**Build\_Docker\_Jenkins\_Pipeline\_to\_Implement\_CI-CD\_Workflow\_steps**

**SECTION A**

**PART 1:-**

#Ubuntu machine setup with docker

Steps:-

1:-Setup Ubuntu-20.04 linux machine on physical machine or vm (i have choosed physical machine)

2:-Launch terminal and run below commands with sudo permission for installing required packages

such as docker , docker-compose with thier dependency

sudo apt update

sudo apt install docker.io

Output :-

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages were automatically installed and are no longer required:

chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi libllvm11 linux-image-5.8.0-63-generic linux-modules-5.8.0-63-generic linux-modules-extra-5.8.0-63-generic shim

Use 'sudo apt autoremove' to remove them.

The following additional packages will be installed:

bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan

Suggested packages:

ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs

git-mediawiki git-svn

The following NEW packages will be installed:

bridge-utils containerd docker.io git git-man liberror-perl pigz runc ubuntu-fan

0 upgraded, 9 newly installed, 0 to remove and 33 not upgraded.

Need to get 79.5 MB of archives.

After this operation, 397 MB of additional disk space will be used.

sudo apt install docker-compose

Output :-

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages were automatically installed and are no longer required:

chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi libllvm11 linux-image-5.8.0-63-generic linux-modules-5.8.0-63-generic linux-modules-extra-5.8.0-63-generic shim

Use 'sudo apt autoremove' to remove them.

The following additional packages will be installed:

python3-attr python3-cached-property python3-docker python3-dockerpty python3-docopt python3-importlib-metadata python3-jsonschema python3-more-itertools python3-pyrsistent python3-texttable

python3-websocket python3-zipp

Suggested packages:

python-attr-doc python-jsonschema-doc

The following NEW packages will be installed:

docker-compose python3-attr python3-cached-property python3-docker python3-dockerpty python3-docopt python3-importlib-metadata python3-jsonschema python3-more-itertools python3-pyrsistent

python3-texttable python3-websocket python3-zipp

0 upgraded, 13 newly installed, 0 to remove and 33 not upgraded.

Need to get 445 kB of archives.

After this operation, 2,574 kB of additional disk space will be used.

3 :- Add user to docker group so that user can do docker operation without sudo

sudo usermod -aG docker gk

Outout :-

gk@gk-ThinkPad-E15-Gen-2:~$ id gk

uid=1000(gk) gid=1000(gk) groups=1000(gk),4(adm),24(cdrom),27(sudo),29(audio),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),998(eveusb),135(docker)

4:- Reload docker group so that we dont require os restart

newgrp docker

5:-Create new folder for our project workspace and enter into it

mkdir /home/gk/Documents/Simplilearn\_Devops\_PG\_2021/DevOps\_Course2/projects/Deploy\_Angular\_Application\_in\_Docker\_Container\_Project/app\_service\_compose

cd /home/gk/Documents/Simplilearn\_Devops\_PG\_2021/DevOps\_Course2/projects/Deploy\_Angular\_Application\_in\_Docker\_Container\_Project/app\_service\_compose

**PART 2 (Jenkins in docker):-**

#Run Jenkins in docker with jenkins data folder persistent using docker volumes

#We will use prebuilt jenkins image created by myself

**vi docker-compose.yml**

version: '3.3'

services:

jenkins:

image: ganeshkale/jenkins\_docker:1.0

ports:

- '8080:8080'

- '50000:50000'

networks:

- jenkins

volumes:

- jenkins-data:/var/jenkins\_home

- jenkins-docker-certs:/certs/client:ro

restart: always

networks:

jenkins:

volumes:

jenkins-data:

external: true

jenkins-docker-certs:

external: true

save file with Esc:wq

**Note :-**

1:-port number “50000” is used for communication with windows slave agents

#Now spin up jenkins service

**docker-compose up -d**

#Goto firefox browser and enter url : **localhost:8080**

**PART 3 :-**

**#Now pull node latest docker image for creating and testing node sample application**

#pull node latest image from docker hub

docker pull node

#Run node image

docker run -itd --name nodejs –network host node

#enter into container shell

docker exec -it nodejs /bin/bash

#Create sample app

ng new sample-angular-app

cd sample-angular-app

#To test application

ng serve

#check in browser

localhost:4200

#Now take out sample angular code from container to current path

docker cp 534f198e7d42:sample-angular-app .

**#Additional steps for saving angular development docker image**

docker commit 534f198e7d42 ganeshkale/angulardev:1.0

**Output :-**

gk@gk-ThinkPad-E15-Gen-2:~/Documents/Simplilearn\_Devops\_PG\_2021/DevOps\_Course2/projects/Deploy\_Angular\_Application\_in\_Docker\_Container\_Project/app\_service\_compose$ docker commit 534f198e7d42 ganeshkale/angulardev:1.0

sha256:41babbce7ad03569f22d74df82e5b7585095787bfce21b5022491a386e2b31a5

gk@gk-ThinkPad-E15-Gen-2:~/Documents/Simplilearn\_Devops\_PG\_2021/DevOps\_Course2/projects/Deploy\_Angular\_Application\_in\_Docker\_Container\_Project/app\_service\_compose$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

ganeshkale/angulardev 1.0 41babbce7ad0 6 seconds ago 1.38GB

<none> <none> 59db36ddf89b About an hour ago 1.24GB

node latest c66552d59c4b 35 hours ago 907MB

nginx latest dd34e67e3371 2 days ago 133MB

-------------------------------------------------------------------------------------------------------------------

**PART 4**

Steps :-

1:- Create folder app\_service\_compose

mkdir app\_service\_compose

cd app\_service\_compose

#Create dockerfile for angular app image

**vi Dockerfile**

#Stage 1 compile source code

FROM node:latest as build

MAINTAINER "GANESH KALE"

LABEL VERSION="1.0"

WORKDIR /usr/local/webapp

#copy source code folder

COPY sample-angular-app /usr/local/webapp/

#will install package dependency mentioned in .json files

RUN npm install

#will compile app

RUN npm run build

#Stage 2 run compile source code from stage1

FROM nginx:latest

#using stage 1 compiled app and copy to nginx default html

COPY --from=build /usr/local/webapp/dist/sample-angular-app /usr/share/nginx/html

#eposing nginx port 80 to docker host machine

EXPOSE 80

save above file with esc:wq

**vi docker-compose.yml**

version: '3.3'

services:

web:

#will build Dockerfile from current directory

build: .

labels:

- "Angular app webhosting"

ports:

#Mapping exposed port to host network port

- "80:80"

networks:

#Separate bridge network

- webhost

#will restart always on os boottime or when docker service restarted

restart: always

networks:

webhost:

save above file with esc:wq

**PART 5:-**

**#Push the source code to github**

git init

git add -f docker-compose.yml Dockerfile sample-angular-app

git branch -M main

git remote add origin [git@github.com](mailto:git@github.com):ganeshjkale/angular\_docker.git

git push -u origin main

git checkout -b docker\_jenkins\_CICD

git push -u origin docker\_jenkins\_CICD

Note:-

1:-laptop ssh public key was added in my github account

#Project Github URL

**https://github.com/ganeshjkale/angular\_docker**

**Branch : docker\_jenkins\_CICD**

**PART 6:-**

**#Run docker compose image for testing purpose**

docker-compose up -d

localhost:80

#For stopping application run below command from same path where compose file of application are located

docker-compose down

**SECTION B**

**PART 1:-**

**#Create Jenkinsfile in vscode on ubuntu machine**

**#Jenkinsfile**

pipeline{

agent{

node {

label "ubuntu\_vm"

}

}

stages{

stage("Build Docker Image"){

steps{

echo "========executing Build Docker Image========"

sh 'docker build -t ganeshkale/sample\_angular:latest .'

