

Mini-Corso introduttivo Flask + PostgreSQL con Docker su WSL



1. Prerequisiti

Assicurati di avere installato su WSL (Ubuntu):


```
sudo apt update
sudo apt install python3 python3-pip
# Docker + Docker Compose
```

Se non li hai:

```
sudo apt install docker.io docker-compose
sudo service docker start
```

Aggiungiti al gruppo docker per evitare sudo ogni volta:

```
sudo usermod -aG docker $USER
```

 Poi esci e rientra nella sessione WSL.

2. Struttura del progetto

All'interno della cartella **my-container-server**, crea:

```
my-container-server/
├── app.py
├── requirements.txt
├── Dockerfile
└── docker-compose.yml
```

3. Codice Python (Flask)

◆ app.py

```
from flask import Flask, jsonify
import psycopg2

app = Flask(__name__)

@app.route("/items")
def get_items():
    conn = psycopg2.connect(
        dbname="mydb",
        user="giacomo",
        password="giacomo_password",
        host="db" # nome del servizio DB nel docker-compose
    )
```

```
cur = conn.cursor()
cur.execute("SELECT id, name FROM items;")
items = cur.fetchall()
cur.close()
conn.close()
return jsonify([{"id": i[0], "name": i[1]} for i in items])

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=3000)
```

4. 📦 Dipendenze Python

◆ requirements.txt

```
flask
psycopg2-binary
```

- ◆ Usiamo psycopg2-binary perché è più facile da installare in Docker.

5. 🐳 Dockerfile per il backend

◆ Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["python3", "app.py"]
```

6. ⚙️ Docker Compose per orchestrare tutto

◆ docker-compose.yml

```
version: '3.8'

services:
  db:
    image: postgres:15
    container_name: my-container-server-db-1
    environment:
      POSTGRES_DB: mydb
      POSTGRES_USER: giacomino
      POSTGRES_PASSWORD: giacomino_password
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data

  web:
    build: .
    container_name: my-container-server-web-1
    ports:
```

```
- "3000:3000"
depends_on:
  - db

volumes:
  pgdata:
```

7. 🚀 Avvio del progetto

Da dentro la cartella:

```
sudo docker-compose up --build
```

Vedrai il log del DB e di Flask.

L'app sarà raggiungibile da: <http://localhost:3000/items>

8. 🗄️ Creare container e tabella nel DB

Avvia tutto:

```
sudo docker-compose up --build
```

Dettaglio configurazione:

- image: postgres:15 → scarica l'immagine ufficiale PostgreSQL v15
- container_name → nome leggibile del container
- environment → variabili di configurazione:
 - POSTGRES_DB: mydb
 - POSTGRES_USER: giacomo
 - POSTGRES_PASSWORD: giacomo_password
- ports → mappa la porta 5432 del container
- volumes → persistono i dati anche se il container viene eliminato

Apri una shell nel container PostgreSQL:

```
sudo docker exec -it my-container-server-db-1 psql -U giacomo -d mydb
```

Crea la tabella:

```
CREATE TABLE items (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL
);
```

Verifica:

```
\d items
```

9. 📁 Inserire dati nel DB

Inserisci 10.000 righe di test:

```
INSERT INTO items (name)
SELECT 'Michael is the best!'
FROM generate_series(1, 10000);
```

(È un mio inside joke! 😊)

Verifica:

```
SELECT COUNT(*) FROM items;
SELECT COUNT(*) FROM items WHERE name = 'Michael is the best!';
```

10. 🔄 Test dell'API

Fuori da psql, prova:

```
curl http://localhost:3000/items
```

Oppure apri in browser.

Dovresti vedere una lista JSON dei record.

✅ Risultato

Hai ottenuto:

- Un'app **Flask** in un container Docker
 - Un DB **PostgreSQL** in un secondo container
 - Comunicazione tra i due via **Docker Compose**
 - Un'API /items che legge dal database
 - 10.000 record reali inseriti nel DB
-

🗑️ Rimuovere i containers

1. Rimuovere tutto

```
sudo docker-compose down
sudo docker volume rm my-container-server_pgdata
sudo docker volume ls
```

2. Rimuovere le immagini (opzionale)

```
sudo docker images
sudo docker rmi my-container-server-web-1
```

3. Pulizia avanzata

```
sudo docker system prune --volumes
```

Vuoi che ti prepari direttamente un **file .docx ben formattato** (con titoli, sottotitoli e codice evidenziato) così puoi scaricarlo e usarlo subito?