

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

Theory:

What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and

others—with high confidence. Thus, integrating static analysis into the SDLC can yield dramatic results in the overall quality of the code developed.

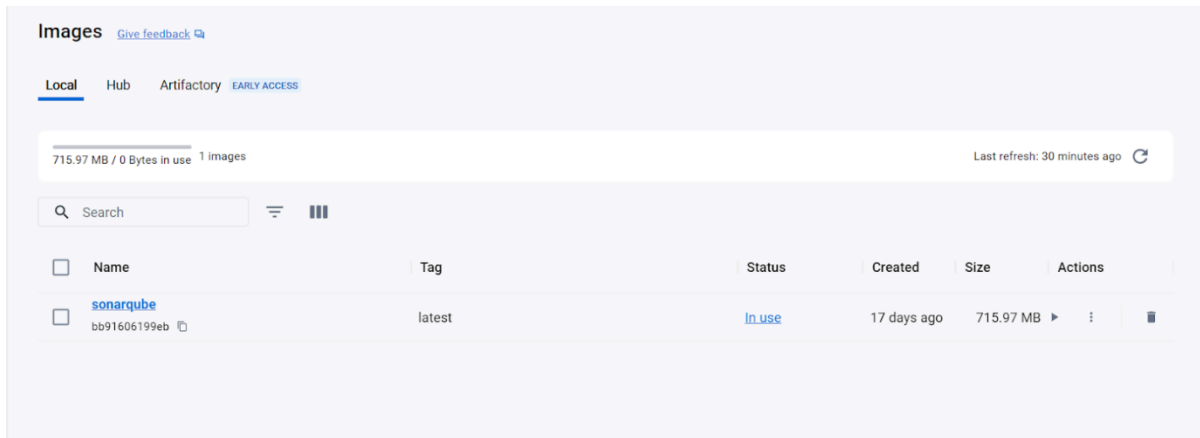
What are the key steps to run SAST effectively?

There are six simple steps needed to perform SAST efficiently in organizations that have a very large number of applications built with different languages, frameworks, and platforms.

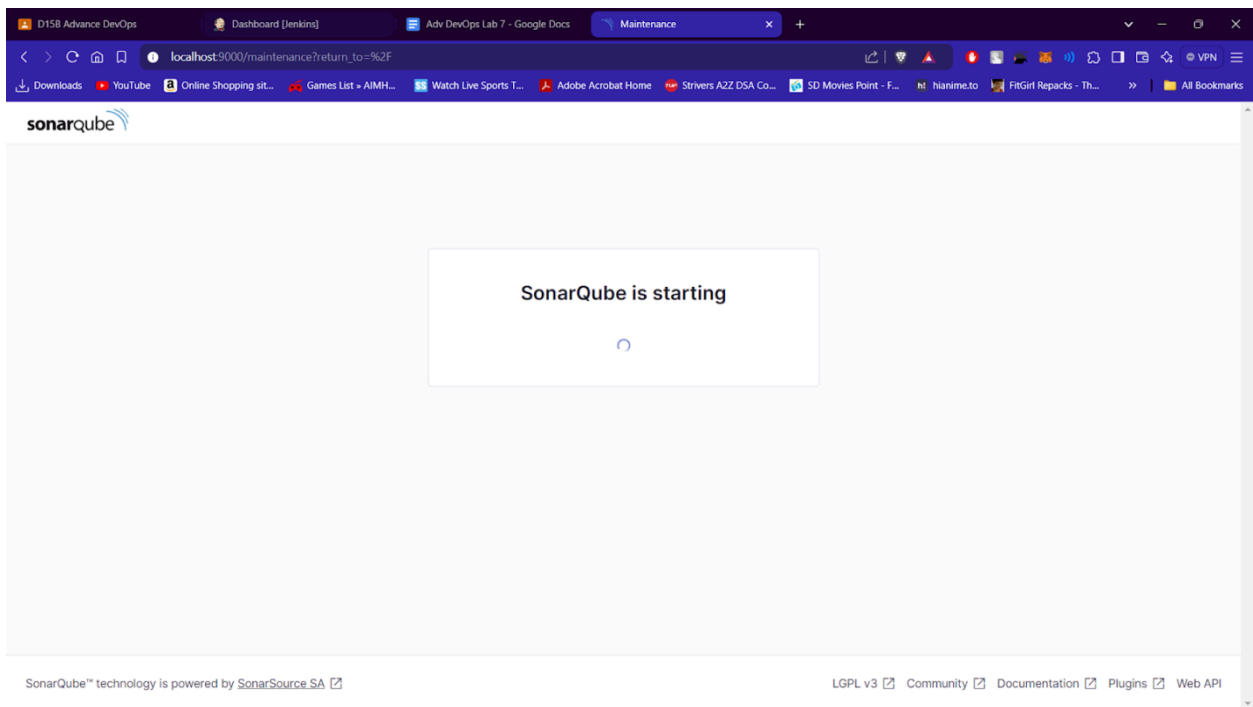
1. **Finalize the tool.** Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.
2. **Create the scanning infrastructure, and deploy the tool.** This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.
3. **Customize the tool.** Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build environment, create dashboards for tracking scan results, and build custom reports.
4. **Prioritize and onboard applications.** Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with application scans synced with release cycles, daily or monthly builds, or code check-ins.
5. **Analyze scan results.** This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.
6. **Provide governance and training.** Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints

Steps:

```
$ docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000  
Sonarqube:latest
```



go to the SonarQube page by typing:
`http://localhost:9000/` on your browser.



Installation is successful if you see this page
Update to new password

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

sonarqube / main

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

Analysis Method

Use this page to manage and set-up the way your analyses are performed.

How do you want to analyze your repository?

With Jenkins

With GitHub Actions

With Bitbucket Pipelines

With GitLab CI

With Azure Pipelines

Other CI
SonarQube integrates with your workflow no matter which CI tool you're using.

Locally
Use this for testing or advanced use-case. Other modes are recommended to help you set up your CI environment.

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.

Create project manually:
Here project name is “AdDevops”

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

1 of 2

Create a local project

Project display name *
sonarqube

Project key *
sonarqube

Main branch name *
main

The name of your project's default branch [Learn More](#)

CancelNext

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.

SonarQube™ technology is powered by SonarSource SA

Community Edition v10.6 (92116) ACTIVE LGPL v3 Community Documentation Plugins Web API

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

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.

☐ Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
Recommended for projects following continuous delivery.

☐ Reference branch

Choose a branch as the baseline for the new code.
Recommended for projects using feature branches.

Create project

open Jenkins

Go to Dashboard ->Manage Jenkins -> Plugin Manager and search for SonarQube Scanner under Available plugins for Jenkins and install without restart.

Plugins

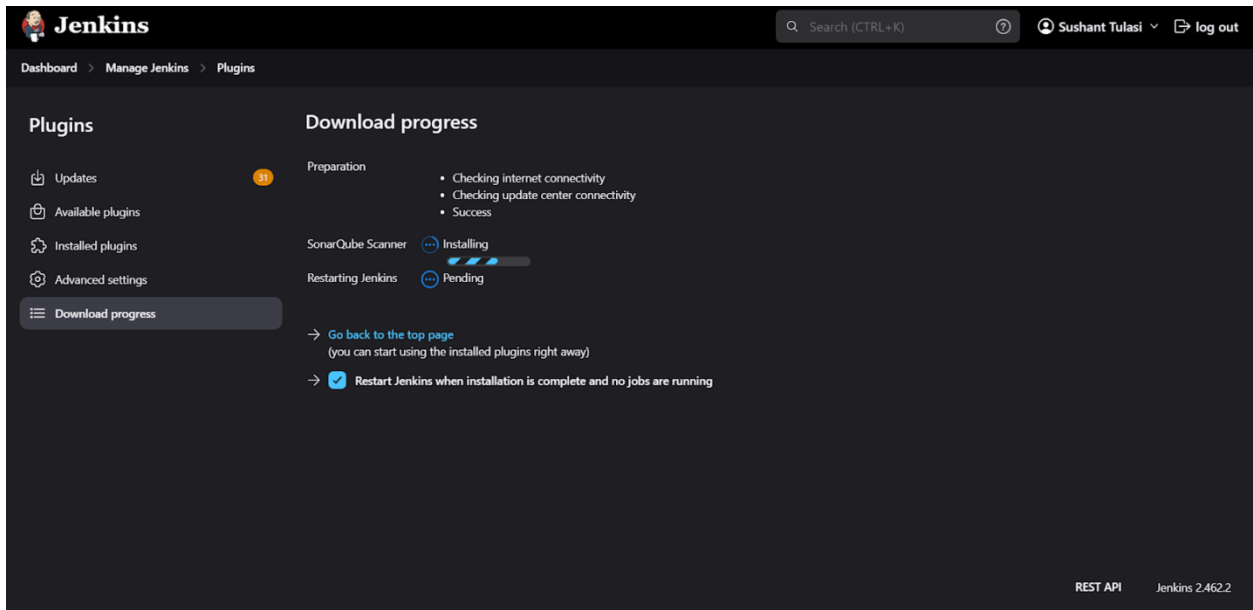
| Install | Name | Released |
|-------------------------------------|---|---------------|
| <input checked="" type="checkbox"/> | <div><div>SonarQube Scanner 2.15</div><div>External Site/Tool Integrations · Build Reports</div><div>This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.</div></div> | 10 mo ago |
| <input type="checkbox"/> | <div><div>Sonar Gerrit 384.vdb_755265c28d</div><div>External Site/Tool Integrations</div><div>This plugin allows to submit issues from SonarQube to Gerrit as comments directly.</div></div> | 20 days ago |
| <input type="checkbox"/> | <div><div>SonarQube Generic Coverage 1.0</div><div>TODO</div></div> | 4 yr 1 mo ago |

