secret and configmap hands on notları

SECRET

1.secret: filedan çektik
kubectl create secret generic db-user-pass --from-file=./username.txt --from-file=./password.txt

2.secret: filedan çektik ancak dosya adını standart olması için yeniden adlandırdık
kubectl create secret generic db-user-pass-key --from-file=username=./username.txt --from-file=password=./password.txt

3. secret: direkt komut satırında oluşturduk
*kubectl create secret generic dev-db-secret --from-literal=username=devuser --from-literal=password='S!B|*d$zDsb='*

4. secret: secret.yaml dosyasına username ve passwordu şifreli yazmak için komut satırında encode ettik sonra yaml dosyasındaki value değerlerine yazdık .

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
```

5. secret: mysecret-pod.yaml dosyasında username ve password bilgilerini 4. aşamada oluşturduğumuz secret.yaml dosyasından aldık(environment variables kullanımı)

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
  labels:
    name: secret-env-pod
spec:
  containers:
```

```
  - name: secret-env-pod
    image: redis
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: username
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password
  restartPolicy: Never
```

bu yaml file ile oluşturulan containerın içine bağlanıp secret variable'ları çağırırsak decode olarak görürürüz

```
kubectl exec -it secret-env-pod -- bash
root@secret-env-pod:/data# echo $SECRET_USERNAME
admin
root@secret-env-pod:/data# echo $SECRET_PASSWORD
1f2d1e2e67df
```

CONFIG MAP

1.configmap: deployment.yaml ve service yaml oluşturuldu. imajın bize kendi içeriğinden direkt olarak verdiği çıktıyı ***web browser : "hello clarusway"'***ı gördük.

2. configmap: oluşturduğumuz deployment yaml dosyasını farklı bir imaj ve bu imajın çalışırken bize arguments'leri yaml dosyasındaki

```
env:
   - name: GREETING
    value: selam
```

değerden alarak yazdıracak. ***web browser : selam, Clarusway!***

3. configmap : deployment yaml güncellendi. sonrasında komut satırına "kubectl create configmap demo-config --from-literal=greeting=Hola" komutu girildi. Sonrasında deployment.yaml apply edildi ve environmetimda

env:
- name: GREETING
valueFrom:
configMapKeyRef:
name: demo-config
key: greeting

"key: greeting" değerimi komut satırında yazmış olduğum "hola" ile ***web browser: Hola, Clarusway!*** şeklinde almış olduk. (edited) (edited)

4. configmap: dosya oluştduk echo "greeting: Hei" > config komut satırında
kubectl create configmap demo-config --from-file=./config dosyadan içeriği çekerek oluşturmasını söyledik.

5. configmap: deployment güncellendi-imaj değşti. artık environment variabldan değil de konteyner içerisine volume aracılığyla attach ettiğimiiz demo.yaml dosyası içerisinden aldık.

6. configmap YAML: fconfigmap .yaml oluşturuldu. değişkeni artık aşağıdaki gibi alcaz.
data:
config: |
greeting: Buongiorno
web-browser: Buongiorno, Clarusway!

7. configmap: configmap.yaml oluşturuldu. deployment.yaml güncllendi. 1(bir değişkeni almak istiyorsam) configmap.yamldan alacak.
CONFİG.YAML
data:
greeting: Hola
DEPLOYMENT.YAML
env:
- name: GREETING
valueFrom:
configMapKeyRef:
name: demo-config
key: greeting

8.configmap: configmap.yaml ve deployment yaml güncellendi. eğer 1

den fazla değişkeni çekmek için sürekli kod tekrarı yapmak yerine 1 den fazla değişkeni configmap.yaml dosyasında
GREETING: merhaba
KEY1: .....
KEY2: .....