Secrets

Sunday, January 2, 2022 6:07 PM

To store Username and password on Kubernetes Cluster , Kubernetes provides Kubernetes secrets.

One of the simplest examples of kubernetes secrets would be running mysql container images inside the Kubernetes Cluster.
As you now mysql is a database and to access the database we need username, password and since we are running the mysql container inside the kubernetes cluster so we need to store those credentials(Username, Password) somewhere inside the kubernetes cluster and it should be safe enough, so for storing mysql username and password we are going to use **Kubernetes Secrets**.

How does Kubernetes Handles the Secret.

When you work with Kubernetes you always run multiple docker containers but one thing Kubernetes is good at running the container in an ephemeral mode so that each container is independent of other and does not share any resources with each other.

Note: Kubernetes secrets are independent objects and are not bound to any kubernetes pod or docker container.

Since Docker containers are running in the ephemeral mode, we need to make the Kubernetes secrets available to the deployment manifest of your docker container.

As a general practice, we always create Kubernetes secrets independently, and the secrets can only be accessed by the PODS.

Kubernetes **kubectl** command.

1. secret-name - test-secret
2. Username - test-user
3. Password - testP@ssw0rd

Command to create secret :

```
kubectl create secret generic test-secret --from-literal=username=test-user --from-literal=password=testP@ssword
secret/test-secret created
```

Verify the secret :

kubectl get secret test-secret

```
$ kubectl get secret test-secret
NAME          TYPE     DATA   AGE
test-secret   Opaque   2      11s
```

Check the secret using Kubectl command.

kubectl describe secret test-secret

```
$ kubectl describe secret test-secret
Name:         test-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
password:  22 bytes
username:  9 bytes
```

Create base64 encoded Kubernetes secrets.

Encode username and password :

echo -n 'test-user' | base64
4oCYdGVzdC11c2Vy4oCZ

$ echo -n 'testP@ssword' | base64
dGVzdFBAc3N3b3Jk

Create a **test-secret.yaml** using base64 encoded username and password.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
type: kubernetes.io/basic-auth
stringData:
  username: 4oCYdGVzdC11c2Vy4oCZ
  password: dGVzdFBAc3N3b3Jk
```

Apply Secrets :

$ kubectl apply -f test-secrets.yaml
secret/mysql-secret created

Create Kubernetes Secret from files :

kubectl create secret generic test-secret --from-file=username.txt --from-file=password.txt

Use the Kubernetes secrets inside your POD or deployment manifest

Kubernetes secrets inside Kubernetes POD

Test-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
```

```
spec:
  containers:
    - image: busybox
      name: busybox
      command: ["/bin/sh"]
      args: ["-c","sleep 600"]
      env:
        - name: myusername
          valueFrom:
            secretKeyRef:
              name: test-secret
              key: username
```

$ kubectl apply -f test-pod.yaml
pod/busybox created

$ kubectl describe pod busybox

Using the kubernetes secrets inside the deployment manifest.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-test-secret
type: kubernetes.io/basic-auth
stringData:
  password: test1234
```

$ kubectl apply -f test-mysql-secret.yaml
secret/mysql-test-secret created

$ kubectl get secret mysql-test-secret

```
NAME                TYPE                       DATA AGE
 mysql-test-secret kubernetes.io/basic-auth 1 17s
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
```

```
      env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-test-secret
              key: password
      ports:
        - containerPort: 3306
          name: mysql
      volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /home/vagrant/storage
    volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: test-pvc
```

$ kubectl apply -f mysql-deployment.yaml

deployment.apps/mysql created

Ref : https://jhooq.com/