



Vivekanand Education Society's

Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra

Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

A.Y. 2024-25

Advance DevOps Lab

Experiment 07

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

| | |
|-----------|--|
| Roll No. | 44 |
| Name | Ganesh Sanjay Pandhre |
| Class | D15B |
| Subject | Advance DevOps Lab |
| LO Mapped | LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements. LO4: To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Techniques. |
| Grade: | |

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

THEORY :

What is SAST?

Static Application Security Testing (SAST) is a method that analyzes source code to find security weaknesses before the code is compiled. This is often called white box testing.

What Problems Does SAST Solve?

SAST occurs early in the Software Development Life Cycle (SDLC) and does not require a working application. It helps developers find and fix vulnerabilities during development, preventing issues from reaching the final product. SAST tools provide real-time feedback, allowing developers to address problems before moving on. They also give visual representations of issues and highlight the exact locations of vulnerabilities, guiding developers on how to fix them without needing extensive security knowledge. It's crucial to run SAST tools regularly, such as during daily builds or code releases.

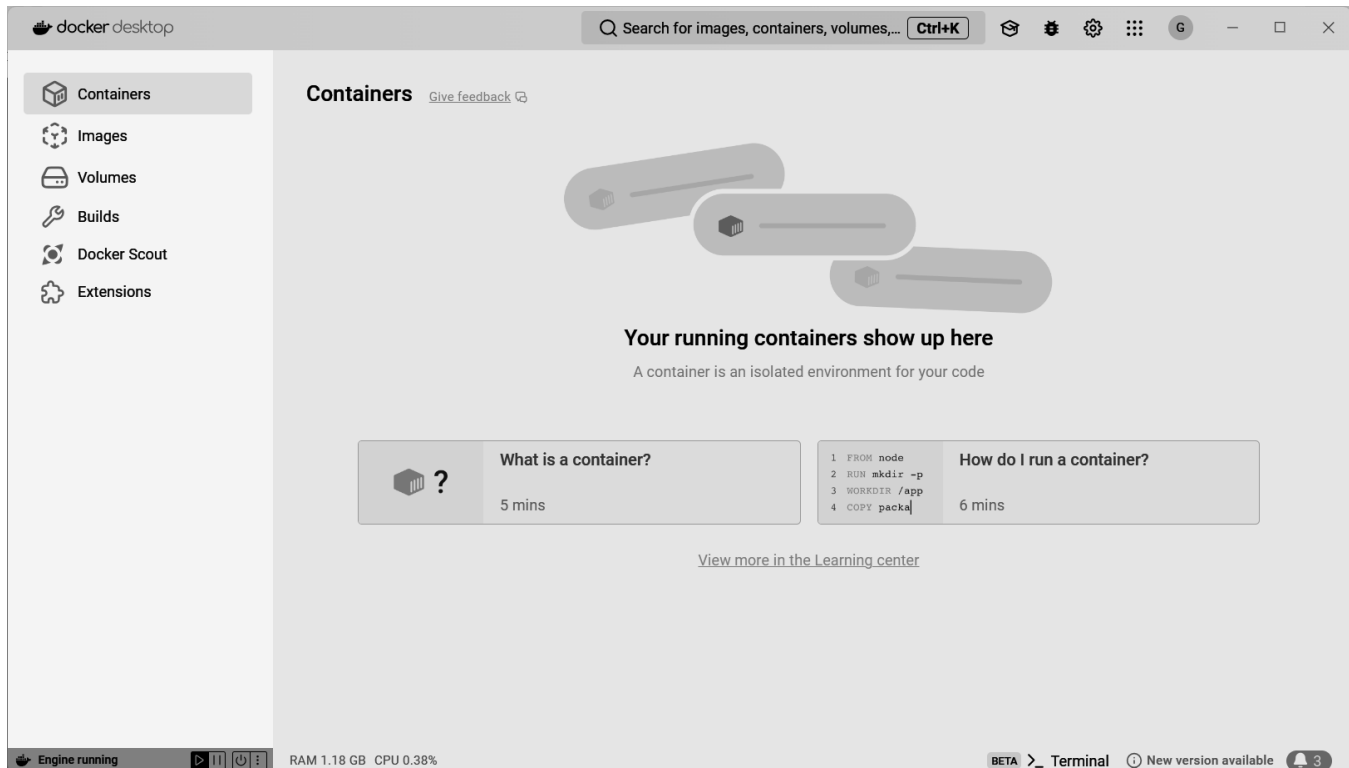
Why is SAST Important?

There are more developers than security staff, making it hard to review all code manually. SAST tools can analyze 100% of the codebase quickly, scanning millions of lines in minutes. They automatically find critical vulnerabilities, such as buffer overflows and SQL injection, which improves the overall quality of the code developed.

Key Steps to Run SAST Effectively:

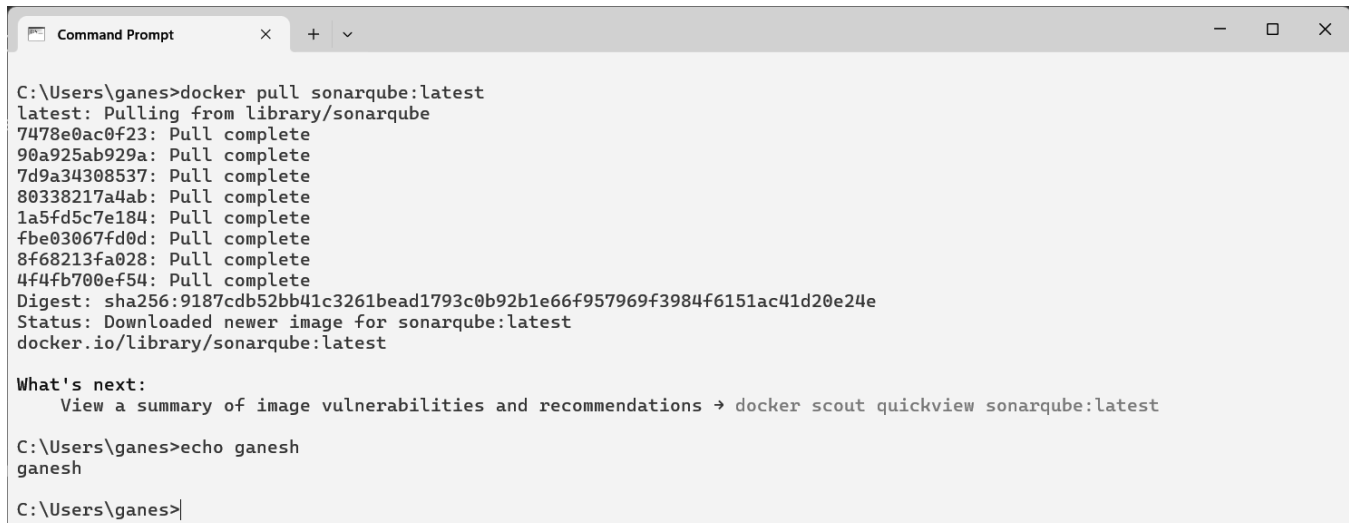
1. **Choose the Right Tool:** Select a SAST tool that supports the programming languages and frameworks you use.
2. **Set Up the Infrastructure:** Handle licensing, access control, and resources needed to deploy the tool.
3. **Customize the Tool:** Fine-tune the tool to meet your organization's needs, reducing false positives and adding rules for better vulnerability detection.
4. **Onboard Applications:** Prioritize scanning high-risk applications first, then gradually onboard all applications for regular scans.
5. **Analyze Results:** Review the scan results, remove false positives, and ensure issues are tracked for timely fixing.
6. **Provide Governance and Training:** Ensure development teams use the tools properly and integrate SAST into the application development process.

Ensure Docker is running.



