A CLOUD GURU

K8s Storage Overview

Will Boyd

DEVOPS TRAINING ARCHITECT

LESSON BREAKDOWN

K8s Storage Overview

Container File Systems

Volumes

Persistent Volumes

Volume Types

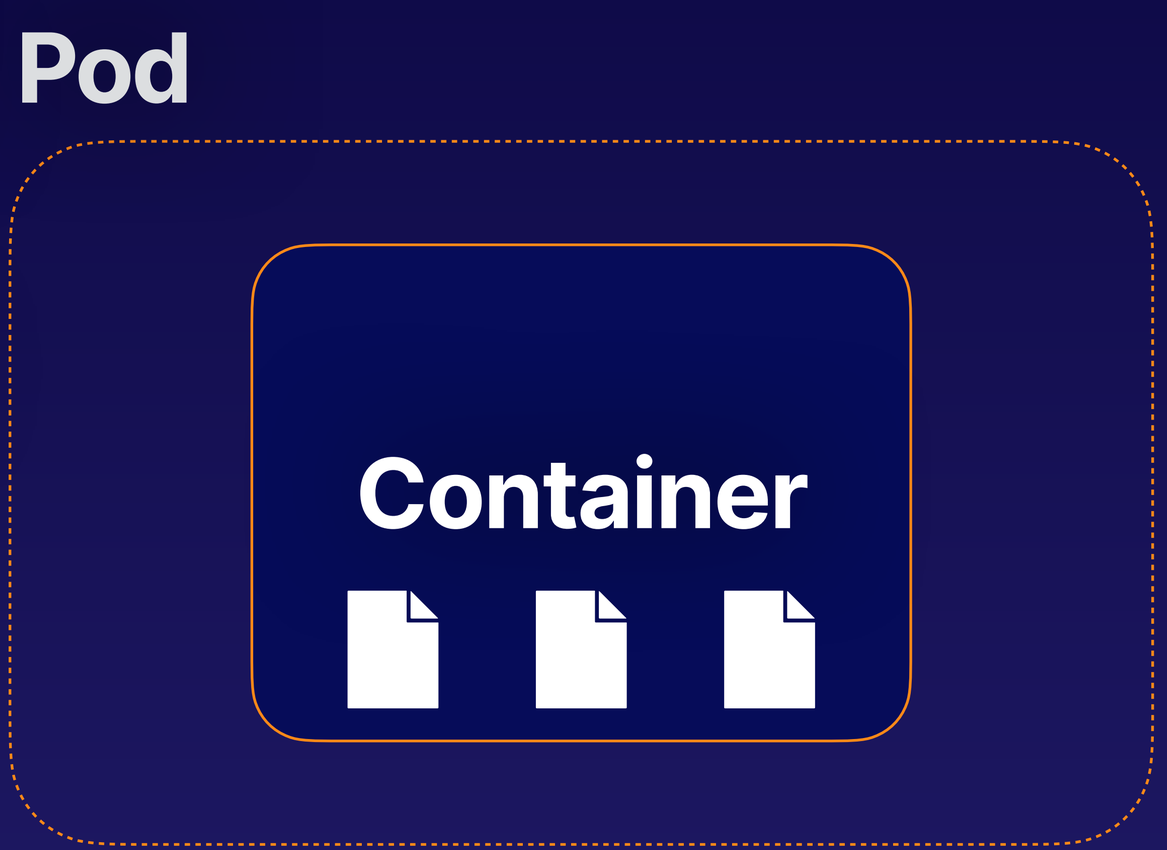
Will Boyd
DEVOPS TRAINING ARCHITECT

A CLOUD GURU

Container File Systems

The container file system is **ephemeral**. Files on the container's file system exist only as long as the container exists.

If a container is deleted or re-created in K8s, data stored on the container file system is lost.

