

Compte rendu sauvegarde :

Introduction : Dans ce projet, nous allons détailler la mise en place d'un système de sauvegarde automatique sous Linux en utilisant Rsync et SSH. L'objectif est d'assurer la protection des données en les sauvegardant localement et à distance, tout en permettant leur restauration rapide en cas de perte.

Ce choix s'est imposé car la gestion des sauvegardes est un enjeu crucial pour assurer la protection des données, que ce soit dans un cadre personnel ou professionnel. De plus, la perte de données peut avoir des conséquences graves, notamment en entreprise, où les fichiers critiques doivent être protégés contre les erreurs humaines, les cyberattaques ou les pannes matérielles.

Méthodologie et choix techniques :

- **Rsync** : pour synchroniser efficacement les fichiers et les répliquer sur d'autres machines.
- **SSH** : pour garantir un transfert sécurisé des données sur un serveur distant.
- **Python avec Tkinter et Flask** : pour développer une interface utilisateur graphique (locale et web).

Répartition des rôles :

Axel Macé :

Étape 2 : Installation de Rsync

Étape 3 : Création du script de sauvegarde

Étape 4 : Mise en place de la sauvegarde automatique (Crontab)

Étape 5 : Restauration des données

Vittorio Gandossi :

Étape 6 : Installation et configuration de SSH

Étape 7 : Configuration du client pour les sauvegardes à distance

Étape 10 : Mise en place de la sauvegarde sur plusieurs serveurs

Organisation : Mis-en-page et correction des comptes-rendus

Mathys Urban :

Étape 8 : Création de l'interface utilisateur Web

Étape 9 : Création de l'interface utilisateur en application

Étape 10 : Vérification et tests de sauvegarde/restauration

Étape 11 et 12 : Mise en place de la sauvegarde divisée et de la restauration

Table des matières :

Méthodologie et choix techniques :	1
Répartition des rôles :	2
Table des matières :.....	3
Étape 1 : Prérequis et configuration :.....	7
Image 1 : Mis-à-jour du système Linux.....	7
Étape 2 : Installation Rsync :.....	8
Image 2 : Installation de Rsync.....	8
Étape 3 : Fichier script :.....	9
Image 3 : Création du fichier “sauvegarde.sh”	9
Image 4 : Edition du fichier “sauvegarde.sh”	9
Image 5 : Mis-à-jour des permissions du fichier “sauvegarde.sh”.....	9
Image 6 : Exécution du fichier “sauvegarde.sh”	9
Image 7 : Ancien emplacement du fichier “sauvegarde.sh”.....	10
Image 8 : Nouvel emplacement du fichier “sauvegarde.sh”	10
Étape 4 : Tâche automatique :.....	11
Image 9 : Éditer des tâches planifiées avec crontab.....	11
Image 10 : Choix de l’heure pour la planification.....	11
Image 11 : Emplacement du fichier “sauvegarde.sh” avant l’exécution de la tâche automatique.....	11
Image 12 : Emplacement du fichier “sauvegarde.sh” après l’exécution de la tâche automatique.....	11
Étape 5 : Restauration des données :.....	12
Image 13 : Sauvegarde et synchronisation avec rsync.....	12
Image 14 : Suppression du fichier “test.txt”	12
Image 15 : Exécution de la commande rsync.....	12
Étape 6 : Sauvegarde à distance :.....	13
Image 16 : Installation du serveur SSH.....	13
Image 17 : Démarrage de SSH.....	13
Image 18 : Ajout d’un nouvel utilisateur nommé “backupuser”	14
Image 19 : Attribution des droits d’administrateur à backupuser.....	14
Image 20 : Création d’un dossier “sauvegardes” et attribution des droits de backupuser sur ce dossier.....	14
Étape 7 : Configuration du client :.....	15
Image 21 : Connexion au serveur SSH distant avec l’utilisateur “backupuser”... 15	
Image 22 : Création du fichier “sauvegarde_client.sh”	15
Image 23 : Edition du fichier “sauvegarde_client.sh”	15
Image 24 : Mis-à-jour des permissions du fichier “sauvegarde_client.sh”	15

Image 25 : Exécution du fichier “sauvegarde_client.sh”	16
Image 26 : Ancien emplacement de “Test.txt”	16
Image 27 : Nouvel emplacement de “Test.txt”	16
Image 28 : Éditer des tâches planifiées crontab avec backupuser	17
Image 29 : Choix de l’heure pour la planification	17
Image 30 : Restauration du dossier local grâce au serveur distant	17
Étape 8 : Création d’une interface (web) :	18
Image 31 : Installation de Python et Flask	18
Image 32 : Création et mis-à-jour des droits du fichier “interface_web_sauvegarde.py”	18
Image 33 : Edition du fichier “interface_web_sauvegarde.py”	19
Image 34 : Lancement du serveur Flask	19
Image 35 : Affichage du serveur Flask	20
Image 36 : Sauvegarde réussi	20
Image 37 : Restauration réussi	20
Étape 9 : Création d’une interface (ordinateur) :	21
Image 38 : Installation Tkinter	21
Image 39 : Création et mis-à-jour des droits du fichier “interface_sauvegardeTk.py”	21
Image 40 : Edition du fichier “interface_sauvegardeTk.py”	21
Image 41 : Exécution du fichier “interface_sauvegardeTk.py”	22
Image 42 : Affichage de l’interface et sauvegarde réussi	22
Image 43 : Restauration réussi	22
Image 44 : Création du raccourcis d’application	23
Image 45 : Édition du raccourcis d’application	23
Image 46 : Mis-à-jour des droits du raccourcis d’application	23
Image 47 : Aperçu du raccourci d’application pour ouvrir l’interface de gestion des sauvegardes	23
Étape 10 : Sauvegarde sur divers serveurs :	24
Image 48 : Installation du serveur SSH	24
Image 49 : Ajout d’une clé SSH vers le serveur distant	24
Image 50 : Connexion SSH au serveur distant	25
Image 51 : Mise en réseau privé	25
Image 52 : Edition du fichier de sauvegarde	26
Image 53 : Edition du fichier de restauration	26
Image 54 : Edition du fichier “interface_sauvegardeTk.py” partie sauvegarde et restauration	27
Image 55 : Edition du fichier “interface_sauvegardeTk.py” partie interface graphique	27
Image 56 : Edition du fichier “interface__web_sauvegarde.py” partie interface	

web.....	28
Image 57 : Edition du fichier “interface__web_sauvegarde.py” partie routes et commandes.....	28
Image 58 : Ancien emplacement de Test txt.txt.....	29
Image 59 : Nouvel emplacement de Test txt.txt.....	29
Étape 11 : Système de sauvegarde divisée :.....	30
Image 60 : Création d’un script de fusion des sauvegardes et de restauration (restore_merge.sh).....	30
Image 61 : Création d’un script de division de la sauvegarde, puis répartition sur 2 serveurs (backup_split.sh).....	31
Image 62 : Édition du fichier “interface__web_sauvegarde.py” pour inclure le nouveau système de sauvegarde au bouton correspondant.....	32
Image 63 : Édition du fichier “interface_sauvegarde_tk.py”.....	33
Image 64 : Test réussi du bon fonctionnement du bouton “sauvegarde” sur les 2 serveurs.....	33
Image 65 : Vérification de la présence du fichier Test txt (Copy).txt dans le bon dossier.....	34
Image 66 : Renommage du fichier “sauvegarde.sh” en “sauvegarde3VM.sh”.....	34
Image 67 : Edition du fichier “backup_split.sh” pour y ajouter la sauvegarde en totalité sur le 3e serveur.....	35
Image 68 : Edition du fichier “restore_merge.sh” pour y ajouter la sauvegarde en totalité sur le 3e serveur.....	36
Image 69 : Premier test : sauvegarde partielle sur le serveur 1 et 2, et sauvegarde entière sur le serveur 3.....	37
Image 70 : Deuxième test : restauration des données du serveur 1 et 2, à l’aide du serveur 3.....	38
Image 71 : Troisième test : restauration des données du serveur 3, à l’aide du serveur 1 et 2.....	38
Conclusion :.....	39

Étape 1 : Prérequis et configuration :

Dans ce projet, nous allons tout d'abord mettre en place notre machine virtuelle(VM) avec Oracle VirtualBox avec comme système d'exploitation Linux, avec une carte réseau en Bridge pour avoir une adresse IP accessible au réseau.

