Experiment - 11

Aim: Docker Compose – multi container tool

Theory:

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.

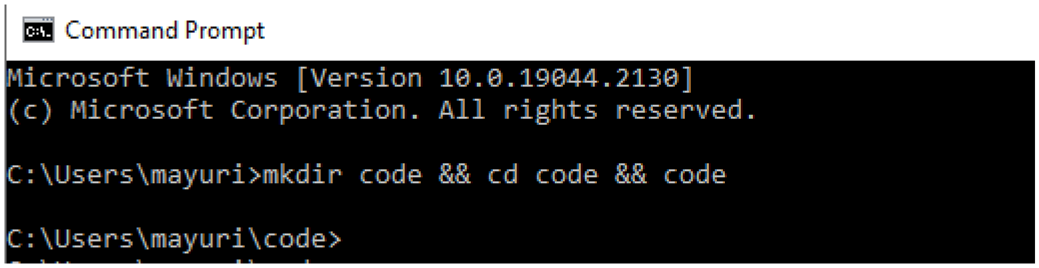
Managing all of these at deployment time is messy, so to clean it up, Docker provides Docker Compose, a configuration tool used for running multiple containers at once. You can define all of the configurations in one YAML file, and then start all the containers with one command. Rather than having all your services in one big container, Docker Compose allows you to split them up into individually manageable containers.

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.

Implementation:

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

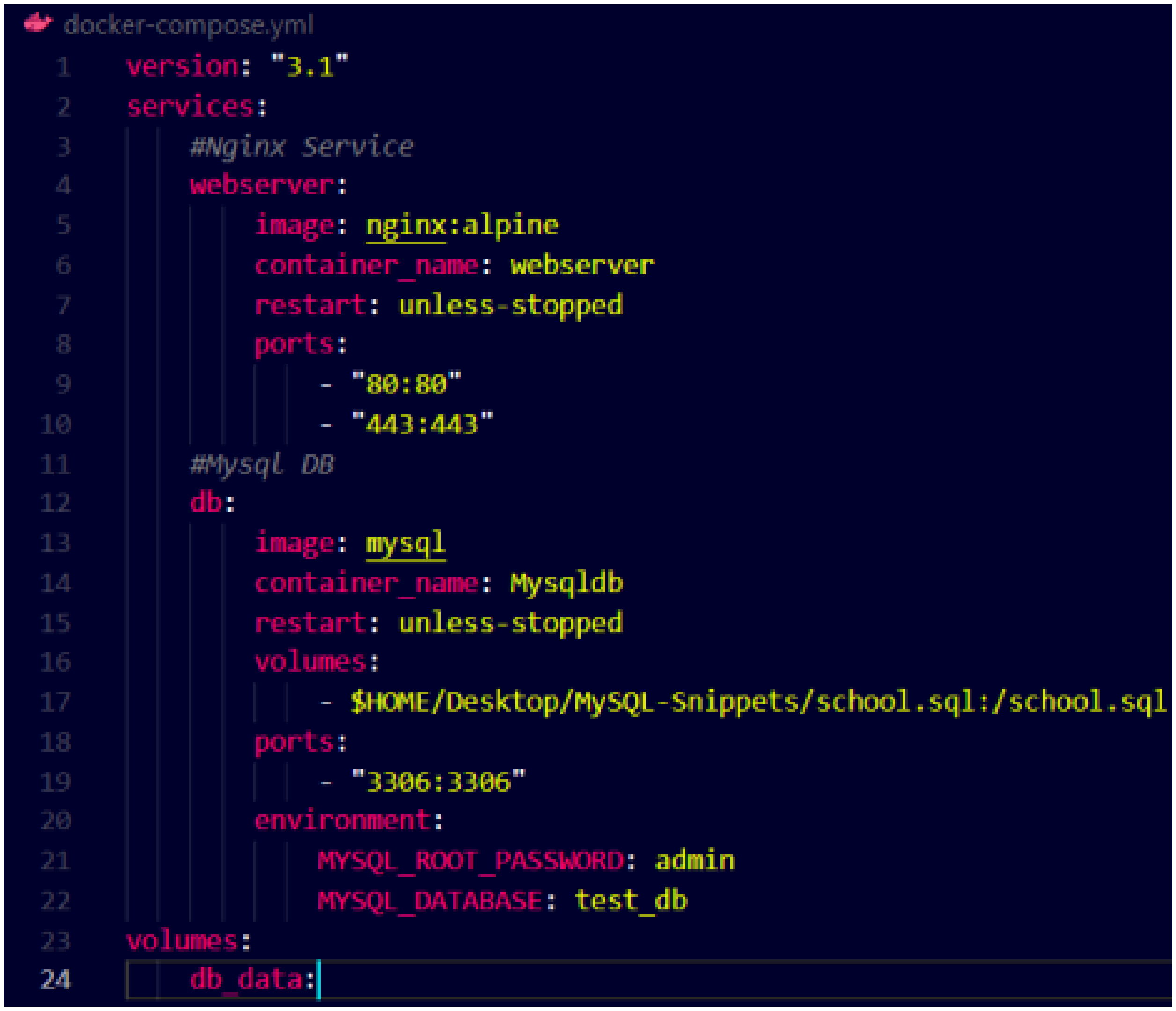


```
Command Prompt
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mayuri>mkdir code && cd code && code

C:\Users\mayuri\code>
```

