



# Kubernetes Volumes



## Table of Contents

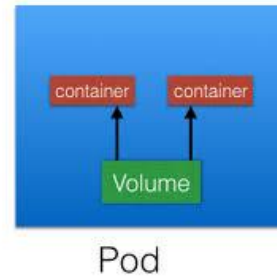


- ▶ Volumes
- ▶ Volume Types
- ▶ PersistentVolumes
- ▶ PersistentVolumeClaims
- ▶ The interaction between PVs and PVCs



1

# Volumes



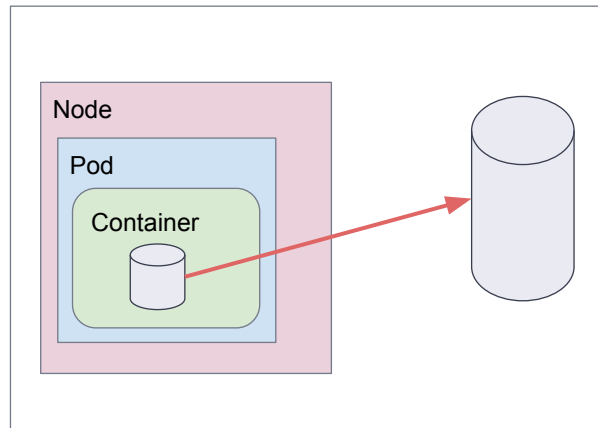
## Volumes

- on-disk files in a Container are ephemeral.
- All data stored inside a container is deleted if the container crashes.
- When a Container crashes, kubelet will restart it, but the files will be lost which means that it will not have any of the old data.
- To overcome this problem, Kubernetes uses **Volumes**. A Volume is essentially a directory backed by a storage medium. The storage medium, content and access mode are determined by the Volume Type.



# Volumes

A **volume** can be thought of as a directory which is accessible to the containers in a pod.

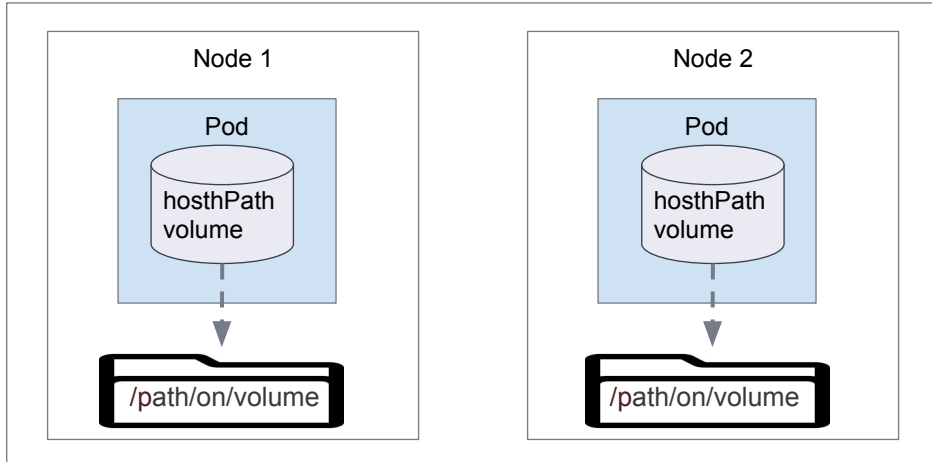


## 2 Volume Types



## Volume Types

- **hostPath:** A hostPath volume mounts a file or directory from the host node's filesystem into your Pod.



## Volume Types

- **emptyDir:** An emptyDir volume is first created when a Pod is assigned to a Node and exists as long as that Pod is running on that node.

**Note:** A container crashing does *not* remove a Pod from a node. The data in an **emptyDir** volume is safe across container crashes.

Some uses for an emptyDir are:

- checkpointing a long computation for recovery from crashes
- as a cache (holding files that a content-manager container fetches while a webserver container serves the data)



