Sairam Konar D15C 27 2024-25

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.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker -v
Docker version 27.0.3, build 7d4bcd8
```

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.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
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
```

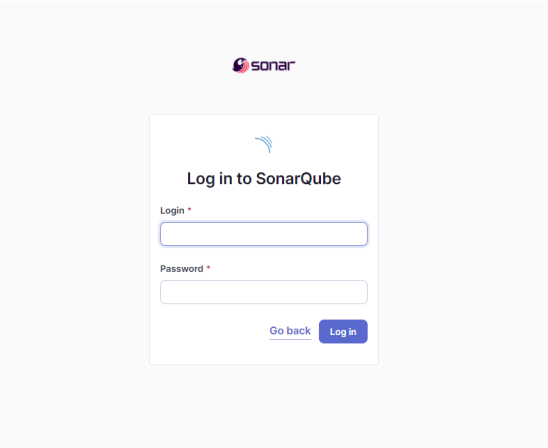
3) Keep jenkins installed on your system.

Experiment Steps:

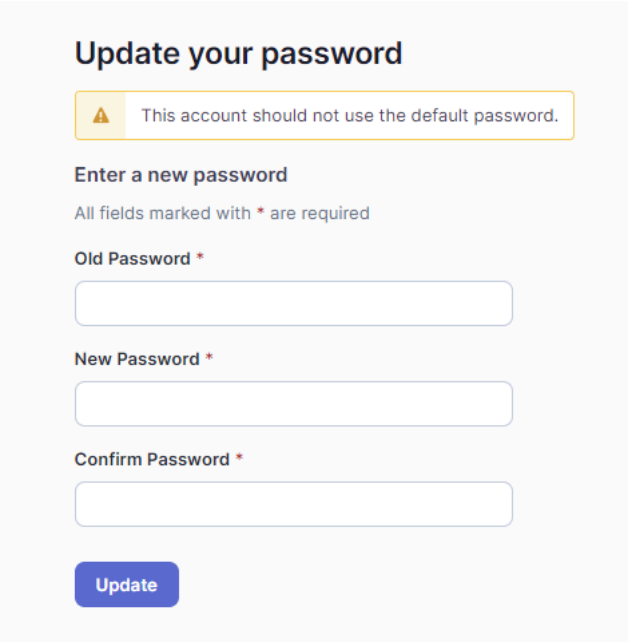
Step 1: Run SonarQube image
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
