Name: Anish Kulkarni Roll No.: 29 Class: D15C AY: 2024-25

Experiment 7

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

Steps:

Step 1: Install a SonarQube image by running the ‘docker pull sonarqube’ command on your terminal. This allows for a Sonarqube image to be used on a local machine without having to install the SonarQube application.

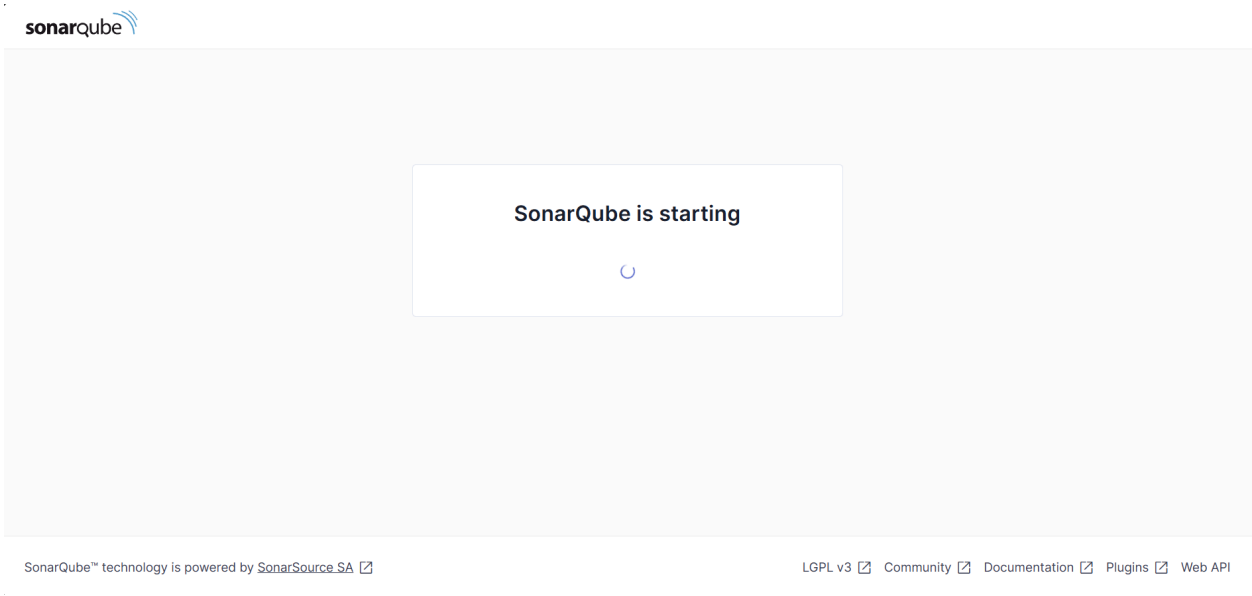
```
PS C:\Users\anish\OneDrive\Desktop\Adv DevOps 7> docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete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

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview sonarqube
```

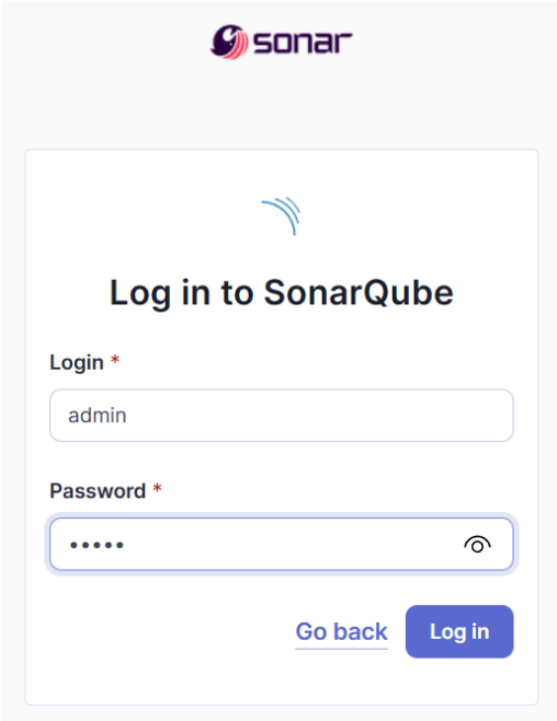
Step 2: Execute the following command:
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
This command will run the SonarQube image that was just installed using docker.

```
PS C:\Users\anish\OneDrive\Desktop\Adv DevOps 7> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:900
0 sonarqube:latest
dce67335909e42d81ec64d3ef0c5e5e2c36cc7ed36d87088033121ae1544f4fb
```

