3 Tier Application Monolithic Application Deployment Using Docker Container

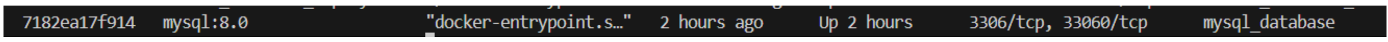
14 April 2025 17:36

1. Working on Database Layer.

Here

1. We will pull the mysql:8.0 image from DockerHub.
   docker pull mysql:8.0
2. Run the mysql image on a container where it is running on default port 3306 in bridge network to connect with out backend service.
   docker run --name mysql_database -e MYSQL_ROOT_PASSWORD=root -d mysql:8.0
   Here default mysql server name will be root and password we mentioned.
   Now we exec into the MySQL container
   docker exec -it mysql_database mysql -u root -p
   and will create the database and Tables associated to our backend mentioned in the Table.md in backend application.
   and mention the same database IP, database name, database username and password carefully.

   docker ps

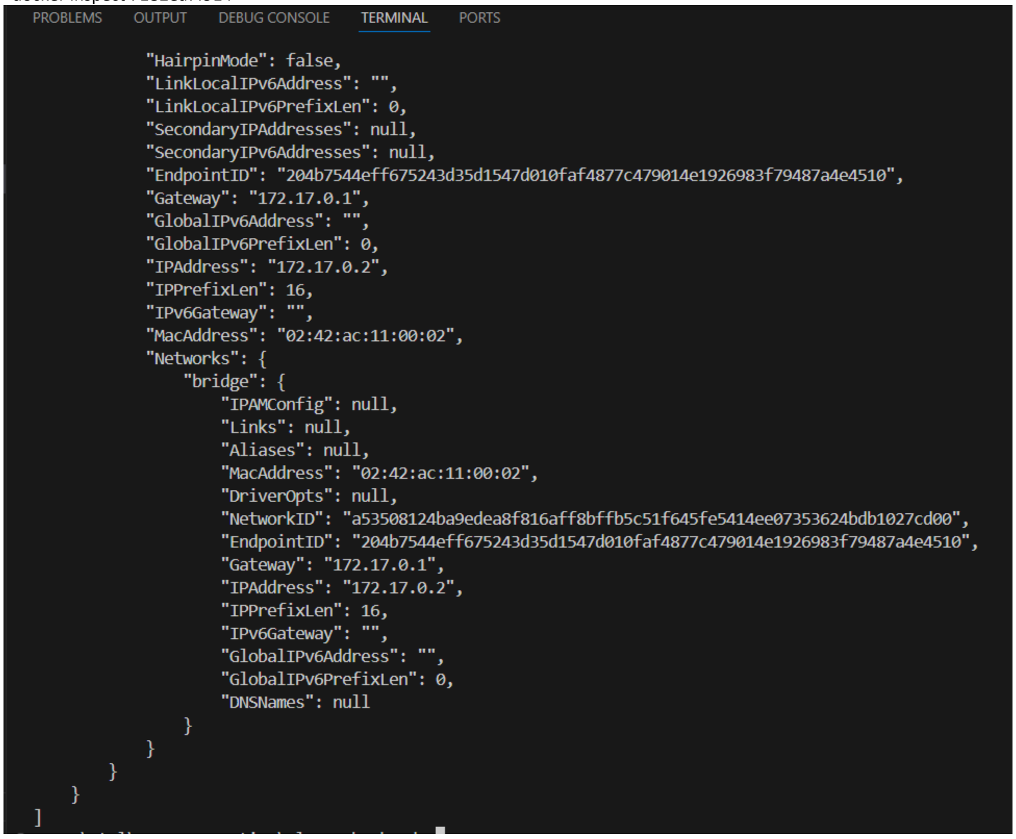


```
7182ea17f914   mysql:8.0   "docker-entrypoint.s…"   2 hours ago   Up 2 hours   3306/tcp, 33060/tcp   mysql_database
```

2. Working on Backend Layer.

Here we go inside the Folder were our appsetting.json file is stored for dotnet based application.
It can change for different configuration file form different backend application language used.
(In Java there is pom.xml File)

