St. Francis Institute of Technology, Mumbai-400 103

**Department Of Information Technology**

A.Y. 2024-2025

Class: TE-ITA/B, Semester: V

Subject: **DevOps Lab**

**Experiment – 9: a. To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.**

**b. To learn Dockerfile instructions, build an image for a sample web application using Dockerfile**

1. **Aim: a.** To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers. b. To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.
2. **Objectives:** Aim of this experiment is that, the students will learn:

* Introduction to Docker Architecture
* To use Docker to Build, ship and manage applications using containerization
* To understand concept of containerization
* To analyze the Containerization of OS images and deployment of applications over Docker

1. **Outcomes:** After study of this experiment, the students will learn following:

* Introduction to Docker Architecture
* Container Life Cycle
* Understanding images and containers
* Publishing image on Docker Hub.
* Create and implement docker images using Dockerfile.
* Container Lifecycle and working with containers.
* To Build, deploy and manage web or software application on Docker Engine.

1. **Prerequisite:** None
2. **Requirements:** Docker Desktop,JDK, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
3. **Pre-Experiment Exercise:**

**Brief Theory:** Refer shared material

1. **Laboratory Exercise**
   * + 1. **Procedure:**

**a. Answer the following:**

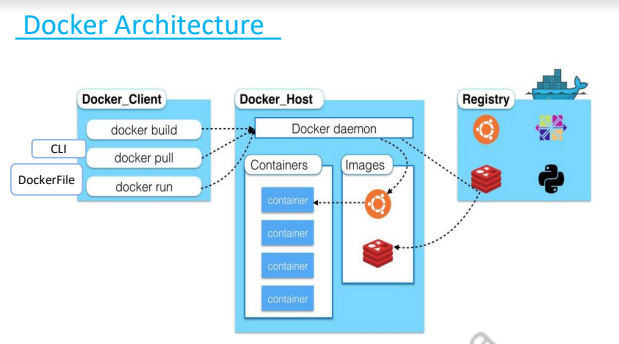
* What are docker containers and docker images?

**Docker Images:**

* A Docker image is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and configuration files.
* Images are read-only templates used to create containers. They are built using a layered filesystem, which helps to optimize storage and allows for efficient use of shared components.

**Docker Containers:**

* A Docker container is a running instance of a Docker image. Containers are isolated environments where applications can run, sharing the OS kernel but with their own filesystem and resources.
* They can be started, stopped, moved, and deleted independently.
* Explain docker architecture with diagram.



Docker's architecture consists of several components:

1. **Docker Daemon (dockerd):** The background service that manages Docker containers, images, networks, and volumes.
2. **Docker Client (docker):** The command-line interface that allows users to interact with the Docker daemon.
3. **Docker Registry:** A repository for storing and distributing Docker images, with Docker Hub being the default public registry.
4. **Docker Objects:** These include images, containers, networks, and volumes.

* What is a Dockerfile?

A **Dockerfile** is a text file that contains a set of instructions on how to build a Docker image. It specifies the base image, environment variables, file copying, installation of dependencies, and commands to run the application.

* Explain Dockerfile commands with syntax and example.
* The FROM command sets the base image for subsequent instructions. Every Dockerfile must start with a FROM command.

FROM <image>[:<tag>]

example=FROM ubuntu:20.04

(sets the base image to Ubuntu 20.04. If no tag is provided, Docker uses latest by default.)

* The RUN command executes commands in the shell during the image build process.

RUN <command>

example:RUN apt-get update && apt-get install -y python3

* The CMD instruction provides defaults for an executing container. It is often used to set the default command to run when a container starts. Only one CMD instruction is allowed in a Dockerfile.

CMD ["executable", "param1", "param2"] # JSON array format

CMD command param1 param2 # Shell form

example=CMD ["python3", "app.py"]

This example tells the container to run python3 app.py when it starts.

* The COPY command copies files or directories from the host machine into the Docker image.

COPY <src> <dest>

example=COPY . /app

This example copies all files from the current directory on the host into the /app directory in the image.

* The ADD command is similar to COPY, but it also allows the extraction of local tar files and can fetch files from remote URLs.

ADD <src> <dest>

example=ADD https://example.com/app.tar.gz /app

This example downloads and extracts a tar file from a URL into the /app directory in the container.

