

### EXPERIMENT - 8

**Aim:** Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to performa static Analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

**Theory:**

## **CI/CD (Continuous Integration and Continuous Deployment)** is a software engineering approach that allows teams to deliver code changes more frequently and reliably. In CI, code changes are automatically tested and integrated into the main codebase. CD automates the release of validated code to a production or staging environment.

* Jenkins, an open-source automation server, is a popular tool used to implement CI/CD pipelines.

# What is Jenkins?

## Jenkins automates various stages of software development, including building, testing, and deploying applications. It integrates with version control systems like GitLab, GitHub, and Bitbucket.

1. **Jenkins-SonarQube-GitLab Pipeline Workflow** The pipeline integrates Jenkins with SonarQube for code quality analysis and with GitLab for version control and collaboration. The typical workflow involves:

# Developers push code to GitLab.

## **Jenkins triggers a build** whenever code is pushed or a pull request is created. This can be done by setting up a webhook in GitLab.

* 1. **Jenkins runs the build** process, which includes pulling the latest code from GitLab, compiling it, and running unit tests.
  2. **SonarQube performs static code analysis** to check for code smells, bugs, and security vulnerabilities.

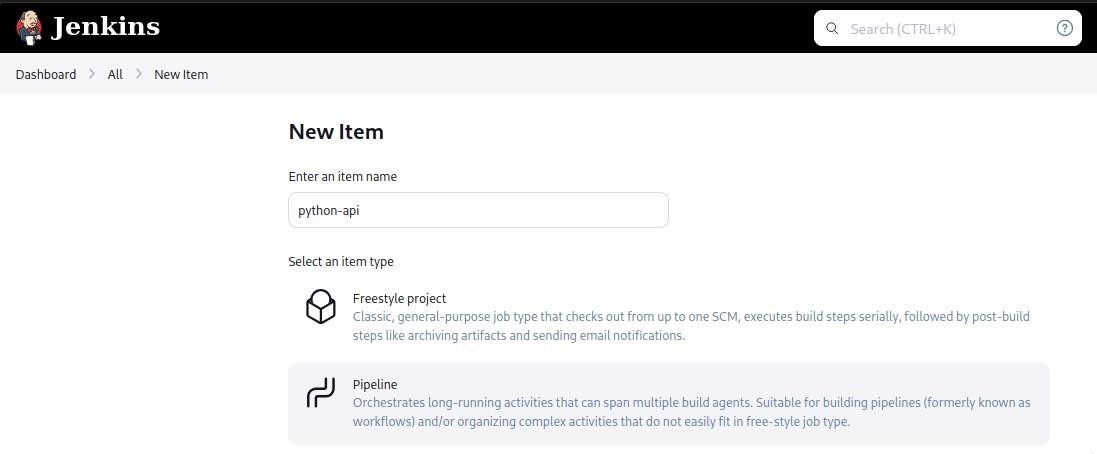
## **SonarQube generates a report** and sends the result back to Jenkins, which will display it in the Jenkins dashboard. The report includes metrics such as the number of bugs, maintainability issues, and security vulnerabilities.

### Benefits of Automated Static Analysis in CI/CD:

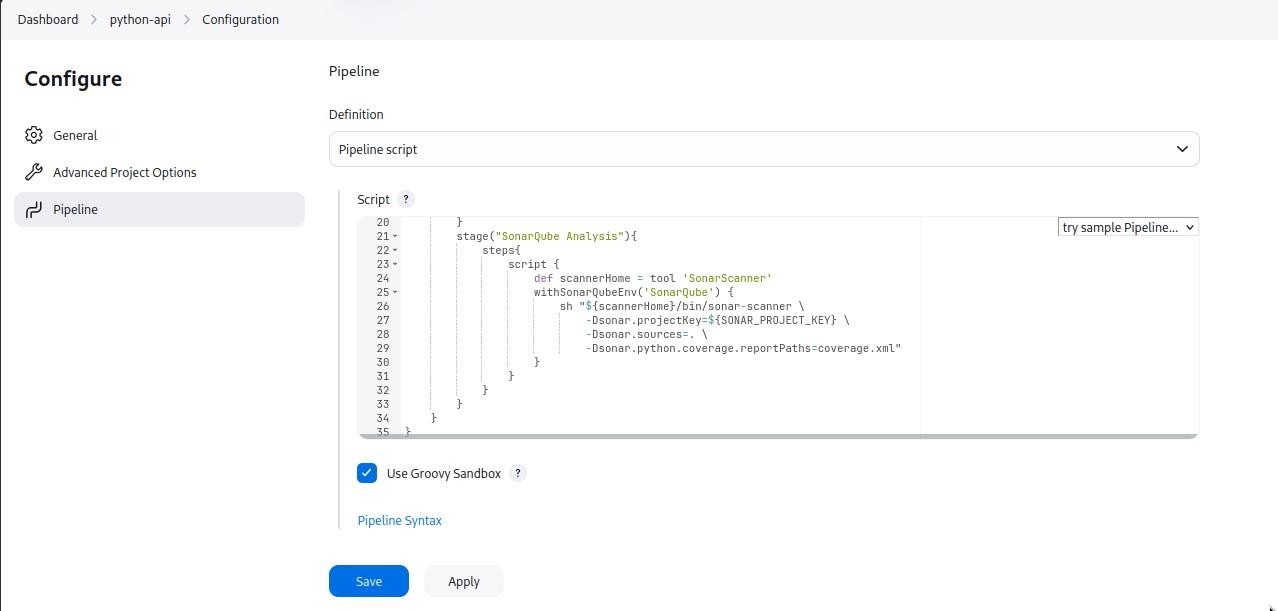
1. Early Detection: Issues are identified as soon as code is committed, allowing forimmediate correction.
2. Consistency: Applies the same quality and security standards across the entire codebaseand development team.
3. Continuous Feedback: Developers receive ongoing insights about their code quality,fostering a culture of continuous improvement.
4. Reduced Review Effort: Automates part of the code review process, allowing humanreviewers 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”



