

## ADVANCED DEVOPS EXP 8

**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.

### THEORY:

#### Static Application Security Testing (SAST)

SAST is a methodology for testing an application's source code to identify security vulnerabilities before the code is compiled. This type of testing, also referred to as white-box testing, helps improve application security by finding weaknesses early in development.

#### Problems SAST Solves

- **Early Detection:** SAST finds vulnerabilities early in the Software Development Life Cycle (SDLC), allowing developers to fix issues without affecting builds or passing vulnerabilities to the final release.
- **Real-Time Feedback:** Developers receive immediate feedback during coding, helping them address security issues before moving to the next stage of development.
- **Graphical Representations:** SAST tools often provide visual aids to help developers navigate the code and identify the exact location of vulnerabilities, offering suggestions for fixes.
- **Regular Scanning:** SAST tools can be configured to scan code regularly, such as during daily builds, code check-ins, or before releases.

#### Importance of SAST

- **Resource Efficiency:** With a larger number of developers than security experts, SAST allows full codebase analysis quickly and efficiently, without relying on manual code reviews.
- **Speed:** SAST tools can analyze millions of lines of code within minutes, detecting critical vulnerabilities such as buffer overflows, SQL injection, and cross-site scripting (XSS) with high accuracy.

#### CI/CD Pipeline

A Continuous Integration/Continuous Delivery (CI/CD) pipeline is a sequence of automated tasks designed to build, test, and deploy new software versions rapidly and consistently. It plays a crucial role in DevOps practices, ensuring fast and reliable software releases.

#### SonarQube

SonarQube is an open-source platform from SonarSource that performs continuous code quality inspections through static code analysis. It identifies bugs, code smells, security vulnerabilities, and code

duplications in a wide range of programming languages. SonarQube is extendable with plugins and integrates seamlessly into CI/CD pipelines.

### Benefits of SonarQube

- **Sustainability:** By reducing complexity and vulnerabilities, SonarQube extends the lifespan of applications and helps maintain cleaner code.
- **Increased Productivity:** SonarQube minimizes maintenance costs and risks, resulting in fewer code changes and a more stable codebase.
- **Quality Code:** Ensures code quality checks are integrated into the development process.
- **Error Detection:** Automatically identifies coding errors and alerts developers to resolve them before moving to production.
- **Consistency:** Helps maintain consistent code quality by detecting and reporting violations of coding standards.
- **Business Scaling:** SonarQube supports scaling as the business grows without any restrictions.

### Implementation:

#### Prerequisites

1. Jenkins installed on your machine.
2. Docker installed to run SonarQube.
3. SonarQube installed via Docker

#### 1. Set Up Jenkins

- Open Jenkins Dashboard on localhost:8080 or your configured port
- Install the necessary plugins:
  - SonarQube Scanner Plugin

#### 2. Run SonarQube in Docker

Run the following command to start SonarQube in a Docker container:

command :

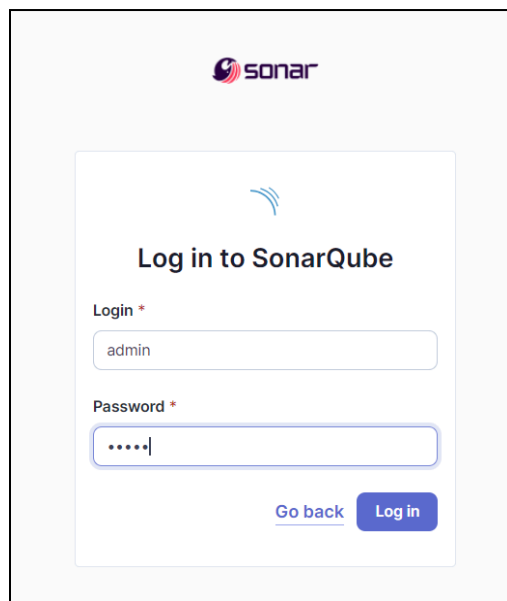
```
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

- Check SonarQube status at <http://localhost:9000>.
- Login with your credentials:

