

## **Adv. Devops Experiment no. 7**

Name: Rohan Lalchandani

Class: D15A Roll no.: 25

**Aim:** To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

### **Theory:**

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled.

It's also known as white box testing.

### **Why SAST is Important in DevOps ?**

**Early Detection of Vulnerabilities:** SAST helps identify vulnerabilities early in the development phase, allowing developers to fix issues before the code is deployed to production.

**Shifts Security Left:** Incorporating SAST in DevOps practices supports the "Shift Left" approach in security, where testing starts earlier in the pipeline rather than waiting until the final stages.

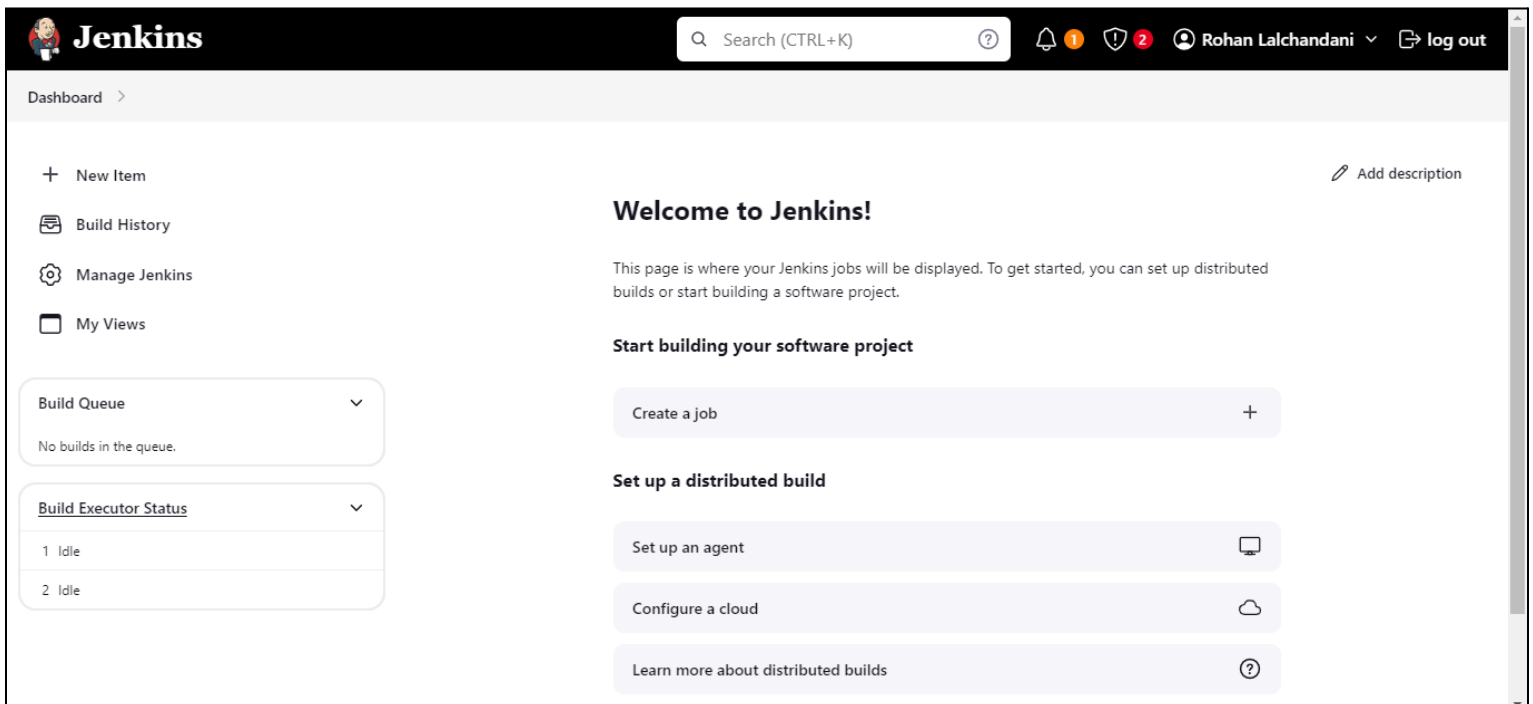
**Automation-Friendly:** SAST tools can be integrated with CI/CD pipelines, ensuring automated security checks on every code commit or pull request.

## SAST in DevOps Workflow Example

1. **Code Commit:** Developers push code changes to a version control system like Git.
2. **Automated Build:** The CI system (e.g., Jenkins) automatically triggers a build and kicks off the SAST scan as part of the pipeline.
3. **SAST Scan:** The SAST tool scans the code and reports vulnerabilities if present.  
Some popular SAST tools for DevOps include:
4. **Build Failures or Warnings:** If critical vulnerabilities are found, the build fails or issues warnings, depending on the security policies in place.
5. **Developer Feedback:** Developers receive feedback, either through IDE plugins, the CI/CD dashboard, or via notifications, so they can address the issues quickly.
6. **Security Approval:** Once all critical vulnerabilities are resolved, the build proceeds, and the application can move to the next stage of deployment.

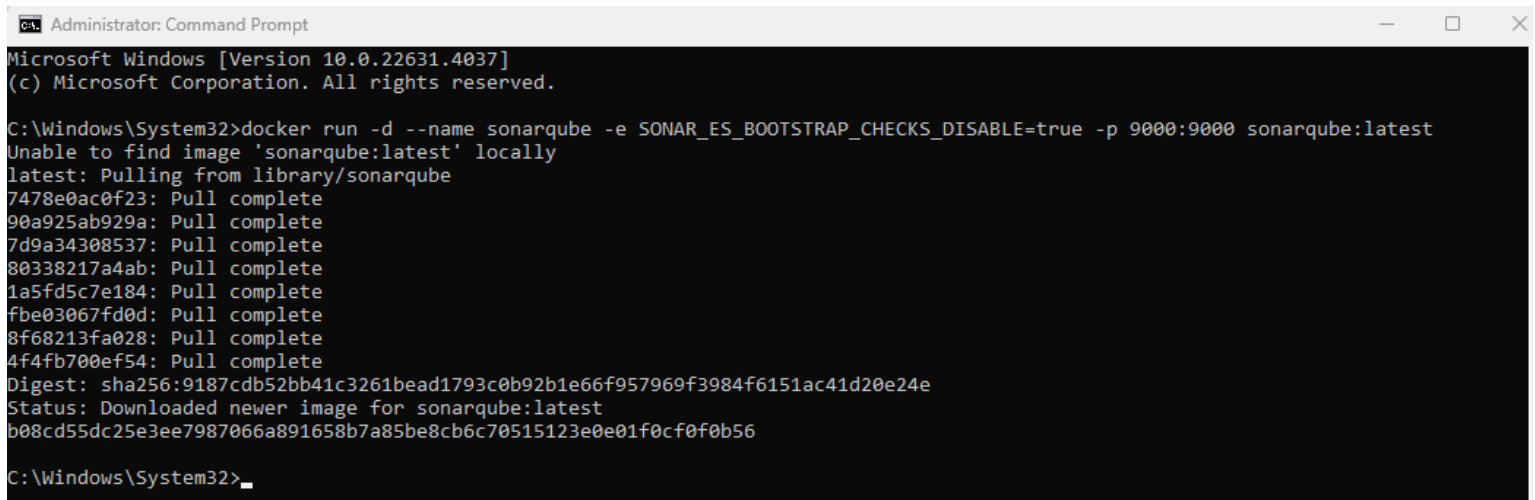
## Implementation:

