Lab Experiment 3: Docker Volume

In this lab experiment, you will learn how to work with Docker volumes, which are used to persist data across containers. Volumes enable data to be stored outside the container filesystem and are crucial for managing data consistency and sharing data between containers.

Prerequisites:

Docker installed and running on your machine.

Objective:

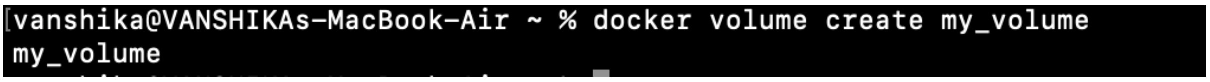
Create a Docker volume, use it with a container, and observe how data persists across container instances.

Steps:

Step 1: Create a Docker Volume

Open a terminal on your machine.

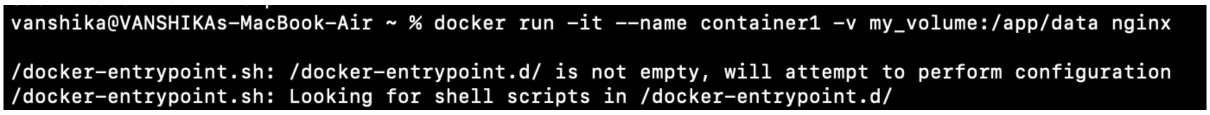
Run the following command to create a Docker volume named “my_volume":



```
[vanshika@VANSHIKAs-MacBook-Air ~ % docker volume create my_volume
my_volume
```

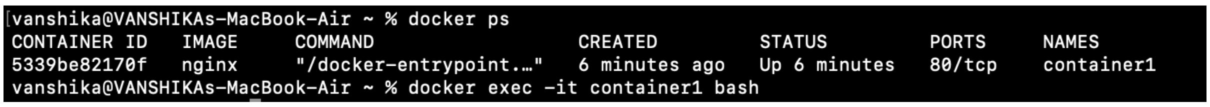
Step 2: Launch Containers with the Volume

Run a container using the volume you created:



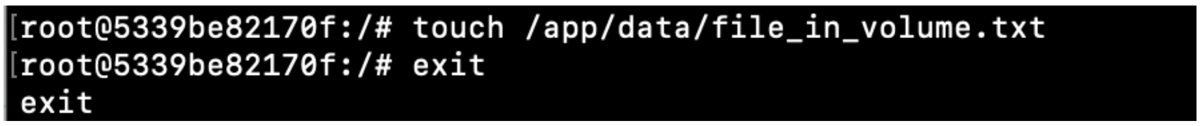
```
vanshika@VANSHIKAs-MacBook-Air ~ % docker run -it --name container1 -v my_volume:/app/data nginx

/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
```



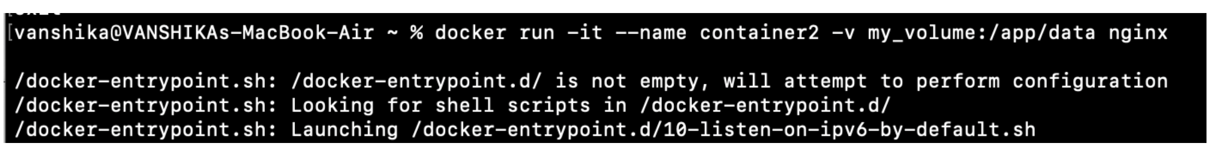
```
[vanshika@VANSHIKAs-MacBook-Air ~ % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
5339be82170f   nginx     "/docker-entrypoint.…"   6 minutes ago   Up 6 minutes   80/tcp    container1
vanshika@VANSHIKAs-MacBook-Air ~ % docker exec -it container1 bash
```

Enter the container to observe the volume and create a file inside it:



```
[root@5339be82170f:/# touch /app/data/file_in_volume.txt
[root@5339be82170f:/# exit
 exit
```

