Kubernetes 6

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

- Create a disrectory
- And add files
- echo "mysql" > DB_TYPE
- echo "prod" > ENV
- echo "true" > FEATURE_FLAG
- **Create a ConfigMap from the directory**
- Use `--from-file` with the directory path
- kubectl create configmap app-config \
- --from-file=app-config/
- Now verify
- Kubectl get configmap app-config -o yaml

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl create configmap app-config \
  --from-file=app-config/
configmap/app-config created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get configmap app-config -o yaml
apiVersion: v1
data:
  DB_TYPE: |
    mysql
  ENV: |
    prod
  FEATURE_FLAG: |
    true
kind: ConfigMap
metadata:
  creationTimestamp: "2026-01-06T06:25:14Z"
  name: app-config
  namespace: default
  resourceVersion: "125671"
  uid: 6265c2c6-3206-4aa8-b6c7-301259e044c0
[ec2-user@ip-172-31-4-234 ~]$
```

- Inject ConfigMap as environment variables into a Pod

Write a yaml file

- Vi pod.yml
- apiVersion: v1
- kind: Pod
- metadata:
- name: configmap-env-pod
- spec:
- containers:
- - name: demo-container
- image: nginx
- envFrom:
- - configMapRef:
- name: app-config
- Deploy it
- Kubectl apply -f pod.yml

```
[ec2-user@ip-172-31-4-234 ~]$ cat pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: configmap-env-pod
spec:
  containers:
    - name: demo-container
      image: nginx
      envFrom:
        - configMapRef:
            name: app-config

[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f pod.yml
pod/configmap-env-pod created
[ec2-user@ip-172-31-4-234 ~]$
```

- Check environmental variables
- kubectl exec -it configmap-env-pod – env

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it configmap-env-pod -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=configmap-env-pod
NGINX_VERSION=1.29.4
NJS_VERSION=0.9.4
NJS_RELEASE=1~trixie
PKG_RELEASE=1~trixie
DYNPKG_RELEASE=1~trixie
DB_TYPE=mysql

ENV=prod

FEATURE_FLAG=true

KUBERNETES_PORT_443_TCP_PROTO=tcp
```

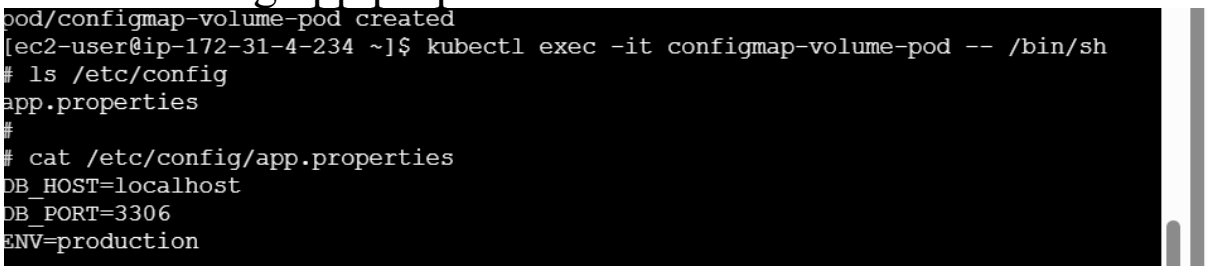
1. 2 . Create a ConfigMap from a file and mount it as a volume inside a pod, ensuring the configuration data is available as files.

    - Vi app.properties
    - Add data to it
    - DB_HOST=localhost
    - DB_PORT=3306
    - ENV=production
    - Save it
    - Create a config map from file
    - kubectl create configmap app-config-file \
    - --from-file=app.properties
    - To verify it
    - kubectl get configmap app-config-file -o yaml

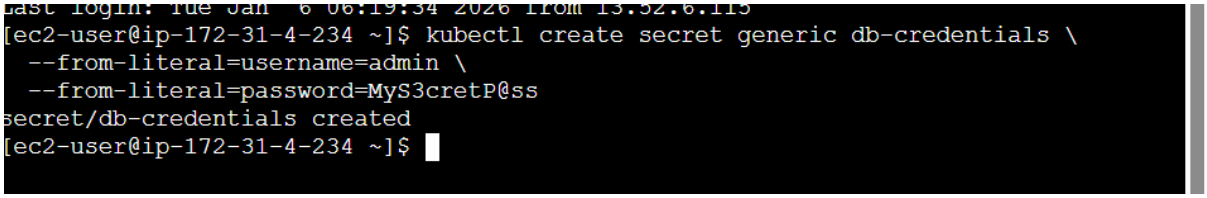
```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get configmap app-config-file -o yaml
apiVersion: v1
data:
  app.properties: |+
    DB_HOST=localhost
    DB_PORT=3306
    ENV=production

kind: ConfigMap
metadata:
  creationTimestamp: "2026-01-06T06:39:51Z"
  name: app-config-file
  namespace: default
  resourceVersion: "127000"
  uid: deeb4fc0-9789-4dc3-9ae6-cd1870d499a9
[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

- pod yaml , mount config map as volume
- vi pod-volume.yml
- write a yaml file
- apiVersion: v1
- kind: Pod
- metadata:
- name: configmap-volume-pod
- spec:
- containers:
- - name: demo-container
- image: nginx
- volumeMounts:
- - name: config-volume
- mountPath: /etc/config
- volumes:
- - name: config-volume
- configMap:
- name: app-config-file
- Deploy it
- Kubectl apply -f pod-volume.yml
- To verify

- Kubectl exec -it configmap-voulme-pod -- /bin/sh
- cat /etc/config/app.properties
- 

1. 3 . Create a Secret with sensitive information (username and password) and inject it into a pod as environment variables.

- kubectl create secret generic db-credentials \
- --from-literal=username=admin \
- --from-literal=password=MyS3cretP@ss
- Secret db credentials will be created .
- 

- Write a yml file
- vi secret-env-pod.yml
- apiVersion: v1
- kind: Pod
- metadata:
- name: secret-env-pod
- spec:
- containers:
- - name: demo-container
- image: nginx
- env:
- - name: DB_USERNAME
- valueFrom:
- secretKeyRef:

- name: db-credentials
- key: username
- - name: DB_PASSWORD
- valueFrom:
- secretKeyRef:
- name: db-credentials
- key: password
- Deploy it
- Kubectl apply -f secret-env-pod.yml
- Verify the pod
- kubectl exec -it secret-env-pod -- env | grep DB_

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it secret-env-pod -- env | grep DB_
DB_USERNAME=admin
DB_PASSWORD=MyS3cretP@ss
[ec2-user@ip-172-31-4-234 ~]$
```

