**AIR UNIVERSITY, ISLAMABAD**

**Department of Cyber Security**

Secure Software Design & Development Lab (CY- 256L)

Lab Task # 12

**Submitted to:** Sir Mahaz Khan

**Submitted by:** Mirza Shaheer Baig (231330)

**CLASS:** BSCYS-4 A

**Submitted on: 23/05/2025**

# Docker:

## What is Docker?

Docker is an open-source platform that allows you to develop, ship, and run applications inside lightweight, portable containers.

* A Docker container is a standardized unit that packages your code, runtime, system tools, libraries, and settings—everything the app needs to run.
* Containers are isolated from each other and the host system, but they share the OS kernel, making them faster and more efficient than virtual machines (VMs).

## How Docker Works

1. **Dockerfile**: You write a file that defines what goes into the container (like a recipe).
2. **Image**: Docker builds an image from the Dockerfile.
3. **Container**: You run an image to create a container (an instance of the image).

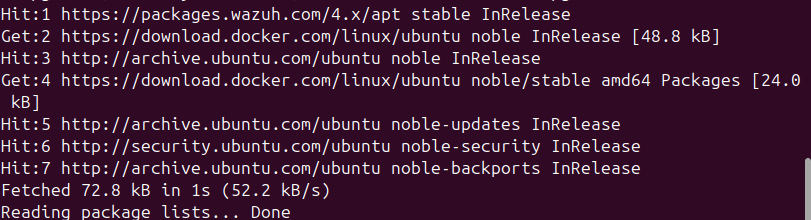
## Uses of Docker

1. **Development Consistency**
   * Developers can run the same code on any machine without worrying about dependencies or environment setup issues.
   * “It works on my machine” problems are eliminated.
2. **Microservices Architecture**
   * Each microservice can run in its own container with its own dependencies.
   * Easy to scale and manage.
3. **CI/CD Pipelines**
   * Integrates seamlessly into Continuous Integration/Continuous Deployment workflows.
   * Automated testing, building, and deployment are simplified.
4. **Environment Replication**
   * Easily replicate production-like environments for testing or staging.
5. **Isolation and Security**
   * Containers isolate applications from each other and from the host system.
6. **Portability**
   * Docker containers can run on any system that supports Docker: Linux, Windows, cloud providers (AWS, Azure, GCP), and even your laptop.
7. **Resource Efficiency**
   * Containers are more lightweight than VMs and use fewer system resource

# Docker Setup:

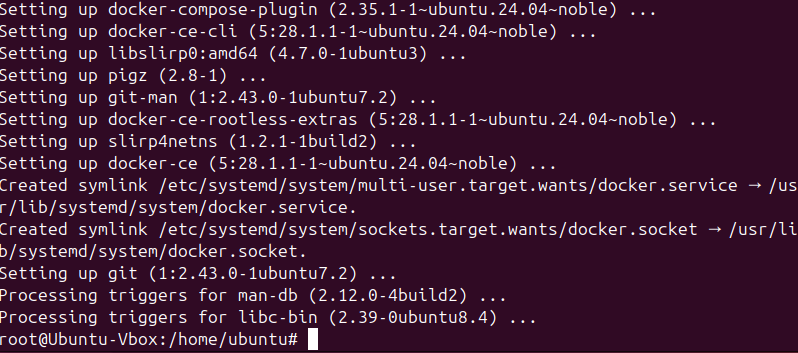
## Step 1: Install Prerequisites

First, let's make sure your system is up to date and install required dependencies:



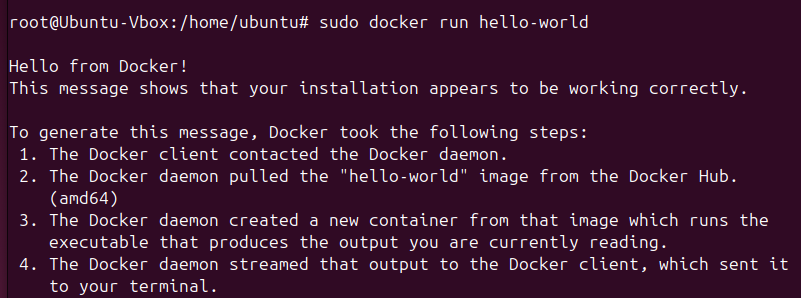
Step 2: Add Docker's Official GPG Key

Now let's install Docker and related components:

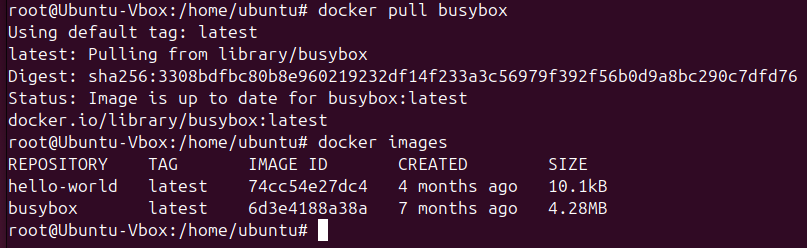


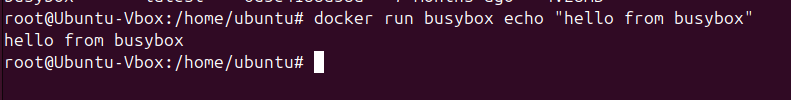
## Step 3: Verify Docker Installation

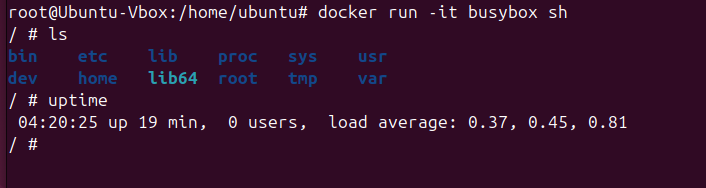
Let's make sure Docker is installed correctly by running the hello-world image:



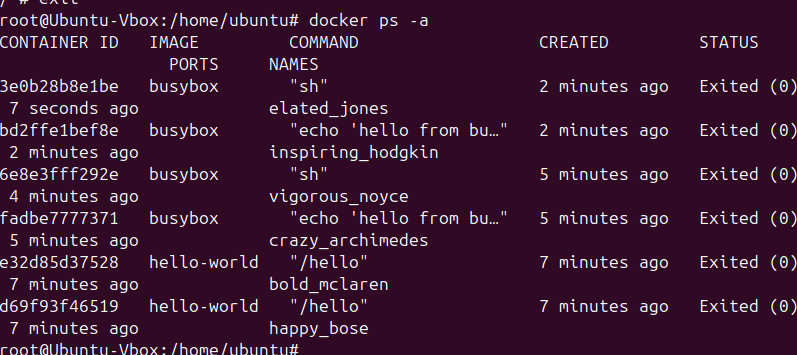
Step 4: Working with Busybox



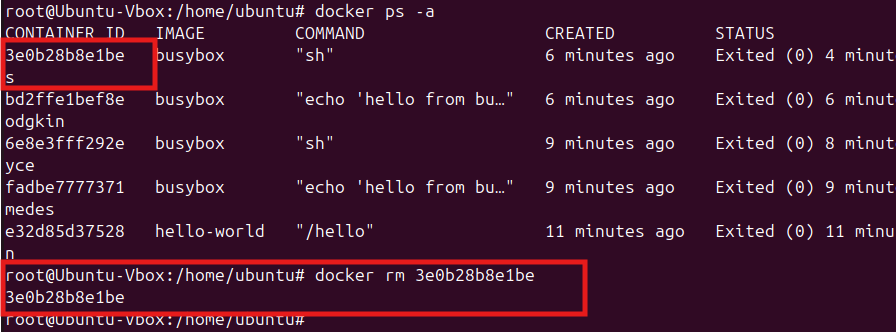




## **Step 5: Exiting Docker:**

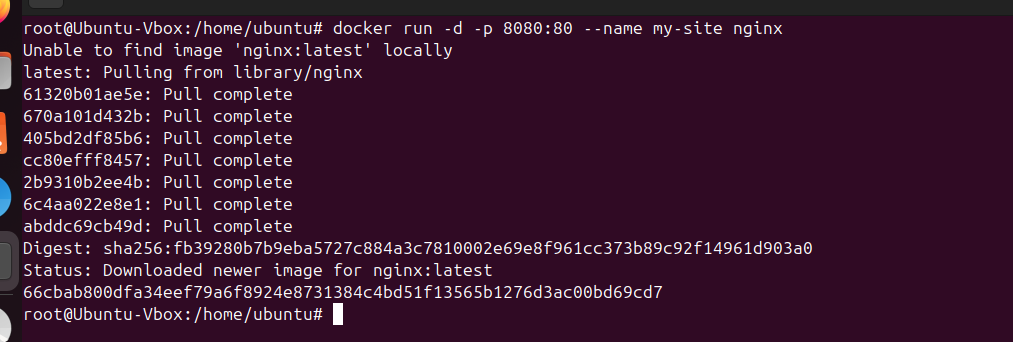


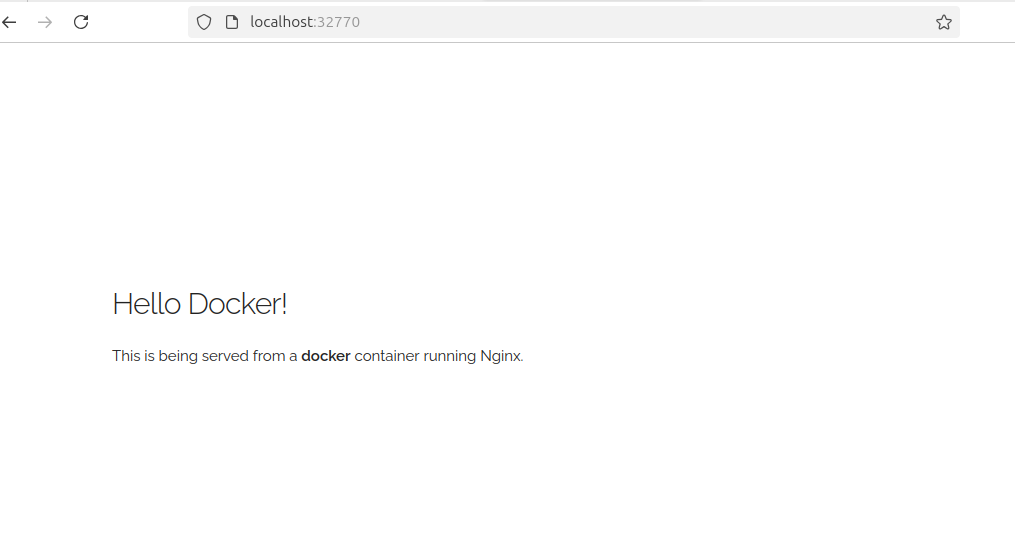
## Step 6: Deleting a container:



## **Step 7: Running a Web Application**

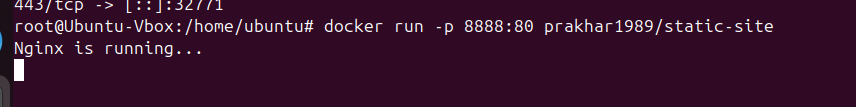
Let's try running a simple web application in a container:

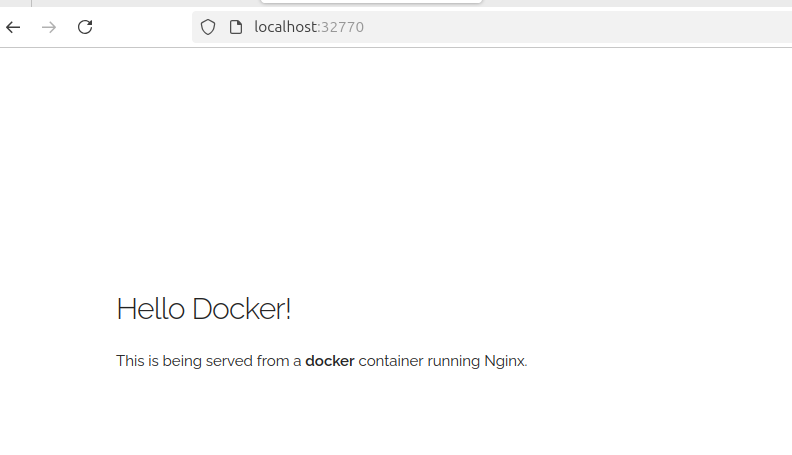




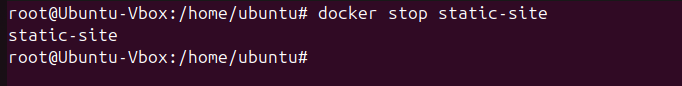
We can also specify a custom port to which the client will forward

