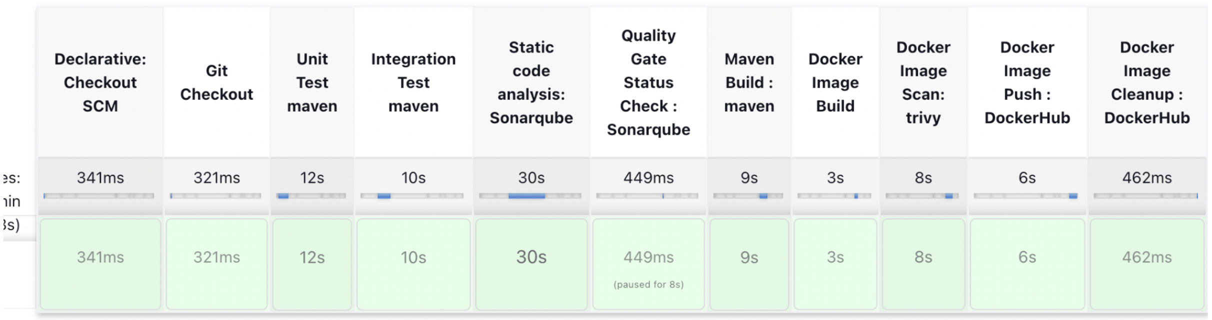


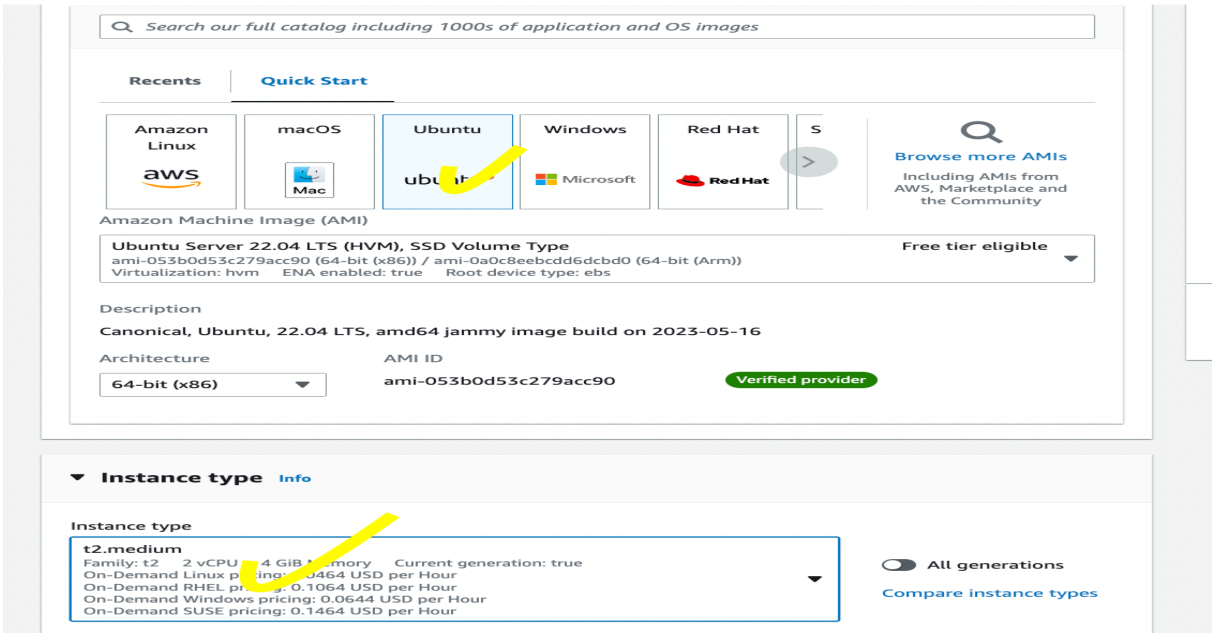
	Declarative: Checkout SCM	Git Checkout	Unit Test maven	Integration Test maven	Static code analysis: Sonarqube	Quality Gate Status Check : Sonarqube	Maven Build : maven	Docker Image Build	Docker Image Scan: trivy	Docker Image Push : DockerHub	Docker Image Cleanup : DockerHub
es: nin 3s)	341ms	321ms	12s	10s	30s	449ms	9s	3s	8s	6s	462ms
	341ms	321ms	12s	10s	30s	449ms (paused for 8s)	9s	3s	8s	6s	462ms

CI PIPELINE SETUP WITH VARIOUS TOOLS

****You need to complete this in one go ****
100 rupees max it will cost
[8 GB –> 30 GB] and t2.micro -> t2.medium [More memory and more cpu]

Step 1 – Create the Ec2 instance in AWS account with these parameters

EC2 type – Ubuntu t2.medium
EBS volume – 30 GB
Region – US-EAST-1


Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
S
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)
Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-053b0d53c279acc90 (64-bit (x86)) / ami-0a0c8eebcdd6dcbd0 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-05-16

Architecture: 64-bit (x86)
AMI ID: ami-053b0d53c279acc90 — Verified provider

Instance type Info

Instance type
t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux pricing: 0.0464 USD per Hour
On-Demand RHEL pricing: 0.1064 USD per Hour
On-Demand Windows pricing: 0.0644 USD per Hour
On-Demand SUSE pricing: 0.1464 USD per Hour

