

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

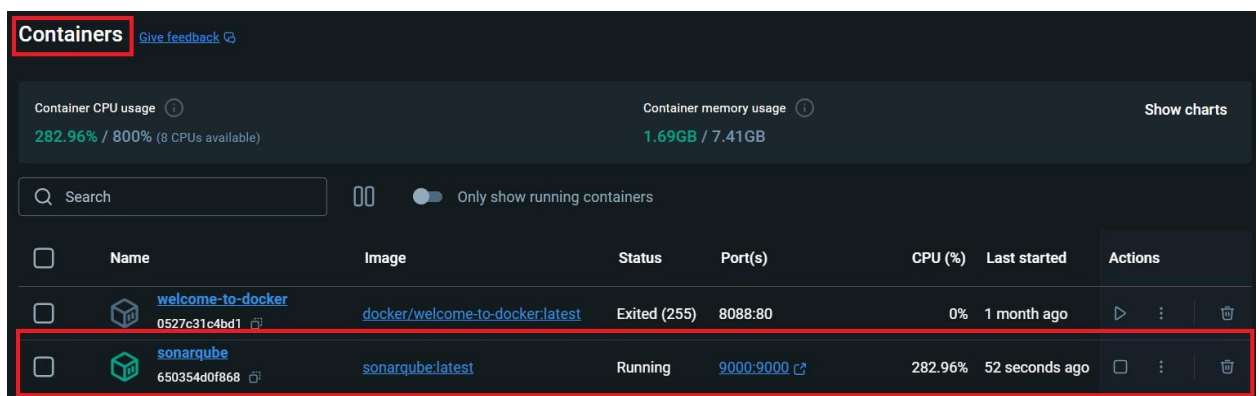
What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview sonarqube
```

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:

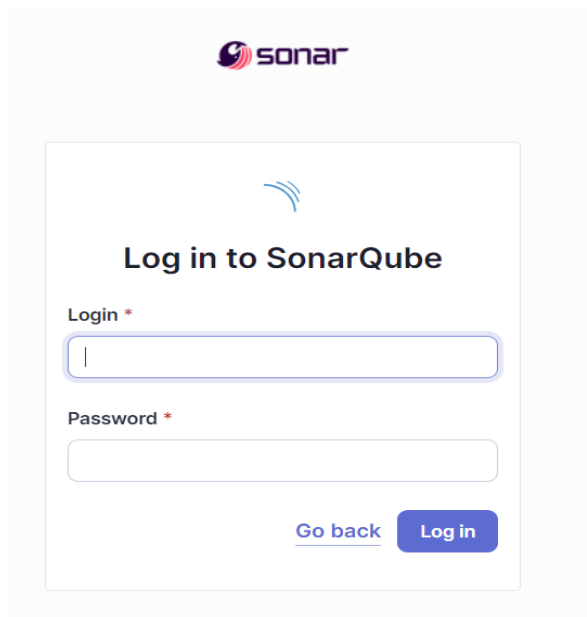
Step1:The command `docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest` starts SonarQube in the background on port 9000 using Docker, allowing you to access it at <http://localhost:9000>


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




	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	welcome-to-docker 0527c31c4bd1	docker/welcome-to-docker:latest	Exited (255)	8088:80	0%	1 month ago	
<input type="checkbox"/>	sonarqube 650354d0f868	sonarqube:latest	Running	9000:9000	282.96%	52 seconds ago	<input type="checkbox"/>

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







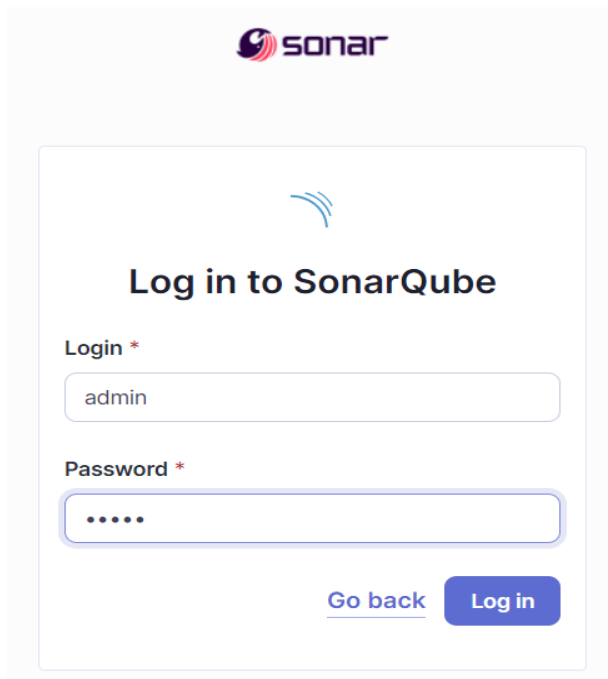
Log in to SonarQube

Login *

Password *

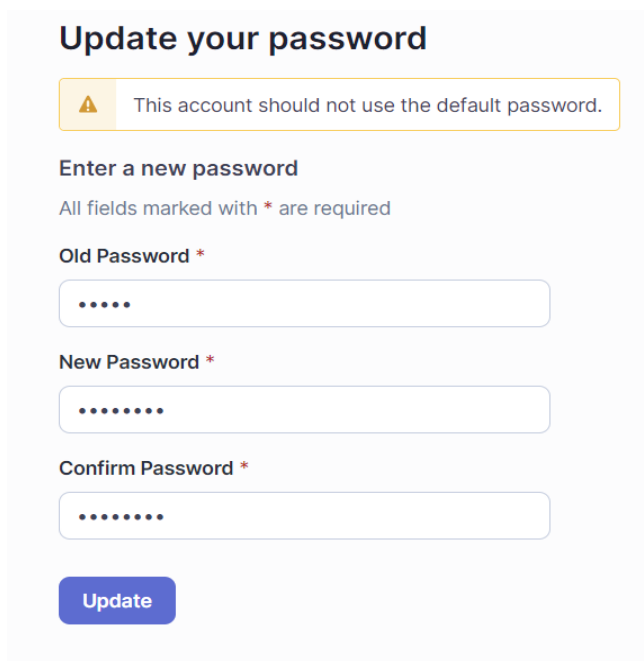
[Go back](#) Log in

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 the 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 masked characters. At the bottom, there are two buttons: "Go back" (a link) and "Log in" (a blue button).

