Deploy custom docker image and create kubernetes container in Lenode CSP

#Install Docker

```
sudo apt-get update


sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \


curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg


echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null


sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

#Install Kubectl

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg


echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/
kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list


sudo apt-get update
sudo apt-get install -y kubectl
```

First we need to create docker image and push it to the docker hub, for this example I create small html website and host that on **nginx** server.

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ nano Index.html
```

```
  GNU nano 4.8                                          index.html
<title> new weboage </title>
<h1>new web page </h1>
```

Okay now we need to create docker file to create the image,

```
  GNU nano 4.8                                          dockerfile
FROM nginx

COPY ./*.html /usr/share/nginx/html/
```

FROM nginx - this command is use for import existing image

COPY ./*.html /usrshare/nginx/html/ - copying all files with .html extension to the imported nginx image's file directory.

Before creating the image first we need to login to the docker hub,

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head ov
er to https://hub.docker.com to create one.
Username: mahela404
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
nobodyy@pc:~/k8s$
```

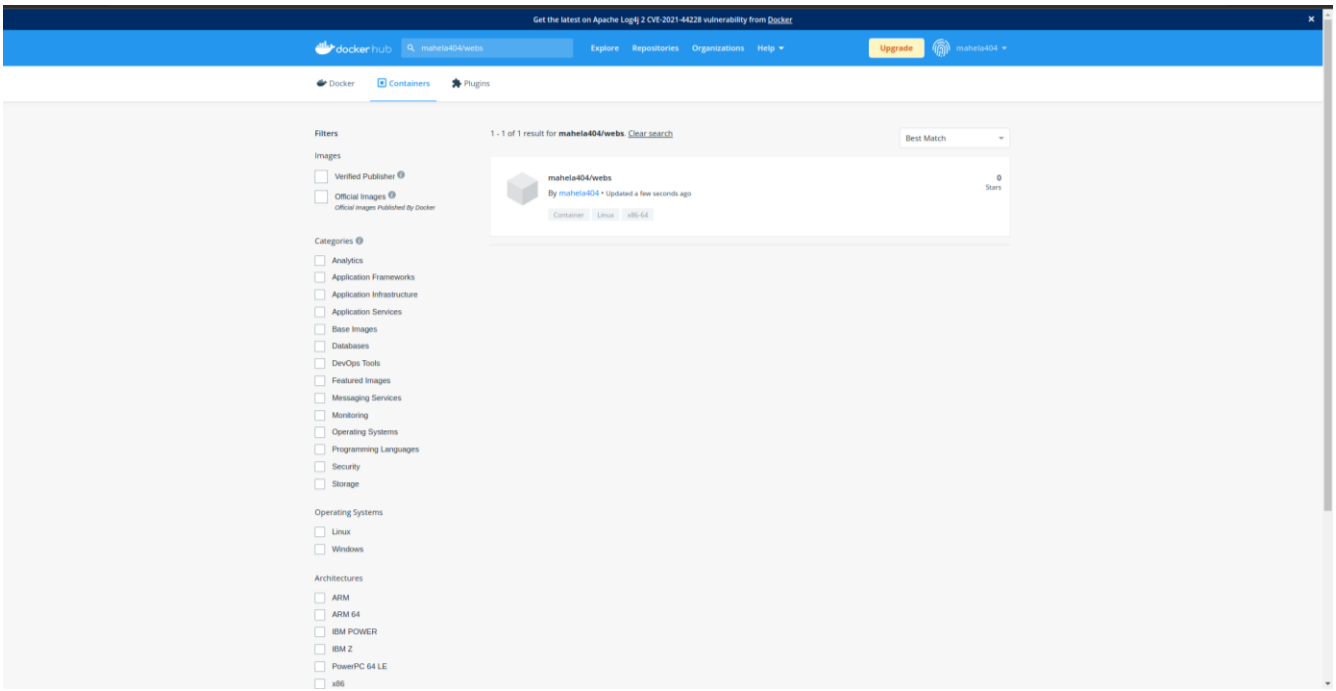
After login we need to create the image,
sudo docker build . -t mahela404/webs

