



Sri Lanka Institute of Information Technology

Year 4 | Semester 2 | 2021

ASSIGNMENT 01

DOCKER

(Individual)

Cloud Computing (IT4090)

Submitted By:

Student ID	Name
IT18001730	T.S. Chethana Fernando

Table of Contents

01 - INTRODUCTION	1
02 - SYSTEM OVERVIEW DIAGRAM.....	2
2.1 Front-end Tier	3
2.2 Application Logic / API Tier	9
2.3 Database Tier	9
03 - SCREENSHOTS OF DOCKER ENVIRONMENT	10
04 - APPENDIX	13

GitHub Link :-

<https://github.com/96Chethana/Docker-Assignment-01>

Video for system :-

<https://drive.google.com/file/d/1rPN4cdiFC-0KYHIvxw90eUkOUap1f3Gy/view?usp=sharing>

01 - Introduction

Docker provides lightweight containers to run services in isolation from infrastructure for deliver the software quickly. In here, I will show that how to Dockerize MERN stack Application with React + Node.js + Express + MongoDB using Docker Compose and Nginx.

As well as this Mini Courseweb application have three tiers, a front-end tier, an application logic / API tier, a database tier. Therefore, application is to containerize a system that requires more than one Docker container:

- ✚ React for UI :- Stateless container
- ✚ Node.js Express for API :- Stateless container
- ✚ MongoDB for database :- Stateful container

Docker Compose helps to setup the system more easily and efficiently than with only Docker. When developing this application, considered theses following steps.

- ✚ Setup Nodejs App working with MongoDB database.
- ✚ Create Dockerfile for Nodejs App.
- ✚ Setup React App.
- ✚ Create Dockerfile for React App.
- ✚ Create Docker Compose configurations in YAML file.
- ✚ Set Environment variables for Docker Compose
- ✚ Run the system Mini Courseweb application.

