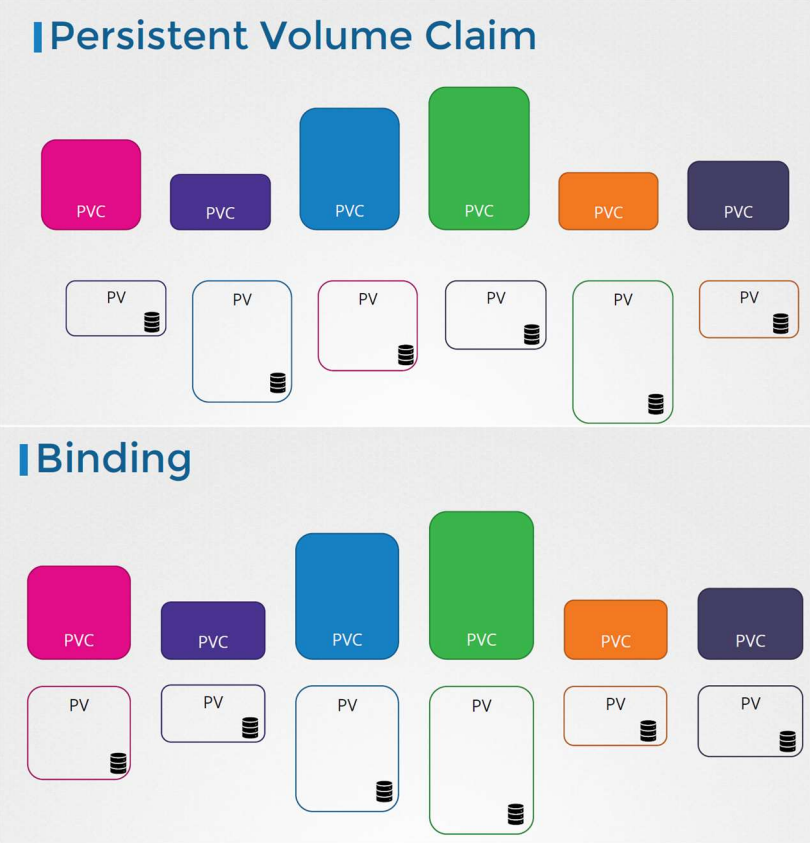
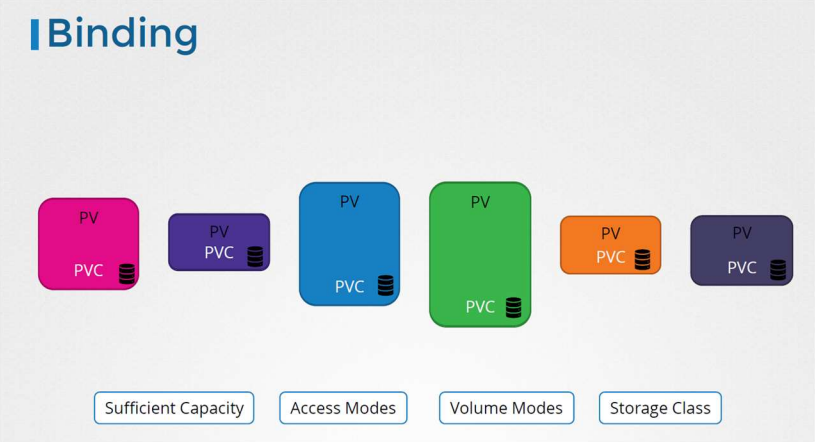
Persistent volume claims

In the previous lecture, we created a Persistent Volume. Now, we will try to create a Persistent Volume Claim to make the storage available to a note. Persistent Volumes and Persistent Volume Claims are two separate objects in the Kubernetes name space.

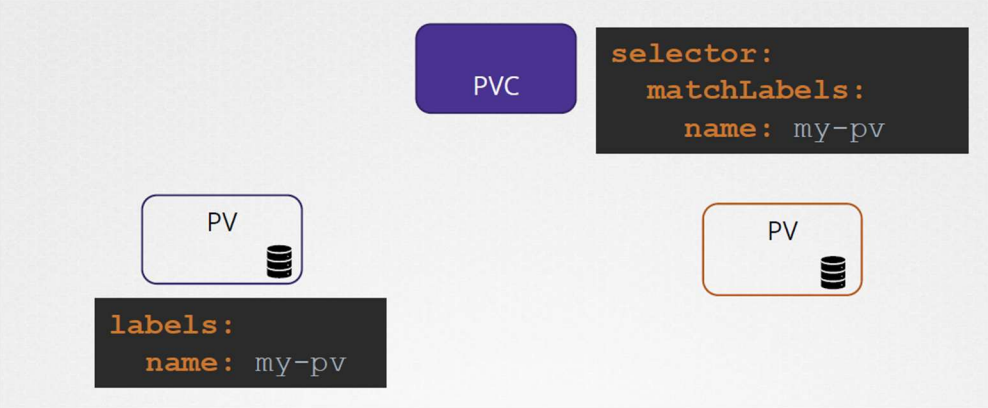
An administrator creates a set of persistent volumes, and a user creates persistent volume claims to use the storage. Once the persistent volume claims are created, Kubernetes binds the persistent volumes to claims based on the request and properties set on the volume.



