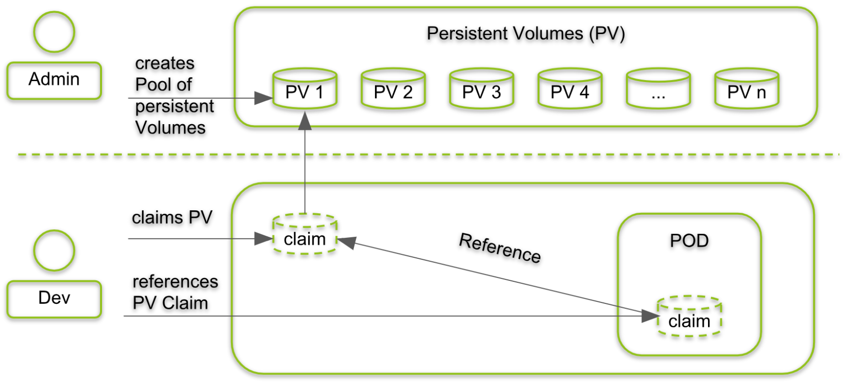
Kubernetes: Persistent Volume

Created By: Ashoka Ganta, Devanshu Sharma, Hemanth Kumar Singh, Molshree Singhal, Shushovan Chakraborty

1. Persistent Volume

A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using storage classes. It is a resource in the cluster just like a node is a cluster resource.



Create a directory on the worker node

```
sudo mkdir /mnt/data
```

Create an index.html file in that directory

```
sudo sh -c "echo 'Hello from Kubernetes storage' > /mnt/data/index.html"
```

Test the index.html file

```
cat /mnt/data/index.html
```

Create a persistent volume

Kubernetes supports hostPath for development and testing on a single-node cluster. A hostPath PersistentVolume uses a file or directory on the Node to emulate network-attached storage.

Create a yaml file

```
apiVersion: v1 #declares the version of the file
kind: PersistentVolume #declares the kind of configuration file
metadata:
  name: task-pv-volume #sets the name of the persistent volume
  labels:
    type: local #sets the label for the persistent volume
spec:
  storageClassName: manual #sets the name of Storage Class which will be used to map the PVC
  capacity:
    storage: 10Gi #sets the size to 10Gigs
  accessModes:
    - ReadWriteOnce #defines how pods can interact with the volume
  hostPath:
    path: "/mnt/data" #the location for the persistent volume
```

The configuration file specifies that the volume is at /mnt/data on the cluster's Node. The configuration also specifies a size of 10 gibibytes and an access mode of `ReadWriteOnce`, which means the volume can be mounted as read-write by a single Node.

Create PersistentVolume

```
kubectl apply -f pv.yaml
```

View information about the PersistentVolume

```
kubectl get pv task-pv-volume
```

The output shows that the PersistentVolume has a `STATUS` of `Available`. This means it has not yet been bound to a PersistentVolumeClaim.

```
NAME             CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS      CLAIM     STORAGECLASS   REASON    AGE
task-pv-volume   10Gi       RWO           Retain          Available             manual                   4s
```

2. Persistent Volume Claim

After we create a persistentVolume(PV) we create the PersistenceVolumeClaim.

A PersistenceVolumeClaim(PVC) is a request for storage.

Like a pod can consume node resources, a PVC consumes PV resources. Pods request for a specific level of resources for eg CPU and Memory and claims request for specific size and access modes for eg ReadWriteOnce, ReadOnlyMany, or ReadWriteMany.

Create a PersistenceVolumeClaim

create a yaml file

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Create the PersistentVolumeClaim

```
kubectl apply -f pv-claim.yaml
```

View the PersistenceVolume again

```
kubectl get pv task-pv-volume
```

The output shows a STATUS of BOUND

```
NAME            CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS    CLAIM                    STORAGECLASS   REASON    AGE
task-pv-volume  10Gi       RWO           Retain          Bound     default/task-pv-claim    manual                   2m
```

View the PersistentVolumeClaim

```
kubectl get pvc task-pv-claim
```

The output shows PersistentVolumeClaim is bound to your PersistentVolume

```
NAME            STATUS    VOLUME           CAPACITY   ACCESSMODES   STORAGECLASS   AGE
task-pv-claim   Bound     task-pv-volume   10Gi       RWO           manual         30s
```