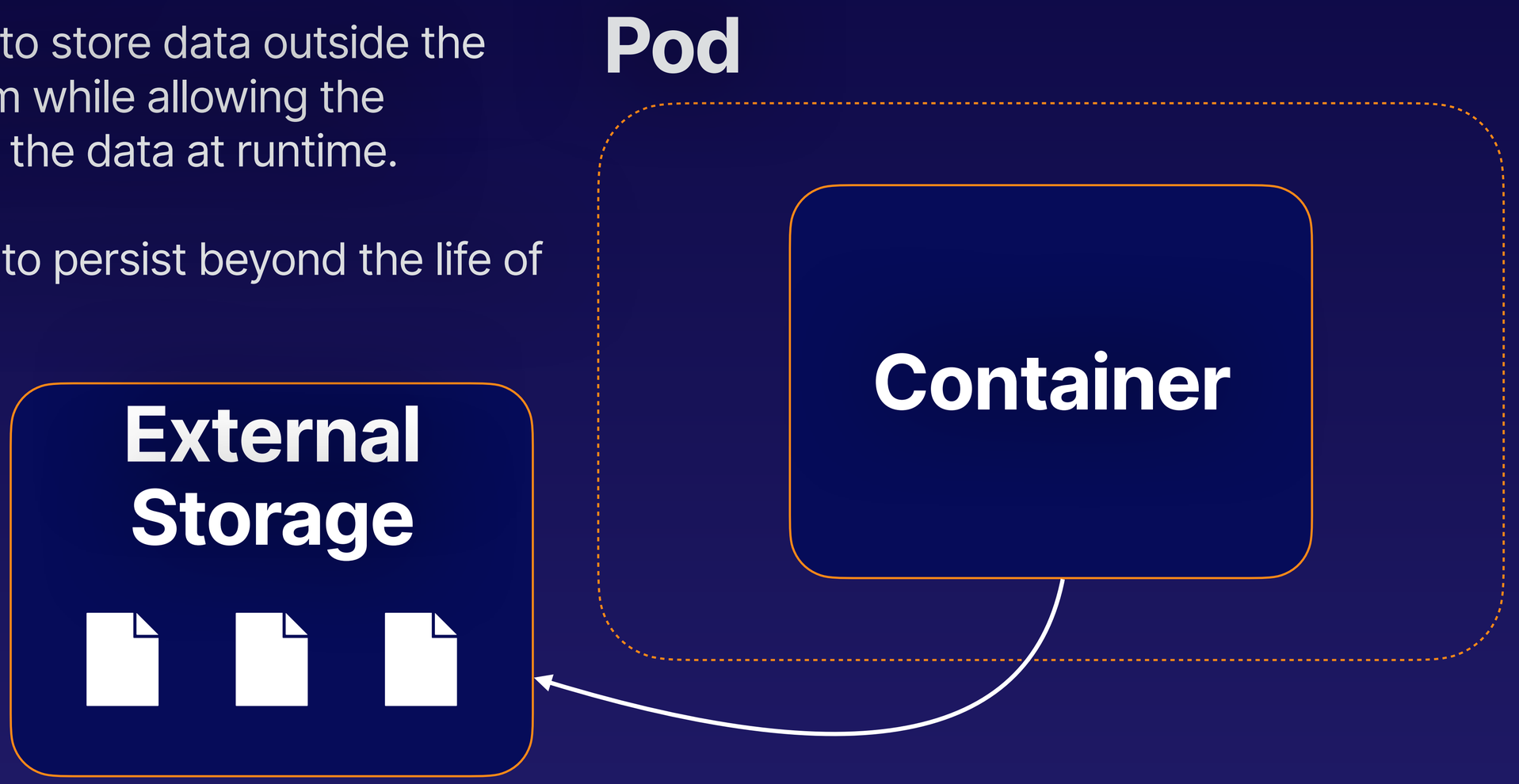


Volumes

Many applications need a more persistent method of data storage.

Volumes allow you to store data outside the container file system while allowing the container to access the data at runtime.