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

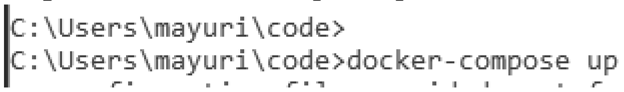
```
CODE
  docker-compose.yml
```

Step 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.

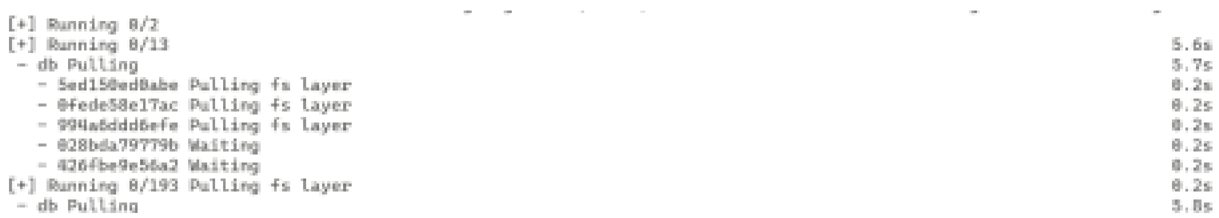


```
docker-compose.yml
version: "3.1"
services:
    #Nginx Service
    webserver:
        image: nginx:alpine
        container_name: webserver
        restart: unless-stopped
        ports:
            - "80:80"
            - "443:443"
    #Mysql DB
    db:
        image: mysql
        container_name: Mysqldb
        restart: unless-stopped
        volumes:
            - $HOME/Desktop/MySQL-Snippets/school.sql:/school.sql
        ports:
            - "3306:3306"
        environment:
            MYSQL_ROOT_PASSWORD: admin
            MYSQL_DATABASE: test_db
volumes:
    db_data:
```

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

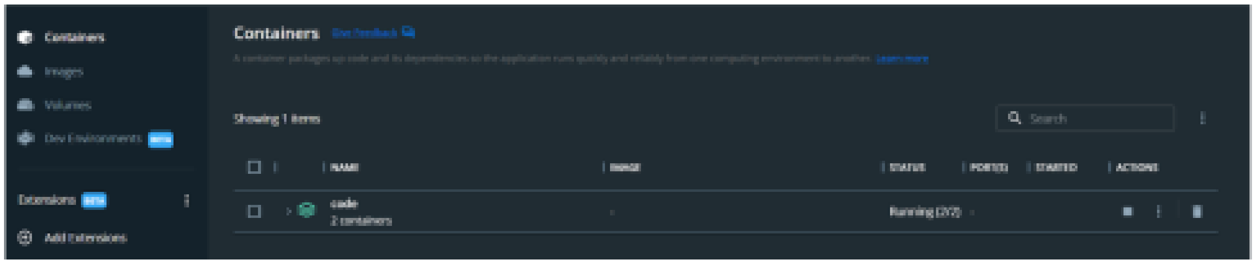


```
C:\Users\mayuri\code>
C:\Users\mayuri\code>docker-compose up
```



```
[+] Running 0/2
[+] Running 0/13                                                              5.6s
 - db Pulling                                                                 5.7s
   - 5ed150ed0abe Pulling fs layer                                            0.2s
   - 0fede58e17ac Pulling fs layer                                            0.2s
   - 994a6ddd6efe Pulling fs layer                                            0.2s
   - 028bda79779b Waiting                                                     0.2s
   - 426fbe9e56a2 Waiting                                                     0.2s
[+] Running 0/193 Pulling fs layer                                            0.2s
 - db Pulling                                                                 5.8s
```