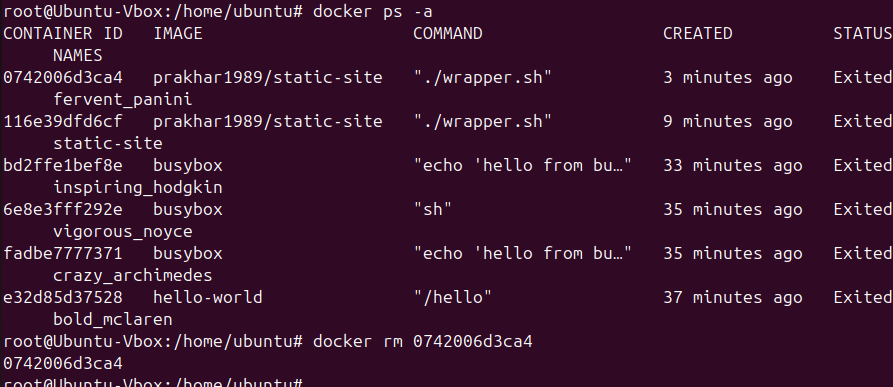
connections to the container.



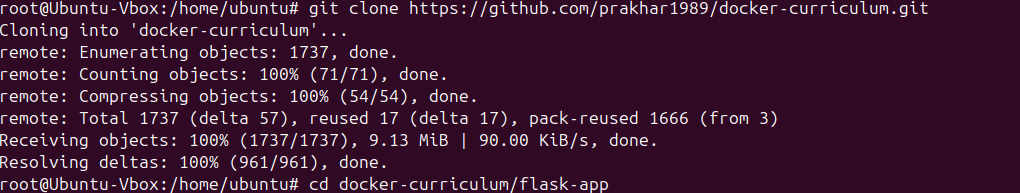


Stopping my website and deleting the docker container

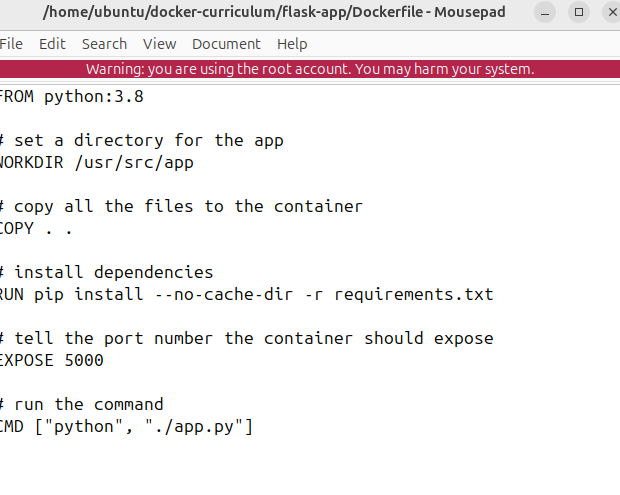




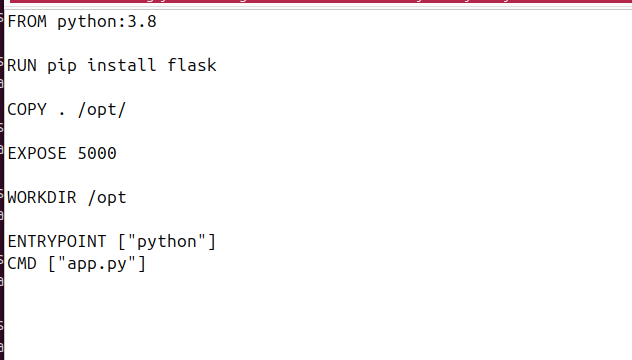
Step 8: Creating Your Own Image



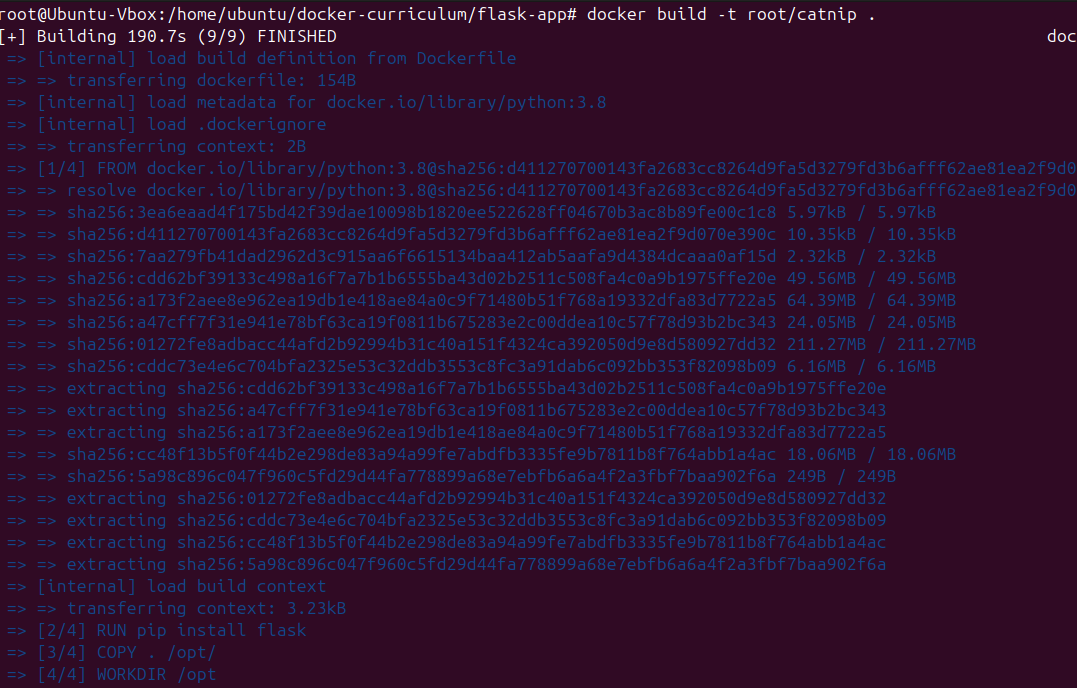
