Lesson 4 Demo 15: Configure a Configmap as a Volume in the Pod

This section will guide you to:

* Configure a configmap as a volume in a pod

This lab has one sub-section, namely:

1. Configuring a configmap as a volume in a pod

**Note:** If you don’t have an existing Kubernetes cluster, refer to the Demo 1.1 of Lesson 1.

**Step 1:** Configuring a configmap as a volume in a pod

* Start the kubernetes cluster in the lab
* As explained in the demo Create Configmaps from Files, when you create a Configmap using --from-file, the filename becomes a key stored in the data section of the Configmap. The file contents become the key’s value
* Run **kubectl delete configmap --all** to delete all existing configmaps
* The examples refer to a Configmap named special-config as shown below:

*cat > configmap-multikeys.yaml*

*apiVersion: v1*

*kind: ConfigMap*

*metadata:*

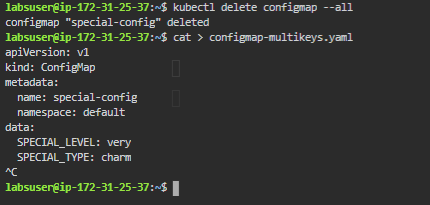
*name: special-config*

*namespace: default*

*data:*

*SPECIAL\_LEVEL: very*

*SPECIAL\_TYPE: charm*



* Next, populate a volume with data stored in a Configmap

*kubectl create -f configmap-multikeys.yaml*



* Add the Configmap name under the volumes section of the pod specification. This adds the Configmap data to the directory specified as volumeMounts.mountPath (in this case, /etc/config). The command section references the special.level item stored in the Configmap and creates the pod using kubectl create command for this file as shown below:

*cat > pod-configmap-volume.yaml*

*apiVersion: v1*

*kind: Pod*

*metadata:*

*name: dapi-test-pod*

*spec:*

*containers:*

*- name: test-container*

*image: k8s.gcr.io/busybox*

*command: [ "/bin/sh", "-c", "ls /etc/config/" ]*

*volumeMounts:*

*- name: config-volume*

*mountPath: /etc/config*

*volumes:*

*- name: config-volume*

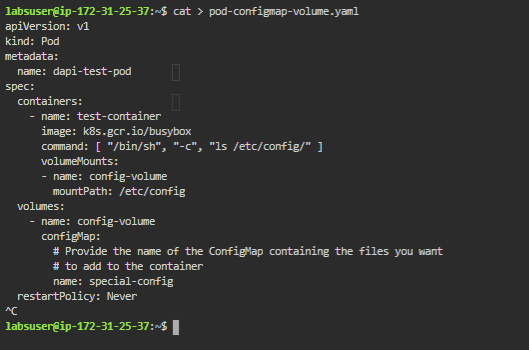
*configMap:*

*# Provide the name of the ConfigMap containing the files you want*

*# to add to the container*

*name: special-config*

*restartPolicy: Never*



* Next, create a pod from this yaml file using:  
    
  *kubectl create -f pod-configmap-volume.yaml*



* When a Configmap is already being consumed in a volume that is being updated, projected keys are eventually updated as well. Kubelet checks whether the mounted Configmap is fresh on every periodic sync. However, it uses its local ttl-based cache for getting the current value of the Configmap. As a result, the total delay from the moment when the Configmap is updated to the moment when new keys are projected to the pod can be as long as kubelet sync period (1 minute by default) + ttl of Configmaps cache (1 minute by default) in kubelet
* Run **kubectl delete configmap --all** to delete all existing configmaps

