



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.	42
Name	NAIKWADI YASH SHIVDAS
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.

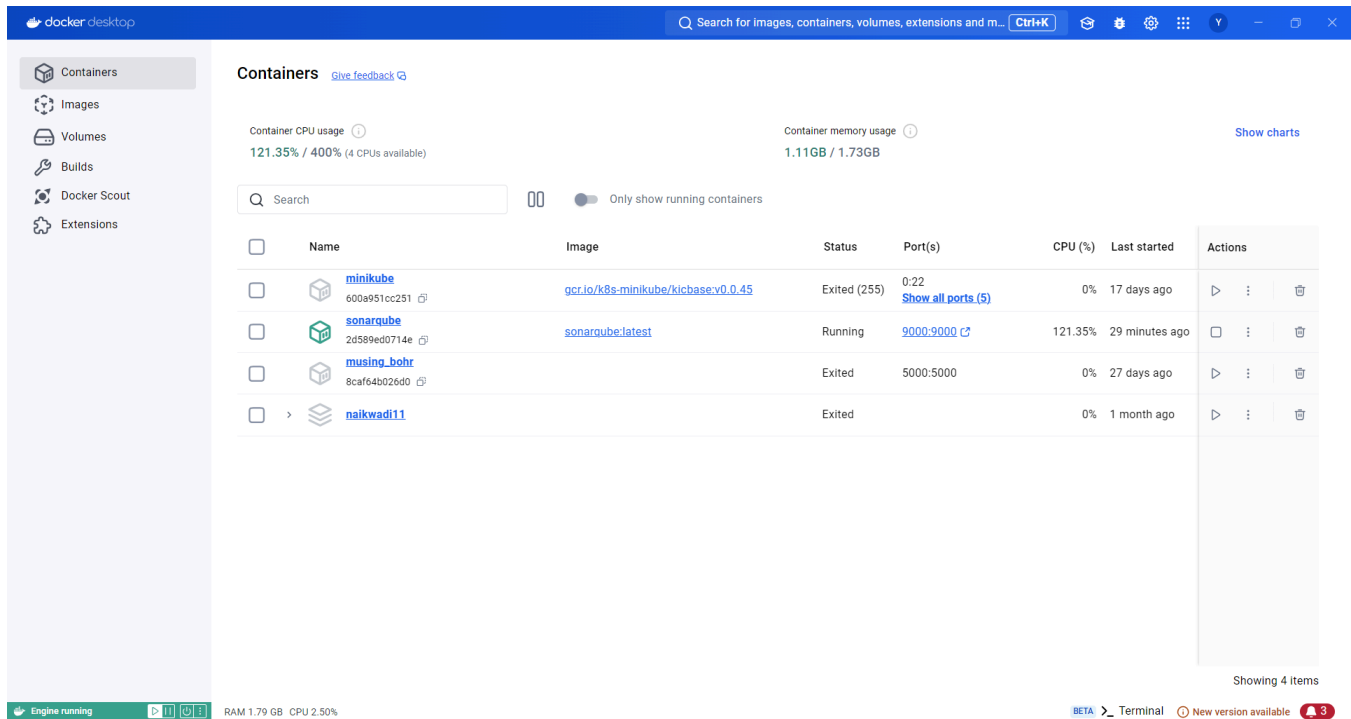
Why is SAST Important?

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`

```

C:\Users\acer>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
e40af92b2267b1f9010bce98466607f969cec21a99ba588aa8d2e0f8038127d2

C:\Users\acer>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
e40af92b2267   sonarqube:latest  "/opt/sonarqube/dock..."  2 hours ago   Up 2 hours   0.0.0.0:9000->9000/tcp         sonarqube

```

Verify that the image has been downloaded successfully by running.

`docker images`

```
Command Prompt

C:\Users\Admin>docker images
REPOSITORY              TAG                IMAGE ID           CREATED            SIZE
yashnaikwadi/mywebapp-demo 01                aa5af6863cd5      2 weeks ago       1.4GB
node                     20                ef970e331526      2 weeks ago       1.1GB
gcr.io/k8s-minikube/kicbase v0.0.45           aeed0e1d4642      6 weeks ago       1.28GB
nginx                    latest            39286ab8a5e1      2 months ago      188MB
sonarqube                 latest            2433ac783140      2 months ago      1.07GB
mysql                     latest            680b8c60dce6      3 months ago      586MB
python                    3.9-slim-buster   c84dbfe3b8de      16 months ago     116MB

C:\Users\Admin>
```

