Lesson 05 Demo 01

Deploying a Multi-Port Service Pod

Objective: To deploy a Kubernetes pod using a multi-port service for accessing the deployment through multiple ports

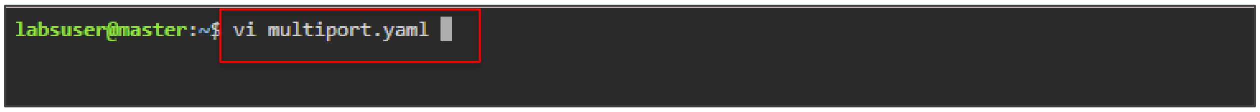
Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Steps to be followed:
1. Create a deployment
2. Define a service
3. Access the deployment from multiple ports

Step 1: Create a deployment

1.1 Create a **multiport.yaml** file using the **vi multiport.yaml** command

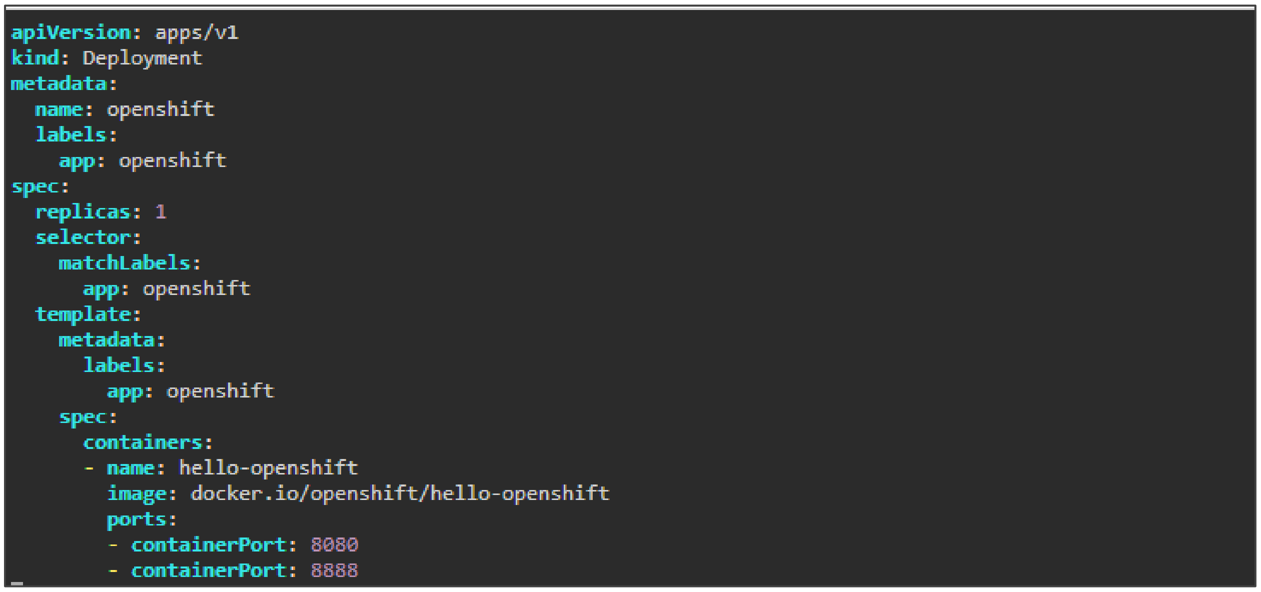


```
labsuser@master:~$ vi multiport.yaml
```

