Secrets

Introduction:

A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. We would be using these objects instead of storing the secret information inside a pod or in a container image.

Because Secrets can be created independently of the Pods that use them, there is less risk of the Secret (and its data) being exposed during the workflow of creating, viewing, and editing Pods.

A pod can use these secrets using below method

- Mount as a volume
- Container environment variable

Objectives:

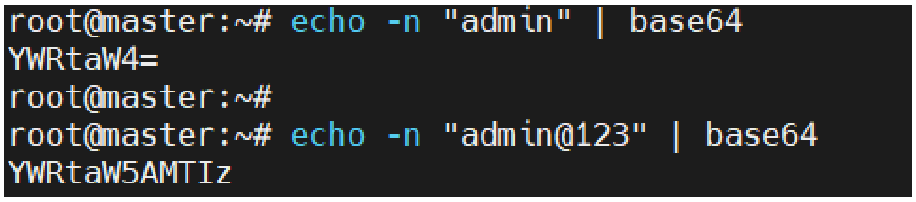
1. Create a secret
2. Use the secret object as volume mount
3. Use the secret object as environment variable

1. Create a secret

First, let’s create a username and a password which will be stored in a secret object. Below we encoding them in base64.

```
echo -n "admin" | base64
YWRtaW4=
```

```
echo -n "admin@123" | base64
YWRtaW5AMTIz
```



```
root@master:~# echo -n "admin" | base64
YWRtaW4=
root@master:~#
root@master:~# echo -n "admin@123" | base64
YWRtaW5AMTIz
```

Above we have encoded **admin** and **admin@123** using base64 and the result is **YWRtaW4=** and **YWRtaW5AMTIz** respectively.

Now, we can use the above information to create a secret.

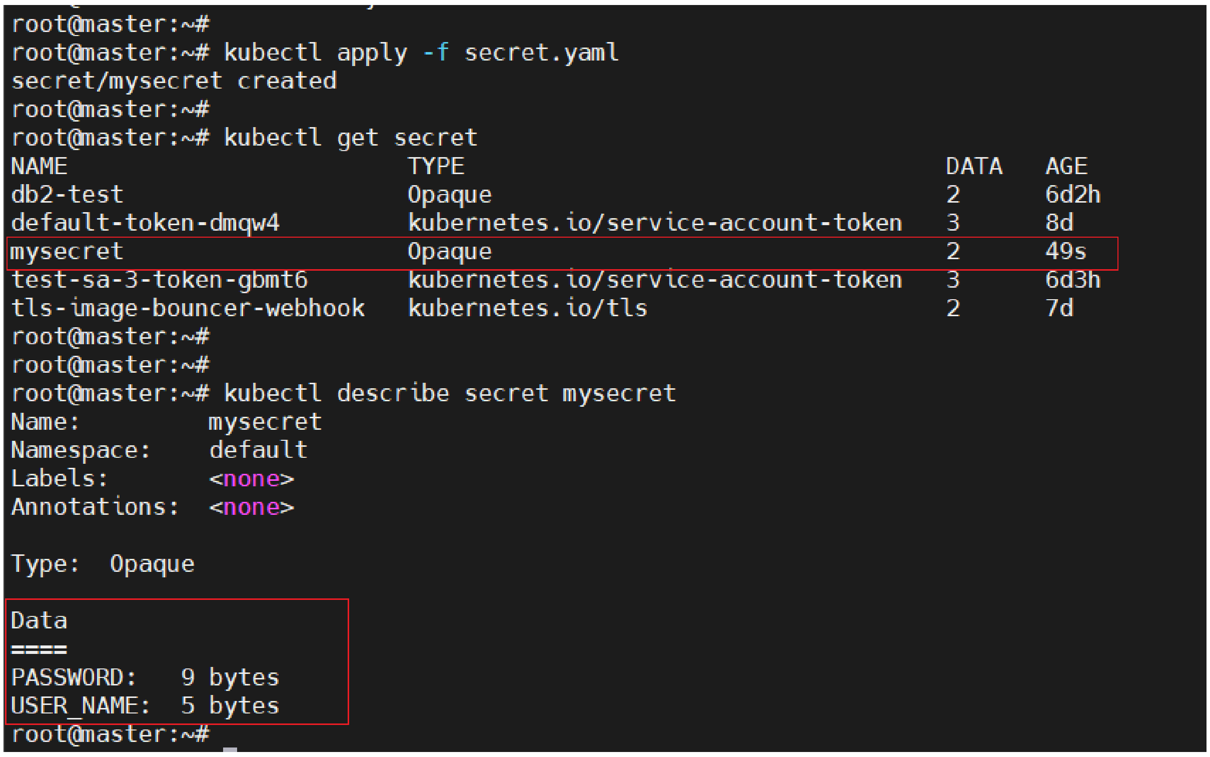
Use the below Yaml file to create a secret.

```
vi secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  USER_NAME: YWRtaW4=
  PASSWORD: YWRtaW5AMTIz
```

