

GANPAT UNIVERSITY

?

Name : Desai Yatrik

Enrollment No. : 22162101003

Branch : CBA

Batch : 51

Practical : 12

Subject : Microservices

Semester : 5

Pre-Requisites :

1. Docker should be installed in our system
2. Node js should be installed
3. Account on docker hub
4. Knowledge of docker commands
5. Mongodb should be installed

Practical 12 :

Develop a Node.js Express application that provides RESTful APIs to manage student data, allowing operations such as retrieving, adding, updating, and deleting student records. Finally build an image of your application and run it as a docker container. Also upload the image to Docker Hub.

Features :

- Retrieve a list of all students.
- Add a new student to the database.

- Retrieve a specific student by ID.
- Update an existing student's information by ID.

- Remove a student from the database by ID.

Data Structure: Each student record should include: id: Unique identifier (integer) , name: Student's name (string) , age: Student's age (integer)

Image name should be following format:

Your_docker_hub_username/Repository name.

Container name: Task_Practical_12

Attach the screenshots and pull command to pull your image from docker hub in the documentation

Screenshots :

1. Code for creating database Student_records and collection Students

```
JS database.js > [M] MongoClient
1  var MongoClient = require('mongodb').MongoClient;
2  var url = "mongodb://127.0.0.1:27017";
3
4  MongoClient.connect(url, function(err, client) {
5      if (err) {
6          console.error("Connection error:", err);
7          return;
8      }
9      console.log("Connected successfully to MongoDB server");
10
11     var dbo = client.db("Student_records");
12     dbo.createCollection("Students", function(err, res) {
13         if (err) throw err;
14         console.log("Collection created");
15         client.close();
16     });
17 });
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

To address all issues, run:
npm audit fix

Run 'npm audit' for details.

PS D:\Semester 5\Microservices\Practicals\Experiment - 12> node database.js

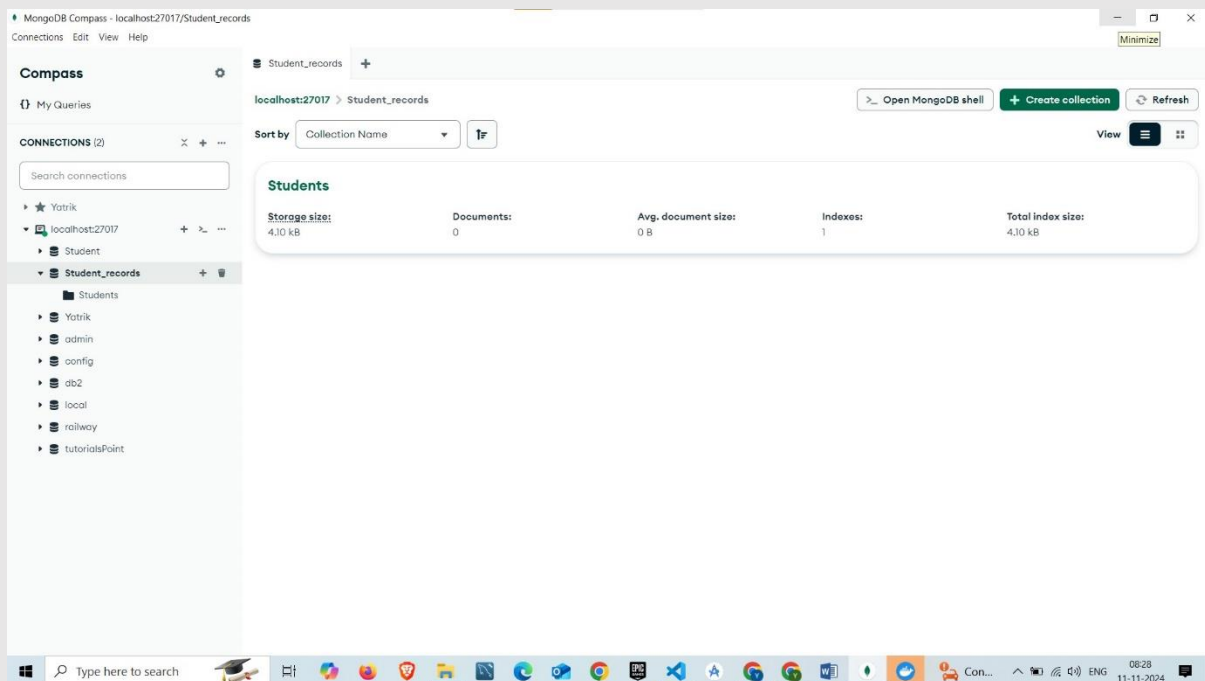
(node:1516) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor. (Use 'node --trace-warnings ...' to show where the warning was created)

Connected successfully to MongoDB server

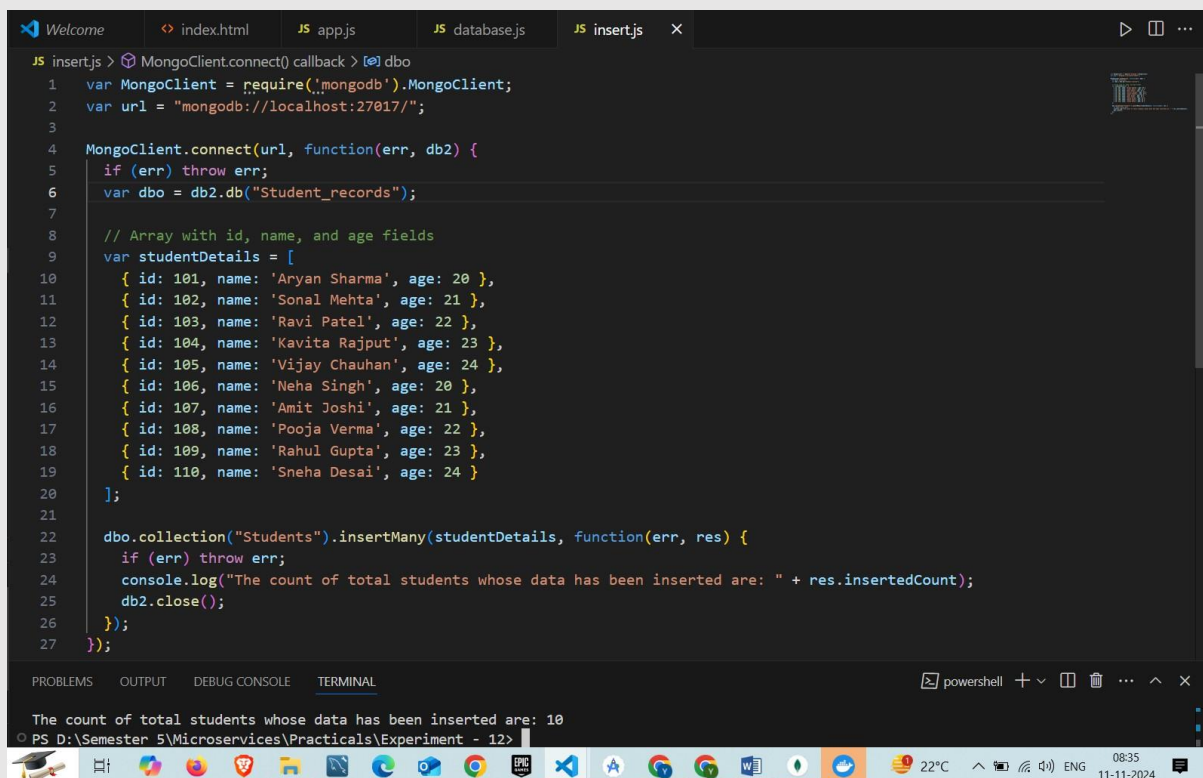
Collection created

PS D:\Semester 5\Microservices\Practicals\Experiment - 12>

2. Successfully created the collection and database



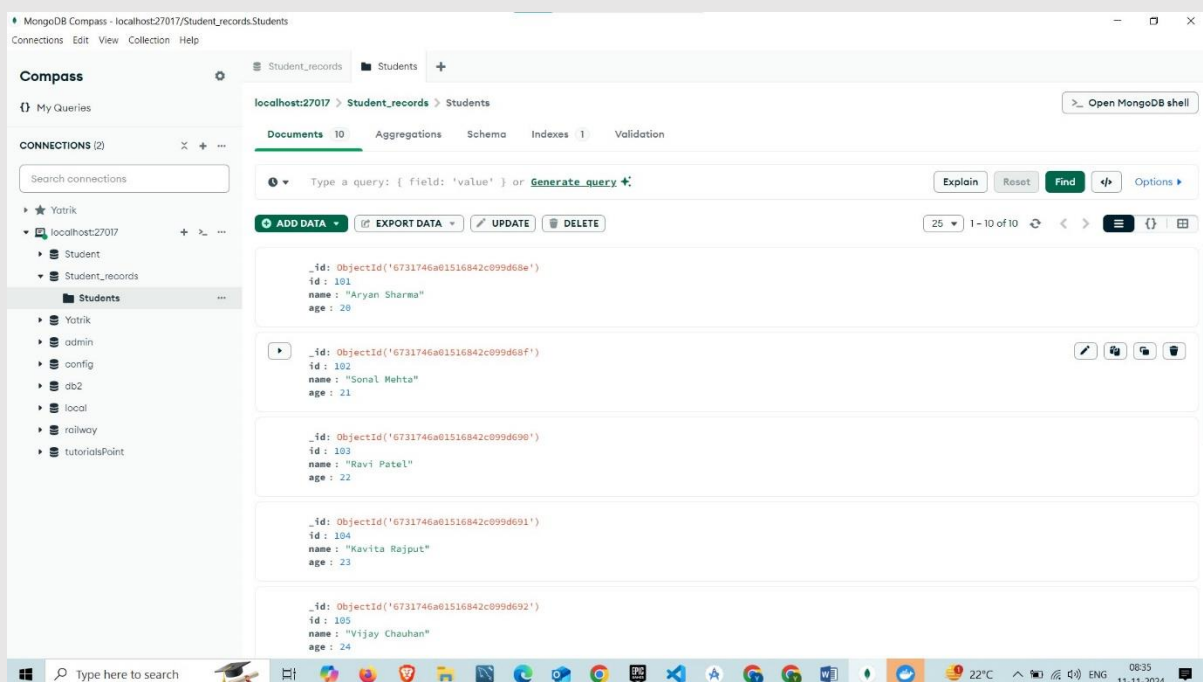
3. Inserting student records into collection Students



