

SonarQube Integration with Jenkins server

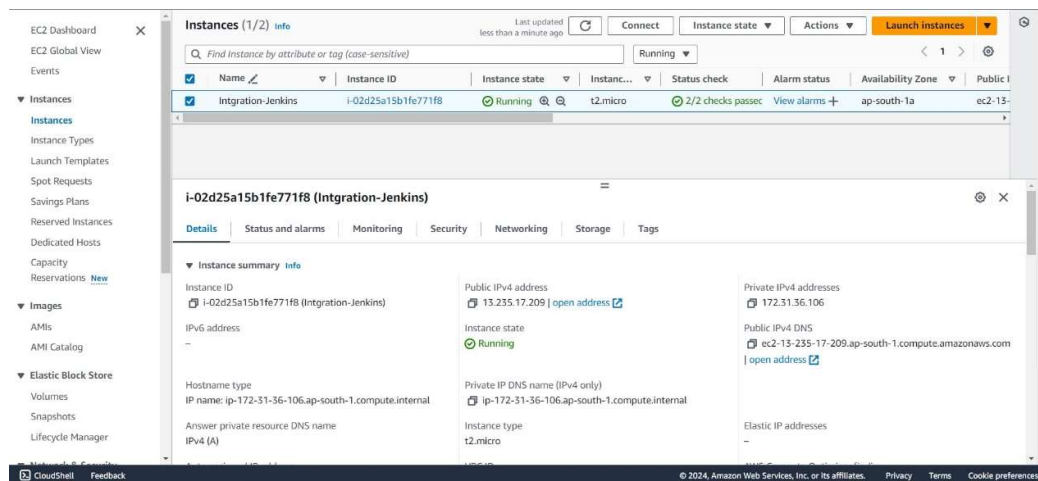
Setup and Configuration:

Prerequisites

Jenkins Installation on AWS EC2:

Step 1: Create and configure an EC2 Instance for Jenkins

1. **Create an EC2 instance** using Ubuntu 20.04 or later (Ubuntu Linux AMI). Select a t2.micro instance type for basic testing.



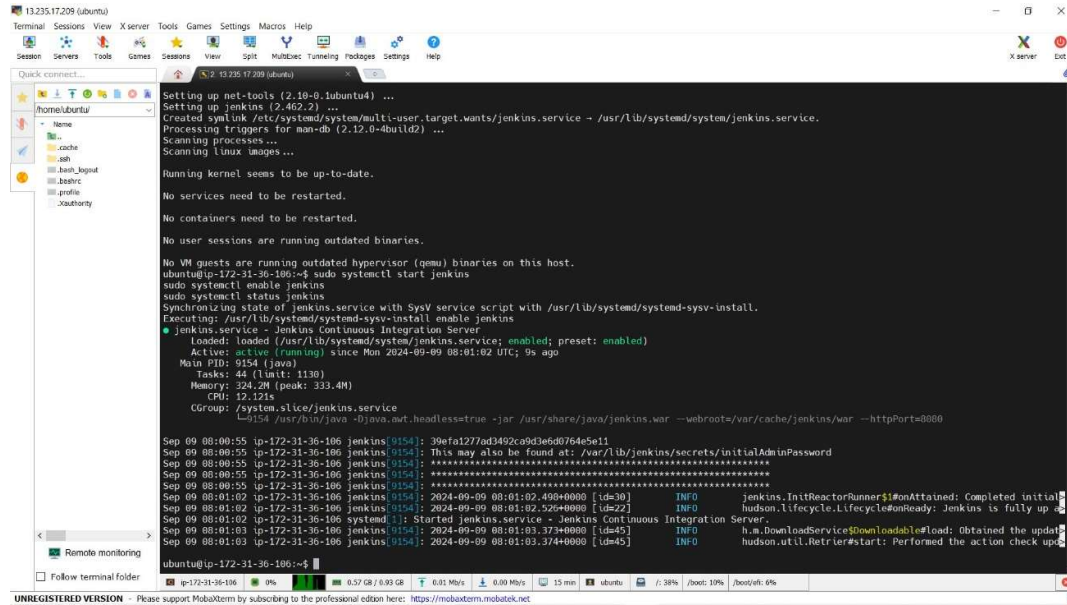
2. **Connect to your EC2 instance** using SSH or a terminal application.
3. **Update all packages:**
 - `sudo apt update -y`
4. **Install Java (Jenkins requires Java to run):**
 - `sudo apt install openjdk-11-jdk -y`

