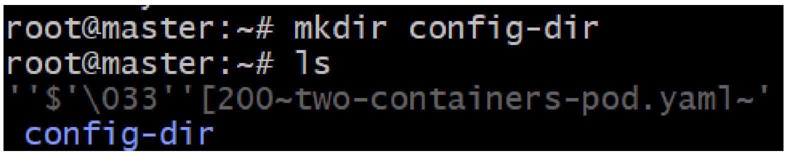
K8s 06

1) Create a ConfigMap from a directory containing multiple files and inject the variables into a pod as environment variables.

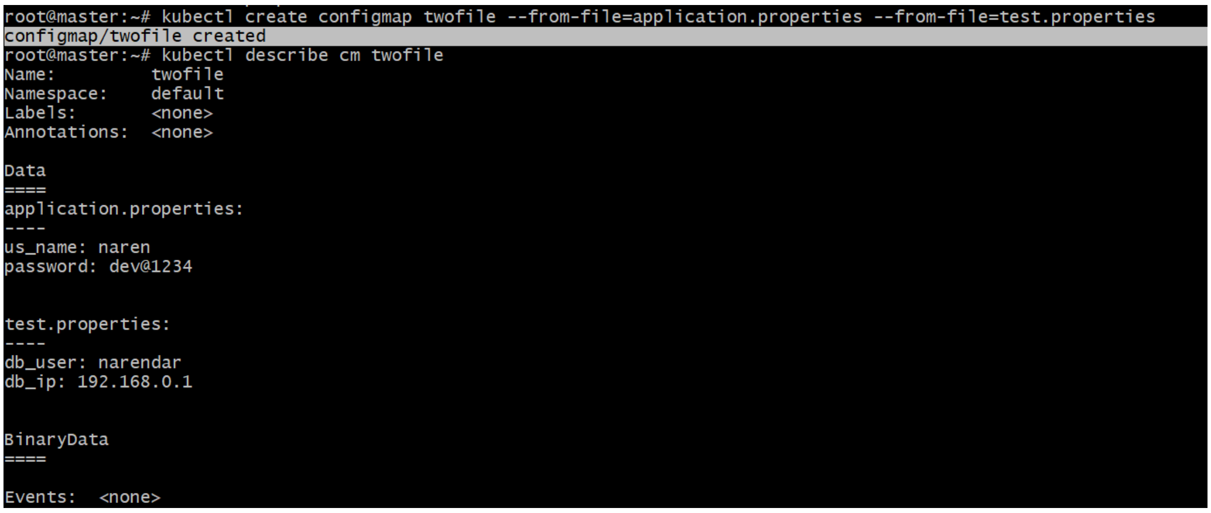
--create a directory



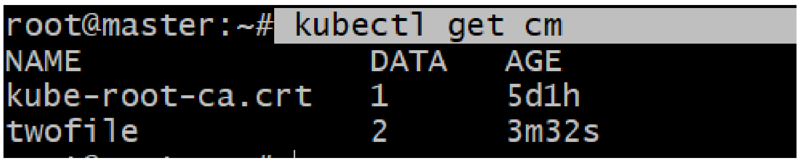
```
root@master:~# mkdir config-dir
root@master:~# ls
''$'\033''[200~two-containers-pod.yaml~'
 config-dir
```

--create configmap with two files(already created files with some data) by using below command

kubectl create configmap twofile --from-file=application.properties --from-file=test.properties



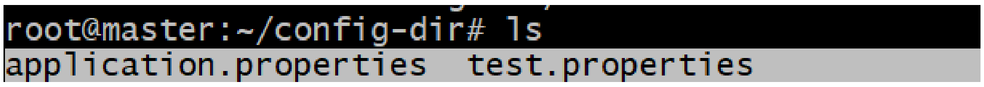
```
root@master:~# kubectl create configmap twofile --from-file=application.properties --from-file=test.properties
configmap/twofile created
root@master:~# kubectl describe cm twofile
Name:         twofile
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
application.properties:
----
us_name: naren
password: dev@1234


test.properties:
----
db_user: narendar
db_ip: 192.168.0.1


BinaryData
====

Events:  <none>
```



```
root@master:~# kubectl get cm
NAME               DATA   AGE
kube-root-ca.crt   1      5d1h
twofile            2      3m32s
```

--now move the files to the directory

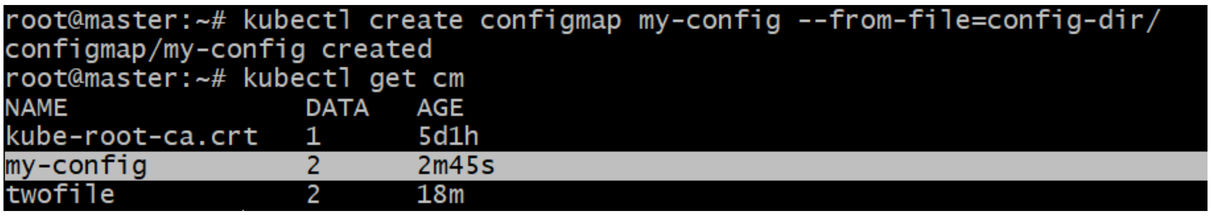
mv application.properties test.properties /root/config-dir/



```
root@master:~/config-dir# ls
application.properties  test.properties
```

