Urban Mathys - Axel Macé - Vittorio Gandossi

B1 Informatique

Compte rendu sauvegarde :

Introduction : Dans ce projet, nous allons détailler la mise en place d'un système de sauvegarde automatisé 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.
- **Tkinter et Flask (Python)** : pour développer une interface utilisateur graphique.

Répartition des rôles :

Axel Macé :

Étape 1 : Prérequis et configuration

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

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 : Système de sauvegarde divisée

Sommaire :

Table des matières :

Étape 10 : Sauvegarde sur divers serveurs :

Étape 11 : Système de sauvegarde divisée :

Étape 1 : Prérequis et configuration :

Dans ce projet, nous allons tout d'abord mettre en place notre 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:
```

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.
```

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
```

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/
```

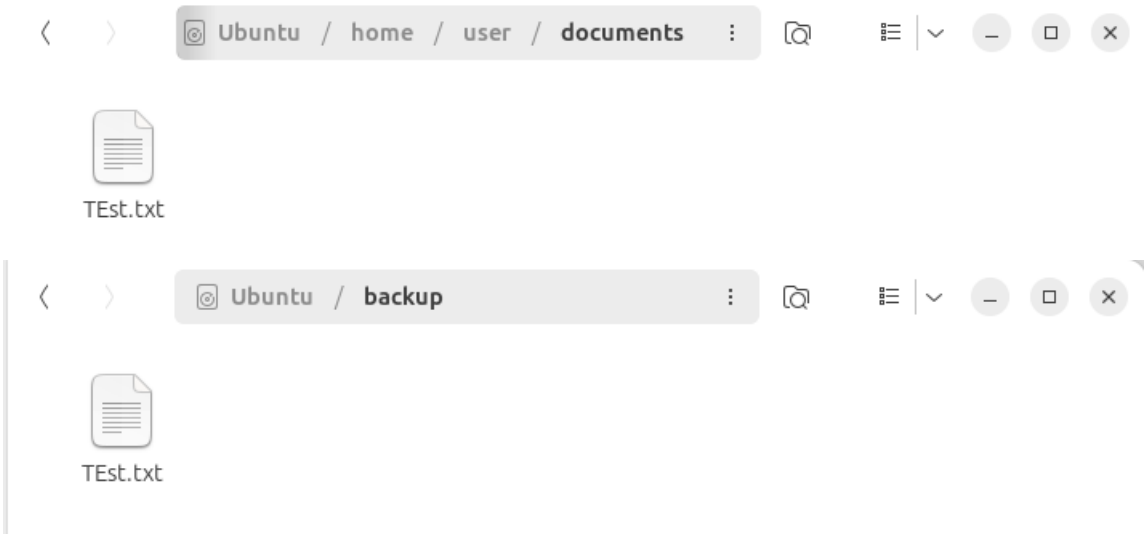
On n'oublie pas de donner les permissions avec la commande « chmod » pour exécuter notre programme.

```
vboxuser@Linuxsave:~$ sudo chmod +x 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
```

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



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

Étape 4 : Tache automatique :

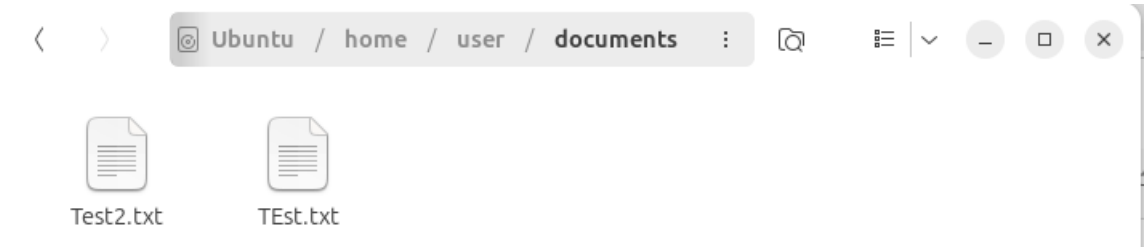
Maintenant on va réaliser une tâche automatique où l'on va sauvegarder automatiquement les données vers un temps précis comme 2 heures du matin. Pour cela on va utiliser la commande « crontab » et choisir nano comme éditeur de script.

