

UE SEC-105

Travaux pratiques

Mise en œuvre d'un outil ou
d'un concept de sécurité

**Mise en place d'une j-1 Passbolt basée sur la
sauvegarde de la veille.**

Suivi du document

Date	Description	Auteur
18/12/2022	Création du document	Anthony GUILLERM
21/01/2024	Création de l'exemple	Anthony GUILLERM

Autorisation de partage

☒ J'autorise le partage du présent document aux autres élèves « Sec-105 » (la présente page sera supprimée pour garantir l'anonymisation du document)

Composition de :

Code premier auditeur CNAM : Enseignant

Description de l'objectif de sécurité visé

Je dispose actuellement d'un Serveur Privé Virtuel (VPS) hébergeant une instance de Passbolt, une solution de gestion de mots de passe, déployée via Docker. L'objectif de ce projet est triple.

Premièrement, l'objectif est d'implémenter une stratégie de sauvegarde basée sur rdiff-backup. Cette approche consistera à créer des sauvegardes différentielles, stockant uniquement les modifications réalisées depuis la dernière sauvegarde complète.

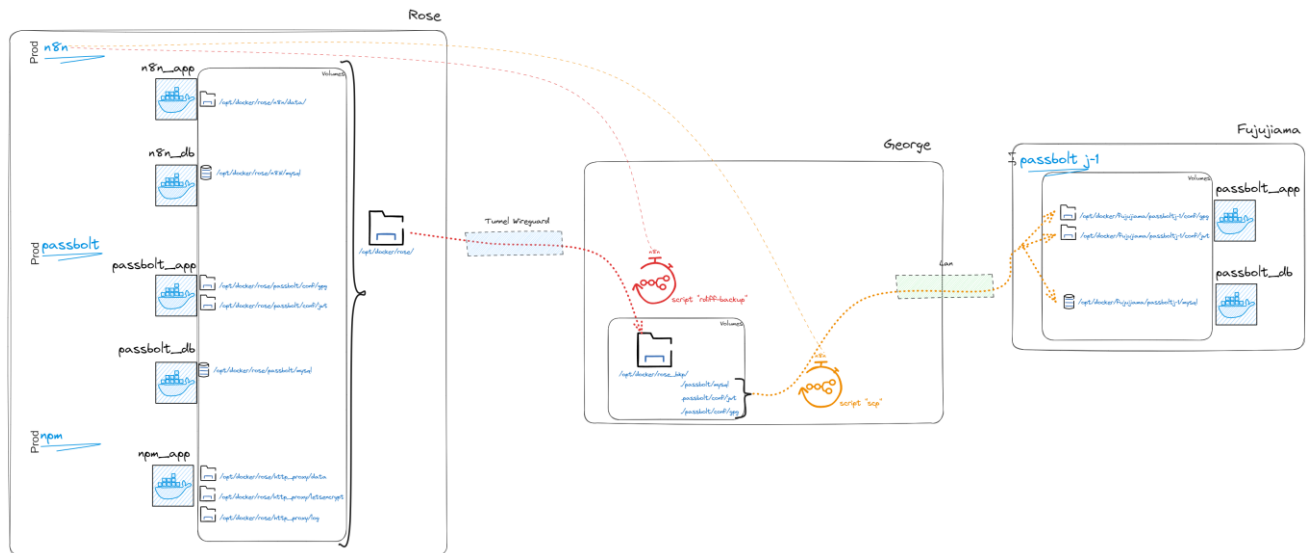
Deuxièmement, il s'agit de mettre en place un VPN Wireguard entre le VPS et le serveur de sauvegarde. Cela ajoutera une couche de sécurité supplémentaire, en s'assurant que les données transmises entre le VPS et le serveur de sauvegarde local soient sécurisées contre les interceptions ou les accès non autorisés. Cela permettra également d'éviter d'exposer directement le port SSH de mon serveur de sauvegarde sur internet.

Enfin, dans un troisième temps, je prévois de mettre en œuvre une instance locale de Passbolt, mise à jour quotidiennement. Ceci dans le but de m'assurer un accès immédiat à mes mots de passe, même en cas de défaillance de l'instance principale.

Schéma de principe

Le schéma ci-après reprend les flux de notre projet :

- La copie des volumes du serveur Rose vers le serveur George à travers un tunnel Wireguard et instancié par une instance n8n
- La copie de la dernière sauvegarde du conteneur Passbolt présente sur le serveur George vers le serveur Fujiyama présent sur le même LAN que George



Implémentation

Mise en œuvre du tunnel VPN

Dans le cadre de mes sauvegardes à destination de mon serveur George, je souhaite pouvoir attaquer n'importe quel port (ssh, SQL ou autre sans avoir à les exposer directement). La mise en place d'un tunnel VPN chiffré est donc requise.

Hésitant entre Open-VPN et Wireguard, j'ai retenu cette dernière, car :

- Wireguard est plus léger que Open-VPN
- Wireguard est plus discret que Open-VPN (le service ne répond pas si l'attaquant ne présente pas de clef)
- Il existe déjà des [configurations pour Crowdsec](#) (aspect qui n'est pas traité dans ce document, mais qui sera déployé ultérieurement afin de bénéficier d'une protection en temps réel contre les adresses IP agressives)

Prérequis

Sur chaque hébergeur, je NATte le port 51830 vers l'hôte du VPN Wireguard

Pour George :

Active	Redirection	IP source	Destination
Active	Protocole: udp WAN : 48213 LAN: 48213 Commentaire: VPN		george

Pour Rose :

Action ▾	Adresses IP autorisées ▾	Protocole ▾	Port(s) ▾	Description ▾
Autoriser		UDP	55619	wireguard vpn

Installation initiale

Sur George

Dans un premier temps, on installe l'application. Noter que les credentials ci-après ne sont pas ceux que j'ai mis en production.

```
sudo apt install wireguard
```

On génère la clef privée du client

```
sudo wg genkey >> /etc/wireguard/private
sudo cat /etc/wireguard/private
kHpqh49F2wah6DYAvpYVSf/NUsoBaH2LNUkIOXNM0EU=
sudo chmod 600 /etc/wireguard/private
```

On génère la clef publique du client

```
sudo cat /etc/wireguard/private.key | wg pubkey  
5e0qnmna8GMSuZeEKa1BjCesJlRKJHWfO+WAnRXYL2w=
```

Sur Rose

Dans un premier temps, on installe l'application

```
apt install wireguard
```

On génère la clef privée du serveur

```
wg genkey >> /etc/wireguard/private  
cat /etc/wireguard/private  
WNsG3gERJbAZJy/NtFuyQp3DHa/1/UBoiGTeJ8memG4=  
chmod 600 /etc/wireguard/private
```

On génère la clef publique du serveur

```
sudo cat /etc/wireguard/private.key | wg pubkey  
x9UWSjr01+fQDIwPKmrUP2kbhujx061Ddole9RVVsh8=
```

Création des interfaces

Ensuite, il convient de paramétrer les interfaces

Sur Rose

On édite le fichier wg0.conf

```
vi /etc/wireguard/wg0.conf
```

tel que:

```
[Interface]  
ListenPort = 51830  
PrivateKey = WNsG3gERJbAZJy/NtFuyQp3DHa/1/UBoiGTeJ8memG4=  
Address = 10.254.0.1/32  
  
[Peer]  
PublicKey = 5e0qnmna8GMSuZeEKa1BjCesJlRKJHWfO+WAnRXYL2w=  
AllowedIPs = 10.254.0.2/32  
Endpoint = client.ddns.net:51830
```

Sur George

On édite le fichier wg0.conf

```
sudo vi /etc/wireguard/wg0.conf
```

tel que:

```
[Interface]
ListenPort = 51830
PrivateKey = kHpqh49F2wah6DYAvpYVSf/NUsoBaH2LNUkIOXNM0EU=
Address = 10.254.0.2/32

[Peer]
PublicKey = x9UWSjr01+fQDIwPKmrUP2kbhujx061DdolE9RVVsh8=
AllowedIPs = 10.254.0.1/32
Endpoint = server.ddns.net:51830
```

Monter les interfaces

Pour monter les interfaces, sur chaque machine, on lance la commande

```
wg-quick up /etc/wireguard/wg0.conf
ip link add wg0 type wireguard
wg setconf wg0 /dev/fd/63
ip -4 address add 10.254.0.2/32 dev wg0
ip link set mtu 1420 up dev wg0
ip -4 route add 10.254.0.1/32 dev wg0
```

On peut vérifier notre configuration avec les commandes ip

```
ip a show wg0
8: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/none
    inet 10.254.0.2/32 scope global wg0
        valid_lft forever preferred_lft forever
```

Ensuite on fait le nécessaire pour que le VPN monte automatiquement en cas de redémarrage du système

```
systemctl enable wg-quick@wg0.service
```

Édition de la résolution de nom

Afin de simplifier mes scripts futurs, j'édite les fichiers "/etc/hosts" de chaque machine afin de pouvoir me passer de leurs IPs

Sur rose

```
vi /etc/hosts
```



```
127.0.0.1 localhost
10.254.0.2 George
```

Sur George

```
vi /etc/hosts
127.0.0.1 localhost
10.254.0.1 rose
```

Enfin, je teste la connectivité :

```

root@george:~# ping rose
PING rose (10.254.0.1) 56(84) bytes of data.
64 bytes from rose (10.254.0.1): icmp_seq=1 ttl=64 time=28.2 ms
64 bytes from rose (10.254.0.1): icmp_seq=2 ttl=64 time=28.3 ms
^C
--- rose ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.184/28.226/28.269/0.042 ms
root@george:~#

```

```

root@rose:~# ping george
PING george (10.254.0.2) 56(84) bytes of data.
64 bytes from george (10.254.0.2): icmp_seq=1 ttl=64 time=28.0 ms
64 bytes from george (10.254.0.2): icmp_seq=2 ttl=64 time=28.3 ms
^C
--- george ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.029/28.143/28.257/0.114 ms
root@rose:~#

```

Mise en œuvre de la sauvegarde

Mes 2 serveurs pouvant maintenant communiquer librement, je vais mettre en œuvre une sauvegarde quotidienne avec une durée de rétention de 1 mois.

Pour ce je vais utiliser :

- rdiff-backup qui est un logiciel de sauvegarde pour les systèmes Unix et Windows Combinant les avantages de la sauvegarde incrémentielle et du miroir ; voici ses caractéristiques principales :
 - Sauvegarde incrémentielle : rdiff-backup stocke les différences entre les sauvegardes successives, ce qui permet d'économiser de l'espace disque tout en conservant la capacité de restaurer différentes versions d'un fichier sur une période donnée.
 - Facilité de restauration : La dernière version des fichiers sauvegardés est toujours une copie directe, ce qui facilite la restauration. Les versions antérieures sont reconstruites en appliquant des différences incrémentielles.

- Transfert de données optimisé : Lors de la sauvegarde de données sur un réseau, rdiff-backup minimise la quantité de données transférées en envoyant uniquement les différences entre les fichiers sources et ceux des précédentes sauvegardes.
- Conservation des métadonnées : Il préserve et restaure diverses métadonnées (comme les permissions de fichier, les liens symboliques, les informations utilisateur/groupe), ce qui est crucial pour la restauration fidèle de l'environnement sauvegardé.
- Compression des données : Les données de sauvegarde peuvent être compressées pour économiser encore plus d'espace.
- Support multi-plateforme : Compatible avec divers systèmes d'exploitation comme les distributions Linux, UNIX et Windows.
- Scriptable : Peut-être intégré dans des scripts pour automatiser les tâches de sauvegarde.
- n8n qui est un outil d'automatisation de flux de travail open source qui permet d'intégrer et de connecter divers systèmes et services. Voici quelques-unes de ses caractéristiques clefs :
 - Intégrations multiples : n8n offre une vaste gamme de nœuds intégrés pour se connecter à des applications et des services populaires tels que Google Sheets, Slack, Trello, GitHub, et bien d'autres. Cela permet d'automatiser les interactions entre différents outils et plateformes.
 - Conception visuelle : Il utilise une interface graphique pour créer des flux de travail, permettant aux utilisateurs de visualiser et de manipuler le processus d'automatisation sans nécessiter de compétences de codage approfondies.

On notera que pour ce sujet, l'usage de n8n est « surdimensionné », un cron aurait très bien fait l'affaire, mais il s'agit pour moi de commencer à manipuler cet outil sur un cas « simple »

Configuration initiale

Je configure les accès ssh et les droits sudo suivants (je ne détaille pas la configuration par souci de confidentialité de ces credentials) :

Sur Rose

- J'ajoute root@george dans le fichier authorized_keys l'utilisateur « root » (requis pour la commande rdiff-backup)
- J'ajoute sauvegarde@george dans le fichier authorized_keys de l'utilisateur « sauvegarde » (utilisateur créer spécifiquement pour la sauvegarde)
- Je créer le fichier /etc/sudoers.d/10-sauvegarde tel que :

```
sudo for user sauvegarde
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/docker ps
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/docker stop
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/docker start
```

Sur George

- J'ajoute sauvegarde@n8n dans le fichier authorized_keys l'utilisateur « sauvegarde » (requis pour que n8n puisse lancer mes commandes)

- Je créer le fichier `/etc/sudoers.d/10-sauvegarde` tel que :

```
sudo for user sauvegarde
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/mount
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/umount
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/sshfs
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/rdiff-backup
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/scp
```

Installation de n8n

Pour installer n8n, je créer un nouveau conteneur docker sur rose (voir [annexe](#))

Puis, je modifie mon le conteneur de mon reverse proxy npm en ajoutant l'accès au réseau de ma stack n8n (voir [annexe](#))


Enfin, je configure l'accès à mon instance n8n sur mon reverse proxy

J'ai ensuite ajouté l'entrée dns de mon nouvel hôte sur l'interface de mon registrar

Du fait d'erreur d'accès du conteneur n8n, j'ai dû éditer les droits suivants :

```
sudo useradd -u 1000 default_user
sudo chown default_user:default_user /opt/docker/rose/n8n/data/.n8n
```

Une fois mon conteneur démarré, j'ai créé mon compte :



Set up owner account

Email *

gmail.com

First Name *

Last Name *

Password *

8+ characters, at least 1 number and 1 capital letter

☒ Inform me about security vulnerabilities if they arise

Next

Dès lors, mon système devant récupérer des droits avancés d'administration, j'ai renforcé la sécurité de mon compte utilisateur en configurant le 2FA

Security

Password

[Change password](#)

Two-factor authentication (2FA)


