Module 4: Expose App, Scale App and Update App

DEMO-7

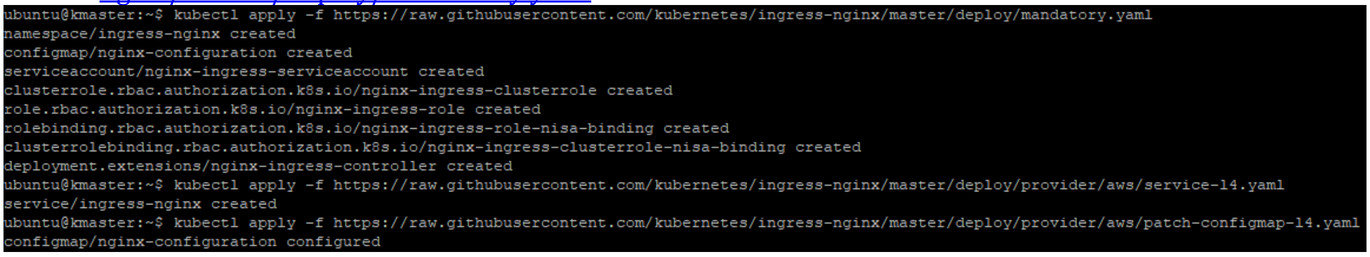
DEMO Steps:

Create an Ingress(nginx) loadbalancer controller

Note: Use the following link for reference: https://github.com/kubernetes/ingress-nginx/blob/master/docs/deploy/index.md

1. The following command is mandatory for all configurations

   Syntax: kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/mandatory.yaml



```
ubuntu@kmaster:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/mandatory.yaml
namespace/ingress-nginx created
configmap/nginx-configuration created
serviceaccount/nginx-ingress-serviceaccount created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created
role.rbac.authorization.k8s.io/nginx-ingress-role created
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-binding created
deployment.extensions/nginx-ingress-controller created
ubuntu@kmaster:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/service-l4.yaml
service/ingress-nginx created
ubuntu@kmaster:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/patch-configmap-l4.yaml
configmap/nginx-configuration configured
```

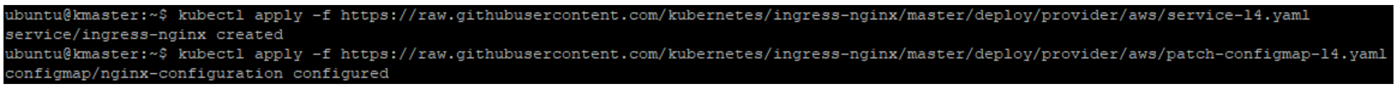
2. Now the next command depends upon the environment you're using you cluster in. The link given in the beginning provides the commands for environments such as Mac, Azure, GKE, AWS, Baremetal and so on

   For this example we're going to use AWS L4 configuration:

   Syntax:

   kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/service-l4.yaml

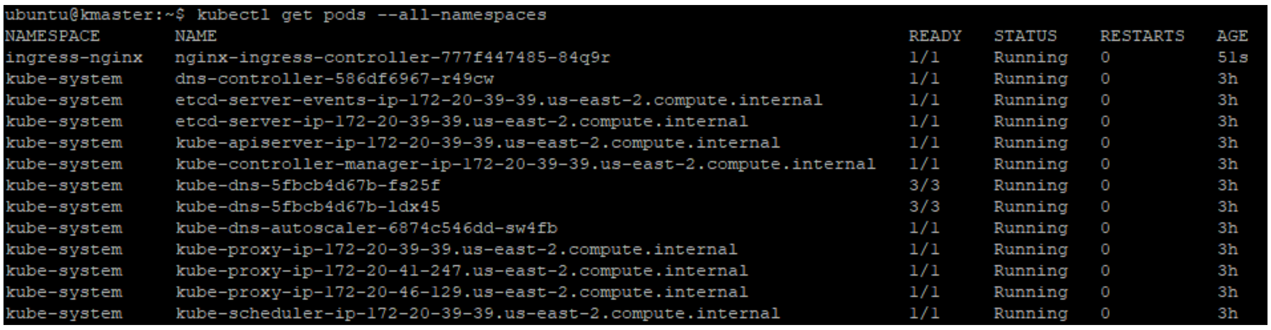
   kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/patch-configmap-l4.yaml



```
ubuntu@kmaster:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/service-l4.yaml
service/ingress-nginx created
ubuntu@kmaster:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/aws/patch-configmap-l4.yaml
configmap/nginx-configuration configured
```

