

Experiment 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.

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. The main area displays a table of build history. The table has columns for status (S), weather icon (W), name, last success, last failure, and last duration. There are four rows of builds: 'sahil 7', 'Sahil exp6', 'SahilExp6', and 'sahiljob'. The 'sahiljob' build is currently in progress, indicated by a red lightning bolt icon.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	sahil 7	24 days #2	N/A	96 ms
✓	☀	Sahil exp6	24 days #3	N/A	1 sec
⚡	☀	SahilExp6	N/A	N/A	N/A
⚡	☁	sahiljob	N/A	24 days #1	1.5 sec

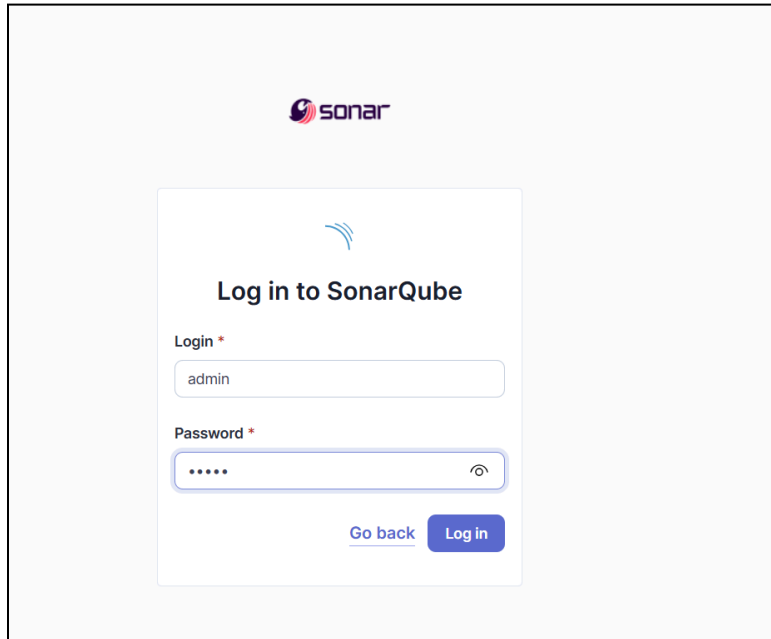
- 2) Run SonarQube in a Docker container using this command -

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

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

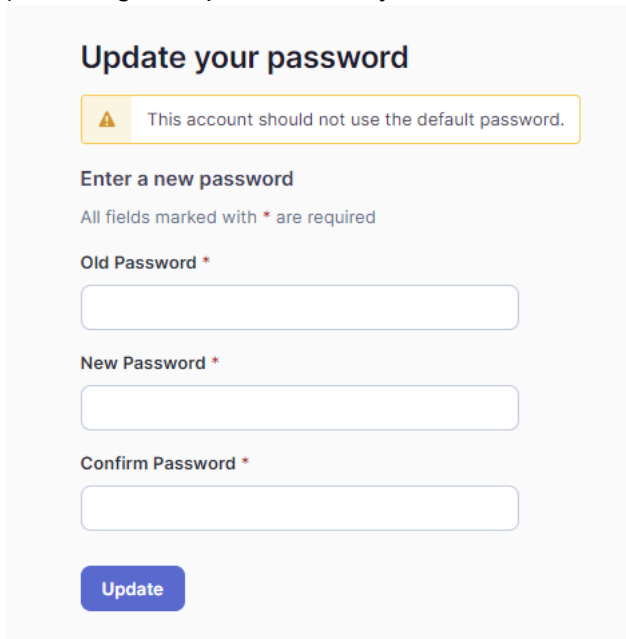
```
C:\Users\Lenovo>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
47f6db8dbf2ed99dbe304bc0ebdf47b9d4144c4e4add42055ba44ce231058272
```

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




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 ".....". To the right of the password field is an eye icon. At the bottom, there are two buttons: "Go back" (a link) and "Log in" (a button).

4) Login to SonarQube using username - *admin* and password - *admin*.
(do change the password as you cannot use the default one)




The image shows the SonarQube password update page. At the top, the text "Update your password" is displayed. Below it, there is a warning message in a yellow box: "This account should not use the default password." Underneath, the text "Enter a new password" is shown, followed by a note: "All fields marked with * are required". There are three input fields: "Old Password *" (empty), "New Password *" (empty), and "Confirm Password *" (empty). At the bottom, there is a blue button labeled "Update".


ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMoreQ?


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](#)

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


1 of 2

Create a local project

Project display name *

Project key *

Main branch name *

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

Cancel

Next

6) Setup the project and come back to Jenkins Dashboard.

The screenshot shows the Jenkins configuration page for setting a baseline for new code. The page has a navigation bar at the top with links: Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. Below the navigation bar, the title is "Choose the baseline for new code for this project". There are three main radio button options:

- Use the global setting** (selected):
 - 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** (unselected):
 - 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 paid. Recommended for projects following continuous delivery.
 - Reference branch**: Choose a branch as the baseline for the new code. Recommended for projects using feature branches.

At the bottom, there are two buttons: "Back" and "Create project".

7) Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.

The screenshot shows the Jenkins 'Manage Jenkins' Plugins page. The top navigation bar includes the Jenkins logo, a search bar, and user information. The left sidebar shows the 'Plugins' section with a search bar and a list of plugins. The main content area displays a table of plugins related to SonarQube.

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 8 days ago
<input type="checkbox"/>	Sonar Gerrit 388.v9b_f1cb_e42306 External Site/Tool Integrations This plugin allows to submit issues from SonarQube to Gerrit as comments directly.	3 mo 22 days ago
<input type="checkbox"/>	SonarQube Generic Coverage 1.0 TODO	5 yr 1 mo ago

8) 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>

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_exp7

Server URL

Default is http://localhost:9000

http://localhost:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

+ Add ▾

Advanced ▾

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

Dashboard > Manage Jenkins > Tools

Dashboard > Manage Jenkins > Tools

Gradle installations

Add Gradle

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild


SonarQube Scanner installations

Add SonarQube Scanner

Ant installations

Add Ant

Maven installations

Maven installations ▾  Edited

Save Apply

Click on **Add SonarQube Scanner** .

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

Check the “Install automatically” option.

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

SonarQube Scanner installations

Add SonarQube Scanner

☰ SonarQube Scanner

Name

sonarqube_scanner_exp7

☒ Install automatically ?

☰ Install from Maven Central

Version

SonarQube Scanner 6.2.0.4584

Add Installer ▾

Add SonarQube Scanner

✓ Saved


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

Dashboard > All >

Enter an item name

exp7

= Required field



Freestyle project
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.

11) 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.

Dashboard > exp7 > Configuration

Configure

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

Source Code Management

☐ None

☒ **Git** ?

Repositories ?

Repository URL ?

Credentials ?

- none -

+ Add

Advanced

Add Repository

Branches to build ?

Branch Snarifier (blank for 'any') ?

Save Apply

12) 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.

Dashboard > exp7 > Configuration

Configure

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

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
- SonarScanner for MSBuild - End Analysis

Add build step ^

Post-build Actions

Add post-build action

Save Apply

Following window will open -

The screenshot shows the 'Configure' window for 'Execute SonarQube Scanner'. The left sidebar lists various configuration categories: General, Source Code Management, Build Triggers, Build Environment, Build Steps (selected), and Post-build Actions. The main area contains the following fields:

- JDK**: A dropdown menu with '(Inherit From Job)' selected.
- Path to project properties**: An empty text input field.
- Analysis properties**: A large text area containing the following properties:


```
sonar.projectKey = exp7
sonar.login = admin
sonar.password = 2923
sonar.host.url = http://localhost:9000
sonar.sources = .
```
- Additional arguments**: An empty text input field.
- JVM Options**: An empty text input field.

At the bottom left, there is an 'Add build step' button.

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.

The screenshot shows the 'Project Information' page in SonarQube. The left sidebar contains the 'About this Project' section with the following details:

- Quality Gate used**: (Default) Sonar way
- Project Key**: exp7 (with a copy icon)
- Visibility**: Public
- Description**: No description added for this project.
- Tags**: No tags

The right sidebar contains the 'Notifications' section with a message: 'A notification is never sent to the author of the event.' Below this is a 'Send me an email for:' section with checkboxes for: Background tasks in failure, Changes in issues/hotspots assigned to me, Quality gate changes, Issues resolved as false positive or accepted, New issues, and My new issues. At the bottom, there is a 'Badges' section with a note: 'Show the status of your project machine on your README or website. Get your badge.'

