DEVOPS PROJECT

I/ CRUD application with React js Node js MySQL

CRUD (Create, Read, Update, Delete) applications are a staple in software development. They allow users to interact with databases, retrieve information, and make changes to data.

In this DevOps project, we will build a CRUD application using React js for the frontend, Node js for the backend, and MySQL as the database. This will involve setting up a development environment, designing a user interface, creating an API, implementing the necessary database queries, and integrating the frontend and backend.

II/ Installation

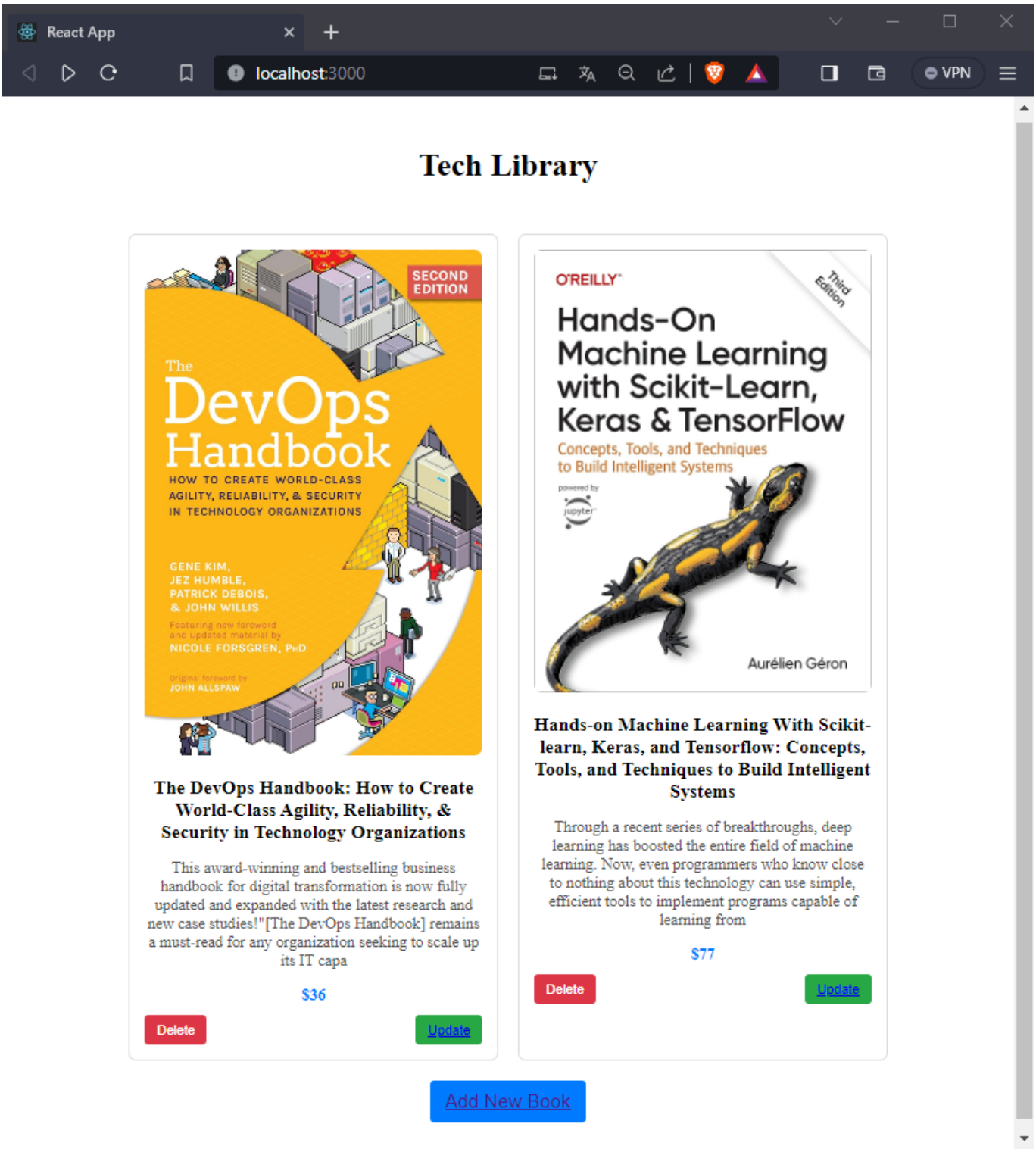
Before starting, run this command in folder **./project/backend** and **./project/client**

```
npm i
```

III/ Functionalities

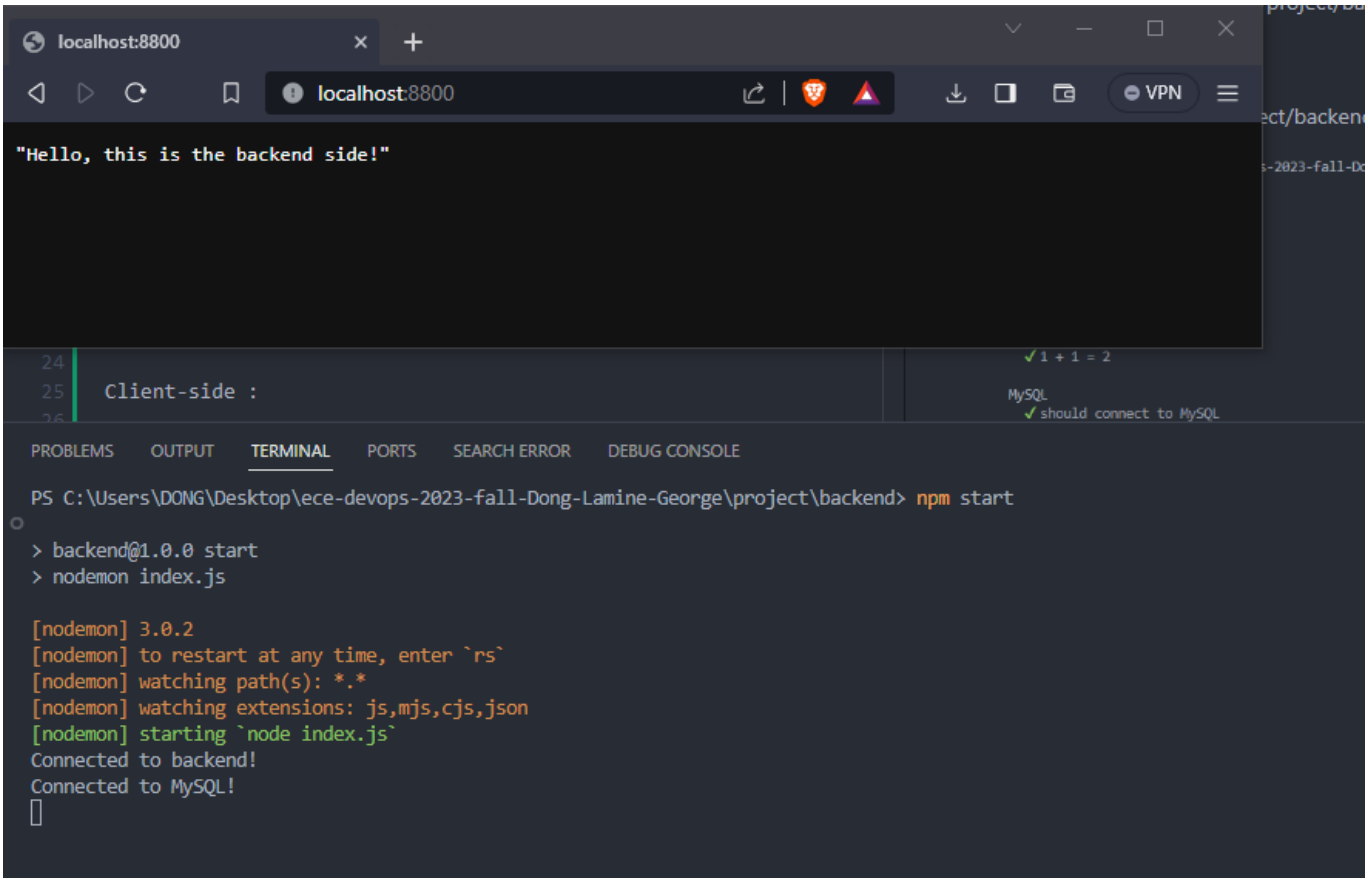
Client-side :

- **npm run start** in folder ./project/client :

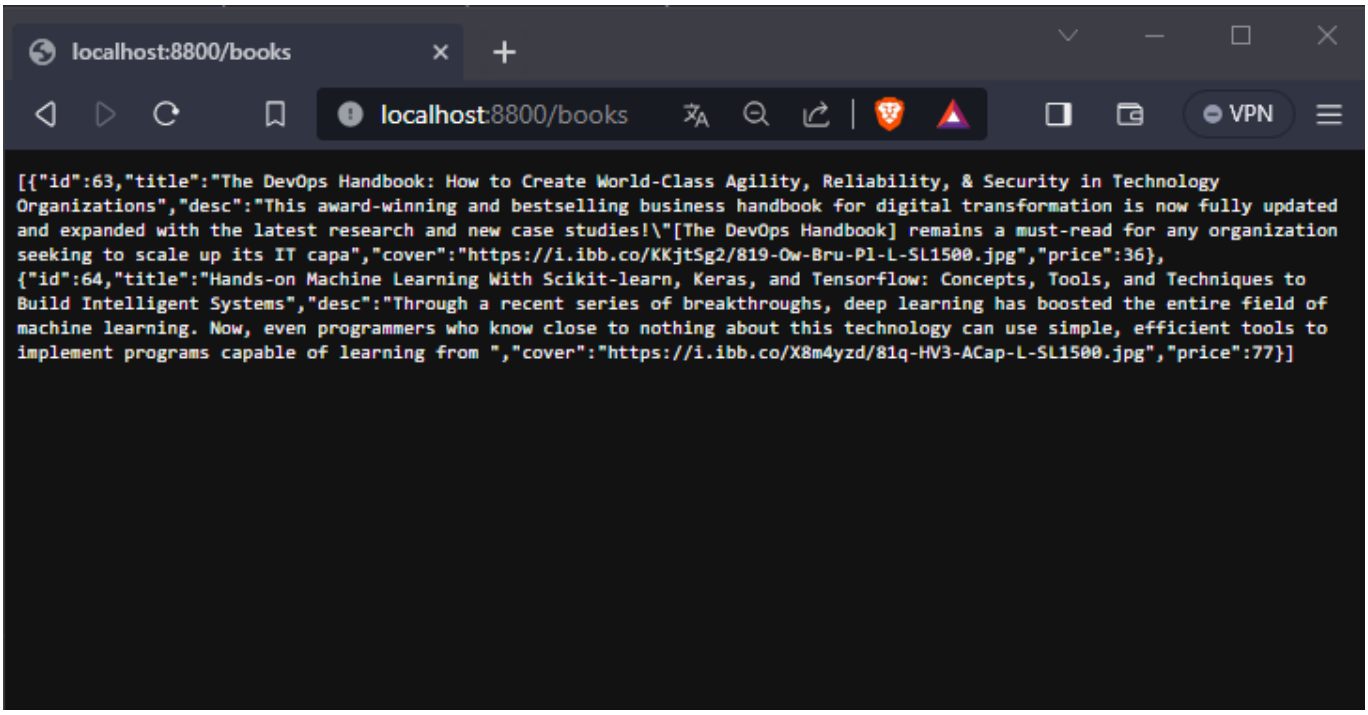


Server-side :

- **npm run start** in folder ./project/backend :



fetching all data from MySQL database :



IV/ Apply CI/CD pipeline

1) Run unit test in local

- **npm test** in folder ./project/backend to run unit test :

```
PS C:\Users\DONG\Desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> npm test

> backend@1.0.0 test
> mocha test/*.js

Connected to backend!
Connected to MySQL!


  Simple test
    ✓ 1 + 1 = 2

  MySQL
    ✓ should connect to MySQL

  Books REST API
    ✓ should GET all books
    ✓ should POST a new book
    ✓ should UPDATE an existing book
    ✓ should DELETE an existing book

  Books BACK TEST
    ✓ should GET all books
    ✓ should create a new book
    ✓ should update an existing book
    ✓ should delete an existing book


  10 passing (65ms)

```

2) Github Action workflow :

build (18.x)
succeeded now in 41s

Search logs

Set up job	1s
Initialize containers	15s
Run actions/checkout@v3	1s
Use Node.js 18.x	2s
Run npm ci	7s
Run npm run build --if-present	0s
Debug Information	0s
Start MySQL Service	4s
Set up MySQL	0s
Set MySQL Host and Port	0s
Verify MySQL connection	0s
Create MySQL Table	0s
Run Tests	1s
Post Use Node.js 18.x	2s
Post Run actions/checkout@v3	0s
Stop containers	1s
Complete job	0s

Check all folders :

Debug Information 0s

```
1  ▶ Run echo "Current directory: $(pwd)"
5  Current directory: /home/runner/work/ece-devops-2023-fall-Dong-Lamine-George/ece-devops-2023-fall-Dong-Lamine-George/project/backend
6  Contents of project/backend: total 160
7  drwxr-xr-x   4 runner docker   4096 Dec 29 13:45 .
8  drwxr-xr-x   4 runner docker   4096 Dec 29 13:45 ..
9  -rw-r--r--   1 runner docker     18 Dec 29 13:45 .gitignore
10 -rw-r--r--   1 runner docker    203 Dec 29 13:45 create_table.sql
11 -rw-r--r--   1 runner docker    466 Dec 29 13:45 dockerfile
12 -rw-r--r--   1 runner docker   3147 Dec 29 13:45 index.js
13 drwxr-xr-x 252 runner docker  12288 Dec 29 13:45 node_modules
14 -rw-r--r--   1 runner docker 115171 Dec 29 13:45 package-lock.json
15 -rw-r--r--   1 runner docker    672 Dec 29 13:45 package.json
16 drwxr-xr-x   2 runner docker   4096 Dec 29 13:45 test
```

Start and configure MySQL service :

Start MySQL Service 4s

```
1  ▶ Run sudo service mysql start
4
```

Set up MySQL 0s

```
1  ▶ Run mysql -e 'CREATE DATABASE test;' -uroot -proot
4
4  Warning: arning] Using a password on the command line interface can be insecure.
```

Set MySQL Host and Port 0s

```
1  ▶ Run echo "MYSQL_HOST=127.0.0.1" >> $GITHUB_ENV
5
```

Check the connection :

```
Verify MySQL connection                                                    0s
1  ▶ Run mysql --version
10 mysql  Ver 8.0.35-0ubuntu0.22.04.1 for Linux on x86_64 ((Ubuntu))
11 MySQL Host: 127.0.0.1
12 MySQL Port: 3306
13 Warning: arning] Using a password on the command line interface can be insecure.
14 Database
15 information_schema
16 mysql
17 performance_schema
18 sys
19 test
Create MySQL Table                                                         0s
1  ▶ Run mysql --host 127.0.0.1 --port 3306 -uroot -proot test < create_table.sql
7  Warning: arning] Using a password on the command line interface can be insecure.
```

