

ADVANCED DEVOPS EXP 7

Static Application Security Testing (SAST)

SAST is a method of security testing that analyzes source code to identify vulnerabilities **without executing the program**. It is also known as **white-box testing**.

SAST Process Breakdown

1. **Code Parsing**
 - The source code is parsed to create an **Abstract Syntax Tree (AST)**, which represents the code structure.
2. **Pattern Matching**
 - The AST is analyzed using predefined rules to detect patterns that may indicate security vulnerabilities.
3. **Data Flow Analysis**
 - This step examines how data moves through the code to identify potential security issues like **SQL Injection** or **Cross-Site Scripting (XSS)**.
4. **Control Flow Analysis**
 - Involves analyzing the paths that the code execution might take to find logical errors or vulnerabilities.
5. **Reporting**
 - The tool generates a report highlighting the vulnerabilities found, their severity, and recommendations for fixing them.

Benefits of SAST

- **Early Detection**
 - Identifies vulnerabilities early in the development lifecycle, reducing the cost and effort required to fix them.
 - **Comprehensive Coverage**
 - Can analyze **100% of the codebase**, including all possible execution paths.
 - **Automated and Scalable**
 - Suitable for large codebases and can be integrated into **CI/CD pipelines** for continuous monitoring.
-

SonarQube and SAST

SonarQube is a popular tool that provides static code analysis to detect bugs, code smells, and security vulnerabilities. Here's how SonarQube fits into the SAST process:

1. Integration

- SonarQube can be integrated into your **CI/CD pipeline** to automatically analyze code every time it is committed.

2. Rule Sets

- It uses a comprehensive set of rules to detect **security vulnerabilities**, **coding standards violations**, and **code quality issues**.

3. Detailed Reporting

- SonarQube generates detailed reports that help developers understand and fix the identified issues efficiently.

4. Continuous Feedback

- Provides continuous feedback to developers, enabling them to maintain high code quality and security standards throughout the development process.

5. Customization

- Allows customization of rule sets to match the specific needs and standards of your project or organization.

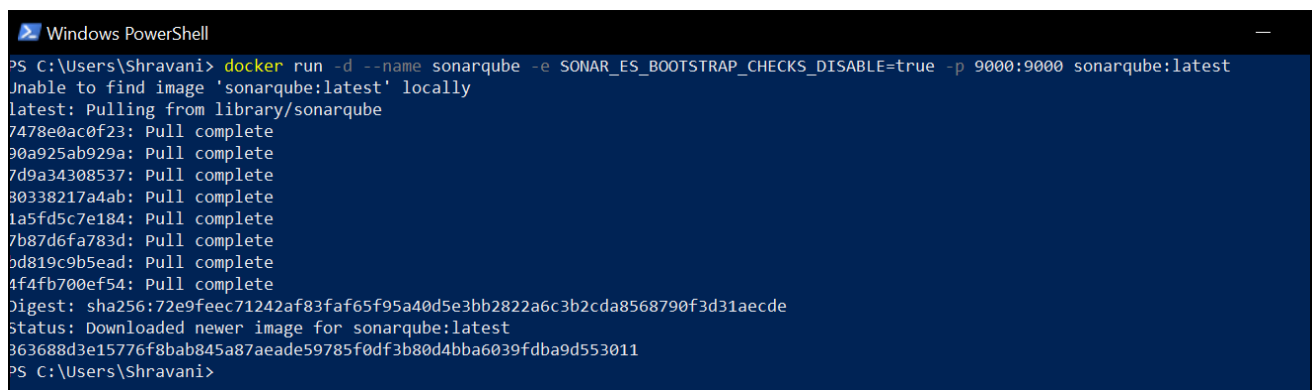
Implementation:

1. Open Jenkins Dashboard

- Access your Jenkins Dashboard by navigating to <http://localhost:8080> (or the port you have configured Jenkins to run on).

