# Chapter 13 ConfigMaps and Secrets

## ConfigMaps

- Decouple the config details from the container image
- Pass configuration data as key-value pairs
- Consumed by Pods or any other system components and controllers using either:
  - env variables
  - sets of commands/arguments
  - volumes

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

```
kubectl get configmaps my-config-o yaml
```

```
$ kubectl get configmaps my-config -o yaml
apiVersion: v1
data:
  key1: value1
  key2: value2
kind: ConfigMap
metadata:
  creationTimestamp: 2019-05-31T07:21:55Z
  name: my-config
  namespace: default
  resourceVersion: "241345"
  selfLink: /api/v1/namespaces/default/configmaps/my-config
  uid: d35f0a3d-45d1-11e7-9e62-080027a46057
```

Creating configmap from a file

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: customer1
data:
  TEXT1: Customer1_Company
  TEXT2: Welcomes You
  COMPANY: Customer1 Company Technology Pct. Ltd.
```

`kubectl create -f cofigmap.yml`

Creating permission configuration with file

```
permission=read-only
allowed="true"
resetCount=3
```

`kubectl create configmap permission-config --from-file=permission-reset.properties`

## Using ConfigMaps inside Pods

- Values of specific ConfigMap keys can be retrieved with env vars

```
containers:
  - name: myapp-full-container
    image: myapp
    envFrom:
    - configMapRef:
      name: full-config-map
```

all the `myapp-full-container` Container's env vars receive the values of the `full-config-map` ConfigMap keys.

```
containers:
  - name: myapp-specific-container
    image: myapp
    env:
    - name: SPECIFIC_ENV_VAR1
      valueFrom:
        configMapKeyRef:
          name: config-map-1
          key: SPECIFIC_DATA
    - name: SPECIFIC_ENV_VAR2
      valueFrom:
        configMapKeyRef:
          name: config-map-2
```

```
          key: SPECIFIC_INFO
```

`myapp-specific-container` Container's env vars receive their values from specific key-value pairs from separate ConfigMaps

Mounting ConfigMap as a Volume inside a Pod

```
containers:
  - name: myapp-vol-container
    image: myapp
    volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  volumes:
  - name: config-volume
    configMap:
      name: vol-config-map
```

For each key in the ConfigMap, a file gets created in the mount path and the content of that file becomes the respective key's value

## Secrets

- referenced in deployments
- encoded sensitive information
- key-value pairs
- stored as plain text inside etcd
  - could be encrypted using a feature at the Api server level

`kubectl create secret generic my-password --from-literal=password=mysqlpassword`

- create a secret called `my-password` with key-value of `password` - `mysqlpassword`

```
$ kubectl get secret my-password
NAME          TYPE      DATA    AGE
my-password   Opaque    1       8m

$ kubectl describe secret my-password
Name:          my-password
Namespace:     default
Labels:        <none>
Annotations:   <none>

Type  Opaque

Data
====
password:  13 bytes
```

- Two types of maps for sensitive information inside a Secret
  - data
    - Must be encoded using base64
  - stringData

Using yaml config file

`echo mysqlpassword | base64` to create an encoded password:
"bXlzcWxwYXNzd29yZAo="

```
apiVersion: v1
kind: Secret
metadata:
```

```
  name: my-password
type: Opaque
data:
  password: bXlzcWxwYXNzd29yZAo=
```

```
kubectl create -f secret-config.yaml
```

## Create a secret from a file and display its details

```
echo -n 'bXlzcWxwYXNzd29yZAo=' > password.txt
```

```
kubectl create secret generic my-file-password --from-file=password.txt
```

```
$ kubectl get secret my-file-password
NAME                 TYPE      DATA    AGE
my-file-password     Opaque    1       8m

$ kubectl describe secret my-file-password
Name:           my-file-password
Namespace:      default
Labels:         <none>
Annotations:    <none>

Type  Opaque

Data
====
password.txt:  13 bytes
```

## Use Secrets Inside Pods

- Using secrets as env vars

```
spec:
```

```
  containers:
  - image: wordpress:4.7.3-apache
    name: wordpress
    env:
    - name: WORDPRESS_DB_PASSWORD
      valueFrom:
        secretKeyRef:
          name: my-password
          key: password
```

  - Referencing `password` key of the `my-password` Secret and assign its value to `WORDPRESS_DB_PASSWORD`

- Using secrets as files from a pod

```
spec:
  containers:
  - image: wordpress:4.7.3-apache
    name: wordpress
    volumeMounts:
    - name: secret-volume
      mountPath: "/etc/secret-data"
      readOnly: true
  volumes:
  - name: secret-volume
    secret:
      secretName: my-password
```