



Proxmox VE — TSSR

Support de cours Proxmox VE pour techniciens systèmes et réseaux : installation, comptes, conteneurs LXC, VM KVM, réseau et stockage, cluster, Ceph, haute disponibilité, pare-feu et sauvegarde.

Ce support est la propriété de Dawan France.

Chapitre 1 — Introduction, Installation et LXC

Ce chapitre part d'un serveur sur lequel Proxmox VE vient d'être installé et se termine sur un site web servi par un conteneur, qui répond depuis le nœud. Entre les deux, chaque notion arrive au moment où elle sert. D'abord pourquoi on virtualise, et ce que Proxmox ajoute à la Debian sur laquelle il repose.

Puis ce qu'il faut régler sur un nœud neuf avant de s'en servir. Enfin, comment faire tourner un premier service dans un conteneur LXC.

Le fil conducteur est votre nœud de lab. Vous le découvrez tel que l'installateur l'a laissé. Vous le basculez sur le dépôt sans abonnement et le mettez à jour.

Vous y créez des comptes aux droits limités, et sécurisez leur accès par la double authentification et les jetons d'API. Puis vous y déployez un conteneur Debian qui héberge nginx.

Objectifs du chapitre

À l'issue de ce chapitre, les stagiaires seront capables de :

- **Expliquer** l'intérêt de la virtualisation, distinguer hyperviseur type 1/2 et conteneurisation, et situer Proxmox VE (histoire, chiffres, marché)
- **Installer** Proxmox VE sur un nœud, réaliser la post-installation et prendre en main l'interface web
- **Distinguer** les realms `pam` et `pve` et déléguer des droits limités. Protéger aussi les accès par la double authentification TOTP (*Time-based One-Time Password*, code à usage unique calculé sur l'heure) et les jetons d'API
- **Organiser** les ressources en pools, et déléguer des droits sur un pool
- **Créer et administrer** un conteneur LXC Debian 13, y déployer un service nginx accessible sur le réseau, et savoir où placer une image Docker

Ce support est la propriété de Dawan France.

1. Pourquoi virtualiser ?

1.1 Le problème que la virtualisation résout

Pendant longtemps, chaque service informatique vivait sur sa propre machine physique. Un serveur pour la messagerie, un pour la base de données, un pour le site web.

Résultat : des salles machine bondées de serveurs sous-utilisés, et des coûts d'électricité et de refroidissement disproportionnés. Surtout, un gaspillage colossal de ressources : un serveur physique typique tournait en moyenne à 10 à 15 % de sa capacité CPU.

La virtualisation répond à cette absurdité en permettant à plusieurs systèmes d'exploitation de coexister sur un seul matériel physique. Chaque système croit disposer de sa propre machine, alors qu'il partage en réalité les ressources avec ses voisins. Cette illusion est produite par une couche logicielle — l'hyperviseur — qui s'intercale entre le matériel et les systèmes d'exploitation invités.

Mais réduire la virtualisation à « faire des économies de serveurs » serait passer à côté de l'essentiel. Ce qui a vraiment transformé le métier d'administrateur, ce sont les propriétés qui viennent avec le fait qu'une machine n'est plus qu'un ensemble de fichiers :

- **L'isolation des pannes.** Chaque service vit dans sa propre machine virtuelle. Un serveur applicatif sature sa mémoire, une mise à jour tourne mal : le problème reste confiné à sa VM.

  Il n'emporte pas les voisins, qui partagent pourtant le même matériel.

- **L'instantané et le retour arrière.** Avant une mise à jour risquée, on prend un *snapshot* ; si ça se passe mal, on revient à l'état d'avant en quelques secondes. Sur une machine physique, il aurait fallu une réinstallation.

- **La maintenance sans interruption.** Une VM peut être déplacée d'un serveur à un autre *pendant qu'elle tourne* (migration à chaud). On vide ainsi un serveur pour le mettre à jour ou remplacer une barrette de RAM, sans que les utilisateurs ne s'en aperçoivent.

- **La reproductibilité.** Une machine virtuelle se clone, se transforme en modèle, se déploie en série.

Dawan
Ce support est la propriété de Dawan France.

Monter dix serveurs identiques devient l'affaire de quelques minutes, là où l'installation manuelle en demandait des heures. C'est le rôle des *templates* (modèles de machines) et de cloud-init (l'outil qui personnalise une machine à son premier démarrage).

- **La reprise après sinistre.** Puisqu'une VM est un ensemble de fichiers, la sauvegarder et la restaurer ailleurs — sur un autre site, après un incident — devient une opération standard, et non un projet.

C'est cette combinaison — densité **et** souplesse d'exploitation — qui explique que la quasi-totalité des serveurs d'entreprise tournent virtualisés. Proxmox VE est l'une des plateformes qui rendent tout cela accessible sans licence propriétaire, et c'est ce que ce chapitre commence à mettre en place.

1.2 Les trois grandes familles

La virtualisation ne se présente pas sous une forme unique. Trois familles coexistent, chacune avec ses compromis.

L'hyperviseur de type 1 — aussi appelé "bare metal" — s'installe directement sur le matériel, sans système d'exploitation hôte intermédiaire. Il est le premier programme à s'exécuter au démarrage, bien avant tout système invité.

Cette position privilégiée lui donne un accès direct aux ressources physiques, ce qui se traduit par des performances proches du natif. VMware ESXi, Microsoft Hyper-V, et Proxmox VE appartiennent à cette famille.

L'hyperviseur de type 2 fonctionne différemment : il s'installe comme une application dans un système d'exploitation hôte déjà en place. VirtualBox que vous avez peut-être déjà utilisé sur votre poste, ou VMware Workstation, sont des hyperviseurs de type 2. Pratiques pour le poste de développeur ou la formation, ils introduisent une couche supplémentaire qui dégrade légèrement les performances et les rend inadaptés à la production.

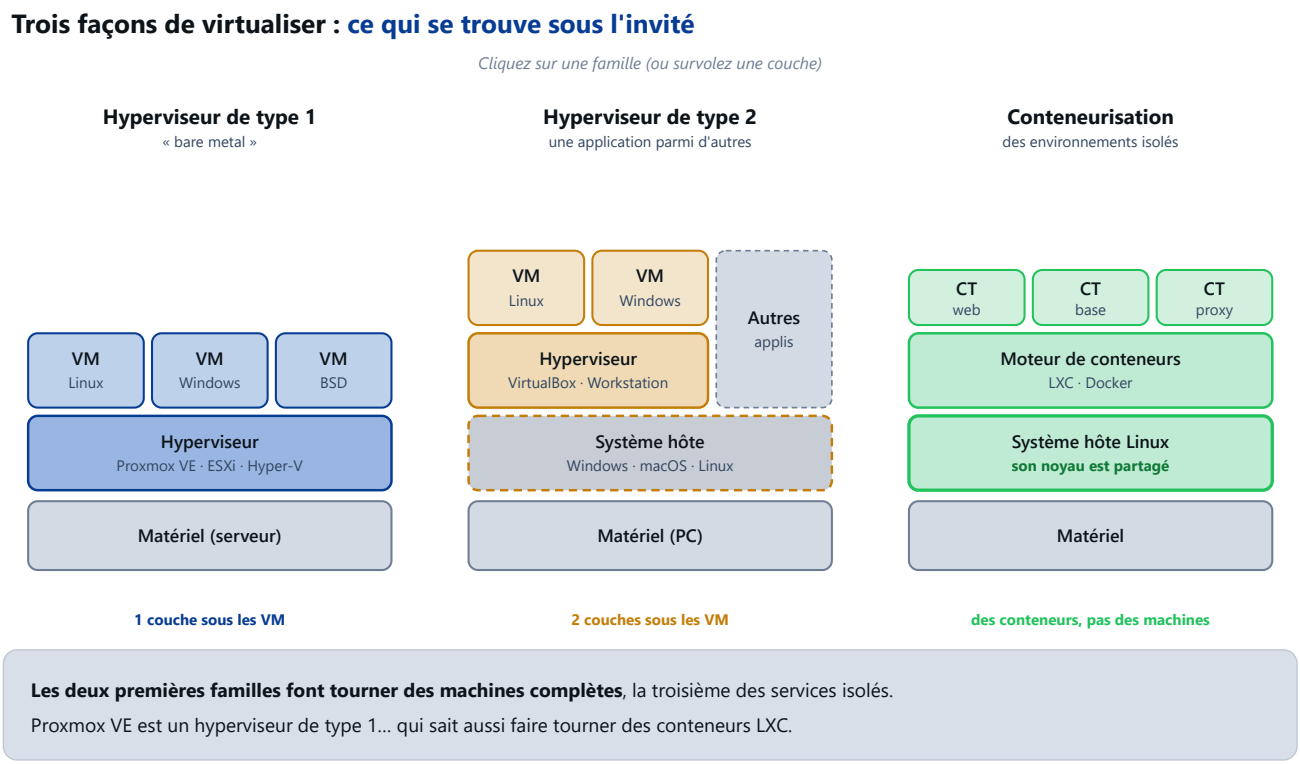
La conteneurisation adopte une approche radicalement différente : plutôt que d'émuler un matériel complet, elle partage le noyau du système hôte entre les environnements isolés. Chaque conteneur ne contient que les binaires et bibliothèques dont il a besoin — pas de noyau, pas de firmware, pas d'émulation matérielle.

LXC (Linux Containers) et Docker sont les représentants les plus connus. Le gain en densité et en vitesse de démarrage est spectaculaire.

Dawan
Ce support est la propriété de Dawan France.

Il a un prix : tous les conteneurs partagent le même noyau Linux. Impossible, donc, de faire tourner un système Windows dans un conteneur.

Le schéma ci-dessous empile les trois familles couche par couche. Cliquez sur l'une d'elles pour afficher ses exemples et son compromis. On y lit d'un coup d'œil la couche supplémentaire du type 2, et le système hôte que tous les conteneurs se partagent.



Proxmox VE se range dans la première famille, et fait aussi tourner des conteneurs LXC : c'est ce qui le distingue.

Pour chaque système invité, deux questions décident de la famille : sur quoi repose-t-il, et partage-t-il le noyau avec ses voisins ?

Terme	Ce que c'est	Dans le laboratoire
Matériel	Le serveur physique : processeurs, mémoire, disques, cartes réseau	Votre nœud `pxe2<N>1`, lui-même une machine virtuelle de l'infrastructure Dawan (virtualisation imbriquée)
Hyperviseur de type 1	Le premier logiciel lancé au démarrage, directement sur le matériel ; il partage ce matériel entre les machines virtuelles	Proxmox VE, installé sur votre nœud (§4)

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
Machine virtuelle	Un ordinateur complet simulé, avec **son propre noyau** : Linux, Windows, BSD (famille de systèmes Unix libres)…	Aucune à ce stade : le nœud ne porte que des conteneurs
Hyperviseur de type 2	Une application posée sur un système hôte (Windows, macOS, Linux) : une couche de plus	Absent du laboratoire ; c'est VirtualBox ou VMware Workstation sur un poste
Noyau	Le cœur du système, qui pilote le matériel et les processus	Le noyau `-pve` du nœud (`uname -r`, §3.1)
Conteneur	Un environnement isolé qui **partage le noyau** de l'hôte : rapide et léger, mais Linux seulement	Les conteneurs LXC du §11, dont le CT 200 `cache-apt` de l'atelier 6

Un défaut du noyau de l'hôte touche tous les conteneurs du nœud : c'est la limite des conteneurs. Une VM a son propre noyau : si elle plante, ses voisines continuent de tourner.

Où voir chaque famille sur votre nœud :

Notion	Où la voir
Le type 1	L'installateur du §4.1 s'installe sur le disque nu, sans système en dessous. `pveversion`, dans le **Shell** du nœud, donne sa version
Les VM et les conteneurs	L'arborescence à gauche de l'interface web : icône d'écran pour une VM, icône de cube pour un conteneur
Leur liste en ligne de commande	`qm list` (les VM) et `pct list` (les conteneurs), dans le **Shell** du nœud
Le noyau partagé	`uname -r` dans le Shell du nœud, puis `pct exec 200 -- uname -r` une fois le CT 200 créé : la même version, car c'est le même noyau

À retenir : la virtualisation fait tourner plusieurs systèmes sur un seul matériel, au lieu d'un serveur sous-utilisé par service. Trois familles. L'hyperviseur de **type 1** : sur le matériel, pour la production.

Le **type 2** : sur un système hôte, pour le poste de travail. L'**isolation** par conteneurs : un noyau partagé.

Ce support est la propriété de Dawan France.

2. Proxmox VE : présentation

2.1 Proxmox VE, en deux mots

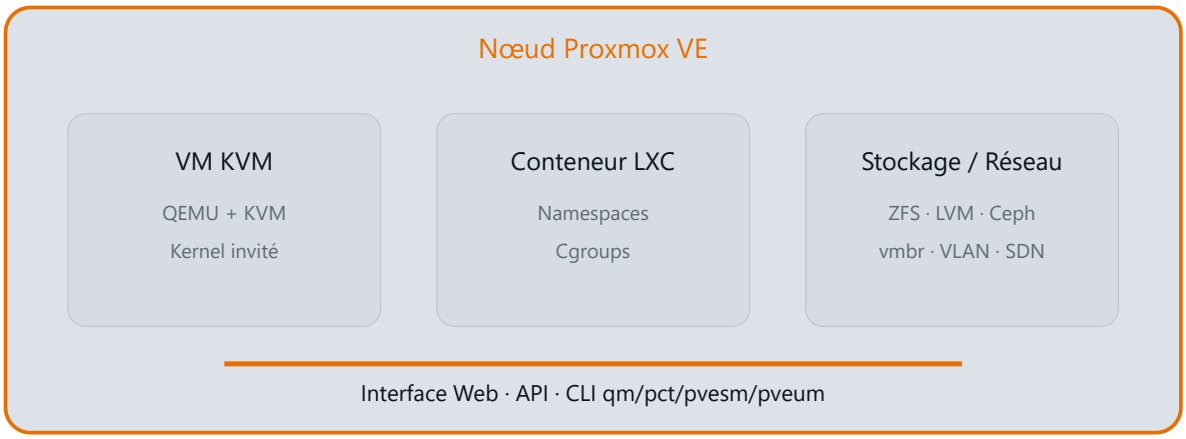
Avant d'ouvrir le capot, posons simplement ce qu'est Proxmox VE et à quoi il sert.

Proxmox Virtual Environment (VE) est une plateforme de virtualisation open source qui repose sur Debian Linux. Sur un même serveur, il fait tourner deux sortes d'invités, et c'est tout l'intérêt. Des **machines virtuelles complètes** (via KVM/QEMU — QEMU, *Quick Emulator*, simule le matériel de la machine) exécutent n'importe quel système : Linux, Windows, BSD, un pare-feu, une *appliance*.

Des **conteneurs légers** (via LXC) partagent le noyau de l'hôte et démarrent en une seconde. On choisit l'un ou l'autre selon le besoin ; les deux se pilotent exactement de la même façon.

Proxmox est un **hyperviseur de type 1** : il s'installe directement sur le matériel, il est le premier logiciel à démarrer, et rien ne s'interpose entre lui et le processeur. C'est ce qui lui donne des performances proches du natif, à la différence d'un VirtualBox posé sur un Windows (type 2).

Architecture Proxmox VE



Un seul nœud porte trois familles de services. Les **VM KVM** : QEMU et KVM, avec leur propre noyau. Les **conteneurs LXC** : namespaces et cgroups.

Le **stockage et le réseau** : ZFS, LVM, Ceph ; ponts `vmbr`, VLAN, SDN. Tout se pilote par la même porte d'entrée : l'interface web, l'API et les commandes `qm`, `pct`, `pvesm`, `pveum`.

Ce support est la propriété de Dawan France.

Trois idées résument sa philosophie, et ce sont elles qui expliquent le reste du chapitre :

- **Tout-en-un.** Calcul (VM et conteneurs), stockage (local, ZFS, NFS, iSCSI, Ceph), réseau, sauvegarde, haute disponibilité, pare-feu : tout est intégré, sans brique payante à ajouter. C'est ce qu'on appelle une infrastructure *hyperconvergée*.
- **Administrable comme on préfère.** Une **interface web** claire suffit à tout gérer — VM, conteneurs, stockage, cluster — depuis n'importe quel nœud.

  Les administrateurs avancés disposent d'une **ligne de commande** (`qm`, `pct`, `pvesm`, `pvesh`) et d'une **API REST** complète. Ce que fait la souris, un script peut le faire.
- **Simple à prendre en main.** La conception vise la simplicité : on peut installer Proxmox et démarrer une première machine en quelques minutes, sur un seul serveur, sans cluster ni configuration réseau élaborée. On assemble un cluster plus tard, quand le besoin apparaît.

Le logiciel est **entièrement gratuit et open source**, sous licence AGPL v3 (*Affero General Public License*, licence libre). Il n'y a pas de version bridée : le cluster, la haute disponibilité, Ceph, la sauvegarde sont tous inclus. L'abonnement payant, facultatif, n'achète pas des fonctionnalités.

Il achète l'accès au dépôt *enterprise* (paquets testés pour la production) et le support de l'éditeur. C'est une différence de nature avec le modèle des solutions propriétaires.

Le tableau ci-dessous situe Proxmox VE face à VMware vSphere, la solution propriétaire de référence.

Critère	Proxmox VE 9	VMware vSphere / ESXi
Modèle	Open source, support payant facultatif	Propriétaire, licence payante
Virtualisation	KVM (VM) **et** LXC (conteneurs)	ESXi (VM), conteneurs via add-on
Administration	Web, CLI, API REST	vCenter Server, client web
Stockage	local, LVM, ZFS, Ceph, NFS, iSCSI	VMFS (système de fichiers de VMware), NFS, iSCSI, vSAN
Haute disponibilité	Cluster, règles d'affinité, migration à chaud	vSphere HA, DRS (*Distributed Resource Scheduler*, répartition automatique de la charge), vMotion
Cible typique	PME (petites et moyennes entreprises), ESN (entreprises de services du numérique), datacenters open source	Grandes entreprises, parcs très étendus

Ce support est la propriété de Dawan France.

Ce que ce tableau ne dit pas encore, nous le construirons pas à pas. La suite de cette section présente l'origine du projet, ce que la plateforme sait faire, et où elle est — ou n'est pas — le bon choix. La section suivante ouvrira ensuite le capot : comment Proxmox est bâti sur Debian, et comment fonctionne son moteur de VM.

2.2 Un peu d'histoire et de positionnement

Le projet Proxmox VE démarre en **2007**, édité par la société autrichienne **Proxmox Server Solutions GmbH**.

Le logiciel est publié sous licence **GNU AGPL v3** (*GNU Affero General Public License*, du projet GNU de logiciel libre). Le code source est donc entièrement disponible, et toute modification doit elle-même être publiée sous les mêmes termes.

Le modèle économique repose sur les abonnements de support — pas sur le logiciel lui-même. Proxmox VE se télécharge, s'installe et s'utilise gratuitement ; l'abonnement apporte l'accès aux dépôts testés pour la production et au support professionnel.

Les étapes clés de son histoire :

Étape	Ce qu'elle a apporté
pmxcfs et **API REST**	Le système de fichiers de cluster, et l'interface de programmation qui pilote tout
LXC (version 4.0)	Les conteneurs LXC remplacent OpenVZ
Gestionnaire HA	La haute disponibilité, simple à mettre en place
ZFS et **Ceph**	Le support intégré de ces deux stockages
Proxmox Backup Server (2020)	Un serveur de sauvegarde dédié
Version 8.2	L'installation automatisée, et un assistant d'importation des machines d'autres hyperviseurs, comme VMware ESXi

2.3 Ce que Proxmox sait faire — les fonctionnalités clés

Avant d'entrer dans le détail de Proxmox, posons les fonctionnalités qu'un hyperviseur professionnel doit couvrir. Proxmox VE coche toutes ces cases, et nous les aborderons progressivement au fil des chapitres.

Dawan
Ce support est la propriété de Dawan France.

La **haute disponibilité** : si un nœud tombe, les VMs qui s'y trouvent redémarrent automatiquement sur un autre nœud du cluster. Proxmox l'implémente avec son propre gestionnaire HA (`ha-manager`, services `pve-ha-crm` et `pve-ha-lrm`).

Celui-ci s'appuie sur Corosync pour savoir quels nœuds sont vivants. Pas de Pacemaker, contrairement à beaucoup de clusters Linux classiques.

La **gestion du stockage** : au-delà du stockage local, une plateforme de virtualisation doit s'interfacer avec NFS, iSCSI, Ceph, et d'autres systèmes de stockage partagé. Proxmox supporte nativement l'ensemble de ces protocoles.

La **sauvegarde et restauration** : protéger les VMs et conteneurs est non négociable en production. Proxmox VE intègre un gestionnaire de sauvegardes, et Proxmox Backup Server pousse cette capacité beaucoup plus loin avec la déduplication et la compression.

Les **templates et cloud-init** : créer des VMs répétitives manuellement n'est pas envisageable à l'échelle. Les templates et le support cloud-init permettent de déployer des VMs préconfigurées en quelques secondes.

La **gestion du réseau** : SDN (Software-Defined Networking), VLAN, bridges, bonds — Proxmox offre une gestion réseau complète, avec une couche SDN dédiée depuis la version 7.

2.4 Les avantages de Proxmox VE, et où trouver de l'aide

Avantage	Ce qu'il veut dire
Logiciel open source	Le code est publié (AGPL v3) : chacun peut le lire, le vérifier, le corriger
Pas de dépendance à un fournisseur unique	*No vendor lock-in* : l'infrastructure évolue sans renégocier une licence
Noyau Linux	Un socle Debian, connu et documenté
Installation rapide, prise en main simple	Un premier nœud et une première machine en quelques minutes
Interface de gestion web	Tout se gère depuis un navigateur, sur n'importe quel nœud
API REST	Ce que fait l'interface, un script peut le faire
Une vaste communauté active	Forum, wiki, documentation
Faibles coûts d'administration, déploiement simple	Aucune fonction n'est payante ; l'abonnement paie le support

Dawan Nous formons votre talent
Ce support est la propriété de Dawan France.

Trois d'entre eux méritent un mot de plus.

Le coût, mais pas de la façon qu'on croit. Proxmox VE est intégralement fonctionnel sans licence : aucune fonctionnalité n'est bridée dans la version gratuite — ni le cluster, ni la haute disponibilité, ni Ceph, ni la sauvegarde.

L'abonnement payant n'achète pas des fonctions, il achète l'accès au **dépôt enterprise** (paquets validés en amont) et le **support de l'éditeur**. C'est une différence de nature avec le modèle VMware, où le prix conditionne les fonctionnalités.

Une pile ouverte de bout en bout. KVM, LXC, ZFS, Ceph, Corosync, Debian : chaque brique est un projet libre documenté, utilisable et diagnostiquable en dehors de Proxmox. Quand quelque chose ne va pas, on peut descendre au niveau du composant — `journalctl -u corosync`, `ceph health detail` — plutôt que d'attendre un ticket de support.

Tout est administrable en ligne de commande. L'interface web n'est qu'un client de l'API REST : `pvesh`, `qm`, `pct` et `pvesm` font tout ce que fait la souris. C'est ce qui rend l'automatisation possible sans surcouche propriétaire.

Obtenir de l'aide : les cinq canaux.

Canal	Ce qu'on y trouve
Wiki Proxmox VE	La source principale d'information : documentation de référence de chaque fonction
Forum de support communautaire	Échanges entre utilisateurs, modérés par l'équipe de support Proxmox
Listes de diffusion	Pour communiquer rapidement avec la communauté
Support commercial	Proxmox Server Solutions GmbH, par abonnement, avec des temps de réponse garantis
Bug Tracker	Outil public pour signaler un problème ou proposer une amélioration

À retenir : Proxmox VE est une plateforme de virtualisation **open source** bâtie sur Debian. Elle fait tourner des **VM KVM** et des **conteneurs LXC**, gérés depuis une interface web, avec cluster, haute disponibilité, stockage distribué et sauvegarde intégrés. Le logiciel est gratuit ; l'abonnement paie le support et le dépôt *enterprise*.

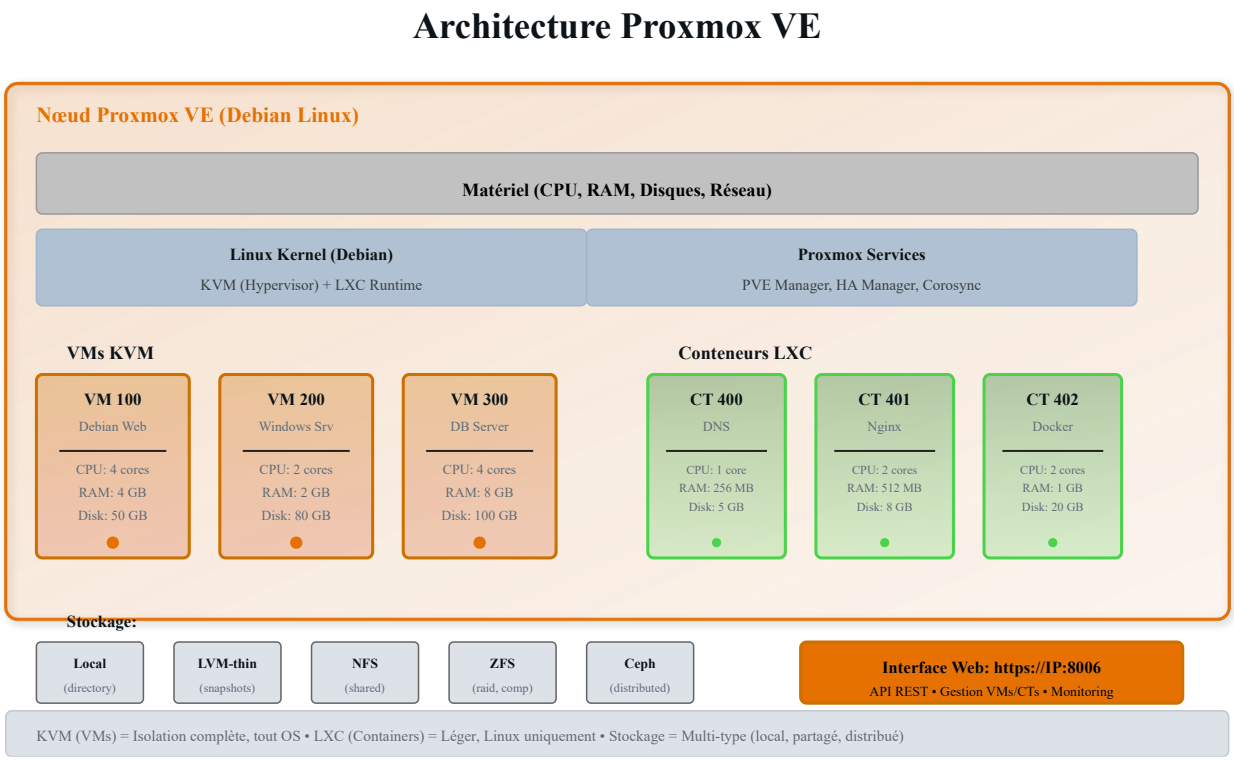
Dawan
Ce support est la propriété de Dawan France.

3. Sous le capot : la pile logicielle et le moteur des VM

Proxmox est maintenant présenté : ce qu'il fait, d'où il vient, où il est le bon choix. Cette section ouvre le capot — de quoi la plateforme est faite, brique par brique, et comment le moteur KVM/QEMU exécute les machines virtuelles.

3.1 La pile logicielle

Proxmox VE n'est pas un système d'exploitation écrit de zéro. C'est une surcouche spécialisée construite sur Debian : l'écosystème Debian est disponible, et un administrateur Linux s'y retrouve immédiatement.



Les composants s'empilent ainsi :

Le noyau : userland Debian, noyau Proxmox. Les paquets et les outils sont ceux de **Debian** (Debian 13 « Trixie » pour Proxmox VE 9).

Mais le **noyau** est fourni par Proxmox, dérivé d'un noyau Ubuntu récent. Il apporte des pilotes plus récents pour le matériel neuf, des correctifs pour KVM et LXC, et le module **ZFS** intégré, ce

Ce support est la propriété de Dawan France.

qui permet d'installer le système directement sur ZFS. On le reconnaît au suffixe `-pve` que répond `uname -r`.

Conséquence pratique, le jour où l'on installe Proxmox sur une Debian déjà en place : il faut **retirer le noyau Debian** (`linux-image-*`). Ce noyau peut entrer en conflit avec le paquet `proxmox-ve` lors des mises à jour. Sur un nœud installé depuis l'ISO, c'est déjà réglé.

QEMU/KVM gère les machines virtuelles. Chaque VM tourne comme un processus QEMU, avec accélération KVM ; sa configuration est dans `/etc/pve/qemu-server/<vmid>.conf`.

LXC (Linux Containers) gère les conteneurs : des conteneurs système, avec leur propre init, leurs services et leur réseau, qui partagent le noyau de l'hôte.

pmxcfs est le système de fichiers de Proxmox : il stocke toute la configuration de la plateforme dans `/etc/pve/`, et la réplique entre les nœuds d'un cluster. Même sur un nœud seul, il est présent et actif.

Corosync assure la communication entre les nœuds d'un cluster. Sur un nœud seul, il est installé mais ne tourne pas : `systemctl status corosync` affiche `inactive (dead)`, ce n'est pas une panne.

L'API REST et l'interface web (`pveproxy`) donnent accès à toutes les fonctions par HTTPS. L'interface est construite au-dessus de cette API : ce qu'on fait à la souris se fait aussi avec `pvesh`.

- 📖 Documentation officielle Proxmox VE — guide d'administration

3.2 Sous le capot : KVM/QEMU et le prérequis processeur

Maintenant que Proxmox est présenté, ouvrons le capot du côté des machines virtuelles. **KVM — Kernel-based Virtual Machine — est un module du noyau Linux, pas un logiciel externe.** Depuis sa fusion dans le noyau Linux 2.6.20 en 2007, KVM transforme n'importe quel noyau Linux en hyperviseur.

Il ne réinvente pas la gestion de la mémoire, des interruptions et de l'ordonnancement. Il réutilise tout ce que le noyau sait déjà faire, et y ajoute la capacité d'isoler des contextes d'exécution. C'est le moteur des VM de Proxmox ; les conteneurs LXC, eux, n'en dépendent pas.

Pour fonctionner, KVM s'appuie sur les extensions matérielles de virtualisation intégrées dans les processeurs modernes. Intel les appelle **VT-x** (*Virtualization Technology*, pour ses processeurs IA

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

— *Intel Architecture* — 32 et 64 bits). AMD (l'autre fabricant de processeurs x86) les appelle **AMD-V** (ou SVM — Secure Virtual Machine).

Ces extensions permettent au processeur de basculer rapidement entre le contexte de l'hyperviseur et celui des machines virtuelles. Les instructions sensibles (accès matériel, gestion de la mémoire) ne peuvent pas "s'échapper" vers le matériel réel.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis le Shell du noeud Proxmox — pas depuis votre poste Windows
# Verifier que les extensions de virtualisation sont presentes ; -m 1 : la premiere ligne trouvee
grep -m 1 -E 'vmx|svm' /proc/cpuinfo

# vmx = Intel VT-x
# svm = AMD-V
```

Résultat attendu — sur un processeur Intel :

```
root@pve1:~# grep -m 1 -E 'vmx|svm' /proc/cpuinfo
flags       : ... vmx sse4_1 sse4_2 x2apic ...
```

Sur un processeur AMD, c'est `svm` qui apparaît à la place :

```
root@pve1:~# grep -m 1 -E 'vmx|svm' /proc/cpuinfo
flags       : ... svm sse4a abm ...
```

Une sortie vide est le seul cas préoccupant : la virtualisation matérielle est désactivée dans le BIOS (*Basic Input/Output System*) ou l'UEFI (*Unified Extensible Firmware Interface*). C'est le micrologiciel de démarrage du serveur.

Proxmox ne pourra alors démarrer aucune machine virtuelle KVM. Les conteneurs LXC, eux, continueront de fonctionner : ils ne s'appuient pas sur ces instructions.

> **Où taper cette commande ?** Dans le **Shell du nœud Proxmox** — sélectionnez votre serveur dans la colonne de gauche, onglet **Shell**. `/proc/cpuinfo` est un fichier Linux : la commande n'existe pas sur votre poste Windows.

Dawan Nous formons votre savoir
Ce support est la propriété de Dawan France.

> Pour faire la même vérification **sur votre poste Windows** — utile le jour où VirtualBox refuse de démarrer une machine :
>
> - `Ctrl` + `Maj` + `Échap` → onglet **Performance** → **Processeur** → ligne **Virtualisation**
> - ou en PowerShell : `Get-CimInstance Win32_Processor | Select-Object Name, VirtualizationFirmwareEnabled`
>
> Attention au cas particulier Windows : si Hyper-V, WSL2 (*Windows Subsystem for Linux*) ou la sécurité VBS (*Virtualization-Based Security*) sont actifs, ils monopolisent la virtualisation matérielle. VirtualBox ne peut alors plus l'utiliser, même quand le BIOS l'autorise. `systeminfo` l'indique par la ligne *Un hyperviseur a été détecté*.

Si la commande ne renvoie rien, les extensions de virtualisation sont soit absentes du processeur (rare sur le matériel récent), soit désactivées dans le BIOS/UEFI. Proxmox affichera une erreur au démarrage des VMs dans ce cas. Sur nos ateliers, le matériel a été vérifié en amont — les extensions sont actives.

KVM seul ne suffit pas pour créer une machine virtuelle complète : il gère la virtualisation CPU et mémoire, mais pas l'émulation des périphériques (carte réseau, disque, clavier). C'est QEMU qui prend en charge cette émulation. Dans Proxmox, KVM et QEMU travaillent toujours en tandem — vous entendrez d'ailleurs parfois parler de "KVM/QEMU" comme d'un seul bloc.

À retenir : Proxmox est une **surcouche de Debian** : son noyau, `pmxcfs` (la configuration partagée dans `/etc/pve`), `pveproxy` (l'interface web sur 8006) et les outils `qm` / `pct`. Les VM passent par **KVM**, un module du noyau, et **QEMU** ; KVM exige les extensions **VT-x** ou **AMD-V** du processeur (`grep -E 'vmx|svm' /proc/cpuinfo`).

Ce support est la propriété de Dawan France.

4. Installation et déploiement

4.1 Installation bare-metal : l'installateur graphique

L'installation bare-metal de Proxmox VE est la méthode recommandée pour la production. L'ISO officielle se télécharge sur `enterprise.proxmox.com/iso/`. Son installateur graphique prend en charge tout le processus : partitionnement, installation du système de base Debian, configuration du noyau PVE, mise en place de pmxcfs et de l'interface web.

Dans nos ateliers, cette étape a déjà été réalisée via un déploiement PXE (*Preboot Execution Environment*, démarrage par le réseau) automatisé fourni pour la session. Chaque stagiaire dispose d'un nœud Proxmox VE 9 opérationnel depuis le démarrage de la formation — c'est le seul élément de l'infrastructure présent à ce stade. Les composants supplémentaires (stockages réseau, cluster, Ceph, PBS) seront ajoutés progressivement, à mesure que les chapitres en ont besoin.

Vous arrivez donc avec un Proxmox VE 9 fonctionnel, accessible par l'URL que le formateur vous donne (`https://pxe2<N>1.<ville>.pm.dawan.ovh`). Il est néanmoins important de comprendre ce que l'installateur a fait sous le capot.

L'installateur demande quatre informations essentielles :

Le disque cible : sur quel disque installer Proxmox. L'installateur propose un choix de système de fichiers pour la partition racine : ext4, xfs, ZFS ou btrfs. ZFS s'installe seul ou en RAID (*Redundant Array of Independent Disks*, grappe de disques) : RAID0 (agrégat sans redondance), RAID1 (miroir), RAID10 (miroirs agrégés), RAIDZ (parité répartie).

Par défaut, en installation graphique, le disque est partitionné avec ext4 pour `/`. L'espace restant va à un groupe LVM nommé `pve`, qui contient un volume LVM-thin nommé `data` : c'est votre stockage `local-lvm`.

La configuration réseau : adresse IP, masque, passerelle, DNS. Cette configuration est appliquée à l'interface de management, qui deviendra le bridge `vmbr0` dans Proxmox.

Le mot de passe root et l'adresse email de l'administrateur (utilisée pour les alertes système).

Le nom d'hôte (FQDN : *Fully Qualified Domain Name*, nom complet avec le domaine) : par exemple `pve.dawan.local`.

Ce support est la propriété de Dawan France.

> **Erreur fréquente :** ne pas configurer un nom d'hôte pleinement qualifié (FQDN) lors de l'installation. Proxmox attend un FQDN (`pve.dawan.local` et non simplement `pve`). Un hostname mal configuré peut causer des problèmes dans un cluster.

Le menu de démarrage de l'ISO — ce qu'on choisit avant même d'installer

Avant l'installateur lui-même, le support de démarrage propose plusieurs entrées, graphiques ou en mode texte (UI : *User Interface*, interface utilisateur), et il faut savoir laquelle prendre selon le contexte. En salle vous n'y toucherez pas (le déploiement est automatisé), mais le jour où vous installez un vrai serveur, ce menu est le premier choix qui compte.

Entrée	Quand la choisir
Install Proxmox VE (Graphical)	Le cas normal, avec écran et clavier. C'est l'option recommandée.
Install Proxmox VE (Terminal UI)	Même installateur en mode texte — meilleure compatibilité avec du matériel très ancien ou très récent dont la carte graphique n'est pas reconnue.
Terminal UI, Serial Console	Serveur *headless* (sans écran), piloté par port série ou carte d'administration.
Advanced → Rescue Boot	Tente de démarrer une installation existante quand le *bootloader* est cassé — l'équivalent d'un mode secours.
Advanced → Test Memory	`memtest86+`, pour éliminer une RAM défectueuse avant de suspecter le reste (Secure Boot à désactiver).

À retenir : Terminal UI et Serial Console ne changent **rien** au résultat installé — c'est le même installateur, seule l'interface diffère. On les choisit pour une contrainte matérielle, pas par préférence.

Le choix du système de fichiers — la seule décision vraiment structurante

L'écran de sélection du disque cache, derrière son bouton **Options**, le choix qui vous suivra le plus longtemps : le système de fichiers de la racine. Il conditionne la présence de LVM, la disponibilité des snapshots au niveau du disque système, et la consommation mémoire du nœud.

Dawan
Ce support est la propriété de Dawan France.

Système de fichiers	Ce qu'il implique
ext4 (défaut)	Stable, éprouvé. L'installateur crée un groupe **LVM** et un pool **LVM-thin** `data` (votre futur `local-lvm`). Le choix par défaut, sans surprise.
xfs	Alternative à ext4, également sur LVM. À l'aise sur les très gros fichiers.
zfs	**Pas de LVM** : ZFS est à la fois le RAID logiciel, le gestionnaire de volumes et le système de fichiers. Snapshots natifs, détection de corruption — mais consomme plus de RAM (cache ARC — *Adaptive Replacement Cache*, le cache de lecture de ZFS en mémoire). C'est le bon choix **sans contrôleur RAID matériel**, car ZFS gère lui-même la redondance (miroir, RAID-Z).
btrfs	*Tech preview* : copy-on-write et snapshots, mais moins mûr que ZFS dans Proxmox. À réserver aux essais.

En **ext4/xfs**, l'installateur expose des options LVM utiles pour ne pas remplir tout le disque. `hdsize` fixe la taille totale utilisée ; on en réserve souvent pour l'étendre plus tard. Suivent `swapsize`, `maxroot` (plafond du volume `/`), `maxvz` (volume de données) et `minfree`.

En **ZFS**, on choisit à la place le niveau de RAID (`raid1` miroir, `raidz-1/2/3`), l'`ashift` (alignement des blocs sur le disque physique) et la compression (`lz4`, `zstd`…). Et surtout `arc-max`, le plafond de RAM du cache : un réglage à ne pas oublier sur un hyperviseur, sous peine de voir ZFS manger la mémoire des VM.

- 📖 Documentation Proxmox — Installation

4.2 Installation sur Debian existante

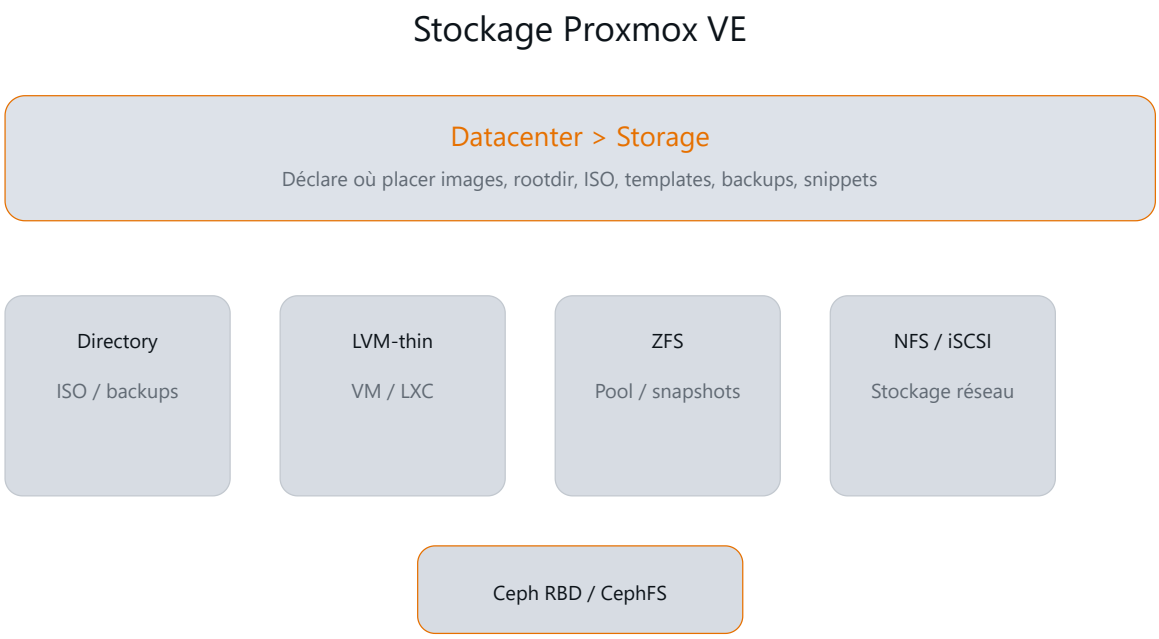
Proxmox VE peut également s'installer sur une Debian 13 Trixie existante, en ajoutant les dépôts Proxmox et en installant le métapaquet `proxmox-ve`. Cette méthode est moins courante en production : elle exige une Debian parfaitement configurée au préalable. Elle est utile dans certains scénarios : un serveur déjà en production que l'on souhaite convertir, des environnements cloud ou des VM.

La procédure officielle est documentée dans le wiki Proxmox. Elle implique notamment de configurer le hostname, d'ajouter la clé GPG (*GNU Privacy Guard*, outil de signature) de Proxmox, d'ajouter les dépôts, et d'installer `proxmox-ve`, `postfix` et `open-iscsi`. Le noyau Debian est remplacé par le noyau Proxmox au cours de ce processus.

Ce support est la propriété de Dawan France.

Dans le cadre de cette formation, nous ne pratiquerons pas cette installation — votre nœud est déjà opérationnel. L'essentiel à retenir est que cette possibilité existe, et que dans les deux cas (ISO ou sur Debian), le résultat final est identique.

4.3 Le stockage local par défaut



Après une installation standard, Proxmox configure automatiquement deux espaces de stockage locaux, visibles dans `Datacenter → Storage`.

`local` est un stockage de type "Directory" qui pointe vers `/var/lib/vz/`. Il stocke les ISOs téléchargées, les templates de conteneurs (CT templates), et les sauvegardes.

Ce type de stockage est basé sur le système de fichiers de la machine — ext4 ou XFS (système de fichiers journalisé) selon la partition racine. Il est simple, compatible avec tout, mais n'offre pas les performances d'un stockage bloc.

`local-lvm` est un stockage de type "LVM-Thin". Il s'appuie sur un pool LVM-thin créé dans le groupe de volumes `pve` lors de l'installation.

Ce stockage est destiné aux disques des VMs et des conteneurs. LVM-thin permet la création rapide de volumes (thin provisioning), les snapshots instantanés, et des performances proches du natif grâce à l'accès bloc direct.

La distinction entre stockage fichier et stockage bloc est fondamentale en virtualisation :

Ce support est la propriété de Dawan France.

Un **stockage fichier** (Directory, NFS) stocke les disques de VMs comme des fichiers image (`.qcow2`, `.raw`, `.vmdk`). L'avantage est la portabilité et la simplicité — un fichier `.qcow2` peut se copier, se déplacer, s'inspecter. L'inconvénient est une surcharge liée à la couche système de fichiers hôte.

Un **stockage bloc** (LVM-Thin, iSCSI, RBD — disque bloc de Ceph) attribue à chaque disque VM un volume bloc dédié. Il n'y a pas de système de fichiers intermédiaire — le disque VM lit et écrit directement sur le volume. Les performances sont meilleures, et les snapshots sont instantanés, mais les disques ne sont pas des fichiers transportables.

4.4 Présentation des stockages réseau

Proxmox VE supporte plusieurs protocoles de stockage réseau que nous présentons ici dans leur principe.

NFS (Network File System) permet de monter un partage réseau comme un répertoire local. Dans Proxmox, un stockage NFS peut héberger des ISOs, des sauvegardes et des images disque au format fichier. Simple à mettre en place, il convient bien aux environnements de taille modeste.

iSCSI transporte des commandes SCSI (*Small Computer System Interface*, protocole d'accès aux disques) sur IP. Un target iSCSI expose des LUNs (Logical Unit Numbers) que Proxmox peut voir comme des disques bloc.

On peut ensuite utiliser LVM sur ces disques pour créer des volumes VM. Les performances dépendent largement de la qualité du réseau et du stockage cible.

Ceph est une solution de stockage distribué open source qui se distingue fondamentalement des autres. Il n'y a pas de serveur central, pas de NFS, pas de SAN (*Storage Area Network*, réseau de stockage dédié). Les données sont réparties sur l'ensemble des nœuds du cluster.

Proxmox intègre Ceph nativement — vous pouvez déployer et gérer un cluster Ceph directement depuis l'interface web de Proxmox.

À retenir : deux façons d'installer : l'**ISO** et son installateur graphique, la voie de production ; une **Debian existante**, à laquelle on ajoute les paquets. Un nœud neuf crée deux stockages : `local` (fichiers) et `local-lvm` (disques en LVM-thin).

> **▶ À pratiquer maintenant — Atelier 1**

Dawan
Ce support est la propriété de Dawan France.

Installer Proxmox VE depuis l'ISO, écran par écran, dans une VM de votre nœud.

Ce support est la propriété de Dawan France.

5. Découverte de l'interface web

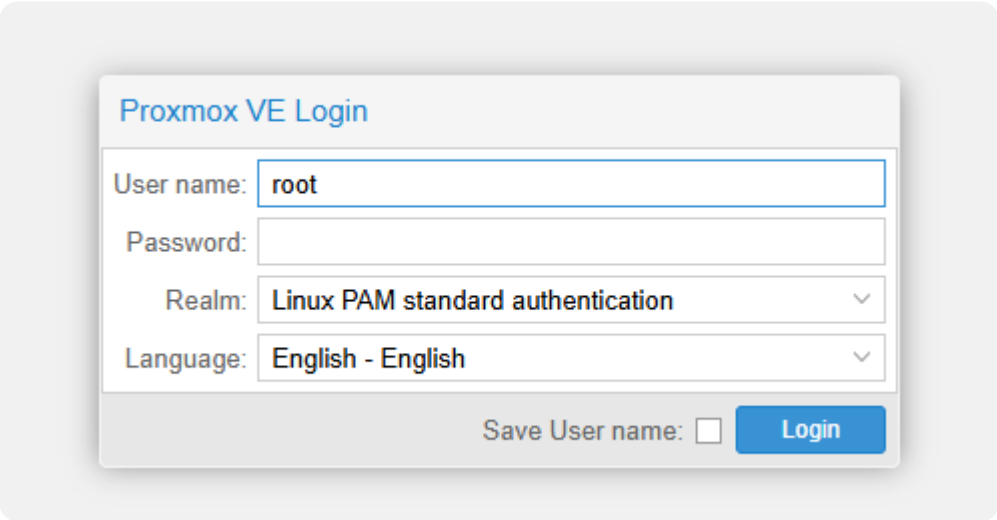
5.1 Connexion et structure générale

L'interface web de Proxmox VE est accessible à l'adresse `https://IP:8006`. Le port 8006 est le port par défaut de `pveproxy`, le serveur web intégré à Proxmox. Le certificat SSL (*Secure Sockets Layer*, chiffrement de la connexion) est auto-signé par défaut — votre navigateur affichera un avertissement que vous pouvez accepter.

À la connexion, le formulaire demande trois éléments : le nom d'utilisateur (par défaut `root`), le mot de passe et le **realm** (domaine d'authentification). Deux realms existent par défaut :

- **pam** : authentification Linux PAM — `root@pam` correspond au compte root du système Debian sous-jacent
- **pve** : authentification Proxmox interne — les comptes créés directement dans Proxmox

Une fois connecté, l'interface se divise en trois zones : l'arborescence à gauche, le panneau de contenu central, et les logs d'activité en bas. L'arborescence est hiérarchique : `Datacenter` → nom du nœud (par exemple `pve`) → VMs et conteneurs.



La fenêtre **Proxmox VE Login**, celle que vous verrez avant toute chose (nœud du lab, Proxmox VE 9.2.21). Quatre champs, dont deux qui décident de la suite.

- **`User name` et `Realm` vont ensemble.** Un compte Proxmox s'écrit toujours `utilisateur@realm` — ici `root@pam`.

Ce support est la propriété de Dawan France.

Le realm (domaine d'authentification) dit **où** Proxmox va vérifier le mot de passe : `Linux PAM standard authentication` interroge les comptes Linux du serveur, `Proxmox VE authentication server` interroge la base interne de Proxmox. Se tromper de realm donne un échec d'authentification alors que le mot de passe est bon.

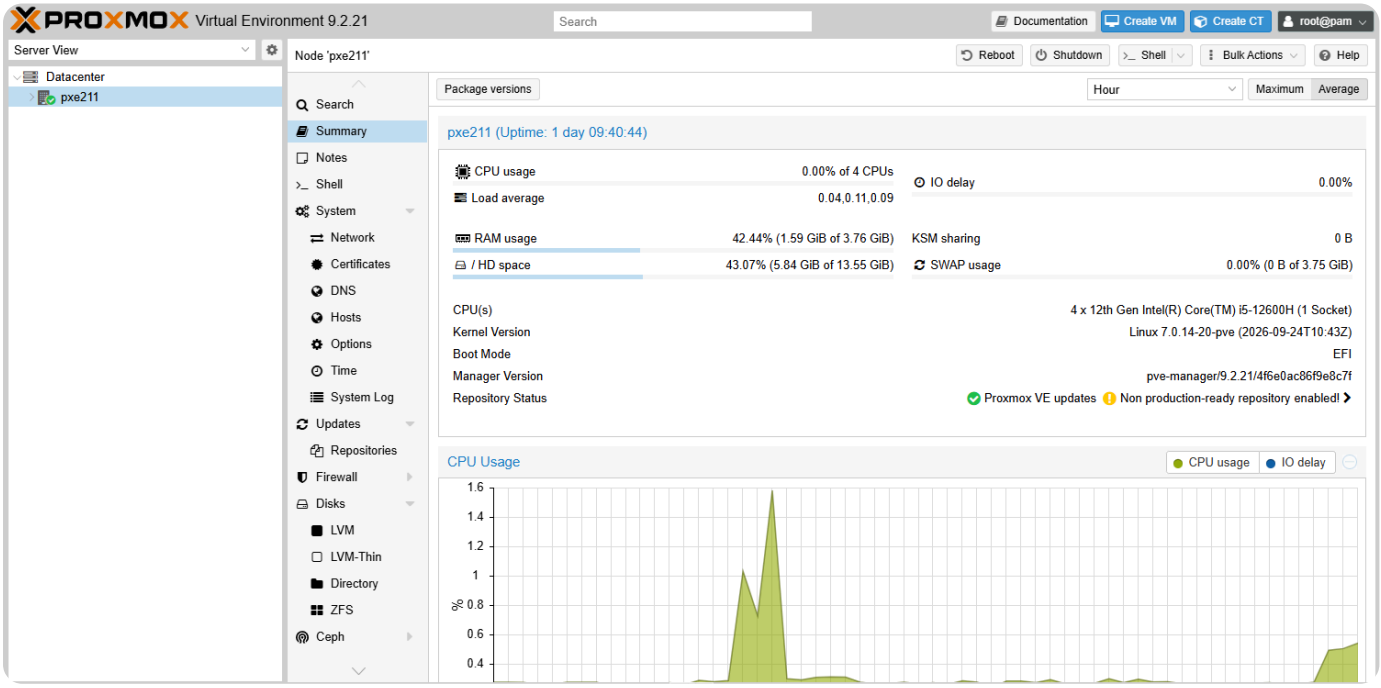
- `Language` **change l'interface, pas le système.** Le choix est enregistré dans votre navigateur, pas sur le serveur : un collègue qui se connecte au même nœud gardera la sienne. Utile à savoir quand vous suivez une documentation en anglais et que votre écran est en français.

- **Regardez la barre d'adresse de votre navigateur : `https://` et le port `8006`.** Proxmox ne répond qu'en HTTPS, avec un certificat auto-signé — d'où l'avertissement de sécurité de votre navigateur au premier accès. Ce n'est pas une anomalie : le certificat est généré à l'installation et ne peut pas être signé par une autorité publique.

5.2 Exploration du nœud

En cliquant sur le nom de votre nœud dans l'arborescence, le panneau central affiche les sections de gestion du nœud. Prenons le temps de les parcourir une par une — cette exploration est la première chose à faire sur un nouveau nœud Proxmox.

Summary affiche l'état général : utilisation CPU, mémoire, stockage, et le résumé des VMs et conteneurs. C'est votre tableau de bord opérationnel.



Ce support est la propriété de Dawan France.

Le **Summary** du nœud `pxe211` du lab. En haut, l'usage du processeur (sur 4 cœurs), de la mémoire et du disque système.

Au milieu, trois lignes à savoir lire. Le **noyau** : `7.0.14-20-pve`, suffixe `-pve`. La **version** de Proxmox : `pve-manager/9.2.21`.

L'état des **dépôts** : « Non production-ready repository enabled », c'est-à-dire le dépôt sans abonnement (§ 6.1). En bas, les courbes d'usage.

Shell ouvre un terminal web directement dans l'interface. Vous êtes connecté en tant que root sur le système Debian.

C'est l'équivalent d'un SSH, sans avoir à configurer quoi que ce soit. Pratique pour les opérations ponctuelles, mais pour un travail régulier en CLI, SSH reste plus confortable.

Network liste les interfaces réseau configurées sur le nœud : bridges (`vmbr0`, `vmbr1`…), interfaces physiques, bonds. C'est ici que vous configurez le réseau de management et les bridges utilisés par vos VMs.

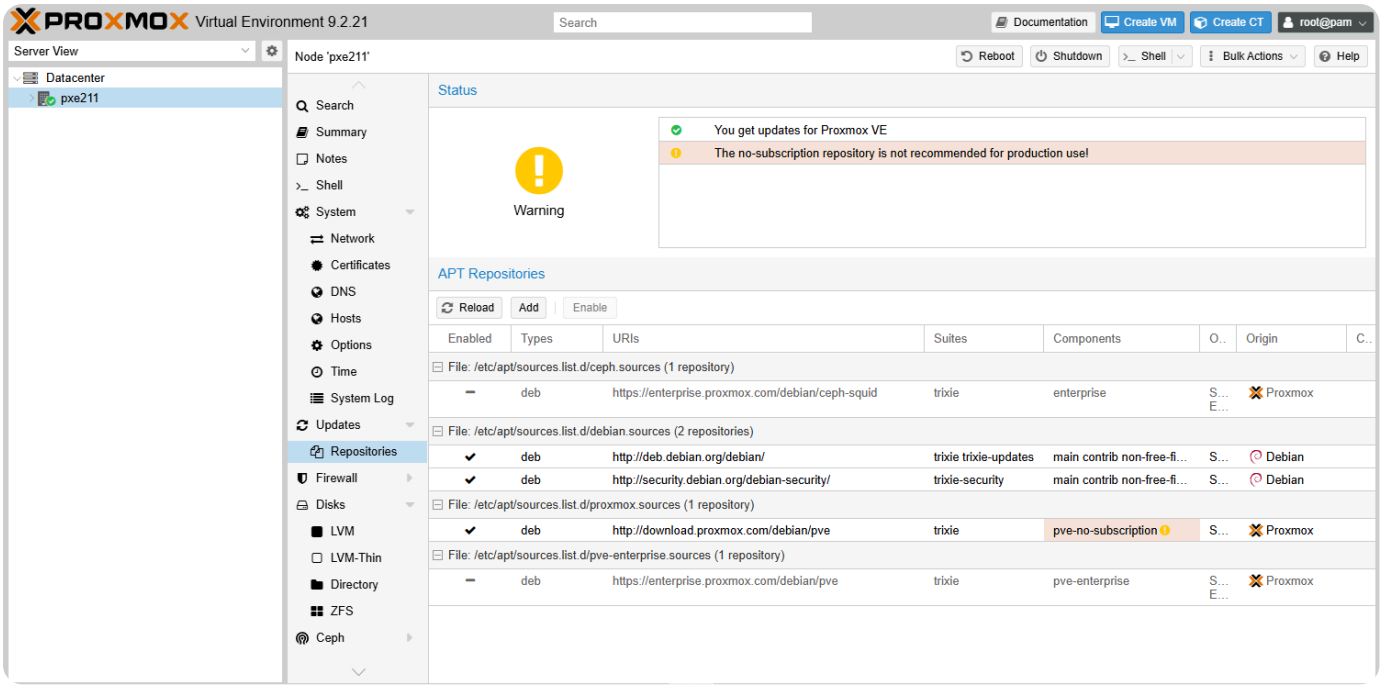
DNS permet de configurer les serveurs DNS et le domaine de recherche du nœud lui-même.

Time gère la synchronisation NTP (*Network Time Protocol*, mise à l'heure par le réseau). Un nœud Proxmox bien synchronisé est essentiel pour le clustering et les certificats.

Certificates permet de remplacer le certificat SSL auto-signé, notamment par un certificat Let's Encrypt si le nœud est accessible depuis Internet.

Repositories liste et permet de modifier les dépôts APT (*Advanced Package Tool*, gestionnaire de paquets de Debian) configurés. C'est sur cette page que se fait la bascule vers le dépôt no-subscription.

Ce support est la propriété de Dawan France.



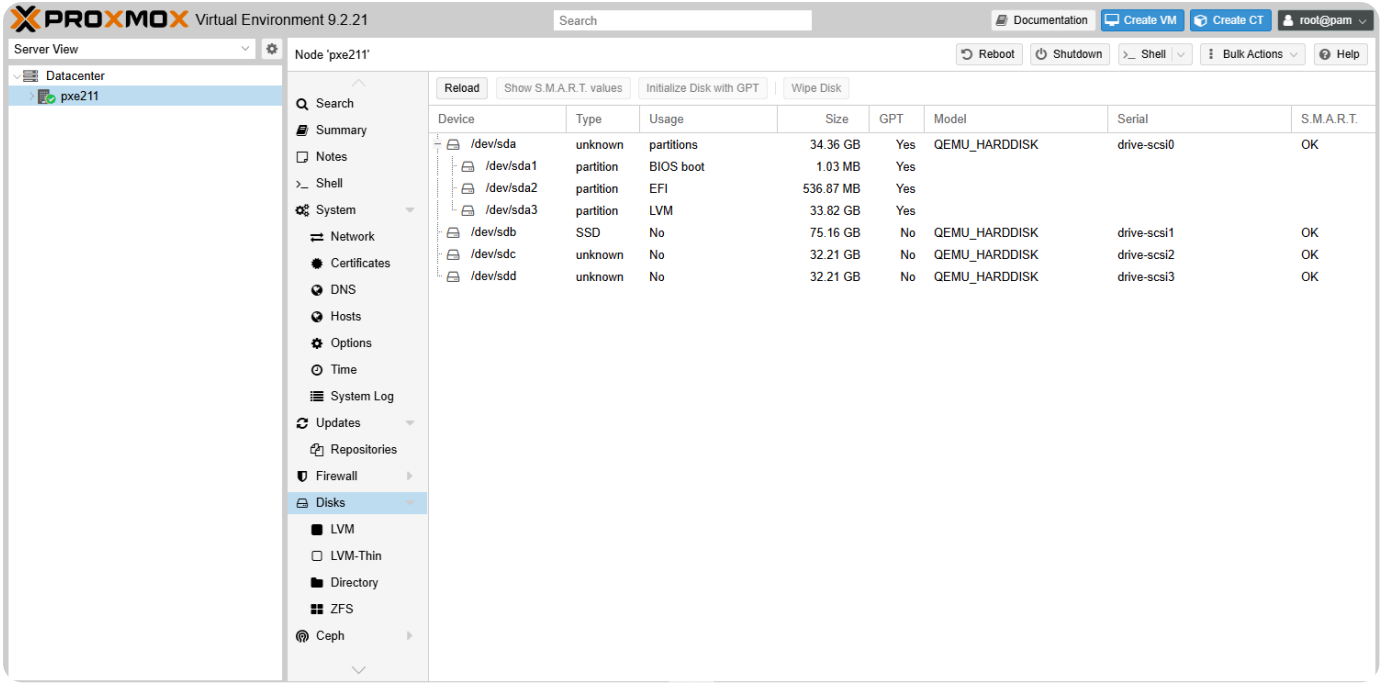
L'onglet **Repositories** d'un nœud déjà réglé : un fichier `.sources` par famille de dépôts. Les dépôts *enterprise* de Proxmox et de Ceph sont désactivés (pas de coche), `pve-no-subscription` est actif, et le bandeau jaune rappelle qu'il n'est pas recommandé en production. C'est l'état que vous obtiendrez à l'atelier 3.

Updates déclenche `apt update` et liste les paquets disponibles depuis l'interface graphique.

Storage affiche l'état des stockages configurés sur ce nœud.

Disks liste les disques physiques du nœud, leurs partitions et leur usage. C'est là qu'on prépare un disque libre pour ZFS, LVM ou Ceph.

Ce support est la propriété de Dawan France.



L'onglet **Disks** de `pxe211`. `sda` porte le système : ses partitions de démarrage (*BIOS boot*, *EFI*) et le volume LVM.

`sdb`, `sdc` et `sdd` sont libres (colonne **Usage** à `No`) : ils sont réservés, `sdc` et `sdd` au miroir ZFS, `sdb` à Ceph. Ne touchez jamais à `sda`.

> **Erreur fréquente :** modifier la configuration réseau depuis l'interface web sans avoir accès physique ou hors-bande au serveur. Si une erreur de configuration réseau est appliquée, vous perdez l'accès à l'interface. Proxmox propose un mode "Apply Configuration" qui vous laisse un délai pour confirmer — utilisez-le.

- 📖 Documentation Proxmox — Web Interface

À retenir : l'interface web s'ouvre sur `https://<ip>:8006`, avec un compte et un **realm** (`pam` ou `pve`). À gauche l'arborescence Datacenter → nœuds → invités et stockages ; au centre, les onglets de l'objet sélectionné. Le **Shell** du nœud donne la ligne de commande sans SSH.

> **▶ À pratiquer maintenant — Atelier 2**
>
> Parcourir l'interface avant de la modifier : l'arbre de gauche, les onglets du nœud, et surtout la différence entre `local` et `local-lvm`, qui explique où vont réellement vos machines.

Dawan
Ce support est la propriété de Dawan France.

Puis créer un troisième stockage, réservé aux ISO, et constater ce qu'il change… et ce qu'il ne change pas.

Ce support est la propriété de Dawan France.

6. Post-installation

6.1 Configuration des dépôts

Une installation neuve de Proxmox VE 9 déclare trois fichiers de dépôts APT. Voici ceux d'un nœud installé depuis l'ISO 9.2 :

Fichier	Dépôt	Abonnement
`debian.sources`	Debian 13 « trixie » (`main` , `trixie-updates` , `trixie-security`)	Non
`pve-enterprise.sources`	Proxmox VE, composant `pve-enterprise`	**Oui**
`ceph.sources`	Ceph Squid, composant `enterprise`	**Oui**

Les deux dépôts enterprise exigent un abonnement actif. Sans lui, `apt update` échoue sur une erreur `401 Unauthorized` .

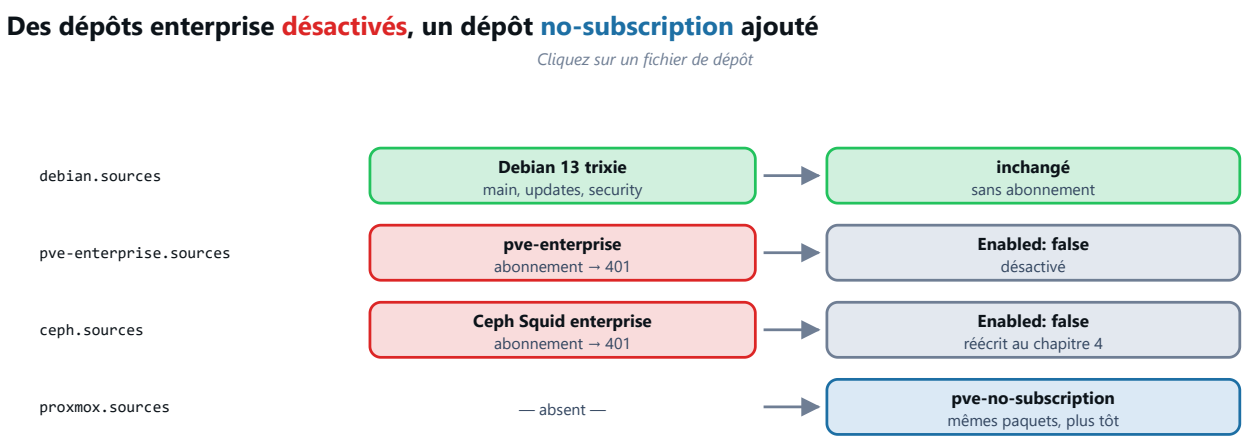
Le système ne reçoit plus les mises à jour Proxmox. L'interface le signale dès le premier jour, dans le **Summary** du nœud et dans `Updates → Repositories` .

Pour nos ateliers — et pour beaucoup d'environnements de test ou de petites infrastructures — Proxmox met à disposition un dépôt **no-subscription**. Il contient les mêmes paquets.

Il les reçoit simplement avant le dépôt enterprise, sans les tests supplémentaires réservés aux abonnés. L'interface le rappelle par un avertissement : *The no-subscription repository is not recommended for production use!* Il convient à un lab ; en production, on prend un abonnement.

La configuration se fait en deux étapes : désactiver **les deux** dépôts enterprise, puis déclarer le dépôt no-subscription.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.



Un fichier .sources par dépôt, au format deb822 (champs Clé: valeur), dans /etc/apt/sources.list.d/.
Sans abonnement, les deux dépôts enterprise font échouer apt update (401 Unauthorized) :
on les désactive, puis on déclare le dépôt no-subscription. Même chose dans Nœud → Updates → Repositories.

Les quatre fichiers de `/etc/apt/sources.list.d/`, avant et après ce paragraphe. Debian ne change pas ; les deux dépôts enterprise sont désactivés ; `proxmox.sources` apporte le dépôt no-subscription. Cliquez sur un fichier pour la commande qui le règle.

Proxmox VE 9 décrit ses dépôts au format deb822 : un fichier `.sources` par dépôt, fait de champs `Clé: valeur`. L'ancien format tenait en une ligne `deb …` dans un fichier `.list`.

Le nouveau format a un avantage direct ici : un dépôt se désactive par un champ `Enabled: false`, sans rien commenter ni supprimer. C'est exactement ce qu'écrit le bouton `Disable` de l'interface.

Depuis l'interface web, `Nœud → Updates → Repositories` fait ces opérations graphiquement : on y voit les dépôts actifs, et les boutons `Disable` et `Add` les gèrent (Atelier 3). Depuis le Shell, la procédure est la suivante :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# 1. Désactiver les deux dépôts enterprise : le champ « Enabled: false » suffit
nano /etc/apt/sources.list.d/pve-enterprise.sources   # ajouter la ligne à la fin du fichier
nano /etc/apt/sources.list.d/ceph.sources             # même chose pour le dépôt Ceph d'entreprise
```

Ce support est la propriété de Dawan France.

