Maven and GIT

Generate the build(git and maven)

- Install openjdk-17-jdk, maven, jenkins

- pull project from github

stage:git-checkout

- generate the build

stage: mvn test

```
node {
   stage('hello'){

      sh 'echo "Hello world"'
   }
   stage('git-checkout'){

      git branch: 'main', url: 'https://github.com/Sangeetha-j-QSP/spring-petclinic.git'
   }
   stage('Build generation'){

      sh 'mvn clean test'
   }
}
```

NOTE:

You'll get the syntax for each stage or for each step in pipeline syntax
- sample step for git is "git:git"
- sample step for maven is "sh:Shell scripting"

Launch a ubuntu instance:
- update system and install openjdk-17-jdk, maven, jenkins
- complete jenkins installation process and set-up jenkins
- Configure maven in jenkins by installing maven plugin
(Maven Integration) and configure maven in "tools"
(by adding name as maven and maven_home as "/usr/share/maven")
- Create a new item of job type "Pipelines"
- Get inside configure and select pipeline template as
"Scripted pipeline"

Sonarqube:

security group port no. "22", "80", "8080", "9000", "8081" should be opened

Scanning the source code(Sonarqube)

- Install openjdk-17-jdk, sonarqube

scan for vulnerabilities in source code

stage: sonarscan

Launch ubuntu instance:
- Install openjdk-17-jdk, sonarqube(refer maven notes)
- Go to browser
  -> http://<instance-ip>:9000
    - setup sonarqube
    - click on profile icon(topmost right corner)
    - click on security
    - generate the token
      -> give token name(sonar-token)
      -> select type as "global analysis token"
      -> click on "generate"
      -> copy the secret text and keep it safe

(eg: "sqa_683bd347f5bfff942cb2f0a1798ff534904b68a4")

- go to jenkins tab
  - install sonar plugin(sonarqube scanner)
  - manage jenkins -> credentials -> global -> Add credentials
    -> select sample kind "secret text"
    -> paste the copied secret text
    -> give id (sonar-token) and description
  - click on "create"

configure system:
- manage jenkins - system - sonar installations - add sonarqube
  - give name (sonarqube)
  - give url (http://<public-ip>:9000)
  - click on "server authentication token"
    -> select the credential(sonar-token)

go to job:
- click on "configure"

```
node {
    stage('git checkout'){
        git branch: 'main', url: 'https://github.com/Sangeetha-j-QSP/spring-
        framework-petclinic.git'

    }

    stage('build genration'){
        sh 'mvn package'

    }

    stage('sonar analysis'){
        withSonarQubeEnv(installationName: 'sonarqube', credentialsId: 'sonar-token') {

            sh 'mvn sonar:sonar'

        }


    }

}
```

Nexus

To upload/deploy project in remote repo(nexus)

- Install openjdk-17-jdk, nexus

- deploy it into remote repo

stage: nexus deploy

launch ubuntu instance:
- install openjdk-17-jdk, nexus(refer maven notes)
- setup nexus by giving new password
- create repository in nexus tab(refer maven notes)

Install and set nexus password, create new repo

jenkins tab:
- manage jenkins -> plugins -> select "nexus artifact uploader" and "install"
- manage jenkins -> credentials -> stored scoped to jenkins(global)
- click on "add credential"
- select "username and password" in kind
    - give username and password as same in nexus (username -> admin, password -> 1234)
    - give id(nexus-credential) and description
- click on create

