

Experiment No :7

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

PREREQUISITES:

1) Docker :

Run **docker -v** command. We use this command to check if docker is installed and running on your system.

```
C:\Users\praja>docker -v
Docker version 27.0.3, build 7d4bcd8
```

2) Install SonarQube Image:

The command **docker pull sonarqube** downloads a SonarQube image from Docker's online repository. This image lets you run SonarQube on your system using Docker without needing to install the full SonarQube software manually. It's like getting a ready-to-use version of SonarQube that can be started with Docker.

```
C:\Users\praja>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
```

3) Make sure **Jenkins** is already installed on your system before starting the process. Jenkins will be used to automate tasks, like running SonarQube for code analysis. If Jenkins isn't installed yet, you can download and set it up from the official Jenkins website.

STEPS:

Step1: The command `docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest` starts SonarQube in the background on port 9000 using Docker, allowing you to access it at <http://localhost:9000>

```
C:\Users\praja>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
650354d0f868ae4ad2d800426080076c604eb09f29b10d4a251aee70f51ce907
C:\Users\praja>
```

Containers

[Give feedback](#)

Container CPU usage ⓘ

282.96% / 800% (8 CPUs available)

Container memory usage ⓘ


1.69GB / 7.41GB


Show charts

Q

Search

Step2: After starting the SonarQube image, open your browser and go to <http://localhost:9000> to access SonarQube.





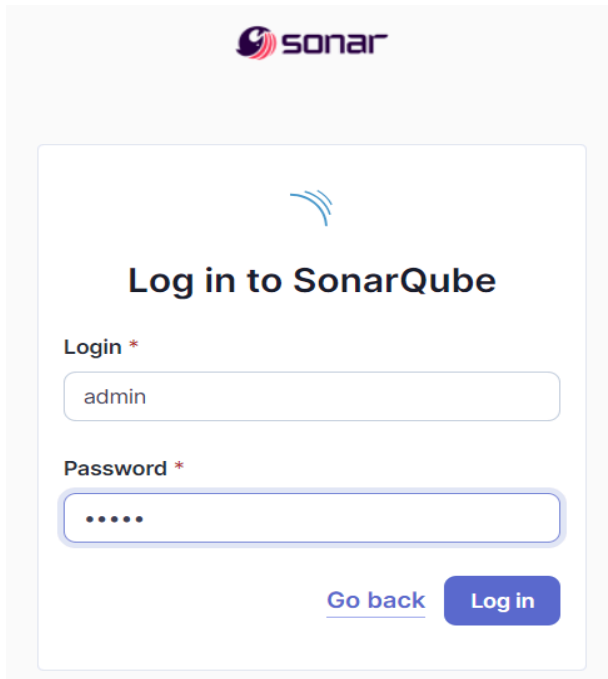
Log in to SonarQube


Login *

Password *

[Go back](#)

Step 3: On the SonarQube login page, use the default credentials: **Username: admin** , **Password: admin**. After logging in, you'll be prompted to change the password. Set a new password and make sure to remember it.

The image shows the SonarQube login page. At the top, there is a Sonar logo. Below it, the text "Log in to SonarQube" is displayed. There are two input fields: "Login *" with the value "admin" and "Password *" with five dots. At the bottom, there is a "Go back" link and a "Log in" button.



Log in to SonarQube

Login *

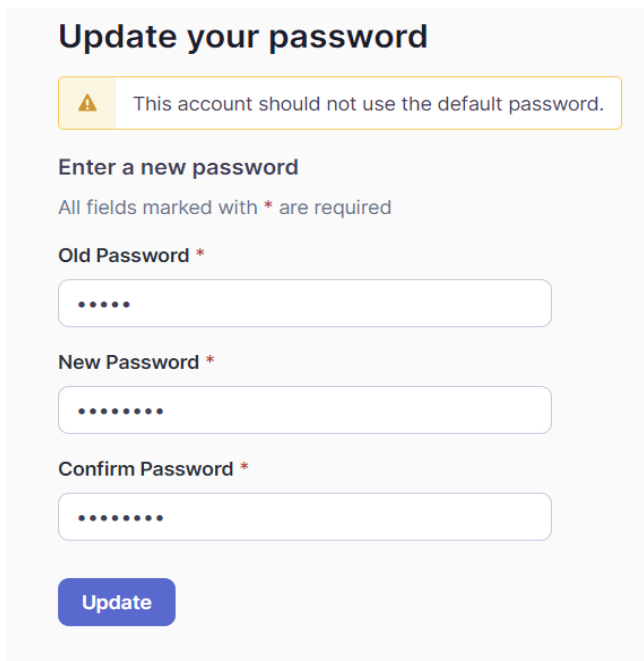
admin

Password *


.....

[Go back](#) Log in

Click on Log in

The image shows the SonarQube password update page. At the top, there is a warning message: "This account should not use the default password." Below this, the text "Update your password" is displayed. There are three input fields: "Old Password *" with five dots, "New Password *" with seven dots, and "Confirm Password *" with seven dots. At the bottom, there is an "Update" button.

Update your password

 This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

.....

New Password *

.....

