Rujuta Medhi **D15A-28**

EXPERIMENT:07

Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Theory:

Static Application Security Testing (SAST) is a method of debugging by examining source code before a program is run. It involves analyzing the application's source code, bytecode, or binary code to identify vulnerabilities and security flaws. SAST tools scan code for common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and buffer overflows, among others.

Problems SAST Solves:

1. **Early Detection of Vulnerabilities**: SAST enables developers to find security flaws early in the development lifecycle, reducing the cost and effort required to fix them later.
2. **Compliance with Security Standards**: It helps organizations comply with various security regulations and standards, such as PCI DSS, OWASP Top Ten, and ISO 27001, by identifying security weaknesses that need to be addressed.
3. **Integration into CI/CD Pipelines**: SAST tools can be integrated into Continuous Integration/Continuous Deployment (CI/CD) pipelines, allowing for automated security checks during the development process.
4. **Comprehensive Coverage**: It scans all code paths and identifies vulnerabilities that may not be detected during dynamic testing (which tests the application while it runs).
5. **Reduction of Technical Debt**: By catching vulnerabilities early, SAST helps prevent the accumulation of technical debt related to security issues, making the codebase more maintainable.
6. **Improved Code Quality**: Besides security, SAST tools often identify coding best practices and help improve overall code quality.
7. **Enhanced Collaboration**: By providing clear reports and insights, SAST tools foster better communication between development and security teams.
8. **Risk Mitigation**: It helps organizations manage risks associated with software vulnerabilities, thereby protecting against data breaches and cyberattacks.

Implementation:

Integrating Jenkins with SonarQube:

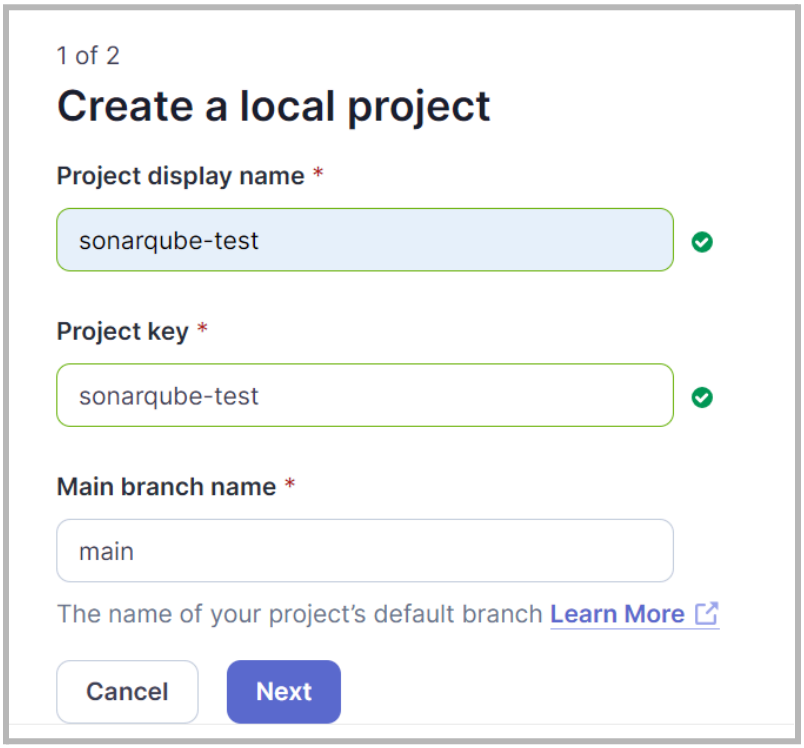
Step 1 Install JDK 1.8

Step 2 download and install jenkins

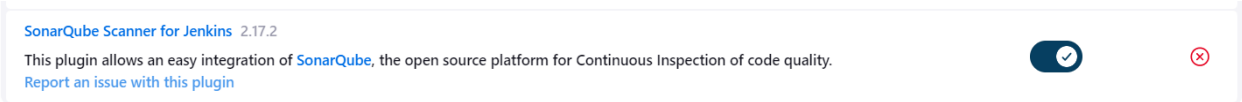
1.Run SonarQube in a Docker container using this command -

```
PS C:\Users\rugved> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:late
st
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
00e224f041dcc60a9cfe77f8e3b79cad3234c7a8ed17d109f58778f655a0d440
PS C:\Users\rugved>
```