Step 2 – Connect to EC2 and Install all tools in that system as root user

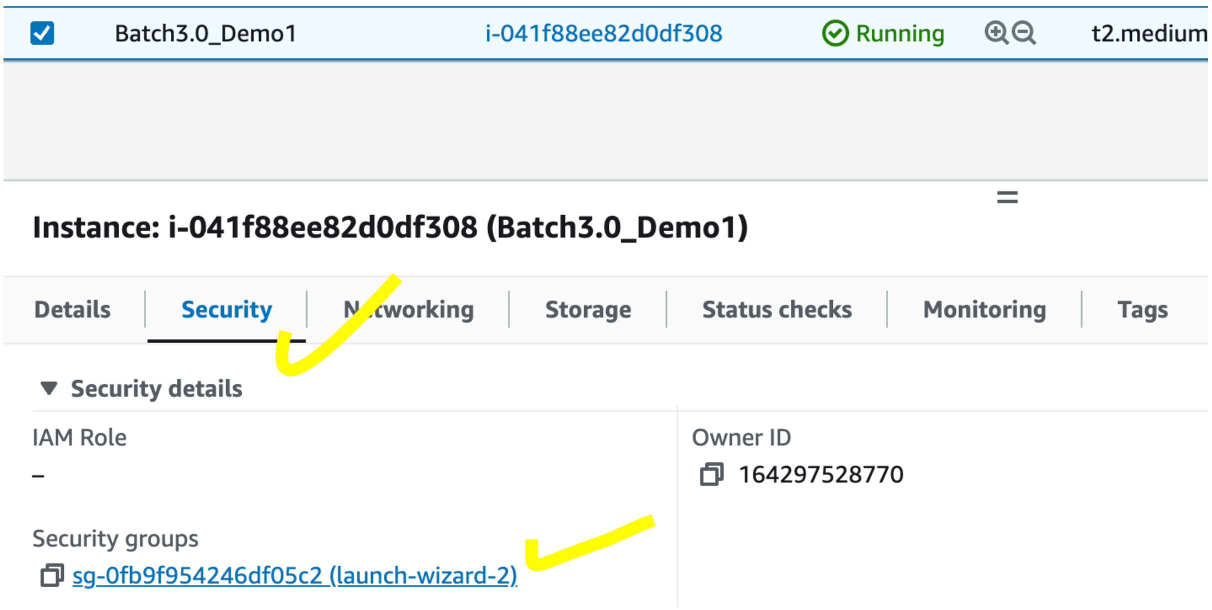
To login as root user - sudo su

Step 3 – Install Jenkins on Ubuntu

Just copy paste the entire commands

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/jenkins.sh

Step 4 – Change the security group of ec2 instance



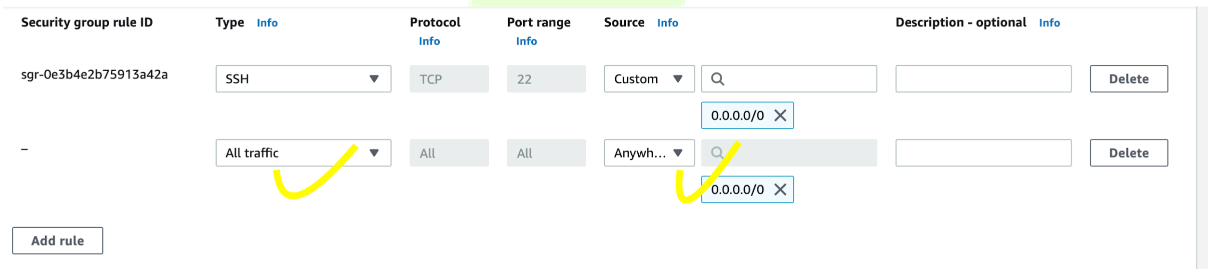
✓ Batch3.0_Demo1	i-041f88ee82d0df308	Running	t2.medium

Instance: i-041f88ee82d0df308 (Batch3.0_Demo1)

▼ Security details

IAM Role	Owner ID
–	164297528770

Security groups
sg-0fb9f954246df05c2 (launch-wizard-2)



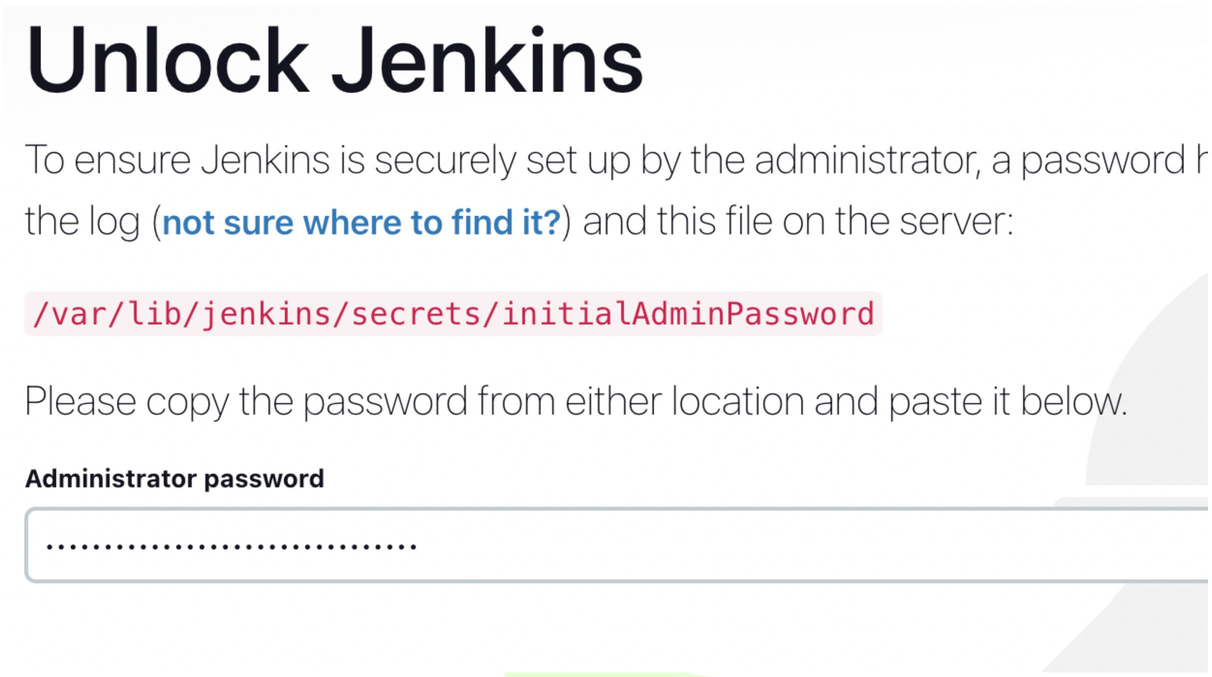
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info
sgr-0e3b4e2b75913a42a	SSH	TCP	22	Custom	0.0.0.0/0	
–	All traffic	All	All	Anywh...	0.0.0.0/0	

