EXPERIEMENT NO : 07

AIM : To understand Static Analysis SAST process and learn tointegrate Jenkins SAST to SonarQube/GitLab.

THEORY :

SonarQube is a Code Quality Assurance tool that collects and analyzes source code, and provides reports for the code quality of your project. It combines static and dynamic analysis tools and enables quality to be measured continually over time. Everything from minor styling choices, to design errors are inspected and evaluated by SonarQube. This provides users with a rich searchable history of thecode to analyze where the code is messing up and determine whether or not it is styling issues, code defeats, code duplication, lack of test coverage, or excessivelycomplex code. The software will analyze source code from different aspects and drills down the code layer by layer, moving module level down to the class level, with each level producing metric values and statistics that should reveal problematic areas in the source code that needs improvement.

Sonarqube also ensures code reliability, Application security, and reduces technical debt by making your code base clean and maintainable. Sonarqube also provides support for 27 different languages, including C, C++, Java, Javascript, PHP, GO, Python, and much more.SonarQube also provides Ci/CD integration, and gives feedback during code review with branch analysis and pull request decoration.

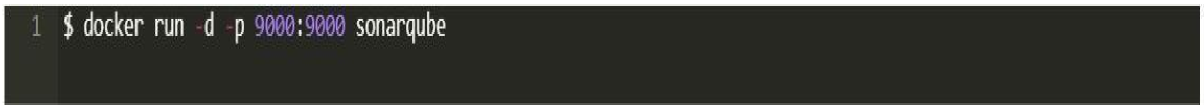
SonarQube Benefits:

- CI tools do not have a plugin which would make all of these tools work easily together
- CI tools do not have plugins to provide nice drill-down features that SonarQube has
- CI Plugins does not talk about overall compliance value
- CI plugins do not provide managerial perspective
- There is no CI plugin for Design or Architectural issues
- CI plugins do not provide a dashboard for overall

project quality

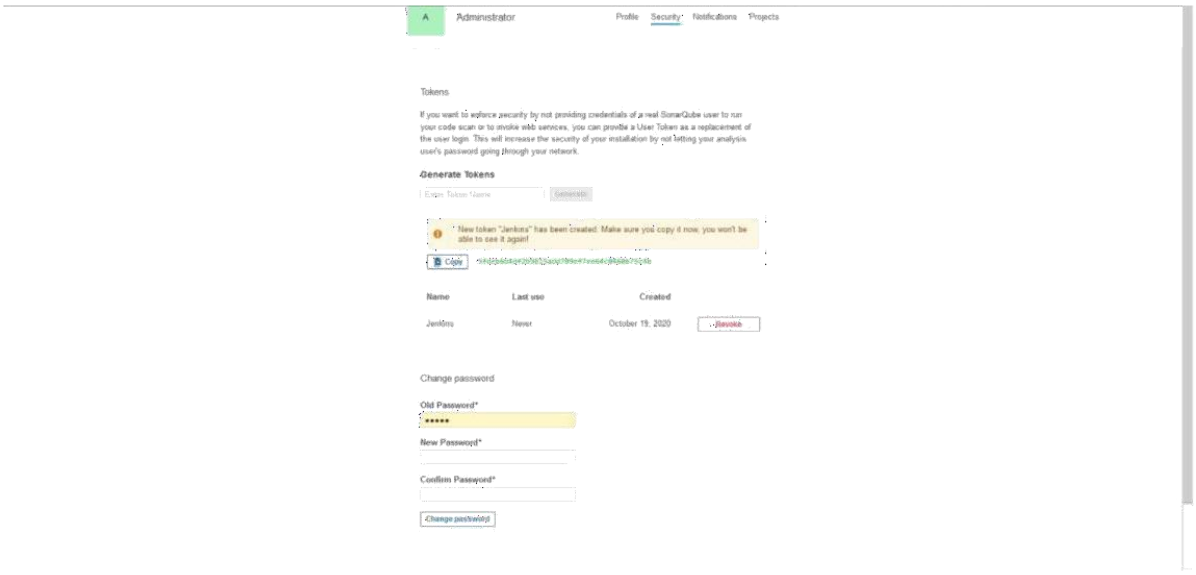
SonarQube Setup

Step 1: SonarQube Instance



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

Step 2: Generate user token



Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

[illegible]

