ADVANCE DEVOPS EXP 7

Name:Aryan Anil Patankar
Class:D15A
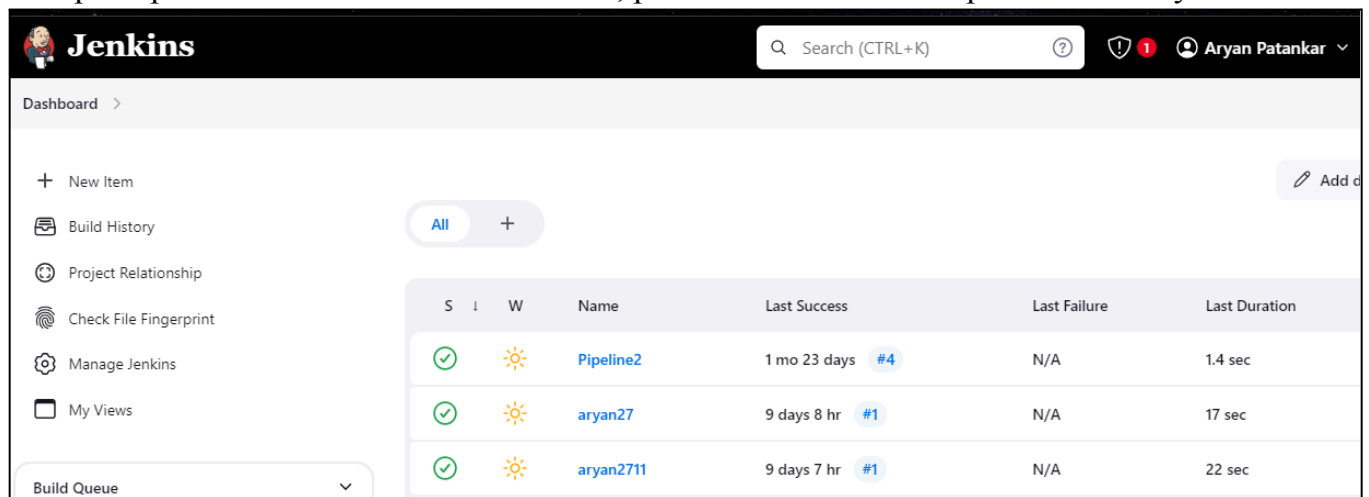**Roll No:34**

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

Integrating Jenkins with SonarQube:

● Jenkins installed
● Docker Installed (for SonarQube)
● SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.
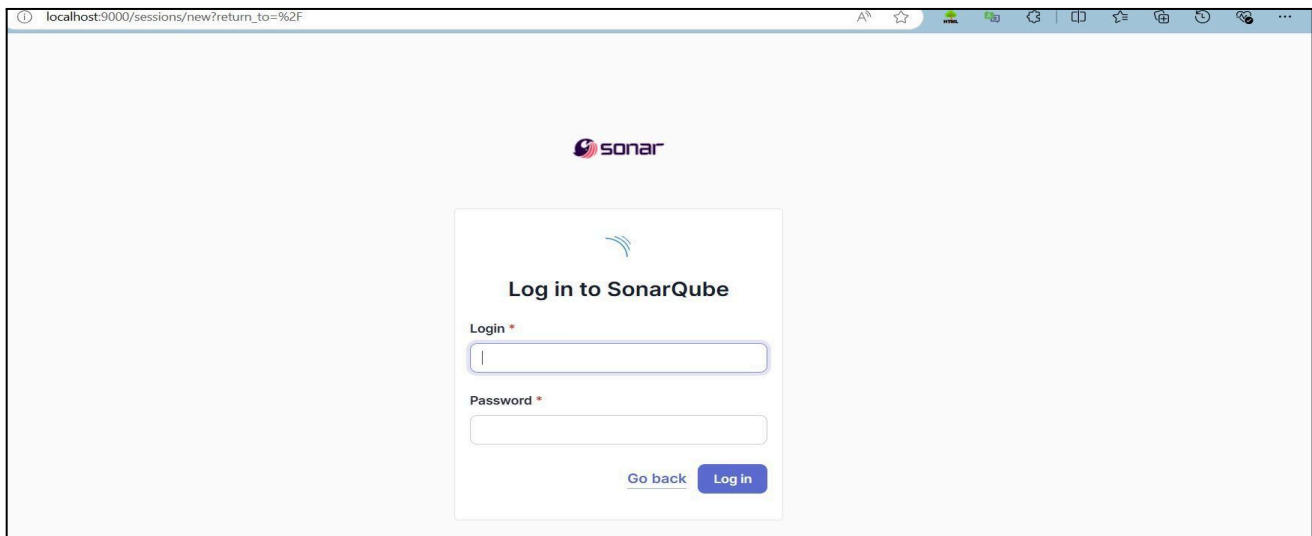