Step 1: Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



Step 2: Run Sonarqube's image using the following command.

```
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

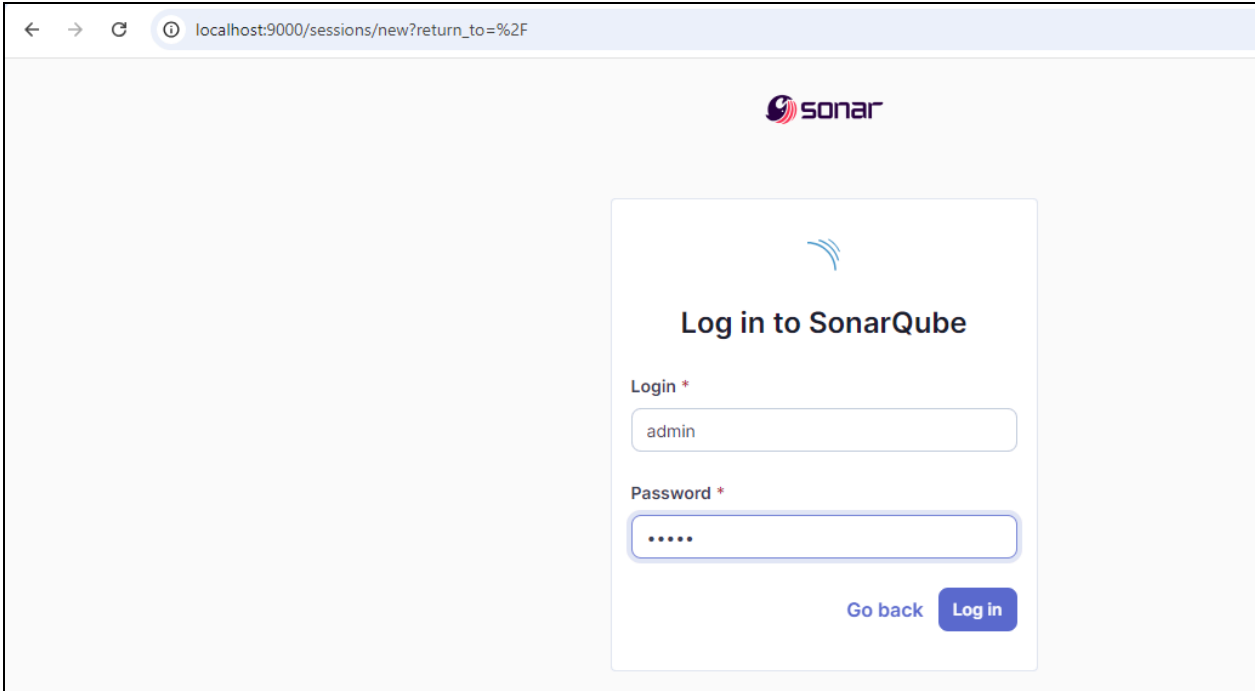
A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the execution of a Docker command to pull and run the SonarQube latest image. The output indicates that the image was not found locally and was pulled from the library. The pull process shows several layers being pulled and completed. The final status is "Downloaded newer image for sonarqube:latest".

```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

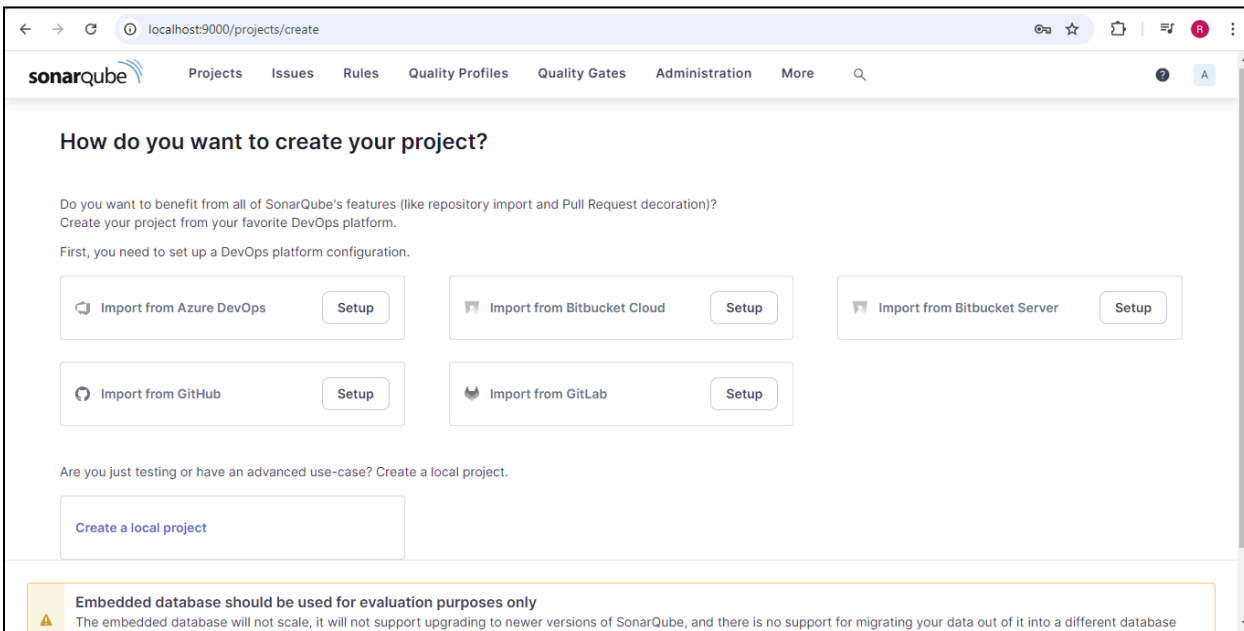
C:\Windows\System32>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
fbc03067fd0d: Pull complete
8f68213fa028: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:9187cdb52bb41c3261bead1793c0b92b1e66f957969f3984f6151ac41d20e24e
Status: Downloaded newer image for sonarqube:latest
b08cd55dc25e3ee7987066a891658b7a85be8cb6c70515123e0e01f0cf0f0b56

