Lesson 03 Demo 03

Launching a Pod and Establishing an Associated Service

Objective: To demonstrate how to seamlessly integrate Kubernetes deployments with services to achieve scalable and accessible pod configurations

Tools required: kubeadm, kubectl, kubelet, and containerd

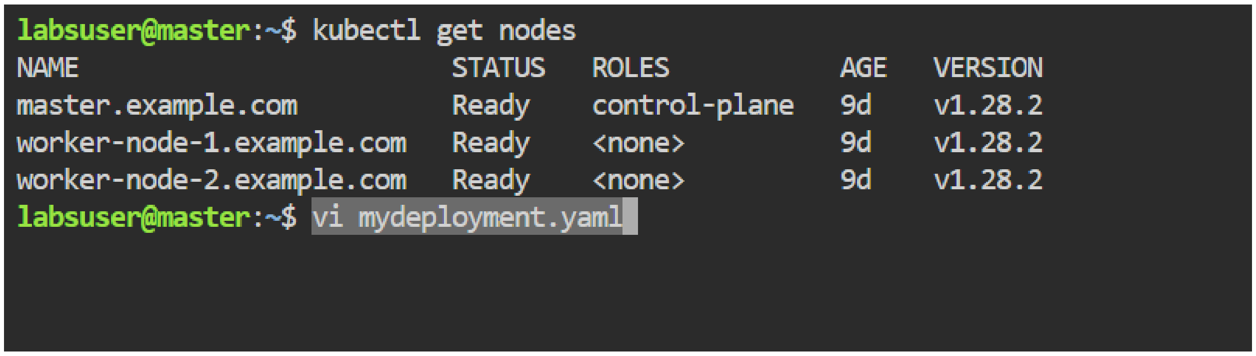
Prerequisites: A Kubernetes cluster should already be set up (refer to the steps provided in Lesson 02, Demo 01 for guidance).

Steps to be followed:

1. Create a deployment object
2. Create a service with label selector for deployment

Step 1: Create a deployment object

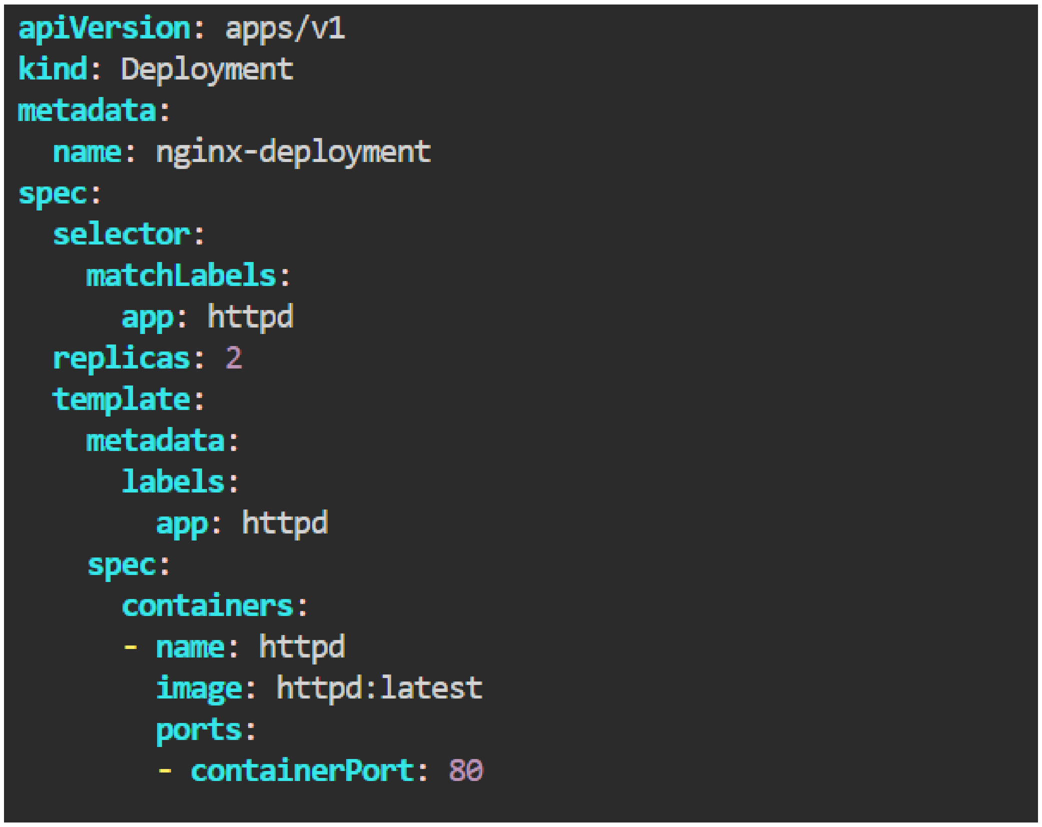
1.1 Create a YAML file for the deployment using the following command:
vi mydeployment.yaml



```
labsuser@master:~$ kubectl get nodes
NAME                         STATUS   ROLES           AGE   VERSION
master.example.com           Ready    control-plane   9d    v1.28.2
worker-node-1.example.com    Ready    <none>          9d    v1.28.2
worker-node-2.example.com    Ready    <none>          9d    v1.28.2
labsuser@master:~$ vi mydeployment.yaml
```