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



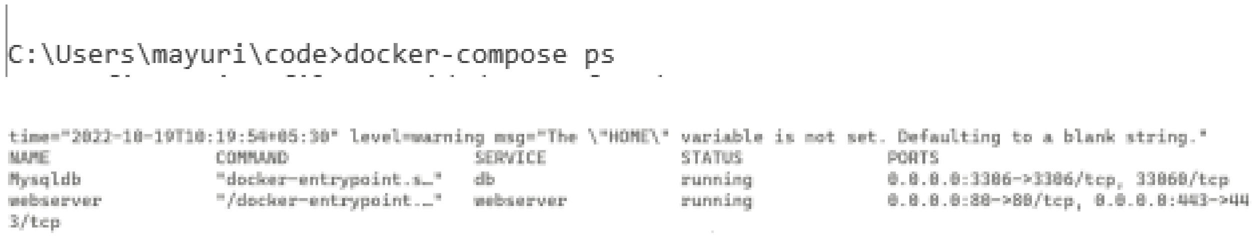
Containers

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Showing 1 items

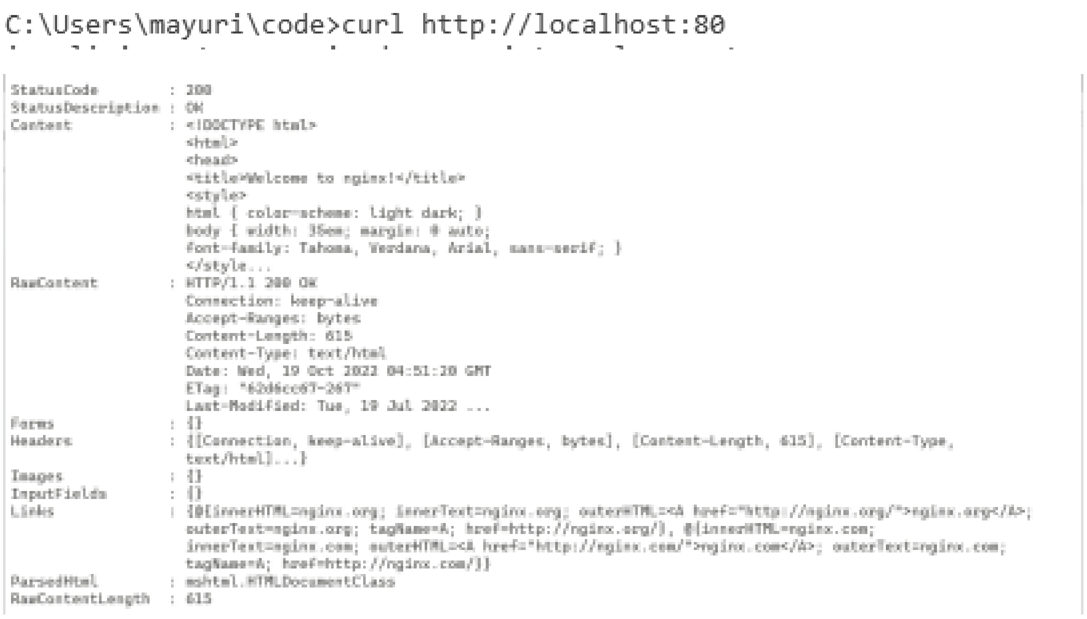
NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
code 2 containers	-	Running (2/2)			

Step 6: You can also verify this from the terminal. Simply open a new terminal in the same folder and enter docker-compose ps



