A ConfigMap allows you to store non-sensitive configuration data in key-value pairs.

Pods can access ConfigMaps in two ways:

- As environment variables.
- As volumes that mount files.

Create a ConfigMap from a Literal Key-Value Pair

```
kubectl create configmap my-config --from-literal=key1=value1 --from-literal=key2=value2
```

💡 Explanation: This command creates a ConfigMap named my-config with two keys: key1 and key2, with respective values.

ConfigMap YAML Definition Example

Here’s an example of a ConfigMap defined in a YAML file. It stores two key-value pairs.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  key1: value1
  key2: value2
```

As Environment Variables

You can expose the keys in a ConfigMap as environment variables inside a Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: mycontainer
      image: nginx
      envFrom:
        - configMapRef:
            name: my-config
```

💡 Explanation: This will make all the key-value pairs in the my-config ConfigMap available as environment variables in the mypod Pod.

2. As Volumes

You can mount the ConfigMap as a volume, which will place each key in the ConfigMap as a separate file inside the Pod.

```
apiVersion: v1
kind: Pod
```

```
metadata:
  name: mypod
spec:
  containers:
    - name: mycontainer
      image: nginx
      volumeMounts:
        - name: config-volume
          mountPath: /etc/config  # Config files will be mounted here
  volumes:
    - name: config-volume
      configMap:
        name: my-config


# ConfigMaps commands

# 1. Create a ConfigMap from a literal key-value pair
kubectl create configmap <configmap-name> --from-literal=key1=value1 --from-
literal=key2=value2

# 2. Create a ConfigMap from a file
kubectl create configmap <configmap-name> --from-file=path/to/config/file

# 3. Create a ConfigMap from a directory
kubectl create configmap <configmap-name> --from-file=path/to/config/directory/

# 4. Apply a ConfigMap from a YAML file
kubectl apply -f configmap.yaml

# 5. List all ConfigMaps in the current namespace
kubectl get configmaps

# 6. Get detailed information about a specific ConfigMap
kubectl describe configmap <configmap-name>

# 7. Edit a ConfigMap (open the editor to modify it)
kubectl edit configmap <configmap-name>

# 8. Delete a ConfigMap
kubectl delete configmap <configmap-name>

# 9. Get the content of a ConfigMap in YAML format
kubectl get configmap <configmap-name> -o yaml
```