3. Check you pods to see all the ingress pods are up and running

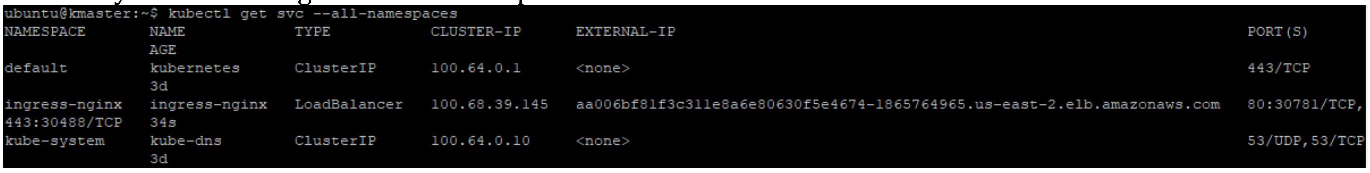
   Syntax: kubectl get pods --all-namespaces



```
ubuntu@kmaster:~$ kubectl get pods --all-namespaces
NAMESPACE       NAME                                                            READY   STATUS    RESTARTS   AGE
ingress-nginx   nginx-ingress-controller-777f447485-84q9r                       1/1     Running   0          51s
kube-system     dns-controller-586df6967-r49cw                                  1/1     Running   0          3h
kube-system     etcd-server-events-ip-172-20-39-39.us-east-2.compute.internal   1/1     Running   0          3h
kube-system     etcd-server-ip-172-20-39-39.us-east-2.compute.internal          1/1     Running   0          3h
kube-system     kube-apiserver-ip-172-20-39-39.us-east-2.compute.internal       1/1     Running   0          3h
kube-system     kube-controller-manager-ip-172-20-39-39.us-east-2.compute.internal 1/1  Running   0          3h
kube-system     kube-dns-5fbcb4d67b-fs25f                                       3/3     Running   0          3h
kube-system     kube-dns-5fbcb4d67b-ldx45                                       3/3     Running   0          3h
kube-system     kube-dns-autoscaler-6874c546dd-sw4fb                            1/1     Running   0          3h
kube-system     kube-proxy-ip-172-20-39-39.us-east-2.compute.internal           1/1     Running   0          3h
kube-system     kube-proxy-ip-172-20-41-247.us-east-2.compute.internal          1/1     Running   0          3h
kube-system     kube-proxy-ip-172-20-46-129.us-east-2.compute.internal          1/1     Running   0          3h
kube-system     kube-scheduler-ip-172-20-39-39.us-east-2.compute.internal       1/1     Running   0          3h
```

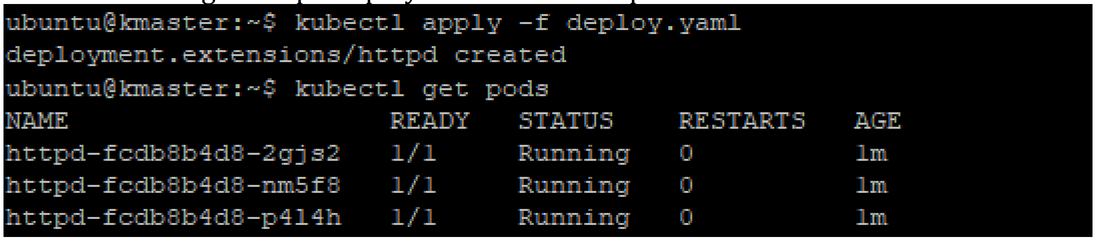
4. Check the services to verify ingress service is working

   Syntax: kubectl get svc --all-namspaces



```
ubuntu@kmaster:~$ kubectl get svc --all-namespaces
NAMESPACE       NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)
                AGE
default         kubernetes      ClusterIP      100.64.0.1      <none>                                                                    443/TCP
                3d
ingress-nginx   ingress-nginx   LoadBalancer   100.68.39.145   aa006bf81f3c311e8a6e80630f5e4674-1865764965.us-east-2.elb.amazonaws.com   80:30781/TCP,
443:30488/TCP   34s
kube-system     kube-dns        ClusterIP      100.64.0.10     <none>                                                                    53/UDP,53/TCP
                3d
```

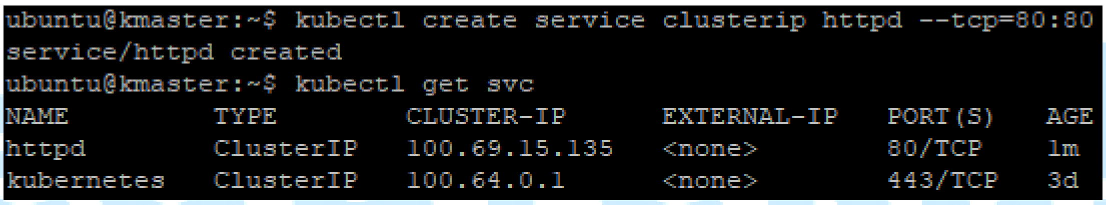
5. Now create a deployment like we did before
   Here we're using an httpd deployment as an example



```
ubuntu@kmaster:~$ kubectl apply -f deploy.yaml
deployment.extensions/httpd created
ubuntu@kmaster:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
httpd-fcdb8b4d8-2gjs2    1/1     Running   0          1m
httpd-fcdb8b4d8-nm5f8    1/1     Running   0          1m
httpd-fcdb8b4d8-p414h    1/1     Running   0          1m
```