La ligne à ajouter à la fin de chacun des deux fichiers, dans `nano`, puis **Ctrl + O**, Entrée et **Ctrl + X** :

```
Enabled: false
```

Dans le Shell du nœud :

```
# 2. Déclarer le dépôt no-subscription de PVE 9 (base Debian 13 « trixie »)
nano /etc/apt/sources.list.d/proxmox.sources          # un fichier neuf
```

Le contenu de `proxmox.sources`, à taper dans `nano` :

```
Types: deb
URIs: http://download.proxmox.com/debian/pve
Suites: trixie
Components: pve-no-subscription
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

Dans le Shell du nœud :

```
# 3. Relire l'état des dépôts : les deux enterprise doivent porter « Enabled: false »
grep -r "Enabled\|Components" /etc/apt/sources.list.d/

# 4. Mettre à jour la liste des paquets, puis le système
apt update                                            # la liste des paquets
apt full-upgrade -y                                   # la mise à jour du système
```

Résultat attendu (nœud `pxe211`, installé depuis l'ISO 9.2) — la relecture, puis la fin d'`apt update` :

```
/etc/apt/sources.list.d/proxmox.sources:Components: pve-no-subscription
/etc/apt/sources.list.d/ceph.sources:Components: enterprise
/etc/apt/sources.list.d/ceph.sources:Enabled: false
/etc/apt/sources.list.d/pve-enterprise.sources:Components: pve-enterprise
/etc/apt/sources.list.d/pve-enterprise.sources:Enabled: false
/etc/apt/sources.list.d/debian.sources:Components: main contrib non-free-firmware
```

Dawan
Ce support est la propriété de Dawan France.

```
/etc/apt/sources.list.d/debian.sources:Components: main contrib non-free-firmware

Get:4 http://download.proxmox.com/debian/pve trixie InRelease [4,357 B]
Get:5 http://download.proxmox.com/debian/pve trixie/pve-no-subscription amd64
Packages [390 kB]
...
181 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Dans la sortie d'`apt update`, **plus aucune ligne ne mentionne** `enterprise.proxmox.com`. La désactivation est prise en compte, et l'erreur `401 Unauthorized` a disparu.

> **Ne pas oublier** `ceph.sources`. Sur le lab, avec le seul dépôt PVE désactivé, `apt update` échouait encore : `Err:6 https://enterprise.proxmox.com/debian/ceph-squid trixie InRelease – 401 Unauthorized`. Il faut donc aussi passer `Enabled: false` dans `ceph.sources`.

Le redémarrage après mise à jour. Si `apt full-upgrade` installe un nouveau noyau, le nœud tourne sur l'ancien jusqu'au redémarrage. `pveversion` le montre.

Sur le lab, juste après la mise à jour, il affichait `pve-manager/9.2.20/49318c671b82f31e (running kernel: 7.0.2-6-pve)`, alors que le noyau `7.0.14-19` était installé. Après `reboot`, il affichait `running kernel: 7.0.14-19-pve`.

> **Un nœud installé en 8.x puis mis à niveau** peut encore porter l'ancien fichier `pve-enterprise.list`, à une seule ligne. On le désactive alors en commentant cette ligne : `nano /etc/apt/sources.list.d/pve-enterprise.list`, puis un `#` devant `deb`.
>
> Une ligne qui commence par `#` est ignorée par apt. Les deux formats cohabitent sans problème : apt lit les `.list` et les `.sources`.

Le dépôt Ceph, lui, ne s'ajoute pas à la main : c'est la commande d'installation de Ceph qui choisit la version et le dépôt, puis réécrit elle-même `ceph.sources`. Ajouter en plus un fichier `ceph.list` déclarerait le même dépôt deux fois.

> **Une erreur** `NO_PUBKEY` **sur un dépôt Proxmox signifie que la clé de signature Proxmox manque.** Elle est fournie par le paquet `proxmox-archive-keyring`, installé d'office sur un

Ce support est la propriété de Dawan France.

système Proxmox mais absent d'une Debian sur laquelle on ajoute Proxmox après coup.

Vérifiez aussi le nom de version : `trixie` correspond à Debian 13, socle de Proxmox VE 9. Sur une installation plus ancienne, ce serait `bookworm`, et un dépôt inexistant renverrait `404 Not Found`.

- 📖 Documentation Proxmox — Package Repositories
- 🎥 Don't run Proxmox without these settings — les réglages post-installation (vidéo en anglais)

6.2 Désactivation de la notification d'abonnement

Au démarrage de l'interface web, Proxmox affiche une popup signalant l'absence d'abonnement actif. Elle ne bloque rien, mais elle est répétitive. Il est important de comprendre ce qu'elle signifie avant de chercher à s'en débarrasser.

Cette popup est un rappel commercial : Proxmox Server Solutions finance son développement par les abonnements de support. Sans abonnement, vous utilisez un logiciel libre et gratuit, mais vous n'avez pas accès au support officiel ni aux dépôts enterprise testés en production. La popup est là pour vous le rappeler.

La solution propre et durable est l'abonnement. Dans le cadre de cette formation, contentez-vous de fermer la popup — elle n'empêche rien de fonctionner.

> **▶ À pratiquer maintenant — Atelier 3**
>
> Basculer sur le dépôt sans abonnement et mettre le système à jour — le premier geste sur tout nœud Proxmox fraîchement installé.

6.3 Vérification de l'installation

Avant de considérer un nœud Proxmox comme opérationnel, quelques vérifications s'imposent. Elles se font depuis le Shell de l'interface web ou via SSH.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Vérifier la version de Proxmox VE installée
pveversion
```

```
# Vérifier les services critiques
systemctl status pveproxy    # Interface web
systemctl status pvedaemon   # Démon principal PVE
systemctl status corosync    # Clustering : « inactive » tant qu'aucun cluster n'existe
systemctl status pve-cluster # pmxcfs, le système de fichiers /etc/pve (actif même seul)

# Vérifier les stockages disponibles
pvesm status

# Consulter l'état du nœud via l'API locale
pvesh get /nodes/pve/status
```

Résultat attendu de pveversion :

```
pve-manager/9.2.3/d0fde103346cf89a (running kernel: 7.0.6-2-pve)
```

Résultat attendu de pvesm status (les tailles dépendent de votre disque) :

```
Name             Type     Status     Total (KiB)      Used (KiB) Available (KiB)
%
local             dir     active        98497780         6703992        86744240
6.81%
local-lvm     lvmthin     active       855855104        84986411       770868692
9.93%
```

La commande `pveversion` est votre première vérification : elle indique la version de `pve-manager` et surtout le noyau en cours d'exécution. Le suffixe `-pve` du noyau (`7.0.6-2-pve` ici) confirme que le noyau Proxmox est bien chargé — et non le noyau Debian standard, qui se terminerait par `-amd64`.

Le numéro lui-même évolue avec les mises à jour : Proxmox fait suivre à sa branche 9.x des noyaux récents, bien plus récents que celui de Debian 13.

Côté services, `pveproxy`, `pvedaemon` et `pve-cluster` doivent être `active (running)`. **`corosync` apparaît `inactive (dead)` avec la mention `start condition unmet` sur un nœud**

Ce support est la propriété de Dawan France.

seul : c'est l'état normal, son service attend le fichier de configuration qu'écrit la création d'un cluster.

`pvesm status` (Proxmox VE Storage Manager) liste tous les stockages configurés avec leur état (`active` ou `error`) et leur taux d'utilisation. Si un stockage apparaît en erreur dès le démarrage, c'est un signal d'alarme à traiter avant de créer quoi que ce soit.

> **Erreur fréquente :** ignorer l'état des services après une mise à jour. Toujours relancer `systemctl status pveproxy pvedaemon` après un `apt full-upgrade` pour s'assurer que les services ont redémarré proprement.

> Vous avez désactivé le dépôt `pve-enterprise`, activé `no-subscription`, et pourtant `apt update` échoue encore sur une erreur `401`. Quelle est la cause la plus probable ?
> ▼ Réponse
> **Le dépôt enterprise de Ceph est resté actif.** Une installation neuve déclare trois fichiers de dépôts, dont `ceph.sources`, qui pointe lui aussi vers `enterprise.proxmox.com`. Tant qu'il n'est pas désactivé, `apt update` bute sur lui. On le désactive comme celui de PVE (bouton **Disable** dans **Repositories**, ou `Enabled: false` dans le fichier).

À retenir : sans abonnement, les dépôts *enterprise* renvoient `401` : on les désactive (**PVE et Ceph**) et on active **no-subscription**, puis `apt update` et `apt full-upgrade`. Un nouveau noyau ne sert qu'après redémarrage (`pveversion` le montre). La fenêtre d'abonnement est un rappel commercial, pas un blocage.

Ce support est la propriété de Dawan France.

7. Comptes et droits : utilisateurs, realms et pools

Un Proxmox fraîchement installé n'a qu'un seul compte utile, `root@pam`, qui sert à tout : l'interface web, l'API et le shell. C'est pratique à l'installation, et c'est une faille dès qu'on travaille à plusieurs.

Cette section pose le modèle des comptes et des droits : utilisateurs, rôles, realms. Puis l'outil qui range les ressources et permet de déléguer sans tout ouvrir : les pools. La section suivante protège les connexions elles-mêmes.

7.1 Utilisateurs, rôles et chemins

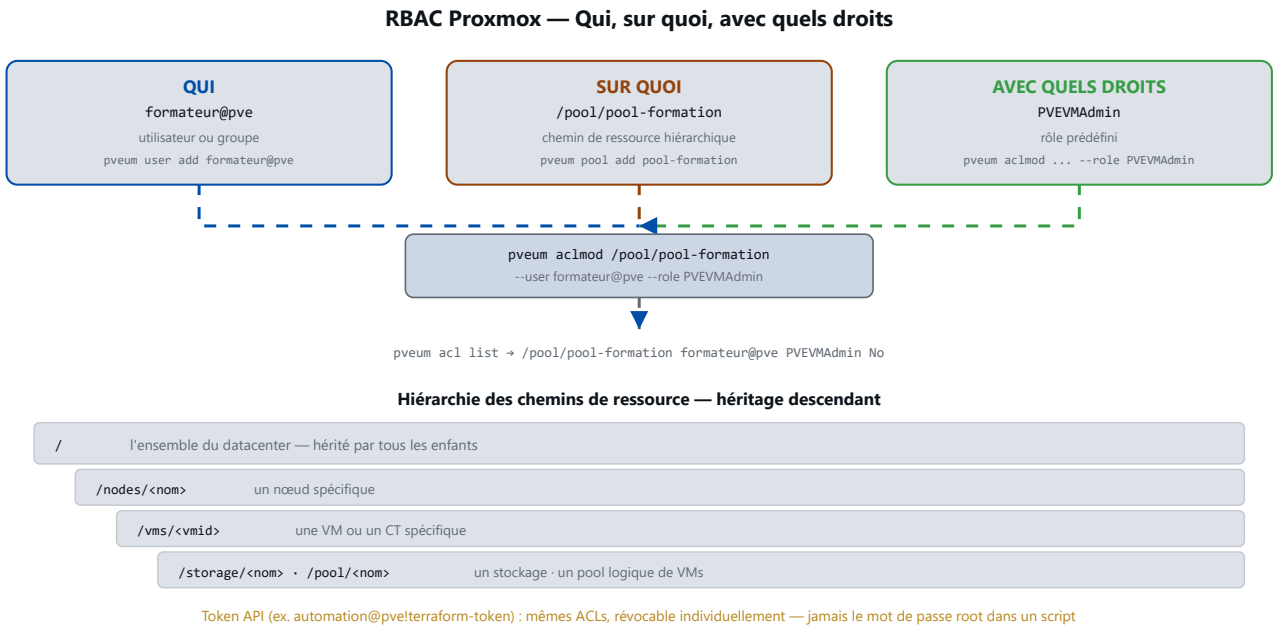
Un **contrôle d'accès** répond à trois questions : **qui** demande, **sur quoi**, **pour faire quoi**. Proxmox y répond par un modèle à rôles (RBAC : *Role-Based Access Control*). Ce modèle associe un **utilisateur** (ou un groupe), un **rôle** qui regroupe des privilèges, et un **chemin** sur lequel le droit s'applique.

Terme	Ce que c'est	Dans le laboratoire
Realm	L'endroit où le mot de passe est contrôlé : `pam` (comptes Linux du nœud) ou `pve` (comptes propres à Proxmox), détaillé au §7.2	`pam` pour `root`, `pve` pour `formation`
Utilisateur	Un compte, toujours écrit `nom@realm`	`formation@pve`
Groupe	Un ensemble d'utilisateurs, à qui l'on donne les droits une seule fois	`stagiaires`
Rôle	Une liste de **privilèges** (`VM.PowerMgmt`, `VM.Console`…)	`PVEVMAdmin`
Chemin	L'objet protégé : `/` (tout), `/nodes/<nœud>`, `/vms/<id>`, `/storage/<id>`, `/pool/<nom>`	`/`
Propagation	Le droit posé sur un chemin descend sur tout ce qui est dessous	Le droit sur `/` vaut pour toutes les VM et tous les conteneurs

Ce support est la propriété de Dawan France.

Les rôles fournis couvrent les besoins courants : `Administrator` (tout), `PVEAdmin` (VM et conteneurs, sans la configuration système du nœud), `PVEVMAdmin` (gestion complète des VM et conteneurs). Viennent ensuite `PVEVMUser` (démarrer, arrêter, console), `PVEAuditor` (lecture seule) et `PVEDatastoreAdmin` (stockages). `pveum role list` affiche la vingtaine de rôles et leurs privilèges.

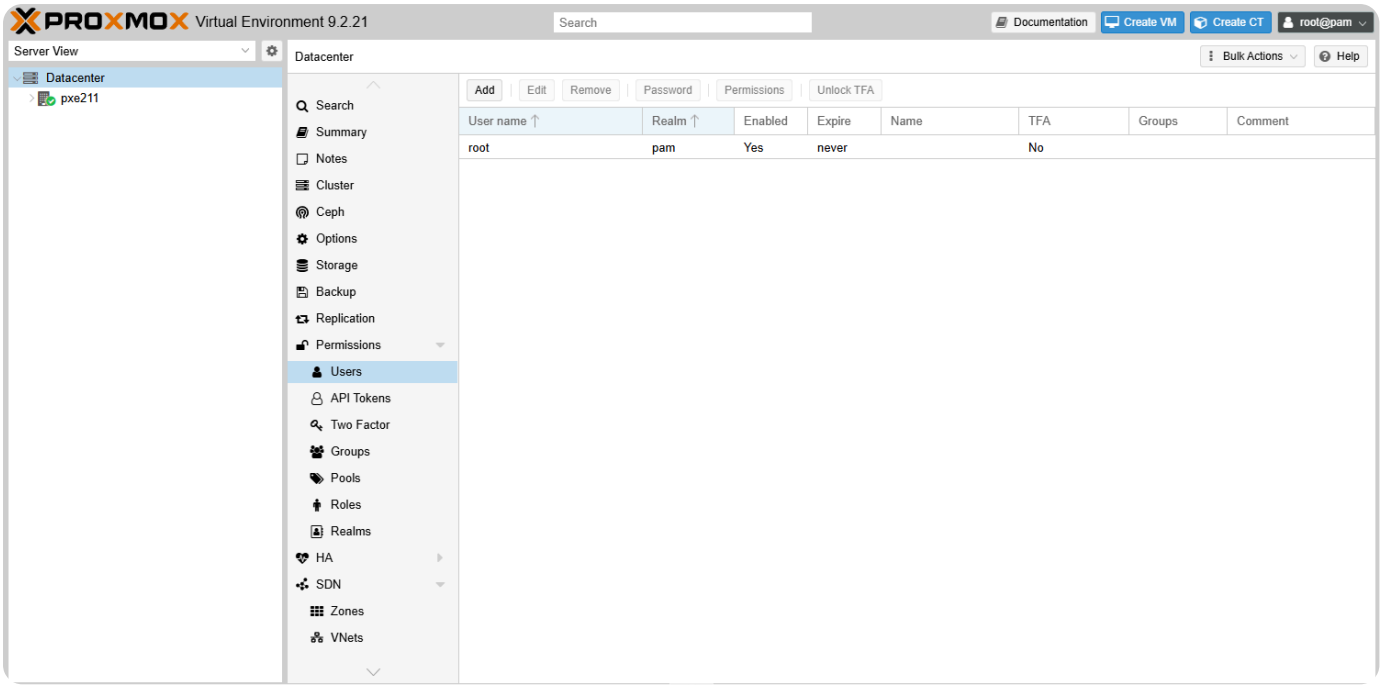
Une permission se lit donc : « l'utilisateur `formation@pve` a le rôle `PVEVMAdmin` sur le chemin `/` », et, par propagation, sur tout ce qu'il contient.



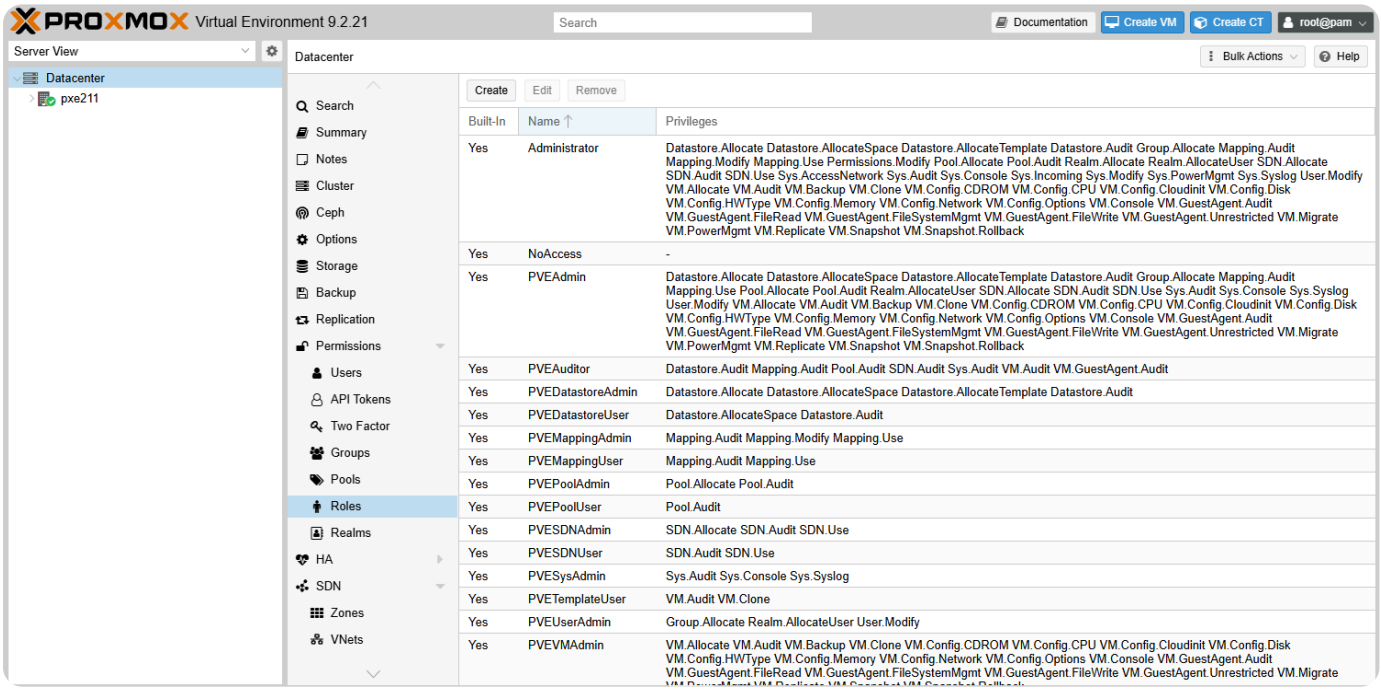
Les trois dimensions d'un droit : **qui**, **sur quoi** (du datacenter entier `/` à une seule VM `/vms/<vmid>`), et **avec quel rôle**. Un droit posé haut dans l'arborescence descend sur tout ce qui est en dessous.

Dans l'interface : **Datacenter** → **Permissions** → **Users** crée les utilisateurs, **Groups** les groupes, **Roles** liste les rôles, et **Permissions** → **Add** crée l'association utilisateur, rôle, chemin.

Ce support est la propriété de Dawan France.



Datacenter → **Permissions** → **Users** sur un nœud neuf : un seul compte, `root@pam`, du realm `pam`, actif et sans date d'expiration. Le bouton **Permissions** affiche tout ce qu'un compte peut faire.



Datacenter → **Permissions** → **Roles** : les rôles fournis (**Built-In** à `Yes`), chacun avec ses **privilèges**. `Administrator` les a tous ; `PVEAuditor` n'a que des privilèges `.Audit`.

En ligne de commande, `pveum` fait les mêmes opérations :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Créer un utilisateur Proxmox interne (realm pve)
pveum user add formation@pve --password 'Dawan2025!' --comment "Compte de formation"

# Créer un groupe, puis y ajouter l'utilisateur
pveum group add stagiaires --comment "Groupe des stagiaires"
pveum user modify formation@pve --groups stagiaires   # rattacher au groupe

# Donner le rôle PVEVMAdmin sur tout le datacenter (chemin /)
pveum acl modify / --users formation@pve --roles PVEVMAdmin

# Vérifier les utilisateurs, puis les droits posés
pveum user list   # les comptes
pveum acl list    # les droits posés
```

Résultat attendu : `pveum user list` montre `formation@pve`, avec `stagiaires` dans la colonne `groups`, `pve` dans `realm-type` et `1` dans `enable` (compte actif). `pveum acl list` montre le triplet `/`, `formation@pve`, `PVEVMAdmin`, avec `propagate` à `1`.

Tout est écrit dans `/etc/pve/user.cfg`. Pour retirer un droit : `pveum acl delete` avec les mêmes options.

Bonne pratique de délégation. Attribuez les droits à des **groupes**, pas à des utilisateurs un par un : l'ajout d'un membre suffit ensuite à lui donner tous les accès prévus. Deux mécanismes complètent ce modèle, au §8 : la double authentification et les jetons d'API.

7.2 Les domaines d'authentification (realms)

Un utilisateur Proxmox n'est que la contrepartie d'une identité vérifiée par un **domaine d'authentification** (*realm*). Deux fichiers structurent l'ensemble, tous deux répliqués par `pmxcfs`. **`/etc/pve/user.cfg`** stocke les *attributs* des utilisateurs : prénom, nom, courriel, groupes, expiration, activé/désactivé, clés 2FA — **jamais les mots de passe**.

`/etc/pve/domains.cfg` déclare les realms. Les mots de passe du realm Proxmox, eux, sont hachés en SHA-256 (*Secure Hash Algorithm*, empreinte de 256 bits) dans `/etc/pve/priv/shadow.cfg`.

Proxmox VE prend en charge cinq types de realms :

Ce support est la propriété de Dawan France.

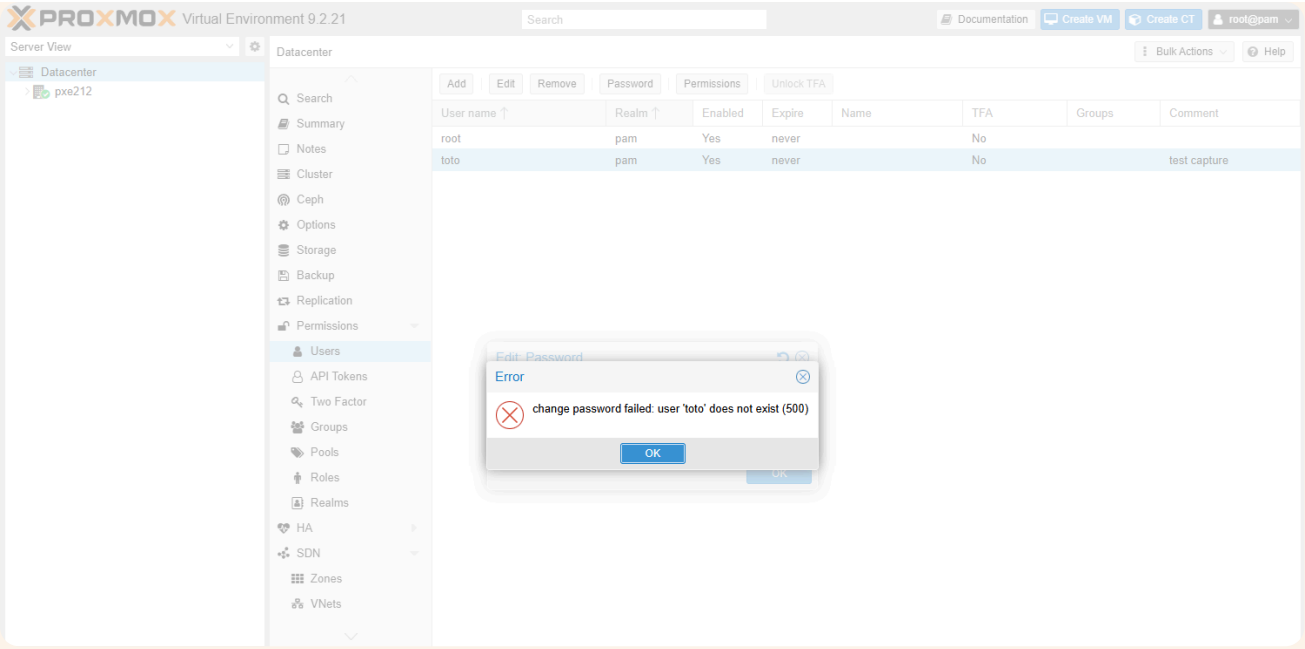
Realm	Où sont vérifiées les identités	Usage typique
Linux PAM	Comptes du système Linux de **chaque nœud** (`adduser`)	Administrateurs ayant déjà un compte shell sur le serveur. Présent par défaut, non supprimable.
Proxmox VE Authentication Server	Base interne à Proxmox (`shadow.cfg` , SHA-256)	Le plus simple : utilisateurs créés et gérés entièrement depuis Proxmox, sans compte Linux. **À préférer pour déléguer.**
LDAP	Annuaire LDAP externe (ex. OpenLDAP)	Parc d'utilisateurs déjà centralisé en LDAP.
Active Directory	Annuaire AD (LDAP sous le capot)	Environnements Windows/AD.
OpenID Connect	Fournisseur d'identité externe (OAuth 2.0)	Authentification déléguée / SSO (*Single Sign-On*, connexion unique).

La distinction entre `@pam` et `@pve` mérite d'être soulignée. Un utilisateur `@pam` existe dans le système Linux — sa création dans Proxmox ne fait que l'autoriser à se connecter à l'interface.

Un utilisateur `@pve` n'existe que dans Proxmox — il n'a pas de compte Linux sur le serveur. Pour la plupart des opérations de formation ou de délégation, les utilisateurs `@pve` sont préférables car ils sont totalement sous le contrôle de Proxmox.

> **Erreur fréquente — connexion impossible avec un utilisateur PAM :** vous créez un utilisateur `monuser@pam` dans Proxmox. Mais le compte Linux correspondant n'existe pas sur le serveur Debian, ou son mot de passe ne correspond pas.
>
> La tentative de connexion retourne alors une erreur d'authentification PAM. Proxmox ne crée pas le compte Linux — il vérifie uniquement l'authentification PAM pour un compte qui doit *déjà exister*.

Dawan
Ce support est la propriété de Dawan France.



L'écran `Datacenter → Permissions → Users`, au moment précis où l'erreur tombe : `change password failed: user 'toto' does not exist (500)`. Sur le lab (Proxmox VE 9.2.21), après avoir déclaré `toto@pam` sans créer de compte Linux `toto`.

Le message paraît absurde — l'utilisateur `toto` est pourtant **visible dans la liste, juste au-dessus**. C'est tout l'intérêt de cette capture.

- **Lisez la colonne `Realm`.** `toto` est en `pam`, comme `root`.

  Or un compte `@pam` n'existe pas *dans* Proxmox : Proxmox ne fait que **déléguer** l'authentification au système Linux du serveur. La ligne dans la grille dit seulement « ce compte Linux a le droit de se connecter ici » — elle ne crée aucun utilisateur.

- **Le bouton `Password` d'un compte `@pam` change le mot de passe Linux.** Encore faut-il que le compte Linux existe.

  Sur le lab : pour un compte Linux créé avec `useradd`, puis déclaré en `@pam`, le changement réussit et `/etc/shadow` est mis à jour. Pour `toto`, sans compte Linux, il échoue avec le message de la capture.

- **Un compte `@pve`, lui, existe réellement dans la base de Proxmox.** Son mot de passe est stocké par Proxmox, sans aucun compte Linux derrière.

La règle à retenir : `@pam` = compte Linux du serveur, créé avec `useradd`, mot de passe géré par Linux. `@pve` = compte interne à Proxmox, créé et géré entièrement depuis l'interface.

Ce support est la propriété de Dawan France.

> Le premier sert aux administrateurs système qui ont déjà un compte sur la machine ; le second, à tous les autres. C'est le second qu'il faut préférer pour déléguer des droits sans ouvrir de session Linux.

Administrer la machine, ou administrer Proxmox ?

La question « dans quel realm créer mon compte ? » se tranche selon ce que la personne doit pouvoir faire. Le rôle `Administrator` donne les mêmes droits dans l'interface quel que soit le realm ; ce qui change, c'est l'accès **au système Linux** du nœud.

	Administrateur de la machine (`@pam`)	**Administrateur Proxmox (`@pve`)**
Où vit le compte	Dans `/etc/passwd` du nœud (`adduser`) **et** dans Proxmox (`pveum user add …@pam`)	Seulement dans Proxmox (`/etc/pve/user.cfg`, mot de passe dans `/etc/pve/priv/shadow.cfg`)
Interface web et API	Oui, avec le mot de passe Linux	Oui, avec le mot de passe Proxmox
Commandes système (`apt`, réseau, `/etc`)	Oui, dans le Shell du nœud	Non
En cluster	Le compte Linux est **local** : à créer sur chaque nœud	Le compte est répliqué par pmxcfs sur tous les nœuds
Pour qui	L'équipe qui exploite les serveurs	Les personnes qui gèrent des VM, des sauvegardes, un projet

La règle qui en découle : **un compte `@pam` nominatif pour chaque administrateur système**, qui doit d'abord exister dans le système (`adduser`) avant d'être déclaré dans Proxmox. Pour tous les autres, **des comptes `@pve`**, avec des droits limités à leur périmètre (§7.3).

> **Ne jamais désactiver ni supprimer le compte `root@pam`.** Il est indispensable au fonctionnement du cluster. C'est avec son mot de passe qu'un nouveau nœud s'authentifie auprès du nœud existant quand il rejoint un cluster.
>
> C'est ensuite en `root`, **par clé SSH**, que les nœuds se parlent pour migrer une VM ou synchroniser les certificats. `root@pam` ne peut d'ailleurs pas être supprimé ; ses attributs (courriel, 2FA) restent modifiables, mais il conserve toujours tous les droits.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

7.3 Les pools

Un **pool** est un regroupement nommé de VM, de conteneurs et de stockages. Une permission posée sur `/pool/<nom>` vaut pour tout ce que le pool contient, y compris ce qu'on y ajoutera par la suite. On délègue ainsi à une équipe « ses » machines sans les énumérer une par une.

Dans l'interface : **Datacenter** → **Permissions** → **Pools** pour créer le pool, puis son onglet **Members** pour y ranger des machines. En ligne de commande : `pveum pool add <nom>`, puis une permission sur `/pool/<nom>`.

- 📖 Pools de ressources — guide d'administration Proxmox VE

> **▶ À pratiquer maintenant — Atelier 4**
>
> Créer un utilisateur et lui attribuer un rôle sur un périmètre précis. Vérifier ce qu'il voit vraiment en se connectant avec son compte. L'atelier n'utilise que des comptes `@pve`, internes à Proxmox : un compte `@pam`, lui, doit d'abord exister dans le système (`adduser`).

- 📖 Documentation Proxmox — User Management

À retenir : un droit = un **utilisateur** (ou groupe) + un **rôle** (liste de privilèges) + un **chemin** (`/`, `/vms/100`, `/pool/<nom>`). L'utilisateur appartient à un **realm** (`pam` = comptes Linux, `pve` = comptes Proxmox).

Ce support est la propriété de Dawan France.

8. Sécuriser les accès : double authentification et jetons d'API

Les droits (§7) disent ce qu'un compte peut faire. Reste à protéger la connexion elle-même. Proxmox fournit deux outils : la **double authentification** pour les personnes, les **jetons d'API** pour les scripts.

8.1 Double authentification de l'interface web

L'interface web de Proxmox est accessible par défaut avec un simple couple identifiant/mot de passe. Pour un hyperviseur exposé sur un réseau ou sur Internet, c'est insuffisant. Proxmox intègre nativement le **TOTP** (Time-based One-Time Password) — le même mécanisme que Google Authenticator ou Authy.

Une fois activé, la connexion nécessite le mot de passe habituel **plus** un code à 6 chiffres qui change toutes les 30 secondes. Même si un attaquant obtient le mot de passe, il ne peut pas se connecter sans le code TOTP généré par le téléphone de l'administrateur.

La configuration se fait entièrement en GUI : **Datacenter** → **Permissions** → **Two Factor** → **Add** → **TOTP** (ou, pour son propre compte, menu utilisateur en haut à droite → **TFA**). Proxmox affiche un code QR (code-barres en deux dimensions) à scanner avec l'application d'authentification. Après validation, le 2FA est actif pour cet utilisateur.

- 📖 Authentification TFA/TOTP dans Proxmox — pve-admin-guide

> **Précaution :** ajoutez aussi des **clés de récupération** (**Add** → **Recovery Keys**), à conserver dans un endroit sécurisé. Si vous perdez accès à votre application d'authentification sans elles, vous serez bloqué hors de votre interface Proxmox. Seul un autre administrateur, ou `root` en console, pourra retirer votre second facteur.

8.2 Les jetons d'API

Un script qui pilote Proxmox (Terraform, Ansible, une supervision) ne doit pas utiliser le mot de passe `root`. On lui donne un **jeton d'API** : un secret rattaché à un utilisateur, révocable seul, dont chaque appel est journalisé, et qui ne s'affiche qu'une fois, à sa création.

Ce support est la propriété de Dawan France.

Ce qui protège n'est pas le jeton lui-même, mais le **rôle** et le **chemin** qu'on lui associe. Un jeton avec le rôle `Administrator` sur `/` vaut le mot de passe `root`. Création dans **Datacenter** → **Permissions** → **API Tokens**, ou par `pveum user token add <utilisateur> <nom>`.

- 📖 Proxmox VE API

> **▶ À pratiquer maintenant — Atelier 5**
>
> Ajouter un TOTP à un compte d'administration, constater qu'il est exigé à la connexion, puis le retirer en `root`, comme après la perte d'un téléphone.

À retenir : le **2FA TOTP** ajoute à l'interface web un code à six chiffres (gardez des clés de récupération). Un **jeton d'API** remplace le mot de passe dans les scripts : révocable seul, traçable, limité par son rôle et son chemin.

Ce support est la propriété de Dawan France.

9. Conteneurs LXC : comprendre avant de faire

9.1 La virtualisation par isolation — comprendre avant de faire

Avant de créer notre premier conteneur, il est utile de s'arrêter une minute sur ce que signifie "virtualiser par isolation". Vous avez sans doute déjà entendu parler de machines virtuelles — avec KVM ou VMware, on émule un ordinateur complet, avec son propre noyau, ses propres pilotes, son BIOS virtuel.

C'est puissant, mais cela a un coût : chaque VM doit démarrer un système d'exploitation entier. Cela prend du temps, de la mémoire et des ressources processeur, même quand la VM ne fait rien d'utile.

La virtualisation par isolation repose sur un principe radicalement différent. Plutôt que d'émuler un ordinateur complet, on **partage le noyau** de la machine hôte entre plusieurs environnements isolés. Ces environnements s'appellent des conteneurs.

Chacun voit son propre système de fichiers, son propre réseau, ses propres processus. Mais tous utilisent le même noyau Linux, celui qui tourne sur l'hôte physique.

Le gain est immédiat : un conteneur démarre en quelques secondes (parfois moins), consomme quelques dizaines de mégaoctets de RAM au repos, et peut être créé ou détruit en un instant. C'est pourquoi LXC est idéal pour héberger des services légers : un serveur web, une base de données, un reverse proxy. On ne gaspille pas les ressources que réclamerait une VM complète pour le même service.

La contrepartie est tout aussi claire : tous les conteneurs partagent le même noyau. On ne peut donc pas faire tourner Windows dans un conteneur LXC.

On ne peut pas non plus charger des modules noyau arbitraires depuis l'intérieur d'un conteneur non-privilégié. LXC est Linux pour Linux.

Un conteneur est un système Linux complet, isolé des autres, qui s'exécute sur le noyau du nœud. Deux mécanismes du noyau assurent cette isolation : les namespaces et les cgroups.

Dawan
Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
Machine virtuelle	Un ordinateur simulé qui démarre **son propre noyau**	Aucune à ce stade
Conteneur LXC	Un système Linux complet (services, utilisateurs, réseau) qui tourne sur le noyau du nœud	Le CT 200 `cache-apt`, créé à l'atelier 6
Noyau de l'hôte	Le noyau `-pve` du nœud, partagé par tous les conteneurs	`uname -r` dans le Shell du nœud
Namespaces	Ce que le conteneur **voit** : ses processus, son réseau, ses fichiers, son nom (§9.2)	Le PID (*Process ID*, numéro de processus) 1 du conteneur, vu par l'hôte avec un autre numéro (§9.2)
cgroups	Ce que le conteneur peut **consommer** : processeur, mémoire, disque (§9.2)	La mémoire et les cœurs d'un conteneur, réglés à chaud au §12.3
Template	Le système de fichiers d'une distribution, prêt à copier (§11)	Le template Debian téléchargé au §11.2

Le noyau partagé étant un noyau Linux, un conteneur ne peut faire tourner que Linux : pas de Windows dans un conteneur. Le schéma interactif du §9.2 montre les namespaces (ce que le conteneur voit) et les cgroups (ce qu'il consomme).

Où voir un conteneur sur votre nœud :

Notion	Où la voir
Le conteneur	L'arborescence à gauche, sous le nœud : l'icône de cube ; `pct list`
Sa configuration	`/etc/pve/lxc/<id>.conf` (celle d'une VM : `/etc/pve/qemu-server/<id>.conf`) ; `pct config <id>`
Le noyau partagé	`pct exec 200 -- uname -r` donne la même version que `uname -r` sur le nœud
Ses limites	L'onglet **Resources** du conteneur (mémoire, cœurs)
Ses processus vus de l'hôte	§9.2 : le `systemd` du conteneur, avec son PID côté hôte

- 📖 Documentation Proxmox — Linux Container
- 📖 Guide complet des conteneurs LXC Proxmox — pve-admin-guide

---

9.2 LXC — histoire et mécanismes noyau

Dawan Nous donnons vos savoirs
Ce support est la propriété de Dawan France.

9.2.1 Origine et place dans l'écosystème

LXC — Linux Containers — a été développé en 2008 par Daniel Lezcano chez IBM (constructeur informatique américain). C'est une interface de haut niveau vers les primitives d'isolation que le noyau Linux avait progressivement intégrées depuis le milieu des années 2000.

Docker est apparu cinq ans plus tard, avec une couche d'abstraction supplémentaire orientée application. LXC, lui, reste proche du noyau : c'est une virtualisation légère de système, pas de processus.

Proxmox VE intègre LXC via son propre outil, **pct** — le Proxmox Container Toolkit. `pct` est la commande qui fait le lien entre l'interface graphique Proxmox, l'API, et les primitives LXC sous-jacentes. Quand vous créez un conteneur depuis l'interface web, c'est `pct create` qui s'exécute en coulisses.

9.2.2 Les deux piliers : namespaces et cgroups

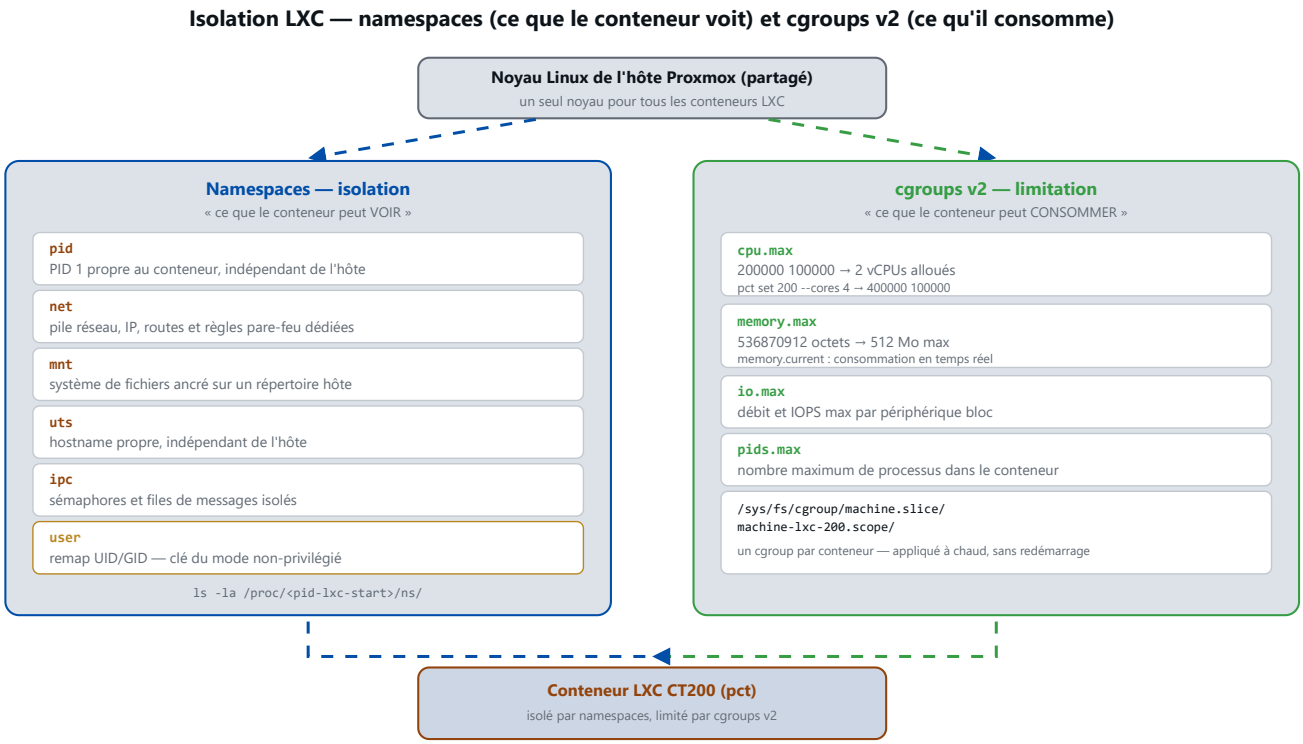
L'isolation dans LXC repose sur deux mécanismes du noyau Linux. Ce sont eux qui définissent à la fois ce qu'un conteneur sait faire, et ses limites.

Les **namespaces** sont la brique d'isolation : ils définissent ce que le conteneur **voit**. LXC utilise les six historiques :

- **pid** — le conteneur voit ses propres processus ; son premier processus a le PID 1 de son point de vue ;
- **net** — sa propre pile réseau : interfaces, routes, pare-feu ; c'est ce qui lui donne une adresse IP indépendante ;
- **mnt** — son propre système de fichiers, ancré dans un répertoire dédié sur l'hôte ;
- **uts** — son propre nom d'hôte ;
- **ipc** — l'isolation des communications entre processus ;
- **user** — le remappage des identifiants, clé des conteneurs non privilégiés (§10.1).

Les **cgroups v2** (*control groups*) sont la brique de limitation : ils définissent combien le conteneur peut **consommer** (CPU, mémoire, disque). Chaque conteneur a son groupe dans `/sys/fs/cgroup/lxc/<id>/`.

Dawan
Ce support est la propriété de Dawan France.



La différence qui compte face à une VM : ces limites ne réservent rien. Un conteneur plafonné à 4 Go n'immobilise pas 4 Go : il n'utilise que ce dont il a besoin, et le reste profite aux autres. C'est pourquoi un même hôte porte bien plus de conteneurs que de VM.

C'est aussi ce qui rend le réglage **modifiable à chaud** : `pct set 200 --memory 2048` change la limite d'un conteneur en marche, sans redémarrage. Deux options se confondent souvent :

- `--cores 2` dit **combien de processeurs** le conteneur voit et peut occuper en parallèle ;
- `--cpulimit 1.5` dit **combien de temps de calcul au total** il peut consommer, tous cœurs confondus : ici, l'équivalent d'un processeur et demi.

Dans un conteneur Proxmox, `free -m` affiche la limite du conteneur, et `nproc` le nombre de cœurs attribués : Proxmox recalcule ces informations pour chaque conteneur.

> **Limite importante :** tous les conteneurs LXC partagent le noyau de l'hôte. Si ce noyau comporte une faille de sécurité, un conteneur compromis peut potentiellement l'exploiter pour sortir de l'isolation. C'est pourquoi les mises à jour du noyau de l'hôte Proxmox restent critiques, même quand on n'utilise que des conteneurs.

9.3 Comparatif technique : Conteneurs système (LXC) vs Conteneurs d'application (Docker)

Ce support est la propriété de Dawan France.

Deux questions reviennent très souvent en formation.

- "*Puisque Docker est de la conteneurisation et LXC aussi, pourquoi ne pas utiliser Docker directement dans Proxmox ?*"
- "*Quelle est la différence concrète entre un conteneur LXC et un conteneur Docker ?*"

Pour bien comprendre, il faut distinguer deux philosophies de la conteneurisation : la **conteneurisation système** (LXC) et la **conteneurisation d'application** (Docker). Bien qu'ils s'appuient tous deux sur les mêmes briques du noyau Linux (namespaces et cgroups), ils les exploitent de manière radicalement différente pour répondre à des besoins distincts.

Voici un tableau comparatif complet pour vous aider à y voir clair :

Critère de comparaison	**Conteneurs Système (LXC / Proxmox)**	**Conteneurs d'Application (Docker)**
Philosophie	**Simuler une machine physique complète** (Virtualisation légère). Le conteneur se comporte comme une VM.	**Isoler un seul processus applicatif** (Micro-services). Le conteneur ne vit que pour exécuter sa commande.
Processus d'initialisation	Exécute un gestionnaire d'initialisation complet (comme `systemd` ou `init`) en tant que PID 1.	Exécute directement l'application cible (ex: `nginx`, `python myscript.py`) en tant que PID 1.
Gestion des services	Permet de faire tourner de multiples services en parallèle (cron, syslog, sshd, bases de données) gérés par systemd.	Conçu pour ne faire tourner qu'un seul processus. Pas de démon système par défaut, pas de sshd ni de cron.
Gestion du réseau	Intégration réseau de niveau OS (système d'exploitation) avec IP dédiée, configuration dans `/etc/network/interfaces` ou par DHCP, pontage direct (`vmbr`).	Réseau géré par le moteur Docker (souvent via un réseau NAT interne `docker0` avec translation de ports).
Persistance des données	Persistance par défaut sur le système de fichiers du conteneur (comme sur un disque dur classique).	Éphémère par défaut. Les données doivent être montées via des volumes ou des bind-mounts pour persister.
Cycle de vie	Longue durée. On met à jour le conteneur en se connectant dessus (via `apt upgrade`), on l'arrête, on le redémarre.	Éphémère et jetable. Pour mettre à jour, on détruit le conteneur et

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Critère de comparaison	Conteneurs Système (LXC / Proxmox)	Conteneurs d'Application (Docker)
		on en recrée un nouveau à partir d'une image mise à jour.
Outil de gestion	`pct` sous Proxmox VE, CLI `lxc` sous Linux.	CLI `docker`, `docker compose`.
Cas d'usage idéaux	Hébergement de serveurs web traditionnels, bases de données classiques, serveurs de fichiers, environnements de test multi-services.	Déploiements micro-services cloud-native, pipelines CI (*Continuous Integration*) et CD (*Continuous Deployment*), applications découplées à scalabilité horizontale.

À retenir : un conteneur LXC **partage le noyau** du nœud : les *namespaces* l'isolent, les *cgroups* limitent ce qu'il consomme. Il démarre en une seconde et ne pèse presque rien, mais il ne peut faire tourner que du Linux. LXC fait un **système complet** ; Docker, **une application**.

Ce support est la propriété de Dawan France.

10. Conteneurs privilégiés et non privilégiés

Tous les conteneurs partagent le noyau de l'hôte : la question qui se pose aussitôt est donc celle de la sécurité. Que se passe-t-il si un service compromis obtient les droits `root` *dans* le conteneur ? La réponse dépend d'un seul réglage, choisi à la création — le mode privilégié ou non privilégié.

10.1 Conteneurs privilégiés et non-privilégiés

Proxmox vous propose, à la création de chaque conteneur, de choisir entre un mode **privilégié** et un mode **non-privilégié**. Ce choix conditionne directement la posture de sécurité du conteneur et certaines de ses capacités.

Tout se joue sur les UID (*User ID*, numéros d'utilisateur) que le noyau attribue aux processus du conteneur. Avant de cocher ou décocher cette case, il faut comprendre ce qui se passe réellement sous le capot.

10.1.1 Ce que signifie "non-privilégié" — le remapping d'UIDs

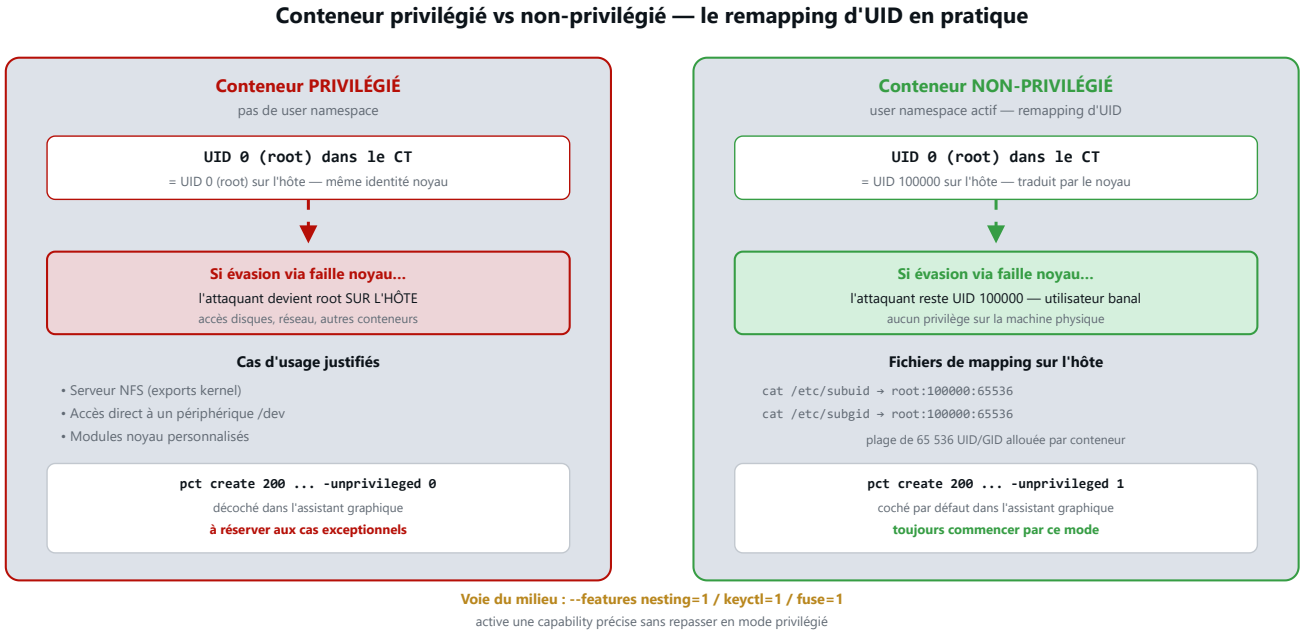
La distinction repose sur une fonctionnalité du noyau Linux appelée **user namespace**. Pour comprendre pourquoi elle change tout, il faut saisir ce que fait root dans un conteneur sans elle.

