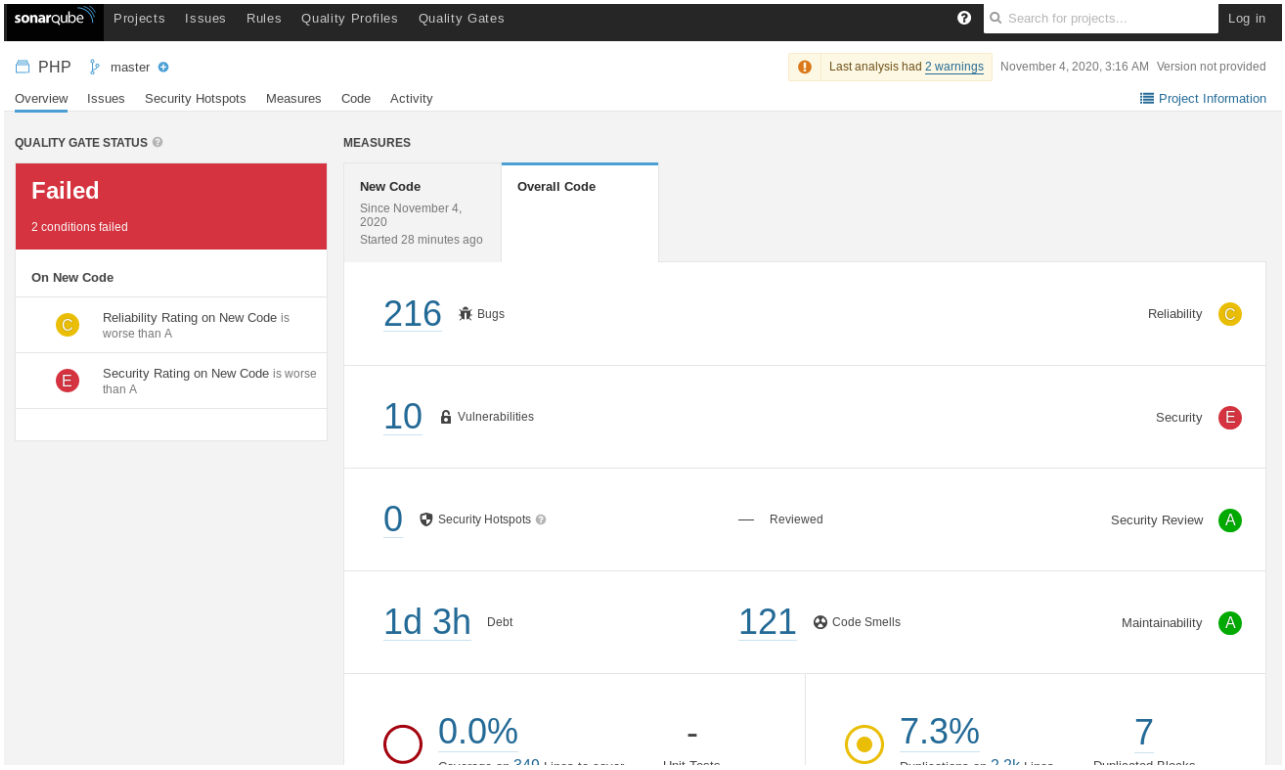
Static analysis using SonarQube

Overview



In this lab, you are going to learn how to use SonarQube to generate static code analysis report.

SonarQube includes support for the programming languages Java (including Android), C#, PHP, JavaScript, TypeScript, C/C++, Ruby, Kotlin, Go, COBOL, PL/SQL, PL/I, ABAP, VB.NET, VB6, Python, RPG, Flex, Objective-C, Swift, CSS, HTML, and XML.