02 - System Overview Diagram

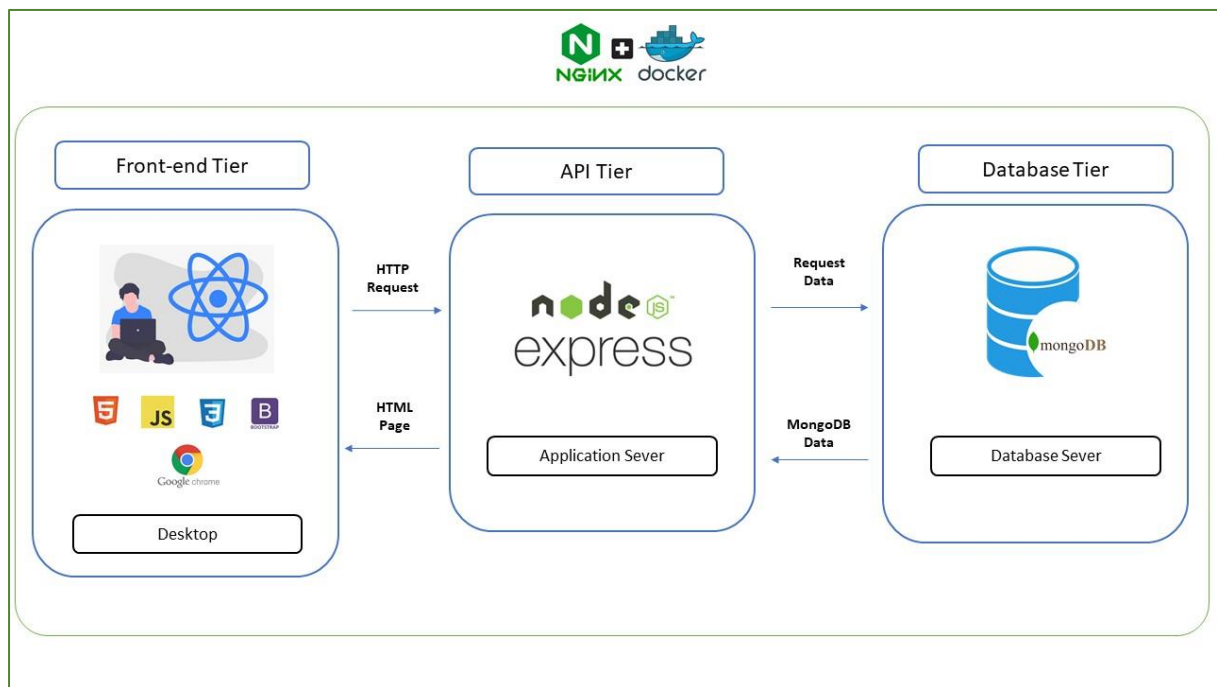


Fig. 1. System Diagram

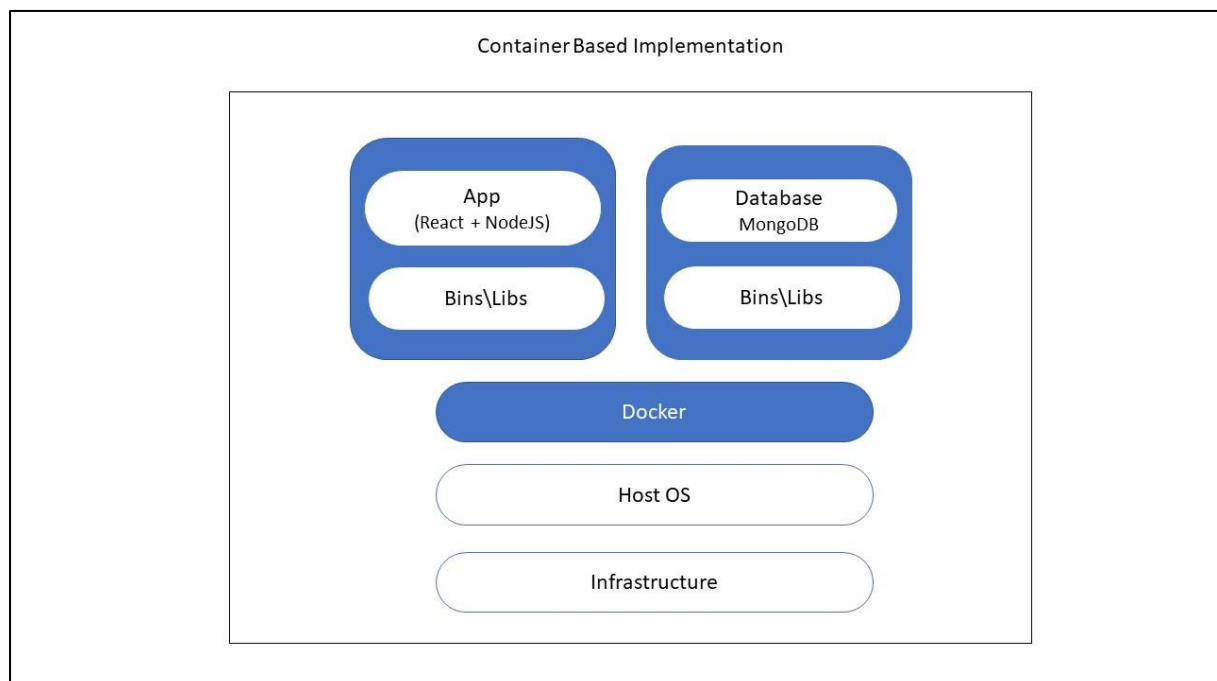
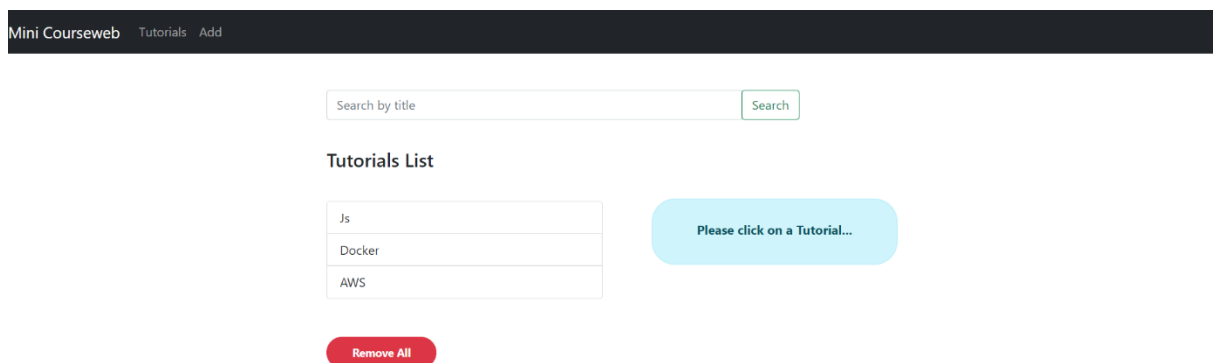


Fig. 2. Container based implementation Architecture

2.1 Front-end Tier

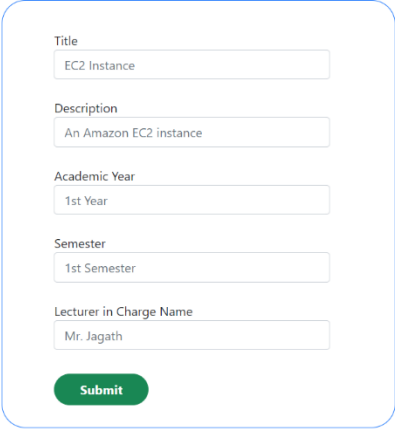
- ✚ UI developer's main concern is ensuring the correct data is delivered to their UI components when they're required, and that actions released by user activity inside the UI are interpreted and acted upon.
- ✚ In here the Mini course web application allowed to lecturer to add tutorial in the Mini Courseweb. As well as they can view the list of the Tutorial they add. Also, they can edit the tutorial details, delete the details form the system and search the tutorial using tutorial title. There is option for lecturers to publish or unpublish the tutorial they added to the Mini Courseweb. Fig 1. Shows the list of the tutorial in the Mini Courseweb.



The screenshot shows the 'Mini Courseweb' interface with a dark header bar containing the text 'Mini Courseweb', 'Tutorials', and 'Add'. Below the header, there is a search bar with the placeholder text 'Search by title' and a 'Search' button. The main content area is titled 'Tutorials List' and contains a table with three rows: 'Js', 'Docker', and 'AWS'. To the right of the table is a light blue button with the text 'Please click on a Tutorial...'. Below the table is a red button with the text 'Remove All'.

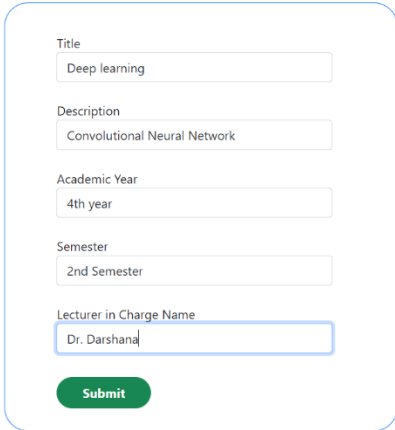
Fig. 3. Tutorial List

- ✚ The tutorial Submission form show below Fig 2. Entering the details of tutorial details, then the user can submit the details by click on the submit button.

A form for submitting a tutorial. It contains six text input fields: 'Title' with 'EC2 Instance', 'Description' with 'An Amazon EC2 instance', 'Academic Year' with '1st Year', 'Semester' with '1st Semester', and 'Lecturer in Charge Name' with 'Mr. Jagath'. A green 'Submit' button is at the bottom.