Step 5 – Sign Into Jenkins console

http://<EC2_PUBLIC _IP>:8080/

Step 6 – Get the Administrator password by hitting the below command in EC2

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```



Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password h
the log (not sure where to find it?) and this file on the server:

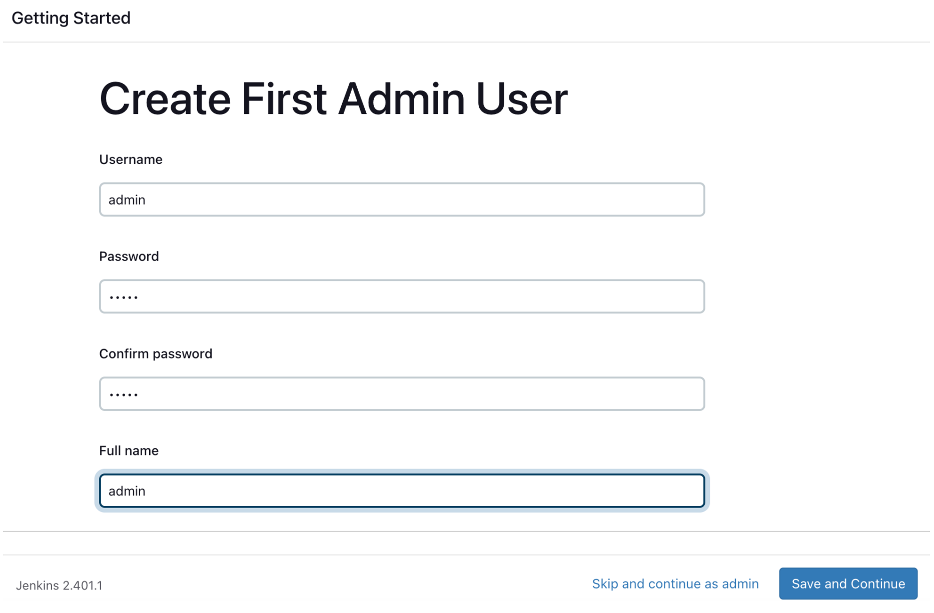
```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password	••••••••••••••••••••••••••••••••

Step 7 – Install all suggested plugins

Step 8 – Create first user



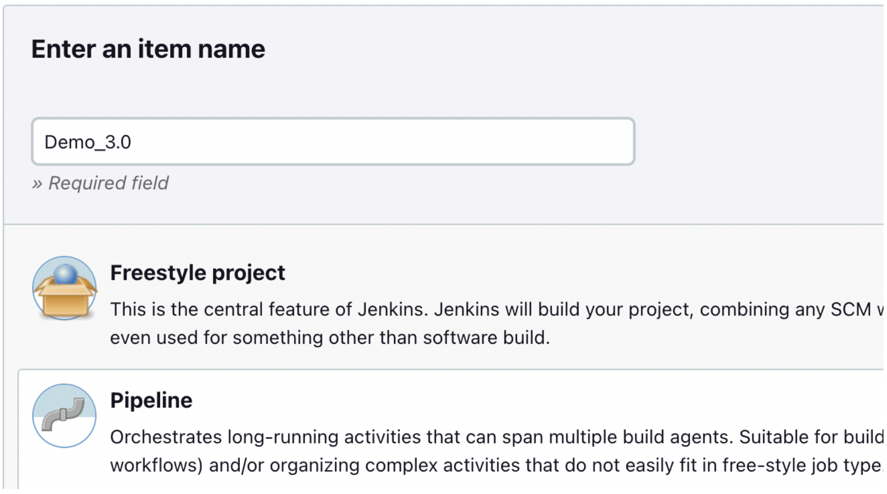
Getting Started

Create First Admin User

Username	admin
Password	•••••
Confirm password	•••••
Full name	admin

Jenkins 2.401.1

Step 9** – Create a pipeline Job


Enter an item name

Demo_3.0

» Required field

Freestyle project

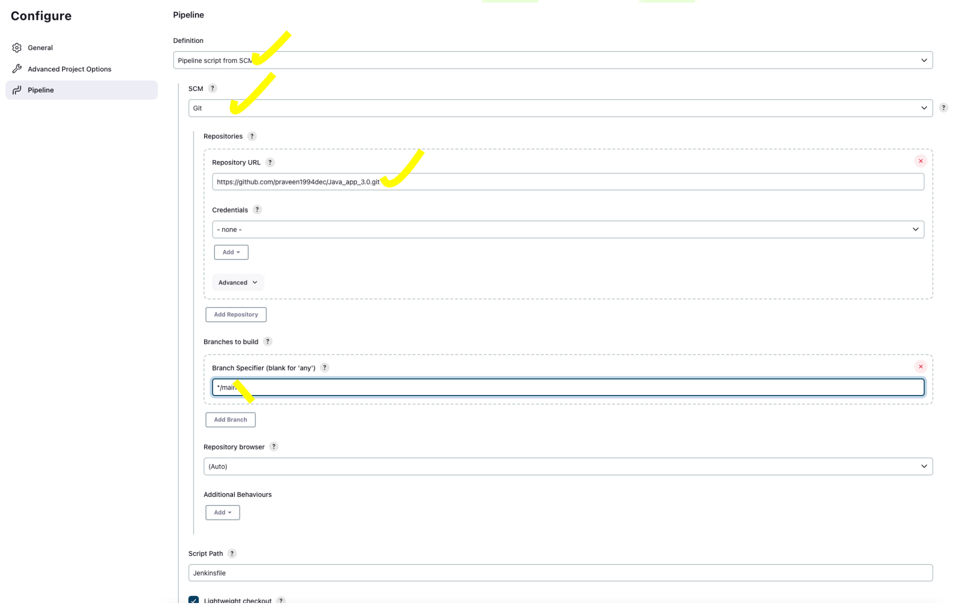
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM w
even used for something other than software build.

Pipeline

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

Step 10 – Add pipeline script as SCM

https://github.com/praveen1994dec/Java_app_3.0.git



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/praveen1994dec/Java_app_3.0.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */main

Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkinsfile

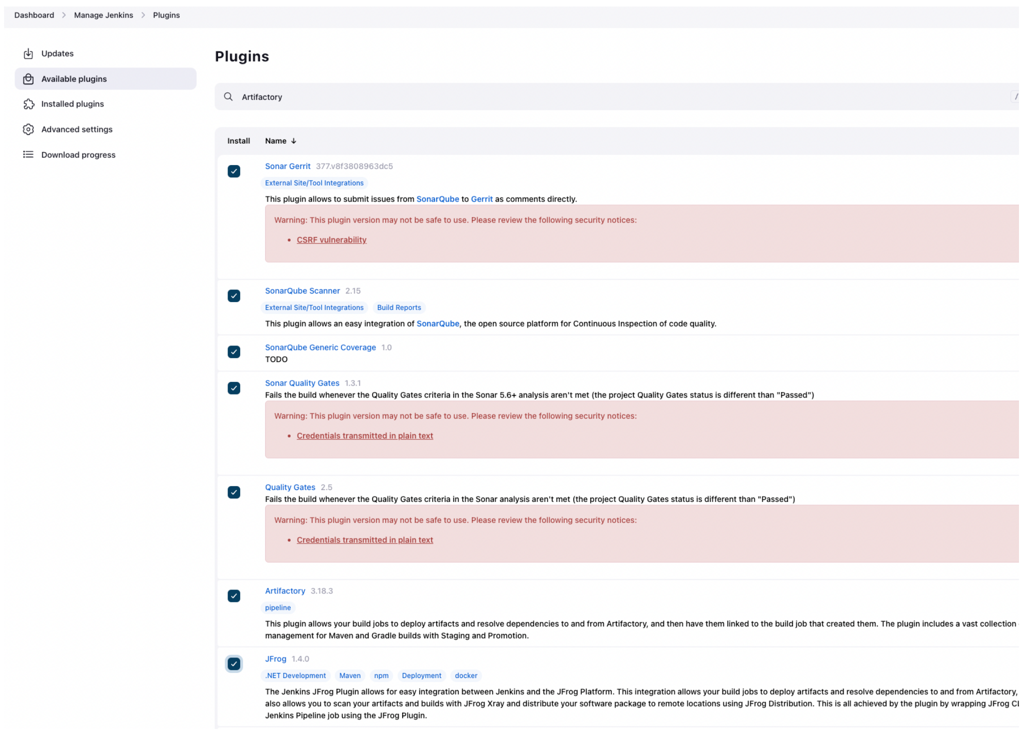
Lightweight checkout: ✓

Step 11 – Add the Plugins

Dashboard -> Manage Jenkins -> Plugins -> Available plugins

Plugins for Sonar/Jfrog –

Sonar Gerrit
SonarQube Scanner
SonarQube Generic Coverage
Sonar Quality Gates
Quality Gates
Artifactory
Jfrog



Dashboard > Manage Jenkins > Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

Plugins

Artifactory

Install	Name ↓