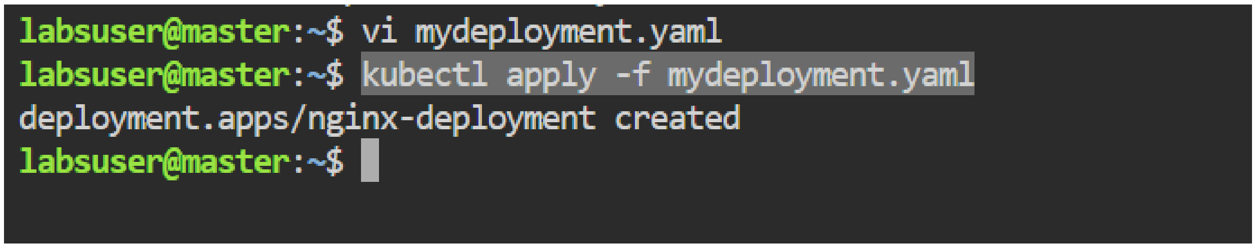
1.2 Add the following code to the **mydeployment.yaml**:

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



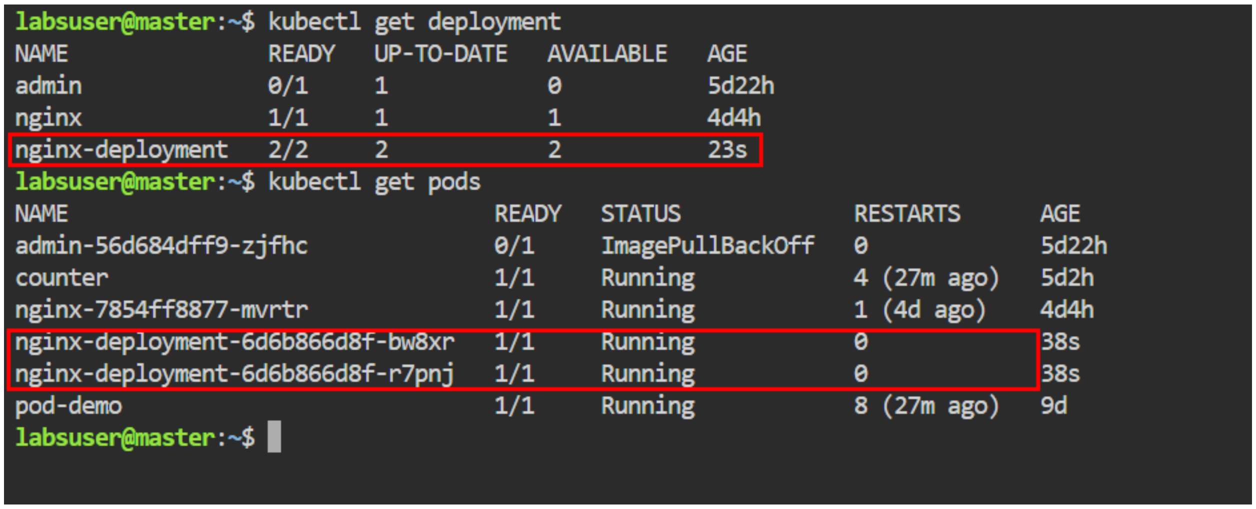
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: httpd
  replicas: 2
  template:
    metadata:
      labels:
        app: httpd
    spec:
      containers:
      - name: httpd
        image: httpd:latest
        ports:
        -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

