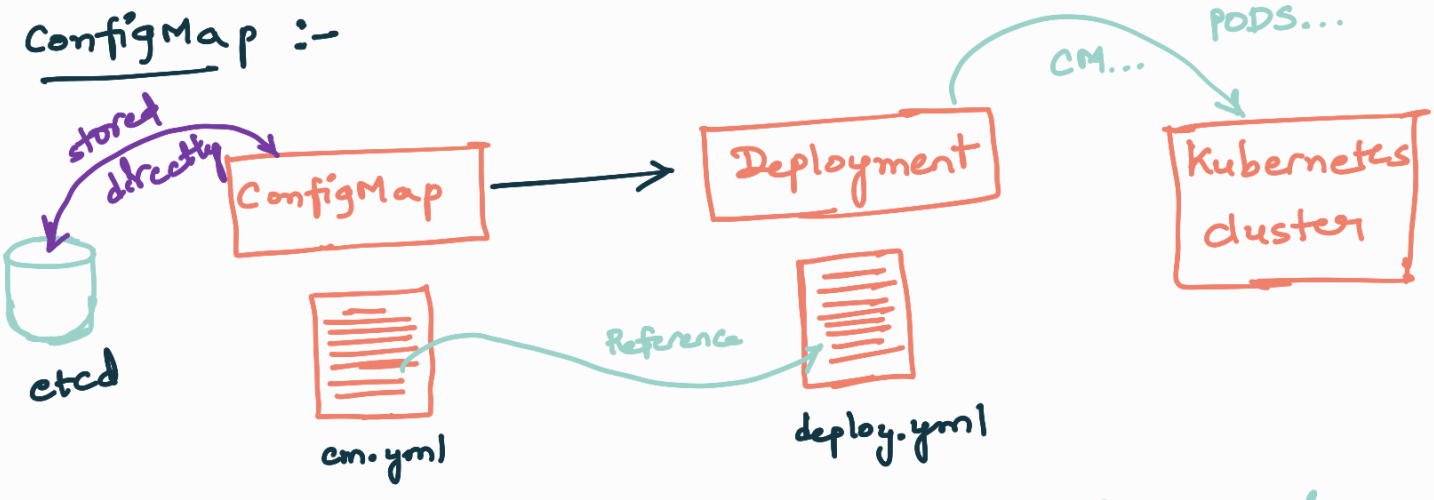
#KubernetesIn30Days challenge :-

#Day17 :-

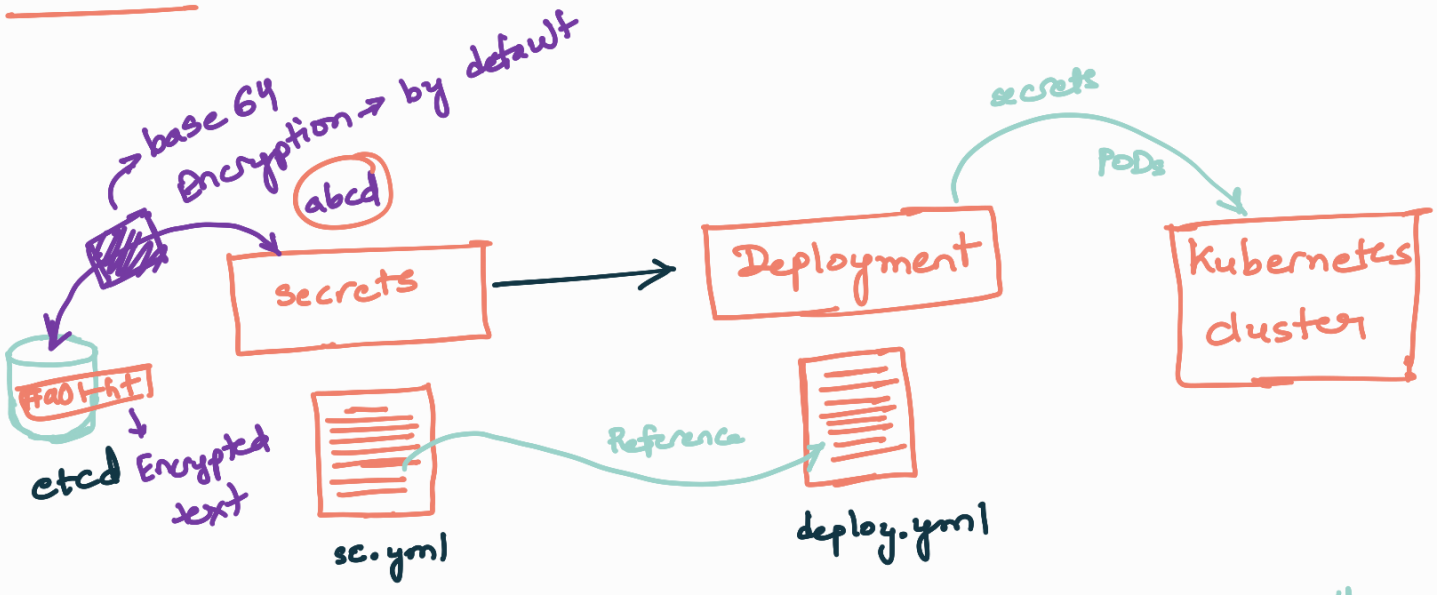
① Hands on ConfigMap and secrets.

