Understanding Storage Options

Dan Wahlin
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

Module Overview

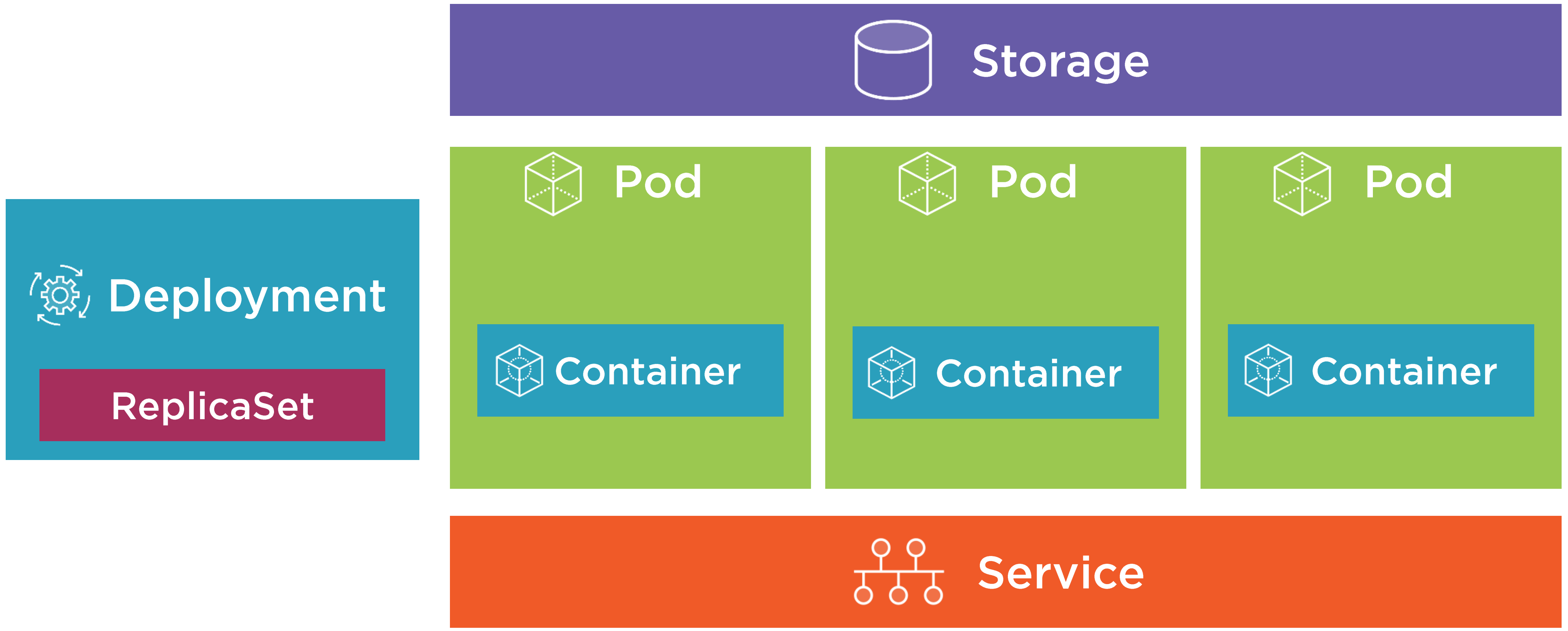
Storage Core Concepts

Volumes

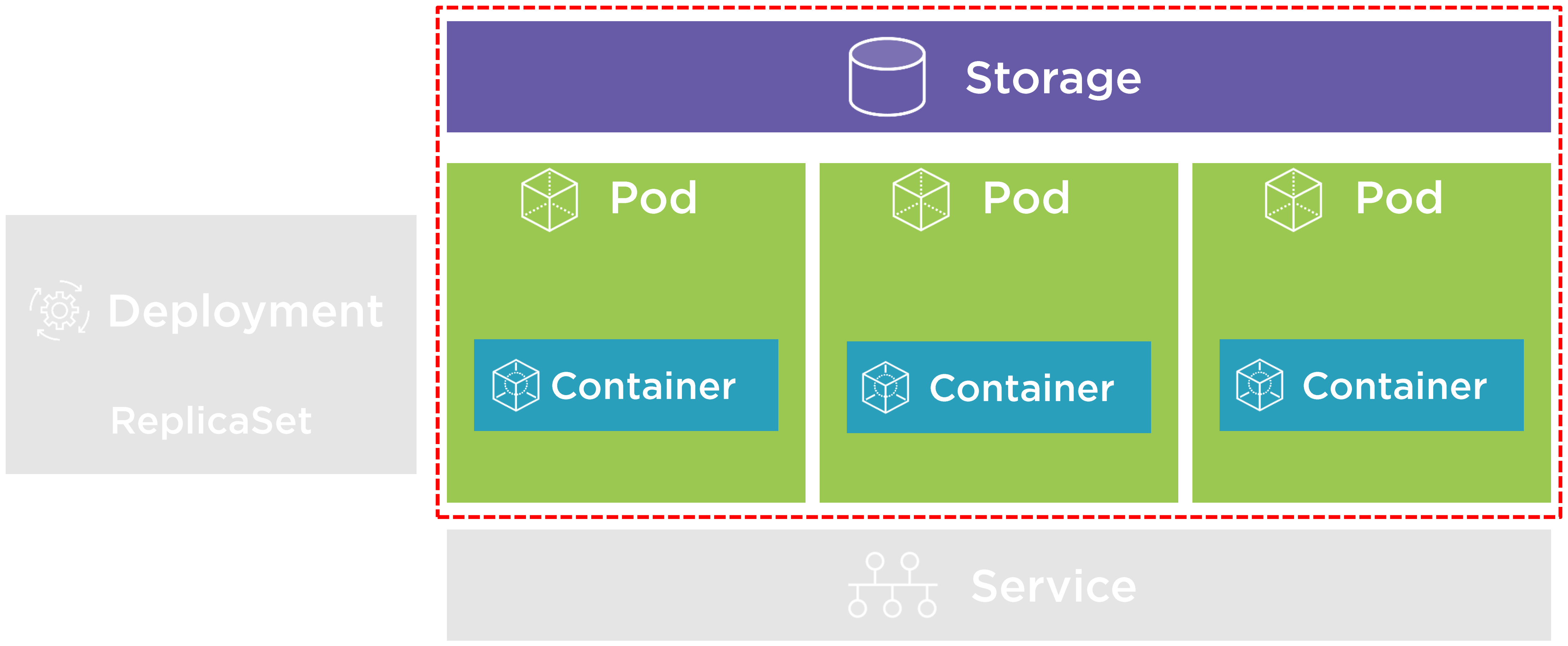
PersistentVolumes and PersistentVolumeClaims

StorageClasses

You Are Here


Storage
Pod
Pod
Pod
Deployment
ReplicaSet
Container
Container
Container
Service

You Are Here


Storage
Pod
Pod
Pod
Container
Container
Container
Deployment
ReplicaSet
Service

Storage Core Concepts

Question:

How do you store application state/data and exchange it between Pods with Kubernetes?