Step 2: Install Jenkins Using apt

1. **Add Jenkins to your apt repository:**
 - `curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \`
`/usr/share/keyrings/jenkins-keyring.asc > /dev/null`
 - `echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]`
`https://pkg.jenkins.io/debian-stable binary/ | sudo tee \`
`/etc/apt/sources.list.d/jenkins.list > /dev/null`
2. **Install Jenkins:**
 - `sudo apt-get update`
 - `sudo apt-get install jenkins -y`

3. Start and enable Jenkins service:

- `sudo systemctl start jenkins`
- `sudo systemctl enable jenkins`
- `sudo systemctl status Jenkins`



```
Setting up net-tools (2.10-0.1ubuntu4) ...
Setting up jenkins (2.462.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes ...
Scanning linux images ...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

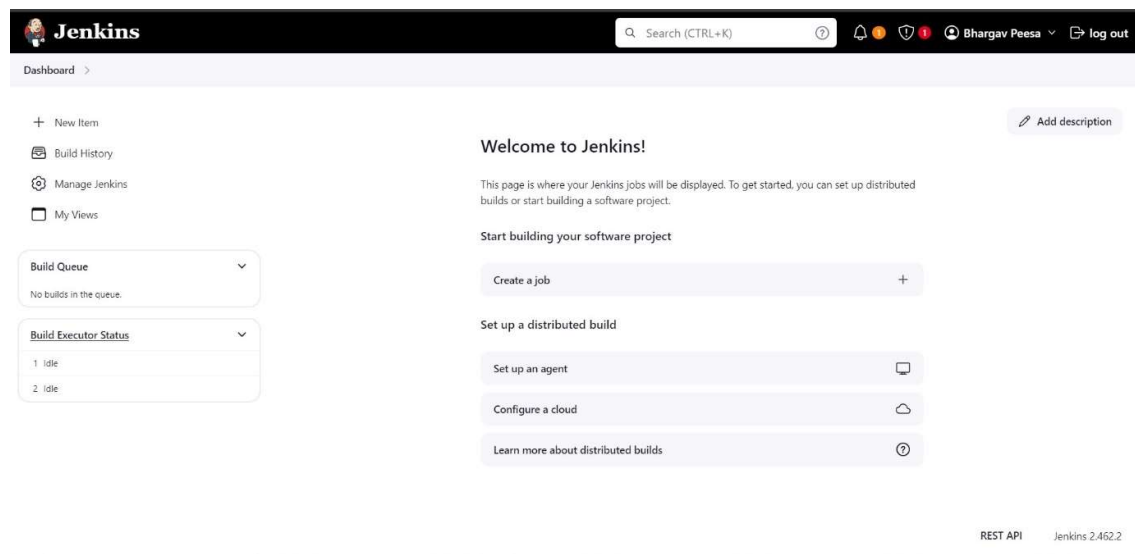
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-36-106:~$ sudo systemctl start jenkins
sudo systemctl enable jenkins
sudo systemctl status jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Mon 2024-09-09 08:01:02 UTC; 9s ago
     Main PID: 9154 (java)
        Tasks: 44 (limit: 1130)
      Memory: 324.2M (peak: 333.4M)
         CPU: 12.121s
    CGroup: /system.slice/jenkins.service
            └─9154 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Sep 09 08:00:55 ip-172-31-36-106 jenkins[9154]: 30efaf127ad3492ca9d36dd0764e5e11
Sep 09 08:00:55 ip-172-31-36-106 jenkins[9154]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 09 08:00:55 ip-172-31-36-106 jenkins[9154]: *****
Sep 09 08:00:55 ip-172-31-36-106 jenkins[9154]: *****
Sep 09 08:00:55 ip-172-31-36-106 jenkins[9154]: *****
Sep 09 08:01:02 ip-172-31-36-106 jenkins[9154]: 2024-09-09 08:01:02.009+0000 [id=20] INFO jenkins.InitReactorRunner$1:onAttained: Completed initial
Sep 09 08:01:02 ip-172-31-36-106 jenkins[9154]: 2024-09-09 08:01:02.525+0000 [id=22] INFO hudson.lifecycle.Lifecycle$onReady: Jenkins is fully up a
Sep 09 08:01:02 ip-172-31-36-106 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 09 08:01:03 ip-172-31-36-106 jenkins[9154]: 2024-09-09 08:01:03.373+0000 [id=5] INFO hudson.DownloadService$Downloadable:load: Obtained the update
Sep 09 08:01:03 ip-172-31-36-106 jenkins[9154]: 2024-09-09 08:01:03.374+0000 [id=5] INFO hudson.util.Retrier$start: Performed the action check up
ubuntu@ip-172-31-36-106:~$
```

