



Mise en place d'une Infrastructure de Virtualisation avec Proxmox

07/01/25

Groupe :

Delcourt Louis

Ifrim Vasile-Alexandru

Enseignant référent :

Djelloul Ziad

1 - Sommaire

1. Sommaire.....	1
2. Introduction.....	2
2.1 Contexte et objectif du projet.....	2
3. Mise en place de la solution.....	4
3.1 Les ressources que nous avons utilisées.....	4
3.2 L'architecture de notre infrastructure.....	4
3.3 Installation des serveurs virtuels.....	6
3.4 Automatisation des installations	7
3.4 Configuration manuelle	23

2 - Introduction

2.1 Contexte et objectif du projet

L'objectif est de mettre en place une infrastructure de virtualisation basée sur proxmox de la façon la plus automatisée et simple possible.

Une **infrastructure de virtualisation** est un ensemble de technologies, de matériels, de logiciels et de processus qui permettent de créer et de gérer des environnements virtuels (VMs), sur des ressources physiques partagées. Elle vise à optimiser l'utilisation des ressources matérielles (serveurs, stockage, réseau) en les abstrayant pour fournir des services virtualisés.

Une infrastructure de virtualisation est composé :

- D'un hyperviseur, qui est un logiciel qui permet de créer et gérer des machines virtuelles. Il agit comme une couche d'abstraction entre le matériel physique et les environnements virtuels.
- Des serveurs physiques, afin de fournir les ressources matériels (CPU, RAM, stockage...) utilisés pour les VMs.
- De stockage, souvent les infrastructures de virtualisation utilisent des systèmes de stockage partagés ou locaux.
- Machine virtuelle, des environnements informatiques virtuels qui fonctionnent comme des ordinateurs indépendants avec leurs propres systèmes d'exploitation et applications.
- Réseau virtuel, en effet les infrastructures de virtualisation incluent souvent des réseaux virtuels (vLAN, vSwitches) qui permettent de connecter les machines virtuelles entre elles et avec le réseau physique.
- Logiciels de gestion avec des outils pour centraliser la gestion des ressources virtualisées permettent d'automatiser, de surveiller et de maintenir les environnements virtualisés.

Qu'est-ce que proxmox ?

Proxmox est une solution de virtualisation open-source qui permet de gérer des machines virtuelles et des conteneurs de manière centralisée. Il est largement utilisé dans les environnements de production et les laboratoires pour simplifier la gestion des infrastructures virtualisées grâce à une interface web conviviale, des fonctionnalités avancées et une prise en charge des technologies modernes de virtualisation.

L'hyperviseur de proxmox est basé sur KVM, ce qui garantit une excellente performance et une gestion optimisée des ressources. Kvm est un hyperviseur de type 1 qui fonctionne directement sur le matériel de la machine physique.

En plus de la virtualisation, proxmox prend aussi en charge les conteneurs lxc.

Mais de façon générale, proxmox dispose d'une interface web intuitive, ce qui permet de gérer simplement les machines virtuelles, les conteneurs, le réseau et le stockage. De plus, il est possible de faire du clustering avec proxmox, permettant de gérer plusieurs nœuds Proxmox comme un seul, pour une meilleure évolutivité et une haute disponibilité. En effet grâce à Proxmox HA Manager, proxmox surveille les VMs et les conteneurs et peut automatiquement les redémarrer sur un autre nœud en cas de panne.

Il compte aussi des fonctionnalités de sauvegarde et de restauration des VMs et conteneurs, avec prise en charge des instantanés et supports de stockage partagé (ZFS pour le stockage local et distribué, NFS pour le stockage partagé etc.

Proxmox offre des outils de gestion avancée des réseaux virtuels, notamment la prise en charge des VLANs, des ponts réseau et des configurations SDN.

De plus, c'est open-source et complètement gratuit.

3 - Mise en place de la solution

3.1 Les ressources que nous avons utilisées

Niveau matériels :

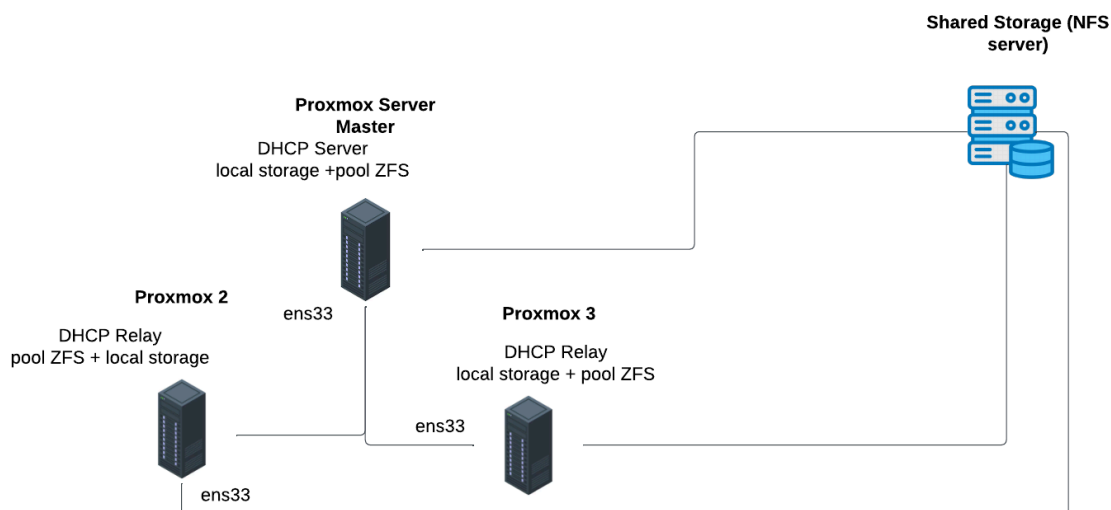
N'ayant pas les ressources nécessaires, ou du moins le matériel physique pour mettre en place notre infrastructure de virtualisation avec Proxmox, nous avons fait le choix de simuler cette infrastructure dans des machines virtuelles.

On peut aisément utiliser VirtualBox ou VMware pour cela.

3.2 L'architecture de notre infrastructure

N'ayant pas la capacité de mettre en place un nombre conséquent et réaliste de serveurs (proxmox, stockage) par un déficit en mémoire physique et CPU nous avons fait le choix de réduire/combiner certains serveurs dans une même VM.

