Compte rendu – Mise en place d’une haute disponibilité

LEDIEU Raphaël

Introduction

Ce projet de mise en place d'une solution de haute disponibilité s'inscrit dans le cadre de ma formation, spécifiquement dans le cadre d'un rattrapage de TP réalisé en autonomie à domicile. L'objectif était de répondre à une demande fictive mais réaliste d'un studio de développement souhaitant proposer à ses clients une offre d'hébergement web fiable et résiliente.

Dans ce contexte, j'ai dû adapter l'environnement technique initialement prévu pour le réseau de la salle de classe à mon propre réseau domestique. Ainsi, l'ensemble de l'infrastructure a été déployé sur des machines virtuelles configurées en réseau ponté avec ma box internet, utilisant la plage d'adresses IP locale 192.168.1.0/24 pour simuler le réseau professionnel demandé.

La mission consistait à concevoir et déployer une infrastructure redondante entre trois serveurs Linux, capable de garantir une continuité de service même en cas de défaillance partielle du système. Pour atteindre cet objectif, j'ai implémenté une solution complète intégrant plusieurs technologies complémentaires : Keepalived pour la gestion de l'IP virtuelle, Lsyncd pour la réplication automatique des données, et HAProxy pour la répartition de charge entre les différents nœuds.

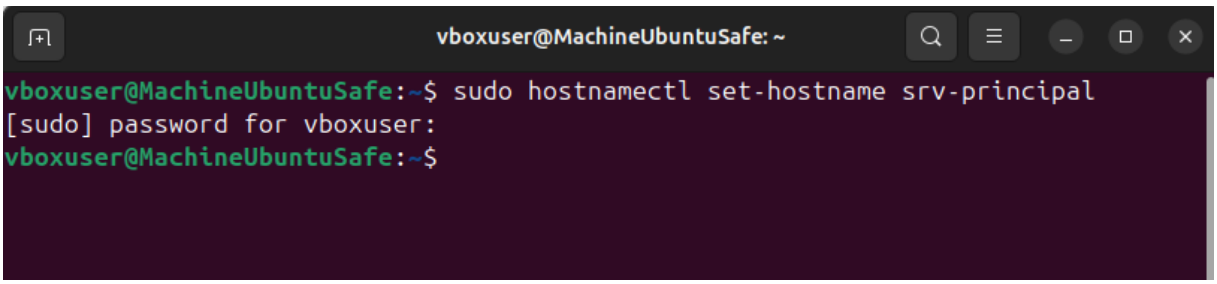
Ce compte-rendu présente de manière détaillée l'ensemble du processus de déploiement, depuis la configuration initiale des serveurs jusqu'aux tests de validation finale, en passant par la mise en place des différents mécanismes de haute disponibilité. Il documente également les adaptations nécessaires pour transposer le TP dans un environnement domestique tout en respectant les exigences pédagogiques et techniques du cahier des charges.

Pour commencer, nous avons créé une machine virtuelle sous Ubuntu Server, qui servira de serveur principal dans notre architecture de haute disponibilité.
Cette machine jouera un rôle central puisqu’elle hébergera le service web et sera surveillée par les serveurs secondaires en cas de panne.

Après l’installation du système, nous avons modifié le nom d’hôte (hostname) afin d’identifier facilement la machine sur le réseau et dans le cluster.

La commande utilisée est :

« sudo hostnamectl set-hostname srv-principal »

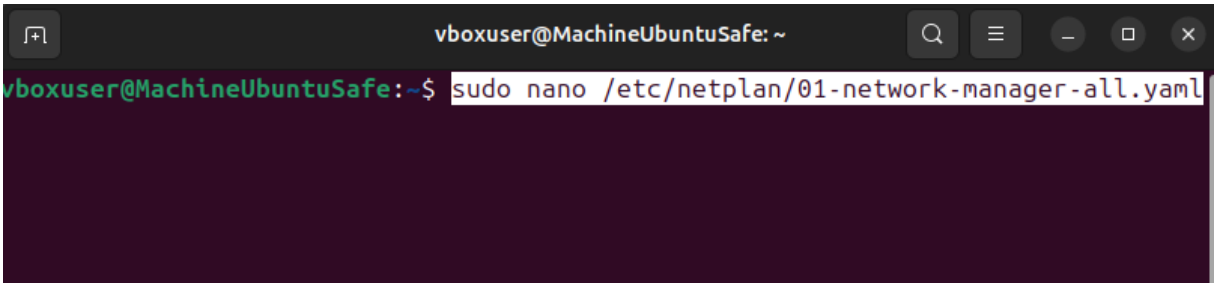


Ce nom permettra de distinguer clairement le serveur principal lors des tests de bascule.

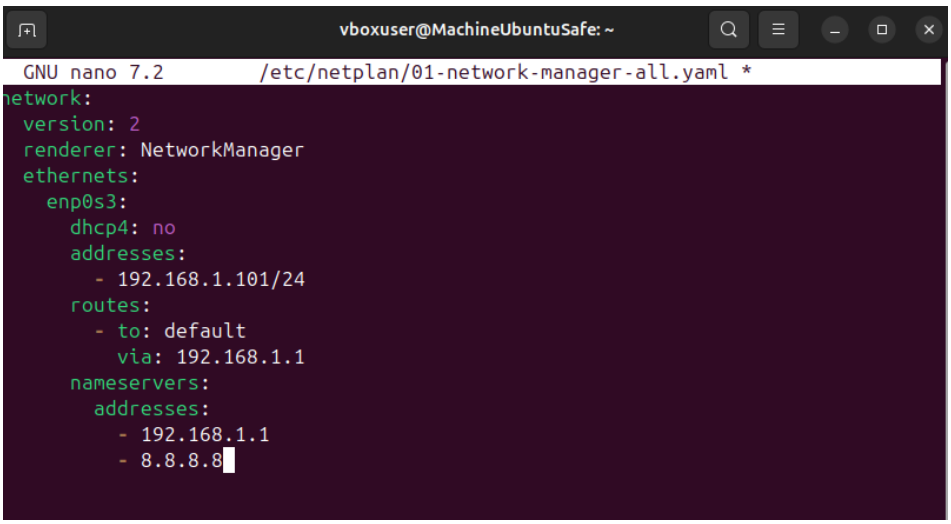
Pour assurer une communication fiable entre les serveurs, il est nécessaire que chacun dispose d'une adresse IP fixe.

Nous avons donc modifié la configuration réseau en éditant le fichier suivant :

« sudo nano /etc/netplan/01-network-manager-all.yaml »



Puis nous avons remplacé son contenu par cette configuration :



Cette configuration permet de définir manuellement les paramètres réseau du serveur principal afin qu'il dispose toujours de la même adresse IP.
Voici le rôle de chaque ligne du fichier :

version: 2 → indique la version du format de configuration Netplan utilisée (ici la version 2, la plus courante sur Ubuntu).

renderer: NetworkManager → précise que la gestion du réseau est confiée à **NetworkManager**, un service qui s'occupe d'activer et de superviser les interfaces réseau.

ethernets: → section où l'on définit la configuration des cartes réseau de la machine.

enp0s3: → nom de l'interface réseau utilisée (cela peut varier selon les machines).

dhcp4: no → désactive le DHCP, c'est-à-dire que la machine n'obtient plus automatiquement son adresse IP ; elle utilisera désormais une adresse fixe.

addresses: → définit l'adresse IP statique de la machine.

