Task 5.1P

1. **Docker Installation**
   - Ensure and check if Docker is installed:
   - *docker –version*

2. **Sample Web Application**
   - I chose the application that I was working on during week 4, the calculator application
   - *https://github.com/mustafaT96/sit737-2025-prac4p.git*

3. **Create a Dockerfile**
   - A **Dockerfile** defines the environment and steps needed to run your application inside a container.
   - I created the **Dockerfile** in the root directory of the cloned project as shown below:

server.js | Dockerfile | docker-compose.yml

Dockerfile > ...

```
# Step 1: Use an official Node.js runtime as the base image
FROM node:14

# Step 2: Set the working directory inside the container
WORKDIR /usr/src/app

# Step 3: Copy the package.json and package-lock.json (if available)
COPY package*.json ./

# Step 4: Install dependencies inside the container
RUN npm install

# Step 5: Copy the rest of the application's code into the container
COPY . .

# Step 6: Expose the port your app runs on (matching the port in your app, e.g., 3000)
EXPOSE 3000

# Step 7: Define the command to run your app (start the server)
CMD ["node", "server.js"]
```

4. **Build the Docker Image**
   - Created the Docker image using the *Dockerfile:*
   - *docker build -t calculator-microservice .*
   - This command will create a Docker image tagged as *calculator-microservice* from the current directory (.).

5. **Create a Docker Compose File**
   - The *docker-compose.yml* file defines the services (containers) required for the application.

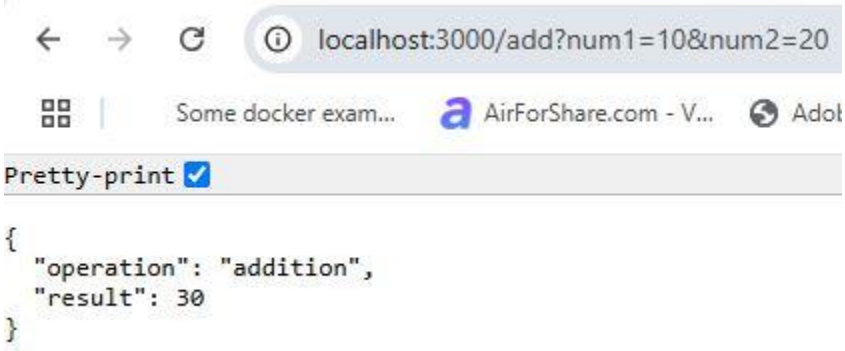
```yaml
version: '3.8'

services:
  calculator:
    build: .
    ports:
      - "3000:3000"
    container_name: calculator-microservice
    restart: always
    networks:
      - app-network
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:3000/health"]
      interval: 30s
      retries: 3
      start_period: 10s
      timeout: 10s

networks:
  app-network:
    driver: bridge
```

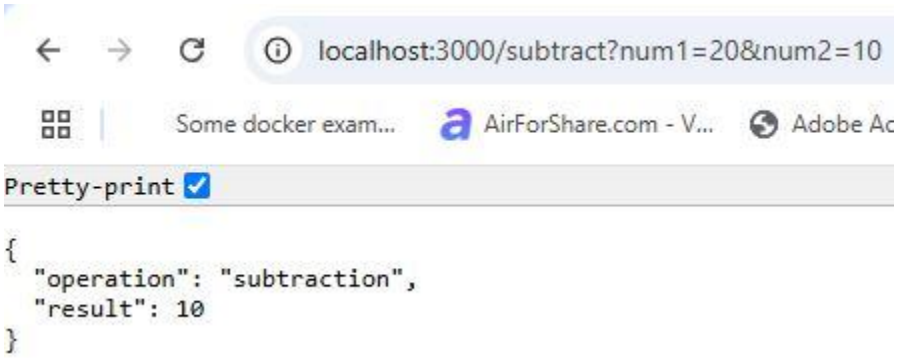
   - This defines a single service (*calculator*) and includes the *healthcheck* block.