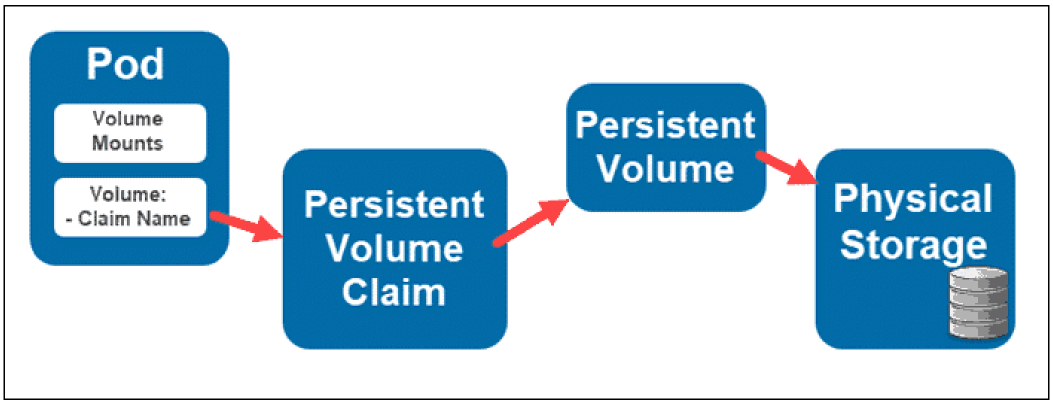
3. Pod Creation from PVC

After creating the Persistent Volume Claim, we have to use it inside of a pod. For this, we will create a new pod. To create a pod, we will create a pod.yaml file.

```
apiVersion: v1 #declares the version of api
kind: Pod  #sets the type of our configuration to Pod
metadata:
  name: task-pv-pod  #sets the name of pod
spec:
  volumes:
    - name: task-pv-storage   #binds the pod with the previously created PVC
      persistentVolumeClaim:
        claimName: task-pv-claim
  containers:
    - name: task-pv-container #sets the name for the container
      image: nginx #the name of the image that should be used
      ports:
        - containerPort: 80
          name: "http-server" #server name and its ports.
      volumeMounts:
        - mountPath: "/usr/share/nginx/html" #mountpath as per the image configuration file
          name: task-pv-storage    #name of the volume
```

From the comments, we can see which part of the YAML file serves what purpose.

💡 Note: We do not mention any Persistent Volume here, this is because from the pod's perspective, the PVC is a volume.



It is time to run the pod. We can run it using the following command.

```
kubectl apply -f pod.yaml
```

You will receive a success message. To check whether the pod is running, we can execute the command below.

```
kubectl get pod task-pv-pod
```

A successfully running pod will give the following output.

```
NAME          READY   STATUS    RESTARTS   AGE
task-pv-pod   1/1     Running   0          143m
```

Now, if we want to check whether the pod is making use of the persistent volume, refer to the next section.

4. Verification

After the execution of kubectl apply -f pod.yaml command we need to expose the pod.

This can be done by using the below command

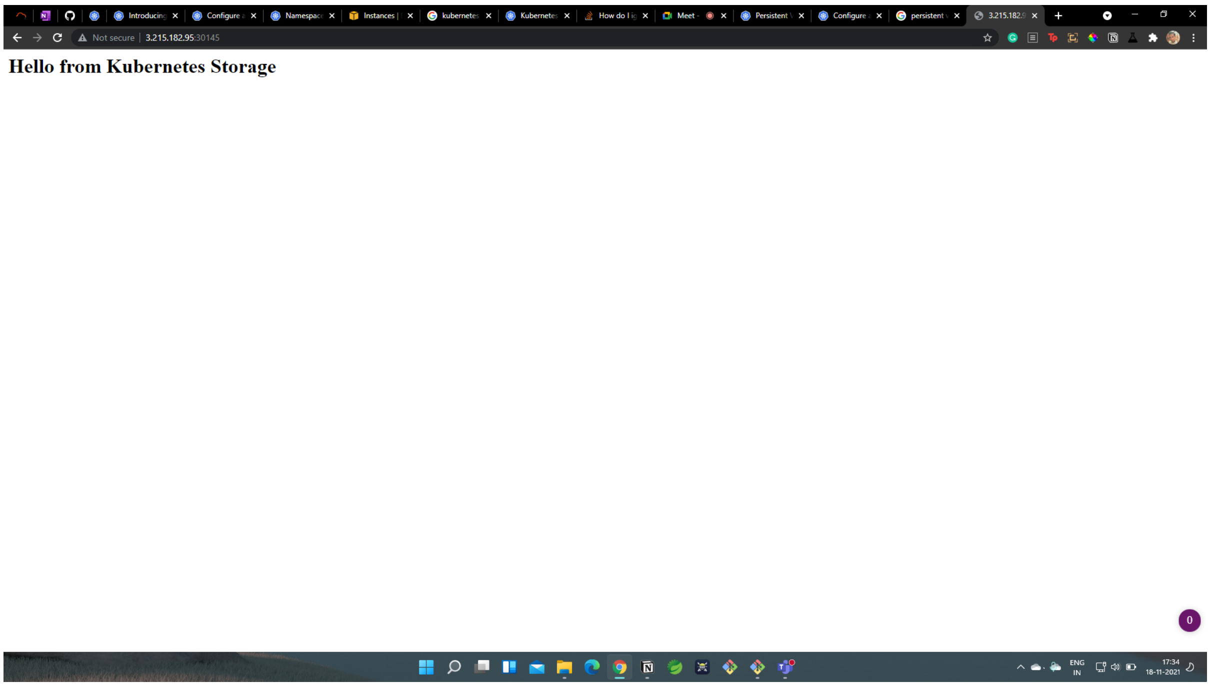
```
root@ip-172-31-72-71:~/capgeminimanifestnov# kubectl expose pod task-pv-pod --type=NodePort --port=80
service/task-pv-pod exposed
```

Here, we can see that the pod task-pv-pod is exposed

Now since the pod is exposed successfully we need to copy the port number of the task-pv-pod by using the command kubectl get svc

```
root@ip-172-31-72-71:~/capgeminimanifestnov# kubectl get svc
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP        6d3h
my-frist-service    NodePort    10.101.97.235    <none>        80:31506/TCP   2d
myapp-deployment    NodePort    10.102.103.44    <none>        80:30914/TCP   27h
task-pv-pod         NodePort    10.104.180.206   <none>        80:30145/TCP   12s
```

The copied port number should be added with the IP address of the node/slave machine


Hello from Kubernetes Storage

This is the message written inside the index.html file his can also be changed as per the user requirement

Open the node/slave machine become the root user get into /mnt/data directory and then open the vi index.html file and make the changes in the vi file

```
root@ip-172-31-78-206:~# cd /mnt/data
root@ip-172-31-78-206:/mnt/data# vi index.html
```

After the changes just refresh the browser page to see the changes .