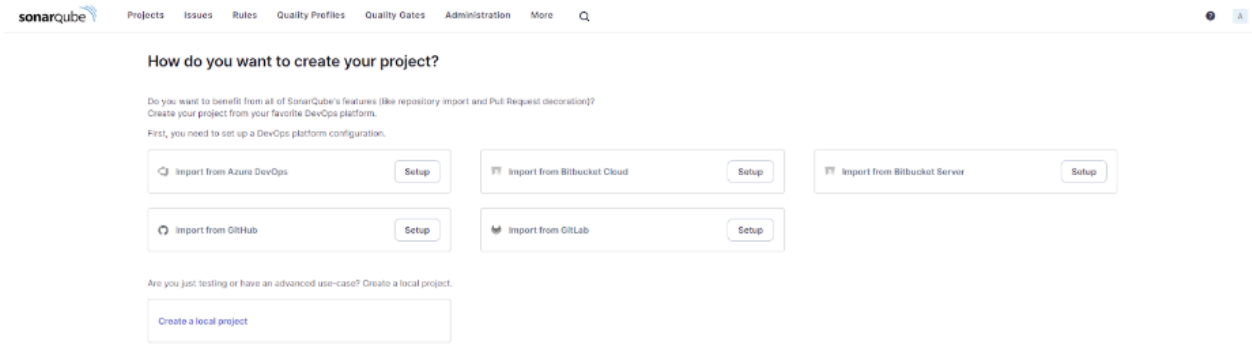
Step 3: Go to http://localhost:9000 on your browser and check if SonarQube is starting or not.



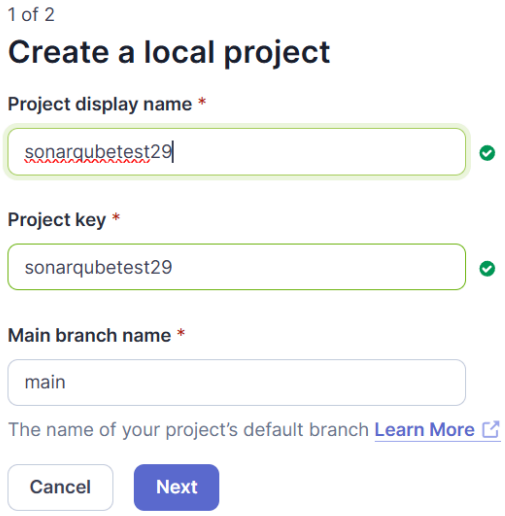
Step 4: On the login page, enter ‘Login’ as admin and ‘Password’ as admin to log in initially. It then asks you to change the password to a password of your choice. Do the same and proceed to the next step.



