

## Adv.DevOps Exp 08

Name- Ishika Devare RollNo- 14 Batch- A

<u>Adv.DevOps</u>	
Experiment No. 8	
Aim - create a Jenkins CI/CD pipeline with sonarQube / GitLab integration to perform a static analysis of the code to detect bugs, code smells and security vulnerabilities on a sample Web / Java / Python application.	
●	Theory -
	What is SAST? Static Application Security Testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack.
●	SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to next phase of software development life cycle (SDLC).
	What is CI/CD pipeline? CI/CD pipeline refers to continuous Integration/continuous Development pipeline. A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases.
	FOR EDUCATIONAL USE

The pipeline is a backbone of DevOps approach. The pipeline is responsible for building codes, running tests and deploying new software versions.

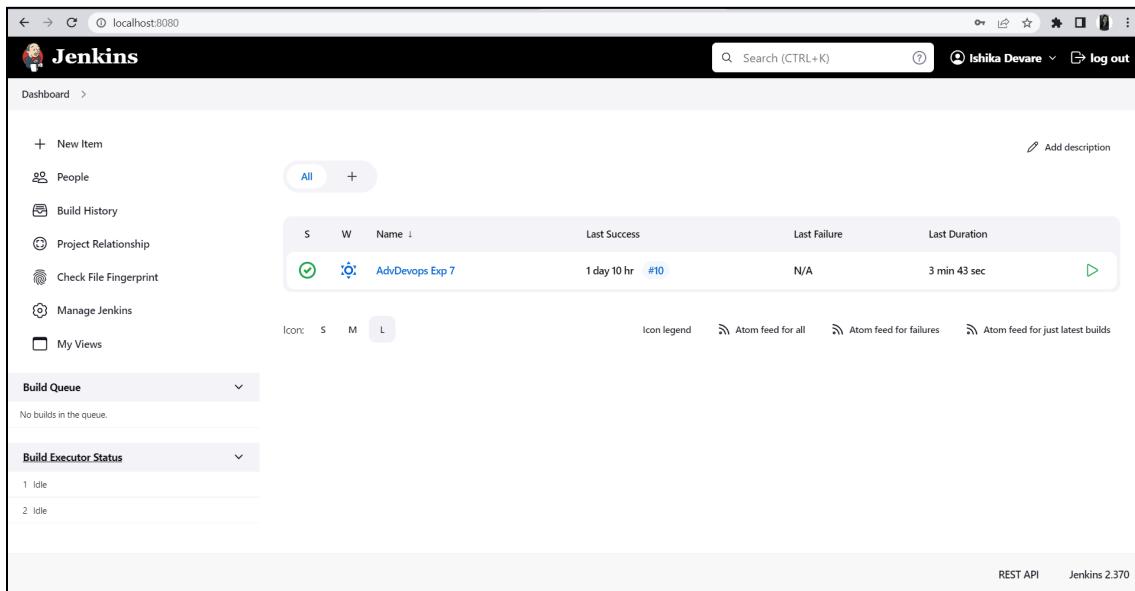
What is SonarQube?

SonarQube is an open-source platform developed by Sonarsource for continuous inspection of code quality.

It supports 25+ major programming languages through built-in rulesets and also be extended with various plugins.

