Lesson 06 Demo 07

Blocking Traffic from Other Namespaces

Objective: To implement a Kubernetes network policy for blocking all inter-namespace traffic

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. Launch the web service
2. Configure the YAML file for the network policy
3. Create a new namespace
4. Verify the network policy
5. Clear the created resources

Step 1: Launch the web service

1.1 Launch the web service using the following command:

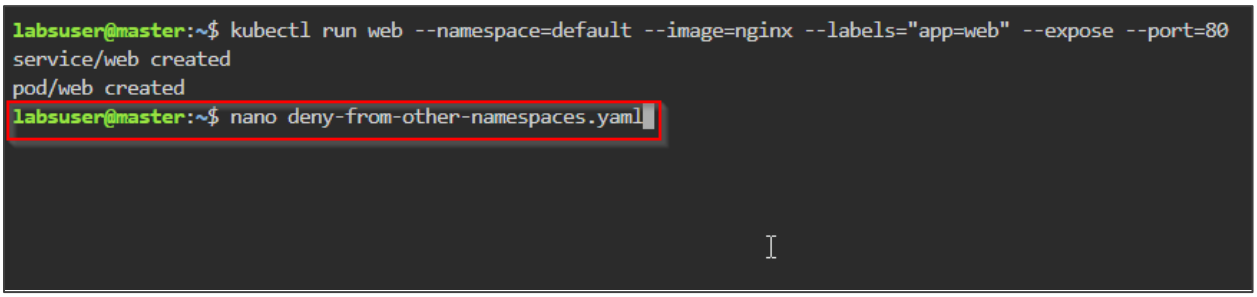
kubectl run web --namespace=default --image=nginx --labels="app=web" --expose --port=80

```
labsuser@master:~$ kubectl run web --namespace=default --image=nginx --labels="app=web" --expose --port=80
service/web created
pod/web created
labsuser@master:~$
```

Step 2: Configure the YAML file for the network policy

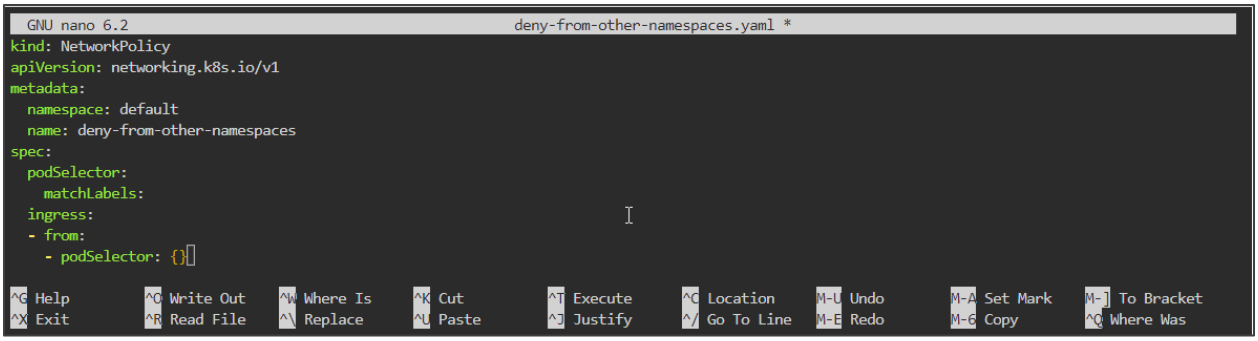
2.1 Create the YAML file using the following command:

nano deny-from-other-namespaces.yaml



2.2 Add the following code to the **deny-from-other-namespaces.yaml** file:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
 namespace: default
 name: deny-from-other-namespaces
spec:
 podSelector:
  matchLabels:
 ingress:
 - from:
  - podSelector: {}
```



2.3 Use the **cat** command to validate the content of the **deny-from-other-namespaces.yaml** file

```
pod/web created
labsuser@master:~$ nano deny-from-other-namespaces.yaml
labsuser@master:~$ cat deny-from-other-namespaces.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: default
  name: deny-from-other-namespaces
spec:
  podSelector:
    matchLabels:
  ingress:
  - from:
    - podSelector: {}
labsuser@master:~$
```

2.4 Create the network policy for traffic allocation using the following command:

kubectl apply -f deny-from-other-namespaces.yaml

