



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.	53
Name	Aryan Deepak Saraf
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.

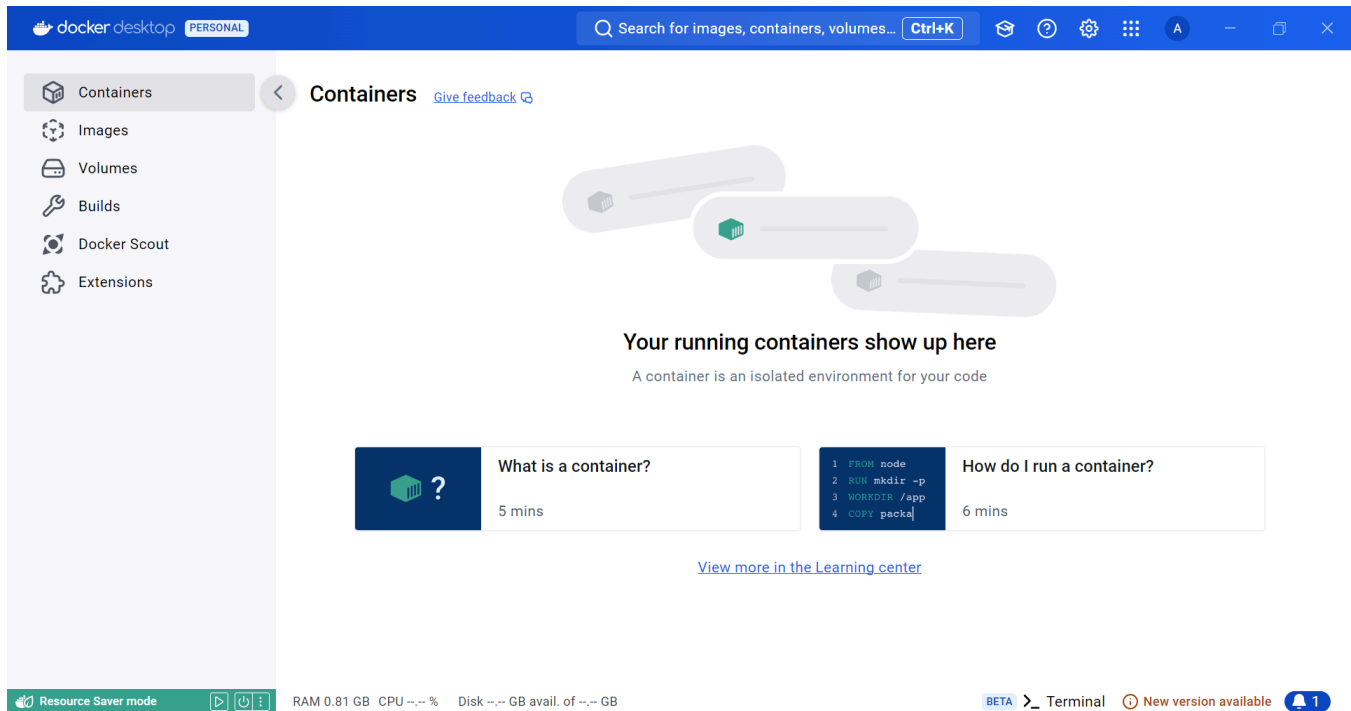
Why is SAST Important?

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`

```
Command Prompt
C:\Users\Admin>docker pull sonarqube: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

C:\Users\Admin>echo aryan
aryan

C:\Users\Admin>
```

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
nginx          latest    39286ab8a5e1   5 weeks ago    188MB
sonarqube      latest    2433ac783140   2 months ago   1.07GB
mysql          latest    680b8c60dce6   2 months ago   586MB