1.2 Add the following code inside the YAML file:

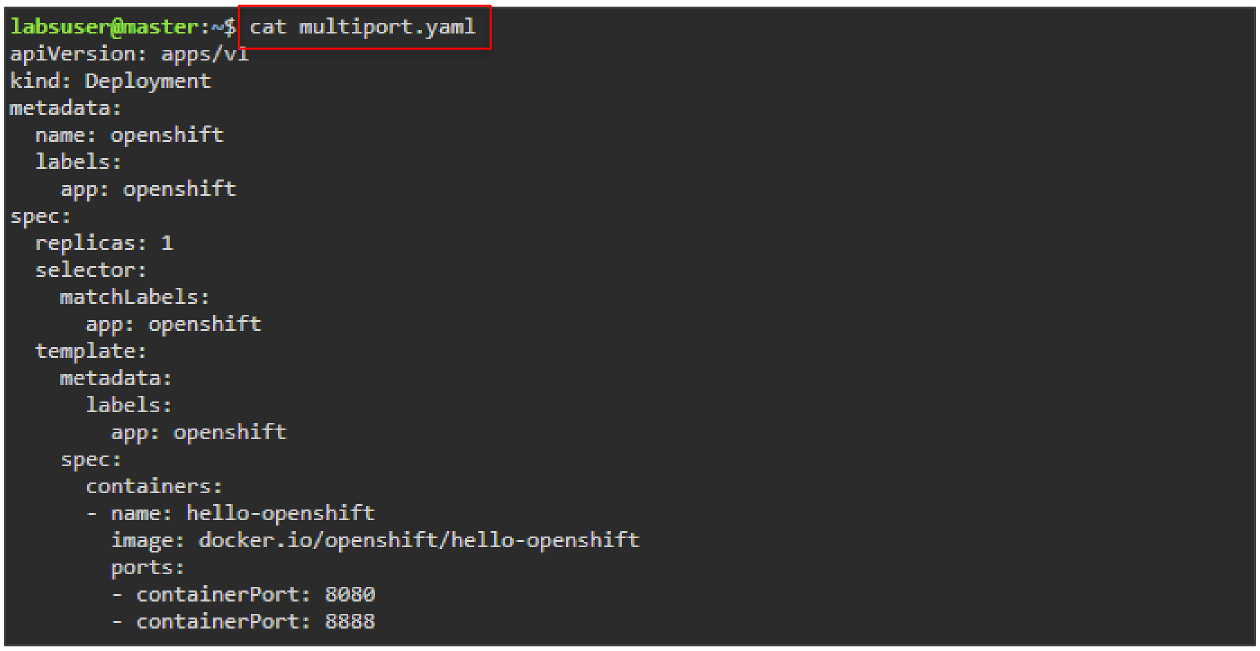
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: openshift
  labels:
    app: openshift
spec:
  replicas: 1
  selector:
    matchLabels:
      app: openshift
  template:
    metadata:
```

```
  labels:
    app: openshift
spec:
  containers:
  - name: hello-openshift
    image: docker.io/openshift/hello-openshift
    ports:
    - containerPort: 8080
    - containerPort: 8888
```



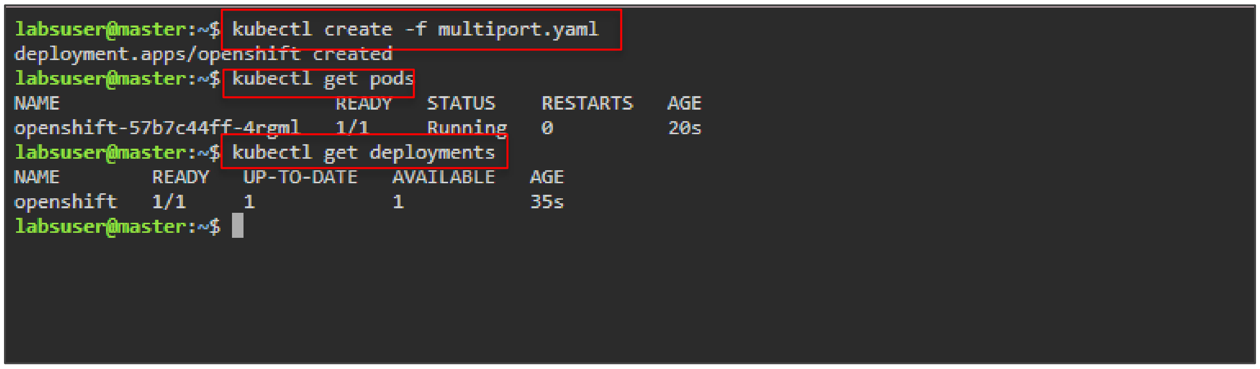
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: openshift
  labels:
    app: openshift
spec:
  replicas: 1
  selector:
    matchLabels:
      app: openshift
  template:
    metadata:
      labels:
        app: openshift
    spec:
      containers:
      - name: hello-openshift
        image: docker.io/openshift/hello-openshift
        ports:
        - containerPort: 8080
        - containerPort: 8888
```

