Lesson 06 Demo 03

Configuring EndpointSlice

Objective: To configure the EndpointSlice to track network endpoints within a cluster

Tools required: kubeadm, kubectl, kubelet, and containerd

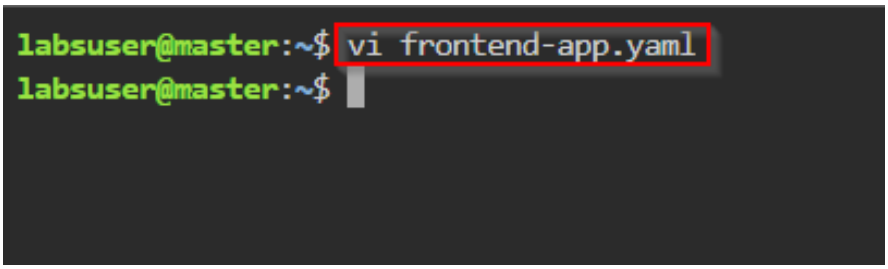
Prerequisites: A Kubernetes cluster should already be set up (refer to the steps in Lesson 02, Demo 01 for guidance).

Steps to be followed:

1. Create a deployment and identify its EndpointSlice
2. Create a YAML file for custom EndpointSlice configuration
3. Create a resource for the custom EndpointSlice configuration

Step 1: Create a deployment and identify its EnpointSlice

1.1 Run the following command and code to create a **frontend-app.yaml** file:
vi frontend-app.yaml




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-app
spec:
  selector:
    matchLabels:
      run: frontend-app
  replicas: 3
  template:
```

```
metadata:
  labels:
    run: frontend-app
spec:
  containers:
  - name: frontend-app
    image: nginx:1.16.1
    ports:
    - containerPort: 80
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-app
spec:
  selector:
    matchLabels:
      run: frontend-app
  replicas: 3
  template:
    metadata:
      labels:
        run: frontend-app
    spec:
      containers:
      - name: frontend-app
        image: nginx:1.16.1
        ports:
        - containerPort: 80
```

1.2 Run the following command to apply the **frontend-app.yaml** file:
kubectl apply -f frontend-app.yaml

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

1.3 Enter the following command to get deploy **frontend-app.yaml** file:
kubectl get deploy frontend-app

```
labsuser@master:~$ vi frontend-app.yaml
labsuser@master:~$ kubectl apply -f frontend-app.yaml
deployment.apps/frontend-app created
labsuser@master:~$ kubectl get deploy frontend-app
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
frontend-app   3/3     3            3           88s
labsuser@master:~$
```

1.4 Run the following command to get pods status:
kubectl get pods -l run=frontend-app

```
labsuser@master:~$ kubectl get pods -l run=frontend-app
NAME                            READY   STATUS    RESTARTS   AGE
frontend-app-58c8686cfc-5bshs   1/1     Running   0          5m9s
frontend-app-58c8686cfc-j448q   1/1     Running   0          5m9s
frontend-app-58c8686cfc-lbrh8   1/1     Running   0          5m9s
labsuser@master:~$
```

We could see 3 pods in running status.

1.5 Enter the following command to deploy frontend-app:
kubectl expose deploy frontend-app --port 80 --target-port 80

```
labsuser@master:~$ kubectl get pods -l run=frontend-app
NAME                            READY   STATUS    RESTARTS   AGE
frontend-app-58c8686cfc-5bshs   1/1     Running   0          5m9s
frontend-app-58c8686cfc-j448q   1/1     Running   0          5m9s
frontend-app-58c8686cfc-lbrh8   1/1     Running   0          5m9s
labsuser@master:~$ kubectl expose deploy frontend-app --port 80 --target-port 80
service/frontend-app exposed
labsuser@master:~$
```

1.6 Run the following command to get cluster IP information:
kubectl get svc frontend-app

```
labsuser@master:~$ kubectl get pods -l run=frontend-app
NAME                            READY   STATUS    RESTARTS   AGE
frontend-app-58c8686cfc-5bshs   1/1     Running   0          5m9s
frontend-app-58c8686cfc-j448q   1/1     Running   0          5m9s
frontend-app-58c8686cfc-lbrh8   1/1     Running   0          5m9s
labsuser@master:~$ kubectl expose deploy frontend-app --port 80 --target-port 80
service/frontend-app exposed
labsuser@master:~$ kubectl get svc frontend-app
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
frontend-app   ClusterIP   10.99.252.131   <none>        80/TCP    80s
labsuser@master:~$
```

1.7 Run the following command to describe svc frontend-app:
kubectl describe svc frontend-app

