

Contexte & objectifs :**Contexte :**

Dans le cadre du développement et de la maintenance de l'infrastructure, l'équipe doit accomplir plusieurs missions nécessitant la rédaction de playbooks et de documents. Ces documents seront partagés au sein de l'équipe, rendant crucial l'établissement de dépôts Git accessibles via SSH pour le stockage et la gestion des travaux. L'évaluation de l'utilisation de Git se fera en tenant compte de divers paramètres, tels que le nombre de commits.

Objectif de la Mission I6 - Automatisation avec Ansible de l'installation des serveurs ap31-prod et ap31-test :

L'objectif principal est de permettre une installation rapide et fiable des serveurs ap31-prod et ap31-test en utilisant Ansible. Les playbooks associés à cette mission ont permis de réaliser les actions suivantes :

1. Installer Apache2, PHP, et le gestionnaire de bases de données MariaDB avec le playbook apbase.yml.
2. Créer la base de données MySQL, les comptes associés, et injecter le dump de la base avec le playbook apdb.yml.
3. Récupérer localement un dump de la base de données distante avec le playbook apdbdump.yml.

Objectifs spécifiques de la Mission I6 (I6-c1, I6-c2, I6-c3) :

- Les serveurs doivent être opérationnels avec Apache2, PHP, et MariaDB installés (I6-c1).
- La base de données doit être créée, les comptes associés doivent être créés, et le dump de la base doit être injecté (I6-c2).
- Le dump de la base de données doit être stocké localement (I6-c3).

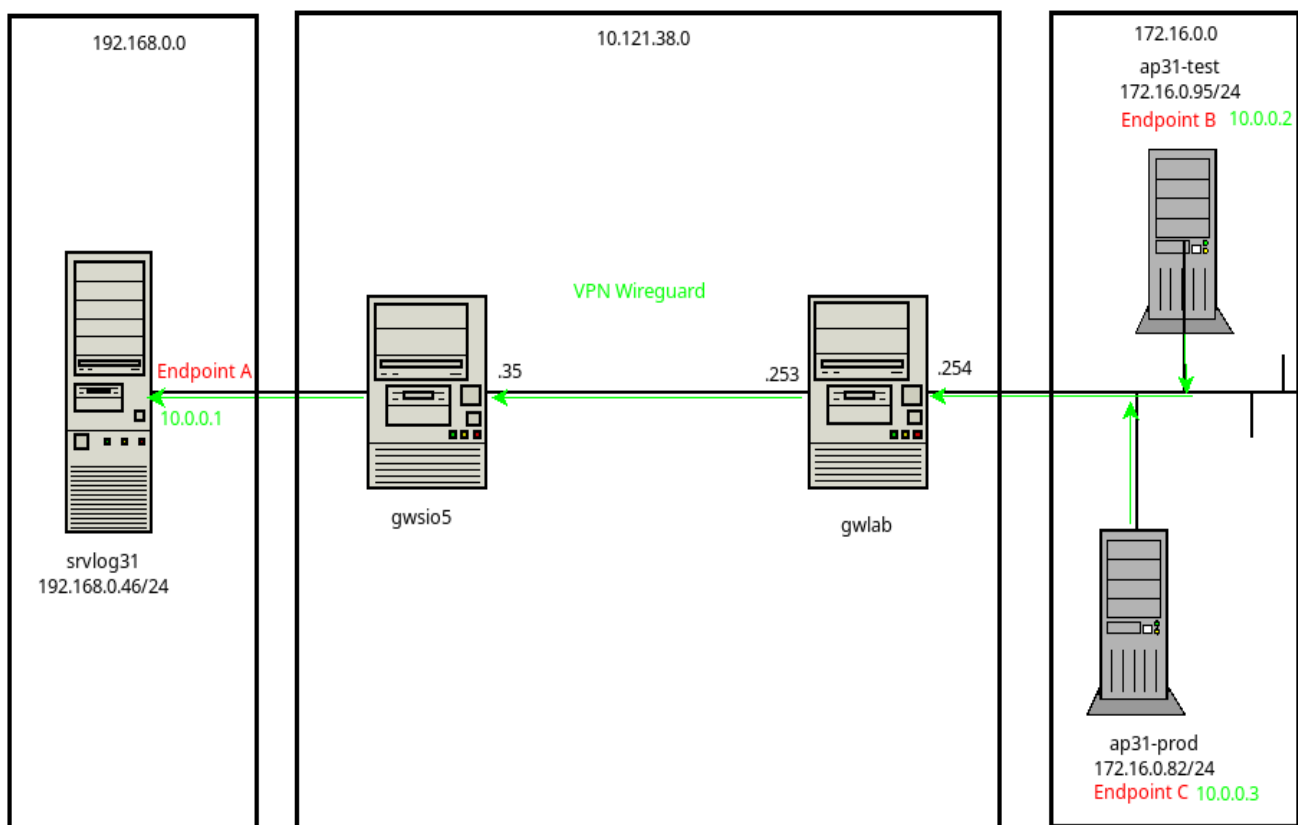
Objectif de la Mission I7 - Mise en œuvre d'un VPN reliant les serveurs ap31-test et ap31-prod avec un serveur de logs (srvlog31) :