2. Run SonarQube in a Docker Container

- Open a terminal and run the following command to start SonarQube in a Docker container
- Command - docker run -d --name sonarqube -e
SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest



```

Windows PowerShell
PS C:\Users\Shravani> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
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
363688d3e15776f8bab845a87aeade59785f0df3b80d4bba6039fdb9d553011
PS C:\Users\Shravani>

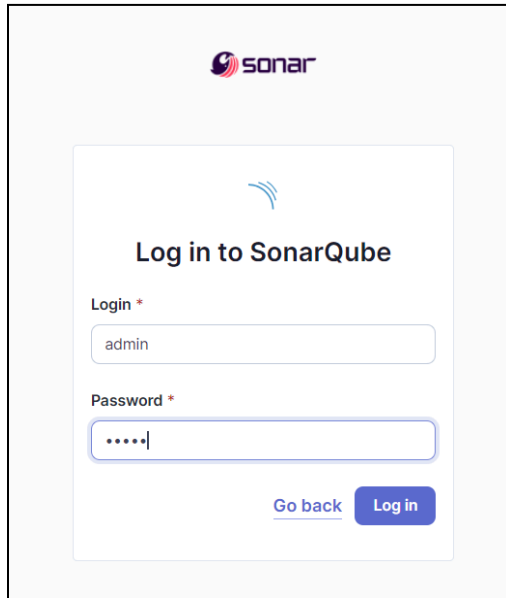
```

3. Check SonarQube Status

- Once the container is up and running, check the status of SonarQube by navigating to <http://localhost:9000>

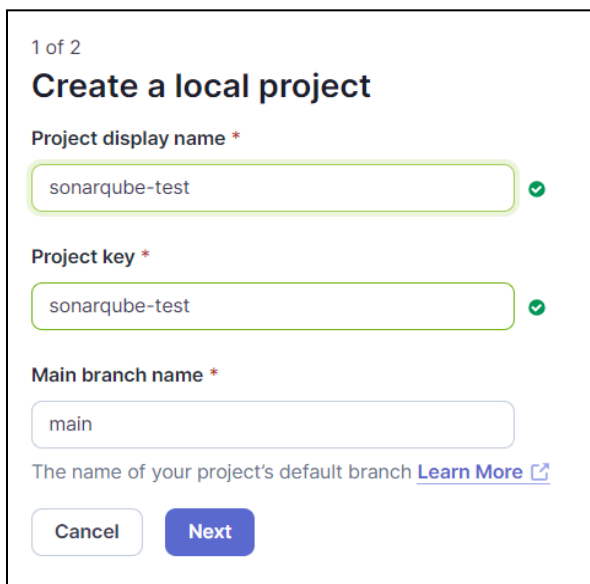
4. Login to SonarQube

- Use the default credentials to log in:
 - Username: admin
 - Password: admin

The image shows the SonarQube login interface. At the top is the Sonar logo. Below it is a white box with the title "Log in to SonarQube". Inside this box, there are two input fields: "Login *" with the value "admin" and "Password *" with masked characters ".....". Below the password field are two buttons: "Go back" (a link) and "Log in" (a blue button).

5. Create a Project in SonarQube

- Create a new project manually in SonarQube and name it sonarqube

The image shows the "Create a local project" form in SonarQube. It is labeled "1 of 2" at the top. The title is "Create a local project". There are three required fields: "Project display name *" with the value "sonarqube-test" and a green checkmark, "Project key *" with the value "sonarqube-test" and a green checkmark, and "Main branch name *" with the value "main". Below the last field is a note: "The name of your project's default branch" followed by a link "Learn More" and an external link icon. At the bottom are two buttons: "Cancel" and "Next".

2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

☒ Use the global setting

Previous version
Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

☐ Previous version
Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

6. Install SonarQube Scanner for Jenkins

- Go back to the Jenkins Dashboard.
- Navigate to Manage Jenkins > Manage Plugins.
- Search for SonarQube Scanner for Jenkins and install it.