This command will run the SonarQube image that was just installed using docker.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=t
rue -p 9000:9000 sonarqube:latest
36ff8a656bd28857ba9a28bf2bb0174099ae3232a9fc9ba2766d46f0c14d08a6
```

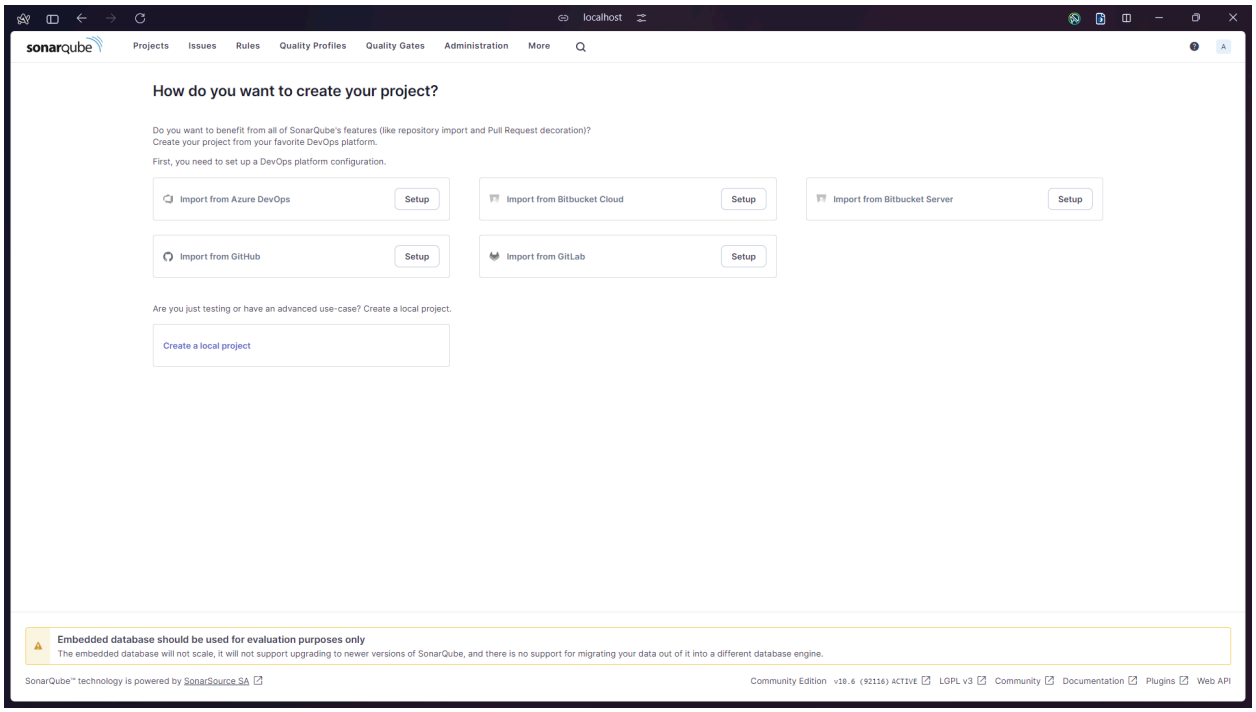
Step 2: Once the SonarQube image is started, you can go to http://localhost:9000 to find the SonarQube that has started



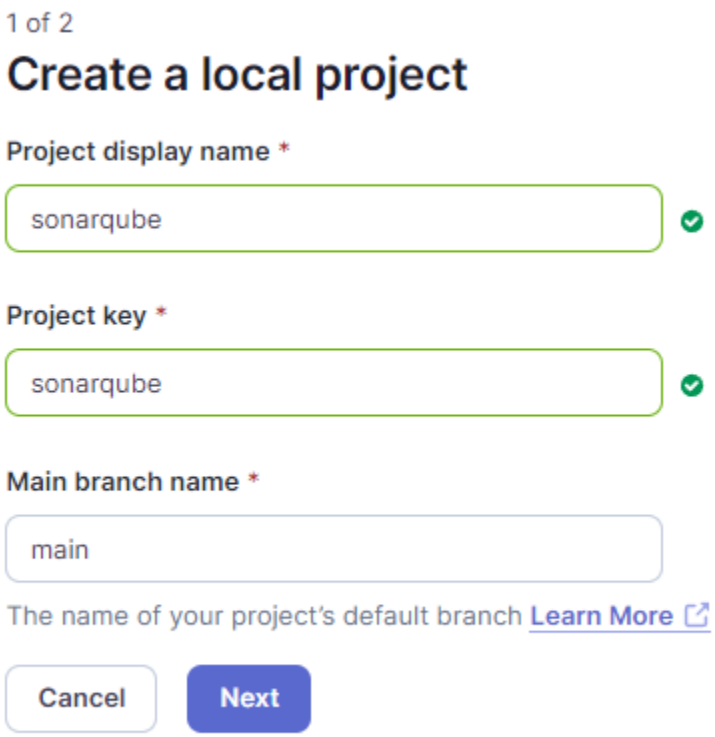
Step 3: On this interface, login with username = ‘admin’ and password = ‘admin’. Once logged in