Ici : 192.168.1.101/24, ce qui correspond à l'adresse du serveur principal sur le réseau local.

routes: → permet de définir la **passerelle par défaut**, c'est-à-dire le chemin à suivre pour accéder à l'extérieur du réseau local.

via: 192.168.1.1 indique que la passerelle est l'adresse 192.168.1.1.

nameservers: → définit les **serveurs DNS** utilisés pour la résolution des noms de domaine.

192.168.1.1 est le DNS local (souvent celui de la box ou du routeur).

8.8.8.8 est le DNS public de Google, utilisé en secours.

Grâce à cette configuration, la machine possède désormais une adresse IP fixe, peut accéder à Internet, et résoudre les noms de domaine, ce qui est indispensable pour la suite de la mise en place du cluster de haute disponibilité.

Une fois le fichier enregistré, nous avons appliqué les changements avec la commande :

« sudo netplan apply »

Le système a affiché une erreur liée aux permissions du fichier /etc/netplan/01-network-manager-all.yaml. Cela signifie que les droits d'accès au fichier n'étaient pas corrects, empêchant Netplan de lire ou d'appliquer la configuration réseau.

Pour corriger ce problème, j'ai exécuté la commande suivante :

« sudo chmod 600 /etc/netplan/01-network-manager-all.yaml »

Cette commande modifie les permissions du fichier afin que seul l'utilisateur root (l'administrateur du système) ait le droit de le lire et de le modifier.

Le chiffre 6 correspond aux droits de lecture et d'écriture pour le propriétaire (root).

Le 0 empêche les autres utilisateurs ou groupes d'y accéder.

Une fois les permissions corrigées, j'ai relancé la commande :

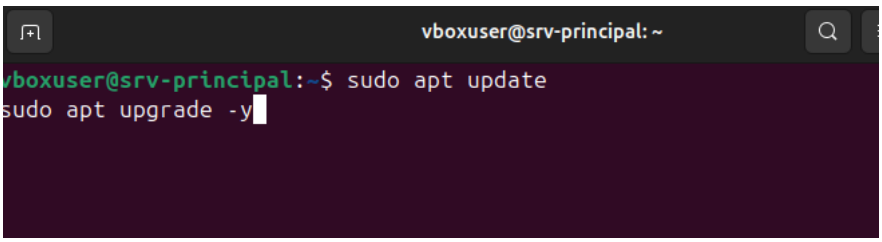
« sudo netplan apply »

Cette fois-ci, la configuration réseau s'est appliquée correctement, permettant au serveur d'obtenir son adresse IP fixe et de se connecter au réseau.

Nous avons procédé à l'installation du serveur web Apache sur la machine principale. Cette étape est essentielle, car Apache assurera l'hébergement du site web au sein du cluster de haute disponibilité. Avant toute installation, il est important de s'assurer que le système est à jour afin d'éviter les problèmes de compatibilité ou de dépendances.
Les commandes utilisées sont :

« sudo apt update »

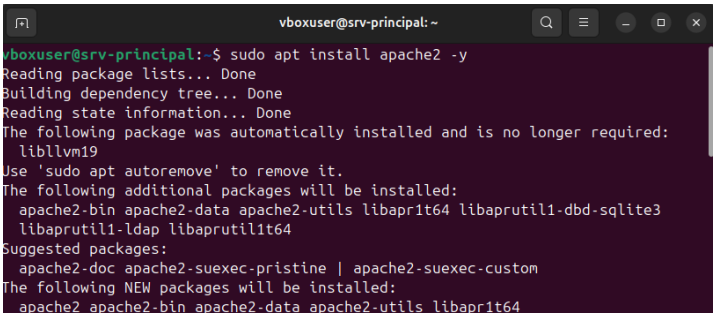
« sudo apt upgrade -y »



La première commande met à jour la liste des paquets disponibles, tandis que la seconde installe automatiquement les dernières versions des paquets déjà présents sur le système.

Une fois le système à jour, nous avons installé le service web Apache à l'aide de la commande suivante :

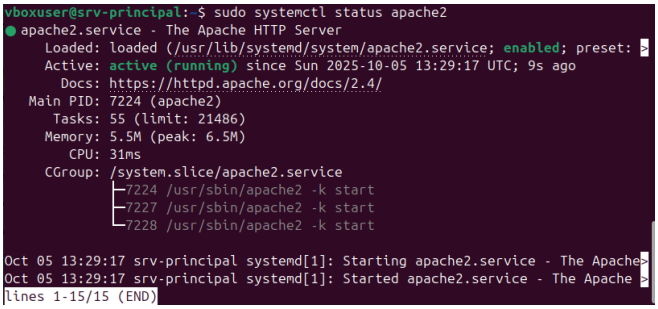
« sudo apt install apache2 -y »



L'option -y permet de valider automatiquement l'installation sans demande de confirmation. Une fois installé, le service Apache se lance automatiquement.

Pour vérifier qu'Apache est bien actif, nous avons utilisé la commande suivante :

« sudo systemctl status apache2 »

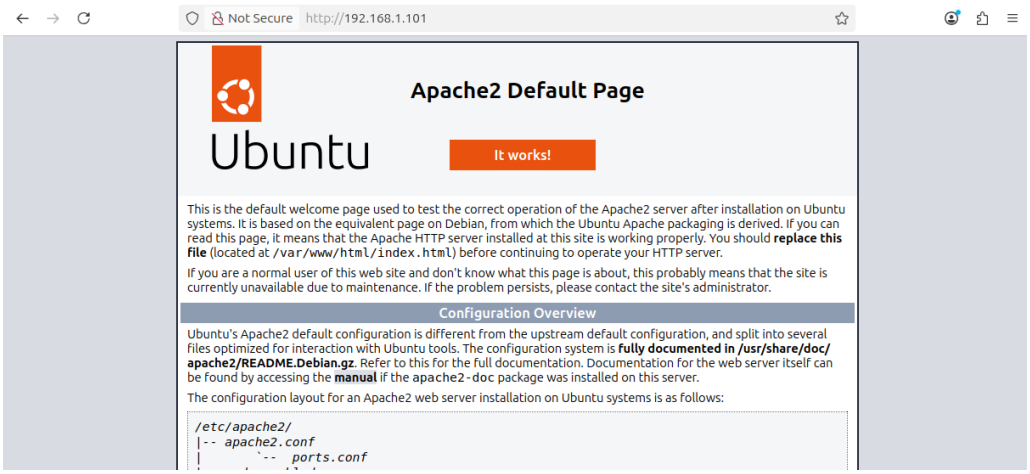


Le statut active (running) en vert confirme que le service est bien démarré et fonctionnel.

Afin de confirmer que le serveur web répond correctement aux requêtes HTTP, nous avons ouvert un navigateur sur le poste hôte et saisi l'adresse suivante :

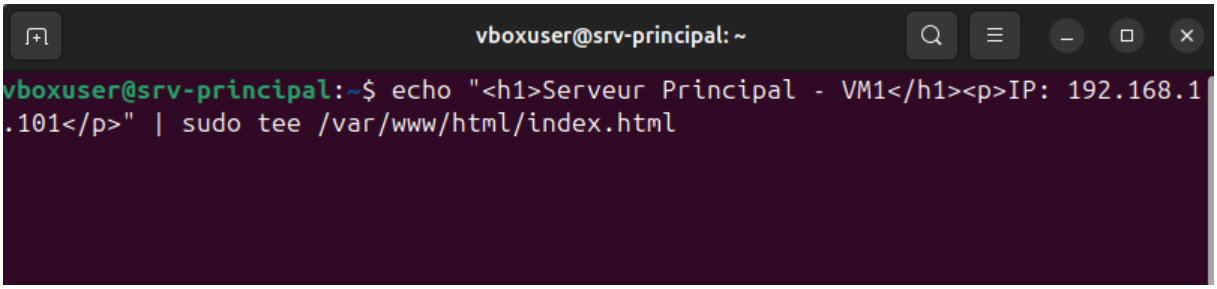
« http://192.168.1.101 »

La page par défaut d'Apache (Apache2 Ubuntu Default Page) s'est bien affichée, prouvant que le serveur est accessible sur le réseau.



Pour distinguer plus facilement ce serveur des autres, nous avons remplacé la page par défaut par une page personnalisée à l'aide de la commande suivante :

« echo "<h1>Serveur Principal - VM1</h1> IP: 192.168.1.101 " | sudo tee /var/www/html/index.html »



Après avoir actualisé la page dans le navigateur, le message personnalisé s'est affiché correctement.

Cela confirme que le serveur web Apache fonctionne parfaitement et que les fichiers hébergés dans le répertoire /var/www/html sont bien servis aux clients.

Après avoir configuré et testé le serveur principal, nous avons procédé à la création des deux serveurs secondaires. Ces derniers serviront de nœuds de secours dans notre configuration de haute disponibilité.

L'objectif de cette étape est d'obtenir trois machines identiques au niveau du système et des services installés, tout en leur attribuant une adresse IP et un nom d'hôte distincts.

Avant le clonage, il est impératif d'éteindre la machine principale pour cloner

Cela arrête le serveur proprement en sauvegardant toutes les modifications système.

Une fois la machine éteinte, nous avons effectué les clonages directement depuis VirtualBox.

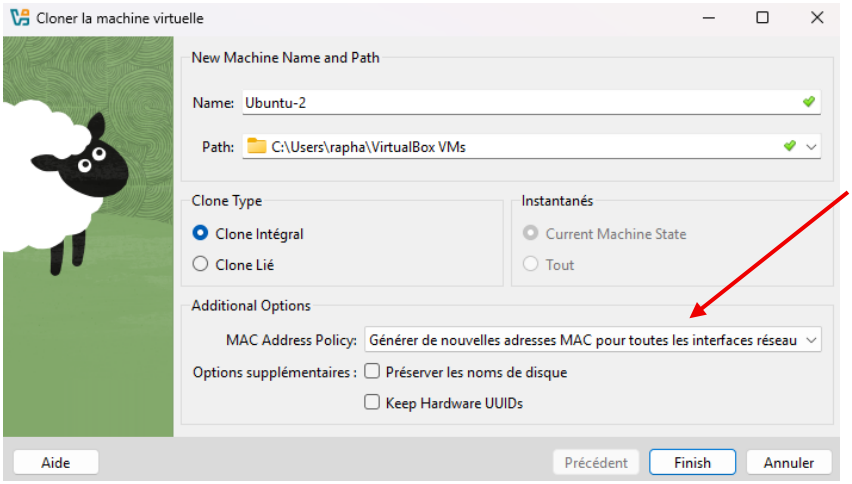
Création du premier serveur secondaire (VM2)

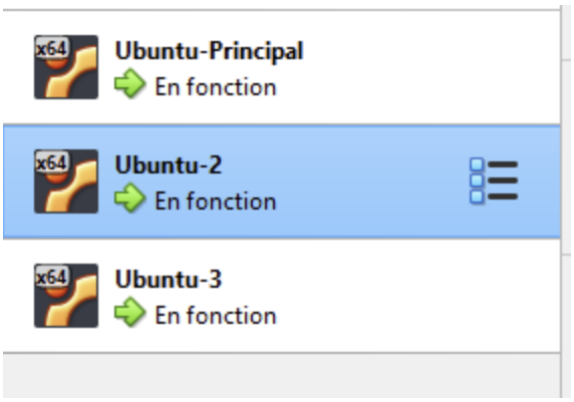
Clic droit sur la machine principale → Cloner

Nom : ubuntu-2

Sélection : Clone complet

Option activée : Générer de nouvelles adresses MAC pour toutes les cartes réseau (étape très importante pour éviter des conflits d'adresse sur le réseau).





Création du second serveur secondaire (VM3)

Même procédure que précédemment

Nom : ubuntu-3

Type : Clone complet

Là encore, il est essentiel de générer une nouvelle adresse MAC afin que chaque machine virtuelle possède une identité réseau unique.

Nous avons désormais trois machines prêtes à être configurées :

srv-principal (192.168.1.101)

ubuntu-2 (futur srv-secondaire-1, 192.168.1.102)

ubuntu-3 (futur srv-secondaire-2, 192.168.1.103)

Ces deux clones seront ensuite démarrés pour modifier leur hostname, leur adresse IP statique et leur page web, exactement de la même manière que pour la VM principale. Cette uniformité de configuration permet de les identifier clairement lors des tests de bascule et de réplication, tout en garantissant que chaque serveur est correctement intégré dans le cluster de haute disponibilité.

Avant de configurer le cluster, il est important de s'assurer que toutes les machines fonctionnent correctement et que les services web sont accessibles.

Nous avons donc testé chaque serveur depuis un navigateur sur le poste hôte :

http://192.168.1.101 → affiche : Serveur Principal - VM1

http://192.168.1.102 → affiche : Serveur Secondaire 1 - VM2

http://192.168.1.103 → affiche : Serveur Secondaire 2 - VM3

Ces tests confirment que :

Chaque VM dispose de la bonne adresse IP statique. Les pages web personnalisées sont présentes et accessibles. La connectivité réseau entre les serveurs et le poste hôte est correcte.

Toutes les conditions préalables sont donc remplies, et nous pouvons maintenant passer à la mise en place de la haute disponibilité.

Pour assurer la continuité du service web, nous allons mettre en place une IP:

« 192.168.1.100 »

Cette IP sera utilisée par les utilisateurs pour accéder au site, indépendamment du serveur physique qui la porte.

Initialement, l'IP **(192.168.1.100)** sera assignée au **serveur principal (VM1)**.

Si VM1 tombe en panne, l'IP **(192.168.1.100)** sera automatiquement **basculée sur VM2**.

Si VM2 tombe également, VM3 prendra le relais.

Ainsi, les utilisateurs accéderont toujours au site via :

« http://192.168.1.100 »

sans remarquer la panne d'un serveur.

Installation de Keepalived

Keepalived est l'outil qui permet de gérer cette IP spéciale et de définir les priorités entre les serveurs. Sur chacune des trois VM, nous avons exécuté les commandes suivantes :

« sudo apt update »

« sudo apt install keepalived -y »

```
vboxuser@srv-principal:~$ sudo apt install keepalived -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  ipvsadm
Suggested packages:
  heartbeat ldirectord
The following NEW packages will be installed:
  ipvsadm keepalived
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 500 kB of archives.
```

Cette installation doit être effectuée sur tous les nœuds du cluster pour que le mécanisme de bascule fonctionne correctement.

Après l'installation, la configuration de Keepalived sera différente sur chaque VM :

VM1 (serveur principal) aura la priorité la plus élevée.

VM2 aura une priorité intermédiaire.