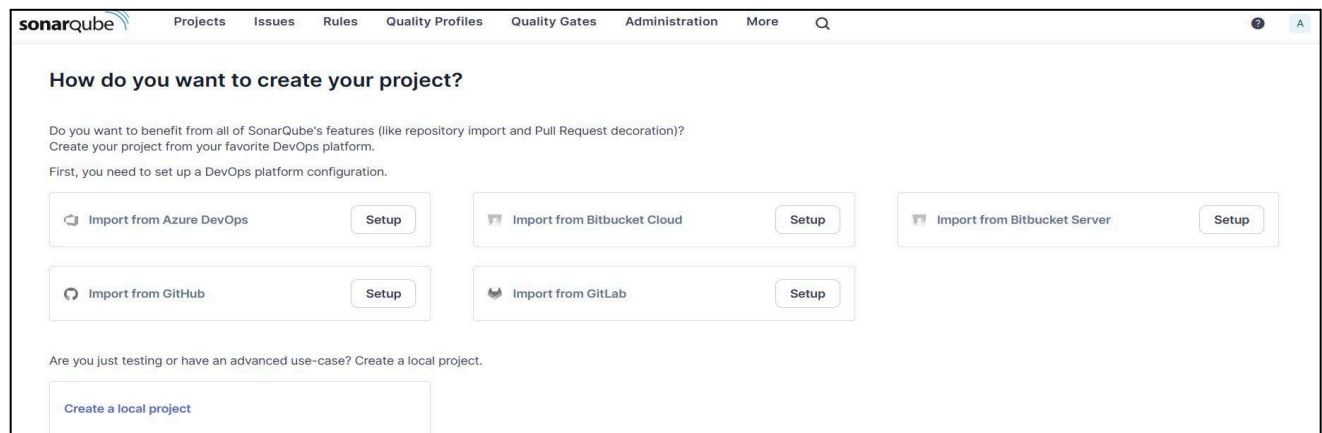2. Run SonarQube in a Docker container using this command -
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

