Exercise 6.5: Testing the Policy

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

```
student@cp:~/app2$ kubectl create -f $HOME/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@cp:~/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@cp:~/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@cp:~/app2$ vim $HOME/app2/allclosed.yaml
```

allclosed.yaml

```yaml
....
spec:
  podSelector: {}
  policyTypes:
  - Ingress
#  - Egress              #<-- Comment out this line
```

```
student@cp:˜/app2$ kubectl replace -f $HOME/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@cp:˜/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 an ingress rule to allow ingress to only the nginx container. Use the IP from the **eth0** range.

```
student@cp:˜/app2$ vim $HOME/app2/allclosed.yaml
```

allclosed.yaml

```yaml
<output_omitted>
 policyTypes:
 - Ingress
 ingress:                    #<-- Add this and following three lines
 - from:
   - podSelector: {}
# - Egress
```

8. Recreate the policy, and verify its configuration.

```
student@cp:˜/app2$ kubectl replace -f $HOME/app2/allclosed.yaml
```

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

```
student@cp:˜/app2$ kubectl get networkpolicy
```

```
NAME           POD-SELECTOR     AGE
deny-default   <none>           3m2s
```

```
student@cp:˜/app2$ kubectl get networkpolicy -o yaml
```

```
apiVersion: v1
items:
- apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
<output_omitted>
```

9. Test access to the container using **ping**, use the IP address of the pod

```
student@cp:˜/app2$ kubectl run -it test --rm=true --image alpine -- ping -c5 192.168.55.91
```

```
If you don't see a command prompt, try pressing enter.
64 bytes from 192.168.1.45: seq=1 ttl=63 time=0.094 ms
64 bytes from 192.168.1.45: seq=2 ttl=63 time=0.102 ms
64 bytes from 192.168.1.45: seq=3 ttl=63 time=0.103 ms
64 bytes from 192.168.1.45: seq=4 ttl=63 time=0.111 ms

--- 192.168.1.45 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.094/0.216/0.672 ms
Session ended, resume using 'kubectl attach test -c test -i -t' command when the pod is running
pod "test" deleted
```

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@cp:˜/app2$ vim $HOME/app2/allclosed.yaml
```

allclosed.yaml

```yaml
<output_omitted>
  - Ingress
  ingress:
  - from:
    - podSelector: {}
    ports:                       #<-- Add this and two following lines
    - port: 80
      protocol: TCP
#  - Egress
```

```
student@cp:˜/app2$ kubectl replace -f $HOME/app2/allclosed.yaml
```

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

```
student@cp:˜/app2$ kubectl run -it test --rm=true --image alpine -- ping -c5 192.168.55.91
```

All five pings should fail, with zero received.

```
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
```

11. You may want to remove the `default-deny` policy, in case you want to get to your registry or other pods.

```
student@cp:~/app2$ kubectl delete networkpolicies deny-default
```

```
networkpolicy.networking.k8s.io "deny-default" deleted
```