Name:Shubham Jha Div:D15C Roll No.19

Adv DevOps Practical 7

Aim:

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

Integrating Jenkins with SonarQube:

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

Steps

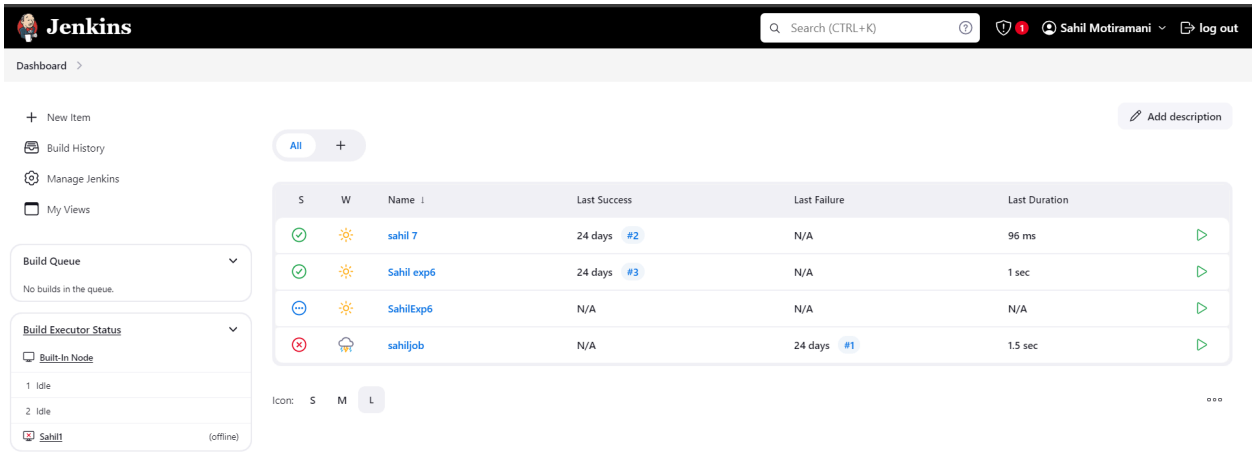
to integrate Jenkins with SonarQube

Prerequisites: Make sure you have docker and jenkins installed.
Run **docker -v** to check the docker installation.

Run

1. Open up Jenkins Dashboard on localhost, port 8090 or whichever port it is at for you.

Name:Shubham Jha Div:D15C Roll No.19



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

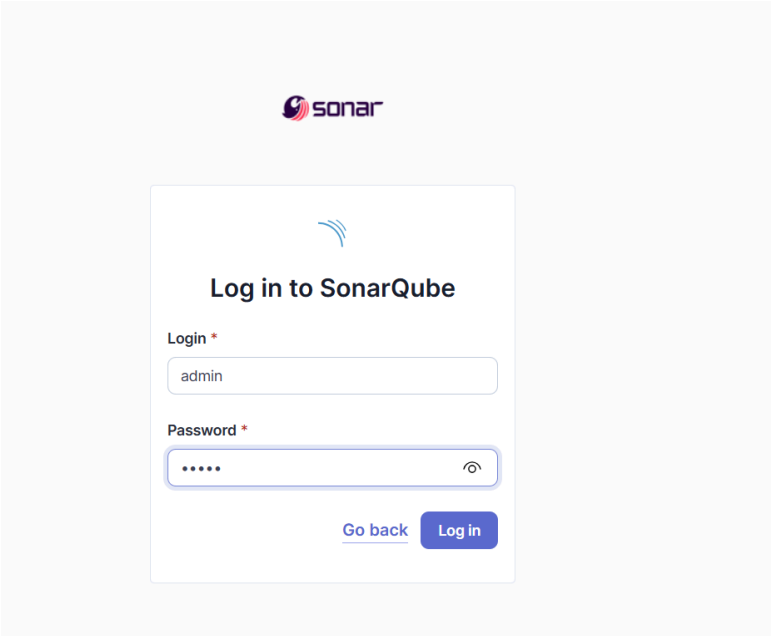
----------Warning: run below command only once

```
C:\Users\Lenovo>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
47f6db8dbf2ed99dbe304bc0ebdf47b9d4144c4e4add42055ba44ce231058272
```

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

localhost port 9000.