Schéma de l'architecture de notre infrastructure :



En effet nous avons fait le choix d'installer 3 serveurs proxmox (debian) afin de faire un **cluster** sur des VMs virtual box/VMware. Nous avons fait le choix d'installer un serveur DHCP sur l'un deux (Proxmox Master), bien qu'il n'y ait pas de Master à proprement parler dans un cluster proxmox, ils sont tous égaux au sein des nœuds. En effet, dans un cas réel le serveur DHCP est sur un serveur spécifique. Par exemple ici, si notre Proxmox Server Master tombe en panne, le DHCP Server aussi (c'est pour économiser des ressources).

Les deux autres nœuds ont DHCP Relay d'installé et tous les serveurs proxmox ont un pool ZFS.

Nous avons également mis en place un serveur NFS dans une autre machine virtuelle VMware/virtualbox sur un serveur ubuntu. Ce serveur NFS fera office de stockage partagé pour le cluster.

A noter : Pour installer proxmox, il est nécessaire d'avoir un debian 11 ou 12.

Il est aussi intéressant de constater que si vous n'avez pas les ressources nécessaires pour 3 serveurs proxmox, vous pouvez créer un serveur arbitraire

Avec les trois serveurs proxmox, nous mettrons en place un cluster Proxmox, qui est donc un groupe de serveurs physiques (nœuds) qui collaborent pour offrir une gestion centralisée et des fonctions avancées, comme la HA et la migration des machines VM et conteneurs entre serveur.

L'avantage, c'est que si un nœud tombe en panne, les services peuvent continuer sur un autre nœud.

Par exemple, si Proxmox 1 tombe en panne, le cluster reste opérationnel, j'accède à l'interface web de proxmox 2. De plus avec l'utilisation d'un stockage partagé, la migration est plus rapide car elle ne transfère pas les données des disques de proxmox1 à proxmox 2/3.

De plus dans le cas d'une panne ou proxmox1 est hors service, les configurations des VM (RAM,CPU. etc) sont synchronisées entre les nœuds via le cluster proxmox, donc proxmox2 sait où trouver les VM et comment les configurer.

A noter : Le mieux est d'avoir 3 nœuds, car avec deux nœuds si un serveur proxmox tombe en panne le quorum peut être perdu, on ne pourrait plus administrer le cluster .

Mais on peut forcer le quorum temporairement sur proxmox 2 avec :

`pvecm expected 1` (risqué)

Le quorum est utilisé pour s'assurer que les décisions prises sont cohérentes et évitent les situations où un nœud pense être le seul à avoir raison.

Un quorum exige que plus de 50 % des nœuds soient opérationnels pour que le cluster fonctionne correctement. Lorsqu'on a que deux nœuds, le cluster est bloqué.

Pour palier à cela on peut créer un arbitre :

`pvecm add <ip_arbitre> --votes 0`

Afin d'avoir un vote supplémentaire !

3.3 Installation des serveurs virtuels

Tout d'abord, sur les machines où seront installées Proxmox, il faut activer via le BIOS VT-x/AMD-V, qui sont des extensions matérielles du processeur qui permettent d'améliorer les performances des environnements de virtualisation. Elles permettent de passer directement des instructions de la machine virtuelle au matériel physique sans passer par un logiciel d'émulation.

Dans le cas d'une installation de Proxmox sur une VM, il faut activer la virtualisation imbriquée pour permettre à votre VM d'exécuter une autre VM.

Il est aussi important d'activer au niveau du bios PAE/NX car si des VM ou CT sont créés via Proxmox et ne sont pas en 64 bits, il est utile d'avoir cette option activée pour pouvoir gérer des systèmes 32 bits avec +4 GO de mémoire.

Pour activer la virtualisation imbriquée sur virtualbox on peut utiliser cette commande :
`VBoxManage modifyvm <Nom_de_la_VM> --nested-hw-virt on`

Et modifier ces paramètres dans les options processeur des VMs :

Fonctions avancées : ☒ Activer PAE/NX
☒ Activer VT-x/AMD-V imbriqué

	Serveurs Proxmox	Serveur NFS
Environnement	Debian 12	Ubuntu server 24

Le plus possible de processeurs doit être alloué aux serveurs Proxmox, même chose niveau RAM, stockage aussi si local. Miser surtout sur le stockage pour le serveur NFS !

A noter : Il est nécessaire de configurer 2 interfaces, de préférence en bridge. L'une servira pour l'accès ssh, l'interface web et communiquer avec les autres serveurs.

L'autre sans IP sera considérée comme une interface brute pour la virtualisation, elle sera connectée à un bridge réseau (vmbf0) et donc qui permettra à nos VMs Proxmox d'accéder au réseau (il permettra donc aux machines virtuelles Proxmox de l'utiliser pour accéder au réseau physique)

3.4 Automatisation des installations

Afin de faciliter la mise en place des playbooks Ansible pour automatiser et faciliter l'installation et la configuration.

Nous avons mis en place 12 playbooks pour faciliter l'installation et configuration de notre solution.

Les principaux sont :

`install.yaml` qui installe et configure proxmox sur les serveurs.

`install_post.yaml` supprime le fichier `/etc/apt/sources.list.d/pve-enterprise.list` qui configure le dépôt pour les utilisateurs disposant d'un abonnement Proxmox, car on peut avoir un souci à cause de ça.

`dhcp-server_setup.yaml` qui permet d'installer et configurer un serveur DHCP avec le paquet **ISC DHCP Server** sur l'hôte proxmox1, il est nécessaire afin de permettre aux VMs hébergées par proxmox de pouvoir communiquer entre-elles.

Ici nos VMs feront partie du réseau 192.168.3.0/24. (voir fichier templates/dhcpd.conf.j2)

`dhcp-relay_setup.yaml` configure un **DHCP relay** sur les autres nœuds. Le **DHCP relay** permet de transmettre les requêtes DHCP provenant de clients situés sur d'autres sous-réseaux au serveur DHCP principal situé sur `proxmox1`. Les VMs qui sont sur les autres nœuds pourront bénéficier du même réseau et du même adressage.

`nfs_install.yaml` configure un **serveur NFS** pour partager un répertoire avec Proxmox.

`nfs_connect.yaml` ajoute ce partage comme stockage dans Proxmox, permettant ainsi une gestion centralisée des ressources (images des VMs, sauvegardes..)

`create_cluster.yaml` configure un nœud Proxmox en tant que **cluster** pour la gestion centralisée des ressources dans un environnement virtualisé.

Les playbooks principaux doivent être exécutés dans cet ordre !

Les autres playbooks :

`upload_iso_local.yaml` : utilisé pour uploader des fichiers ISO depuis un répertoire source local vers le stockage local d'un hôte Proxmox (Éviter de le faire via l'interface).

