UNG
Roland
20231933
L3ASR

Rapport de projet

Thème 1 : sauvegarde avec Borg

Projet administration réseau - ASR

Année universitaire 2025/2026

Table des matières

1. **Sauvegardes avec Borgbackup** 4
   - 1.1. **Ecriture du script** 4
   - 1.2. **Exécution du script** 4
   - 1.3. **Différence lien physique et lien symbolique** 5
     - 1.3.1. **Lien symbolique** 5
     - 1.3.2. **Lien physique** 5
   - 1.4. **Déduplication** 5
   - 1.5. **Compression** 6
   - 1.6. **Chiffrement des sauvegardes** 6
     - 1.6.1. **Méthode `none`** 6
     - 1.6.2. **Méthode `repokey`** 6
     - 1.6.3. **Méthode `keyfile`** 6
     - 1.6.4. **Méthode recommandée** 7
   - 1.7. **Procédures de restauration de données** 7
     - 1.7.1. **Initialisation Borg** 7
     - 1.7.2. **Restauration d'un dossier** 8
     - 1.7.3. **Restauration d'un fichier** 8
     - 1.7.4. **Restauration complète** 8
2. **Test de d'autres outils de sauvegardes** 9
   - 2.1. **Outil Rsync** 9
   - 2.2. **Outil Rsnapshot** 9
     - 2.2.1. **Configuration fichier** 9
     - 2.2.2. **Test de configuration** 10
   - 2.3. **Outil Restic** 10
     - 2.3.1. **Initialisation de l'outil** 10
     - 2.3.2. **Test de sauvegarde** 10
     - 2.3.3. **Liste des sauvegardes** 11
     - 2.3.4. **Test de restauration** 11
   - 2.4. **Outil Duplicati** 11
     - 2.4.1. **Installation du paquet** 11
     - 2.4.2. **Configuration dans l'interface** 11

2.5. Comparaison des outils de sauvegardes 15
2.5.1. Fonctionnalités 15
2.5.2. Restauration à l'identique 15
2.5.3. Performance 16
3. Sécurité des sauvegardes 16
3.1. Problématiques 16
3.1.1. Confidentialité 16
3.1.2. Intégrité 16
3.1.3. Disponibilité et ransomwares 17
3.2. Solutions des outils de sauvegardes 17
3.2.1. Chiffrement 17
3.2.2. Déduplication 17
3.2.3. Immuabilité 17

1. Sauvegardes avec Borgbackup

1.1. Ecriture du script

Script à écrire dans cree-arbo.sh pour chaque utilisateur (cd /home/test1, cd /home/test2, cd /home/test3)

```
#!/bin/bash

USER=$(whoami)
BASE="$HOME/data-$USER"

mkdir -p "$BASE"
cd "$BASE" || exit 1

echo "Fichier de test pour $USER" > test-$USER.txt
echo "Fichier caché pour $USER" > .test-dot-$USER.txt

ln -s test-$USER.txt test-sl-$USER.txt
ln test-$USER.txt test-hl-$USER.txt
```

1.2. Exécution du script

Rendre le script exécutable : chmod +x cree-arbo.sh

Exécuter le script depuis chaque utilisateur :

- su – nameUser
- ./cree-arbo.sh

Vérifier le résultat :

```
test1@debian:~$ ls -la data-test1
total 20
drwxr-xr-x  2 test1 test1 4096 24 janv. 19:08 .
drwx------ 14 test1 test1 4096 24 janv. 19:08 ..
-rw-r--r--  1 test1 test1   26 24 janv. 19:08 .test-dot-test1.txt
-rw-r--r--  2 test1 test1   27 24 janv. 19:08 test-hl-test1.txt
lrwxrwxrwx  1 test1 test1   14 24 janv. 19:08 test-sl-test1.txt -> test-test1.txt
-rw-r--r--  2 test1 test1   27 24 janv. 19:08 test-test1.txt
```

```
test2@debian:~$ ls -la data-test2
total 20
drwxr-xr-x 2 test2 test2 4096 24 janv. 19:54 .
drwx------ 3 test2 test2 4096 24 janv. 19:54 ..
-rw-r--r-- 1 test2 test2   26 24 janv. 19:54 .test-dot-test2.txt
-rw-r--r-- 2 test2 test2   27 24 janv. 19:54 test-hl-test2.txt
lrwxrwxrwx 1 test2 test2   14 24 janv. 19:54 test-sl-test2.txt -> test-test2.txt
-rw-r--r-- 2 test2 test2   27 24 janv. 19:54 test-test2.txt

test3@debian:~$ ls -la data-test3
total 20
drwxr-xr-x 2 test3 test3 4096 24 janv. 19:56 .
drwx------ 3 test3 test3 4096 24 janv. 19:56 ..
-rw-r--r-- 1 test3 test3   26 24 janv. 19:56 .test-dot-test3.txt
-rw-r--r-- 2 test3 test3   27 24 janv. 19:56 test-hl-test3.txt
lrwxrwxrwx 1 test3 test3   14 24 janv. 19:56 test-sl-test3.txt -> test-test3.txt
-rw-r--r-- 2 test3 test3   27 24 janv. 19:56 test-test3.txt
```

