



PSMM :

- **Création et Configuration des VM**
- **Explications et Démonstrations.**





La Plateforme

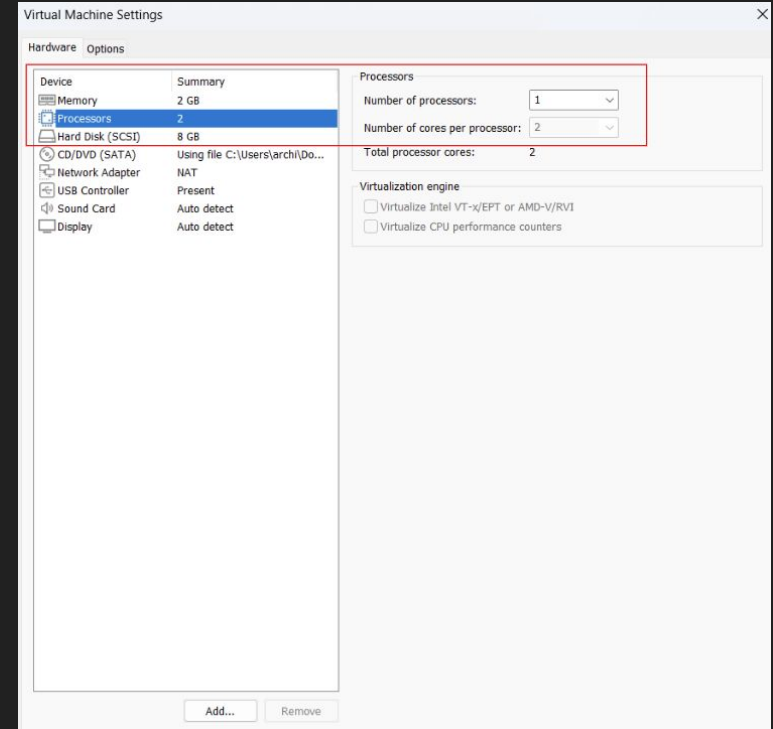
/ PSMM

PROJET

1. Création et configuration des environnements virtuels .
2. Explications et Démonstrations.

Serveur FTP : 1 Go RAM - 1 vCPU - 8 Go disque

On commence par faire la configuration matérielle (Hardware) de notre VM ; en lui attribuant 2 Go de RAM pour que ce soit plus rapide, 1 vCPU et un disque de 8 Go.



- Avant de passer à l'installation de notre serveur FTP on vérifie d'abord que notre adresse ip soit en statique et que notre serveur soit déclaré :

```
GNU nano 7.2 interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug ens33
iface ens33 inet static
    address 192.168.197.134
    netmask 255.255.255.0
    gateway 192.168.197.2
```



```
GNU nano 7.2 hosts
127.0.0.1    localhost
127.0.1.1    Jftp
192.168.197.134 Jftp
# The following lines are desirable for IPv6 capable hosts
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

- On passe à la création du groupe sftp pour le service sftp :

```
monitor@Jftp:~$ sudo groupadd sftp_user|
monitor@Jftp:~$ mkdir /sftp
```

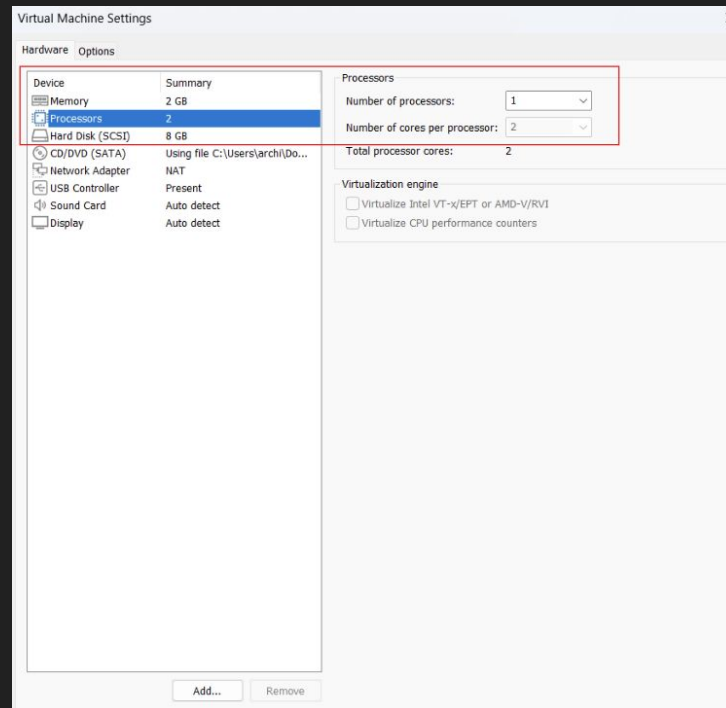
On se rend dans le fichier config `sshd_config` et on décommente et modifie les lignes suivante :

```
# override default of no subsystems
#Subsystem      sftp      /usr/lib/openssh/sftp-server
Subsystem       sftp      internal-sftp
# Example of overriding settings on a per-user basis
Match Group sftp
    X11Forwarding no
    AllowTcpForwarding no
#    PermitTTY no
    ForceCommand internal-sftp
```

