Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

AY: 2025-26

Semester: V

Subject: DevOps Laboratory (DJS23OLOE501)

Experiment 4b

(Continuous Integration with Jenkins)

Aim: To Secure and Analyze CI/CD Pipelines Using Jenkins and SonarQube.

Theory:

Modern software delivery relies heavily on **Continuous Integration and Continuous Deployment (CI/CD)** pipelines. While CI/CD automates building, testing, and deploying software, ensuring **code quality and security** in these pipelines is equally important.

- **Security in CI/CD:** Prevents vulnerabilities from entering production.
- **Code Analysis in CI/CD:** Ensures the code is reliable, maintainable, and follows best practices.

SonarQube Overview

- **SonarQube** is an open-source platform for **continuous inspection of code quality**.
- It performs **Static Application Security Testing (SAST)** by analyzing source code for:
  - Bugs
  - Code smells (bad practices)
  - Security vulnerabilities
  - Test coverage
- Supports multiple languages: Java, Python, C/C++, JavaScript, etc.
- Generates **quality gates**: thresholds that determine whether the code can be promoted to the next pipeline stage.

Securing CI/CD Pipelines

Security in CI/CD pipelines involves:

- **Source Code Protection:** Using access control, branch policies, and secure SCM connections (SSH/HTTPS).
- **Dependency Management:** Scanning third-party libraries for vulnerabilities.
- **Secrets Management:** Avoiding hard-coded credentials, using tools like Jenkins Credentials Manager or Vault.

- **Automated Security Scans:** Integrating tools like SonarQube, OWASP Dependency-Check, and Trivy.
- **Quality Gates:** Blocking deployments if critical vulnerabilities are detected.

Lab experiment to be performed in this session:

Set up a Jenkins CI/CD pipeline that integrates with SonarQube for code quality analysis. Your pipeline should:

- Checkout the source code from a Git repository.
- Build the project using any suitable build tool.
- Run tests to validate the code.
- Analyze the code using SonarQube Scanner.
- Verify results using Quality Gates in SonarQube.

Attach screenshots of every stage of the pipeline execution in Jenkins (Checkout, Build, Test, SonarQube Analysis, Quality Gate), along with the SonarQube dashboard displaying the code quality metrics (bugs, vulnerabilities, code smells, test coverage, duplications).

Run SonarQube in Docker and Setup SonarQube

On Jenkins VM:

sudo docker run -d --name sonarqube \

-p 9000:9000 \

-e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true \

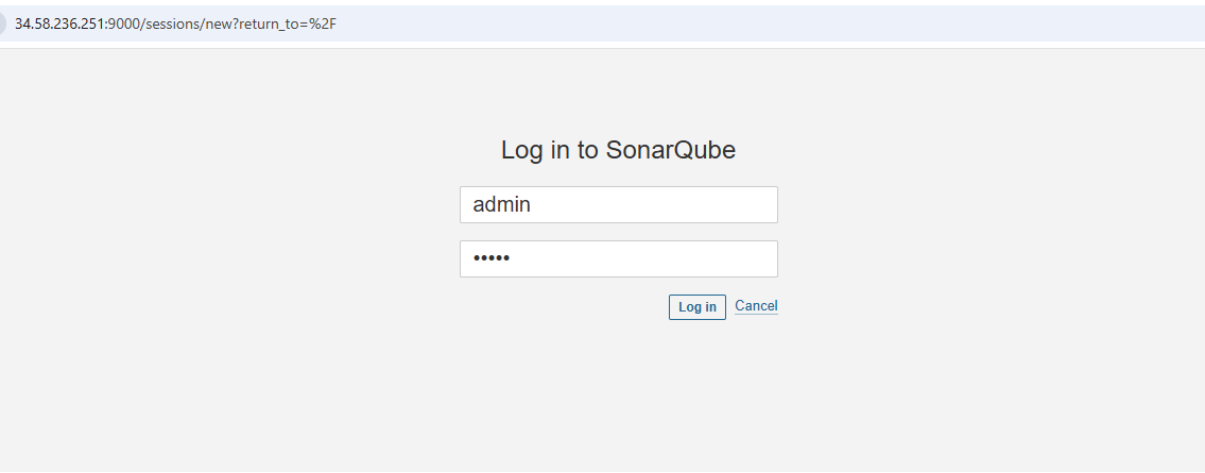
sonarqube:lts-community

```
student-03-23ecc81338de@instance-20250930-182337:~$ sudo docker run -d --name sonarqube \
  -p 9000:9000 \
  -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true \
  sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
60d98d907669: Pull complete
e24a8b9e652f: Pull complete
f3929ce9ef98: Pull complete
1df735f481ad: Pull complete
5d5a1fad7028: Pull complete
eb27e3a98da1: Pull complete
c7ad1fe61e07: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:f709975ab31d2d08f5a3ae2dc73a31ee011afc8cf28845082c17c55d45df9df5
Status: Downloaded newer image for sonarqube:lts-community
df609f22d24b930afdc101218a521e0cc422f5de6af720814f33525998c05c6b
```