C:\Windows\System32>
```

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



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



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

localhost:9000/projects/create?mode=manual

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

1 of 2

### Create a local project

Project display name \*

sonarqube ✓

Project key \*

sonarqube ✓

Main branch name \*

main

The name of your project's default branch [Learn More](#)

Cancel Next

**Embedded database should be used for evaluation purposes only**

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

Setup the project by selecting the following options.

localhost:9000/projects/create?mode=manual&setncd=true

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

2 of 2

### Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

#### Choose the baseline for new code for this project

☒ Use the global setting

**Previous version**

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

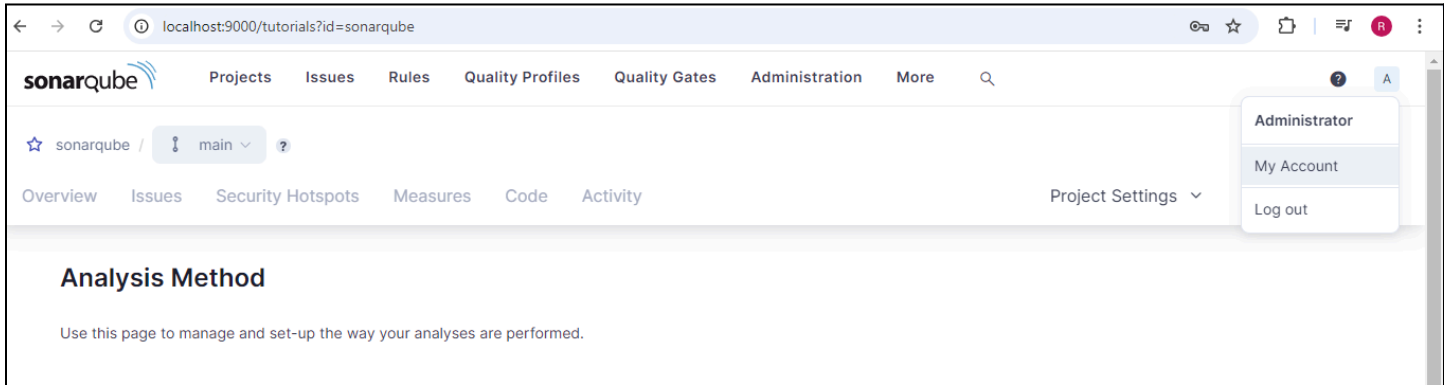
☐ Define a specific setting for this project

☐ Previous version

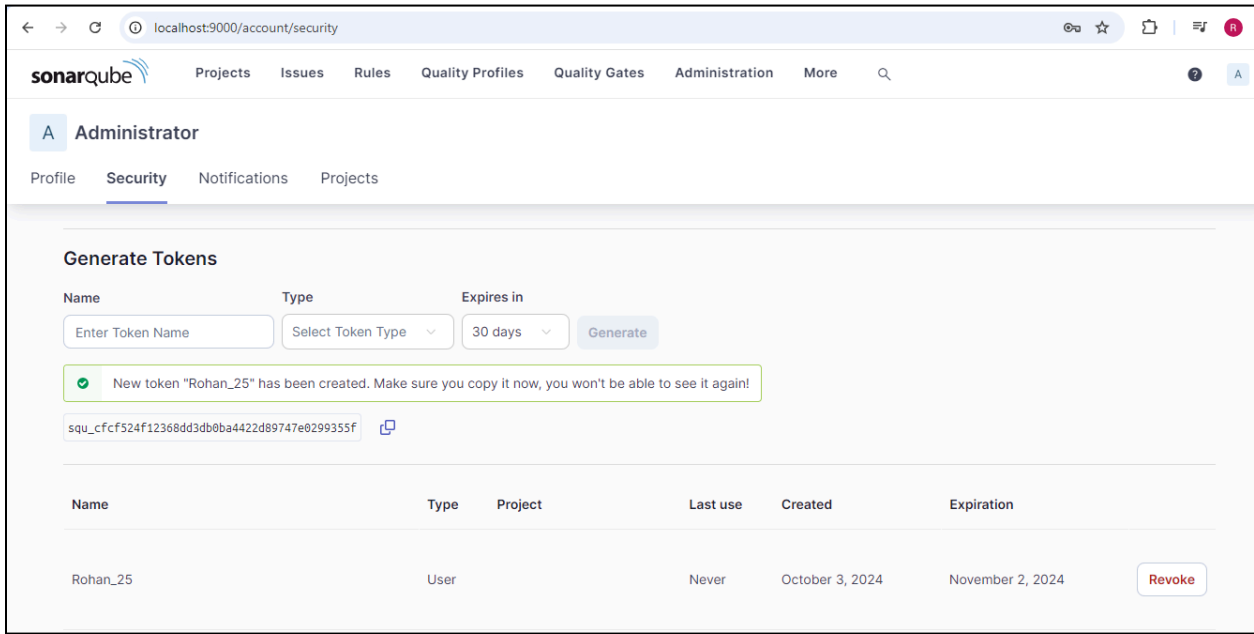
Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

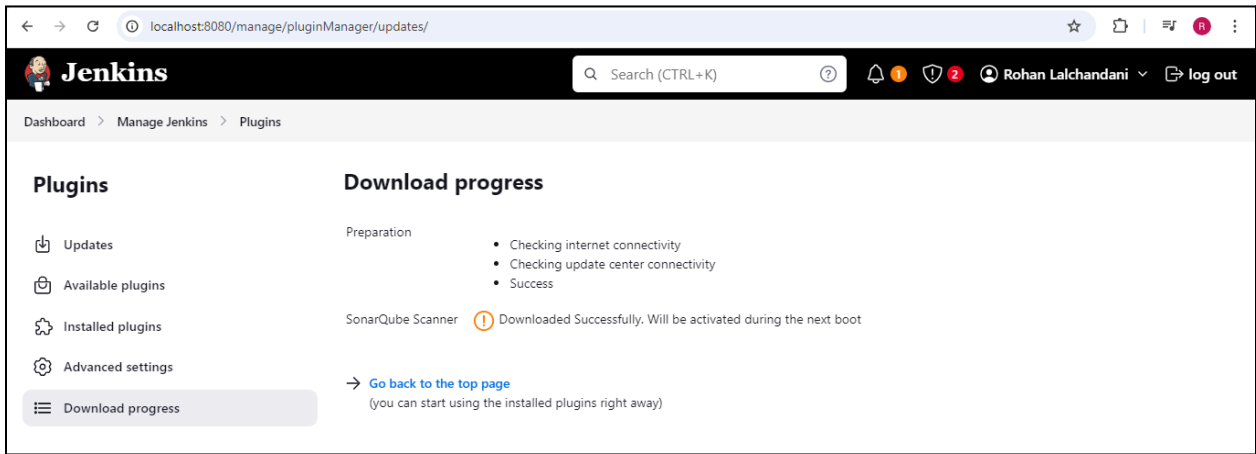
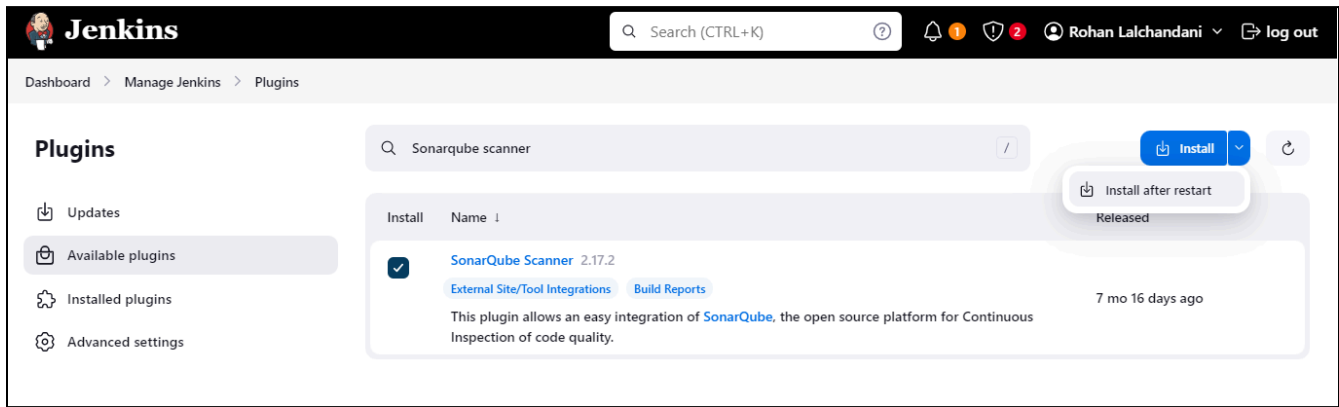
Now, for setting up sonarqube in Jenkins we need to have the login and password, for that we need to generate token.  
Go to My Account > Security



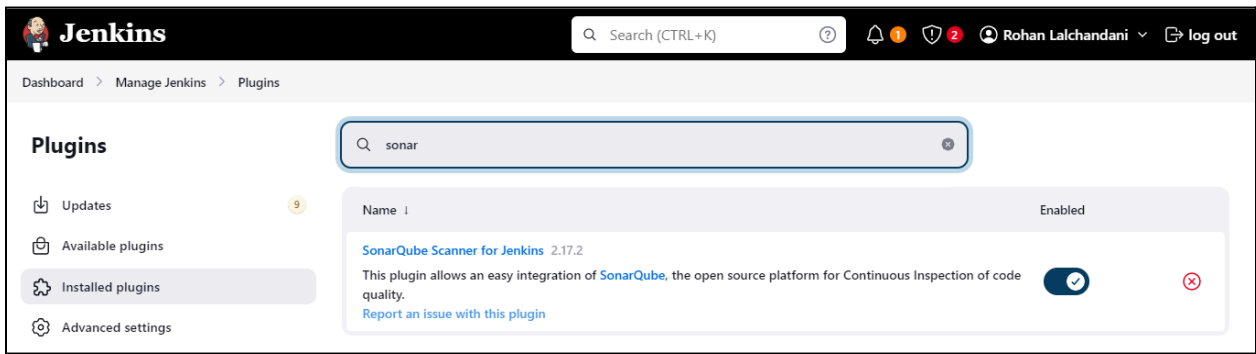
Type the details and click on generate token, copy the token and save it in a notepad.



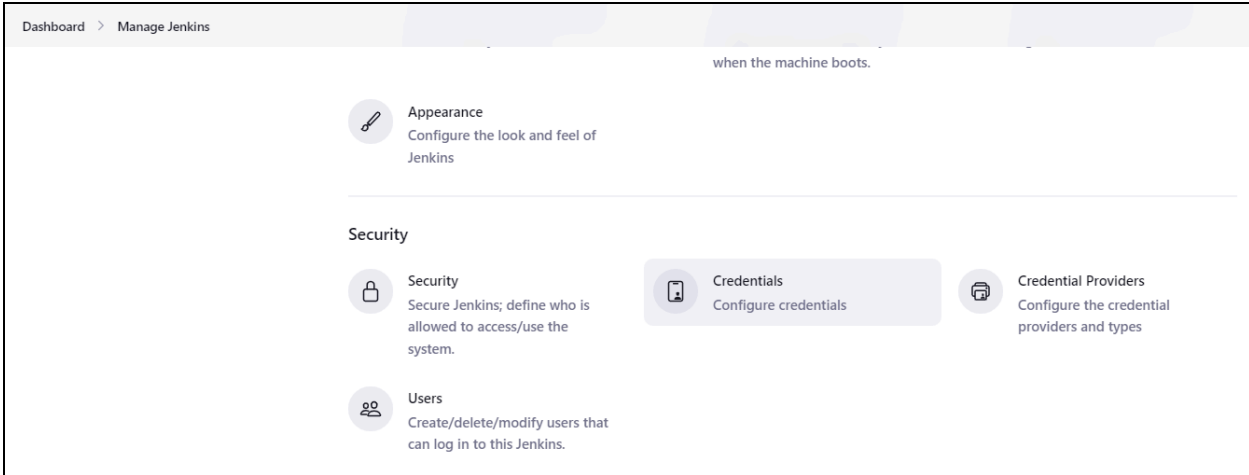
Setup the project and come back to Jenkins Dashboard. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



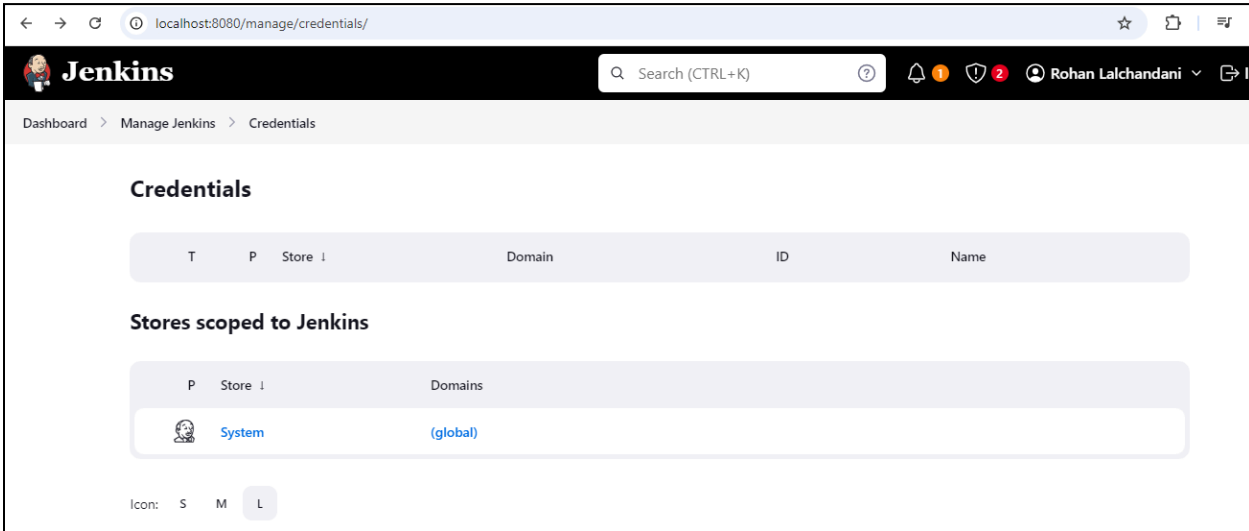
Go to Installed plugins to see if sonarqube scanner is successfully installed



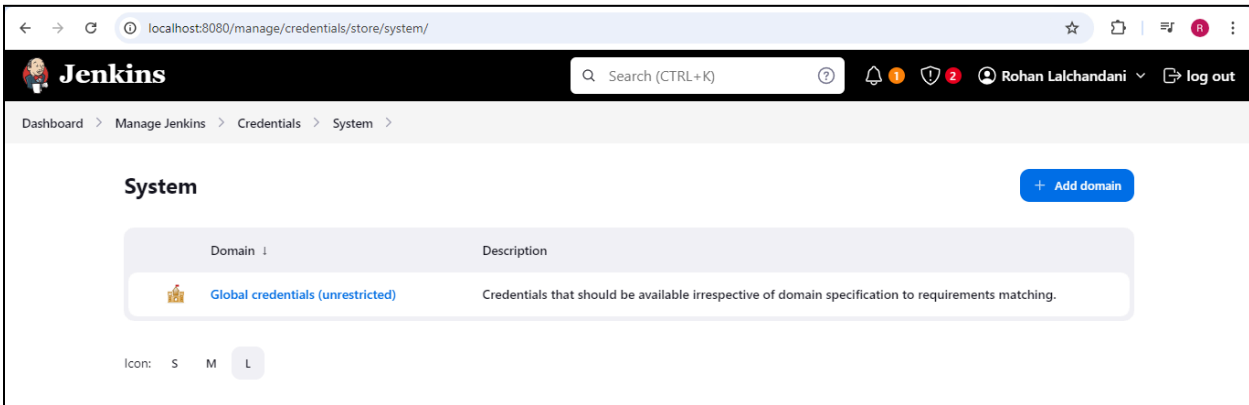
Go to Manage Jenkins > Credentials



## Click on System

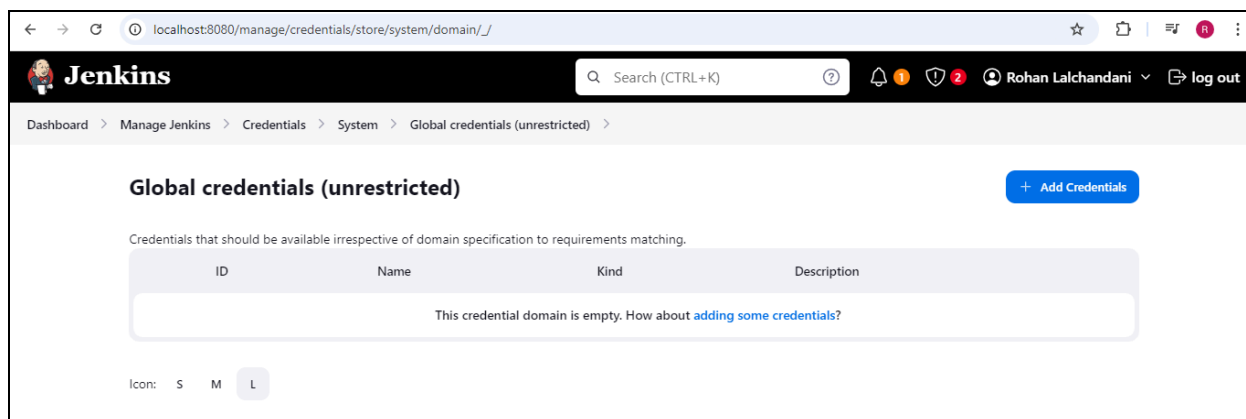


## Click on Global Credentials

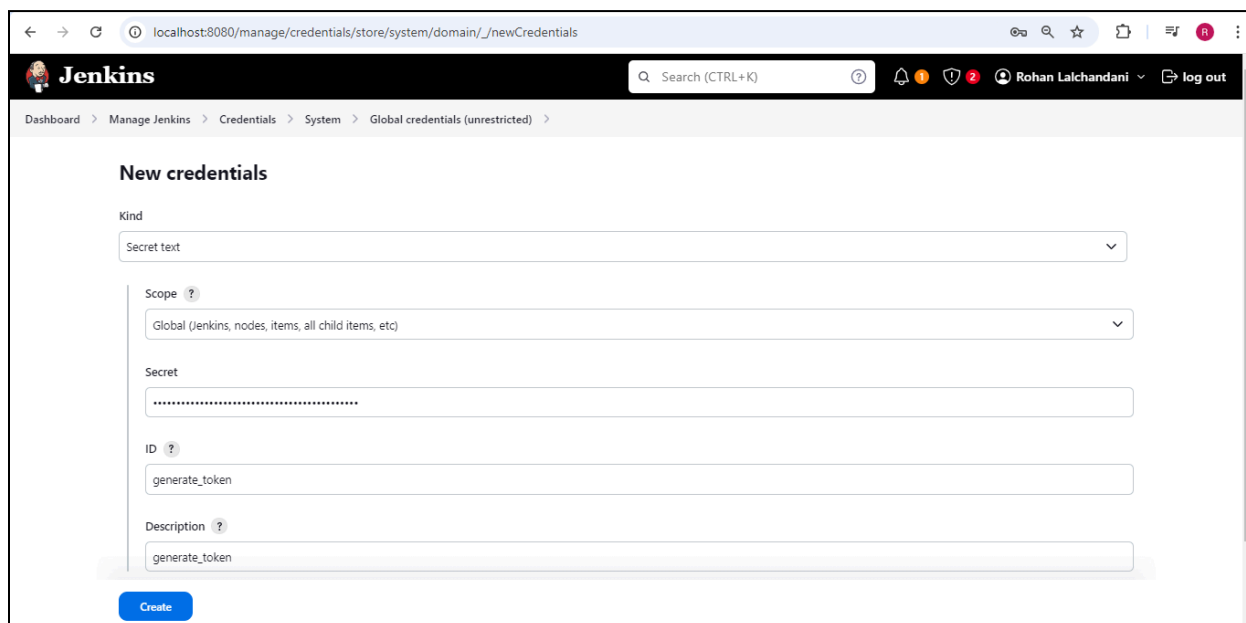