1.3. Différence lien physique et lien symbolique

1.3.1. Lien symbolique

Un lien symbolique est un fichier spécial qui contient un chemin vers un autre fichier

Comme on le voit dans les trois captures précédentes, le fichier test-sl-testX.txt contient le chemin vers le fichier test-testX.txt

1.3.2. Lien physique

Lien physique : deux noms différents qui pointent vers le même contenu, ils ont le même inode (numéro interne).

```
test1@debian:~$ ls -li data-test1
total 8
914398 -rw-r--r-- 2 test1 test1 27 24 janv. 19:08 test-hl-test1.txt
914400 lrwxrwxrwx 1 test1 test1 14 24 janv. 19:08 test-sl-test1.txt -> test-test1.txt
914398 -rw-r--r-- 2 test1 test1 27 24 janv. 19:08 test-test1.txt
```

Comme on le voit dans la capture, il y a deux fichiers différents qui ont le même inode 914398.

C'est important pour les sauvegardes car certains outils de sauvegardes utilisent les liens physiques pour économiser de l'espace. Borg détecte que deux fichiers sont identiques

1.4. Déduplication

La déduplication consiste à ne stocker qu'une seule fois les données identiques. Borg découpe le fichier en bloc et ne stocke qu'une seule fois le bloc s'il n'existe pas encore dans le dépôt. Son intérêt est de réduire l'espace du disque et de faire des sauvegardes incrémentales automatiques.

Elle s'effectue avant la compression et avant le chiffrement. Elle s'applique sur les blocs de données, pas sur les fichiers entiers

1.5. Compression

La compression s'applique après la déduplication, sur les blocs uniques. Borg compresse la taille des blocs stockés pour réduire l'espace du disque. Comme les données sont compressés avant l'envoi, cela réduit le volume transféré sur le réseau.

Mode auto : Borg choisit automatiquement la meilleure méthode selon le type de données. Cela permet de ne pas perdre du temps à compresser ce qui ne peut pas l'être, gagner de l'espace quand c'est utile et rester rapide.

1.6. Chiffrement des sauvegardes

1.6.1. Méthode `none`

Les données sont stockées en clair dans le dépôt, sur le serveur de sauvegarde. N'importe qui ayant accès au dépôt pourra lire les fichiers. Tout est visible, il n'y a donc pas de confidentialité et n'est pas adapté s'il y a des données importantes.

1.6.2. Méthode `repokey`

Les données sont chiffrées du côté client, et la clé de chiffrement est stockée dans le dépôt Borg lui-même sur le serveur, protégée par une passphrase. Le seul inconvénient est si quelqu'un connait ou force le passphrase, il pourra déchiffrer toutes les données.

1.6.3. Méthode `keyfile`

Les données sont chiffrées du côté client, et la clé de chiffrement est stockée dans un fichier local sur la machine cliente, protégée par une passphrase. L'inconvénient est que si le fichier est perdu, alors la clé aussi, donc les sauvegardes ne seront plus jamais déchiffrées

1.6.4. Méthode recommandée

La meilleure méthode est `repokey` car tout d'abord, les données sont chiffrées, donc elles ne sont pas visibles par tous le monde. La clé de chiffrement est stockée dans un dépôt, ce qui évite de gérer un fichier séparé. De plus cette clé est protégée par une passphrase. Cette méthode offre une bonne sécurité et est simple à utiliser.

1.7. Procédures de restauration de données

1.7.1. Initialisation Borg

Sur serveur2, création d'un dossier dédié pour la sauvegarde :

sudo mkdir -p /srv/borg/station1

sudo chown debian:debian /srv/borg/station1

Depuis station 1, initialisation borg :

```
root@debian:~# borg init --encryption=repokey-blake2 debian@192.168.10.4:/srv/borg/station1
debian@192.168.10.4's password:
Enter new passphrase:
Enter same passphrase again:
Do you want your passphrase to be displayed for verification? [yN]: y
Your passphrase (between double-quotes): "borg"
Make sure the passphrase displayed above is exactly what you wanted.

By default repositories initialized with this version will produce security
errors if written to with an older version (up to and including Borg 1.0.8).

If you want to use these older versions, you can disable the check by running:
borg upgrade --disable-tam ssh://debian@192.168.10.4/srv/borg/station1

See https://borgbackup.readthedocs.io/en/stable/changes.html#pre-1-0-9-manifest-spoofing-vulnerability
 for details about the security implications.

IMPORTANT: you will need both KEY AND PASSPHRASE to access this repo!
If you used a repokey mode, the key is stored in the repo, but you should back it up separately.
Use "borg key export" to export the key, optionally in printable format.
Write down the passphrase. Store both at safe place(s).
```

Création d'une archive et liste des archives :

```
root@debian:~# borg create debian@192.168.10.4:/srv/borg/station1::archive /home/test1/data-test1
debian@192.168.10.4's password:
Enter passphrase for key ssh://debian@192.168.10.4/srv/borg/station1:
root@debian:~# borg list debian@192.168.10.4:/srv/borg/station1
debian@192.168.10.4's password:
Enter passphrase for key ssh://debian@192.168.10.4/srv/borg/station1:
test-backup                          Sun, 2026-01-25 19:46:09 [13c1be9524dfb07eb1996525d224ab7183c9c44b7f45a04c7c9b1b7ba4282b29]
archive                              Sun, 2026-01-25 22:10:13 [4249c2d74ddaf81bbb1331ddb4ae38a43464d4b5c73dc9ea28f6fa1baa57594d]
```

1.7.2. Restauration d'un dossier

Exemple pour le dossier data-test1

```
root@debian:~# borg extract debian@192.168.10.4:/srv/borg/station1::archive /home/test1/data-test1
debian@192.168.10.4's password:
Enter passphrase for key ssh://debian@192.168.10.4/srv/borg/station1:
```

1.7.3. Restauration d'un fichier

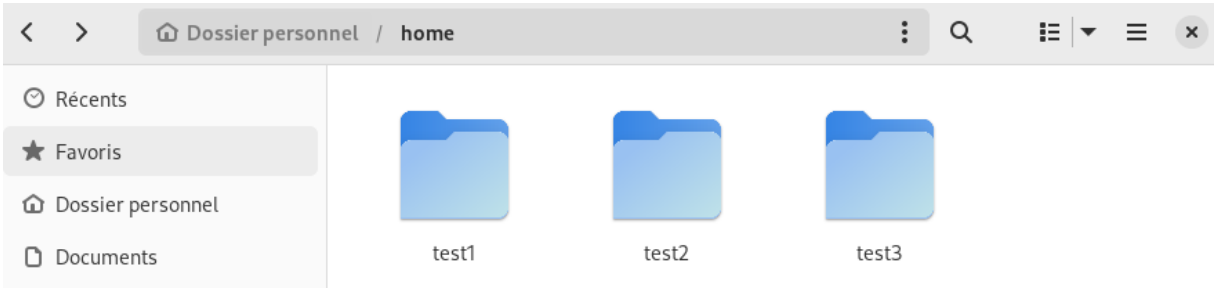
Exemple pour le fichier test-hl-test2.txt

```
root@debian:~# borg create debian@192.168.10.4:/srv/borg/station1::arc_test2 /home/test2/data-test2
debian@192.168.10.4's password:
Enter passphrase for key ssh://debian@192.168.10.4/srv/borg/station1:
root@debian:~# borg extract debian@192.168.10.4:/srv/borg/station1::arc_test2 /home/test2/data-test2/test-hl-test2.txt
debian@192.168.10.4's password:
Enter passphrase for key ssh://debian@192.168.10.4/srv/borg/station1:
```

1.7.4. Restauration complète

Exemple pour l'archive qui contient les dossiers test2 et test3

```
root@debian:~# borg create debian@192.168.10.4:/srv/borg/station1::gros_archive \
> /home/test2/data-test2 \
> /home/test3/data-test3
debian@192.168.10.4's password:
Enter passphrase for key ssh://debian@192.168.10.4/srv/borg/station1:
root@debian:~# borg extract debian@192.168.10.4:/srv/borg/station1::gros_archive
debian@192.168.10.4's password:
Enter passphrase for key ssh://debian@192.168.10.4/srv/borg/station1:
```



2. Test de d'autres outils de sauvegardes

On va sauvegarder : /home/test1/data-test1

Et stocker les sauvegardes dans : /sauvegardes

On crée d'abord le dossier (dans la machine cliente station1)

sudo mkdir -p /sauvegardes

sudo chown $USER:$USER /sauvegardes

2.1. Outil Rsync

Après l'avoir installé :

```
root@debian:~# rsync -av /home/test1/data-test1/ /sauvegardes/rsync-test1/
sending incremental file list
created directory /sauvegardes/rsync-test1
./
.test-dot-test1.txt
test-hl-test1.txt
test-sl-test1.txt -> test-test1.txt
test-test1.txt

sent 399 bytes  received 126 bytes  1.050,00 bytes/sec
total size is 94  speedup is 0,18
```

2.2. Outil Rsnapshot

2.2.1. Configuration fichier

Après avoir installé, on édite la configuration du fichier : sudo nano /etc/rsnapshot.conf

snapshot_root /sauvegardes/rsnapshot/

retain daily 3

cmd_rsync /usr/bin/rsync

backup /home/test1/data-test1/ test1/

On remplace ou met en commentaire toutes les autres lignes avec snapshot_root, retain et backup. On utilise que des tabulations, il n'y a aucun espace

