

# Révision Système

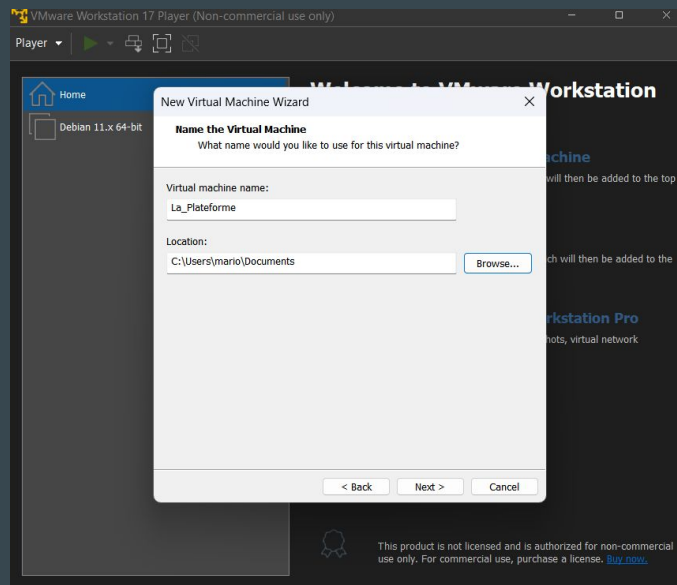
Ilian Bonsens  
Lucas Bendia  
Marion Borne

## TABLE DES MATIÈRES

Création d'une VM .....	1
Commandes et recherches avancées.....	1
Compression et décompression de fichier .....	2
Manipulation de texte .....	3
Gestion des processus .....	4
Surveillance des ressources système .....	5
Scripting avancé .....	5
Automatisation des mises à jour logiciel.....	6
Gestion des dépendances logicielles .....	7
Sécuriser ses scripts .....	8
Utilisation d'API Web dans un script .....	9

# Création d'une VM

En premier lieu, on télécharge la version Gratuite de VM Ware (ou la Pro en ayant récupéré un code gratuit)  
Une fois l'installation de VM Ware faite, on utilise une image de Debian (pour notre groupe AMD64), puis  
l'installation de la VM peut commencer.

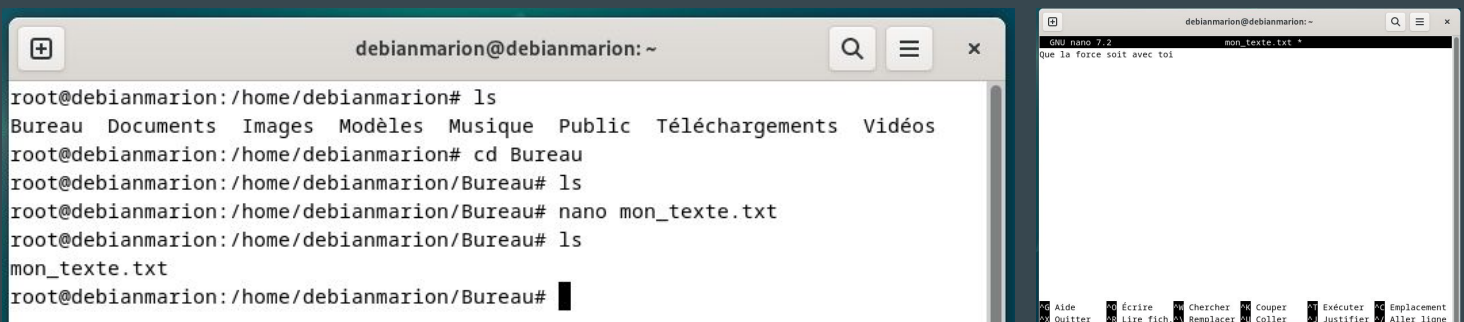


Nous avons rencontré plusieurs problèmes (notamment l'installation de certains paquets) avec la version Debian du doc.