Access SonarQube UI

- Open browser:

Access at http://34.41.178.220:9000

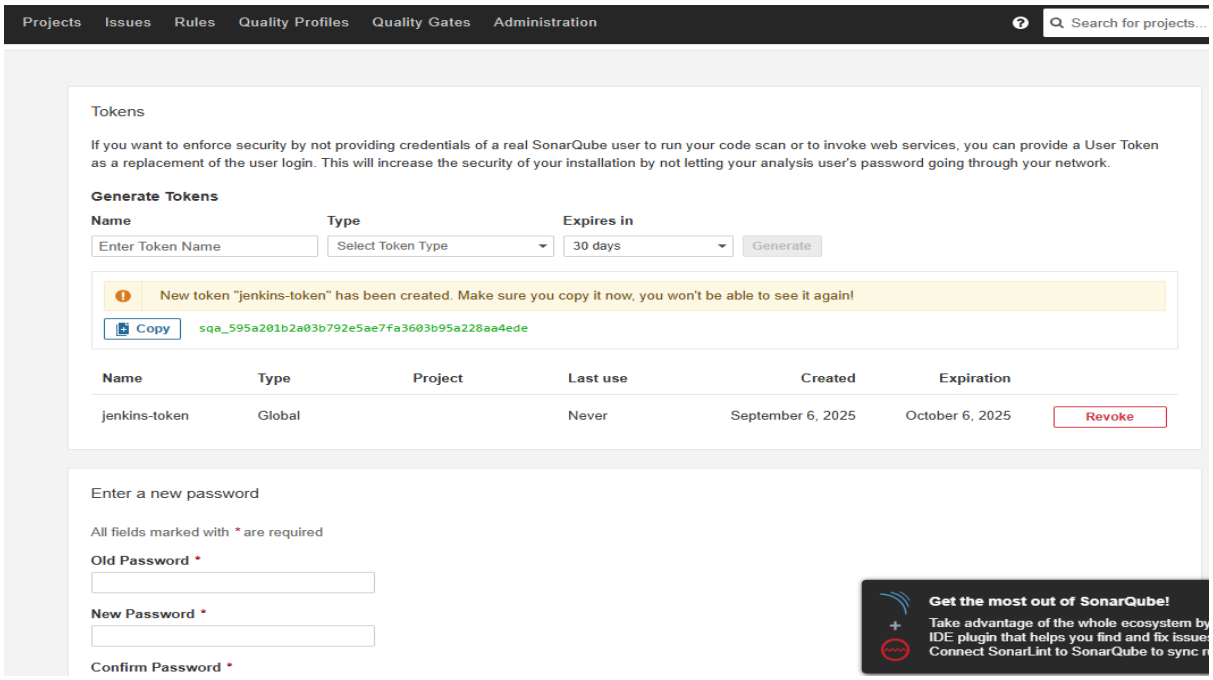


Generate token → Save in Jenkins credentials as **sonar-token**.