Outcomes

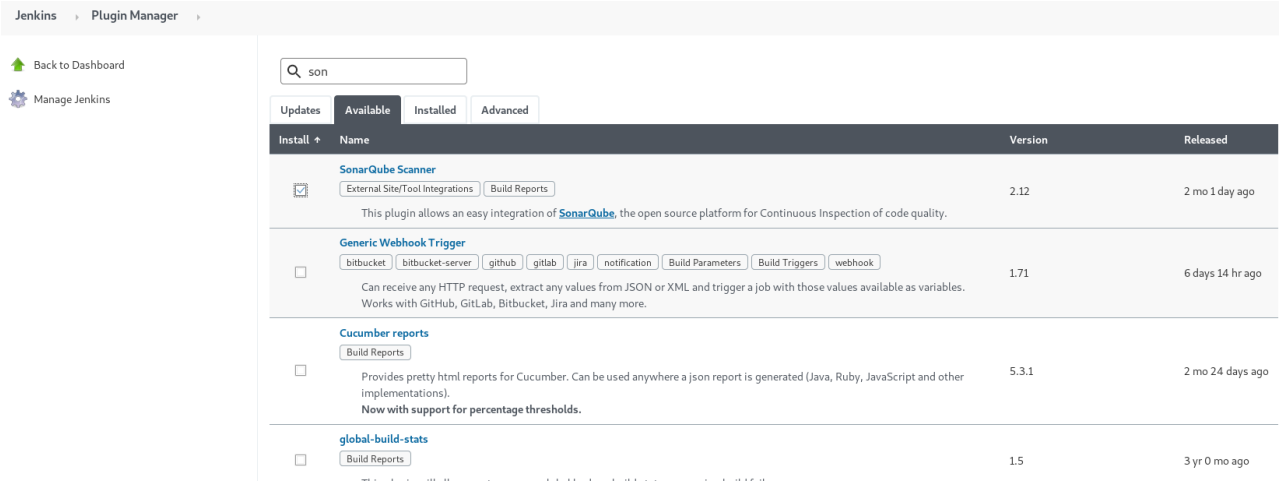
Upon completion of this session, you should be able to

- Use SonarQube to analysis source code
- Incorporating Jenkins Pipeline with SonarQube into your team project
- Analysis the findings generated by SonarQube

1: Installation

This lab is based on the instruction https://docs.sonarqube.org/latest/setup/get-started-2-minutes/ and the OWASP - Vulnerable Web Application https://github.com/OWASP/Vulnerable-Web-Application, but it also requires different docker SAST image / software to be installed before you can incorporate Jenkins Pipeline.