Puis on va s'assurer que notre VM est déjà à jour en utilisant la commande « sudo apt install & upgrade ».

```
vboxuser@LinuxVPN:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for vboxuser:
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
213 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  libllvm17t64 python3-netifaces
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  libllvm19 libmalcontent-0-0 mesa-libgallium
The following upgrades have been deferred due to phasing:
```

Image 1 : Mis-à-jour du système Linux

Étape 2 : Installation Rsync :

On va maintenant, pour ce début de projet, installer Rsync pour faire des sauvegardes de données, il s'agit d'un programme basique qui est facile à comprendre et d'utilisation. On va utiliser la commande « `sudo apt install` » pour installer ce programme.

```
vboxuser@Linuxsave:~$ sudo apt update && sudo apt install rsync -y
Hit:1 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
rsync is already the newest version (3.2.7-1ubuntu1.2).
rsync set to manually installed.
The following packages were automatically installed and are no longer required:
  libllvm17t64 python3-netifaces
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

Image 2 : Installation de Rsync

Maintenant qu'on a installé Rsync, on va créer notre script pour pouvoir enregistrer les données.

Étape 3 : Fichier script :

Pour créer un fichier script, on va utiliser la commande « nano » pour créer ou modifier le fichier.

```
vboxuser@Linuxsave:~$ sudo nano sauvegarde.sh
```

Image 3 : Création du fichier "sauvegarde.sh"

Puis on va ajouter dans ce fichier script, une commande où l'on va préserver les fichiers avec leurs permissions et les déplacer de documents vers le dossier « backup ».

```
#!/bin/bash  
rsync -av --delete /home/user/documents/ /backup/
```

Image 4 : Edition du fichier "sauvegarde.sh"

On n'oublie pas de donner les permissions avec la commande « chmod » pour exécuter notre programme.

```
vboxuser@Linuxsave:~$ sudo chmod +x sauvegarde.sh
```

Image 5 : Mis-à-jour des permissions du fichier "sauvegarde.sh"

Puis, on exécute notre fichier avec cette commande.

```
vboxuser@Linuxsave:~$ ./sauvegarde.sh  
sending incremental file list  
./  
TEst.txt  
  
sent 154 bytes  received 38 bytes  384.00 bytes/sec  
total size is 5  speedup is 0.03
```

Image 6 : Exécution du fichier "sauvegarde.sh"

On peut voir après que notre fichier texte a été déplacé depuis le dossier documents sur le dossier backup.



Image 7 : Ancien emplacement du fichier “sauvegarde.sh”



Image 8 : Nouvel emplacement du fichier “sauvegarde.sh”

Maintenant qu’on a sauvegardé manuellement nos données, nous allons les sauvegarder automatiquement.

Étape 4 : Tâche automatique :

Maintenant on va réaliser une tâche automatique où l'on va sauvegarder automatiquement les données vers un temps précis comme 2 heures du matin. Pour cela, on va utiliser la commande « crontab » et choisir nano comme éditeur de script.

```
vboxuser@Linuxsave:~$ crontab -e
no crontab for vboxuser - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano          <---- easiest
 2. /usr/bin/vim.tiny
 3. /bin/ed

Choose 1-3 [1]: 1
crontab: installing new crontab
```

Image 9 : Éditer des tâches planifiées avec crontab

Puis on ajoute cette ligne pour que nos données soient sauvegardées automatiquement.

```
0 2 * * * /home/vboxuser/sauvegarde.sh
```

Image 10 : Choix de l'heure pour la planification

Puis, si on règle l'heure à 2h du matin, on voit que le fichier « Test2.txt » a été déplacé automatiquement sur le dossier « backup ».

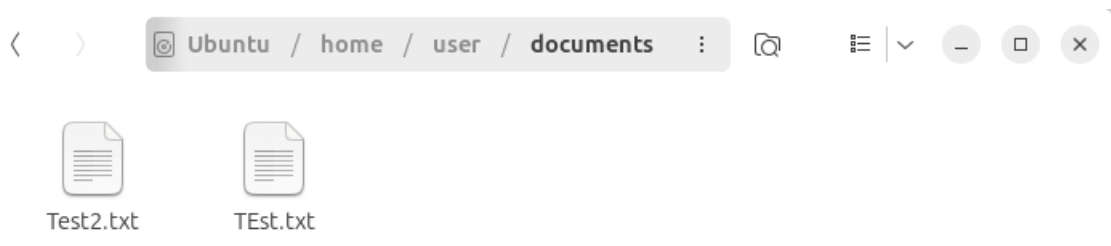


Image 11 : Emplacement du fichier “sauvegarde.sh” avant l'exécution de la tâche automatique



Image 12 : Emplacement du fichier “sauvegarde.sh” après l'exécution de la tâche automatique

Maintenant, il faut trouver un moyen de restaurer les données sauvegardées de manière rapide si on perd les fichiers originaux.

Étape 5 : Restauration des données :

Pour restaurer les données, on va utiliser la commande `rsync` avec « `-av` », pour récupérer les fichiers et leurs permissions respectivement depuis le dossier « `backup` », puis ils seront copiés et déplacés sur le dossier « `documents` ».

```
vboxuser@Linuxsave:/home/user/documents$ sudo rsync -av /backup/ /home/user/documents/  
sending incremental file list  
./  
TEst.txt  
  
sent 166 bytes  received 38 bytes  408.00 bytes/sec  
total size is 11  speedup is 0.05
```

Image 13 : Sauvegarde et synchronisation avec `rsync`

Comme on peut le voir, avant de faire la commande, j'ai supprimé le fichier « `test.txt` » du dossier « `document` », puis j'ai exécuté la commande, et maintenant notre fichier « `test.txt` » a bien été restauré.

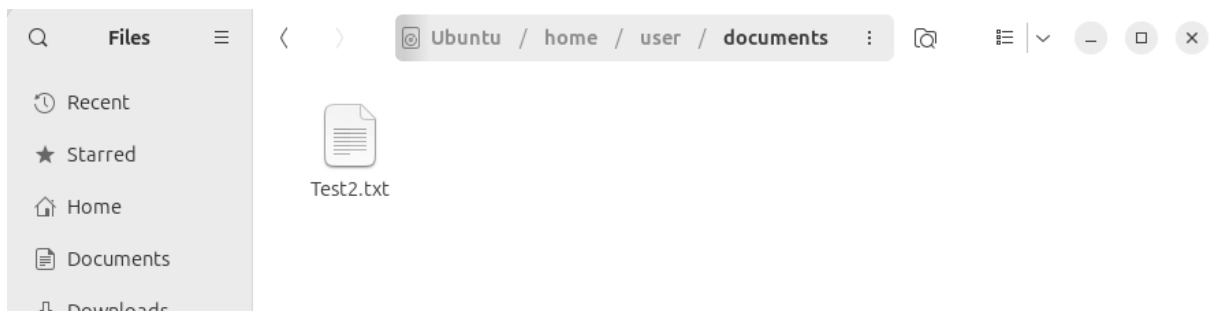


Image 14 : Suppression du fichier “`test.txt`”

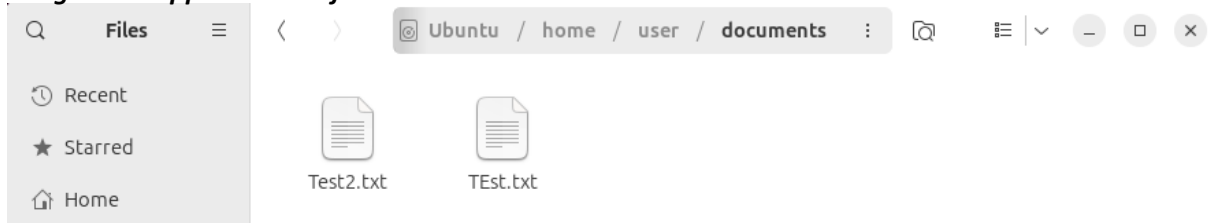


Image 15 : Exécution de la commande `rsync`

Maintenant, il faut pouvoir sauvegarder ses données à distance depuis une machine cliente.

Étape 6 : Sauvegarde à distance :

Pour faire une sauvegarde à distance, on va utiliser SSH pour envoyer les données de manière sécurisée et parce qu'il est intégré avec Rsync. Pour cela, on va utiliser la même commande « `sudo apt install` » pour installer le programme.

```
vboxuser@Linuxsave:~$ sudo apt install openssh-server -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libllvm17t64 python3-netifaces
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh_askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 10 not upgraded.
Need to get 832 kB of archives.
After this operation, 6,747 kB of additional disk space will be used.
Get:1 http://fr.archive.ubuntu.com/ubuntu noble-updates/main amd64 openssh-sftp
```

Image 16 : Installation du serveur SSH

Puis on active le service avec « `systemctl` ».