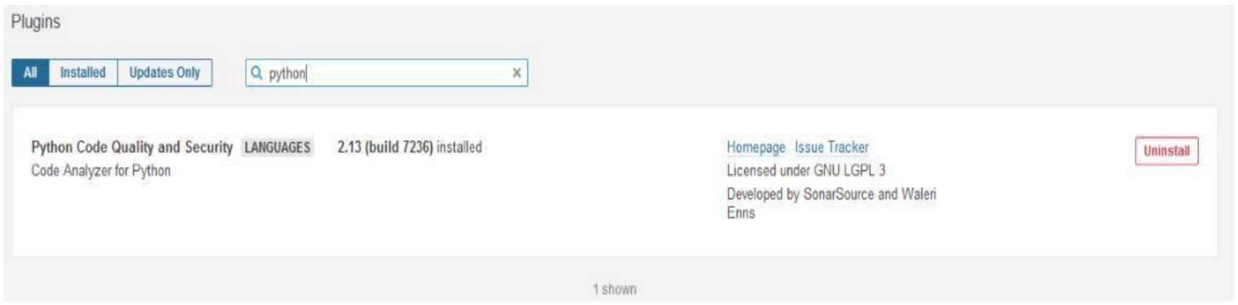
Name	Last use	Created	
Jenkins	Never	October 19, 2020	Revoke

Change password

Old Password*: *****

New Password*:

Confirm Password*:



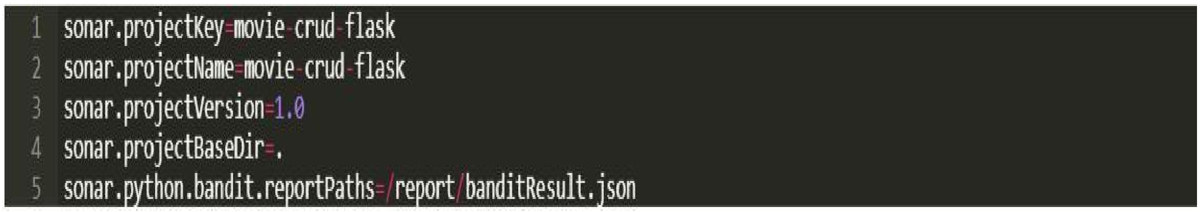
Plugins

Python Code Quality and Security LANGUAGES 2.13 (build 7236) installed
Code Analyzer for Python

Homepage Issue Tracker
Licensed under GNU LGPL 3
Developed by SonarSource and Waleri Enns

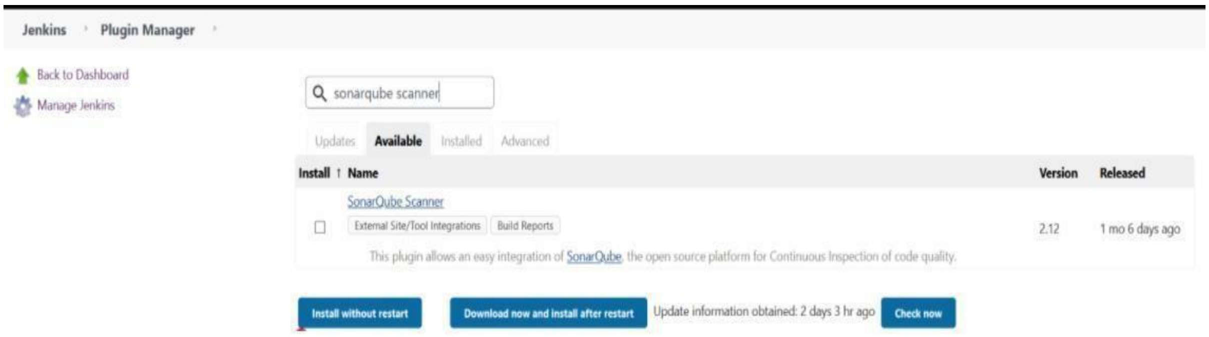
1 shown

Step 3: Add necessary plugin



```
sonar.projectKey=movie-crud-flask
sonar.projectName=movie-crud-flask
sonar.projectVersion=1.0
sonar.projectBaseDir=.
sonar.python.bandit.reportPaths=/report/banditResult.json
```