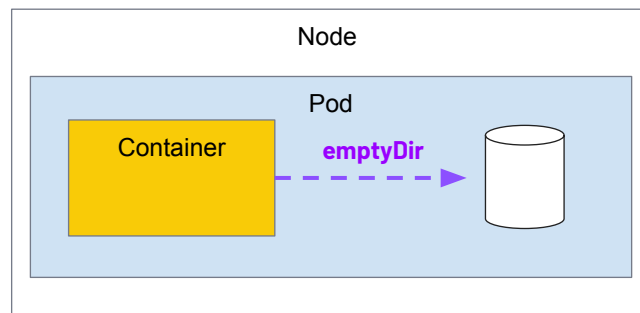
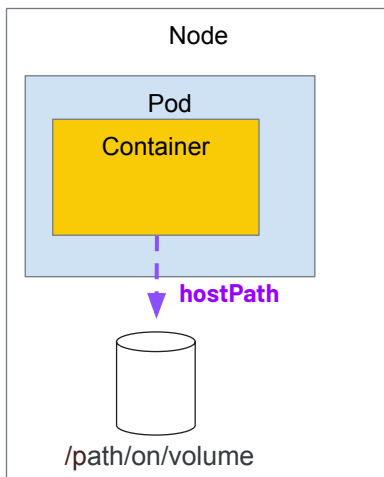
# Volume Types

- ```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```



# Volume Types

- hostPath vs emptyDir**





# Volume Types

- **Secret:** A secret volume is used to pass sensitive information, such as passwords, to Pods.
- **configMap:** The configMap resource provides a way to inject configuration data, or shell commands and arguments into a Pod.
- **persistentVolumeClaim:** A persistentVolumeClaim volume is used to mount a persistentVolume into a Pod.



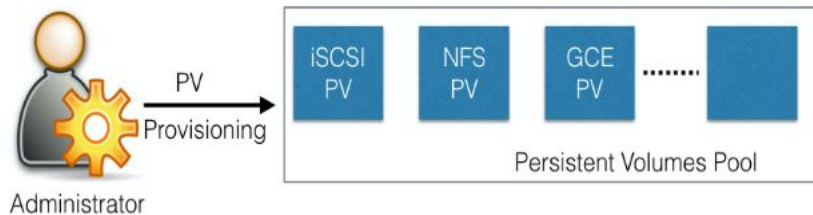
3

## PersistentVolumes



# PersistentVolumes

A **PersistentVolume (PV)** is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes.



# PersistentVolumes

Each PV gets its own set of **access modes** describing that specific PV's capabilities. There are four access modes:

- **ReadWriteOnce** (read-write by a single node)
- **ReadOnlyMany** (read-only by many nodes)
- **ReadWriteMany** (read-write by many nodes).
- **ReadWriteOncePod** (read-write only one pod in the cluster)

Once a suitable **PersistentVolume** is found, it is bound to a **PersistentVolumeClaim**.



4