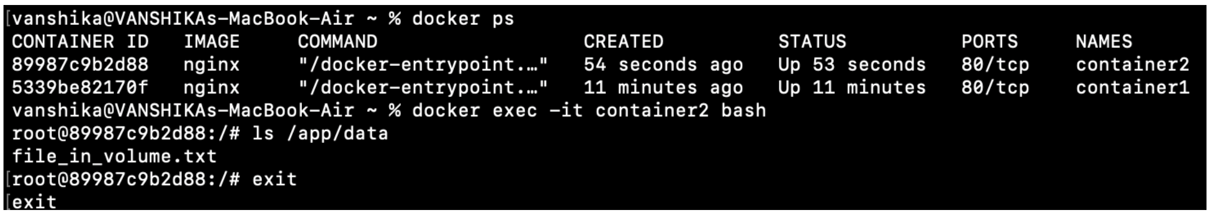
Run a second container, using the same volume, to verify data persistence:



```
[vanshika@VANSHIKAs-MacBook-Air ~ % docker run -it --name container2 -v my_volume:/app/data nginx

 /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
 /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
 /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
```

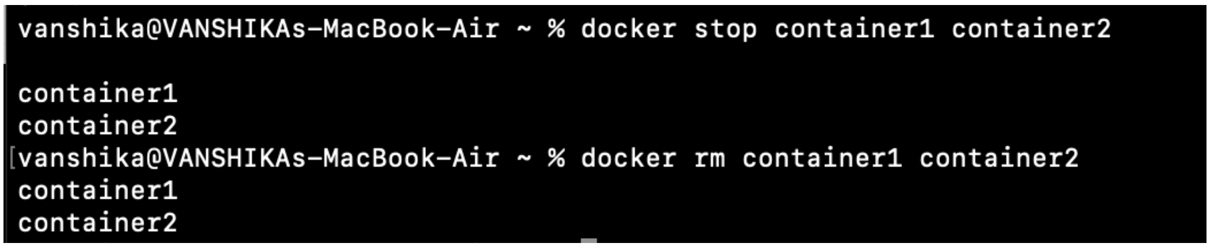
Enter the second container and check if the file exists:



```
[vanshika@VANSHIKAs-MacBook-Air ~ % docker ps
 CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
 89987c9b2d88   nginx     "/docker-entrypoint.…"   54 seconds ago   Up 53 seconds   80/tcp    container2
 5339be82170f   nginx     "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   80/tcp    container1
 vanshika@VANSHIKAs-MacBook-Air ~ % docker exec -it container2 bash
 root@89987c9b2d88:/# ls /app/data
 file_in_volume.txt
[root@89987c9b2d88:/# exit
[exit
```

Step 3: Cleanup

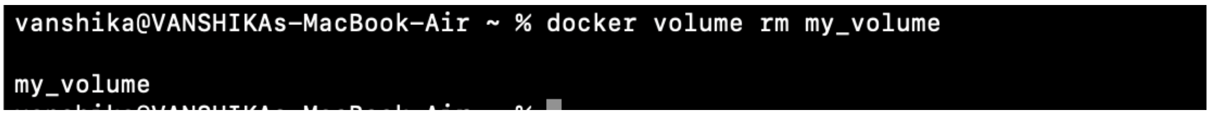
Stop and remove the containers:



```
 vanshika@VANSHIKAs-MacBook-Air ~ % docker stop container1 container2

 container1
 container2
[vanshika@VANSHIKAs-MacBook-Air ~ % docker rm container1 container2
 container1
 container2
```

Remove the volume:



```
vanshika@VANSHIKAs-MacBook-Air ~ % docker volume rm my_volume

my_volume
```

Conclusion:

In this experiment, you learned how to create a Docker volume, associate it with containers, and observed how data persisted between different container instances. Docker volumes are essential for maintaining data integrity, sharing data between containers, and ensuring data persistence even when containers are removed or replaced. This skill is crucial for managing stateful applications and databases within a Dockerized environment.