EXPERIMENT NO . 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 :-

SonarQube is a universal tool for static code analysis that has become more or less the industry standard. Keeping code clean, simple, and easy to read is also a lot easier with SonarQube. SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.

Benefits of SonarQube

1. Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.
2. Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
3. Quality code - Code quality control is an inseparable part of the process of software development.
4. Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
5. Increase consistency - Determines where the code criteria are breached and enhances the quality

Business scaling - No restriction on the number of projects to be evaluated.

6. Enhance developer skills - Regular feedback on quality problems helps developers to improve their coding skills.

SonarQube Setup

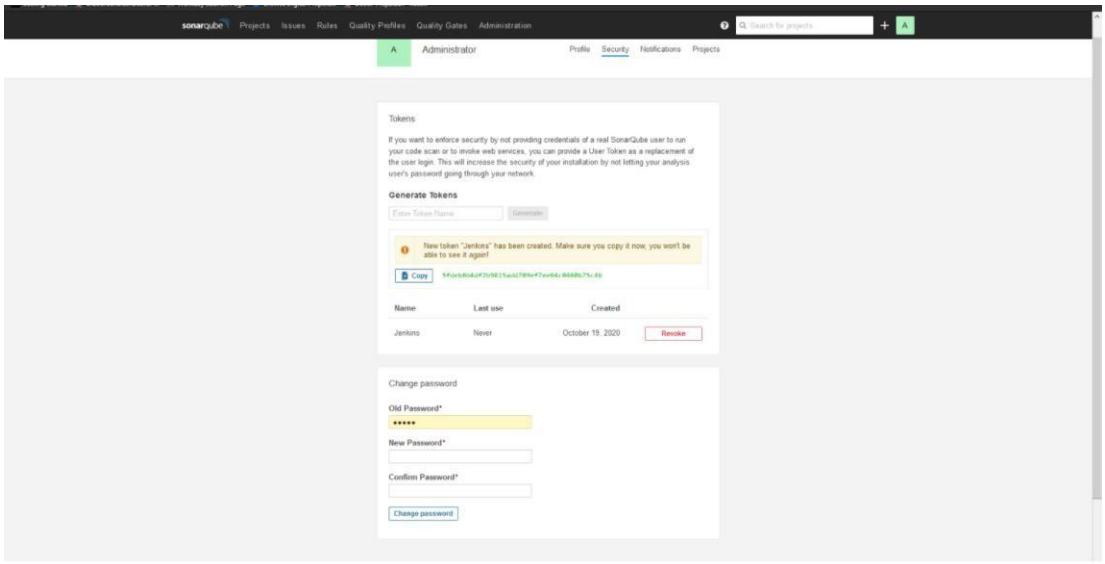
SonarQube Instance

Before proceeding with the integration, we will setup SonarQube Instance. Choice of the platform is yours.

```
$ docker run -d -p 9000:9000 sonarqube
```

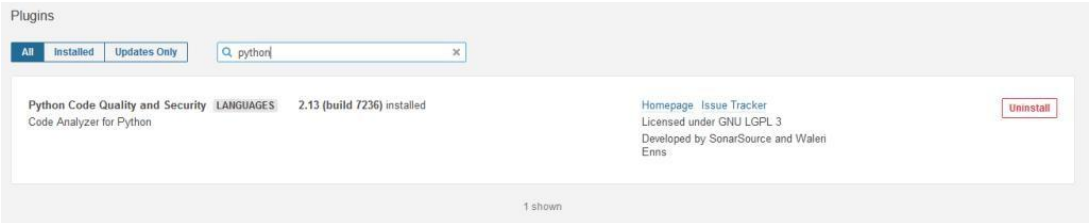
Generate User Token

Now, we need to get the SonarQube user token to make connection between Jenkins and SonarQube. For the same, go to User > My Account > Security and then, from the bottom of the page you can create new tokens by clicking the Generate Button. Copy the Token and keep it safe.