Sans user namespace (conteneur **privilégié**), l'UID 0 à l'intérieur du conteneur correspond exactement à l'UID 0 sur l'hôte. Du point de vue du noyau Linux, c'est le même utilisateur.

Si un attaquant exploite une faille dans le conteneur et parvient à en sortir, il est immédiatement root sur la machine physique. Il a alors accès à tous les autres conteneurs, aux disques, à la configuration réseau. L'isolation LXC protège contre les accès normaux, mais une faille noyau suffit à tout compromettre.

Avec user namespace (conteneur **non-privilégié**), Proxmox configure un **remapping d'UIDs** : l'UID 0 à l'intérieur du conteneur est traduit en UID 100000 sur l'hôte. L'UID 1 devient 100001, l'UID 1000 devient 101000, et ainsi de suite sur une plage de 65 536 UIDs. Ce remapping est défini dans deux fichiers sur l'hôte :

Dawan
Ce support est la propriété de Dawan France.



Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Plages UID/GID allouées à Proxmox pour les conteneurs non-privilégiés
cat /etc/subuid
# → root:100000:65536

cat /etc/subgid
# → root:100000:65536
```

Résultat attendu :

```
root:100000:65536
```

Cela signifie que l'utilisateur `root` sur l'hôte peut utiliser la plage d'UIDs 100000–165535 pour les conteneurs non-privilégiés. Proxmox gère automatiquement cette allocation — vous n'avez jamais à modifier ces fichiers manuellement.

Concrètement : un attaquant peut obtenir les droits root à l'intérieur du conteneur (UID 0 dans le conteneur = UID 100000 sur l'hôte). Il n'est pourtant qu'un utilisateur ordinaire, sans privilèges, sur la machine physique. La faille noyau ne suffit plus.

10.1.2 Tableau de décision — quand choisir quel mode

Ce support est la propriété de Dawan France.

La règle de départ est simple : **toujours commencer en non-privilégié**. La majorité des services courants fonctionnent parfaitement dans ce mode. Le mode privilégié n'est justifié que dans des cas précis, et encore — souvent évitables grâce aux `features` (voir section suivante).

Cas d'usage	Mode recommandé	Raison
Serveur web (nginx, Apache)	Non-privilégié	Aucune interaction noyau spéciale
Base de données (MariaDB, PostgreSQL)	Non-privilégié	Fonctionne parfaitement
Serveur DNS, DHCP, mail	Non-privilégié	Aucun besoin de privilege
Docker à l'intérieur du LXC	Non-privilégié + `nesting=1`	Feature suffisante depuis PVE 7
Serveur NFS (exports)	Privilégié	NFS kernel server exige l'UID réel
Accès à un périphérique `/dev/`	Privilégié	Remapping incompatible avec `/dev`
Modules noyau personnalisés	Privilégié	Nécessite des capabilities kernel
Tout le reste	Non-privilégié	Par défaut

10.1.3 Les `features` — la voie du milieu

Entre « non privilégié strict » et « privilégié total », Proxmox permet d'activer des **fonctions précises** sur un conteneur non privilégié, par l'option `--features`. On évite ainsi de sacrifier toute la sécurité du remappage pour un besoin ponctuel.

Feature	Usage	Exemple typique
`nesting=1`	systemd récent, Docker ou LXC dans le conteneur	Debian 13, environnement de développement
`fuse=1`	Montages FUSE (sshfs, rclone…)	Stockage distant monté dans le conteneur

> **Le message `WARN: Systemd 257 detected. You may need to enable nesting.`**
>
> À la création d'un conteneur Debian 13, Proxmox affiche cet avertissement. Le conteneur est créé et démarre, mais il faut le traiter : Debian 13 embarque systemd 257, qui a besoin de la

Ce support est la propriété de Dawan France.

> fonction *nesting* dans un conteneur non privilégié. Sans elle, des services peuvent ne pas démarrer, ou les journaux rester vides.
>
> Le nom est trompeur : *nesting* évoque l'imbrication (Docker dans le conteneur), mais c'est surtout ce qui permet à systemd lui-même de fonctionner normalement.

L'activer sur un conteneur déjà créé — interface : `CT 200` → **Options** → **Features** → **Edit** → cocher **Nesting** → **OK**. En ligne de commande :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pct set 200 --features nesting=1   # ajouter la fonction nesting
pct config 200 | grep features      # vérifier la configuration
pct reboot 200                      # redémarrer pour l'appliquer
```

Résultat attendu : `features: nesting=1`.

LXC applique les *features* **au démarrage** du conteneur : le redémarrage est obligatoire, deux à trois secondes pour un conteneur. Pour ne pas voir l'avertissement du tout, ajoutez `--features nesting=1` dès la création, avec `pct create`.

10.1.4 Créer un conteneur dans chaque mode

La commande `pct create` reste identique dans les deux cas, seule la valeur de `--unprivileged` change :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Mode non-privilégié — à utiliser par défaut
# --unprivileged 1 : root du CT = UID 100000 sur l'hôte
pct create 200 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst \
  --hostname ct-secure \
  --unprivileged 1 \
  --memory 512 \
  --rootfs local-lvm:8 \
  --net0 name=eth0,bridge=vmbr0,ip=10.44.1.50/24,gw=10.44.1.1 \
  --password dawan

# Mode privilégié — uniquement si NFS server ou accès /dev indispensable
```

Ce support est la propriété de Dawan France.

```
# --unprivileged 0 : root du CT = root sur l'hôte — risque réel
pct create 300 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst \
  --hostname ct-nfs-server \
  --unprivileged 0 \
  --memory 512 \
  --rootfs local-lvm:8 \
  --net0 name=eth0,bridge=vmbr0,ip=10.44.1.50/24,gw=10.44.1.1 \
  --password dawan

# Vérifier le mode d'un conteneur existant
pct config 200 | grep unprivileged
```

Résultat attendu :

```
unprivileged: 1
```

La valeur `1` confirme le mode non-privilégié. Sur un conteneur privilégié, la commande n'affiche **rien** : `0` est la valeur par défaut, et Proxmox n'écrit pas dans la configuration une option laissée à sa valeur par défaut. Un `grep` muet n'est donc pas une erreur, c'est la réponse « privilégié ».

> **Limitation importante du mode non-privilégié :** les montages NFS *côté serveur* (export via `/etc/exports`) ne fonctionnent pas — le kernel NFS server a besoin d'UIDs réels. En revanche, monter un partage NFS *en tant que client* (`mount -t nfs`) fonctionne parfaitement en non-privilégié.

> **Règle de sécurité :** en production, tous les conteneurs doivent être non-privilégiés sauf justification documentée. Si un service semble nécessiter le mode privilégié, vérifiez d'abord si une `feature` spécifique ne suffit pas — c'est presque toujours le cas.

> Un service compromis donne à un attaquant le compte `root` **à l'intérieur** d'un conteneur non privilégié. Sur le nœud, qui est-il ?

Ce support est la propriété de Dawan France.

▼ Réponse

Un utilisateur sans aucun droit. Dans un conteneur non privilégié, les UID sont décalés : le root du conteneur (UID 0) correspond sur l'hôte à l'UID `100000`, déclaré dans `/etc/subuid`. S'il sort du conteneur, il n'a pas plus de pouvoir qu'un compte ordinaire. Dans un conteneur **privilégié**, il serait le vrai root du nœud : c'est pourquoi le non privilégié est la règle.

À retenir : un conteneur **non privilégié** décale ses UID : le root du conteneur est un utilisateur sans droit sur l'hôte (`/etc/subuid`). C'est le choix par défaut et la règle en production. `nesting` et `fuse` ajoutent des capacités sans passer en privilégié.

Ce support est la propriété de Dawan France.

11. Templates et création d'un conteneur

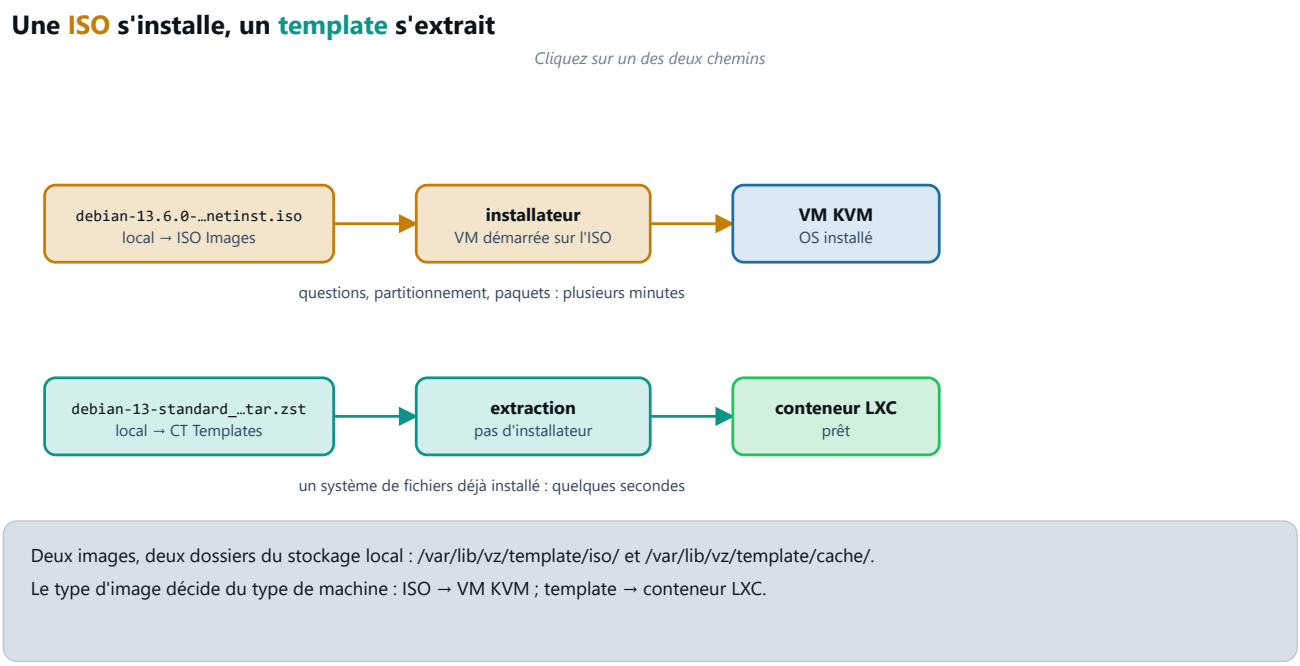
Un conteneur se crée toujours à partir d'un **template** : une archive qui contient le système de fichiers d'une distribution déjà installée. Cette section part des images disponibles sur le nœud, présente les templates LXC et leur gestion, puis crée un premier conteneur — à l'interface, puis en ligne de commande.

11.1 Télécharger des images : ISOs pour les VMs, templates pour les LXC

Avant de créer une VM ou un conteneur, Proxmox a besoin d'une image de référence stockée localement sur le nœud. Il en existe deux types, et ils répondent à des besoins différents.

Une **ISO** est l'image d'un CD d'installation — elle sert à démarrer une VM pour la première fois et lancer l'installateur du système d'exploitation. C'est exactement ce qu'on fait quand on installe Linux ou Windows sur un PC depuis une clé USB. Les ISOs se stockent dans `local → ISO Images` et pointent vers `/var/lib/vz/template/iso/`.

Un **template LXC** est différent : c'est un système de fichiers complet, déjà installé et compressé, prêt à être extrait et démarré en quelques secondes. Il n'y a pas d'installateur — le conteneur est opérationnel dès sa création. Les templates se stockent dans `local → CT Templates` et pointent vers `/var/lib/vz/template/cache/`.



Deux images, deux dossiers du stockage local : /var/lib/vz/template/iso/ et /var/lib/vz/template/cache/.
Le type d'image décide du type de machine : ISO → VM KVM ; template → conteneur LXC.

Ce support est la propriété de Dawan France.

Les deux chemins, de l'image rangée dans `local` à la machine prête. L'ISO passe par un installateur, le template est simplement extrait : d'où les minutes d'un côté et les secondes de l'autre. Cliquez sur un chemin pour le détail.

Dans `local`	Dossier	Exemple	Sert à
ISO Images	`/var/lib/vz/template/iso/`	`debian-13.6.0-amd64-netinst.iso`	créer des VMs KVM
CT Templates	`/var/lib/vz/template/cache/`	`debian-13-standard_13.6-1_amd64.tar.zst`	créer des LXC

Télécharger une ISO pour les VMs

Une ISO est l'image du disque d'installation d'un système : c'est d'elle qu'on installe une VM KVM.

Via l'interface web : cliquez sur votre nœud → **local** → onglet **ISO Images** → bouton **Download from URL**.



On colle l'adresse, puis **Query URL** interroge le serveur : le nom du fichier, sa taille et son type se remplissent seuls. **Download** télécharge l'ISO directement sur le nœud, sans passer par votre poste. L'adresse de l'ISO Debian du cours :

`https://cdimage.debian.org/cdimage/archive/13.6.0/amd64/iso-cd/debian-13.6.0-amd64-netinst.iso`

Le dossier `archive/` conserve toutes les versions de Debian : l'adresse fonctionnera toujours. En ligne de commande, le même fichier se télécharge avec `wget` dans `/var/lib/vz/template/iso/`, le dossier des ISO du stockage `local`.

Ce support est la propriété de Dawan France.

> **Choisissez la version netinst :** environ 750 Mo, contre 4 à 5 Go pour une ISO complète. Si le serveur n'a pas d'accès internet, **local** → **ISO Images** → **Upload** envoie l'ISO depuis votre poste.

11.2 Les templates LXC

Pour créer un conteneur LXC, Proxmox a besoin d'un point de départ — un système de fichiers minimal correspondant à une distribution Linux. Ces archives s'appellent des **templates**. Ce sont l'équivalent des images Docker ou des ISOs pour les VM, en beaucoup plus léger : un template Debian 13 pèse environ 130 Mo une fois compressé.

Proxmox maintient un dépôt de templates officiels pour les distributions les plus courantes : Debian, Ubuntu, Alpine, CentOS, Fedora, et d'autres. Ces templates sont téléchargeables directement depuis l'interface graphique ou depuis la ligne de commande.

- 📖 Proxmox — modèles de conteneurs (pveam) : télécharger et gérer les modèles de conteneurs.

11.2.1 Gérer les templates avec pveam

`pveam` — Proxmox VE Appliance Manager — est l'outil en ligne de commande dédié à la gestion des templates. Pour voir ce qui est disponible dans le dépôt officiel, on consulte la section `system` :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Rafraîchir l'index des modèles (fait aussi chaque jour par une tâche planifiée)
pveam update

# Lister les templates disponibles dans le dépôt (section système)
pveam available --section system
```

Résultat attendu (extrait — la liste complète compte une quarantaine de lignes) :

```
system          alpine-3.24-default_20260714_amd64.tar.xz
system          debian-12-standard_12.12-1_amd64.tar.zst
system          debian-13-standard_13.6-1_amd64.tar.zst
```

Ce support est la propriété de Dawan France.

```
system          debian-13-standard_13.6-1_arm64.tar.zst
system          ubuntu-24.04-standard_24.04-2_amd64.tar.zst
```

La colonne de gauche indique la section (ici `system`), la colonne de droite le nom du fichier template. Le nom se lit en trois morceaux. D'abord la distribution et sa version (`debian-13`).

Puis le type : `standard`, un système minimal avec SSH (`default` chez Alpine). Enfin la **révision du modèle** (`13.6-1`) et l'architecture (`amd64`). Le format `.tar.zst` indique une compression Zstandard, plus rapide que gzip pour des tailles similaires.

Le nom exact change à chaque nouvelle révision. Recopiez-le depuis cette liste plutôt que depuis un support : un `pveam download` avec une révision disparue répond `400 Parameter verification failed`. Et si la liste semble ancienne, c'est l'index local qui date — `pveam update` le rafraîchit.

Pour télécharger un template sur le stockage local de votre nœud Proxmox :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Télécharger le template Debian 13 sur le stockage "local"
pveam download local debian-13-standard_13.6-1_amd64.tar.zst
```

Résultat attendu :

```
downloading https://download.proxmox.com/images/system/debian-13-standard_13.6-
1_amd64.tar.zst to /var/lib/vz/template/cache/debian-13-standard_13.6-
1_amd64.tar.zst
...
download finished
```

Une fois téléchargé, le template est stocké dans `/var/lib/vz/template/cache/`. Pour lister les templates disponibles localement :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Lister les templates présents sur le stockage local
pveam list local
```

Ce support est la propriété de Dawan France.

Résultat attendu :

```
NAME                                                     SIZE
local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst      134.47MB
```

La référence `local:vztmpl/...` est exactement ce qu'on utilisera dans la commande `pct create` pour désigner ce template.

- 📖 Documentation Proxmox — Linux Container
- 📖 Templates LXC disponibles — download.proxmox.com

11.3 Créer un conteneur LXC

11.3.1 Via l'interface graphique

L'interface graphique de Proxmox propose un assistant de création en plusieurs étapes. Pour le lancer, cliquez sur **Create CT** en haut à droite de l'interface. L'assistant vous guidera à travers :

1. **General** — VMID (identifiant numérique unique du conteneur), hostname, mot de passe root, clé SSH facultative
2. **Template** — sélection du template téléchargé (stockage local, sous CT Templates)
3. **Disks** — taille du disque racine et stockage cible (local-lvm recommandé)
4. **CPU** — nombre de cœurs alloués
5. **Memory** — RAM et swap en mégaoctets
6. **Network** — interface réseau, bridge, IP statique ou DHCP, passerelle
7. **DNS** — serveur DNS (hérite des paramètres de l'hôte par défaut)
8. **Confirm** — récapitulatif avant création

> **À noter :** la case "Unprivileged container" en bas de l'onglet General est cochée par défaut depuis Proxmox VE 7. Ne la décochez pas sans raison valable.

11.3.2 Via la ligne de commande avec pct create

La commande `pct create` offre exactement les mêmes options que l'assistant graphique, mais dans un format reproductible et scriptable. C'est ce que les administrateurs utilisent pour automatiser le déploiement de dizaines de conteneurs identiques.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Voici la commande complète pour créer un conteneur webserver avec une IP statique :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Créer le conteneur ID 200 à partir du template Debian 13
# RAM 512 Mo, swap 512 Mo, disque 8 Go sur local-lvm
# IP fixe sur la plage vmbr0, première ressource en .50 (voir la règle d'adressage)
# Mode non-privilégié, démarrage automatique après création
pct create 200 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst \
  --hostname webserver \
  --memory 512 \
  --swap 512 \
  --rootfs local-lvm:8 \
  --net0 name=eth0,bridge=vmbr0,ip=10.44.1.50/24,gw=10.44.1.1 \
  --password Dawan2025! \
  --unprivileged 1 \
  --start 1
```

Résultat attendu (sur PVE 9.2, template Debian 13) :

```
extracting archive '/var/lib/vz/template/cache/debian-13-standard_13.6-
1_amd64.tar.zst'
Total bytes written: 312475648 (298MiB, 156MiB/s)
Detected container architecture: amd64
Creating SSH host key 'ssh_host_ed25519_key' - this may take some time ...
done: SHA256:6/X+dOaSB4jVFWJSk0Y0z6IdYkZY9Shrw2nJt8Jvxl0 root@webserver
WARN: Systemd 257 detected. You may need to enable nesting.
Task finished with 1 warning(s)!
```

Chaque option a un rôle précis. `--memory 512` alloue 512 Mo de RAM. `--rootfs local-lvm:8` crée un volume logique de 8 Go sur le groupe de volumes LVM.

L'option `--net0` configure la première interface réseau avec les paramètres IP statiques. L'option `--start 1` démarre le conteneur immédiatement après la création, sans avoir à lancer `pct start` manuellement.

> **Le message `WARN: Systemd 257 detected. You may need to enable nesting.` est normal sur un conteneur Debian 13, et ce n'est pas une erreur.** Le `Task finished with 1`

Ce support est la propriété de Dawan France.

`warning(s)!` qui suit ne signale que cet avertissement.

La version récente de systemd (257, celle de Trixie) *conseille* d'activer le *nesting*, pour que certaines de ses unités fonctionnent au mieux. Le conteneur démarre et fonctionne parfaitement sans. Si un service précis à l'intérieur s'en plaint, on ajoute `--features nesting=1` (voir §10.1.3) — mais pour un serveur web ordinaire, on l'ignore.

Attention aux adresses IP : votre conteneur prend une **adresse fixe** (jamais DHCP), prise sur la **plage de votre** `vmbr0`, **à partir de** `.50` (voir la règle d'adressage de l'introduction). Elle ne doit pas être déjà utilisée. Ici `10.44.1.50` est un exemple : reportez la plage qui vous est indiquée pour la salle.

À retenir : une **ISO** sert à installer une VM ; un **template** est le système de fichiers prêt à l'emploi d'un conteneur (`pveam update`, `pveam download`). On crée un conteneur par l'assistant **Create CT** ou par `pct create`, en **IP fixe** prise sur la plage du `vmbr0` à partir de `.50`.

Ce support est la propriété de Dawan France.

12. Administrer un conteneur

Le conteneur existe : il faut maintenant le piloter au quotidien — le démarrer et l'arrêter, y entrer, lire et modifier sa configuration, souvent à chaud.

12.1 Le cycle de vie d'un conteneur

Une fois votre conteneur créé, vous interagissez avec lui via un ensemble de commandes `pct` qui forment un vocabulaire cohérent. Comprendre ce cycle de vie est essentiel avant de passer aux opérations plus avancées.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pct list                    # lister tous les conteneurs et leur état
pct start 200               # démarrer le conteneur 200
pct stop 200                # arrêt immédiat (équivalent à couper l'alimentation)
pct shutdown 200            # arrêt propre, confié au systemd du conteneur
(recommandé)
pct reboot 200              # redémarrer le conteneur
pct status 200              # afficher l'état actuel du conteneur
pct destroy 200             # supprimer le conteneur (doit être arrêté au préalable)
```

Résultat de pct list :

```
VMID       Status     Lock         Name
200        running                 webserver
```

La distinction entre `pct stop` et `pct shutdown` est importante. `pct stop` coupe brutalement le conteneur sans lui laisser le temps de terminer ses processus proprement — c'est l'équivalent de débrancher une machine physique. `pct shutdown` demande au premier processus du conteneur, son `systemd`, d'arrêter le système : les services s'arrêtent dans l'ordre, les fichiers sont fermés, exactement comme un `poweroff` tapé à l'intérieur.

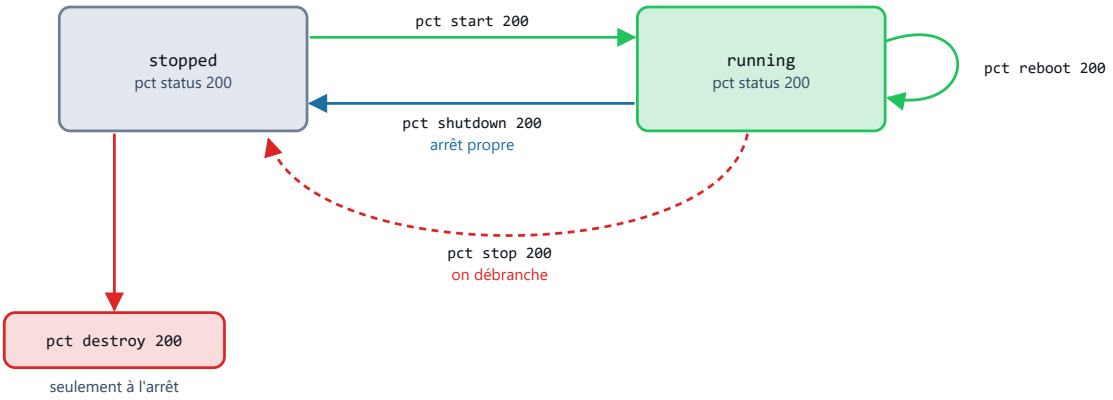
Un conteneur n'a pas de matériel virtuel, donc pas de signal ACPI (*Advanced Configuration and Power Interface*, gestion de l'alimentation) comme une VM. C'est le noyau partagé qui transmet la demande.

Ce support est la propriété de Dawan France.

Si le conteneur ne s'est pas arrêté au bout de 60 secondes, la commande échoue — l'option `--forceStop 1` le coupe alors d'office. Préférez toujours `pct shutdown` pour éviter la corruption des données.

Le cycle de vie d'un conteneur, en commandes pct

Cliquez sur une transition



Deux états seulement, lus par pct list ou pct status : stopped et running.
Chaque bouton de l'interface (Start, Shutdown, Stop, Reboot, Remove) appelle la commande pct correspondante.

Les deux états d'un conteneur et les commandes qui le font passer de l'un à l'autre. Les deux flèches d'arrêt n'ont pas le même sens : `shutdown` laisse les services se terminer, `stop` les coupe. Cliquez sur une transition pour le détail.

12.2 Accéder à un conteneur et l'administrer

12.2.1 La console interactive

Pour accéder à un shell dans votre conteneur, Proxmox propose deux méthodes. La première, `pct enter`, ouvre une session interactive directement dans le contexte du conteneur, en tant que root. Ni mot de passe ni réseau : elle passe par le noyau de l'hôte, pas par SSH.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Ouvrir une session shell interactive dans le conteneur 200
pct enter 200
```

Résultat attendu :

Ce support est la propriété de Dawan France.

```
root@webserver:/#
```

Vous êtes maintenant à l'intérieur du conteneur. Le prompt change pour refléter le hostname du conteneur.

Depuis cette session, vous pouvez travailler comme sur n'importe quel système Debian : installer des paquets, éditer des fichiers de configuration, gérer des services. Pour quitter, tapez `exit` ou `Ctrl+D`.

Ne confondez pas avec `pct console 200`, qui ouvre la console texte du conteneur, celle d'un écran branché. Elle demande un identifiant et un mot de passe ; on en sort par `Ctrl+a` puis `q`. `pct enter` est plus direct, mais ne fonctionne que depuis le nœud qui héberge le conteneur.

12.2.2 Exécuter des commandes sans entrer dans le conteneur

La seconde méthode, `pct exec`, permet d'exécuter une commande dans le conteneur sans ouvrir de session interactive — très utile pour les scripts ou les opérations ponctuelles :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Exécuter bash dans le conteneur (session interactive via exec)
pct exec 200 -- bash

# Mettre à jour les paquets sans entrer dans le conteneur
pct exec 200 -- apt update

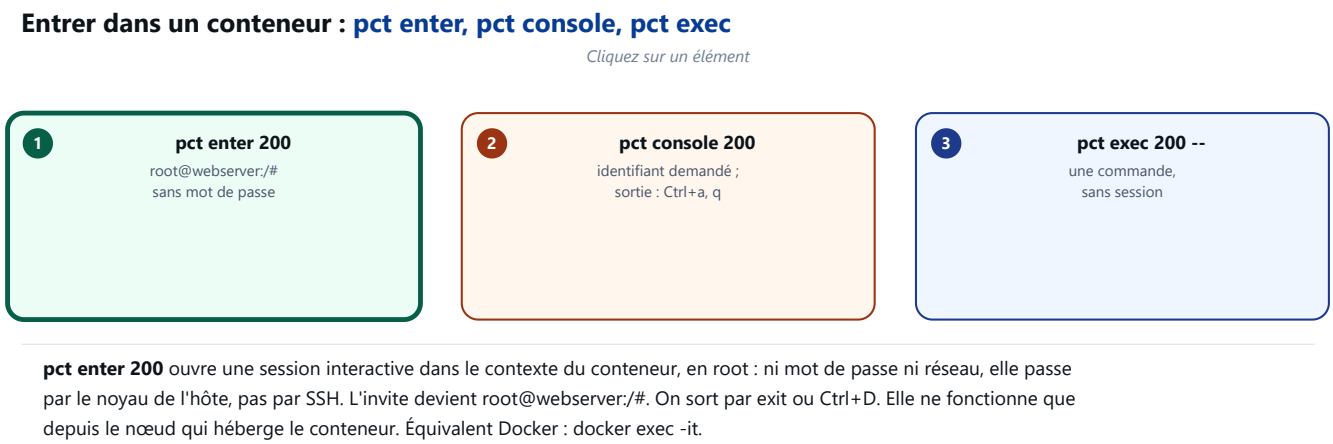
# Vérifier que nginx fonctionne
pct exec 200 -- systemctl status nginx
```

Résultat de apt update :

```
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian-security trixie-security InRelease
Reading package lists... Done
```

Le double tiret `--` est une convention Unix qui sépare les options de `pct exec` de la commande à exécuter à l'intérieur du conteneur. Sans ce séparateur, certaines options pourraient être interprétées par `pct` lui-même plutôt que transmises à la commande cible.

Ce support est la propriété de Dawan France.



Cliquez sur chaque commande : ce qu'elle ouvre, et comment on en sort.

> **Équivalent Docker :** `pct enter` correspond à `docker exec -it` avec un shell interactif. `pct exec` correspond à `docker exec` pour une commande ponctuelle. Si vous avez déjà travaillé avec Docker, ces deux commandes vous seront immédiatement familières.

12.3 Configuration et modification à chaud

12.3.1 Lire la configuration d'un conteneur

Chaque conteneur est décrit par un fichier texte, `/etc/pve/lxc/<VMID>.conf` : la source de vérité pour toutes ses options. Comme tout `/etc/pve/`, il est répliqué sur tous les nœuds d'un cluster.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Afficher la configuration effective du conteneur 200
pct config 200
```

Résultat attendu :

```
arch: amd64
cores: 1
hostname: webserver
memory: 512
```

Ce support est la propriété de Dawan France.

```
net0:
name=eth0,bridge=vmbr0,gw=10.44.1.1,hwaddr=BC:24:11:xx:xx:xx,ip=10.44.1.50/24,type=v
eth
ostype: debian
parent: avant-upgrade
rootfs: local-lvm:vm-200-disk-0,size=8G
swap: 512
unprivileged: 1
```

`pct config` montre l'état courant. Le fichier brut (`cat /etc/pve/lxc/200.conf`) contient en plus une section `[nom]` par snapshot, qui fige la configuration au moment de la prise. C'est là qu'on regarde le jour où un retour arrière semble avoir « perdu » un réglage.

> **N'éditez pas ce fichier à la main pendant que le conteneur tourne** : Proxmox peut le réécrire et perdre la modification. On passe par `pct set` ou par l'interface.

12.3.2 Modifier les ressources à chaud

L'un des avantages des conteneurs LXC par rapport aux VM est la possibilité de modifier certaines ressources sans redémarrer. Proxmox exploite les cgroups v2 pour modifier à chaud la mémoire allouée et le nombre de cœurs CPU :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Augmenter la RAM à 1024 Mo sans redémarrer le conteneur
pct set 200 --memory 1024

# Passer le conteneur à 2 cœurs CPU
pct set 200 --cores 2

# Vérifier que les changements sont pris en compte
pct config 200
```

Résultat de pct config après modification :

```
arch: amd64
cores: 2
```

Ce support est la propriété de Dawan France.

```
hostname: webserver
memory: 1024
...
```

La commande `pct set` modifie simultanément le fichier de configuration et les paramètres actifs du conteneur via les cgroups. Les changements sont persistants : si le conteneur redémarre, il utilisera les nouvelles valeurs.

> **Attention :** on peut augmenter la RAM à chaud. Mais la réduire en dessous de la mémoire utilisée par le conteneur peut provoquer des erreurs, ou forcer l'OOM (*Out Of Memory*) killer à tuer des processus. Vérifiez la consommation mémoire actuelle avant de réduire.

12.3.3 Partager un dossier de l'hôte

Un conteneur peut recevoir, en plus de son disque, un **point de montage** : un répertoire de l'hôte (`pct set 200 --mp0 /srv/partage,mp=/data`) ou un volume Proxmox dédié (`pct set 200 --mp1 local-lvm:5,mp=/var/lib/mysql`).

Le volume Proxmox suit le conteneur dans ses sauvegardes, snapshots et migrations. Le répertoire de l'hôte, lui, n'appartient pas au conteneur : il n'est pas sauvegardé avec lui, et il empêche la migration vers un autre nœud.

> **Dossier partagé et conteneur non privilégié :** le `root` du conteneur correspond à l'UID 100000 de l'hôte. Un répertoire qui appartient à `root` sur l'hôte reste donc inaccessible en écriture depuis le conteneur : `chown 100000:100000 /srv/partage` sur l'hôte avant de démarrer le conteneur.

À retenir : le cycle de vie passe par `pct start/stop/shutdown/reboot` . On entre dans un conteneur par `pct enter` (session) ou on y lance une commande par `pct exec` . Sa configuration vit dans `/etc/pve/lxc/<id>.conf` ; on la modifie par `pct set` ou par l'interface, souvent à chaud (mémoire, cœurs).

Ce support est la propriété de Dawan France.

13. Réseau, sauvegardes et premier service

Il reste à relier le conteneur au réseau, à le protéger avant toute manipulation risquée, puis à y faire tourner un premier vrai service. Cette dernière section du chapitre enchaîne les trois, jusqu'à une page web servie par nginx.

13.1 Réseau des conteneurs LXC

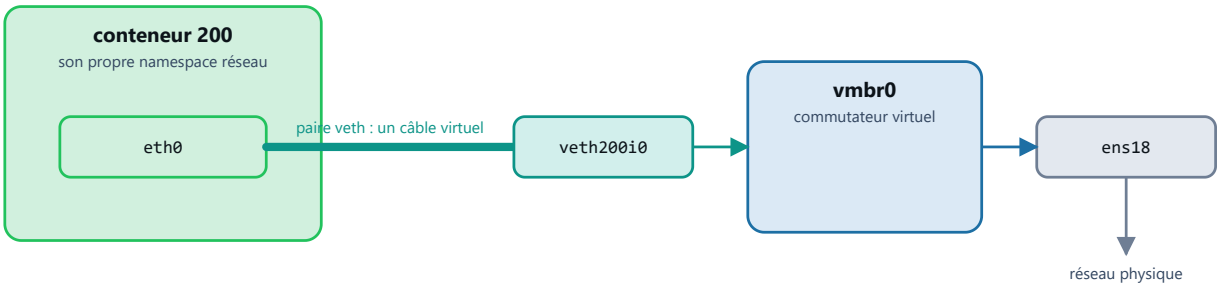
13.1.1 Le bridge vmbr0 et les paires veth

Proxmox configure automatiquement un bridge Linux — `vmbr0` — lors de l'installation. Ce bridge fonctionne comme un commutateur virtuel : il connecte les interfaces réseau des conteneurs (et des VM) au réseau physique de l'hôte.

Chaque interface réseau d'un conteneur LXC est en réalité une **paire veth** (virtual Ethernet) : deux interfaces virtuelles reliées comme les deux extrémités d'un câble. L'une est dans le namespace réseau du conteneur, vue comme `eth0` ; l'autre est sur l'hôte, nommée `vethXXXi0`, et connectée au bridge `vmbr0`.

Du conteneur au câble : une paire veth branchée sur vmbr0

Cliquez sur un élément



Chaque carte réseau d'un conteneur est une paire veth : deux interfaces reliées comme les deux bouts d'un câble.
Un bout est dans le conteneur (eth0), l'autre sur l'hôte (vethXXXi0), branché sur un pont.

Le chemin d'un paquet, du conteneur 200 au réseau physique. `eth0` et `veth200i0` sont les deux bouts d'un même câble virtuel ; `vmbr0` les relie à la carte du nœud. Cliquez sur un élément pour la commande qui le montre.

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Voir les paires veth créées par les conteneurs actifs
# (veth200i0 = conteneur 200, interface net0)
ip link show type veth

# Vérifier quel bridge est associé à cette interface veth
bridge link show | grep veth200

# Voir tous les conteneurs connectés à vmbr0
brctl show vmbr0
```

Résultat attendu de `brctl show vmbr0` (nœud `pxe211` du lab, CT 200 démarré ; vos noms de cartes et d'autres conteneurs peuvent différer). Si la case **Firewall** de la carte du conteneur est décochée (`firewall=0`), la paire veth est branchée directement sur le pont :

```
bridge name bridge id       STP enabled interfaces
vmbr0       8000.bc241101a111   no      ens18
                            veth200i0
```

Chaque `vethXXXi0` est l'extrémité hôte de la paire veth d'un conteneur actif : le nombre est l'identifiant du conteneur, `i0` le numéro de son interface (`net0`). `ens18` est la carte du nœud reliée à `vmbr0` .

`ip link show type veth` affiche de son côté `veth200i0@if2` : le `@if2` est le numéro de l'autre extrémité, l' `eth0` vu de l'intérieur du conteneur.

Avec le pare-feu activé sur la carte (`firewall=1` , la case cochée par défaut), Proxmox intercale un petit pont par conteneur, sur lequel le pare-feu filtre. Sur le même nœud.

Dans le Shell du nœud :

```
# Tous les ponts du nœud, ceux du pare-feu compris
brctl show
```

```
bridge name bridge id       STP enabled interfaces
vmbr0       8000.bc241101a111   no      ens18
                            fwpr200p0
```

Dawan
Ce support est la propriété de Dawan France.

```
fwbr200i0       8000.1ac7a959441a    no      fwln200i0
                                             veth200i0
```

Le chemin devient : `veth200i0` → pont `fwbr200i0` → paire `fwln200i0` / `fwpr200p0` → `vmbr0`. C'est pour cela qu'on voit des interfaces `fwpr…` sur `vmbr0` et des ponts `fwbr…` dans `brctl show` : elles appartiennent au pare-feu Proxmox, pas au conteneur.

13.1.2 IP statique et DHCP

La configuration réseau se précise à la création du conteneur dans l'option `--net0`. Deux modes sont disponibles.

En IP statique, on spécifie l'adresse et le masque en notation CIDR (*Classless Inter-Domain Routing* : l'adresse suivie de la longueur du masque), plus la passerelle.

Dans le Shell du nœud, l'option de `pct create` :

```
# Configuration réseau en IP statique
--net0 name=eth0,bridge=vmbr0,ip=<ip-libre>/24,gw=<ip-de-la-passerelle>
```

En DHCP, le conteneur obtiendra une adresse automatiquement depuis le serveur DHCP du réseau.

Dans le Shell du nœud, l'option de `pct create` :

```
# Configuration réseau en DHCP (hors lab : ici, adresses fixes uniquement)
--net0 name=eth0,bridge=vmbr0,ip=dhcp
```

13.1.3 Vérifier la connectivité depuis le conteneur

Une fois le conteneur démarré, on entre dedans pour vérifier que la configuration réseau est correcte :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis l'hôte Proxmox, entrer dans le conteneur
pct enter 200
```

Ce support est la propriété de Dawan France.

```
# Vérifier les interfaces et l'IP attribuée
ip addr show eth0

# Tester la connectivité vers la passerelle
ping -c 3 <ip-de-la-passerelle>

# Tester la connectivite sur le reseau local
ping -c 3 <ip-de-votre-noeud>
```

Résultat de ip addr show eth0 :

```
2: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether bc:24:11:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 10.44.1.50/24 brd 10.44.1.255 scope global eth0
       valid_lft forever preferred_lft forever
```

L'interface `eth0@if7` indique que cette interface est un veth pair — une paire d'interfaces virtuelles reliées, dont une extrémité (`if7`) se trouve côté hôte, connectée au bridge `vmbr0` .

13.2 Sauvegardes et snapshots

13.2.1 Les snapshots

Un snapshot capture l'état complet d'un conteneur à un instant donné — son système de fichiers, sa configuration, et si on le demande sa mémoire vive. C'est un filet de sécurité précieux avant toute opération risquée : mise à jour majeure, modification de configuration critique, installation d'un nouveau service.

Depuis l'interface graphique, on navigue vers le conteneur, puis **Snapshots** → **Take Snapshot**. On donne un nom explicite au snapshot (par exemple `avant-upgrade-nginx`) et une description facultative.

Pour revenir à un état précédent, on sélectionne le snapshot dans la liste et on clique **Rollback**.

13.2.2 Les sauvegardes avec vzdump

`vzdump` crée une archive complète du conteneur, restaurable sur n'importe quel nœud Proxmox. Il sert aussi aux VM, et Proxmox Backup Server en est la version dédiée.

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Sauvegarder le conteneur 200 en mode snapshot, sans l'arrêter
vzdump 200 --storage local --mode snapshot
# Restaurer l'archive sous un nouvel identifiant ; --unique 1 : nouvelle adresse MAC
pct restore 201 /var/lib/vz/dump/vzdump-lxc-200-<date>.tar.zst --storage local-lvm --unique 1
```

Résultat attendu : `vzdump` termine par `INFO: Finished Backup of VM 200`, et l'archive est dans `/var/lib/vz/dump/`. La restauration crée un volume neuf, `vm-201-disk-0`.

> **La copie reprend tout, adresse IP comprise.** Démarrer le 201 pendant que le 200 tourne met deux machines sur la même adresse : changez l'adresse et le nom avant le premier démarrage (`pct set 201 --net0 …`, `--hostname`).

- 📖 Documentation Proxmox — Backup and Restore

13.3 Déployer nginx dans un conteneur — exemple complet

Pour conclure ce chapitre par un cas concret, voici le déploiement complet d'un serveur nginx dans notre conteneur. Cette séquence illustre le workflow typique qu'on retrouvera tout au long de la formation.

On commence par entrer dans le conteneur et mettre à jour la liste des paquets :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Entrer dans le conteneur
pct enter 200

# Mettre à jour la liste des paquets
apt update
```

Résultat attendu :

```
Hit:1 http://deb.debian.org/debian trixie InRelease
Reading package lists... Done
```

Dawan Nous donnons vos envies
Ce support est la propriété de Dawan France.

On installe nginx et on l'active pour qu'il démarre automatiquement au prochain redémarrage :

Dans le conteneur (sa **Console**, ou `pct enter <ctid>` depuis le Shell du nœud) :

```
# Installer nginx
apt install -y nginx

# Activer le démarrage automatique au boot
systemctl enable nginx

# Vérifier que nginx est bien démarré
systemctl status nginx
```

Résultat de systemctl status nginx :

```
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled)
     Active: active (running) since Fri 2026-05-29 10:15:00 UTC; 3s ago
   Main PID: 312 (nginx)
```

Récupérez l'adresse du conteneur, puis interrogez le service **depuis le Shell du nœud Proxmox** :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis le Shell du noeud
pct exec 200 -- ip addr show eth0
curl -I http://<IP-du-conteneur>   # interroger nginx dans le conteneur, en-têtes seulement
```

Résultat attendu :

```
root@pxe211:~# pct exec 200 -- ip addr show eth0
2: eth0@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether ba:9c:1e:47:2f:08 brd ff:ff:ff:ff:ff:ff
    inet 10.44.1.50/24 brd 10.44.1.255 scope global eth0
       valid_lft forever preferred_lft forever

root@pxe211:~# curl -I http://10.44.1.50
```

Ce support est la propriété de Dawan France.

```
HTTP/1.1 200 OK
Server: nginx
Content-Type: text/html
```

Deux confirmations en une : **le conteneur porte bien l'adresse fixe qui lui a été assignée**, et le service qu'il héberge répond depuis le nœud.

`valid_lft forever` est la signature d'une adresse fixe : une adresse obtenue en DHCP afficherait une durée de bail décomptée. Sur l'infrastructure de lab en `10.44.1.x`, où chaque ressource créée doit porter une adresse fixe, c'est la vérification à faire.

`pct exec 200 -- commande` exécute dans le conteneur sans y ouvrir de session. **Les deux tirets sont indispensables** : ils marquent la fin des options de `pct`, faute de quoi `ip addr show eth0` serait interprété comme des arguments de `pct`.

> **Si `curl` répond `Connection refused`**, le conteneur est joignable mais aucun service n'écoute sur le port 80 — vérifiez avec `pct exec 200 -- systemctl status nginx`. **S'il ne répond rien du tout et finit en timeout**, c'est le pare-feu du nœud ou du conteneur qu'il faut regarder.

Vous devez obtenir `HTTP/1.1 200 OK` et l'en-tête `Server: nginx`. Si c'est le cas, votre conteneur LXC est correctement isolé, connecté au réseau, et héberge un service web fonctionnel.

> **Pourquoi pas simplement le navigateur de votre poste ?** Selon la configuration réseau de la salle, votre poste n'atteint souvent que l'interface Proxmox (port 8006), et pas les adresses des machines du lab.
>
> Taper l'IP du conteneur dans votre navigateur ne donnerait alors rien — et ce ne serait pas une panne du conteneur. La vérification par `curl` depuis le nœud, elle, fonctionne toujours : le nœud est sur le même réseau que ses conteneurs.
>
> Pour voir la page dans un vrai navigateur, il faut une machine **à l'intérieur** du réseau du lab. Typiquement, une VM avec un bureau graphique léger, pilotée par la console Proxmox.

À retenir : chaque conteneur rejoint le pont `vmbr0` par une paire **veth**. Avant une opération risquée, un **snapshot** (retour immédiat) ; pour une vraie sauvegarde, **vzdump** vers un stockage, restaurée par `pct restore`. On vérifie un service depuis le Shell du nœud (`curl`), pas seulement depuis son poste.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Le service de l'atelier 6 : un cache de paquets APT. Le conteneur 200 `cache-apt` fait tourner `apt-cacher-ng`, un mandataire HTTP sur le port 3142.

Une machine réglée pour passer par lui demande ses paquets au cache. Le premier téléchargement d'un paquet va sur Internet, et le cache en garde une copie. Les demandes suivantes du même paquet sont servies par le cache, sans repasser par Internet.

L'intérêt est concret dès qu'on a plusieurs machines : dix VM qui installent les mêmes paquets ne les téléchargent qu'une fois. Les installations vont plus vite, et le lien Internet est moins chargé.

Le cache n'est **pas indispensable** : une VM ou un conteneur sans réglage de mandataire télécharge directement sur Internet. C'est le cas par défaut dans ce support. L'installateur Debian d'une VM propose un champ **Mandataire HTTP** : on peut y mettre l'adresse du cache, ou le laisser vide.

Mais une machine réglée sur le cache **en dépend**. Sur le lab, avec le service arrêté : `apt-get update` échoue sur `Unable to connect to …:3142`. Le conteneur 200 doit donc rester démarré, et démarrer avec le nœud.

> **▶ À pratiquer maintenant — Atelier 6**
>
> Le gros morceau du chapitre : du téléchargement du template jusqu'à un service d'exploitation qui répond. Ce service est un cache de paquets, dont vous prouverez qu'il sert. Entre les deux : le réseau, les ressources et le cycle de vie du conteneur.

Ce support est la propriété de Dawan France.

Ateliers — chapitre 1

> **Infrastructure** : nœud `pxe2<N>1` (Proxmox VE 9), déployé avant la session — `root` / `proxmoxx`

Le plan des disques de la formation — à respecter : chaque disque a un rôle réservé.

Disque	Réservé à
`sda`	Le système Proxmox — **ne jamais y toucher**
`sdc` et `sdd`	Le pool ZFS en miroir
`sdb`	Ceph — un OSD (*Object Storage Daemon*, le service qui gère un disque Ceph) par nœud

Ceph refuse tout disque portant encore une signature de partition, de LVM ou de ZFS. Monter le pool ZFS sur `sdb` obligerait à le détruire et à effacer le disque avant de pouvoir déployer Ceph.

Comment vous accédez au lab — à lire avant de commencer. Vous atteignez votre nœud **uniquement par son interface web en HTTPS**. Selon la salle, vous n'avez ni accès SSH direct, ni route réseau vers les machines que vous allez créer.

Deux conséquences pratiques pour toute la formation :

- **La ligne de commande passe par l'onglet Shell** de l'interface, pas par un client SSH.
- **Pour tester un service qui tourne dans une VM ou un conteneur**, utilisez `curl` depuis ce Shell. Taper l'adresse de la machine dans le navigateur de votre poste ne donnera rien : votre poste n'est pas sur ce réseau.

État du lab en fin de chapitre :

- Nœud Proxmox VE opérationnel, dépôt no-subscription actif, système à jour
- Trois profils RBAC créés (`comptable@pve` restreint à la VM vide 290 `compta` , groupes `dev` / `support` , `admin-jr@pve` sans accès shell)

Ce support est la propriété de Dawan France.

- Double authentification TOTP pratiquée sur un compte de test (atelier 5), puis compte supprimé
- Proxmox VE installé dans la VM 190 de l'atelier 1, puis la VM supprimée (l'ISO reste dans `local`)
- Conteneur 200 (`cache-apt`, `10.<site>.<N>.50`) avec le cache de paquets `apt-cacher-ng`, snapshot testé, sauvegarde vzdump réalisée

---

Atelier 1 — Installer Proxmox VE dans une VM de votre nœud

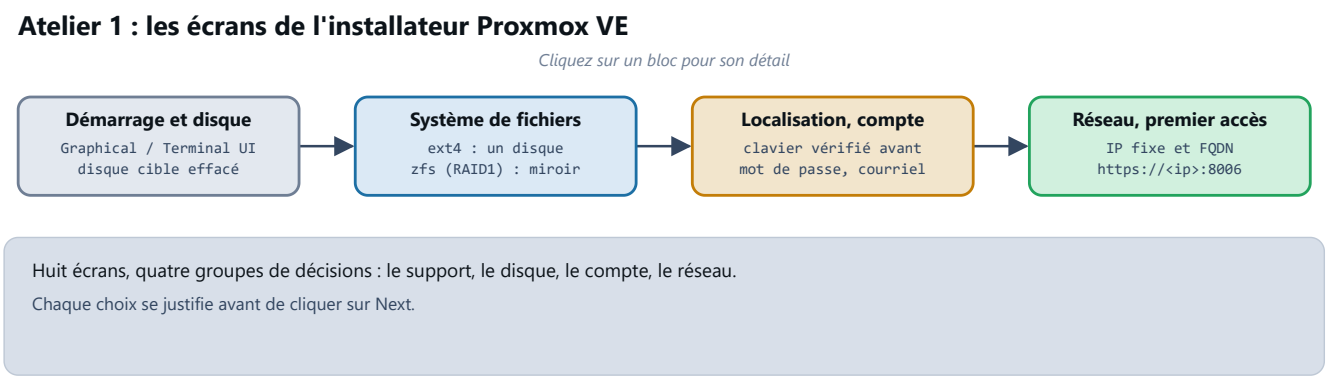
Contexte

Votre nœud de lab a été déployé avant la session, par un démarrage réseau PXE automatisé. Mais un jour, vous poserez Proxmox sur un serveur réel, chez un client ou pour votre propre hyperviseur. C'est alors cet installateur que vous verrez, et ses choix que vous devrez trancher.

Vous installez donc Proxmox VE vous-même, dans une machine virtuelle de votre nœud : une **installation imbriquée**, un Proxmox qui tourne dans un Proxmox. L'installateur est exactement celui d'un serveur réel ; seul le disque est virtuel.

Pour la connexion à l'interface web de votre nœud, suivez le début de l'atelier 2 : adresse fournie par votre formateur, compte `root`, mot de passe `proxmoxx`. La création d'une VM est détaillée au chapitre 2 ; ici, les valeurs à saisir sont données écran par écran.

Résultat à obtenir : Proxmox VE installé dans la VM 190 `pve-essai`, à l'adresse `10.<site>.<N>.90`, dont l'interface répond sur le port 8006. Pour chaque écran de l'installateur, savoir dire ce qu'il règle et justifier la valeur retenue. La VM est supprimée en fin d'atelier.



Ce support est la propriété de Dawan France.

Le schéma regroupe les écrans de l'installateur dans l'ordre où ils défilent. Cliquez sur chaque bloc pour les questions qu'il pose.

Arborescence attendue