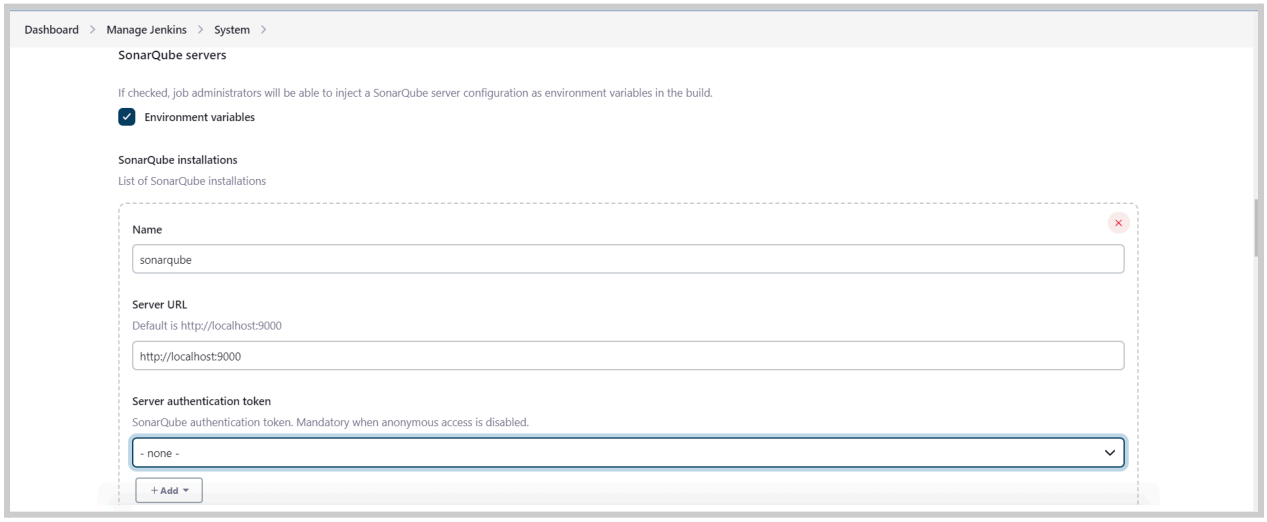
2. Create a manual project in SonarQube with the name sonarqube



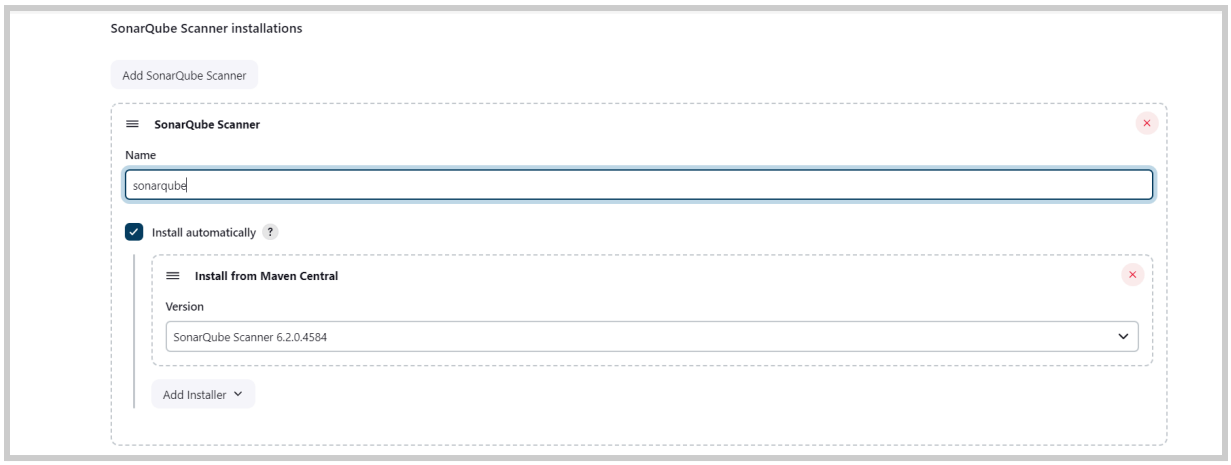
3. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



4. Under Jenkins ‘Configure System’, look for SonarQube Servers and enter the details.



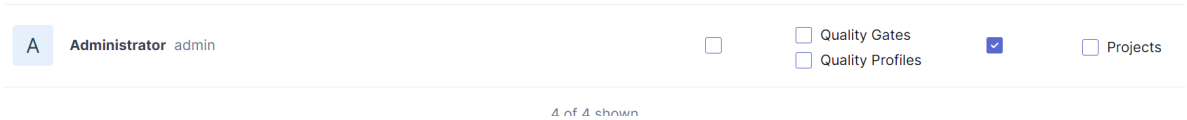
5. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.



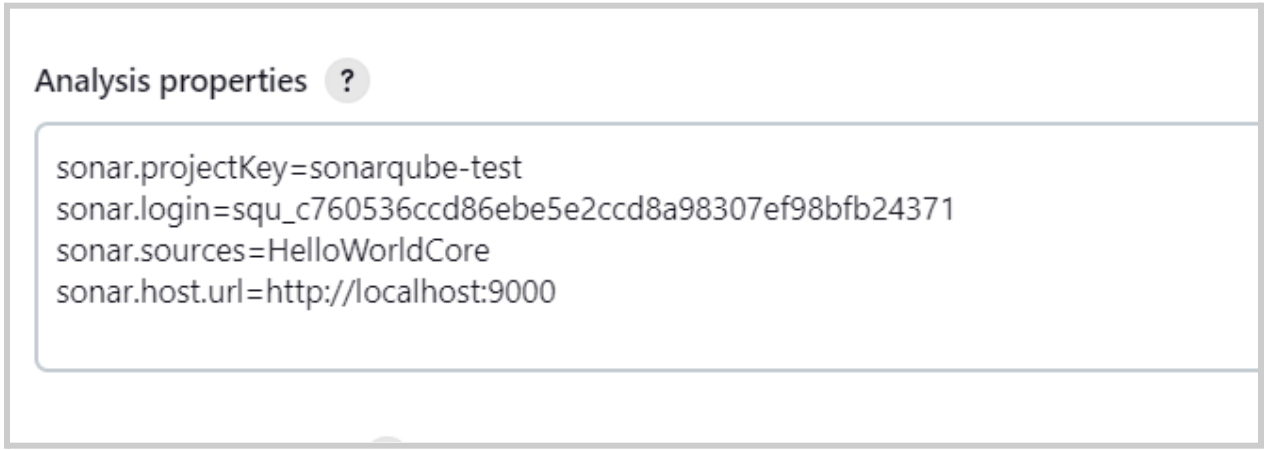
6. create a New Item in Jenkins, choose a freestyle project.

7. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git
It is a sample hello-world project with no vulnerabilities and issues, just to test

8. allow Execute Permissions to the Admin user.



9. Under Build-> Execute SonarQube Scanner, enter these Analysis properties.
Generate the token from Sonarqube->My Account -> Security -> type-User->generate token.



10. After build,check console output

Console Output

Download Copy View as plain text

```
Started by user admin
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.1.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[SonarQube] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube-test -Dsonar.login=squ_c760536ccd86ebe5e2ccd8a98307ef98bfb24371 -
Dsonar.host.url=http://localhost:9000 -Dsonar.sources=HelloWorldCore -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
18:16:45.347 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
18:16:45.361 INFO  Scanner configuration file:
C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-scanner.properties
```

```
18:16:45.396 INFO  SonarScanner CLI 6.2.0.4584
18:16:45.398 INFO  Java 17.0.12 Eclipse Adoptium (64-bit)
18:16:45.399 INFO  Windows 11 10.0 amd64
18:16:45.430 INFO  User cache: C:\WINDOWS\system32\config\systemprofile\.sonar\cache
18:16:47.767 INFO  JRE provisioning: os[windows], arch[amd64]
18:16:59.174 INFO  Communicating with SonarQube Server 10.6.0.92116
18:17:00.085 INFO  Starting SonarScanner Engine...
18:17:00.088 INFO  Java 17.0.11 Eclipse Adoptium (64-bit)
18:17:01.786 INFO  Load global settings
18:17:02.214 INFO  Load global settings (done) | time=428ms
18:17:02.220 INFO  Server id: 147B411E-AZIpEKOrG4EowtOFhNu0
18:17:02.245 INFO  Loading required plugins
18:17:02.246 INFO  Load plugins index
18:17:02.545 INFO  Load plugins index (done) | time=300ms
18:17:02.546 INFO  Load/download plugins
18:17:06.679 INFO  Load/download plugins (done) | time=4133ms
18:17:07.558 INFO  Process project properties
18:17:07.576 INFO  Process project properties (done) | time=18ms
18:17:07.599 INFO  Project key: sonarqube-test
18:17:07.600 INFO  Base dir: C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
18:17:07.601 INFO  Working dir: C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.scannerwork
18:17:07.631 INFO  Load project settings for component key: 'sonarqube-test'
18:17:08.083 INFO  Load project settings for component key: 'sonarqube-test' (done) | time=452ms
18:17:08.134 INFO  Load quality profiles
18:17:08.656 INFO  Load quality profiles (done) | time=522ms
18:17:08.670 INFO  Auto-configuring with CI 'Jenkins'
18:17:08.747 INFO  Load active rules
18:17:36.634 INFO  Load active rules (done) | time=27819ms
```

```
18:17:44.043 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the
SonarScanner for .NET 5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
18:17:44.051 INFO  Sensor C# [csharp] (done) | time=0ms
18:17:44.053 INFO  Sensor Analysis Warnings import [csharp]
18:17:44.056 INFO  Sensor Analysis Warnings import [csharp] (done) | time=0ms
18:17:44.057 INFO  Sensor C# File Caching Sensor [csharp]
18:17:44.062 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
18:17:44.063 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=0ms
18:17:44.064 INFO  Sensor Zero Coverage Sensor
18:17:44.064 INFO  Sensor Zero Coverage Sensor (done) | time=8ms
18:17:44.066 INFO  SCM Publisher SCM provider for this project is: git
18:17:44.066 INFO  SCM Publisher 2 source files to be analyzed
18:17:44.805 INFO  SCM Publisher 2/2 source files have been analyzed (done) | time=742ms
18:17:44.816 INFO  CPD Executor Calculating CPD for 0 files
18:17:44.816 INFO  CPD Executor CPD calculation finished (done) | time=0ms
18:17:44.825 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
18:17:45.202 INFO  Analysis report generated in 165ms, dir size=199.0 kB
18:17:45.279 INFO  Analysis report compressed in 58ms, zip size=20.6 kB
18:17:45.587 INFO  Analysis report uploaded in 305ms
18:17:45.589 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test
18:17:45.589 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
18:17:45.590 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=f1f1bf2b-6ac8-498b-afc3-de9bf4555929
18:17:45.608 INFO  Analysis total time: 38.695 s
18:17:45.611 INFO  SonarScanner Engine completed successfully
18:17:45.722 INFO  EXECUTION SUCCESS
18:17:45.815 INFO  Total time: 1:00.370s
Finished: SUCCESS
```

11. Once the build is complete, check the project in SonarQube.