```
JS insert.js > MongoClient.connect() callback > [0] dbo
1 var MongoClient = require("mongodb").MongoClient;
2 var url = "mongodb://localhost:27017/";
3
4 MongoClient.connect(url, function(err, db2) {
5   if (err) throw err;
6   var dbo = db2.db("Student_records");
7
8   // Array with id, name, and age fields
9   var studentDetails = [
10    { id: 101, name: 'Aryan Sharma', age: 20 },
11    { id: 102, name: 'Sonal Mehta', age: 21 },
12    { id: 103, name: 'Ravi Patel', age: 22 },
13    { id: 104, name: 'Kavita Rajput', age: 23 },
14    { id: 105, name: 'Vijay Chauhan', age: 24 },
15    { id: 106, name: 'Neha Singh', age: 20 },
16    { id: 107, name: 'Amit Joshi', age: 21 },
17    { id: 108, name: 'Pooja Verma', age: 22 },
18    { id: 109, name: 'Rahul Gupta', age: 23 },
19    { id: 110, name: 'Sneha Desai', age: 24 }
20  ];
21
22  dbo.collection("Students").insertMany(studentDetails, function(err, res) {
23    if (err) throw err;
24    console.log("The count of total students whose data has been inserted are: " + res.insertedCount);
25    db2.close();
26  });
27 });
```

The count of total students whose data has been inserted are: 10

4. Successfully inserted 10 records into collection



5. Code to simply connect with database and providing schema

```
JS app.js > ...
6 // Connect to MongoDB
7 mongoose.connect('mongodb://localhost:27017/Student_records', {
8   useNewUrlParser: true,
9   useUnifiedTopology: true
10 });
11
12 // Create a schema for the student
13 const studentSchema = new mongoose.Schema({
14   id: String,
15   name: String,
16   age: Number
17 });
18
19 // Use the correct collection name (Mongoose will use "Students" collection)
20 const Student = mongoose.model('Student', studentSchema, 'Students'); // 'Students' is the collection name
21
22 // Middleware
23 app.use(bodyParser.json());
24 app.use(express.static('public'));
25
```

6. Code for retrieving all students