Two-factor authentication is currently disabled. [Learn more](#)

[Enable 2FA](#)

Setup Authenticator app [1/2]

1. Scan the QR code

Use an authenticator app from your phone to scan. If you can't scan the QR code, enter [this text code](#)



2. Enter the code from the app

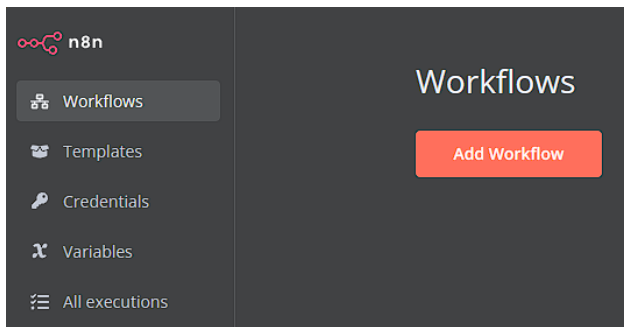
Code from your authenticator app

e.g. 123456

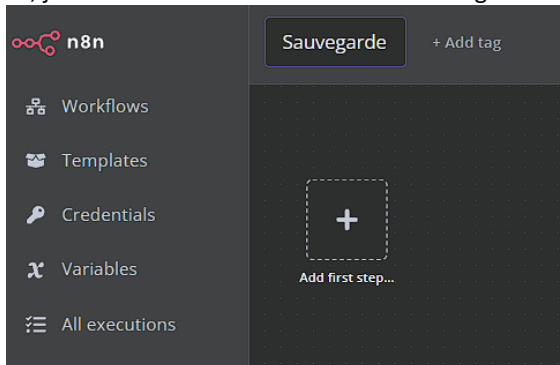
Continue

Création du workflow de sauvegarde sur n8n

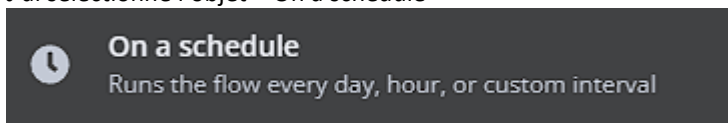
Sur l'interface, dans le menu « Workflows » j'ai cliqué sur « Add Workflow »



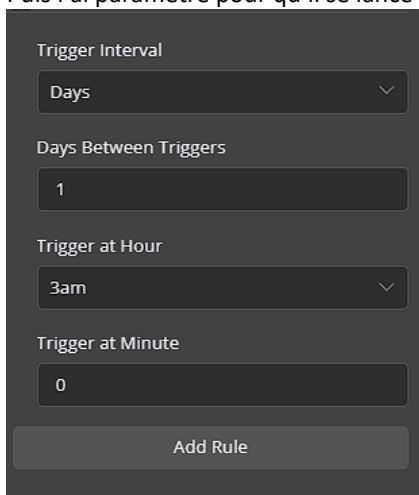
Là, j'ai renommé mon workflow en « Sauvegarde » puis ai ajouté ma première étape en cliquant sur « Add first step »



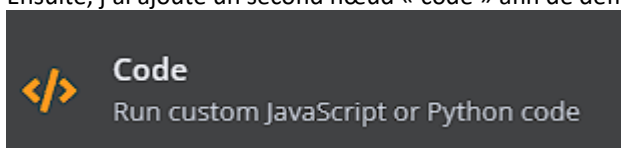
J'ai sélectionné l'objet « On a schedule »



Puis l'ai paramétré pour qu'il se lance tous les jours à 3 heures du matin :



Ensuite, j'ai ajouté un second nœud « code » afin de définir mes variables d'entrée



Je l'ai configuré ainsi :

setVarRoseBkp Execute node

Parameters Docs

Mode
Run Once for All Items

Language
JavaScript

JavaScript

```

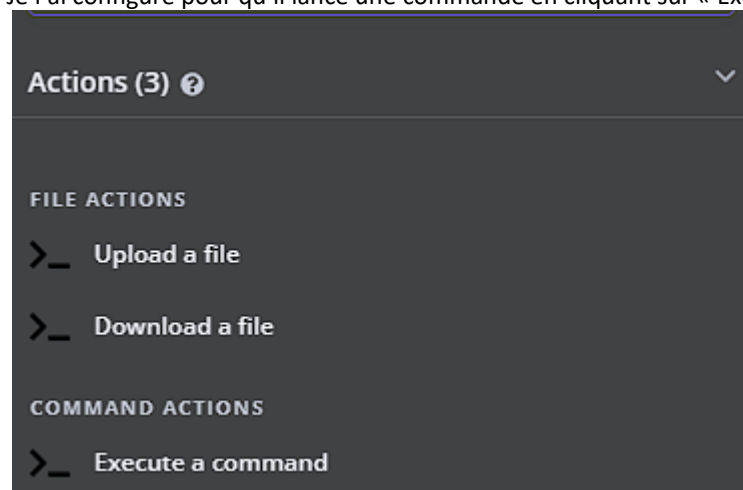
1  var entree = [
2    { user: "root@rose" },
3    { repertoireASauvegarder: "/opt/docker/rose" },
4    { repertoireTemporaire: "/mnt/docker_rose" },
5    { repertoireDeSauvegarde: "/opt/docker/rose_bkp/" },
6  ];
7
8  // Créer un nouvel objet qui combine toutes les propriétés des objets du tableau
9  var objetCombine = entree.reduce(function(resultat, objetCourant) {
10   for (var cle in objetCourant) {
11     resultat[cle] = objetCourant[cle];
12   }
13   return resultat;
14 }, {});
15
16 // Maintenant `objetCombine` contient toutes les propriétés
17 return objetCombine; // Cela retournera un seul objet combiné
18