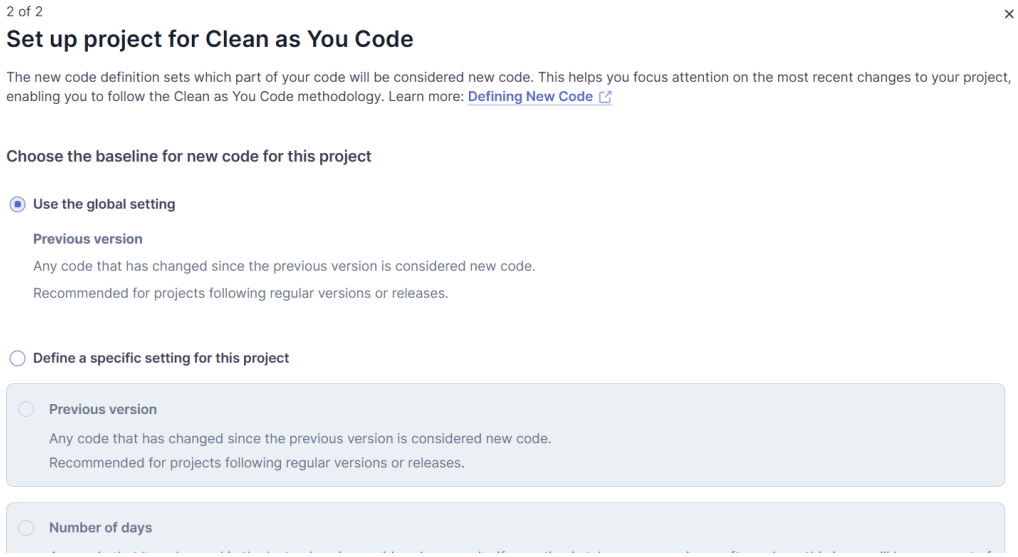
Step 5: On the SonarQube dashboard, click on ‘Create a local project’.



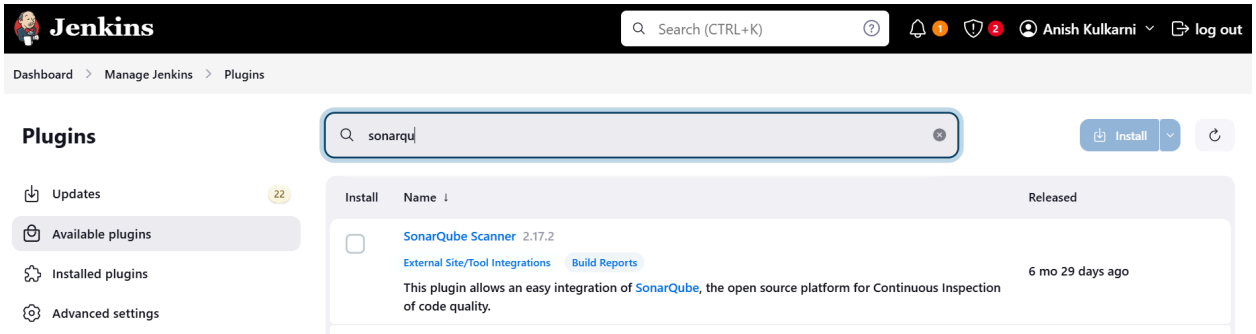
Step 6: Create a local project by entering the project name and key and click on ‘Next’.



Step 7: Set up your project and click on ‘Create project’.