`upload_iso_shared.yaml` : Similaire mais vers le serveur NFS (répertoire partagé).

`vm_qm-install.yaml` automatise la création, la configuration et le démarrage d'une VM sur un hôte Proxmox en utilisant les commandes qm.

`vm_api-install.yaml` : Même chose mais en utilisant l'api proxmox.

`create_user.yaml` automatise la création d'un utilisateur PAM pour Proxmox, configure ses permissions et attribue un rôle d'administration.

Inventaire Ansible :

hosts

```
all:
  children:
    storage:
      hosts:
        nfs:
          ansible_host: 192.168.38.149
          ansible_user: root
          ansible_ssh_private_key_file: ./ssh/id_ed25519
          ansible_python_interpreter: /usr/bin/python3
    proxmoxs:
      hosts:
        proxmox1:
          ansible_host: 192.168.38.146
          ansible_user: root
          ansible_ssh_private_key_file: ./ssh/id_ed25519
          ansible_python_interpreter: /usr/bin/python3
        proxmox2:
          ansible_host: 192.168.38.148
          ansible_user: root
          ansible_ssh_private_key_file: ./ssh/id_ed25519
          ansible_python_interpreter: /usr/bin/python3
        proxmox3:
          ansible_host: 192.168.38.151
          ansible_user: root
          ansible_ssh_private_key_file: ./ssh/id_ed25519
          ansible_python_interpreter: /usr/bin/python3
```

Il décrit les hôtes dans deux groupes : **storage** (contenant le serveur NFS) et **proxmoxs** (contenant trois hôtes Proxmox).

Ici on utilise les utilisateurs root pour la simplicité mais ce n'est pas top pour la sécurité !
Il faudra donc, pour faire fonctionner les playbooks, créer une paire de clés pour les serveurs Proxmox et NFS afin de permettre à Ansible d'effectuer les installations et configurations.

Détails sur les playbooks :

install.yaml

```
---
- name: Install and Configure Proxmox VE
  hosts: proxmox # Le groupe ou hôte sur lequel exécuter les tâches
  become: true
  gather_facts: yes
  vars:
    domainname: ssi.edu
    dhcp_interface: "vmbr0"
    out_interface: "ens33"
    priv_interface: "ens36"
    proxmox_interfaces: |
      source /etc/network/interfaces.d/*

    auto lo
    iface lo inet loopback

    auto {{ out_interface }}
    iface {{ out_interface }} inet static
    address {{ hostvars[inventory_hostname]['ansible_host'] }}
    netmask {{ ansible_default_ipv4.netmask }}
    gateway {{ ansible_default_ipv4.gateway }}
    dns-nameservers 8.8.8.8, 8.8.4.4

    iface {{ priv_interface }} inet manual

    auto vmbr0
    iface vmbr0 inet static
    address 192.168.3.1/24
    bridge-ports {{ priv_interface }}
    bridge-stp off
    bridge-fd 0
  tasks:
    - name: Remove packages
      ansible.builtin.package:
        name:
        - os-prober
        state: absent

    - name: Install packages
      ansible.builtin.package:
        name:
        - gpg
        state: present
```

```

- name: Add Proxmox GPG key
  ansible.builtin.get_url:
    url: "https://enterprise.proxmox.com/debian/proxmox-release-{{ ansible_distribution_release }}.gpg"
    dest: "/etc/apt/trusted.gpg.d/proxmox-release-{{ ansible_distribution_release }}.gpg"
    owner: root
    group: root
    mode: "0644"

- name: Add Proxmox repository and update cache
  ansible.builtin.apt_repository:
    repo: "deb [arch=amd64] http://download.proxmox.com/debian/pve {{ ansible_distribution_release }}
pve-no-subscription"
    state: present
    update_cache: true

- name: Clean /etc/hosts
  ansible.builtin.lineinfile:
    path: /etc/hosts
    regexp: "^127.0.1.1.*"
    owner: root
    group: root
    mode: "0644"
    state: absent

- name: Add host in /etc/hosts
  ansible.builtin.lineinfile:
    path: /etc/hosts
    regexp: "^{{ ansible_all_ipv4_addresses[0] }}"
    owner: root
    group: root
    mode: "0644"
    state: present
    line: "{{ ansible_all_ipv4_addresses[0] }} {{ ansible_hostname }}.{{ domainname }} {{ ansible_hostname
}}}"

- name: Set IPV4
  ansible.posix.sysctl:
    sysctl_file: /etc/sysctl.d/proxmox.conf
    name: "{{ item.name }}"
    value: "{{ item.value }}"
    state: present
    loop:
      - {name: net.ipv4.conf.all.rp_filter, value: 1 }
      - {name: net.ipv4.icmp_echo_ignore_broadcasts, value: 1 }
      - {name: net.ipv4.ip_forward, value: 1 }

- name: Installer le package bridge-utils
  apt:
    name: bridge-utils
    state: present

- name: Configure network interfaces
  ansible.builtin.copy:
    dest: /etc/network/interfaces
    content: "{{ proxmox_interfaces }}"
    owner: root
    group: root

```

mode: "0644"

- name: Reload networking
ansible.builtin.command:
cmd: systemctl restart networking

- name: Wait for Proxmox to come back online
ansible.builtin.wait_for:
port: 22
host: '{{ ansible_host }}'
timeout: 60

- name: Appliquer la configuration sysctl
ansible.builtin.command:
cmd: /usr/sbin/sysctl -p

- name: Installer le paquet iptables-persistent
ansible.builtin.apt:
name: iptables-persistent
state: present
update_cache: yes

- name: Ajouter une règle iptables pour le NAT
ansible.builtin.command:
cmd: iptables -t nat -A POSTROUTING -o {{ out_interface }} -s 192.168.3.0/24 -j MASQUERADE

- name: Enregistrer les règles iptables (optionnel, selon le système)
ansible.builtin.shell:
cmd: iptables-save > /etc/iptables/rules.v4
when: ansible_distribution in ['Debian', 'Ubuntu']

- name: Fix broken dependencies
ansible.builtin.command:
cmd: apt --fix-broken install -y

- name: Install missing Proxmox dependencies
ansible.builtin.apt:
name:
- pve-firmware
- proxmox-kernel-6.8.12-5-pve
state: present
update_cache: yes

- name: Perform full upgrade
ansible.builtin.apt:
upgrade: dist
update_cache: yes

- name: Install Proxmox and tools
ansible.builtin.package:
name:
- proxmox-ve
- ksm-control-daemon
- locales-all
- chrony
- libguestfs-tools

```

state: present

- name: Remove kernel
ansible.builtin.package:
name: "linux-image-6.1*"
state: absent

- name: Installer le package bridge-utils
apt:
name: bridge-utils
state: present

- name: Restart Proxmox proxy service
ansible.builtin.service:
name: pveproxy
state: restarted

- name: Activer le routage IP dans sysctl
ansible.builtin.lineinfile:
path: /etc/sysctl.conf
regexp: '^net.ipv4.ip_forward'
line: 'net.ipv4.ip_forward=1'
state: present

- name: Appliquer la configuration sysctl
ansible.builtin.command:
cmd: /usr/sbin/sysctl -p

```

Possibilité de choisir le nom des variables des interfaces de connexion externe et l'interface utilisée comme bridge-port pour **vmbro**. (pour permettre au VM de communiquer entre-elles).