python         3.9-slim-buster c84dbfe3b8de   15 months ago  116MB

C:\Users\Admin>echo aryan
aryan

C:\Users\Admin>
```

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

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

```
Command Prompt
C:\Users\Admin>echo aryan
aryan

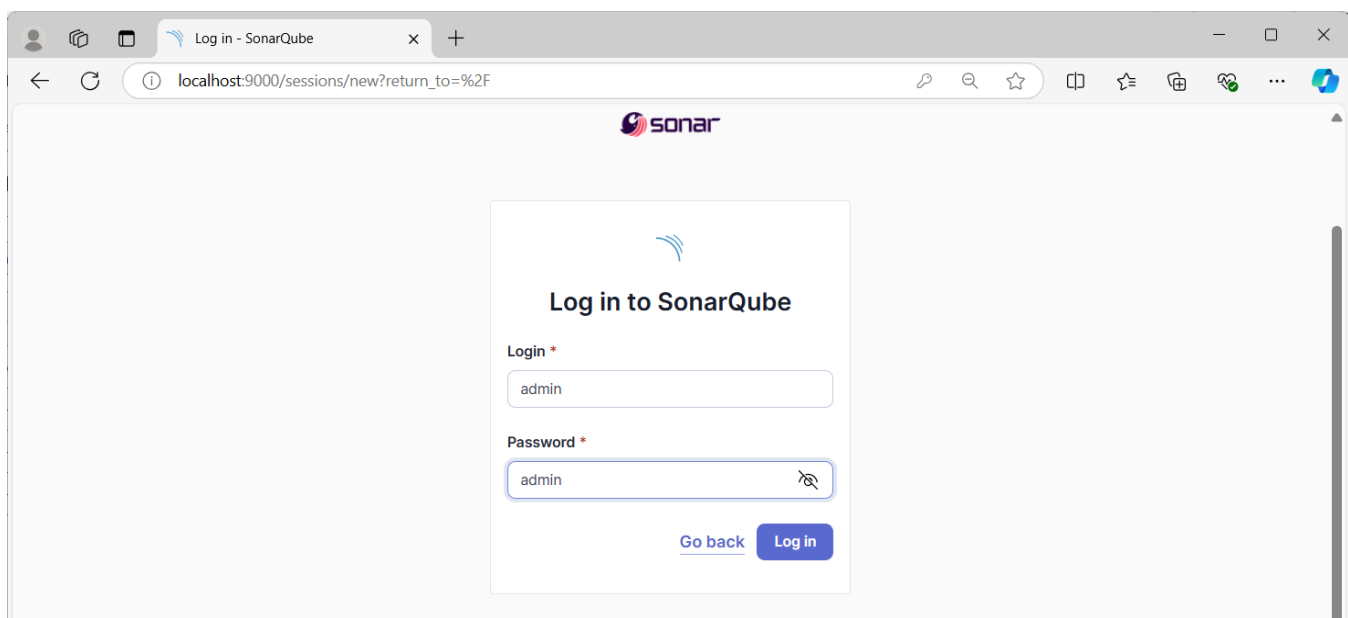
C:\Users\Admin>docker run -d --name aryan-sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
fb67be4680e39e74c8a70cc1a4ec4031e114319f478d90a6356c6c68cade637d

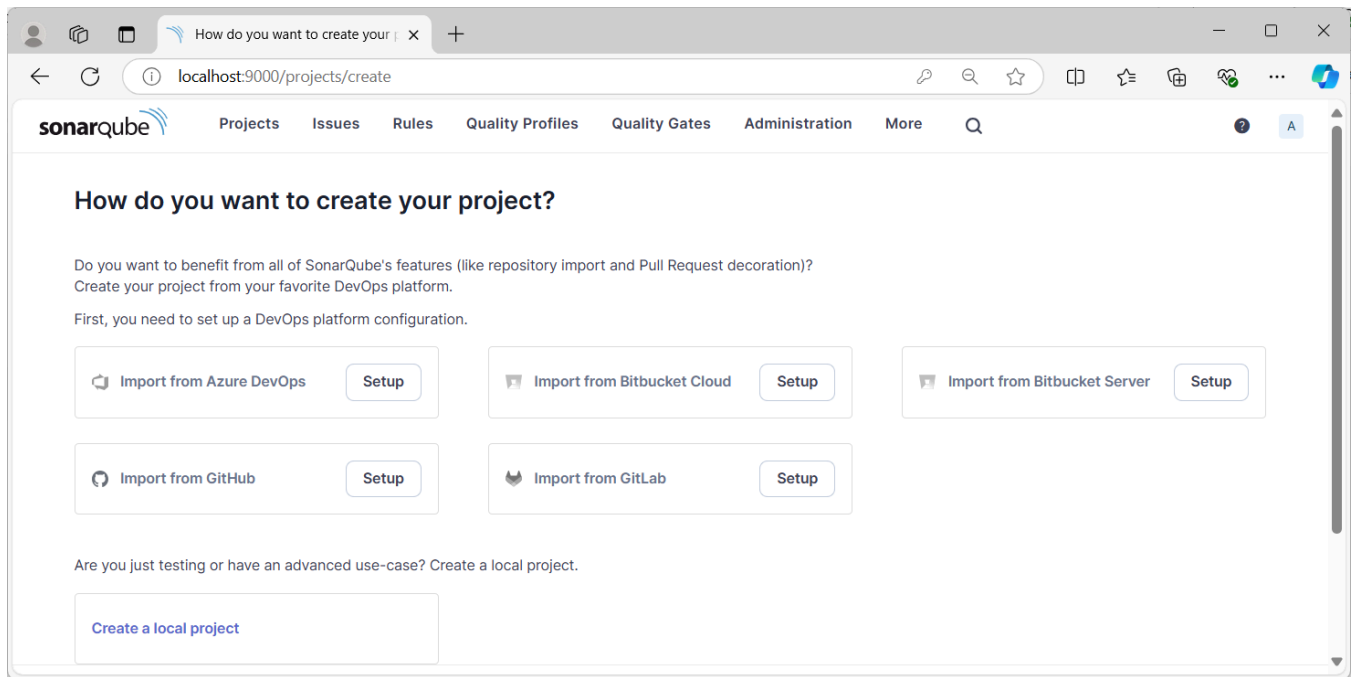
C:\Users\Admin>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
