

Experiment 11 - Docker Compose

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	DevOps Lab
LO Mapped	<p>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</p> <p>LO5: To understand the concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker</p>

Aim: To understand 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

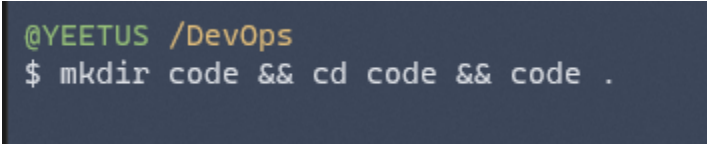
Docker Compose can also be used within a Dockerfile project and can be set up to build and run an image locally rather than pulling from the Docker Hub.

Demo:

In this demo, we will create two containers (nginx and MySQL) and set them up using one docker-compose file.

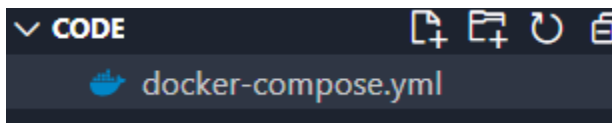
Steps:

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



```
@YEETUS /DevOps  
$ mkdir code && cd code && code .
```

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.

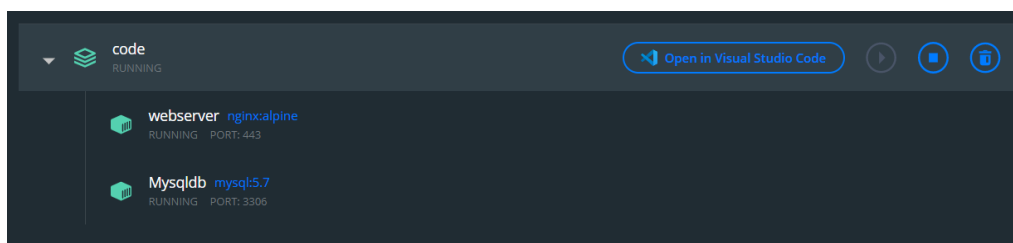
You can get this docker-compose file from [this GitHub repository](#).

```
docker-compose.yml X
1  version: "3.1"
2  services:
3    #Nginx Service
4    webserver:
5      image: nginx:alpine
6      container_name: webserver
7      restart: unless-stopped
8      ports:
9        - "80:80"
10       - "443:443"
11    #Mysql DB
12    db:
13      image: mysql
14      container_name: Mysqldb
15      restart: unless-stopped
16      volumes:
17        - $HOME/Desktop/MySQL-Snippets/school.sql:/school.sql
18      ports:
19        - "3306:3306"
20      environment:
21        MYSQL_ROOT_PASSWORD: 
22        MYSQL_DATABASE: test_db
23  volumes:
24    db_data:
25
```

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

```
@YEETUS /code
$ docker-compose up
```

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

```
@YEETUS /code
$ docker-compose ps
NAME                COMMAND                                SERVICE    STATUS    PORTS
Mysqldb             "docker-entrypoint.s..."            db         running   0.0.0.0:3306->3306/tcp
webserver           "/docker-entrypoint..."            webserver  running   0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
```

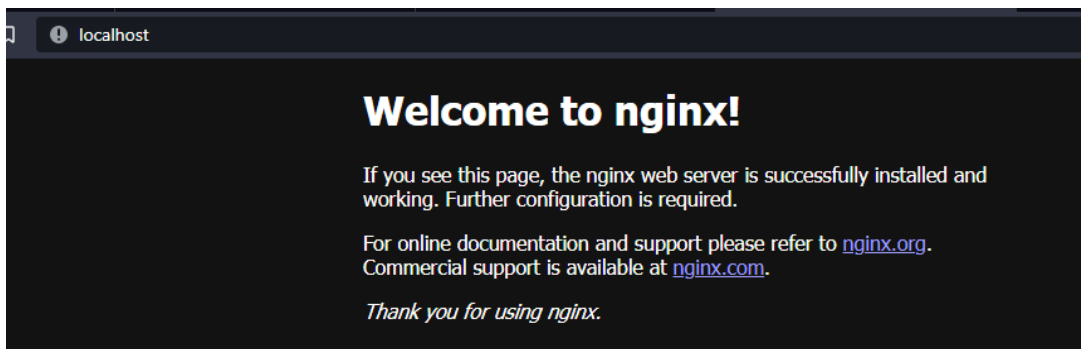
7. You can verify that Nginx is running from another terminal using curl.

`curl http://localhost:80`

```
Windows PowerShell
~ curl http://localhost:80

StatusCode      : 200
StatusDescription : OK
Content         : <!DOCTYPE html>
                  <html>
                  <head>
                  <title>Welcome to nginx!</title>
                  <style>
                    html { color-scheme: light dark; }
                    body { width: 35em; margin: 0 auto;
                      font-family: Tahoma, Verdana, Arial, sans-serif; }
                  </style>...
RawContent      : HTTP/1.1 200 OK
                  Connection: keep-alive
                  Accept-Ranges: bytes
                  Content-Length: 615
                  Content-Type: text/html
                  Date: Fri, 22 Oct 2021 14:32:19 GMT
                  ETag: "61378a62-267"
                  Last-Modified: Tue, 07 Sep 2021 ...
Forms           : {}
Headers         : {[Connection, keep-alive], [Accept-Ranges, bytes], [Content
                  text/html] }
```

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.

```
@YEETUS /code
$ docker exec -it Mysqldb mysql -uroot -proot
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.27 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| test_db |
+-----+
5 rows in set (0.01 sec)

mysql> 
```

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.