1. Préparation du système

1. Supprimer les paquets inutiles : Retire **os-prober**.
2. Installer des paquets nécessaires :
 - **gpg**: Pour gérer les clés GPG.
 - **bridge-utils**: Pour gérer les ponts réseau.
3. Ajouter la clé GPG de Proxmox : Télécharge et installe la clé GPG pour vérifier les paquets Proxmox.
4. Ajouter le dépôt Proxmox : Ajoute le dépôt **pve-no-subscription** pour éviter les avertissements liés aux abonnements.
5. Nettoyer **/etc/hosts** :
 - Supprime toute ligne contenant **127.0.1.1**.
 - Ajoute une ligne avec l'adresse IP actuelle, le nom d'hôte et le domaine.

2. Configuration réseau

6. Configurer des paramètres réseau IPv4 avec **sysctl** :
 - **net.ipv4.conf.all.rp_filter**: Active le filtrage des adresses source.
 - **net.ipv4.icmp_echo_ignore_broadcasts**: Ignore les requêtes ping en broadcast.
 - **net.ipv4.ip_forward**: Active le routage IP.

7. Configurer les interfaces réseau : Copie la configuration réseau définie dans `proxmox_interfaces` vers `/etc/network/interfaces`.
8. Redémarrer le service réseau : Applique la nouvelle configuration.
9. Attendre le retour en ligne : Attend que l'hôte soit accessible via SSH après les modifications.

3. Configuration des règles IP et NAT

10. Installer `iptables-persistent` : Sauvegarde des règles de pare-feu.
11. Ajouter une règle NAT : Configure une règle `iptables` pour masquer les adresses source (SNAT) sur l'interface externe.
12. Enregistrer les règles `iptables` : Sauvegarde les règles dans `/etc/iptables/rules.v4`.

4. Installation et mise à jour de Proxmox

13. Corriger les dépendances cassées : Résout les problèmes potentiels avec `apt --fix-broken`.
14. Installer des dépendances Proxmox :
 - `pve-firmware`: Firmware pour Proxmox.
 - `proxmox-kernel-6.8.12-5-pve`: Noyau spécifique pour Proxmox.
15. Mettre à jour le système : Effectue une mise à jour complète (`dist-upgrade`).
16. Installer Proxmox et ses outils :
 - `proxmox-ve`: Paquet principal pour Proxmox.
 - `kvm-control-daemon`: Outil d'optimisation mémoire.
 - `locales-all`: Localisation.
 - `chrony`: Synchronisation du temps.
 - `libguestfs-tools`: Gestion des systèmes invités.
17. Supprimer un ancien noyau Linux : Nettoie les noyaux inutiles (`linux-image-6.1*`).

5. Configuration finale

18. Redémarrer le service proxy de Proxmox : Relance le service `pveproxy`.
19. Activer le routage IP dans `sysctl` : Vérifie et applique `net.ipv4.ip_forward=1`.
20. Appliquer les paramètres `sysctl` : Recharge les configurations système avec `sysctl -p`.

Il permet donc la mise en place d'une infrastructure Proxmox avec un réseau configuré pour permettre l'utilisation de NAT et le routage d'adresses IP entre le réseau local et Internet.

```

1 ---
2 - name: Install and Configure Proxmox VE
3   hosts: proxmox3
4   become: true
5   gather_facts: yes
6   tasks:
7     - name: Add Proxmox GPG key
8       ansible.builtin.get_url:
9         url: "https://enterprise.proxmox.com/debian/proxmox-release-{{ ansible_distribution_release }}.gpg"
10        dest: "/etc/apt/trusted.gpg.d/proxmox-release-{{ ansible_distribution_release }}.gpg"
11        owner: root
12        group: root
13        mode: "0644"
14
15    - name: Add Proxmox repository and update cache
16      ansible.builtin.apt_repository:
17        repo: "deb [arch=amd64] http://download.proxmox.com/debian/pve {{ ansible_distribution_release }} pve-no-subscription"
18        state: present
19        update_cache: true
20
21    - name: Remove enterprise repo
22      ansible.builtin.file:
23        path: /etc/apt/sources.list.d/pve-enterprise.list
24        state: absent
25
26

```

Permet comme dit plus haut de fixer le souci de **pve-enterprise** après un redémarrage. Le playbook supprime le fichier de configuration du dépôt Proxmox Enterprise pour éviter des messages d'erreur liés à l'absence d'un abonnement valide.

dhcp-server_setup.yaml

```

---
- name: Install and Configure DHCP server|
  hosts: proxmox1
  become: false
  gather_facts: yes
  vars:
    dhcp_interface: "vmbr0"
  tasks:
    - name: Installer le paquet ISC DHCP Server
      ansible.builtin.apt:
        name: isc-dhcp-server
        state: present
        update_cache: yes

    - name: Configurer l'interface DHCP
      ansible.builtin.lineinfile:
        path: /etc/default/isc-dhcp-server
        regexp: '^INTERFACESv4=.*'
        line: 'INTERFACESv4="{{ dhcp_interface }}"'
        state: present

    - name: Configurer le fichier dhcpd.conf
      ansible.builtin.template:
        src: templates/dhcpd.conf.j2
        dest: /etc/dhcp/dhcpd.conf
        owner: root
        group: root
        mode: '0644'

    - name: Redémarrer le service DHCP
      ansible.builtin.systemd:
        name: isc-dhcp-server
        state: restarted
        enabled: true

```

Ce playbook permet de configurer un **serveur DHCP** sur un hôte Proxmox, notamment pour :

1. **Gérer le réseau local des VM :**
 - Le DHCP peut attribuer dynamiquement des adresses IP aux machines virtuelles connectées au bridge réseau défini (`vmbbr0`).
2. **Automatiser le provisionnement des VM :**
 - En utilisant le DHCP, chaque machine virtuelle peut recevoir une adresse IP sans intervention manuelle.
3. **Créer un environnement réseau isolé :**
 - En attribuant des adresses IP uniquement sur l'interface privée, le réseau des VM peut être séparé du réseau principal.