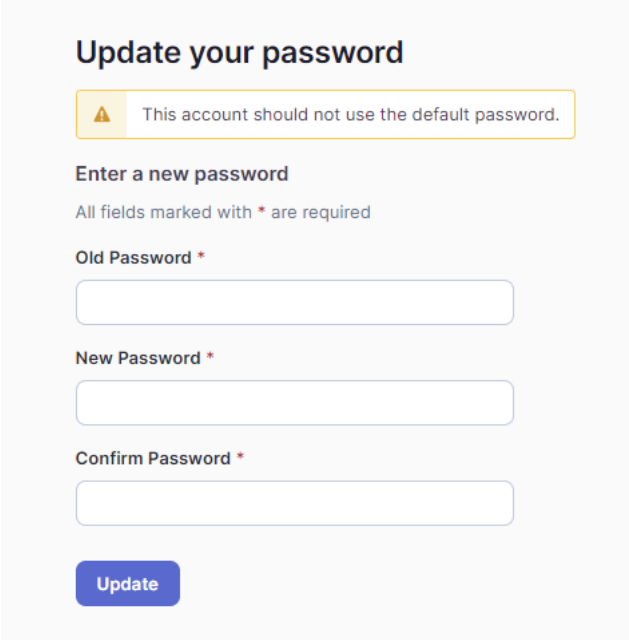
Name:Shubham Jha Div:D15C Roll No.19

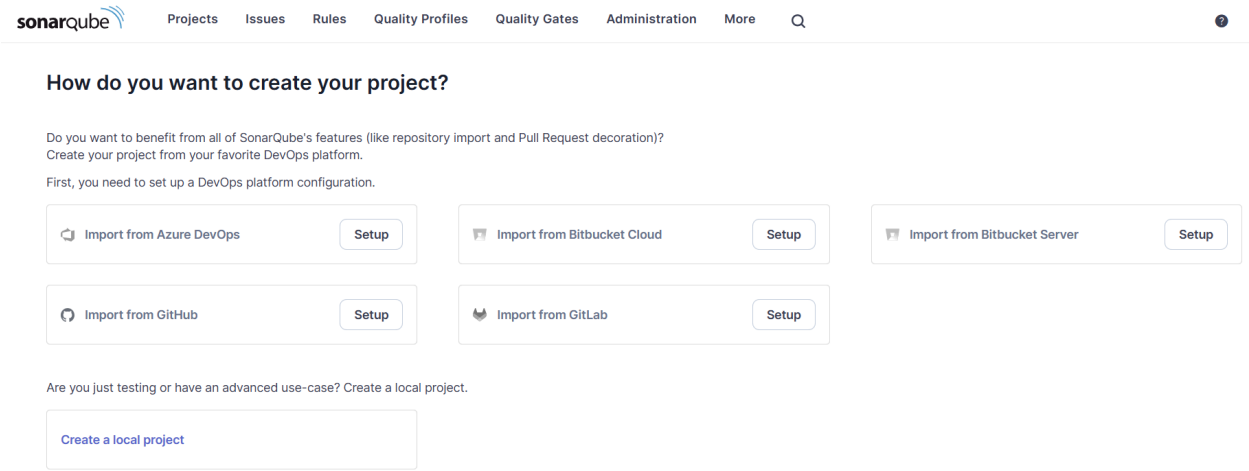


4. Login to SonarQube using username admin and password admin.

Name:Shubham Jha Div:D15C Roll No.19

(do change the password as you cannot use the default one)





5. Create a manual project in SonarQube with the name sonarqube
(Click on create local project)

Name:Shubham Jha Div:D15C Roll No.19

1 of 2

Create a local project

Project display name *

exp7

Project key *

exp7

Main branch name *

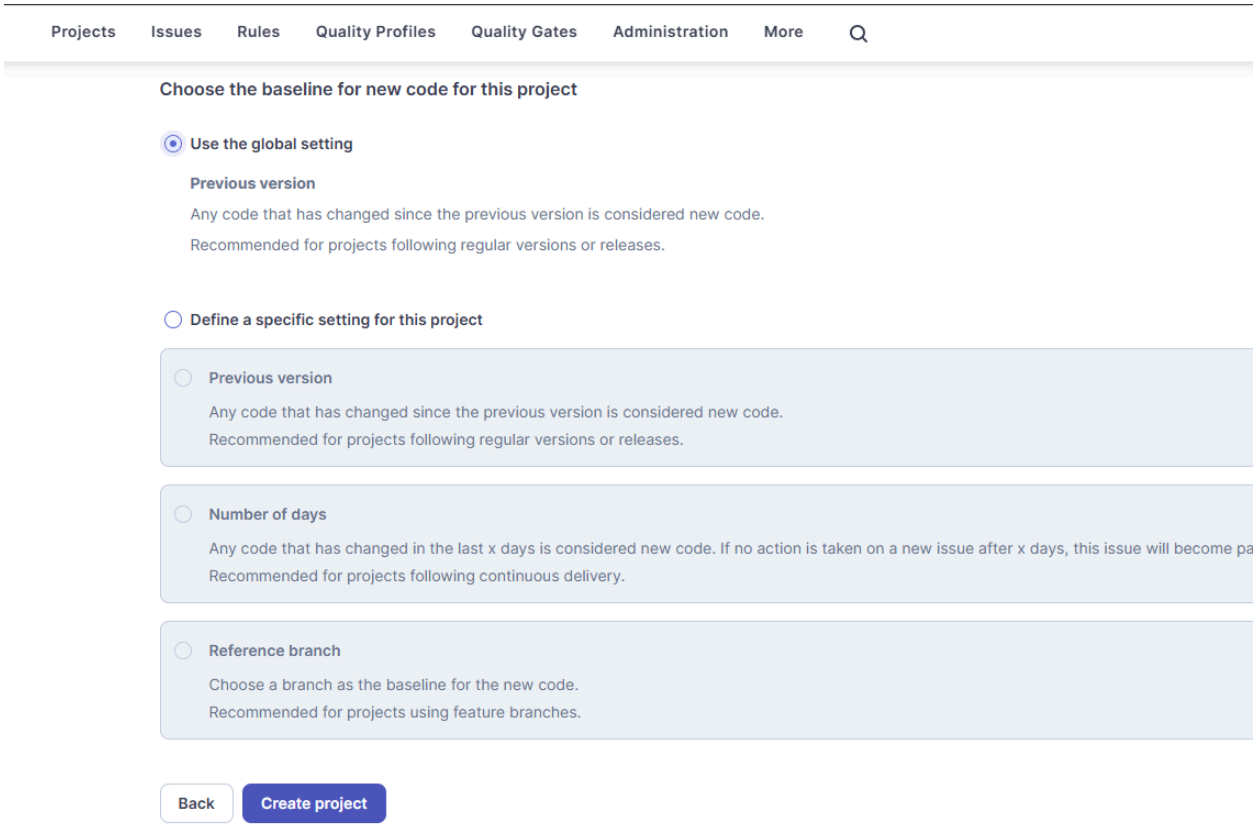
main

The name of your project's default branch Learn More

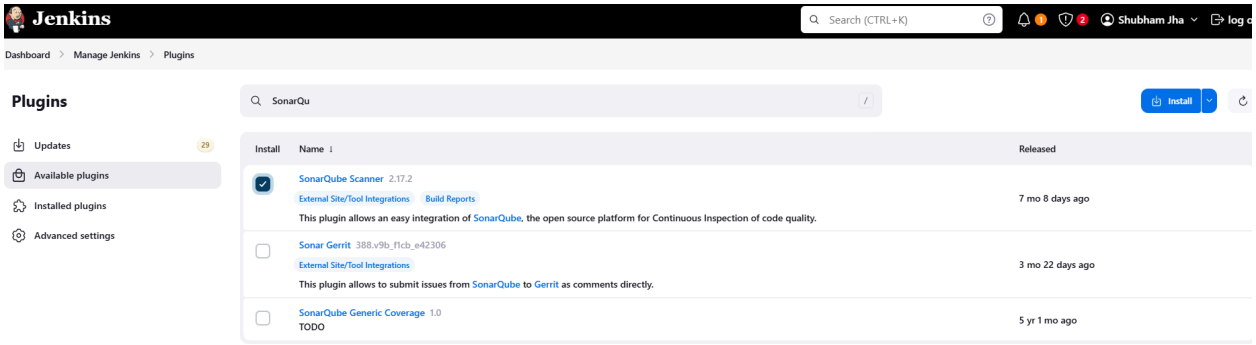
Cancel Next

Name:Shubham Jha Div:D15C Roll No.19

Setup the project



And come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.