```
vboxuser@Linuxsave:~$ sudo systemctl enable --now ssh
Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
Created symlink /etc/systemd/system/ssh.service → /usr/lib/systemd/system/ssh.s
```

Image 17 : Démarrage de SSH

Ensuite, on va créer un nouvel utilisateur client avec « `adduser` ».

```
vboxuser@Linuxsave:~$ sudo adduser backupuser
info: Adding user `backupuser' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `backupuser' (1001) ...
info: Adding new user `backupuser' (1001) with group `backupuser (1001)' ..
info: Creating home directory `/home/backupuser' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for backupuser
Enter the new value, or press ENTER for the default
```

Image 18 : Ajout d'un nouvel utilisateur nommé "backupuser"

On donne à cet utilisateur des droits d'administrateur en l'ajoutant sur le groupe « sudo ».

```
vboxuser@Linuxsave:~$ sudo usermod -aG sudo backupuser
```

Image 19 : Attribution des droits d'administrateur à backupuser

Et enfin, on crée un dossier on donne à backupuser, toutes les permissions à ce dossier.

```
vboxuser@Linuxsave:~$ sudo mkdir -p /home/backupuser/sauvegardes
vboxuser@Linuxsave:~$ sudo chown backupuser:backupuser /home/backupuser/sauvegardes
```

Image 20 : Création d'un dossier "sauvegardes" et attribution des droits de backupuser sur ce dossier

Étape 7 : Configuration du client :

Maintenant, depuis le client, on va configurer l'adresse du serveur pour que les données soient synchronisées.

```
backupuser@Linuxsave:~$ sudo ssh backupuser@10.0.2.15
[sudo] password for backupuser:
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.
ED25519 key fingerprint is SHA256:TmX4RWyNN5C40lUkVh42nre4uLBPDDQAap1IzC1UYJo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.15' (ED25519) to the list of known hosts.
backupuser@10.0.2.15's password:
Permission denied, please try again.
backupuser@10.0.2.15's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-52-generic x86_64)
```

Image 21 : Connexion au serveur SSH distant avec l'utilisateur "backupuser"

Maintenant, on va créer un fichier script avec la commande « nano » pour sauvegarder les données avec le serveur.

```
backupuser@Linuxsave:~$ sudo nano sauvegarde_client.sh
```

Image 22 : Création du fichier "sauvegarde_client.sh"

Puis, on va créer notre script pour sauvegarder les données au serveur.

```
#!/bin/bash

# Variables
SRC="/home/backupuser/Documents/"
DEST="backupuser@10.0.2.15:/home/backupuser/sauvegardes/"

# Lancer la sauvegarde via Rsync
rsync -avz --delete -e "ssh" "$SRC" "$DEST"

echo "Sauvegarde terminée avec succès."
```

Image 23 : Edition du fichier "sauvegarde_client.sh"

On donne toutes les permissions au fichier et on l'exécute.

```
sudo chmod +x sauvegarde_client.sh
```

Image 24 : Mis-à-jour des permissions du fichier "sauvegarde_client.sh"

```
backupuser@Linuxsave:~$ ./sauvegarde_client.sh
backupuser@10.0.2.15's password:
sending incremental file list
./
Test.txt

sent 152 bytes  received 38 bytes  25.33 bytes/sec
total size is 6  speedup is 0.03
Sauvegarde terminée avec succès.
```

Image 25 : Exécution du fichier “sauvegarde_client.sh”

On peut voir que notre fichier Test.txt a bien été déplacé vers l'utilisateur du serveur.

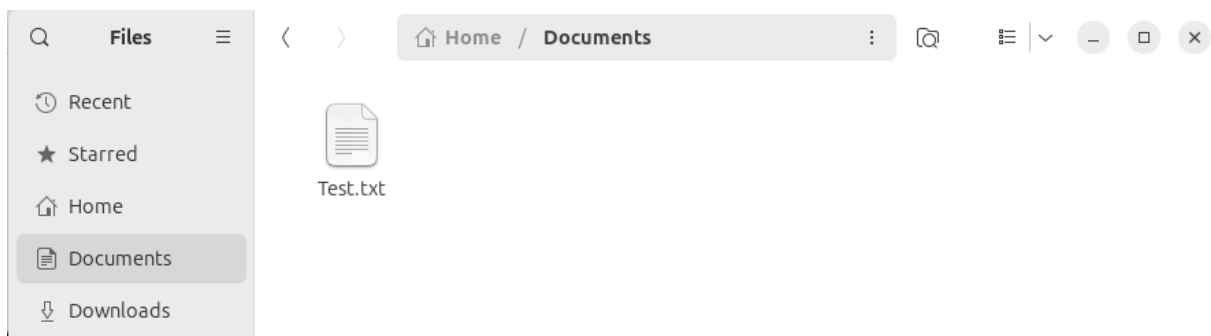


Image 26 : Ancien emplacement de “Test.txt”



Image 27 : Nouvel emplacement de “Test.txt”

Maintenant, On sauvegarde les données du client automatiquement avec « crontab ».

```
backupuser@Linuxsave:~$ crontab -e
no crontab for backupuser - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.tiny
 3. /bin/ed
```

Image 28 : Éditer des tâches planifiées crontab avec backupuser

```
#
# m h dom mon dow  command
0 3 * * * /home/user/sauvegarde_client.sh >> /home/user/sauvegarde.log 2>&1
```

Image 29 : Choix de l'heure pour la planification

Maintenant que les sauvegardes sont faites automatiquement, on doit pouvoir les restaurer avec cette commande.

```
backupuser@Linuxsave:~$ rsync -avz backupuser@10.0.2.15:/home/backupuser/sauvegardes/ /home/backupuser/Documents/
backupuser@10.0.2.15's password:
receiving incremental file list
./
Test.txt

sent 46 bytes  received 148 bytes  35.27 bytes/sec
total size is 6  speedup is 0.03
```

Image 30 : Restauration du dossier local grâce au serveur distant

Étape 8 : Création d'une interface (web) :

Pour créer notre interface, on va installer Python avec un module flask qui sera nécessaire pour l'affichage graphique. Il est recommandé pour sa comptabilité, et son intégration avec divers outils.

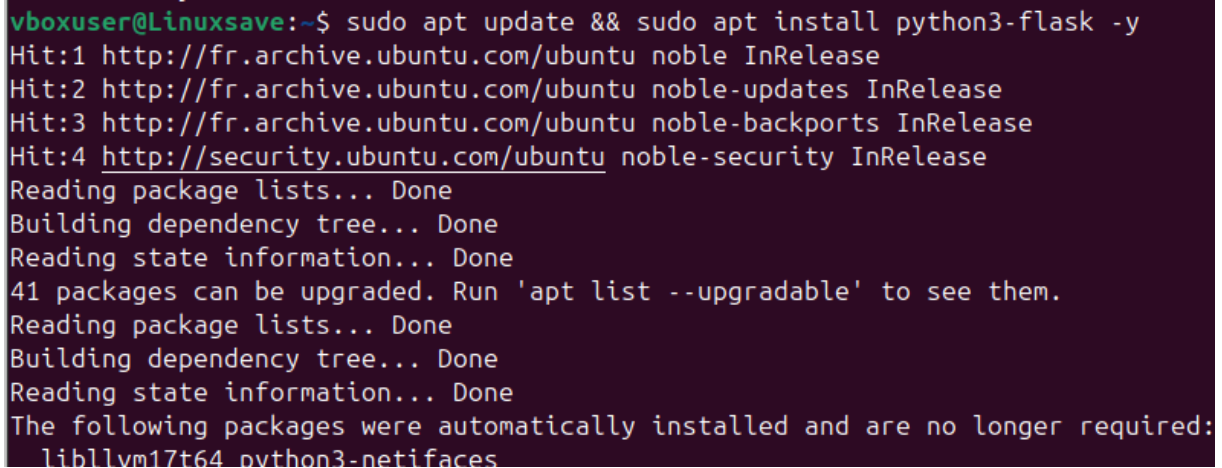
A terminal window with a dark purple background. The prompt is 'vboxuser@Linuxsave:~\$'. The command 'sudo apt update && sudo apt install python3-flask -y' has been executed. The output shows four 'Hit' messages for different Ubuntu mirrors, followed by 'Reading package lists... Done', 'Building dependency tree... Done', and 'Reading state information... Done'. It then states '41 packages can be upgraded. Run \'apt list --upgradable\' to see them.' and repeats the dependency and state information steps. Finally, it lists packages that were automatically installed and are no longer required: 'liblvm17t64 python3-netifaces'.

Image 31 : Installation de Python et Flask

Puis on va créer un fichier avec « nano » qui consiste à créer un site web pour sauvegarder et dont on va lui donner toutes les permissions.

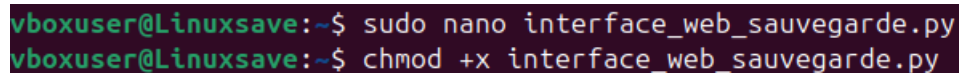
A terminal window with a dark purple background. The prompt is 'vboxuser@Linuxsave:~\$'. The command 'sudo nano interface_web_sauvegarde.py' has been executed, followed by 'vboxuser@Linuxsave:~\$ chmod +x interface_web_sauvegarde.py'.

Image 32 : Création et mis-à-jour des droits du fichier "interface_web_sauvegarde.py"

Puis on crée le script suivant en Python.