**b**. **Execute following (Refer the shared material) and attach screenshots:**

* + - Create Docker Hub account
* Download and install Docker Desktop
* Execute docker commands to manage images and interact with containers
  + - Create a Dockerfile
* Create an html file
* Build and run the web application on Docker Engine

1. **Post-Experiments Exercise**
2. **Extended Theory:**

Nil

1. **Questions:**

* Write all Docker commands with syntax and example
* Explain differences between VMs and docker containers
* What is a Docker cheat sheet?
* Why do we require volumes for Docker?

1. **Conclusion:**

* Write what was performed in the experiment.
* Write the significance of the topic studied in the experiment.

1. **References:**

<https://www.youtube.com/watch?v=zJ6WbK9zFpI>

<https://www.simplilearn.com/tutorials/docker-tutorial>

<https://www.edureka.co/blog/docker-explained/>

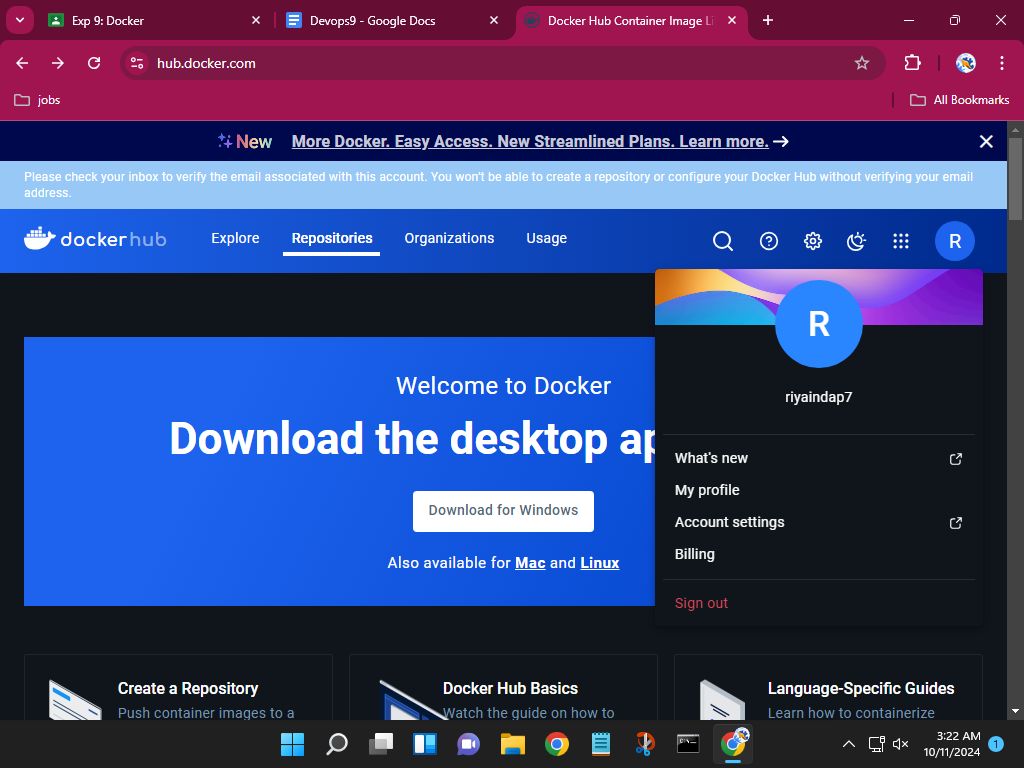
<https://www.youtube.com/watch?v=zJ6WbK9zFpI>

<https://www.simplilearn.com/tutorials/docker-tutorial>

<https://www.edureka.co/blog/docker-explained/>

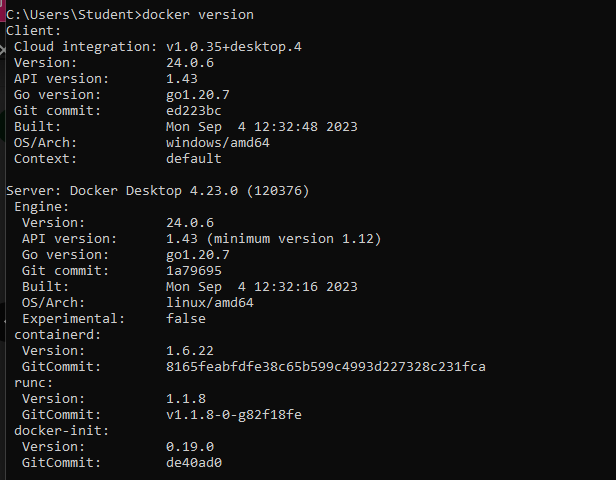
<https://www.youtube.com/watch?v=3c-iBn73dDE>

