Exercise 5.3: Using ConfigMaps Configure Containers

In an earlier lab we added a second container to handle logging. Now that we have learned about using `ConfigMaps` and attaching storage we will use configure our `basic` pod.

1. Review the YAML for our earlier simple pod. Recall that we added an Ambassador style logging container to the pod but had not fully configured the logging.

```
student@cp:~$ cat basic.yaml
```

```
<output_omitted>
  containers:
  - name: webcont
    image: nginx
    ports:
    - containerPort: 80
  - name: fdlogger
    image: fluentd
```

2. Let us begin by adding shared storage to each container. We will use the `hostPath` storage class to provide the `PV` and `PVC`. First we create the directory.

```
student@cp:~$ sudo mkdir /tmp/weblog
```

3. Now we create a new `PV` to use that directory for the `hostPath` storage class. We will use the `storageClassName` of manual so that only `PVC`s which use that name will bind the resource.

```
student@cp:~$ vim weblog-pv.yaml
```

weblog-pv.yaml

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: weblog-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/tmp/weblog"
```

4. Create and verify the new `PV` exists and shows an `Available` status.

```
student@cp:~$ kubectl create -f weblog-pv.yaml
```

```
persistentvolume/weblog-pv-volume created
```

```
student@cp:~$ kubectl get pv weblog-pv-volume
```

```
NAME             CAPACITY ACCESS MODES  RECLAIM POLICY
  STATUS        CLAIM    STORAGECLASS   REASON    AGE

weblog-pv-volume 100Mi     RWO           Retain
  Available              manual                   21s
```

5. Next we will create a PVC to use the PV we just created.

```
student@cp:~$ vim weblog-pvc.yaml
```

weblog-pvc.yaml

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: weblog-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

6. Create the PVC and verify it shows as Bound to the the PV we previously created.

```
student@cp:~$ kubectl create -f weblog-pvc.yaml
```

```
persistentvolumeclaim/weblog-pv-claim created
```

```
student@cp:~$ kubectl get pvc weblog-pv-claim
```

```
NAME             STATUS  VOLUME           CAPACITY ACCESS MODES
   STORAGECLASS   AGE
weblog-pv-claim  Bound   weblog-pv-volume 100Mi    RWO
   manual         79s
```

7. We are ready to add the storage to our pod. We will edit three sections. The first will declare the storage to the pod in general, then two more sections which tell each container where to make the volume available.

```
student@cp:~$ vim basic.yaml
```

basic.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: basicpod
  labels:
    type: webserver
spec:
  volumes:                            #<-- Add three lines, same depth as containers
    - name: weblog-pv-storage
      persistentVolumeClaim:
        claimName: weblog-pv-claim
  containers:
  - name: webcont
```

```yaml
      image: nginx
      ports:
      - containerPort: 80
      volumeMounts:                    #<-- Add three lines, same depth as ports
        - mountPath: "/var/log/nginx/"
          name: weblog-pv-storage       # Must match volume name above
    - name: fdlogger
      image: fluentd
      volumeMounts:                    #<-- Add three lines, same depth as image:
        - mountPath: "/var/log"
          name: weblog-pv-storage       # Must match volume name above
```

8. At this point we can create the pod again. When we create a shell we will find that the `access.log` for **nginx** is no longer a symbolic link pointing to `stdout` it is a writable, zero length file. Leave a **tailf** of the log file running.

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

```
pod/basicpod created
```

```
student@cp:˜$ kubectl exec -c webcont -it basicpod -- /bin/bash
```

On Container

```
root@basicpod:/#  ls -l /var/log/nginx/access.log
```

```
-rw-r--r-- 1 root root 0 Oct 18 16:12 /var/log/nginx/access.log
```

```
root@basicpod:/# tail -f /var/log/nginx/access.log
```

9. Open a second connection to your cp node. We will use the pod IP as we have not yet configured a `service` to expose the pod.

```
student@cp:˜$ kubectl get pods -o wide
```

```
NAME     READY STATUS    RESTARTS  AGE    IP                NODE
   NOMINATED NODE
basicpod 2/2   Running   0         3m26s  192.168.213.181   cp
   <none>
```

10. Use **curl** to view the welcome page of the webserver. When the command completes you should see a new entry added to the log. Right after the GET we see a 200 response indicating success. You can use **ctrl-c** and **exit** to return to the host shell prompt.

```
student@cp:˜$ curl http://192.168.213.181
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<output_omitted>
```

On Container

```
192.168.32.128 - - [18/Oct/2022:16:16:21 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.47.0" "-"
```

11. Now that we know the `webcont` container is writing to the `PV` we will configure the logger to use that directory as a source. For greater flexibility we will configure **fluentd** using a `configMap`.

