While deploying an application, we may need to pass such runtime parameters like configuration details, permissions, passwords, tokens, etc. When we want to pass sensitive information, we can use the Secret API resource.

ConfigMap:
Allows you to decouple the configuration details from the container
Using the configmap api we can send the configuration data in (key - value) format which will later consumed by the pod and any other component of the kubernetes

Simple config map file with key value pair:

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: custom-map
data:
 cluster_name: ExportExecutor
 ip: localhost
 company_name: contentserv
```

Pick Config map from the file:
First, we need to create a file permission-reset.properties with the following configuration data:

```
permission=read-only

allowed="true"

resetCount=3
```

We can then create the ConfigMap with the following command

It will automatically generate the configmap file form the given properties file

In the container we can get either get config map value from a single file or from multiple files

Secrets :

We can provide the username and password for the application directly from the deployment file or from the config map but then it will exposed to end user, Hence you will have to encrypt such information by using the secrets

Secret object can help by allowing us to encode the sensitive information before sharing it. With Secrets, we can share sensitive information like passwords, tokens, or keys in the form of key-value pairs, similar to ConfigMaps

We can create a Secret manually from a YAML configuration file. The example file below is named mypass.yaml. There are two types of maps for sensitive information inside a Secret: data and stringData.

Please note that base64 encoding does not mean encryption, and anyone can easily decode our encoded data:

```
$ echo "bXlzcWxwYXNzd29yZAo=" | base64 --decode

mysqlpassword
```

Therefore, make sure you do not commit a Secret's configuration file in the source code.

Pick secret From file

We can create the secret from the plain text file as well

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

Now we can create the Secret from the password.txt file:

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

secret/my-file-password created
```

Using secrets

Using Secrets as Environment Variables

Below we reference only the password key of the my-password Secret and assign its value to the WORDPRESS_DB_PASSWORD environment variable:

....

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

....

Using Secrets as Files from a Pod

We can also mount a Secret as a Volume inside a Pod. The following example creates a file for each my-password Secret key (where the files are named after the names of the keys), the files containing the values of the Secret:

```
....
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
....
```