Hands-on Mini Project: Git → Jenkins → SonarQube → Artifactory CI Pipeline

🔧 Objective:

Simulate a complete CI pipeline where:

- Developer commits code → Jenkins triggers build
- SonarQube performs static analysis
- Artifacts are published to Artifactory
- QA reviews build, identifies failure, and fixes it

🧾 Project: Sample Python App (Calculator)

📁 Directory Structure:

```
python-ci-demo/
├── app.py
├── tests/
│   └── test_app.py
├── requirements.txt
├── sonar-project.properties
├── Jenkinsfile

```

👩‍💻 Sample Code:

`app.py`

```
def divide(a, b):
    return a / b

if __name__ == "__main__":
    print("Division:", divide(10, 2))

```

`tests/test_app.py`

```
from app import divide

def test_divide():
    assert divide(10, 2) == 5

```

> 💡 You'll intentionally introduce a failing test in a later step.

`requirements.txt`

```
pytest
flake8
```

```

```

`sonar-project.properties`

```
sonar.projectKey=python-ci-demo
sonar.projectName=Python CI Demo
sonar.sources=.
sonar.language=py
sonar.sourceEncoding=UTF-8

```

📜 Jenkinsfile

```
pipeline {
    agent any

    environment {
        SONAR_SCANNER_HOME = tool 'SonarQube Scanner'
    }

    stages {
        stage('Checkout') {
            steps {
                git 'https://bitbucket.company.com/scm/devops/python-ci-demo.git'
            }
        }

        stage('Install Deps') {
            steps {
                sh 'pip install -r requirements.txt'
            }
        }

        stage('Lint') {
            steps {
                sh 'flake8 app.py'
            }
        }

        stage('Test') {
            steps {
                sh 'pytest tests/'
            }
        }

        stage('SonarQube Scan') {
            steps {
                withSonarQubeEnv('SonarQube') {
                    sh "${SONAR_SCANNER_HOME}/bin/sonar-scanner"
                }
            }
        }

        stage('Publish to Artifactory') {
            steps {
```

```
                sh 'zip -r app.zip app.py tests/'
                rtUpload (
                    serverId: 'Artifactory-Server',
                    spec: '''{
                      "files": [
                        {
                          "pattern": "app.zip",
                          "target": "libs-release-local/python-ci-demo/"
                        }
                      ]
                    }'''
                )
            }
        }
    }
}

```

🧪 Execution Flow

1. **Dev commits code** to Bitbucket → Jenkins auto-triggers pipeline
2. **Jenkins stages**:
   - Code checkout
   - Dependency install
   - Code lint (via `flake8`)
   - Unit tests (`pytest`)
   - Static code scan (SonarQube)
   - Artifact publishing (Artifactory)

🧩 Mini Challenge: Simulate & Fix Build Failure

1. **Break the unit test** (simulate a bad commit):

```
def test_divide():
    assert divide(10, 3) == 5  # <-- Incorrect test

```

2. Commit and push → Jenkins fails test stage.
3. **QA or Dev reviews**:
   - Check **console output** in Jenkins
   - Check **SonarQube dashboard** for code smells or bugs
4. **Fix the code/test**, commit, and re-run pipeline

✅ Outcome

✔️ Realistic CI/CD pipeline using Git → Jenkins → SonarQube → Artifactory
✔️ Understand pipeline failures and fix flow
✔️ Full traceability across SCM, quality, and artifact management