Name: Shubham Jha D15C Roll No. 19

08 Advanced DevOps Lab

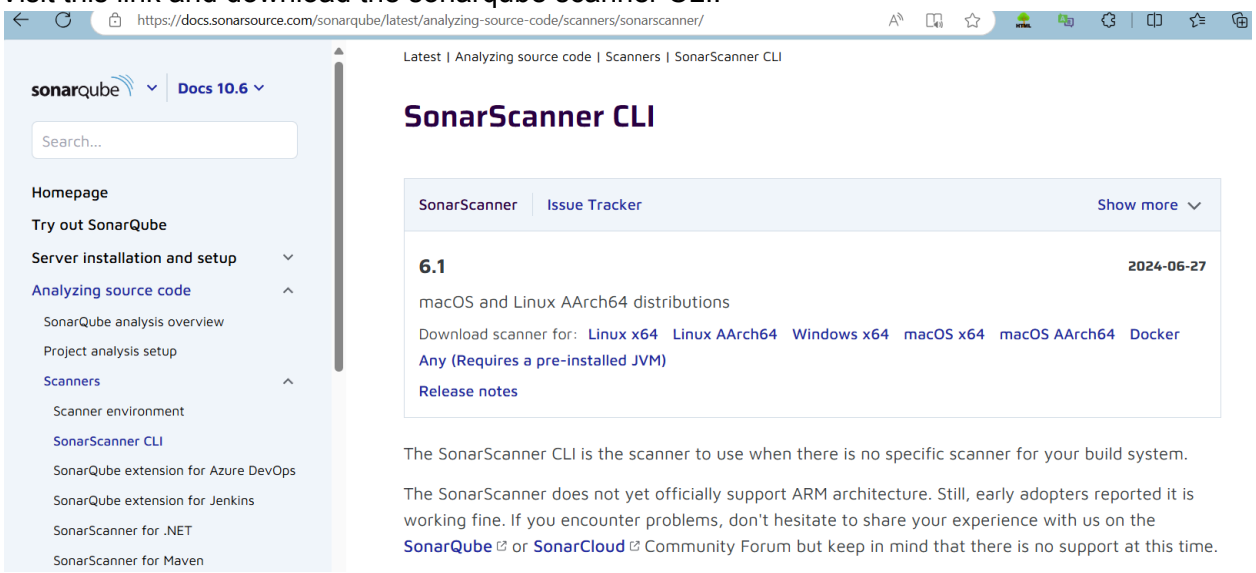
Aim:

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.

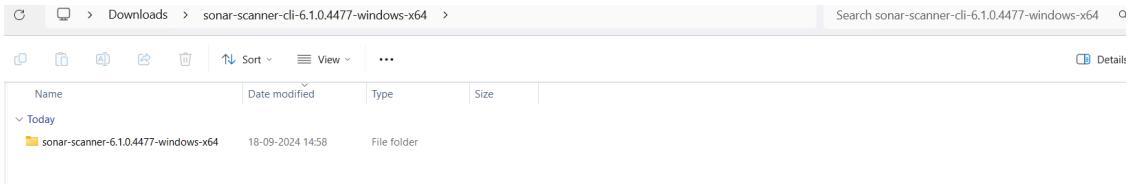
Steps

Step 1: Download sonar scanner

https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/
Visit this link and download the sonarqube scanner CLI.



Extract the downloaded zip file in a folder.



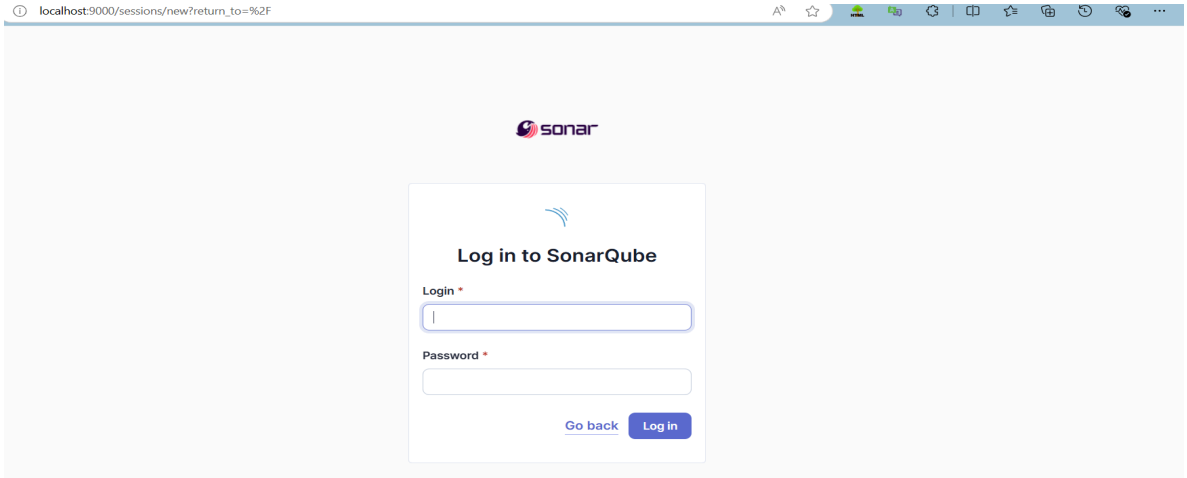
1. Install sonarqube image

Command: **docker pull sonarqube** (skip if already installed we did install it in exp 7)
Then run the image