```
PS C:\Users\archi> sftp monitor@192.168.197.134
Connected to 192.168.197.134.
sftp>
```

Serveur WEB : 1 Go RAM - 1 vCPU - 8 Go disque

- On commence par faire la configuration matérielle(Hardware) de notre VM ; en lui attribuant 2 Go de RAM pour que ce soit plus rapide, 1 vCPU et un disque de 8 Go.



- Avant de passer à l'installation de notre serveur WEB on vérifie d'abord que notre adresse ip soit en statique

```
GNU nano 7.2 interfaces *
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug ens33
iface ens33 inet static
    address 192.168.197.135
    gateway 192.168.197.2
```



La Plateforme

/ PSMM

- On installe les paquets correspondant à “apache2” :

```
monitor@Jweb:/etc/network$ sudo apt install apache2
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4
  liblua5.3-0 ssl-cert
Paquets suggérés :
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
Les NOUVEAUX paquets suivants seront installés :
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4
  liblua5.3-0 ssl-cert
0 mis à jour, 11 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 2 368 ko/2 727 ko dans les archives.
Après cette opération, 9 224 ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] o
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 libcurl4 amd64 7.88.1-10+deb12u7 [390 kB]
Réception de :2 http://deb.debian.org/debian bookworm/main amd64 apache2-bin amd64 2.4.62-1~deb12u1 [1 385 kB]
Réception de :3 cdrom://[Debian GNU/Linux 12.4.0 _Bookworm_ - Official amd64 DVD Binary-1 with firmware 20231210-17:57]
```


- On démarre notre service et on vérifie qui fonctionne :

```
monitor@Jweb:/etc/network$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Tue 2024-09-24 12:30:40 CEST; 39s ago
     Docs: https://httpd.apache.org/docs/2.4/
    Main PID: 2250 (apache2)
      Tasks: 55 (limit: 1065)
    Memory: 13.0M
       CPU: 139ms
    CGroup: /system.slice/apache2.service
            └─2250 /usr/sbin/apache2 -k start
              └─2251 /usr/sbin/apache2 -k start
                └─2252 /usr/sbin/apache2 -k start

sept. 24 12:30:40 Jweb systemd[1]: Starting apache2.service - The Apache HTTP Server...
sept. 24 12:30:40 Jweb apachectl[2249]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, please see the mod_ssl module docs
sept. 24 12:30:40 Jweb systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-16/16 (END)
```


- On se rend ensuite sur notre machine hôte et on vérifie que apache s'ouvre correctement

⚠ Non sécurisé | 192.168.197.135



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/  
|-- apache2.conf  
|   |-- ports.conf  
|-- mods-enabled  
|   |-- *.load  
|   |-- *.conf  
|-- conf-enabled  
|   |-- *.conf  
|-- sites-enabled  
|   |-- *.conf  
|
```

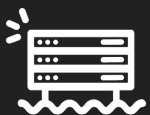
- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining

- Nous allons maintenant configurer une authentification de base pour l'utilisateur "monitor", afin de lui permettre de se connecter à notre site web de manière sécurisée.
- Ensuite , si ce n'est pas déjà fait, nous installons le paquet "apache2-utils", qui fournit une série d'outils facilitant la gestion et l'administration d'Apache.

```
monitor@Jweb:/etc/network$ sudo apt-get install apache2-utils
[sudo] Mot de passe de monitor :
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
apache2-utils est déjà la version la plus récente (2.4.62-1~deb12u1).
apache2-utils passé en « installé manuellement ».
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

- Cette commande permet d'ajouter l'utilisateur "monitor" ainsi que son mot de passe à la base de données d'authentification.

```
monitor@Jweb:/etc/network$ sudo htpasswd -c /etc/apache2/.htpasswd monitor
New password:
Re-type new password:
Adding password for user monitor
```



La Plateforme

/ PSMM

- En consultant le contenu du dossier, on peut voir le nom d'utilisateur ainsi que le mot de passe chiffré pour chaque enregistrement.

```
monitor@Jweb:/etc/network$ cat /etc/apache2/.htpasswd  
monitor:$apr1$KkhYnyKE$yxIN8VhWH7rTNwbzMzQRu/
```

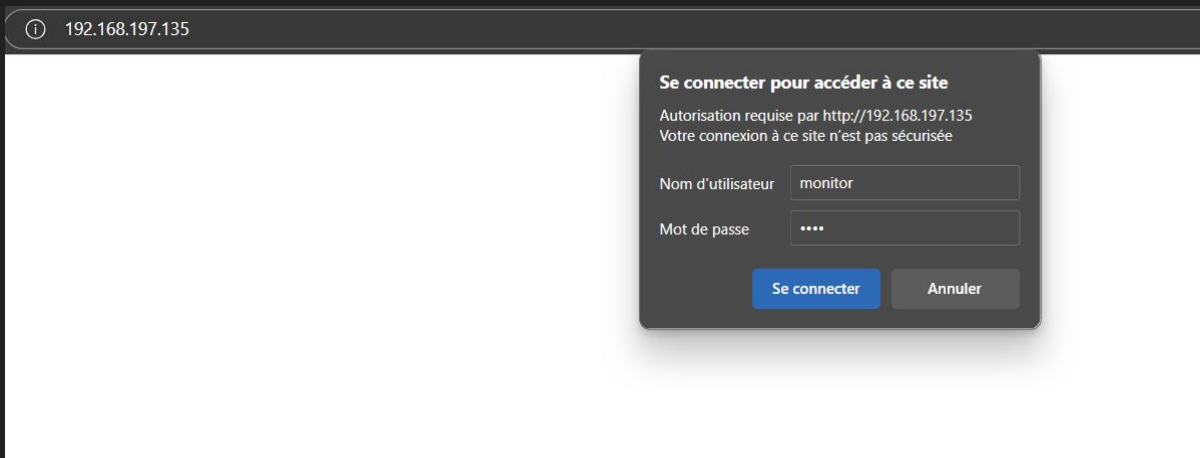
- Ensuite, nous accédons au fichier de configuration par défaut d'Apache pour y définir les règles d'authentification appliquées au répertoire spécifié : /var/www/html.