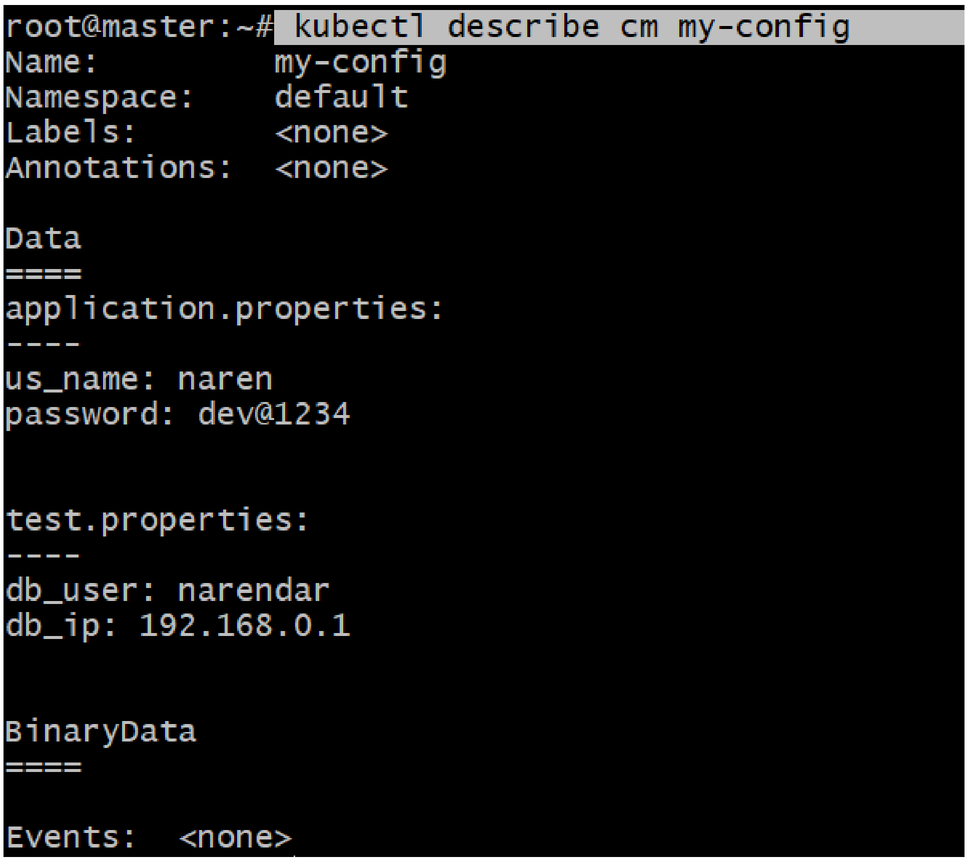
--now Create a ConfigMap from a directory containing multiple files

kubectl create configmap my-config --from-file=config-dir/



```
root@master:~# kubectl create configmap my-config --from-file=config-dir/
configmap/my-config created
root@master:~# kubectl get cm
NAME               DATA   AGE
kube-root-ca.crt   1      5d1h
my-config          2      2m45s
twofile            2      18m
```

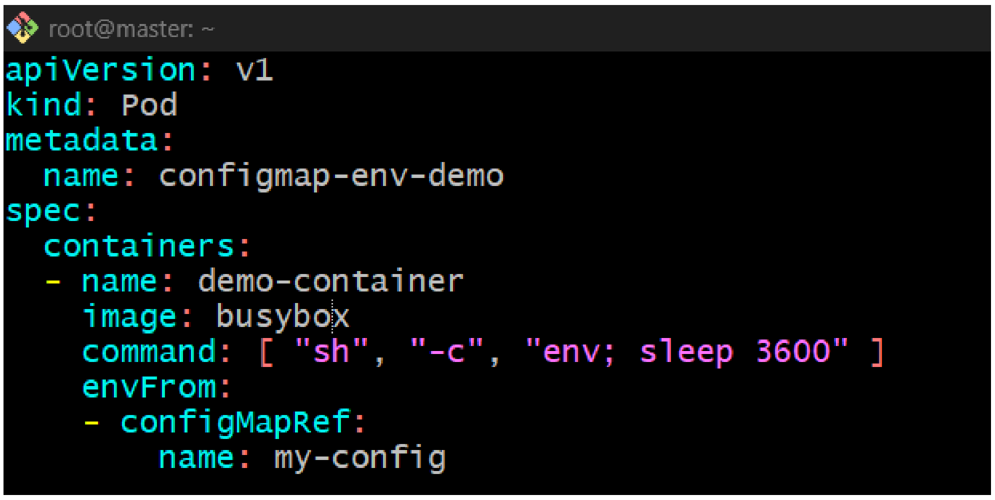
Describe my-config drectory



```
root@master:~# kubectl describe cm my-config
Name:         my-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
application.properties:
----
us_name: naren
password: dev@1234


test.properties:
----
db_user: narendar
db_ip: 192.168.0.1


BinaryData
====

Events:  <none>
```

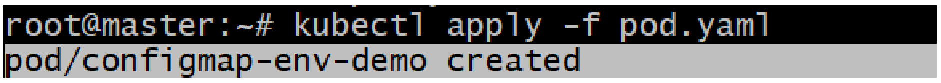
-- Inject ConfigMap into Pod as Environment Variables

Yaml file



```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: configmap-env-demo
spec:
  containers:
  - name: demo-container
    image: busybox
    command: [ "sh", "-c", "env; sleep 3600" ]
    envFrom:
    - configMapRef:
        name: my-config
```

Run yaml file

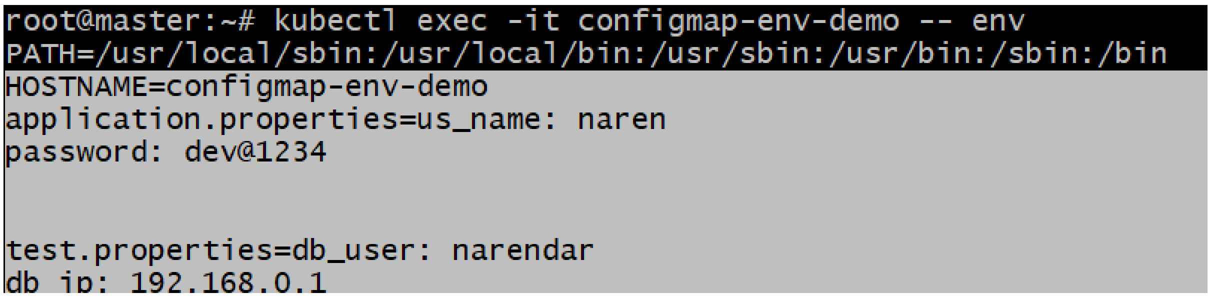


```
root@master:~# kubectl apply -f pod.yaml
pod/configmap-env-demo created
```

-- injected the variables into a pod as environment variables

Check with below command

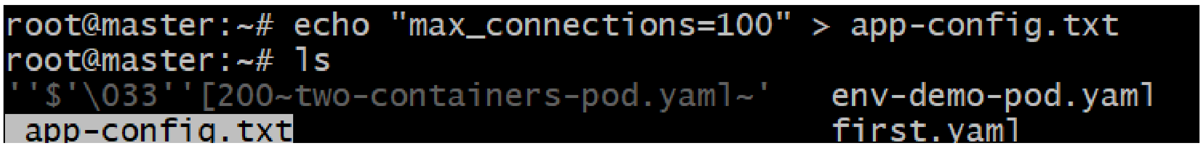
kubectl exec -it configmap-env-demo – env



```
root@master:~# kubectl exec -it configmap-env-demo -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=configmap-env-demo
application.properties=us_name: naren
password: dev@1234


test.properties=db_user: narendar
db ip: 192.168.0.1
```

2) Create a ConfigMap from a file and mount it as a volume inside a pod, ensuring the configuration data is available as files.

