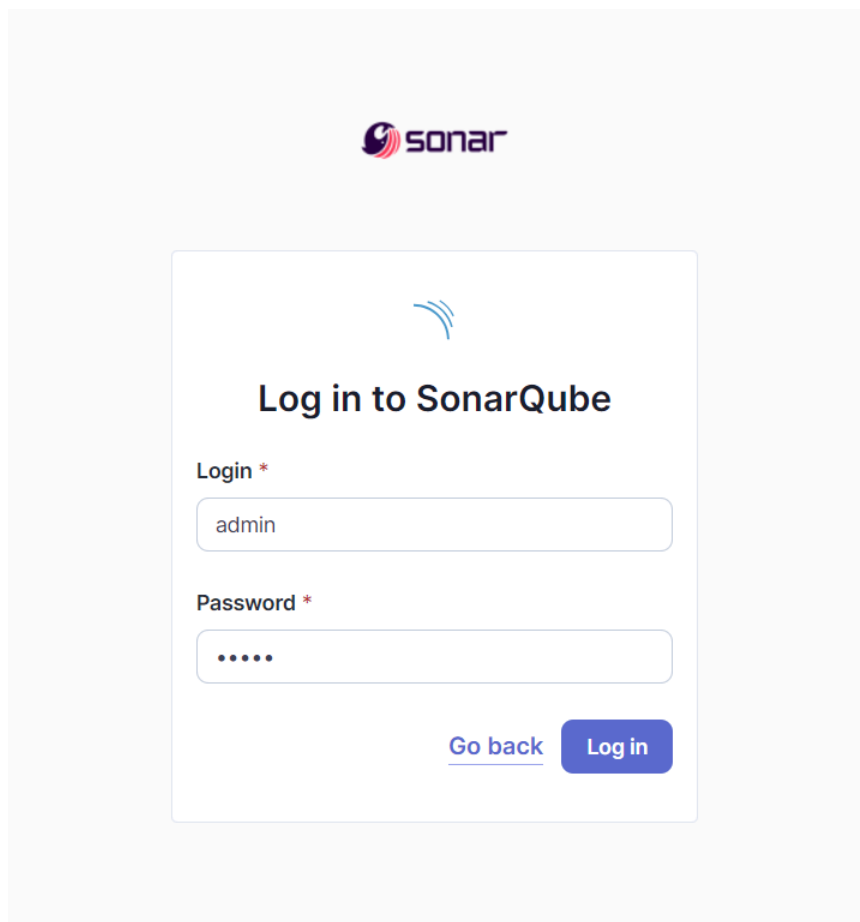


Experiment 8

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.
2. Run SonarQube in a Docker container using this command -

```
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
b6dd73afc810a20ec3d643e9a148ab9643a3b5beff2766406df21f5f54a090c1
```

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



4. Create a new project

1 of 2

Create a local project

Project display name *

sonarqube-test



Project key *

sonarqube-test



Main branch name *

main

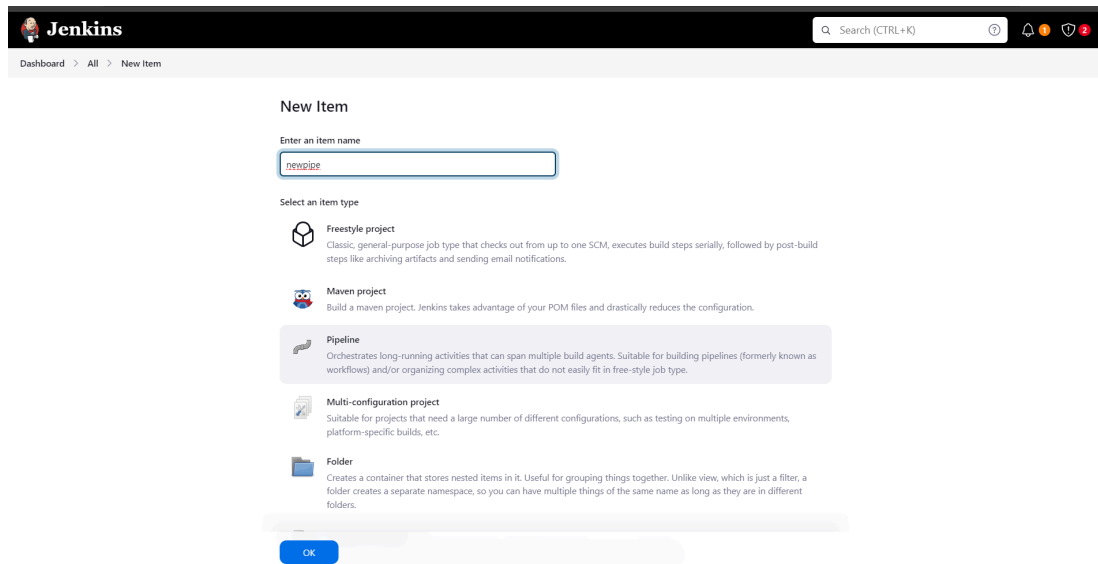
The name of your project's default branch [Learn More](#)