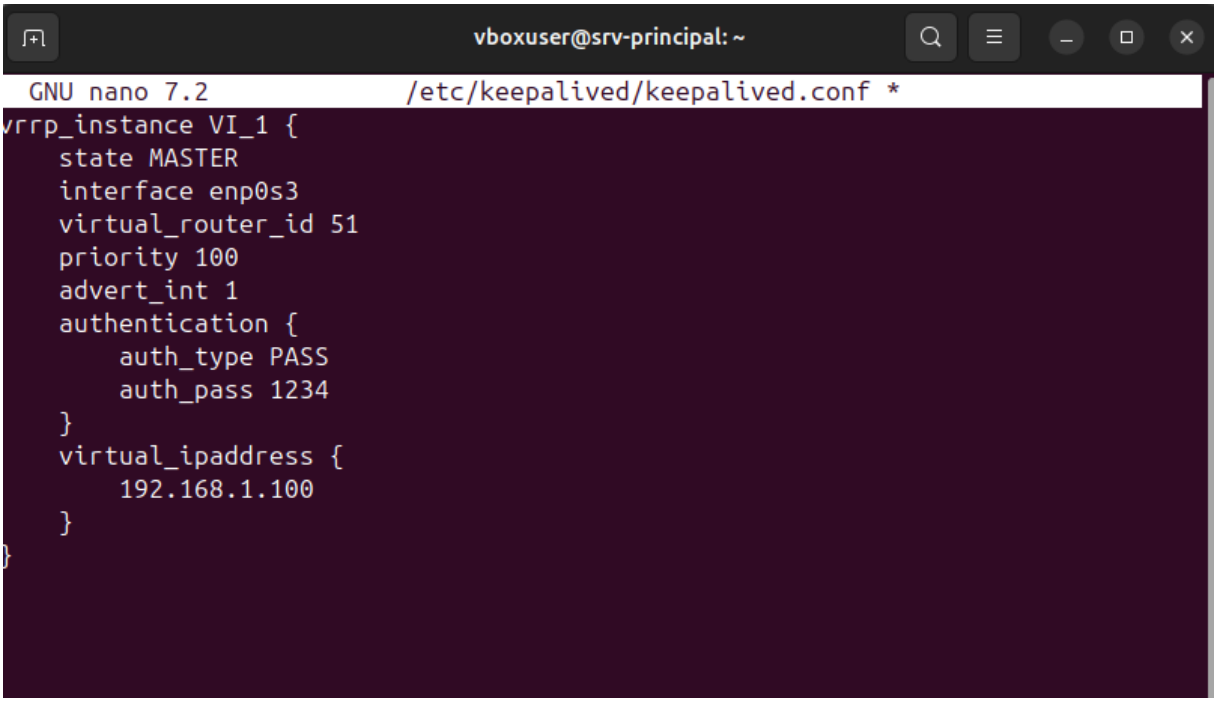
VM3 aura la priorité la plus basse.

Ces priorités permettent de déterminer quel serveur prend le contrôle de l'IP spéciale en cas de panne, assurant ainsi la haute disponibilité du service web.

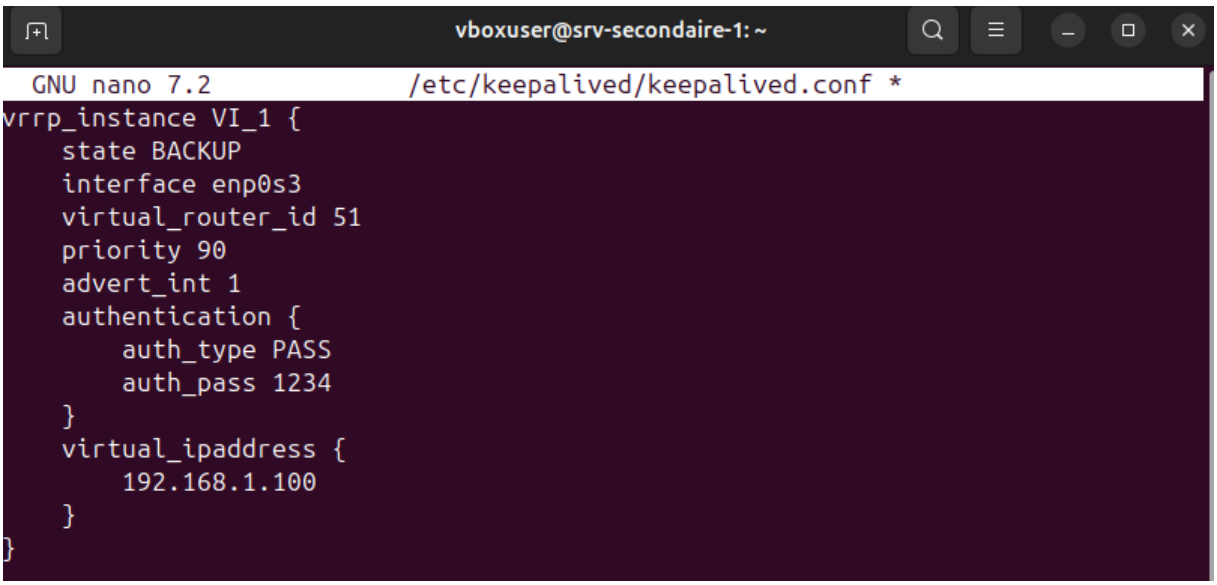
Pour mettre en place la haute disponibilité, il est nécessaire de configurer Keepalived différemment sur chaque VM afin de définir les priorités entre les serveurs.

Priorités attribuées

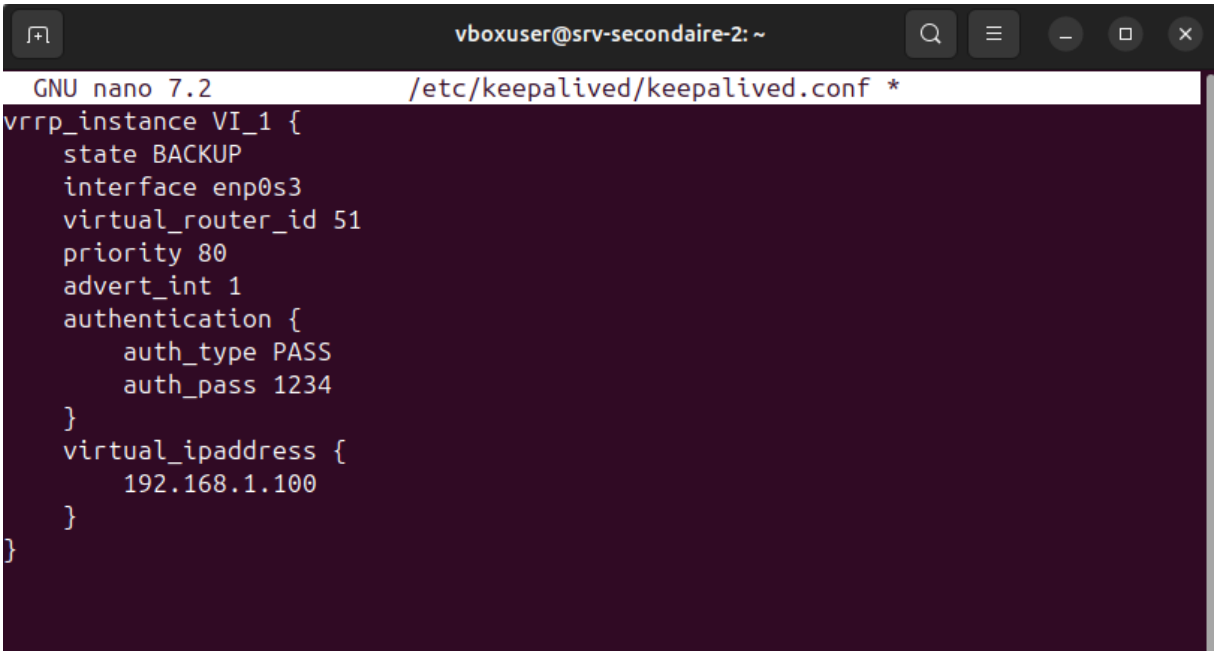
VM1 (serveur principal) : priorité 100



VM2 (serveur secondaire 1) : priorité 90



VM3 (serveur secondaire 2) : priorité 80



Le serveur avec la priorité la plus élevée portera l'IP spéciale en premier. Si ce serveur tombe en panne, le nœud suivant dans l'ordre de priorité prendra automatiquement le relais.

