Storage with PersistentVolumes

While volumes are a simple way to add storage to Kubernetes containers, PersistentVolumes provide a way to manage storage as a more abstract pool of resources. This lesson will explore PersistentVolumes and PersistentVolumeClaims, and how they can be used to provide storage for containers. This will prepare us to use PersistentVolumes to manage storage resources in the real world.

Relevant Documentation

- Persistent Volumes

Lesson Reference

Access UCP in a browser at `https://<UCP_SERVER_PUBLIC_IP>`.

We can create Kubernetes objects in UCP by navigating to **Kubernetes** > **+ Create**. Make sure to use the default namespace throughout this exercise.

Make sure to view the `default` namespace (**Kubernetes** > **Namespaces**, then click **Set Context** next to `default`).

Create a StorageClass and PersistentVolume

Create a StorageClass that supports volume expansion.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: localdisk
provisioner: kubernetes.io/no-provisioner
allowVolumeExpansion: true
```

Create a PersistentVolume.

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: my-pv
spec:
  storageClassName: localdisk
  persistentVolumeReclaimPolicy: Recycle
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /tmp/pvoutput
```

Create a PersistentVolumeClaim

Create a PersistentVolumeClaim.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  storageClassName: localdisk
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

Check the claim's status to make sure it is `Bound` to the PersistentVolume.

Create the Pod

Create a pod that uses the PersistentVolumeClaim and writes some data to it.

```
apiVersion: v1
kind: Pod
metadata:
  name: pvc-pod
spec:
  containers:
  - name: busybox
    image: busybox
    command: ['sh', '-c', 'while true; do echo "Successfully written to log." >> /output/output.log; sleep 10
    volumeMounts:
    - name: pv-storage
      mountPath: /output
  volumes:
  - name: pv-storage
    persistentVolumeClaim:
      claimName: my-pvc
```

Verify the pod can start up.

Take note of which node the pod is running on and log in to that node via SSH.

Check the output data from the pod.

```
ls /tmp/pvoutput

cat /tmp/pvoutput/output.log
```

Recycle the Persistent Storage

In the UCP interface, navigate to **Kubernetes** > **Pods**, and use the three-dot menu for the `pvc-pod` pod to delete the pod.

> **Note:** `Force Remove` is faster and is okay to use here since we are just testing and do not need the pod to shut down gracefully.

Navigate to **Kubernetes** > **Storage**, and use the three-dot menu for the `my-pvc` PersistentVolumeClaim to delete the PersistentVolumeClaim.

We should see the status for the `my-pv` PersistentVolume return to `Available`.

Create a new PersistentVolumeClaim.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-new-pvc
spec:
  storageClassName: localdisk
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

We should see both the PersistentVolume and PersistentVolumeClaim return their status to `Bound`.

Expand the PersistentVolumeClaim

Navigate to **Kubernetes** > **Storage** > **my-new-pvc**. Click the gear icon to edit the PersistentVolumeClaim.

Find the line that says `storage: 100Mi` and change it to increase the claim size:

```
spec:

  ...

  resources:
    requests:
      storage: 200Mi
```

Click **Save**.