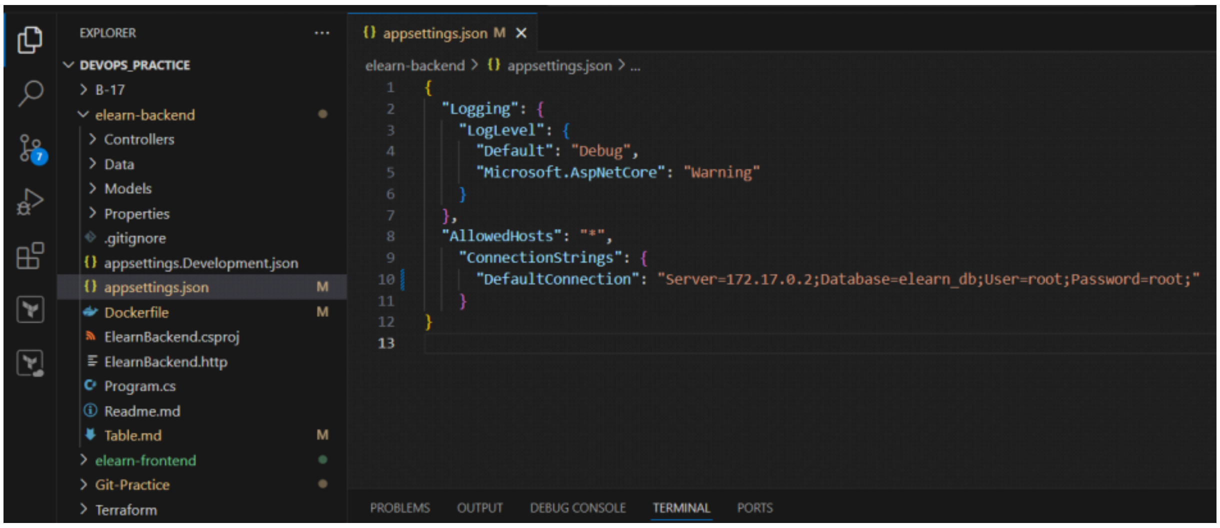
docker inspect 7182ea7f914



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

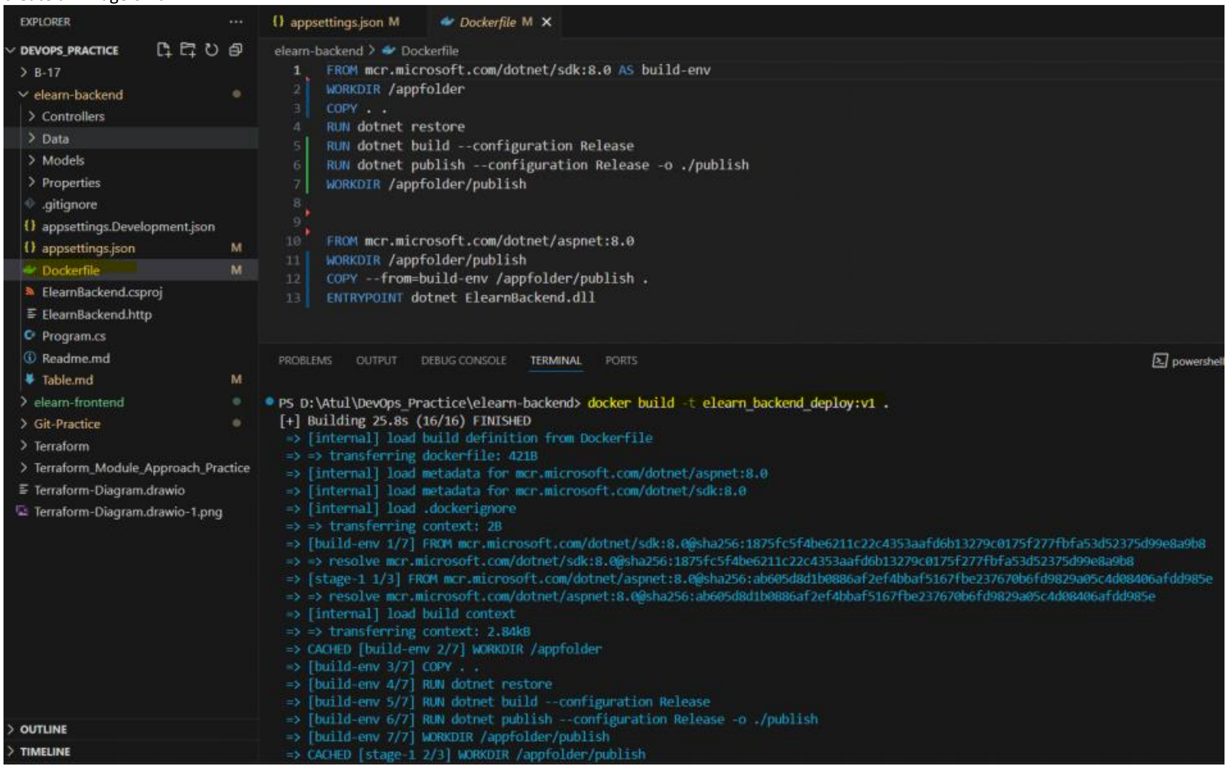
```
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "204b7544eff675243d35d1547d010faf4877c479014e1926983f79487a4e4510",
            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "MacAddress": "02:42:ac:11:00:02",
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "MacAddress": "02:42:ac:11:00:02",
                    "DriverOpts": null,
                    "NetworkID": "a53508124ba9edea8f816aff8bffb5c51f645fe5414ee07353624bdb1027cd00",
                    "EndpointID": "204b7544eff675243d35d1547d010faf4877c479014e1926983f79487a4e4510",
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "DNSNames": null
                }
            }
        }
    }
]
```

