KPLABS Course

Docker Certified Associate 2020

Storage and Volumes

ISSUED BY
Zeal Vora

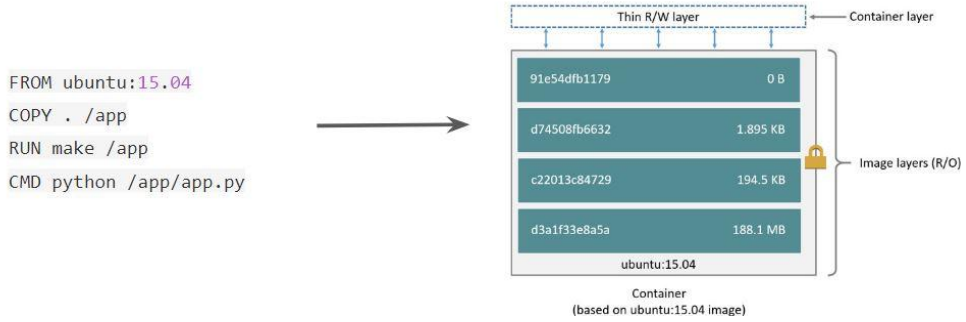
REPRESENTATIVE
instructors@kplabs.in

Module 1: Storage Drivers

A Docker image is built up from a series of layers.

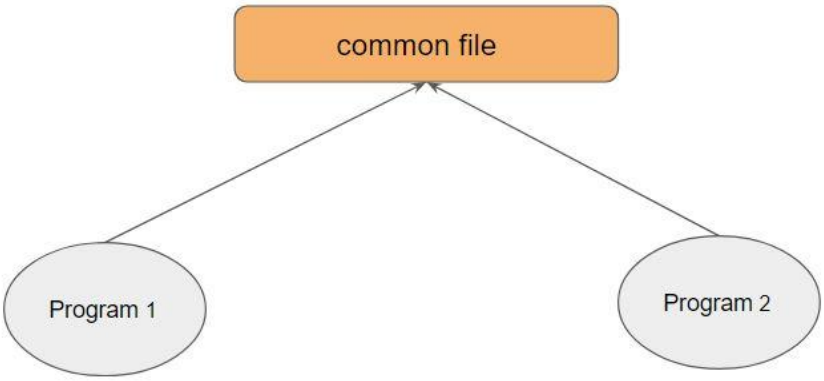
Storage Driver piece all these things together for you

There are multiple storage drivers available with different trade-offs..



It is also important for us to understand the copy-on-write (CoW) strategy

Copy on write" means: everyone has a single shared copy of the same data until it's written, and then a copy is made.



There are various storage drivers available in Docker, these includes:

- AUFS
- Device Mapper
- OverlayFS
- ZFS
- VFS
- Brtfs

Module 2: Block vs Object Storage

In block storage, the data is stored in terms of blocks

Data stored in blocks are normally read or written entirely a whole block at the same time

Most of the file systems are based on block devices.

2.1 Block Storage:

Every block has an address and the application can be called via SCSI call via its address

There is no storage side meta-data associated with the block except the address.

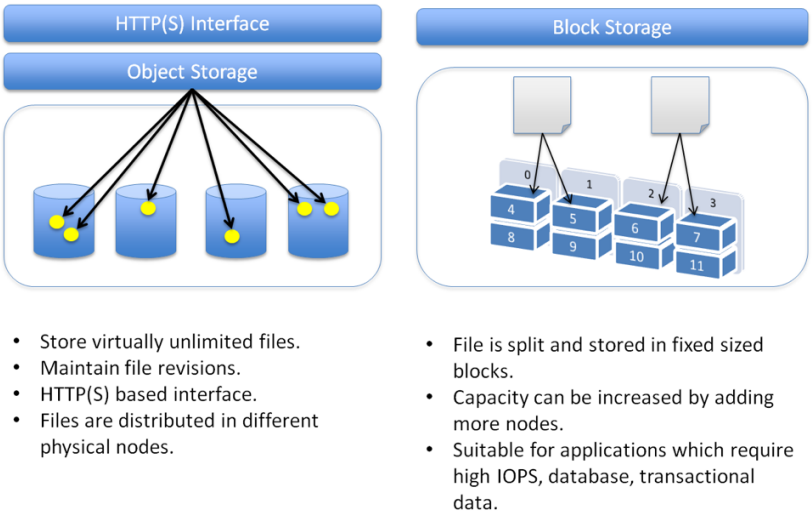
Thus block has no description, no owner

2.2 Object Storage:

Object storage is a data storage architecture that manages data as objects as opposed to blocks of storage.