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

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

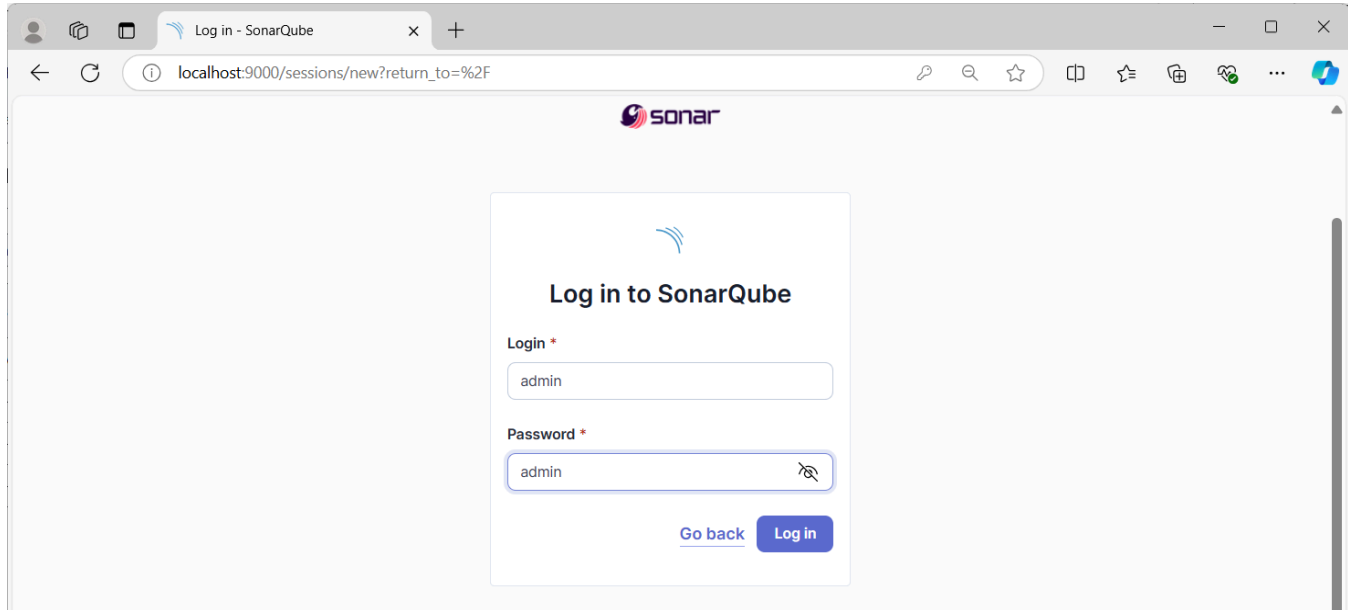
```
Command Prompt

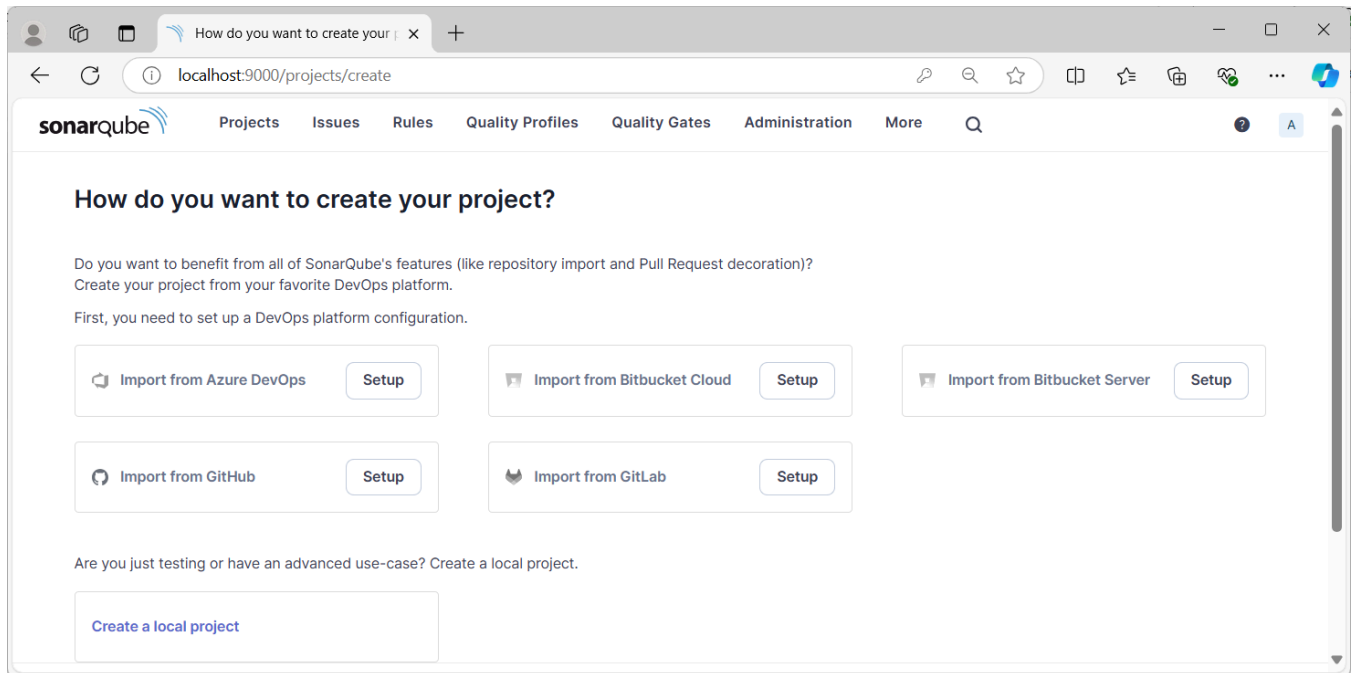
C:\Users\Admin>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
0504e3f9771244bd3bcd4c66bf92905e59169a3d0baa198164a20bb788daa80

C:\Users\Admin>
```