Install without restart

Download now and install after restart

Update information obtained: 1 day 11 hr ago

Check now



Under Jenkins ,
Dashboard -> Manage Jenkins -> Configure System ,
Look for SonarQube Servers and enter the details. Enter the Server Authentication Token if needed.

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

Search SonarQube Scanner under Dashboard -> Manage Jenkins -> Global Tool Configuration.
Choose the latest configuration and choose Install Automatically.

SonarQube Scanner

Name

SonarQube

☒ Install automatically ?

Install from Maven Central

Version

SonarQube Scanner 6.2.0.4584

Add Installer

Add SonarQube Scanner

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

New Item

Enter an item name

AdvDevOps7

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.

Choose this GitHub repository in Source Code Management.

https://github.com/shazforiot/MSBuild_firstproject.git

Under Build ->Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, and Host URL.

sonar.projectKey=AdDevops

sonar.login=admin

sonar.password=abc

sonar.hosturl=<http://localhost:9000/>

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?

- none -

+ Add ▾

Advanced ▾

Add Repository

Execute SonarQube Scanner

JDK ?

JDK to be used for this SonarQube analysis

(Inherit From Job) ▾

Path to project properties ?

Analysis properties ?

```
sonar.projectKey=sonarqube
sonar.login=admin
sonar.password=9136591220
sonar.sources=
sonar.host.url=http://localhost:9000
```

Additional arguments ?

▾

JVM Options ?

▾

Go to <http://localhost:9000/> and enter your previously created username.
Go to Permissions and grant the Admin user Execute Permissions.

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More

AdDevops / main

Overview Issues Security Hotspots Measures Code Activity

Project Settings Project Information

Permissions

Grant and revoke project-level permissions. Permissions can be granted to groups or individual users. This project is public. Anyone can browse and see the source code.

☒ Public ☐ Private

| | Users | Groups | Search for users or groups... | Administer Issues | Administer Security Hotspots | Administer | Execute Analysis |
|--|-------|--------|-------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| sonar-administrators System administrators | | | | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| sonar-users Every authenticated user automatically belongs to this group | | | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Anyone DEPRECATED Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users. | | | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Administrator admin | | | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |

4 of 4 shown

Build and Run:

Dashboard > AdDevopsLab7

Status

Changes

Workspace

Build Now

Configure

Delete Project

SonarQube

Rename

Project AdDevopsLab7

This is Lab 7.

Permalinks

[Edit description](#)

[Disable Project](#)

Build History trend

Filter builds...

1
Sep 18, 2023, 10:11 PM

[Atom feed for all](#) [Atom feed for failures](#)

Jenkins

Dashboard > AdDevopsLab7 > #1

Status

Changes

Console Output

Edit Build Information

Delete build '#1'

Git Build Data

Build #1 (Sep 18, 2023, 10:31:52 PM)

No changes.