1.3 Apply the deployment object using the following command:
kubectl apply -f mydeployment.yaml



```
labsuser@master:~$ vi mydeployment.yaml
labsuser@master:~$ kubectl apply -f mydeployment.yaml
deployment.apps/nginx-deployment created
labsuser@master:~$
```

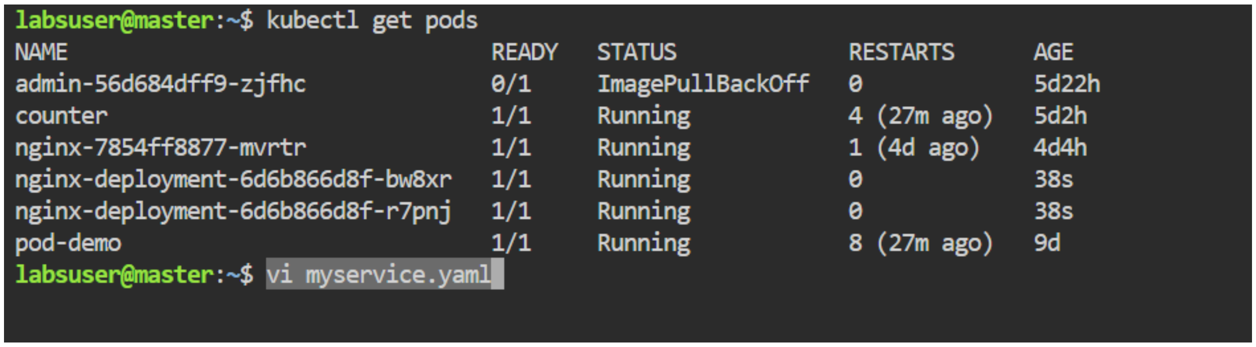
1.4 Verify the deployment and its pods using the following commands:
kubectl get deployment
kubectl get pods



```
labsuser@master:~$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
admin              0/1     1            0           5d22h
nginx              1/1     1            1           4d4h
nginx-deployment   2/2     2            2           23s
labsuser@master:~$ kubectl get pods
NAME                                READY   STATUS             RESTARTS      AGE
admin-56d684dff9-zjfhc              0/1     ImagePullBackOff   0             5d22h
counter                             1/1     Running            4 (27m ago)   5d2h
nginx-7854ff8877-mvrtr              1/1     Running            1 (4d ago)    4d4h
nginx-deployment-6d6b866d8f-bw8xr   1/1     Running            0             38s
nginx-deployment-6d6b866d8f-r7pnj   1/1     Running            0             38s
pod-demo                            1/1     Running            8 (27m ago)   9d
labsuser@master:~$
```

Step 2: Create a service with label selector for deployment

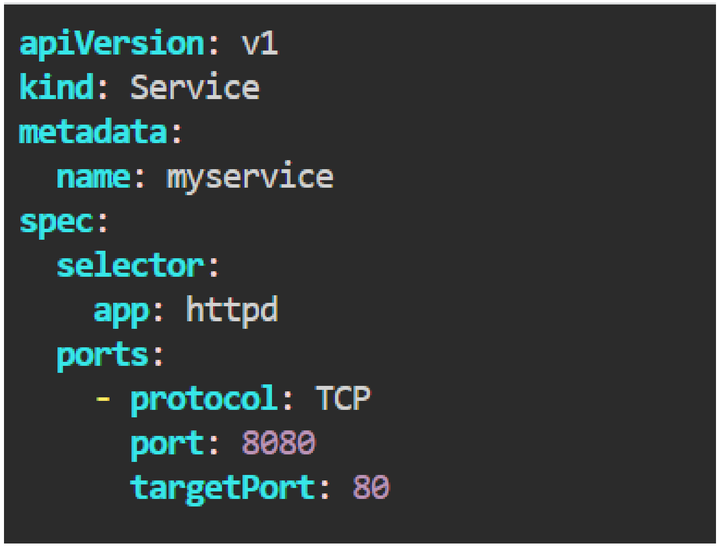
2.1 Create a new YAML file for the service using the command below:
vi myservice.yaml



```
labsuser@master:~$ kubectl get pods
NAME                                READY   STATUS             RESTARTS      AGE
admin-56d684dff9-zjfhc              0/1     ImagePullBackOff   0             5d22h
counter                             1/1     Running            4 (27m ago)   5d2h
nginx-7854ff8877-mvrtr              1/1     Running            1 (4d ago)    4d4h
nginx-deployment-6d6b866d8f-bw8xr   1/1     Running            0             38s
nginx-deployment-6d6b866d8f-r7pnj   1/1     Running            0             38s
pod-demo                            1/1     Running            8 (27m ago)   9d
labsuser@master:~$ vi myservice.yaml
```

