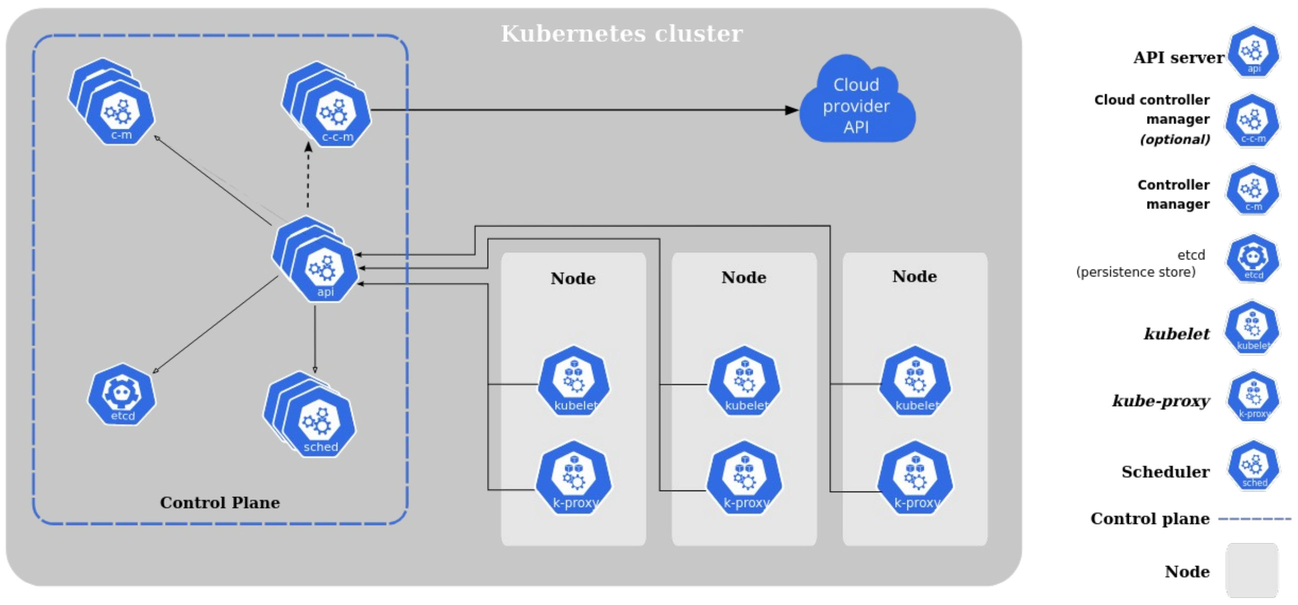
Kubernetes Installation and Deployment

Objectives

Kubernetes Installation and Deployment

<u>1.Introduction:</u> Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

Kubernetes Cluster: A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.
Basic architecture of a cluster with different components.



<u>2.Installation Prerequisite:</u>

- ✔ A compatible Linux host with 2 GB or more of RAM & 2 CPUs or more.
- ✔ Full network connectivity between all machines in the cluster (public or private network)
- ✔ Docker in the machine.

Different Installation Toolbox:

- ◆ Kubeadm → Bare Metal Installation.
- ◆ MiniKube → Virtualized Environment for Kubernetes
- ◆ Kops → Kubernetes on AWS
- ◆ Kubernetes on GCP → Kubernetes running on Google Cloud Platform
- ◆ AKS → Azure Kubernetes Services on Azure platform.

Installation Packages: Below packages need to install on the all machines.

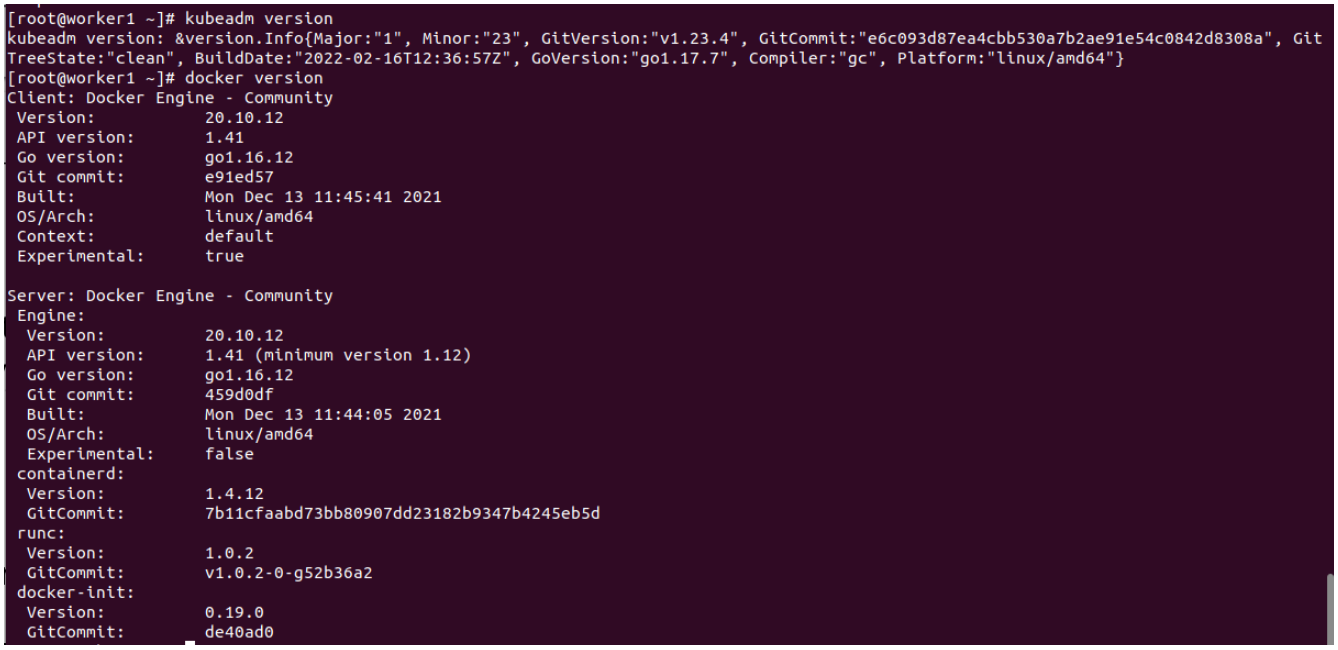
- ◆ Kudeadm → the command to bootstrap the cluster.
- ◆ Kubelet → the component that runs on all of the machines in the cluster and does things like starting pods and containers.
- ◆ Kubectl → the command line utility to talk to the cluster.

