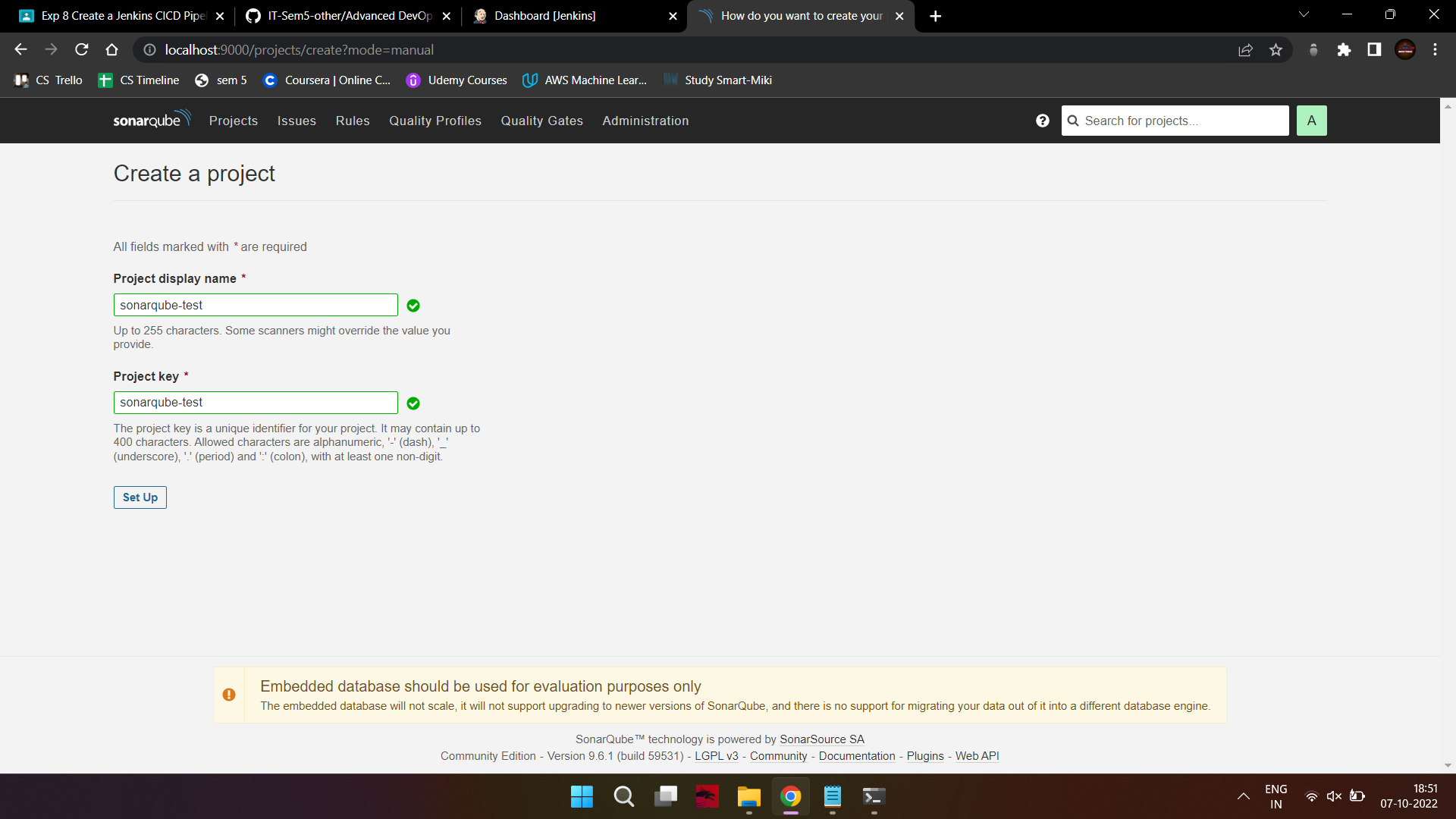
3. Once the container is up and running, you can check the status of SonarQube at localhost

port 9000.



