**Lab Exercise 8- Understanding Docker Storage Drivers Lab Exercise**

**Objective:**

To explore and understand the various Docker storage drivers, such as overlay2, aufs, devicemapper, and btrfs, and how they manage container storage.

**Prerequisites:**

* Docker installed on your machine
* Administrator access to configure Docker settings (changing storage drivers requires restarting Docker)
* Basic understanding of Docker container layers and images

**Part 1: Overview of Docker Storage Drivers**

Docker storage drivers are responsible for managing the read/write layer of a container. Some common storage drivers include:

1. **Overlay2:** The default and recommended storage driver for modern Linux distributions. It is fast, efficient, and works with multiple layers.
2. **AUFS:** Advanced Multi-layered Unification File System, an older driver, still used on some systems but being phased out.
3. **Devicemapper:** Uses block-level storage and snapshots, providing strong isolation between containers. It's more complex than overlay2 and not preferred unless necessary.
4. **Btrfs:** A file system with built-in snapshot capabilities, commonly used for advanced setups that require file system-level snapshots.

**Part 2: Checking and Changing the Storage Driver**

1. **Check the Current Storage Driver:**

To find out which storage driver is currently in use:

docker info | grep "Storage Driver"

This will display the active storage driver, which is likely overlay2 on most modern systems.

1. **Change the Storage Driver:**

**Note:** Changing the storage driver involves restarting Docker and removing existing containers and images. Ensure you back up any important data before proceeding.

To change the storage driver, follow these steps:

* + Stop Docker:

sudo systemctl stop docker

* + Edit the Docker daemon configuration file (/etc/docker/daemon.json) to specify the desired storage driver (e.g., to use devicemapper):

{

"storage-driver": "devicemapper"

}

* + Restart Docker:

sudo systemctl start docker

* + Verify the change:

docker info | grep "Storage Driver"

**Part 3: Exploring Overlay2 Storage Driver**

1. **Create a Dockerfile with Layers:**

Create a directory called docker-storage-overlay2 and navigate to it:

mkdir docker-storage-overlay2 && cd docker-storage-overlay2

Create a simple Dockerfile:

# Use an official Ubuntu base image

FROM ubuntu:20.04

# Install some basic utilities

RUN apt-get update && apt-get install -y curl

# Create a test file

RUN echo "This is a test file" > /test.txt

# Set default command

CMD ["cat", "/test.txt"]

1. **Build and Run the Image:**

Build the image:

docker build -t overlay2-example .

Run the container:

docker run overlay2-example

The output should display the content of /test.txt.

1. **Inspect the Image Layers:**

Use the docker history command to see the image layers:

docker history overlay2-example

This shows the layers created by each RUN instruction. In the overlay2 driver, these layers are managed efficiently, allowing file changes between layers to be handled with low overhead.

**Part 4: Exploring AUFS Storage Driver (if available)**

1. **Switch to AUFS (if supported):**

If your system supports AUFS, you can change the storage driver to aufs by following the steps in Part 2. Once Docker is using aufs, repeat the steps from Part 3 to observe how AUFS manages container storage.

1. **Inspect AUFS Layers:**

After running the same container, inspect the image layers and see how AUFS handles them:

docker history overlay2-example

Compare the output with that of overlay2. AUFS, like overlay2, manages the container’s layers, but it uses a different method for stacking these layers.

**Part 5: Exploring Devicemapper Storage Driver**

1. **Switch to Devicemapper:**

Edit /etc/docker/daemon.json to use devicemapper:

{

"storage-driver": "devicemapper"

}

Restart Docker and verify the change using:

docker info | grep "Storage Driver"

1. **Run the Same Container:**

Rebuild and run the overlay2-example container:

docker build -t devicemapper-example .

docker run devicemapper-example

1. **Inspect the Layers:**

Run the docker history command again:

docker history devicemapper-example

Devicemapper uses block-level snapshots to create layers, making it more complex than overlay2 or aufs.