```
pxe2<N>1 (votre nœud)
├── local                    ← stockage des fichiers
│   └── ISO Images
│       └── proxmox-ve_9.2-1.iso   ← téléchargée à l'étape 1
└── VM 190 « pve-essai »     ← 2 processeurs, 4 Go de RAM, disque de 32 Go sur
local-lvm
    └── Proxmox VE 9.2 installé : pve-essai.lab.local, 10.<site>.<N>.90
```

Ce que vous devez faire

1. **L'ISO.** Téléchargez l'ISO officielle `proxmox-ve_9.2-1.iso` dans le stockage `local` de votre nœud, par son adresse sur `enterprise.proxmox.com/iso/`.
2. **La VM.** Créez la VM 190 `pve-essai` : l'ISO dans le lecteur, 2 processeurs, 4 Go de RAM, un disque de 32 Go sur `local-lvm`, une carte sur `vmbr0`.
3. **L'installation.** Démarrez la VM, ouvrez sa console, et installez Proxmox VE. Pour chaque écran, répondez à la question avant de valider : - **Menu de démarrage.** Dans quel cas prendrait-on **Terminal UI** plutôt que **Graphical** ? - **Disque cible.** Que devient le contenu du disque choisi ? Sur un serveur à plusieurs disques, que faut-il vérifier ici ? - **Système de fichiers** (bouton **Options**). Quel choix feriez-vous en production sur un serveur à **deux disques identiques**, et pourquoi garder `ext4` ici ? - **Localisation.** Pourquoi faut-il vérifier la disposition du clavier **avant** l'écran suivant ? - **Mot de passe et courriel.** Quelle longueur minimale est exigée ? À quoi sert l'adresse de courriel ? - **Réseau.** Que désigne la **Management Interface** ? Pourquoi un hyperviseur doit-il avoir une adresse **fixe** et un nom complet (FQDN) ? - **Résumé.** Que se passe-t-il si l'ISO reste dans le lecteur au redémarrage ?
4. **Le premier démarrage.** Retirez l'ISO du lecteur, laissez la VM redémarrer, et lisez la console : sur quel port répond l'interface web, et avec quel compte se connecte-t-on ?
5. **La vérification.** Depuis le Shell de votre nœud, vérifiez que l'interface de `pve-essai` répond.
6. **Le nettoyage.** Arrêtez et supprimez la VM 190.

Dawan Nous formons votre savoir
Ce support est la propriété de Dawan France.

▼ **Cliquez pour voir la solution pas à pas**

Étape 1 — L'ISO dans le stockage `local`

Dans l'arbre de gauche, déroulez votre nœud, cliquez sur le stockage **local**, puis sur **ISO Images**. Le bouton **Download from URL** ouvre une fenêtre :

- **URL** : `https://enterprise.proxmox.com/iso/proxmox-ve_9.2-1.iso` ;
- **Query URL** : Proxmox interroge le serveur et remplit le nom du fichier et sa taille ;
- **Download** : le téléchargement se fait directement sur le nœud, sans passer par votre poste.

La tâche apparaît en bas de l'écran ; à la fin, l'ISO figure dans la liste **ISO Images**. Lancée dès votre première connexion, elle se termine pendant le cours : à l'atelier, l'ISO est déjà là.

Étape 2 — La VM 190

Le bouton **Create VM**, en haut à droite, ouvre l'assistant. Les valeurs à saisir, onglet par onglet :

- **General** : VM ID `190`, Name `pve-essai` ;
- **OS** : Storage `local`, ISO image `proxmox-ve_9.2-1.iso`, Type `Linux` ;
- **System** : les valeurs proposées ;
- **Disks** : Storage `local-lvm`, Disk size `32` ;
- **CPU** : Cores `2` ;
- **Memory** : `4096` ;
- **Network** : Bridge `vmbr0`, Model `VirtIO` ;
- **Confirm** : relisez le résumé, puis **Finish**.

Étape 3 — L'installation, écran par écran

Sélectionnez la VM 190, **Start**, puis **Console** : la console noVNC affiche l'écran de la VM dans le navigateur.

> **Les captures ci-dessous ont été faites sur le lab**, avec l'ISO officielle `proxmox-ve_9.2-1.iso`, dans une VM de test de 4 Go, 2 processeurs et 32 Go, nommée `pve-demo`. Dans votre VM, seuls changent le nom et l'adresse IP.

Ce support est la propriété de Dawan France.

Les valeurs à saisir : système de fichiers `ext4` ; Country `France`, Time zone `Europe/Paris`, Keyboard Layout `French` ; mot de passe `proxmoxx`.

Pour le réseau : Hostname (FQDN) `pve-essai.lab.local`, IP `10.<site>.<N>.90/24`, Gateway `10.<site>.<N>.1`. Le DNS est celui de votre nœud, affiché dans **Nœud** → **System** → **DNS**.

1 — Menu de démarrage



La première ligne donne la version exacte de l'ISO : `Proxmox VE 9.2 (iso release 1)`. **Graphical** est l'assistant à la souris.

Terminal UI est le même assistant en mode texte. On le prend quand la carte graphique du serveur pose problème, ou à travers une console distante qui affiche mal le mode graphique. **Advanced Options** donne accès aux modes de dépannage et à l'installation automatisée (cours, section 4.3).

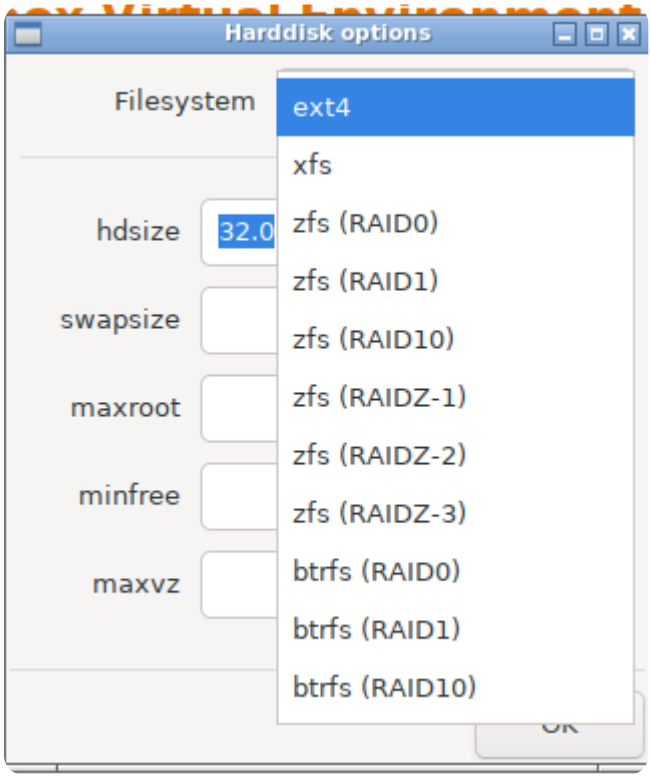
2 — Disque cible

Ce support est la propriété de Dawan France.



Target Harddisk désigne le disque qui sera **entièrement effacé** : toutes ses partitions et données sont perdues, le texte de droite le rappelle. Sur un serveur à plusieurs disques, on vérifie ici le nom et la taille du disque, pour ne pas installer sur celui qui porte des données.

3 — Système de fichiers



Ce support est la propriété de Dawan France.

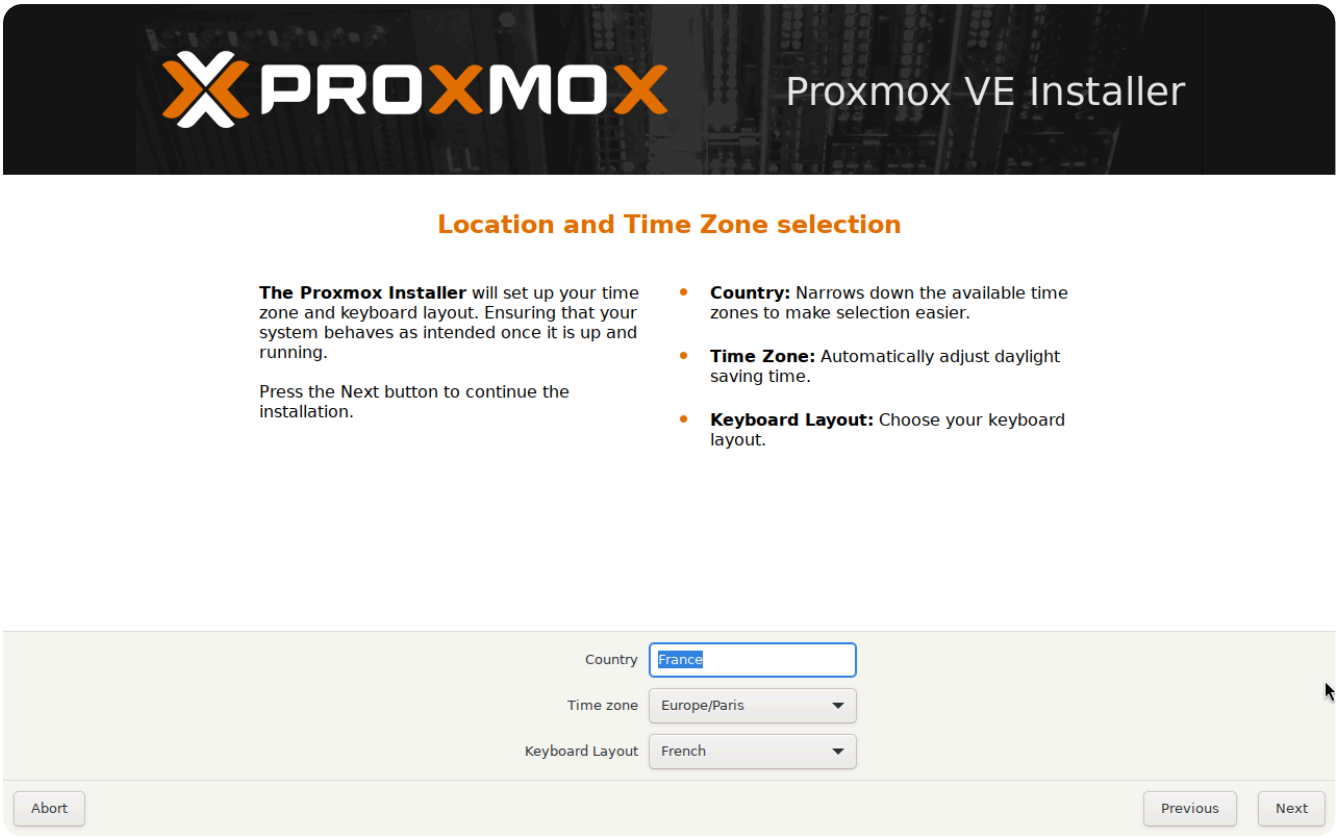
Système de fichiers proposé	Disques	Usage
ext4	1	Serveur simple, installation la plus rapide
xfs	1	Alternative à ext4, à l'aise avec les gros fichiers
zfs (RAID0)	1 ou plus	ZFS sans redondance : un disque perdu, tout est perdu
zfs (RAID1)	2 ou plus	Miroir : le système survit à la perte d'un disque
zfs (RAID10)	4 ou plus	Miroirs agrégés : redondance et performances
zfs (RAIDZ-1 / -2 / -3)	3, 4, 5 ou plus	Parité simple, double ou triple (1, 2 ou 3 disques peuvent tomber)
btrfs (RAID0 / RAID1 / RAID10)	1, 2, 4 ou plus	Autre système de fichiers à instantanés, moins répandu que ZFS sous Proxmox

En production, sur deux disques identiques : **zfs (RAID1)**. Proxmox installe son propre système en miroir sur les deux disques ; si l'un tombe en panne, le serveur continue de fonctionner.

Dans la VM d'essai, on garde **ext4** : une seule VM, un seul disque virtuel, rien à protéger. Sous la liste, les champs `hdsize`, `swapsize`, `maxroot`, `minfree` et `maxvz` règlent le découpage du disque ; laissés vides, ils prennent des valeurs automatiques.

4 — Localisation

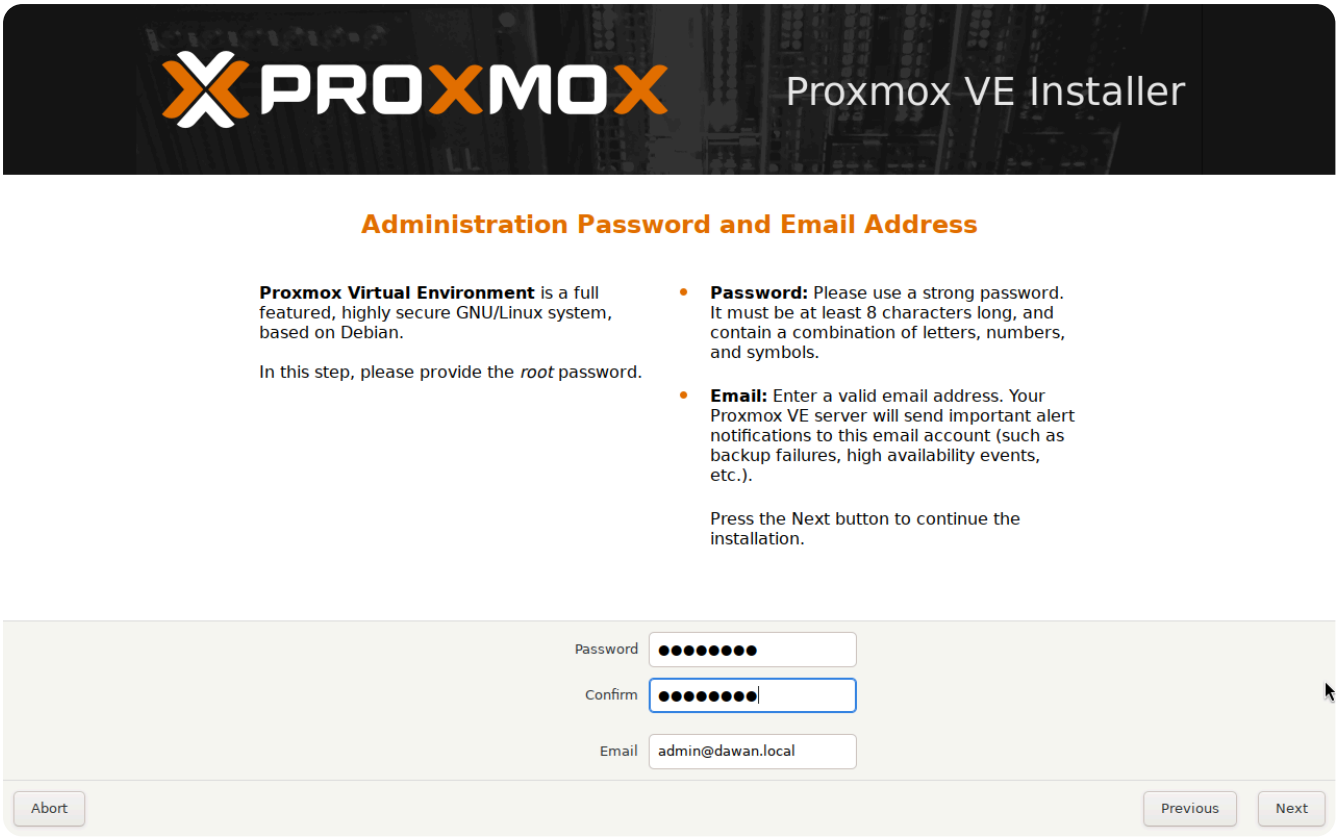
Ce support est la propriété de Dawan France.



Country `France`, Time zone `Europe/Paris`, Keyboard Layout `French`. Le clavier se vérifie **avant** le mot de passe. Avec une disposition fausse, le mot de passe tapé à l'écran suivant n'est pas celui que l'on croit, et l'on ne peut plus se connecter.

5 — Mot de passe et courriel

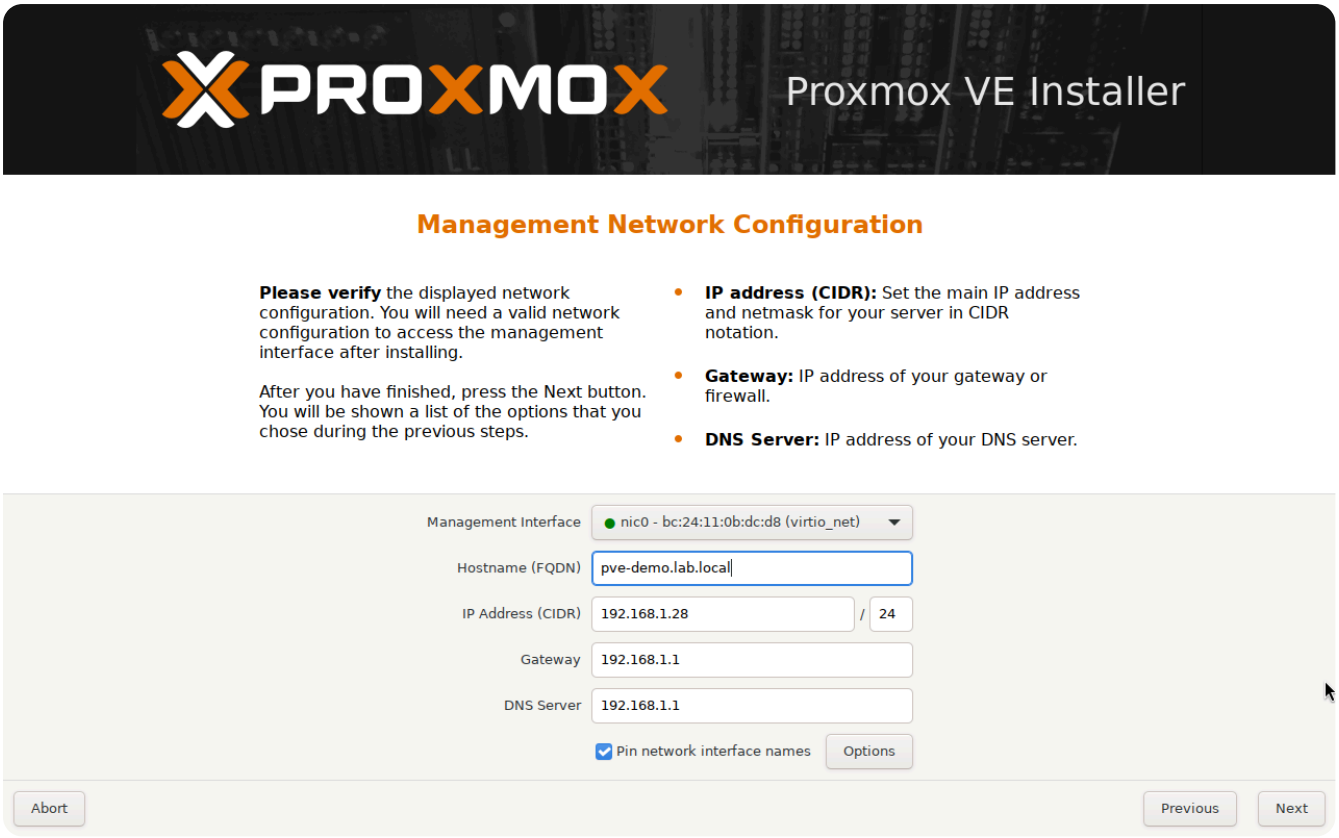
Ce support est la propriété de Dawan France.



Le mot de passe doit faire **au moins 8 caractères** : c'est pour cela que celui du lab s'écrit `proxmoxx`, avec deux x. Le courriel reçoit les alertes du nœud : sauvegarde en échec, événement de haute disponibilité. La valeur par défaut, `mail@example.invalid`, est à remplacer.

6 — Réseau

Ce support est la propriété de Dawan France.

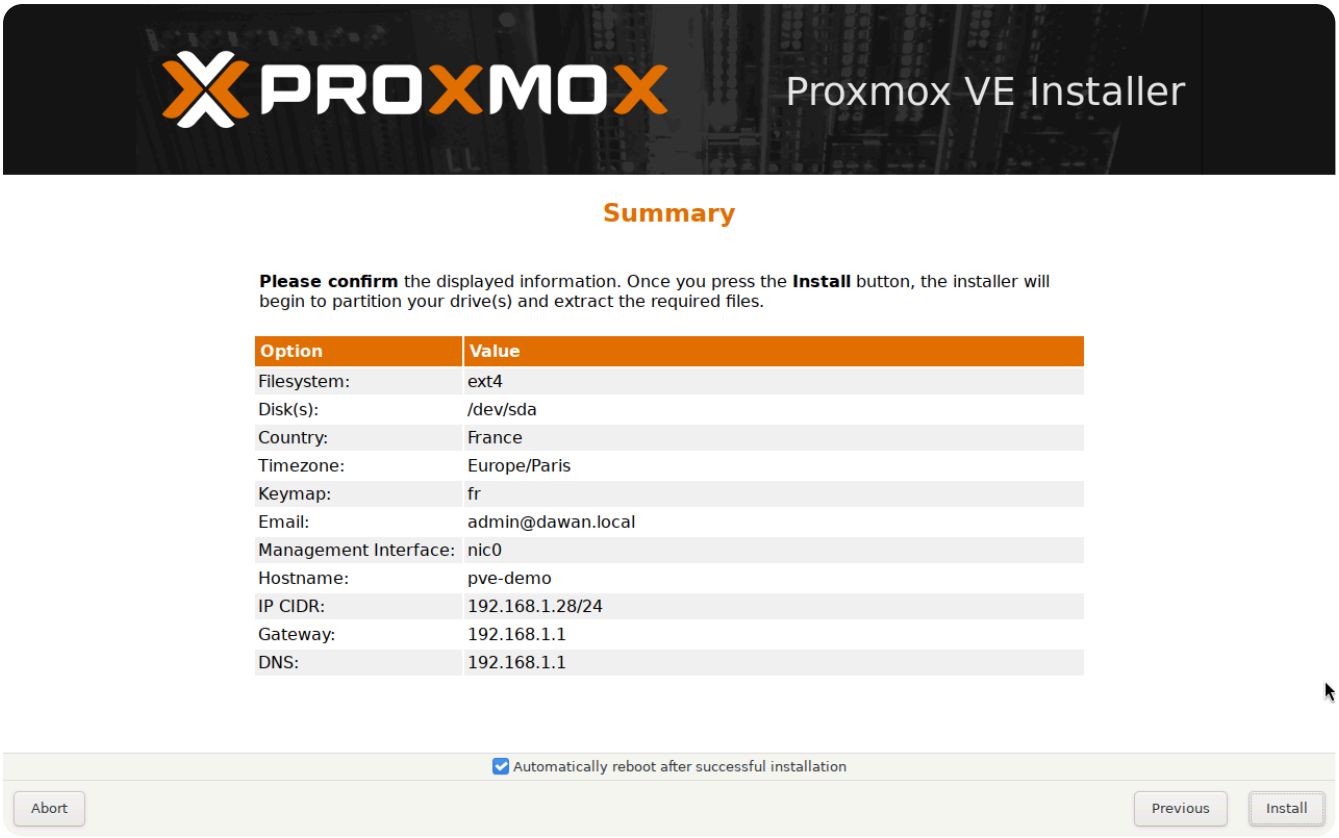


Management Interface est la carte qui portera l'interface web ; en 9.2, elle s'appelle `nic0`. La case **Pin network interface names**, cochée par défaut, fige les noms des cartes : ils ne changeront pas si l'on en ajoute une.

L'hyperviseur a une adresse **fixe** parce que tout le reste s'y adresse : votre navigateur, les autres nœuds du cluster, le serveur de sauvegarde. Le **FQDN** (`nom.domaine`) identifie le nœud dans le cluster et dans ses certificats ; la valeur proposée, `pve.home`, est à remplacer.

7 — Résumé

Ce support est la propriété de Dawan France.



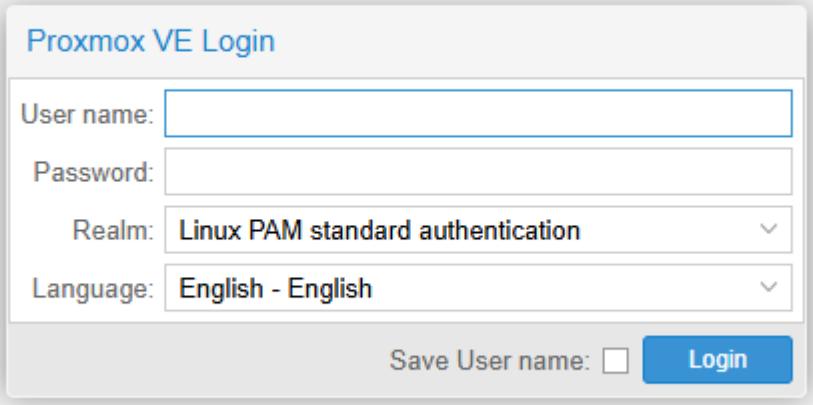
Le tableau récapitule tous les choix ; **Previous** permet encore de corriger. Si l'ISO reste dans le lecteur, la machine redémarre dessus et revient au menu de l'étape 1. On retire donc l'ISO avant le redémarrage : c'est l'étape 4 de la solution.

8 — Premier démarrage



La console donne l'adresse de l'interface, `https://<IP>:8006/` : le port **8006** est celui de l'interface web de Proxmox. L'invite reprend le nom court choisi : `pve-demo login:` sur la capture, `pve-essai login:` dans votre VM.

Ce support est la propriété de Dawan France.



La fenêtre **Proxmox VE Login** est celle que l'on obtient sur un poste qui a une route vers la nouvelle machine. Votre poste n'en a pas : l'étape 5 vérifie la réponse de l'interface depuis le Shell de votre nœud. Le realm proposé par défaut est **Linux PAM standard authentication** : c'est le compte `root` du système, avec le mot de passe de l'écran 5.

Votre propre nœud a été installé avec les mêmes choix, mais sans personne devant l'écran : un fichier de réponses les a fournis. La section 4.3 du cours en donne le principe.

Étape 4 — Retirer l'ISO

À la fin de l'installation, l'installateur annonce le redémarrage. Il faut alors vider le lecteur optique virtuel de la VM, celui des CD et des DVD (*Digital Versatile Disc*).

Avant le redémarrage, ou dès que le menu de démarrage réapparaît : **Hardware** → **CD/DVD Drive** → **Edit** → **Do not use any media**, puis **Reset** si le menu est affiché. La VM démarre alors sur son disque, et la console montre l'écran de l'étape 8.

Étape 5 — La vérification, depuis le Shell du nœud

Votre poste n'a pas de route vers `10.<site>.<N>.90` : la vérification se fait depuis le Shell de votre nœud, qui est sur le même réseau.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
ping -c 2 10.<site>.<N>.90                                    # la VM répond-elle ?
```

Ce support est la propriété de Dawan France.

```
curl -k -s -o /dev/null -w "%{http_code}\n" https://10.<site>.<N>.90:8006   # le code HTTP de l'interface web
```

Résultat attendu (les temps diffèrent) :

```
PING 10.<site>.<N>.90 (10.<site>.<N>.90) 56(84) bytes of data.
64 bytes from 10.<site>.<N>.90: icmp_seq=1 ttl=64 time=0.412 ms
64 bytes from 10.<site>.<N>.90: icmp_seq=2 ttl=64 time=0.388 ms
…
200
```

`-k` accepte le certificat auto-signé de la nouvelle installation ; `-o /dev/null` jette la page, et `-w` n'affiche que le code HTTP. `200` : l'interface web de `pve-essai` répond, l'installation a abouti.

Étape 6 — Le nettoyage

Sélectionnez la VM 190, **Shutdown**, puis **More** → **Remove** : cochez **Purge from job configurations** et confirmez en tapant `190`. Le disque de 32 Go est libéré sur `local-lvm`.

L'ISO peut rester dans `local` : elle resservira au chapitre 2.

Votre propre nœud a été installé avec les mêmes choix, mais sans personne devant l'écran : un fichier de réponses les a fournis. La section 4.3 du cours en donne le principe.

Atelier 2 — Prise en main de l'interface et des stockages

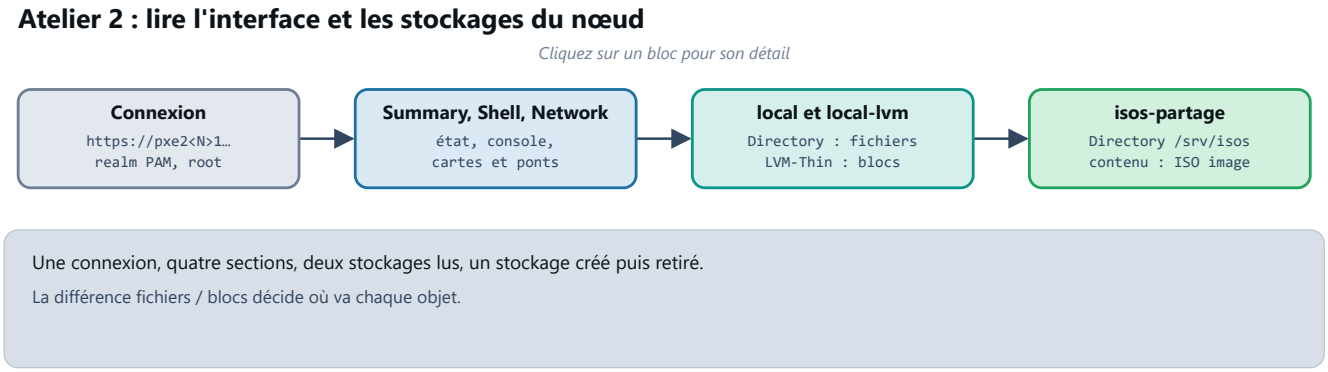
Contexte

C'est votre première connexion au nœud sur lequel vous travaillerez toute la formation. Avant de créer quoi que ce soit, il faut savoir lire l'interface : où se trouve quoi, et surtout quels stockages existent déjà et à quoi ils servent. C'est cette lecture qui vous évitera, plus tard, de déposer une ISO là où seuls des disques de VM ont leur place.

Prenez le temps de cette exploration : chaque section que vous ouvrez maintenant est une section vers laquelle un atelier vous renverra dans les jours qui viennent.

Ce support est la propriété de Dawan France.

Résultat à obtenir : être connecté à l'interface web du nœud, et savoir situer les sections Summary, Shell, Network et Storage. Pouvoir dire ce que contiennent `local` et `local-lvm`, et pourquoi ils diffèrent. Puis avoir créé, vérifié et retiré un troisième stockage, réservé aux ISO.



Le schéma suit l'atelier de la connexion jusqu'au stockage `isos-partage`. Cliquez sur chaque bloc pour ce qu'il faut y relever.

Arborescence attendue

```
Aucun fichier produit : exploration de l'interface web uniquement.
Ce que vous devez savoir situer à la fin :

Datacenter
└── nœud pxe2<N>1
    ├── Summary   ← état, charge, uptime du nœud
    ├── Shell     ← la ligne de commande, faute d'accès SSH direct
    ├── Network   ← les bridges (vmbr0…)
    └── Storage
        ├── local       ← type « répertoire » : ISOs, templates, sauvegardes
        └── local-lvm   ← type « LVM-thin » : disques de VMs et CT
```

Ce que vous devez faire

1. **Vous connecter** à l'URL de votre nœud 1, `https://pxe2<N>1.<ville>.pm.dawan.ovh` (donnée par le formateur), avec le realm **PAM standard** (l'utilisateur système `root`), pas un realm PVE.
2. **Parcourir les quatre sections principales** du nœud — Summary, Shell, Network, Storage — et formuler pour chacune, en une phrase, ce à quoi elle sert.

Ce support est la propriété de Dawan France.

3. **Ouvrir les deux stockages par défaut** et répondre à deux questions pour chacun : quel est son **type**, et quel **type de contenu** il accepte. Déduisez-en pourquoi une ISO ne peut pas aller sur `local-lvm`.
4. **Un cran plus loin : créer un stockage réservé aux ISO.** Déclarez un stockage de type répertoire, `isos-partage`, sur le dossier `/srv/isos`, qui n'accepte **que** des images ISO. Vérifiez à qui il est proposé (ISO ? disques ?), comparez sa taille à celle de `local`, puis retirez-le et regardez ce qui reste sur le disque.

   💡 La distinction `local` / `local-lvm` n'est pas un détail : c'est elle qui explique, tout au long de la formation, pourquoi tel stockage apparaît ou non dans telle liste déroulante.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Connexion et exploration

Ouvrez un navigateur à l'URL de votre nœud 1 (`https://pxe2<N>1.<ville>.pm.dawan.ovh`). Pas d'avertissement de certificat : l'URL Dawan porte un certificat reconnu.

Connectez-vous avec `root` / `proxmoxx`, realm **Linux PAM standard authentication**.

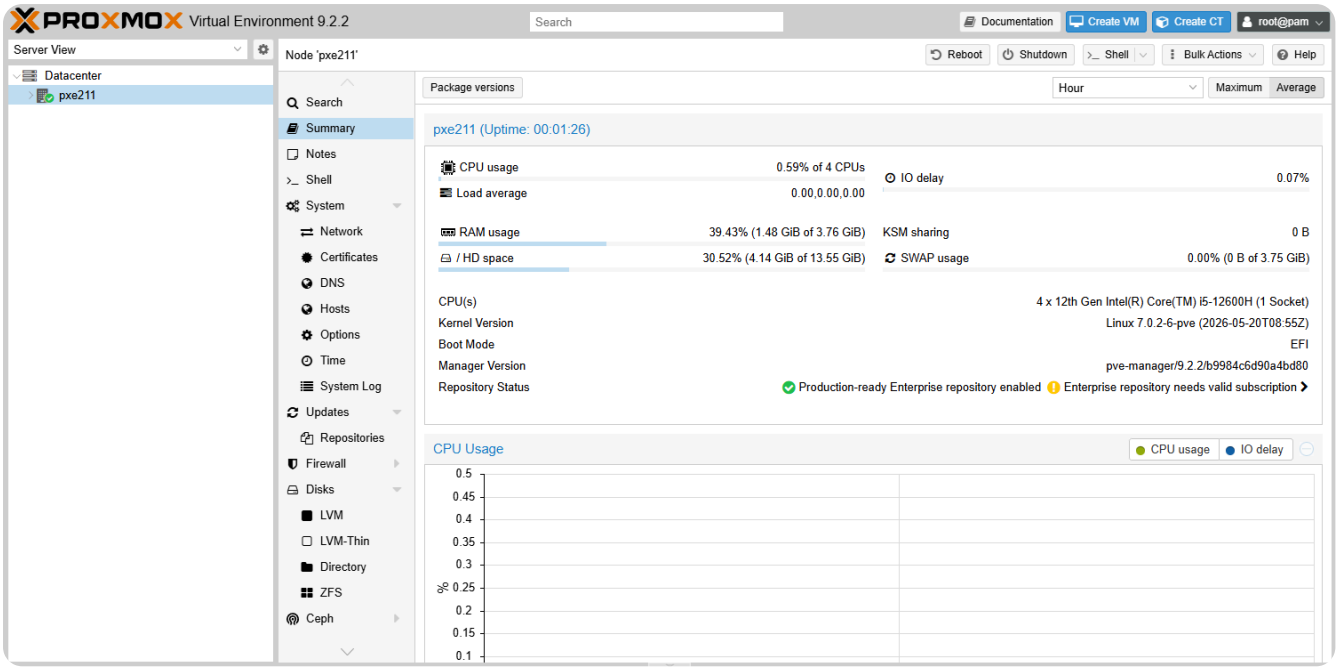
Cliquez sur votre nœud dans l'arborescence de gauche, puis explorez le menu du milieu :

Section	Rôle
Summary	Vue d'ensemble : charge CPU/RAM/disque, uptime, versions, état des dépôts
Shell	Terminal du nœud dans le navigateur, sans client SSH
System → Network	Interfaces et bridges réseau du nœud
System → DNS	Résolveurs DNS utilisés par le nœud
Updates	Paquets à mettre à jour
Updates → Repositories	Dépôts APT actifs (enterprise ou no-subscription)
Disks	Disques physiques et création de stockages (LVM, LVM-Thin, ZFS…)

Les stockages n'ont pas d'entrée dans ce menu : ils apparaissent **sous le nœud**, dans l'arborescence de gauche, et dans `Datacenter → Storage`.

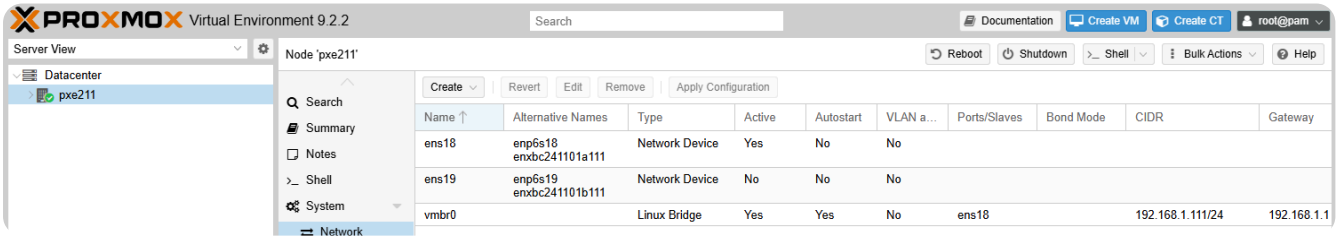
Résultat attendu (nœud `pxe211`, Proxmox VE 9.2.2 juste installé) :

Ce support est la propriété de Dawan France.



L'en-tête donne le nom du nœud et son uptime. Le premier bloc donne la charge : CPU, mémoire, espace disque de `/`, swap. Le second donne l'identité du système : processeurs, **version du noyau** (`7.0.2-6-pve`), mode de démarrage (EFI), **version de Proxmox** (`pve-manager/9.2.2`).

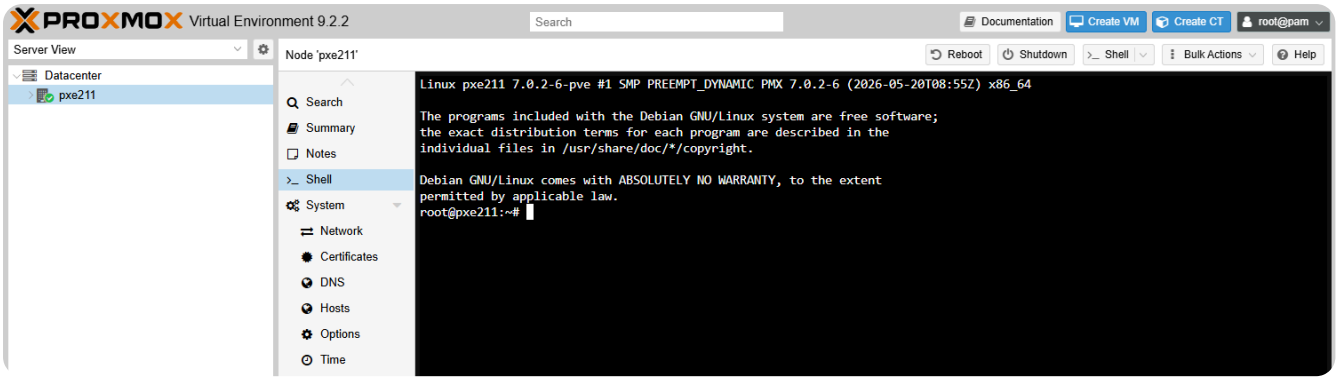
La ligne **Repository Status** est à regarder dès le premier jour : sur un nœud neuf, elle signale un dépôt *Enterprise* activé qui **exige un abonnement valide**. C'est l'objet de l'atelier suivant. En bas, le journal des tâches montre déjà un `Update package database` en erreur, pour la même raison.



Deux cartes réseau (`ens18`, `ens19`) et un bridge, `vmbr0`. Le bridge porte l'adresse du nœud (colonne CIDR) et la passerelle ; sa colonne **Ports/Slaves** indique la carte physique qui le relie au réseau (`ens18`).

La colonne **Active** montre que `ens19` n'est pas utilisée.

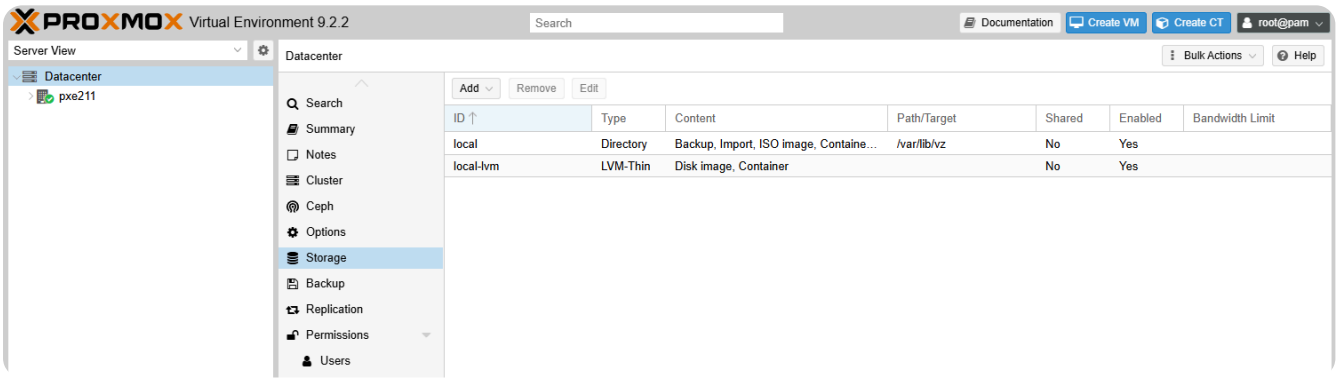
Ce support est la propriété de Dawan France.



Un terminal `root` ouvert directement dans le navigateur. C'est par lui que passeront toutes les commandes de la formation, puisque vous n'avez pas d'accès SSH direct.

Étape 2 — Identifier les stockages par défaut

Naviguez vers `Datacenter → Storage` :



Les deux stockages créés par l'installation. Lisez les colonnes **Type** et **Content** : c'est le type qui décide du contenu accepté.

Stockage	Type	Contenu autorisé par défaut
`local`	Directory (`/var/lib/vz`)	`iso` (images ISO), `vztmpl` (templates de conteneurs), `backup` (sauvegardes vzdump), `import` (disques à importer)
`local-lvm`	LVM-Thin (pool `data` du groupe `pve`)	`images` (disques de VM), `rootdir` (disques de conteneurs)

Réponse à la question de l'énoncé : une ISO est un **fichier**. Or `local-lvm` ne stocke que des **volumes** en mode bloc, utilisés comme disques.

Il n'a pas de système de fichiers où déposer une ISO, et Proxmox ne propose donc pas ce type de contenu. Les ISO vont sur `local` .

Ce support est la propriété de Dawan France.

> **À retenir :** `local` stocke des **fichiers** (ISO, templates, archives), `local-lvm` stocke des **blocs** (volumes logiques utilisés comme disques). Les *snippets* (fichiers cloud-init personnalisés) ne sont pas activés par défaut : il faut les ajouter au contenu de `local` pour s'en servir.

Étape 3 — Un stockage réservé aux ISO

Un stockage Proxmox, c'est un **emplacement** plus la liste des **contenus** qu'il accepte. On en crée un de type répertoire, limité aux ISO.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le dossier qui portera le stockage
mkdir -p /srv/isos
```

À la souris : **Datacenter** → **Storage** → **Add** → **Directory** :

Champ	Valeur
ID	`isos-partage`
Directory	`/srv/isos`
Content	`ISO image` seulement (décochez le reste)

En ligne de commande, la même déclaration.

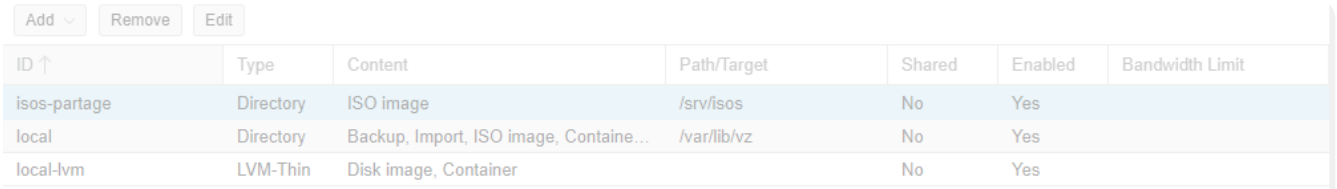
Dans le Shell du nœud :

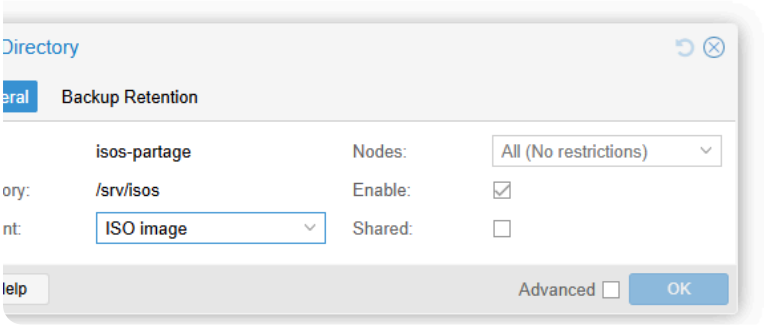
```
# Un stockage répertoire, qui n'accepte que des ISO (et crée le dossier s'il manque)
pvesm add dir isos-partage --path /srv/isos --content iso --create-base-path 1
# Les stockages qui acceptent des ISO…
pvesm status --content iso
# … et ceux qui acceptent des disques de VM
pvesm status --content images
```

Résultat attendu :

Ce support est la propriété de Dawan France.

```
Name                 Type     Status     Total (KiB)      Used (KiB) Available
(KiB)        %
isos-partage          dir     active        14210064         6444344
7022092   45.35%
local                 dir     active        14210064         6444344
7022092   45.35%
Name             Type     Status     Total (KiB)      Used (KiB) Available (KiB)
%
local-lvm     lvmthin     active        12439552         1105876        11333675
8.89%
```





La liste compte maintenant trois stockages. `isos-partage` est de type **Directory**, sur `/srv/isos`, avec pour seul contenu **ISO image**. La fenêtre **Edit** (double-clic sur la ligne) montre les mêmes réglages ; **Nodes : All** veut dire qu'il est déclaré pour tous les nœuds d'un cluster.

Deux constats à retenir :

- **Il n'apparaît que là où il sert.** Proposé pour les ISO, il est absent de la liste des stockages qui acceptent des disques. Dans l'assistant de création d'une VM, il ne figurera pas parmi les emplacements de disque.
- **Il n'ajoute aucun espace.** Son total est **identique** à celui de `local` : les deux dossiers sont sur le même disque système. Un stockage répertoire organise un disque, il ne

Ce support est la propriété de Dawan France.

l'agrandit pas.

Retirez-le.

Dans le Shell du nœud :

```
# Retirer la déclaration du stockage
pvesm remove isos-partage
# Le dossier, lui, est toujours là
ls -d /srv/isos
```

Résultat attendu : `pvesm status` ne liste plus que `local` et `local-lvm`, mais `ls` répond `/srv/isos` : **retirer un stockage ne supprime pas ses fichiers**. Proxmox oublie l'emplacement, le disque garde son contenu. Supprimez le dossier vous-même s'il ne sert plus (`rm -r /srv/isos`).

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesm status   # état des stockages du nœud
```

Résultat attendu :

```
Name             Type     Status     Total (KiB)      Used (KiB) Available (KiB)
%
local             dir     active        14210064         4337680         9128756
30.53%
local-lvm     lvmthin     active        12439552               0        12439552
0.00%
```

`local` et `local-lvm` sont tous deux `active`, de types `dir` et `lvmthin`. Les tailles dépendent du disque de votre nœud.

Atelier 3 — Dépôts APT et mise à jour du système

Ce support est la propriété de Dawan France.

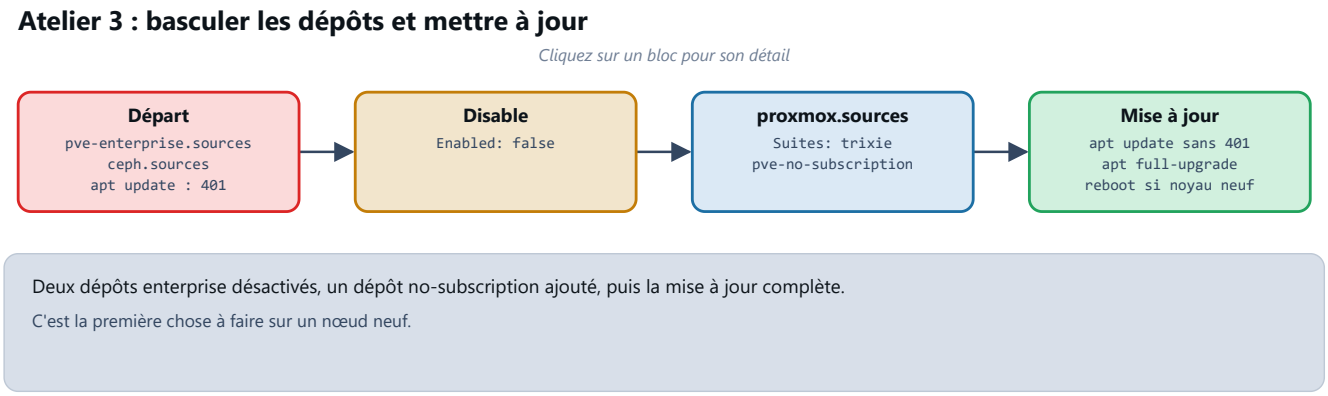
Contexte

Une installation neuve de Proxmox pointe vers les dépôts **enterprise**, réservés aux détenteurs d'un abonnement payant. Il y en a deux : celui de Proxmox VE et celui de Ceph.

Sans abonnement, `apt update` échoue sur une erreur 401, et le système ne se met plus à jour. Beaucoup de nœuds de test restent dans cet état.

Cet atelier bascule le nœud sur le dépôt **no-subscription**, adapté à un lab ou à une petite infrastructure, puis applique la première mise à jour complète. C'est la toute première chose à faire sur un Proxmox fraîchement installé.

Résultat à obtenir : un nœud dont les deux dépôts enterprise sont désactivés, le dépôt no-subscription actif, et sur lequel `apt update` puis `apt full-upgrade` passent sans erreur.



Le schéma part de l'erreur 401 et aboutit au nœud à jour. Cliquez sur chaque bloc pour le fichier ou la commande concernés.

Arborescence attendue

```
Fichiers de dépôts APT touchés (format DEB822, PVE 9) :

/etc/apt/sources.list.d/
├── pve-enterprise.sources   ← à DÉSACTIVER (Enabled: false)
├── ceph.sources             ← à DÉSACTIVER aussi (dépôt Ceph enterprise)
├── debian.sources           ← dépôts Debian, inchangés
└── proxmox.sources          ← à créer : dépôt no-subscription
```

Ce que vous devez faire

Ce support est la propriété de Dawan France.

1. **Lire l'état de départ** dans `Updates → Repositories` : quels dépôts sont actifs, et que dit le bandeau **Status** ?
2. **Désactiver les deux dépôts enterprise** depuis l'interface, puis constater dans le Shell ce que le bouton a modifié.
3. **Activer le dépôt no-subscription**, par le bouton *Add* de la même page ou en écrivant le fichier `.sources` au format DEB822 (blocs de lignes `clé: valeur`) depuis le Shell.
4. **Mettre à jour** : `apt update` doit passer **sans erreur 401**, puis `apt full-upgrade`. Si le noyau a été mis à jour, un redémarrage est nécessaire — sachez le reconnaître.

   ⚠️ Le format a changé en PVE 9 : les dépôts sont décrits dans des fichiers `.sources` (DEB822), plus dans l'ancien `sources.list` d'une seule ligne. Vérifiez que vous éditez le bon.

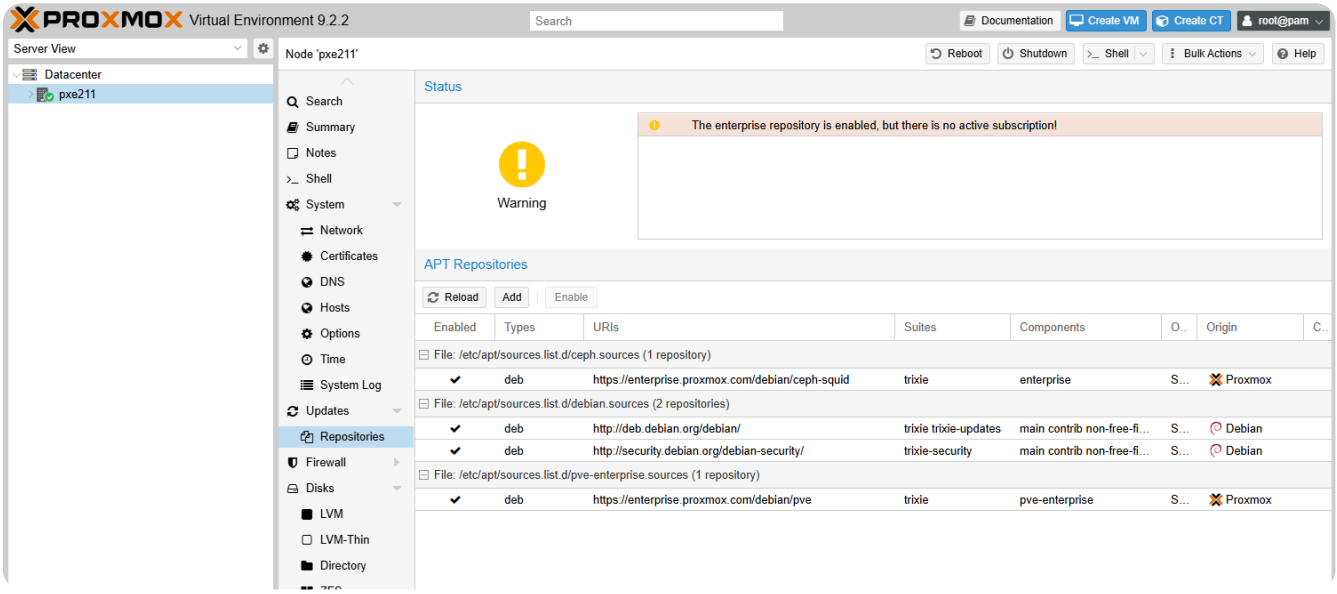
Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe211` juste après son installation depuis l'ISO Proxmox VE 9.2 (`pve-manager/9.2.2`). Les captures et les sorties ci-dessous viennent de ce test sur le lab.

Étape 1 — Lire l'état de départ

Dans le panneau de votre nœud, ouvrez **Updates** → **Repositories**.



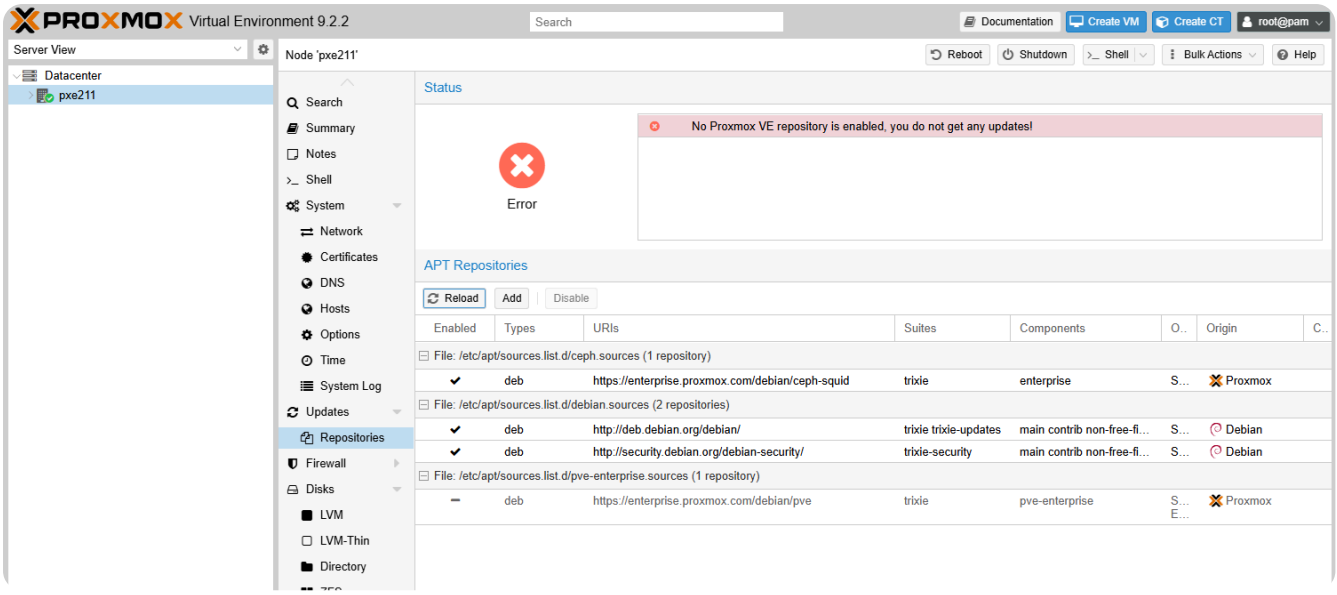
Le bandeau **Status** avertit : *The enterprise repository is enabled, but there is no active subscription!* La liste est groupée par fichier. Deux dépôts Proxmox sont cochés dans la colonne **Enabled** : `ceph-squid` (fichier `ceph.sources`, composant `enterprise`) et `pve` (fichier `pve-enterprise.sources`, composant `pve-enterprise`).

Les deux pointent vers `enterprise.proxmox.com`. Les deux dépôts Debian, eux, sont gratuits et restent actifs.

Étape 2 — Désactiver les deux dépôts enterprise

Sélectionnez la ligne `https://enterprise.proxmox.com/debian/pve` et cliquez **Disable**.

Ce support est la propriété de Dawan France.



La ligne passe en grisé, et un tiret remplace la coche dans **Enabled**. Le bandeau devient rouge : *No Proxmox VE repository is enabled, you do not get any updates!* C'est normal : aucun dépôt Proxmox n'est actif tant que le no-subscription n'est pas ajouté.

Faites de même pour la ligne `https://enterprise.proxmox.com/debian/ceph-squid`. C'est l'oubli le plus fréquent. Sur le lab, avec seulement le dépôt PVE désactivé, `apt update` échouait encore.

Dans le Shell du nœud :

```
# Relire la liste des paquets
apt update
```

```
Err:6 https://enterprise.proxmox.com/debian/ceph-squid trixie InRelease
  401  Unauthorized [IP: 51.91.38.34 443]
Error: Failed to fetch https://enterprise.proxmox.com/debian/ceph-
squid/dists/trixie/InRelease  401  Unauthorized [IP: 51.91.38.34 443]
Error: The repository 'https://enterprise.proxmox.com/debian/ceph-squid trixie
InRelease' is not signed.
```

Le bon dépôt Ceph sera écrit par la commande d'installation de Ceph elle-même.

Ouvrez ensuite le **Shell** du nœud et relisez ce que le bouton **Disable** a modifié :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
cat /etc/apt/sources.list.d/pve-enterprise.sources   # lire le fichier du dépôt enterprise
```

Résultat attendu :

```
Types: deb
URIs: https://enterprise.proxmox.com/debian/pve
Suites: trixie
Components: pve-enterprise
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
Enabled: false
```

La dernière ligne, `Enabled: false`, est tout ce que fait **Disable** : le dépôt reste décrit, apt l'ignore. C'est plus sûr qu'une suppression.

Le jour où l'entreprise prend un abonnement, il suffit de retirer cette ligne. `ceph.sources` a reçu la même ligne.

Étape 3 — Ajouter le dépôt no-subscription

Déclarez le dépôt no-subscription dans son propre fichier. Le bouton **Add** de la page Repositories écrirait le même contenu.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Dépôt no-subscription de PVE 9, au format DEB822
nano /etc/apt/sources.list.d/proxmox.sources          # un fichier neuf
```

Le contenu de `proxmox.sources`, à taper dans `nano`, puis **Ctrl + O**, Entrée et **Ctrl + X** :

```
Types: deb
URIs: http://download.proxmox.com/debian/pve
Suites: trixie
Components: pve-no-subscription
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

Dans le Shell du nœud :

Ce support est la propriété de Dawan France.

```
cat /etc/apt/sources.list.d/proxmox.sources          # relire le fichier créé
```

Résultat attendu : les cinq lignes du dépôt, `Components: pve-no-subscription` en quatrième position. La relecture prouve que le fichier est bien enregistré.

Étape 4 — Mettre à jour le système

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
apt update   # relire la liste des paquets
```

Résultat attendu (fin de la sortie) :

```
Get:4 http://download.proxmox.com/debian/pve trixie InRelease [4,357 B]
Get:5 http://download.proxmox.com/debian/pve trixie/pve-no-subscription amd64
Packages [390 kB]
...
181 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Plus aucune ligne vers `enterprise.proxmox.com`, plus d'erreur 401. Le nombre de paquets dépend de la date de votre ISO.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
apt full-upgrade -y   # appliquer toutes les mises à jour
```

Sur le lab, la mise à jour a notamment installé `pve-manager (9.2.20)` et un nouveau noyau, `proxmox-kernel-7.0.14-19-pve-signed`. Or le nœud tournait encore sur l'ancien :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pveversion   # version de Proxmox VE et noyau en cours
```

```
pve-manager/9.2.20/49318c671b82f31e (running kernel: 7.0.2-6-pve)
```

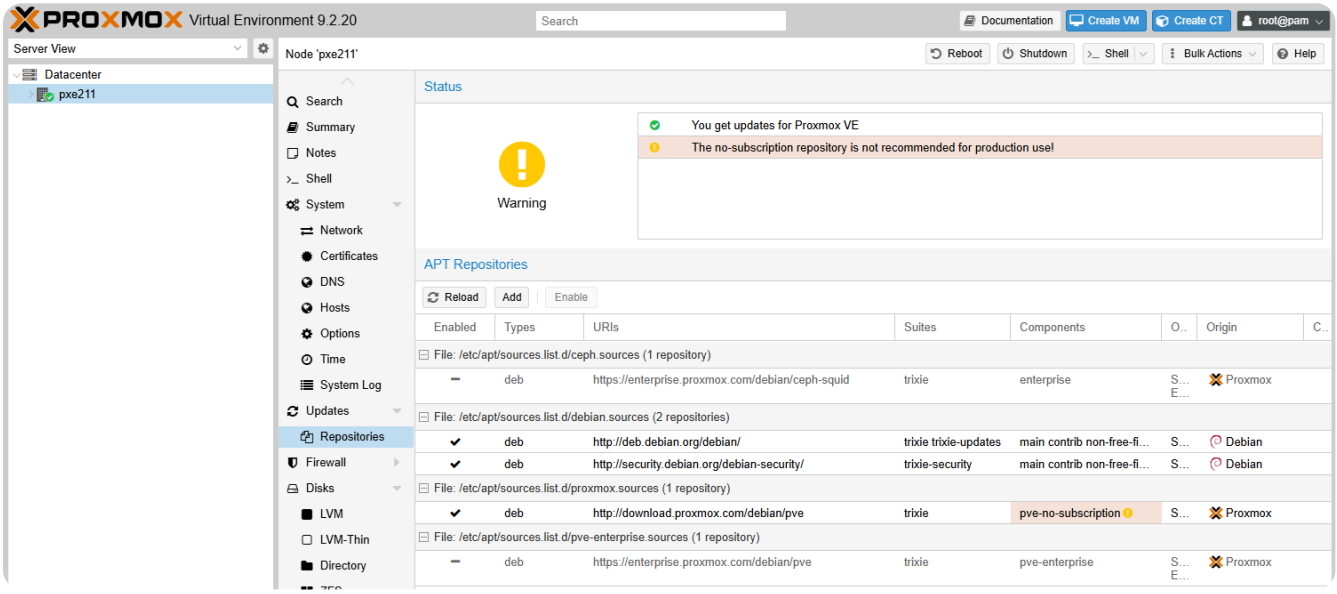
Ce support est la propriété de Dawan France.

Comment reconnaître qu'un redémarrage est nécessaire : `pveversion` affiche un `running kernel` plus ancien que le dernier noyau installé (`ls /boot/vmlinuz-*`). Redémarrez avec le bouton **Reboot** de l'interface, ou :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
reboot    # redémarrer sur le nouveau noyau
```

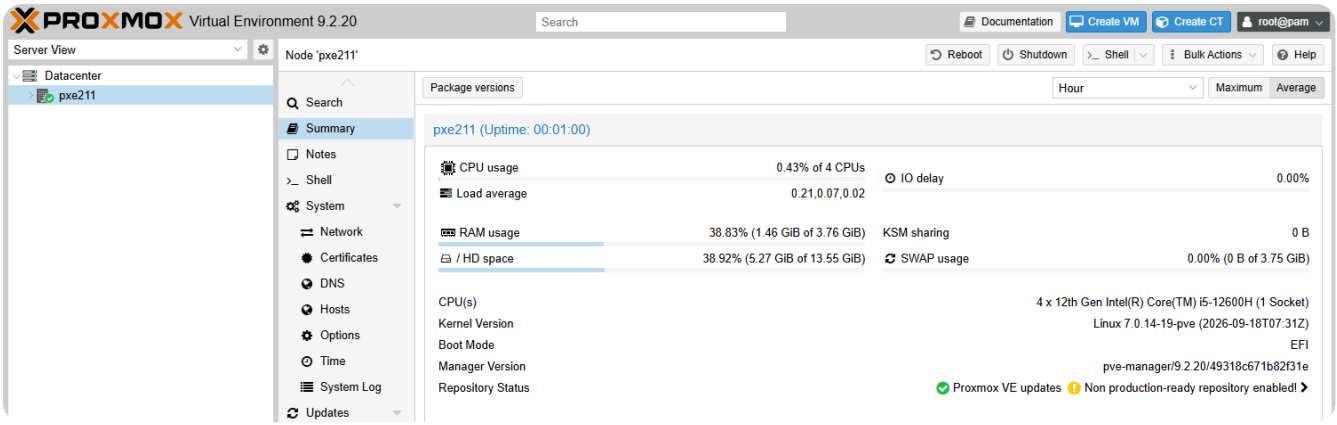
Résultat attendu après le redémarrage : `pve-manager/9.2.20/49318c671b82f31e (running kernel: 7.0.14-19-pve)`.



Les deux dépôts enterprise sont grisés. Le nouveau fichier `proxmox.sources` porte le dépôt `pve-no-subscription`, coché. Le bandeau confirme *You get updates for Proxmox VE*.

Il ajoute un avertissement jaune : *The no-subscription repository is not recommended for production use!* Ce dépôt reçoit les paquets avant le dépôt enterprise. Il convient à un lab ; en production, on prend un abonnement.

Ce support est la propriété de Dawan France.