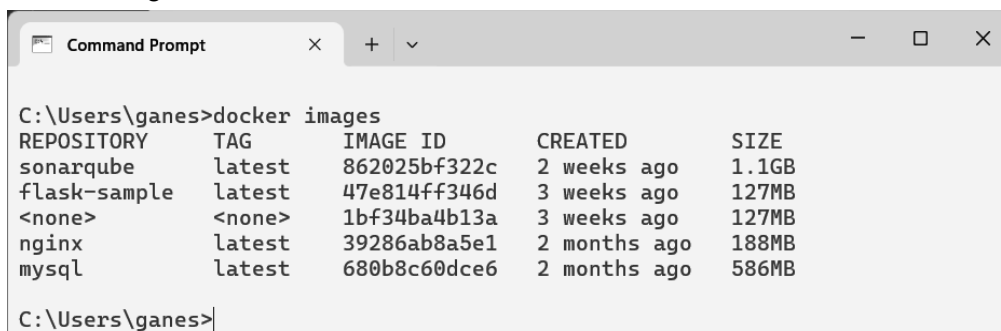
Open Command Prompt or PowerShell and execute the following command to pull the SonarQube image from Docker Hub.

`docker pull sonarqube:latest`



Verify that the image has been downloaded successfully by running.

`docker images`



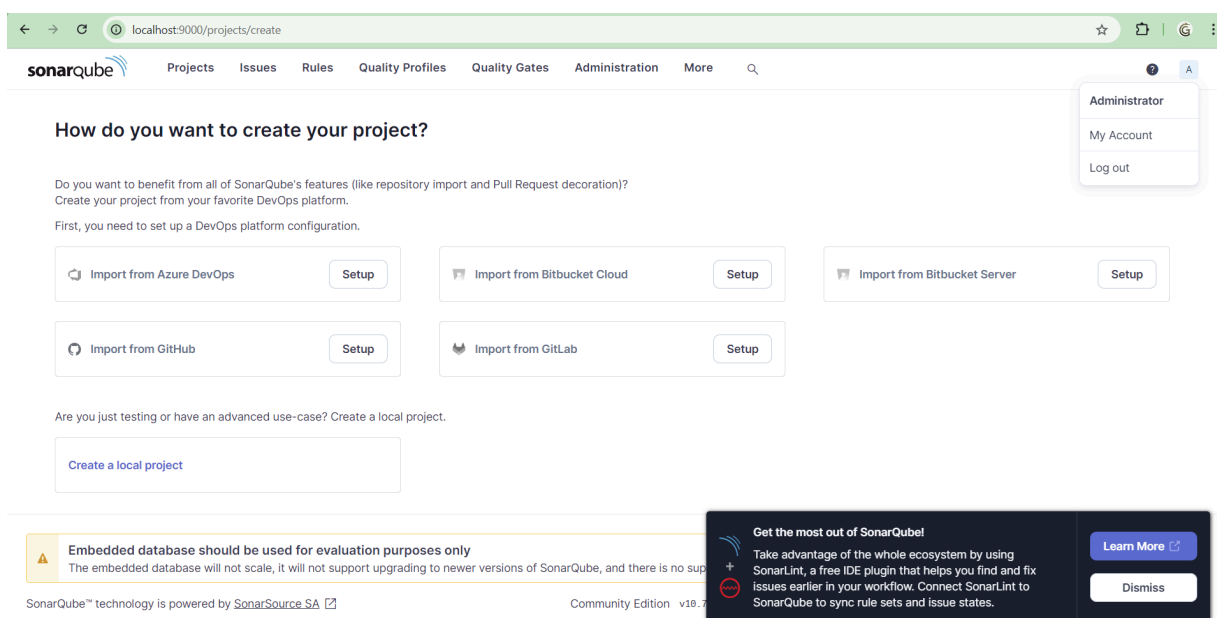
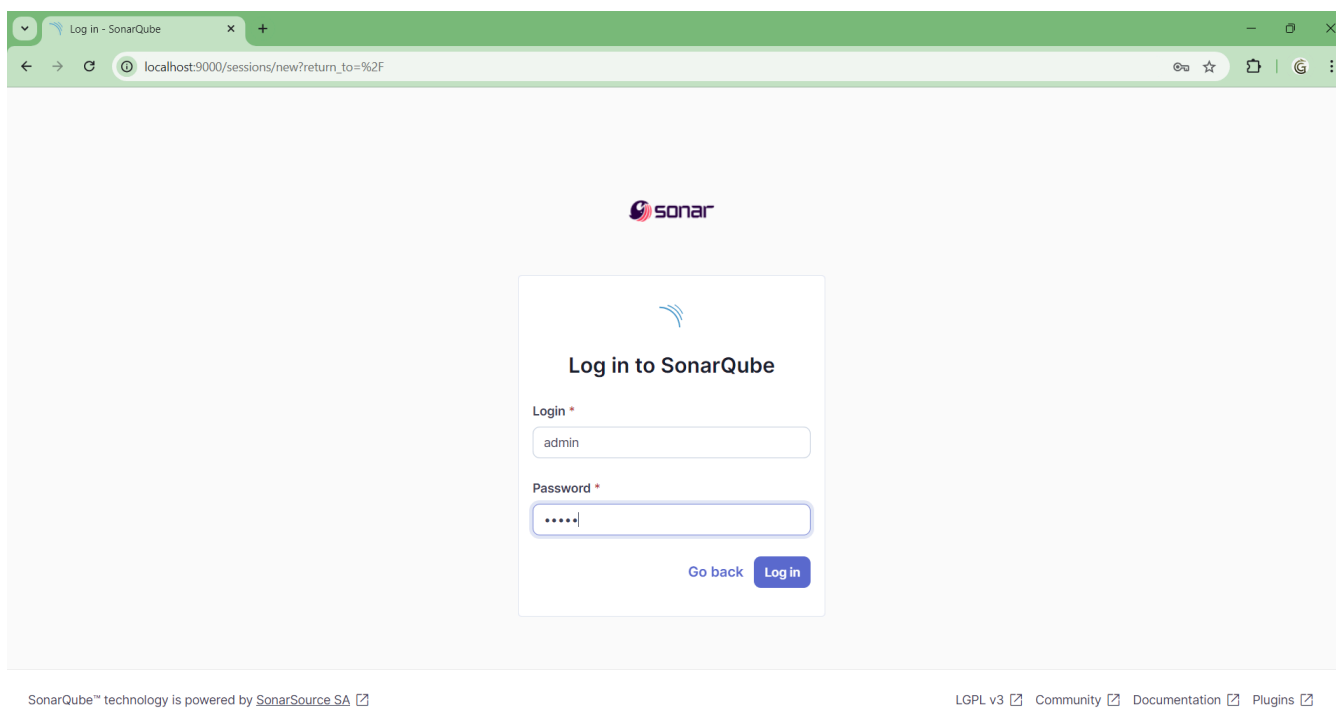
Execute the following command in your terminal to run the SonarQube Container.