La mise en place d'un VPN (Wireguard) a pour objectif de permettre le transfert sécurisé des logs des serveurs ap31-prod et ap31-test vers le serveur srvlog31. Cela garantit que les informations ne circulent pas en clair sur le réseau.

Objectifs spécifiques de la Mission I7 (I7-c1, I7-c2) :

- Le VPN doit être opérationnel, assurant la joignabilité des trois machines (ap31-prod, ap31-test, srvlog31) via le VPN (I7-c1).
- Les logs des serveurs ap31-prod et ap31-test doivent être envoyés à la machine srvlog31 via le VPN (I7-c2).

Tous les fichiers de configuration sont déposés dans [notre dépôt Git \(uap21-2024-rd\)](#).

Schéma Réseau :

MISSION 6 : PlayBook avec Ansible

Script apbase.yml

```
---
- name: install ap et bd
  #hosts: ap31-prod, ap31-test
  #hosts: ap31-prodt
  hosts: all
  # become_method: sudo
  # remote_user: debian
  become: yes

tasks:
- name: install apache2, php, mariadb-server, python3-pymysql
  ansible.builtin.apt:
    name:
      - apache2
      - php
      - mariadb-server
      - python3-pymysql
    state: present
```

Script apdb.yml

```
---
- name: Creer la BDD, creer les comptes et injecter la BDD
  hosts: all
  become: true

tasks:
- name: creation de la BDD sdis29-1
  community.mysql.mysql_db:
    name: sdis29-1
    state: present
    login_unix_socket: /var/run/mysqld/mysqld.sock

- name: copie de la base de donnees
  copy:
    src: sdis2023.sql
    dest: /tmp

- name: restauration de la base de donnees
  community.mysql.mysql_db:
    name: sdis29-1
    state: import
    target: /tmp/sdis2023.sql
    force: true
    login_unix_socket: /var/run/mysqld/mysqld.sock

- name: creation de lutilisateur slam
  community.mysql.mysql_user:
    name: slam
    password: Azerty1+
    priv: ' *.*:ALL,GRANT'
    state: present
    login_unix_socket: /var/run/mysqld/mysqld.sock
```

Script apdbdump.yml

```
---
- name: recuperer localement un dump de la BDD distante
  hosts: all
  become: true

tasks:
- name: Copie du dump distant
  ansible.builtin.fetch:
    src: /tmp/sdis2023.sql
    dest: /root/sauvegarde/sql/
    flat: true
```

Hosts

ap31-prodt

on peut rajouté d'autres hôtes, les clés SSH doivent être mises dans .ssh/authorized_keys

MISSION 7 : VPN et redirection de logs

On a mis jour les machines et on a installé Wireguard et ses outils :

```
apt update && apt upgrade -y  
apt install -y wireguard wireguard-tools
```

