# Placement Empowerment Program

***Cloud Computing and DevOps Centre***

Run Multiple Docker Containers and Monitor Them: Run multiple containers (e.g., Nginx and MySQL) and monitor their resource usage.

Name: Sharon Hanna A Department:Ads



Docker is a containerization platform that allows developers to package applications and their dependencies into isolated environments called **containers**. Running multiple containers efficiently is crucial for microservices-based architectures. In this Proof of Concept (POC), we will deploy and manage multiple Docker containers—**Nginx** (a web server) and **MySQL** (a database). We will also monitor their resource usage using docker stats.

# Overview

This POC demonstrates the process of:

### Setting up Docker on Windows

1. **Running multiple containers** (Nginx and MySQL)
2. **Managing containers** (starting, stopping, removing)
3. **Monitoring container resource usage** (CPU, memory, network, and disk I/O)

We will use:

docker run to launch the containers docker ps to check running containers

docker stats to monitor container performance

1. Understand the fundamentals of **Docker containerization**.
2. Learn how to **deploy multiple containers** using the Docker CLI.
3. Gain hands-on experience with **managing containerized applications**.
4. Explore **resource monitoring techniques** for containerized applications.
5. Learn to troubleshoot **performance issues** using docker stats.

# Importance

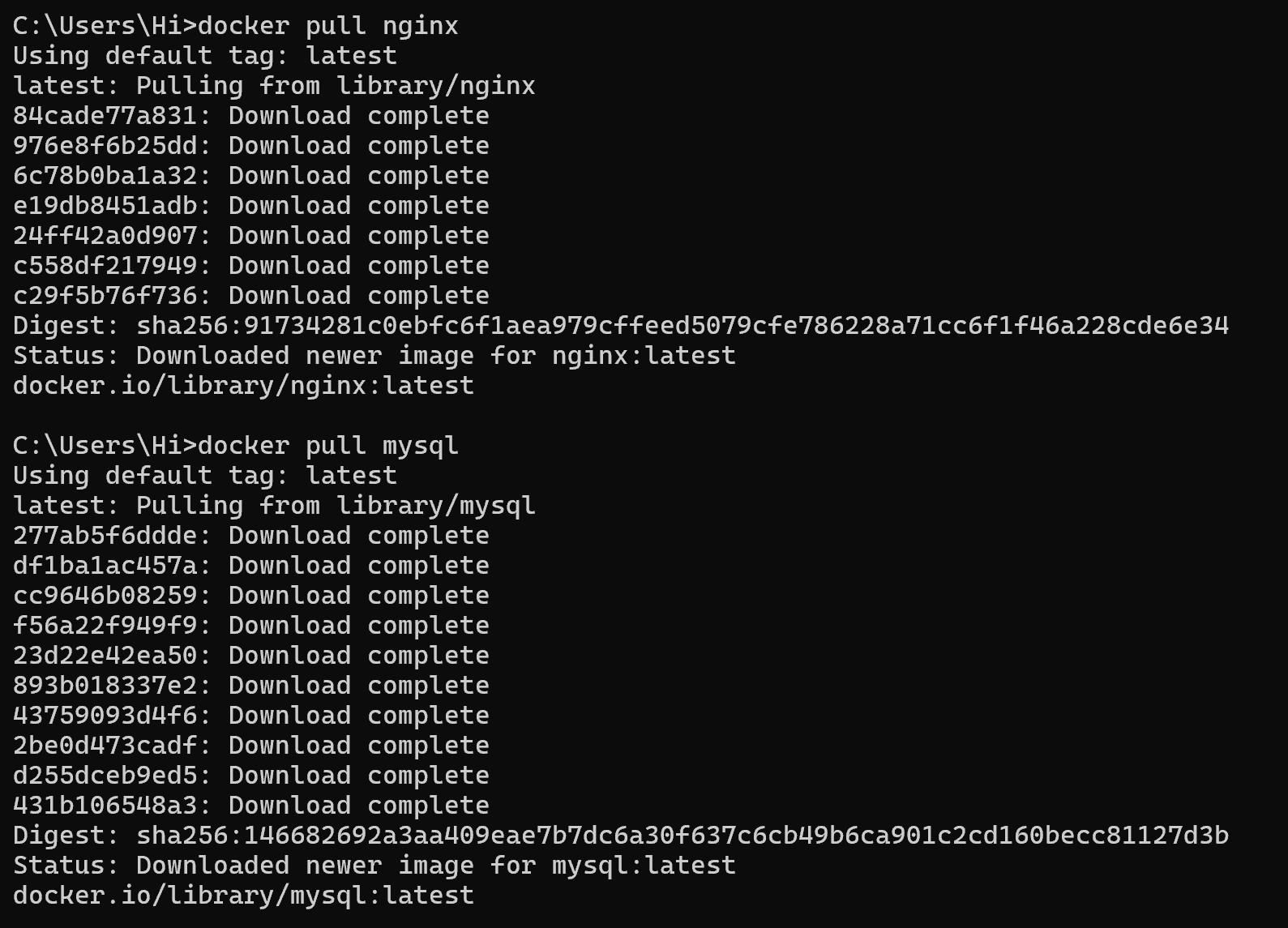
1. **Real-World Relevance** – Running multiple containers is essential for building scalable applications in **DevOps** and **Cloud environments**.
2. **Microservices & Scalability** – Modern applications rely on **multiple services** running in separate containers, such as **frontend, backend, and database services**.
3. **Performance Optimization** – Monitoring CPU, memory, and network usage helps **optimize resource allocation**, preventing application slowdowns.
4. **Foundation for Kubernetes & Docker Compose** – Understanding container monitoring lays the groundwork for **orchestrating containers using Kubernetes or Docker Compose**.