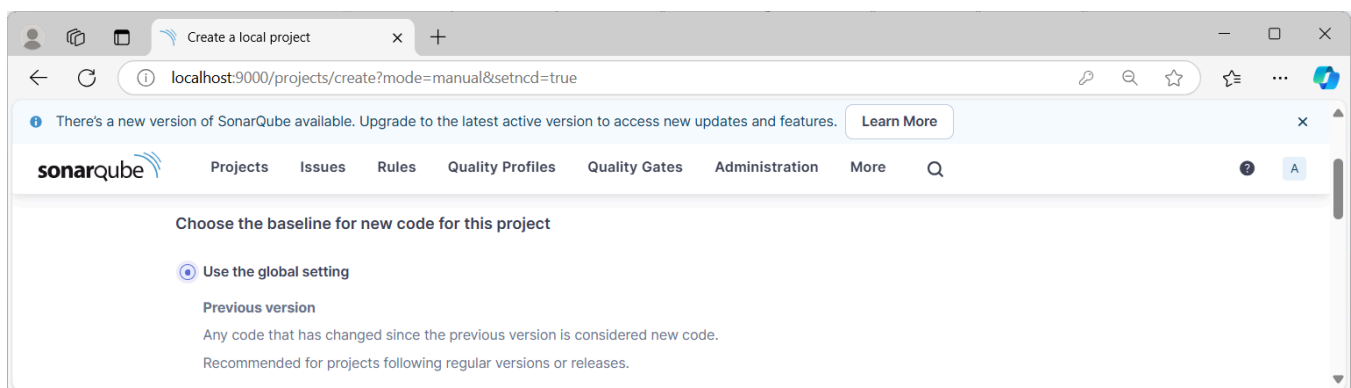
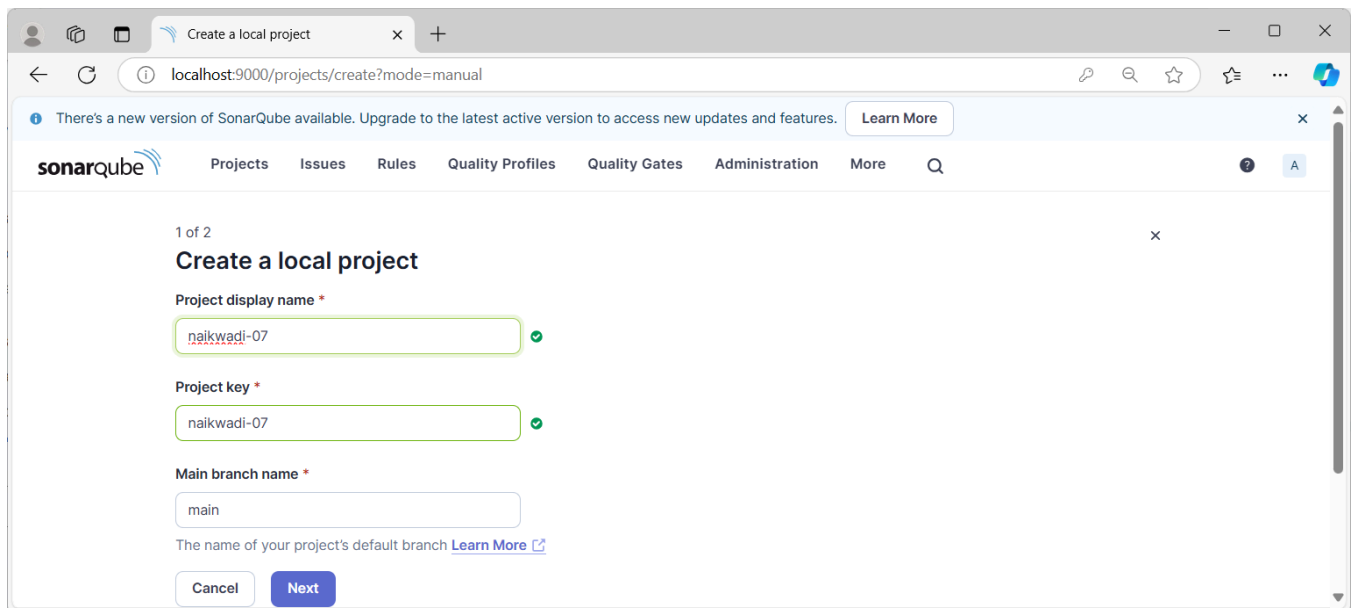
Login to SonarQube: Use the default credentials:

- Username: admin
- Password: admin

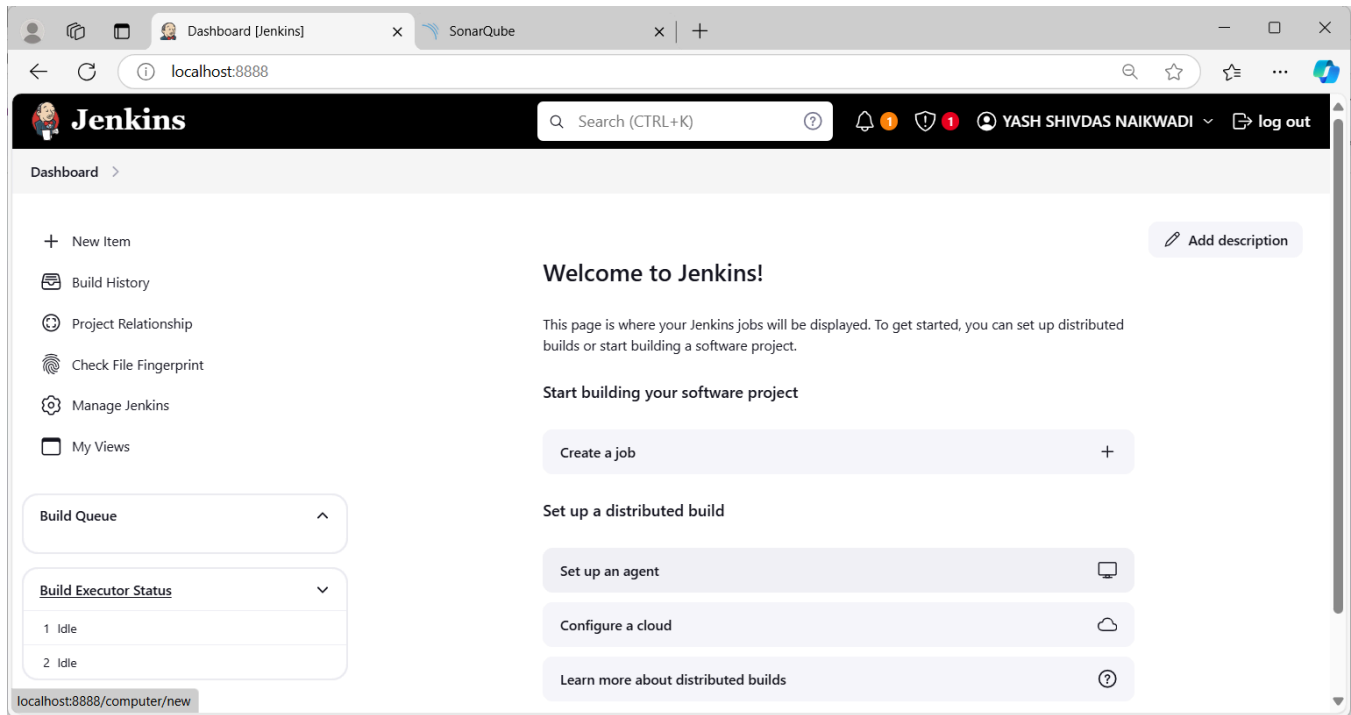




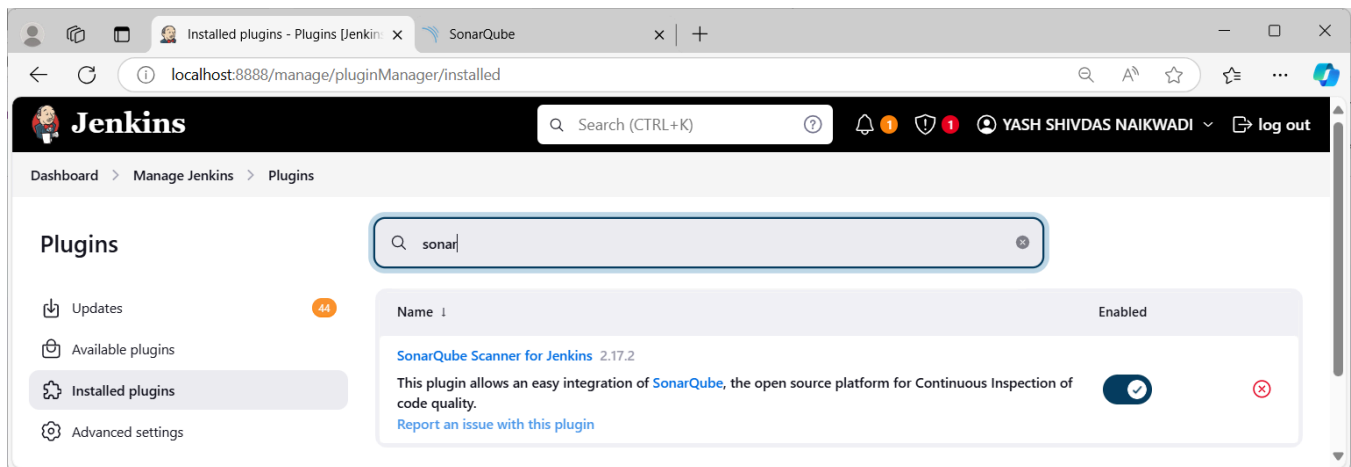
Click on Create Project. Name the project and follow the prompts to set it up.



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



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



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

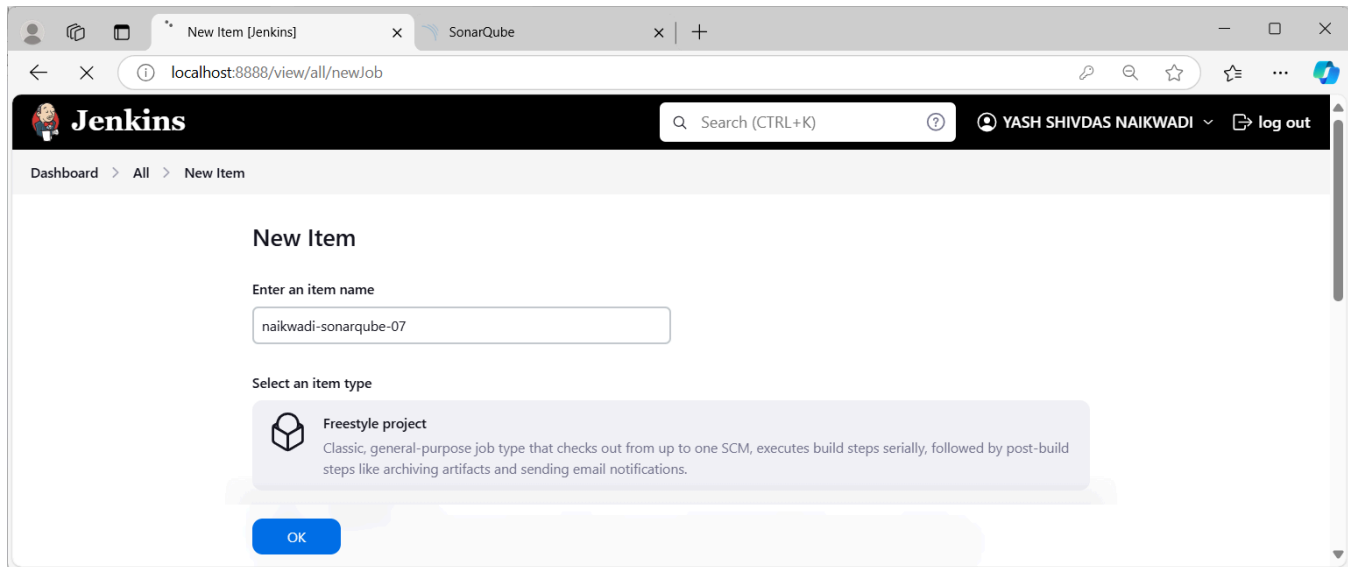
- Name: Give it a name (e.g., SonarQube).
- Server URL: Enter `http://localhost:9000`.