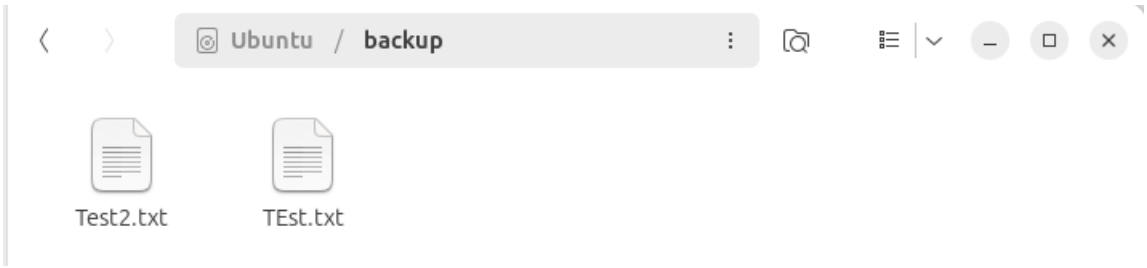
```
vboxuser@Linuxsave:~$ crontab -e
no crontab for vboxuser - using an empty one

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
```

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

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

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 ».





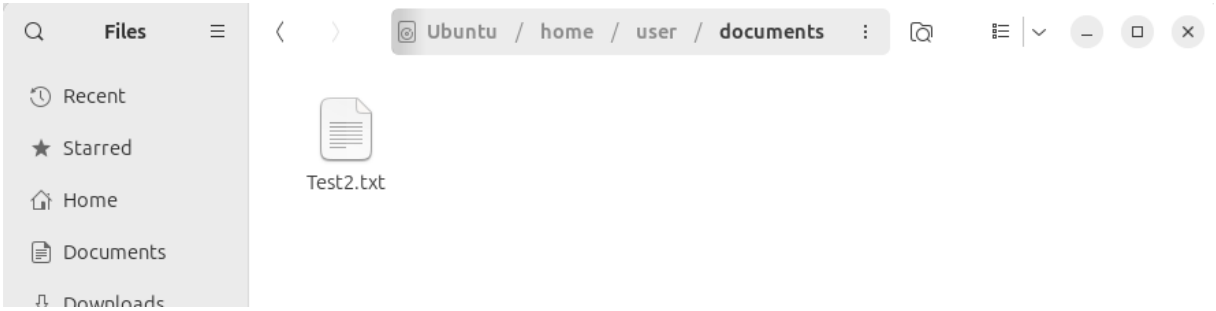
Maintenant, il faudrait faire un moyen de restaurer les données sauvegardées de maniére rapide si on perd les fichiers originaux.

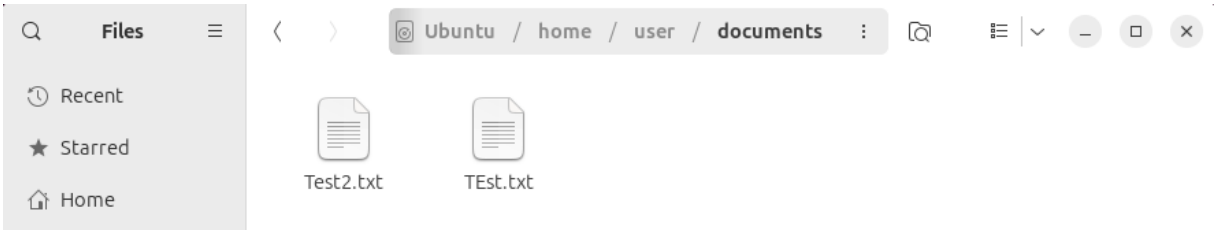
Étape 5 : Restauration des données :

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
```

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.





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
```

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/system
d/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
Created symlink /etc/systemd/system/sshd.service → /usr/lib/systemd/system/ssh.s
```

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
```

On n’hésite pas aussi à ajouter cet utilisateur sur le groupe « sudo ».

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

Et enfin on crée un dossier et on lui donne toutes les permissions à celui-ci.

