



Semester: V
Academic Year: 2024-25
Class / Branch: TE IT
Subject: Advanced Devops Lab (ADL)
Name of Instructor: Prof. Manjusha Kashilkar

Name of Student: Harsh Prajapati
Student ID: 22104188
Date of Submission: 30/09/24

EXPERIMENT NO. 07

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

1) Install and configure a Jenkins and SonarQube CICD environment using Docker containers.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** Pipeline: Milestone Step ** JavaScript GUI Lib: jQuery bundles (jQuery and jQuery UI) ** Jackson 2 API ** JavaScript GUI Lib: ACE Editor bundle ** Pipeline: SCM Step ** Pipeline: Groovy ** Pipeline: Input Step ** Pipeline: Stage Step ** Pipeline: Job ** Pipeline Graph Analysis ** Pipeline: REST API ** JavaScript GUI Lib: Handlebars bundle ** JavaScript GUI Lib: Moment.js bundle Pipeline: Stage View ** Pipeline: Build Step ** Pipeline: Model API ** Pipeline: Declarative Extension Points API ** Apache HttpComponents Client 4.x API ** JSch dependency
✓ Timestampers	✓ Workspace Cleanup	✓ Ant	✓ Gradle	
🔗 Pipeline	🔗 GitHub Branch Source	🔗 Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View	
🔗 Git	🔗 Subversion	🔗 SSH Slaves	🔗 Matrix Authorization Strategy	
🔗 PAM Authentication	🔗 LDAP	🔗 Email Extension	🔗 Mailer	

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Click Start using Jenkins to visit the main Jenkins dashboard:

The screenshot shows the Jenkins Dashboard interface. On the left, there is a sidebar with navigation links: 'New Item', 'People', 'Build History', and 'Manage Jenkins'. Below these are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing 1 Idle and 2 Idle executors). The main area displays a table of build items. The table has columns for 'S' (status), 'W' (weather icon), 'Name', 'Last Success', 'Last Failure', 'Last Duration', and 'Coverage'. The table lists several builds, including 'aniruddha', 'helloworld06', 'p2', and multiple 'sonarqube' builds with various IDs and durations.

S	W	Name	Last Success	Last Failure	Last Duration	Coverage
✓	☁	aniruddha	5 days 23 hr #16	5 days 23 hr #15	13 sec	n/a
✓	☀	helloworld06	1 mo 24 days #1	N/A	1.1 sec	n/a
✗	☁	p2	N/A	1 hr 8 min #18	1.3 sec	n/a
✓	☁	sonarqube	1 yr 0 mo #12	1 yr 0 mo #11	1.3 sec	n/a
✗	☀	Sonarqube	N/A	N/A	N/A	n/a
✓	☀	Sonarqube1	1 hr 7 min #4	N/A	0.46 sec	n/a
✗	☁	sonarqube2	N/A	1 yr 0 mo #4	4.2 sec	n/a
✗	☁	sonarqube_1	N/A	26 days #10	0.55 sec	n/a
✗	☁	sonarqube_prasad	N/A	11 days #6	0.5 sec	n/a
✗	☁	sonarqube_vandan	N/A	1 hr 1 min #1	0.84 sec	n/a
✓	☀	sonarqube_VB	1 mo 10 days #1	N/A	1.1 sec	n/a