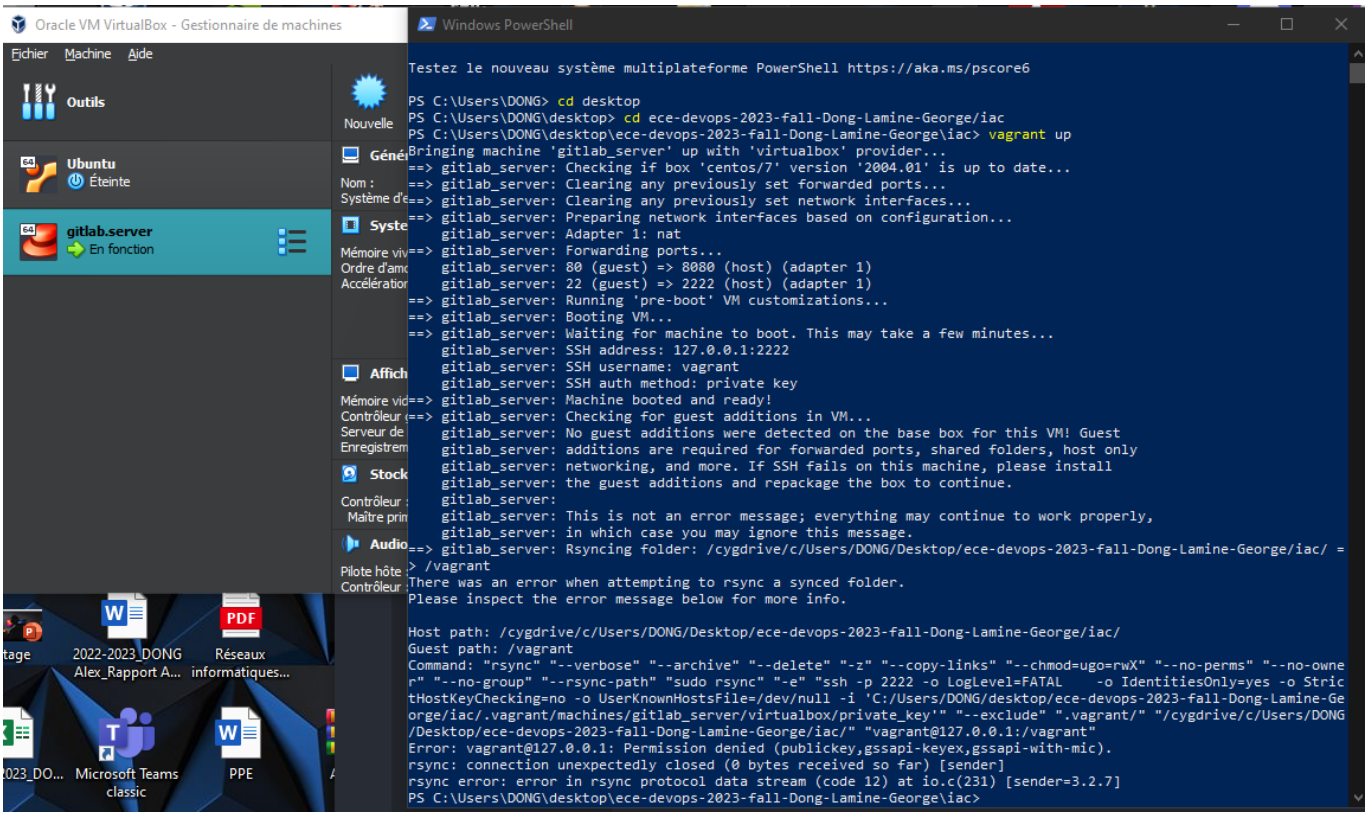
Check unit test :

```
build (18.x)                                                   Search logs
succeeded now in 41s
Create MySQL Table                                                         0s
Run Tests                                                                  1s
1  ▶ Run npm test
7
8
9  > backend@1.0.0 test
10 > mocha test/*.js
12
13 Connected to backend! listening port: 8880
14 Connected to MySQL!
15
16
17   Simple test
18     ✓ 1 + 1 = 2
19
20   MySQL
21     ✓ should connect to MySQL
22
23   Books REST API
24     ✓ should GET all books
25     ✓ should POST a new book
26     ✓ should UPDATE an existing book
27     ✓ should DELETE an existing book
26
27   Books BACK TEST
28     ✓ should GET all books
29     ✓ should create a new book
30     ✓ should update an existing book
31     ✓ should delete an existing book
32 MySQL connection closed successfully.
Post Use Node.js 18.x                                                      2s
```

V/ Configure and provision a virtual environment and run your application using the IaC approach

Problem encountered in this section :

Unable to synchronize the project folder between the guest machine and the host machine.



Solution used are (but doesn't work for us):

Basic usage configuration :

```
config.vm.synced_folder "../project", "/srv/website"
```

link source

Install a plugin:

```
$ vagrant plugin install vagrant-vbguest
```

link source

```
$ vagrant plugin install vagrant-rsync-back
```

link source

VI/ Build Docker image of your application

1) Build MySQL image :

Firstly we need to pull an official MySQL image

```
docker pull mysql
```

then run the image

```
docker run mysql
```

but it ask to specify a password in the environment

```
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend>
                                                                                > docker run mysql
2023-12-28 22:39:52+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
2023-12-28 22:39:52+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2023-12-28 22:39:52+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
2023-12-28 22:39:52+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option is not specified
    You need to specify one of the following as an environment variable:
    - MYSQL_ROOT_PASSWORD
    - MYSQL_ALLOW_EMPTY_PASSWORD
    - MYSQL_RANDOM_ROOT_PASSWORD
```

```
docker run --name some-mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root -d mysql
```

`-e` to set the environment

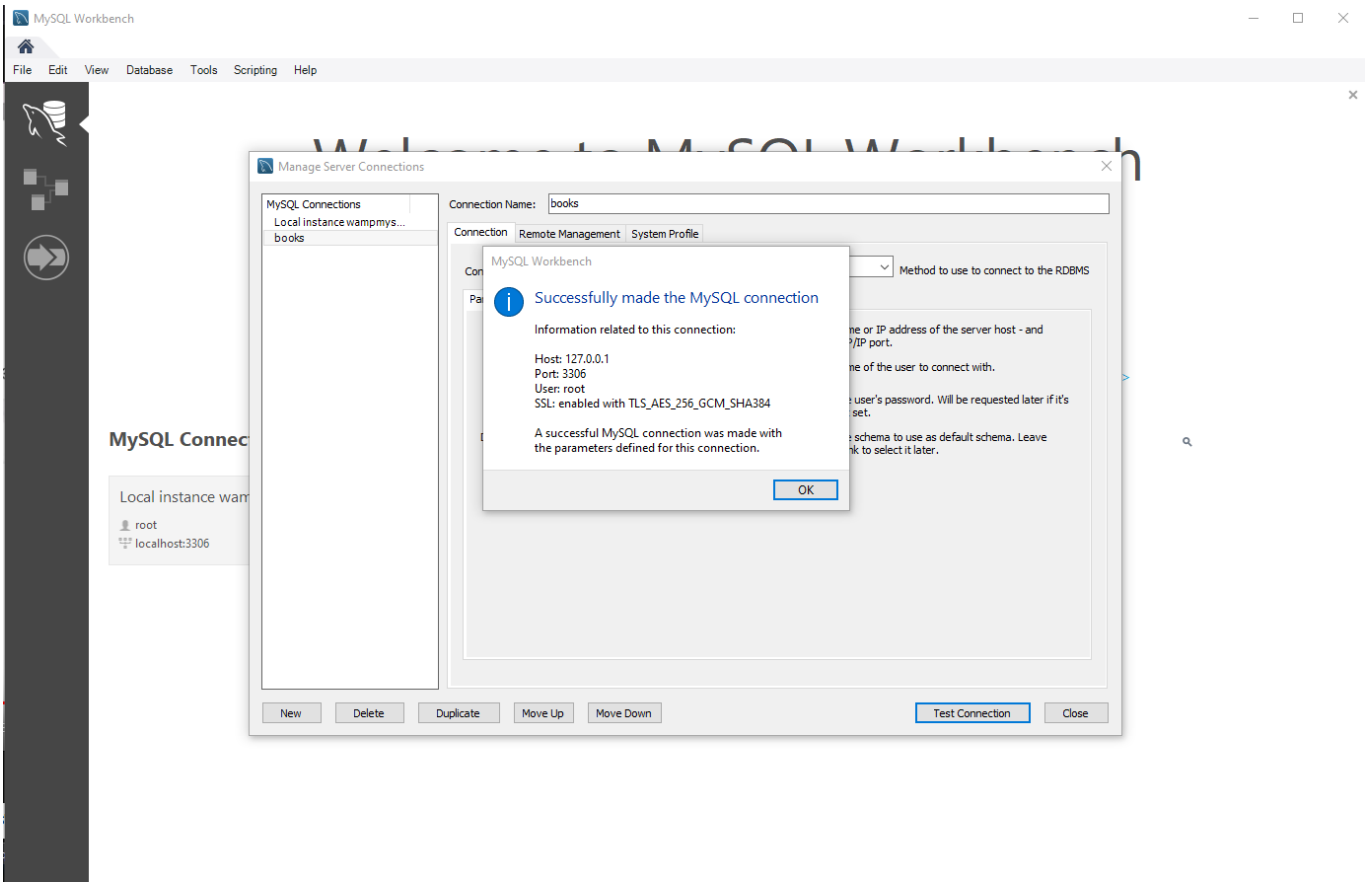
`-p` set or add port number

`-d` to run the image in the background

Now MySQL is running

```
Windows PowerShell
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> docker run --name some-mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD
=root -d mysql
2f61176b1bdce453c420c0e5cd7fc7cd3fcbe1b4a1f685b1f388e6292ffce5ee
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                                   NAMES
2f61176b1bdc   mysql     "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->3306/tcp, 33060/tcp   some-mysql
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend>
```

