**Name: ROTHA Dapravith**

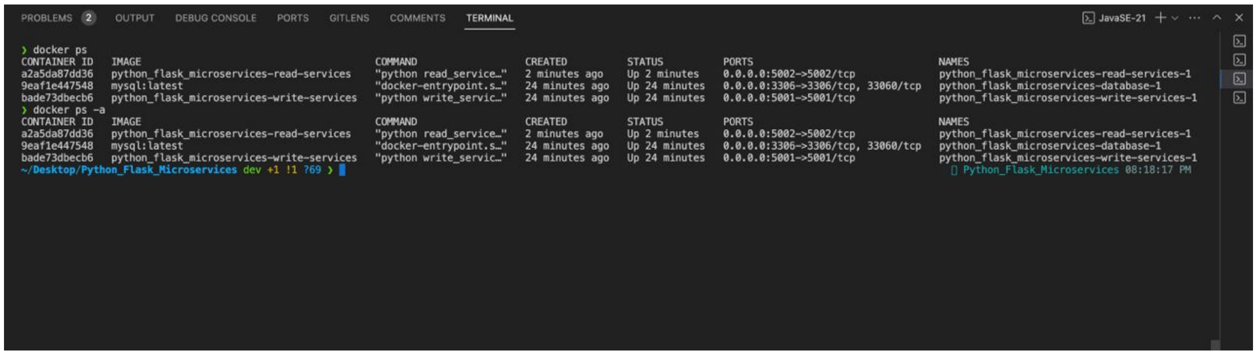
**ID: e20190915**

**Group: I5-GIC(B)**

# Final Lab Submission

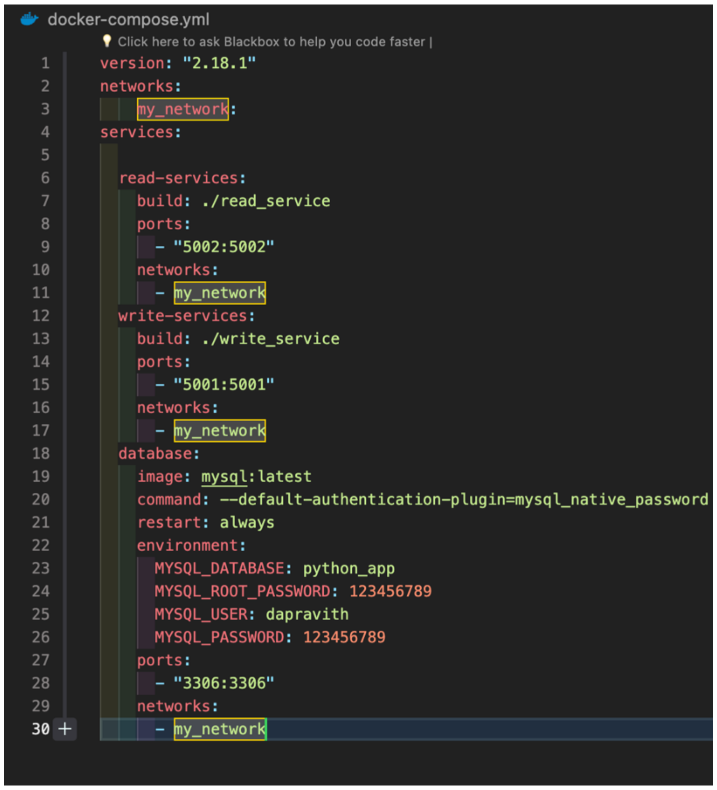
**Objective :** building and deploying a python microservices with docker container, docker compose and testing APIs only two methods use Read (GET HTTP method) and Write service (POST HTTP method).

