Exercise 6.5: Testing the Policy

1. Now that we have tested both ingress and egress we can implement the network policy.

```
student@ckad-1:~/app2$ kubectl create -f ~/app2/allclosed.yaml
networkpolicy.networking.k8s.io/deny-default created
```

2. Use the ingress and egress tests again. Three of the four should eventually timeout. Start by testing from outside the cluster, and interrupt if you get tired of waiting.

```
[user@laptop ~]$ curl http://35.184.219.5:32000
curl: (7) Failed to connect to 35.184.219.5 port
32000: Connection timed out
```

3. Then test from the host to the container.

```
student@ckad-1:~/app2$ curl http://10.97.96.75:80
curl: (7) Failed to connect to 10.97.96.75 port 80: Connection timed out
```

4. Now test egress. From container to container should work, as the filter is outside of the pod. Then test egress to an external web page. It should eventually timeout.

```
student@ckad-1:~/app2$ kubectl exec -it -c busy secondapp sh
```

On Container

```
/ $ nc -vz 127.0.0.1 80
127.0.0.1 (127.0.0.1:80) open

/ $ nc -vz www.linux.com 80
nc: bad address 'www.linux.com'

/ $ exit
```

5. Update the NetworkPolicy and comment out the Egress line. Then replace the policy.

```
student@ckad-1:~/app2$ vim ~/app2/allclosed.yaml
```

allclosed.yaml

```
1 ....
2 spec:
3   podSelector: {}
4   policyTypes:
5   - Ingress
6 #  - Egress            #<-- Comment out this line
```

```
student@ckad-1:~/app2$ kubectl replace -f ~/app2/allclosed.yaml
```

```
networkpolicy.networking.k8s.io/deny-default replaced
```

6. Test egress access to an outside site. Get the IP address of the **eth0** inside the container while logged in. The IP is 192.168.55.91 in the example below, yours may be different.

```
student@ckad-1:~/app2$ kubectl exec -it -c busy secondapp sh
```

On Container

```
/ $ nc -vz www.linux.com 80
www.linux.com (151.101.185.5:80) open

/ $ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
4: eth0@if59: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 1e:c8:7d:6a:96:c3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.55.91/32 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::1cc8:7dff:fe6a:96c3/64 scope link
       valid_lft forever preferred_lft forever

/ $ exit
```

7. Now add a selector to allow ingress to only the nginx container. Use the IP from the **eth0** range.

```
student@ckad-1:~/app2$ vim ~/app2/allclosed.yaml
```

allclosed.yaml

```
1  <output_omitted>
2   policyTypes:
3    - Ingress
4    ingress:                    #<-- Add this and following three lines
5    - from:
6      - ipBlock:
7          cidr: 192.168.0.0/16
8  # - Egress
```

8. Recreate the policy, and verify its configuration.

```
student@ckad-1:~/app2$ kubectl replace -f ~/app2/allclosed.yaml
networkpolicy.networking.k8s.io/deny-default replaced

student@ckad-1:~/app2$ kubectl get networkpolicy
NAME           POD-SELECTOR     AGE
deny-default   <none>           3m2s

student@ckad-1:~/app2$ kubectl get networkpolicy -o yaml
```

```
apiVersion: v1
items:
- apiVersion: extensions/v1beta1
  kind: NetworkPolicy
  metadata:
<output_omitted>
```

9. Test access to the container both using **curl** as well as **ping**, the IP address to use was found from the IP inside the container. You may need to install **iputils-ping** or other software to use **ping**.

```
student@ckad-1:~/app2$ curl http://192.168.55.91
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<output_omitted>
```

```
student@ckad-1:~/app2$ ping -c5 192.168.55.91
```

```
PING 192.168.55.91 (192.168.55.91) 56(84) bytes of data.
64 bytes from 192.168.55.91: icmp_seq=1 ttl=63 time=1.11 ms
64 bytes from 192.168.55.91: icmp_seq=2 ttl=63 time=0.352 ms
64 bytes from 192.168.55.91: icmp_seq=3 ttl=63 time=0.350 ms
64 bytes from 192.168.55.91: icmp_seq=4 ttl=63 time=0.359 ms
64 bytes from 192.168.55.91: icmp_seq=5 ttl=63 time=0.295 ms

--- 192.168.55.91 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4054ms
rtt min/avg/max/mdev = 0.295/0.495/1.119/0.312 ms
```

10. Update the policy to only allow ingress for TCP traffic on port 80, then test with **curl**, which should work. The ports entry should line up with the from entry a few lines above.

```
student@ckad-1:~/app2$ vim ~/app2/allclosed.yaml
```

allclosed.yaml

```
<output_omitted>
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: 192.168.0.0/16
    ports:                    #<-- Add this and two following lines
    - port: 80
      protocol: TCP
#  - Egress
```

```
student@ckad-1:~/app2$ kubectl replace -f ~/app2/allclosed.yaml
```

```
networkpolicy.networking.k8s.io/deny-default replaced
```

```
student@ckad-1:~/app2$ curl http://192.168.55.91
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<output_omitted>
```

11. All five pings should fail, with zero received.

```
student@ckad-1:~/app2$ ping -c5 192.168.55.91
PING 192.168.55.91 (192.168.55.91) 56(84) bytes of data.

--- 192.168.55.91 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4098ms
```