```
from flask import Flask, render_template
import subprocess

app = Flask(__name__)

@app.route('/')
def index():
    return '''
        <h1>Gestion de Sauvegarde</h1>
        <button onclick="location.href='/sauvegarde'">Sauvegarder</button>
        <button onclick="location.href='/restauration'">Restaurer</button>
    '''

@app.route('/sauvegarde')
def sauvegarde():
    try:
        subprocess.run(["rsync", "-av", "~/Documents/", "~/backup/"], check=True)
        return "Sauvegarde effectuée avec succès !"
    except subprocess.CalledProcessError:
        return "Échec de la sauvegarde."
```

Image 33 : Edition du fichier "interface_web_sauvegarde.py"

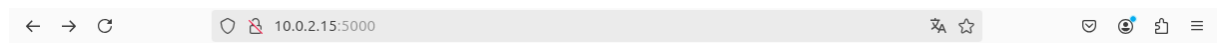
Maintenant, si on va vers l'interface web, on devrait avoir notre page pour la restauration et sauvegarde :

```
vboxuser@Linuxsave:~$ python3 interface_web_sauvegarde.py
* Serving Flask app 'interface_web_sauvegarde'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead
* Running on http://10.0.2.15:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 216-365-974
10.0.2.15 - - [17/Feb/2025 11:01:13] "GET / HTTP/1.1" 200 -
sending incremental file list
./
Test txt.txt

sent 148 bytes received 38 bytes 372.00 bytes/sec
total size is 10 speedup is 0.05
10.0.2.15 - - [17/Feb/2025 11:01:16] "GET /sauvegarde HTTP/1.1" 200 -
sending incremental file list
./
test2.txt

sent 188 bytes received 38 bytes 452.00 bytes/sec
total size is 20 speedup is 0.09
10.0.2.15 - - [17/Feb/2025 11:02:16] "GET /restauration HTTP/1.1" 200 -
```

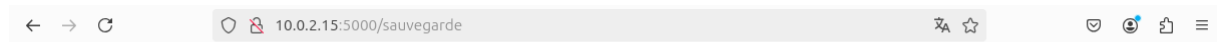
Image 34 : Lancement du serveur Flask



Gestion de Sauvegarde

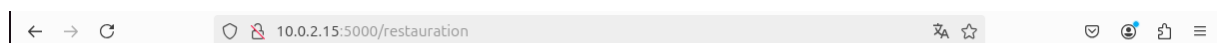
[Sauvegarder](#) [Restaurer](#)

Image 35 : Affichage du serveur Flask



Sauvegarde effectuée avec succès !

Image 36 : Sauvegarde réussie



Restauration effectuée avec succès !

Image 37 : Restauration réussie

Étape 9 : Création d'une interface (ordinateur) :

On va tout d'abord installer tkinter pour Python afin d'avoir une interface graphique, car elle est légère et adaptée pour une application locale.

```
vboxuser@Linuxsave:~$ sudo apt update && sudo apt install python3-tk -y-y
t:1 http://fr.archive.ubuntu.com/ubuntu noble InRelease
t:2 http://security.ubuntu.com/ubuntu noble-security InRelease
t:3 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
t:4 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
ading package lists... Done
ilding dependency tree... Done
ading state information... Done
packages can be upgraded. Run 'apt list --upgradable' to see them.
ading package lists... Done
ilding dependency tree... Done
ading state information... Done
```

Image 38 : Installation Tkinter

Puis on va créer un fichier pour l'interface et lui donner les permissions.

```
vboxuser@Linuxsave:~$ sudo nano interface_sauvegarde Tk.py
vboxuser@Linuxsave:~$ chmod +x interface_sauvegarde Tk.py
```

Image 39 : Création et mis-à-jour des droits du fichier "interface_sauvegarde Tk.py"

Puis on va mettre le script pour l'interface :

```
import tkinter as tk
from tkinter import messagebox
import subprocess

def sauvegarde():
    try:
        subprocess.run(["sudo", "rsync", "-av", "/home/vboxuser/Documents/", "/home/backupuser/backup/"], check=True)
        messagebox.showinfo("Succès", "Sauvegarde effectuée avec succès !")
    except subprocess.CalledProcessError:
        messagebox.showerror("Erreur", "Échec de la sauvegarde.")

def restauration():
    try:
        subprocess.run(["sudo", "rsync", "-av", "/home/backupuser/backup/", "/home/vboxuser/Documents/"], check=True)
        messagebox.showinfo("Succès", "Restauration effectuée avec succès !")
    except subprocess.CalledProcessError:
        messagebox.showerror("Erreur", "Échec de la restauration.")

# Création de l'interface
tk_root = tk.Tk()
tk_root.title("Gestion Sauvegarde")
tk_root.geometry("300x200")

tk.Label(tk_root, text="Gestion de sauvegarde", font=("Arial", 14)).pack(pady=10)

tk.Button(tk_root, text="Sauvegarder", command=sauvegarde).pack(pady=5)
tk.Button(tk_root, text="Restaurer", command=restauration).pack(pady=5)
```

Image 40 : Edition du fichier "interface_sauvegarde Tk.py"

Si on exécute notre programme, on devrait avoir l'interface qui apparaît.

```
vboxuser@Linuxsave:~$ python3 interface_sauvegardeTk.py

sending incremental file list
./
test2.txt

sent 185 bytes  received 38 bytes  446.00 bytes/sec
total size is 17  speedup is 0.08
sending incremental file list

sent 128 bytes  received 12 bytes  280.00 bytes/sec
total size is 17  speedup is 0.12
```

Image 41 : Exécution du fichier "interface_sauvegardeTk.py"

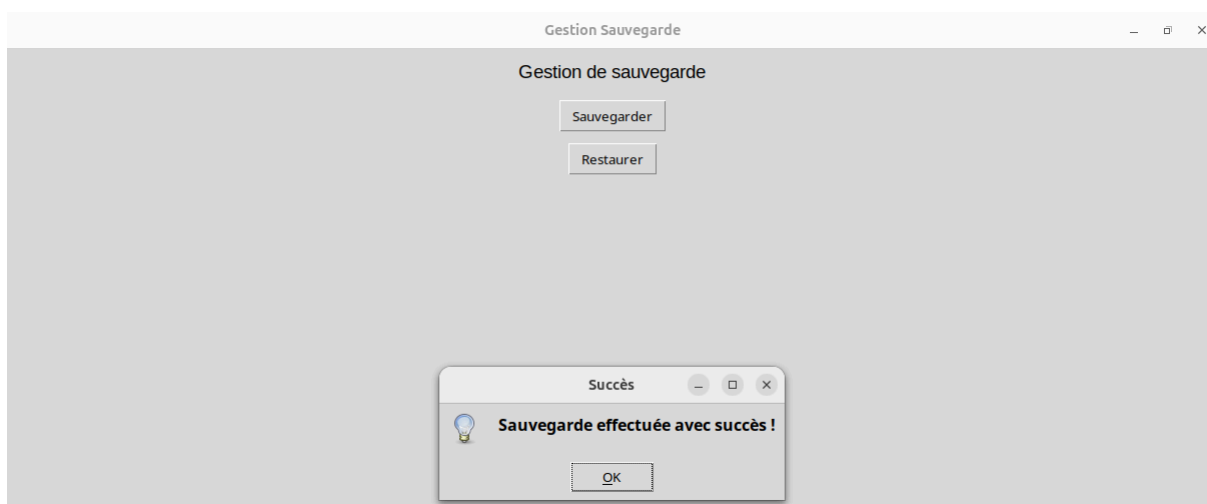


Image 42 : Affichage de l'interface et sauvegarde réussie



Image 43 : Restauration réussie

En bonus, tu peux faire un raccourci bureau pour l'interface de sauvegarde avec « nano ».