Note: kubeadm will not install or manage kubelet or kubectl.

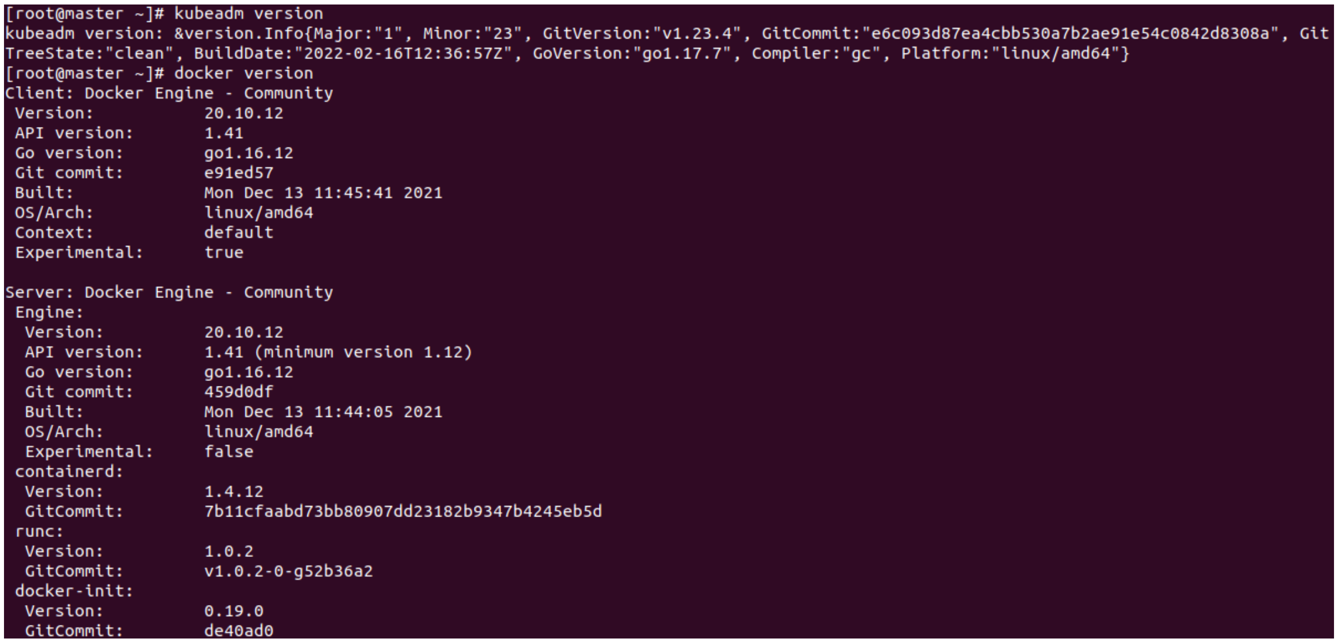
3.Installation on Centos 7: In this setup one virtual machine will prepare as master and another machine will paly the role of slave. First install docker and Kubernetes on both machines.

```
#turn of the swap
$swapoff -a
# Comment out swap line in fstab so that it remains disabled after reboot
$vi /etc/fstab

#common utilities install
$yum install -y yum-utils device-mapper-persistent-data lvm2

#it is preferred to disable the firewall
$sudo systemctl disable --now firewalld
#if needs to active the firewall than add the following ports in the firewall on Master
$sudo firewall-cmd --add-port={6443,2379-2380,10250,10251,10252,5473,179,5473}/tcp --permanent
$sudo firewall-cmd --add-port={4789,8285,8472}/udp --permanent
$sudo firewall-cmd –reload
#for worker nodes
sudo firewall-cmd --add-port={10250,30000-32767,5473,179,5473}/tcp --permanent
sudo firewall-cmd --add-port={4789,8285,8472}/udp --permanent
sudo firewall-cmd –reload

#Set the net.bridge.bridge-nf-call-iptables to ‘1’ in sysctl config file. This ensures that
packets are properly processed by IP tables during filtering and port forwarding
$cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

$sysctl --system

#Enable br_netfilter Kernel Module
$sudo modprobe overlay
$sudo modprobe br_netfilter

$sudo 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

$sudo sysctl –system

#in this installation Docker run will used. So need to install the docker run time.
# Install packages
$sudo yum install -y yum-utils device-mapper-persistent-data lvm2
$sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
$sudo yum install docker-ce docker-ce-cli containerd.io

# Create required directories
$sudo mkdir /etc/docker
$sudo mkdir -p /etc/systemd/system/docker.service.d

# Create daemon json config file
$sudo tee /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ]
}
EOF

# Start and enable Services
$sudo systemctl daemon-reload
$sudo systemctl restart docker
$sudo systemctl enable docker

#add the repository for the Kubernetes.
$cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch
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
exclude=kubelet kubeadm kubectl
EOF

# Set SELinux in permissive mode (effectively disabling it)
$sudo setenforce 0
$sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

#install kubelet kubeadm kubectl
$sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

#check the version
$sudo kubeadm version
```



```
[root@worker1 ~]# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.4", GitCommit:"e6c093d87ea4cbb530a7b2ae91e54c0842d8308a", Git
TreeState:"clean", BuildDate:"2022-02-16T12:36:57Z", GoVersion:"go1.17.7", Compiler:"gc", Platform:"linux/amd64"}
[root@worker1 ~]# docker version
Client: Docker Engine - Community
 Version:           20.10.12
 API version:       1.41
 Go version:        go1.16.12
 Git commit:        e91ed57
 Built:             Mon Dec 13 11:45:41 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.12
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.16.12
  Git commit:       459d0df
  Built:            Mon Dec 13 11:44:05 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.12
  GitCommit:        7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc:
  Version:          1.0.2
  GitCommit:        v1.0.2-0-g52b36a2
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```



