Pre-Assessment

You've been asked to spin up an API on the following architecture:

- Application layer: Either Node.js with express, or Golang
- Database: MongoDB or Postgres

The backend API, containerized to docker, should expose following functionality:

1. POST requests to `/orgs/<org-name>/comments` should allow the user to persist comments (in a MongoDB collection or Postgres table) against a given github organization.

For example, the following request should save a comment against the Xendit org:

```
POST /orgs/xendit/comments/
Content-Type: application/json

{
    "comment": "Looking to hire SE Asia's top dev talent!"
}
```

2. `GET` requests to `/orgs/<org-name>/comments/` should return an array of all the comments that have been registered against the organization.

3. `DELETE` requests to `/orgs/<org-name>/comments` should soft delete all comments associated with a particular organization. We define a "soft delete" to mean that deleted items should not be returned in GET calls, but should remain in the database for emergency retrieval and audit purposes.

4. `GET` requests to `/orgs/<org-name>/members/` should return an array of members of an organization (with their login, avatar url, the numbers of followers they have, and the number of people they're following), sorted in descending order by the number of followers.

At the very minimum, you should have:
1. Proper documentation
2. Adequate unit testing

We also require that your solution handle as many of the following as you can manage, given the available time:

1. Deployed the two endpoints above (i.e., `/orgs/<org-name>/comments`, and `/orgs/<org-name>/members`) on separate microservices, either as
   (a) Lambda functions, or
   (b) docker containers orchestrated using Kubernetes, Docker swarm, or a similar tool

2. Adequately logged what needed to be logged

3. Securely handled secrets (such as DB connection strings and other sensitive data)

4. Integration and/or E2E testing