Lab: CICDPipeline -DockerPushPullDeploy using Groovy script

Pre-Requisites: -

1. Jenkins server should be set up. If already set up, take the IP dedicated for your group. If not set, please coordinate with trainer for the same. Use the credentials provided by Trainer for login into Jenkins
2. Docker should be installed in Jenkins Server. For training purpose, Docker is already installed and your Jenkins servers is already configured to be used with docker. Steps taken are mentioned in section **Configuring Jenkins server to be used docker for deploying code**
3. Jenkins already have following plugins installed: -
    a. **GitHub Integration:** This provides configurations fields required for Jenkins and GitHub Integration.
    b. **Maven Integration Plugin:** This plugin provides integration between Jenkins and Maven and help in automatic building projects.
    c. **Pipeline:** A suite of plugins that lets you orchestrate automation, simple or complex.
    d. **Docker:** This plugin integrates jenkins with docker.
    e. SSH Agent: This plugin allows you to provide SSH credentials to builds via a ssh-agent in jenkins.
4. Jenkins need some additional settings to be configured from UI for deploying code using docker container. The configuration settings already done as mentioned in section **Configuring Jenkins front end to be used docker for deploying code**
5. To publish docker image on DockerHub, Jenkins need to be configured. Configuration done following the settings mention in section **Configuring Jenkins for publishing docker image into DockerHub** **. Refer and add for your dockerhub password similarly.**
6. To pull and run docker image in deployment server. Some settings are configured in Jenkins following the section **Configuring Jenkins for pulling and running docker image in Deployment Server**
7. Deployment Server should be set up with docker installed in it. For training, docker is already installed in the server with IP **35.168.86.36**. We need to configure Jenkins for deploying docker image in server.
8. Git Repository with the code to be deployed should be available. The Git repo to be used in training. You can fork the repo and use wherever required in this lab. https://github.com/rajnikhattarrsinha/java-tomcat-maven-example
9. GitHub should be configured to support Jenkins. Steps to be used mentioned in **Configure GitHub Webhook for Jenkins**
10. Jenkins should already be configured to support following features and tools: -
    a. GitHub (Steps used mentioned in **Configure Jenkins for GitHub**
    b. Maven (Steps used mentioned in **Configure Jenkins for building Maven Projects)**
11. Git Repo should have the Jenkinsfile**.** Script written in Jenkinsfile placed in root directory of Git Repo is mentioned in section **Jenkinsfile Script.** Please note that you would need to modify the script to support your docker hub following the steps mentioned in section **Steps to Follow at GitHub end:**

12. Git Repo should have the Dockerfile**.** Script written in Dockerfile placed in root directory of Git Repo is mentioned in section **Dockerfile Script.** Please note that this step is already done for this training.
13. DockerHub should be Up. Logged in and opened in browser

Note: To know how to install plugin, follow instructions mentioned in document **Installing Plugins in Jenkins**

Steps to Follow at GitHub end:

1. Open Jenkinsfile in your repo, click pencil icon to edit it, Replace **rajnikhattarrsinha** with <your dockerhub username> throughout the script.
2. Commit the changes.
3. Open Jenkinsfile in your repo, click pencil icon to edit it,
   In line `withCredentials([string(credentialsId: 'dockerpwd', variable: 'dockerPWD')])`
   Replace "dockerpwd" with "dockerpwd<yourname>" and "`dockerPWD`" with "`dockerPWD`<yourname>".
   In Line , `sh "docker login -u rajnikhattarrsinha -p ${dockerPWD}",`
   `Replace "dockerPWD" with "dockerPWD`<yourname>"
4. Open pom.xml, click pencil icon to edit it, Replace line `<artifactId>java-tomcat-maven-example_reliance</artifactId>` with `<artifactId>java-tomcat-maven-example_<YOURNAME></artifactId>`
5. Commit the changes.

Steps to Follow at Jenkins end:

6. Click **New Item** link on left panel
7. **Enter an Item name** like <yourname>_CICDGroovyDockerPipeline
8. Select **Pipeline**
9. Click OK
10. Select General Tab, **GitHub Project** as <repo to be deployed>
11. Under Build Triggers section, Select **GitHub hook trigger for GITScm polling** checkbox.
12. Under Pipeline section, Select **Definition** as "Pipe line script from SCM"
13. Select **SCM** as Git
14. Enter Repository URL as <repo to be used>
15. Keep **Script Path** as "Jenkinsfile"
    Note: We are not modifying the path as we have our Jenkinsfile in root directory under Git repo, else set the path of the Jenkinsfile.
16. Click Save
17. On GitHub end, Edit Sample.txt and Commit the changes.
18. Open the Jenkins and observe that the build is triggered automatically as soon as you committed the file in GitHub in your repo.
19. Click on the latest triggered build number
20. View the Console Output

The complete log of the steps involved in successful building of job are displayed.

Configure GitHub Webhook for Jenkins

1. Open GitHub
2. Navigate to Git Repo;
3. Navigate to Settings of repository
4. Click Webhook
5. Click Add Webhook
6. Enter Payload URL-http://<Public IP of Jenkins Server>:8080/github-webhook/
7. Click Save Webhook

Configure Jenkins for GitHub

1. Navigate to Manage Jenkins ->Open Configure System
2. In GitHub section, Select Add GitHub Server->GitHub Server
    a. Enter **Name**-Git Server
    b. Click on the Advanced… button present below the Add GitHub Server drop down
    c. Select Additional actions->Manage additional github actions->Select Convert login and password to token
    d. Select **from login and password** radio button
    e. Enter your GitHub login credentials
    f. Click **Create token credentials** button
    g. Scroll up and select the generated token in the **Credentials** field

Note: If by any chance, token is not populated in dropdown, Save the Configure System page and Open again, scroll down to see generated token is now available in Credentials field.

