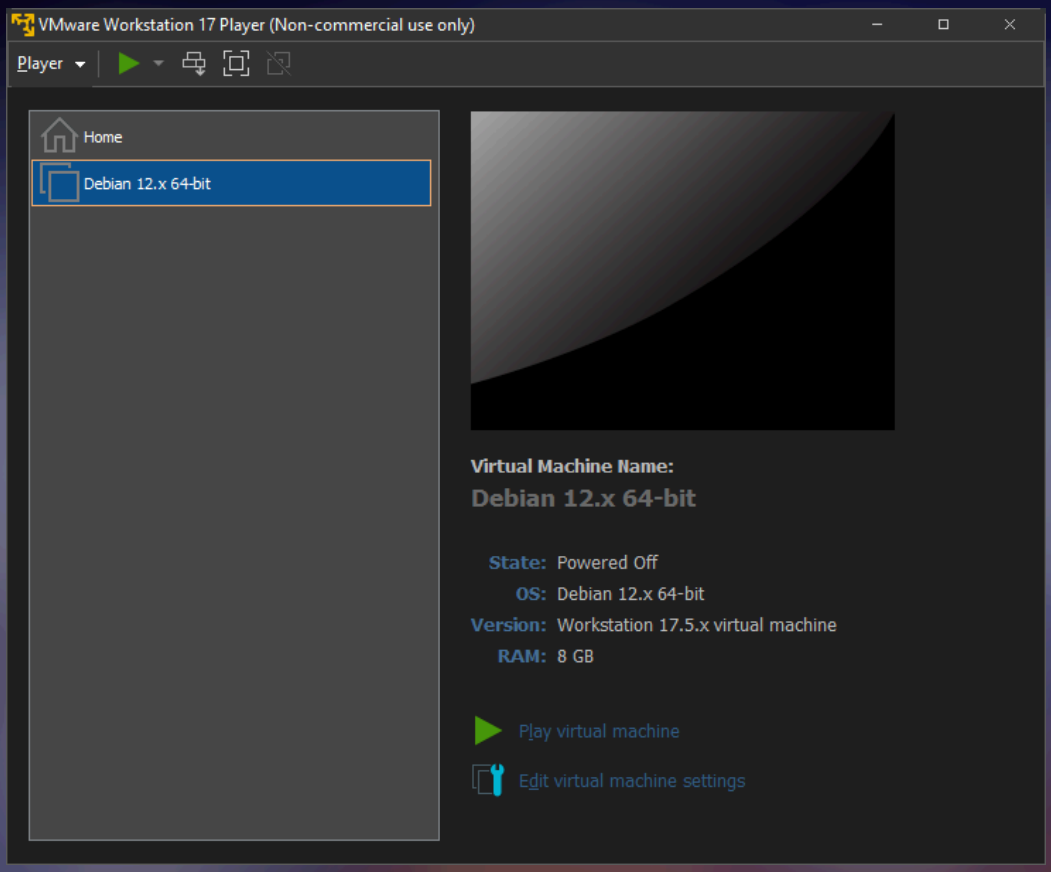
SYSTÈME, SCRIPTS ET SÉCURITÉ

SOMMAIRE :

