Lab 1 - Containers

In this lab practice, you are required to build two containers, server and client, configure network connection between the two containers, mount a volume for server, and configure multiple containers with `docker-compose`.

Requirements

Please first download the data for this lab and extract it:

```
$ wget ntu.im/IM5057/lab1.tar.gz
$ tar zxvf lab1.tar.gz
```

The `lab1` folder contains the following directories:

```
lab1/
|
+-- client/
|   |
|   +-- client.py
|
+-- server/
    |
    +-- server.py
```

The `.py` files for server and client are provided, and you are not allowed to modify these `.py` files. You are allowed to add any files (but not code being executed).

1. Connection between Server and Client

As the server process starts running, it listens on port `5000`. After receiving request from client, it saves a data `msg.txt` into the `data` directory:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def get():
    with open('data/msg.txt', 'w') as f:
        f.write("Hi!\n")
    return "Hi!"

app.run()
```

The client process does nothing but sends an http request to the url `http://server:5000`:

```
import requests

requests.get('http://server:5000')
```

The server is implemented using `flask`, which should be installed in the container so that the server can be successfully executed. Similarly, the `requests` package should be installed in the client container.

To run the server process, you may use the command `flask run` under the directory containing the `server.py` file, with the environment variables `FLASK_APP` and `FLASK_RUN_HOST` set to `server.py` and `0.0.0.0`, respectively.

Since the url should not be modified, you should guarantee that the client is able to access the server using the url `http://server:5000`. Also, the server container should be deployed before the client container, so that the server is ready when the client sends its request.

2. Mounting Server Volume

As the server receives request from client, it saves a data `msg.txt` into the `data` directory (in the container). To persist the data, you are required to use volume for the server container. More specifically, the data should be accessible in the `./data` directory (out of the container), even after the container is stopped.

3. Docker-Compose

You may use multiple `Dockerfile`s to build your server and client containers. However, you should also provide a `docker-compose.yml` file, so that the two containers can be deployed using a single `docker compose up` command. Note that your images should be built locally, that is, pulling images in `docker-compose.yml` is not allowed.

Submission

Rename your `lab1` folder to `<Student ID>_lab1`, and compress the whole folder in `<Student ID>_lab1.zip`. For grading, we `cd` into your `lab1` directory and run a test as below:

```
$ ls
client              docker-compose.yml server
$ docker compose up -d --build > /dev/null
[+] Running 2/2
 ⠿ Container lab1_d-server-1  Started 0.3ss
 ⠿ Container lab1_d-client-1  Started 1.0ss
$ cat data/msg.txt
Hi!
$ docker compose down
[+] Running 1/2
 ⠿ Container lab1_d-client-1  Removed                0.0s
 ⠿ Container lab1_d-server-1  Removed               10.2s
 ⠿ Network lab1_d_default     Removed                0.1s
$ cat data/msg.txt
```

```
Hi!
```

Late submissions will not be accepted.