

Projet Administration système Linux



Ce travail est réalisé par :

- Mohyi Eddine NAHI
- Khalil EL JOULALI
- Saad ALAHYANE

Sous la supervision de : Pr. Meryiem CHERGUI

Remerciement :

Nous tenons à remercier toutes les personnes qui ont contribué au succès de notre projet et qui nous ont aidés lors de la rédaction de ce rapport.

Tout d'abord, nous adressons nos remerciements à notre professeur, Mme. Meryiem CHERGUI qui nous a beaucoup aidé pendant nos cours d'administration système et nous a donné la possibilité de travailler sur ce projet.

Son explication et ses conseils nous ont permis de cibler les compétences nécessaires et assimiler tant de nouveaux concepts.

Nous remercions également tous les membres du groupe pour leurs efforts et leur esprit d'équipe, ce qui nous a permis de comprendre les problématiques du projet choisi et de réaliser ce dernier.

Enfin, nous tenons à remercier toutes les personnes qui nous ont conseillées lors de la réalisation du projet et l'établissement de son rapport.

Table de matières

Cahier des charges	4
Introduction	5
I- Partie préliminaire	6
1-définition SSH	6
2-guide SSH	6
2.1-Installation SSH	6
2.2 Connexion automatique +accès automatique au root	6
2.3 Aspect sécurité	9
3-autres configuration	9
II- Partie modélisation	10
1- Présentation de l'architecture du projet	10
1.1- Présentation des différents éléments (serveurs, emplacement ...)	10
1.2- Fonctionnement du système	11
2. Cas envisagés	12
III- Script Shell et mise en marche	12
1- Présentation des différents éléments utilisés	12
1.1- Les variables	12
2.1- Les fonctions	14
2.3- Commandes importantes	15
2- Fichier log	17
IV- Planification et notification de sauvegarde (crontab)	17
V- Résultats obtenus	18
VI-Conclusion	20
Bibliographie et webographie	21
Annexe	22

Cahier de charges :

Pour réaliser le projet il faut respecter les points suivants :

- Des serveurs qui vont communiquer entre eux.
- Le serveur maître communique d'une manière sécurisée via un protocole SSH et agence des actions entre les autres machines.
- Développer l'exploitation des clés SSH.
- Communications beaucoup plus complexes et un nombre de serveurs applicatifs plus important.
- Automatiser la sauvegarde de la base de données et localiser la sauvegarde sur un autre serveur.
- L'arrêt de la base de données impose l'arrêt des applications qui utilisent cette base, le temps de la sauvegarde.
- Automatiser l'arrêt puis le démarrage des applicatifs et de la base de données, de manière séquentielle et cohérente.
- Planification du processus de sauvegarde par crontab.

Introduction :

L'administration d'un système Linux est une charge importante, sensible et stratégique qui détermine l'intégrité, la pérennité, l'accessibilité et la confidentialité des ressources (matériels, logiciels, données) d'un système d'information. Les responsabilités de l'administrateur système (le super utilisateur) et de son équipe est une question de quantité, de qualité et d'intention.

I- Partie préliminaire

1-Définition SSH:

Le ssh (appelé aussi Secure Shell) est un protocole et en même temps un programme informatique de communication qui rend la communication entre client-serveur plus sécurisé. Pour ce faire, on utilise d'abord le chiffrement asymétrique pour s'échanger discrètement une clé secrète de chiffrement symétrique, le but du chiffrement asymétrique c'est le fait de communiquer une clé entre le destinataire et le destinataire, de telle sorte à rendre la tâche d'interception - en analysant le trafic- quasi-impossible. Ainsi, on utilise, tout le temps, cette dernière pour le chiffrement symétrique pour chiffrer les échanges.

2-Guide SSH:

2.1-Installation ssh:

Pour profiter des services du SSH, on installe OpenSSH - dans notre cas on est sur 'Debian'- donc on utilise la commande suivante :

```
$ apt install openssh-server
```

2.2-Connexion automatique SSH:

Après l'installation du SSH, on peut accéder à ce dernier localement ou à distance grâce à la redirection des ports. On est amené à utiliser la syntaxe suivante:

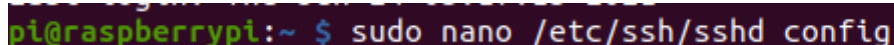
```
$ ssh username@IpAdresse
```

Ensuite, un code va être demandé afin d'accéder au Shell de la machine en question. Mais pour le cas de notre projet, on est amené à programmer un script qui exécute des commandes dans plusieurs serveurs, ce qui rend la tâche aberrante si on est obligé à insérer le mot de passe à chaque exécution et étape du programme. En plus, notre script perdra son but initial qui est l'automatisation de la tâche. Pour se faire, on va expliquer comment se connecter à un serveur avec le protocole ssh sans devoir entrer le passwd à chaque fois.

Côté serveur, modification du sshd:

Premièrement, on doit accéder au fichier "sshd_config" par la commande :

```
$ sudo nano /etc/ssh/sshd_config
```



```
pi@raspberrypi:~ $ sudo nano /etc/ssh/sshd_config
```

Deuxièmement, on modifie le fichier "sshd_config" en décommentant la ligne ci-dessous:

```
AuthorizedKeysFile    %h/.ssh/authorized_keys
```

```
GNU nano 3.2 /etc/ssh/sshd_config

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin yes
#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
```

Finalement, pour s'assurer que les changements ont été bien appliqués, on redémarre le service ssh à l'aide la commande :

\$ service ssh restart

```
pi@raspberrypi:~ $ service ssh restart
```

Cette action permettra de réaliser les étapes suivantes :

Au niveau du client il faut générer une clé qui va être stockés dans /root/.ssh/authorized_keys au serveur en question, ainsi à chaque tentative de connexion de ce même serveur client , aucun mot de passe ne vas être demandé.