✓	Sonar Gerrit 377.v8f3808963dc5 External Site/Tool Integrations This plugin allows to submit issues from SonarQube to Gerrit as comments directly. Warning: This plugin version may not be safe to use. Please review the following security notices: • CSRF vulnerability
✓	SonarQube Scanner 2.15 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.
✓	SonarQube Generic Coverage 1.0 TODO
✓	Sonar Quality Gates 1.3.1 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") Warning: This plugin version may not be safe to use. Please review the following security notices: • Credentials transmitted in plain text
✓	Quality Gates 2.5 Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed") Warning: This plugin version may not be safe to use. Please review the following security notices: • Credentials transmitted in plain text
✓	Artifactory 3.18.3 pipeline This plugin allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them. The plugin includes a vast collection management for Maven and Gradle builds with Staging and Promotion.
✓	JFrog 1.4.0 .NET Development Maven npm Deployment docker The Jenkins JFrog Plugin allows for easy integration between Jenkins and the JFrog Platform. This integration allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, also allows you to scan your artifacts and builds with JFrog Xray and distribute your software package to remote locations using JFrog Distribution. This is all achieved by the plugin by wrapping JFrog CI Jenkins Pipeline job using the JFrog Plugin.

Step 12 – Setup Docker

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/docker.sh

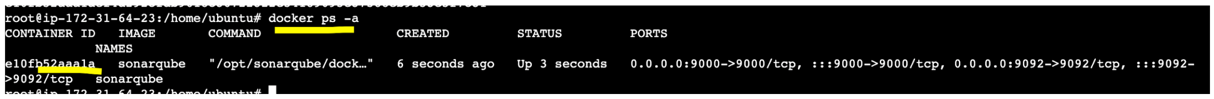
docker -v

Step 13- Install SonarQube

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/sonarqube.sh

Step 13 .1 -> Start docker container if it's not up

docker ps -a [Get the container ID]