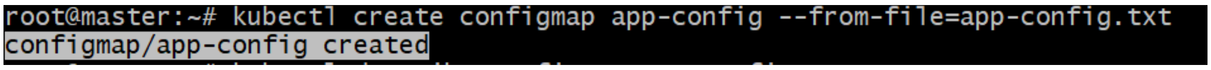
--create a file



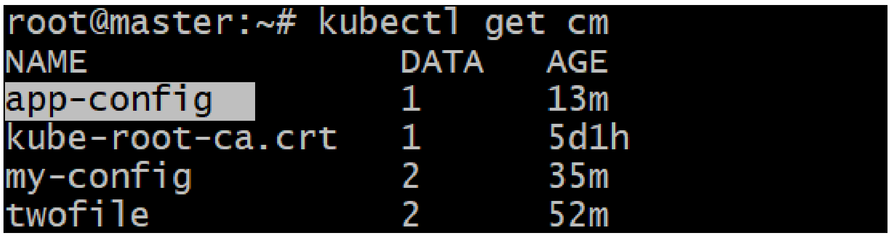
```
root@master:~# echo "max_connections=100" > app-config.txt
root@master:~# ls
''$'\033''[200~two-containers-pod.yaml~'    env-demo-pod.yaml
 app-config.txt                             first.yaml
```

-- Create the ConfigMap from the File

kubectl create configmap app-config --from-file=app-config.txt

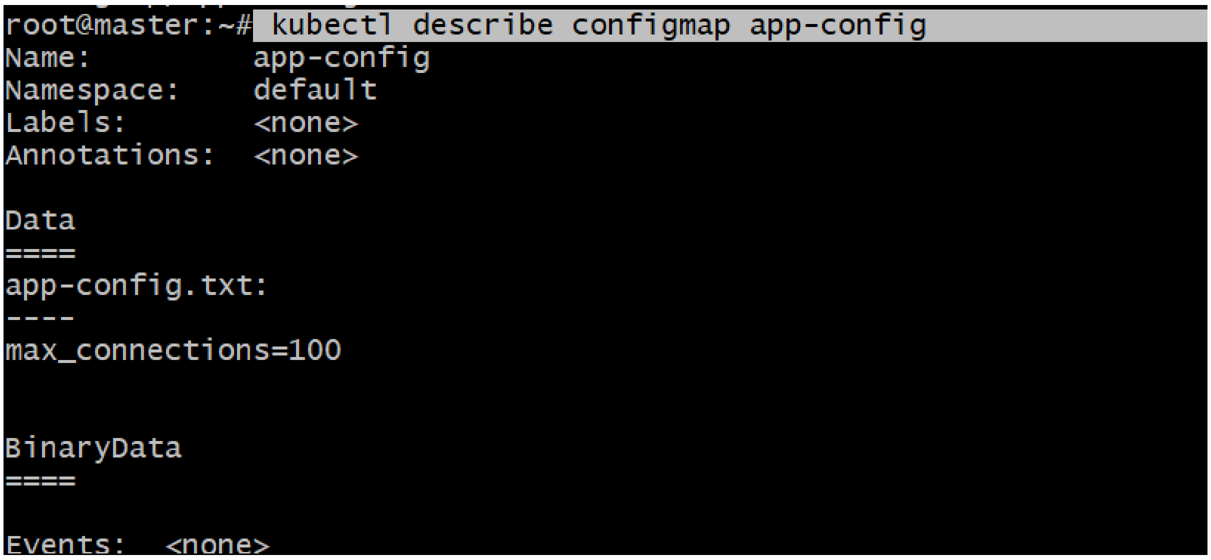


```
root@master:~# kubectl create configmap app-config --from-file=app-config.txt
configmap/app-config created
```



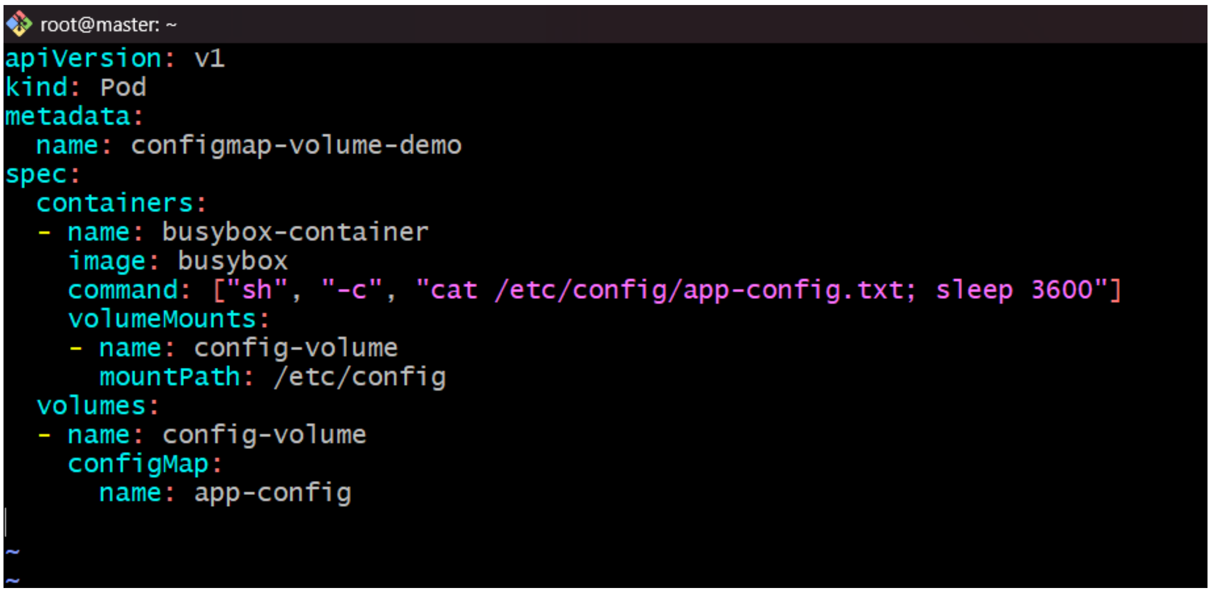
```
root@master:~# kubectl get cm
NAME               DATA   AGE
app-config         1      13m
kube-root-ca.crt   1      5d1h
my-config          2      35m
twofile            2      52m
```

Describe it



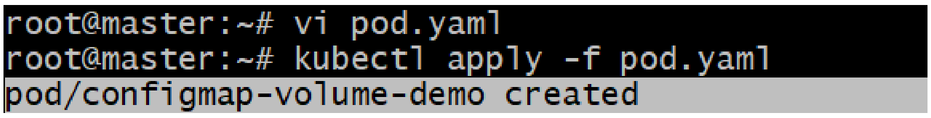
```
root@master:~# kubectl describe configmap app-config
Name:         app-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
app-config.txt:
----
max_connections=100


BinaryData
====

Events:  <none>
```