Mention the IP in the database where developer has mentioned in the README.md file.



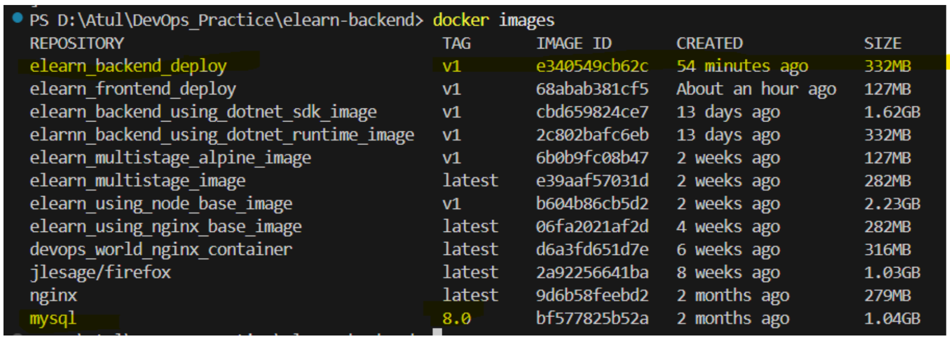
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
    "ConnectionStrings": {
      "DefaultConnection": "Server=172.17.0.2;Database=elearn_db;User=root;Password=root;"
    }
}
```

We will now create Docker File and will provide all the steps to run our application in runtime container these steps would be there in the README.md and we need write Dockerfile for it and create an image of it.



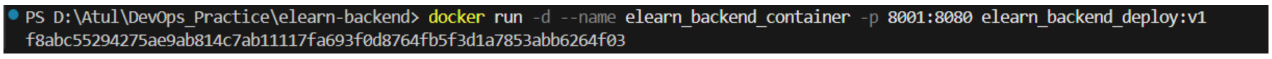
```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build-env
WORKDIR /appfolder
COPY . .
RUN dotnet restore
RUN dotnet build --configuration Release
RUN dotnet publish --configuration Release -o ./publish
WORKDIR /appfolder/publish


FROM mcr.microsoft.com/dotnet/aspnet:8.0
WORKDIR /appfolder/publish
COPY --from=build-env /appfolder/publish .
ENTRYPOINT dotnet ElearnBackend.dll
```