2.2 Add the following code to the **myservice.yaml**:

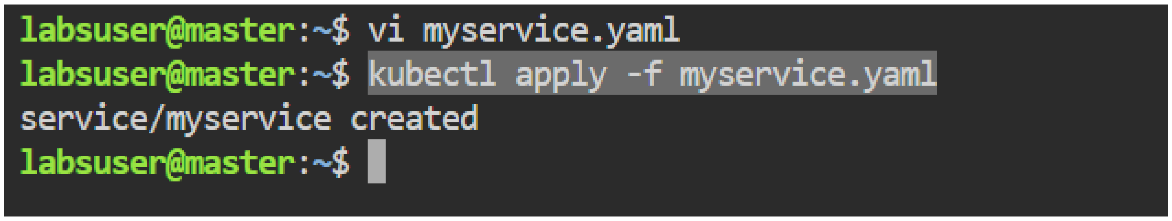
```
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  selector:
    app: httpd
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```



```
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  selector:
    app: httpd
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```

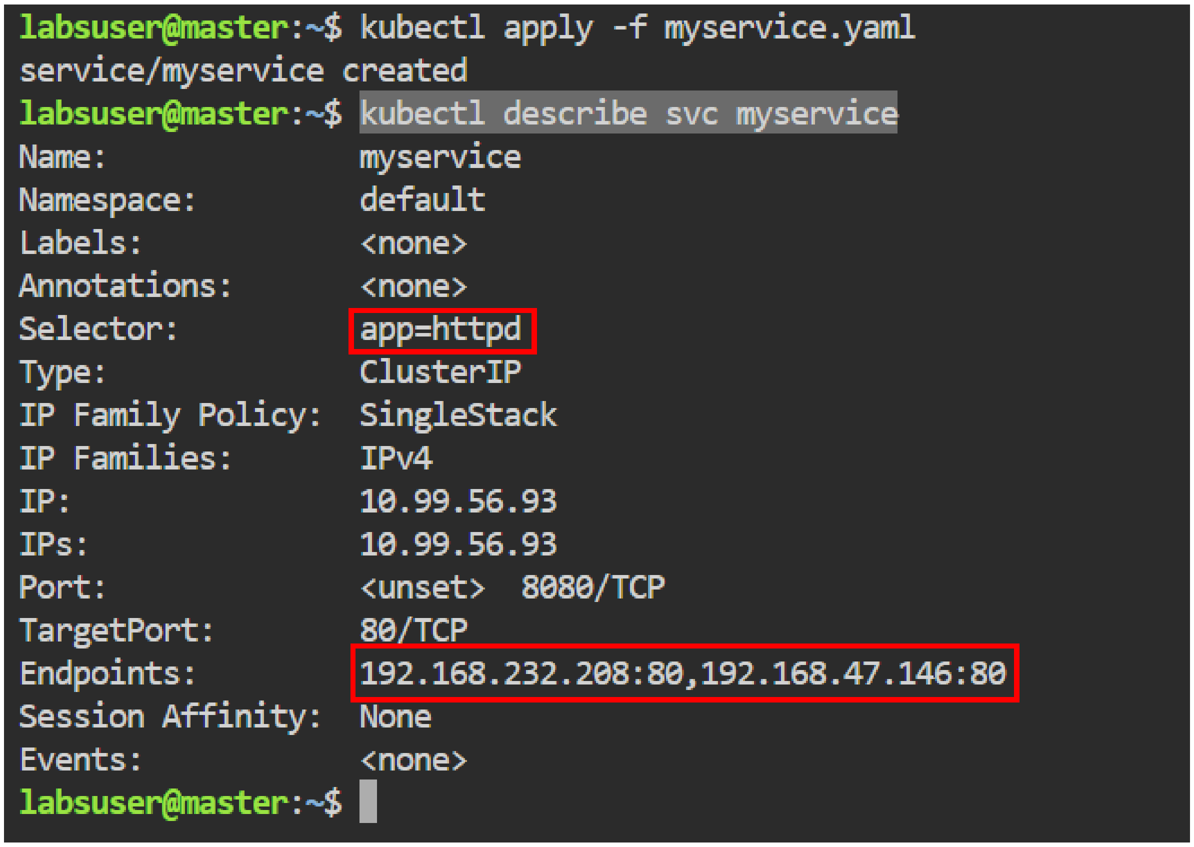
2.3 Apply the service object using the following command:

kubectl apply -f myservice.yaml



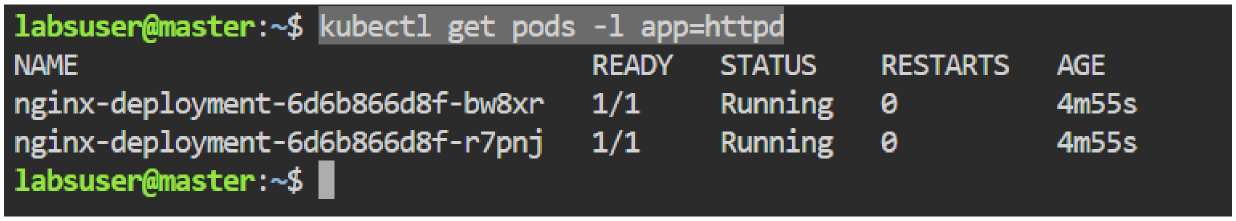
```
labsuser@master:~$ vi myservice.yaml
labsuser@master:~$ kubectl apply -f myservice.yaml
service/myservice created
labsuser@master:~$
```

2.4 Describe the service to verify its connection to the pods using the command below:
kubectl describe svc myservice



```
labsuser@master:~$ kubectl apply -f myservice.yaml
service/myservice created
labsuser@master:~$ kubectl describe svc myservice
Name:              myservice
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=httpd
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.99.56.93
IPs:               10.99.56.93
Port:              <unset>  8080/TCP
TargetPort:        80/TCP
Endpoints:         192.168.232.208:80,192.168.47.146:80
Session Affinity:  None
Events:            <none>
labsuser@master:~$
```

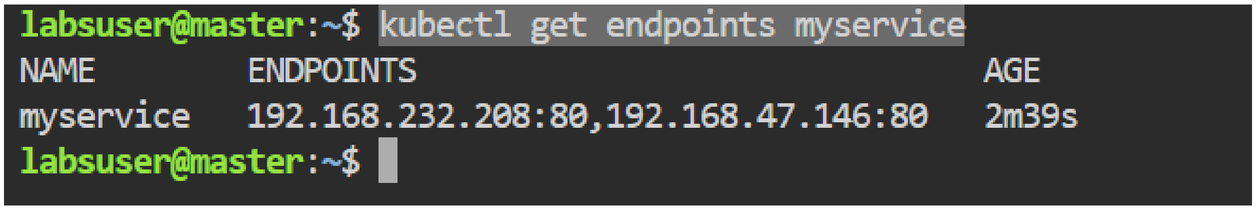
2.5 Check the targeted pods by listing them using the service's selector using the command below:
kubectl get pods -l app=httpd



```
labsuser@master:~$ kubectl get pods -l app=httpd
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6d6b866d8f-bw8xr   1/1     Running   0          4m55s
nginx-deployment-6d6b866d8f-r7pnj   1/1     Running   0          4m55s
labsuser@master:~$
```

2.6 List the service's endpoints to view the IP addresses of the pods it targets:

kubectl get endpoints myservice



```
labsuser@master:~$ kubectl get endpoints myservice
NAME        ENDPOINTS                                   AGE
myservice   192.168.232.208:80,192.168.47.146:80   2m39s
labsuser@master:~$ 
```

By following these steps, you have successfully set up a Kubernetes deployment that offers scalability and a service that ensures the deployed pods accessibility within the cluster or to external clients.