Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Prerequisites:
1) Docker
Run docker -v command.
Use this command to check if docker is installed and running on your system.

```
C:\Users\bhumi>docker -v
Docker version 27.2.0, build 3ab4256
```

2) Install SonarQube image Command:
docker pull sonarqube
This command helps you to install an image of SonarQube that can be used on the local system without actually installing the SonarQube installer.

```
C:\Users\bhumi>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest
```

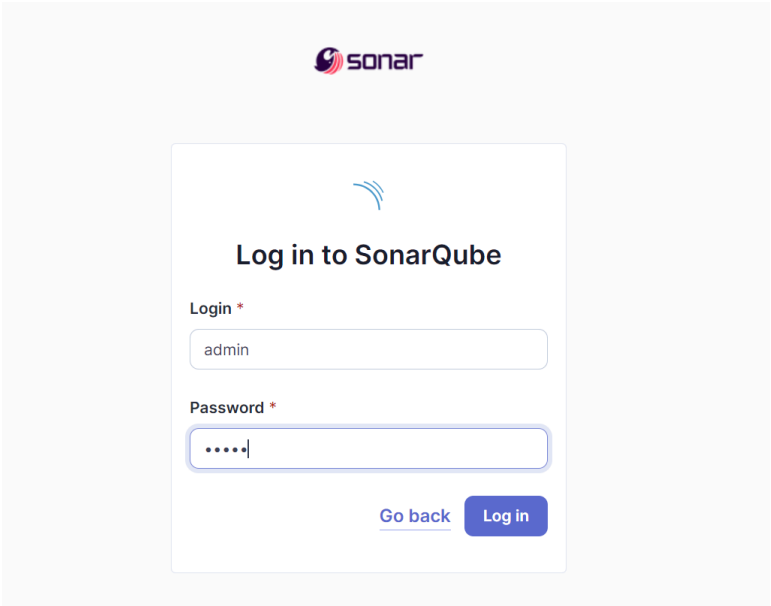
3) Keep jenkins installed on your system.

Experiment Steps:

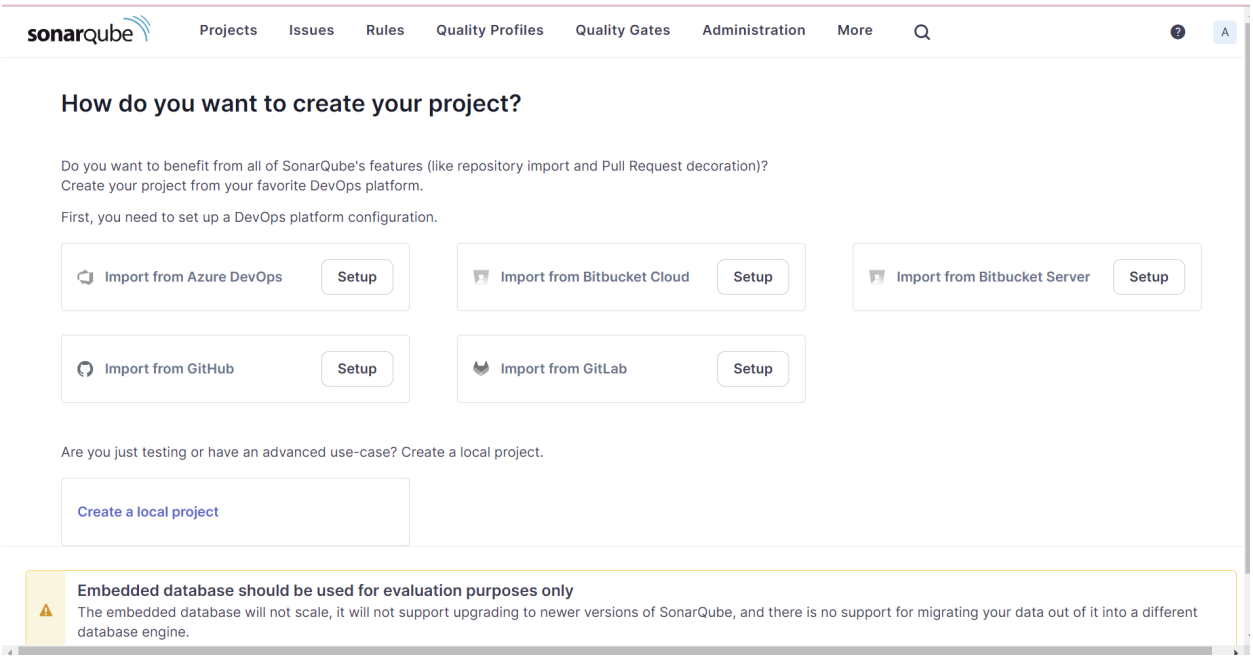
1. Run SonarQube in a Docker container using this command -
   docker run -d --name sonarqube -e
   SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

```
C:\Users\bhumi>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
7401befce9e7a7248c0e5648a2913e99c3843cce08e91d403e5af3a1479a151e
```

