How to instructions:

1. **tar -xzvf http_traffic_monitor.tar.gz**
2. **cd http_traffic_monitor**
3. **docker build . -t ssriram1978/http_traffic_monitor:latest**
4. **docker run -itd ssriram1978/http_traffic_monitor:latest**
   (or)
5. **docker swarm init**
6. **docker stack deploy -c docker-compose.yml http_traffic_monitor**
   (or)
   docker-compose up -f docker-compose-no-swarm.yml -d

Explanation:
This is a Python package.
The package name is http_traffic_monitor.
It has 3 sub components..

1. HTTP Log Analyzer.
   a. The main entry point is http_log_analyzer/http_log_analyzer.py
   b. This class creates two threads.
      Thread 1. Instantiates web server on port 8080 for displaying
         a. Current 10 second statistics of the HTTP traffic.
         b. Historic statistics of alarms and events.
      Thread 2. a. Used for computing two minute average of all the 10 second hits.
         The sampling rate is 120//10 = 12 samples.
         b. This thread raises an alarm if it finds that the 2 min average of hits exceeds the threshold limit.

2. HTTP Web server.
   class TestHTTPTrafficMonitor.
   Instantiates web server on port 8080 for displaying
      a. Current 10 second statistics of the HTTP traffic.
      b. Historic statistics of alarms and events.
   This class will handles any incoming request from the browser.

3. Unit test.
   class TestHTTPTrafficMonitor.
   This is used to test the alarm generation and alarm clearing procedure.

SRIRAM SRIDHAR's comments:

- Consume an actively written-to w3c-formatted HTTP access log (https://www.w3.org/Daemon/User/Config/Logging.html). It should default to reading /tmp/access.log and be overrideable

  Sriram: The package is zipped into http_traffic_monitor.tar.gz file.

Example log lines:

```
127.0.0.1 - james [09/May/2018:16:00:39 +0000] "GET /report HTTP/1.0" 200 123

127.0.0.1 - jill [09/May/2018:16:00:41 +0000] "GET /api/user HTTP/1.0" 200 234

127.0.0.1 - frank [09/May/2018:16:00:42 +0000] "POST /api/user HTTP/1.0" 200 34

127.0.0.1 - mary [09/May/2018:16:00:42 +0000] "POST /api/user HTTP/1.0" 503 12
```

Sriram: I've used these lines for unit testing the software package.

- Display stats every 10s about the traffic during those 10s: the sections of the web site with the most hits, as well as interesting summary statistics on the traffic as a whole. A section is defined as being what's before the second '/' in the resource section of the log line. For example, the section for "/pages/create" is "/pages"
  Sriram: The HTTP verb is first used as the key to identify if the stat is W3C log. It would have been much better to use W3C parser in the code. But for now, I've used this logic to extract section from the log.

- Make sure a user can keep the app running and monitor the log file continuously
  Sriram: The web server thread and the event logger threads are set to run continuously. The docker-compose.yml has the following added to it to make it automatically restart as a docker swarm service deployment when the container crashes.
  ```
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
  ```

- Whenever total traffic for the past 2 minutes exceeds a certain number on average, add a message saying that "High traffic generated an alert - hits = {value}, triggered at {time}". The default threshold should be 10 requests per second, and should be overridable.
  Sriram: The alert log shall be visible via the HTTP web server client by visiting localhost:8085 (default port) number
  http://localhost:8085/
  ```
  Historic Stats.
  ^^^^^^^^^^^^^^^^^^^.
  High traffic generated an alert - hits = 1970, triggered at Fri Dec 21 22:12:04 2018
  Clearing the alert at Fri Dec 21 22:14:04 2018 as average hit count 0 is less than the threshold 10
  --------------------.
  Current Running Stats.
  ^^^^^^^^^^^^^^^^^^^^^^.
  10 second aggregate of resource_section to Hit count.
  ****************************************************.
  Resource_section ==> Hits .
  ********************************************************.
  Total sum of all the hits = 0.There has been a constant flow of web traffic of 0 hits every 10 seconds.
  --------------------.
  ```

- Whenever the total traffic drops again below that value on average for the past 2 minutes, add another message detailing when the alert recovered.
  Sriram: As shown above, when you visit http://localhost:8085/, you should be able to view the alert Historic log which is persistent.

- Make sure all messages showing when alerting thresholds are crossed remain visible on the page for historical reasons.
  Sriram: Yes, the historic alert log is logged to a seperate file which gets displayed at http://localhost:8085/ and stays on.

- Write a test for the alerting logic.
- **Sriram: http_traffic_monitor/unit_test/test_http_traffic_monitor.py has the unit test to test the alerting logic and the output snippet of http://localhost:8085/ is displayed in the above said section.**

- Explain how you'd improve on this application design.
  Sriram:
  1. **Instead of doing multi threading, I would split the thought into docker containers and scale them horizontally. In other words, I would make use of REDIS Database to store the 10 second stats and the 120 second alarm stats.**
  2. **I would have the web server directly pull the stats from the REDIS database to display the stats.**
  3. **I would have a load balancer (NGINX as reverse proxy) to distribute the HTTP GET request to a docker or kubernetes swarm of web servers to display the result.**
  4. **I would use portainer kind of docker swarm orchestration interface to manage the docker swarm.**
  5. **I would use a well defined W3C parser to parse the logs to avoid any protocol glitches and mis-interpretation.**
  6. **I would volume mount localhost /var/log/access.log into the docker container and feed the W3C log from the localhost into the container.**
  7. **I would use ElasticSearch Logstash Kibana stack to display the graphical plot of the 10 second statistics and the historical alarms.**

- If you have access to a linux docker environment, we'd love to be able to docker build and run your project! If you don't though, don't sweat it. As an example for a solution based on python 3:

```
FROM python:3

RUN touch /var/log/access.log  # since the program will read this by default

WORKDIR /usr/src

ADD . /usr/src

ENTRYPOINT ["python", "main.py"]# this is an example for a python program, pick the language of your
choice
```

- and we'll have something else write to that log file.

Sriram: Here are the steps that I followed to run this Docker image.

7. **tar -xzvf http_traffic_monitor.tar.gz**
8. **cd http_traffic_monitor**
9. **docker build . -t ssriram1978/http_traffic_monitor:latest**
10. **docker run -itd ssriram1978/http_traffic_monitor:latest**
    (or)
11. **docker swarm init**
12. **docker stack deploy -c docker-compose.yml http_traffic_monitor**
    (or)
    docker-compose up -f docker-compose-no-swarm.yml -d

Snippet of the output that gets displayed when you visit http://localhost:8085/

```
Historic Stats.
^^^^^^^^^^^^^^^^^^^^.
High traffic generated an alert - hits = 1970, triggered at Fri Dec 21 22:12:04 2018
Clearing the alert at Fri Dec 21 22:14:04 2018 as average hit count 0 is less than the threshold 10
--------------------.
Current Running Stats.
^^^^^^^^^^^^^^^^^^^^^^^.
10 second aggregate of resource_section to Hit count.
*******************************************************.
Resource_section ==> Hits .
***********************************************************.
Total sum of all the hits = 0.There has been a constant flow of web traffic of 0 hits every 10 seconds.
--------------------.
```