```
GNU nano 7.2                                000-default.conf  
<VirtualHost *:80>  
    # The ServerName directive sets the request scheme, hostname and port that  
    # the server uses to identify itself. This is used when creating  
    # redirection URLs. In the context of virtual hosts, the ServerName  
    # specifies what hostname must appear in the request's Host: header to  
    # match this virtual host. For the default virtual host (this file) this  
    # value is not decisive as it is used as a last resort host regardless.  
    # However, you must set it for any further virtual host explicitly.  
    #ServerName www.example.com  
  
    ServerAdmin webmaster@localhost  
    DocumentRoot /var/www/html  
  
    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,  
    # error, crit, alert, emerg.  
    # It is also possible to configure the loglevel for particular  
    # modules, e.g.  
    #LogLevel info ssl:warn  
  
    <Directory "/var/www/html">  
        AuthType Basic  
        AuthName "Restricted Access"  
        AuthUserFile /etc/apache2/.htpasswd  
        Require valid-user  
    </Directory>  
    # For most configuration files from conf-available/, which are
```

- Nous vérifions la configuration à l'aide de la commande suivante :

```
monitor@Jweb:/etc/apache2/sites-available$ sudo apachectl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

- N'oubliez pas de redémarrer le service Apache2 une fois la configuration vérifiée. Ensuite, en retournant sur notre navigateur, la page devrait s'afficher de la manière suivante.



192.168.197.135

Se connecter pour accéder à ce site

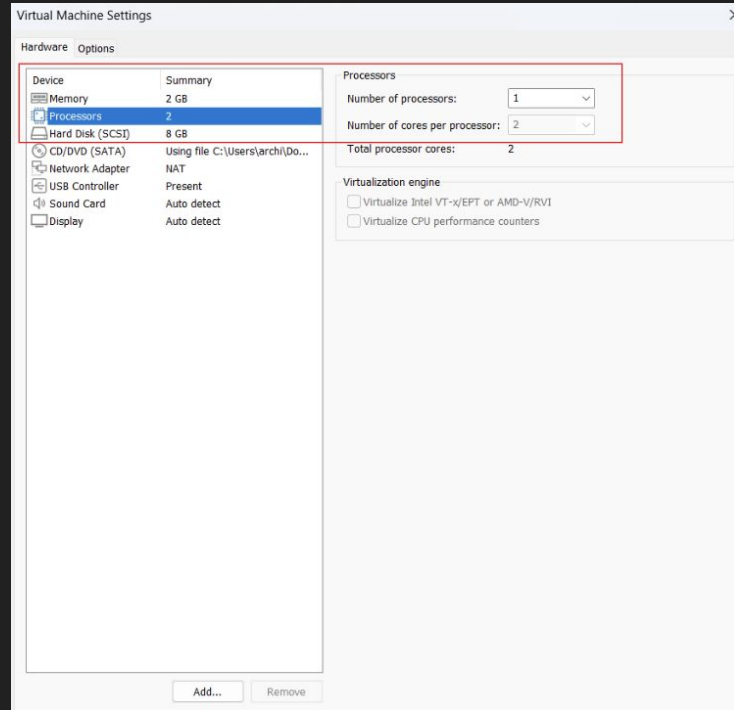
Autorisation requise par http://192.168.197.135
Votre connexion à ce site n'est pas sécurisée

Nom d'utilisateur

Mot de passe

Serveur SQL : 2 Go RAM, 2 vCPU, 8 Go disque

- On commence par faire la configuration matérielle (Hardware) de notre VM ; en lui attribuant 2 Go de RAM pour que ce soit plus rapide, 1 vCPU et un disque de 8 Go.