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

4. Login to SonarQube using username admin and password admin.

5. Create a manual project in SonarQube with the name sonarqube

Setup the project and come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.
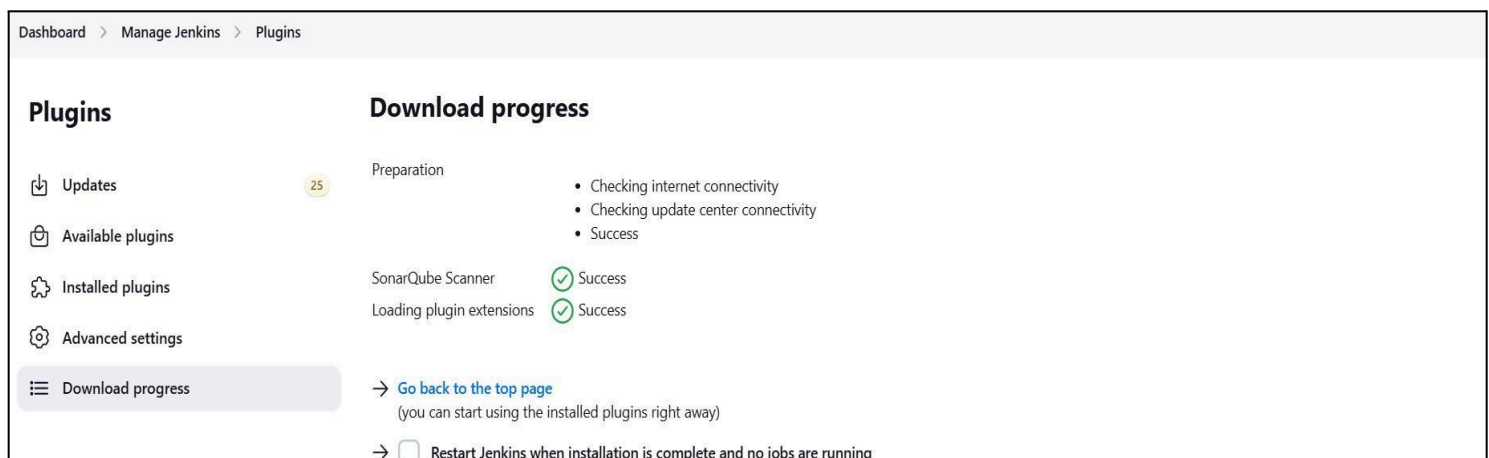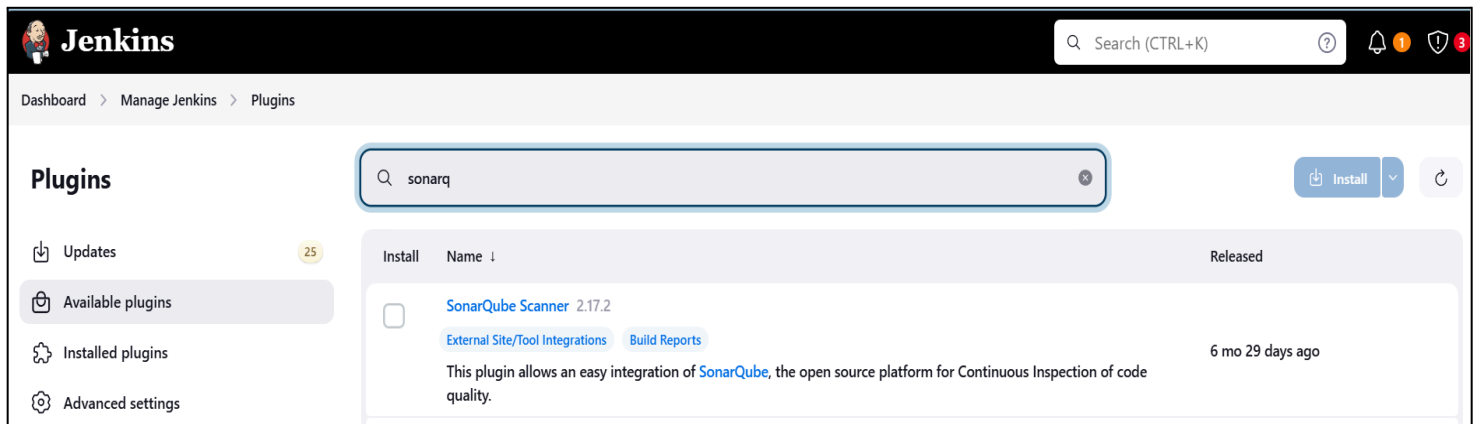
6. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers**
and enter the details.
Enter the Server Authentication token if needed.

In SonarQube installations: Under **Name** add <project name of sonarqube>,here we have named it as **adv_devops_7_sonarqube**
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