```
labsuser@master:~$ kubectl describe svc frontend-app
Name:              frontend-app
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          run=frontend-app
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.99.252.131
IPs:               10.99.252.131
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.181.90:80,192.168.181.91:80,192.168.181.92:80
Session Affinity:  None
Events:            <none>
labsuser@master:~$
```

1.8 Enter the following command to identify the service endpoints:

kubectl get ep frontend-app

kubectl get endpointslices

```
labsuser@master:~$ kubectl get ep frontend-app
NAME           ENDPOINTS                                            AGE
frontend-app   192.168.181.90:80,192.168.181.91:80,192.168.181.92:80   54m
labsuser@master:~$ kubectl get endpointslices
NAME                ADDRESSTYPE   PORTS   ENDPOINTS                                      AGE
admin-dbh9j         IPv4          80      192.168.181.89                                 2d2h
frontend-app-q27j7  IPv4          80      192.168.181.90,192.168.181.92,192.168.181.91   55m
kubernetes          IPv4          6443    172.31.36.62                                   11d
my-nginx-sqwd4      IPv4          80      192.168.181.85,192.168.181.87                  149m
labsuser@master:~$
```

Note down the endpointslices named as **frontend-app-q27j7.** The endpointslices suffix is auto generated, so it will vary for cluster, we must identify and copy them to perform the next command.

1.9 Run the following endpointslices command to get the details of frontend-app:

kubectl get endpointslices frontend-app-q27j7 -o yaml

```
labsuser@master:~$ kubectl get endpointslices frontend-app-t9ckl -o yaml
Error from server (NotFound): endpointslices.discovery.k8s.io "frontend-app-t9ckl" not found
labsuser@master:~$ kubectl get endpointslices
NAME                ADDRESSTYPE   PORTS   ENDPOINTS                                      AGE
admin-dbh9j         IPv4          80      192.168.181.89                                 2d2h
frontend-app-q27j7  IPv4          80      192.168.181.90,192.168.181.92,192.168.181.91   59m
kubernetes          IPv4          6443    172.31.36.62                                   11d
my-nginx-sqwd4      IPv4          80      192.168.181.85,192.168.181.87                  153m
labsuser@master:~$ kubectl get endpointslices frontend-app-q27j7 -o yaml
addressType: IPv4
apiVersion: discovery.k8s.io/v1
endpoints:
- addresses:
  - 192.168.181.90
  conditions:
    ready: true
    serving: true
    terminating: false
  nodeName: ip-172-31-29-25
  targetRef:
    kind: Pod
    name: frontend-app-58c8686cfc-lbrh8
    namespace: default
    uid: 68f3c9c2-b885-4b20-98f5-acdb52300ba0
- addresses:
  - 192.168.181.92
  conditions:
    ready: true
    serving: true
```