fb67be4680e3   sonarqube:latest "/opt/sonarqube/dock...  19 seconds ago Up 16 seconds  0.0.0.0:9000->9000/tcp   aryan-sonarqube

C:\Users\Admin>
```

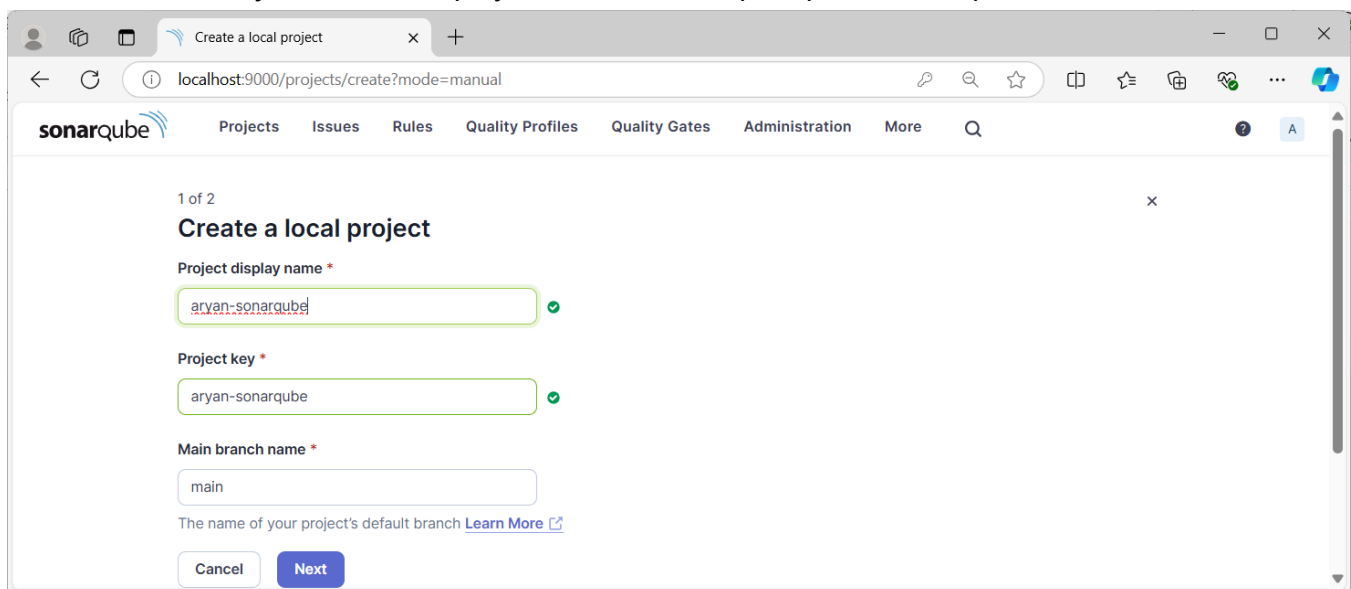
Login to SonarQube: Use the default credentials:

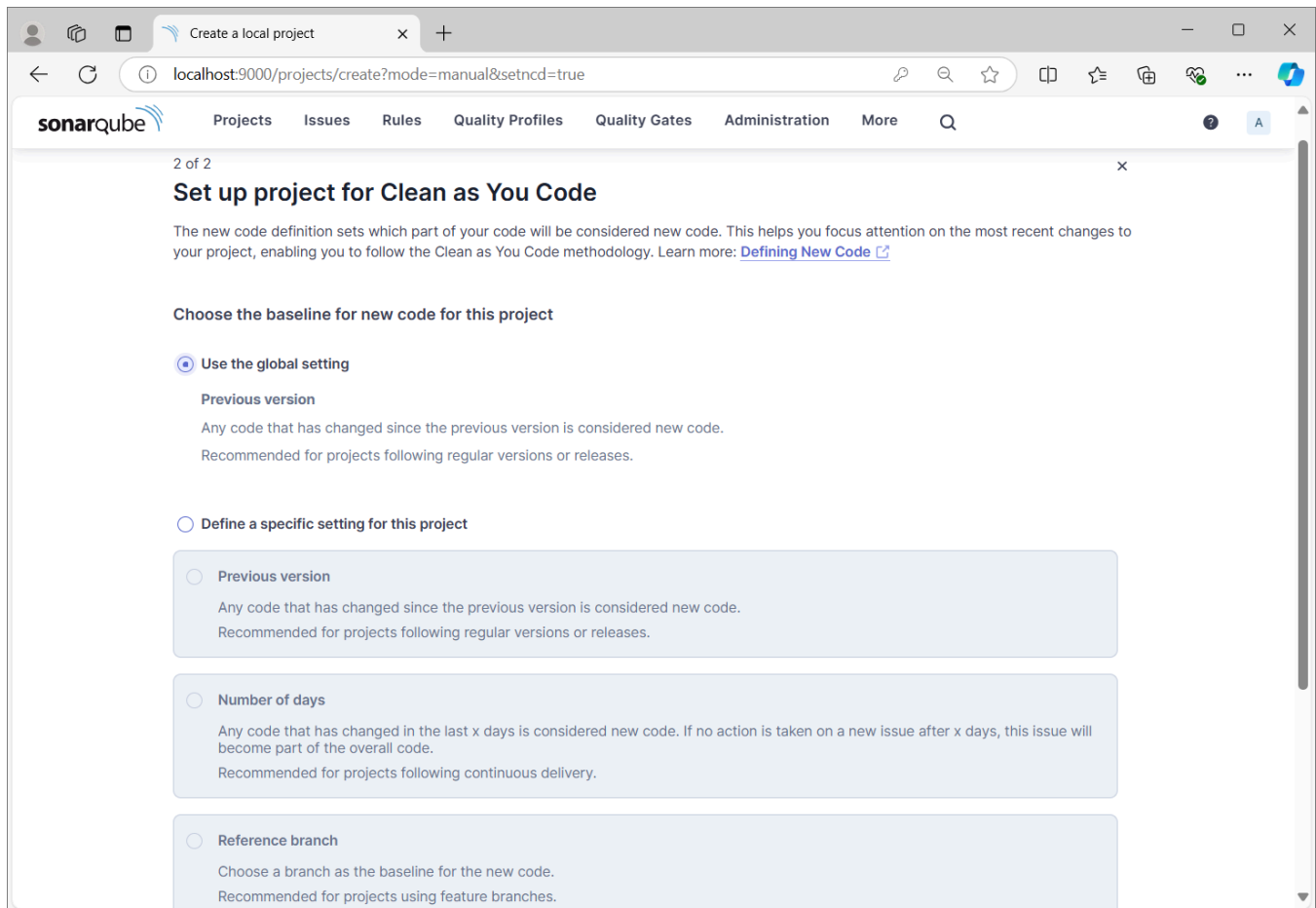
- Username: admin
- Password: admin



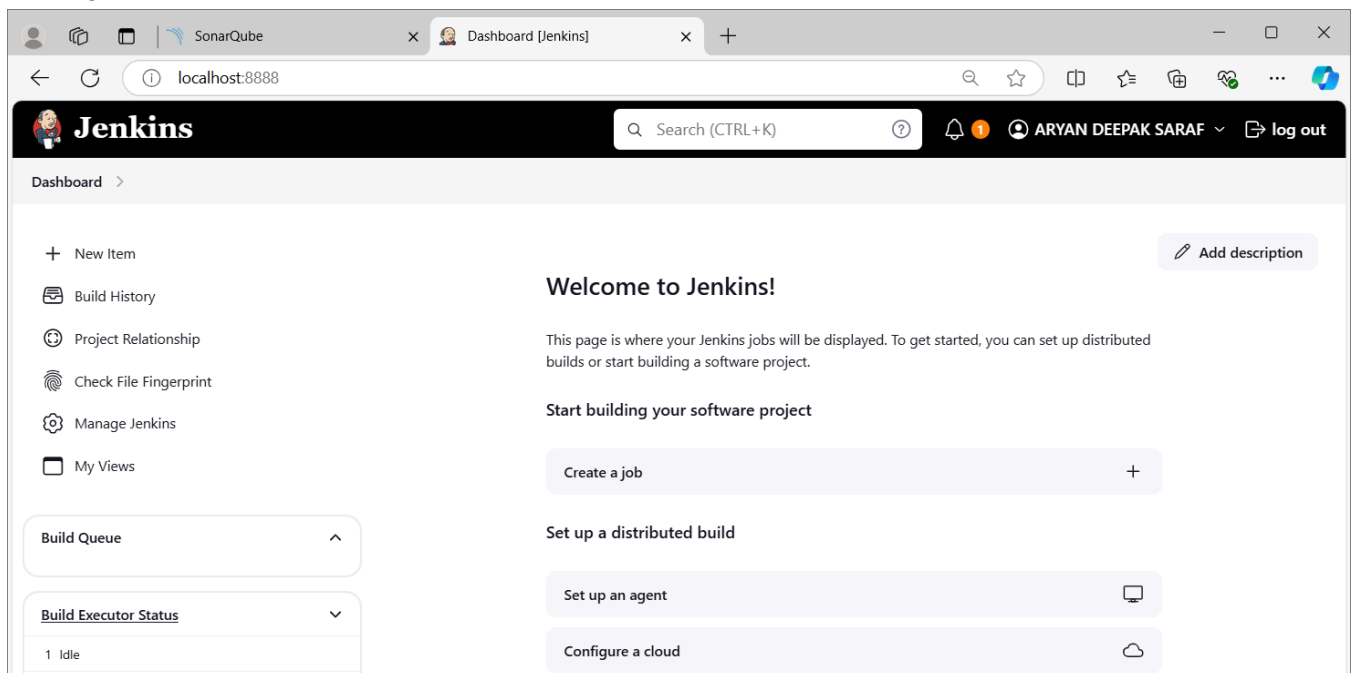


Click on Create Project. Name the project and follow the prompts to set it up.





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



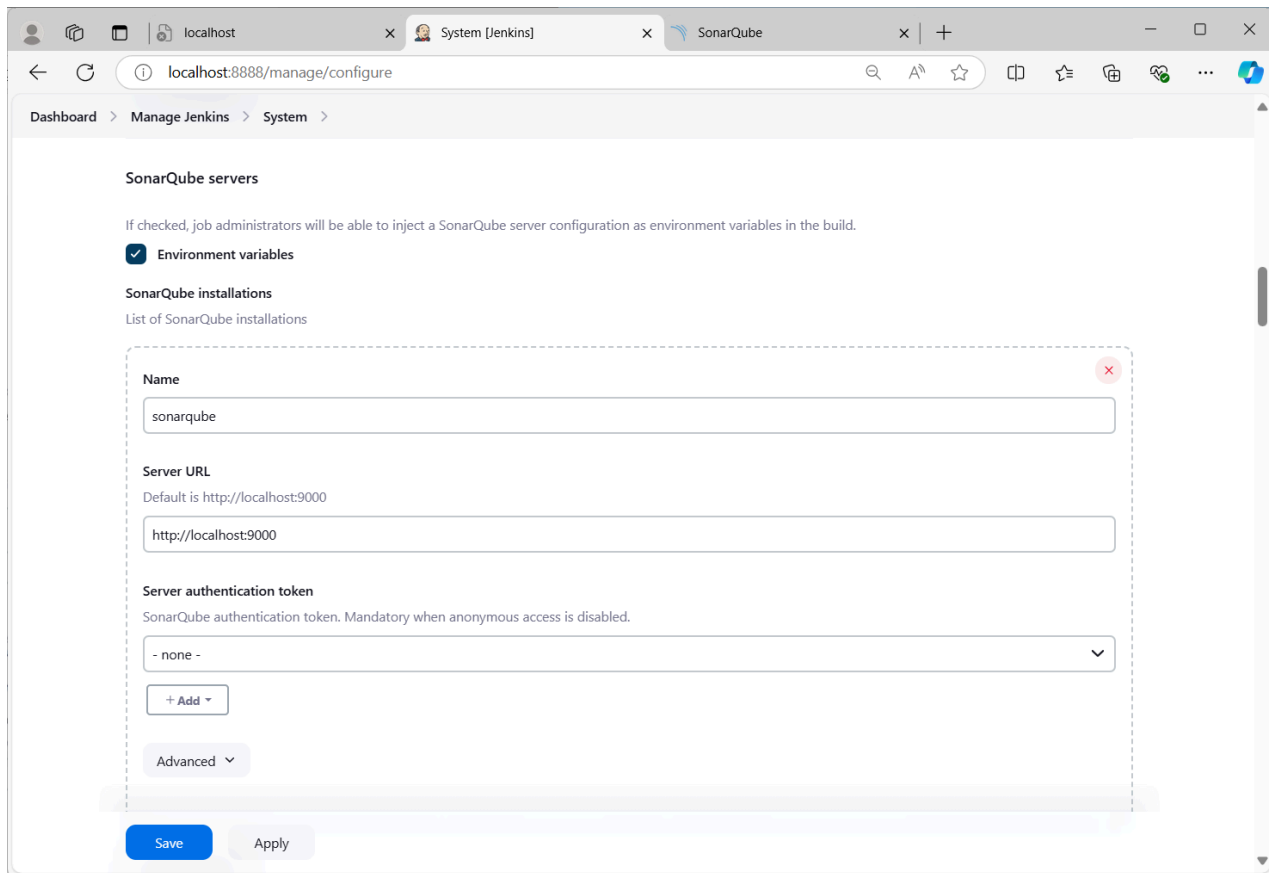
Manage Jenkins → Manage Plugins → Available tab, search for SonarQube Scanner. Check the box next to it and click Install without Restart.

