Docker - Storage

Storage Drivers

Docker has multiple storage drivers that allow one to work with the underlying storage devices. The following table shows the different storage drivers along with the technology used for the storage drivers.

|  |  |
| --- | --- |
| **Technology** | **Storage Driver** |
| OverlayFS | overlay or overlay2 |
| AUFS | aufs |
| Btrfs | brtfs |
| Device Manager | devicemanager |
| VFS | vfs |
| ZFS | zfs |

Let us now discuss some of the instances in which you would use the various storage drivers −

AUFS

* This is a stable driver; can be used for production-ready applications.
* It has good memory usage and is good for ensuring a smooth Docker experience for containers.
* There is a high-write activity associated with this driver which should be considered.
* It’s good for systems which are of Platform as a service type work.

Devicemapper

* This is a stable driver; ensures a smooth Docker experience.
* This driver is good for testing applications in the lab.
* This driver is in line with the main Linux kernel functionality.

Btrfs

* This driver is in line with the main Linux kernel functionality.
* There is a high-write activity associated with this driver which should be considered.
* This driver is good for instances where you maintain multiple build pools.

Ovelay

* This is a stable driver and it is in line with the main Linux kernel functionality.
* It has a good memory usage.
* This driver is good for testing applications in the lab.

ZFS

* This is a stable driver and it is good for testing applications in the lab.
* It’s good for systems which are of Platform-as-a-Service type work.

To see the storage driver being used, issue the **docker info** command.

Syntax

docker info

Options

None

Return Value

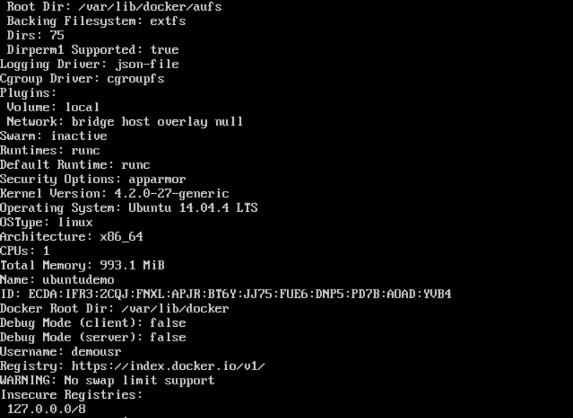
The command will provide all relative information on the Docker component installed on the Docker Host.

Example

sudo docker info

Output

The following output shows that the main driver used is the **aufs** driver and that the root directory is stored in **/var/lib/docker/aufs**.



Data Volumes

In Docker, you have a separate volume that can shared across containers. These are known as **data volumes**. Some of the features of data volume are −

* They are initialized when the container is created.
* They can be shared and also reused amongst many containers.
* Any changes to the volume itself can be made directly.
* They exist even after the container is deleted.

Let’s look at our Jenkins container. Let’s do a **docker inspect** to see the details of this image. We can issue the following command to write the output of the **docker inspect** command to a text file and then view the file accordingly.

sudo docker inspect Jenkins > tmp.txt

When you view the text file using the **more command**, you will see an entry as **JENKINS\_HOME=/var/Jenkins\_home**.

This is the mapping that is done within the container via the Jenkins image.



Now suppose you wanted to map the volume in the container to a local volume, then you need to specify the –v option when launching the container. An example is shown below −

sudo docker run –d –v /home/demo:/var/jenkins\_home –p 8080:8080 –p 50000:50000 jenkins

The –v option is used to map the volume in the container which is **/var/jenkins\_home** to a location on our Docker Host which is **/home/demo**.

V Option

Now if you go to the **/home/demo** location on your Docker Host after launching your container, you will see all the container files present there.



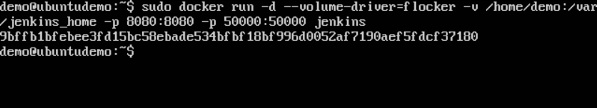
Changing the Storage Driver for a Container

If you wanted to change to the storage driver used for a container, you can do so when launching the container. This can be done by using the **–volume-driver** parameter when using the **docker run** command. An example is given below −

sudo docker run –d --volume-driver=flocker

–v /home/demo:/var/jenkins\_home –p 8080:8080 –p 50000:50000 jenkins

The **–volume-driver** option is used to specify another storage driver for the container.



To confirm that the driver has been changed, first let’s use the **docker ps** command to see the running containers and get the container ID. So, issue the following command first −

sudo docker ps

Then issue a **docker inspect** against the container and put the output in a text file using the command.

sudo docker inspect 9bffb1bfebee > temp.txt



If you browse through the text file and go to the line which says **VolumeDriver**, you will see that the driver name has been changed.



Creating a Volume

A volume can be created beforehand using the **docker** command. Let’s learn more about this command.

Syntax

docker volume create –-name=volumename –-opt options

Options

* **name** − This is the name of the volume which needs to be created.
* **opt** − These are options you can provide while creating the volume.

Return Value

The command will output the name of the volume created.

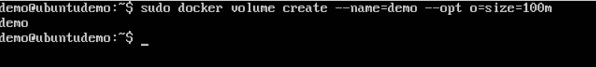
Example

sudo docker volume create –-name = demo –opt o = size = 100m

In the above command, we are creating a volume of size 100MB and with a name of demo.

Output

The output of the above command is shown below −



Listing all the Volumes

You can also list all the **docker volumes** on a **docker host**. More details on this command is given below −

Syntax

docker volume ls

Options

None

Return Value

The command will output all the volumes on the **docker host**.

Example

sudo docker volume ls

Output

The output of the above command is shown below −