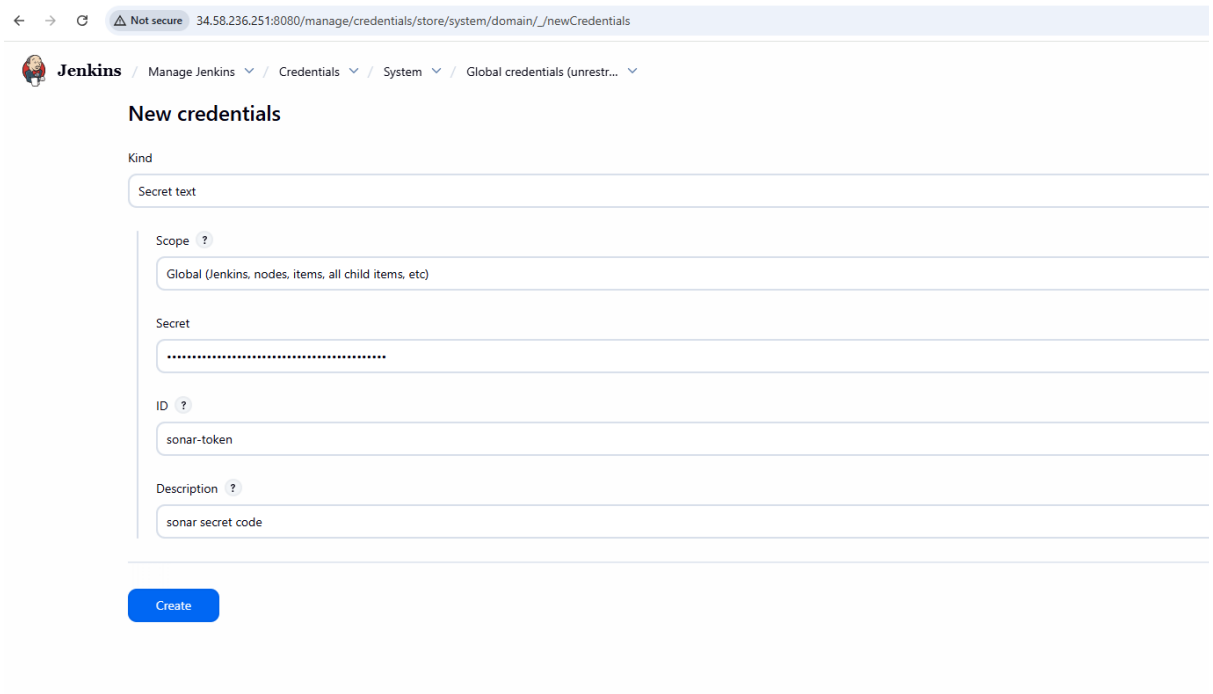
Generate a SonarQube Token

1. After login, click on your profile avatar (top-right).
2. Select **My Account** → **Security** tab.
3. Under **Generate Tokens**:
    - Enter a name (e.g., jenkins-token)
    - Click **Generate**
4. Copy the token shown

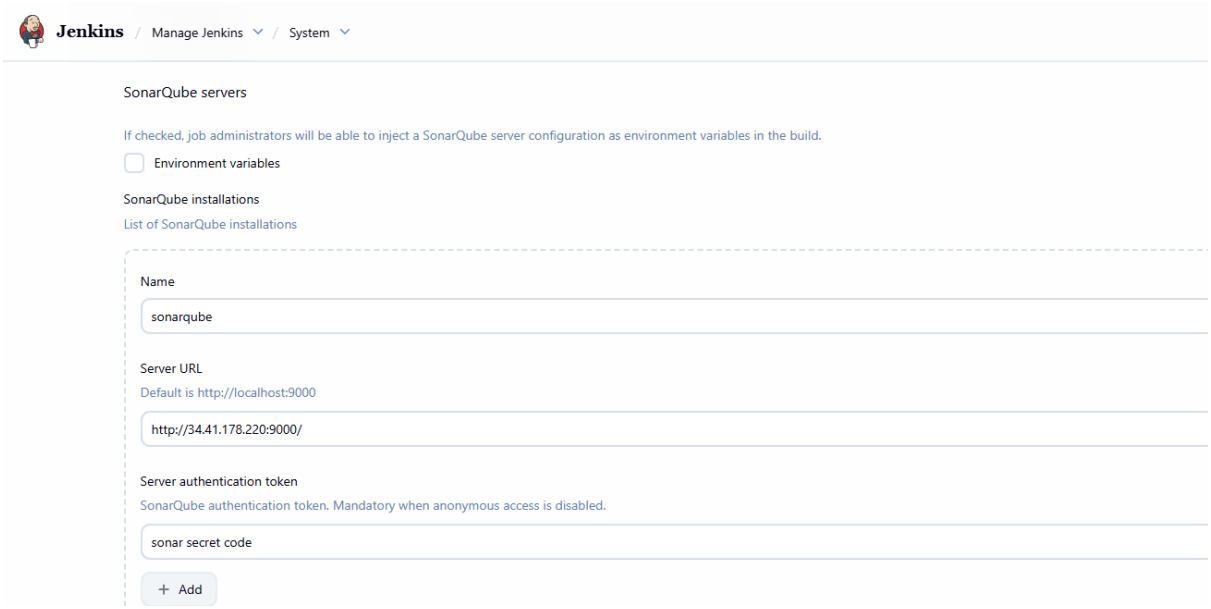
Department of Computer Science and Engineering (Data Science)



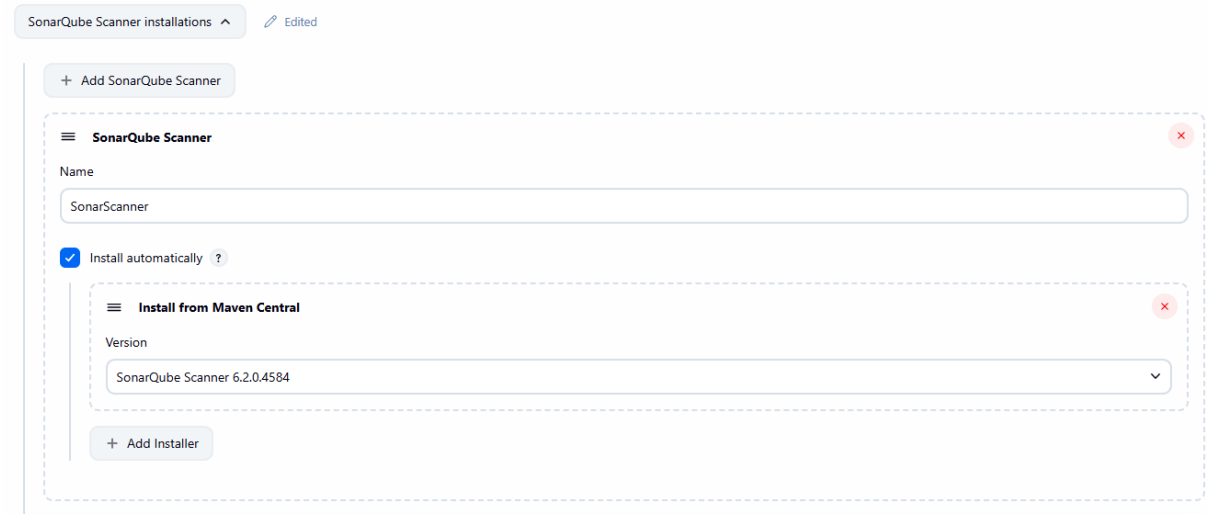
sqa_595a201b2a03b792e5ae7fa3603b95a228aa4ede