La Plateforme

/ PSMM

- Avant de commencer on met à jour nos paquets et après on installe MariaDb :

```
monitor@Jmariadb:/etc/apt$ sudo apt update
Atteint :1 http://deb.debian.org/debian bookworm InRelease
Atteint :2 http://deb.debian.org/debian bookworm-updates InRelease
Atteint :3 http://security.debian.org/debian-security bookworm-security InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Tous les paquets sont à jour.
monitor@Jmariadb:/etc/apt$ sudo apt upgrade -y
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Calcul de la mise à jour... Fait
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

```
monitor@Jmariadb:/etc/apt$ sudo apt install mariadb-server -y
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdaxctl1 libdbd-mariadb-perl
libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl libgpm2 libhtml-parser-perl
libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
liblwp-mediatypes-perl liblzo2-2 libmariadb3 libmpfr6 libncurses6 libndctl6 libnuma1 libpmem1 libregexp-ipv6-perl
libsigsegv2 libsnappy1v5 libterm-readkey-perl libtimedate-perl liburi-perl liburing2 mariadb-client
mariadb-client-core mariadb-common mariadb-plugin-provider-bzip2 mariadb-plugin-provider-lz4
mariadb-plugin-provider-lzma mariadb-plugin-provider-lzo mariadb-plugin-provider-snappy mariadb-server-core
mysql-common psmisc pv rsync socat
Paquets suggérés :
gawk-doc libltdb-perl libnet-daemon-perl libsql-statement-perl gpm libdata-dump-perl libipc-sharedcache-perl
libbusiness-isbn-perl libwww-perl mailx mariadb-test netcat openbsd doc-base python3-braceexpand
Les NOUVEAUX paquets suivants seront installés :
galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdaxctl1 libdbd-mariadb-perl
libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl libgpm2 libhtml-parser-perl
libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
liblwp-mediatypes-perl liblzo2-2 libmariadb3 libmpfr6 libncurses6 libndctl6 libnuma1 libpmem1 libregexp-ipv6-perl
libsigsegv2 libsnappy1v5 libterm-readkey-perl libtimedate-perl liburi-perl liburing2 mariadb-client
mariadb-client-core mariadb-common mariadb-plugin-provider-bzip2 mariadb-plugin-provider-lz4
mariadb-plugin-provider-lzma mariadb-plugin-provider-lzo mariadb-plugin-provider-snappy mariadb-server
mariadb-server-core mysql-common psmisc pv rsync socat
0 mis à jour, 50 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 20,5 Mo dans les archives.
```



La Plateforme

/ PSMM

- Démarrer MariaDB et configurer son démarrage automatique au démarrage du système

```
monitor@Jmariadb:/etc/apt$ sudo systemctl start mariadb
monitor@Jmariadb:/etc/apt$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.11.6 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
   Active: active (running) since Tue 2024-09-24 16:28:32 CEST; 39s ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Main PID: 3365 (mariadb)
   Status: "Taking your SQL requests now..."
     Tasks: 13 (Limit: 2273)
   Memory: 178.8M
      CPU: 834ms
   CGroup: /system.slice/mariadb.service
           └─3365 /usr/sbin/mariabdb

sept. 24 16:28:32 Jmariadb mariabdb[3365]: 2024-09-24 16:28:32 0 [Note] InnoDB: log sequence number 45582; transaction
sept. 24 16:28:32 Jmariadb mariabdb[3365]: 2024-09-24 16:28:32 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/my
sept. 24 16:28:32 Jmariadb mariabdb[3365]: 2024-09-24 16:28:32 0 [Note] Plugin 'FEEDBACK' is disabled.
sept. 24 16:28:32 Jmariadb mariabdb[3365]: 2024-09-24 16:28:32 0 [Warning] You need to use --log-bin to make --expire-l
sept. 24 16:28:32 Jmariadb mariabdb[3365]: 2024-09-24 16:28:32 0 [Note] Server socket created on IP: '127.0.0.1'.
sept. 24 16:28:32 Jmariadb mariabdb[3365]: 2024-09-24 16:28:32 0 [Note] InnoDB: Buffer pool(s) load completed at 240924
sept. 24 16:28:32 Jmariadb mariabdb[3365]: 2024-09-24 16:28:32 0 [Note] /usr/sbin/mariabdb: ready for connections.
sept. 24 16:28:32 Jmariadb mariabdb[3365]: Version: '10.11.6-MariaDB-0+deb12u1' socket: '/run/mysqld/mysqld.sock' por
sept. 24 16:28:32 Jmariadb systemd[1]: Started mariadb.service - MariaDB 10.11.6 database server.
sept. 24 16:28:32 Jmariadb /etc/mysql/debian-start[3382]: Upgrading MySQL tables if necessary.
lines 1-23/23 (END)
```

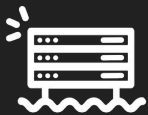
- Se connecter à MariaDB en tant que root, puis créer l'utilisateur "monitor".

```
monitor@Jmariadb:/etc/apt$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```



La Plateforme

/ PSMM

- Création d'un utilisateur 'monitor' avec accès distant :

```
MariaDB [(none)]> CREATE USER 'monitor'@'%' IDENTIFIED BY 'root';
Query OK, 0 rows affected (0,006 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'monitor'@'%';
Query OK, 0 rows affected (0,003 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,001 sec)

MariaDB [(none)]> exit
Bye
```

- Modifier le fichier de configuration pour autoriser les connexions distantes

sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf on remplace bind-address = 127.0.0.1 par :
bind-address = 0.0.0.0

```
GNU nano 7.2 50-server.cnf

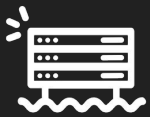
# Broken reverse DNS slows down connections considerably and name resolve is
# safe to skip if there are no "host by domain name" access grants
#skip-name-resolve

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0

#
# * Fine Tuning
#

#key_buffer_size        = 128M
#max_allowed_packet     = 1G
#thread_stack           = 192K
#thread_cache_size      = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
#myisam_recover_options = BACKUP
#max_connections        = 100
#table_cache            = 64

#
# * Logging and Replication
#
```

