Module-9: Container Orchestration using Kubernetes Part - II

Demo Document - 4

edureka!

DEMO-4: ConfigMaps and Secrets

Note: All commands are executed as root.

Using ConfigMaps

1. Create a Kubernetes YAML file of kind ConfigMap and define a key-value pair in the data field

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-cred
  namespace: default
data:
  user: edureka
```

2. Now, use the create or apply command to create the config map

   Syntax: kubectl apply -f conf-map.yaml

```
edureka@kmaster:~/kube-demo$ kubectl apply -f conf-map.yaml
configmap/user-cred created
```

3. To use the data inside the ConfigMap as an environment variable define it inside the deployment/pod spec:

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-dep
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        env:
        - name: CONF-USER
          valueFrom:
            configMapKeyRef:
              name: user-cred
              key: user
```

4. Now create the deployment/pod using the create/apply command:

   Syntax: kubectl apply -f nginx-dep.yaml

```
edureka@kmaster:~/kube-demo$ kubectl apply -f nginx-dep.yaml
deployment.apps/nginx-dep created
```

5. List the pods and check into the deployed pod using exec command:

   Syntax: kubectl get pods
   kubectl exec -it podName bash

```
edureka@kmaster:~/kube-demo$ kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
nginx-dep-6bf74cbfcc-jxtbt    1/1       Running   0          36s
edureka@kmaster:~/kube-demo$ kubectl exec -it nginx-dep-6bf74cbfcc-jxtbt bash
```

6. Now to verify, use the env command to list down all the environment variables inside the container and check if the one defined in ConfigMap is there or not

```
root@nginx-dep-6bf74cbfcc-jxtbt:/# echo $CONF-USER
-USER
root@nginx-dep-6bf74cbfcc-jxtbt:/# env
HOSTNAME=nginx-dep-6bf74cbfcc-jxtbt
NJS_VERSION=1.15.3.0.2.3-1~stretch
NGINX_VERSION=1.15.3-1~stretch
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_PORT=tcp://10.96.0.1:443
PWD=/
HOME=/root
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP_PORT=443
CONF-USER=edureka
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
TERM=xterm
SHLVL=1
KUBERNETES_SERVICE_PORT=443
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_SERVICE_HOST=10.96.0.1
_=/usr/bin/env
```

Using Secrets inside a Pod

1. Encode the data you want to inject in your pods using base64 format
   Syntax: echo “your data” | base64

```
edureka@kmaster:~/kube-demo$ echo edureka123 | base64
ZWR1cmVrYTEyMwo=
```

2. Copy the encoded data and paste in your secret YAML file

```
apiVersion: v1
kind: Secret
metadata:
  name: ng-secret
type: opaque
data:
  password: ZWR1cmVrYTEyMwo=
```

3. Create the secret by using kubectl create or apply command
   Syntax: kubectl apply -f ng-secret.yaml

```
edureka@kmaster:~/kube-demo$ kubectl apply -f ng-secret.yaml
secret/ng-secret created
```

4. Now, use this secret inside of your deployment or pod spec file

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-dep
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        env:
        - name: UPASS
          valueFrom:
            secretKeyRef:
              name: ng-secret
              key: password
```

5. Create the the pod/deployment

   Syntax: kubectl create -f nginx-dep.yaml

```
edureka@kmaster:~/kube-demo$ kubectl apply -f nginx-dep.yaml
deployment.apps/nginx-dep created
```

```
edureka@kmaster:~/kube-demo$ kubectl get pods
NAME                         READY     STATUS    RESTARTS   AGE
httpd-bz7wp                  1/1       Running   1          3d
httpd-sc4tg                  1/1       Running   1          3d
nginx-dep-7476bf8dbc-9qslc   1/1       Running   0          49s
```

6. To verify go inside the pod and check the Environment variables

   Syntax: kubectl exec -it containerID bash

```
edureka@kmaster:~/kube-demo$ kubectl exec nginx-dep-7476bf8dbc-9qslc -it bash
```

```
root@nginx-dep-7476bf8dbc-9qslc:/# env
HOSTNAME=nginx-dep-7476bf8dbc-9qslc
NJS_VERSION=1.15.3.0.2.3-1~stretch
NGINX_VERSION=1.15.3-1~stretch
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_PORT=tcp://10.96.0.1:443
PWD=/
HOME=/root
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
TERM=xterm
SHLVL=1
UPASS=edureka123

KUBERNETES_SERVICE_PORT=443
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_SERVICE_HOST=10.96.0.1
_=/usr/bin/env
```

Mounting a Secret inside a Pod

1. Create a file that you want to mount as a secret
   Syntax: vi file.txt

```
secret demo
```

2. Create the secret using kubectl create command
   Syntax: kubectl create secret generic secretName –from-file=/file/path/file.txt

```
edureka@kmaster:~/kube-demo$ kubectl create secret generic secret-file --from-file=./file.txt
secret/secret-file created
edureka@kmaster:~/kube-demo$ kubectl get secrets
NAME                  TYPE                                  DATA      AGE
dashboard-token-znw2q   kubernetes.io/service-account-token   3         7d
default-token-948l9     kubernetes.io/service-account-token   3         7d
ng-secret               opaque                                1         1d
secret-file             Opaque                                1         29m
```

3. You can verify the secret content by running
   Syntax: kubectl get secret secretName -o yaml

```
edureka@kmaster:~/kube-demo$ kubectl get secret secret-file -o yaml
apiVersion: v1
data:
  file.txt: c2VjcmV0IGRlbW8K
kind: Secret
metadata:
  creationTimestamp: 2018-09-12T11:58:52Z
  name: secret-file
  namespace: default
  resourceVersion: "132869"
  selfLink: /api/v1/namespaces/default/secrets/secret-file
  uid: 37c4468c-b683-11e8-b32d-0800277e6765
type: Opaque
```

4. Now, use the secret inside your deployment/pod spec file by specifying mount path

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-dep
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      volumes:
      - name: edureka
        secret:
          secretName: secret-file
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        volumeMounts:
        - mountPath: /etc/pass/
          name: edureka
```

5. Create the deployment/pod
   Syntax: kubectl apply -f nginx-dep.yaml

```
edureka@kmaster:~/kube-demo$ kubectl apply -f nginx-dep.yaml
deployment.apps/nginx-dep created
```

6. To verify the secret inside the pod container, execute
   Syntax: kubectl exec -it podName bash

```
edureka@kmaster:~/kube-demo$ kubectl exec -it nginx-dep-7566f469d6-gpl95 bash
```

7. Change your directory to the mount directory and check the file you have passed through secret
   Syntax: cd /path/to/mount/directory
   cat file.txt

```
root@nginx-dep-7566f469d6-gpl95:/# cd /etc/pass
root@nginx-dep-7566f469d6-gpl95:/etc/pass# ls
file.txt
root@nginx-dep-7566f469d6-gpl95:/etc/pass# cat file.txt
secret demo
```