Cancel

Next

5. Click on Security and create a new name and token, select type as user token

6. Create a new pipeline



7. Go to sonarqube website and download the SonarScanner CLI

SonarScanner CLI

SonarScanner

Issue Tracker

Show more ▾

6.2

2024-09-17

Support PKCS12 truststore generated with OpenSSL

Download scanner for: [Linux x64](#) [Linux AArch64](#) [Windows x64](#) [macOS x64](#) [macOS AArch64](#) [Docker](#)

[Any \(Requires a pre-installed JVM\)](#)

[Release notes](#)

Once it is downloaded, copy its path and add to the script

8. Inside the pipeline section add the following script

```
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/"
    }
  }
}
```

Pipeline

Definition

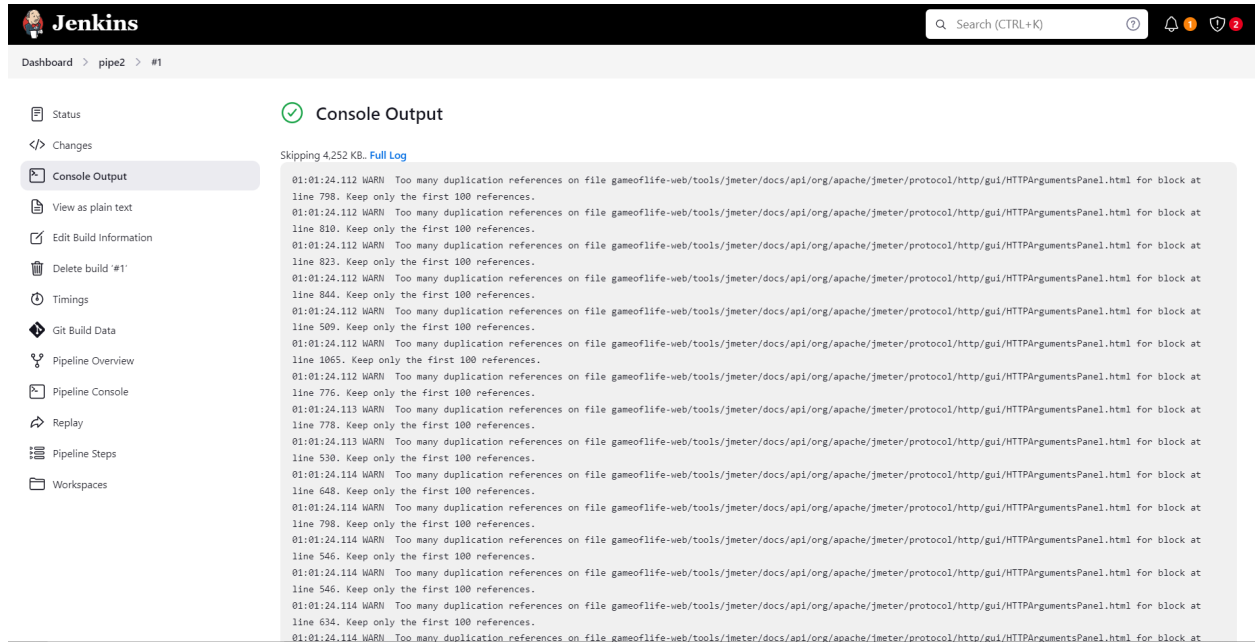
Pipeline script ▾

Script ?

```
1- node {
2-   stage('Cloning the GitHub Repo') {
3-     git 'https://github.com/shazforiot/GOL.git'
4-   }
5-   stage('SonarQube analysis') {
6-     withSonarQubeEnv('sonarqube-test') { // Use the correct name here
7-       bat """
8-       C:\Users\veyda\Desktop\sonar-scanner-6.2.0.4584-windows-x64\bin\sonar-scanner.bat \
9-       -Dsonar.login=squ_5f856d0ac21503b7232d5c635fc63663e9b4ddfd \
10-       -Dsonar.projectKey=sonarqube-test \
11-       -Dsonar.exclusions=vendor/**,resources/**,**/*.java \
12-       -Dsonar.host.url=http://localhost:9000/
13-       """
14-     }
15-   }
16- }
17-
```

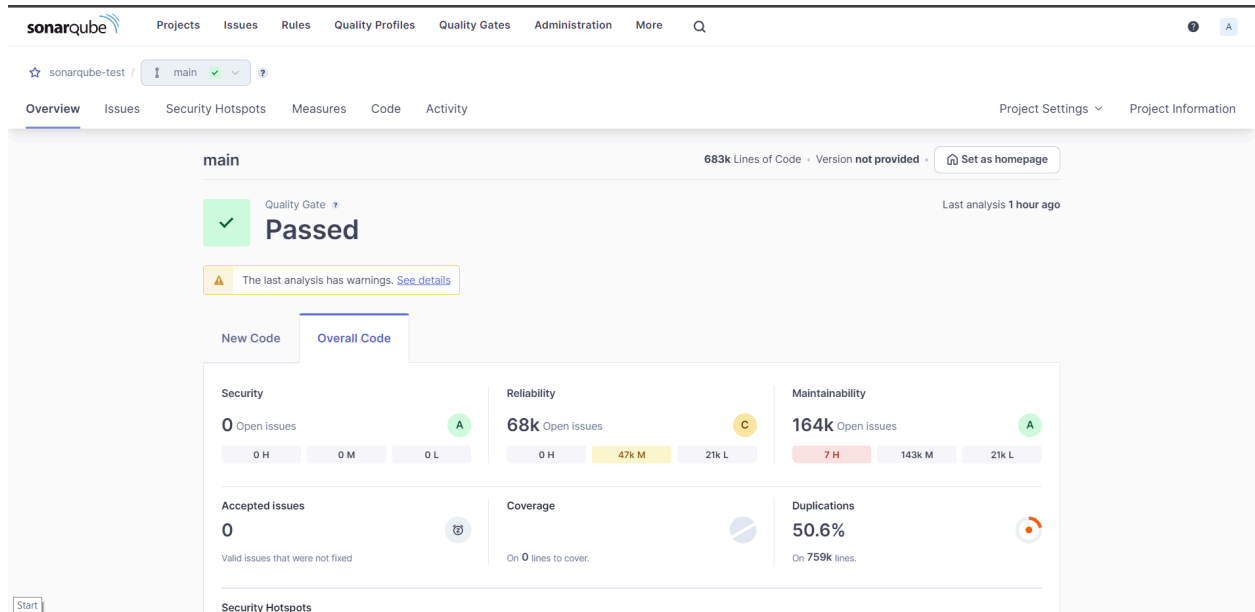
☒ Use Groovy Sandbox ?

9. Save and build the pipeline, this step may take a while. After the build is successful, check the console output



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and notification icons. The breadcrumb trail is 'Dashboard > pipe2 > #1'. The left sidebar contains a list of links: Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete build '#1', Timings, Git Build Data, Pipeline Overview, Pipeline Console, Replay, Pipeline Steps, and Workspaces. The main content area is titled 'Console Output' with a green checkmark icon. It displays a log of build output, starting with 'Skipping 4.252 KB. Full Log'. The log contains multiple warning messages: '01:01:24.112 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 798. Keep only the first 100 references.' This message is repeated for lines 810, 823, 844, 509, 1065, 776, 778, 530, 648, 798, 546, 634, and 114.

10. Check the sonarqube project, whether it has passed or not.



The screenshot shows the SonarQube web interface for the 'sonarqube-test' project. The top navigation bar includes the SonarQube logo and links to Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The breadcrumb trail is 'sonarqube-test / main'. The main content area is titled 'main' and shows '683k Lines of Code' and 'Version not provided'. A green checkmark icon indicates the project has passed the Quality Gate. A warning message states: 'The last analysis has warnings. See details'. The 'Overall Code' tab is selected, showing a summary of metrics: Security (0 Open Issues, A grade), Reliability (68k Open Issues, C grade), Maintainability (164k Open Issues, A grade), Accepted Issues (0), Coverage (50.6%), and Duplications (759k lines). The 'Security Hotspots' section is also visible at the bottom.

Conclusion:

- We created a sonarqube project locally
- We created a jenkins pipeline and added a script with github link
- We also added sonarqube project details like key, token etc.
- Many issues were faced with the script as the build was initially failing, later succeeded with some changes.
- The build was successful on jenkins and passed on sonarqube