We check the connection to the DB using MySQL workbench for exemple :



2) Build the backend :

build back-end app image :

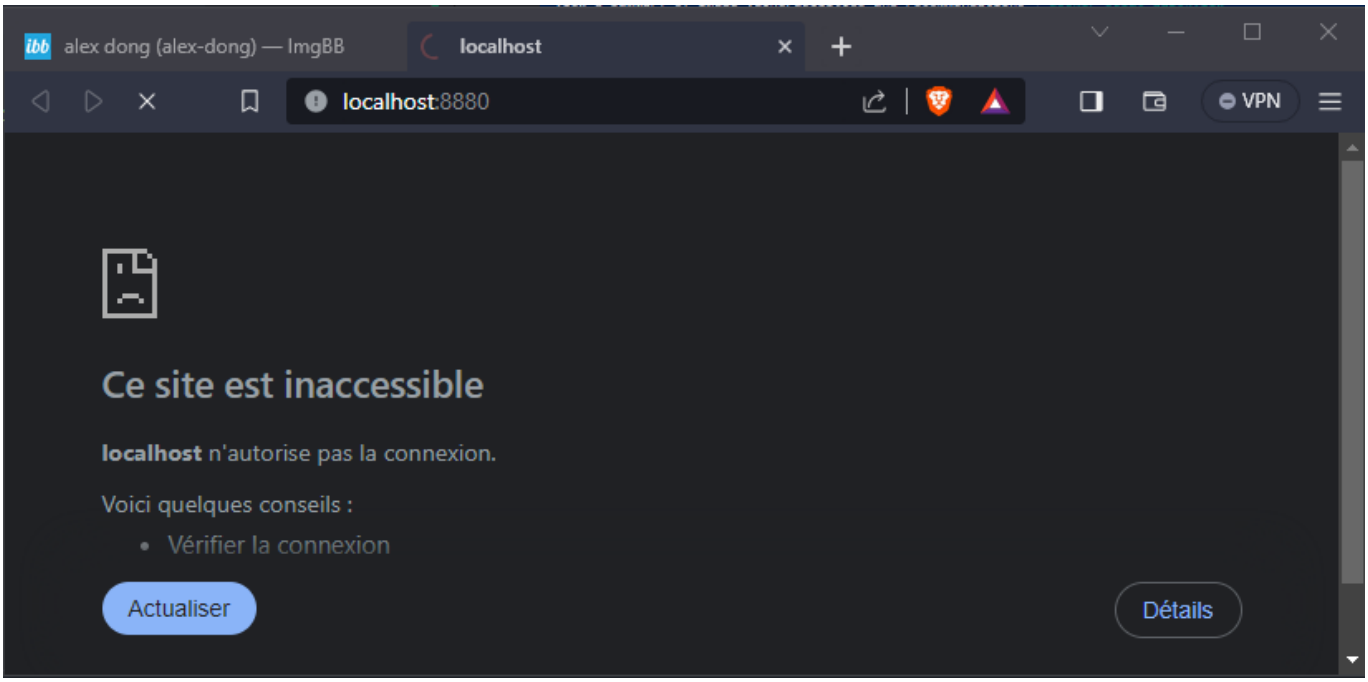
```
docker build -t bookstore .
```

Run the app in Docker container :

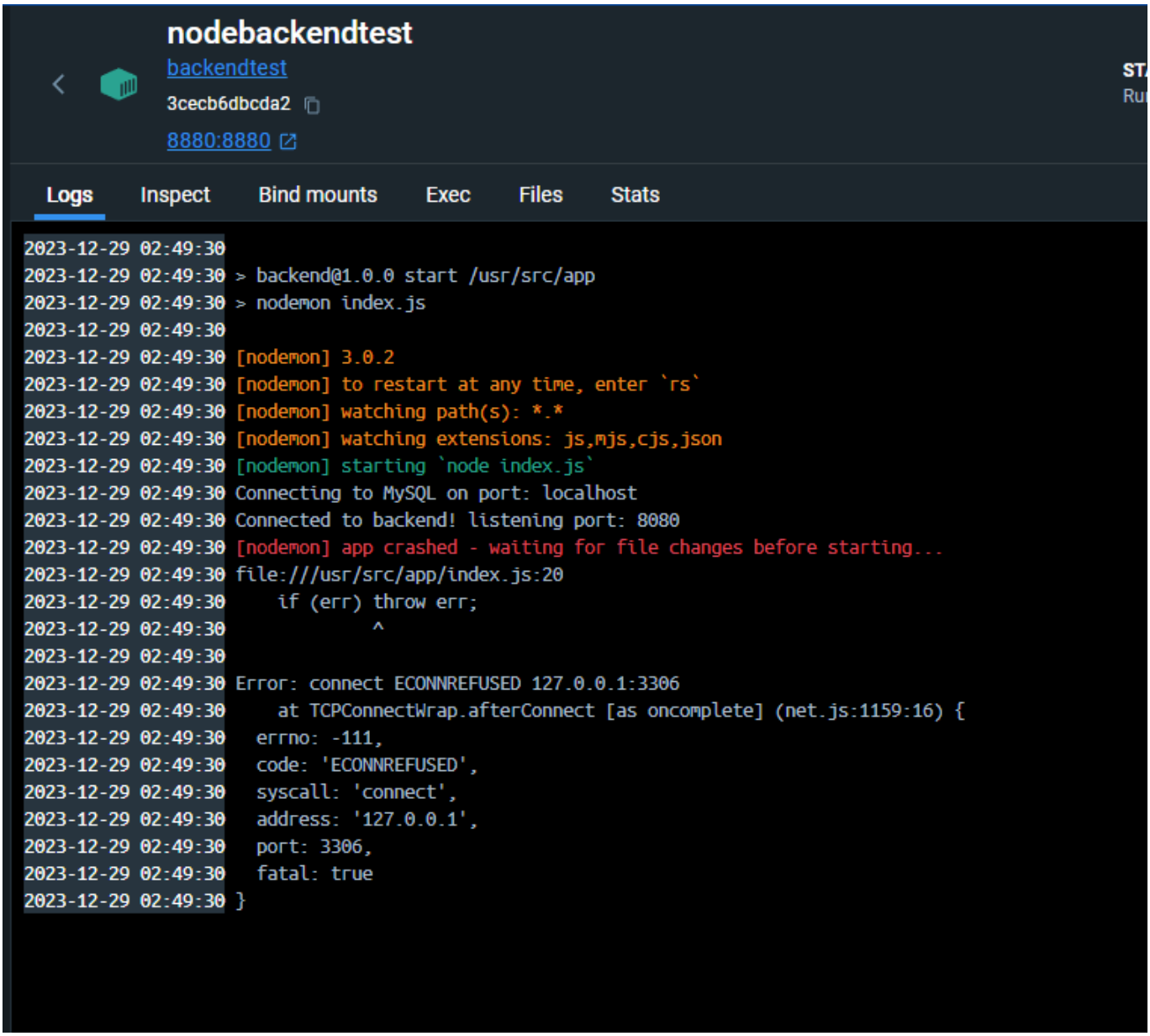
```
docker run --name nodebackendtest -p 8880:8880 -d bookstore
```

If we set our app in different port (8080) and run it in our local machine we can access and retrieve the remote data from docker container MySQL database.