1. Install the **SonarQube Scanner** under the Plugin Manager



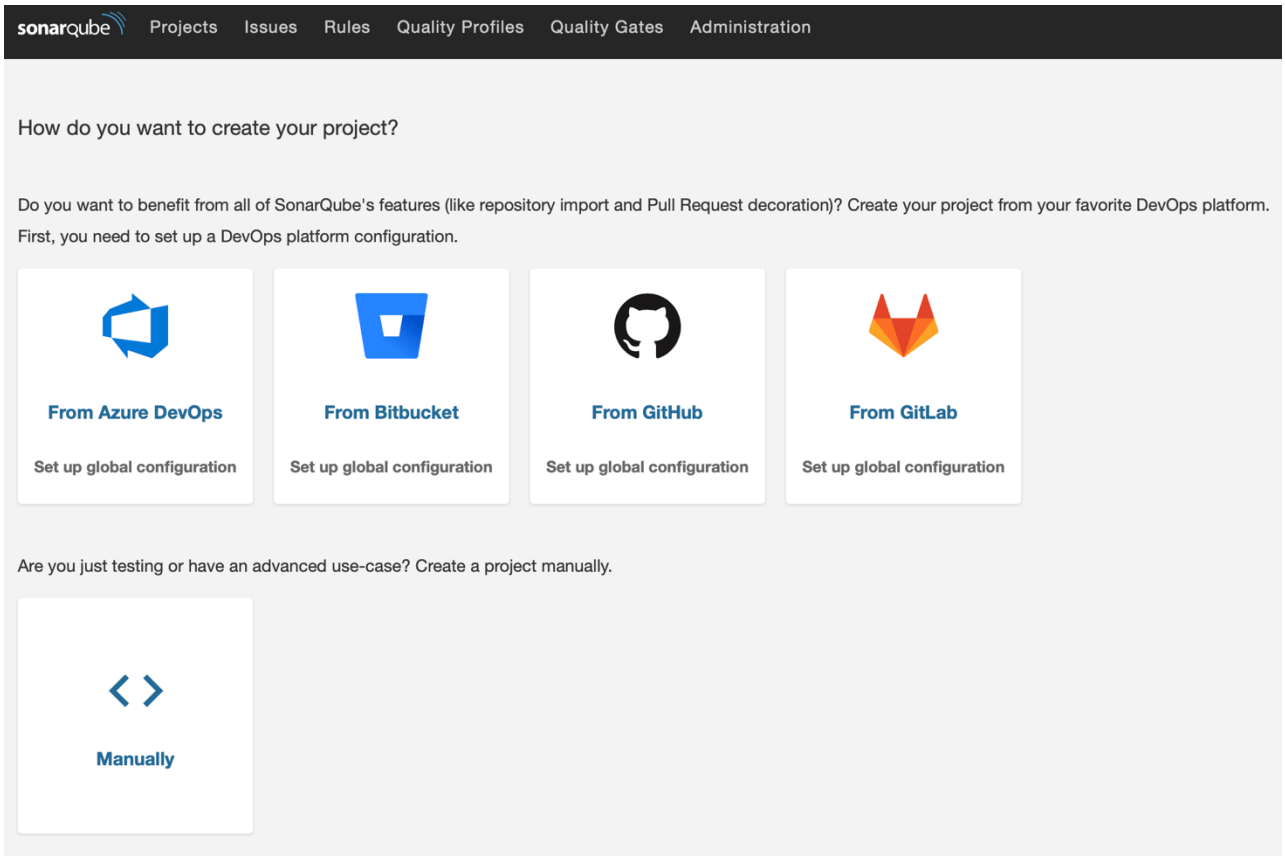
2. Restart Jenkins

3. Install the **SonarQube docker** by

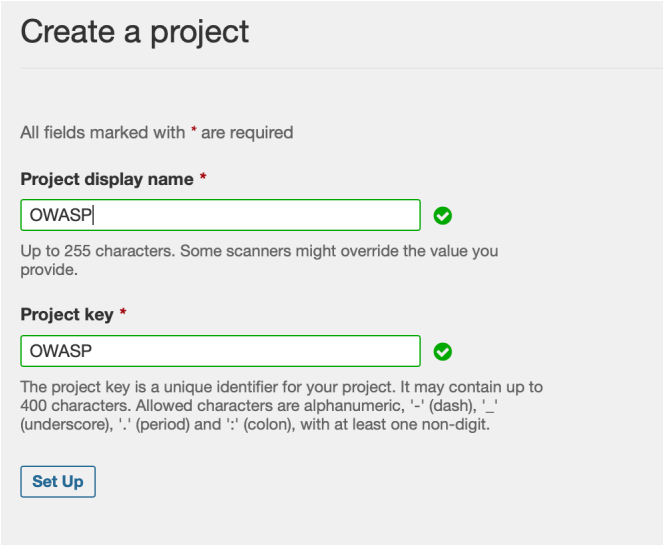
```
$ docker pull sonarqube
```

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

