Exercise 5.1: Configure the Deployment: ConfigMaps

Very Important

Save a copy of your `$HOME/app1/simpleapp.yaml` file, in case you would like to repeat portions of the labs, or you find your file difficult to use due to typos and whitespace issues.

```
student@cp:~$ cp $HOME/app1/simpleapp.yaml $HOME/beforeLab5.yaml
```

We will cover the use of **secrets** in the `Security` chapter lab.

Overview

In this lab we will add resources to our deployment with further configuration you may need for production.

There are three different ways a **ConfigMap** can ingest data, from a literal value, from a file, or from a directory of files.

1. Create a **ConfigMap** containing primary colors. We will create a series of files to ingest into the **ConfigMap**. First create a directory `primary` and populate it with four files. Then we create a file in our home directory with our favorite color.

```
student@cp:~/app1$ cd

student@cp:~$ mkdir primary
student@cp:~$ echo c > primary/cyan
student@cp:~$ echo m > primary/magenta
student@cp:~$ echo y > primary/yellow
student@cp:~$ echo k > primary/black
student@cp:~$ echo "known as key" >> primary/black
student@cp:~$ echo blue > favorite
```

2. Generate a **configMap** using each of the three methods.

```
student@cp:~$ kubectl create configmap colors \
  --from-literal=text=black \
  --from-file=./favorite \
  --from-file=./primary/
```

```
configmap/colors created
```

3. View the newly created **configMap**. Note the way the ingested data is presented.

```
student@cp:~$ kubectl get configmap colors
```

```
NAME     DATA     AGE
colors   6        11s
```

```
student@cp:~$ kubectl get configmap colors -o yaml
```

```
apiVersion: v1
data:
  black: |
    k
    known as key
  cyan: |
    c
  favorite: |
    blue
  magenta: |
    m
  text: black
  yellow: |
    y
kind: ConfigMap
metadata:
<output_omitted>
```

4. Update the YAML file of the application to make use of the **configMap** as an environmental parameter. Add the six lines from the `env: line` to `key:favorite`.

```
student@cp:~$ vim $HOME/app1/simpleapp.yaml
```

simpleapp.yaml

```yaml
....
    spec:
      containers:
      - image: 10.97.40.62:5000/simpleapp
        env:                                #<-- Add from here
        - name: ilike
          valueFrom:
            configMapKeyRef:
              name: colors
              key: favorite                #<-- to here
        imagePullPolicy: Always
....
```

5. Delete and re-create the deployment with the new parameters.

```
student@cp-lab-7xtx:~$ kubectl delete deployment try1
```

```
deployment.apps "try1" deleted
```

```
student@cp-lab-7xtx:~$ kubectl create -f $HOME/app1/simpleapp.yaml
```

```
deployment.apps/try1 created
```

6. Even though the `try1` pod is not in a fully ready state, it is running and useful. Use **kubectl exec** to view a variable's value. View the pod state then verify you can see the `ilike` value within the `simpleapp` container. Note that the use of double dash (- -) tells the shell to pass the following as standard in.

```
student@cp:~$ kubectl get pod
```

```
<output_omitted>
```

```
student@cp:~$ kubectl exec -c simpleapp -it try1-5db9bc6f85-whxbf \
    -- /bin/bash -c 'echo $ilike'
```

```
blue
```

7. Edit the YAML file again, this time adding the another method of using a **configMap**. Edit the file to add three lines. `envFrom` should be indented the same amount as `env` earlier in the file, and `configMapRef` should be indented the same as `configMapKeyRef`.

```
student@cp:˜$ vim $HOME/app1/simpleapp.yaml
```

simpleapp.yaml

```yaml
....
           configMapKeyRef:
              name: colors
              key: favorite
        envFrom:            #<-- Add this and the following two lines
        - configMapRef:
            name: colors
        imagePullPolicy: Always
....
```

8. Again delete and recreate the deployment. Check the pods restart.

```
student@cp:˜$ kubectl delete deployment try1
```

```
deployment.apps "try1" deleted
```

```
student@cp:˜$ kubectl create -f $HOME/app1/simpleapp.yaml
```

```
deployment.apps/try1 created
```

```
student@cp:˜$ kubectl get pods
```

```
NAME                        READY  STATUS       RESTARTS   AGE
nginx-6b58d9cdfd-9fnl4      1/1    Running      1          23h
registry-795c6c8b8f-hl5w    1/1    Running      2          23h
try1-d4fbf76fd-46pkb        1/2    Running      0          40s
try1-d4fbf76fd-9kw24        1/2    Running      0          39s
try1-d4fbf76fd-bx9j9        1/2    Running      0          39s
try1-d4fbf76fd-jw8g7        1/2    Running      0          40s
try1-d4fbf76fd-lppl5        1/2    Running      0          39s
try1-d4fbf76fd-xtfd4        1/2    Running      0          40s
```