Now apply the above definition file.

```
kubectl apply -f secret.yaml
```



```
root@master:~#
root@master:~# kubectl apply -f secret.yaml
secret/mysecret created
root@master:~#
root@master:~# kubectl get secret
NAME                        TYPE                                  DATA   AGE
db2-test                    Opaque                                2      6d2h
default-token-dmqw4         kubernetes.io/service-account-token   3      8d
mysecret                    Opaque                                2      49s
test-sa-3-token-gbmt6       kubernetes.io/service-account-token   3      6d3h
tls-image-bouncer-webhook   kubernetes.io/tls                     2      7d
root@master:~#
root@master:~#
root@master:~# kubectl describe secret mysecret
Name:         mysecret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
PASSWORD:   9 bytes
USER_NAME:  5 bytes
root@master:~#
```

The above output shows that our secret has been created.

2. Use the secret object as volume mount

We have our secret ready and we can mount this secret object as a volume inside our pod. Let's create a pod using below definition file to use the secret as volume.

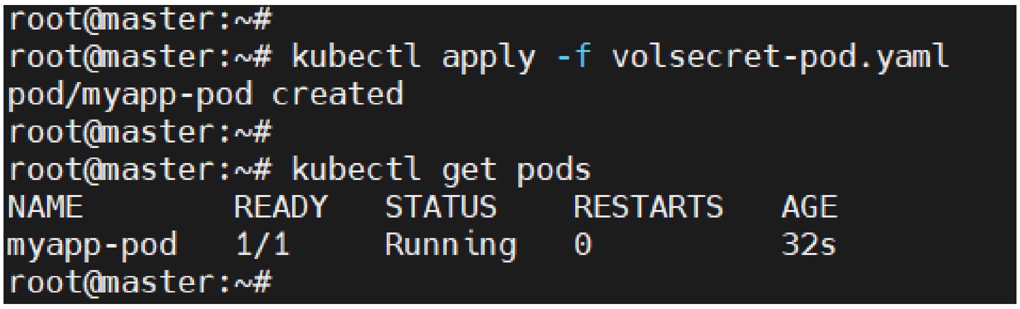
```
vi volsecret-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: httpd-container
    image: httpd
    volumeMounts:
    - name: credentials
      mountPath: /tmp/creds
      readOnly: true
  volumes:
  - name: credentials
    secret:
       secretName: mysecret
```

In above definition file, we are mounting the secret to container's path **/tmp/creds**.

Now apply this definition file.