```
PS D:\Atul\DevOps_Practice\elearn-backend> docker build -t elearn_backend_deploy:v1 .
[+] Building 25.8s (16/16) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 421B
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:8.0
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:8.0
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [build-env 1/7] FROM mcr.microsoft.com/dotnet/sdk:8.0@sha256:1875fc5f4be6211c22c4353aafd6b13279c0175f277fbfa53d52375d99e8a9b8
 => => resolve mcr.microsoft.com/dotnet/sdk:8.0@sha256:1875fc5f4be6211c22c4353aafd6b13279c0175f277fbfa53d52375d99e8a9b8
 => [stage-1 1/3] FROM mcr.microsoft.com/dotnet/aspnet:8.0@sha256:ab605d8d1b0886af2ef4bbaf5167fbe237670b6fd9829a05c4d08406afdd985e
 => => resolve mcr.microsoft.com/dotnet/aspnet:8.0@sha256:ab605d8d1b0886af2ef4bbaf5167fbe237670b6fd9829a05c4d08406afdd985e
 => [internal] load build context
 => => transferring context: 2.84kB
 => CACHED [build-env 2/7] WORKDIR /appfolder
 => [build-env 3/7] COPY . .
 => [build-env 4/7] RUN dotnet restore
 => [build-env 5/7] RUN dotnet build --configuration Release
 => [build-env 6/7] RUN dotnet publish --configuration Release -o ./publish
 => [build-env 7/7] WORKDIR /appfolder/publish
 => CACHED [stage-1 2/3] WORKDIR /appfolder/publish
```



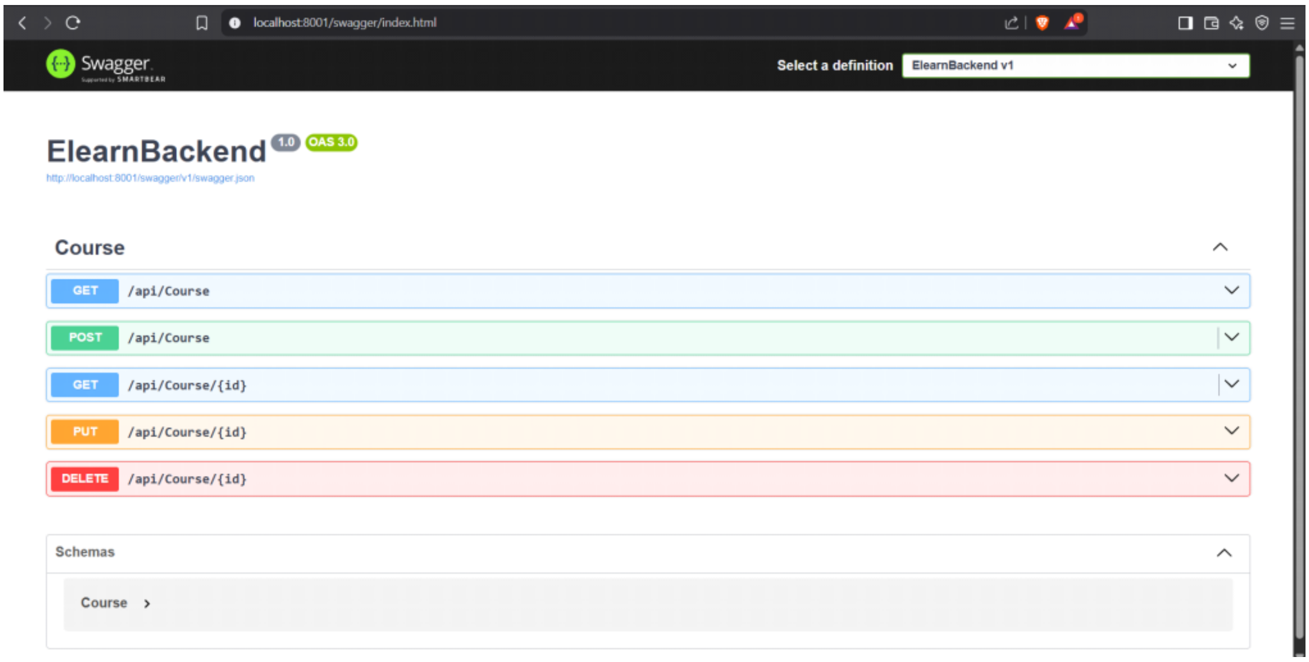
```
PS D:\Atul\DevOps_Practice\elearn-backend> docker images
REPOSITORY                           TAG       IMAGE ID       CREATED              SIZE
elearn_backend_deploy                v1        e340549cb62c   54 minutes ago       332MB
elearn_frontend_deploy               v1        68abab381cf5   About an hour ago    127MB
elearn_backend_using_dotnet_sdk_image    v1    cbd659824ce7   13 days ago          1.62GB
elarnn_backend_using_dotnet_runtime_image v1   2c802bafc6eb   13 days ago          332MB
elearn_multistage_alpine_image       v1        6b0b9fc08b47   2 weeks ago          127MB
elearn_multistage_image              latest    e39aaf57031d   2 weeks ago          282MB
elearn_using_node_base_image         v1        b604b86cb5d2   2 weeks ago          2.23GB
elearn_using_nginx_base_image        latest    06fa2021af2d   4 weeks ago          282MB
devops_world_nginx_container         latest    d6a3fd651d7e   6 weeks ago          316MB
jlesage/firefox                      latest    2a92256641ba   8 weeks ago          1.03GB
nginx                                latest    9d6b58feebd2   2 months ago         279MB
mysql                                8.0       bf577825b52a   2 months ago         1.04GB
```

Now It is ready to run our backend image on a container
docker run -d --name elearn_backend_container -p 8001:8080 elearn_backend_deploy:v1



```
PS D:\Atul\DevOps_Practice\elearn-backend> docker run -d --name elearn_backend_container -p 8001:8080 elearn_backend_deploy:v1
f8abc55294275ae9ab814c7ab11117fa693f0d8764fb5f3d1a7853abb6264f03
```

Now our Backend Application is up and running.
It will not run on the local port mentioned and will run on Swagger(API documentation Tool.)



ElearnBackend 1.0 OAS 3.0

http://localhost:8001/swagger/v1/swagger.json

Course

```
GET     /api/Course
POST    /api/Course
GET     /api/Course/{id}
PUT     /api/Course/{id}
DELETE  /api/Course/{id}
```

Schemas

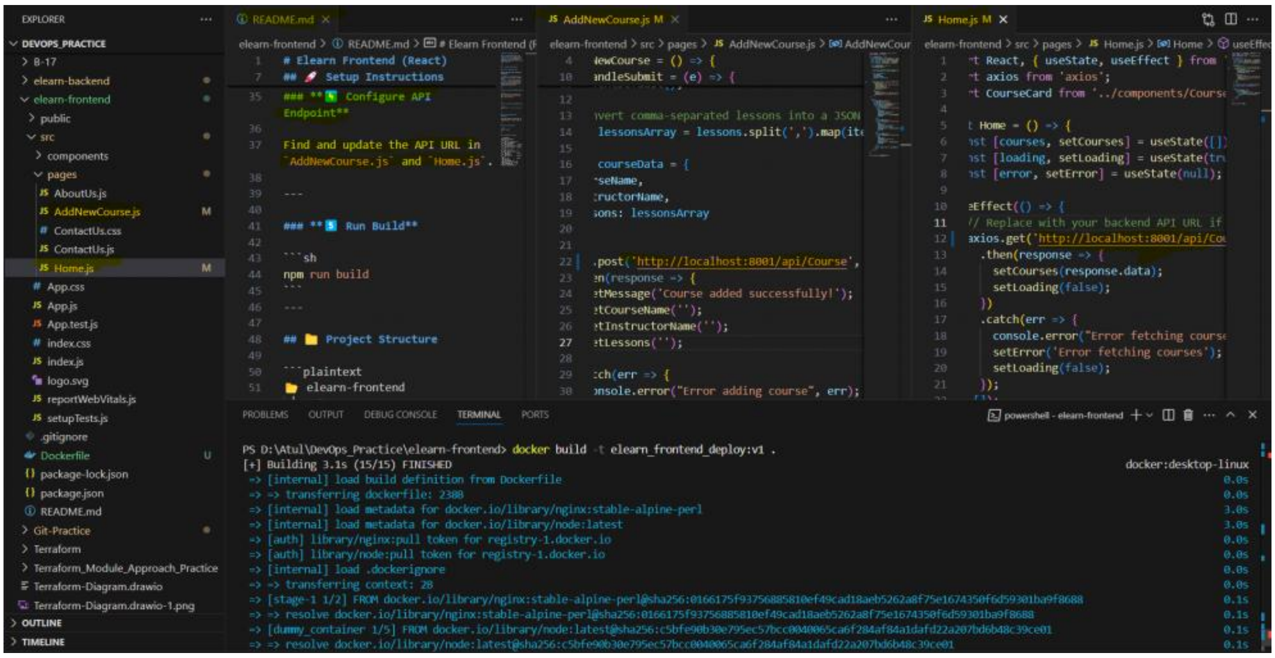
Course >

To troubleshot any failure during backend deployment on a container we uses
  docker logs <container_name_or_id>
Use help to see more options.

3. Working on Frontend Layer.

Now make the Changes in the Frontend application mentioned in README.md file for binding the connection between Frontend and backend application.
Here we are giving Local host IP of the Backend application as Out browser run on Local it will create the list of call on the local machine to port where our backend application is running.

We now go into the frontend application folder where out package.json is stored and create the Dockerfile for the build and Create an image for it.



elearn-frontend > README.md > # Elearn Frontend (F

~~~
# Elearn Frontend (React)
## 🚀 Setup Instructions
### **🔄 Configure API
Endpoint**

Find and update the API URL in
`AddNewCourse.js` and `Home.js`.