`dhcpd.conf`

```
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.3.0 netmask 255.255.255.0 {
    range 192.168.3.2 192.168.3.100;
    option routers 192.168.3.1;
    option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

`dhcp-relay_setup.yaml`

```
---
- name: configure DHCP relay on proxmox node
  hosts: proxmox3
  become: false
  vars:
    dhcp_relay_interfaces: # Interfaces on proxmox node
      - vmbbr0
      - ens33
    dhcp_relay_options: "" # Enable detailed logging for debugging
  tasks:
    - name: Retrieve Proxmox1 IP (aka the DHCP server) address from inventory
      set_fact:
        dhcp_server_ip: "{{ hostvars['proxmox1']['ansible_host'] }}"

    - name: Install ISC DHCP relay
      ansible.builtin.package:
        name: isc-dhcp-relay
        state: present

    - name: Configure ISC DHCP relay
      ansible.builtin.template:
        src: dhcp-relay.conf.j2
        dest: /etc/default/isc-dhcp-relay
        owner: root
        group: root
        mode: "0644"

    - name: Restart DHCP relay service
      ansible.builtin.systemd:
        name: isc-dhcp-relay
        state: restarted
        enabled: true

    - name: Verify DHCP relay service status
      ansible.builtin.systemd:
        name: isc-dhcp-relay
        state: started
        enabled: true
```


Le playbook précédent configure un **serveur DHCP** sur le nœud proxmox 'master'. Ce playbook configure un **relay DHCP** sur un autre nœud (proxmox3, mais on dans évidemment, il faut aussi l'exécuter sur proxmox2) pour rediriger les requêtes DHCP vers proxmox1.

Ensemble, ces playbooks permettent :

1. Une attribution centralisée d'adresses IP (gérée par proxmox1).
2. Une couverture réseau étendue grâce au relay DHCP sur proxmox 2 et 3.

dhcp-relay.conf

```
# DHCP Relay Configuration
SERVERS="{{ dhcp_server_ip }}"
INTERFACES="{{ dhcp_relay_interfaces | join(' ') }}"
OPTIONS="{{ dhcp_relay_options }}"
```

nfs_install.yml

```
---
- name: Configure NFS Server and Add Storage to Proxmox
  hosts: nfs
  become: false
  vars:
    export_net1: "192.168.38.0/24"
    export_net2: "192.168.3.0/24"
  tasks:
    # 1. Installer le serveur NFS
    - name: Install NFS server
      apt:
        name: nfs-kernel-server
        state: present

    # 2. Créer le répertoire à partager
    - name: Create NFS share directory
      file:
        path: /srv/proxmox-nfs
        state: directory
        mode: '0777'

    # 3. Configurer le partage NFS
    - name: Configure NFS exports
      lineinfile:
        path: /etc/exports
        line: "/srv/proxmox-nfs {{ export_net1 }}(rw,sync,no_root_squash,no_subtree_check) {{ export_net2 }}(rw,sync,no_root_squash,no_subtree_check)"
        create: yes

    # 4. Redémarrer le service NFS
    - name: Restart NFS server
      systemd:
        name: nfs-kernel-server
        state: restarted

    # 5. Activer NFS au démarrage
    - name: Enable NFS server on boot
      systemd:
        name: nfs-kernel-server
        enabled: true
```

Permet simplement d'installer le serveur NFS sur la machine sous ubuntu server. Pensez à bien changer la variable `export_net1` si le réseau hébergement les serveurs proxmox n'est pas le même que celui-ci.

nfs_connect.yaml

```
---
- name: Add NFS Storage to Proxmox
  hosts: proxmox1
  become: true
  tasks:
    # 6. Ajouter le stockage NFS dans Proxmox
    - name: Add NFS Storage in Proxmox
      shell: |
        pvesm add nfs proxmox-nfs --server {{ hostvars['nfs']['ansible_host'] }} --export /srv/proxmox-nfs --content images,backup
      args:
        executable: /bin/bash
```

Ce playbook configure un stockage NFS sur le nœud **proxmox1** en :

1. Ajoutant un partage NFS (**/srv/proxmox-nfs**) accessible via le serveur spécifié.
2. Spécifiant les types de contenu pris en charge : **images ISO** et **sauvegardes**.

Cela permet à Proxmox d'utiliser un stockage externe via NFS pour améliorer la flexibilité et la gestion des ressources et donc d'avoir un stockage partagé pour les nœuds. du cluster.

create_cluster.yaml

```
---
- name: Create Proxmox Cluster
  hosts: proxmox1
  become: true
  vars:
    default_cluster_name: "my-cluster"
  tasks:
    - name: Set cluster name
      ansible.builtin.set_fact:
        cluster_name: "{{ cluster_name | default(default_cluster_name) }}"
    - name: Create Proxmox Cluster
      ansible.builtin.command:
        cmd: "pvecm create {{ cluster_name }}"
      register: cluster_output
    - name: Display Cluster Output
      ansible.builtin.debug:
        msg: "{{ cluster_output.stdout }}"
```

Ce playbook initialise un cluster Proxmox avec un nom défini (**my-cluster** par défaut). Le nœud **proxmox1** devient le **nœud maître** du cluster.

Il faut ensuite rejoindre ce cluster avec les autres nœuds, on ne l'a pas automatisé avec ansible car cela nécessite le mot de passe root du nœud maître, du moins on peut l'automatiser si on a les clés ssh maître au préalable sur les autres nœuds. Le plus simple reste de passer directement par l'interface et de rejoindre via celle-ci le cluster.

Les playbooks moins essentiels :

create_user.yaml

