SAÉ S4.Deploi.01 : Rendus Test Vortex Network

Nolan Dupont, Nils Rayot, Kylian Metayer, Jess Rousselard, Flavien Riondet

6 avril 2025

Table des matières

1 Script d'automatisation

1.1 Diffusion des clés SSH

Fonction

Ce script Bash automatise la détection des machines actives sur un réseau et tente d'établir une connexion SSH avec elles en utilisant une liste de mots de passe fournie dans un fichier. Il permet ensuite d'ajouter des clés SSH afin d'établir une connexion sécurisée sans mot de passe.

Environnement d'exécution

Pour fonctionner correctement, ce script nécessite les conditions suivantes :
— Il doit être exécuté sur le routeur du réseau privé.
— Un fichier `passwords.txt` doit être présent dans le répertoire d'exécution et contenir les mots de passe des stations, dans l'ordre.
— Les clés à diffuser doivent être stockées dans `/home/etu/.ssh/authorized_keys`.

Script

Vous pouvez le trouver dans le dossier `automatisation/script_ssh.sh`

Explication du script

1. Vérifications préliminaires
   — Vérifie l'existence du fichier contenant les mots de passe (`passwords.txt`).
   — Vérifie que les outils nécessaires (`nmap`, `sshpass`, `SSH`, `scp`) sont installés.
2. Scan du réseau
   — Parcourt plusieurs plages d'adresses IP (`192.168.1.0/24`, `192.168.10.0/24`, `192.168.2.0/24`).
   — Utilise `nmap` pour détecter les machines actives.
   — Stocke les adresses IP des machines détectées.
3. Tentative de connexion SSH
   — Essaye de se connecter aux machines détectées en utilisant les mots de passe de `passwords.txt`.
   — Si la connexion SSH réussit :
      — Vérifie et crée le fichier `authorized_keys` sur la machine cible.
      — Copie le fichier `authorized_keys` depuis la machine source vers la cible pour activer l'authentification par clé SSH.
      — Désactive l'authentification par mot de passe en modifiant `/etc/ssh/sshd_config` et en redémarrant le service SSH.
4. Gestion des erreurs et affichage
   — Gère les erreurs à chaque étape (échec de connexion, problème de copie, etc.).

— Affiche des messages détaillés pour informer l'utilisateur du statut des opérations.

1.2 Installation des agents zabbix-agent

Fonction

Ce script a pour but d'automatiser l'installation et la configuration de l'agent zabbix-agent sur un hôte. Il permet une intégration rapide de la machine dans le système de supervision.

Environnement d'exécution

Le script est prévu pour être exécuté sur un système GNU/Linux (Debian, Ubuntu, PVE) disposant d'un accès root. Il suppose un accès à internet pour pouvoir télécharger les paquets nécessaires depuis les dépôts officiels de zabbix-agent.

Script

Vous pouvez le trouver dans le dossier `automatisation/script_agent_zabbix.sh`

Explication du script

— Téléchargement du paquet `deb` permettant l'ajout du dépôt officiel zabbix-agent aux dépôts d'`apt`.
— Installation du paquet `deb`.
— Mise à jour de la liste des paquets via `apt update`.
— Installation du paquet `zabbix-agent`.
— Modification du fichier de configuration de l'agent :
  — Définition de l'adresse du serveur zabbix-agent.
  — Définition du nom de l'hôte (hostname) tel qu'il apparaîtra dans l'interface zabbix-agent.
— Démarrage et activation du service `zabbix-agent` pour qu'il se lance au démarrage.
— Vérification du bon fonctionnement avec la commande `systemctl` status.

1.3 Augmentation du stockage des hyperviseurs

Fonction

Ce script Bash permet de réduire la taille du volume logique de swap (`pve/swap`) puis de ré allouer l'espace libéré au volume de données (`pve/data`).

