BATCH : B107 AWS-DevOps

LESSON : **Kubernetes**

DATE : 24.05.2023

SUBJECT : **Volumes**

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ

Kubernetes

Review

In a K8s cluster:

○ For communication inside cluster use service ..	ClusterIP
○ For traffic coming out of the cluster use ..	NodePort
○ For traffic coming out of the cloud cluster use ..	LoadBalancer
○ For the service to pick up the proper pods use ..	Labels, Selectors

Storage

Pods by themselves are useful, but many workloads require **exchanging data between containers**, or **persisting some form of data**.

For this we have **Volumes, PersistentVolumes, PersistentVolumeClaims, and StorageClasses**, etc.

Volumes

- Storage that is tied to the Pod's Lifecycle.
- A pod can have one or more types of volumes attached to it.
- Can be consumed by any of the containers within the pod.
- **Survive Pod restarts**; however their durability beyond that is dependent on the Volume Type.

Volume Types

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- configMap
- csi
- downwardAPI
- emptyDir
- fc (fibre channel)
- flocker
- gcePersistentDisk
- gitRepo
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume

Persistent Volume Supported

Volumes

- `volumes:` A list of volume objects to be attached to the Pod. Every object within the list must have it's own unique `name`.
- `volumeMounts:` A container specific list referencing the Pod volumes by `name`, along with their desired `mountPath`.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-example
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
      ReadOnly: true
  - name: content
    image: alpine:latest
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          date >> /html/index.html;
          sleep 5;
        done
    volumeMounts:
    - name: html
      mountPath: /html
  volumes:
  - name: html
    emptyDir: {}
```

Volumes

- `volumes:` A list of volume objects to be attached to the Pod. Every object within the list must have it's own unique `name`.
- `volumeMounts:` A container specific list referencing the Pod volumes by `name`, along with their desired `mountPath`.

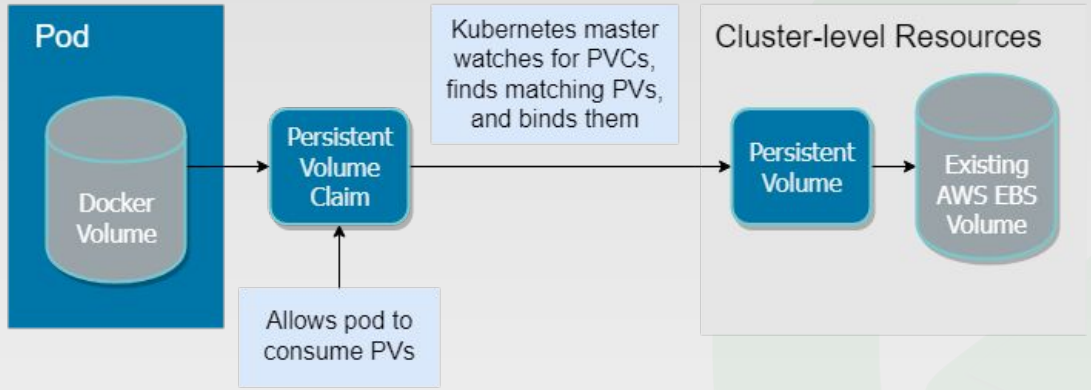
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-example
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
      ReadOnly: true
  - name: content
    image: alpine:latest
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          date >> /html/index.html;
          sleep 5;
        done
    volumeMounts:
    - name: html
      mountPath: /html
  volumes:
  - name: html
    emptyDir: {}
```