3. Click **Save** on Configure System

Configuring Jenkins server to be used with docker for deploying code

1.) SSH to the jenkins server
2.) Type command sudo su
3.) Type command adduser jenkins
4.) Type sudo passwd jenkins
5.) Type any password
6.) Retype password (Remember it or keep a note of it for future use)
7.) Type command sudo groupadd docker
8.) Type command sudo usermod -aG docker $USER
    Note: this will add root user to the docker group
9.) Type sudo usermod -aG docker jenkins

Note: this will add jenkins user to the docker group

10.) Type command vim /etc/sudoers
11.) Press I to insert

12.) Add line **jenkins ALL=(ALL:ALL) ALL** under section # User privilege specification root ALL=(ALL:ALL) ALL
13.) Press escape key :wq!
14.) Reboot ubuntu Jenkins server by typing command **shutdown -r now**
15.) Type docker version -f '{{.Server.Experimental}}' and ensure it returns false.

Configuring Jenkins front end to be used with docker for deploying code

1.) Navigate to **Manage Jenkins** in Jenkins opened in browser
2.) Open **Configure System**
3.) Scroll down till end.
4.) In section **Cloud**, Select **Add a new cloud**= docker
5.) Click **Docker cloud details** button
6.) Enter **Docker Host URI =**unix:///var/run/docker.sock
7.) Click **Test Connection**
8.) It should display something like "Version = 17.03.2-ce, API Version = 1.27"
9.) Click **Save**

Configuring Jenkins for publishing docker image into DockerHub

1.) Navigate to **Credentials** in Jenkins opened in browser
2.) Click "**global**" link in section Stores scoped to Jenkins
3.) Click **Add Credentials** link
4.) Select **Kind**= Secret text
5.) Enter **Secret** =<Type the docker hub password>
6.) Enter **ID** as "dockerpwd<yourname>"
Note: please keep value as mentioned. This is used in pipeline script.
7.) Enter **Description** as "dockerpwd<yourname>"
8.) Click **OK**

Configuring Jenkins for pulling and running docker image in Deployment server

1.) Navigate to **Credentials** in Jenkins opened in browser
2.) Click "**global**" link in section Stores scoped to Jenkins
3.) Click **Add Credentials** link
4.) Select **Kind**= SSH Username with private key
5.) Enter Username=root
Note: We entered root as username because docker is installed on deployment server as root user.

6.) For **Private key,** select **Enter directly** radio button**.**
7.) Enter **Key =<Private key of deployment server>**
8.) Enter **ID** as “dockerdeployserver2”
Note: please keep value as mentioned. This is used in pipeline script.
9.) Enter **Description** as “dockerdeployserver2”
10.) Click **OK**

Installing Plugins in Jenkins

1. In Jenkins, On Left Navigation links
2. Click Manage Jenkins.
3. Click Manage Plugins
4. Navigate to Available Tab
5. Search the required plugin using search text box provided above;
6. Select the checkbox;
7. Repeat the steps 6 and 7 for more plugins;
8. Click Install without restart button
9. After seeing the blue colored Success icon against the installed plugin name.
10. Click Restart Jenkins when Installation is complete and no jobs are running checkbox.

(Note:-In case you do not find any of the plugin in Available Tab, please check in Installed Tab.It may got installed as part of default plugins while installing Suggested plugins)

Jenkinsfile Script

```
node{

      stage('Checkout'){
         git
'https://github.com/rajnikhattarrsinha/java-
tomcat-maven-example'
      }

      stage('Build'){
         // Get maven home path and build
         def mvnHome =  tool name: 'Maven
3.5.4', type: 'maven'
         sh "${mvnHome}/bin/mvn package"
      }

     stage ('Test'){
         def mvnHome =  tool name: 'Maven
3.5.4', type: 'maven'
         sh "${mvnHome}/bin/mvn verify;
sleep 3"
```

```
      }

    stage('Build Docker Image'){
         sh 'docker build -t
rajnikhattarrsinha/javademo:2.0.0 .'
      }

      stage('Publish Docker Image'){

withCredentials([string(credentialsId:
'dockerpwd', variable: 'dockerPWD')]) {
              sh "docker login -u
rajnikhattarrsinha -p ${dockerPWD}"
         }
        sh 'docker push
rajnikhattarrsinha/javademo:2.0.0'
      }

   stage('Stop running containers'){
         def listContainer='sudo docker ps'
         def scriptRunner='sudo
./stopscript.sh'
         def stopContainer='sudo docker stop
$(docker ps -a -q)'
         sshagent(['dockerdeployserver2']) {
              sh "ssh -o
StrictHostKeyChecking=no ubuntu@35.168.86.36
${scriptRunner}"
         }
    }
   stage('Pull Docker Image and Deploy'){

            def dockerContainerName =
'javademo-$BUILD_NUMBER'
            def dockerRun= "sudo docker run
-p 8080:8080 -d --name
${dockerContainerName}
rajnikhattarrsinha/javademo:2.0.0"

sshagent(['dockerdeployserver2']) {
              sh "ssh -o
StrictHostKeyChecking=no ubuntu@35.168.86.36
${dockerRun}"
         }
   }
 }
```

Dockerfile Script

```
FROM tomcat:8
# Take the war and copy to
webapps of tomcat
COPY target/*.war
/usr/local/tomcat/webapps/
```