--now Create a Pod That Mounts the ConfigMap as a Volume



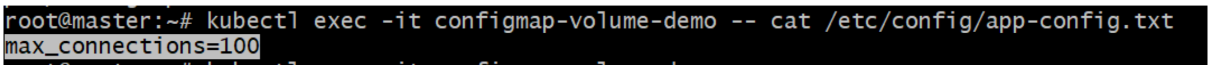
```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: configmap-volume-demo
spec:
  containers:
  - name: busybox-container
    image: busybox
    command: ["sh", "-c", "cat /etc/config/app-config.txt; sleep 3600"]
    volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  volumes:
  - name: config-volume
    configMap:
      name: app-config

~
~
```

Run the yaml



```
root@master:~# vi pod.yaml
root@master:~# kubectl apply -f pod.yaml
pod/configmap-volume-demo created
```

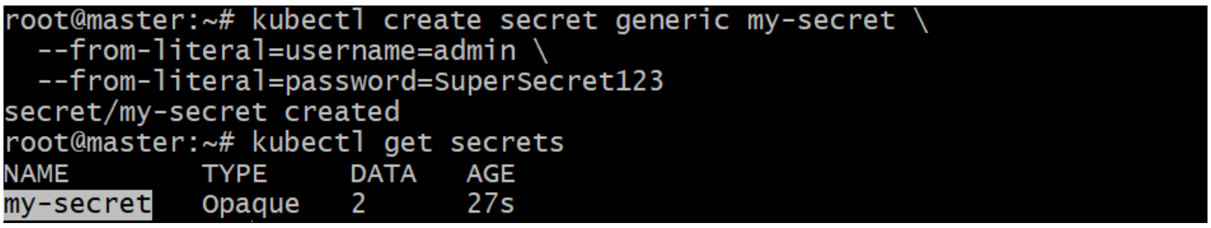
-- Verify Inside the Pod



```
root@master:~# kubectl exec -it configmap-volume-demo -- cat /etc/config/app-config.txt
max_connections=100
```

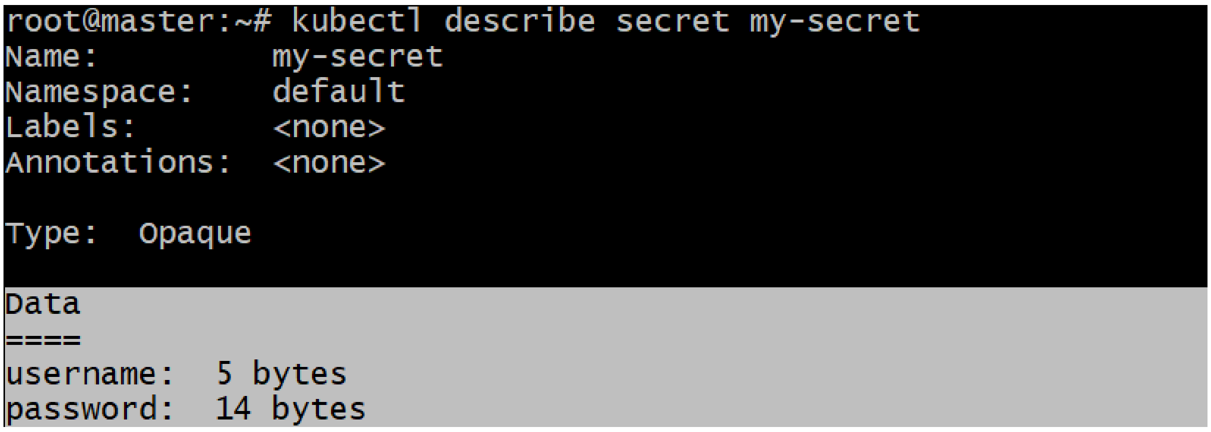
3) Create a Secret with sensitive information (username and password) and inject it into a pod as environment variables.

--create the secret



```
root@master:~# kubectl create secret generic my-secret \
  --from-literal=username=admin \
  --from-literal=password=SuperSecret123
secret/my-secret created
root@master:~# kubectl get secrets
NAME         TYPE      DATA   AGE
my-secret    Opaque    2      27s
```

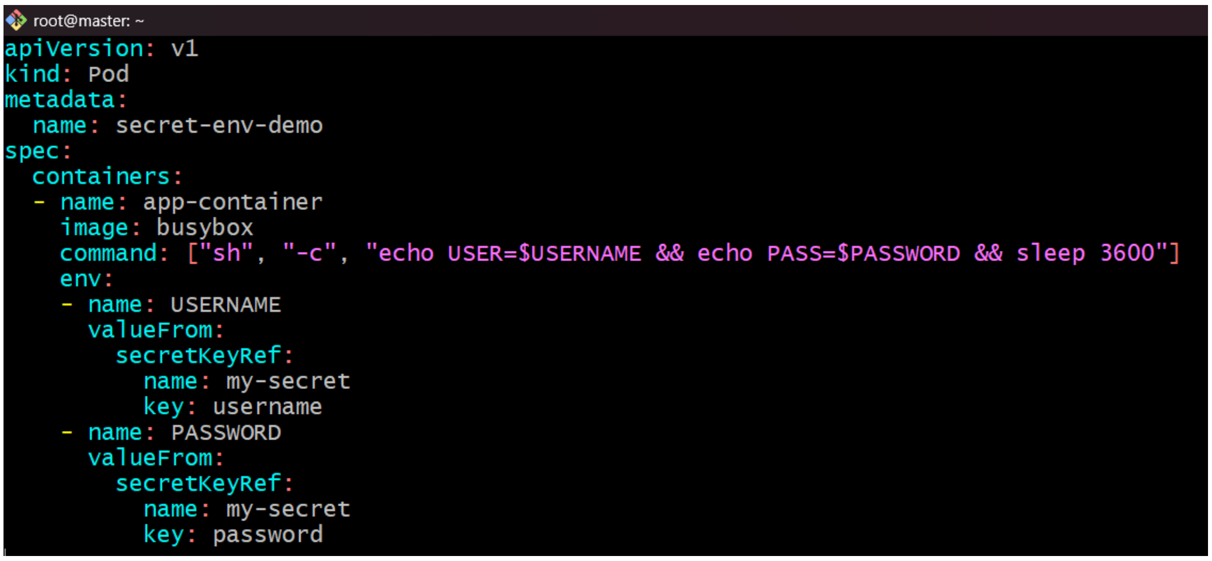
describe it—we can see details as bytes it encrypted



```
root@master:~# kubectl describe secret my-secret
Name:         my-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
username:  5 bytes
password:  14 bytes
```