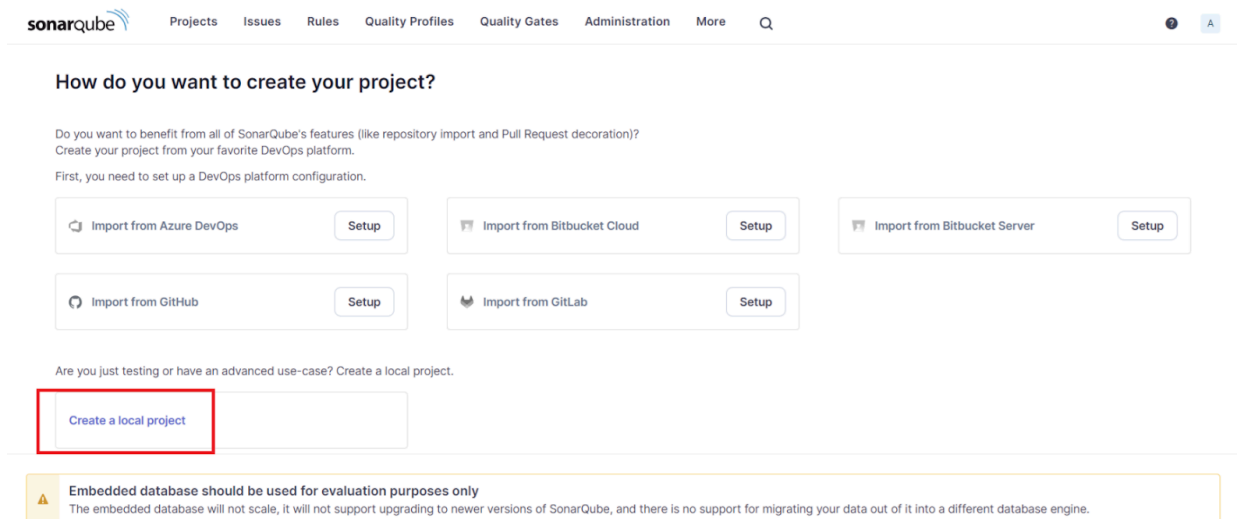
Confirm Password *

.....

Update

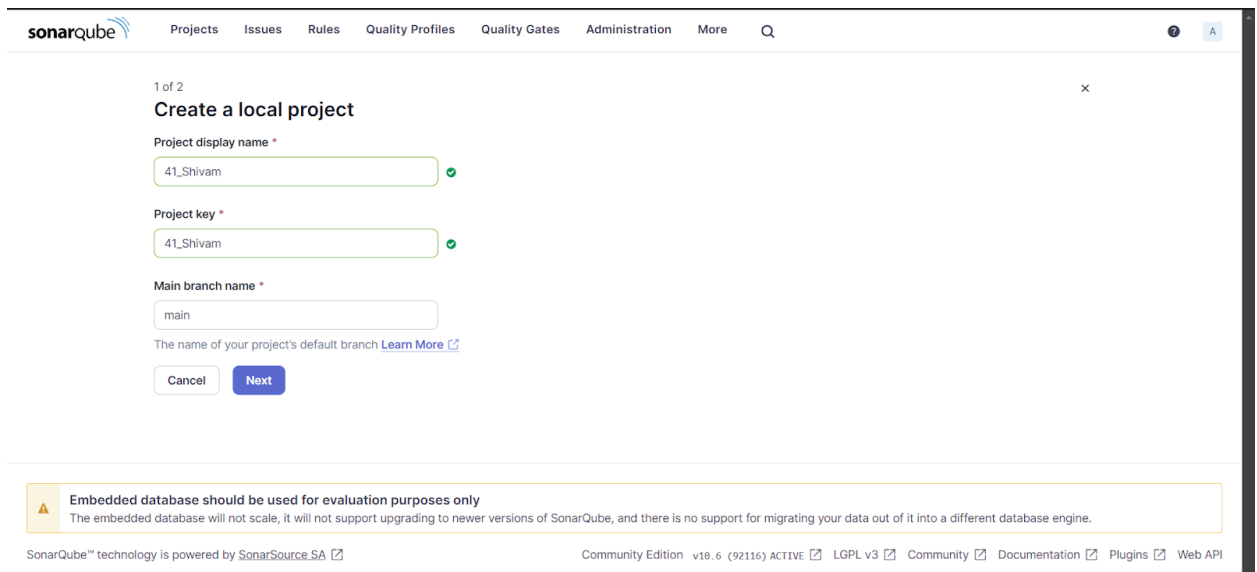
Click on Update .

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



The screenshot shows the SonarQube web interface with the title "How do you want to create your project?". It offers several options to import projects from external platforms like Azure DevOps, Bitbucket Cloud, Bitbucket Server, GitHub, and GitLab, each with a "Setup" button. Below these, there is a section for creating a local project, which includes a button labeled "Create a local project" (highlighted with a red box) and an empty text input field. A warning message at the bottom states: "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."

Step 5: Give your project , a display name and project key



The screenshot shows the "Create a local project" form in SonarQube. It is titled "1 of 2" and "Create a local project". The form contains three required fields: "Project display name *" with the value "41_Shivam", "Project key *" with the value "41_Shivam", and "Main branch name *" with the value "main". Each field has a green checkmark indicating it is valid. Below the fields, there is a note: "The name of your project's default branch [Learn More](#)". At the bottom of the form, there are "Cancel" and "Next" buttons. A warning message at the bottom of the page is identical to the one in Step 4. The footer of the page includes "SonarQube™ technology is powered by SonarSource SA" and "Community Edition v19.6 (92116) ACTIVE" along with links to documentation, plugins, and the web API.

Step 6: Configure the project by providing the necessary settings like choosing the baseline for the new code for the project , then click **Create** to finalize the setup.

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.

☐ Number of days
Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.

Scroll Down

☐ Previous version
Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

☐ Number of days
Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
Recommended for projects following continuous delivery.

☐ Reference branch
Choose a branch as the baseline for the new code.
Recommended for projects using feature branches.

[Back](#) [Create project](#)

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.

Click on Create project

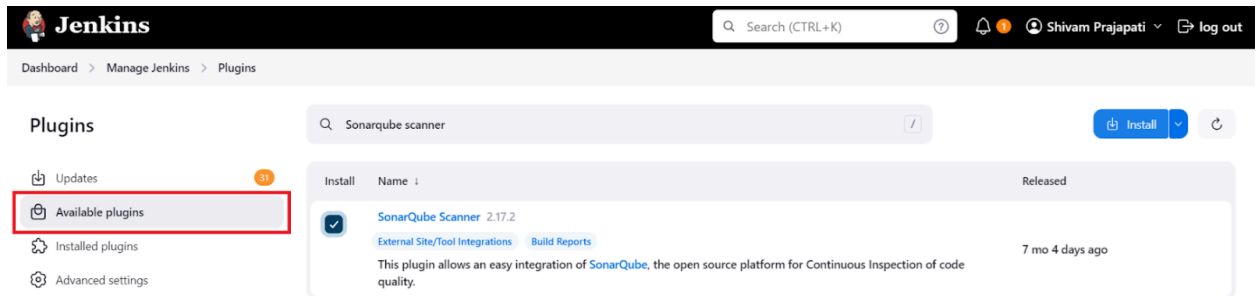
Step 7: Open Jenkins by going to **http://localhost:<port_number>** in your browser, replacing <port_number> with the specific port Jenkins is running on.

