**Experiment 11 - Docker Compose**

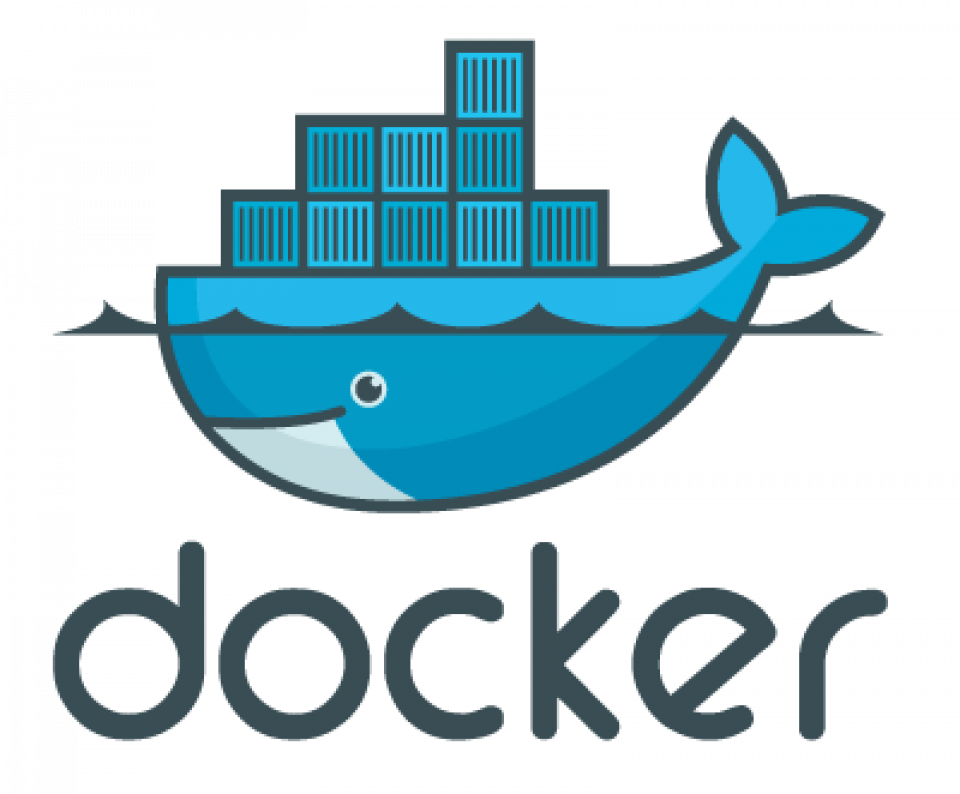
| Roll No. | 37 |
| --- | --- |
| Name | Mikil Lalwani |
| Class | D15-B |
| Subject | DevOps Lab |
| LO Mapped | LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.  LO5: To understand the concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker. |
|

**Aim**:

Docker Compose – multi-container tool.

**Introduction**:

**What is Docker Compose?**



Docker Compose is a tool you can use to centrally manage the deployments of many different

Docker containers. It’s an important tool for any application that needs multiple microservices,

as it allows each service to easily be in a separately managed container.

**What Does Docker Compose Do?**

Docker containers are used for running applications in an isolated environment. It’s quite

common nowadays to see application deployments done in Docker for the numerous benefits it

brings. However, it’s often not as simple as just running a single container. Usually, you may

have many containers coming together to act as one cohesive service made up of many moving

parts.

Managing all of these at deployment time is messy, so to clean it up, Docker provides Docker

Compose, a configuration tool used for running multiple containers at once. You can define all of

the configurations in one YAML file, and then start all the containers with one command.

Rather than having all your services in one big container, Docker Compose allows you to split

them up into individually manageable containers. This is both better for building and

deployment, as you can manage all of them in separate codebases, and don’t need to manually

start each individual container.

Using Docker Compose is a three-step process:

* Build the component images using their Dockerfiles, or pull them from a registry.
* Define all of the component services in a docker-compose.yml file.
* Run all of them together using the docker-compose CLI.

Docker Compose isn’t another kind of Dockerfile. You will still need to build and publish your

Docker containers using a Dockerfile. But, instead of running them directly, you can use Docker

Compose to manage the configuration of a multi-container deployment.

**Basic Commands in Docker Compose-**

* Start all services: Docker Compose up
* Stop all services: Docker Compose down
* Install Docker Compose using pip: pip install -U Docker-compose
* Check the version of Docker Compose: Docker-compose-v
* Run Docker Compose file: Docker-compose up -d
* List the entire process: Docker ps
* Scale a service - Docker Compose up -d -scale
* Use YAML files to configure application services - Docker Compose.yml

**Building With Docker Compose-**

Docker Compose can also be used within a Dockerfile project and can be set up to build and run

an image locally rather than pulling from the Docker Hub.

