TP - Powershell

Votre mission

En utilisant les fichiers de logs fournis

- Lister les requêtes les plus fréquentes
- Blacklister les IP de requêtes malicieuses
- Générer une liste des mise online/offline du serveur

Ce TP est divisé en étapes digestes, utilisez un maximum les outils d'aides pour tous les éléments que vous ne savez pas (*encore*) faire

Validez chaque étape avant de commencer l'étape d'après

Pour vous aider:

- `Get-Help <commande>`
- Stack Overflow
- Microsoft Learn
- Google
- Moi

1 - Récupérer les arguments

Ajouter **3** paramètres *nommés* au script

Créer une condition pour s'assurer que tous les arguments ont été renseignés

Si non, lève une erreur avec un message affichant l'usage correct:

```
Usage: $PSCommandPath -actix_log <actix_log> -nginx_access <nginx_access_log> -nginx_error <nginx_error_log>
```

2 - Initialiser les variables

Pour chaque argument, initialiser une variable avec le *chemin absolu* du fichier passé en paramètre

- **ACTIX_LOG** récupère le chemin absolu de l'argument **actix_log**
- **NGINX_ACCESS_LOG** récupère le chemin absolu de l'argument **nginx_access**
- **NGINX_ERROR_LOG** récupère le chemin absolu de l'argument **nginx_error**

3 - Les requêtes les plus fréquentes

Dans **ACTIX_LOG**, récupérer toutes les requêtes `GET` avec un status `200`

Parmis ces requêtes, n'afficher que le chemin (ex: `/index.html`).
Note: Il s'agit du 7e élément de la ligne de log

Comptez l'occurence de chaque ligne unique

Trier le résultat par le nombre d'occurence

Enregistrer ce résultat dans une variable **MOST_SERVED**

4 - Filtrer les requêtes

Boucler à travers une liste d'extensions à filtrer: *png, css, ico et js*

Pour chaque filtre, retirer de **MOST_SERVED** les lignes possédant cette extension
Attention à ne pas retirer d'autres requêtes ! (ex: /p **ico** n/biere)

Écraser la valeur de **MOST_SERVED** avec cette version filtrée

5 - Ne garder que les meilleurs

Parmis les requêtes restantes, ne garder que celles qui ont des occurences `> 10`

Formatter la ligne en `/<path> : <count>`

Enregistrer le résultat dans un fichier `most_served.txt`

6 - IP Blacklist

Dans **NGINX_ACCESS_LOG**, récupérer les requêtes aux endpoints: (case insensitive)
- admin
- debug
- login
- .git

Ajouter les IP de toutes ces requêtes dans une variable **IP_BLACKLISTS**

7 - Encore plus de blacklist

Ajoutez à `$IP_BLACKLISTS` les IP des requêtes ayant une méthode qui ne soit pas:

- GET
- POST
- HEAD

Trier les IP, retirer tous les doublons

Écrire la liste dans un fichier `ip_blacklist.txt`

8 - Repérer les downtimes

Downtime = Le moment `nginx` n'arrive pas à dialoguer avec `actix_web`

Dans le fichier **NGINX_ERROR_LOG**, récupérer les erreurs du type:
`connect() failed (111: Unknown error) while connecting to upstream`

Récupérer la date et l'heure de chacune de ces erreurs

Formater les données au format: `<date> <heure> DOWN`

Enregistrer le résultat dans une variable **DOWNTIME**

9 - Récupérer les uptimes

À partir des regex:

```
DATE_REGEX="([0-9]{4})-([0-9]{2})-([0-9]{2})"
TIME_REGEX="([0-9]{2}):([0-9]{2}):([0-9]{2})"
```

Sélectionner dans les lignes de **ACTIX_LOG** la date et l'heure

Utiliser `$VARIABLE = $_.Matches.Groups[N].Value` pour récupérer les *capture group*

Formatter la ligne en `YYYY/MM/DD HH:MM:SS UP`

Enregistrer le tout dans une variable **UPTIME**

10 - Gros travail ForEach-Object

Créer une variable `$state = "DOWN"`

Trier les objets par date, puis par heure
Note: `Sort-Object { <critère 1> },{ <critère 2> }`

Puis créer un algorithme avec `ForEach-Object` pour chaque ligne de `$DOWNTIME + $UPTIME` :

- Si la ligne indique “UP” et l’état précédent était “DOWN”:
  - Afficher la date, l’heure et le message “UP”
  - L’état est maintenant “UP”
- Si la ligne indique “DOWN” et l’état précédent était “UP”:
  - Afficher la date, l’heure et le message “DOWN”
  - L’état est maintenant “DOWN”