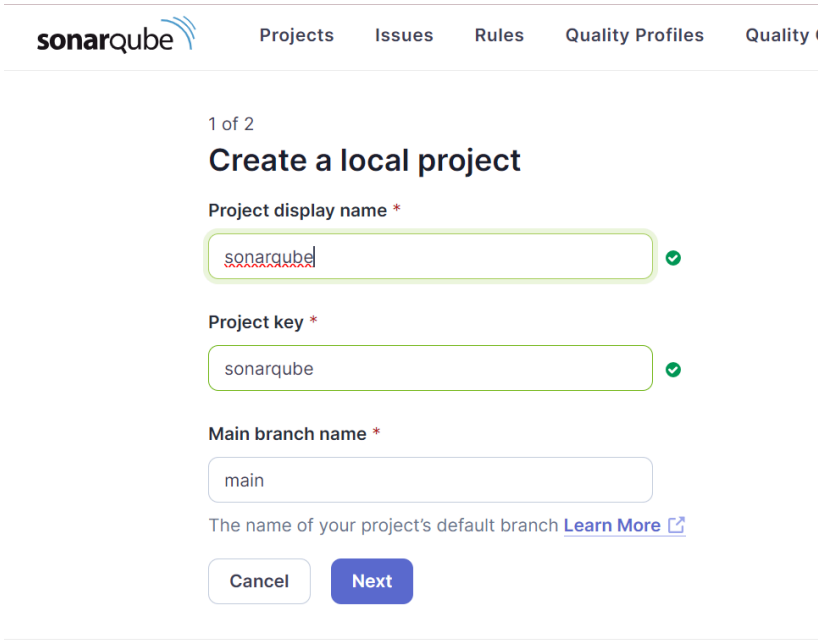
2. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.
3. Login to SonarQube using username admin and password admin.



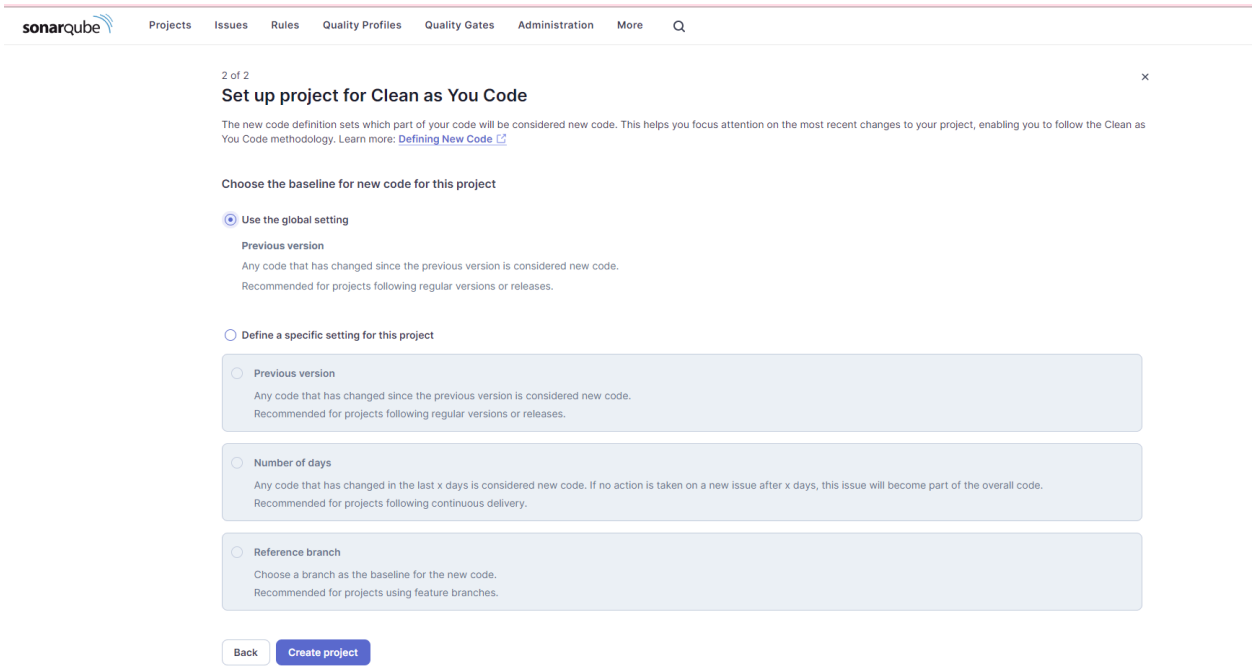
4. Create a manual project in SonarQube with the name sonarqube.



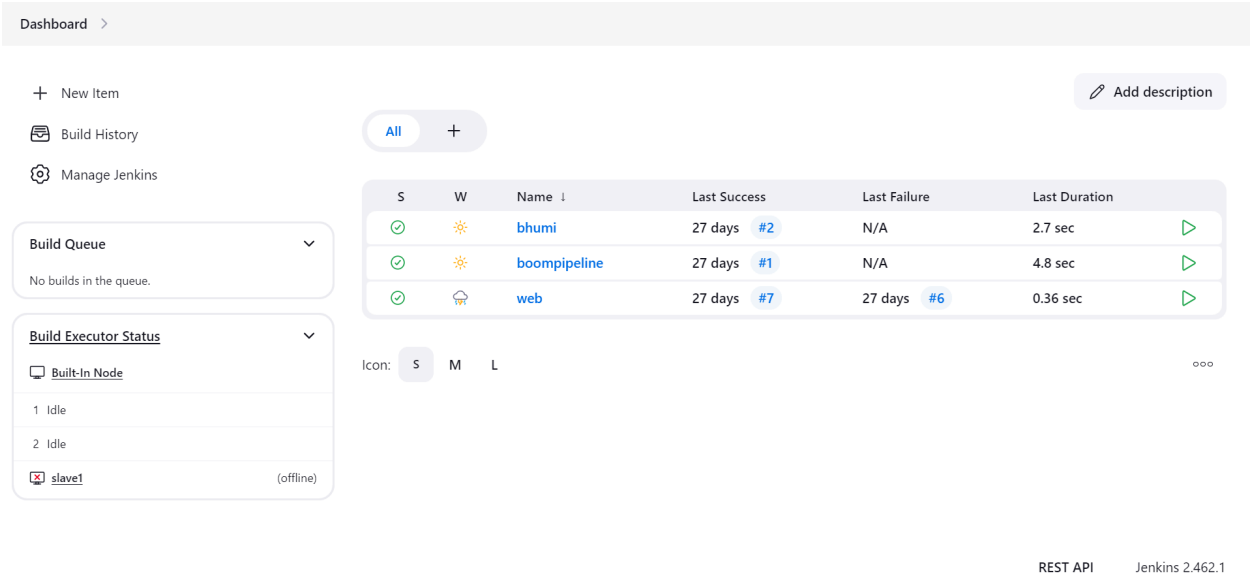
Click on Create a Local Project.
Here, I have given the name ‘sonarqube’. This will be your project key as well. We’ll use it later.
Keep the branch main only and click on next.