La Plateforme

/ PSMM

- On teste la connexion distante depuis une autre machine (par exemple, depuis ta machine hôte Windows)

```
PS C:\Users\archi> mysql -u monitor -p -h 192.168.197.136
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 31
Server version: 5.5.5-10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

- Entre chaque modification on n'oublie pas de restart notre service

Clés publique et privée : différences et utilités

Les clés publiques et privées sont fondamentales dans le chiffrement asymétrique, utilisé pour sécuriser les communications, comme avec SSH.

Clé publique

- Fonction : Chiffre les données.
- Partage : Publique et peut être partagée sans risque.
- Utilité dans SSH : Elle est stockée sur le serveur (dans `~/.ssh/authorized_keys`) et sert à vérifier l'identité du client lors d'une connexion.

Clé privée

- Fonction : Déchiffre les données chiffrées et permet de signer numériquement.
- Sécurité : Doit rester secrète ; sa divulgation permettrait à un tiers d'accéder à vos systèmes.
- Utilité dans SSH : Conservée sur le client (dans `~/.ssh/id_rsa`), elle permet de répondre correctement aux défis du serveur, prouvant ainsi l'identité de l'utilisateur.

Processus SSH

1. Clé publique sur le serveur : ajoutée dans `~/.ssh/authorized_keys`.
2. Clé privée sur le client : gardée secrète.
3. Modifier `sshd_config`
4. Connexion : le serveur envoie un défi, et le client utilise sa clé privée pour y répondre. Si la réponse est correcte, l'accès est accordé.

En résumé

- Clé publique : chiffrer et partager.
- Clé privée : pour déchiffrer et prouver son identité, doit rester secrète.

- On génère ensuite les clés public et privé

```
monitor@Jpython:/etc/ssh$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/monitor/.ssh/id_rsa):
/home/monitor/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/monitor/.ssh/id_rsa
Your public key has been saved in /home/monitor/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1a3Moh67nbS2Vfv3hN0VaUXWHTuj/v7KEaBVeZpnnl4 monitor@Jpython
The key's randomart image is:
+---[RSA 4096]-----+
|
|             .+B|
|            ..B|
|           .o o @|
|          o o .o =*|
|         .+ S . .+o|
|        . . . . .o +E|
|       o . . . . =o|
|      . =,+ . .o.o|
|     +o= . . . =+o|
+---[SHA256]-----+
```

VM dédiée aux scripts Python :

- Debian sans interface graphique, avec Python, MySQL/MariaDB (client uniquement), FTP et outils d'envoi de mails.
- On commence par retourner dans notre VM serveur, nous avons sélectionné la VM nommée "Jftp". Nous allons accéder à cette VM et modifier le fichier `/etc/ssh/sshd_config` pour rétablir l'accès par mot de passe, car sans cela, il sera impossible de configurer la nouvelle clé.

```
GNU nano 7.2                                sshd_config
# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no
```

- On passe à l'installation de Python :

Pour commencer, nous mettons à jour nos paquets et installons les dépendances nécessaires à l'installation et à la compilation de Python ainsi que des bibliothèques et outils associés.

```
monitor@Jpython:~$ sudo apt install -y build-essential libssl-dev libbz2-dev libreadline-dev libsqlite3-dev wget  
et curl llvm libgdbm-dev liblzma-dev python3-openssl git  
[sudo] Mot de passe de monitor :  
Lecture des listes de paquets... Fait  
Construction de l'arbre des dépendances... Fait  
Lecture des informations d'état... Fait  
build-essential est déjà la version la plus récente (12.9).
```

Ensuite, nous téléchargeons la dernière version de Python en utilisant la commande suivante.

```
monitor@Jpython:~$ wget https://www.python.org/ftp/python/3.11.4/Python-3.11.4.tgz  
--2024-09-25 13:14:19-- https://www.python.org/ftp/python/3.11.4/Python-3.11.4.tgz  
Résolution de www.python.org (www.python.org)... 199.232.80.223, 2a04:4e42:7d::223  
Connexion à www.python.org (www.python.org)|199.232.80.223|:443... connecté.  
requête HTTP transmise, en attente de la réponse... 200 OK  
Taille : 26526163 (25M) [application/octet-stream]  
Sauvegarde en : « Python-3.11.4.tgz.1 »  
  
Python-3.11.4.tgz.1      100%[=====] 25,30M 17,5MB/s  ds 1,4s  
  
2024-09-25 13:14:21 (17,5 MB/s) - « Python-3.11.4.tgz.1 » sauvegardé [26526163/26526163]
```

- Ensuite, nous décompressons notre fichier avec la commande suivante :

```
tar -xvf Python-3.11.4.tgz
```

On teste ensuite si tout fonctionne :

```
monitor@Jpython:~/Python-3.11.4$ python3
Python 3.11.2 (main, Aug 26 2024, 07:20:54) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Salut")
Salut
>>>
```

Pour sftp Pour sftp, il suffit juste de vérifier si openssh client est bien installé, et on se connecte avec l'adresse ip du serveur sftp que l'on a configuré sur la vm serveur.