```

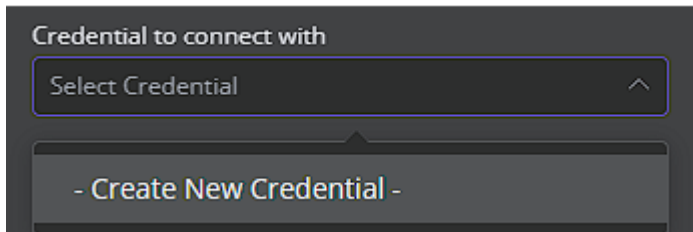
Pour poursuivre, j'ai ajouté un objet ssh :



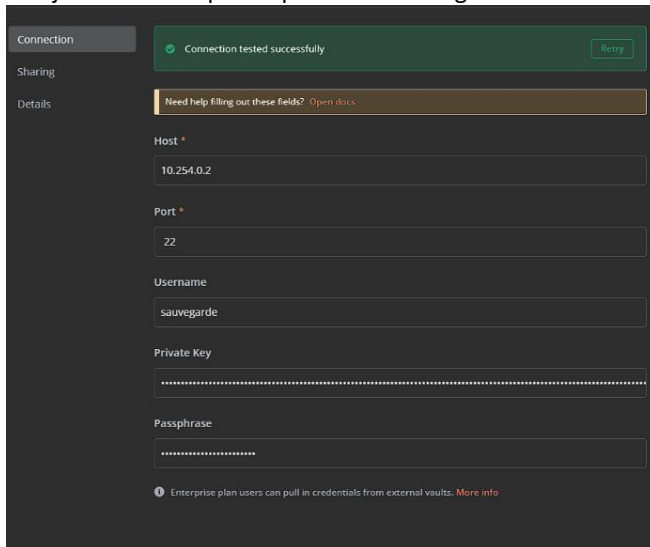
Je l'ai configuré pour qu'il lance une commande en cliquant sur « Exécute a command »



Immédiatement après j'ai configuré mon accès à mon serveur Georges



En ajoutant la clef privée précédemment générée

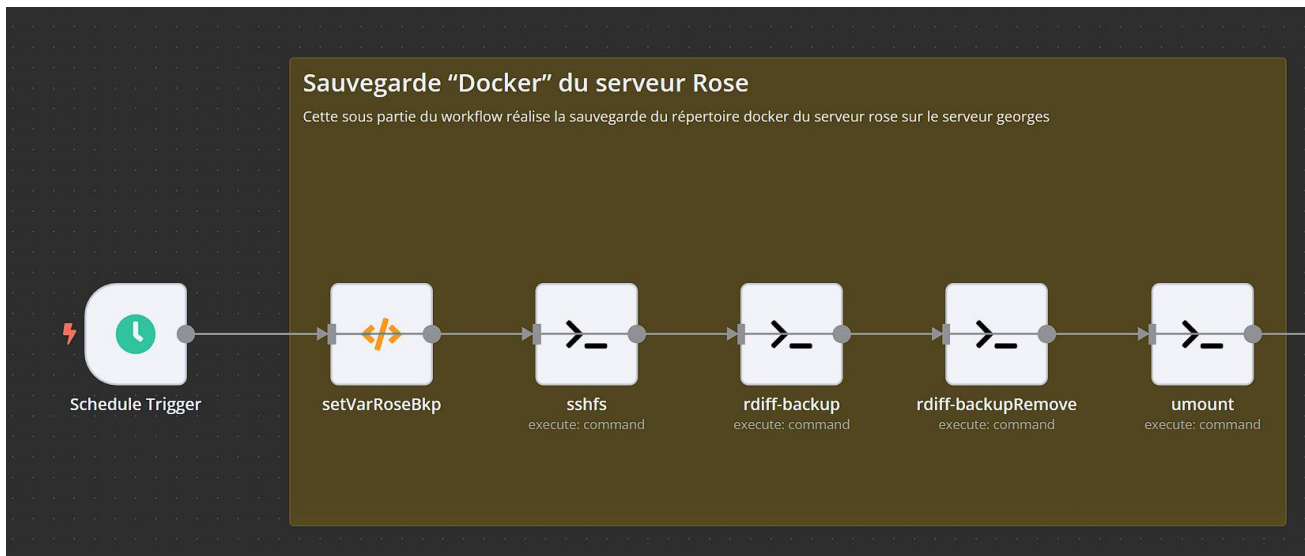


Ici j'ai tout d'abord testé l'exécution d'un « ps » sur mon serveur pour m'assurer de la bonne exécution de mes commandes

Une fois cela fait, j'ai dupliqué 3 fois ce bloc et les ai configurés ainsi

Nom du bloc	Commande (au format « Expression ») Fixed Expression	Explication
sshfs	sudo sshfs {{\${node['setVarRoseBkp']}.json['user']}}:{{\${node['setVarRoseBkp']}.json['repertoireASauvegarder']}} {{\${node['setVarRoseBkp']}.json['repertoireTemporaire']}} 2>&1	Cette commande monte le dossier « /opt/docker/rose » du serveur rose sur « /mnt/docker_rose » sur le serveur George
rdiff-backup	sudo rdiff-backup {{\${node['setVarRoseBkp']}.json['repertoireTemporaire']}} {{\${node['setVarRoseBkp']}.json['repertoireDeSauvegarde']}} } 2>&1	Cette commande exécute la mise à jour de la sauvegarde « /opt/docker/rose_bkp/ » depuis les données « /mnt/docker_rose »
rdiff-backupRe move	sudo rdiff-backup --remove-older-than 1M --force {{\${node['setVarRoseBkp']}.json['repertoireDeSauvegarde']}} } 2>&1	Cette commande supprime les incréments de plus d'un mois
umount	sudo umount {{\${node['setVarRoseBkp']}.json['repertoireTemporaire']}} 2>&1	Cette commande démonte le répertoire réseau

Une fois la configuration terminée, cela nous donne ceci



Une fois le traitement exécuté, mes données sont bien mises à jour

```

root@george:/opt/docker/rose_bkp# ls -rtl
total 48
drwxr-xr-x 6 root root 4096 Aug 25 11:47 portainer
drwxr-xr-x 5 root root 4096 Aug 30 16:58 http_proxy
drwxr-xr-x 6 root root 4096 Aug 31 15:55 passbolt
drwxr-xr-x 4 root root 4096 Nov 23 13:35 n8n
drwx----- 3 root root 4096 Jan 22 22:13 rdiff-backup-data
  
```

Mise en œuvre de la j-1 passbolt

Ma sauvegarde étant opérationnelle, j'automatise maintenant ma synchro « j-1 ». Il s'agit pour moi de m'assurer que même en cas de défaillance de mon hébergeur que je puisse toujours disposer d'une copie récente (celle de la nuit passée) de mes mots de passe sur mon serveur local.

Volontairement, je ne vous communique pas le dockerfile « Passbolt » ni ma configuration npm (reverse proxy) locale afin que ceux qui le souhaitent puissent réaliser leur dossier sur ce logiciel. Mais pour ceux qui voudrait tenter de reproduire cet exemple, sachez que les seuls paramètres que j'ai changés entre le docker file de prod et le docker file « j-1 » sont l'URL et les volumes.

Configuration initiale

Je configure les accès ssh et les droits sudo suivants (je ne détaille pas la configuration par souci de confidentialité de ces crédits) :

Sur George

- J'édite le fichier /etc/sudoers.d/10-sauvegarde en ajoutant la ligne :

```
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/scp
```

sur Fujiyama

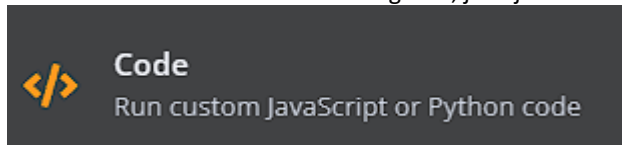
- J'ajoute root@george dans le fichier authorized_keys l'utilisateur « root » (requis pour la commande rcp)

- J'ajoute sauvegarde@george dans le fichier authorized_keys de l'utilisateur « sauvegarde » (utilisateur créer spécifiquement pour la sauvegarde)
- Je créer le fichier /etc/sudoers.d/10-sauvegarde tel que :

```
sudo for user sauvegarde
sauvegarde ALL=(ALL) NOPASSWD:/usr/bin/docker
```

Édition du workflow de sauvegarde

Dans la suite du workflow de sauvegarde, j'ai ajouté un nœud « code » afin de définir mes variables d'entrée