Génération d'une clé ssh:

```

mohyi@hp:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/mohyi/.ssh/id_ed25519):
/home/mohyi/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mohyi/.ssh/id_ed25519
Your public key has been saved in /home/mohyi/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:qN1+rSf7gj9kSSNE/xANhFf/gUhuGr2GXqNbFn47xpQ mohyi@hp
The key's randomart image is:
+--[ED25519 256]--+
|      ..O+=.      |
|      O.=.O..     |
|      . oo=  ...  |
|      .. Bo.  ..  |
|      . S= O. . .  |
|      o .. O oE    |
|      . . .* =O.   |
|      .. O ++.    |
|      .++Oo..     |
+-----[SHA256]-----+

```

La génération des clés ssh peut se faire avec divers algorithmes de cryptage dans notre cas c'est : ed25519 .

Envoi de la clé et authentification :

Lors de l'envoi de la clé une authentification est demandée, c'est la dernière demande de mot de passe. Pour se faire on exécute la commande ci-dessous :

\$ ssh-copy-id **username@IpAdresse**

```

mohyi@hp:~$ ssh-copy-id user@serverAdress

```

Dans notre cas cette étape est bien faite , la commande nous affiche que cette action est déjà faite.(Normalement le mot de passe est demandé après l'exécution de la commande)

```

mohyi@hp:~$ ssh-copy-id pi@172.20.10.8
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)

```


Finalement, le résultat obtenu est le suivant : Connection automatique

```
mohyl@hp:~$ ssh pi@172.20.10.8
Linux raspberrypi 5.10.17-v7+ #1421 SMP Thu May 27 13:59:01 BST 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jun 24 20:02:13 2021 from 172.20.10.10
pi@raspberrypi:~$
```

2.3-Aspect sécurité (Interdire la connexion par mot de passe) :

On a pensé que la connexion entre les serveurs doit être sécurisée, c'est à dire que seul le serveur maître doit accéder et naviguer entre les serveurs. Pour ce faire, on décommente PasswordAuthentication et on change son paramètre à 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 no
#PermitEmptyPasswords no