```
Windows PowerShell
PS C:\Users\Shravani> 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
363688d3e15776f8bab845a87aeade59785f0df3b80d4bba6039fdbba9d553011
PS C:\Users\Shravani>
```

### 3. Create a Project in SonarQube

- Go to Projects > Create Project.
- Name the project (e.g., sonarqube-test)



### 4. Generate SonarQube Token

- Go to My Account > Security > Generate Tokens.
- Copy the generated token for later use


## 5. Create a Jenkins Pipeline

- Go to Jenkins Dashboard, click New Item, and select Pipeline.


### New Item

Enter an item name


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.



**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

OK

## 6. Under Pipeline Script, enter the following script:

```

pipeline {
  agent any

  stages {
    stage('Create Docker Network') {
      steps {
        script {
          bat 'docker network rm sonarnet || echo "Network not found, creating a new one."'
          bat 'docker network create sonarnet'
        }
      }
    }
  }

  stage('Cloning the GitHub Repo') {
    steps {
      git 'https://github.com/shazforiot/GOL.git'
    }
  }
}

```

```

    }
  }

  stage('SonarQube analysis') {
    steps {
      withSonarQubeEnv('sonarqube') {
        bat """
        docker run --rm --network sonarnet ^
        -e SONAR_HOST_URL=http://192.168.133.16:9000 ^
        -e SONAR_LOGIN=admin ^
        -e SONAR_PASSWORD=Shravani@0212 ^
        -e SONAR_PROJECT_KEY=sonarqube-test ^
        -v ${WORKSPACE}:/usr/src ^
        sonarsource/sonar-scanner-cli ^
        -Dsonar.projectKey=sonarqube-test ^
        -Dsonar.exclusions=vendor/**,resources/**,*/*.java ^
        -Dsonar.login=admin ^
        -Dsonar.password=Shravani@0212
        """
      }
    }
  }
}

```

Pipeline

Definition

Pipeline script

Script ?

```

1 pipeline {
2   agent any
3
4   stages {
5     stage('Create Docker Network') {
6       steps {
7         script {
8           bat 'docker network rm sonarnet || echo "Network not found, creating a new one."'
9           bat 'docker network create sonarnet'
10        }
11      }
12    }
13
14    stage('Cloning the GitHub Repo') {
15      steps {
16        git 'https://github.com/shazforiot/GOL.git'
17      }
18    }
19  }
20 }

```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save
Apply

## 7. Run the Pipeline

- Save the pipeline and click Build Now
- Monitor the console output for any errors

Dashboard > SonarPipeline > #13

Status Console Output Changes Edit Build Information Timings Git Build Data Pipeline Overview Pipeline Console Thread Dump Pause/resume Replay Pipeline Steps Workspaces Previous Build

Download Copy View as plain text

Started by user Shravani Rasam

```
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\SonarPipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Create Docker Network)
[Pipeline] script
[Pipeline] {
[Pipeline] bat

C:\ProgramData\Jenkins\jenkins\workspace\SonarPipeline>docker network rm sonarnet || echo "Network not found, creating a new one."
sonarnet
[Pipeline] bat