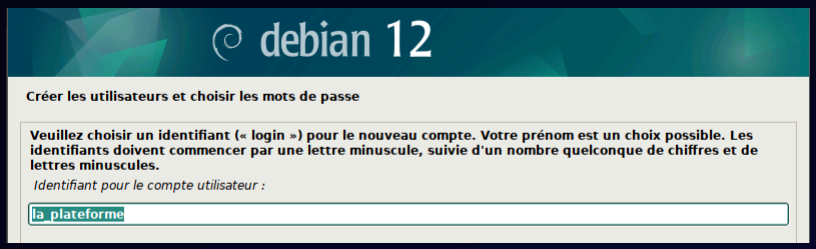
CRÉATION DE LA VM DEBIAN

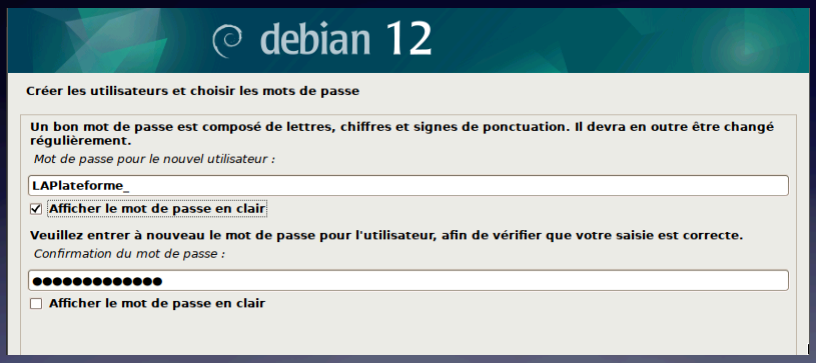
Pour commencer, nous allons choisir un hyperviseur pour créer notre machine virtuelle Debian avec interface graphique. Dans ce projet, nous avons utilisé VMWare car nous l'avons déjà utilisé auparavant et il répond aux exigences de notre projet.

Ensuite , nous créons une nouvelle machine virtuelle dans notre hyperviseur. Nous configurons la machine virtuelle en fonction de nos préférences, en allouant suffisamment de mémoire vive et d'espace disque.



Ensuite, nous démarrons notre machine virtuelle et nous installons le système d'exploitation Debian en utilisant l'image que nous avons téléchargée précédemment. Lors de l'installation, nous créons un utilisateur avec le nom "La_Plateforme" et le mot de passe "LAPlateforme_" comme demandé dans la consigne. Nous configurons également l'interface graphique pour qu'elle démarre automatiquement lors du démarrage de la machine virtuelle.





En ce qui concerne la connexion internet, notre hyperviseur VMWare la configure automatiquement. Cependant, nous avons vérifié que la connexion fonctionnait correctement en utilisant la commande suivante dans le terminal de notre machine virtuelle

```
laplateforme_@laplateforme:~$ ping google.com
PING google.com (142.251.37.206) 56(84) bytes of data.
64 bytes from mrs09s15-in-f14.1e100.net (142.251.37.206): icmp_seq=1 ttl=128 time=5.95 ms
64 bytes from mrs09s15-in-f14.1e100.net (142.251.37.206): icmp_seq=2 ttl=128 time=4.71 ms
```

COMMANDES DE RECHERCHES AVANCÉES

Pour commencer, nous allons utiliser les commandes de terminal pour créer cinq fichiers texte nommés "mon_texte.txt" contenant le texte "Que la force soit avec toi." et les répartir dans les répertoires "Bureau", "Documents", "Téléchargements", "Vidéos" et "Images".

Tout d'abord, nous allons utiliser la commande "echo" pour créer les fichiers texte et ajouter le texte demandé. Pour ce faire, nous utilisons les commandes suivantes :

```
laplateforme_@laplateforme:~$ echo "Que la force soit avec toi" > Bureau/mon_texte.txt
laplateforme_@laplateforme:~$ echo "Que la force soit avec toi" > Documents/mon_texte.txt
laplateforme_@laplateforme:~$ echo "Que la force soit avec toi" > Téléchargements/mon_texte.txt
laplateforme_@laplateforme:~$ echo "Que la force soit avec toi" > Vidéos/mon_texte.txt
laplateforme_@laplateforme:~$ echo "Que la force soit avec toi" > Images/mon_texte.txt
```

Ces commandes créent les fichiers "mon_texte.txt" dans chaque répertoire et ajoutent le texte "Que la force soit avec toi." à l'intérieur.