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

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no
```

3-Autre configuration :

Pour éviter le fait d'entrer le mot de passe après chaque commande exécutée, ce qui est le cas pour notre projet -comme on a un script qui contient plusieurs commandes de ce genre- il est préférable d'automatiser cette action . Il suffit d' aller au fichier **/etc/sudoers** et d'ajouter une ligne à la fin du fichier de type: **user ALL=(ALL) NOPASSWD: ALL** .

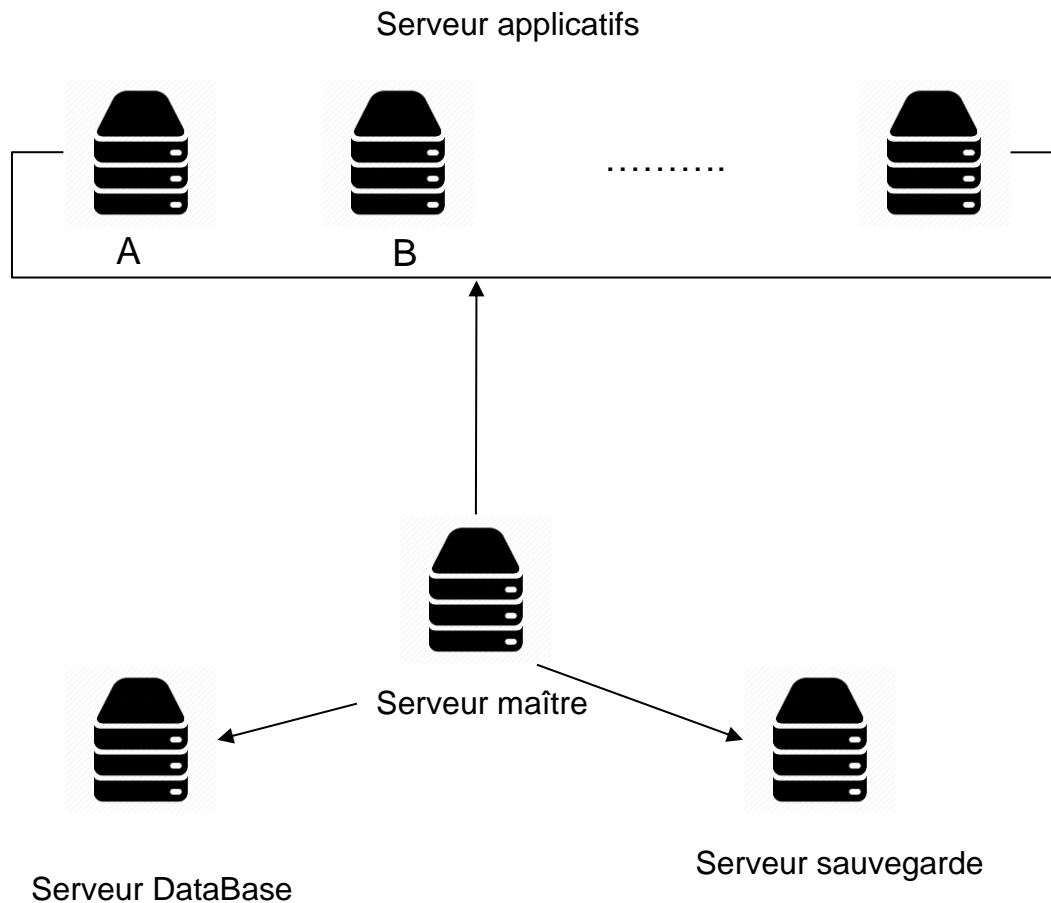
```
GNU nano 4.5 /etc/sudoers
##
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "#include" directives:
#include_dir /etc/sudoers.d
kalimohyi ALL=(ALL) NOPASSWD: ALL
```

II- Partie modélisation

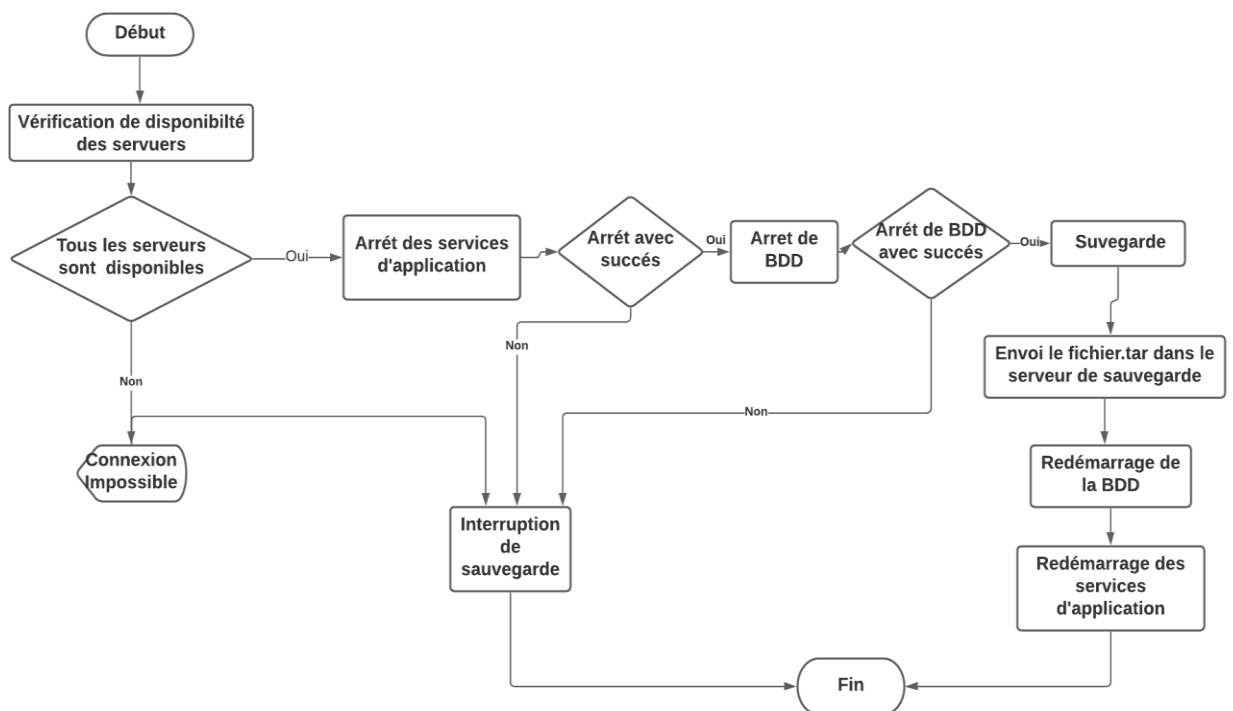
1- Présentation de l'architecture du projet

1.1- Présentation des différents éléments (serveurs, emplacement ...)

- Serveur maître : Est le serveur qui contiendra le programme de sauvegarde et le fichier log. C'est celui qui va naviguer et organiser les tâches entre les autres serveurs, grâce à ce qu'on a vu précédemment.
- Les Serveurs applicatifs : Sont les serveurs qui vont contenir les applications.
- Serveur de base de données : C'est le serveur qui contiendra la base de données (dans notre cas on utilise mysql comme SGBD)
- Serveur de sauvegarde : c'est le serveur qui recevra les sauvegardes quotidiennes de la base de données.



1.2 Fonctionnement du système



- Étape 1 : Vérifier si le serveur maître peut joindre -en ssh- tous les serveurs applicatifs, cette étape est importante avant de procéder à une sauvegarde ; car il faut que tous les serveurs applicatifs participent à la sauvegarde. Ceci se fait grâce à la commande suivante : `$ ssh -q username@IpAdresse`
- Étape 2: Après avoir vérifié que tous les serveurs communiquent entre eux (avec ssh), on passe à la phase d'arrêt des applications. On procède à l'arrêt du serveur le plus dépendant des autres au moins dépendant (Dans le cas le plus simple ou on possède deux serveurs A et B si A dépend de B on Arrêt A puis B).
- Étape 3: Après l'arrêt avec succès des serveurs applicatifs, on arrête la base de données et on compresse le fichier /etc/mysql .
- Étape 4: Ensuite, on envoie ce fichier au serveur de sauvegarde en passant par le serveur maître .
- La dernière étape consiste à redémarrer le serveur de base de données puis les serveurs applicatifs dans le sens inverse d'arrêt.

2- Cas envisageables :

Dans la première étape en cas de non accessibilité à un ou plusieurs serveurs la sauvegarde doit s'arrêter.

Pour la deuxième étape en cas de problème d'arrêt d'un ou de plusieurs serveurs, on commence à redémarrer les serveurs précédents, tout en suivant le démarrage vu dans la partie précédente. (Parler du cas s'il y'a un problème dans le démarrage)

Pour la troisième partie, si un problème est survenu lors de l'arrêt de la base de données, on redémarre les serveurs applicatifs, alors que la sauvegarde est interrompue.

III- Script Shell et mise en marche

1- Présentation des différents éléments utilisés

1.1 Les variables :

L'entrée du script est de type :

InfoServer = (user_1 IpAdress_1 user_2 IpAdress_2 user_n IpAdress_n)