The first screenshot shows the Jenkins 'Available plugins' page. The search bar contains 'SONARQUBE SCANNER'. The 'SonarQube Scanner' plugin (version 2.17.2) is listed with an 'Install' button. The description states: 'This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.' The release date is '7 mo 9 days ago'.

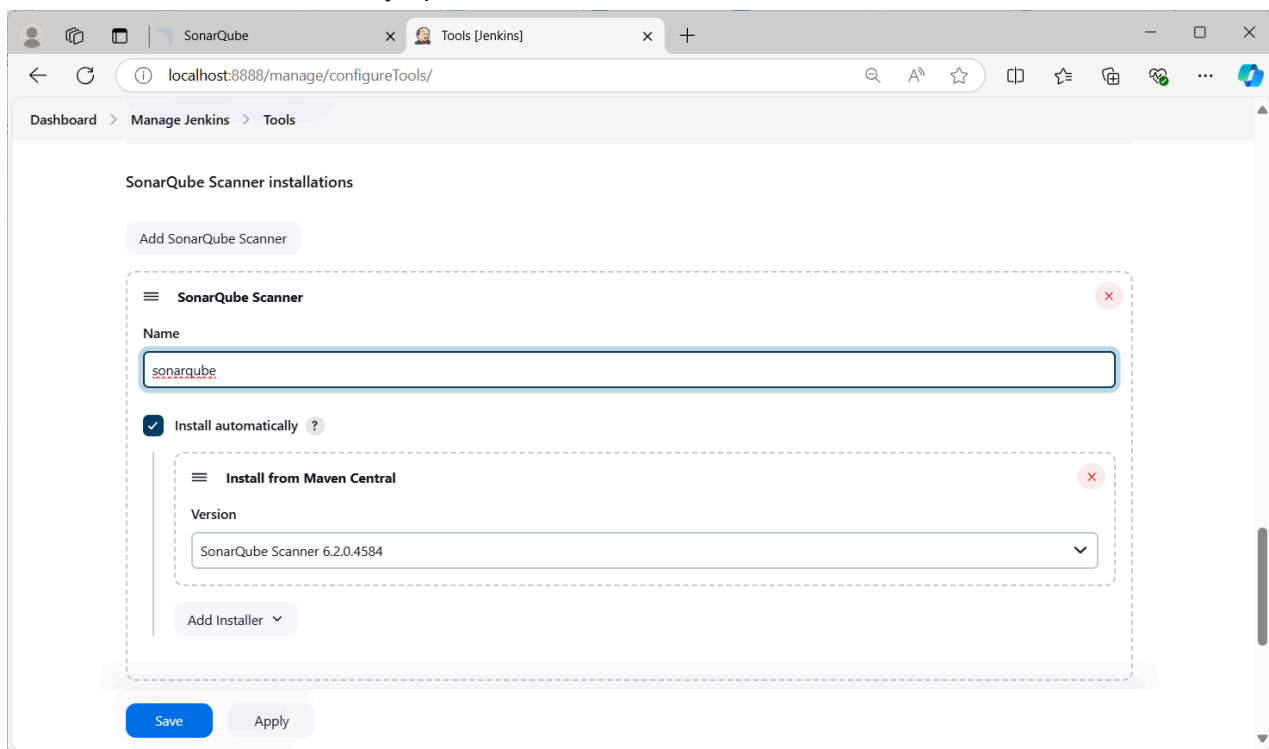
The second screenshot shows the 'Download progress' page for the SonarQube Scanner plugin. The progress is 100% complete. The steps shown are: Preparation (Checking internet connectivity, Checking update center connectivity, Success), SonarQube Scanner (Success), and Loading plugin extensions (Success). A link to 'Go back to the top page' is provided, along with a checkbox to '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.

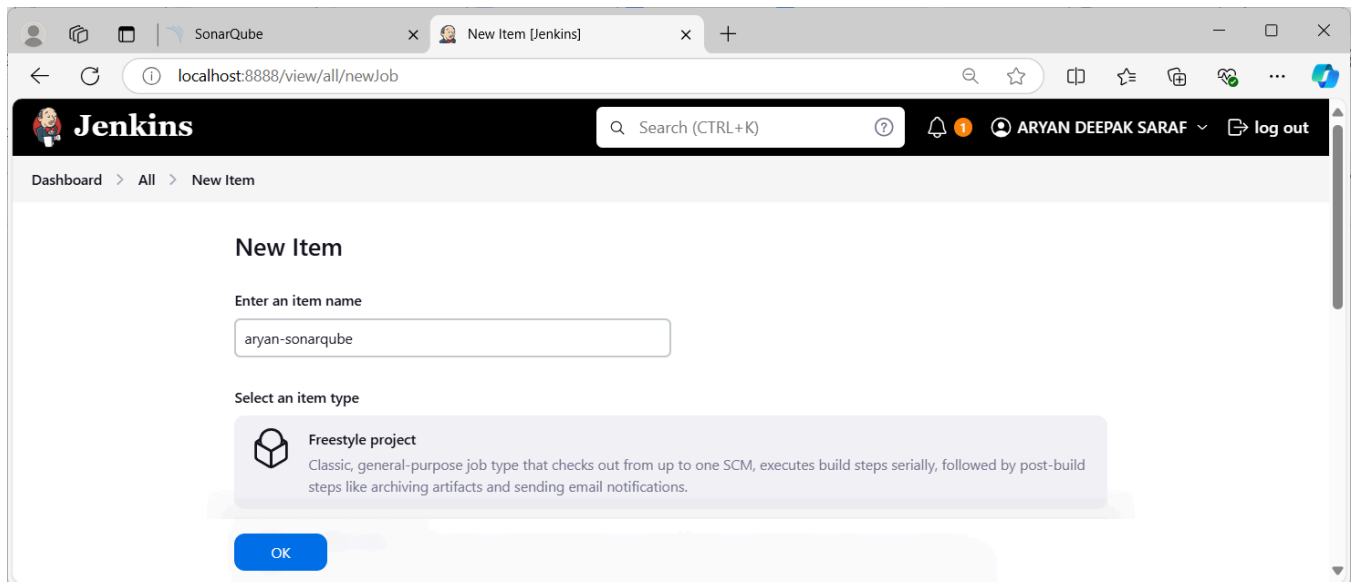
- Name: Give it a name (e.g., SonarQube).
