Controlling Network Access with NetworkPolicies - Part 1

Relevant Documentation

- NetworkPolicies
- Cluster Networking

Exam Tips

- If a Pod is not selected by any NetworkPolicies, the Pod is non-isolated, and all traffic is allowed.
- If a Pod is selected by any NetworkPolicy, traffic will be blocked unless it is allowed by at least 1 NetworkPolicy that selects the Pod.
- If you combine a `namespaceSelector` and a `podSelector` within the same rule, the traffic must meet both the Pod- and Namespace-related conditions in order to be allowed.

Lesson Reference

Log in to the **control plane node**.

Let's begin by creating a setup with a Pod that communicates with another Pod via the network.

Create 2 Namespaces.

```
kubectl create namespace np-test-a

kubectl create namespace np-test-b
```

Attach labels to these Namespaces.

```
kubectl label namespace np-test-a team=ateam

kubectl label namespace np-test-b team=bteam
```

Create a server Pod.

```
vi server-pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: server-pod
  namespace: np-test-a
  labels:
    app: np-test-server
spec:
  containers:
  - name: nginx
    image: nginx:stable
    ports:
    - containerPort: 80
```

```
kubectl apply -f server-pod.yml
```

Get the cluster IP address of the server Pod. You may need to wait a few moments for the Pod to be `Running` for the IP address to appear.

```
kubectl get pods server-pod -n np-test-a -o wide
```

Create a client Pod.

```
vi client-pod.yml
```

In the `command`, provide the cluster IP address of the server Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: client-pod
  namespace: np-test-b
  labels:
    app: np-test-client
spec:
  containers:
  - name: busybox
    image: radial/busyboxplus:curl
    command: ['sh', '-c', 'while true; do curl -m 2 <server Pod IP address>; sleep 5; done']
```

```
kubectl apply -f client-pod.yml
```

Check the client Pod's log. You should see that it is able to successfully reach the server Pod.

```
kubectl logs client-pod -n np-test-b
```

Create a default deny ingress NetworkPolicy. This will block incoming traffic for all Pods in the `np-test-a` Namespace by default.

```
vi np-test-a-default-deny-ingress.yml
```

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: np-test-a-default-deny-ingress
  namespace: np-test-a
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

```
kubectl apply -f np-test-a-default-deny-ingress.yml
```

Check the client Pod's log again. The traffic should now be blocked, resulting in error messages in the log.

```
kubectl logs client-pod -n np-test-b
```