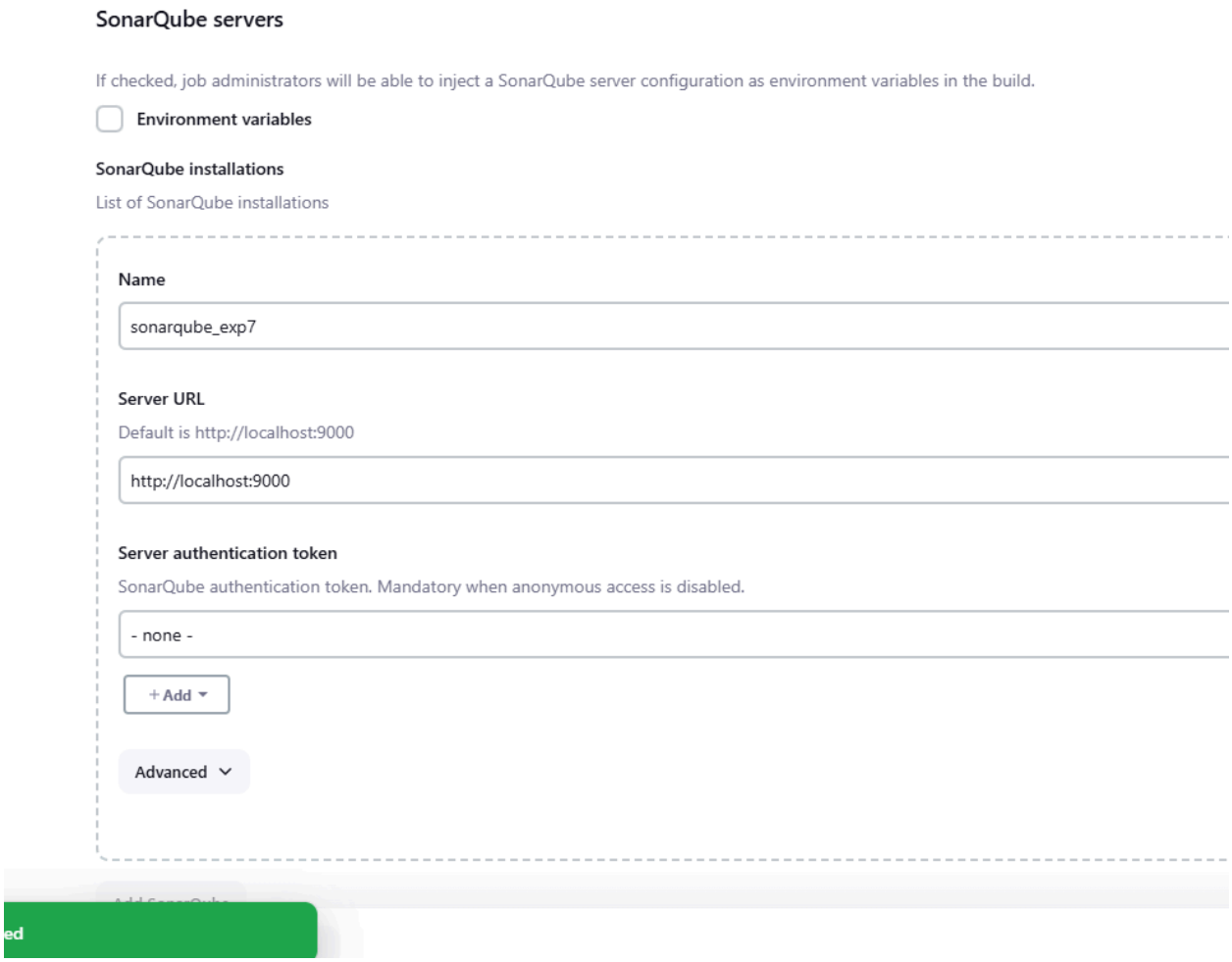


6. Under Jenkins ‘Manage Jenkins’ then go to ‘system’, scroll and look for **SonarQube Servers** and click on **add SonarQube** and then enter the details.

Enter the Server Authentication token if needed.(I didn’t do it)