With that, our Docker file is now ready. This is how it looks



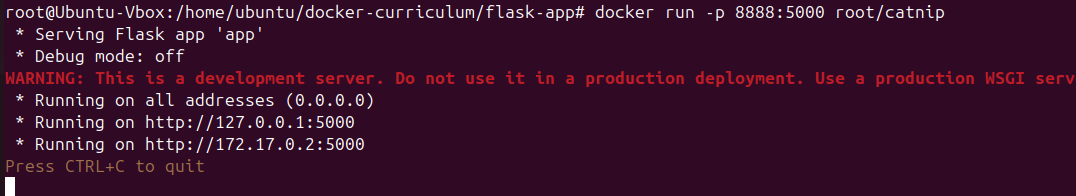
Now that we have our Dockerfile, we can build our image. The docker build command does the heavy-lifting of creating a Docker image from a Dockerfile.



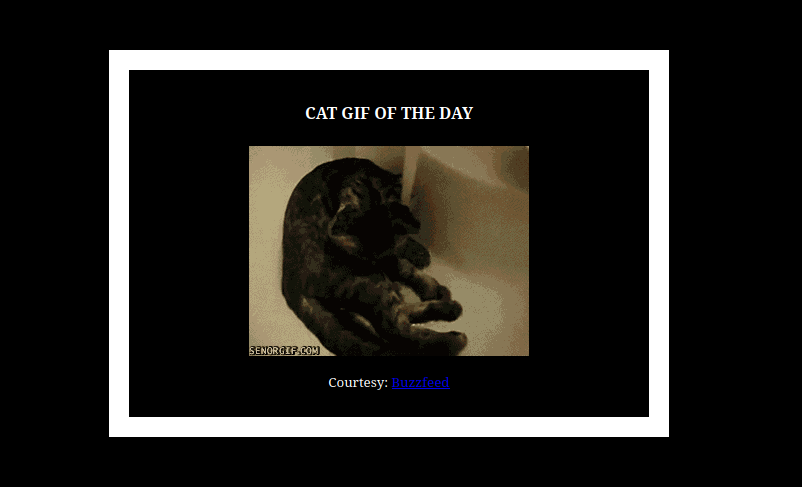
Step 9: Building Image:



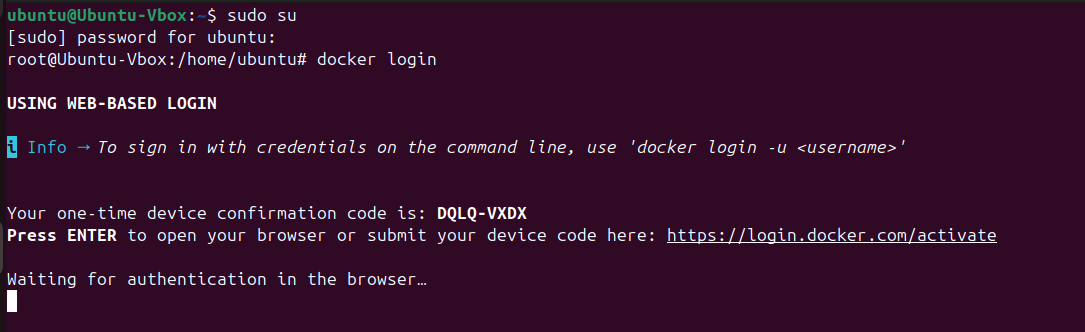
## Step 10: Running Image:

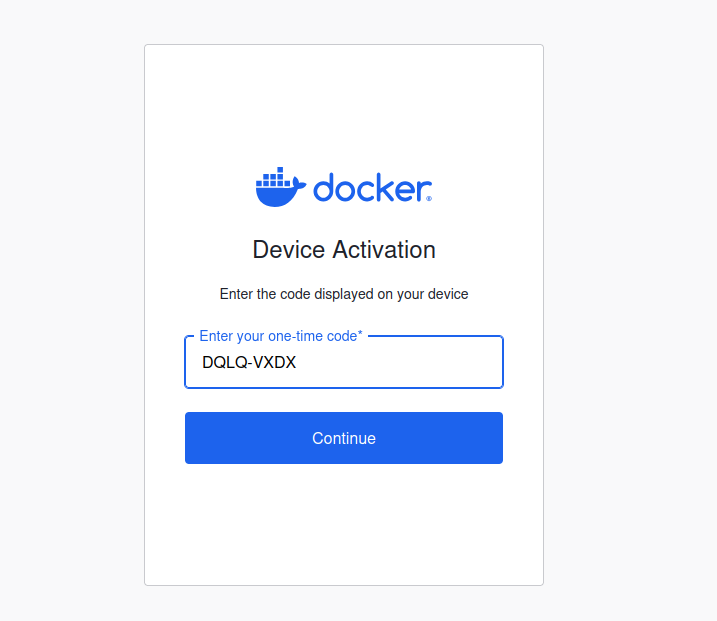






# Docker push

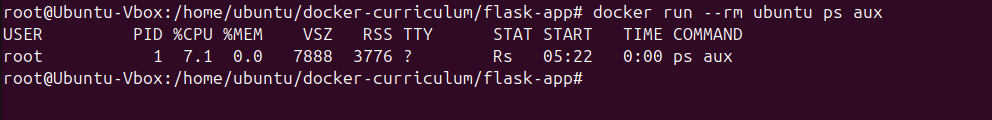




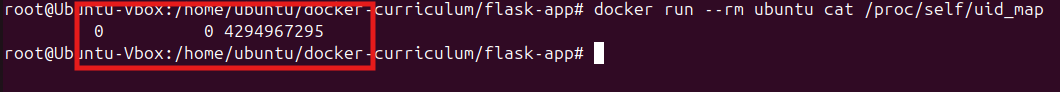
# Security Checks

* Run the commands given in Rule 2, Rule 3, Rule 8 to see if they work fine.

Check that containers can't access host processes (Rule 2)



Check user namespace isolation (Rule 3)



Check for container resource limits (Rule 8)

* In Rule 9 tools are suggested to detect containers with known vulnerabilities - scan images.
* Use any of these free tools (preferably Trivy) to test the image you have created above in the task or any other docker image.