successfully, SonarQube will ask you to reset this password. Reset it and remember this password.



Step 4: After changing the password, you will be directed to this screen. Click on Create a Loal Project.

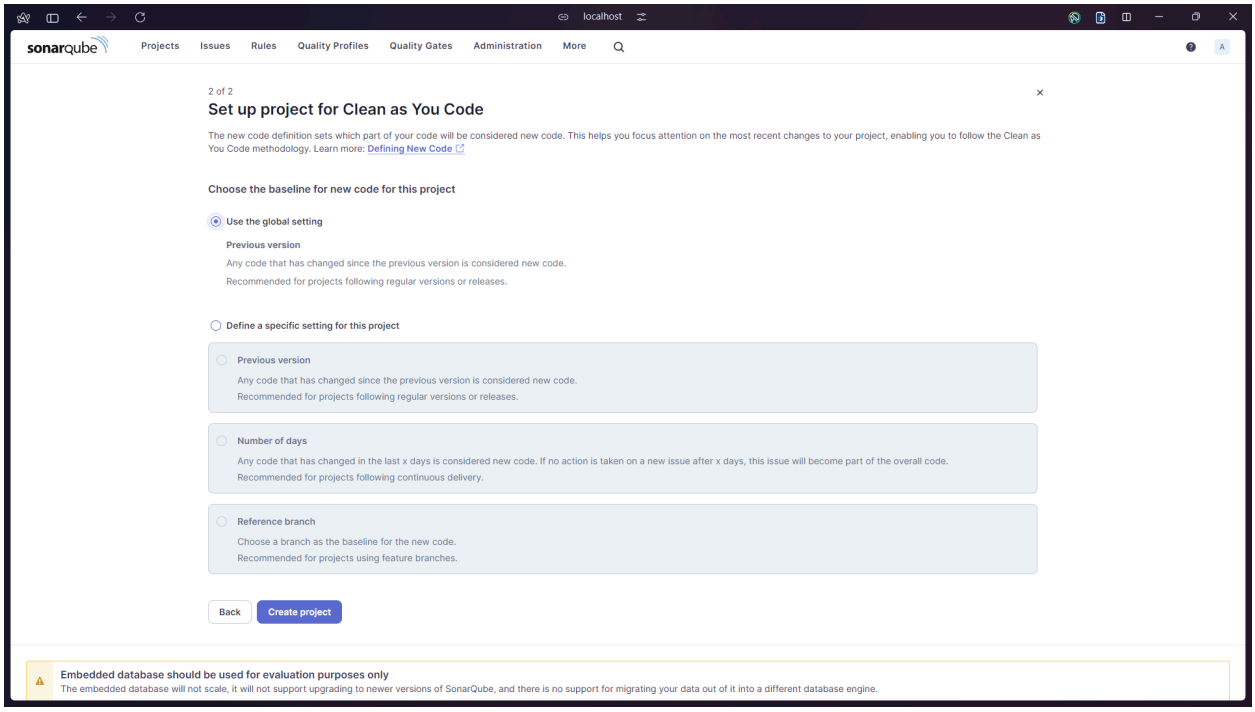


Give the project a display name and project key

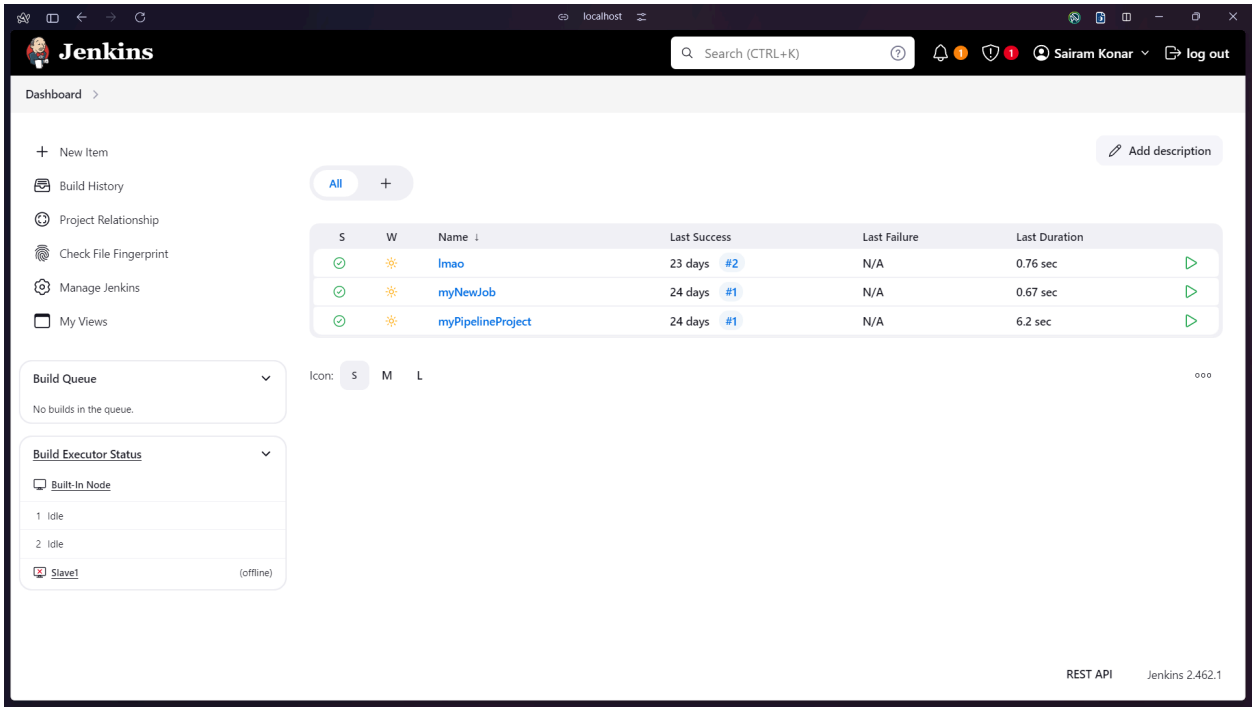


Sairam Konar D15C 27 2024-25

Set up the project as required and click on create.

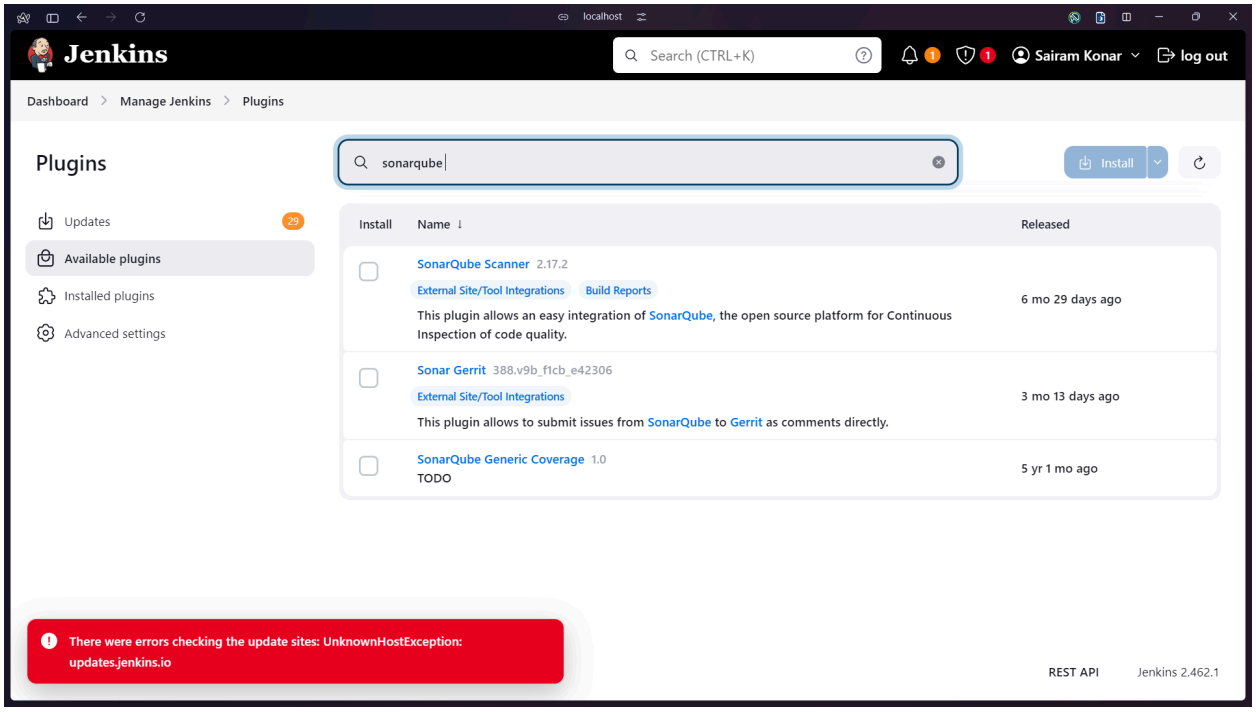


Step 5: Open Jenkins on whichever port it is installed. (http://loaclhost:<port_number>).