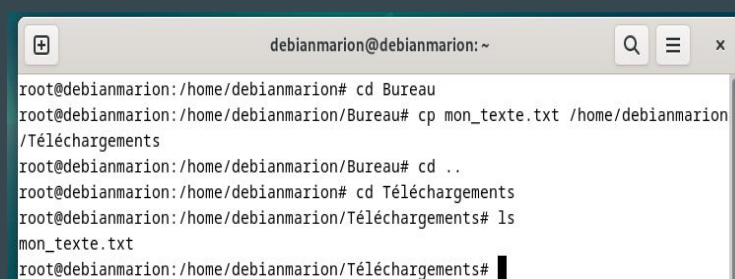
On a donc réinstallé une version antérieure 12.5.0 qui elle, n'a pas posé de problèmes.

## Commandes et recherches avancées

Afin de créer cinq fichiers textes només "mon\_texte.txt" contenant le texte "Que la force soit avec toi." dans cinq répertoire différent, nous avons commencé par créer un premier fichier avec la commande :  
**nano mon\_texte.txt** dans lequel nous avons inséré le texte demandé.



Puis pour copier coller ce fichier texte dans un différent répertoire nous avons fait la commande suivante (en nous trouvant à l'endroit ou a été mis le fichier initial) :  
**cp mon\_texte.txt /chemin/vers/emplacement/souhaité**



A partir du répertoire de notre session, en utilisant le mot “force” pour localiser les cinq fichier mon\_texte.txt nous avons utilisé la commande :

**grep -r --include="\*.txt" force /home/la\_plateforme**

```
root@debian:/home/la_plateforme# grep -r --include="*.txt" force /home/la_plateforme
/home/la_plateforme/Images/mon_texte.txt:Que la force soit avec toi.
/home/la_plateforme/Documents/mon_texte.txt:Que la force soit avec toi.
/home/la_plateforme/Bureau/mon_texte.txt:Que la force soit avec toi.
/home/la_plateforme/Téléchargements/mon_texte.txt:Que la force soit avec toi.
/home/la_plateforme/Vidéos/mon_texte.txt:Que la force soit avec toi.
```

## Compression et décompression de fichiers

Pour créer un fichier nommé “Plateforme” nous utilisons la commande **mkdir Plateforme** tout en étant dans le dossier Documents.

Pour y ajouter le fichier txt créé précédemment, nous avons utilisé cette commande :  
**cp mon\_texte.txt /chemin/vers/repertoire/Plateforme**

```
root@debianmarion:/home/debianmarion# cd Documents
root@debianmarion:/home/debianmarion/Documents# mkdir Plateforme
root@debianmarion:/home/debianmarion/Documents# ls
mon_texte.txt  Plateforme
root@debianmarion:/home/debianmarion/Documents# cp mon_texte.txt /home/debianmarion/Documents/Plateforme
root@debianmarion:/home/debianmarion/Documents# cd Plateforme
root@debianmarion:/home/debianmarion/Documents/Plateforme# ls
mon_texte.txt
root@debianmarion:/home/debianmarion/Documents/Plateforme#
```

Pour dupliquer quatre fois ce fichier en le renommant, nous avons utilisé la commande :  
**cp mon\_texte.txt mon\_texte\_numero.txt** (cp fichier\_original fichier\_nouveau)

```
root@debianmarion:/home/debianmarion/Documents/Plateforme# cp mon_texte.txt mon_texte2.txt
root@debianmarion:/home/debianmarion/Documents/Plateforme# cp mon_texte.txt mon_texte3.txt
root@debianmarion:/home/debianmarion/Documents/Plateforme# cp mon_texte.txt mon_texte4.txt
root@debianmarion:/home/debianmarion/Documents/Plateforme# ls
mon_texte1.txt mon_texte2.txt mon_texte3.txt mon_texte4.txt mon_texte.txt
root@debianmarion:/home/debianmarion/Documents/Plateforme#
```