## PersistentVolumeClaims



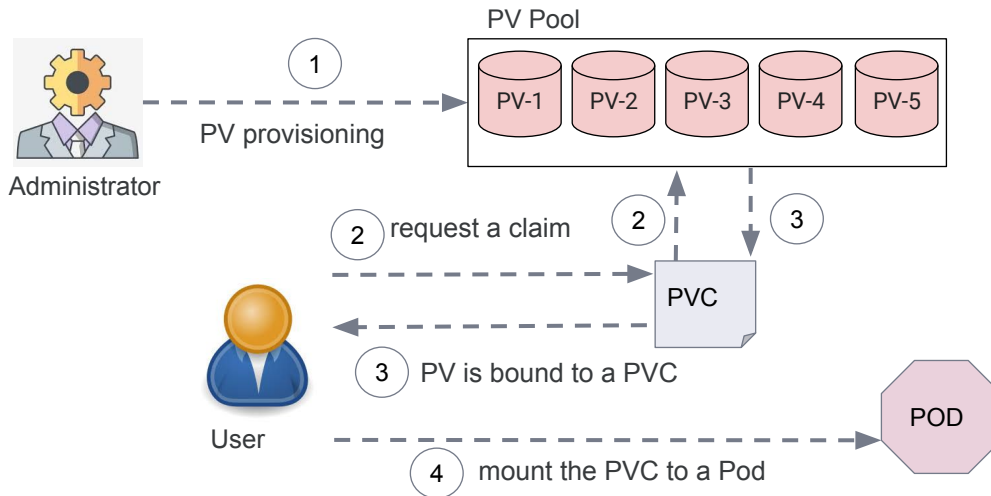
## PersistentVolumeClaims

- A **PersistentVolumeClaim (PVC)** is a request for storage by a user.
- It is similar to a Pod. Pods consume node resources and **PVCs** consume **PV resources**.
- Pods can request specific levels of resources (CPU and Memory). Claims can request **specific size** and **access modes**.
- Once a suitable **PersistentVolume** is found, it is bound to a **PersistentVolumeClaim**.





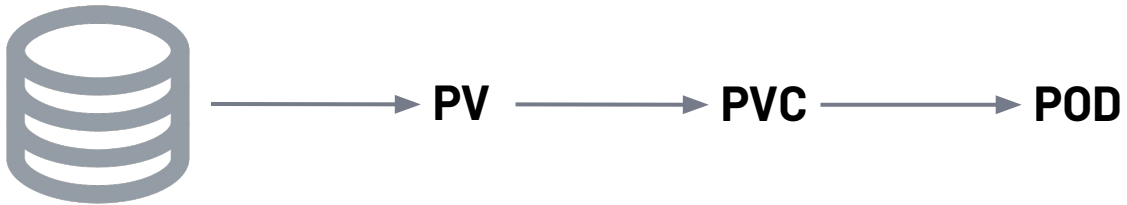
# PersistentVolumeClaims



5

## The interaction between PVs, PVCs and Pods

## ▶ The interaction between PVs, PVCs and Pods ▶▶



## ▶ The interaction between PVs and PVCs ▶▶

### **Provisioning**

There are two ways PVs may be provisioned: statically or dynamically.

### **Static**

A cluster administrator creates a number of PVs. They carry the details of the real storage, which is available for use by cluster users. They exist in the Kubernetes API and are available for consumption.

### **Dynamic**

When none of the static PVs the administrator created match a user's PersistentVolumeClaim, the cluster may try to dynamically provision a volume specially for the PVC. This provisioning is based on **StorageClasses**.