Department of Computer Science and Engineering (Data Science)

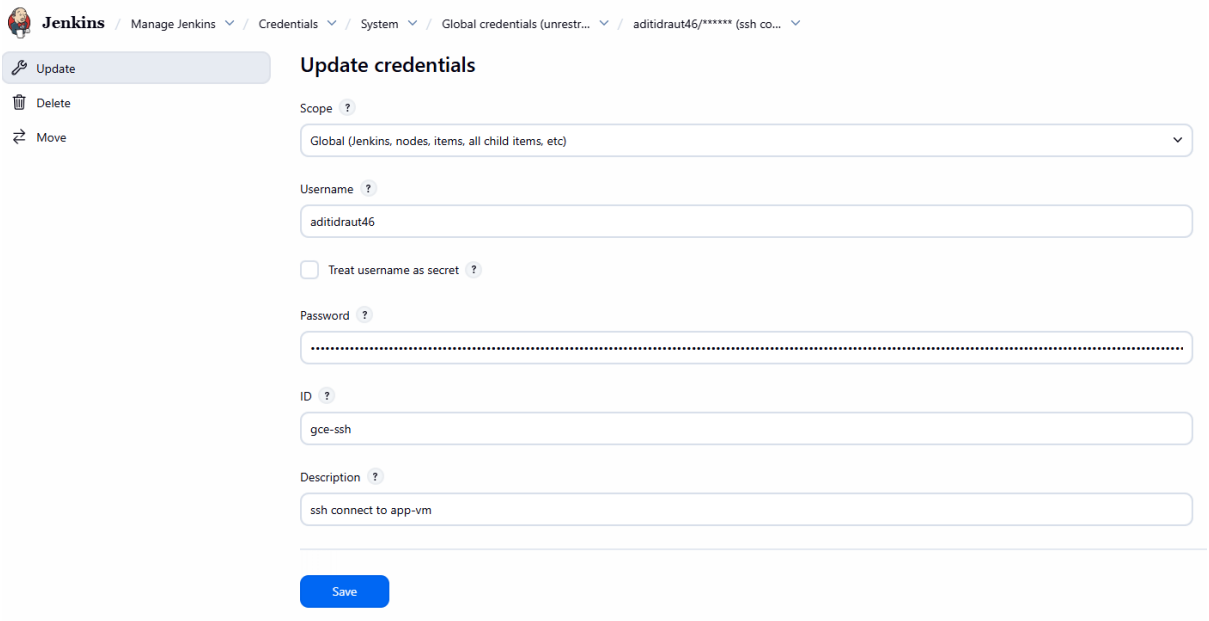


SonarQube Scanner installations



```
student-03-23ecc81338de@instance-20250930-182337:~$ ls -l ~/.ssh/
total 16
-rw------- 1  student-03-23ecc81338de student-03-23ecc81338de  419 Sep  6 08:50 id_ed25519
-rw-r--r-- 1 student-03-23ecc81338de student-03-23ecc81338de  105 Sep  6 08:50 id_ed25519.pub
-rw-r--r-- 1 student-03-23ecc81338de student-03-23ecc81338de  379 Sep  6 08:46 known_hosts
-rw-r--r-- 1 student-03-23ecc81338de student-03-23ecc81338de  142 Sep  6 08:43 known_hosts.old
```

```
student-03-23ecc81338de@instance-20250930-182337:~$ cat ~/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNhY2lzemdqZWFhYWE2NXZibVVhYWEzbm9zZ0FBQUFBQUFBQUFBQUFBQUE=
QynTuXQOAAABE01w6nObkq1mI5EmJvm1PCnQt7BY4HIkcVvS0GLwAAAKchfunzoX7p
8wAAATzc2g7wQynTuXQOAAABE01w6nObkq1mI5EmJvm1PCnQt7BY4HIkcVvS0GLw
AAAFABkOefuXfcw0lmt2SyWD918sPKCDF4tMQoeldkTQj2dqc4GSrUZYjkQyo+au
8kdcOp9Jgc1KQdVLYAAAF2FkaXJp2HjhdQONKBqW5rawS2LXZtAQIDBAUG
-----END OPENSSH PRIVATE KEY-----
```