## Step 1:

Pull the Required Docker Images

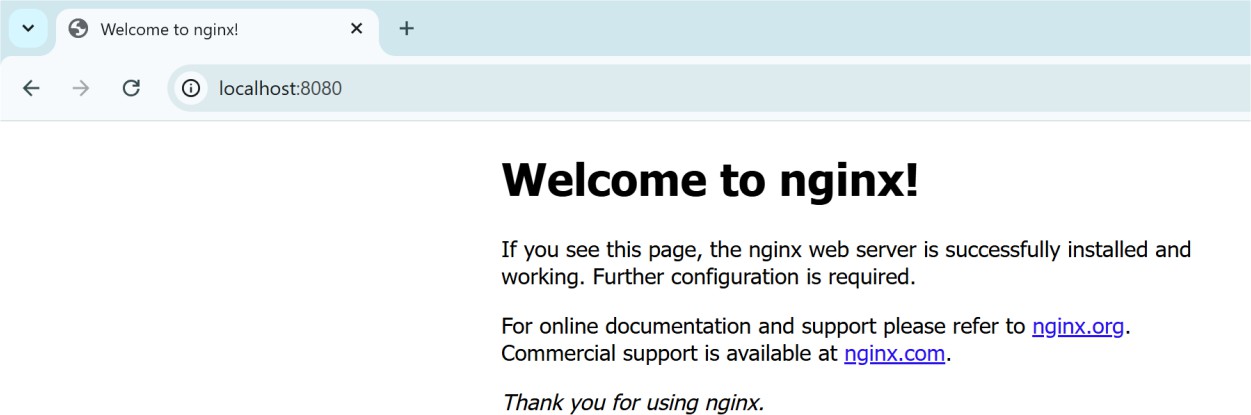
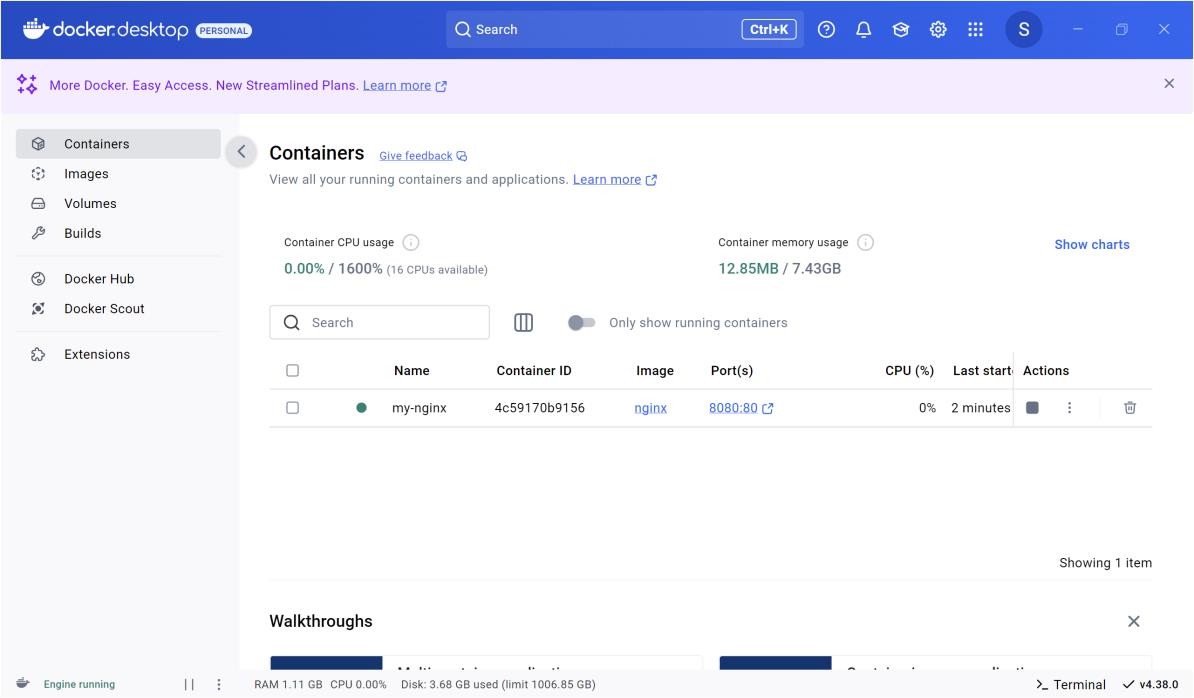
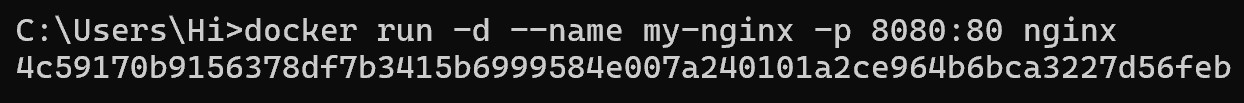
Before running the containers, pull the necessary images from Docker Hub.