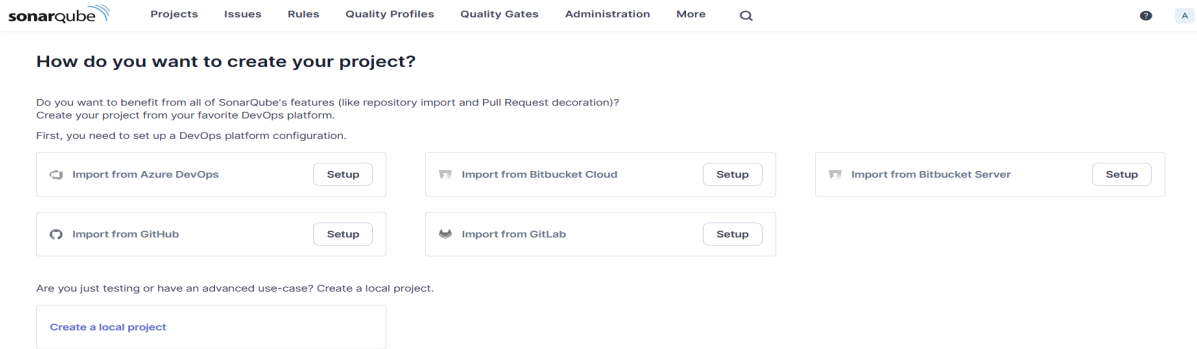
```
C:\Users\Lenovo>docker run -d -p 9000:9000 sonarqube
5007285df5d17d62fef087bc6b74409e37fff333d6308ee62bd323fed5716d5d

C:\Users\Lenovo>
```

Name: Shubham Jha D15C Roll No. 19

2. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.

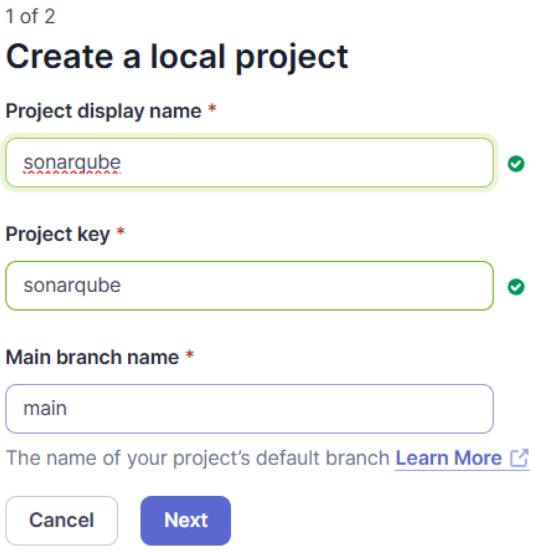


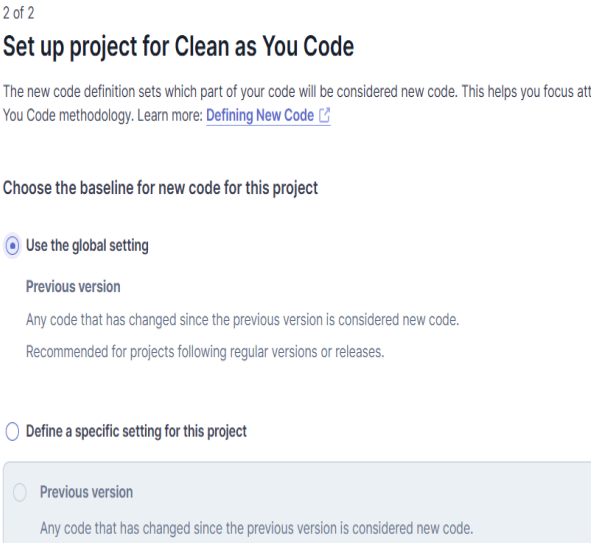
3. Login to SonarQube using username admin and password admin.