Ensuite, nous allons utiliser la commande "grep" pour localiser les fichiers contenant le mot "force" à partir du répertoire de notre session. Pour ce faire, nous utilisons la commande suivante :

```
laplateforme_@laplateforme:~$ grep -r "force"
Vidéos/mon_texte.txt:Que la force soit avec toi
grep: .cache/tracker3/files/http%3A%2F%2Ftracker.
grep: .cache/tracker3/files/meta.db-wal : fichier
.cache/gnome-software/odrs/ratings.json:    "io.g
.cache/gnome-software/odrs/ratings.json:    "io.g
.cache/gnome-software/odrs/ratings.json:    "io.s
.cache/gnome-software/odrs/ratings.json:    "io.s
.cache/gnome-software/odrs/ratings.json:    "io.s
grep: .cache/gnome-software/appstream/components.
Téléchargements/mon_texte.txt:Que la force soit a
Documents/mon_texte.txt:Que la force soit avec to
.bashrc:#force_color_prompt=yes
.bashrc:if [ -n "$force_color_prompt" ]; then
.bashrc:unset color_prompt force_color_prompt
Images/mon_texte.txt:Que la force soit avec toi
Bureau/mon_texte.txt:Que la force soit avec toi
```

COMPRESSION ET DÉCOMPRESSION DE FICHIERS

Maintenant que nous avons créé cinq fichiers texte nommés "mon_texte.txt" contenant le texte "Que la force soit avec toi." et les avons répartis dans les répertoires "Bureau", "Documents", "Téléchargement", "Vidéos" et "Images", nous allons créer un répertoire nommé "Plateforme" dans le dossier "Documents" de notre session et ajouter le fichier "mon_texte.txt" précédemment créé.

Pour ce faire, nous allons utiliser la commande "mkdir" pour créer le répertoire "Plateforme" dans le dossier "Documents" et, nous allons copier le fichier "mon_texte.txt" dans le nouveau répertoire "Plateforme" à l'aide de la commande "cp" :

```
laplateforme_@laplateforme:~/Documents$ mkdir Plateforme
laplateforme_@laplateforme:~/Documents$ cp mon_texte.txt Plateforme
```

Maintenant que nous avons copié le fichier "mon_texte.txt" dans le répertoire "Plateforme", nous allons dupliquer ce fichier quatre fois dans le même répertoire, formant ainsi un total de cinq fichiers dans le répertoire "Plateforme".

Pour ce faire, nous allons utiliser la commande "cp" avec l'option "-n" pour éviter d'écraser les fichiers existants :

```
laplateforme_@laplateforme:~/Documents/Plateforme$ ls
mon_texte.txt
laplateforme_@laplateforme:~/Documents/Plateforme$ cp -n mon_texte.txt mon_texte1.txt
laplateforme_@laplateforme:~/Documents/Plateforme$ cp -n mon_texte.txt mon_texte2.txt
laplateforme_@laplateforme:~/Documents/Plateforme$ cp -n mon_texte.txt mon_texte3.txt
laplateforme_@laplateforme:~/Documents/Plateforme$ cp -n mon_texte.txt mon_texte4.txt
laplateforme_@laplateforme:~/Documents/Plateforme$ ls
mon_texte1.txt  mon_texte2.txt  mon_texte3.txt  mon_texte4.txt  mon_texte.txt
```

Nous avons maintenant cinq fichiers texte dans le répertoire "Plateforme". Nous allons maintenant archiver le répertoire "Plateforme" en utilisant les commandes "tar" et "gzip".

Pour explorer différentes options de compression, nous allons créer trois archives avec différents niveaux de compression

```
laplateforme_@laplateforme:~/Documents/Plateforme$ tar -czvf ../Plateforme.tar.gz
./
./mon_texte3.txt
./mon_texte1.txt
./mon_texte2.txt
./mon_texte4.txt
./mon_texte.txt
```

```
laplateforme_@laplateforme:~/Documents/Plateforme$ tar -cjvf ../Plateforme.tar.bz2 .
./
./mon_texte3.txt
./mon_texte1.txt
./mon_texte2.txt
./mon_texte4.txt
./mon_texte.txt
laplateforme_@laplateforme:~/Documents/Plateforme$ tar -cJvf ../Plateforme.tar.xz .
./
./mon_texte3.txt
./mon_texte1.txt
./mon_texte2.txt
./mon_texte4.txt
./mon_texte.txt
```

```
laplateforme_@laplateforme:~/Documents$ ls
mon_texte.txt  Plateforme  Plateforme.tar.bz2  Plateforme.tar.gz  Plateforme.tar.xz
laplateforme_@laplateforme:~/Documents$
```

Nous avons maintenant trois archives compressées du répertoire "Plateforme". Pour décompresser ces archives, nous allons utiliser les commandes appropriées pour chaque type de compression :

```
laplateforme_@laplateforme:~$ tar -xzvf Documents/Plateforme.tar.gz -C Documents/
./
./mon_texte3.txt
./mon_texte1.txt
./mon_texte2.txt
./mon_texte4.txt
./mon_texte.txt
laplateforme_@laplateforme:~$ tar -xjvf Documents/Plateforme.tar.bz2 -C Documents/
./
./mon_texte3.txt
./mon_texte1.txt
./mon_texte2.txt
./mon_texte4.txt
./mon_texte.txt
laplateforme_@laplateforme:~$ tar -xJvf Documents/Plateforme.tar.xz -C Documents/
./
./mon_texte3.txt
./mon_texte1.txt
./mon_texte2.txt
./mon_texte4.txt
./mon_texte.txt
```

MANIPULATION DE TEXTE

Tout d'abord, nous créons un script Python nommé **spy.py** contenant le code suivant :

```
  GNU nano 7.2                                spy.py
import csv

donnees = [
        ["Jean", 25, "Paris"],
        ["Marie", 30, "Lyon"],
        ["Pierre", 22, "Marseille"],
        ["Sophie", 35, "Toulouse"]
]

cible_csv = "donnees.csv"

with open(cible_csv, mode='w', newline='') as fichier_csv:
        writer = csv.writer(fichier_csv)
        writer.writerow(['Nom', 'Age', 'Ville'])
        writer.writerows(donnees)

print("Les donnes sont bien dans le csv")



                [ Lecture de 17 lignes ]
^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^T Exécuter   ^C Emplacement
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^J Justifier  ^/ Aller ligne
```

Ensuite, nous avons utilisé la commande **awk** avec l'option **-F** pour spécifier le séparateur de colonnes (dans ce cas, une virgule) et la commande **{print $3}** pour extraire la troisième colonne du fichier CSV, qui contient les noms des villes. Nous avons exécuté cette commande dans le répertoire où se trouve le fichier CSV et avons vérifié que les résultats correspondaient aux noms des villes dans le fichier. Enfin, nous avons utilisé la commande **cat** pour afficher le contenu du fichier CSV et vérifier que les données ont été correctement écrites.

```
laplateforme_@laplateforme:~/Documents$ nano spy.py
laplateforme_@laplateforme:~/Documents$ chmod +x spy.py
laplateforme_@laplateforme:~/Documents$ python3 spy.py
Les donnes sont bien dans le csv
laplateforme_@laplateforme:~/Documents$ awk -F',' '{print $3}' donnees.csv
Ville
Paris
Lyon
Marseille
Toulouse
laplateforme_@laplateforme:~/Documents$ cat donnees.csv
Nom,Age,Ville
Jean,25,Paris
Marie,30,Lyon
Pierre,22,Marseille
Sophie,35,Toulouse
```

GESTION DE PROCESSUS

```
laplateforme_@laplateforme:~$ ps ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:02 /sbin/init
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
      4 ?        I<     0:00 [rcu_par_gp]
      5 ?        I<     0:00 [slub_flushwq]
      6 ?        I<     0:00 [netns]
      8 ?        I<     0:00 [kworker/0:0H-events_highpri]
     10 ?        I<     0:00 [mm_percpu_wq]
     11 ?        I      0:00 [rcu_tasks_kthread]
     12 ?        I      0:00 [rcu_tasks_rude_kthread]
     13 ?        I      0:00 [rcu_tasks_trace_kthread]
     14 ?        S      0:14 [ksoftirqd/0]
     15 ?        I      0:05 [rcu_preempt]
```