1. 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"

}

}

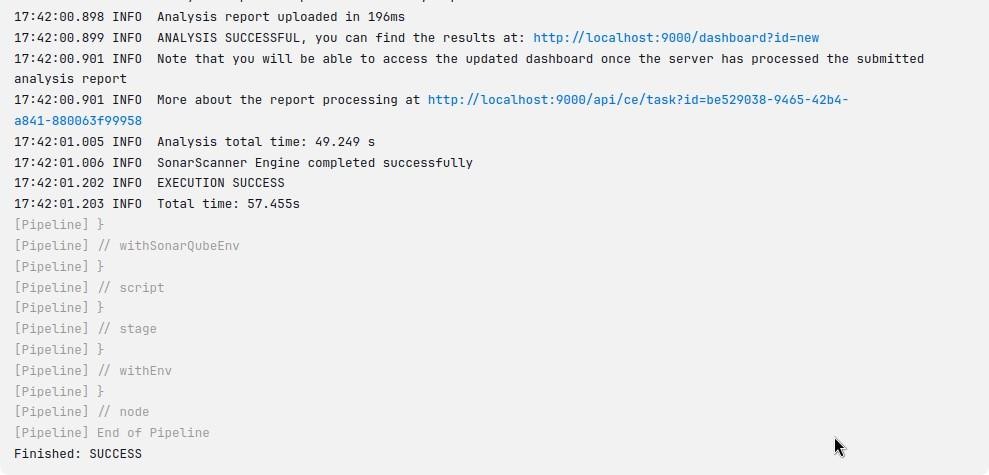
}

}

}

}

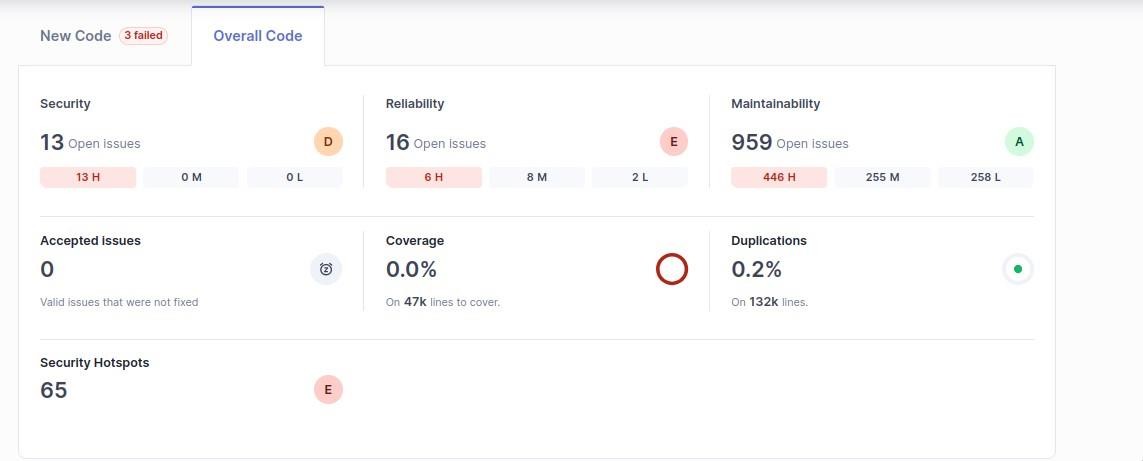
1. Then save, and build.



1. Then On Successful build you will have your report of sonar-scanner containing Failedchecks 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 detectbugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.