```
[root@master ~]# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.4", GitCommit:"e6c093d87ea4cbb530a7b2ae91e54c0842d8308a", Git
TreeState:"clean", BuildDate:"2022-02-16T12:36:57Z", GoVersion:"go1.17.7", Compiler:"gc", Platform:"linux/amd64"}
[root@master ~]# docker version
Client: Docker Engine - Community
 Version:           20.10.12
 API version:       1.41
 Go version:        go1.16.12
 Git commit:        e91ed57
 Built:             Mon Dec 13 11:45:41 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.12
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.16.12
  Git commit:       459d0df
  Built:            Mon Dec 13 11:44:05 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.12
  GitCommit:        7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc:
  Version:          1.0.2
  GitCommit:        v1.0.2-0-g52b36a2
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

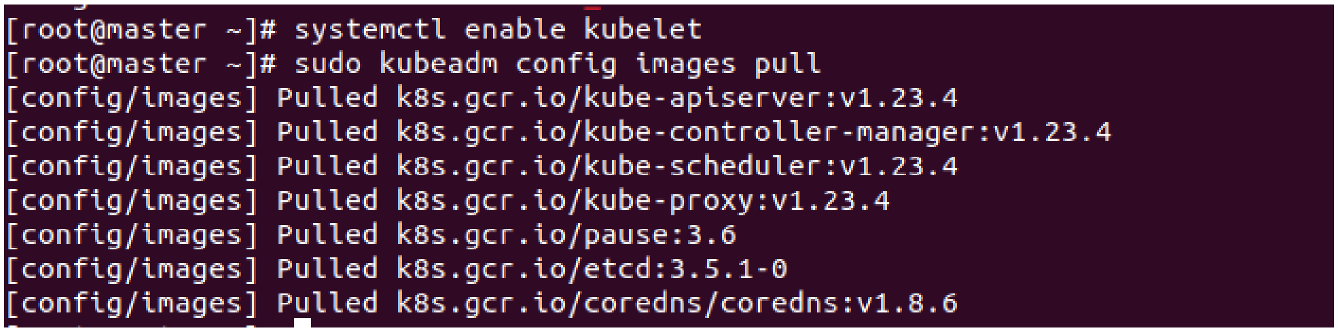
Note: After installation reboot the both machines

#start and enable the kubelet service
$sudo systemctl start kubelet && sudo systemctl enable kubelet

#Change the cgroup-driver
$sudo docker info | grep -i cgroup

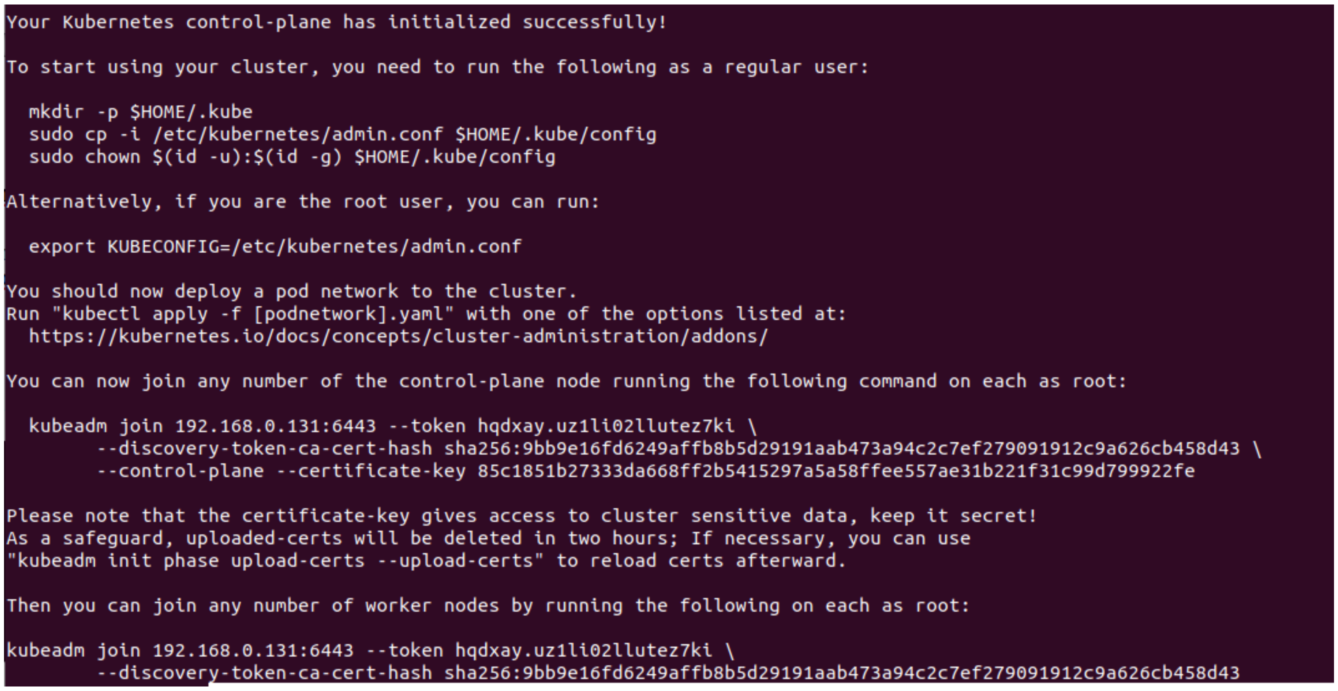
#need to make sure the docker-ce and kubernetes are using same 'cgroup'
sed -i 's/cgroup-driver=systemd/cgroup-driver=cgroupfs/g' /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

4.Creating the cluster:
#pull the container images so that it could easy crate the cluster during offline.
$sudo kubeadm config images pull



```
[root@master ~]# systemctl enable kubelet
[root@master ~]# sudo kubeadm config images pull
[config/images] Pulled k8s.gcr.io/kube-apiserver:v1.23.4
[config/images] Pulled k8s.gcr.io/kube-controller-manager:v1.23.4
[config/images] Pulled k8s.gcr.io/kube-scheduler:v1.23.4
[config/images] Pulled k8s.gcr.io/kube-proxy:v1.23.4
[config/images] Pulled k8s.gcr.io/pause:3.6
[config/images] Pulled k8s.gcr.io/etcd:3.5.1-0
[config/images] Pulled k8s.gcr.io/coredns/coredns:v1.8.6
```

