

**Name:** Manav Jawrani

**Roll No.:** 19

**Subject:** Advanced DevOps

**Experiment No.:** 8

## Experiment 8.

Aim: Create a Jenkins CI/CD pipeline with SonarQube / GitLab integration to perform a static analysis of code to detect bugs, code smells and security vulnerabilities on a sample web / Java / Python application.

### Theory:

- What is 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 example, there is a task, that task has got five different stages and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be completed.

- What is a CI/CD pipeline?

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. The CI/CD pipeline acts as the backbone of DevOps approach. This pipeline is responsible for building codes, running tests and deploying software version. The pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.



## • SAST with CI/CD Pipeline?

SAST tools analyze application source code to find security weaknesses or vulnerabilities that malicious content can exploit. SAST tools use a white-box approach to testing which analyzes the application from inside. Integrating SAST tools into a process is critical to building sustainable projects. SAST must be automated and must be integrated into the CI/CD toolset to improve efficiency, consistency and early detection.

SAST can be applied to all stages of software development cycle and can catch both unintentional bugs and malicious tampering. Here is how SAST can contribute at each stage of development:

1. Initial build
2. Staging and acceptance
3. Production deployment

## Implementation:

### Prerequisites:

1. Docker Installed

Download from here: <https://www.docker.com>

2. Jenkins

Download from here: <https://www.jenkins.io/download/>

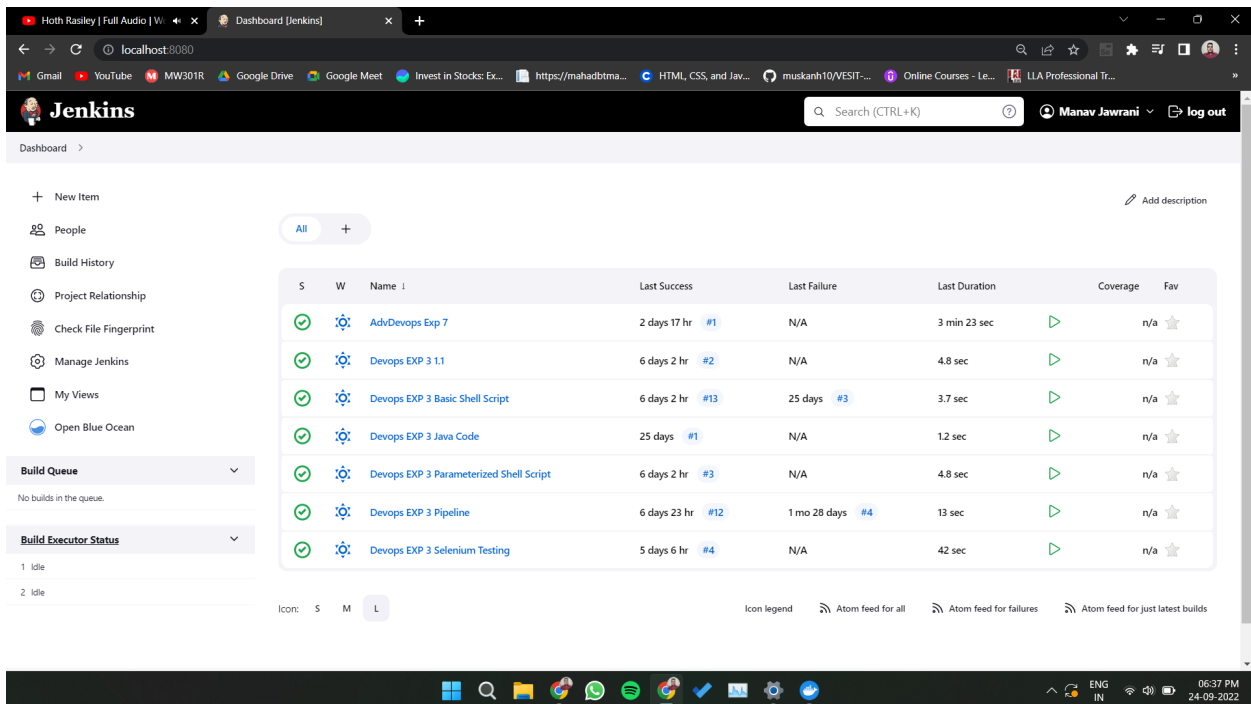
3. SonarQube Docker image.