4. Get the initial administration password:

- `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
- **Open Jenkins in a browser:**
Access Jenkins using the public IP of your EC2 instance with port 8080 (`http://<public_IP>:8080/`).
Ensure you have added an inbound rule for port 8080 in your EC2 security group.



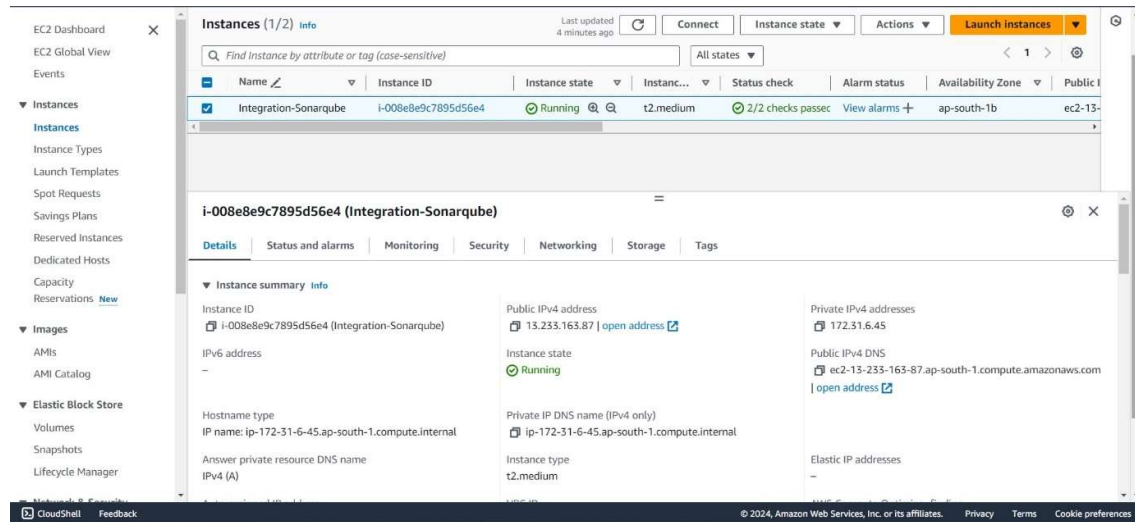
Step 3: Install Plugins in Jenkins

- After accessing Jenkins, provide the initial admin password.
- Select "Install Suggested Plugins" to install common plugins.