#initialize the master
$sudo kubeadm init   --pod-network-cidr=10.10.0.0/16   --upload-certs   --control-plane-endpoint=192.168.0.131

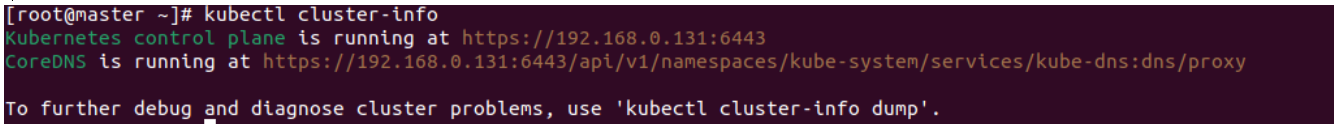


```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of the control-plane node running the following command on each as root:

  kubeadm join 192.168.0.131:6443 --token hqdxay.uz1li02llutez7ki \
        --discovery-token-ca-cert-hash sha256:9bb9e16fd6249affb8b5d29191aab473a94c2c7ef279091912c9a626cb458d43 \
        --control-plane --certificate-key 85c1851b27333da668ff2b5415297a5a58ffee557ae31b221f31c99d799922fe

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.0.131:6443 --token hqdxay.uz1li02llutez7ki \
        --discovery-token-ca-cert-hash sha256:9bb9e16fd6249affb8b5d29191aab473a94c2c7ef279091912c9a626cb458d43
```

#Configure kubectl using commands in the output
$mkdir -p $HOME/.kube
$sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$sudo chown $(id -u):$(id -g) $HOME/.kube/config

#check the cluster status

$kubectl cluster-info



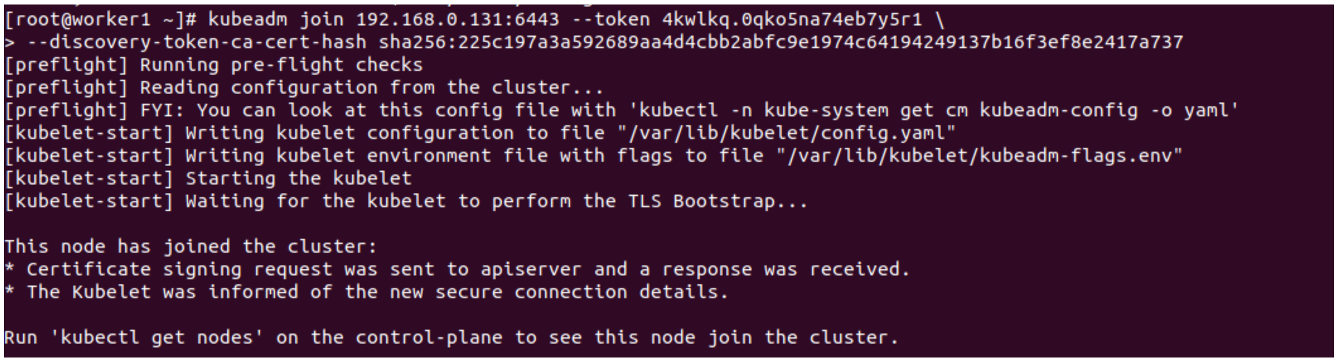
```
[root@master ~]# kubectl cluster-info
Kubernetes control plane is running at https://192.168.0.131:6443
CoreDNS is running at https://192.168.0.131:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

#deploy the flannel network to the kubernetes cluster using the kubectl command

$kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

#joining the worker node with master

$kubeadm join 192.168.0.131:6443 --token 4kwlkq.0qko5na74eb7y5r1 --discovery-token-ca-cert-hash sha256:225c197a3a592689aa4d4cbb2abfc9e1974c64194249137b16f3ef8e2417a737



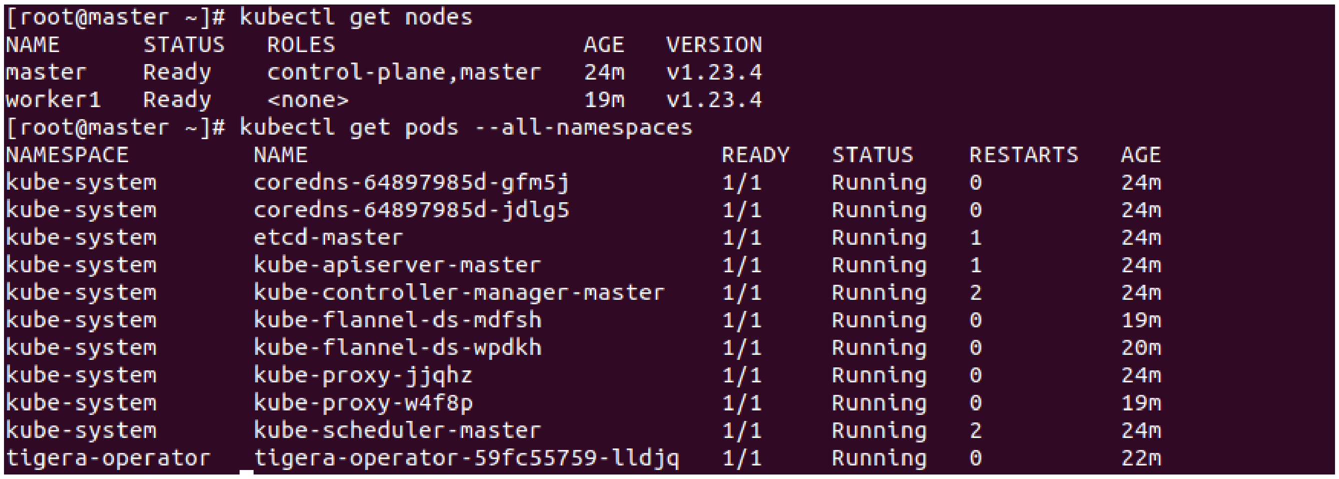
```
[root@worker1 ~]# kubeadm join 192.168.0.131:6443 --token 4kwlkq.0qko5na74eb7y5r1 \
> --discovery-token-ca-cert-hash sha256:225c197a3a592689aa4d4cbb2abfc9e1974c64194249137b16f3ef8e2417a737
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

