Name: Rohit Harawade

Roll No: A09

ID: TU4F2122009

Experiment No. 6

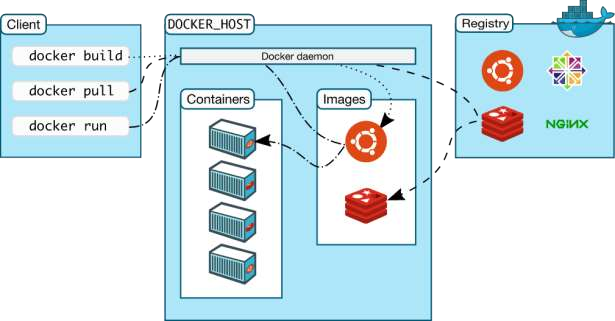
# Aim:- To understand installation of Docker and study of commands for containerization

**Theory:-**

# Docker architecture

Docker uses a client-server architecture. The Docker *client* talks to the

Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.



# The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

# The Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

# Docker registries

A Docker *registry* stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker

push command, your image is pushed to your configured registry.

Docker objects

When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects. This section is a brief overview of some of those objects.

# Images

An *image* is a read-only template with instructions for creating a Docker container. Often, an image is *based on* another image, with some additional customization. For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

You might create your own images or you might only use those created by others and published in a registry. To build your own image, you create a *Dockerfile* with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.

This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.

# Containers

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container’s network, storage, or other underlying subsystems are from other containers or from the host machine.

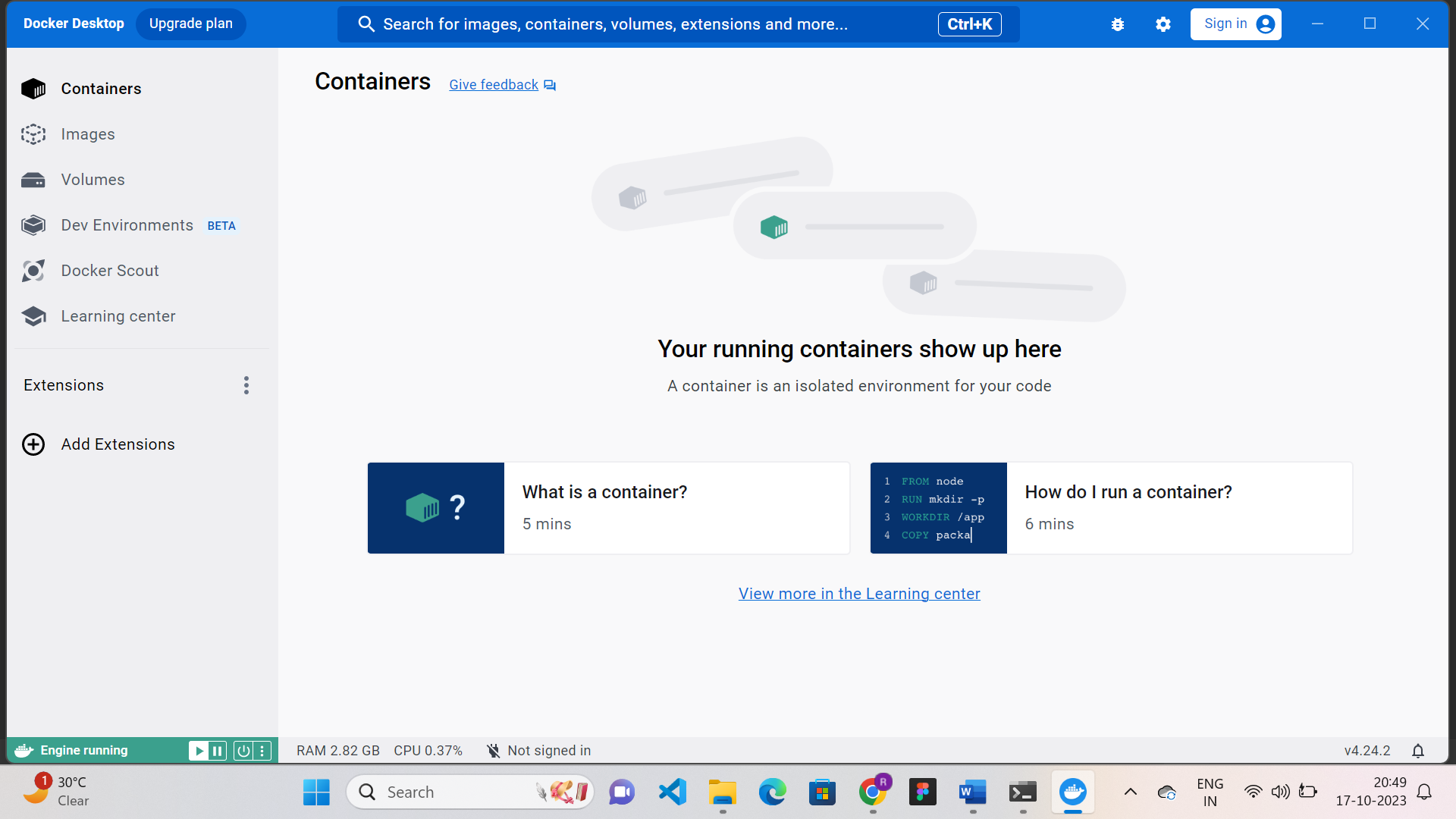
A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

# Docker Installation-

1. Double-click Docker Desktop Installer.exe to run the installer.
2. When prompted, ensure the Use WSL 2 instead of Hyper-V option on the Configuration page is selected or not depending on your choice of backend.

If your system only supports one of the two options, you will not be able to select which backend to use.