1. 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.

    - Will use base64
    - Encoding
    - Data to printable ASCII characters
    - echo -n 'admin' | base64
    - echo -n 'MyS3cretP@ss' | base64

```
[ec2-user@ip-172-31-4-234 ~]$ echo -n 'admin' | base64
echo -n 'MyS3cretP@ss' | base64
YWRtaW4=
TXlTM2NyZXRQQHNz
[ec2-user@ip-172-31-4-234 ~]$
```

    - Vi secret.yml
    - apiVersion: v1
    - kind: Secret
    - metadata:

- name: db-secret
- type: Opaque
- data:
- username: YWRtaW4=
- password: TXlTM2NyZXRQQHNz
- Now deploy it
- Kubectl apply -f secret.yml
- Now write one more yml file
- vi secret-volume-pod.yml
- apiVersion: v1
- kind: Pod
- metadata:
- name: secret-volume-pod
- spec:
- containers:
- - name: demo-container
- image: nginx
- volumeMounts:
- - name: secret-volume
- mountPath: /etc/secret
- readOnly: true
- volumes:
- - name: secret-volume
- secret:
- secretName: db-secret
- Deploy it
- Kubectl apply -f secret-volume-podf.yml
- Verify secret values in pod
- kubectl exec -it secret-volume-pod -- /bin/sh
- ls /etc/secret
- cat /etc/secret/username

- cat /etc/secret/password

```
pod/secret-volume-pod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it secret-volume-pod -- /bin/sh
# ls /etc/secret
password  username
#
# cat /etc/secret/username
cat /etc/secret/password
admin# MyS3cretP@ss#
#
```

1. 5 . Inject a ConfigMap as environment variables and a Secret as files into the same pod, ensuring both are accessible within the pod.

- Create config map
- kubectl create configmap app-config \
- --from-literal=APP_ENV=production \
- --from-literal=APP_MODE=debug
- Verify it

```
configmap/app-config created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get configmap app-config -o yaml
apiVersion: v1
data:
  APP_ENV: production
  APP_MODE: debug
kind: ConfigMap
metadata:
  creationTimestamp: "2026-01-06T06:58:29Z"
  name: app-config
  namespace: default
  resourceVersion: "128706"
  uid: 9500a8d8-29f7-4f5d-9501-c304f70864f2
[ec2-user@ip-172-31-4-234 ~]$
```

- i-00bbf6e27330e6872 (master)
- Create a secret
- kubectl create secret generic app-secret \
- --from-literal=username=admin \
- --from-literal=password=MyS3cretP@ss
- Now ferify by
- Kubectl get secret app-secret

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl create secret generic app-secret \
  --from-literal=username=admin \
  --from-literal=password=MyS3cretP@ss
secret/app-secret created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get secret app-secret
NAME         TYPE     DATA   AGE
app-secret   Opaque   2      8s
[ec2-user@ip-172-31-4-234 ~]$
```

- Write a pod yaml file for env and secret
- vi combined-pod.yml
- apiVersion: v1
- kind: Pod
- metadata:
- name: configmap-secret-pod
- spec:
- containers:
- - name: demo-container
- image: nginx
-
- # ConfigMap injected as environment variables
- envFrom:
- - configMapRef:
- name: app-config
-
- # Secret mounted as files
- volumeMounts:
- - name: secret-volume
- mountPath: /etc/secret
- readOnly: true
-
- volumes:
- - name: secret-volume
- secret:
- secretName: app-secret
- Deploy it

- Kubectl apply -f po-combined.yml
- Now
- Check
- Kubectl exec -it configmap-secret-pod – env | grep APP
- OUT PUT

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get secret app-secret
NAME         TYPE     DATA   AGE
app-secret   Opaque   2      8s
[ec2-user@ip-172-31-4-234 ~]$ vi combined-pod.yml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f combined-pod.yml
pod/configmap-secret-pod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it configmap-secret-pod -- env | grep
 APP_
APP_ENV=production
APP_MODE=debug
[ec2-user@ip-172-31-4-234 ~]$
```

- NOW check secret files
- kubectl exec -it configmap-secret-pod -- ls /etc/secret
- kubectl exec -it configmap-secret-pod -- cat /etc/secret/username
- kubectl exec -it configmap-secret-pod -- cat /etc/secret/password
- output

```
kubectl exec -it configmap-secret-pod -- cat /etc/secret/password
adminMyS3cretP@ss[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it configmap-secret-
pod -- cat /etc/secret/password
MyS3cretP@ss[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it configmap-secret-pod -
- cat /etc/secret/password
MyS3cretP@ss[ec2-user@ip-172-31-4-234 ~]$
```