ThoughtWorks®

STORAGE & VOLUMES

Rise of the Containers Workshop

Topics

1. Volumes
2. Persistent Volumes
3. Persistent Volume Claims
4. Storage Classes

Containers are ephemeral

Volumes

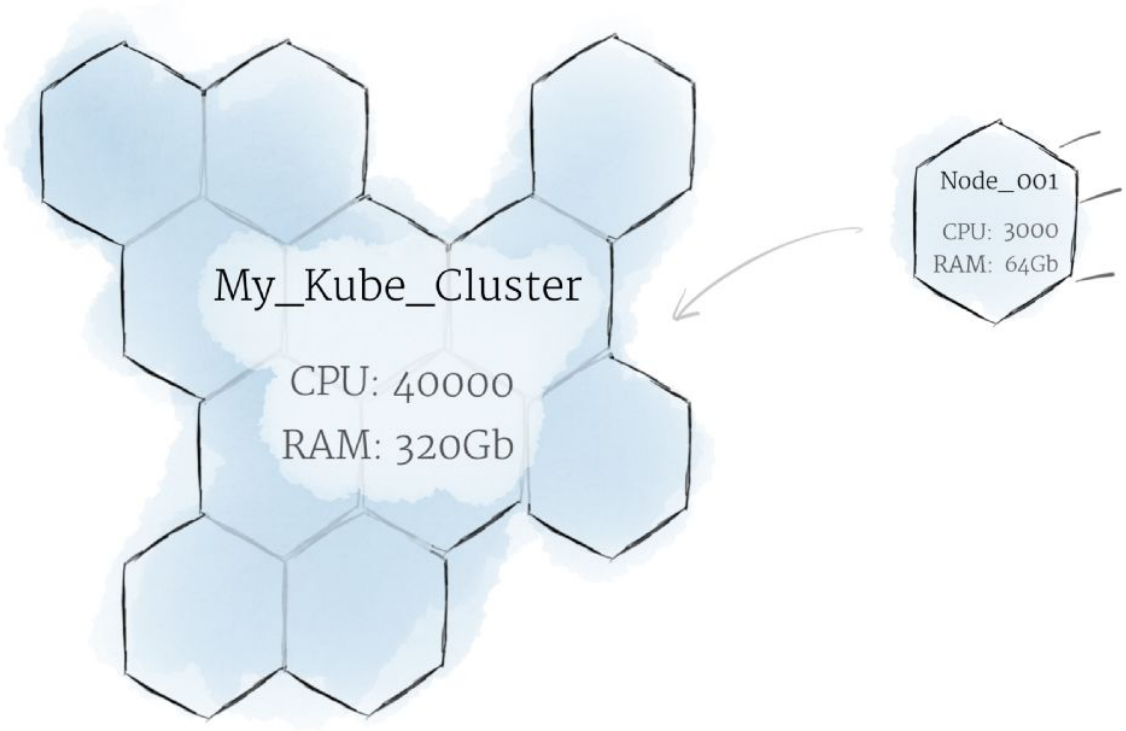
- Just a directory, accessible to all containers in pod
- Lasts as long as a pod
- The backing medium and contents depend on type of the volume
- Some important volume types:
  - `emptyDir`
  - `hostPath`
  - `awsElasticBlockStore`
  - `gcePersistentDisk`
  - `nfs`
  - `persistentVolumeClaim`