Step 4: Configuring SonarQube in Codebase



Jenkins › Plugin Manager ›

Back to Dashboard
Manage Jenkins

sonarqube scanner

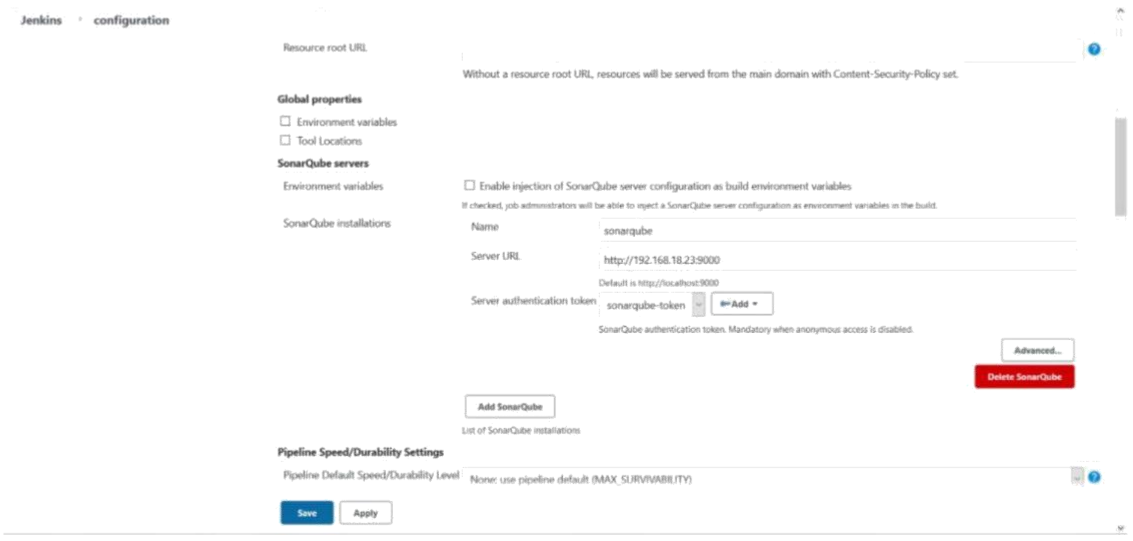
Updates | **Available** | Installed | Advanced

Install ↑	Name	Version	Released
☐	SonarQube Scanner External Site/Tool Integrations · Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	2.12	1 mo 6 days ago

Install without restart | Download now and install after restart | Update information obtained: 2 days 3 hr ago | Check now

Step 5: Jenkins Setup for SonarQube

Step 6: Tool configuration with SonarQube Scanner



Jenkins › configuration

Resource root URL

Without a resource root URL, resources will be served from the main domain with Content-Security-Policy set.

Global properties

☐ Environment variables
☐ Tool Locations

SonarQube servers

Environment variables ☐ Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name	sonarqube
Server URL	http://192.168.18.23:9000
	Default is http://localhost:9000
Server authentication token	sonarqube-token
	SonarQube authentication token. Mandatory when anonymous access is disabled.

Advanced...
Delete SonarQube

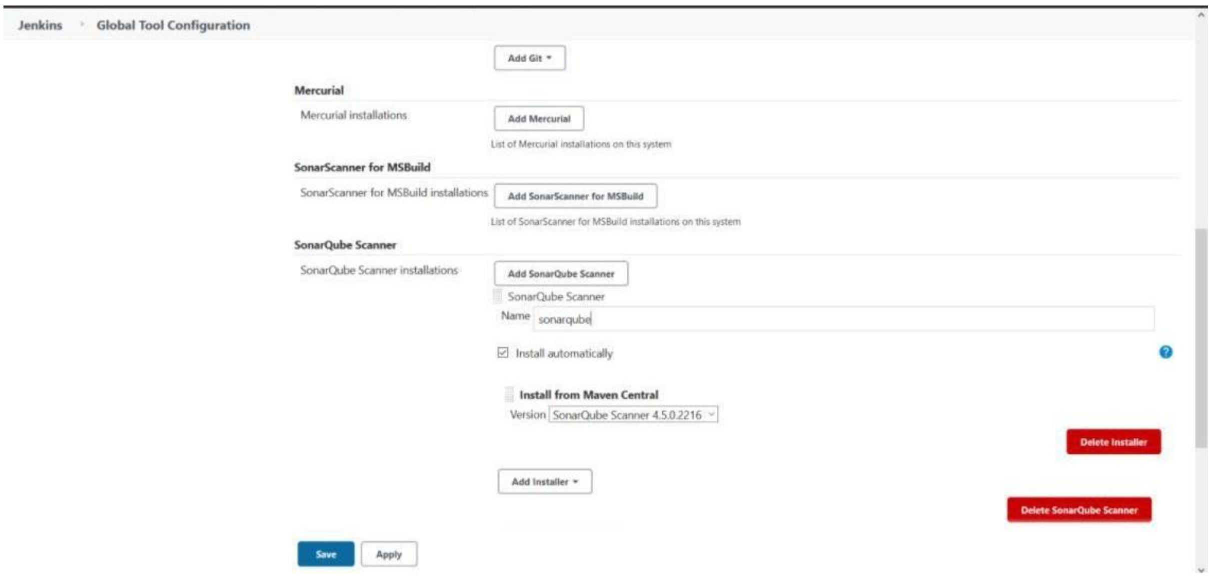
Add SonarQube

List of SonarQube installations

Pipeline Speed/Durability Settings

Pipeline Default Speed/Durability Level: None: use pipeline default (MAX_SURVIVABILITY)

Save | Apply


Jenkins › Global Tool Configuration

Add Git

Mercurial

Mercurial installations — Add Mercurial

List of Mercurial installations on this system

SonarScanner for MSBuild

SonarScanner for MSBuild installations — Add SonarScanner for MSBuild

List of SonarScanner for MSBuild installations on this system

SonarQube Scanner

SonarQube Scanner installations — Add SonarQube Scanner

SonarQube Scanner

Name: sonarqube

☑ Install automatically

Install from Maven Central

Version: SonarQube Scanner 4.5.0.2216

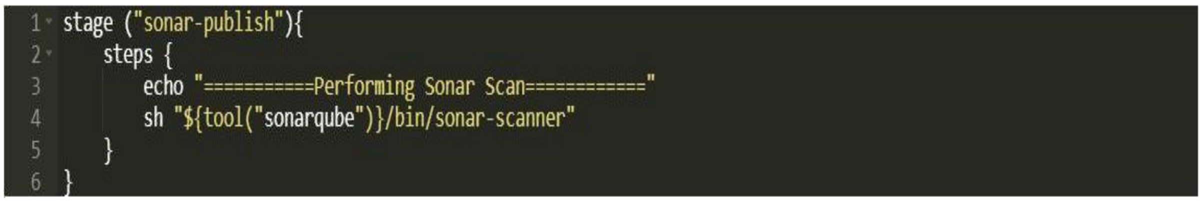
Delete Installer

Add Installer

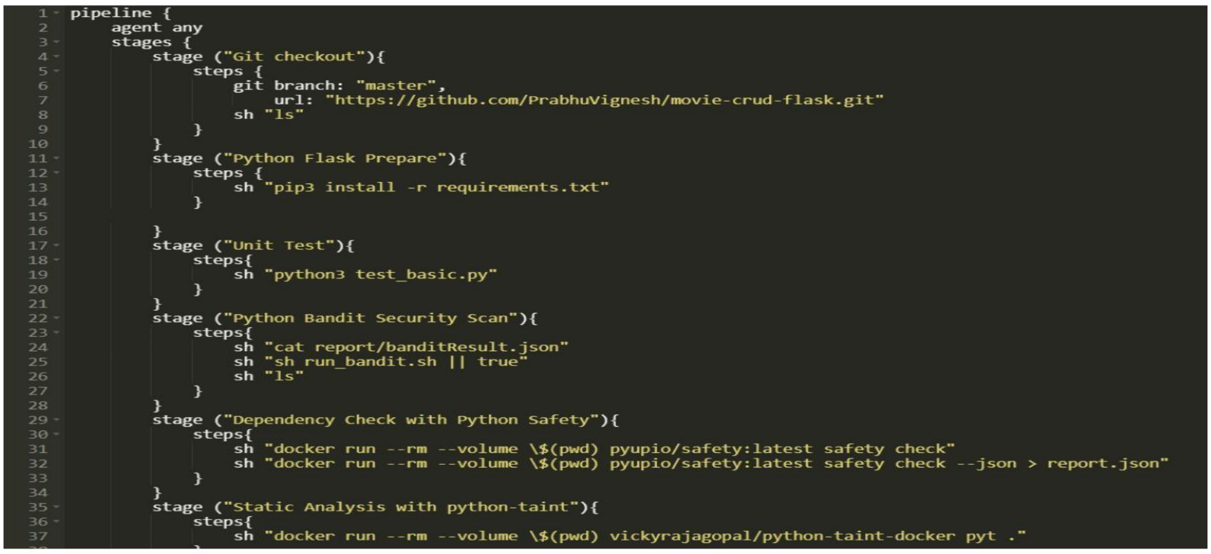
Delete SonarQube Scanner

Save Apply

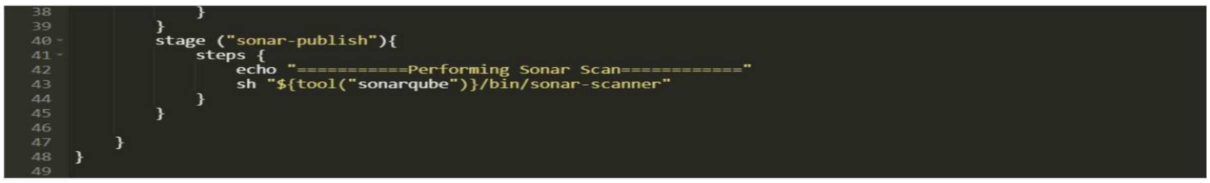
Step 7: SonarQube Scanner in Jenkins Pipeline



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

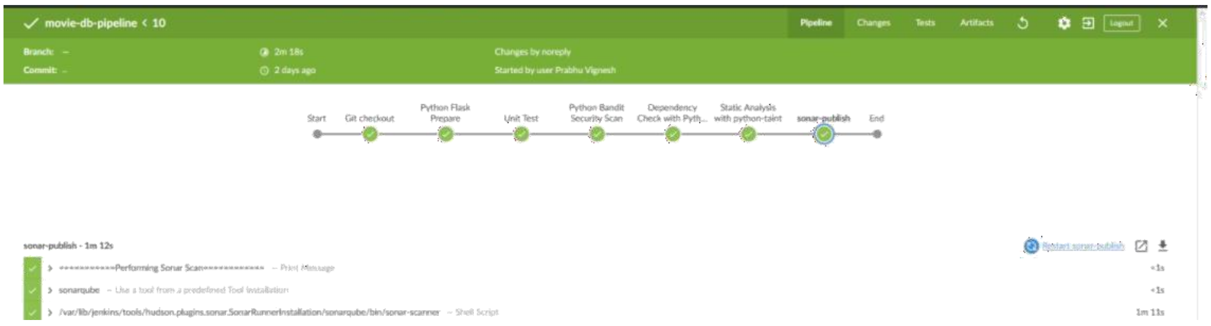


```
pipeline {
    agent any
    stages {
        stage ("Git checkout"){
            steps {
                git branch: "master",
                    url: "https://github.com/PrabhuVignesh/movie-crud-flask.git"
                sh "ls"
            }
        }
        stage ("Python Flask Prepare"){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

    }
}
```

Step 8: Check the output



```
sonar-publish - 1m 12s
> ===========Performing Sonar Scan============   – Print Message                                   <1s
> sonarqube   – Use a tool from a predefined Tool Installation                                      <1s
> /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarqube/bin/sonar-scanner   – Shell Script   1m 11s
```

CONCLUSION : Hence, we Successfully understood Static Analysis SAST process and learnt to integrate Jenkins SAST to SonarQube/GitLab.