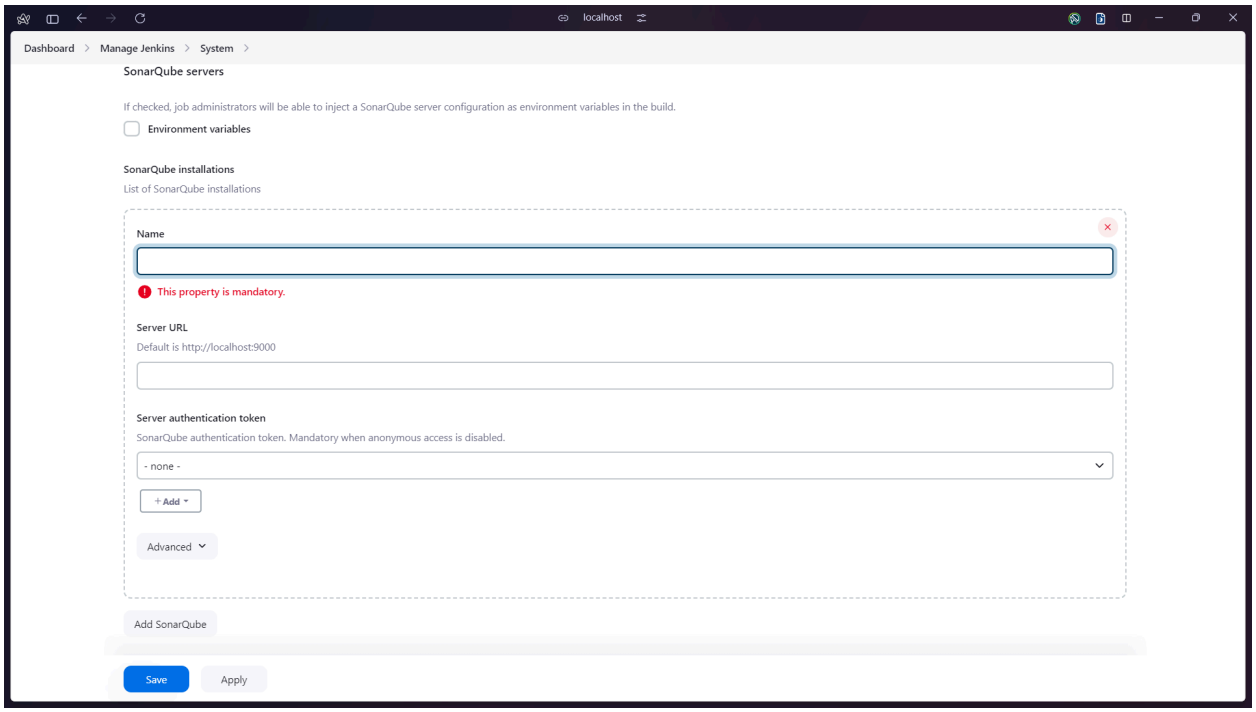


Sairam Konar D15C 27 2024-25

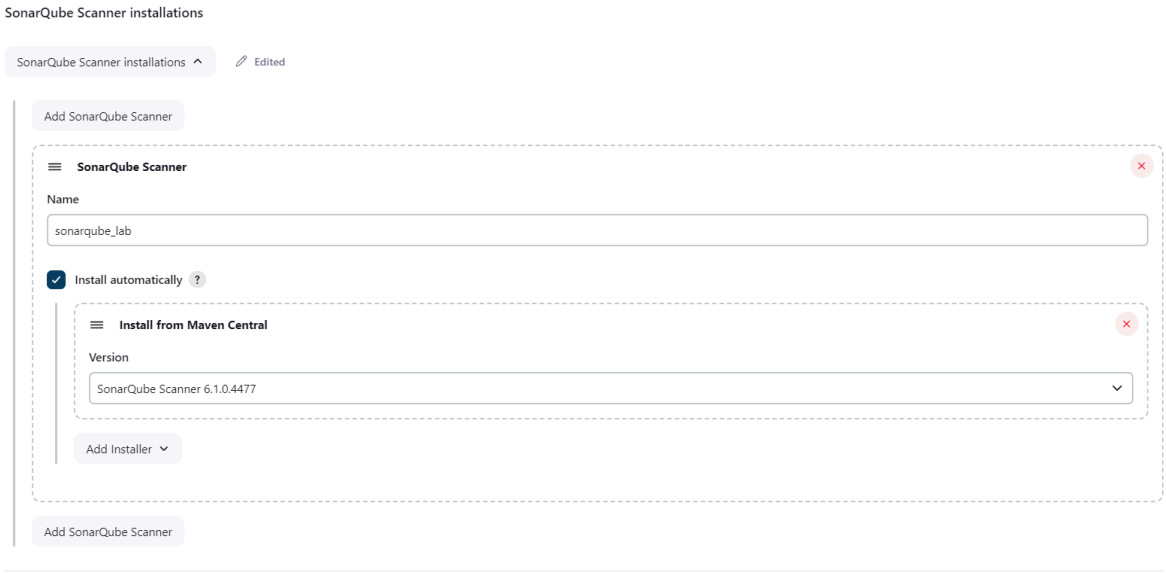
Step 6: Go to manage jenkins → Search for Sonarqube Scanner for Jenkins and install it.