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

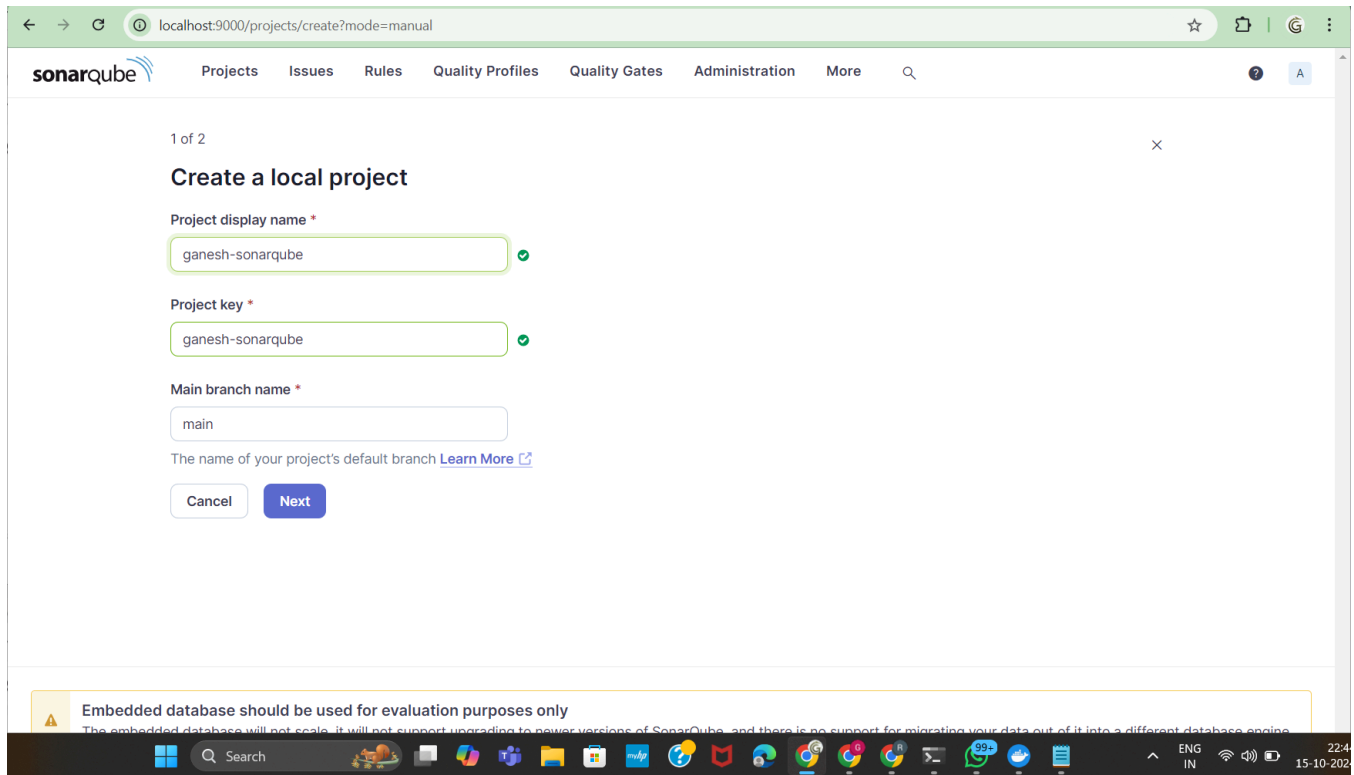
```
Command Prompt
ganesh
C:\Users\ganes>docker run -d --name ganesh-sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
76792a38bd54a0e40c7173d3e1291516352b559852db1a2a2fc654ed0bd16bd4
C:\Users\ganes>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
76792a38bd54   sonarqube:latest  "/opt/sonarqube/dock..."  22 seconds ago Up 20 seconds  0.0.0.0:9000->9000/tcp    ganesh-sonarqube
C:\Users\ganes>
```

Login to SonarQube: Use the default credentials:

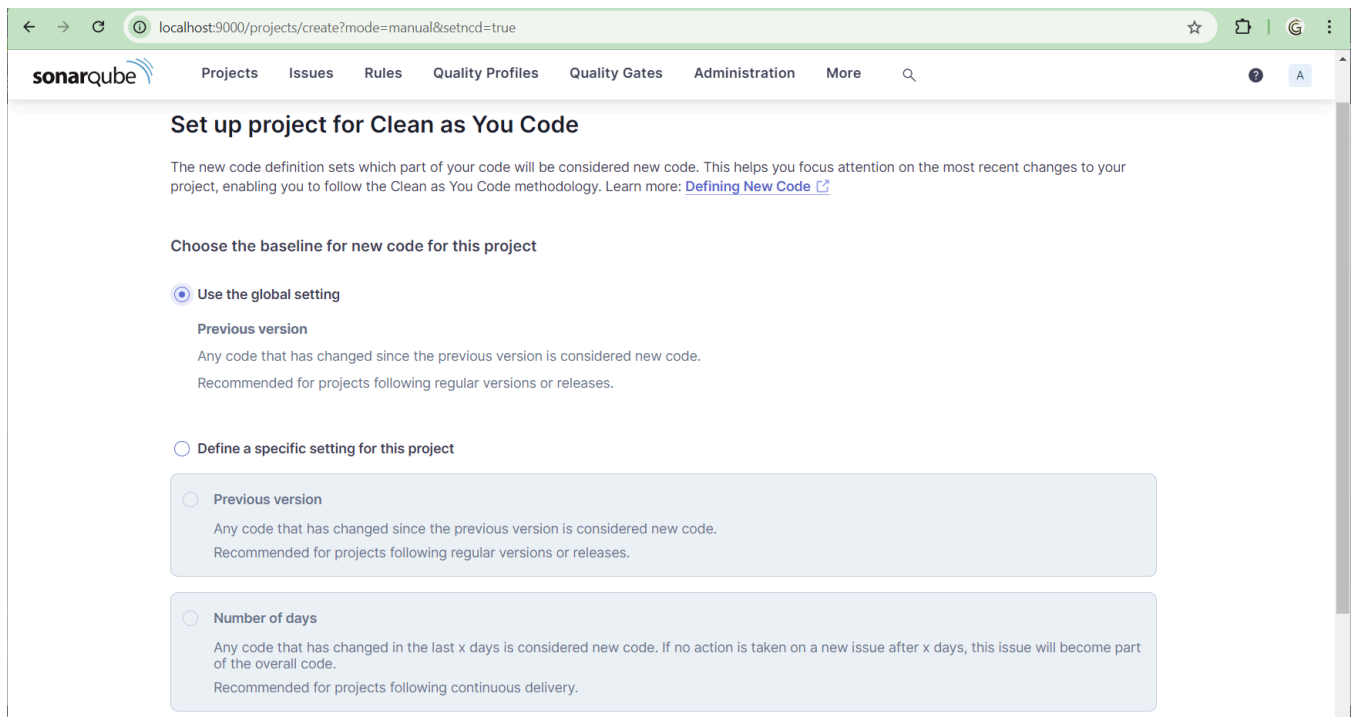
- Username: admin
- Password: admin



Click on Create Project. Name the project and follow the prompts to set it up.

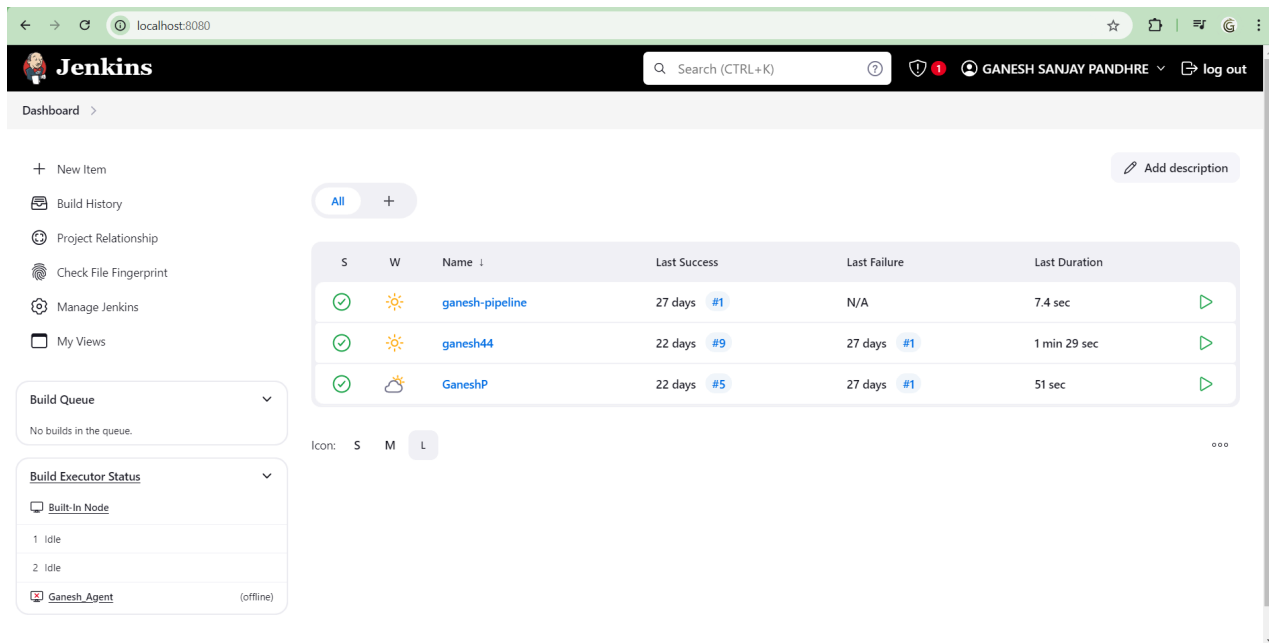


The screenshot shows the SonarQube web interface at localhost:9000/projects/create?mode=manual. The page is titled 'Create a local project' and is the first of two steps. It contains three input fields: 'Project display name' with the value 'ganesh-sonarqube', 'Project key' with the value 'ganesh-sonarqube', and 'Main branch name' with the value 'main'. Below these fields is a link 'Learn More' and two buttons: 'Cancel' and 'Next'. 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.'



