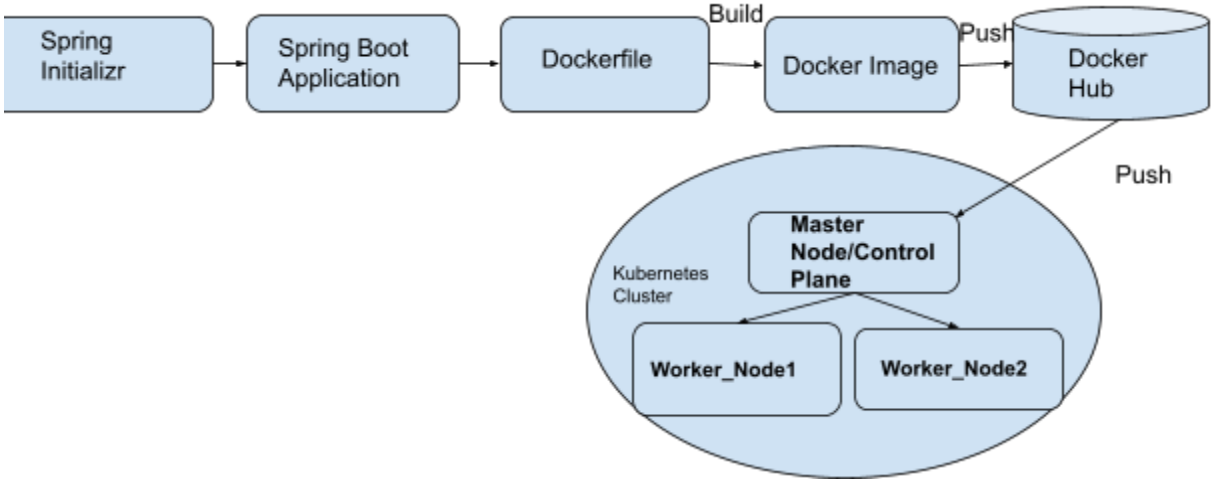
Kubernetes_Project

Requirement : An organization XYZ Private Limited has recently transformed their IT application from Monolithic to Microservice Architecture. Now they have been struggling with deployment in such a complex infrastructure and inconsistency across the system. The organization has hired you to help them with simplifying their deployment process by containerizing their applications. They are using spring boot to develop their microservices. So we need to deploy our application in a container.

Here is the step wise step procedure to implement the solution.



Here we are going to deploy our application in our kubernetes cluster. Application Development to Image push in the Docker Hub we already did. Kindly check how to do that from the below reference.
https://www.linkedin.com/feed/update/urn:li:activity:7103891715752673280/

Step 1 : Create a Kubernetes Cluster.

Step 2: Pull the Image to Master node

Step 3: Write Deployment,service file and Deploy the Application.

Step 1: Create a kubernetes cluster.

Reference:
https://kubeguide.medium.com/how-to-set-up-kubernetes-cluster-on-ubuntu-20-04-a366514d40a9