Title	EC2 Instance
Description	An Amazon EC2 instance
Academic Year	1st Year
Semester	1st Semester
Lecturer in Charge Name	Mr. Jagath
<input type="submit" value="Submit"/>	

Fig. 4. Tutorial submission form

A form for submitting tutorial details. It contains five text input fields: 'Title' with 'Deep learning', 'Description' with 'Convolutional Neural Network', 'Academic Year' with '4th year', 'Semester' with '2nd Semester', and 'Lecturer in Charge Name' with 'Dr. Darshana'. A green 'Submit' button is at the bottom.

Title	Deep learning
Description	Convolutional Neural Network
Academic Year	4th year
Semester	2nd Semester
Lecturer in Charge Name	Dr. Darshana
<input type="submit" value="Submit"/>	

Fig. 5. Submit details

After submitting the tutorial details, user can view message as “You Submitted Successfully”. If user wants to add another tutorial by click on the Add button, user can add new tutorial to the system.

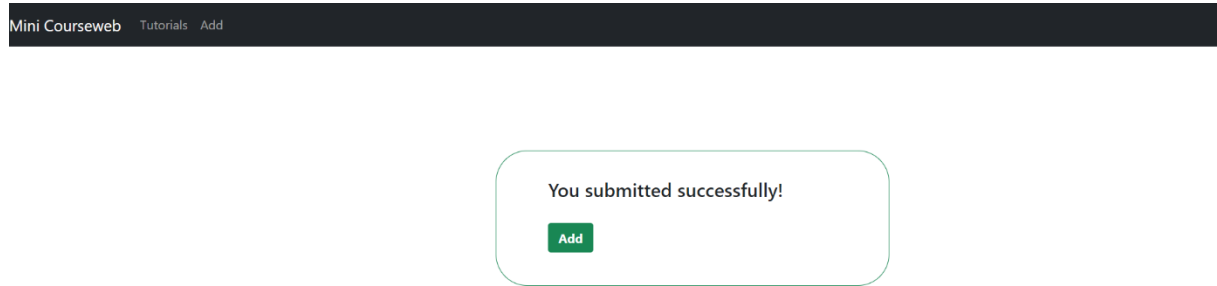


Fig. 6. Tutorial submission form

- After adding the new tutorial to the system, it will display the Mini Courseweb page in a tutorial list. The user able to click on one item in the tutorial list, it will display the details of the tutorial. (See fig 6.)

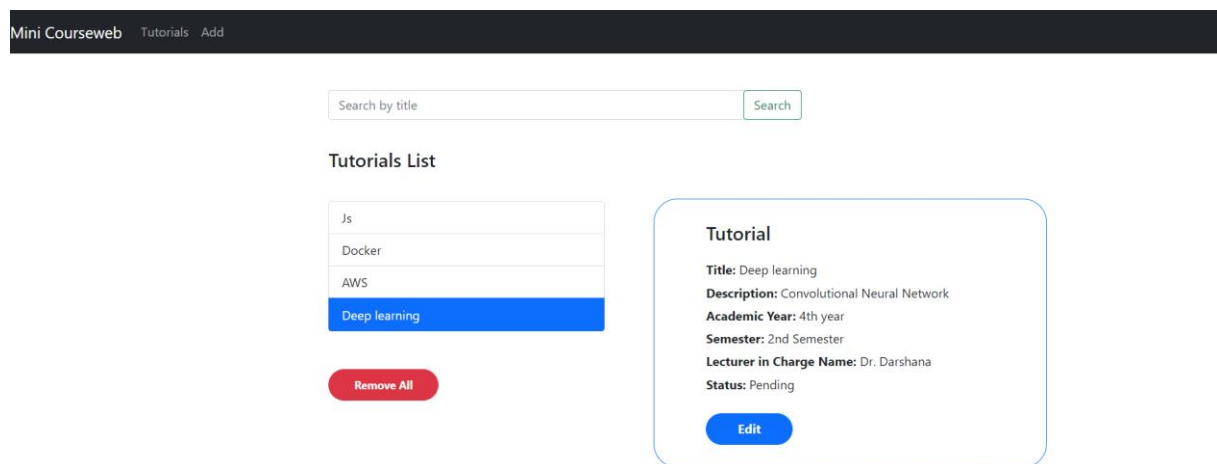


Fig. 7. View new tutorial list

- When click on the edit button in Mini Courseweb tutorial details box, it will redirect to the Tutorial Edit, delete, publish page.

Apps OneDrive YouTube Virtual machines ~ Day 1: Data Types j... Blogger: Ceylon Tec... SICCAT Software In... UX/UI Designer Res... Airport & Aviation... Other bookmarks Reading list

Mini Courseweb Tutorials Add

Tutorial

Title
Deep learning

Description
Convolutional Neural Network

Academic Year
4th year

Semester
2nd Semester

Lecturer in Charge Name
Dr. Darshana

Status: Pending

Publish Delete Update

Fig. 8. Tutorial edit form

Then click on the publish button the tutorial will be published. Then the user can see the status as Published.

Mini Courseweb Tutorials Add

Tutorial

Title
Deep learning

Description
Convolutional Neural Network

Academic Year
4th year

Semester
2nd Semester

Lecturer in Charge Name
Dr. Darshana

Status: Published

UnPublish Delete Update

Fig. 9. Change status

After edit the details, user can click on the update button for update the details and it will display a successfully updated message.

Mini Courseweb Tutorials Add

Tutorial

Title
Deep learning

Description
Convolutional Neural Network

Academic Year
4th year

Semester
2nd Semester

Lecturer in Charge Name
Dr. Darshana

Status:Published

UnPublish

Delete

Update

The tutorial was updated successfully!