ConfigMap :-

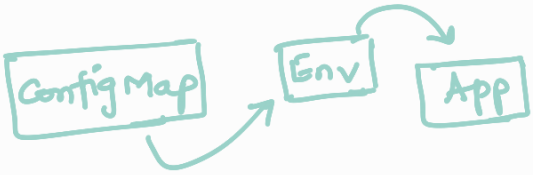


→ usage of ConfigMap is very useful in realtime like DB_PORT, DB_ENDPOINT. (Not DB username and password)

secrets :-

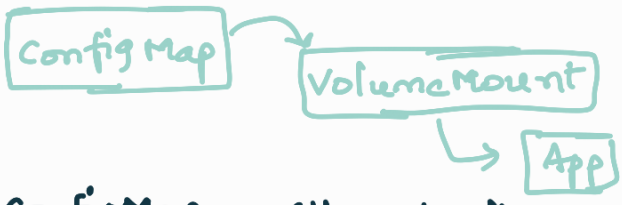


→ The usage of secrets are very specific to the sensitive data such as DB_PWD, DB_USER and may TLS certs and keys



Directly creating Env Variables on container

(But these are fixed we cannot change them)



ConfigMap will actually use concept of volumes to store variables so these can be dynamic. We can change at anytime.

① create a configMap file to be used to create our configMap. Use below file to generate CM.

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: configmap-test
data:
  db-port: "3306"
~
```

→ Basic configMap object file definition

② Use sample deployment file to create the PODs and here we are using concept of Env variables to store variables.

```
PS D:\k8s\CM_Secrets> cat .\deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-app-v1
  labels:
    app: python-app-v1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: python-app-v1
  template:
    metadata:
      labels:
        app: python-app-v1
    spec:
      containers:
      - name: python-app-v1
        image: abhishekf5/python-sample-app-demo:v1
        env:
         - name: DB_PORT
           valueFrom:
            configMapKeyRef:
             name: configmap-test
             key: db-port
        ports:
        - containerPort: 8000
```

→ using concept of Environment Variables

③ check the Pod status. After coming into running state you can go into the container check if Env variable DB_PORT is created or not.