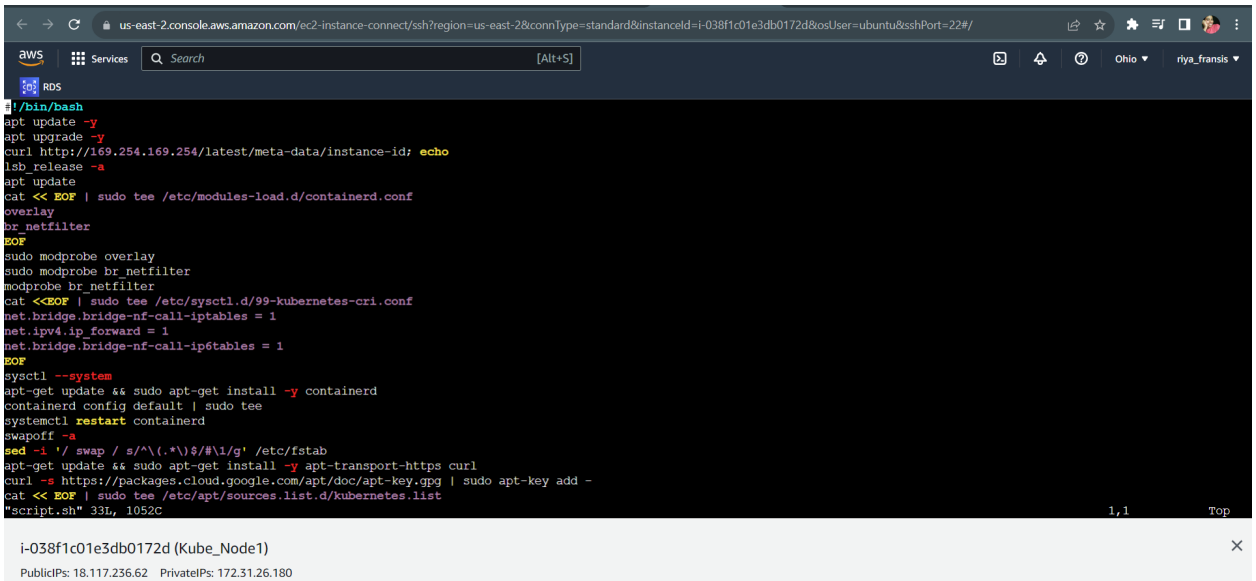
For Creating the cluster we need servers which have at least 2 GB ram. So I choose t2.medium(which is not under the free tier). Enabled all traffic in the security group for these 3 servers and gave 30 GB storage.

In both master and worker nodes create the below script file and run that file. It will install all required configurations on all the nodes.

sudo vi script.sh

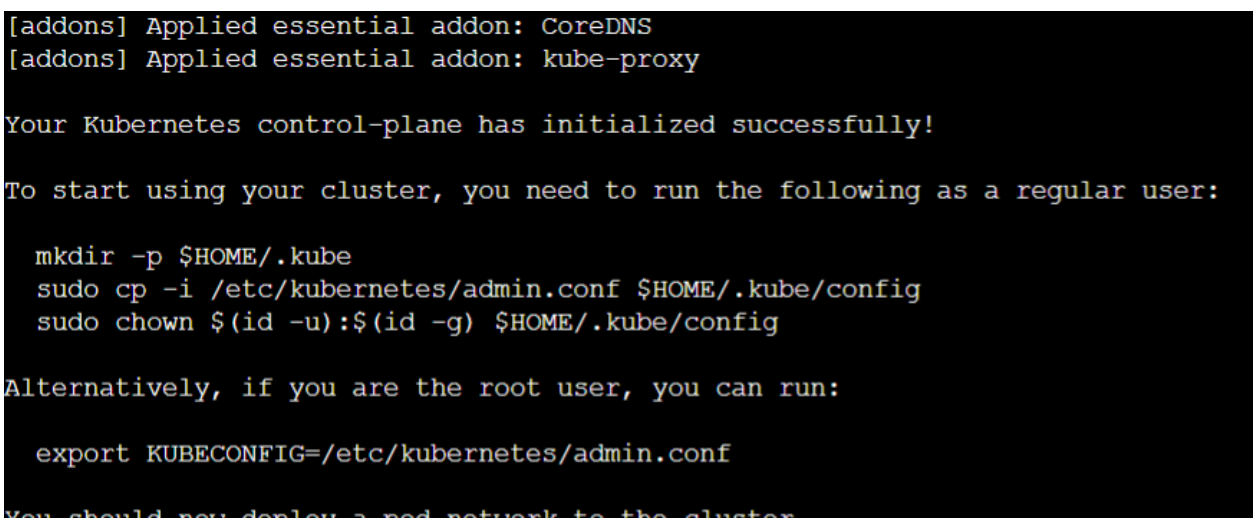
```
#!/bin/bash
apt update -y
apt upgrade -y
curl http://169.254.169.254/latest/meta-data/instance-id; echo
lsb_release -a
apt update
cat << EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
modprobe br_netfilter
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
sysctl --system
apt-get update && sudo apt-get install -y containerd
containerd config default | sudo tee
systemctl restart containerd
swapoff -a
sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
apt-get update && sudo apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add -
cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
sudo apt-get install -y kubelet=1.24.0-00 kubeadm=1.24.0-00
kubectl=1.24.0-00
sudo apt-mark hold kubelet kubeadm kubectl
```

chmod +x script.sh
sudo su //make sure that script is running as a root user
./script.sh