Pour afficher tous les processus en cours d'exécution sur le système, y compris ceux qui n'ont pas de terminal de contrôle, vous pouvez utiliser la commande

ps ax

Cela affichera une liste de tous les processus, avec leur ID de processus (PID), le nom de l'utilisateur qui a lancé le processus, le pourcentage d'utilisation du processeur et de la mémoire, ainsi que la commande qui a lancé le processus.

```
13656 ?        Sl     0:07 /usr/lib/firefox-esr/firefox-esr
13718 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr -contentproc
13744 ?        Sl     0:02 /usr/lib/firefox-esr/firefox-esr -contentproc
13795 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr -contentproc
13840 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr -contentproc
13845 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr -contentproc
13877 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr -contentproc
13914 pts/0    R+     0:00 ps ax
aplateforme_@laplateforme:~$ kill 13656
```

Pour terminer un processus spécifique, vous pouvez utiliser la commande

kill

suivie de son ID de processus. Par exemple, pour terminer le processus avec l'ID 13656, vous pouvez utiliser la commande dans la capture d’écran.

```
 14089 ?        Sl     0:03 /usr/lib/firefox-esr/firefox-esr
 14154 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14167 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14212 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14250 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14253 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14255 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14307 pts/0    R+     0:00 ps ax
aplateforme_@laplateforme:~$ kill -9 14089
```

Si le processus ne se termine pas après quelques secondes, vous pouvez utiliser la commande **kill -9** pour forcer sa terminaison.
Cette commande envoie un signal SIGKILL au processus, ce qui le force à se terminer immédiatement.
Notez que cette méthode doit être utilisée avec précaution, car elle peut entraîner une perte de données ou des problèmes de stabilité du système.

```
 14450 ?        Sl     0:05 /usr/lib/firefox-esr/firefox-esr
 14514 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14530 ?        Sl     0:01 /usr/lib/firefox-esr/firefox-esr
 14572 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14614 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14617 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14651 ?        Sl     0:00 /usr/lib/firefox-esr/firefox-esr
 14680 ?        I      0:00 [kworker/0:2-events]
 14682 pts/0    R+     0:00 ps ax
laplateforme_@laplateforme:~$ pkill firefox
```

Vous pouvez également utiliser la commande **pkill** pour terminer un processus en fonction de son nom. Par exemple, pour terminer tous les processus nommés "firefox"
Cette commande envoie un signal SIGTERM à tous les processus nommés "firefox". Vous pouvez également utiliser la commande **pkill -9** pour forcer la terminaison de ces processus.

SURVEILLANCE DES RESSOURCES SYSTÈME

```
laplateforme_@laplateforme:~/Documents$ top

top - 10:26:22 up  2:08,  1 user,  load average: 0,15, 0,06, 0,01
Tâches: 159 total,   1 en cours, 158 en veille,   0 arrêté,   0 zombie
%Cpu(s):  1,9 ut,  1,6 sy,  0,0 ni, 95,8 id,  0,0 wa,  0,0 hi,  0,7 si,  0,0 st
MiB Mem :   3915,3 total,   2523,8 libr,   1034,7 util,    590,6 tamp/cache
MiB Éch :    975,0 total,    975,0 libr,      0,0 util.   2880,6 dispo Mem

    PID UTIL.     PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TEMPS+ COM.
   1241 laplate+  20   0 3788776 262468 130028 S   9,6   6,5   2:10.50 gnome-shell
   1983 laplate+  20   0  558092  51216  39112 S   1,3   1,3   0:11.91 gnome-terminal-
   2604 root      20   0       0      0      0 I   0,3   0,0   0:04.81 kworker/1:1-ev+
   2681 root      20   0       0      0      0 I   0,3   0,0   0:01.65 kworker/1:2-ev+
   2715 laplate+  20   0   11828   5512   3344 R   0,3   0,1   0:00.87 top
      1 root      20   0  102508  12548   9208 S   0,0   0,3   0:01.02 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.02 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-e+
     10 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_kthr+
     12 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_rude+
```

```
laplateforme_@laplateforme:~/Documents$ vmstat 1
procs ----------mémoire---------- -échange- -----io---- -système- ------cpu-----
 r  b   swpd  libre tampon  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 2584284  33052 571724    0    0    38     7   83   64  0  1 99  0  0
 1  0      0 2584284  33052 571724    0    0     0     0  233  154  2  1 97  0  0
 1  0      0 2584284  33052 571724    0    0     0     4  214  160  4  2 94  0  0
 2  0      0 2584284  33052 571724    0    0     0     0  214  129  3  1 96  0  0
 1  0      0 2584284  33052 571724    0    0     0     0  171  124  2  1 97  0  0
^C
laplateforme_@laplateforme:~/Documents$ vmstat 1 > ressources.csv
^C
^C
laplateforme_@laplateforme:~/Documents$ cat ressources.csv
procs ----------mémoire---------- -échange- -----io---- -système- ------cpu-----
 r  b   swpd  libre tampon  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 2571032  33316 572008    0    0    34     6   81   61  0  1 99  0  0
 1  0      0 2571032  33316 572008    0    0     0     0  195  160  1  1 98  0  0
 1  0      0 2571032  33316 572008    0    0     0     0  162   97  2  1 97  0  0
 1  0      0 2571032  33316 572008    0    0     0     0  155   93  1  1 98  0  0
 2  0      0 2571032  33316 572008    0    0     0     0  144   86  0  0 100  0  0
```