Fig. 10. Update edit form

- 🎨 If the user wants to delete a tutorial record form the system, user can click on the delete button and after redirect to the tutorial list user can see that record will be deleted form the system. If the user wants to delete all the record form the system; user can click on the Remove all button.

Mini Courseweb Tutorials Add

Tutorial

Title
AWS

Description
Create AWS Account

Academic Year
4th year

Semester
2nd Semester

Lecturer in Charge Name
Ms. Thamali

Status:Published

UnPublish

Delete

Update

Fig. 11. Delete record form system

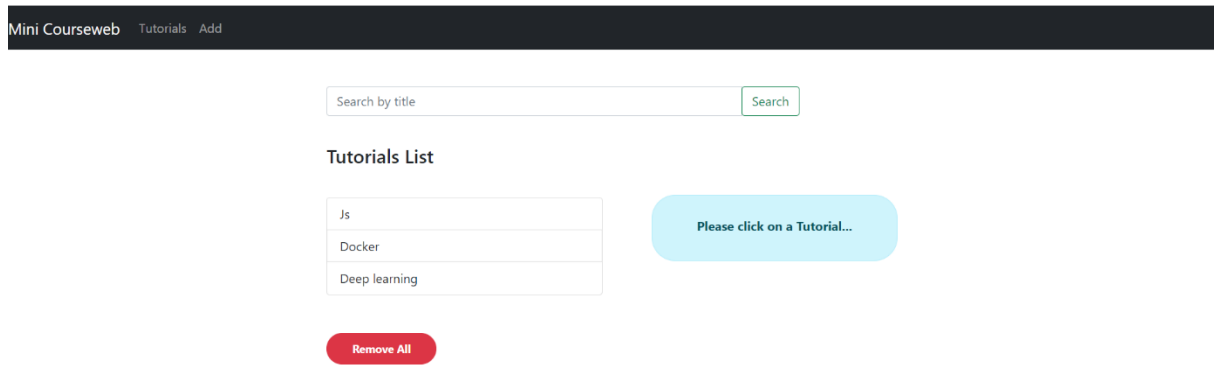



Fig. 12. After deleting, view list

 User can search the tutorial entering the Tutorial title in the search bar.

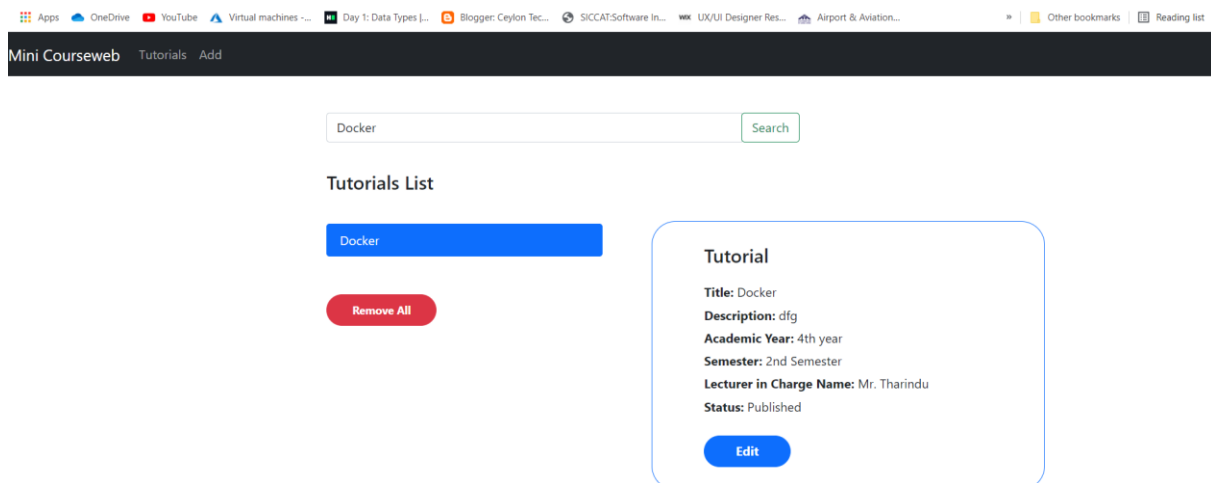


Fig. 13. Search entering tutorial title

2.2 Application Logic / API Tier

The API Layer manages of accepting HTTP requests and processing the payload that they include. This layer would then send the payload to the next Service layer, stripped of any HTTP-specific information. At this level, structural input validation is conducted, such as determining if a received string represents a number and converting a result from the Service layer to an HTTP status code and JSON response.

Only on this level does Express.js exist. You'd have a primary app.js file that configures the server, as well as specific route files. All req and res objects are left behind in route files, which specify validations and call service layers.

2.3 Database Tier

Database tier is the base of a web application and in a 3-tier architecture application, the database tier manages the data.

- Here, creates a database called db and create a collection called tutorials to assign the data types. The database creates MongoDB.
- _id is automatically created. And user can add, delete, update through this _id. db.tutorials consists of _id, title, description, academic_year, semester, lecturer_name and createdAt, and the updatedAt