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ sudo docker build . -t mahela404/webs
Sending build context to Docker daemon  6.656kB
Step 1/2 : FROM nginx
 ---> 605c77e624dd
Step 2/2 : COPY ./*.html /usr/share/nginx/html/
 ---> Using cache
 ---> 845f802749e7
Successfully built 845f802749e7
Successfully tagged mahela404/webs:latest
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$
```

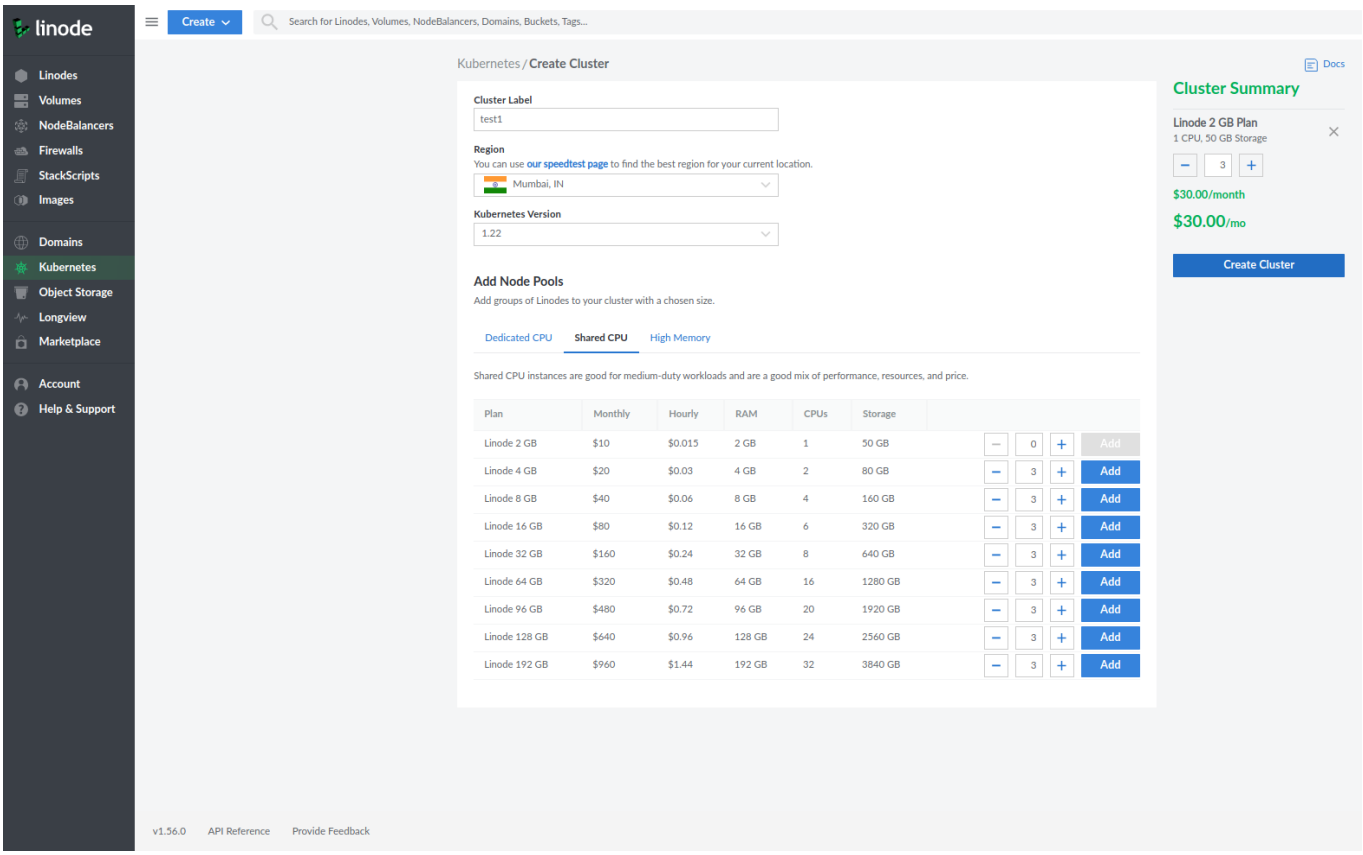
Now lets push(export) this image to the docker hub,
sudo docker push mahela404/webs

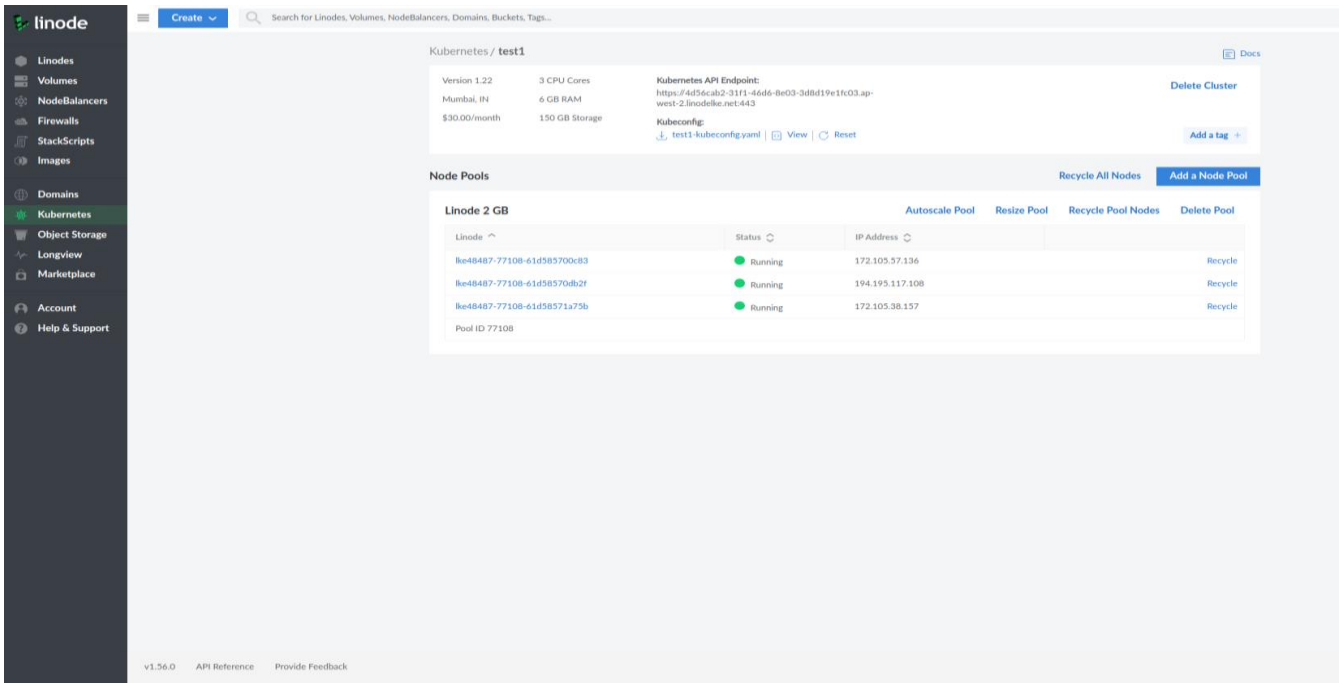