6. Create a httpd clusterip service

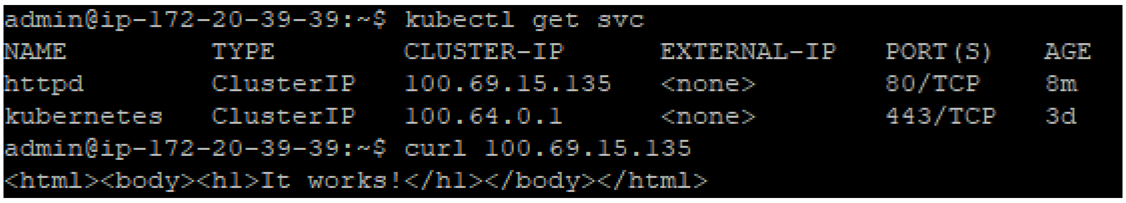
   Syntax: kubectl create service clusterip httpd --tcp=80:80



```
ubuntu@kmaster:~$ kubectl create service clusterip httpd --tcp=80:80
service/httpd created
ubuntu@kmaster:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
httpd        ClusterIP   100.69.15.135   <none>        80/TCP    1m
kubernetes   ClusterIP   100.64.0.1      <none>        443/TCP   3d
```

7. Curl the service IP to make sure it is attached to the pods

   Syntax: curl <Cluster IP address>



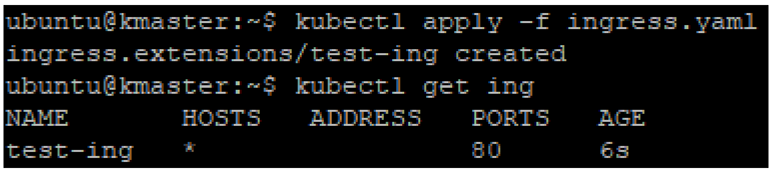
```
admin@ip-172-20-39-39:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
httpd        ClusterIP   100.69.15.135   <none>        80/TCP    8m
kubernetes   ClusterIP   100.64.0.1      <none>        443/TCP   3d
admin@ip-172-20-39-39:~$ curl 100.69.15.135
<html><body><h1>It works!</h1></body></html>
```

8. Now, create an ingress rule for your service so you can access the service at /test
   Syntax: vi ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ing
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /test
        backend:
          serviceName: httpd
          servicePort: 80
```

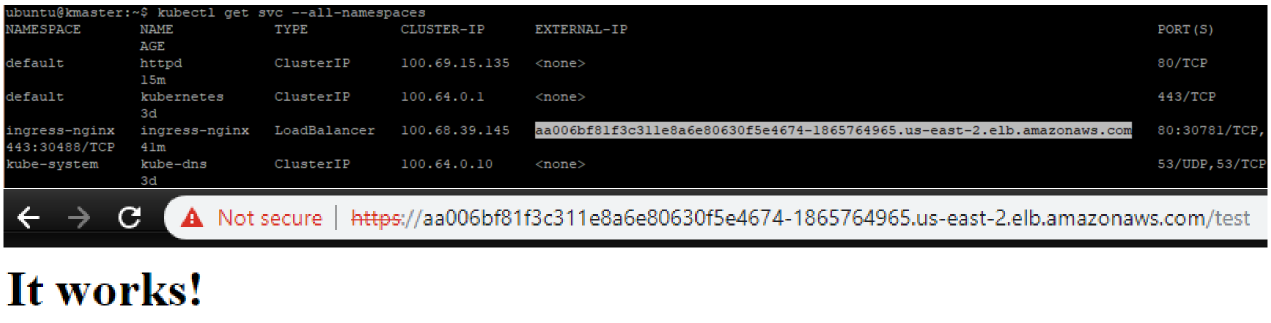
9. Execute the ingress rule

   Syntax: kubectl apply -f ingress.yaml



```
ubuntu@kmaster:~$ kubectl apply -f ingress.yaml
ingress.extensions/test-ing created
ubuntu@kmaster:~$ kubectl get ing
NAME        HOSTS    ADDRESS    PORTS    AGE
test-ing    *                   80       6s
```

10. Now copy the ingress service external IP and add /test to it in your browser to verify



```
ubuntu@kmaster:~$ kubectl get svc --all-namespaces
NAMESPACE       NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)
                AGE
default         httpd           ClusterIP      100.69.15.135   <none>                                                                   80/TCP
                15m
default         kubernetes      ClusterIP      100.64.0.1      <none>                                                                   443/TCP
                3d
ingress-nginx   ingress-nginx   LoadBalancer   100.68.39.145   aa006bf81f3c311e8a6e80630f5e4674-1865764965.us-east-2.elb.amazonaws.com   80:30781/TCP,
443:30488/TCP   41m
kube-system     kube-dns        ClusterIP      100.64.0.10     <none>                                                                   53/UDP,53/TCP
                3d
```

Not secure | ~~https~~://aa006bf81f3c311e8a6e80630f5e4674-1865764965.us-east-2.elb.amazonaws.com/test

It works!