4. Log in to http://localhost:9000 with System Administrator credentials (login=admin, password=admin).

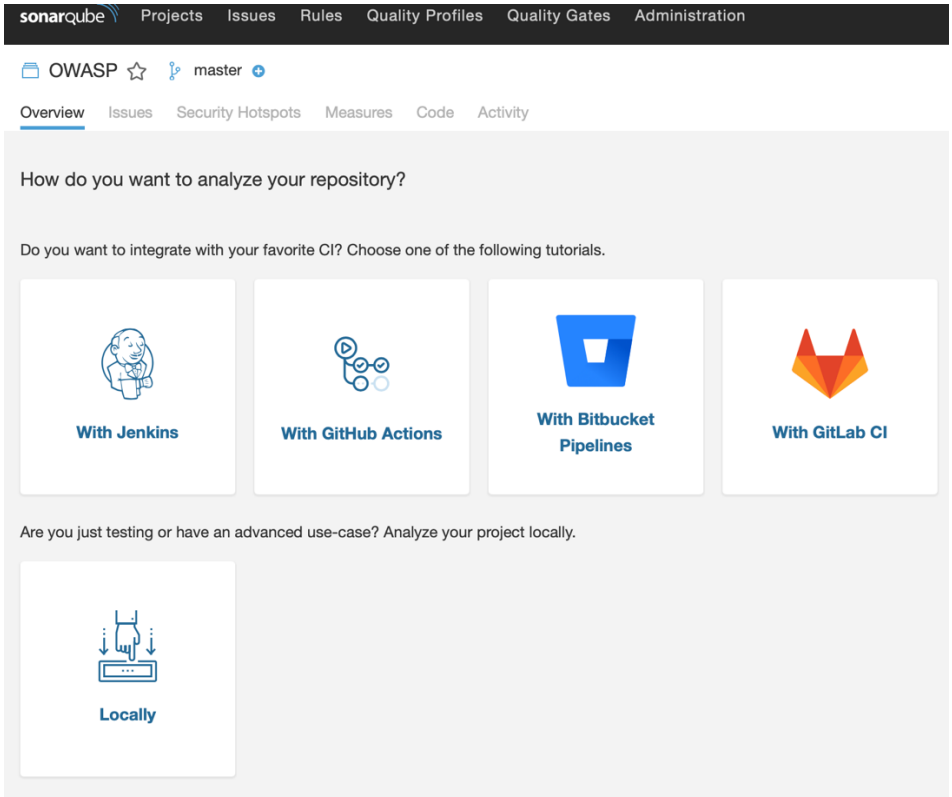


5. Press the “Manually” icon.

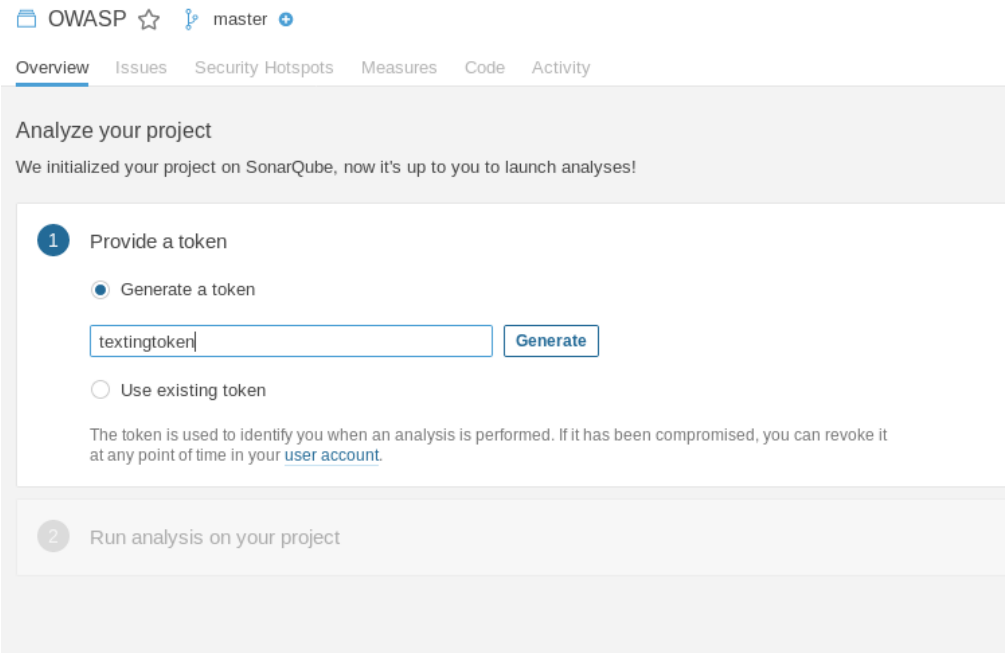


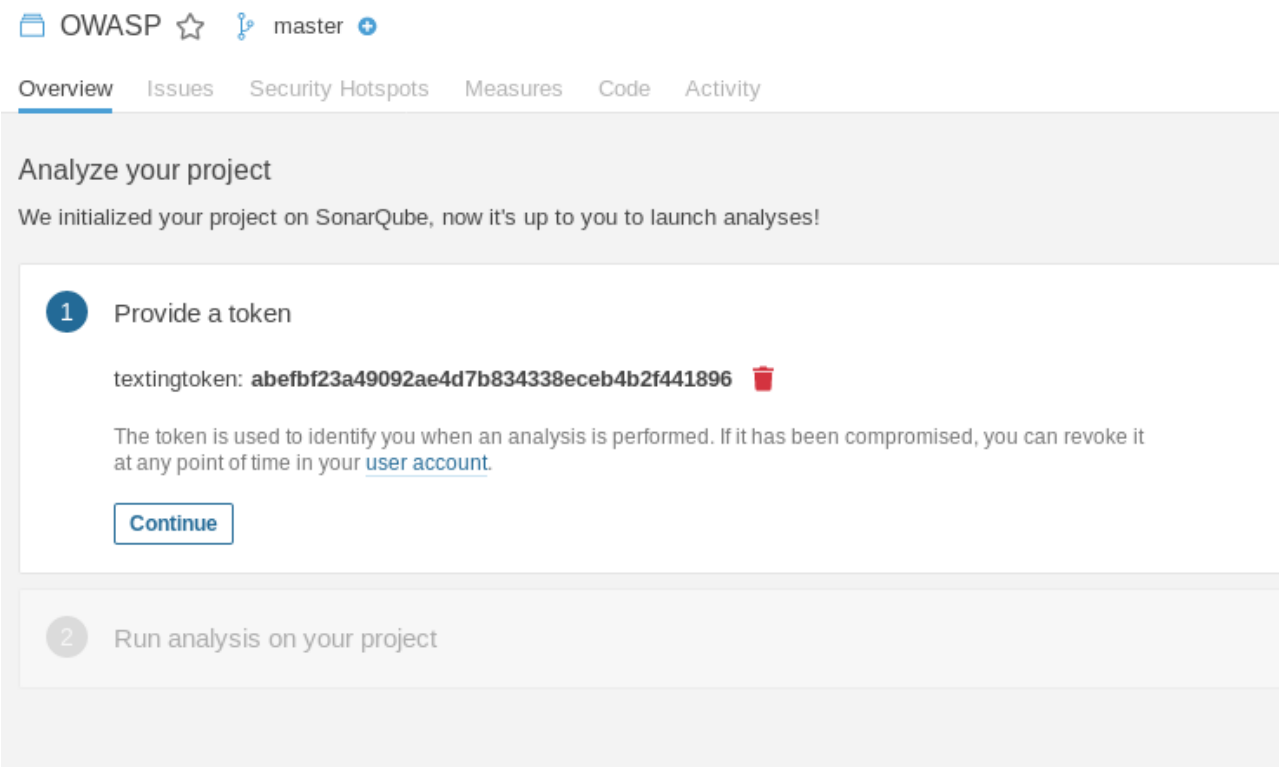
6. Input the project key and project name.