Install nexus plugin, create nexus credential

```
node {
    stage('git checkout'){
            git branch: 'main', url: 'https://github.com/Sangeetha-j-QSP/spring-framework-petclinic.git'
     }
    stage('build genration'){
            sh 'mvn package'
    }
    stage('sonar analysis'){
            withSonarQubeEnv(installationName: 'sonarqube', credentialsId: 'sonar-token') {
                    sh 'mvn sonar:sonar'
            }
     }
     stage('nexus artifact'){
            nexusArtifactUploader artifacts:
            [[artifactId: 'spring-framework-petclinic',
              classifier: '',
              file: 'target/petclinic.war',
              type: 'war']],
              credentialsId: 'nexus',
              groupId: 'org.springframework.samples',
              nexusUrl: '54.183.136.240:8081',
              nexusVersion: 'nexus3',
              protocol: 'http',
              repository: 'petclinic',
              version: '1.0-SNAPSHOT'
     }
}
```

NOTE: Click on pipeline syntax to get the code of this particular stage

- for groupId, artifactId => refer pom.xml

Tomcat

To host the application (tomcat10)

- install java and tomcat
- deploy/host the project on the web-server

stage: tomcat deploy

comment valve in context.xml

<!-- -->

- Install java and tomcat
- start tomcat server
  -> extract tar file
  -> cd apache-tomcat => cd bin => ./startup.sh
- copy public ip of instance in browser with port no. 8080
  => http://<public-ip>:8080
- comment valve in context.xml
  => cd apache-tomcat.../webapps/manager/META-INF
  => nano context.xml
  <!-- <valve> ..... </valve> -->
- add manager role in tomcat-users.xml
  => cd apache-tomcat..../conf
  => nano tomcat-users.xml
  <role rolename="manager-gui"/>
  <user username="manager" password="1234" roles="manager-gui"/>
- stop and start tomcat
  => cd apache-tomcat/bin
  => ./shutdown.sh
  =>./startup.sh

Jenkins browser:
- install "ssh agent" plugin
- create tomcat credential using private key file of tomcat instance
  => click on manage jenkins => crdentials => global => add credentials
  => select kind as "ssh username with private key"
  => user_name => ubuntu
  => add private key
- add the tomcat stage in pipeline

```
sh """
  scp -o StrictHostKeyChecking=no target/*.war ubuntu@<tomcat-public-ip>:/home/ubuntu/apache-tomcat-10.1.42/webapps
  ssh -o StrictHostKeyChecking=no ubuntu@<tomcat-public-ip> /home/ubuntu/apache-tomcat-10.1.42/bin/shutdown.sh
  ssh -o StrictHostKeyChecking=no ubuntu@54.173.252.57 /home/ubuntu/apache-tomcat-10.1.42/bin/startup.sh
  """
```

```
node {
    stage('git checkout'){
        git branch: 'main', url: ' https://github.com/Sangeetha-j-QSP/spring-framework-petclinic.git'

    }

    stage('mvn build generation'){
        sh 'mvn clean package'

    }
    stage('sonar analysis'){
        withSonarQubeEnv(installationName: 'sonarqube', credentialsId: 'sonarqube-token') {
            sh 'mvn sonar:sonar'
        }

    }
    stage('nexus deployment'){
        nexusArtifactUploader artifacts:
            [[artifactId: 'spring-framework-petclinic',
              classifier: '',
              file: 'target/petclinic.war',
              type: 'war']],
              credentialsId: 'nexus',
              groupId: 'org.springframework.samples',
              nexusUrl: '3.101.14.69:8081',
              nexusVersion: 'nexus3',
              protocol: 'http',
              repository: 'spring-framework-petclinic',
              version: '1.0-SNAPSHOT'
    }

    stage('application hosting'){
        sshagent(['tomcat-credentials']) {
        sh """
        scp -o StrictHostKeyChecking=no target/petclinic.war ubuntu@54.153.109.142:/home/ubuntu/apache-tomcat-10.1.42/webapps
        ssh -o StrictHostKeyChecking=no ubuntu@54.153.109.142 /home/ubuntu/apache-tomcat-10.1.42/bin/shutdown.sh
        ssh -o StrictHostKeyChecking=no ubuntu@54.153.109.142 /home/ubuntu/apache-tomcat-10.1.42/bin/startup.sh
        """
        }
    }
}
```