Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

Theory:

What is Docker?

Docker is a platform that allows developers to build, package, and deploy applications in lightweight, portable containers. These containers include everything needed to run an application, such as code, runtime, system tools, libraries, and dependencies.

Containerization Technology

Containers are isolated environments where applications run independently. Unlike traditional virtualization, which requires separate operating systems for each application, containers share the host OS kernel, making them faster and more efficient.

Docker Architecture

Docker follows a client-server architecture consisting of the following key components:

1. Docker Client

- It is the command-line interface (CLI) that allows users to interact with Docker.
- Commands such as docker run, docker build, and docker stop are executed through the client.

2. Docker Daemon (dockerd)

- It runs in the background and manages Docker containers, images, volumes, and networks.
- It listens for requests from the Docker Client and executes commands.

3. Docker Images

- A Docker image is a read-only template containing the application code, libraries, and dependencies.
- Images are created using Dockerfiles, which define the steps to build an image.
- Images are stored in Docker Hub or private repositories.

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

4. Docker Containers

- A container is an instance of a Docker image running as an isolated process on a host machine.
- Containers are lightweight, portable, and can be started, stopped, or removed as needed.

5. Docker Registry

- It is a storage system for Docker images.
- The public registry, Docker Hub, provides access to a vast collection of pre-built images.
- Users can also create private registries for security and control.

Docker Container Life Cycle

The **life cycle of a container** follows these steps:

1. **Create** – A container is created from an image using the docker create command.
2. **Start** – The container starts running using the docker start command.
3. **Run** – A new container can be started directly using docker run.
4. **Pause/Unpause** – Containers can be temporarily paused and resumed.
5. **Stop** – The container can be stopped using docker stop.
6. **Restart** – A stopped container can be restarted.
7. **Kill** – A container can be forcefully stopped using docker kill.
8. **Remove** – Containers that are no longer needed can be deleted using docker rm.

Benefits of Docker

1. Portability

- Containers can run on any platform that supports Docker.
- Applications behave consistently across different environments.

2. Efficiency

- Containers share the host OS kernel, reducing overhead and improving performance.
- They consume fewer resources compared to virtual machines.

2

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

3. Isolation

- Each container runs in its own isolated environment, preventing dependency conflicts.

4. Scalability

- Applications can be scaled up quickly by launching multiple containers.
- Docker enables automatic load balancing in large-scale deployments.

5. Consistency

- Ensures that the application runs the same way in development, testing, and production.
- Eliminates the "works on my machine" problem.

<u>Docker Engine:</u>

At the core of Docker is the Docker Engine, which is responsible for building, running, and managing containers. It consists of the Docker daemon, which manages containers, images, networks, and volumes, and the Docker client, which allows users to interact with the daemon through the Docker API.

<u>Docker Images:</u>

Docker images are read-only templates used to create containers. They contain the application code, runtime, libraries, dependencies, and other files needed to run the application. Images are built using Dockerfiles, which are text files that define the steps needed to create the image.