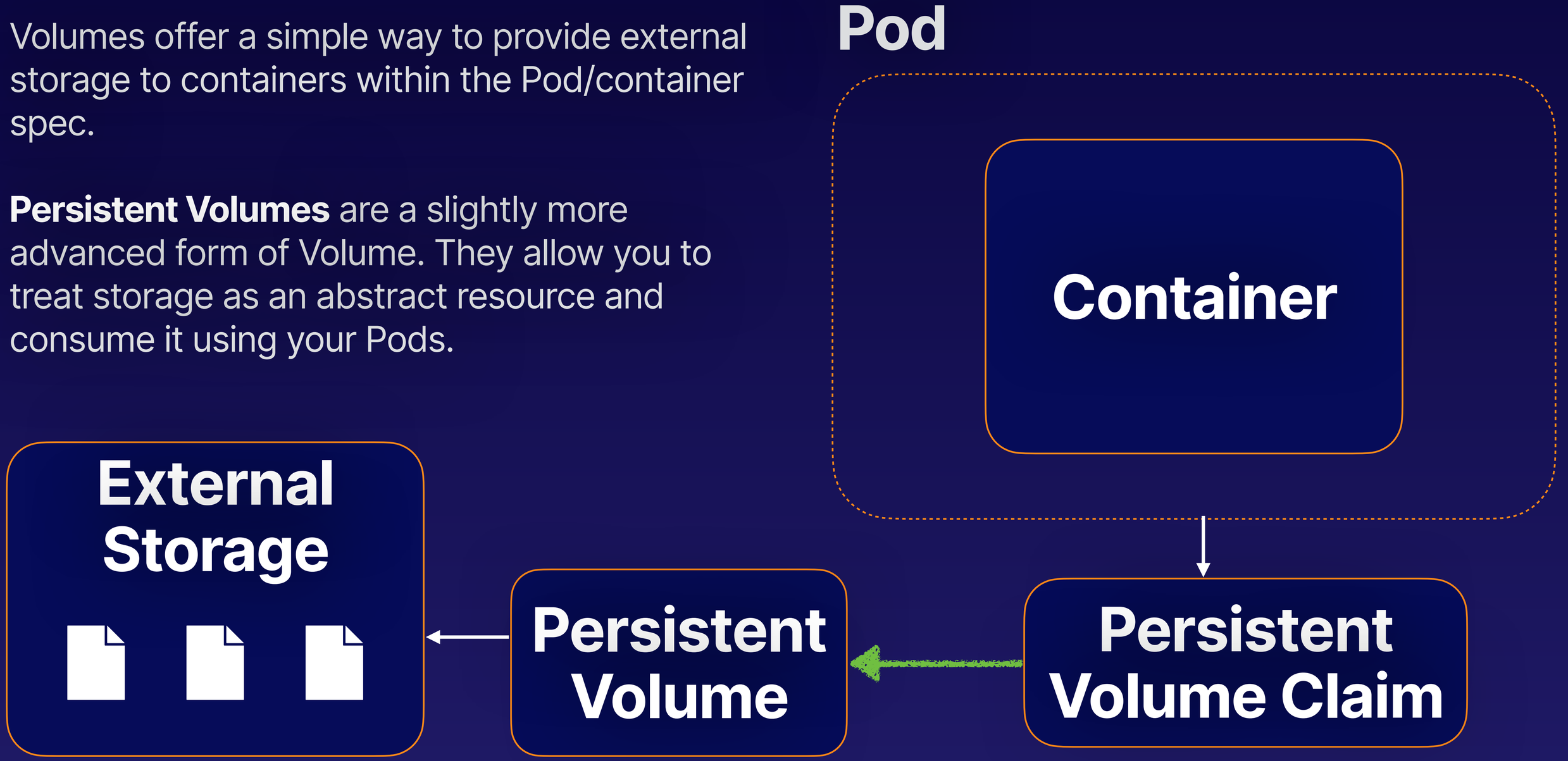
This can allow data to persist beyond the life of the container.

