**Lab Exercise 1**

**Performing CRUD Operation on Containers**

**Objective:** Performing CRUD Operation on Containers

**Tools required:** Docker Configuration

**Pre-requisites:** Ubuntu Configuration, Docker

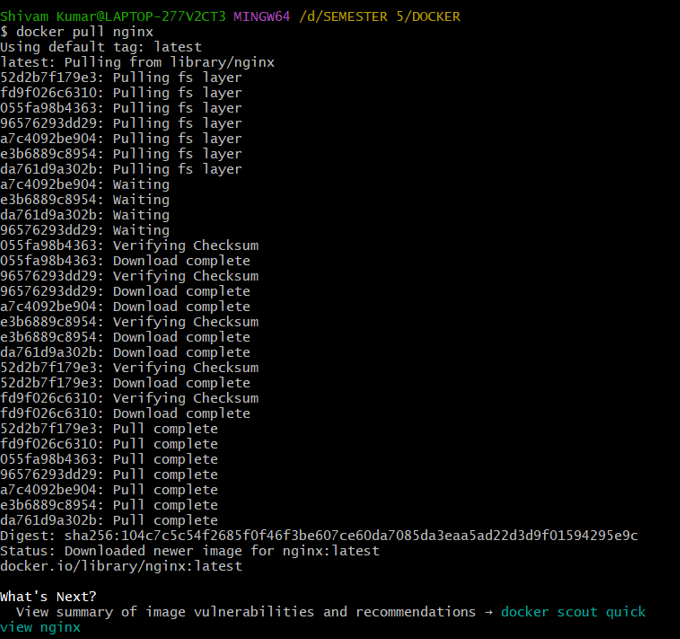
Steps to be followed:

1. Pulling a Docker image
2. Creating a new container
3. Stopping the container
4. Listing all the containers
5. Deleting the container
6. Removing the image

**Step 1: Pulling a Docker image**

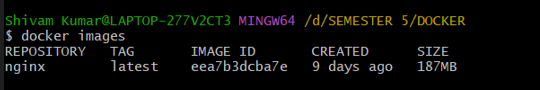
1.1 Open the terminal and pull an image using the command:

***sudo docker pull nginx***



1.2 List all the docker images to check the newly pulled *nginx* image:

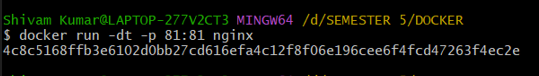
***sudo docker images***



**Step 2: Creating a new container**

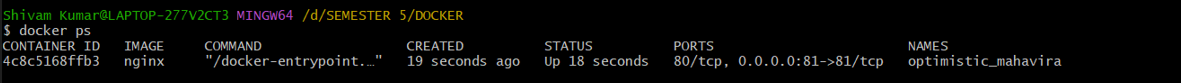
2.1 Create a new container from the *nginx* image:

***sudo docker run -dt -p 81:81 nginx***



2.2 List all the running containers to check the newly created container. You can find various details like port of container, it’s time of creation and ID.

***sudo docker ps***

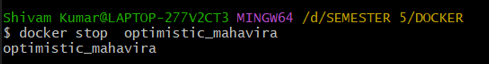


**Step 3: Stopping the container**

3.1 Use the following command to stop the running container. ***(***You can also us the container ID to stop the container: *sudo docker stop CONTAINER\_ID****)***

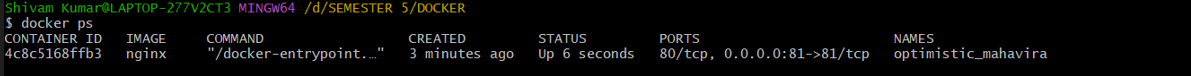
***sudo docker stop CONTAINER\_NAME***

**Note**: Replace CONTAINER\_NAME with the name of the newly created container. In this case CONTAINER\_NAME is optimistic\_mahavira. The container name may differ from the one shown in the image below.



3.2 Use the following command to list all the running containers and verify if the container has stopped running:

***sudo docker ps***

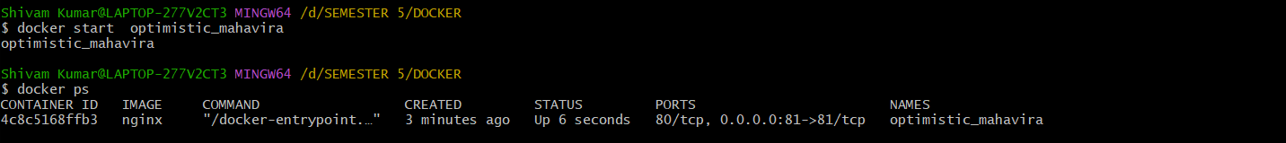


3.3 You can start the container again and check the running containers. ***(***You can also us the container ID to start the container: *sudo docker start CONTAINER\_ID****)***

***sudo docker start CONTAINER\_NAME***

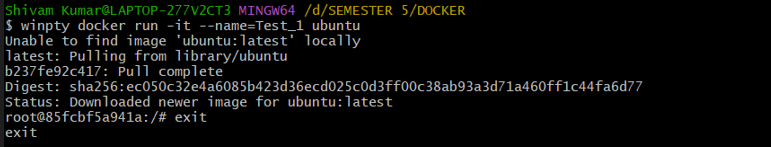
***sudo docker ps***

**Note**: Replace CONTAINER\_NAME with the name of the newly created container. In this case CONTAINER\_NAME is stoic\_darwin. The container name may differ from the one shown in the image below.



