Krishna Ramchandani
T22-82

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

LO Mapping: LO1, LO4

Theory:
Static Application Security Testing (SAST) Overview: Static Application Security Testing (SAST) is a white-box testing method that analyzes source code or binaries of applications to identify vulnerabilities and security flaws without executing the program. SAST is performed early in the development lifecycle, allowing developers to detect and remediate vulnerabilities before deployment. This proactive approach helps reduce the cost and effort required for security fixes, ultimately leading to a more secure application.

The SAST Process

The SAST process typically involves the following steps:

1. **Code Analysis:**
    - SAST tools scan the source code, binaries, or bytecode to detect vulnerabilities, code quality issues, and compliance violations. These tools use various techniques, including pattern matching, control flow analysis, and data flow analysis.
2. **Vulnerability Detection:**
    - The analysis results in the identification of potential vulnerabilities, such as SQL injection, cross-site scripting (XSS), buffer overflows, and insecure coding practices. SAST tools may also provide recommendations for remediation.
3. **Reporting:**
    - After scanning, SAST tools generate detailed reports outlining identified issues, their severity, and suggested remediation steps. These reports can be integrated into development environments or CI/CD pipelines for easy access by developers.

4. **Remediation:**
    - Developers review the report and take corrective actions to address the identified vulnerabilities. The iterative nature of SAST allows for continuous improvement in code quality and security.
5. **Re-Scanning:**
    - After remediation, the code is re-scanned to verify that vulnerabilities have been resolved and to ensure that no new vulnerabilities have been introduced.

Integrating Jenkins SAST with SonarQube/GitLab

To enhance your CI/CD pipeline with SAST capabilities, integrating Jenkins with tools like SonarQube or GitLab is essential. Below are steps for integrating Jenkins SAST with both SonarQube and GitLab.

Integrating Jenkins with SonarQube

1. **Install Jenkins and SonarQube:**
    - Ensure Jenkins and SonarQube are installed and accessible. You can use Docker for both applications or set them up on dedicated servers.
2. **Install Required Plugins:**
    - In Jenkins, navigate to `Manage Jenkins` > `Manage Plugins` and install the following plugins:
        - **SonarQube Scanner**
        - **SonarQube plugin**
3. **Configure SonarQube in Jenkins:**
    - Go to `Manage Jenkins` > `Configure System`. Under the SonarQube section, add your SonarQube server details, including the server URL and authentication token.
4. **Create a Jenkins Pipeline Job:**
    - In Jenkins, create a new pipeline job. You can use either a freestyle job or a declarative pipeline, depending on your preference.
5. **Add SonarQube Analysis Stage:**

In your Jenkins pipeline configuration, include a stage for SonarQube analysis. For example:
groovy
Copy code

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                // Your build commands
            }
        }
        stage('SonarQube Analysis') {
            steps {
                script {
                    def scannerHome = tool
'SonarQubeScanner'