Je l'ai configuré ainsi :

```
setVar

Parameters Docs

Mode
Run Once for All Items

Language
JavaScript

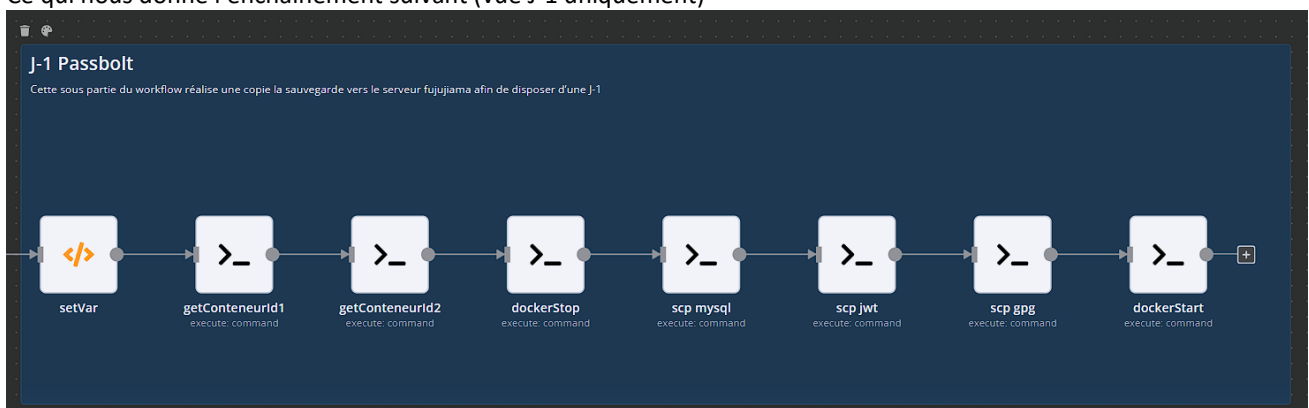
JavaScript
1 var entree = [
2   { user1: "sauvegarde@fujuiama" },
3   { user2: "root@fujuiama" },
4   { repertoireDeSauvegarde: "/opt/docker/rose_bkp/passbolt" },
5   { repertoireJ1: "/opt/docker/fujuiama/passboltj-1" },
6   { conteneur1: "passbolt_app" },
7   { conteneur2: "passbolt_db" }
8 ];
9
10 // Créer un nouvel objet qui combine toutes les propriétés des objets du tableau
11 var objetCombine = entree.reduce(function(resultat, objetCourant) {
12   for (var cle in objetCourant) {
13     resultat[cle] = objetCourant[cle];
14   }
15   return resultat;
16 }, {});
17
18 // Maintenant 'objetCombine' contient toutes les propriétés
19 return objetCombine; // Cela retournera un seul objet combiné
20
```

Ensuite, j'ai créé 7 nœuds ssh tel que :

Nom du bloc	Commande (au format « Expression ») Fixed Expression	Explication
getConteneurId1	ssh {{\$node['setVar'].json['user1']}} sudo docker ps grep {{\$node['setVar'].json['conteneur1']}} awk '{print \$1}'	Je récupère l'id du conteneur passbolt_app
getConteneurId2	ssh {{\$node['setVar'].json['user1']}} sudo docker ps grep {{\$node['setVar'].json['conteneur2']}} awk '{print \$1}'	Je récupère l'id du conteneur passbolt_db
dockerStop	ssh {{\$node['setVar'].json['user1']}} sudo docker stop {{\$node['getConteneurId1'].json['stdout']}} {{\$node['getConteneurId2'].json['stdout']}}	J'arrête les conteneurs Passbolt sur fujuiama
scp mysql	sudo scp -r {{\$node['setVar'].json['repertoireDeSauvegarde']}}/mysql/* {{\$node['setVar'].json['user2']}}:{{\$node['setVar'].json['re	Je copie les données mysql

	pertoireJ1'}})/mysql/ 2>&1	
scp jwt	sudo scp -r {{\$node['setVar'].json['repertoireDeSauvegarde']}}/conf/jwt/* {{\$node['setVar'].json['user2']}}:{{\$node['setVar'].json['repertoireJ1']}}/conf/jwt/ 2>&1	Je copie les données « jwt »
scp pgp	sudo scp -r {{\$node['setVar'].json['repertoireDeSauvegarde']}}/conf/pgp/* {{\$node['setVar'].json['user2']}}:{{\$node['setVar'].json['repertoireJ1']}}/conf/pgp/ 2>&1	Je copie les données « pgp »
dockerStart	ssh {{\$node['setVar'].json['user1']}} sudo docker start {{\$node['setVar'].json['conteneur1']}} {{\$node['setVar'].json['conteneur2']}}	Je redémarre mes conteneurs

Ce qui nous donne l'enchaînement suivant (vue J-1 uniquement)



(vue complète)



Une fois le traitement terminé, je peux me connecter sur mon instance j-1 (à noter que l'instance est complètement décorrélée de mon instance de production, aussi à la 1re connexion, j'ai dû me réauthentifier avec ma clef privée)

← ↻ 🔒 https://secretj1 [REDACTED]

mots de passe utilisateurs administration aide

passbolt 🔑

+ Créer ⬆

Rechercher des mots de passe

📋 Copier ✎ Modifier

Tous les éléments

Favoris

Récemment modifiés

Partagés avec moi

Possédés par moi

Tous les éléments 534

<input type="checkbox"/> ★ Name
<input type="checkbox"/> ★ n8n - sshkey
<input type="checkbox"/> ★ n8n

Conclusion

En conclusion, ce projet m'a permis de mettre en place une architecture résiliente et sécurisée pour ma gestion des mots de passe basée sur Passbolt, d'apprendre l'usage de la solution Wireguard et de débiter la mise en œuvre de l'automatisation de mes actions récurrentes avec n8n.