## Click on Add Credentials

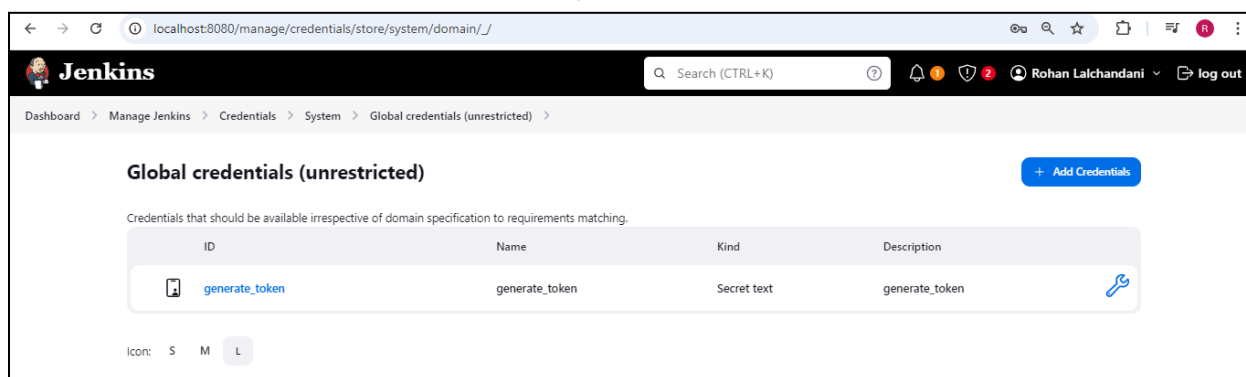




Now paste the generated token from sonarqube in the secret box and fill in other details as shown below.

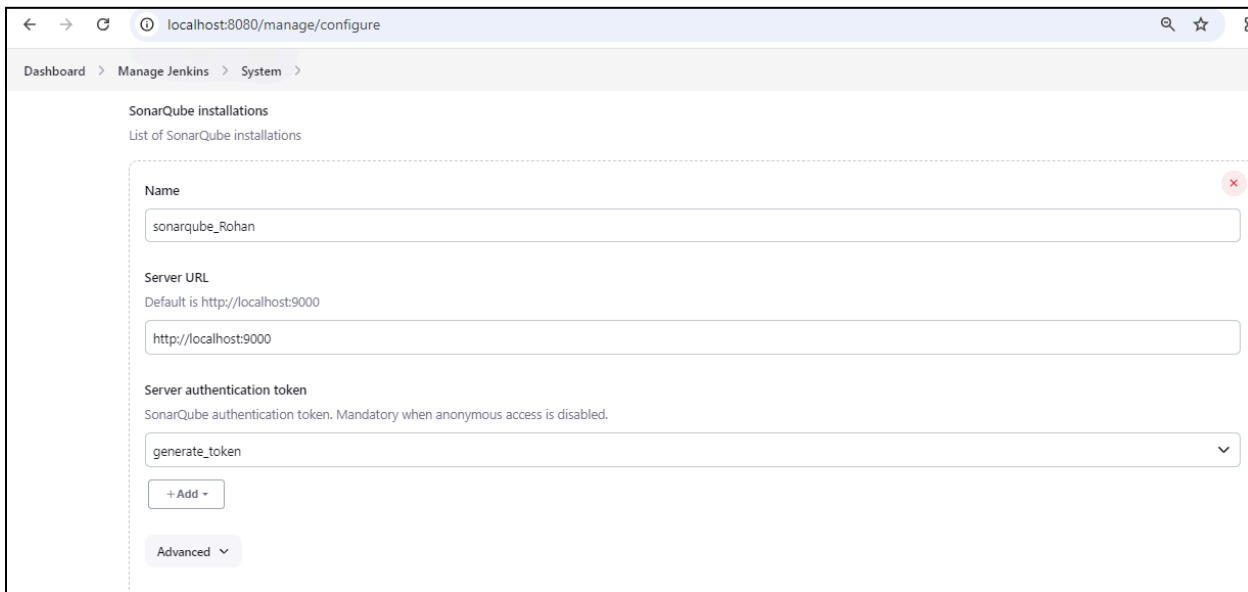
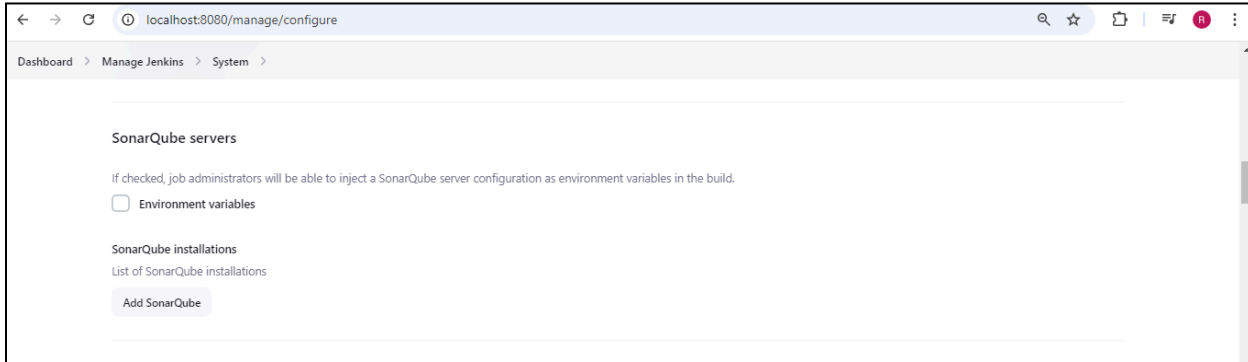


Token has been added successfully to the Jenkins



Configure SonarQube Scanner in Jenkins

- Go to Manage Jenkins > System. Scroll down to Sonarqube servers, check on environment variables.
- Select the token from drop down menu.
- Click on save.



## Configure SonarQube Scanner in Jenkins

- Go to Manage Jenkins > Global Tool Configuration.
- Scroll down to SonarQube Scanner.
- Choose the latest version and select Install automatically

localhost:8080/manage/configureTools/

Dashboard > Manage Jenkins > Tools

### SonarQube Scanner installations

Add SonarQube Scanner

**SonarQube Scanner**

Name

sonarqube\_Rohan

☒ Install automatically ?

**Install from Maven Central**

Version

SonarQube Scanner 6.2.1.4610

Add Installer

Add SonarQube Scanner

Save Apply

Go to your Jenkins Dashboard and Create a new Job

localhost:8080

**Jenkins** Search (CTRL+K) Rohan Lalchandani log out

Dashboard >

+ New Item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

1	Idle
2	Idle

## Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

### Start building your software project

Create a job +

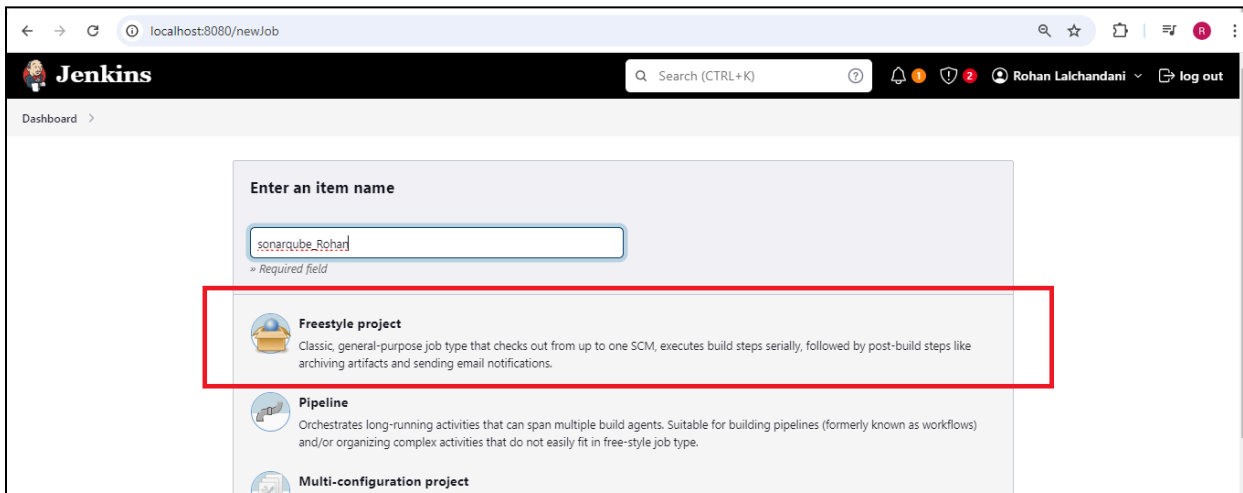
### Set up a distributed build

Set up an agent

Configure a cloud

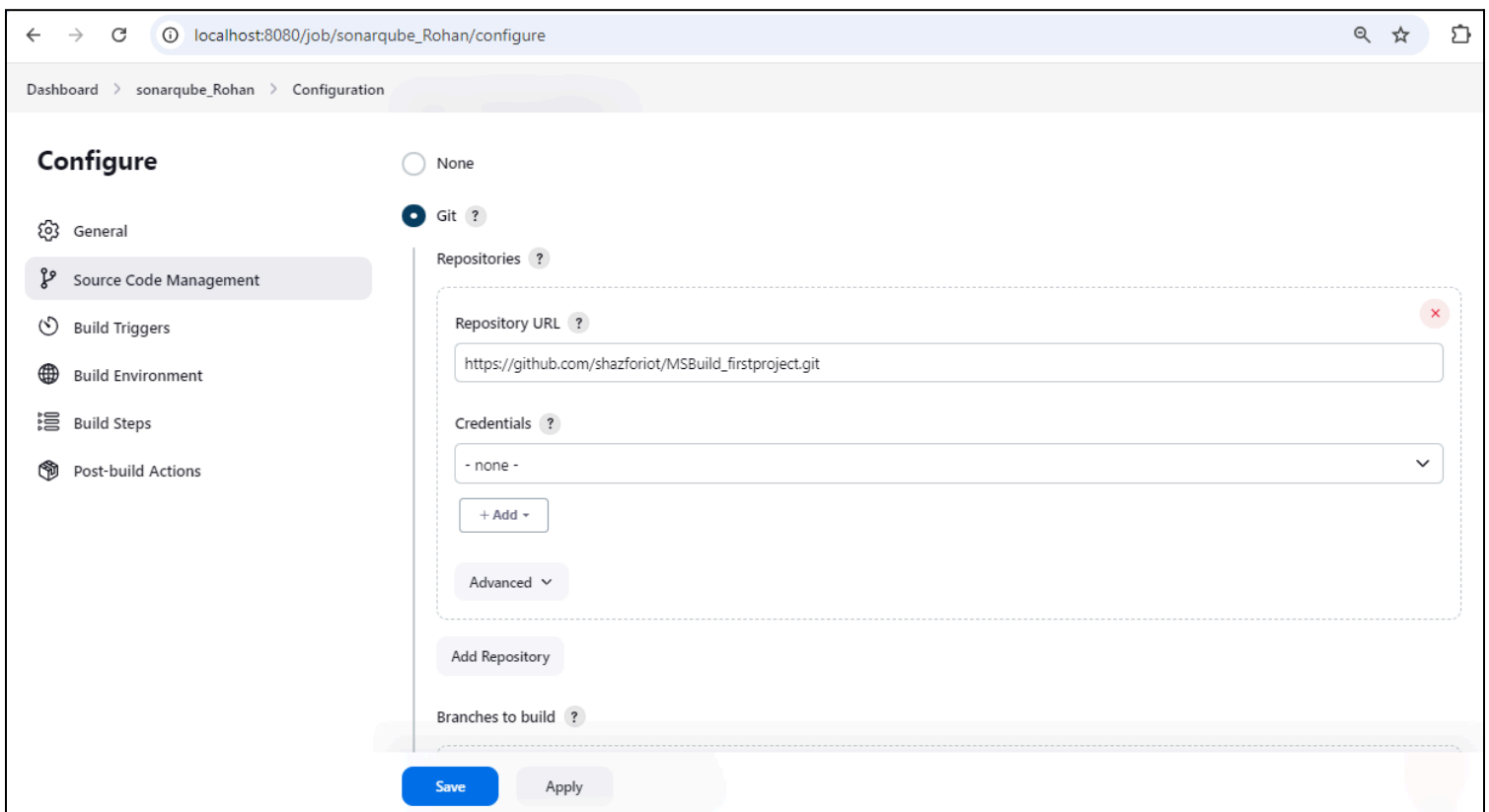
Learn more about distributed builds

Enter name of the project and select freestyle project



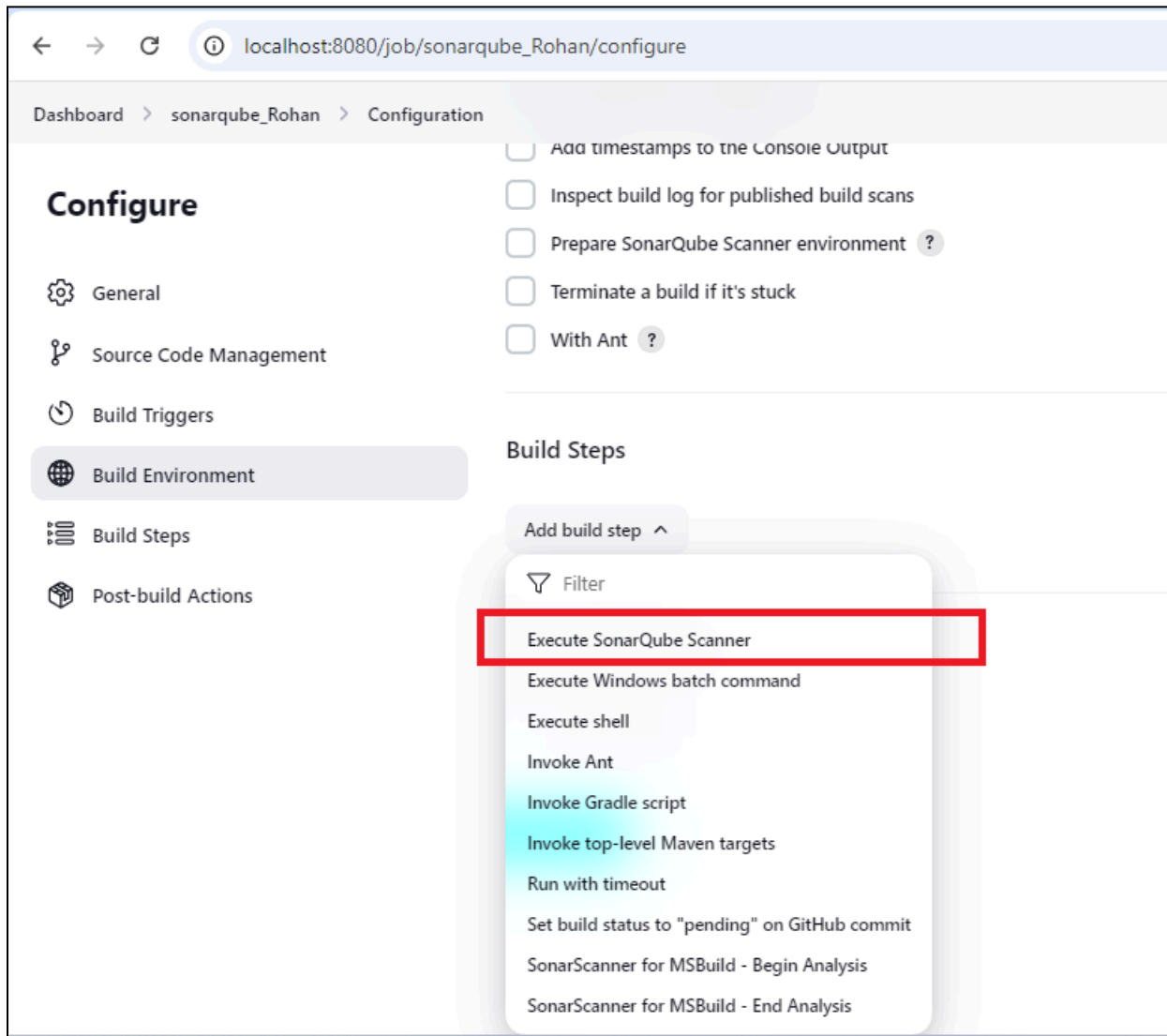
The screenshot shows the Jenkins 'New Job' page. The browser address bar indicates the URL is localhost:8080/newJob. The Jenkins logo and a search bar are at the top. Below the header, there's a section titled 'Enter an item name' with a text input field containing 'sonarqube\_Rohan'. A red box highlights the 'Freestyle project' option, which is described as a 'Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.' Other options like 'Pipeline' and 'Multi-configuration project' are visible below.