```
monitor@Jpython:~$ sftp monitor@192.168.197.134
Connected to 192.168.197.134.
sftp> _
```

- On passe maintenant à la création de notre environnement virtuel :

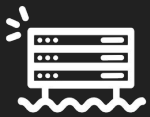
```
monitor@Jpython:~$ python3 -m venv mon_environnement/  
monitor@Jpython:~$ source mon_environnement/bin/activate  
(mon_environnement) monitor@Jpython:~$
```

- Pour les scripts j'ai créé un répertoire "scripts" :

```
monitor@Jpython:~/scripts$ source mon_environnement/bin/activate  
(mon_environnement) monitor@Jpython:~/scripts$ ls  
mon_environnement  ssh_login.py  
(mon_environnement) monitor@Jpython:~/scripts$
```


2. Explications et Démonstrations.





Scripts de Gestion SSH

1

ssh_login.py

Connexion SSH et exécution de commandes

2

ssh_login_sudo.py

Connexion SSH avec commandes sudo

3

ssh_mysql.py

Vérification des accès au serveur MariaDB/MySQL

```
#!/usr/bin/perl
# ssh_login.py
# Connexion SSH et exécution de commandes

use Net::SSH::Perl;
use strict;
use warnings;

my $host = '192.168.1.100';
my $user = 'root';
my $port = 22;

my $ssh = Net::SSH::Perl->new($host, $user, $port);
$ssh->login();

my $cmd = 'ls -la /etc/passwd';
my $output = $ssh->exec($cmd);

print "Output of '$cmd':\n";
print $output;