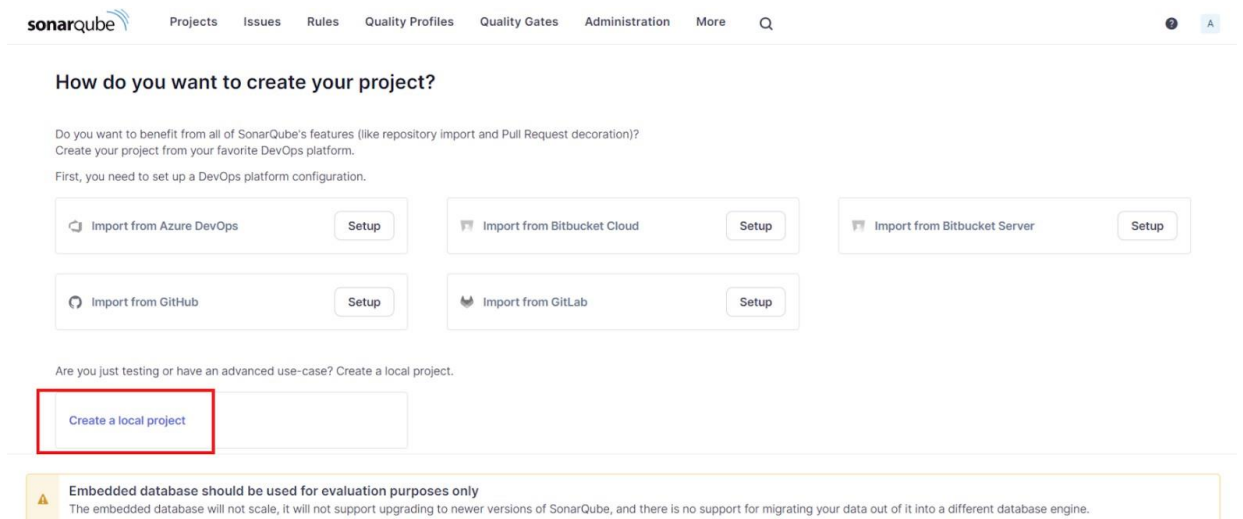
Click on Log in



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 "Enter a new password" is displayed. There is a note: "All fields marked with * are required". There are three input fields: "Old Password *" with masked characters, "New Password *" with masked characters, and "Confirm Password *" with masked characters. At the bottom, there is a blue button labeled "Update".

Click on Update .

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



sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More Q

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)?
Create your project from your favorite DevOps platform.
First, you need to set up a DevOps platform configuration.

Import from Azure DevOps Setup

Import from Bitbucket Cloud Setup

Import from Bitbucket Server Setup

Import from GitHub Setup

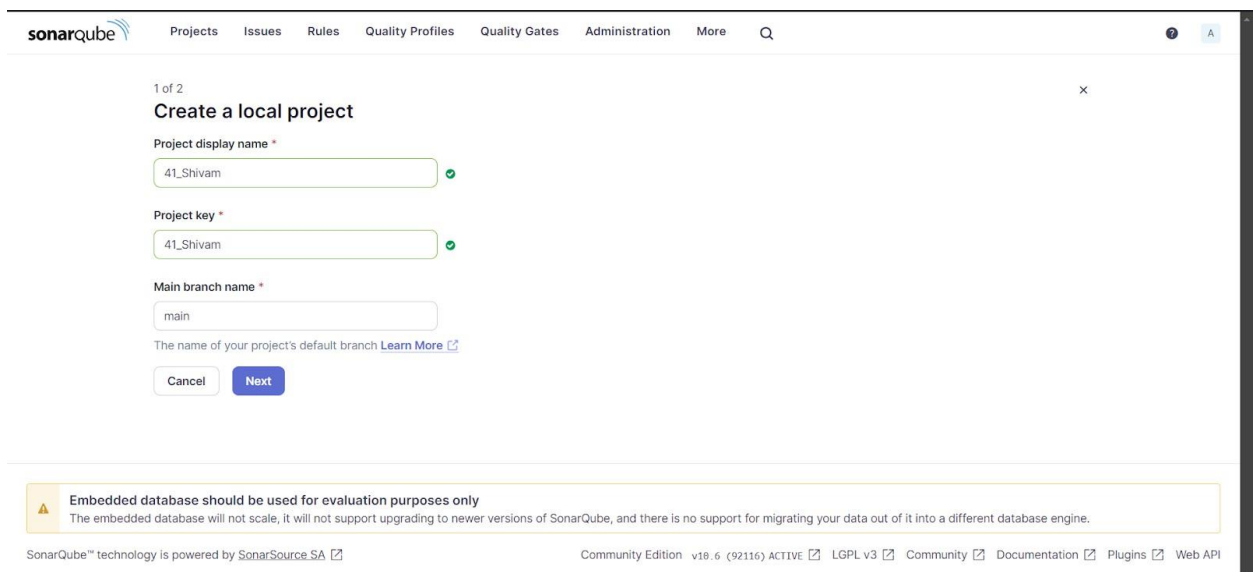
Import from GitLab Setup

Are you just testing or have an advanced use-case? Create a local project.