- Server URL: Enter `http://localhost:9000`.



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

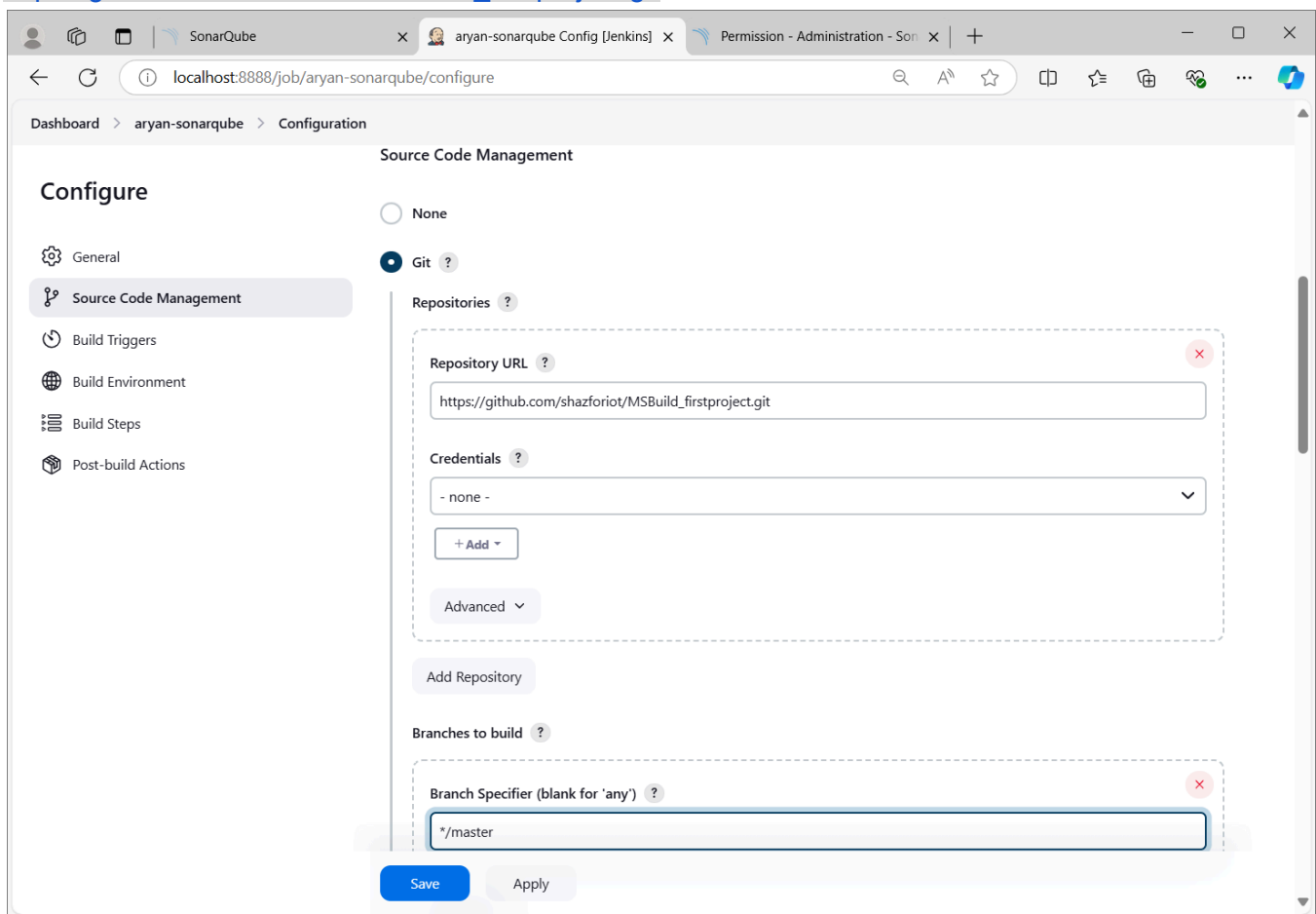


On the Jenkins dashboard, click on New Item. Enter a name for your project. Choose the Freestyle project and click OK.

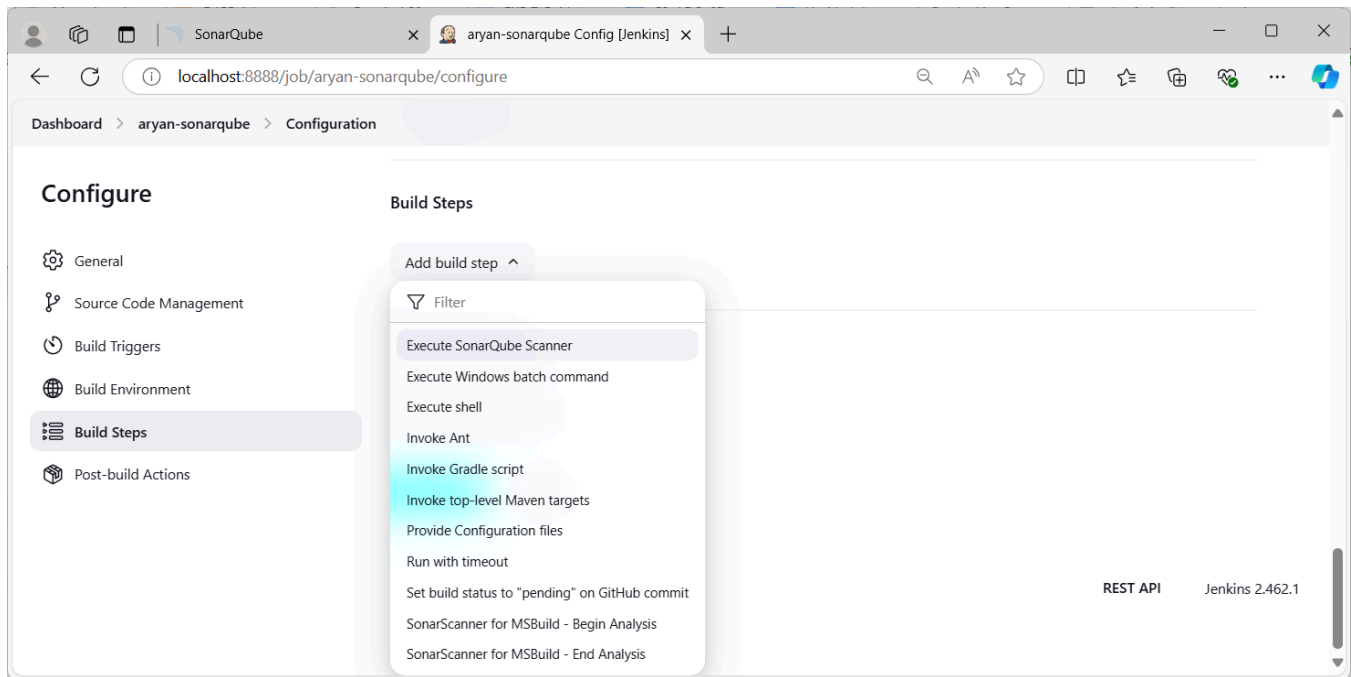


In the project configuration, look for the Source Code Management section. Choose Git and enter the repository URL.