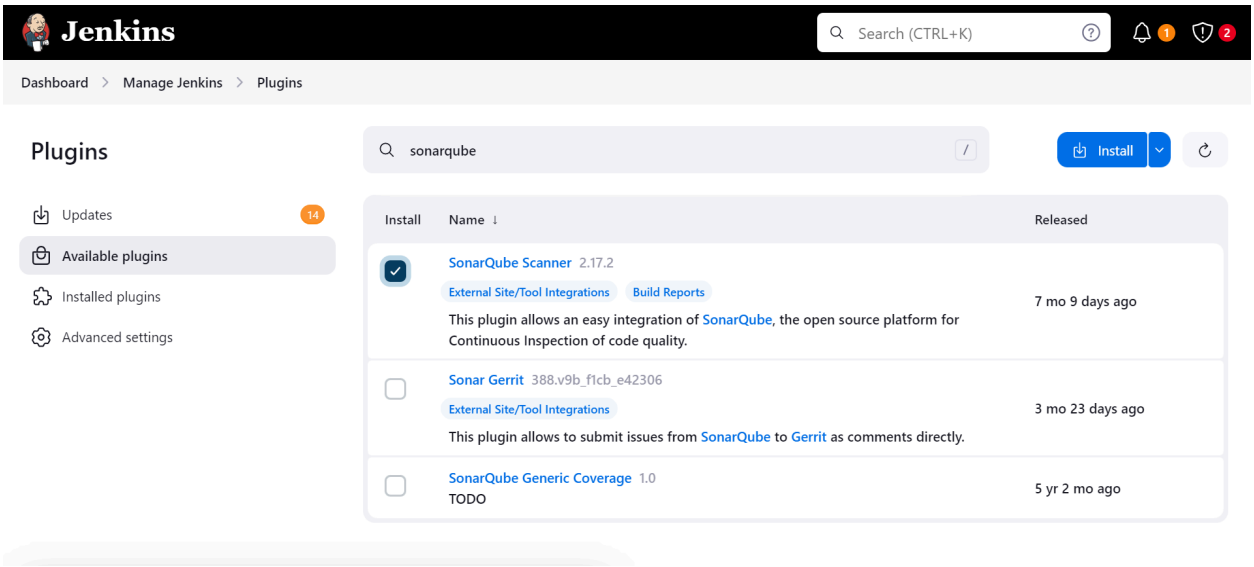
Set up the project as required and click on create.



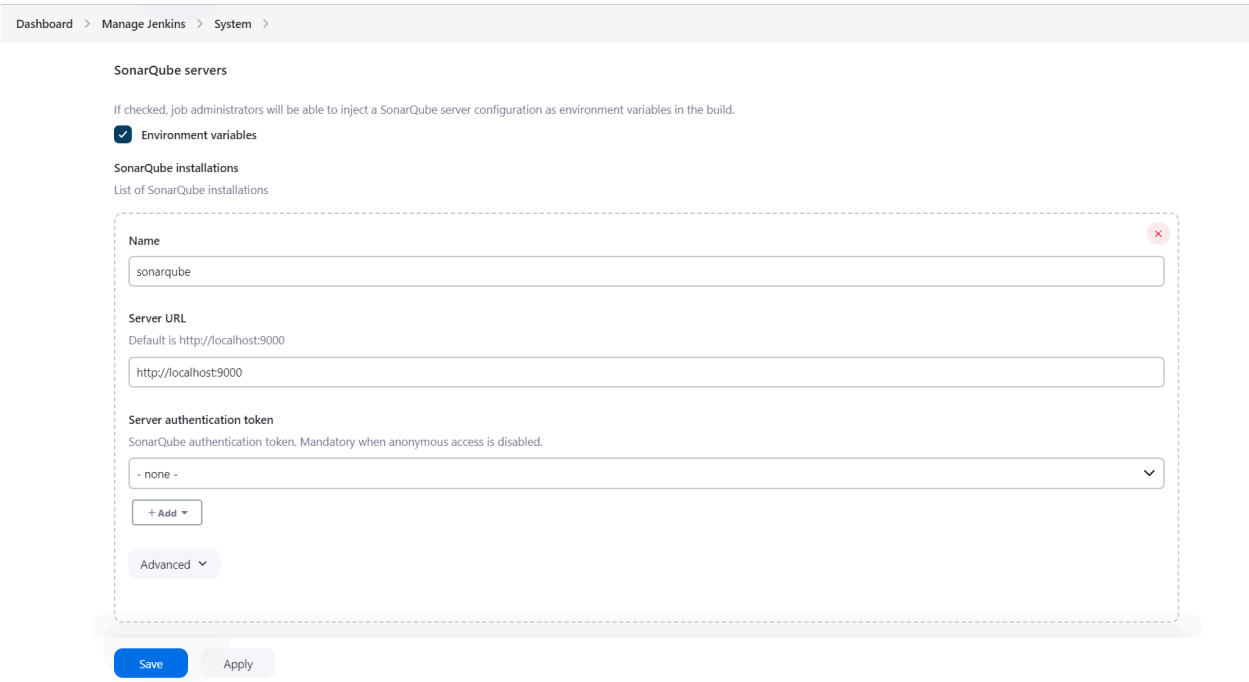
5. Open Jenkins on whichever port it is installed. (http://loaclhost:<port number>).



