gh-cobra

A simple exercise using GitHub Cobra CLI to create an app that accesses the GitHub REST and GraphQl API's. It also gives help on basic Bash utilities.

Note : I use GitHub Copilot in Visual Studio Code, which made it much easier and quicker to get working. Especially working with the GitHub GraphQL API

Commands

- $gh-cobra api
  - Print a list of public repositories on GitHub that belong to a specified owner.
  - Uses the GitHub REST API
  - see cmd/api.go
- $gh-cobra graphql
  - Print a list of public repositories on GitHub that belong to a specified owner.
  - Uses the GitHub Graphql API with a manual POST request
  - This command requires a GitHub bearer token in the environment variable GITHUB_TOKEN
  - see cmd/graphql.go
- $gh-cobra shurcool
  - Print a list of public repositories on GitHub that belong to a specified owner.
  - Uses the GitHub Graphql API using the shurcooL/graphql client package.
  - This command requires a GitHub bearer token in the environment variable GITHUB_TOKEN
  - see cmd/shurcool.go
- $gh-cobra gogithub
  - Print a list of public repositories on GitHub that belong to a specified owner.
  - Uses the GitHub Graphql API using the google/go-github client package.
  - This command requires a GitHub bearer token in the environment variable GITHUB_TOKEN
  - see cmd/shurcool.go
- $gh-cobra explain
  - print help for basic Linux Bash utilties
  - This command requires an OpenAI api key in the environment variable OPENAI_API_KEY
  - see cmd/explain.go and cmd/lc.go

Notes

- Built with Go version 1.8 or later
  - Run the app without building : 'go run . args..."
  - Build the app : 'go build ."
- The 'explain' command may give a warning like "Failed to calculate number of tokens...".
  - This warning is generated by the version of Langchain used in this app.
- You can add more to the list of supported names in the 'explain' command by updating the list in lc.go

shurcooL vs gogithub

There is a tradeoff between using the go-github client vs the shurcooL client.

- The go-github client package is much much easier to use, as it provides direct functions to access the API and takes care of all the types for you.
    - A drawback of the go-github package functions is that GitHub API types can contain a LOT of data elements. For example, when you use the client.Repositories.ListByUser, you get a ton of information and the associated overhead. This sort of defeats the purpose of GraphQL, which is designed to let you request only the data you want. You might as well use the REST API.
- the shurcooL client package is a bit more complicated to use.
    - you have to figure out the types and formats of the queries and mutations. Which can take a bit of exploring using the GitHub schema and Explorer. In my case using GitHub Copilot made it 10 times easier to figue this stuff out.