* Create Docker Hub account – screenshot of steps related to account creation



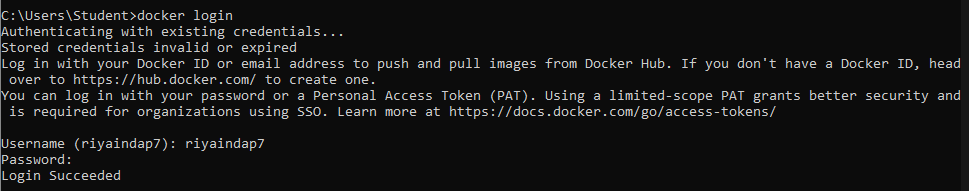
DOCKER COMMANDS

1. Docker version

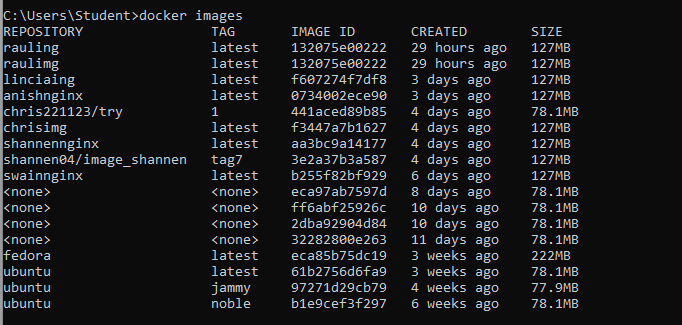




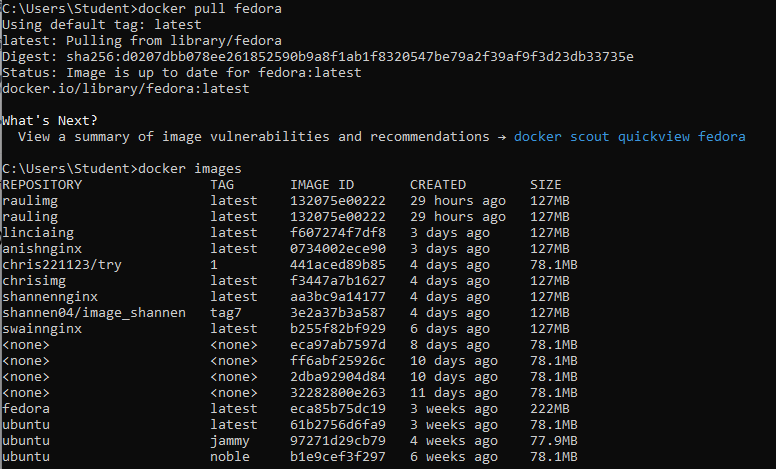
2. Docker login



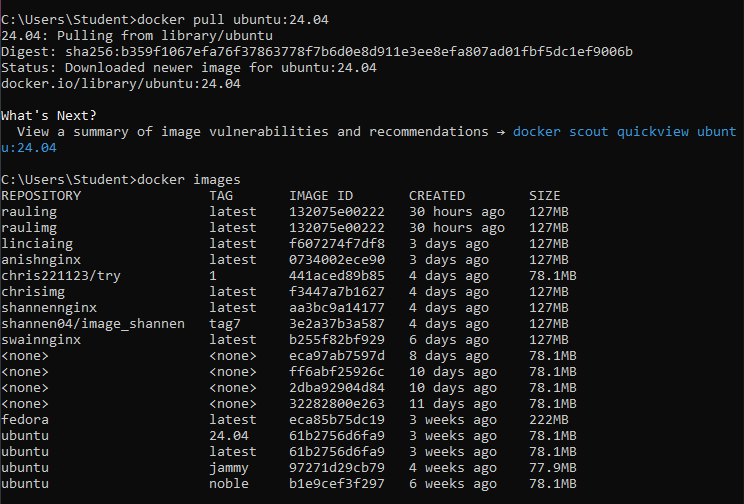
3. Docker images



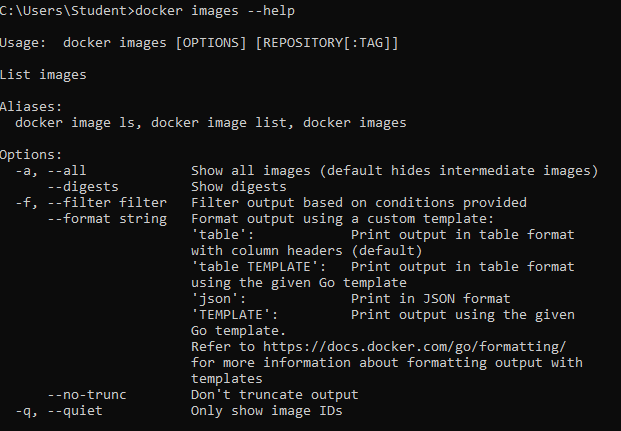
4. Docker pull image

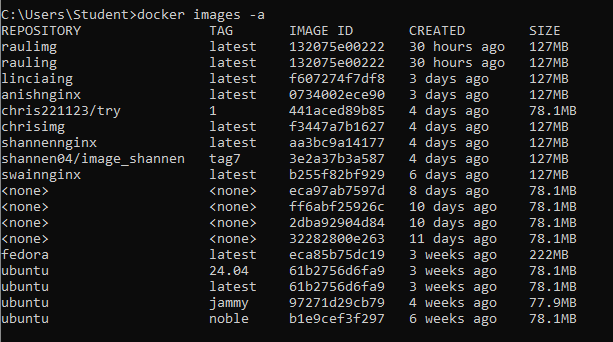


