Mariam Mostafa Amin Mostafa
2205084

Log File Analysis with Bash Script

Using Kali Linux

Step 1 (Setup):

- We will have to setup **kali linux** on our **Windows** to be able to achieve our task requirements
- We will have to download **Kali Linux** first from **Microsoft Store**

- After downloading **Kali Linux** from **Microsoft Store** we will then have to install and initialize it on our system as a subsystem with the Hypervisor Platform

- Then open **PowerShell** as an administrator and install **KaliLinux**

```
PS C:\WINDOWS\system32> wsl --install -d kali-linux
Downloading: Kali Linux Rolling
[===                          6.2%                           ]
```

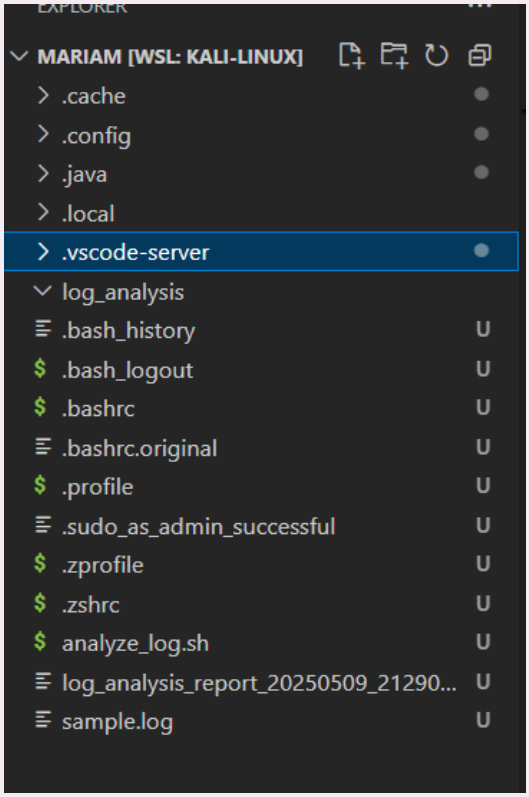
- Then set **WSL** as default

```
PS C:\WINDOWS\system32> wsl --set-default-version 2
For information on key differences with WSL 2 please visit https://aka.ms/wsl2
The operation completed successfully.
PS C:\WINDOWS\system32>
```

- And then we can update the system too for full functionality in the **Kali** terminal

```
┌──(mariam㉿marioma)-[~]
└─$ sudo apt update && sudo apt upgrade -y
[sudo] password for mariam:
```

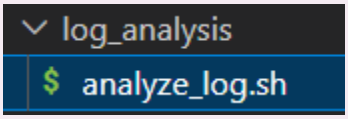
- Then we open **Vs Code** to install the **WSL** Extensions
- In the **Vs Code** we can open WSL with **Kali Linux** to start the terminal and the Virtual Environment into **Vs Code**



Step 2 (Creating the code & Explanation):

- First we will need to create a directory to obtain our task files and create a file named **analyze_log.sh**

```
┌──(mariam㉿marioma)-[~]
└─$ mkdir ~/log_analysis && cd ~/log_analysis
touch analyze_log.sh
chmod +x analyze_log.sh

┌──(mariam㉿marioma)-[~/log_analysis]
└─$
```



- After that we can create our own code which I will explain now:

We First before all have to check if the **LogFile** exists or not

```
1  #!/bin/bash
2
3  if [ $# -eq 0 ]; then
4      echo "Usage: $0 <log_file>"
5      exit 1
6  fi
7
8  LOG_FILE=$1
```

Then We Output the report FileName

```
REPORT_FILE="log_analysis_report_$(date +%Y%m%d).txt"
```

Afterwards we count all the required **Requests,Gets & Posts**
wc -l : count total lines (Requests)
grep -c (GET): Count occurrences of **GET**
grep -c (POST): Count occurrences of **POST**

```
total_requests=$(wc -l < "$LOG_FILE")
get_requests=$(grep -c 'GET' "$LOG_FILE")
post_requests=$(grep -c 'POST' "$LOG_FILE")
```

