Movatic Backend Coding Challenge

The following code challenge is an opportunity for you to show off and share your coding ability and process. During the technical interview we will discuss your implementation with an emphasis on your process and decisions. You should approach this the same way you would approach writing long-lasting code that would be used in a production environment.

Scenario

A large national bikeshare operator wants to be able to view station data from bike share systems from across the world. They want to do this by creating an application that stores and provides data that is sourced from third-party GBFS feeds.

- This is an example of a working GBFS feed: https://gbfs.bcycle.com/bcycle_madison/gbfs.json

You have been tasked to create the backend API for a demo of this product to get customer buy-in. Assume that this demo would be used as the foundation for a full implementation, so it should be developed in a way that is maintainable and extensible in the future. You have full control over the database schema and the implementation details and will only be provided with the basic API requirements.

Requirements

1. Implement a `GET /stations` endpoint that returns a JSON response with a list of stations stored in the database. For each station, include the following fields: “station_id”, “name” (string), “address” (string), “latitude” (number), “longitude” (number).

2. Implement a `GET /stations/{station_id}/status` endpoint that returns a JSON response with the status information of a particular station stored in the database. The fields to include are: “is_returning” (boolean), “is_renting” (boolean), “is_installed” (boolean), “num_docks_available” (number), “num_bikes_available” (number), “last_reported” (datetime string).

3. Implement a `POST /ingest` endpoint that accepts a “gbfs_url” parameter in a JSON message body. The endpoint should query the provided URL and then insert or update data in the database with the latest information about the stations. Assume that the data provided will always be in the same format as https://gbfs.bcycle.com/bcycle_madison/gbfs.json. This should:

    a. Look for the “station_information” and “station_status” URLs
    b. Read the latest data from those URLs
    c. Update the database accordingly

4. Use Movatic's stack:

    a. Docker containers (one for the web application and one for the database)
    b. Python with Flask for the web application
    c. CockroachDB for the database

5. Use a `docker-compose` file. The `docker-compose` file should be used to orchestrate all the containers. Running the project should be as simple as doing `docker-compose up`.

6. Spend no more than 4 hours **writing the code** for this project. This time does not include the time needed for any other tasks such as setting up the project, doing research, installing dependencies, writing the README, etc. The goal is to spend as much time as you need figuring things out, but only 4 hours of coding at most to keep things simple.

7. Include a `README.md` with information about the project

8. Push your project to a public Git repository and the link with us when done

9. Be prepared to showcase the project and discuss the following questions in the technical interview:

    a. What was your approach for the project (development process)?
    b. What was your biggest challenge while developing this project?
    c. Which additional libraries or tools did you use (if any) and why?
    d. What would be the next changes or features that you would add to this project if you had more time?
    e. How would you develop a feature that would allow the user to see historic data (i.e. being able to see data from the past)?
    f. Imagine that this application would eventually manage a very large amount of data. Which parts of your implementation are not very scalable? How would you improve them?

Good luck!