*The **emptyDir** volume type*

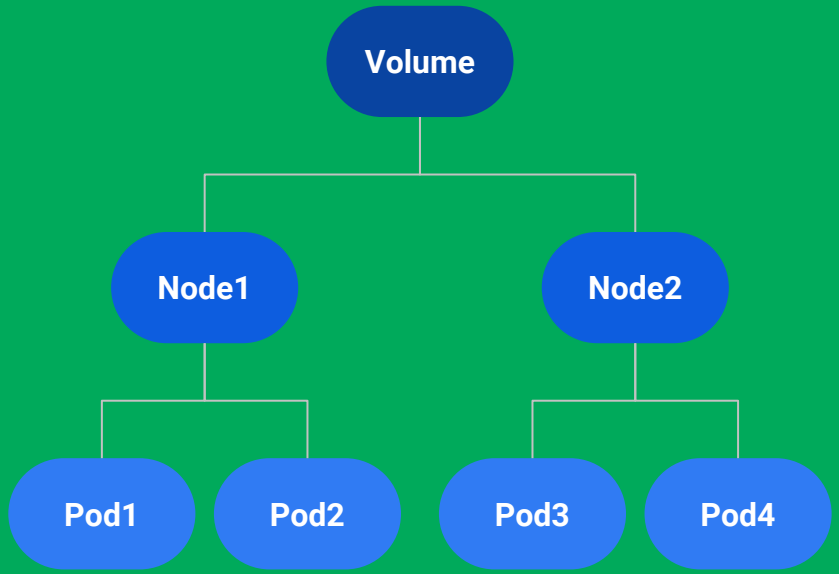
- Created at pod creation time and accessible by all containers in the pod
- Stored on the backing node

```
apiVersion: v1
kind: Pod
metadata:
  name: empty-dir-pod
spec:
  containers:
  - image: gcr.io/google-samples/node-hello:1.0
    name: empty-dir-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

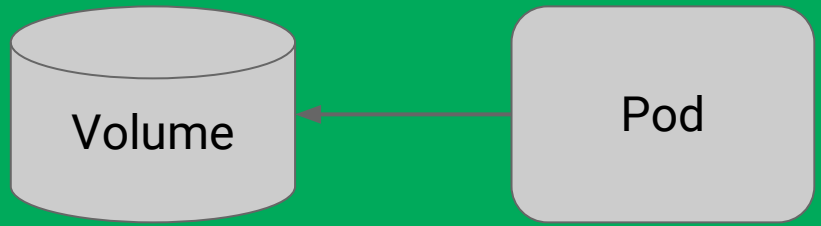
Why do we need better volumes?



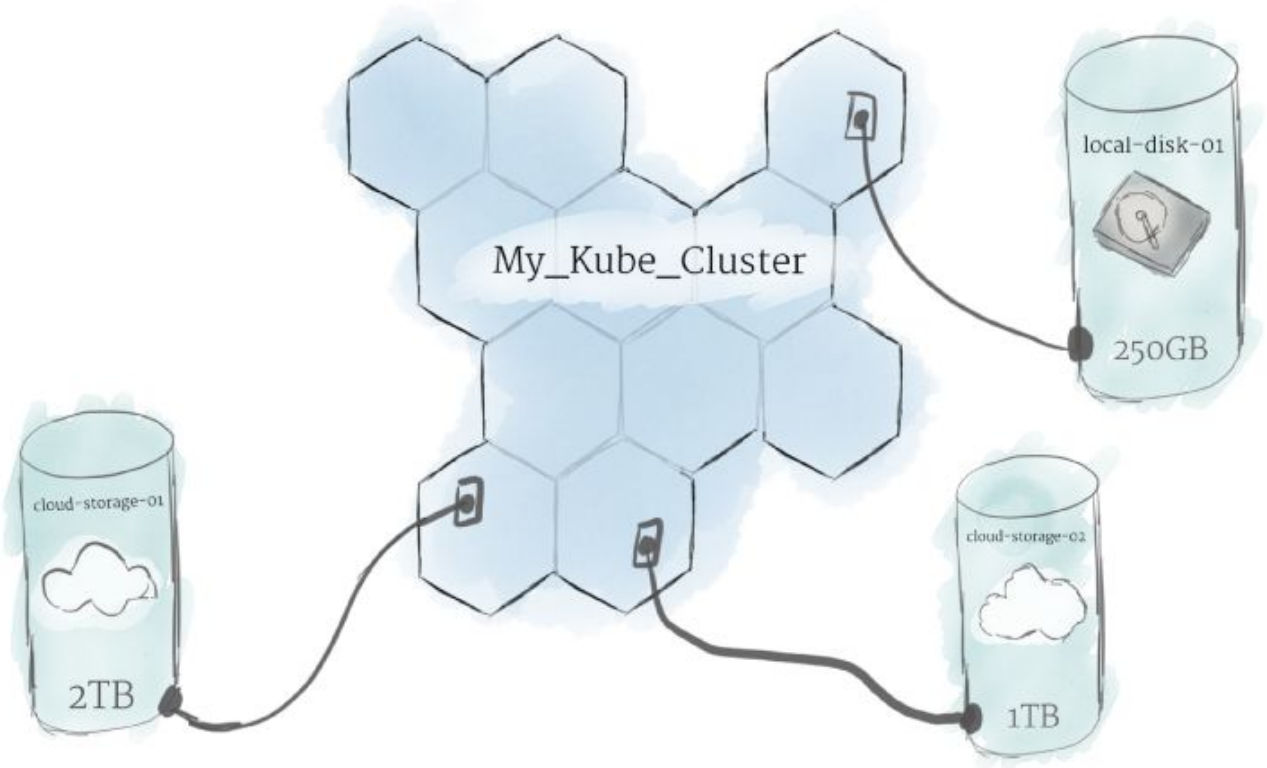
We need volumes that are available across the cluster and reliable.



We need to de-couple provision of storage from consumption of storage.



The perfect way to represent a volume in k8s



Persistent Volumes (PV)

- Persistent Volumes are a kubernetes resource that represents a volume
- Administered and provisioned independently of other resources
- Can be accessed from kubectl:

```
$ kubectl get pv
```

```
kind: PersistentVolume
```

Persistent Volume Claims (PVC)

- PVCs are kubernetes resources that represents a claim to a persistent volume (big surprise)
- A claim is a request for access to a persistent volume
- Claims can be used as volumes for pods

```
kind: PersistentVolumeClaim
```

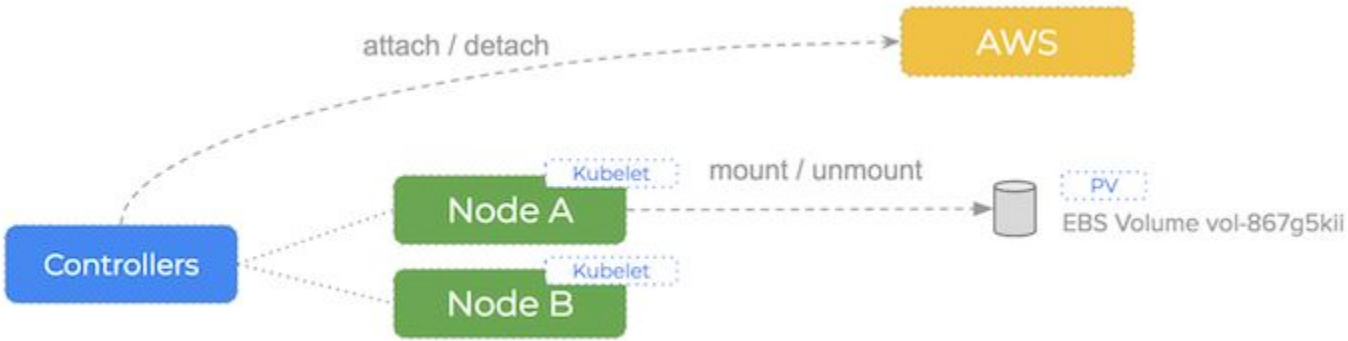
Analogy Question (2 Marks)

PersistentVolumeClaim:PersistentVolume::_____:_____

1) Pod:Service
2) Service:Pod
3) Pod:Node
4) Pod:Deployment

Storage Classes

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  zone: us-east-1d
  iopsPerGB: "10"
```



Hands On Time!

```
$ minikube ssh

$ mkdir -p /tmp/test

$ echo “Hello Storage!” > /tmp/test/index.html
```

Let's create a persistent volume

```
kind: PersistentVolume
apiVersion: v1
metadata:
 name: mypv
 labels:
   type: local
spec:
 storageClassName: manual
 capacity:
   storage: 500Mi
 accessModes:
   - ReadWriteOnce
 persistentVolumeReclaimPolicy: Retain
 hostPath:
   path: "/tmp/test"
```

Now, create the corresponding volume claim

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: mypvc
spec:
 storageClassName: manual
 accessModes:
 - ReadWriteOnce
 resources:
   requests:
     storage: 500Mi
```

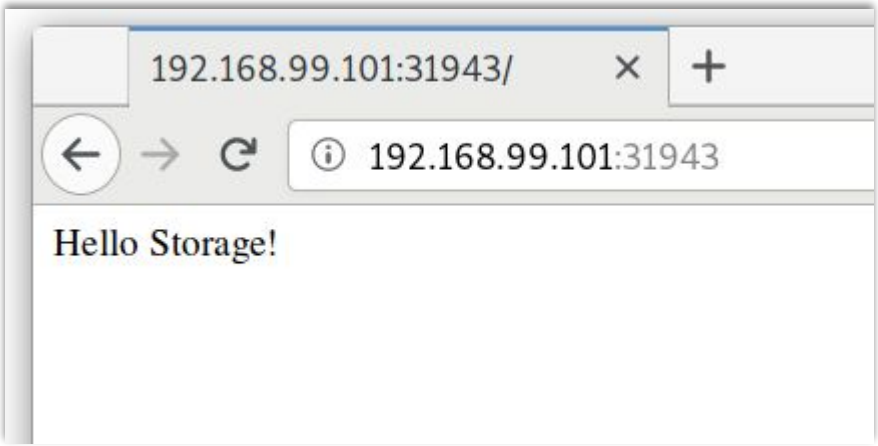
Spin up a pod to use the volume

```
kind: Pod
apiVersion: v1
metadata:
 name: mywebserver
 labels:
   type: mywebserver
spec:
 volumes:
   - name: public
     persistentVolumeClaim:
       claimName: mypvc
 containers:
   - name: mywebserver
     image: nginx
     ports:
       - containerPort: 80
         name: "http-server"
     volumeMounts:
       - mountPath: "/usr/share/nginx/html"
         name: public
```

Create a service to expose the app