```
vboxuser@Linuxsave:~$ nano ~/Desktop/Sauvegarde.desktop
```

Image 44 : Création du raccourcis d'application

```
[Desktop Entry]
Version=1.0
Type=Application
Name=Interface Sauvegarde
Comment=Ouvre l'interface de sauvegarde
Exec=sh -c 'x-terminal-emulator -e "sudo /usr/bin/python3 /home/vboxuser/interface_sauvegarde_tk.py"'
Icon=utilities-terminal
Terminal=false
Categories=Utility;
```

Image 45 : Édition du raccourcis d'application

Puis, vous autorisez les permissions et vous pouvez voir que notre interface en raccourci apparaît.

```
vboxuser@Linuxsave:~$ chmod +x ~/Desktop/Sauvegarde.desktop
```

Image 46 : Mis-à-jour des droits du raccourcis d'application

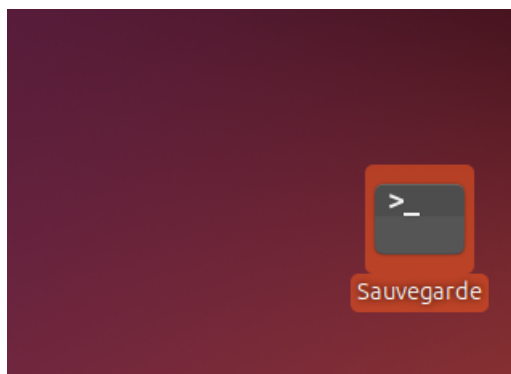


Image 47 : Aperçu du raccourci d'application pour ouvrir l'interface de gestion des sauvegardes

Étape 10 : Sauvegarde sur divers serveurs :

Maintenant, on va faire fonctionner la sauvegarde sur deux serveurs à distance, on va mettre en place deux VM pour cette étape. On va installer OpenSSL SSH.

```
vboxuser@Linuxsaveserver2:~$ sudo apt update && sudo apt install openssh-server -y
Warning: The unit file, source configuration file or drop-ins of apt-news.service changed on disk. Run 'systemctl daemon-reload' to reload units.
Warning: The unit file, source configuration file or drop-ins of esm-cache.service changed on disk. Run 'systemctl daemon-reload' to reload units.
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [670 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [919 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [130 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [9,
```

Image 48 : Installation du serveur SSH

Puis on va créer et copier notre clé sur les deux VMs afin qu'elles puissent communiquer.

```
vboxuser@Linuxsaveserver2:~$ ssh-copy-id vboxuser@10.0.2.15
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/vboxuser/.ssh/id_rsa.pub"
/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: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
vboxuser@10.0.2.15's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'vboxuser@10.0.2.15'"
and check to make sure that only the key(s) you wanted were added.
```

Image 49 : Ajout d'une clé SSH vers le serveur distant

On peut tester avec la commande SSH suivante pour savoir si les deux VMs communiquent et ont leur clé SSH.

```
vboxuser@Linuxsaveserver2:~$ ssh vboxuser@10.0.2.15
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.
ED25519 key fingerprint is SHA256:MbRai3Dvdg7boCAgZM7jstrePUwjY8cJD7rUJr5oOp8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.15' (ED25519) to the list of known hosts.
vboxuser@10.0.2.15's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.11.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

228 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

Image 50 : Connexion SSH au serveur distant

Après vérification, on va devoir mettre nos VMs en réseau privé host car elles partagent une seule et même adresse. Avec ce changement de réseau, les informations de sauvegarde pourront être correctement reçues. Il va juste falloir refaire les copies SSH avec les nouvelles IP de chaque VM.

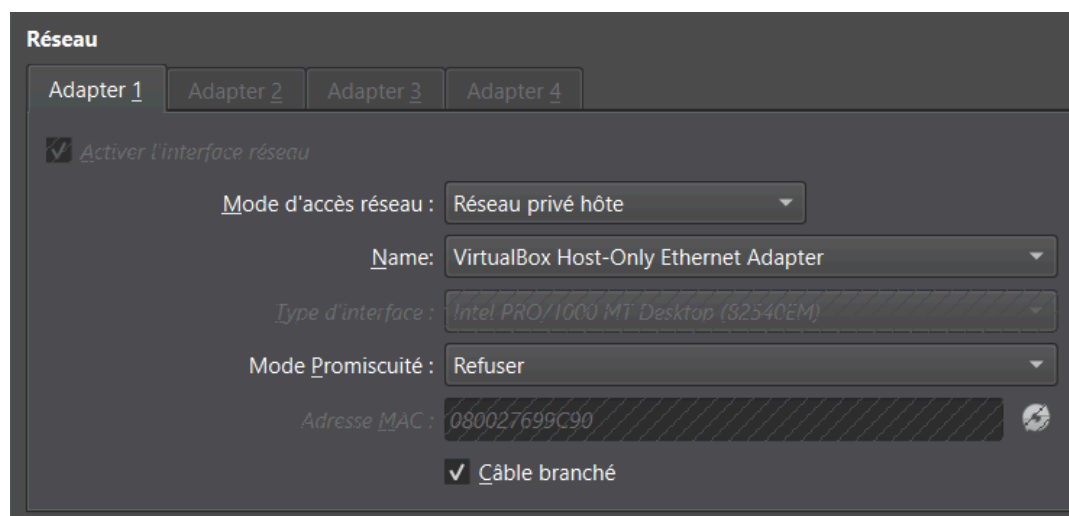


Image 51 : Mise en réseau privé

Maintenant, il va falloir modifier nos programmes pour qu'ils sauvegardent sur les IP des VM, pour cela on va tout d'abord modifier le fichier de restauration et le fichier de sauvegarde.

```
#!/bin/bash

SRC="/home/vboxuser/Documents/"
DEST1="vboxuser@192.168.56.102:/home/vboxuser/sauvegardes/"
DEST2="vboxuser@192.168.56.103:/home/vboxuser/sauvegardes/"

rsync -avz --delete -e "ssh" "$SRC" "$DEST1"
rsync -avz --delete -e "ssh" "$SRC" "$DEST2"

echo "Sauvegarde terminée avec succès sur VM2 et VM3."
```

Image 52 : Edition du fichier de sauvegarde

```
#!/bin/bash

SOURCE1="vboxuser@192.168.56.102:/home/vboxuser/sauvegardes/"
SOURCE2="vboxuser@192.168.56.103:/home/vboxuser/sauvegardes/"
DEST="/home/vboxuser/Documents/"

echo "Restaurer depuis :"
echo "1) VM2"
echo "2) VM3"
read choix

if [ "$choix" == "1" ]; then
    rsync -avz --delete -e "ssh" "$SOURCE1" "$DEST"
    echo "Restauration depuis VM2 terminée."
elif [ "$choix" == "2" ]; then
    rsync -avz --delete -e "ssh" "$SOURCE2" "$DEST"
    echo "Restauration depuis VM3 terminée."
else
    echo "Choix invalide."
fi
```

Image 53 : Edition du fichier de restauration

Maintenant, il faut modifier les programmes de l'application pour sauvegarder et restaurer l'application, et aussi pour la version web :

```
GNU nano 7.2 interface_sauvegarde Tk.py
import tkinter as tk
from tkinter import messagebox
import subprocess

def sauvegarde():
    try:
        subprocess.run(["pkexec", "/home/vboxuser/sauvegarde3VM.sh"], check=True)
        messagebox.showinfo("Succès", "Sauvegarde effectuée avec succès sur VM2 et VM3.")
    except subprocess.CalledProcessError:
        messagebox.showerror("Erreur", "Échec de la sauvegarde.")

def restauration():
    vm = var_vm.get()
    if vm == "VM2":
        cmd = ["pkexec", "rsync", "-av", "vboxuser@192.168.56.102:/home/vboxuser/sauvegardes/", "/home/vboxuser/Document"]
    elif vm == "VM3":
        cmd = ["pkexec", "rsync", "-av", "vboxuser@192.168.56.103:/home/vboxuser/sauvegardes/", "/home/vboxuser/Document"]
    else:
        messagebox.showerror("Erreur", "Veuillez choisir une source de restauration.")
        return

    try:
        subprocess.run(cmd, check=True)
        messagebox.showinfo("Succès", f"Restauration effectuée avec succès depuis {vm}.")
    except subprocess.CalledProcessError:
        messagebox.showerror("Erreur", "Échec de la restauration.")

# Interface Tkinter
```

Image 54 : Edition du fichier "interface_sauvegarde Tk.py" partie sauvegarde et restauration

```
# Interface Tkinter
tk_root = tk.Tk()
tk_root.title("Gestion Sauvegarde")
tk_root.geometry("300x250")

tk.Label(tk_root, text="Gestion de sauvegarde", font=("Arial", 14)).pack(pady=10)
tk.Button(tk_root, text="Sauvegarder", command=sauvegarde).pack(pady=5)

tk.Label(tk_root, text="Restaurer depuis :", font=("Arial", 12)).pack(pady=5)
var_vm = tk.StringVar(value="VM2")
tk.Radiobutton(tk_root, text="VM2", variable=var_vm, value="VM2").pack()
tk.Radiobutton(tk_root, text="VM3", variable=var_vm, value="VM3").pack()
tk.Button(tk_root, text="Restaurer", command=restauration).pack(pady=5)

tk_root.mainloop()
```

Image 55 : Edition du fichier "interface_sauvegarde Tk.py" partie interface graphique

```

GNU nano 7.2                                interface_web_sauvegarde.py
from flask import Flask, render_template, request
import subprocess

app = Flask(__name__)

@app.route('/')
def index():
    return '''
        <h1>Gestion de Sauvegarde</h1>
        <button onclick="location.href='/sauvegarde'">Sauvegarder</button>
        <br><br>
        <form action="/restauration" method="post">
            <label for="vm">Restaurer depuis :</label>
            <select name="vm">
                <option value="VM2">192.168.56.102</option>
                <option value="VM3">192.168.56.103</option>
            </select>
            <button type="submit">Restaurer</button>
        </form>
    '''

```

Image 56 : Edition du fichier "interface__web_sauvegarde.py" partie interface web

```

@app.route('/sauvegarde')
def sauvegarde():
    try:
        subprocess.run(["sudo", "/home/vboxuser/sauvegarde3VM.sh"], check=True)
        return "Sauvegarde effectuée avec succès sur VM2 et VM3."
    except subprocess.CalledProcessError:
        return "Échec de la sauvegarde."

@app.route('/restauration', methods=['POST'])
def restauration():
    vm = request.form['vm']
    try:
        if vm == "192.168.56.102":
            subprocess.run(["sudo", "rsync", "-av", "vboxuser@VM2:/home/vboxuser/sauvegardes/", "/home/vboxuser/Documents/"])
        elif vm == "192.168.56.103":
            subprocess.run(["sudo", "rsync", "-av", "vboxuser@VM3:/home/vboxuser/sauvegardes/", "/home/vboxuser/Documents/"])
        return f"Restauration effectuée avec succès depuis {vm}."
    except subprocess.CalledProcessError:
        return "Échec de la restauration."

if __name__ == '__main__':
    app.run(host='192.168.56.101', port=5000, debug=True)

```

Image 57 : Edition du fichier "interface__web_sauvegarde.py" partie routes et commandes

Et enfin, on vérifie que la sauvegarde et la restauration fonctionnent correctement.



Image 58 : Ancien emplacement de Test txt.txt

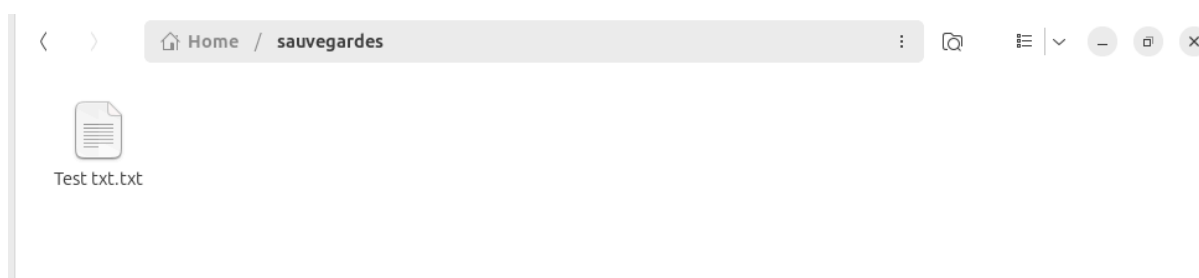


Image 59 : Nouvel emplacement de Test txt.txt

Étape 11 : Système de sauvegarde divisée :

Dans cette étape, nous avons séparé les sauvegardes pour qu'en cas de problème avec le serveur sur lequel on travaille (ou un des autres serveurs), l'entièreté des données puisse être récupérée.

L'autre raison pour laquelle on divise les données, c'est pour éviter le ralentissement sur le serveur de travail qui est constamment utilisé, ou sur celui de sauvegarde pour éviter de devoir faire des sauvegardes qui durent bien trop longtemps.

Commençons par sauvegarder nos fichiers en divisant en deux la taille pour que l'un soit sauvegardé sur un serveur et de l'autre sur un serveur différent.

Pour ce faire, on va devoir modifier en premier les fichiers de sauvegarde et restauration, que l'on a réalisés précédemment.

```
#!/bin/bash

TMP_DIR="/tmp/restore_merge"
DEST="/home/vboxuser/Documents/"
SRC1="vboxuser@192.168.56.102:/home/vboxuser/sauvegardes/"
SRC2="vboxuser@192.168.56.103:/home/vboxuser/sauvegardes/"

# Nettoyage
echo "[INFO] Préparation du dossier temporaire de restauration..."
rm -rf "$TMP_DIR"
mkdir -p "$TMP_DIR"

# Récupération des deux dossiers de sauvegarde
rsync -avz "$SRC1" "$TMP_DIR/part1"
rsync -avz "$SRC2" "$TMP_DIR/part2"

# Fusion des fichiers restaurés dans le dossier Documents
cp -r "$TMP_DIR/part1/*" "$DEST"
cp -r "$TMP_DIR/part2/*" "$DEST"

# Nettoyage final
rm -rf "$TMP_DIR"
echo "✅ Restauration fusionnée terminée depuis VM2 et VM3."
```

Image 60 : Création d'un script de fusion des sauvegardes et de restauration (restore_merge.sh)

```
#!/bin/bash

SRC="/home/vboxuser/Documents/"
TMP_DIR="/tmp/backup_split"
PART1="$TMP_DIR/part1"
PART2="$TMP_DIR/part2"
DEST1="vboxuser@192.168.56.102:/home/vboxuser/sauvegardes/"
DEST2="vboxuser@192.168.56.103:/home/vboxuser/sauvegardes/"

# Nettoyage et préparation
echo "[INFO] Préparation du dossier temporaire..."
rm -rf "$TMP_DIR"
mkdir -p "$PART1" "$PART2"

# Répartition des fichiers
i=0
for file in "$SRC"*; do
    if [ $((i % 2)) -eq 0 ]; then
        cp -r "$file" "$PART1/"
    else
        cp -r "$file" "$PART2/"
    fi
    i=$((i + 1))
done

# Envoi direct des fichiers vers les VMs
rsync -avz "$PART1/" "$DEST1"
rsync -avz "$PART2/" "$DEST2"

# Nettoyage final
rm -rf "$TMP_DIR"
echo "✅ Sauvegarde divisée envoyée sur VM2 et VM3."
```

Image 61 : Création d'un script de division de la sauvegarde, puis répartition sur 2 serveurs (backup_split.sh)

Puis, on modifie les interfaces web et l'application pour inclure ce nouveau système de sauvegarde.

```
from flask import Flask
import subprocess

app = Flask(__name__)

@app.route('/')
def index():
    return '''
    <h1>Gestion de Sauvegarde</h1>
    <button onclick="location.href='/sauvegarde'">Sauvegarder</button>
    <br><br>
    <button onclick="location.href='/restauration'">Restaurer</button>
    '''

@app.route('/sauvegarde')
def sauvegarde():
    try:
        subprocess.run(["sudo", "/home/vboxuser/sauvegarde3VM.sh"], check=True)
        return "Sauvegarde divisée effectuée avec succès sur VM1 et VM2."
    except subprocess.CalledProcessError:
        return "Échec de la sauvegarde."

@app.route('/restauration')
def restauration():
    try:
        subprocess.run(["sudo", "/home/vboxuser/restauration3VM.sh"], check=True)
        return "Restauration fusionnée réussie depuis VM1 et VM2."
    except subprocess.CalledProcessError:
        return "Échec de la restauration."

if __name__ == '__main__':
    app.run(host='192.168.56.103', port=5000, debug=True)
```

Image 62 : Édition du fichier “interface__web_sauvegarde.py” pour inclure le nouveau système de sauvegarde au bouton correspondant.


```

import tkinter as tk
from tkinter import messagebox
import subprocess

def sauvegarde():
    try:
        subprocess.run(["pkexec", "/home/vboxuser/sauvegarde3VM.sh"], check=True)
        messagebox.showinfo("Succès", "Sauvegarde divisée envoyée sur VM1 et VM2.")
    except subprocess.CalledProcessError:
        messagebox.showerror("Erreur", "Échec de la sauvegarde.")

def restauration():
    try:
        subprocess.run(["pkexec", "/home/vboxuser/restauration3VM.sh"], check=True)
        messagebox.showinfo("Succès", "Restauration fusionnée réussie depuis VM1 et VM2.")
    except subprocess.CalledProcessError:
        messagebox.showerror("Erreur", "Échec de la restauration.")

# Interface Tkinter
tk_root = tk.Tk()
tk_root.title("Gestion Sauvegarde")
tk_root.geometry("300x200")

tk.Label(tk_root, text="Gestion de sauvegarde", font=("Arial", 14)).pack(pady=10)
tk.Button(tk_root, text="Sauvegarder", command=sauvegarde).pack(pady=5)
tk.Button(tk_root, text="Restaurer", command=restauration).pack(pady=5)

tk_root.mainloop()

```

Image 63 : Édition du fichier "interface_sauvegarde_tk.py"

Maintenant, on peut voir que la sauvegarde fonctionne parfaitement :

```

vboxuser@192.168.56.103's password:
sending incremental file list
./
Test txt (Copy).txt

sent 129 bytes  received 44 bytes  23.07 bytes/sec
total size is 10  speedup is 0.06
✓ Sauvegarde divisée envoyée sur VM2 et VM3.

```

Image 64 : Test réussi du bon fonctionnement du bouton “sauvegarde” sur les 2 serveurs.

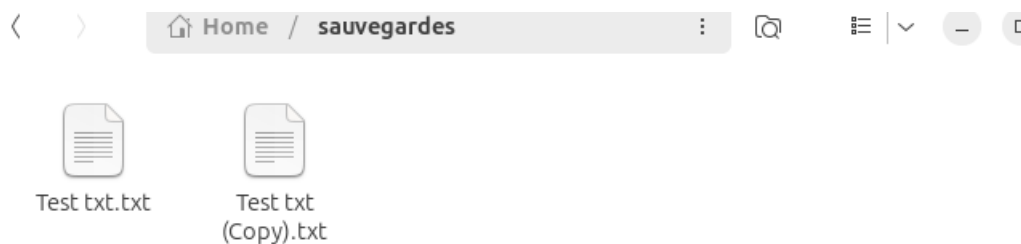


Image 65 : Vérification de la présence du fichier Test txt (Copy).txt dans le bon dossier

De plus, étant donné que j’ai utilisé ou renommé le fichier pour la sauvegarde à distance. Pour les 3 VM, on change la crontab avec le nom du bon fichier script.

```
0 2 * * * /home/vboxuser/sauvegarde3VM.sh
```

Image 66 : Renommage du fichier “sauvegarde.sh” en “sauvegarde3VM.sh”

Étape 12 : Amélioration du système de sauvegarde et restauration :

Bien que sauvegarder 2 moitié des données sur 2 serveurs différents permette d'éviter de ralentir le serveur sur lequel nous travaillons et évite la perte totale de données, nous avons décidé d'aller plus loin en utilisant le 3e serveur uniquement pour restaurer les 2 autres en cas de problème majeur.

Ainsi, peu importe quel serveur tombe en panne, toutes les données seront toujours présentes.

Voici comment nous avons réparti les sauvegardes des 2 premiers serveurs et l'entièreté des données sur le troisième serveur.

```
#!/bin/bash

SRC="/home/vboxuser/Documents/"
TMP_DIR="/tmp/backup_split"
PART1="$TMP_DIR/part1"
PART2="$TMP_DIR/part2"
DEST1="vboxuser@192.168.56.101:/home/vboxuser/sauvegardes/" # VM1
DEST2="vboxuser@192.168.56.102:/home/vboxuser/sauvegardes/" # VM2
DEST3="vboxuser@192.168.56.103:/home/vboxuser/sauvegarde_complet/" # VM3 (sauvegarde complète)

# Nettoyage
echo "[INFO] Préparation du dossier temporaire..."
rm -rf "$TMP_DIR"
mkdir -p "$PART1" "$PART2"

# Division des fichiers
i=0
for file in "$SRC"*; do
    if [ $((i % 2)) -eq 0 ]; then
        cp -r "$file" "$PART1/"
    else
        cp -r "$file" "$PART2/"
    fi
    i=$((i + 1))
done

# Sauvegardes réparties
rsync -avz "$PART1/" "$DEST1"
rsync -avz "$PART2/" "$DEST2"

# Sauvegarde complète sur VM3
rsync -avz "$SRC" "$DEST3"

# Nettoyage
rm -rf "$TMP_DIR"
echo "✅ Sauvegarde divisée sur VM1/VM2 et complète sur VM3."
```

Image 67 : Edition du fichier "backup_split.sh" pour y ajouter la sauvegarde en totalité sur le 3e serveur.

```

#!/bin/bash

TMP_DIR="/tmp/restore_merge"
DEST="/home/vboxuser/Documents/"
SRC1="vboxuser@192.168.56.101:/home/vboxuser/sauvegardes/"
SRC2="vboxuser@192.168.56.102:/home/vboxuser/sauvegardes/"
SRC3="vboxuser@192.168.56.103:/home/vboxuser/sauvegarde_complet/"

rm -rf "$TMP_DIR"
mkdir -p "$TMP_DIR/part1" "$TMP_DIR/part2"

echo "[INFO] Tentative de restauration répartie (VM1 + VM2)..."

SUCCESS1=false
SUCCESS2=false

rsync -avz "$SRC1" "$TMP_DIR/part1" && SUCCESS1=true
rsync -avz "$SRC2" "$TMP_DIR/part2" && SUCCESS2=true

if $SUCCESS1 && $SUCCESS2; then
    cp -r "$TMP_DIR/part1/"* "$DEST"
    cp -r "$TMP_DIR/part2/"* "$DEST"
    echo "✅ Restauration fusionnée réussie depuis VM1 et VM2."
else
    echo "[WARNING] Échec de la restauration répartie. Tentative depuis VM3..."
    rm -rf "$DEST"/*
    rsync -avz "$SRC3" "$DEST" && echo "✅ Restauration complète réussie depuis VM3." || echo "❌ Échec de la restauration depuis VM3."
fi

rm -rf "$TMP_DIR"

```

Image 68 : Edition du fichier “restore_merge.sh” pour y ajouter la sauvegarde en totalité sur le 3e serveur.

