.

Lesson 05 Demo 06

Limiting the Traffic to an Application

Objective: To limit network traffic to a specific application within a Kubernetes cluster for controlled access and resource utilization

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. Launch the API server
2. Configure the YAML file for the network policy
3. Verify the network policy
4. Clear the created resources

Step 1: Launch the API server

1.1 Launch the API server using the following command:
kubectl run apiserver --image=nginx --labels="app=simplilearn,role=cka" --expose --port=80

```
labsuser@master:~$ kubectl run apiserver --image=nginx --labels="app=simplilearn,role=cka" --expose --port=80
service/apiserver created
pod/apiserver created
labsuser@master:~$
```

Note: The above command creates a REST API server with the labels **app=simplilearn** and **role=cka**.

.

Step 2: Configure the YAML file for the network policy

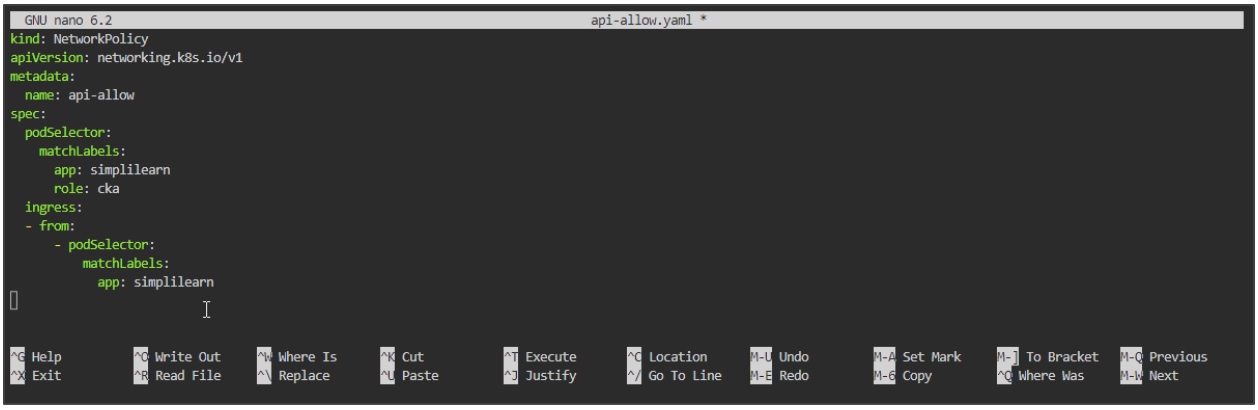
2.1 Create the YAML file using the following command:
nano api-allow.yaml

```
labsuser@master:~$ kubectl run apiserver --image=nginx --labels="app=simplilearn,role=cka" --expose --port=80
service/apiserver created
pod/apiserver created
labsuser@master:~$ nano api-allow.yaml
```

