



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.



Department of Computer Science and Engineering (Data Science)

- **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:



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)

Access at <http://34.41.178.220:9000>

34.58.236.251:9000/sessions/new?return_to=%2F

Log in to SonarQube

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)

Projects Issues Rules Quality Profiles Quality Gates Administration ? Search for projects...

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

Enter Token Name

Select Token Type

30 days

Generate

New token "jenkins-token" has been created. Make sure you copy it now, you won't be able to see it again!

Copy `sqa_595a201b2a03b792e5ae7fa3603b95a228aa4ede`

| Name | Type | Project | Last use | Created | Expiration | |
|---------------|--------|---------|----------|-------------------|-----------------|--------|
| jenkins-token | Global | | Never | September 6, 2025 | October 6, 2025 | Revoke |

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Get the most out of SonarQube!

Take advantage of the whole ecosystem by IDE plugin that helps you find and fix issues. Connect SonarLint to SonarQube to sync results.

sqa_595a201b2a03b792e5ae7fa3603b95a228aa4ede

< > ↺ Not secure 34.58.236.251:8080/manage/credentials/store/system/domain/_/newCredentials

Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestricted)

New credentials

Kind

Secret text

Scope ?

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

Secret

.....

ID ?

sonar-token

Description ?

sonar secret code

Create



Department of Computer Science and Engineering (Data Science)

← → ↻ ⚠ Not secure 34.58.236.251:8080/manage/pluginManager/available ☆ 📄 🗨



Jenkins / Manage Jenkins ▾ / Plugins

🔍 ⚙️ 👤

Plugins

- 📅 Updates
- 📦 Available plugins**
- 🔗 Installed plugins
- ⚙️ Advanced settings
- 📊 Download progress

🔍 sonar

📦 Install ▾ 🔄

| Install | Name ↓ | Released | Health |
|-------------------------------------|--|------------------|--------|
| <input checked="" type="checkbox"/> | SonarQube Scanner 2.18 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality. | 7 mo 10 days ago | 84 |
| <input type="checkbox"/> | Sonar Quality Gates 352.vdcdb_d7994fb_6 Library plugins (for use by other plugins) analysis Other Post-Build Actions Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality | 6 mo 15 days ago | 100 |



Jenkins / 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

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.

SonarQube Scanner installations

SonarQube Scanner installations ▾

[✎ Edited](#)

☰ SonarQube Scanner

Name

☒ Install automatically ?

☰ Install from Maven Central

Version



Department of Computer Science and Engineering (Data Science)

```
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-----
b3B1bnNhY2lzemdqZWZhYWE2NXZibVVhYWEzbn9zZ0FBQUFBQUFBQUFBQUFBQUE=
QynTuXQOAAABE0lw6nObkq1mI5EmJvmlPCnQt7BY4HIkcVvS0GLwAAAKchfunzoX7p
8wAAATzc2g7wQynTuXQOAAABE0lw6nObkq1mI5EmJvmlPCnQt7BY4HIkcVvS0GLw
AAAFABkOefuXfcw0lmt2SyWD9l8sPKCDF4tMQoeldkTQj2dqc4GSrUZYjkQyo+au
8kdcOp9Jgc1KQdVLYAAAF2FkaXJp2HjhdQONKBqW5rawS2LXZtAQIDBAUG
-----END OPENSSH PRIVATE KEY-----
```

Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestr...) / aditidraut46/***** (ssh co...)

Update

Delete

Move

Update credentials

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

Username ?
aditidraut46

☐ Treat username as secret ?

Password ?
.....

ID ?
gce-ssh

Description ?
ssh connect to app-vm

Save

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



Department of Computer Science and Engineering (Data Science)

- Script Path: Jenkinsfile

3. Save

← → 🔍 Not secure 34.58.236.251:8080/job/hello-python-pipeline/configure

Jenkins / hello-python-pipeline / Configuration

Configure

- General
- Triggers
- Pipeline**
- Advanced

Git

Repositories ?

Repository URL ? ✖

https://github.com/aditidraut46/hello-python.git

Credentials ?

- none - ▼

+ Add

Advanced ▼

+ Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ? ✖

main

Save Apply

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.



Department of Computer Science and Engineering (Data Science)

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

My Favorites All

Search by project name or key

1 project(s) Perspective: Overall Status Sort by: Name

hello-python Passed Last analysis: 3 minutes ago

| | | | | | | |
|------|-----------------|-------------------|-------------|----------|--------------|--------------|
| Bugs | Vulnerabilities | Hotspots Reviewed | Code Smells | Coverage | Duplications | Lines |
| 0 A | 0 A | 0.0% E | 0 A | 0.0% | 0.0% | 13 XS Python |

1 of 1 shown

Embedded database should be used for evaluation purposes

→ 34.58.236.251:9000/quality_gates/show/AZkeemIFmleUQ7PPwY0d

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Quality Gates Create

Sonar way DEFAULT 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.



Department of Computer Science and Engineering (Data Science)

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'        }  
      }  
    }  
  }  
}
```



Department of Computer Science and Engineering (Data Science)

```
}  
  
}  
  
}  
  
}  
  
}
```

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']) {
```



Department of Computer Science and Engineering (Data Science)

```
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 &"
```

```
'''
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

| S | W | Name | Last Success | Last Failure | Last Duration |
|---|---|-----------------------|-----------------|--------------|---------------|
| ✓ | ☁ | deplycheck | 2 hr 5 min #12 | 21 hr #4 | 2.5 sec |
| ✓ | ☀ | firstproject | 1 day 2 hr demo | N/A | 60 ms |
| ✓ | ☀ | hello-python-pipeline | 37 min #6 | N/A | 25 sec |
| ⌚ | ☀ | sonarqualitycheck | N/A | N/A | N/A |
| ✓ | ☀ | sonarqubetest | 21 hr #1 | N/A | 14 sec |
| ✓ | ☀ | Test SSH deploy key | 22 hr #2 | N/A | 1.3 sec |
| ✓ | ☀ | test-sonarscanner | 22 hr #1 | N/A | 1.9 sec |
| ✓ | ☀ | test-sonartoken | 22 hr #1 | N/A | 0.83 sec |
| ✓ | ☀ | testrepo | 21 hr #3 | N/A | 15 sec |

Prepare App VM

On App VM:

```
mkdir -p ~/app
```

```
cd ~/app
```



Department of Computer Science and Engineering (Data Science)

```
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+aU8KdC0PsFjgciQKdVL  
QYv 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
```



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)

[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
```



Department of Computer Science and Engineering (Data Science)

```
← → ↻ ⚠ Not secure 34.41.178.220:8080/job/hello-python-pipeline/6/console

Jenkins / hello-python-pipeline / #6

+ curl -s -u <username>:<password> http://34.41.178.220:8080/api/queue/getters/project_<id>
+ jq .projectStatus.status
"OK"
[Pipeline] echo
  You can manually inspect the SonarQube dashboard for full details.
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
  Pipeline Succeeded
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

output after deployment

```
← → ↻ ⚠ Not secure 34.16.36.23:8080

Hello from App VM!
```

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