Coding Challenge

Rules

- Should be completed in 3 days
- Should be done exclusively by candidate
- Should be delivered by pushing to a git project
    - Several commits along the way (optional)
    - One final commit (mandatory)
- Accepted Sources of deliverables
    - Written by candidate
    - Generated by IDE
    - Generated by online application initialisers
    - Generated by documentation generators

Preferred Languages

- Java
- Golang
- Python
- Javascript/Typescript

Deliverables

- Source Code (mandatory)
- README with instructions on how to run (mandatory)
- Docker file (mandatory)
- Docker compose file (mandatory)
- API Documentation (optional)
- Manual tests for the API (optional)
    - cURL
    - Postman
    - ...

Evaluation

Evaluation will be done based on the following points (by order of importance):

1. Delivered project runs, exposes API, is delivered in GIT, includes README and build/run script
2. Full Requirements implemented
3. Clean code following best practices and using common Design Patterns
4. Efficient code
5. Reasonable Unit Test coverage
6. Comprehensive documentation
7. Manual tests present
8. Regular commits on project milestones8

General Advice to Candidate

1. Read all before starting
2. If additional information is needed, ask!
3. Order of importance in evaluation matters

Description of the Challenge

We're building a data usage processing system for mobile phones at ACME Corp and need to expose that usage with an API.

The cellular network produces events with the following schema:

```
{
    "sim-card-id": "89440001",
    "bytes-used": "1024",
    "date": "2022-02-18T00:00:00Z"
}
```

Inside ACME Corp, SIM Cards are partitioned across multiple organizations. Consider the following SIM Card Inventory:

```
[
  { "sim-card-id": "89440001", "org-id": "a01b7" },
  { "sim-card-id": "89440002", "org-id": "a01b7" },
  { "sim-card-id": "89440003", "org-id": "a01b7" },
  { "sim-card-id": "89440004", "org-id": "a01b7" },
  { "sim-card-id": "89440005", "org-id": "a01b7" },
  { "sim-card-id": "89440006", "org-id": "x00g8" },
  { "sim-card-id": "89440007", "org-id": "x00g8" },
  { "sim-card-id": "89440008", "org-id": "x00g8" },
  { "sim-card-id": "89440009", "org-id": "x00g8" },
  { "sim-card-id": "89440010", "org-id": "x00g8" },
  { "sim-card-id": "89440011", "org-id": "f00ff" },
  { "sim-card-id": "89440012", "org-id": "f00ff" },
  { "sim-card-id": "89440013", "org-id": "f00ff" },
  { "sim-card-id": "89440014", "org-id": "f00ff" },
  { "sim-card-id": "89440016", "org-id": "f00ff" }
]
```

We need a service that is able to consume a stream of network usage events and store them in a way that it is able to scale, and expose the information over an API with the following characteristics:

- The ability to retrieve the total usage for all SIM cards under a given organisation over an interval of time
- The ability to retrieve the total usage for a single SIM card over an interval of time
- Possible Usage granularities: DAILY, HOURLY
- Extra Points: Paginated responses

To complete the challene you will need to:

- Run the setup already provided by executing the command `docker-compose up`, which will create:
  - A zookeeper instance running on `zookeeper:2181 / localhost:22181` (Taken from https://www.baeldung.com/ops/kafka-docker-setup)
  - A kafka instance running on `kafka:9092 / localhost:29092` (Taken from https://www.baeldung.com/ops/kafka-docker-setup)
  - A python container that runs a script which injects ~40k events into the topic `usage` every time it runs (its just for demo purposes, so the container will terminate after the script, and you can run it multiple times)
- Design a database schema that is able to store/aggregate the events according to what the API needs to output
- Implement a kafka consumer that reads the events from the `usage` topic and stores them on a database with the schema created in the previous point
  - **Suggestion**: Use the kafka-python guide https://kafka-python.readthedocs.io/en/master/usage.html
- An API that allows querying the data stored in the previous point
  - **Suggestion**: Can be the same application as the kafka consumer
  - Implement the endpoints you find appropriate
  - Doesn't need to be 100% full REST compliant, just needs to make sense

Example Requests and expected responses (Values are dummy data, not the actual expected ones)

"I want the DAILY usage for ALL simcards in orgainzation f00ff between 2022-02-10 and 2022-02-12"

```
[
    {
        "bytes-used-total": 3123322,
        "date": "2022-02-10"
    },
    {
        "bytes-used-total": 485223128,
        "date": "2022-02-11"
    },

    {
        "bytes-used-total": 123646544,
        "date": "2022-02-12"
    }
]
```

"I want the HOURLY usage for SIMCARD 89440006 between 2022-02-10T00:00:00 and 2022-02-10T02:00:00"

```
[
    {
        "bytes-used-total": 412312,
        "date": "2022-02-10:00:00:00"
    },
    {
        "bytes-used-total": 432432,
        "date": "2022-02-10:01:00:00"
    },

    {
        "bytes-used-total": 432432,
        "date": "2022-02-10:02:00:00"
    }
]
```