```
deepa@ubuntu:~/devops/kind/091125$ kubectl apply -f dep-load.yaml
namespace/my-namespace unchanged
secret/my-secret unchanged
configmap/my-config unchanged
deployment.apps/my-app-deployment created
service/my-app-service unchanged
deepa@ubuntu:~/devops/kind/091125$ kubectl get pods -n my-namespace
NAME                                 READY   STATUS    RESTARTS   AGE
my-app-deployment-589bc7bfd8-mc6t6   1/1     Running   0          61s
my-app-deployment-589bc7bfd8-nwzgd   1/1     Running   0          61s
my-app-deployment-589bc7bfd8-wqt2b   1/1     Running   0          61s
deepa@ubuntu:~/devops/kind/091125$
deepa@ubuntu:~/devops/kind/091125$ kubectl get all -n my-namespace
NAME                                     READY   STATUS    RESTARTS   AGE
pod/my-app-deployment-589bc7bfd8-mc6t6   1/1     Running   0          2m52s
pod/my-app-deployment-589bc7bfd8-nwzgd   1/1     Running   0          2m52s
pod/my-app-deployment-589bc7bfd8-wqt2b   1/1     Running   0          2m52s

NAME                     TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
service/my-app-service   LoadBalancer   10.96.143.5   <pending>     80:32396/TCP   40m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-app-deployment   3/3     3            3           2m52s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/my-app-deployment-589bc7bfd8   3         3         3       2m52s
deepa@ubuntu:~/devops/kind/091125$
```

```yaml
#Namespace
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
---

#Secret
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
  namespace: my-namespace
type: Opaque
data:
  username: bXl1c2Vy
  password: bXlwYXNz

---
#ConfigMap
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
  namespace: my-namespace
data:
  APP_MODE: "production"
  LOG_SCALE: "info"
  WELCOME_MESSAGE: "welcome to my-app"
```

```yaml
---
#Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: nginx:latest
        ports:
        - containerPort: 80
        volumeMounts:
        - name: secret-volume
          mountPath: /etc/secret-data
          readOnly: true
        envFrom:
        - configMapRef:
            name: my-config
        - secretRef:
            name: my-secret
      volumes:
      - name: secret-volume
        secret:
          secretName: my-secret
```

```yaml
---
#service
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
  namespace: my-namespace
  labels:
    app: my-app
spec:
  type: LoadBalancer
  selector:
    app: my-app
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
```

In Deployment YAML, you've done **two different secret injections**:

As **environment variables**
each key in your Secret (like username, password) is automatically injected as an environment variable inside your container.

As a **mounted volume**
same Secret is also **mounted as files** under /etc/secret-data inside the container.
So, inside the pod you'll see:
/etc/secret-data/username
/etc/secret-data/password

Verify inside your running pod
1.Check environment variables:
kubectl exec -it my-app-deployment-589bc7bfd8-mc6t6 -n my-namespace -- env | grep username
kubectl exec -it my-app-deployment-589bc7bfd8-mc6t6 -n my-namespace -- env | grep password

```
deepa@ubuntu:~/devops/kind/091125$ kubectl exec -it my-app-deployment-589bc7bfd8-mc6t6 -n my-namespace -- env | grep username
username=myuser
deepa@ubuntu:~/devops/kind/091125$ kubectl exec -it my-app-deployment-589bc7bfd8-mc6t6 -n my-namespace -- env | grep password
password=mypass
deepa@ubuntu:~/devops/kind/091125$
```

2.Check mounted files:

kubectl exec -it my-app-deployment-589bc7bfd8-mc6t6 -n my-namespace -- ls /etc/secret-data

```
deepa@ubuntu:~/devops/kind/091125$ kubectl exec -it my-app-deployment-589bc7bfd8-mc6t6 -n my-namespace -- ls /etc/secret-data
password  username
deepa@ubuntu:~/devops/kind/091125$
```

To **view and decode your Kubernetes Secret** directly from the command line, without entering the pod.

```
deepa@ubuntu:~/devops/kind/091125$ kubectl get secret my-secret -n my-namespace -o yaml
apiVersion: v1
data:
  password: bXlwYXNz
  username: bXl1c2Vy
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"password":"bXlwYXNz","username":"bXl1c2Vy"},"kind":"Secret","metadata":{"annotations":{},"name":"my-secret","
namespace":"my-namespace"},"type":"Opaque"}
  creationTimestamp: "2025-11-09T07:23:24Z"
  name: my-secret
  namespace: my-namespace
  resourceVersion: "82340"
  uid: 6ab82499-8778-40f8-9092-bc5222f197b9
type: Opaque
deepa@ubuntu:~/devops/kind/091125$
```

Decode using base64 –decode:

```
deepa@ubuntu:~/devops/kind/091125$ kubectl get secret my-secret -n my-namespace -o jsonpath='{.data.username}' | base64 --decode
myuserdeepa@ubuntu:~/devops/kind/091125$ kubectl get secret my-secret -n my-namespace -o jsonpath='{.data.username}' | base64 --decode;echo
myuser
deepa@ubuntu:~/devops/kind/091125$
deepa@ubuntu:~/devops/kind/091125$
```

```
deepa@ubuntu:~/devops/kind/091125$ kubectl get secret my-secret -n my-namespace -o jsonpath='{.data.password}' | base64 --decode
mypassdeepa@ubuntu:~/devops/kind/091125$ kubectl get secret my-secret -n my-namespace -o jsonpath='{.data.password}' | base64 --decode;echo
mypass
deepa@ubuntu:~/devops/kind/091125$
```

How can you update a Kubernetes Secret without deleting and recreating it?

kubectl create secret generic my-secret --from-literal=username=admin -n my-namespace --dry-run=client -o yaml | kubectl apply -f -

```
deepa@ubuntu:~/devops/kind/091125$ kubectl create secret generic my-secret --from-literal=username=admin -n my-namespace --dry-run=client -o ya
ml | kubectl apply -f -
secret/my-secret configured
deepa@ubuntu:~/devops/kind/091125$
```

kubectl create secret generic my-secret --from-literal=password=mypass -n my-namespace --dry-run=client -o yaml | kubectl apply -f –

```
deepa@ubuntu:~/devops/kind/091125$ kubectl create secret generic my-secret --from-literal=password=mypass -n my-namespace --dry-run=client -o y
aml | kubectl apply -f -
secret/my-secret configured
deepa@ubuntu:~/devops/kind/091125$
```