Create a local 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.

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



sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More Q

1 of 2

Create a local project

Project display name *

41_Shivam

Project key *

41_Shivam

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.

SonarQube™ technology is powered by [SonarSource SA](#)

Community Edition v10.6 (92116) ACTIVE LGPL v3 Community Documentation Plugins 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.

The screenshot shows the SonarQube interface with the 'Set up project for Clean as You Code' configuration page. The page title is 'Set up project for Clean as You Code' and it includes a sub-header 'Choose the baseline for new code for this project'. There are two main radio button options: 'Use the global setting' (selected) and 'Define a specific setting for this project'. Under 'Define a specific setting for this project', there are three sub-options: 'Previous version', 'Number of days', and 'Reference branch'. The 'Previous version' option is selected. The page also includes a 'Back' button and a 'Create project' button.

Scroll Down

This screenshot shows the same SonarQube configuration page as the previous one, but with the 'Create project' button highlighted by a red rectangle. The 'Create project' button is located at the bottom of the configuration section. Below the configuration section, there is a warning message: '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 this is a 'Build Queue' section showing 'No builds in the queue.' and a 'Build Executor Status' section showing two executors: 'Shivam' and 'Slave1', both offline. The main area displays a table of build history with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The table lists several builds, including '41_SonarQube', 'Devops pipeline1', 'DevOps_Pipeline', 'Pipeline_DevOps', 'SonarQube_41', 'sonarqube_41_lab7', 'sonarqube_41_lab8', 'Sonarqube_lab7_41', and 'Sonarqube_lab8_41'. A search bar and a 'log out' button are visible at the top right.

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 'New version of Jenkins (2.462.2) is available for download (changelog)' message with a 'Downgrade to 2.452.3' button. Below this is a 'System Configuration' section with four tabs: 'System', 'Tools', 'Nodes', and 'Plugins'. The 'Plugins' tab is highlighted with a red box. The 'Plugins' section shows a list of available plugins, including 'SonarQube Scanner'. The 'Appearance' section is also visible at the bottom right.

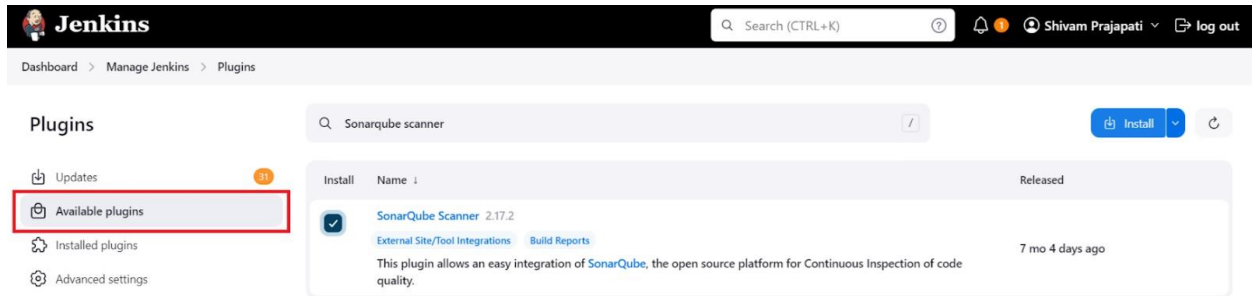
Manage Jenkins

New version of Jenkins (2.462.2) is available for [download](#) ([changelog](#)). [Or Upgrade Automatically](#)

[Restore the previous version of Jenkins](#) [Downgrade to 2.452.3](#)

System Configuration

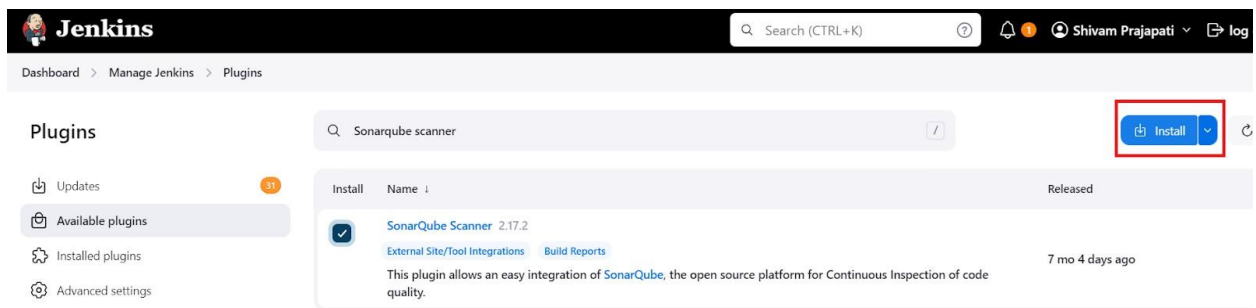
- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Appearance**: Configure the look and feel of Jenkins.



