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

Create a directory and move inside to that directory.

- **mkdir config-dir**
- **cd config-dir**

```
[root@master ~]# mkdir config-dir
[root@master ~]# ls
config-dir
[root@master ~]# cd config-dir/
```

Create files inside that directory

- **touch APP_ENV**

inside that file write this

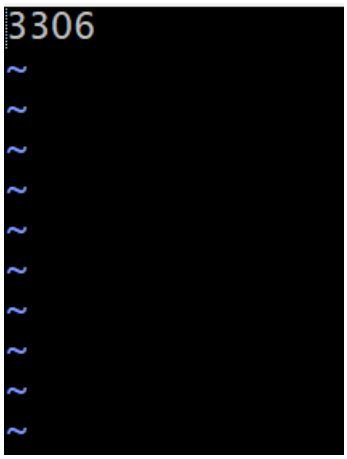
```
production
~
~
~
~
~
~
~
~
~
~
~
~
~
```

- **touch DB_HOST**

inside that write this

- **touch DB_PORT**

inside that give port number




```
[root@master ~]# cd config-dir/
[root@master config-dir]# ls
APP_ENV  DB_HOST  DB_PORT
```

Come to root directory and create configmap

- **kubectl create configmap app-config --from-file=config-dir/**

```
[root@master ~]# kubectl create configmap app-config --from-file=config-dir/
configmap/app-config created
[root@master ~]#
```

- **kubectl get configmap app-config**

```
[root@master ~]# kubectl get configmap app-config
NAME         DATA   AGE
app-config   3      103s
[root@master ~]#
```

- **kubectl describe cm app-config**

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

Data
====
APP_ENV:
----
production


DB_HOST:
----
mysql-service


DB_PORT:
----
3306



BinaryData
====

Events:  <none>
[root@master ~]#
```

- vi pod.yml

apiVersion: v1

kind: Pod

metadata:

name: configmap-env-pod

spec:

containers:

- name: nginx

image: nginx

envFrom:

- configMapRef:

name: app-config

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: configmap-env-pod
spec:
  containers:
  - name: nginx
    image: nginx
    envFrom:
    - configMapRef:
        name: app-config
```

- **kubectl apply -f pod.yml**
- **kubectl get pods**
- **kubectl exec -it configmap-env-pod -- env**

```
[root@master ~]# vi pod.yml
[root@master ~]# kubectl apply -f pod.yml
pod/configmap-env-pod created
[root@master ~]# kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
configmap-env-pod   1/1     Running   0          10s
[root@master ~]# 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
APP_ENV=production

DB_HOST=mysql-service

DB_PORT=3306

KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
TERM=xterm
HOME=/root
[root@master ~]#
```

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

```
APP_NAME=sample-app
APP_ENV=production
APP_PORT=8080
~
~
```

- **kubectl create configmap app-config --from-file=app.properties**

```
[root@master ~]# vi app.properties
[root@master ~]# kubectl create configmap app-config --from-file=app.properties
configmap/app-config created
[root@master ~]#
```

- **kubectl describe cm app-config**

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

Data
====
app.properties:
----
APP_NAME=sample-app
APP_ENV=production
APP_PORT=8080



BinaryData
====

Events:  <none>
[root@master ~]#
```

Create volume for cm

apiVersion: v1

kind: Pod

metadata:

name: configmap-volume-pod

spec:

containers:

- name: nginx

image: nginx

volumeMounts:

- name: config-volume

mountPath: /etc/config

volumes:

- name: config-volume

configMap:

name: app-config

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: configmap-volume-pod
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  volumes:
  - name: config-volume
    configMap:
      name: app-config
```

- **kubectl apply -f configmap-volume-pod.yml**
- **kubectl get pods**

```
[root@master ~]# vi configmap-volume-pod.yml
[root@master ~]# kubectl apply -f configmap-volume-pod.yml
pod/configmap-volume-pod created
[root@master ~]# kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
configmap-volume-pod    1/1     Running   0          26s
[root@master ~]#
```