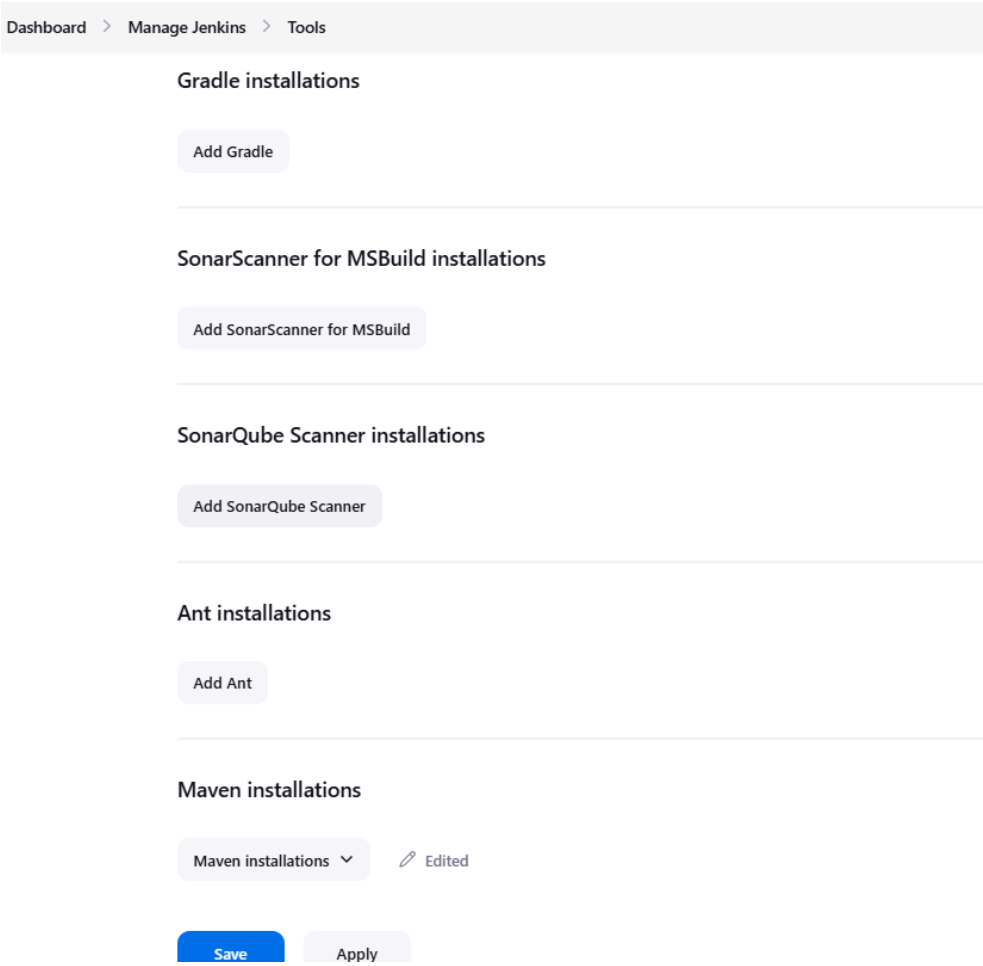
In SonarQube installations: Under **Name** add <project name of sonarqube> for me its sonarqube_exp7

In **Server URL** Default is http://localhost:9000



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

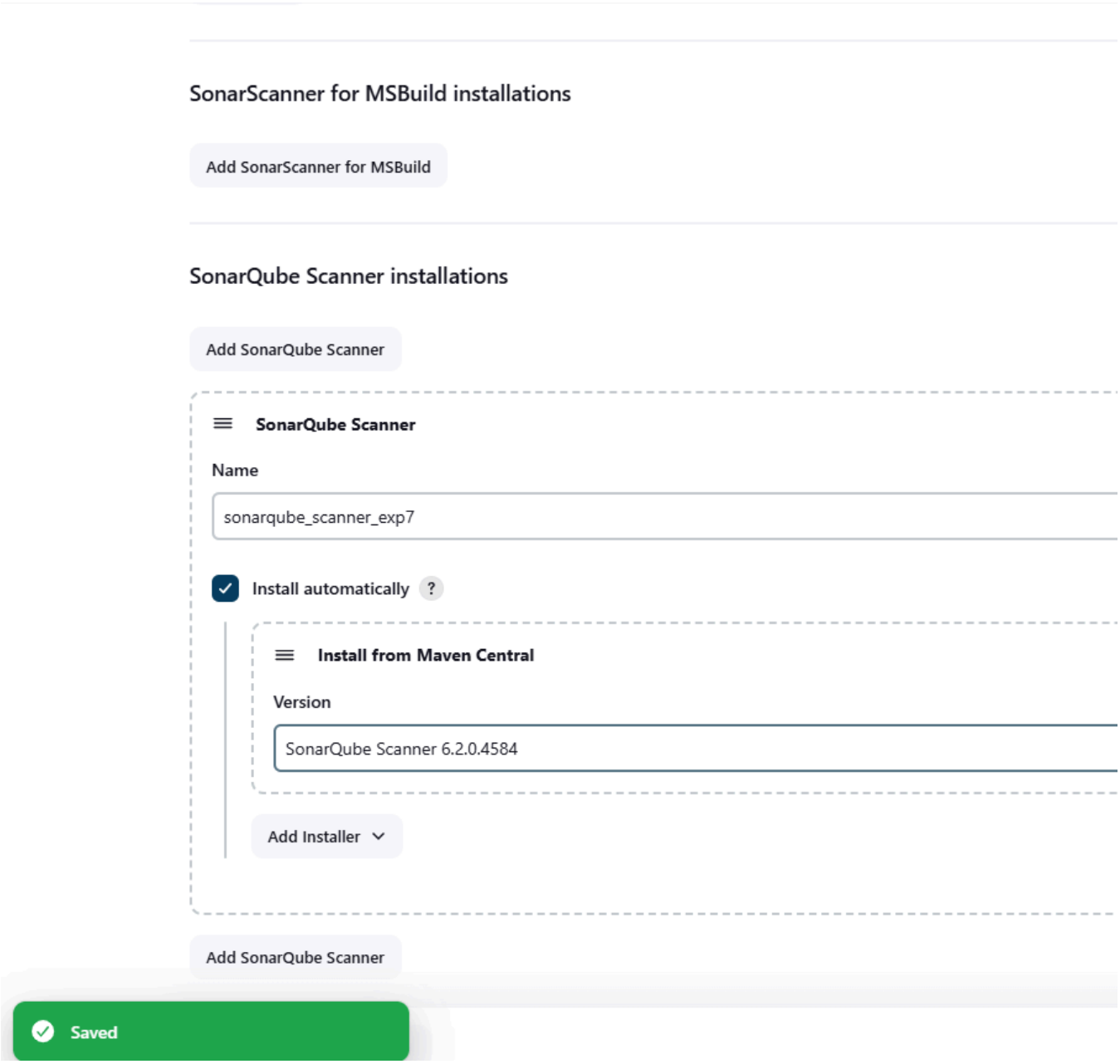
Dashboard > Manage Jenkins > Tools



Click on **Add SonarQube Scanner** .