Demo:

In this demo, we will create two containers (nginx and MySQL) and set them up using one

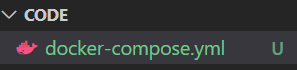
docker-compose file.

Steps:

1. Setup your environment. Create a new folder code and open VSCode in that folder.



2. Create a docker-compose.yml file inside this folder.



3. Inside docker-compose.yml, add the following data to create an nginx container and a

MySQL container.

In this file, you specify the details of both the containers you are going to create.

We also specify the ports on which both containers will run along with the credentials for

the MySQL DB.

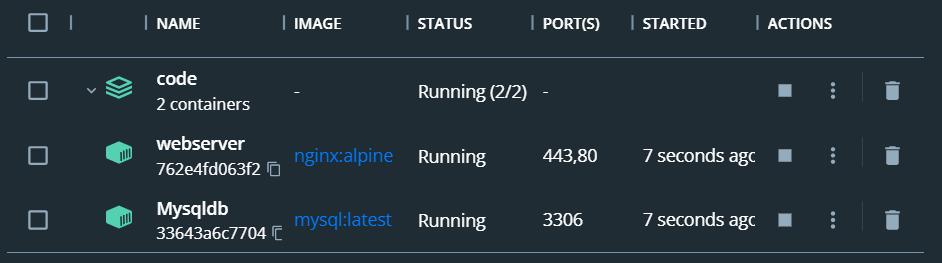
You can get this docker-compose file from [this GitHub repository.](https://github.com/Mikil03/docker-compose-sample.git)



4. Use docker-compose up to create these containers.

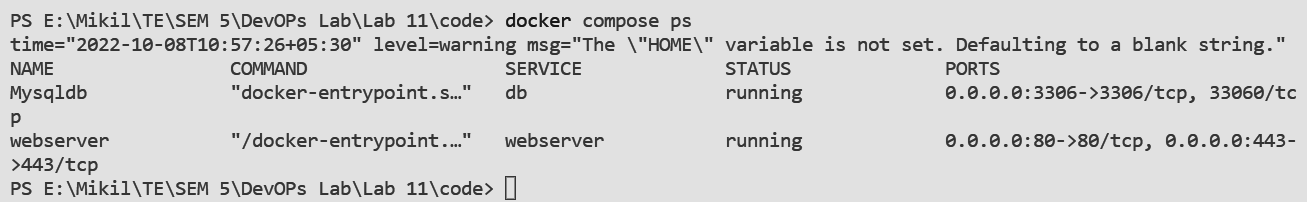


5. If you are using Docker Desktop, you can see these containers running in the menu -



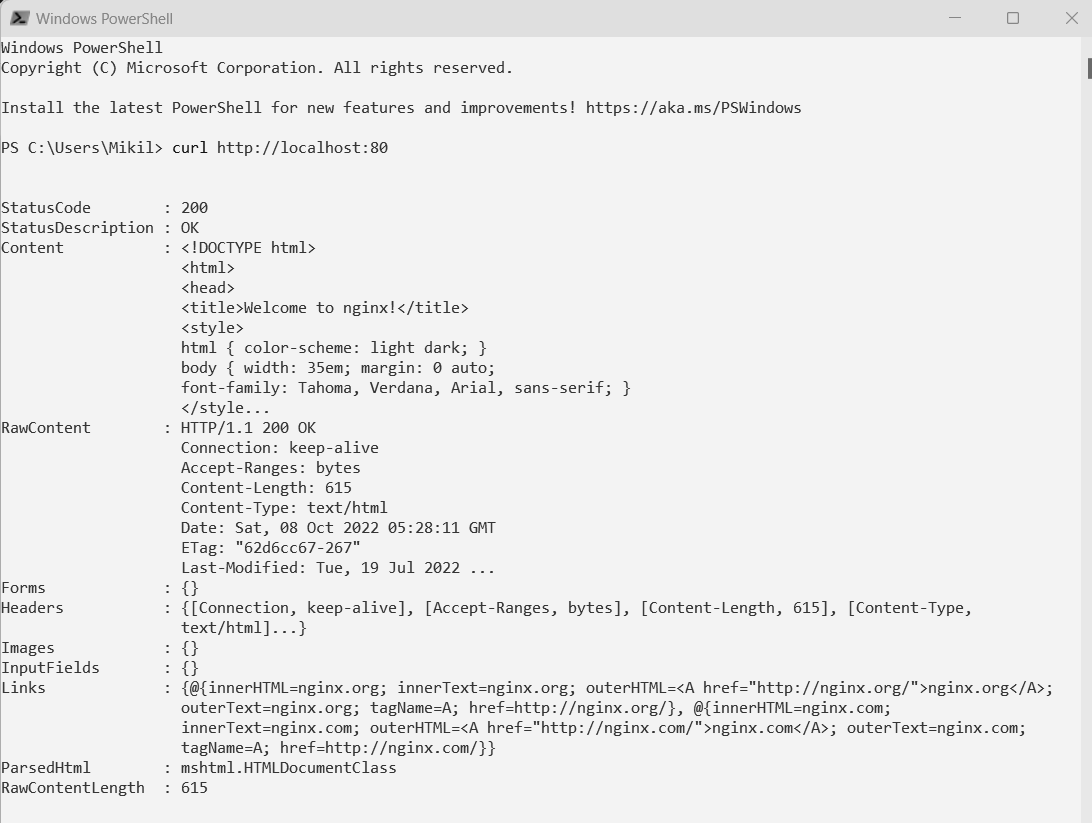
6. You can also verify this from the terminal. Simply open a new terminal in the same folder