    adv_devops_7_sonarqube

Server URL

Default is http://localhost:9000

    https://localhost:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

    - none -   ⌄

    + Add ▾

Advanced ⌄

7.   Search for SonarQube Scanner under Global Tool Configuration.

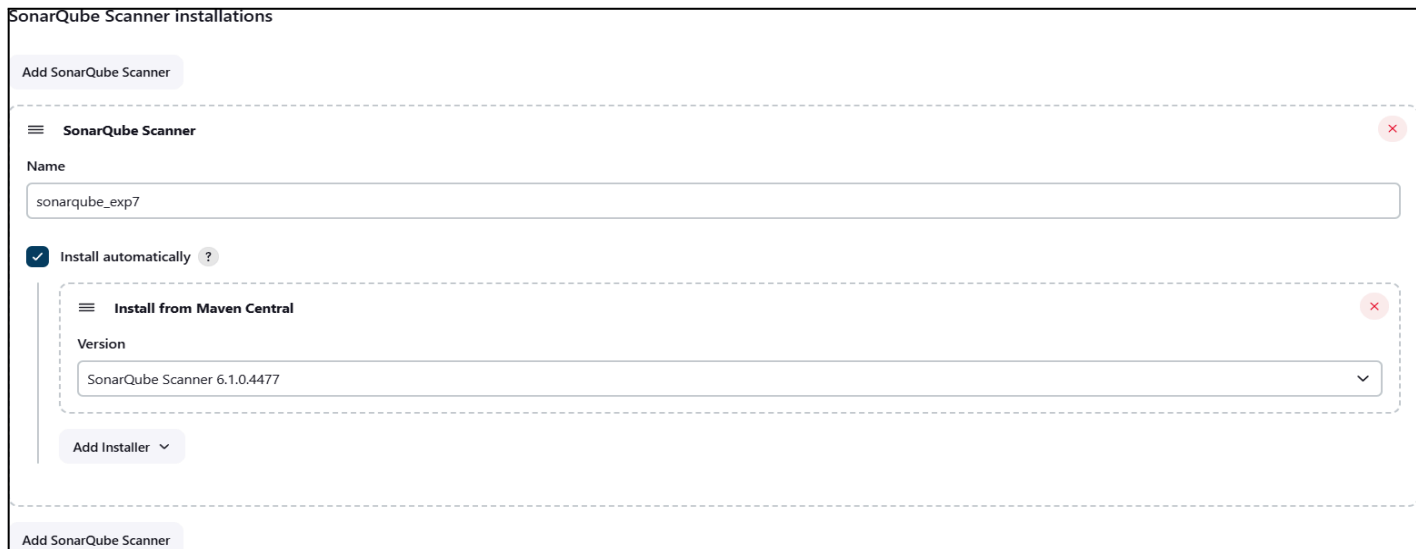Choose the latest configuration and choose Install automatically.

Dashboard > Manage Jenkins > Tools

Check the "Install automatically" option. → Under name any name as identifier → Check the "Install automatically" option.

