**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.

**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.

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.

**Docker Engine:**

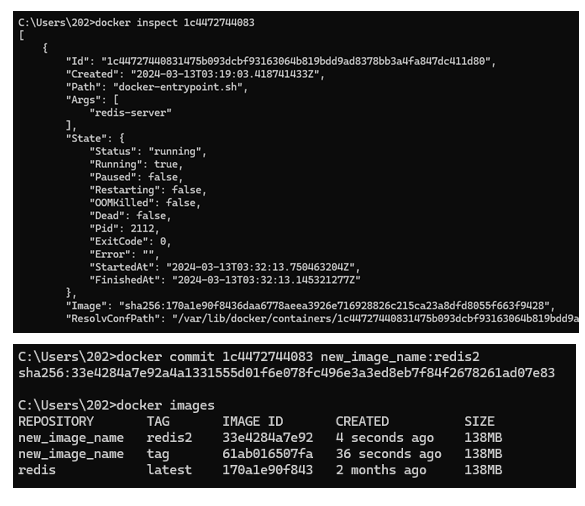
At the core of Docker is the Docker Engine, which is responsible for building, running, and managing containers. It consists of the Docker daemon, which manages containers, images, networks, and volumes, and the Docker client, which allows users to interact with the daemon through the Docker API.

**Docker Images:**

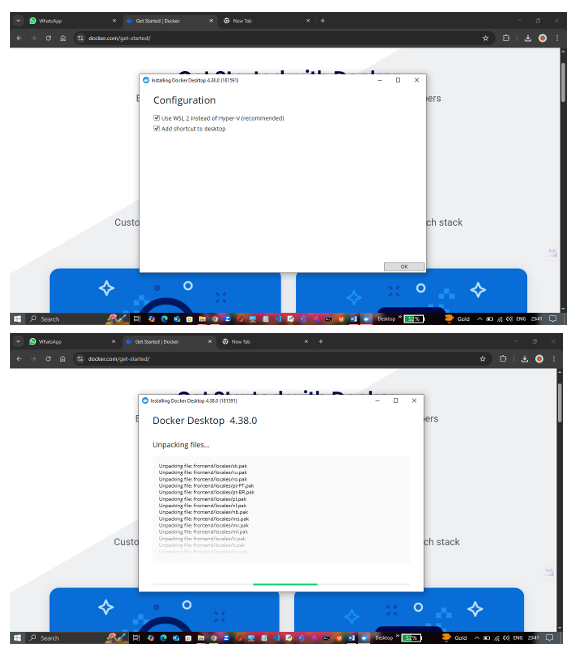
Docker images are read-only templates used to create containers. They contain the application code, runtime, libraries, dependencies, and other files needed to run the application. Images are built using Dockerfiles, which are text files that define the steps needed to create the image.