On a vérifié que le démon est lancé automatiquement et on a vérifié son statut :

```
systemctl enable --now wg-quick@wg0  
systemctl status wg-quick@wg0
```

On a édité puis exécuté le script de configuration de Wireguard pour avoir les fichiers de configuration pour le serveur de logs srvlog31 et les serveurs ap31-test et ap31-prod :

```
nano mkwgconf-p2p.sh  
sudo bash mkwgconf-p2p.sh
```

mkwgconf-p2p.sh

```
#!/bin/bash  
set -u  
set -e  
  
AddressAwg=10.0.0.1/32 # Adresse VPN Wireguard extremite A  
EndpointA=10.121.38.35 # Adresse extremite A (gwsio avec redirection vers srvlog31)  
PortA=51820           # Port ecoute extremite A  
  
AddressBwg=10.0.0.2/32 # Adresse VPN Wireguard extremite B  
EndpointB=172.16.0.95 # Adresse extremite B (ap31-test)  
PortB=51820           # Port ecoute extremite B  
  
AddressCwg=10.0.0.3/32 # Adresse VPN Wireguard extremite C  
EndpointC=172.16.0.82 # Adresse extremite C (ap31-prod)  
PortC=51820           # Port ecoute extremite C  
  
umask 077 ;  
wg genkey > endpoint-a.key  
wg pubkey < endpoint-a.key > endpoint-a.pub  
  
wg genkey > endpoint-b.key  
wg pubkey < endpoint-b.key > endpoint-b.pub  
  
wg genkey > endpoint-c.key  
wg pubkey < endpoint-c.key > endpoint-c.pub  
  
PKA=$(cat endpoint-a.key)  
pKA=$(cat endpoint-a.pub)  
  
PKB=$(cat endpoint-b.key)  
pKB=$(cat endpoint-b.pub)  
  
PKC=$(cat endpoint-c.key)  
pKC=$(cat endpoint-c.pub)  
  
cat <<FINI > wg0-a.conf  
# local settings for Endpoint A  
[Interface]  
PrivateKey = $PKA  
Address = $AddressAwg  
ListenPort = $PortA  
  
# remote settings for Endpoint B  
[Peer]  
PublicKey = $pKB  
Endpoint = ${EndpointB}:$PortB  
AllowedIPs = $AddressBwg  
  
# remote settings for Endpoint C  
[Peer]  
PublicKey = $pKC
```

```
Endpoint = ${EndpointC}:$PortC
AllowedIPs = $AddressCwg
FINI
```

```
cat <<FINI > wg0-b.conf
# local settings for Endpoint B
[Interface]
PrivateKey = $PKB
Address = $AddressBwg
ListenPort = $PortB

# remote settings for Endpoint A
[Peer]
PublicKey = $pKA
Endpoint = ${EndpointA}:$PortA
AllowedIPs = $AddressAwg
FINI
```

```
cat <<FINI > wg0-c.conf
# local settings for Endpoint C
[Interface]
PrivateKey = $PKC
Address = $AddressCwg
ListenPort = $PortC

# remote settings for Endpoint A
[Peer]
PublicKey = $pKA
Endpoint = ${EndpointA}:$PortA
AllowedIPs = $AddressAwg
FINI
```

On a vérifié qu'après exécution on a bien nos fichiers de configurations pour les serveurs qui émettent leur logs à travers le VPN :

[cat wg0-a.conf](#)

```
#Fichier wg0-a.conf
# local settings for Endpoint A
[Interface]
PrivateKey = iD/nJRCNKIVrCjJQh8ay49jaWuo/WF7iXyIGg6Gvwml=
Address = 10.0.0.1/32
ListenPort = 51820

# remote settings for Endpoint B
[Peer]
PublicKey = O1jQuesiC2HEP2Sght/usrjV7KtqF+JLHJ77JhsMPDY=
Endpoint = 172.16.0.95:51820
AllowedIPs = 10.0.0.2/32

# remote settings for Endpoint C
[Peer]
PublicKey = K78Gxh0wHPPpbQxJg8JpbOsS9dRLrG2AqZcCs3W2UR0=
Endpoint = 172.16.0.82:51820
AllowedIPs = 10.0.0.3/32
```