An object is defined as a data (file) along with all its meta-data which is combined together as an object.

This object is given an ID which is calculated from the content of the object (from the data and metadata). The application can then call the object with the unique object ID.



- Store virtually unlimited files.
- Maintain file revisions.
- HTTP(S) based interface.
- Files are distributed in different physical nodes.

- File is split and stored in fixed sized blocks.
- Capacity can be increased by adding more nodes.
- Suitable for applications which require high IOPS, database, transactional data.

Module 3: Docker Volumes

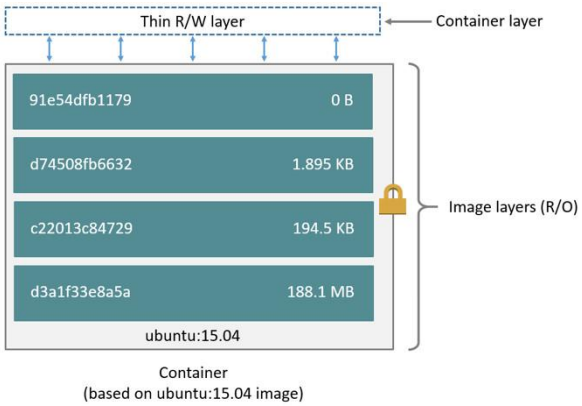
3.1 Challenges with files in Container Writable Layer

By default, all files created inside a container are stored on a writable container layer. This means that:

The data doesn't persist when that container no longer exists, and it can be difficult to get the data out of the container if another process needs it.

Writing into a container's writable layer requires a storage driver to manage the filesystem. The storage driver provides a union filesystem, using the Linux kernel.

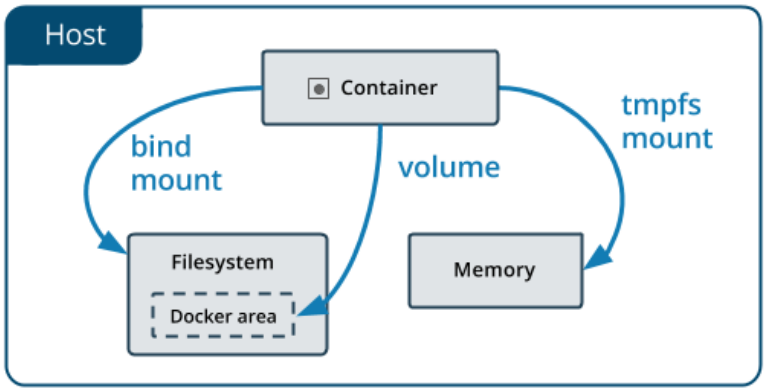
This extra abstraction reduces performance as compared to using data volumes, which write directly to the host filesystem.



3.2 Ideal Approach for Persistent Data

Docker has two options for containers to store files in the host machine, so that the files are persisted even after the container stops: volumes, and bind mounts.

If you're running Docker on Linux you can also use a tmpfs mount.



3.3 Important Pointers to Remember:

A given volume can be mounted into multiple containers simultaneously.

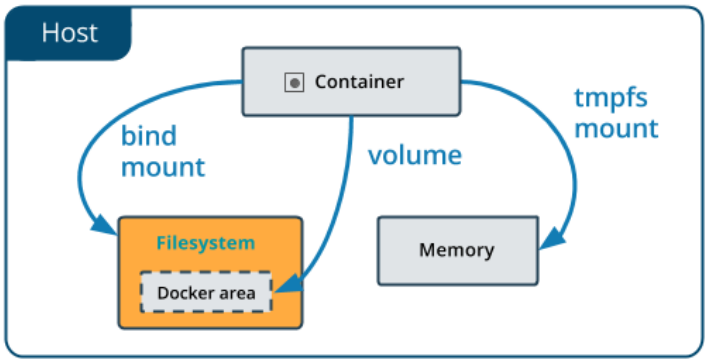
When no running container is using a volume, the volume is still available to Docker and is not removed automatically.

When you mount a volume, it may be named or anonymous. Anonymous volumes are not given an explicit name when they are first mounted into a container, so Docker gives them a random name that is guaranteed to be unique within a given Docker host.

Module 4: Bind Mount

When you use a bind mount, a file or directory on the host machine is mounted into a container.