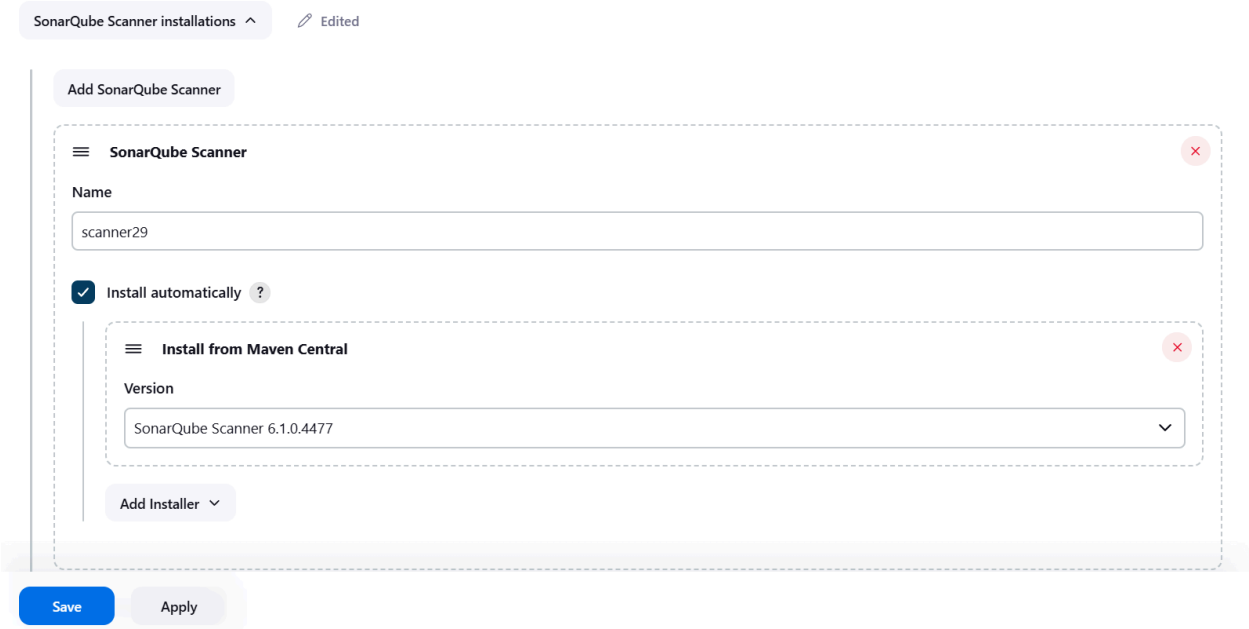


Step 8: Navigate to your Jenkins server (on whichever port it has been installed), click on ‘Manage Jenkins’, click on ‘Plugins’ and search for the ‘SonarQube Scanner’ plugin and install it.

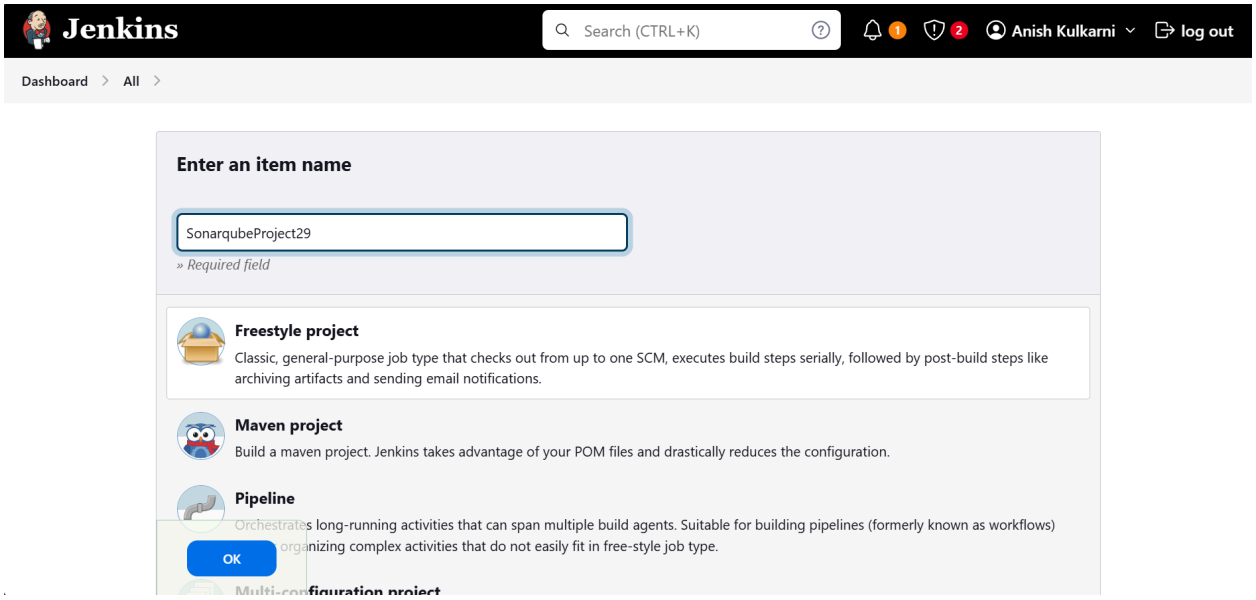


Step 9: Under ‘Manage Jenkins’, click on System. Under the ‘Sonarqube installations’ section, add a server and add a server authentication token if needed.