```
28 // Get all students
29 app.get('/students', async (req, res) => {
30   try {
31     const students = await Student.find();
32     res.json(students);
33   } catch (err) {
34     res.status(500).send(err);
35   }
36 });
37
```

7. Code for adding new student into database

```
38 // Add a new student
39 app.post('/students', async (req, res) => {
40   const newStudent = new Student(req.body);
41   try {
42     await newStudent.save();
43     res.status(201).send(newStudent);
44   } catch (err) {
45     res.status(400).send(err);
46   }
47 });
48
```

8. Code for retrieving student by specific id

```
50 app.get('/students/:id', async (req, res) => {
51   try {
52     const student = await Student.findOne({ id: req.params.id }); // Make sure to use 'id' field, not '_id'
53     if (!student) {
54       return res.status(404).send('Student not found');
55     }
56     res.json(student);
57   } catch (err) {
58     res.status(500).send(err);
59   }
60 });
```

9. Code for updating information of students

```
63 app.put('/students/:id', async (req, res) => {
64   try {
65     const student = await Student.findOneAndUpdate(
66       { id: req.params.id },
67       req.body,
68       { new: true }
69     );
70     if (!student) {
71       return res.status(404).send('Student not found');
72     }
73     res.json(student);
74   } catch (err) {
75     res.status(500).send(err);
76   }
77 });
```

10. Code for removing student based on specific id

