Injecting Configuration from a Single File

In this lesson, we will learn to inject configuration from a single file using the ConfigMap Volume.

WE'LL COVER THE FOLLOWING

Creating a ConfigMap

In its purest, and probably the most common form, a ConfigMap takes a single file. For example, we can create one from the `cm/prometheus-conf.yml` file.

```
kubectl create cm my-config \
    --from-file=cm/prometheus-conf.yml
```

We created a ConfigMap (`cm`) called `my-config`. The data of the map is the content of the `cm/prometheus-conf.yml` file.

Looking into the Description

Let's describe it, and see what we'll get.

```
kubectl describe cm my-config
```

The **output** is as follows.

```
Name:         my-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
prometheus-conf.yml:
----
global:
  scrape_interval:     15s

scrape_configs:
  - job_name: prometheus
    metrics_path: /prometheus/metrics
    static_configs:
      - targets:
        - localhost:9090

Events:  <none>
```

The important part is located below `Data`. We can see the key which, in this case, is the name of the file (`prometheus-conf.yml`). Further down you can see the content of the file. If you execute `cat cm/prometheus-conf.yml`, you'll see that it is the same as what we saw from the ConfigMap's description.

Mounting the ConfigMap

ConfigMap is useless by itself. It is yet another Volume which, like all the others, needs a mount.

Pod with Mounted ConfigMap

Let's take a look at a Pod specification defined in `cm/alpine.yml`.

```
cat cm/alpine.yml
```

The **output** is as follows.

```
apiVersion: v1
kind: Pod
metadata:
  name: alpine
spec:
  containers:
  - name: alpine
    image: alpine
    command: ["sleep"]
    args: ["100000"]
    volumeMounts:
    - name: config-vol
```

```
      - name: config-vol
        mountPath: /etc/config
  volumes:

  - name: config-vol
    configMap:
      name: my-config
```

The essential sections are `volumeMounts` and `volumes`. Since `volumeMounts` are the same no matter the type of the Volume, there's nothing special about it. We defined that it should be based on the volume called `config-vol` and that it should mount the path `/etc/config`. The `volumes` section uses `configMap` as the type and, in this case, has a single item `name`, that coincides with the name of the ConfigMap we created earlier.

Creating the Pod

Let's create the Pod and see what happens.

```
kubectl create -f cm/alpine.yml
kubectl get pods
```

Please confirm that the Pod is indeed running before moving on.

Verification

Let's see the content of the `/etc/config` directory inside the Pod's only container.

```
kubectl exec -it alpine -- \
    ls /etc/config
```

The **output** is as follows.

```
prometheus-conf.yml
```

The `/etc/config` now has a single file that coincides with the file we stored in the ConfigMap.

Let's add `-l` to the `ls` command we executed a moment ago.

```
kubectl exec -it alpine --  \
  ls -l /etc/config
```

The **output** is as follows.

```
total 0
lrwxrwxrwx    1 root     root            26 Jun 20 06:04 prometheus-conf.yml -> ..data/promet
```

You'll see that `prometheus-conf.yml` is a link to `..data/prometheus-conf.yml`.

If you dig deeper, you'll see that `..data` is also a link to the directory named from a timestamp. And so on, and so forth. For now, the exact logic behind all the links and the actual files is not of great importance. From the functional point of view, there is `prometheus-conf.yml`, and our application can do whatever it needs to do with it.

Let's confirm that the content of the file inside the container is indeed the same as the source file we used to create the ConfigMap.

```
kubectl exec -it alpine -- \
    cat /etc/config/prometheus-conf.yml
```

The **output** should be the same as the contents of the `cm/prometheus-conf.yml` file.

We saw one combination of ConfigMap. Let's see what else we can do with it.

Deleting the Objects

We'll remove the objects we created thus far and start over.

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

The command to delete the ConfigMap is as follows.

```
kubectl delete cm my-config
```

We are not limited to a single `--from-file` argument. We can specify as many as we need.

In the next lesson, we will learn to inject configurations from multiple files into containers.