Environnement d'exécution

Ce script est destiné à être lancé sur un système utilisant PVE, avec une gestion des volumes via LVM (comme c'est le cas de Proxmox). Dans notre cas, nous l'avons utilisé

pour agrandir l'espace disponible sur l'hyperviseur Proxmox. Ce script est écrit pour être exécuté en tant que root.

Script

Vous pouvez le trouver dans le dossier `automatisation/script_swap.sh`

Explication du script

1. Affiche la structure des disques avec la commande `lsblk`.
2. Demande à l'utilisateur combien de mégaoctets de swap il souhaite retirer.
3. Vérifie que la valeur entrée est bien un nombre valide.
4. Récupère la taille actuelle du swap.
5. Calcule la nouvelle taille du swap après soustraction.
6. Vérifie que la nouvelle taille est positive et inférieure à l'ancienne.
7. Désactive temporairement le swap.
8. Réduit la taille du volume logique du swap à la nouvelle valeur.
9. Reformate le volume swap avec `mkswap`.
10. Réactive le swap.
11. Étend le volume logique de données pour utiliser l'espace libéré.
12. Répare le thin pool `pve/data` si nécessaire.
13. Affiche l'état final des disques pour vérification.

2 Script de test

Pour assurer la maintenabilité et analyser ou sont les problèmes nous avons écrit des scripts de test. Nous allons les détailler ci-dessous.

2.1 Exécution des test

Pour commencer dans le fichier `/script/tests/run.sh`. Il sert a mettre en place l'environnement d'exécution des test,puis a tous les exécuter en effectuant la liste d'étapes suivante :

1. **Initialisation de l'environnement virtuel** :
   Le script vérifie si le dossier `.venv` existe. S'il n'existe pas, il crée un environnement virtuel Python avec `python3 -m venv`, l'active, puis installe les dépendances listées dans `requirements.txt`.
2. **Préparation des interpréteurs Python** :
   Le script crée un dossier `.pythons` pour stocker différentes versions de Python. Il utilise `python-build` (associé à `pyenv`) pour installer les versions indiquées dans le tableau `PYTHON_VERSIONS`.

3. **Installation des dépendances pour chaque version** :
   Si la variable d'environnement `DEPENDANCE` vaut `0`, le script installe silencieusement les dépendances dans chaque interpréteur Python, puis exécute un test de vérification avec `pytest` sur `test_installation_python.py`.
4. **Exécution des tests** :
   Si la variable `AVIS` vaut `0`, le script exécute `pytest` avec tous les interpréteurs installés sur le fichier `generic.py`, en passant une IP et une liste de ports. Sinon, il utilise uniquement Python 3.13.

Lors de l'exécution de se script tout les scripts suivants sont exécuter :

2.2 Test : Configuration SSH de la machine hôte

Objectif

Vérifier que le service `openssh-server` est correctement installé et configuré de manière sécurisée sur une machine hôte.

Dépendance

Le test s'appuie sur la bibliothèque `pytest` et les outils de `testinfra`, ainsi que sur un système de journalisation via `logging`.

Vérifications effectuées

- **Installation du service SSH :** Vérifie que `openssh-server` est bien installé.
- **Existence du fichier de configuration :** Vérifie la présence du fichier `/etc/ssh/sshd_config`.
- **Accès root désactivé :** Vérifie que `PermitRootLogin` est positionné sur `no`.
- **Port standard SSH :** Vérifie que le port 22 est bien utilisé.
- **Authentification sécurisée :**
  - `PasswordAuthentication` doit être désactivé (`no`).
  - `PubkeyAuthentication` doit être activé (`yes`).

Code

Vous pouvez le trouver dans le dossier `tests/serveur.py`

2.3 Test : Connectivité réseau et ouverture des ports

Objectif

Vérifier la connectivité réseau d'un appareil via son adresse IP et tester si certains ports sont bien ouverts à l'écoute.

Dépendance

Utilise :
— la commande système `ping`,
— la bibliothèque `nmap` (Python-nmap).