```
    name: frontend-app-58c8686cfc-j448q
    namespace: default
    uid: 3e4083a5-b595-4a98-b6be-456d2adde144
kind: EndpointSlice
metadata:
  annotations:
    endpoints.kubernetes.io/last-change-trigger-time: "2023-11-06T08:35:55Z"
  creationTimestamp: "2023-11-06T08:35:55Z"
  generateName: frontend-app-
  generation: 1
  labels:
    endpointslice.kubernetes.io/managed-by: endpointslice-controller.k8s.io
    kubernetes.io/service-name: frontend-app
  name: frontend-app-q27j7
  namespace: default
  ownerReferences:
  - apiVersion: v1
    blockOwnerDeletion: true
    controller: true
    kind: Service
    name: frontend-app
    uid: 094afa0f-f2bc-47a2-b169-69b67679abf4
  resourceVersion: "63818"
  uid: a236434c-68b5-46c3-9bc3-07c19533a276
ports:
- name: ""
  port: 80
  protocol: TCP
labsuser@master:~$
```

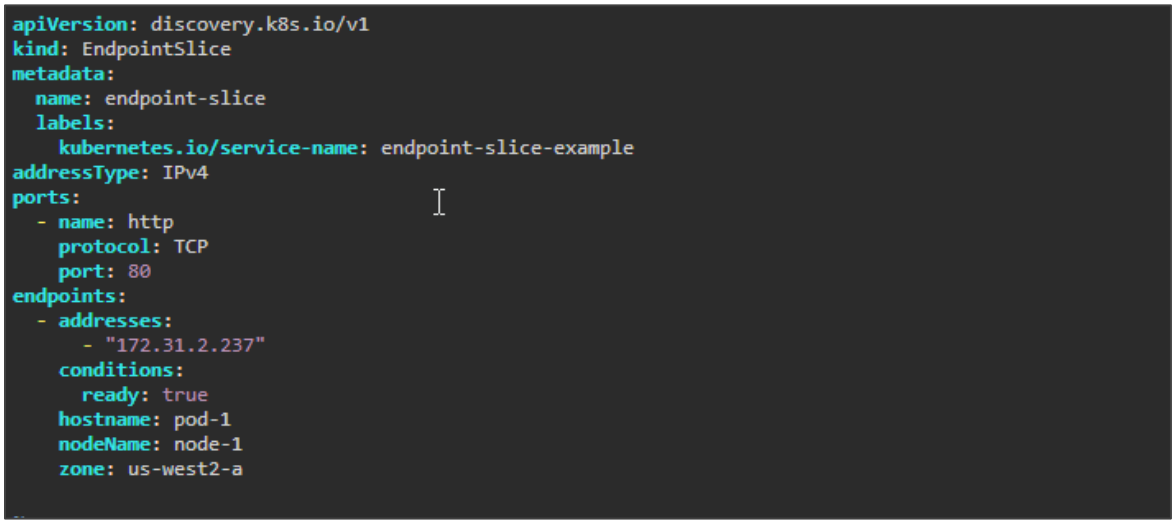
Step 2: Create a YAML file for custom EndpointSlice configuration

2.1 In the master node, create a configuration file for the EndpointSlice using the following command:
vi endpoint-slice.yaml

```
labsuser@master:~$ vi endpoint-slice.yaml
```

2.2 Add the following code to the configuration file:

```
apiVersion: discovery.k8s.io/v1
kind: EndpointSlice
metadata:
 name: endpoint-slice
 labels:
  kubernetes.io/service-name: endpoint-slice-example
addressType: IPv4
ports:
 - name: http
  protocol: TCP
  port: 80
endpoints:
 - addresses:
   - "172.31.2.237"
  conditions:
   ready: true
  hostname: pod-1
  nodeName: node-1
  zone: us-west2-a
```



2.3 View the content of the **endpoint-slice.yaml** file using the following command:
cat endpoint-slice.yaml

```
labsuser@master:~$ cat endpoint-slice.yaml
apiVersion: discovery.k8s.io/v1
kind: EndpointSlice
metadata:
  name: endpoint-slice
  labels:
    kubernetes.io/service-name: endpoint-slice-example
addressType: IPv4
ports:
  - name: http
    protocol: TCP
    port: 80
endpoints:
  - addresses:
      - "172.31.2.237"
    conditions:
      ready: true
    hostname: pod-1
    nodeName: node-1
    zone: us-west2-a

labsuser@master:~$
```

Step 3: Create a resource for the custom EndpointSlice configuration

3.1 Create a resource for the EndpointSlice using the following command:
kubectl apply -f endpoint-slice.yaml

```
labsuser@master:~$ kubectl apply -f endpoint-slice.yaml
endpointslice.discovery.k8s.io/endpoint-slice created
labsuser@master:~$
```

3.2 Check the created resource using the following command:
kubectl get endpointslices

```
labsuser@master:~$ kubectl get endpointslices
NAME              ADDRESSTYPE   PORTS       ENDPOINTS         AGE
endpoint-slice    IPv4          80          172.31.2.237      2m14s
kubernetes        IPv4          6443        172.31.42.117     111m
openshift-bjxk4   IPv4          8888,8080   192.168.232.193   102m
labsuser@master:~$
```

3.3 View the details of the created EndpointSlice using the following command:
kubectl describe endpointslices endpoint-slice

```
labsuser@master:~$ kubectl describe endpointslices endpoint-slice
Name:         endpoint-slice
Namespace:    default
Labels:       kubernetes.io/service-name=endpoint-slice-example
Annotations:  <none>
AddressType:  IPv4
Ports:
  Name  Port  Protocol
  ----  ----  --------
  http  80    TCP
Endpoints:
  - Addresses:  172.31.2.237
    Conditions:
      Ready:    true
    Hostname:   pod-1
    NodeName:   node-1
    Zone:       us-west2-a
Events:         <none>
labsuser@master:~$
```

By following the above steps, you have successfully configured an EndpointSlice file that tracks network endpoints within a cluster.