The screenshot shows the Jenkins 'System' configuration page for 'SonarQube servers'. The page has a breadcrumb trail: Dashboard > Manage Jenkins > System >. Under 'SonarQube servers', there is a note: 'If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.' Below this, the 'Environment variables' checkbox is checked. The 'SonarQube installations' section is titled 'List of SonarQube installations' and contains a dashed box with the following fields: 'Name' (text input with 'sonarqube'), 'Server URL' (text input with 'http://localhost:9000', with a default note 'Default is http://localhost:9000'), and 'Server authentication token' (dropdown menu with '- none -' selected, with a note 'SonarQube authentication token. Mandatory when anonymous access is disabled.'). There is also an '+ Add' button and an 'Advanced' dropdown. At the bottom are 'Save' and 'Apply' buttons.

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

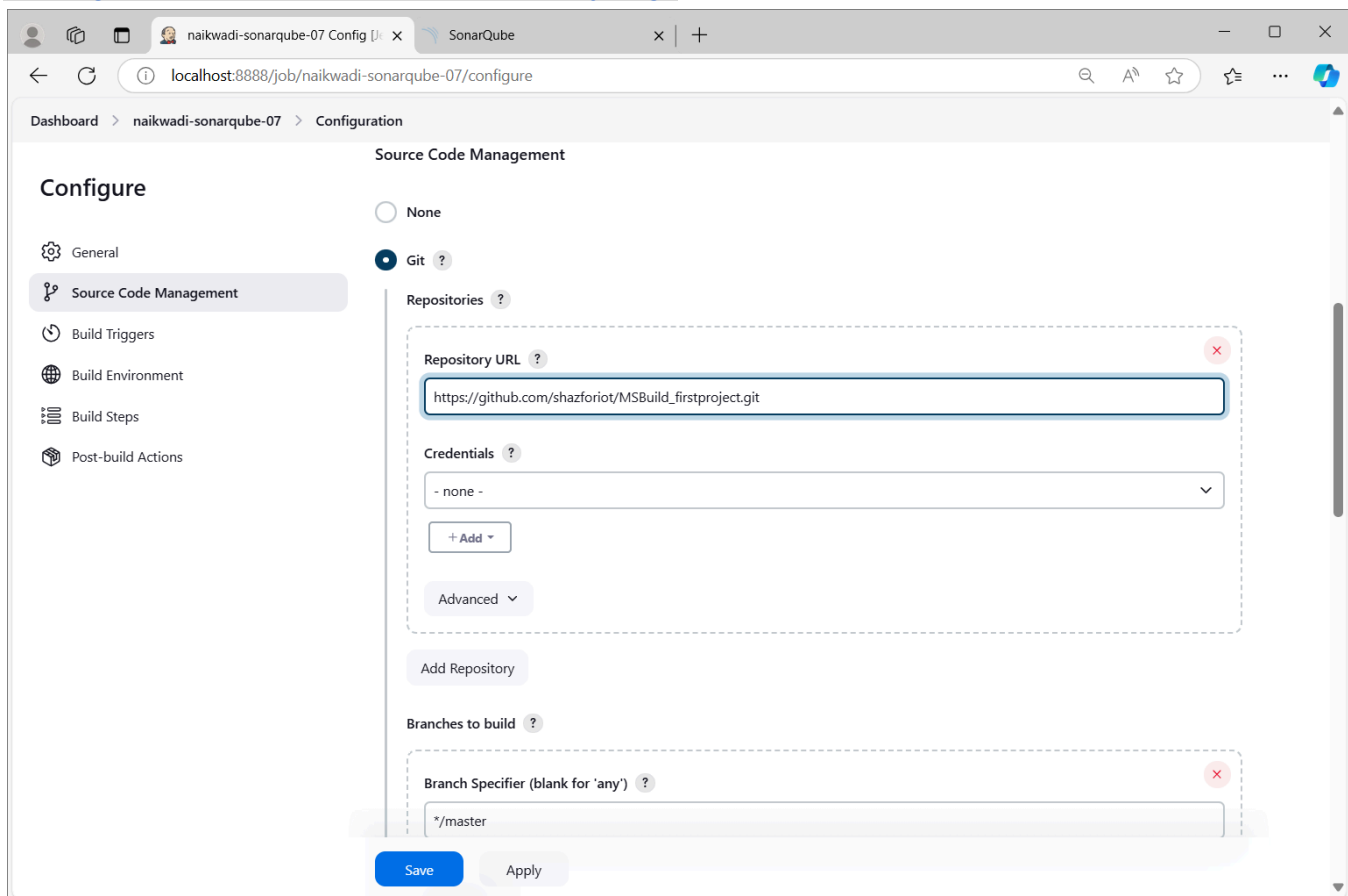
This screenshot is identical to the one above, showing the Jenkins 'System' configuration page for 'SonarQube servers'. It includes the same breadcrumb trail, note about environment variables, checked 'Environment variables' checkbox, 'SonarQube installations' section with a dashed box containing 'Name' (sonarqube), 'Server URL' (http://localhost:9000), and 'Server authentication token' (- none -), along with the '+ Add' button, 'Advanced' dropdown, and 'Save'/'Apply' buttons.