⌘ <u>**OUTPUT**</u>**:-**

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

```
C:\Users\202>docker run redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
8a1e25ce7c4f: Pull complete
8ab039a68e51: Pull complete
2b12a49dcfb9: Pull complete
cdf9868f47ac: Pull complete
e73ea5d3136b: Pull complete
890ad32c613f: Pull complete
4f4fb700ef54: Pull complete
ba517b76f92b: Pull complete
Digest: sha256:7dd707032d90c6eaafd566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Downloaded newer image for redis:latest
1:C 13 Mar 2024 03:19:03.928 * oO0Oo00OoO0Oo Redis is starting oO0Oo00OoO0Oo
1:C 13 Mar 2024 03:19:03.928 * Redis version=7.2.4, bits=64, commit=00000000, mod
1:C 13 Mar 2024 03:19:03.928 # Warning: no config file specified, using the defa
s.conf
1:M 13 Mar 2024 03:19:03.929 * monotonic clock: POSIX clock_gettime
1:M 13 Mar 2024 03:19:03.929 * Running mode=standalone, port=6379.
1:M 13 Mar 2024 03:19:03.929 * Server initialized
1:M 13 Mar 2024 03:19:03.929 * Ready to accept connections tcp
1:signal-handler (1710300105) Received SIGINT scheduling shutdown...
1:M 13 Mar 2024 03:21:45.877 * User requested shutdown...
1:M 13 Mar 2024 03:21:45.877 * Saving the final RDB snapshot before exiting.
1:M 13 Mar 2024 03:21:45.887 * DB saved on disk
1:M 13 Mar 2024 03:21:45.887 # Redis is now ready to exit, bye bye...
```

```
C:\Users\202>docker images
REPOSITORY      TAG        IMAGE ID       CREATED         SIZE
redis           latest     170a1e90f843   2 months ago    138MB
```

```
C:\Users\202>docker pull redis
Using default tag: latest
latest: Pulling from library/redis
Digest: sha256:7dd707032d90c6eaafd566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Image is up to date for redis:latest
docker.io/library/redis:latest
```

```
C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED     STATUS     PORTS     NAMES

C:\Users\202>docker ps
CONTAINER ID   IMAGE    COMMAND              CREATED          STATUS         PORTS      NAMES
052aecb0ee88   redis    "docker-entrypoint.s…"   About a minute ago   Up 4 seconds    6379/tcp   container121