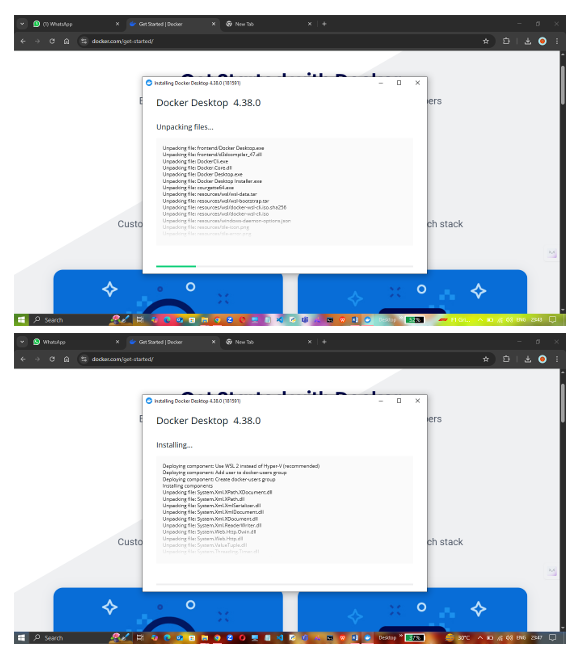


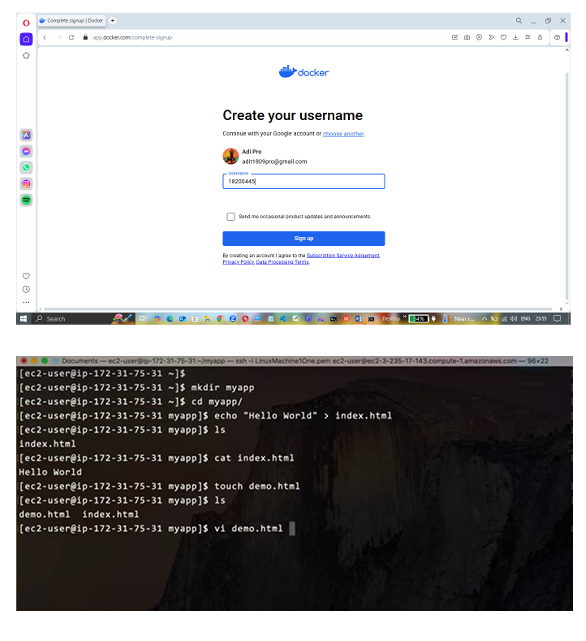


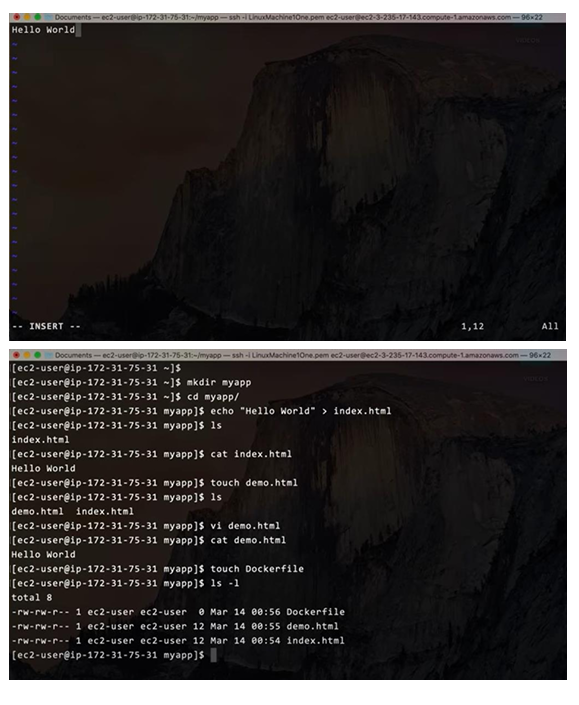


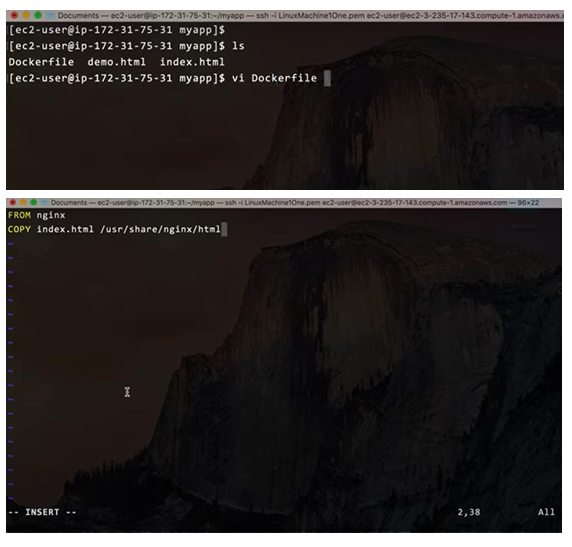
**IMPLEMENTATION:**

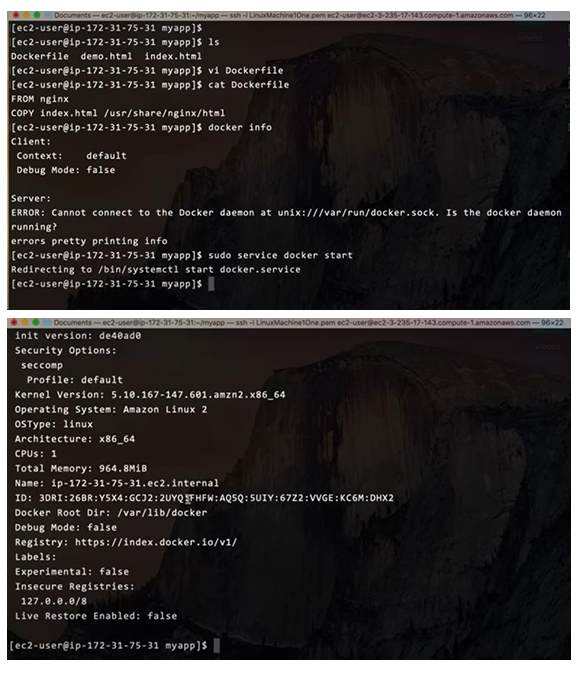