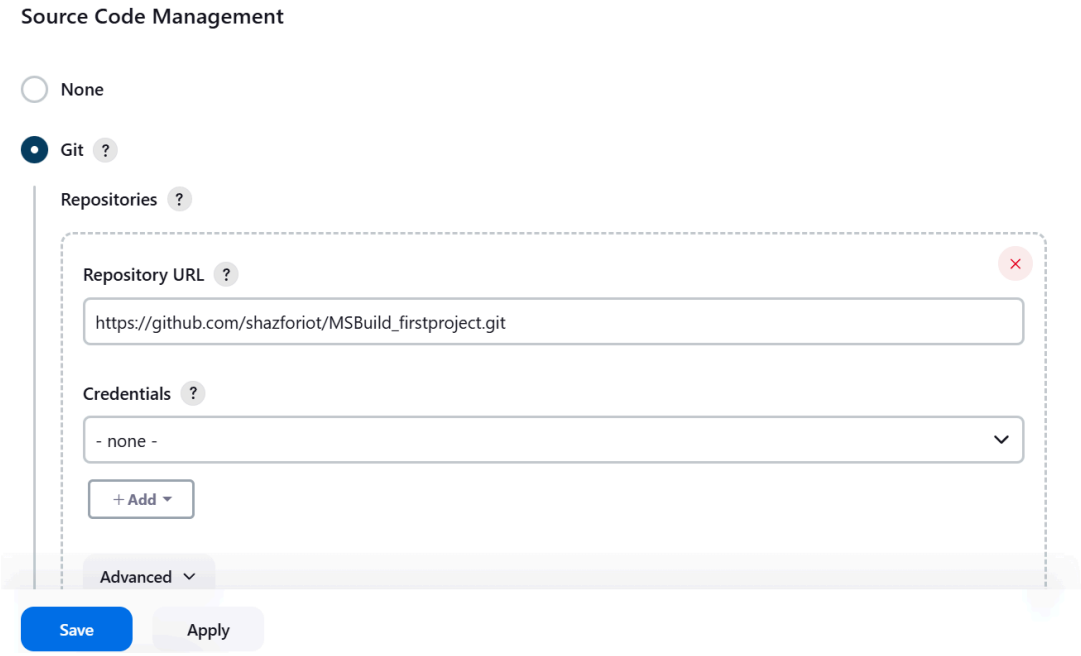
Step 10: Under ‘Manage Jenkins’, click on ‘Tools’. Under the ‘SonarQube Scanner installations’ section, give your scanner a name, choose the latest version and click on ‘Install automatically’.