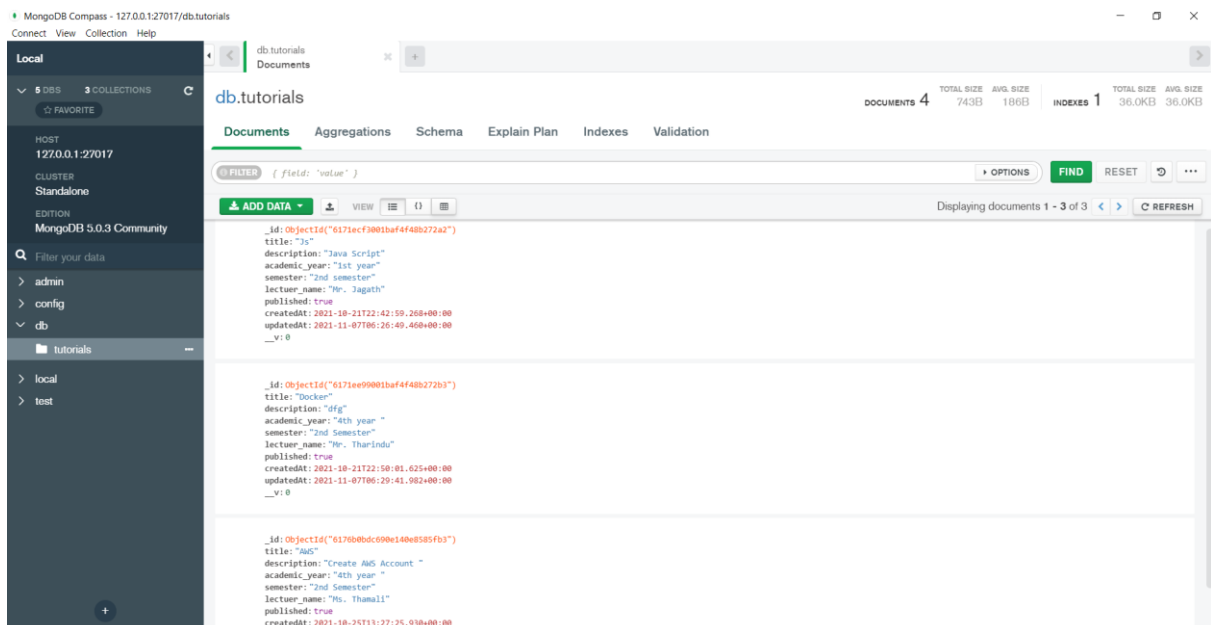


Fig. 1. *MongoDB database db.tutorials*

03 - Screenshots Of Docker Environment

The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile content is as follows:

```
Dockerfile
FROM node:14
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
```

The terminal output shows the build process:

```
> Executing task: docker build --pull --rm -f "api/Dockerfile" -t docker-mern-nginx-master:latest "api"
[+] Building 180.4s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load .dockerignore
=> [internal] load context for docker.io/library/node:14
=> [auth] library/node:pull token for registry-1.docker.io
=> [1/5] FROM docker.io/library/node:14
=> resolve docker.io/library/node:14@sha256:abc8c3d2006f8a4c1c795e55d4fbc7f54f5a3fb7318596ecb355cab8f5e7182
=> sha256:abc8c3d2006f8a4c1c795e55d4fbc7f54f5a3fb7318596ecb355cab8f5e7182 776B / 776B
=> sha256:e0dd89382d4f7da3b2be5d57b74c65bf1836deeca3c5f88919f44d69025f9 2.21kB / 2.21kB
=> sha256:312167212e3889e0e0d811015969e9e1c089792b7381f4447024d07 2.64kB / 2.64kB
=> sha256:2f0ef431671c388925ba36952f8bf2cef38072d82d1aeb269d6b6b1b84c 45.38kB / 45.38kB
=> sha256:243ae34810fbd80a0a3ec1fa7da887386e7a115913bf13ed95681c9a1cf8a4 4.34kB / 4.34kB
=> sha256:1a3d3c11106306de19f422e0da6f9b0de14702c6213d6dbbc395be1b3 11.30kB / 11.30kB
=> sha256:d01c447bcabc60448b9d84f429dd8c8d836952f1061c9c6e9847230b096 49.70kB / 49.70kB
=> sha256:18078a24def02090e47f4476eb7802121380248e99130737d991f4f057 214.44kB / 214.44kB
=> sha256:5df57a3695e0d7d2081c5c4e9138a7348a897c9eb1d2c71778a86f92d80 4.19kB / 4.19kB
=> sha256:5037d4710e071cf1f65bf24831fab509322a1e5fc589ff5309fa193b7c 35.14kB / 35.14kB
=> extracting sha256:2f0ef431671c388925ba36952f8bf2cef38072d82d1aeb269d6b6b1b84c 2.33kB
=> extracting sha256:18078a24def02090e47f4476eb7802121380248e99130737d991f4f057 12.08kB
=> extracting sha256:1a3d3c11106306de19f422e0da6f9b0de14702c6213d6dbbc395be1b3 0.55kB
=> extracting sha256:243ae34810fbd80a0a3ec1fa7da887386e7a115913bf13ed95681c9a1cf8a4 0.25kB
=> sha256:6a56af5a7808121dd05f01cfe718eb46f8cc22a34539eca6274e193ba2237 64.25kB
=> extracting sha256:d01c447bcabc60448b9d84f429dd8c8d836952f1061c9c6e9847230b096 2.09kB
=> extracting sha256:18078a24def02090e47f4476eb7802121380248e99130737d991f4f057 12.08kB
=> extracting sha256:5df57a3695e0d7d2081c5c4e9138a7348a897c9eb1d2c71778a86f92d80 0.11kB
=> extracting sha256:5037d4710e071cf1f65bf24831fab509322a1e5fc589ff5309fa193b7c 2.33kB
=> extracting sha256:7143d6fa07f3c8b4573637c3d4d6e768b07d7583677b3046dd8d76eff72270a 0.25kB
=> extracting sha256:6a56af5a7808121dd05f01cfe718eb46f8cc22a34539eca6274e193ba2237 0.08kB
=> [internal] load build context
=> transferring context: 34.49kB
=> [2/5] WORKDIR /api
=> [3/5] COPY package.json .
=> [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> exporting layers
=> writing image sha256:e7bdc537bb8e248922f56fd8e849377a10377aff61ca01ce70a5c0df8706
=> naming to docker.io/library/docker-mern-nginx-master:latest
```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

Terminal will be reused by tasks, press any key to close it.