C:\ProgramData\Jenkins\jenkins\workspace\SonarPipeline>docker network create sonarnet
80d9792e98edf5c4c1ebd1e64aa20408da61adf20a61f92ab0bacab12b00206
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
```

Dashboard > SonarPipeline >

Status SonarPipeline

Changes Build Now Configure Delete Pipeline Full Stage View Stages Rename Pipeline Syntax

Build History trend

Filter...

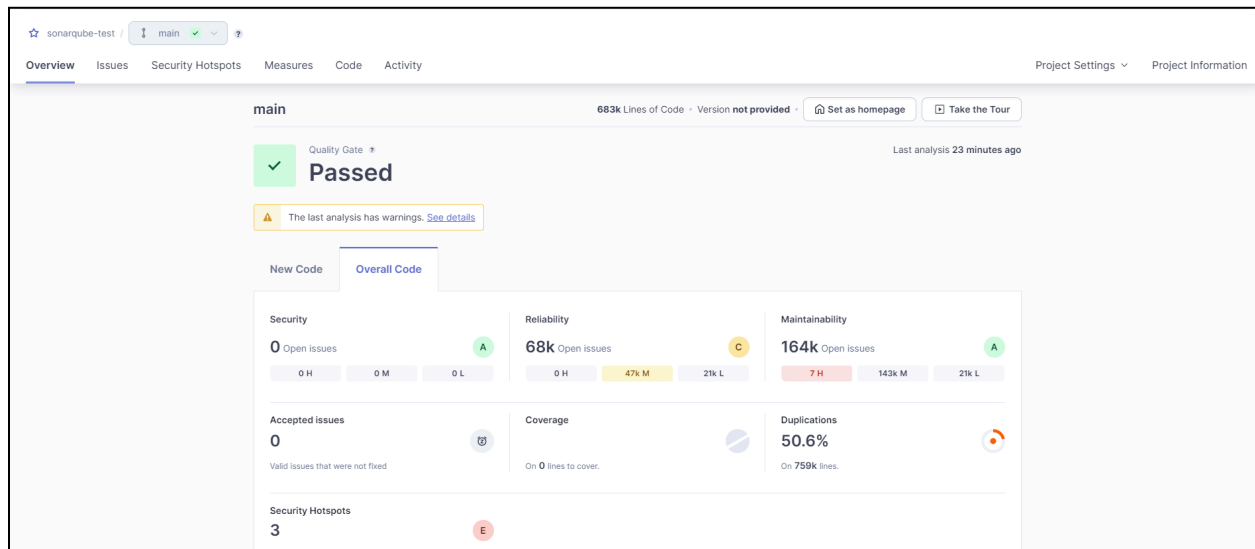
Stage View

Average stage times:  
(Average full run time: ~10min 57s)

	Create Docker Network	Cloning the GitHub Repo	SonarQube analysis
#13 Sep 27 01:14 No Changes	1s	2s	10min 51s
#12 Sep 27 01:05 No Changes	1s	2s	2min 31s failed
#11 Sep 27 00:48 No Changes	1s	1s	2min 51s aborted

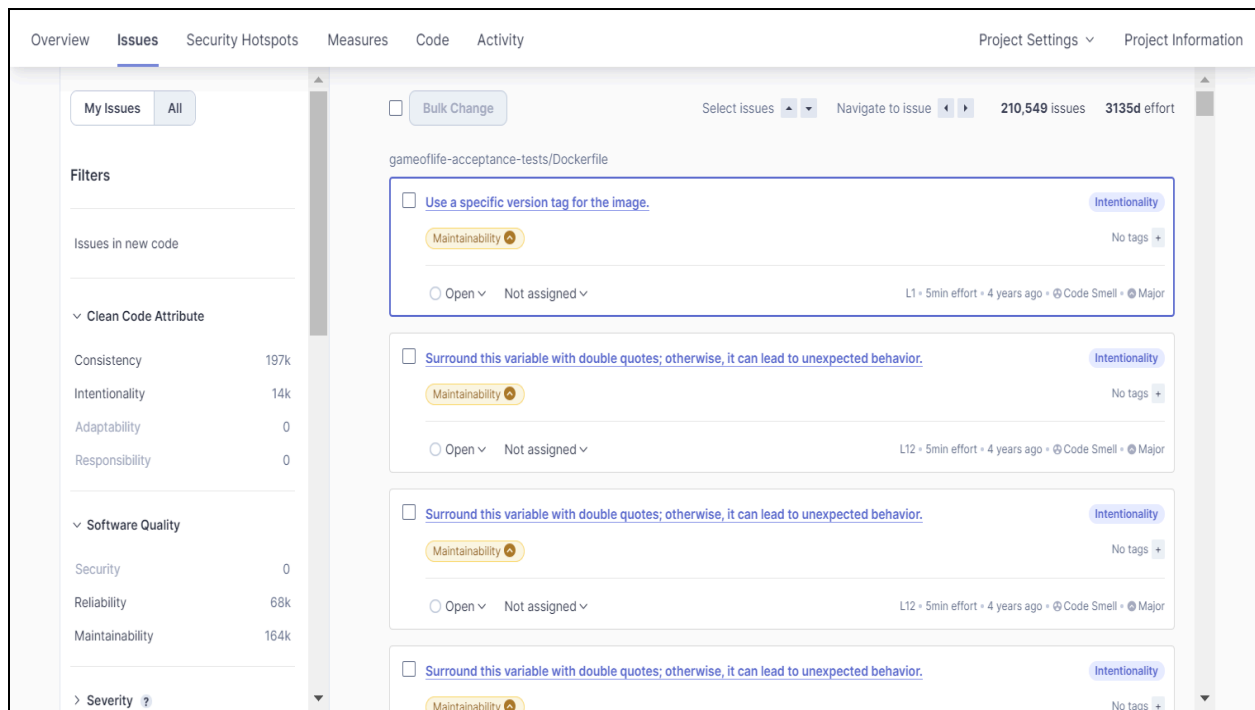
## 9. Check SonarQube for Analysis Results

- Go to your SonarQube dashboard and check the project for issues such as bugs, code smells, and security vulnerabilities.



## 10. Checking SonarQube for Analysis Results of a Code File with Bugs , Code Smells, Security Vulnerabilities, Cyclomatic Complexities and Duplicates .

- Issues -



sonarqube-test / main

Overview Issues **Security Hotspots** Measures Code Activity Project Settings Project Information

0.0% Security Hotspots Reviewed

To review Acknowledged Fixed Safe

3 Security Hotspots

Review priority: Medium

Permission 1

The tomcat image runs with root as the default user. Make sure it is safe here.

Review priority: Low

Encryption of Sensitive Data 1

Others 1

3 of 3 shown

**The tomcat image runs with root as the default user. Make sure it is safe here.**

Running containers as a privileged user is security-sensitive [docker:S6471](#)

Status: To review  
This security hotspot needs to be reviewed to assess whether the code poses a risk. [Review](#)

Where is the risk? What's the risk? Assess the risk How can I fix it? Activity

gameoflife-web/Dockerfile [Open in IDE](#)