1c4472744083   redis    "docker-entrypoint.s…"   6 minutes ago        Up 10 seconds   6379/tcp   modest_herschel

C:\Users\202>
```
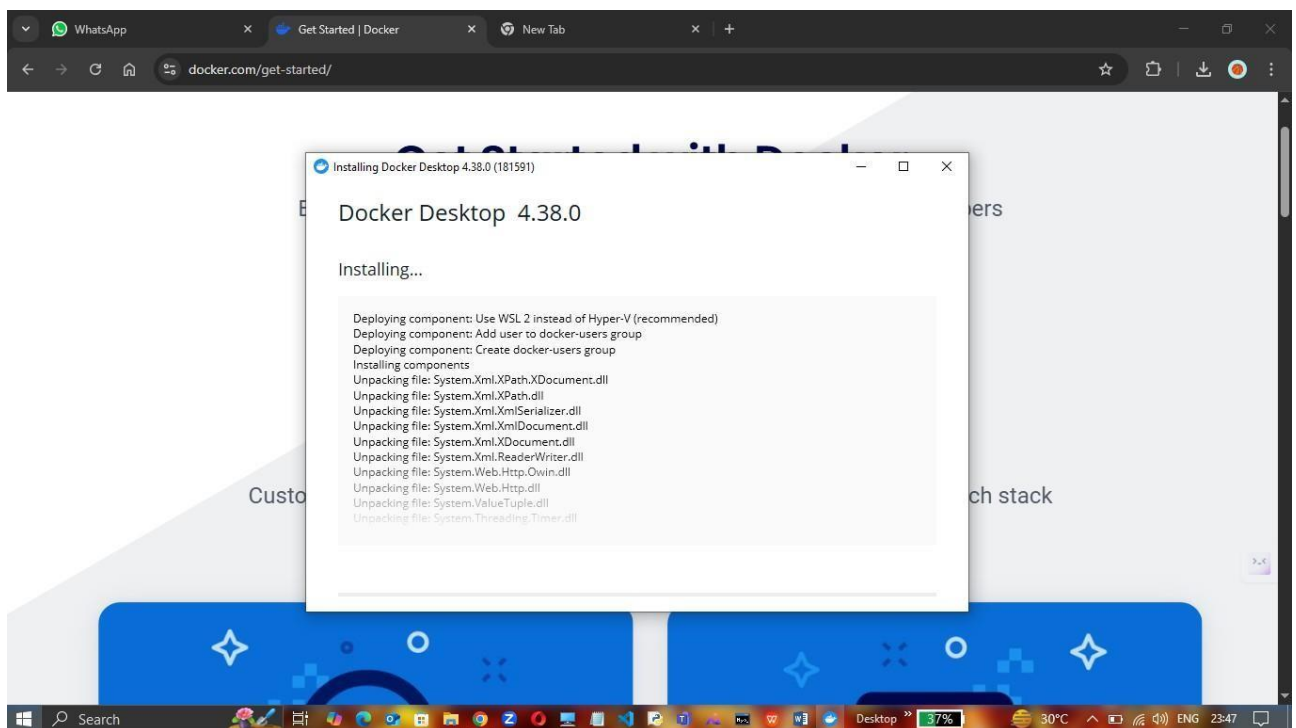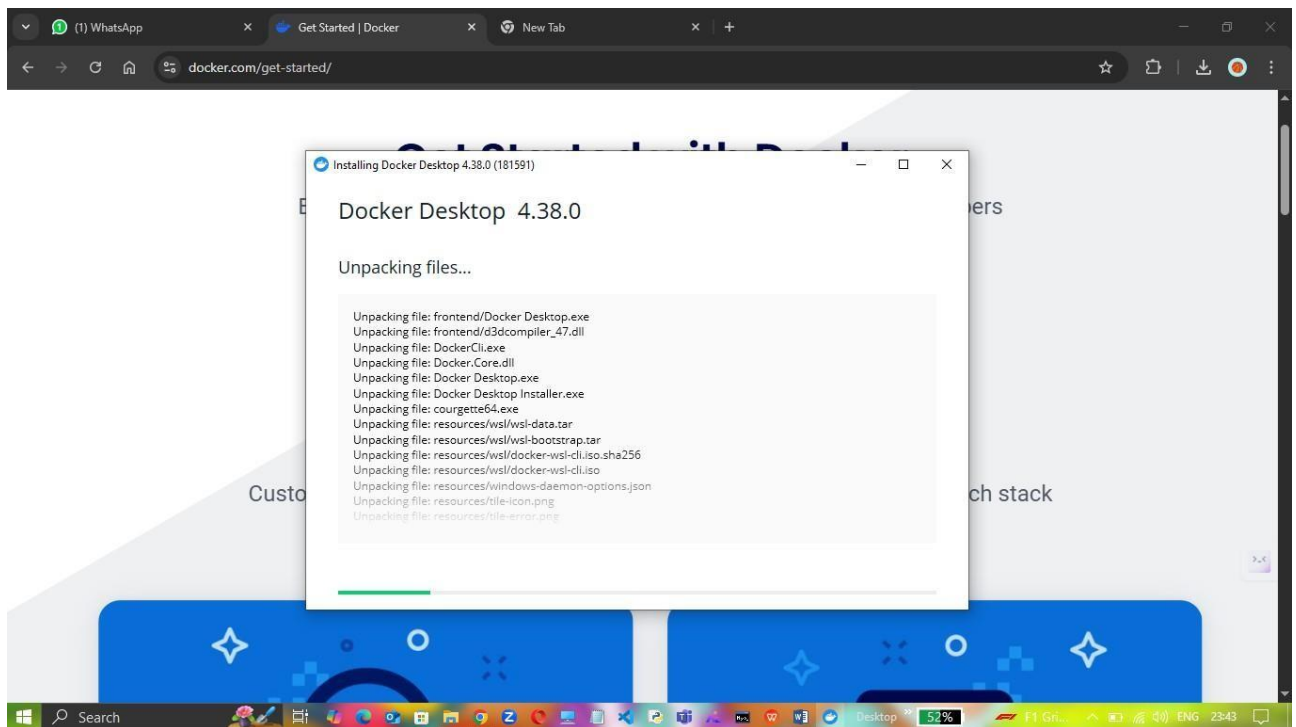
Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers
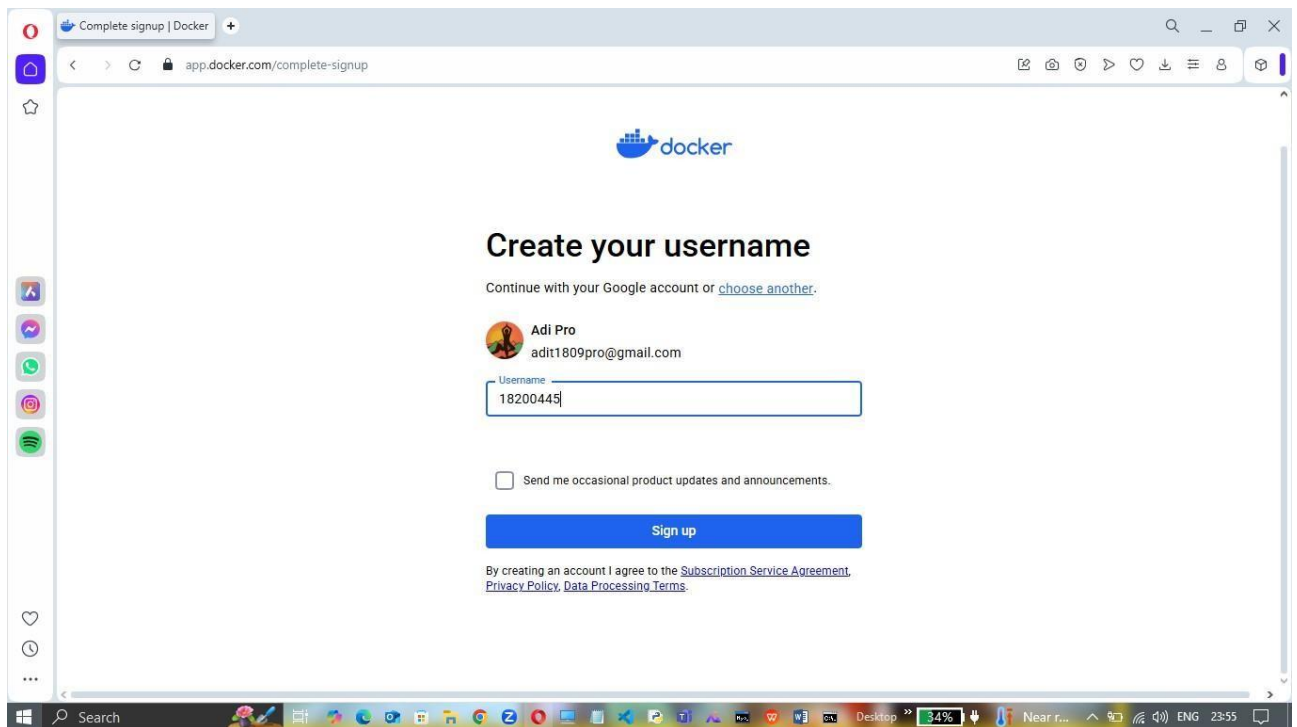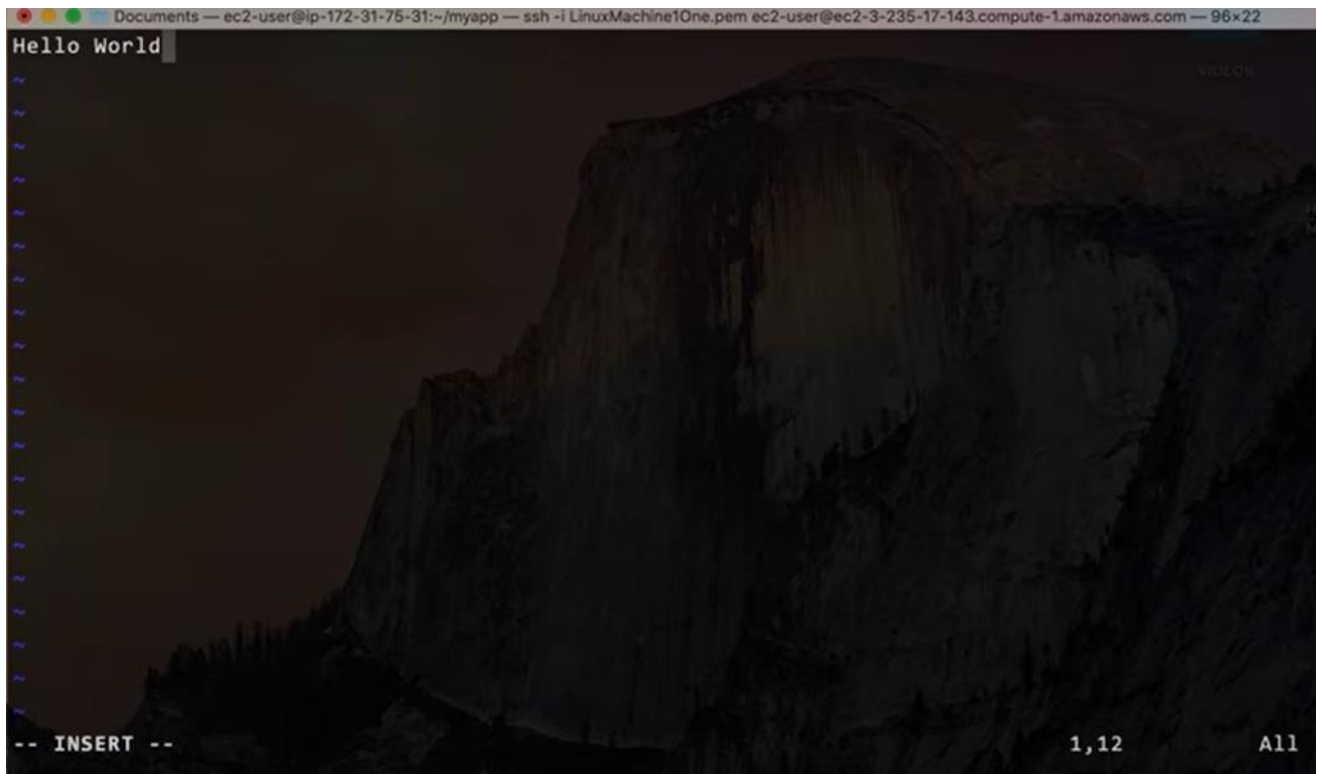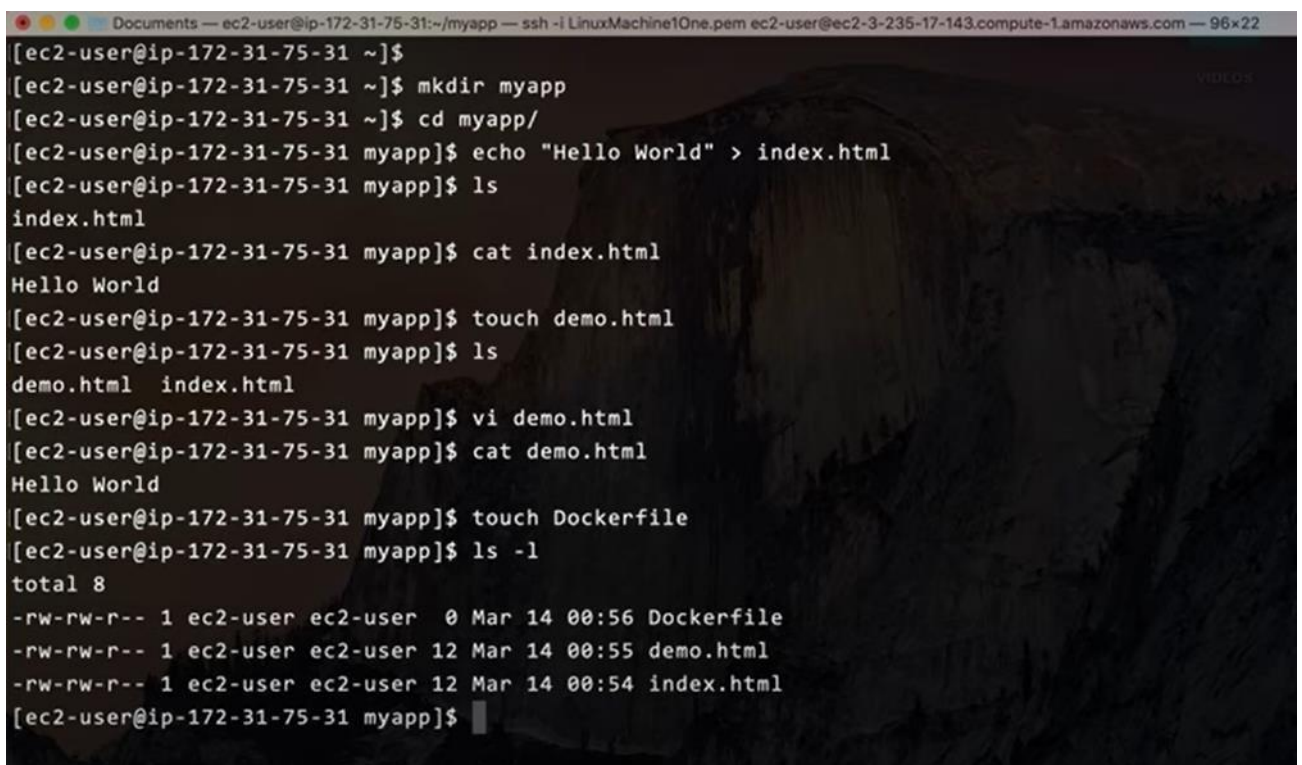
```
C:\Users\202>docker ps
CONTAINER ID   IMAGE    COMMAND              CREATED            STATUS          PORTS       NAMES
052aecb0ee88   redis    "docker-entrypoint.s…"  About a minute ago  Up 4 seconds    6379/tcp    container121
1c4472744083   redis    "docker-entrypoint.s…"  6 minutes ago       Up 10 seconds   6379/tcp    modest_herschel

C:\Users\202>
C:\Users\202>
C:\Users\202>docker stop 052aecb0ee88
052aecb0ee88

C:\Users\202>docker ps
CONTAINER ID   IMAGE    COMMAND              CREATED         STATUS        PORTS       NAMES
1c4472744083   redis    "docker-entrypoint.s…"  9 minutes ago   Up 2 minutes  6379/tcp    modest_herschel
```

```