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

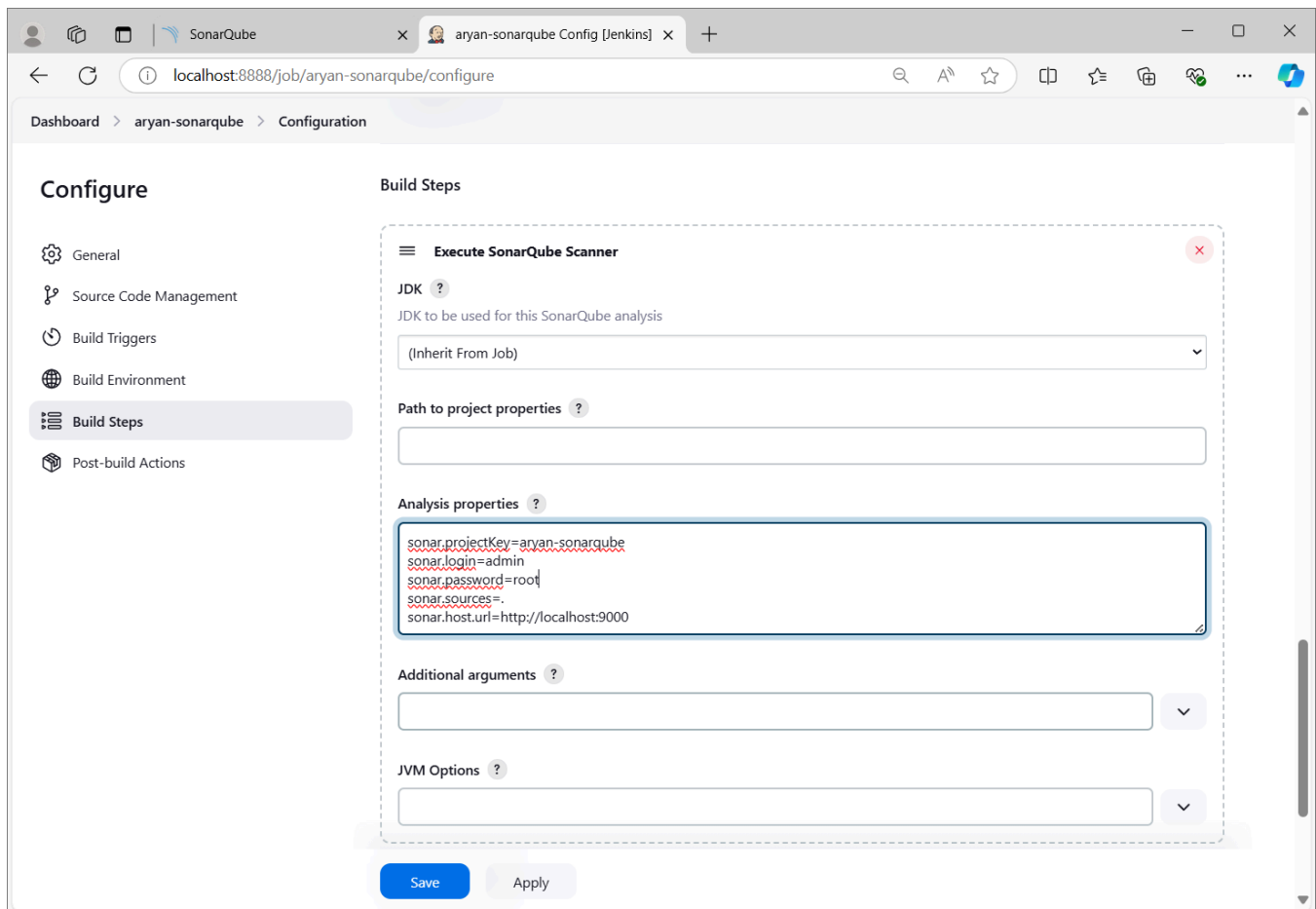
sonar.projectKey=sonarqube

sonar.login=admin

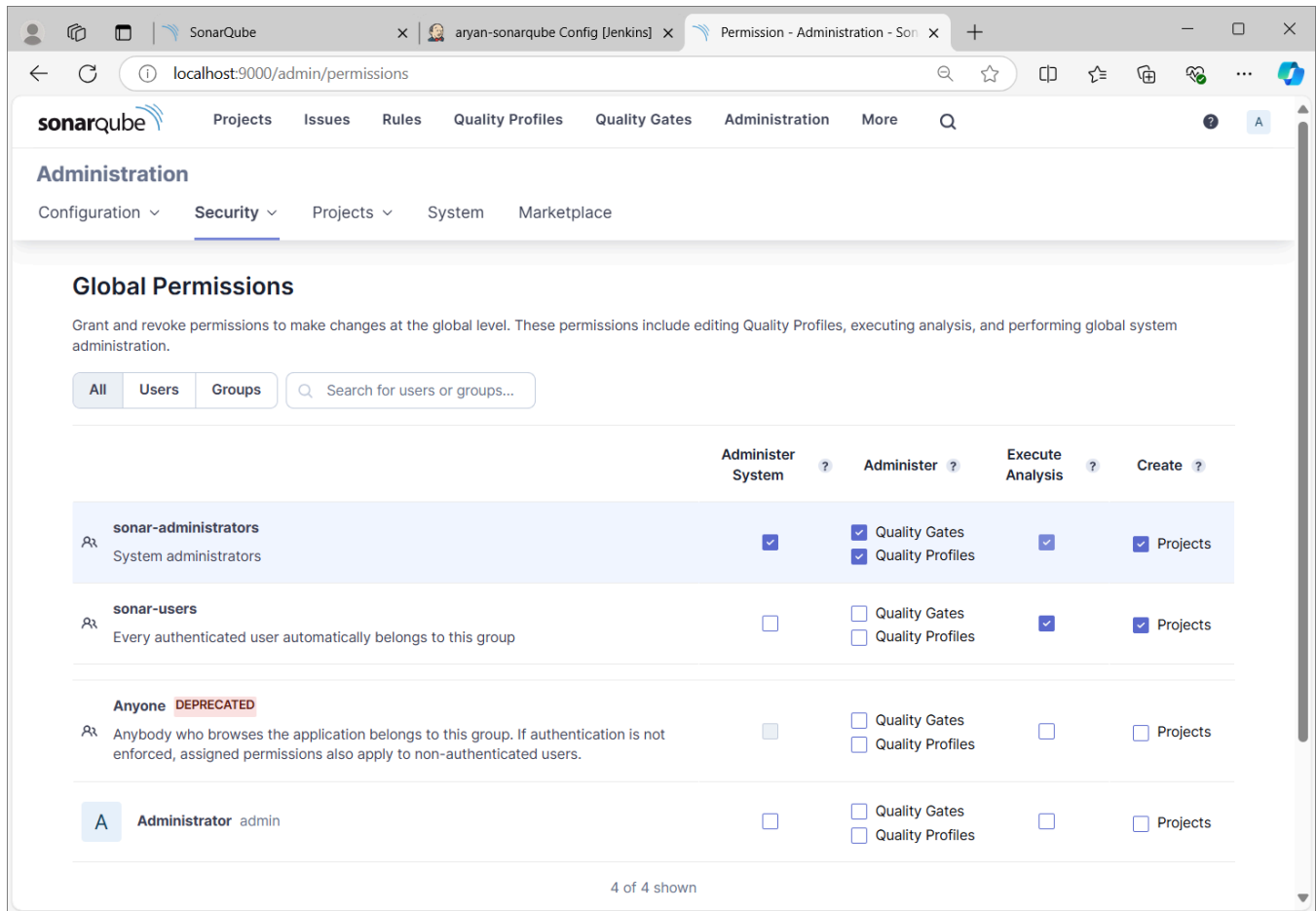
sonar.password=admin

sonar.sources=.