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are sections for 'Build Queue' (empty) and 'Build Executor Status' (showing 2 idle executors: 'Shivam' and 'Slave1', both offline). The main area displays a table of recent builds. The table has columns for status (S), icon (W), name, last success, last failure, and last duration. The builds listed are '41_SonarQube', 'Devops pipeline1', 'DevOps_Pipeline', 'Pipeline_DevOps', 'SonarQube_41', 'sonarqube_41_lab7', 'sonarqube_41_lab8', 'Sonarqube_lab7_41', and 'Sonarqube_lab8_41'. The 'sonarqube_41_lab7' and 'sonarqube_41_lab8' builds are marked as failed with red 'X' icons and error codes #1 and #2 respectively.

S	W	Name	Last Success	Last Failure	Last Duration
...	☀	41_SonarQube	N/A	N/A	N/A
✓	☀	Devops pipeline1	1 mo 23 days #24	N/A	1.3 sec
...	☀	DevOps_Pipeline	N/A	N/A	N/A
✓	☀	Pipeline_DevOps	1 mo 3 days #2	N/A	0.75 sec
...	☀	SonarQube_41	N/A	N/A	N/A
✗	☁	sonarqube_41_lab7	N/A	3 days 12 hr #1	3.8 sec
✗	☁	sonarqube_41_lab8	N/A	3 days 6 hr #2	2.9 sec
✓	☁	Sonarqube_lab7_41	3 days 11 hr #42	3 days 12 hr #40	25 sec
✓	☁	Sonarqube_lab8_41	2 days 11 hr #14	2 days 11 hr #13	15 min

Step 8: Now go to Manage Jenkin then go for Plugins followed by Available plugins search for **Sonarqube Scanner** where we are going to install it as a plugin.

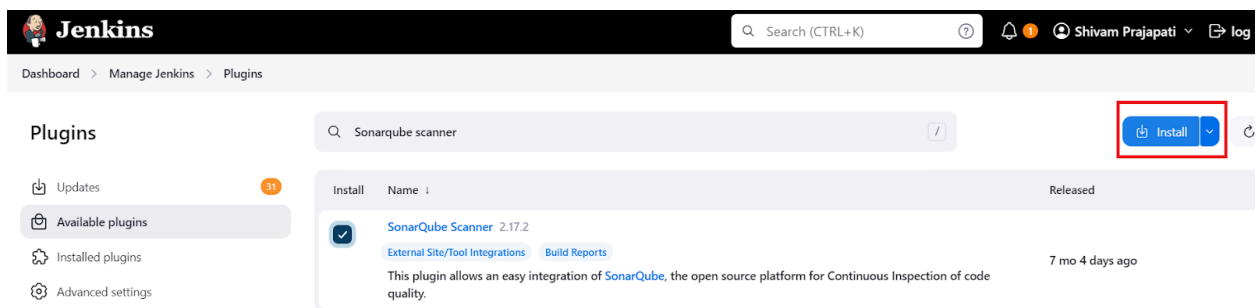
The screenshot shows the 'Manage Jenkins' page. At the top, there's a notification about a new version of Jenkins (2.462.2) available for download. Below this, there's a section for 'System Configuration' with four options: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), and 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand). The 'Plugins' option is highlighted with a red box. The 'Plugins' section describes adding, removing, disabling, or enabling plugins that can extend the functionality of Jenkins.



The screenshot shows the Jenkins 'Plugins' page. On the left sidebar, the 'Available plugins' option is highlighted with a red rectangle. The main content area shows a search bar with 'Sonarqube scanner' entered. Below the search bar, a table lists available plugins. The first entry is 'SonarQube Scanner' version 2.17.2, released 7 months and 4 days ago. A red box highlights the 'Install' button for this plugin.

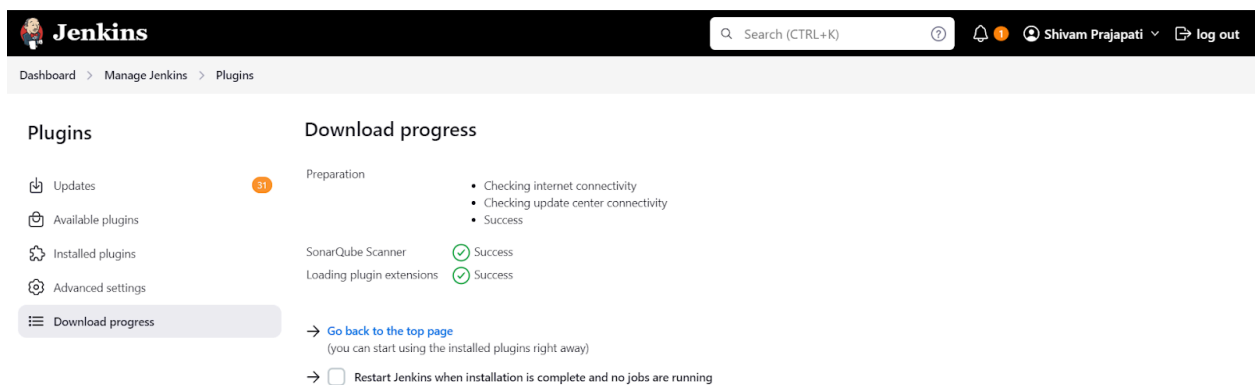
Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	7 mo 4 days ago

Click on Available Plugins.



This screenshot is similar to the previous one, but the 'Install' button for the 'SonarQube Scanner' plugin is highlighted with a red rectangle.

Search in the Search bar the required Plugin Name and click on Install.



The screenshot shows the 'Download progress' page in Jenkins. The left sidebar has 'Download progress' selected. The main content area shows the progress of installing the 'SonarQube Scanner' plugin. The progress is 100% complete, with a green checkmark indicating success. Below the progress bar, there are links to 'Go back to the top page' and 'Restart Jenkins when installation is complete and no jobs are running'.