We aim to establish communication between our application, residing in one container, and our database located in another container. When attempting to launch our application from the container and access it via our local browser at "http://localhost:8880," we anticipate the homepage to showcase the message '**Hello, this is the backend side!.**' Regrettably, this expected outcome does not materialize



The issue stemmed from a disparity in network configurations. Specifically, the application attempted to connect to the '127.0.0.1' network, while the Docker container hosting the database operated on the "172.17.0.1" network

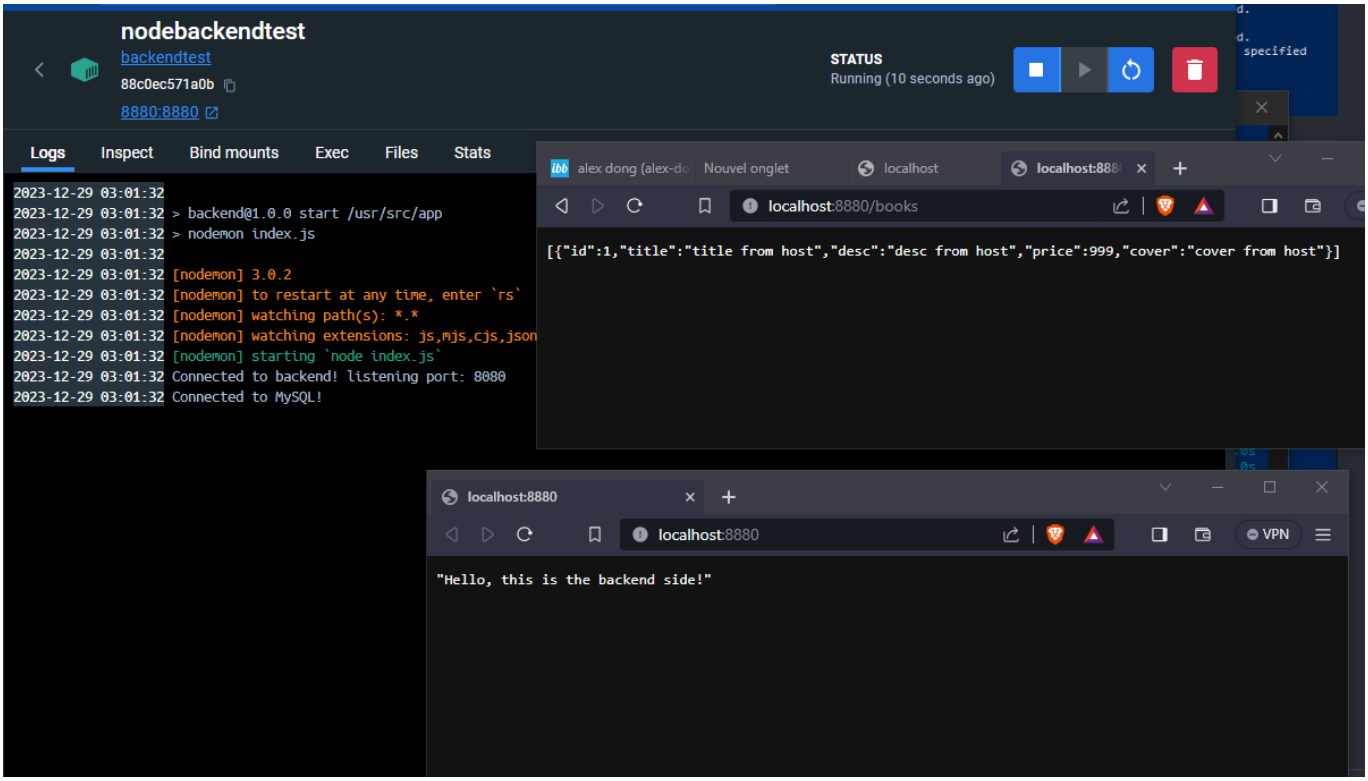


Executing this command allows us to identify the host network to which the database is connected.

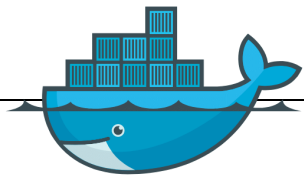
```
docker inspect [name-mysql-container]
```

```
                "HostIP": "0.0.0.0",
                "HostPort": "3306"
            }
        ],
        "33060/tcp": null
    },
    "SandboxKey": "/var/run/docker/netns/94c8dc97bf27",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "41c31d67819d09c4125821a70b1177c802234adc2543593a4dd89cf634892eb
    "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,
```


Ultimately, we can now gain access to our server-side.



VII/ Make container orchestration using Docker Compose



Create a docker-compose.yml file and run this command to start :

```
docker-compose up
```