```
spec:
  podSelector:
    matchLabels:
  ingress:
  - from:
    - podSelector: {}
labsuser@master:~$ kubectl apply -f deny-from-other-namespaces.yaml
networkpolicy.networking.k8s.io/deny-from-other-namespaces created
labsuser@master:~$
```

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

kubectl get networkpolicy

```
labsuser@master:~$ kubectl apply -f deny-from-other-namespaces.yaml
networkpolicy.networking.k8s.io/deny-from-other-namespaces created
labsuser@master:~$ kubectl get networkpolicy
NAME                          POD-SELECTOR      AGE
deny-from-other-namespaces    <none>            110s
simplilearn-deny-all          app=simplilearn   34h
labsuser@master:~$
```

Step 3: Create a new namespace

3.1 Create a new namespace named **ckad** using the following command:

kubectl create namespace ckad

```
deny-from-other-namespaces   <none>           110s
simplilearn-deny-all         app=simplilearn   34h
labsuser@master:~$ kubectl create namespace ckad
namespace/ckad created
labsuser@master:~$
```

Step 4: Verify the network policy

4.1 Validate the network policy for the namespace **ckad** using the following command:

kubectl run test-$RANDOM --namespace=ckad --rm -i -t --image=alpine -- sh
wget -qO- --timeout=2 http://web.default

```
labsuser@master:~$ kubectl create namespace ckad
namespace/ckad created
labsuser@master:~$ kubectl run test-$RANDOM --namespace=ckad --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://web.default
wget: download timed out
/ #
```

The network policy blocks the traffic from the **ckad** namespace.

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

4.2 Verify if the network policy allows traffic for any other pods in the default namespace using the following commands:

kubectl run test-$RANDOM --namespace=default --rm -i -t --image=alpine -- sh
wget -qO- --timeout=2 http://web.default

```
labsuser@master:~$ kubectl run test-$RANDOM --namespace=ckad --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://web.default
wget: download timed out
/ # exit
Session ended, resume using 'kubectl attach test-25989 -c test-25989 -i -t' command when the pod is running
pod "test-25989" deleted
labsuser@master:~$ kubectl run test-$RANDOM --namespace=default --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://web.default
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
```

```
</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>
/ # exit
Session ended, resume using 'kubectl attach test-27664 -c test-27664 -i -t' command when the pod is running
pod "test-27664" deleted
labsuser@master:~$
```

Step 5: Clear the created resources

5.1 Delete the pod using the following command:

kubectl delete pod web -n default

```
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ # exit
Session ended, resume using 'kubectl attach test-27664 -c test-27664 -i -t' command when the pod is running
pod "test-27664" deleted
labsuser@master:~$ kubectl delete pod web -n default
pod "web" deleted
labsuser@master:~$
```

5.2 Delete the service using the following command:

kubectl delete service web -n default

```
labsuser@master:~$ kubectl delete pod web -n default
pod "web" deleted
labsuser@master:~$ kubectl delete service web -n default
service "web" deleted
labsuser@master:~$
```

5.3 Delete the network policy using the following command:

kubectl delete networkpolicy deny-from-other-namespaces -n default

```
labsuser@master:~$ kubectl delete pod web -n default
pod "web" deleted
labsuser@master:~$ kubectl delete service web -n default
service "web" deleted
labsuser@master:~$ kubectl delete networkpolicy deny-from-other-namespaces -n default
networkpolicy.networking.k8s.io "deny-from-other-namespaces" deleted
labsuser@master:~$
```

5.4 Delete the namespace using the following command:

kubectl delete namespace ckad

```
labsuser@master:~$ kubectl delete pod web -n default
pod "web" deleted
labsuser@master:~$ kubectl delete service web -n default
service "web" deleted
labsuser@master:~$ kubectl delete networkpolicy deny-from-other-namespaces -n default
networkpolicy.networking.k8s.io "deny-from-other-namespaces" deleted
labsuser@master:~$ kubectl delete namespace ckad
namespace "ckad" deleted
labsuser@master:~$
```

By following these steps, you have successfully set up and verified a Kubernetes network policy that blocks inter-namespace traffic.