EXPERIMENT 3

HARSH SHARMA

500097351

AIM: Working with 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.

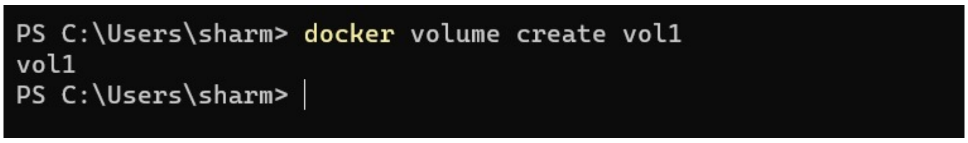
Prerequisites: Docker installed and running on your machine. Objective: Create a Docker volume, use it with a container, and observe how data persists across container instances.

Steps to Complete:

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

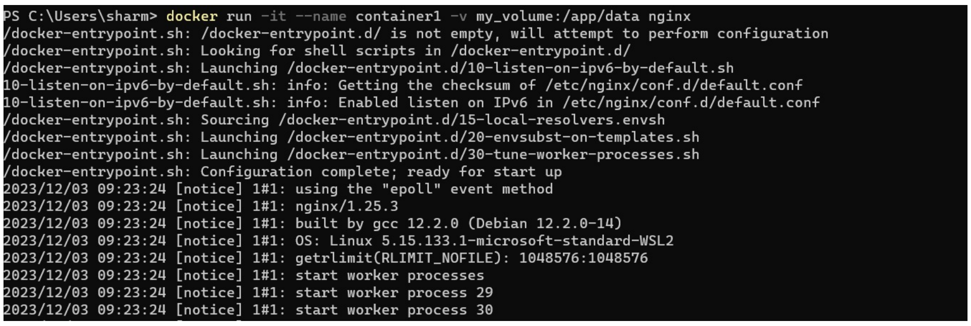


```
PS C:\Users\sharm> docker volume create vol1
vol1
PS C:\Users\sharm> |
```

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

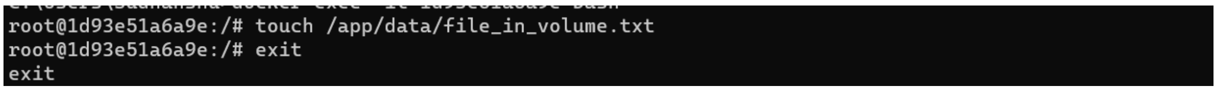


```
PS C:\Users\sharm> docker run -it --name container1 -v my_volume:/app/data nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/12/03 09:23:24 [notice] 1#1: using the "epoll" event method
2023/12/03 09:23:24 [notice] 1#1: nginx/1.25.3
2023/12/03 09:23:24 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/12/03 09:23:24 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2023/12/03 09:23:24 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/12/03 09:23:24 [notice] 1#1: start worker processes
2023/12/03 09:23:24 [notice] 1#1: start worker process 29
2023/12/03 09:23:24 [notice] 1#1: start worker process 30
```

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

touch /app/data/file_in_volume.txt

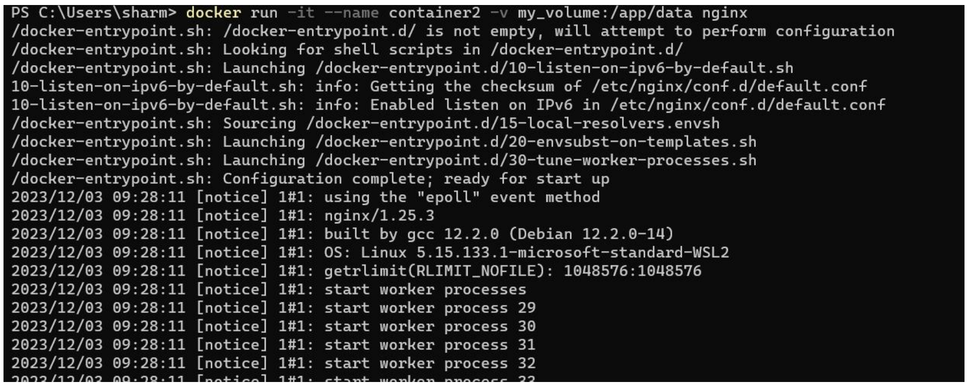
Exit



```
root@1d93e51a6a9e:/# touch /app/data/file_in_volume.txt
root@1d93e51a6a9e:/# exit
exit
```

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

docker run -it --name container2 -v my_volume:/app/data nginx

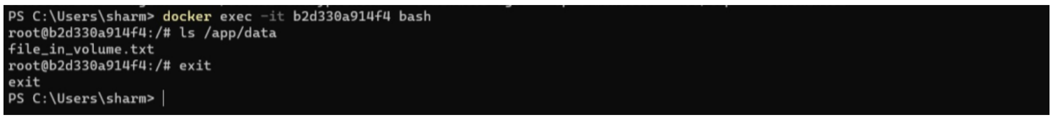


```
PS C:\Users\sharm> docker run -it --name container2 -v my_volume:/app/data nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/12/03 09:28:11 [notice] 1#1: using the "epoll" event method
2023/12/03 09:28:11 [notice] 1#1: nginx/1.25.3
2023/12/03 09:28:11 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/12/03 09:28:11 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2023/12/03 09:28:11 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/12/03 09:28:11 [notice] 1#1: start worker processes
2023/12/03 09:28:11 [notice] 1#1: start worker process 29
2023/12/03 09:28:11 [notice] 1#1: start worker process 30
2023/12/03 09:28:11 [notice] 1#1: start worker process 31
2023/12/03 09:28:11 [notice] 1#1: start worker process 32
```

Enter the second container and check if the file exists:

ls /app/data

Exit



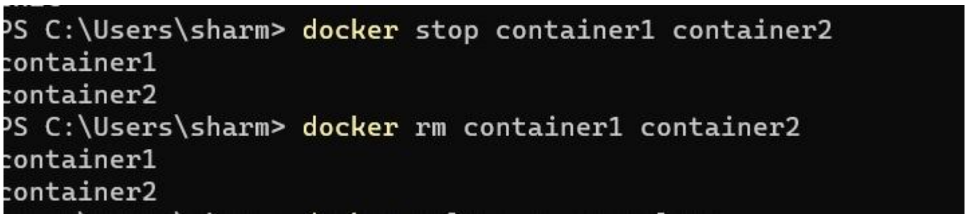
```
PS C:\Users\sharm> docker exec -it b2d330a914f4 bash
root@b2d330a914f4:/# ls /app/data
file_in_volume.txt
root@b2d330a914f4:/# exit
exit
PS C:\Users\sharm> |
```

Step 3: Cleanup

Stop and remove the containers:

docker stop container1 container2

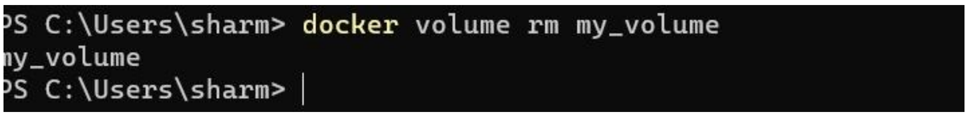
docker rm container1 container2



```
PS C:\Users\sharm> docker stop container1 container2
container1
container2
PS C:\Users\sharm> docker rm container1 container2
container1
container2
```

Remove the volume:

docker volume rm my_volume



```
PS C:\Users\sharm> docker volume rm my_volume
my_volume
PS C:\Users\sharm> |
```

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