1.3 View the contents of the **multiport.yaml** file using the following command:
cat multiport.yaml



```
labsuser@master:~$ cat multiport.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: openshift
  labels:
    app: openshift
spec:
  replicas: 1
  selector:
    matchLabels:
      app: openshift
  template:
    metadata:
      labels:
        app: openshift
    spec:
      containers:
      - name: hello-openshift
        image: docker.io/openshift/hello-openshift
        ports:
        - containerPort: 8080
        - containerPort: 8888
```

1.4 Execute the following commands to create and verify the deployment:
kubectl create -f multiport.yaml
kubectl get pods
kubectl get deployments

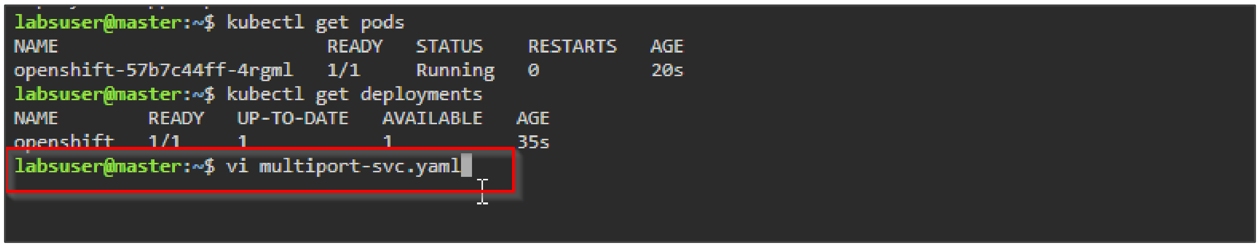


```
labsuser@master:~$ kubectl create -f multiport.yaml
deployment.apps/openshift created
labsuser@master:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
openshift-57b7c44ff-4rgml    1/1     Running   0          20s
labsuser@master:~$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
openshift   1/1     1            1           35s
labsuser@master:~$
```

The OpenShift deployment is successfully created.

Step 2: Define a service

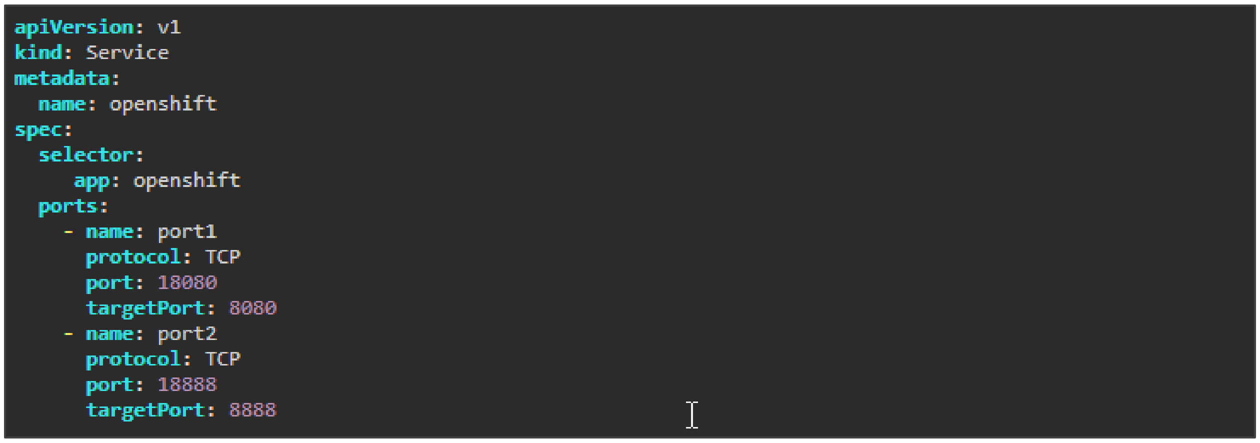
2.1 Create the **multiport-svc.yaml** file using the **vi multiport-svc.yaml** command



```
labsuser@master:~$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
openshift-57b7c44ff-4rgml   1/1     Running   0          20s
labsuser@master:~$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
openshift   1/1     1            1           35s
labsuser@master:~$ vi multiport-svc.yaml
```