Vérifications effectuées

— **Connectivité IP :** Teste la réponse ICMP (via `ping`).
— **État des ports :** Pour chaque port listé, vérifie qu'il est en état `open`.

Code

Vous pouvez le trouver dans le dossier `tests/generic.py`

2.4 Test : Attribution d'une adresse IP via DHCP

Objectif

Vérifier que le client DHCP de la machine obtient correctement une adresse IP depuis un serveur DHCP, et que cette adresse est bien appliquée à une interface réseau spécifique.

Dépendances

Le test repose sur les éléments suivants :
— La bibliothèque `pytest` pour organiser et exécuter les tests.
— La bibliothèque `testinfra` (présumée) pour interagir avec la machine distante via la *fixture* `host`.
— Le module `logging` pour consigner des informations lors des tests.
— Le module `json` est importé mais non utilisé dans le code fourni.

Vérifications effectuées

— **Réception d'une réponse DHCP :**
  — La commande `dhclient -r` libère l'adresse IP existante.
  — La commande `dhclient -v` effectue une nouvelle requête DHCP.
  — Le test vérifie la présence du message `DHCPACK` pour s'assurer que le serveur a bien répondu.
  — Il vérifie également la mention `bound to` pour confirmer qu'une adresse IP a été attribuée.
— **Attribution effective de l'IP :**
  — Le test vérifie la présence d'une adresse IP sur l'interface `ens18` via la commande `ip addr show`.
  — La présence du mot-clé `inet` dans la sortie permet de valider l'attribution.

Code

Vous pouvez le trouver dans le dossier `tests/dhcp-client.py`

2.5 Test : Configuration du serveur DHCP Kea

Objectif

Vérifier que le serveur DHCP `kea-dhcp4-server` est correctement installé, activé, exécuté et configuré conformément aux bonnes pratiques, à travers l'analyse de son fichier de configuration `kea-dhcp4.conf`.

Dépendances

Le test repose sur les éléments suivants :
- La bibliothèque `pytest` pour la structuration et l'exécution des tests.
- La bibliothèque `testinfra` pour l'interaction avec la machine distante via la *fixture* `host`.
- Le module `json` pour parser le fichier de configuration Kea.
- Le module `logging` pour consigner les actions et les vérifications.

Vérifications effectuées

- **Installation du service :**
  - Le paquet `kea-dhcp4-server` doit être installé.
  - Le binaire `kea-dhcp4` doit exister dans le `PATH`.
  - Le service `kea-dhcp4-server` doit être activé et en cours d'exécution.
- **Fichier de configuration :**
  - Le fichier `/etc/kea/kea-dhcp4.conf` doit exister et être un fichier valide.
  - Il est chargé et interprété au format JSON.
- **Interfaces réseau DHCP :**
  - Au moins une interface doit être configurée.
  - Les noms doivent correspondre aux standards (`eth` ou `en`).
- **Paramètres de durée des baux :**
  - `valid-lifetime` = 691200 secondes.
  - `renew-timer` = 345600 secondes.
  - `rebind-timer` = 604800 secondes.
- **Mode authoritative :**
  - Le serveur doit être défini comme `authoritative`.
- **Base de données des baux :**
  - Type = `memfile`.
  - `persist` doit être activé.
  - Chemin du fichier : `/var/lib/kea/kea-leases4.csv`.
  - `lfc-interval` = 3600 secondes.
- **Configuration des sous-réseaux :**

  - Au moins un sous-réseau doit être défini.
  - Chaque sous-réseau doit contenir un champ `subnet` et au moins un `pool`.
- **Options DHCP :**
  - L'option `domain-name-servers` doit être définie.
  - L'option `routers` doit être présente.
- **Réservations d'adresses IP :**
  - Si des réservations existent, elles doivent inclure :
    - Une adresse MAC (`hw-address`),
    - Une adresse IP réservée (`ip-address`),
    - Un nom d'hôte (`hostname`).