The file or directory is referenced by its full or relative path on the host machine.



,

Module 5: Automatically Remove Volume on Container Removal

Whenever a container is created with -dt flag and if the main process completes/stops, then it goes into the Exited stage.

We can see list of all containers (Running/Exited) with docker ps -a command

Many a times, containers performs a job and we want container to automatically get deleted once it exits. This can be achieved with the --rm option.

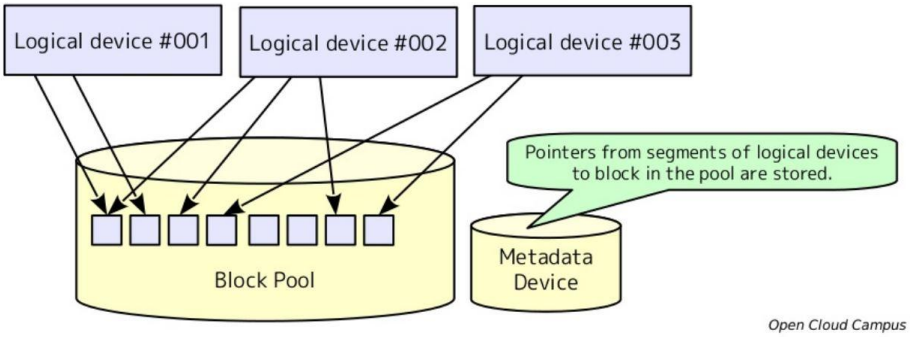
Module 6: Device Mapper

The device mapper is a framework provided by the Linux kernel for mapping physical block devices onto higher-level virtual block devices.

Device mapper works by passing data from a virtual block device, which is provided by the device mapper itself, to another block device.

Device mapper creates logical devices on top of physical block device and provides feature likes :-

RAID, Encryption, Cache and various others.



There are two modes for devicemapper storage driver:

i) loop-lvm mode:

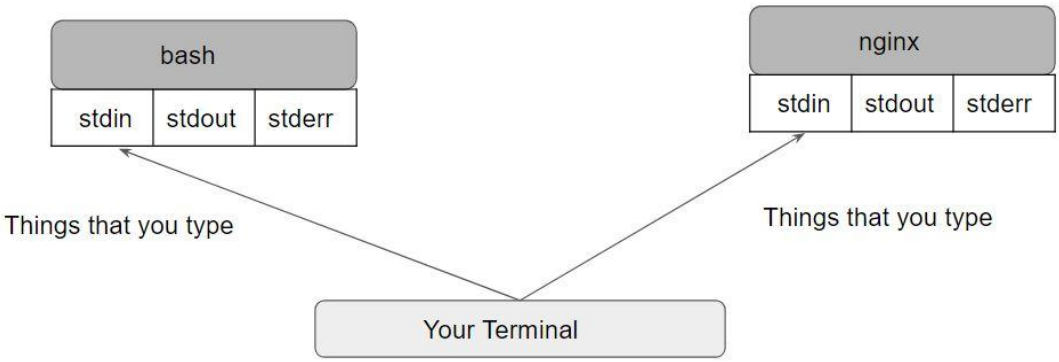
Should only be used in testing environment.
Makes use of loopback mechanism which is generally on the slower side.

ii) direct-lvm mode:

Production hosts using the devicemapper storage driver must use direct-lvm mode.
This is much more faster then the loop-lvm mode.

Module 7: Logging Driver

UNIX and Linux commands typically open three I/O streams when they run, called STDIN, STDOUT, and STDERR



There are a lot of logging driver options available in Docker, some of these include:

- json-file
- none
- syslog
- local
- journald
- splunk
- awslogs

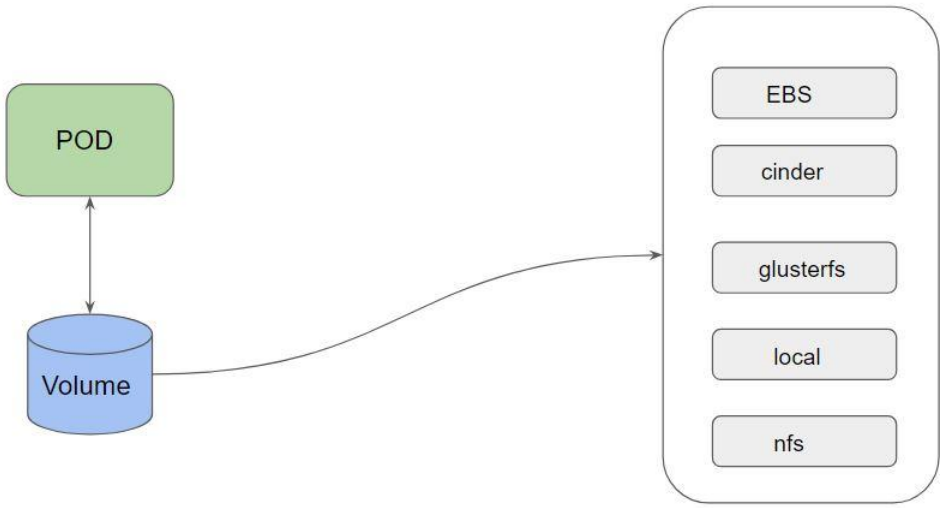
The docker logs command is not available for drivers other than json-file and journald.

Module 8: Kubernetes Volumes

On-disk files in a Container are ephemeral.

When there are multiple containers who wants to share same data, it becomes a challenge.

One of the benefits of Kubernetes is that it supports multiple types of volumes.



Module 9: PersistentVolume and PersistentVolumeClaim

9.1 PersistentVolume (PV)

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

Every Volume which is created can be of different type.

This can be taken care by the Storage Administrator / Ops Team

9.2 PersistentVolumeClaim (PVC)

A PersistentVolumeClaim is a request for the storage by a user.

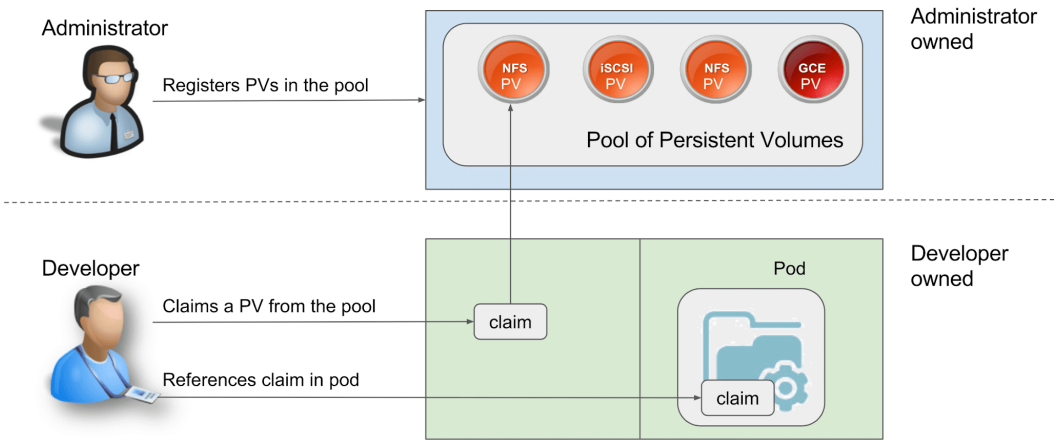
Within the claim, user need to specify the size of the volume along with access mode.

Developer:

I want a volume of size 10 GB which is has speed of Fast for my pod.

9.3 High-Level Working Steps:

- Storage Administrator takes care of creating PV.
- Developer can raise a “Claim” (I want a specific type of PV).
- Reference that claim within the PodSpec file.



Module 10: Volume Expansion in K8s

It can happen that your persistent volume has become full and you need to expand the storage for additional capacity.

Here are the high-level steps needed for volume expansion in Kubernetes.

Step 1 - Enable Volume Expansion

Volume Expansion must be enabled in the storage class.

Ensure that allowVolumeExpansion is set to true.

```
bash-4.2# kubectl describe storageclass do-block-storage
Name:            do-block-storage
IsDefaultClass:  Yes
Annotations:     kubectl.kubernetes.io/last-applied-configuration={"allowVolumeExpansion":true,"apiVersion":"storage.k
8s.io/v1","kind":"StorageClass","metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"},"name
":"do-block-storage"},"provisioner":"dobs.csi.digitalocean.com","reclaimPolicy":"Delete"}
,storageclass.kubernetes.io/is-default-class=true
Provisioner:           dobs.csi.digitalocean.com
Parameters:            <none>
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>
```

Step 2: Resize the PVC

Next step is to resize the PVC with the storage capacity that is needed.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"PersistentVolumeClaim",
ault"},"spec":{"accessModes":["ReadWriteOnce"],"resource
-storage"}}
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: dobs.
  creationTimestamp: "2020-08-30T07:08:20Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: csi-pvc
  namespace: default
  resourceVersion: "2154"
  selfLink: /api/v1/namespaces/default/persistentvolumec
  uid: 2ec4833a-bdb7-49b1-8c32-6b73aaafea04
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 15Gi
```

Step 3: Restart the POD

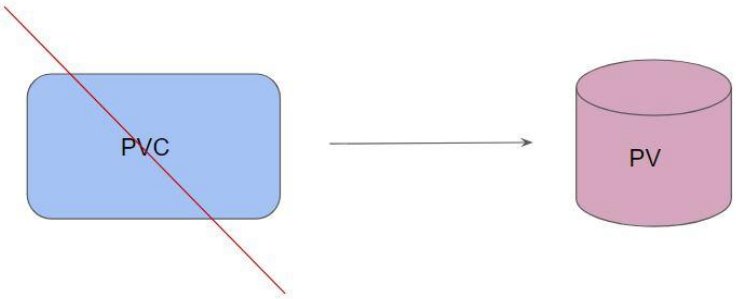
Once PVC object is modified, you will have to restart the POD.

kubectl delete pod [POD-NAME]

kubectl apply -f pod-manifest.yaml

Module 11: Reclaim Policy

The reclaim policy is responsible for what happens to the data in persistent volume when the kubernetes persistent volume claim has been deleted.



The following table shows the type of reclaim policies available:

Reclaim Policy	Description
Retain	When the PersistentVolumeClaim is deleted, the PersistentVolume still exists and the volume is considered "released"
Delete	The persistent volume is deleted when the claim is deleted.
Recycle	If supported by the underlying volume plugin, the Recycle reclaim policy performs a basic scrub (rm -rf /thevolume/*) on the volume and makes it available again for a new claim.

11.1 Retain Reclaim Policy

When PVC is deleted, the PersistentVolume still exists and the volume is considered "released".

It is not yet available for another claim because the previous claimant's data remains on the volume.

```
bash-4.2# kubectl get pv
kNAME                                      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS     CLAIM
 STORAGECLASS       REASON   AGE
pvc-41872ac3-55b7-412c-ac2c-a44e0a04fe33   5Gi        RWO            Retain           Released   default/kplabs-pvc
```

Steps for Reclaimation:

An administrator can manually reclaim the volume with the following steps.

Delete the PersistentVolume. The associated storage asset in external infrastructure (such as an AWS EBS, GCE PD, Azure Disk, or Cinder volume) still exists after the PV is deleted.

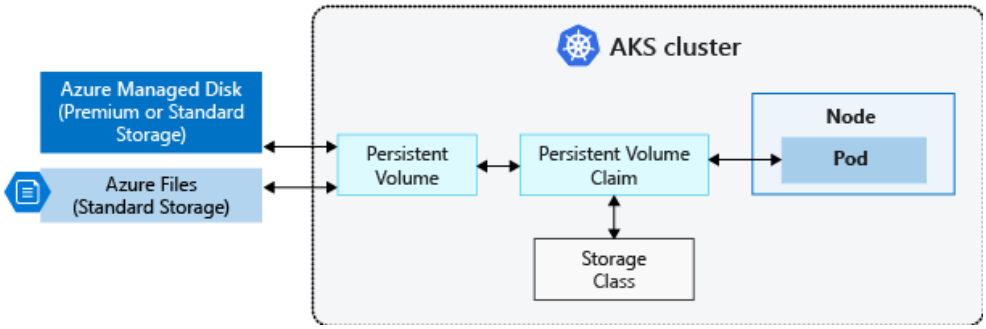
Manually clean up the data on the associated storage asset accordingly.

Manually delete the associated storage asset, or if you want to reuse the same storage asset, create a new PersistentVolume with the storage asset definition.

Module 12: Storage Classes

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

Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators.



12.1 Storage Class Resource

Each StorageClass contains the fields provisioner, parameters, and reclaimPolicy, which are used when a PersistentVolume belonging to the class needs to be dynamically provisioned.

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
allowVolumeExpansion: true
mountOptions:
  - debug
volumeBindingMode: Immediate
```

12.2 Understanding Provisioner:

Each StorageClass also has a provisioner that determines what volume plugin is used for provisioning PVs. This field must be specified.

AWSElasticBlockStore
AzureFile
Local
StorageOS
Glusterfs


```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
allowVolumeExpansion: true
mountOptions:
  - debug
volumeBindingMode: Immediate
```