Software Engineer Assignment: GitHub Crawler

Scenario / Deliverables

Using GitHub's GraphQL API, obtain a table of the number of stars of 100,000 Github repos. Respect all rate limits (see Github docs for info) and consider retry mechanisms.

Please store the crawled data in a table in a relational database like Postgres. Use a schema for your data. The schema should be flexible, and updating the data inside the DB should be efficient. Ideally, we want to crawl this data continuously daily to keep up with the activity on GitHub.

Write a list of what you would do differently if this were run to collect data on 500 million repositories instead of 100,000.

How will the schema evolve if you want to gather more metadata in the future, for example, issues, pull requests, commits inside pull requests, comments inside PRs and issues, reviews on a PR, CI checks, etc.? A PR can get 10 comments today and then 20 comments tomorrow. Updating the DB with this new information should be an efficient operation (minimal rows affected)

What's Important

We pay particular attention to the duration of crawl-stars, which should have run as quickly as possible.

We also pay attention to how you structure your code and general software engineering practices, like using an anti-corruption layer, as much immutability as possible, separation of concerns and following clean architecture principles.

Your GitHub pipeline should contain the following:

1. A postgres [service container](https://docs.github.com/en/actions/use-cases-and-examples/using-containerized-services/creating-postgresql-service-containers
2. Any number of steps for setup & dependency installs.
3. A setup-postgres step that creates any tables and their respective schemas.
4. A crawl-stars step that uses the GitHub API to obtain a list of 100,000 GitHub repositories and their respective star counts (not necessarily ordered by anything - it can be any subset of repositories).
5. Any number of steps which dump the contents of your database and uploads the result as an artifact (csv, json, etc. - anything works here)
6. Ensure your GitHub Actions pipeline works using the default GitHub Token, and does not require elevated permissions or access to private secrets.

How to submit your solution

1. Complete the assignment in a **public GitHub repository under your own account**.
2. Ensure your **GitHub Actions pipeline** is functional and has at least one successful run.
3. Share the link to your repository with us via email before the submission deadline.