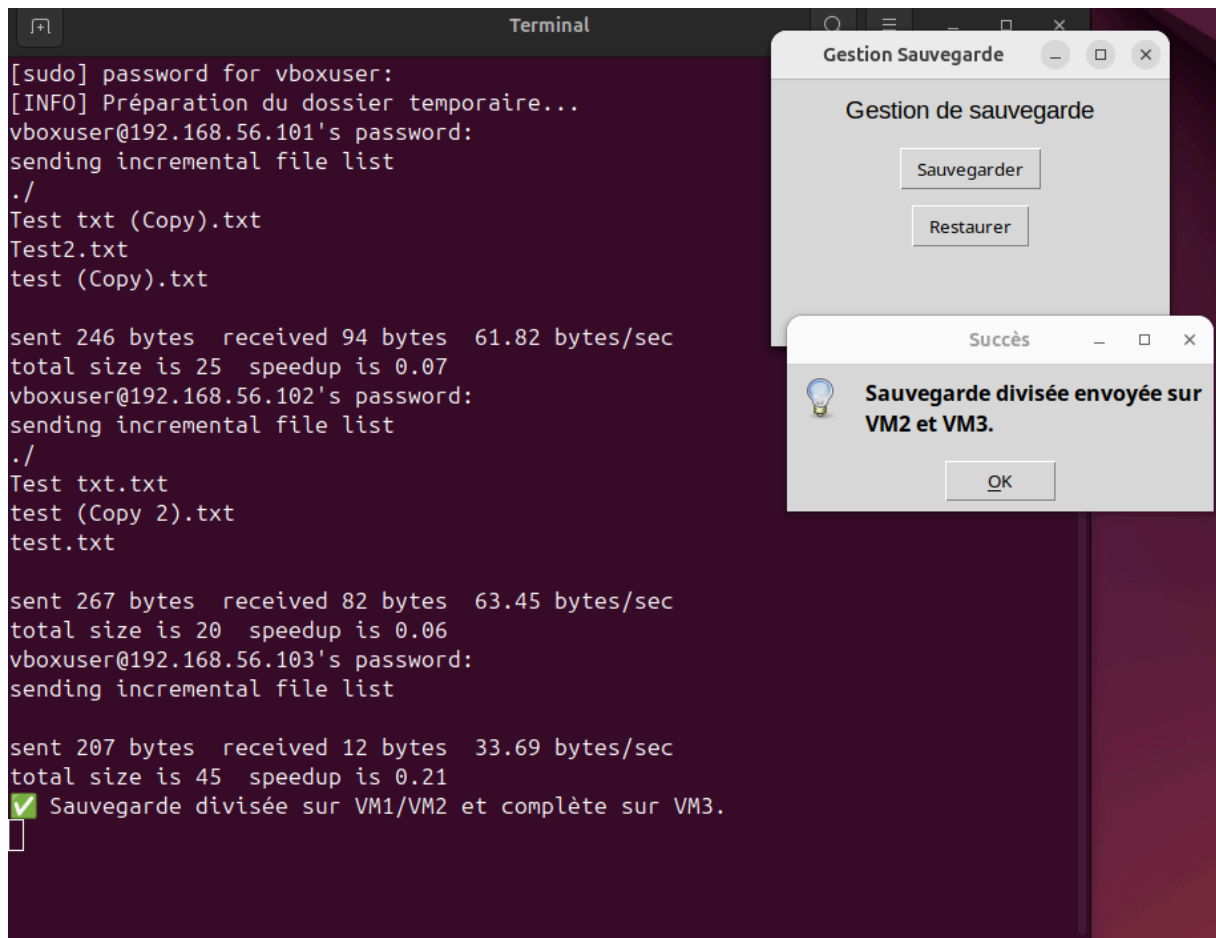


Image 69 : Premier test : sauvegarde partielle sur le serveur 1 et 2, et sauvegarde entière sur le serveur 3

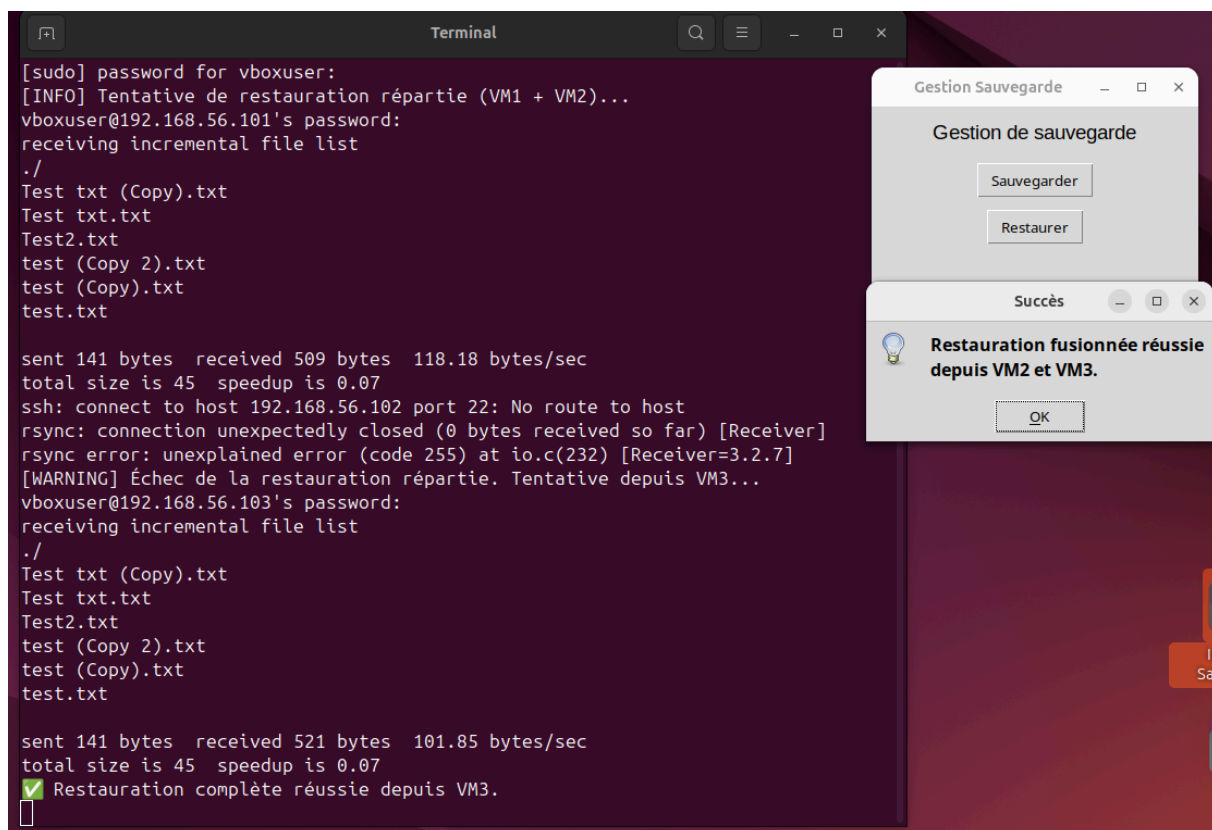


Image 70 : Deuxième test : restauration des données du serveur 1 et 2, à l'aide du serveur 3

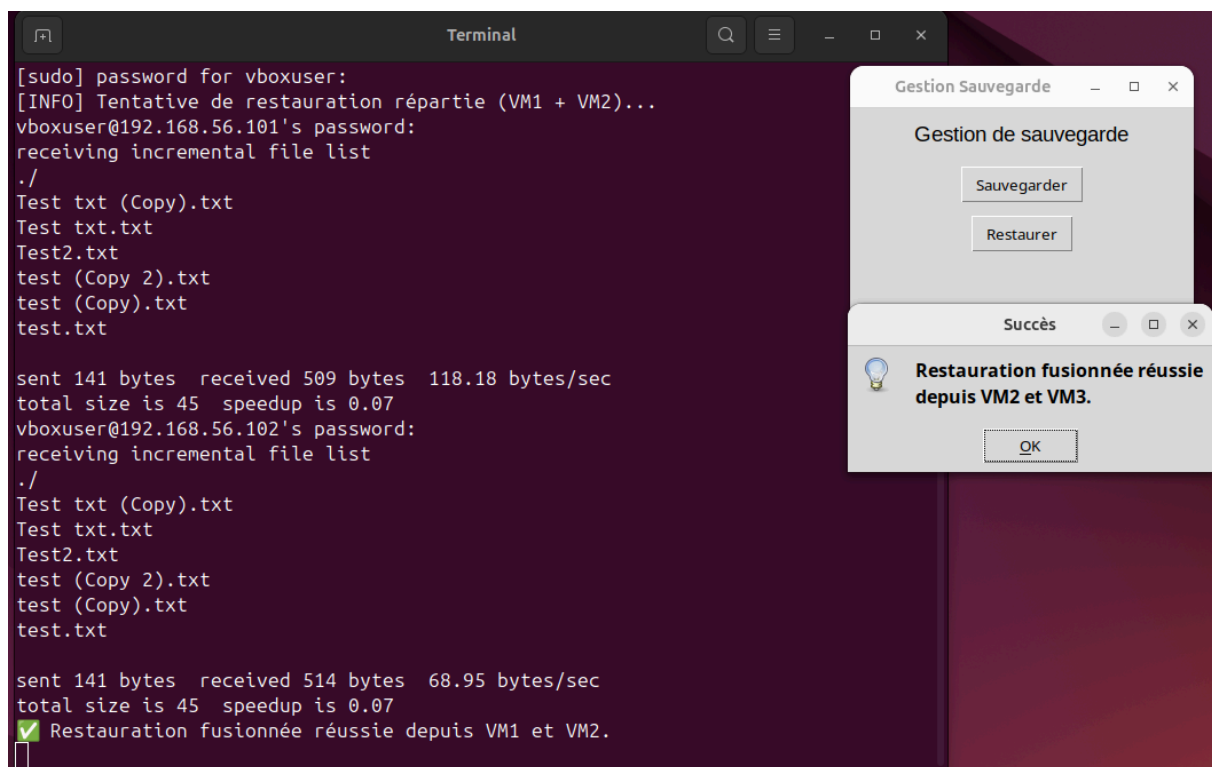


Image 71 : Troisième test : restauration des données du serveur 3, à l'aide du serveur 1 et 2

Conclusion :

Finalement, on a réussi à sauvegarder et restaurer les données sur les 3 machines virtuelles qui représentent ici, les 3 serveurs :

- Le serveur 1 représente le serveur de travail ;
- Le serveur 2 représente le serveur de sauvegarde ;
- Le serveur 3 représente le serveur de restauration ;

Mais on a aussi créé une interface application et web, pour faciliter ce processus. De plus, on peut les faire communiquer entre eux par les clés SSH.

Ainsi, si l'un des serveurs tombe en panne, on peut le restaurer à l'aide des 2 autres serveurs et ne perdre aucune donnée.

De plus, le serveur sur lequel nous travaillons est plus efficace qu'avec l'ajout de toutes les sauvegardes.