```
root@ip-172-31-64-23:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS
          NAMES
e10fb52aaa1a   sonarqube   "/opt/sonarqube/dock…"   6 seconds ago   Up 3 seconds   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 0.0.0.0:9092->9092/tcp, :::9092-
>9092/tcp   sonarqube
root@ip-172-31-64-23:/home/ubuntu#
```

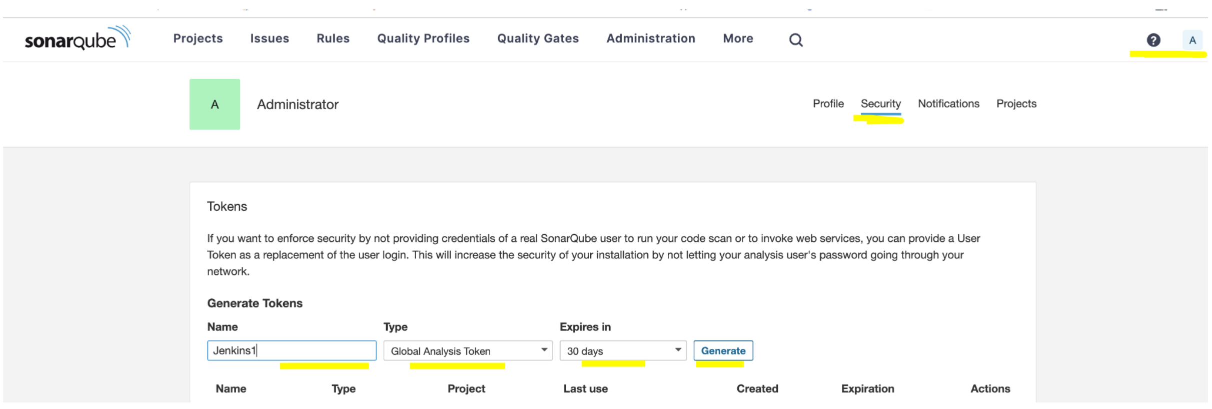
docker start <containerID>

Step 13.2 -> Login into sonar dashboard

Username – admin
Password – admin

Step 13.3 -> Create Sonar token for Jenkins

Sonar Dashboard -> Administration -> My Account -> Security -> Create token -> Save the token to some text file



Tokens

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

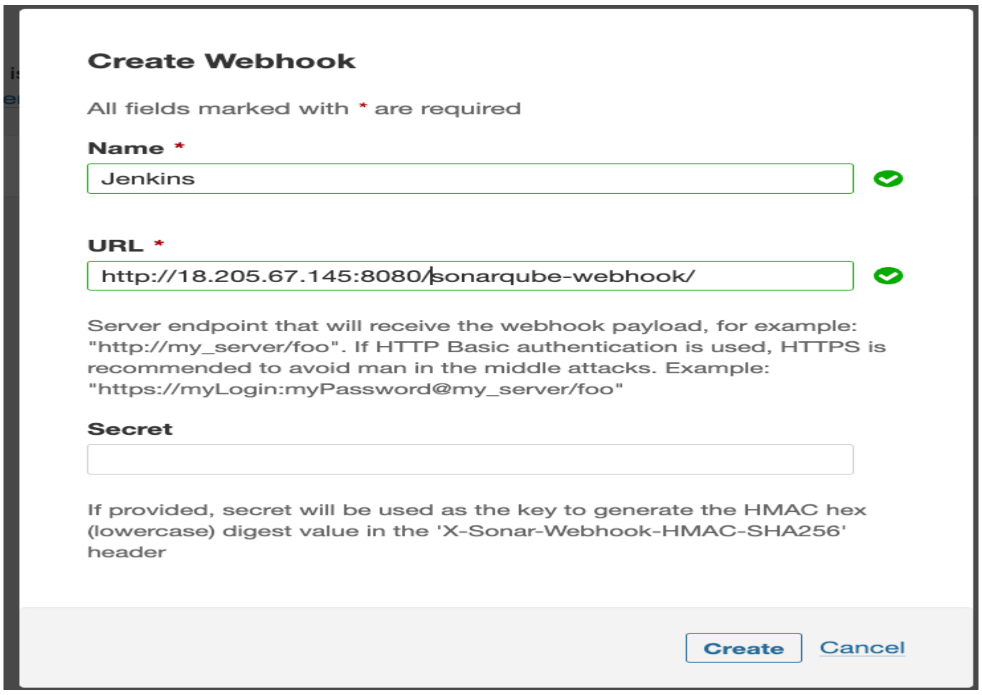
Name	Type	Expires in	
Jenkins1	Global Analysis Token	30 days	Generate

Name	Type	Project	Last use	Created	Expiration	Actions

Step 13.4 -> Integrate Sonar to Jenkins

Sonar Dashboard -> Administration -> Configuration -> webhooks -> Add the below name and url and save

http://<EC2_IP>:8080/sonarqube-webhook/


Create Webhook

All fields marked with * are required

Name *

Jenkins

URL *

http://18.205.67.145:8080/sonarqube-webhook/

Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header

Step 14 – Install Maven

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/Maven.sh