The screenshot shows the Jenkins 'Plugins' page. The left sidebar has a red box around 'Available plugins'. The main area has a search bar with 'Sonarqube scanner' and an 'Install' button. Below the search bar is a table with columns 'Install', 'Name', and 'Released'. The table lists 'SonarQube Scanner 2.17.2' with a checkmark in the 'Install' column and a description of the plugin's functionality.

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

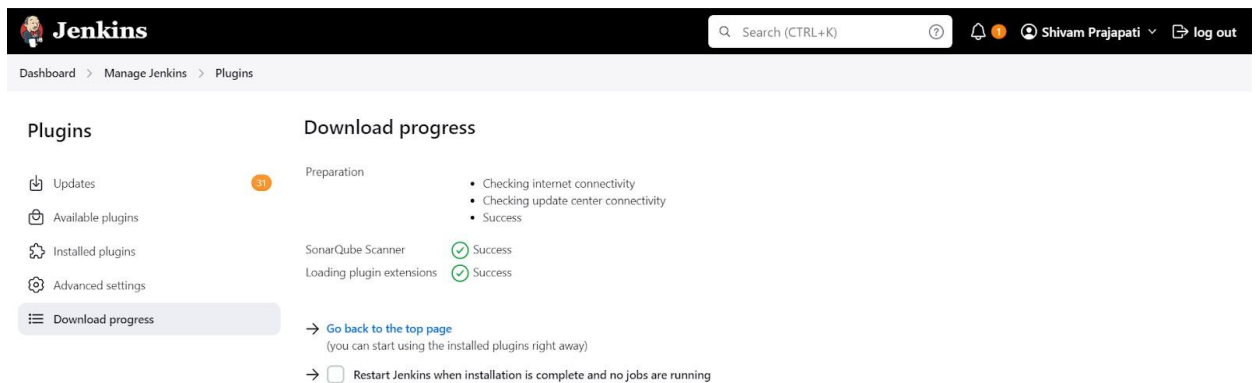
Click on Available Plugins.



The screenshot shows the Jenkins 'Plugins' page. The left sidebar has a red box around 'Available plugins'. The main area has a search bar with 'Sonarqube scanner' and an 'Install' button. Below the search bar is a table with columns 'Install', 'Name', and 'Released'. The table lists 'SonarQube Scanner 2.17.2' with a checkmark in the 'Install' column and a description of the plugin's functionality.

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

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



The screenshot shows the Jenkins 'Download progress' page. The left sidebar has a red box around 'Download progress'. The main area shows the progress of the installation. The 'Preparation' step is completed. The 'SonarQube Scanner' and 'Loading plugin extensions' steps are also completed with green checkmarks. Below the progress bar, there are links to 'Go back to the top page' and a checkbox to 'Restart Jenkins when installation is complete and no jobs are running'.

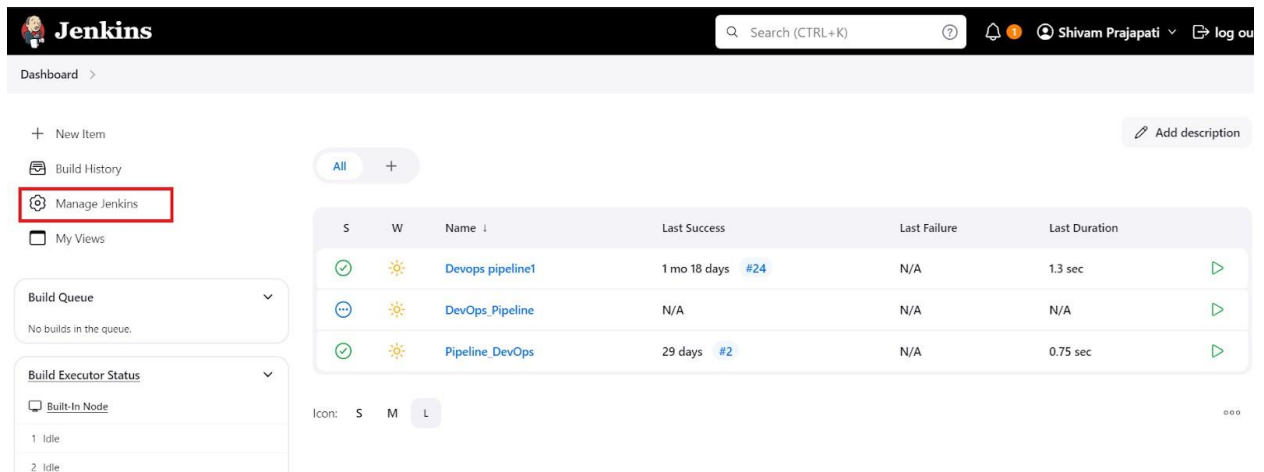
Step	Status
Preparation	Success
SonarQube Scanner	Success
Loading plugin extensions	Success

→ [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.