Under Configure put the github link in source code management  
[https://github.com/shazforiot/MSBuild\\_firstproject.git](https://github.com/shazforiot/MSBuild_firstproject.git)



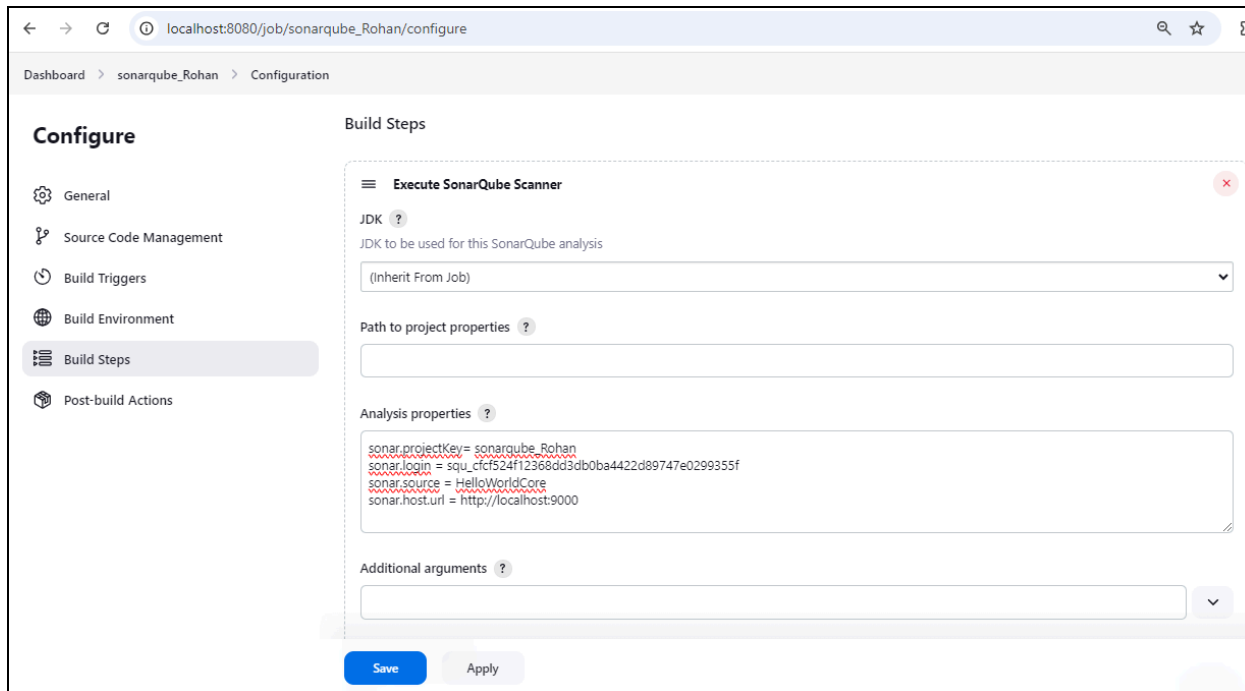
The screenshot shows the 'Configure' page for the job 'sonarqube\_Rohan'. The left sidebar has a 'Configure' section with options: General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main area shows the 'Git' source code management option selected. Under 'Repositories', the 'Repository URL' is set to 'https://github.com/shazforiot/MSBuild\_firstproject.git'. The 'Credentials' dropdown is set to '- none -'. There are buttons for '+ Add', 'Advanced', and 'Add Repository'. At the bottom, there are 'Save' and 'Apply' buttons.

In build steps Select “Execute SonarQube scanner”

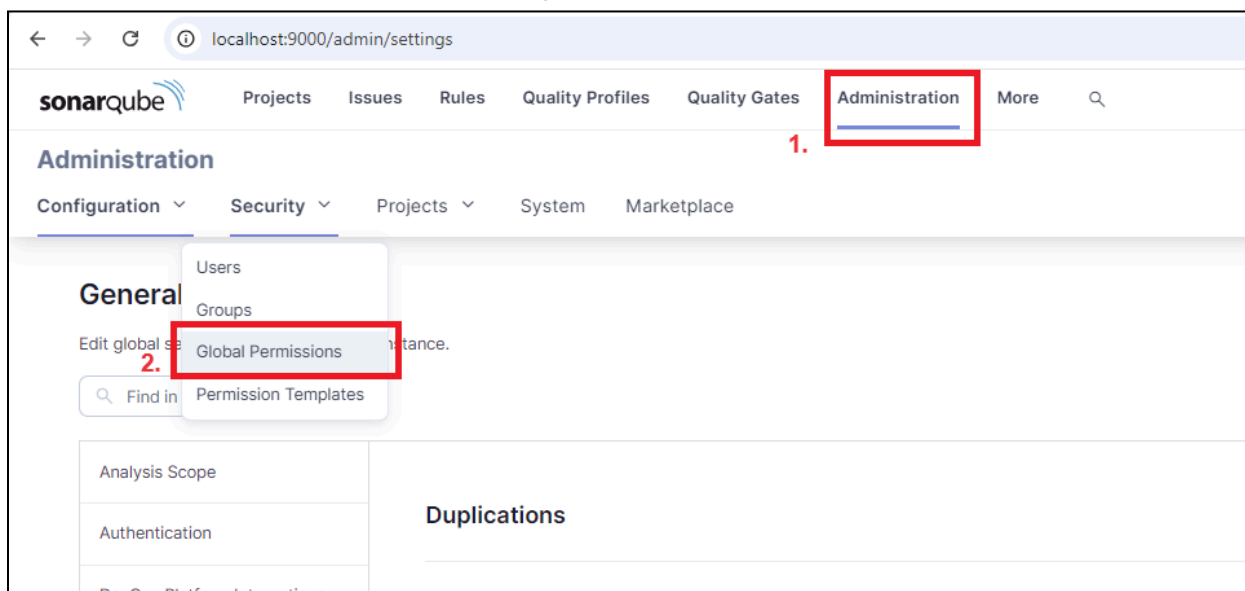


Type the following in analysis properties:

- sonar.projectKey=my\_project\_name
- sonar.login=your\_generated\_token
- sonar.sources>HelloWorldCore
- sonar.host.url=http://localhost:9000



Now we need to enable access to sonarqube.  
For that go to the sonarqube website  
click on administration and then global permissions.



Click on **Anyone** in the checkbox as shown below.

sonarqube

Projects

Issues

Rules

Quality Profiles

Quality Gates

Administration

More

Administration

Configuration

Security

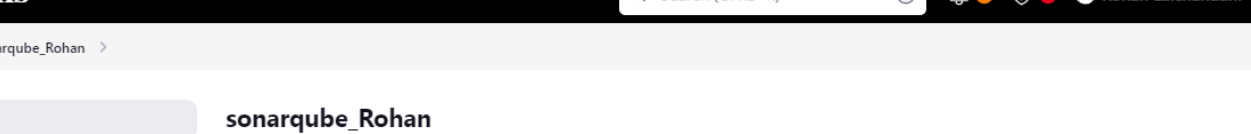
Projects

System

Marketplace

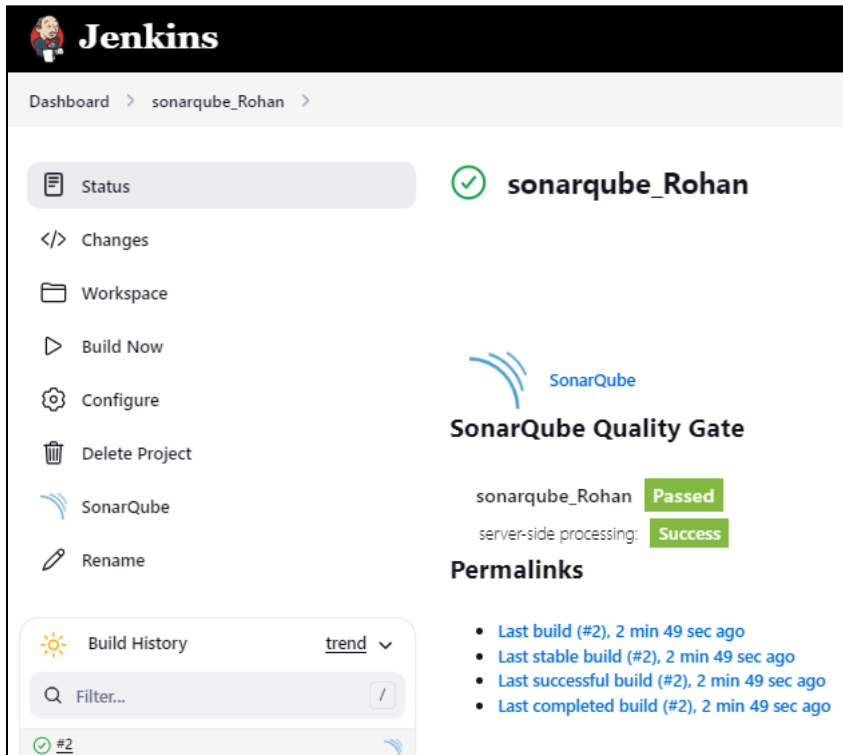
	Administer System	Administer	Execute Analysis	Create
<div>sonar-administrators</div> <div>System administrators</div>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
<div>sonar-users</div> <div>Every authenticated user automatically belongs to this group</div>	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects
<div>Anyone <span>DEPRECATED</span></div> <div>Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users.</div>	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

Go to the Jenkins Dashboard and click on “Build Now”



The screenshot shows the Jenkins interface for a project named 'sonarqube\_Rohan'. The 'Build Now' button is highlighted with a red rectangle. The page includes a sidebar with navigation options: Status, Changes, Workspace, Build Now, Configure, Delete Project, SonarQube, and Rename. The main area displays the project name and a 'Build Now' button. There are also links for 'Permalinks' and 'SonarQube'. The top navigation bar shows the Jenkins logo, a search bar, and user information (Rohan Lalchandani) with a 'log out' button.

The build has been completed successfully.



The Jenkins dashboard for the 'sonarqube\_Rohan' project shows a successful build status. The left sidebar contains links to Status, Changes, Workspace, Build Now, Configure, Delete Project, SonarQube, and Rename. The main area displays the SonarQube logo and 'SonarQube Quality Gate' with a 'Passed' status. Below this, 'server-side processing' is marked as 'Success'. A 'Permalinks' section lists build details for build #2, which completed 2 minutes and 49 seconds ago. A 'Build History' section at the bottom left shows a filter and a list of builds.

Dashboard > sonarqube\_Rohan >

Status

Changes

Workspace

Build Now

Configure

Delete Project

SonarQube

Rename

Build History trend

Filter...

#2

SonarQube

SonarQube Quality Gate

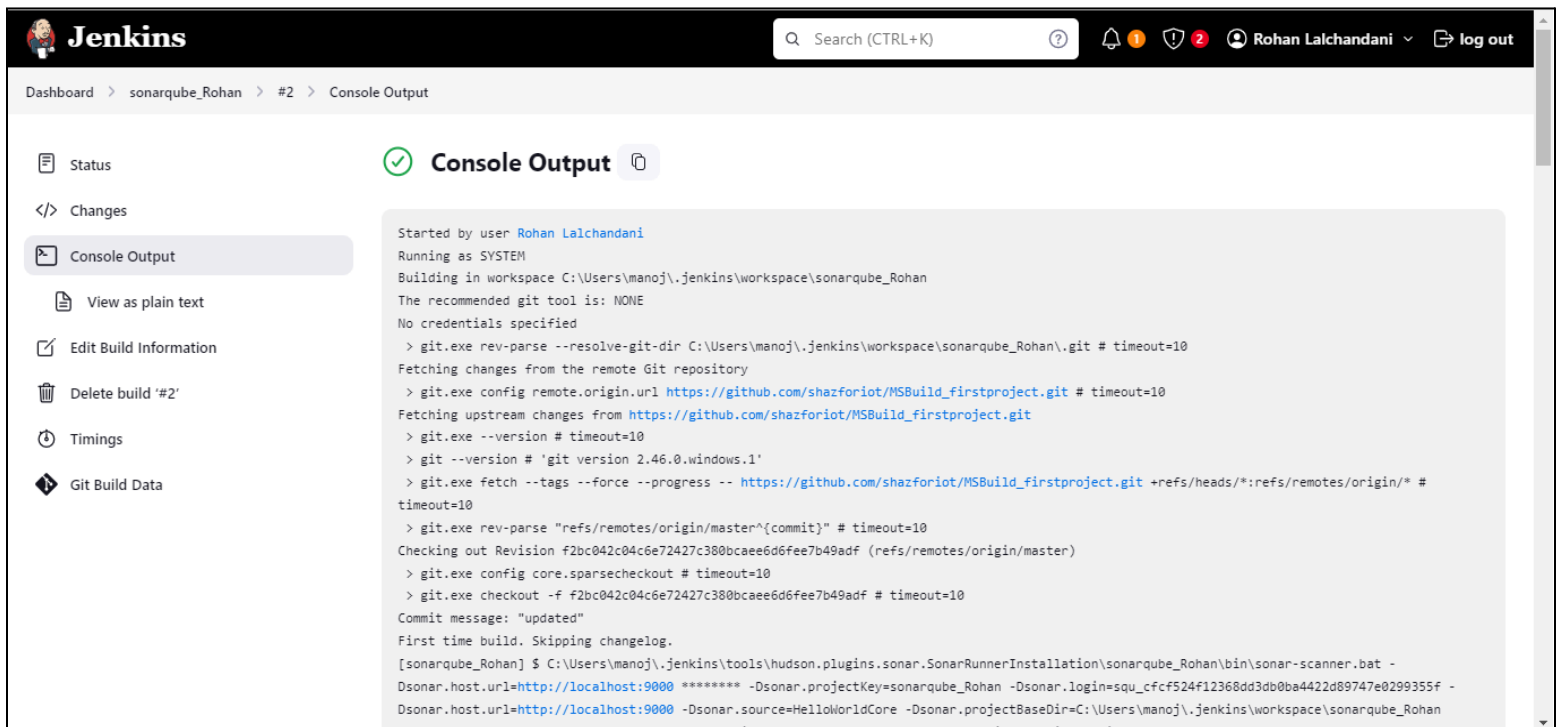
sonarqube\_Rohan Passed

server-side processing: Success

Permalinks

- Last build (#2), 2 min 49 sec ago
- Last stable build (#2), 2 min 49 sec ago
- Last successful build (#2), 2 min 49 sec ago
- Last completed build (#2), 2 min 49 sec ago

Go to the console output.