Answer:

Volumes (although other data storage options exist)

A Volume can be used to hold data and state for Pods and containers.

Pod State and Data

Pods live and die so their file system is short-lived (ephemeral)

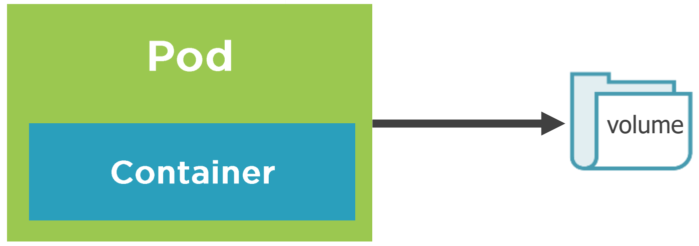
Volumes can be used to store state/data and use it in a Pod

A Pod can have multiple Volumes attached to it

Containers rely on a mountPath to access a Volume

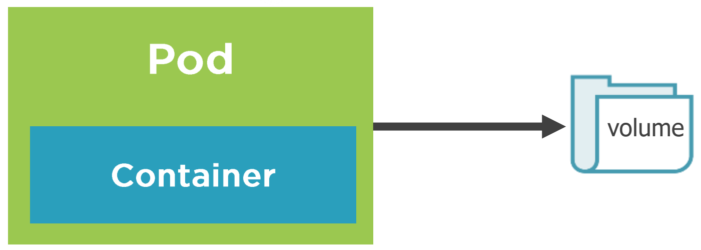
Kubernetes supports:

- Volumes
- PersistentVolumes
- PersistentVolumeClaims
- StorageClasses



Volumes

Volumes and Volume Mounts



A Volume references a storage location

Must have a unique name

Attached to a Pod and may or may not be tied to the Pod's lifetime (depending on the Volume type)

A Volume Mount references a Volume by name and defines a mountPath

Volumes Type Examples

Pod
Container
volume

emptyDir – Empty directory for storing "transient" data (shares a Pod's lifetime) useful for sharing files between containers running in a Pod

hostPath – Pod mounts into the node's filesystem

nfs – An NFS (Network File System) share mounted into the Pod

configMap/secret – Special types of volumes that provide a Pod with access to Kubernetes resources

persistentVolumeClaim – Provides Pods with a more persistent storage option that is abstracted from the details

Cloud – Cluster-wide storage

Volume Types

awsElasticBlockStore	azureDisk	azureFile	cephfs	configMap
csi	downwardAPI	emptyDir	fc	flexVolume
flocker	gcePersistentDisk	glusterfs	hostPath	iscsi
local	nfs	persistentVolumeClaim	projected	portworxVolume
quobyte	rbd	scaleIO	secret	storageos
vsphereVolume				

Defining an emptyDir Volume

```yaml
apiVersion: v1
kind: Pod
spec:
 volumes:
     - name: html
       emptyDir: {}
  containers:
  - name: nginx
    image: nginx:alpine
    volumeMounts:
      - name: html
        mountPath: /usr/share/nginx/html
        readOnly: true
  - name: html-updater
    image: alpine
    command: ["/bin/sh", "-c"]
    args:
      - while true; do date >> /html/index.html;
          sleep 10; done
    volumeMounts:
      - name: html
        mountPath: /html
```

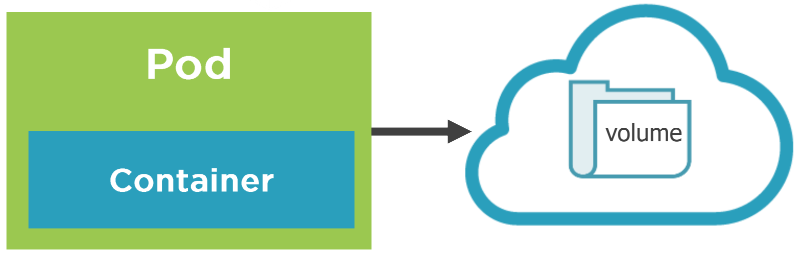
- Define initial Volume named "html" that is an empty directory (lifetime of the Pod)
- Reference "html" Volume and define a mountPath
- Update file in Volume mount /html path with latest date every 10 seconds
- Reference "html" Volume (defined above) and define a mountPath

Defining a hostPath Volume

```
apiVersion: v1
kind: Pod
spec:
 volumes:
    - name: docker-socket
      hostPath:
        path: /var/run/docker.sock
        type: Socket
  containers:
  - name: docker
    image: docker
    command: ["sleep"]
    args: ["100000"]
    volumeMounts:
      - name: docker-socket
        mountPath: /var/run/docker.sock
```

- Define a socket volume on host that points to /var/run/docker.sock
- Reference "docker-socket" Volume and define mountPath

Cloud Volumes



Cloud providers (Azure, AWS, GCP, etc.) support different types of Volumes:

- Azure – Azure Disk and Azure File
- AWS – Elastic Block Store
- GCP – GCE Persistent Disk

Defining an Azure File Volume

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  volumes:
  - name: data
    azureFile:
      secretName: <azure-secret>
      shareName: <share-name>
      readOnly: false
  containers:
  - image: someimage
    name: my-app
    volumeMounts:
    - name: data
      mountPath: /data/storage
```

◀ Define initial Volume named "data" that is Azure File storage

◀ Reference "data" Volume and define a mountPath

Defining an AWS Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  volumes:
  - name: data
    awsElasticBlockStore:
      volumeID: <volume_ID>
      fsType: ext4
  containers:
  - image: someimage
    name: my-app
    volumeMounts:
    - name: data
      mountPath: /data/storage
```

◀ Define initial Volume named "data" that is a awsElasticBlockStore

◀ Reference "data" Volume and define a mountPath

Defining a Google Cloud gcePersistentDisk Volume

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  volumes:
  - name: data
    gcePersistentDisk:
      pdName: datastorage
      fsType: ext4
  containers:
  - image: someimage
    name: my-app
    volumeMounts:
    - name: data
      mountPath: /data/storage
```

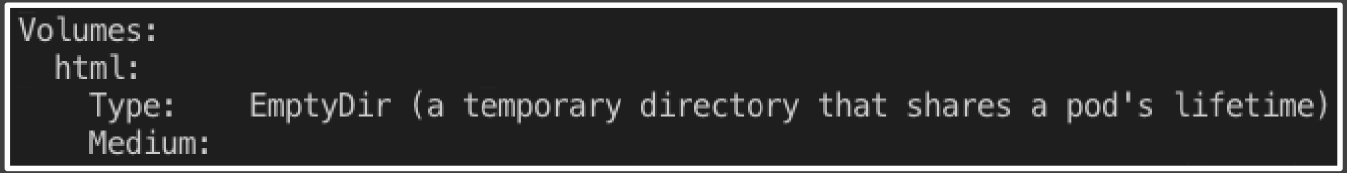
◂ Define initial Volume named "data" that is a gcePersistentDisk

◂ Reference "data" Volume and define a mountPath

Viewing a Pod's Volumes

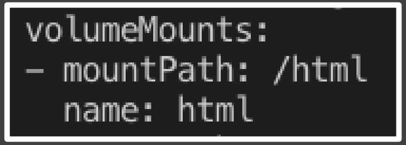
Several different techniques can be used to view a Pod's Volumes