Initialize Kubernetes Master Node

```
sudo kubeadm init --pod-network-cidr 192.168.0.0/16 --kubernetes-version 1.24.0
```



Run the following command to start the cluster

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

Create a Pod Network in the master node using the following command

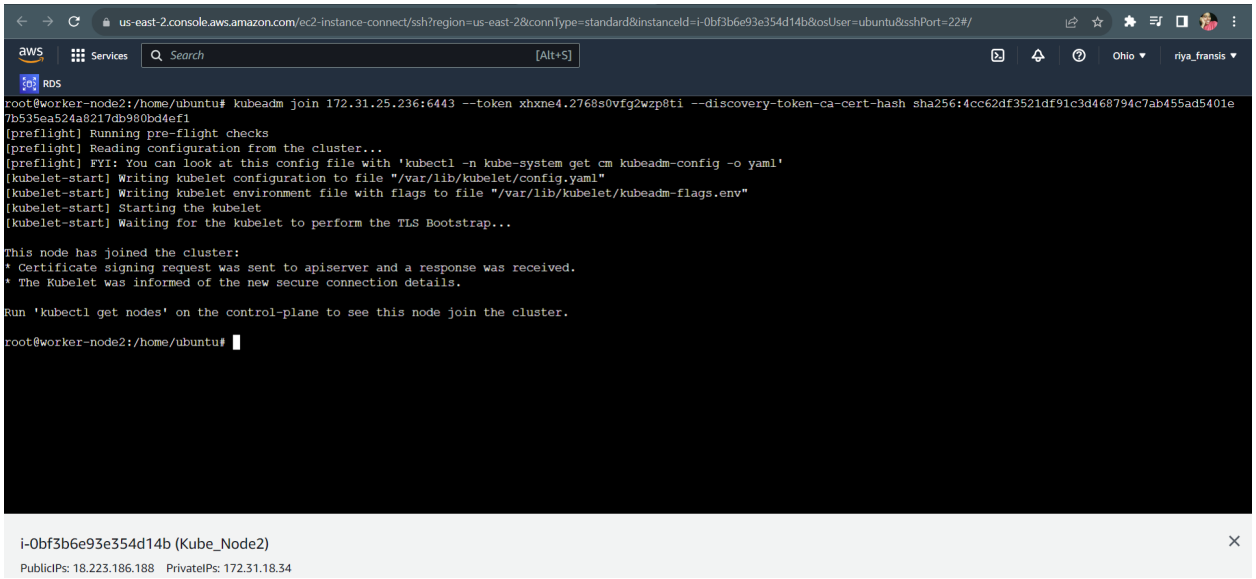
```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

```
root@master-node:/home/ubuntu# kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
```

```
us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0d8d9bdfd33fe9005&osUser=ubuntu&region=us-east-2&sshPort=22#/
aws  Services  Search  [Alt+S]  Ohio  riya_fransis
RDS
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
root@master-node:/home/ubuntu#
```

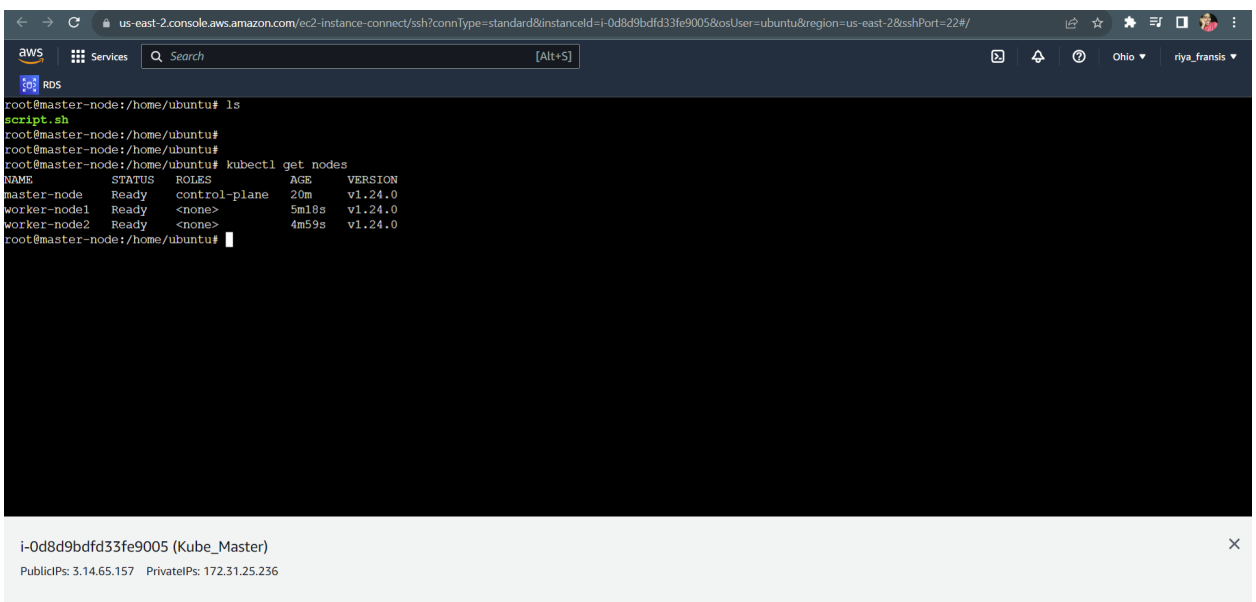
Join all the worker nodes to the kubernetes cluster. The below command will give the joining command, which we need to run in each worker node to connect with the cluster.