The Jenkins console output for build #2 shows the build process details. The output starts with the user 'Rohan Lalchandani' and the system running as 'SYSTEM'. It shows the build directory and the recommended git tool is 'NONE'. The build then fetches changes from the remote Git repository and fetches upstream changes from 'https://github.com/shazforiot/MSBuild\_firstproject.git'. The build then checks out the revision 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf' and configures the core.sparsecheckout. The build then checks out the revision 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf' and commits the message 'updated'. The build then runs the SonarScanner with the following command: 'C:\Users\manoj\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\_Rohan\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube\_Rohan -Dsonar.login=squ\_cfcf524f12368dd3db0ba4422d89747e0299355f -Dsonar.host.url=http://localhost:9000 -Dsonar.source=HelloWorldCore -Dsonar.projectBaseDir=C:\Users\manoj\.jenkins\workspace\sonarqube\_Rohan'.

Dashboard > sonarqube\_Rohan > #2 > Console Output

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#2'

Timings

Git Build Data

Console Output

Started by user Rohan Lalchandani

Running as SYSTEM

Building in workspace C:\Users\manoj\.jenkins\workspace\sonarqube\_Rohan

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\Users\manoj\.jenkins\workspace\sonarqube\_Rohan\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url https://github.com/shazforiot/MSBuild\_firstproject.git # timeout=10

Fetching upstream changes from https://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 -- https://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"

First time build. Skipping changelog.

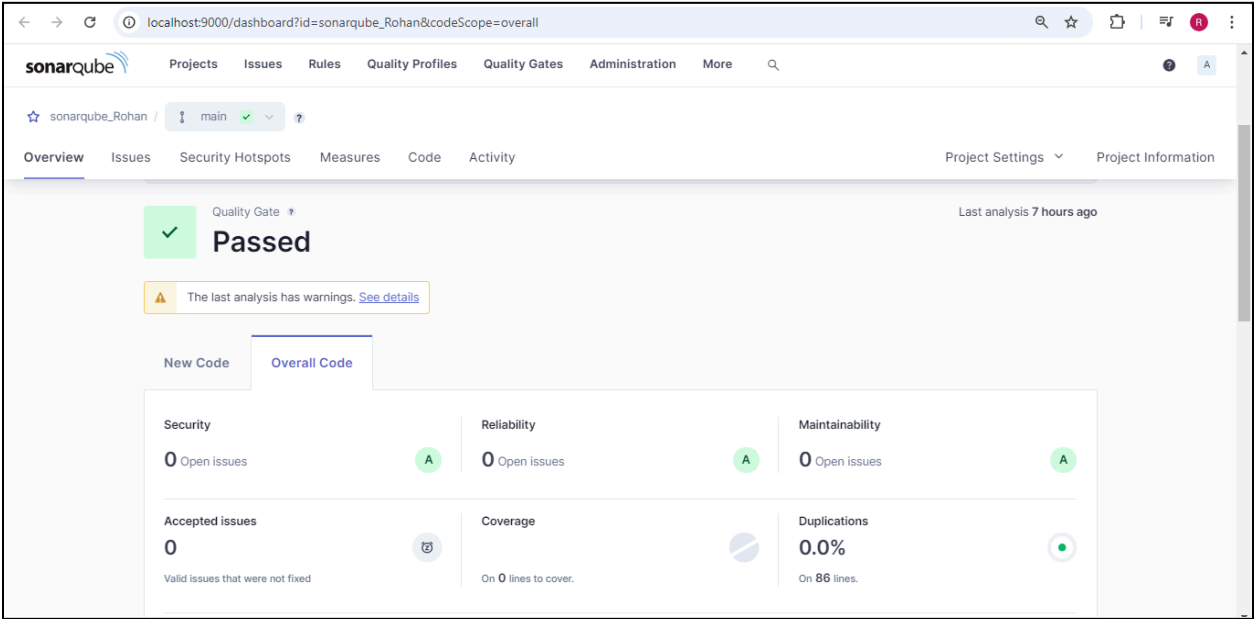
[sonarqube\_Rohan] \$ C:\Users\manoj\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\_Rohan\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube\_Rohan -Dsonar.login=squ\_cfcf524f12368dd3db0ba4422d89747e0299355f -Dsonar.host.url=http://localhost:9000 -Dsonar.source=HelloWorldCore -Dsonar.projectBaseDir=C:\Users\manoj\.jenkins\workspace\sonarqube\_Rohan



Scroll down click on the link as shown below to view the SonarQube analysis report.