Create image using this command:

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

### Steps

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



The screenshot shows the Jenkins Dashboard in a web browser. The dashboard has a sidebar on the left with navigation links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, and Open Blue Ocean. The main area displays a table of builds. The table has columns for S (Status), W (Workspace), Name, Last Success, Last Failure, Last Duration, Coverage, and Fav (Favorite). The builds listed are:

S	W	Name	Last Success	Last Failure	Last Duration	Coverage	Fav
✓	✓	AdvDevops Exp 7	2 days 17 hr #1	N/A	3 min 23 sec	n/a	★
✓	✓	Devops EXP 3 1.1	6 days 2 hr #2	N/A	4.8 sec	n/a	★
✓	✓	Devops EXP 3 Basic Shell Script	6 days 2 hr #13	25 days #3	3.7 sec	n/a	★
✓	✓	Devops EXP 3 Java Code	25 days #1	N/A	1.2 sec	n/a	★
✓	✓	Devops EXP 3 Parameterized Shell Script	6 days 2 hr #3	N/A	4.8 sec	n/a	★
✓	✓	Devops EXP 3 Pipeline	6 days 23 hr #12	1 mo 28 days #4	13 sec	n/a	★
✓	✓	Devops EXP 3 Selenium Testing	5 days 6 hr #4	N/A	42 sec	n/a	★





At the bottom of the dashboard, there is a section for 'Build Queue' and 'Build Executor Status'. The 'Build Queue' shows 'No builds in the queue.' and the 'Build Executor Status' shows '1 Idle' and '2 Idle'. The bottom of the browser window shows the Windows taskbar with various icons and the system clock indicating 06:37 PM on 24-09-2022.

**Step 2:** Start the Docker Container in which the SonarQube image is stored.

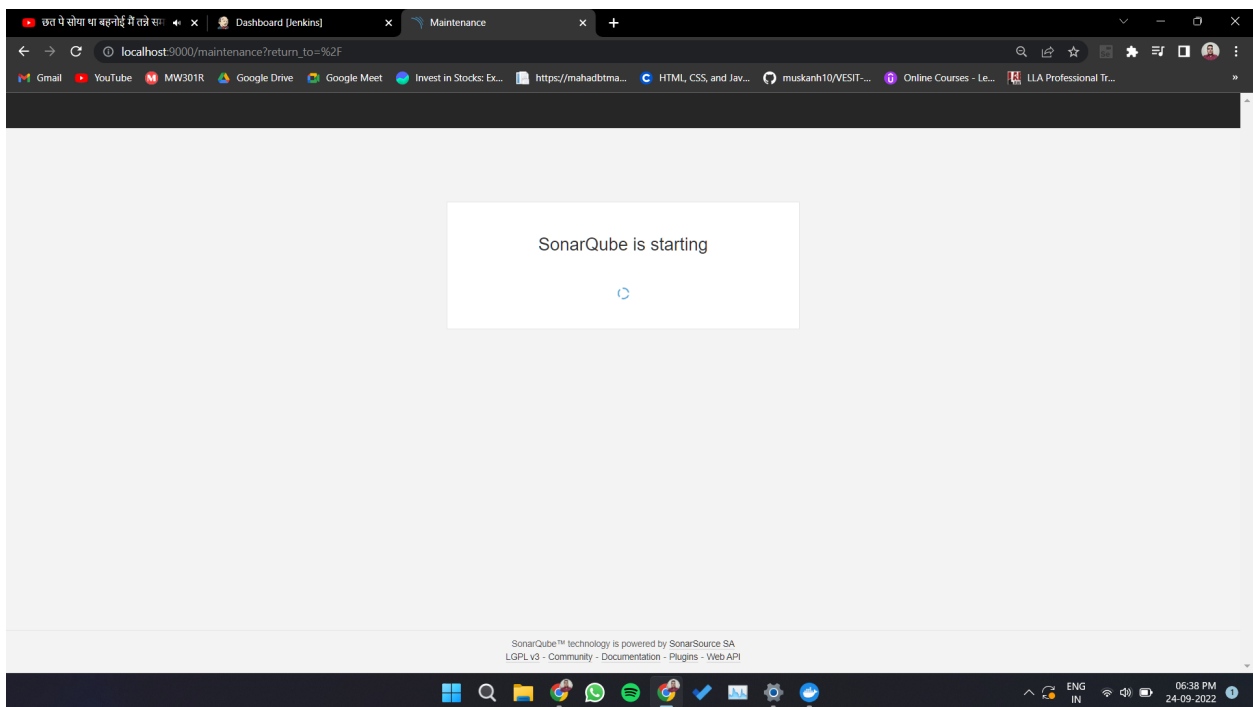
Containers [Give Feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