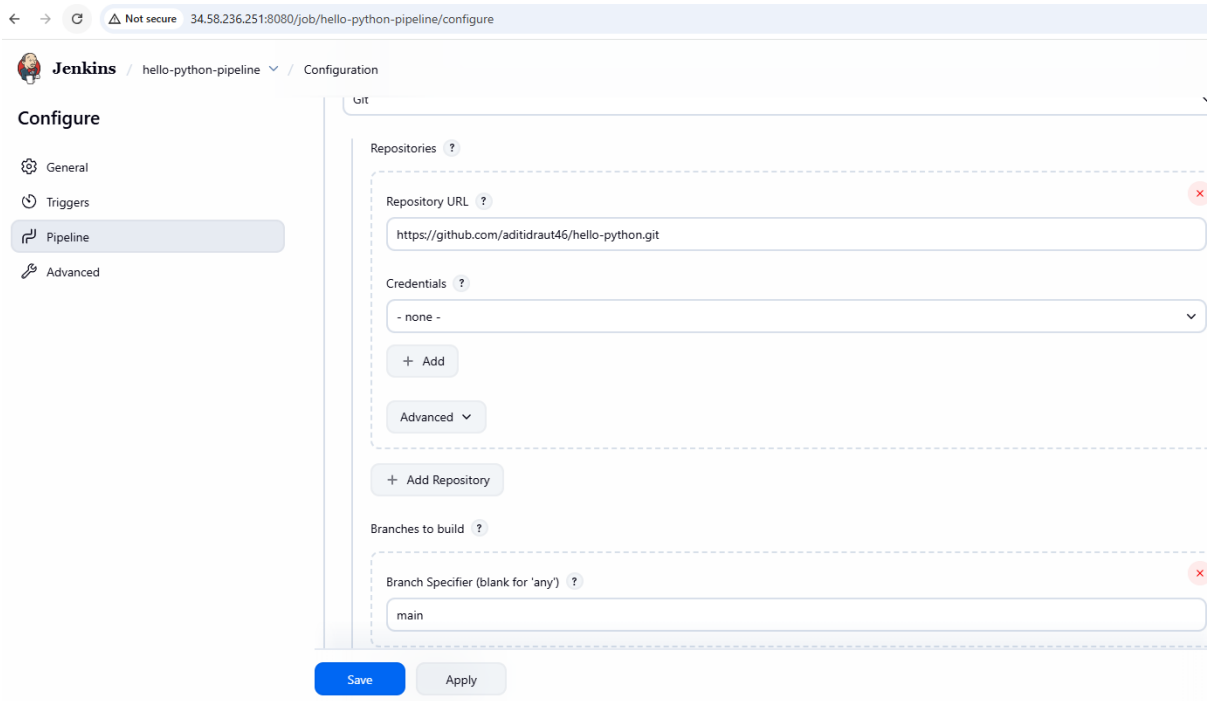


Create Jenkins Pipeline Job

1. Jenkins → **New Item** → hello-python-pipeline → Pipeline → OK
2. Pipeline:
    - Definition: Pipeline script from SCM
    - SCM: Git
    - Repo: https://github.com/ParthSavla2345/hello-python.git
    - Branch: main