```
kubeadm token create --print-join-command
```



Check all the nodes are connected and they are in the ready state

kubectl get nodes



kubectl cluster-info

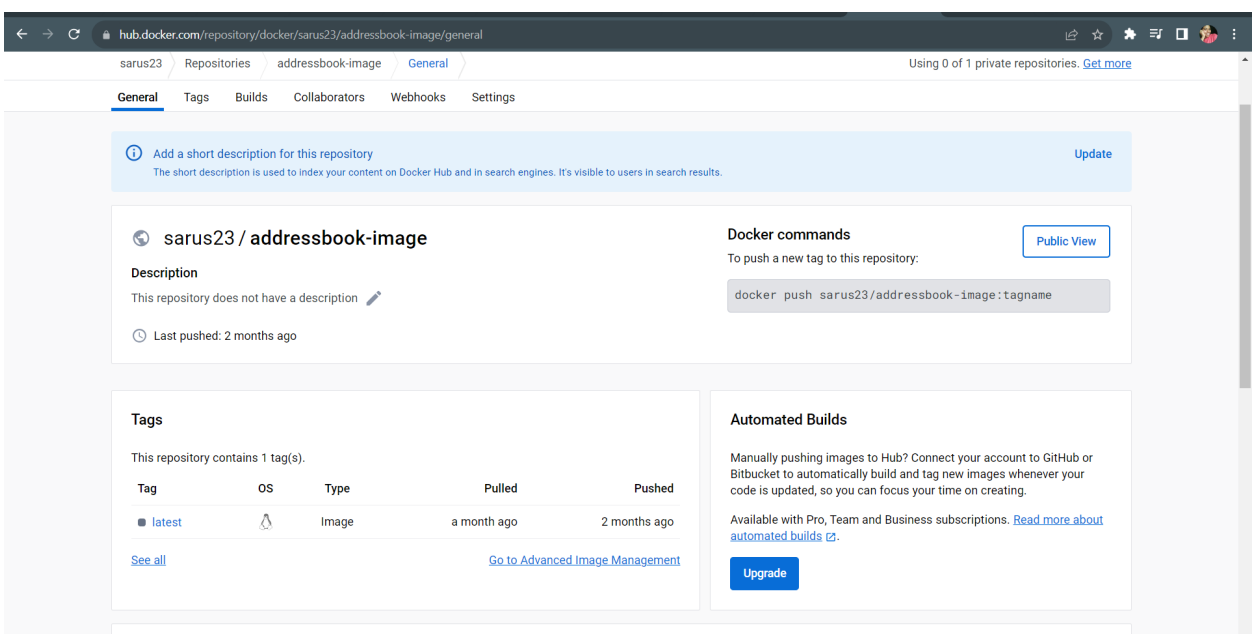
```
ubuntu@master-node:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
master-node   Ready    control-plane   15m   v1.24.0
ubuntu@master-node:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
master-node   Ready    control-plane   16m   v1.24.0
ubuntu@master-node:~$ kubectl get nodes
NAME           STATUS   ROLES           AGE     VERSION
master-node    Ready    control-plane   19m     v1.24.0
worker-node1   Ready    <none>          2m38s   v1.24.0
worker-node2   Ready    <none>          96s     v1.24.0
