Lab Exercise 3: Working with Docker Volumes

Objective:

- Learn how to create and manage Docker volumes.
- Understand how Docker volumes can be used to persist data across container restarts.
- Practice mounting Docker volumes to containers.

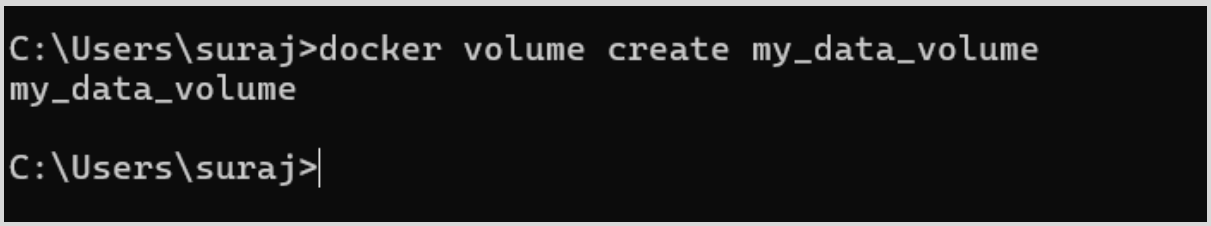
Prerequisites:

- Docker installed on your system.
- Basic understanding of Docker commands and container concepts.

Step 1: Create a Docker Volume

Create a new Docker volume:

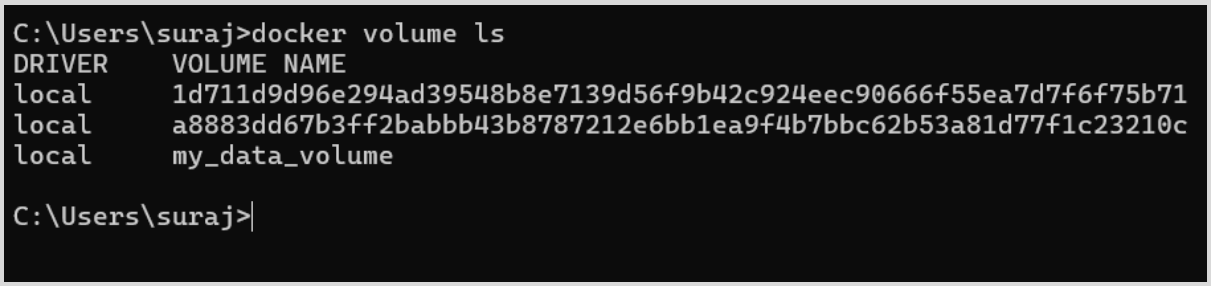
```
docker volume create my_data_volume
```



This command creates a Docker volume named my_data_volume.

Verify that the volume was created:

```
docker volume ls
```



You should see my_data_volume listed among the volumes.

Step 2: Run a Container with the Volume Mounted

Run an Nginx container with the volume mounted:

```
docker run -d --name my_nginx -v my_data_volume:/usr/share/nginx/html -p 8008:80 nginx
```

```
C:\Users\suraj>docker run -d --name my_nginx -v my_data_volume:/usr/share/ng
inx/html -p 8008:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
e4fff0779e6d: Already exists
2a0cb278fd9f: Pull complete
7045d6c32ae2: Pull complete
03de31afb035: Pull complete
0f17be8dcff2: Pull complete
14b7e5e8f394: Pull complete
23fa5a7b99a6: Pull complete
Digest: sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577
add
Status: Downloaded newer image for nginx:latest
4bf1023ae45a36e0109d3a14af94793f188e320d99283993b809dd9eb2480351

C:\Users\suraj>
```

This command starts an Nginx container named my_nginx and mounts the my_data_volume volume to the /usr/share/nginx/html directory inside the container.

Verify that the container is running:

```
docker ps
```

```
C:\Users\suraj>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
       PORTS                    NAMES
4bf1023ae45a   nginx     "/docker-entrypoint.…"   24 seconds ago   Up 22 sec
onds   0.0.0.0:8008->80/tcp   my_nginx

C:\Users\suraj>
```

You should see my_nginx listed as one of the running containers.

Step 3: Interact with the Volume

Create a simple HTML file in the volume:

docker exec -it my_nginx bash

echo "<h1>Hello, Docker Volume!</h1>" > /usr/share/nginx/html/index.html

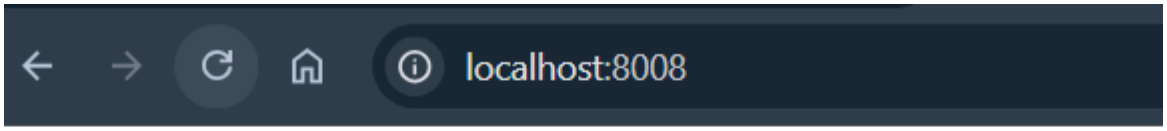
Exit

```
C:\Users\suraj>docker exec -it my_nginx bash
root@4bf1023ae45a:/# echo "<h1>Hello, Docker Volume</h1>" > /usr/share/nginx
/html/index.html
root@4bf1023ae45a:/# exit
exit

C:\Users\suraj>
```

This command creates an HTML file inside the /usr/share/nginx/html directory, which is backed by my_data_volume.

Access the Nginx server to see your file: Open a browser and navigate to http://localhost:8008. You should see the message "Hello, Docker Volume!" displayed on the page.



Hello, Docker Volume

Step 4: Test Data Persistence

Stop and remove the container:

docker stop my_nginx

```
C:\Users\suraj>docker stop my_nginx
my_nginx
```

docker rm my_nginx

```
C:\Users\suraj>docker rm my_nginx
my_nginx
```

Run a new Nginx container using the same volume:

docker run -d -p 8011:80 -v my_data_volume:/usr/share/nginx/html nginx

```
C:\Users\suraj>docker run -d -p 8011:80 -v my_data_volume:/usr/share/nginx/h
tml nginx
e90aa7076105f71dcbc5675e18c171986a2aba09a07303dd22e083a58241a4b0
```

Access the Nginx server again: Navigate to http://localhost in your browser. You should still see the "Hello, Docker Volume!" message, demonstrating that the data persisted across container instances.

Step 5: Clean Up

Stop and remove the container:

docker stop new_nginx

```
C:\Users\suraj>docker stop e90aa7076105
e90aa7076105

C:\Users\suraj>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

docker rm new_nginx

```
C:\Users\suraj>docker rm e90aa7076105
e90aa7076105
```

Remove the Docker volume:

```
docker volume rm my_data_volume
```

```
C:\Users\suraj>docker volume rm my_data_volume
my_data_volume
```

Verify that the volume is removed:

```
docker volume ls
```

```
C:\Users\suraj>
```

Ensure that my_data_volume is no longer listed.