

**Aim:** Create a Jenkins CICD 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.

1. Open up the Jenkins dashboard. Usually on localhost: 8080
2. Run SonarQube in a Docker container using this command -

```
PS C:\Windows\system32> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
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
5008a1b64daaff9ca9430d6426a4ceb303d964d039798555e9fb548ff7c89073
```

3. Once the container is running, you can check it on localhost:9000. If you have already made a container named sonarqube, run the following command instead.

```
PS C:\Windows\system32> docker start sonarqube
>>
sonarqube
```

4. Login to sonarqube with username admin, password admin, or if you have setup any other credentials use that.
5. Create a local project. Give it a name(sonarqube-test2). Configure the project and click on create.

## Create a local project

Project display name \*



Project key \*



Main branch name \*

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

Cancel

Next

6. Now we need to install the SonarQube CLI.

Go to following

link:<https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/>

Download the suitable version compatible with your device.

Extract the zip file downloaded. Inside the folder go to bin/sonar-scanner, copy the path of this file.

7. Create a new project in Jenkins. Give it a name and select the project type as pipeline in Jenkins.

### New Item

Enter an item name

advdevops8

Select an item type



#### 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.

8. Inside the pipeline script, give the following script. Use the path copied in earlier step instead of Path to sonarqube folder.

```
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/"
```

```
}  
}  
}
```

Click on save to create the pipeline.

## Pipeline

### Definition

Pipeline script ▼

Script ?

try sample Pipeline... ▼

```
1 node {  
2   stage('Cloning the GitHub Repo') {  
3     git 'https://github.com/shazforiot/GOL.git'  
4   }  
5  
6   stage('SonarQube analysis') {  
7     // Performing SonarQube analysis  
8     withSonarQubeEnv('sonarqube') {  
9       sh '''  
10        sonar-scanner \  
11        -Dsonar.login=admin \  
12        -Dsonar.password=rakshit \  
13        -Dsonar.projectKey=sonarqube-test2 \  
14        -Dsonar.projectVersion=1.0 \  
15        -Dsonar.sources=src \  
16        -Dsonar.exclusions=vendor/**,resources/**,/**/*.java \  
17        -Dsonar.host.url=http://127.0.0.1:9000  
18      '''  
19    }  
20  }  
21 }
```

☒ 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.

9. Run the build.



Status

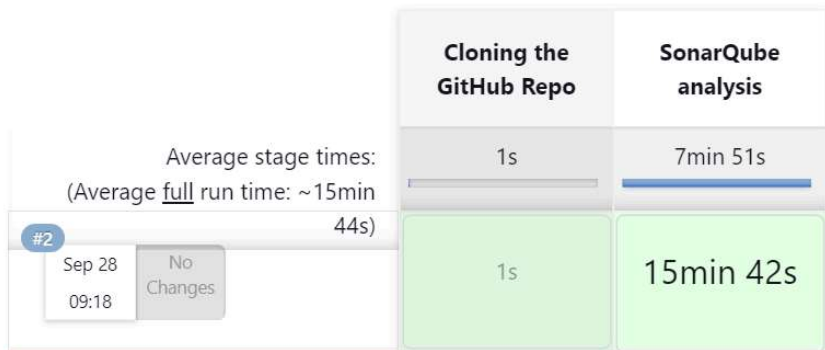


Changes



Build Now

## Stage View



## 10. Check the console output



## Console Output

[Download](#)[Copy](#)[View as plain text](#)Skipping 4,249 KB.. [Full Log](#)

```
09:27:10.137 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 215. Keep
only the first 100 references.
09:27:10.137 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 296. Keep
only the first 100 references.
09:27:10.137 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 17. Keep
only the first 100 references.
09:27:10.137 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 215. Keep
```

```
09:27:15.430 INFO CPU Executor CPU calculation finished (done) | time=28/036ms
09:27:15.451 INFO SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
09:30:22.050 INFO Analysis report generated in 138747ms, dir size=127.2 MB
09:33:36.168 INFO Analysis report compressed in 194118ms, zip size=29.6 MB
09:33:36.981 INFO Analysis report uploaded in 813ms
09:33:36.982 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-test2
09:33:36.982 INFO Note that you will be able to access the updated dashboard once the server has processed the
submitted analysis report
09:33:36.982 INFO More about the report processing at http://127.0.0.1:9000/api/ce/task?id=0fd50499-d2c7-460f-a5c0-06a3b9908b8b
09:33:43.711 INFO Analysis total time: 15:38.015 s
09:33:43.729 INFO SonarScanner Engine completed successfully
09:33:44.597 INFO EXECUTION SUCCESS
09:33:44.597 INFO Total time: 15:41.397s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

11. Since the build is successful, now check the project in sonarqube.

The screenshot shows the SonarQube interface for a project named 'sonarqube-test2'. The main section displays a green checkmark and the word 'Passed' under the 'Quality Gate' section, indicating a successful analysis. Below this, a warning message states 'The last analysis has warnings. See details'. The 'Overall Code' tab is selected, showing three categories of issues: Security (0 Open Issues, grade A), Reliability (68k Open Issues, grade C), and Maintainability (164k Open Issues, grade A). Each category has a breakdown of issues by severity: High (H), Medium (M), and Low (L).

Category	Open Issues	Grade	Severity Breakdown
Security	0	A	0 H, 0 M, 0 L
Reliability	68k	C	0 H, 47k M, 21k L
Maintainability	164k	A	7 H, 143k M, 21k L

You will be able to see different issues with the code under different tabs listed.

12. Code problems:

Reliability:

The screenshot shows a list of Reliability issues. The first issue is 'Add "lang" and/or "xml:lang" attributes to this "<html>" element', categorized under 'Intentionality' with a 'Reliability' grade. It is marked as 'Open' and 'Not assigned'. The second issue is 'Insert a <!DOCTYPE> declaration to before this <html> tag.', categorized under 'Consistency' with a 'Reliability' grade. It is also marked as 'Open' and 'Not assigned'. The third issue is 'Add "<th>" headers to this "<table>".', categorized under 'Intentionality' with a 'Reliability' grade. It is marked as 'Open' and 'Not assigned'. Each issue entry includes a brief description, a grade, a severity level (L1), effort (2min or 5min), age (4 years ago), and a bug type (Bug, Major).

Issue Description	Category	Grade	Status	Severity	Effort	Age	Bug Type
Add "lang" and/or "xml:lang" attributes to this "<html>" element	Intentionality	Reliability	Open	Not assigned	L1	2min effort	4 years ago
Insert a <!DOCTYPE> declaration to before this <html> tag.	Consistency	Reliability	Open	Not assigned	L1	5min effort	4 years ago
Add "<th>" headers to this "<table>".	Intentionality	Reliability	Open	Not assigned	L1	5min effort	4 years ago

## Maintainability:

☐ Bulk Change    Select issues ▲ ▼    Navigate to issue ◀ ▶    163,781 issues    1705d effort

gameoflife-acceptance-tests/Dockerfile




☐ Use a specific version tag for the image.    Intentionality  
Maintainability ▲    No tags +  
○ Open ▼    Not assigned ▼    L1 • 5min effort • 4 years ago • ☹ Code Smell • ⚠ Major

☐ Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.    Intentionality  
Maintainability ▲    No tags +  
○ Open ▼    Not assigned ▼    L12 • 5min effort • 4 years ago • ☹ Code Smell • ⚠ Major

## Duplications:

sonarqube-test2    View as List ▼    Select files ▼ ▲    Navigate ◀ ▶    1,147 files

Duplicated Lines (%) 50.6%    [See history](#)

	Duplicated Lines (%)	Duplicated Lines
 gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../ReportCellRenderrer.html	92.4%	1,282
 gameoflife-web/tools/jmeter/docs/api/org/apache/jo.../RightAlignRenderrer.html	92.4%	1,198
 gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../JMeterCellRenderrer.html	92.1%	1,281

## Bug:



The screenshot shows the SonarQube project dashboard for the path `gameoflife-core/build/reports/tests/all-tests.html`. At the top, there are controls for "Bulk Change", "Select issues", and "Navigate to issue", along with statistics: "46,515 issues" and "1426d effort".

Two issues are listed:

- Issue 1:** `Add "lang" and/or "xml:lang" attributes to this "<html>" element`. It is categorized under "Intentionality" with a "Reliability" tag. The status is "Open" and "Not assigned". It is labeled as "L1", "2min effort", "4 years ago", "Bug", and "Major".
- Issue 2:** `Insert a <!DOCTYPE> declaration to before this <html> tag.`. It is categorized under "Consistency" with a "Reliability" tag. The status is "Open" and "Not assigned". It is labeled as "L1", "5min effort", "4 years ago", "Bug", and "Major".

#### Code smells:

The screenshot shows the SonarQube project dashboard for the path `gameoflife-acceptance-tests/Dockerfile`. At the top, there are controls for "Bulk Change", "Select issues", and "Navigate to issue", along with statistics: "164,034 issues" and "1708d effort".

Two issues are listed:

- Issue 1:** `Use a specific version tag for the image.`. It is categorized under "Intentionality" with a "Maintainability" tag. The status is "Open" and "Not assigned". It is labeled as "L1", "5min effort", "4 years ago", "Code Smell", and "Major".
- Issue 2:** `Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.`. It is categorized under "Intentionality" with a "Maintainability" tag. The status is "Open" and "Not assigned". It is labeled as "L12", "5min effort", "4 years ago", "Code Smell", and "Major".

And various other issues or problems can be seen in the sonarqube project dashboard.

#### Conclusion:

In this experiment, a Jenkins CI/CD pipeline was successfully integrated with SonarQube to perform static code analysis on a sample application. SonarQube, running in a Docker



container, analyzed the code for issues like bugs, code smells, and security vulnerabilities. The pipeline was automated by cloning the source code from a GitHub repository and using the SonarQube Scanner to detect potential code problems. After a successful Jenkins build, the SonarQube project dashboard provided information about code's reliability, maintainability, duplications, and detected bugs, offering feedback that help improve the code quality.