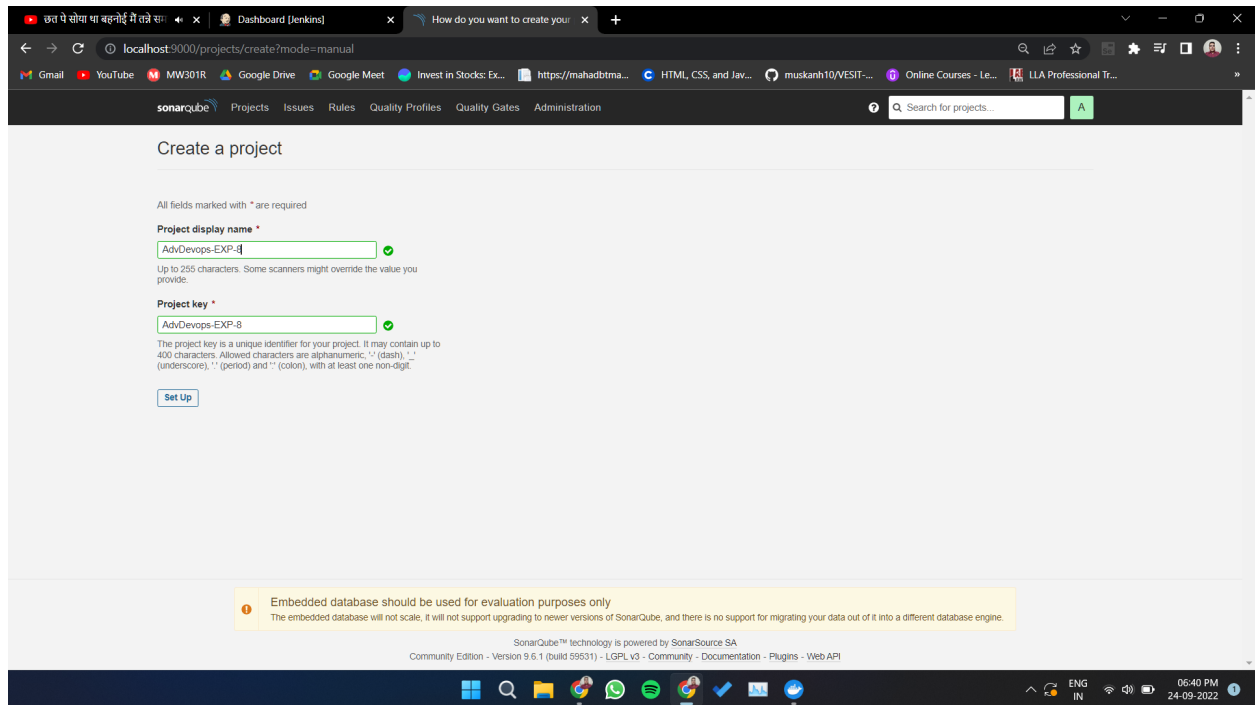
Showing 4 items

	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	 sonarqube 9b736d4dcedc	<a href="#">sonarqube:latest</a>	Running	9000	14 seconds ago	  