```
InfoServer=(kalimohyi 172.20.10.3 pi 172.20.10.8 )
## ordonné les serveurs dans l'ordre décroissant d'indépendance
```

Le tableau **InfoServer** contient les **user_id** et **ipAddress** des serveurs applicatifs, il s'agit d'une suite de couple "id ip", donc il faut que ce tableau soit rempli des couples **user_id** et **ipAddress** des serveurs applicatifs entrant dans le fonctionnement du système. Ces serveurs doivent être accessible avec ssh sans demande de password (voir dans la partie précédente).

Pour les informations des deux autres serveurs : de base de données et de sauvegarde. Ils sont représentés comme suite :

```
##Server DataBase informations
IDServerDataBase=pi
IPServerDataBase=172.20.10.8
```

```
##Server Backup informations
IDServerSauvegard=kali mohyi
IPServerSauvegard=172.20.10.3
```

On renseigne les informations de notre serveur de base de données en indiquant son **IDuser** et **ipAdress** idem pour le serveur de Backup.

Nombre des serveurs :

```
##Le nombre de serveurs applicatifs
Nbr_Server=2
```

Nbr_Server est le nombre de serveurs applicatifs , autrement dit c'est la taille du tableau **infoServer** sur 2.

```
##the name of service
service=bluetooth
```

Le service ici représente l'application en question qui va s'exécuter dans les serveurs applicatifs (dans notre cas on a testé avec l'application bluetooth).

2.1 Les fonctions :

Notre script contenu dans le fichier SAUVEGARDE.sh est dévisé en 4 parties:

La première partie est constituée de la fonction **stop** (\$IDadresse \$IPadresse \$service). Cette fonction a pour rôle d'arrêter le service du serveur indiqué.

```
stop ${InfoServer[2*$i]} ${InfoServer[2*$i+1]} $service >> /dev/null
```

La fonction a pour première entrée l'identifiant ou le nom du serveur, suivie du deuxième paramètre qui est l'adresse ip du même serveur et le troisième c'est le nom de l'application ou du service. La fonction retourne trois sorties : 1 si un échec est survenu lors de l'arrêt, 0 si l'arrêt a été bien effectué et 2 si un cas imprévu est survenu.

>> /dev/null pour diriger la sortie normale vers le fichier /dev/null .

```
start ${InfoServer[2*$i]} ${InfoServer[2*$i+1]} $service >> /dev/null
```

La deuxième partie est constituée de la fonction **start** (\$IDserver \$IPserver \$service), elle a pour but de redémarrer le service du serveur indiqué par le premier et deuxième paramètre de la fonction. La fonction retourne deux valeurs 0 si on a un démarrage avec succès et 1 sinon.

```
startPreviousServer $NBRM1
```

La troisième est constituée de la fonction **startPreviousServer**(\$indice). Elle redémarre tous les serveurs d'indice inférieur ou égale à \$indice.

Les trois fonctions précédentes écrivent les résultats de leurs sorties dans le fichier log , afin de suivre l'exécution du script.

Pour finir, la partie principale du script contient tous les traitements et les cas possibles à traiter.

```

GNU nano 4.8                               SAUVEGARDE.sh
)
#####
gnome-terminal -- tail -f Sauvegarde.log

echo "#####Tentative de Sauvegarde à $var2#####" >> Sauvegarde.log
echo $NBRM1
echo $Nbr_Server
echo $NBRP1

for i in `seq 0 $NBRM1`
do

    ssh -q ${InfoServer[2*$i]}@${InfoServer[2*$i+1]} echo > /dev/null

    if [ "$?" == "255" ]
    then
        echo "Connection impossible avec ${InfoServer[2*$i]} ${InfoServer[2*$i+1]} " >>Sauvegarde.log

    else
        echo "Connection établie avec ${InfoServer[2*$i]} ${InfoServer[2*$i+1]} " >>Sauvegarde.log
        ((j+=1))
    fi
done

```

echo “chaîne” >>**Sauvegarde.log** permet d’écrire la chaîne dans le fichier Sauvegarde.log

2.3-Commandes importantes :

```
ssh -q $IDServerDataBase@$IPServerDataBase echo > /dev/null
```

C’est la commande qui teste si la connexion ssh est établie ou si le serveur est injoignable. Elle retourne 255 si le serveur est disponible, autres valeurs sinon.