The screenshot shows the SonarQube web interface at localhost:9000/projects/create?mode=manual&setncd=true. The page is titled 'Set up project for Clean as You Code'. It explains that the new code definition sets which part of the code will be considered new code. Below this, there are two main options: 'Use the global setting' (selected) and 'Define a specific setting for this project'. Under 'Define a specific setting for this project', there are two sub-options: 'Previous version' and 'Number of days'. The 'Previous version' option is described as 'Any code that has changed since the previous version is considered new code. Recommended for projects following regular versions or releases.' The 'Number of days' option is described as '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.'

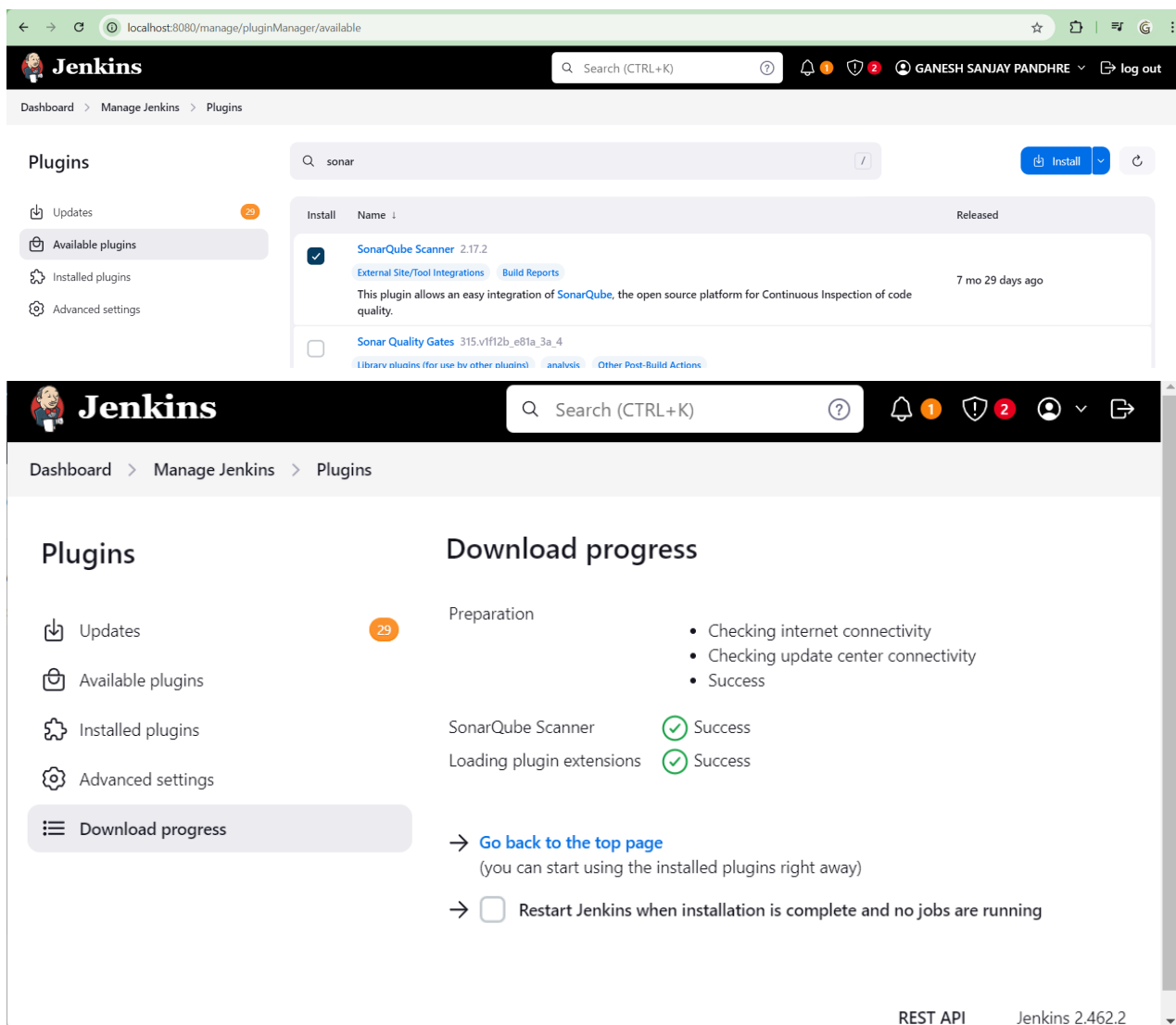
Open Jenkins Dashboard: Go to <http://localhost:8080> in your web browser (or the port where Jenkins is running).



The screenshot shows the Jenkins Dashboard at localhost:8080. The top navigation bar includes the Jenkins logo, a search bar, and the user name GANESH SANJAY PANDHRE with a log out button. The left sidebar contains links to New Item, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, and My Views. The main content area displays a table of build jobs with columns for status, name, last success, last failure, and last duration. Below the table, there are sections for Build Queue (showing no builds) and Build Executor Status (showing two idle executors: Built-In Node and Ganesh_Agent).

| S | W | Name | Last Success | Last Failure | Last Duration |
|---|---|-----------------|--------------|--------------|---------------|
| ✓ | ☀ | ganesh-pipeline | 27 days #1 | N/A | 7.4 sec |
| ✓ | ☀ | ganesh44 | 22 days #9 | 27 days #1 | 1 min 29 sec |
| ✓ | ☁ | GaneshP | 22 days #5 | 27 days #1 | 51 sec |

Manage Jenkins → Manage Plugins → Available tab, search for SonarQube Scanner. Check the box next to it and click Install without Restart.



The screenshot shows the Jenkins Manage Plugins page at localhost:8080/manage/pluginManager/available. The top navigation bar includes the Jenkins logo, a search bar, and the user name GANESH SANJAY PANDHRE with a log out button. The left sidebar contains links to Updates, Available plugins, Installed plugins, and Advanced settings. The main content area displays a table of available plugins with columns for install status, name, and release date. Below the table, there are sections for Download progress and a link to Go back to the top page.

| 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 29 days ago |
| <input type="checkbox"/> | Sonar Quality Gates 315.v1f12b_e81a_3a_4 Library plugins (for use by other plugins) analysis Other Post-Build Actions | |

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

Manage Jenkins → Configure System. Scroll down to the SonarQube Servers section and add a new SonarQube server.

- Name: Give it a name (e.g., SonarQube).
- Server URL: Enter `http://localhost:9000`.