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



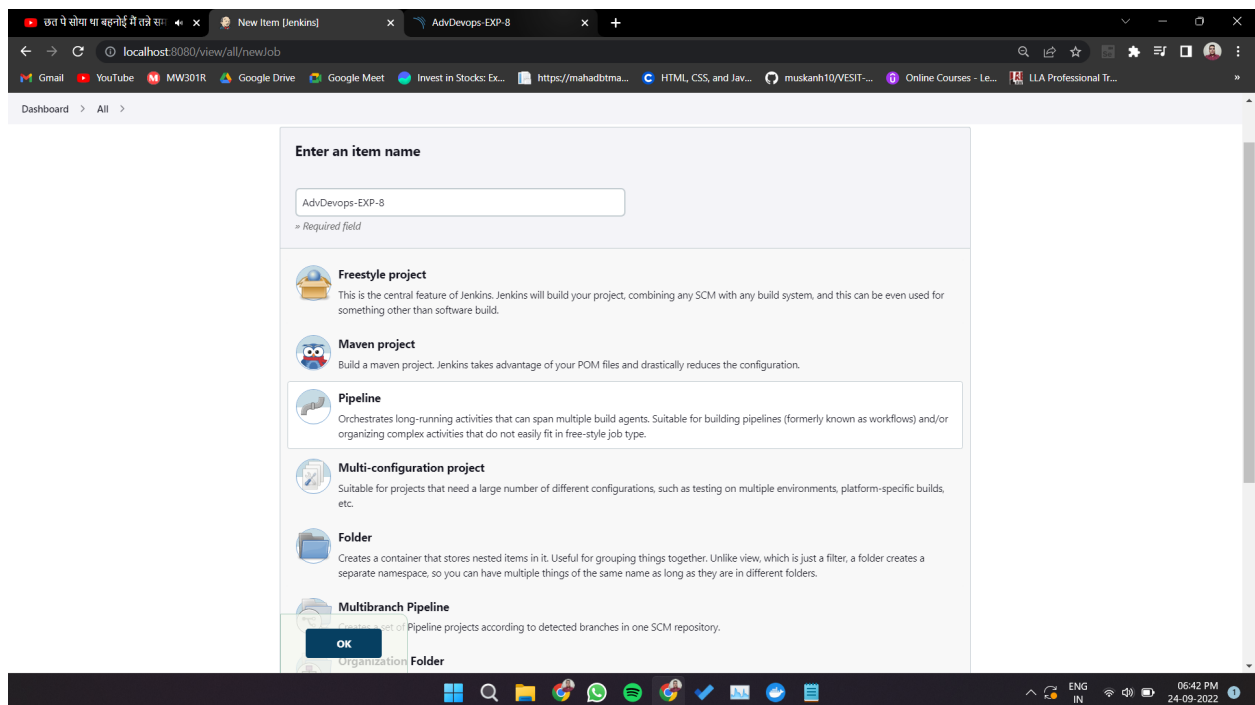
**Step 4:** Login to SonarQube using username and password and then create a manual project in SonarQube with the name “AdvDevops-EXP-8”



The screenshot shows the SonarQube web interface for creating a new project. The browser address bar shows 'localhost:9000/projects/create?mode=manual'. The page title is 'Create a project'. Below the title, there is a note: 'All fields marked with \* are required'. The form has two main fields: 'Project display name \*' and 'Project key \*'. Both fields contain the text 'AdvDevops-EXP-8' and have a green checkmark icon to their right. Below the 'Project key' field, there is a small text block explaining that the project key is a unique identifier and lists allowed characters. At the bottom of the form is a 'Set Up' button. A yellow warning banner at the bottom of the page states: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.'

Setup the project and come back to Jenkins Dashboard.

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



The screenshot shows the Jenkins 'New Item' screen. The browser address bar shows 'localhost:8080/view/all/newJob'. The page title is 'New Item [Jenkins]'. The 'Enter an item name' field contains 'AdvDevops-EXP-8'. Below this field is a list of project types: 'Freestyle project', 'Maven project', 'Pipeline', 'Multi-configuration project', 'Folder', and 'Multibranch Pipeline'. The 'Pipeline' option is highlighted with a blue border. At the bottom of the screen is an 'OK' button.

**Step 6:** Under Pipeline Script, enter the following:

```
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=*your username* \
      -D sonar.password=*your password* \
      -D sonar.projectKey=*your projectkey* \
      -D sonar.exclusions=vendor/**,resources/**,**/*.java \
      -D sonar.host.url=http://localhost:9000/"
    }
  }
}
```