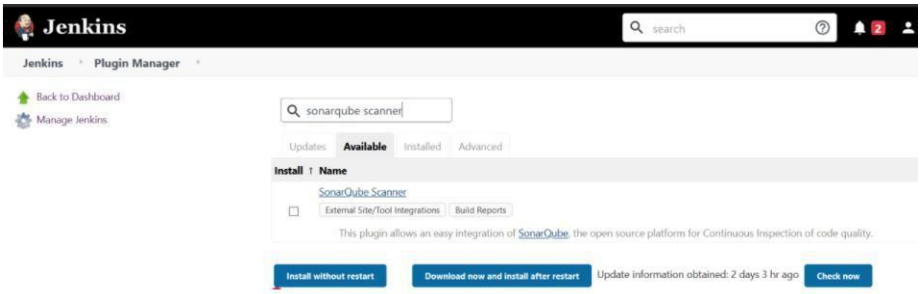


Add necessary plugin

In this Tutorial, we are following a Python-based application. So, we need to add a python plugin in the SonarQube so that it will collect the Bugs and Static code analysis from Jenkins. For the same, go to Administration > Marketplace > Plugins. Then in the search box, search for Python. Then, you will see Python Code Quality and Security (Code Analyzer for Python). Just install. That's all from the SonarQube side.

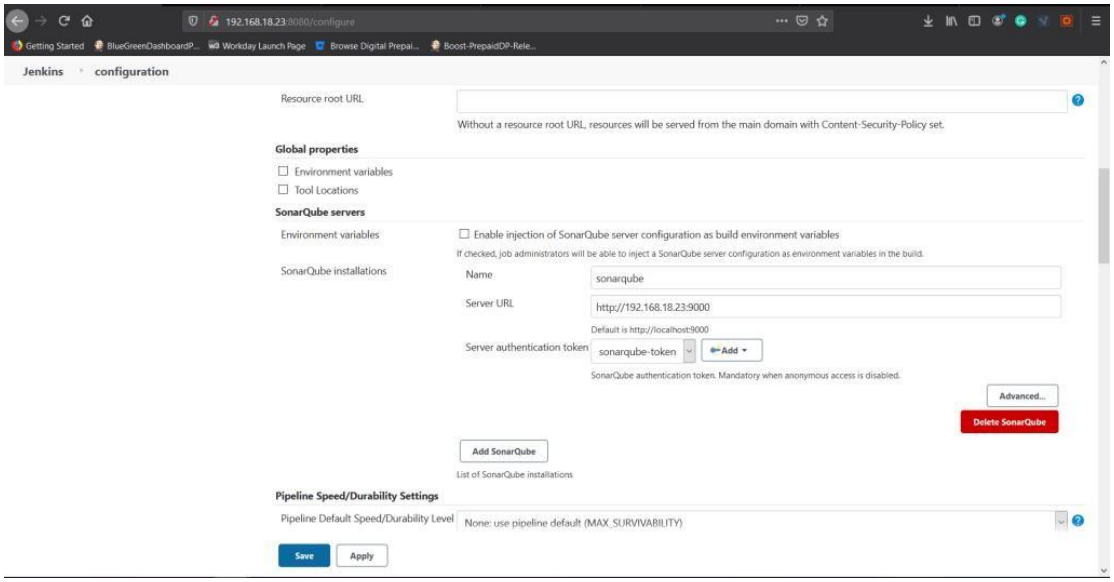


Jenkins Setup for SonarQube
SonarQube plugin in Jenkins.
Before all, we need to install the SonarQube Scanner plugin in Jenkins. For the same, go to Manage Jenkins > Plugin Manager > Available. From here, type SonarQube Scanner then select and install.