[cat wg0-b.conf](#)

```
#Fichier wg0-b.conf
# local settings for Endpoint B
[Interface]
PrivateKey = WEFEtAFdTwClzY7/AGKW4k0LFqL4Gc4iKIdFYokUa0M=
Address = 10.0.0.2/32
ListenPort = 51820

# remote settings for Endpoint A
[Peer]
PublicKey = JkdGvyuNoJRKGF3Vc4OGgiY59WhROJfCDCwnyVgCmjs=
Endpoint = 10.121.38.35:51820
AllowedIPs = 10.0.0.1/32
```

[cat wg0-c.conf](#)

```
#Fichier wg0-c.conf
# local settings for Endpoint C
[Interface]
PrivateKey = ilMEJjrl1chq2kLfUO9G5f25cJrVbHgK+BgOiHTdUI=
Address = 10.0.0.3/32
ListenPort = 51820

# remote settings for Endpoint A
[Peer]
PublicKey = JkdGvyuNoJRKGF3Vc4OGgiY59WhROJfCDCwnyVgCmjs=
Endpoint = 10.121.38.35:51820
AllowedIPs = 10.0.0.1/32
```

On renomme **wg0-a.conf** et **wg0-b.conf** en **wg0.conf**, et on copie **wg0-b.conf** sur **ap31-test** :

```
mv wg0-a.conf wg0.conf      pour srvlog31
mv wg0-b.conf wg0.conf      pour ap31-test
mv wg0-c.conf wg0.conf      pour ap31-prod
```

On a fait une redirection de ports, c'est-à-dire que tout ce que recois gwsio5 est renvoyé vers **srvlog31** puis on a testé avec « **wg** » sur toutes les serveurs

On a testé en pingant le 10.0.0.1 depuis nos serveurs qui envoient leur logs :

`ping 10.0.0.1`

On s'est occupé ensuite de rediriger les logs de **ap31-test** et **ap31-prod** sur le serveur **srvlog31** :

Pour cela on a utilisé les deux scripts suivant dont l'un sert pour les serveurs émetteurs (**ap31-test** et **ap31-prod**) et l'autre pour le serveur receveur (**srvlog31**) :

Script `journald-rcv.sh` (script pour **srvlog31** qui reçoit les logs)

```
#!/bin/bash
sudo timedatectl set-timezone Europe/Paris
sudo apt-get update
sudo apt-get install -y systemd-journal-remote
sudo systemctl enable --now systemd-journal-remote.socket
sudo cp /lib/systemd/system/systemd-journal-remote.service /etc/systemd/system
sudo sed -i 's/--listen-https=-3/--listen-http=-3/' /etc/systemd/system/systemd-journal-remote.service
[[ -d /var/log/journal/remote ]] || sudo mkdir /var/log/journal/remote
sudo chown systemd-journal-remote /var/log/journal/remote
sudo systemctl daemon-reload
```

Script `journald-snd.sh` (script pour **ap31-prod** et **ap31-test** qui envoient leurs logs)

```
#!/bin/bash
# usage : ./journald-snd.sh 10.0.0.1
#sudo timedatectl set-timezone Europe/Paris
#sudo apt-get update
#sudo apt-get install -y systemd-journal-remote
rpl="s/^# URL=/URL=http://V${1}:19532/" # $1 represente l'adresse du recepateur
sudo sed -i "$rpl" /etc/systemd/journal-upload.conf
sudo systemctl enable --now systemd-journal-upload.service
sudo systemctl restart systemd-journal-upload.service
```

On lance le script pour les serveurs émetteurs avec `bash journald-snd.sh 10.0.0.1`