ubuntu@master-node:~$ kubectl cluster-info
Kubernetes control plane is running at https://172.31.25.236:6443
CoreDNS is running at https://172.31.25.236:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
ubuntu@master-node:~$
```

Now the Kubernetes cluster is created.

Step 2: Pull the Image to Master node

This time I am going to deploy my address book application which is available in my Docker hub account. First install docker in the master node using the following command

Install docker
sudo apt install -y docker.io



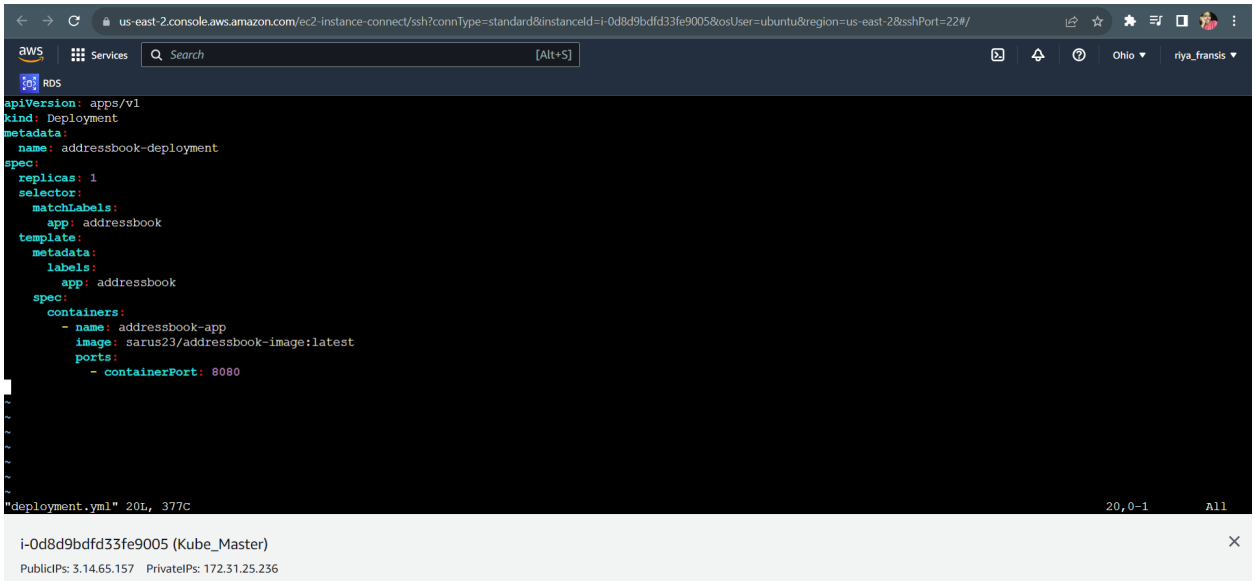
Docker login
Give the username and password. After successful login, pull the image to our master node.
docker pull sarus23/addressbook-image:latest

