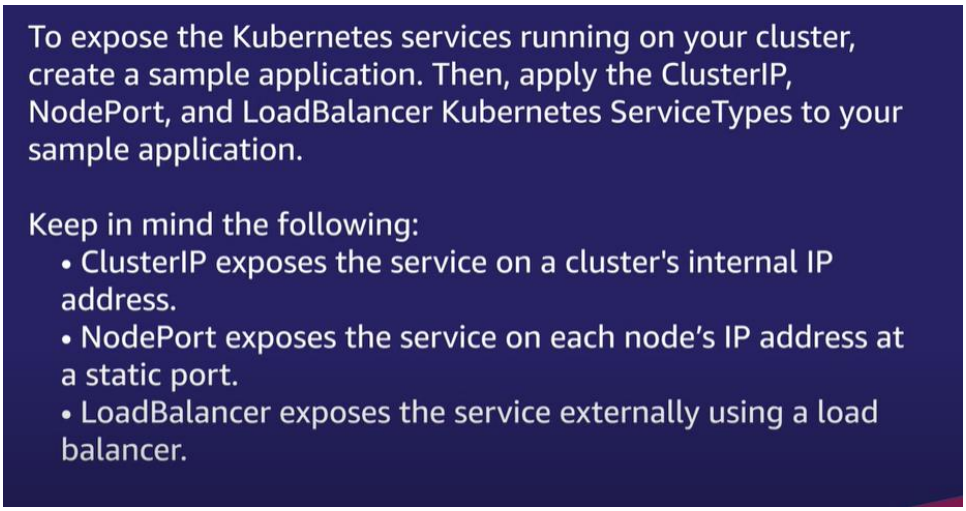



To expose the k8s services running on your cluster, create a sample application. Then apply the ClusterIP, NodePort, and LoadBalancer kubernetes Service Types to your sample application

Keep in mind the following

--> ClusterIP exposes the services on a cluster's internal IP address

--> NodePort exposes the service on each node's IP address at a static Port

--> LoadBalancer exposes the service externally using a load balancer

NOTE:- Amazon EKS supports network load balancers and classic load balancers for pods running on Amazon Elastic Compute Cloud instance worker nodes. Amazon EKS provides this support by using the LoadBalancer

We can load balance network traffic to a network load balancer, with instance or IP targets or a classic load balancer with instance target only.

The first step is to create a sample application to do

Define and apply the deployment file

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

The following example creates a replica set that spins up two nginx pods and then creates a file called nginx-deployment.yml

Create the deplpoyment by running the following command

```
Password:
$ kubectl apply -f nginx-deployment.yaml
```

Now verify that your pods are running and have their own internal IP addresses

```
$ kubectl get pods -l 'app=nginx' -o wide | awk {'print $1" " $3 " " $6'} | column -t
NAME                                STATUS   IP
nginx-deployment-66b6c48dd5-kxkbs   Running  192.168.73.64
nginx-deployment-66b6c48dd5-w976c   Running  192.168.57.140
$
```

Now that we have successfully created a sample application

Let's start with the next step to create the cluster IP service

First create a file called clusterip.yml and then set type to cluster ip for example

```
$ sudo nano clusterip.yaml
```

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-cluster-ip
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Create the cluster ip object in k8s using either a declarative or imperative command to create the object and apply the clusterip.yml file run the following declarative command

```
$ kubectl create -f clusterip.yaml
service/nginx-service-cluster-ip created
$
```

To expose a deployment of clusterip type run the following command

```
kubectl expose deployment nginx-deployment  --type=ClusterIP
--name=nginx-service-cluster-ip
```

NOTE:- The expose command creates a service without creating a YAML file. However, kubectl translates your imperative command into a declarative kubernetes deployment object

To access the cluster ip using the following command

```
$ kubectl get service nginx-service-cluster-ip
NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
nginx-service-cluster-ip   ClusterIP   10.[redacted]  <none>        80/TCP    51s
$
```

First create a test pod using the following command

```
$ kubectl run test-pod --image nicolaka/netshoot --command sleep 9999
pod/test-pod created
$
```

Next use exec into the pod and curl the cluster ip

```
$ kubectl exec -it test-pod -- /bin/bash
bash-5.1#
```

==========#### NODE PORT SERVICE ####============

To create node port service to create file called node,yml and then set type to node port

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-nodeport
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

For example to create an output object in k8s using either a declarative or imperative command to create the object and apply the node.yml

```
$ sudo nano nodeport.yaml
Password:
$ kubectl create -f nodeport.yaml
```