1. Follow the instructions on the installation wizard to authorize the installer and proceed with the install.
2. When the installation is successful, select Close to complete the installation process.
3. If your admin account is different to your user account, you must add the user to the docker-users group. Run Computer Management as an administrator and navigate to Local Users and Groups > Groups > docker-users. Right-click to add the user to the group. Sign out and sign back in for the changes to take effect.



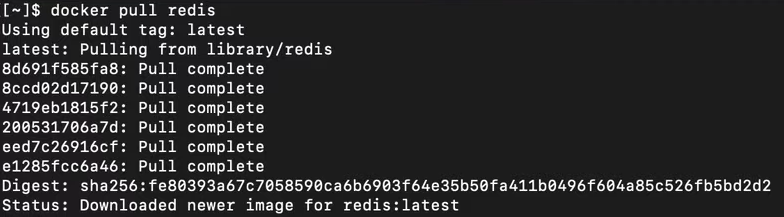
**Docker commands:**

### **docker pull:**

Pull an image or a repository from a registry

Synatx:

$ docker pull image\_name

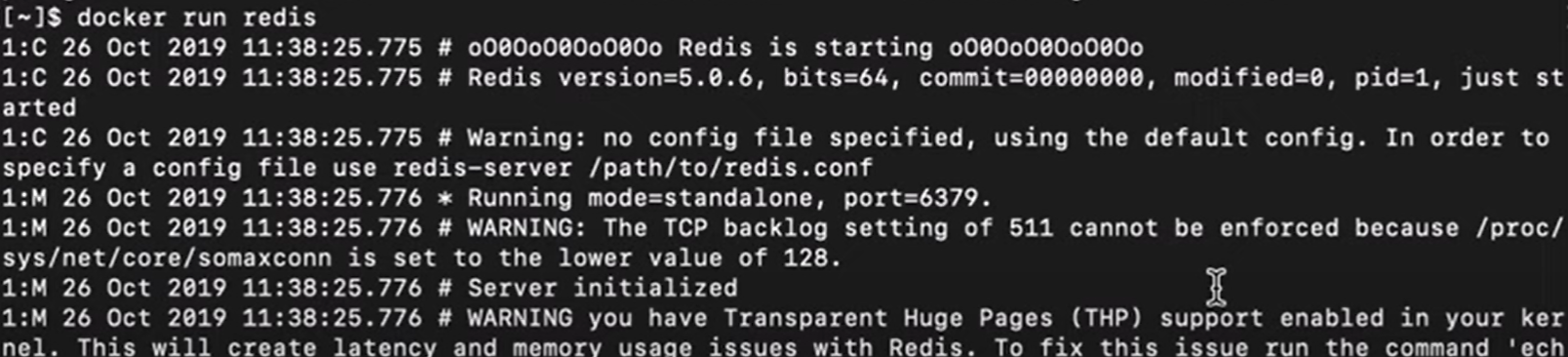


### **docker run**

This command is used to create a container from an image

Syntax

docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

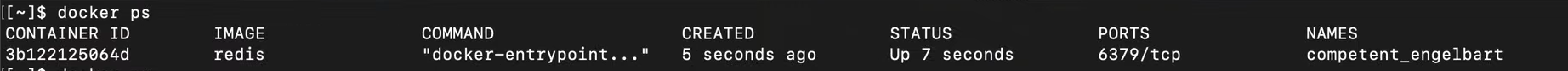


1. **docker ps:**

This command is used to list all the containers

Syntax

docker ps [OPTIONS]



### 4.  **docker start:**

This command is used to start one or more running containers.

Syntax

docker start [OPTIONS] CONTAINER [CONTAINER...]



### 5. **docker stop:**

This command is used to stop one or more running containers.

Syntax:

docker stop [OPTIONS] CONTAINER [CONTAINER...]

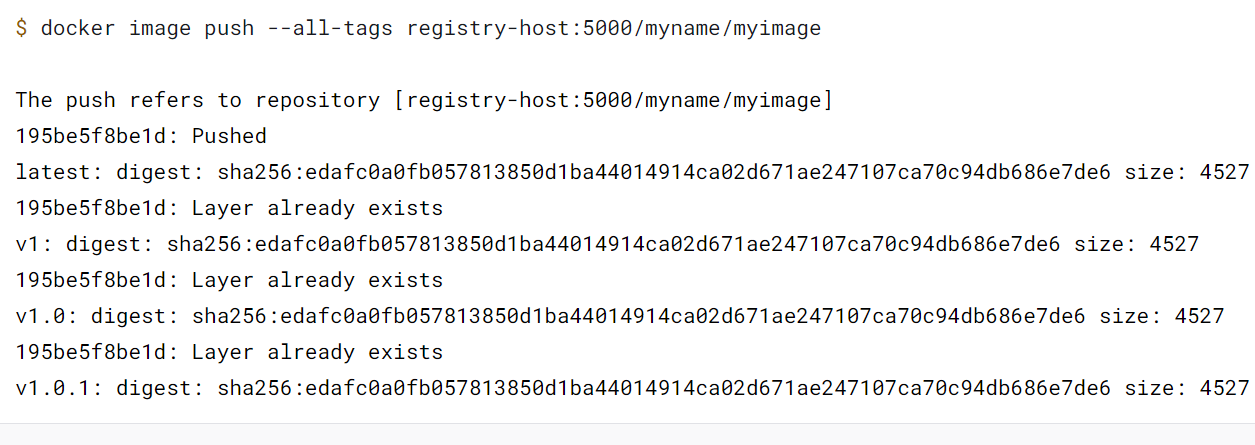


1. **docker push:**

Upload an image to a registry

Syntax:

docker push [OPTIONS] NAME[:TAG]

****

**Conclusion:-** We were able to understand installation of Docker and studied commands for containerization.