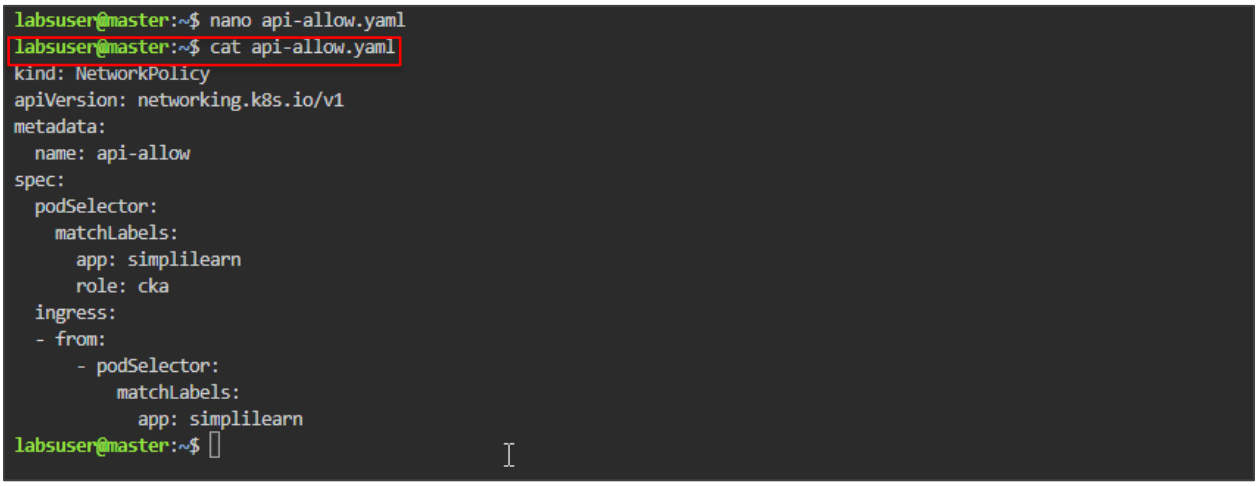
2.2 Add the following code to the **api-allow.yaml** file:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: api-allow
spec:
  podSelector:
    matchLabels:
      app: simplilearn
      role: cka
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: simplilearn
```

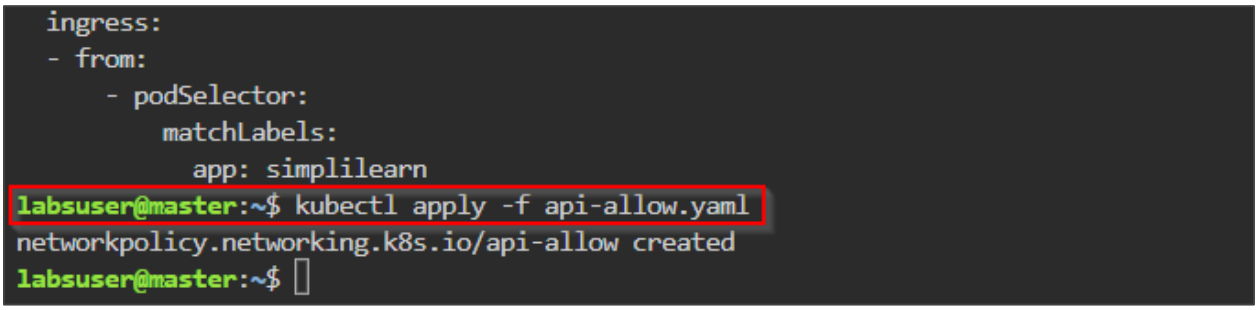
.



2.3 Use the **cat** command to validate the content of the **api-allow.yaml** file



2.4 Create the network policy for traffic allocation using the following command:
kubectl apply -f api-allow.yaml



.

2.5 Validate if the policy was created successfully using the following command:
kubectl get networkpolicy

```
labsuser@master:~$ kubectl apply -f api-allow.yaml
networkpolicy.networking.k8s.io/api-allow created
labsuser@master:~$ kubectl get networkpolicy
NAME        POD-SELECTOR               AGE
api-allow   app=simplilearn,role=cka   78s
labsuser@master:~$
```

Step 3: Verify the network policy

3.1 Validate if the network policy blocks traffic by launching a pod without the **app=simplilearn** label using the following commands:
kubectl run test-$RANDOM --rm -i -t --image=alpine -- sh
wget -qO- --timeout=2 http://apiserver

```
labsuser@master:~$ kubectl get networkpolicy
NAME        POD-SELECTOR               AGE
api-allow   app=simplilearn,role=cka   78s
labsuser@master:~$ kubectl run test-$RANDOM --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://apiserver
wget: download timed out
/ #
```

The traffic is blocked.

Note: Type **exit** and press the **enter** key to exit the command prompt

.

3.2 Verify if the network policy allows traffic by running the pod with the **app=simplilearn** label using the following commands:

kubectl run test-$RANDOM --rm -i -t --image=alpine labels="app=simplilearn,role=ckad" -- sh

wget -qO- --timeout=2 http://apiserver