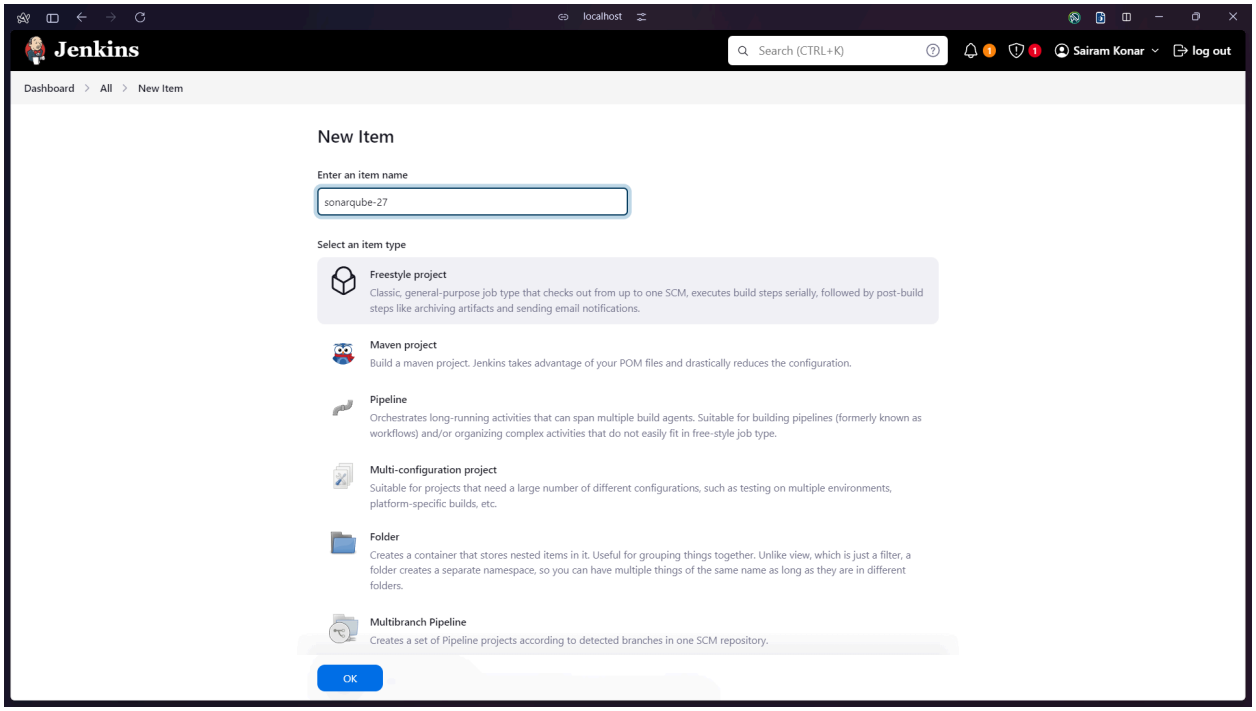
Step 7: Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.



Step 8: Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose install automatically.