```
80 ✓ app.delete('/students/:id', async (req, res) => {
81   ✓ try {
82     const student = await Student.findOneAndDelete({ id: req.params.id });
83   ✓ if (!student) {
84     |   return res.status(404).send('Student not found');
85   }
86   res.status(204).send();
87   ✓ } catch (err) {
88     res.status(500).send(err);
89   }
90 });
```


11. Dockerfile for our node js application

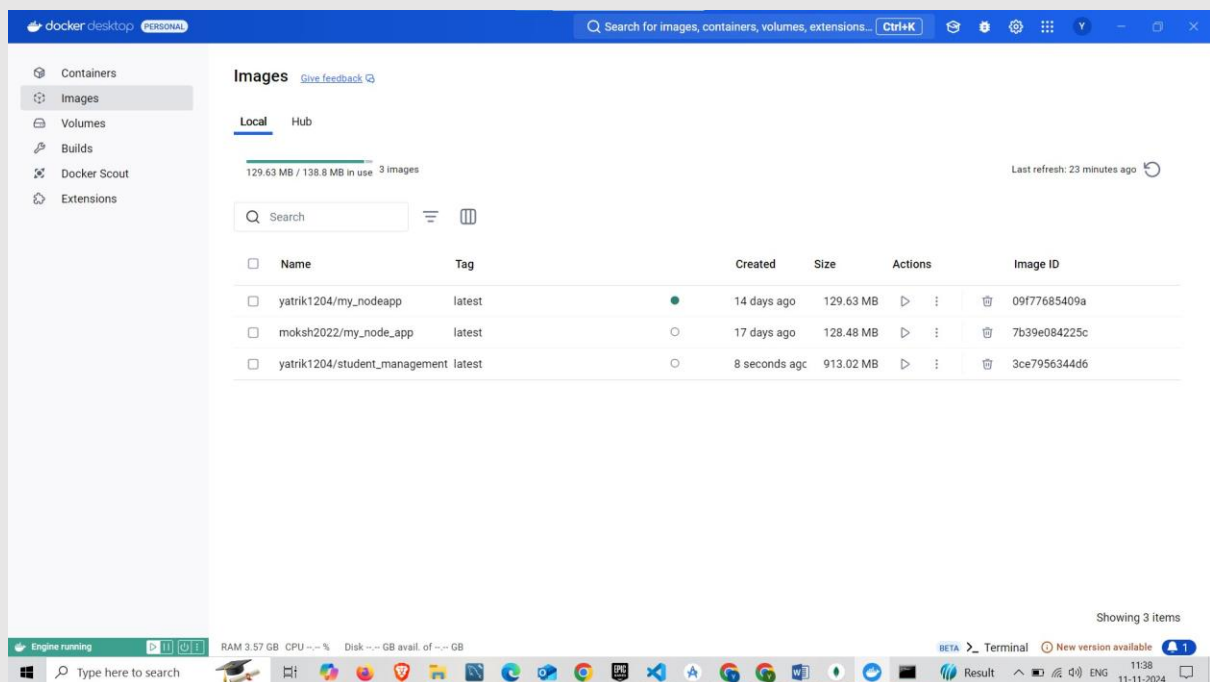
```
1 # Use official Node.js image
2 FROM node:14
3
4 # Create and set the working directory
5 WORKDIR /usr/src/app
6
7 # Copy package.json and install dependencies
8 COPY package*.json ./
9 RUN npm install
10
11 # Copy the rest of the application code
12 COPY . .
13
14 # Expose port 3000
15 EXPOSE 3000
16
17 # Command to run the app
18 CMD ["node", "app.js"]
19
```

12. Command : docker build -t yatrik1204/student_management .

=> Building an image yatrik1204/student_management

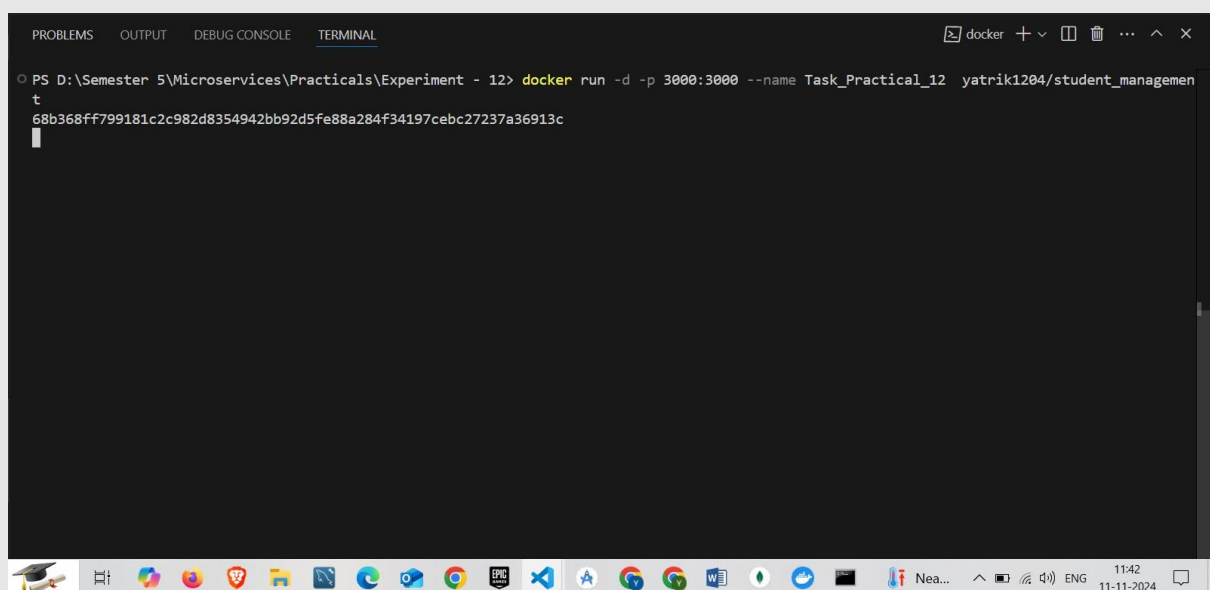
```
PS D:\Semester 5\Microservices\Practicals\Experiment - 12> docker build -t yatrik1204/student_management .
[+] Building 0.0s (0/0) docker:desktop-linux
[+] Building 1248.1s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 375B
=> [internal] load metadata for docker.io/library/node:14
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
=> => resolve docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
=> => sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa 776B / 776B
=> => sha256:2cafa3fbb0b6529ee4726b4f599ec27ee557ea3dea7019182323b3779959927f 2.21kB / 2.21kB
