

## Adv DevOps Practical 7

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

### Integrating Jenkins with SonarQube:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

### Steps to integrate Jenkins with SonarQube

1. Open up Jenkins Dashboard on localhost, port 8090 or whichever port it is at for you.

The screenshot shows the Jenkins Dashboard interface. The top navigation bar includes the Jenkins logo, a search bar, and user information (Sahil Motiramani). The left sidebar contains links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area displays a table of build history with columns for status, icon, name, last success, last failure, and last duration. Below the table, there are sections for 'Build Queue' (showing no builds) and 'Build Executor Status' (showing two idle executors and one offline executor named 'Sahil').

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	sahil 7	24 days #2	N/A	96 ms
✓	☀	Sahil exp6	24 days #3	N/A	1 sec
⏸	☀	SahilExp6	N/A	N/A	N/A
✗	☁	sahiljob	N/A	24 days #1	1.5 sec

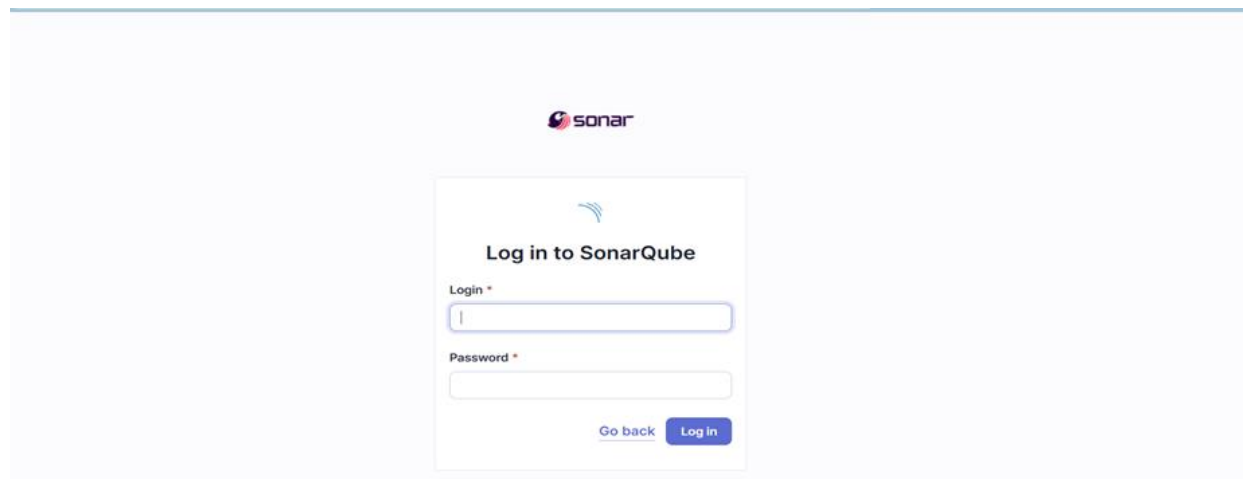
2. Run SonarQube in a Docker container using this command -

***docker run -d --name sonarqube -e SONAR\_ES\_BOOTSTRAP\_CHECKS\_DISABLE=true -p 9000:9000 sonarqube:latest***

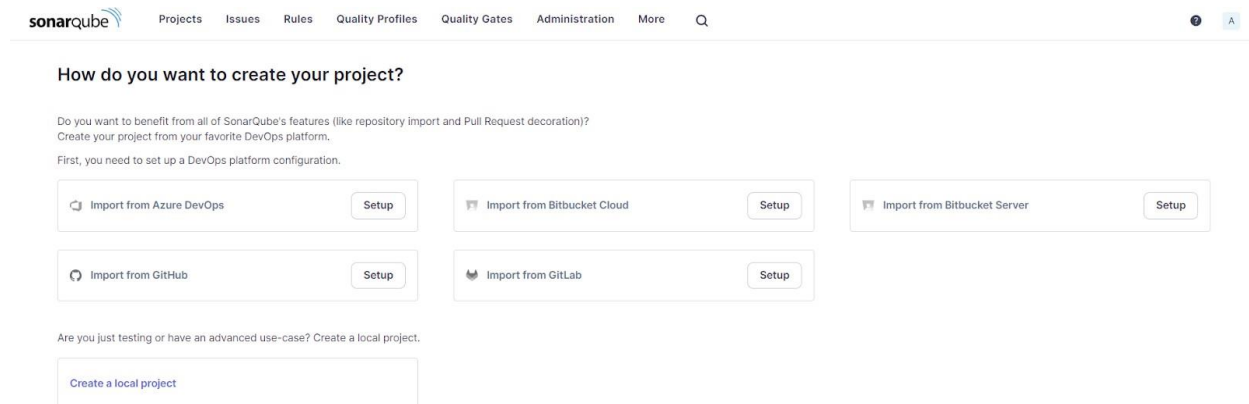
**----- Warning: run below command only once**

```
C:\Windows\System32>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
c1c57f6ace6123c4160745bf57640c75fcbfe70225eabb02b7bf7a40450e8001
```

3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.