SonarQube Installation on AWS EC2:

Step 1: Create and configure an EC2 Instance for SonarQube

1. **Create an EC2 instance** with at least 4 GB of RAM (e.g., t2.medium) to meet SonarQube's minimum requirements.



2. **Connect to your EC2 instance** using SSH or a terminal application.
3. **Switch to the root user:**
 - `sudo -i`
4. **Update all packages:**
 - `sudo apt update -y`
5. **Install wget package:**
 - `sudo apt install wget -y`
6. **Install Java (SonarQube requires Java to run):**
 - `sudo apt install openjdk-11-jdk -y`
 - `java --version`

Step 2: Install SonarQube

1. **Change to the /opt directory:**
 - `cd /opt`
2. **Download the SonarQube zip file:**
 - `wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.4.0.35506.zip`

3. Unzip the SonarQube package:

- `sudo apt install unzip -y`
- `unzip sonarqube-8.4.0.35506.zip`

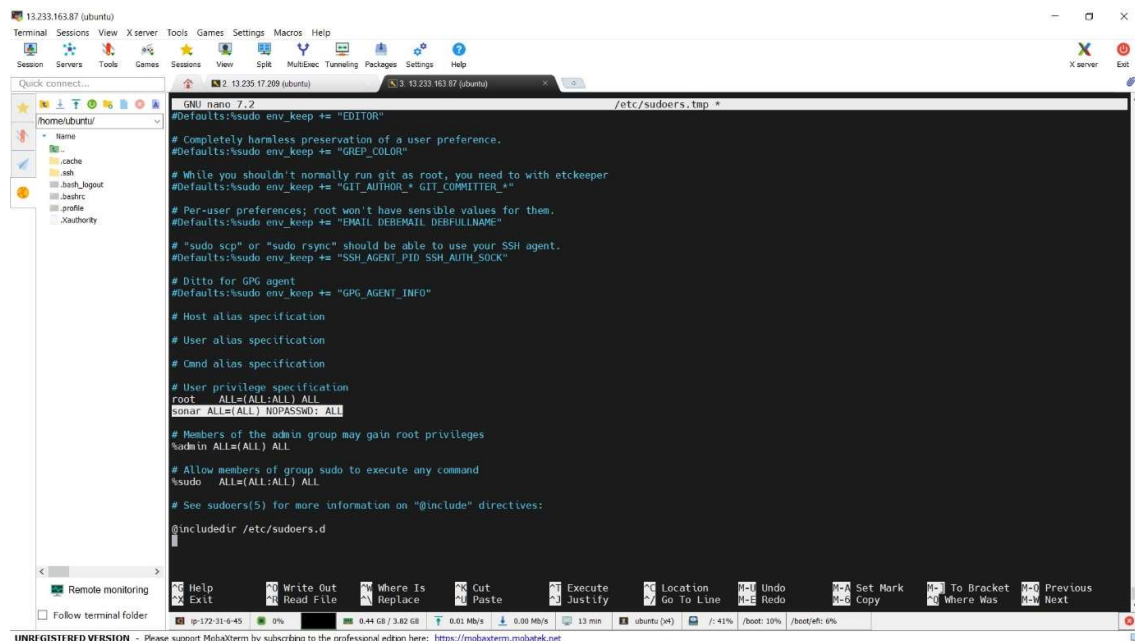
Step 3: Configure SonarQube User and Permissions

1. Create a new user called sonar:

- `sudo useradd sonar`

2. Modify sudoers file to give sonar user necessary permissions:

- `sudo visudo`
- Add the following line:
- `sonar ALL=(ALL) NOPASSWD: ALL`



```
GNU nano 2.2 /etc/sudoers.tmp
#Defaults:sudo env_keep += "EDITOR"
# Defaults:sudo env_keep += "EDITOR"
# Completely harmless preservation of a user preference.
#Defaults:sudo env_keep += "GREP_COLOR"
# While you shouldn't normally run git as root, you need to with etc/keeper
#Defaults:sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_*"
# Per-user preferences; root won't have sensible values for them.
#Defaults:sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"
# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"
# Ditto for GPG agent
#Defaults:sudo env_keep += "GPG_AGENT_INFO"
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL) ALL
sonar    ALL=(ALL) NOPASSWD: ALL
# Members of the admin group may gain root privileges
%admin    ALL=(ALL) ALL
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL) ALL
# See sudoers(5) for more information on "@include" directives:
@include /etc/sudoers.d
```