Persistent Volumes

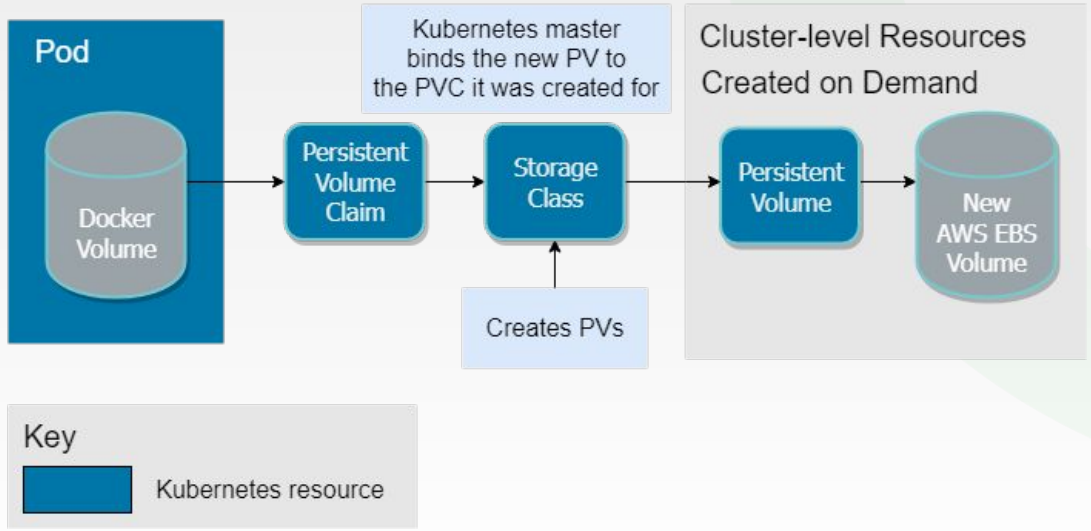
- A PersistentVolume (PV) represents a storage resource.
- **PVs are a cluster wide resource** linked to a backing storage provider: NFS, GCEPersistentDisk, RBD etc.
- Generally provisioned by an administrator.
- Their lifecycle is handled independently from a pod
- **CANNOT be attached to a Pod directly.** Relies on a PersistentVolumeClaim

Persistent Volume Types

Setting Up Existing Persistent Volumes



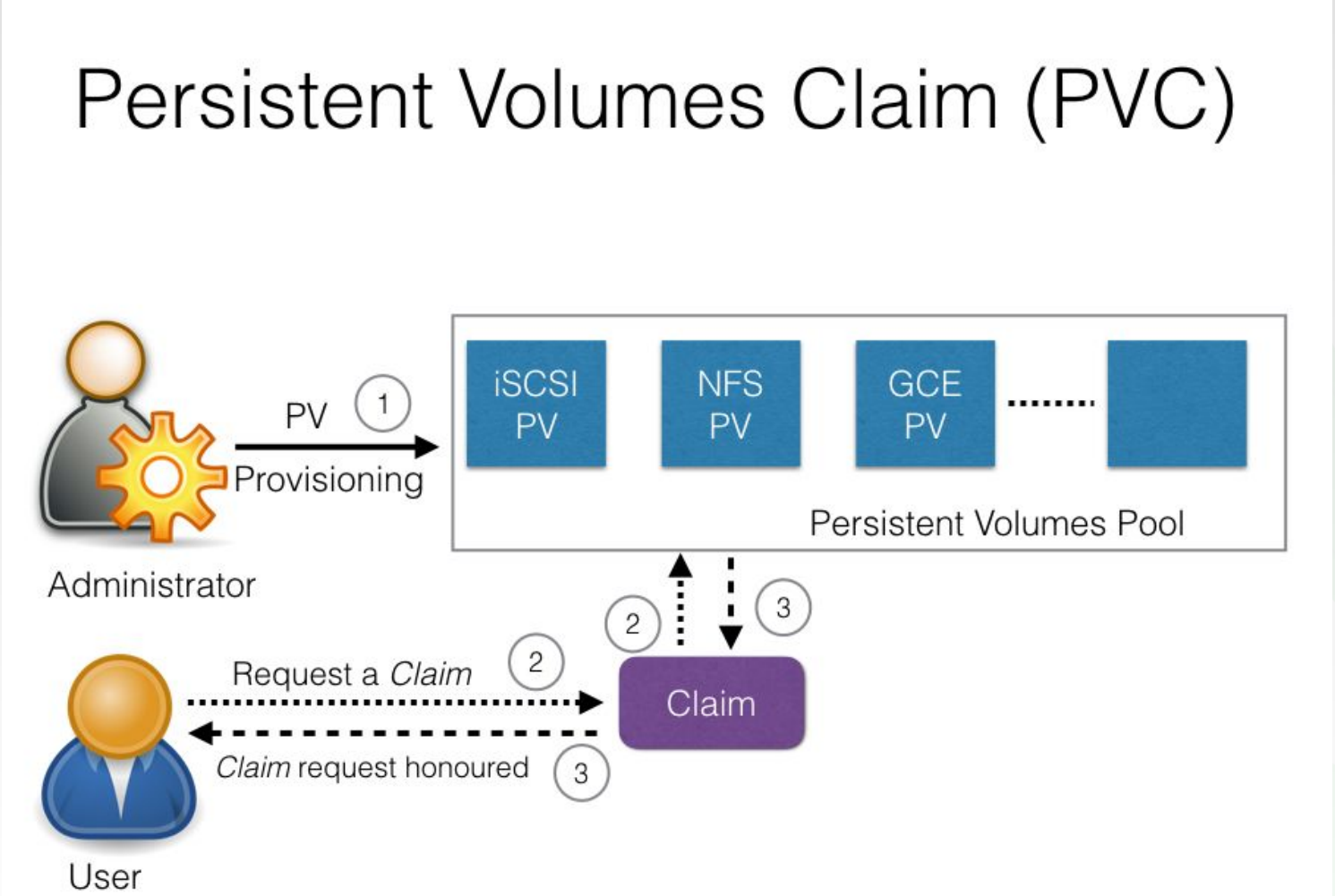
Dynamically Provisioning New Persistent Volumes