Then we identifies unique IPs and their request methods

```
unique_ips=$(awk '{print $1}' "$LOG_FILE" | sort -u | wc -l)
ip_method_counts=$(awk '{print $1,$6}' "$LOG_FILE" | sort | uniq -c | sort -nr)
```

We assign a failure analysis to identify failed requests

```
failed_requests=$(awk '$9 ~ /^[45][0-9][0-9]$/' "$LOG_FILE" | wc -l)
failure_percentage=$(awk "BEGIN {printf \"%.2f\", ($failed_requests/$total_requests)*100}")
```

We Also assigned a line to identify the top user by requests

```
top_ip=$(awk '{print $1}' "$LOG_FILE" | sort | uniq -c | sort -nr | head -n 1)
```

Then we calculate the daily traffic pattern

```
daily_requests=$(awk -F'[' '{print $2}' "$LOG_FILE" | awk '{print $1}' | sort | uniq -c)
total_days=$(echo "$daily_requests" | wc -l)
avg_daily=$(awk "BEGIN {printf \"%.2f\", $total_requests/$total_days}")
```

This line here calculates which days has the most failed requests

```
failure_days=$(awk '$9 ~ /^[45][0-9][0-9]$/' "$LOG_FILE" | awk -F'[' '{print $2}' | awk '{print $1}' | sort | uniq -c | sort -nr)
```

Then I added the additional requirements in the task as :

1- Hourly Request Distribution : Calculate the number of requests made each hour of the day

2- Status Code Frequency Breakdown : Provides a status code breakdown and their frequency

3- Most Active Users : Identify which used GET or POST requests the most

4- Pattern In Failure Requests : Identify if there are patterns in the failures

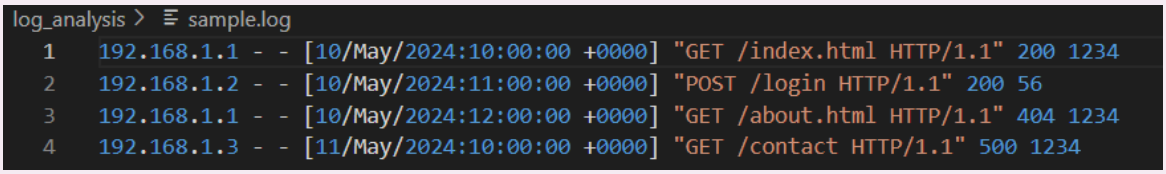
```
hourly_requests=$(awk -F'[:[]' '{print $3}' "$LOG_FILE" | awk -F: '{print $1}' | sort | uniq -c)
status_codes=$(awk '{print $9}' "$LOG_FILE" | sort | uniq -c | sort -nr)
top_get_user=$(awk '$6 == "\"GET" {print $1}' "$LOG_FILE" | sort | uniq -c | sort -nr | head -n 1)
top_post_user=$(awk '$6 == "\"POST" {print $1}' "$LOG_FILE" | sort | uniq -c | sort -nr | head -n 1)
failure_hours=$(awk '$9 ~ /^[45][0-9][0-9]$/' "$LOG_FILE" | awk -F'[:[]' '{print $3}' | awk -F: '{print $1}' | sort | uniq -c | sort
```