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

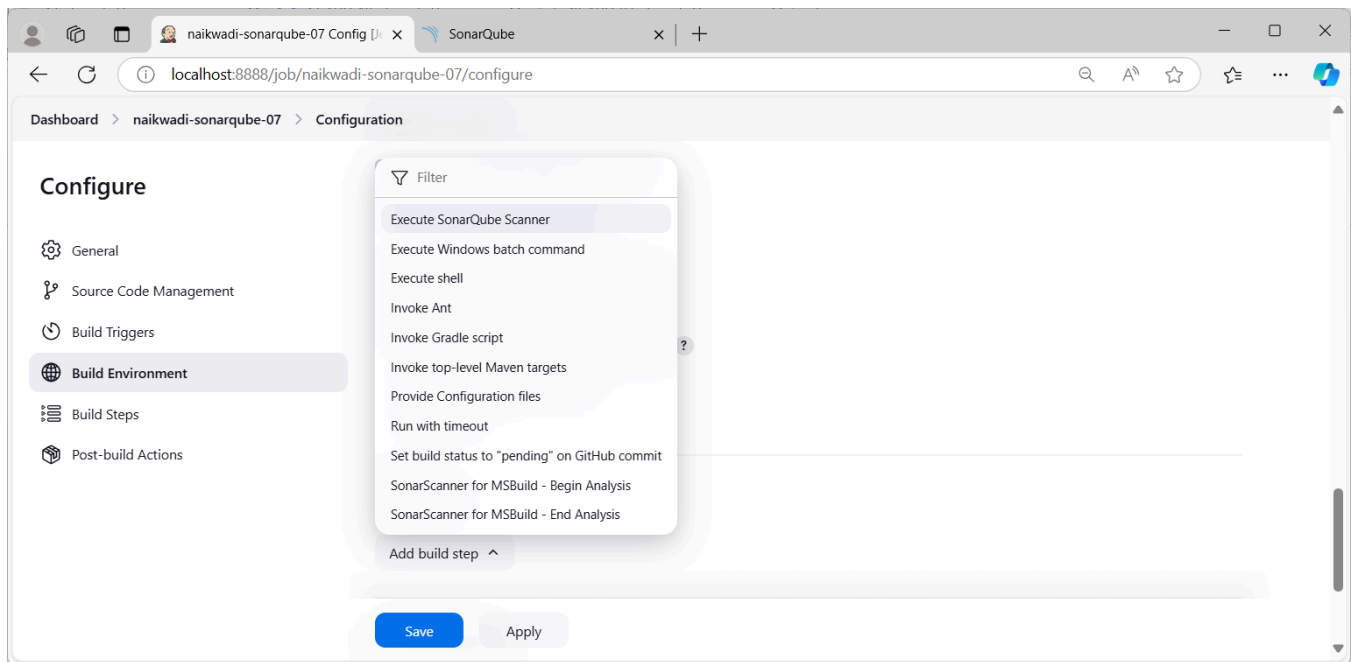


In the project configuration, look for the Source Code Management section. Choose Git and enter the repository URL.