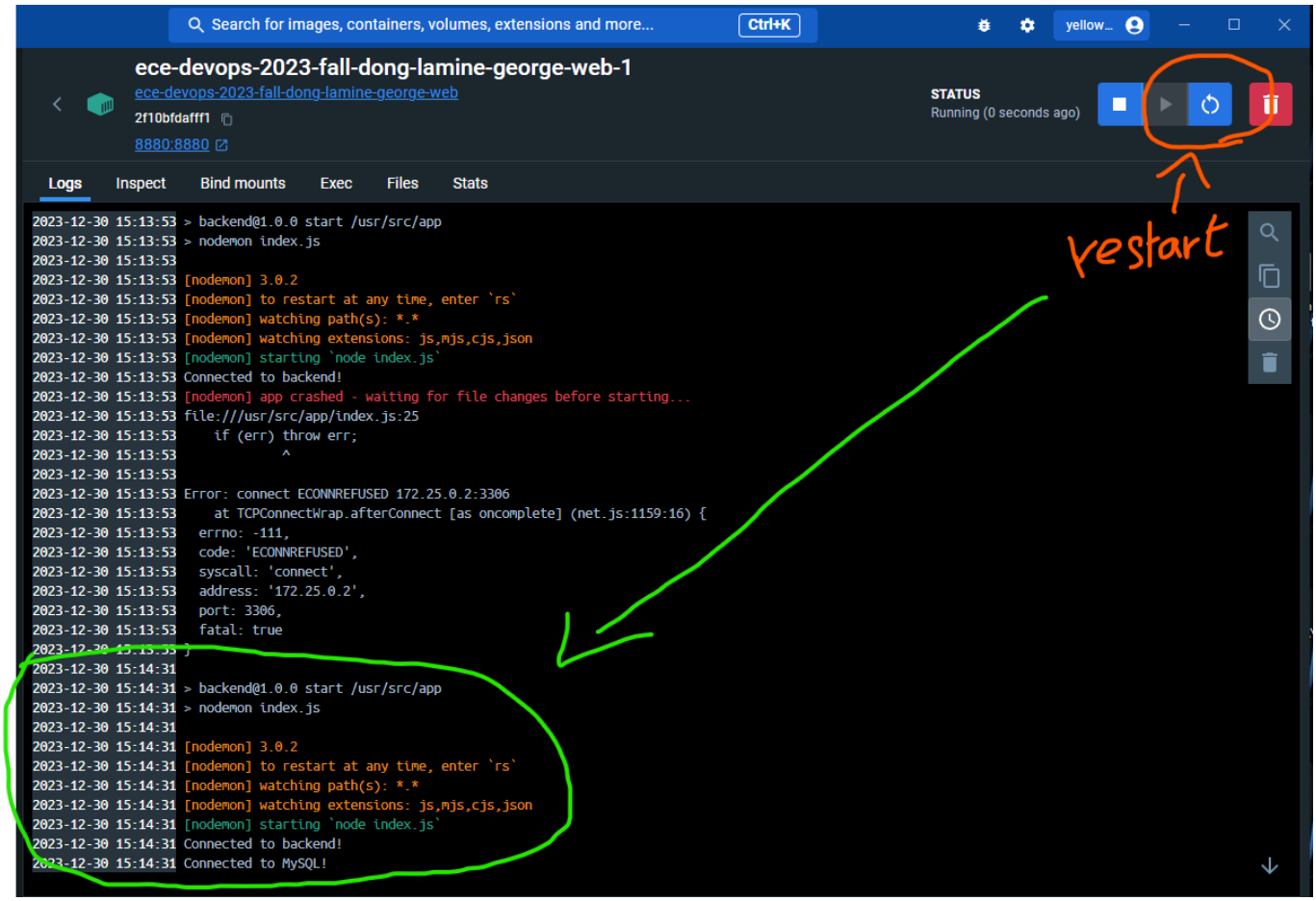
We encounter an issue during the startup of our Docker Compose file where the application launches before the MySQL initialization process is complete.

```
PS C:\Users\DONG\Desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> docker-compose up
2023/12/30 15:13:49 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
 [+] Building 2.1s (19/19) FINISHED                                                                                    docker:default
  => [mysql internal] load build definition from Dockerfile                                                                      0.0s
  => => transferring dockerfile: 116B                                                                                            0.0s
  => [mysql internal] load .dockerignore                                                                                         0.0s
  => => transferring context: 2B                                                                                                 0.0s
  => [mysql internal] load metadata for docker.io/library/mysql:latest                                                           0.8s
  => [mysql auth] library/mysql:pull token for registry-1.docker.io                                                              0.0s
  => [mysql internal] load build context                                                                                         0.0s
  => => transferring context: 38B                                                                                                0.0s
  => [mysql 1/2] FROM docker.io/library/mysql@sha256:4ef30b2c11a3366d7bb9ad95c70c0782ae435df52d046553ed931621ea36ffa5             0.0s
  => CACHED [mysql 2/2] COPY ./create_table.sql /docker-entrypoint-initdb.d/                                                     0.0s
  => [mysql] exporting to image                                                                                                  0.0s
  => => exporting layers                                                                                                         0.0s
  => => writing image sha256:6ad717e859869a6e6d7a5a69380b8ca44563435bf3b9a78c5af0efef22fb75b9                                    0.0s
  => => naming to docker.io/library/ece-devops-2023-fall-dong-lamine-george-mysql                                                0.0s
  => [web internal] load .dockerignore                                                                                           0.0s
  => => transferring context: 2B                                                                                                 0.0s
  => [web internal] load build definition from Dockerfile                                                                        0.0s
  => => transferring dockerfile: 518B                                                                                            0.0s
  => [web internal] load metadata for docker.io/library/node:14                                                                  0.5s
  => [web auth] library/node:pull token for registry-1.docker.io                                                                 0.0s
  => [web 1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa              0.0s
  => [web internal] load build context                                                                                           0.3s
  => => transferring context: 252.90kB                                                                                           0.3s
  => CACHED [web 2/5] WORKDIR /usr/src/app                                                                                       0.0s
  => CACHED [web 3/5] COPY package*.json ./                                                                                      0.0s
  => CACHED [web 4/5] RUN npm install                                                                                            0.0s
  => CACHED [web 5/5] COPY . .                                                                                                   0.0s
  => [web] exporting to image                                                                                                    0.0s
  => => exporting layers                                                                                                         0.0s
  => => writing image sha256:39db40e0770975c6ebe41bc3994673e12b0266def5086b8fc3bbf9be7d8617af                                    0.0s
  => => naming to docker.io/library/ece-devops-2023-fall-dong-lamine-george-web                                                  0.0s
 [+] Running 3/3
  ✓ Network ece-devops-2023-fall-dong-lamine-george_default  Created                                                            0.1s
  ✓ Container some-mysql                                     Created                                                             0.0s
de Link   Generate Commit Message   Explain Code   Comment Code   Code Chat   Blackbox   Search Error   Ln 33, Col 17   Spaces: 2   UTF-8   CRLF   Compose   Go Live   Blackbox   Prettier
```

```
Attaching to ece-devops-2023-fall-dong-lamine-george-web-1, some-mysql
some-mysql                                  | 2023-12-30 14:13:52+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
some-mysql                                  | 2023-12-30 14:13:52+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
some-mysql                                  | 2023-12-30 14:13:52+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
some-mysql                                  | 2023-12-30 14:13:52+00:00 [Warn] [Entrypoint]: MYSQL_PASSWORD specified, but missing MYSQL_USER; MYSQL_PASSWORD will be ignored
some-mysql                                  | 2023-12-30 14:13:52+00:00 [Note] [Entrypoint]: Initializing database files
some-mysql                                  | 2023-12-30T14:13:52.849535Z 0 [System] [MY-015017] [Server] MySQL Server Initialization - start.
some-mysql                                  | 2023-12-30T14:13:52.851808Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release
. Please use SET GLOBAL host_cache_size=0 instead.
some-mysql                                  | 2023-12-30T14:13:52.851925Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.2.0) initializing of server in progress as process 82
some-mysql                                  | 2023-12-30T14:13:52.864575Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
ece-devops-2023-fall-dong-lamine-george-web-1  |
ece-devops-2023-fall-dong-lamine-george-web-1  | > backend@1.0.0 start /usr/src/app
ece-devops-2023-fall-dong-lamine-george-web-1  | > nodemon index.js
ece-devops-2023-fall-dong-lamine-george-web-1  |
ece-devops-2023-fall-dong-lamine-george-web-1  | [nodemon] 3.0.2
ece-devops-2023-fall-dong-lamine-george-web-1  | [nodemon] to restart at any time, enter `rs`
ece-devops-2023-fall-dong-lamine-george-web-1  | [nodemon] watching path(s): *.*
ece-devops-2023-fall-dong-lamine-george-web-1  | [nodemon] watching extensions: js,mjs,cjs,json
ece-devops-2023-fall-dong-lamine-george-web-1  | [nodemon] starting `node index.js`
some-mysql                                  | 2023-12-30T14:13:53.451006Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
ece-devops-2023-fall-dong-lamine-george-web-1  | Connected to backend!
ece-devops-2023-fall-dong-lamine-george-web-1  | file:///usr/src/app/index.js:25
ece-devops-2023-fall-dong-lamine-george-web-1  |     if (err) throw err;
ece-devops-2023-fall-dong-lamine-george-web-1  |             ^
ece-devops-2023-fall-dong-lamine-george-web-1  |
ece-devops-2023-fall-dong-lamine-george-web-1  |
ece-devops-2023-fall-dong-lamine-george-web-1  | Error: connect ECONNREFUSED 172.25.0.2:3306
ece-devops-2023-fall-dong-lamine-george-web-1  |     at TCPConnectWrap.afterConnect [as oncomplete] (net.js:1159:16) {
ece-devops-2023-fall-dong-lamine-george-web-1  |   errno: -111,
ece-devops-2023-fall-dong-lamine-george-web-1  |   code: 'ECONNREFUSED',
ece-devops-2023-fall-dong-lamine-george-web-1  |   syscall: 'connect',
ece-devops-2023-fall-dong-lamine-george-web-1  |   address: '172.25.0.2',
ece-devops-2023-fall-dong-lamine-george-web-1  |   port: 3306,
ece-devops-2023-fall-dong-lamine-george-web-1  |   fatal: true
ece-devops-2023-fall-dong-lamine-george-web-1  | }
ece-devops-2023-fall-dong-lamine-george-web-1  | [nodemon] app crashed - waiting for file changes before starting...
some-mysql                                  | 2023-12-30T14:13:55.289196Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the
```

```
some-mysql                                  | 2023-12-30T14:14:00.275034Z 0 [System] [MY-015016] [Server] MySQL Server - end.
some-mysql                                  | 2023-12-30 14:14:00+00:00 [Note] [Entrypoint]: Temporary server started.
some-mysql                                  | '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
some-mysql                                  | Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
some-mysql                                  | Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
some-mysql                                  | Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
some-mysql                                  | Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping it.
some-mysql                                  | Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
some-mysql                                  | Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
some-mysql                                  | 2023-12-30 14:14:02+00:00 [Note] [Entrypoint]: Creating database test
some-mysql                                  |
some-mysql                                  | 2023-12-30 14:14:02+00:00 [Note] [Entrypoint]: /usr/local/bin/docker-entrypoint.sh: running /docker-entrypoint-initdb.d/create_table.sql
some-mysql                                  |
some-mysql                                  |
some-mysql                                  | 2023-12-30 14:14:02+00:00 [Note] [Entrypoint]: Stopping temporary server
some-mysql                                  | 2023-12-30T14:14:02.729100Z 12 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld (Version: 8.2.0).
some-mysql                                  | 2023-12-30T14:14:04.132674Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.2.0)  MySQL Community Server - GPL.
some-mysql                                  | 2023-12-30T14:14:04.135682Z 0 [System] [MY-015016] [Server] MySQL Server - end.
some-mysql                                  | 2023-12-30 14:14:04+00:00 [Note] [Entrypoint]: Temporary server stopped
some-mysql                                  |
some-mysql                                  | 2023-12-30 14:14:04+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.
some-mysql                                  |
some-mysql                                  | 2023-12-30T14:14:04.744594Z 0 [System] [MY-015015] [Server] MySQL Server - start.
some-mysql                                  | 2023-12-30T14:14:04.924281Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release
. Please use SET GLOBAL host_cache_size=0 instead.
some-mysql                                  | 2023-12-30T14:14:04.925548Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.2.0) starting as process 1
some-mysql                                  | 2023-12-30T14:14:04.932570Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
some-mysql                                  | 2023-12-30T14:14:05.063298Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
some-mysql                                  | 2023-12-30T14:14:05.332453Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
some-mysql                                  | 2023-12-30T14:14:05.332494Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported
 for this channel.
some-mysql                                  | 2023-12-30T14:14:05.336112Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is acc
essible to all OS users. Consider choosing a different directory.
some-mysql                                  | 2023-12-30T14:14:05.354691Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysq
ld/mysqlx.sock
some-mysql                                  | 2023-12-30T14:14:05.354790Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.2.0'  socket: '/var/run/mysqld/m
ysqld.sock'  port: 3306  MySQL Community Server - GPL.
```