Le **Summary** confirme le nouvel état : noyau `7.0.14-19-pve`, `pve-manager/9.2.20`, et une ligne **Repository Status** qui annonce *Proxmox VE updates*.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
grep -r "Enabled\|Components" /etc/apt/sources.list.d/   # état des dépôts : Enabled et Components
apt update 2>&1 | grep -i "401\|error"   # ne doit rien retourner
```

Résultat attendu : `Enabled: false` dans `pve-enterprise.sources` et dans `ceph.sources`, `Components: pve-no-subscription` dans `proxmox.sources`, et aucune ligne pour la seconde commande.

Atelier 4 — RBAC : utilisateurs, rôles et permissions

Contexte

Donner le mot de passe root à toute personne qui doit toucher à une VM est la faute de sécurité la plus courante sur un hyperviseur. Proxmox offre un contrôle d'accès fin — **RBAC** : on définit des rôles (ensembles de privilèges) qu'on attribue à des utilisateurs ou des groupes, sur des ressources précises.

Vous allez le mettre en œuvre sur trois profils réalistes. Un comptable qui ne doit voir qu'**une seule** VM. Deux équipes aux droits distincts.

Ce support est la propriété de Dawan France.

Un administrateur junior à qui l'on confie les VMs, **mais pas** le shell système du nœud. C'est en construisant ces trois cas qu'on comprend le modèle privilège / rôle / ACL (*Access Control List*, la règle qui lie un utilisateur, un rôle et un chemin).

Résultat à obtenir : trois profils fonctionnels et vérifiés. `comptable@pve` limité à une VM ; les groupes `dev` et `support` avec des rôles différents sur `/vms` ; `admin-jr@pve` explicitement privé de `Sys.Console`, `Sys.Syslog` et `Sys.PowerMgmt`.

La gestion se fait sous **Datacenter** → **Permissions** : utilisateurs, groupes, rôles, et la liste des permissions elle-même.



Chaque bloc correspond à un profil de l'atelier, avec le rôle et le chemin qui le définissent. Cliquez dessus pour son détail.

Arborescence attendue

```
Aucun fichier : toute la configuration vit dans la base de Proxmox
(/etc/pve/user.cfg, géré par l'outil pveum, jamais édité à la main).
État cible, vérifiable par « pveum acl list » :

comptable@pve   → rôle perso (VM.Audit/Console/PowerMgmt) sur UNE VM
groupe dev      → rôle sur /vms
groupe support  → autre rôle sur /vms
admin-jr@pve    → rôle perso SANS les privilèges Sys.*
```

Ce que vous devez faire

1. **Profil comptable :** créer un rôle personnalisé limité à `VM.Audit`, `VM.Console`, `VM.PowerMgmt`, et l'attribuer à `comptable@pve` **uniquement sur la VM ciblée** — surtout

Ce support est la propriété de Dawan France.

pas sur `/`.

2. **Profil équipes :** créer les groupes `dev` et `support`, leur attribuer des rôles différents sur `/vms`, et placer au moins un utilisateur dans chacun.
3. **Profil admin restreint :** créer un rôle qui **exclut** explicitement `Sys.Console`, `Sys.Syslog` et `Sys.PowerMgmt`, et l'attribuer à `admin-jr@pve`.
4. **Vérifier réellement** : connectez-vous avec chaque compte de test et constatez que les restrictions s'appliquent (ressources invisibles, boutons grisés). Confirmez avec `pveum acl list` qu'aucun droit superflu ne traîne.

> 💡 Le piège classique : attribuer un rôle sur `/` « pour que ça marche ». C'est l'inverse du principe de moindre privilège — on attribue toujours sur la ressource la plus étroite possible.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Pourquoi le RBAC dans Proxmox ?** En entreprise, vous ne donnez jamais le mot de passe `root` à tout le monde.
>
> Un comptable qui doit juste démarrer sa VM de travail n'a pas besoin de voir les disques, les bridges réseau ou les configurations système. Le RBAC (Role-Based Access Control) de Proxmox permet de créer des utilisateurs avec exactement les droits nécessaires — ni plus, ni moins.
>
> Le système repose sur 3 briques : un **utilisateur**, un **rôle** (liste de droits), et une **ressource** (sur quoi s'applique le rôle). L'assemblage des trois forme une **ACL** (Access Control List).

Étape 1 — Scénario comptable : accès limité à une seule VM

Une personne du service comptabilité doit pouvoir démarrer sa VM de travail, l'arrêter et ouvrir sa console — et rien d'autre.

Via l'interface : `Datacenter → Permissions → Users → Add` — Login `comptable`, Realm `Proxmox VE authentication server`, Password `proxmoxx`, Comment `Service comptabilité`.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Compte interne Proxmox (realm pve) : il n'existe pas sous Linux
pveum user add comptable@pve --password 'proxmoxx' --comment "Service
comptabilité"
# Relire la liste des comptes
pveum user list
```

Résultat attendu (nœud `pxe211`, PVE 9.2.20 ; colonnes vides retirées pour la lisibilité) :

```
┌──────────────┬──────────────────────┬──────────────────┬────────┬────────┬──
──────┬──────────────┐
│ userid       │ comment              │ email            │ enable │ expire │
```

Ce support est la propriété de Dawan France.

```
groups │ realm-type │
╞══════════════╪═══════════════════════╪═══════════════════╪═══════╪═══════╪═══
═══╪═══════════╡
│ comptable@pve │ Service comptabilité │                   │ 1     │     0 │
│ pve           │
├──────────────┼───────────────────────┼───────────────────┼───────┼───────┼──
──┼───────────┤
│ root@pam      │                       │ admin@dawan.local │ 1     │     0 │
│ pam           │
└──────────────┴───────────────────────┴───────────────────┴───────┴───────┴──
──┴───────────┘
```

Lisez la colonne **realm-type** : `comptable` est un compte du realm `pve`, qui n'existe que dans Proxmox ; `root` est un compte `pam`, c'est-à-dire un vrai compte Linux du nœud. Le courriel de `root` est celui saisi à l'installation.

L'utilisateur existe mais n'a encore aucun droit. Créez maintenant un rôle personnalisé.

Via l'interface : `Datacenter → Permissions → Roles → Create` — Name `RoleComptable`, Privileges cocher `VM.Audit`, `VM.Console`, `VM.PowerMgmt`.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# VM.Audit     = voir la VM dans la liste (sans ça, elle est invisible)
# VM.Console   = ouvrir la console noVNC dans le navigateur
# VM.PowerMgmt = démarrer, arrêter, redémarrer, mettre en pause
pveum role add RoleComptable --privs "VM.Audit,VM.Console,VM.PowerMgmt"
```

> **Sans `VM.Audit`, la VM n'apparaît pas du tout dans l'interface de cet utilisateur** — même s'il a les autres droits. C'est un point qui piège souvent les débutants.

Il faut maintenant la VM du comptable. Créez une VM vide, sans disque : elle ne sert qu'à porter le droit.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
qm create 290 --name compta --memory 128   # VM vide 290 : un nom, 128 Mo de mémoire, rien d'autre
```

Assignez le rôle uniquement sur cette VM, pas sur l'ensemble du datacenter. C'est la différence entre `/` (tout) et `/vms/290` (uniquement cette VM).

Via l'interface : `Datacenter → Permissions → Add → User Permission` — Path `/vms/290`, User `comptable@pve`, Role `RoleComptable`, Propagate décoché.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le rôle sur UN SEUL objet : la VM 290, et rien au-dessus
pveum acl modify /vms/290 --users comptable@pve --roles RoleComptable
# Relire les permissions
pveum acl list
```

Résultat attendu :

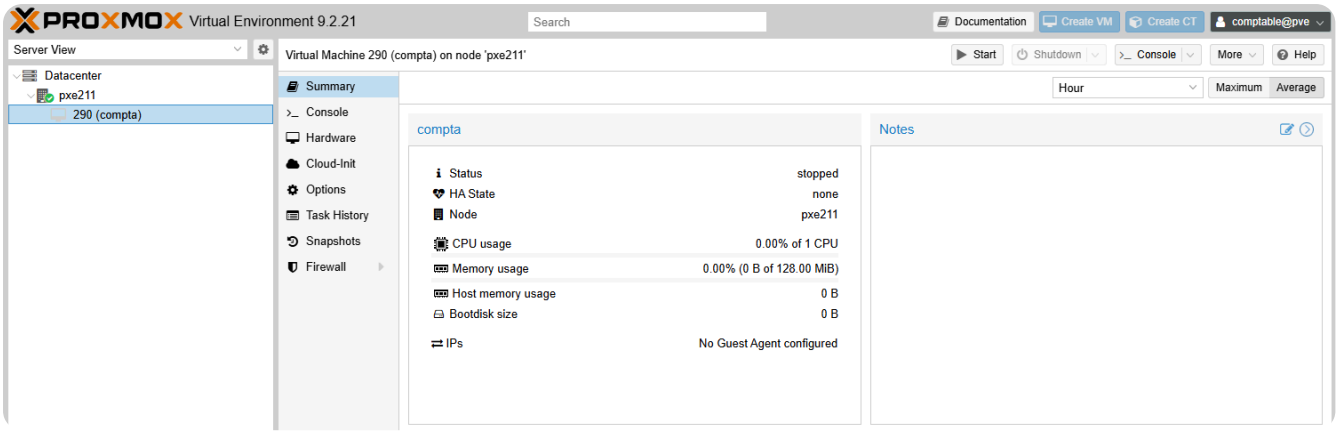
```
┌──────────┬────────────────┬──────┬────────────────┬─────────────┐
│ path     │ roleid         │ type │ ugid           │ propagate │
╞══════════╪════════════════╪══════╪════════════════╪═════════════╡
│ /vms/290 │ RoleComptable  │ user │ comptable@pve  │ 1           │
└──────────┴────────────────┴──────┴────────────────┴─────────────┘
```

Chaque ligne se lit de gauche à droite : **sur** `/vms/290`, **le rôle** `RoleComptable` est donné **à** `comptable@pve`, qui est un utilisateur (`type user` — ce serait `group` pour un groupe). `propagate` à `1` est la valeur par défaut en ligne de commande ; sur un objet sans enfants comme une VM, elle ne change rien.

Testez : ouvrez un onglet de navigation privée, connectez-vous avec `comptable` / `proxmoxx`, realm **Proxmox VE authentication server**. Vous devez voir uniquement la VM 290 dans l'arborescence — les onglets Network, Shell ou Repositories du nœud ne doivent pas être accessibles.

Résultat attendu (nœud `pxe211`) :

Dawan Ce support est la propriété de Dawan France.



L'arborescence ne contient que le nœud et la VM `290 (compta)` : ni stockage, ni autre machine. En haut, **Create VM** et **Create CT** sont grisés.

Le comptable dispose des boutons **Start**, **Shutdown** et **Console** de la VM : ce sont exactement `VM.PowerMgmt` et `VM.Console`. `VM.Audit` lui donne la vue *Summary*, en lecture seule. Ici, **Shutdown** est grisé parce que la VM est arrêtée.

Étape 2 — Scénario équipes : groupes avec rôles différents

Créez deux équipes : **dev** (peut créer des VMs de test) et **support** (consulte et redémarre uniquement).

Via l'interface : `Datacenter → Permissions → Groups → Create` — deux groupes : `dev` (Équipe développeurs), `support` (Équipe support utilisateur).

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Les deux groupes d'abord : un utilisateur ne peut rejoindre qu'un groupe existant
pveum group add dev     --comment "Équipe développeurs"
pveum group add support --comment "Équipe support utilisateur"   # le second groupe

# Un membre dans chaque groupe, rattaché dès la création
pveum user add alice@pve --password 'proxmoxx' --groups dev
pveum user add bob@pve   --password 'proxmoxx' --groups support   # le second membre, dans support
```

Ce support est la propriété de Dawan France.

```
# Vérifier les membres de chaque groupe
pveum group list
```

Résultat attendu :

```
┌─────────┬──────────────────────────────┬────────────┐
│ groupid │ comment                      │ users      │
╞═════════╪══════════════════════════════╪════════════╡
│ dev     │ Équipe développeurs          │ alice@pve  │
├─────────┼──────────────────────────────┼────────────┤
│ support │ Équipe support utilisateur   │ bob@pve    │
└─────────┴──────────────────────────────┴────────────┘
```

Via l'interface : `Datacenter → Permissions → Add → Group Permission` — Path `/vms`, Group `dev`, Role `PVEVMAdmin` ; puis Path `/vms`, Group `support`, Role `PVEVMUser`.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

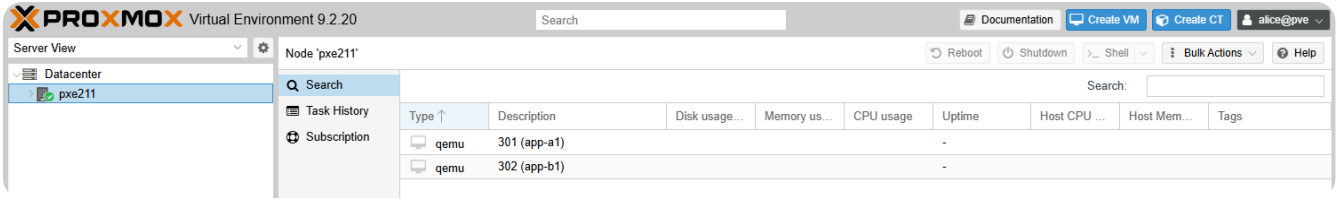
```
# PVEVMAdmin = créer, modifier, supprimer, cloner des VMs
pveum acl modify /vms --groups dev --roles PVEVMAdmin

# PVEVMUser = voir, accéder à la console, démarrer/arrêter — mais pas créer ni supprimer
pveum acl modify /vms --groups support --roles PVEVMUser
```

> Le chemin `/vms` (sans numéro) s'applique à **toutes les VMs actuelles et futures** sur ce nœud. C'est pratique pour une équipe entière, mais à utiliser avec discernement : une VM confidentielle créée plus tard serait automatiquement accessible à `alice`.

Sur le lab, en se connectant avec `alice@pve` (realm *Proxmox VE authentication server*) pendant que les deux VM de l'étape 4 existaient :

Ce support est la propriété de Dawan France.



`alice` voit les **deux** VM, `301 (app-a1)` et `302 (app-b1)`, alors qu'elles appartiennent à deux projets différents. C'est exactement l'effet de `/vms` décrit ci-dessus.

Le menu du nœud est réduit à *Search*, *Task History* et *Subscription*. Les boutons **Reboot**, **Shutdown** et **Shell** sont grisés : le rôle ne donne aucun droit sur le nœud lui-même. Comparez avec `dev-a`, à l'étape 4.

Étape 3 — Scénario admin restreint : accès sans shell système

Un administrateur junior doit gérer toute l'infrastructure Proxmox : VMs, stockage, sauvegardes. Mais **sans jamais pouvoir ouvrir un shell sur le nœud lui-même**. Il s'agit de séparer les droits de virtualisation des droits d'accès au système hôte.

Via l'interface : `Datacenter → Permissions → Users → Add` — Login `admin-jr`, Realm `Proxmox VE authentication server`, Password `proxmoxx`.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Compte de l'administrateur junior, sans aucun droit pour l'instant
pveum user add admin-jr@pve --password 'proxmoxx' --comment "Administrateur junior"
```

> **Pourquoi le CLI ici ?** La liste de privilèges est trop longue pour l'interface web. Le principe GUI existe (`Datacenter → Permissions → Roles → Create` puis cocher chaque privilège), mais `pveum role add --privs` est bien plus rapide pour une liste de 20+ droits.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Reprend les droits d'administrateur Proxmox SAUF Sys.Console (shell sur le nœud)
```

Ce support est la propriété de Dawan France.

```
# et Sys.Syslog (lecture des logs système bas niveau)
pveum role add AdminSansShell --privs \
  "VM.Allocate,VM.Audit,VM.Clone,VM.Config.CDROM,VM.Config.CPU,\
VM.Config.Cloudinit,VM.Config.Disk,VM.Config.HWType,VM.Config.Memory,\
VM.Config.Network,VM.Config.Options,VM.Console,VM.Migrate,\
VM.PowerMgmt,VM.Snapshot,VM.Snapshot.Rollback,\
Datastore.Audit,Datastore.AllocateSpace,Datastore.AllocateTemplate,\
Pool.Audit,SDN.Audit,\
Sys.Audit,Sys.Modify"
```

Les privilèges changent d'une version de Proxmox à l'autre. `VM.Monitor`, présent jusqu'aux versions 8.x, **n'existe plus sur Proxmox VE 9** : le laisser dans la liste fait échouer la création du rôle. Message obtenu sur le lab :

```
400 Parameter verification failed.
privs: invalid format - invalid privilege 'VM.Monitor'
```

Le réflexe à prendre, avant de composer un rôle personnalisé : demander à **votre** nœud ce qu'il connaît.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Tous les roles existants et les privileges que chacun contient
pveum role list

# S'inspirer de PVEAdmin, le role le plus large hors Administrator,
# puis retirer les privileges qu'on ne veut pas accorder
pveum role list | grep PVEAdmin
```

C'est plus fiable que de recopier une liste trouvée dans une documentation écrite pour une autre version.

Volontairement absent de ce rôle : `Sys.Console` (shell sur le nœud, équivalent d'un accès root Linux), `Sys.Syslog` (logs `/var/log/syslog`), `Sys.PowerMgmt` (redémarrer/éteindre le nœud physique). En production, on part souvent du rôle `PVEAdmin` (qui inclut tout) et on en retire les droits sensibles par un rôle personnalisé.

Ce support est la propriété de Dawan France.

Via l'interface : `Datacenter → Permissions → Add → User Permission` — Path `/`, User `admin-jr@pve`, Role `AdminSansShell`, Propagate coché.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Rôle donné sur / avec propagation : il couvre tout le datacenter
pveum acl modify / --users admin-jr@pve --roles AdminSansShell
```

Vérification : connexion privée avec `admin-jr@pve` / `proxmoxx`, realm *Proxmox VE authentication server*.

Résultat attendu : `admin-jr` voit tout le datacenter : le nœud, les VM et les stockages. Le menu du nœud donne accès à *Network*, *DNS*, *Disks*, *Firewall*… Mais les boutons **Reboot**, **Shutdown** et **Shell** sont grisés, et l'entrée *Shell* a disparu du menu : il manque `Sys.PowerMgmt` et `Sys.Console`.

Updates et *System Log* ont aussi disparu. Il administre la virtualisation, pas le système hôte.

> **Piège courant :** si vous assignez un rôle sur `/` et qu'une règle plus permissive existe sur un sous-chemin (ex. `/nodes/pxe2<N>1`), la règle la **plus spécifique** l'emporte. Toujours vérifier `pveum acl list` après chaque modification.

Retirer un droit

Une permission se révoque avec une sous-commande dédiée, `pveum acl delete` :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Retirer exactement le triplet chemin / utilisateur / rôle posé à l'étape 1
pveum acl delete /vms/290 --users comptable@pve --roles RoleComptable
pveum acl list   # vérifier que la ligne a disparu
```

Résultat attendu : la ligne `/vms/290` disparaît de `pveum acl list`, et l'utilisateur retrouve une interface vide à sa prochaine connexion. **Remettez la permission** (`pveum acl modify /vms/290 --users comptable@pve --roles RoleComptable`) : la validation finale en a besoin.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Il n'existe pas d'option `--delete` sur `pveum acl modify`. Sur le lab :

```
Unknown option: delete
400 unable to parse option
```

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Les comptes, les groupes, les deux rôles personnalisés, puis toutes les ACL
pveum user list
pveum group list   # les groupes
pveum role list   # les deux rôles personnalisés ; à lire : RoleComptable, AdminSansShell
pveum acl list   # toutes les ACL
```

Résultat attendu (les lignes qui comptent) : `comptable@pve`, `alice@pve`, `bob@pve` et `admin-jr@pve` dans la liste des comptes. Les groupes `dev` et `support` avec chacun leur membre, et les deux rôles avec leurs privilèges.

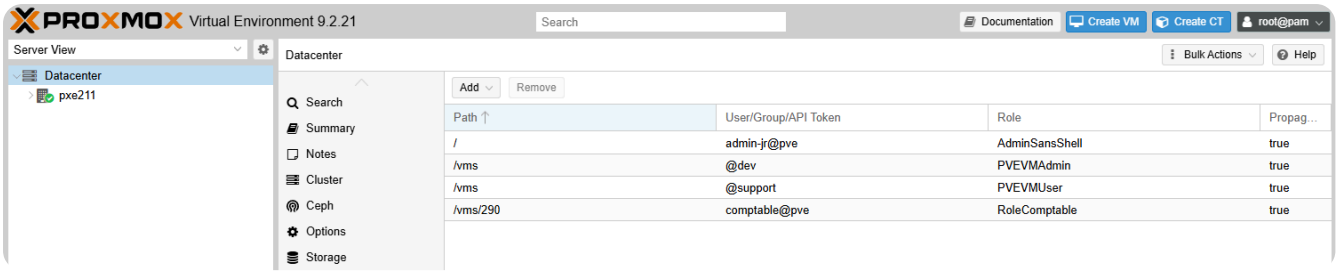
Dans les ACL, quatre lignes seulement : `/vms/290` pour le comptable, `/vms` pour chacun des deux groupes, `/` pour `admin-jr@pve`. Une cinquième ligne serait un droit oublié.

Résultat attendu de `pveum acl list` :

```
┌─────────┬─────────────────┬───────┬─────────────────┬─────────────┐
│ path    │ roleid          │ type  │ ugid            │ propagate   │
╞═════════╪═════════════════╪═══════╪═════════════════╪═════════════╡
│ /       │ AdminSansShell  │ user  │ admin-jr@pve    │ 1           │
├─────────┼─────────────────┼───────┼─────────────────┼─────────────┤
│ /vms    │ PVEVMUser       │ group │ support         │ 1           │
├─────────┼─────────────────┼───────┼─────────────────┼─────────────┤
│ /vms    │ PVEVMAdmin      │ group │ dev             │ 1           │
├─────────┼─────────────────┼───────┼─────────────────┼─────────────┤
│ /vms/290 │ RoleComptable  │ user  │ comptable@pve   │ 1           │
└─────────┴─────────────────┴───────┴─────────────────┴─────────────┘
```

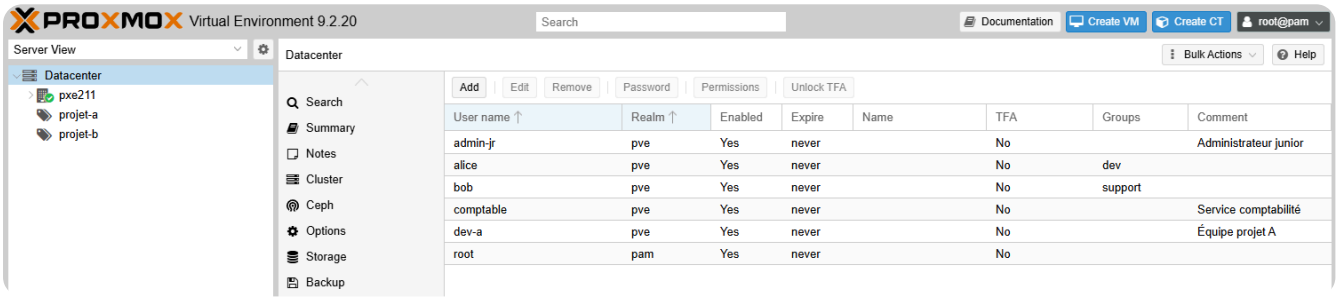
La même liste dans l'interface, `Datacenter → Permissions` :

Ce support est la propriété de Dawan France.

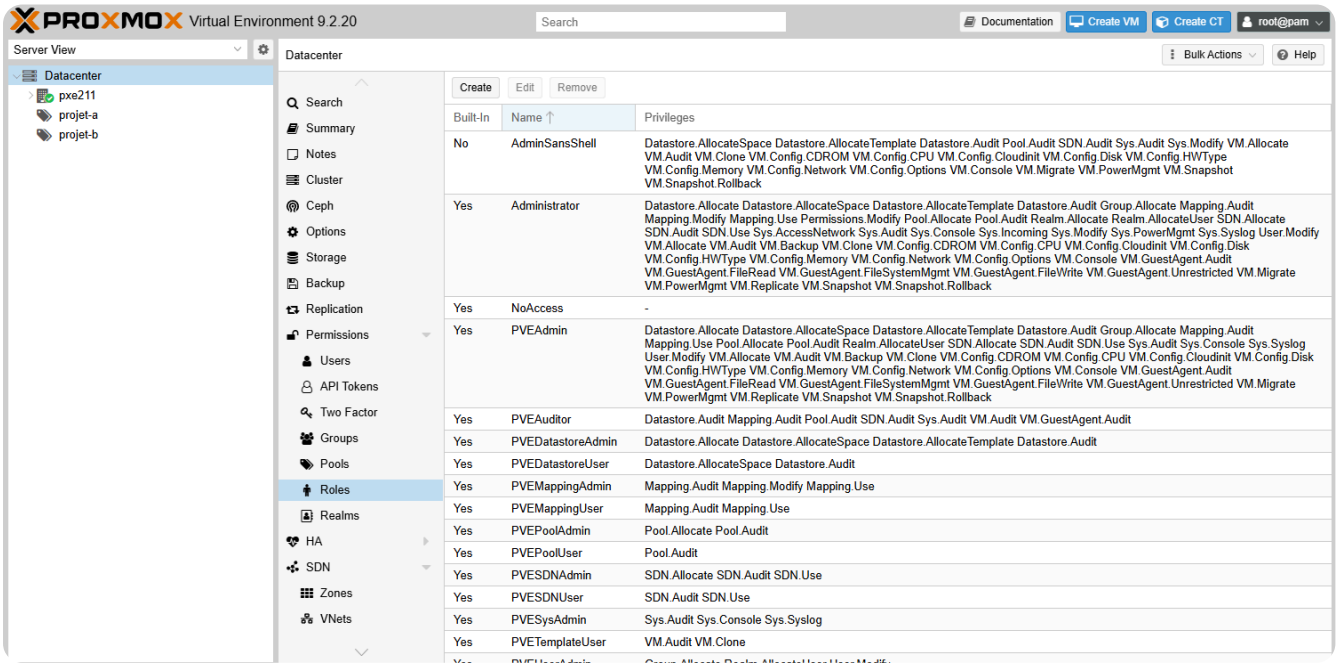


Chaque ligne est un triplet chemin / compte / rôle. Un groupe se reconnaît au `@` placé devant son nom (`@dev` , `@support`). La colonne **Propagate** à `true` signifie que le droit descend aux objets enfants du chemin.

Les comptes et les rôles se relisent de la même façon, sous `Permissions → Users` et `Permissions → Roles` :



La colonne **Realm** sépare les comptes Proxmox (`pve`) du compte système `root` (`pam`). La colonne **Groups** montre l'appartenance d' `alice` et de `bob` . **TFA** à `No` : aucun second facteur n'est encore configuré ; c'est l'objet de l'atelier 5.



Ce support est la propriété de Dawan France.

Les rôles fournis par Proxmox portent `Built-In: Yes` ; `AdminSansShell` et `RoleComptable` portent `No`. Comparez `AdminSansShell` à `PVEAdmin` : ce dernier contient `Sys.Console` et `Sys.Syslog`, que le rôle personnalisé n'a pas.

Atelier 5 — Double authentification (TOTP)

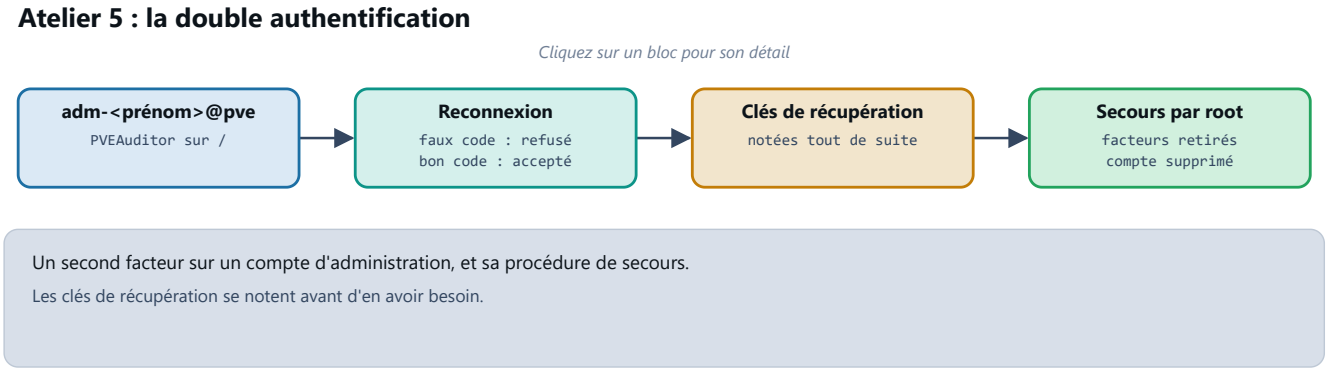
Contexte

Un mot de passe peut fuiter : réutilisé ailleurs, noté sur un post-it, intercepté. Pour un compte qui administre un hyperviseur, on ajoute un **second facteur** (cours, §8.1). C'est un code à six chiffres, calculé par une application du téléphone, qui change toutes les 30 secondes.

Vous créez un compte d'administration, vous lui ajoutez ce second facteur, et vous constatez qu'il est exigé à la connexion. Puis vous jouez le scénario de secours : le téléphone est perdu, `root` retire le second facteur.

> 📱 Il vous faut une application TOTP sur votre téléphone : Google Authenticator, Microsoft Authenticator, FreeOTP ou équivalent.

Résultat à obtenir : un compte `adm-<prénom>@pve` qui ne peut plus se connecter sans son code, et des clés de récupération générées. Puis le second facteur retiré par `root`, et le compte supprimé.



Le schéma suit le compte, de sa création jusqu'à la procédure de secours. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

Ce support est la propriété de Dawan France.

```
Aucun fichier à créer. Les facteurs sont enregistrés par Proxmox dans
/etc/pve/priv/tfa.cfg (jamais édité à la main).
État vérifiable par « pveum user tfa list adm-<prénom>@pve » :

adm-<prénom>@pve
├── totp       ← l'application du téléphone
└── recovery   ← dix clés de récupération, à usage unique
```

Ce que vous devez faire

1. **Créer le compte** `adm-<prénom>@pve` (realm Proxmox VE), avec le rôle `PVEAuditor` sur `/` : il voit tout, ne modifie rien.
2. **Vous connecter avec ce compte** dans une fenêtre de navigation privée, puis **lui ajouter un TOTP** depuis son propre menu.
3. **Vous déconnecter et vous reconnecter** : d'abord avec un faux code, puis avec le bon. Que demande l'interface ?
4. **Générer des clés de récupération.** Pourquoi faut-il les noter tout de suite ?
5. **Scénario de secours** : en `root`, constater les facteurs du compte en ligne de commande, puis les retirer. Le compte se connecte-t-il de nouveau avec son seul mot de passe ?
6. **Supprimer le compte.**

> 💡 Gardez votre session `root` ouverte dans l'onglet normal pendant tout l'atelier : c'est elle qui sert de secours si le compte de test se bloque.

Dawan
Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe211` (Proxmox VE 9.2.21), avec un compte `adm-totp@pve`. Chez vous, le compte s'appelle `adm-<prénom>@pve`.

Étape 1 — Le compte d'administration

Via l'interface, en `root` : `Datacenter → Permissions → Users → Add` — User name `adm-<prénom>`, Realm `Proxmox VE authentication server`, Password `proxmoxx`. Puis `Datacenter → Permissions → Add → User Permission` — Path `/`, User `adm-<prénom>@pve`, Role `PVEAuditor`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pveum user add adm-<prénom>@pve --password 'proxmoxx' --comment "Atelier TOTP"
# le compte de l'atelier
pveum acl modify / --users adm-<prénom>@pve --roles PVEAuditor   # lecture seule sur tout le datacenter
```

Étape 2 — Ajouter le TOTP, par le compte lui-même

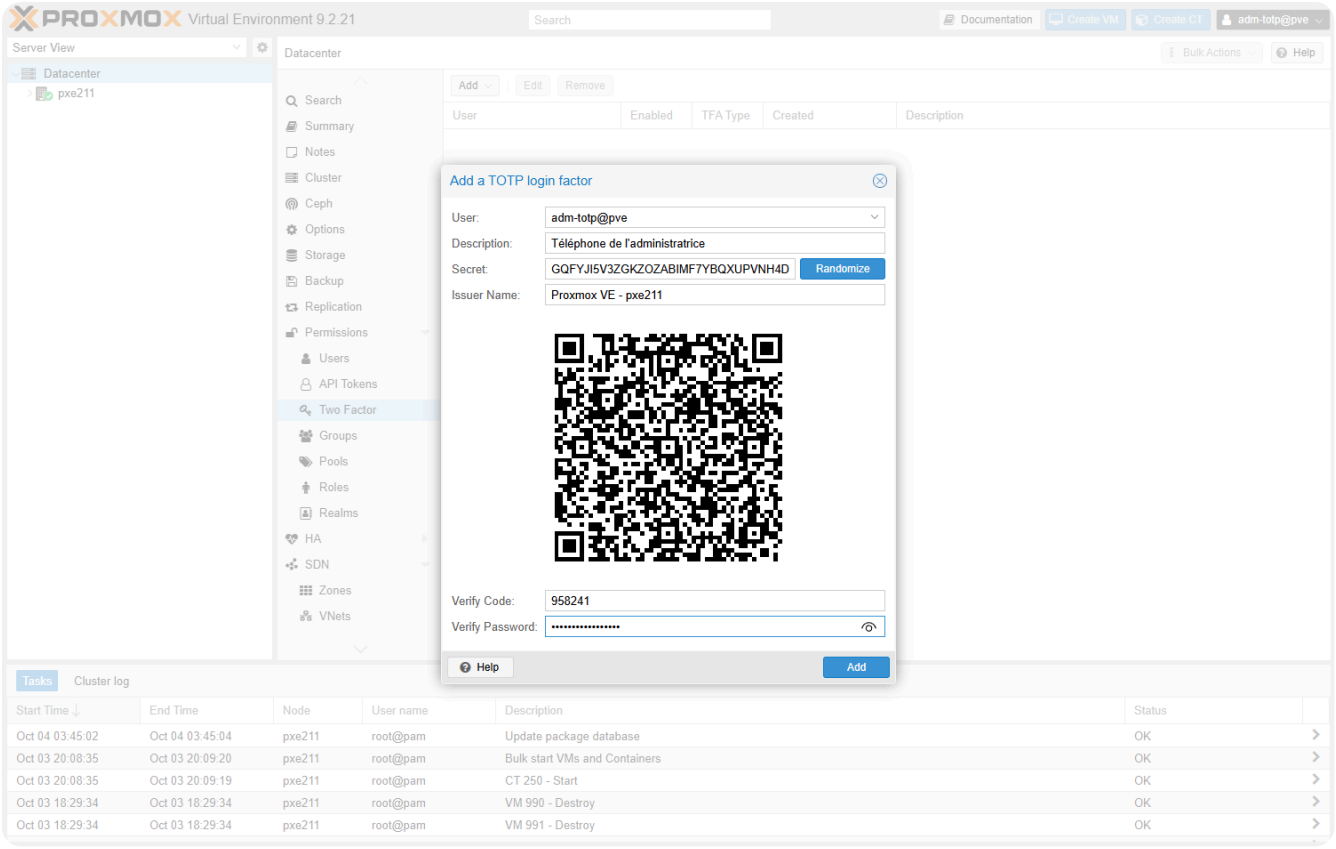
Ouvrez une **fenêtre de navigation privée**, connectez-vous avec `adm-<prénom>` / `proxmoxx`, realm **Proxmox VE authentication server**.

Menu du compte, en haut à droite (`adm-<prénom>@pve`) → **TFA**. L'interface ouvre `Datacenter → Permissions → Two Factor`. Puis **Add** → **TOTP** :

Champ	Valeur
User	`adm-<prénom>@pve` (pré-rempli)
Description	`Téléphone de <prénom>`
Secret	généré par Proxmox : ne le changez pas
QR code	à scanner avec l'application du téléphone
Verify Code	le code à six chiffres affiché par l'application
Verify Password	le mot de passe du compte, `proxmoxx`

→ **Add**.

Ce support est la propriété de Dawan France.



La fenêtre **Add a TOTP login factor** sur le lab. Le QR code contient le **Secret** et l'**Issuer Name** (`Proxmox VE - pxe211`) : c'est ce que l'application enregistre.

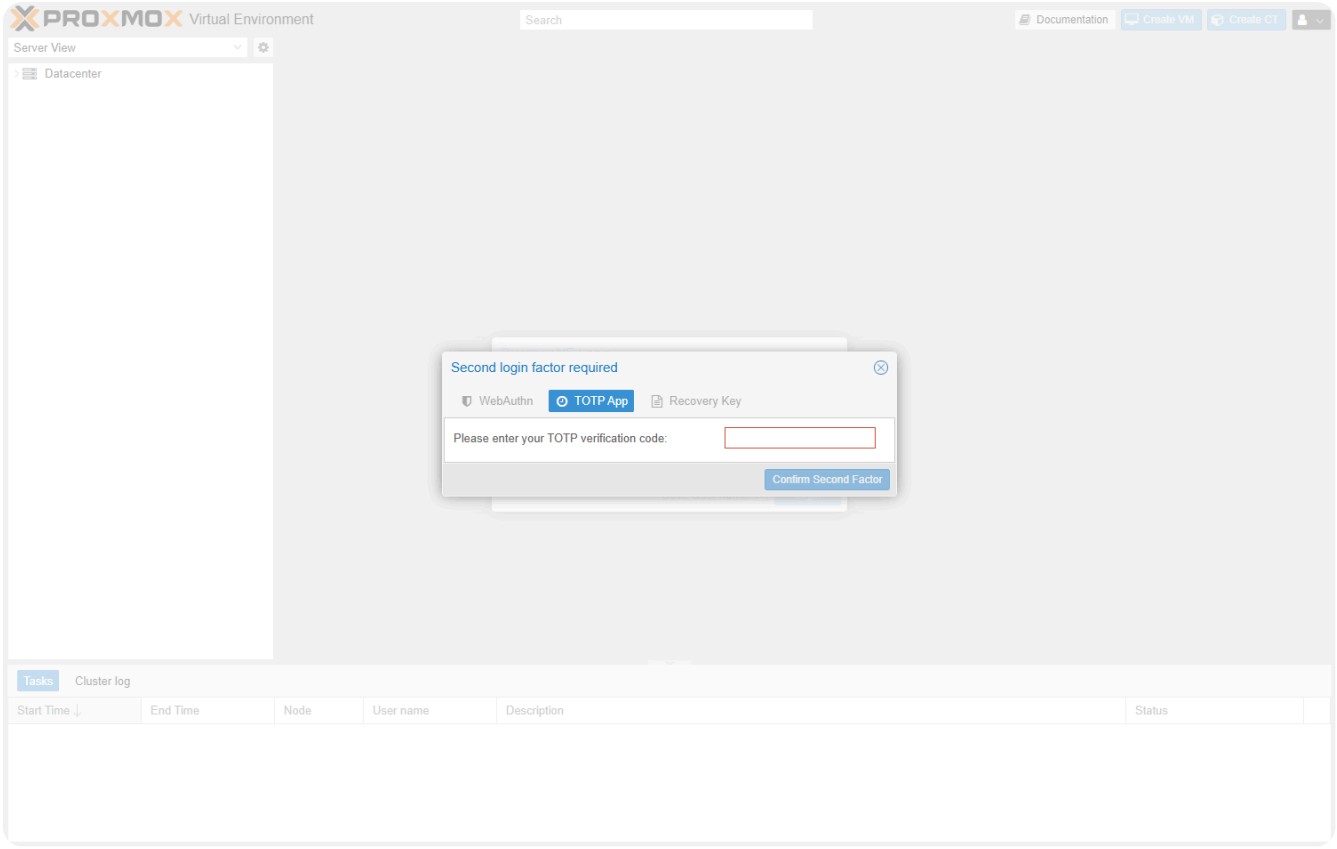
Deux contrôles avant d'accepter. **Verify Code** prouve que le téléphone calcule les mêmes codes que le serveur.

Verify Password, encadré en rouge tant qu'il est vide, prouve que c'est bien le titulaire du compte qui ajoute le facteur. Sans ces deux champs, le bouton **Add** reste grisé.

Étape 3 — Se reconnecter

Menu du compte → **Logout**, puis reconnexion avec le même compte. Après le mot de passe, une seconde fenêtre s'ouvre.

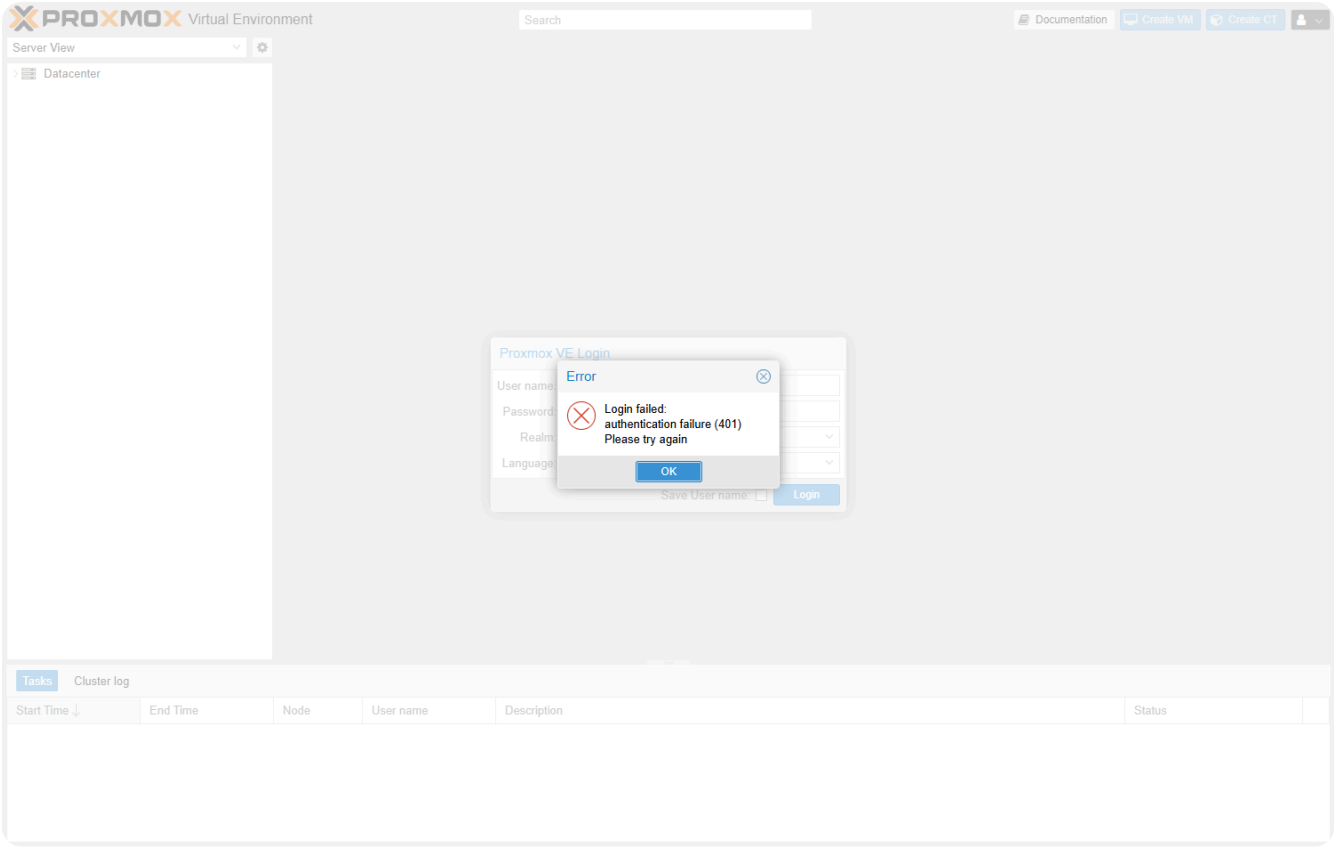
Ce support est la propriété de Dawan France.



Second login factor required : le mot de passe a été accepté, mais il ne suffit plus. L'onglet **TOTP App** attend le code du téléphone. L'onglet **Recovery Key** servira si le téléphone est perdu.

Un faux code d'abord (`000000`), puis **Confirm Second Factor** :

Ce support est la propriété de Dawan France.

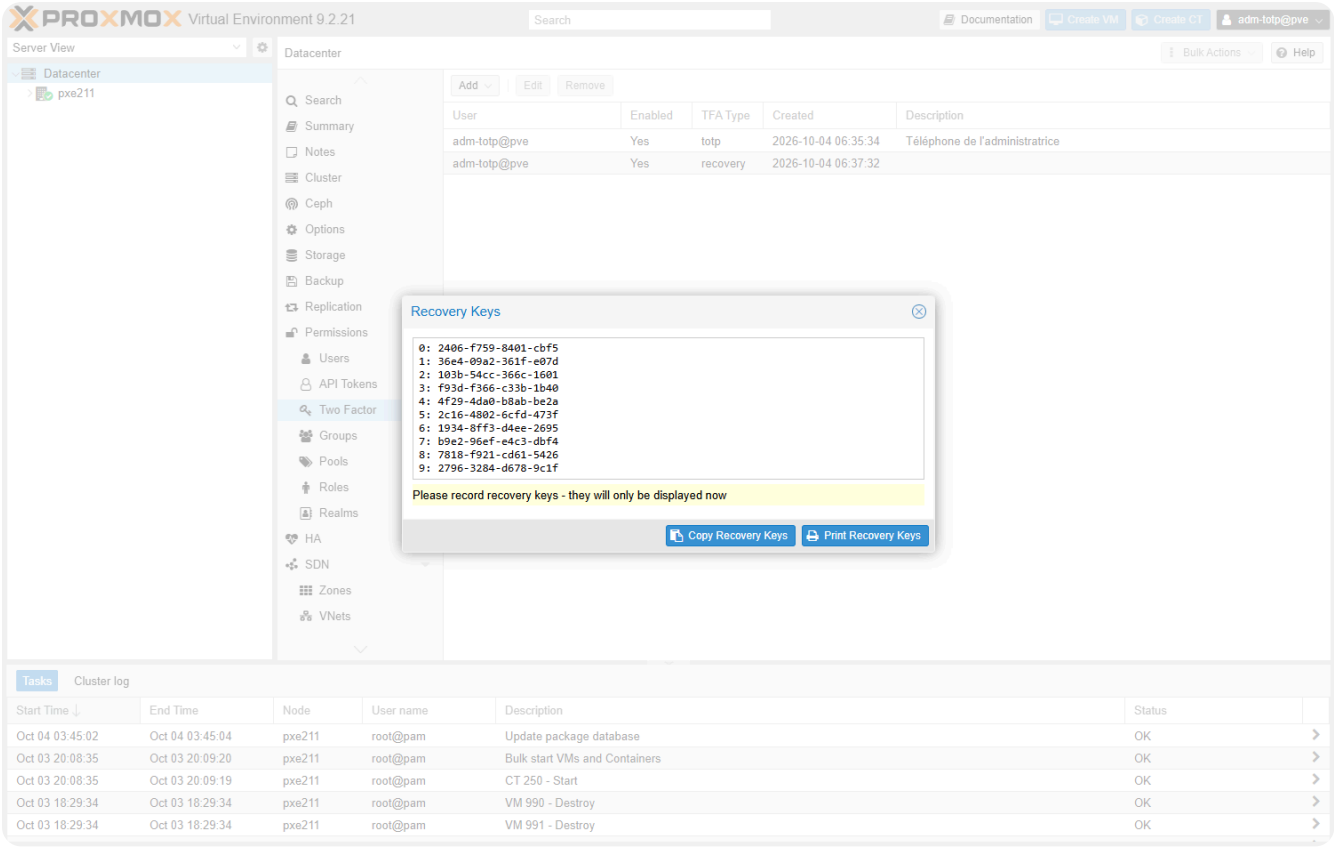


`Login failed: authentication failure (401)`. Le message ne dit pas lequel des deux facteurs est faux : c'est voulu, il ne renseigne pas un attaquant. Recommencez avec le bon code : l'interface s'ouvre normalement.

Étape 4 — Les clés de récupération

Menu du compte → **TFA** → **Add** → **Recovery Keys** → **Add**.

Ce support est la propriété de Dawan France.



Dix clés, numérotées de 0 à 9, chacune utilisable **une seule fois** à la place du code. Le bandeau jaune le dit : *they will only be displayed now*.

Elles ne seront plus jamais affichées : on les copie ou on les imprime tout de suite (**Copy Recovery Keys**, **Print Recovery Keys**). Derrière la fenêtre, la liste montre les deux facteurs du compte, de type `totp` et `recovery`.

Étape 5 — Scénario de secours : `root` retire le second facteur

Dans votre session `root`, **dans le Shell du nœud** :

```
# Les facteurs enregistrés pour le compte
pveum user tfa list adm-<prénom>@pve

# Retirer tous ses facteurs (téléphone perdu)
pveum user tfa delete adm-<prénom>@pve
pveum user tfa list adm-<prénom>@pve   # vérifier : plus aucun facteur
```

Résultat attendu, avant le retrait :

Ce support est la propriété de Dawan France.

```
recovery: recovery

totp:     bb06b50d-41e5-4ed0-ab22-84b531815761
    Téléphone de l'administratrice
```

Après `pveum user tfa delete`, la liste est vide : le compte se connecte de nouveau avec son seul mot de passe. C'est la procédure quand un administrateur a perdu son téléphone et ses clés. Elle se fait en `root`, ou avec un compte qui a les droits sur les utilisateurs.

Validation finale

Contrôle	Attendu
Connexion avec un faux code	`authentication failure (401)`
Connexion avec le bon code	interface ouverte
`pveum user tfa list` après l'étape 4	deux facteurs, `totp` et `recovery`
`pveum user tfa list` après l'étape 5	vide

Nettoyage (le compte ne sert plus) :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pveum user delete adm-<prénom>@pve   # supprimer le compte
```

La suppression du compte retire aussi sa permission sur `/`. Supprimez l'entrée `Proxmox VE - pxe2<N>1` de l'application de votre téléphone.

Atelier 6 — Conteneurs LXC : du template au service d'exploitation

> Section de cours : 11 à 13 (templates, création, administration, réseau, sauvegardes).

Contexte

Le conteneur LXC est la façon la plus légère de faire tourner un service sous Proxmox. Pas de système invité complet à démarrer, un partage du noyau de l'hôte, quelques secondes pour être

Ce support est la propriété de Dawan France.

opérationnel.

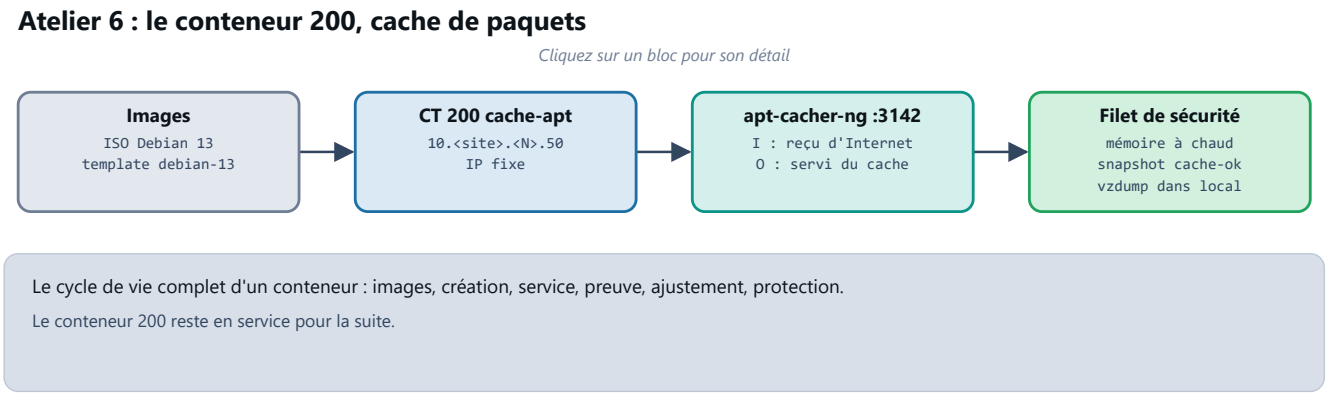
Le cours a déployé un serveur web. Ici, vous montez un service dont une équipe d'exploitation a vraiment besoin : un **cache de paquets APT** (`apt-cacher-ng`).

Chaque machine du lab qui passe par lui télécharge un paquet une seule fois depuis Internet ; les suivantes le reçoivent du cache, en local. Vous suivez le cycle de vie complet du conteneur : récupérer les images, le créer, y installer le service.

Puis **prouver** qu'il sert du cache et ajuster ses ressources **à chaud**. Enfin, éprouver le filet de sécurité : snapshot avant une bêtise, retour arrière, et sauvegarde vzdump.

Résultat à obtenir : le conteneur 200 (`cache-apt`) en marche, et `apt-cacher-ng` qui répond sur le port 3142. La preuve qu'un second téléchargement du même paquet sort du cache.

Une modification de ressources appliquée sans redémarrage. Un snapshot restauré avec succès, et une sauvegarde vzdump visible dans `local` .



Le schéma suit la vie du conteneur 200, du template au service sauvegardé. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
Stockage « local » (/var/lib/vz)
├── ISO Images
│   └── debian-13.6.0-amd64-netinst.iso            ← pour installer des VM
├── CT Templates
│   └── debian-13-standard_13.6-1_amd64.tar.zst    ← modèle du conteneur
└── Backups
    └── vzdump-lxc-200-<date>.tar.zst              ← sauvegarde de l'étape 8
```

Ce support est la propriété de Dawan France.

```
CT 200 « cache-apt » (10.<site>.<N>.50)          ← Debian 13 + apt-cacher-ng (port
3142)
└── snapshot « cache-ok »                          ← filet de l'étape 7
```

Ce que vous devez faire

1. **Récupérer les images** : une ISO Debian dans `local → ISO Images`, et un template LXC Debian dans `local → CT Templates`.
2. **Créer le conteneur 200** nommé `cache-apt`, avec une **IP fixe** (jamais DHCP sur les ressources du lab), puis le démarrer et vérifier son réseau.
3. **Installer** `apt-cacher-ng` dans le conteneur, et vérifier depuis le Shell du nœud qu'il répond sur le port 3142.
4. **Prouver le cache** : demander deux fois le même paquet Debian **à travers** le cache, et montrer, temps et journal à l'appui, que le second vient du cache.
5. **Modifier les ressources à chaud** (mémoire, CPU) et vérifier **depuis l'intérieur** du conteneur (`free -h`, `nproc`) que le changement a pris sans redémarrage.
6. **Éprouver snapshot et sauvegarde** : prendre un snapshot, puis casser volontairement le service (désinstaller `apt-cacher-ng`). Restaurer le snapshot et constater le retour du service **et de son cache**. Puis réaliser une sauvegarde vzdump vers `local`.

> 💡 La modification à chaud et le retour arrière d'un snapshot sont exactement ce qui distingue une plateforme de virtualisation d'un simple serveur. Profitez-en pour mesurer le temps que chacune fait gagner.

Dawan
Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Deux types d'images à connaître :

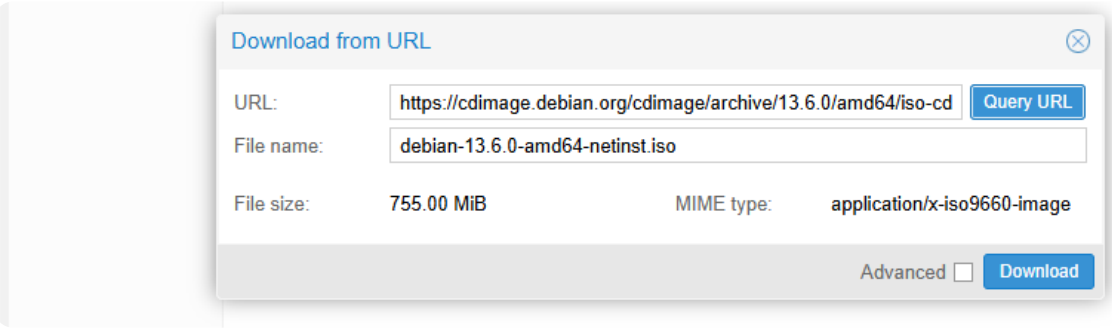
Type	Pour quoi	Où ça se stocke	Extension
ISO	Installer une VM KVM (comme un CD d'installation)	`local → ISO Images`	`.iso`
Template LXC	Créer un conteneur directement prêt	`local → CT Templates`	`.tar.zst` ou `.tar.xz`

Les ISO servent à installer des VM. Les templates servent dès ce chapitre pour les LXC.

Étape 1 — Télécharger une ISO pour les VMs

Méthode 1 — Via l'interface web : `pxe2<N>1 → local → ISO Images → Download from URL`, URL `https://cdimage.debian.org/cdimage/archive/13.6.0/amd64/iso-cd/debian-13.6.0-amd64-netinst.iso`, puis **Query URL** et **Download**.

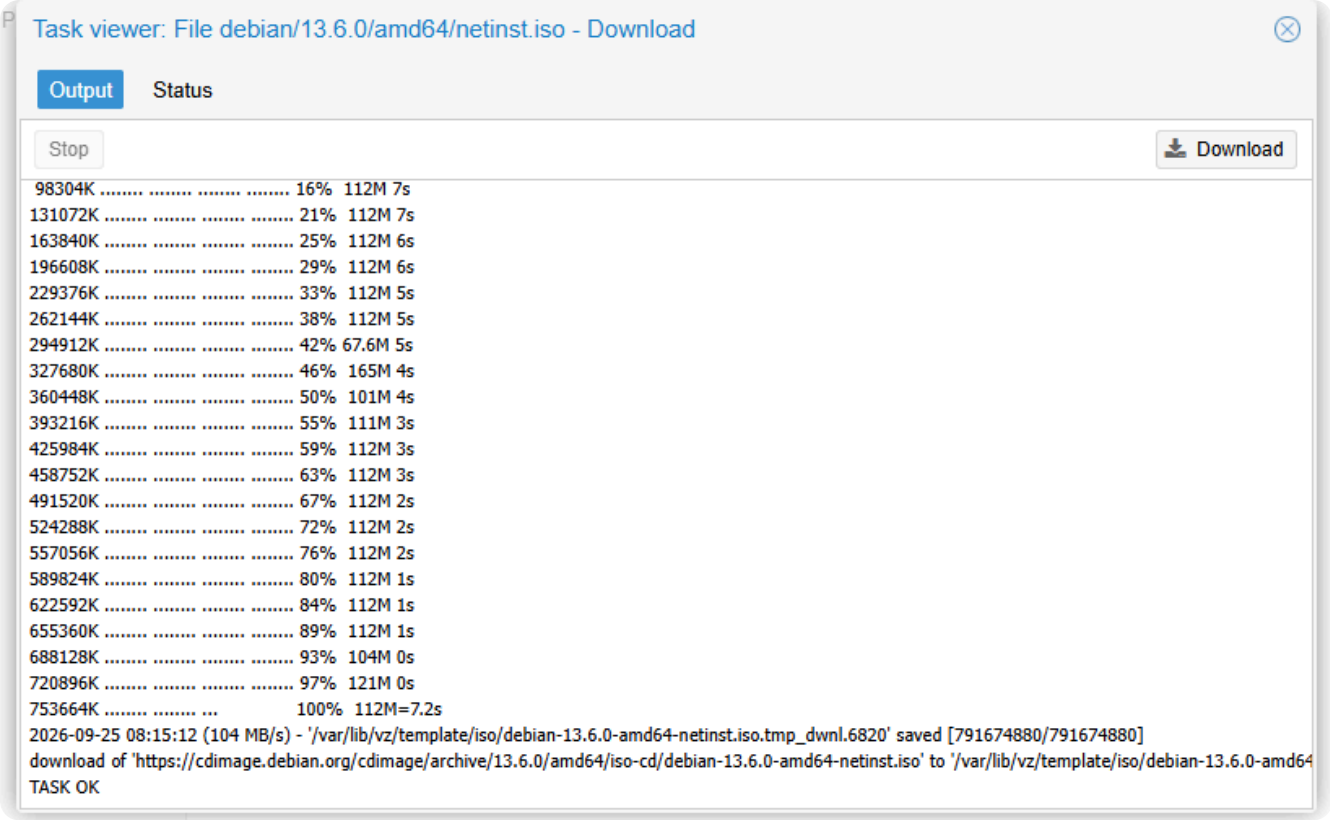
Machine d'exemple : nœud `pxe211` (Proxmox VE 9.2.21, lab `10.59.1.x`). Le CT y a reçu l'adresse `10.59.1.50` ; chez vous, c'est `10.<site>.<N>.50`.



Query URL interroge le serveur avant de télécharger : il remplit le nom du fichier, sa taille (`755.00 MiB`) et son type MIME (`application/x-iso9660-image`). C'est la vérification qu'on a la bonne URL, avant de lancer 750 Mo de téléchargement.

Résultat attendu (fin du journal de la tâche) :

Ce support est la propriété de Dawan France.



La progression par tranches, puis `saved [791674880/791674880]` et `TASK OK`. Aucune ligne de somme de contrôle : Proxmox ne vérifie l'intégrité que si on lui fournit la somme publiée par Debian. Cela se fait via la case **Advanced** de la fenêtre précédente (algorithme et valeur de la somme).

Méthode 2 — Via le shell :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
cd /var/lib/vz/template/iso/   # le dossier des ISO du stockage local
wget https://cdimage.debian.org/cdimage/archive/13.6.0/amd64/iso-cd/debian-13.6.0-amd64-netinst.iso   # télécharger l'ISO Debian 13
ls -lh /var/lib/vz/template/iso/   # vérifier la présence du fichier
```

Méthode 3 — Upload : `local → ISO Images → Upload` depuis votre poste, si pas d'accès internet depuis le lab.