withSonarQubeEnv('SonarQubeServer') { // Use the name
configured in Jenkins
                        sh
"${scannerHome}/bin/sonar-scanner"
                    }
                }
            }
        }
    }
}
```

-

6. **Run the Pipeline:**
   - Trigger the Jenkins job. The pipeline will build the application and perform SAST analysis using SonarQube, providing results in the SonarQube dashboard.

Krishna Ramchandani
T22-82

Integrating Jenkins with GitLab

1. **Install GitLab and Jenkins:**
   - Ensure both GitLab and Jenkins are installed and configured correctly.
2. **Add Webhook in GitLab:**
   - Go to your GitLab repository, navigate to `Settings` > `Webhooks`, and add a new webhook pointing to your Jenkins job URL. This enables GitLab to trigger builds in Jenkins upon code changes.
3. **Configure Jenkins Pipeline:**
   - Create a Jenkins pipeline similar to the SonarQube integration, ensuring you have the required GitLab API tokens and repository settings.
4. **Add SAST Analysis Stage:**
   - In the Jenkins pipeline, add stages for building the application and performing SAST analysis. You can use SAST tools like SonarQube within this pipeline to analyze the code.
5. **Trigger Build and Analysis:**
   - Each time a commit is made to the GitLab repository, the webhook will trigger the Jenkins pipeline, executing the defined stages, including SAST analysis.

Krishna Ramchandani
T22-82
AdvDevOps Lab7

Output:

Step – 1 : Download and Install Oracle Java 17

Before installing SonarQube, ensure you have Java installed. Follow these steps:

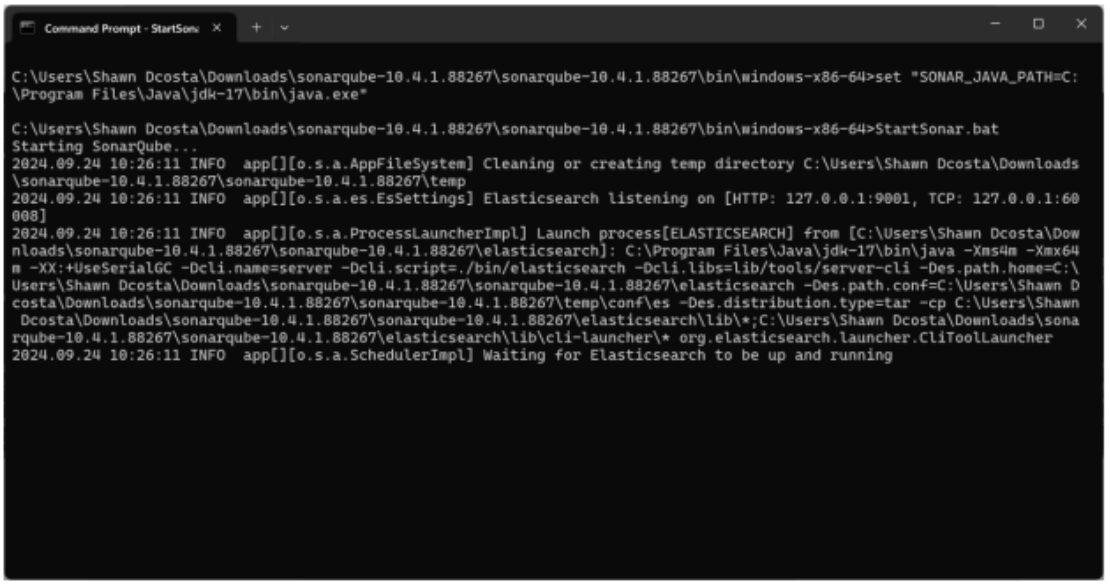
1. Download Oracle Java 17 from Oracle's website.
2. Run the downloaded installer and follow the installation wizard.
3. Set the JAVA_HOME environment variable to C:\Program Files\Java\jdk-17\bin.

Step – 2 : Download and Install SonarQube

1. Download the SonarQube zip file from SonarSource.
2. Extract the contents of the zip file to C:\sonarqube-10.4.1.88267.
3. Set the SONAR_JAVA_PATH environment variable using CMD:

set "SONAR_JAVA_PATH=C:\Program Files\Java\jdk-17\bin\java.exe"

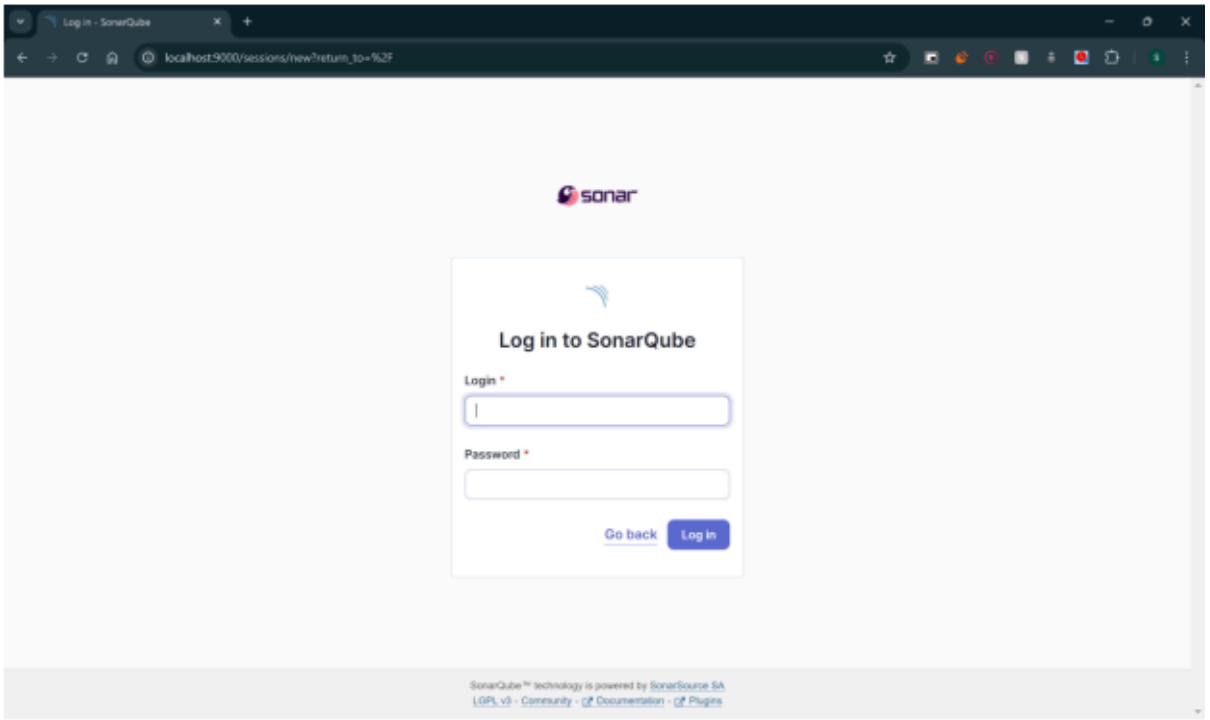
Step – 3 : Start SonarQube.

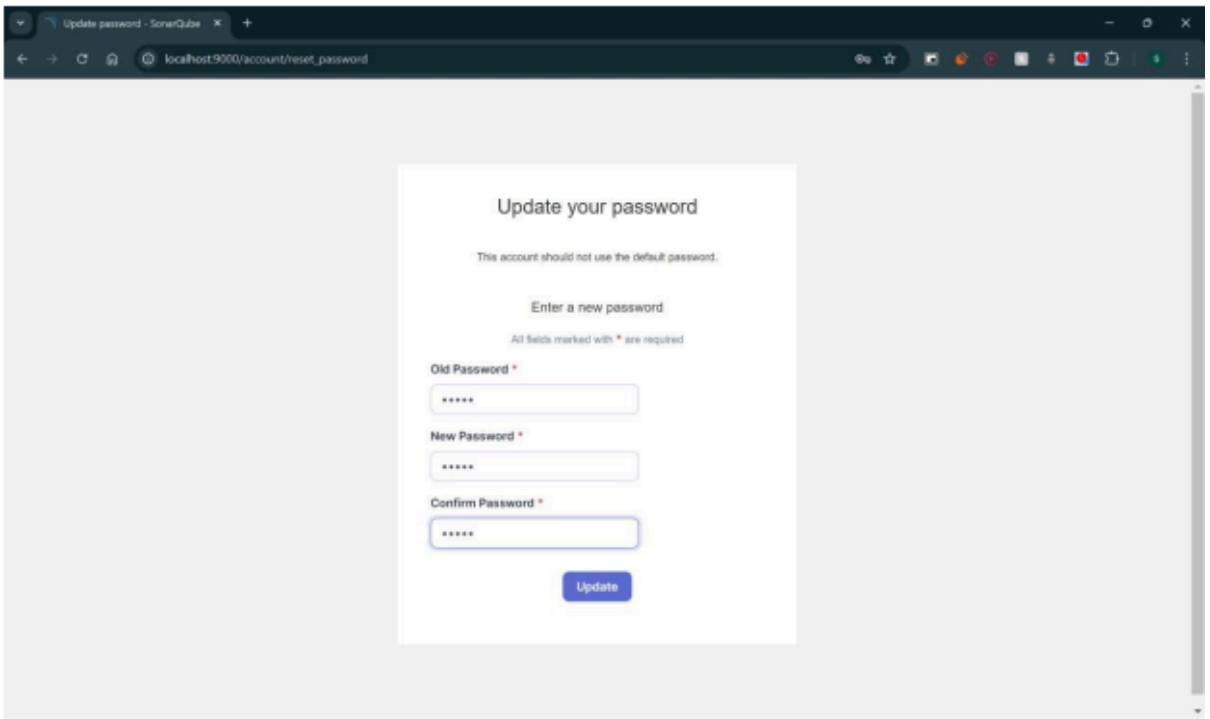


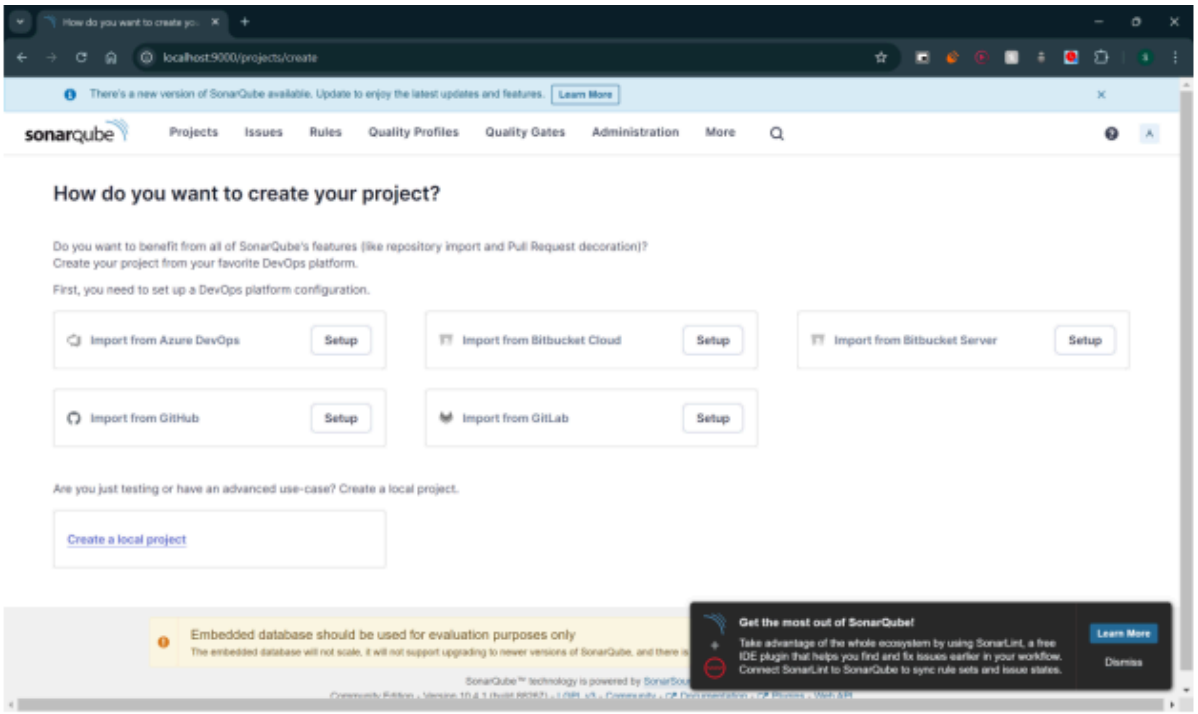
Step – 4 Access SonarQube Web Interface

1. **Open a web browser and navigate to http://localhost:9000.**
2. **Log in with the default credentials (admin/admin).**
3. **Follow the prompts to change the password.**

Krishna Ramchandani
T22-82
AdvDevOps Lab7







Step - 5: Install SonarScanner

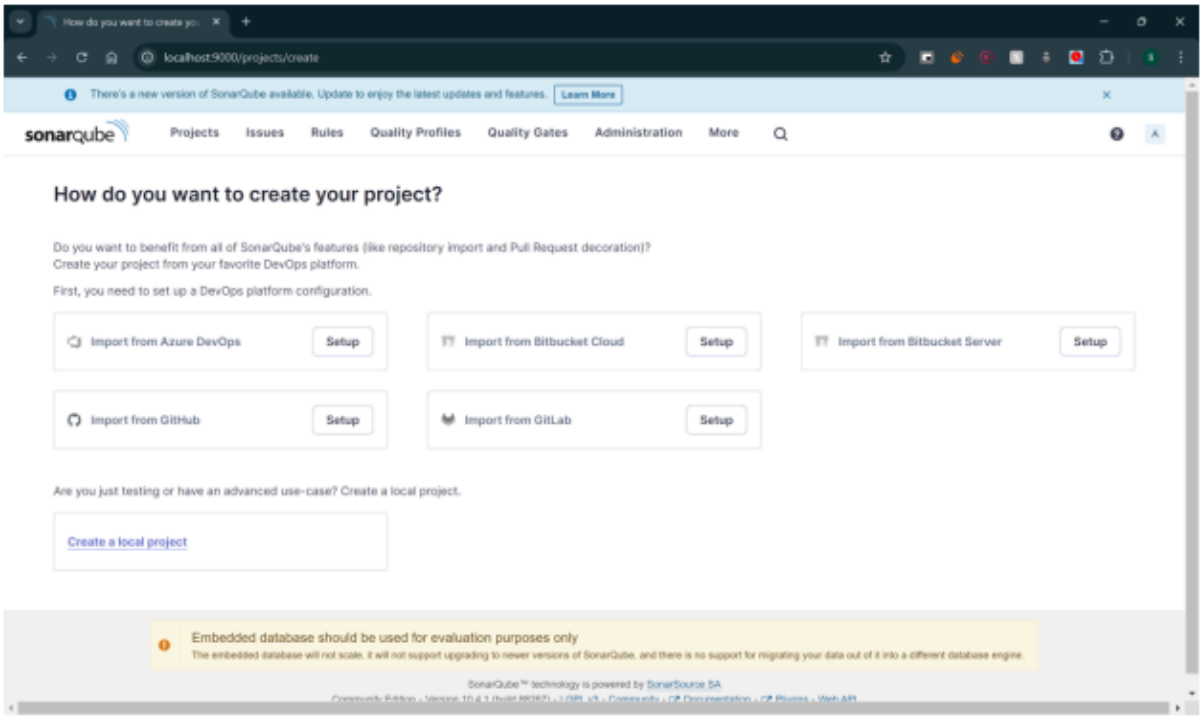
SonarScanner allows you to analyze your projects for code quality. Here's how to install it:

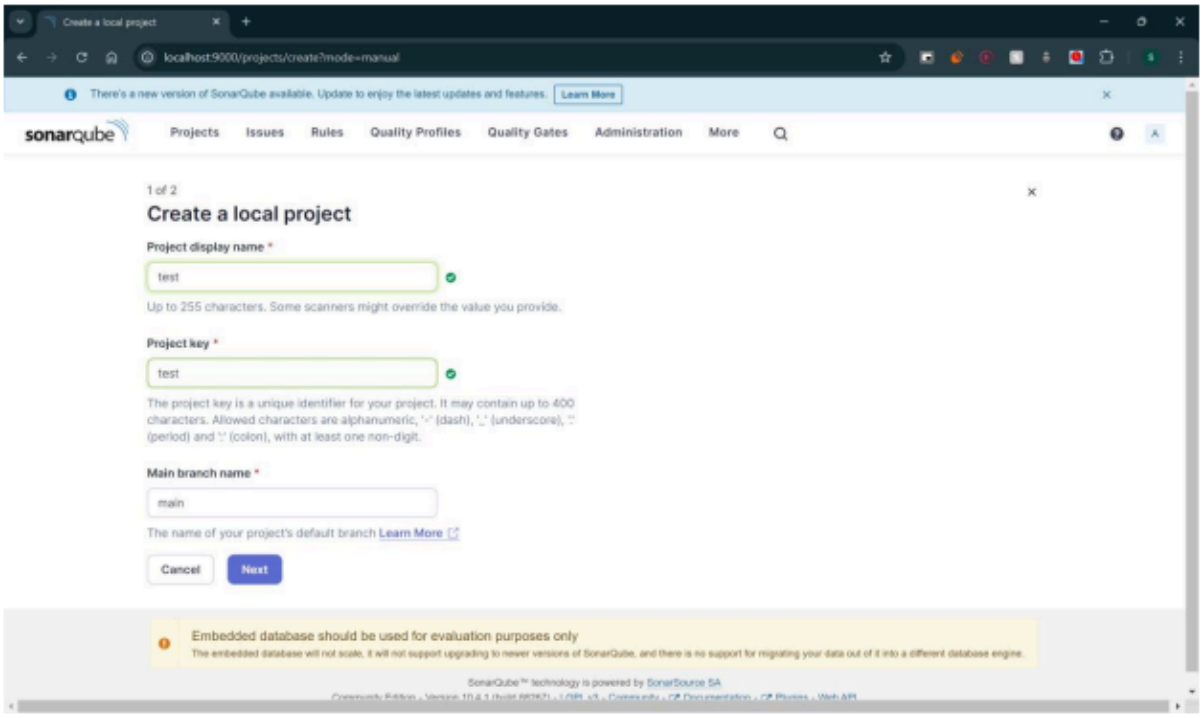
1. **Download SonarScanner CLI from SonarSource.**
2. **Extract the contents to a directory, e.g., C:\sonar-scanner-5.0.1.3006-windows.**

Step – 6 : Create and Analyze a Project in SonarQube

1. **Log in to SonarQube dashboard.**
2. **Navigate to Projects tab > Create Project.**

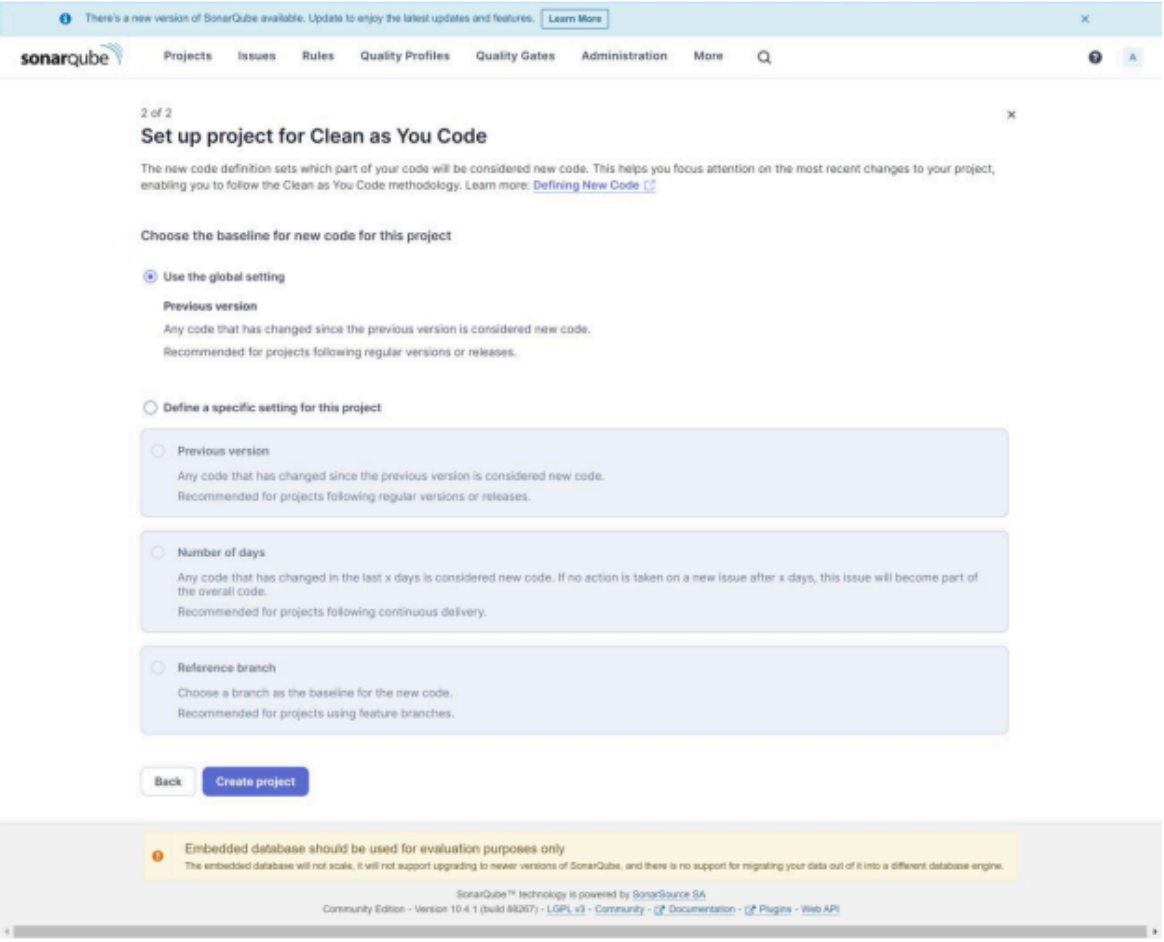
Krishna Ramchandani
T22-82
AdvDevOps Lab7



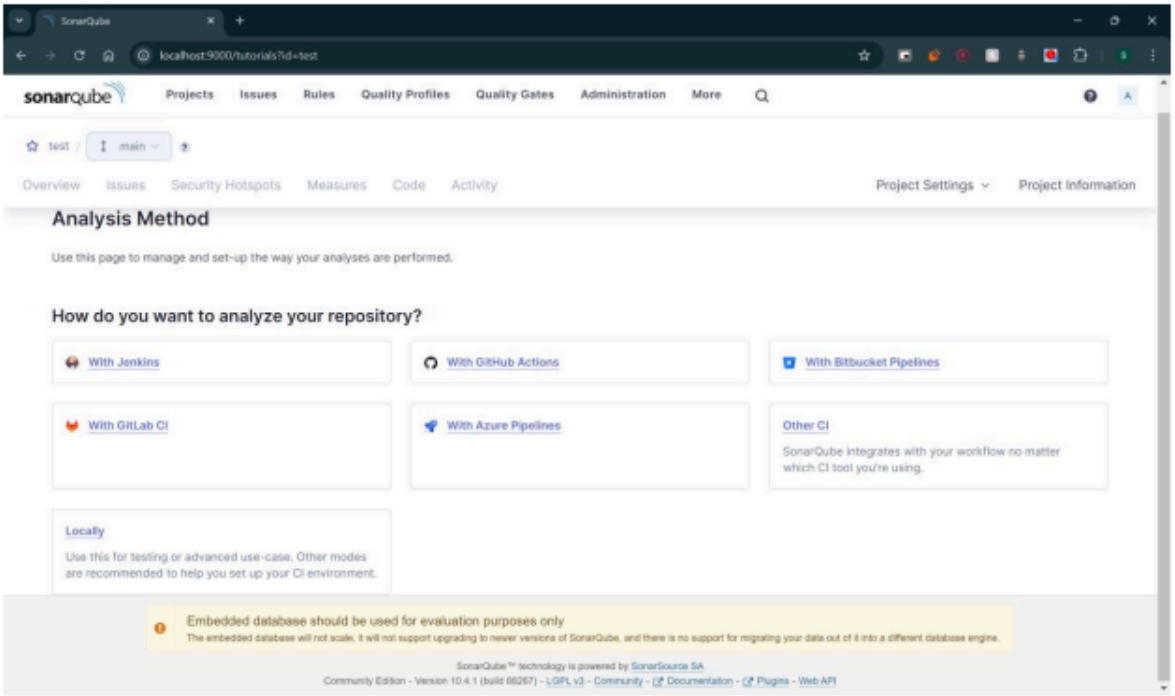


Krishna Ramchandani
T22-82
AdvDevOps Lab7

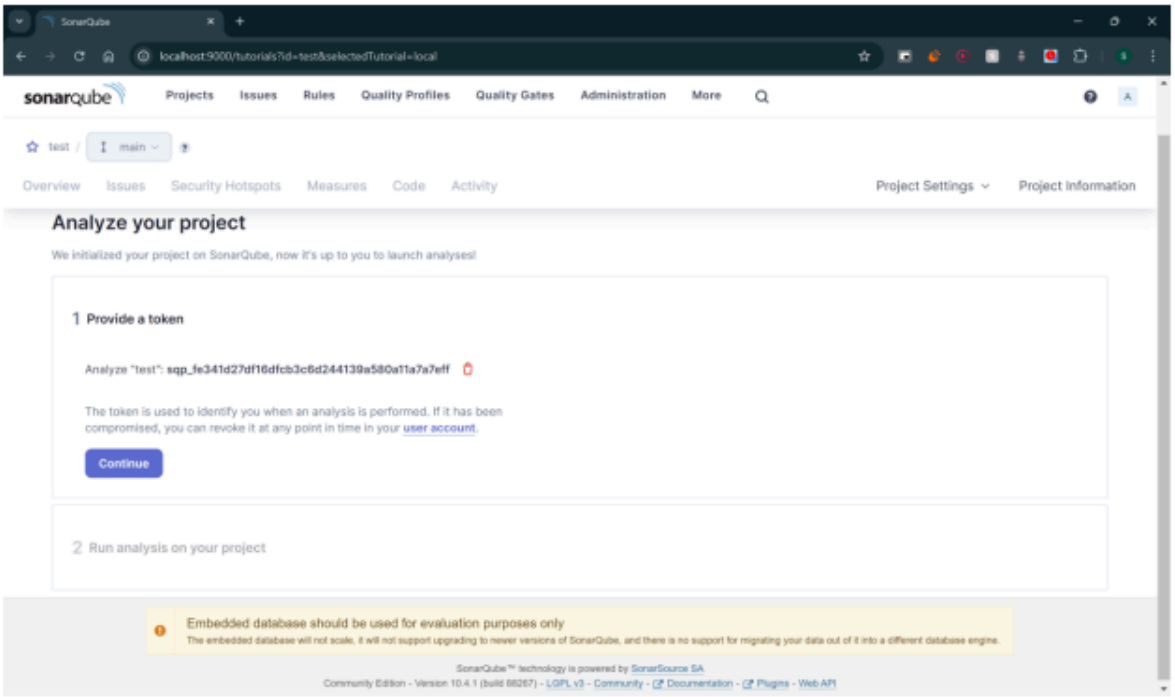
3. Set up the project using global settings.



4. Analyze your project locally using SonarScanner.



5. Provide a token for authentication and execute the SonarScanner command.



Krishna Ramchandani
T22-82
AdvDevOps Lab7

sonarqube | Projects | Issues | Rules | Quality Profiles | Quality Gates | Administration | More

test / main

Overview | Issues | Security Hotspots | Measures | Code | Activity | Project Settings | Project Information

Analysis Method > Locally

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Analyze "test": sqp_fe341d27df16dfcb3c6d244139a580a11a7a7eff

2 Run analysis on your project

What option best describes your build?

Maven | Gradle | .NET | Other (for JS, TS, Go, Python, PHP, ...)

What is your OS?

Linux | Windows | macOS

Download and unzip the Scanner for Windows

Visit the official documentation of the Scanner to download the latest version, and add the `bin` directory to the `%PATH%` environment variable

Execute the Scanner

Running a SonarQube analysis is straighforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner.bat -D"sonar.projectKey=test" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -
D"sonar.token=sqp_fe341d27df16dfcb3c6d244139a580a11a7a7eff"
```

Copy

Please visit the official documentation of the Scanner for more details.

Is my analysis done? If your analysis is successful, this page will automatically refresh in a few moments.

You can set up Pull Request Decoration under the project settings. To set up analysis with your favorite CI tool, see the tutorials.

Check these useful links while you wait:

Branch Analysis

Pull Request Analysis

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

Krishna Ramchandani
T22-82
AdvDevOps Lab7

Step – 7 : Open a new Command Prompt Terminal and paste the command received in the above step in this terminal inorder to analyze the project.
Make sure the path is set to the one in the image below.

```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shawn Dcosta>cd C:\Users\Shawn Dcosta\Downloads\sonar-scanner-cli-5.0.1.3006-windows\sonar-scanner-5.0.1.3006-w
indows\bin

C:\Users\Shawn Dcosta\Downloads\sonar-scanner-cli-5.0.1.3006-windows\sonar-scanner-5.0.1.3006-windows\bin>sonar-scanner.
bat -h
INFO:
INFO: usage: sonar-scanner [options]
INFO:
INFO: Options:
INFO:  -D,--define <arg>     Define property
INFO:  -h,--help             Display help information
INFO:  -v,--version          Display version information
INFO:  -X,--debug            Produce execution debug output

C:\Users\Shawn Dcosta\Downloads\sonar-scanner-cli-5.0.1.3006-windows\sonar-scanner-5.0.1.3006-windows\bin>sonar-scanner.
bat -D"sonar.projectKey=test" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.token=sqp_fe341d27df
16dfcb3c6d244139a580a11a7a7eff"
INFO: Scanner configuration file: C:\Users\Shawn Dcosta\Downloads\sonar-scanner-cli-5.0.1.3006-windows\sonar-scanner-5.0
.1.3006-windows\bin\..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 5.0.1.3006
INFO: Java 17.0.7 Eclipse Adoptium (64-bit)
INFO: Windows 11 10.0 amd64
INFO: User cache: C:\Users\Shawn Dcosta\.sonar\cache
INFO: Analyzing on SonarQube server 10.4.1.88267
INFO: Default locale: "en_IN", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings
INFO: Load global settings (done) | time=140ms
INFO: Server id: 147B411E-AZIiY8Gwb_2sTQ_8ARzZ
INFO: User cache: C:\Users\Shawn Dcosta\.sonar\cache
WARN: sonar.plugins.downloadOnlyRequired is false, so ALL available plugins will be downloaded
INFO: Loading all plugins
INFO: Load plugins index
INFO: Load plugins index (done) | time=107ms
INFO: Load/download plugins
INFO: Load/download plugins (done) | time=3512ms
INFO: Process project properties
INFO: Process project properties (done) | time=55ms
INFO: Execute project builders
INFO: Execute project builders (done) | time=10ms
INFO: Project key: test
INFO: Base dir: C:\Users\Shawn Dcosta\Downloads\sonar-scanner-cli-5.0.1.3006-windows\sonar-scanner-5.0.1.3006-windows\bi
n
INFO: Working dir: C:\Users\Shawn Dcosta\Downloads\sonar-scanner-cli-5.0.1.3006-windows\sonar-scanner-5.0.1.3006-windows
\bin\.scannerwork
INFO: Load project settings for component key: 'test'
INFO: Load project settings for component key: 'test' (done) | time=64ms
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=617ms
WARN: SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the S
CM Sensor in the project settings.
INFO: Load active rules
INFO: Load active rules (done) | time=10093ms
INFO: Load analysis cache
INFO: Load analysis cache (404) | time=22ms
INFO: Preprocessing files...
INFO: 0 languages detected in 2 preprocessed files
INFO: Load project repositories
INFO: Load project repositories (done) | time=101ms
INFO: Indexing files...
INFO: Project configuration:
INFO: 2 files indexed
INFO: ------------- Run sensors on module test
INFO: Load metrics repository
INFO: Load metrics repository (done) | time=66ms
INFO: Sensor JaCoCo XML Report Importer [jacoco]
INFO: 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations: target/site/jacoco/jacoco.xml,targ
et/site/jacoco-it/jacoco.xml,build/reports/jacoco/test/jacocoTestReport.xml
INFO: No report imported, no coverage information will be imported by JaCoCo XML Report Importer
INFO: Sensor JaCoCo XML Report Importer [jacoco] (done) | time=11ms
INFO: Sensor CSS Rules [javascript]
INFO: No CSS, PHP, HTML or VueJS files are found in the project. CSS analysis is skipped.
INFO: Sensor CSS Rules [javascript] (done) | time=3ms
INFO: Sensor C# Project Type Information [csharp]
INFO: Sensor C# Project Type Information [csharp] (done) | time=1ms
INFO: Sensor C# Analysis Log [csharp]
INFO: Sensor C# Analysis Log [csharp] (done) | time=62ms
INFO: Sensor C# Properties [csharp]
INFO: Sensor C# Properties [csharp] (done) | time=0ms
INFO: Sensor HTML [web]
INFO: Sensor HTML [web] (done) | time=4ms
INFO: Sensor TextAndSecretsSensor [text]
INFO: Sensor TextAndSecretsSensor [text] (done) | time=1988ms
INFO: Sensor VB.NET Project Type Information [vbnet]
INFO: Sensor VB.NET Project Type Information [vbnet] (done) | time=1ms
INFO: Sensor VB.NET Analysis Log [vbnet]
INFO: Sensor VB.NET Analysis Log [vbnet] (done) | time=76ms
INFO: Sensor VB.NET Properties [vbnet]
INFO: Sensor VB.NET Properties [vbnet] (done) | time=1ms
INFO: Sensor IaC Docker Sensor [iac]
INFO: 0 source files to be analyzed
INFO: 0/0 source files have been analyzed
INFO: Sensor IaC Docker Sensor [iac] (done) | time=253ms
INFO: ------------- Run sensors on project
INFO: Sensor Analysis Warnings import [csharp]
INFO: Sensor Analysis Warnings import [csharp] (done) | time=3ms
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=10ms
INFO: SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
INFO: CPD Executor Calculating CPD for 0 files
INFO: CPD Executor CPD calculation finished (done) | time=0ms
INFO: Analysis report generated in 263ms, dir size=189.0 kB
INFO: Analysis report compressed in 47ms, zip size=10.3 kB
INFO: Analysis report uploaded in 181ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=test
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis re
port
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=1f92814c-2c2a-460c-bc63-580c249071d2
INFO: Analysis total time: 34.466 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 41.402s
INFO: Final Memory: 28M/74M
INFO: ------------------------------------------------------------------------

C:\Users\Shawn Dcosta\Downloads\sonar-scanner-cli-5.0.1.3006-windows\sonar-scanner-5.0.1.3006-windows\bin>
```

Conclusion:

In conclusion, integrating Static Application Security Testing (SAST) into your CI/CD pipeline using tools like Jenkins and SonarQube or GitLab is crucial for maintaining secure software development practices. By implementing SAST early in the development lifecycle, you can identify and address vulnerabilities proactively, enhancing the overall security posture of your applications. The integration process involves configuring Jenkins to communicate with SonarQube or GitLab, adding analysis stages to the pipeline, and setting up webhooks for seamless interaction between systems. This approach ensures continuous security assessments and fosters a culture of security-first development within teams, ultimately leading to the delivery of high-quality, secure software.