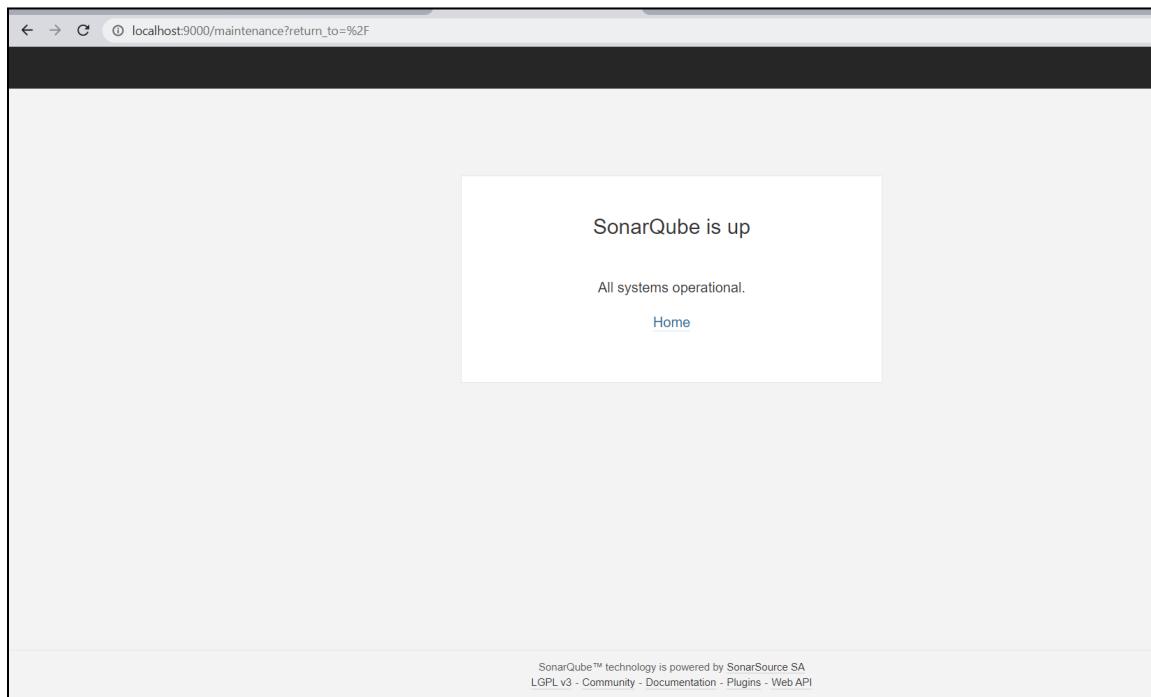
## Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST:

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



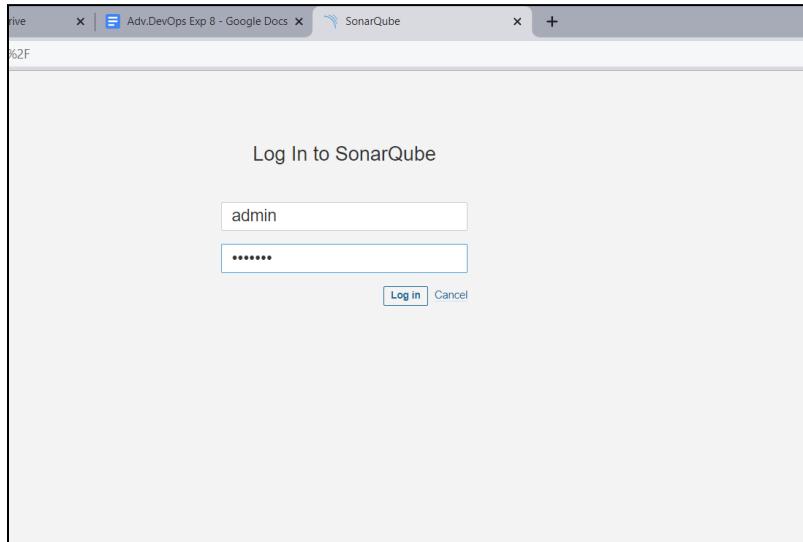
The screenshot shows the Jenkins dashboard at [localhost:8080](http://localhost:8080). The top navigation bar includes links for 'Dashboard', 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. A search bar and a user profile for 'Ishika Devare' are also present. The main content area displays a table of builds. One build, 'AdvDevops Exp 7', is highlighted with a green checkmark icon. Its details show 'Last Success' as 1 day 10 hr ago, '#10', and 'Last Failure' as N/A. The 'Last Duration' is listed as 3 min 43 sec. Below the table, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (1 idle). At the bottom right, there are links for 'REST API' and 'Jenkins 2.370'.

