

NAME: MAVIA ALAM KHAN(2303.KHI.DEG.017)

PAIRING WITH :Mohammad Hussam (2303.KHI.DEG.020)

&

AQSA TAUHEED(2303.KHI.DEG.011)

ASSIGNMENT NO :4.4

Browse to:

tasks/4_microservices_development/day_4_best_practices/
app_that_doesnt_follow_best_practices/

Analyze the application - which Microservice best practices does it not follow?

Think about what needs to be improved first. Have a look at the areas_for_improvement.txt file for hints.

Improve the application.

SOLUTION :

Here are the files :

Main.py:

```
import os
import logging
import psycopg2
from flask import Flask, render_template, request

app = Flask(__name__)
app.config['DEBUG'] = os.environ.get('FLASK_ENV') == 'development'
app.config['DB_HOST'] = os.environ.get('DB_HOST', 'db')
app.config['DB_PORT'] = int(os.environ.get('DB_PORT', 5432))
app.config['DB_NAME'] = os.environ.get('DB_NAME', 'your_database_name')
app.config['DB_USER'] = os.environ.get('DB_USER', 'your_username')
app.config['DB_PASSWORD'] = os.environ.get('DB_PASSWORD', 'your_password')

# Configure logging to output to the container output
root_logger = logging.getLogger()
root_logger.setLevel(logging.INFO)
stream_handler = logging.StreamHandler()
root_logger.addHandler(stream_handler)

def add_todo_item(item):
    conn = psycopg2.connect(
        host=app.config['DB_HOST'],
        port=app.config['DB_PORT'],
        dbname=app.config['DB_NAME'],
        user=app.config['DB_USER'],
        password=app.config['DB_PASSWORD']
    )
    c = conn.cursor()
    c.execute('INSERT INTO todo (content) VALUES (%s)', (item,))
    conn.commit()
    conn.close()

def get_todo_items():
    conn = psycopg2.connect(
        host=app.config['DB_HOST'],
        port=app.config['DB_PORT'],
        dbname=app.config['DB_NAME'],
        user=app.config['DB_USER'],
        password=app.config['DB_PASSWORD']
    )
    c = conn.cursor()
    c.execute('SELECT content FROM todo')
    todo_items = [item[0] for item in c.fetchall()]
    conn.close()
    return todo_items

@app.route("/", methods=["GET", "POST"])
def main():
    if request.method == "POST":
        content = request.form["content"]
        add_todo_item(content)

    todo_items = get_todo_items()

    return render_template("index.html", todo_items=todo_items)

if __name__ == "__main__":
    if app.config['DEBUG']:
        app.run(host="0.0.0.0", port=int(os.environ.get("PORT", 5000)), debug=True)
    else:
        app.run(host="0.0.0.0", port=int(os.environ.get("PORT", 5000)), debug=False)
```

Docker-compose.yml:

```
docker-compose.yml
1  version: '3'
2  services:
3    app:
4      build:
5        context: .
6        dockerfile: Dockerfile
7      ports:
8        - 5000:5000
9      environment:
10       - FLASK_ENV=development
11      depends_on:
12       - db
13      volumes:
14       - ./data:/app/data
15
16    db:
17      image: postgres:13
18      environment:
19       - POSTGRES_USER=your_username
20       - POSTGRES_PASSWORD=your_password
21       - POSTGRES_DB=your_database_name
22      volumes:
23       - ./pgdata:/var/lib/postgresql/data
24
```

Docker file :

```
Dockerfile > ...
1  # Use an official Python runtime as the base image
2  FROM python:3.8-slim-buster
3
4  # Set the working directory in the container
5  WORKDIR /app
6
7  # Copy the application code to the container
8  COPY . .
9
10 # Install the application dependencies at build time
11 RUN pip install --no-cache-dir -r requirements.txt
12
13 # Expose the application port
14 EXPOSE 5000
15
16 # Run the application
17 CMD ["python", "main.py"]
18
```

1- The logs shouldn't be written to a file, but to the container output.

For this in main.py we modified it as :

```
# Configure logging to output to the container
root_logger = logging.getLogger()
root_logger.setLevel(logging.INFO)
stream_handler = logging.StreamHandler()
root_logger.addHandler(stream_handler)
```

By configuring the logger with a StreamHandler and a custom log message format, the logs will be output to the container console instead of being written to a file.

2- It should be stateless, so that:

- it can easily be restarted without loss of data,
- it is easy to spawn multiple instances of the application

```
def add_todo_item(item):
    conn = psycopg2.connect(
        host=app.config['DB_HOST'],
        port=app.config['DB_PORT'],
        dbname=app.config['DB_NAME'],
        user=app.config['DB_USER'],
        password=app.config['DB_PASSWORD']
    )
    c = conn.cursor()
    c.execute('INSERT INTO todo (content) VALUES (%s)', (item,))
    conn.commit()
    conn.close()

def get_todo_items():
    conn = psycopg2.connect(
        host=app.config['DB_HOST'],
        port=app.config['DB_PORT'],
        dbname=app.config['DB_NAME'],
        user=app.config['DB_USER'],
        password=app.config['DB_PASSWORD']
    )
    c = conn.cursor()
    c.execute('SELECT content FROM todo')
    todo_items = [item[0] for item in c.fetchall()]
    conn.close()
    return todo_items
```

The application is stateless as it connects to a PostgreSQL database for data storage. This allows easy restarts without losing data and enables the spawning of multiple instances of the application

3 - Requirements installation should be moved from runtime to build time.

Requirements installation is moved to the build time in the Dockerfile.

```
9
10 # Install the application dependencies at build time
11 RUN pip install --no-cache-dir -r requirements.txt
```

4 -App should be able to be executed both during development, with debugging enabled, and in production, with debugging disabled.

if the DEBUG mode is enabled, the application runs with debug mode and the specified host and port. If the DEBUG mode is disabled, the application runs without debug mode and uses the specified host and port.

```
if __name__ == "__main__":
    if app.config['DEBUG']:
        app.run(host="0.0.0.0", port=int(os.environ.get("PORT", 5000)), debug=True)
    else:
        app.run(host="0.0.0.0", port=int(os.environ.get("PORT", 5000)), debug=False)
```

5 -The application should be built in such a way that the database can easily be replaced (development with production instance).

provided configuration for the PostgreSQL database in the Docker Compose file allows for easy replacement of the database between development and production instances.

```
db:
  image: postgres:13
  environment:
    - POSTGRES_USER=your_username
    - POSTGRES_PASSWORD=your_password
    - POSTGRES_DB=your_database_name
  volumes:
    - ./pgdata:/var/lib/postgresql/data
```

OUTPUT:

After setting all files , I used commnad docker compose build :

```
[sudo] password for mhussam:
mhussam@mhussam:~/Downloads/day_4_best_practices_tools_common_pitfalls/4.4$ docker compose build
[+] Building 45.3s (9/9) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 28B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 429B
=> [internal] load metadata for docker.io/library/python:3.8-slim-buster
=> [internal] load build context
=> => transferring context: 49.63MB
=> [1/4] FROM docker.io/library/python:3.8-slim-buster@sha256:89ad1c2cd09bda5bc85ada7eb93b5db57d32dc0105b7c942d272d68f376f67c3
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN pip install --no-cache-dir -r requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:c6587530ee1459e8a2abac8f67c785c0da1c758d0398263d8a0584b432ef85c1
=> => naming to docker.io/library/44-app
mhussam@mhussam:~/Downloads/day_4_best_practices_tools_common_pitfalls/4.4$
```

And now for running I used docker compose up :

```
mhussam@mhussam:~/Downloads/day_4_best_practices_tools_common_pitfalls/4.4$ docker compose up
[+] Running 3/3
 ✓ Network 44_default Created
 ✓ Container 44-db-1 Created
 ✓ Container 44-app-1 Created
Attaching to 44-app-1, 44-db-1
44-db-1 | The files belonging to this database system will be owned by user "postgres".
44-db-1 | This user must also own the server process.
44-db-1 |
44-db-1 | The database cluster will be initialized with locale "en_US.utf8".
44-db-1 | The default database encoding has accordingly been set to "UTF8".
44-db-1 | The default text search configuration will be set to "english".
44-db-1 |
44-db-1 | Data page checksums are disabled.
44-db-1 |
44-db-1 | fixing permissions on existing directory /var/lib/postgresql/data ... ok
44-db-1 | creating subdirectories ... ok
44-db-1 | selecting dynamic shared memory implementation ... posix
44-db-1 | selecting default max_connections ... 100
44-db-1 | selecting default shared_buffers ... 128MB
44-db-1 | selecting default time zone ... Etc/UTC
44-db-1 | creating configuration files ... ok
44-app-1 | * Serving Flask app 'main'
44-app-1 | * Debug mode: on
44-app-1 | 'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
44-app-1 | 'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
44-app-1 | 'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
44-app-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
44-app-1 | * Running on all addresses (0.0.0.0)
44-app-1 | * Running on http://127.0.0.1:5000
44-app-1 | * Running on http://172.20.0.3:5000
```

Output on port:5000 as :

Add TODO item

Please provide the TODO item content

Submit

TODO items

hussam
mcqs
are

My data will never be lost if I stop port , when I restore it ,It will show todo items