```
/ # wget -qO- --timeout=2 http://apiserver
wget: download timed out
/ # exit
Session ended, resume using 'kubectl attach test-7421 -c test-7421 -i -t' command when the pod is running
pod "test-7421" deleted
labsuser@master:~$ kubectl run test-$RANDOM --rm -i -t --image=alpine --labels="app=simplilearn,role=ckad" -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://apiserver
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
```

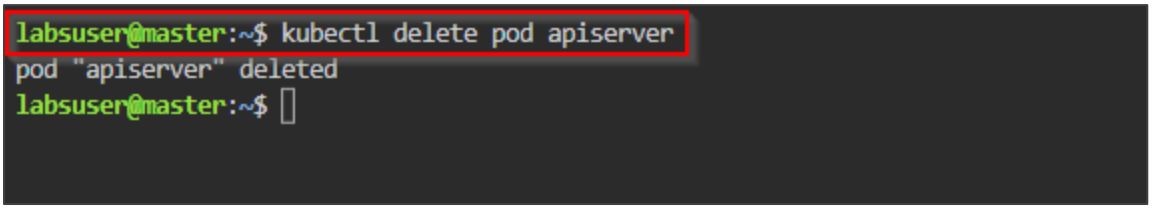
```
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

.

Step 4: Clear the created resources

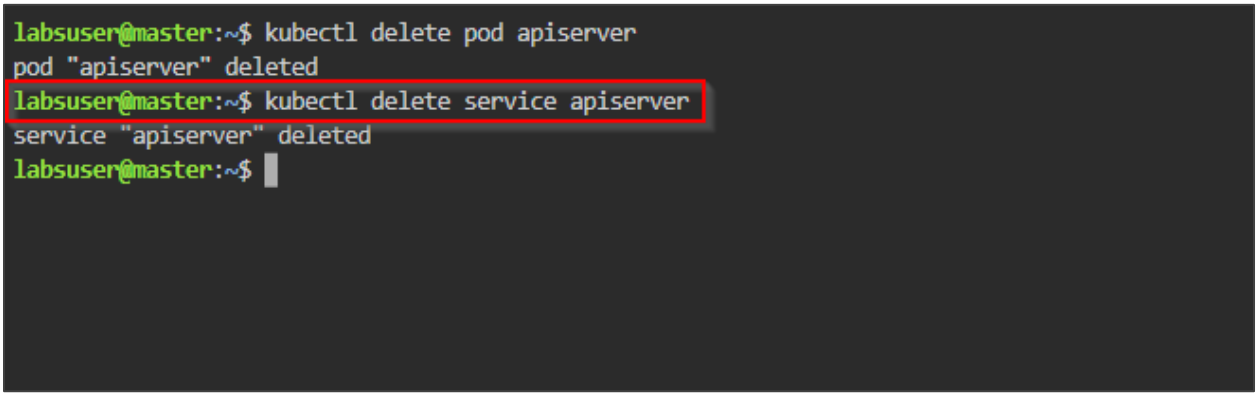
4.1 Delete the pod using the following command:

kubectl delete pod apiserver



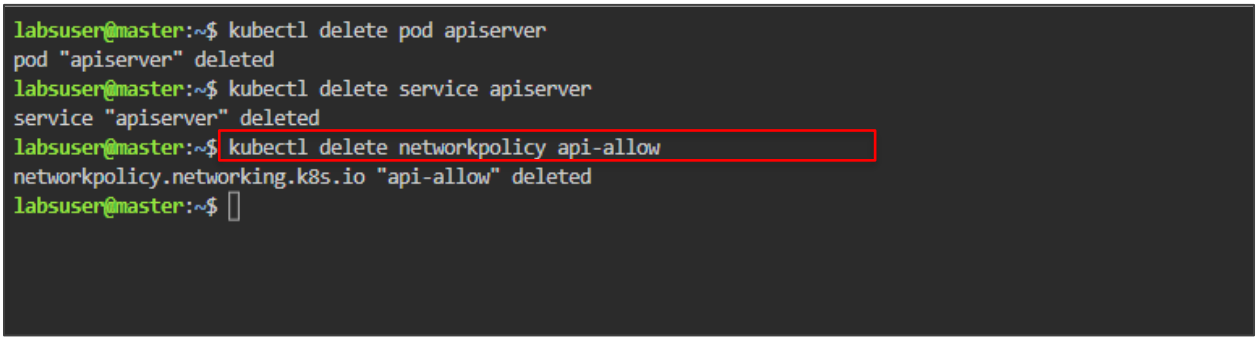
4.2 Delete the service using the following command:

kubectl delete service apiserver



4.3 Delete the network policy using the following command:

kubectl delete networkpolicy api-allow



By following these steps, you have successfully restricted all network traffic to a specific application within a Kubernetes cluster, promoting controlled access and resource utilization.