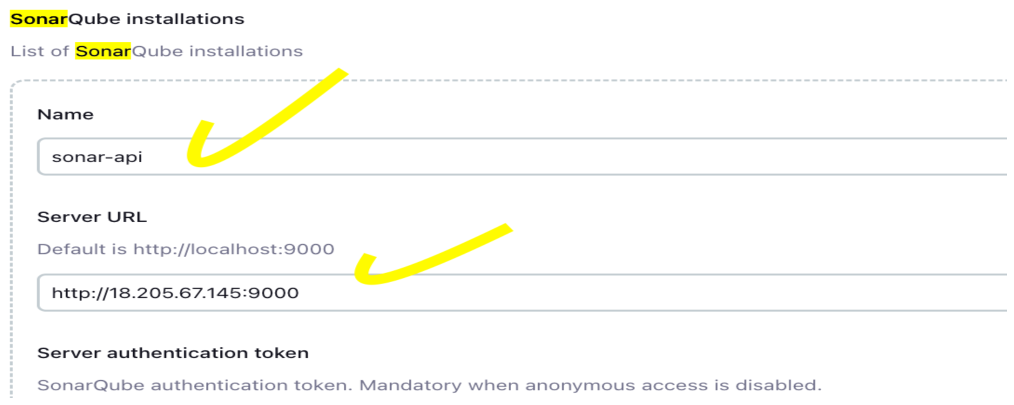
Step 15 – Install TRIVY for docker image scan

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/trivy.sh

Integrate All tools with Jenkins

Jenkins Dashboard -> Manage Jenkins -> configure system

Step 16 – ADD SONARQUBE



SonarQube installations

List of SonarQube installations

Name

sonar-api

Server URL

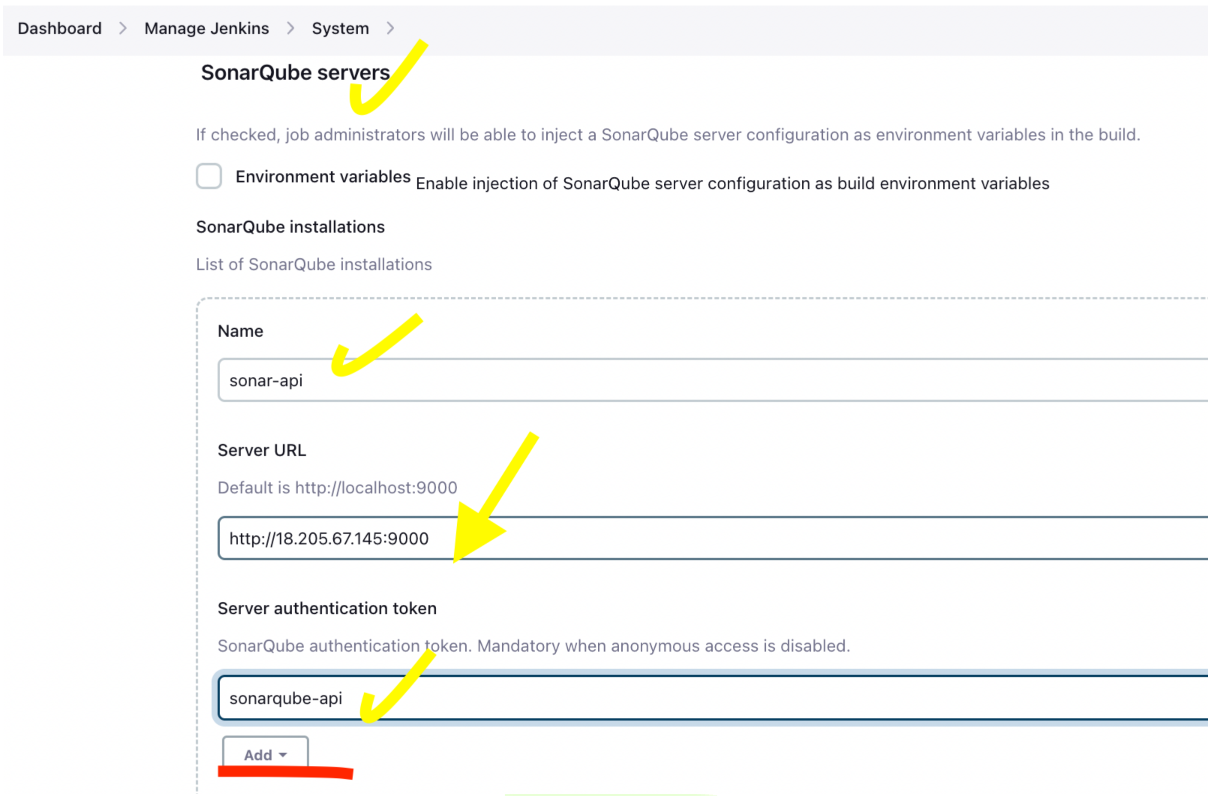
Default is http://localhost:9000

http://18.205.67.145:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Step 16.1 -> Click on sonarqube servers -> add url and name -> Click on add token -> Select Secret text -> Add the sonar token from step13.3 -> Give name of token as sonarqube-api


Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name

sonar-api

Server URL

Default is http://localhost:9000

http://18.205.67.145:9000

Server authentication token

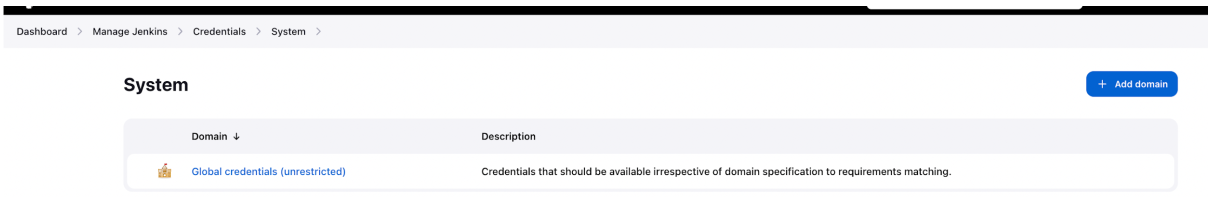
SonarQube authentication token. Mandatory when anonymous access is disabled.

sonarqube-api

Add

Step 17 - Add the docker HUB credentials ID

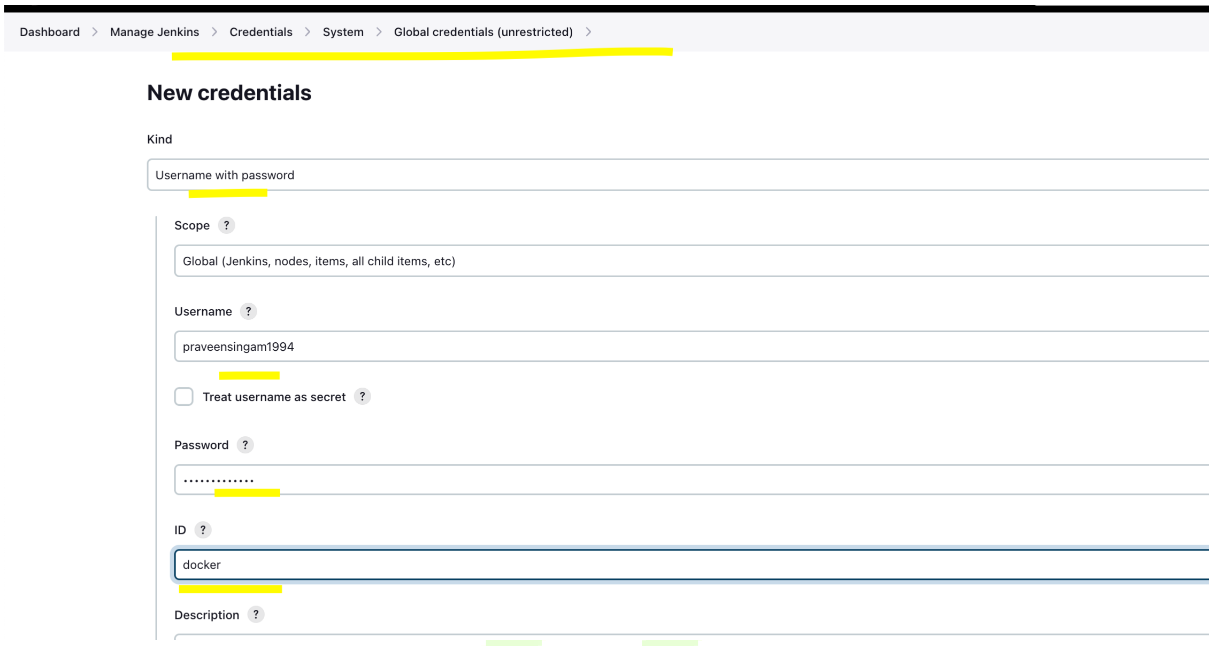
Jenkins dashboard -> Manage Jenkins -> Credentials -> System -> click on global credentials


Dashboard > Manage Jenkins > Credentials > System >

System

Domain ↓	Description
Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