```
kind: Service
apiVersion: v1
metadata:
 name: mywebservice
spec:
 selector:
   type: mywebserver
 ports:
 - protocol: TCP
   port: 80
   targetPort: 80
 type: NodePort
```

Access the application in the browser

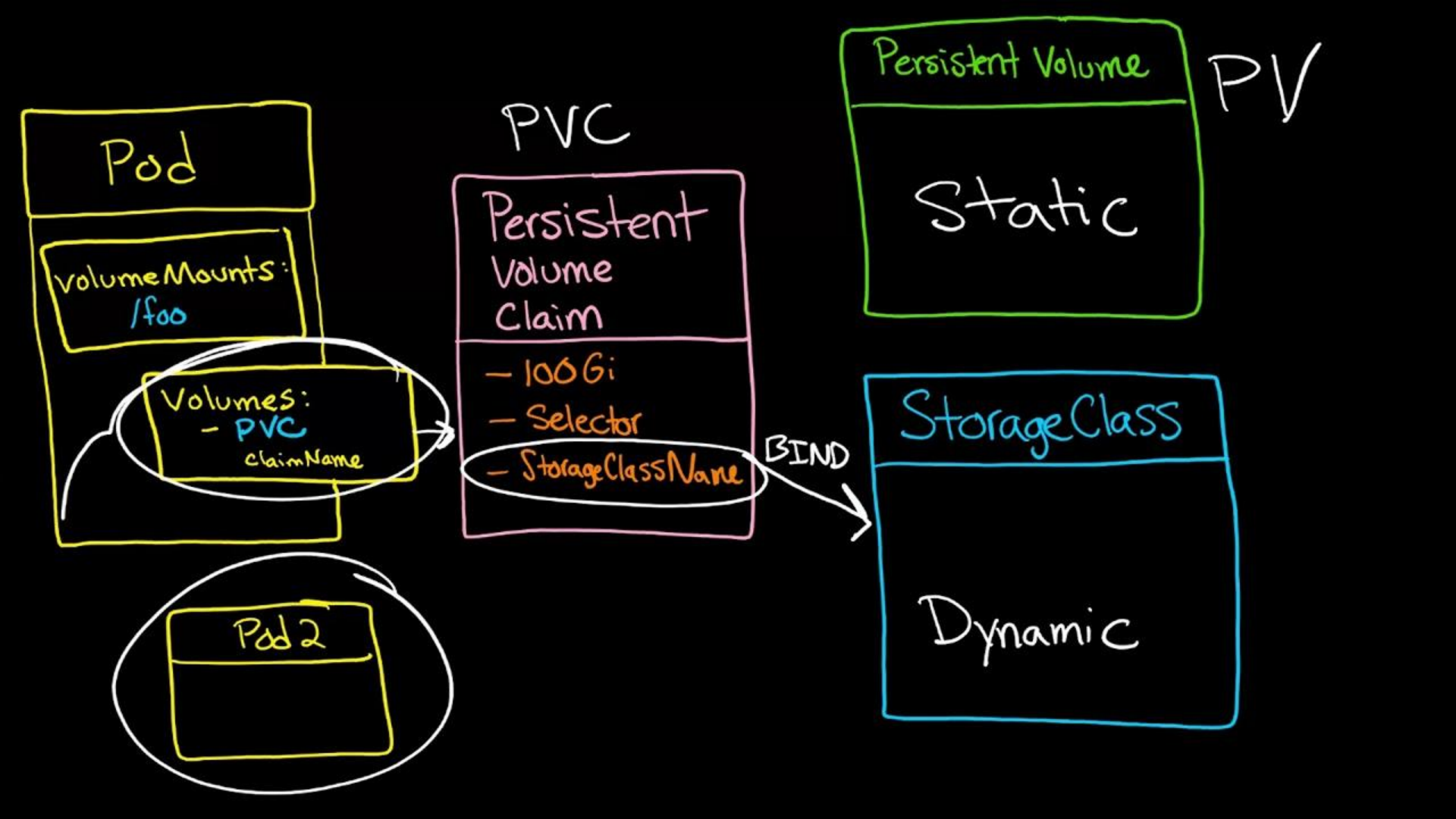


Exercise time!

Run the mongo database with the persistent volume we created

P.S. The mount path for mongo should be /data/db

ThoughtWorks®


Pod
VolumeMounts:
/foo
Volumes:
- PVC
claimName
Pod 2
PVC
Persistent
Volume
Claim
- 100Gi
- Selector
- StorageClassName
BIND
Persistent Volume
Static
PV
StorageClass
Dynamic

Persistent Volume Framework