```
02:34:34.415 INFO Analysis report compressed in 44ms, zip size=23.8 kB
02:34:34.448 INFO Analysis report uploaded in 31ms
02:34:34.451 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube_Rohan
02:34:34.451 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
02:34:34.451 INFO More about the report processing at http://localhost:9000/api/ce/task?id=91e88697-16f7-4df8-9bcc-69a576460a6c
02:34:34.468 INFO Analysis total time: 18.278 s
02:34:34.471 INFO SonarScanner Engine completed successfully
02:34:34.558 INFO EXECUTION SUCCESS
02:34:34.559 INFO Total time: 21.879s
Finished: SUCCESS
```

You can see the code has successfully passed.



## (EXTRA) : Building and Analysis of Maven project.

1. Create a maven project and paste the github link in source code management  
<https://github.com/edureka27/webapp.git>

The screenshot shows the SonarQube configuration page for a project named 'sonarqube2\_Rohan'. The 'Git' tab is selected in the 'Configure' sidebar. The 'Repositories' section contains a 'Repository URL' field with the value 'https://github.com/edureka27/webapp.git'. The 'Credentials' dropdown is set to '- none -'. Below this is an 'Add Repository' button. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' field with the value '\*/main'. Below this is an 'Add Branch' button. At the bottom of the configuration area are 'Save' and 'Apply' buttons.

2. Check on prepare sonarqube scanner

The screenshot shows the SonarQube configuration page for the same project, but with the 'Build Environment' tab selected. The 'Prepare SonarQube Scanner environment' checkbox is checked. Below this, the 'Server authentication token' dropdown is set to '- none -'. There are also checkboxes for 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Add timestamps to the Console Output', 'Inspect build log for published build scans', 'Terminate a build if it's stuck', and 'With Ant'.



Go to the console output and scroll down click on the analysis report link and you will be able to see the sonarqube analysis report.