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

adv_devops_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.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
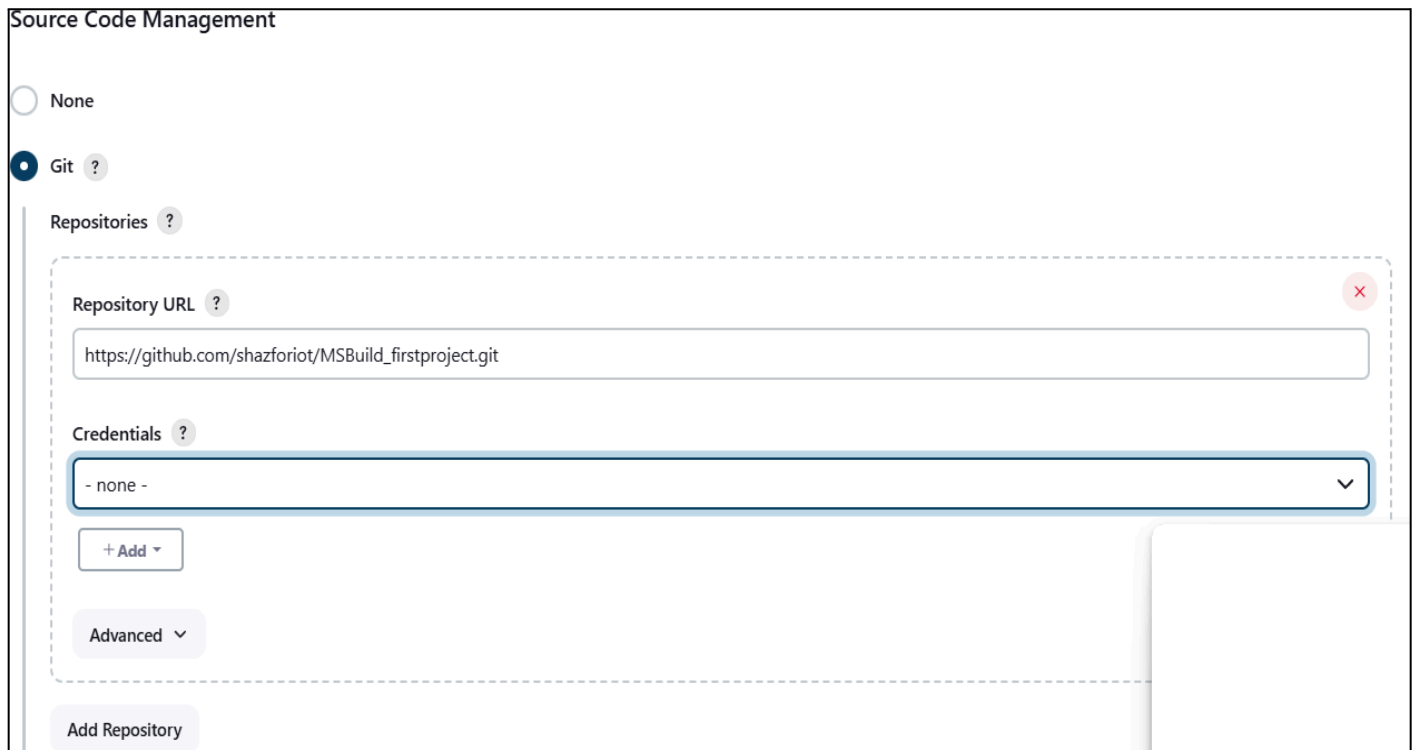
OK ranch Pipeline

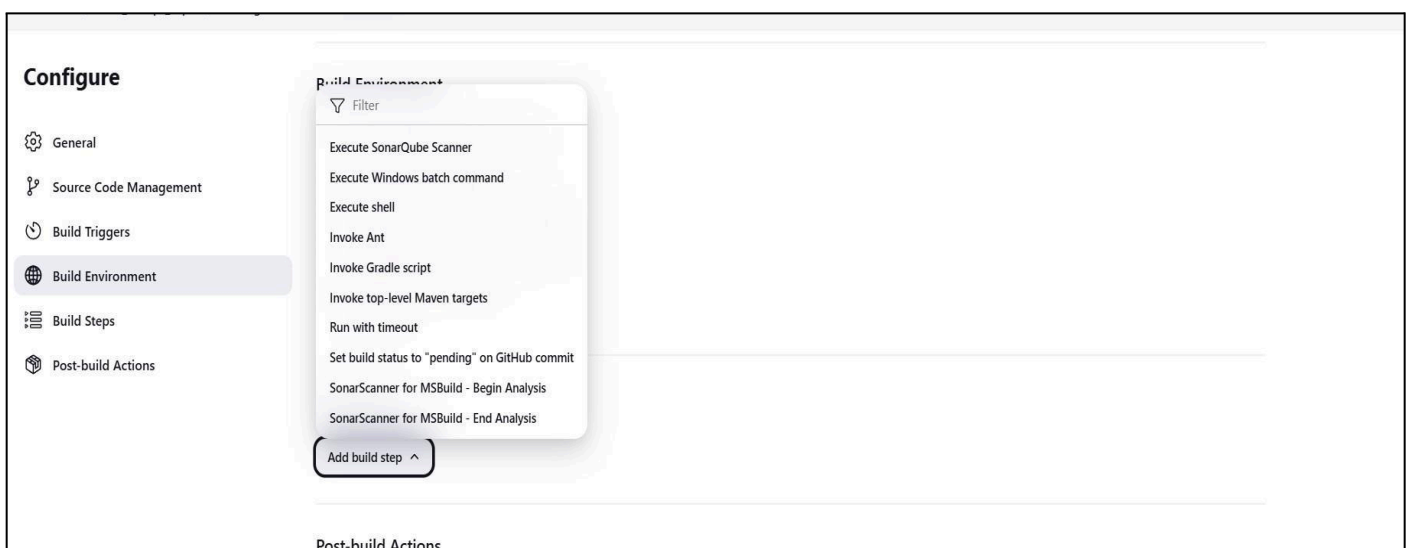9.   Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git
It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

Source Code Management

○ None

● Git ?