**Step 2:** Open SonarQube dashboard on localhost, **port 9000** or whichever port it is at for you.



The screenshot shows the SonarQube maintenance page at [localhost:9000/maintenance?return\\_to=%2F](http://localhost:9000/maintenance?return_to=%2F). The page has a dark header and a light body. In the center, a white box contains the text 'SonarQube is up' and 'All systems operational.' Below this, a 'Home' link is visible. At the very bottom of the page, a footer notes: 'SonarQube™ technology is powered by SonarSource SA' and 'LGPL v3 - Community - Documentation - Plugins - Web API'.

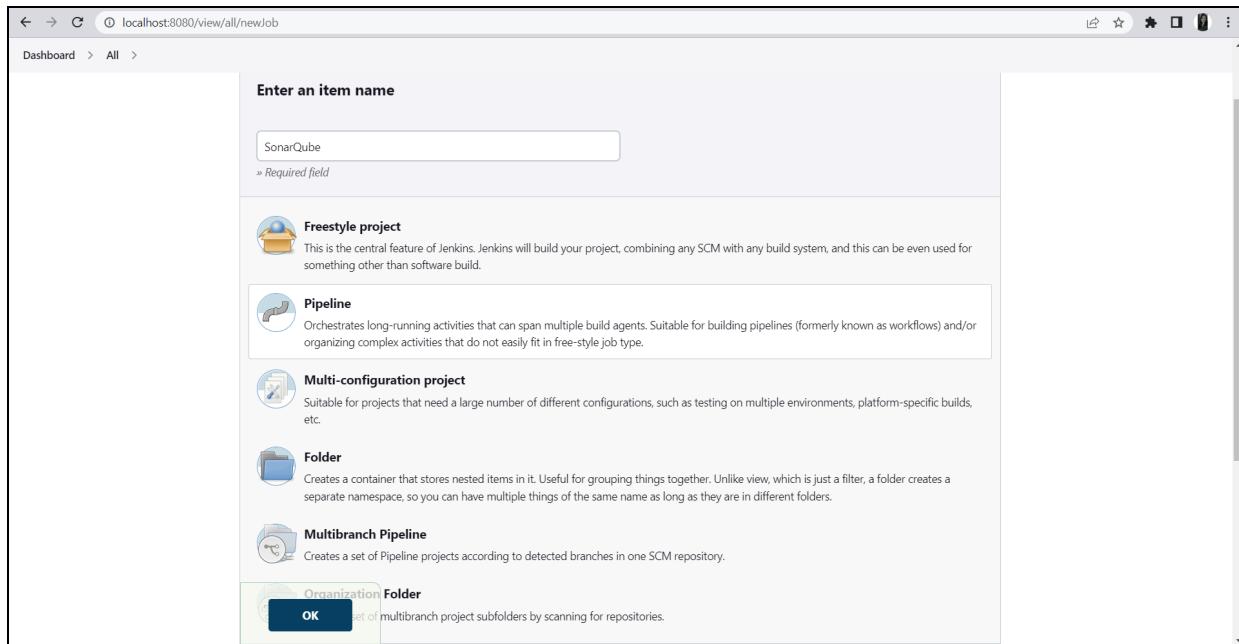
**Step 3:** Login to SonarQube using username *admin* and password *which you have set.*



**Step 4:** Create a manual project in SonarQube with the name sonarqube-test. Setup the project and come back to Jenkins Dashboard.

A screenshot of a web browser window showing the 'Create a project' page on SonarQube. The URL in the address bar is 'localhost:9000/projects/create?mode=manual'. The page has a dark header with the SonarQube logo and navigation links for 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. The main content area has a light gray background with the heading 'Create a project'. Below it, a note says 'All fields marked with \* are required'. There are two main input fields: 'Project display name \*' containing 'sonarqube\_test' and 'Project key \*' also containing 'sonarqube\_test'. Both fields have a green checkmark icon to their right. A small note below the project key field states: 'Up to 255 characters. Some scanners might override the value you provide.' and 'The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '\_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.' At the bottom of the form is a blue 'Set Up' button.