```
(ssh $IDServer@$IPServer sudo systemctl is-active $Application
```

sudo systemctl is-active \$Application donne le statut du service de l’application. Deux cas sont possibles : le service est “**active**” si l’application est en marche - sinon il est “**inactive**”. **Pour** exécuter la commande dans l’un des serveurs, on ajoute cette dernière à la commande ssh.

```
ssh $IDServer@$IPServer sudo systemctl stop $Application
```

sudo systemctl stop \$Application permet d’arrêter l’application passée en paramètre .

```
ssh $IDServer@$IPServer systemctl is-failed $Application >> /dev/null 2>&1
```

systemctl is-failed \$Application est la commande qui teste si le démarrage est bien fait ou pas. Elle retourne 0 si le démarrage est effectué avec succès ,1 sinon.

```
ssh $IDServerDataBase@$IPServerDataBase sudo tar -czf /home/pi/AllSauvegardeDataBase/${NameFileSauvegarde}.tar /var/lib/mysql
```

tar -czf pour compresser le dossier **/var/lib/mysql** contenant toutes les informations du SGBD notamment des bases de données (table etc)

\$(NameFileSauvegarde) désigne le nom attribué à ce fichier compressé, dans notre cas on ajoute la date du jour à “Sauvegarder” pour différencier les sauvegardes ainsi d’éviter l’écrasement des données. Tout cela est exécutée dans le serveur de base de données grâce à la commande ssh.

Le protocole ssh permet d'envoyer et de recevoir des fichiers tout en gardant l’aspect de la sécurité. C’est grâce à la commande **scp** :

```
pi@raspberrypi:~/AllSauvegardeDataBase $ ls
Sauvegarde20-06-2021.tar Sauvegarde21-06-2021.tar Sauvegarde23-06-2021.tar Sauvegarde25-06-2021.tar
pi@raspberrypi:~/AllSauvegardeDataBase $
```

Pour télécharger le fichier compresser dans le serveur maître on spécifie **l’id** et **l’ip** du serveur de base de données. Le fichier **Sauvegardejj-mm--yyyy.tar** va être stocké dans **/home/mohyi/Sauvegard** .

```
scp $IDServerDataBase@$IPServerDataBase:/home/pi/AllSauvegardeDataBase/${NameFileSauvegarde}.tar /home/mohyi/Sauvegard
```

Pour envoyer ce fichier .tar au serveur de sauvegarde on utilise la commande inverse:

```
scp /home/mohyi/Sauvegard/${NameFileSauvegarde}.tar $IDServerSauvegard@$(IPServerSauvegard):/home/kalimohyi/Sauvegarde
```

Le fichier est envoyé dans **/home/kalimohyi/Sauvegarde**.

```
kalimohyi@kaliMohyi:~/Sauvegarde$ ls
Sauvegarde20-06-2021.tar Sauvegarde21-06-2021.tar Sauvegarde23-06-2021.tar Sauvegarde25-06-2021.tar
```



```
#####Tentative de Sauvegarde à 25/06/21_ 8:17:18 pm +01#####
Connection établie avec kalimohyi 172.20.10.3
Connection établie avec pi 172.20.10.8
Connection établie avec le serveur pi 172.20.10.8 de Base de données
Connection établie avec le serveur kalimohyi 172.20.10.3 de sauvegarde
Impossible de sauvegard : un ou plusieurs serveurs sont injoignable en SSH
#####Tentative de Sauvegarde à 25/06/21_ 8:19:12 pm +01#####
Connection établie avec kalimohyi 172.20.10.3
Connection établie avec pi 172.20.10.8
Connection établie avec le serveur pi 172.20.10.8 de Base de données
Connection établie avec le serveur kalimohyi 172.20.10.3 de sauvegarde
Impossible de sauvegard : un ou plusieurs serveurs sont injoignable en SSH
#####Tentative de Sauvegarde à 25/06/21_ 8:21:20 pm +01#####
Connection établie avec kalimohyi 172.20.10.3
Connection établie avec pi 172.20.10.8
Connection établie avec le serveur pi 172.20.10.8 de Base de données
Connection établie avec le serveur kalimohyi 172.20.10.3 de sauvegarde
La disponibilité des serveurs pour la communication SSH est vérifiée avec succès
Start Sauvegard
Vérification du status de bluetooth dans kalimohyi 172.20.10.3
bluetooth arrêté avec succes au kalimohyi 172.20.10.3
Vérification du status de bluetooth dans pi 172.20.10.8
bluetooth arrêté avec succes au pi 172.20.10.8
Start shutdown Database
mariadb arrêté avec succes au pi 172.20.10.8
Mariage avec succès de mariadb au pi 172.20.10.8
Démarrage avec succès de bluetooth au pi 172.20.10.8
Démarrage avec succès de bluetooth au kalimohyi 172.20.10.3
```

IV- Planification et notification de sauvegarde :

Crontab n'est pas préinstallé sur les machines linux, donc on procède à l'installation à l'aide de la commande :

Pour automatiser l'exécution d'un script avec crontab il faut ajouter une ligne contenant deux informations: L'emplacement du script et le temps de l'exécution. La syntaxe est comme suit:

Ce résultat doit être ajouté à un fichier ,pour accéder à ce dernier on exécute la commande :

Dans notre cas, on a programmé l'exécution du programme de sauvegarde SAUEGARDE.sh quotidiennement à 04 : 30 du matin .