7. Press “Locally”.

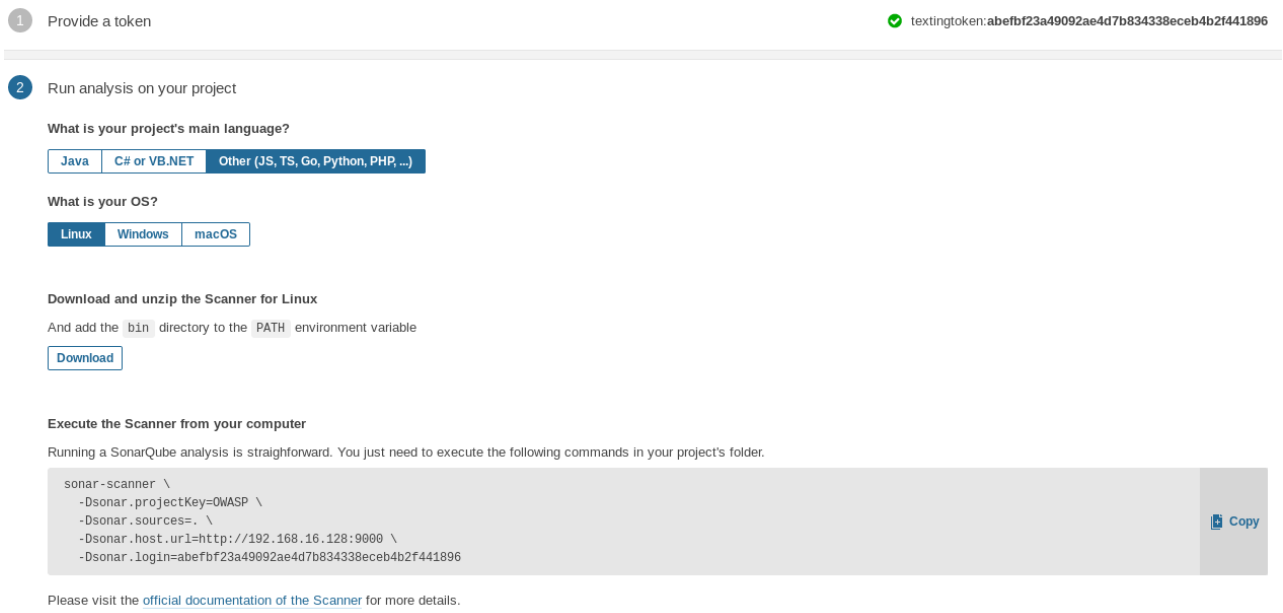


7. Generate a token for your project, copy the token and you will be using the token in the Jenkinsfile.





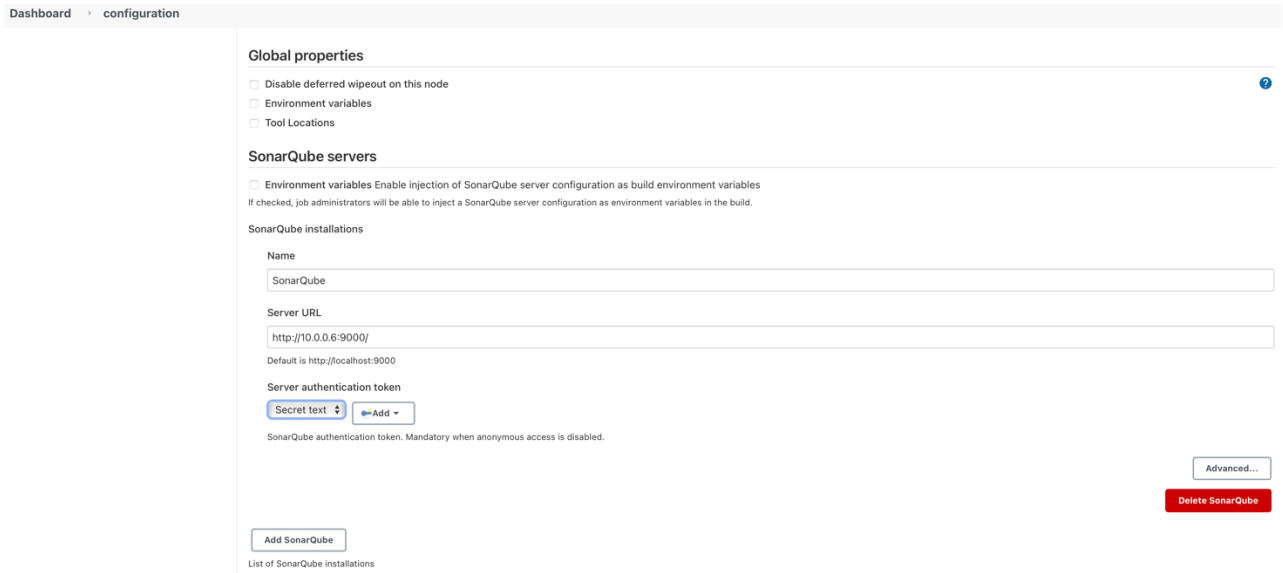