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



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

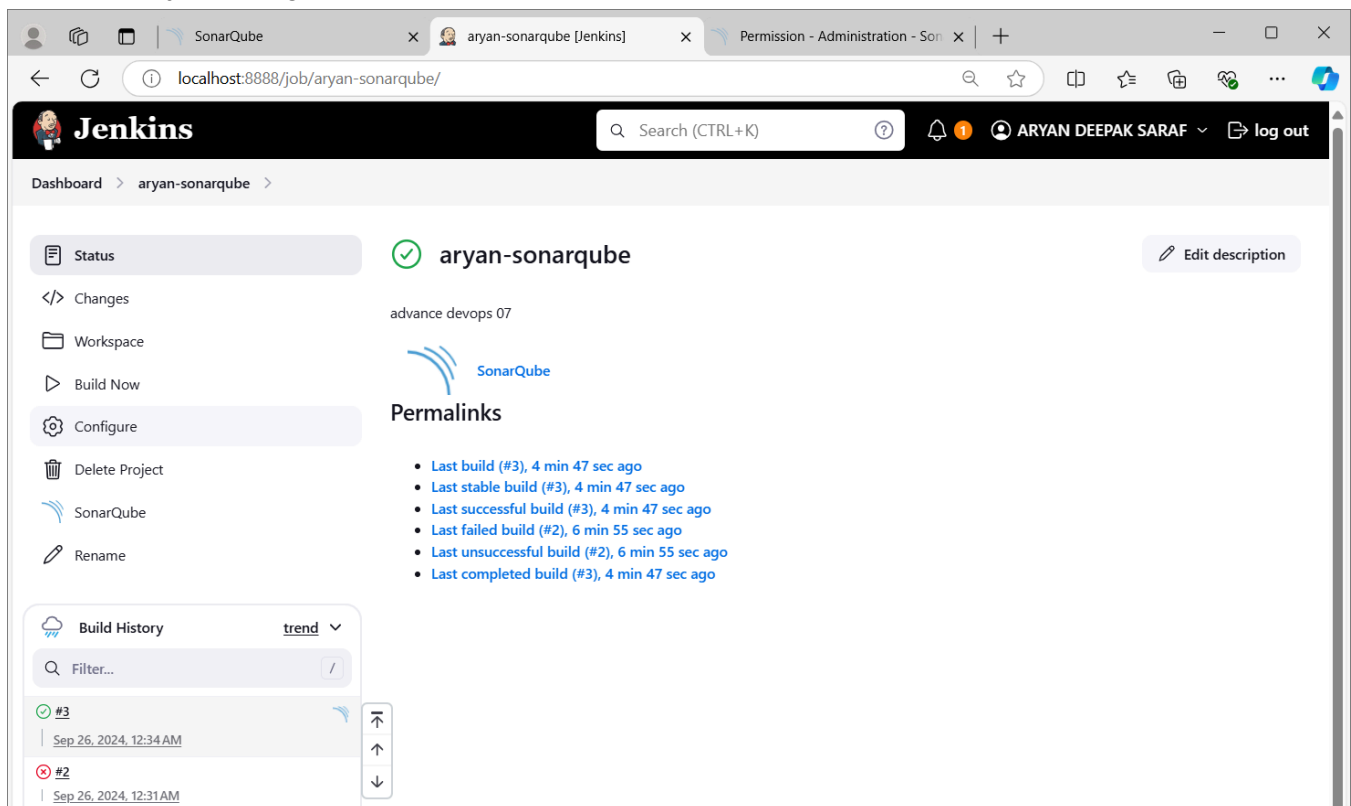


The screenshot shows the SonarQube Administration interface, specifically the 'Global Permissions' section under 'Security'. The page title is 'Administration' with a sub-header 'Global Permissions'. Below the title, there is a description: 'Grant and revoke permissions to make changes at the global level. These permissions include editing Quality Profiles, executing analysis, and performing global system administration.' There are tabs for 'All', 'Users', and 'Groups', with a search bar for 'Search for users or groups...'. The main table lists permissions for four groups: 'sonar-administrators', 'sonar-users', 'Anyone DEPRECATED', and 'Administrator admin'. The columns are 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The 'Administrator admin' row shows checkboxes for 'Quality Gates', 'Quality Profiles', 'Execute Analysis', and 'Create'.

	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
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
Administrator admin	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input type="checkbox"/> Projects

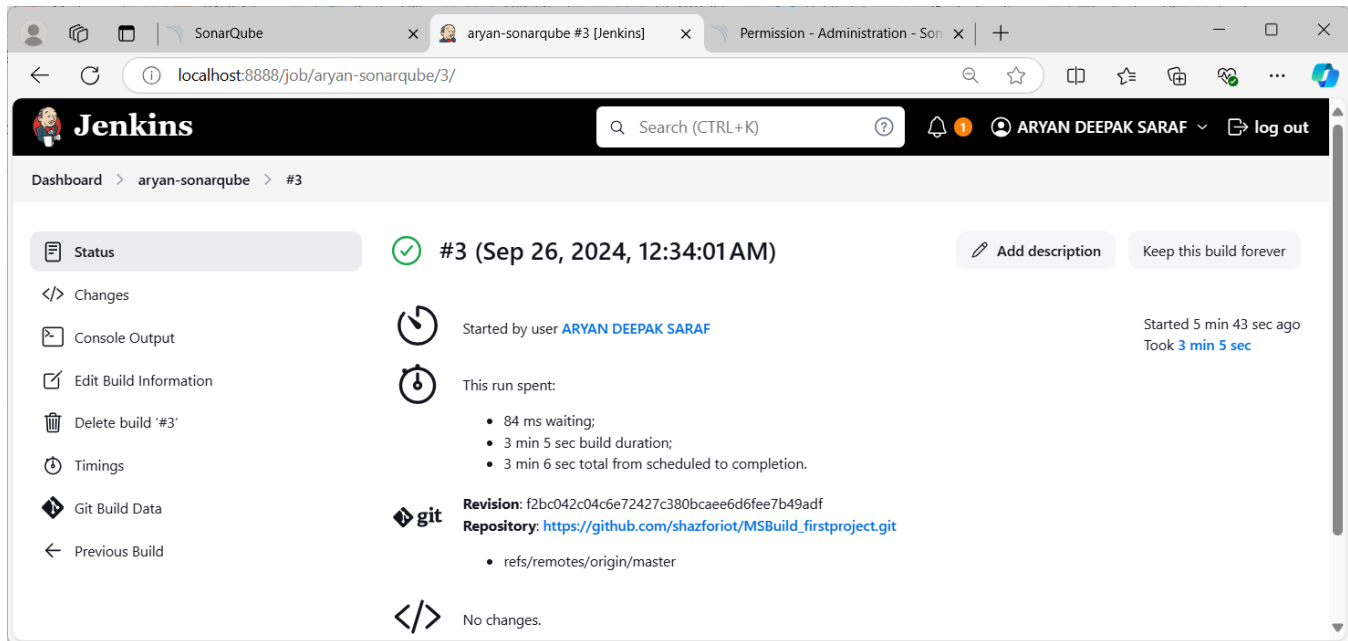
4 of 4 shown

Save the project configuration and click Build Now.