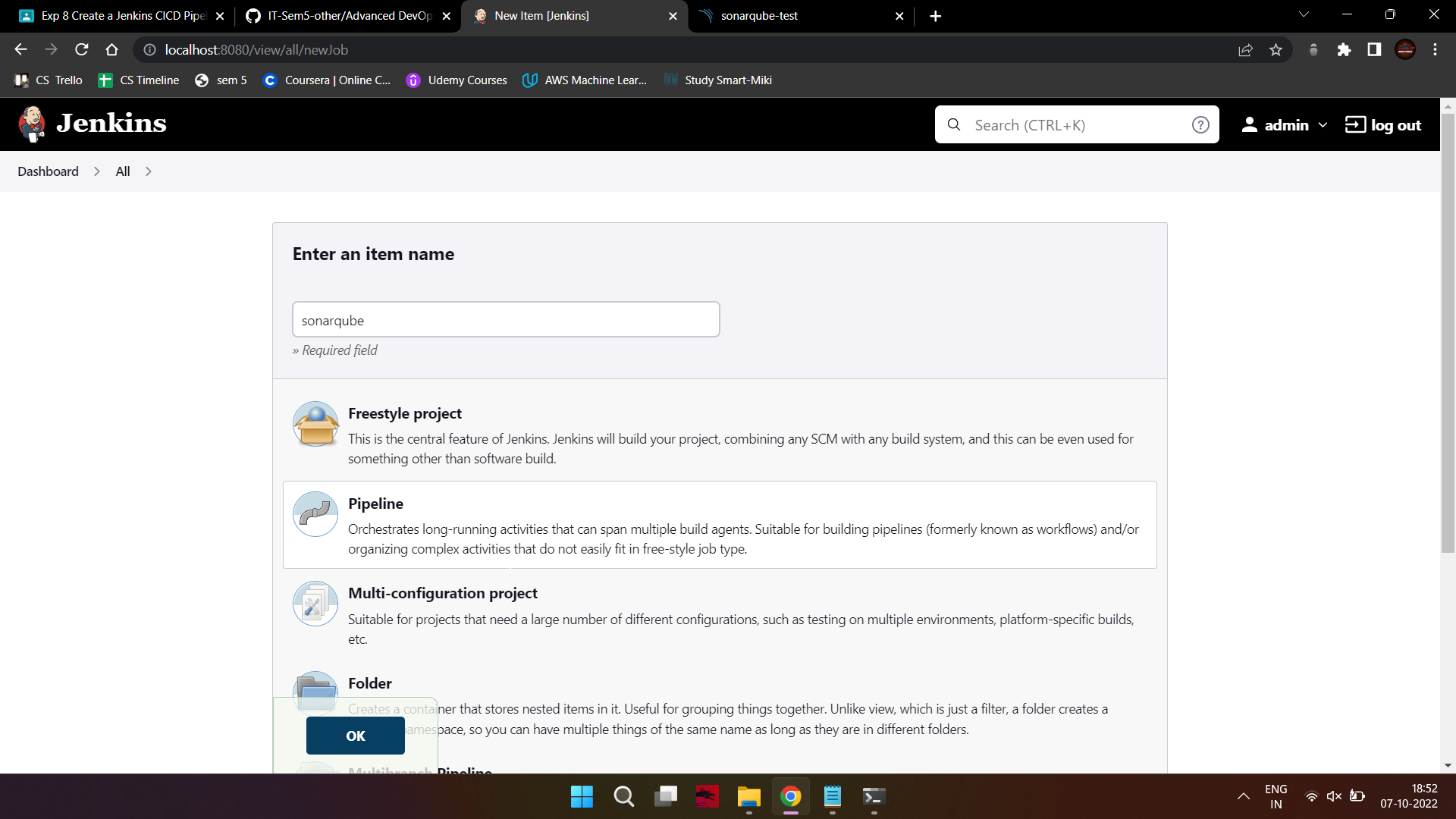
4. Login to SonarQube using username admin and password admin.

5. Create a manual project in SonarQube with the name sonarqube-test



Set up the project and come back to Jenkins Dashboard.

6. Create a New Item in Jenkins, choose Pipeline.



7. 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 "/c/ProgramData//Jenkins/.jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarqube/bin//sonar-scanner \

-D sonar.login=admin \

-D sonar.password=mikami \

-D sonar.projectKey=demoapp-project \

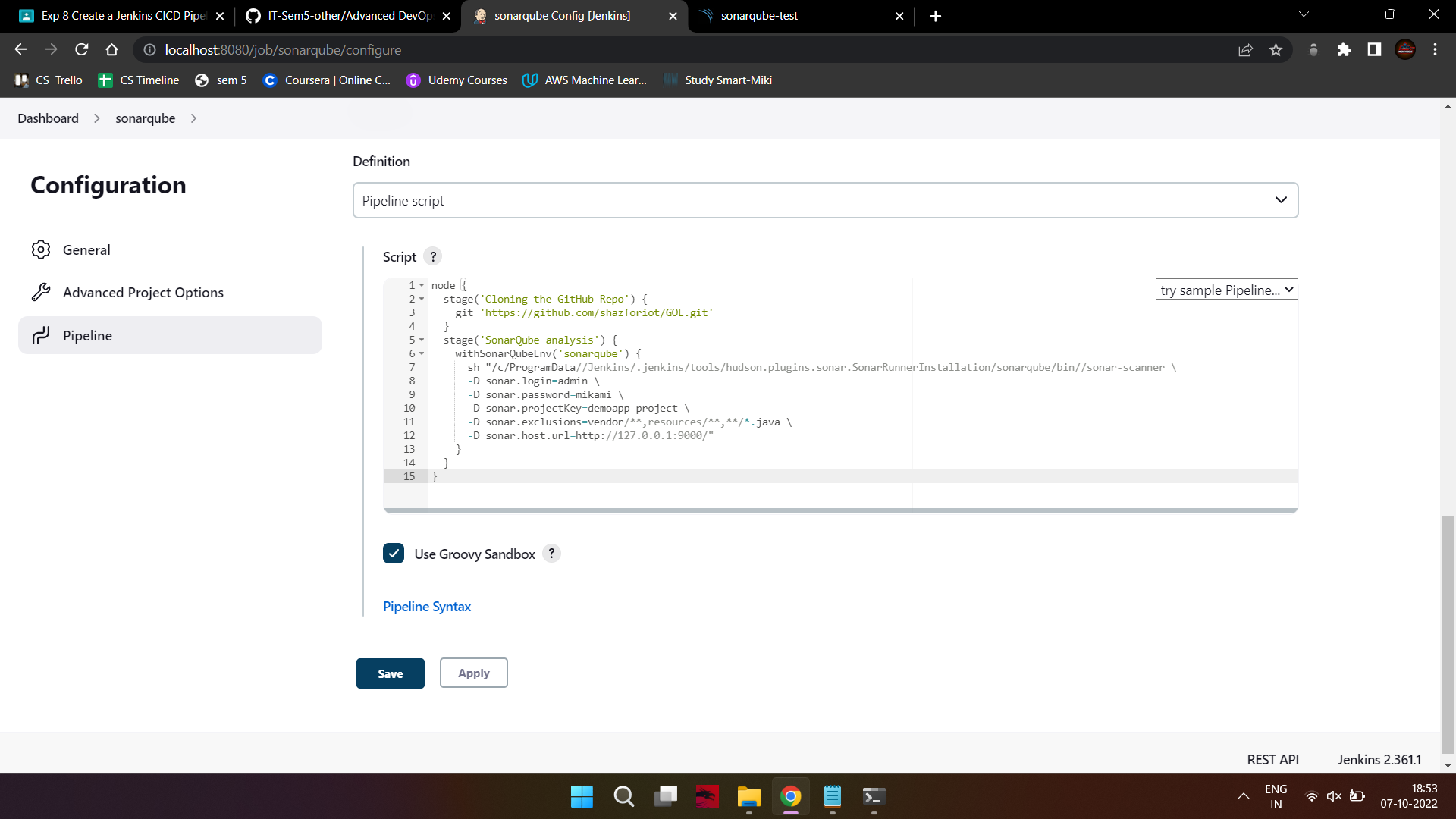
-D sonar.exclusions=vendor/\*\*,resources/\*\*,\*\*/\*.java \

