

**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: JOSHUA MOSES Department: IT



**Objective**

The objective of this PoC is to demonstrate how to run multiple Docker containers, manage them efficiently, and monitor their resource usage, including CPU, memory, and network utilization.

**Prerequisites**

Before proceeding, ensure you have the following:

* Docker installed on your system ([Installation Guide)](https://docs.docker.com/get-docker/)
* Basic understanding of Docker commands
* Docker images for Nginx and MySQL **Step-by-Step Guide**

Step 1: Pull the Required Docker Images

Docker images are pre-built environments that allow containers to run specific applications. To pull the required images:

1. Open a terminal or command prompt.
2. Pull the latest Nginx and MySQL images using the following commands:

**docker pull nginx:alpine**  **docker pull mysql:latest** o The nginx:alpine image is a lightweight version of Nginx.

The mysql:latest image fetches the latest version of

MySQL.

Step 2: Run Multiple Containers

After pulling the necessary images, start the containers.

Run the Nginx Container

1. Start an Nginx container using the command:

**docker run -d --name my-nginx -p 8080:80 nginx:alpine** o -d runs the container in detached mode, meaning it runs in the background. o --name my-nginx assigns the name my-nginx to the container. o -p 8080:80 maps port 80 in the container to port 8080 on the host machine.

1. Verify that the container is running: **docker ps** o This command lists all running containers.

o Check if my-nginx is listed.

Run the MySQL Container

1. Start a MySQL container using the command:

**docker run -d --name my-mysql -e MYSQL\_ROOT\_PASSWORD=root -p 3306:3306 mysql:latest**

o -e MYSQL\_ROOT\_PASSWORD=root sets the

MySQL root password to root.

-p 3306:3306 maps MySQL's default port (3306) in the container to port 3306 on the host machine.

1. Verify that MySQL is running by listing the running containers:

**docker ps**

Step 3: Monitor Running Containers

To ensure efficient resource usage, monitor the running containers.

List Running Containers

1. Display all running containers: **docker ps**

o Shows container names, IDs, and ports they are mapped to.

Monitor Resource Usage

1. Use the docker stats command to check resource usage: **docker stats**

* Displays real-time CPU, memory, and network usage for each running container.
* Helps in identifying resource-intensive containers.

Step 4: Manage Running Containers

To clean up and free resources, manage the containers properly.

Stop Containers

1. To stop both containers:

**docker stop my-nginx my-mysql**

Gracefully stops the containers.

Remove Containers

1. After stopping the containers, remove them:

**docker rm my-nginx my-mysql** o Deletes the containers permanently.

Remove Docker Images

1. To free up storage, remove the downloaded images:

**docker rmi nginx:alpine mysql:latest**

o Deletes the images from your system.

**References**

* [Docker Monitoring](https://docs.docker.com/config/containers/resource_constraints/)
* [Docker Commands Cheat Sheet](https://dockerlabs.collabnix.com/docker/cheatsheet/)