```
# Describe Pod
kubectl describe pod [pod-name]
```



```
Volumes:
  html:
    Type:    EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:
```

```
# Get Pod YAML
kubectl get pod [pod-name] -o yaml
```

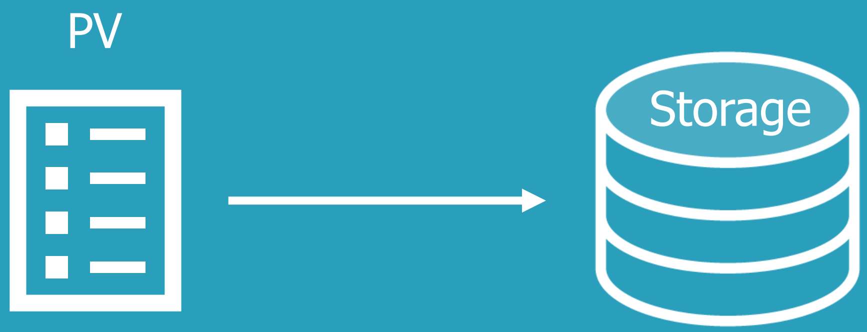


```
volumeMounts:
- mountPath: /html
  name: html
```

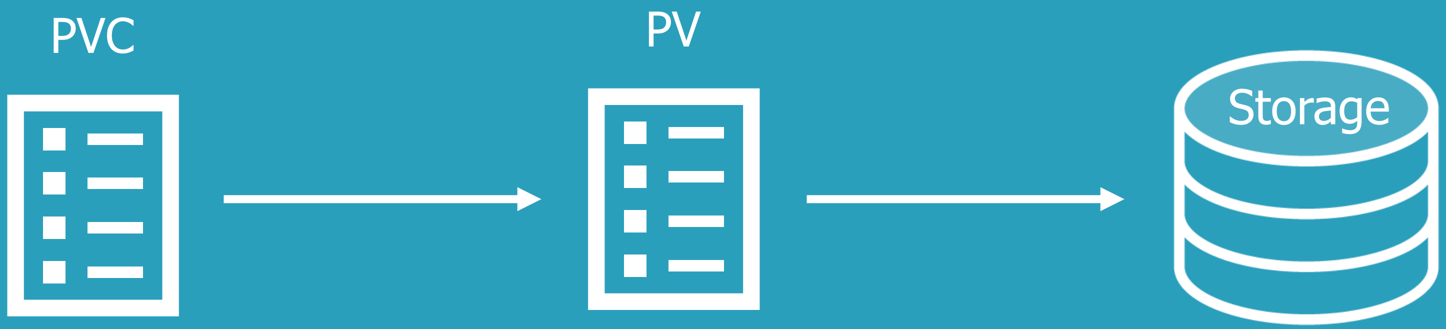
Volumes in Action

PersistentVolumes and PersistentVolumeClaims

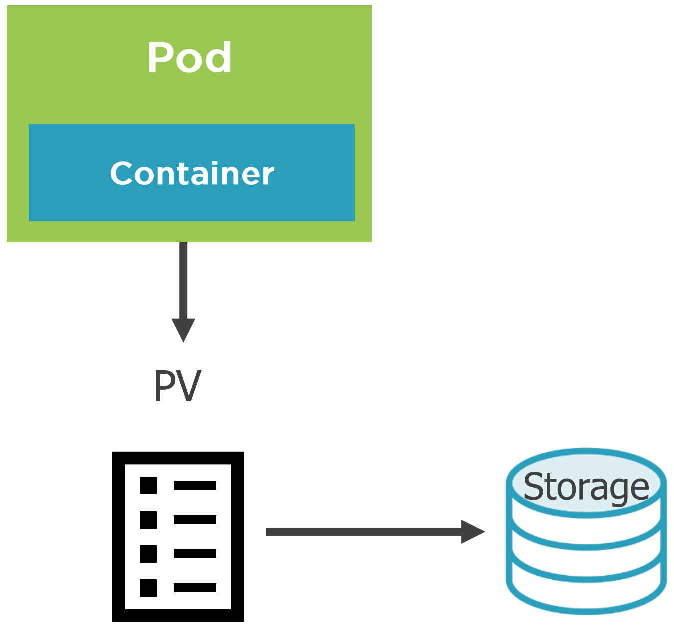
A PersistentVolume (PV) is a cluster-wide storage unit provisioned by an administrator with a lifecycle independent from a Pod.



A PersistentVolumeClaim (PVC) is a request for a storage unit (PV).



PersistentVolume



A PersistentVolume is a cluster-wide storage resource that relies on network-attached storage (NAS)

Normally provisioned by a cluster administrator

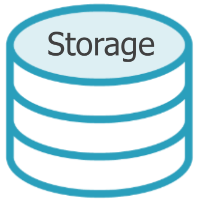
Available to a Pod even if it gets rescheduled to a different Node

Rely on a storage provider such as NFS, cloud storage, or other options

Associated with a Pod by using a PersistentVolumeClaim (PVC)