```
GNU nano 4.8 /tmp/crontab.L9vC
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
30 04 * * * /home/mohyi/SAUVEGARDE.sh
```

V- Résultats obtenus

Dans notre cas, on a utilisé deux serveurs applicatifs :

pi,172.20.10.8 : qui joue aussi le rôle du serveur de base de données , un SGBD (mysql mariadb) est déjà installé dans ce dernier.

kalimohyi,172.20.10.3 : qui joue le rôle d'un serveur applicatif et qui est notre serveur de sauvegarde. Alors que notre serveur maître est **mohyi@hp**.

Vu qu'on n'a pas suffisamment de machines, c'était une alternative pour tester que tout marche bien. On précise ici qu'on a travaillé dans un réseau local, mais notre script reste valable aussi dans le cas d'un réseau distant. il suffit d'ajouter **l'idserver** et **l'ipserver** dans le tableau des serveurs applicatifs ou même pour le serveur de base de données et de sauvegarde.

```
mohyi@hp:~/ProjetLinux$ ls
Logoproject.txt  Sauvegarde.log  SAUVEGARDE.sh  SauvegardIntermédiaire
mohyi@hp:~/ProjetLinux$
```

Ce répertoire contient le fichier log Sauvegarde.log, le script exécutable SAUVEGARDE.sh et un dossier contenant les sauvegardes téléchargées depuis le serveur de base de données qui est prêt à être envoyé au serveur de sauvegarde. Le fichier **Logoproject** contient le logo du script de notre application.

On exécute le programme :

```
mohyi@hp:~/ProjetLinux$ ./SAUVEGARDE.sh
```

```
ENSEMDATABASE
```

```
tar: Suppression de « / » au début des noms des membres
Sauvegarde25-06-2021.tar          100% 657KB 2.0MB/s 00:00
Sauvegarde25-06-2021.tar          100% 657KB 1.8MB/s 00:00
Sauvegarde Terminer
```

```
mohyi@hp:~/ProjetLinux$ ./SAUVEGARDE.sh
```

```
ENSEMDATABASE
```

```
tar: Suppression de « / » au début des noms des membres
Sauvegarde25-06-2021.tar
Sauvegarde25-06-2021.tar
Sauvegarde Terminer
mohyi@hp:~/ProjetLinux$
```

```
bluetooth arrêté avec succes au pi 172.20.10.8
Start shutdown Database
mariadb arrêté avec succes au pi 172.20.10.8
Démarrage avec succès de mariadb au pi 172.20.10.8
Démarrage avec succès de bluetooth au pi 172.20.10.8
Démarrage avec succès de bluetooth au kalimohyi 172.20.10.3
#####Tentative de Sauvegarde à 25/06/21_10:49:04 pm +01#####
Connection établie avec kalimohyi 172.20.10.3
Connection établie avec pi 172.20.10.8
Connection établie avec le serveur pi 172.20.10.8 de Base de données
Connection établie avec le serveur kalimohyi 172.20.10.3 de sauvegarde
La disponibilité des serveurs pour la communication SSH est vérifiée avec succès

Start Sauvegard
Vérification du status de bluetooth dans kalimohyi 172.20.10.3
bluetooth arrêté avec succes au kalimohyi 172.20.10.3
Vérification du status de bluetooth dans pi 172.20.10.8
bluetooth arrêté avec succes au pi 172.20.10.8
Start shutdown Database
mariadb arrêté avec succes au pi 172.20.10.8
Démarrage avec succès de mariadb au pi 172.20.10.8
Démarrage avec succès de bluetooth au pi 172.20.10.8
Démarrage avec succès de bluetooth au kalimohyi 172.20.10.3
```

Un terminale apparaît pour afficher les phases de sauvegarde, ces phases sont stockés d'une manière permanente dans le fichier Sauvegarde.log.

```
GNU nano 4.8                               Sauvegarde.log
#####Tentative de Sauvegarde à 25/06/21_10:46:39 pm +01#####
Connection établie avec kalimohyi 172.20.10.3
Connection établie avec pi 172.20.10.8
Connection établie avec le serveur pi 172.20.10.8 de Base de données
Connection établie avec le serveur kalimohyi 172.20.10.3 de sauvegarde
La disponibilité des serveurs pour la communication SSH est vérifiée avec succès
Start Sauvegard
Vérification du status de bluetooth dans kalimohyi 172.20.10.3
bluetooth arrêté avec succes au kalimohyi 172.20.10.3
Vérification du status de bluetooth dans pi 172.20.10.8
bluetooth arrêté avec succes au pi 172.20.10.8
Start shutdown Database
mariadb arrêté avec succes au pi 172.20.10.8
Démarrage avec succès de mariadb au pi 172.20.10.8
Démarrage avec succès de bluetooth au pi 172.20.10.8
Démarrage avec succès de bluetooth au kalimohyi 172.20.10.3
#####Tentative de Sauvegarde à 25/06/21_10:49:04 pm +01#####
Connection établie avec kalimohyi 172.20.10.3
Connection établie avec pi 172.20.10.8
Connection établie avec le serveur pi 172.20.10.8 de Base de données
Connection établie avec le serveur kalimohyi 172.20.10.3 de sauvegarde
La disponibilité des serveurs pour la communication SSH est vérifiée avec succès
Start Sauvegard
Vérification du status de bluetooth dans kalimohyi 172.20.10.3
bluetooth arrêté avec succes au kalimohyi 172.20.10.3
Vérification du status de bluetooth dans pi 172.20.10.8
bluetooth arrêté avec succes au pi 172.20.10.8
Start shutdown Database
mariadb arrêté avec succes au pi 172.20.10.8
Démarrage avec succès de mariadb au pi 172.20.10.8
Démarrage avec succès de bluetooth au pi 172.20.10.8
Démarrage avec succès de bluetooth au kalimohyi 172.20.10.3
```