```
1 ---
2 - name: Create a PAM user with no login and assign PVEAdmin role
3   hosts: proxmox3
4   become: true
5   tasks:
6
7     - name: Ensure the system user exists with no login shell
8       ansible.builtin.user:
9         name: "{{ new_user }}"
10        password: "{{ new_user_hashed_password }}"
11        comment: "{{ new_user_comment }}"
12        shell: /usr/sbin/nologin
13      vars:
14        new_user_hashed_password: "{{ new_user_password | password_hash('sha512') }}"
15
16     - name: Add the user to Proxmox PAM realm
17       ansible.builtin.command:
18         cmd: "pveum useradd {{ new_user }}@pam -comment '{{ new_user_comment }}'"
19
20     - name: Validate the user exists in Proxmox
21       ansible.builtin.command:
22         cmd: "pveum user list"
23       register: user_list_result
24       failed_when: "'{{ new_user }}@pam' not in user_list_result.stdout"
25
26     - name: Set the Proxmox user's password
27       ansible.builtin.shell:
28         cmd: "echo '{{ new_user_password }}' | pveum passwd {{ new_user }}@pam"
29       no_log: true
30       register: passwd_result
31       failed_when: "'unable to connect' in passwd_result.stderr"
32
33     - name: Assign PVEAdmin role to the user
34       ansible.builtin.command:
35         cmd: "pveum aclmod / -user {{ new_user }}@pam -role PVEAdmin"
36
37   vars_prompt:
38     - name: "new_user"
39       prompt: "Enter the username for the new PAM user"
40       private: no
41
42     - name: "new_user_comment"
43       prompt: "Enter a comment for the new user (optional)"
44       private: no
45
46     - name: "new_user_password"
47       prompt: "Enter the password for the new user"
48       private: yes
```

Pour la création d'un utilisateur, il faut également le mot de passe du root si on passe par l'API REST, pour une question de sécurité et de simplicité on est passé via les commandes de proxmox.

vm_qm-install.yml

```
hosts: proxmox1
become: true
vars:
  vm_name: "vm-test"
  vm_cores: 1
  vm_memory: 1024
  storage_pool: "proxmox-nfs"
  vm_disk_size: "4"
  iso_dir_path: "/mnt/pve/proxmox-nfs/template/iso/"
  proxmox_node: "proxmoxvm"
  network_bridge: "vmbbr0"

tasks:
  - name: Find the next available VM ID
    ansible.builtin.command:
      cmd: "pvesh get /cluster/nextid"
    register: next_vm_id
    changed_when: false

  - name: List available ISO files in Proxmox storage
    ansible.builtin.find:
      paths: "{{ iso_dir_path }}"
      patterns: "*.iso"
    register: iso_files

  - name: Display available ISO files
    debug:
      msg: "Available ISO files: {{ iso_files.files | map(attribute='path') | join(', ') }}"

  - name: Prompt user to select an ISO file
    pause:
      prompt: "Enter the name of the ISO file to use (without path):"
    register: user_input

  - name: Debug a variable
    ansible.builtin.debug:
      var: user_input

  - name: Validate selected ISO file and save to variable
    set_fact:
      selected_iso: "{{ user_input.user_input }}"
    when: "'{{ iso_dir_path }}' + user_input.user_input in (iso_files.files | map(attribute='path'))"
    failed_when: "'{{ iso_dir_path }}' + user_input.user_input not in (iso_files.files | map(attribute='path'))"

  - name: Create a disk image for the VM
    ansible.builtin.command:
      cmd: >
      qm create {{ next_vm_id.stdout }} --name {{ vm_name }} --memory {{ vm_memory }} --cores {{ vm_cores }} --net0 virtio,bridge={{ network_bridge }} --boot order=scsi0;ide2
    args:
      chdir: /root

  - name: Attach the disk to the VM
    ansible.builtin.command:
      cmd: >
      qm set {{ next_vm_id.stdout }} --scsihw virtio-scsi-pci --scsi0 {{ storage_pool }}:{{ vm_disk_size }}
    args:
      chdir: /root

  - name: Attach the ISO to the VM
    ansible.builtin.command:
      cmd: qm set {{ next_vm_id.stdout }} --cdrom {{ storage_pool }}:iso/{{ user_input.user_input }}

  - name: Start the VM
    ansible.builtin.command:
      cmd: qm start {{ next_vm_id.stdout }}
```

Ce playbook configure une VM avec les spécifications suivantes :

- 1 Go de RAM, 1 cœur CPU, disque virtuel de 4 Go (sur le serveur de stockage partagé BFS).
- Fichier ISO monté comme lecteur CD-ROM pour l'installation.
- Connexion au réseau via un bridge (**vmbbr0**).
- Création entièrement automatisée, avec une étape interactive pour sélectionner le fichier ISO.

vm_api-install.yaml

```
- name: Create a new VM in Proxmox
  hosts: proxmox1
  become: true
  vars_prompt:
    - name: proxmox_password
      prompt: "Enter the Proxmox root password"
      private: yes # Hides input during the prompt
  vars:
    vm_name: "vm-test"
    vm_cores: 1
    vm_memory: 1024
    vm_storage: "proxmox-nfs"
    vm_disk_size: "4G"
    iso_dir_path: "/mnt/pve/proxmox-nfs/template/iso/"
    proxmox_node: "proxmoxvm"
    network_bridge: "vmbr0"

  tasks:
    - name: Ensure libguestfs-tools is installed
      ansible.builtin.package:
        name: libguestfs-tools
        state: present

    #- name: Ensure pip3 is installed
    #  ansible.builtin.package:
    #    name: python3-pip
    #    state: present

    - name: Ensure Py lib 'proxmoxer' is installed
      ansible.builtin.package:
        name: python3-proxmoxer
        state: present

    - name: List available ISO files in Proxmox storage
      ansible.builtin.find:
        paths: "{{ iso_dir_path }}"
        patterns: "*.iso"
        register: iso_files

    - name: Display available ISO files
      debug:
        msg: "Available ISO files: {{ iso_files.files | map(attribute='path') | join(', ') }}"

    - name: Prompt user to select an ISO file
      pause:
        prompt: "Enter the name of the ISO file to use (without path):"
        register: user_input

    - name: Validate selected ISO file and save to variable
      set_fact:
        selected_iso: "{{ user_input.user_input }}"
      when: " '{{ iso_dir_path }}' + user_input.user_input in (iso_files.files | map(attribute='path'))"
      failed_when: "'{{ iso_dir_path }}' + user_input.user_input not in (iso_files.files | map(attribute='path'))"

    - name: Create a new virtual machine
      community.general.proxmox_kvm:
        api_host: "{{ ansible_host }}"
        api_user: "root@pam"
        api_password: "{{ proxmox_password }}"
        node: "{{ proxmox_node }}"
        name: "{{ vm_name }}"
        cores: "{{ vm_cores }}"
        memory: "{{ vm_memory }}"
        ostype: "l26"
        scsihw: "virtio-scsi-single" # Set SCSI controller to VirtIO SCSI single
        scsi:
          storage: "proxmox-nfs" # Use proxmox-nfs for the disk
          size: "{{ vm_disk_size }}" # Disk size
        ide:
          file: "{{ iso_dir_path }}/{{ selected_iso }}" # ISO file
          media: "cdrom" # Specify CD-ROM for ISO boot
        net:
          bridge: "{{ network_bridge }}" # Bridge to attach
        state: present
        register: create_vm_output