PersistentVolume Workflow

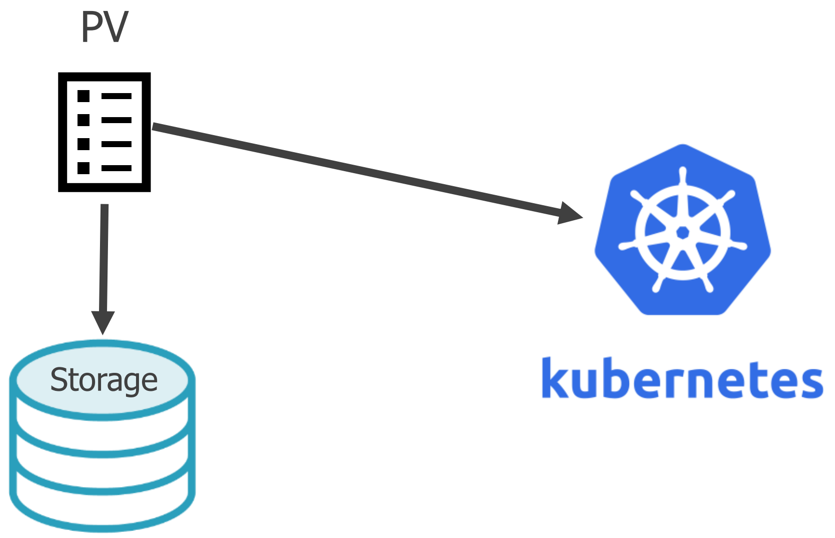
1 **Create network storage resource (NFS, cloud, etc.)**



PersistentVolume Workflow

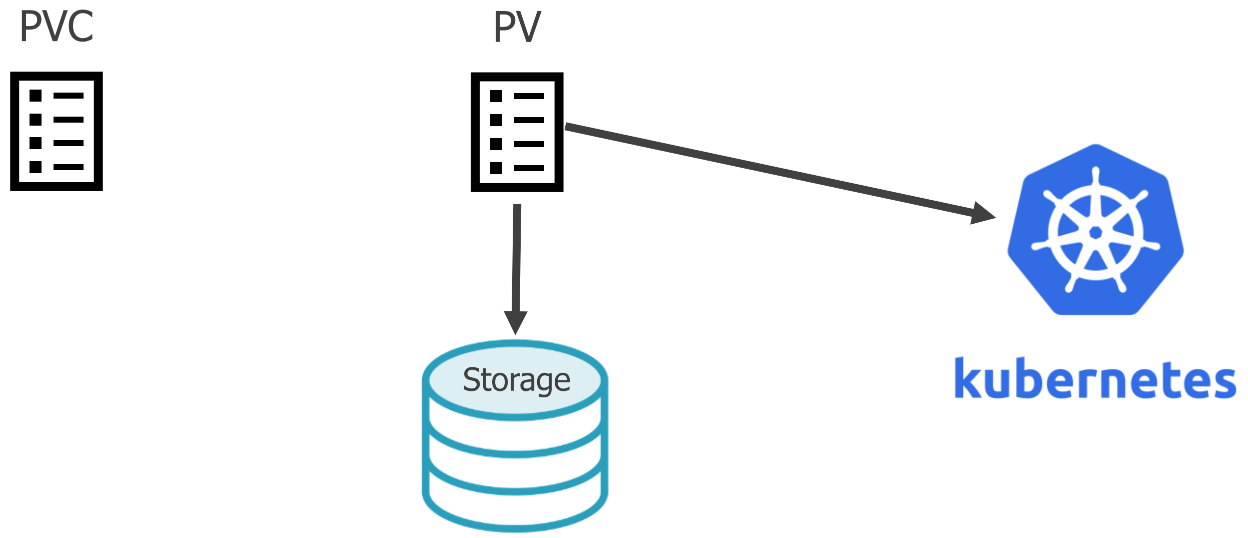
2

Define a Persistent Volume (PV) and send to the Kubernetes API



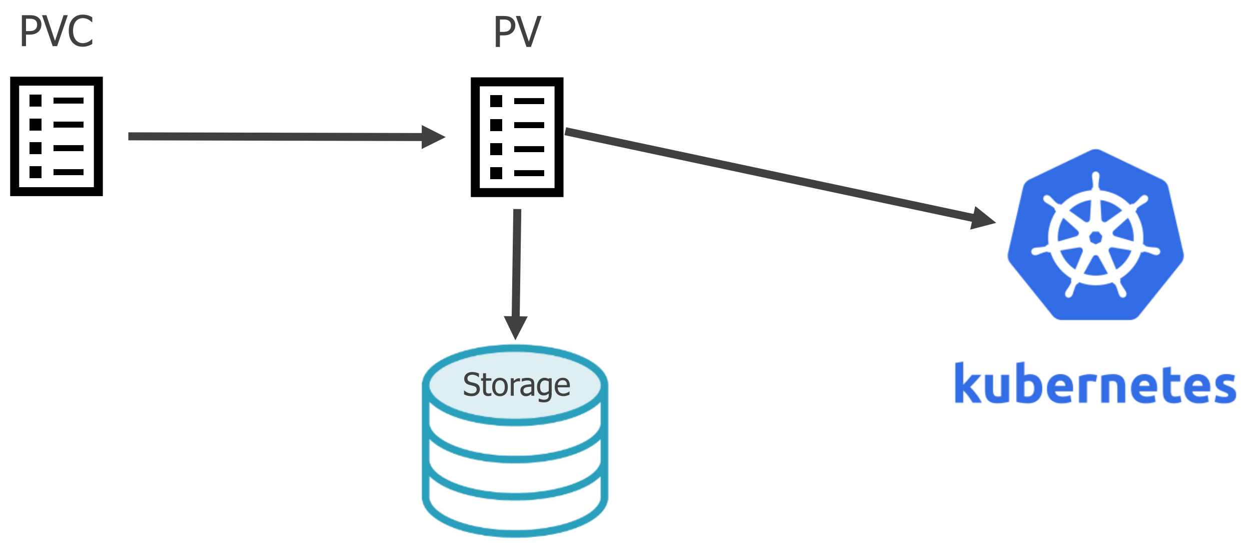
PersistentVolume Workflow

3 **Create a PersistentVolumeClaim (PVC)**



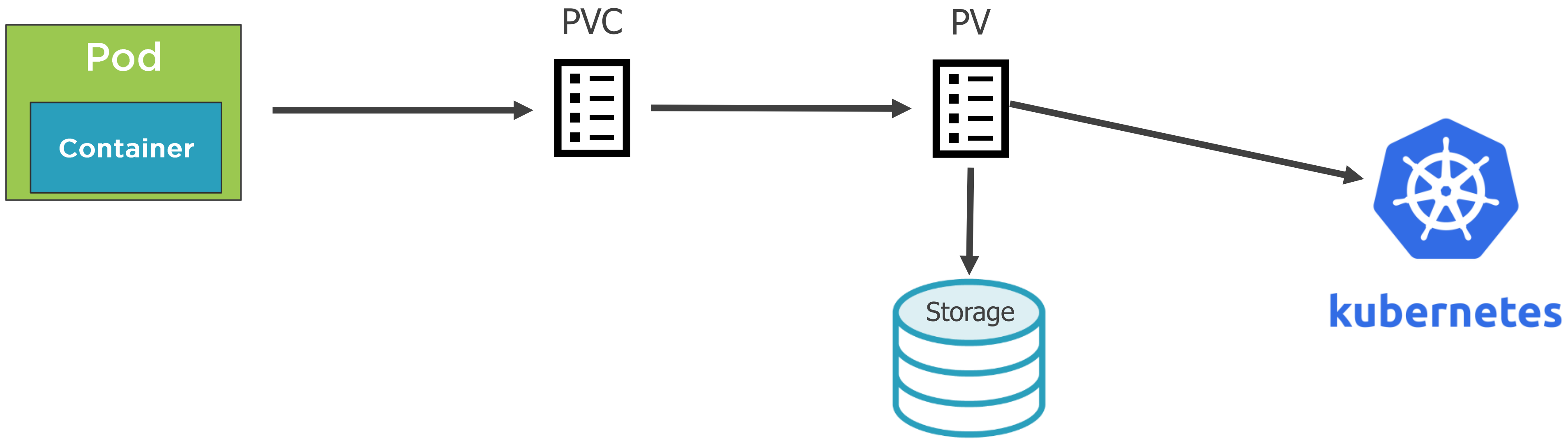
PersistentVolume Workflow

4 **Kubernetes binds the PVC to the PV**



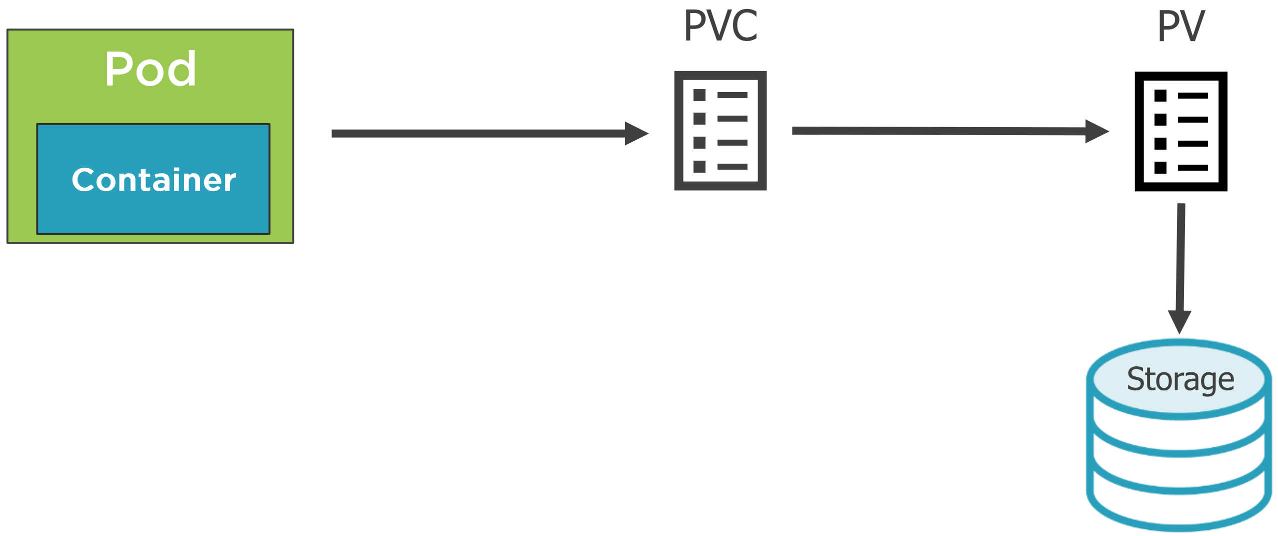
PersistentVolume Workflow

5 **Pod Volume references the PVC**

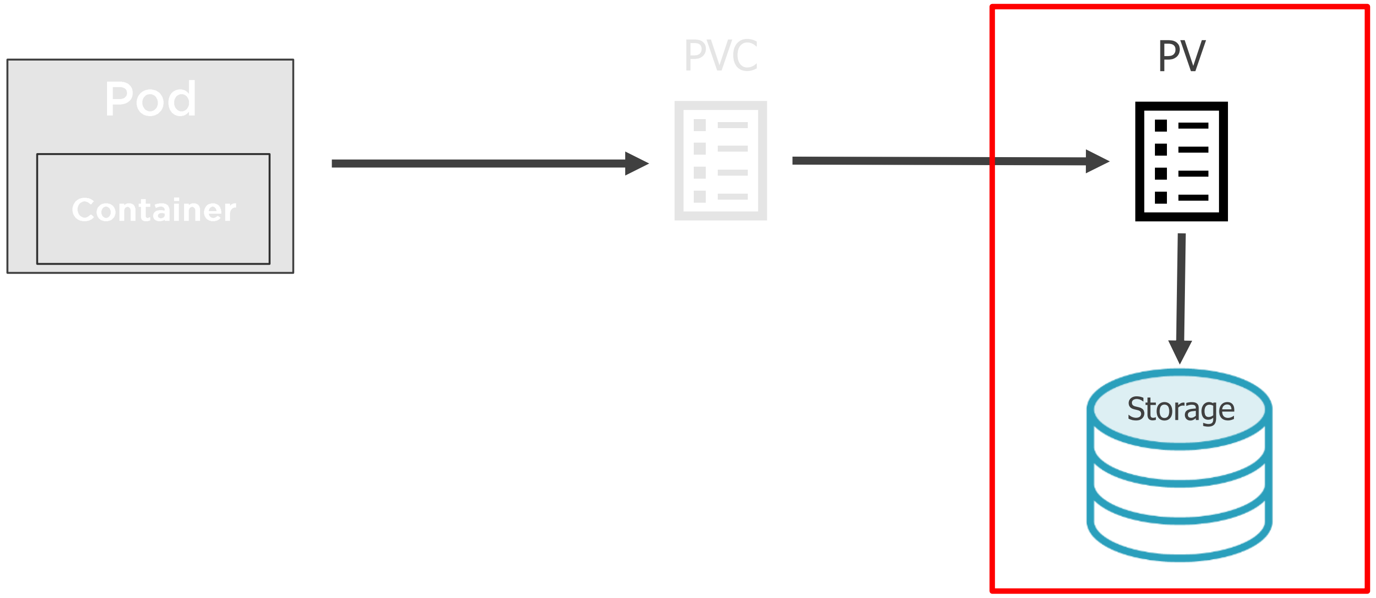


PersistentVolume and PersistentVolumeClaim YAML

Defining a PV and PVC



Creating a PersistentVolume



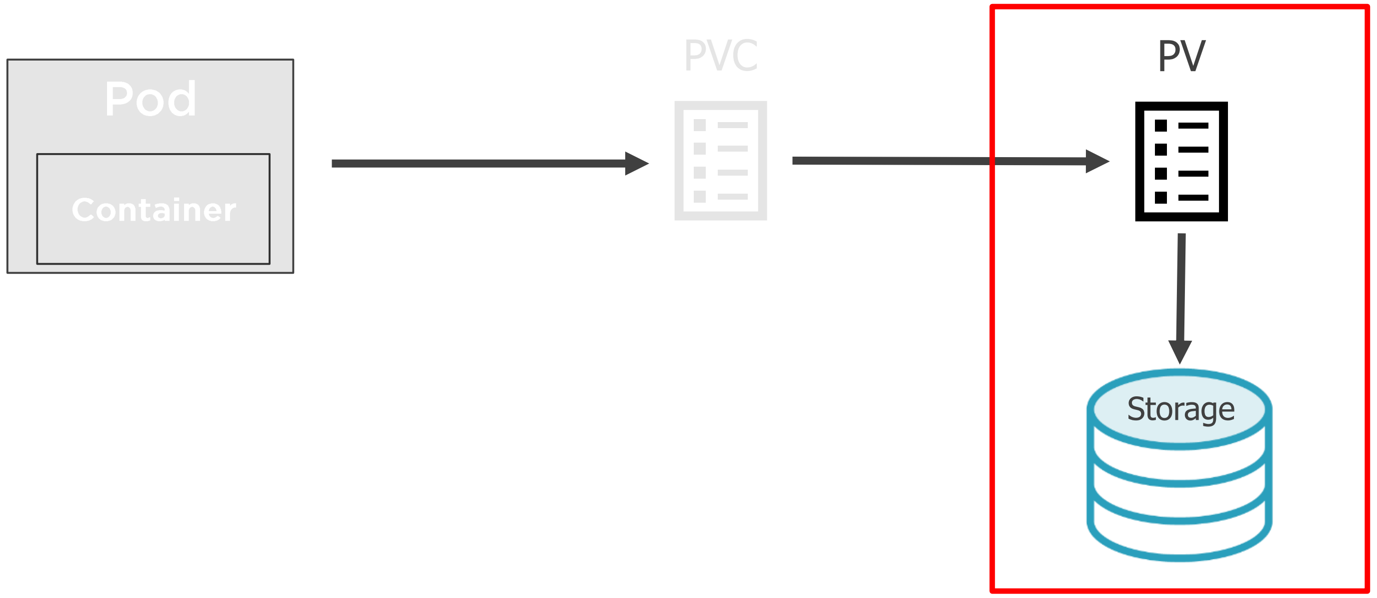
Defining a PersistentVolume for Azure

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity: 10Gi
  accessModes:
    - ReadWriteOnce
    - ReadOnlyMany
  persistentVolumeRelaimPolicy: Retain
  azureFile:
    secretName: <azure-secret>
    shareName: <name_from_azure>
    readOnly: false
```