-- Inject the Secret into a Pod as Environment Variables

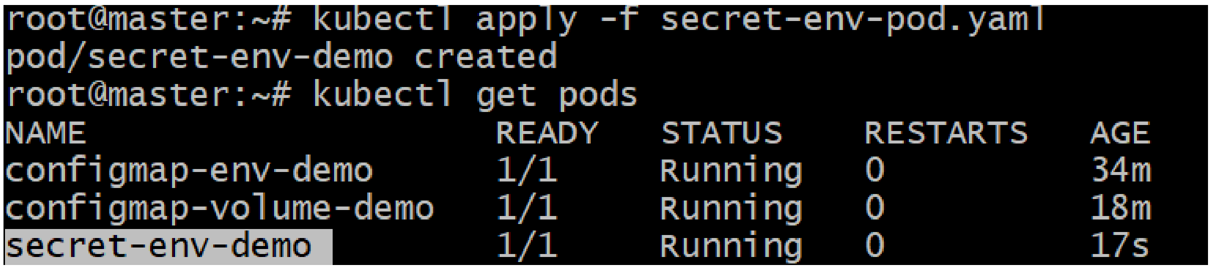
Yaml file



```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-demo
spec:
  containers:
  - name: app-container
    image: busybox
    command: ["sh", "-c", "echo USER=$USERNAME && echo PASS=$PASSWORD && sleep 3600"]
    env:
    - name: USERNAME
      valueFrom:
        secretKeyRef:
          name: my-secret
          key: username
    - name: PASSWORD
      valueFrom:
        secretKeyRef:
          name: my-secret
          key: password
```

run yaml file

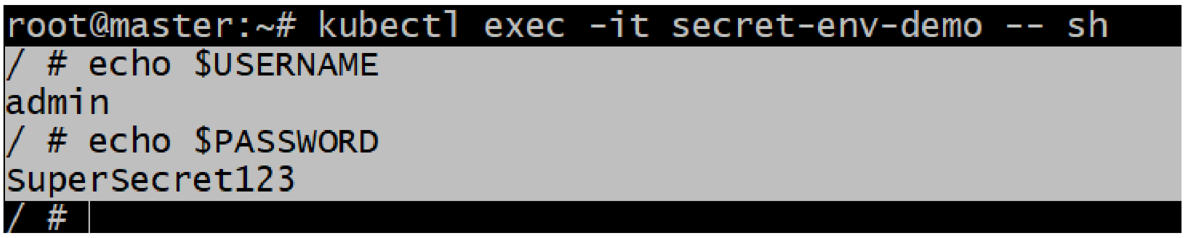
Pod created



```
root@master:~# kubectl apply -f secret-env-pod.yaml
pod/secret-env-demo created
root@master:~# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
configmap-env-demo       1/1     Running   0          34m
configmap-volume-demo    1/1     Running   0          18m
secret-env-demo          1/1     Running   0          17s
```

--injected it into a pod as environment variables

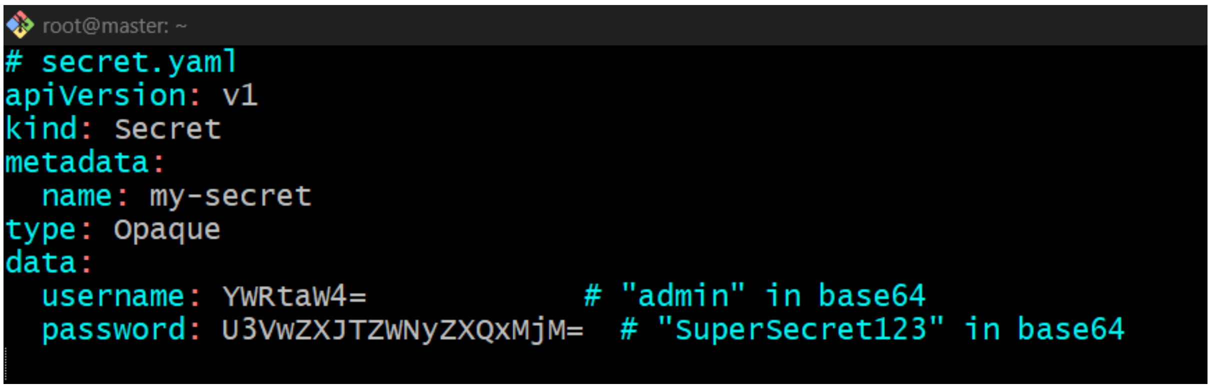
verify it



```
root@master:~# kubectl exec -it secret-env-demo -- sh
/ # echo $USERNAME
admin
/ # echo $PASSWORD
SuperSecret123
/ #
```

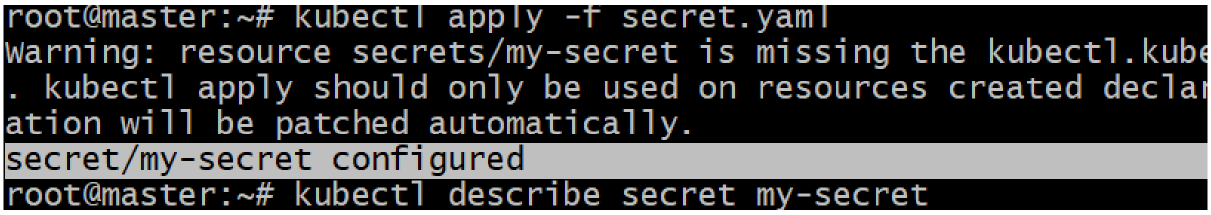
4) Create a Secret using a YAML file, mount it as a volume in a pod, and verify the specific Secret values are available as files.

