EXPERIMENT - 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:

Security in CI/CD: Integrating security into CI/CD pipelines is crucial in today's software development landscape. This approach, often called "shifting left," involves incorporating security practices earlier in the development process. Static analysis plays a key role by automatically identifying potential security vulnerabilities during the build phase. This early detection allows developers to address security issues before they reach production, significantly reducing the risk and cost associated with security breaches.

Code Quality Metrics: Static analysis tools evaluate code based on various metrics that indicate code quality, maintainability, and potential issues. Common metrics include:

- Code Duplication: Identifies repeated code segments. Excessive duplication can lead to maintenance difficulties and inconsistencies.
- Comment Density: Measures the ratio of comments to code. While not definitive, it can indicate code clarity.
- Code Coverage: Although typically a dynamic analysis metric, it's often reported alongside static analysis results to show the extent of code exercised by tests.

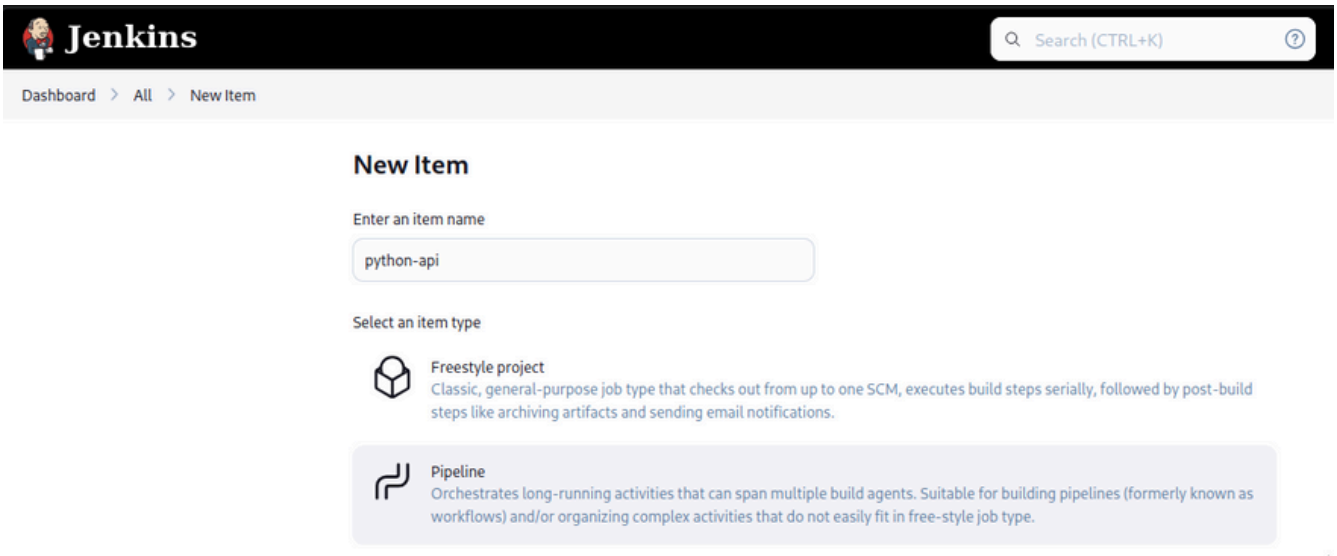
Benefits of Automated Static Analysis in CI/CD:

a) Early Detection: Issues are identified as soon as code is committed, allowing for immediate correction.
b) Consistency: Applies the same quality and security standards across the entire codebase and development team.
c) Continuous Feedback: Developers receive ongoing insights about their code quality, fostering a culture of continuous improvement.
d) Reduced Review Effort: Automates part of the code review process, allowing human reviewers to focus on higher-level concerns.

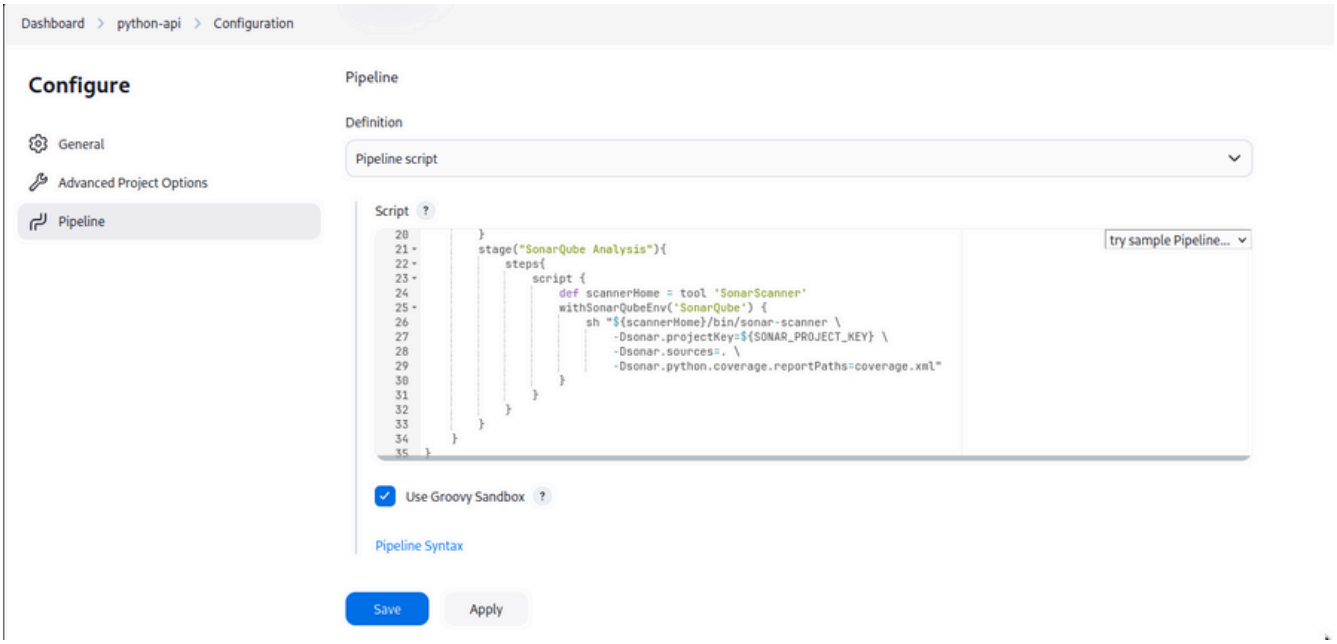
STEPS:

1. Add SonarQube to your jenkins, and setup it.

2. Create a new item in jenkins with ”pipeline” type”



3. Then, Create Pipeline Script in groovy includes steps to build project and run sonar-scanner on it to check for security issues.



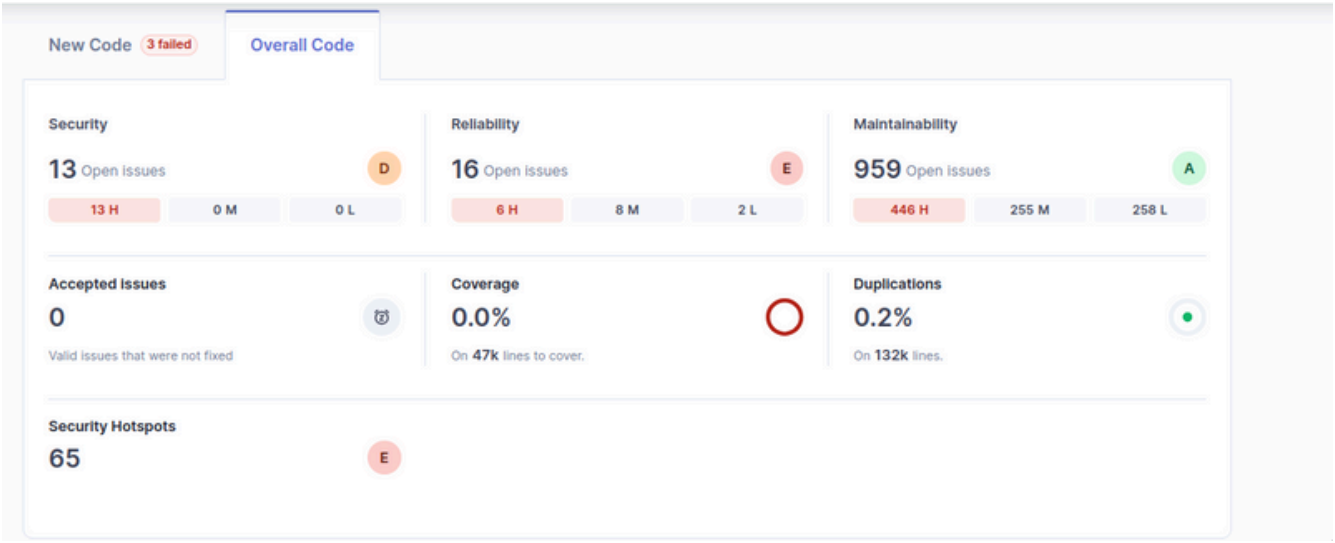
```
pipeline {
    agent any
    environment {
        SONAR_PROJECT_KEY = 'new'
    }
    stages {
        stage("Code"){
            steps{
                git url: "https://github.com/Dark-Kernel/minipy.git", branch: "master"
            }
        }
        stage("Install dependencies") {
            steps{
                sh "python -m venv env"
                sh "source env/bin/activate"
                sh "pip install -r requirements.txt"
            }
        }
        stage("SonarQube Analysis") {
            steps{
                script {
                    def scannerHome = tool 'SonarScanner'
                    withSonarQubeEnv('SonarQube') {
                        sh "${scannerHome}/bin/sonar-scanner \
                            -Dsonar.projectKey=${SONAR_PROJECT_KEY} \
                            -Dsonar.sources=. \
                            -Dsonar.python.coverage.reportPaths=coverage.xml"
                    }
                }
            }
        }
    }
}
```

4. Then save, and build.

```
17:42:00.898 INFO  Analysis report uploaded in 196ms
17:42:00.899 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=new
17:42:00.901 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted
analysis report
17:42:00.901 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=be529038-9465-42b4-
a841-880063f99958
17:42:01.005 INFO  Analysis total time: 49.249 s
17:42:01.006 INFO  SonarScanner Engine completed successfully
17:42:01.202 INFO  EXECUTION SUCCESS
17:42:01.203 INFO  Total time: 57.455s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

5. Then On Successful build you will have your report of sonar-scanner containing Failed checks and vulnerabilities.

Visit SonarQube Dashboard and fix your application security issues.



Conclusion: Thus, we have successfully Created 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.