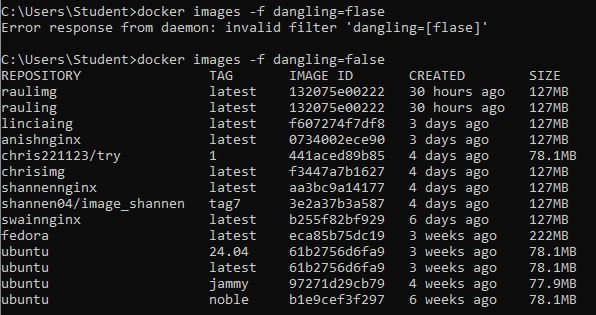
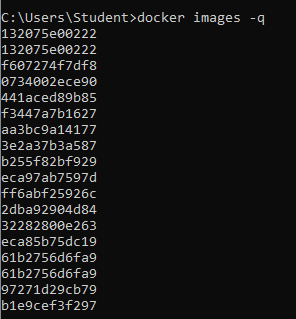
5. Docker pull image-tag



6. Docker images help

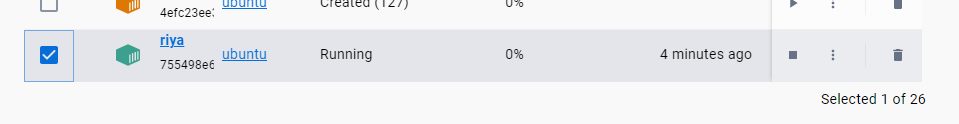




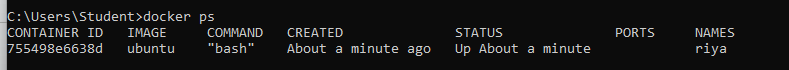


7. Docker run commands

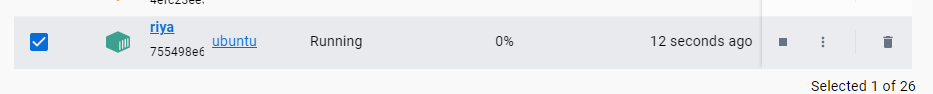




8. Docker ps

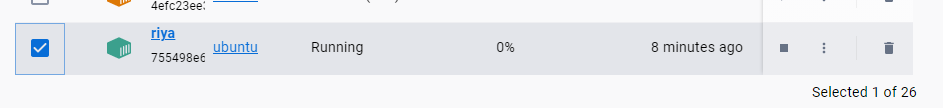
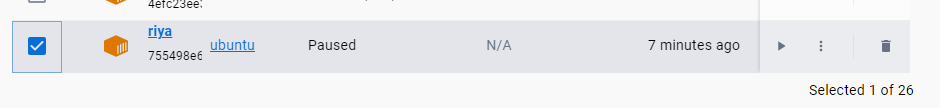