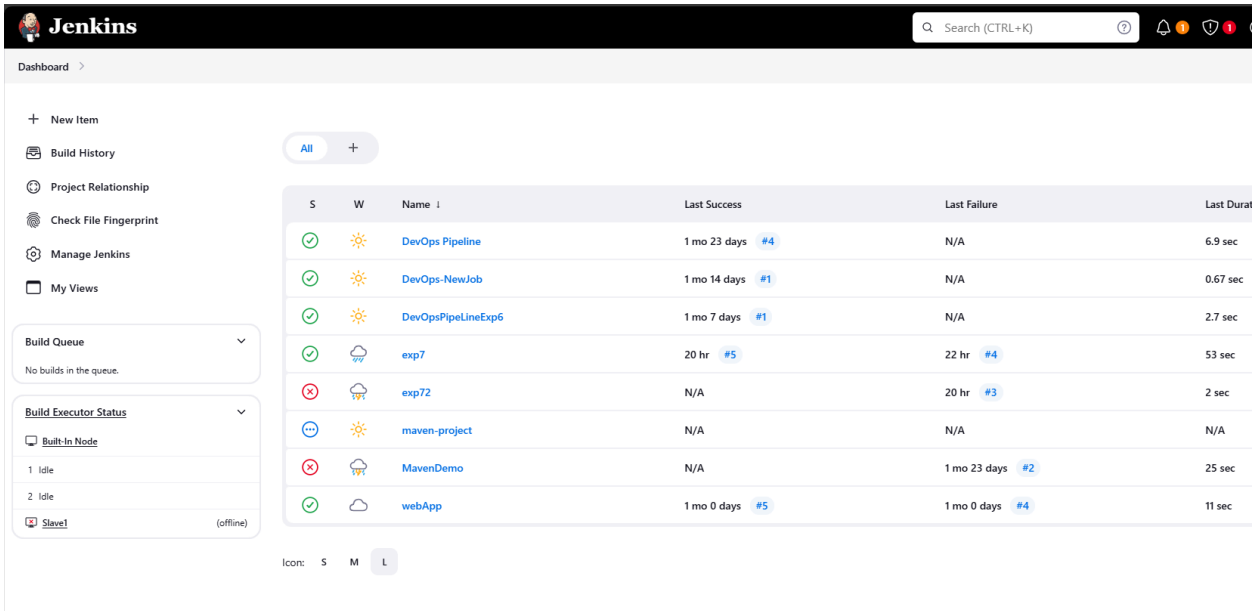
4. Create a local project in SonarQube with the name sonarqube