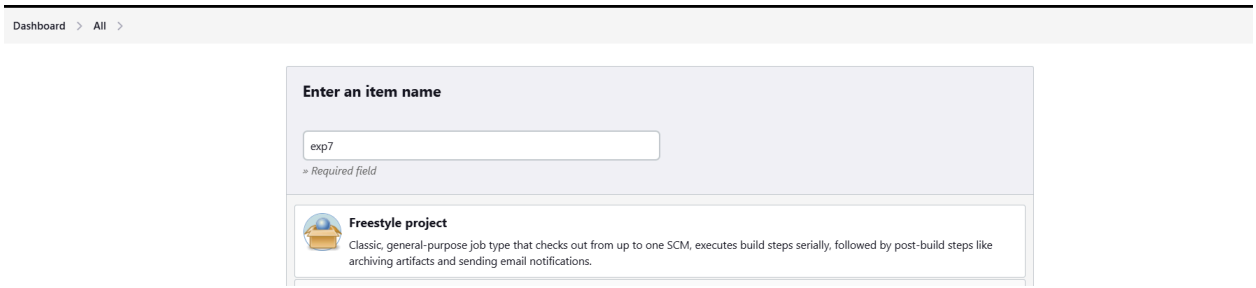
Name:Shubham Jha Div:D15C Roll No.19

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

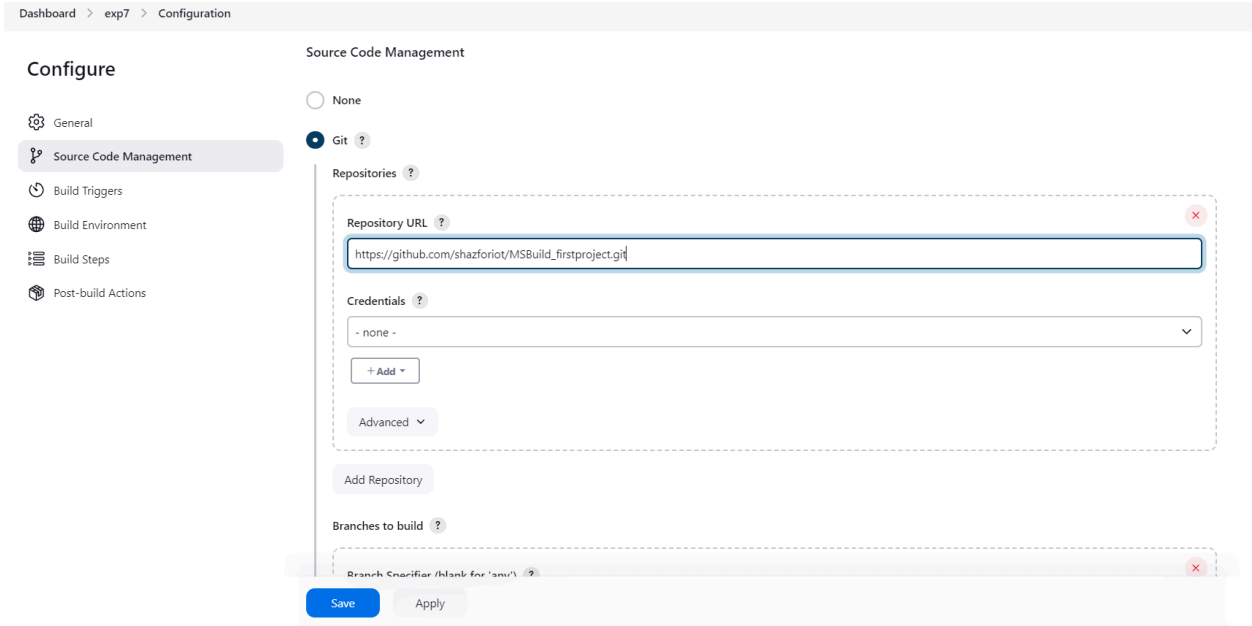


Name:Shubham Jha Div:D15C Roll No.19

8. After the configuration, create a New Item in Jenkins, choose a freestyle project.

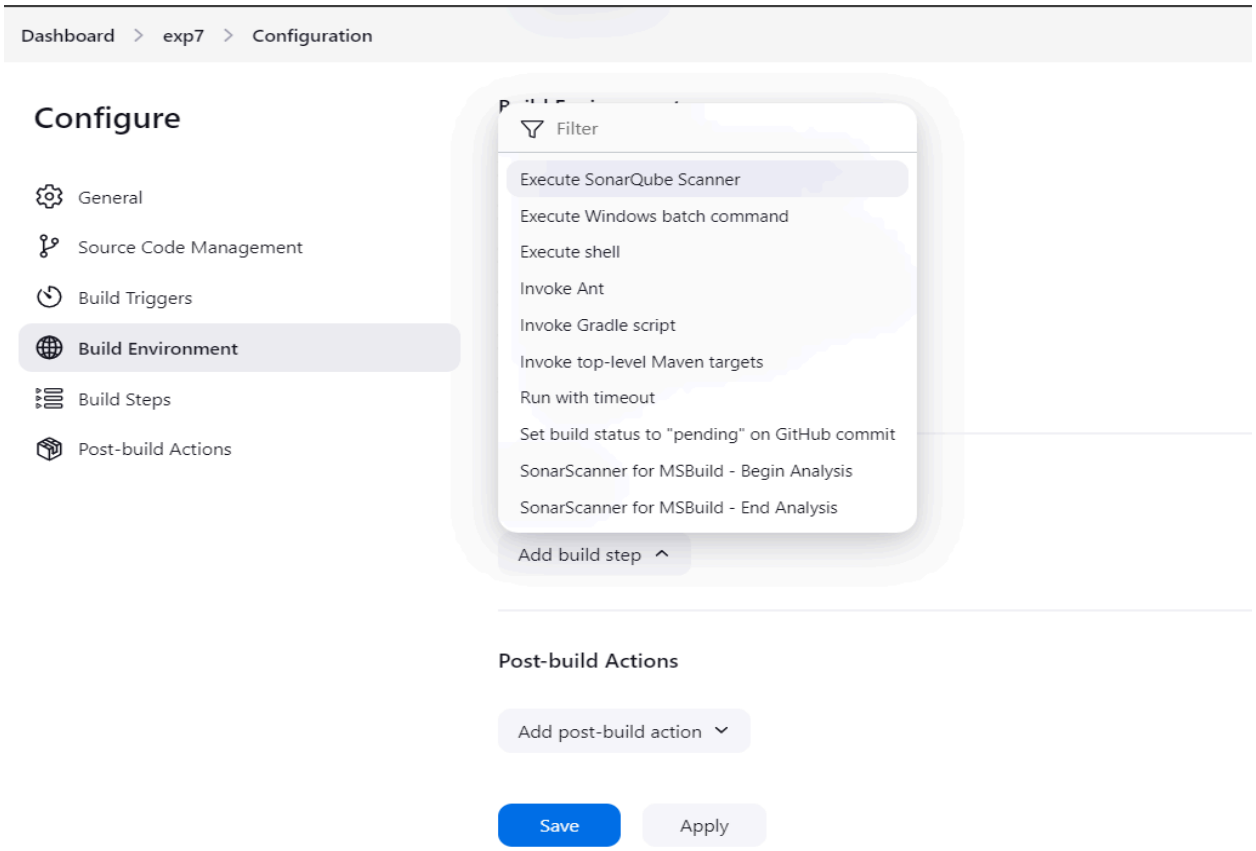


9. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

