UFIRSTGROUP

The UFirst Home Work

Thank you again for your agreement in spending your time solving the UFirst Assignment. We are looking forward to working with you and are excited to see how your solution will look and how you solved all the little obstacles.

Many thanks in advance and - surprise us ! 😎

The Task

Server access analysis

This document contains an assignment for candidates applying for a job as software engineer at UFirst. We are aware that the assignment is rather substantial. However, you may decide to solve only the backend part or only the frontend part.

If you would decide that you don't wanna work on the backend task we would provide you the required JSON-Files.

How to solve the problem

The main programming languages at UFirst are PHP and Javascript. However, feel free to solve the task in **any language of your choice**. Furthermore, you are free in the choice of the setup; your solution may run on a web server but may also be a standalone command line script and static html/js. Also consider using open source modules and libraries where it makes sense - you do not need to reinvent the wheel.

Submission of your solution

Please submit your solution within **7 days** from the moment you receive it. If your solution depends on specific system components (e.g. a web server), please deploy the application somewhere on the web and provide us with a link. If you need more time we kindly ask you to provide us with your suggested delivery date **before you start with the work**.

The assignment

Your task is to import the access logs for the EPA from 1995, restructure the data and provide a graphical analysis of the data.

The data

Enclosed in this archive is a file named **epa-http.txt**

There is as well as the description from the following website (not always available): http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html

Description

This trace contains a day's worth of all HTTP requests to the EPA WWW server located at Research Triangle Park, NC.

Format

The logs are an ASCII file with one line per request, with the following columns:

1. **host** making the request. A hostname when possible, otherwise the Internet address if the name could not be looked up.
2. **date** in the format "[DD:HH:MM:SS]", where **DD** is either "29" or "30" for August 29 or August 30, respectively, and **HH:MM:SS** is the time of day using a 24-hour clock. Times are EDT (four hours behind GMT).
3. **request** given in quotes.
4. **HTTP reply code**.
5. **bytes in the reply**.

Measurement

The logs were collected from 23:53:25 EDT on Tuesday, August 29 1995 through 23:53:07 on Wednesday, August 30 1995, a total of 24 hours. There were 47,748 total requests, 46,014 **GET** requests, 1,622 **POST** requests, 107 **HEAD** requests, and 6 invalid requests. Timestamps have one-second precision. The WWW server software used was not recorded.

Privacy

The logs fully preserve the originating host and HTTP request. Please do not however attempt any analysis beyond general traffic patterns.

Acknowledgements

The logs were collected by Laura Bottomley (*laurab@ee.duke.edu*) of Duke University. Please include a corresponding acknowledgement in publications analyzing the logs.

Restrictions

The trace may be freely redistributed.

Assignment Tasks

1. Write a script that imports the access log file and creates a **new** file that holds the log data structured as a JSON-Array. (See the example at the bottom of this page)

2. Create one or more HTML- and Javascript-Files that read the JSON-File and render the following analysis graphically as charts:
    - Requests per minute over the entire time span
    - Distribution of HTTP methods (GET, POST, HEAD,...)
    - Distribution of HTTP answer codes (200, 404, 302,...)
    - Distribution of the size of the answer of all requests with code 200 and size < 1000B

Advice

- Please make sure ALL log records are being imported by your importer script.
- The data is from 1995 and might contain uncommon characters - clean them in the first part.
- The folder “template” can be used as the basis for the second part.
- For each analysis, pick the chart type that in your opinion makes most sense to describe the data (lines, bars, pie charts,...).
- Your application does **not** have to be supported by a variety of browsers - just state in which browser it runs best.

Example

Example for the JSON output file from part 1 (we added line breaks and indentations for the sake of readability)


```
[
 {
  "host": "141.243.1.172",
  "datetime": {
   "day": "29",
   "hour": "23",
   "minute": "53",
   "second": "25"
  },
  "request": {
   "method": "GET",
   "url": "/Software.html",
   "protocol": "HTTP",
   "protocol_version": "1.0"
  },
  "response_code": "200",
  "document_size": "1497"
 },
 ...more data sets...
]
```