}

post{

always{

echo "========always========"

}

success{

echo "========Build Docker Image executed successfully========"

}

failure{

echo "========Build Docker Image execution failed========"

}

}

}

stage("Push Docker Image"){

steps{

withCredentials([usernamePassword(credentialsId: 'gkdockerhub', passwordVariable: 'gkdockerhubPassword', usernameVariable: 'gkdockerhubUser')]) {

sh "docker login -u ${env.gkdockerhubUser} -p ${env.gkdockerhubPassword}"

sh 'docker push ganeshkale/sample\_angular:latest'

}

}

}

stage("Test Docker Image"){

steps{

sh "docker run -itd --rm --name gktest -v /var/run/docker.sock:/var/run/docker.sock ganeshkale/sample\_angular"

//dont use -i

sh 'docker exec -t gktest bash -c "curl localhost:80"'

sh "docker stop gktest"

}

}

}

post{

always{

echo "========always========"

}

success{

echo "========pipeline executed successfully ========"

}

failure{

echo "========pipeline execution failed========"

}

}

}

Add Jenkinsfile in git scm

Note:-

1:-used git functionality from vscode (shown in screenshot)

**PART 2:-**

**#Create New Pipeline Job in Jenkins**

1:-Goto jenkins url : localhost:8080

2:-New -> Pipeline -> docker\_jenkins\_pipeline\_project

3:-open step job

4:-Add pipeline through Git SCM

5:-Save

6:-Run Build Now

**PART3:-**

**#Automatic source code build on github push events**

1:-Added github secret text API in jenkins credential for managing hooks

2:-created webhook on github webhook section

[http://ganeshjkale1988sl0018.simplilearnlabs.com:42006/github-webhook](https://ganeshjkale1988sl0018.simplilearnlabs.com:42001/github-webhook)

**Note:-**

1:-only port 42006 was open on internet side and was mapped to ssh

2:-i have killed ssh service modified sshd config to port 22

3:-stopped running jenkins docker image and runned with port 42006:8080

4:-Now i am able to access jenkins over internet

**PART 4:-**

**#Jenkins CI/CD Build jobs console output :-**

**Console Output**

Started by user jenkins

Obtained Jenkinsfile from git https://github.com/ganeshjkale/angular\_docker.git

Running in Durability level: MAX\_SURVIVABILITY

[Pipeline] Start of Pipeline

[Pipeline] node

Running on ubuntu\_vm in /home/ganeshjkale1988/jenkins/workspace/docker\_jenkins\_pipeline\_project

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

No credentials specified

Fetching changes from the remote Git repository

Checking out Revision 8d85a81d7e025a5070b3ee90f0f15b8056aa575f (refs/remotes/origin/docker\_jenkins\_CICD)

Commit message: "syntax fixed"

> /usr/bin/git rev-parse --resolve-git-dir /home/ganeshjkale1988/jenkins/workspace/docker\_jenkins\_pipeline\_project/.git # timeout=10

> /usr/bin/git config remote.origin.url https://github.com/ganeshjkale/angular\_docker.git # timeout=10

Fetching upstream changes from https://github.com/ganeshjkale/angular\_docker.git

> /usr/bin/git --version # timeout=10

> git --version # 'git version 2.7.4'

> /usr/bin/git fetch --tags --progress https://github.com/ganeshjkale/angular\_docker.git +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> /usr/bin/git rev-parse refs/remotes/origin/docker\_jenkins\_CICD^{commit} # timeout=10

> /usr/bin/git config core.sparsecheckout # timeout=10

> /usr/bin/git checkout -f 8d85a81d7e025a5070b3ee90f0f15b8056aa575f # timeout=10

> /usr/bin/git rev-list --no-walk 471c77f68d26a8b49d4f9d2ebc113d65246c88b0 # timeout=10

[Pipeline] }

[Pipeline] // stage

[Pipeline] withEnv

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Build Docker Image)

[Pipeline] echo

========executing Build Docker Image========

[Pipeline] sh

+ docker build -t ganeshkale/sample\_angular:latest .