Repositories ?

Repository URL ?

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

Credentials ?

- none -

+ Add ▾

Advanced ∨

Add Repository

10.   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.

Configure

Build Environment

▽ Filter

{ } General

Execute SonarQube Scanner

Execute Windows batch command

⅄ Source Code Management

Execute shell

🕓 Build Triggers

Invoke Ant

🌐 Build Environment

Invoke Gradle script

Invoke top-level Maven targets

≔ Build Steps

Run with timeout

Post-build Actions

Set build status to "pending" on GitHub commit

SonarScanner for MSBuild - Begin Analysis

SonarScanner for MSBuild - End Analysis

Add build step ∧

Post-build Actions

Then save

11.   Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user
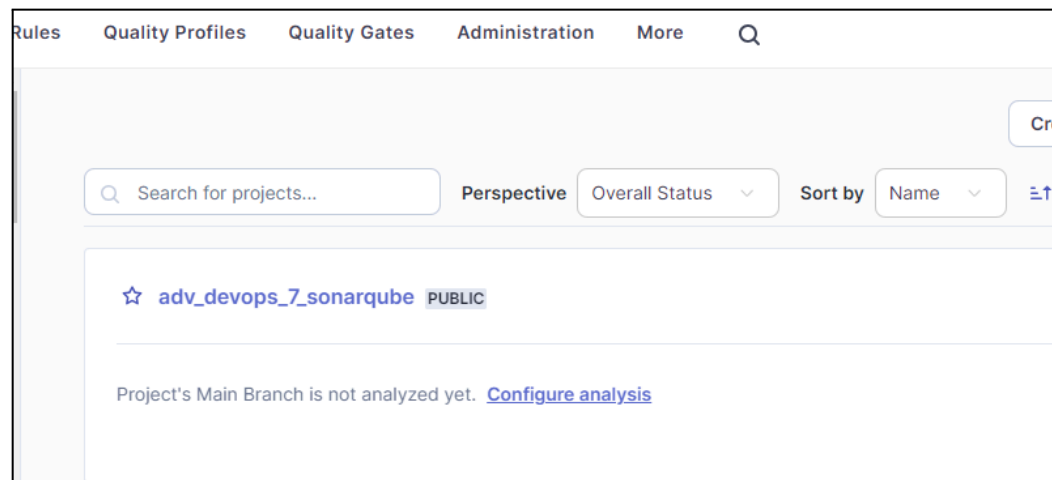
IF CONSOLE OUTPUT FAILED:

Step 1: Generate a New Authentication Token in SonarQube
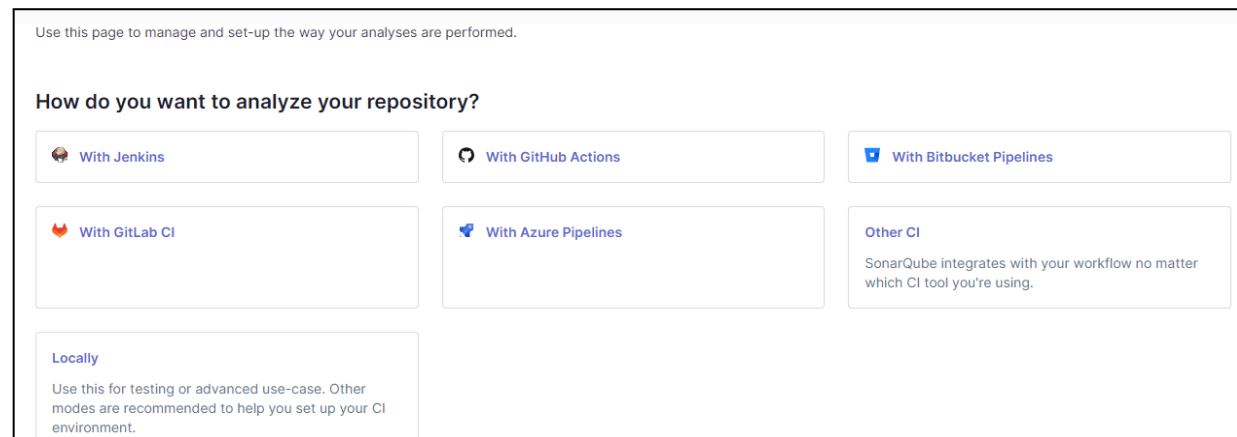
1. **Login to SonarQube:**
   ○ Open your browser and go to **http://localhost:9000**.
   ○ Log in with your admin credentials (default username is **admin**, and the password is either **admin** or your custom password if it was changed).