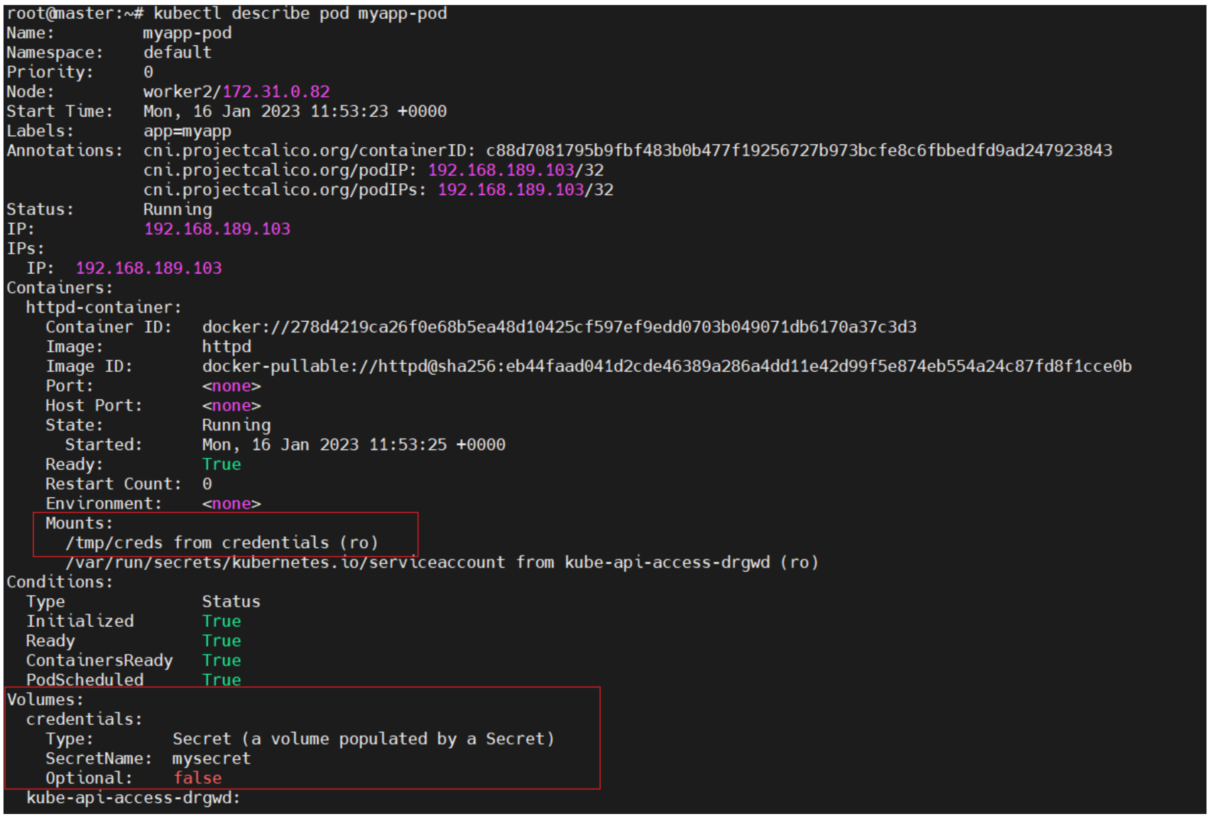
```
kubectl apply -f volsecret-pod.yaml
```



```
root@master:~#
root@master:~# kubectl apply -f volsecret-pod.yaml
pod/myapp-pod created
root@master:~#
root@master:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
myapp-pod   1/1     Running   0          32s
root@master:~#
```