The screenshot shows the Jenkins 'Manage Jenkins > System' configuration page. The 'SonarQube installations' section is active, showing a list of installations. A new installation is being added with the following details:

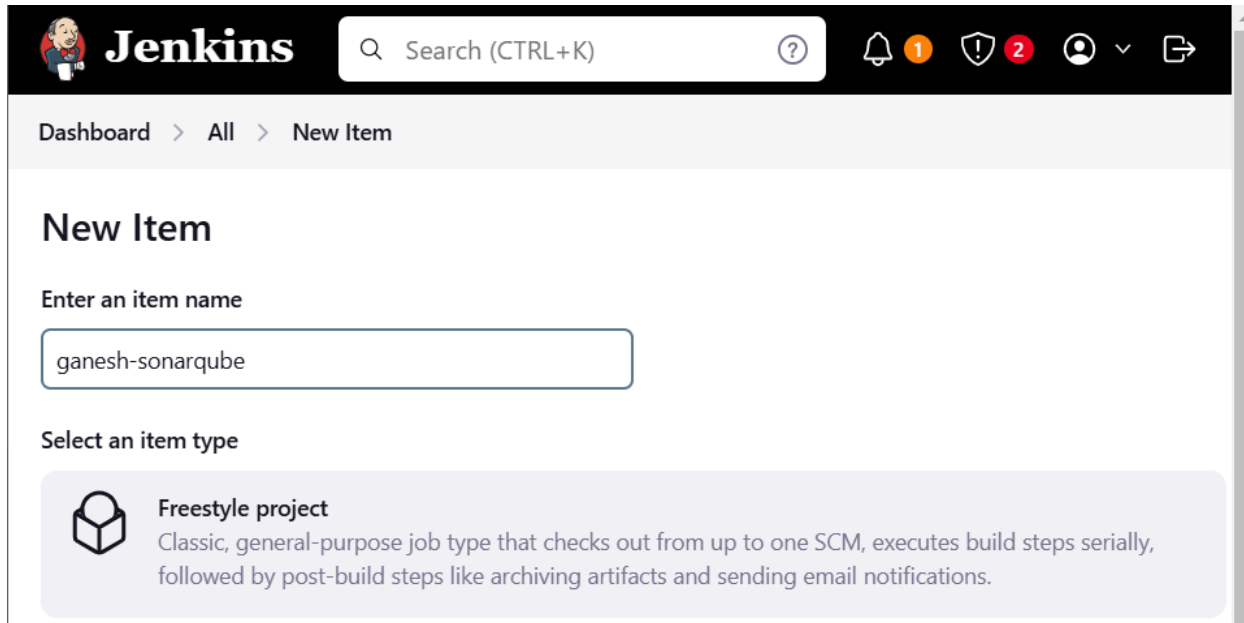
- Name:** sonarqube
- Server URL:** http://localhost:9000 (Default is http://localhost:9000)
- Server authentication token:** - none - (Mandatory when anonymous access is disabled)
- Buttons:** + Add, Advanced (dropdown), Save, Apply

Manage Jenkins → Global Tool Configuration. Find SonarQube Scanner and choose the latest version. Check the Install automatically option.

The screenshot shows the Jenkins 'Manage Jenkins > Tools' configuration page. The 'SonarQube Scanner installations' section is active, showing a list of installations. A new installation is being added with the following details:

- Name:** sonarqube
- Install automatically:** ☒ (with a help icon)
- Install from Maven Central:**
 - Version:** SonarQube Scanner 6.2.0.4584
 - Buttons:** Add Installer (dropdown), Save, Apply

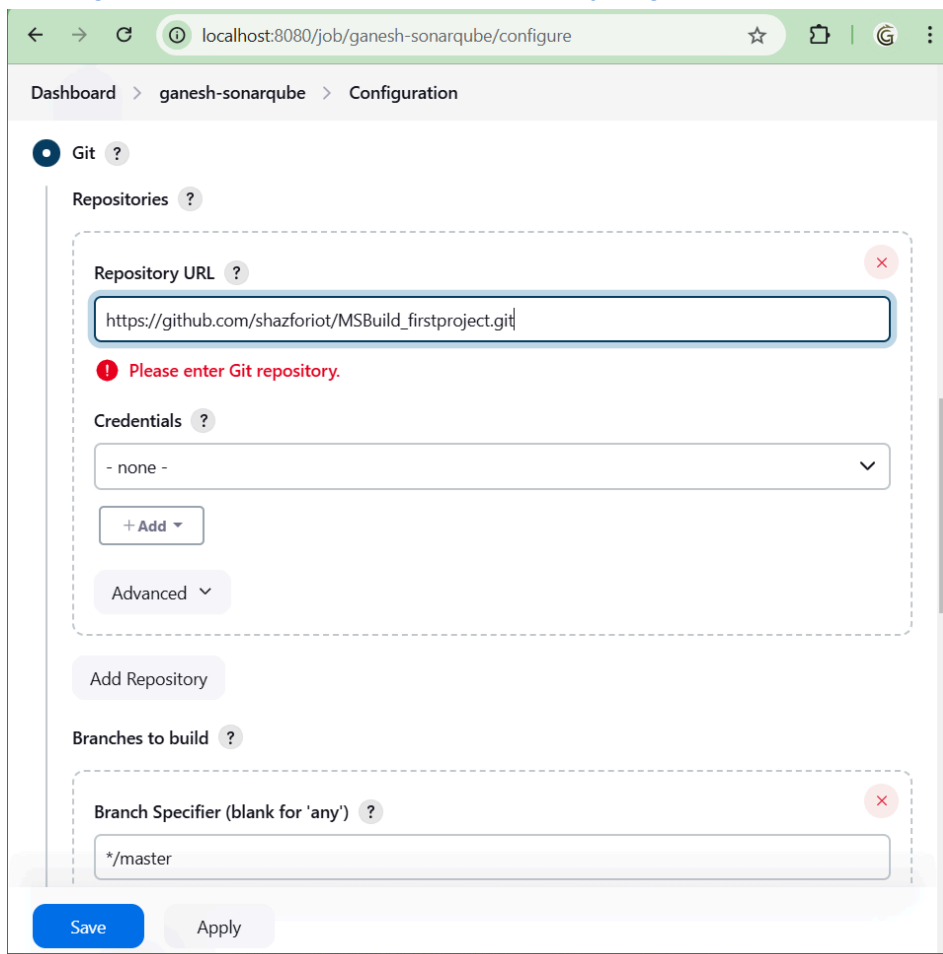
On the Jenkins dashboard, click on New Item. Enter a name for your project. Choose the Freestyle project and click OK.



The image shows the Jenkins 'New Item' configuration page. At the top, there's a search bar and navigation icons. Below the header, the breadcrumb trail is 'Dashboard > All > New Item'. The main heading is 'New Item'. Under 'Enter an item name', the text 'ganesh-sonarqube' is entered in the input field. Under 'Select an item type', the 'Freestyle project' option is selected, highlighted with a light blue background. A description for 'Freestyle project' is provided: '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.'

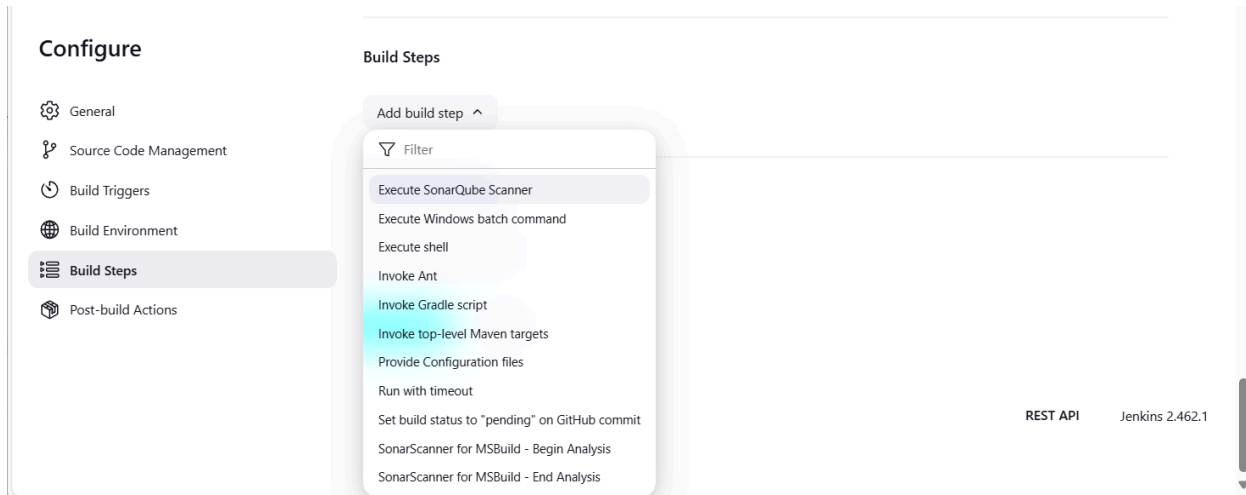
In the project configuration, look for the Source Code Management section. Choose Git and enter the repository URL.

https://github.com/shazforiot/MSBuild_firstproject.git



The image shows the Jenkins 'Configuration' page for the project 'ganesh-sonarqube'. The breadcrumb trail is 'Dashboard > ganesh-sonarqube > Configuration'. The 'Git' option is selected under 'Source Code Management'. The 'Repositories' section is expanded, showing a 'Repository URL' input field with the value 'https://github.com/shazforiot/MSBuild_firstproject.git'. A red error message 'Please enter Git repository.' is displayed below the input field. The 'Credentials' dropdown is set to '- none -'. There is a '+ Add' button and an 'Advanced' dropdown. Below the 'Repositories' section is an 'Add Repository' button. The 'Branches to build' section is expanded, showing a 'Branch Specifier (blank for 'any')' input field with the value '*/master'. At the bottom, there are 'Save' and 'Apply' buttons.