=> => sha256:1d12470fa662a2a5cb50378dc8c8ea228c1735747db410bbefb8e2d9144b5452 7.51kB / 7.51kB
=> => sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408 50.45MB / 50.45MB
=> => sha256:3d2201bd995ccccf12851a50820de03d34a17011dcbb9ac9fd3a50c952cbb131 10.00MB / 10.00MB
=> => sha256:b253aefaeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5 7.86MB / 7.86MB
=> => sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2 51.88MB / 51.88MB
=> => sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fd8ba8e23d1b1569 191.85MB / 191.85MB
=> => sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8 4.19kB / 4.19kB
=> => extracting sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408
=> => sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590 35.24MB / 35.24MB
=> => extracting sha256:b253aefaeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5
=> => extracting sha256:3d2201bd995ccccf12851a50820de03d34a17011dcbb9ac9fd3a50c952cbb131
=> => sha256:0c8cc2f24a4dc64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3 2.29MB / 2.29MB
```

13. Successfully created an image on docker

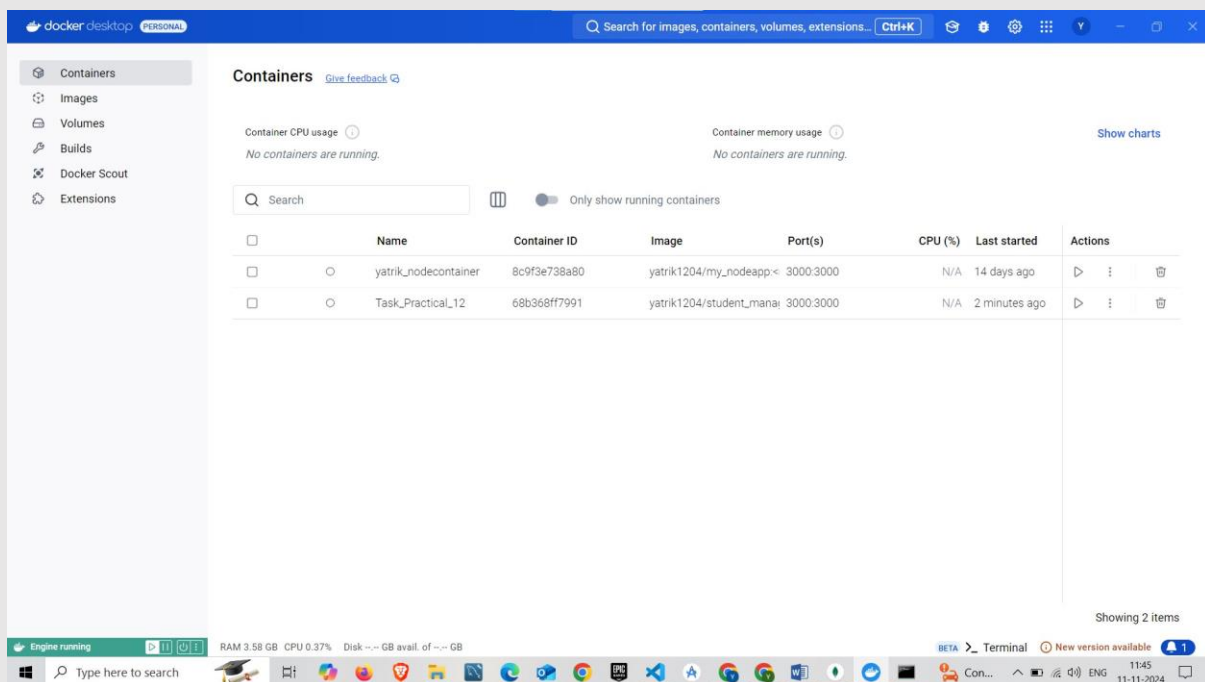


14. Command : `docker run -d -p 3000:3000 --name Task_Practical_12 yatrik1204/student_management`

=> it will create a container named Task_Practical_12 from image yatrik1204/student_management



15. Successfully Created Task_Practical_12 Container on Docker



16. Command : docker push yatrik1204/student_management

=> it will push the image yatrik1204/student_management