```
1 FROM tomcat:8-jre8
2
3
4 RUN rm -rf /usr/local/tomcat/webapps/*
5
6 COPY target/gameoflife.war /usr/local/tomcat/webapps/ROOT.war
7
8 EXPOSE 8080
9 CMD ["catalina.sh", "run"]
```

The tomcat image runs with root as the default user. Make sure it is safe here.

sonarqube-test / main

Overview **Issues** Security Hotspots Measures Code Activity Project Settings Project Information

> Severity ?

Type 1

Bug 47k

Vulnerability 0

Code Smell 164k

Add to selection Ctrl + click

> Scope

> Status

> Security Category

> Creation Date

☐ Bulk Change Select issues Navigate to issue 46,515 issues 1426d effort

gameoflife-core/build/reports/tests/all-tests.html

☐ Insert a <!DOCTYPE> declaration to before this <html> tag. Consistency

Reliability user-experience

Open Not assigned L1 • 5min effort • 4 years ago • Bug • Major

☐ Add "lang" and/or "xml:lang" attributes to this "<html>" element Intentionality

Reliability accessibility wcag2-a

Open Not assigned L1 • 2min effort • 4 years ago • Bug • Major

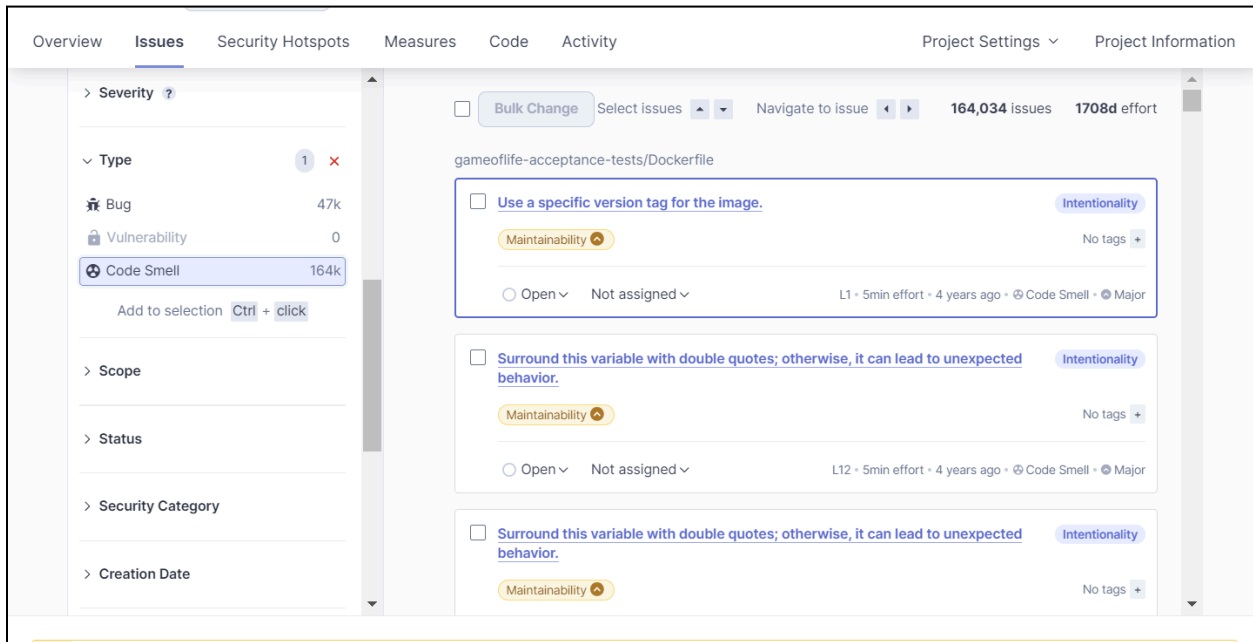
☐ Add "<th>" headers to this "<table>". Intentionality

Reliability accessibility wcag2-a

Open Not assigned L9 • 2min effort • 4 years ago • Bug • Major

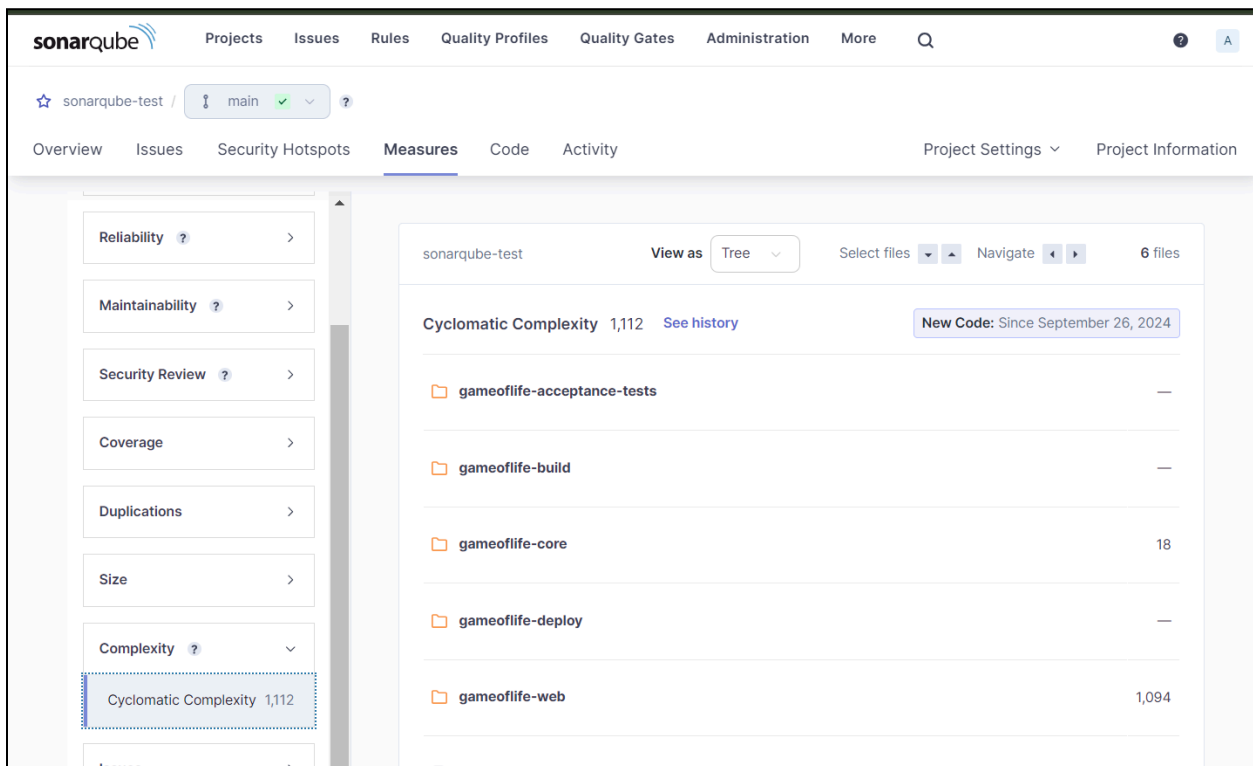


- Security Hotspot (Security Vulnerabilities) -



The screenshot displays the SonarQube interface for Security Hotspots. The left sidebar shows filters for Severity (Type, Scope, Status, Security Category, Creation Date) and a list of issues: Bug (47k), Vulnerability (0), and Code Smell (164k). The main panel shows a list of hotspots for the project 'gameoflife-acceptance-tests/Dockerfile'. The first hotspot is 'Use a specific version tag for the image.' with a Maintainability