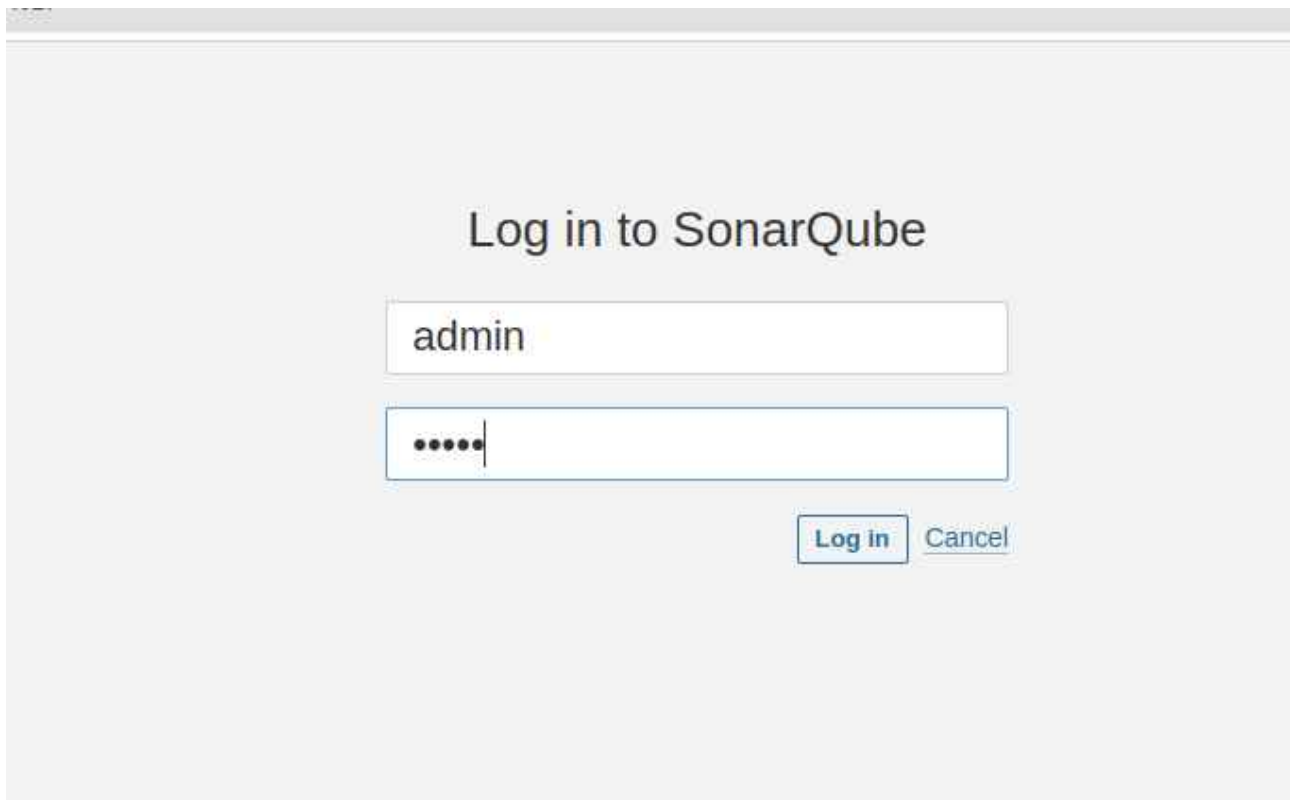
SonarQube Setup

Before proceeding with the integration, we will setup SonarQube Instance. we are using SonarQube Docker Container.

manjusha@apsit:~\$docker run -d -p 9000:9000 sonarqube

```
Processing triggers for ureadahead (0.100.0-21) ...
root@ubuntu:/home/manasi# docker run -d -p 9000:9000 sonarqube
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
9621f1afde84: Pull complete
1220b1fb64e6: Pull complete
f0a3b7127ede: Pull complete
Digest: sha256:9ca40ae23bb2228a6c4cc8c20de41fcd72a8ed7358331b4bd5910cd20dcee995
Status: Downloaded newer image for sonarqube:latest
ed54d42e5aa9a31f212c204e48e47b80e63478abbf7960ce65c6c56a99a35e24
root@ubuntu:/home/manasi#
```

In the above command, we are forwarding port 9000 of the container to the port 9000 of the host machine as SonarQube is will run on port 9000. Then, from the browser, enter <http://localhost:9000>. After That, you will see the SonarQube is running. Then, login using default credentials (admin:admin).

The image shows the SonarQube login page. At the top, it says "Log in to SonarQube". Below this, there are two input fields. The first field contains the text "admin". The second field contains five dots, indicating a password. To the right of the password field, there are two buttons: "Log in" and "Cancel".