Afin d'archiver le répertoire Plateforme, nous avons utilisé tar et gzip :  
**tar -czvf Plateforme.tar.gz /Chemin/vers/repertoire/Plateforme**

```
root@debianmarion:/home/debianmarion/Documents/Plateforme# tar -czvf Plateforme.tar.gz /home/debianmarion/Documents/Plateforme
tar: Suppression de « / » au début des noms des membres
/home/debianmarion/Documents/Plateforme/
/home/debianmarion/Documents/Plateforme/mon_texte4.txt
/home/debianmarion/Documents/Plateforme/mon_texte1.txt
/home/debianmarion/Documents/Plateforme/mon_texte.txt
/home/debianmarion/Documents/Plateforme/mon_texte2.txt
/home/debianmarion/Documents/Plateforme/mon_texte3.txt
tar: /home/debianmarion/Documents/Plateforme : fichier modifié pendant sa lecture
root@debianmarion:/home/debianmarion/Documents/Plateforme# ls
mon_texte1.txt mon_texte3.txt mon_texte.txt
mon_texte2.txt mon_texte4.txt Plateforme.tar.gz
root@debianmarion:/home/debianmarion/Documents/Plateforme#
```

Pour décompresser les archives créées nous avons utilisé la commande suivante :  
`tar -xzf Plateforme.tar.gz -C /chemin/vers/emplacement/archive`

```
debianlucas@debianlucas:~/Documents$ tar -xzf Plateforme.tar.gz -C /home/debianlucas/Documents
home/debianlucas/Documents/Plateforme/
home/debianlucas/Documents/Plateforme/Mon_texte.txt
home/debianlucas/Documents/Plateforme/Mon_texte3.txt
home/debianlucas/Documents/Plateforme/Mon_texte2.txt
home/debianlucas/Documents/Plateforme/Mon_texte1.txt
home/debianlucas/Documents/Plateforme/Mon_texte4.txt
debianlucas@debianlucas:~/Documents$ ls
home Mon_texte.txt Plateforme Plateforme.tar.gz
debianlucas@debianlucas:~/Documents$
```

## Manipulation de texte

Pour créer un script python permettant la création d'un fichier CVS, nous avons créé un fichier en utilisant nano pour éditer notre script : `nano script.py`  
Puis nous avons lancé le script afin de vérifier que le fichier CVS se crée bien à l'emplacement souhaité en utilisant la commande `python3 script.py`

```
GNU nano 7.2 script.py
import csv

class Csv:
    def __init__(self, data):
        self.data = data

    def create_csv(self, liste):
        with open(liste, 'w', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(self.data)
        print(f"Le fichier '{liste}' a été créé avec succès.")

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

csv_file = Csv(data)
csv_file.create_csv('personnes.csv')
```

```
la_plateforme@debian:~$ python3 script.py
Le fichier 'personnes.csv' a été créé avec succès.
la_plateforme@debian:~$ ls
Bureau Documents Images Modèles Musique personnes.csv Public script.py
```

Afin d'extraire les informations relatives aux villes du script précédemment créé, nous avons tapé cette commande :

`awk -F ' ' '{print $3}' personnes.csv`

(le \$3 faisant référence à la troisième rangée d'informations des personnes donc les villes)

```
la_plateforme@debian:~$ awk -F ' ' '{print $3}' personnes.csv
Paris
Lyon
Marseille
Toulouse
la_plateforme@debian:~$
```

# Gestion des processus

Nous avons utilisés plusieurs outils pour pouvoir recenser tous les processus actifs sur notre système :

**ps -e** (qui est très basique mais exploitable)

**htop** qui est plus évolué (après avoir dû installer le paquet)

**top** qui est plus détaillé et avec lequel nous avons continué pour ce job

```
root@debianmarion:/home/debianmarion/script# ps -e
  PID TTY          TIME CMD
    1 ?            00:00:03 systemd
    2 ?            00:00:00 kthreadd
    3 ?            00:00:00 rcu_gp
    4 ?            00:00:00 rcu_par_gp
    5 ?            00:00:00 slub_flushwq
    6 ?            00:00:00 netns
    8 ?            00:00:00 kworker/0:0H-events_highpri
   10 ?            00:00:00 mm_percpu_wq
   11 ?            00:00:00 rcu_tasks_kthread
   12 ?            00:00:00 rcu_tasks_rude_kthread
   13 ?            00:00:00 rcu_tasks_trace_kthread
   14 ?            00:00:00 ksoftirqd/0
   15 ?            00:00:01 rcu_preempt
   16 ?            00:00:00 migration/0
   18 ?            00:00:00 cpuhp/0
   19 ?            00:00:00 cpuhp/1
```