Name: Shubham Jha D15C Roll No. 19

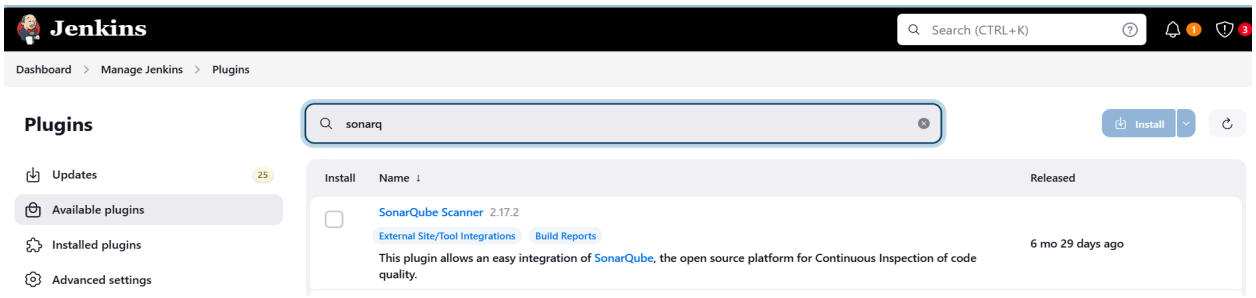


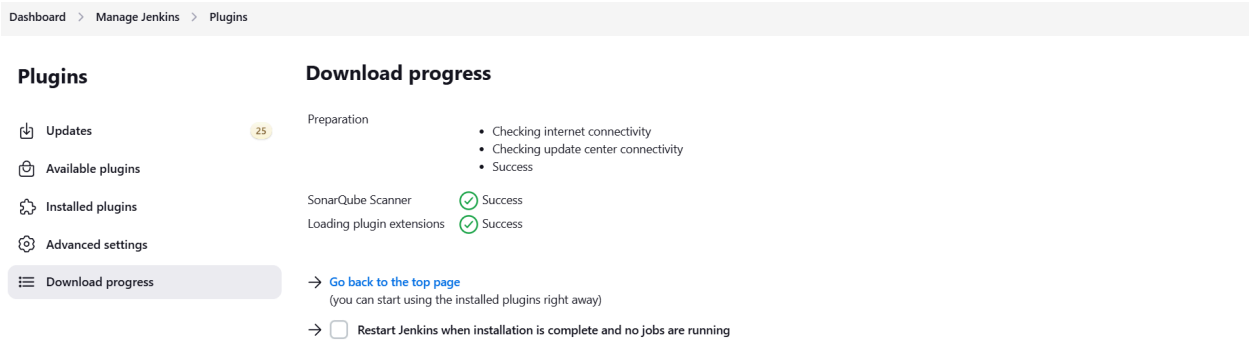


5. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



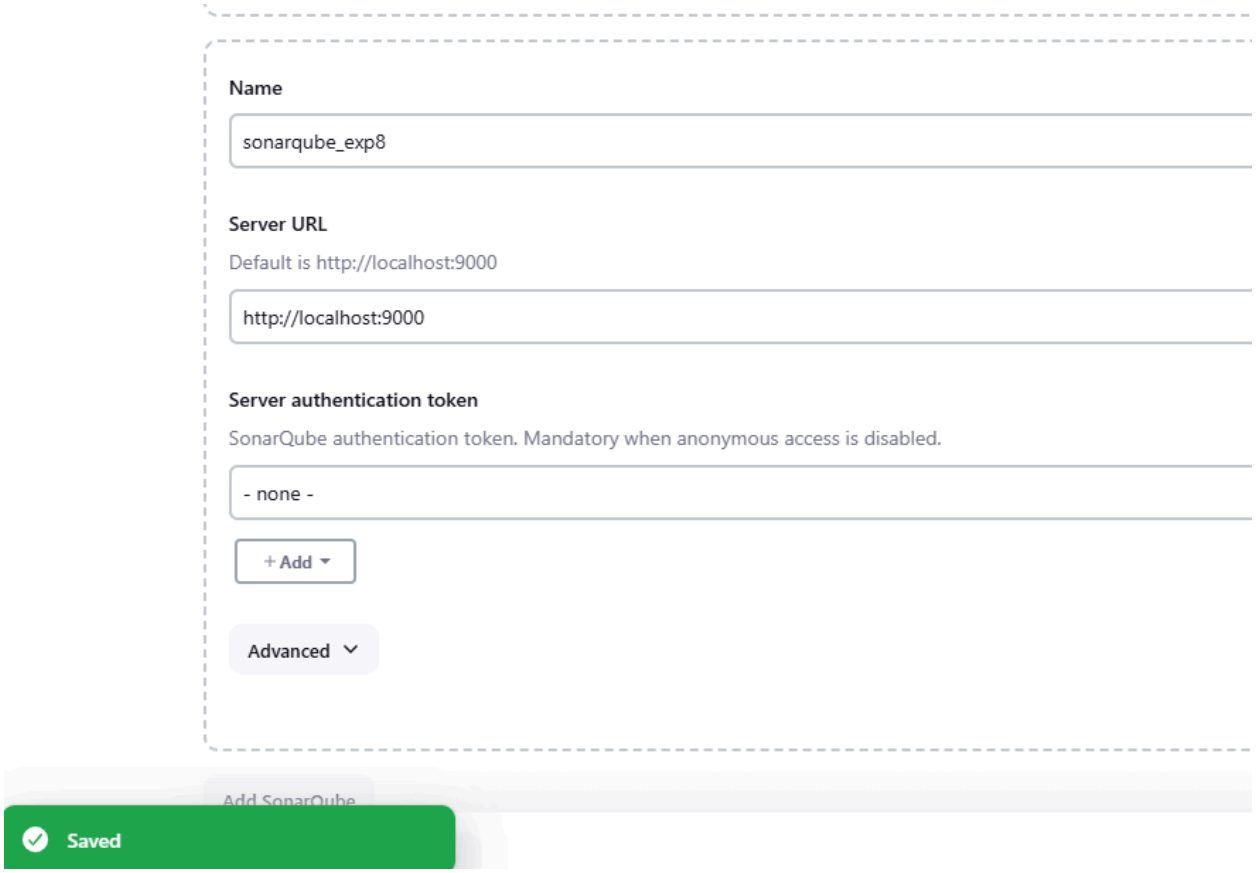
6. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.(we already installed it for exp 7 so you can skip)





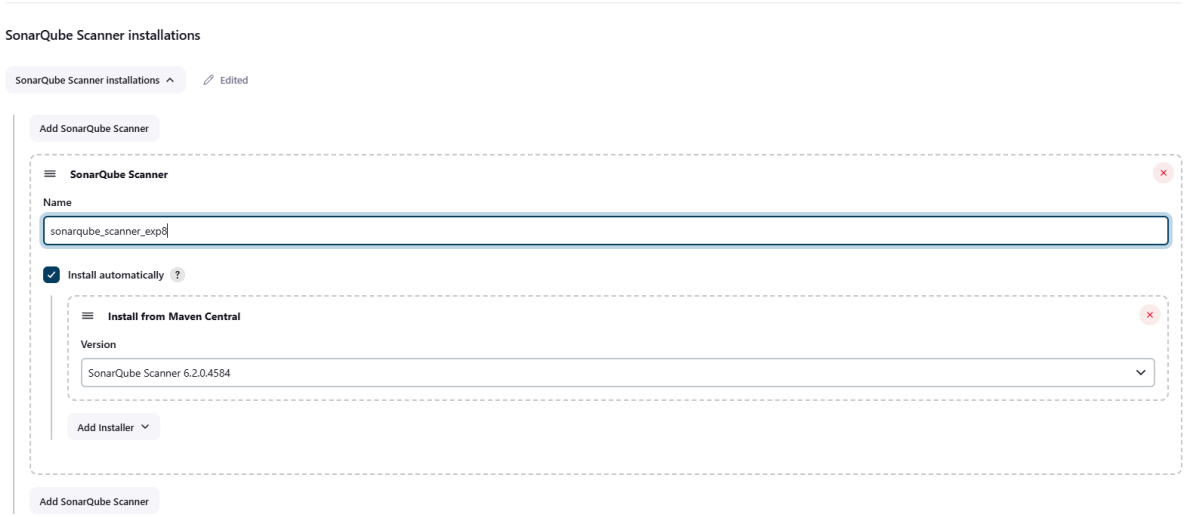
7. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.
Enter the Server Authentication token if needed.(we dont need the token and this step was done in previous exp but jic)