    Fluentd has many options for input and output of data. We will read from a file of the `webcont` container and write to standard out of the `fdlogger` container. The details of the data settings can be found in **fluentd** documentation here: https://docs.fluentd.org/configuration/config-file-yaml

    ```
    student@cp:~$ vim weblog-configmap.yaml
    ```

    weblog-configmap.yaml

    ```yaml
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: fluentd-config
      namespace: default
    data:
      fluent.conf: |
        <source>
          @type tail
          format none
          path /var/log/access.log
          tag count.format1
        </source>

        <match *.**>
        @type stdout
        id stdout_output
        </match>
    ```

12. Create the new configMap.

    ```
    student@cp:~$ kubectl create -f weblog-configmap.yaml
    ```

    ```
    configmap/fluentd-config created
    ```

13. View the logs for both containers in the `basicpod`. You should see some startup information, but not the HTTP traffic.

    ```
    student@cp:~$ kubectl logs basicpod webcont
    ```

    ```
    /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
    /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
    /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
    10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
    10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
    /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
    /docker-entrypoint.sh: Configuration complete; ready for start up
    ```

    ```
    student@cp:~$  kubectl logs basicpod fdlogger
    ```

    ```
    2020-09-02 19:32:59 +0000 [info]: reading config file path="/etc/fluentd-config/fluentd.conf"
    2020-09-02 19:32:59 +0000 [info]: starting fluentd-0.12.29
    2020-09-02 19:32:59 +0000 [info]: gem 'fluent-mixin-config-placeholders' version '0.4.0'
    2020-09-02 19:32:59 +0000 [info]: gem 'fluent-mixin-plaintextformatter' version '0.2.6'

    <output_omitted>

      <source>
        @type tail
        format none
        path /var/log/access.log
    ```

```
<output_omitted>
```

14. Now we will edit the pod yaml file so that the **fluentd** container will mount the configmap as a volume and reference the variables inside the config file. You will add three areas, the volume declaration to the pod, the `env` parameter and the mounting of the volume to the fluentd container

```
student@cp:~$ vim basic.yaml
```

basic.yaml

```yaml
....
  volumes:
    - name: weblog-pv-storage
      persistentVolumeClaim:
        claimName: weblog-pv-claim
    - name: log-config                        #<-- This and two lines following
      configMap:
        name: fluentd-config                  # Must match existing configMap
....
    image: fluentd
    env:                                      #<-- This and two lines following
    - name: FLUENTD_OPT
      value: -c /fluentd/etc/fluent.conf
....
    volumeMounts:
      - mountPath: "/var/log"
        name: weblog-pv-storage
      - name: log-config                      #<-- This and next line
        mountPath: "/fluentd/etc"
```

15. At this point we can delete and re-create the pod, which would cause the configmap to be used by the new pod, among other changes.

```
student@cp:~$ kubectl delete pod basicpod
```

```
pod "basicpod" deleted
```

```
student@cp:~$ kubectl create -f basic.yaml
```

```
pod/basicpod created
```

```
student@cp:~$ kubectl get pod basicpod -o wide
```

```
NAME       READY   STATUS    RESTARTS   AGE   IP                NODE     NOMINATED....
basicpod   2/2     Running   0          8s    192.168.171.122   worker   <none>    ....
```

16. Use **curl** a few times to look at the default page served by `basicpod`

```
student@cp:~$ curl http://192.168.171.122
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
<output_omitted>
```

17. Look at the logs for both containers. In addition to the standard startup information, you should also see the HTTP requests from the curl commands you just used at the end of the `fdlogger` output.

```
student@cp:˜$ kubectl logs basicpod webcont
```

```
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

```
student@cp:˜$  kubectl logs basicpod fdlogger
```

```
2020-09-02 19:32:59 +0000 [info]: reading config file path="/etc/fluentd-config/fluentd.conf"
2020-09-02 19:32:59 +0000 [info]: starting fluentd-0.12.29
2020-09-02 19:32:59 +0000 [info]: gem 'fluent-mixin-config-placeholders' version '0.4.0'
2020-09-02 19:32:59 +0000 [info]: gem 'fluent-mixin-plaintextformatter' version '0.2.6'

<output_omitted>

  <source>
    @type tail
    format none
    path /var/log/access.log

<output_omitted>

2020-09-02 19:47:38 +0000 count.format1: {"message":"192.168.219.64 - - [02/Sep/2020:19:47:38
↪  +0000] \"GET / HTTP/1.1\" 200 612 \"-\" \"curl/7.58.0\" \"-\""}
2020-09-02 19:47:41 +0000 count.format1: {"message":"192.168.219.64 - - [02/Sep/2020:19:47:41
↪  +0000] \"GET / HTTP/1.1\" 200 612 \"-\" \"curl/7.58.0\" \"-\""}
2020-09-02 19:47:47 +0000 count.format1: {"message":"192.168.219.64 - - [02/Sep/2020:19:47:47
↪  +0000] \"GET / HTTP/1.1\" 200 612 \"-\" \"curl/7.58.0\" \"-\""}
```