```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ sudo docker push mahela404/webs
Using default tag: latest
The push refers to repository [docker.io/mahela404/webs]
3c07d0121739: Pushed
d874fd2bc83b: Mounted from library/nginx
32ce5f6a5106: Mounted from library/nginx
f1db227348d0: Mounted from library/nginx
b8d6e692a25e: Mounted from library/nginx
e379e8aedd4d: Mounted from library/nginx
2edcec3590a4: Mounted from library/nginx
latest: digest: sha256:7f2f1e98b32b70bc7a176747f9afcd09b00a8997dfbb7c1128b6a0036ae2e212 size: 1777
nobodyy@pc:~/k8s$
```

Lets login and see if the image is on the docker hub,

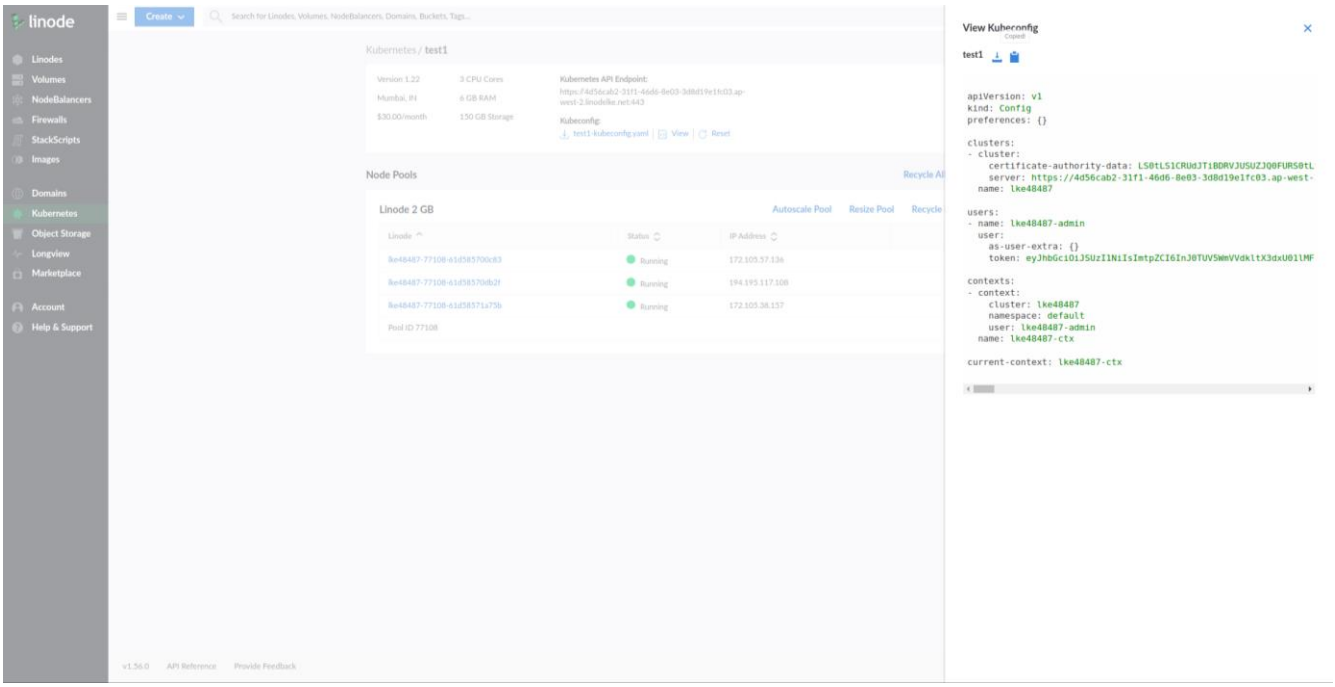


Create kubernetes cluster,

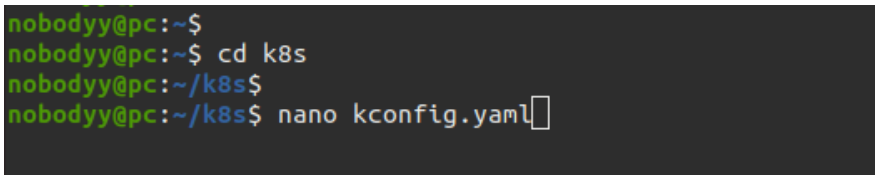


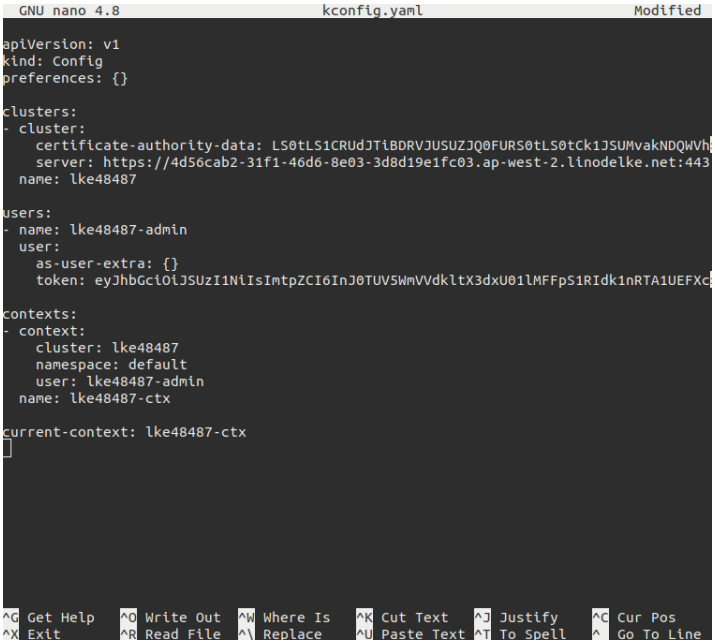


Copying config file to connect API server,



Paste in another file and save it as .yaml file (you can save it with any name but with .yaml).





And then run **‘export KUBECONFIG=kconfig.yaml’** command to connect to the kubernetes cluster,

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ export KUBECONFIG=kconfig.yaml
```

Now you are connected to the cluster, you can list nodes by running **‘kubectl get nodes’,**

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ kubectl get nodes
NAME                           STATUS   ROLES    AGE     VERSION
lke48487-77108-61d585700c83    Ready    <none>   5m29s   v1.22.2
lke48487-77108-61d58570db2f    Ready    <none>   5m32s   v1.22.2
lke48487-77108-61d58571a75b    Ready    <none>   5m33s   v1.22.2
nobodyy@pc:~/k8s$
```

Lets create kubernetes container by using that image we created,

kubectl run test1 –image=mahela404//webs –port=80

To list pods,

kubectl get pods

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ kubectl run test1 --image=mahela404/webs --port=80
pod/test1 created
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
test1   1/1     Running   0          9s
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$
```

To more details about pods,

kubectl describe pods

```
nobodyy@pc:~/k8s$ kubectl describe pods
Name:         test1
Namespace:    default
Priority:     0
Node:         lke48487-77108-61d58570db2f/192.168.132.148
Start Time:   Wed, 05 Jan 2022 18:15:27 +0530
Labels:       run=test1
Annotations:  cni.projectcalico.org/podIP: 10.2.1.2/32
              cni.projectcalico.org/podIPs: 10.2.1.2/32
Status:       Running
IP:           10.2.1.2
IPs:
  IP:  10.2.1.2
Containers:
  test1:
    Container ID:   docker://f36c96e11ff9504c0a4b75e3e51490c0db6023dbe422fb3da8f353af9529a6c0
    Image:          mahela404/webs
    Image ID:       docker-pullable://mahela404/webs@sha256:7f2f1e98b32b70bc7a176747f9afcd09b00a8997dfbb7
c1128b6a0036ae2e212
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 05 Jan 2022 18:15:36 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-p7pbr (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-p7pbr:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  73s   default-scheduler  Successfully assigned default/test1 to lke48487-77108-61d58
570db2f
  Normal  Pulling    72s   kubelet            Pulling image "mahela404/webs"
  Normal  Pulled     65s   kubelet            Successfully pulled image "mahela404/webs" in 7.847930314s
  Normal  Created    64s   kubelet            Created container test1
  Normal  Started    64s   kubelet            Started container test1
nobodyy@pc:~/k8s$
```

Delete pods,

kubectl delete pods test1

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ kubectl delete pods test1
\pod "test1" deleted
nobodyy@pc:~/k8s$
```

Deploy pods with deployment file with .yaml extension,

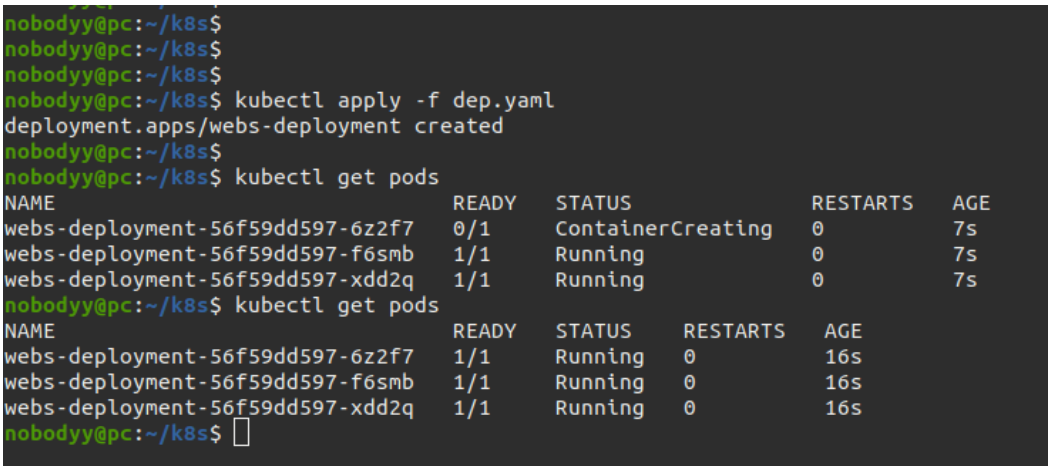
on: apps/v1
kind: Deployment
metadata:
name: webs-deployment
labels:
app: test1
spec:
replicas: 3
selector:
matchLabels:
app: test1
template:
metadata:
labels:
app: test1
spec:
containers:
- name: test1
image: mahela404/webs
ports:
- containerPort: 80

