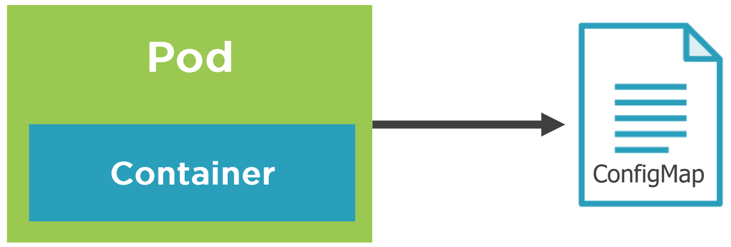
ConfigMaps provide a way to store configuration information and provide it to containers.

ConfigMaps

Provides a way to inject configuration data into a container

Can store entire files or provide key/value pairs:

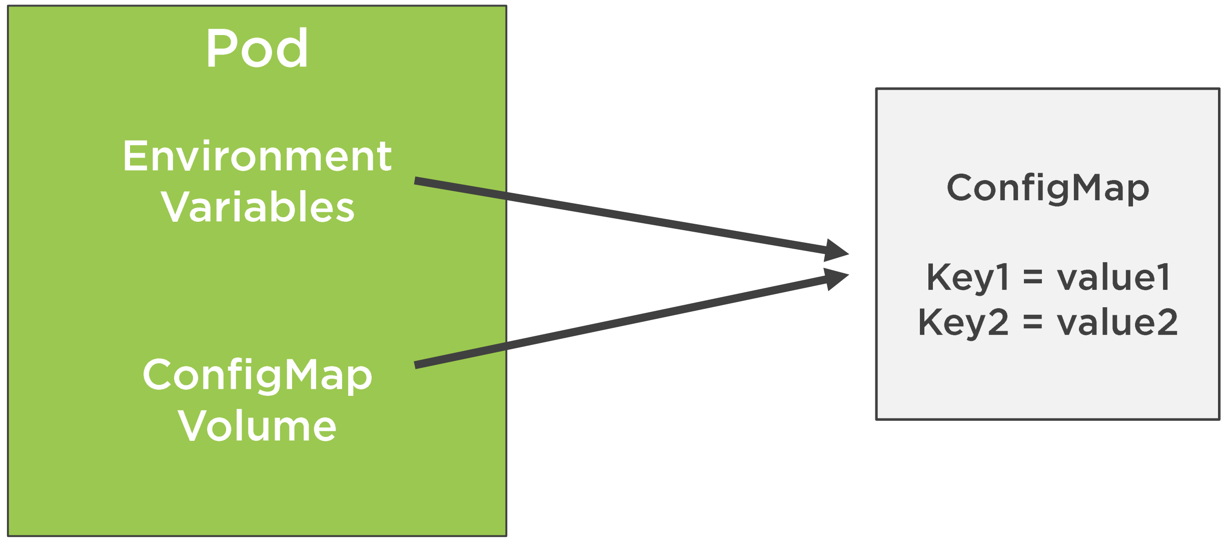
- Store in a File. Key is the filename, value is the file contents (can be JSON, XML, keys/values, etc.).
- Provide on the command-line
- ConfigMap manifest



Accessing ConfigMap Data in a Pod

ConfigMaps can be accessed from a Pod using:

- **Environment variables (key/value)**
- **ConfigMap Volume (access as files)**



Defining Values in a ConfigMap Manifest

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-settings
  labels:
    app: app-settings
data:
  enemies: aliens
  lives: "3"
  enemies.cheat: "true"
  enemies.cheat.level=noGoodRotten
```

```
# Create from a ConfigMap manifest
kubectl create -f file.configmap.yml
```

◀ A ConfigMap resource

◀ Name of ConfigMap

◀ ConfigMap data

Defining Key/Value Pairs in a File

```
enemies=aliens

lives=3

enemies.cheat=true

enemies.cheat.level=noGoodRotten
```

```
# Create a ConfigMap using data from a file
kubectl create configmap [cm-name]
  --from-file=[path-to-file]
```

```
apiVersion: v1

kind: ConfigMap

data:

  game.config: |-

    enemies=aliens

    lives=3

    enemies.cheat=true

    enemies.cheat.level=noGoodRotten
```

- Key/value pairs defined in a file named game.config
- Nested properties can be defined and assigned a value
- Note that the file name is used as the key for the values
- Your application can now work with the content just as it would a normal configuration file (JSON, XML, keys/values, could be used)

Defining Key/Value Pairs in an Env File

```
enemies=aliens
lives=3
enemies.cheat=true
enemies.cheat.level=noGoodRotten
```

```
# Create a env ConfigMap using data from a file
kubectl create configmap [cm-name]
  --from-env-file=[path-to-file]
```

```
apiVersion: v1
kind: ConfigMap
data:
  enemies=aliens
  lives=3
  enemies.cheat=true
  enemies.cheat.level=noGoodRotten
```

- Key/value pairs can be defined in an "environment" variables file (game-config.env)
- Nested properties can be defined and assigned a value
- Note that the file name is NOT included as a key

```
# Create a ConfigMap using data from a config file
kubectl create configmap [cm-name] --from-file=[path-to-file]

# Create ConfigMap from an env file
kubectl create configmap [cm-name] --from-env-file=[path-to-file]


# Create a ConfigMap from individual data values
kubectl create configmap [cm-name]
  --from-literal=apiUrl=https://my-api
  --from-literal=otherKey=otherValue

# Create from a ConfigMap manifest
kubectl create -f file.configmap.yml
```

Creating a ConfigMap

A ConfigMap can be created using **kubectl create**

Key command-line switches include:

- --from-file
- --from-env-file
- --from-literal

```
# Get a ConfigMap
kubectl get cm [cm-name] -o yaml
```

Getting a ConfigMap

kubectl get cm can be used to get a ConfigMap and view its contents

Accessing a ConfigMap: Environment Vars

Pods can access ConfigMap values through environment vars

ENEMIES environment variable created (value=aliens)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-settings
data:
  enemies: aliens
  lives: "3"
  enemies.cheat: "true"
  enemies.cheat.level=noGoodRotten
```

```
apiVersion: apps/v1
...
spec:
  template:
    ...
  spec:
    containers: ...
    env:
    - name: ENEMIES
      valueFrom:
        configMapKeyRef:
          name: app-settings
          key: enemies
```

Environment variable name

Accessing a ConfigMap: Environment Vars

envFrom can be used to load all ConfigMap keys/values into environment variables

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-settings
data:
  enemies: aliens
  lives: "3"
  enemies.cheat: "true"
  enemies.cheat.level=noGoodRotten
```

```
apiVersion: apps/v1
...
spec:
  template:
    ...
  spec:
    containers: ...
      envFrom:
      - configMapRef:
        name: app-settings
```

Environment variables created for all data keys

Accessing a ConfigMap: Volume

ConfigMap values can be loaded through a Volume

Each key is converted to a file - value is added into the file

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-settings
data:
  enemies: aliens
  lives: "3"
  enemies.cheat: "true"
  enemies.cheat.level=noGoodRotten
```

```
apiVersion: apps/v1
...
spec:
  template:
    ...
  spec:
    volumes:
      - name: app-config-vol
        configMap:
          name: app-settings
     containers:
        volumeMounts:
          - name: app-config-vol
            mountPath: /etc/config
```

ConfigMap values stored at /etc/config