3.4 To start the container in interactive mode, use the –i and –t options.

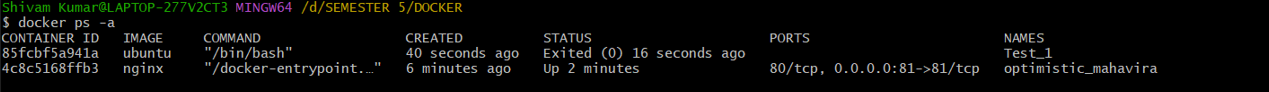
***sudo docker run -it --name=Test\_1 ubuntu***



**Step 4: Listing all the containers**

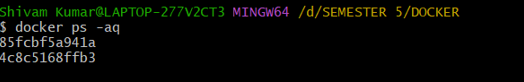
4.1 Use the below command to list all the containers started and the once which are stopped:

***sudo docker ps -a***



4.2 To list the containers by their ID, use the below command

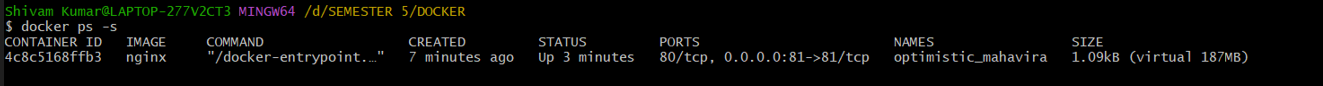
***sudo docker ps -aq***



You can see the containers with ID are listed.

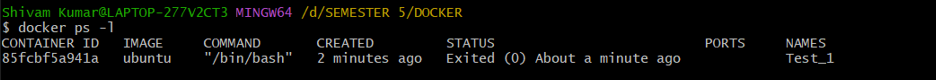
4.3 To list the total file size of each container, use the below command:

***sudo docker ps -s***



4.4 To list the latest created containers, use the following command:

***sudo docker ps -l***

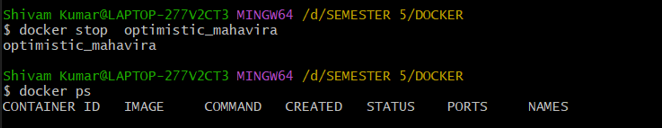


**Step 5: Deleting the container**

5.1 Stop the running container and remove it using the following commands:

***sudo docker stop CONTAINER\_NAME***

***sudo docker container rm CONTAINER\_NAME***

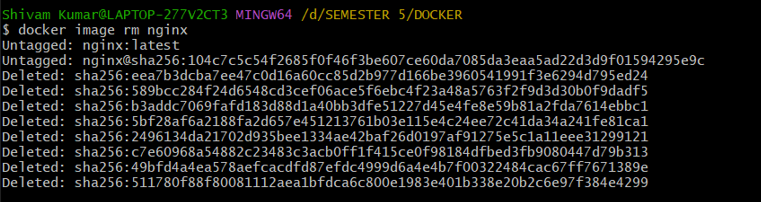


**Note**: Replace CONTAINER\_NAME with the name of the newly created container. In this case CONTAINER\_NAME is sweet\_brown. The container name may differ from the one shown in the image below.

**Step 6: Removing the image**

6.1 Remove the image using the command:

***sudo* *docker image rm nginx***



**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":

docker volume create my\_volume



Step 2: Launch Containers with the Volume

Run a container using the volume you created:

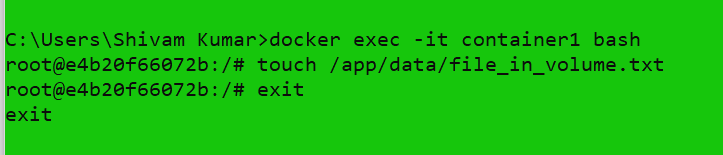
docker run -it --name container1 -v my\_volume:/app/data nginx