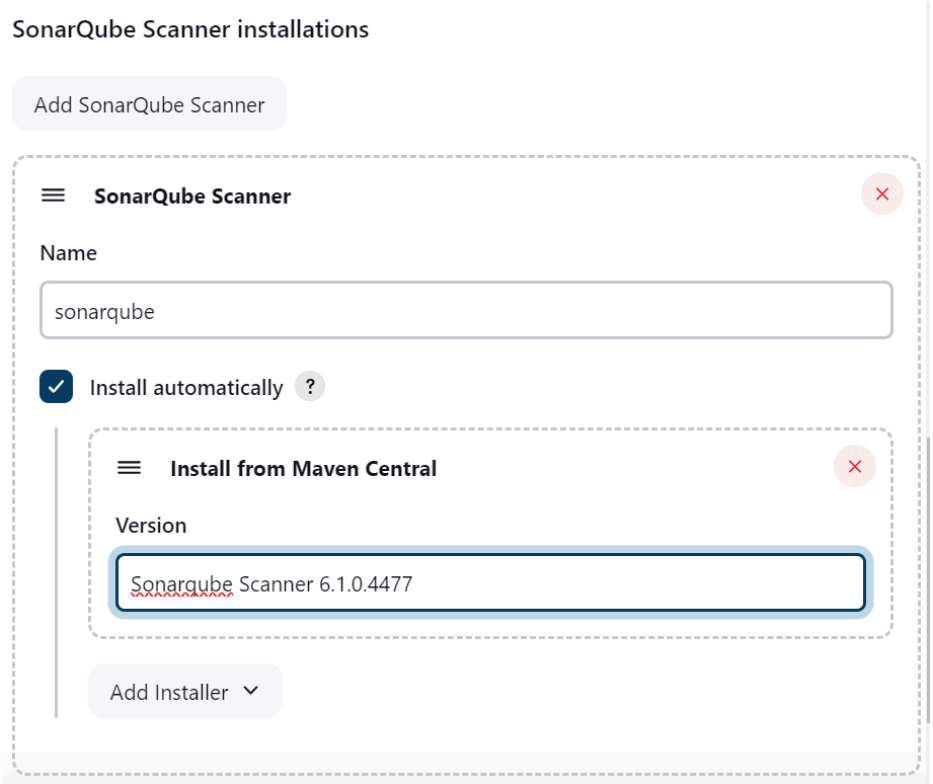
6. Go to Manage jenkins → Plugins → Available Plugins
Search for Sonarqube Scanner for Jenkins and install it.



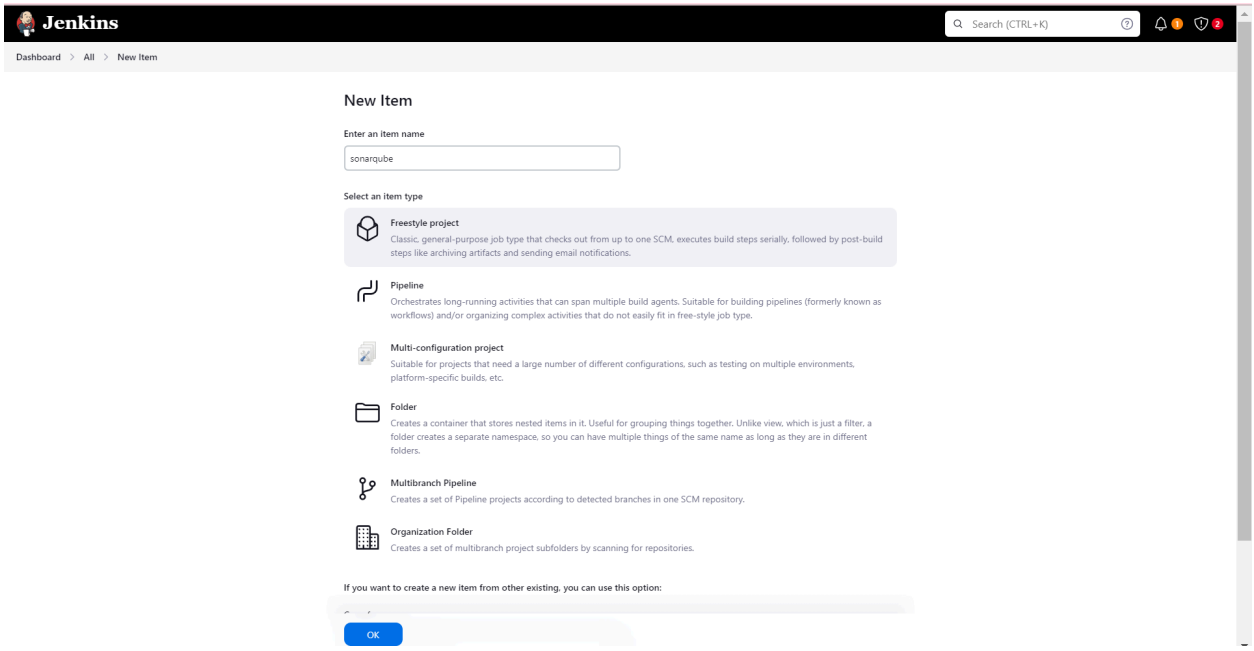
7. Under Jenkins ‘Configure System’, look for SonarQube Servers and enter the details.
I have named the server ‘sonarqube’ and added the server url for jenkins.
Enter the Server Authentication token if needed.