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

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:TmX4RWyNN5C4OlUkVh42nre4uLBPDDQAap1IzC1UYJo.
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)
```

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
```

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

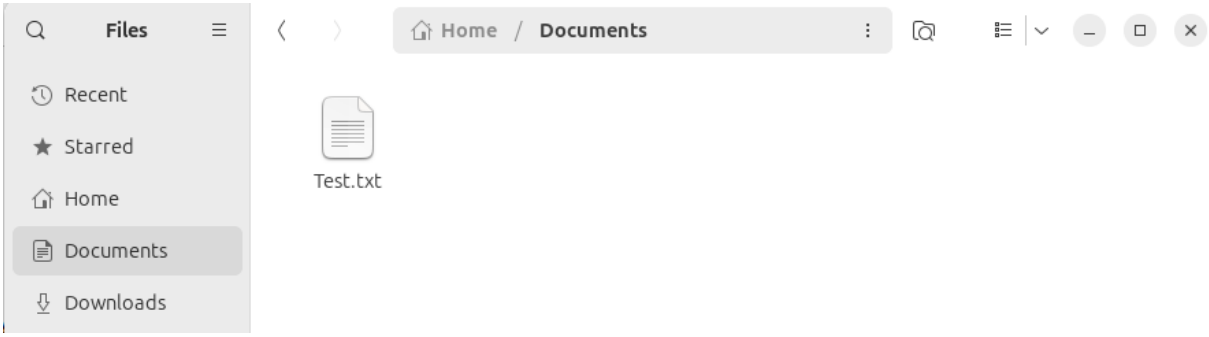
```
#!/bin/bash

# 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."
```

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

```
sudo chmod +x 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.
```

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



On n'hésite pas à sauvegarder 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
```

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

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
```

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.

```
vboxuser@Linuxsave:~$ sudo apt update && sudo apt install python3-flask -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
41 packages can be upgraded. Run 'apt list --upgradable' to see them.
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
```

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

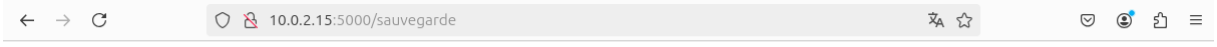
```
vboxuser@Linuxsave:~$ sudo nano interface_web_sauvegarde.py
vboxuser@Linuxsave:~$ chmod +x interface_web_sauvegarde.py
```

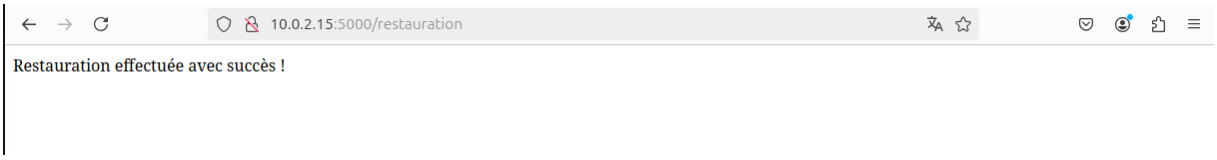
Puis on créer le script suivant en Python.

```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=
        return "Sauvegarde effectuée avec succès !"
    except subprocess.CalledProcessError:
        return "Échec de la sauvegarde."
```

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 -
```





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 adapté 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
```

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

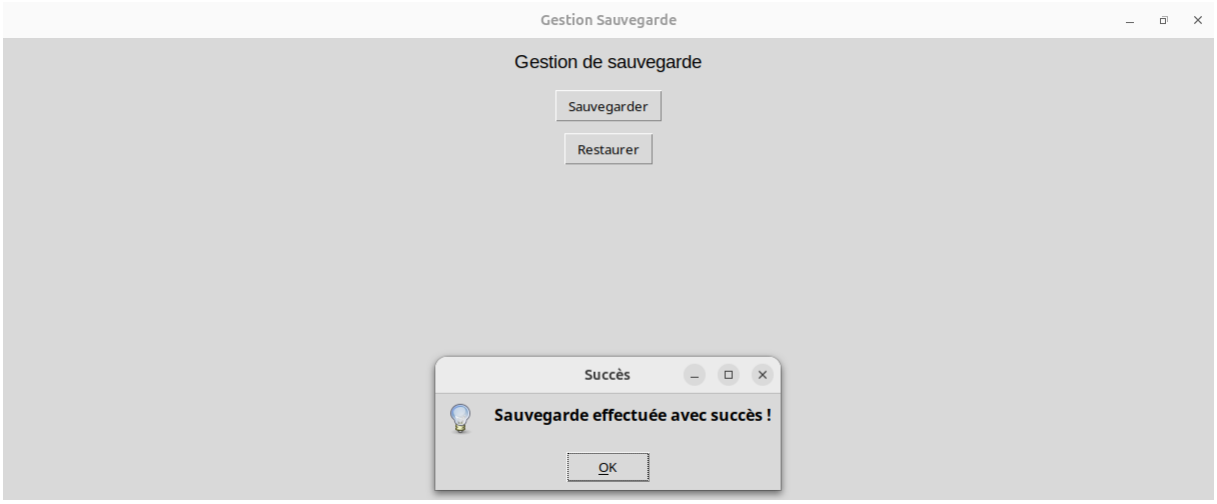
```
vboxuser@Linuxsave:~$ sudo nano interface_sauvegarde_tk.py
vboxuser@Linuxsave:~$ chmod +x interface_sauvegarde_tk.py
```

Puis on va mettre le script pour l'interface :

```python
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)
```

Puis, si on exécute notre programme on devrait avoir l'interface qui apparait.

```
vboxuser@Linuxsave:~$ python3 interface_sauvegarde_tk.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
```



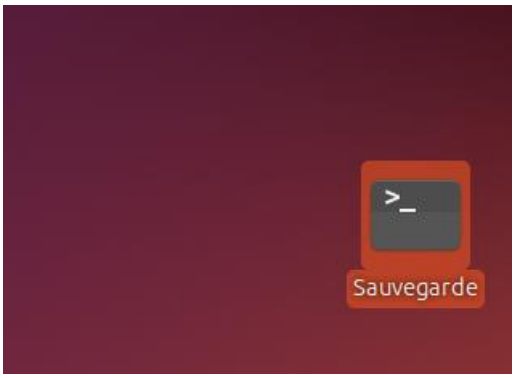
En bonus, tu peux faire un raccourci bureau pour l'interface de sauvegarde avec « nano ».

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

```
[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;
```

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

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



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.servic
e changed on disk. Run 'systemctl daemon-reload' to reload units.
Warning: The unit file, source configuration file or drop-ins of esm-cache.servi
ce 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,
```

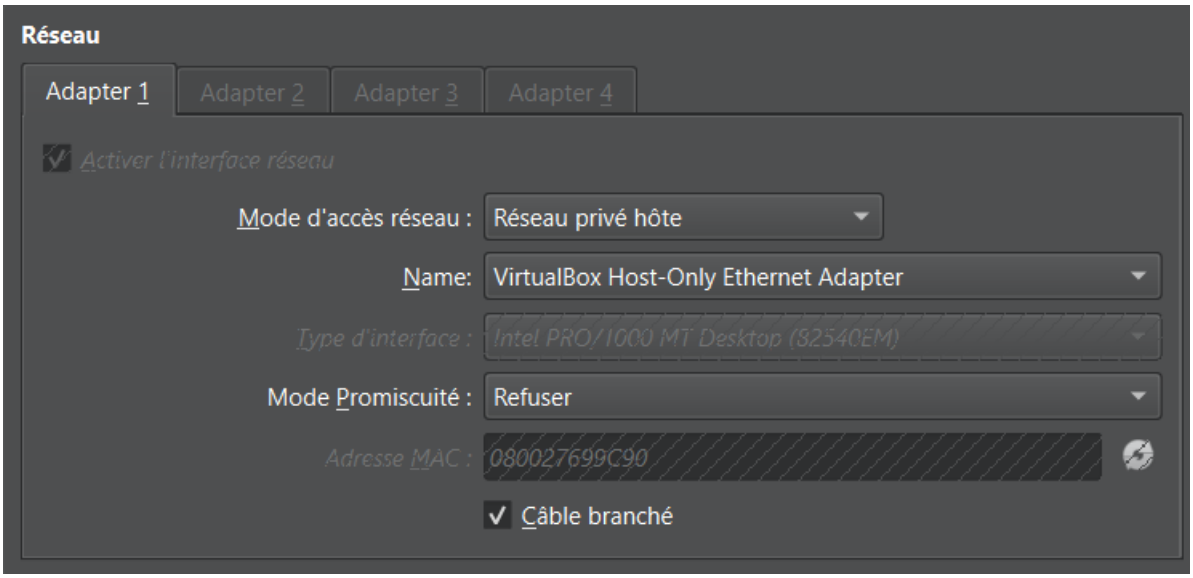
Puis on va créer et copier notre clé sur les deux VMs afin qu'elles puissent communiquer.