2. **Generate a New Token:**
   ○ Go to the project that you have created on SonarQube.

   ○ Click on **Locally**

   ○ Further,Generate a Project token with the following details and click on generate.

- ○ Copy the token you get here and save it securely as we would need it in Jenkins.

Step 2: Update the Token in Jenkins

1. **Go to Jenkins Dashboard:**
   - ○ Open Jenkins and log in with your credentials.

2. Go to Dashboard—>Manage Jenkins—>Credentials
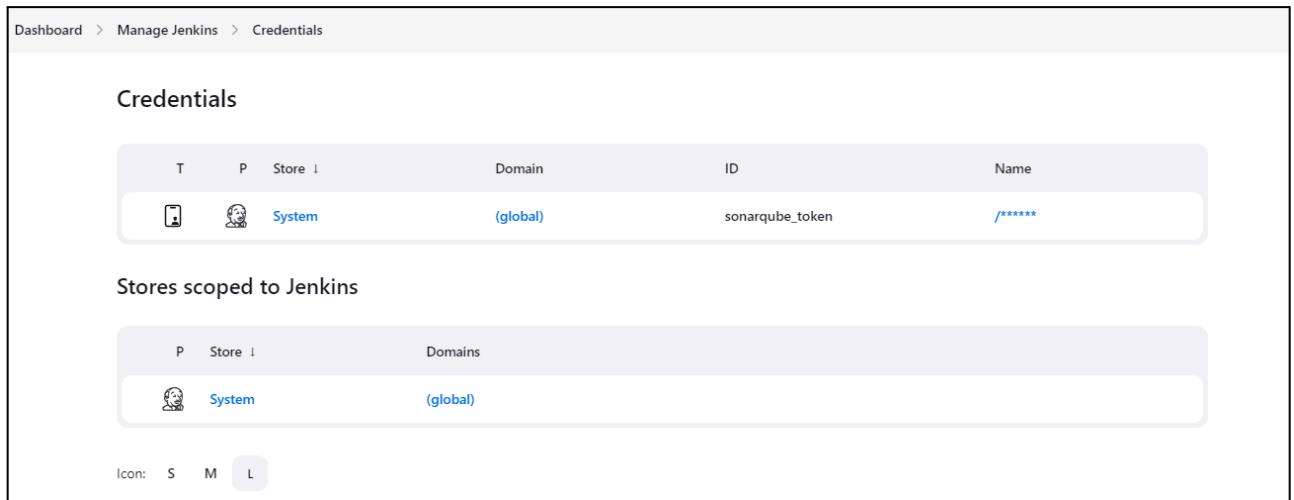
3. Click on **global** under the domains part of Stores scoped to Jenkins section.Further click on add credentials.Proceed with the following details.Make sure to copy the token generated earlier in sonarqube and give any suitable name as the ID.

4.After clicking on create we see that the given token has been added in Jenkins credentials.

5.Now go to **Manage Jenkins—>System—>SonarQube servers** and proceed with the following details. Reference the authentication token generated in the previous step.

6.Check the SonarQube Scanner Environment and add the server authentication token

12.Run the Jenkins build.

Check the console Output

13. Once the build is complete, check project on SonarQube

In this way, we have integrated Jenkins with SonarQube for SAST.