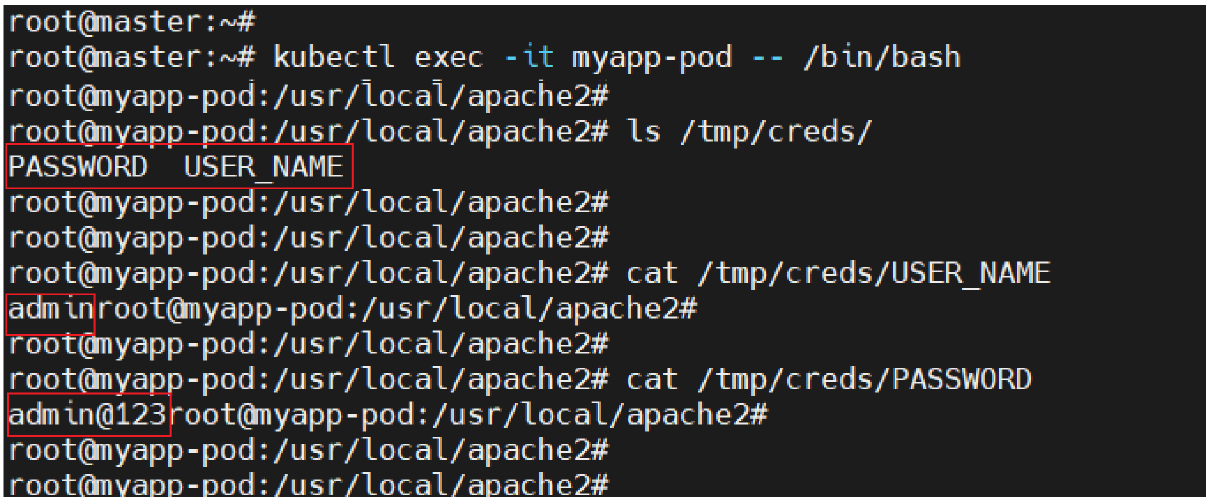
```
kubectl describe pod myapp-pod
```

Above command will describe the pod and we can see the **mysecret** has been mounted as a volume.



```
root@master:~# kubectl describe pod myapp-pod
Name:         myapp-pod
Namespace:    default
Priority:     0
Node:         worker2/172.31.0.82
Start Time:   Mon, 16 Jan 2023 11:53:23 +0000
Labels:       app=myapp
Annotations:  cni.projectcalico.org/containerID: c88d7081795b9fbf483b0b477f19256727b973bcfe8c6fbbedfd9ad247923843
              cni.projectcalico.org/podIP: 192.168.189.103/32
              cni.projectcalico.org/podIPs: 192.168.189.103/32
Status:       Running
IP:           192.168.189.103
IPs:
  IP:  192.168.189.103
Containers:
  httpd-container:
    Container ID:   docker://278d4219ca26f0e68b5ea48d10425cf597ef9edd0703b049071db6170a37c3d3
    Image:          httpd
    Image ID:       docker-pullable://httpd@sha256:eb44faad041d2cde46389a286a4dd11e42d99f5e874eb554a24c87fd8f1cce0b
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 16 Jan 2023 11:53:25 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /tmp/creds from credentials (ro)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-drgwd (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  credentials:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  mysecret
    Optional:    false
  kube-api-access-drgwd:
```

Now let's access the pod and check if the secret has been stored at the location or not.



```
root@master:~#
root@master:~# kubectl exec -it myapp-pod -- /bin/bash
root@myapp-pod:/usr/local/apache2#
root@myapp-pod:/usr/local/apache2# ls /tmp/creds/
PASSWORD  USER_NAME
root@myapp-pod:/usr/local/apache2#
root@myapp-pod:/usr/local/apache2#
root@myapp-pod:/usr/local/apache2# cat /tmp/creds/USER_NAME
adminroot@myapp-pod:/usr/local/apache2#
root@myapp-pod:/usr/local/apache2#
root@myapp-pod:/usr/local/apache2# cat /tmp/creds/PASSWORD
admin@123root@myapp-pod:/usr/local/apache2#
root@myapp-pod:/usr/local/apache2#
root@myapp-pod:/usr/local/apache2#
```

The above result shows us that secret information is stored in directory **/tmp/creds/**

3. **Use the secret object as environment variable:**

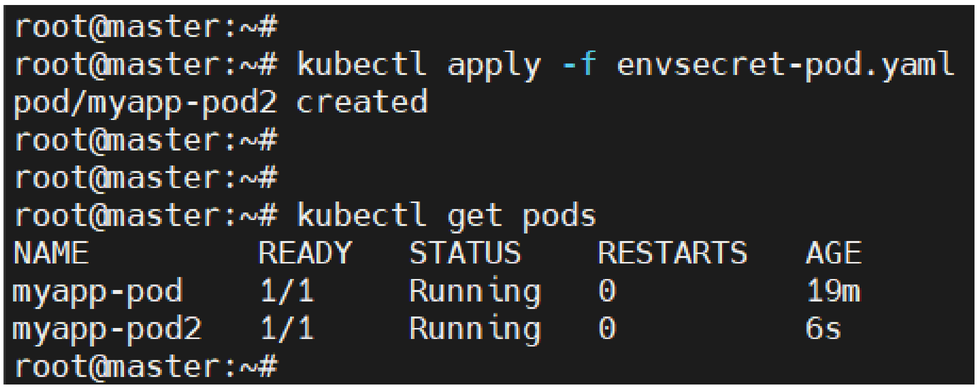
We can store secret object as environment variable as well. Use the below given Yaml file.

```
vi envsecret-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod2
  labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: httpd-container
    image: httpd
    env:
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
          name: mysecret
          key: user_name
    - name: SECRET_PASSWD
      valueFrom:
        secretKeyRef:
          name: mysecret
          key: PASSWORD
```

