Building Multi-Container Pods

Relevant Documentation

- The Distributed System ToolKit: Patterns for Composite Containers
- Pods - Resource Sharing and Communication
- Shared Volumes

Exam Tips

- A sidecar container performs some task that helps the main container.
- An ambassador container proxies network traffic to and/or from the main container.
- An adapter container transforms the main container's output.

Lesson Reference

Log in to the **control plane node**.

Create a Pod with a sidecar container that interacts with the main container using a shared volume.

```
vi sidecar-test.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: sidecar-test
spec:
  containers:
  - name: writer
    image: busybox:stable
    command: ['sh', '-c', 'echo "The writer wrote this!" > /output/data.txt; while true; do sleep 5;
done']
    volumeMounts:
    - name: shared
      mountPath: /output
  - name: sidecar
    image: busybox:stable
    command: ['sh', '-c', 'while true; do cat /input/data.txt; sleep 5; done']
    volumeMounts:
    - name: shared
      mountPath: /input
  volumes:
  - name: shared
    emptyDir: {}
```

```
kubectl apply -f sidecar-test.yml
```

Check the Pod status and wait for both containers to become ready.

```
kubectl get pod sidecar-test
```

Check the logs for the sidecar container. You should be able to see the data that was written by the main container, `The writer wrote this!`.

```
kubectl logs sidecar-test -c sidecar
```

Create a Pod with an ambassador container that interacts with the main container via shared network resources.

First, create a Service that our Pod can communicate with.

```
vi ambassador-test-setup.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: ambassador-test-webserver
  labels:
    app: ambassador-test
spec:
  containers:
  - name: nginx
    image: nginx:stable
    ports:
    - containerPort: 80

---

apiVersion: v1
kind: Service
metadata:
  name: ambassador-test-svc
spec:
  selector:
    app: ambassador-test
  ports:
  - protocol: TCP
    port: 8081
    targetPort: 80
```

```
kubectl apply -f ambassador-test-setup.yml
```

Create the Pod with the ambassador container.

```
vi ambassador-test.yml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: haproxy-config
data:
  haproxy.cfg: |
    frontend ambassador
      bind *:8080
      default_backend ambassador_test_svc
    backend ambassador_test_svc
      server svc ambassador-test-svc:8081

---

apiVersion: v1
kind: Pod
metadata:
  name: ambassador-test
spec:
  containers:
  - name: main
    image: radial/busyboxplus:curl
    command: ['sh', '-c', 'while true; do curl localhost:8080; sleep 5; done']
  - name: ambassador
    image: haproxy:2.4
    volumeMounts:
    - name: config
      mountPath: /usr/local/etc/haproxy/
  volumes:
  - name: config
    configMap:
      name: haproxy-config
```

```
kubectl apply -f ambassador-test.yml
```

Check the logs for the main Pod to verify that the setup is working.

```
kubectl logs ambassador-test -c main
```