Claiming Persistent Volumes

In this lesson, we will learn why and how to claim a persistent volume.

WE'LL COVER THE FOLLOWING

Usage of Persistent Volumes

Kubernetes persistent volumes are useless if no one uses them. They exist only as objects with relation to, in our case, specific EBS volumes. They are waiting for someone to claim them through the PersistentVolumeClaim resource.

Just like Pods which can request specific resources like memory and CPU, PersistentVolumeClaims can request particular sizes and access modes. Both are, in a way, consuming resources, even though of different types. Just as Pods should not specify on which node they should run, PersistentVolumeClaims cannot define which volume they should mount. Instead, Kubernetes scheduler will assign them a volume depending on the claimed resources.

How to Claim Persistent Volumes?

We'll use `pv/pvc.yml` to explore how we could claim a persistent volume.

```
cat pv/pvc.yml
```

The **output** is as follows.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: jenkins
  namespace: jenkins
spec:
  storageClassName: manual-ebs
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

The YAML file defines a `PersistentVolumeClaim` with the storage class name `manual-ebs`. That is the same class as the persistent volumes `manual-ebs-*` we created earlier. The access mode and the storage request are also matching what we defined for the persistent volume.

Please note that we are not specifying which volume we'd like to use. Instead, this claim specifies a set of attributes (`storageClassName`, `accessModes`, and `storage`). Any of the volumes in the system that match those specifications might be claimed by the `PersistentVolumeClaim` named `jenkins`.

Bear in mind that `resources` do not have to be the exact match. Any volume that has the same or bigger amount of storage is considered a match. A claim for `1Gi` can be translated to *at least* `1Gi`. In our case, a claim for `1Gi` matches all three persistent volumes since they are set to `5Gi`.

Creating the Claim

Now that we explored the definition of the claim, we can proceed, and create it.

```
kubectl create -f pv/pvc.yml \
    --save-config --record
```

The **output** indicates that the `persistentvolumeclaim "jenkins"` was `created`.

Verification

Let's list the claims and see what we got.

```
kubectl --namespace jenkins \
    get pvc
```

The **output** is as follows.

```
NAME    STATUS VOLUME        CAPACITY ACCESS MODES STORAGECLASS AGE
jenkins Bound  manual-ebs-02 5Gi      RWO          manual-ebs   17s
```

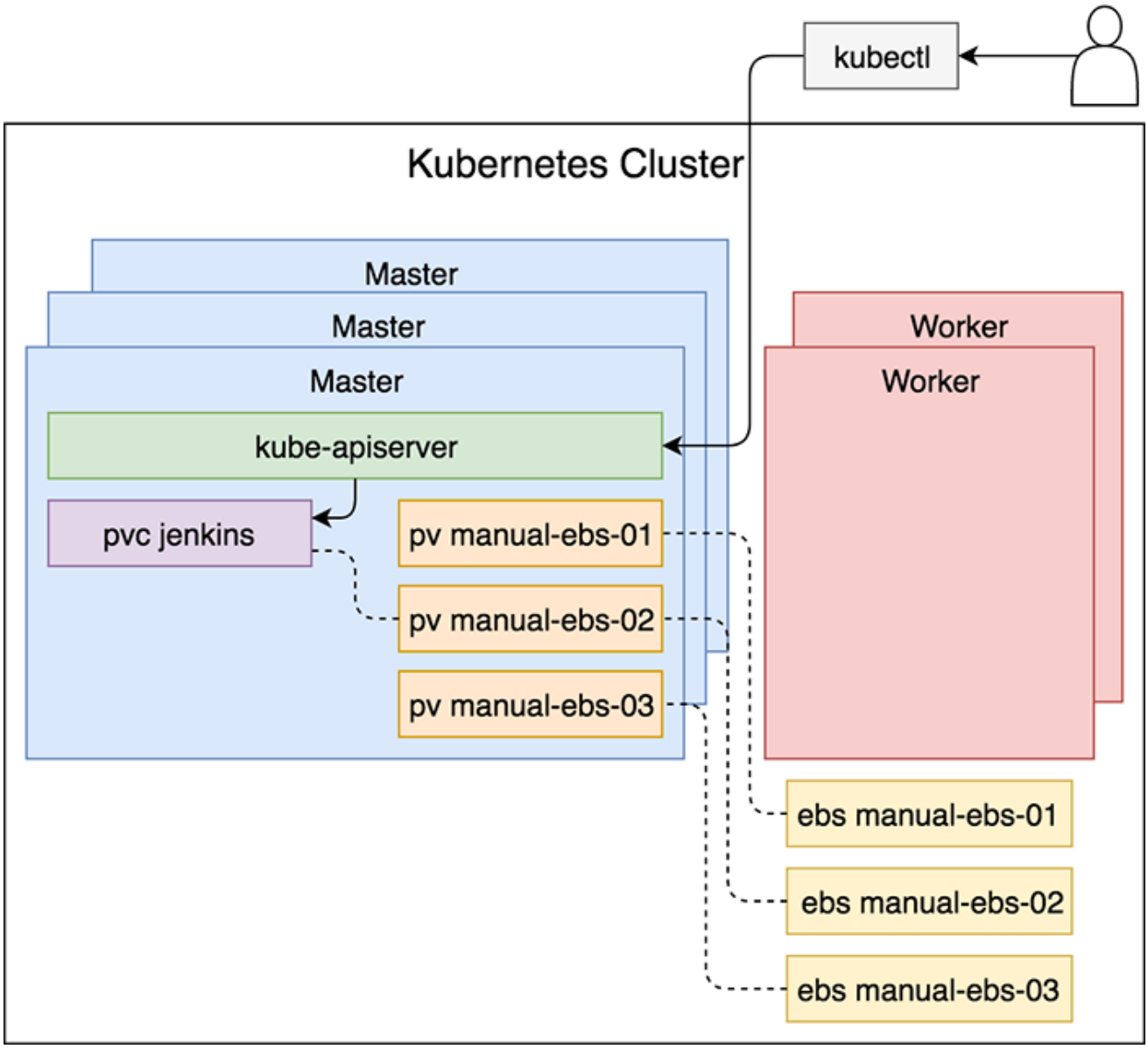
We see from the output that the status of the claim is `bound`. That means that the claim found a matching persistent volume and bounded it. We can confirm that by listing the volumes.

```
kubectl get pv
```

The **output** is as follows.

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

We can see that one of the volumes (`manual-ebs-02`) changed the status from `available` to `bound`. That is the volume bound to the claim we created a moment ago. We can see that the claim comes from `jenkins` Namespace and `jenkins` PersistentVolumeClaim.



Creation of a Persistent Volume Claim

> 📝 Please note that if a PersistentVolumeClaim cannot find a matching volume, it will remain unbound indefinitely, unless we add a new PersistentVolume with the matching specifications.

We still haven't accomplished our goal. The fact that we claimed a volume does not mean that anyone uses it. On the other hand, our Jenkins needs to persist its state.

In the next lesson, we'll join our PersistentVolumeClaim with a Jenkins container.