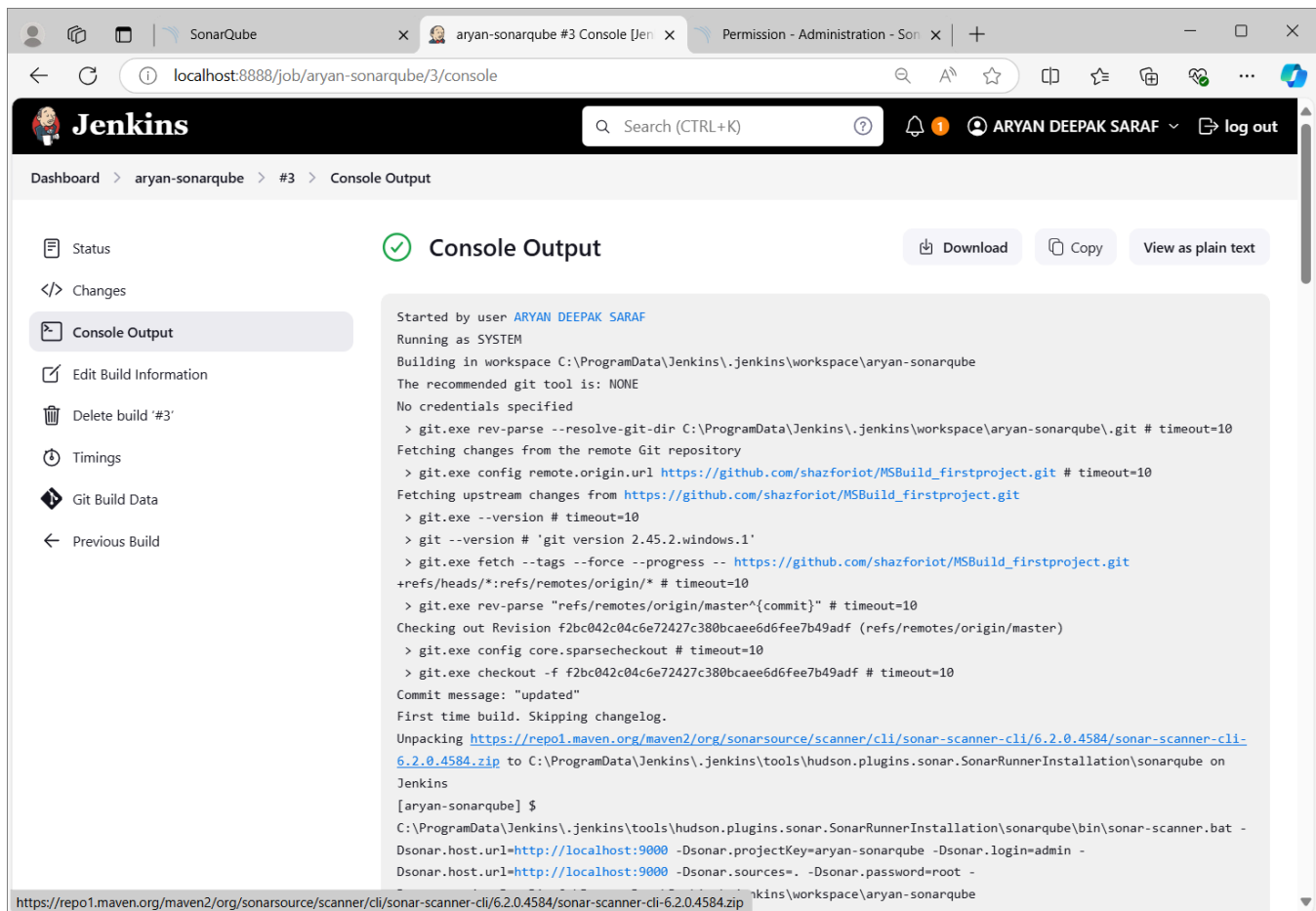
The screenshot shows the Jenkins Job Configuration page for the job 'aryan-sonarqube'. The page has a sidebar with navigation links: 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', 'SonarQube', and 'Rename'. The main content area shows the job name 'aryan-sonarqube' with a green checkmark, a description 'advance devops 07', and the SonarQube logo. Below this, there is a 'Permalinks' section with a list of build links: 'Last build (#3), 4 min 47 sec ago', 'Last stable build (#3), 4 min 47 sec ago', 'Last successful build (#3), 4 min 47 sec ago', 'Last failed build (#2), 6 min 55 sec ago', 'Last unsuccessful build (#2), 6 min 55 sec ago', and 'Last completed build (#3), 4 min 47 sec ago'. At the bottom, there is a 'Build History' section with a search bar and a list of builds: '#3' (Sep 26, 2024, 12:34 AM) and '#2' (Sep 26, 2024, 12:31 AM).

Click on the build number in the build history.



The screenshot shows the Jenkins web interface for build #3 of the 'aryan-sonarqube' job. The page title is 'Jenkins' and the URL is 'localhost:8888/job/aryan-sonarqube/3/'. The build status is 'Success' (green checkmark) and the build number is '#3 (Sep 26, 2024, 12:34:01 AM)'. The build was started by user 'ARYAN DEEPAK SARAF' and took 3 min 5 sec to complete. The console output shows the build process, including fetching changes from the remote Git repository and checking out the revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf. The build is successful and no changes were detected.

Click on Console Output to see the build logs.



The screenshot shows the Jenkins web interface for the console output of build #3. The page title is 'Jenkins' and the URL is 'localhost:8888/job/aryan-sonarqube/3/console'. The console output shows the build process, including fetching changes from the remote Git repository and checking out the revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf. The build is successful and no changes were detected. The console output is displayed in a text area with a download button and a copy button.

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

The screenshot displays the SonarQube web interface for a project named 'aryan-sonarqube'. The browser address bar shows the URL 'localhost:9000/dashboard?id=aryan-sonarqube&codeScope=overall'. The interface includes a navigation bar with links for 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. The main content area shows the 'main' branch with a 'Passed' status, indicated by a green checkmark. A warning message states 'The last analysis has warnings. See details'. The 'Overall Code' tab is selected, displaying a dashboard with metrics for Security, Reliability, Maintainability, Accepted issues, Coverage, and Duplications. All metrics show '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	A
Coverage	On 0 lines to cover.	A
Duplications	0.0% On 86 lines.	A

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.