-- Create the Secret YAML file



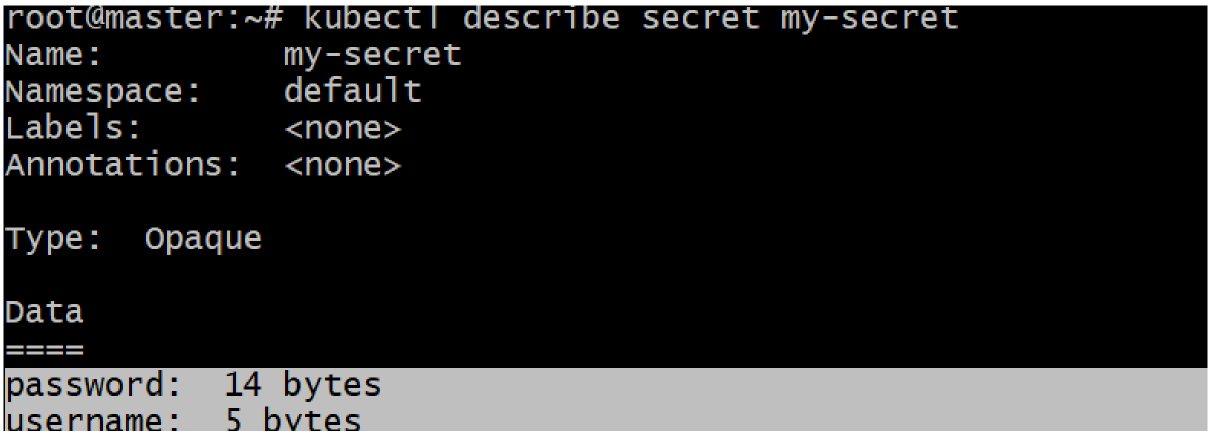
```
root@master: ~
# secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque
data:
  username: YWRtaW4=              # "admin" in base64
  password: U3VwZXJTZWNyZXQxMjM=  # "SuperSecret123" in base64
```

run the yaml



```
root@master:~# kubectl apply -f secret.yaml
Warning: resource secrets/my-secret is missing the kubectl.kube
. kubectl apply should only be used on resources created declar
ation will be patched automatically.
secret/my-secret configured
root@master:~# kubectl describe secret my-secret
```

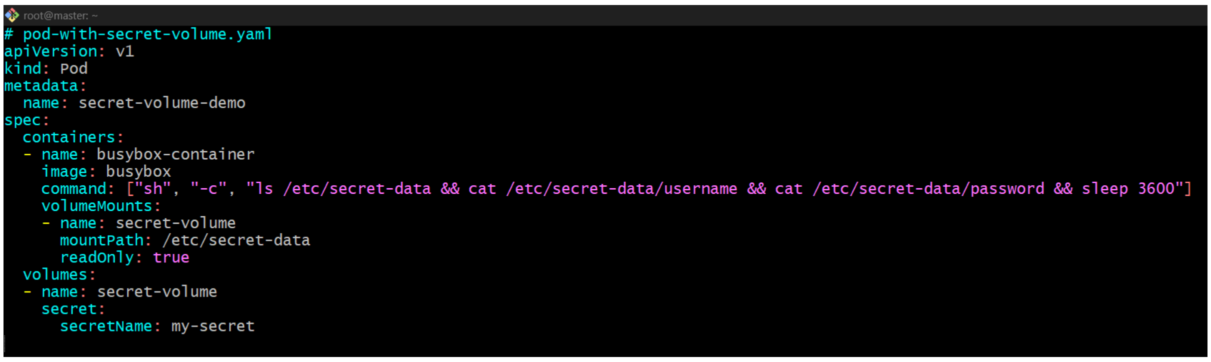
describe it



```
root@master:~# kubectl describe secret my-secret
Name:         my-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
password:  14 bytes
username:  5 bytes
```

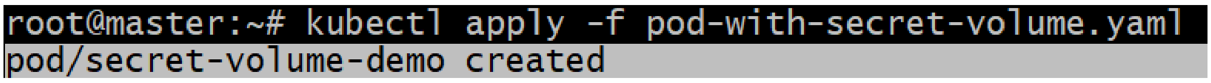
-- Create a Pod that Mounts the Secret as a Volume

Yaml file



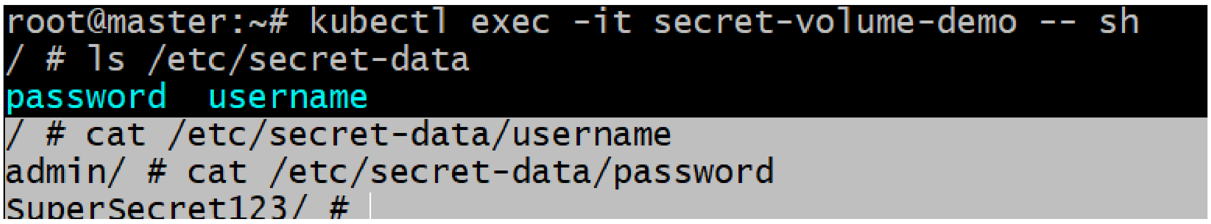
```
# pod-with-secret-volume.yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-volume-demo
spec:
  containers:
  - name: busybox-container
    image: busybox
    command: ["sh", "-c", "ls /etc/secret-data && cat /etc/secret-data/username && cat /etc/secret-data/password && sleep 3600"]
    volumeMounts:
    - name: secret-volume
      mountPath: /etc/secret-data
      readOnly: true
  volumes:
  - name: secret-volume
    secret:
      secretName: my-secret
```

Run the yaml file



```
root@master:~# kubectl apply -f pod-with-secret-volume.yaml
pod/secret-volume-demo created
```

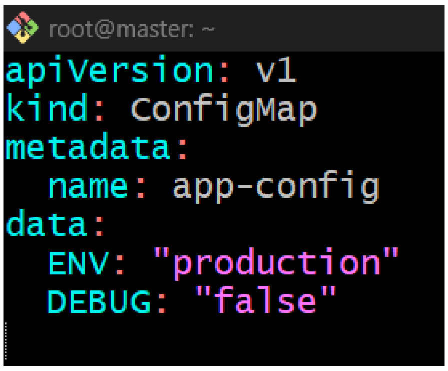
-- Verify Secret Files Inside the Pod



```
root@master:~# kubectl exec -it secret-volume-demo -- sh
/ # ls /etc/secret-data
password  username
/ # cat /etc/secret-data/username
admin/ # cat /etc/secret-data/password
SuperSecret123/ #
```

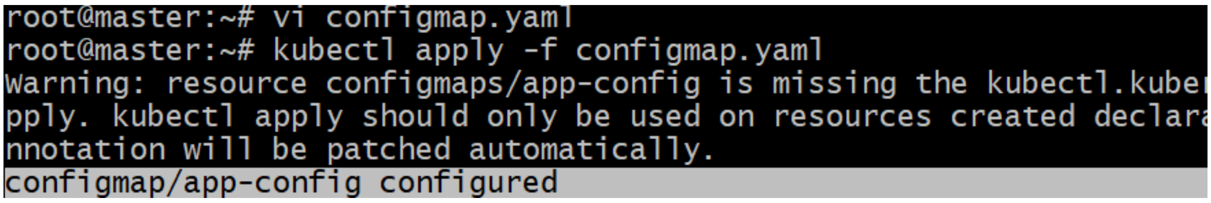
5) Inject a ConfigMap as environment variables and a Secret as files into the same pod, ensuring both are accessible within the pod.