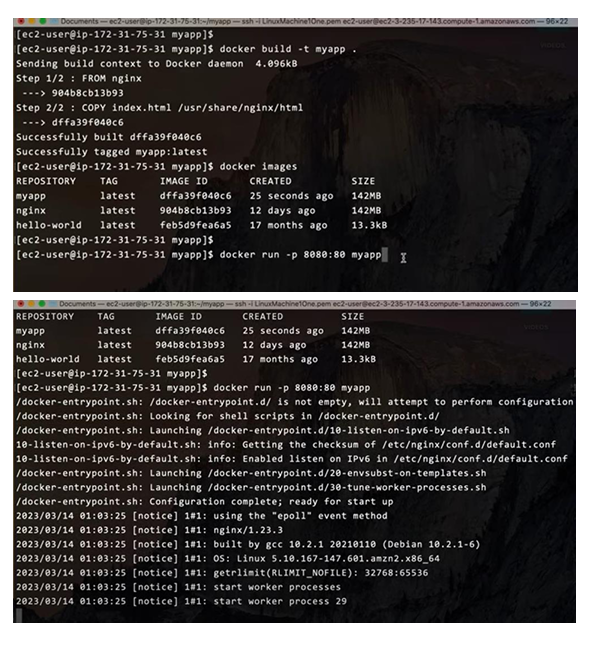


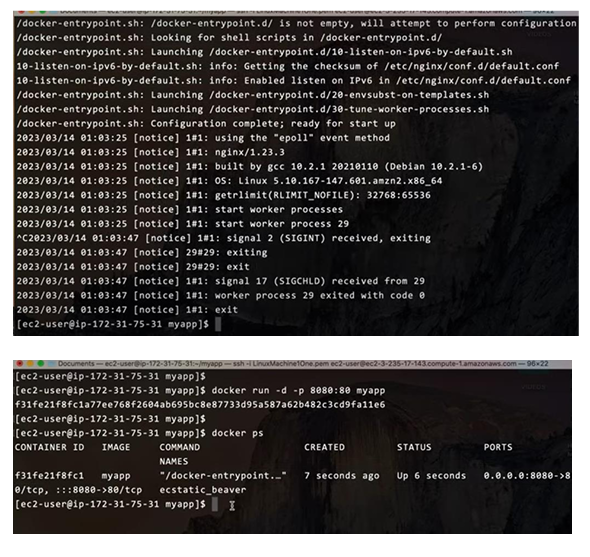












**CONCLUSION:** Thus, we have successfully installed Docker and execute docker commands to manage images and interact with containers.