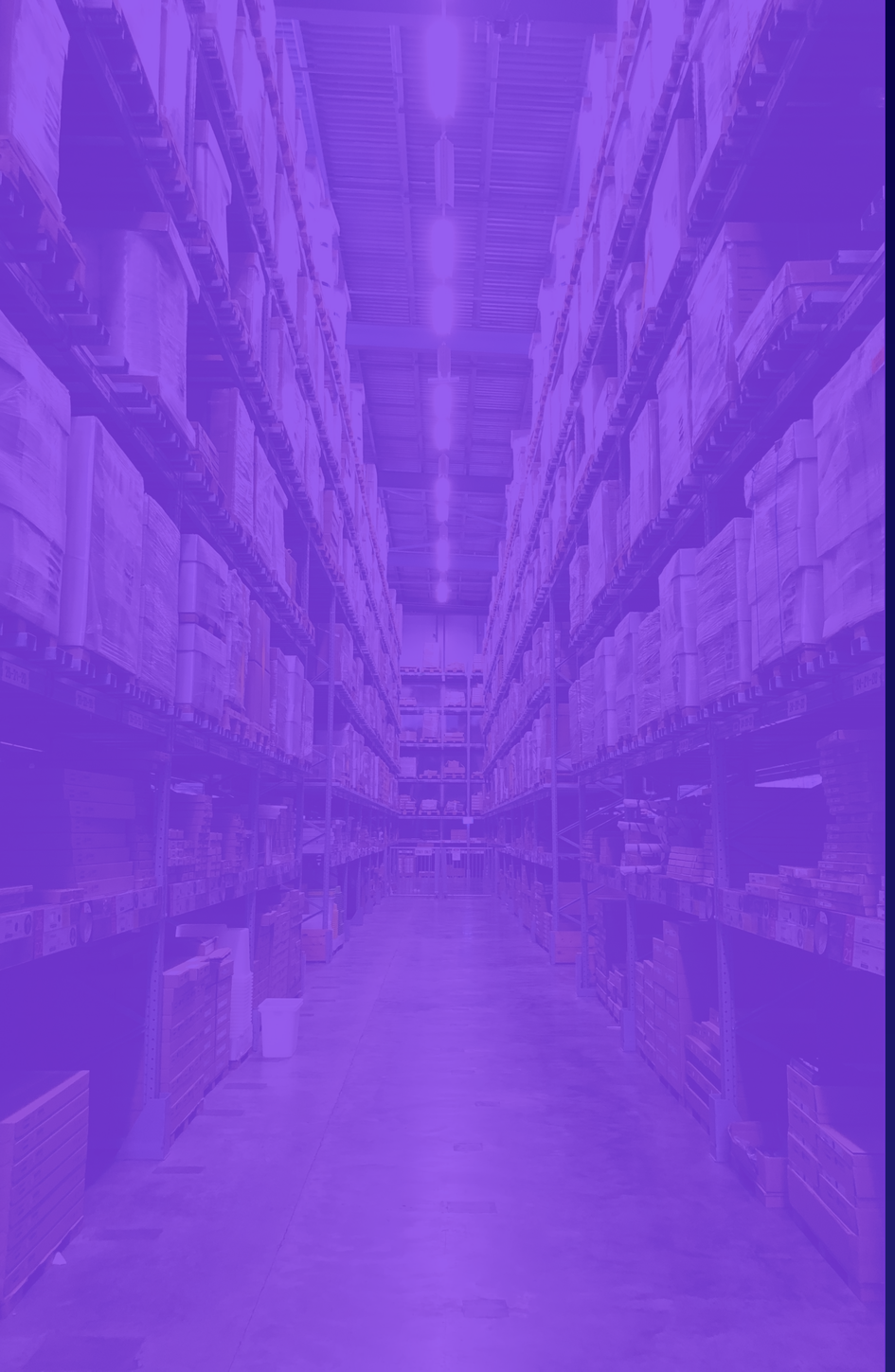


Persistent Volumes

Volumes offer a simple way to provide external storage to containers within the Pod/container spec.

Persistent Volumes are a slightly more advanced form of Volume. They allow you to treat storage as an abstract resource and consume it using your Pods.





