



## Exercise 5.1: Configure the Deployment: Secrets and ConfigMap



### Very Important

Save a copy of your `~/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@ckad-1:~$ cp ~/app1/simpleapp.yaml ~/beforeLab5.yaml
```

### 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@ckad-1:~/app1$ cd

student@ckad-1:~$ mkdir primary

student@ckad-1:~$ echo c > primary/cyan
student@ckad-1:~$ echo m > primary/magenta
student@ckad-1:~$ echo y > primary/yellow
student@ckad-1:~$ echo k > primary/black
student@ckad-1:~$ echo "known as key" >> primary/black
student@ckad-1:~$ echo blue > favorite
```

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

```
student@ckad-1:~$ 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@ckad-1:~$ kubectl get configmap colors

NAME      DATA      AGE
colors    6          11s

student@ckad-1:~$ 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:
  creationTimestamp: 2018-04-05T19:49:59Z
  name: colors
  namespace: default
  resourceVersion: "13491"
  selfLink: /api/v1/namespaces/default/configmaps/colors
  uid: 86457ce3-390a-11e8-ba73-42010a800003

```

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@ckad-1:~$ vim ~/app1/simpleapp.yaml
```

```

1 ....
2 spec:
3   containers:
4     - image: 10.105.119.236:5000/simpleapp:latest
5       env: # Add from here
6     - name: ilike
7       valueFrom:
8         configMapKeyRef:
9           name: colors
10          key: favorite # To here
11       imagePullPolicy: Always
12 ....

```

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

```
student@ckad-1-lab-7tx:~$ kubectl delete deployment try1
deployment.apps "try1" deleted
```

```
student@ckad-1-lab-7tx:~$ kubectl create -f ~/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@ckad-1:~$ kubectl get po
<output_omitted>
```

```
student@ckad-1:~$ 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@ckad-1:~$ vim ~/app1/simpleapp.yaml
```



## simpleapp.yaml

```

1  ....
2      configMapKeyRef:
3          name: colors
4          key: favorite
5  envFrom: #<-- Add this and the following two lines
6  - configMapRef:
7      name: colors
8  imagePullPolicy: Always
9  ....

```

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

```

student@ckad-1:~$ kubectl delete deployment try1
deployment.apps "try1" deleted

student@ckad-1:~$ kubectl create -f ~/app1/simpleapp.yaml
deployment.apps/try1 created

```

```
student@ckad-1:~$ 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@ckad-1:~$ 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=10.105.119.236
HOSTNAME=try1-d4fbf76fd-46pkb
TERM=xterm
yellow=y
<output_omitted>
REGISTRY_SERVICE_HOST=10.105.119.236
KUBERNETES_SERVICE_PORT=443
REGISTRY_PORT_5000_TCP=tcp://10.105.119.236:5000
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@ckad-1:~$ vim car-map.yaml
```

YAML

car-map.yaml

```
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: fast-car
5    namespace: default
6  data:
7    car.make: Ford
8    car.model: Mustang
9    car.trim: Shelby
```

```
student@ckad-1:~$ 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@ckad-1:~$ kubectl get configmap fast-car -o yaml
apiVersion: v1
data:
  car.make: Ford
  car.model: Mustang
  car.trim: Shelby
kind: ConfigMap
metadata:
  creationTimestamp: 2018-07-26T16:36:32Z
  name: fast-car
  namespace: default
  resourceVersion: "105700"
  selfLink: /api/v1/namespaces/default/configmaps/fast-car
  uid: aa19f8f3-39b8-11e8-ba73-42010a800003
```

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@ckad-1:~$ vim ~/app1/simpleapp.yaml
```

YAML

simpleapp.yaml

```
1  ....
2    spec:
3      containers:
4        - image: 10.105.119.236:5000/simpleapp:latest
5          volumeMounts:                                #<-- Add this and following two lines
6            - mountPath: /etc/cars
7              name: car-vol
8          env:
9            - name: ilike
10  ....
```



```

11     securityContext: {}
12     terminationGracePeriodSeconds: 30
13     volumes:                                #<-- Add this and following four lines
14     - name: car-vol
15       configMap:
16         defaultMode: 420
17         name: fast-car
18   status:
19     ....

```

13. Delete and recreate the deployment.

```

student@ckad-1:~$ kubectl delete deployment try1
deployment.apps "try1" deleted

student@ckad-1:~$ kubectl create -f ~/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@ckad-1:~$ 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.

```

student@ckad-1:~$ kubectl delete deployment try1
deployment.apps "try1" deleted

student@ckad-1:~$ vim ~/app1/simpleapp.yaml

```



simpleapp.yaml

```

1     ....
2     readinessProbe:
3       exec:
4         command:
5         - ls /tmp/healthy #<-- Add/Edit this and following line.
6         - /etc/cars
7       periodSeconds: 5
8     ....

```

```

student@ckad-1:~$ kubectl create -f ~/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@ckad-1:~$ 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@ckad-1:~$ 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@ckad-1:~$ kubectl exec -c simpleapp -it try1-7865dcb948-stb2n \
  -- /bin/bash -c 'cat /etc/cars/car.trim'
Shelby student@ckad-1:~$
```