```
PS D:\k8s> kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
python-app-v1-56ff87c94c-hxtcz      1/1     Running   0          5m42s
python-app-v1-56ff87c94c-j6dxg      1/1     Running   0          5m42s
PS D:\k8s> kubectl exec -it python-app-v1-56ff87c94c-hxtcz -- /bin/bash
root@python-app-v1-56ff87c94c-hxtcz:/app#
root@python-app-v1-56ff87c94c-hxtcz:/app# env | grep DB
DB_PORT=3306
root@python-app-v1-56ff87c94c-hxtcz:/app#
```

→ Env variable is created successfully

```
betha@Rajasekhar_PC MINGW64 /d/k8s/CM_Secrets
$ kubectl apply -f cm.yml
configmap/configmap-test created

betha@Rajasekhar_PC MINGW64 /d/k8s/CM_Secrets
$ kubectl describe cm/configmap-test
Name:         configmap-test
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
db-port:
----
3306

BinaryData
====

Events:  <none>
```

key → db-port, value → 3306 (ConfigMap)

⑤ what if the DB_PORT has been changed to something else?

+

this won't work and you need to create new pods/cont

→ To solve this we can have volumes in place.

④ Now, Modify the deployment accordingly below.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-app-v1
  labels:
    app: python-app-v1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: python-app-v1
  template:
    metadata:
      labels:
        app: python-app-v1
    spec:
      containers:
      - name: python-app-v1
        image: abhishekf5/python-sample-app-demo:v1
        volumeMounts:
         - name: db-connection
           mountPath: /opt

        ports:
        - containerPort: 8000
      volumes:
       - name: db-connection
         configMap:
          name: configmap-test
~
deployment.yaml [unix] (10:59 23/12/2023)
"deployment.yaml" [unix] 29L, 573B
```

} → mounting same volume to container

→ creating volume and linking configMap to volume

⑤ Now Modify the DB_PORT to "3380". and apply changes to ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
 name: configmap-test
data:
 db-port: "3380"
```

⑥ check the below ss for more details.

```
PS D:\k8s\CM_Secrets> vim .\cm.yml
PS D:\k8s\CM_Secrets> kubectl apply -f .\cm.yml
configmap/configmap-test configured
PS D:\k8s\CM_Secrets> kubectl describe cm/configmap-test
Name:         configmap-test
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
db-port:
----
3380

BinaryData
====

Events:  <none>
PS D:\k8s\CM_Secrets> kubectl exec -it python-app-v1-56d988b8fd-t2jjh -- /bin/bash
root@python-app-v1-56d988b8fd-t2jjh:/app# cat /opt/db-port | more
3380
```

→ updated CM

→ Value is updated to CM → volume → /opt/ → file

→ updated PORT without pod restart/new pod creation.

secrets are also works same way Except the use cases are different. Just will have a look on secrets as well :-

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-basic-auth
type: kubernetes.io/basic-auth
stringData:
  username: admin # required field for kubernetes.io/basic-auth
  password: tOp-Secret # required field for kubernetes.io/basic-auth
```

→ create a secret file

→ userid/password → sensitive data

② create the secret and check the details of secret

```
$ kubectl get secret secret-basic-auth -o yaml
apiVersion: v1
data:
  password: dDBwLVNlY3JldA==
  username: YWRtaW4=
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Secret","metadata":{"annotations":{},"name":"secret-basic-auth","namespace":"default"},"stringData":{"password":"t0p-Secret",
"username":"admin"},"type":"kubernetes.io/basic-auth"}
  creationTimestamp: "2023-12-23T05:58:34Z"
  name: secret-basic-auth
  namespace: default
  resourceVersion: "580"
  uid: 74c143a7-c6d7-4419-8454-85b916e3ee50
type: kubernetes.io/basic-auth
```

} → base64 encoded text for password / username

Kubernetes do some encryption for secrets but in serious note it is very weak to crack so there are external integration you can do to your secrets than using default Encryption.

③ Even I took the Encrypted text and decoded to orignal username/ passwd

```
betha@Rajasekhar_PC MINGW64 /d/k8s/CM_Secrets
$ echo -n YWRtaW4= | base64 --decode
admin
betha@Rajasekhar_PC MINGW64 /d/k8s/CM_Secrets
$ echo -n dDBwLVNlY3JldA== | base64 --decode
t0p-Secret
```

same creds that we have injected in secrets

This is a very Good Handson on ConfigMaps, Secrets and usage of VolumeMounts.

A Big Thanks to Abhishek Veeramalla for the exceptional content through his YT channel.