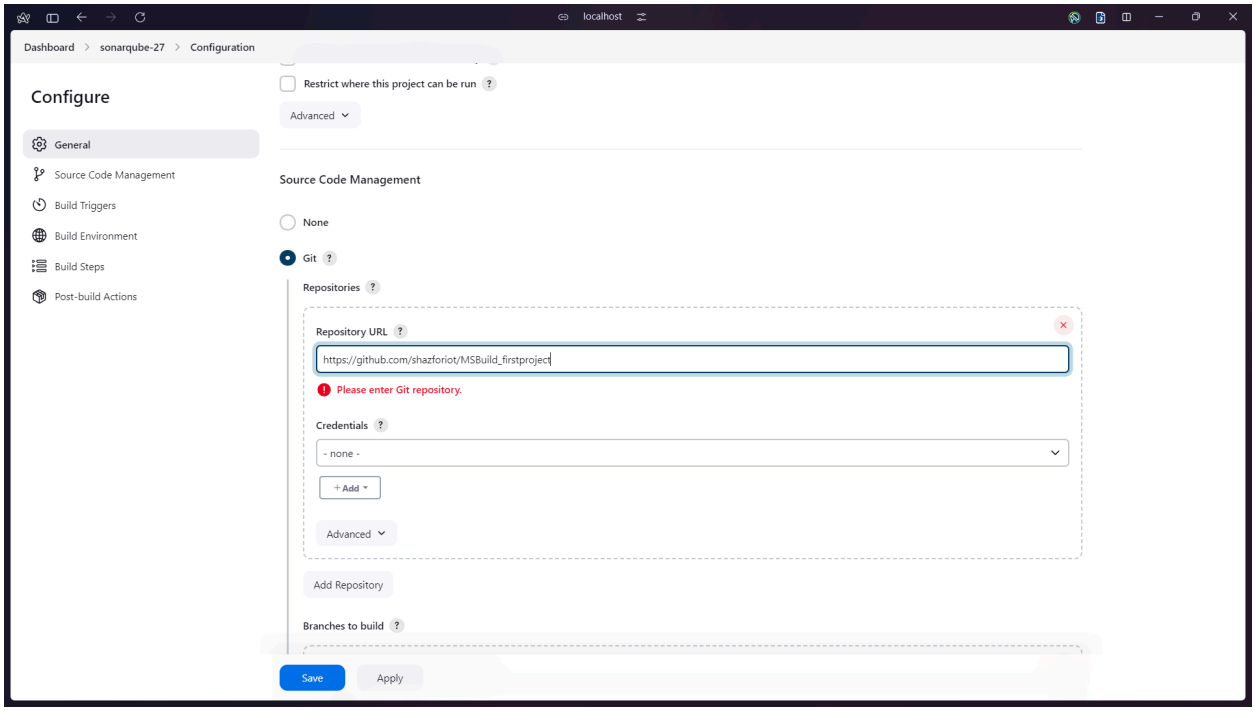


Step 9: After configuration, create a New Item → choose a freestyle project.

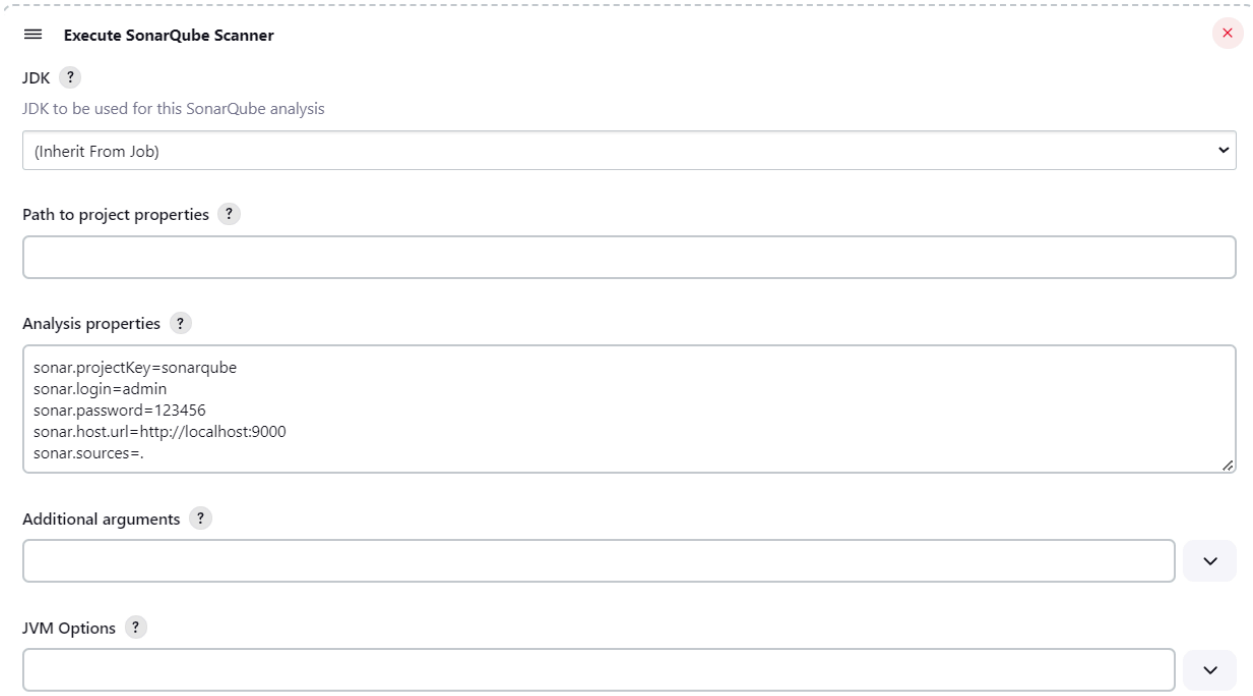


Sairam Konar D15C 27 2024-25

Step 10: Use this github repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject
It is a sample hello-world project with no vulnerabilities.

