QA Assignment:

For this assignment, JumpCloud has implemented a password hashing application in Golang and we have intentionally left bugs in it. The assignment is to write the test cases needed to test the application, explain your choices for coverage, execute the test cases and report the bugs you find. Deliverables should be submitted along with a README in a GitHub repo that you share with us.
The style, depth, scope and type of tests and bug reports you write are up to you. Use them to demonstrate your style and strengths.

Obtaining the Password Hashing Application

We are storing the password hashing application in a public S3 bucket. You can get it in the following manner:

Linux example:

```
$ wget --no-check-certificate --no-proxy
'https://s3.amazonaws.com/qa-broken-hashserve/broken-hashserve.tgz'
```

Windows example:
On modern Windows systems with PowerShell 3.x, curl/wget/iwr are just aliases for `Invoke-WebRequest`. It has equivalent curl/wget/iwr parameters and output; they are just named different things. On Windows 10, you will also need to install `7zip` or equivalent extraction tool to unpack the `*.tgz` archive.

```
C:/> iwr -Uri
https://s3.amazonaws.com/qa-broken-hashserve/broken-hashserve.tgz
-UseBasicParsing -o ./broken-hashserve.tgz
```

Password Hashing Application Execution

The `broken-hashserve.tgz` archive contains binaries for Linux, Windows & Mac OS X operating systems. Unpack and use the binary corresponding to your OS of choice. We are certain that the application will not run amok and trash your workstation/laptop; that being said, you should always run untrusted binaries in a virtual environment for a number of reasons :)

You ***must*** set a `PORT` environment variable before executing the application. It will crash otherwise.

Linux example:
```
$ export PORT=8088
```

Windows example:
```
1. C:/> SET PORT=8088
2. Use the Control Panel to set a System or User variable for
   PORT.  Remember to reopen your cmd window after doing this.
```

We tested the password hashing application on the following operating systems:
- Ubuntu 16.04
- Mac OS X - Sierra, High Sierra
- Windows 10

Password Hashing Application Specification

The following is the requirement specification that was used in building the password hashing application. It describes what the application ***should*** do.

- When launched, the application should wait for http connections.
- It should answer on the `PORT` specified in the `PORT` environment variable.
- It should support three endpoints:
  - A `POST` to `/hash` should accept a password. It should return a job identifier immediately. It should then wait 5 seconds and compute the password hash. The hashing algorithm should be SHA512.
  - A `GET` to `/hash` should accept a job identifier. It should return the base64 encoded password hash for the corresponding POST request.
  - A `GET` to `/stats` should accept no data. It should return a JSON data structure for the total hash requests since the server started and the average time of a hash request in milliseconds.
- The software should be able to process multiple connections simultaneously.
- The software should support a graceful shutdown request. Meaning, it should allow any in-flight password hashing to complete, reject any new requests, respond with a *200* and shutdown.
- No additional password requests should be allowed when shutdown is pending.

Interacting with the Password Hashing Application

You can interact/test the application using curl. The following are examples that would/should generate similar returns - the job identifier does not need to conform to a specification.

- Post to the /hash endpoint

```
$ curl -X POST -H "application/json" -d '{"password":"angrymonkey"}'
http://127.0.0.1:8088/hash
> 42
```

- Get the base64 encoded password

```
$ curl -H "application/json" http://127.0.0.1:8088/hash/1
> zHkbvZDdwYYiDnwtDdv/FIWvcy1sKCb7qi7Nu8Q8Cd/MqjQeyCI0pWKDGp74A1g==
```

- Get the stats

```
$ curl http://127.0.0.1:8088/stats
> {"TotalRequests":3,"AverageTime":5004625}
```

- Shutdown

```
$ curl -X POST -d ‘shutdown’ http://127.0.0.1:8088/hash
> 200 Empty Response
```