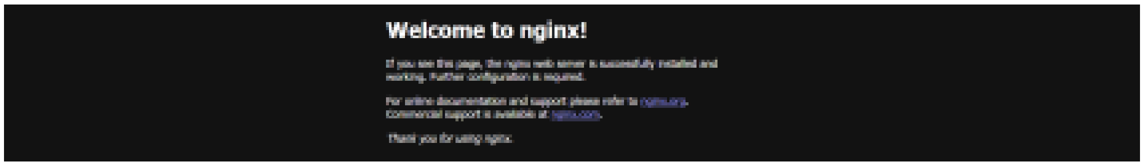
```
C:\Users\mayuri\code>docker-compose ps

time="2022-10-19T10:19:54+05:30" level=warning msg="The \"HOME\" variable is not set. Defaulting to a blank string."
NAME                COMMAND                  SERVICE             STATUS              PORTS
Mysqldb             "docker-entrypoint.s…"   db                  running             0.0.0.0:3306->3306/tcp, 33060/tcp
webserver           "/docker-entrypoint.…"   webserver           running             0.0.0.0:80->80/tcp, 0.0.0.0:443->44
3/tcp
```

Step 7: You can verify that Nginx is running from another terminal using curl. curl http://localhost:80



```
C:\Users\mayuri\code>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: Wed, 19 Oct 2022 04:51:20 GMT
                    ETag: "62d6cc67-267"
                    Last-Modified: Tue, 19 Jul 2022 ...
Forms             : {}
Headers           : {[Connection, keep-alive], [Accept-Ranges, bytes], [Content-Length, 615], [Content-Type,
                    text/html]...}
Images            : {}
InputFields       : {}
Links             : {@{innerHTML=nginx.org; innerText=nginx.org; outerHTML=<A href="http://nginx.org/">nginx.org</A>;
                    outerText=nginx.org; tagName=A; href=http://nginx.org/}, @{innerHTML=nginx.com;
                    innerText=nginx.com; outerHTML=<A href="http://nginx.com/">nginx.com</A>; outerText=nginx.com;
                    tagName=A; href=http://nginx.com/}}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 615
```

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



Welcome to nginx!

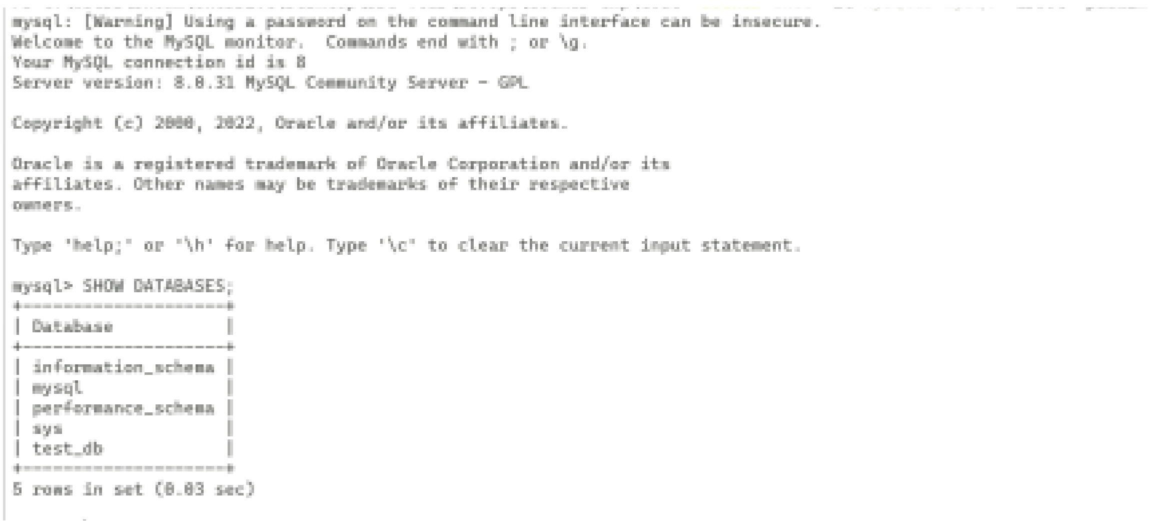
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Step 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.



```
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 8
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, 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 (0.03 sec)
```

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.