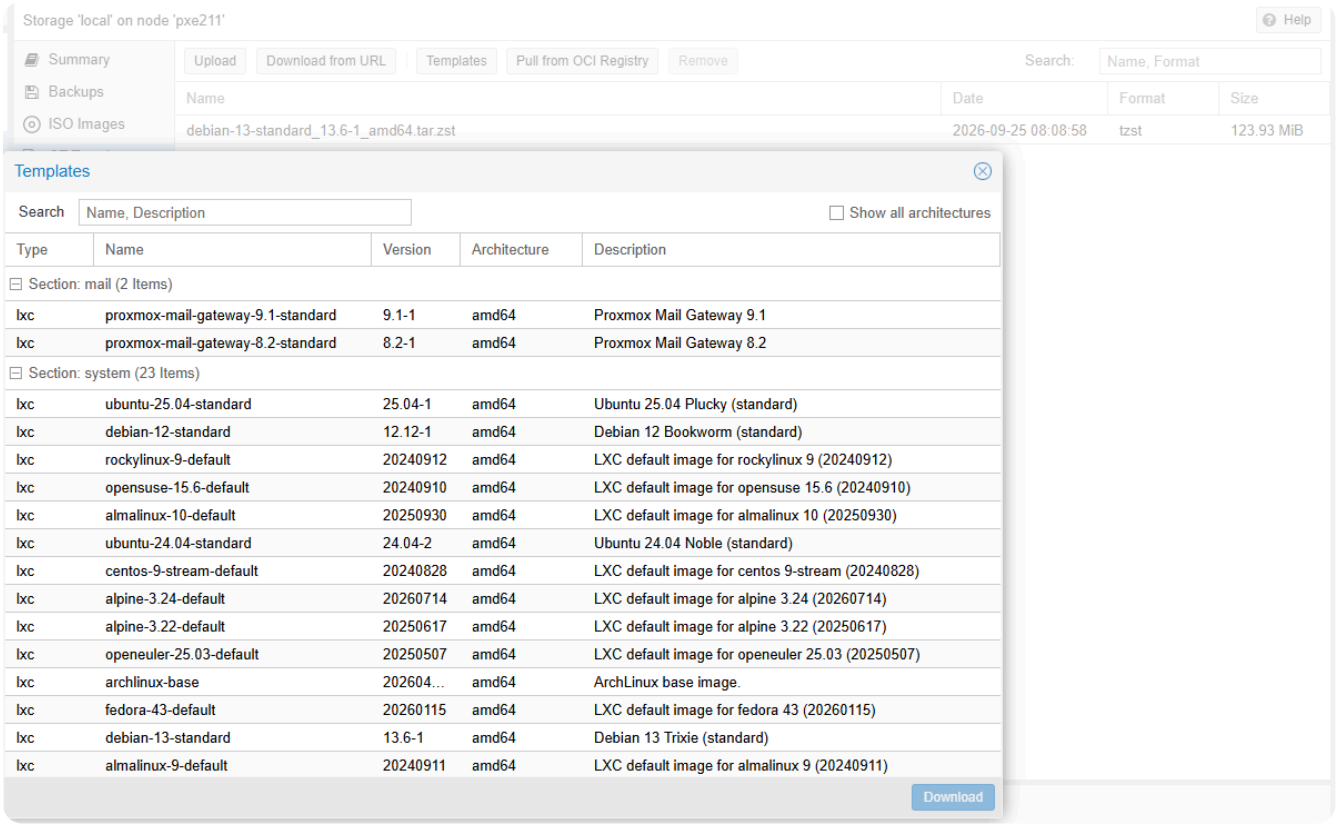
> **Attention à la taille des ISOs :** Ubuntu Desktop fait 5 Go, Windows Server 2022 fait 5-6 Go — vérifiez l'espace disponible (`pvesm status | grep local`) avant d'uploader. Pour les formations, préférez toujours les ISOs **netinst** (~650 Mo).

Ce support est la propriété de Dawan France.

Étape 2 — Télécharger un template LXC

Un template LXC est un OS préinstallé compressé, prêt à déployer en quelques secondes — l'équivalent d'une image Docker pour des conteneurs système complets.

Méthode 1 — Via l'interface : `local → CT Templates → Templates`, recherchez `debian-13`, sélectionnez `debian-13-standard_...` → **Download**.



En fond, `local → CT Templates` contient déjà `debian-13-standard_13.6-1_amd64.tar.zst` (123.93 MiB). Au premier plan, le catalogue **Templates** : les modèles *standard* maintenus par Proxmox (Debian, Ubuntu) et les images *default* d'autres distributions (Rocky, Alma, Alpine, Fedora…), plus deux modèles *Proxmox Mail Gateway*.

La barre du haut montre aussi un bouton lié à l'OCI (*Open Container Initiative*, format standard des images de conteneurs) : **Pull from OCI Registry**. Il récupère une image de conteneur applicatif.

Méthode 2 — Via le shell :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Rafraîchir l'index, puis repérer le nom exact du modèle Debian 13
pveam update
pveam available --section system | grep debian   # repérer le modèle Debian
# Télécharger ce modèle sur le stockage « local », puis vérifier sa présence
pveam download local debian-13-standard_13.6-1_amd64.tar.zst
pveam list local   # vérifier sa présence
```

Résultat attendu :

```
NAME                                                 SIZE
local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst 123.93MB
```

Étape 3 — Créer un conteneur LXC depuis l'interface

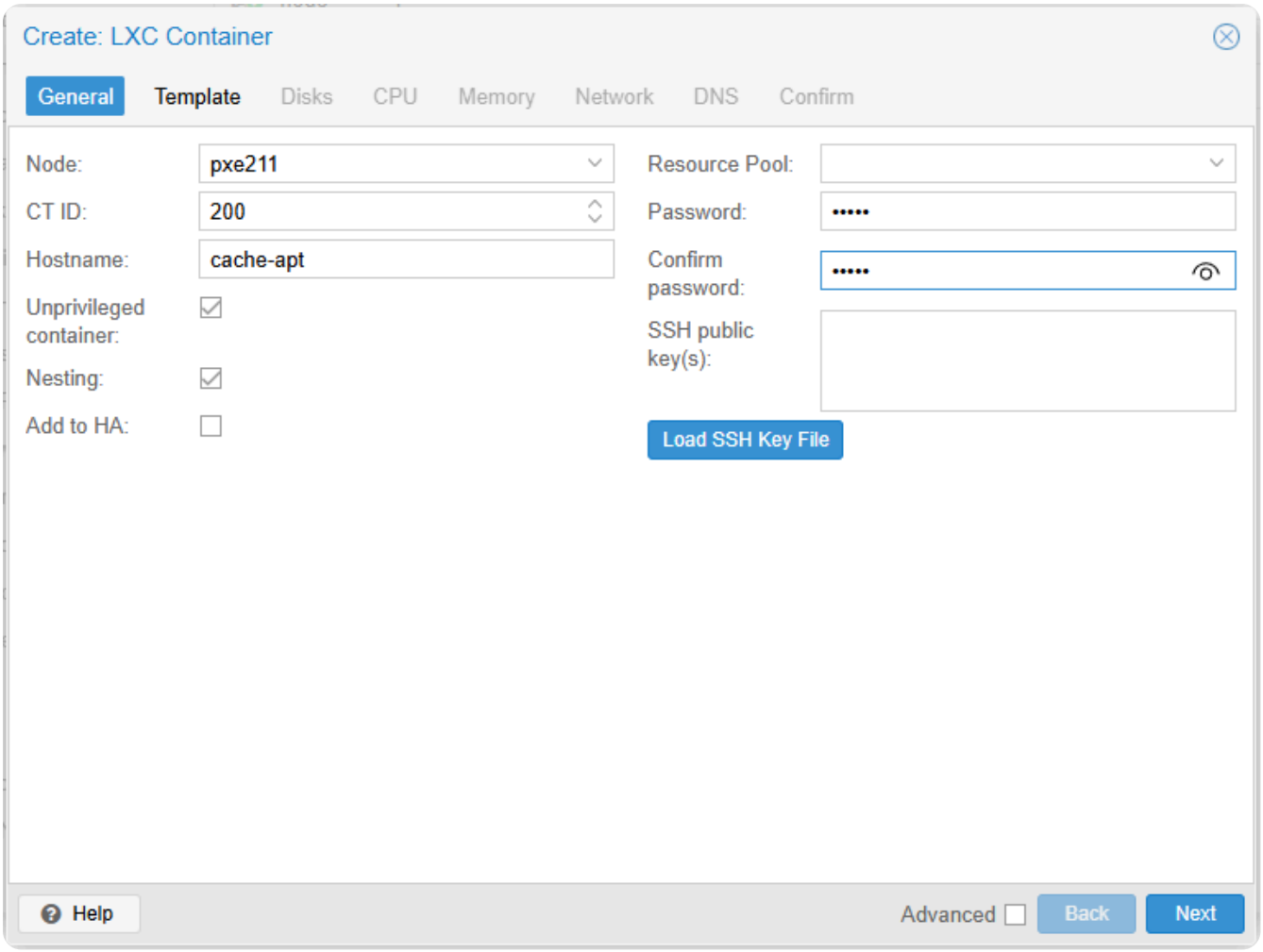
Create CT en haut à droite :

Onglet	Champ	Valeur
General	CT ID	`200`
General	Hostname	`cache-apt`
General	Password	`dawan`
Template	—	`local` → `debian-13-standard_13.6-1_amd64.tar.zst`
Disks	Storage / Size	`local-lvm` / `8 Go`
CPU	Cores	`1`
Memory	Memory / Swap	`512 Mo` / `512 Mo`
Network	Bridge / IPv4 / Gateway	`vmbr0` / **Static** `10.<site>.<N>.50/24` / `10.<site>.<N>.1`

L'adresse est **fixe**, comme toute ressource créée sur le lab : `10.<site>.<N>.50`. `<site>` vaut 44 (Nantes), 75 (Paris) ou 67 (Strasbourg) ; `<N>` est votre numéro de stagiaire (celui de `pxe2<N>1`).

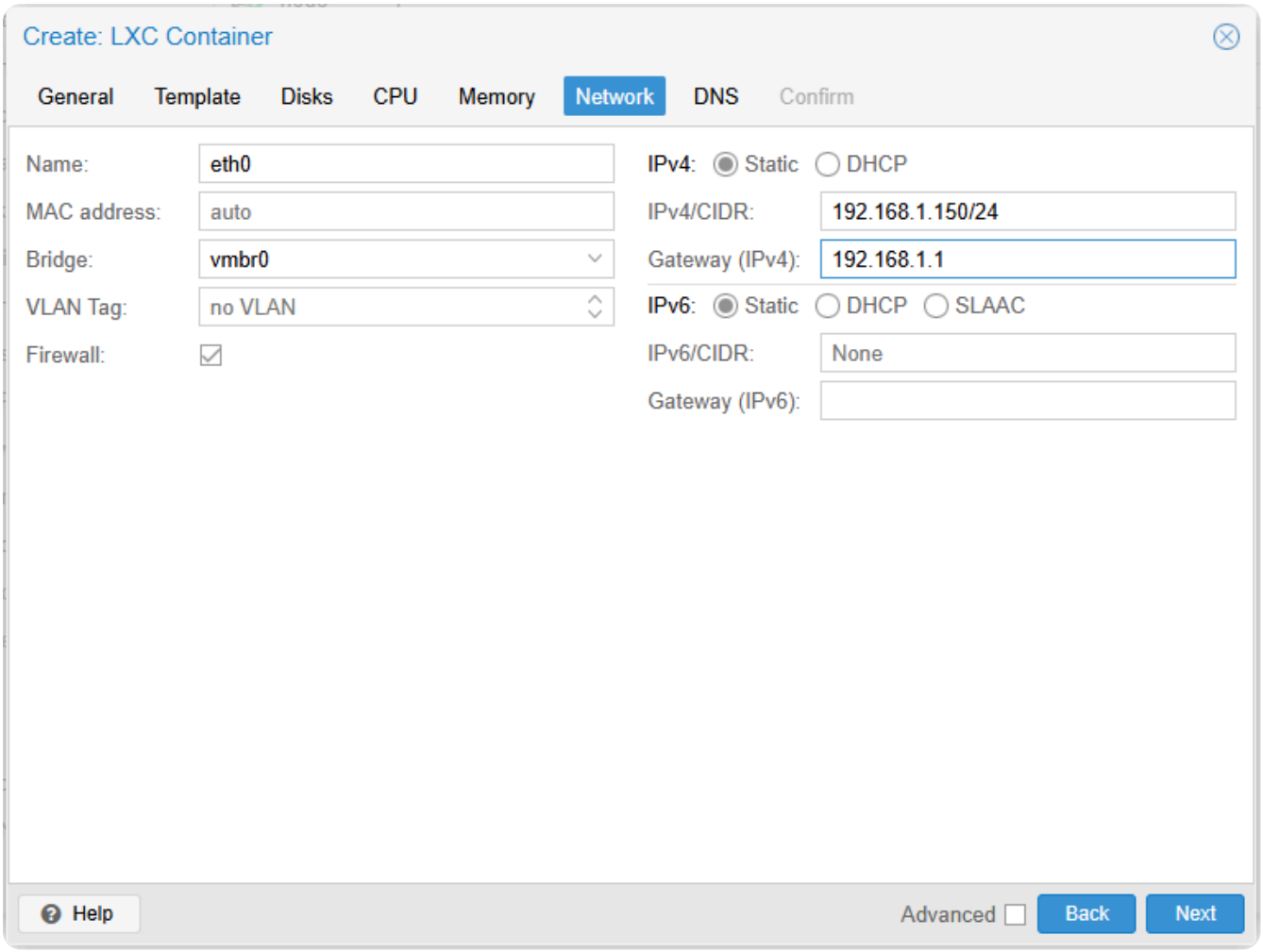
La passerelle est en `.1` (introduction générale, règle d'adressage). Un service que d'autres machines doivent joindre n'a de toute façon pas d'autre choix : son adresse ne doit jamais changer.

Ce support est la propriété de Dawan France.



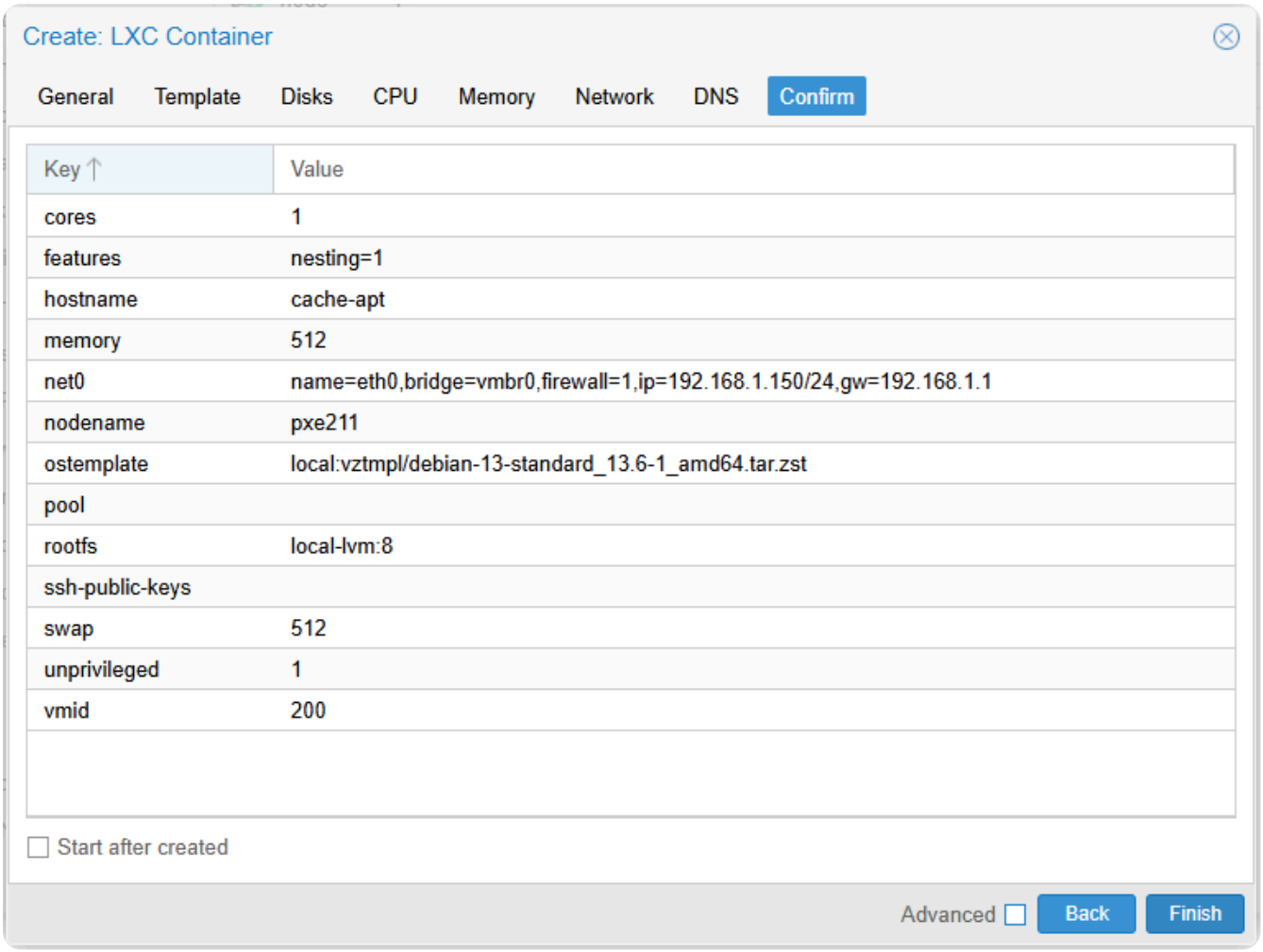
CT ID et **Hostname** identifient le conteneur. Deux cases sont cochées par défaut : **Unprivileged container** (le root du conteneur n'est pas le root de l'hôte, cours §10) et **Nesting** (utile à systemd dans les distributions récentes). Le mot de passe est celui du `root` **du conteneur**, pas celui du nœud.

Ce support est la propriété de Dawan France.



IPv4 : Static, avec l'adresse en notation CIDR (`/24`) et la passerelle. La carte s'appelle `eth0` dans le conteneur et se branche sur le bridge `vmbr0`. La case **Firewall** est cochée par défaut : elle ne filtre rien tant que le pare-feu n'est pas activé.

Ce support est la propriété de Dawan France.

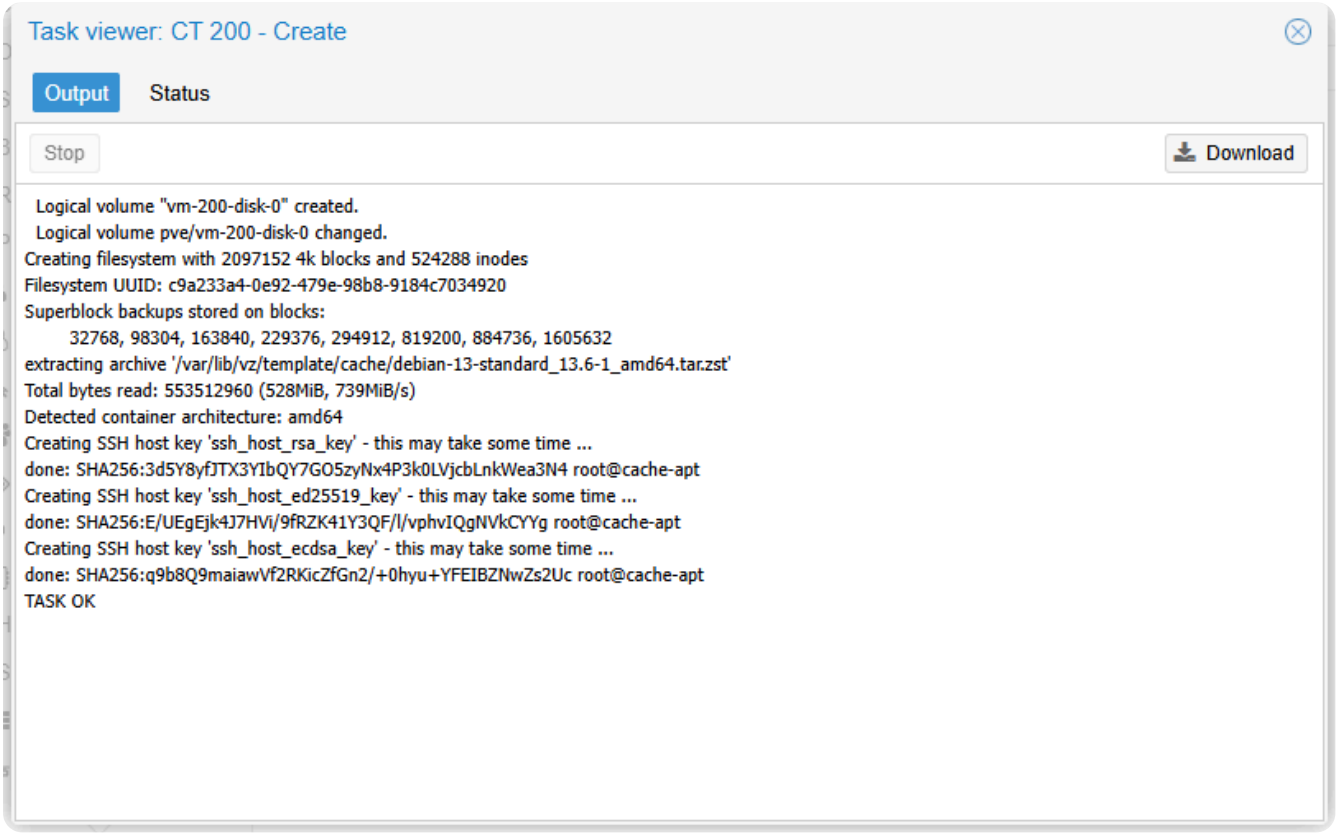


Le résumé liste les paramètres réellement envoyés : `cores 1`, `hostname cache-apt`, `memory 512`, `swap 512`, `rootfs local-lvm:8`, `net0 …ip=…/24,gw=…`, et les valeurs par défaut `unprivileged 1` et `features nesting=1`. La case **Start after created** est **décochée par défaut** : le conteneur sera créé, mais pas démarré.

Cliquez **Finish**.

Résultat attendu :

Ce support est la propriété de Dawan France.



Proxmox crée le volume `vm-200-disk-0` sur `local-lvm`, le formate, y extrait le modèle, puis génère les clés SSH propres au conteneur (`root@cache-apt`). `TASK OK` : deux secondes sur le lab.

En ligne de commande, le même conteneur en une commande (à la place de l'assistant, pas en plus) :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Chaque option reprend un champ de l'assistant
pct create 200 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst \
  --hostname cache-apt --password dawan \
  --rootfs local-lvm:8 --cores 1 --memory 512 --swap 512 \
  --net0 name=eth0,bridge=vmbr0,ip=10.<site>.<N>.50/24,gw=10.<site>.
<N>.1,firewall=1 \
  --unprivileged 1 --features nesting=1
```

La commande affiche le même journal que la tâche de l'assistant, et la configuration obtenue est identique (`pct config 200`, étape 4). Comme dans l'assistant, le conteneur est créé arrêté ; `--start 1` le démarrerait aussitôt.

Ce support est la propriété de Dawan France.

Étape 4 — Démarrer et vérifier le conteneur

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Démarrer le conteneur, puis contrôler son état et sa configuration
pct start 200
pct status 200   # son état
pct config 200   # sa configuration

# Ouvrir un shell root dans le conteneur, sans mot de passe ni réseau
pct enter 200
```

Résultat attendu (`pct config 200`) :

```
arch: amd64
cores: 1
features: nesting=1
hostname: cache-apt
memory: 512
net0:
name=eth0,bridge=vmbr0,firewall=1,gw=10.59.1.1,hwaddr=BC:24:11:FE:44:FB,ip=10.59.
1.50/24,type=veth
ostype: debian
rootfs: local-lvm:vm-200-disk-0,size=8G
swap: 512
unprivileged: 1
```

`status: running` avant cela ; avec `pct enter 200`, le prompt devient `root@cache-apt:~#`.

Dans le conteneur (le prompt `root@cache-apt:~#` ouvert par `pct enter 200`) :

```
# L'adresse fixe est-elle bien portée par eth0 ?
ip -4 addr show eth0
# La route par défaut doit passer par la passerelle déclarée
ip route
# Le serveur qui l'héberge, sur le même réseau
ping -c 3 -q <ip-de-votre-noeud>
# Quelle distribution tourne dans le conteneur ?
```

Ce support est la propriété de Dawan France.

```
grep PRETTY /etc/os-release
exit   # revenir au Shell du nœud
```

Résultat attendu :

```
2: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000 link-netnsid 0
    inet 10.59.1.50/24 scope global eth0
       valid_lft forever preferred_lft forever
default via 10.59.1.1 dev eth0 proto kernel onlink
10.59.1.0/24 dev eth0 proto kernel scope link src 10.59.1.50
3 packets transmitted, 3 received, 0% packet loss, time 2075ms
rtt min/avg/max/mdev = 0.039/0.061/0.074/0.015 ms
PRETTY_NAME="Debian GNU/Linux 13 (trixie)"
```

`valid_lft forever` signe une adresse fixe (une adresse DHCP aurait une durée de bail).

Étape 5 — Installer le cache de paquets

`apt-cacher-ng` pose une question à l'installation. Lancé par `pct exec`, sans clavier pour y répondre, il resterait bloqué : `DEBIAN_FRONTEND=noninteractive` lui fait prendre la réponse par défaut.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis le shell Proxmox (pas dans le conteneur) : « -- » sépare pct de la
commande
pct exec 200 -- apt update
# Installation sans question ; le service démarre et s'active tout seul
pct exec 200 -- env DEBIAN_FRONTEND=noninteractive apt install -y apt-cacher-ng
# Actif au démarrage du conteneur ? Actif maintenant ?
pct exec 200 -- systemctl is-enabled apt-cacher-ng
pct exec 200 -- systemctl is-active apt-cacher-ng   # le service tourne-t-il ?

# Interroger le service depuis le nœud, sur son port 3142
curl -s -o /dev/null -w "rapport -> HTTP %{http_code}\n" http://10.<site>.
<N>.50:3142/acng-report.html
```

Résultat attendu (fin de l'installation, puis les trois vérifications) :

Dawan
Ce support est la propriété de Dawan France.

```
Created symlink '/etc/systemd/system/multi-user.target.wants/apt-cacher-
ng.service' → '/usr/lib/systemd/system/apt-cacher-ng.service'.
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for libc-bin (2.41-12+deb13u3) ...
enabled
active
rapport -> HTTP 200
```

L'installation affiche aussi plusieurs lignes `locale: Cannot set LC_CTYPE to default locale` : la langue du nœud n'est pas installée dans le conteneur. C'est sans conséquence.

La ligne `Created symlink` est la preuve que le service démarrera avec le conteneur. `acng-report.html` est la page de rapport du cache ; interrogée à la racine (`/`), le service répond `406 Usage Information`, ce qui est normal : il attend des demandes de paquets, pas une visite.

Étape 5 bis — Prouver que le cache sert

Un service qui répond ne prouve pas qu'il fait son travail. On demande deux fois le même paquet Debian **à travers** le cache (`curl -x`, qui passe par un proxy), sans rien modifier sur le nœud.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Un vrai paquet du dépôt Debian 13
# Deux demandes identiques, à travers le cache : taille et durée de chacune
curl -x http://10.<site>.<N>.50:3142 -s -o /dev/null -w "essai 1 : HTTP %
{http_code}, %{size_download} octets, %{time_total} s\n"
http://deb.debian.org/debian/pool/main/h/hello/hello_2.10-5_amd64.deb   # la
première
curl -x http://10.<site>.<N>.50:3142 -s -o /dev/null -w "essai 2 : HTTP %
{http_code}, %{size_download} octets, %{time_total} s\n"
http://deb.debian.org/debian/pool/main/h/hello/hello_2.10-5_amd64.deb   # la
même, de nouveau
# Le journal du cache, dans le conteneur
pct exec 200 -- tail -3 /var/log/apt-cacher-ng/apt-cacher.log
```

Résultat attendu :

Ce support est la propriété de Dawan France.

```
essai 1 : HTTP 200, 53228 octets, 0.051535 s
essai 2 : HTTP 200, 53228 octets, 0.000364 s
1791380405|I|54762|10.59.1.11|debrep/pool/main/h/hello/hello_2.10-5_amd64.deb
1791380405|O|53507|10.59.1.11|debrep/pool/main/h/hello/hello_2.10-5_amd64.deb
1791380405|O|53507|10.59.1.11|debrep/pool/main/h/hello/hello_2.10-5_amd64.deb
```

Sur le lab (lab `10.59.1.x`). Le premier essai a pris 52 ms, le second 0,4 ms : plus de cent fois moins. Le journal dit pourquoi.

La ligne `I` (*in*) est le fichier **reçu d'Internet**, une seule fois. Les deux lignes `O` (*out*) sont les deux envois vers le nœud (`10.59.1.11` sur le lab, votre nœud chez vous). Le paquet est désormais rangé dans le conteneur, sous `/var/cache/apt-cacher-ng/debrep/pool/main/h/hello/`.

Pour qu'une machine fasse passer tout son `apt` par ce cache, une ligne suffit dans `/etc/apt/apt.conf.d/` : `Acquire::http::Proxy "http://10.<site>.<N>.50:3142";`. Ne la posez **pas** sur votre nœud : le jour où le conteneur serait arrêté, les mises à jour du nœud échoueraient.

Étape 6 — Modifier les ressources à chaud

Les ressources d'un conteneur LXC se modifient sans arrêt — les cgroups appliquent les limites immédiatement.

Via l'interface : CT 200 → Resources → double-clic Memory (`768` Mo / Swap `512` Mo) → OK, puis Processors → `2` cœurs → OK.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Deux ressources modifiées d'un coup, conteneur en marche
pct set 200 --memory 768 --cores 2

# Vérifier depuis l'intérieur : lxcfs fait voir au conteneur ses propres limites
pct exec 200 -- free -h
pct exec 200 -- nproc   # nombre de cœurs vus par le conteneur
```

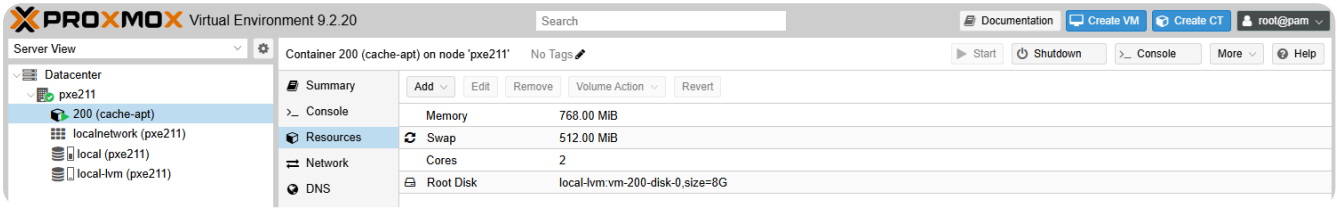
Résultat attendu : `free -h` passe de `512Mi` à `768Mi` sur la ligne `Mem:` **immédiatement** — la limite du conteneur, et non la mémoire de l'hôte. `nproc`, lui, répondait encore `1` juste

Ce support est la propriété de Dawan France.

après la commande, puis `2` une vingtaine de secondes plus tard. Aucun redémarrage n'a eu lieu : `pct status 200` n'a jamais cessé d'afficher `running`.

Pourquoi ce délai sur les cœurs ? La mémoire est appliquée tout de suite par les cgroups.

Le nombre de cœurs, lui, est traduit en une liste de processeurs (`cpuset`) par le service `pvestatd` du nœud, qui répartit les conteneurs à chaque cycle de 10 secondes. Si `nproc` affiche l'ancienne valeur, attendez un cycle et relancez-le.



Le CT `200 (cache-apt)` tourne (icône verte dans l'arborescence) avec `768.00 MiB` de mémoire et `2` cœurs : les nouvelles valeurs sont en place sans arrêt. Pour les modifier à la souris : double-clic sur la ligne.

Étape 7 — Snapshot et restauration

Créer le snapshot : CT 200 → Snapshots → Take Snapshot, Name `cache-ok`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Snapshot nommé, avec une description qui dit POURQUOI il a été pris
pct snapshot 200 cache-ok --description "apt-cacher-ng installe et fonctionnel"
pct listsnapshot 200   # vérifier que le snapshot existe
```

Résultat attendu :

```
`-> cache-ok                    2026-09-28 07:58:43     apt-cacher-ng installe et
fonctionnel
 `-> current                                            You are here!
```

Deux lignes : `cache-ok` avec sa date et sa description, puis `current`, l'état présent du conteneur, « posé » sur ce snapshot.

Simuler une erreur :

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# La « bêtise » : désinstaller le service et sa configuration
pct exec 200 -- env DEBIAN_FRONTEND=noninteractive apt purge -y apt-cacher-ng
# Le service a disparu, et le port 3142 ne répond plus
pct exec 200 -- systemctl is-active apt-cacher-ng
curl -s -m 5 -o /dev/null -w "curl -> %{http_code} (code retour %{exitcode})\n"
http://10.<site>.<N>.50:3142/acng-report.html   # interroger le port 3142
```

Résultat attendu :

```
Removing apt-cacher-ng (3.7.5-1) ...
Purging configuration files for apt-cacher-ng (3.7.5-1) ...
inactive
curl -> 000 (code retour 7)
```

`000` et le code retour `7` de `curl` signifient « connexion impossible » : plus rien n'écoute sur le port.

Restaurer (le conteneur doit être arrêté avant le retour arrière) :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Arrêter, revenir au snapshot, redémarrer
pct stop 200
pct rollback 200 cache-ok   # revenir au snapshot
pct start 200   # redémarrer le conteneur

# Le service est revenu… et son cache aussi : le paquet de l'étape 5 bis sort
directement du cache
pct exec 200 -- systemctl is-active apt-cacher-ng
curl -x http://10.<site>.<N>.50:3142 -s -o /dev/null -w "après retour : HTTP %
{http_code}, %{time_total} s\n" $U   # nouveau téléchargement à travers le cache
pct exec 200 -- tail -1 /var/log/apt-cacher-ng/apt-cacher.log   # dernière ligne
du journal du cache
```

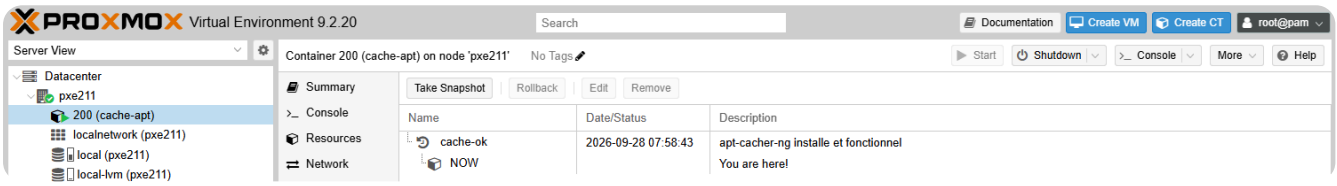
Résultat attendu :

Ce support est la propriété de Dawan France.

```
active
après retour : HTTP 200, 0.003473 s
1791380522|0|53507|10.59.1.11|debrep/pool/main/h/hello/hello_2.10-5_amd64.deb
```

Une seule ligne `0`, aucune ligne `I` : le fichier n'a pas été retéléchargé. Le snapshot avait figé tout le système de fichiers, cache compris. (Si votre Shell a été rouvert entre-temps, redéfinissez `U` comme à l'étape 5 bis.)

Pendant `pct rollback`, LVM affiche des avertissements `WARNING: Sum of all thin volume sizes … exceeds the size of thin pool` : c'est le sur-provisionnement du pool thin : les volumes promis dépassent la place réelle. Il n'empêche rien ici, mais il se surveille.



L'arbre des snapshots : `cache-ok`, avec sa date et sa description, et `NOW` juste en dessous — l'état courant repart de ce snapshot. **Rollback** ramène le conteneur à `cache-ok` ; il faut sélectionner le snapshot avant de cliquer.

> **Ce qui vient de se passer :** le système de fichiers est revenu exactement à l'état du snapshot. La désinstallation a été annulée entièrement, et le contenu du cache avec elle.

Étape 8 — Sauvegarde vzdump

Via l'interface : CT 200 → Backup → Backup Now — Storage `local`, Mode `Snapshot`, Compression `ZSTD`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

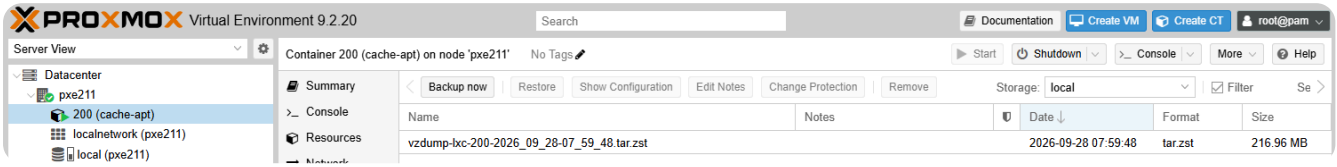
```
# Sauvegarde à chaud (mode snapshot), compressée en zstd, sur le stockage local
vzdump 200 --storage local --compress zstd --mode snapshot

# Lister les sauvegardes du stockage local, via l'API
pvesh get /nodes/$(hostname)/storage/local/content --content backup
```

Résultat attendu (fin de `vzdump`) :

Ce support est la propriété de Dawan France.

```
INFO: Total bytes written: 730071040 (697MiB, 189MiB/s)
INFO: archive file size: 206MB
INFO: Finished Backup of VM 200 (00:00:04)
INFO: Backup job finished successfully
```



L'archive `vzdump-lxc-200-2026_09_28-07_59_48.tar.zst`, au format `tar.zst`, sur le stockage `local`. Le nom contient la date et l'heure de la sauvegarde.

Les 697 Mio du conteneur sont devenus environ 206 Mo après compression zstd (l'interface affiche `216.96 MB`, en millions d'octets). **Restore** recrée le conteneur à partir de cette archive.

> **Snapshot vs Backup :** le snapshot est un instantané sur le même disque, non transportable — pour les retours arrière rapides avant une opération risquée. Le backup vzdump est une archive complète, transportable sur un autre stockage ou nœud — seule vraie protection contre la perte du disque physique. En production, les deux sont complémentaires.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# État du conteneur, du service, du snapshot et de la sauvegarde
pct status 200
pct exec 200 -- systemctl is-active apt-cacher-ng   # le service
pct listsnapshot 200   # les snapshots
pvesh get /nodes/$(hostname)/storage/local/content --content backup   # les
sauvegardes du stockage local
```

Résultat attendu : `status: running`, `active`, le snapshot `cache-ok` listé, et une ligne `vzdump-lxc-200-…tar.zst` dans les sauvegardes.

Ce support est la propriété de Dawan France.

Chapitre 2 — Machines Virtuelles KVM

Le chapitre précédent s'est terminé sur un conteneur qui sert une page web, léger et démarré en deux secondes. Mais il partage le noyau de l'hôte : il ne peut donc faire tourner ni Windows, ni un pare-feu BSD, ni un noyau choisi. Ce chapitre passe à l'autre modèle de Proxmox, la **machine virtuelle KVM** : un ordinateur complet, avec son propre micrologiciel, son propre noyau, ses propres pilotes.

Le fil conducteur reste votre nœud. Vous y créez d'abord une VM à la main, onglet par onglet, pour comprendre chaque choix de l'assistant. Puis vous changez de méthode : vous fabriquez un **modèle** (*template*) à partir d'une image cloud.

Vous le clonez en quelques secondes, et cloud-init personnalise chaque clone au premier démarrage. Le chapitre se termine sur ce qui protège ces VM : snapshots, sauvegardes, rétention. La VM clonée ici servira jusqu'au dernier chapitre.

Objectifs du chapitre

À l'issue de ce chapitre, les stagiaires seront capables de :

- **Choisir** entre VM KVM et conteneur LXC, et expliquer la chaîne KVM/QEMU/Proxmox
- **Créer et administrer** une VM KVM (GUI et `qm`), son cycle de vie, sa console et ses médias ISO
- **Configurer** le stockage LVM-thin et le réseau VirtIO d'une VM
- **Préparer** une VM avec cloud-init, la transformer en template, et la cloner en série
- **Utiliser** les snapshots, effectuer un rollback, et sauvegarder une VM avec vzdump

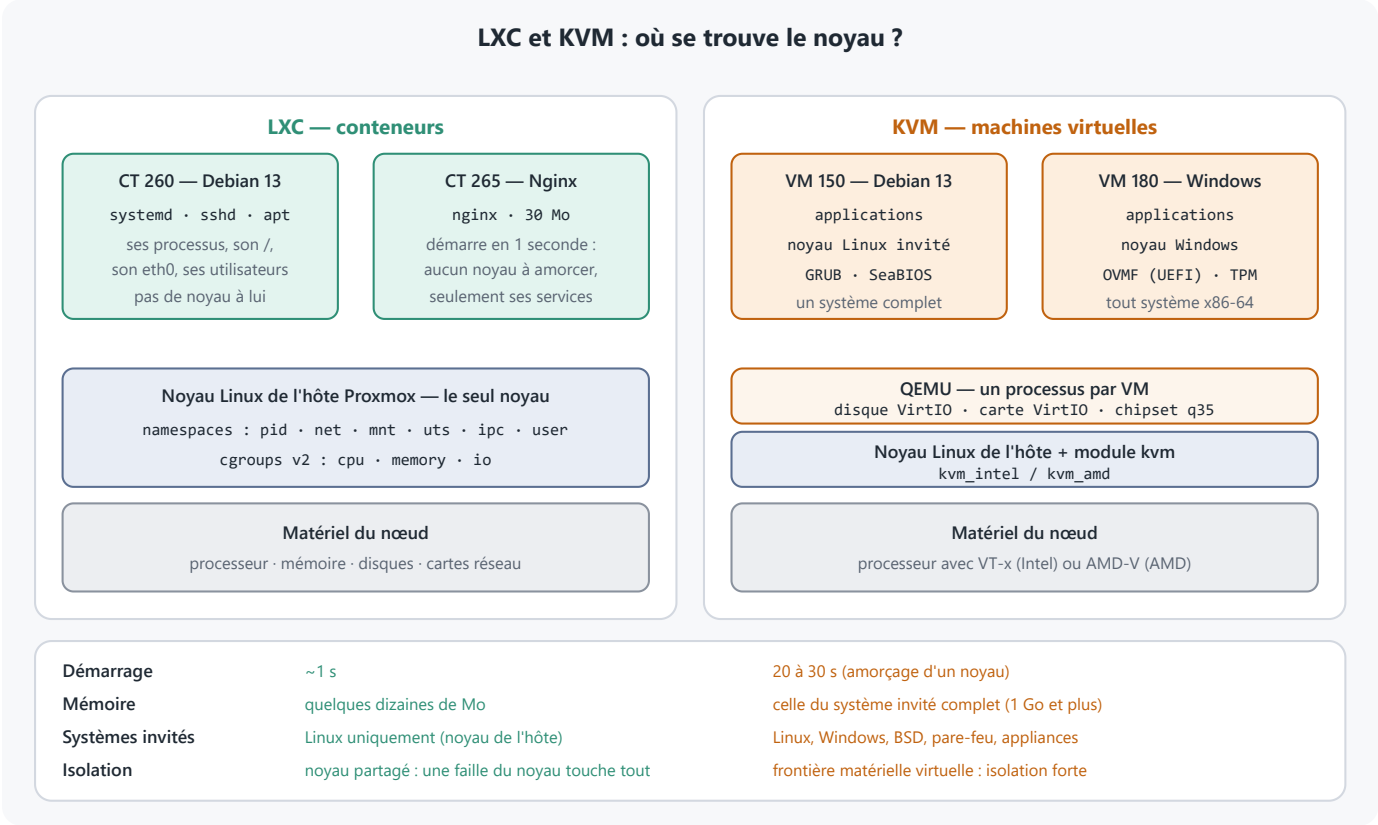
Dawan
Nous formons votre avenir
Ce support est la propriété de Dawan France.

1. VM KVM versus conteneur LXC — choisir le bon outil

1.1 Deux philosophies de virtualisation : conteneur et machine virtuelle

Proxmox VE propose deux façons radicalement différentes d'exécuter un système : le conteneur LXC, abordé au chapitre précédent, et la machine virtuelle KVM, qui fait l'objet de ce chapitre. Un conteneur partage le noyau Linux de l'hôte Proxmox — il est rapide à démarrer, léger en mémoire, mais limité aux systèmes Linux et exposé à certains risques d'isolation.

Une machine virtuelle KVM, elle, possède son propre noyau, son propre BIOS ou UEFI, son propre espace mémoire entièrement isolé. Le noyau hôte n'est pas partagé : la VM ne sait même pas qu'elle tourne sur Proxmox.



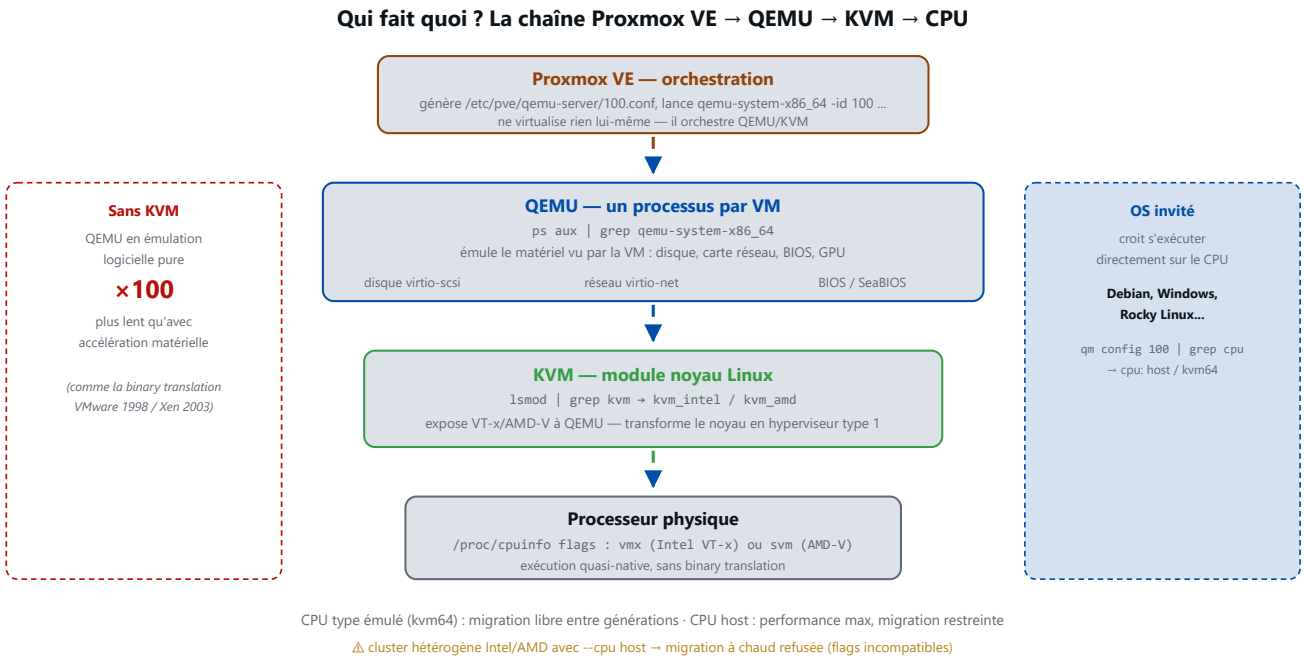
Cette isolation totale a un coût — quelques centaines de mégaoctets de RAM minimum, un démarrage qui prend quelques secondes, et une couche d'émulation matérielle. Mais elle ouvre des possibilités qu'un conteneur ne peut pas offrir : faire tourner Windows, un BSD, un kernel

Ce support est la propriété de Dawan France.

personnalisé. Ou des services qui s'attendent à trouver un vrai matériel sous leurs pieds : Home Assistant OS, pfSense, TrueNAS.

Le schéma ci-dessus résume cette opposition. À gauche, plusieurs conteneurs LXC empilés sur un seul noyau partagé. À droite, plusieurs VMs KVM totalement indépendantes, chacune avec son propre noyau, émulé par QEMU et accéléré par KVM.

En résumé, le conteneur mise sur la légèreté et le partage ; la VM, sur l'isolation et l'indépendance totale vis-à-vis de l'hôte. C'est ce compromis qui guide le choix entre les deux pour chaque nouveau service à déployer.



Le tableau ci-dessous met les deux technologies face à face, sur les critères qui décident réellement du choix. C'est lui qu'on garde en tête au moment de créer une nouvelle ressource :

Critère	LXC (conteneur)	KVM (machine virtuelle)
Noyau	Partagé avec l'hôte	Noyau propre à la VM
Démarrage	< 1 seconde	Plusieurs dizaines de secondes
Surcoût mémoire	Très faible (pas de noyau invité)	~200 à 500 Mo pour l'OS invité
Systèmes supportés	Linux uniquement	Linux, Windows, BSD, appliances…
Isolement	Namespaces + cgroups (noyau)	Virtualisation matérielle (KVM/QEMU)

Ce support est la propriété de Dawan France.

Critère	LXC (conteneur)	KVM (machine virtuelle)
Modules noyau propres	Non (noyau partagé)	Oui (noyau dédié)
Migration à chaud	Non (redémarrage requis)	Oui
Cas d'usage typique	Services Linux légers (web, DNS, proxy, supervision)	Windows, appliances réseau, isolation forte, noyau spécifique

1.2 Quand utiliser une VM KVM

Utilisez une VM KVM lorsque vous avez besoin de :

- Faire tourner **Windows** (Server ou Desktop) — impossible dans un LXC
- Exécuter **Docker** de façon propre, avec son propre noyau
- Déployer une **appliance réseau** : pfSense, OPNsense, VyOS — ces systèmes sont compilés pour un matériel spécifique
- Tester des **noyaux expérimentaux** ou des distributions alternatives sans risque pour l'hôte
- Obtenir une **isolation maximale** pour un service sensible. Un attaquant qui compromet une VM ne peut pas remonter vers l'hôte sans exploiter une vulnérabilité de l'hyperviseur. C'est autrement plus difficile que d'exploiter un espace de noms Linux

En résumé : si ce que vous installez a besoin d'un vrai système d'exploitation indépendant, choisissez une VM KVM. Pour tout ce qui est Linux léger (nginx, MariaDB, Pi-hole, DNS), un LXC reste le bon choix.

La règle pratique à retenir : pour un service Linux standard, sans modules noyau particuliers ni Windows, le conteneur LXC est le bon choix. Pour tout le reste, la VM KVM s'impose. Le choix n'est d'ailleurs jamais exclusif : sur un même nœud, VM et conteneurs cohabitent et partagent les mêmes stockages et les mêmes bridges.

Ils se sauvegardent avec les mêmes outils : `vzdump`, ou Proxmox Backup Server. Une infrastructure typique mêle donc les deux — des conteneurs pour les petits services Linux, des VM pour Windows, les appliances et tout ce qui exige une isolation forte.

Deux différences pèsent en exploitation et méritent d'être connues dès maintenant. La **migration à chaud** n'existe que pour les VM : un conteneur se déplace d'un nœud à l'autre en redémarrant. Et une VM utilise la mémoire qu'on lui attribue, *plus* celle de son propre noyau et de QEMU.

Un conteneur ne consomme que ce que ses processus utilisent réellement. D'où la densité très supérieure des conteneurs sur un même serveur.

Dawan
Ce support est la propriété de Dawan France.

À retenir : un **conteneur LXC** partage le noyau du nœud : léger, rapide, mais Linux seulement. Une **VM KVM** a son propre noyau et son matériel virtuel : on la choisit pour Windows, un autre noyau, une isolation forte ou une appliance. Pour un service Linux léger, le LXC reste le bon choix.

Ce support est la propriété de Dawan France.

2. KVM, QEMU et Proxmox : qui fait quoi ?

2.1 La chaîne de virtualisation

Comprendre la répartition des rôles entre KVM, QEMU et Proxmox évite beaucoup de confusion, notamment lorsqu'un message d'erreur nomme l'un ou l'autre de ces composants.

KVM — Kernel-based Virtual Machine — est un module du noyau Linux. Il ne fait qu'une chose : exposer les instructions de virtualisation matérielle du CPU (Intel VT-x ou AMD-V) à l'espace utilisateur.

KVM transforme le noyau Linux en hyperviseur de type 1. Sans KVM, QEMU fonctionnerait en émulation pure, en traduisant chaque instruction du processeur invité — dix fois plus lent, souvent davantage.

QEMU est l'émulateur. Il simule le matériel que voit la VM : carte réseau virtuelle, contrôleur de disque, carte graphique, BIOS.

Chaque VM en cours d'exécution est un processus QEMU sur l'hôte. C'est QEMU qui lit le fichier de configuration de la VM et instancie les périphériques virtuels correspondants.

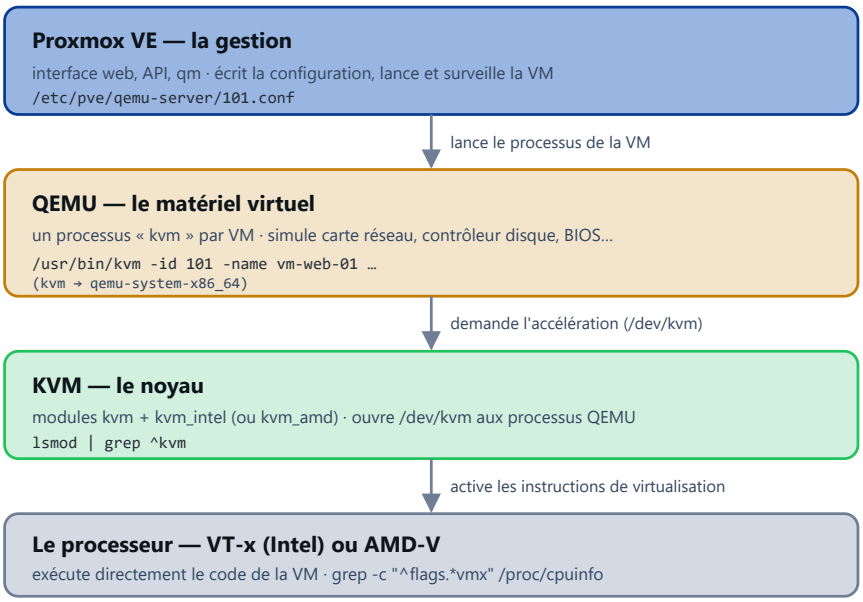
Proxmox VE est la couche de gestion au-dessus. Il crée les fichiers de configuration, appelle QEMU avec les bons paramètres, gère le stockage, le réseau, les sauvegardes, l'interface web. Proxmox ne virtualise rien directement — il orchestre QEMU/KVM.

- 📖 Documentation Proxmox — KVM Virtual Machines
- 📖 KVM — Kernel-based Virtual Machine, présentation Red Hat

Dawan
Ce support est la propriété de Dawan France.

Une VM Proxmox, quatre couches : qui fait quoi

Cliquez sur une couche



De haut en bas, chaque couche s'appuie sur la suivante. Un message d'erreur nomme l'une d'elles : savoir laquelle, c'est savoir où chercher.

Les quatre couches d'une VM en marche, de la gestion au matériel, avec ce qu'on en voit réellement sur un nœud du lab. Le fichier de configuration écrit par Proxmox, le processus `kvm` de la VM.

Puis les modules du noyau et l'extension du processeur. Cliquez sur une couche pour lire son rôle et le résultat correspondant.

Revenons à la définition de base : **virtualiser une machine**, c'est faire croire à un système qu'il a un ordinateur pour lui seul. Trois logiciels se partagent ce travail, avec chacun un rôle distinct :

Terme	Ce que c'est	Dans le laboratoire
Proxmox VE	La couche de gestion : il écrit la configuration de la VM et lance QEMU avec les bons paramètres	L'interface web et `qm`, sur votre nœud
Fichier de configuration	La description de la VM : processeur, mémoire, disques, cartes réseau	`/etc/pve/qemu-server/100.conf`
QEMU	Le programme qui simule le matériel de la VM (disque, carte réseau, BIOS) ; un processus par VM en marche	Le processus `/usr/bin/kvm -id 100`
KVM	Un module du noyau Linux, qui donne à QEMU l'accès direct au processeur	Les modules `kvm` et `kvm_intel` (ou `kvm_amd`)

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
VT-x / AMD-V	Les extensions de virtualisation du processeur ; sans elles, QEMU émule tout, au moins dix fois plus lentement	Le drapeau `vmx` (Intel) ou `svm` (AMD) dans `/proc/cpuinfo`

Quand un message d'erreur nomme `qm`, le problème est dans la couche de gestion ; `QEMU`, dans le processus de la VM ; `KVM`, dans le module du noyau ou le processeur. On sait ainsi où chercher.

Où voir chaque couche sur votre nœud :

Couche	Où la voir
Proxmox et sa configuration	La VM → **Hardware** et **Options** ; `qm config 100` ; le fichier `/etc/pve/qemu-server/100.conf`
QEMU, une VM en marche	`ps aux` : une ligne `/usr/bin/kvm -id <VMID>` par VM en marche (§2.3) ; `qm showcmd 100 --pretty` traduit la configuration en options de QEMU
KVM, le module du noyau	`lsmod` : les modules `kvm` et `kvm_intel` ou `kvm_amd` (§2.2)
Le processeur	Le drapeau `vmx` ou `svm` dans `/proc/cpuinfo` (commande du §2.2)

2.2 Vérifier les prérequis matériels

Avant de créer une VM, le CPU de l'hôte doit exposer ses extensions de virtualisation. Sur du matériel récent elles sont actives par défaut, mais il arrive que le BIOS/UEFI les désactive.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis le Shell du noeud Proxmox (pas depuis votre poste Windows)
# Verifier la presence des extensions de virtualisation dans /proc/cpuinfo
# "vmx" = Intel VT-x  |  "svm" = AMD-V
grep -m 1 -E '^flags.*\b(vmx|svm)\b' /proc/cpuinfo   # -m 1 : la première ligne trouvée

# Vérifier que les modules KVM sont bien chargés dans le noyau
lsmod | grep -E '^kvm'
```

Résultat attendu :

Ce support est la propriété de Dawan France.

```
flags: ... vmx ... (ou svm pour AMD)
kvm_intel             397312  0
kvm                  1032192  1 kvm_intel
```

Si la commande `grep` ne renvoie rien, les extensions sont absentes ou désactivées dans le BIOS. Sur les machines de l'atelier, cette vérification a été faite en amont lors du déploiement PXE — vous devriez voir `vmx` ou `svm` sans aucune action supplémentaire.

2.3 Une VM, un processus QEMU sur l'hôte

Chaque VM Proxmox correspond à un processus QEMU sur l'hôte :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Voir les processus QEMU actifs (une ligne par VM démarrée)
# Proxmox lance QEMU sous le nom « kvm » : c'est ce nom qu'on cherche
pgrep -a kvm
```

Résultat attendu (les lignes qui comptent) :

```
12345 /usr/bin/kvm -id 100 -name debian-base,debug-threads=on -no-shutdown -chardev
socket,id=qmp,path=/va
```

Chaque ligne est une VM. Proxmox lance QEMU par le lien `/usr/bin/kvm` : chercher `qemu-system` ne trouve donc rien. L'option `-id` porte le VMID.

C'est ce qui permet de relier un processus gourmand vu dans `top` à la VM responsable. Le PID donne accès aux informations détaillées via `/proc/<PID>/`. La ligne complète fait plusieurs milliers de caractères : c'est la traduction, option par option, du fichier de configuration de la VM, que `qm showcmd 100 --pretty` affiche lisiblement.

2.4 Types de CPU : emulated, host, custom

Proxmox propose plusieurs types de CPU avec un compromis performances/portabilité.

CPU émulé (ex: `kvm64`, `qemu64`) : QEMU présente un processeur générique. La VM peut migrer entre hôtes de générations différentes, mais n'a pas accès aux instructions avancées, comme AVX-512 (*Advanced Vector Extensions*, calcul vectoriel).

Dawan Ce support est la propriété de Dawan France.

CPU host : QEMU expose directement le CPU physique. Performances maximales, mais la VM ne peut migrer qu'entre hôtes avec des CPU strictement identiques.