#to check all nodes status

$kubectl get nodes

#to check all podes

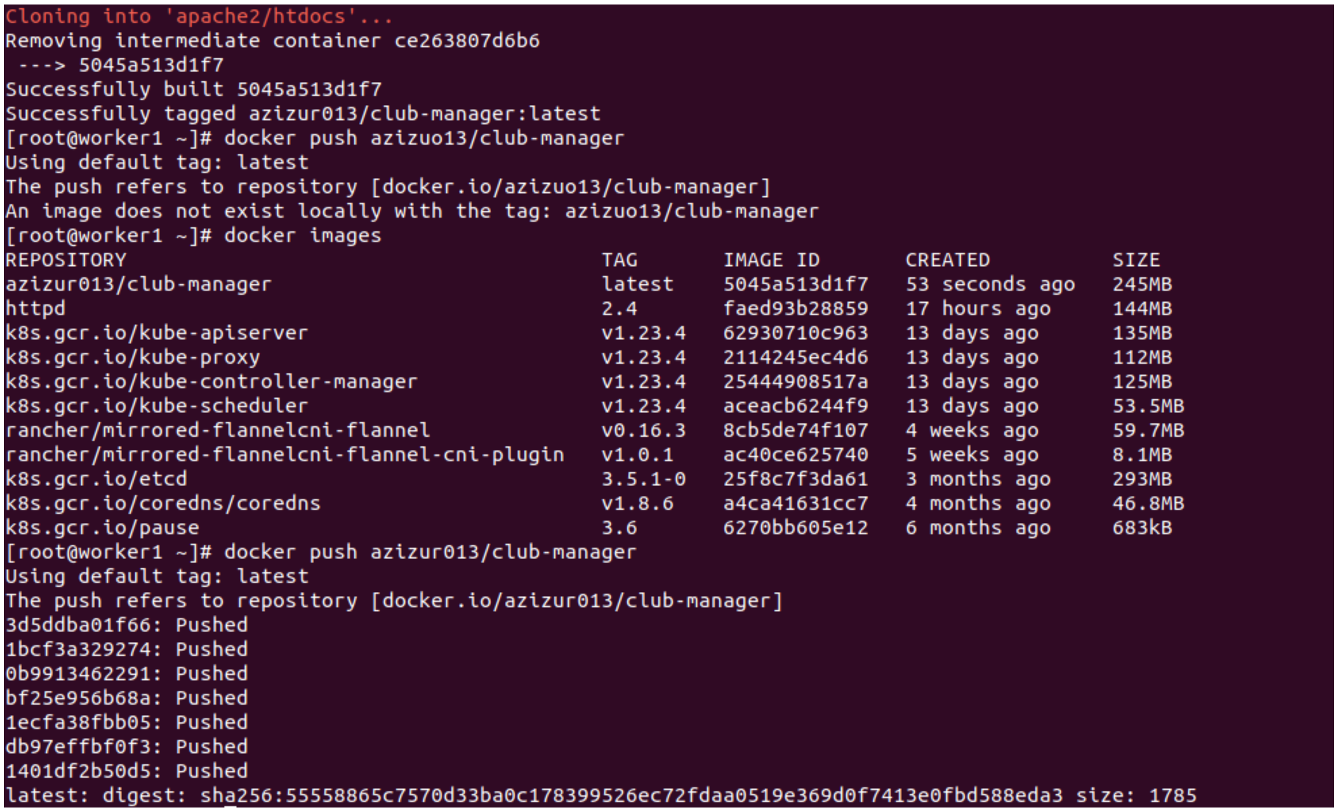
$kubectl get pods --all-namespaces



```
[root@master ~]# kubectl get nodes
NAME      STATUS   ROLES                  AGE   VERSION
master    Ready    control-plane,master   24m   v1.23.4
worker1   Ready    <none>                 19m   v1.23.4
[root@master ~]# kubectl get pods --all-namespaces
NAMESPACE         NAME                               READY   STATUS    RESTARTS   AGE
kube-system       coredns-64897985d-gfm5j            1/1     Running   0          24m
kube-system       coredns-64897985d-jdlg5            1/1     Running   0          24m
kube-system       etcd-master                        1/1     Running   1          24m
kube-system       kube-apiserver-master              1/1     Running   1          24m
kube-system       kube-controller-manager-master     1/1     Running   2          24m
kube-system       kube-flannel-ds-mdfsh              1/1     Running   0          19m
kube-system       kube-flannel-ds-wpdkh              1/1     Running   0          20m
kube-system       kube-proxy-jjqhz                   1/1     Running   0          24m
kube-system       kube-proxy-w4f8p                   1/1     Running   0          19m
kube-system       kube-scheduler-master              1/1     Running   2          24m
tigera-operator   tigera-operator-59fc55759-lldjq    1/1     Running   0          22m
```