## Pipeline

### Definition

Pipeline script

#### Script ?

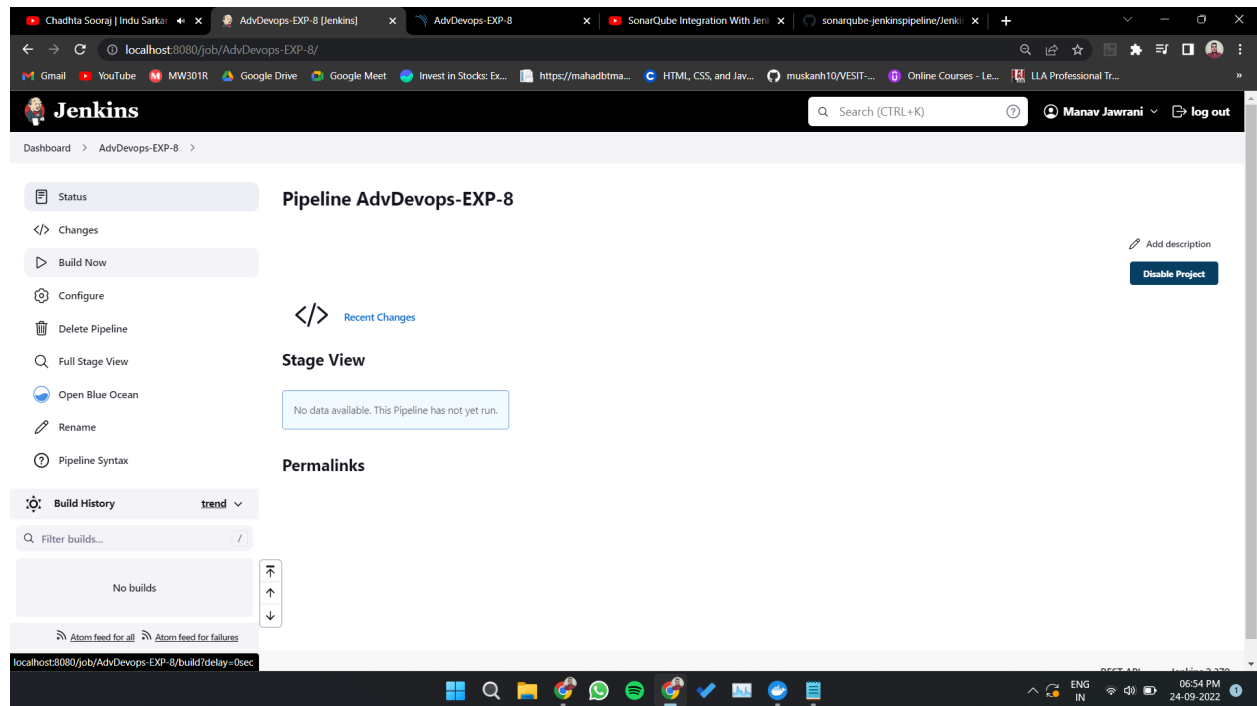
```
1 node {
2   stage('Clone the Git') {
3     git 'https://github.com/shazforiot/GOL.git'
4   }
5
6   stage('SonarQube analysis') {
7     def scannerHome = tool 'sonarqube';
8     withSonarQubeEnv('sonarqube') {
9       bat "${scannerHome}/bin/sonar-scanner \
10        -D sonar.login=admin \
11        -D sonar.password=admin123 \
12        -D sonar.projectKey=AdvDevops-EXP-8 \
13        -D sonar.exclusions=vendor/**,resources/**,**/*.java \
14        -D sonar.host.url=http://localhost:9000/"
15      }
16    }
17  }
18 }
19 }
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

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

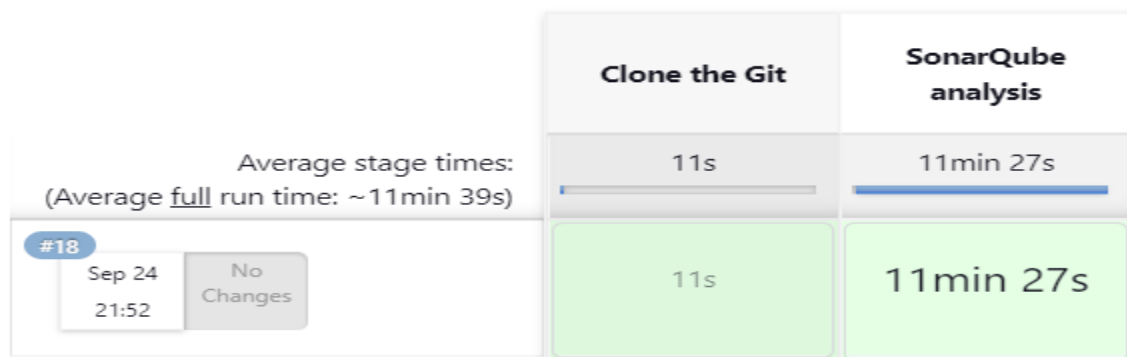
## Step 7: Run The Build.



## Pipeline AdvDevops-EXP-8

 [Recent Changes](#)

### Stage View





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

The screenshot displays the Jenkins web interface for a build named 'AdvDevs-EXP-8 #18'. The left sidebar shows navigation options: Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Git Build Data, Open Blue Ocean, Thread Dump, Pause/resume, Replay, Pipeline Steps, and Workspaces. The main area is titled 'Console Output' and shows the following log:

```
Started by user Manav Jawrani
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\workspace\AdvDevs-EXP-8
[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\workspace\AdvDevs-EXP-8\.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.exe --version # 'git version 2.37.3.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e54 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e54 # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master ba799ba7e1b576f04a4612322b0412c5e6e1e54 # timeout=10
Commit message: "Update Jenkinsfile"
First time build. Skipping changelog.
```

The second screenshot shows the continuation of the console output, including warnings about duplication references and information about the analysis report generation and upload.