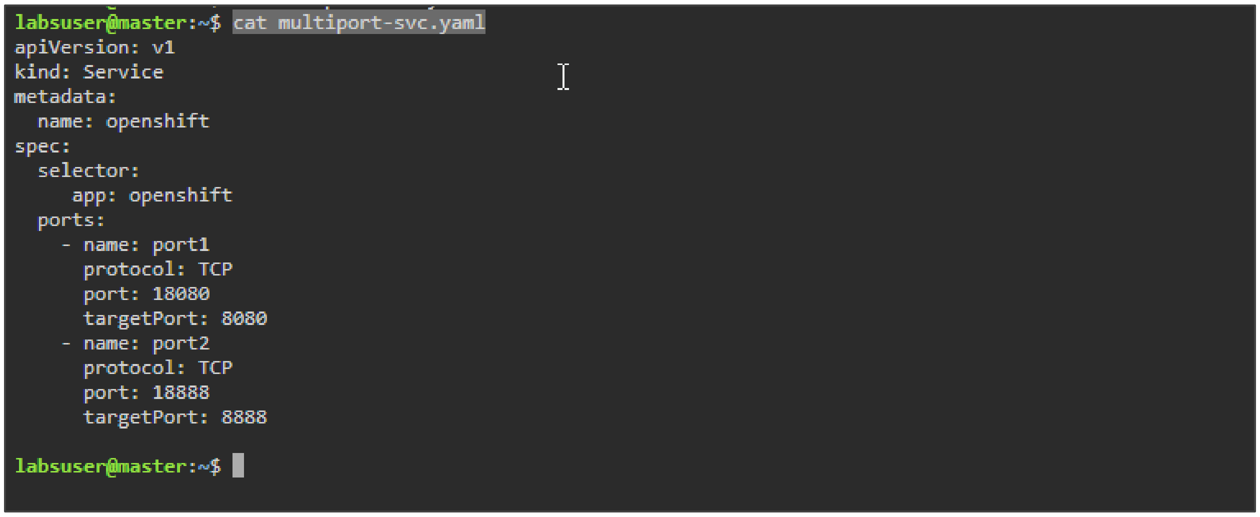
2.2 Add the following code inside the YAML file:

```
apiVersion: v1
kind: Service
metadata:
  name: openshift
spec:
  selector:
     app: openshift
  ports:
    - name: port1
      protocol: TCP
      port: 18080
      targetPort: 8080
    - name: port2
      protocol: TCP
      port: 18888
      targetPort: 8888
```



```yaml
apiVersion: v1
kind: Service
metadata:
  name: openshift
spec:
  selector:
     app: openshift
  ports:
    - name: port1
      protocol: TCP
      port: 18080
      targetPort: 8080
    - name: port2
      protocol: TCP
      port: 18888
      targetPort: 8888
```