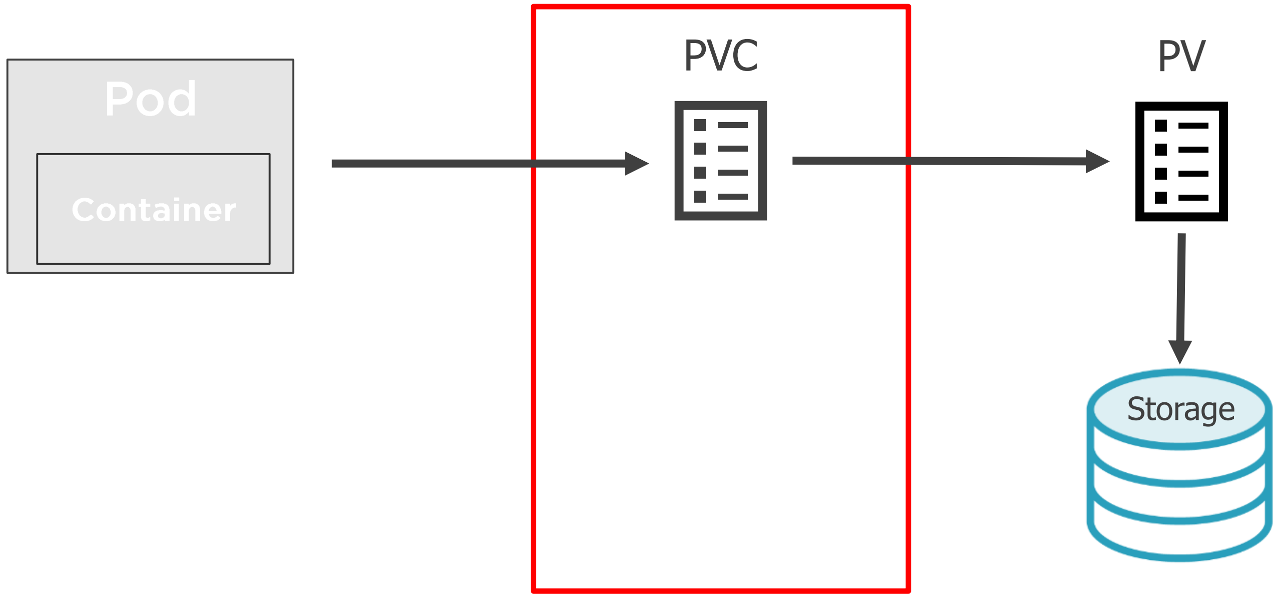
- Create PersistentVolume kind
- Define storage capacity
- One client can mount for read/write
- Many clients can mount for reading
- Retain even after claim is deleted (not erased/deleted)
- Reference storage to use (specific to Cloud provider, NFS setup, etc.)