```
WARN: Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/jms/sampler/QueueExecutor.html for block at line 41. Keep only the first 100 references.
WARN: Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/jms/sampler/QueueExecutor.html for block at line 17. Keep only the first 100 references.
WARN: Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/jms/sampler/QueueExecutor.html for block at line 190. Keep only the first 100 references.
INFO: CPD Executor CPD calculation finished (done) | time=205639ms
INFO: Analysis report generated in 5003ms, dir size=129.8 MB
INFO: Analysis report compressed in 23517ms, zip size=29.8 MB
INFO: Analysis report uploaded in 1084ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=AdvDevs-EXP-8
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=AThMhRexY353-5bXK-V-
INFO: Analysis total time: 11:15.621 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 11:23.516s
INFO: Final Memory: 17M/67M
INFO: -----
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

The bottom of the interface shows the REST API endpoint and the Jenkins version (2.370). The system tray at the bottom indicates the time is 10:05 PM on 24-09-2022.

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

The screenshot shows the SonarQube project dashboard for 'AdvDevs-EXP-8' on the 'master' branch. The 'QUALITY GATE STATUS' is 'Passed' with 'All conditions passed.' The 'MEASURES' section displays the following metrics:

- New Code:** Since September 24, Started 18 minutes ago.
- Overall Code:**
- 111k** Bugs (Reliability: C)
- 0** Vulnerabilities (Security: A)
- 0** Security Hotspots (Security Review: A)
- 1477d** Debt (Maintainability: A)
- 142k** Code Smells
- 50.6%** Duplications on 685k Lines
- 43k** Duplicated Blocks

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

**Step 10: Code Problems**  
**Bugs:**

The screenshot shows the SonarQube 'Issues' page for 'AdvDevs-EXP-8'. The left sidebar contains filters for 'My Issues' and 'All' tabs, with 'Bug' selected. The main area displays a list of issues, including:

- gameoflife-core/build/reports/tests/all-tests.html**
- Insert a <!DOCTYPE> declaration to before this <html> tag.** (Major, Open, Not assigned, 5min effort, Comment)
- Add "lang" and/or "xml:lang" attributes to this "html" element** (Major, Open, Not assigned, 2min effort, Comment)
- Add "th" headers to this "table".** (Major, Open, Not assigned, 2min effort, Comment)
- Add a description to this table.** (Minor, Open, Not assigned, 5min effort, Comment)
- Add a description to this table.** (Minor, Open, Not assigned, 5min effort, Comment)

# Code Smell:

The screenshot displays the SonarQube web interface for a project named 'AdvDevOps-EXP-8'. The browser address bar shows the URL: `localhost:9000/project/issues?resolved=false&types=CODE_SMELL&id=AdvDevOps-EXP-8`. The SonarQube header includes navigation links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration, along with a search bar and a user profile icon.

The main content area is titled 'AdvDevOps-EXP-8' and shows the 'Issues' tab. A warning message states: 'Last analysis of this Branch had 1 warning' on September 24, 2022 at 9:49 PM. The project summary indicates '1 / 141,803 issues' and '1477d effort'.

On the left, the 'Filters' sidebar is expanded, showing the following counts:

- Period: New code (111k)
- Type: **CODE SMELL** (142k), Bug (0), Vulnerability (0)
- Severity: Blocker (0), Critical (0), Major (142k), Minor (0), Info (2)
- Scope, Resolution, Status, Security Category, Creation Date, Language, Rule

The main list of issues shows several 'Remove this deprecated' warnings, such as 'Remove this deprecated "width" attribute' and 'Remove this deprecated "align" attribute'. Each issue entry includes a checkbox, a description, a severity level (Major), a status (Open), an assigned user (Not assigned), an effort (5min), a comment link, and a 'html5, obsolete' tag.

The Windows taskbar at the bottom shows the system clock as 10:13 PM on 24-09-2022, with the language set to ENG IN.

### Conclusion:

As we have already demonstrated how to integrate Jenkins previously, with this experiment we will now move forward and build a pipeline that will aid in SAST analysis. The pipeline will investigate, review and produce a static analysis report using the SonarQube web application's built-in features. We discovered bugs, code smells in the sample code, and some duplications in the code after executing the pipeline. At the conclusion, we learned how to use SonarQube tool going forward to analyse code and aid in the creation of better software in future.