The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile content is as follows:

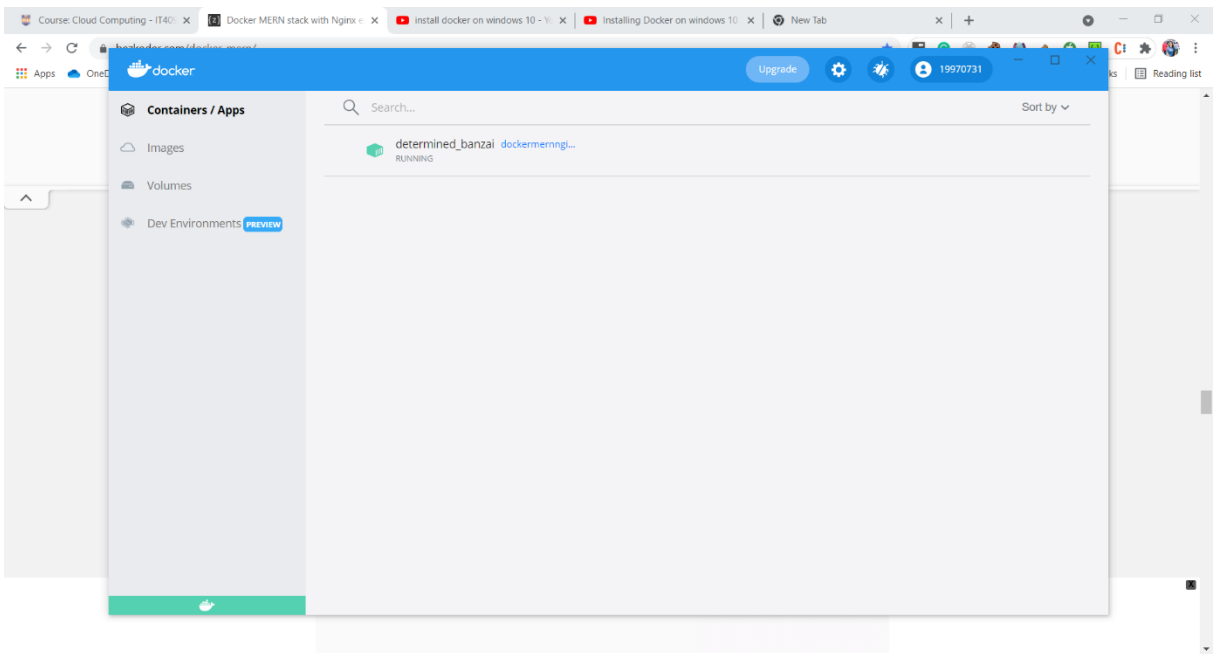
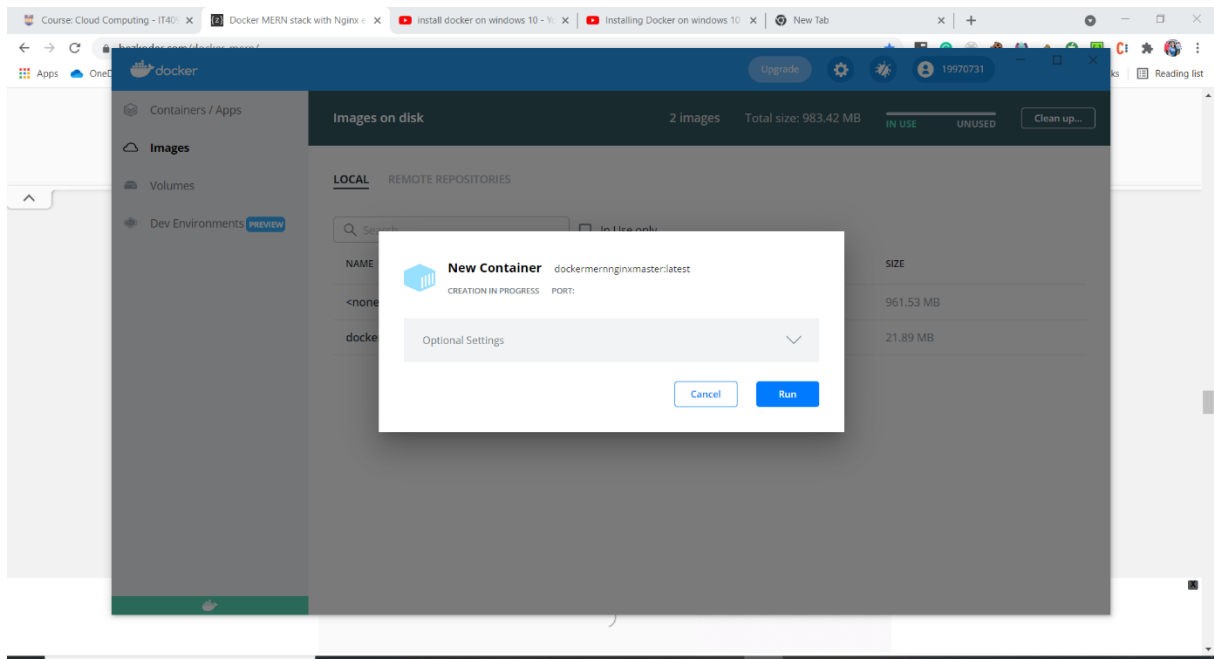
```
Dockerfile
FROM node:14
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
```