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

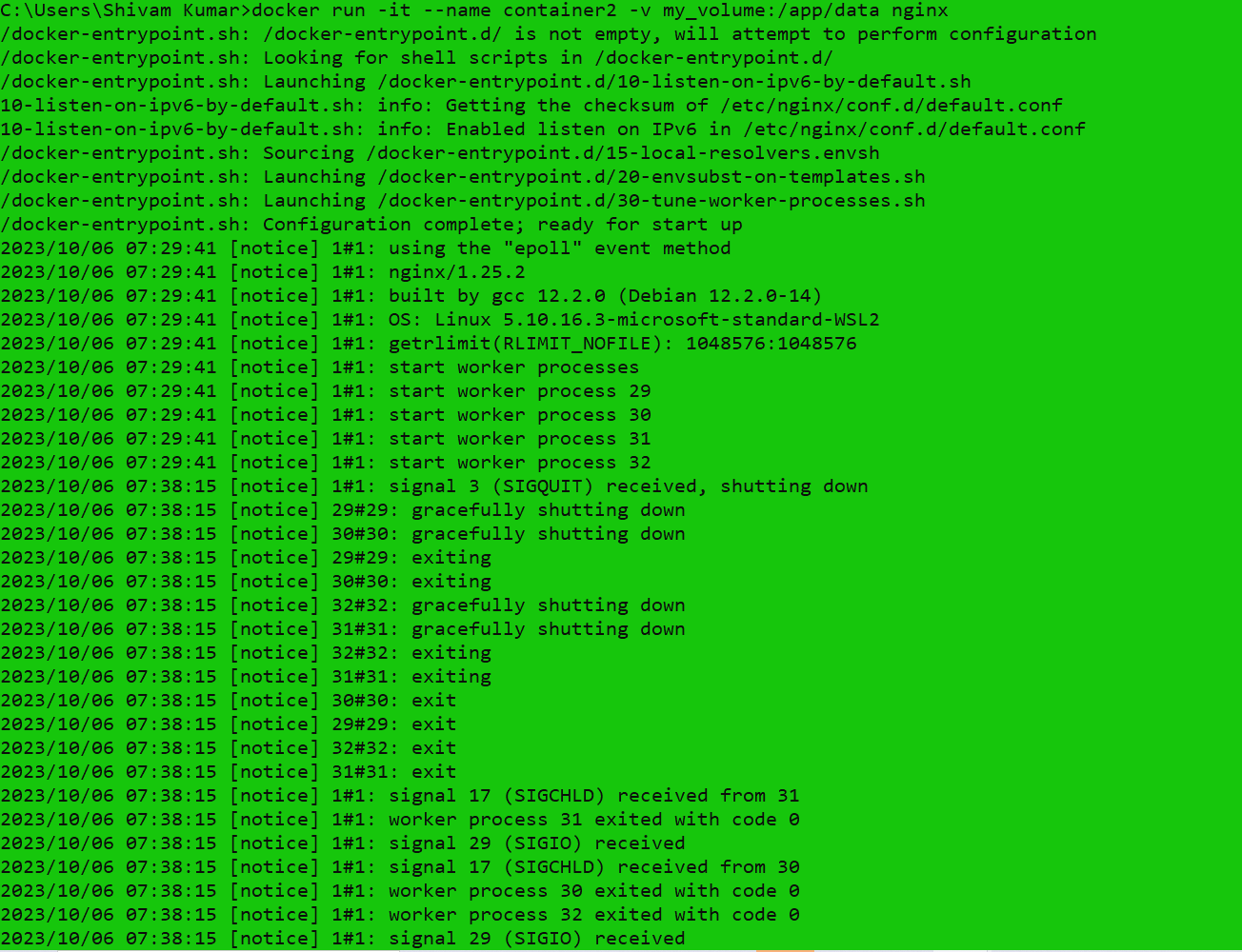
touch /app/data/file\_in\_volume.txt

exit



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

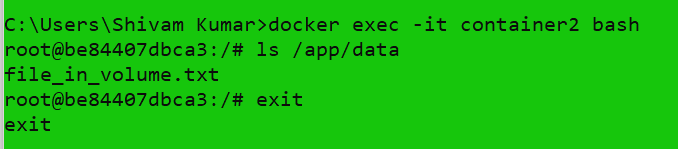
docker run -it --name container2 -v my\_volume:/app/data nginx



Enter the second container and check if the file exists:

ls /app/data

exit

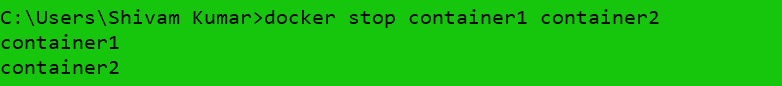


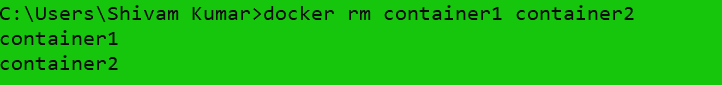
Step 3: Cleanup

Stop and remove the containers:

docker stop container1 container2

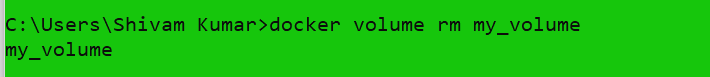
docker rm container1 container2





Remove the volume:

docker volume rm 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.