Code

Vous pouvez le trouver dans le dossier `tests/dhcp.py`

2.6 Test : Configuration du serveur DNS BIND9

Objectif

Vérifier que le serveur DNS `BIND9` est installé, en fonctionnement, et que ses fichiers de configuration sont correctement structurés et renseignés pour le domaine `vortex-network.fr`.

Dépendances

Ce test repose sur les éléments suivants :
- La bibliothèque `testinfra`, utilisée via la classe parente `TestServer`, pour interagir avec le système distant.
- Le module `pathlib.Path` pour la manipulation des chemins de fichiers.
- Un système de journalisation via `logger` (hérité de `TestServer`).

Vérifications effectuées

- **Installation du paquet :** Vérifie que le paquet `bind9` est bien installé.
- **Service actif :** Le service `bind9` doit être activé et en cours d'exécution.
- **Fichier `named.conf.options` :**
  - Le fichier `/etc/bind/named.conf.options` doit exister et être un fichier valide.
- **Fichier `named.conf.local` :**
  - Le fichier `/etc/bind/named.conf.local` doit exister.
  - Il doit contenir la configuration de la zone `vortex-network.fr`.
  - Le type doit être `master`, avec fichier source `db.vortex-network.fr` et `allow-update` désactivé.
- **Fichier de zone DNS (`db.vortex-network.fr`) :**
  - Le fichier `/etc/bind/db.vortex-network.fr` doit exister et être un fichier.
  - Il doit contenir les enregistrements suivants :
    - Un enregistrement `NS` pour `srv-dns.vortex-network.fr`.

— Un enregistrement `A` pour `srv-dns` (192.168.1.10).
— Des enregistrements `A` pour toutes les machines du datacenter (dhcp, web, sgbd, file, log, cas, ipam, ldap, superviseur, etc.).
— Des enregistrements `A` pour les postes clients et administratifs.
— Des alias en `CNAME` pour des noms abrégés (ex. `web`, `dhcp-p`, `superviseur`, etc.).

Code

Vous pouvez le trouver dans le dossier `tests/dns.py`

Glossaire

lsblk Outil Linux qui affiche la structure des disques et des partitions sous forme d'arborescence, permettant de visualiser les périphériques de stockage disponibles sur un système. [**Wiki_Lsblk**] . 4

LVM Logical Volume Manager, une technologie de gestion des volumes logiques sous Linux, permettant de gérer plus facilement les volumes de stockage et d'adapter leur taille. [**Wiki_LVM**] . 3

mkswap Outil Linux permettant de préparer une partition ou un fichier pour l'utilisation comme espace de swap (mémoire virtuelle). [**Wiki_Mkswap**] . 4

nmap Network Mapper, un outil open source utilisé pour scanner les réseaux, découvrir des hôtes et des services, et tester la sécurité d'un réseau. [**Wiki_Nmap**] . 2

PVE Proxmox Virtual Environment, une plateforme de virtualisation qui permet de gérer des machines virtuelles et des conteneurs à l'aide de la gestion LVM et d'autres outils.. 3

sshd_config Fichier de configuration du serveur SSH, utilisé pour configurer les paramètres de connexion du service SSH. [**Wiki_Sshd_config**] . 2

sshpass Un utilitaire qui permet de fournir un mot de passe de manière non interactive à la commande SSH pour l'automatisation des connexions. [**Wiki_Sshpass**] . 2

thin pool Un type de volume logique sous LVM qui permet une gestion dynamique et flexible de l'espace de stockage, souvent utilisé pour les environnements virtuels. . 4

zabbix-agent Agent de surveillance Zabbix, un logiciel utilisé pour collecter des informations sur un hôte à des fins de supervision par le serveur Zabbix. [**Wiki_Zabbix-agent**] . 1, 3

Acronymes

SSH Secure Shell. 1, 2