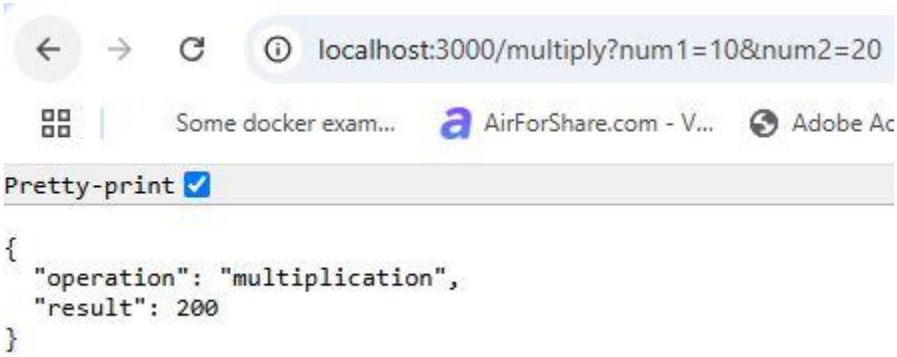
6. **Start the Docker Compose Environment**
   - Start all the containers defined in the *docker-compose.yml* file:
   - *docker-compose up –d*
   - The *-d* flag runs the containers in detached mode, so they run in the background.

7. **Test the Application**

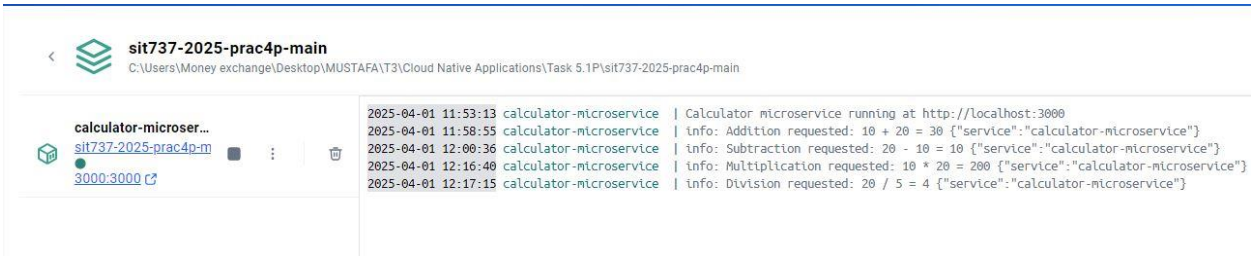
- To test if the application is running, open a browser to access the web application:





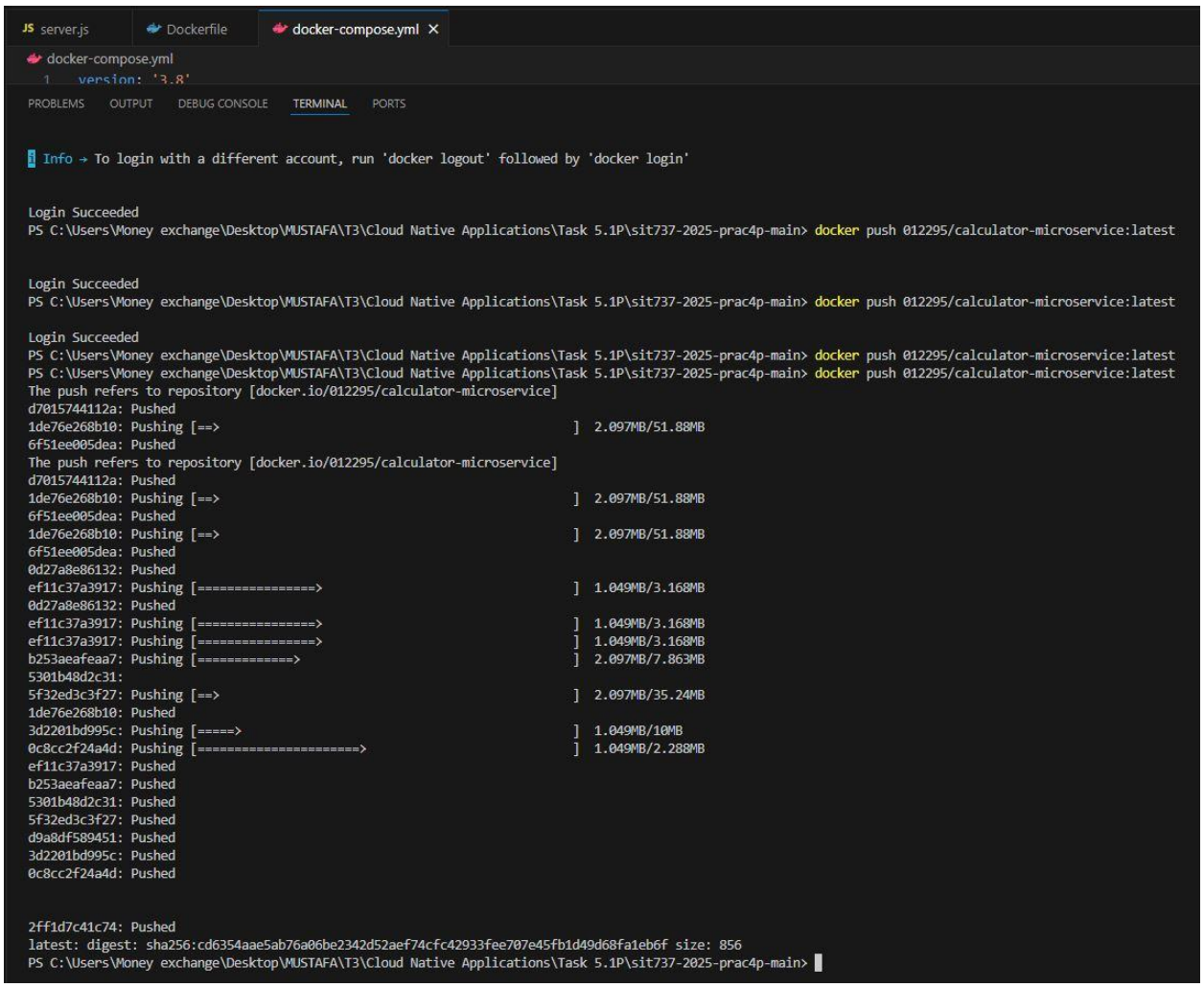


- **Container logs**



8. Push the Docker Image to a Registry

- Tag the image to push it to a Docker registry
- *docker push 012295/calculator-microservice:latest*



Part II: Health Check Implementation

1. **Add a Health Endpoint**
   - In the *server.js* file, add a */health* route to return a healthy status:

     ```
     app.get('/health', (req, res) => {
       res.status(200).send('OK');
     })
     ```

   - This endpoint responds with *200 OK* when the application is healthy.

2. **Modify Docker Compose File to Include Health Check**
   - Update the *docker-compose.yml* file to include the health check:

     ```
     healthcheck:
       test: ["CMD", "curl", "-f", "http://localhost:3000/health"]
       interval: 30s
       retries: 3
       start_period: 10s
       timeout: 10s
     ```

   - This configures the health check to call the */health* endpoint every 30 seconds, retrying up to 3 times if it fails.

3. **Start or Restart the Docker Compose Environment**
   - Restart containers to apply the changes:

     ```
     docker-compose down
     docker-compose up –d
     ```

4. **Testing the health check**
   - Use the following command to check the health status of the container:
     docker inspect --format='{{json .State.Health}}' calculator-microservice

   - To test the health check, stop the container:
     docker stop calculator-microservice

   - After stopping, check the health status again:
     docker inspect --format='{{json .State.Health}}' calculator-microservice
   - The health status will now show as "unhealthy"

- Restart the container:
  docker start calculator-microservice

- After restarting, check the health status once more:
  docker inspect --format='{{json .State.Health}}' calculator-microservice
- It should show the container as "healthy"