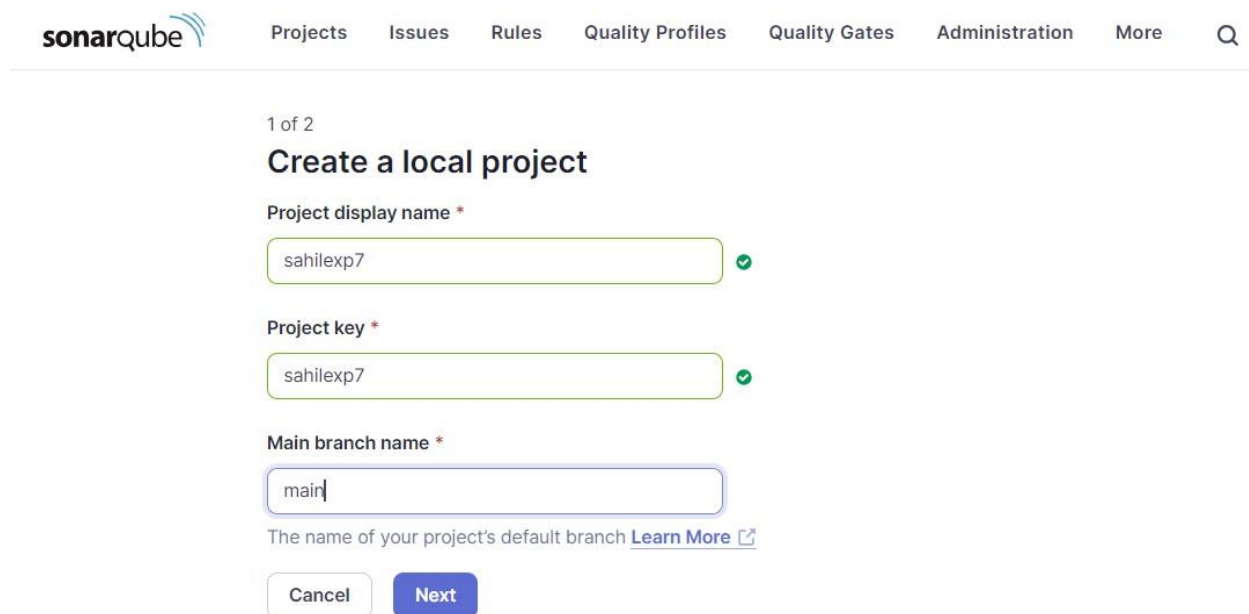


## 4. Login to SonarQube using username admin and password admin.



The image shows the SonarQube project creation wizard. The header includes the SonarQube logo and navigation links: Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search icon. The main heading is "How do you want to create your project?". Below this, a sub-heading asks if the user wants to benefit from all features (like repository import and Pull Request decoration) and suggests creating a project from a favorite DevOps platform. A note states that a DevOps platform configuration is needed first. There are five "Import from" buttons: Azure DevOps, Bitbucket Cloud, Bitbucket Server, GitHub, and GitLab, each with a "Setup" button. At the bottom, there is a link to "Create a local project" for users who are just testing or have an advanced use-case.

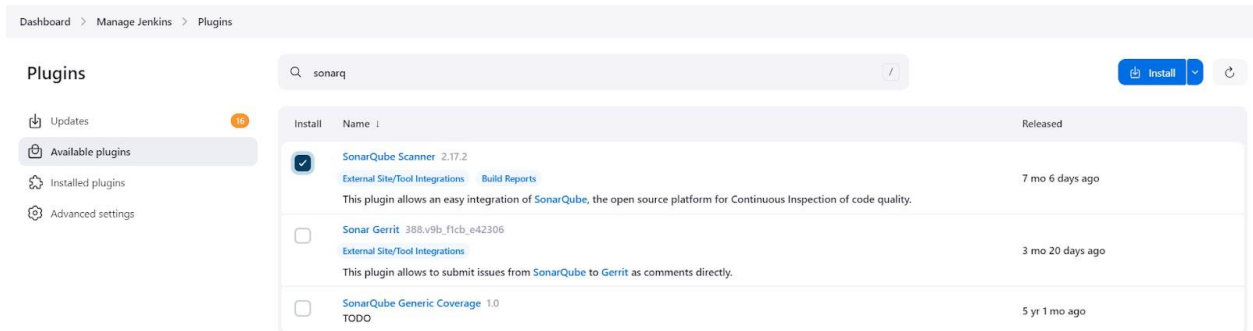
## 5. Create a manual project in SonarQube with the name sonarqube



The image shows the "Create a local project" form in SonarQube. The header includes the SonarQube logo and navigation links: Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search icon. The main heading is "Create a local project". Below this, there are three required fields: "Project display name \*" with the value "sahilexp7", "Project key \*" with the value "sahilexp7", and "Main branch name \*" with the value "main". Each field has a green checkmark indicating it is valid. Below the fields, there is a note: "The name of your project's default branch [Learn More](#)". At the bottom, there are two buttons: "Cancel" and "Next".