Sending build context to Docker daemon 5.759MB

Step 1/10 : FROM node:latest as build

---> c66552d59c4b

Step 2/10 : MAINTAINER "GANESH KALE"

---> Using cache

---> 5a2130db1f09

Step 3/10 : LABEL VERSION="1.0"

---> Using cache

---> 0a97444b6477

Step 4/10 : WORKDIR /usr/local/webapp

---> Using cache

---> 355a3f39eb1e

Step 5/10 : COPY sample-angular-app /usr/local/webapp/

---> Using cache

---> e3aa7d0887e3

Step 6/10 : RUN npm install

---> Using cache

---> fdba61f2e819

Step 7/10 : RUN npm run build

---> Using cache

---> 433044409829

Step 8/10 : FROM nginx:latest

---> dd34e67e3371

Step 9/10 : COPY --from=build /usr/local/webapp/dist/sample-angular-app /usr/share/nginx/html

---> Using cache

---> b9a6cf0c6a52

Step 10/10 : EXPOSE 80

---> Using cache

---> 2526eae510c9

Successfully built 2526eae510c9

Successfully tagged ganeshkale/sample\_angular:latest

Post stage

[Pipeline] echo

========always========

[Pipeline] echo

========Build Docker Image executed successfully========

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Push Docker Image)

[Pipeline] withCredentials

Masking supported pattern matches of $gkdockerhubPassword

[Pipeline] {

[Pipeline] sh

Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.

Affected argument(s) used the following variable(s): [gkdockerhubPassword]

See https://jenkins.io/redirect/groovy-string-interpolation for details.

+ docker login -u ganeshkale -p \*\*\*\*

WARNING! Using --password via the CLI is insecure. Use --password-stdin.

WARNING! Your password will be stored unencrypted in /home/ganeshjkale1988/.docker/config.json.

Configure a credential helper to remove this warning. See

https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

[Pipeline] sh

+ docker push ganeshkale/sample\_angular:latest

The push refers to repository [docker.io/ganeshkale/sample\_angular]

302a9477947d: Preparing

fb04ab8effa8: Preparing

8f736d52032f: Preparing

009f1d338b57: Preparing

678bbd796838: Preparing

d1279c519351: Preparing

f68ef921efae: Preparing

d1279c519351: Waiting

f68ef921efae: Waiting

8f736d52032f: Layer already exists

009f1d338b57: Layer already exists

fb04ab8effa8: Layer already exists

302a9477947d: Layer already exists

678bbd796838: Layer already exists

f68ef921efae: Layer already exists

d1279c519351: Layer already exists

latest: digest: sha256:a70350873452b55dfe5308bab2222f1606d4cb2b7d898b6fde75f12252eda172 size: 1779

[Pipeline] }

[Pipeline] // withCredentials

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Test Docker Image)

[Pipeline] sh

+ docker run -itd --rm --name gktest -v /var/run/docker.sock:/var/run/docker.sock ganeshkale/sample\_angular

7e111216fab30423416a26f6c99846678ad0d4fd9d3144d303778dcce9bea0e6

[Pipeline] sh

+ docker exec -t gktest bash -c curl localhost:80

<!DOCTYPE html><html lang="en"><head>

<meta charset="utf-8">

<title>SampleAngularApp</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<link rel="stylesheet" href="styles.31d6cfe0d16ae931b73c.css"></head>

<body>

<app-root></app-root>

<script src="runtime.c60d61fd264d501c7ac8.js" defer></script><script src="polyfills.a3e1fd6068c4e016fb2a.js" defer></script><script src="main.73d1f9ba7edf232ed210.js" defer></script>

</body></html>

[Pipeline] sh

+ docker stop gktest

gktest

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Declarative: Post Actions)

[Pipeline] echo

========always========

[Pipeline] echo

========pipeline executed successfully ========

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline

Finished: SUCCESS