score of 1 and a severity of Major. The second and third hotspots are 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a Maintainability score of 12 and a severity of Major. The interface includes a 'Bulk Change' button, 'Select issues' dropdown, and 'Navigate to issue' buttons. The top right shows '164,034 issues' and '1708d effort'.

## Cyclomatic Complexity -



The screenshot displays the SonarQube interface for Measures. The left sidebar shows filters for Reliability, Maintainability, Security Review, Coverage, Duplications, Size, and Complexity. The main panel shows the 'Cyclomatic Complexity' measure for the project 'sonarqube-test'. The complexity score is 1,112, and the 'New Code' is dated 'Since September 26, 2024'. The interface includes a 'View as' dropdown set to 'Tree', 'Select files' dropdown, and 'Navigate' buttons. The top right shows '6 files'.

File	Cyclomatic Complexity
gameoflife-acceptance-tests	—
gameoflife-build	—
gameoflife-core	18
gameoflife-deploy	—
gameoflife-web	1,094

The screenshot shows the SonarQube interface for a project named 'sonarqube-test'. The 'Measures' tab is selected, and the 'Cyclomatic Complexity' measure is displayed for the 'gameoflife-acceptance-tests > Dockerfile'. The complexity score is 9090, which is highlighted in red, indicating a high level of complexity. The left sidebar shows various quality gates: Security, Reliability, Maintainability, Security Review, Coverage, Duplications, Size, Complexity, and Issues. The main content area displays the Dockerfile code with line numbers 1 through 23. The code includes instructions for setting the base image, environment variables, user, and running commands to install dependencies and build the application.