C:\Users\202>docker start 052aecb0ee88
052aecb0ee88

C:\Users\202>docker ps
CONTAINER ID   IMAGE    COMMAND              CREATED          STATUS        PORTS       NAMES
052aecb0ee88   redis    "docker-entrypoint.s…"  5 minutes ago    Up 8 seconds  6379/tcp    container121
1c4472744083   redis    "docker-entrypoint.s…"  10 minutes ago   Up 3 minutes  6379/tcp    modest_herschel
```

```
C:\Users\202>docker rm 052aecb0ee88
052aecb0ee88
```

```
C:\Users\202>docker images
REPOSITORY    TAG       IMAGE ID        CREATED         SIZE
redis         latest    170a1e90f843    2 months ago    138MB

C:\Users\202>
```

```
C:\Users\202>docker exec -d 1c4472744083 touch /tmp/execWorks

C:\Users\202>docker exec -it 1c4472744083 bash
root@1c4472744083:/data#
```

```
C:\Users\202>docker restart 1c4472744083
1c4472744083

C:\Users\202>docker ps
CONTAINER ID   IMAGE    COMMAND              CREATED          STATUS        PORTS       NAMES
1c4472744083   redis    "docker-entrypoint.s…"  13 minutes ago   Up 3 seconds  6379/tcp    modest_herschel
```