- **docker ps and docker ps -a**



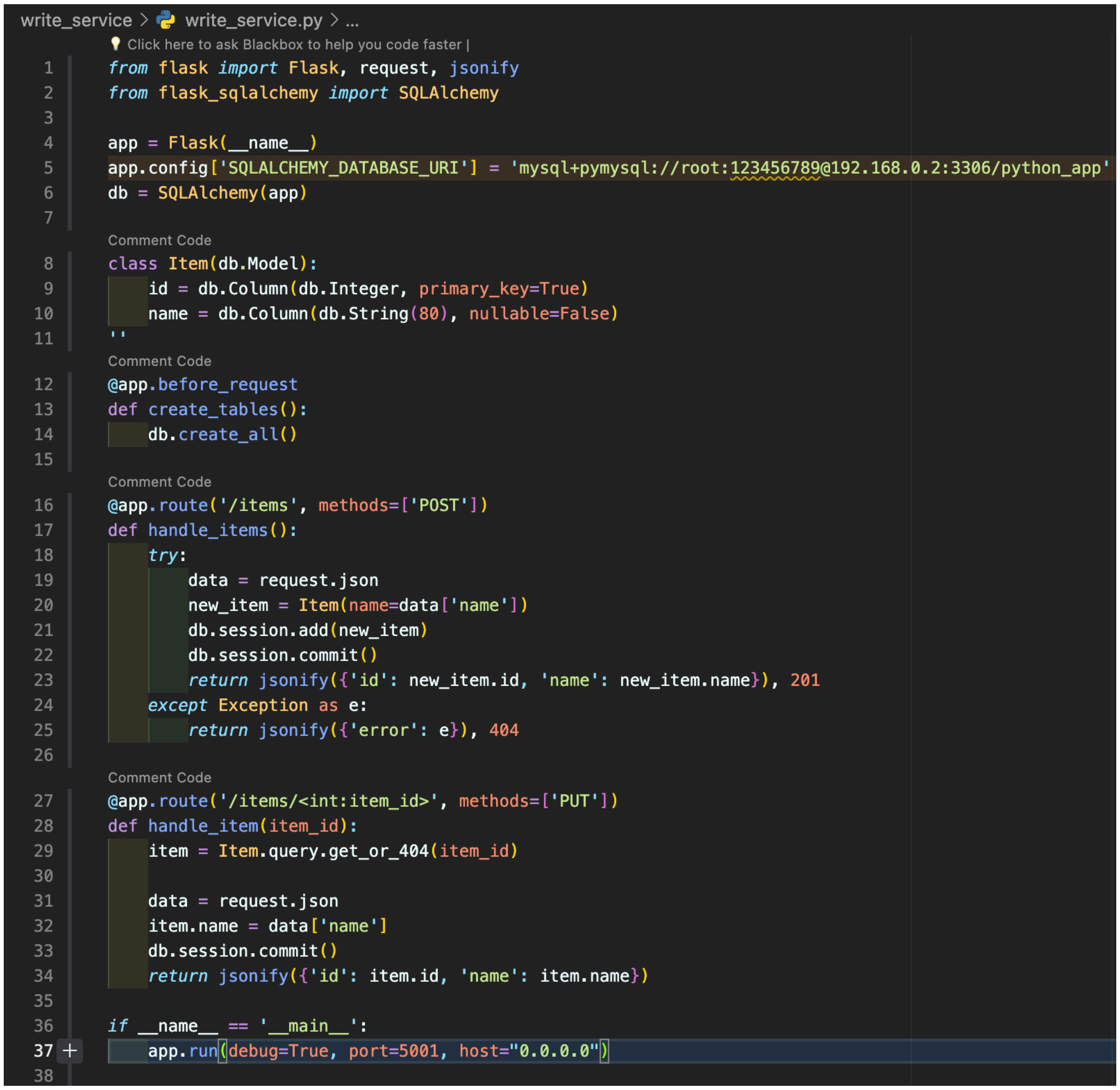
```
❯ docker ps
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS          PORTS                               NAMES
a2a5da87dd36   python_flask_microservices-read-services    "python read_service…"   2 minutes ago    Up 2 minutes    0.0.0.0:5002->5002/tcp              python_flask_microservices-read-services-1
9eaf1e447548   mysql:latest                                 "docker-entrypoint.s…"   24 minutes ago   Up 24 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp   python_flask_microservices-database-1
bade73dbecb6   python_flask_microservices-write-services   "python write_servic…"   24 minutes ago   Up 24 minutes   0.0.0.0:5001->5001/tcp              python_flask_microservices-write-services-1
❯ docker ps -a
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS          PORTS                               NAMES
a2a5da87dd36   python_flask_microservices-read-services    "python read_service…"   2 minutes ago    Up 2 minutes    0.0.0.0:5002->5002/tcp              python_flask_microservices-read-services-1
9eaf1e447548   mysql:latest                                 "docker-entrypoint.s…"   24 minutes ago   Up 24 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp   python_flask_microservices-database-1
bade73dbecb6   python_flask_microservices-write-services   "python write_servic…"   24 minutes ago   Up 24 minutes   0.0.0.0:5001->5001/tcp              python_flask_microservices-write-services-1
~/Desktop/Python_Flask_Microservices dev +1 !1 ?69 ❯                                                                                             Python_Flask_Microservices 08:18:17 PM
```