### docker pull nginx docker pull mysql

****

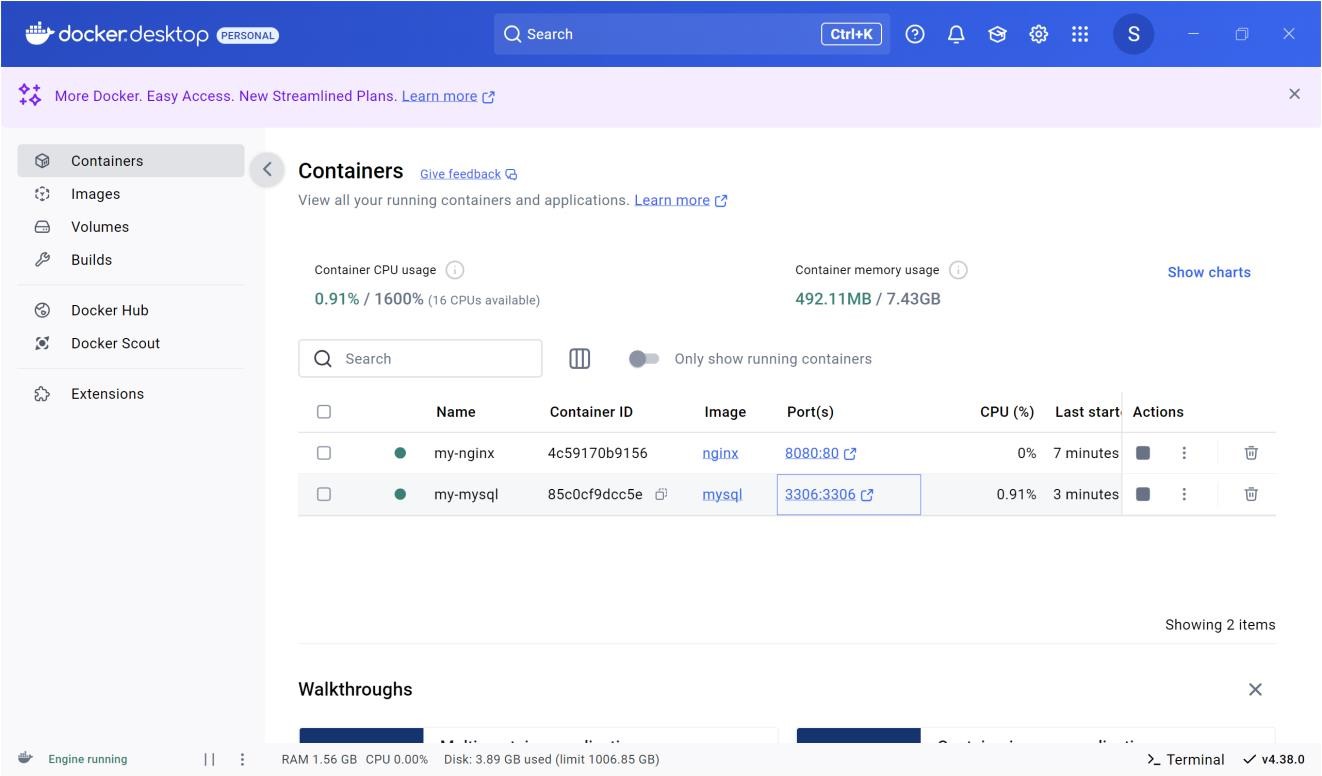
Run an **Nginx** container in detached mode (-d), mapping port 8080 on your host to port 80 inside the container. Verify it by Opening a new tab and search for **localhost:8080**