Log in to SonarQube

admin

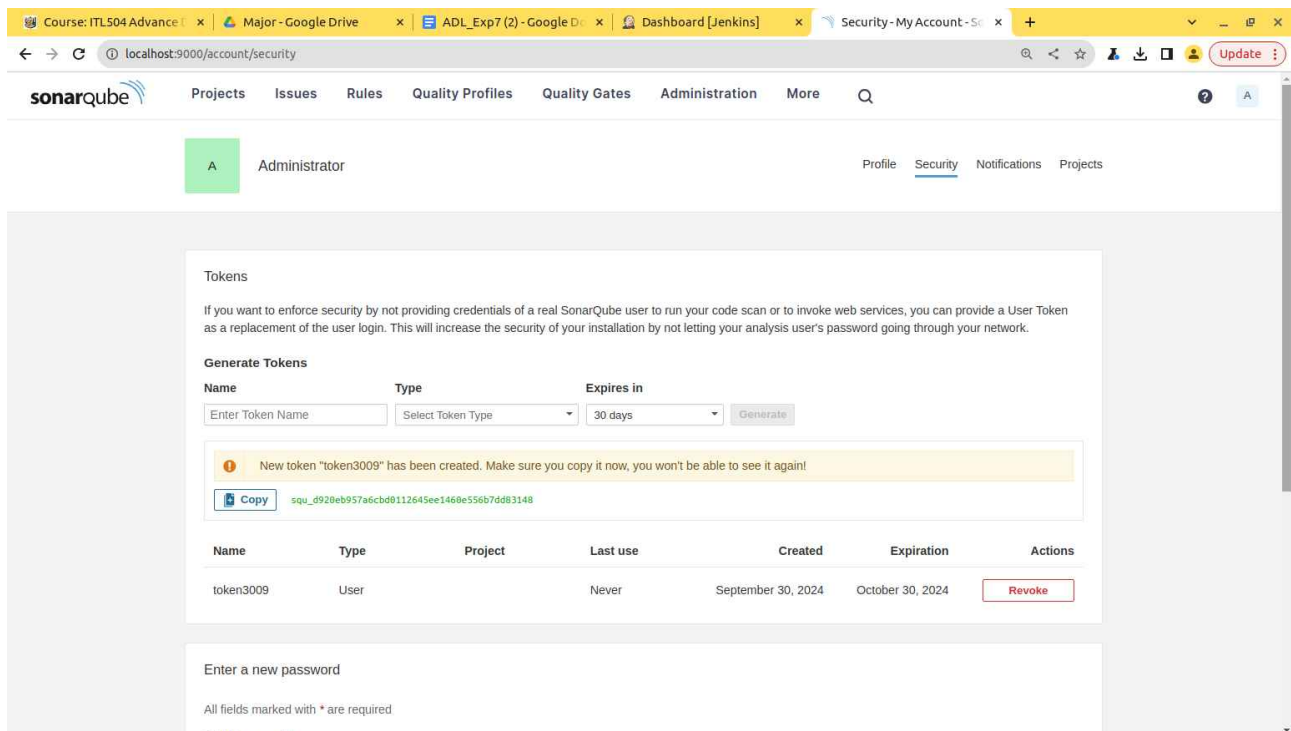
.....

Log in Cancel

Generate User Token

Now, we need to get the SonarQube user token to make connection between Jenkins and SonarQube. For the same, go to **Administration > User > My Account > Security** and then, from the bottom of the page you can create new tokens by clicking the Generate Button. Copy the Token and keep it safe.

C96798e9bd081e117189b516c868ddb7d87ee785 **SonarQube**



2) Configure Jenkins with the SonarQube Scanner plugin for automated static code analysis.

Jenkins Setup for SonarQube

Before all, we need to install the SonarQube Scanner plugin in Jenkins. For the same, go to **Manage Jenkins > Plugin Manager > Available**. From here, type SonarQube Scanner then select and install.

The screenshot shows the Jenkins 'Available plugins' page. A search bar at the top contains the text 'sonar'. Below the search bar, there are two plugin entries:

- SonarQube Scanner** (version 2.17.2): Released 7 mo 14 days ago. Description: 'This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.'
- Sonar Quality Gates** (version 1.3.1): Released 6 yr 1 mo ago. Description: 'Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")'. A warning box is present: 'Warning: This plugin version may not be safe to use. Please review the following security notices: Credentials transmitted in plain text'.

Tool Configuration SonarQube Scanner

Now, we need to configure the Jenkins plugin for SonarQube Scanner to make a connection with the SonarQube Instance. For that, got to **Manage Jenkins > Configure System > SonarQube Server**. Then, Add SonarQube. In this, give the Installation Name, Server URL then Add the Authentication token in the Jenkins Credential Manager and select the same in the configuration.