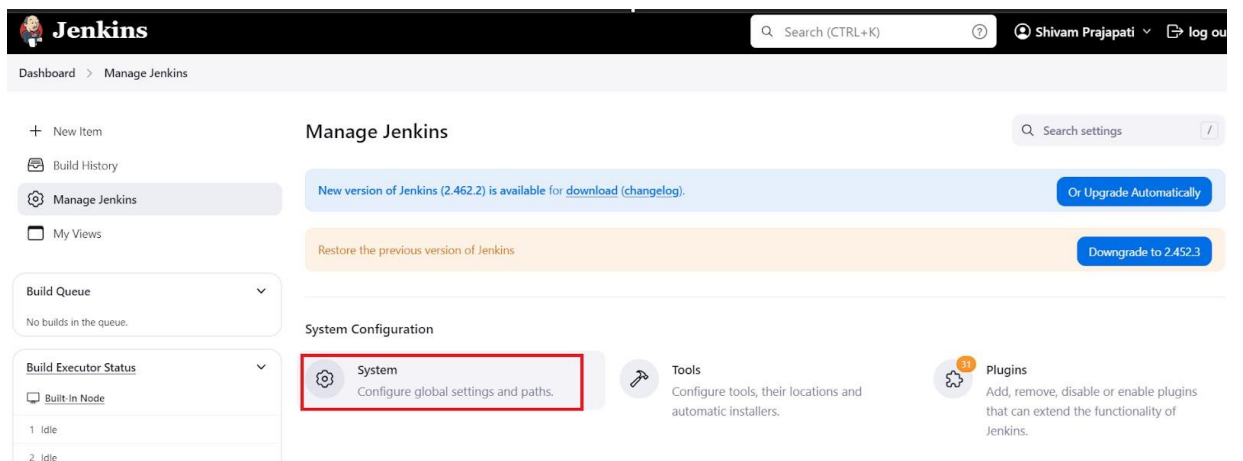
Step 9: In Jenkins, go to **Manage Jenkins** → **System**, then find **SonarQube** servers. Add a new server, and if required, include the authentication token for secure access



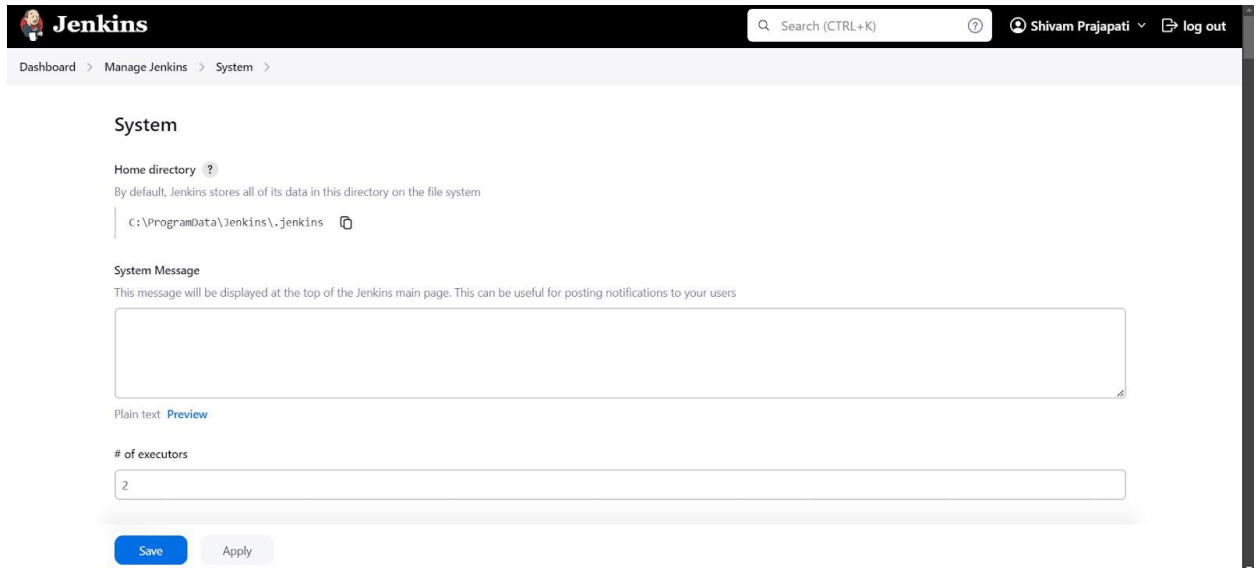
The screenshot shows the Jenkins Dashboard. The left sidebar contains navigation links: New Item, Build History, Manage Jenkins (highlighted with a red box), and My Views. Below these are sections for Build Queue (No builds in the queue) and Build Executor Status (2 Idle). The main content area displays a table of builds with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The table lists three builds: Devops pipeline1, DevOps_Pipeline, and Pipeline_DevOps. Below the table, there are icons for S, M, and L.