Setup the project and come back to Jenkins Dashboard.

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



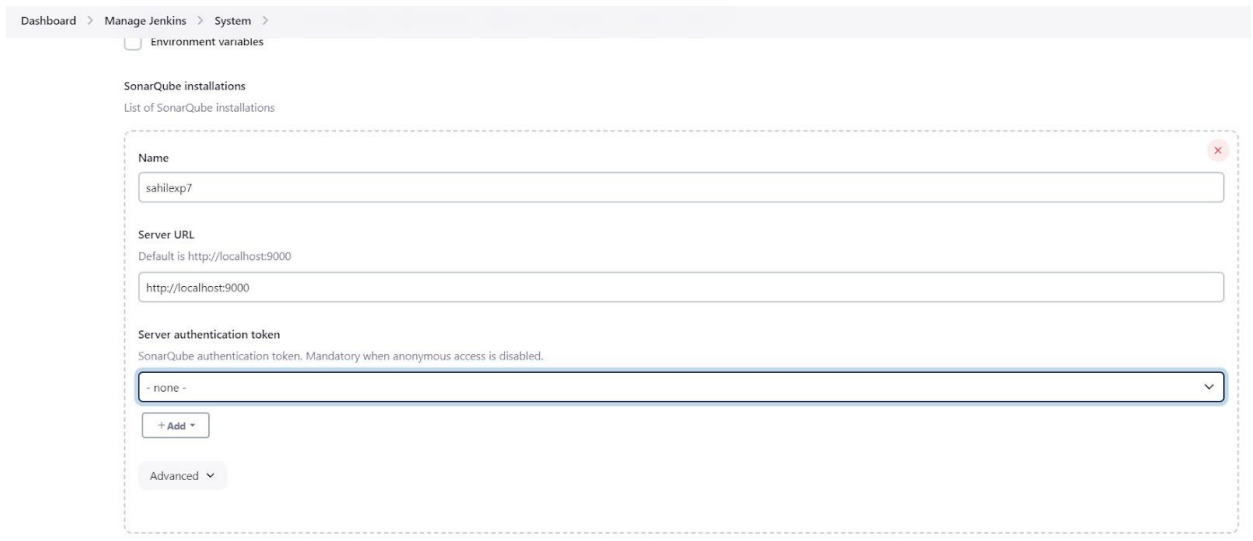
6. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.

Enter the Server Authentication token if needed.

In SonarQube installations: Under **Name** add <project name of sonarqube> for me

**sahilexp7**

In **Server URL** Default is <http://localhost:9000>



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

## Dashboard > Manage Jenkins > Tools

Dashboard > Manage Jenkins > Tools

Add Git ▼

---

Gradle installations

Add Gradle

---

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

---

SonarQube Scanner installations

Add SonarQube Scanner

---

Ant installations

Add Ant

Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.

Dashboard > Manage Jenkins > Tools

Add SonarQube Scanner

☰ SonarQube Scanner

Name

sonarqube\_exp7

☒ Install automatically ?

☰ Install from Maven Central

Version

SonarQube Scanner 6.2.0.4584

Add Installer ▼

Add SonarQube Scanner

---

Ant installations

Save Apply





8. After the configuration, create a New Item in Jenkins, choose a freestyle project.

Dashboard > All > New Item

### New Item

Enter an item name

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.
-  **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 folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

9. Choose this GitHub repository in Source Code Management.

[https://github.com/shazforiot/MSBuild\\_firstproject.git](https://github.com/shazforiot/MSBuild_firstproject.git)

It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

Dashboard > exp7 > Configuration

### Configure

- General
- Source Code Management**
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

#### Source Code Management

☐ None

☒ **Git** ?

Repositories ?

Repository URL ?

Credentials ?

- none -

+ Add

Advanced

Add Repository

Branches to build ?

Branch Specifier (blank for '\*/') ?