9. Docker start container



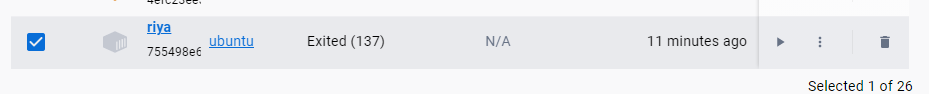
10. Docker pause container





11. Docker stop container

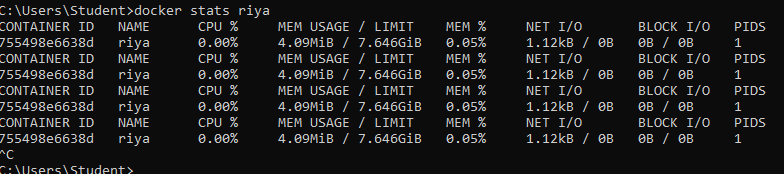


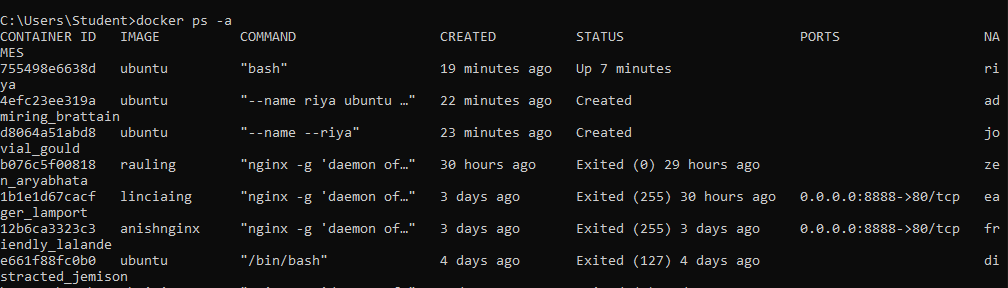
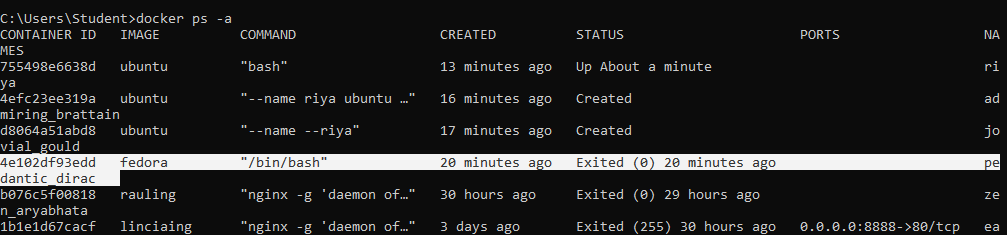