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

Go to system



The screenshot shows the Jenkins Manage Jenkins page. 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 (2 Idle). The main content area displays the Manage Jenkins page with a search bar and a list of settings categories: System (highlighted with a red box), Tools, and Plugins. The System category is described as 'Configure global settings and paths.' The Tools category is described as 'Configure tools, their locations and automatic installers.' The Plugins category is described as 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.'



The screenshot shows the Jenkins 'System' configuration page. At the top, there's a navigation bar with 'Dashboard', 'Manage Jenkins', and 'System'. The 'System' section includes a 'Home directory' field with a help icon, a description, and a text input field containing 'C:\ProgramData\jenkins\jenkins'. Below this is a 'System Message' section with a text area and a 'Preview' link. The '# of executors' field contains the value '2'. At the bottom, there are 'Save' and 'Apply' buttons.

System

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

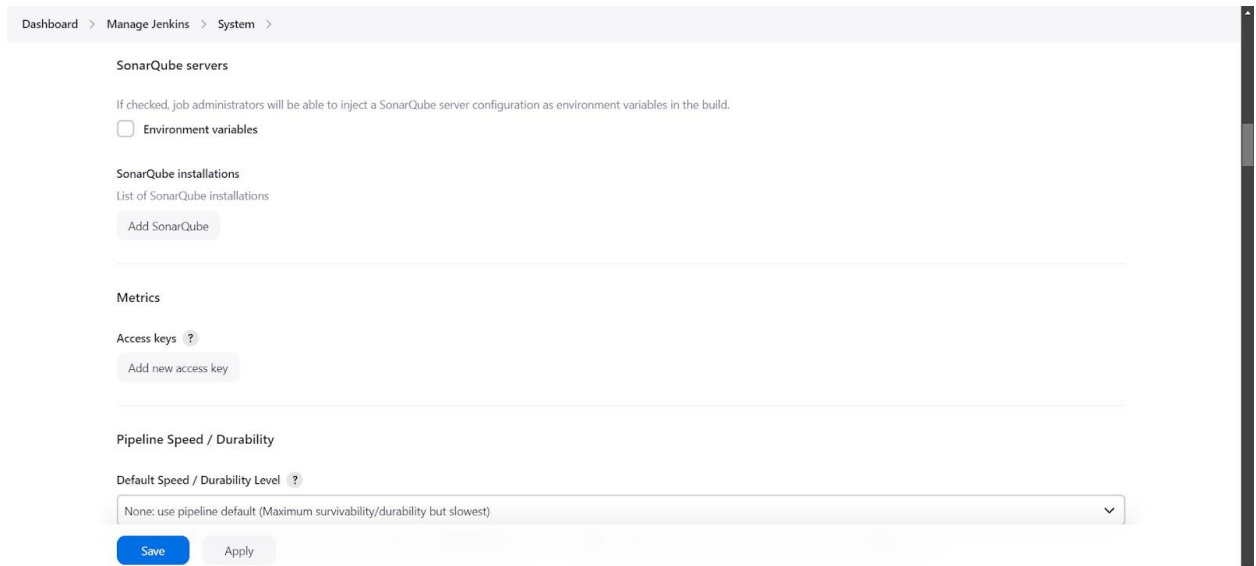
System Message
This message will be displayed at the top of the Jenkins main page. This can be useful for posting notifications to your users

Plain text [Preview](#)

of executors
2

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

Scroll Down



The screenshot shows the Jenkins 'System' configuration page scrolled down. It displays the 'SonarQube servers' section with a checkbox for 'Environment variables'. Below this is the 'SonarQube installations' section with a list of installations and an 'Add SonarQube' button. The 'Metrics' section includes an 'Access keys' field with a help icon and an 'Add new access key' button. The 'Pipeline Speed / Durability' section shows 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.

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.