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

SonarQube Scanner ☒ Success

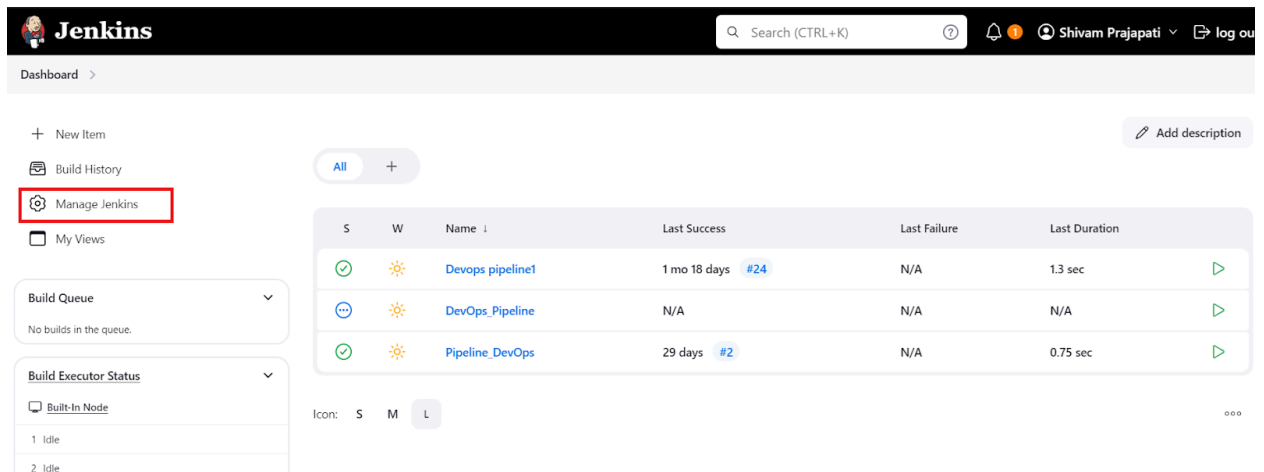
Loading plugin extensions ☒ Success

→ [Go back to the top page](#)
(you can start using the installed plugins right away)

→ ☐ Restart Jenkins when installation is complete and no jobs are running

Plugin Installed Successfully.

Step 9: In Jenkins, go to **Manage Jenkins** → **System**, then find **SonarQube** servers. Add a new server, and if required, include the authentication token for secure access

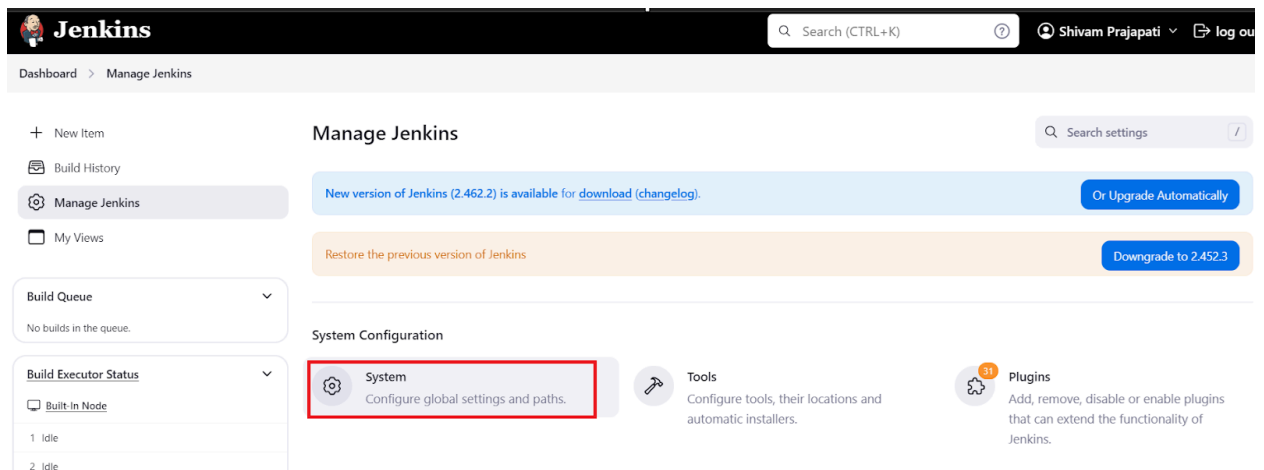


The screenshot shows the Jenkins Dashboard. The left sidebar contains navigation links: '+ New Item', 'Build History', 'Manage Jenkins' (highlighted with a red box), and 'My Views'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node, 1 Idle, 2 Idle). The main content area shows a table of pipelines:

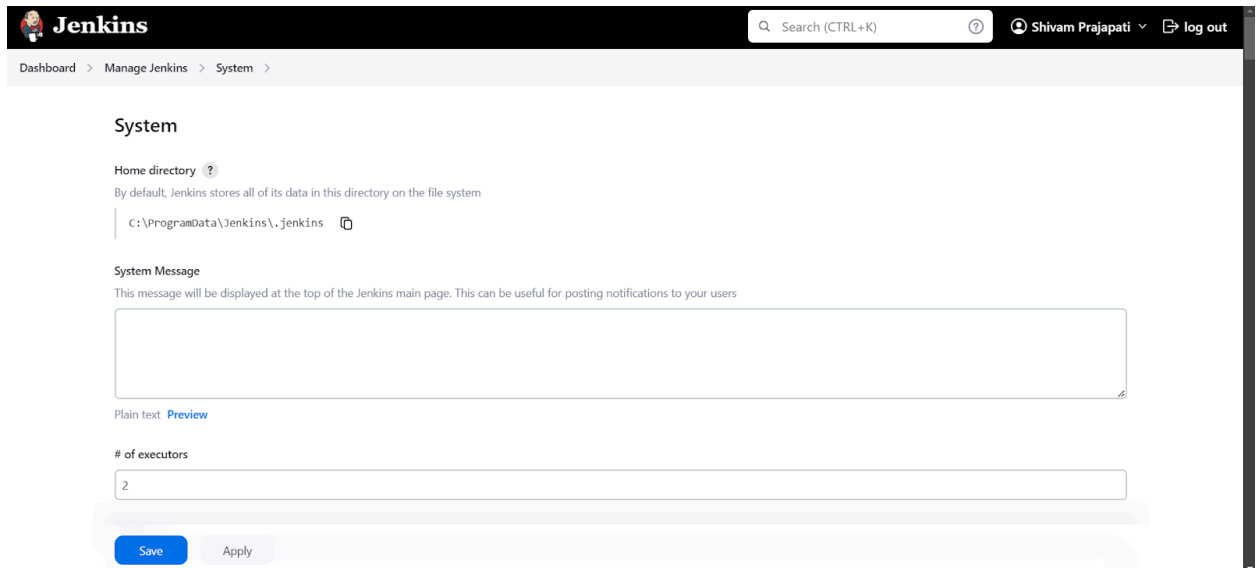
S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	Devops pipeline1	1 mo 18 days #24	N/A	1.3 sec
⋮	☀	DevOps_Pipeline	N/A	N/A	N/A
✓	☀	Pipeline_DevOps	29 days #2	N/A	0.75 sec

Below the table, there are icons for 'S', 'M', and 'L'.