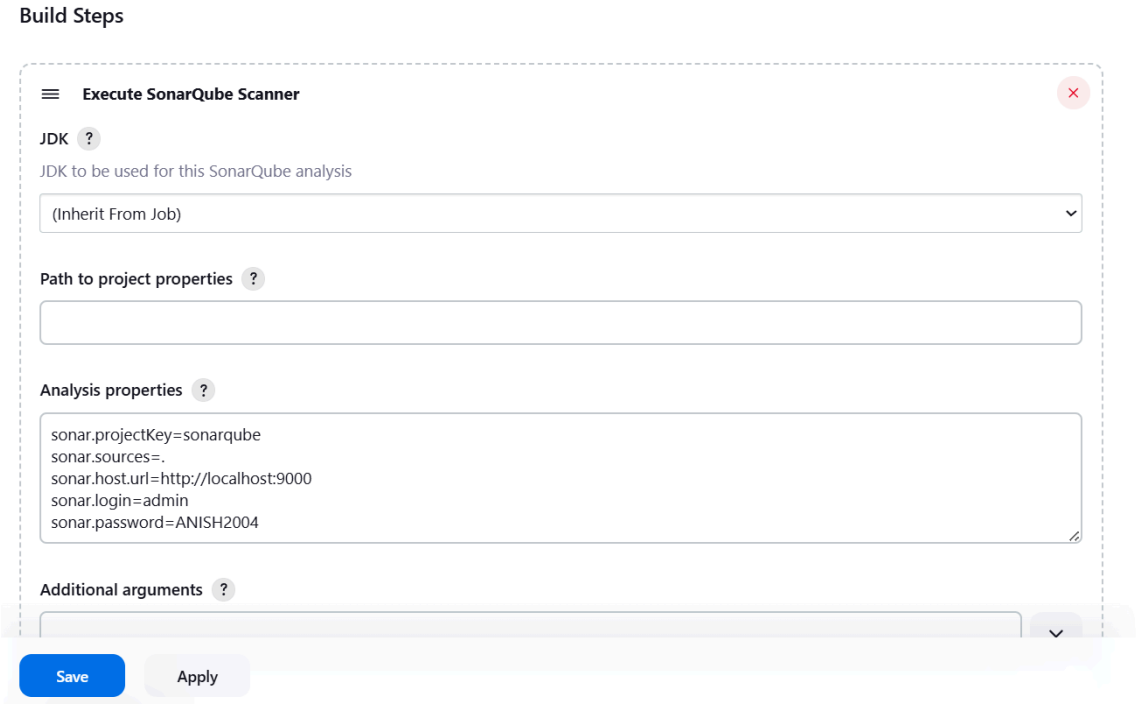
Step 11: Create a new Jenkins project by giving it a name and ensure that it is a freestyle project.



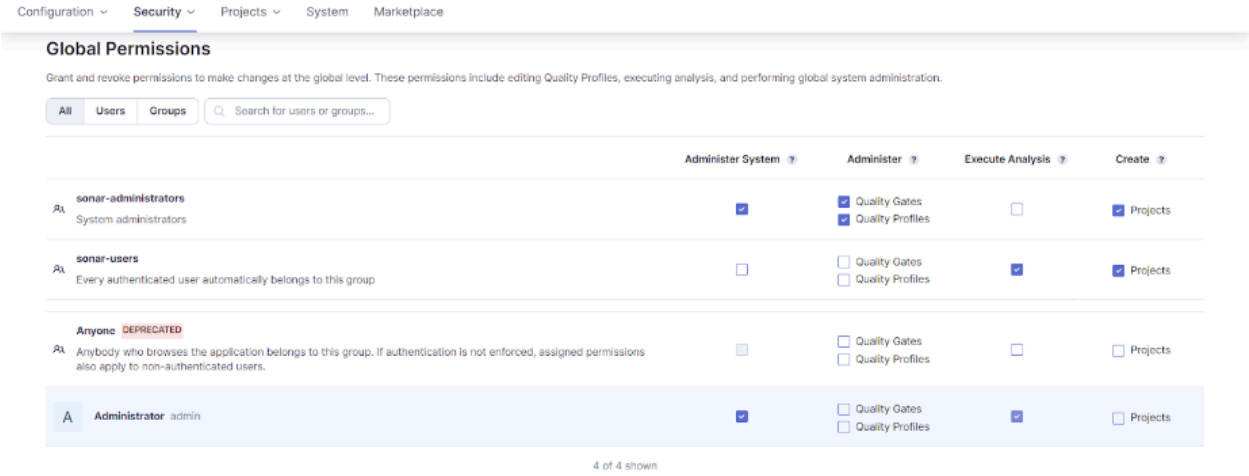
Step 12: In ‘Source Code Management’ section, choose ‘Git’ and enter the following repository URL:-
https://github.com/shazforiot/MSBuild_firstproject
The above is a sample hello-world project with no vulnerabilities.