$ssh->logout();
```

ssh_login_sudo.py

- **Lignes 1-13 :** Le script importe `paramiko` pour la connexion SSH. Il définit les variables de configuration : le chemin vers la clé privée SSH, le nom d'utilisateur, et l'adresse IP du serveur FTP ainsi que le chemin vers le fichier de log FTP.
- **Lignes 16-20 :** La fonction `execute_ftp_command` crée un client SSH avec Paramiko, accepte automatiquement les nouvelles clés hôtes, et se connecte au serveur avec les paramètres fournis.
- **Lignes 22-25 :** Le script demande à l'utilisateur une commande à exécuter sur le serveur distant nécessitant des droits sudo et lit ensuite le mot de passe sudo pour exécuter cette commande avec `sudo -S`.
- **Lignes 27-31 :** Le script lit les lignes de sortie standard (stdout) de la commande exécutée et les affiche dans le terminal.
- **Ligne 33 :** Après avoir exécuté la commande et affiché les résultats, la connexion SSH est fermée.
- **Lignes 39-41 :** Cette section appelle la fonction `execute_ftp_command` avec les paramètres configurés, ce qui déclenche la connexion SSH et l'exécution de la commande sur le serveur distant.

Scripts d'Extraction d'Erreurs

FTP

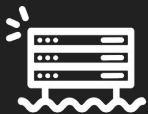
ssh_ftp_error.py : Extraction des logs d'erreurs FTP

Web

ssh_web_error.py : Récupération des erreurs d'accès web

MySQL

ssh_mysql_error.py : Récupération des erreurs de connexion



La Plateforme

/ PSMM

ssh_mysql_error.py

Le script ci-dessus effectue un travail complet d'extraction et d'insertion des logs de connexions échouées depuis un serveur distant vers une base de données, assurant ainsi une traçabilité des événements de sécurité.

- **Connexion à la base de données (lignes 7-8) :**

Il se connecte à la base MariaDB/MySQL en utilisant `pymysql.connect` avec les informations de connexion (hôte, utilisateur, mot de passe, base de données).

- **Configuration SSH (lignes 11-18) :**

Il configure la connexion SSH à un serveur distant, utilise `paramiko` pour ouvrir une connexion SSH, et exécute la commande pour lire les fichiers de logs sur le serveur distant.

- **Traitement du fichier de log (lignes 21-26) :**

Le contenu du fichier log est analysé, ligne par ligne, pour détecter les tentatives de connexion échouées ("Access Denied"). Il stocke les connexions pertinentes dans une liste `connections`.

- **Insertion des tentatives de connexion dans la base de données (lignes 41-47) :**

Chaque tentative échouée est insérée dans la table `access_logs` avec les détails de l'utilisateur, le temps de tentative, l'adresse IP et l'état.

- **Fermeture des connexions (lignes 50-52) :**

Enfin, il ferme la connexion SSH ainsi que la connexion à la base de données après avoir inséré les informations.

Notifications et Sauvegardes

1

Rapport quotidien

ssh_serveur_mail.py : Envoi des connexions échouées

2

Sauvegarde BDD

ssh_cron_backup.py : Toutes les 3 heures, 7 sauvegardes retenues

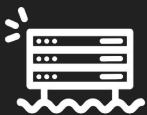




- Le but de ce script est de récupérer les tentatives de connexion échouées de trois serveurs (web, FTP et MariaDB) pour la veille, de générer un rapport détaillé des échecs de connexion, puis d'envoyer ce rapport par email à l'administrateur système.
- Il collecte les tentatives échouées depuis les journaux d'accès de chaque machine, compile les résultats et les présente dans un format lisible pour l'administrateur, incluant les informations comme l'utilisateur, l'heure, l'adresse IP, et le statut de la tentative.

Ce script se divise en plusieurs parties clés :

- **Connexion à la base de données MySQL (lignes 6-12) :** Le script établit une connexion avec la base de données MySQL en utilisant pymysql avec les identifiants stockés dans un fichier de configuration.
- **Calcul de la date d'hier (ligne 14) :** Il calcule la date de la veille pour filtrer les tentatives de connexion échouées qui ont eu lieu uniquement la veille.



- **Fonction `get_failed_logins()` (lignes 19-53)** : Cette fonction extrait les tentatives de connexion échouées des logs de trois serveurs : web, FTP, et MariaDB. Pour chaque machine, elle exécute une requête SQL afin de récupérer les tentatives échouées dans les 24 heures précédentes et stocke les résultats dans un dictionnaire `failed_logins`.
- **Fonction `send_email(report)` (lignes 56-73)** : Elle envoie un email contenant un rapport des tentatives de connexion échouées par machine. L'email est généré à partir d'un objet `MIMEText` et envoyé via le serveur SMTP configuré.
- **Récupération des connexions échouées (lignes 75-77)** : Cette partie du code appelle la fonction `get_failed_logins()` pour obtenir les données des tentatives de connexion échouées de chaque machine pour la veille.
- **Génération du rapport (lignes 79-92)** : Le script parcourt chaque machine (web, FTP, mariadb) et génère un rapport détaillant les tentatives de connexion échouées pour chaque serveur. Si aucune tentative n'est trouvée pour une machine, un message indiquant l'absence de tentatives est ajouté au rapport.
- **Envoi du rapport (ligne 94)** : Le rapport généré est envoyé par email grâce à la fonction `send_email()`.
- **Fermeture de la connexion à la base de données (ligne 96)** : Une fois le processus



La Plateforme

/ PSMM

ssh_cron_backup.py

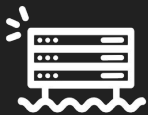
Ce script est conçu pour effectuer des sauvegardes régulières de la base de données, conserver uniquement les sept dernières sauvegardes et supprimer les plus anciennes. Il génère une nouvelle sauvegarde chaque fois qu'il est exécuté, en créant un fichier horodaté pour éviter tout conflit de noms. L'objectif principal est d'automatiser le processus de sauvegarde tout en évitant que le stockage ne soit saturé par des sauvegardes trop anciennes. Ce script est donc utile pour la gestion et la protection des données de manière automatisée.

- **Lignes 5-6** : Création du dossier où seront stockées les sauvegardes. Si le dossier existe déjà, il n'est pas recréé.
- **Lignes 9-10** : Le script génère un nom de fichier pour la sauvegarde basée sur l'horodatage actuel (par exemple, `backup_2024-10-03_12-00-00.sql`).
- **Ligne 13** : Construction de la commande `mysqldump` pour effectuer la sauvegarde de la base de données. Cette commande utilise les identifiants de la base de données (hôte, utilisateur, mot de passe, nom de la base de données).
- **Ligne 16** : Exécution de la commande `mysqldump` pour créer la sauvegarde dans le fichier précédemment généré.