5. Creating Docker image:

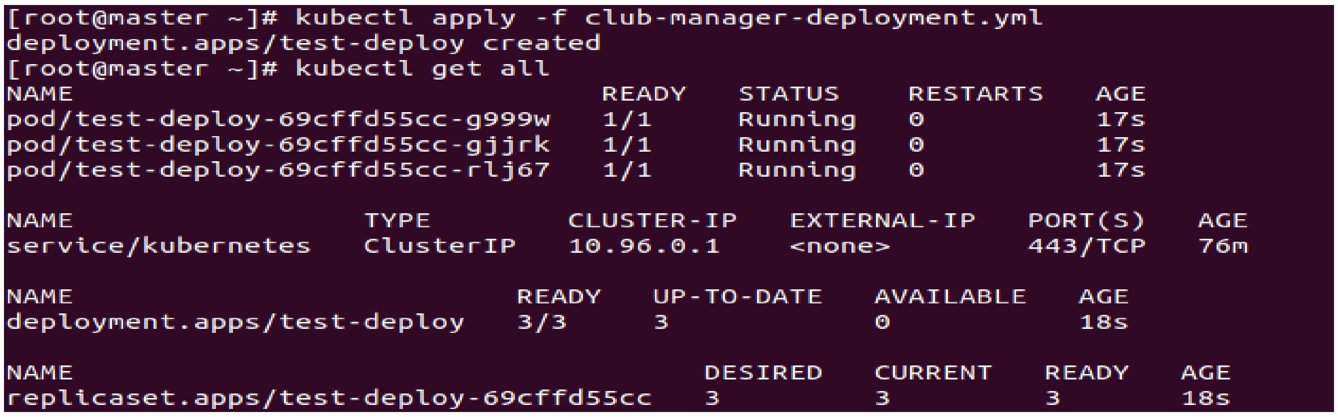
```
#Dockerfile configuration file
FROM httpd:2.4
WORKDIR /usr/local/
EXPOSE 80
RUN sed -i "s/DirectoryIndex index.html/DirectoryIndex index.php/g" apache2/conf/httpd.conf
RUN apt-get update -y \
        && apt-get install git -y \
        && rm -rf apache2/htdocs/index.html \
        && git clone https://github.com/shindesharad71/Club-Manager.git apache2/htdocs/
#docker image build
$sudo docker build -t azizur013/club-manager .
#push the image into docker-hub
$sudo docker push azizur013/club-manager
```



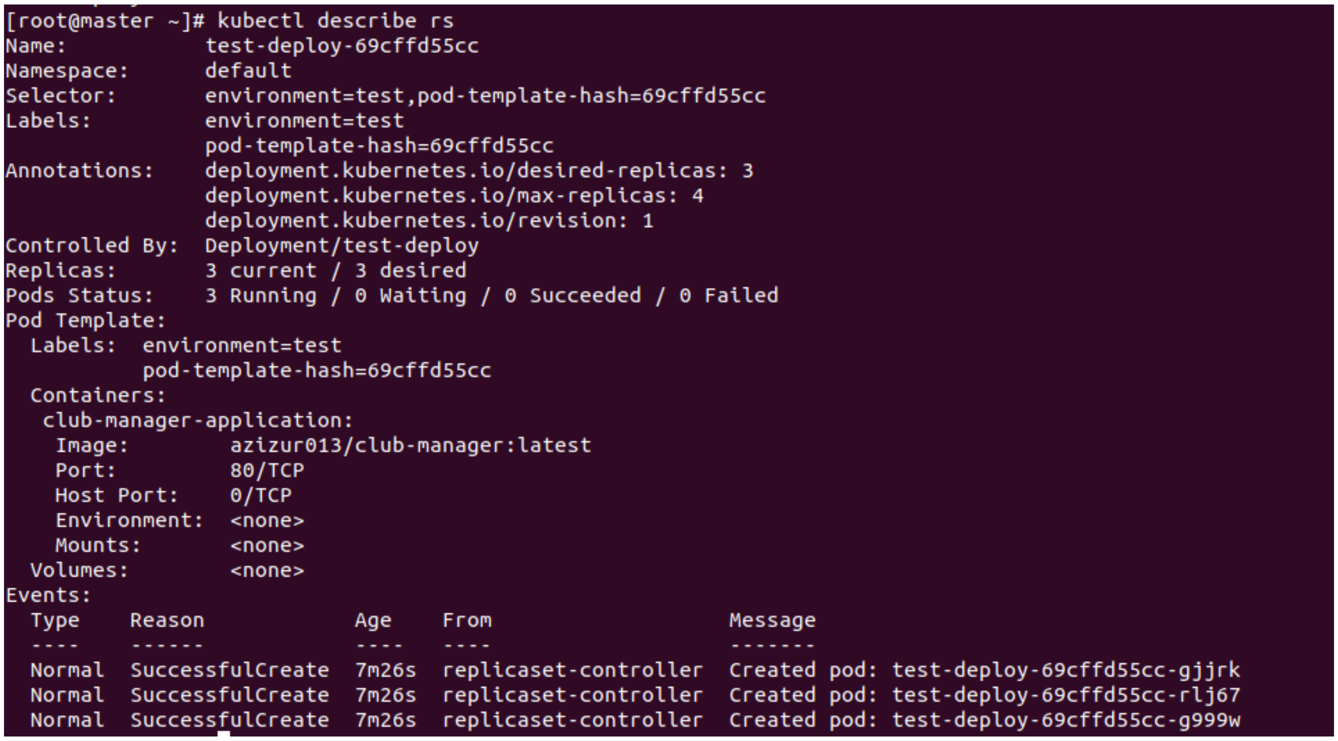
```
Cloning into 'apache2/htdocs'...
Removing intermediate container ce263807d6b6
 ---> 5045a513d1f7
Successfully built 5045a513d1f7
Successfully tagged azizur013/club-manager:latest
[root@worker1 ~]# docker push azizuo13/club-manager
Using default tag: latest
The push refers to repository [docker.io/azizuo13/club-manager]
An image does not exist locally with the tag: azizuo13/club-manager
[root@worker1 ~]# docker images
REPOSITORY                                  TAG       IMAGE ID       CREATED          SIZE
azizur013/club-manager                      latest    5045a513d1f7   53 seconds ago   245MB
httpd                                       2.4       faed93b28859   17 hours ago     144MB
k8s.gcr.io/kube-apiserver                   v1.23.4   62930710c963   13 days ago      135MB
k8s.gcr.io/kube-proxy                       v1.23.4   2114245ec4d6   13 days ago      112MB
k8s.gcr.io/kube-controller-manager          v1.23.4   25444908517a   13 days ago      125MB
k8s.gcr.io/kube-scheduler                   v1.23.4   aceacb6244f9   13 days ago      53.5MB
rancher/mirrored-flannelcni-flannel         v0.16.3   8cb5de74f107   4 weeks ago      59.7MB
rancher/mirrored-flannelcni-flannel-cni-plugin  v1.0.1   ac40ce625740   5 weeks ago   8.1MB
k8s.gcr.io/etcd                             3.5.1-0   25f8c7f3da61   3 months ago     293MB
k8s.gcr.io/coredns/coredns                  v1.8.6    a4ca41631cc7   4 months ago     46.8MB
k8s.gcr.io/pause                            3.6       6270bb605e12   6 months ago     683kB
[root@worker1 ~]# docker push azizur013/club-manager
Using default tag: latest
The push refers to repository [docker.io/azizur013/club-manager]
3d5ddba01f66: Pushed
1bcf3a329274: Pushed
0b9913462291: Pushed
bf25e956b68a: Pushed
1ecfa38fbb05: Pushed
db97effbf0f3: Pushed
1401df2b50d5: Pushed
latest: digest: sha256:55558865c7570d33ba0c178399526ec72fdaa0519e369d0f7413e0fbd588eda3 size: 1785
```