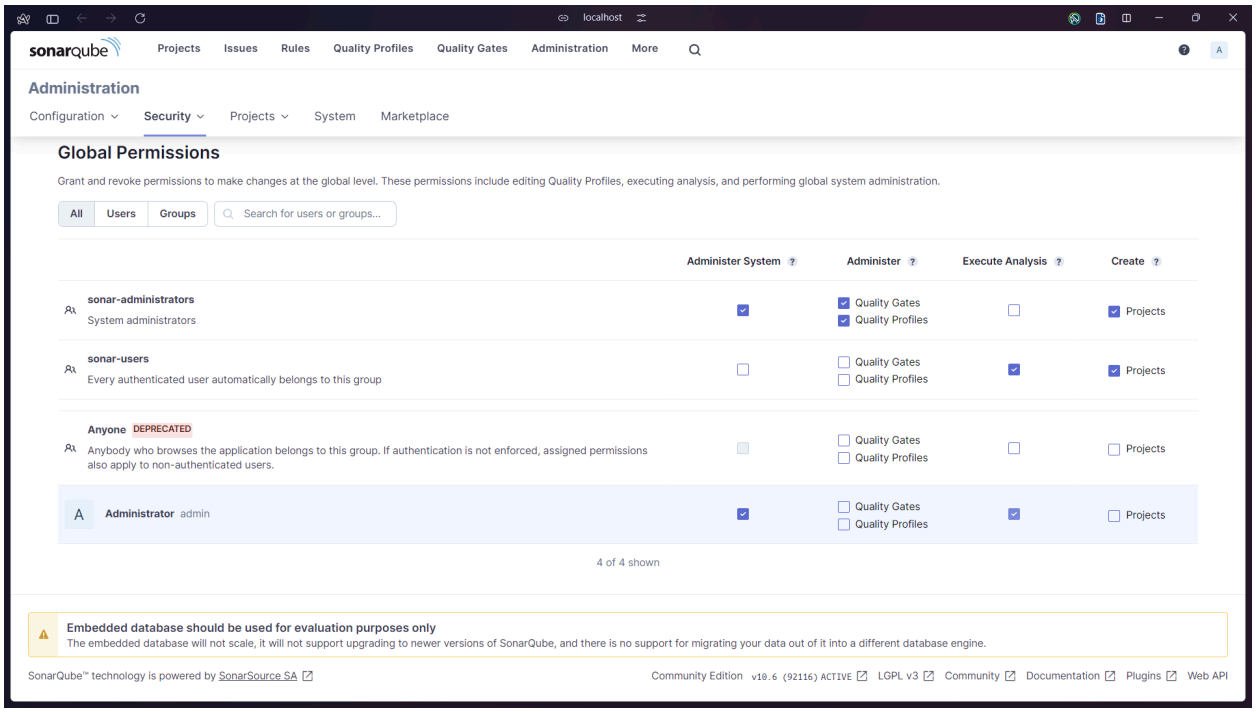


Step 11: Under Build Steps, enter Sonarqube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

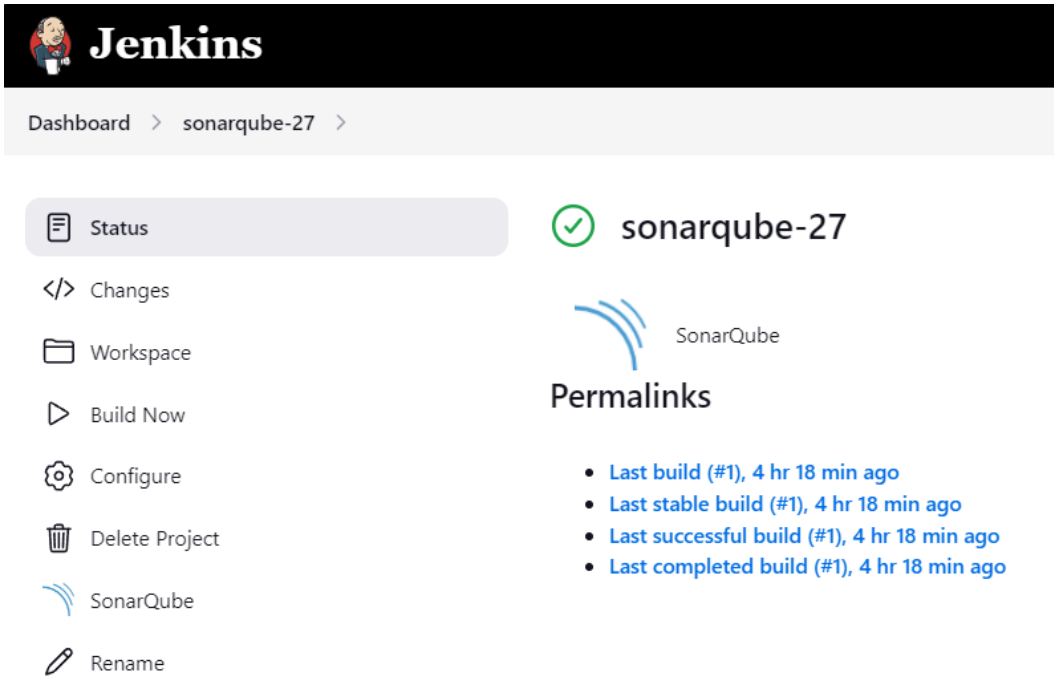


Sairam Konar D15C 27 2024-25

Step 12: Now, you need to grant the locl user (here admin user) permissions to Execute the Analysis stage on SoanrQube. For this, go to http://loaclhost:<port_number>/admin/permissions and check the 'Execute Analysis' checkbox under Administrator.