--create the configmap as environment variables



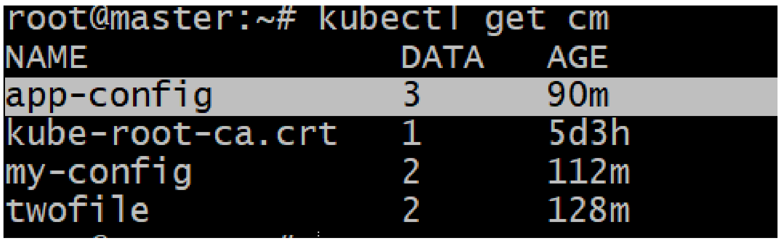
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  ENV: "production"
  DEBUG: "false"
```

Run yaml



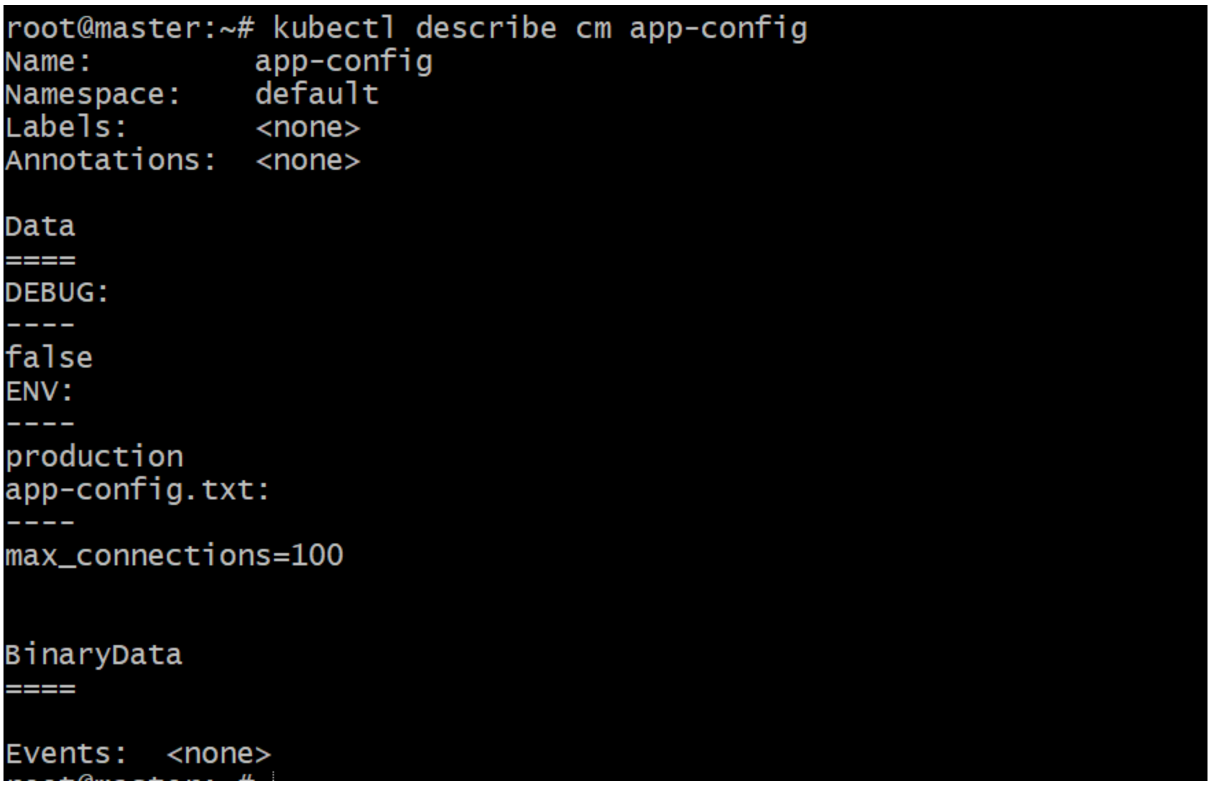
```
root@master:~# vi configmap.yaml
root@master:~# kubectl apply -f configmap.yaml
Warning: resource configmaps/app-config is missing the kubectl.kube
pply. kubectl apply should only be used on resources created declara
nnotation will be patched automatically.
configmap/app-config configured
```

Check cm



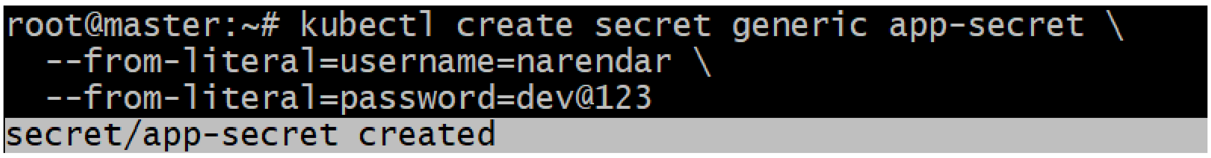
```
root@master:~# kubectl get cm
NAME                DATA   AGE
app-config          3      90m
kube-root-ca.crt    1      5d3h
my-config           2      112m
twofile             2      128m
```

Describe it



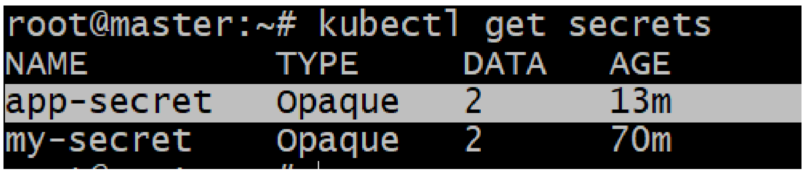
```
root@master:~# kubectl describe cm app-config
Name:         app-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
DEBUG:
----
false
ENV:
----
production
app-config.txt:
----
max_connections=100


