Introduction

At Echo Global Logistics, we value working on real solutions to real problems, and as such we think the best way to understand your capabilities is to give you the opportunity to solve a problem that features various skill sets that we use on a daily basis. As part of the next step in our process, we ask that you write a simple console program that monitors HTTP traffic on your machine. Treat this as an opportunity to show us how you would write something efficient, scalable, well-structured and maintainable.

We will primarily be looking at the following criteria:

- Logical and maintainable code structure
- Efficient use of data structures
- Comments and unit tests
- Correctness
- Insight into potential improvements

Please set aside at least 15 minutes to provide us with a README write-up about the challenge. Documentation, notes, reflection, etc.

We understand that your time is valuable, we take that seriously, this is not an open ended challenge. Please timebox yourself to about three hours. We care about how you manage your time and what your general thought process is, not how much time you can dedicate to a task over the weekend. Your challenge results will be reviewed by multiple members of our team. Once completed, please submit your code, write-up, and any other artifacts to whomever gave you the challenge. The deadline is at least one full business day before your scheduled on-site interview, to give us time to review, but the more time the better. Additionally, have a local copy available for possible improvements and discussion during the on-site interview.

The Challenge

HTTP Log monitoring console program:

- Read a CSV-encoded HTTP access log.
- It should either take the file as a parameter or read from standard input. Ideally, both should work.
- For every 10 seconds of log lines, display stats about the traffic during those 10 seconds: the sections of the web site with the most hits, as well as statistics that might be useful for debugging. A section is defined as being what's before the second '/' in the resource section of the log line. For example, the section for "/api/user" is "/api" and the section for "/report" is "/report".
- Whenever total traffic for the past 2 minutes exceeds a certain number *on average*, print a message to the console saying that "High traffic generated an alert - hits = {value},

triggered at {time}". The default threshold should be 10 requests per second but should be configurable.
- Whenever the total traffic drops again below that value on average for the past 2 minutes, print another message detailing when the alert recovered, +/- a second.
- Consider the efficiency of your solution and how it would scale to process high volumes of log lines - don't assume you can read the entire file into memory.
- Write your solution as you would any piece of code that others might need to modify and maintain, both in terms of logic, structure and style.
- Write a test for the alerting logic
- Explain how you'd improve on this application design

README Write-up Topics:

- General documentation
- Instructions on how to run it
- List of areas for improvement
- Reflection on how you thought you did with the challenge

Log File:

Should be provided along with this document. Called `echo-coding-challenge-log.csv`
Example log file (first line is in the header):

```
"remotehost","rfc931","authuser","date","request","status","bytes"
"10.0.0.2","-","apache",1549573860,"GET /api/user HTTP/1.0",200,1234
"10.0.0.4","-","apache",1549573860,"GET /api/user HTTP/1.0",200,1234
"10.0.0.4","-","apache",1549573860,"GET /api/user HTTP/1.0",200,1234
"10.0.0.2","-","apache",1549573860,"GET /api/help HTTP/1.0",200,1234
"10.0.0.5","-","apache",1549573860,"GET /api/help HTTP/1.0",200,1234
"10.0.0.4","-","apache",1549573859,"GET /api/help HTTP/1.0",200,1234
"10.0.0.5","-","apache",1549573860,"POST /report HTTP/1.0",500,1307
```

Guidelines and Clarifications

- The time in the alert message can be formatted however you like (using a timestamp or something more readable are both fine), but the time cited must be in terms of when the alert or recovery was triggered in the log file, *not* the current time.
- Make reasonable assumptions about how to handle the 10-second intervals, there are a couple of valid options here. Make a similar assumption about how frequently stats should be displayed as you process (but don't just print them at the end!).
- Try to only make one pass over the file overall, for both the statistics and the alerting, as if you were reading it in real time.
- The date is in Unix time (https://en.wikipedia.org/wiki/Unix_time)
- You are free to use Google, StackOverflow, etc, as well as standard and open source libraries, but the core of the alerting logic must be your own
- Duplicate alerts should not be triggered - a second alert should not be triggered before a

recovery of the first
- The alerting state does not need to persist across program runs
- Package your source code along with instructions on how to run it into a zip/tar file of some sort.

Deadline and Deliverables

Note: Again, we ask that you spend about 3 hours on the challenge

Deadline is “At least One Full Business Day before On-site Interview”. We need time to review the submission and prepare!

Deliverables:

Please zip everyone up and include your name and the date in the file name. Please do not put the challenge, or any of your answers, in a public git repo. If you want to host it somewhere, please make sure it’s a private repo and just send us the zip.

- Code
- README with write-up answers
- Any notes, diagrams, whiteboards, brainstorming or other pieces that were part of your planning process.
- Git history, if you applicable