The screenshot shows the 'SonarQube servers' configuration page in Jenkins. The page has a section titled 'SonarQube installations' with the following fields:

- Name:** SonarQube
- Server URL:** http://localhost:9000 (Default is http://localhost:9000)
- Server authentication token:** sonarqube (with an 'Add' button next to it)

Below the token field, there is a note: 'SonarQube authentication token. Mandatory when anonymous access is disabled.'

Then, we need to set-up the SonarQube Scanner to scan the source code in the various stage. For the same, go to **Manage Jenkins > Global Tool Configuration > SonarQube Scanner**. Then, Click **Add SonarQube Scanner Button**. From there, give some name of the scanner type and **Add Installer** of your choice. In this case, I have selected SonarQube Scanner from Maven Central.

SonarQube Scanner

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name

SonarQube

☒ Install automatically

Install from Maven Central

Version

SonarQube Scanner 4.6.2.2472 ▼


Add Installer ▼


SonarQube Scanner in Jenkins Pipeline


Now, It's time to integrate the SonarQube Scanner in the Jenkins Pipeline. For the same, we are going to add one more stage in the Jenkinsfile called SonarQube and inside that, I am adding the following settings and code.


Enter an item name


» Required field

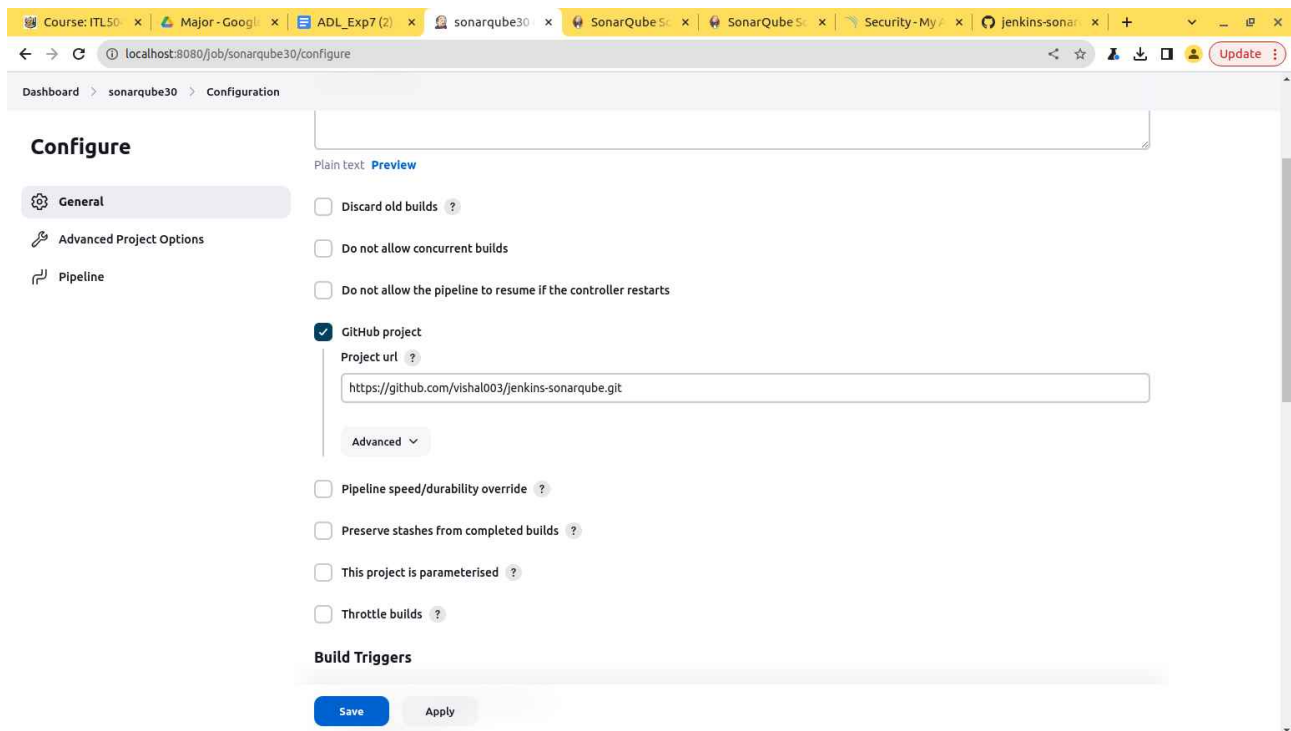
**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**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.

