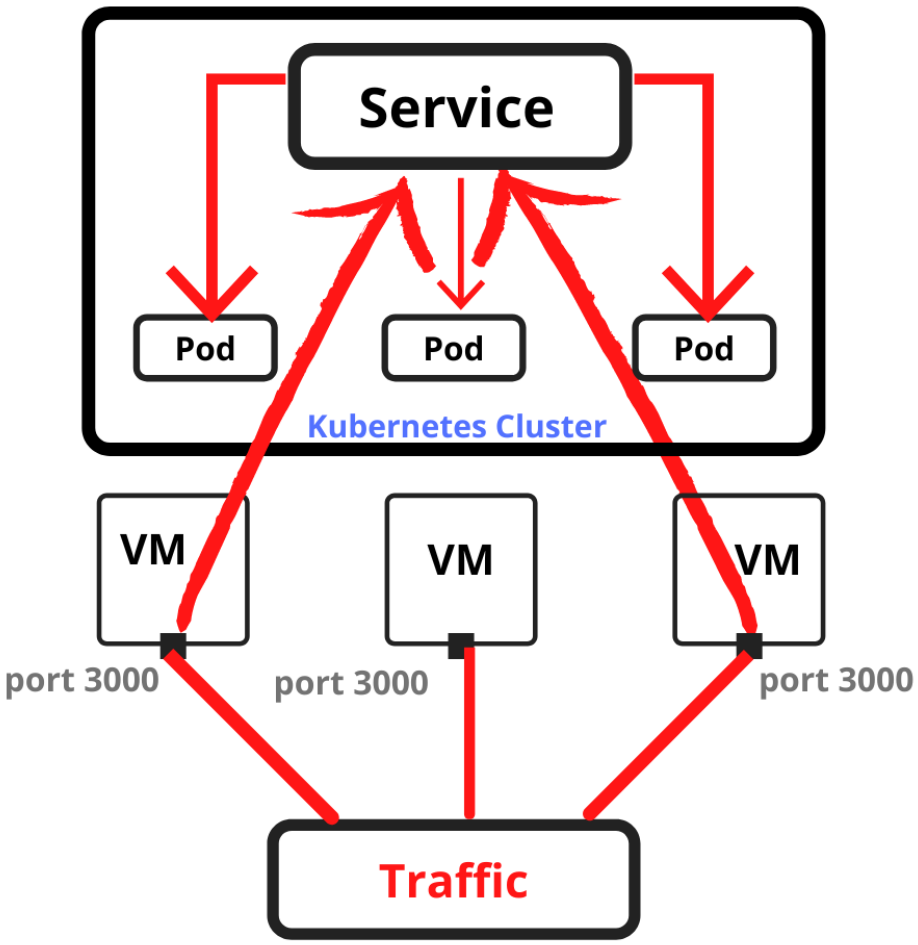
Lab: Exposing Applications using NodePort Services

Introduction:

Exposes the Service on each Node's IP at a static port (the NodePort). A ClusterIP Service, to which the NodePort Service routes, is automatically created. You'll be able to contact the NodePort Service, from outside the cluster, by requesting **<NodeIP>:<NodePort>.**

NodePort are in the **30000-32767** range by default which means a NodePort is unlikely to match a service's intended port (for example, 8080 may be exposed as 31020).



NodePort

Objective:

- **Create NodePort Service**
- **Cleanup**

Ensure that you have logged-in as **root** user on **eoc-controller** node.

1. Create NodePort Service

1.1 Let's **view** the yaml manifest file by executing below command

```
# cat -n ~/kubernetes/service-pods.yml
```

Output:

```
[root@eoc-controller ~]#cat -n ~/kubernetes/service-pods.yml
     1  apiVersion: v1
     2  kind: Pod
     3  metadata:
     4    name: firstpod
     5    labels:
     6      app: hello-world-app
     7  spec:
     8    containers:
     9    - name: first
    10      image: "gcr.io/google-samples/hello-app:2.0"
    11  ---
    12  apiVersion: v1
    13  kind: Pod
    14  metadata:
    15    name: secondpod
    16    labels:
    17      app: hello-world-app
    18  spec:
    19    containers:
    20    - name: second
    21      image: "gcr.io/google-samples/hello-app:2.0"
```

1.2 Let's **create** couple of pods by executing below command.

```
# kubectl apply -f ~/kubernetes/service-pods.yml
```

Output:

```
[root@eoc-controller ~]#kubectl apply -f ~/kubernetes/service-pods.yml
pod/firstpod created
pod/secondpod created
```

1.3 Let's create a Service of type **NodePort,** by executing the below command.

```
# kubectl get pods -o wide
```

Output:

```
[root@eoc-controller ~]#kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP          NODE        NOMINATED NODE   READINESS GATES
firstpod    1/1     Running   0          6m44s   10.32.0.2   eoc-node1   <none>           <none>
secondpod   1/1     Running   0          6m44s   10.32.0.3   eoc-node1   <none>           <none>
```

1.4 Let's **view** the yaml manifest file of NodePort by executing below command

```
# cat -n ~/kubernetes/service-np.yml
```

Output:

```
[root@eoc-controller ~]#cat -n ~/kubernetes/service-np.yml
     1  apiVersion: v1
     2  kind: Service
     3  metadata:
     4    name: np-service
     5  spec:
     6    type: NodePort
     7    selector:
     8      app: hello-world-app
     9    ports:
    10    - protocol: TCP
    11      port: 80
    12      targetPort: 8080
    13      nodePort: 30007
```

1.5 Let's **create** a service of type **NodePort** by executing the below command.

```
# kubectl apply -f ~/kubernetes/service-np.yml
```

Output:

```
[root@eoc-controller ~]#kubectl apply -f ~/kubernetes/service-np.yml
service/np-service created
```

1.6 Let's **view** the service by executing below command.

```
# kubectl get service np-service
```

Output:

```
[root@eoc-controller ~]#kubectl get service np-service
NAME         TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
np-service   NodePort   10.105.169.230   <none>        80:30007/TCP     2m26s
```

1.7 Let's **capture** clusterIP as variable.

```
# CLUSTER_IP=$(kubectl get svc np-service -o
jsonpath='{.spec.clusterIP}')
```

1.8 Let's **access** the pods via cluster-ip.

```
# curl $CLUSTER_IP
```

Output:

```
[root@eoc-controller ~]#curl $CLUSTER_IP
Hello, world!
Version: 2.0.0
Hostname: firstpod
```

1.9 Let's use the **node address** and **node port** to access the application.

Note: Replace 192.168.100.11 IP with your Master Instance Private IP.

Note: To get Private IP execute the command – ifconfig eth0

```
# curl http://PrivateIP:30007
```

Output:

```
[root@eoc-controller ~]#curl http://192.168.100.11:30007
Hello, world!
Version: 2.0.0
Hostname: secondpod
```

Open below URL in your favorite web browser.

Note: To get Public IP execute the command curl ifconfig.io.

```
http://PublicIP:30007
```

2. Cleanup.

2.1 Let's delete the **service** by executing below command.

```
# kubectl delete service np-service
```

Output:

```
[root@eoc-controller ~]#kubectl delete service np-service
service "np-service" deleted
```

2.2 Let's delete the **pods** by executing below command.

```
# kubectl delete pods firstpod secondpod
```

Output:

```
[root@eoc-controller ~]#kubectl delete pods firstpod secondpod
pod "firstpod" deleted
pod "secondpod" deleted
```