9. View the settings inside the `try1` container of a pod. The following output is truncated in a few places. Omit the container name to observe the behavior. Also execute a command to see all environmental variables instead of logging into the container first.

```
student@cp:˜$ kubectl exec -it try1-d4fbf76fd-46pkb -- /bin/bash -c 'env'
```

```
Defaulting container name to simpleapp.
Use 'kubectl describe pod/try1-d4fbf76fd-46pkb -n default' to see all of the containers in this
↪  pod.
REGISTRY_PORT_5000_TCP_ADDR=10.97.40.62
HOSTNAME=try1-d4fbf76fd-46pkb
TERM=xterm
yellow=y
<output_omitted>
REGISTRY_SERVICE_HOST=10.97.40.62
KUBERNETES_SERVICE_PORT=443
REGISTRY_PORT_5000_TCP=tcp://10.97.40.62:5000
KUBERNETES_SERVICE_HOST=10.96.0.1
text=black
REGISTRY_SERVICE_PORT_5000=5000
<output_omitted>
black=k
known as key
```

```
<output_omitted>
ilike=blue
<output_omitted>
magenta=m

cyan=c
<output_omitted>
```

10. For greater flexibility and scalability **ConfigMaps** can be created from a YAML file, then deployed and redeployed as necessary. Once ingested into the cluster the data can be retrieved in the same manner as any other object. Create another **configMap**, this time from a YAML file.

```
student@cp:˜$ vim car-map.yaml
```

car-map.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: fast-car
  namespace: default
data:
  car.make: Ford
  car.model: Mustang
  car.trim: Shelby
```

```
student@cp:˜$ kubectl create -f car-map.yaml
```

```
configmap/fast-car created
```

11. View the ingested data, note that the output is just as in file created.

```
student@cp:˜$ kubectl get configmap fast-car -o yaml
```

```
apiVersion: v1
data:
  car.make: Ford
  car.model: Mustang
  car.trim: Shelby
kind: ConfigMap
metadata:
<output_omitted>
```

12. Add the **configMap** settings to the `simpleapp.yaml` file as a volume. Both containers in the try1 deployment can access to the same volume, using `volumeMounts` statements. Remember that the volume stanza is of equal depth to the containers stanza, and should come after the containers have been declared, the example below has the volume added just before the `status`: output..

```
student@cp:˜$ vim $HOME/app1/simpleapp.yaml
```

simpleapp.yaml

```yaml
....
    spec:
      containers:
```

```yaml
      - image: 10.97.40.62:5000/simpleapp
        volumeMounts:             #<-- Add this and following two lines
        - mountPath: /etc/cars
          name: car-vol
        env:
        - name: ilike
....
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:                    #<-- Add this and following four lines
      - name: car-vol
        configMap:
          defaultMode: 420
          name: fast-car
status:
....
```

13. Delete and recreate the deployment.

```
student@cp:~$ kubectl delete deployment try1
```

```
deployment.apps "try1" deleted
```

```
student@cp:~$ kubectl create -f $HOME/app1/simpleapp.yaml
```

```
deployment.apps/try1 created
```

14. Verify the deployment is running. Note that we still have not automated the creation of the `/tmp/healthy` file inside the container, as a result the `AVAILABLE` count remains zero until we use the **for** loop to create the file. We will remedy this in the next step.

```
student@cp:~$ kubectl get deployment
```

```
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx       1/1     1            1           1d
registry    1/1     1            1           1d
try1        0/6     6            0           39s
```

15. Our health check was the successful execution of a command. We will edit the command of the existing `readinessProbe` to check for the existence of the mounted configMap file and re-create the deployment. After a minute both containers should become available for each pod in the deployment. Be sure you edit the `simpleapp` section, not the `goproxy` section.

```
student@cp:~$ kubectl delete deployment try1
```

```
deployment.apps "try1" deleted
```

```
student@cp:~$ vim $HOME/app1/simpleapp.yaml
```

simpleapp.yaml

```yaml
....
        readinessProbe:
          exec:
            command:
            - ls                        #<-- Add/Edit this and following line.
```

```yaml
                - /etc/cars
            periodSeconds: 5
....
```

```
student@cp:˜$ kubectl create -f $HOME/app1/simpleapp.yaml
```

```
deployment.apps/try1 created
```

16. Wait about a minute and view the deployment and pods. All six replicas should be running and report that 2/2 containers are in a ready state within.

```
student@cp:˜$ kubectl get deployment
```

```
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
nginx      1/1     1            1           1d
registry   1/1     1            1           1d
try1       6/6     6            6           1m
```

```
student@cp:˜$ kubectl get pods
```

```
NAME                        READY     STATUS    RESTARTS   AGE
nginx-6b58d9cdfd-9fnl4      1/1       Running   1          1d
registry-795c6c8b8f-hl5wf   1/1       Running   2          1d
try1-7865dcb948-2dzc8       2/2       Running   0          1m
try1-7865dcb948-7fkh7       2/2       Running   0          1m
try1-7865dcb948-d85bc       2/2       Running   0          1m
try1-7865dcb948-djrcj       2/2       Running   0          1m
try1-7865dcb948-kwlv8       2/2       Running   0          1m
try1-7865dcb948-stb2n       2/2       Running   0          1m
```

17. View a file within the new volume mounted in a container. It should match the data we created inside the configMap. Because the file did not have a carriage-return it will appear prior to the following prompt.

```
student@cp:˜$ kubectl exec -c simpleapp -it try1-7865dcb948-stb2n \
    -- /bin/bash -c 'cat /etc/cars/car.trim'
```

```
Shelby student@cp:˜$
```