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 =

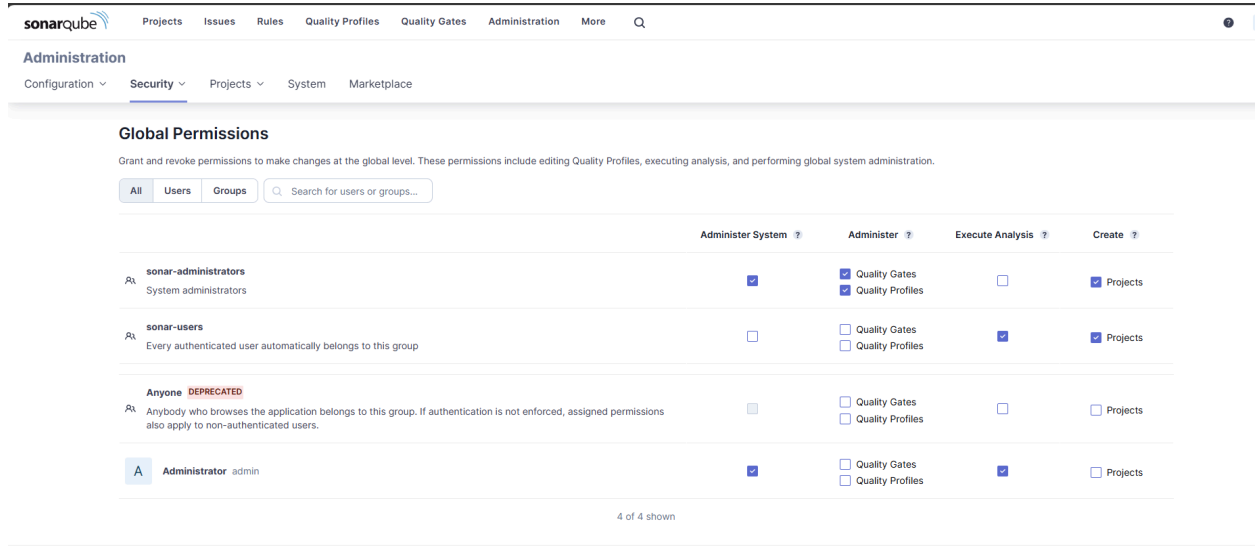
This screenshot is identical to the one above, but the 'Analysis properties' text area is now filled with the following code:

```
sonar.projectKey = exp7
sonar.login = admin
sonar.password = 2923
sonar.host.url = http://localhost:9000
sonar.sources = .
```

Apply and save.

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

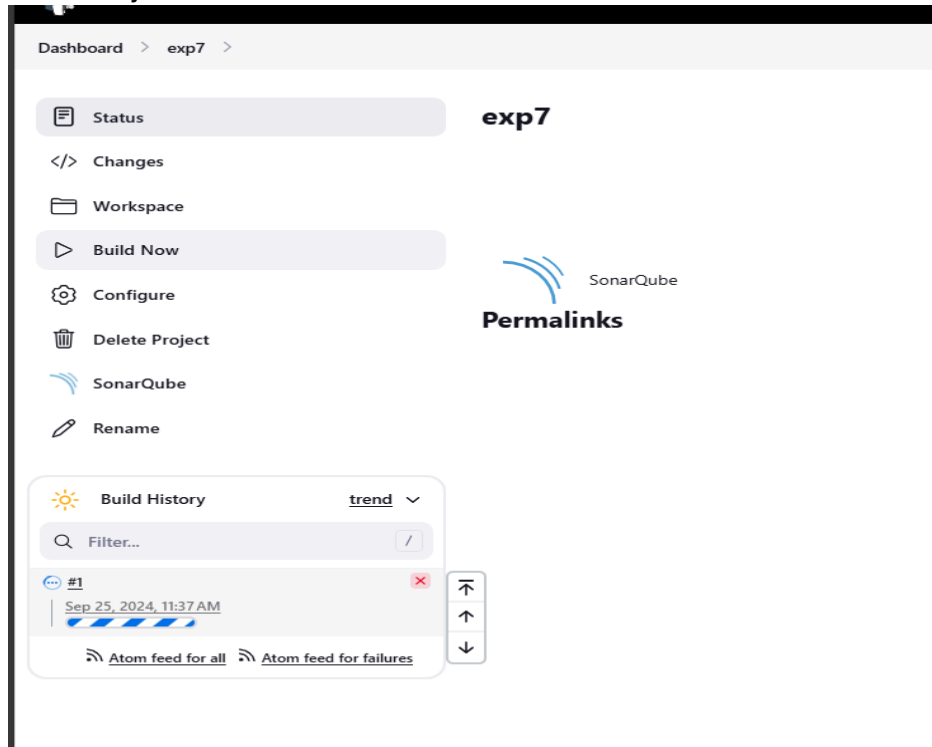
See the administrator below and check the boxes as checked below..