- **docker-compose.yml file**



```yaml
version: "2.18.1"
networks:
    my_network:
services:

  read-services:
    build: ./read_service
    ports:
      - "5002:5002"
    networks:
      - my_network
  write-services:
    build: ./write_service
    ports:
      - "5001:5001"
    networks:
      - my_network
  database:
    image: mysql:latest
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_DATABASE: python_app
      MYSQL_ROOT_PASSWORD: 123456789
      MYSQL_USER: dapravith
      MYSQL_PASSWORD: 123456789
    ports:
      - "3306:3306"
    networks:
      - my_network
```

## - write_service.py code

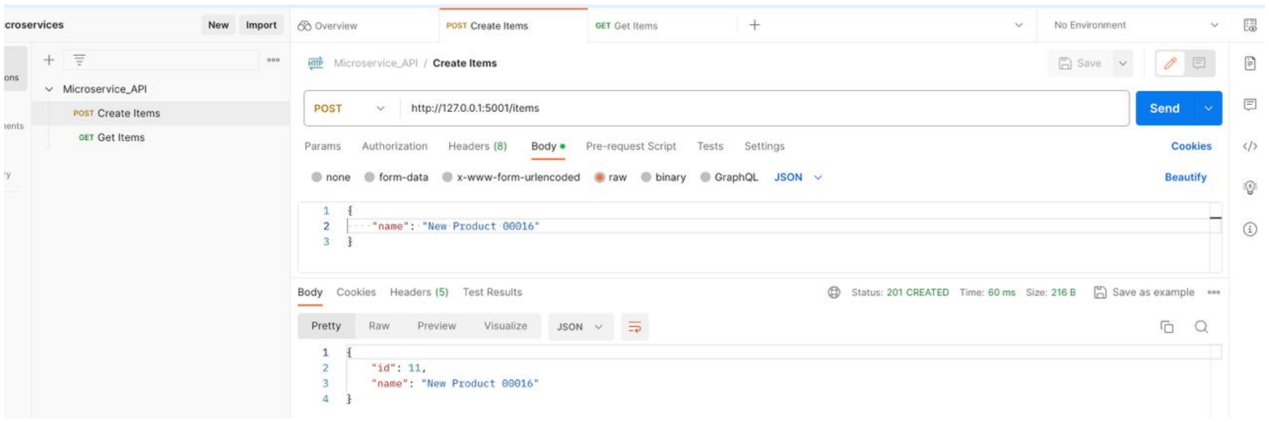


write_service > write_service.py > ...

💡 Click here to ask Blackbox to help you code faster |

```python
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:123456789@192.168.0.2:3306/python_app'
db = SQLAlchemy(app)

Comment Code
class Item(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)
''
Comment Code
@app.before_request
def create_tables():
    db.create_all()

Comment Code
@app.route('/items', methods=['POST'])
def handle_items():
    try:
        data = request.json
        new_item = Item(name=data['name'])
        db.session.add(new_item)
        db.session.commit()
        return jsonify({'id': new_item.id, 'name': new_item.name}), 201
    except Exception as e:
        return jsonify({'error': e}), 404

Comment Code
@app.route('/items/<int:item_id>', methods=['PUT'])
def handle_item(item_id):
    item = Item.query.get_or_404(item_id)

    data = request.json
    item.name = data['name']
    db.session.commit()
    return jsonify({'id': item.id, 'name': item.name})

if __name__ == '__main__':
    app.run(debug=True, port=5001, host="0.0.0.0")

```

## - API write_service (POST HTTP method)



POST http://127.0.0.1:5001/items

Body: none, form-data, x-www-form-urlencoded, [✓ raw], binary, GraphQL — JSON

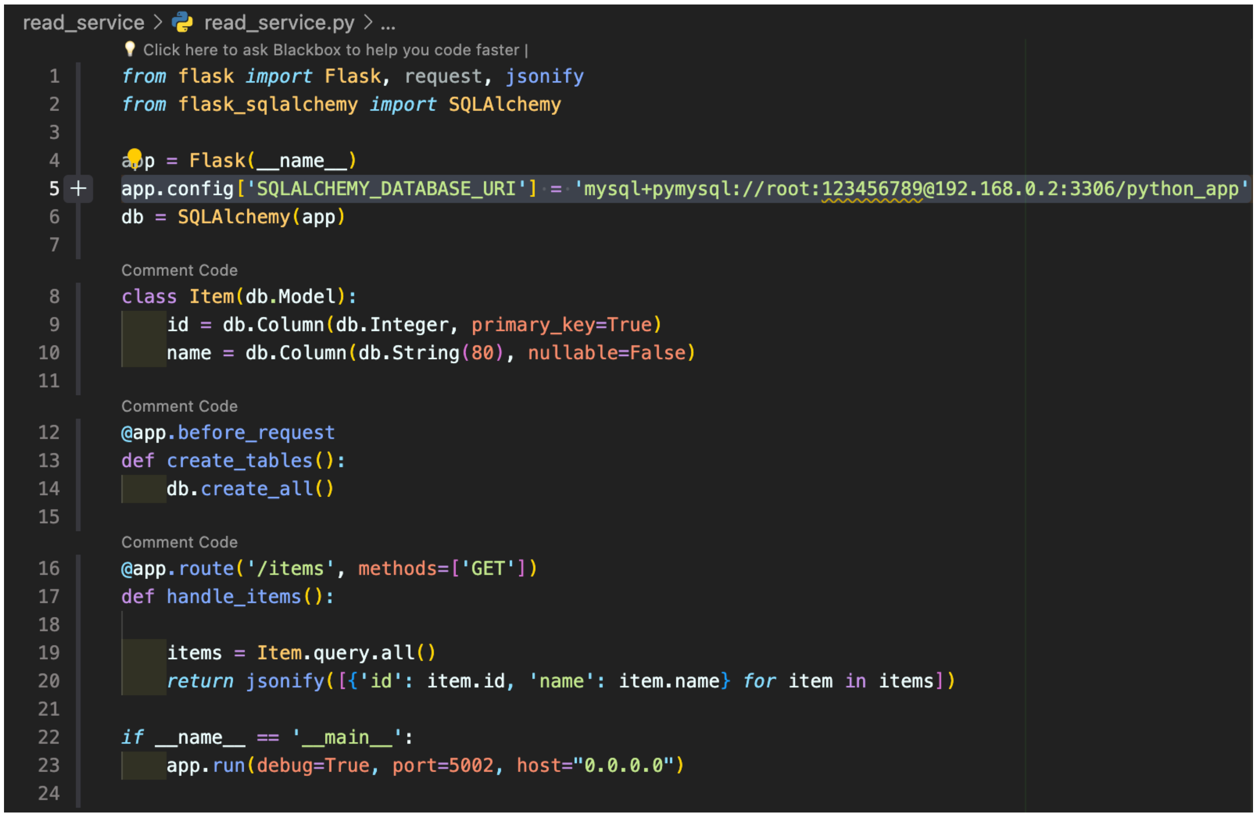
```json
{
    "name": "New Product 00016"
}
```

Status: 201 CREATED Time: 60 ms Size: 216 B

Body — Pretty — JSON

```json
{
    "id": 11,
    "name": "New Product 00016"
}
```

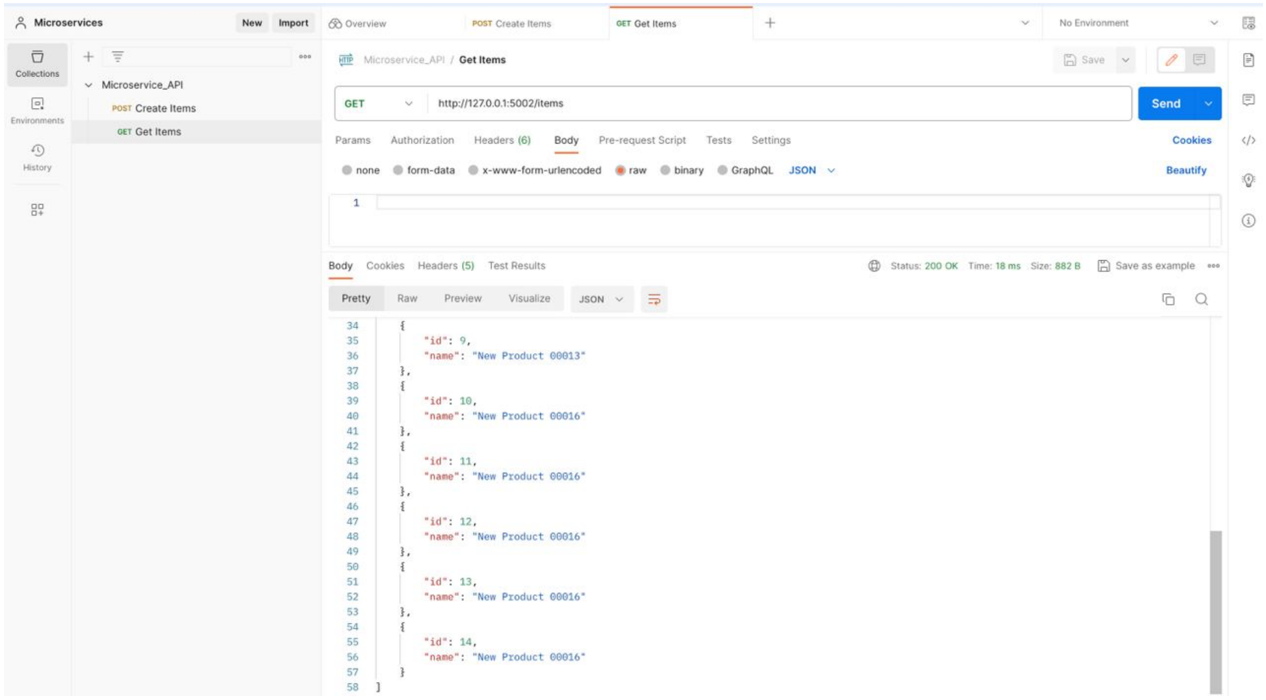
## - read_service.py code



read_service > read_service.py > ...

```python
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:123456789@192.168.0.2:3306/python_app'
db = SQLAlchemy(app)

class Item(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)

@app.before_request
def create_tables():
    db.create_all()

@app.route('/items', methods=['GET'])
def handle_items():

    items = Item.query.all()
    return jsonify([{'id': item.id, 'name': item.name} for item in items])

if __name__ == '__main__':
    app.run(debug=True, port=5002, host="0.0.0.0")
```