Go to system



The screenshot shows the 'Manage Jenkins' page. The left sidebar is the same as the dashboard. The main content area has a 'Manage Jenkins' header with a search bar. Below the header, there are two banners: 'New version of Jenkins (2.462.2) is available for download (changelog)' with a button 'Or Upgrade Automatically', and 'Restore the previous version of Jenkins' with a button 'Downgrade to 2.452.3'. Below these banners is the 'System Configuration' section, which includes three items: 'System' (highlighted with a red box, with the description 'Configure global settings and paths.'), 'Tools' (with the description 'Configure tools, their locations and automatic installers.'), and 'Plugins' (with the description 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.').



The image shows the Jenkins 'System' configuration page. At the top, there's a header with the Jenkins logo, a search bar, and user information. Below the header, a breadcrumb trail shows 'Dashboard > Manage Jenkins > System'. The main section is titled 'System'. It includes a 'Home directory' field with a help icon and a description: 'By default, Jenkins stores all of its data in this directory on the file system'. The value is 'C:\ProgramData\jenkins\jenkins'. Below this is a 'System Message' section with a text area and a 'Plain text' label. The '# of executors' field is set to '2'. At the bottom, there are 'Save' and 'Apply' buttons.

Jenkins

Search (CTRL+K) Shivam Prajapati log out

Dashboard > Manage Jenkins > System

System

Home directory ?
By default, Jenkins stores all of its data in this directory on the file system
C:\ProgramData\jenkins\jenkins

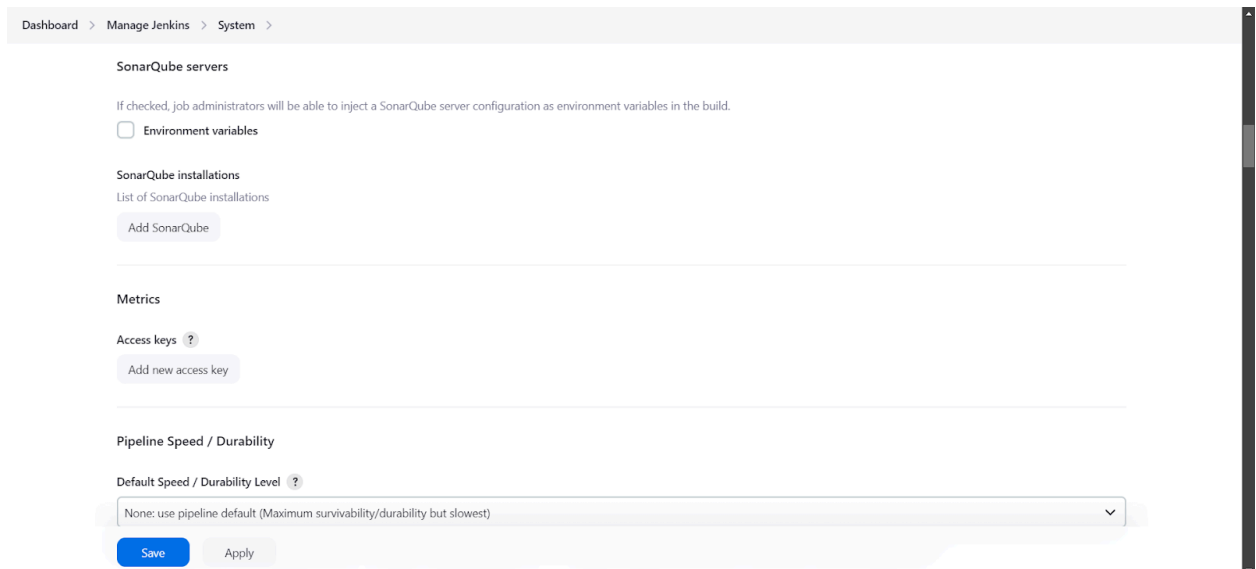
System Message
This message will be displayed at the top of the Jenkins main page. This can be useful for posting notifications to your users

Plain text [Preview](#)

of executors
2

[Save](#) [Apply](#)

Scroll Down



The image shows the Jenkins 'SonarQube' configuration page. It has the same header and breadcrumb trail as the previous page. The main section is titled 'SonarQube servers'. It includes a checkbox for 'Environment variables' with a description: 'If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.' Below this is a 'SonarQube installations' section with a description: 'List of SonarQube installations' and an 'Add SonarQube' button. The 'Metrics' section has an 'Access keys' label and an 'Add new access key' button. The 'Pipeline Speed / Durability' section has a 'Default Speed / Durability Level' dropdown menu set to 'None: use pipeline default (Maximum survivability/durability but slowest)'. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > Manage Jenkins > System

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables

SonarQube installations

List of SonarQube installations

[Add SonarQube](#)

Metrics

Access keys ?
[Add new access key](#)

Pipeline Speed / Durability

Default Speed / Durability Level ?
None: use pipeline default (Maximum survivability/durability but slowest)

[Save](#) [Apply](#)

Select Environment Variable and Click on Add Sonar Qube button in order to Add SonarQube Server to Jenkin

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables

SonarQube installations

List of SonarQube installations

[Add SonarQube](#)

Metrics

Access keys ?

[Add new access key](#)

Pipeline Speed / Durability

Do the required entries as shown below

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables

SonarQube installations

List of SonarQube installations

Name ✕

sonarqube

Server URL
Default is http://localhost:9000

http://localhost:9000

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.