6.Test Deployment:

```
#manifest for the deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deploy
  labels:
    environment: test
spec:
  replicas: 3
  selector:
    matchLabels:
      environment: test
  minReadySeconds: 10

  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate

  template:
    metadata:
      labels:
        environment: test
    spec:
      containers:
      - name: club-manager-application
        image: azizur013/club-manager:latest
        ports:
        - containerPort: 80
#deployment command
$kubectl apply -f club-manager-deployment.yml
```



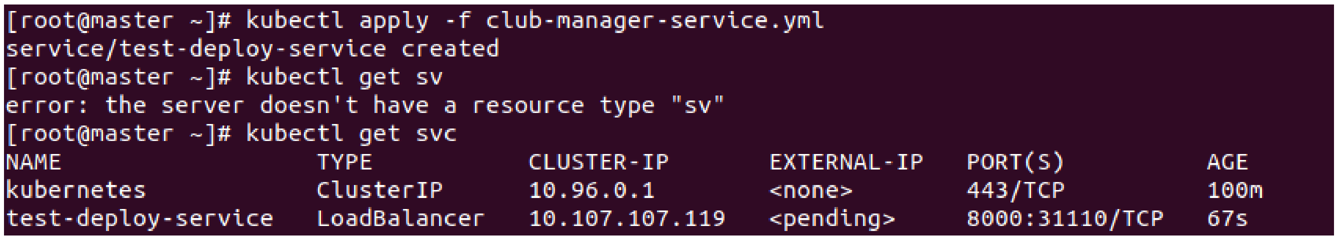
```
[root@master ~]# kubectl apply -f club-manager-deployment.yml
deployment.apps/test-deploy created
[root@master ~]# kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/test-deploy-69cffd55cc-g999w     1/1     Running   0          17s
pod/test-deploy-69cffd55cc-gjjrk     1/1     Running   0          17s
pod/test-deploy-69cffd55cc-rlj67     1/1     Running   0          17s

NAME                     TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes       ClusterIP   10.96.0.1     <none>        443/TCP   76m

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/test-deploy     3/3     3            0           18s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/test-deploy-69cffd55cc     3         3         3       18s
```



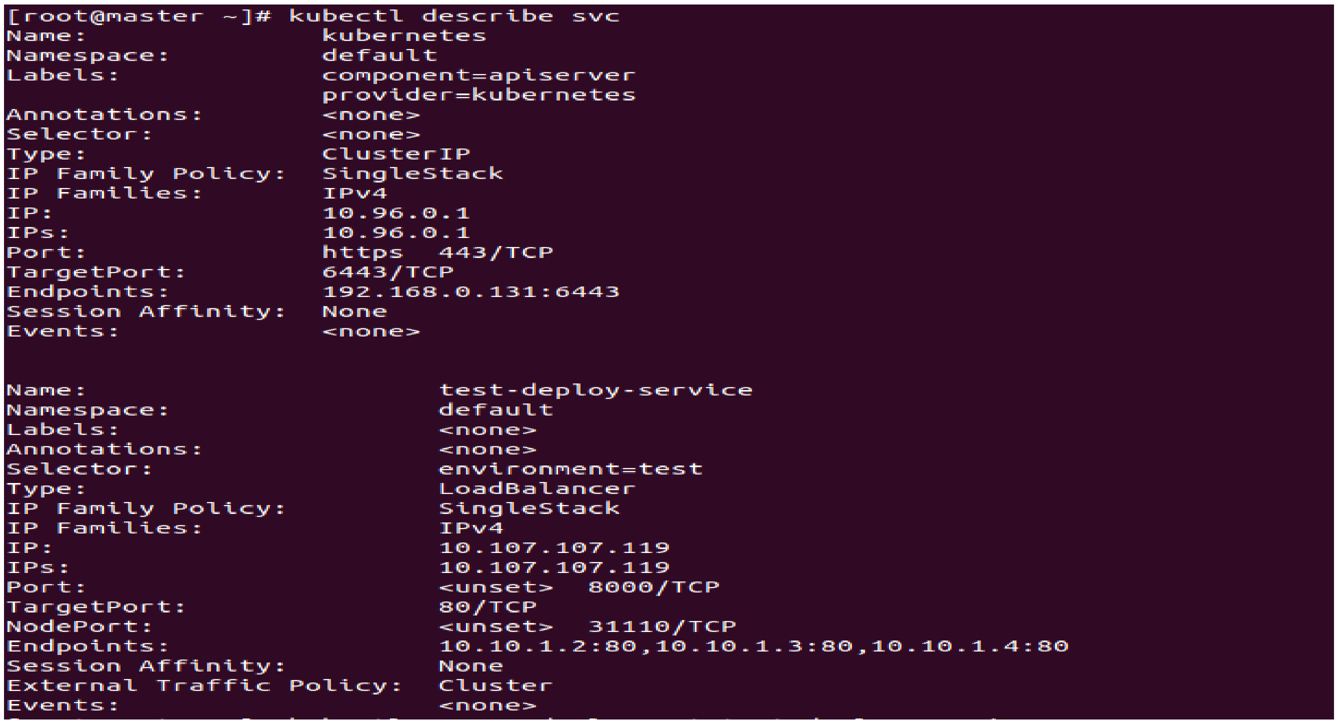
```
[root@master ~]# kubectl describe rs
Name:           test-deploy-69cffd55cc
Namespace:      default
Selector:       environment=test,pod-template-hash=69cffd55cc
Labels:         environment=test
                pod-template-hash=69cffd55cc
Annotations:    deployment.kubernetes.io/desired-replicas: 3
                deployment.kubernetes.io/max-replicas: 4
                deployment.kubernetes.io/revision: 1
Controlled By:  Deployment/test-deploy
Replicas:       3 current / 3 desired
Pods Status:    3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  environment=test
           pod-template-hash=69cffd55cc
  Containers:
   club-manager-application:
    Image:        azizur013/club-manager:latest
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age    From                   Message
  ----    ------            ----   ----                   -------
  Normal  SuccessfulCreate  7m26s  replicaset-controller  Created pod: test-deploy-69cffd55cc-gjjrk
  Normal  SuccessfulCreate  7m26s  replicaset-controller  Created pod: test-deploy-69cffd55cc-rlj67
  Normal  SuccessfulCreate  7m26s  replicaset-controller  Created pod: test-deploy-69cffd55cc-g999w
```

7. Creating the Service:

```
#yml to create the services
apiVersion: v1
kind: Service
metadata:
  name: test-deploy-service
spec:
  selector:
    environment: test
  type: LoadBalancer
  ports:
  - protocol: TCP
    port: 8000
    targetPort: 80
    nodePort: 31110
```

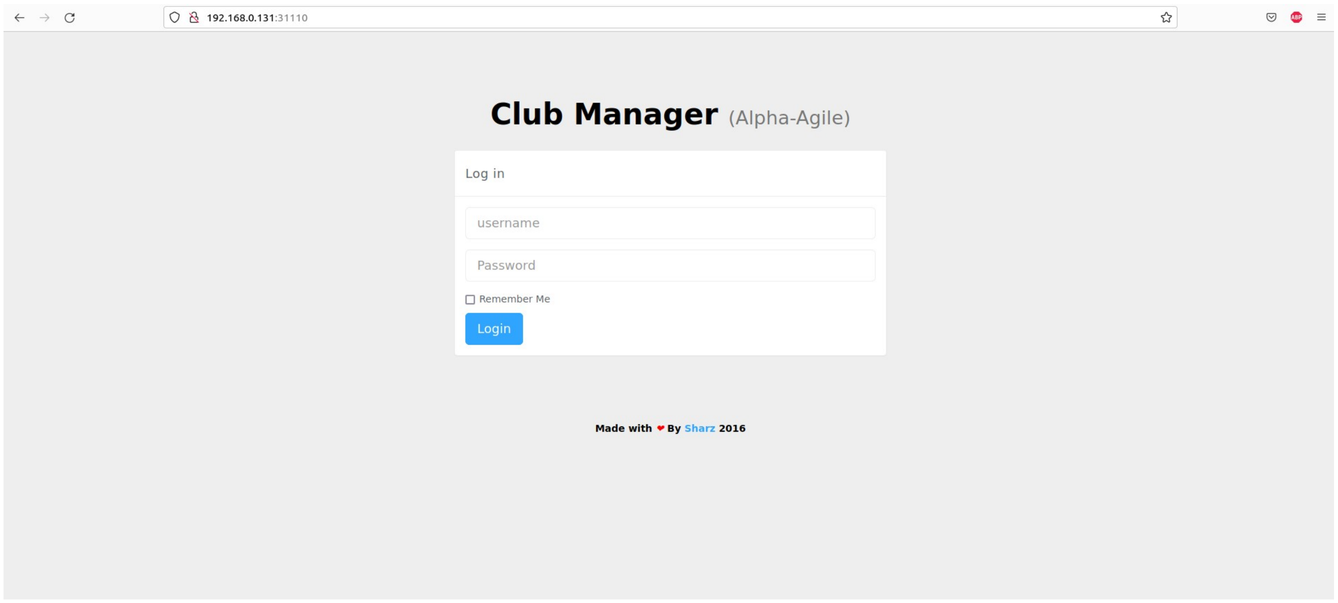


```
[root@master ~]# kubectl apply -f club-manager-service.yml
service/test-deploy-service created
[root@master ~]# kubectl get sv
error: the server doesn't have a resource type "sv"
[root@master ~]# kubectl get svc
NAME                  TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP      10.96.0.1        <none>        443/TCP          100m
test-deploy-service   LoadBalancer   10.107.107.119   <pending>     8000:31110/TCP   67s
```



```
[root@master ~]# kubectl describe svc
Name:                     kubernetes
Namespace:                default
Labels:                   component=apiserver
                          provider=kubernetes
Annotations:              <none>
Selector:                 <none>
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.96.0.1
IPs:                      10.96.0.1
Port:                     https  443/TCP
TargetPort:               6443/TCP
Endpoints:                192.168.0.131:6443
Session Affinity:         None
Events:                   <none>


Name:                            test-deploy-service
Namespace:                       default
Labels:                          <none>
Annotations:                     <none>
Selector:                        environment=test
Type:                            LoadBalancer
IP Family Policy:                SingleStack
IP Families:                     IPv4
IP:                              10.107.107.119
IPs:                             10.107.107.119
Port:                            <unset>  8000/TCP
TargetPort:                      80/TCP
NodePort:                        <unset>  31110/TCP
Endpoints:                       10.10.1.2:80,10.10.1.3:80,10.10.1.4:80
Session Affinity:                None
External Traffic Policy:         Cluster
Events:                          <none>
```

8.Final output :



192.168.0.131:31110

Club Manager (Alpha-Agile)

Log in

username

Password

Remember Me

Login

Made with ❤ By Sharz 2016

9. References:

a) Github Code Repo → https://github.com/shindesharad71/Club-Manager.git
b) Docker image → https://hub.docker.com/repository/docker/azizur013/club-manager
c) Kubeadm install documentation by Kubernetes→ https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
d) Kubectl commands → https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands
e) External code for centos 7 → https://computingforgeeks.com/install-kubernetes-cluster-on-centos-with-kubeadm/
f) Kubernetes Dashboard configure → https://computingforgeeks.com/how-to-install-kubernetes-dashboard-with-nodeport/
g) Youtube video reference → https://www.youtube.com/watch?v=CfPRbdT-wXo