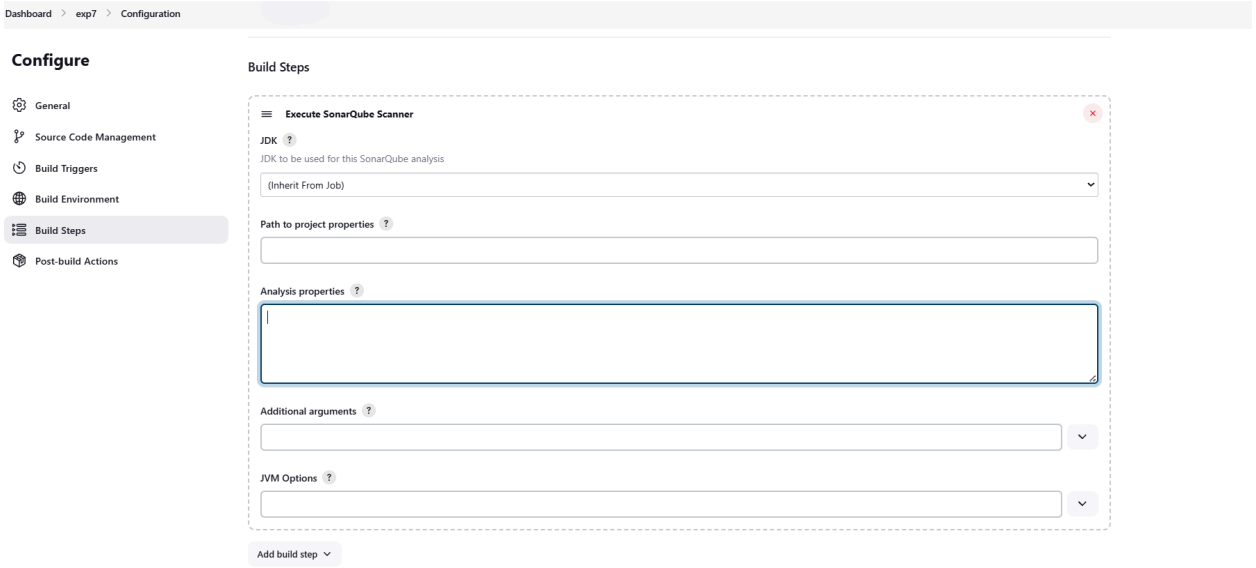


10. Under **Select project → Configuration → Build steps → Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Name:Shubham Jha Div:D15C Roll No.19

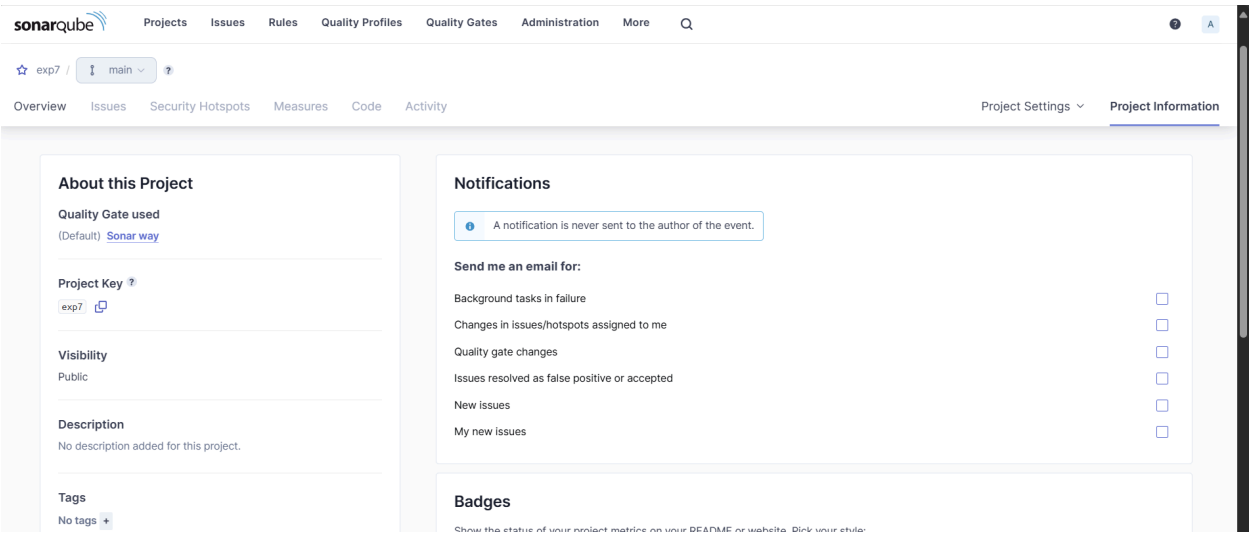


You will see something like this:



Open sonarQube again and go to Project Information appearing in the right side. Click on it and you can copy the project key from About the Project Section.