ADD the docker hub credentials with name as docker



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: praveensingam1994

Treat username as secret

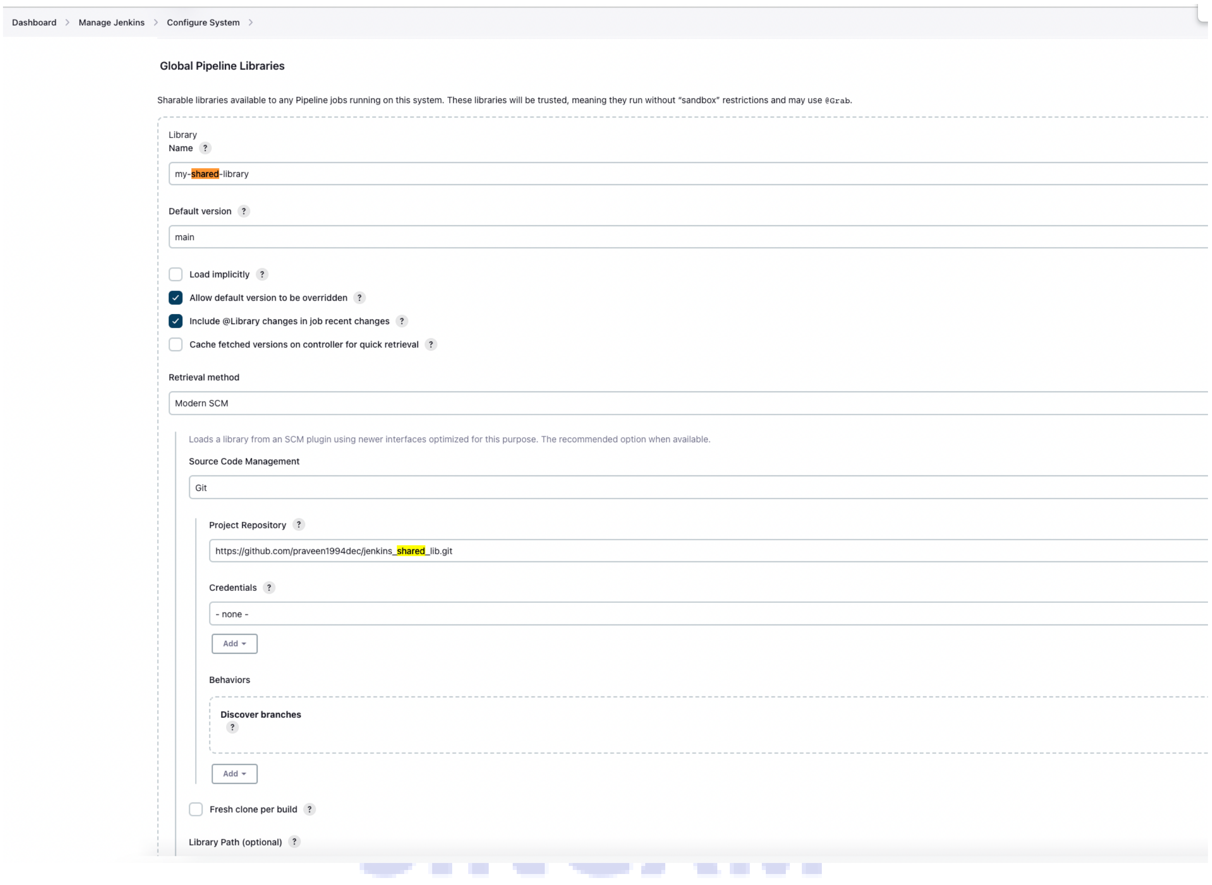
Password: ••••••••••••••

ID: docker

Description

Step 18 – Add the Jenkins Shared library

Go to Manage Jenkins -> Configure system -> Global pipeline library -> Add below data
Name - my-shared-library
Default version – main
Git - https://github.com/praveen1994dec/jenkins_shared_lib.git


Dashboard > Manage Jenkins > Configure System >

Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use `@Grab`.

Library
Name ?

my-shared-library

Default version ?

main

☐ Load implicitly ?
☑ Allow default version to be overridden ?
☑ Include @Library changes in job recent changes ?
☐ Cache fetched versions on controller for quick retrieval ?

Retrieval method

Modern SCM

Loads a library from an SCM plugin using newer interfaces optimized for this purpose. The recommended option when available.

Source Code Management

Git

Project Repository ?

https://github.com/praveen1994dec/jenkins_shared_lib.git

Credentials ?

- none -

Add ▾

Behaviors

Discover branches ?

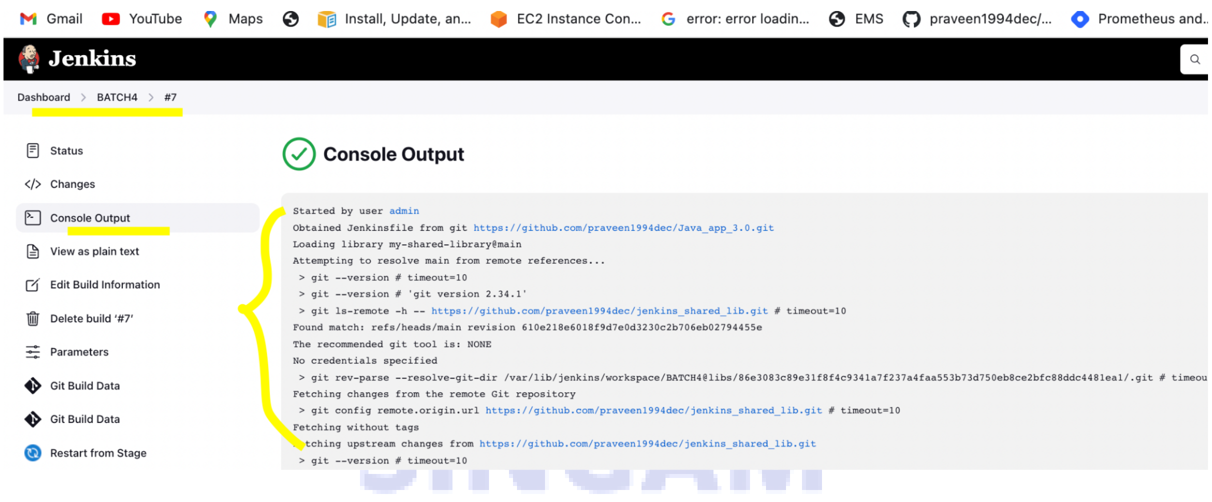
Add ▾

☐ Fresh clone per build ?

Library Path (optional) ?

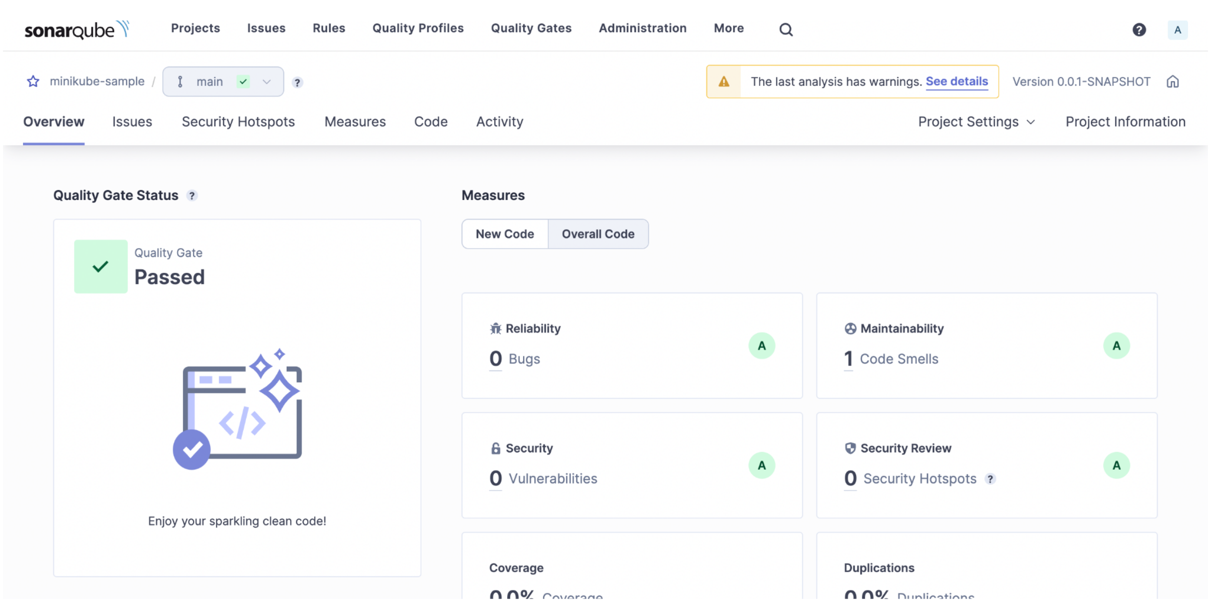
