### **Lab Exercise 12- Volume in Kubernetes Cluster**

Volumes in Kubernetes provide a way to store and share data among containers in a Pod. This lab exercise will guide you through the creation and use of different types of volumes in Kubernetes.

**Step 1: Set Up Kubernetes Cluster**

Ensure you have access to a Kubernetes cluster. You can use a local setup with Minikube, kind, or use a cloud-based Kubernetes service.

**Step 2: Create a Pod with an emptyDir Volume**

Create a file named emptydir-pod.yaml with the following content:

apiVersion: v1

kind: Pod

metadata:

  name: test-vol1

spec:

  containers:

    - image: nginx

      name: test-container

      volumeMounts:

        - mountPath: /data

          name: first-volume

  volumes:

    - name: first-volume

      emptyDir: {}

**apiVersion**: v1

**kind**: Pod

**metadata**:

**name**: redis

**spec**:

**containers**:

- **name**: redis

**image**: redis

**volumeMounts**:

- **name**: redis-storage

**mountPath**: /data/redis

**volumes**:

- **name**: redis-storage

**emptyDir**: {}

In this manifest:

The emptyDir volume is a temporary storage that is created when the Pod is assigned to a node and exists as long as the Pod is running.

Apply the manifest to create the Pod:

kubectl apply -f emptydir-pod.yaml

**Step 3: Verify the Pod and Volume**

Check the status of the Pod:

kubectl get pods

Access the Pod's shell:

kubectl exec -it test-vol1 -- bash

Inside the Pod, create a file in the /data directory:

touch 1 2 3 4

You should see the text "Hello, Kubernetes!".

Exit the Pod shell:

exit

**Step 4: Create a Pod with a hostPath Volume**

Create a file named **hostpath-pod.yaml** with the following content:

apiVersion: v1

kind: Pod

metadata:

name: test-vol1

spec:

containers:

- image: coolgourav147/nginx-custom

name: test-container

volumeMounts:

- mountPath: /data

name: first-volume

volumes:

- name: first-volume

hostPath:

path: /tmp/data

In this manifest:

The hostPath volume mounts a directory from the host into the Pod at /mnt/data.

The busybox container writes the current date to a log file every 5 seconds.

**Apply the manifest to create the Pod:**

kubectl apply -f hostpath-pod.yaml

Step 3: Verify the Pod and Volume

Check the status of the Pod:

kubectl get pods

Access the Pod's logs to see the data being written:

kubectl exec -it test-vol2 – bash

You should see the dates being appended to the log file.

**Check the host directory to verify the volume:**

If you have access to the node running the Pod, you can verify the data in /tmp/data/log.txt.