Creating Deployment for Attaching Claimed Volumes to Pods

In this lesson, we will create a Jenkins deployment for attaching claimed volumes to pods and look into the sequence of associated events.

WE'LL COVER THE FOLLOWING

In the previous lesson, we looked into claiming the persistent volumes. The next step is to attach these claimed volumes to pods.

Looking into the Definition

Let’s look into a Jenkins definition.

```
cat pv/jenkins-pv.yml
```

The relevant parts of the **output** as as follows.

```
...
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  namespace: jenkins
spec:
  ...
  template:
    ...
    spec:
      containers:
      - name: jenkins
```

```
        ...
        volumeMounts:
        - name: jenkins-home
          mountPath: /var/jenkins_home
        ...
      volumes:
      - name: jenkins-home
        persistentVolumeClaim:
          claimName: jenkins
      ...
```

You'll notice that, this time, we added a new volume `jenkins-home`, which references the *PersistentVolumeClaim* called `jenkins`. From the container's perspective, the claim is a volume.

Deploying Resources

Let's deploy Jenkins resources and confirm that everything works as expected.

```
kubectl apply \
    -f pv/jenkins-pv.yml \
    --record
```

The **output** is as follows.

```
namespace "jenkins" configured
ingress "jenkins" configured
service "jenkins" configured
deployment "jenkins" configured
```

Verification

We'll wait until the Deployment rolls out before proceeding with a test that will confirm whether Jenkins state is now persisted.

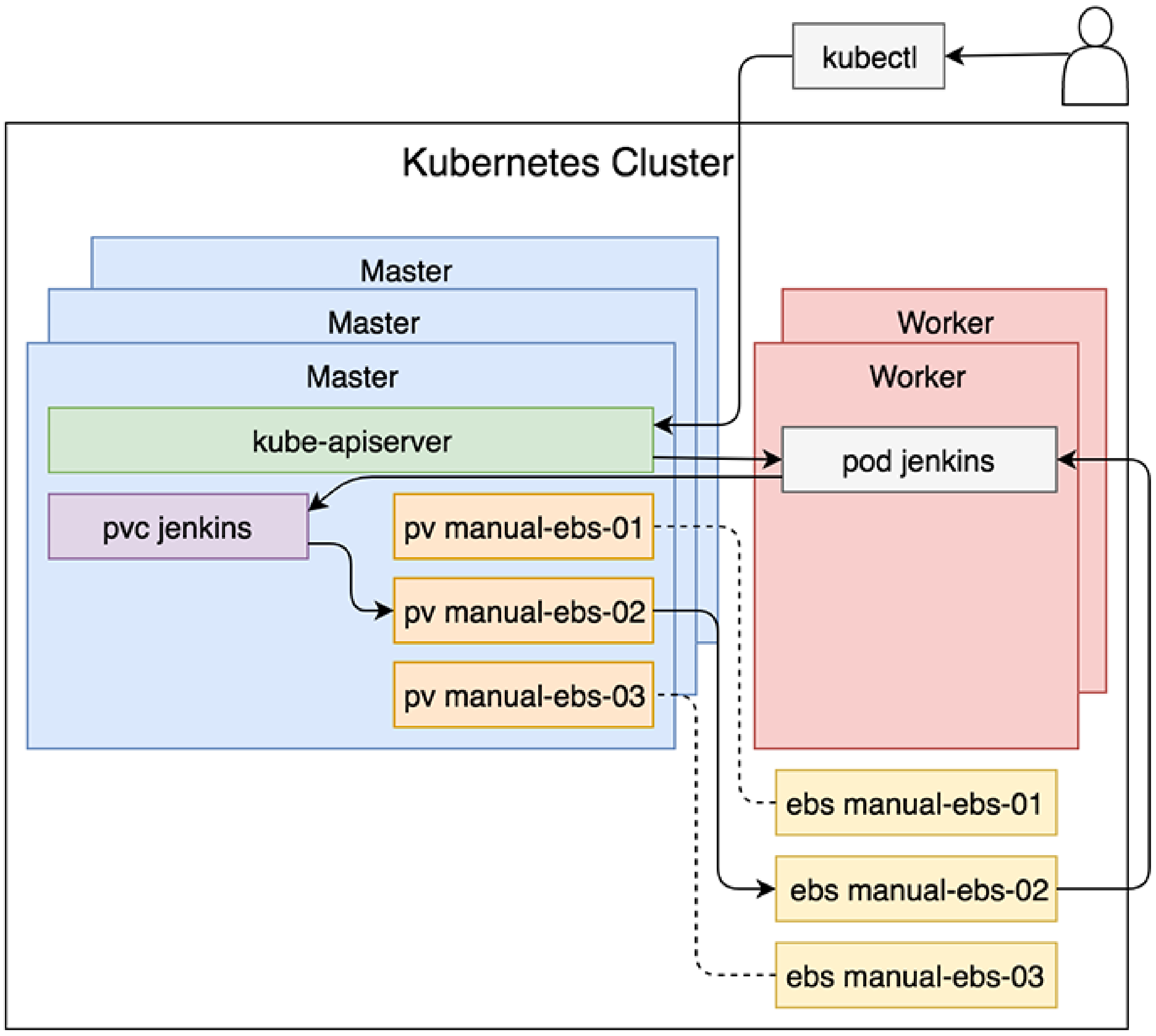
```
kubectl --namespace jenkins \
    rollout status \
    deployment jenkins
```

Once the rollout is finished, we'll see a message stating that the `deployment "jenkins"` was `successfully rolled out`.

We sent a request to the Kubernetes API to create a Deployment. As a result, we got a ReplicaSet that, in turn, created the `jenkins` Pod. It mounted the PersistentVolumeClaim, which is bound to the PersistenceVolume, that is tied to the EBS volume. As a result, the EBS volume was mounted to the `jenkins` container running in a Pod.

The Sequential Break-down

A simplified version of the sequence of events is depicted in the below illustration.



The sequence of events initiated with a request to create a Jenkins Pod with the PersistentVolumeClaim

1. We executed `kubectl` command.
2. `kubectl` sent a request to `kube-apiserver` to create the resources defined in `pv/jenkins-pv.yml`.
3. Among others, the `jenkins` Pod was created in one of the worker nodes.
4. Since `jenkins` container in the Pod has a PersistentVolumeClaim, it mounted it as a logical volume.

5. The PersistentVolumeClaim was already bound to one of the PersistentVolumes.
6. The PersistentVolume is associated with one of the EBS volumes.
7. The EBS volume was mounted as a physical volume to the `jenkins` Pod.

In the next lesson, we will verify the persistence of state and explore different types of failures that can occur.