🧪 CSI Lab – Kubernetes Storage with Static & Dynamic Provisioning (Kind)

This lab introduces the Container Storage Interface (CSI) in Kubernetes through two stages:

1. **Static provisioning** using `hostPath` volumes
2. **Dynamic provisioning** using the official **HostPath CSI driver**

> 🚀 *Reuses the `cni-lab` Kind cluster from previous labs.*

🌟 Objectives

- Understand CSI and its role in Kubernetes
- Perform **static provisioning** with a manually defined PersistentVolume
- Install a lightweight **CSI driver** (HostPath)
- Perform **dynamic provisioning** using a StorageClass

🚀 Part 1: Static Provisioning (hostPath)

Step 1.1: Create a PersistentVolume (PV)

```
mkdir -p /tmp/k8s-csi-lab
```

pv-hostpath.yaml:

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

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: hostpath-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/tmp/k8s-csi-lab"
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""
```

```
kubectl apply -f pv-hostpath.yaml
```

Step 1.2: Create a PersistentVolumeClaim (PVC)

pvc.yaml:

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: hostpath-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
  volumeName: hostpath-pv
  storageClassName: ""
```

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

```
kubectl apply -f pvc.yaml
```

Ensure it binds:

```
kubectl get pvc
```

Step 1.3: Attach PVC to a Pod

pod-with-pvc.yaml:

```
apiVersion: v1
kind: Pod
metadata:
  name: csi-demo-pod
spec:
  containers:
  - name: busybox
    image: busybox
    command: ["sh", "-c", "while true; do sleep 3600; done"]
    volumeMounts:
    - mountPath: "/data"
      name: csi-volume
  volumes:
  - name: csi-volume
    persistentVolumeClaim:
      claimName: hostpath-pvc
```

```
kubectl apply -f pod-with-pvc.yaml
```

Step 1.4: Test Persistence

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

> 💡 ***Note for macOS users**: Since Kind runs Kubernetes nodes as Docker containers, the `hostPath` (`/tmp/k8s-csi-lab`) refers to the container's filesystem, not your local macOS `/tmp`. You won't see the file from your host. To verify:*
>
> ```
> docker exec -it kind-worker bash
> ls /tmp/k8s-csi-lab/
> ```
>
> *Replace `kind-worker` with the actual node your pod is scheduled on (check with `kubectl get pod csi-demo-pod -o wide`).*

```
kubectl exec -it csi-demo-pod -- sh -c "echo 'Hello from CSI' > /data/test.txt && cat /data/test.txt"
kubectl delete pod csi-demo-pod
kubectl apply -f pod-with-pvc.yaml
kubectl exec -it csi-demo-pod -- cat /data/test.txt
```

⚙️ Part 2: Dynamic Provisioning with HostPath CSI Driver

Step 2.1: Install VolumeSnapshot CRDs and Snapshot Controller

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

```
SNAPSHOTTER_BRANCH=release-6.3
kubectl apply -f "https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNAPSHOTTER_BRANCH}/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml"
kubectl apply -f "https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNAPSHOTTER_BRANCH}/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml"
kubectl apply -f "https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNAPSHOTTER_BRANCH}/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml"

SNAPSHOTTER_VERSION=v6.3.3
kubectl apply -f "https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNAPSHOTTER_VERSION}/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml"
kubectl apply -f "https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNAPSHOTTER_VERSION}/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml"
```

Step 2.2: Deploy the HostPath CSI Driver (latest Kubernetes support)

```
cd ~
git clone https://github.com/kubernetes-csi/csi-driver-host-path.git
cd csi-driver-host-path
deploy/kubernetes-latest/deploy.sh
```

Step 2.3: Deploy Example StorageClass, PVC, and App Pod

```
kubectl apply -f examples/csi-storageclass.yaml
kubectl apply -f examples/csi-pvc.yaml
kubectl apply -f examples/csi-app.yaml
```

Validate:

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

```
kubectl get pvc
kubectl get pv
kubectl describe pod my-csi-app
```

Write to volume:

```
```bash
kubectl exec -it my-csi-app -- sh -c "echo 'Dynamic CSI tes
t' > /data/hello-world"
```

```
Check from CSI plugin container:

```bash
kubectl exec -it $(kubectl get pods --selector app.kubernete
s.io/name=csi-hostpathplugin -o jsonpath='{.items[0].metadat
a.name}') -c hostpath -- find / -name hello-world
```

🧪 Final Challenge – Snapshot and Restore

▶ ▶ **Click to expand challenge details**

You have successfully set up a dynamic provisioning environment using the HostPath CSI driver. Now, let's take it a step further with a real-world scenario: creating a snapshot of your application's data and restoring it.

🎯 **Challenge**: Without step-by-step instructions, your task is to:

1. Create a **VolumeSnapshot** from the PVC used by your application.
2. Create a **new PersistentVolumeClaim** from that snapshot.
3. Mount the restored PVC into a **second pod**.

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

4. Validate that the file you created (`hello-world`) still exists.

This challenge will test your ability to:

- Work with snapshot APIs
- Understand StorageClasses and restore workflows
- Operate independently using Kubernetes documentation

Use `kubectl explain` and official snapshot documentation if needed.

🧹 Clean Up

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

```
cd ~/csi-driver-host-path
kubectl delete -f examples
deploy/kubernetes-latest/destroy.sh

kubectl delete -f pod-with-pvc.yaml
kubectl delete -f pvc.yaml
kubectl delete -f pv-hostpath.yaml
rm -rf /tmp/k8s-csi-lab

SNAPSHOTTER_BRANCH=release-6.3
kubectl delete -f "https://raw.githubusercontent.com/kuberne
tes-csi/external-snapshotter/${SNAPSHOTTER_BRANCH}/client/co
nfig/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml"
kubectl delete -f "https://raw.githubusercontent.com/kuberne
tes-csi/external-snapshotter/${SNAPSHOTTER_BRANCH}/client/co
nfig/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l"
kubectl delete -f "https://raw.githubusercontent.com/kuberne
tes-csi/external-snapshotter/${SNAPSHOTTER_BRANCH}/client/co
nfig/crd/snapshot.storage.k8s.io_volumesnapshots.yaml"

SNAPSHOTTER_VERSION=v6.3.3
kubectl delete -f "https://raw.githubusercontent.com/kuberne
tes-csi/external-snapshotter/${SNAPSHOTTER_VERSION}/deploy/k
ubernetes/snapshot-controller/rbac-snapshot-controller.yaml"
kubectl delete -f "https://raw.githubusercontent.com/kuberne
tes-csi/external-snapshotter/${SNAPSHOTTER_VERSION}/deploy/k
ubernetes/snapshot-controller/setup-snapshot-controller.yam
l"
```

✅ Checklist

- ☐ Performed static provisioning with hostPath
- ☐ Installed HostPath CSI driver using official method
- ☐ Deployed VolumeSnapshot CRDs and Controller
- ☐ Created a StorageClass and PVC for dynamic provisioning

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

- ☐ Mounted PVC to a pod and validated persistence
- ☐ Reviewed CSI architecture and validated volume in plugin container
- ☐ Cleaned up all resources

💬 What's Next?

You're now equipped to understand both static and dynamic CSI usage in Kubernetes. Future labs may explore real-world drivers like AWS EBS, Longhorn, or Ceph.

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