Lab Exercise 10- Creating ConfigMaps in Kubernetes

In this lab exercise, you will learn how to create ConfigMaps in Kubernetes using different methods: from literal values, from a file, from an environment file, and from multiple files. ConfigMaps allow you to decouple configuration artifacts from image content to keep containerized applications portable.

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 ConfigMap from Literal Values

Create a ConfigMap named example-configmap with literal values:

```
kubectl create configmap example-configmap --from-literal=key1=value1 --from-literal=key2=value2
```

Verify the ConfigMap:

```
kubectl get configmap example-configmap -o yaml
```

You should see the example-configmap with the keys and values specified.

Step 3: Create a ConfigMap from a File

Create a file named **config.txt** with the following content:

```
key1=value1
key2=value2
```

Create a ConfigMap named file-configmap from the file:

```
kubectl create configmap file-configmap --from-file=config.txt
```

Verify the ConfigMap:

```
kubectl get configmap file-configmap -o yaml
```

You should see the file-configmap with the content of config.txt.

Step 4: Create a ConfigMap from an Environment File

Create an environment file named **env-config.env** with the following content:

```
ENV_VAR1=value1
ENV_VAR2=value2
```

Create a ConfigMap named env-configmap from the environment file:

```
kubectl create configmap env-configmap --from-env-file=env-config.env
```

Verify the ConfigMap:

```
kubectl get configmap env-configmap -o yaml
```

You should see the env-configmap with the environment variables specified in the file.

Step 5: Create a ConfigMap from Multiple Files

Create multiple files:

config1.txt with content:

```
key1=value1
```

config2.txt with content:

```
key2=value2
```

Create a ConfigMap named multi-file-configmap from multiple files:

```
kubectl create configmap multi-file-configmap --from-file=config1.txt --from-file=config2.txt
```

Verify the ConfigMap:

```
kubectl get configmap multi-file-configmap -o yaml
```

You should see the multi-file-configmap with the content of both config1.txt and config2.txt.

Step 6: Use ConfigMaps in a Pod

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

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod
spec:
  containers:
  - name: nginx
    image: nginx
    envFrom:
    - configMapRef:
        name: env-configmap
```

In this manifest:

- The Pod reads environment variables from the env-configmap.
- The Pod mounts the file-configmap at /etc/config.

Apply the manifest to create the Pod:

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

Check the status of the Pod:

```
kubectl get pods
```

View the Pod logs to see the environment variables and file content:

```
kubectl exec -it configmap-pod -- printenv
```

You should see the environment variables and the content of the file-configmap.

Step 7: Clean Up

After completing the exercise, clean up the resources created:

```
kubectl delete pod configmap-pod
kubectl delete configmap example-configmap
kubectl delete configmap file-configmap
kubectl delete configmap env-configmap
kubectl delete configmap multi-file-configmap
rm config.txt env-config.env config1.txt config2.txt
```