- **Ligne 19** : Le script récupère la liste des sauvegardes présentes dans le dossier, les trie dans l'ordre inverse (de la plus récente à la plus ancienne).
- **Lignes 22-27** : Le script conserve uniquement les sept dernières sauvegardes. Si plus de sept sauvegardes sont présentes, les plus anciennes sont supprimées pour libérer de l'espace.
- **Ligne 29** : Affichage d'un message de succès une fois la sauvegarde effectuée.

Surveillance des Ressources

Script	Fonction
ssh_system_status.py	Suivi RAM, CPU, disque
ssh_system_mail.py	Alerte e-mail si seuils critiques dépassés





La Plateforme

/ PSMM

ssh_system_mail.py

Ce script surveille les ressources système (RAM, CPU, disque) de plusieurs serveurs via SSH, enregistre les informations dans une base de données et envoie un email récapitulatif en cas de dépassement de seuils prédéfinis. Il vérifie également si un email a déjà été envoyé dans l'heure pour éviter les doublons. Enfin, il supprime les données vieilles de plus de 72 heures pour maintenir une base de données propre.

- **Seuils d'alerte : (lignes 4-6)** Définit des limites d'utilisation pour la RAM, le CPU et le disque.
- **Connexion à la base de données : (ligne 11-14)** Connexion à MySQL pour stocker les données de surveillance.
- **Récupération des ressources système :**

Fonction get_system_status (ligne 17-38) : Connecte à un serveur via SSH, exécute des commandes pour récupérer l'utilisation de la RAM, du CPU et du disque, et renvoie ces valeurs.

- **Insertion des données :**

Fonction insert_system_status (ligne 41-48) : Insère l'utilisation des ressources du serveur dans la base de données avec un horodatage.



La Plateforme

/ PSMM

- Envoi d'un email récapitulatif :

Fonction `send_alert_email` (ligne 50-92) : Envoie un email récapitulatif si les seuils sont dépassés. Elle vérifie également si un email a déjà été envoyé récemment pour éviter les doublons.

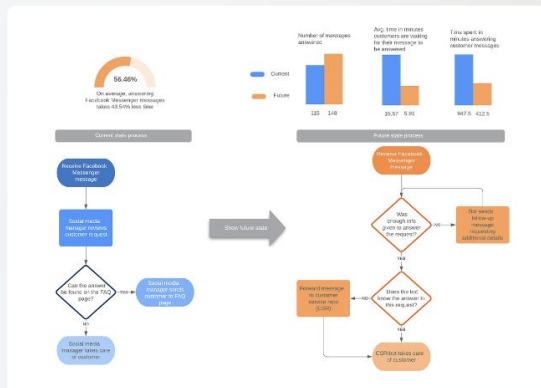
- Suppression des anciennes données :

Fonction `delete_old_status` (ligne 94-98) : Supprime les entrées de la base de données datant de plus de 72 heures.

- Surveillance des serveurs : (ligne 100-125)

Récupère l'état des ressources pour chaque serveur, vérifie si elles dépassent les seuils et enregistre les données dans la base de données. Si des seuils sont franchis, une alerte est ajoutée à la liste.

- Finalisation : (ligne 126) Ferme la connexion à la base de données.



Mises à Jour et Automatisation



Mises à jour

ssh_update.py : Vérification et application des mises à jour Alcasar



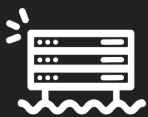
Google Chat

Envoi de rapports périodiques à l'équipe



Automatisation

Limitation des alertes à une par heure



`ssh_update.py`

Ce script sert principalement à surveiller l'état des ressources système (RAM, CPU, disque) sur plusieurs serveurs, à alerter en cas de dépassement des seuils prédéfinis, et à maintenir des informations à jour dans une base de données. Voici un résumé de son utilité

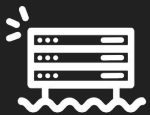
- Surveillance des serveurs : Il récupère l'état des ressources via SSH pour chaque serveur surveillé.
- Stockage des données : Les données récupérées sont stockées dans une base de données MySQL pour une analyse ultérieure.
- Alerte par email : Si les ressources dépassent des seuils (pour la RAM, le CPU ou l'espace disque), le script envoie un email récapitulatif à l'administrateur.
- Maintenance des données : Les enregistrements de la base de données vieux de plus de 72 heures sont supprimés pour éviter l'encombrement.



La Plateforme

/ PSMM

- **Définition des seuils d'alerte (lignes 4-6) :** Les seuils pour le CPU, la RAM et le disque sont définis, et si ces limites sont dépassées, une alerte est déclenchée.
- **Connexion à la base de données (lignes 9-12) :** Cette fonction initialise une connexion à la base de données MySQL où les informations des ressources seront stockées.
- **Récupération des ressources système via SSH (lignes 16-34) :** La fonction `get_system_status()` récupère l'utilisation de la RAM, du CPU, et du disque en se connectant aux serveurs via SSH, en utilisant Paramiko pour exécuter les commandes appropriées. Elle retourne les valeurs récupérées pour ces ressources.
- **Insertion des données dans la base de données (lignes 36-43) :** La fonction `insert_system_status()` insère les valeurs des ressources collectées (RAM, CPU, disque) ainsi que leur horodatage dans une table MySQL.
- **Envoi d'un email récapitulatif (lignes 45-93) :** La fonction `send_alert_email()` est responsable de l'envoi d'un email si des alertes sont déclenchées. Elle vérifie également si un email a déjà été envoyé dans l'heure et, si c'est le cas, empêche un nouvel envoi.
- **Suppression des données plus anciennes (lignes 94-98) :** La fonction `delete_old_status()` supprime les enregistrements de la base de données vieux de plus de 72 heures pour maintenir les données à jour.
- **Vérification des serveurs (lignes 99-123) :** Cette boucle itère sur la liste des serveurs définie, récupère l'état des ressources via `get_system_status()`, insère ces informations dans la base de données et vérifie si une alerte doit être déclenchée en comparant les valeurs aux seuils définis.
- **Fermeture de la connexion (ligne 127) :** La connexion à la base de données est fermée une fois toutes les opérations terminées.



Automatisation Google Chat

Ce script sert à récupérer les informations sur l'état des ressources (RAM, CPU et disque) des serveurs distants via SSH, puis à envoyer ces informations sous forme de message dans un espace Google Chat.

Importation des modules (lignes 1-4) : Ce script utilise les modules paramiko, json, httpplib2, et un fichier de configuration pour récupérer les informations système et les envoyer à Google Chat.

Récupération de l'état des ressources système (lignes 6-21) : La fonction `get_system_status(server)` se connecte à un serveur via SSH (Paramiko) et exécute des commandes Linux pour obtenir l'état de la RAM, du disque et du CPU. Les résultats sont renvoyés sous forme de texte.

Envoi du message à Google Chat (lignes 23-39) : La fonction `send_message_to_google_chat(message)` envoie les données système sous forme de message à un espace Google Chat en utilisant une requête HTTP POST.

Exécution principale (lignes 41-50) : Le script récupère l'état des ressources des serveurs spécifiés dans config et les envoie à l'espace Google Chat via la fonction `send_message_to_google_chat`.