vmstat est une commande qui permet de rapporter des statistiques sur la mémoire virtuelle, la mémoire, le CPU, les processus, les entrées/sorties et les interruptions système, on ajoute **1** pour spécifier l'intervalle de temps en secondes auquel les statistiques doivent être collectées et affichées et on redirige ses données vers un fichier nommé **ressources.csv**

SCRIPTING AVANCÉ

```
#!/bin/bash

repertoire_cible="Plateforme"
repertoire_sauvegarde="Document"
mkdir -p "repertoire_cible"
archive="backup_$(date +%Y%m%d%_%H%M%S) . tar.gz"
chemin_archive="$repertoire_cible/$archive"
tar -czf "$chemin_archive" "$repertoire_cible"
echo "Sauvegarde de $repertoire_cible fait dans $chemin_archive"
find "$repertoire_cible" -type f -mtime +7 -exec rm {} \;
```

```
laplateforme_@laplateforme:~/Documents$ nano sash.sh
laplateforme_@laplateforme:~/Documents$ ./sash.sh
tar: Plateforme : fichier modifié pendant sa lecture
Sauvegarde de Plateforme fait dans Plateforme/backup_20240320%_131003 . tar.gz
laplateforme_@laplateforme:~/Documents$ ls
donnees.csv  Plateforme.tar.gz  ressources.csv  sash.sh  script1.sh  spy.py
Plateforme   repertoire_cible   rm.sh           script1  scriptmv.sh
```

Ce script assure une sauvegarde du répertoire "**Plateforme**" tout en nettoyant les fichiers anciens, le tout suivi d'une confirmation visuelle.

Il crée un répertoire spécifié appelé "Plateforme" s'il n'existe pas déjà.

En utilisant la commande **tar**, il crée une archive compressée (format tar.gz) du répertoire "**Plateforme**" avec un nom de fichier comprenant la date et l'heure actuelles. Cette archive est sauvegardée dans le répertoire "**Plateforme**".

À l'aide de la commande **find**, il cherche les fichiers dans le répertoire "**Plateforme**" qui ont été modifiés il y a plus de 7 jours.
Il supprime ces fichiers.

```
# Edit this file to introduce tasks to be run by cron.
4 13 * * * /Documents/sash.sh
#
```