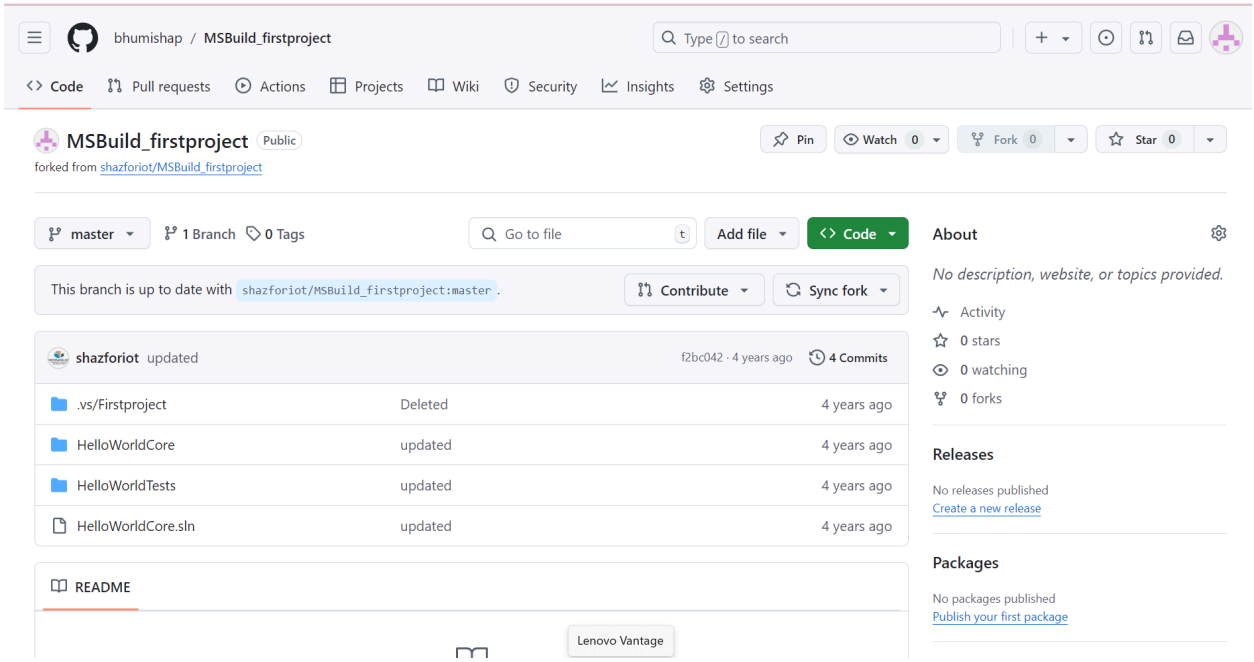
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.



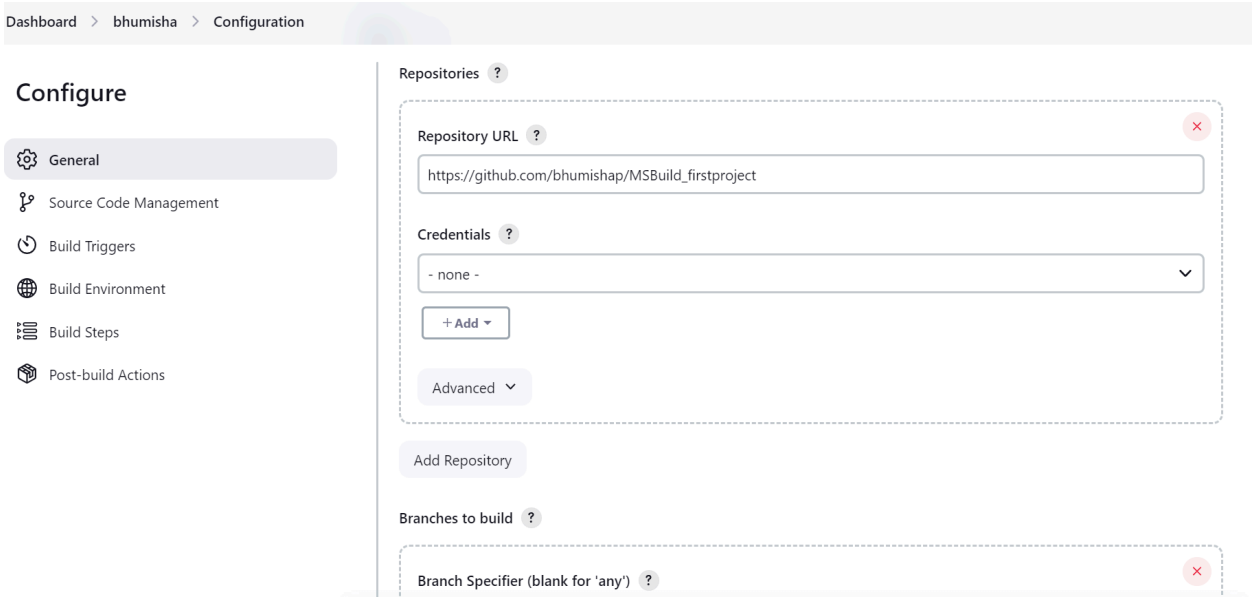
9. After configuration, create a New Item → choose a freestyle project and name your project. Here, I have given the ‘sonarqube’, then click on OK.



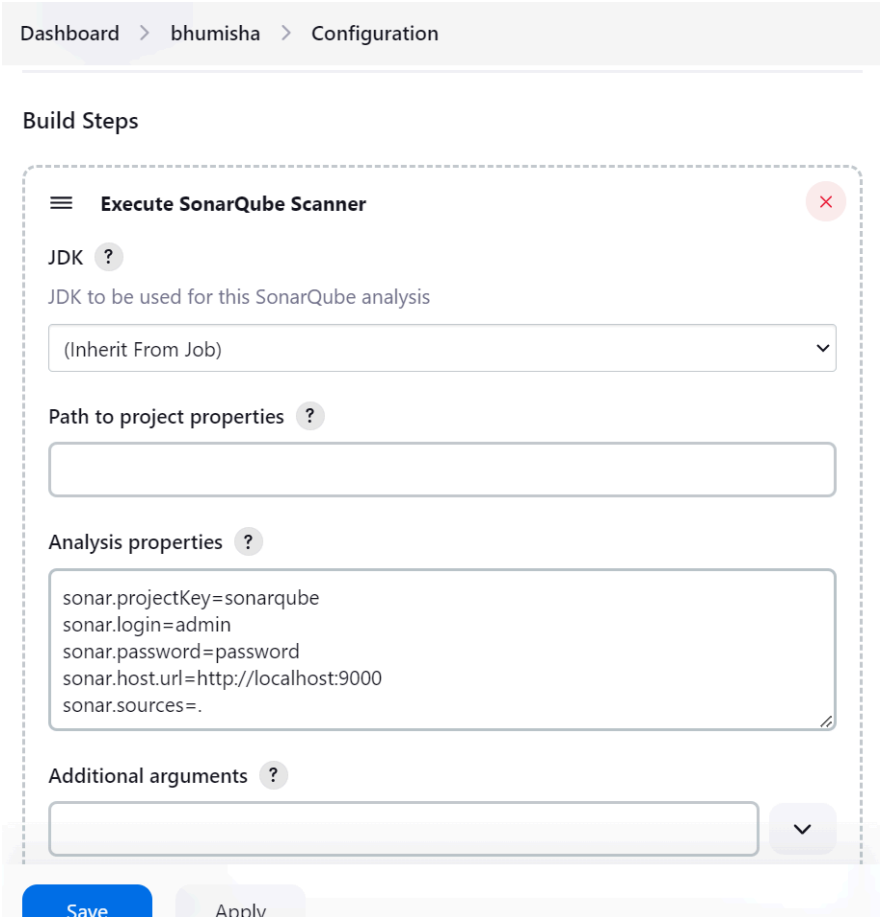
10. Choose the github repository https://github.com/shazforiot/MSBuild_firstproject.
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.
Fork this repository.



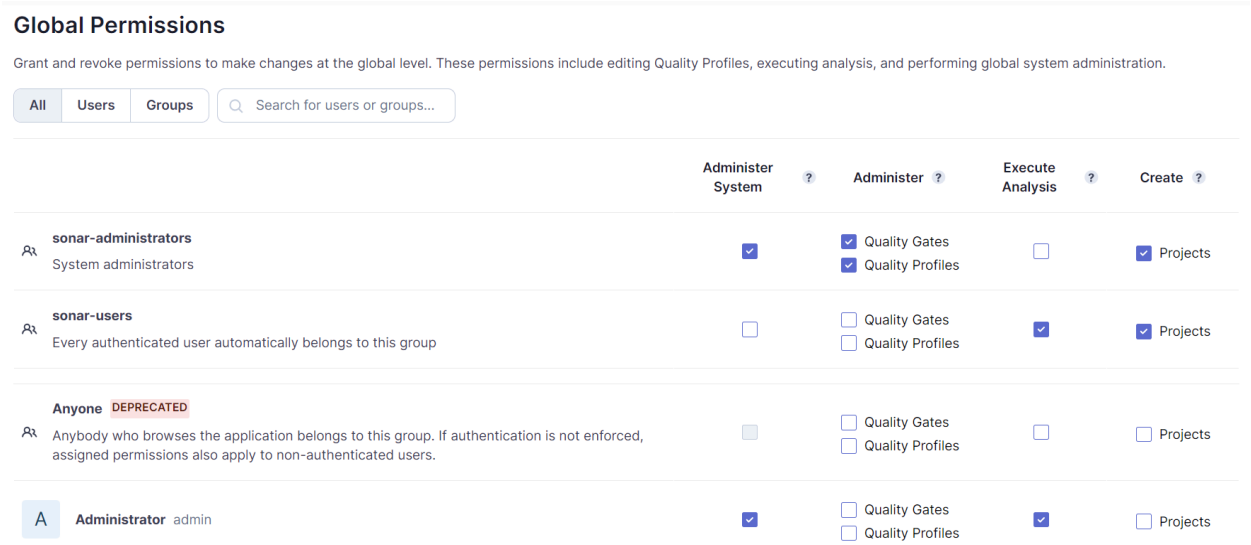
11. Enable git and add the repository you forked.