7.2% Tasks: 106, 249 thr, 102 kthr; 1 runni  
10.6% Load average: 0.31 0.29 0.53  
Mem 681M/1.89G Uptime: 00:59:26  
Swp 343M/975M

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1502	debianmari	20	0	3757M	203M	81628	S	15.2	10.5	1:57.58	/usr/bin/gnom
1531	debianmari	20	0	3757M	203M	81628	S	5.3	10.5	0:38.09	/usr/bin/gnom
3569	root	20	0	8184	4496	3392	R	4.6	0.2	0:00.36	htop
1527	debianmari	20	0	3757M	203M	81628	S	4.0	10.5	0:37.69	/usr/bin/gnom
1701	debianmari	20	0	305M	7260	5924	S	0.7	0.4	0:03.54	/usr/bin/ibus
2272	debianmari	20	0	678M	24092	17932	S	0.7	1.2	0:07.31	/usr/libexec/
2279	debianmari	20	0	678M	24092	17932	S	0.7	1.2	0:00.01	/usr/libexec/
1	root	20	0	164M	8900	5904	S	0.0	0.4	0:03.20	/sbin/init
264	root	20	0	49852	12028	11316	S	0.0	0.6	0:01.27	/lib/systemd/
294	root	20	0	27612	4472	2776	S	0.0	0.2	0:00.43	/lib/systemd/
370	systemd-ti	20	0	90228	2792	2080	S	0.0	0.1	0:00.14	/lib/systemd/
399	systemd-ti	20	0	90228	2792	2080	S	0.0	0.1	0:00.01	/lib/systemd/
400	root	20	0	231M	6116	5708	S	0.0	0.3	0:00.21	/usr/libexec/
416	avahi	20	0	8292	2524	2156	S	0.0	0.1	0:00.24	avahi-daemon:
435	root	20	0	6608	2476	2224	S	0.0	0.1	0:00.03	/usr/sbin/cro

Pour terminer un processus de façon normale on tape la commande : **kill 4293** (numéro PID affilié à dans notre cas Firefox). Lorsqu'on tape cette commande, firefox se ferme et ne s'affiche plus en cours dans le programme top.

Lorsque nous forçons la fermeture d'un processus (par exemple lorsque celui-ci freeze), nous utilisons la commande **kill -KILL 4293** (car la commande kill 4293 ne fonctionne pas)

MiB Éch : 975,0 total, 519,8 libr, 455,2 util. 704,1 dispo Mem

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
4293	debianm+	20	0	10,7g	302076	146760	R	45,8	15,3	0:12.63	firefox-esr
4432	debianm+	20	0	2518812	206536	95864	S	44,9	10,4	0:05.03	Isolated Web Co
1502	debianm+	20	0	3855880	216612	86356	S	42,5	10,9	2:57.67	gnome-shell
2605	debianm+	20	0	244208	21972	17924	S	4,7	1,1	0:09.28	Xwayland
45	root	20	0	0	0	0	S	3,3	0,0	0:10.32	kswapd0
2272	debianm+	20	0	695512	40268	31688	S	1,7	2,0	0:12.79	gnome-terminal-
4433	debianm+	20	0	2439864	100100	80468	S	0,7	5,1	0:00.79	WebExtensions
15	root	20	0	0	0	0	I	0,3	0,0	0:01.81	rcu_preempt
219	root	20	0	0	0	0	S	0,3	0,0	0:00.85	jbd2/sda1-8
1701	debianm+	20	0	312744	7248	5912	S	0,3	0,4	0:04.46	ibus-daemon
2085	debianm+	20	0	360752	11276	5144	S	0,3	0,6	0:11.41	blueman-tray
3921	root	20	0	0	0	0	I	0,3	0,0	0:00.21	kworker/u256:0-events_unbou+
3956	root	20	0	0	0	0	D	0,3	0,0	0:03.69	kworker/0:3+pm
3958	root	20	0	0	0	0	I	0,3	0,0	0:00.94	kworker/1:3-events
3969	root	20	0	0	0	0	I	0,3	0,0	0:00.06	kworker/u256:2-writeback
1	root	20	0	168040	8848	5908	S	0,0	0,4	0:03.28	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.02	kthreadd

```
root@debianmarion:/home/debianmarion# kill 4293
root@debianmarion:/home/debianmarion# kill -KILL 4293
```

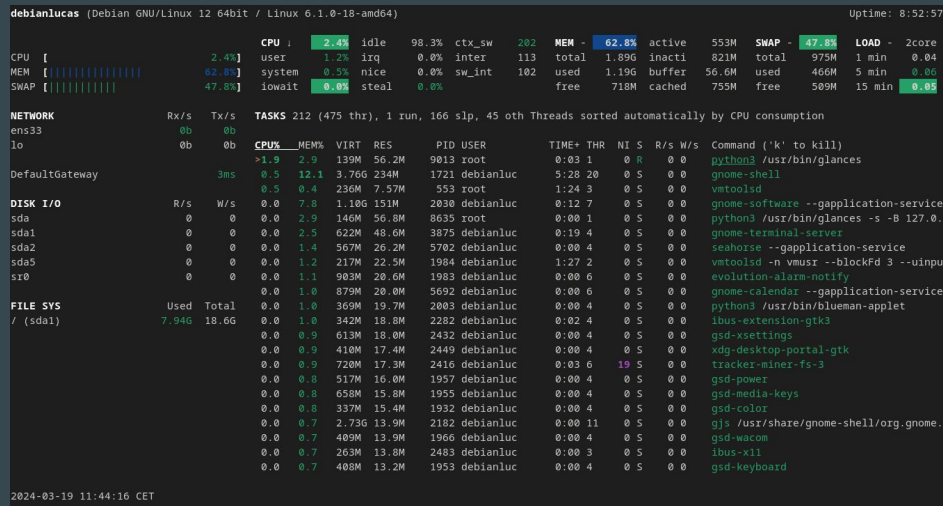


# Surveillance des ressources système

Afin de mettre en place une surveillance en temps réel du CPU, de la mémoire et d'autres ressources système, nous avons dû installer le paquet glances :

**sudo apt install glances** puis on lance le programme en tapant **glances**

```
root@debianlucas: /home/debianlucas/script-python# sudo apt install glances
```



Pour enregistrer les informations visualisés dans le programme dans un fichier CSV, nous avons utilisé la commande :

**glances --export csv --export-csv-file /home/utilisateur/Documents/InfoSystm.csv --time 5**

```
root@debianlucas: /home/debianlucas# glances --export csv --export-csv-file /home  
/debianlucas/Documents/InfoSystm.csv --time 5  
root@debianlucas: /home/debianlucas# *
```

## Scripting avancé

Pour créer un script bash permettant d'automatiser la sauvegarde périodique du répertoire Plateforme, nous avons tout d'abord créé notre script : **nano sauvegardes.sh**

```
GNU nano 7.2 sauvegardes.sh  
#!/bin/bash  
  
# Chemin du répertoire à sauvegarder  
SOURCE_DIR="/home/debianmarion/Documents/LaPlateforme"  
  
# Chemin du répertoire où les sauvegardes seront stockées  
BACKUP_DIR="/home/debianmarion/Documents/sauvegardes"  
  
# Format de l'horodatage (année-mois-jour_heure-minute)  
TIMESTAMP=$(date +%Y-%m-%d_%H-%M)  
  
# Nom du fichier de sauvegarde  
BACKUP_FILE="$BACKUP_DIR/Plateforme_${TIMESTAMP}.tar.gz"  
  
# Créer une sauvegarde  
tar -czf "$BACKUP_FILE" "$SOURCE_DIR"  
  
# Afficher un message de confirmation  
echo "Sauvegarde de $SOURCE_DIR effectuée avec succès dans $BACKUP_FILE"  
  
# Gestion de l'historique des sauvegardes  
# (Optionnel) Limiter le nombre de sauvegardes conservées pour économiser l'espace disque  
  
MAX_BACKUPS=5  
NUM_BACKUPS=$(ls -1 "$BACKUP_DIR"/*.tar.gz | wc -l)  
  
# Supprimer les anciennes sauvegardes si le nombre maximum est dépassé  
if [ "$NUM_BACKUPS" -gt "$MAX_BACKUPS" ]; then  
    echo "Suppression des anciennes sauvegardes pour ne pas dépasser $MAX_BACKUPS fichiers."  
    ls -1 "$BACKUP_DIR"/*.tar.gz | head -n -"$MAX_BACKUPS" | xargs rm  
fi
```

Nous donnons les autorisations nécessaires à l'exécution du script grâce à la commande :

**chmod +x sauvegardes.sh**

Puis afin que le script puisse se lancer automatiquement et périodiquement, nous utilisons les crons.

On accepte au paramétrage des crons grâce à la commande :

**crontab -e**

```
root@debianmarion:/home/debianmarion/Documents/sauvegardes# chmod +x sauvegardes.sh
root@debianmarion:/home/debianmarion/Documents/sauvegardes# crontab -e
```

Nous paramétrons notre cron pour que le script s'exécute toute les 2 min,  
en lui donnant le chemin vers le script

```
*/* * * * * /home/debianmarion/Documents/sauvegardes/sauvegardes.sh
```

Afin de ne pas surcharger notre fichier de sauvegarde, nous avons mis dans notre script une option permettant de ne garder que les 5 dernières sauvegardes et de supprimer les plus anciennes.

```
root@debianmarion:/home/debianmarion/Documents/sauvegardes# ls
Plateforme_2024-03-20_11-46.tar.gz Plateforme_2024-03-20_11-58.tar.gz
Plateforme_2024-03-20_11-48.tar.gz sauvegardes.sh
root@debianmarion:/home/debianmarion/Documents/sauvegardes#
```

## Automatisation des mises à jour logicielles

Pour créer un script bash permettant d'automatiser la mise à jour des logicielles existants sur notre système,  
nous avons tout d'abord créé notre script : **nano maj.sh**

```
GNU nano 7.2 maj.sh *
#!/bin/bash

echo "Vérification des mises à jour disponibles..."
sudo apt-get update

echo "Les mises à jour suivantes sont disponibles :"
apt list --upgradable

read -p "Voulez-vous mettre à jour ces logiciels ? (O/N) " reponse

if [[ $reponse =~ ^[Oo]$ ]]
then
    echo "Mise à jour des logiciels..."
    sudo apt-get upgrade -y
else
    echo "Mise à jour annulée."
fi

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

Nous lui avons donné les autorisation nécessaires grâce à la commande **chmod +x chemin/vers/script**  
puis nous l'avons lancé pour vérifier qu'il fonctionne

```
root@debianlucas:/home/debianlucas/Documents# chmod +x /home/debianlucas/Documents/maj.sh
```

```
root@debianlucas:/home/debianlucas/Documents# ./maj.sh
```

Pour finir nous avons paramétré notre crontab afin que cette mise à jour s'exécute toute les 15 jours

```
debianlucas@debianlucas: ~
GNU nano 7.2 /tmp/crontab.9rd42j/crontab
30 15 * * * /home/debianlucas/Documents/script.sh

* 15 * * * /home/debianlucas/Documents/maj.sh
```

# Gestion des dépendances logicielles

Afin de créer un script pour simplifier l'installation et la gestion des dépendances logicielles pour un projet web, tout en assurant la compatibilité entre les différentes versions, nous devons créer un script bash qui doit installer :

- un serveur web (nous avons choisi Apache)
  - phpMyAdmin
- Un système de gestion de base de données relationnelle (MySQL ou MariaDB)
  - Un environnement JavaScript côté serveur (Node.js) avec npm
  - Un système de contrôle de version (Git)

Voici notre script :

```
GNU nano 7.2 dependances.sh
#!/bin/bash

apt install apache2 -y
apt install mariadb-server -y
apt install git -y
apt install nodejs npm -y
apt install php libapache2-mod-php php-mysql phpmyadmin -y
sudo service apache2 restart
```

Pour récupérer le fichier d'installation de PhpMyAdmin, nous avons effectué cette commande :

wget <https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.tar.gz>

```
root@debian:/home/la_plateforme/Documents# wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.tar.gz
```

Pour configurer phpmyadmin dans le serveur apache2 installé, nous accédons au fichier `apache2.conf`

et rentre : `Include /etc/phpmyadmin/apache.conf`

Puis nous relançons le serveur apache2 pour que celui-ci prenne en compte les modifications faites

```
root@debianlucas:/home/debianlucas/Documents# nano /etc/apache2/apache2.conf
```

```
# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf
Include /etc/phpmyadmin/apache.conf
```

```
root@debianlucas:/home/debianlucas/Documents# systemctl restart apache2
```

Pour se connecter, nous avons fourni un mot de passe.

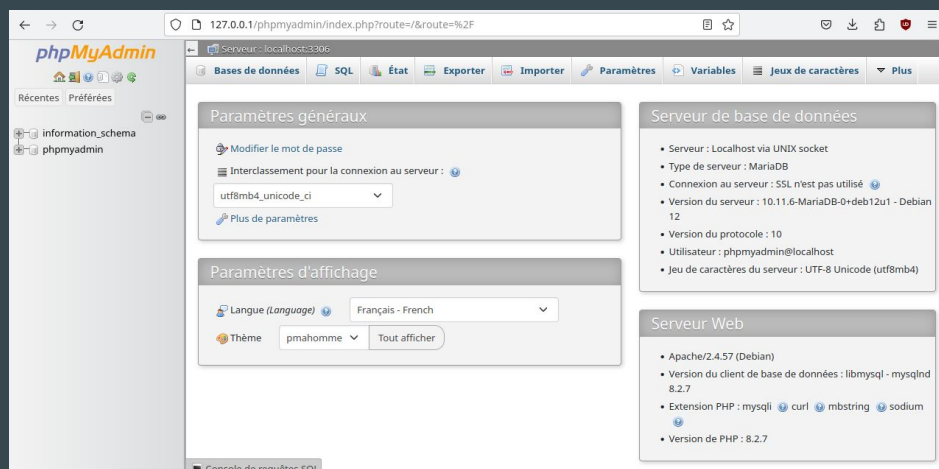
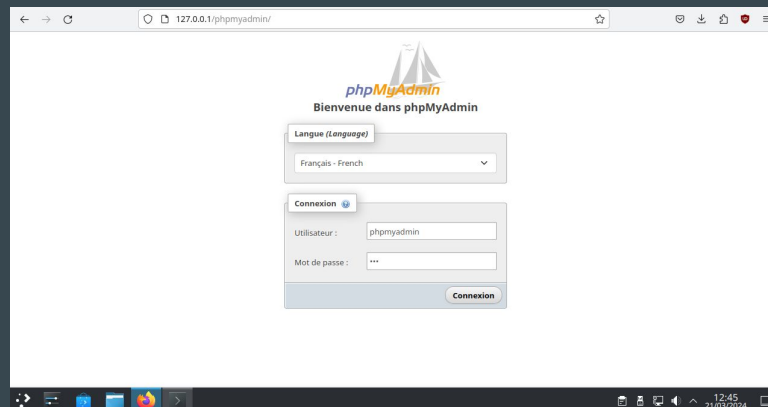
Pour se connecter et connaître notre identifiant, nous pouvons les trouver dans le fichier `/etc/phpmyadmin/config-db.php`

Nous voyons que l'utilisateur est un utilisateur par défaut phpmyadmin et que le mot de passe est celui que nous avons rentré précédemment. (pratique si on a perdu nos informations)



```
GNU nano 7.2 config-db.php
#?php
##
## database access settings in php format
## automatically generated from /etc/dbconfig-common/phpmyadmin.conf
## by /usr/sbin/dbconfig-generate-include
##
## by default this file is managed via ucf, so you shouldn't have to
## worry about manual changes being silently discarded. *however*,
## you'll probably also want to edit the configuration file mentioned
## above too.
##
$dbuser='phpmyadmin';
$dbpass='marion';
$basepath='';
$dbname='phpmyadmin';
$dbserver='localhost';
$dbport='3306';
$dbtype='mysql';
```

Nous nous sommes bien connectés à PhpMyAdmin :



## Sécuriser ses scripts

Pour sécuriser nos scripts, il a suffit de rendre leur modification impossible pour les utilisateurs n'ayant pas les privilèges. Ainsi, les scripts ne seront que visualisables et exécutables par les autres utilisateurs.

```
root@debian:/home/la_plateforme/Documents# chmod -w pokeapi.sh
```

Nous avons aussi découvert la commande shc. Il permet de chiffrer n'importe quel script interprété sous un terminal Linux.

Après la compilation de l'outil, un binaire du nom de "shc" est présent dans le répertoire. nous avons donc tapé la commande : `apt install shc` pour installer le paquet

```
root@debian:/home/la_plateforme/Documents# apt install shc
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
```

- script.sh.x est chiffré et exécutable (ci dessous script chiffré en capture d'écran)
  - script.sh.x.c est chiffré mais non exécutable
  - script.sh reste le même

[illegible]

```

GNU nano 7.2                                pokeapi.sh *
#!/bin/bash

echo "Entrez le nom ou le numéro d'un pokemon:"
read poke_name
base_url="https://pokeapi.co/api/v2/pokemon/"
base_url+=$poke_name
if [ "$(curl -s $base_url)" = "Not Found" ]; then
    printf "\nErreur: Le pokemon n'existe pas\n\n"
else
    weight=$(curl -s $base_url | jq -r '.weight')
    weight_corrected=$((weight / 10))
    height=$(curl -s $base_url | jq -r '.height')
    height_corrected=$((height * 10))
    type=$(curl -s $base_url | jq -r '.types[0].type.name')
    name=$(curl -s $base_url | jq -r '.name')
    ability=$(curl -s $base_url | jq -r '.abilities[0].ability.name')
    ability1=$(curl -s $base_url | jq -r '.abilities[1].ability.name')
    if [ "$ability1" = 'null' ]; then
        ability1=""
    fi
    printf "\nInformations sur: ${name}\n\n"
    printf "taille:\t${NC}${height_corrected}cm\n"
    printf "poinds:\t${NC}${weight_corrected}kg\n"
    printf "type:\t${NC}${type}\n"
    printf "capacites:\t${NC}${ability}/${ability1}\n"
fi

```

Les données nous sont renvoyées en JSON, on utilise donc la commande `jq` pour rechercher les valeurs désirées dans le fichier.

```
root@debian:/home/la_plateforme/Documents# ./pokeapi.sh
Entrez le nom ou le numéro d'un pokemon:
23

Informations sur: ekans

taille: 200cm
poinds: 6kg
type:  poison
capacites:  intimidate/shed-skin
```

Voici le résultat lorsque l'on fait une requête avec un numéro de pokemon.  
Cela fonctionnera aussi avec les noms de pokemon (en anglais).

```
user=$(whoami)
timestamp=$(date +"%Y-%m-%d %H:%M:%S")
echo -e "${timestamp} - Utilisateur: ${user}" >> pokemon_log.txt
```

Pour garder une trace des utilisateurs qui font des requêtes à cette API, nous avons rajouté ces lignes à la fin de notre script.  
Cela récupère la date et l'heure exacte de la requête ainsi que l'utilisateur qui a exécuté le script ; ce qui nous donne le fichier suivant.

```
GNU nano 7.2
2024-03-25 10:17:02 - Utilisateur: root
2024-03-25 10:39:12 - Utilisateur: root
```