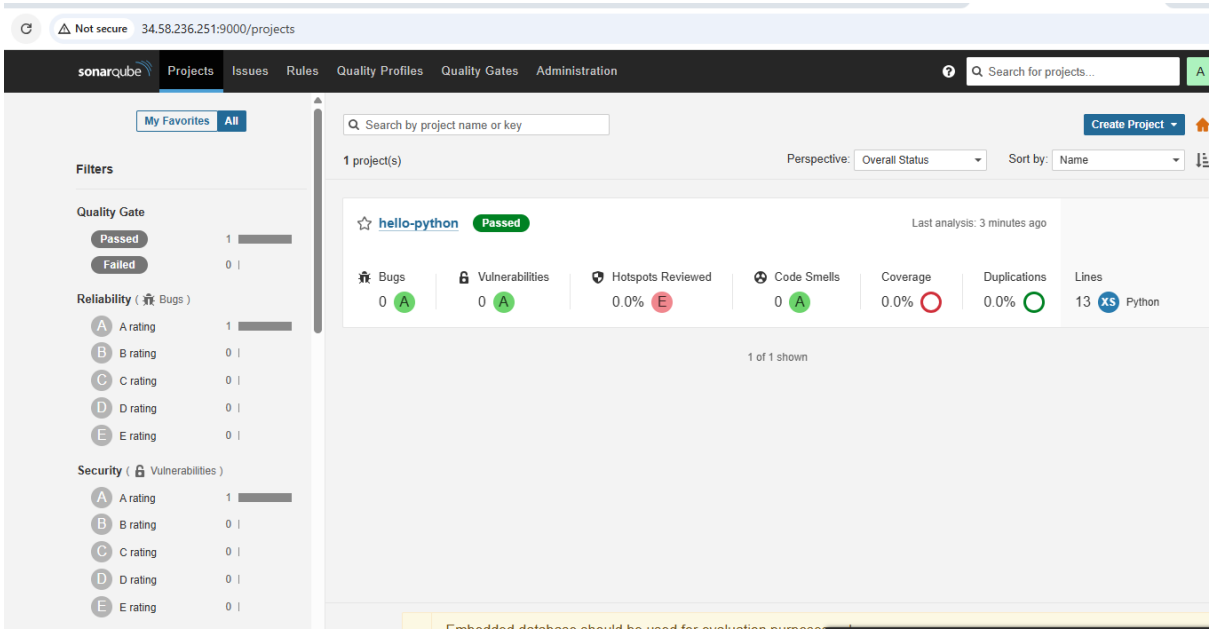
- Script Path: Jenkinsfile

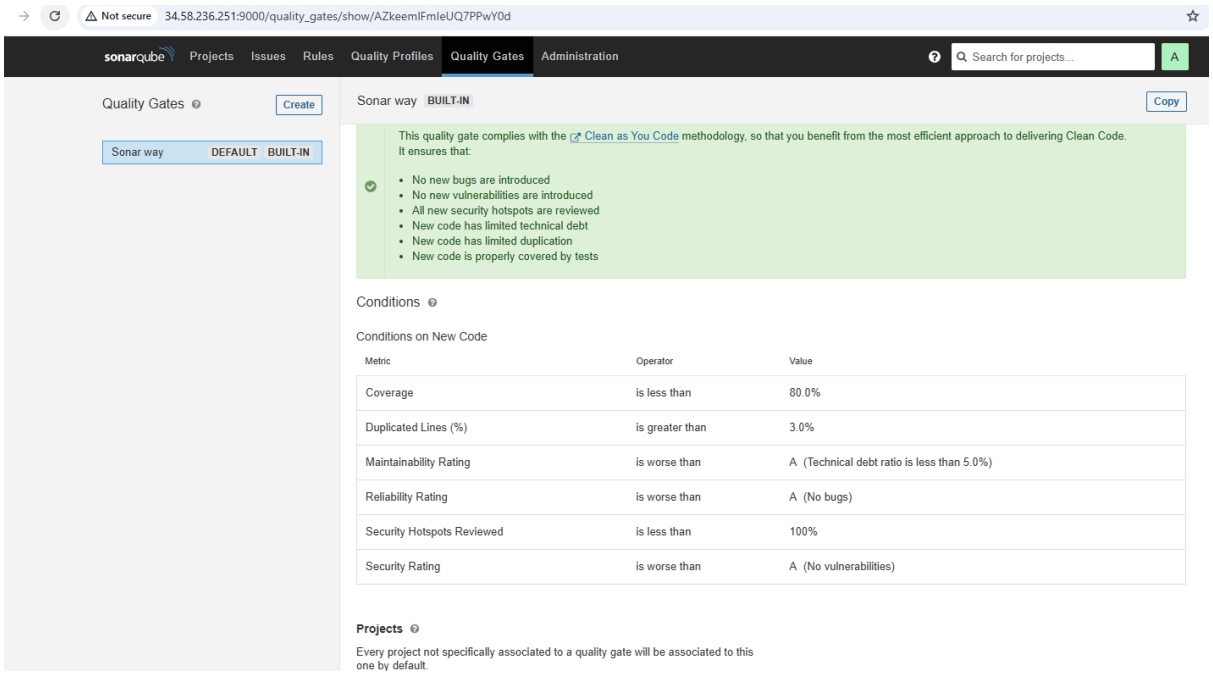
3. Save



How to monitor SonarQube analysis

1. Open your browser:
2. http://<JENKINS_VM_IP>:9000
3. Login (admin / your password).
4. Go to **Projects → hello-python**.
    - You'll see issues, code smells, and vulnerabilities.
    - Quality Gate (pass/fail) is shown at the top.