Scroll down to the Build section. Click on Add build step and select Execute SonarQube Scanner.



Enter Analysis Properties

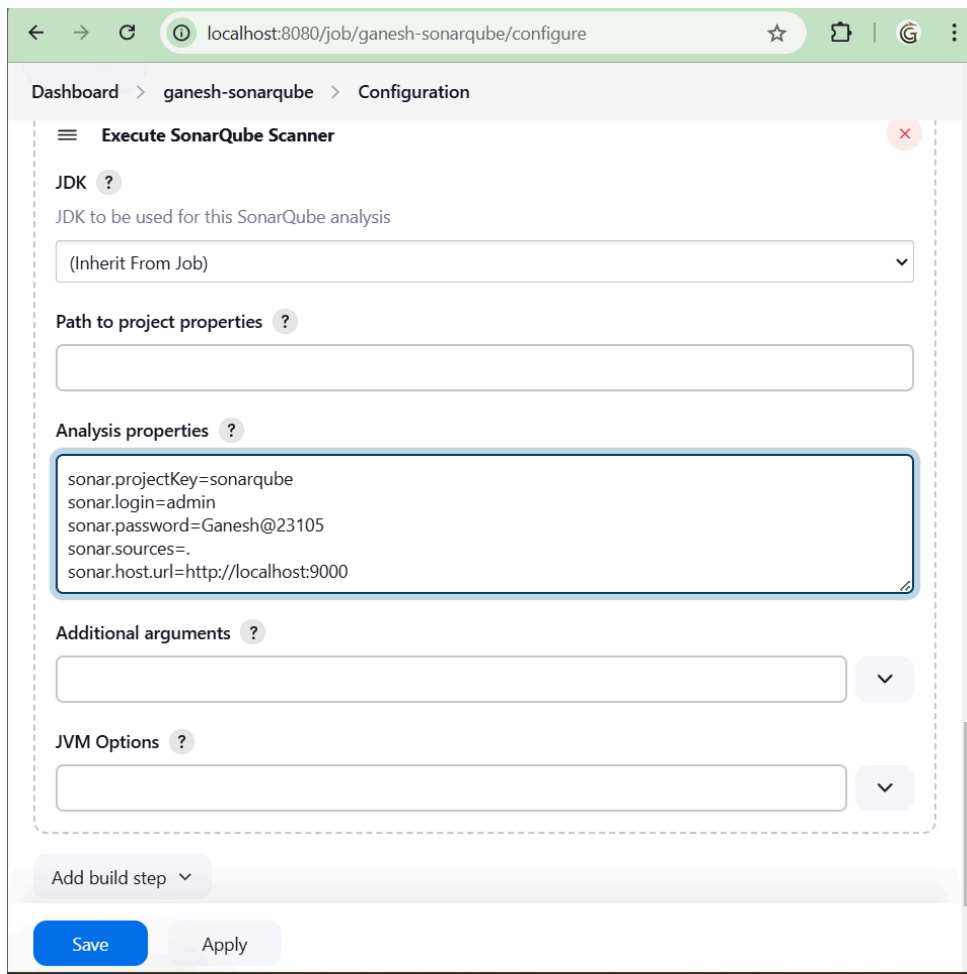
sonar.projectKey=sonarqube

sonar.login=admin

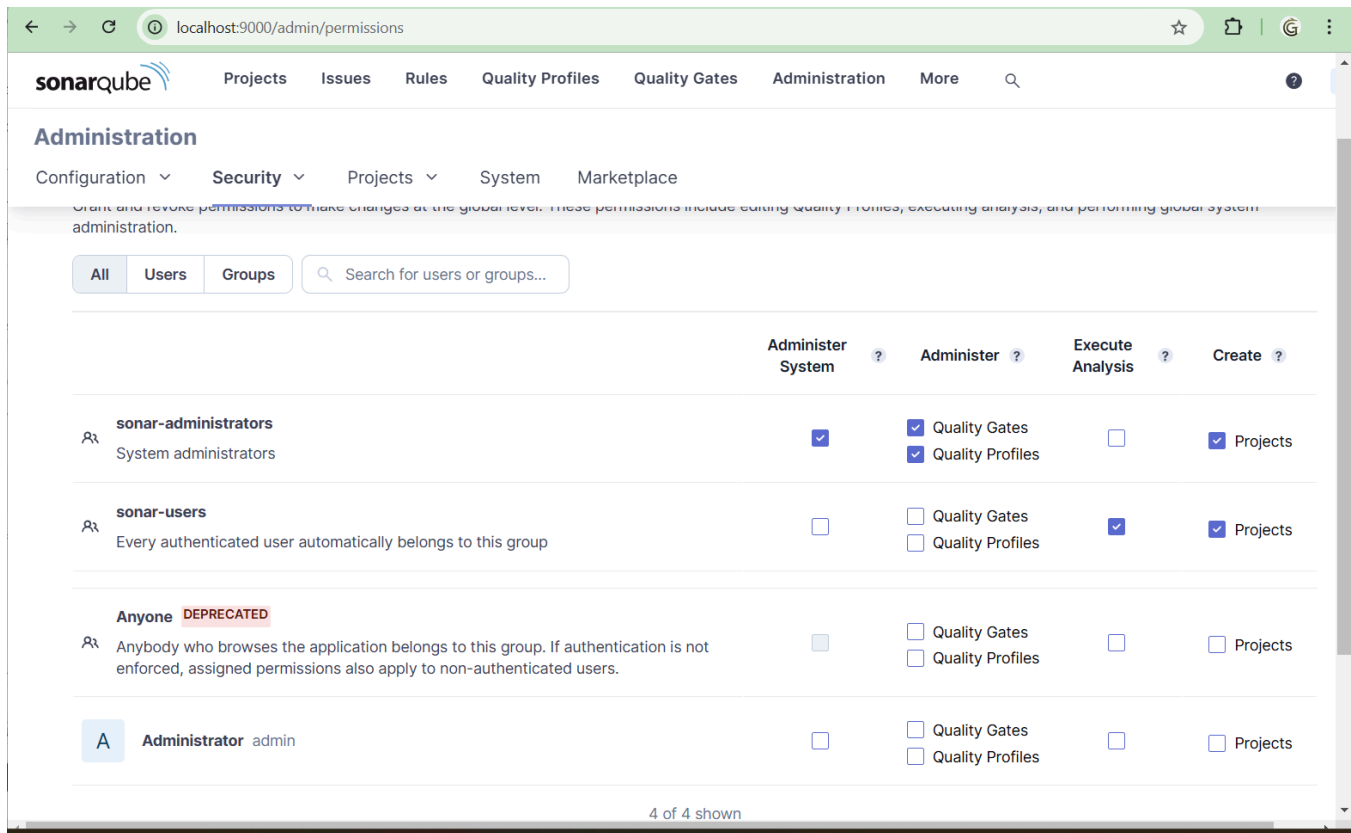
sonar.password=admin

sonar.sources=.