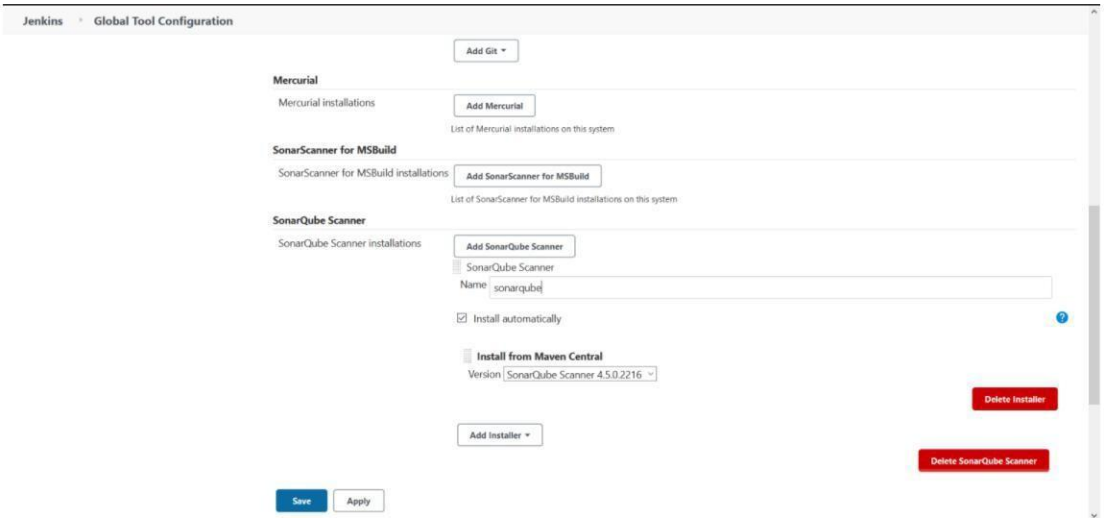


SonarQube Scanner Plugin for Jenkins

Tool Configuration SonarQube Scanner
Now, we need to configure the Jenkins plugin for SonarQube Scanner to make a connection with the SonarQube Instance. For that, got to Manage Jenkins > Configure System > SonarQube Server. Then, Add SonarQube. In this, give the Installation Name, Server URL then Add the Authentication token in the Jenkins Credential Manager and select the same in the configuration.



Then, we need to set-up the SonarQube Scanner to scan the source code in the various stage. For the same, go to Manage Jenkins > Global Tool Configuration > SonarQube Scanner. Then, Click Add SonarQube Scanner Button. From there, give some name of the scanner type and Add Installer of your choice. In this case, I have selected SonarQube Scanner from Maven Central.



SonarQube Scanner in Jenkins Pipeline

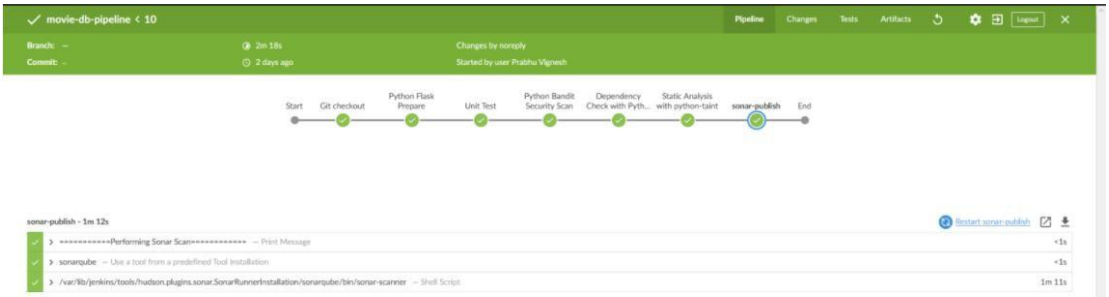
Now, It's time to integrate the SonarQube Scanner in the Jenkins Pipeline. For the same, we are going to add one more stage in the Jenkinsfile called sonar-publish.

```
stage ("sonar-
        publish"){ steps {
                echo "===========Performing Sonar
Scan============"
                sh "${tool("sonarqube")}/bin/sonar-scanner"
        }
}
```

