# Kubernetes `hostPath` Lab

## Prerequisites

- Kubernetes cluster (e.g. Minikube, Kind)
- `kubectl` installed and configured
- Access to the node (for file verification)

---

## Step 1: Prepare the host directory

Login to your Kubernetes node (if using Minikube, run: `minikube ssh`) and create a directory:

```
sudo mkdir -p /data/hostpath-test
sudo chmod 777 /data/hostpath-test
```

---

## Step 2: Create a Pod with a `hostPath` volume

Create a file named `hostpath-pod.yaml`:

```
apiVersion: v1
kind: Pod
metadata:
  name: hostpath-demo
spec:
  containers:
  - name: busybox
    image: busybox
    command: [ "sh", "-c", "sleep 3600" ]
    volumeMounts:
    - mountPath: /data
      name: host-volume
  volumes:
  - name: host-volume
    hostPath:
      path: /data/hostpath-test
      type: DirectoryOrCreate
```

Apply the Pod:

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

---

## Step 3: Interact with the Pod

Enter the pod and write a file:

```
kubectl exec -it hostpath-demo -- sh
echo "This is written from the pod" > /data/hello.txt
exit
```

---

## Step 4: Verify on the Host

On the host (e.g. `minikube ssh`):

```
cat /data/hostpath-test/hello.txt
```

You should see:

```
This is written from the pod
```

---

## Step 5 (Optional): Test persistence

Delete and recreate the Pod:

```
kubectl delete pod hostpath-demo
kubectl apply -f hostpath-pod.yaml
```

Then check again in the pod:

```
kubectl exec -it hostpath-demo -- cat /data/hello.txt
```

The file should still be there.

---

## What have You Learned

- `hostPath` allows pods to access host node files/directories.
- Changes from inside the pod are reflected on the host and vice versa.
- This is useful for debugging, logs, or interacting with host-mounted devices — but **not recommended for production** due to tight coupling with host nodes.

August 22, 2025