### docker run -d --name my-nginx -p 8080:80 nginx

****

Run a **MySQL** container with environment variables for database credentials.

**docker run -d --name my-mysql -e MYSQL\_ROOT\_PASSWORD=rootpassword -e MYSQL\_DATABASE=mydb -p 3306:3306 mysql**

****

## Step 4:

To check if the containers are running, use:

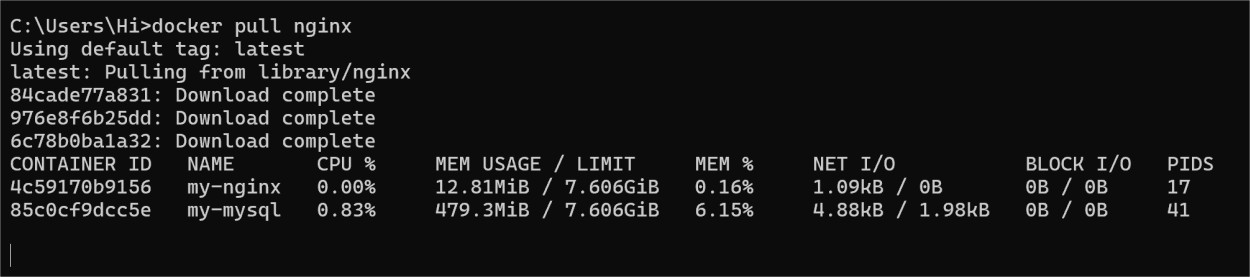
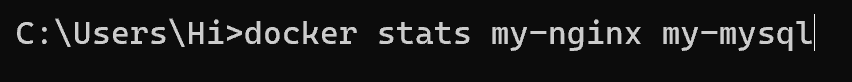
### docker ps

This will show a list of active containers with details like container ID, image, ports, and status.



To monitor specific containers:

**docker stats my-nginx my-mysql**

****

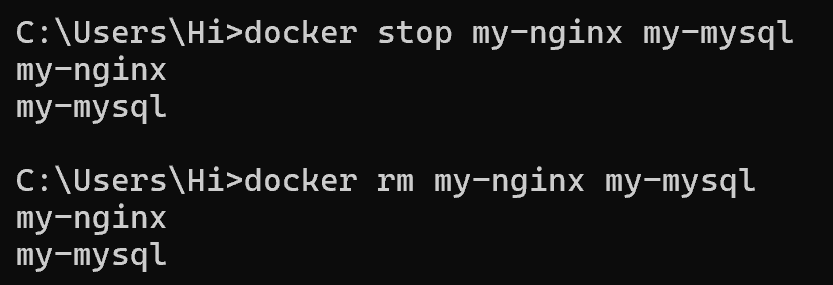
## Step 6:

To stop the containers:

### docker stop my-nginx my-mysql

To remove the containers:

**docker rm my-nginx my-mysql**

****

# Outcomes

By completing this POC, you will:

1. **Run Multiple Containers** – Deploy and manage multiple containers (Nginx and MySQL) simultaneously.
2. **Use Essential Docker Commands** – Gain hands-on experience with docker run, docker ps, docker stop, and docker rm for container management.
3. **Monitor Container Resource Usage** – Learn to track CPU, memory, and network usage using docker stats.
4. **Expose and Access Services** – Map host ports to container ports to interact with running applications (Nginx on port 8080, MySQL on 3306).
5. **Set Up and Manage Environment Variables** – Use -e flags to configure MySQL credentials inside a container.
6. **Understand Containerization Benefits** – Explore how Docker simplifies application deployment, enhances portability, and optimizes resource management.
7. **Perform Cleanup Operations** – Learn how to free up system resources by removing unused containers and images using docker system prune -a.