🔪 CNI Lab – Kubernetes Networking with Cilium

This lab guides you through installing a Kubernetes cluster **without a default CNI**, deploying **Cilium**, and experimenting with pod communication and **Network Policies**.

🌟 Objectives

- Create a Kind cluster **without a default CNI**
- Preload Cilium image and install it using Helm
- Validate pod communication across namespaces
- Apply and test **Network Policies**

🚀 Step 1: Create Kind Cluster Without CNI

Create `kind-config.yaml`:

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

```yaml
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
name: cni-lab
nodes:
  - role: control-plane
    image: kindest/node:v1.32.3
  - role: worker
    image: kindest/node:v1.32.3
  - role: worker
    image: kindest/node:v1.32.3
  - role: worker
    image: kindest/node:v1.32.3
networking:
  disableDefaultCNI: true
```

Create the cluster:

```
kind create cluster --config kind-config.yaml
```

> *Pods will stay **Pending** until a CNI is installed.*

🚨 Step 2: Show Cluster is Waiting for CNI

```
kubectl get pods -A
kubectl describe pod <pod-name> -n <namespace>
kubectl describe node <node-name>
```

> *Look for `NetworkUnavailable=True` and scheduling issues*

🌐 Step 3: Preload Cilium Image

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

```
docker pull quay.io/cilium/cilium:v1.17.5
kind load docker-image quay.io/cilium/cilium:v1.17.5 --name
 cni-lab
```

🔧 Step 4: Install Cilium Using Helm

Setup Helm repo

```
helm repo add cilium https://helm.cilium.io/
helm repo update
```

Install Cilium with Helm

```
helm install cilium cilium/cilium \
  --version 1.17.5 \
  --namespace kube-system \
  --set image.pullPolicy=IfNotPresent \
  --set ipam.mode=kubernetes
```

🔍 Optional: Validate cgroup namespaces (advanced) (Only for linux users)

```
docker exec cni-lab-control-plane ls -al /proc/self/ns/cgrou
p
docker exec cni-lab-worker ls -al /proc/self/ns/cgroup
ls -al /proc/self/ns/cgroup
```

> *Ensure the cgroup values are different between host and containers.*

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

🎓 Step 5: Install and Use Cilium CLI

macOS (brew)

```
brew install cilium-cli
```

Linux (manual)

```
CILIUM_CLI_VERSION=$(curl -s https://raw.githubusercontent.com/cilium/cilium-cli/main/stable.txt)
CLI_ARCH=amd64
[ "$(uname -m)" = "aarch64" ] && CLI_ARCH=arm64
curl -LO https://github.com/cilium/cilium-cli/releases/download/${CILIUM_CLI_VERSION}/cilium-linux-${CLI_ARCH}.tar.gz
sudo tar -xvzf cilium-linux-${CLI_ARCH}.tar.gz -C /usr/local/bin
```

✅ Validate Installation

```
cilium status --wait
```

📄 Step 6: Deploy Test Pods Across Namespaces

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

```
kubectl create ns ns-a
kubectl create ns ns-b

kubectl run web-a --image=nginx -n ns-a --restart=Never --po
rt=80
kubectl run web-b --image=nginx -n ns-b --restart=Never --po
rt=80

kubectl get pods -A -o wide
```

Create Headless Service for DNS Resolution

```yaml
# web-b-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: web-b
  namespace: ns-b
spec:
  clusterIP: None
  selector:
    run: web-b
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Apply the service and label:

```
kubectl label pod web-b run=web-b -n ns-b
kubectl apply -f web-b-svc.yaml
```

Test Communication

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

```
kubectl exec -n ns-a web-a -- curl -s web-b.ns-b.svc.cluste
r.local
```

🔒 Step 7: Apply Network Policy

Create `deny-ns-b.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all
  namespace: ns-b
spec:
  podSelector: {}
  policyTypes:
    - Ingress
```

Apply it:

```
kubectl apply -f deny-ns-b.yaml
```

Re-test communication:

```
kubectl exec -n ns-a web-a -- curl -s web-b.ns-b.svc.cluste
r.local
```

> ❌ *Curl should now fail due to denied ingress.*

🧪 Optional: Test Connectivity with Cilium CLI

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

```
cilium connectivity test
```

> *Run extended tests to validate end-to-end networking.*

🧹 Step 8: Clean Up

We will keep the cluster for further labs, but you can delete the namespaces to clean up resources:

```
kubectl delete ns ns-a ns-b
```

✅ Checklist

- ☑ Created a Kind cluster with CNI disabled
- ☑ Installed Cilium CNI using Helm
- ☑ Validated installation using `cilium status`
- ☑ Deployed nginx pods across namespaces
- ☑ Created Service for DNS-based communication
- ☑ Applied a Network Policy to restrict traffic
- ☑ (Optional) Ran `cilium connectivity test`

💬 Next Steps

Explore how CRI (container runtimes) and CSI (storage provisioning) extend Kubernetes capabilities. These are covered in the next labs.

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