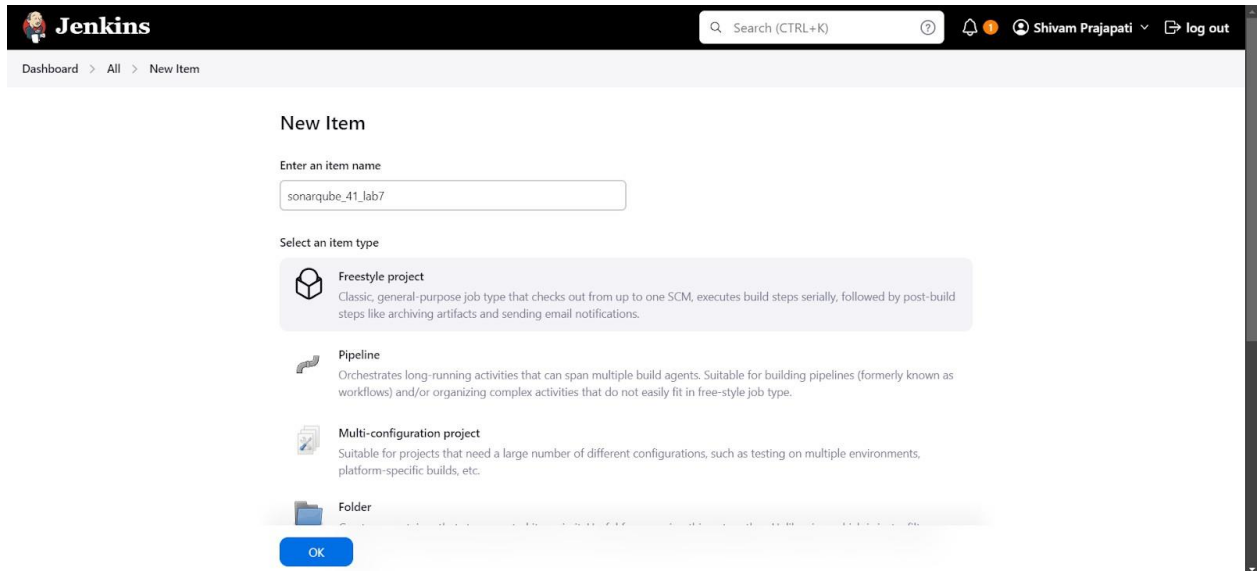
- none - ▼

[+ Add](#)

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

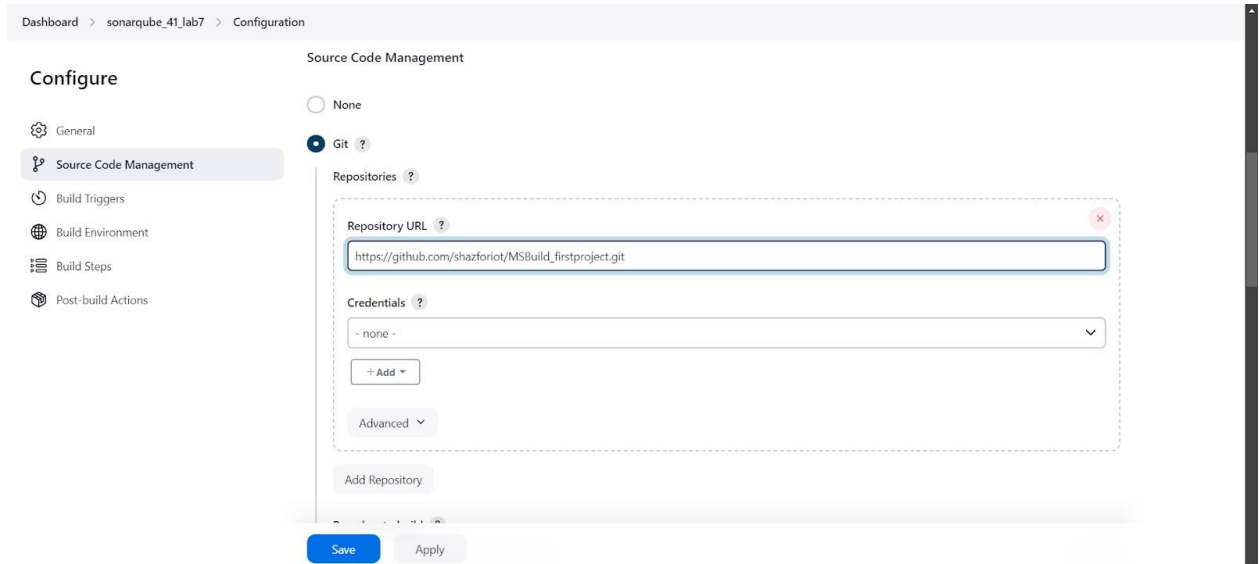
Click on save

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 ?

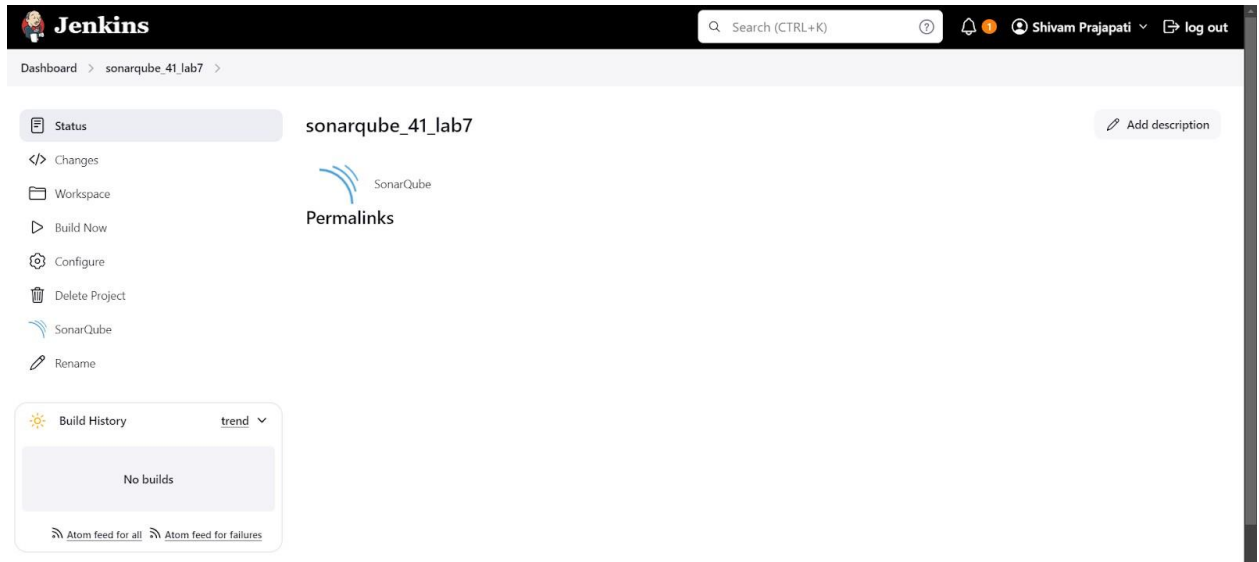
sonar.projectKey=Shivam_41
sonar.login=admin
sonar.password=Shivam2@
sonar.host.url=http://localhost:9000
sonar.sources=.