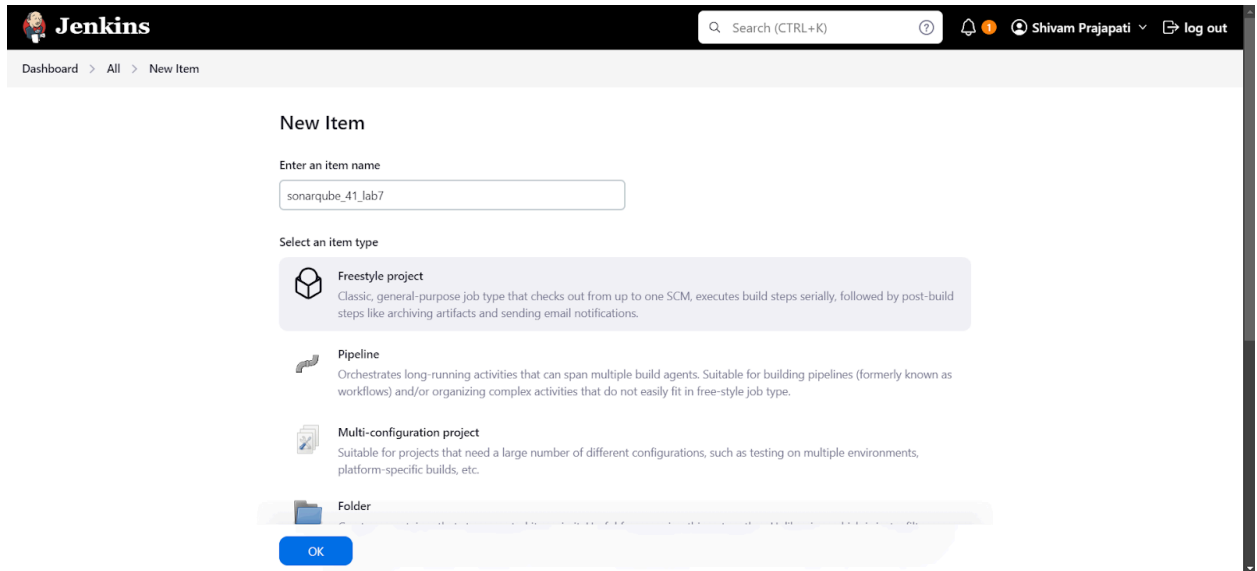
- none -

[+ Add](#)

[Save](#) [Apply](#)

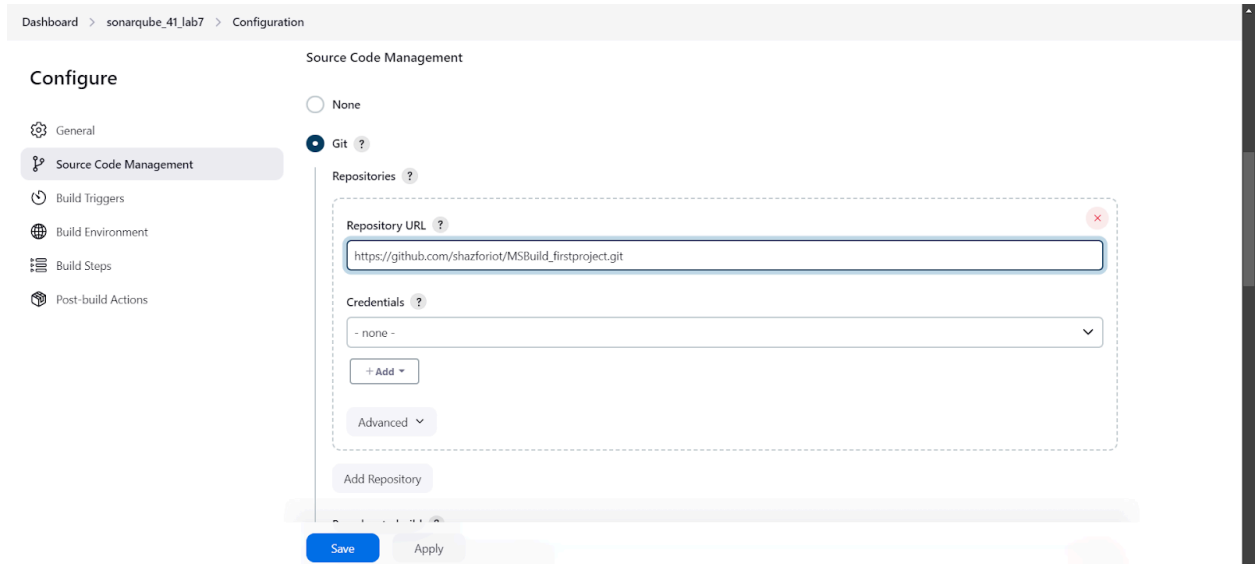
Click on save

Step 10: After configuration, create a New Item → choose a freestyle project.



Step 11: 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 12: Under Build Steps, enter Sonarqube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Dashboard > sonarqube_41_lab7 > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

☐ Add timestamps to the Console Output

☐ Inspect build log for published build scans

☐ Prepare SonarQube Scanner environment ?

☐ Terminate a build if it's stuck

☐ With Ant ?

Build Steps

Add build step ^

Filter

- Execute SonarQube Scanner
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit
- SonarScanner for MSBuild - Begin Analysis

REST API Jenkins 2.462.1

Click on execute sonar scanner

Dashboard > Sonarqube_lab7_41 > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

Build Steps

Execute SonarQube Scanner

JDK ?

JDK to be used for this SonarQube analysis

(Inherit From Job)

Path to project properties ?