Not secure 34.58.236.251:9000/projects
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
My Favorites
All
Filters
Quality Gate
Passed 1
Failed 0
Reliability (Bugs)
A rating 1
B rating 0
C rating 0
D rating 0
E rating 0
Security (Vulnerabilities)
A rating 1
B rating 0
C rating 0
D rating 0
E rating 0
Search by project name or key
Create Project
1 project(s)
Perspective: Overall Status
Sort by: Name
hello-python
Passed
Last analysis: 3 minutes ago
Bugs 0 A
Vulnerabilities 0 A
Hotspots Reviewed 0.0% E
Code Smells 0 A
Coverage 0.0%
Duplications 0.0%
Lines 13 XS Python
1 of 1 shown


Not secure 34.58.236.251:9000/quality_gates/show/AZkeemlFmleUQ7PPwY0d
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
Quality Gates
Create
Sonar way DEFAULT BUILT-IN
Sonar way BUILT-IN
Copy
This quality gate complies with the Clean as You Code methodology, so that you benefit from the most efficient approach to delivering Clean Code.
It ensures that:
No new bugs are introduced
No new vulnerabilities are introduced
All new security hotspots are reviewed
New code has limited technical debt
New code has limited duplication
New code is properly covered by tests
Conditions
Conditions on New Code
Metric
Operator
Value
Coverage
is less than
80.0%
Duplicated Lines (%)
is greater than
3.0%
Maintainability Rating
is worse than
A (Technical debt ratio is less than 5.0%)
Reliability Rating
is worse than
A (No bugs)
Security Hotspots Reviewed
is less than
100%
Security Rating
is worse than
A (No vulnerabilities)
Projects
Every project not specifically associated to a quality gate will be associated to this one by default.

Stage-wise Pipeline Testing

Before the final pipeline, test each stage separately.

Test SSH Connection

```
pipeline {
  agent any
  stages {
    stage('Check SSH Connection to App VM') {
      steps {
        sshagent(credentials: ['gce-ssh']) {
          sh '''
            echo "Testing SSH..."
            ssh -o StrictHostKeyChecking=no ParthSavla2345@34.16.36.23 "echo Connected
&& hostname"
          '''
        }
      }
    }
  }
}
```

Test SonarQube Token

```
pipeline {
  agent any
  stages {
    stage('Check SonarQube Token') {
      steps {
        withCredentials([string(credentialsId: 'sonar-token', variable: 'SONAR_TOKEN')]) {
          sh 'curl -s -u ${SONAR_TOKEN}: http://34.41.178.220:9000/api/server/version'
```

```
      }
    }
  }
 }
}
```

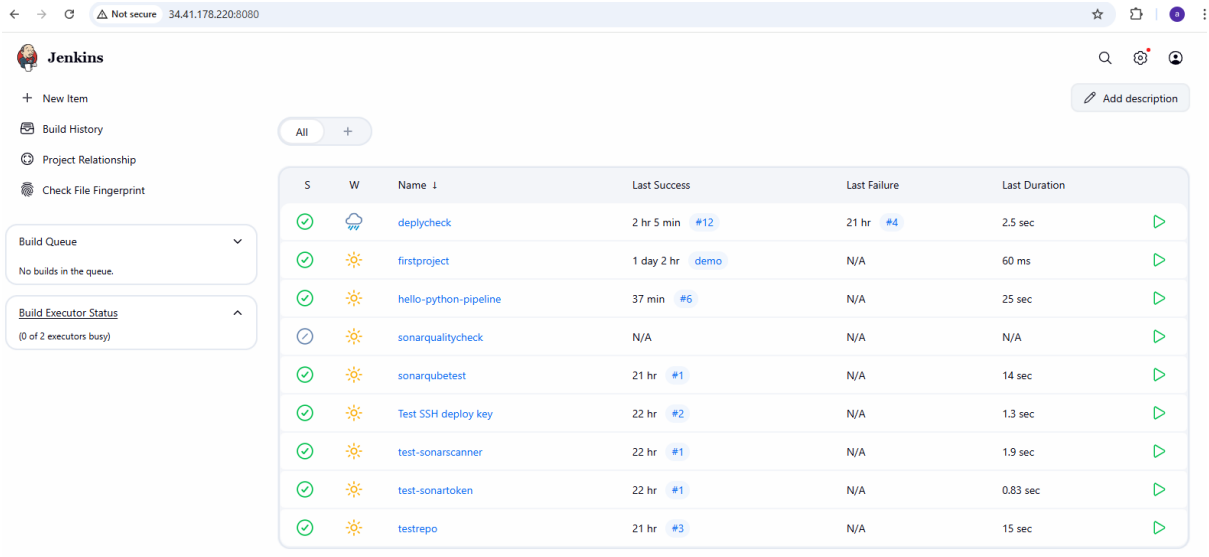
Test SonarScanner

```
pipeline {
  agent any
  stages {
    stage('Check SonarScanner') {
      steps {
        withSonarQubeEnv('sonarqube') {
          withEnv(["PATH+SONAR=${tool 'SonarScanner'}/bin"]) {
            sh 'sonar-scanner -v'
          }
        }
      }
    }
  }
}
```

Test Deployment

```
pipeline {
  agent any
  stages {
    stage('Deploy to App VM') {
      steps {
        sshagent(credentials: ['gce-ssh']) {
```

```
        sh '''
          ssh -o StrictHostKeyChecking=no ParthSavla2345@34.16.36.23 "mkdir -p ~/app"
          scp -o StrictHostKeyChecking=no -r * aditidraut46@34.16.36.23:/home/ ParthSavla2345 /app/
          ssh -o StrictHostKeyChecking=no ParthSavla2345@34.16.36.23 "nohup python3 /home/ ParthSavla2345 /app/app.py > app.log 2>&1 &"
        '''
      }
    }
  }
}
```



Prepare App VM

On **App VM**:

mkdir -p ~/app

cd ~/app

python3 -m pip install --upgrade pip

pip3 install Flask pytest

python3 app.py

On App VM do this

mkdir -p ~/.ssh

echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIETQjzDqc4GSrU2YjkQyO+aU8KdC0PsFjgciQKdVLQYv ParthSavla2345@jenkins-vm">> ~/.ssh/authorized_keys

chmod 600 ~/.ssh/authorized_keys

chmod 700 ~/.ssh

curl http://34.16.36.23:8080

ps -ef | grep app.py

// tail -n 30 /home/student-03-23ecc81338de /app/app.log

cd /home/student-03-23ecc81338de /app

python3 app.py

sudo vim /etc/systemd/system/flaskapp.service

sudo systemctl daemon-reload

sudo systemctl start flaskapp

sudo systemctl status flaskapp

sudo journalctl -u flaskapp -f

This assumes you've already created a flaskapp.service file on your **App VM** at /etc/systemd/system/flaskapp.service.

Example flaskapp.service (on App VM)

[Unit]

Description=Flask App Service

After=network.target

[Service]

User=student-03-23ecc81338de

WorkingDirectory=/home/student-03-23ecc81338de /app

ExecStart=/usr/bin/python3 app.py

Restart=always

[Install]

WantedBy=multi-user.target

Enable it once on App VM:

sudo systemctl daemon-reload

sudo systemctl enable flaskapp

sudo systemctl start flaskapp

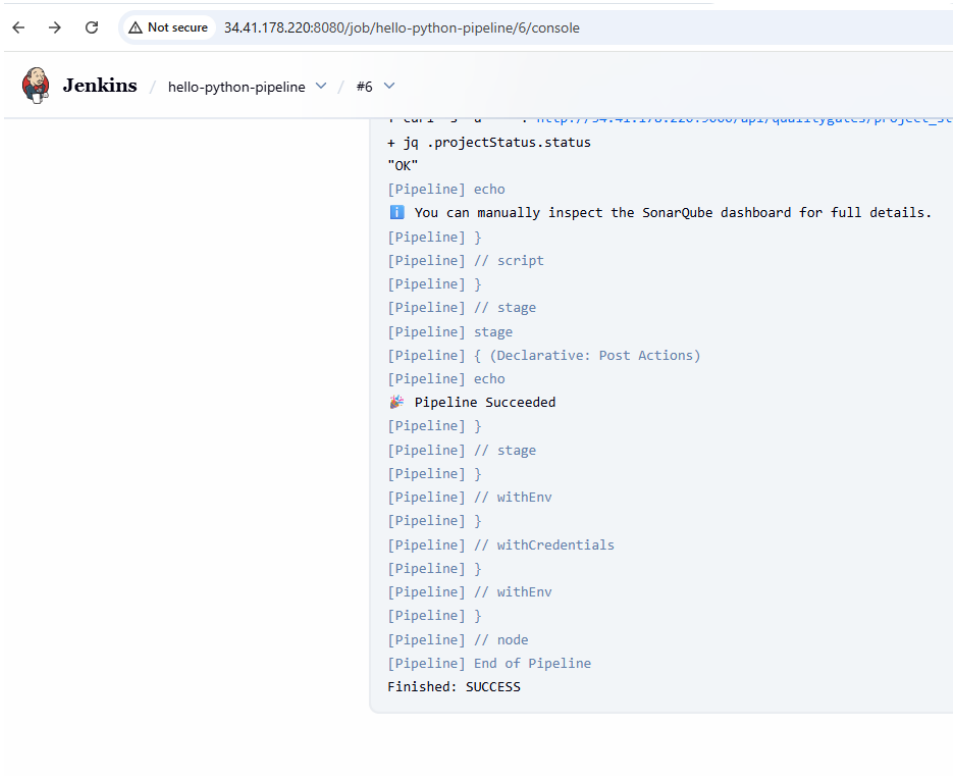
Since restarting a systemd service requires **sudo**, you'll need to configure **password less sudo** for student-03-23ecc81338de on the App VM:

On **App VM**:

student-03-23ecc81338de@app-vm:~/app$ sudo vim visudo

Add this line at the end

ParthSavla2345 ALL=(ALL) NOPASSWD: /bin/systemctl



output after deployment


```
student-03-23ecc81338de@instance-20250930-182337:~$ curl http://34.16.36.23:8080
Hello from App VM!
```