Creation of Configmaps

Creating and Viewing Config-maps Inside a Kubernetes Cluster –

Step 1 – Creating a directory

To start creating config-maps connect to the master ec2 instance via SSH as the root user.You need to first create a directory for storing config-maps and the relevant files which can be done by the command –

mkdir <directory-name>

e.g mkdir configmaps (Here configmaps is the name of the directory)

After creating the directory browse inside the directory by giving the command –

cd directory-name

Step 2–Creating files to be used for config-maps creation–

In this context to create the config-maps you will be creating 2 plaintext files with some sample data, in actual scenario config-maps can be created with 1 or more files containing the relevant configuration of the required environment and need not be only plaintext files.Command to create the 2 files for this demo –

echo -n 'Welcome to File 1 for Config Map Creation' > test\_file\_1.txt

echo -n 'Welcome to File 2 for Config Map Creation' > test\_file\_2.txt

Above commands creates two files ‘test\_file\_1.txt’ and ‘test\_file\_2.txt’ with the sample data provided.

Step 3– Creating the configmap–

Now you can create the configmap using the 2 files created in the previous step by giving the below command–

kubectl create configmap demo-configmap --from-file=test\_file\_1.txt --from-file=test\_file\_2.txt

Command Syntax –

kubectl create configmap <configmap-name> --from-file=<file-path/file-name>

You can verify that the configmap got created by giving the below command –

kubectl get configmaps

This will display the configmaps created with the name provided during creation.

Step 4–Viewing the configmap –You can output the contents of the configmap by providing the below command –

kubectl get configmaps demo-configmap -o yaml

Command Syntax –

kubectl get configmaps <configmap-name> -o yaml

This will display the contents and details of the configmap mentioned.

Creating a Pod and consuming the configmap created –

Step 1–

Creating Pod Manifest file –

You will need to create a Pod Manifest file (.yaml file) using which a nginx Pod application will be deployed and inside the manifest file you need to configure the Pod to use the configmap created.

The manifest file will have the below specifications –

#nginx-pod-demo-configmap.yaml

vi nginx-pod-demo-configmap.yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod-demo-configmap

spec:

containers:

- name: nginx-container

image: nginx

volumeMounts:

- name: test-vol

mountPath: /etc/config-map-data

readOnly: true

volumes:

- name: test-vol

configMap:

name: demo-configmap

items:

- key: test\_file\_1.txt

path: test\_file\_1.txt

- key: test\_file\_2.txt

path: test\_file\_2.txt

save it

You need to create a YAML file with the above content. In the ConfigMap section highlighted you need to mention the name of the config that was created in the earlier steps. Mount Path is the location inside the Pod where the configmap will be stored.

Here the name of the YAML file to be created is nginx-pod-demo-configmap.yaml

Step 2 – Deploying the Pod –

After creating the manifest file, the Pod can be deployed by giving the below command –

kubectl create -f nginx-pod-demo-configmap.yaml

Command Syntax–

kubectl create -f <pod manifest file name/path>

Step 3 –Validating the configmap inside the Pod –

You can validate whether the Pod got successfully created with the configmap that was created in the earlier steps by giving the below command –

kubectl exec nginx-pod-demo-configmap -- cat /etc/config-map-data/test\_file\_1.txt

Command Syntax –

kubectl exec <podname> -- cat <mountPath>/<filename>

Cleaning up the cluster by removing the configmaps and Pods created–

If you need to delete the configmaps created or the Pods deployed then the below commands can be used to perform the clean-up.

kubectl delete configmaps <config-map-name>

kubectl delete pods <pod-name>