-D sonar.host.url=http://127.0.0.1:9000/"

}

}

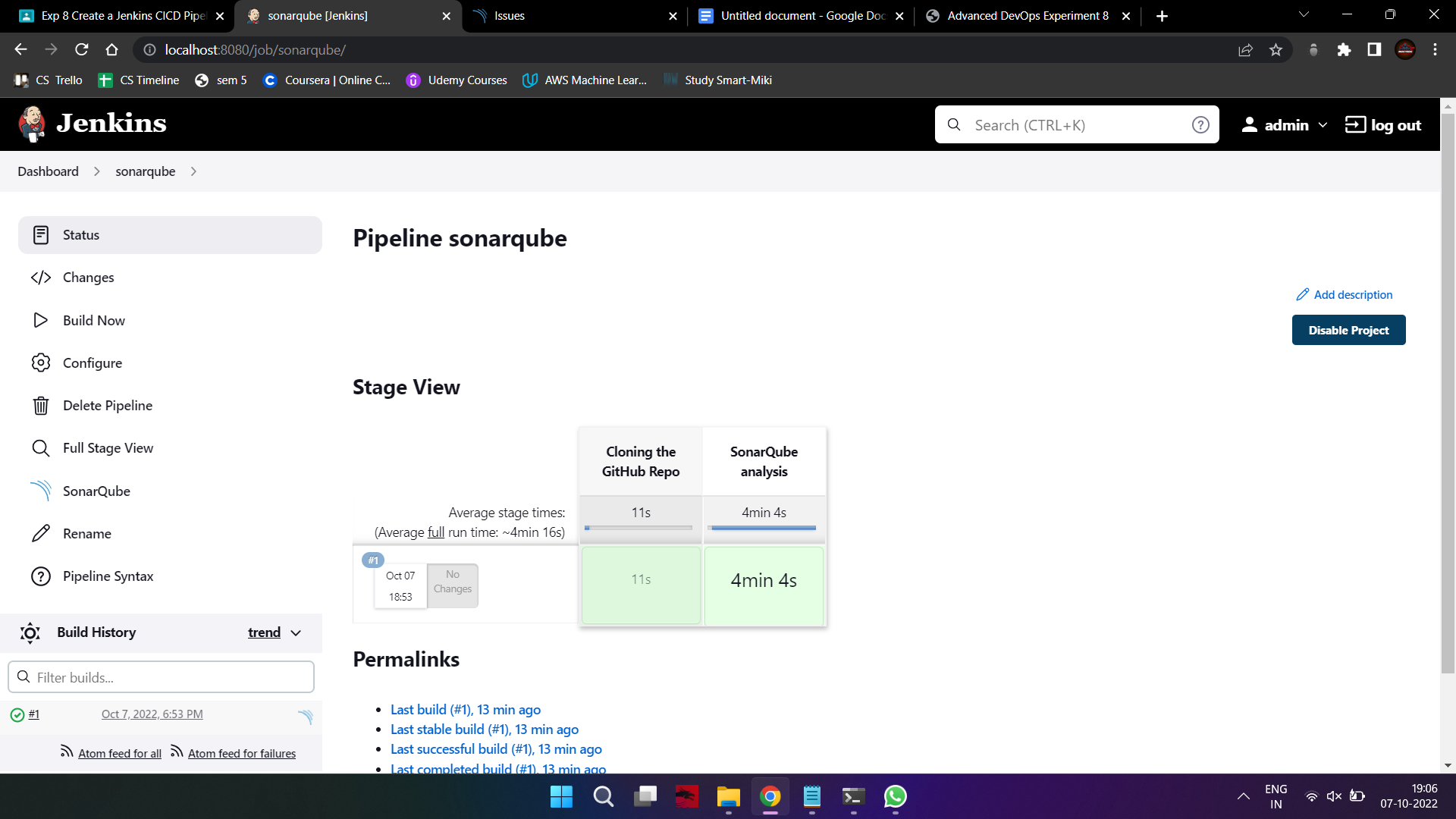
}



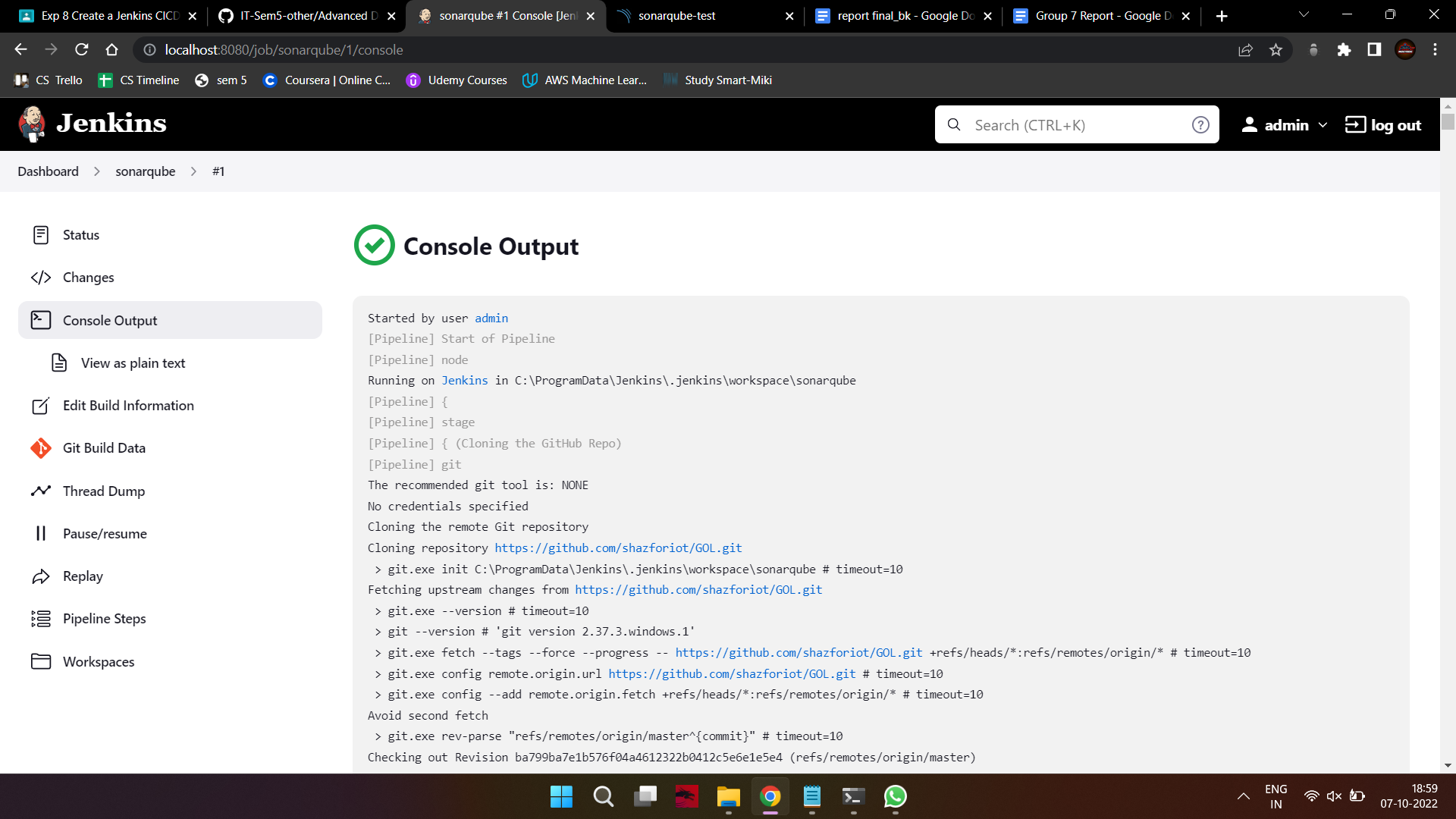
It is a java sample project which has a lot of repetitions and issues that will be detected by