Cependant, concernant Wireguard, il me reste à mettre en place une veille et une restriction d'accès dynamique via Crowdsec.

D'autre part, l'implémentation d'une stratégie de sauvegarde basée sur rdiff-backup est fonctionnelle, toutefois, pour garantir une consistance optimale des données, il conviendra que j'intègre un processus d'export/import SQL. Et, dans la foulée que je complète le dispositif par une sauvegarde complémentaire (j'envisage une sauvegarde mensuelle sur AWS Backup). En effet à l'heure actuelle, je ne respecte pas la « règle » du

- 3 : 3 copies de données > je ne suis qu'à 2 pour Passbolt, à 1 pour le reste.
- 2 : 2 médias différents > je ne suis qu'à 2 pour Passbolt, à 1 pour le reste.
- 1 : 1 copie hors ligne > je ne suis à 0 copie hors ligne
- 0 : 0 (voir paragraphe suivant)

Enfin, il est impératif que je peaufine le workflow de sauvegarde pour mieux gérer les erreurs potentielles, surveiller l'espace disque disponible et mettre en place un système d'alerte par e-mail en cas d'anomalie. Ces améliorations me permettront de m'assurer une plus grande résilience et fiabilité au système.

En résumé, bien que les objectifs initiaux du projet aient été atteints avec succès, les points d'amélioration identifiés offrent des pistes pour optimiser davantage le système. Ces ajustements permettront de renforcer la sécurité, la fiabilité et l'efficacité de la solution de gestion des mots de passe, assurant une tranquillité d'esprit accrue dans la gestion quotidienne de ces données au combien sensibles.

Sources

Sources WEB

n8n

- [Docker | n8n Docs](#)
- [n8n : TOUT COMPRENDRE pour DÉMARRER FACILEMENT - part 1 \(youtube.com\)](#)

Passbolt

- [Docker passbolt installation](#)

rdiff-backup

- [Sauvegardes avec RDiff Backup | n0tes.fr](#)

scp

- [Comment Utiliser La Commande SCP Linux - Guide Ultime](#)

sudo

- [15 exemples utiles de Sudoers pour configurer sudo sur Linux - malekal.com](#)

WireGuard

- [Installation - WireGuard](#)
- [Wireguard - Connexion du bureau à distance avec Wireguard \(youtube.com\)](#)
- [WireGuard vs OpenVPN: 7 Key Differences in 2024](#)

Annexes

Docker compose

n8n

```
version: "3.9"
services:
  n8n_app:
    image: n8nio/n8n:latest
    restart: unless-stopped
    container_name: n8n_app
    ports:
      - 5678:5678
    environment:
      - N8N_BASIC_AUTH=true
      - N8N_BASIC_AUTH_USERNAME=n8n
      - N8N_BASIC_AUTH_PASSWORD=XXXXXXXXX
      - N8N_HOST=pilote.vbn.ovh
      - N8N_PORT=5678
      - N8N_PROTOCOL=http
      - NODE_ENV=production
      - WEBHOOK_URL=http://n8n_app

    volumes:
      - /opt/docker/rose/n8n/data/.n8n:/home/node/.n8n
    networks:
      - n8n_BE
      - n8n_FE
    depends_on:
      - n8n_db

  n8n_db:
    image: mariadb:latest
    restart: unless-stopped
    container_name: n8n_db
    environment:
      - MYSQL_RANDOM_ROOT_PASSWORD=true
      - MYSQL_DATABASE=n8n
      - MYSQL_USER=n8n
      - MYSQL_PASSWORD=XXXXXXXXX
      - TZ=Europe/Paris
    volumes:
      - /opt/docker/rose/n8n/mysql:/var/lib/mysql/
    networks:
      - n8n_BE

networks:
  n8n_FE:
    external: true
  n8n_BE:
    external: true
```

Proxy npm

```
version: '3.8'
```

```
services:
  npm_app:
    image: 'jc21/nginx-proxy-manager:latest'
    container_name: npm_app
    restart: always
    ports:
      - '80:80'
      - '443:443'
    volumes:
      - /opt/docker/rose/http_proxy/data:/data
      - /opt/docker/rose/http_proxy/letsencrypt:/etc/letsencrypt
      - /opt/docker/rose/http_proxy/log:/data/log/
    environment:
      - TZ=Europe/Paris
    networks:
      - passbolt_FE
      - portainer_FE
      - n8n_FE

networks:
  passbolt_FE:
    external: true
  portainer_FE:
    external: true
  n8n_FE:
    external: true
```