    - name: Start the newly created VM
      community.general.proxmox_kvm:
        api_host: "{{ ansible_host }}"
        api_user: "root@pam"
        api_password: "{{ proxmox_password }}"
        vmid: "{{ create_vm_output.vmid }}"
        node: "{{ proxmox_node }}"
        state: started
```

Même chose mais avec le module `community.general.proxmox_kvm` pour interagir avec l'api proxmox.

upload_iso_shared.yaml

```
1 - name: Upload ISO files to a shared NFS directory
2 hosts: proxmox1 # The node to interact with the NFS share
3 become: false
4 vars:
5   nfs_mount_point: /mnt/pve/proxmox-nfs # Shared NFS mount point
6   iso_source_directory: ./iso_images # Local directory with ISO files
7   nfs_server_ip: "{{ hostvars['nfs']['ansible_host'] }}"
8   nfs_server_srcpath: /srv/proxmox-nfs
9 tasks:
10  - name: Ensure NFS share is mounted
11    ansible.builtin.mount:
12      path: "{{ nfs_mount_point }}"
13      src: " {{ nfs_server_ip }}:{{ nfs_server_srcpath }}"
14      fstype: nfs
15      opts: rw
16      state: mounted
17
18  - name: Find all ISO files in the source directory
19    delegate_to: localhost
20    ansible.builtin.find:
21      paths: "{{ iso_source_directory }}"
22      patterns: "*.iso"
23      recurse: yes
24      register: found_iso_files
25
26  - name: Upload ISO files to the shared NFS directory
27    ansible.builtin.copy:
28      src: "{{ item.path }}"
29      dest: "{{ nfs_mount_point }}/template/iso/"
30      remote_src: false
31      owner: root
32      group: root
33      mode: '0644'
34    with_items: "{{ found_iso_files.files }}"
35  - name: Fix ownership and permissions
36    ansible.builtin.file:
37      path: "{{ nfs_mount_point }}/template/iso/{{ item.path | basename }}"
38      owner: root
39      group: root
40      mode: '0644'
41    with_items: "{{ found_iso_files.files }}"
42  - name: Verify ISO files in the NFS directory
43    ansible.builtin.command:
44      cmd: ls -l "{{ nfs_mount_point }}/template/iso/"
45    register: iso_list
46
47  - name: Display uploaded ISO files
48    ansible.builtin.debug:
49      msg: "{{ iso_list.stdout_lines }}"
```

Montage du partage NFS :

- Vérifie et monte le partage NFS sur le nœud Proxmox à un point de montage local.

Recherche locale des fichiers ISO :

- Identifie tous les fichiers ISO dans un répertoire source local.

Transfert des fichiers vers le NFS :

- Copie les fichiers ISO trouvés vers le répertoire `template/iso` sur le partage NFS.

Correction des permissions :

- Applique les permissions et propriétés correctes aux fichiers transférés.

Vérification des fichiers :

- Affiche la liste des fichiers ISO présents sur le partage NFS pour confirmation.

upload_iso_local.yaml