Started by user [Prajakta Upadhye](#)

Revision: f2bc042c04c6e72427c380bcaee6d6fee7b49adf
Repository: https://github.com/PrajaktaUpadhye6/MSBuild_firstproject.git

- refs/remotes/origin/master

Console Output:

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete Build 'IT'

Git Build Data

✓ Console Output

Started by user [Prajakta Upadhye](#)
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\jenkins\jenkins\workspace\AdDevopsLab7
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/PrajaktaUpadhye/MSBuild_FirstProject.git
> git.exe init C:\ProgramData\jenkins\jenkins\workspace\AdDevopsLab7 # timeout=10
Fetching upstream changes from https://github.com/PrajaktaUpadhye/MSBuild_FirstProject.git
> git.exe --version # timeout=10
> git --version # git version 2.39.2.windows.1
> git.exe fetch --tags --force --progress -- https://github.com/PrajaktaUpadhye/MSBuild_FirstProject.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/PrajaktaUpadhye/MSBuild_FirstProject.git # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcae6d6fee7b49adf (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
Commit message: "update"
First time build. Skipping changelog.
Unpacking <https://repo.maven.org/maven2/org/sonarsource/scanner/cli/sonar-scanner-cli/5.0.1.3006/sonar-scanner-cli-5.0.1.3006.zip> to C:\ProgramData\jenkins\jenkins\tools\udson\plugins\sonar.SonarRunnerInstallation\AdDevops on Jenkins
[AdDevopsLab7] \$ C:\ProgramData\jenkins\jenkins\tools\udson\plugins\sonar.SonarRunnerInstallation\AdDevops\bin\sonar-scanner.bat -Dsonar.projectkey=AdDevops -Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.password=admin -Dsonar.projectBaseDir=C:\ProgramData\jenkins\jenkins\workspace\AdDevopsLab7
INFO: Scanner configuration file: C:\ProgramData\jenkins\jenkins\tools\udson\plugins\sonar.SonarRunnerInstallation\AdDevops\bin\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 5.0.1.3006
INFO: Java 18.0.2.1 Oracle Corporation (64-bit)

14:42:33.145 INFO CPU executor CPU calculation finished (done) | time=0ms

14:42:33.151 INFO SCM revision ID 'f2bc042c04c6e72427c380bcae6d6fee7b49adf'

14:42:33.466 INFO Analysis report generated in 95ms, dir size=201.0 kB

14:42:33.546 INFO Analysis report compressed in 54ms, zip size=22.4 kB

14:42:33.834 INFO Analysis report uploaded in 285ms

14:42:33.847 INFO ANALYSIS SUCCESSFUL, you can find the results at: <http://localhost:9000/dashboard?id=sonarqube>

14:42:33.847 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report

14:42:33.847 INFO More about the report processing at <http://localhost:9000/api/ce/task?id=7db13258-220d-47ae-9e4e-dcbe63771ea>

14:42:33.862 INFO Analysis total time: 27.320 s

14:42:33.863 INFO SonarScanner Engine completed successfully

14:42:33.947 INFO EXECUTION SUCCESS

14:42:33.949 INFO Total time: 42.241s

Finished: SUCCESS

The screenshot shows the SonarQube interface for a project named 'main'. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main header shows 'Overview', 'Issues', 'Security Hotspots', 'Measures', 'Code', and 'Activity'. The 'Quality Gate' section is highlighted, showing a 'Passed' status with a green checkmark. A warning message states: 'The last analysis has warnings. See details'. The 'Last analysis' was performed '1 minute ago'. Below the status, there are tabs for 'New Code' and 'Overall Code'. The 'Overall Code' tab is active, displaying a grid of metrics: Security (0 Open issues), Reliability (0 Open issues), Maintainability (0 Open issues), Accepted issues (0), Coverage (0.0%), and Duplications (0.0%). Each metric is accompanied by a circular progress indicator showing the current status relative to the target.

| Metric | Value | Status |
|-----------------|---------------|---------|
| Quality Gate | Passed | Success |
| Security | 0 Open issues | Success |
| Reliability | 0 Open issues | Success |
| Maintainability | 0 Open issues | Success |
| Accepted issues | 0 | Success |
| Coverage | 0.0% | Failure |
| Duplications | 0.0% | Success |