- **kubectl exec -it configmap-volume-pod -- ls /etc/config**
- **kubectl exec -it configmap-volume-pod -- cat /etc/config/app.properties**

```
[root@master ~]# kubectl exec -it configmap-volume-pod -- ls /etc/config
app.properties
[root@master ~]# kubectl exec -it configmap-volume-pod -- cat /etc/config/app.properties
APP_NAME=sample-app
APP_ENV=production
APP_PORT=8080

[root@master ~]#
```

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

create a secret with username and password

**kubectl create secret generic db-secret **

**--from-literal=username=admin **

--from-literal=password=Admin@123

```
[root@master ~]# kubectl create secret generic db-secret \
  --from-literal=username=admin \
  --from-literal=password=Admin@123
secret/db-secret created
[root@master ~]#
```

- **kubectl get secrets**

- **kubectl describe secret db-secret**

```
[root@master ~]# kubectl get secrets
NAME        TYPE     DATA   AGE
db-secret   Opaque   2      4m51s
[root@master ~]# kubectl describe secret db-secret
Name:         db-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
password:  9 bytes
username:  5 bytes
[root@master ~]#
```

vi secret-env-pod.yml

apiVersion: v1

kind: Pod

metadata:

name: secret-env-pod

spec:

containers:

- name: nginx

image: nginx

env:

- name: DB_USERNAME

valueFrom:

secretKeyRef:

name: db-secret

key: username

- name: DB_PASSWORD

valueFrom:

secretKeyRef:

name: db-secret

key: password

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: nginx
    image: nginx
    env:
    - name: DB_USERNAME
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: username
    - name: DB_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: password
```

- **kubectl apply -f secret-env-pod.yml**
- **kubectl get pods**
- **kubectl exec -it secret-env-pod -- env | grep DB_**

```
[root@master ~]# vi secret-env-pod.yml
[root@master ~]# kubectl apply -f secret-env-pod.yml
pod/secret-env-pod created
[root@master ~]# kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
secret-env-pod   1/1     Running   0          9s
[root@master ~]# kubectl exec -it secret-env-pod -- env | grep DB_
DB_USERNAME=admin
DB_PASSWORD=Admin@123
[root@master ~]#
```

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.

- **echo -n admin | base64**
- **echo -n Admin@123 | base64**

```
[root@master ~]# echo -n admin | base64
YWRtaW4=
[root@master ~]# echo -n Admin@123 | base64
QWRtaW5AMTIz
[root@master ~]#
```

vi db-secret.yml

apiVersion: v1

kind: Secret

metadata:

name: db-secret

type: Opaque

data:

username: YWRtaW4=

password: QWRtaW5AMTIz

```
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
data:
  username: YWRtaW4=
  password: QWRtaW5AMTIz
```

- **kubectl apply -f db-secret.yml**
- **kubectl get secret db-secret**
- **kubectl describe secret db-secret**

```
[root@master ~]# vi db-secret.yml
[root@master ~]# kubectl apply -f db-secret.yml
secret/db-secret created
[root@master ~]# kubectl get secret db-secret
NAME        TYPE     DATA   AGE
db-secret   Opaque   2      17s
[root@master ~]# kubectl describe secret db-secret
Name:         db-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
password:  9 bytes
username:  5 bytes
[root@master ~]#
```

- **vi secret-volume-pod.yml**

apiVersion: v1

kind: Pod

metadata:

name: secret-volume-pod

spec:

containers:

- name: nginx

image: nginx

volumeMounts:

- name: secret-volume

mountPath: /etc/secret-data

readOnly: true

volumes:

- name: secret-volume

secret:

secretName: db-secret

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-volume-pod
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - name: secret-volume
      mountPath: /etc/secret-data
      readOnly: true
  volumes:
  - name: secret-volume
    secret:
      secretName: db-secret
```