```
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/.s
sh/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 prompt
ed 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.
```

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
```

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



Maintenant il va falloir modifier nos programmes pour qu'elles sauvegardent sur les IPs des VMS, 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."
```

```
#!/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
```

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

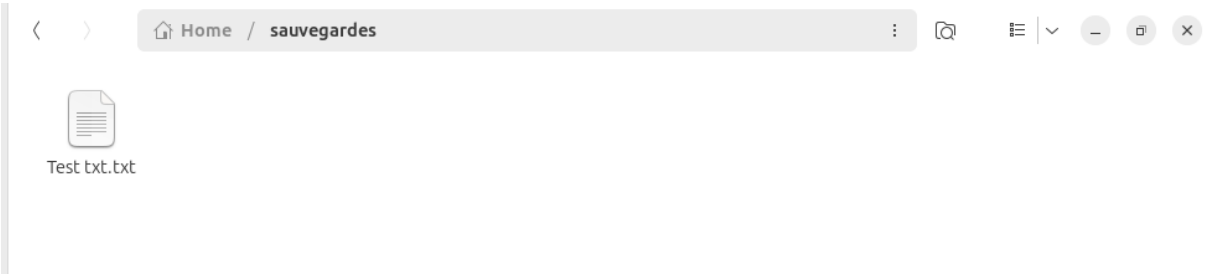
```
 GNU nano 7.2                                  interface_sauvegarde_tk.py
import tkinter as tk
from tkinter import messagebox
import subprocess

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/Documen>
    elif vm == "VM3":
        cmd = ["pkexec", "rsync", "-av", "vboxuser@192.168.56.103:/home/vboxuser/sauvegardes/", "/home/vboxuser/Documen>
    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
```

```
# 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()
```

```
 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>
    '''
```

```
@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/Documen
        elif vm == "192.168.56.103":
            subprocess.run(["sudo", "rsync", "-av", "vboxuser@VM3:/home/vboxuser/sauvegardes/", "/home/vboxuser/Documen
        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)
```

Et enfin, on peut voir que si je fais le test pour sauvegarder ou restaurer, cela fonctionne parfaitement.



Étape 10 : Système de sauvegarde divisée :

Maintenant nous allons 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 avoir de meilleures performances, surtout si on sauvegarde des fichiers lourds.

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."
```

```bash
#!/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."
```

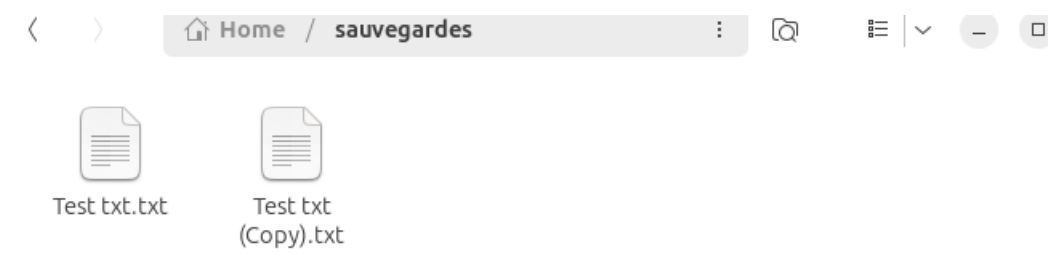
Puis, il faudra aussi modifier les interfaces web et de l'application pour inclure ce nouveau système de sauvegarde.

```
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)
```

```python
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()
```

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.
```



Conclusion :

En conclusion, on a réussi à sauvegarder et restaurer les données sur une machine, 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. Et enfin, on va maintenant, pour plus tard, sécuriser les dossiers pour éviter qu'ils soient observés par d'autres utilisateurs.