Name:Shubham Jha Div:D15C Roll No.19



Use this key in place of <projectKey> in the following code

sonar.projectKey =<projectKey>

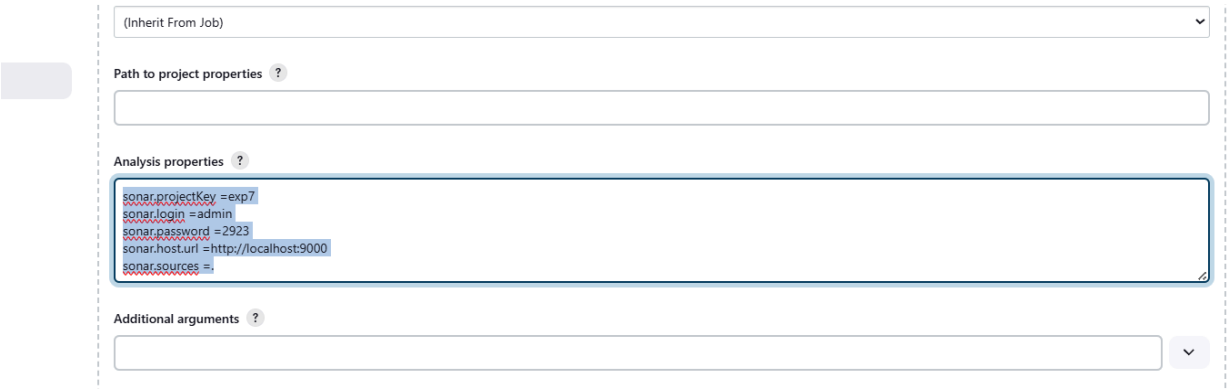
sonar.login =admin

sonar.password =<yourpassword for sonar qube>

sonar.host.url =http://localhost:9000

sonar.sources =.

I personally preferred not keeping any spaces after '=' .

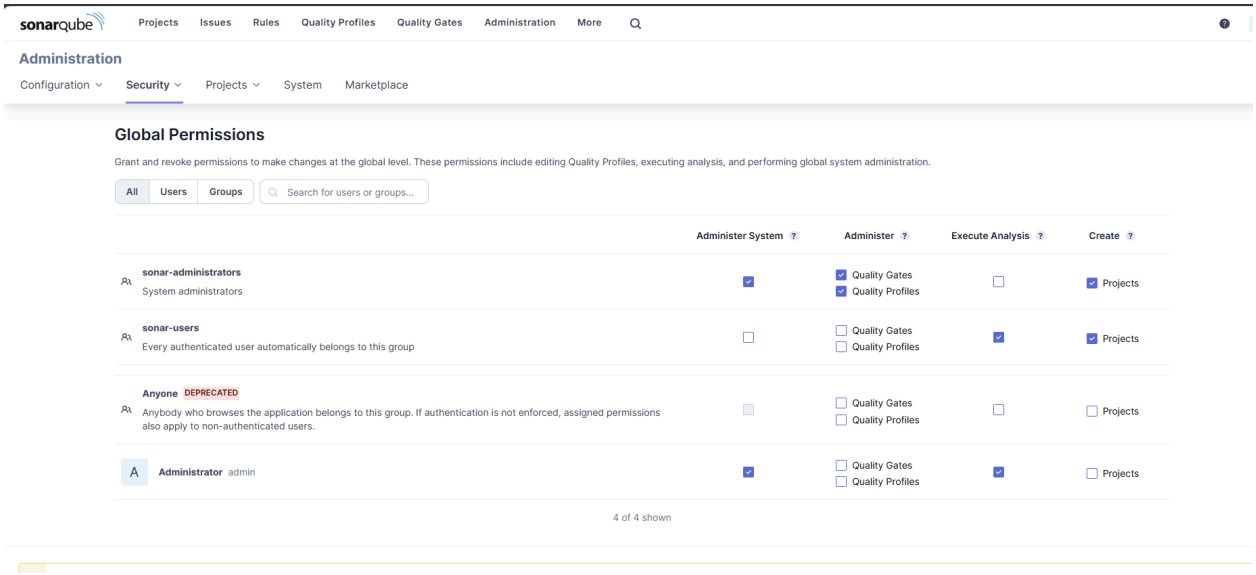


Apply and save.