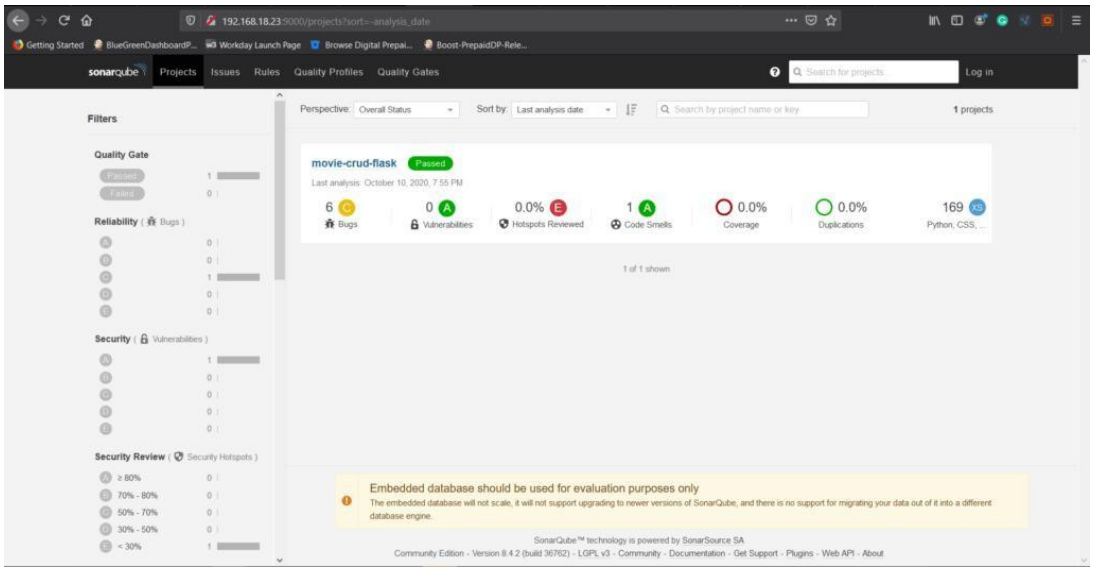
Were this will collect the SonarQube Server information from the sonar-project.properties file and publish the collected information to the SonarQube Server. So, the overall code will look like the below snippet.

```
pipeline {
        agent any
        stages {
                stage ("Git
                        checkout"){ step
                        s {
                                git branch: "master",
                                        url:
"https://github.com/PrabhuVignesh/movie-crud-flask.git"
                                sh "ls"
                        }
                }
                stage ("Python Flask
                        Prepare"){ steps {
                                sh "pip3 install -r requirements.txt"
                        }

                }
                stage ("Unit
                        Test"){ steps{
                                sh "python3 test_basic.py"
                        }
                }
                stage ("Python Bandit Security Scan"){
```

```
                steps{
                        sh "cat report/banditResult.json"
                        sh "sh run_bandit.sh || true"
                        sh "ls"
                }
        }
        stage ("Dependency Check with Python
                Safety"){ steps{
                        sh "docker run --rm --volume \$(pwd)
pyupio/safety:latest safety check"
                        sh "docker run --rm --volume \$(pwd)
pyupio/safety:latest safety check --json > report.json"
                }
        }
        stage ("Static Analysis with python-
                taint"){ steps{
                        sh "docker run --rm --volume \$(pwd)
vickyrajagopal/python-taint-docker pyt ."
                }
        }
        stage ("sonar-
                publish"){ steps {
                        echo "===========Performing Sonar
Scan============"
                        sh "${tool("sonarqube")}/bin/sonar-
scanner"
                }
        }

    }
}
```



Where it will just execute the SonarQube Scanner and collect the SAST information and Python bandit report in the format of JSON. Then, it will publish the same in the SonarQube Server. If you login to the SonarQube and visit the Dashboard, you will see the Analysis of the project there.



CONCLUSION :- Hence , Creating 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 is successfully implemented.