## - API read_service GET HTTP method)



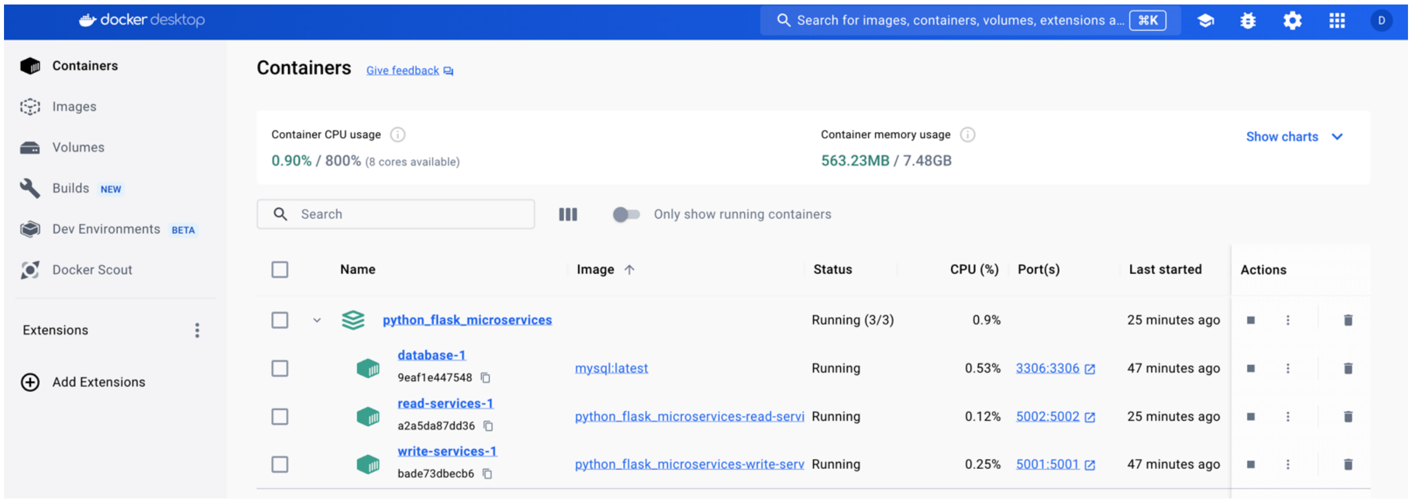
Microservices / Microservice_API / Get Items

GET http://127.0.0.1:5002/items

Status: 200 OK Time: 18 ms Size: 882 B

```
    {
        "id": 9,
        "name": "New Product 00013"
    },
    {
        "id": 10,
        "name": "New Product 00016"
    },
    {
        "id": 11,
        "name": "New Product 00016"
    },
    {
        "id": 12,
        "name": "New Product 00016"
    },
    {
        "id": 13,
        "name": "New Product 00016"
    },
    {
        "id": 14,
        "name": "New Product 00016"
    }
]
```

- **Docker desktop**



Containers

Container CPU usage: 0.90% / 800% (8 cores available)

Container memory usage: 563.23MB / 7.48GB

| Name | Image | Status | CPU (%) | Port(s) | Last started | Actions |
|---|---|---|---|---|---|---|
| python_flask_microservices | | Running (3/3) | 0.9% | | 25 minutes ago | |
| database-1 9eaf1e447548 | mysql:latest | Running | 0.53% | 3306:3306 | 47 minutes ago | |
| read-services-1 a2a5da87dd36 | python_flask_microservices-read-servi | Running | 0.12% | 5002:5002 | 25 minutes ago | |
| write-services-1 bade73dbecb6 | python_flask_microservices-write-serv | Running | 0.25% | 5001:5001 | 47 minutes ago | |

- Docker command essential use to build microservices includes:
  - **docker network list :** list all network in docker environment.
  - **docker network inspect network_name:** displays detailed information about a specific docker network.
  - **docker-compose up -d** : start and runs entire applications build in docker-compose.yml file.
  - **docker-compose up –build -d** : use for rebuild of the image before start containers.