8. Select the language and OS you are using, copy the Scanner information and you will be using it in the Jenkinsfile.



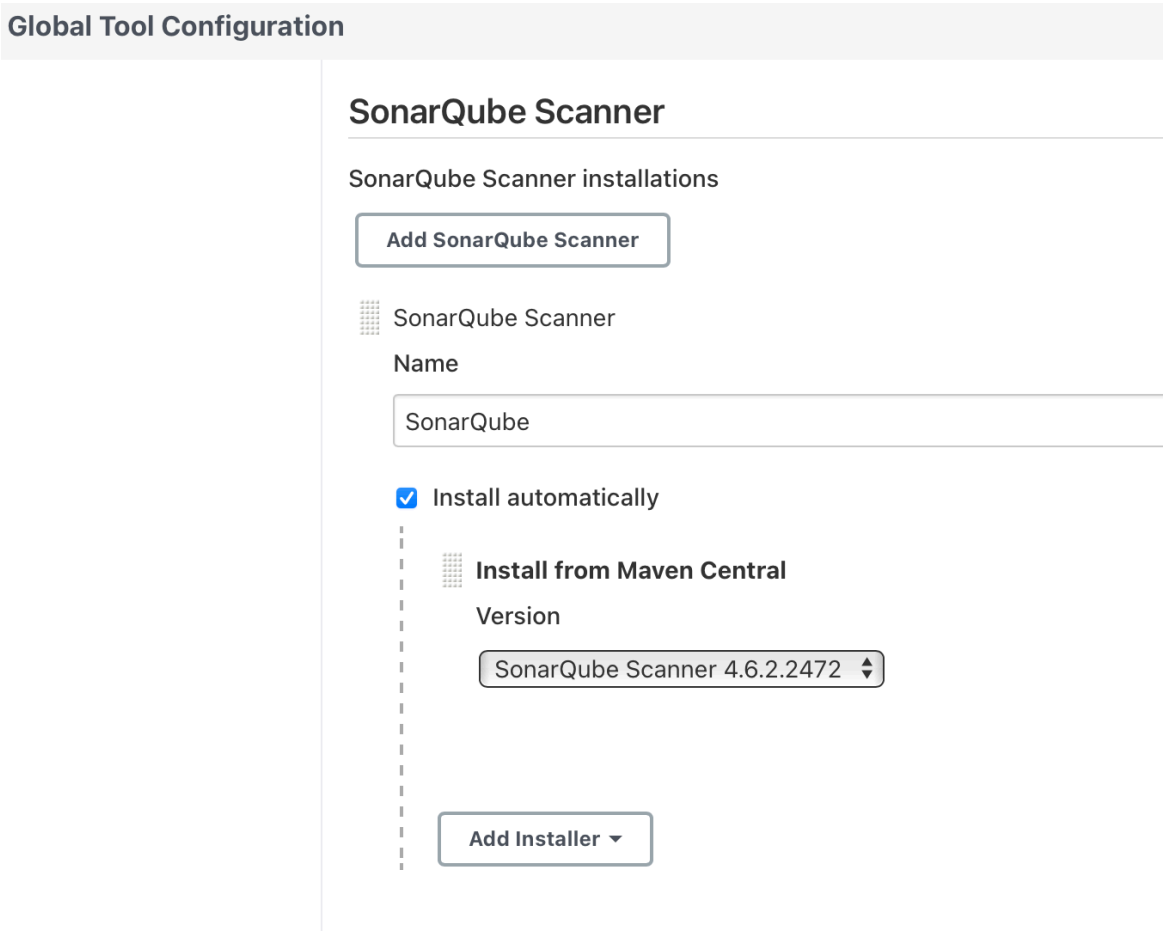
9. Back to Jenkin, go to Configure System to enable SonarQube Scanner.