Dashboard > Manage Jenkins > Plugins

Plugins

☒ Updates 20
 ☐ Available plugins
 ☐ Installed plugins
 ☐ Advanced settings

☒ Install
 ☐ Name
 ☐ Released

<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	7 mo 7 days ago
<input type="checkbox"/>	Sonar Quality Gates 315.v1f12b_e81a_3a_4 Library plugins (for use by other plugins) analysis Other Post-Build Actions Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")	28 days ago

Dashboard > Manage Jenkins > Plugins

Plugins

☒ Updates 20
 ☐ Available plugins
 ☐ Installed plugins
 ☐ Advanced settings
 ☒ Download progress

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

SonarQube Scanner

Success

Loading plugin extensions

Success

→ [Go back to the top page](#)

(you can start using the installed plugins right away)

→ ☐ Restart Jenkins when installation is complete and no jobs are running

7. Configure SonarQube in Jenkins

- Go to Manage Jenkins > Configure System
- Scroll down to the SonarQube Servers section and enter the required details:
 - **Name:** Any name you prefer.
 - **Server URL:** http://localhost:9000
 - **Server Authentication Token:** (Generate this token in SonarQube under My Account > Security > Generate Tokens).
 - **Add Jenkins:** Select Kind - Secret Text > Secret (Paste Generated Token)

Security

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

Name	Type	Expires in	
<input type="text" value="Enter Token Name"/>	<div>Select Token Type</div>	<div>30 days</div>	<div>Generate</div>

Name	Type	Project	Last use	Created	Expiration	
sonarqube	Global		Never	September 26, 2024	October 26, 2024	<div>Revoke</div>

SonarQube installations

List of SonarQube installations

Name

Server URL

Default is http://localhost:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

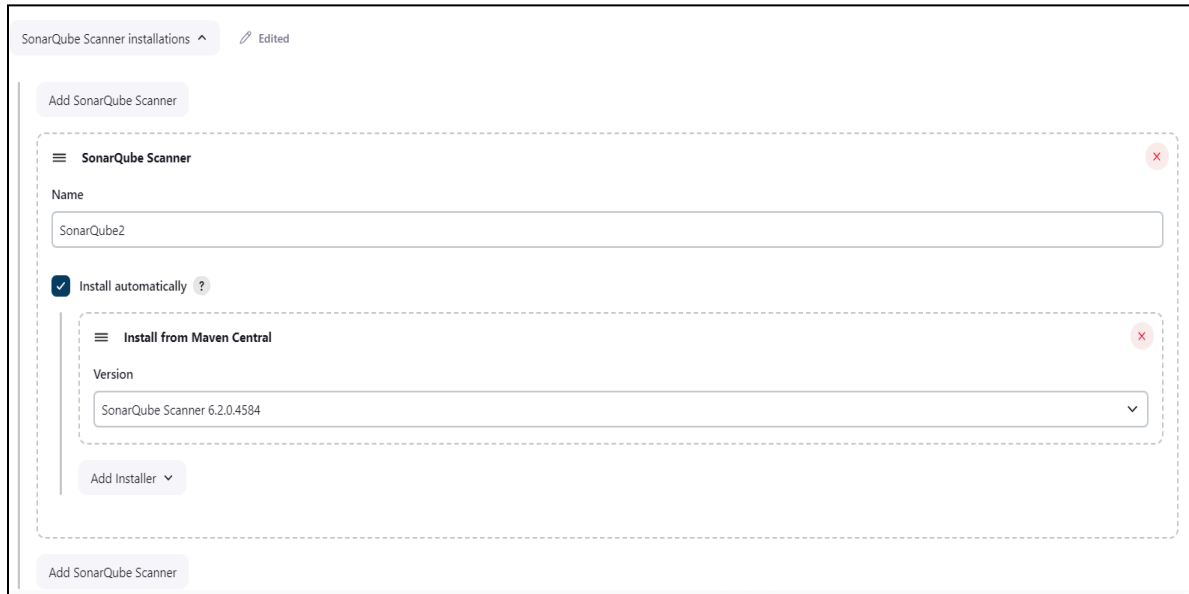
+ Add

Advanced

Add SonarQube

8. Configure SonarQube Scanner in Jenkins

- Go to Manage Jenkins > Global Tool Configuration.
- Scroll down to SonarQube Scanner.
- Choose the latest version and select Install automatically