Additional arguments ?

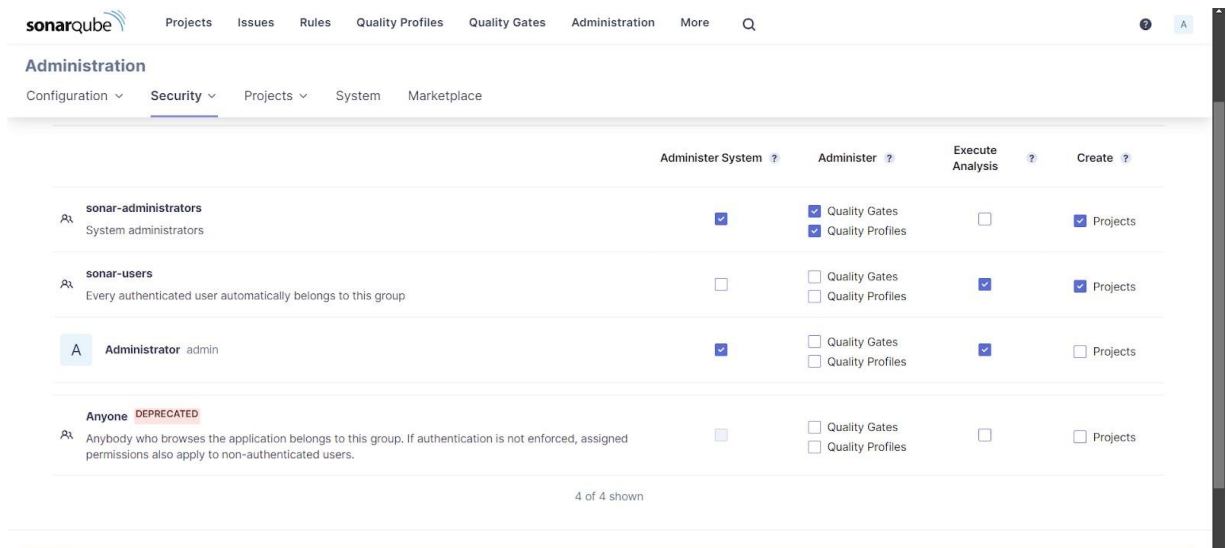
JVM Options ?

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 dashboard for the job 'Sonarqube_lab7_41'. The job status is 'Success' (green checkmark). The left sidebar contains a list of actions: Status, Changes, Workspace, Build Now, Configure, Delete Project, SonarQube, and Rename. The main area displays the job name, a SonarQube logo, and a list of permalinks for various builds. Below this, a 'Build History' section shows a list of builds with filters and a trend graph.

Permalinks

- Last build (#41), 1 min 23 sec ago
- Last stable build (#41), 1 min 23 sec ago
- Last successful build (#41), 1 min 23 sec ago
- Last failed build (#40), 27 min ago
- Last unsuccessful build (#40), 27 min ago
- Last completed build (#41), 1 min 23 sec ago

Build History

Build	Time
#41	Sep 22, 2024, 8:32 AM
#40	Sep 22, 2024, 8:06 AM

Check the Console Output

The screenshot shows the Jenkins console output for the job 'Sonarqube_lab7_41'. The console output is displayed in a text area with a 'Download' button. The output shows the build process, including fetching changes from the remote Git repository, checking out the revision, and running the SonarQube scanner.

```
Started by user Shivam Prajapati
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\Sonarqube_lab7_41
The recommended git tool is: NONE
No credentials specified
> C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\Sonarqube_lab7_41\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> git --version # 'git version 2.44.0.windows.1'
> C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject
refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse 'refs/remotes/origin/master^{commit}' # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcae6d6fee7b49adf (refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
Commit message: "updated"
> C:\Program Files\Git\bin\git.exe rev-list --no-walk f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
[Sonarqube_lab7_41] $ C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\Sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=Shivam_41 -Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -
Dsonar.password=Shivam2@ -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\Sonarqube_lab7_41
```

```
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 'f2bc042c04cbe72427c380bcaee6d6fee7b49adf'
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 screenshot 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's main status is 'Passed', indicated by a large green checkmark. Below this, a warning message states 'The last analysis has warnings. See details'. The dashboard is divided into several sections: 'New Code' and 'Overall Code' tabs, 'Security' (0 Open issues), 'Reliability' (0 Open issues), 'Maintainability' (0 Open issues), 'Accepted issues' (0 Valid issues that were not fixed), 'Coverage' (On 0 lines to cover), 'Duplications' (0.0% On 86 lines), and 'Security Hotspots' (0). Each section has a corresponding 'A' grade indicator.

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.