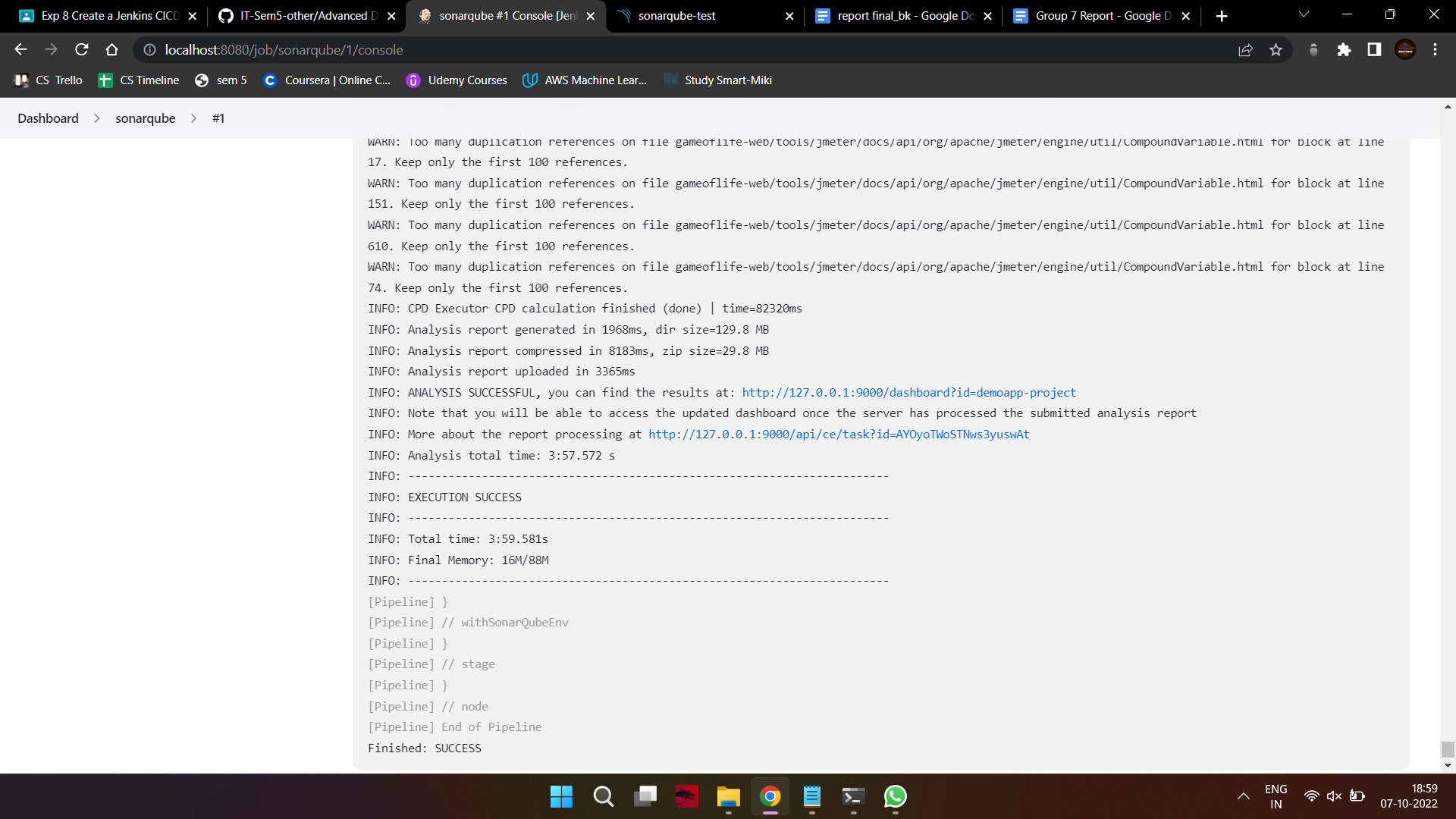
SonarQube.