```
root@master-node:/home/ubuntu# sudo docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
root@master-node:/home/ubuntu# docker pull sarus23/addressbook-image:latest
latest: Pulling from sarus23/addressbook-image
9d19ee268e0d: Pull complete
f2b566cb887b: Pull complete
b375e6654ef5: Pull complete
19452d1108a6: Pull complete
b82f37793aff: Pull complete
194515f21e10: Pull complete
fe49462914ba: Pull complete
b4ac37f59bbf: Pull complete
Digest: sha256:72f0286ef5224cc00e21e12b0acab306e1c4e97ab983262f320732638fe3633c
Status: Downloaded newer image for sarus23/addressbook-image:latest
docker.io/sarus23/addressbook-image:latest
root@master-node:/home/ubuntu# sudo docker images
REPOSITORY                  TAG       IMAGE ID       CREATED       SIZE
sarus23/addressbook-image   latest    da9a06e56ff1   7 weeks ago   491MB
root@master-node:/home/ubuntu#
```

Step 3: Write Deployment,service file and Deploy the Application.

Create deployment.yml (sudo vi deployment.yml)

```yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: addressbook-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: addressbook
  template:
    metadata:
      labels:
        app: addressbook
    spec:
      containers:
        - name: addressbook-app
          image: sarus23/addressbook-image:latest
          ports:
            - containerPort: 8080
```



Create service.yml (sudo vi service.yml)

```
apiVersion: v1
kind: Service
metadata:
  name: addressbook-service
spec:
  selector:
    app: addressbook
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

```
us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0d8d9bdfd33fe9005&osUser=ubuntu&region=us-east-2&sshPort=22#/
aws  Services  Search [Alt+S]  Ohio  riya_fransis
RDS
apiVersion: v1
kind: Service
metadata:
  name: addressbook-service
spec:
  selector:
    app: addressbook
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

Then apply both deployment file and service file

kubectl apply -f deployment.yml
kubectl apply -f service.yml

```
ubuntu@master-node:~$ kubectl get nodes
NAME           STATUS   ROLES           AGE    VERSION
master-node    Ready    control-plane   121m   v1.24.0
worker-node1   Ready    <none>          104m   v1.24.0
worker-node2   Ready    <none>          103m   v1.24.0
ubuntu@master-node:~$ ls
deployment.yml  script.sh  service.yml
ubuntu@master-node:~$ kubectl apply -f deployment.yml
deployment.apps/addressbook-deployment created
ubuntu@master-node:~$
```

Checking the status of pods
kubectl get pods

```
service/addressbook-service created
root@master-node:/home/ubuntu# kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
master-node   Ready    control-plane   28m   v1.24.0
slave-node    Ready    <none>          25m   v1.24.0
slave-node2   Ready    <none>          25m   v1.24.0
root@master-node:/home/ubuntu# kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
addressbook-deployment-77ff5879f-r9hb8    1/1     Running   0          95s
root@master-node:/home/ubuntu#
```

Scale up the addressbook application up to 10 instances.
kubectl scale deployment/addressbook-deployment --replicas=10

```
root@master-node:/home/ubuntu# kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
addressbook-deployment-77ff5879f-r9hb8    1/1     Running   0          95s
root@master-node:/home/ubuntu# kubectl scale deployment/addressbook-deployment --replicas=10
deployment.apps/addressbook-deployment scaled
root@master-node:/home/ubuntu#
root@master-node:/home/ubuntu# kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
addressbook-deployment-77ff5879f-2jfhf    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-49xzs    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-bq29l    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-hhbjc    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-jnh4c    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-k82wp    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-qfkvj    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-qqxpd    1/1     Running   0          3m5s
addressbook-deployment-77ff5879f-r9hb8    1/1     Running   0          6m59s
addressbook-deployment-77ff5879f-s55nx    1/1     Running   0          3m5s
root@master-node:/home/ubuntu#
```

Get the service ports by running below command

kubectl get svc addressbook-service

```
us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-00d12e408f9c8689d&osUser=ubuntu&region=us-east-2&sshPort=22#/
aws  Services  Search [Alt+S]  Ohio  riya_fransis
RDS
root@master-node:/home/ubuntu# kubectl get svc addressbook-service
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
addressbook-service  LoadBalancer   10.97.107.47   <pending>     80:32528/TCP   149m
root@master-node:/home/ubuntu#
root@master-node:/home/ubuntu#
root@master-node:/home/ubuntu#
root@master-node:/home/ubuntu#
root@master-node:/home/ubuntu#
root@master-node:/home/ubuntu#
root@master-node:/home/ubuntu# kubectl describe svc addressbook-service
Name:                     addressbook-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=addressbook
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.107.47
IPs:                      10.97.107.47
Port:                     <unset>  80/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  32528/TCP
Endpoints:                192.168.182.65:8080,192.168.182.66:8080,192.168.182.67:8080 + 7 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
root@master-node:/home/ubuntu#
```

It will create a external ip for load balancer. We can access the application in this ip address. Kubernetes application deployment is successful.