3. Change ownership of the SonarQube directory:

- `sudo chown -R sonar:sonar /opt/sonarqube-8.4.0.35506`

4. Change file permissions:

- `sudo chmod -R 775 /opt/sonarqube-8.4.0.35506`

5. Switch to the sonar user:

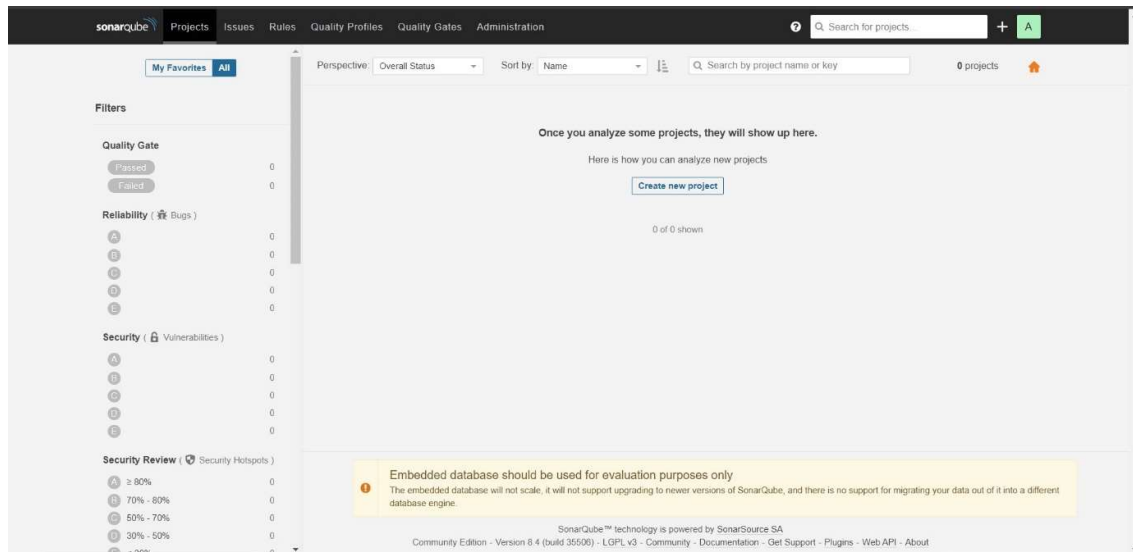
- `sudo su - sonar`

6. Navigate to the SonarQube bin directory and start SonarQube:

- `cd /opt/sonarqube-8.4.0.35506/bin/linux-x86-64`
- `./sonar.sh start`

7. Access SonarQube:

Open the browser and enter `http://<EC2-Public-IP>:9000/`.
Make sure port 9000 is enabled in your EC2 security group.