---

### **5 Run Build**

```sh
npm run build
~~~

📁 Project Structure

```plaintext
📁 elearn-frontend
```

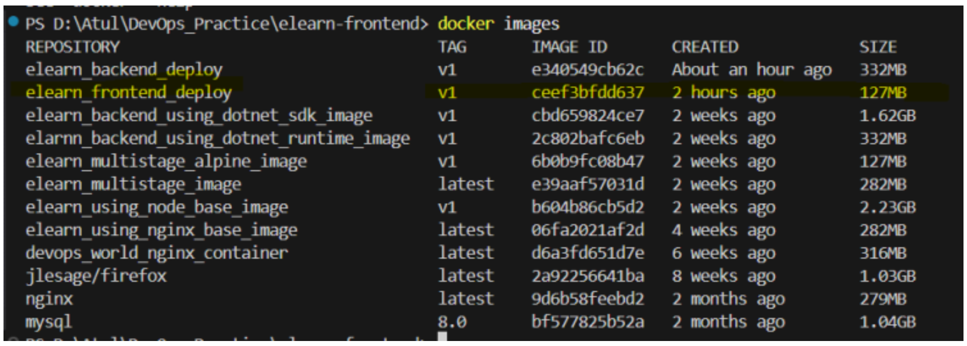
elearn-frontend > src > pages > AddNewCourse.js > AddNewCour

```
lewCourse = () => {
andleSubmit = (e) => {

vert comma-separated lessons into a JSON
 lessonsArray = lessons.split(',').map(it

 courseData = {
'seName,
:ructorName,
sons: lessonsArray


.post('http://localhost:8001/api/Course',
en(response => {
etMessage('Course added successfully!');
etCourseName('');
etInstructorName('');
etLessons('');

:ch(err => {
onsole.error("Error adding course", err);
```

elearn-frontend > src > pages > Home.js > Home > useEffec

```
rt React, { useState, useEffect } from
rt axios from 'axios';
rt CourseCard from '../components/Course

t Home = () => {
nst [courses, setCourses] = useState([]
nst [loading, setLoading] = useState(tru
nst [error, setError] = useState(null);

eEffect(() => {
// Replace with your backend API URL if
axios.get('http://localhost:8001/api/Co
  .then(response => {
    setCourses(response.data);
    setLoading(false);
  })
  .catch(err => {
    console.error("Error fetching course
    setError('Error fetching courses');
    setLoading(false);
  });
```

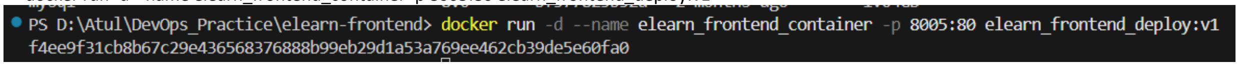
```
PS D:\Atul\DevOps_Practice\elearn-frontend> docker build -t elearn_frontend_deploy:v1 .
[+] Building 3.1s (15/15) FINISHED                                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                        0.0s
 => => transferring dockerfile: 238B                                                                        0.0s
 => [internal] load metadata for docker.io/library/nginx:stable-alpine-perl                                 3.0s
 => [internal] load metadata for docker.io/library/node:latest                                              3.0s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                0.0s
 => [auth] library/node:pull token for registry-1.docker.io                                                 0.0s
 => [internal] load .dockerignore                                                                           0.0s
 => => transferring context: 2B                                                                             0.0s
 => [stage-1 1/2] FROM docker.io/library/nginx:stable-alpine-perl@sha256:0166175f93756885810ef49cad18aeb5262a8f75e1674350f6d59301ba9f8688  0.1s
 => => resolve docker.io/library/nginx:stable-alpine-perl@sha256:0166175f93756885810ef49cad18aeb5262a8f75e1674350f6d59301ba9f8688  0.1s
 => [dummy_container 1/5] FROM docker.io/library/node:latest@sha256:c5bfe90b30e795ec57bcc0040065ca6f284af84a1dafd22a207bd6b48c39ce01  0.1s
 => => resolve docker.io/library/node:latest@sha256:c5bfe90b30e795ec57bcc0040065ca6f284af84a1dafd22a207bd6b48c39ce01  0.1s
```