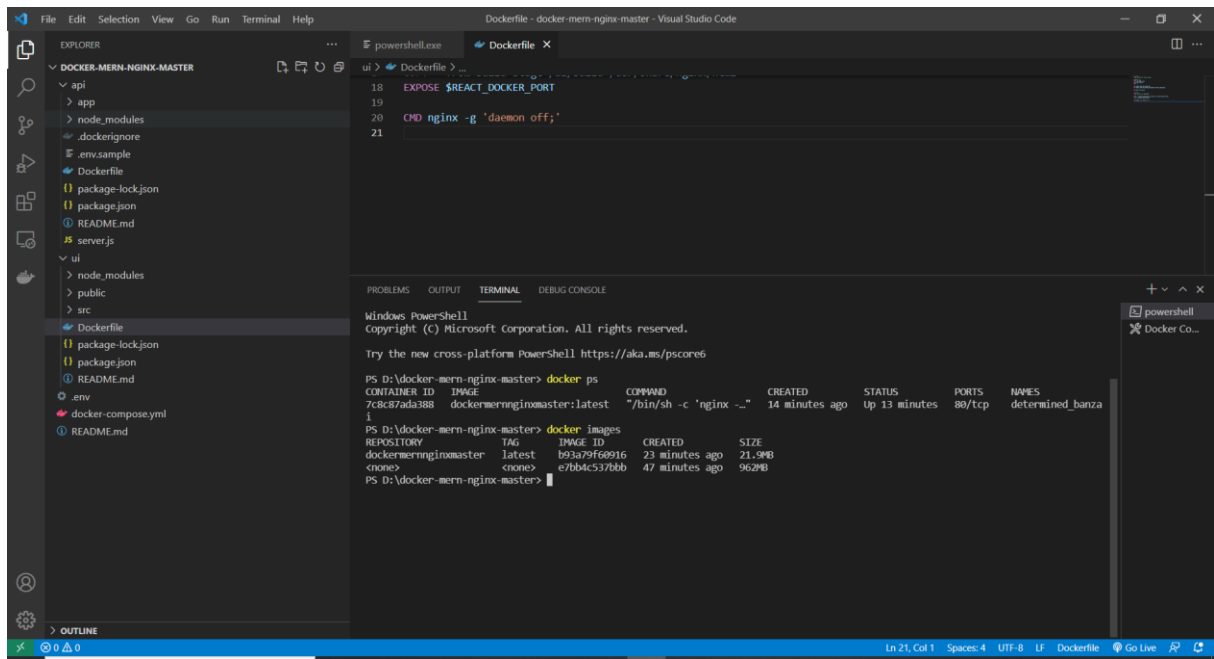
The terminal output shows the build process:

```
> Executing task: docker build --pull --rm -f "api/Dockerfile" -t docker-mern-nginx-master:latest "api"
[+] Building 1039.9s (16/16) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load .dockerignore
=> [internal] load context for docker.io/library/nginx:1.17.0-alpine
=> [internal] load metadata for docker.io/library/node:14
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load build context
=> transferring context: 289.38kB
=> CACHED [build-stage 1/6] FROM docker.io/library/node:14@sha256:abc8c3d2006f8a4c1c795e55d4fbc7f54f5a3fb7318596ecb355cab8f5e7182
=> [stage-1 1/2] FROM docker.io/library/nginx:1.17.0-alpine@sha256:b126fee820be927b1e04ae36b3f51aa4708b73bf6b1826ff19a59d22b2b4c6 0.08kB
=> resolve docker.io/library/nginx:1.17.0-alpine@sha256:b126fee820be927b1e04ae36b3f51aa4708b73bf6b1826ff19a59d22b2b4c6 0.06kB
=> sha256:b126fee820be927b1e04ae36b3f51aa4708b73bf6b1826ff19a59d22b2b4c6 1.41kB / 1.41kB
=> sha256:8b1169af0380526854e07c3b27f34b261c728f7d60f72b583450922 739B / 739B
=> sha256:b126fee820be927b1e04ae36b3f51aa4708b73bf6b1826ff19a59d22b2b4c6 7.24kB / 7.24kB
=> sha256:e7c96db7181be991f19a9fb975cddbd73c65f4a2681348e03a141a2192a5f10 2.70kB / 2.70kB
=> sha256:f0eae5c5e0b730fc083c0fbae0043a261ca70814d32092e89d4d5f99a28 5.69kB / 5.69kB
=> extracting sha256:e7c96db7181be991f19a9fb975cddbd73c65f4a2681348e03a141a2192a5f10 1.55kB
=> extracting sha256:f0eae5c5e0b730fc083c0fbae0043a261ca70814d32092e89d4d5f99a28 3.75kB
=> [build-stage 3/6] COPY package.json .
=> [build-stage 4/6] RUN npm install
=> [build-stage 5/6] COPY . .
=> [stage-1 2/2] COPY --from=build-stage /ui/build /usr/share/nginx/html
=> exporting to image
=> exporting layers
=> writing image sha256:b93a79f6091604a0e2d9ec21bf6381ef36f38728b9c9784aed05a24e9fed
=> naming to docker.io/library/docker-mern-nginx-master:latest
```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

Terminal will be reused by tasks, press any key to close it.





