Introduction

At Datadog, 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 similar to the ones we solve on a daily basis. As 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

We ask that you submit what you've completed - we promise we will read it carefully, even if it is incomplete! We also recommend leaving at least 15 minutes to write instructions and how you would improve the program.

The problem / assignment

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. All time-based calculations should be relative to the timestamps in the log-file. Avoid using system time in your program.

Example log file (first line is the header):

```
"remotehost","rfc931","authuser","date","request","status","bytes"
"10.0.0.1","-","apache",1549574332,"GET /api/user HTTP/1.0",200,1234
"10.0.0.4","-","apache",1549574333,"GET /report HTTP/1.0",200,1136
"10.0.0.1","-","apache",1549574334,"GET /api/user HTTP/1.0",200,1194
```

```
"10.0.0.4","-","apache",1549574334,"POST /report HTTP/1.0",404,1307
```

- For every 10 seconds of log lines, display stats about the traffic during those 10s: 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 structure and style
- Write a test for the alerting logic
- Explain how you’d improve on this application design

You may download the log file here

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 a reasonable assumption 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 only 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