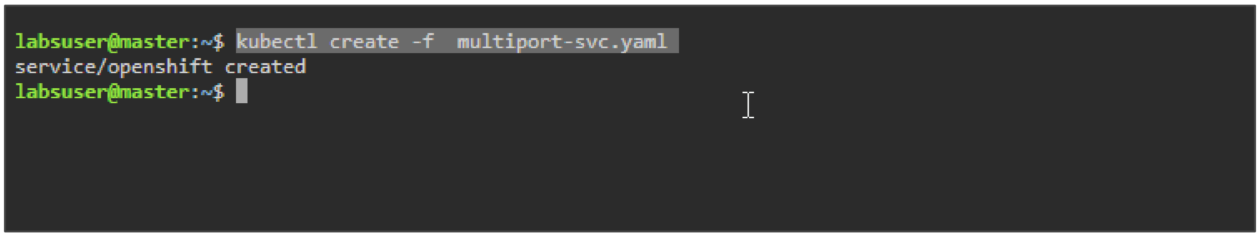
2.3 View the contents of the **multiport-svc.yaml** file using the following command:
cat multiport-svc.yaml



```
labsuser@master:~$ cat multiport-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: openshift
spec:
  selector:
     app: openshift
  ports:
    - name: port1
      protocol: TCP
      port: 18080
      targetPort: 8080
    - name: port2
      protocol: TCP
      port: 18888
      targetPort: 8888

labsuser@master:~$
```

2.4 Create a service for deployment with the following command to access the OpenShift deployment through multiple ports:
kubectl create -f multiport-svc.yaml

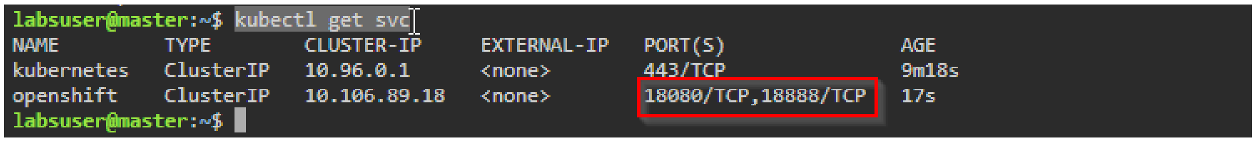


```
labsuser@master:~$ kubectl create -f  multiport-svc.yaml
service/openshift created
labsuser@master:~$
```

The OpenShift deployment is successfully created.

Step 3: Access the deployment from multiple ports

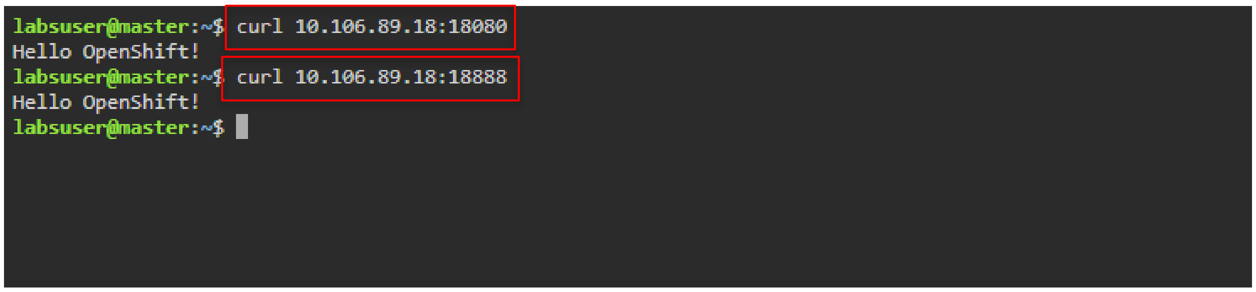
3.1 Run the following command to verify the OpenShift deployment and which ports it can be exposed to:
kubectl get svc



```
labsuser@master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)               AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP               9m18s
openshift    ClusterIP   10.106.89.18    <none>        18080/TCP,18888/TCP   17s
labsuser@master:~$
```

The OpenShift deployment service is available with ports **18080** and **18888**.

3.2 Check if the deployment is accessible using the current cluster IP with different ports:
curl 10.106.89.18:18080
curl 10.106.89.18:18888



```
labsuser@master:~$ curl 10.106.89.18:18080
Hello OpenShift!
labsuser@master:~$ curl 10.106.89.18:18888
Hello OpenShift!
labsuser@master:~$
```

By following these steps, you have successfully deployed a Kubernetes pod using a multi-port service that can be accessed through multiple ports.