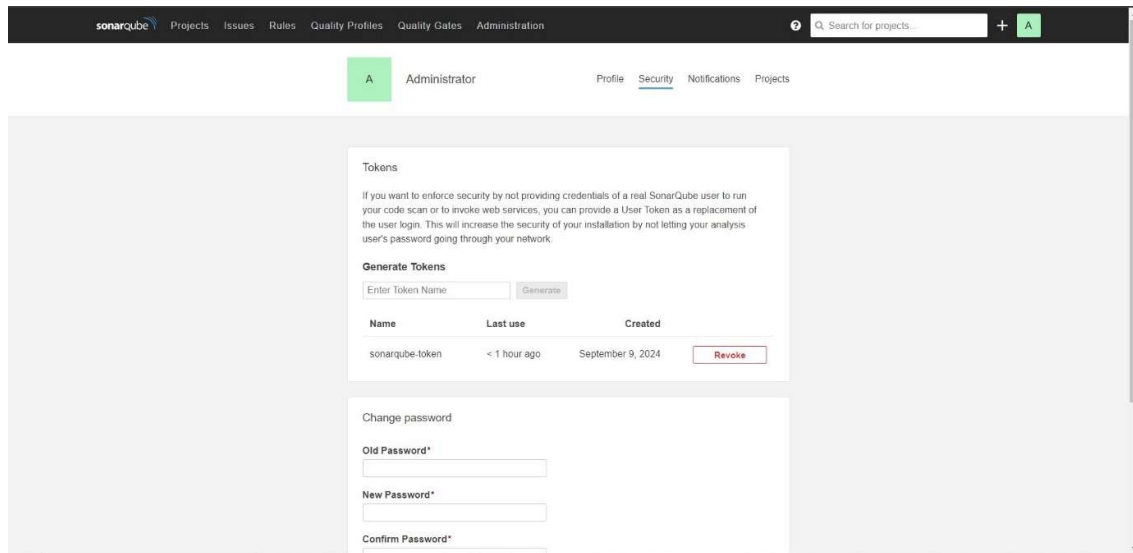
Integration Steps

1. Install SonarQube Scanner Plugin in Jenkins:

- Go to Manage Jenkins -> Manage Plugins.
- Click on Available tab, search for "SonarQube Scanner", and install it.

2. Configure SonarQube Server in Jenkins:

- Go to Manage Jenkins -> Configure System.
- Scroll down to SonarQube servers and add a new server.
- Fill in the required details:
 - **Name:** Sonar -server -8.4
 - **Server URL:** `http://<SonarQube-EC2-Public-IP>:9000`
 - **Server Authentication Token:** (Generate this token in SonarQube under "My Account" -> "Security" -> "Generate Tokens").
 - Back in Jenkins, click on Add next to the Server Authentication Token.
 - Choose Jenkins credential provider and select Secret text as the credential type.
 - Enter the token generated in SonarQube in the Secret field and provide an ID and description.
 - Click Add to save the credentials.



3. Configure SonarQube Scanner in Jenkins:

- Go to Manage Jenkins -> Global Tool Configuration.
- Scroll down to SonarQube Scanner and click Add SonarQube Scanner.
- Enter details like Name: Sonar -Scanner-4.7 and select the installed version.

4. Install Apache Maven on Jenkins Server:

Connect to the Jenkins EC2 instance and execute:

- `sudo su`
- `cd /opt`
- `wget https://dlcdn.apache.org/maven/maven-3/3.8.8/binaries/apache-maven-3.8.8-bin.tar.gz`
- `tar -xvf apache-maven-3.8.8-bin.tar.gz`

5. Create Jenkins Pipeline for Code Analysis:

Pipeline Script Example:

```
pipeline {
    agent any

    environment {
        PATH = "$PATH:/opt/apache-maven-3.8.8/bin" // Ensure Maven is in the PATH
    }

    stages {
        stage("Get Code") {
            steps {
                git " https://github.com/bhargavdevopsaws/SonarQube-Integration-with-Jenkins-server.git"
            }
        }
    }
}
```

```

}

stage("Build") {

    steps {

        sh "mvn clean package"

    }

}

stage("SonarQube Analysis") {

    steps {

        withSonarQubeEnv('Sonar -server -8.4') {

            sh "mvn sonar:sonar"

        }

    }

}

}

```

6. Run the Jenkins Job:

- Go to Jenkins Dashboard.
- Click on New Item, give it a name, select Pipeline, and click OK.
- Paste the pipeline script above in the pipeline configuration.
- Apply and save the configuration.
- Click on Build Now to run the job.