CPU custom : profils intermédiaires qui exposent certaines extensions tout en gardant une compatibilité de migration entre nœuds d'une même génération.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Changer le type de CPU d'une VM arrêtée
qm set 100 --cpu host

# Voir le type de CPU actuel
qm config 100 | grep cpu
# → cpu: host
```

Résultat attendu :

```
root@pxe211:~# qm set 100 --cpu host
update VM 100: -cpu host
root@pxe211:~# qm config 100 | grep cpu
cpu: host
```

`qm set` renvoie toujours cette ligne `update VM <id>: -<option> <valeur>` : elle confirme l'écriture dans `/etc/pve/qemu-server/100.conf`, rien de plus.

> **Le changement de type de CPU ne prend effet qu'au prochain démarrage complet de la VM.** Un simple `reboot` depuis l'invité ne suffit pas — il faut `qm stop 100` puis `qm start 100`, car le processus QEMU doit être relancé avec les nouveaux paramètres.
>
> `cpu: host` expose au système invité toutes les instructions du processeur physique, ce qui donne les meilleures performances. **Mais c'est incompatible avec la migration à chaud vers un nœud au processeur différent** : la VM verrait des instructions disparaître en vol et planterait. Sur un cluster hétérogène, le type `x86-64-v2-AES` est le compromis à retenir.

> **Erreur fréquente en cluster hétérogène :** créer une VM avec `--cpu host` sur un cluster qui mêle des Intel et des AMD, ou des processeurs de générations différentes. La migration à chaud échoue, ou la VM plante à l'arrivée faute d'instructions attendues.

Dawan Ce support est la propriété de Dawan France.

Sur un tel cluster, utilisez `x86-64-v2-AES` — commun à tous les processeurs x86 de la dernière décennie — ou un profil personnalisé qui n'expose que les instructions présentes partout. `kvm64`, plus ancien encore, fonctionne aussi mais prive les VM d'instructions utiles comme l'accélération AES (*Advanced Encryption Standard*, chiffrement).

▶ À pratiquer maintenant — Atelier 1

Vérifier que le nœud sait virtualiser, et lire l'espace disponible avant de créer la première VM.

Ce support est la propriété de Dawan France.

3. Gestion des médias — ISOs et images

3.1 Uploader et gérer les ISOs

Le stockage `local` de Proxmox héberge les ISOs et les templates dans deux répertoires distincts sur le système de fichiers : `/var/lib/vz/template/iso/` pour les ISOs et `/var/lib/vz/template/cache/` pour les templates LXC.

Depuis l'interface web, naviguez vers `Datacenter → local → ISO Images → Upload`. Vous pouvez aussi uploader directement par URL si l'hôte a accès à internet.

Deux points expliquent des erreurs fréquentes à ce stade.

Tout stockage ne peut pas héberger tout type de contenu. Chaque stockage déclare ce qu'il accepte : ISOs, images de disque, sauvegardes, templates de conteneurs. C'est pourquoi un stockage bloc comme LVM-thin **n'apparaît pas** dans la liste au moment de téléverser une ISO : il ne sait stocker que des volumes de disque, pas des fichiers.

Il faut un stockage de type répertoire ou NFS. C'est le sens de la colonne *Content* dans `Datacenter → Storage`.

Uploader par URL est préférable au téléversement depuis le poste, et pas seulement pour la vitesse. Proxmox télécharge alors directement sur le nœud, sans faire transiter plusieurs gigaoctets par le navigateur.

Et il propose de **vérifier l'empreinte SHA-256** de l'image. Sur un ISO d'installation, cette vérification est indispensable — c'est ce qui garantit que l'image n'a pas été altérée ou tronquée.

L'emplacement réel sur le disque, `/var/lib/vz/template/iso/`, permet aussi de déposer une ISO en `scp` quand l'interface n'est pas pratique. Elle apparaît dans la liste dès qu'elle est dans ce répertoire, sans manipulation supplémentaire.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Copier une ISO depuis le réseau directement sur l'hôte
wget -P /var/lib/vz/template/iso/ \
  https://cdimage.debian.org/cdimage/archive/13.6.0/amd64/iso-cd/debian-13.6.0-amd64-netinst.iso
```

Ce support est la propriété de Dawan France.

```
# Lister les ISOs disponibles sur le stockage local
pvesm list local --content iso

# Attacher une ISO différente à une VM existante
qm set 100 --ide2 local:iso/debian-13.6.0-amd64-netinst.iso,media=cdrom

# Détacher le lecteur CD après installation
qm set 100 --ide2 none,media=cdrom
```

Résultat attendu de `pvesm list local --content iso` (nœud `pxe211` du lab, qui porte aussi une image cloud Ubuntu) :

```
Volid                                      Format  Type            Size VMID
local:iso/debian-13.6.0-amd64-netinst.iso  iso     iso       791674880
local:iso/noble-server-cloudimg-amd64.img  iso     iso       625256960
```

Chaque ISO est listée avec son identifiant de volume (`Volid`), son format, son type de contenu et sa taille en octets. Le `Volid` — `stockage:iso/nom` — est la référence à utiliser dans les commandes `qm set` .

3.2 Nettoyer les médias obsolètes

Le stockage `local` accumule rapidement des ISOs volumineuses au fil des installations — une ISO Debian netinst pèse environ 700 Mo, une ISO Windows Server dépasse souvent 5 Go. Sans nettoyage régulier, ce stockage fichier peut se remplir et empêcher l'upload d'une nouvelle image.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Lister les ISOs avec leur taille pour repérer les plus volumineuses
ls -lh /var/lib/vz/template/iso/

# Supprimer une ISO devenue inutile (ici l'ancienne révision de Debian 13)
rm /var/lib/vz/template/iso/debian-13.0.0-amd64-netinst.iso

# Vérifier l'espace libéré sur le stockage local
pvesm status | grep local
```

Ce support est la propriété de Dawan France.

Résultat attendu :

```
root@pxe211:~# ls -lh /var/lib/vz/template/iso/
total 5,3G
-rw-r--r-- 1 root root 632M 12 mars   09:14 debian-13.0.0-amd64-netinst.iso
-rw-r--r-- 1 root root 1,4G  8 avril  11:02 almalinux-9.4-x86_64-minimal.iso
-rw-r--r-- 1 root root 5,1G  2 mai    16:40 fr-fr_windows_server_2022.iso

root@pxe211:~# pvesm status | grep local
local            dir     active     98559220    22417684    71095996   22.75%
```

`pvesm status` donne les tailles en **kibioctets**, pas en octets ni en gibioctets : la colonne du milieu est l'espace utilisé, la suivante l'espace disponible. La dernière colonne, en pourcentage, est celle qu'on lit en pratique.

> `rm` **sur une ISO ne demande aucune confirmation et ne vérifie rien.** Si une VM est encore en cours d'installation depuis cette image, ou si son lecteur CD y pointe toujours, elle se retrouvera avec un périphérique manquant au prochain démarrage.
>
> Vérifiez avant de supprimer : `grep -l "nom-de-l-iso" /etc/pve/qemu-server/*.conf` liste les VM qui la référencent. Le passage par l'interface — **Datacenter** → **Storage** → **local** → **ISO Images** → **Remove** — fait ce contrôle et refuse la suppression d'une image utilisée.

Les ISO Windows sont les principales coupables d'un `local` saturé : 5 Gio pièce, et on en accumule vite plusieurs versions.

Une bonne pratique : ne conserver que les ISOs des versions activement utilisées dans l'atelier courant. Et noter, dans les notes de VM ou dans un pense-bête d'équipe, la version exacte utilisée pour construire chaque template. Cette information devient précieuse le jour où il faut reconstruire un template après une suppression accidentelle.

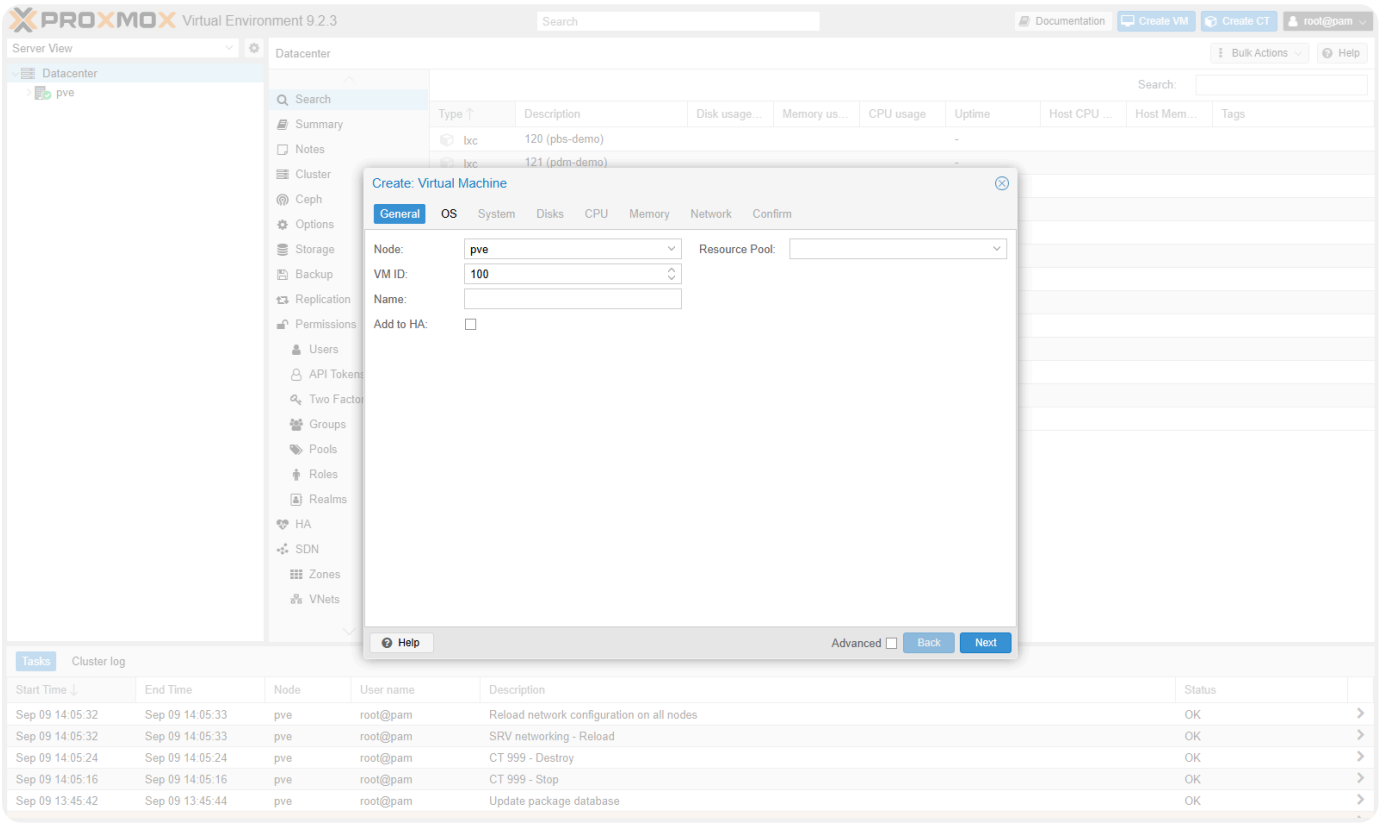
À retenir : le stockage `local` range les ISO dans `/var/lib/vz/template/iso/` et les templates LXC dans `.../cache/`. On y dépose une image par **Upload**, par URL ou par `wget`. Avant de supprimer une ISO, on vérifie qu'aucune VM ne l'utilise encore.

Dawan Ce support est la propriété de Dawan France.

4. Créer une VM KVM

4.1 L'assistant graphique pas à pas

Dans l'interface Proxmox, cliquez sur **Create VM** en haut à droite. L'assistant s'ouvre en plusieurs onglets. Parcourons-les dans l'ordre, car chaque choix influence les suivants.



L'assistant de création de VM — la barre d'onglets en haut est le fil conducteur de toute la section.

Repérez la **barre d'onglets** en haut de la fenêtre : `General`, `OS`, `System`, `Disks`, `CPU`, `Memory`, `Network`, `Confirm`. C'est exactement le plan de cette section — chaque sous-partie qui suit correspond à l'un de ces onglets, dans le même ordre.

L'assistant impose ce cheminement parce que les choix se conditionnent. Le type d'OS influence les valeurs par défaut du matériel ; le contrôleur de disque dépend du système visé, et ainsi de suite. On avance donc onglet par onglet, sans revenir en arrière sans raison.

Onglet General

Ce support est la propriété de Dawan France.

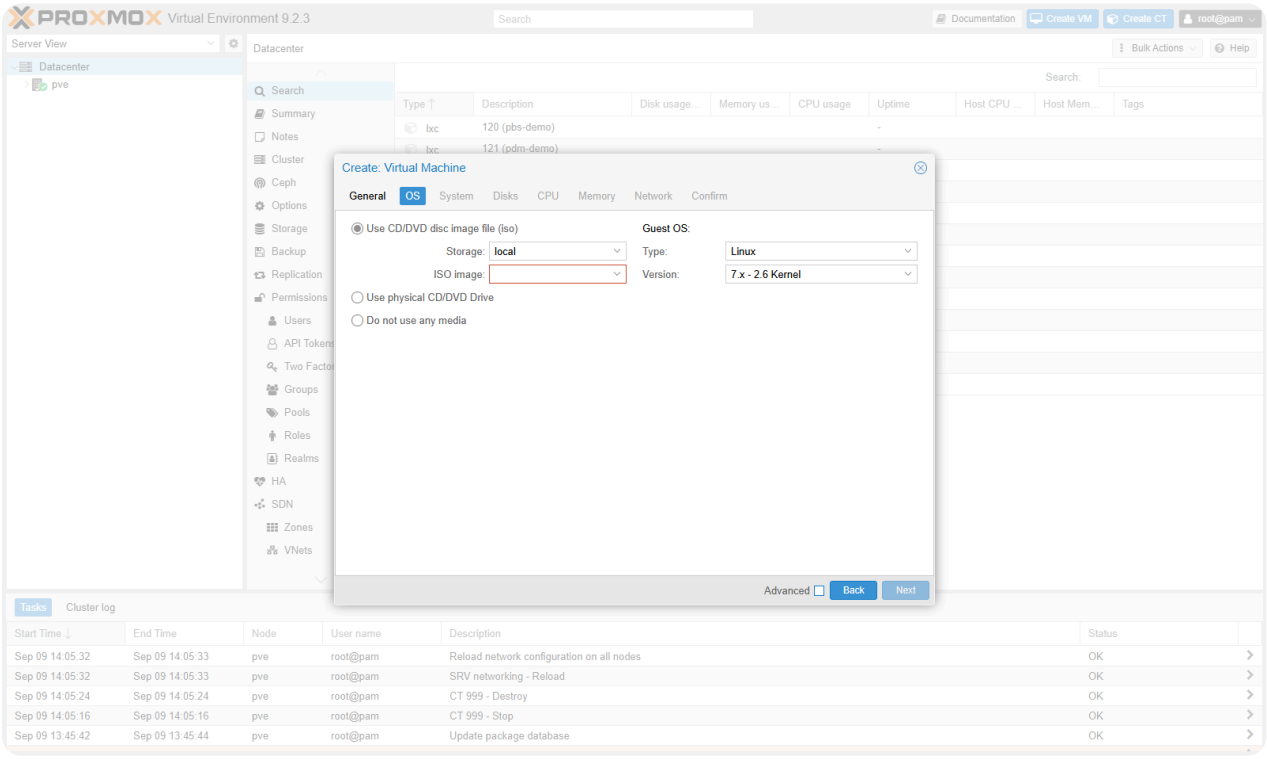
L'identifiant VM (VM ID) est un numéro unique dans le datacenter, partagé entre VM et conteneurs : une VM 200 et un conteneur 200 ne peuvent pas coexister. Ce support suit la convention des ateliers : **100 à 199 pour les VM, 200 à 299 pour les conteneurs, 9000 et plus pour les modèles**.

Toute convention fait l'affaire, pourvu qu'elle soit tenue. Donnez un nom explicite — ce nom apparaît dans l'interface et dans les logs.

Champ	Valeur exemple	Remarque
Node	pve	nœud actuel
VM ID	105	unique dans le datacenter
Name	vm-debian	sans espace, sans accent
Start at boot	décoché	à activer en production

Onglet OS

Sélectionnez l'ISO depuis le stockage `local`. Si elle n'y est pas encore, chargez-la d'abord depuis `Datacenter → local → ISO Images → Upload`.



L'onglet **OS**, coupé en deux : le support à gauche, le système invité à droite.

- **Le champ `ISO image` est encadré de rouge et vide.** L'assistant refuse de passer à l'onglet suivant tant qu'il l'est. Si la liste déroulante ne propose rien alors que vous avez chargé une

ISO, vérifiez le champ `Storage` juste au-dessus : il doit pointer sur le stockage qui contient réellement l'image.

- **Trois options exclusives à gauche.** `Use CD/DVD disc image file (iso)` est le cas normal : la VM lit l'image ISO d'un CD ou d'un DVD (disque optique).

  `Use physical CD/DVD Drive` n'a de sens que sur un serveur avec un lecteur, ce qui est devenu rare. `Do not use any media` sert quand le disque est déjà prêt — c'est le cas d'une VM bâtie depuis une image cloud, décrite au §10.

- `Type` **et** `Version` **ne changent pas l'installation.** Ils règlent la façon dont QEMU présente le matériel virtuel à l'OS. Se tromper ne bloque rien, mais dégrade les performances ou l'horloge.

- `Version` **ne propose que deux entrées sous PVE 9 :** `7.x - 2.6 Kernel` **et** `2.4 Kernel`**.** Ce n'est pas la version de votre distribution : c'est un marqueur de génération de noyau.

  Toute distribution moderne prend la première, y compris Debian 13. La seconde n'existe que pour des systèmes des années 2000. En ligne de commande, ces deux valeurs s'appellent `l26` et `l24`.

Champ	Valeur	Remarque
Use CD/DVD disc image	✓	le cas normal
Storage	`local`	le stockage qui contient l'ISO
ISO image	`debian-13.x-amd64-netinst.iso`	chargée au préalable
Guest OS Type	`Linux`	
Version	`7.x - 2.6 Kernel`	valeur unique pour toute distribution actuelle

Onglet System

Champ	Valeur	Remarque
Machine	`q35`	à choisir : le défaut est `i440fx`
BIOS	`SeaBIOS`	ou OVMF (*Open Virtual Machine Firmware*) pour UEFI — Debian supporte les deux

Ce support est la propriété de Dawan France.

Champ	Valeur	Remarque
SCSI	`VirtIO SCSI`	
Controller	`single`	
Qemu Agent	✓ coché	important : permet à Proxmox de communiquer avec la VM

Pourquoi `q35` plutôt que le défaut. Le champ **Machine** choisit la carte mère que QEMU simule, et la liste n'offre que deux choix.

`Default (i440fx)` reproduit un chipset Intel de 1996, avec un bus PCI (*Peripheral Component Interconnect*, le bus des cartes d'extension). `q35` reproduit un chipset Intel plus récent, qui fournit un bus **PCI Express** virtuel.

La documentation de Proxmox en tire la conséquence pratique : `q35` est le choix à faire pour passer une carte PCIe physique à la VM. C'est le *PCI passthrough*, pour une carte graphique ou une carte réseau dédiée.

C'est aussi avec `q35` qu'on peut activer un vIOMMU. Les VM des ateliers sont toutes en `q35`.

Changer de chipset après l'installation n'a pas empêché Debian de démarrer sur le lab. Une VM installée en `i440fx`, passée en `q35`, a redémarré avec la même carte `ens18` et la même adresse.

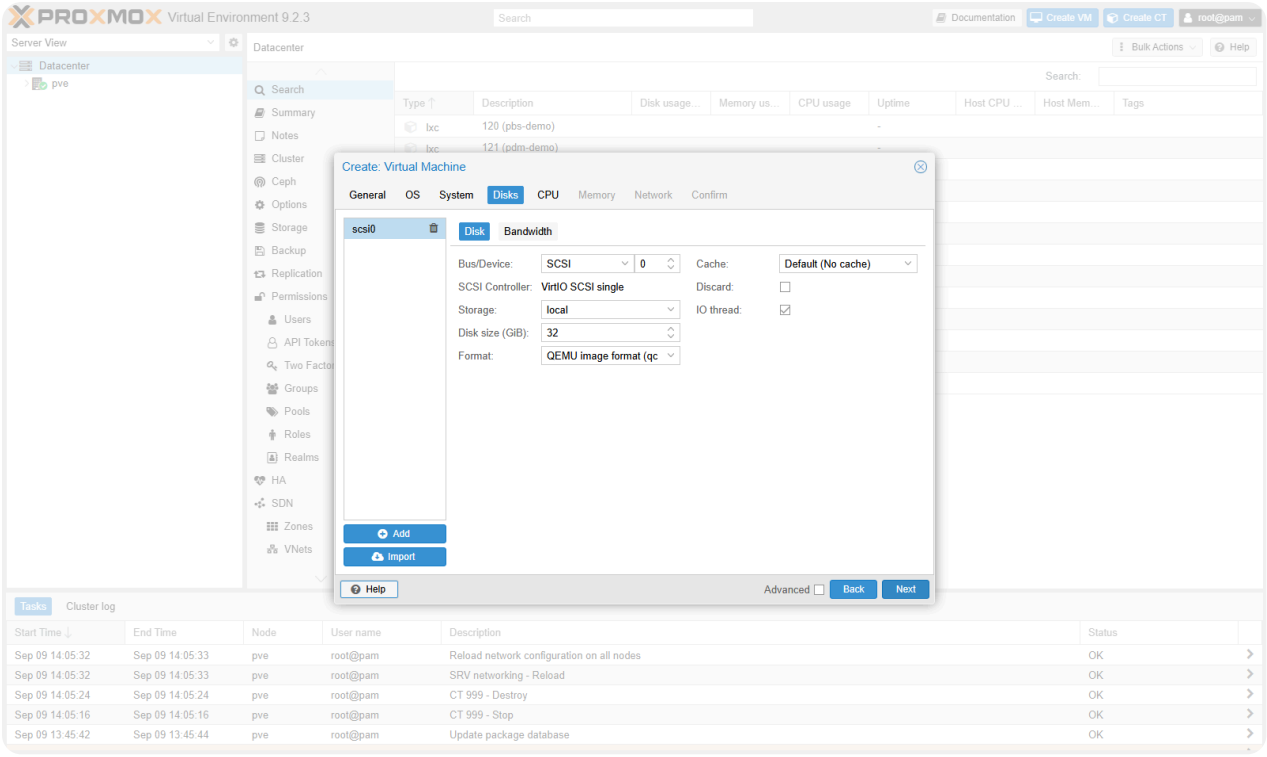
Mais l'ordre de ses deux disques a changé : le disque système est devenu `/dev/sdb`. Un montage écrit avec `/dev/sdb` aurait alors visé le mauvais disque.

D'où deux règles : choisir `q35` à la création. Il faut aussi monter les disques par étiquette (`LABEL=`) ou par UUID (*Universally Unique Identifier*, identifiant unique du système de fichiers), jamais par leur lettre.

Si vous optez pour OVMF (UEFI), Proxmox ajoutera automatiquement un disque EFI — ne le supprimez pas.

Onglet Disks

Ce support est la propriété de Dawan France.



L'onglet **Disks**, avec un disque `scsi0` déjà présent dans la colonne de gauche — l'assistant en crée toujours un.

- **`Storage` vaut `local` par défaut, et ce n'est presque jamais ce qu'on veut.** `local` est un stockage de type répertoire : il accepte les disques de VM, mais il vit sur la partition système du nœud, en `qcow2`. Basculez sur **`local-lvm`**, qui est un pool LVM-thin — c'est lui qui donne le provisionnement fin et les instantanés efficaces expliqués au §6.
- **Le champ `Format` disparaît quand vous choisissez `local-lvm`.** Ce n'est pas un bug : un volume LVM est toujours brut (`raw`), le format n'a pas lieu d'être. Sa présence ou son absence vous dit donc, d'un coup d'œil, sur quel type de stockage vous êtes.

- **`IO thread` est coché par défaut, `Discard` ne l'est pas.** `IO thread` sort les entrées-sorties du fil principal de QEMU : à laisser.

  `Discard` fait remonter le TRIM (la commande qui signale les blocs libérés) de la VM au stockage, ce qui **restitue réellement l'espace** quand la VM efface des fichiers. Sur SSD (disque à mémoire flash) ou LVM-thin, cochez-le — sans lui, un disque provisionné finement ne dégonfle jamais.

- **`Cache` sur `Default (No cache)` est le bon choix.** Les modes `writeback` accélèrent les écritures en les gardant en mémoire de l'hôte — au prix d'une perte de données si le serveur tombe. On ne s'en écarte qu'en sachant précisément pourquoi.

Ce support est la propriété de Dawan France.

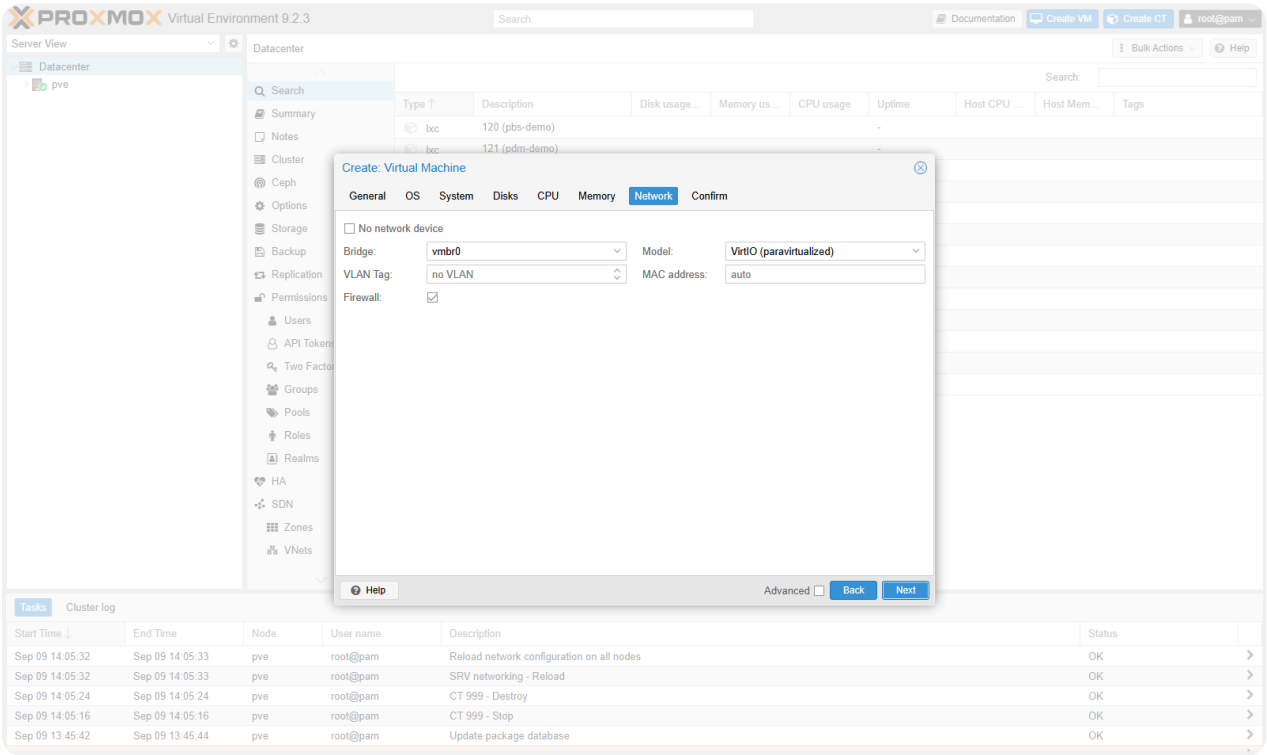
- **Les boutons `Add` et `Import`, en bas à gauche.** `Add` ajoute un second disque dès la création. `Import` récupère un disque existant, typiquement lors d'une migration depuis VMware.

Champ	Valeur	Ce que ça décide
Bus/Device	`SCSI 0`	Le bus vu par l'OS ; SCSI + VirtIO est le couple performant
Storage	`local-lvm`	Pool LVM-thin, pas la partition système
Disk size	`32` Go	Taille annoncée ; l'espace réel suit l'usage
Cache	`Default (No cache)`	Sûr par défaut
Discard	✓	Restitue l'espace effacé au pool
IO thread	✓	Entrées-sorties hors du fil principal

Onglets CPU et Memory

Pour une VM d'atelier : 2 cœurs, 1024 ou 2048 Mo de RAM. En production, on calque sur les besoins réels du service.

Onglet Network



L'onglet **Network**, cinq champs seulement — mais trois pièges.

Ce support est la propriété de Dawan France.

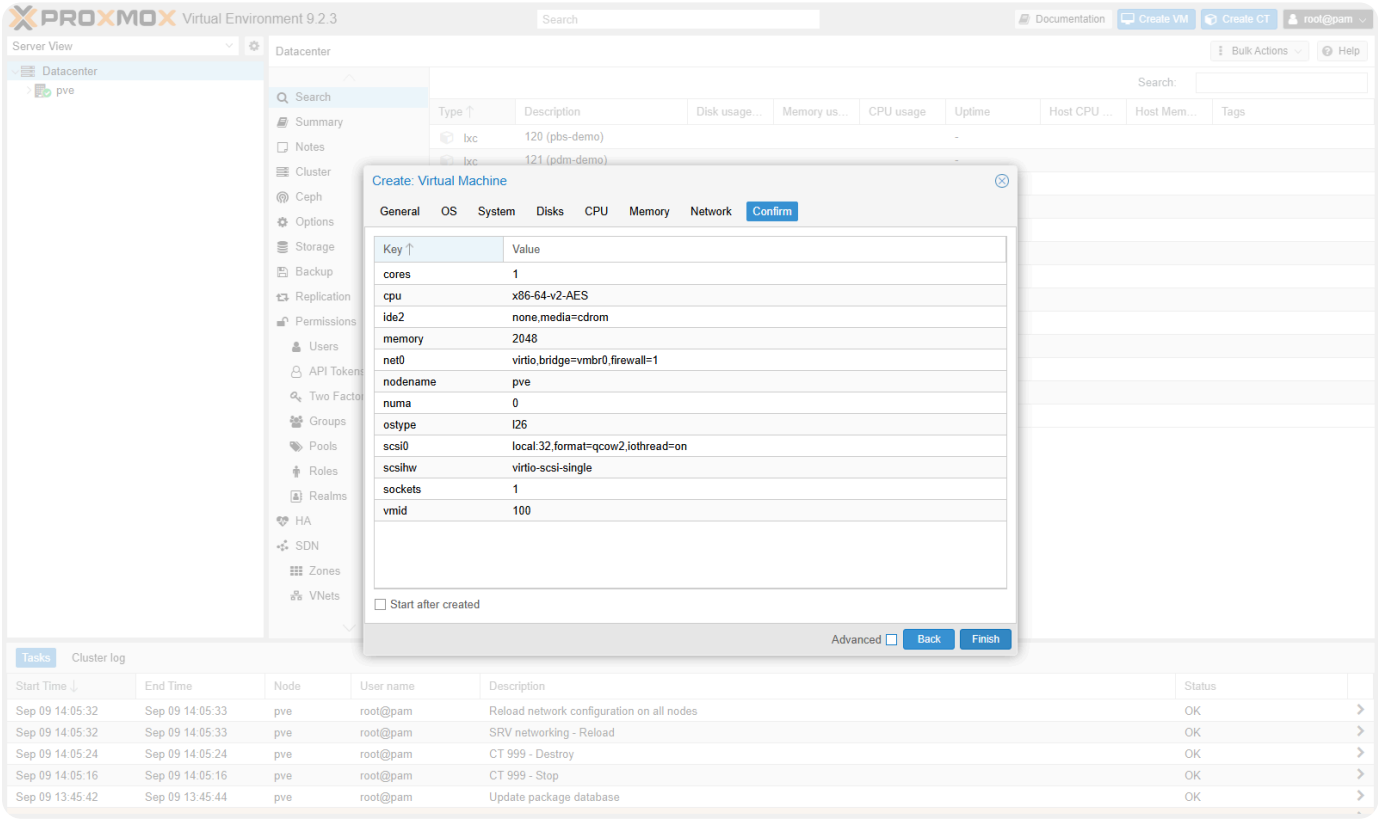
- `Bridge` **vaut** `vmbr0`**, le pont d'administration.** C'est le bon choix pour une VM qui doit être jointe depuis la salle. La liste propose tous les ponts du nœud.
- `Model` **vaut** `VirtIO (paravirtualized)`**, et il faut le laisser.** VirtIO signifie que la VM *sait* qu'elle est virtualisée et coopère avec l'hyperviseur au lieu de faire semblant d'avoir une carte physique. Les modèles émulés (`Intel E1000` , `Realtek RTL8139`) existent pour les systèmes anciens qui n'ont pas les pilotes VirtIO ; ils divisent le débit par plusieurs.

- `Firewall` **est coché par défaut — et c'est le piège le plus courant du pare-feu.** Cette case n'active rien à elle seule : le pare-feu Proxmox a trois interrupteurs (datacenter, nœud, machine) et il les faut tous les trois.

  Mais le jour où vous activerez le pare-feu au niveau du datacenter, **cette case déjà cochée le rendra actif sur cette carte**, avec la politique par défaut. Une VM subitement injoignable après une manipulation de pare-feu vient presque toujours de là.

- `VLAN Tag` **sur** `no VLAN` **.** À renseigner uniquement si `vmbr0` est déclaré *VLAN-aware* et que vous voulez isoler cette VM dans un VLAN précis.

- `MAC address` **sur** `auto` **.** Laissez : Proxmox génère une adresse dans sa plage réservée. Une adresse figée à la main n'a de sens que si un DHCP externe distribue des baux statiques.



Ce support est la propriété de Dawan France.

L'onglet **Confirm**, sur une VM d'essai créée sans rien changer aux valeurs proposées : le récapitulatif de ce qui sera réellement envoyé, une clé par ligne. Trois lignes se lisent avant de cliquer **Finish**. `scsi0 local:32,format=qcow2` : le disque est parti sur `local`, le stockage proposé par défaut, au lieu de `local-lvm` — le piège du §4.3.

`net0 …firewall=1` : la case Firewall est déjà cochée. `ide2 none,media=cdrom` : un lecteur de CD vide, ajouté d'office. La case **Start after created** reste décochée : on relit d'abord la configuration.

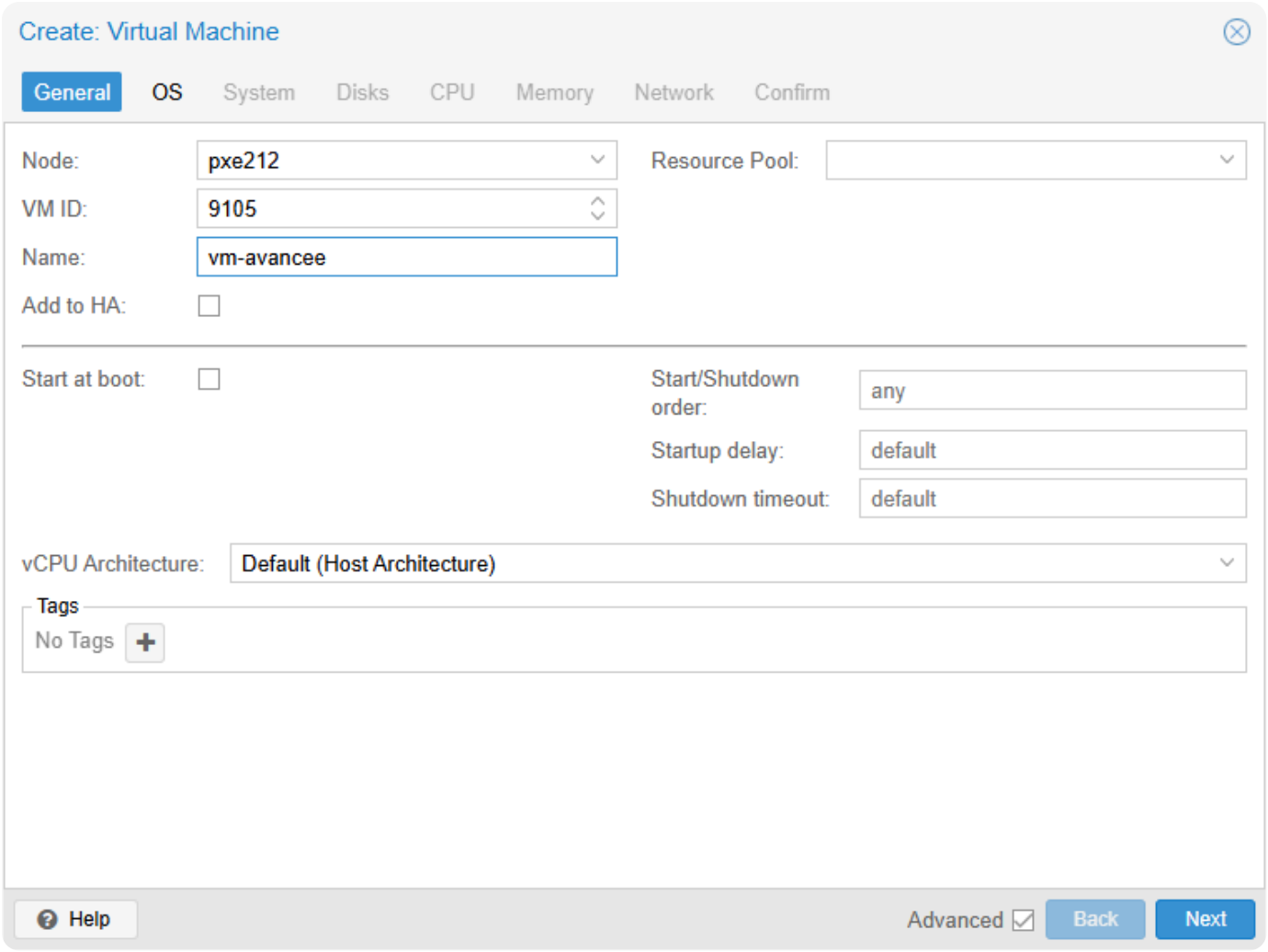
Les options avancées : la case `Advanced`

En bas de l'assistant, à côté du bouton **Back**, la case **Advanced** est décochée par défaut. Cochée, elle fait apparaître dans chaque onglet des champs supplémentaires, sous un trait horizontal.

Ils ont tous une valeur par défaut correcte pour une première VM : on ne les touche que pour un besoin précis. Mais il faut savoir qu'ils existent, car c'est là que se règlent le démarrage automatique, le plafond de processeur ou l'exclusion d'un disque des sauvegardes. Tous restent modifiables après la création, dans l'onglet **Hardware** ou **Options** de la VM.

Sur `pxe212` (Proxmox VE 9.2.21), avec une VM d'essai non créée.

Dawan
Nous formons votre avenir
Ce support est la propriété de Dawan France.



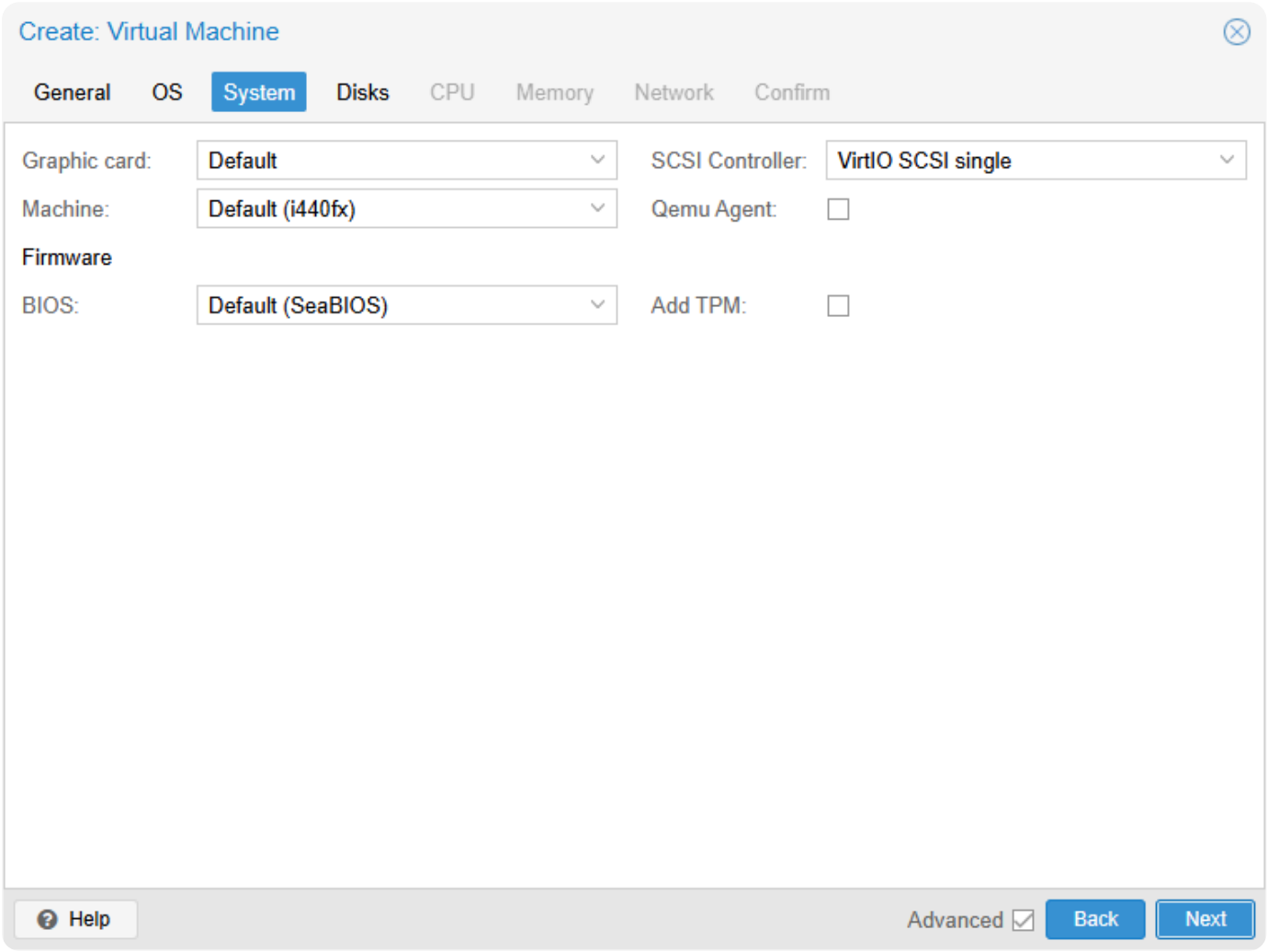
L'onglet **General** en mode avancé : sous le trait, le démarrage automatique et son ordonnancement, puis l'architecture du processeur virtuel et les tags. La case **Advanced** cochée se voit en bas à droite.

Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
General	Add to HA	décoché	Confie la VM à la haute disponibilité, dans un cluster
General	Resource Pool	vide	Range la VM dans un pool, pour les droits (chapitre 1)
General	Start at boot	décoché	La VM démarre avec le nœud (`onboot`)
General	Start/Shutdown order	`any`	Priorité de démarrage : `1` démarre en premier. L'arrêt se fait dans l'ordre inverse. Une VM sans ordre démarre après celles qui en ont un
General	Startup delay	`default`	Secondes d'attente après le démarrage de cette VM, avant de démarrer la suivante

Ce support est la propriété de Dawan France.

Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
General	Shutdown timeout	`default`	Secondes laissées à la VM pour s'éteindre ; 180 par défaut. Au-delà, Proxmox l'arrête de force
General	vCPU Architecture	`Default (Host Architecture)`	L'architecture émulée. Une autre que celle du nœud (`aarch64` , l'ARM — autre famille de processeurs — en 64 bits) se passe des extensions du processeur : beaucoup plus lent
General	Tags	aucun	Étiquettes pour filtrer les VM dans l'arborescence (chapitre 1)

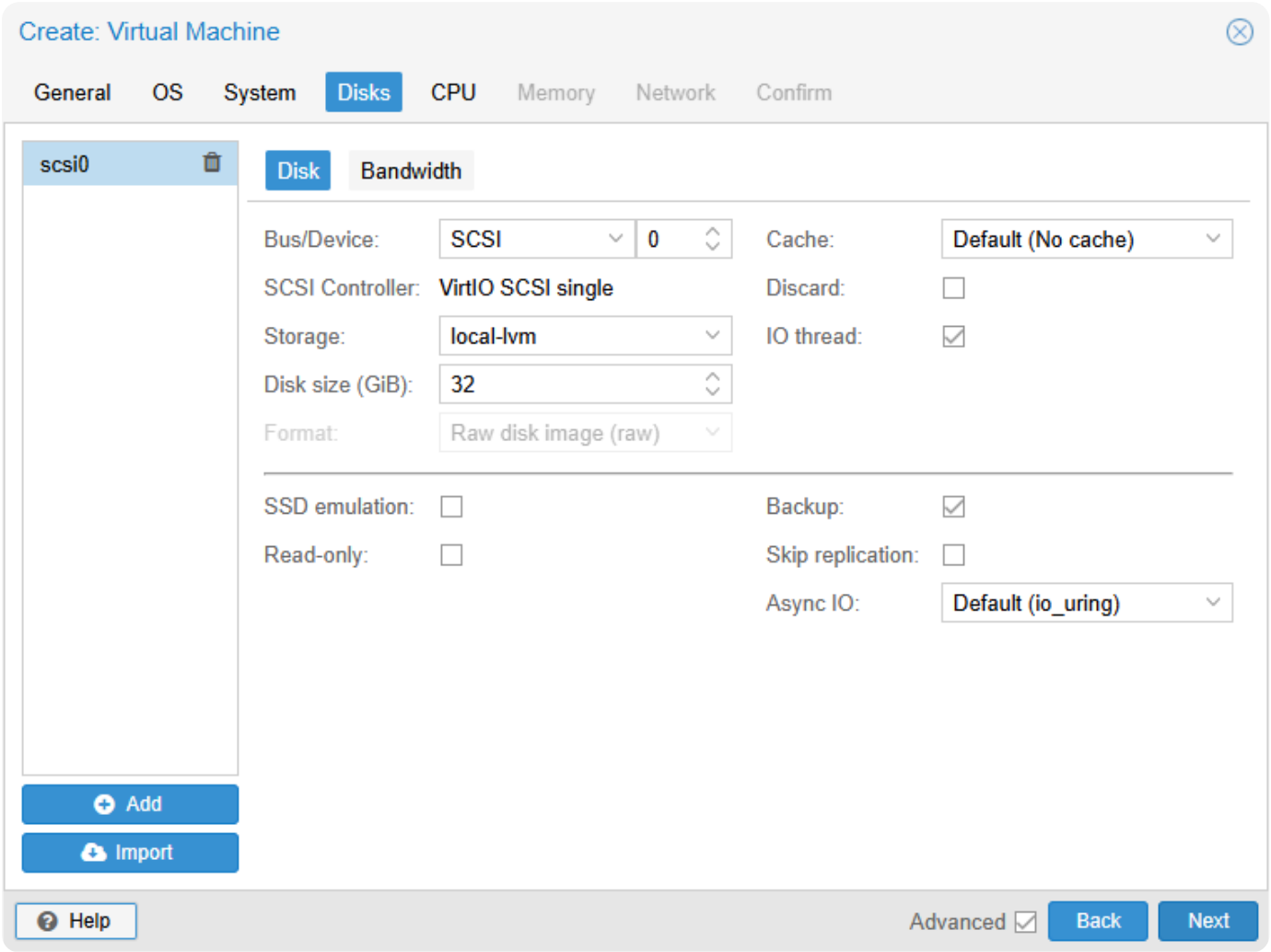
L'ordre de démarrage ne vaut qu'entre les VM d'un même nœud, pas sur tout le cluster. Cas typique : le serveur de base de données en `1` , avec un délai de 60 secondes, puis les serveurs d'application en `2` .



L'onglet **System** en mode avancé : deux champs de plus, **Graphic card** et **Add TPM**.

Ce support est la propriété de Dawan France.

Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
System	Graphic card	`Default`	La carte graphique émulée. `Default` donne `std`, une carte VGA (*Video Graphics Array*) standard. `Serial terminal 0` remplace l'écran par une console série : c'est le choix des images cloud (§10)
System	Add TPM	décoché	Ajoute une puce TPM virtuelle, qui garde des secrets comme des clés de chiffrement. Windows 11 l'exige. Une fois créée, elle ne se modifie plus : elle se supprime seulement

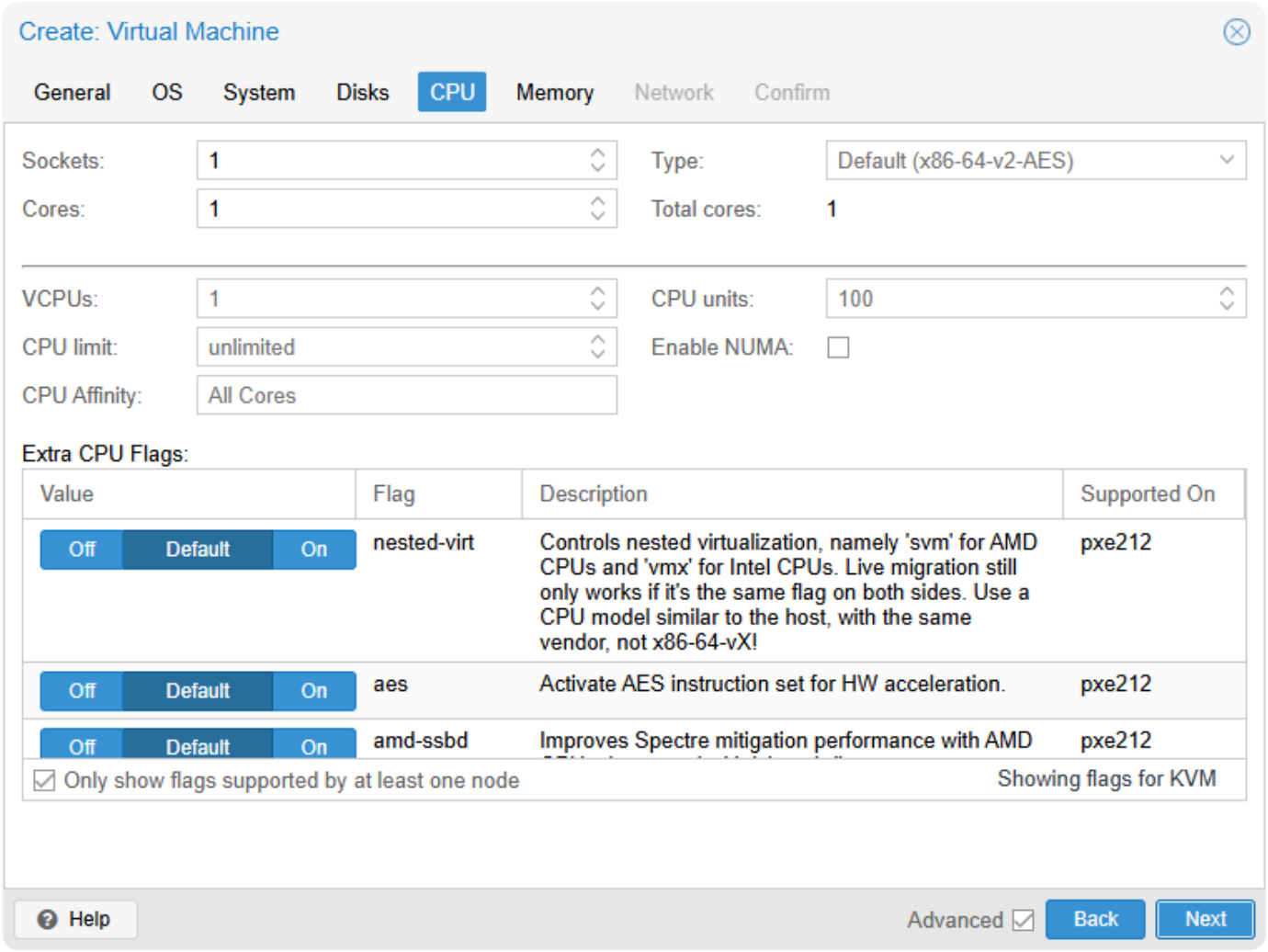


L'onglet **Disks** en mode avancé : cinq champs sous le trait, et un second onglet, **Bandwidth**, pour les limites de débit.

Ce support est la propriété de Dawan France.

Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
Disks	SSD emulation	décoché	Présente le disque comme un SSD et non comme un disque rotatif. Le système invité adapte alors sa gestion du disque
Disks	Read-only	décoché	Disque en lecture seule pour la VM
Disks	Backup	coché	Le disque est inclus dans les sauvegardes `vzdump`. Décoché : il est sauté. Utile pour un disque de cache ou de données temporaires
Disks	Skip replication	décoché	Exclut le disque de la réplication entre nœuds, qui exige un stockage ZFS
Disks	Async IO	`Default (io_uring)`	La méthode d'entrées-sorties de QEMU : `io_uring`, `native` ou `threads`. À laisser par défaut
Disks → Bandwidth	limites	aucune	Plafonds de lecture et d'écriture, en Mo/s et en opérations par seconde. Empêche une VM de saturer le stockage partagé

Ce support est la propriété de Dawan France.



L'onglet **CPU** en mode avancé : le partage du processeur du nœud entre les VM, puis la liste des drapeaux du processeur, avec leur description et les nœuds qui les supportent.

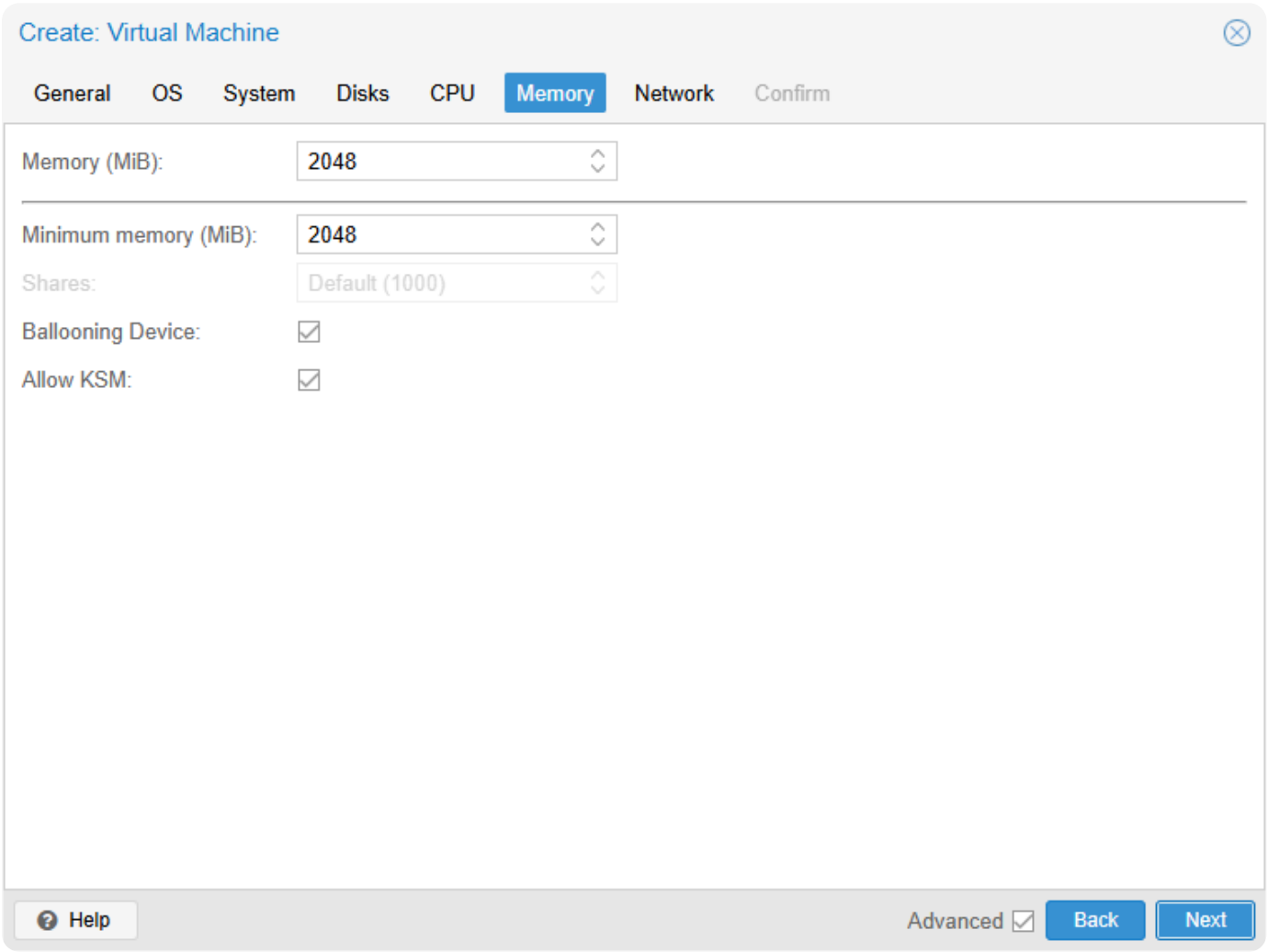
Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
CPU	VCPUs	`1` (= le total de cœurs)	Le nombre de processeurs branchés au démarrage, s'il doit être inférieur au total. Sert à en ajouter à chaud ensuite, sous Linux
CPU	CPU limit	`unlimited`	Plafond de temps processeur : `1` = un cœur du nœud à 100 %, `2.5` = 250 %. La VM garde ses cœurs, mais ne consomme pas plus
CPU	CPU Affinity	`All Cores`	Les cœurs du nœud autorisés à exécuter la VM, par exemple `0-3`. Ce n'est pas une mesure de sécurité, et cela complique la migration vers un nœud plus petit
CPU	CPU units	`100`	Le poids de la VM quand plusieurs VM se disputent le processeur. Une VM à `200` reçoit deux fois plus de temps

Ce support est la propriété de Dawan France.

Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
			qu'une VM à `100`
CPU	Enable NUMA	décoché	Présente à la VM la topologie mémoire des serveurs à plusieurs processeurs physiques
CPU	Extra CPU Flags	`Default` partout	Active ou retire une instruction précise, comme `aes` ou `nested-virt` (la virtualisation imbriquée)

CPU limit ou CPU units ? Le premier est un plafond, appliqué même quand le nœud est libre.

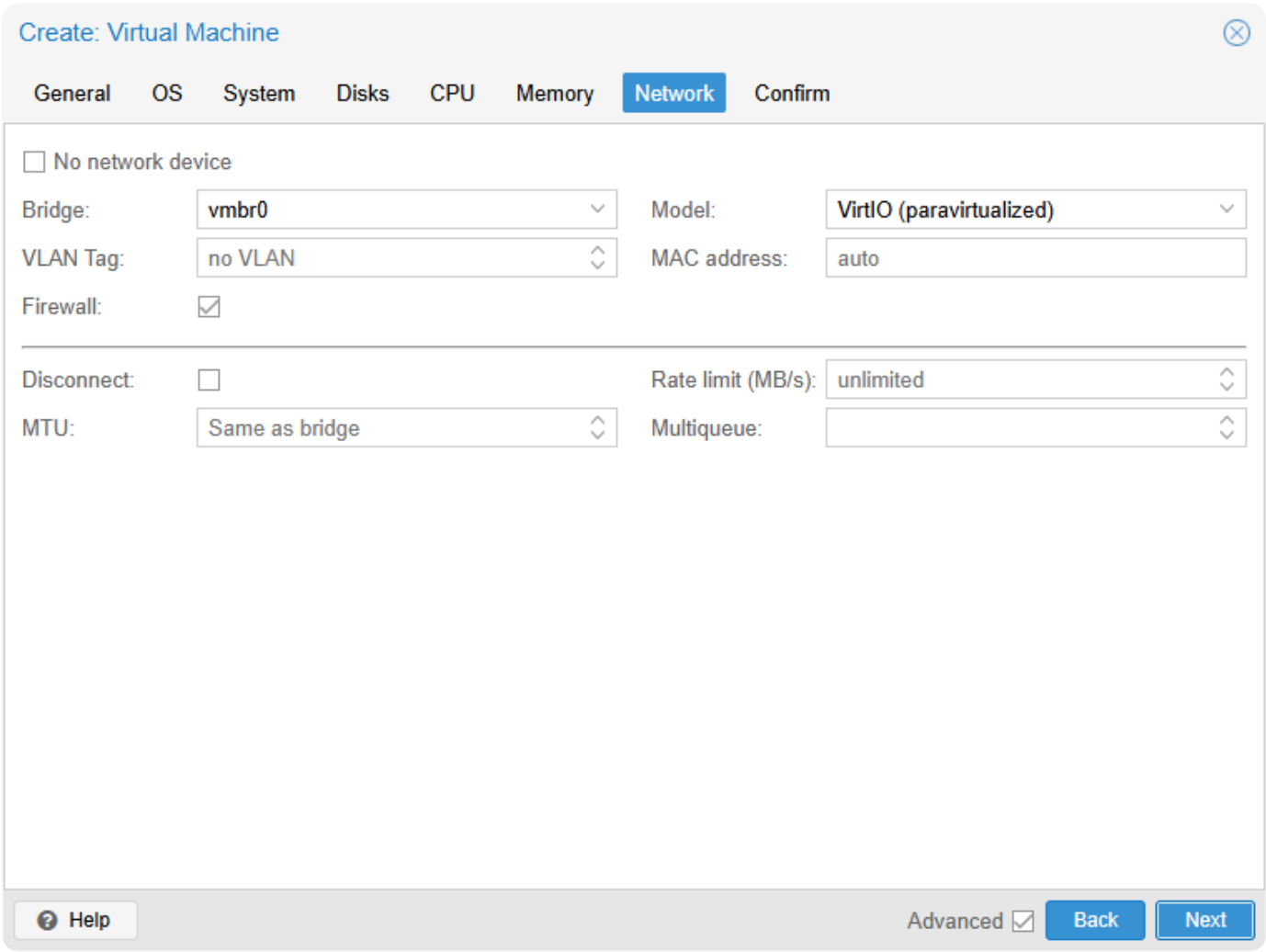
Le second est une priorité, qui ne joue que quand le processeur du nœud est saturé. En production, on règle d'abord les `CPU units`.



L'onglet **Memory** en mode avancé : le plancher du ballooning (§7.4), et deux cases cochées d'office.

Ce support est la propriété de Dawan France.

Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
Memory	Minimum memory (MiB)	égal à Memory (`2048`)	Le plancher du ballooning (`balloon`). Égal à **Memory** : la mémoire est fixe
Memory	Shares	`Default (1000)` , grisé	Le poids de la VM quand Proxmox redistribue la mémoire. Grisé tant que le plancher égale le maximum
Memory	Ballooning Device	coché	Le pilote de ballon. Même avec une mémoire fixe, il remonte l'usage réel de la mémoire de la VM. Décoché : `balloon: 0`
Memory	Allow KSM	coché	Autorise la fusion des pages mémoire identiques entre VM (KSM, Kernel Samepage Merging), qui économise la mémoire du nœud. Décoché : la VM en est exclue



Ce support est la propriété de Dawan France.

L'onglet **Network** en mode avancé : quatre champs sous le trait.

Onglet	Champ avancé	Valeur par défaut	Ce qu'il décide
Network	Disconnect	décoché	La carte est créée débranchée, comme un câble retiré (`link_down=1`). L'atelier 6 s'en sert pour restaurer une copie sans conflit d'adresse
Network	MTU	`Same as bridge`	La taille maximale des trames. Vide : celle du bridge
Network	Rate limit (MB/s)	`unlimited`	Plafond de débit de la carte, en mégaoctets par seconde
Network	Multiqueue	vide	Le nombre de files de la carte VirtIO. Recommandé : autant que de vCPU

À retenir — les options avancées

- La case **Advanced** ne change aucune valeur : elle montre des champs qui ont déjà un défaut correct.
- Les plus utiles en production : **Start at boot** et son ordre, **Backup** sur un disque, **CPU units**, **Disconnect**.
- Tout se modifie après la création : onglets **Options** et **Hardware** de la VM.

Ce support est la propriété de Dawan France.

4. Créer une VM KVM — Créer une VM en ligne de commande

4.2 Créer une VM en ligne de commande

La commande `qm` est l'interface CLI de Proxmox pour les VMs. Chaque paramètre correspond à un champ de l'assistant graphique.

C'est précisément l'intérêt de la faire une fois en ligne de commande. On voit d'un seul coup d'œil tout ce que l'assistant a rempli à votre place, valeurs par défaut comprises.

Pourquoi apprendre la CLI alors que l'interface web existe. Trois raisons, et aucune n'est une question de préférence :

Elle est scriptable. Créer trente VMs par l'assistant demande trente fois les mêmes clics, avec trente occasions de se tromper. Une boucle sur `qm create` ne se trompe pas.

Elle est reproductible et documentable. La commande complète tient dans un fichier, se relit, se versionne, se colle dans un ticket. Une suite de captures d'écran ne remplace pas cela.

Elle est parfois la seule disponible. Sur un nœud dont l'interface web ne répond plus, ou en dépannage via une console de secours, `qm` reste accessible.

Le VMID est le premier argument, et il n'est pas anodin. C'est l'identifiant unique de la VM **sur tout le cluster** : il ne peut pas être réutilisé tant que la VM existe.

Et c'est lui, pas le nom, qui sert de référence à toutes les autres commandes, aux sauvegardes et à l'API. Le nom, lui, peut changer librement.

D'où l'utilité d'une convention de numérotation dès le premier jour. Celle de ce support — les 100 pour les VM, les 200 pour les conteneurs, les 9000 pour les modèles — est appliquée dans tous les chapitres et tous les ateliers.

C'est elle qui permet au conteneur 200 du chapitre 1 et aux VM de ce chapitre de cohabiter sur le même nœud. Elle évite aussi les collisions le jour où plusieurs personnes créent des machines en parallèle.

Dawan
Ce support est la propriété de Dawan France.

Les valeurs par défaut ne sont pas neutres. Ce que la commande ne précise pas est choisi par Proxmox, et rarement au mieux. Sans `--cpu`, la ligne de commande retient `kvm64`, un processeur générique volontairement conservateur qui masque les jeux d'instructions modernes.

D'où le `x86-64-v2-AES` de l'exemple, qui expose notamment l'accélération matérielle du chiffrement. L'assistant graphique, lui, présélectionne `x86-64-v2-AES` depuis Proxmox VE 8 : une même VM créée des deux façons n'a donc pas le même processeur si l'on ne précise rien. `qm config <VMID>` affiche la configuration réellement retenue, valeurs implicites comprises : c'est la commande à passer juste après une création pour vérifier ce qu'on a obtenu.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Créer une VM Debian 13 en CLI
qm create 105 \
  --name vm-debian \
  --machine q35 \
  --memory 2048 \
  --cores 2 \
  --sockets 1 \
  --cpu cputype=x86-64-v2-AES \
  --net0 virtio,bridge=vmbr0 \
  --scsi0 local-lvm:20 \
  --scsihw virtio-scsi-single \
  --ide2 local:iso/debian-13.6.0-amd64-netinst.iso,media=cdrom \
  --boot 'order=scsi0;ide2' \
  --ostype l26 \
  --agent enabled=1

# Démarrer la VM
qm start 105
```

Résultat attendu : `qm create` et `qm start` ne répondent rien en cas de succès. La VM 105 apparaît dans le panneau de gauche avec un indicateur vert, et sa console affiche l'installateur Debian.

