***configmap and secrets***

you can understand config map from this logic that if you required the multiple file or single file in some stages in your pods to run the application we can used this. For example we have multiple env in our deployment so for deveopmet time we need some file fo run app and for testing time we need some file to test app so configmap create the virtual memory and save file from it once we define in ym file we need so, file at that time will come and support the application

Configma and secretes work same but in secretes we save user,pass and certificate

***LABS***

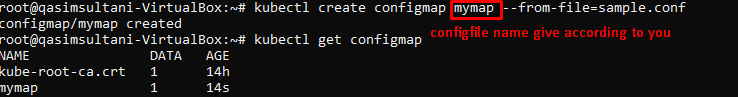
1. ***Create configmap as a volume***

//first create sample config file to support

* nano sample.conf

//create configmap object and attached samplefiles you want

* kubectl create configmap mymap --from-file=sample.conf
* kubectl get configmap



* kubectl describe configmap mymap //to check details in file

***//now the real scenario is there is cluster and the I cluster there is node and pod and container so whenever continer is create this configfile is available there***

* nano deploycomfig.yml

apiVersion: v1

kind: Pod

metadata:

name: myvolconfig

spec:

containers:

- name: c1

image: centos

command: ["/bin/bash", "-c", "while true; do echo HAPPY DAY; sleep 5 ; done"]

volumeMounts:

- name: testconfigmap

mountPath: "/tmp/config" # the config files will be mounted as ReadOnly by default here

volumes:

- name: testconfigmap

configMap:

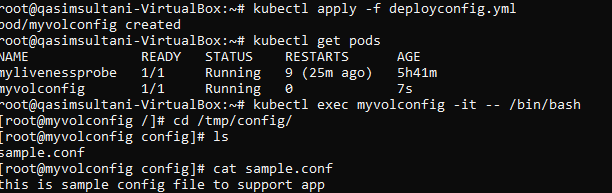
name: mymap # this should match the config map name created in the first step

items:

- key: sample.conf

path: sample.conf

* kubectl apply -f deployconfig.yml
* kubectl get pods
* kubectl exec myvolconfig -it -- /bin/bash
* cd /tmp/config/
* ls



* kubectl delete -f deployconfig.yml

1. ***Create configmap as a ENV-VARIABLE***

nano env.yml

apiVersion: v1

kind: Pod

metadata:

name: myenvconfig

spec:

containers:

- name: c1

image: centos

command: ["/bin/bash", "-c", "while true; do echo Good BY; sleep 5 ; done"]

env:

- name: MYENV # env name in which value of the key is stored

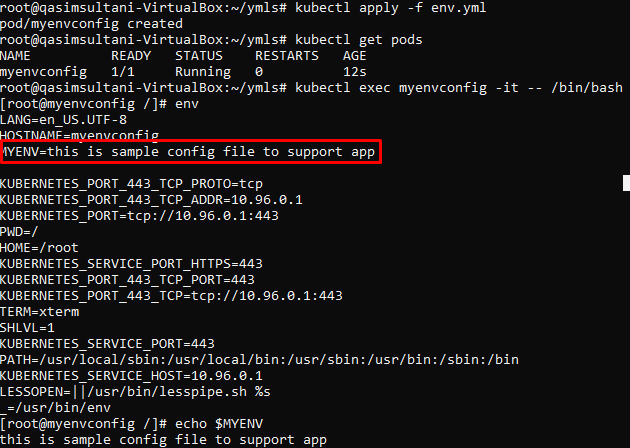
valueFrom:

configMapKeyRef:

name: mymap # name of the config created

key: sample.conf

* kubectl apply -f env.yml
* kubectl get pods
* kubectl exec myenvconfig -it -- /bin/bash
* env // type this because this time we used env variable
* echo $MYENV //to check output



* kubectl delete -f env.yml

***SECRETS LABS***

//Create 2 files & let them created by secrets object

* touch username.txt password
* nano username.txt

Qasim

* nano password

1234

* kubectl create secret generic mysecret --from-file=username

.txt --from-file=password

* kubectl get secrets
* kubectl describe secret mysecret //here you can check the files in secret but didn’t find the content in these files.this is hide

//now if you want this information in pods & container

* nano depsecret.yml

apiVersion: v1

kind: Pod

metadata:

name: myvolsecret

spec:

containers:

- name: c1

image: centos

command: ["/bin/bash", "-c", "while true; do echo IT”S RAINING; sleep 5 ; done"]

volumeMounts:

- name: testsecret

mountPath: "/tmp/mysecrets" # the secret files will be mounted as ReadOnly by default here

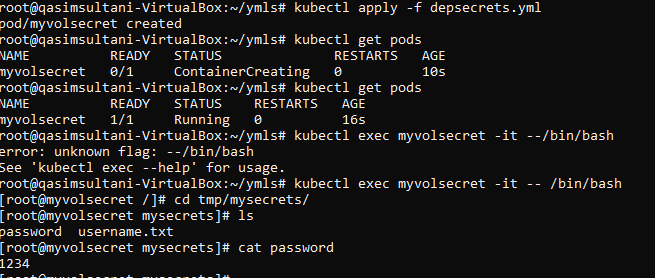
volumes:

- name: testsecret

secret:

secretName: mysecret

* kubectl apply -f depsecrets.yml
* kubectl get pods
* kubectl exec myvolsecret -it -- /bin/bash
* cd tmp/mysecrets/
* ls
* cat password



//you can see password creditentionals & secretes files details in container