The screenshot shows the Jenkins Dashboard with the 'Console Output' tab selected for a pipeline named 'Sonarqube integration with Jenkins'. The console output displays the following information:

```

Started by user bhargav peesa
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Sonarqube integration with Jenkins
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Get Code)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Sonarqube integration with Jenkins/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/bhargavdevopsaws/SONARQUBE-INTEGRATION-WITH-JENKINS.git # timeout=10
Fetching upstream changes from https://github.com/bhargavdevopsaws/SONARQUBE-INTEGRATION-WITH-JENKINS.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/bhargavdevopsaws/SONARQUBE-INTEGRATION-WITH-JENKINS.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out revision 108c9c10fbb079103e6f0ea2967eb10e3e6db (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 108c9c10fbb079103e6f0ea2967eb10e3e6db # timeout=10

```

```
Dashboard > All > Sonarqube integration with Jenkins > #2

[INFO] 16:44:47.276 Sensor Java CPD Block Indexer
[INFO] 16:44:47.327 Sensor Java CPD Block Indexer (done) | time=51ms
[INFO] 16:44:47.333 SCH Publisher SCH provider for this project is: git
[INFO] 16:44:47.336 SCH Publisher 9 source files to be analyzed
[INFO] 16:44:48.321 SCH Publisher 9/9 source files have been analyzed (done) | time=980ms
[INFO] 16:44:48.374 CPD Executor 3 files had no CPD blocks
[INFO] 16:44:48.378 CPD Executor Calculating CPD for 4 files
[INFO] 16:44:48.397 CPD Executor CPD calculation finished (done) | time=19ms
[INFO] 16:44:48.626 Analysis report generated in 225ms, dir size=102 KB
[INFO] 16:44:48.831 Analysis report compressed in 188ms, zip size=27 KB
[INFO] 16:44:49.283 Analysis report uploaded in 446ms
[INFO] 16:44:49.293 ANALYSIS SUCCESSFUL, you can browse http://13.233.163.87:9000/dashboard?id=com.spring_rest.app%3ASpringRestIntegrationApp
[INFO] 16:44:49.293 Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] 16:44:49.293 More about the report processing at http://13.233.163.87:9000/api/cv/task?id=AZHCeHRS098ayFDoyls
[INFO] 16:44:49.323 Analysis total time: 13.842 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 17.845 s
[INFO] Finished at: 2024-09-09T16:44:49Z
[INFO] -----
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withinV
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Jenkins

Search (CTRL+K)

Bhargav Peesa

log out

Dashboard > All > Sonarqube integration with Jenkins >

Status

Changes

Build Now

Configure

Delete Pipeline

SonarQube

Stages

Rename

Pipeline Syntax

Sonarqube integration with Jenkins

Add description

SonarQube Quality Gate

SpringRestIntegrationApp Maven Webapp **Passed**

server-side processing: **Success**

Permalinks

- Last build (#2), 5 min 52 sec ago
- Last stable build (#2), 5 min 52 sec ago
- Last successful build (#2), 5 min 52 sec ago
- Last unsuccessful build (#1), 8 hr 5 min ago
- Last completed build (#2), 5 min 52 sec ago

Build History

trend

Filter...

#2

Sep 9, 2024, 4:44 PM

#1

Sep 9, 2024, 9:44 AM

sonarqube

Projects

Issues

Rules

Quality Profiles

Quality Gates

Administration

Search for projects

September 9, 2024 at 10:14 PM · Version 0.0.1-SNAPSHOT

SpringRestIntegrationApp Maven Webapp

master

Overview

Issues

Security Hotspots

Measures

Code

Activity

Project Settings

Project Information

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code

Overall Code

4 Bugs

0 Vulnerabilities

0 Security Hotspots

1h 1min Debt

9 Code Smells

Reliability **C**

Security **A**

Security Review **A**

Maintainability **A**

0.0%

0.0%

0