Step 19 - Once pipeline is Run Check

- The Jenkins logs
- The Trivy scan vulnerabilities
- The sonarqube dashboard for report



Console Output

```
Started by user admin
Obtained Jenkinsfile from git https://github.com/praveen1994dec/Java_app_3.0.git
Loading library my-shared-library@main
Attempting to resolve main from remote references...
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git ls-remote -h -- https://github.com/praveen1994dec/jenkins_shared_lib.git # timeout=10
Found match: refs/heads/main revision 610e218e6018f9d7e0d3230c2b706eb02794455e
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/BATCH4@libs/86e3083c89e31f8f4c9341a7f237a4faa553b73d750eb8ce2bfc88ddc4481ea1/.git # timeou
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/praveen1994dec/jenkins_shared_lib.git # timeout=10
Fetching without tags
 tching upstream changes from https://github.com/praveen1994dec/jenkins_shared_lib.git
 > git --version # timeout=10
```



The last analysis has warnings. See details Version 0.0.1-SNAPSHOT

Quality Gate Status

Quality Gate
Passed

Enjoy your sparkling clean code!

Measures

Reliability A
0 Bugs

Maintainability A
1 Code Smells

Security A
0 Vulnerabilities

Security Review A
0 Security Hotspots

Coverage
0.0% Coverage

Duplications
0.0% Duplications