Removing the Resources and Exploring the Effects

In this lesson, we will play around with the deployment by removing the created resources and exploring the effects of removal.

WE'LL COVER THE FOLLOWING

Deleting the Deployment

Let's delete the `jenkins` Deployment.

```
kubectl --namespace jenkins delete \
    deploy jenkins
```

The **output** shows us that the `deployment "jenkins"` was `deleted`.

Did anything happen with the PersistentVolumeClaim and the PersistentVolume?

```
kubectl --namespace jenkins get pvc

kubectl get pv
```

The combined **output** of both commands is as follows.

```
NAME    STATUS VOLUME        CAPACITY ACCESS MODES STORAGECLASS   AGE
jenkins Bound  manual-ebs-02 5Gi      RWO          manual-ebs     57s
```

```
NAME          CAPACITY ACCESS MODES RECLAIM POLICY STATUS    CLAIM           STORAGECLASS REA
manual-ebs-01 5Gi      RWO          Retain         Available jenkins/jenkins manual-ebs

manual-ebs-02 5Gi      RWO          Retain         Bound     jenkins/jenkins manual-ebs
manual-ebs-03 5Gi      RWO          Retain         Available jenkins/jenkins manual-ebs
```

Even though we removed Jenkins Deployment and, with it, the Pod that used the claim, both the PersistentVolumeClaim and PersistentVolumes are intact. The `manual-ebs-01` volume is still bound to the `jenkins` claim.

Deleting the PersistentVolumeClaim

What would happen if we remove the PersistentVolumeClaim `jenkins`?

```
kubectl --namespace jenkins \
    delete pvc jenkins
```

The **output** shows that the `persistentvolumeclaim "jenkins"` was `deleted`.

Now, let's see what happened with the PersistentVolumes.

```
kubectl get pv
```

The **output** is as follows.

```
NAME          CAPACITY ACCESS MODES RECLAIM POLICY STATUS   CLAIM           STORAGECLASS REAS
manual-ebs-01 5Gi      RWO          Retain         Available jenkins/jenkins manual-ebs
manual-ebs-02 5Gi      RWO          Retain         Released  jenkins/jenkins manual-ebs
manual-ebs-03 5Gi      RWO          Retain         Available jenkins/jenkins manual-ebs
```

This time, the `manual-ebs-2` volume is `Released`.

Reclaim Policies

This might be a good moment to explain the `Retain` policy applied to the PersistentVolumes we created.

> `ReclaimPolicy` defines what should be done with a volume after it's released from its claim.

The policy was applied the moment we deleted the PersistentVolumeClaim

that was bound to `manual-ebs-02`. When we created the PersistentVolumes, we did not specify `ReclaimPolicy`, so the volumes were assigned the default policy which is `Retain`.

Retain Type Reclaim Policy

The `Retain` reclaim policy enforces manual reclamation of the resource. When the PersistentVolumeClaim is deleted, the PersistentVolume still exists, and the volume is considered `released`. But it is not yet available for other claims because the previous claimant's data remains on the volume. In our case, that data is Jenkins state. If we'd like this PersistentVolume to become available, we'd need to delete all the data on the EBS volume.

Since we are running the cluster in AWS, it is easier to delete than to recycle resources, so we'll remove the released PersistentVolume instead of trying to clean everything we generated inside the EBS. Actually, we'll remove all the volumes since we are about to explore how we can accomplish the same effects dynamically.

Other Types of Reclaim Policy

The other two reclaim policies are `Recycle` and `Delete`. Recycle is considered deprecated so we won't waste time explaining it. The `Delete` policy requires dynamic provisioning, but we'll postpone the explanation until we explore that topic.

Deleting the Resources

Let's delete some stuff.

```
kubectl delete -f pv/pv.yml
```

The **output** is as follows.

```
persistentvolume "manual-ebs-01" deleted
persistentvolume "manual-ebs-02" deleted
persistentvolume "manual-ebs-03" deleted
```

We can see that all three PersistentVolumes were deleted. However, only Kubernetes resources were removed. We still need to manually delete the EBS

volumes.

If you go to your AWS console, you'll see that all three EBS volumes are now in the `available` state and waiting to be mounted. We'll delete them all.

```
aws ec2 delete-volume \
    --volume-id $VOLUME_ID_1

aws ec2 delete-volume \
    --volume-id $VOLUME_ID_2

aws ec2 delete-volume \
    --volume-id $VOLUME_ID_3
```

We are finished with our tour around manual creation of persistent volumes. If we'd use this approach to volume management, the cluster administrator would need to ensure that there is always an extra number of available volumes that can be used by new claims. It is tedious work that often results in having more volumes than we need. On the other hand, if we don't have a sufficient number of available (unused) volumes, we're risking that someone will create a claim that will not find a suitable volume to mount.

Manual volume management is sometimes unavoidable, especially if you chose to use on-prem infrastructure combined with NFS.

AWS is all about dynamic resource provisioning, in the next lesson, we'll look into dynamic provision of persistent volumes.