**Step 5:** Create a New Item in Jenkins, choose **Pipeline**



## Step 6: Under Pipeline Script, enter the following -

```
node {  
    stage('Cloning the GitHub Repo') {  
        git 'https://github.com/shazforiot/GOL.git'  
    }  
    stage('SonarQube analysis') {  
        withSonarQubeEnv('sonarqube') {  
            sh "<PATH_TO SONARQUBE FOLDER>/bin//sonar-scanner \  
                -D sonar.login=<SonarQube_USERNAME> \  
                -D sonar.password=<SonarQube_PASSWORD> \  
                -D sonar.projectKey=<Project_KEY> \  
                -D sonar.exclusions=vendor/**,resources/**, **/*.java \  
                -D sonar.host.url=http://127.0.0.1:9000/"  
        }  
    }  
}
```

Definition

Pipeline script

Script ?

```

1 node {
2   stage('Clone the Git') {
3     git 'https://github.com/shazforiot/GOL.git'
4   }
5   stage('SonarQube analysis') {
6     def scannerHome = tool 'sonarqube';
7     withSonarQubeEnv('sonarqube') {
8       bat "${scannerHome}/bin/sonar-scanner \
9         -D sonar.login=admin \
10        -D sonar.password=ishai123 \
11        -D sonar.projectKey=sonarqube_test \
12        -D sonar.exclusions=vendor/**,resources/**,**/*.java \
13        -D sonar.host.url=http://localhost:9000/"
14     }
15   }
16 }
17 }
18 }
19 }
20

```

Use Groovy Sandbox ?

Pipeline Syntax

**Save** **Apply**

It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

## Step 7: Run the build

The screenshot shows the Jenkins Pipeline SonarQube page. The pipeline script is displayed in the 'Pipeline script' section:

```

node {
  stage('Clone the Git') {
    git 'https://github.com/shazforiot/GOL.git'
  }
  stage('SonarQube analysis') {
    def scannerHome = tool 'sonarqube';
    withSonarQubeEnv('sonarqube') {
      bat "${scannerHome}/bin/sonar-scanner \
        -D sonar.login=admin \
        -D sonar.password=ishai123 \
        -D sonar.projectKey=sonarqube_test \
        -D sonar.exclusions=vendor/**,resources/**,**/*.java \
        -D sonar.host.url=http://localhost:9000/"
    }
  }
}

```

The 'Use Groovy Sandbox' checkbox is checked. Below the script, there are 'Save' and 'Apply' buttons. The Jenkins interface includes a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. A 'Stage View' section shows a message: 'No data available. This Pipeline has not yet run.' At the bottom, there are sections for Build History (with a single entry for Sep 25, 2022, 2:22 PM) and Permalinks, along with Atom feed links.

## Step 8: Check the console output once the build is complete.

The screenshot shows the Jenkins interface for a SonarQube build. The left sidebar contains links like Status, Changes, Console Output (which is selected), View as plain text, Edit Build Information, Delete build '#38', Git Build Data, Replay, Pipeline Steps, and Workspaces. The main content area is titled 'Console Output' with a green checkmark icon. It displays the build logs:

```
Started by user Ishika Devare
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone the Git)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> git.exe --version # timeout=10
> git -v --version # 'git' version 2.37.1.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/*
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a461232b0412c5e6e1e5e4 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f ba799ba7e1b576f04a461232b0412c5e6e1e5e4 # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master ba799ba7e1b576f04a461232b0412c5e6e1e5e4 # timeout=10
```