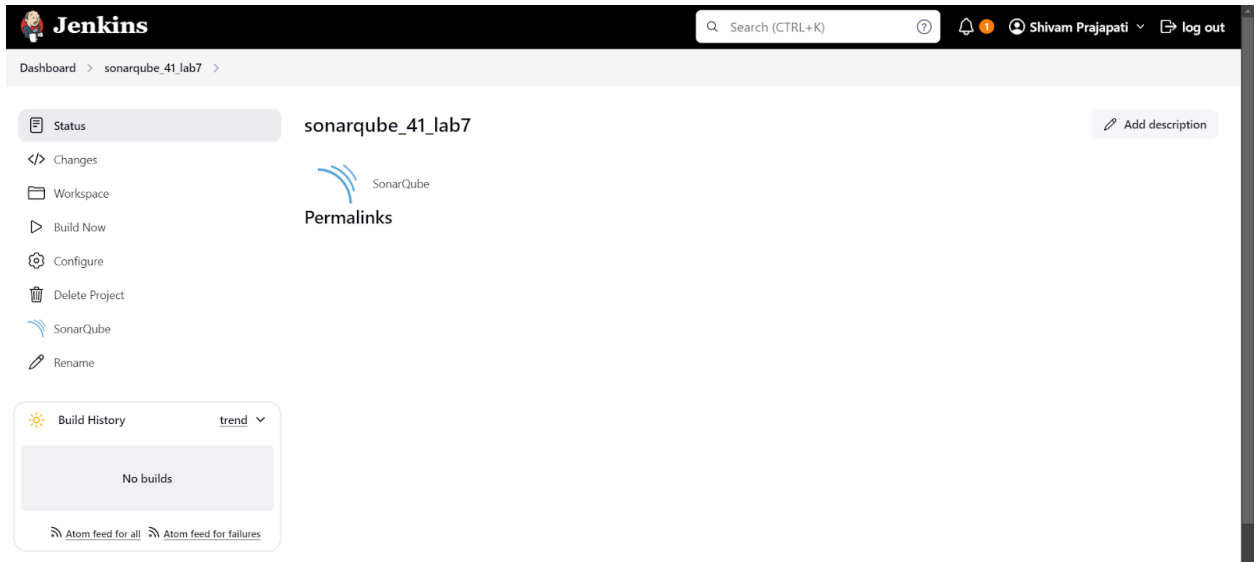
Analysis properties ?

sonar.projectKey=Shivam_41
sonar.login=admin
sonar.password=Shivam2@
sonar.host.url=http://localhost:9000
sonar.sources=.

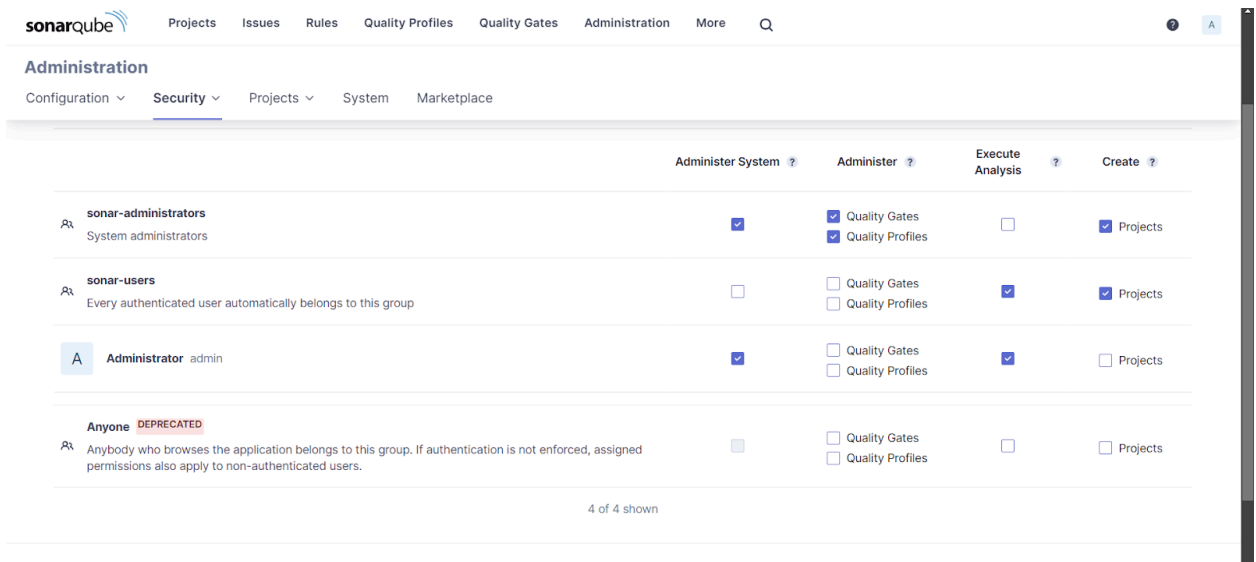
Additional arguments ?

JVM Options ?

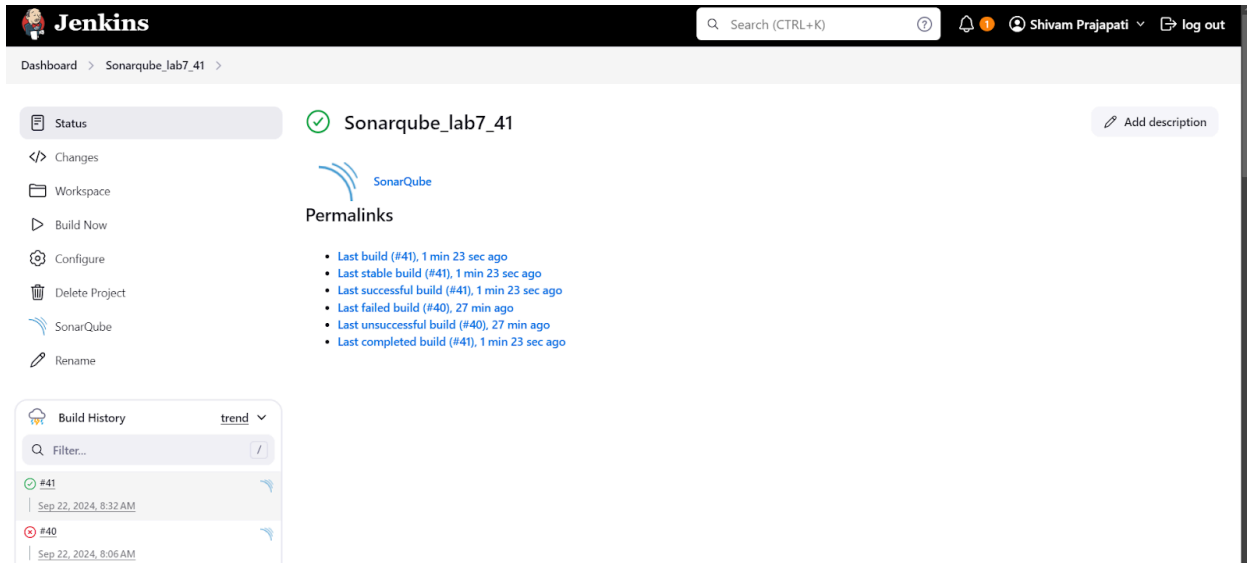
Save Apply



Step 13: Now, you need to grant the local user (here admin user) permissions to Execute the Analysis stage on SonarQube. For this, go to http://localhost:<port_number>/admin/permissions and check the 'Execute Analysis' checkbox under Administrator.