```

1 shazfo... FROM selenium/standalone-firefox:latest
2
3 ENV MAVEN_VERSION 3.3.3
4 ENV DISPLAY :99
5
6 USER root
7
8 RUN apt-get update -qqy \
9   && apt-get install -y openjdk-8-jdk && \
10  rm -rf /var/lib/apt/lists/*
11
12 RUN wget -O- http://archive.apache.org/dist/maven/maven-3/$MAVEN_VERSION
13   /binaries/apache-maven-$MAVEN_VERSION-bin.tar.gz | tar xzf - -C /opt \
14   && mv /opt/apache-maven-$MAVEN_VERSION /opt/maven \
15   && ln -s /opt/maven/bin/mvn /usr/bin/mvn
16
17 USER seluser
18
19 ENV MAVEN_HOME /opt/maven
20
21 EXPOSE 9090
22
23 CMD ["mvn"]

```

## • Duplications -

The screenshot shows the SonarQube interface for the same project, but now the 'Duplications' measure is selected. The 'Overview' sub-tab is active, showing a table with the following data:

New Code	
Duplicated Lines	0
Duplicated Blocks	0
Overall Code	
Density	0.0%
Duplicated Lines	0

The main content area still displays the Dockerfile code, which is identical to the previous screenshot. The left sidebar shows the 'Duplications' measure selected, with a dropdown menu showing 'Overview' and 'New Code'.

## Conclusion:

In this experiment, we performed a static analysis of the code to detect bugs, code smells, and security vulnerabilities on our sample codes.