4. Create a Jenkins CI/CD pipeline with SonarQube integration to perform static analysis code analysis.

Step 1: Install SonarQube and SonarQube Scanner

a) Running SonarQube via Docker

sudo docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

- Open **http://localhost:9000** in a browser.
- Default credentials: **admin/admin**

b) Installing SonarQube Scanner in Jenkins

- Go to **Jenkins Dashboard → Manage Jenkins → Manage Plugins**.
- Install **SonarQube Scanner** from Available Plugins.
- Go to **Manage Jenkins → Global Tool Configuration**.
- Configure **SonarQube Scanner** by adding a new installation.

Step 2: Configure SonarQube in Jenkins

- **Navigate to Manage Jenkins → Configure System**.
- Find **SonarQube Servers** section and click **Add SonarQube**.
- Add the **SonarQube Server URL** (`http://localhost:9000`).
- Generate a **SonarQube Token** from SonarQube UI:
  - Go to **My Account → Security → Generate Token**.
  - Copy the token and configure it in Jenkins.
- In Jenkins, under **Manage Credentials**, add a new secret text credential.

Step 3: Create a Jenkins Pipeline

1. Navigate to **Jenkins Dashboard → New Item**.
2. Select **Pipeline**, provide a name, and click **OK**.
3. In the pipeline script section, add the following **Jenkinsfile**:

Jenkinsfile:

```
pipeline {
    agent any

    environment {
        SONARQUBE_URL = 'http://localhost:9000'
        SONAR_TOKEN = credentials('sonar-token-id')  // Replace with your credentials ID
    }
```

```
    stages {
        stage('Checkout Code') {
            steps {
                git 'https://github.com/your-repo/sample-project.git'
            }
        }

        stage('Build') {
            steps {
                sh 'mvn clean install' // Replace with your build command
            }
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('SonarQube') { // Use the name configured in Jenkins
                    sh 'mvn sonar:sonar -Dsonar.projectKey=sample-project -
Dsonar.host.url=$SONARQUBE_URL -Dsonar.login=$SONAR_TOKEN'
                }
            }
        }

        stage('Quality Gate') {
            steps {
                script {
                    timeout(time: 1, unit: 'MINUTES') {
                        def qg = waitForQualityGate()
                        if (qg.status != 'OK') {
                            error "Pipeline failed due to quality gate failure: ${qg.status}"
                        }
                    }
                }
            }
        }
    }
}
```

Step 4: Run the Pipeline

- Click **Build Now** in Jenkins.
- Check the console output for:
  - **SonarQube Analysis Completion**.
  - **Quality Gate Status**.
- Open **SonarQube UI** (`http://localhost:9000`), navigate to the project, and view the code analysis results.

Expected Outcome:

- Jenkins successfully builds the project.
- SonarQube performs **static code analysis**.

- If the **Quality Gate** fails, the pipeline stops execution.

2. Build a pipeline of jobs using maven in Jenkins.

Step 1: Create a New Pipeline Job in Jenkins

- Open Jenkins Dashboard.
- Click on **"New Item"** → Select **"Pipeline"** → Enter a job name.
- Click **OK**.

Step 2: Add Pipeline Script

In the **Pipeline section**, select **Pipeline script** and add the following script:

Jenkins Pipeline Script for Maven Build:

```
groovy
CopyEdit
pipeline {
    agent any

    tools {
        maven 'Maven_3' // Ensure Maven is configured in Jenkins
    }

    environment {
        SONARQUBE_SERVER = 'SonarQube' // SonarQube configuration name in Jenkins
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/your-repo.git'
            }
        }

        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }

        stage('Unit Tests') {
            steps {
                sh 'mvn test'
            }
        }

        stage('SonarQube Analysis') {
```

```
        steps {
            withSonarQubeEnv(SONARQUBE_SERVER) {
                sh 'mvn sonar:sonar -Dsonar.host.url=http://localhost:9000'
            }
        }
    }

    stage('Quality Gate') {
        steps {
            timeout(time: 1, unit: 'MINUTES') {
                waitForQualityGate abortPipeline: true
            }
        }
    }

    stage('Integration Tests') {
        steps {
            sh 'mvn verify'
        }
    }

    stage('Deploy') {
        steps {
            sh 'mvn deploy'
        }
    }
  }
}
```

Step 3: Save and Run the Pipeline

- Click **Save** and then **Build Now**.
- Monitor the **Console Output** for logs.