```
 GNU nano 4.8                                    dep.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webs-deployment
  labels:
    app: test1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: test1
  template:
    metadata:
      labels:
        app: test1
    spec:
      containers:
      - name: test1
        image: mahela404/webs
        ports:
        - containerPort: 80
```

Applying deployment file,

kubectl apply -f dep.yaml

After applying its automatically create pods,

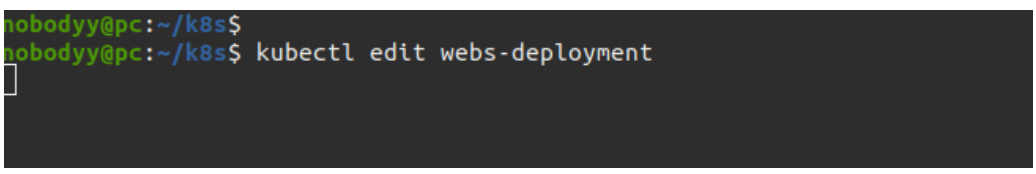


To list deployments,

kubectl get deployments

We only created 3 pods. Lets make it 10 by editing deployment,

kubectl edit webs-deployment



change replicas value to change number of pods, in here I change it as 10 replicas

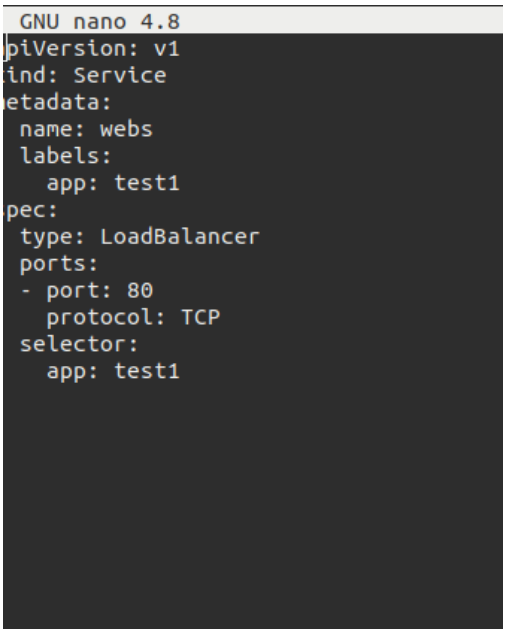
```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"app":"test1"},"
name":"webs-deployment","namespace":"default"},"spec":{"replicas":3,"selector":{"matchLabels":{"app":"tes
t1"}},"template":{"metadata":{"labels":{"app":"test1"}},"spec":{"containers":[{"image":"mahela404/webs","
name":"test1","ports":[{"containerPort":80}]}]}}}}
  creationTimestamp: "2022-01-05T12:49:30Z"
  generation: 1
  labels:
    app: test1
  name: webs-deployment
  namespace: default
  resourceVersion: "3917"
  uid: 28ba7a05-84a5-48c7-bd8d-e2579217e673
spec:
  progressDeadlineSeconds: 600
  replicas: 10
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: test1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: test1
    spec:
      containers:
      - image: mahela404/webs
        imagePullPolicy: Always
        name: test1
        ports:
        - containerPort: 80
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
"/tmp/kubectl-edit-731006083.yaml" 71 lines, 2290 characters
```

After save and list the pods we can see 10 pods instead of 3.

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
webs-deployment-56f59dd597-6z2f7   1/1     Running   0          3m16s
webs-deployment-56f59dd597-8ckjv   1/1     Running   0          22s
webs-deployment-56f59dd597-9x5d4   1/1     Running   0          22s
webs-deployment-56f59dd597-f6smb   1/1     Running   0          3m16s
webs-deployment-56f59dd597-jjc6f   1/1     Running   0          22s
webs-deployment-56f59dd597-lzsms   1/1     Running   0          22s
webs-deployment-56f59dd597-mx68r   1/1     Running   0          22s
webs-deployment-56f59dd597-r2x92   1/1     Running   0          22s
webs-deployment-56f59dd597-xdd2q   1/1     Running   0          3m16s
webs-deployment-56f59dd597-z4swb   1/1     Running   0          22s
nobodyy@pc:~/k8s$
```

