



ConfigMaps & Secrets

ConfigMaps

- We can look at ConfigMaps in several ways:
 - Set of key/value pairs
 - Config file
- ConfigMaps are combined with Pods at start of execution

Config file example

```
# This is a sample config file that I might use to configure an application
```

```
parameter1 = value1
```

```
parameter2 = value2
```

```
kubectl create configmap my-config \
```

```
--from-file=my-config.txt \
```

```
--from-literal=extra-param=extra-value \
```

```
--from-literal=another-param=another-value
```

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

Using a ConfigMap

- Filesystem
 - You can mount a ConfigMap into a Pod. A file is created for each entry based on the key name. The contents of that file are set to the value.
- Environment variable
 - A ConfigMap can be used to dynamically set the value of an environment variable.
- Command-line argument
 - Kubernetes supports dynamically creating the command line for a container based on ConfigMap values.

ConfigMap usage

```
apiVersion: v1
kind: Pod
metadata:
  name: test-config
spec:
  containers:
    - name: test-container
      image: <some-image>
      imagePullPolicy: Always
      command:
        - "$(EXTRA_PARAM)"
```

```
env:
  - name: ANOTHER_PARAM
    valueFrom:
      configMapKeyRef:
        name: my-config
        key: another-param
  - name: EXTRA_PARAM
    valueFrom:
      configMapKeyRef:
        name: my-config
        key: extra-param
```

volumeMounts:

- name: config-volume
mountPath: /config

volumes:

- name: config-volume

configMap:

name: my-config

restartPolicy: Never

Secrets

- ConfigMaps are great for most configuration data, there is certain data that is extra-sensitive.
 - Passwords
 - security tokens
 - or other types of private keys
- K8s has native support for storing and handling this data with care
- Secrets are exposed to pods via explicit declaration in pod manifests

Creating Secrets

```
kubectl create secret generic my-app-tls \  
  --from-file=my-app.crt \  
  --from-file=my-app.key
```

```
kubectl describe secrets my-app-tls
```


Creating Secrets

- `--from-file=<filename>`
Load from the file with the secret data key the same as the filename.
- `--from-file=<key>=<filename>`
Load from the file with the secret data key explicitly specified.
- `--from-file=<directory>`
Load all the files in the specified directory where the filename is an acceptable key name.
- `--from-literal=<key>=<value>`
Use the specified key/value pair directly.

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app-tls
spec:
  containers:
    - name: my-app-tls
      image: nginx
      imagePullPolicy: Always
      volumeMounts:
        - name: tls-certs
          mountPath: "/tls"
          readOnly: true
  volumes:
    - name: tls-certs
      secret:
        secretName: my-app-tls
```

Updating Secrets

```
kubectl replace -f <filename>
```

```
# if you created them from manifest before
```

```
kubectl apply -f <filename>
```

```
# recreate from files
```

```
kubectl create secret generic my-app-tls \  
--from-file=my-app.crt --from-file=my-app.key \  
--dry-run -o yaml | kubectl replace -f -
```