

Experiment 7

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

Steps:

Step 1: Install a SonarQube image by running the 'docker pull sonarqube' command on your terminal. This allows for a Sonarqube image to be used on a local machine without having to install the SonarQube application.

```
PS C:\Users\anish\OneDrive\Desktop\Adv DevOps 7> 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
```

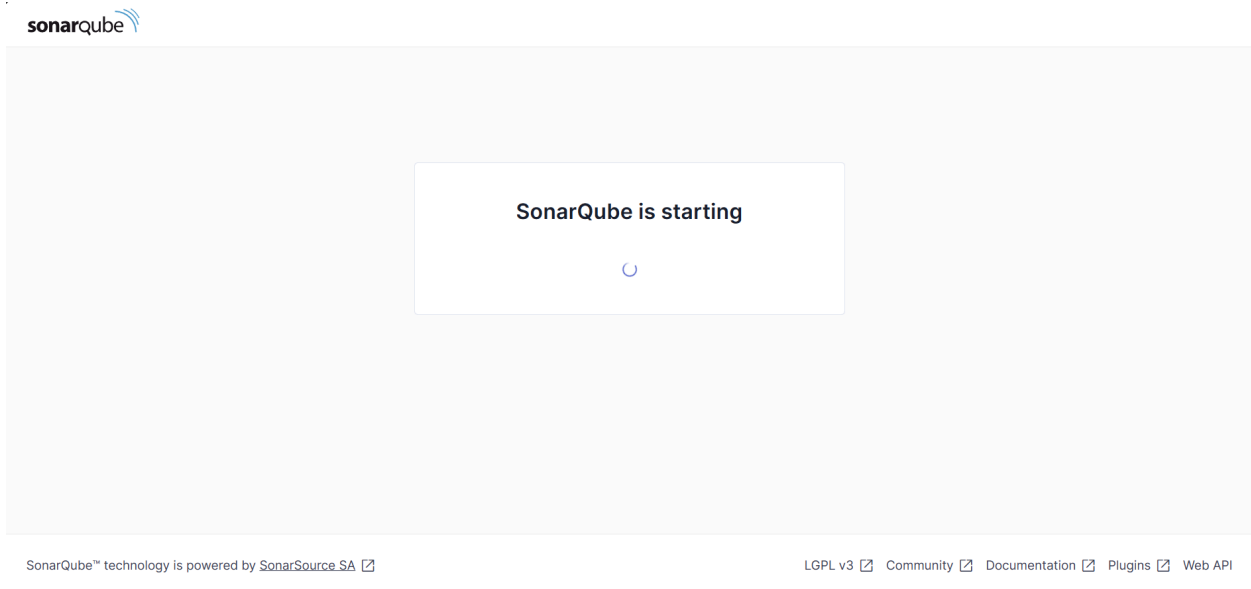
Step 2: Execute the following command:

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

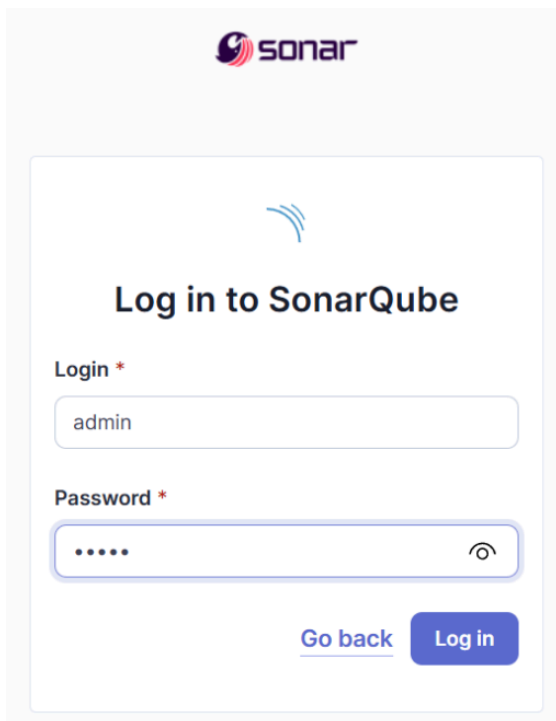
This command will run the SonarQube image that was just installed using docker.

```
PS C:\Users\anish\OneDrive\Desktop\Adv DevOps 7> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
dce67335909e42d81ec64d3ef0c5e5e2c36cc7ed36d87088033121ae1544f4fb
```