Nous avons édité le fichier de configuration de Keepalived sur chaque machine.

Les paramètres principaux à configurer pour chaque VM sont :

L'état du serveur (MASTER pour la VM1, BACKUP pour les secondaires),

L'interface réseau utilisée,

La priorité,

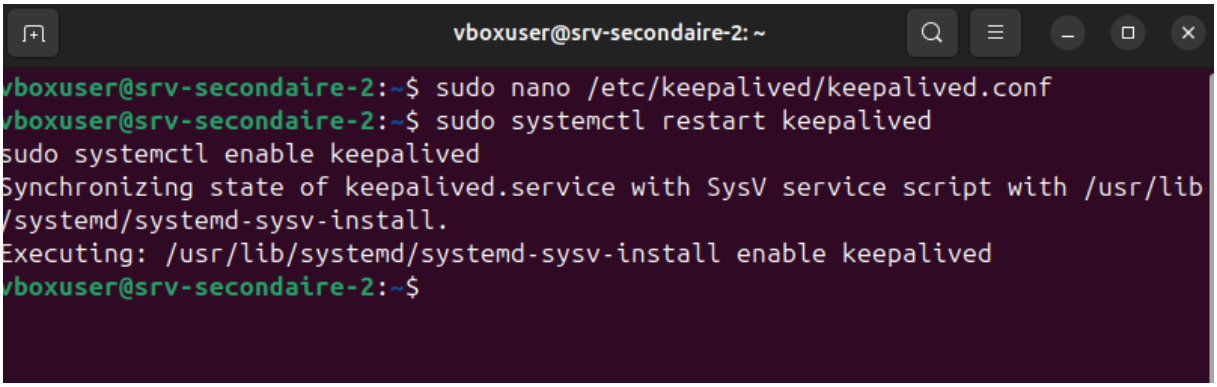
L'adresse de l'IP spéciale,

Et l'authentification VRRP pour sécuriser la communication entre les nœuds.

Après modification, le service Keepalived a été redémarré et activé pour démarrer automatiquement au boot avec :

« sudo systemctl restart keepalived »

« sudo systemctl enable keepalived »

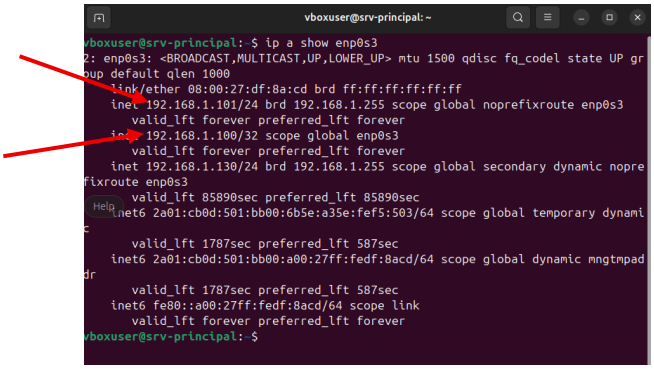


Grâce à cette configuration, l'IP **(192.168.1.100)** sera automatiquement transférée d'un serveur à l'autre en cas de panne, garantissant la continuité du service web.

Pour s'assurer que la configuration de Keepalived fonctionne correctement, nous avons vérifié que l'IP était bien assignée au serveur principal (VM1).

Sur VM1, nous avons exécuté la commande :

« ip a show enp0s3 »



Nous avons observé deux adresses IP sur la même interface :

- **192.168.1.101** → l'adresse IP statique de la VM1
- **192.168.1.100** → l'IP qui est celle utilisée par les utilisateurs pour accéder au service web

Cette vérification confirme que le serveur principal porte correctement l'IP spéciale et que le mécanisme de haute disponibilité est opérationnel.

Tests de la haute disponibilité

Après avoir configuré Keepalived sur les trois VM, nous avons effectué plusieurs tests pour vérifier que l'IP spéciale bascule correctement entre les serveurs en cas de panne.

Test 2 : Accès via l'IP spéciale

Depuis un navigateur sur le poste hôte, nous avons saisi l'adresse :

http://192.168.1.100

La page affichée correspondait bien à VM1 (Serveur Principal), ce qui confirme que l'IP est correctement assignée au serveur principal et que les utilisateurs peuvent accéder au service web sans connaître l'adresse IP réelle du serveur.

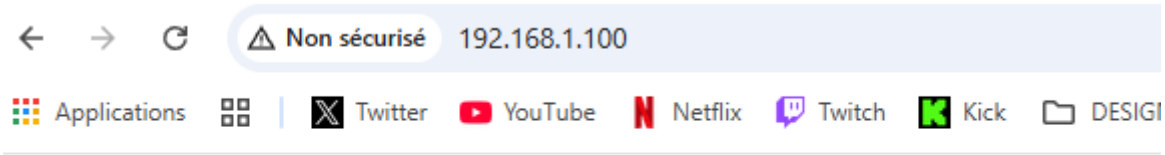
Simulation de la panne du serveur principal

Pour tester la bascule automatique, nous avons éteint VM1

Après 5 à 10 secondes, nous avons actualisé le navigateur sur http://192.168.1.100 et avons constaté que la page affichée correspondait à VM2 (Serveur Secondaire 1).
Cela démontre que Keepalived a automatiquement transféré l'IP spéciale vers le serveur secondaire avec la priorité suivante.

Panne du secondaire 1

Nous avons ensuite éteint VM2 également et actualisé à nouveau la page sur http://192.168.1.100.



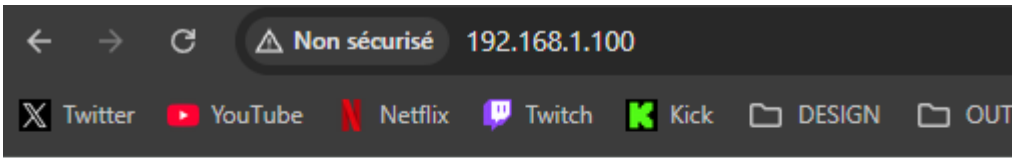
Serveur Secondaire 2 - VM3

IP: 192.168.1.103

La page affichée correspondait alors à VM3 (Serveur Secondaire 2).
Ce test confirme que l'IP spéciale continue de basculer correctement même si plusieurs serveurs tombent en panne.

Retour du serveur principal

Enfin, nous avons rallumé VM1. Après son démarrage, nous avons actualisé le navigateur : l'IP spéciale est revenue automatiquement sur VM1, le serveur principal ayant la priorité la plus élevée.



Serveur Principal - VM1

IP: 192.168.1.101

Ces tests confirment que la haute disponibilité fonctionne parfaitement, assurant la continuité du service web quel que soit l'état des serveurs physiques.

