🧪 Lab 5.3 – Network Policies

🎯 Objectives

- Create and test **NetworkPolicies** to restrict pod communication
- Use labels and namespaces to simulate segmentation

🔧 Prerequisites

- A running **Kind cluster** with a **CNI that supports Network Policies** (e.g., Calico or Cilium)

Create Pods with Labels

```
kubectl run frontend --image=nginx --labels=app=frontend --e
xpose --port=80
kubectl run backend --image=nginx --labels=app=backend --exp
ose --port=80
```

Test Initial Connectivity

```
kubectl run tester --image=busybox:1.28 -it --rm -- wget -O-
backend
```

✅ Should succeed before applying any policies.

🚫 Step 2 – Apply Default Deny Policy

Edixos Labs
hello@edixos.com
+33 6 47 43 99 30

```yaml
# default-deny.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
    - Ingress
```

```
kubectl apply -f module-5/manifests/default-deny.yaml
```

Re-test from `tester` pod

```
kubectl run tester --image=busybox:1.28 -it --rm -- wget -O-
backend
```

❌ Should now fail

✅ Step 3 – Allow Specific Ingress (From Frontend Only)

Edixos Labs
hello@edixos.com
+33 6 47 43 99 30

```yaml
# allow-frontend.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-frontend
spec:
  podSelector:
    matchLabels:
      app: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app: frontend
```

```
kubectl apply -f module-5/manifests/allow-frontend.yaml
```

🔍 Test with frontend pod

```
kubectl exec -it frontend -- curl backend
```

✅ Should succeed

Test again with tester

```
kubectl run tester --image=busybox:1.28 -it --rm -- wget -O-
backend
```

❌ Should still fail

Edixos Labs
hello@edixos.com
+33 6 47 43 99 30

🧽 Cleanup

```
kubectl delete ns lab5-netpol
```

✅ End of Lab 5.3 – You've implemented and tested Network Policies

Edixos Labs
hello@edixos.com
+33 6 47 43 99 30