Once the initialization process is completed, it is necessary to restart the container for our backend application



It looks like we have a Docker Compose file that defines two services, mysql and web. The web service depends on the mysql service, which means Docker Compose will ensure that the mysql service is started before the web service.

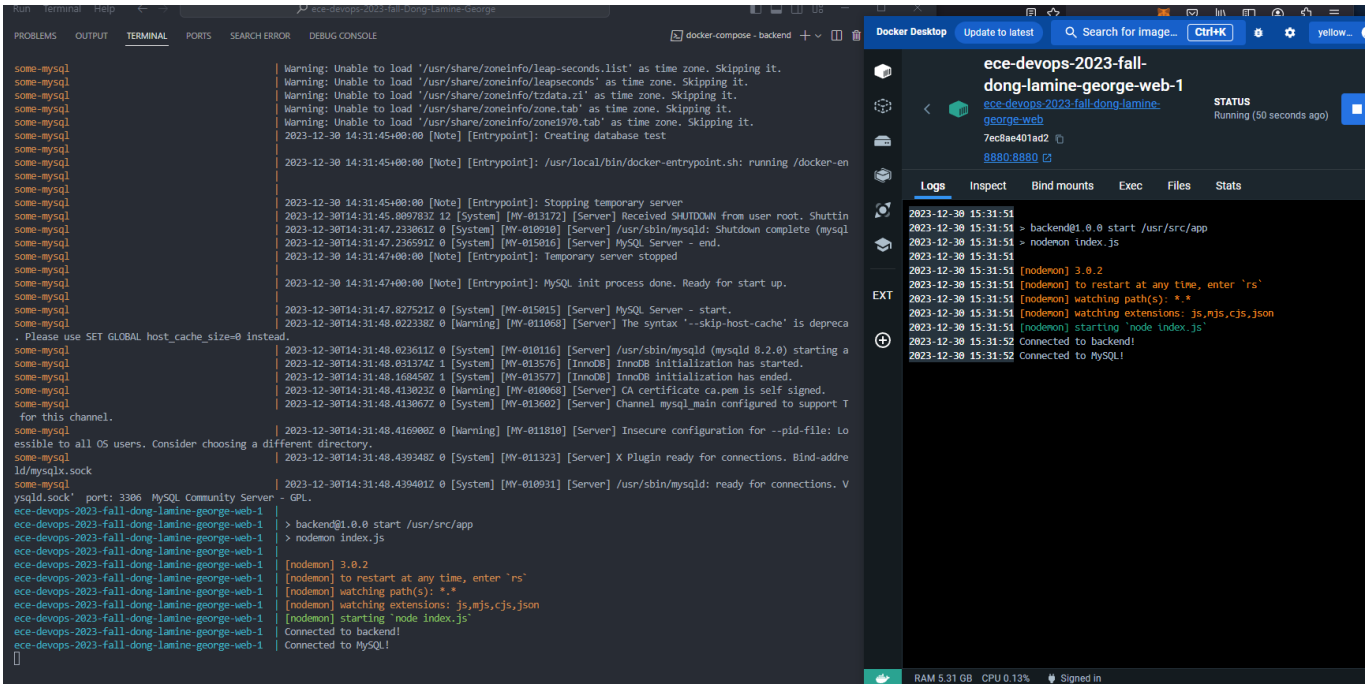
```
services:
  mysql:
    build: ./project/backend/db
    container_name: some-mysql
    environment:
      MYSQL_DATABASE: test
      MYSQL_PASSWORD: root
      MYSQL_ROOT_PASSWORD: root
    ports:
      - '3306:3306'

  web:
    build:
      context: ./project/backend
      dockerfile: Dockerfile
    ports:
      - "8880:8880"
    environment:
      MYSQL_DATABASE: test
      MYSQL_USER: root
      MYSQL_PASSWORD: root
      MYSQL_HOST: mysql
    depends_on:
```

```
      - mysql

    restart: always
```

However, depending on the size of our MySQL database and the resources available on our machine, there might be some delay during the initialization of the MySQL service. To address this, we can add a health check to the web service to wait until the MySQL service is ready before starting your app.

```
services:
  mysql:
    build: ./project/backend/db
    container_name: some-mysql
    environment:
      MYSQL_DATABASE: test
      MYSQL_PASSWORD: root
      MYSQL_ROOT_PASSWORD: root
    ports:
      - '3306:3306'
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", "-
proot"]
      interval: 5s
      timeout: 30s
      retries: 5

  web:
    build:
      context: ./project/backend
      dockerfile: Dockerfile
    ports:
      - "8880:8880"
    environment:
      MYSQL_DATABASE: test
      MYSQL_USER: root
      MYSQL_PASSWORD: root
      MYSQL_HOST: mysql
    depends_on:
      mysql:
        condition: service_healthy
    restart: always
```

Now, our backend application waits for the completion of the MySQL initialization process and health check before initiating its launch.



VIII/ Make docker orchestration using Kubernetes

1) Build a Persistent Volume Claim (PVC)

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
  labels:
    app: mysql
    tier: database
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

2) MySQL pod's deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  labels:
    app: mysql
    tier: database
spec:
  replicas: 1
  selector:
```

```
      matchLabels:
        app: mysql
        tier: database
    template:
      metadata:
        labels:
          app: mysql
          tier: database
      spec:
        containers:
        - name: mysql
          image: mysql:8.0
          env:
          - name: MYSQL_ROOT_PASSWORD
            value: "root"
          - name: MYSQL_DATABASE
            value: "test"
          ports:
          - containerPort: 3306
            name: mysql
          volumeMounts:
          - name: mysql-persistent-storage
            mountPath: /var/lib/mysql
        volumes:
        - name: mysql-persistent-storage
          persistentVolumeClaim:
            claimName: mysql-pvc
```

3) Create a service object that will permit other pods to access the MySQL database pod

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql
    tier: database
spec:
  ports:
    - port: 3306
      targetPort: 3306
  selector:
    app: mysql
    tier: database
  #type: ClusterIP
  clusterIP: None
```