```

1 ---
2 - name: Upload ISO files to Proxmox local storage
3   hosts: proxmox2
4   become: false
5   vars:
6     iso_source: ./iso_images
7     iso_destination: /var/lib/vz/template/iso
8   tasks:
9     - name: Ensure ISO destination directory exists
10      ansible.builtin.file:
11        path: "{{ iso_destination }}"
12        state: directory
13        owner: root
14        group: root
15        mode: '0755'
16
17     - name: Find all ISO files in the source dir
18      delegate_to: localhost
19      ansible.builtin.find:
20        paths: "{{ iso_source }}"
21        patterns: "*.iso"
22        recurse: yes
23      register: found_iso_files
24
25     - name: Upload ISO files
26      ansible.builtin.copy:
27        src: "{{ item.path }}"
28        dest: "{{ iso_destination }}"
29        owner: root
30        group: root
31        mode: '0644'
32      loop: "{{ found_iso_files.files }}"
33
34     - name: List uploaded ISO files
35      ansible.builtin.command:
36        cmd: ls -l "{{ iso_destination }}"
37      register: iso_list
38
39     - name: Display uploaded ISO files
40      ansible.builtin.debug:
41        msg: "{{ iso_list.stdout_lines }}"

```

Préparation :

- Vérifie que le répertoire de destination des ISO existe sur le nœud Proxmox, sinon le crée.

Recherche des fichiers ISO :

- Identifie tous les fichiers **.iso** présents dans un répertoire local sur la machine exécutant Ansible.

Transfert :

- Copie les fichiers ISO trouvés vers le stockage local du nœud Proxmox.

Vérification :

- Liste et affiche les fichiers présents dans le répertoire de destination sur le nœud Proxmox.

3.5 Configuration manuelle

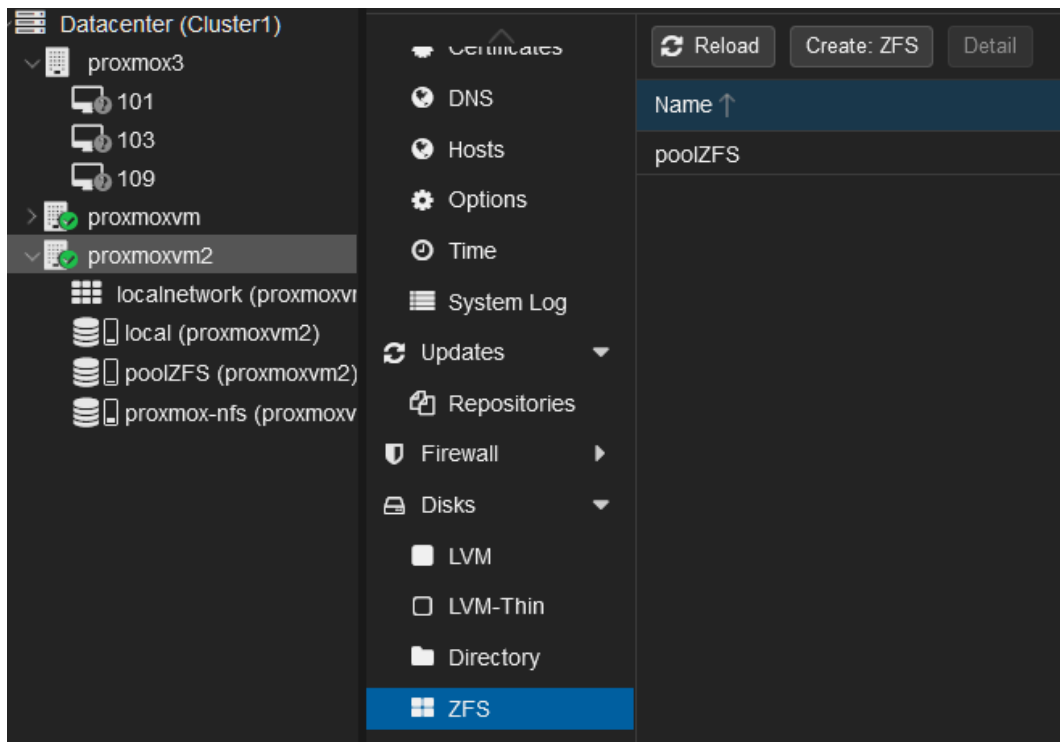
ZFS Replication

1. Definition, scope

La réplication ZFS est une fonctionnalité puissante de l'écosystème ZFS (Zettabyte File System) conçue pour une redondance des données, une reprise après un problème et une migration efficaces et fiables. Elle implique la copie en temps réel ou programmée d'ensembles de données et d'instantanés d'un pool de stockage vers un autre, soit localement, soit sur des systèmes géographiquement séparés. En exploitant les capacités uniques d'instantanés et d'envoi/réception de ZFS, la réplication minimise la surcharge de transfert de données tout en garantissant l'intégrité des données.

2. Setup storage device

Pour activer cette fonctionnalité, nous devons préparer un nouveau disque. Dans VMWare, nous avons choisi d'ajouter un disque dur SCSI à chaque nœud, qui doit être redémarré. Pour vérifier si le nouveau périphérique a été ajouté avec succès, il suffit de faire un « fdisk -l ». Retour à l'interface graphique de Proxmox, depuis le contexte d'un nœud -> Disks / ZFS -> Create ZFS.



Dans l'image ci-dessous, nous pouvons voir la fenêtre de configuration minimale pour créer un pool ZFS : ici, nous sélectionnons le disque nouvellement ajouté, sélectionnons son niveau RAID, la compression, ashift. Dans un environnement d'entreprise, une exploration plus approfondie des options RAID doit être effectuée : le miroir serait la meilleure option pour obtenir une redondance, en sélectionnant 2 ou plusieurs périphériques de stockage qui refléteront chacun les données stockées. Dans notre environnement de test virtuel, après avoir déployé un seul disque, cette option doit être laissée par défaut.

Les algorithmes de compression ont un impact important sur les performances et les ressources - l'option par défaut de l'algorithme LZ4 est la meilleure pour un équilibre.

La documentation de Proxmox n'entre pas dans les détails sur « ashift », suggérant fortement de le laisser comme valeur par défaut.

Create: ZFS

Name: RAID Level:

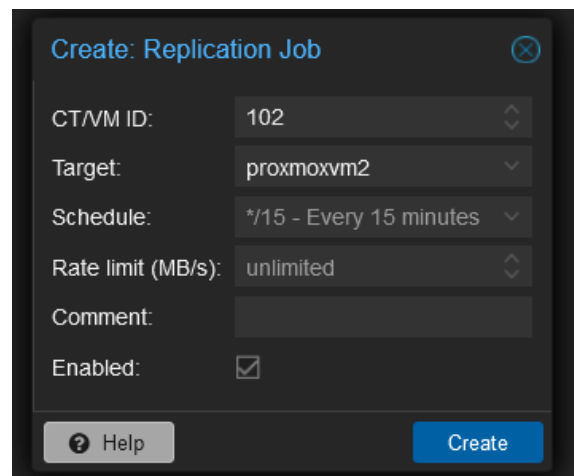
Add Storage: ☒ Compression:

ashift:

<input checked="" type="checkbox"/> Device ↑	Model	Serial	Size	Order
<input checked="" type="checkbox"/> /dev/sda2			1.02 KB	

3. Schedule replication jobs

Depuis l'interface graphique Proxmox, dans l'un des contextes de nœuds -> Replication -> Add.



Cela créera une tâche de réplication qui, toutes les 15 minutes, répliquera le stockage matériel du CT/VM désigné vers le nœud cible. Lorsque le CT/VM doit être migré pour diverses raisons, le processus est accéléré en s'assurant que des copies fiables de ses données sont déjà en place sur le nœud cible.

High-Availability Manager

1. Definition, scope

Le gestionnaire de haute disponibilité (HA Manager) de Proxmox VE est une infrastructure robuste qui garantit un accès ininterrompu aux charges de travail virtualisées en orchestrant des mécanismes de basculement au sein d'un environnement en cluster. Il permet aux services et aux machines virtuelles de migrer ou de redémarrer automatiquement sur des nœuds sains en cas de panne matérielle ou de problème système, minimisant ainsi les temps d'arrêt et préservant la continuité des activités.

2. Groups

Le principe de fonctionnement de HA est de séparer les CT/VM en groupes configurables :

- où nous sélectionnons quels nœuds assureront la continuité et avec quelle priorité
- En cas de panne d'un nœud et de déplacement de ses CT/VM, lors de son redémarrage, les ressources sont par défaut déplacées vers le nœud d'origine. Le paramètre « nofailback » protège contre un nœud qui redémarre sans cesse, par exemple. Sans « nofailback », nous aurions beaucoup de transferts dans les deux sens.

Pour créer un groupe, accédez au contexte du centre de données -> HA / Groups -> Create.

Create: HA Group ⓧ

ID: restricted: ☐

nofailback: ☒

Comment:

<input checked="" type="checkbox"/> Node ↑	Memory usage %	CPU usage	Priority
<input checked="" type="checkbox"/> proxmoxvm	0.0 %		10
<input checked="" type="checkbox"/> proxmoxvm2	66.5 %	2.2% of 1 CPU	1

3. Managed resources

Après avoir créé les groupes, nous devons sélectionner chaque ressource individuelle (CT/VM) que nous souhaitons affecter à un groupe géré par HA. Cela peut être fait à partir du contexte du centre de données -> HA -> Add. Un état de demande « démarré » indique au gestionnaire HA de faire l'effort d'essayer de maintenir la ressource en ligne pendant le transfert, et si ce n'est pas le cas, de la démarrer au moins automatiquement une fois le transfert terminé.

Add: Resource: Container/Virtual Machine ⓧ

VM: ✕ ▼ Group: ▼

Max. Restart: ⬆ ⬇ Request State: ▼

Max. Relocate: ⬆ ⬇

Comment: