**DevOps**

**Lab1--Install Docker n create docker image**

Create vm with ubuntu

sudosu root

cd /etc/gdm3

vi custom.conf🡺AllowRoot =True

cd ..

cd pam.d

vi gwd-password🡺comment auth required (3rd line)

apt update

#apt-get install docker -y

#systemctl start and enable docker

create the docker account

#docker login

🡪username and password

#docker pull ubuntu ----install image from docker

#docker image ls ----list of docker image

#docker run -ti ubuntu ----create container

#docker attach container id

#docker inspect container id ----show ip of container

#dockerps -a----show the list of containers

Q.Create two container centos and ubuntu run python code on both1.swapping and factorial

yum or apt update

yum install -y python3

pyhon –V

vi swap.py

pyhon swap.py

**Lab2-- Port mapping and binding and host website**

#docker ps -a

# vi /var/www/html/index.html

# mkdir website

#cd website cd

#vi index.html

docker run - -name \_\_\_\_\_ -d -p 8080:80 - -n/w \_\_\_\_\_ -v /root/website/:/usr/share/nginx/html nginx

#docker run - -name

web2 -p 8080:80 -d -v /var/www/html/:/usr/share/nginx/html/ nginx

Q.Deploy one html application and create docker container

🡺docker container

Apt-get install nano curl apache2

Create index.html

Service apache2 start

Service apache2 enable

Curl http://localhost

**Lab-3 networking**

#docker network ls

#docker network create n/wname

#docker network connect

#docker network disconnect

#docker network inspect

#docker network create - -subnet=\_\_\_\_\_\_\_\_\_\_\_ - -ip range =\_\_\_\_\_\_\_\_ networkname

#docker run - -name \_\_\_\_\_ -d -p 8080:80 - -n/w \_\_\_\_\_ -v /root/website/:/usr/share/nginx/html nginx

**Lab-4 Docker Compose**

#docker compose –version

#apt-get install docker-compose

#nano hpcsa.yaml

version: ‘3.9’

Services:

Web1:

image:nginx

Port:

-9000:80

volumes:

-/root/website/:/usr/share/nginx/html/

Web2:

image:httpd

ports:

* 9010:80

Volumes:

* /root/var/www/html/index.html

C1:

image:centos

tty:true

stdin\_open:true

#docker -compose -f hpcsa.yaml down

#docker -compose -f hpcsa.yaml up

#docker -compose -f hpcsa

On browser http:/ip:9000

Note:

Nginx🡺/usr/share/nginx/html nginx

Html🡺/usr/local/apache2/htdocs

Q.Create 3 container using docker-compose

1.nginx—html

2.http—prn and name

3.centos—python file

**Lab-5 Create Docker file to deploy python application**

docker run -ti ubuntu 🡺automatically gone in shell

apt update

apt-get install python nano -y

vi sample.py

nano Dockerfile🡺

FROM ubuntu

RUN apt update -y

RUN apt install python3 -y

RUN mkdir /app

COPY prog\_1.py /app/

CMD ["python3", "/app/prog\_1.py"]

save

//FROM ==> Pull the image

RUN ==> Run the command inside the container

COPY ==> copy the file or data from host machine to container path

CMD ==> Execute or Run the program/application

Application or Program should be save in same directory//

docker build -t app:1.0 .

docker run - -name p1 -ti app:1.0

it show the ouput of python file

For html

#mkdir web1

#nanoDockerfile🡺

FROM ubuntu

COPY /usr/share/nginx/html

save

#vi index.html

#docker build -t myapp:1.0 .

#docker run - -name lab1 -d -p 9000:80 myapp:1.0

on browser ip of machine:9000

#docker save filename > filename.tar🡪conver image into tar file

**Lab-6 Installation of kubernets**

Create 3 vm master and 2 nodes

On master as well as nodes

#Do the perquisites

#vi /etc/fsatd🡺swapoff -a

#modprobe overlay

#modprobebr\_netfilter

#tee /etc/sysctl.d/kubernetes.conf<<EOF🡺

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

EOF

#sysctl --system

#tee /etc/yum.repos.d/kubernetes.repo<<EOF🡺

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

EOF

#yum clean all && yum -y makecache

#yum -y install epel-release git curl wgetkubeletkubeadmkubectl--disableexcludes=Kubernetes

