Converting ConfigMap Output into Environment Variables

In this lesson, we will explore the conversion of ConfigMap into environment variables.

WE'LL COVER THE FOLLOWING

- Altering the Perpetual Process
  - Looking Into First Definition
  - Creating the Pod
  - Looking Into Second Definition
  - Creating the Pod

Altering the Perpetual Process

All the examples we've seen so far are differing only in the source. The destination is always the same. No matter whether ConfigMap is created from a file, from a directory, from literal values, or from an environment file, it perpetually resulted in one or more files being injected into a container.

This time we'll try something different. We'll see how we can convert a ConfigMap into environment variables.

Looking Into First Definition

Let's take a look at a sample definition.

```
cat cm/alpine-env.yml
```

The **output** is as follows.

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

```
  name: alpine-env
spec:
  containers:
  - name: alpine
    image: alpine
    command: ["sleep"]
    args: ["100000"]
    env:
    - name: something
      valueFrom:
        configMapKeyRef:
          name: my-config
          key: something
    - name: weather
      valueFrom:
        configMapKeyRef:
          name: my-config
          key: weather
```

> The major difference, when compared with `cm/alpine.yml`, is that `volumeMounts` and `volumes` sections are gone. This time we have an `env` section.

Instead of a `value` field, we have `valueFrom`. Further on, we declared that it should get values from a ConfigMap (`configMapKeyRef`) named `my-config`. Since that ConfigMap has multiple values, we specified the `key` as well.

Creating the Pod

Let's create the Pod.

```
kubectl create \
    -f cm/alpine-env.yml
#Wait for a few seconds before executing the below command
kubectl exec -it alpine-env -- env
```

We created the Pod and executed the `env` command inside its only container. The output of the latter command, limited to the relevant parts, is as follows.

```
...
weather=sunny
something=else
...
```

There's another, often more useful way to specify environment variables from a ConfigMap. Before we try it, we'll remove the currently running Pod.

```
kubectl delete \
    -f cm/alpine-env.yml
```

Looking Into Second Definition

Let's take a look at yet another definition.

```
cat cm/alpine-env-all.yml
```

The **output** is as follows.

```
apiVersion: v1
kind: Pod
metadata:
  name: alpine-env
spec:
  containers:
  - name: alpine
    image: alpine
    command: ["sleep"]
    args: ["100000"]
    envFrom:
    - configMapRef:
        name: my-config
```

> The difference is only in the way environment variables are defined.

This time, the syntax is much shorter. We have `envFrom`, instead of the `env` section. It can be either `configMapRef` or `secretRef`. Since we did not yet explore Secrets, we'll stick with the prior. Inside `configMapRef` is the `name` reference to the `my-config` ConfigMap.

Creating the Pod

Let's see it in action.

```
kubectl create \
    -f cm/alpine-env-all.yml
#Wait for a few seconds before executing the below command
kubectl exec -it alpine-env -- env
```

We created the Pod and retrieved all the environment variables from inside its only container. The output of the latter command, limited to the relevant parts, is as follows.

```
...
something=else
weather=sunny
...
```

> The result is the **same** as before. The difference is only in the way we define environment variables.

With `env.valueFrom.configMapKeyRef` syntax, we need to specify each ConfigMap key separately. That gives us control over the scope and the relation with the names of container variables.

The `envFrom.configMapRef` converts all ConfigMap's data into environment variables. That is often a better and simpler option if you don't need to use different names between ConfigMap and environment variable keys. The syntax is short, and we don't need to worry whether we forgot to include one of the ConfigMap's keys.

In the next lesson, we will learn to define ConfigMaps as a YAML file.