```
^C Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo        M-A Mark Text    M-] To Bracket
^X Exit          ^R Read File    ^V Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line    M-E Redo        M-6 Copy Text   ^Q Where Was
```

Conclusion :

Pour conclure, nous avons eu cette chance de travailler sur ce projet de l'administration des systèmes Linux en tant que des étudiants en première année en génie informatique à l'ENSEM. Cela nous a permis d'acquérir, d'échanger et de partager nos compétences et nos acquis après avoir terminé le cours de l'administration des systèmes Linux, et ainsi s'aider mutuellement.

Une fois face à une problématique, nous effectuons un travail d'équipe afin d'aboutir à de nombreuses solutions pour y extraire la solution idéale. Cette approche que nous avons adoptée, consiste à partager les tâches pour rendre avantageux le travail d'équipe en matière de temps, nouvelles perspectives et gestion des problèmes.

Pour finir, ce projet nous a donné une petite idée sur ce qui nous attend dans le monde de l'entreprise

Bibliographie et webographie :

[Utilisez les clefs ssh pour des connexions sans mot de passe – Buzut](#)

[\(1425\) How to use sudo su root | sudo no password | visudo - YouTube](#)

[sudo - Sudoers file, enable NOPASSWD for user, all commands - Ask Ubuntu](#)

[La connexion sécurisée à distance avec SSH - Reprenez le contrôle à l'aide de Linux ! - OpenClassrooms](#)

[SSH : Installer et configurer un serveur SSH - Wiki - Wiki](#)

[Transfert de fichier via SSH | IT-Connect](#)

[Debian - Autoriser l'accès root via SSH - cloriou.fr](#)

[Comment utiliser Systemctl pour gérer les services et les unités de Systemd | DigitalOcean](#)

[Install a MariaDB server on CentOS -](#)

[Start, Stop & Restart Services in Ubuntu and Other Linux](#)

[bash - In the shell, what does " 2>&1 " mean? - Stack Overflow](#)

[Debian - Autoriser l'accès root via SSH - cloriou.fr](#)

[MySQL :: MySQL 5.0 Reference Manual :: 2.5.2.2 Lancer et arrêter MySQL automatiquement](#)

Annexe :

```
#!/bin/bash
InfoServer=(kalimohyi 172.20.10.3 pi 172.20.10.8 )
## ordonné les serveurs dans l'ordre décroissant d'indépendance
##Le nombre de serveurs applicatifs
Nbr_Server=2
##Server DataBase informations
IDServerDataBase=pi
IPServerDataBase=172.20.10.8

##Server Backup informations
IDServerSauvegard=kalimohyi
IPServerSauvegard=172.20.10.3

##the name of service
service=bluetooth

#date +%d-%m-%Y"
#var=$date
var=`date +%d-%m-%Y"`

#date +%x_%r"
#var2=$date
var2=`date +%x_%r"`
NameFileSauvegarde="Sauvegarde${var}"

let NBRM1=$Nbr_Server-1 ##nombre serveur moin 1
let NBRP1=$Nbr_Server+1      ## nombre de serveur plus 1
let NBRP2=$Nbr_Server+2    ##nombre de servuer plus 2

j=0
k=0
T=0
function Stop(){
IDServer=$1
IPServer=$2
Application=$3
SleepTime=5

ligne=$(ssh $IDServer@$IPServer sudo systemctl is-active $Application)
```

```

if [ "$ligne" = "active" ]
then
    ssh $IDServer@$IPServer sudo systemctl stop $Application
    sleep $SleepTime

    LIG=$(ssh $IDServer@$IPServer sudo systemctl is-active $Application)

    if [ "$LIG" = "active" ]
    then
        echo "Échec d'arrêt de $Application au $IDServer $IPServer " >>Sauvegarde.log
        return 1

    elif [ "$LIG" = "inactive" ]
    then
        echo "$Application arrêté avec succes au $IDServer $IPServer " >>Sauvegarde.log
        return 0

    else
        echo "Cas imprévue l'ors de l'arrêt de $Application au serveur $IDServer $IPServer "
        >>Sauvegarde.log
        return 2
    fi

elif [ "$ligne" = "inactive" ] ; then echo "$Application est déjà arrêté au $IDServer
$IPServer"

else
return 2

fi
}

function start(){

IDServer=$1
IPServer=$2
Application=$3
SleepTime=5

ssh $IDServer@$IPServer sudo systemctl start $Application

ssh $IDServer@$IPServer systemctl is-failed $Application >> /dev/null 2>&1
echo $?
if [ "$?" == "0" ]

```