Save Apply

10. Under **Select project** → **Configuration** → **Build steps** → **Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Dashboard > exp7 > Configuration

Configure

General

Source Code Management

Build Triggers

**Build Environment**

Build Steps

Post-build Actions

Filter

Execute SonarQube Scanner

Execute Windows batch command

Execute shell

Invoke Ant

Invoke Gradle script

Invoke top-level Maven targets

Run with timeout

Set build status to "pending" on GitHub commit

SonarScanner for MSBuild - Begin Analysis

SonarScanner for MSBuild - End Analysis

Add build step ^

Post-build Actions

Add post-build action v

Save

Apply

Then save

Dashboard ✓ exp7 >

Status

</> Changes

Workspace

Build Now


Configure

Delete Project

SonarQube

Rename

✓ exp7

 SonarQube

Permalinks

- [Last build \(#20\), 4 min 15 sec ago](#)
- [Last stable build \(#20\), 4 min 15 sec ago](#)
- [Last successful build \(#20\), 4 min 15 sec ago](#)
- [Last failed build \(#18\), 6 min 38 sec ago](#)
- [Last unsuccessful build \(#19\), 4 min 55 sec ago](#)
- [Last completed build \(#20\), 4 min 15 sec ago](#)

11. Go to [http://localhost:9000/<user\\_name>/permissions](http://localhost:9000/<user_name>/permissions) and allow Execute Permissions to the Admin user.

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More Q





Administration

Configuration Security Projects System Marketplace

Grant and revoke permissions to make changes at the global level. These permissions include creating quality profiles, executing analysis, and performing global system administration.

All Users Groups

Search for users or groups...

	Administer System ?	Administer ?	Execute Analysis ?	Create ?
 <b>sonar-administrators</b> System administrators	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
 <b>sonar-users</b> Every authenticated user automatically belongs to this group	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects
 <b>Anyone DEPRECATED</b> 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"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input type="checkbox"/> Projects
 <b>Administrator</b> admin	<input checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

4 of 4 shown



***IF CONSOLE OUTPUT FAILED:*****Step 1: Generate a New Authentication Token in SonarQube****1. Login to SonarQube:**

- Open your browser and go to `http://localhost:9000`.
- Log in with your admin credentials (default username is `admin`, and the password is either `admin` or your custom password if it was changed).

**2. Generate a New Token:**

- Click on your **username** in the top-right corner of the SonarQube dashboard.
- Select **My Account** from the dropdown menu.
- Go to the **Security** tab.
- Under **Generate Tokens**, type a name for the token (e.g., "Jenkins-SonarQube").
- Click **Generate**.
- Copy the token and save it securely. You will need it in Jenkins.

**Step 2: Update the Token in Jenkins****1. Go to Jenkins Dashboard:**

- Open Jenkins and log in with your credentials.

**2. Configure the Jenkins Job:**

- Go to the job that is running the SonarQube scanner (`adv_devops_exp7`).
- Click **Configure**.

**3. Update the SonarQube Token:**

- In the SonarQube analysis configuration (either in the pipeline script or under "Build" section, depending on your job type), update the `sonar.login` parameter with the new token.

Dashboard > exp7 > Configuration

### Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

#### Execute SonarQube Scanner

JDK ?  
JDK to be used for this SonarQube analysis

(Inherit From Job)

Path to project properties ?

Analysis properties ?

```
sonar.projectKey=sahilexp7
sonar.host.url=http://localhost:9000
sonar.login=sqa_6a06a6333aecc198878c652a50cfc3f1e9cf82c1
sonar.sources=.
```

Additional arguments ?

JVM Options ?

[Save](#) [Apply](#)

## 12. Run the Jenkins build.

Dashboard > exp7 >

Status

Changes

Workspace

Build Now

**Configure**

Delete Project

SonarQube

Rename

exp7

SonarQube

### Permalinks

- Last build (#20), 8 min 21 sec ago
- Last stable build (#20), 8 min 21 sec ago
- Last successful build (#20), 8 min 21 sec ago
- Last failed build (#18), 10 min ago
- Last unsuccessful build (#19), 9 min 1 sec ago
- Last completed build (#20), 8 min 21 sec ago

Build History trend

Filter...

#20

Sep 23, 2024, 7:49 PM

Check the console Output

## Console output;

Dashboard > exp7 > #20 > Console Output

- Status
- Changes
- Console Output**
- Edit Build Information
- Delete build '#20'
- Timings
- Git Build Data
- Previous Build

### Console Output

```
Started by user Sahil Motiramani
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\exp7
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\exp7\.git # t
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # t
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
```

## 13. Once the build is complete, check project on SonarQube

☆ sahilexp7 / main ✓ ?

Overview Issues Security Hotspots Measures Code Activity

### main

✓

Quality Gate ?  
**Passed**

⚠ The last analysis has warnings. [See details](#)

🔄

### Activity

In this way, we have integrated Jenkins with SonarQube for SAST.

**Conclusion:**

In this project, we integrated Jenkins with SonarQube for automated static application security testing (SAST). We set up SonarQube using Docker, configured Jenkins with the necessary plugins and authentication, and linked it to a GitHub repository. The SonarQube scanner was added as a build step, enabling continuous code analysis for vulnerabilities, code smells, and quality issues, ensuring automated reporting and continuous code quality improvement.