Réplication des données

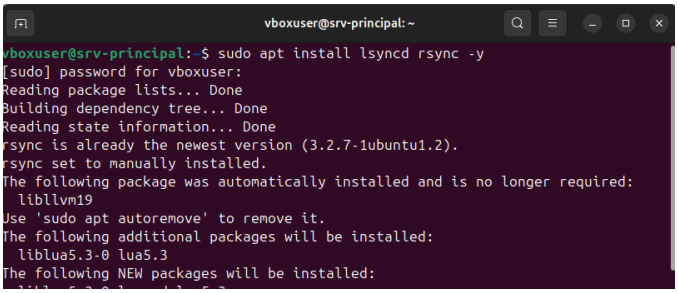
Pour garantir que les fichiers web soient synchronisés automatiquement entre les trois serveurs, nous avons mis en place une réplication avec lsyncd.

L'objectif est que toute modification dans **« /var/www/html/ »** sur VM1 se réplique automatiquement sur VM2 et VM3. Lsyncd utilise rsync et inotify pour effectuer cette synchronisation en temps réel.

Installation de lsyncd et rsync

Sur chaque VM, nous avons installé les paquets nécessaires :

« sudo apt install lsyncd rsync -y »

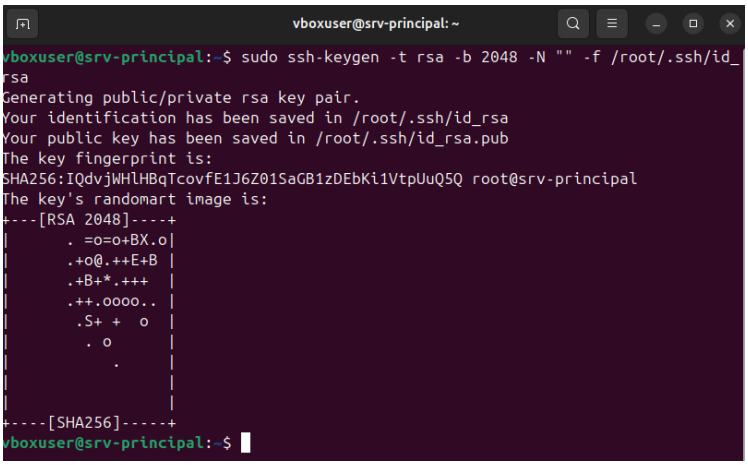


Configuration des clés SSH pour la synchronisation sans mot de passe

Pour que lsyncd puisse copier les fichiers automatiquement, nous devons configurer l'authentification par clé SSH depuis VM1 vers VM2 et VM3.

Sur VM1, génération d'une clé SSH :

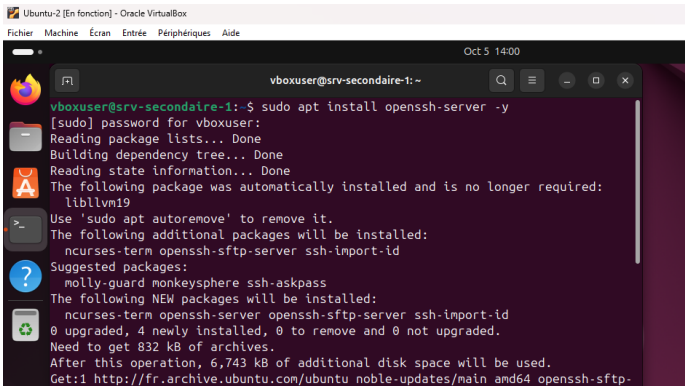
« sudo ssh-keygen -t rsa -b 2048 -N "" -f /root/.ssh/id_rsa »



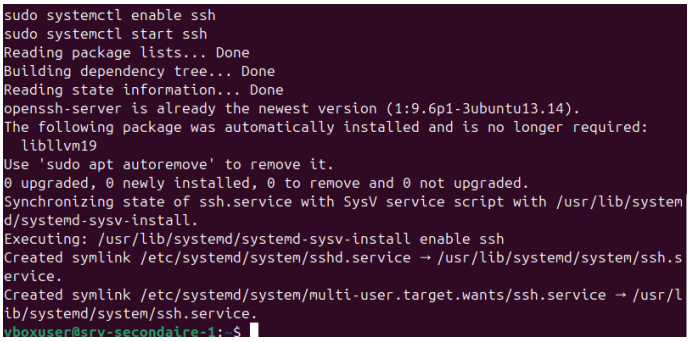
Activation du serveur SSH sur VM2 et VM3

Comme SSH n'était pas actif par défaut, nous avons installé et activé OpenSSH sur les deux serveurs secondaires :

« sudo apt install openssh-server -y »



« sudo systemctl enable ssh »

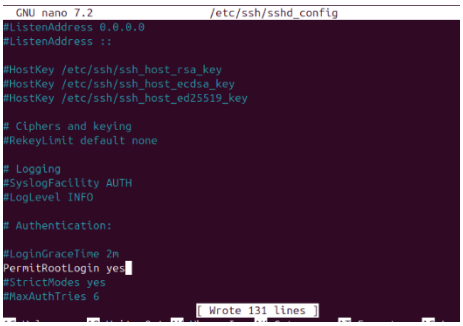


« sudo systemctl start ssh »

Autoriser temporairement le login root

Sur VM2 et VM3, nous avons modifié le fichier /etc/ssh/sshd_config pour permettre la connexion root :

« PermitRootLogin yes »



Puis nous avons défini un mot de passe root temporaire et redémarré le service SSH :

« sudo passwd root »

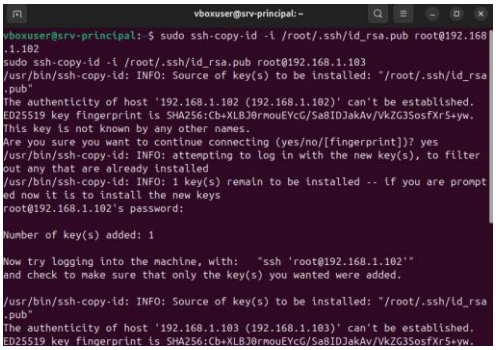
« sudo systemctl restart ssh »

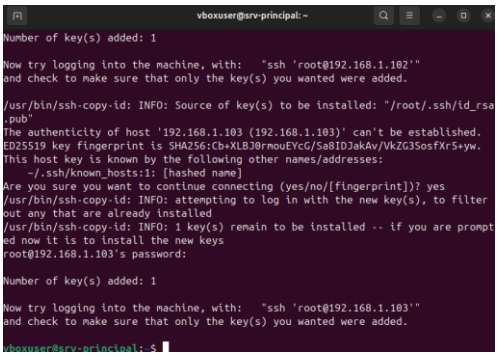
Copie des clés SSH depuis VM1

Depuis VM1, nous avons copié la clé vers VM2 et VM3 :

« sudo ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.1.102 »

« sudo ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.1.103 »





Lors de la première connexion, nous avons accepté la clé de l'hôte (yes) et saisi le mot de passe root pour chaque serveur.

La configuration est maintenant opérationnelle : VM1 peut se connecter à VM2 et VM3 sans mot de passe, ce qui permettra à lsyncd de répliquer automatiquement les fichiers web entre les trois serveurs.

Pour assurer la réplication automatique des fichiers web, nous avons configuré lsyncd sur VM1 afin de synchroniser le dossier **« /var/www/html/ »** vers les deux serveurs secondaires (VM2 et VM3).