BinaryData
====

Events:  <none>
```

-- Create the Secret (as mounted files)



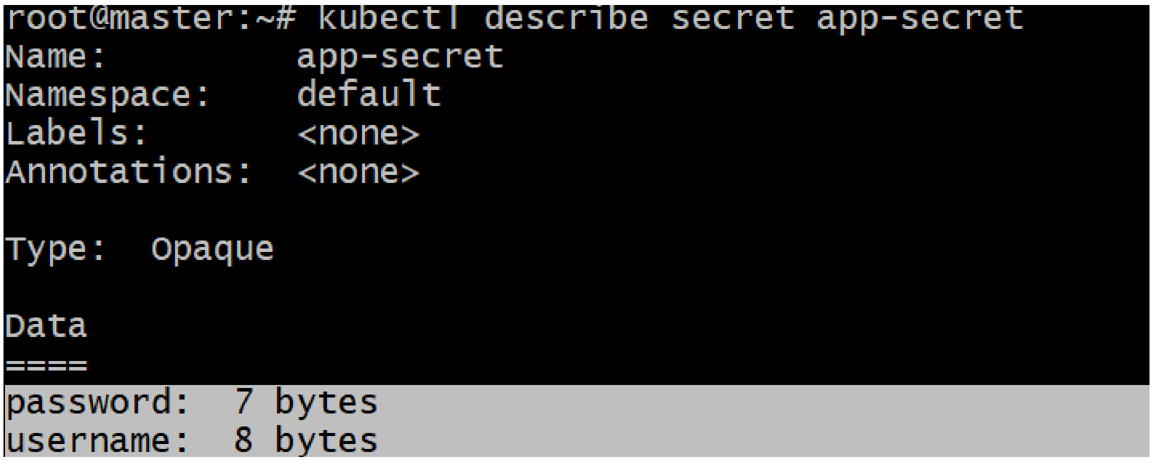
```
root@master:~# kubectl create secret generic app-secret \
  --from-literal=username=narendar \
  --from-literal=password=dev@123
secret/app-secret created
```

Check



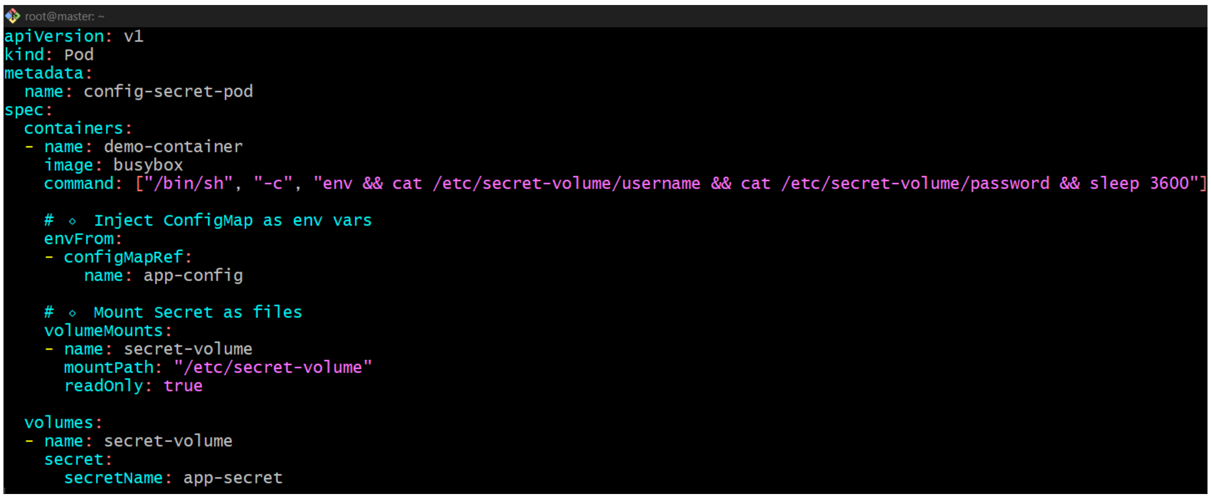
```
root@master:~# kubectl get secrets
NAME         TYPE     DATA   AGE
app-secret   Opaque   2      13m
my-secret    Opaque   2      70m
```

Describe secret



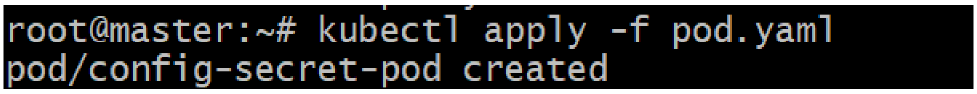
```
root@master:~# kubectl describe secret app-secret
Name:         app-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
password:  7 bytes
username:  8 bytes
```

--yaml file-This is where we inject the ConfigMap and mount the Secret with in a same pod:



```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: config-secret-pod
spec:
  containers:
  - name: demo-container
    image: busybox
    command: ["/bin/sh", "-c", "env && cat /etc/secret-volume/username && cat /etc/secret-volume/password && sleep 3600"]

    # ◦ Inject ConfigMap as env vars
    envFrom:
    - configMapRef:
        name: app-config

    # ◦ Mount Secret as files
    volumeMounts:
    - name: secret-volume
      mountPath: "/etc/secret-volume"
      readOnly: true

  volumes:
  - name: secret-volume
    secret:
      secretName: app-secret
```

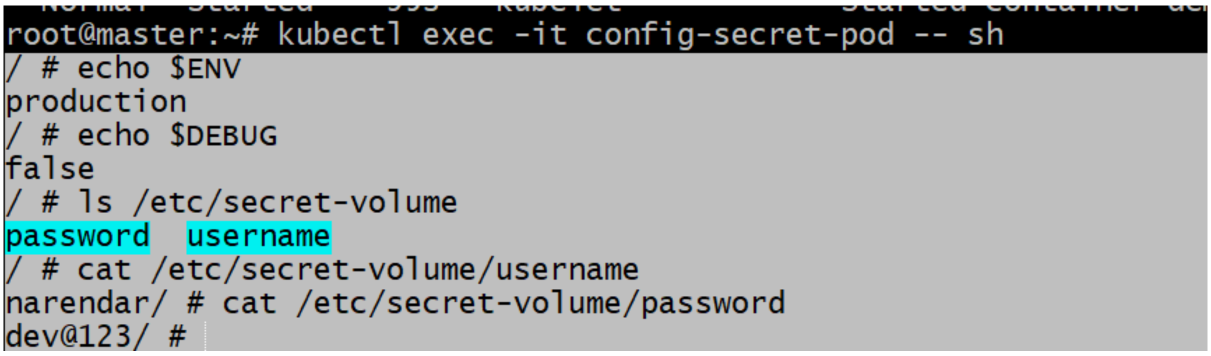
run the yaml



```
root@master:~# kubectl apply -f pod.yaml
pod/config-secret-pod created
```

--log into the config-secret-pod

We can see both environment variables and secret are accessible within the pod



```
root@master:~# kubectl exec -it config-secret-pod -- sh
/ # echo $ENV
production
/ # echo $DEBUG
false
/ # ls /etc/secret-volume
password  username
/ # cat /etc/secret-volume/username
narendar/ # cat /etc/secret-volume/password
dev@123/ #
```