**Real-time deployment on a Kubernetes cluster using Jenkins CI/CD**

1. Configure two Ubuntu servers on the AWS environment named Jenkins Server and Ansible Server, and install the necessary software.

On Jenkins Server:

Please follow the installation steps provided on the official website:

<https://www.jenkins.io/doc/book/installing/linux/>

Make sure to install Java before installing Jenkins.

$ sudo apt update

$ sudo apt install openjdk-11-jre

$ java -version

### Weekly release

A new release is produced weekly to deliver bug fixes and features to users and plugin developers. It can be installed from the [debian apt repository](https://pkg.jenkins.io/debian/).

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc **>** /dev/null

echo deb **[**signed-by**=**/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list **>** /dev/null

sudo apt-get update

sudo apt-get install jenkins

After installing the Jenkins server, access it via SSH using the following URL:

<Public IP of Jenkins Server>:8080 and then configure it.

Create a user during the installation of the Jenkins server and access the Jenkins server via the newly created user.

The first step is to install the SSH Agent Plugin.

Click on "Manage Jenkins," then select "Manage Plugins," and click on "Available." Install SSH Agent without restarting, then restart Jenkins.

Login to Ansible Server and install the Ansible.

<https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html>

**Installing Ansible on Ubuntu**

Ubuntu builds are available in a PPA here.

To configure the PPA on your system and install Ansible run these commands:

$ sudo apt update

$ sudo apt install software-properties-common

$ sudo add-apt-repository --yes --update ppa:ansible/ansible

$ sudo apt install ansible

$ sudo ansible –version

Also, Install docker package into Ansible server.

Follow the installation steps from the official site:

<https://docs.docker.com/engine/install/ubuntu/>

Install using the apt repository

#### **Set up the repository**

1. Update the apt package index and install packages to allow apt to use a repository over HTTPS:
2. $ sudo apt-get update

$ sudo apt-get install ca-certificates curl gnupg

1. Add Docker’s official GPG key:
2. $ sudo install -m 0755 -d /etc/apt/keyrings
3. $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

$ sudo chmod a+r /etc/apt/keyrings/docker.gpg

Use the following command to set up the repository:

$ echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

#### **Install Docker Engine**

1. Update the apt package index:
2. $ sudo apt-get update
3. Install Docker Engine, containerd, and Docker Compose.
   * Latest
   * Specific version

To install the latest version, run:

$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

2. On a Windows machine, search for "download Git Bash for Windows" on Google and install the Git Bash software.

3. Please log in to your GitHub account and create a repository named "kubernetes\_projects".

4. "Please return to the Windows machine and create a directory on the D drive named 'kubernetes\_projects.'

Once created, navigate to it by typing 'D:\kubernetes\_projects' in the file explorer. Right-click on the directory and select 'Git Bash Here' to open a new terminal."

5. Now, clone your GitHub repository to your local machine.

# git clone <url of your github repository>

# git config --global --list

# git config --global user.name "Shikhar Verma"

# git config --global user.email "shikhardevops@gmail.com"

6. Your developers will create a Dockerfile or application code for your project according to the requirements.

vi Dockerfile

FROM centos:7

MAINTAINER shikhardevops@gmail.com

RUN yum install -y httpd \

zip\

unzip

ADD https://www.free-css.com/assets/files/free-css-templates/download/page254/photogenic.zip /var/www/html/

WORKDIR /var/www/html/

RUN unzip photogenic.zip

RUN cp -rvf photogenic/\* .

RUN rm -rf photogenic photogenic.zip

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]

EXPOSE 80

7. Now, we need to push this new file into the GitHub repository "kubernetes\_projects."

There are three stages for git:

- Working area/untracked area

- Staging area/tracked area

- Local repository

# git status

# git add Dockerfile

# git commit -m "Added a new file Dockerfile"

# git status

# git push origin main

8. Login to the Jenkins Server

New items, enter an item name pipeline\_demo

and choose pipeline option only and click on.

9. Check out the project from GitHub to the Jenkins server and write a simple pipeline as shown below:

node {

stage ('Git Checkout') {

git branch: 'main', url: 'https://github.com/Shikhar82/Kubernetes\_Project.git'

}

}

10. Now configure a webhook so that if any new file is created or modified in your GitHub repository, your job will be automatically triggered.

11. Click on your job, then select 'configure'. Under 'build triggers', choose.

GitHub hook trigger for GITScm polling (Select this option)

12. Now, go back to your GitHub repository and click on "Settings". Then, click on "Webhooks" on the left-hand side.

And click on Add webhooks

and fill the below details:

- Payload URL

url of jenkins server with webhook as shown below:

http://65.0.30.44:8080/github-webhook/

- Content type:

application/json

- Secret

Please return to the Jenkins server and click on your profile name, which should be "Shikhar Verma".

Then, click on "Configure" and create a new API token. Copy the generated token and paste it here. Please ensure that the token is pasted correctly.

13. Please return to Git Bash terminal on your Windows machine.

and edit your Dockerfile using vi editor and just add a new port like 22

# git add Dockerfile

# git commit -m "Added a new port"

# git push origin main

A new job will be initiated immediately, which will check out the kubernetes\_projects into the Jenkins server with a modified Dockerfile.

You can cross-verify this by logging into the Jenkins server and going to /var/lib/jenkins/workspace/pipeline\_demo/.

Sending the Dockerfile to Ansible using SSH Agent.

14. Click on the Jenkins pipeline job name "pipeline-demo" and then click on "configure". Add the following steps to the code.

Use sshagent to communicate from jenkins server to ansible and here you have to provide the details of user "ubuntu" with private key through with your jenkins server able to access ansible server.

Click on pipeline syntax:

sshagent: SSH Agent

Click on Add and select Jenkins

Under kind, choose SSH Username with private key

ID - ansible\_server

Description - ansible\_server

Username - ubuntu

Tick private key and add private key

And then click on ADD.

And then choose ubuntu(ansible\_server)

and click on generate pipeline script

sshagent(['ansible\_server']) {

// some block

}

===========

stage ('Sending docker file to Ansible Server over ssh') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo1/Dockerfile ubuntu@172.31.44.57:/home/ubuntu'

}

}

where, 172.31.44.57 is the ip of the ansible server.

===========

Final pipeline

node {

stage ('Git Checkout') {

git branch: 'main', url: 'https://github.com/Shikhar82/Kubernetes\_Project.git'

}

stage ('Sending docker file to Ansible Server over ssh') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo1/\* ubuntu@172.31.44.57:/home/ubuntu'

}

}

}

==================

==================

Building & tagging docker Image

15. Please edit the pipeline and add the following code below to build the Docker image using the Dockerfile.

stage ('Docker Build Image') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image build -t $JOB\_NAME:v1.$BUILD\_ID -f /home/ubuntu/Dockerfile .'

}

}

===========

node {

stage ('Git Checkout') {

git branch: 'main', url: 'https://github.com/Shikhar82/Kubernetes\_Project.git'

}

stage ('Sending docker file to Ansible Server over ssh') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo1/\* ubuntu@172.31.44.57:/home/ubuntu'

}

}

stage ('Docker Build Image') {

sshagent(['ansible\_demo']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image build -t $JOB\_NAME:v1.$BUILD\_ID -f /home/ubuntu/Dockerfile .'

}

}

}

=======================

16. Tag the image as required before transferring it to Docker Hub.

Login to docker hub and get its userid, for my dockerhub, my userid is shikhardocker/$JOB\_NAME

and tag the image with the same name as your dockerhub user id

Log in to Docker Hub and obtain your user ID. For my Docker Hub account, my user ID is "shikhardocker".

Tag the image with the same name as your Docker Hub user ID/$JOB\_NAME.

Here, $JOB\_NAME is pipeline-demo1

# sudo docker image tag pipeline-demo1 shikhardocker/pipeline-demo1

# sudo docker image ls

pipeline-demo1 v1.12 5005b02caad8 31 hours ago 457MB

shikhardocker/pipeline-demo1 v1.12 5005b02caad8 31 hours ago 457MB

shikhardocker/pipeline-demo1 latest 5005b02caad8 31 hours ago 457MB

node {

stage ('Git Checkout') {

git branch: 'main', url: 'https://github.com/Shikhar82/Kubernetes\_Project.git'

}

stage ('Sending docker file to Ansible Server over ssh') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo1/\* ubuntu@172.31.44.57:/home/ubuntu'

}

}

stage ('Docker Build Image') {

sshagent(['ansible\_demo']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image build -t $JOB\_NAME:v1.$BUILD\_ID -f /home/ubuntu/Dockerfile .'

}

}

stage ('Docker Image Tagging') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:latest'

}

}

}

Please log in to the Ansible server and verify the presence of Docker images.

# sudo docker image ls

pipeline-demo1 v1.12 5005b02caad8 31 hours ago 457MB

shikhardocker/pipeline-demo1 v1.12 5005b02caad8 31 hours ago 457MB

shikhardocker/pipeline-demo1 latest 5005b02caad8 31 hours ago 457MB

Complete Declarative CICD Pipelines

17. Edit your pipeline job and below

=============

stage ('Docker Image Tagging') {

sshagent(['ansible\_demo']) {

sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker login -u shikhardocker -p #####"

}

}

This is not the correct way to pass the password in plain text, it should be encrypted for that you have to use withCredentials: Bind credentials to variables

=============

18. Click on the Pipeline Syntax and search for withCredentials: Bind credentials to variables

Click on ADD and select secret text

Variable - give any name dockerhub\_pipeline\_passwd

Click on ADD and select Jenkins

Under kind, choose secret text

ID - dockerhub\_pipeline\_passwd

Description - dockerhub\_pipeline\_passwd

Secret - Give your docker hub passwd

Then click on generate pipeline script

withCredentials([string(credentialsId: 'dockerhub\_pipeline\_newpasswd', variable: 'dockerhub\_pipeline\_newpasswd')]) {

Final script:

stage ('Push Docker Image to Docker Hub') {

sshagent(['ansible\_demo']) {

withCredentials([string(credentialsId: 'dockerhub\_pipeline\_passwd', variable: 'dockerhub\_pipeline\_passwd')]) {

sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker login -u shikhardocker -p $dockerhub\_pipeline\_passwd"

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:latest'

}

}

========

Final Script

node {

stage ('Git Checkout') {

git branch: 'main', url: 'https://github.com/Shikhar82/Kubernetes\_Project.git'

}

stage ('Sending docker file to Ansible Server over ssh') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo1/\* ubuntu@172.31.44.57:/home/ubuntu'

}

}

stage ('Docker Build Image') {

sshagent(['ansible\_demo']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image build -t $JOB\_NAME:v1.$BUILD\_ID -f /home/ubuntu/Dockerfile .'

}

}

stage ('Docker Image Tagging') {

sshagent(['ansible\_server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:latest'

}

}

stage ('Push Docker Image to Docker Hub') {

sshagent(['ansible\_demo']) {

withCredentials([string(credentialsId: 'dockerhub\_pipeline\_passwd', variable: 'dockerhub\_pipeline\_passwd')]) {

sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker login -u shikhardocker -p $dockerhub\_pipeline\_passwd"

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:latest'

}

}

}

}

=========

=========

19. Create a Kubernetes cluster in an AWS environment using t2.medium. The cluster type should be Minikube.

==========

apt-get update

apt -y install docker.io

curl -LO https://dl.k8s.io/release/v1.24.0/bin/linux/amd64/kubectl

install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client

r=https://api.github.com/repos/kubernetes/minikube/releases

curl -LO $(curl -s $r | grep -o 'http.\*download/v.\*beta.\*/minikube-linux-amd64' | head -n1)

sudo install minikube-linux-amd64 /usr/local/bin/minikube

minikube start --vm-driver=none

apt install conntrack

minikube start --vm-driver=none

===========

20. Create a passwordless authentication between the Ansible server and the Kubernetes server, as well as the Jenkins server, for the Ubuntu user.

Login to kubernetes server:

- set the root passwd

# vi /etc/ssh/sshd\_config

PermitRootLogin yes

PasswordAuthentication yes

service sshd restart

Now login to Ansible server and generate ssh key

# ssh-keygen -t rsa

public key will be created in /root/.ssh/id\_rsa.pub ==> this file we need to copy to kubernetes server

# ssh-copy-id -i root@<ip of kubernetes\_server>

give try

ssh root@<ip of kubernetes\_server> ==> this time, it won't ask for any passwd.

# ansible --version

# Install ansible

# ansible --version

login as ubuntu user:

21. Log in to the Ansible server and edit the file /etc/ansible/hosts and add the entry of kubernetes server.

# /etc/ansible/hosts

[node]

<IP of the kubernetes server>

esc wq!

# ansible -m ping node

# ansible -m ping <ip of the kubernetes server>

22. Please return to the Jenkins server.

Go to /var/lib/jenkins/workspace/pipeline-demo1/ and you will find all the files.

Deployment.yml Dockerfile Service.yml ansible.yml

Please ensure that all files are present on your Windows machine.

During stage 1, check out the project. It will copy all the files from the Windows machine to the Jenkins server.

First, create a new SSH agent named "kubernetes-server" to allow Jenkins access to the Kubernetes server.

stage ('Copy the files jenkins server to kubernetes server') {

sshagent(['kubernetes-server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.38.177'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Deployment.yml ubuntu@172.31.38.177:/home/ubuntu'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Service.yml ubuntu@172.31.38.177:/home/ubuntu'

}

}

22. Sending the Ansible playbook to the Ansible Server via SSH.

stage ('Sending Ansible playbook to Ansible Server over ssh') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/ansible.yml ubuntu@172.31.44.57:/home/ubuntu'

}

}

So final pipeline

==========

node {

stage ('Git Checkout') {

git branch: 'main', url: 'https://github.com/Shikhar82/Kubernetes\_Project.git'

}

stage ('Sending docker file to Ansible Server over ssh') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Dockerfile ubuntu@172.31.44.57:/home/ubuntu'

}

}

stage ('Docker Build Image') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image build -t $JOB\_NAME:v1.$BUILD\_ID -f /home/ubuntu/Dockerfile .'