Configuration de lsyncd

Sur VM1, nous avons créé le fichier de configuration :

« sudo nano /etc/lsyncd/lsyncd.conf.lua »

Dans ce fichier, nous avons défini :

L'emplacement des fichiers de logs et du status de lsyncd,

La fréquence de mise à jour du status (statusInterval),

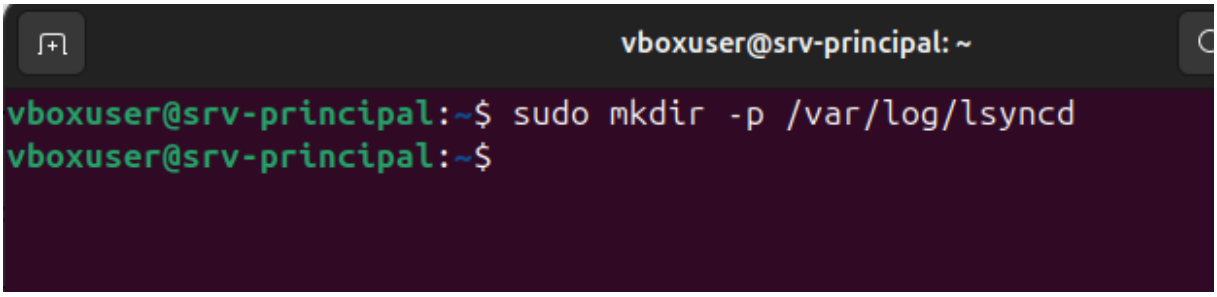
La source (/var/www/html/) et les cibles (VM2 et VM3) pour la synchronisation,

Les options archive et compress pour rsync, et la suppression automatique des fichiers obsolètes (delete = "running").

Création du dossier de logs

Nous avons créé le dossier pour stocker les logs de lsyncd :

« sudo mkdir -p /var/log/lsyncd »



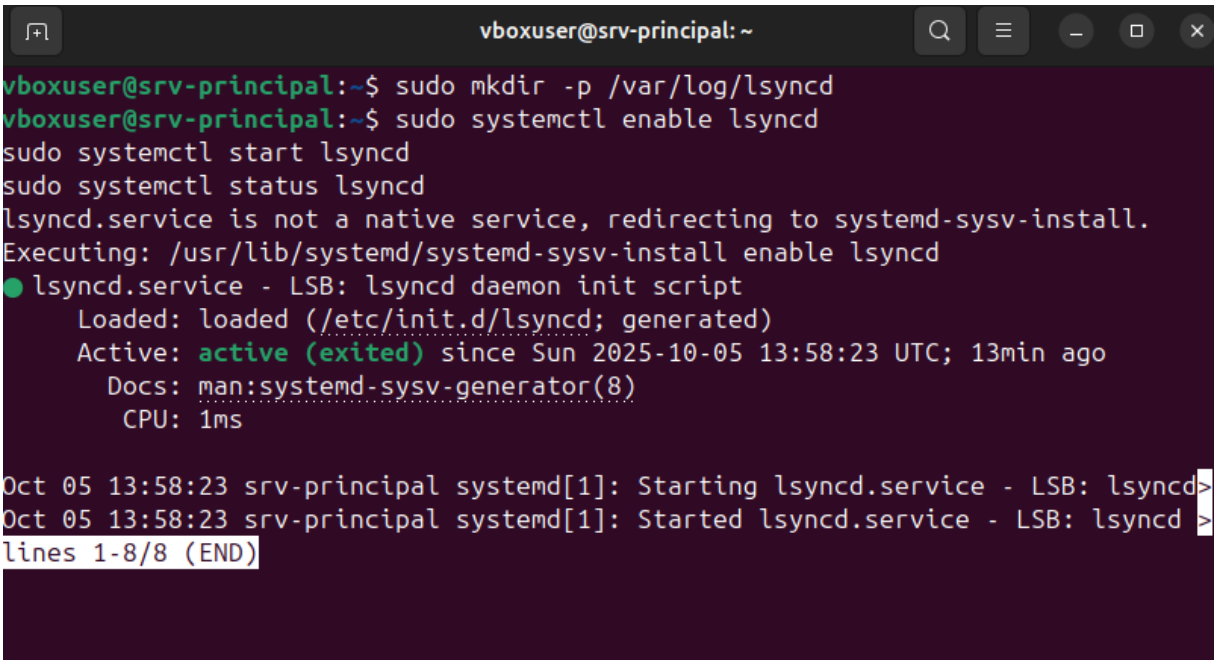
Démarrage et activation du service

Enfin, nous avons activé et démarré le service lsyncd pour qu'il fonctionne automatiquement au démarrage de la VM :

« sudo systemctl enable lsyncd »

« sudo systemctl start lsyncd »

« sudo systemctl status lsyncd »



Grâce à cette configuration :

Toute modification dans **« /var/www/html/ »** sur VM1 est répliquée automatiquement sur VM2 et VM3.

Le service est actif en permanence et les logs permettent de suivre les synchronisations.

Après avoir configuré et démarré lsyncd, il est important de vérifier que la synchronisation fonctionne correctement et qu'aucune erreur n'apparaît.

Pour cela, nous avons consulté le fichier de logs d'lsyncd sur VM1 :

« sudo cat /var/log/lsyncd/lsyncd.log »

```
vboxuser@srv-principal:~$ sudo cat /var/log/lsyncd/lsyncd.log
Sun Oct  5 14:12:22 2025 Normal: --- Startup, daemonizing ---
Sun Oct  5 14:12:22 2025 Normal: recursive startup rsync: /var/www/html/ -> root
@192.168.1.102:/var/www/html/
Sun Oct  5 14:12:22 2025 Normal: recursive startup rsync: /var/www/html/ -> root
@192.168.1.103:/var/www/html/
Sun Oct  5 14:12:23 2025 Normal: Startup of /var/www/html/ -> root@192.168.1.103
:/var/www/html/ finished.
Sun Oct  5 14:12:25 2025 Normal: Startup of /var/www/html/ -> root@192.168.1.102
:/var/www/html/ finished.
vboxuser@srv-principal:~$
```

Ce fichier contient toutes les informations sur les opérations de synchronisation effectuées vers VM2 et VM3.

Il permet d'identifier rapidement si des fichiers n'ont pas pu être copiés ou si une erreur de connexion SSH s'est produite.

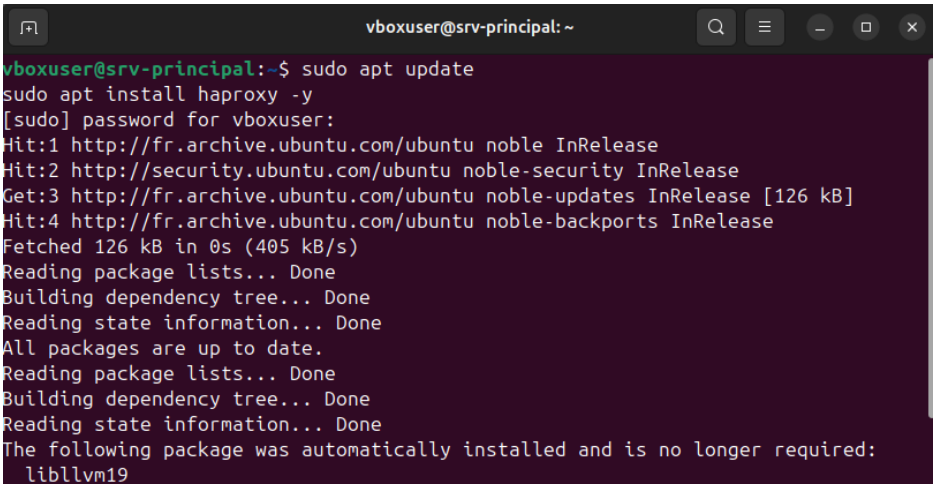
Pour assurer la répartition de charge entre les trois serveurs web, nous avons installé et configuré HAProxy sur la machine principale.

