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: Sylashri Rajendran
Department: IT

Introduction

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:

1. **Setting up Docker on Windows**
2. **Running multiple containers** (Nginx and MySQL)
3. **Managing containers** (starting, stopping, removing)

4. 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-by-Step Overview

Step 1:

Pull the Required Docker Images

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

docker pull nginx docker

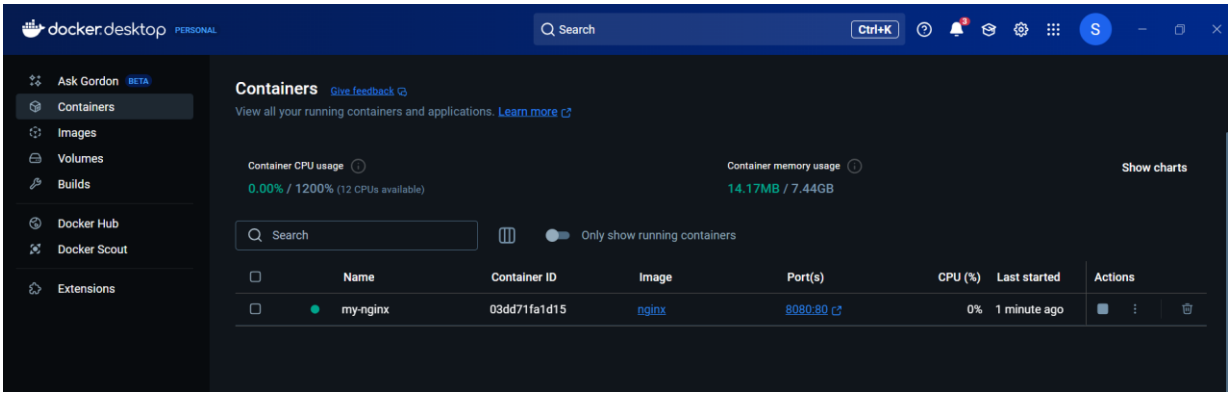
pull mysql

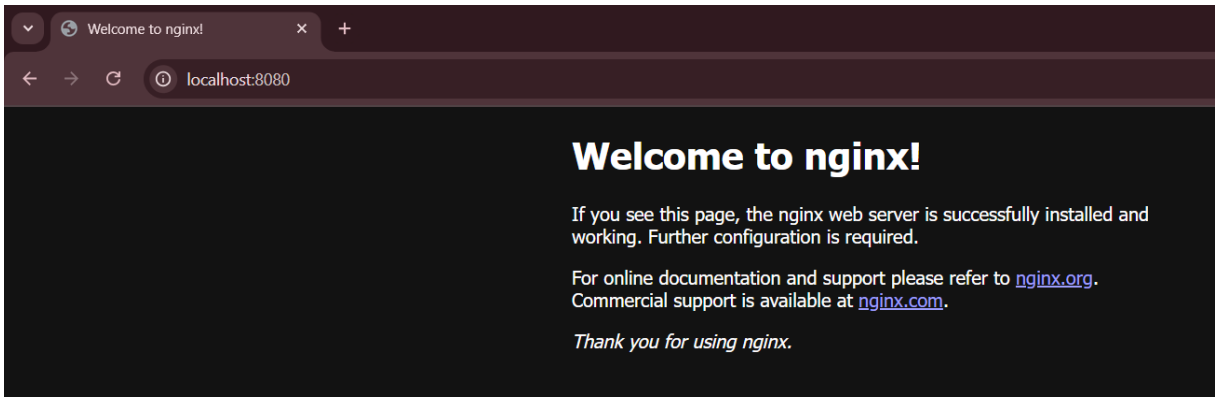
```
C:\Users\sylas>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:124b44bfc9ccd1f3cedf4b592d4d1e8bddb78b51ec2ed5056c52d3692baebc19
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

C:\Users\sylas>docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
69e76254f502: Pull complete
bc0c792ca096: Pull complete
1b515e7ceb69: Pull complete
eaa11c0a9f08: Pull complete
8d73d2a73425: Pull complete
8d18181893b8: Pull complete
4a7e00d873b9: Pull complete
e0a910cc8604: Pull complete
27a2553d6a80: Pull complete
804bb8ae89de: Pull complete
Digest: sha256:9b9d0aab4860798acff13d2a0ece3bc26639fe18b83fa5cd3e3d0e16b3ed05dd
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

Step 2:

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**

```
C:\Users\sylas>docker run -d --name my-nginx -p 8080:80 nginx
03dd71fa1d15371e2230639acdc98d3abea93fe8d936455039eb90aba32d2a23
```





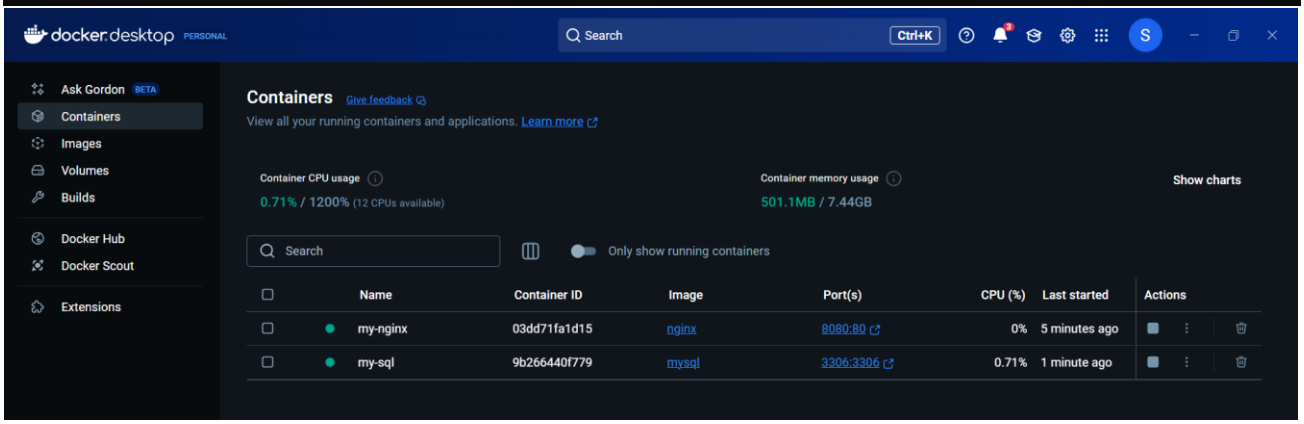
Step 3:

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

```
C:\Users\sylas>docker run -d --name my-sql -e MYSQL_ROOT_PASSWORD=rootpassword -e MYSQL_DATABASE=mydb -p 3306:3306 mysql
9b266440f7792c4c1be98af7bf49caa9650a41091df895872728d77524d7d2da
```



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.

```
C:\Users\sylas>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                               NAMES
9b266440f779   mysql     "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->3306/tcp, 33060/tcp   my-sql
03dd71fa1d15   nginx     "/docker-entrypoint.…"   6 minutes ago        Up 6 minutes        0.0.0.0:8080->80/tcp                my-nginx
```

Step 5:

To monitor specific containers: **docker stats**

my-nginx my-mysql

```
C:\Users\sylas>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

```
C:\Users\sylas>docker stop my-nginx my-mysql
my-nginx
my-mysql

C:\Users\sylas>docker rm my-nginx my-mysql
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.