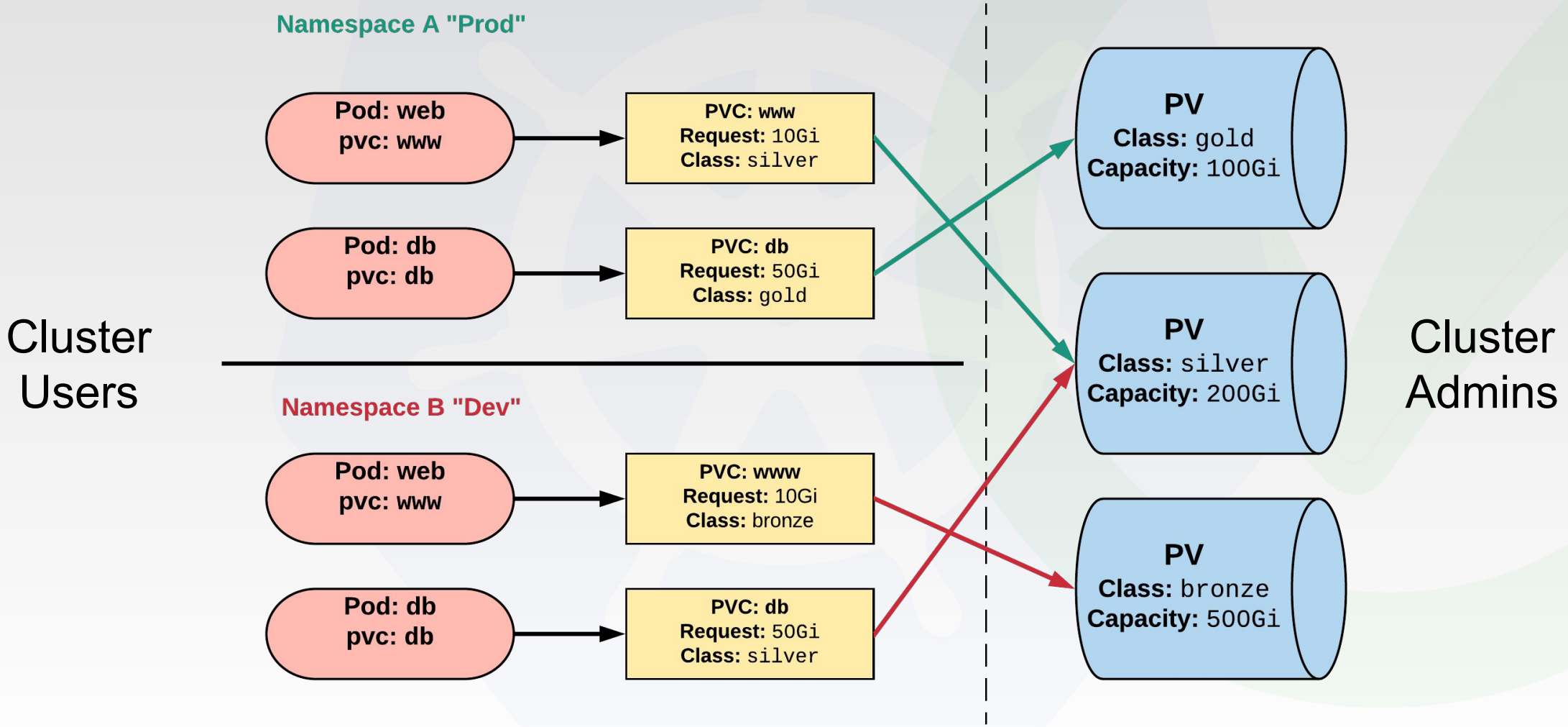
PersistentVolumeClaims

- A PersistentVolumeClaim (PVC) is a namespaced **request for storage**.
- Satisfies a set of requirements instead of mapping to a storage resource directly.
- Ensures that an application's '*claim*' for storage is portable across numerous backends or providers.