Installation de HAProxy

Sur la machine où HAProxy sera installé, nous avons exécuté :

« sudo apt update »

« sudo apt install haproxy -y »

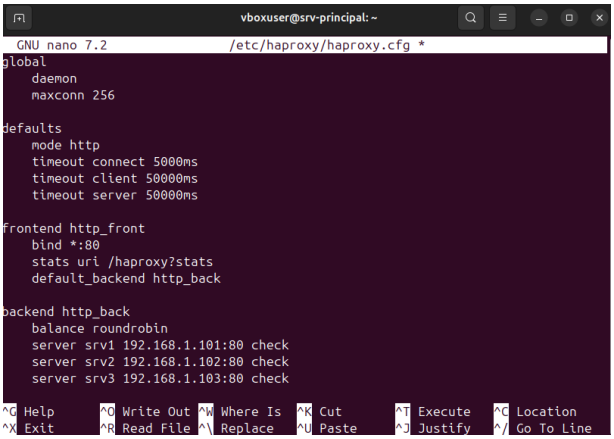


Configuration pour le port 80

Nous avons modifié le fichier de configuration **« /etc/haproxy/haproxy.cfg »** afin de gérer le trafic HTTP sur le port 80 :

La section frontend écoute sur toutes les interfaces (*:80) et redirige le trafic vers le backend http_back.

La section backend inclut les trois serveurs web (VM1, VM2, VM3) et utilise l'algorithme roundrobin pour répartir les requêtes équitablement.



Arrêt d'Apache sur VM1

Pour éviter les conflits de port sur le serveur principal, nous avons stoppé et désactivé Apache :

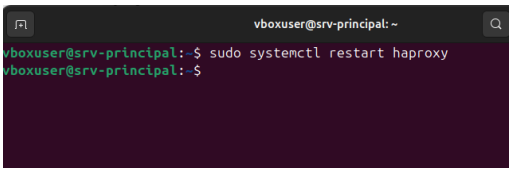
« sudo systemctl stop apache2 »

« sudo systemctl disable apache2 »

Redémarrage d'HAProxy

Après modification, on redémarre le service haproxy avec la commande :

« sudo systemctl restart haproxy »



Test de l'IP avec load balancing

Depuis un navigateur, nous avons accédé à :

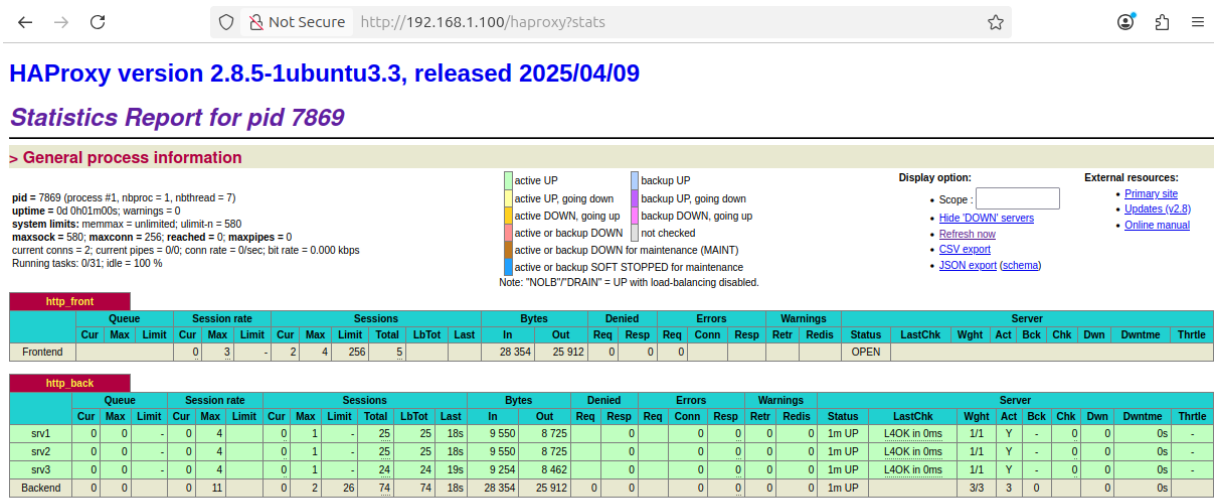
http://192.168.1.100

En actualisant plusieurs fois la page, nous avons pu constater que les trois serveurs tournaient successivement, confirmant le bon fonctionnement de la répartition de charge.

Vérification des statistiques

Pour contrôler le fonctionnement interne de HAProxy, nous avons consulté la page des statistiques :

« http://192.168.1.100/haproxy?stats »



Cette interface permet de visualiser l'état des serveurs, le nombre de requêtes et les éventuels problèmes de connexion.

Partie juridique : CGV et maintien du service

Dans le cadre de l'offre d'hébergement, les Conditions Générales de Vente (CGV) doivent définir précisément les engagements de l'entreprise et la protection du client.

Maintien du service et engagement de disponibilité (SLA) :
L'entreprise s'engage à assurer la continuité de l'accès aux sites hébergés grâce à une infrastructure de haute disponibilité comprenant Keepalived, Lsyncd et HAProxy. Cette configuration permet de garantir un service stable même en cas de panne d'un serveur ou d'un incident technique.
L'entreprise garantit une disponibilité minimale de 99,5% par mois, soit un maximum de 3h36 d'indisponibilité. Au-delà de cette limite, le client bénéficie d'un crédit de 25% du montant mensuel de l'hébergement.

Pénalités en cas d'interruption :
Les pénalités sont calculées en fonction de la durée réelle d'indisponibilité. L'entreprise met à disposition un processus clair pour le client afin de demander une compensation :

Le client signale toute interruption de service via le support.

L'entreprise mesure la durée exacte de l'interruption à partir des logs du service et de l'IP spéciale (Keepalived et HAProxy).

Si la durée dépasse le seuil défini dans le SLA, le crédit ou remboursement est automatiquement appliqué sur la facture suivante.

Responsabilités et limites :
Les CGV précisent les limites de responsabilité de l'entreprise en cas de panne indépendante de sa volonté. Cela inclut notamment :

Les catastrophes naturelles (inondations, tempêtes, incendies),

Les cyberattaques ciblant le datacenter ou l'infrastructure réseau,

Les pannes majeures du datacenter ou du fournisseur d'énergie.

Conclusion

Ce TP m'a permis de mettre en place une infrastructure de haute disponibilité complète et fonctionnelle avec trois serveurs web, en utilisant Keepalived pour le failover automatique, Lsyncd pour la réplication des données et HAProxy pour la répartition de charge. Même si j'ai rencontré quelques difficultés techniques, notamment lors de la configuration SSH et de la gestion du réseau en mode pont, j'ai réussi à les résoudre et à comprendre comment fonctionne réellement un système de haute disponibilité. Ce projet était vraiment intéressant et sympa à réaliser car il correspond à un cas d'usage concret en entreprise. J'ai particulièrement apprécié de voir l'infrastructure basculer automatiquement d'un serveur à l'autre lors des tests de panne. Les compétences acquises sur ces technologies seront très utiles dans un contexte professionnel où la continuité de service est primordiale.