Assignment 1: Docker Containers and NoSQL

Topics Covered:

- Docker containers and Docker-compose
- NoSQL

Marking Scheme (Total 100 points):

- Write a small Python app to download data (10 points)
- Dockerize the app to collect data from the source (20 points)
- Publish the Docker image of your app in Docker Hub in your account (20 points)
- Change the app to store data in MongoDB. Run the app with MongoDB (using images from Docker Hub) using Docker-compose (50 points)

Date of Submission: 10th May

Format of Submission (ZIP file) with:

1. A text file with the names of the people in the group and a link to the published Docker image in Docker Hub.
2. Code of your application with all necessary keys. (Don't worry; you can revoke and delete your key once your assignment is over)
3. DockerFile for your app you used to dockerize the app.
4. Final Docker-compose file to run either the published Docker image.

Instructions:

1. **Collecting Data from Source:**
    - We are going to capture weather data.
    - Go to WeatherAPI, register for the test API, and then copy the code for making API calls for real-time weather.
    - This API returns the current weather of a given location. Location is passed as a query string and can be zip codes.
    - Parse the CSV Top100-US.csv and store the zip code and city name in a collection called city_weather in MongoDB. The document you store in MongoDB should look like this:

```
{
    _id: <this is auto generated>,
    zip: <this is from the CSV>,
    city: <this is from the CSV>,
    created_at: <the current time when this document was created>,
    weather: <this is the output of the weather result from the Weather API>
}
```

- Note: Whenever a user runs your Python code, it should parse the CSV and insert one new document per row in the CSV. So if you run the Python code 10 times, it should create 1000 entries in the MongoDB collection.

2. **Dockerize the Python App:**
    - Dockerize your application using any approach you want. You can refer to this link for guidance.
3. **Publish the Docker Image of Your App in Docker Hub in Your Account:**
    - Once you test your Docker image locally, publish the Docker image in your Docker Hub account. You would need to create an account on Docker Hub (https://hub.docker.com/) and make sure to login to Docker using the command:

```
$ docker login
```

4. **Create Docker-Compose:**
    - Create the Docker-compose file having two services:
        - Your Dockerized Python app (using the published image)
        - MongoDB
    - When a user runs the Docker-compose file, it should spin up your MongoDB database and run the Python code to parse the 100 cities and update its data in the MongoDB database.
    - Make sure the data in the MongoDB database is persisted using Docker volumes so that even if we shut down the Docker containers, the data in your MongoDB is not lost. So if a user runs your Docker-compose file 2 times, it should create 200 records in the database.