Persistent Volume Claim



Persistent Volumes and Claims



PersistentVolume

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy:
Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

- **`capacity.storage:`** **The total amount of available storage.**
- **`volumeMode:`** **The type of volume, this can be either `Filesystem` or `Block`.**
- **`accessModes:`** **A list of the supported methods of accessing the volume. Options include:**
  - `ReadWriteOnce` (read write by single node)
  - `ReadOnlyMany` (read only by many nodes)
  - `ReadWriteMany` (read write by many nodes)
  - `ReadWriteOncePod` (read write by single pod)

PersistentVolume

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

- `persistentVolumeReclaimPolicy:` The behaviour for PVC's that have been deleted. Options include:
  - `Retain` - manual clean-up
  - `Delete` - storage asset deleted by provider.
- `storageClassName:` Optional name of the storage class that PVC's can reference. If provided, ONLY PVC's referencing the name consume it.
- `mountOptions:` Optional mount options for the PV.

PersistentVolumeClaim

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-sc-example
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: slow
```

- **accessModes:** The selected method of accessing the storage. This MUST be a subset of what is defined on the target PV or Storage Class.
  - ReadWriteOnce
  - ReadOnlyMany
  - ReadWriteMany
- **resources.requests.storage:** The desired amount of storage for the claim
- **storageClassName:** The name of the desired Storage Class

PVs and PVCs with Selectors

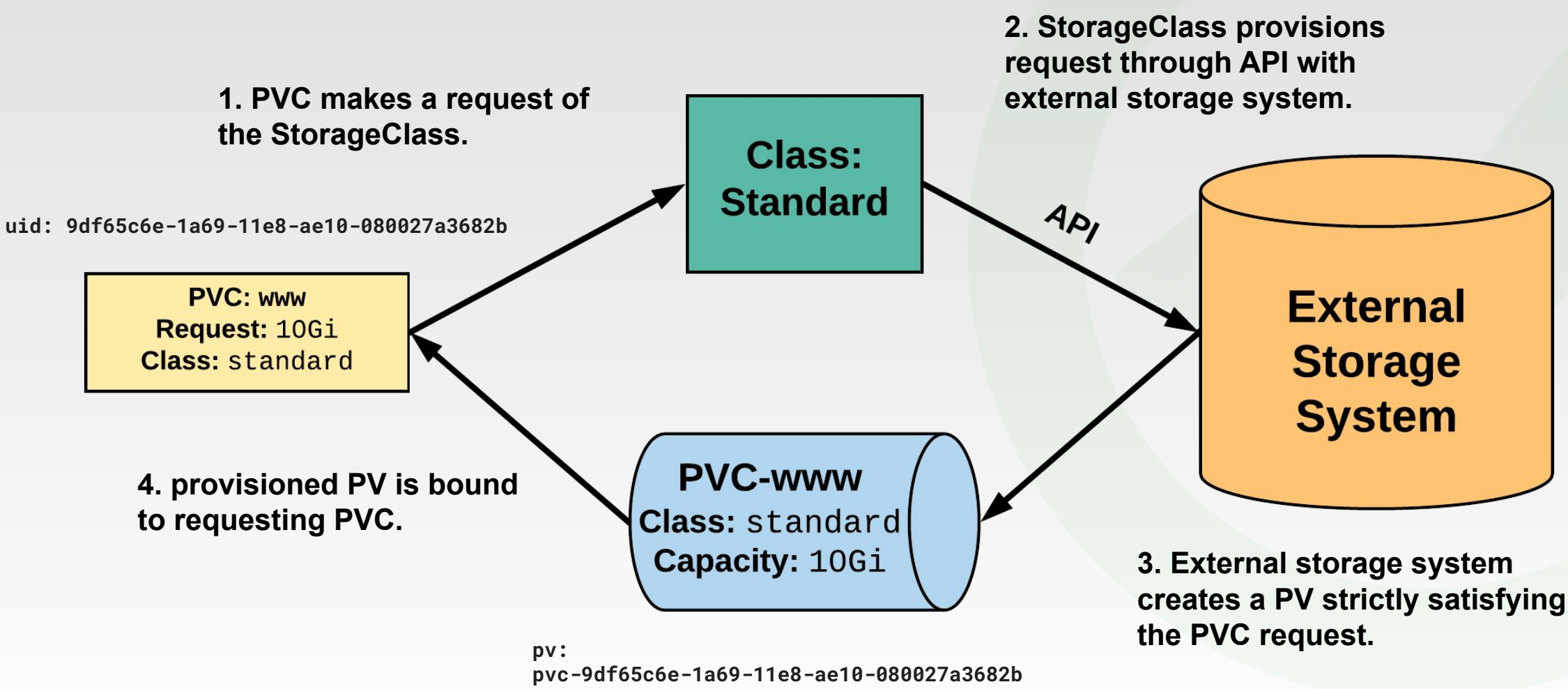
```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-selector-example
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      type: hostpath
```

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv-selector-example
  labels:
    type: hostpath
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
```