4) Minikube

Minikube quickly sets up a local Kubernetes cluster on macOS, Linux, and Windows.

```
minikube start
```

```
Windows PowerShell
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> minikube start
W1230 22:25:47.096167   23036 main.go:291] Unable to resolve the current Docker CLI context "default": context "default": conte
exts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f\meta.json: The system cannot find the path specified
* minikube v1.32.0 sur Microsoft Windows 10 Home 10.0.19045.3803 Build 19045.3803
* Utilisation du pilote docker basé sur le profil existant
* Démarrage du noeud de plan de contrôle minikube dans le cluster minikube
* Extraction de l'image de base...
* docker "minikube" container est manquant, il va être recréé.
* Création de docker container (CPU=2, Memory=8100Mo) ...
* Préparation de Kubernetes v1.28.3 sur Docker 24.0.7...
  - Génération des certificats et des clés
  - Démarrage du plan de contrôle ...
  - Configuration des règles RBAC ...
* Configuration de bridge CNI (Container Networking Interface)...
* Vérification des composants Kubernetes...
  - Utilisation de l'image gcr.io/k8s-minikube/storage-provisioner:v5
* Modules activés: storage-provisioner, default-storageclass
* Terminé ! kubectl est maintenant configuré pour utiliser "minikube" cluster et espace de noms "default" par défaut.
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend>
```

5) Applying this MySQL deployment file (content: PVC, MySQL pod and Service)

```
kubectl apply -f mysql-deployment.yaml
```

```
Windows PowerShell
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> kubectl apply -f mysql-deployment.yaml
persistentvolumeclaim/mysql-pvc created
deployment.apps/mysql-deployment created
service/mysql-service created
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend>
```

6) Display pod and deployment

```
kubectl get deployment
```

```
kubectl get pod
```

```
Windows PowerShell
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment    0/1     1            0           17s
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> kubectl get pod
NAME                                READY   STATUS              RESTARTS   AGE
mysql-deployment-7bf6b759c6-jkvs4   0/1     ContainerCreating   0          21s
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend>
```

```
Windows PowerShell
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment    1/1     1            1           69s
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend> kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
mysql-deployment-7bf6b759c6-jkvs4   1/1     Running   0          71s
PS C:\Users\DONG\desktop\ece-devops-2023-fall-Dong-Lamine-George\project\backend>
```

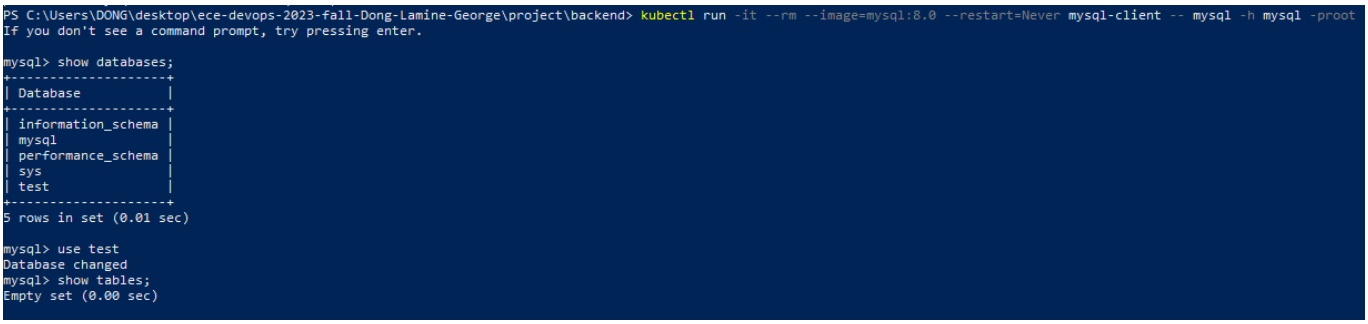
```
kubectl port-forward <POD NAME> 3308:3306
```

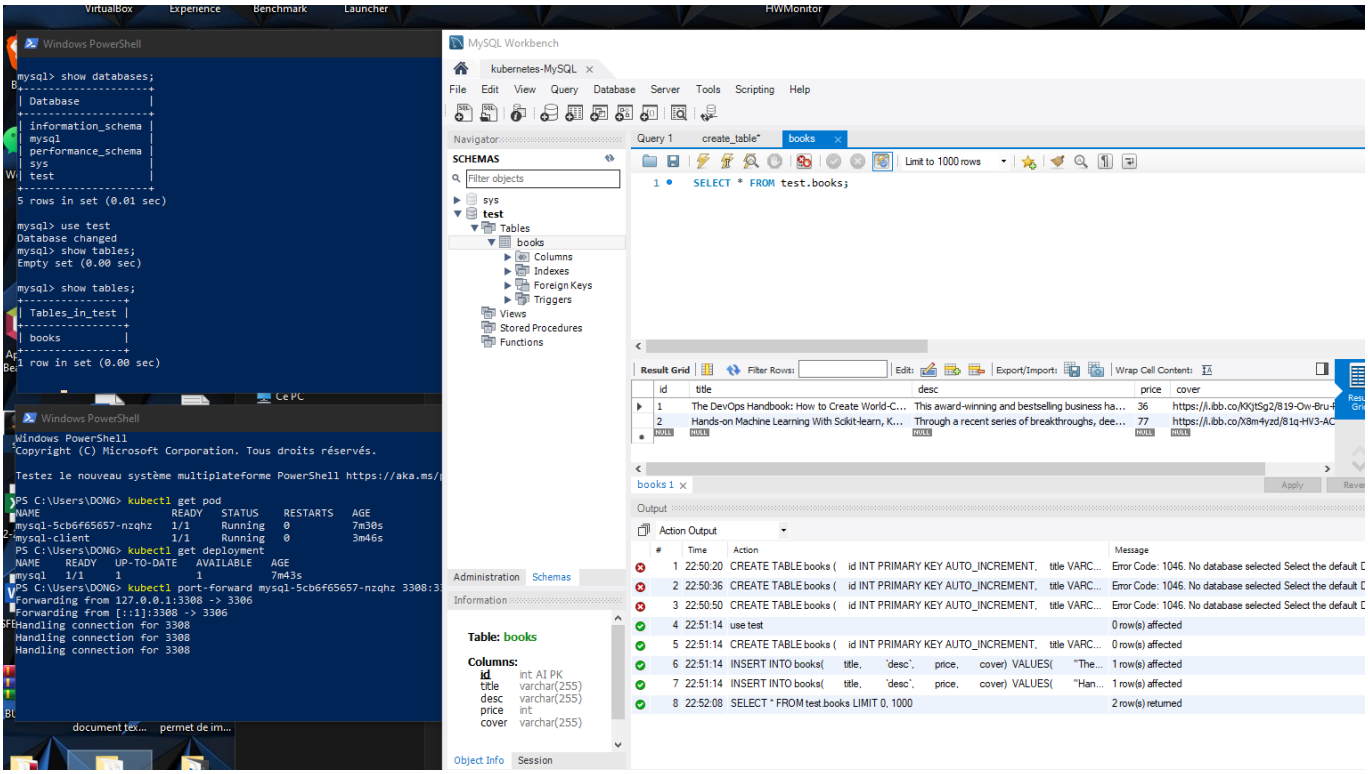
```
PS C:\Users\DONG> kubectl port-forward mysql-5cb6f65657-nzqhz 3308:3306
Forwarding from 127.0.0.1:3308 -> 3306
Forwarding from [::1]:3308 -> 3306
Handling connection for 3308
Handling connection for 3308
Handling connection for 3308
```

7) Check Database

```
kubectl run -it --rm --image=mysql:8.0 --restart=Never mysql-client -- <POD NAME>
-h mysql -proot
```

This command runs the MySQL container in an interactive mode, which allows you to execute commands at the time of running the container. A MySQL shell will open and you could create new databases, new tables, insert data to tables and do more SQL commands.





8) Create a deployment for our backend node js app

a) Use the prodived in "./project/backend" Dockerfile and build the image.

```
docker build -t backend-app .
```

b) Let’s create our Kubernetes deployment for our app.

kubectl delete service kubectl delete deployment $DEPLOYMENT_NAME

```
```

7) Create a deployment for our frontend react app

Make a service mesh using Istio

Implement Monitoring to your containerized application