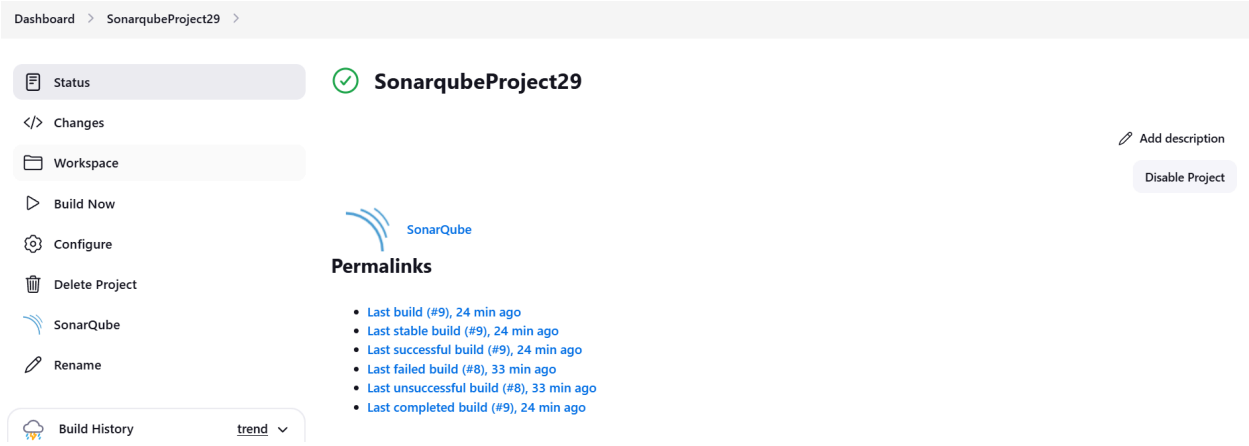
Step 13: Under Build Steps, enter Sonarqube Scanner and enter these analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



Step 14: On your browser, go to http://localhost:<port_number>/admin/permissions and check the ‘Execute Analysis’ checkbox for Administrator. This gives the required permissions to the user for the analysis stage on SonarQube.



Step 15: Navigate to your Jenkins project and click on ‘Build Now’.