Now lets create the load balancer, for that we need to create service file with .yaml extension.

```
apiVersion: v1
kind: Service
metadata:
  name: webs
  labels:
    app: test1
spec:
  type: LoadBalancer
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: test1
```

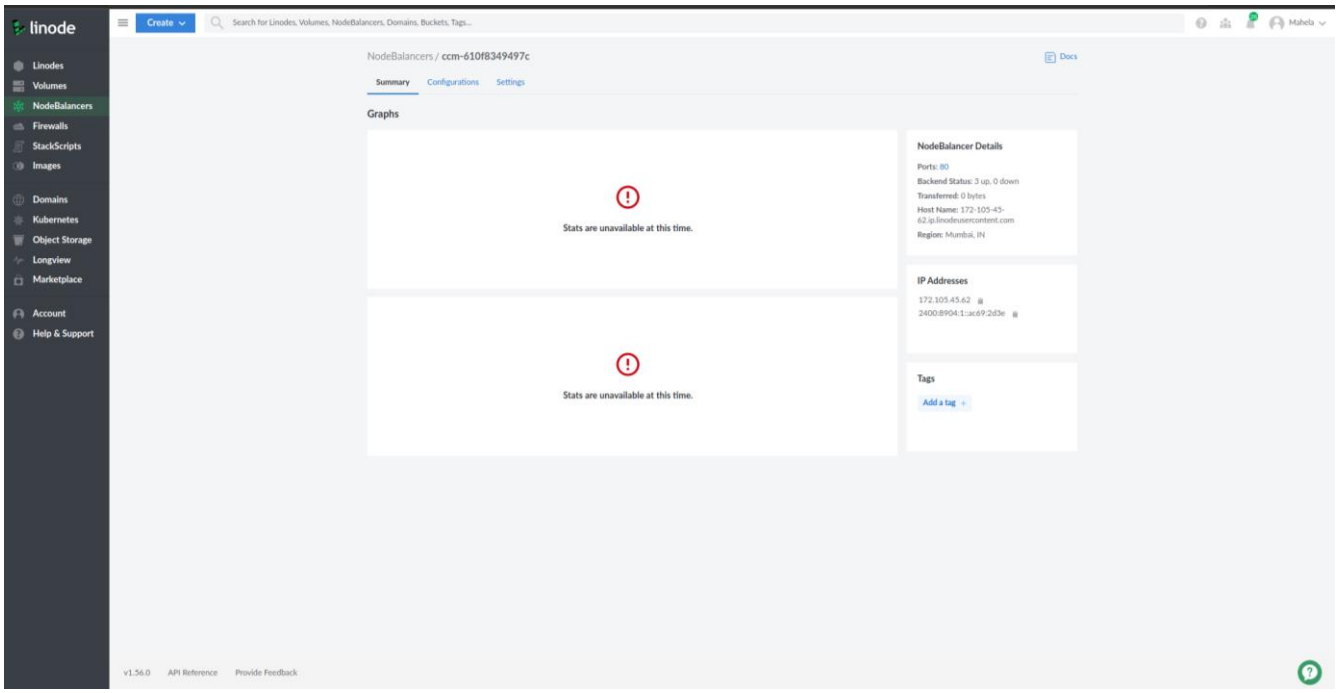


and apply the service
kubectl apply -f ser.yaml

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ nano ser.yaml
nobodyy@pc:~/k8s$ kubectl apply -f ser.yaml
service/webs created
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$
```

After creating a load balancer we can access our website through the internet,

Lets check on Linode for load balancer, in Linode they call it Node balancer



To get public IP of our website,

kubectl get services

```
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$ kubectl get services
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
kubernetes   ClusterIP      10.128.0.1      <none>          443/TCP        66m
webs         LoadBalancer   10.128.147.79   172.105.45.62   80:31203/TCP   25s
nobodyy@pc:~/k8s$
nobodyy@pc:~/k8s$
```

Lets check our website with the IP.