}

}

stage ('Docker Image Tagging') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:latest'

}

}

stage ('Push Docker Image to Docker Hub') {

sshagent(['ansible\_newserver']) {

withCredentials([string(credentialsId: 'dockerhub\_pipeline\_newpasswd', variable: 'dockerhub\_pipeline\_newpasswd')]) {

sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker login -u shikhardocker -p $dockerhub\_pipeline\_newpasswd"

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:latest'

}

}

stage ('Copy the files jenkins server to kubernetes server') {

sshagent(['kubernetes-server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.38.177'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Deployment.yml ubuntu@172.31.38.177:/home/ubuntu'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Service.yml ubuntu@172.31.38.177:/home/ubuntu'

}

}

stage ('Sending Ansible playbook to Ansible Server over ssh') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/ansible.yml ubuntu@172.31.44.57:/home/ubuntu'

}

}

}

}

==============

23. Kubernetes deployment using ansible

stage ('Kubernetes deployment using Ansible') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 ansible-playbook ansible.yml'

========

=======

Final Script

node {

stage ('Git Checkout') {

git branch: 'main', url: 'https://github.com/Shikhar82/Kubernetes\_Project.git'

}

stage ('Sending docker file to Ansible Server over ssh') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Dockerfile ubuntu@172.31.44.57:/home/ubuntu'

}

}

stage ('Docker Build Image') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image build -t $JOB\_NAME:v1.$BUILD\_ID -f /home/ubuntu/Dockerfile .'

}

}

stage ('Docker Image Tagging') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image tag $JOB\_NAME:v1.$BUILD\_ID shikhardocker/$JOB\_NAME:latest'

}

}

stage ('Push Docker Image to Docker Hub') {

sshagent(['ansible\_newserver']) {

withCredentials([string(credentialsId: 'dockerhub\_pipeline\_newpasswd', variable: 'dockerhub\_pipeline\_newpasswd')]) {

sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker login -u shikhardocker -p $dockerhub\_pipeline\_newpasswd"

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:v1.$BUILD\_ID'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 sudo docker image push shikhardocker/$JOB\_NAME:latest'

}

}

stage ('Copy the files jenkins server to kubernetes server') {

sshagent(['kubernetes-server']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.38.177'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Deployment.yml ubuntu@172.31.38.177:/home/ubuntu'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/Service.yml ubuntu@172.31.38.177:/home/ubuntu'

}

}

stage ('Sending Ansible playbook to Ansible Server over ssh') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57'

sh 'scp /var/lib/jenkins/workspace/pipeline-demo2/ansible.yml ubuntu@172.31.44.57:/home/ubuntu'

}

}

stage ('Kubernetes deployment using Ansible') {

sshagent(['ansible\_newserver']) {

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 cd /home/ubuntu'

sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.44.57 ansible-playbook ansible.yml'

}

}

}

}

==========

==========

FROM centos:latest

MAINTAINER shikhardevops@gmail.com

RUN yum install -y httpd \

zip\

unzip

ADD https://www.free-css.com/assets/files/free-css-templates/download/page254/photogenic.zip /var/www/html/

WORKDIR /var/www/html/

RUN unzip photogenic.zip

RUN cp -rvf photogenic/\* .

RUN rm -rf photogenic photogenic.zip

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]

EXPOSE 80

https://www.free-css.com/assets/files/free-css-templates/download/page254/photogenic.zip

https://www.free-css.com/assets/files/free-css-templates/download/page36/football-card.zip

https://www.free-css.com/assets/files/free-css-templates/download/page36/football-card.zip

https://github.com/vikash-kumar01/Kubernetes\_Project/blob/master/Docker\_Project

# cat Deployment.yml

kind: Deployment

apiVersion: apps/v1

metadata:

name: mrdevops

spec:

replicas: 2

selector: # tells the controller which pods to watch/belong to

matchLabels:

app: mrdevops

template:

metadata:

labels:

app: mrdevops

spec:

containers:

- name: mrdevops

image: shikhardocker/pipeline-demo

imagePullPolicy: Always

ports:

- containerPort: 80

# cat Service.yml

kind: Service

apiVersion: v1

metadata:

name: mrdevops

labels:

app: mrdevops

spec:

ports:

- port: 8080

targetPort: 80

nodePort: 31200

selector:

app: mrdevops

type: LoadBalancer

# cat ansible.yml

- hosts: node

become: true

tasks:

- name: delete old deployment

command: kubectl delete -f /home/ubuntu/Deployment.yml

- name: delete old service

command: kubectl delete -f /home/ubuntu/Service.yml

- name: create new deployment

command: kubectl apply -f /home/ubuntu/Deployment.yml

- name: create new service

command: kubectl apply -f /home/ubuntu/Service.yml

============

=============