2.2.2. Test de configuration

```
root@debian:~# sudo rsnapshot configtest
Syntax OK
```

On lance une sauvegarde et on vérifie :

```
root@debian:~# sudo rsnapshot daily
root@debian:~# ls -l /sauvegardes/rsnapshot
total 12
drwxr-xr-x 3 root root 4096 27 janv. 11:38 daily.0
drwxr-xr-x 3 root root 4096 27 janv. 11:38 daily.1
drwxr-xr-x 3 root root 4096 27 janv. 11:38 daily.2
```

2.3. Outil Restic

2.3.1. Initialisation de l'outil

Après l'avoir installé, on l'initialise :

mkdir -p /sauvegardes/restic

export RESTIC_PASSWORD=motdepassefort

restic init --repo /sauvegardes/restic

2.3.2. Test de sauvegarde

```
root@debian:~# restic -r /sauvegardes/restic backup /home/test1/data-test1
repository 27cbeefe opened (repository version 2) successfully, password is correct
created new cache in /root/.cache/restic
no parent snapshot found, will read all files

Files:           3 new,     0 changed,     0 unmodified
Dirs:            3 new,     0 changed,     0 unmodified
Added to the repository: 2.592 KiB (1.495 KiB stored)

processed 3 files, 80 B in 0:00
snapshot 75bfe324 saved
```

La sauvegarde doit être relancée à chaque modification du fichier

2.3.3. Liste des sauvegardes

```
root@debian:~# restic -r /sauvegardes/restic/ snapshots
repository 27cbeefe opened (repository version 2) successfully, password is correct
ID        Time                 Host        Tags        Paths
----------------------------------------------------------------------------
75bfe324  2026-01-27 10:59:34  debian                  /home/test1/data-test1
----------------------------------------------------------------------------
1 snapshots
```

2.3.4. Test de restauration

```
root@debian:~# restic -r /sauvegardes/restic restore latest --target /tmp/restic-restore
repository 27cbeefe opened (repository version 2) successfully, password is correct
restoring <Snapshot 75bfe324 of [/home/test1/data-test1] at 2026-01-27 10:59:34.112208902
+0100 CET by root@debian> to /tmp/restic-restore
```

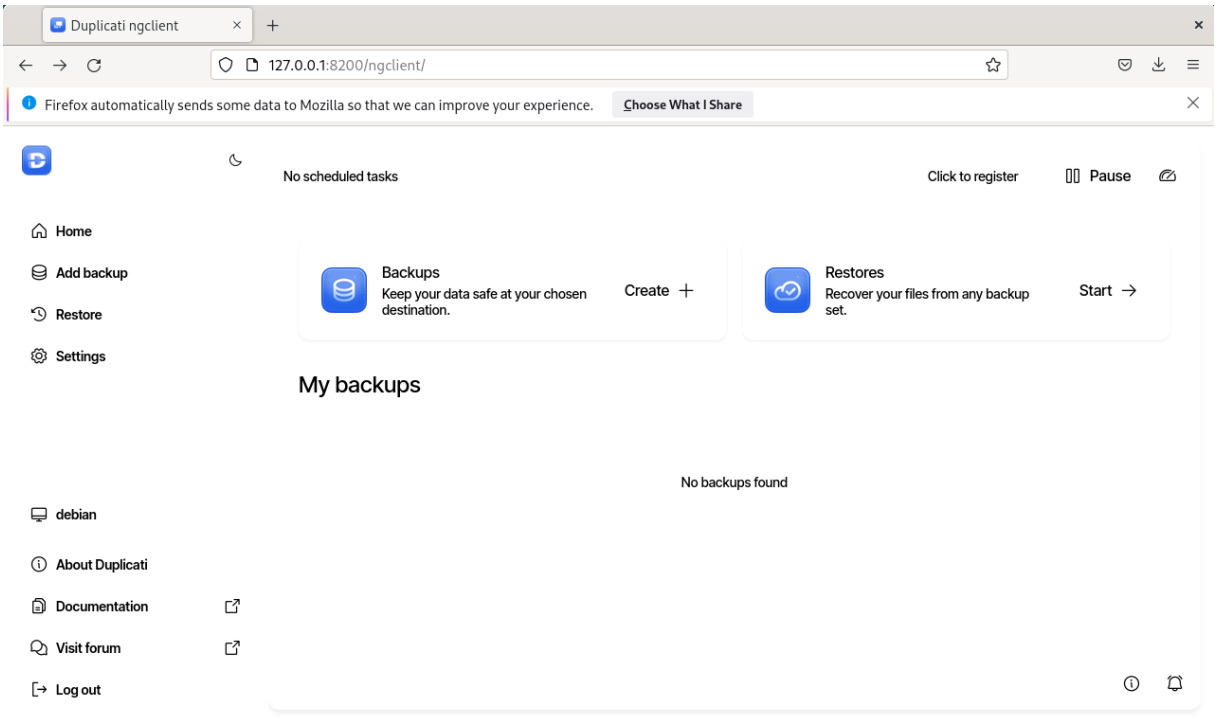
2.4. Outil Duplicati

2.4.1. Installation du paquet

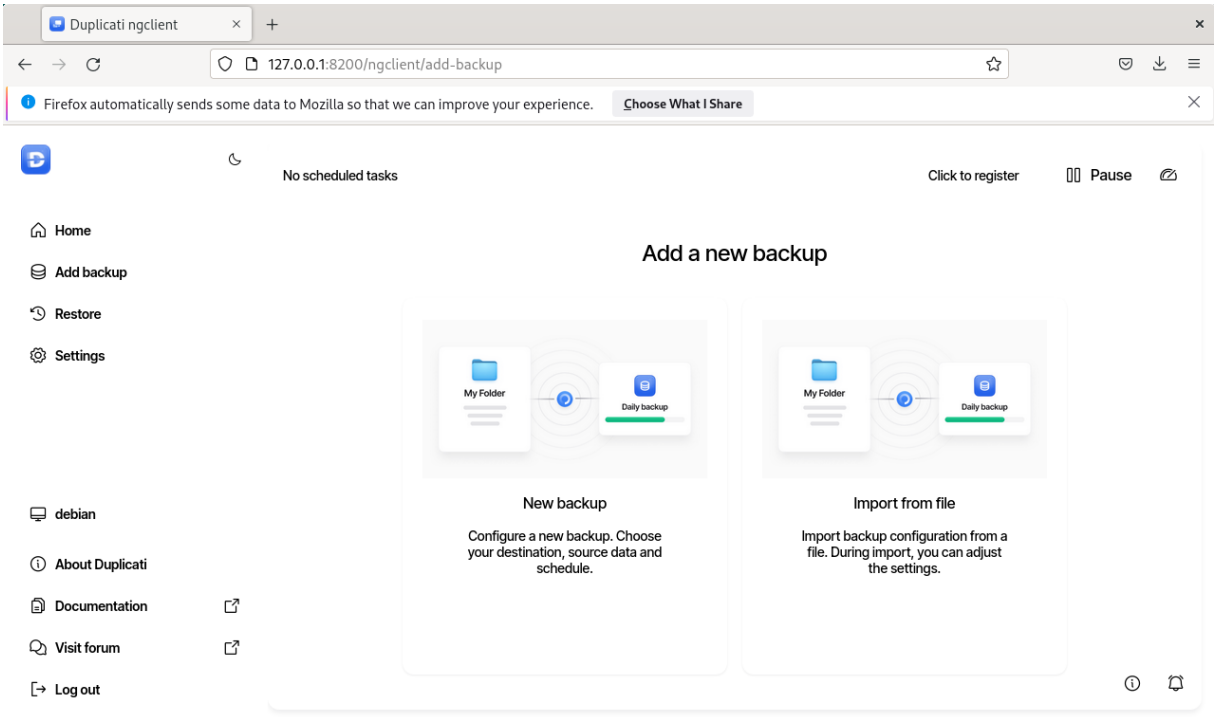
Télécharger le paquet depuis : https://www.duplicati.com/download

Puis installer : sudo dpkg -i nomFichier.deb

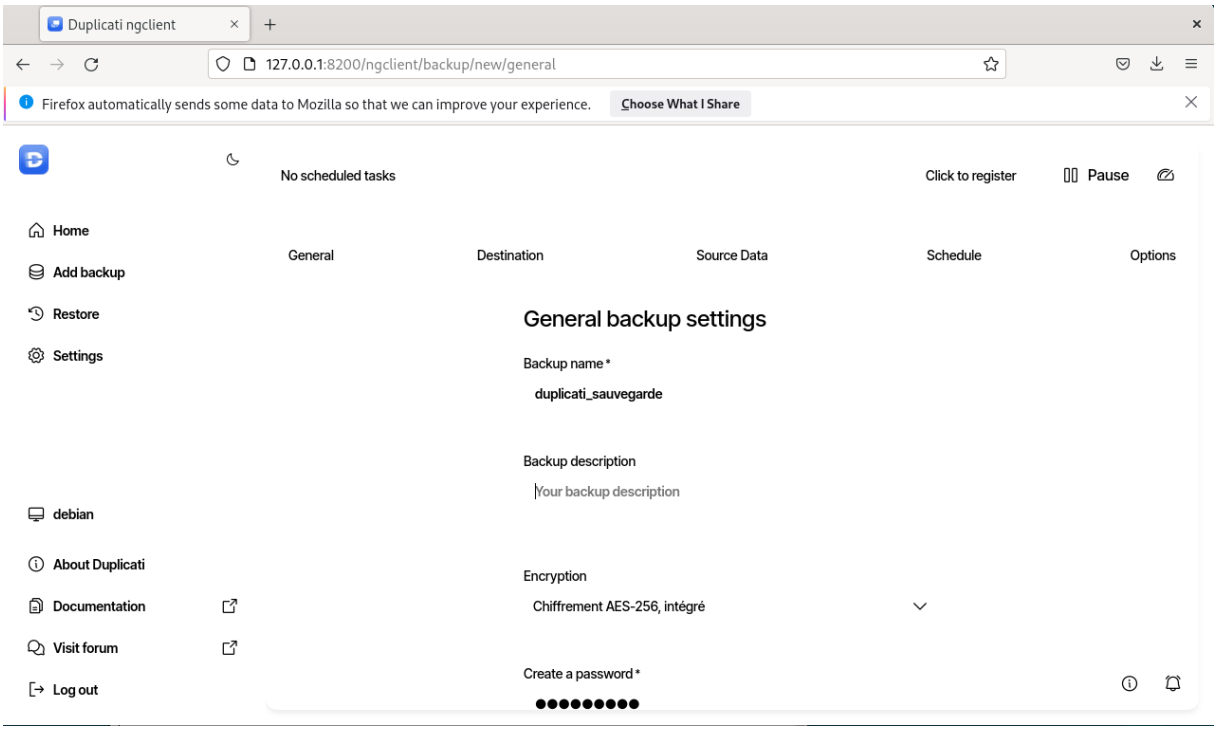
2.4.2. Configuration dans l'interface



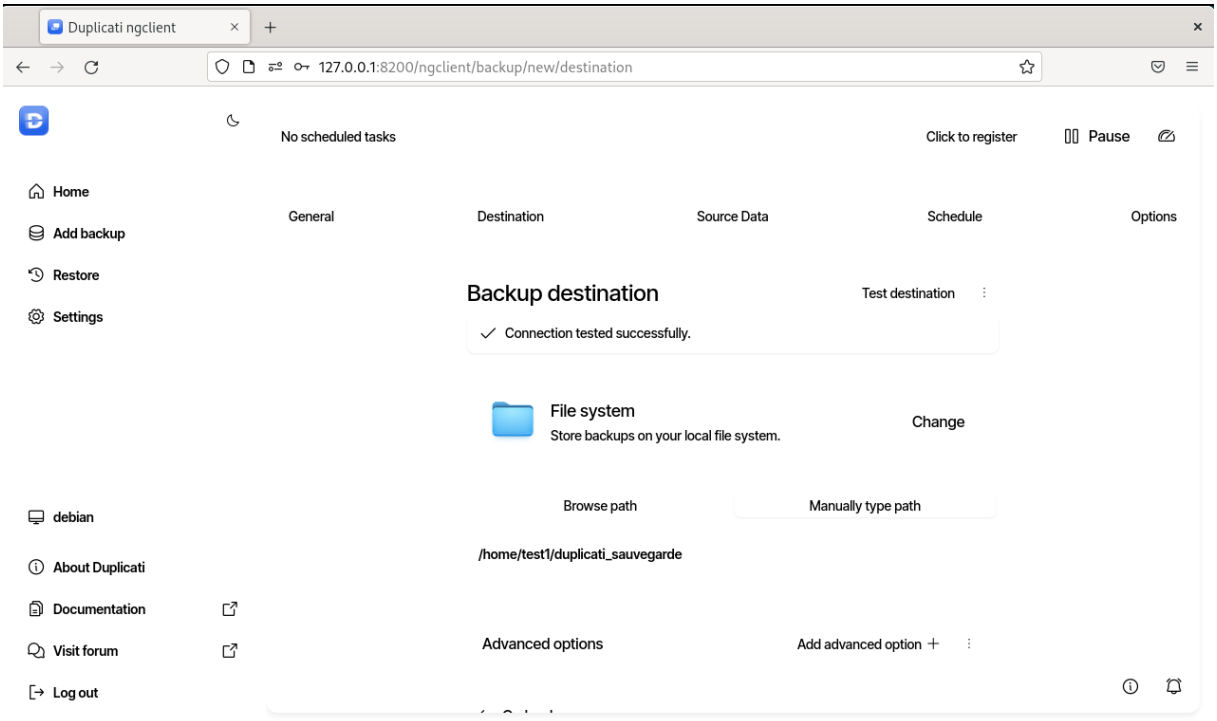
Cliquer Add backup



Cliquer New backup



Rentrer le nom de la sauvegarde, le chiffrement et créer le mot de passe (duplicati)

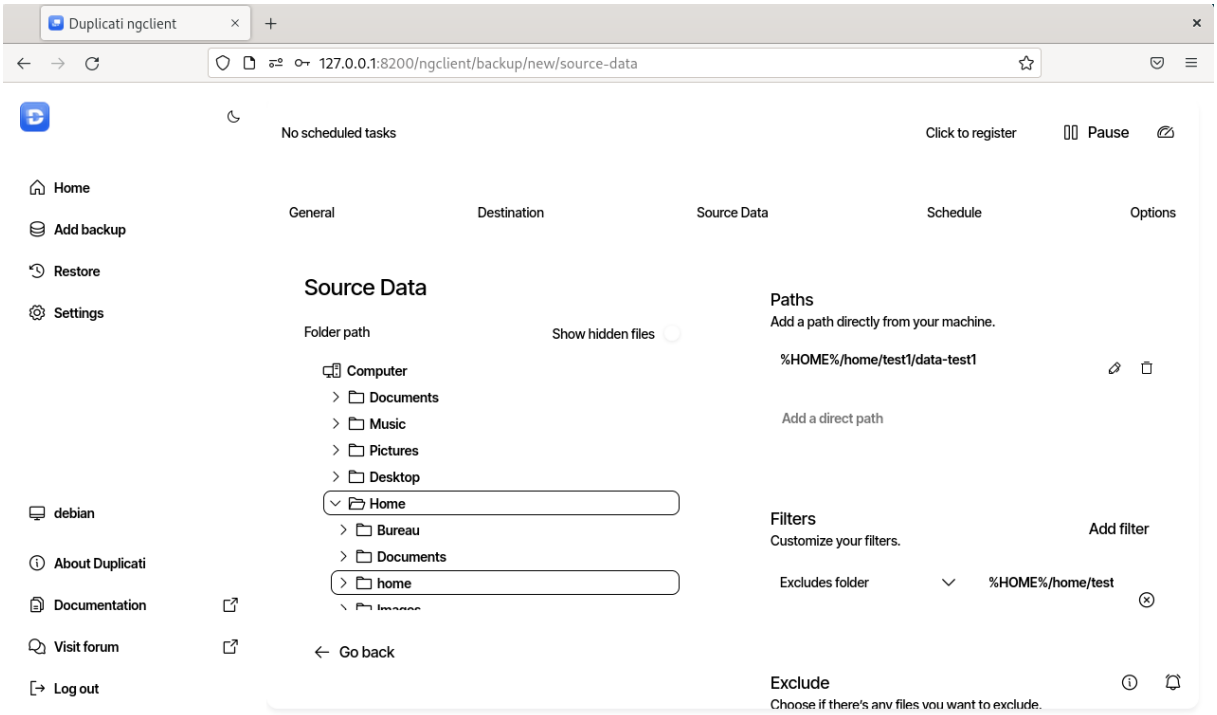


Rentrer l'hôte du serveur (où on veut sauvegarder) : 192.168.10.4

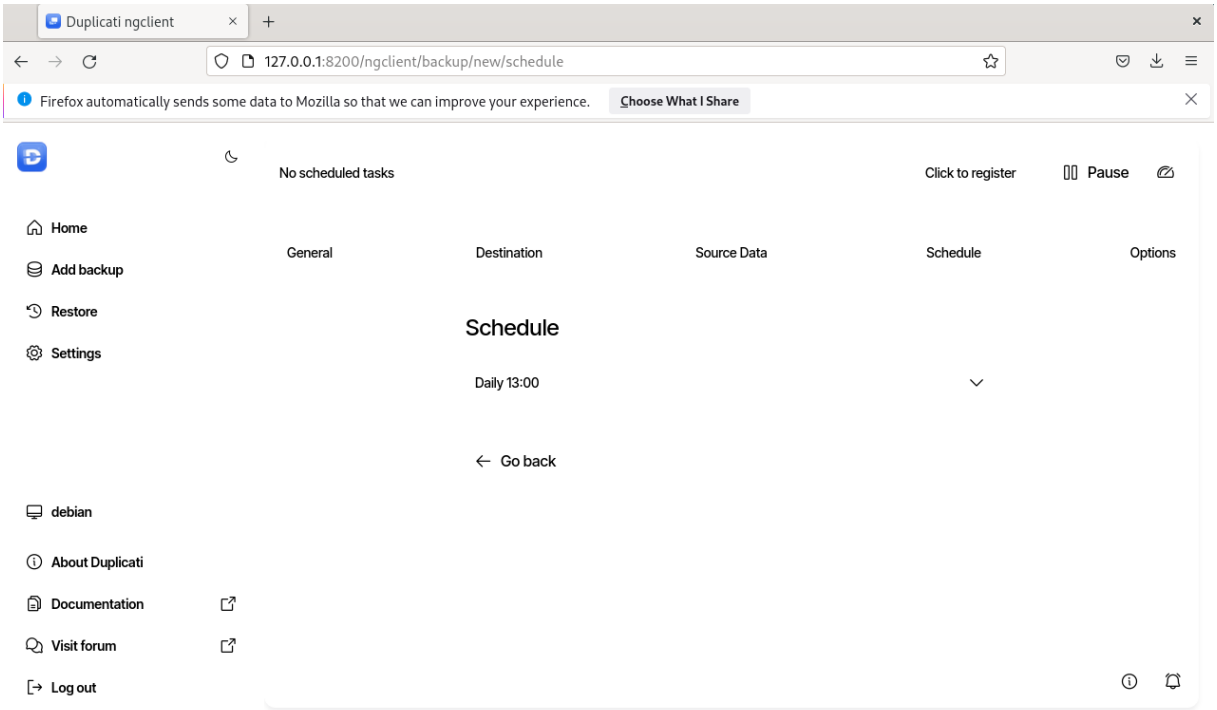
Le port 22

Le nom d'utilisateur test1

Le dossier de stockage /home/test1/duplicati_sauvegarde



On configure le dossier qu'on veut sauvegarder



Sauvegarde automatique quotidiennement à 13h

Duplicati chiffre les fichiers avec AES-256 (celui qu'on a choisi). Il demande donc le mot de passe pour accéder aux données de sauvegardes

2.5. Comparaison des outils de sauvegardes

2.5.1. Fonctionnalités

Les outils de sauvegarde possèdent des fonctionnalités différentes, notamment dans la gestion de la déduplication et du chiffrement. Rsync et Rsnapshot sont limités à la copie de fichiers alors que les nouveaux outils modernes comme Restic, Borg et Duplicati transforment les données en blocs pour ne stocker que les modifications uniques, ce qui permet de réduire et d'optimiser l'espace disque.

Au niveau de la sécurité, Borg, Restic et Duplicati permettent le chiffrement (souvent AES-256) pour protéger les données avant même qu'elles soient envoyées.

Enfin, certains outils sont plus flexibles que d'autres pour la destination : Duplicati et Restic fonctionnent très bien avec les services Cloud (comme Google Drive), alors que Borg et Rsnapshot sont surtout utilisés pour des sauvegardes locales ou via SSH

2.5.2. Restauration à l'identique

Rsync et Rsnapshot sont très bons pour ça, car ils conservent l'arborescence originale sans la modifier, ce qui permet de vérifier visuellement. BorgBackup est plus particulièrement connu pour sa capacité à comparer les fichiers octet par octet avec l'original, ce qui permet de vérifier que tout correspond parfaitement. Restic permet aussi de bien restaurer, mais ça peut être plus compliqué quand il y a beaucoup de fichiers. Duplicati est surtout adapté à la restauration de données personnelles (documents, photos, etc.). Il peut restaurer correctement, mais il n'est pas idéal pour restaurer un système complet avec toutes ses données

2.5.3. Performance

La performance d'un outil de sauvegarde dépend de sa vitesse de transfert de l'efficacité du stockage. BorgBackup est souvent le plus rapide en local, car il permet une bonne déduplication avec une compression. Restic est performant surtout pour les sauvegardes vers le Cloud, car il peut envoyer plusieurs blocs en parallèle. Duplicati est généralement plus lent car il repose sur le framework .NET, ce qui peut rallonger le temps de traitement lorsqu'il y a beaucoup de données. Rsync reste rapide pour transférer des fichiers, mais il devient moins efficace si les fichiers sont déplacés ou renommés, car il ne fait pas de déduplication comme les outils modernes.

3. Sécurité des sauvegardes

3.1. Problématiques

3.1.1. Confidentialité

Les sauvegardes contiennent des données sensibles comme des mots de passe, des documents personnels, une base de données ext... Si les données ne sont pas chiffrées, n'importe qui ayant accès aux sauvegardes peut lire l'intégralité des informations

3.1.2. Intégrité

Une sauvegarde doit être exacte et non altérée. En effet, les données peuvent être modifiées sans qu'on le sache ou corrompues avec le temps ce qui peut rendre les fichiers irrécupérables. Il est important que les données restaurées soient identiques à l'original

3.1.3. Disponibilité et ransomwares

Une sauvegarde doit être restaurable même en cas de panne du serveur principal, de perte de réseau, ou de suppression, qui peut arriver à cause des ransomwares. Ils ciblent aussi les sauvegardes et peuvent les chiffrer, les supprimer.

3.2. Solutions des outils de sauvegardes

3.2.1. Chiffrement

Les outils permettent le chiffrement du côté client. Les données sont chiffrées avant qu'on quitte notre machine. Donc si le stockage Cloud est piraté, les fichiers restent illisibles, cela permet de répondre à la problématique de confidentialité.

3.2.2. Déduplication

Les fichiers sont découpés en blocs. Si un ransomware essaie de modifier un fichier, cela ne touchera pas les blocs précédents déjà sauvegardés.

De plus, chaque bloc de données possède une empreinte numérique unique (le hachage) qui permet de vérifier qu'il n'a pas été modifié. Cela permet de répondre à la problématique d'intégrité

3.2.3. Immuabilité

Pour se protéger contre les ransomwares, on peut les empêcher de modifier la sauvegarde. Certains outils utilisent des fonctions de verrouillage (Object Lock) sur le stockage ce qui empêchent la suppression des fichiers pendant une période qu'on aura défini, même si on accède en tant qu'administrateur.