Graded Assignment 4.4

Name: Saad Sameer Khan

Employee #: 2303.KHI.DEG.034

Collaborated with: Mohammad Hamza Asim (2303.KHI.DEG.014)

Logging:

We see that logs are being shown on the terminal and that they are in debug mode by default.

```
(base) saadsameerkhan@all-MS-7D35:~/Documents/Assignments/Unit_4.4/test_assignment$ docker comp
[+] Building 5.5s (9/9) FINISHED
[+] Running 2/0
✓ Container test_assignment_myapp_2 Recreated
✓ Container test_assignment_myapp_1 Recreated
Attaching to test_assignment-myapp-1, test_assignment-myapp-2
                         | [2023-05-17 02:56:36 +0000] [1] [DEBUG] Current configuration:
                             config: ./gunicorn.conf.py
                             wsgi_app: None
                             bind: ['0.0.0.0:5000']
                             backlog: 2048
                             workers: 1
                             worker_class: sync
                             threads: 1
                             worker_connections: 1000
                             max requests: 0
                             max_requests_jitter: 0
                             timeout: 30
                             graceful_timeout: 30
                             keepalive: 2
                             limit_request_line: 4094
                             limit request fields: 100
                             limit_request_field_size: 8190
                             reload: False
                             reload_engine: auto
                             reload_extra_files: []
                             spew: False
                             check_config: False
                             print_config: False
                             preload_app: False
                             sendfile: None
                             reuse_port: False
                             chdir: /home/app
                             daemon: False
                             raw_env: []
pidfile: None
                             worker_tmp_dir: None
                             user: 0
                             group: 0
```

Now let's change the environment variable LOG_LEVEL from 'debug' to 'info' in the docker-compose file:

```
areas_for_improvement.txt × *docker-compose.yaml ×

1 version: '3'
2 services:
3 myapp:
4 build:
5 context: .
6 dockerfile: Dockerfile #basing our dockercompose on the Dockerfile
7 ports:
8 - "5005-5006:5000" #specifying a range of ports for the multiple instances
9 volumes:
10 - vol_data:/home/app/vol
11 deploy:
12 replicas: 2 #setting replicates allowing for multiple instances of containers to be created
13 environment:
14 - LOG_LEVEL=Info #this can be changed to info as needed
15
16 volumes:
17 vol_data: #setting up volume so that data can be persisted
```

Now only INFO logs are being shown:

Multiple instances of the application:

To have multiple instances of our app running, we have defined replicates to be 2 in the docker-compose file and we have also set the range of ports these replicates will be able to use.

```
areas_for_improvement.txt × *docker-compose.yaml ×

1 version: '3'
2 services:
3 myapp:
4 build:
5 context:
6 dockerfile: Dockerfile #basing our dockercompose on the Dockerfile
7 ports:
8 - "5005-5006:5000" #specifying a range of ports for the multiple instances
9 volumes:
10 - vol_data:/home/app/vol
11 deploy:
12 replicas: 2 #setting replicates allowing for multiple instances of containers to be created
13 environment:
14 - LOG_LEVEL=info #this can be changed to info as needed
15
16 volumes:
17 vol_data: #setting up volume so that data can be persisted
```

Installation in build time:

Moved requirements installation from run time to build time:

Efficient saving and other functionalities:

In the main3.py, we have made the following changes:

- Removed the functionality of logs being written to a logs.txt
- Instead of being saved every 10 seconds, the logs are saved on every hit
- Added a functionality that checks the environment variable and sets the debug mode of the app to on or off accordingly

```
import json
import logging
import threading
import os
from flask import Flask, render_template, request
app = Flask( name )
logging.basicConfig(
    format="%(asctime)s,%(msecs)d %(name)s %(levelname)s %(message)s",
    datefmt="%H:%M:%S",
TODO_FILE_NAME = "/home/app/vol/todo.json"
if os.path.exists(TODO_FILE_NAME):
    with open(TODO_FILE_NAME) as f:
        TODO_ITEMS = json.load(f)
    TODO_ITEMS = []
def save_items():
    while True:
        f=open(TODO FILE NAME, "w")
        json.dump(TODO ITEMS, f)
@app.route("/", methods=["GET", "POST"])
def main():
    tf request.method == "POST":
        content = request.form["content"]
        TODO_ITEMS.append(content)
        save items()
    return render_template("index.html", todo_items=TODO_ITEMS)
   __name__ == "__main__":
    if os.environ.get('LOG_LEVEL') == "info":
        app.debug = False
        app.debug = True
    app.run(host="0.0.0.0")
```

Persisting data through Volumes:

For the data we write in the todo-items to persist, we have defined named volumes in both Dockerfile and docker-compose.

```
areas_for_improvement.txt × docker-compose.yaml

1 FROM python:3.8-slim-buster

3 WORKDIR /home/app/

5 COPY ./ /home/app/

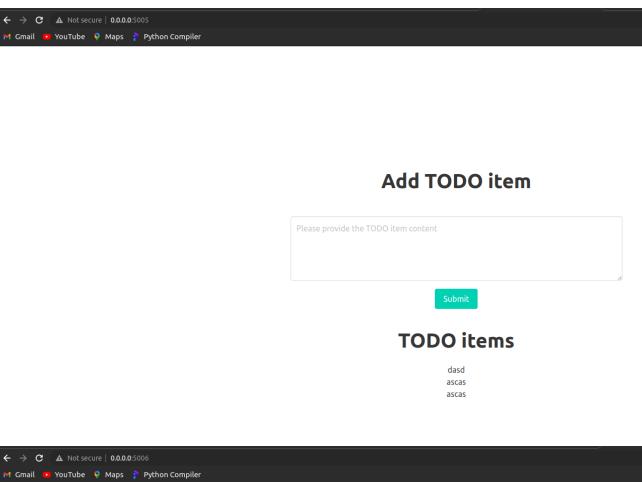
6
7 #moved the requirements.txt installation from run time to build time
8 RUN pip install -r requirements.txt

9
10
11 ENV PYTHONPATH=${PYTHONPATH}:/home/app/
12
13 #specified the volume where the data will be stored
14 VOLUME /home/app/vol
15
16
17 # here we have specified LOG_LEVEL which can be set in the docker-compose file
18 CMD ["bash", "-c", "gunicorn main3:app -b 0.0.0.0:5000 --log-level=$LOG_LEVEL"]
```

Now even if the containers are restarted, the data will still be there when we start the application again.

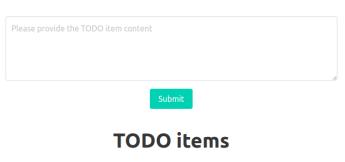
Results:

After running our application we can access it on the ports (5005 & 5006) as defined in the docker-compose file:





Add TODO item



dasd ascas ascas