https://github.com/kubernetes/examples

Creating a PersistentVolume



Creating a PersistentVolumeClaim



Defining a PersistentVolumeClaim

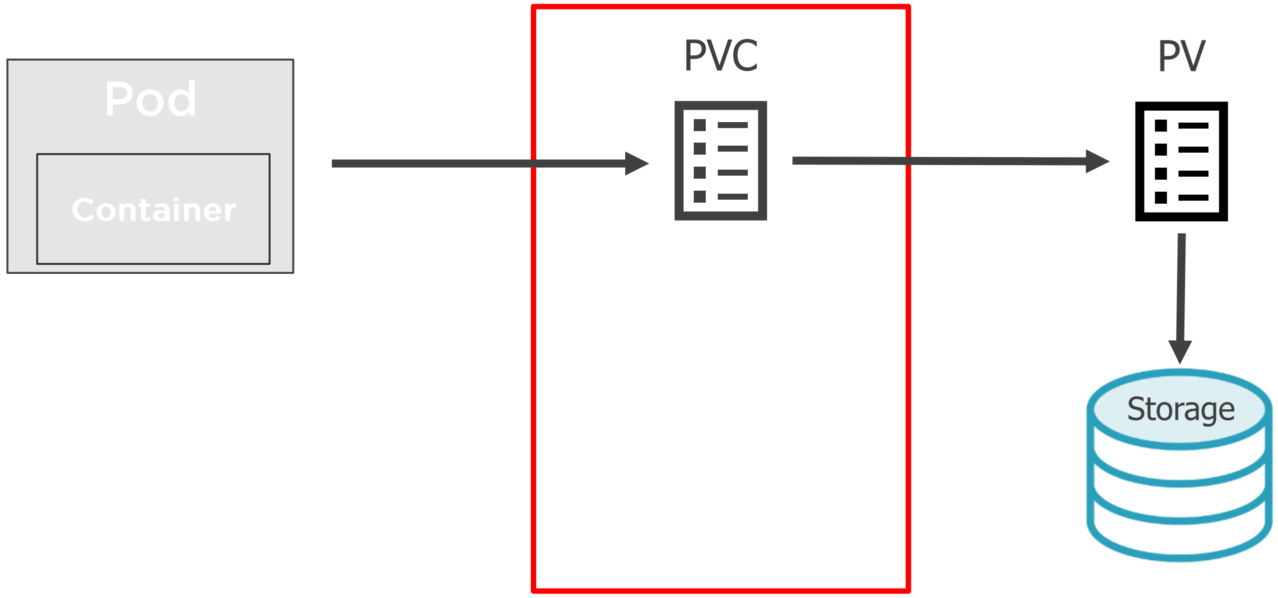
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pv-dd-account-hdd-5g
  annotations:
    volume.beta.kubernetes.io/storage-class: accounthdd
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

◀ Define a PersistentVolumeClaim (PVC)