AUTOMATISATION DES MISES À JOUR LOGICIELS

```
#!/bin/bash
read -p "Voulez-vous mettre  jour les logiciels ? (O/n) : " choix
#choix de l'utilisateur
if [[ $choix == "O" || $choix == "o" || $choix == "" ]]; then
        sudo apt update && sudo apt upgrade -y
        if [ $? -eq 0 ]; then
                echo "Maj des logiciels fait !"
        else
                echo "Aucune maj n'est faite"
        fi
else
        echo "Aucune maj dispo"
fi
```

```
laplateforme_@laplateforme:~/Documents$ nano majlo.sh
laplateforme_@laplateforme:~/Documents$ ./majlo.sh
Voulez-vous mettre  jour les logiciels ? (O/n) : o
Ign :1 cdrom://[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 with f
irmware 20231210-17:57] bookworm InRelease
Err :2 cdrom://[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 with f
irmware 20231210-17:57] bookworm Release
  Veuillez utiliser apt-cdrom afin de faire reconnaître ce cédérom par votre APT. apt-g
et update ne peut être employé pour ajouter de nouveaux cédéroms
Lecture des listes de paquets... Fait
E: Le dépôt cdrom://[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 w
ith firmware 20231210-17:57] bookworm Release n'a pas de fichier Release.
N: Les mises à jour depuis un tel dépôt ne peuvent s'effectuer de manière sécurisée, et
 sont donc désactivées par défaut.
N: Voir les pages de manuel d'apt-secure(8) pour la création des dépôts et les détails
de configuration d'un utilisateur.
Aucune maj n'est fait
```

Ce script utilise la commande **read -p** pour demander à l'utilisateur s'il souhaite mettre à jour les logiciels, avec "O" comme réponse par défaut.

Si l'utilisateur répond "O" ou "o", ou appuie simplement sur Entrée, le script exécute **sudo apt update** pour mettre à jour la liste des paquets disponibles, suivi de **sudo apt upgrade -y** pour mettre à jour les logiciels installés. Si les commandes précédentes se sont exécutées avec succès, le script affiche "Maj des logiciels faite !", sinon il affiche "Aucune maj n'est faite". Si l'utilisateur répond autre chose que "O" ou "o", le script affiche "Aucune maj dispo".

GESTION DES DÉPENDANCES LOGICIELS

```python
#!/usr/bin/env python3

import subprocess

def installer_paquets():
    print("Installation des paquets nécessaires...")

    serveur_web = input("Voulez-vous installer Apache (A/a) ou Nginx (N/n) ? : ")
    if serveur_web.lower() == "a":
        subprocess.run(["sudo", "apt", "install", "apache2", "-y"])
    elif serveur_web.lower() == "n":
        subprocess.run(["sudo", "apt", "install", "nginx", "-y"])
    else:
        print("Choix non valide. Installation d'Apache par défaut.")
        subprocess.run(["sudo", "apt", "install", "apache2", "-y"])

    subprocess.run(["sudo", "apt", "install", "phpmyadmin", "-y"])

    base_de_donnees = input("Voulez-vous installer MySQL (M/m) ou MariaDB (R/r) ? : ")
    if base_de_donnees.lower() == "m":
        subprocess.run(["sudo", "apt", "install", "mysql-server", "-y"])
    elif base_de_donnees.lower() == "r":
        subprocess.run(["sudo", "apt", "install", "mariadb-server", "-y"])
    else:
        print("Choix non valide. Installation de MySQL par défaut.")
        subprocess.run(["sudo", "apt", "install", "mysql-server", "-y"])

    subprocess.run(["sudo", "apt", "install", "nodejs", "npm", "-y"])

    subprocess.run(["sudo", "apt", "install", "git", "-y"])

    print("Installation des paquets terminée.")

installer_paquets()

```

Ce script Python automatise l'installation de logiciels nécessaires pour un serveur web de base. Il utilise la fonction installer_paquets() pour installer les logiciels.

L'utilisateur est invité à choisir entre Apache (A/a) ou Nginx (N/n), et la commande d'installation appropriée est exécutée via la bibliothèque subprocess.

Le script installe ensuite PHPMyAdmin pour gérer les bases de données MySQL/MariaDB.

L'utilisateur est invité à choisir entre MySQL (M/m) ou MariaDB (R/r), et le choix approprié est installé.