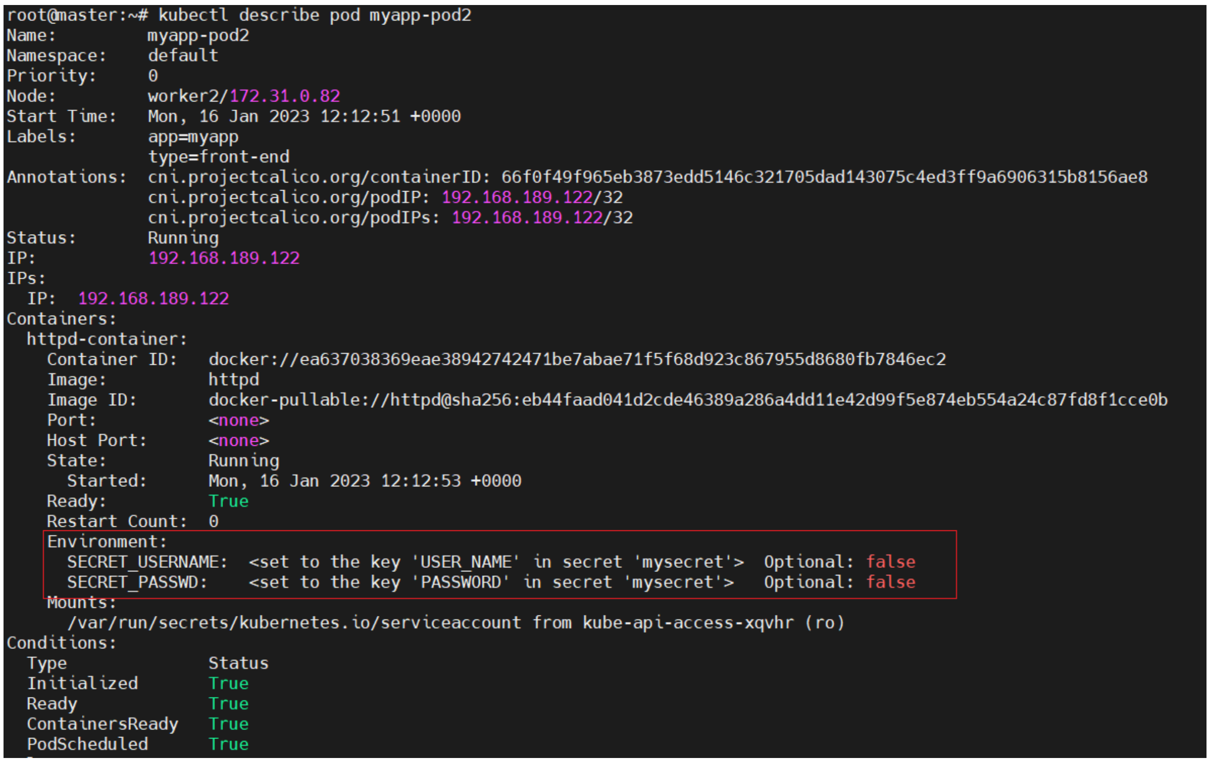
Now apply the definition file using below command.

```
kubectl apply -f envsecret-pod.yaml
```