and enter docker-compose ps

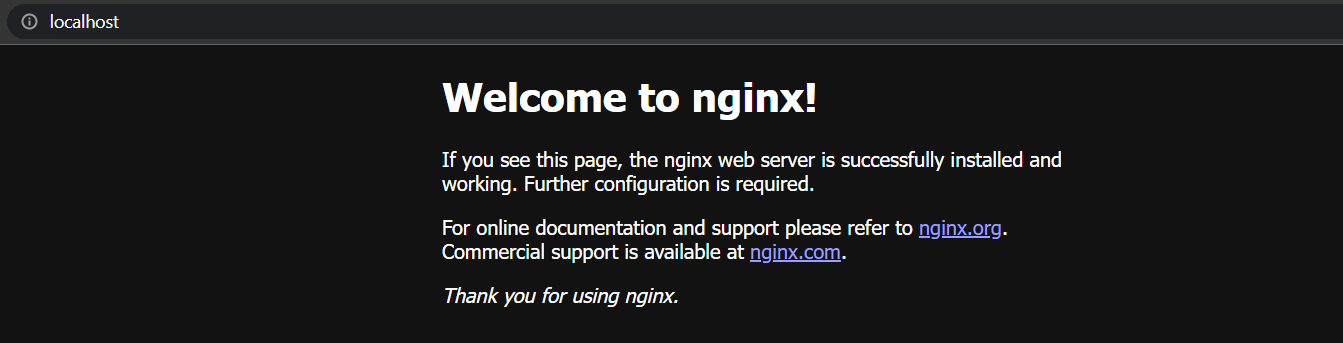


7. You can verify that Nginx is running from another terminal using curl.

curl <http://localhost:80>



You can also check the result on your browser, simply by going to localhost.



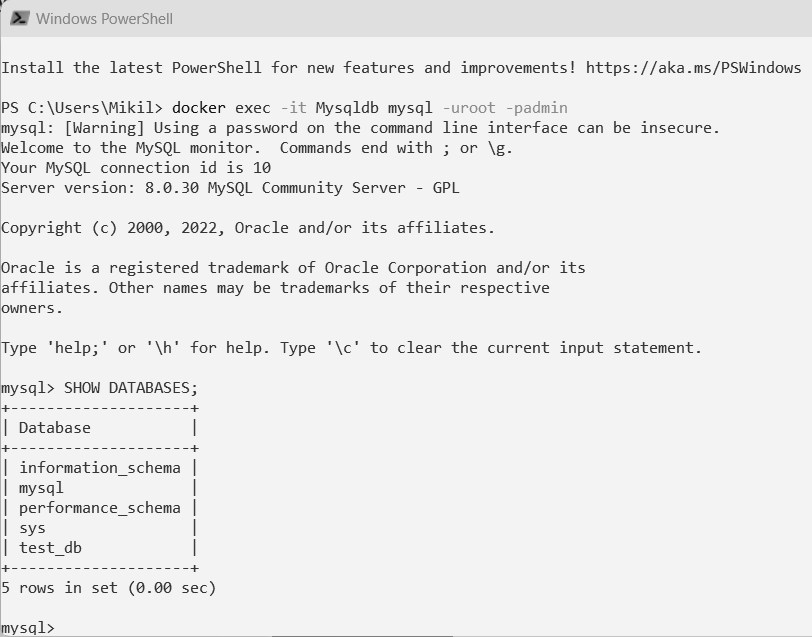
8. To verify the MySQL DB, you can use docker exec and login to the DB. Specify your

password at -p.

docker exec -it Mysqldb mysql -uroot -p<your\_root\_password>

You can try SQL queries like SHOW DATABASES; inside the shell to see the available

databases and further verify the server is running properly.



You can exit out of the mysql shell using quit.

**Conclusion**:

In this way, we learned about Docker Compose and created our first docker-compose file to simultaneously create 2 containers.