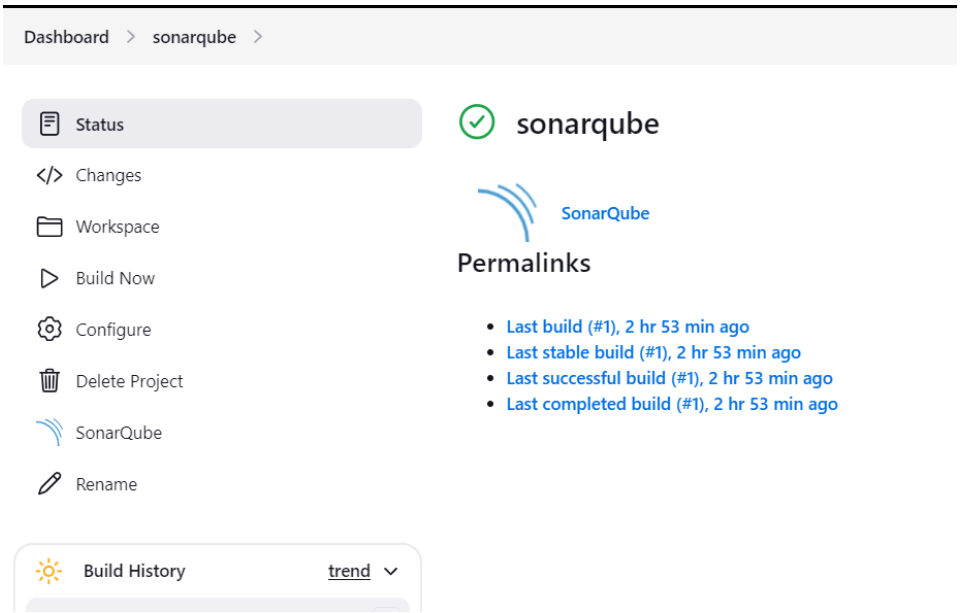
12. Under Build Steps, select Execute Sonarqube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



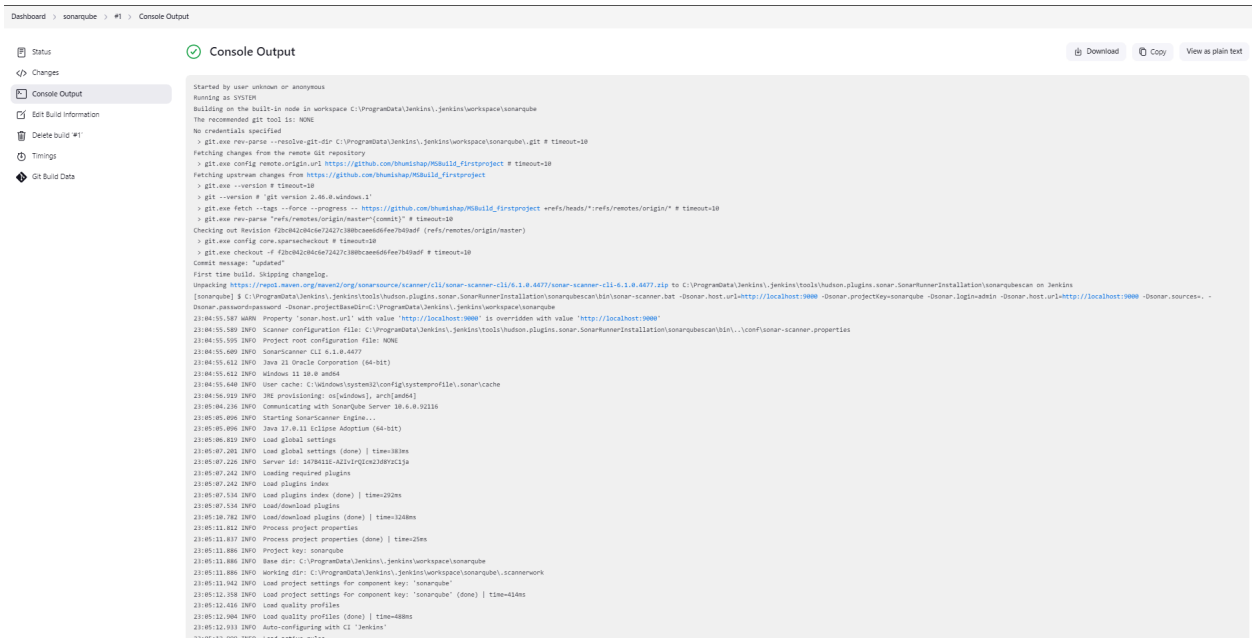
13. Then click on save.Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.