And alternatively to expose a deployment of node port type run the following imperative command get the information about nginx service

```
service/nginx-service-nodeport created
$ kubectl get service/nginx-service-nodeport
NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
nginx-service-nodeport     NodePort    10.             <none>        80:30051/TCP   23s
$
```

An important note the service type is a node port and cluster ip are created automatically for the service the output from the preceding command shows that the node port service is export externally on the port 30051 of the available worker nodes EC2 instance before you access node ip : node port from outside the cluster you must set the security group of the nodes to allow the incoming traffic ,you can allow the incoming traffic through the port 30051 that's listed in the out put of the preceding **kubectl get service** command if the node is in the public subnet and is reachable from the internet then check the node's public ip address using the following command

```
$ kubectl get nodes -o wide |  awk {'print $1" " $2 " " $7'} | column -t
NAME                          STATUS  EXTERNAL-IP
ip-192-168-46-195.ec2.internal  Ready
ip-192-168-88-239.ec2.internal  Ready
$
```

**kubectl get nodes –o wide | awk {'print $1" " $2 " " $7'} | column –t**

Or if the node is in private subnet and is reachable only inside or through a vpc then check the node's private ip address using the following command to test the configuration let's create a test port using the following command

```
$ kubectl get nodes -o wide |  awk {'print $1" " $2 " " $6'} | column -t
NAME                          STATUS  INTERNAL-IP
ip-192-168-46-195.ec2.internal  Ready   192.168.46.195
ip-192-168-88-239.ec2.internal  Ready   192.168.88.239
$ kubectl run test-pod1 --image nicolaka/netshoot --command sleep 9999
pod/test-pod1 created
$
```

Next we will exec in to the pod and curl the node ip including the node port

```
pod/test-pod1 created
$ kubectl exec -it test-pod1 -- /bin/bash
```

Now if you would like to delete the node port service

**Kubectl delete service nginx-service-nodeport**

=====###### LOAD BALANCER SERVICE ######====

Now let's create the load balancer service to create a load balancer service create a file called loadbalancer.yaml

```
 GNU nano 2.0.6                                                  F

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

For example apply the loadbalancer.yml using the follwoing command

**Kubectl create –f loadbalancer.yaml**

Or to expose a deployment of loadbalancer type run the following

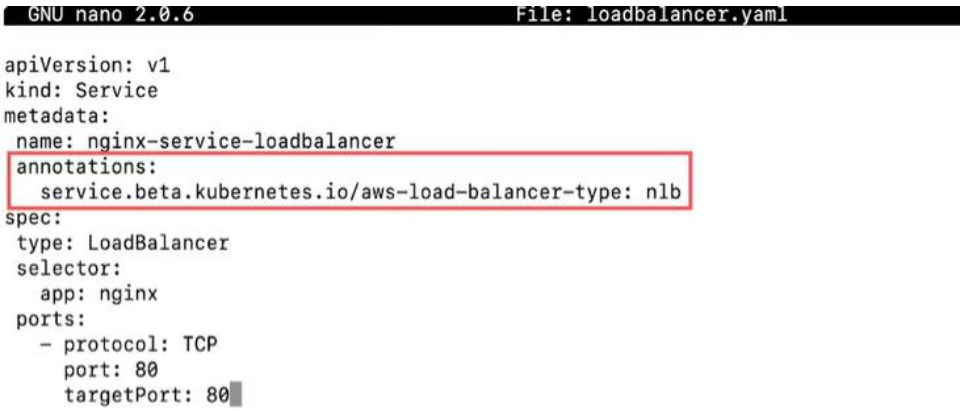
```
$ kubectl expose deployment nginx-deployment  --type=LoadBalancer  --name=nginx-service-loadbalancer
service/nginx-service-loadbalancer exposed
$
```

Get the information about the nginx service

```
service/nginx-service-loadbalancer exposed
$ kubectl get service/nginx-service-loadbalancer |  awk {'print $1" " $2 " " $4 " " $5'} | column -t
NAME                        TYPE          EXTERNAL-IP                                                    PORT(S)
nginx-service-loadbalancer  LoadBalancer  a7239                                          .us-east-1.elb.amazonaws.com  80:32441/TCP
$
```

Verify that you can access the load balancer externally

NOTE: by default the preceding load balancer service creates a classic load balancer to create a network load balancer with the instance type target add the following annotations




To the service manifest or you can create a network load balancer with IP targets deploy the load balancer controller and then create a load balancer that uses ip targets to learn more