Exercise 5.3: Using ConfigMaps Configure Ambassador Containers

In an earlier lab we added a second `Ambassador` 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@ckad-1:~$ cat basic.yaml
<output_omitted>
  containers:
  - name: webcont
    image: nginx
    ports:
    - containerPort: 80
  - name: fdlogger
    image: fluent/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@ckad-1:~$ 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 `PVCs` which use that name will bind the resource.

```
student@ckad-1:~$ vim weblog-pv.yaml
```

weblog-pv.yaml

```
1  kind: PersistentVolume
2  apiVersion: v1
3  metadata:
4    name: weblog-pv-volume
5    labels:
6      type: local
7  spec:
8    storageClassName: manual
9    capacity:
10     storage: 100Mi
11   accessModes:
12     - ReadWriteOnce
13   hostPath:
14     path: "/tmp/weblog"
```

4. Create and verify the new `PV` exists.

```
student@ckad-1:~$ kubectl create -f weblog-pv.yaml
persistentvolume/weblog-pv-volume created

student@ckad-1:~$ 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@ckad-1:~$ vim weblog-pvc.yaml
```

weblog-pvc.yaml

```
1  kind: PersistentVolumeClaim
2  apiVersion: v1
3  metadata:
4    name: weblog-pv-claim
5  spec:
6    storageClassName: manual
7    accessModes:
8      - ReadWriteOnce
9    resources:
10     requests:
11       storage: 100Mi
```

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

```
student@ckad-1:~$ kubectl create -f weblog-pvc.yaml
persistentvolumeclaim/weblog-pv-claim created

student@ckad-1:~$ 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@ckad-1:~$ vim basic.yaml
```

basic.yaml

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: basicpod
5    labels:
6      type: webserver
7  spec:
8    volumes:                          #<-- Add three lines, same depth as containers
9      - name: weblog-pv-storage
10       persistentVolumeClaim:
11         claimName: weblog-pv-claim
12   containers:
13   - name: webcont
14     image: nginx
15     ports:
16     - containerPort: 80
```

```
17      volumeMounts:                       #<-- Add three lines, same depth as ports
18        - mountPath: "/var/log/nginx/"
19          name: weblog-pv-storage         # Must match volume name above
20    - name: fdlogger
21      image: fluent/fluentd
22      volumeMounts:                       #<-- Add three lines, same depth as image:
23        - mountPath: "/var/log"
24          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@ckad-1:~$ kubectl create -f basic.yaml
pod/basicpod created

student@ckad-1:~$ 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 node. We will use the pod IP as we have not yet configured a `service` to expose the pod.

```
student@ckad-1:~$ kubectl get pods -o wide
NAME      READY STATUS    RESTARTS  AGE     IP               NODE
   NOMINATED NODE
basicpod 2/2    Running   0         3m26s   192.168.213.181  ckad-1
   <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@ckad-1:~$ 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/2018:16:16:21 +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`. The details of the data settings can be found in **fluentd** documentation here: `https://docs.fluentd.org/v1.0/categories/config-file`

```
student@ckad-1:~$ vim weblog-configmap.yaml
```

weblog-configmap.yaml

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

    <match *.**>
      @type forward

      <server>
        name localhost
        host 127.0.0.1
      </server>
    </match>
```

12. Create the new configMap.

```
student@ckad-1:~$ kubectl create -f weblog-configmap.yaml
configmap/fluentd-config created
```

13. 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@ckad-1:~$ vim basic.yaml
```

basic.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: fluent/fluentd
    env:                                  #<-- This and two lines following
    - name: FLUENTD_ARGS
      value: -c /etc/fluentd-config/fluentd.conf
....
    volumeMounts:
      - mountPath: "/var/log"
        name: weblog-pv-storage
      - name: log-config                  #<-- This and next line
        mountPath: "/etc/fluentd-config"
```

14. At this point we can delete and re-create the pod. If we had a listening agent running on `localhost`, where the we messages are forwarded as declared in the configMap, we would see access messages.

```
student@ckad-1:~$ kubectl delete pod basicpod
pod "basicpod" deleted

student@ckad-1:~$ kubectl create -f basic.yaml
pod/basicpod created

student@ckad-1:~$ kubectl get pod basicpod
NAME       READY   STATUS    RESTARTS   AGE
basicpod   2/2     Running   0          8s
```

15. Look at the logs for both containers. You should see some output for the `fdlogger` but not for `webcont`.

```
student@ckad-1:~$ kubectl logs basicpod webcont

student@ckad-1:~$  kubectl logs basicpod fdlogger
2019-03-06 18:55:33 +0000 [info]: parsing config file is succeeded path="/fluentd/etc/fluent.conf"
2019-03-06 18:55:33 +0000 [warn]: [output_docker1] 'time_format' specified without 'time_key', will be ignored
2019-03-06 18:55:33 +0000 [warn]: [output1] 'time_format' specified without 'time_key', will be ignored
2019-03-06 18:55:33 +0000 [info]: using configuration file: <ROOT>
  <source>
    @type forward
    @id input1
<output_omitted>
```