#systemctl start kubelet

#systemctl enable kubelet

#curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable.repo<https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/devel:kubic:libcontainers:stable.repo>

#curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable:cri-o:$VERSION.repo https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/devel:kubic:libcontainers:stable:cri-o:$VERSION.repo

Install CRI-O

#yum remove docker-ce docker-ce-cli containerd.io

#yum install cri-o -y

# Start and enable Service

#systemctl daemon-reload

#systemctl start crio

#systemctl enable crio

Only on master

#lsmod | grep br\_netfilter

#kubeadm config images pull

#kubeadminit --pod-network-cidr=10.85.0.0/16 --upload-certs --control-plane-endpoint=master

🡺tocken will create(last 3 line)

that token we have to save in one txt file as join-token

scp that token file to nodes

#mkdir -p $HOME/.kube

#sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

#sudo chown $(id -u):$(id -g) $HOME/.kube/config

On node

./join-token

#kubectl create -f<https://raw.githubusercontent.com/projectcalico/calico/v3.24.5/manifests/tigera-operator.yaml>

#kubectl create -f <https://raw.githubusercontent.com/projectcalico/calico/v3.24.5/manifests/custom-resources.yaml>

On master

Deployment of multiple images

#kubectl create deployment ngnix-image - -image=nginx - -replicas=2

#kubectl get deployment

#kubectl expose deployment nginx-image - -port=9000 - -target-port=80 - -name=web1

#kubectl get services

Types of services

1.Cluster IP

2.NodePort

3.LoadBalancer

4.DNS

#kubectl expose deployment nginx-image - -port=9000 - -target-port=80 - -name=web2

- -type=NodePort

#kubectl expose deployment nginx-image - -port=9000 - -target-port=80 - -name=web3

- -type=NodePort

#kubectl get services

#kubectl get deployment -o wide

#kubectl expose deployment nginx-image - -port=9000 - -target-port=80 - -name=web4

- -type=LoadBalancer

#kubectl delete service web1

#kubectl get deployment -o wide

#kubectldeletedeployment ngnix-image

#kubectl get pods

#kubectl get pods -o wide

#kubectl get services -o wide

* node port given

on browser nachineip:pod port

Depolyment using yaml file

in yaml file indentation matter

**Lab-7 Git and Github**

Create two ubuntu machine node

one already created wirh docker

#apt update

#apt-get install git

#mkdirtmp

#cd /tmp

#curl -o git.tar.gz <https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.26.2.tar.gz>

#tar -zxf git.tar.gz

#cd git-\*

#make prefix=/usr/local all

#make prefix=/usr/local install

#exec bash

#git config --global user.name "Your Name"

#git config --global user.email[youremail@domain.com](mailto:youremail@domain.com)

#touch file\_{1..3}

#echo “content1” > file1

#echo “content2” > file2

#echo “content3” > file3

On Githtb

login to github

Create new repo

name🡪public🡪readme🡪createrepo

code🡪copylink of http

#git remote add origin paste link

#git pull origin master

#git add -A

#git commit -m “Commit 1”

#git config --global user.name "Your Name"

#git config --global user.email[youremail@domain.com](mailto:youremail@domain.com)

on github

setting🡪developer setting🡪personal access token🡪token🡪generate new token🡪no expiration🡪click repo on delete repo🡪copy token

ghp\_xqNMBQeuYF9jMd1G02RA9QCbZWbih50MKMNa

#git push

🡺username:

password: paste token

Create with change in branch

on github

Create repo with master

#mkdir branch

#cd branch

#git remote add origin “link http”

#git pull origin master

#git add -A

#git commit -m “commit 1”

#git push origin master

create branch

#git branchbranchname

#git checkout branchname

#git push origin branchname

merge files in two braches

#git checkout branchname == to shit into branch

#git merge branchname ==from shift

#git add -A

Q. Create one repository in github and clone it into your local machine

Q. Create 1 html file and 2 txt file push it into github remote repository.

Git clone

**Lab-8 Jenkins**

Server1

1.Docker

2.Git

3.Jenkins🡪

1.fetch data

2.Deploy docker container on Jenkins

3.Transfer the data without password

Server 3

Developer

1.Git

2.Create application

Server 2

Git and github

1.Fetch Data

On Server 3

1.Root login

2.Install Git

3.Create one html file in git

4.Create one repo on gihub and add html file in it

On sever 1

#apt update

#apt-get install git docker

#apt -get install open-jdk

#wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |sudogpg --dearmor -o /usr/share/keyrings/jenkins.gpg

#sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

#apt-get install Jenkins

#systemctl start jenkins.service

#systemctl status Jenkins

On Browser

<http://server1>ip :8080

#vi /var/lib/jenkins/secrets/initial🡺AdminPassword

password jenkins

**1.Fetch the data from github**

New item🡪Free project🡪Create

Check github project🡪copy github link without .git

repository🡪copylinkgithub

Build Trigger🡪build periodically🡺\* 5 \* \* \*

Apply and Save

Jen-1 build

**2.Deploy the docker container**

Docker install and enable docker

#cd var/lib/Jenkins/workspace/Jen1

#vi /etc/suders🡺Authentication --- jenkinsALL=(ALL) NOPASSWD: ALL

#nanoDockerfile

🡺FROM nginx

COPY index.html/usr/share/nginx/html

On browser

New item🡪Build Trigger🡪Jen-1—trigger stable

Build Steps🡺

cd /var/lib/Jenkins/workspace/Jen-1

sudo docker build -t myapp:1.0

sudo docker -rm -f web1

sudo docker run - -name web1 -p 9000:80 -d myapp:1.0

Apply and Save

Build Jen-2

3.Deploy the image and transfer the image

i)Convert image into tar file

ii)Scptransfer the tar file

iii)with help of sshuntar file

iv)Create container

on sever 1 and 2

#apt-get install openshh -y

on server 2

#vi /etc/ssh/sshd\_config🡺

Authentication

PermitRootLogin yes

uncomment🡪PasswordAuthentication yes

on server 1

#su -s /bin/bash

#ssh-keygen

#ssh-copy-id root@\_\_\_\_

On browser

add Execute shell

cd /var/lib/Jenkins/workspace/jen-1

sudo docker save myapp:1.0 > myapp.tar

scp myapp.tar root@\_\_\_\_:/root/

ssh root@\_\_\_ “docker load < myapp.tar”

ssh root@\_\_\_\_ “docker rm -f web1”

ssh root@\_\_\_\_ “docker run - -name web1 -d -p 9050:80”

on browser ip of base machine :9050

Gradle

Q1. Create two container on docker

i. Container of Centos 🡪run python application(Swapping of two numbers)

ii. Container of Uubuntu🡪install python and run python application (Factorial of no)

Q3. Create Docker Container using docker Compose

i. Create Container using nginx

ii. Create using http (Deploy html file PRN and Username)

iii. Conatiner using centos image (Deploy python application)

Q4. Install Kubernets and use yaml file

Q5 Create 1 repository in github and clone it into local machine

Q6 Create html file and two text file push it into remote repository.

**AWS**

Q1. Create instance of ubuntu and install apache and post the webpage

Q2. VPC

1.Create VPC of n/w 10.0.0.0/16 then create subnet1 10.10.1.0/24 and subnet 2 10.10.2.0/24 and attach it and vpc to access the EC2 instance

2. Create VPC 192.168.0./16 using subnet 192.168.10.0/16 and attach to EC2 and install tomcat and host webpage of grpname, prn , roll no with help of port 80

Q3. Create the html file on local machine and host it on s3 bucket

Q4. IAM

1.Create Policy for developer

i.read write for S3 and lambda

ii. read for EC2 EKS ECR

2. Create Policy for Quality Analyst

i.read write for S3 except delete

ii. read for EC2, EKS, ECR, lambda

3. Create the Security group

i. Create policy read write for ec2 except delete and attach to iamuser1

ii. Login as user1 and create ec2 and install tomcat and host the web page

iii. Create policy with read foe ec2 attach it to iam user2 and try to delete that instance using iamuser2

IAM Role

Q1. Create s3 bucket 🡺 1.prnlab for pdf and 2. namelab for jpg

Create two lambda function and attach IAM role to function s3 bucket

Create SNS and Destination