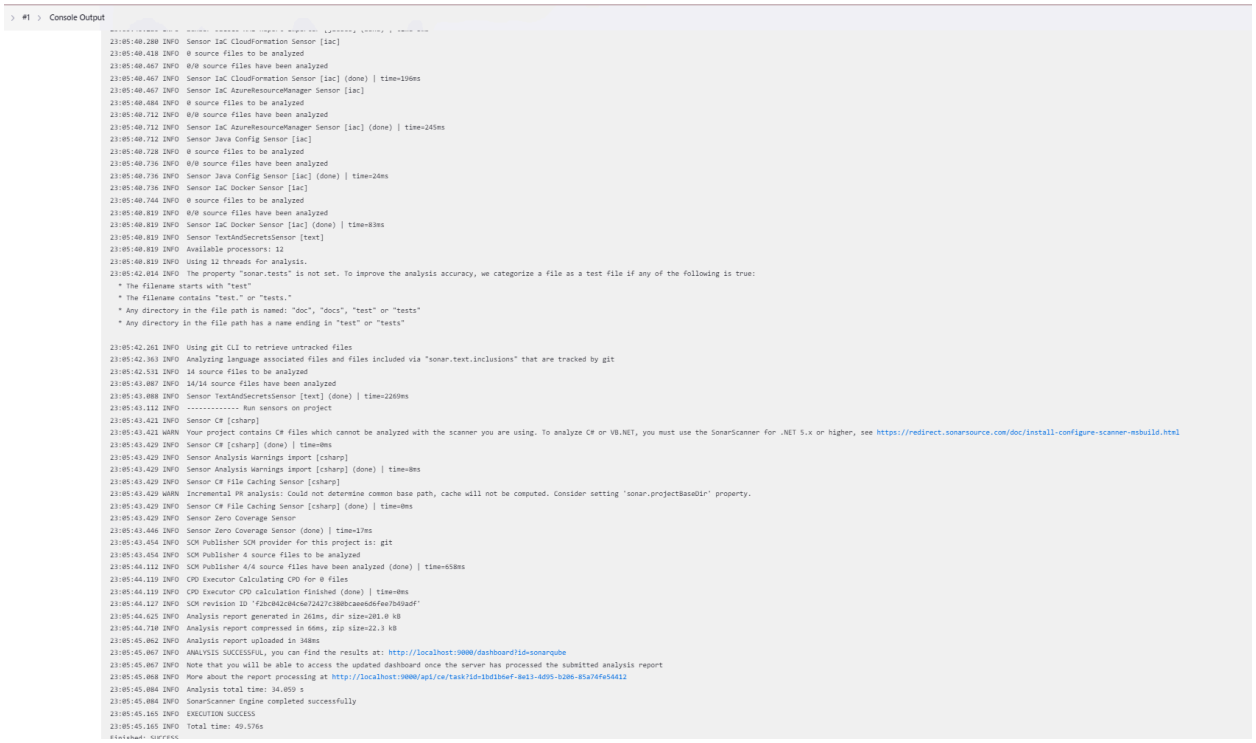


14. Go back to jenkins. Go to the job you had just built and click on Build Now.

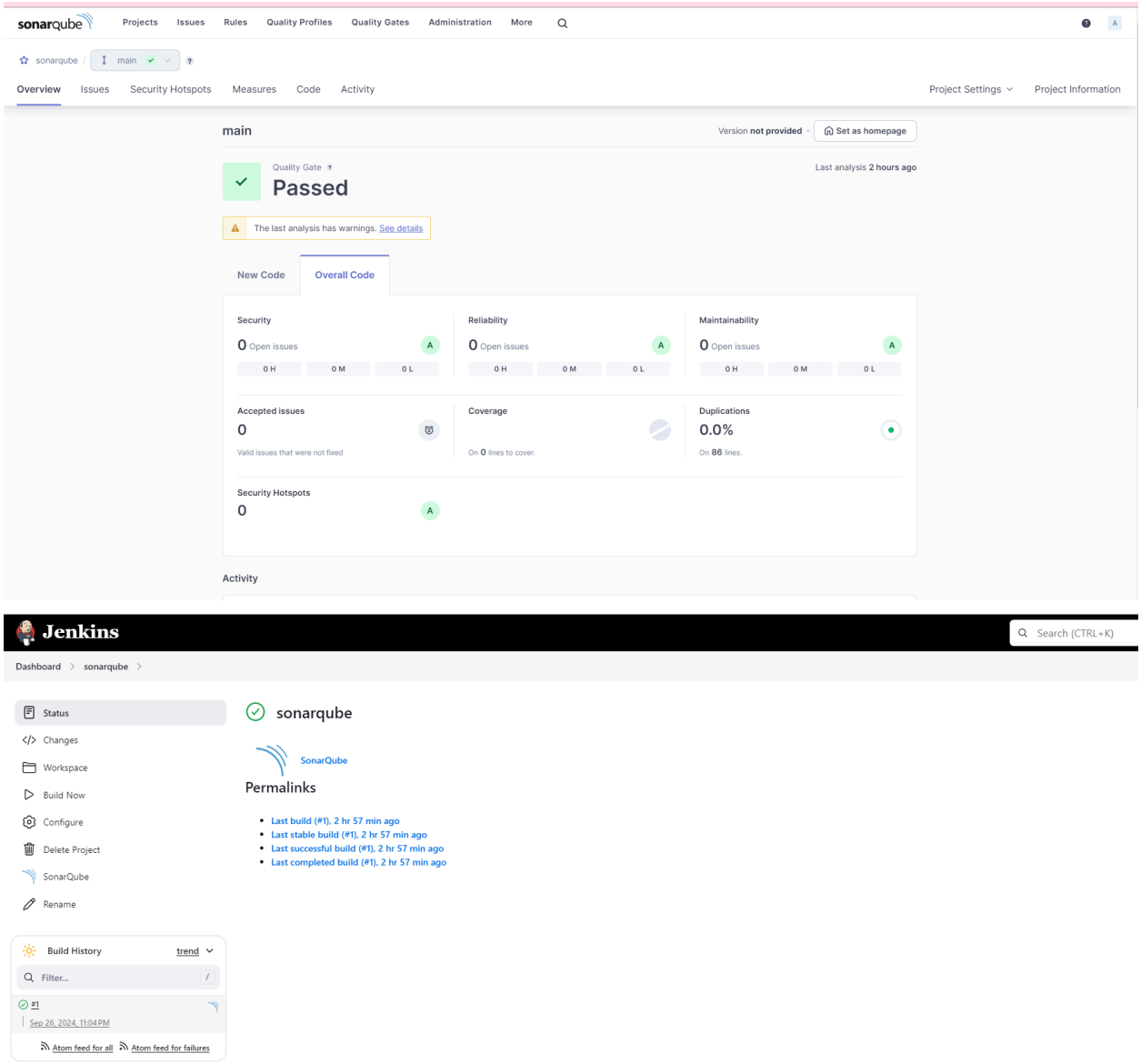


15. Check the console Output.





16. Once the build is complete, go back to SonarQube and check the project linked.



Conclusion:

In this experiment, we explored how to perform Static Application Security Testing (SAST) in Jenkins using SonarQube. Instead of installing SonarQube directly on the local system, we utilized a Docker image for a more streamlined setup. After configuring Jenkins with the necessary SonarQube settings, we integrated it with a GitHub repository containing code for analysis. Upon building the project, SonarQube successfully scanned the code, confirming that no errors were present. This process demonstrated how Jenkins and SonarQube can be effectively used together for automated code quality and security analysis.