In SonarQube installations: Under **Name** add <project name of sonarqube> for me **sonarqube_exp8**
In **Server URL** Default is **http://localhost:9000**

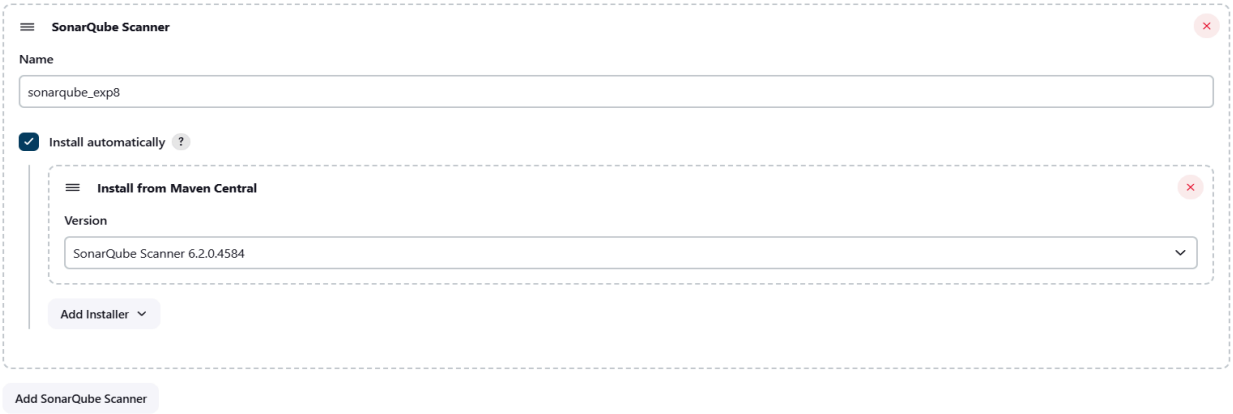


8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

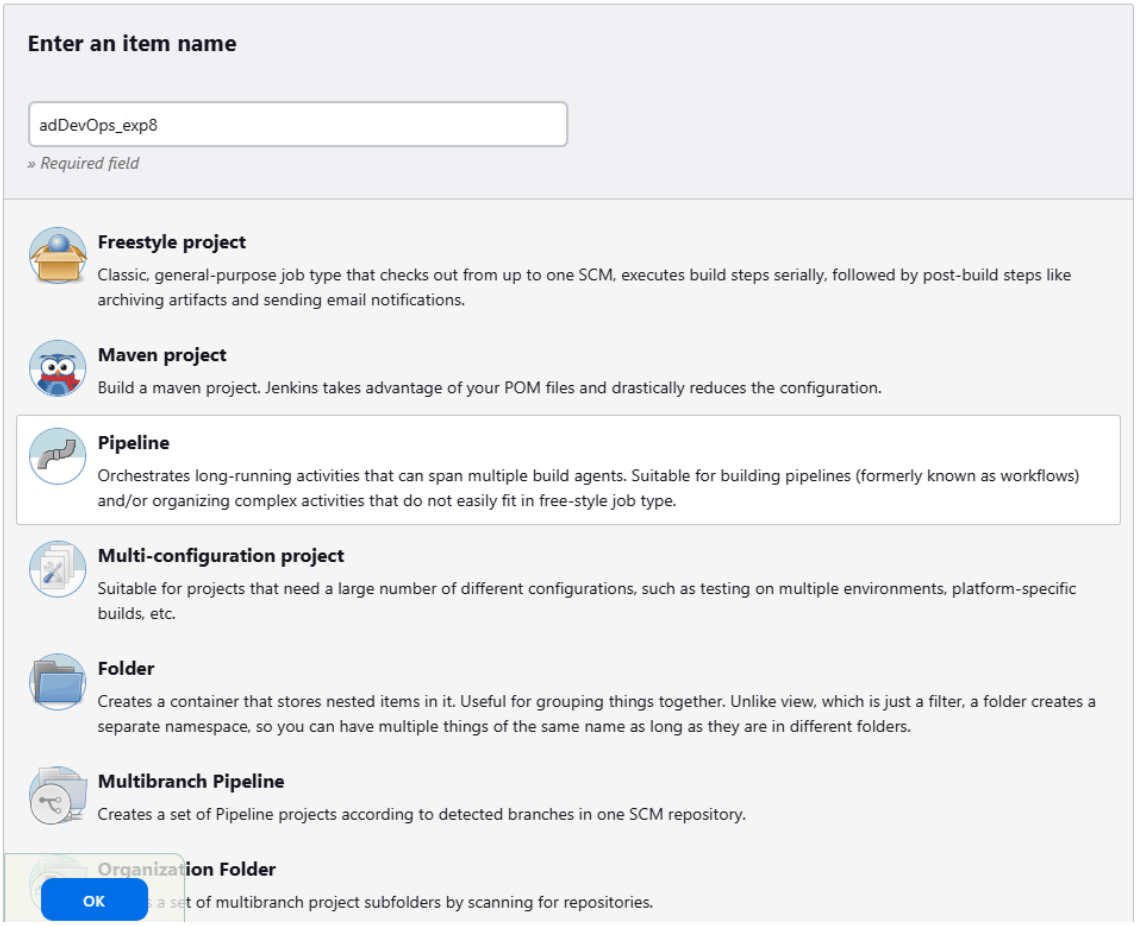
Dashboard > Manage Jenkins > Tools > SonarQube Scanner (i kept the default setting from last experiment and just changed the name.



Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.



9. After configuration, create a New Item → choose a pipeline project.



10. Under Pipeline script, enter the following:

```
node {
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
   }

   stage('SonarQube Analysis') {
      withSonarQubeEnv('exp8') {
         bat """
         "C:\\Program Files\\Sonar
Scanner\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat" ^
         -Dsonar.login=<username> ^
         -Dsonar.password=<password> ^
         -Dsonar.projectKey=<project-key> ^
         -Dsonar.exclusions=vendor/*,resources/,/.java ^
```

```
        -Dsonar.host.url=http://127.0.0.1:9000/
        """
    }
  }
}
```

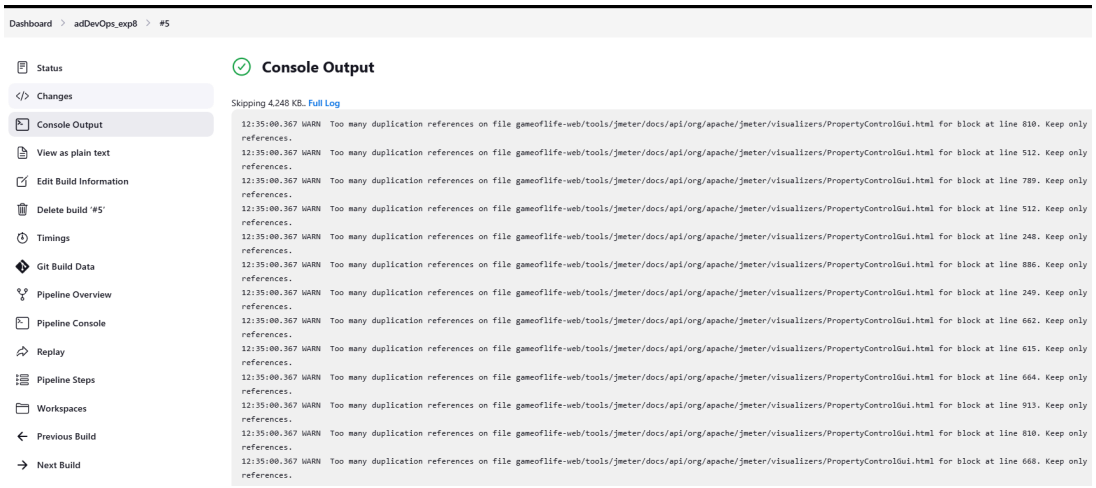
*Note that the code has placeholders

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

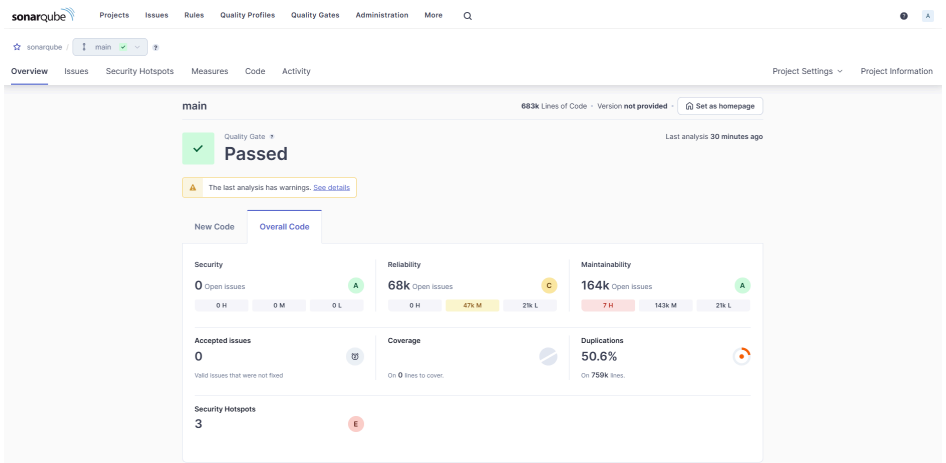
11. Build project

12. Check console

Name: Shubham Jha D15C Roll No. 19

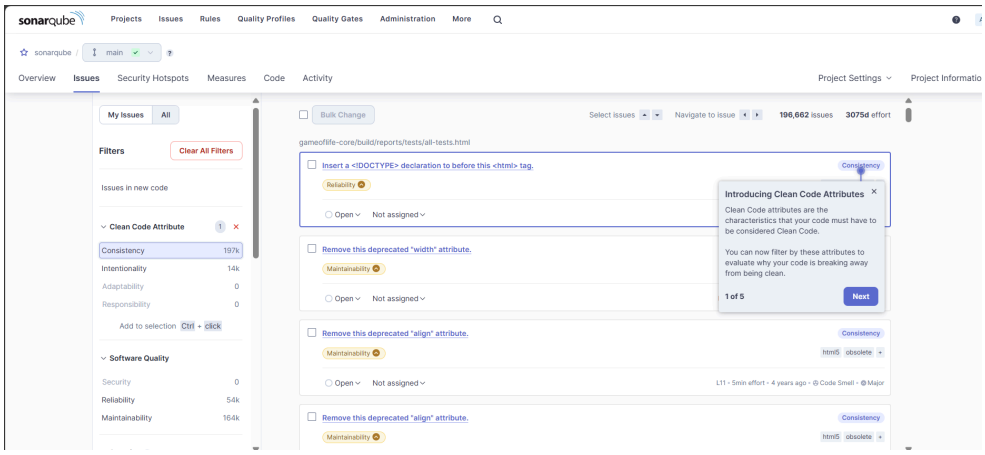


13. Now, check the project in SonarQube



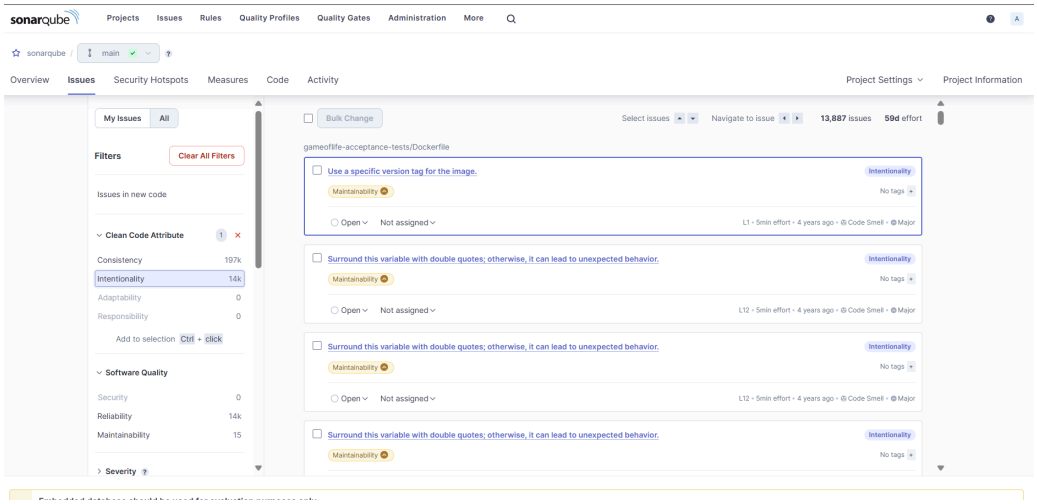
14. Code Problems

- Consistency

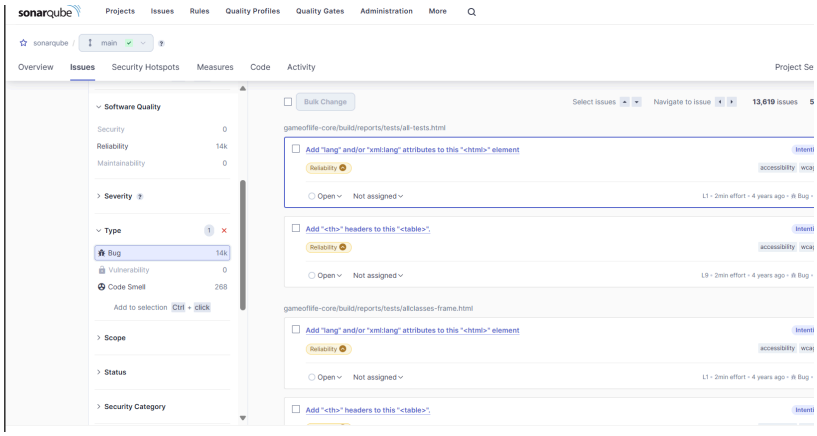


- Intentionality

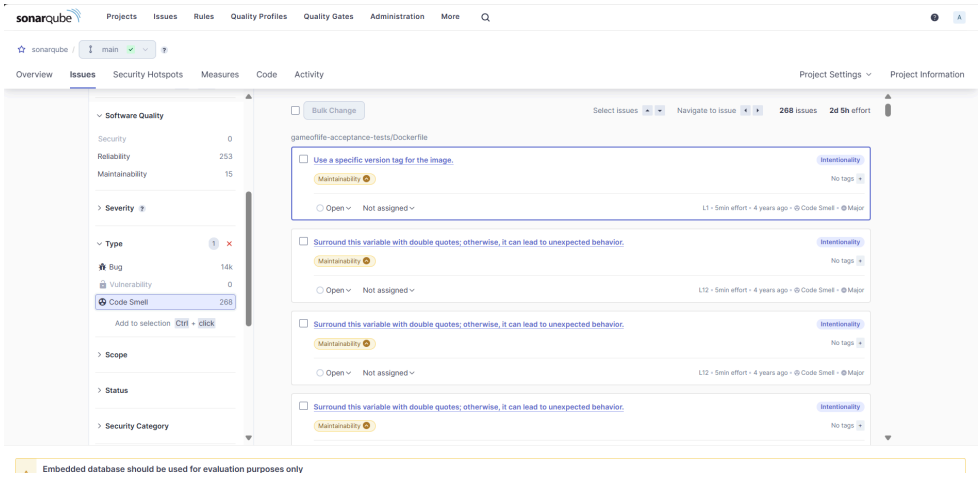
Name: Shubham Jha D15C Roll No. 19



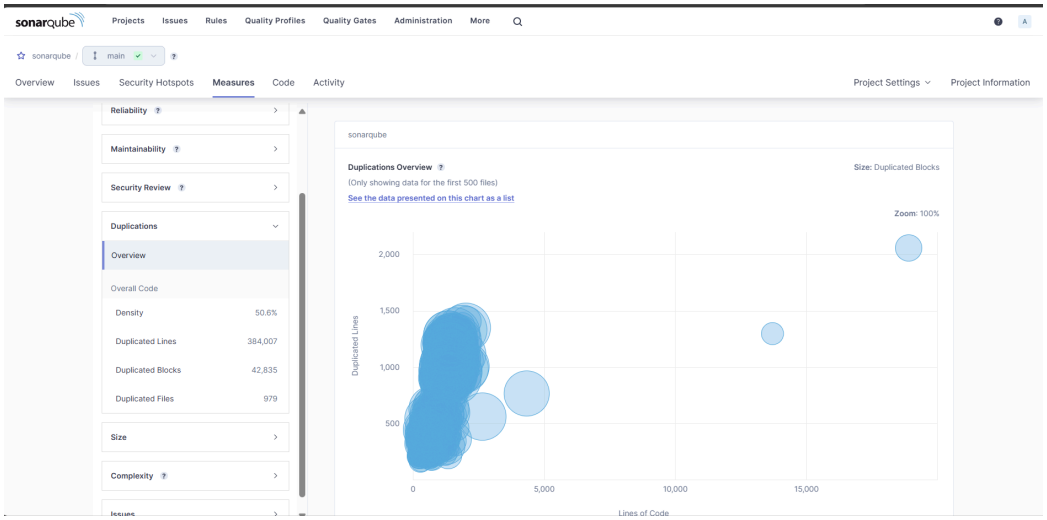
- Bugs



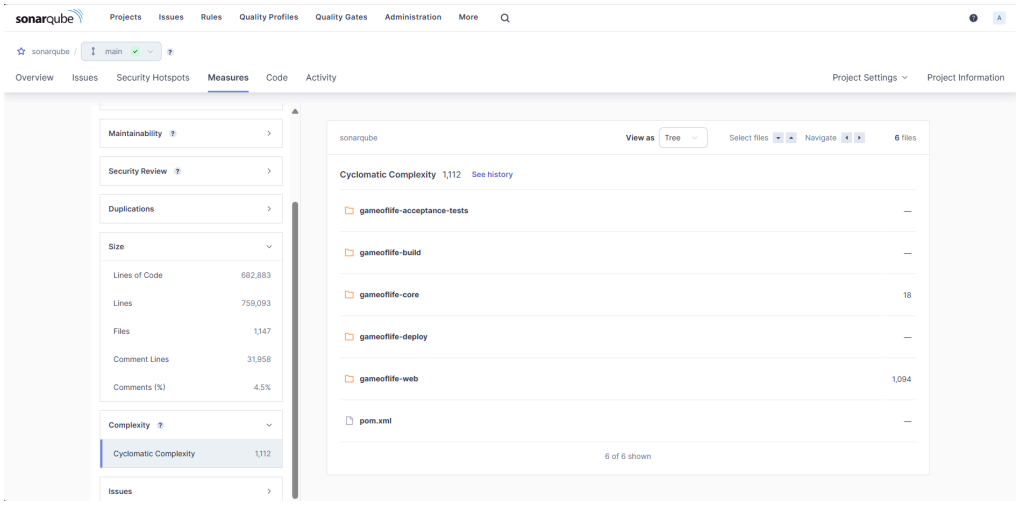
- Code Smells



- Duplications



- Cyclomatic Complexities



In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion:

In this experiment, we successfully integrated Jenkins with SonarQube to automate continuous code quality monitoring within our CI/CD pipeline. This process involved deploying SonarQube via Docker, setting up a project for analysis, and configuring Jenkins with the SonarQube Scanner plugin. After configuring the necessary tools and adding SonarQube server details, we created a Jenkins pipeline that automates cloning from GitHub and running static analysis on the code. This integration allows us to detect potential bugs, code smells, and security vulnerabilities at every stage of development, ensuring improved code quality and streamlined development workflows.