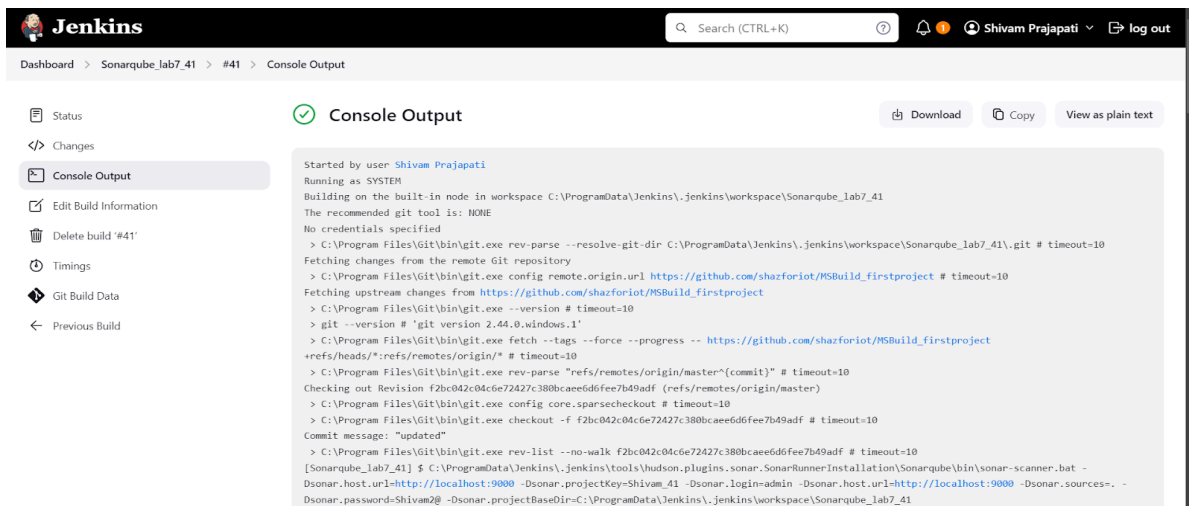


Step 14: Go back to jenkins. Go to the job you had just built and click on Build Now.



The screenshot shows the Jenkins web interface for the job 'Sonarqube_lab7_41'. The top navigation bar includes the Jenkins logo, a search bar, and user information. The left sidebar contains a menu with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, SonarQube, and Rename. The main content area displays the job's status as 'Success' with a green checkmark. Below this, there are 'Permalinks' for various build types: Last build (#41), Last stable build (#41), Last successful build (#41), Last failed build (#40), Last unsuccessful build (#40), and Last completed build (#41). A 'Build History' table is visible at the bottom left, showing the last two builds: #41 (successful) and #40 (failed).

Check the Console Output



The screenshot shows the 'Console Output' page for the job 'Sonarqube_lab7_41'. The left sidebar is updated to include 'Console Output' as the selected option. The main content area displays the console output, which shows the build process starting with the user 'Shivam Prajapati'. The output includes commands for fetching changes from a remote Git repository, checking out a specific revision, and running the SonarScanner. The build is successful, and the console output is displayed in a scrollable area.

```
Dashboard > Sonarqube_lab7_41 > #41 > Console Output
08:32:55.264 INFO Sensor Analysis Warnings import [csharp] (done) | time=4ms
08:32:55.264 INFO Sensor C# File Caching Sensor [csharp]
08:32:55.264 WARN Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
08:32:55.264 INFO Sensor C# File Caching Sensor [csharp] (done) | time=0ms
08:32:55.264 INFO Sensor Zero Coverage Sensor
08:32:55.279 INFO Sensor Zero Coverage Sensor (done) | time=15ms
08:32:55.288 INFO SCM Publisher SCM provider for this project is: git
08:32:55.290 INFO SCM Publisher 4 source files to be analyzed
08:32:55.515 INFO SCM Publisher 4/4 source files have been analyzed (done) | time=225ms
08:32:55.522 INFO CPD Executor Calculating CPD for 0 files
08:32:55.523 INFO CPD Executor CPD calculation finished (done) | time=0ms
08:32:55.524 INFO SCM revision ID 'f2bc942c04c6e72427c380bcaee6d6fee7b49adf'
08:32:55.809 INFO Analysis report generated in 125ms, dir size=201.0 kB
08:32:55.858 INFO Analysis report compressed in 40ms, zip size=22.5 kB
08:32:56.061 INFO Analysis report uploaded in 201ms
08:32:56.062 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=Shivam_41
08:32:56.063 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
08:32:56.063 INFO More about the report processing at http://localhost:9000/api/ce/task?id=94824f79-1689-41ea-99d7-abfee5815e63
08:32:56.077 INFO Analysis total time: 28.732 s
08:32:56.078 INFO SonarScanner Engine completed successfully
08:32:56.158 INFO EXECUTION SUCCESS
08:32:56.158 INFO Total time: 38.798s
Finished: SUCCESS
```

REST API Jenkins 2.462.1

Step 15: Once the build is complete, go back to SonarQube and check the project linked

The image shows the SonarQube web interface for a project named 'Shivam_41'. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The project overview page displays a 'Passed' status with a green checkmark. A warning message states: 'The last analysis has warnings. See details'. Below this, there are tabs for 'New Code' and 'Overall Code'. The 'Overall Code' tab is active, showing a grid of quality metrics: Security (0 Open issues, A grade), Reliability (0 Open issues, A grade), Maintainability (0 Open issues, A grade), Accepted issues (0, Valid issues that were not fixed), Coverage (0 lines to cover), Duplications (0.0%, On 86 lines), and Security Hotspots (0, A grade). The bottom of the page shows the last analysis was completed 4 minutes ago.

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 the SonarQube project displays that the code has no errors.