



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: SUBASHINI P

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

pull mysql

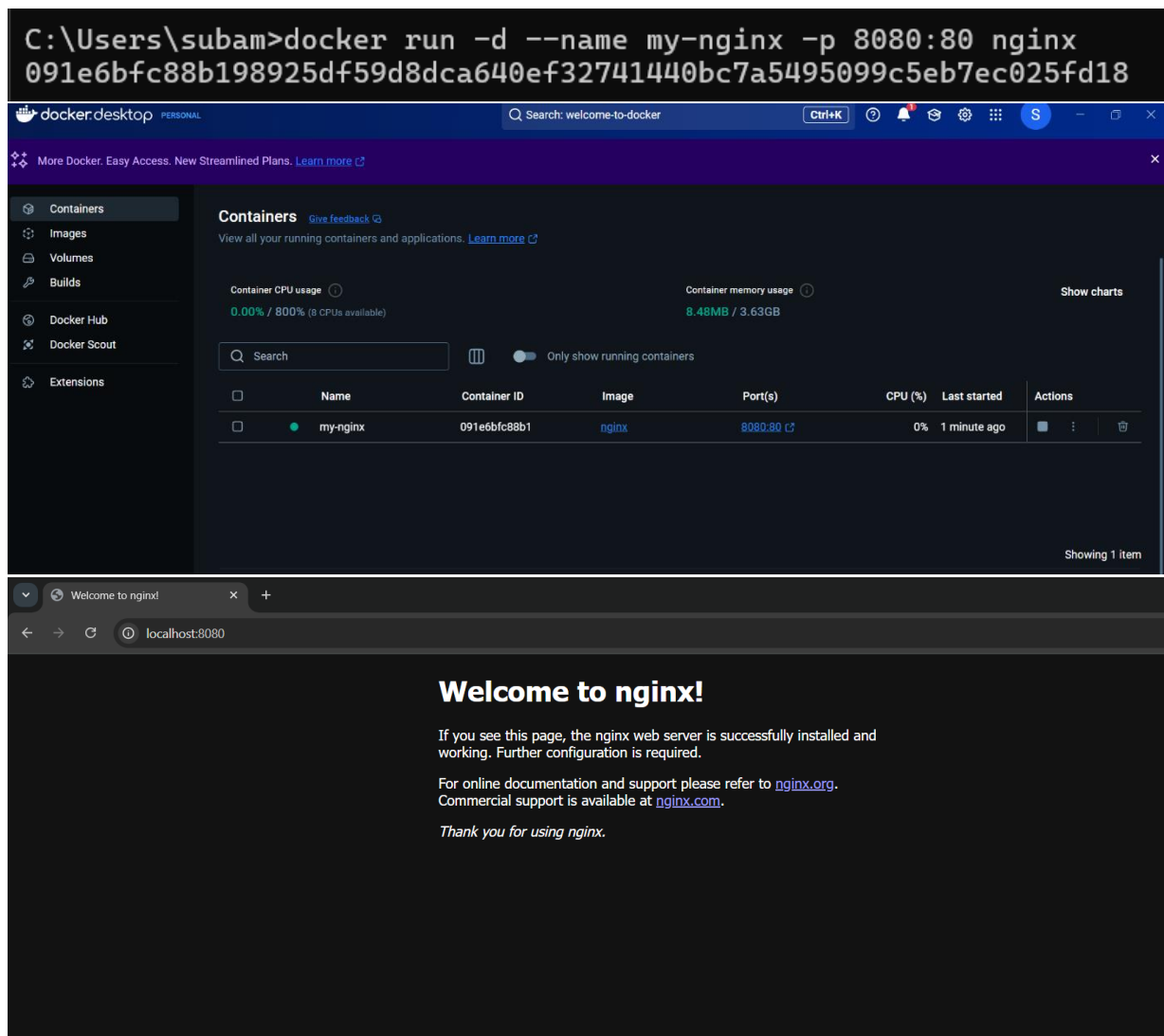
```
C:\Users\subam>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:9d6b58feebd2dbd3c56ab5853333d627cc6e281011cfd6050fa4bcf2072c9496
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

C:\Users\subam>docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
43759093d4f6: Download complete
23d22e42ea50: Download complete
d255dceb9ed5: Download complete
431b106548a3: Download complete
2be0d473cadf: Download complete
f56a22f949f9: Download complete
277ab5f6ddde: Download complete
893b018337e2: Download complete
df1ba1ac457a: Download complete
cc9646b08259: Download complete
Digest: sha256:146682692a3aa409eae7b7dc6a30f637c6cb49b6ca901c2cd160becc81127d3b
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

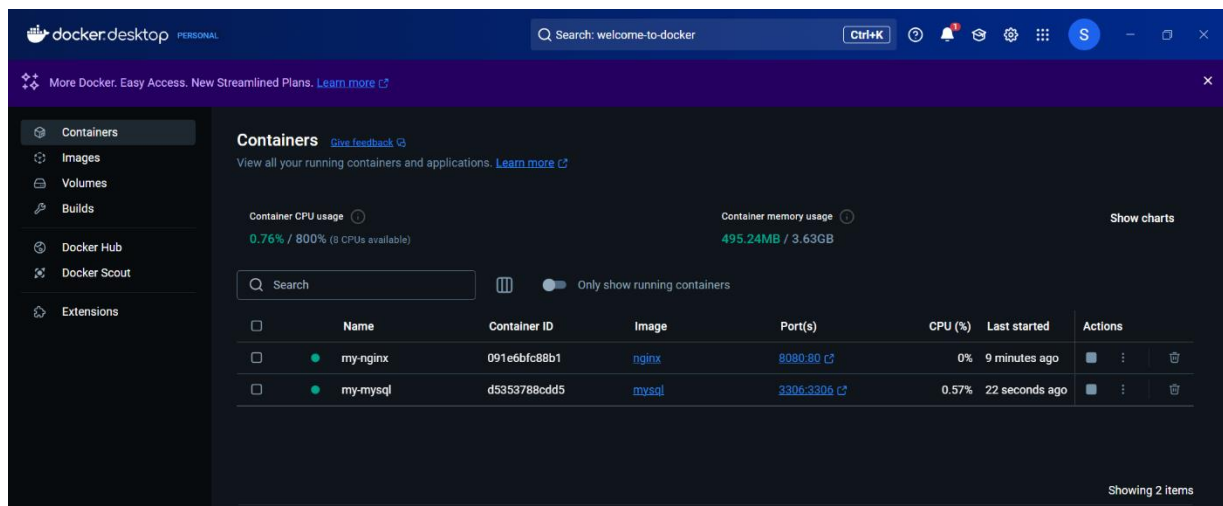


Step 3:

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\subam>docker run -d --name my-mysql -e MYSQL_ROOT_PASSWORD=rootpassword -e MYSQL_DATABASE=mydb -p 3306:3306 mysql  
d5353788cdd57c5cb87f9823b6f6ce5a76e759291ef650eb3cd7c294ec812294
```



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\subam>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
d5353788cdd5   mysql    "docker-entrypoint.s..." 42 seconds ago Up 42 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp  my-mysql
091e6bfc88b1   nginx    "/docker-entrypoint..." 9 minutes ago  Up 9 minutes  0.0.0.0:8080->80/tcp              my-nginx
```

Step 5:

To monitor specific containers: **docker**

stats my-nginx my-mysql

```
C:\Users\subam>docker stats my-nginx my-mysql
```

```
C:\Users\subam>docker pull nginx
Using default tag: latest
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O     PIDS
091e6bfc88b1   my-nginx  0.00%     7.238MiB / 3.721GiB  0.19%     3.02kB / 2.15kB  0B / 0B       9
d5353788cdd5   my-mysql  0.89%     475.7MiB / 3.721GiB  12.48%     746B / 0B       0B / 0B       35
```

Step 6:

To stop the containers: **docker**

stop my-nginx my-mysql To

remove the containers:

docker rm my-nginx my-mysql

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

C:\Users\subam>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.