```

then
    echo "Démarrage avec succès de $Application au $IDServer $IPServer "
>>Sauvegarde.log
    return 0
else
    echo "Échec démarrage de $Application au serveur $IDServer $IPServer "
>>Sauvegarde.log
    return 1
fi
}

function startPreviousServer(){

Indice=$1

for i in `seq $Indice -1 0`
do

start ${InfoServer[2*$i]} ${InfoServer[2*$i+1]} $service >> /dev/null
done
}
#####

gnome-terminal -- tail -f Sauvegarde.log

echo "#####Tentative de Sauvegarde à $var2##### " >> Sauvegarde.log

echo $NBRM1
echo $Nbr_Server
echo $NBRP1

for i in `seq 0 $NBRM1`
do
    ssh -q ${InfoServer[2*$i]}@${InfoServer[2*$i+1]} echo > /dev/null

    if [ "$?" == "255" ]

    then
        echo "Connection impossible avec ${InfoServer[2*$i]}
${InfoServer[2*$i+1]} " >>Sauvegarde.log
    else
        echo "Connection établie avec ${InfoServer[2*$i]} ${InfoServer[2*$i+1]} "
>>Sauvegarde.log
    fi
done
}

```



```

        ((j+=1))
    fi
done
ssh -q $IDServerDataBase@$IPServerDataBase echo > /dev/null

    if [ "$?" == "255" ]
    then
        echo "Connection impossible avec $IDServerDataBase $IPServerDataBase "
>>Sauvegarde.log
    else
        echo "Connection établie avec $IDServerDataBase $IDServerDataBase "
>>Sauvegarde.log

        ((j+=1))
    fi
ssh -q $IDServerSauvegard@$IPServerSauvegard echo > /dev/null

    if [ "$?" == "255" ]
    then
        echo "Connection impossible avec $IDServerSauvegard $IPServerSauvegard
" >>Sauvegarde.log
    else
        echo "Connection établie avec $IDServerSauvegard $IPServerSauvegard "
>>Sauvegarde.log

        ((j+=1))
    fi

if [ "$j" == "$NBRP2" ]
then
    echo "La disponibilité des serveurs pour la communication SSH est vérifiée avec succès "
>>Sauvegarde.log
    echo "Start Sauvegard" >>Sauvegarde.log

    let check=$NBRP1
    i=0
    while [ $i -le $NBRM1 ]
    do
        if [ "$check" == "$NBRP1" ]
        then

            echo "Vérification du status de $service dans ${InfoServer[2*$i]}
${InfoServer[2*$i+1]} ">>Sauvegarde.log

```

```

        ligne=$(ssh ${InfoServer[2*$i]}@${InfoServer[2*$i+1]} systemctl is-active
$service )

        if [ "$ligne" = "active" ]
            then

                Stop ${InfoServer[2*$i]} ${InfoServer[2*$i+1]} $service >>
/dev/null

                if [ "$?" == "0" ]
                    then
                        ((T++))

                    else
                        check=$i
                        echo "Problème d'arrêt $service au serveur
${InfoServer[2*$i]} ${InfoServer[2*$i+1]}" >>Sauvegarde.log

                        fi

                        elif [ "$ligne" = "inactive" ]
                            then
                                echo " L'application $service est déjà en arrêt au serveur
${InfoServer[2*$i]} ${InfoServer[2*$i+1]}" >>Sauvegarde.log

                                check=$i
                                echo $check

                            fi
                        else
                            break
                        fi
                    ((i++))

                done
                if [ "$check" != "0" ] && [ $check -le $Nbr_Server ]
                    then
                        echo "Intéruption du sauvgarde : redémarrage des applications " >>Sauvegarde.log
                        startPreviousServer $check >>/dev/null

                    fi

                if [ "$check" == "0" ]
                    then

```

```

        echo "Intéruption du sauvgarde : redémarrage des applications " >>Sauvegarde.log
        start ${InfoServer[0]} ${InfoServer[1]} $service >> /dev/null
    fi

    if [ "$T" == "$Nbr_Server" ]
    then
        echo " Start shutdown Database " >>Sauvegarde.log

        Stop $IDServerDataBase $IPServerDataBase mariadb

        if [ "$?" == "0" ]
        then
            ##start sauvegard
            ssh $IDServerDataBase@$IPServerDataBase sudo tar -czf
/home/pi/AllSauvegardeDataBase/${NameFileSauvegarde}.tar /var/lib/mysql
            scp
$IDServerDataBase@$IPServerDataBase:/home/pi/AllSauvegardeDataBase/${NameFileSau
vegarde}.tar /home/mohyi/Sauvegard
            scp /home/mohyi/Sauvegard/${NameFileSauvegarde}.tar
$IDServerSauvegard@${IPServerSauvegard}:/home/kalimohyi/Sauvegarde

            start $IDServerDataBase $IPServerDataBase mariadb >> /dev/null
            startPreviousServer $NBRM1
            echo "Sauvegarde Terminer"

        else
            echo "Échec d'arrêt de mariadb" >>Sauvegarde.log
            echo "Intéruption du sauvegarde : redémarrage des applications "
>>Sauvegarde.log

            StartPreviousServer $NBRM1

            echo "Error: Vérifie Sauvegarde.log"
        fi
    else
        echo "arret Sauvegaed vérifier les Serveurs "
        echo " Vérifie Sauvegarde.log"
    fi

else
    echo "Impossible de sauvegard : un ou plusieurs serveurs sont injoignable en SSH"
>>Sauvegarde.log
    echo " Error: Vérifie Sauvegarde.log"

```

```
fi  
exit
```