After we finished we can now generate the report

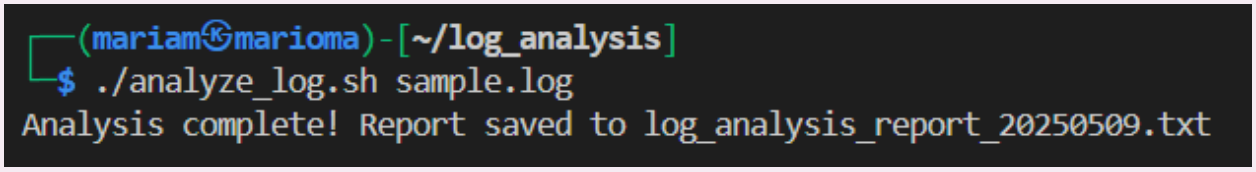
```
{
    echo "=== LOG ANALYSIS REPORT ==="
    echo "Generated on: $(date)"
    echo "Analyzed file: $LOG_FILE"
    echo ""

    echo "1. REQUEST COUNTS"
    echo "   Total requests: $total_requests"
    echo "   GET requests: $get_requests"
    echo "   POST requests: $post_requests"
    echo ""

    echo "2. UNIQUE IP ADDRESSES"
    echo "   Total unique IPs: $unique_ips"
    echo "   Requests per IP and method:"
    echo "$ip_method_counts"
    echo ""

    echo "3. FAILURE REQUESTS"
    echo "   Failed requests (4xx/5xx): $failed_requests"
    echo "   Failure percentage: $failure_percentage%"
    echo ""
```

Step 3 (Creating a sample and generating the report):

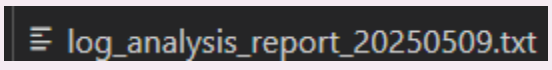
- Before all we must create a sample to test on
- I created a file named **sample.log** and inserted 1000 Requests in it



- Then we run our command **./analyze_log.sh sample.log** to run the bash script



As we see our report has been created successfully



Step 4 (Analysing the generated report):

Here it counts the total requests as I added 1000 requests in my **sample.log** so it should view 1000 with how many **GETs** and **POSTs**

```
1. REQUEST COUNTS
   Total requests: 1000
   GET requests: 491
   POST requests: 509
```

Then it counts all the unique IPs

```
2. UNIQUE IP ADDRESSES
   Total unique IPs: 50
   Requests per IP and method:
     18 192.168.1.4 "GET
     17 192.168.1.37 "POST
     17 192.168.1.33 "POST
     16 192.168.1.40 "GET
```

And Here I faced 423 issues in the requests which are server errors

```
3. FAILURE REQUESTS
   Failed requests (4xx/5xx): 423
   Failure percentage: 42.30%
```

Viewing the most active user(IP)

```
4. TOP USER
   Most active IP: 28 192.168.1.37
```

Here it shows the Daily Requests counts

```
5. DAILY REQUEST AVERAGES
   Average requests per day: 1.00
   Daily request counts:
     1 10/May/2024:00:03:48
     1 10/May/2024:00:26:41
     1 10/May/2024:01:01:26
     1 10/May/2024:01:08:06
```

Showing too the days which requests failed in

```
6. FAILURE ANALYSIS
   Days with most failures:
      1 19/May/2024:23:22:44
      1 19/May/2024:23:13:41
      1 19/May/2024:23:12:48
```

And for the additional metrics we added it shows:

1- Hourly Request Distribution

2- Status Code Breakdown

3- Most Active Users By Method (GET/POST)

4- Failure Patterns By Hour

```
ADDITIONAL METRICS
a. Hourly request distribution:
     38 00
     49 01
     36 02
     47 03
     43 04
```

```
b. Status code breakdown:
    577 200
    149 500
    139 404
    135 403

c. Most active users by method:
   Top GET user:      18 192.168.1.4
   Top POST user:      17 192.168.1.37

d. Failure patterns by hour:
     25 06
     23 04
     22 20
     21 22
```

We made recommendations on how to fix and reduce the number of failures and analysis on which days, hours or users facing problems and suggested security recommendations to improve the security and check the errors

```
ANALYSIS SUGGESTIONS
1. Failure reduction:
   - Investigate top error codes: 200,500,404,
   - Focus on peak failure hours: 06:00 (25 failures),04:00 (23 failures),20:00 (22 failures),

2. Performance optimization:
   - Highest traffic hour: 20:00 (53 requests)
   - Consider load balancing or caching during peak times

3. Security recommendations:
   - Review activity from top POST user (     17 192.168.1.37)
   - Check for brute force patterns (many rapid failures from single IP)
   - Monitor suspicious user agents or unusual request patterns
```