A CLOUD GURU

Volume Types

Both Volumes and Persistent Volumes each have a **volume type**. The volume type determines how the storage is actually handled.

Various volume types support storage methods such as:

- NFS
- Cloud storage mechanisms (AWS, Azure, GCP)
- ConfigMaps and Secrets
- A simple directory on the K8s node

LESSON BREAKDOWN

K8s Storage Overview

Container File Systems

Volumes

Persistent Volumes

Volume Types

Will Boyd
DEVOPS TRAINING ARCHITECT

A CLOUD GURU

A CLOUD GURU

Using K8s Volumes

Will Boyd

DEVOPS TRAINING ARCHITECT

LESSON BREAKDOWN

Using K8s Volumes

- Volumes and volumeMounts
- Sharing Volumes between Containers
- Common Volume Types
- Hands-On Demonstration

Will Boyd
DEVOPS TRAINING ARCHITECT

A CLOUD GURU

Volumes and volumeMounts

Regular Volumes can be set up relatively easily within a Pod/container specification.

volumes: In the Pod spec, these specify the storage volumes available to the Pod. They specify the volume type and other data that determines where and how the data is actually stored.

volumeMounts: In the container spec, these reference the volumes in the Pod spec and provide a `mountPath` (the location on the file system where the container process will access the volume data).

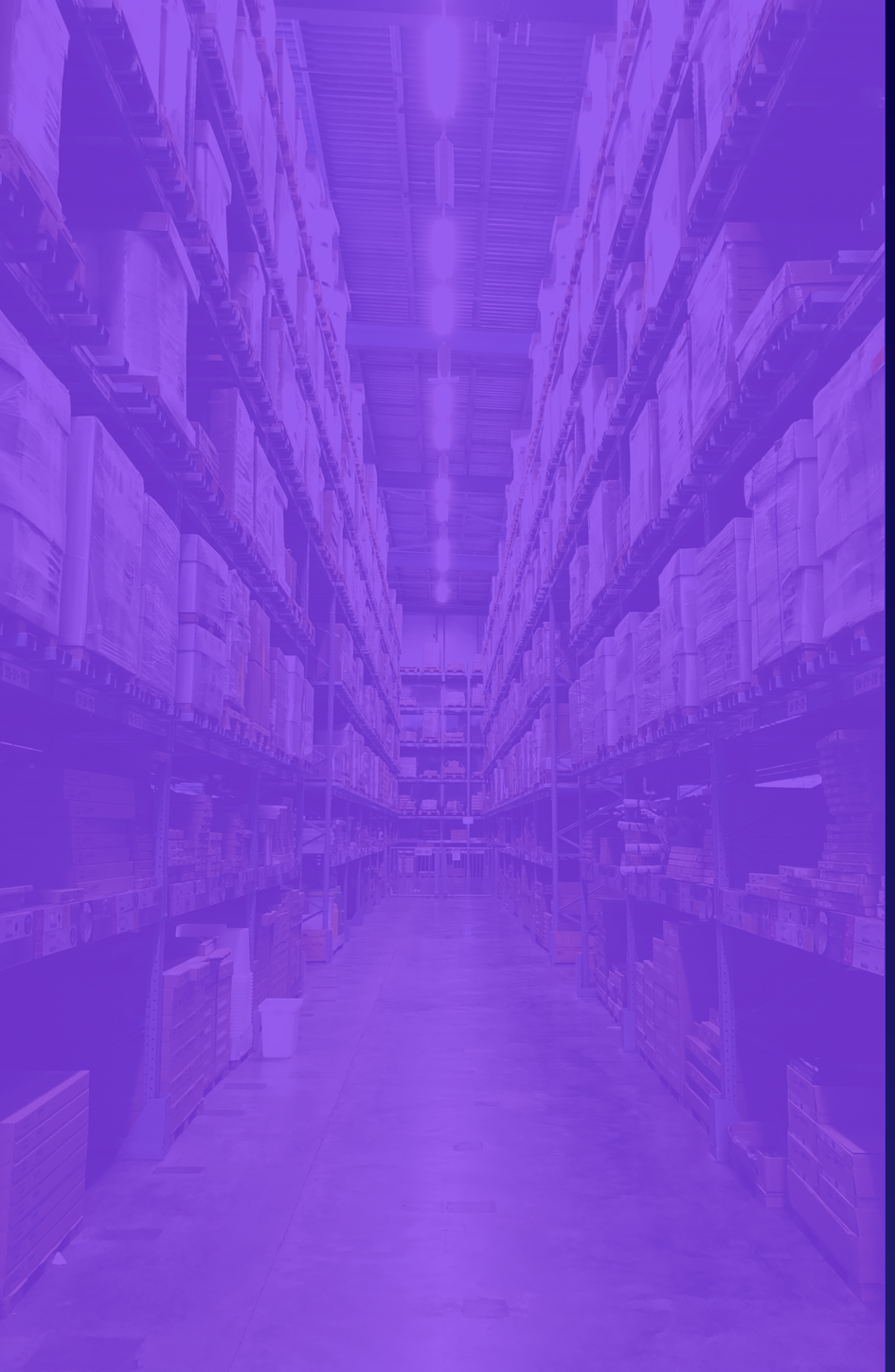
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-pod
spec:
  containers:
  - name: busybox
    image: busybox
    volumeMounts:
    - name: my-volume
      mountPath: /output
  volumes:
  - name: my-volume
    hostPath:
      path: /data
```

Sharing Volumes Between Containers

You can use `volumeMounts` to mount the same volume to multiple containers within the same Pod.

This is a powerful way to have multiple containers interact with one another. For example, you could create a secondary sidecar container that processes or transforms output from another container.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-pod
spec:
  containers:
  - name: busybox1
    image: busybox
    volumeMounts:
    - name: my-volume
      mountPath: /output
  - name: busybox2
    image: busybox
    volumeMounts:
    - name: my-volume
      mountPath: /input
  volumes:
  - name: my-volume
    emptyDir: {}
```