12. Docker top container

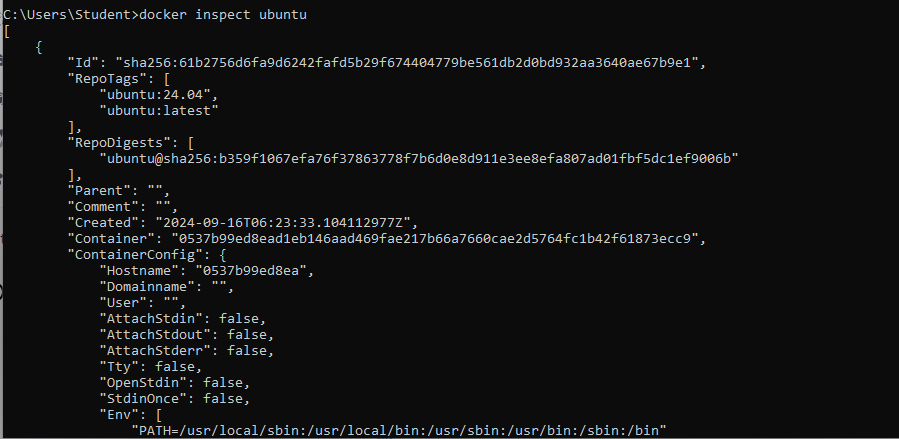


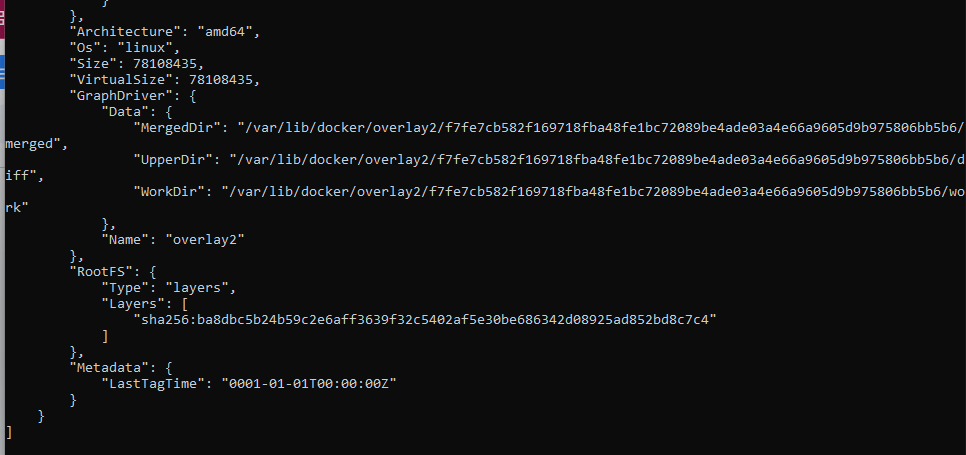
13. Docker stats container



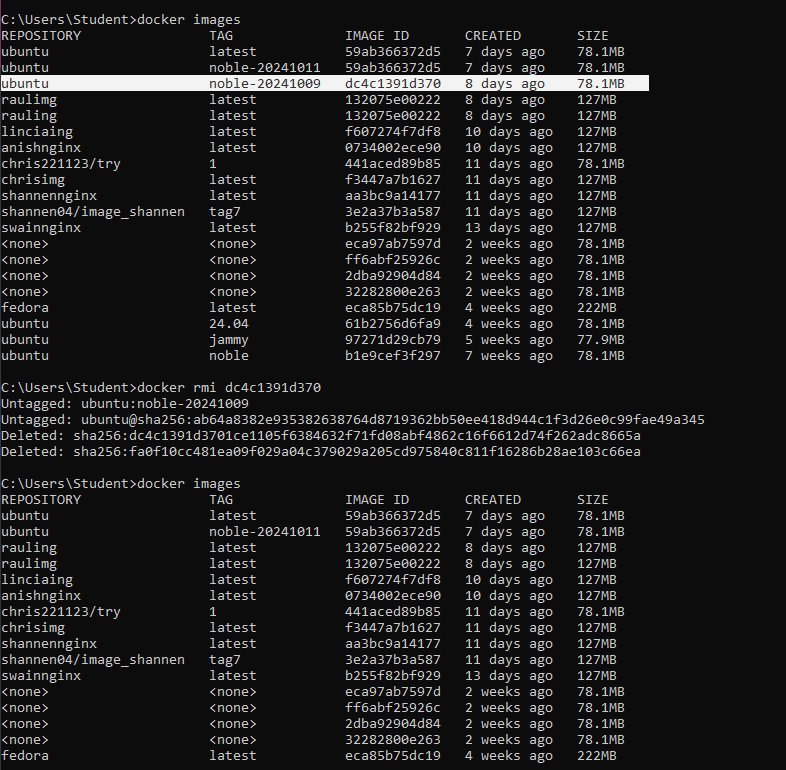
14. Docker rm container

15. Docker inspect





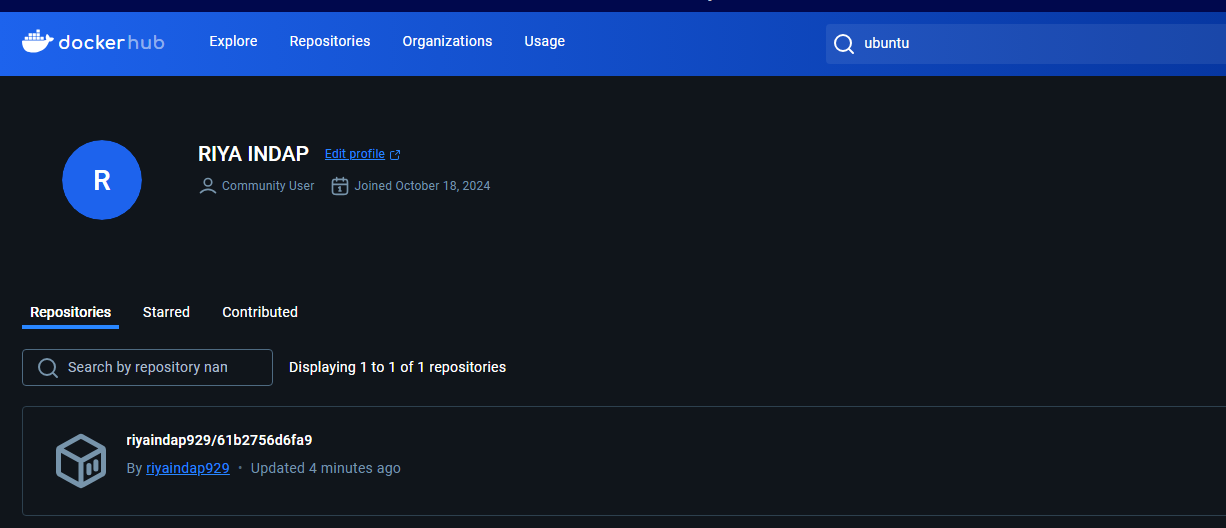
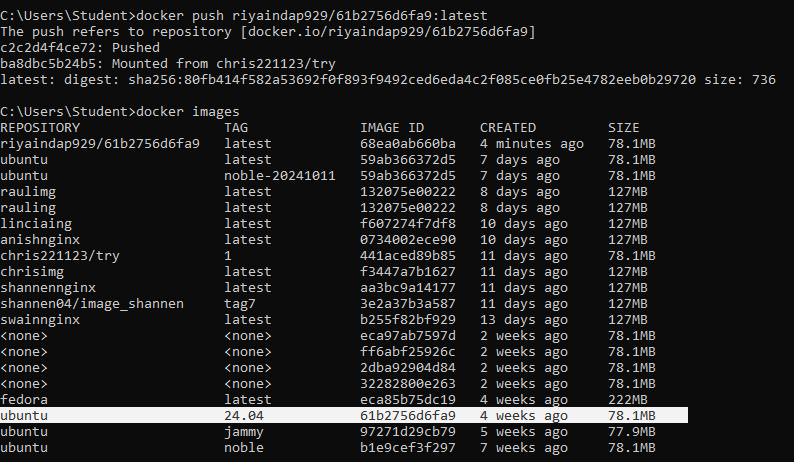