The screenshot shows the SonarQube Administration interface, specifically the 'Global Permissions' section under 'Security'. The page title is 'Administration' with a sub-header 'Global Permissions'. Below the title, there is a description: 'Grant and revoke permissions to make changes at the global level. These permissions include editing Quality Profiles, executing analysis, and performing global system administration.' There are three tabs: 'All', 'Users', and 'Groups'. A search bar is present with the text 'Search for users or groups...'. The main content is a table with columns: 'User/Group', 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The table lists four entries: 'sonar-administrators' (System administrators), 'sonar-users' (Every authenticated user automatically belongs to this group), 'Anyone DEPRECATED' (Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users.), and 'Administrator admin'. The 'Administrator admin' entry has checkboxes checked for 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The 'sonar-administrators' entry has checkboxes checked for 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The 'sonar-users' entry has checkboxes checked for 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The 'Anyone DEPRECATED' entry has checkboxes checked for 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The 'Administrator admin' entry has checkboxes checked for 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The table is followed by a footer '4 of 4 shown'.

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

12. Go to jenkins and click build:



The screenshot shows the Jenkins Dashboard for a project named 'exp7'. The dashboard has a left sidebar with a menu: 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', 'SonarQube', and 'Rename'. The main area displays the 'exp7' project name, the SonarQube logo, and the text 'Permalinks'. Below this, there is a 'Build History' section with a 'trend' dropdown and a 'Filter...' input. The build history shows a single build with the status '#1' and the timestamp 'Sep 25, 2024, 11:37 AM'. The build is represented by a blue bar with a red 'X' icon. Below the build history, there are two RSS feed links: 'Atom feed for all' and 'Atom feed for failures'. A vertical arrow icon is visible on the right side of the build history section.

```
Dashboard > exp7 > #5 > Console Output

Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#5'
Timings
Git Build Data
Previous Build

Console Output

Started by user Shubham Jha
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\jenkins\jenkins\workspace\exp7
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\jenkins\jenkins\workspace\exp7\.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.exe --version # 'git version 2.45.0.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 f2bc042c046e72427c380bccee6dfee7b49adf (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f f2bc042c046e72427c380bccee6dfee7b49adf # timeout=10
Commit message: "updated"
> git.exe rev-list --no-walk f2bc042c046e72427c380bccee6dfee7b49adf # timeout=10
[exp7] $ C:\ProgramData\jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube_scanner_exp7\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=exp7
-Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -Dsonar.password=2923 -Dsonar.projectBaseDir=C:\ProgramData\jenkins\jenkins\workspace\exp7
13:47:35.366 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
13:47:35.382 INFO Scanner configuration file: C:\ProgramData\jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube_scanner_exp7\bin\..\conf\sonar-scanner.properties
13:47:35.382 INFO Project root configuration file: NONE
13:47:35.403 INFO SonarScanner CLI 6.2.0.4584
13:47:35.403 INFO Java 21.0.4 Oracle Corporation (64-bit)
13:47:35.413 INFO Windows 11 10.0 amd64
13:47:35.429 INFO User cache: C:\Windows\system32\config\systemprofile\.sonar\cache
13:47:37.015 INFO JRE provisioning: os[jindows], arch[amd64]
13:47:47.897 INFO Communicating with SonarQube Server 10.6.0.92116
13:47:47.665 INFO Starting SonarScanner Engine...
```

Conclusion:

In this project, we 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 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.