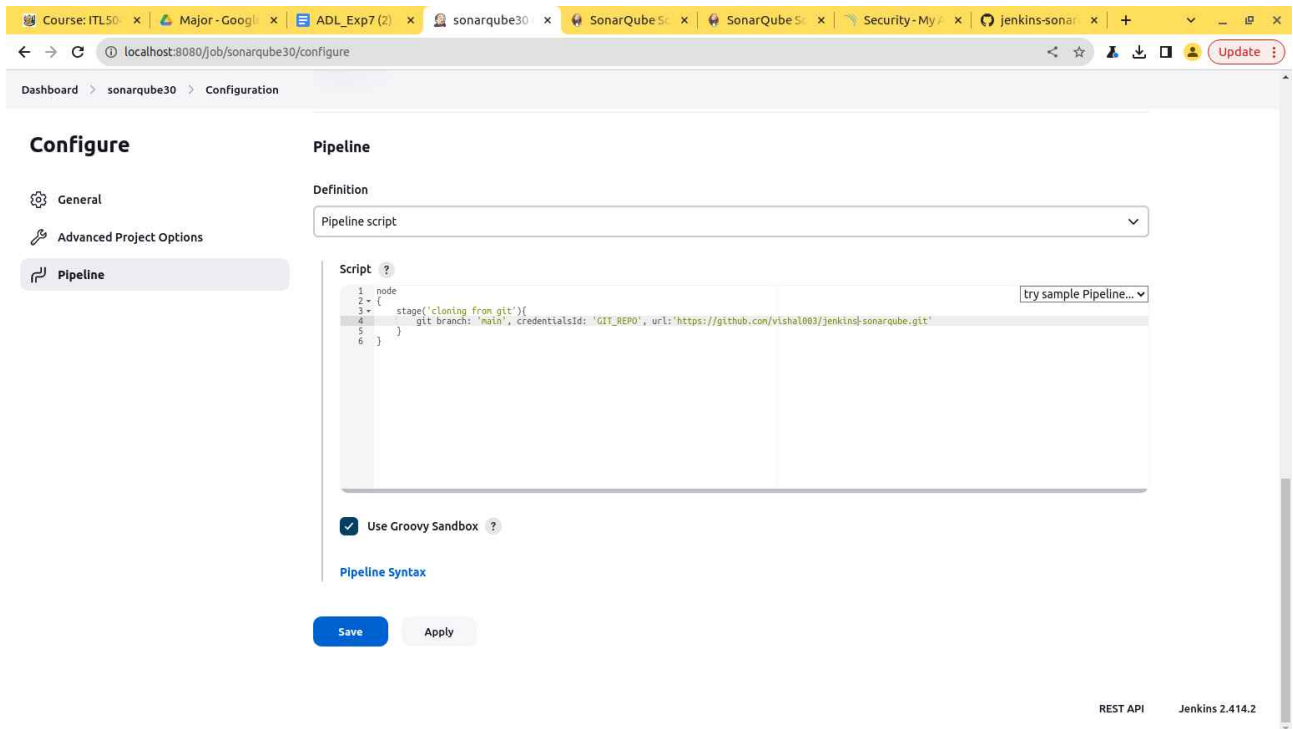
**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Bitbucket Team/Project**
Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.