Name:Shubham Jha Div:D15C Roll No.19

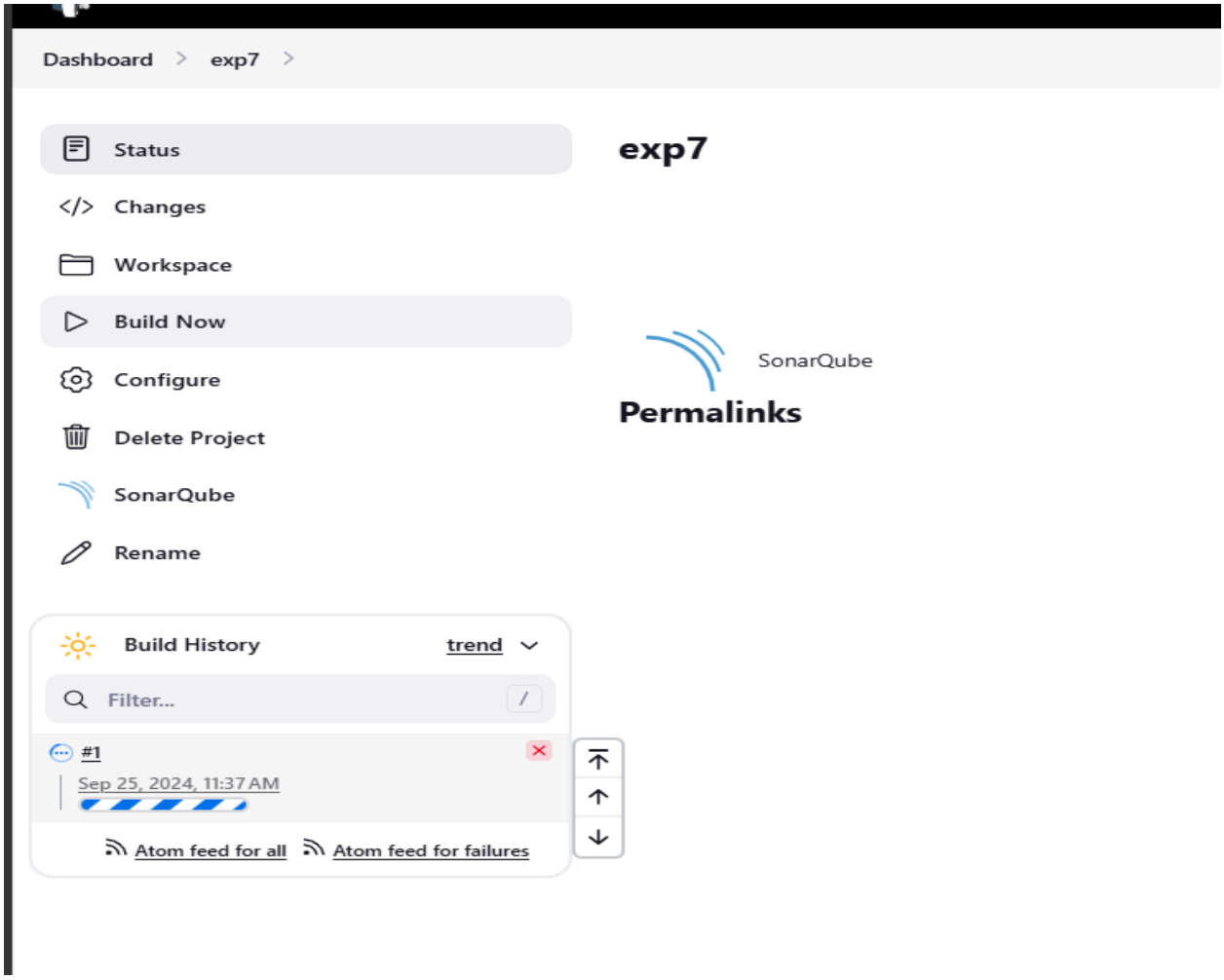
11. Go to sonarQube and go to administration → Security (dropdown) → Global Permissions.

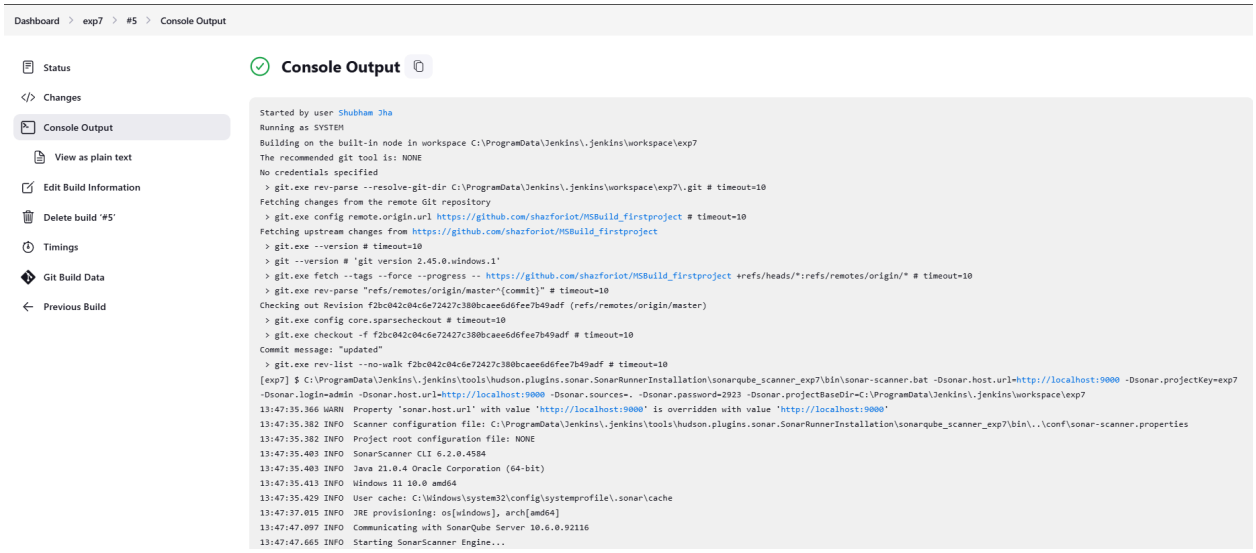
See the administrator below and check the boxes which i checked.



Name:Shubham Jha Div:D15C Roll No.19

12. Go to jenkins and click build:





Name:Shubham Jha Div:D15C Roll No.19

Conclusion:

In this project, I successfully integrated Jenkins with SonarQube to establish a robust automated static application security testing (SAST) pipeline. The setup involved deploying SonarQube using Docker, ensuring smooth container orchestration and efficient resource management. A key component was configuring Jenkins with the appropriate SonarQube plugins, authentication mechanisms, and linking it to a GitHub repository for continuous integration.

One of the challenges I faced was configuring Docker on the Jenkins environment, which required resolving networking issues between the Docker containers and ensuring that the SonarQube server was reachable from Jenkins. Additionally, setting up secure authentication between Jenkins and SonarQube involved troubleshooting token-based authentication and resolving environment path issues, particularly with the **`JAVA_HOME`** setup for the SonarQube scanner.

After overcoming these obstacles, I integrated the SonarQube scanner as a build step, allowing for continuous code analysis. This setup provided automated detection of code vulnerabilities, code smells, and quality issues. It helped ensure that any new commits triggered immediate analysis, generating detailed reports and promoting continuous improvement in code quality.