8. Run The Build.



9. Check the console output once the build is complete.

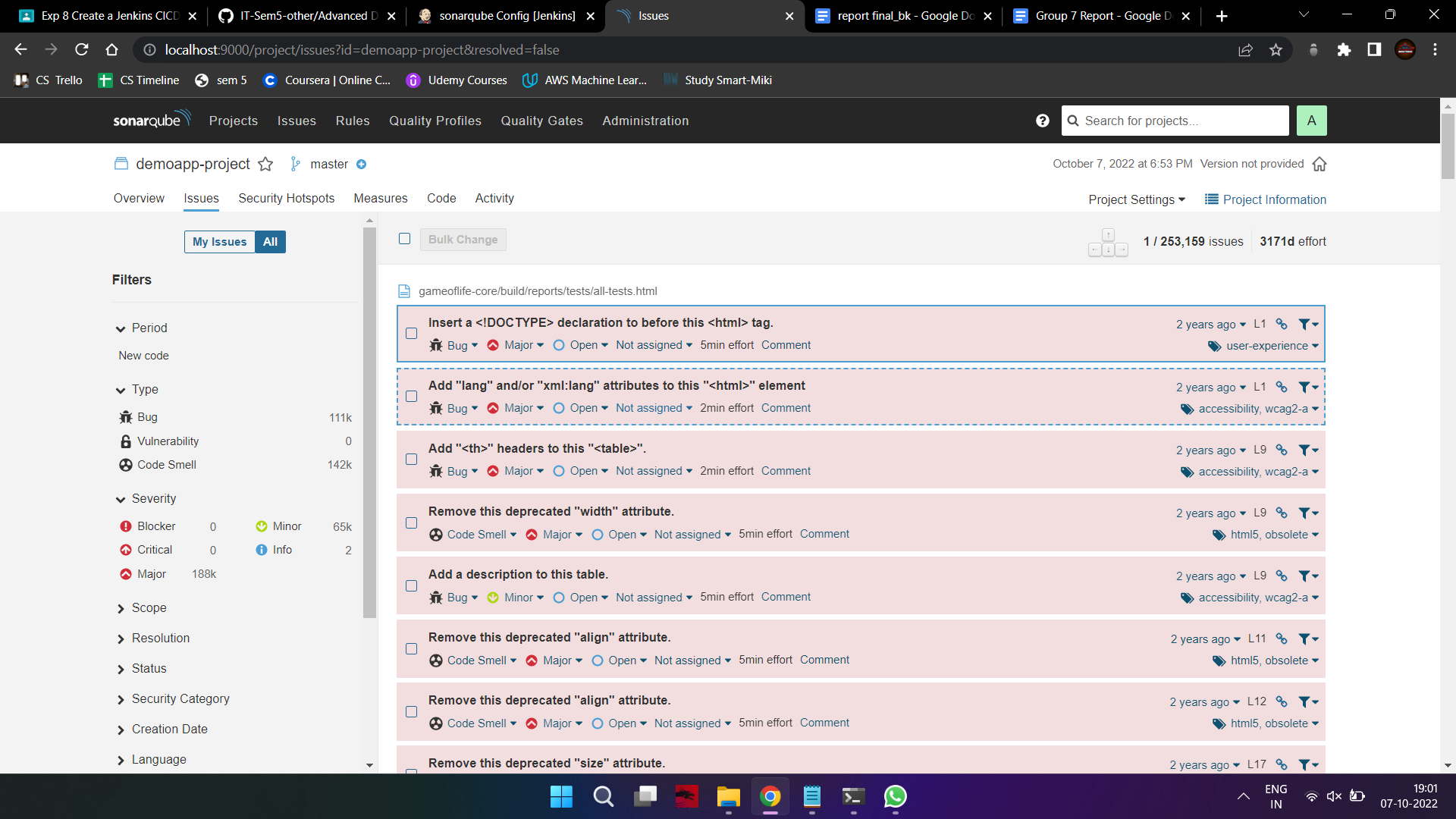


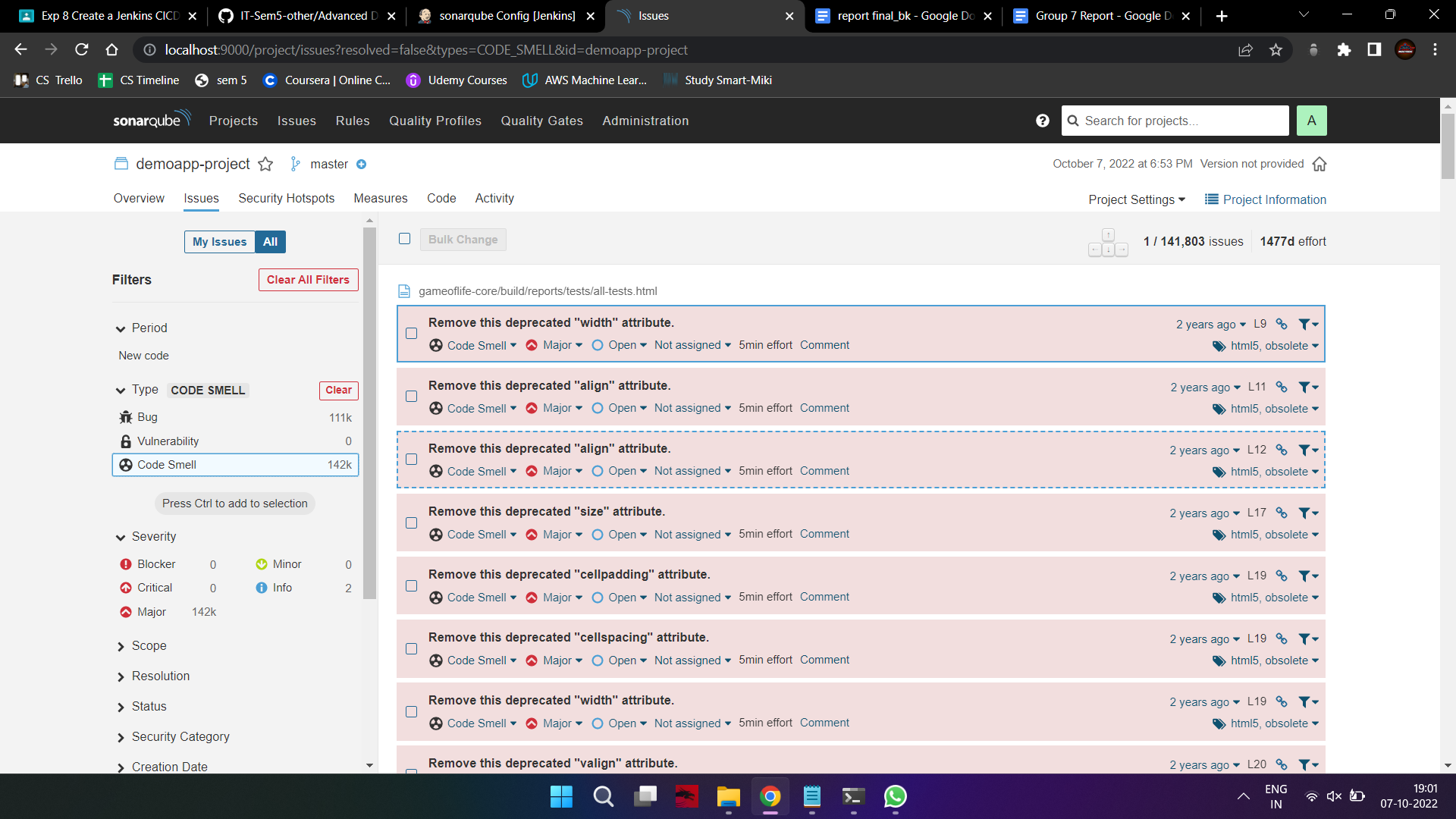


10. After that, check the project in SonarQube.

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

11. Code Problems -





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.