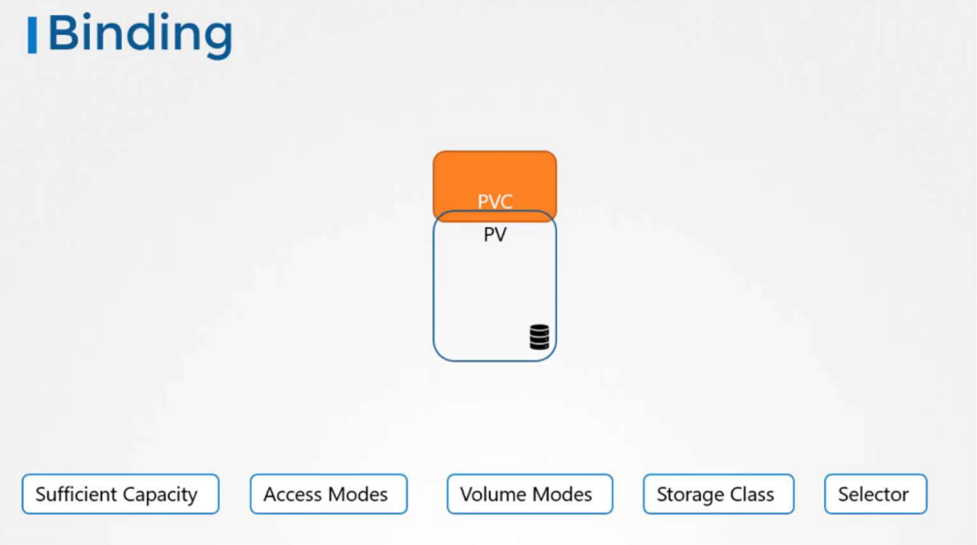
Every Persistent Volume Claim is bound to a single Persistent Volume. During the binding process, Kubernetes tries to find a Persistent Volume that has sufficient capacity, as requested by the claim. And any other request properties such as, access modes, volume modes, storage class, et cetera.



However, if there are multiple possible matches for a single claim, and you would like to specifically use a particular volume, you could still use labels and selectors to bind to the right volumes.



Finally, note that a smaller claim may get bound to a larger volume if all the other criteria matches, and there are no better options.



There is a one to one relationship between claims and volumes, so no other claims can utilize the remaining capacity in the volume. If there are no volumes available, the persistent volume claim will remain in a pending state until newer volumes are made available to the cluster. Once newer volumes are available, the claim would automatically be bound to the newly available volume.

Let us now create a persistent volume claim. We start with a blank template. Set the API version to V1 and kind to **PersistentVolumeClaim**. We will name it, "My Claim", under specification. Set the access modes to **ReadWriteOnce**. And set resources to request a storage of 500 megabytes. Create the claim using cube control, create command. To view the created claim, run the cube control, get persistent volume claim command.

We see the claim in a pending state. When the claim is created, Kubernetes looks at the volume created previously. The access modes match. The capacity requested is 500 megabytes, but the volume is configured with 1GB of storage.Since there are no other volumes available, the Persistent Volume Claim is bound to the Persistent Volume. When we run the get Persistantvolumeclaim command, again, we see the claim is bound to the Persistent Volume we created.

pvc-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
```

pv-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

```
kubectl get persistentvolumeclaim
NAME        STATUS   VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
myclaim     Bound    pv-vol1   1Gi        RWO                           43m
```

To delete a PVC, run the cube control, delete persistent volume claim command. But what happens to the underlying persistent volume when the claim is deleted? You can choose what is to happen to the volume. By default, it is set to **retain**. Meaning the persistent volume will remain until it is manually deleted by the administrator. It is not available for reuse by any other claims.

```yaml
persistentVolumeReclaimPolicy: Retain
```

Or, it can be deleted automatically. This way, as soon as the claim is deleted, the volume will be deleted as well. Thus, freeing up storage on the end storage device.

```yaml
persistentVolumeReclaimPolicy: Delete
```

Or, a third option is to recycle. In this case, the data in the data volume will be scrubbed before making it available to other claims.

```yaml
persistentVolumeReclaimPolicy: Recycle
```

Once you create a PVC use it in a POD definition file by specifying the PVC Claim name under persistentVolumeClaim section in the volumes section like this:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

The same is true for ReplicaSets or Deployments. Add this to the pod template section of a Deployment on ReplicaSet.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mypod
spec:
  selector:
    matchLabels:
      app: mypod
  replicas: 1
  template:
    spec:
      containers:
      - name: myfrontend
        image: nginx
        volumeMounts:
        - mountPath: "/var/www/html"
          name: mypd
      volumes:
      - name: mypd
        persistentVolumeClaim:
          claimName: myclaim
```