# The interaction between PVs and PVCs

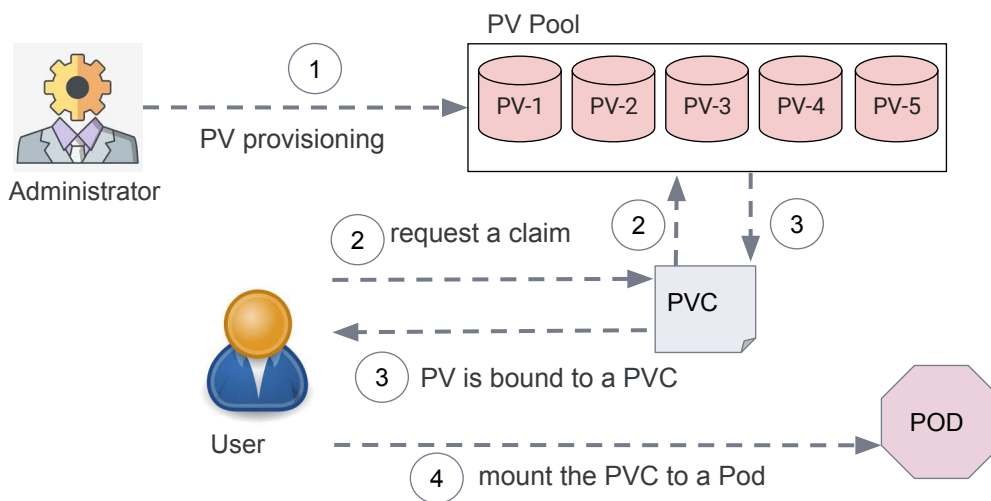
**Static**



**Dynamic**



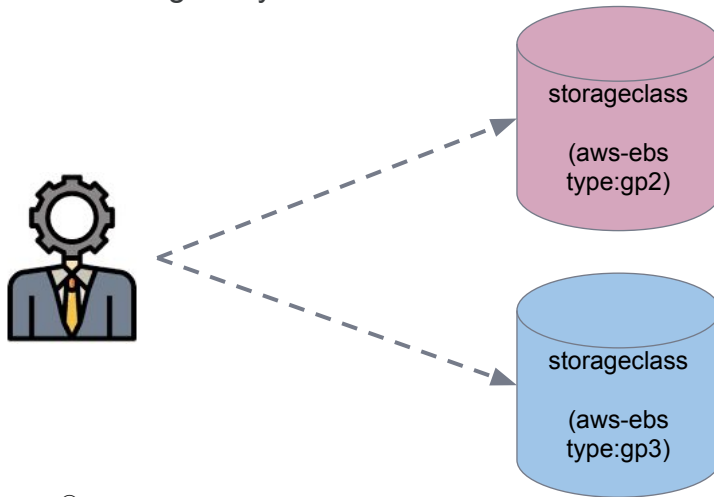
## Static PV Provisioning



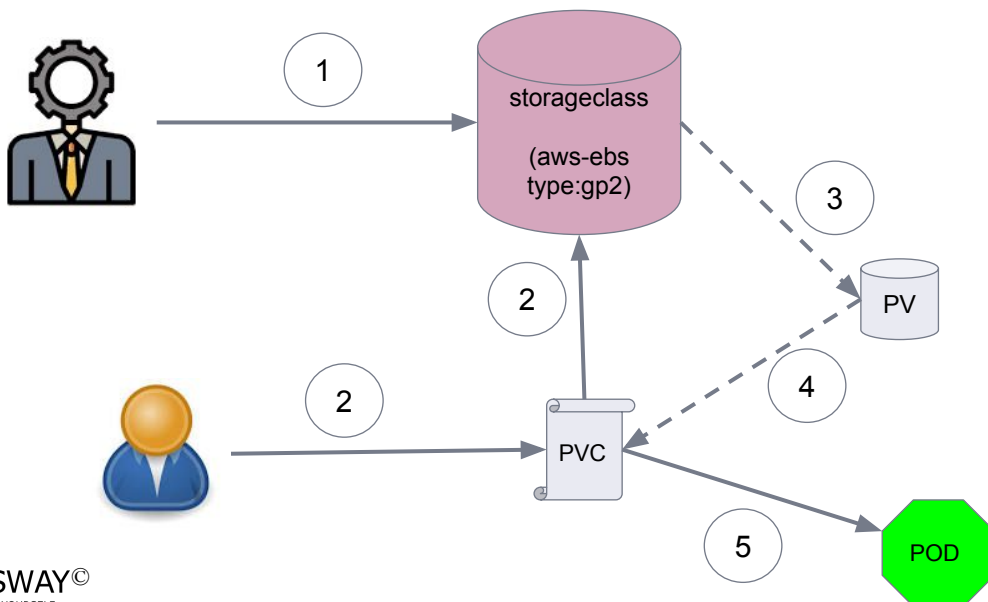


# Dynamic PV Provisioning

A **StorageClass** provides a way for administrators to describe the "classes" of storage they offer.



# Dynamic PV Provisioning



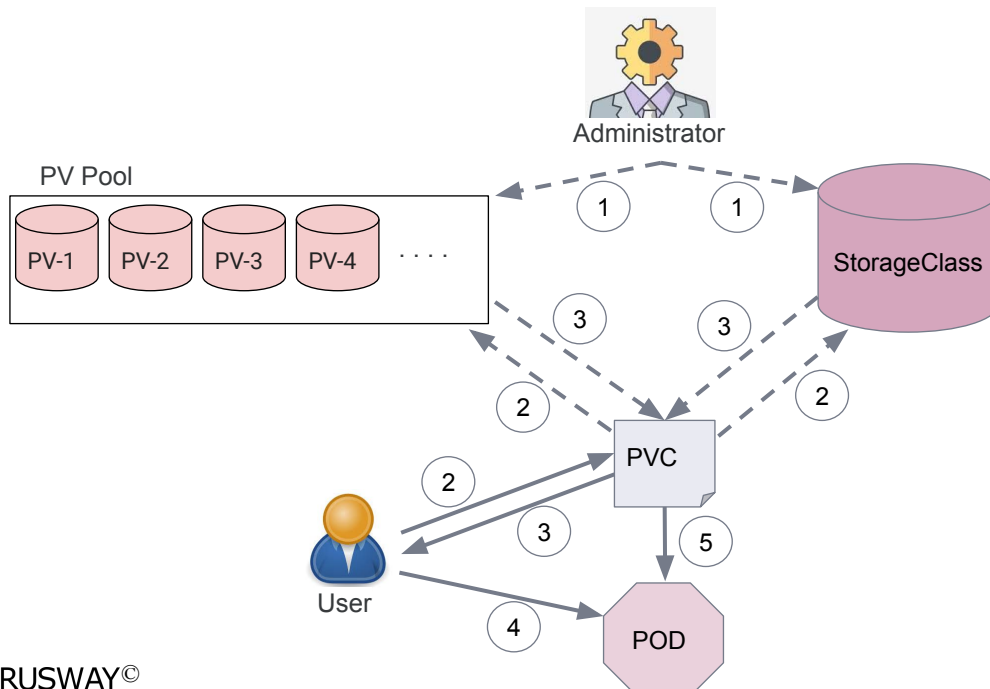


# Storage Class

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: aws-standard
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
parameters:
  type: gp2
  fsType: ext4
```

**Provisioner:** Each StorageClass has a provisioner that determines what volume plugin is used for provisioning PVs.

**Parameters:** Storage Classes have parameters that describe volumes belonging to the storage class. Different parameters may be accepted depending on the provisioner





# THANKS!

## Any questions?

You can find me at:

► [james@clarusway.com](mailto:james@clarusway.com)



CLARUSWAY<sup>®</sup>  
REINVENT YOURSELF

Students, write your response!

Pear Deck Interactive Slide  
Do not remove this bar

2  
7