04 - Appendix

The screenshot shows the Visual Studio Code interface with a Dockerfile open. The Dockerfile is configured for a two-stage build. Stage 1 builds the React application using Node.js 14. Stage 2 uses an nginx image to serve the built application. The terminal shows the command to run the Docker build.

```
1 # Stage 1
2 FROM node:14 as build-stage
3
4 WORKDIR /ui
5 COPY package.json .
6 RUN npm install
7 COPY . .
8
9 ARG REACT_APP_API_BASE_URL
10 ENV REACT_APP_API_BASE_URL=$REACT_APP_API_BASE_URL
11
12 RUN npm run build
13
14 # Stage 2
15 FROM nginx:1.17.0-alpine
16
17 COPY --from=build-stage /ui/build /usr/share/nginx/html
18 EXPOSE $REACT_DOCKER_PORT
19
20 CMD nginx -g 'daemon off;'
```

Terminal output:

```
PS D:\docker-mern-nginx-master> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
7c8c87ada388   dockermernnginxmaster:latest        "/bin/sh -c 'nginx -..." 27 minutes ago Up 26 minutes 80/tcp       determined_banza

PS D:\docker-mern-nginx-master> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
dockermernnginxmaster   latest    b93a79f60916   35 minutes ago 21.9MB
<none>         <none>    e7bb4c537bbb   59 minutes ago 962MB
PS D:\docker-mern-nginx-master>
```

The screenshot shows the Visual Studio Code interface with a Dockerfile open. The Dockerfile is configured for a single-stage build of a React application using Node.js 14. The terminal shows the command to run the Docker build.

```
1 FROM node:14
2
3 WORKDIR /api
4 COPY package.json .
5 RUN npm install
6 COPY . .
7 CMD npm start
```

Terminal output:

```
PS D:\docker-mern-nginx-master> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
7c8c87ada388   dockermernnginxmaster:latest        "/bin/sh -c 'nginx -..." 27 minutes ago Up 26 minutes 80/tcp       determined_banza

PS D:\docker-mern-nginx-master> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
dockermernnginxmaster   latest    b93a79f60916   35 minutes ago 21.9MB
<none>         <none>    e7bb4c537bbb   59 minutes ago 962MB
PS D:\docker-mern-nginx-master>
```

The screenshot shows the Visual Studio Code interface for the `docker-mern-nginx-master` project. The Explorer sidebar on the left shows the file structure, with the `.env` file selected. The main editor displays the contents of `.env`:

```
1 MONGODB_USER=root
2 MONGODB_PASSWORD=
3 MONGODB_DATABASE=jb
4 MONGODB_LOCAL_PORT=27017
5 MONGODB_DOCKER_PORT=27017
6
7 NODE_LOCAL_PORT=6868
8 NODE_DOCKER_PORT=8888
9
10 CLIENT_ORIGIN=http://127.0.0.1:8888
11 CLIENT_API_BASE_URL=http://127.0.0.1:6868/api
12
13 REACT_LOCAL_PORT=8888
14 REACT_DOCKER_PORT=80
```

The terminal at the bottom shows the output of the `docker ps` command:

```
PS D:\docker-mern-nginx-master> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
7c8c87ada388   dockermerginginmaster:latest        "/bin/sh -c 'nginx -..." 27 minutes ago Up 26 minutes 80/tcp       determined_banza
```

Below the `docker ps` output, the `docker images` command output is shown:

```
PS D:\docker-mern-nginx-master> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
dockermerginginmaster   latest    b93a79f69916   35 minutes ago 21.9MB
<none>         <none>    e7bb4c537bbb   59 minutes ago 962MB
```

The status bar at the bottom indicates the file is at line 3, column 18, in UTF-8 encoding with LF line endings.

The screenshot shows the Visual Studio Code interface for the `docker-mern-nginx-master` project, with the `.env.sample` file selected in the Explorer sidebar. The main editor displays the contents of `.env.sample`:

```
1 DB_HOST=localhost
2 DB_USER=root
3 DB_PASSWORD=""
4 DB_NAME=db
5 DB_PORT=27017
6
7 NODE_DOCKER_PORT=8888
8
9 CLIENT_ORIGIN=http://127.0.0.1:8081
10
11
12
```

The terminal at the bottom shows the same output as the first screenshot, with the `docker ps` and `docker images` commands executed.


```
1 version: '3.8'
2
3 services:
4   mongodb:
5     image: mongo:5.0.2
6     restart: unless-stopped
7     env_file: ./env
8     environment:
9       - MONGO_INITDB_ROOT_USERNAME=$MONGODB_USER
10      - MONGO_INITDB_ROOT_PASSWORD=$MONGODB_PASSWORD
11     ports:
12       - $MONGODB_LOCAL_PORT:$MONGODB_DOCKER_PORT
13     volumes:
14       - db:/data/db
15     networks:
16       - backend
17
18   api:
19     depends_on:
20       - mongodb
21     build: ./api
22     restart: unless-stopped
23     env_file: ./env
24     ports:
25       - $NODE_LOCAL_PORT:$NODE_DOCKER_PORT
26     environment:
27       - DB_HOST=mongodb
28       - DB_USER=$MONGODB_USER
29       - DB_PASSWORD=$MONGODB_PASSWORD
30       - DB_NAME=$MONGODB_DATABASE
31       - DB_PORT=$MONGODB_DOCKER_PORT
32       - CLIENT_ORIGIN=$CLIENT_ORIGIN
33     networks:
34       - backend
35       - frontend
36
37   ui:
38     depends_on:
```

```
24 ports:
25   - $NODE_LOCAL_PORT:$NODE_DOCKER_PORT
26
27 environment:
28   - DB_HOST=mongodb
29   - DB_USER=$MONGODB_USER
30   - DB_PASSWORD=$MONGODB_PASSWORD
31   - DB_NAME=$MONGODB_DATABASE
32   - DB_PORT=$MONGODB_DOCKER_PORT
33   - CLIENT_ORIGIN=$CLIENT_ORIGIN
34
35 networks:
36   - backend
37   - frontend
38
39 ui:
40   depends_on:
41     - api
42   build:
43     context: ./ui
44     args:
45       - REACT_APP_API_BASE_URL=$CLIENT_API_BASE_URL
46   ports:
47     - $REACT_LOCAL_PORT:$REACT_DOCKER_PORT
48   networks:
49     - frontend
50
51 volumes:
52   db:
53
54 networks:
55   backend:
56   frontend:
```