Lab: CICDPipeline-DeploymentToK8

Pre-Requisites: -

1. Git Repository with the code to be deployed should be available. The git repo to be used in training: https://github.com/rajnikhattarrsinha/Java-Demo-Application/ Please fork the repo in your GitHub account.
2. Configure GitHub Webhook for Jenkins on your forked repo. follow the steps mentioned in **Configure GitHub Webhook for Jenkins**
3. To publish docker image on DockerHub, Follow the settings mentioned in section **Configuring Jenkins for publishing docker image into DockerHub .**
4. **If your K8 cluster is on Azure**, then ha-proxy Public IP is to be used as Deployment Server IP in Step 4 and Step A below AND **If your K8 cluster is on AWS**, then master Public IP is to be used as Deployment Server IP in Step 4 and Step A below.
5. To deploy the application on K8 Cluster, Do the settings mentioned in section **Configuring Jenkins for connecting to K8 Server**

Steps to Follow at GitHub end:

1. Open **Jenkinsfile** in your root directory of your repo, click pencil icon to edit it, In **SCM Checkout** stage, Replace the https://github.com/rajnikhattarrsinha/Java-Demo-Application/ with url of your repo.
2. In stages **Build Docker Image** and **Publish Docker Image**, Replace **rajnikhattarrsinha** with **<your dockerhub username>** in all docker commands and Replace "**dockerpwd**" with "**dockerpwd<yourname>**"
3. In stage **Deploy**, replace "**k8pwd**" with "**k8pwd<yourname>**" and IP **104.211.177.6** with **your DEPLOYMENT SERVER IP** and commit the changes.
4. Open **pom.xml** from root directory, click pencil icon to edit it, replace line <finalName>**demopipeline_rajni1**</finalName> with <finalName>**demopipeline_<yourname>**</finalName> and commit the changes.
5. Open **deployment.yaml** from root directory, replace "**rajni-deployment**" with "**<your name>-deployment**" .
6. Replace "**rajnikhattarrsinha**" in image name "**rajnikhattarrsinha/javademoapp_#JOB-NAME#:#BUILD-NUMBER#**" with **<your dockerhub username>** and commit the changes.

Steps to Follow at Jenkins end:

7. Click **New Item** link on left panel
8. **Enter an Item name** like <yourname>_pipeline

Note: PLEASE DO NOT INCLUDE CAPITAL LETTER IN THE NAME IN STEP-8

9. Select **Pipeline**
10. Click OK

11. Select General Tab, **GitHub Project** as <repo to be deployed>
12. Under Build Triggers section, Select **GitHub hook trigger for GITScm polling** checkbox.
13. Under Pipeline section, Select **Definition** as “Pipe line script from SCM”
14. Select **SCM** as Git
15. Enter Repository URL as <repo to be used>
16. Keep **Script Path** as “Jenkinsfile”
17. Click Save
18. On Left Panel, Click on **Build now**
19. View the Pipeline being build and showing Stages.
20. Once all stages are successfully completed. Click on the latest triggered build number from left panel.
21. View the Console Output

Verifying the deployed application:

A. From your local system, ssh to k8 cluster workstation using the following command:
   a. **For AZURE Cluster,** ssh ubuntu@<IP of your k8 cluster>
   OR
   b. **For AWS Cluster,** ssh devops@<IP of your k8 cluster>
B. Enter yes when prompted
C. Enter Password when prompted.

For AZURE Cluster, the password is **Dev0p!!/1122** and **For AWS** Cluster, the password is **Dev0p$!!/**
Please copy from here only.

D. Run command **kubectl get svc** to know the **Port** associated with your deployment
E. Run command **kubectl get po -o wide** to know the **Node** where deployment is running.
F. Take the **Public IP** of the **Node** from your Node Machine details.
G. Open any browser
H. Type http://**<Public IP of the Node>**:<**Port>/<file name of the war given in pom.xml>**
I. Hit Enter;

Steps to Follow at Jenkins end again for automated CICD pipeline:

22. On GitHub end, edit pom.xml, Change the name of the war file name again of your choice and Commit the changes.
23. Open the Jenkins end, Open the Job Details page and observe that the build is triggered automatically as soon as you committed the file in GitHub in your repo.
24. View the Pipeline being build and showing Stages.
25. Once all stages are successfully completed. Click on the latest triggered build number from left panel.
26. View the Console Output

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

27. **I**n order to view the latest deployment, repeat the steps mentioned in section **Verifying the deployed application** above.

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 connecting to K8 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**= Secret text
For AZURE Cluster, the password is **Dev0p!!/1122** and **For AWS** Cluster, the password is **Dev0p\$!!/**
5.) Enter **Secret** =<Type the your k8 deployment server password>
NOTE: PLEASE USE THE PASSWORD provided above according to your cluster and copy from here only.
6.) Enter **ID** as "k8pwd<yourname>"
Note: please keep value as mentioned. This is used in pipeline script.
7.) Enter **Description** as "k8pwd<yourname>"
8.) Click **OK**

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