Step 13: Go back to jenkins. Go to the job you had just built and click on Build Now.

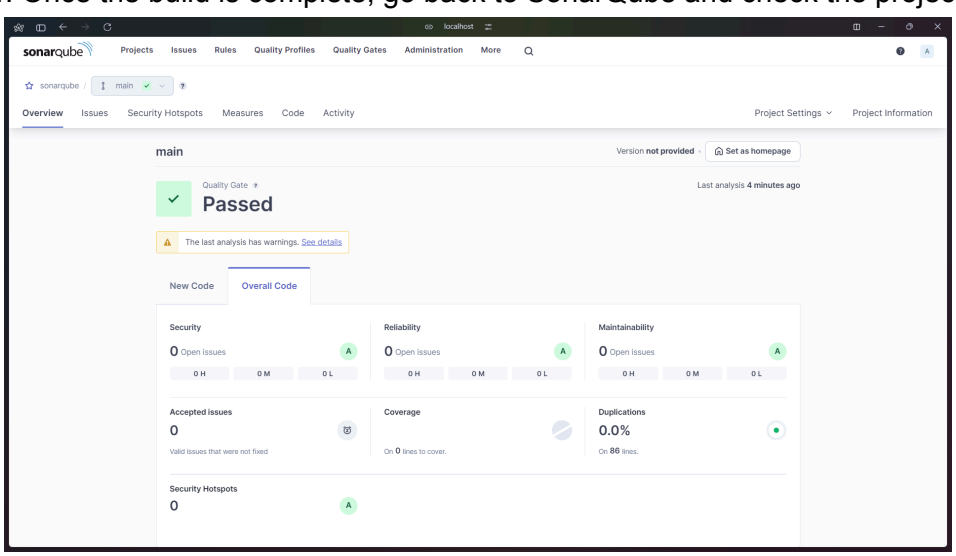


Check the console Output

```
Started by user Sairam Konar
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-27
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-27\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.2.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject +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"
First time build. Skipping changelog.
[sonarqube-27] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube_lab\bin\sonar-scanner.bat -Dsonar.projectKey=sonarqube -
Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -Dsonar.password=123456 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-27
09:24:56.227 INFO  Scanner configuration file: C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube_lab\bin\..\conf\sonar-
scanner.properties
09:24:56.231 INFO  Project root configuration file: NONE
```

```
09:25:27.830 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the SonarScanner for .NET
5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
09:25:27.831 INFO  Sensor C# [csharp] (done) | time=1ms
09:25:27.831 INFO  Sensor Analysis Warnings import [csharp]
09:25:27.833 INFO  Sensor Analysis Warnings import [csharp] (done) | time=2ms
09:25:27.834 INFO  Sensor C# File Caching Sensor [csharp]
09:25:27.835 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
09:25:27.835 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=1ms
09:25:27.835 INFO  Sensor Zero Coverage Sensor
09:25:27.852 INFO  Sensor Zero Coverage Sensor (done) | time=18ms
09:25:27.856 INFO  SCM Publisher SCM provider for this project is: git
09:25:27.858 INFO  SCM Publisher 4 source files to be analyzed
09:25:28.360 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=501ms
09:25:28.363 INFO  CPD Executor Calculating CPD for 0 files
09:25:28.363 INFO  CPD Executor CPD calculation finished (done) | time=0ms
09:25:28.369 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
09:25:28.697 INFO  Analysis report generated in 119ms, dir size=201.0 kB
09:25:28.754 INFO  Analysis report compressed in 43ms, zip size=22.4 kB
09:25:28.977 INFO  Analysis report uploaded in 221ms
09:25:28.981 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
09:25:28.981 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
09:25:28.981 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=288a1357-213b-46ce-aeab-eb463fb79292
09:25:28.996 INFO  Analysis total time: 22.822 s
09:25:28.997 INFO  SonarScanner Engine completed successfully
09:25:29.032 INFO  EXECUTION SUCCESS
09:25:29.034 INFO  Total time: 32.809s
Finished: SUCCESS
```

Step 14: Once the build is complete, go back to SonarQube and check the project linked.



Conclusion:

In this experiment, we have learned how to perform Jenkins SAST using SonarQube. For this, we used a docker image of SonarQube so as to not install it locally on our system. After installing the required configurations on Jenkins, using a coe from a gihub repository, we analyze its code using SonarQube. Once we build the project, we can see that SonarQube project displays that the code has no errors.