16. Docker rmi



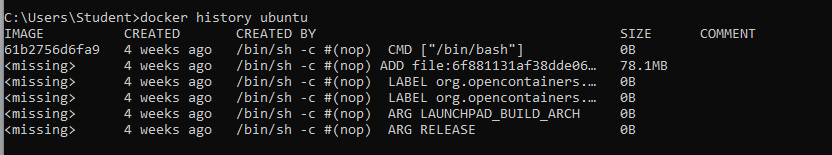
17. Docker commit



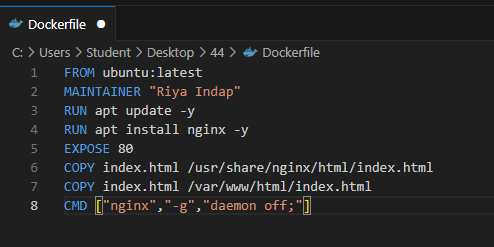
18. Docker push

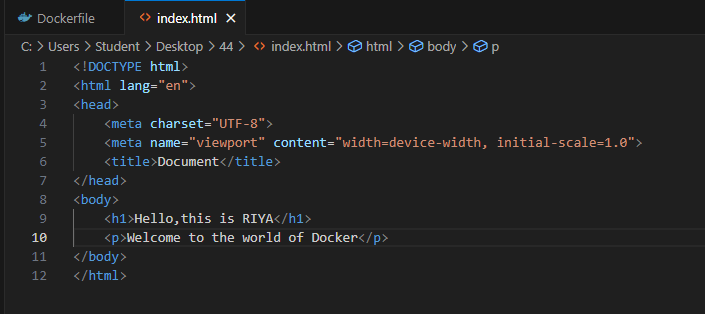


19. Docker history image



* Dockerfile and html file screenshots





* running the web application on Docker Engine screenshots