StorageClass

- Storage classes are an abstraction on top of an external storage resource (PV)
- Work **hand-in-hand with the external storage** system to enable dynamic provisioning of storage
- Eliminates the need for the cluster admin to pre-provision a PV

StorageClass



StorageClass

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
  zones: us-central1-a, us-central1-b
reclaimPolicy: Delete
```

- `provisioner:` Defines the '*driver*' to be used for provisioning of the external storage.
- `parameters:` A hash of the various configuration parameters for the provisioner.
- `reclaimPolicy:` The behaviour for the backing storage when the PVC is deleted.
  - `Retain` - manual clean-up
  - `Delete` - storage asset deleted by provider

Available StorageClasses

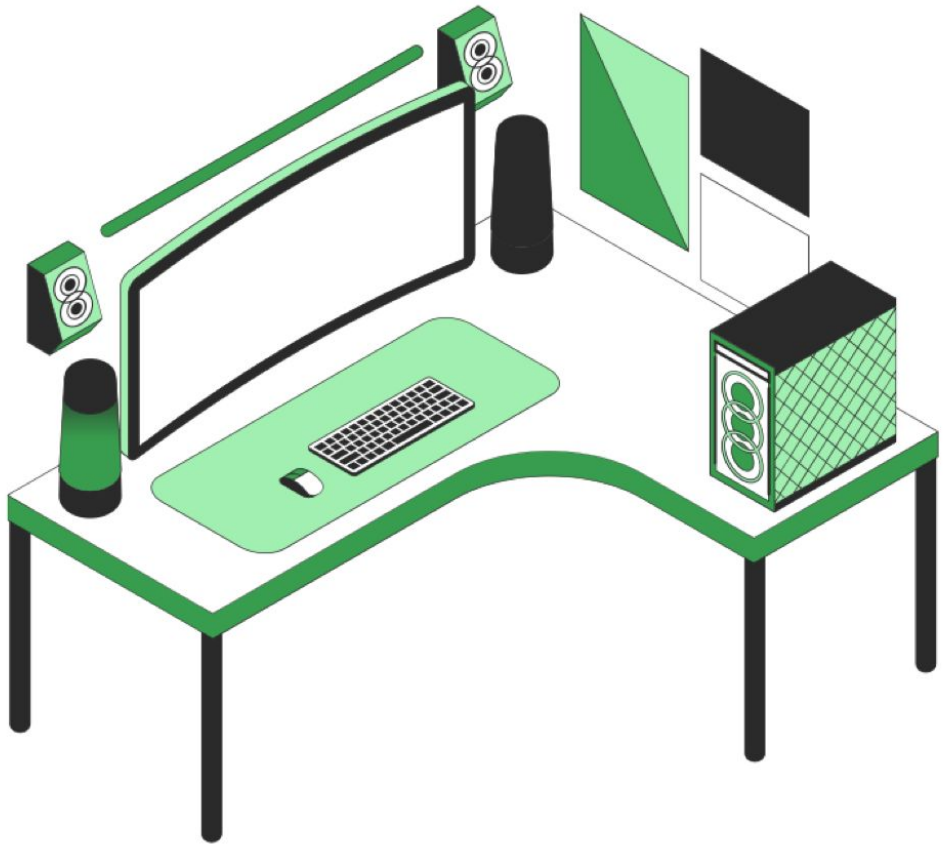
- AWSElasticBlockStore
- AzureFile
- AzureDisk
- CephFS
- Cinder
- FC
- Flocker
- GCEPersistentDisk
- Glusterfs
- iSCSI
- Quobyte
- NFS
- RBD
- VsphereVolume
- PortworxVolume
- ScaleIO
- StorageOS
- Local

Internal Provisioner

Summary

In a K8s cluster:

○ For temporary storage use ..	Volumes
○ This type of storage opens nodes' file system ..	hostPath
○ For permanent storage use ..	PersistentVolume
○ To be able to use permanent storage create ..	PersistentVolumeClaim



Do you have any questions?

Send it to us! We hope you learned something new.