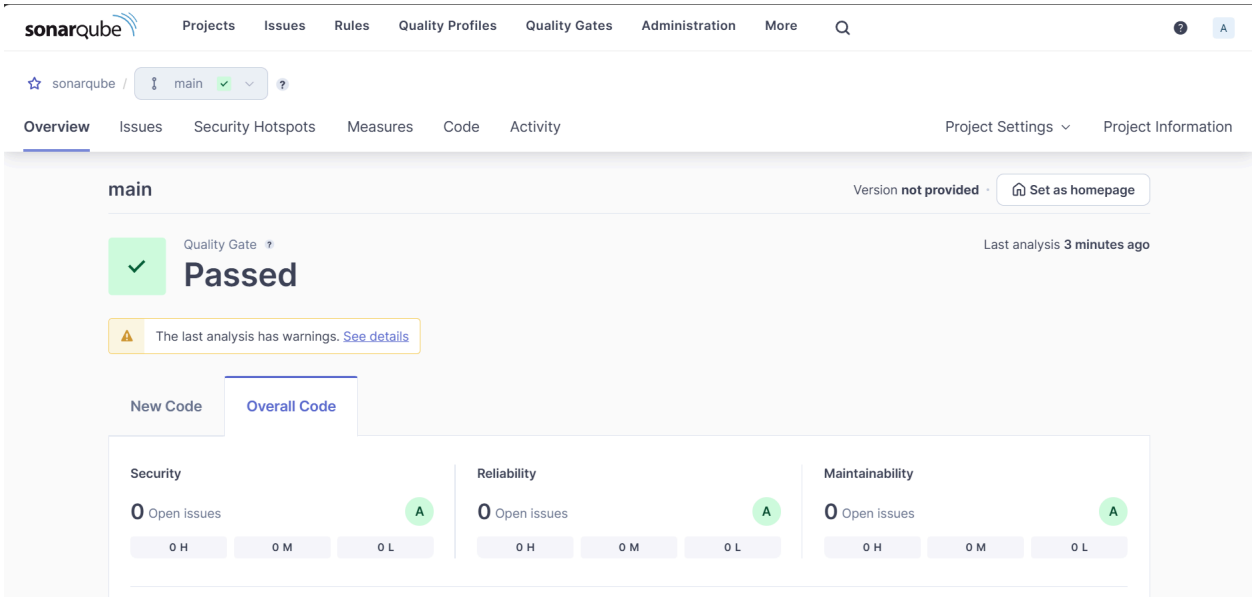


Step 16: Once the build is successful, check the console output.

```
Started by user Anish Kulkarni
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SonarqubeProject29
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarqubeProject29\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url http://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from http://github.com/shazforiot/MSBuild_firstproject.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.46.0.windows.1'
 > git.exe fetch --tags --force --progress -- http://github.com/shazforiot/MSBuild_firstproject.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[SonarqubeProject29] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\scanner29\bin\sonar-
scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.host.url=http://localhost:9000 -
Dsonar.login=admin -Dsonar.sources=. -Dsonar.password=ANISH2004 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\SonarqubeProject29
16:13:33.220 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
16:13:33.236 INFO  Scanner configuration file:
C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\scanner29\bin\..\conf\sonar-scanner.properties
16:13:33.246 INFO  Project root configuration file: NONE
```

```
16:14:12.549 INFO  Sensor C# [csharp] (done) | time=2ms
16:14:12.549 INFO  Sensor Analysis Warnings import [csharp]
16:14:12.549 INFO  Sensor Analysis Warnings import [csharp] (done) | time=0ms
16:14:12.549 INFO  Sensor C# File Caching Sensor [csharp]
16:14:12.549 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting
'sonar.projectBaseDir' property.
16:14:12.549 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=0ms
16:14:12.549 INFO  Sensor Zero Coverage Sensor
16:14:12.555 INFO  Sensor Zero Coverage Sensor (done) | time=6ms
16:14:12.555 INFO  SCM Publisher SCM provider for this project is: git
16:14:12.567 INFO  SCM Publisher 4 source files to be analyzed
16:14:13.180 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=613ms
16:14:13.184 INFO  CPD Executor Calculating CPD for 0 files
16:14:13.184 INFO  CPD Executor CPD calculation finished (done) | time=0ms
16:14:13.191 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
16:14:13.474 INFO  Analysis report generated in 117ms, dir size=201.0 kB
16:14:13.522 INFO  Analysis report compressed in 39ms, zip size=22.2 kB
16:14:13.756 INFO  Analysis report uploaded in 231ms
16:14:13.759 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
16:14:13.759 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis
report
16:14:13.759 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=fe09e0c2-e06a-411d-ae21-10cce1b3c081
16:14:13.787 INFO  Analysis total time: 25.432 s
16:14:13.789 INFO  SonarScanner Engine completed successfully
16:14:13.842 INFO  EXECUTION SUCCESS
16:14:13.842 INFO  Total time: 40.606s
Finished: SUCCESS
```

Step 17: Go back to SonarQube and check your project.



Conclusion: In this experiment, we learned how to integrate Jenkins SAST to SonarQube. We first used a docker image of SonarQube in order to avoid having to install it on our system. Next, SonarQube was configured and a SonarQube project was created. Then, required configurations were done on Jenkins and a Jenkins freestyle project was created which contained links to a code file from a Github repository and also our SonarQube project. When the Jenkins project was built, the SonarQube project displayed that there were no issues with the code in the Jenkins project.