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



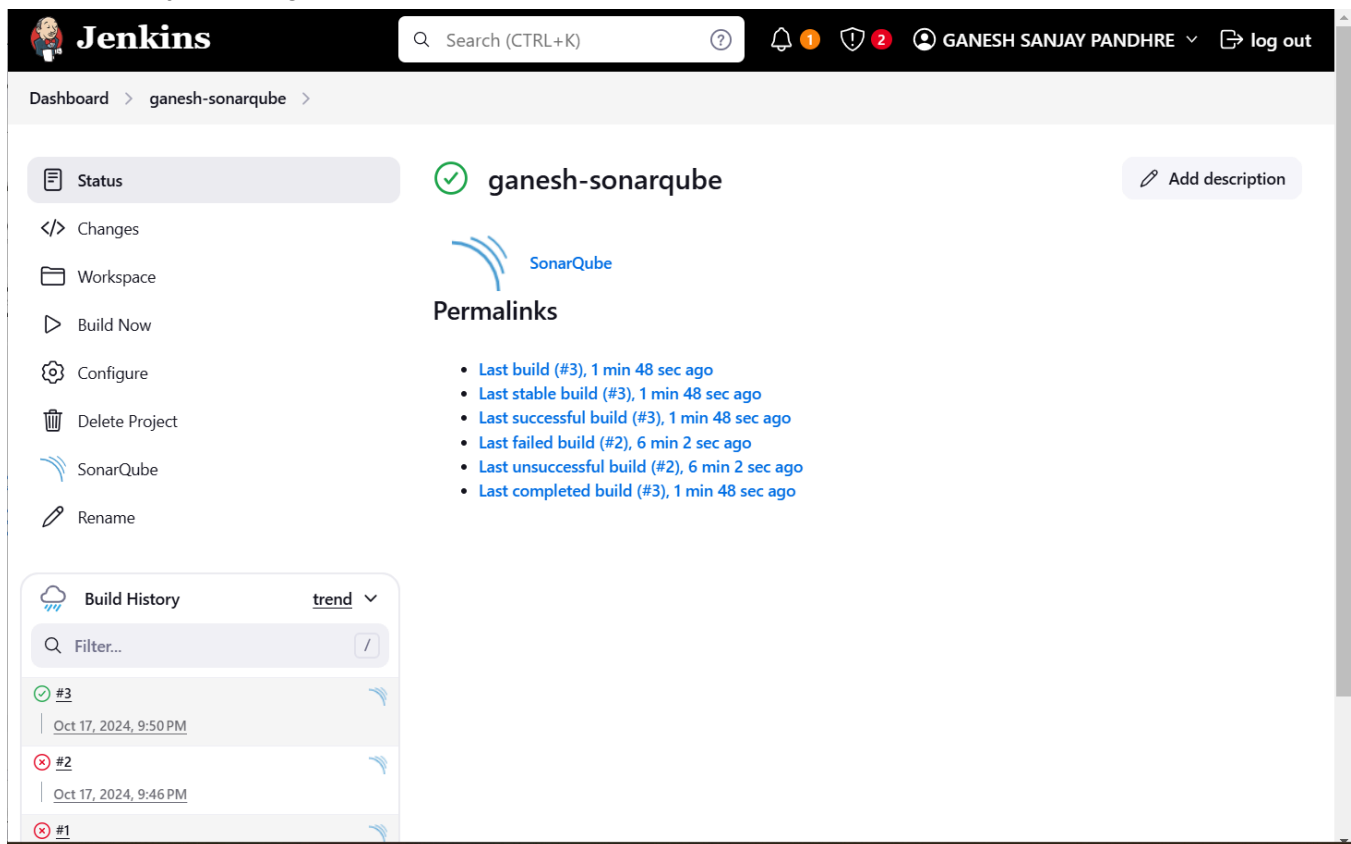
Go to <http://localhost:9000/admin/permissions> and give the Admin user execute permissions.



The screenshot shows the SonarQube Administration interface, specifically the Security section. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main header is 'Administration' with sub-tabs for Configuration, Security (selected), Projects, System, and Marketplace. Below the header, there's a search bar and tabs for 'All', 'Users', and 'Groups'. The main content area displays a table of users and groups with their permissions. The table has columns for 'System administrators', 'Quality Gates', 'Quality Profiles', 'Execute Analysis', and 'Create'. The rows are: 'sonar-administrators' (checked for all), 'sonar-users' (checked for Quality Profiles, Execute Analysis, and Create), 'Anyone' (DEPRECATED, unchecked for all), and 'Administrator' (unchecked for all). The bottom of the table shows '4 of 4 shown'.

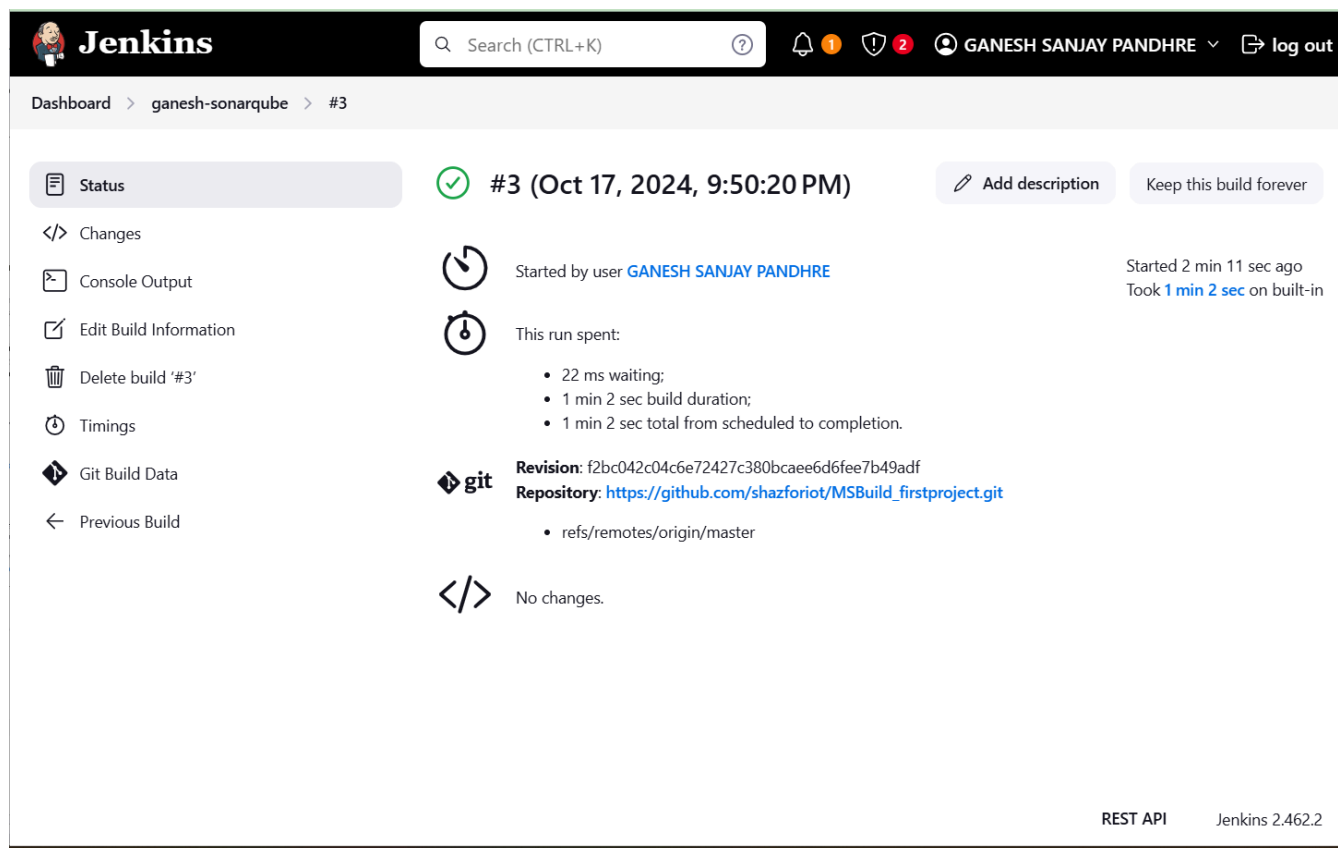
| | System administrators | Quality Gates | Quality Profiles | Execute Analysis | Create |
|--|-------------------------------------|---|-------------------------------------|--|--------|
| sonar-administrators System administrators | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> 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 type="checkbox"/> | <input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles | <input type="checkbox"/> | <input type="checkbox"/> Projects | |

Save the project configuration and click Build Now.



The screenshot shows the Jenkins Dashboard for the project 'ganesh-sonarqube'. The top navigation bar includes a search bar, a user profile for 'GANESH SANJAY PANDHRE', and a 'log out' button. The main content area shows the project status as 'Success' (green checkmark) and a 'Build Now' button. Below the status, there's a 'Permalinks' section with a list of build links: 'Last build (#3), 1 min 48 sec ago', 'Last stable build (#3), 1 min 48 sec ago', 'Last successful build (#3), 1 min 48 sec ago', 'Last failed build (#2), 6 min 2 sec ago', 'Last unsuccessful build (#2), 6 min 2 sec ago', and 'Last completed build (#3), 1 min 48 sec ago'. On the left sidebar, there's a 'Build History' section with a filter bar and a list of builds: '#3' (Oct 17, 2024, 9:50 PM), '#2' (Oct 17, 2024, 9:46 PM), and '#1' (Oct 17, 2024, 9:46 PM).

Click on the build number in the build history.



The screenshot shows the Jenkins dashboard for a build named '#3' (Oct 17, 2024, 9:50:20 PM). The build status is 'Success' (green checkmark). The user 'GANESH SANJAY PANDHRE' started the build. The console output shows the build process, including fetching changes from the remote Git repository and checking out the revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf. The build took 1 min 2 sec on built-in.

Dashboard > ganesh-sonarqube > #3

Status **#3 (Oct 17, 2024, 9:50:20 PM)** Add description Keep this build forever

Changes

Console Output

Edit Build Information

Delete build '#3'

Timings

Git Build Data

Previous Build

Started by user **GANESH SANJAY PANDHRE** Started 2 min 11 sec ago Took 1 min 2 sec on built-in

This run spent:

- 22 ms waiting;
- 1 min 2 sec build duration;
- 1 min 2 sec total from scheduled to completion.

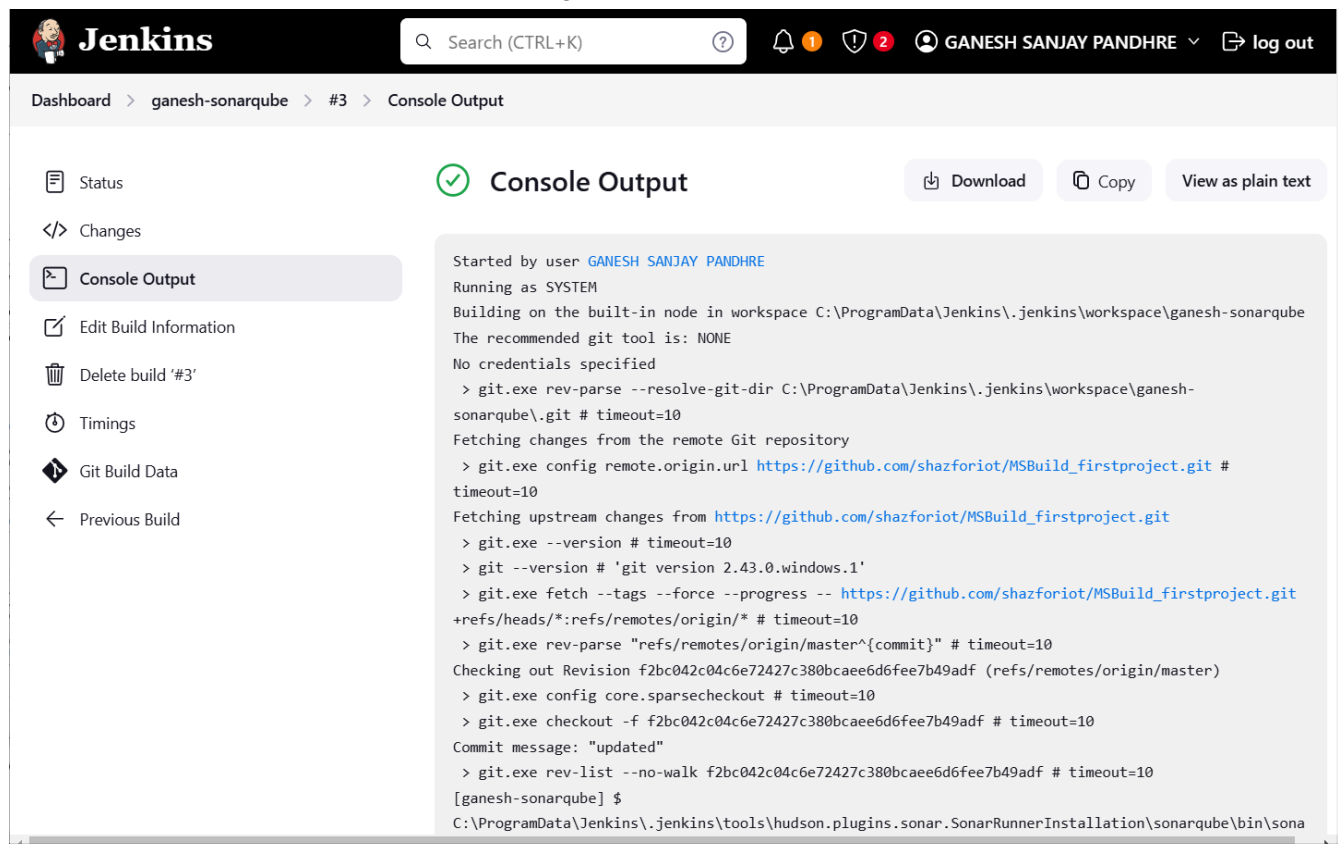
Revision: f2bc042c04c6e72427c380bcaee6d6fee7b49adf
Repository: https://github.com/shazforiot/MSBuild_firstproject.git

refs/remotes/origin/master

No changes.

REST API Jenkins 2.462.2

Click on Console Output to see the build logs.



The screenshot shows the Jenkins console output for the build '#3'. The output displays the build process, including fetching changes from the remote Git repository and checking out the revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf. The build took 1 min 2 sec on built-in.

Dashboard > ganesh-sonarqube > #3 > Console Output

Status **Console Output** Download Copy View as plain text

Changes

Console Output

Edit Build Information

Delete build '#3'

Timings

Git Build Data

Previous Build

Started by user **GANESH SANJAY PANDHRE**

Running as SYSTEM

Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\ganesh-sonarqube

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\ganesh-sonarqube\.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 2.43.0.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"

> git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10

[ganesh-sonarqube] \$

C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sona

Go back to <http://localhost:9000> and navigate to your project. Review the results of the analysis.

The screenshot displays the SonarQube dashboard for a project named 'sonarqube'. The main navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The current view is the 'Overview' tab, which shows the 'Quality Gate' status as 'Passed' with a green checkmark. A warning message indicates that the last analysis has warnings, with a link to 'See details'. Below this, there are two tabs: 'New Code' and 'Overall Code'. The 'Overall Code' tab is active, showing a grid of metrics: Security (0 Open issues, grade A), Reliability (0 Open issues, grade A), Maintainability (0 Open issues), Accepted issues (0, with a note 'Valid issues that were not fixed'), Coverage (On 0 lines to cover), and Duplications (0.0%, On 86 lines).

| Metric | Value | Grade/Status |
|-----------------|----------------------|--------------|
| Security | 0 Open issues | A |
| Reliability | 0 Open issues | A |
| Maintainability | 0 Open issues | |
| Accepted issues | 0 | |
| Coverage | On 0 lines to cover. | |
| Duplications | 0.0% On 86 lines. | |

CONCLUSION :

Integrating Static Application Security Testing (SAST) into the software development process helps identify and fix security vulnerabilities early, improving the overall quality of applications. By regularly running SAST tools, organizations can ensure safer code and reduce the risk of security issues in their final products.