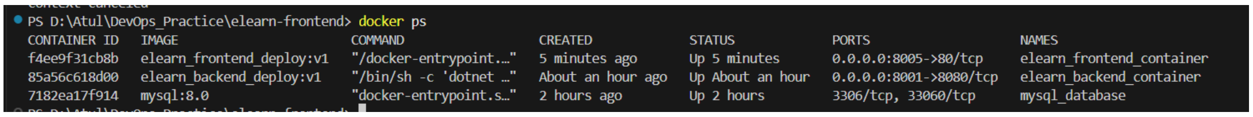
```
PS D:\Atul\DevOps_Practice\elearn-frontend> docker images
REPOSITORY                               TAG      IMAGE ID       CREATED              SIZE
elearn_backend_deploy                    v1       e340549cb62c   About an hour ago    332MB
elearn_frontend_deploy                   v1       ceef3bfdd637   2 hours ago          127MB
elearn_backend_using_dotnet_sdk_image    v1       cbd659824ce7   2 weeks ago          1.62GB
elarnn_backend_using_dotnet_runtime_image v1      2c802bafc6eb   2 weeks ago          332MB
elearn_multistage_alpine_image           v1       6b0b9fc08b47   2 weeks ago          127MB
elearn_multistage_image                  latest   e39aaf57031d   2 weeks ago          282MB
elearn_using_node_base_image             v1       b604b86cb5d2   2 weeks ago          2.23GB
elearn_using_nginx_base_image            latest   06fa2021af2d   4 weeks ago          282MB
devops_world_nginx_container             latest   d6a3fd651d7e   6 weeks ago          316MB
jlesage/firefox                          latest   2a92256641ba   8 weeks ago          1.03GB
nginx                                    latest   9d6b58feebd2   2 months ago         279MB
mysql                                    8.0      bf577825b52a   2 months ago         1.04GB
```

We will run the created frontend image on the container:

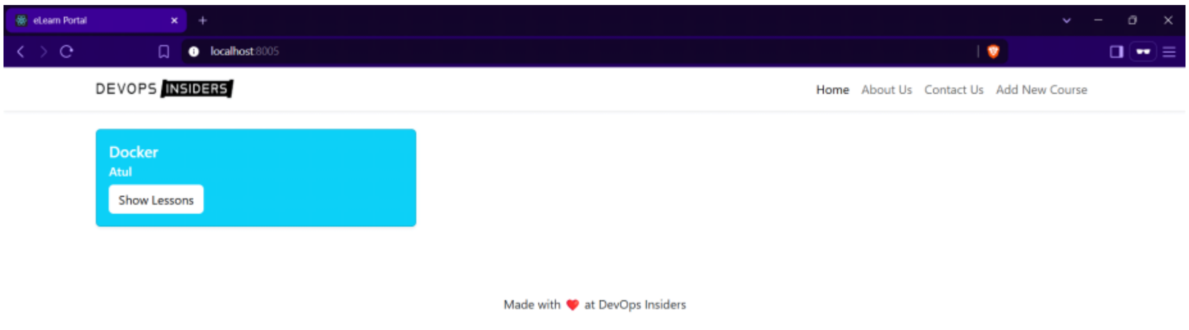
```
docker run -d --name elearn_frontend_container -p 8005:80 elearn_frontend_deploy:v1
```

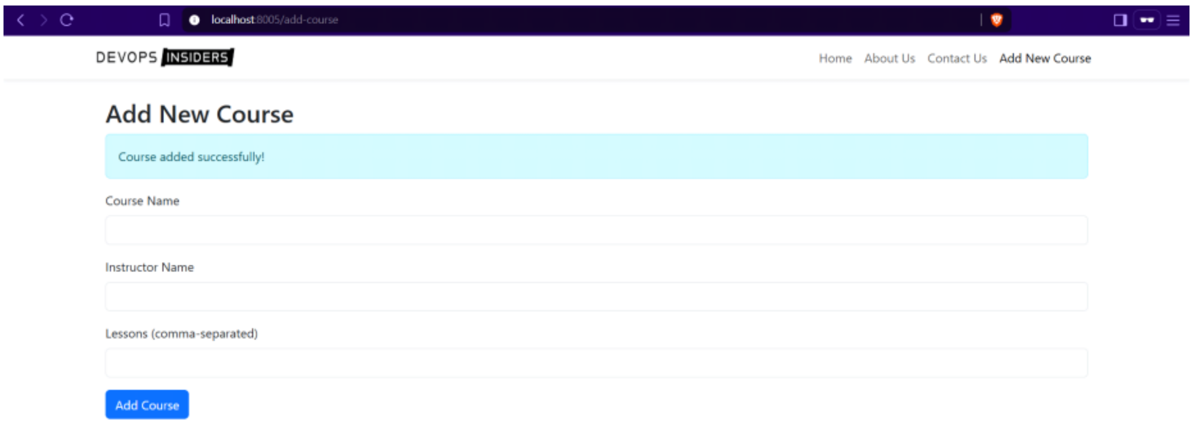


```
PS D:\Atul\DevOps_Practice\elearn-frontend> docker run -d --name elearn_frontend_container -p 8005:80 elearn_frontend_deploy:v1
f4ee9f31cb8b67c29e436568376888b99eb29d1a53a769ee462cb39de5e60fa0
```



```
PS D:\Atul\DevOps_Practice\elearn-frontend> docker ps
CONTAINER ID   IMAGE                       COMMAND                  CREATED             STATUS              PORTS                    NAMES
f4ee9f31cb8b   elearn_frontend_deploy:v1   "/docker-entrypoint.…"   5 minutes ago       Up 5 minutes        0.0.0.0:8005->80/tcp     elearn_frontend_container
85a56c618d00   elearn_backend_deploy:v1    "/bin/sh -c 'dotnet …"   About an hour ago   Up About an hour    0.0.0.0:8001->8080/tcp   elearn_backend_container
7182ea17f914   mysql:8.0                   "docker-entrypoint.s…"   2 hours ago         Up 2 hours          3306/tcp, 33060/tcp      mysql_database
```





