Management of configuration data is one of the challenges involved in building and maintaining complex application infrastructures. Luckily, Kubernetes offers functionality that helps to maintain application configurations in the form of ConfigMaps. In this lesson, we will discuss what ConfigMaps are, how to create them, some of the ways that ConfigMap data can be passed in to containers running within Kubernetes Pods.

Relevant Documentation

- https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/

Lesson Reference

Here's an example of a yaml descriptor for a ConfigMap containing some data:

```
apiVersion: v1
kind: ConfigMap
metadata:
   name: my-config-map
data:
   myKey: myValue
   anotherKey: anotherValue
```

Passing ConfigMap data to a container as an environment variable looks like this:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-configmap-pod
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', "echo $(MY_VAR) && sleep 3600"]
    env:
    - name: MY_VAR
      valueFrom:
        configMapKeyRef:
          name: my-config-map
          key: myKey
```

It's also possible to pass ConfigMap data to containers, in the form of file using a mounted volume, like so:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-configmap-volume-pod
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', "echo $(cat /etc/config/myKey) && sleep 3600"]
    volumeMounts:
      - name: config-volume
        mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: my-config-map
```

In the lesson, we'll also use the following commands to explore how the ConfigMap data interacts with pods and containers:

```
kubectl logs my-configmap-pod

kubectl logs my-configmap-volume-pod

kubectl exec my-configmap-volume-pod -- ls /etc/config

kubectl exec my-configmap-volume-pod -- cat /etc/config/myKey
```