ADS LAB03 PIPELINE

Auteur : Bastien Pillonel, Loïc Brasey

Date : 04.03.2024

TASK 1

1/

a) redirectionne stdout dans le fichier

b) redirectionne stderr dans le fichier

c) redirectionne stdout dans le fichier puis duplique stderr dans la stdout les deux se retrouve dans le fichier

d) redirectionne duplique stderr dans stdout et redirige stdout dans le fichier les deux se retrouve dans le fichier

e) redirectionne stdout et stderr dans le fichier

2/

a) Cherche toutes les occurence du token edit (non sensible à la casse) dans le fichier README et affiche chaque ligne

b) N'affiche rien car lorsque l'on redirige stderr sur stdout les caractères ont seront mélanger le patern devient imprédictible et la chance de se retrouver avec les 5 e à la suite est très réduite.

c) Ici la sortie standard étant redirigé vers /dev/null le pipe ne prend en compte que stderr qui sortait eeeee

3/

a)

```
sudo ls -a -R /home/ > /tmp/homefileslist
```

b)

```
ls /home | grep -E '\.txt$|\.md$|\.pdf$' > /tmp/homedocumentslist
```

TASK 2

How many log entries are in the file?

```
wc -l < ads_website.log
```

2781

How many accesses were successful (server sends back a status of 200) and how many had an error of "Not Found" (status 404)?

```
echo "  Requests Succes:     $(cut -d$'\t' -f10 < ads_website.log | grep 200 | wc -l)"
```

```
echo "  Requests Not Found : $(cut -d$'\t' -f10 < ads_website.log | grep 404 | wc -l)"
```

Requests Succes: 1610

Requests Not Found : 21

What are the URIs that generated a Not Found response? Be careful in specifying the correct search criteria: avoid selecting lines that happen to have the character sequence 404 in the URI.

```
cut -d$'\t' -f9,10 < ads_website.log | grep 404$ | cut -d$'\t' -f1 | cut -d' ' -f2
```

/heigvd-ads?website

/heigvd-ads?lifecycle

/heigvd-ads?cors

/heigvd-ads?policy

/heigvd-ads?website

/heigvd-ads?lifecycle

/heigvd-ads?policy

/heigvd-ads?cors

/heigvd-ads?cors

/heigvd-ads?policy

/heigvd-ads?website

/heigvd-ads?lifecycle

/heigvd-ads?lifecycle

/heigvd-ads?cors

/heigvd-ads?cors

/heigvd-ads?policy

/heigvd-ads?lifecycle

/heigvd-ads?policy

/heigvd-ads?policy

/heigvd-ads?policy

/heigvd-ads?cors

How many different days are there in the log file on which requests were made?

```
cut -d$'\t' -f3 < ads_website.log | egrep '[[:digit:]]{2,2}/[[:alpha:]]{3,3}/[[:digit:]]{4,4
```

21

How many accesses were there on 4th March 2021?

```
cut -d$'\t' -f3 < ads_website.log | grep '14/Mar/2021' -c
```

62

Which are the three days with the most accesses? Hint: Create first a pipeline that produces a list of dates preceded by the count of log entries on that date.

```
cut -d$'\t' -f3 < ads_website.log | egrep '[[:digit:]]{2,2}/[[:alpha:]]{3,3}/[[:digit:]]{4,4
```

13/Mar/2021

06/Mar/2021

04/Mar/2021

Which is the user agent string with the most accesses?

```
cut -d$'\t' -f17 < ads_website.log | uniq -c | sort -k1,1nr | head -n1 | grep '".*"' -o
```

"Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0"

echo "If a web site is very popular and accessed by many people the user agent strings appearing in the server's log can be used to estimate the relative market share of the users' computers and operating systems. How many accesses were done from browsers that declare that they are running on Windows, Linux and Mac OS X (use three commands)?"

```
echo "windows : $(cut -d$'\t' -f17 < ads_website.log | grep -i 'windows' -c)"
```

```
echo "linux : $(cut -d$'\t' -f17 < ads_website.log | grep -i 'linux' -c)"
```

```
echo "mac os x : $(cut -d$'\t' -f17 < ads_website.log | grep -i 'mac os x' -c)"
```

windows : 1751

linux : 180

mac os x : 693

Read the documentation for the tee command. Repeat the analysis of the previous question for browsers running on Windows and insert tee into the pipeline such that the user agent strings (including repeats) are written to a file for further analysis (the filename should be useragents.txt)

```
cut -d$'\t' -f17 < ads_website.log | grep -i 'windows' | tee useragents.txt > /dev/null
```

Why is the file access.log difficult to analyse, consider for example the analysis of question 7, with the commands you have seen so far?

Because we aren't able to easily differentiate separation whitespace and and a whitespace within a field