5

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

```
C:\Users\202>docker inspect 1c4472744083
[
    {
        "Id": "1c44727440831475b093dcbf93163064b819bdd9ad8378bb3a4fa847dc411d80",
        "Created": "2024-03-13T03:19:03.418741433Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 2112,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2024-03-13T03:32:13.750463204Z",
            "FinishedAt": "2024-03-13T03:32:13.145321277Z"
        },
        "Image": "sha256:170a1e90f8436daa6778aeea3926e716928826c215ca23a8dfd8055f663f9428",
        "ResolvConfPath": "/var/lib/docker/containers/1c44727440831475b093dcbf93163064b819bdd9a
```

```
C:\Users\202>docker commit 1c4472744083 new_image_name:redis2
sha256:33e4284a7e92a4a1331555d01f6e078fc496e3a3ed8eb7f84f2678261ad07e83

C:\Users\202>docker images
REPOSITORY        TAG        IMAGE ID        CREATED          SIZE
new_image_name    redis2     33e4284a7e92    4 seconds ago    138MB
new_image_name    tag        61ab016507fa    36 seconds ago   138MB
redis             latest     170a1e90f843    2 months ago     138MB
```

⊠ **SCREENSHOTS:**

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

TSEC   Batch:-T11   Name & Roll No:- Pratham Bohra ,12

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

TSEC   Batch:-T11   Name & Roll No:- Pratham Bohra ,12

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

TSEC   Batch:-T11   Name & Roll No:- Pratham Bohra ,12

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

TSEC   Batch:-T11   Name & Roll No:- Pratham Bohra ,12

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

```
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ ls
Dockerfile  demo.html  index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi Dockerfile
```

```
FROM nginx
COPY index.html /usr/share/nginx/html
-- INSERT --                                              2,38           All
```

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers


```
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ ls
Dockerfile  demo.html  index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi Dockerfile
[ec2-user@ip-172-31-75-31 myapp]$ cat Dockerfile
FROM nginx
COPY index.html /usr/share/nginx/html
[ec2-user@ip-172-31-75-31 myapp]$ docker info
Client:
 Context:    default
 Debug Mode: false

Server:
ERROR: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon
running?
errors pretty printing info
[ec2-user@ip-172-31-75-31 myapp]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-75-31 myapp]$
```



```
init version: de40ad0
Security Options:
 seccomp
  Profile: default
Kernel Version: 5.10.167-147.601.amzn2.x86_64
Operating System: Amazon Linux 2
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 964.8MiB
Name: ip-172-31-75-31.ec2.internal
ID: 3DRI:26BR:Y5X4:GCJ2:2UYQ:FHFW:AQ5Q:5UIY:67Z2:VVGE:KC6M:DHX2
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

[ec2-user@ip-172-31-75-31 myapp]$
```

12

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

TSEC   Batch:-T11   Name & Roll No:- Pratham Bohra ,12

Software Engineering & Project Management Lab Experiment No: - 09

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers

Conclusion: **Thus, we have successfully installed Docker and execute docker commands to manage images and interact with containers.**