Le script installe également Node.js, npm et Git.

```
laplateforme_@laplateforme:~/Documents$ python3 servweb.py
Installation des paquets nécessaires...
Voulez-vous installer Apache (A/a) ou Nginx (N/n) ? : a
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Aucune version du paquet apache2 n'est disponible, mais il existe dans la base
de données. Cela signifie en général que le paquet est manquant, qu'il est devenu obsol
ète
ou qu'il n'est disponible que sur une autre source

E: Le paquet « apache2 » n'a pas de version susceptible d'être installée
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet phpmyadmin
Voulez-vous installer MySQL (M/m) ou MariaDB (R/r) ? : m
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet mysql-server
```

À chaque étape, le script gère les réponses de l'utilisateur et exécute les commandes d'installation appropriées. Une fois toutes les installations terminées, un message indique que l'installation des paquets est terminée.

```
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet nodejs
E: Impossible de trouver le paquet npm
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet git
Installation des paquets terminée.
```

SÉCURISER SES SCRIPTS

Pour sécuriser vos scripts, il est essentiel d'adopter plusieurs bonnes pratiques.

Il faut que les permissions des scripts soient strictement nécessaires et utilisent des privilèges minimum pour leur exécution, limitant ainsi l'accès non autorisé aux ressources du système. L'utilisation du chmod +x ne permet une sécurité optimal dans l'exécution des scripts car l'option x du chmod donne toutes les permissions dans les systèmes linux. C'est à dire qu'une personne avec des droits restreint (non sudoers) peut alors inclure des commandes sudo dans le script en le modifiant, elle peut alors contourner ses restrictions.

Nous pouvons aussi enregistrer les activités des scripts via une journalisation adéquate est également crucial pour surveiller les tentatives d'accès non autorisées et pour identifier les problèmes de sécurité potentiels.

Lors du codage d'un script, il faut éviter de mettre les commandes sudo dans le script. Si ce dernier en a besoin pour la bonne exécution du script pour un téléchargement de paquet par exemple, nous pouvons tout simplement exécuter le script en sudo comme ceci ; sudo ./script.sh

UTILISATION D'API WEB DANS UN SCRIPT

```bash
#!/bin/bash

API_URL="https://jsonplaceholder.typicode.com/users"

# Répertoire de logs
LOG_DIR="/var/log/api_logs"
LOG_FILE="$LOG_DIR/api_log_$(date +%Y-%m-%d).log"

envoyer_requete() {
    local url="$1"
    local response=""

    response=$(curl -s "$url")

    if [ $? -eq 0 ]; then
        logger -t API "Requête: GET $url"
        logger -t API "Réponse: $response"

        echo "Réponse de l'API :"
        echo "$response"
    else
        logger -t API "Erreur : Impossible d'envoyer la requête à l'API."
        exit 1
    fi
}

if [ ! -d "$LOG_DIR" ]; then
    mkdir -p "$LOG_DIR"
fi

envoyer_requete "$API_URL"

```

Ce script Shell exploite les données de l'API JSONPlaceholder, une API REST de test qui fournit des données factices pour le développement et les tests. La communication avec l'API se fait de manière sécurisée en utilisant la commande curl.

Le script définit une fonction envoyer_requete() qui prend en paramètre l'URL de l'API à appeler. La fonction envoie une requête GET à l'API en utilisant curl et stocke la réponse dans une variable. Si la requête échoue, la fonction affiche un message d'erreur et quitte le script avec un code d'erreur.

Le script enregistre les requêtes et les réponses de l'API dans un fichier de journalisation situé dans le répertoire /var/log/api_logs. Le nom du fichier de journalisation contient la date courante au format YYYY-MM-DD.

Le script appelle ensuite la fonction envoyer_requete() avec l'URL de l'API JSONPlaceholder pour récupérer une liste d'utilisateurs. Les résultats sont affichés dans la console.

En cas d'erreur avec l'API, le script gère la situation en affichant un message d'erreur approprié et en quittant avec un code d'erreur. Le logging est un concept important en cybersécurité, car il permet de suivre les activités et de diagnostiquer les problèmes éventuels. Dans ce script, le logging est utilisé pour enregistrer les requêtes et les réponses de l'API à des fins de suivi et de débogage.


```
Réponse de l'API :
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
```