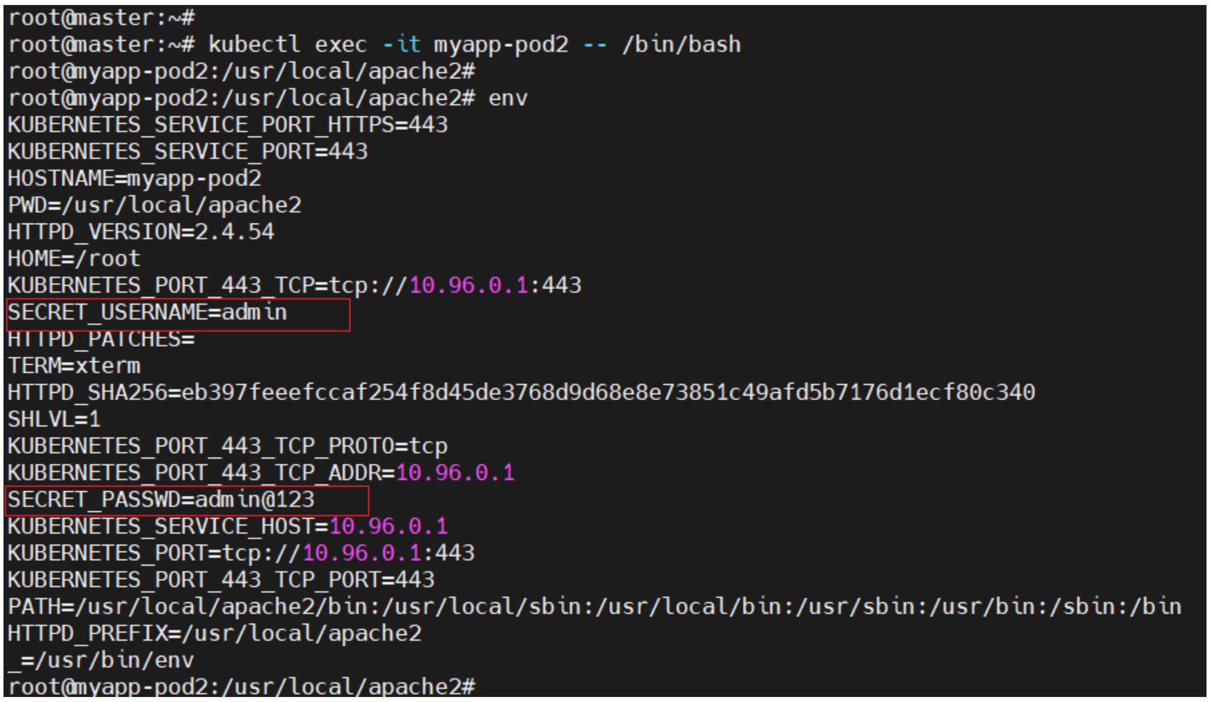
```
root@master:~#
root@master:~# kubectl apply -f envsecret-pod.yaml
pod/myapp-pod2 created
root@master:~#
root@master:~#
root@master:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
myapp-pod     1/1     Running   0          19m
myapp-pod2    1/1     Running   0          6s
root@master:~#
```

Our pod is running now, lets describe it to get more information.



```
root@master:~# kubectl describe pod myapp-pod2
Name:         myapp-pod2
Namespace:    default
Priority:     0
Node:         worker2/172.31.0.82
Start Time:   Mon, 16 Jan 2023 12:12:51 +0000
Labels:       app=myapp
              type=front-end
Annotations:  cni.projectcalico.org/containerID: 66f0f49f965eb3873edd5146c321705dad143075c4ed3ff9a6906315b8156ae8
              cni.projectcalico.org/podIP: 192.168.189.122/32
              cni.projectcalico.org/podIPs: 192.168.189.122/32
Status:       Running
IP:           192.168.189.122
IPs:
  IP:  192.168.189.122
Containers:
  httpd-container:
    Container ID:   docker://ea637038369eae38942742471be7abae71f5f68d923c867955d8680fb7846ec2
    Image:          httpd
    Image ID:       docker-pullable://httpd@sha256:eb44faad041d2cde46389a286a4dd11e42d99f5e874eb554a24c87fd8f1cce0b
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 16 Jan 2023 12:12:53 +0000
    Ready:          True
    Restart Count:  0
    Environment:
      SECRET_USERNAME:  <set to the key 'USER_NAME' in secret 'mysecret'>  Optional: false
      SECRET_PASSWD:    <set to the key 'PASSWORD' in secret 'mysecret'>   Optional: false
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-xqvhr (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
```

Above we can see, the **SECRET_USERNAME** and **SECRET_PASSWD** environment variable has been created. Let's go inside the pod and check for the actual values of these variables.



```
root@master:~#
root@master:~# kubectl exec -it myapp-pod2 -- /bin/bash
root@myapp-pod2:/usr/local/apache2#
root@myapp-pod2:/usr/local/apache2# env
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=myapp-pod2
PWD=/usr/local/apache2
HTTPD_VERSION=2.4.54
HOME=/root
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
SECRET_USERNAME=admin
HTTPD_PATCHES=
TERM=xterm
HTTPD_SHA256=eb397feeefccaf254f8d45de3768d9d68e8e73851c49afd5b7176d1ecf80c340
SHLVL=1
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
SECRET_PASSWD=admin@123
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PORT=443
PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HTTPD_PREFIX=/usr/local/apache2
_=/usr/bin/env
root@myapp-pod2:/usr/local/apache2#
```

Above we can see the same username and password has been stored as environment variables.