- **kubectl apply -f secret-volume-pod.yml**
- **kubectl get pods**
- **kubectl exec -it secret-volume-pod -- ls /etc/secret-data**

```
[root@master ~]# vi secret-volume-pod.yml
[root@master ~]# kubectl apply -f secret-volume-pod.yml
pod/secret-volume-pod created
[root@master ~]# kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
secret-volume-pod   1/1     Running   0          9s
[root@master ~]# kubectl exec -it secret-volume-pod -- ls /etc/secret-data
password  username
```

- **kubectl exec -it secret-volume-pod -- cat /etc/secret-data/username**
- **kubectl exec -it secret-volume-pod -- cat /etc/secret-data/password**

```
[root@master ~]# kubectl exec -it secret-volume-pod -- cat /etc/secret-data/username
admin[root@masterkubectl exec -it secret-volume-pod -- cat /etc/secret-data/passwordsword
Admin@123[root@master ~]#
```

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

Create a configmap

**kubectl create configmap app-config **

**--from-literal=APP_ENV=production **

--from-literal=APP_DEBUG=false

- **kubectl describe configmap app-config**

```
[root@master ~]# kubectl create configmap app-config \
  --from-literal=APP_ENV=production \
  --from-literal=APP_DEBUG=false
configmap/app-config created
[root@master ~]# kubectl describe configmap app-config
Name:         app-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
APP_DEBUG:
----
false

APP_ENV:
----
production


BinaryData
====

Events:  <none>
[root@master ~]#
```

Create a secret

**kubectl create secret generic db-secret **

**--from-literal=username=admin **

--from-literal=password=Admin@123

- **kubectl describe secret db-secret**

```
[root@master ~]# kubectl create secret generic db-secret \
  --from-literal=username=admin \
  --from-literal=password=Admin@123
secret/db-secret created
[root@master ~]# kubectl describe secret db-secret
Name:         db-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
password:  9 bytes
username:  5 bytes
[root@master ~]#
```

vi config-secret-pod.yml

apiVersion: v1

kind: Pod

metadata:

name: config-secret-pod

spec:

containers:

- name: nginx

image: nginx

envFrom:

- configMapRef:

name: app-config

volumeMounts:

- name: secret-volume

mountPath: /etc/secret-data

readOnly: true

volumes:

- name: secret-volume

secret:

secretName: db-secret

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: config-secret-pod
spec:
  containers:
  - name: nginx
    image: nginx
    envFrom:
    - configMapRef:
        name: app-config
    volumeMounts:
    - name: secret-volume
      mountPath: /etc/secret-data
      readOnly: true
  volumes:
  - name: secret-volume
    secret:
      secretName: db-secret
```

- **kubectl apply -f config-secret-pod.yml**
- **kubectl get pods**
- **kubectl exec -it config-secret-pod -- env | grep APP**

```
[root@master ~]# vi config-secret-pod.yml
[root@master ~]# kubectl apply -f config-secret-pod.yml
pod/config-secret-pod created
[root@master ~]# kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
config-secret-pod   1/1     Running   0          9s
secret-volume-pod   1/1     Running   0          14m
[root@master ~]# kubectl exec -it config-secret-pod -- env | grep APP
APP_DEBUG=false
APP_ENV=production
[root@master ~]#
```

- **kubectl exec -it config-secret-pod -- ls /etc/secret-data**
- **kubectl exec -it config-secret-pod -- cat /etc/secret-data/username**
- **kubectl exec -it config-secret-pod -- cat /etc/secret-data/password**

```
[root@master ~]# kubectl exec -it config-secret-pod -- ls /etc/secret-data
password  username
[root@master ~]# kubectl exec -it config-secret-pod -- cat /etc/secret-data/username
admin[root@masterkubectl exec -it config-secret-pod -- cat /etc/secret-data/passwordsword
Admin@123[root@master ~]#
```