JumpCloud Interview Assignment

Prerequisites

- You will need a computer that meets the system requirements of running the Go tools.
- Basic Git knowledge
- A GitHub.com account

Setup

Install Go Tools

If you are new to Go please follow the install instructions to get the Go tools installed and working on your machine. After you have completed the "Test Your Installation" section you should be ready to proceed.

Directory Structure

By default Go assumes that your `$GOPATH` is set to `$HOME/go`. If you use the defaults, your environment will require no additional setup after you have installed the Go Tools.

Create the directory
`$HOME/go/src/github.com/<your_github_username>/<name_of_your_repo_for_the_assignment>`
In Go, all files in a directory have to be in the same package. If you plan to use more than one package you should create sub-directories.

You should now be ready to start the assignment.

Requirements

The project code should only import packages available in the standard library.

There are no super secret hidden solutions. The assignment is designed to see how you approach writing code to solve problems. We are also looking to see code that represents your view of "production quality". If at any step along the way you have questions feel free to reach out.

To submit your work please push to a public GitHub project, and be sure to document any configuration or run instructions. We're looking for a solution to the problem as well as attention to detail and code craftsmanship. Good luck and have fun!

Hash and Encode a Password String

Write an HTTP server that listens on a given port. Your server should be able to process multiple connections simultaneously. And provide the following endpoints:

Accept `POST` requests on the `/hash` endpoint with a form field named `password` to provide the value to hash. An incrementing identifier should return immediately but the password should not be hashed for 5 seconds. The hash should be computed as base64 encoded string of the SHA512 hash of the provided password.

For example, the first request to:
`curl —data “password=angryMonkey” http://localhost:8080/hash`
should return `1` immediately. The 42nd request should return `42` immediately.

5 seconds after the `POST` to `/hash` that returned an identifer you should be able to `curl http://localhost:8080/hash/42` and get back the value of `“ZEHhWB65gUlzdVwtDQArEyx+KVLzp/aTaRaPlBzYRIFj6vjFdqEb0Q5B8zVKCZ0vKbZPZklJz0Fd7su2A+gf7Q==”`

Statistics End-Point

Provide a statistics endpoint to get basic information about your password hashes.

A `GET` request to `/stats` should return a JSON object with 2 key/value pairs. The `“total”` key should have a value for the count of `POST` requests to the `/hash` endpoint made to the server so far. The `“average”` key should have a value for the average time it has taken to process all of those requests in microseconds.

For example: `curl http://localhost:8080/stats` should return something like:
`{“total”: 1, “average”: 123}`

Graceful Shutdown

Provide support for a “graceful shutdown request”. If a request is made to `/shutdown` the server should reject new requests. The program should wait for any pending/in-flight work to finish before exiting.