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



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



Enter Analysis Properties

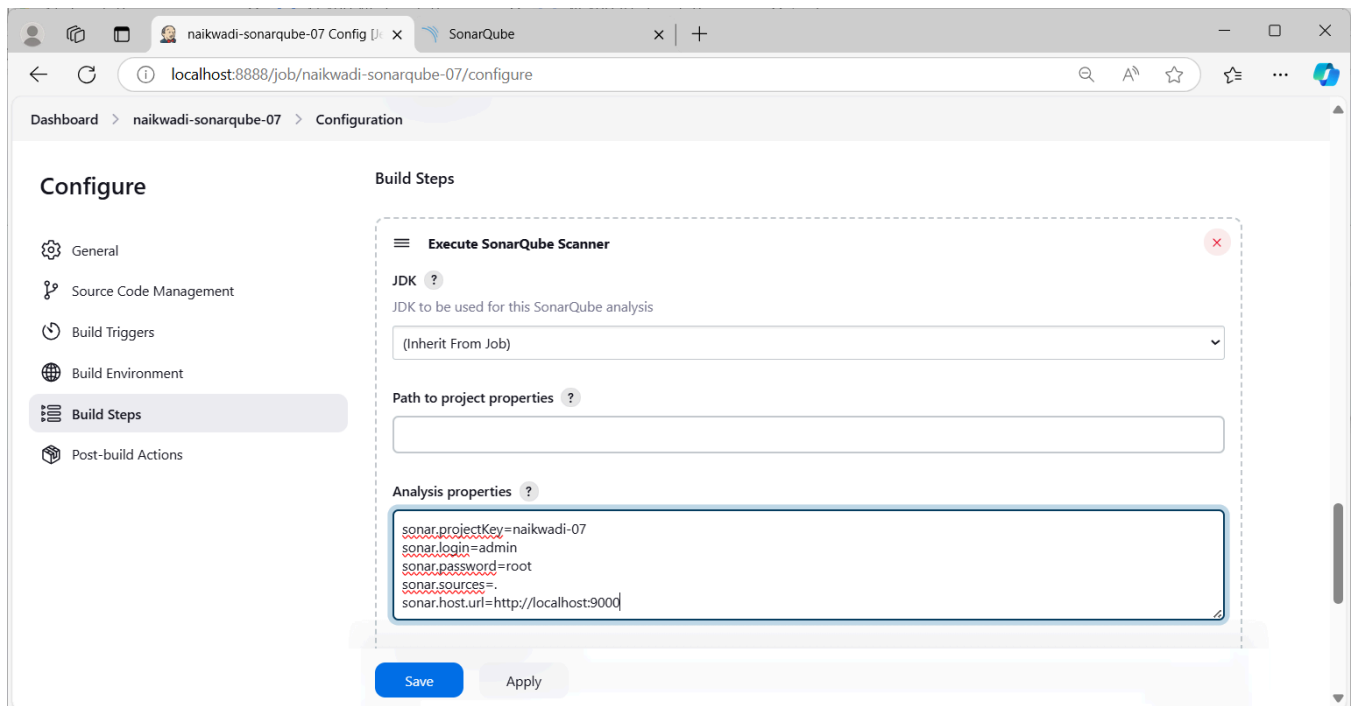
`sonar.projectKey=naikwadi-07`

`sonar.login=admin`

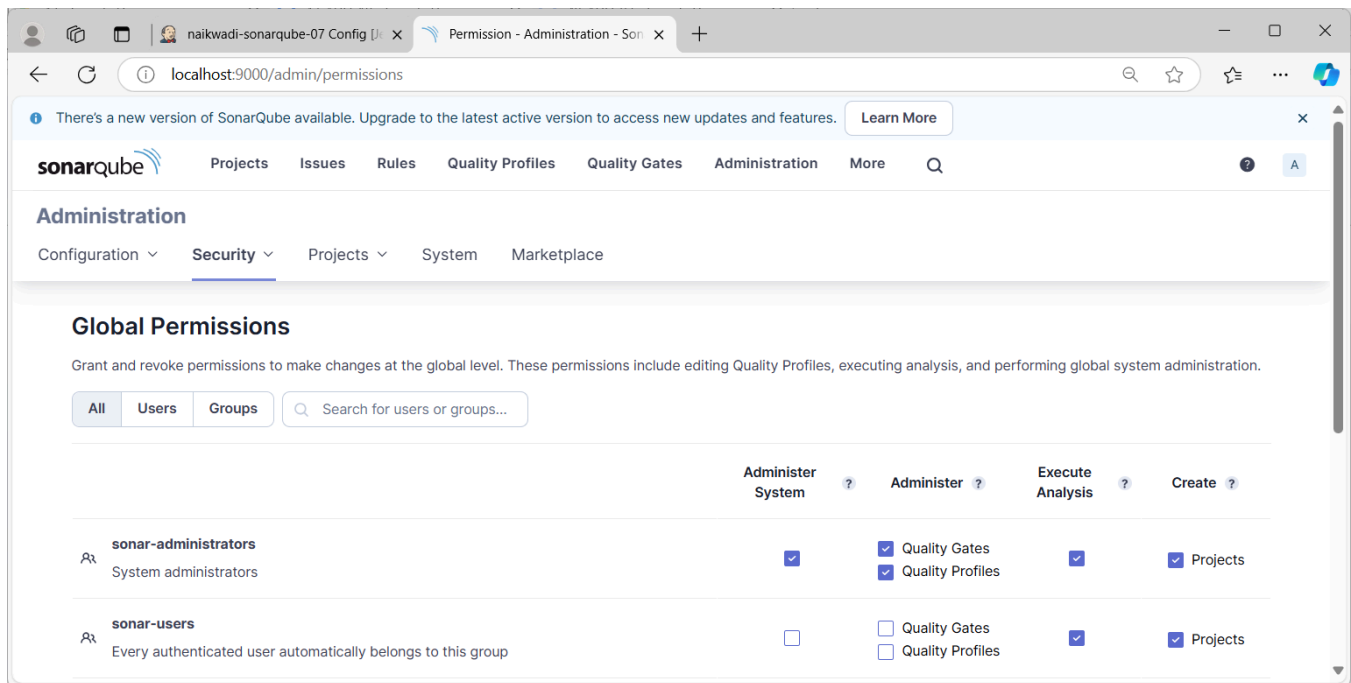
`sonar.password=root`

`sonar.sources=.`

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



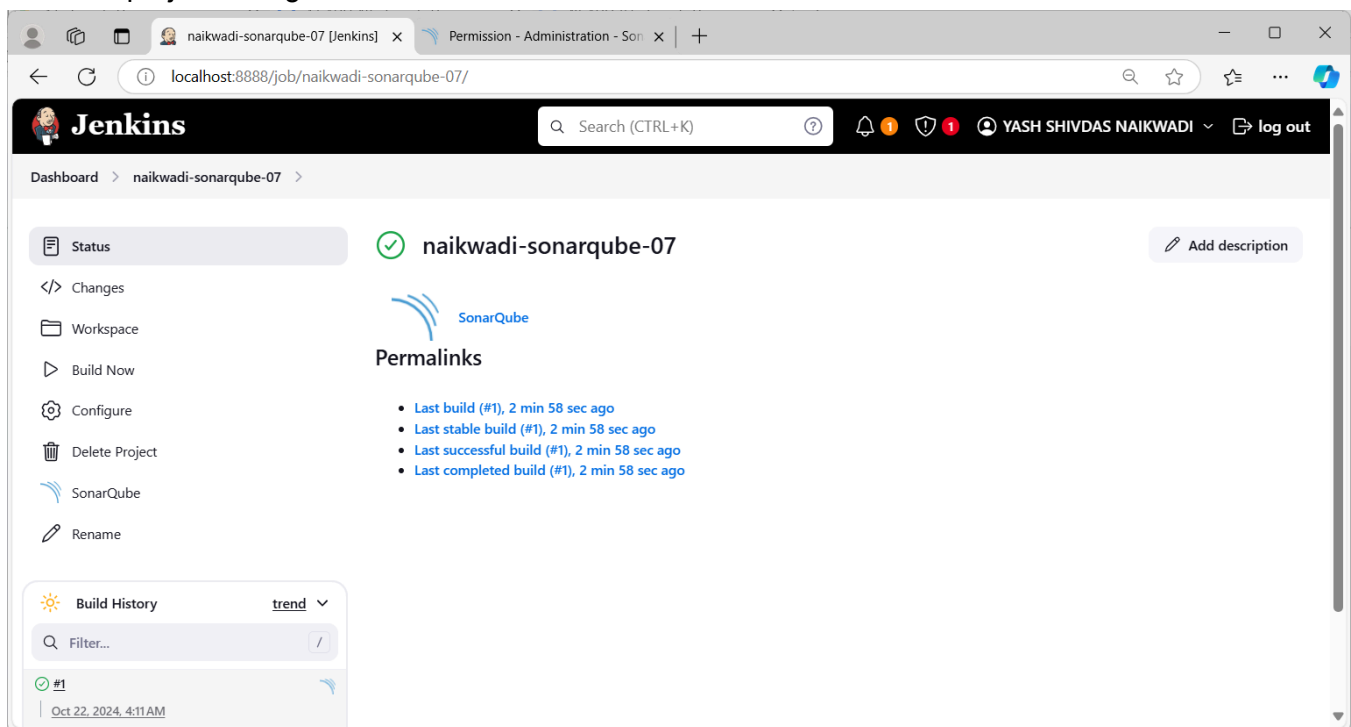
Go to `http://localhost:9000/admin/permissions` and give the Admin user execute permissions.



The screenshot shows the SonarQube Administration interface, specifically the 'Global Permissions' section. The page title is 'Administration' with a sub-tab 'Security'. The main heading is 'Global Permissions', with 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.' Below this, there are tabs for 'All', 'Users', and 'Groups', with a search bar 'Search for users or groups...'. The main content area displays a table of permissions for two groups: 'sonar-administrators' and 'sonar-users'. The 'sonar-administrators' group has permissions for 'Administer System', 'Administer' (Quality Gates, Quality Profiles), 'Execute Analysis', and 'Create' (Projects). The 'sonar-users' group has permissions for 'Administer' (Quality Gates, Quality Profiles), 'Execute Analysis', and 'Create' (Projects).

	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 checked="" 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

Save the project configuration and click Build Now.



The screenshot shows the Jenkins job page for 'naikwadi-sonarqube-07'. The page has a sidebar with links: Status, Changes, Workspace, Build Now, Configure, Delete Project, SonarQube, and Rename. The main content area shows the job name 'naikwadi-sonarqube-07' with a green status icon and a 'Add description' button. Below this, there is a 'Permalinks' section with a list of build links: 'Last build (#1), 2 min 58 sec ago', 'Last stable build (#1), 2 min 58 sec ago', 'Last successful build (#1), 2 min 58 sec ago', and 'Last completed build (#1), 2 min 58 sec ago'. At the bottom, there is a 'Build History' section with a 'trend' dropdown and a 'Filter...' input. The build history shows a single build '#1' with a green status icon and a timestamp 'Oct 22, 2024, 4:11 AM'.