The screenshot shows the 'SonarQube Scanner installations' configuration page in Jenkins. At the top, there's a header 'SonarQube Scanner installations' with an 'Edited' status. Below it, a button 'Add SonarQube Scanner' is visible. The main configuration area is titled 'SonarQube Scanner' and contains a 'Name' field with the value 'SonarQube2'. Below the name field, the 'Install automatically' checkbox is checked. Under this, there's a section titled 'Install from Maven Central' which contains a 'Version' dropdown menu showing 'SonarQube Scanner 6.2.0.4584'. At the bottom of this section is an 'Add Installer' button. The entire configuration area is enclosed in a dashed box with a close button (X) in the top right corner.

9. Create a New Jenkins Job

- In Jenkins, create a new item and select Freestyle project.
- Under Source Code Management, choose Git and enter the repository URL:
https://github.com/shazforiot/MSBuild_firstproject.git

New Item

Enter an item name

SonarQube2

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a

OK

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

`https://github.com/shazfortiot/MSBuild_firstproject.git`

Credentials ?

- none -

+ Add

Advanced

10. Configure Build Steps

- Under the Build section, add a build step to Execute SonarQube Scanner
- Enter the following analysis properties:
 - `sonar.projectKey=my_project_name`
 - `sonar.login=your_generated_token`
 - `sonar.sources=HelloWorldCore`
 - `sonar.host.url=http://localhost:9000`

Build Steps

Execute SonarQube Scanner

JDK ?

JDK to be used for this SonarQube analysis

(Inherit From Job)

Path to project properties ?

Analysis properties ?

`sonar.projectKey=sonarqube-test`
`sonar.login=sqa_7d80b92445edfedadb52b0bfa57c5978e192bf8`
`sonar.sources=HelloWorldCore`
`sonar.host.url=http://localhost:9000`

Additional arguments ?

JVM Options ?

11. Set Permissions in SonarQube

- Navigate to <http://localhost:9000/<user-name>/permissions>.
- Allow Execute Permissions to the Admin user.

Global Permissions

Grant and revoke permissions to make changes at the global level. These permissions include editing Quality Profiles, executing analysis, and performing global system administration.

All
Users
Groups

	Administer System ?	Administer ?	Execute Analysis ?	Create ?
sonar-administrators <small>System administrators</small>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
sonar-users <small>Every authenticated user automatically belongs to this group</small>	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects
Administrator admin	<input checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects
Anyone <small>DEPRECATED</small> <small>Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users.</small>	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input type="checkbox"/> Projects

4 of 4 shown

12. Run the Build

- Go back to Jenkins and run the build.
- Check the console output for any errors or issues.

Dashboard > SonarQube2 > #6 > Console Output

Status

Changes

Console Output

Edit Build Information

Timings

Git Build Data

Previous Build

Console Output

Download
Copy
View as plain text

Started by user **Shravani Rasam**

Running as SYSTEM

Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\SonarQube2

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\SonarQube2\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10

Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git

> git.exe --version # timeout=10

> git --version # 'git version 2.43.0.windows.1'

> git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject.git +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

[SonarQube2] \$ C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\SonarQube2\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube-test -Dsonar.login=sqa_7d80b92445edfedadb52b0fbfa57c5978e192bf8 -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=HelloWorldCore -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\SonarQube2

21:58:29.773 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'

21:58:29.784 INFO Scanner configuration file: C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\SonarQube2\bin\...\conf\sonar-scanner.properties

13. Verify in SonarQube

- Once the build is complete, check the project in SonarQube to see the analysis results.