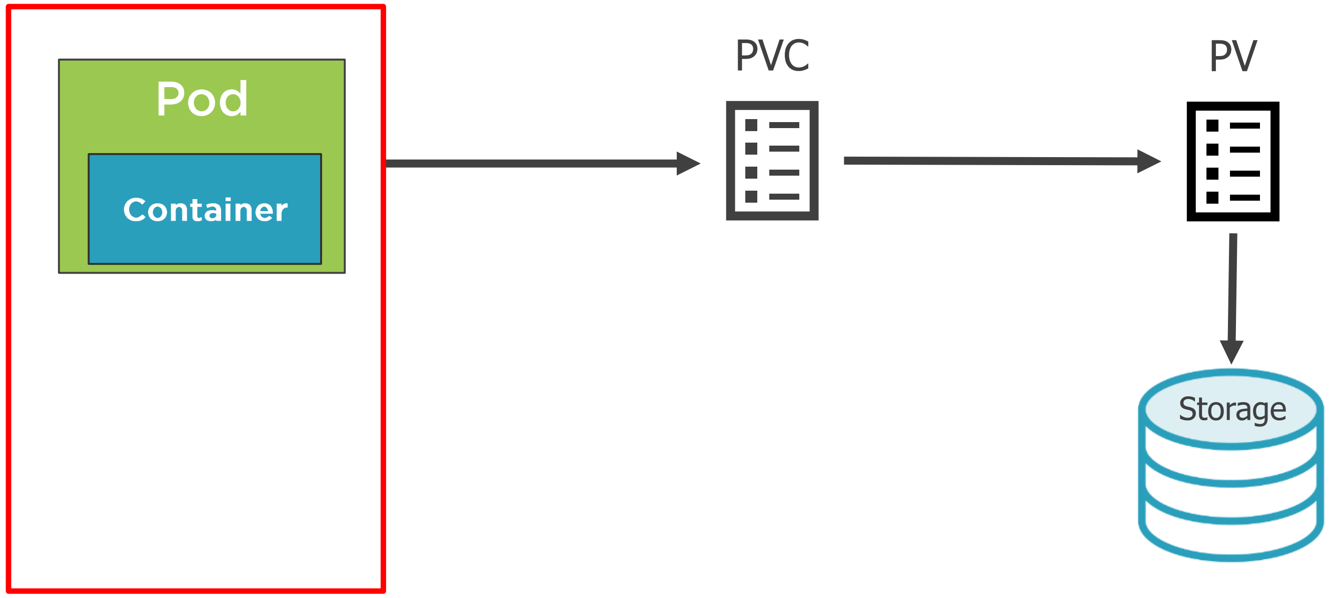
◀ Define access mode

◀ Request storage amount

Creating a PersistentVolumeClaim



Defining a Volume that Uses a PVC



Using a PersistentVolumeClaim

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: pod-uses-account-hdd-5g
  labels:
    name: storage
spec:
  containers:
  - image: nginx
    name: az-c-01
    command:
    - /bin/sh
    - -c
    - while true; do echo $(date) >>
      /mnt/blobdisk/outfile; sleep 1; done
    volumeMounts:
    - name: blobdisk01
      mountPath: /mnt/blobdisk
  volumes:
  - name: blobdisk01
    persistentVolumeClaim:
      claimName: pv-dd-account-hdd-5g
```

◀ **Mount to Volume**

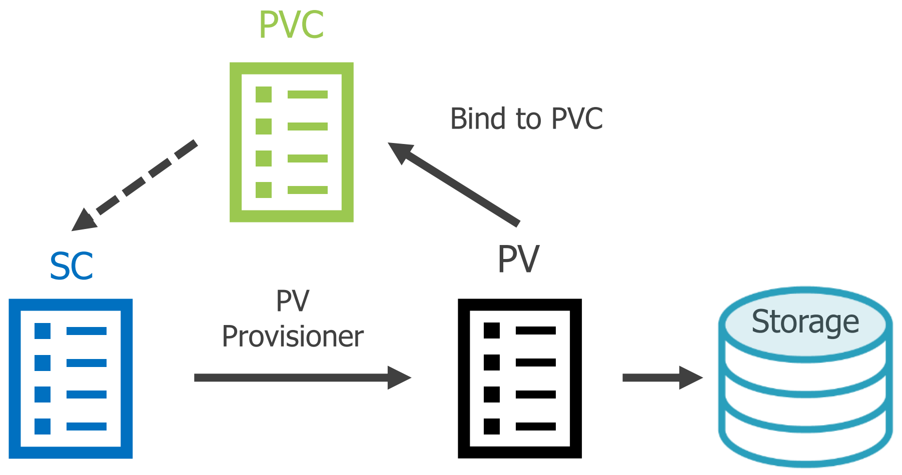
◀ **Create Volume that binds to PersistentVolumeClaim**

StorageClasses

A StorageClass (SC) is a type of storage template that can be used to dynamically provision storage.

PVC

StorageClass



Used to define different "classes" of storage

Act as a type of storage template

Supports dynamic provisioning of PersistentVolumes

Administrators don't have to create PVs in advance

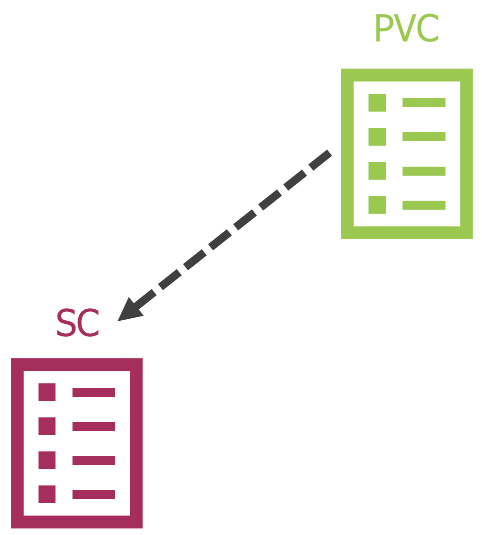
StorageClass Workflow

1 **Create Storage Class**

SC

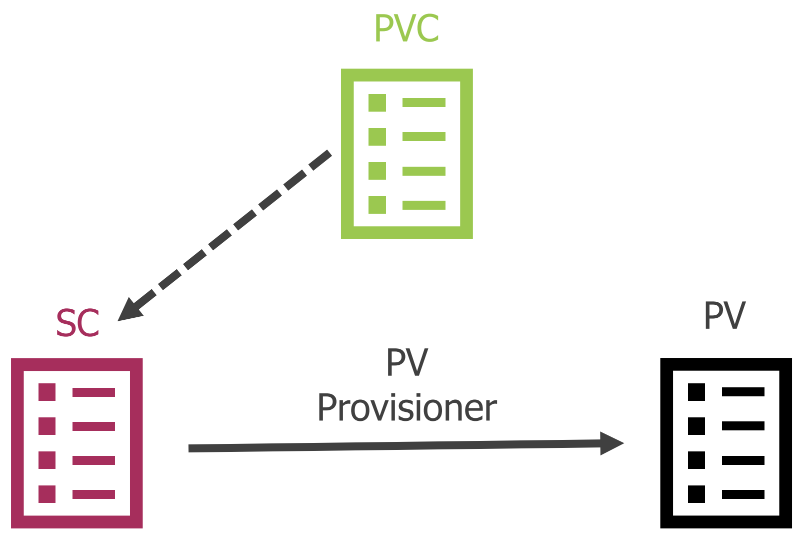
StorageClass Workflow

2 **Create PersistentVolumeClaim that references StorageClass**



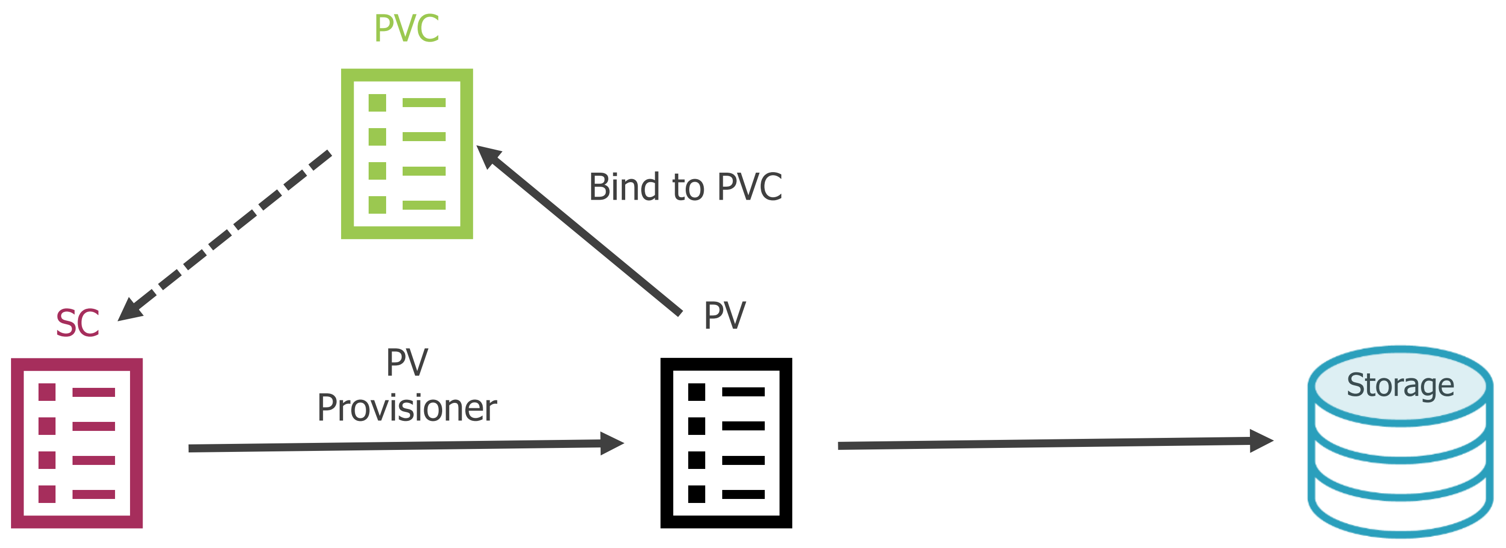
StorageClass Workflow

3 **Kubernetes uses StorageClass provisioner to provision a PersistentVolume**



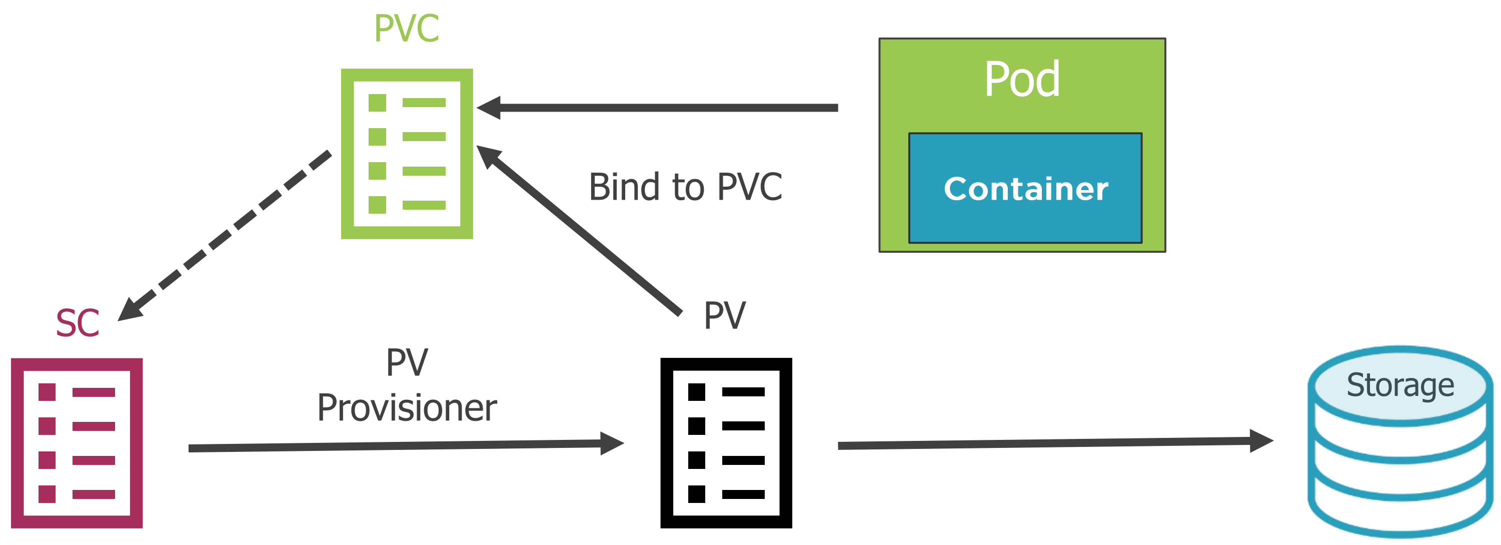
StorageClass Workflow

4 **Storage provisioned, PersistentVolume created and bound to PersistentVolumeClaim**



StorageClass Workflow

5 **Pod volume references PersistentVolumeClaim**



Defining a Local Storage StorageClass

