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
 => [internal] load build definition from Dockerfile                          0.0s
 => => transferring dockerfile: 492B                                          0.0s
 => [internal] load .dockerignore                                             0.0s
 => => transferring context: 2B                                               0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim-buster  1.7s
 => [internal] load build context                                             0.0s
 => => transferring context: 34.43kB                                          0.0s
 => [1/4] FROM docker.io/library/python:3.8-slim-buster@sha256:89ad1c2cd0  0.0s
 => CACHED [2/4] WORKDIR /home/app/                                           0.0s
 => [3/4] COPY ./ /home/app/                                                  0.0s
 => [4/4] RUN pip install -r requirements.txt                                 3.7s
 => exporting to image                                                        0.1s
 => => exporting layers                                                       0.1s
 => => writing image sha256:7db022090f543ee1060c5b1957b95164c1b9f32bb3f29  0.0s
 => => naming to docker.io/library/test_assignment-myapp                      0.0s
[+] Running 2/0
 ✔ Container test_assignment_myapp_2  Recreated                               0.0s
 ✔ Container test_assignment_myapp_1  Recreated                               0.0s
Attaching to test_assignment-myapp-1, test_assignment-myapp-2
test_assignment-myapp-1  | [2023-05-17 02:56:36 +0000] [1] [DEBUG] Current configuration:
test_assignment-myapp-1  |   config: ./gunicorn.conf.py
test_assignment-myapp-1  |   wsgi_app: None
test_assignment-myapp-1  |   bind: ['0.0.0.0:5000']
test_assignment-myapp-1  |   backlog: 2048
test_assignment-myapp-1  |   workers: 1
test_assignment-myapp-1  |   worker_class: sync
test_assignment-myapp-1  |   threads: 1
test_assignment-myapp-1  |   worker_connections: 1000
test_assignment-myapp-1  |   max_requests: 0
test_assignment-myapp-1  |   max_requests_jitter: 0
test_assignment-myapp-1  |   timeout: 30
test_assignment-myapp-1  |   graceful_timeout: 30
test_assignment-myapp-1  |   keepalive: 2
test_assignment-myapp-1  |   limit_request_line: 4094
test_assignment-myapp-1  |   limit_request_fields: 100
test_assignment-myapp-1  |   limit_request_field_size: 8190
test_assignment-myapp-1  |   reload: False
test_assignment-myapp-1  |   reload_engine: auto
test_assignment-myapp-1  |   reload_extra_files: []
test_assignment-myapp-1  |   spew: False
test_assignment-myapp-1  |   check_config: False
test_assignment-myapp-1  |   print_config: False
test_assignment-myapp-1  |   preload_app: False
test_assignment-myapp-1  |   sendfile: None
test_assignment-myapp-1  |   reuse_port: False
test_assignment-myapp-1  |   chdir: /home/app
test_assignment-myapp-1  |   daemon: False
test_assignment-myapp-1  |   raw_env: []
test_assignment-myapp-1  |   pidfile: None
test_assignment-myapp-1  |   worker_tmp_dir: None
test_assignment-myapp-1  |   user: 0
test_assignment-myapp-1  |   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

```yaml
version: '3'
services:
  myapp:
    build:
      context: .
      dockerfile: Dockerfile #basing our dockercompose on the Dockerfile
    ports:
      - "5005-5006:5000" #specifying a range of ports for the multiple instances
    volumes:
      - vol_data:/home/app/vol
    deploy:
      replicas: 2 #setting replicates allowing for multiple instances of containers to be created
    environment:
      - LOG_LEVEL=info #this can be changed to info as needed

volumes:
  vol_data: #setting up volume so that data can be persisted
```

Now only INFO logs are being shown:

```
(base) saadsameerkhan@all-MS-7D35:~/Documents/Assignments/Unit_4.4/test_assignment$ docker compose up
[+] Running 3/1
 ✔ Network test_assignment_default    Created
 ✔ Container test_assignment-myapp-2  Created
 ✔ Container test_assignment-myapp-1  Created
Attaching to test_assignment-myapp-1, test_assignment-myapp-2
test_assignment-myapp-1  | [2023-05-17 02:59:07 +0000] [1] [INFO] Starting gunicorn 20.1.0
test_assignment-myapp-1  | [2023-05-17 02:59:07 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
test_assignment-myapp-1  | [2023-05-17 02:59:07 +0000] [1] [INFO] Using worker: sync
test_assignment-myapp-1  | [2023-05-17 02:59:07 +0000] [8] [INFO] Booting worker with pid: 8
test_assignment-myapp-2  | [2023-05-17 02:59:08 +0000] [1] [INFO] Starting gunicorn 20.1.0
test_assignment-myapp-2  | [2023-05-17 02:59:08 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
test_assignment-myapp-2  | [2023-05-17 02:59:08 +0000] [1] [INFO] Using worker: sync
test_assignment-myapp-2  | [2023-05-17 02:59:08 +0000] [8] [INFO] Booting worker with pid: 8
```

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

```yaml
version: '3'
services:
  myapp:
    build:
      context: .
      dockerfile: Dockerfile #basing our dockercompose on the Dockerfile
    ports:
      - "5005-5006:5000" #specifying a range of ports for the multiple instances
    volumes:
      - vol_data:/home/app/vol
    deploy:
      replicas: 2 #setting replicates allowing for multiple instances of containers to be created
    environment:
      - LOG_LEVEL=info #this can be changed to info as needed

volumes:
  vol_data: #setting up volume so that data can be persisted
```

Installation in build time:

Moved requirements installation from run time to build time:

areas_for_improvement.txt | docker-compose.yaml | Dockerfile

```
FROM python:3.8-slim-buster

WORKDIR /home/app/

COPY ./ /home/app/

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


ENV PYTHONPATH=${PYTHONPATH}:/home/app/

#specified the volume where the data will be stored
VOLUME /home/app/vol


# here we have specified LOG_LEVEL which can be set in the docker-compose file
CMD ["bash", "-c", "gunicorn main3:app -b 0.0.0.0:5000 --log-level=$LOG_LEVEL"]
```

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

```python
import json
import logging
import threading
import time
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)
else:
    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():
    if request.method == "POST":
        content = request.form["content"]
        TODO_ITEMS.append(content)
        save_items()

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


if __name__ == "__main__":
    #this will run the app in debug mode or info mode according the ENV variable that is passed in docker-compose file
    if os.environ.get('LOG_LEVEL') == "info":
        app.debug = False
    else:
        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

```yaml
version: '3'
services:
  myapp:
    build:
      context: .
      dockerfile: Dockerfile #basing our dockercompose on the Dockerfile
    ports:
      - "5005-5006:5000" #specifying a range of ports for the multiple instances
    volumes:
      - vol_data:/home/app/vol
    deploy:
      replicas: 2 #setting replicates allowing for multiple instances of containers to be created
    environment:
      - LOG_LEVEL=debug #this can be changed to info as needed

volumes:
  vol_data: #setting up volume so that data can be persisted
```

areas_for_improvement.txt × docker-compose.yaml

```dockerfile
FROM python:3.8-slim-buster

WORKDIR /home/app/

COPY ./ /home/app/

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


ENV PYTHONPATH=${PYTHONPATH}:/home/app/

#specified the volume where the data will be stored
VOLUME /home/app/vol


# here we have specified LOG_LEVEL which can be set in the docker-compose file
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: