KUBERNETES STORAGE

I. Configmap

1. Create configmap

a. Create a PROD ConfigMap

```
kubectl create configmap appconfigprod --from-
literal=DATABASE_SERVERNAME=sql.example.local --from-
literal=BACKEND_SERVERNAME=be.example.local
```

b. Create a QA ConfigMap

```
kubectl create configmap appconfigqa --from-file=appconfigqa

appconfigqa
DATABASE_SERVERNAME="sqlqa.example.local"
BACKEND_SERVERNAME="beqa.example.local"
```

c. Each creation method yeilded a different structure in the ConfigMap

```
kubectl get configmap appconfigprod -o yaml
kubectl get configmap appconfigqa -o yaml
```

2. Using ConfigMaps in Pod Configurations

a. First as environment variables

```
https://github.com/hungtran84/k8s-
cka/blob/master/d2_workloads/02_configmap_secret/01_configMap/deployment-
configmaps-env-prod.yaml
kubectl apply -f deployment-configmaps-env-prod.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world-configmaps-env-prod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world-configmaps-env-prod
  template:
    metadata:
```

```
    labels:
      app: hello-world-configmaps-env-prod
  spec:
    containers:
    - name: hello-world
      image: gcr.io/google-samples/hello-app:1.0
      envFrom:
        - configMapRef:
            name: appconfigprod
      ports:
      - containerPort: 8080
```

b. Let's see or configured enviroment variables

```
PODNAME=$(kubectl get pods | grep hello-world-configmaps-env-prod | awk
'{print $1}' | head -n 1)
echo $PODNAME

    kubectl exec -it $PODNAME -- printenv | sort
```

c. Second as files

```
https://github.com/hungtran84/k8s-
cka/blob/master/d2_workloads/02_configmap_secret/01_configMap/deployment-
configmaps-files-qa.yaml
kubectl apply -f deployment-configmaps-files-qa.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world-configmaps-files-qa
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world-configmaps-files-qa
  template:
    metadata:
      labels:
        app: hello-world-configmaps-files-qa
    spec:
      volumes:
        - name: appconfig
```

```
      configMap:
        name: appconfigqa
  containers:
  - name: hello-world
    image: gcr.io/google-samples/hello-app:1.0
    ports:
    - containerPort: 8080
    volumeMounts:
      - name: appconfig
        mountPath: "/etc/appconfig"
```

d. Let's see our configmap exposed as a file using the key as the file name.

```
PODNAME=$(kubectl get pods | grep hello-world-configmaps-files-qa | awk '{print $1}' | head -n 1)

echo $PODNAME

kubectl exec -it $PODNAME -- /bin/sh
ls /etc/appconfig
cat /etc/appconfig/appconfigqa
exit
```

e. Our ConfigMap key, was the filename we read in, and the values are inside the file. This is how we can read in whole files at a time and present them to the file system with the same name in one ConfigMap

```
kubectl get configmap appconfigqa -o yaml
```

f. Updating a configmap, change BACKEND_SERVERNAME to beqa1.example.local

```
kubectl edit configmap appconfigqa
kubectl exec -it $PODNAME -- watch cat /etc/appconfig/appconfigqa
```

II. Secrect

1. Create a secrect:

```
# Create a secret from a local file, directory or literal value
kubectl create secret generic app1 --from-literal=USERNAME=app1login --from-literal=PASSWORD='S0methingS@Str0ng!'

# check the secret

kubectl get secrets
```

2. Access secrect in the pod:

a. Access as env:

```
# create deployment file
kubectl apply -f deployment-secrets-env.yaml

# check the env

kubectl exec -it $PODNAME -- /bin/sh
printenv | grep ^app1
exit
```

b. Access as file:

```
# create deployment file
https://github.com/hungtran84/k8s-cka/blob/master/d2_workloads/02_configmap_secret/02_secrets/deployment-secrets-files.yaml
kubectl apply -f deployment-secrets-files.yaml

PODNAME=$(kubectl get pods | grep hello-world-secrets-files | awk '{print $1}' | head -n 1)
echo $PODNAME
```

```
# check the file
kubectl exec -it $PODNAME -- /bin/sh

ls /etc/appconfig
cat /etc/appconfig/USERNAME
cat /etc/appconfig/PASSWORD
exit
```