```
apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

  name: local-storage

reclaimPolicy: Retain

provisioner: kubernetes.io/no-provisioner

volumeBindingMode: WaitForFirstConsumer
```

- API version
- A StorageClass resource
- Retain storage or Delete (default) after PVC is released
- Provisioner (volume plugin) that will be used to create PersistentVolume resource.
- Wait to create until Pod making PVC is created. Default is Immediate (create once PVC is created)

Defining a Local Storage PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 10Gi
  volumeMode: Block
  accessModes:
  - ReadWriteOnce
  storageClassName: local-storage
  local:
    path: /data/storage
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - <node-name>
```

- One client can mount for read/write
- Reference StorageClass
- Path where data is stored on Node
- Select the Node where the local storage PV is created

Defining a PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: local-storage
  resources:
    requests:
      storage: 1Gi
```

◀ Define a PersistentVolumeClaim (PVC)

◀ Access Mode and storage classification PV needs to support

◀ Storage request information

Using a PersistentVolumeClaim

```
apiVersion: apps/v1
kind: [Pod | StatefulSet | Deployment]
...
  spec:
    volumes:
    - name: my-volume
      persistentVolumeClaim:
        claimName: my-pvc
```

- Define a Volume
- Use a PVC to claim the required storage

PersistentVolumes in Action

Summary

Kubernetes supports several different types of storage:

- Ephemeral storage (emptyDir)
- Persistent storage (many options)
- PersistentVolumes, PersistentVolumeClaims, and StorageClasses
- ConfigMaps (key/value pairs)
- Secrets