Add New Course

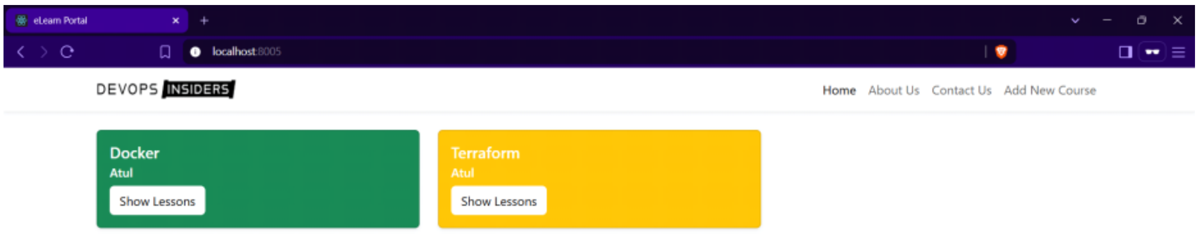
Course added successfully!

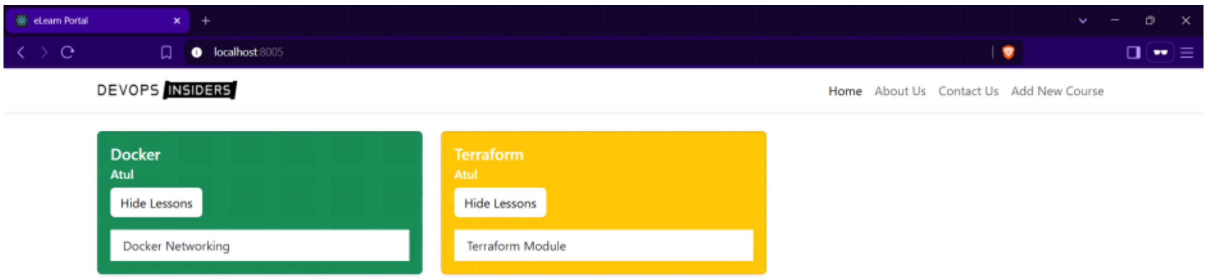
Course Name

Instructor Name

Lessons (comma-separated)

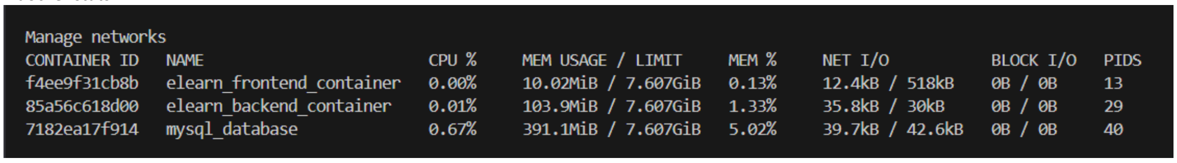
Add Course





To troubleshot any failure in Frontend application on Container
   Use browser network logs and inspect there.

You can see the Utilization using
   docker stats



```
Manage networks
CONTAINER ID   NAME                        CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O   PIDS
f4ee9f31cb8b   elearn_frontend_container   0.00%     10.02MiB / 7.607GiB   0.13%     12.4kB / 518kB    0B / 0B     13
85a56c618d00   elearn_backend_container    0.01%     103.9MiB / 7.607GiB   1.33%     35.8kB / 30kB     0B / 0B     29
7182ea17f914   mysql_database              0.67%     391.1MiB / 7.607GiB   5.02%     39.7kB / 42.6kB   0B / 0B     40
```