Click on the build number in the build history.

The screenshot shows the Jenkins web interface for a job named 'naikwadi-sonarqube-07'. The build #1 is shown as successful (green checkmark) and completed on Oct 22, 2024, at 4:11:03 AM. The build was started by user YASH SHIVDAS NAIKWADI. The console output shows the build process, including cloning the repository from https://github.com/shazforiot/MSBuild_firstproject.git and running the analysis. The build took 1 min 53 sec.

Click on Console Output to see the build logs.

The screenshot shows the Jenkins web interface for the same job, but with the 'Console Output' tab selected. The output shows the build process, including cloning the repository from https://github.com/shazforiot/MSBuild_firstproject.git and running the analysis. The build took 1 min 53 sec.

The screenshot shows the Jenkins web interface for the same job, but with the 'Console Output' tab selected. The output shows the build process, including cloning the repository from https://github.com/shazforiot/MSBuild_firstproject.git and running the analysis. The build took 1 min 53 sec.

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

The screenshot displays the SonarQube web application interface. At the top, there's a navigation bar with tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main content area shows the 'naikwadi-07' project overview. A prominent green checkmark and the word 'Passed' indicate that the quality gate has been successfully completed. Below this, a warning message states 'The last analysis has warnings. See details'. The interface also features a 'New Code' and 'Overall Code' tab, with 'Overall Code' currently selected. The dashboard below the tabs provides a comprehensive overview of the project's quality metrics, including Security, Reliability, Maintainability, Accepted issues, Coverage, and Duplications, all of which are currently at '0' or '0.0%' with a green 'A' grade.

Metric	Value	Grade
Security	0 Open Issues	A
Reliability	0 Open Issues	A
Maintainability	0 Open Issues	A
Accepted issues	0	
Coverage	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.