To Configure your SonarQube server(s):

1. Log into Jenkins as an administrator and go to **Manage Jenkins > Configure System**.
2. Scroll down to the SonarQube configuration section, click **Add SonarQube**, and add the values you're prompted for.
3. The server authentication token should be created as a 'Secret Text' credential.



10. Back to Jenkin, go to Global Tool Configuration to define the SonarQube Scanner Tool.



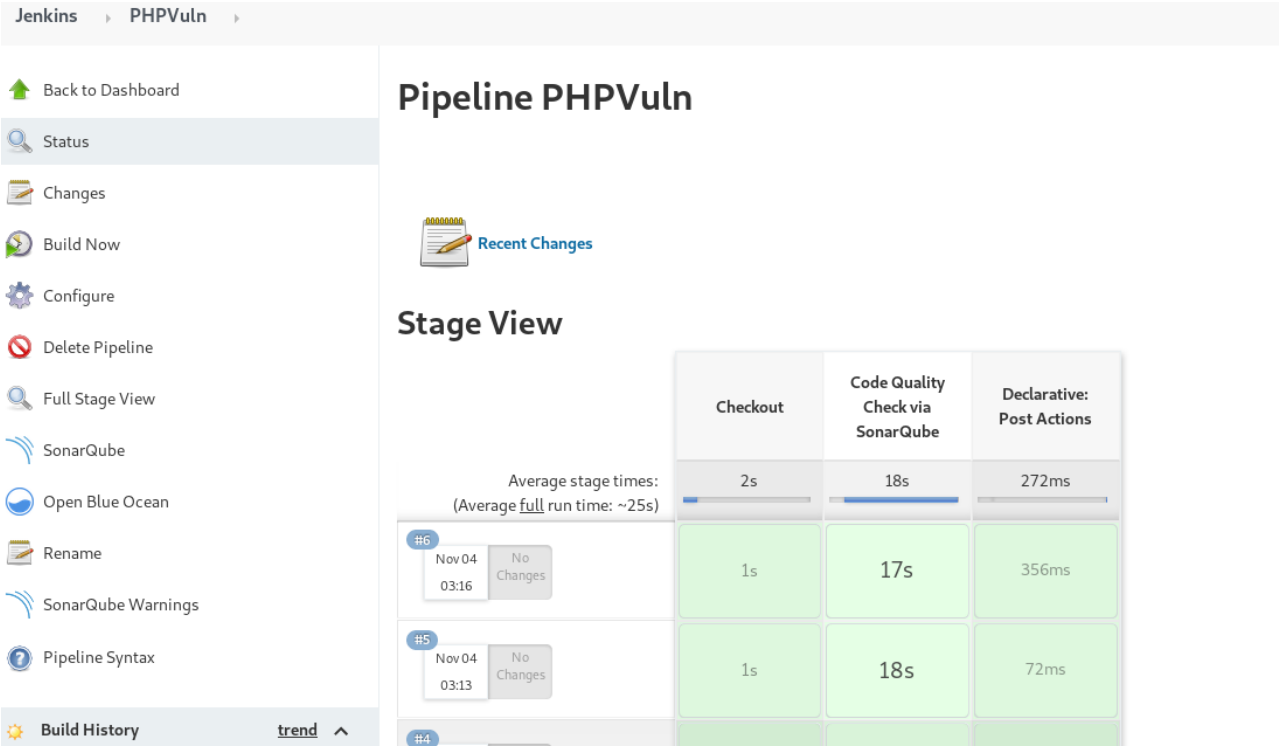
2: Configuration

Use the following Jenkinsfile in your pipeline:

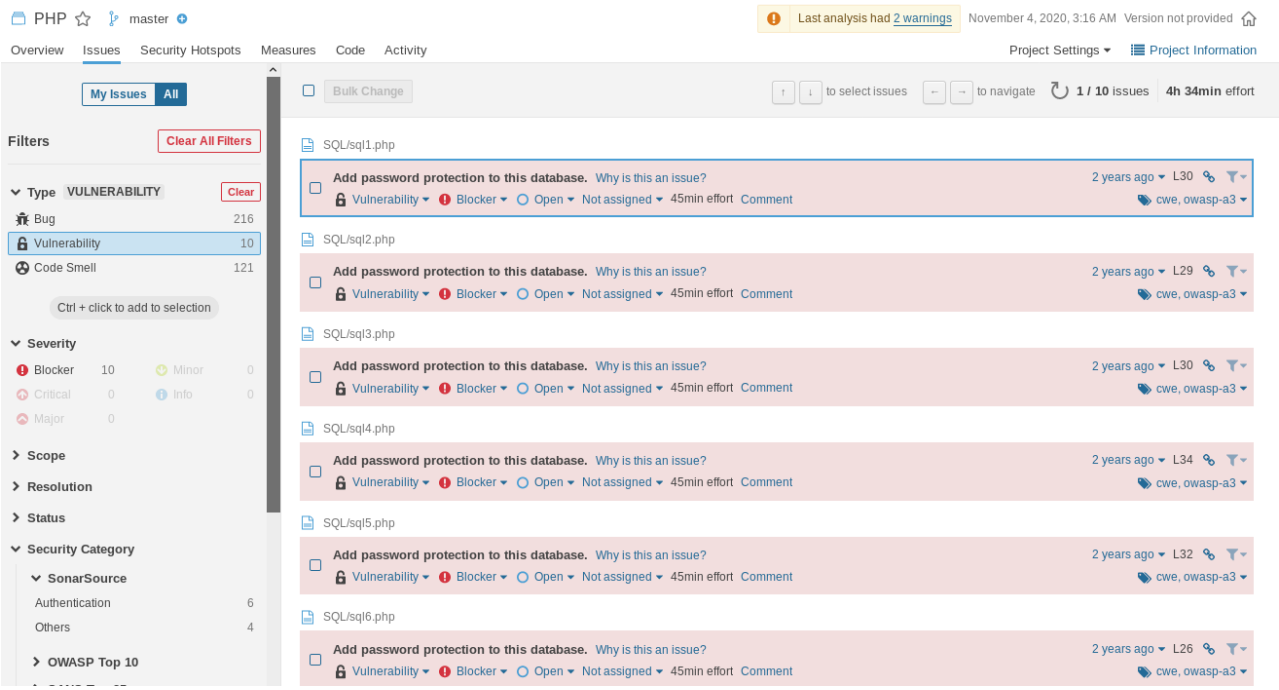
```
pipeline {
   agent any
   stages {
      stage ('Checkout') {
         steps {
            git branch:'master', url: 'https://github.com/OWASP/Vulnerable-Web-
Application.git'
         }
      }

      stage('Code Quality Check via SonarQube') {
        steps {
           script {
            def scannerHome = tool 'SonarQube';
              withSonarQubeEnv('SonarQube') {
              sh "${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=OWSAP -
Dsonar.sources=."
              }
           }
        }
      }
   }
   post {
      always {
         recordIssues enabledForFailure: true, tool: sonarQube()
      }
   }
}
```

Note that:
"SonarQube" is the name you used in System configuration.
"OWASP" is the SonarQube project name



Press “Build Now” and then “SonarQube”



You may start analyzing the result!

3: Standalone SonarQube Scanner

Alternatively, you can run the standalone SonarQube Scanner in your project folder to scan the source code. Especially when you are the 3rd party code auditor.

```
docker run --rm -e SONAR_HOST_URL=http://192.168.16.128:9000 -e
SONAR_LOGIN=2742fdcd3a1fe63a0912d32ebd77a1c74a4e212d -it -v "$(pwd):/usr/src"
sonarsource/sonar-scanner-cli -Dsonar.projectKey=OWASP
```

4: Docker memory issue

You may face the out-of-memory issue on the Jenkins / SonarQube docker. You may use the following command to increase the memory from 2GB to 3GB:

```
docker run --memory=3g the-remaining-command
```

5: Reference

https://docs.sonarqube.org/latest/

https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-jenkins/

END OF DOCUMENT