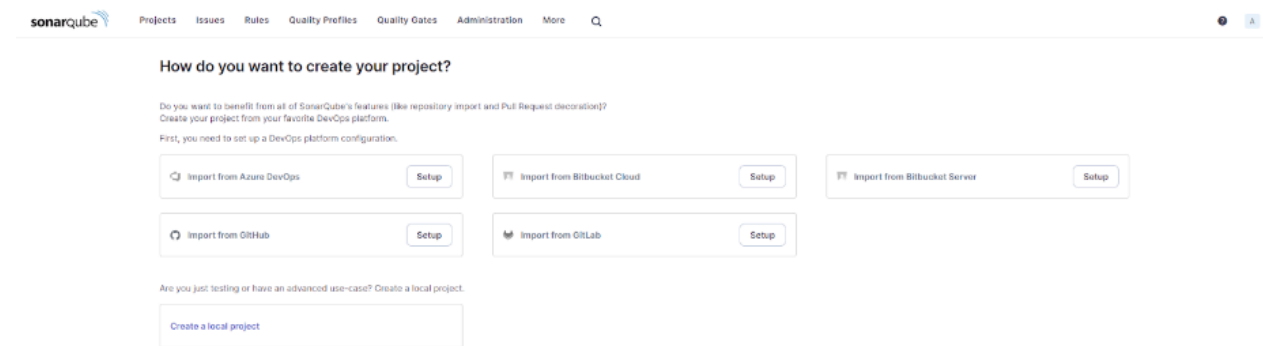
Step 3: Go to <http://localhost:9000> on your browser and check if SonarQube is starting or not.



Step 4: On the login page, enter 'Login' as admin and 'Password' as admin to log in initially. It then asks you to change the password to a password of your choice. Do the same and proceed to the next step.



Step 5: On the SonarQube dashboard, click on 'Create a local project'.



Step 6: Create a local project by entering the project name and key and click on 'Next'.

1 of 2

Create a local project

Project display name *

sonarqubetest29 ✓

Project key *

sonarqubetest29 ✓

Main branch name *

main

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

Cancel Next

Step 7: Set up your project and click on 'Create project'.

2 of 2

X

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

☒ Use the global setting

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

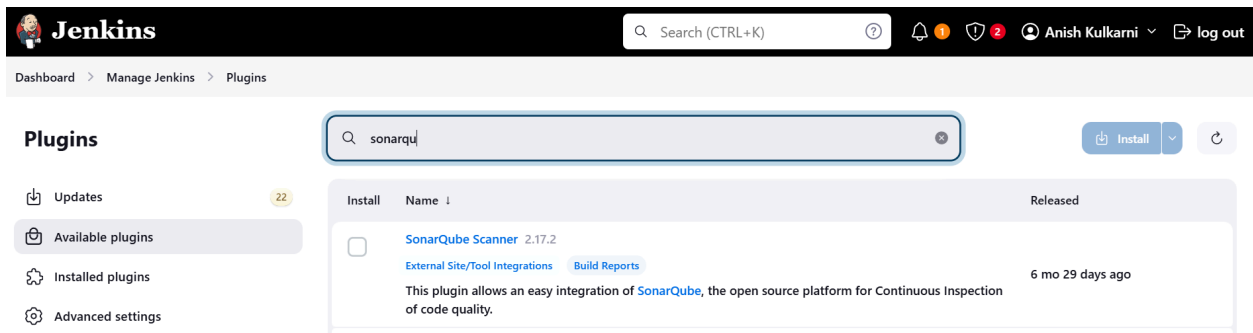
☐ Previous version

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

☐ Number of days

Step 8: Navigate to your Jenkins server (on whichever port it has been installed), click on 'Manage Jenkins', click on 'Plugins' and search for the 'SonarQube Scanner' plugin and install it.



The screenshot shows the Jenkins web interface. At the top, there's a header with the Jenkins logo, a search bar, and user information. Below the header, the breadcrumb trail is 'Dashboard > Manage Jenkins > Plugins'. The main section is titled 'Plugins' and features a search bar with 'sonarqu' entered. To the left of the search results are navigation links: 'Updates' (with a badge '22'), 'Available plugins', 'Installed plugins', and 'Advanced settings'. The search results table has columns 'Install', 'Name', and 'Released'. It lists the 'SonarQube Scanner' version 2.17.2, which is currently not installed. 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 '6 mo 29 days ago'.

Step 9: Under 'Manage Jenkins', click on System. Under the 'Sonarqube installations' section, add a server and add a server authentication token if needed.

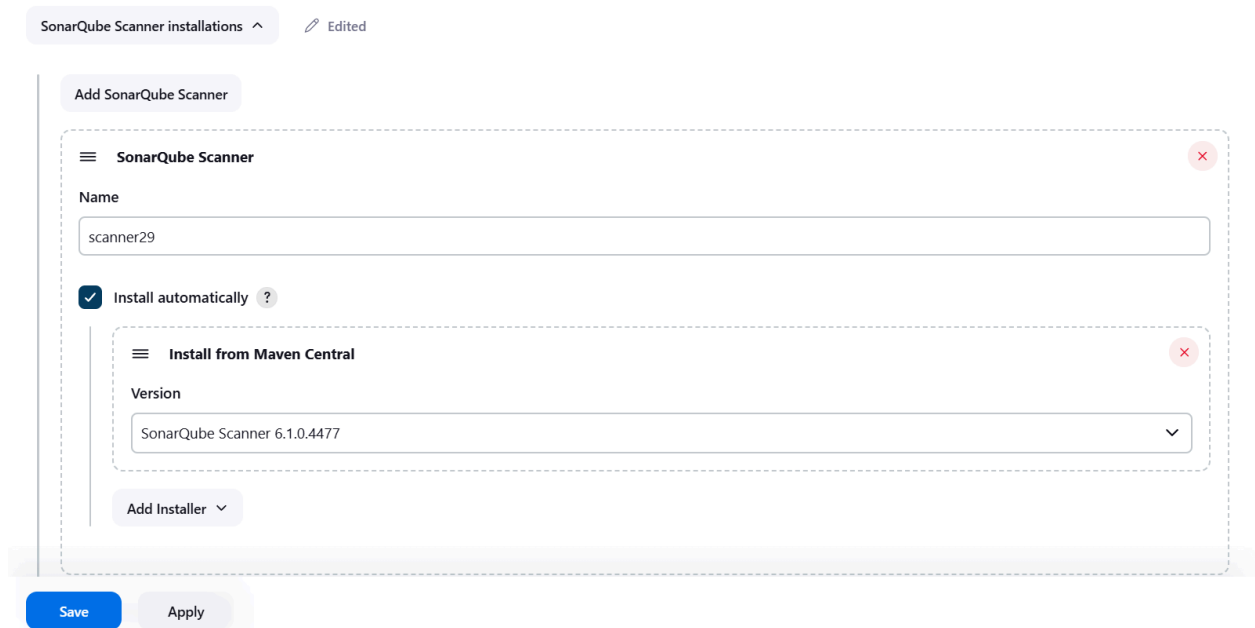


The screenshot shows the 'SonarQube installations' configuration page. It has a title 'SonarQube installations' and a subtitle 'List of SonarQube installations'. The form is enclosed in a dashed border and contains the following fields:

- Name:** A text input field containing 'sonarqube29'.
- Server URL:** A text input field containing 'http://localhost:9000'. Below the field, it says 'Default is http://localhost:9000'.
- Server authentication token:** A dropdown menu with the selected option '- none -'. Below the dropdown, it says 'SonarQube authentication token. Mandatory when anonymous access is disabled.'

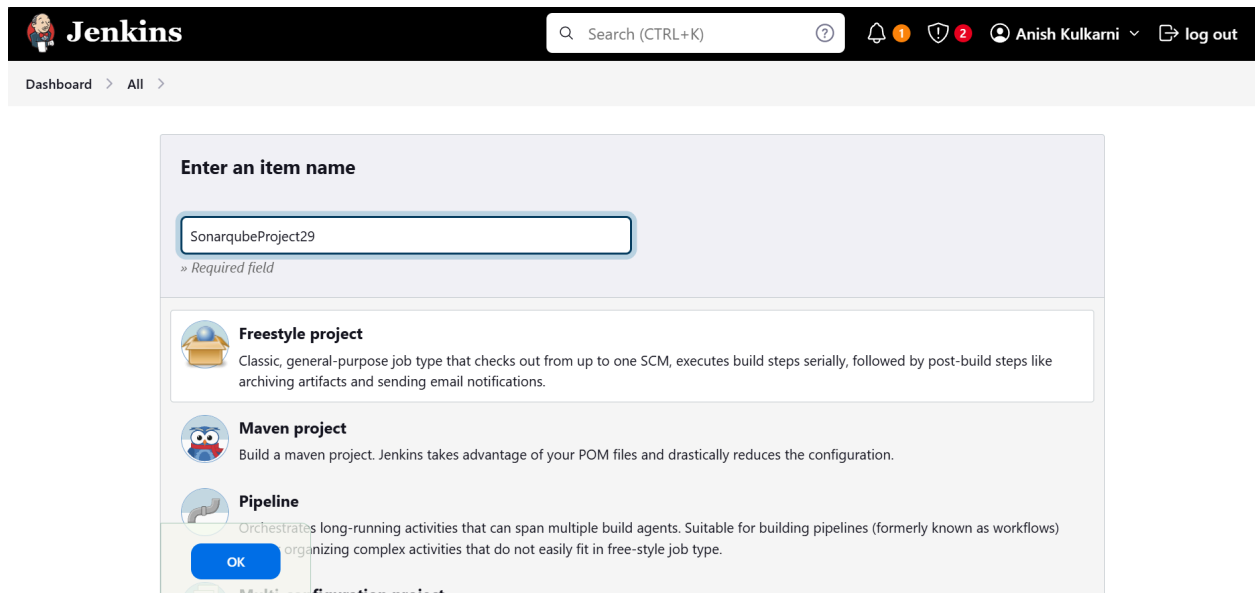
At the bottom of the form is a '+ Add' button. Below the form, there are two buttons: 'Save' and 'Apply'.

Step 10: Under 'Manage Jenkins', click on 'Tools'. Under the 'SonarQube Scanner installations' section, give your scanner a name, choose the latest version and click on 'Install automatically'.



The screenshot shows the 'SonarQube Scanner installations' configuration page in Jenkins. At the top, there's a header 'SonarQube Scanner installations' with a dropdown arrow and an 'Edited' status. Below this is a section titled 'Add SonarQube Scanner'. Inside this section, there's a sub-section 'SonarQube Scanner' with a red 'X' icon in the top right corner. This sub-section contains a 'Name' field with the value 'scanner29', a checked 'Install automatically' checkbox with a help icon, and an 'Install from Maven Central' sub-section. The 'Install from Maven Central' sub-section has a 'Version' dropdown menu showing 'SonarQube Scanner 6.1.0.4477'. Below these fields is an 'Add Installer' button. At the bottom of the main configuration area are 'Save' and 'Apply' buttons.

Step 11: Create a new Jenkins project by giving it a name and ensure that it is a freestyle project.



The screenshot shows the 'Enter an item name' dialog in Jenkins. The dialog has a header 'Enter an item name' and a text input field containing 'SonarqubeProject29'. Below the input field is a note '» Required field'. Below the input field, there are three project type options: 'Freestyle project' (with a box icon), 'Maven project' (with a robot icon), and 'Pipeline' (with a hand icon). Each option has a brief description. At the bottom of the dialog is an 'OK' button.

Step 12: In 'Source Code Management' section, choose 'Git' and enter the following repository URL:-

https://github.com/shazforiot/MSBuild_firstproject

The above is a sample hello-world project with no vulnerabilities.

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?

+ Add

Advanced

Save Apply

Step 13: Under Build Steps, enter Sonarqube Scanner and enter these analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Build Steps

Execute SonarQube Scanner

JDK ?

JDK to be used for this SonarQube analysis

Path to project properties ?

Analysis properties ?

Additional arguments ?

Save Apply

Step 14: On your browser, go to http://localhost:<port_number>/admin/permissions and check the 'Execute Analysis' checkbox for Administrator. This gives the required permissions to the user for the analysis stage on SonarQube.

Configuration ▾ Security ▾ Projects ▾ System Marketplace

Global Permissions

Grant and revoke permissions to make changes at the global level. These permissions include editing Quality Profiles, executing analysis, and performing global system administration.

All Users Groups 🔍 Search for users or groups...

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

4 of 4 shown

Step 15: Navigate to your Jenkins project and click on 'Build Now'.

Dashboard > SonarqubeProject29 >

Status

</> Changes

Workspace

Build Now


Configure

Delete Project

SonarQube

Rename

✓ SonarqubeProject29

**Permalinks**

- Last build (#9), 24 min ago
- Last stable build (#9), 24 min ago
- Last successful build (#9), 24 min ago
- Last failed build (#8), 33 min ago
- Last unsuccessful build (#8), 33 min ago
- Last completed build (#9), 24 min ago

✎ Add description

Disable Project

Build History

trend ▾

Step 16: Once the build is successful, check the console output.

```
Started by user Anish Kulkarni
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SonarqubeProject29
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarqubeProject29\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url http://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from http://github.com/shazforiot/MSBuild_firstproject.git
> git.exe --version # timeout=10
> git --version # 'git version 2.46.0.windows.1'
> git.exe fetch --tags --force --progress -- http://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 f2bc042c04c6e72427c380bcae6d6fee7b49adf (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
Commit message: "updated"
> git.exe rev-list --no-walk f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
[SonarqubeProject29] $ C:\ProgramData\Jenkins\.jenkins\tools\udson.plugins.sonar.SonarRunnerInstallation\scanner29\bin\sonar-
scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.host.url=http://localhost:9000 -
Dsonar.login=admin -Dsonar.sources=. -Dsonar.password=ANISH2004 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\SonarqubeProject29
16:13:33.220 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
16:13:33.236 INFO Scanner configuration file:
C:\ProgramData\Jenkins\.jenkins\tools\udson.plugins.sonar.SonarRunnerInstallation\scanner29\bin\..\conf\sonar-scanner.properties
16:13:33.246 INFO Project root configuration file: NONE
```

```
16:14:12.549 INFO Sensor C# [csharp] (done) | time=2ms
16:14:12.549 INFO Sensor Analysis Warnings import [csharp]
16:14:12.549 INFO Sensor Analysis Warnings import [csharp] (done) | time=0ms
16:14:12.549 INFO Sensor C# File Caching Sensor [csharp]
16:14:12.549 WARN Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting
'sonar.projectBaseDir' property.
16:14:12.549 INFO Sensor C# File Caching Sensor [csharp] (done) | time=0ms
16:14:12.549 INFO Sensor Zero Coverage Sensor
16:14:12.555 INFO Sensor Zero Coverage Sensor (done) | time=6ms
16:14:12.555 INFO SCM Publisher SCM provider for this project is: git
16:14:12.567 INFO SCM Publisher 4 source files to be analyzed
16:14:13.180 INFO SCM Publisher 4/4 source files have been analyzed (done) | time=613ms
16:14:13.184 INFO CPD Executor Calculating CPD for 0 files
16:14:13.184 INFO CPD Executor CPD calculation finished (done) | time=0ms
16:14:13.191 INFO SCM revision ID 'f2bc042c04c6e72427c380bcae6d6fee7b49adf'
16:14:13.474 INFO Analysis report generated in 117ms, dir size=201.0 kB
16:14:13.522 INFO Analysis report compressed in 39ms, zip size=22.2 kB
16:14:13.756 INFO Analysis report uploaded in 231ms
16:14:13.759 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
16:14:13.759 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis
report
16:14:13.759 INFO More about the report processing at http://localhost:9000/api/ce/task?id=fe09e0c2-e06a-411d-ae21-10cce1b3c081
16:14:13.787 INFO Analysis total time: 25.432 s
16:14:13.789 INFO SonarScanner Engine completed successfully
16:14:13.842 INFO EXECUTION SUCCESS
16:14:13.842 INFO Total time: 40.606s
Finished: SUCCESS
```


Step 17: Go back to SonarQube and check your project.

The screenshot displays the SonarQube web interface. At the top, the navigation bar includes the SonarQube logo and links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. A search icon is also present. Below the navigation bar, the breadcrumb trail shows 'sonarqube / main'. The main content area is titled 'main' and shows a 'Quality Gate' status of 'Passed' with a green checkmark. A warning message states 'The last analysis has warnings. See details'. The 'Last analysis' timestamp is '3 minutes ago'. Below the status, there are tabs for 'New Code' and 'Overall Code'. The 'Overall Code' tab is active, showing three metrics: Security, Reliability, and Maintainability. Each metric has a green circle with the letter 'A' and a bar chart showing '0 H', '0 M', and '0 L' (High, Medium, Low) issues.

Metric	Open Issues	High (H)	Medium (M)	Low (L)
Security	0	0	0	0
Reliability	0	0	0	0
Maintainability	0	0	0	0

Conclusion: In this experiment, we learned how to integrate Jenkins SAST to SonarQube. We first used a docker image of SonarQube in order to avoid having to install it on our system. Next, SonarQube was configured and a SonarQube project was created. Then, required configurations were done on Jenkins and a Jenkins freestyle project was created which contained links to a code file from a Github repository and also our SonarQube project. When the Jenkins project was built, the SonarQube project displayed that there were no issues with the code in the Jenkins project.