A CLOUD GURU

Common Volume Types

There are many volume types, but there are two you may want to be especially aware of.

hostPath: Stores data in a specified directory on the K8s node.

emptyDir: Stores data in a dynamically created location on the node. This directory exists only as long as the Pod exists on the node. The directory and the data are deleted when the Pod is removed. This volume type is very useful for simply sharing data between containers in the same Pod.

Hands-On Demonstration

LESSON BREAKDOWN

Using K8s Volumes

- Volumes and volumeMounts
- Sharing Volumes between Containers
- Common Volume Types
- Hands-On Demonstration

Will Boyd
DEVOPS TRAINING ARCHITECT

A CLOUD GURU

A CLOUD GURU

Exploring K8s Persistent Volumes

Will Boyd

DEVOPS TRAINING ARCHITECT

LESSON BREAKDOWN

Exploring K8s Persistent Volumes

PersistentVolumes

Storage Classes

PersistentVolumeClaims

Using a PersistentVolumeClaim in a Pod

Resizing a PersistentVolumeClaim

Will Boyd
DEVOPS TRAINING ARCHITECT

A CLOUD GURU

PersistentVolumes


```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: my-pv
spec:
  storageClassName: localdisk
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /var/output
```

PersistentVolumes are K8s objects that allow you to treat storage as an abstract resource to be consumed by Pods, much like K8s treats compute resources such as memory and CPU.

A PersistentVolume uses a set of attributes to describe the underlying storage resource (such as a disk or cloud storage location) which will be used to store data.

PersistentVolumes


```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: my-pv
spec:
  storageClassName: localdisk
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /var/output
```

PersistentVolumes are K8s objects that allow you to treat storage as an abstract resource to be consumed by Pods, much like K8s treats compute resources such as memory and CPU.

A PersistentVolume uses a set of attributes to describe the underlying storage resource (such as a disk or cloud storage location) which will be used to store data.