This screenshot shows the Jenkins console output for the same build, but with a different log message. It includes a warning about duplicate references and other standard build logs.

```
WARN: Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/NewDriver.html for block at line 189. Keep only the first 100 references.
WARN: Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/NewDriver.html for block at line 192. Keep only the first 100 references.
INFO: CPD Executor CPD calculation finished (done) | time=130667ms
INFO: Analysis report generated in 2637ms, dir size=128.9 MB
INFO: Analysis report compressed in 132232ms, zip size=29.7 MB
INFO: Analysis report uploaded in 11236ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube_test
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AYOJ6gSFo2hJPR00sDy
INFO: Analysis total time: 19:00.882 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 19:12.943s
INFO: Final Memory: 16M/67M
INFO: -----
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

## Step 9: After that, check the project in SonarQube.

SonarQube #38 Console [Jenkins] | localhost:9000/dashboard?id=sonarqube\_test | My Drive - Google Drive | Adv.DevOps Exp 8 - Google Docs

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

sonarqube\_test master

September 26, 2022 at 9:02 PM Version not provided

Project Settings Project Information

The Project Analysis has failed. More details available on the [Background Tasks](#) page.

QUALITY GATE STATUS: Passed (All conditions passed)

MEASURES

New Code	Overall Code
111k Bugs	Reliability (C)
0 Vulnerabilities	Security (A)
0 Security Hotspots	Reviewed
1477d Debt	Security Review (A)
142k Code Smells	Maintainability (A)
50.6% Duplications on 685k Lines	Duplicated Blocks

Under different tabs, check all different issues with the code.

## Step 10: Code Problems

### Bugs

Dashboard [Jenkins] | localhost:9000/project/issues?resolved=false&types=BUG&id=sonarqube\_test | My Drive - Google Drive | Adv.DevOps Exp 8 - Google Docs

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

sonarqube\_test master

September 26, 2022 at 9:02 PM Version not provided

Project Settings Project Information

The Project Analysis has failed. More details available on the [Background Tasks](#) page.

Issues

My Issues All

Filters

Period: New code

Type: BUG

- Bug: 111k
- Vulnerability: 0
- Code Smell: 142k

Severity:

- Blocker: 0
- Critical: 0
- Major: 46k
- Minor: 65k

Scope, Resolution, Status, Security Category, Creation Date, Language, Rule

1 / 111,356 issues | 1694d effort

Issue Description	Created	Severity	Type	Assignee	Effort	Comments
Insert a <!DOCTYPE> declaration to before this <html> tag.	2 years ago	L1	Bug	user-experience	5min effort	Comment
Add "lang" and/or "xml:lang" attributes to this "<html>" element	2 years ago	L1	Bug	accessibility, wcag2-a	2min effort	Comment
Add "<th>" headers to this "<table>".	2 years ago	L9	Bug	accessibility, wcag2-a	2min effort	Comment
Add a description to this table.	2 years ago	L9	Bug	accessibility, wcag2-a	5min effort	Comment
Add a description to this table.	2 years ago	L19	Bug	accessibility, wcag2-a	5min effort	Comment
Insert a <!DOCTYPE> declaration to before this <html> tag.	2 years ago	L1	Bug	user-experience	5min effort	Comment
Add "lang" and/or "xml:lang" attributes to this "<html>" element	2 years ago	L1	Bug	accessibility, wcag2-a	2min effort	Comment
Add "<th>" headers to this "<table>".	2 years ago	L9	Bug	accessibility, wcag2-a	2min effort	Comment

## Code smell

The screenshot shows the SonarQube Issues page for the project 'sonarqube\_test'. The page displays a list of 142k code smell issues found in the file 'gameoflife-core/build/reports/tests/all-tests.html'. The issues are listed in descending order of priority, with the first few being 'Remove this deprecated "width" attribute.' and 'Remove this deprecated "align" attribute.'. The filters sidebar on the left shows that all issues are of type 'CODE SMELL'.

Issue Description	Priority	File	Last Updated	Comments
Remove this deprecated "width" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L9, html5, obsolete
Remove this deprecated "align" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L11, html5, obsolete
Remove this deprecated "align" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L12, html5, obsolete
Remove this deprecated "size" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L17, html5, obsolete
Remove this deprecated "cellpadding" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L19, html5, obsolete
Remove this deprecated "cellspacing" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L19, html5, obsolete
Remove this deprecated "width" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L19, html5, obsolete
Remove this deprecated "valign" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L20, html5, obsolete
Remove this deprecated "width" attribute.	Major	gameoflife-core/build/reports/tests/all-tests.html	2 years ago	L24, html5, obsolete

In this way, we have created a CI/CD Pipeline with Jenkins and integrated it with SonarQube to find issues in the code like bugs, code smells, duplicates, cyclomatic complexities, etc.

\* Conclusion - We already integrated jenkins previously , with this experiment we will see how to build pipeline that will aid in SAST analysis . We discovered bugs , code smells in sample code after executing pipeline.