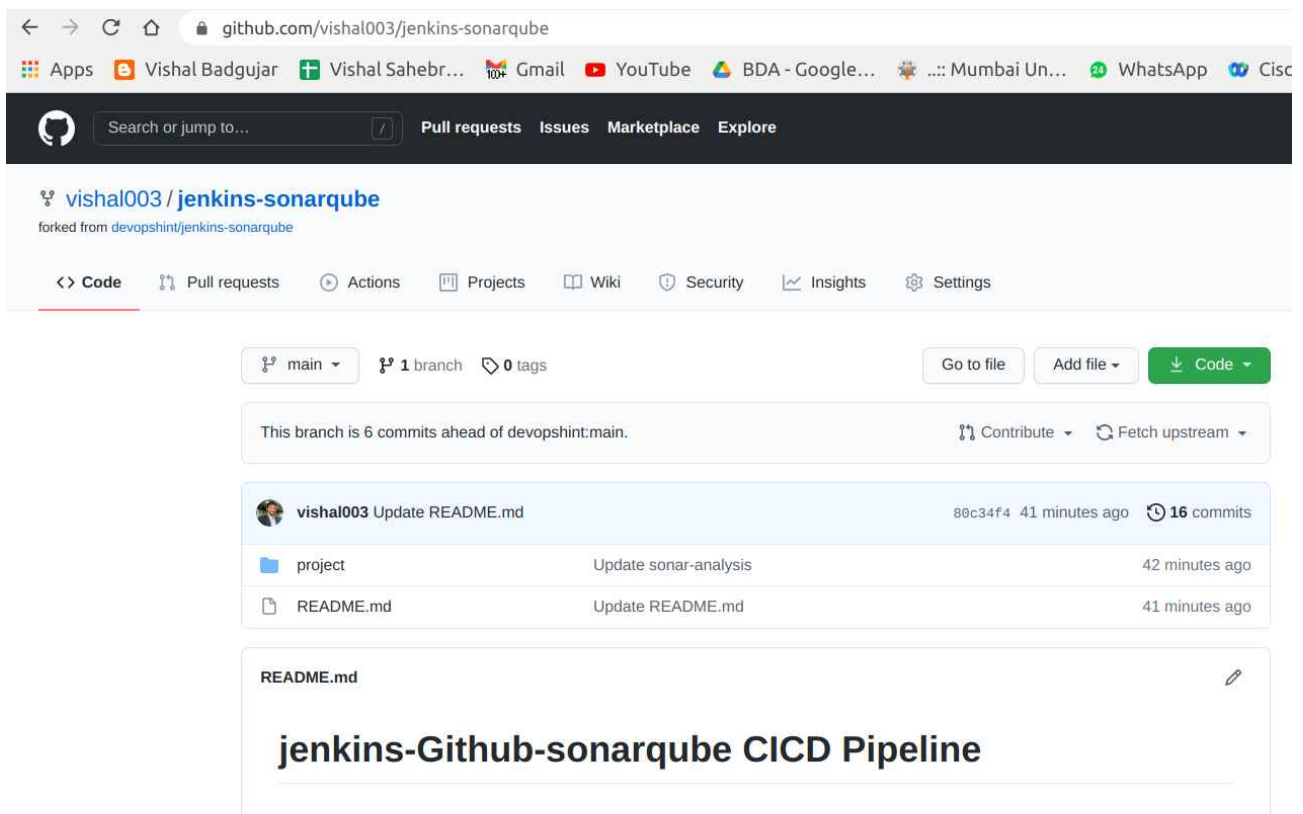
**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Github Configuration in Jenkins Pipeline



Git Clonning into Jenkins



Github Repository Contents

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8080/job/sonarqube30/1/console`. The Jenkins logo and name are at the top left. A search bar is at the top right. Below the header, the breadcrumb navigation shows `Dashboard > sonarqube30 > #1`. On the left sidebar, the 'Console Output' tab is selected. The main area displays the console output for build #1, which includes the following text:

```
Started by user unknown or anonymous
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/sonarqube30
[Pipeline] {
[Pipeline] stage
[Pipeline] { (cloning from git)
[Pipeline] git
Failed to get git executable
Failed to get git executable
The recommended git tool is: NONE
Warning: CredentialId "GIT_REPO" could not be found.
Cloning the remote Git repository
Cloning repository https://github.com/vishal003/jenkins-sonarqube.git
> git init /var/lib/jenkins/workspace/sonarqube30 # timeout=10
Fetching upstream changes from https://github.com/vishal003/jenkins-sonarqube.git
> git --version # timeout=10
> git --version # 'git version 2.17.1'
> git fetch --tags --progress -- https://github.com/vishal003/jenkins-sonarqube.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/vishal003/jenkins-sonarqube.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 80c34f4818e25f7733e50784c2f7639d9884ed90 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 80c34f4818e25f7733e50784c2f7639d9884ed90 # timeout=10
```

Successfully Build Github Repository in Jenkins

Pre-requisite required for Integration settings of Jenkins SAST with SonarQube we have done here successfully, now in order to Integrate of Jenkins CICD with SonarQube with the help of sample JAVA program we will implement in next experiment.

Conclusion: In this experiment we have succesfully installed jenkins and SonarQube and