PersistentVolumes


```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: my-pv
spec:
  storageClassName: localdisk
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /var/output
```

PersistentVolumes are K8s objects that allow you to treat storage as an abstract resource to be consumed by Pods, much like K8s treats compute resources such as memory and CPU.

A PersistentVolume uses a set of attributes to describe the underlying storage resource (such as a disk or cloud storage location) which will be used to store data.

PersistentVolumes


```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: my-pv
spec:
  storageClassName: localdisk
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /var/output
```

PersistentVolumes are K8s objects that allow you to treat storage as an abstract resource to be consumed by Pods, much like K8s treats compute resources such as memory and CPU.

A PersistentVolume uses a set of attributes to describe the underlying storage resource (such as a disk or cloud storage location) which will be used to store data.

Storage Classes

Storage Classes allow K8s administrators to specify the types of storage services they offer on their platform.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: localdisk
provisioner: kubernetes.io/no-provisioner
```

Storage Classes


```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: slow
provisioner: kubernetes.io/no-provisioner
```

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/no-provisioner
```

For example, an administrator could create a StorageClass called **slow** to describe low-performance but inexpensive storage resources, and another called **fast** for high-performance but more costly resources.

This would allow users to choose storage resources that fit the needs of their applications.

allowVolumeExpansion

The **allowVolumeExpansion** property of a StorageClass determines whether or not the StorageClass supports the ability to resize volumes after they are created.

If this property is not set to true, attempting to resize a volume that uses this StorageClass will result in an error.

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: localdisk
provisioner: kubernetes.io/no-provisioner
allowVolumeExpansion: true
```

reclaimPolicies

A PersistentVolume's **persistentVolumeReclaimPolicy** determines how the storage resources can be reused when the PersistentVolume's associated PersistentVolumeClaims are deleted.

- **Retain:** Keeps all data. This requires an administrator to manually clean up the data and prepare the storage resource for reuse.
- **Delete:** Deletes the underlying storage resource automatically (only works for cloud storage resources).
- **Recycle:** Automatically deletes all data in the underlying storage resource, allowing the PersistentVolume to be reused.

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: my-pv
spec:
  storageClassName: localdisk
  persistentVolumeReclaimPolicy: Recycle
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /etc/output
```

PersistentVolumeClaims

A **PersistentVolumeClaim** represents a user's request for storage resources. It defines a set of attributes similar to those of a PersistentVolume (StorageClass, etc.).

When a PersistentVolumeClaim is created, it will look for a PersistentVolume that is able to meet the requested criteria. If it finds one, it will automatically be **bound** to the PersistentVolume.

Persistent Volume

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  storageClassName: localdisk
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

Using a PersistentVolumeClaim in a Pod


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pv-pod
spec:
  containers:
  - name: busybox
    image: busybox
    volumeMounts:
    - name: pv-storage
      mountPath: /output
  volumes:
  - name: pv-storage
    persistentVolumeClaim:
      claimName: my-pvc
```

PersistentVolumeClaims can be **mounted** to a Pod's containers just like any other volume.

If the PersistentVolumeClaim is bound to a PersistentVolume, the containers will use the underlying PersistentVolume storage.

Resizing a PersistentVolumeClaim


```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  storageClassName: localdisk
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 200Mi
```

You can **expand** PersistentVolumeClaims without interrupting applications that are using them.

Simply edit the **spec.resources.requests.storage** attribute of an existing PersistentVolumeClaim, increasing its value.

However, the StorageClass must support resizing volumes and must have **allowVolumeExpansion** set to **true**.

LESSON BREAKDOWN

Exploring K8s Persistent Volumes

PersistentVolumes

Storage Classes

PersistentVolumeClaims

Using a PersistentVolumeClaim in a Pod

Resizing a PersistentVolumeClaim

Will Boyd
DEVOPS TRAINING ARCHITECT

A CLOUD GURU

A CLOUD GURU

Using K8s Persistent Volumes

Will Boyd

DEVOPS TRAINING ARCHITECT

Hands-On Demonstration