Ce support est la propriété de Dawan France.

L'ordre de démarrage `scsi0;ide2` place le disque **avant** le lecteur CD. Le premier démarrage trouve un disque vide, passe au CD et lance l'installation ; les suivants démarrent sur le système installé, sans qu'il faille penser à retirer l'ISO. Avec `order=ide2` seul, la VM relancerait l'installateur à chaque démarrage.

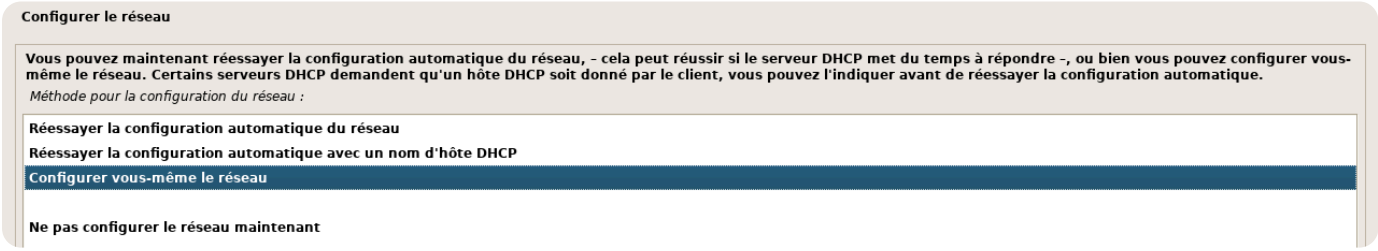
Et l'adresse IP ? Elle ne figure pas dans la commande, et c'est normal : `--net0 virtio,bridge=vmbr0` ne décrit que la **carte réseau** de la VM, branchée sur le pont `vmbr0`.

L'adresse appartient au système installé dans la VM : elle se saisit dans l'installateur Debian, à l'étape **Configurer le réseau**. `qm` ne sait fixer l'adresse d'une VM qu'au travers de **cloud-init** (`--ipconfig0 ip=…/24,gw=…`), présenté au §10. C'est ce que font les ateliers 7 et 8, avec une image cloud au lieu de cette ISO.

Avec l'ISO `debian-13.6.0-amd64-netinst.iso` (installation graphique, en français), sur un nœud du lab :

- **S'il n'y a pas de serveur DHCP** sur le réseau, l'installateur échoue à se configurer seul et propose la méthode de configuration.
- **S'il y en a un**, il prend une adresse sans rien demander et passe directement au nom de machine. Pour garder une adresse fixe, revenez en arrière (bouton **Revenir en arrière**, ou touche **Échap**) : le choix de la méthode apparaît.

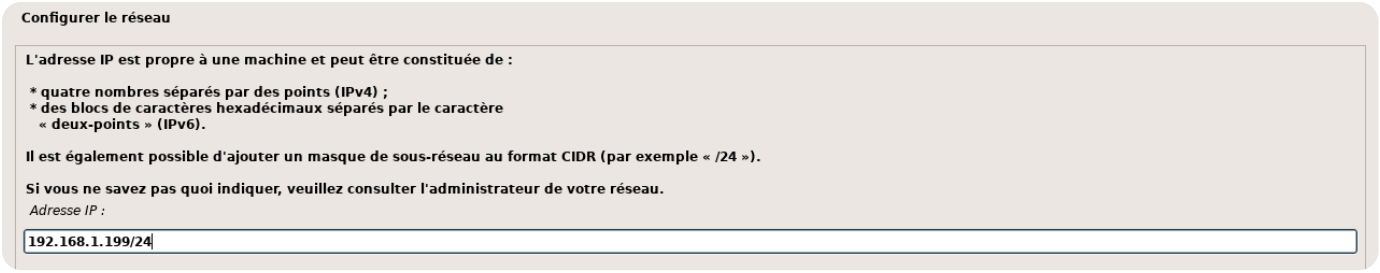
Choisissez **Configurer vous-même le réseau** :



Les deux premières lignes relancent le DHCP ; la troisième ouvre la saisie manuelle ; la dernière laisse la VM sans réseau, à configurer plus tard.

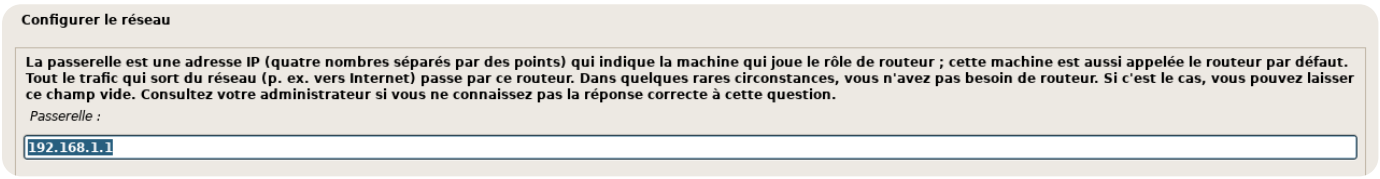
Saisissez ensuite l'adresse fixe du plan d'adressage, avec son masque au format CIDR : `10.<site>.<N>.54/24` pour la VM 105, la première VM des ateliers.

Ce support est la propriété de Dawan France.



L'écran le dit lui-même : le masque s'ajoute directement à l'adresse (`/24`). Sur la capture, l'adresse d'essai du lab d'exemple ; chez vous, celle de votre `vmbr0` , à partir de `.50` . Une valeur invalide est refusée par le message **Adresse IP mal formée**.

Viennent ensuite la **Passerelle**, puis les **Adresses des serveurs de noms** :



L'installateur propose d'office l'adresse en `.1` du réseau saisi, pour la passerelle comme pour le serveur de noms : vérifiez qu'elle correspond à votre réseau, puis **Continuer**. L'installation reprend au nom de machine.

> **Vérifiez l'adresse au premier démarrage.** Sur le lab, sur un réseau qui annonce de l'IPv6 : malgré la saisie manuelle, l'installateur a écrit `iface ens18 inet6 auto` dans `/etc/network/interfaces` , et la VM a démarré sans adresse IPv4. `ip -br a` dans la console le montre tout de suite ; l'atelier 2 (étape 3) donne la correction.

- 📖 Documentation Proxmox — les machines virtuelles dans le guide d'administration
- 📖 Documentation Proxmox — la commande qm

4.3 Pièges fréquents à la création d'une VM

Certaines erreurs reviennent systématiquement lors des premières créations de VM, aussi bien via l'assistant graphique qu'en CLI. Les connaître à l'avance évite de perdre du temps en diagnostic.

La VM ne démarre pas, `TASK ERROR: start failed` **.** VM ID déjà utilisé, ou pool LVM-thin saturé. Remède : `qm list` pour vérifier les VMID existants, `lvs` pour l'espace disponible.

Écran noir dans la console au boot. Ordre de boot incorrect (`--boot order=...`) ou ISO non attachée. Remède : `qm config <vmid> | grep boot` puis corriger avec `qm set` .

Ce support est la propriété de Dawan France.

Réseau invisible dans l'OS installé. Pilote VirtIO non supporté par un OS ancien. Remède : Basculer temporairement sur le modèle E1000 (carte réseau Intel émulée) le temps d'installer les pilotes VirtIO.

L'agent QEMU ne répond jamais. `qemu-guest-agent` non installé dans la VM, ou service non démarré. Remède : Cocher l'agent (`qm set <vmid> --agent enabled=1`), arrêter puis démarrer la VM, et `apt install qemu-guest-agent` dans le guest ; le service démarre alors tout seul.

Disque visible dans `lvs` mais absent dans la VM. Bus/Device mal choisi (SCSI vs IDE — ancien bus de disques — vs VirtIO Block) sans pilote correspondant dans l'OS. Remède : Vérifier la compatibilité du bus avec l'OS installé, privilégier VirtIO SCSI pour les OS modernes.

Face à une VM qui se comporte anormalement, le réflexe est le même que pour un serveur physique. Consulter d'abord les journaux (`journalctl` , `qm config` , `qm monitor`), avant de recréer la VM depuis zéro. Cette rigueur diagnostique est particulièrement précieuse en production, où recréer une VM n'est pas toujours une option instantanée.

À retenir : une VM se crée par l'assistant **Create VM** (onglets General, OS, System, Disks, CPU, Memory, Network) ou par `qm create` , dont chaque option reprend un champ de l'assistant. Les valeurs par défaut ne sont pas neutres : on choisit explicitement contrôleur, carte réseau et type de CPU.

> **▶ À pratiquer maintenant — Atelier 2**
>
> Créer la première VM de bout en bout : l'assistant onglet par onglet, l'installation de Debian depuis l'ISO, l'adresse fixe, la connexion.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

5. Accès console et cycle de vie d'une VM

La console d'une VM passe par VNC (*Virtual Network Computing*, protocole d'affichage à distance).

5.1 La console VNC et noVNC

Proxmox expose la console de chaque VM via **noVNC**, une implémentation VNC qui fonctionne dans le navigateur web sans plugin. Pour y accéder, cliquez sur la VM dans le panneau de gauche, puis sur **Console**. Un onglet s'ouvre avec l'écran de la VM.

La console VNC est particulièrement utile pendant l'installation de l'OS et pour diagnostiquer une VM qui ne répond plus au réseau. C'est votre accès de secours — indépendant de l'état du réseau de la VM.

Pour les VMs équipées de l'agent QEMU (`qemu-guest-agent`), Proxmox peut également obtenir des informations depuis l'intérieur de la VM (adresses IP, état des disques) sans passer par le réseau.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis l'hôte Proxmox : ouvrir un terminal série vers la VM (si configuré)
qm terminal 100

# Vérifier que l'agent QEMU répond
qm guest cmd 100 ping
```

Résultat attendu de `qm guest cmd 100 ping` : aucun affichage, et un code de retour `0` (`echo $?`). Le silence est la réponse positive : l'agent a répondu.

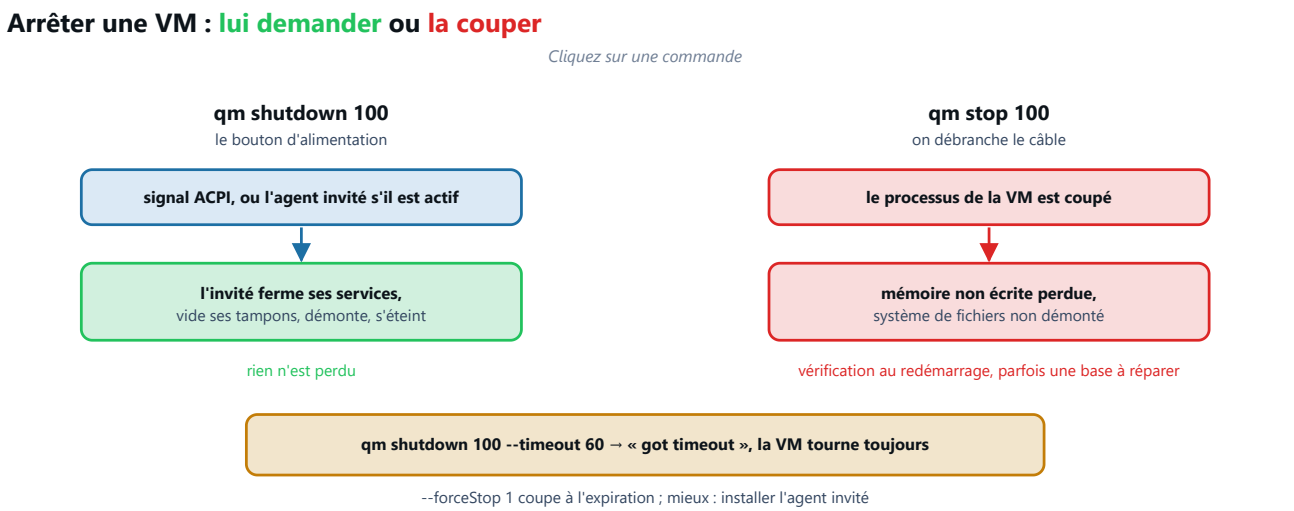
Si l'agent n'est pas installé, ou si le service `qemu-guest-agent` n'est pas démarré dans la VM, la commande échoue au bout de quelques secondes avec `QEMU guest agent is not running` . `qm agent` est un ancien nom de la même commande, toujours accepté.

5.2 Gérer le cycle de vie

Au-delà du démarrage initial, les commandes suivantes couvrent le cycle de vie complet d'une VM au quotidien.

Ce support est la propriété de Dawan France.

La distinction à ne pas manquer est celle entre `shutdown` et `stop`, parce qu'elle a des conséquences directes sur les données.



La distinction a des conséquences directes sur les données : shutdown demande, stop impose.
Mêmes boutons dans l'interface : Shutdown et Stop.

Ce que déclenche chacune des deux commandes, et le piège du `shutdown` ignoré. Cliquez sur une commande pour le détail.

`qm shutdown` envoie un signal **ACPI** à la VM — l'équivalent d'un appui sur le bouton d'alimentation d'une machine physique. C'est le système invité qui décide : il ferme ses services, vide ses tampons d'écriture sur disque, démonte ses systèmes de fichiers, puis s'éteint. Rien n'est perdu.

`qm stop` coupe l'alimentation du processus, sans prévenir personne. C'est l'équivalent de débrancher le câble.

Tout ce qui était en mémoire et pas encore écrit disparaît. Le système de fichiers reste marqué comme non démonté : d'où la vérification au redémarrage suivant, et parfois une base de données à réparer.

Le piège pratique : `qm shutdown` peut ne rien faire du tout. Si la VM n'a pas de gestionnaire ACPI — un système minimal, un invité figé — le signal est ignoré.

Quand l'agent invité est installé et activé, Proxmox lui passe la demande d'arrêt par son canal plutôt que par ACPI, ce qui est plus fiable. L'option `--timeout` borne l'attente, **mais ne force**

Ce support est la propriété de Dawan France.

rien à elle seule :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Arrêt propre, abandonné si la VM n'a pas répondu au bout de 60 secondes
qm shutdown 100 --timeout 60
```

Résultat attendu :

```
root@pxe211:~# qm shutdown 100 --timeout 60
VM quit/powerdown failed - got timeout
root@pxe211:~# qm status 100
status: running
```

Le message dit exactement ce qui s'est passé : la VM n'a pas répondu au signal ACPI dans le délai imparti et **tourne toujours**. Pour qu'elle soit coupée d'office à l'expiration du délai, il faut le demander explicitement : `qm shutdown 100 --timeout 60 --forceStop 1`. Mais une VM qui ignore l'arrêt propre signale surtout un agent absent : c'est à ce moment-là qu'on installe l'agent invité (§7.2), plutôt que de prendre l'habitude du `qm stop`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Démarrer une VM
qm start 100

# Arrêt propre (envoie un signal ACPI — comme appuyer sur le bouton power)
qm shutdown 100

# Forcer l'arrêt (équivalent débrancher le câble — à éviter sauf urgence)
qm stop 100

# Redémarrer
qm reboot 100

# Mettre en pause (suspend la VM, garde l'état en RAM)
qm suspend 100

# Voir l'état de toutes les VMs
qm list
```

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

```
# Voir la configuration complète d'une VM
qm config 100

# Modifier des paramètres : sur une VM allumée, ils restent « en attente »
# jusqu'au prochain démarrage, sauf si le hotplug mémoire/CPU a été activé
qm set 100 --memory 4096
qm set 100 --cores 4   # 4 cœurs, appliqué au prochain démarrage
```

Résultat attendu de `qm list` :

```
      VMID NAME                 STATUS     MEM(MB)    BOOTDISK(GB) PID
       100 debian-base          running       2048             32.00 12345
       101 debian-cloud         stopped       2048             20.00 0
```

Chaque VM est listée avec son VMID, son nom et son statut (`running` ou `stopped`). Suivent la RAM allouée, la taille du disque système, et le PID du processus QEMU (0 si arrêtée).

5.3 Dépannage et journaux

Quand une VM refuse de démarrer ou se bloque, les journaux Proxmox sont le premier réflexe.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Journaux du service pve-cluster (état général)
journalctl -u pve-cluster --since "1 hour ago"

# Tâches récentes de la VM 100 (démarrage, arrêt, migration...), en JSON lisible
pvesh get /nodes/$(hostname)/tasks --vmid 100 --limit 5 --output-format json-pretty

# Démarrer au premier plan affiche l'erreur QEMU complète, sans passer par un
journal
qm start 100

# Moniteur QEMU interactif (diagnostic avancé)
qm monitor 100
# Dans le moniteur : "info status", "info block", "quit"
```

Résultat attendu de la commande `pvesh` (extrait, une tâche) :

Ce support est la propriété de Dawan France.

```
[
   {
      "endtime" : 1781683203,
      "id" : "100",
      "node" : "pxe211",
      "starttime" : 1781683200,
      "status" : "OK",
      "type" : "qmstart",
      "upid" : "UPID:pxe211:00003A2B:0012F3C4:6A31D2C0:qmstart:100:root@pam:",
      "user" : "root@pam"
   }
]
```


`qm start 100` lancé à la main a un avantage sur le bouton de l'interface : l'erreur complète de QEMU s'affiche directement dans le terminal. Les trois messages les plus fréquents se reconnaissent vite.

`VM is locked (backup)` ou `(snapshot)` : une opération précédente a laissé un verrou, visible dans `qm config` par une ligne `lock:` ; si l'opération est réellement terminée, `qm unlock 100` le lève. `timeout waiting on systemd` : QEMU n'a pas démarré dans le délai, souvent faute de mémoire libre sur l'hôte. `no such logical volume` : le disque référencé dans la configuration n'existe plus — typiquement après une suppression manuelle dans LVM.

`qm monitor 100` ouvre enfin le moniteur de QEMU, pour interroger la VM en cours d'exécution. `info status` dit si elle tourne ou est en pause : une VM figée par un pool thin plein y apparaît `paused (io-error)`. `info block` liste ses disques et leur état.

Chaque tâche Proxmox (démarrage, arrêt, migration, backup) est enregistrée avec son type, son statut (`OK` , ou le message d'erreur), et ses horodatages Unix. L'`upid` est son identifiant complet : `pvenode task log <upid>` en affiche le journal détaillé, exactement celui que l'interface montre en double-cliquant sur la tâche dans le panneau du bas. C'est le premier endroit à consulter lors d'un dysfonctionnement.

> **Erreur fréquente :** une VM refuse de démarrer avec `TASK ERROR: start failed`. La cause est souvent un manque d'espace dans le pool LVM-thin, ou un conflit de VM ID. Vérifiez `lvs` et `qm list` en premier.

Ce support est la propriété de Dawan France.

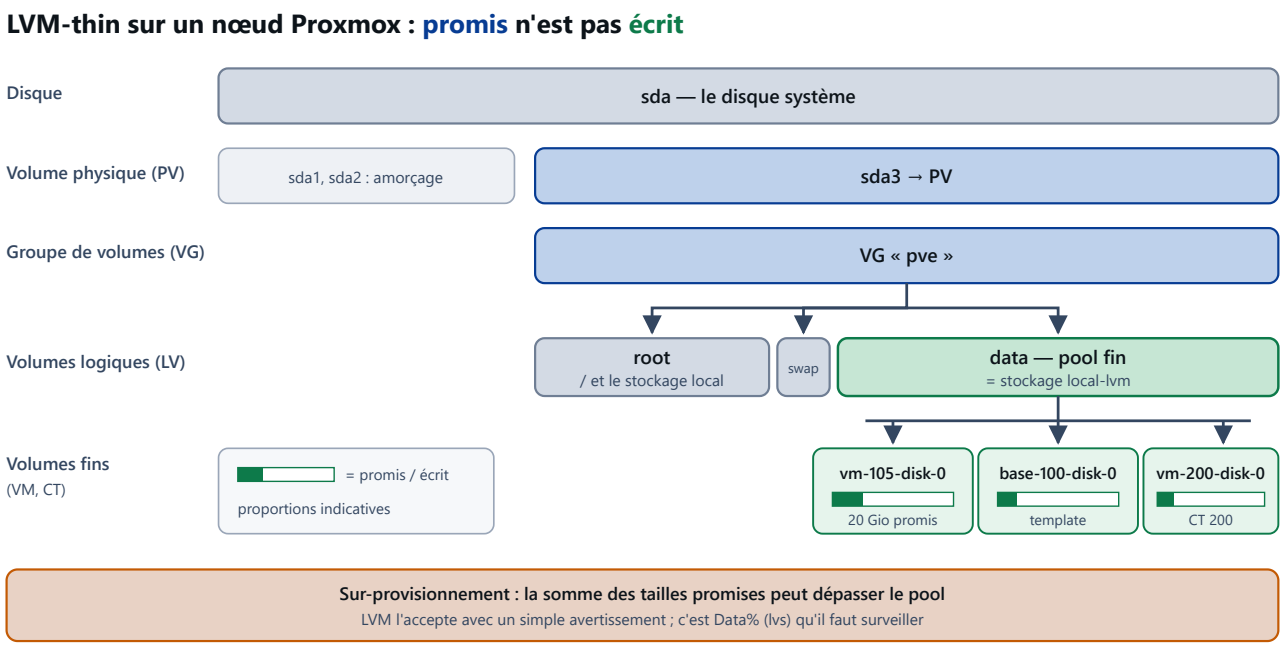
À retenir : la console **noVNC** reste l'accès de secours quand le réseau de la VM ne répond plus. `qm shutdown` demande un arrêt propre à l'invité ; `qm stop` coupe l'alimentation. En cas de problème : `qm config`, les journaux (`journalctl`), puis l'espace du pool.

> **▶ À pratiquer maintenant — Atelier 3**
>
> Piloter la VM : console, arrêt propre ou brutal, historique des tâches ; puis retirer l'ISO et la faire démarrer avec le nœud.

Ce support est la propriété de Dawan France.

6. Le stockage LVM-thin — pourquoi c'est intelligent

6.1 Le thin-provisioning expliqué



Survolez un élément pour lire son rôle et le résultat du lab.

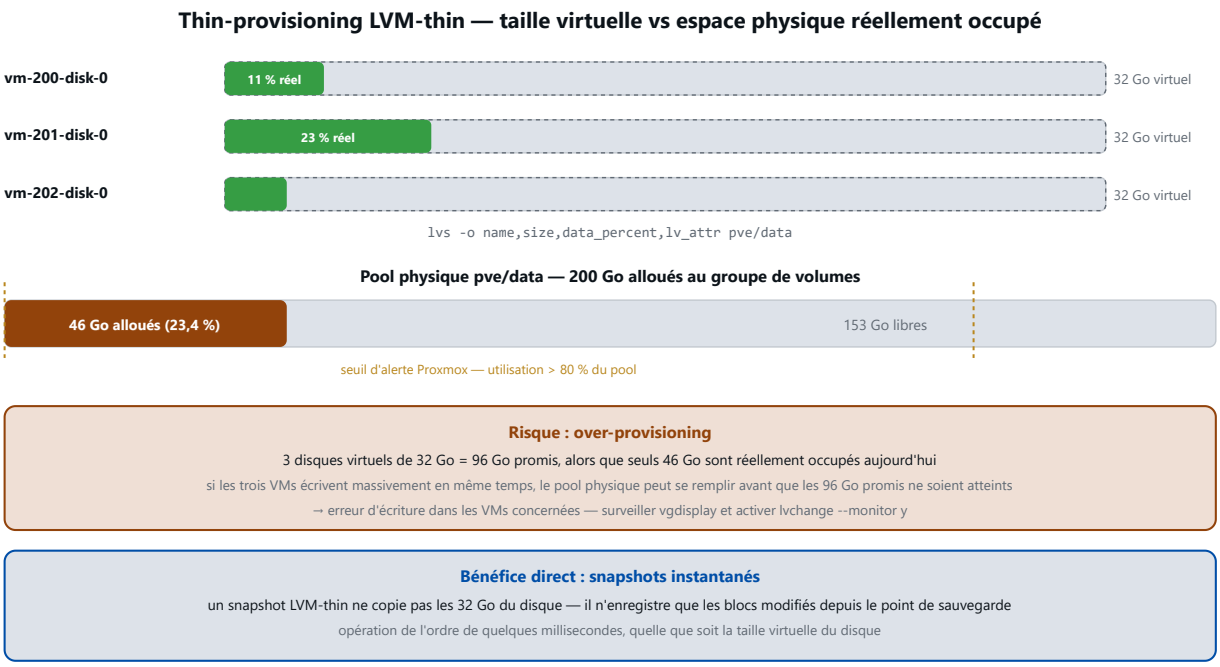
La pile LVM que l'installateur a posée sur votre nœud, du disque aux disques des VM. La partition `sda3` devient un volume physique, réuni dans le groupe de volumes `pve`.

Celui-ci est découpé en `root`, `swap` et `data`, le pool fin que Proxmox appelle `local-lvm`. Chaque disque de VM ou de conteneur y est un volume fin. Survolez un élément pour lire son rôle et le résultat du lab.

Quand vous créez un disque de 32 Go pour une VM dans `local-lvm`, Proxmox ne réserve pas immédiatement 32 Go d'espace physique. C'est le principe du **thin provisioning** — ou provisioning à la demande. Le volume LVM-thin n'occupe de l'espace réel que pour les données effectivement écrites.

Une VM fraîchement installée avec un OS de 3 Go n'utilisera que 3 à 4 Go sur le stockage physique, même si son disque virtuel est configuré à 32 Go. Cette différence est fondamentale pour l'utilisation de l'espace disque dans un atelier ou une infrastructure partagée.

Ce support est la propriété de Dawan France.



Le thin-provisioning rend également les **snapshots instantanés**. Plutôt que de copier 32 Go de données, un snapshot LVM-thin ne note que les différences entre l'état actuel et le point de sauvegarde — une opération qui prend quelques millisecondes.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Voir l'utilisation réelle du pool LVM-thin
lvs -o name,size,data_percent,lv_attr pve/data
```

Résultat attendu (nœud `pxe211`, Proxmox VE 9.2.21, en fin de chapitre 2) :

```
root@pxe211:~# lvs -o name,size,data_percent,lv_attr pve/data
  LV   LSize  Data%  Attr
  data 61.87g 64.62  twi-aotz--
```

La colonne `Data%` est celle qu'il faut surveiller, et elle n'est visible que là. `df -h` ne montre rien du pool thin : il affiche l'espace vu par chaque VM, non l'espace réellement consommé dans le pool.

Les attributs `twi-aotz--` se décodent : `t` pour *thin pool*, `w` pour accessible en écriture, `i` pour l'allocation héritée. Puis `a` pour actif, `o` pour ouvert, `z` pour la remise à zéro des blocs neufs.

Ce support est la propriété de Dawan France.

Un pool thin saturé met en pause toutes les VM qui s'y trouvent, d'un coup. Le sur-provisionnement autorise la somme des disques virtuels à dépasser la taille du pool ; tant que les VM n'écrivent pas tout, cela tient. Le jour où elles le font, LVM n'a plus de blocs à distribuer et gèle les écritures — les systèmes invités passent en lecture seule ou plantent.

Au-delà de **80 % de** `Data%`, il faut agir : étendre le pool, supprimer des snapshots, ou déplacer des disques. Une supervision de cette valeur est indispensable en production, et c'est la première valeur à confier à un outil de supervision.

Revenons à la définition de base : **allouer** un disque, c'est réserver de l'espace sur le stockage. Le *thin provisioning* réserve sur le papier, et n'occupe qu'au moment où l'on écrit. Les termes, du plus large au plus fin :

Terme	Ce que c'est	Dans le laboratoire
Groupe de volumes	L'espace disque physique géré par LVM	`pve`, créé à l'installation
Pool thin	Le volume qui distribue des blocs aux disques des VM, au fil des écritures	`data`, vu par Proxmox comme le stockage `local-lvm`
Disque thin	Le disque d'une VM, de taille promise	`vm-100-disk-0`
Taille virtuelle (`LSize`)	Ce que la VM croit avoir	20 Go pour la VM 105 (atelier 2)
Données écrites (`Data%`)	Les blocs occupés pour de vrai	La colonne `Data%` de `lvs`
Sur-provisionnement	La somme des disques promis dépasse la taille du pool	Possible sur `local-lvm`
Pool saturé	Plus de bloc libre : les écritures gèlent, les VM se figent	À éviter : agir au-delà de 80 %

Le sur-provisionnement tient tant que les VM n'écrivent pas toutes leur taille promise en même temps. Le jour où elles le font, le pool sature et fige toutes ses VM.

Où voir l'espace sur votre nœud :

Notion	Où la voir
Le pool et son remplissage	Le stockage **local-lvm** → **Summary** (*Usage*) ; `pvesm status`
La part réellement écrite, pool et disques	`lvs -o name,size,data_percent,lv_attr pve` : colonnes `LSize` et `Data%`

Ce support est la propriété de Dawan France.

Notion	Où la voir
La taille promise à une VM	La VM → **Hardware** → **Hard Disk** (`size=32G`)
Ce que `df -h` ne montre pas	`df -h` sur le nœud ignore le pool thin : seul `lvs` en donne le vrai remplissage

6.2 Surveiller et gérer l'espace

Le danger du thin-provisioning est le **over-provisioning**. La somme des disques virtuels dépasse l'espace physique disponible. Si toutes les VMs commencent à écrire intensivement, le pool peut se remplir.

La jauge du stockage passe à l'orange puis au rouge dans l'interface, mais **aucune alerte n'est envoyée**. Sans supervision, on découvre le problème quand les VM se figent en erreur d'écriture.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Vérifier l'espace disponible dans le groupe de volumes LVM
# (vgdisplay écrit « Free  PE » avec deux espaces : d'où le « + » du motif)
vgdisplay pve   # à lire : VG Size, Alloc PE, Free PE
```

Résultat attendu (les lignes qui comptent) :

```
root@pxe211:~# vgdisplay pve | grep -E 'VG Size|Alloc +PE|Free +PE'
  VG Size               <91.50 GiB
  Alloc PE / Size       23423 / <91.50 GiB
  Free  PE / Size       0 / 0
```

Ces trois lignes indiquent la taille totale du groupe de volumes (`VG Size`), l'espace alloué aux volumes existants, et l'espace libre restant.

Deux espaces à ne pas confondre. `Data%` (de `lvs`) mesure le remplissage du *thin pool* : c'est lui qui gèle les VM quand il atteint 100 %.

`Free PE` mesure ce qui reste **hors du pool** dans le groupe de volumes, disponible pour agrandir le pool par `lvextend` . Ici, 0 : l'installateur a donné tout le groupe au système et au pool. Pour agrandir ce pool, il faut d'abord agrandir le disque.

Ce support est la propriété de Dawan France.

> **Erreur fréquente :** créer trop de VMs avec des disques surdimensionnés, puis oublier de surveiller le taux d'utilisation du pool. Mettez en place une alerte à 70 % en production.

6.3 Formats de disque et migration entre stockages

Proxmox supporte plusieurs formats de disques virtuels. Le choix du format dépend du type de stockage et du cas d'usage.

Et il n'est pas toujours libre : **le type de stockage impose souvent le format**. C'est la première chose à comprendre.

Un stockage **bloc** — LVM-thin, ZFS, RBD Ceph — ne connaît que le format `raw`. Il gère lui-même les instantanés et le provisionnement fin au niveau du volume.

La VM n'a donc pas besoin d'un format qui les apporte. Un stockage **fichier** — un répertoire, du NFS — accepte au contraire `qcow2`, parce qu'il faut bien un fichier capable de porter seul ses métadonnées et ses snapshots.

D'où la conséquence pratique qui surprend : **sur LVM-thin, on ne perd rien à être en** `raw`. Les snapshots existent quand même, mais ils sont assurés par la couche de stockage, pas par le format du disque. Croire qu'il faut du `qcow2` pour avoir des snapshots est une erreur fréquente venue du monde du fichier.

Et c'est ce qui rend la **migration entre stockages** intéressante : déplacer un disque d'un stockage fichier vers un stockage bloc **convertit automatiquement** `qcow2` en `raw`, et inversement. Proxmox gère la conversion de façon transparente pendant le `Move Disk`, y compris à chaud sur une VM allumée.

Format	Description	Cas d'usage
raw	Format brut, pas de métadonnées. Performances maximales, accès direct aux blocs.	LVM-thin, ZFS — les snapshots sont gérés par la couche sous-jacente
qcow2	Format QEMU natif avec snapshots internes, compression, sparsification. Légèrement moins performant en IOPS (opérations d'entrée-sortie par seconde).	Stockages répertoire (NFS, `local`) et images cloud
vmdk	Format VMware. Utile pour les migrations depuis VMware. Non recommandé pour de nouvelles VMs.	Migration VMware → Proxmox uniquement

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Voir le format et l'occupation réelle d'un disque virtuel
qemu-img info /dev/pve/vm-100-disk-0
# → file format: raw
# → virtual size: 20 GiB
# → disk size: 20 GiB

qemu-img info /var/lib/vz/images/100/vm-100-disk-0.qcow2
# → file format: qcow2
# → virtual size: 20 GiB
# → disk size: 3.2 GiB   ← espace réellement utilisé (sparsification)
```

Résultat attendu : les valeurs notées en commentaire (`→`) — `raw` pour le volume LVM, `qcow2` pour le fichier, chacun avec sa taille virtuelle et sa taille sur disque.

`qemu-img info` distingue deux tailles, et c'est tout l'intérêt de la commande. La **taille virtuelle** est celle que voit la VM : le disque de 20 Gio annoncé à son système. La **taille sur disque** est ce que le fichier occupe réellement sur le stockage.

Pour un fichier `qcow2`, l'écart est le provisionnement fin en action : seuls les blocs écrits existent. Pour un volume LVM brut, `qemu-img` ne sait pas ce que le pool thin a réellement alloué — c'est `lvs` et sa colonne `Data%` qu'il faut lire, comme au §6.1.

Le déplacement d'un disque, enfin, se fait sans arrêter la VM :

Dans l'interface : la VM → **Hardware** → le disque → **Disk Action** → **Move Storage**. Le stockage cible doit accepter le contenu **Disk image**. Sur un nœud neuf, seul `local-lvm` l'accepte : pour l'essai ci-dessous, un stockage de type répertoire, `disques-dir`, a été déclaré avec ce contenu.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Vers local : refusé, local n'accepte pas les disques
qm disk move 105 scsi1 local --format qcow2 --delete 1

# Vers un répertoire qui accepte les disques, VM allumée, au format qcow2
qm disk move 105 scsi1 disques-dir --format qcow2
qm config 105   # le disque déplacé, et l'ancien resté en unused ; à lire : scsi1, unused
```

Dawan Ce support est la propriété de Dawan France.

Résultat attendu (les lignes qui comptent) :

```
400 Parameter verification failed.
storage: storage 'local' does not support vm images

copying volume 'local-lvm:vm-105-disk-1' from current storage 'local-lvm' to target
storage 'disques-dir'
moving disk with snapshots, snapshots will not be moved!
create full clone of drive scsi1 (local-lvm:vm-105-disk-1)
Formatting '/var/lib/disques-dir/images/105/vm-105-disk-0.qcow2', fmt=qcow2
cluster_size=65536 extended_l2=off preallocation=metadata compression_type=zlib
size=6442450944 lazy_refcounts=off refcount_bits=16
allocated target volume 'disques-dir:105/vm-105-disk-0.qcow2'
drive mirror is starting for drive-scsi1
mirror-scsi1: transferred 2.3 GiB of 6.0 GiB (38.02%) in 1s
mirror-scsi1: transferred 6.0 GiB of 6.0 GiB (100.00%) in 2s, ready
all 'mirror' jobs are ready
mirror-scsi1: Completing block job...
mirror-scsi1: Completed successfully.
mirror-scsi1: mirror-job finished
adding source volume 'local-lvm:vm-105-disk-1' as 'unused0' to config

scsi1: disques-dir:105/vm-105-disk-0.qcow2,discard=on,iothread=1,size=6G
unused0: local-lvm:vm-105-disk-1
```

La sortie raconte la méthode. Sur une VM allumée, Proxmox crée le nouveau disque, puis lance un **miroir**.

QEMU recopie les blocs en arrière-plan, et répercute sur la copie les écritures qui arrivent pendant ce temps. Quand les deux côtés sont identiques (`ready`), il bascule la VM sur le nouveau disque, sans interruption.

Trois points à retenir :

- **L'ancien disque reste**, détaché, en `unused0` : il occupe toujours sa place. On le supprime ensuite (**Hardware** → `unused0` → **Remove**).
- `--delete 1` supprimerait la source dans la foulée. Mais il est refusé sur une VM qui a des snapshots : `you can't move a disk with snapshots and delete the source` (sur la VM 105, qui porte le snapshot `avant-essai`).

Dawan
Ce support est la propriété de Dawan France.

- **Les snapshots ne suivent pas le disque** : `moving disk with snapshots, snapshots will not be moved!`.

`qm move-disk`, l'ancien nom de la commande, fonctionne encore.

6.4 Redimensionner un disque à chaud

Un besoin très courant en exploitation est d'agrandir le disque d'une VM déjà en production, sans devoir l'éteindre ni recréer son volume. Proxmox permet ce redimensionnement à chaud grâce au support du hot-plug de la plupart des bus de disque modernes (VirtIO SCSI, VirtIO Block).

L'opération se déroule en deux temps. Proxmox agrandit d'abord le volume logique côté hyperviseur : LVM-thin, ZFS ou fichier qcow2 selon le stockage. Puis il informe la VM en cours d'exécution que la nouvelle taille est disponible.

C'est ensuite au système de fichiers **à l'intérieur** de la VM d'utiliser cet espace. Proxmox ne redimensionne jamais la partition ni le système de fichiers du guest : seulement le disque virtuel vu depuis l'hyperviseur.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Agrandir le disque scsi0 de la VM 100 de 10 Go supplémentaires
qm resize 100 scsi0 +10G

# Vérifier la nouvelle taille côté hyperviseur
qm config 100 | grep scsi0
```

Résultat attendu de `qm resize 100 scsi0 +10G` :

```
  Size of logical volume pve/vm-100-disk-0 changed from 32.00 GiB (8192 extents) to 42.00 GiB (10752 extents).
  Logical volume pve/vm-100-disk-0 successfully resized.
```

Le `+` compte : `+10G` **ajoute** 10 Gio, alors que `10G` sans signe demanderait une taille finale de 10 Gio — plus petite que l'actuelle, ce que Proxmox refuse (`shrinking disks is not supported`).

Ce support est la propriété de Dawan France.

Une fois le disque agrandi côté hyperviseur, il reste à agrandir la partition et le système de fichiers depuis l'intérieur de la VM. Sur une distribution Linux moderne utilisant `growpart` et un système de fichiers ext4 ou XFS, la procédure est directe :

Dans la VM (sa **Console**, ou en SSH) :

```
# Depuis l'intérieur de la VM Debian
growpart /dev/sda 1          # agrandit la partition 1 sur le disque /dev/sda
resize2fs /dev/sda1          # agrandit le système de fichiers ext4 (xfs_growfs pour
XFS)

# Vérifier l'espace disponible après agrandissement
df -h /
```

Résultat attendu :

```
CHANGED: partition=1 start=2048 old: size=67106816 end=67108863 new: size=88078303
end=88080350
resize2fs 1.47.2 (1-Jan-2025)
Filesystem at /dev/sda1 is mounted on /; on-line resizing required
The filesystem on /dev/sda1 is now 11009787 (4k) blocks long.

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        42G  2.1G   38G   6% /
```

Les trois étapes se lisent dans l'ordre. `growpart` déplace la **fin** de la partition (`CHANGED` , avec l'ancienne et la nouvelle taille en secteurs). `resize2fs` étend le système de fichiers **à chaud** (`on-line resizing`).

`df` confirme la nouvelle taille vue par le système. Une partition agrandie sans `resize2fs` ne sert à rien : `df` afficherait toujours l'ancienne taille.

> **Sur une Debian installée depuis l'ISO, `growpart /dev/sda 1` échoue souvent.** Le partitionnement assisté place la racine en `sda1` , puis une partition étendue `sda2` qui contient le swap `sda5` — **après** la racine. La partition 1 est donc coincée : l'espace ajouté se trouve au-delà du swap, et `growpart` répond `NOCHANGE: partition 1 ... could only be grown by 0` .

Ce support est la propriété de Dawan France.

> `lsblk` montre la situation d'un coup d'œil. Il faut alors désactiver le swap (`swapoff -a`), supprimer puis recréer les partitions 2 et 5 en fin de disque, ou tout simplement ajouter un **second disque** plutôt qu'agrandir le premier. Les images cloud, elles, placent la racine en dernier et ne connaissent pas ce problème.

Attention : `growpart` n'est pas installé par défaut. Il vient du paquet `cloud-guest-utils`, que ni l'ISO netinst ni la tâche « utilitaires usuels du système » n'installent. Sur une VM installée depuis une ISO, il faut donc commencer par :

Dans la VM (sa **Console**, ou en SSH) :

```
apt update                                  # la liste des paquets
apt install -y cloud-guest-utils            # installer growpart (paquet cloud-guest-utils)
which growpart                              # où il est installé
```

Résultat attendu (la fin) : `/usr/bin/growpart`.

Sur les **images cloud-init**, en revanche, `cloud-guest-utils` est préinstallé : ces images sont conçues pour s'adapter au disque qu'on leur attribue. L'opération y est transparente, ce qui explique que beaucoup de tutoriels omettent cette étape. La commande `qm resize` ne permet que d'**agrandir** un disque.

Il n'existe volontairement aucune commande pour réduire un disque virtuel à chaud. Cette opération risquerait de tronquer des données déjà écrites en fin de volume. Pour réduire un disque, il faut passer par une réinstallation ou une migration vers un nouveau disque plus petit après avoir vérifié que l'espace occupé le permet.

> **▶ À pratiquer maintenant — Atelier 4**
>
> Mesurer le provisionnement fin, ajouter un second disque, puis l'agrandir sans arrêter la VM.

Ce support est la propriété de Dawan France.

7. VirtIO : réseau, cache disque, mémoire et agent QEMU

7.1 Le modèle réseau des VMs

Le réseau d'une VM Proxmox passe par le même bridge `vmbr0` que les conteneurs LXC. La différence est dans le pilote : pour une VM, Proxmox présente un périphérique réseau virtuel, et c'est le driver à l'intérieur de la VM qui détermine les performances.

VirtIO est le standard de paravirtualisation pour les IO (spécification OASIS — organisme de normalisation). Avec un pilote émulé (e1000, RTL8139 — carte Realtek émulée), la VM écrit dans les registres d'un périphérique fictif. QEMU intercepte chaque accès → simule le comportement → extrait le paquet : des dizaines d'intercepts pour chaque IO.

Avec VirtIO, la VM et l'hyperviseur communiquent directement via une mémoire partagée en **ring buffer** — sans émulation de périphérique matériel. La VM "sait" qu'elle est virtualisée et utilise un pilote optimisé pour ça.

Le résultat est mesurable. Mesuré sur un nœud du lab, entre le nœud et une VM Debian 13, `iperf3` a donné :

Carte de la VM	Nœud → VM	VM → nœud
`virtio`	67,0 Gbit/s	74,0 Gbit/s
`e1000` (émulée)	6,48 Gbit/s	1,91 Gbit/s

Dix fois moins dans un sens, près de quarante fois moins dans l'autre, pour la même VM sur le même matériel. Les valeurs absolues dépendent du processeur ; l'écart, lui, se retrouve partout. C'est pourquoi toutes les VM de cette formation utilisent `virtio`.

Les pilotes émulés (Intel E1000, RTL8139) simulent un vrai matériel. Utiles pour les vieux OS (Windows XP — sorti en 2001 —, FreeBSD sans pilotes VirtIO) ou des cas particuliers, déconseillés en production pour des flux importants.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Ajouter une seconde interface réseau à la VM 100
qm set 100 --net1 virtio,bridge=vmbr0

# Vérifier la configuration réseau de la VM
qm config 100 | grep net
```

Résultat attendu de `qm config 100 | grep net` :

```
net0: virtio=AA:BB:CC:DD:EE:FF,bridge=vmbr0
net1: virtio=AA:BB:CC:DD:EE:11,bridge=vmbr0
```

Deux interfaces réseau sont maintenant configurées. Chaque interface reçoit une adresse MAC (*Media Access Control*, adresse matérielle de la carte), générée aléatoirement par Proxmox. Elle prend le préfixe `BC:24:11` que l'éditeur réserve à cet usage.

L'ajout se fait à chaud : le *hotplug* réseau est actif par défaut (option `hotplug` de la VM, qui vaut `network,disk,usb`), la carte apparaît aussitôt dans le système invité. Seuls les changements que le hotplug ne couvre pas, comme le modèle de la carte, restent « en attente » jusqu'au prochain démarrage. L'interface les affiche alors en rouge dans l'onglet **Hardware**.

- 📖 Documentation Proxmox — Network Configuration
- 📖 Configuration réseau dans Proxmox VE — Documentation Resinfo

7.2 QEMU Guest Agent : intégration entre la VM et l'hyperviseur

Trois fonctions justifient à elles seules d'activer l'agent sur toutes les machines, et elles tombent régulièrement en question d'examen comme en exploitation.

L'arrêt propre. Sans agent, le bouton « Arrêter » de Proxmox envoie un signal ACPI, que l'invité peut ignorer.

Un serveur Windows qui attend une confirmation reste allumé ; le délai d'attente finit par provoquer un arrêt brutal. Avec l'agent, Proxmox demande l'extinction au système lui-même, comme le ferait un administrateur connecté.

L'adresse IP dans l'interface. La colonne d'adresse d'une VM dans Proxmox ne vient pas du réseau : elle vient de l'agent, qui la lit dans l'invité et la remonte. Une VM sans agent affiche donc un champ vide, même parfaitement joignable — et c'est la première question posée en formation devant l'interface.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

La cohérence des sauvegardes, décrite plus bas : c'est la plus importante des trois, car elle décide si une sauvegarde est restaurable ou seulement rassurante.

L'agent ne coûte rien : un paquet dans l'invité, une case à cocher côté hyperviseur. Ne pas l'installer est un oubli, jamais un choix.

L'agent QEMU est un daemon qui tourne à l'intérieur de la VM et communique avec l'hyperviseur via un canal VirtIO dédié. Il permet à Proxmox d'effectuer des opérations coordonnées avec le système d'exploitation de la VM.

Comment l'agent parle à Proxmox. Activer l'option `agent` ajoute à la VM un petit périphérique série VirtIO, que le système invité voit comme `/dev/virtio-ports/org.qemu.guest_agent.0`.

Le démon `qemu-guest-agent` écoute sur ce canal, et Proxmox lui envoie des commandes JSON — sans réseau, sans SSH, sans mot de passe. Deux conséquences pratiques.

L'agent fonctionne même si la carte réseau de la VM est mal configurée. Et **l'option ne prend effet qu'après un arrêt puis un démarrage complets**, puisqu'il faut un nouveau périphérique : un `reboot` depuis l'invité ne suffit pas.

Sans agent, un snapshot Proxmox capture l'état de la RAM et du disque sans garantie de cohérence des données en vol. Une transaction de base de données à moitié écrite peut se retrouver dans le snapshot. Avec l'agent, Proxmox demande au système d'exploitation de **geler les écritures** (file system freeze) avant de créer le snapshot, puis de les reprendre — le snapshot est applicativement cohérent.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# 1. Sur l'hôte Proxmox : ajouter le canal de l'agent, puis arrêter et redémarrer la VM
#    (un reboot depuis l'invité ne crée pas le nouveau périphérique)
qm set 100 --agent enabled=1
qm shutdown 100                 # arrêt par Proxmox
qm start 100                    # puis démarrage par Proxmox

# 2. Dans la VM : installer l'agent et le lancer tout de suite ;
#    aux démarrages suivants, il démarrera seul, déclenché par son périphérique
apt install -y qemu-guest-agent
systemctl start qemu-guest-agent   # le lancer tout de suite
```

Ce support est la propriété de Dawan France.

```
# Interroger la VM via l'agent sans SSH — obtenir les IPs de toutes les interfaces
qm guest cmd 100 network-get-interfaces
```

Résultat attendu (extrait JSON) :


```
[
   {
      "hardware-address" : "bc:24:11:5e:3a:91",
      "ip-addresses" : [
         {
            "ip-address" : "10.44.1.55",
            "ip-address-type" : "ipv4",
            "prefix" : 24
         }
      ],
      "name" : "eth0"
   }
]
```


L'agent répond pour chaque interface de la VM ; la boucle `lo` figure aussi dans la réponse complète. Il donne son adresse MAC, et ses adresses IP telles que **le système invité** les voit.

C'est la seule façon, pour Proxmox, de connaître l'adresse d'une VM : le bridge ne fait que transporter des trames. Cette information s'affiche aussi dans l'interface web : **VM** → **Summary** → **IPs**, à condition que l'agent soit actif.

> **▶ À pratiquer maintenant — Atelier 5**
>
> Vérifier ce que l'agent apporte à la VM installée depuis l'ISO : adresse IP, système, état des fichiers, commandes sans SSH.

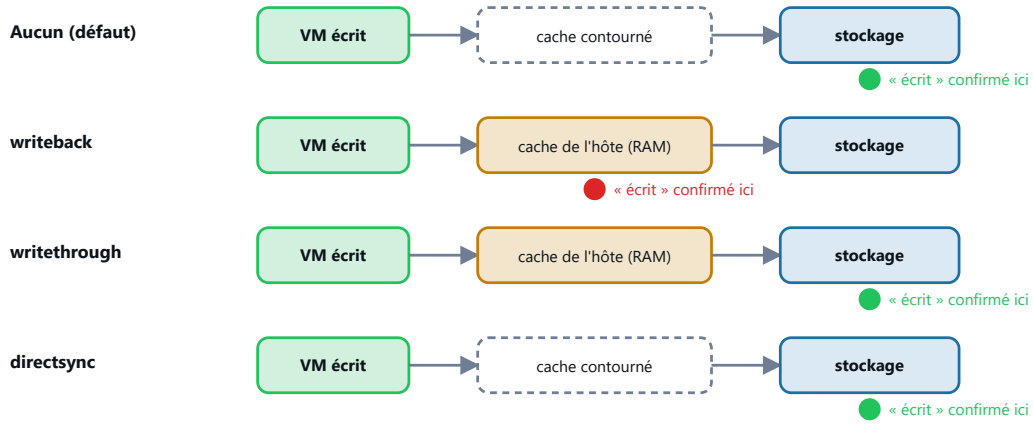
7.3 Le cache disque : un choix de risque, selon le stockage

Le mode de cache d'un disque décide **à quel moment** une écriture de la VM est considérée comme faite. Il se règle disque par disque (**Hardware** → **Hard Disk** → **Edit** → **Cache**, ou l'option `cache=` de `scsi0`).

Ce support est la propriété de Dawan France.

Le cache disque décide quand une écriture est « faite »

Cliquez sur un mode



Réglage disque par disque : Hardware → Hard Disk → Edit → Cache (option cache= de scsi0).
Règle : none partout par défaut ; jamais writeback sur ZFS, qui a déjà son cache (l'ARC).

Le trajet d'une écriture dans chacun des quatre modes, et l'endroit où la VM la croit terminée. Seul `writeback` la confirme avant le stockage : c'est son gain et son risque. Cliquez sur un mode pour le détail et les mesures du lab.

Mode	Comportement	Quand l'utiliser
Aucun (`Default (No cache)`, défaut)	Les écritures contournent le cache de page de l'hôte et partent vers le stockage	Le choix sûr, partout ; indispensable sur un stockage **partagé** (Ceph, iSCSI), où un cache local serait incohérent lors d'une migration
`writeback`	L'écriture est confirmée dès qu'elle est dans la mémoire de l'hôte	Gain réel en écriture ; perte des écritures récentes si l'hôte tombe brutalement — seulement avec une alimentation secourue et en connaissance de cause
`writethrough`	Les lectures profitent du cache de l'hôte, les écritures vont au disque	Rarement utile
`directsync`	Aucun cache, et chaque écriture est synchronisée	Charges qui exigent une écriture garantie à chaque opération

La règle à retenir : `none` **partout par défaut**. **Jamais** `writeback` **sur ZFS**, qui a déjà son propre cache en mémoire (l'ARC, *Adaptive Replacement Cache*) : deux caches empilés coûtent de la mémoire sans rien apporter.

Ce support est la propriété de Dawan France.

Sur Ceph, `writeback` s'appuie sur le cache client de `librbd` et reste un arbitrage performance contre risque. Deux autres options accompagnent le cache sur **VirtIO SCSI single**.

`iothread=1` donne au disque son propre fil d'entrées-sorties. `discard=on` transmet au stockage les blocs libérés par la VM ; il n'est utile que sur un stockage à provisionnement fin (LVM-thin, ZFS, Ceph).

Sur le lab, une écriture aléatoire de blocs de 4 Ko est passée de 45 800 IOPS en cache par défaut à 85 400 IOPS en `writeback`. Le gain est réel ; la garantie de l'écriture, elle, a disparu.

7.4 La mémoire : le ballooning

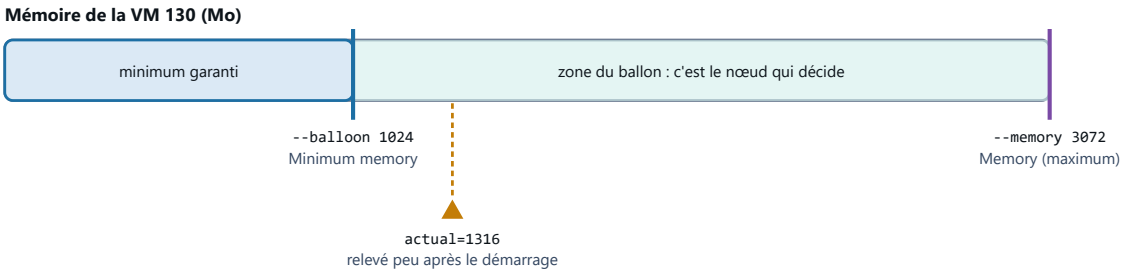
Le pilote **VirtIO balloon** permet au nœud de reprendre de la mémoire à une VM et de la lui rendre, sans l'arrêter. Deux valeurs le règlent, dans **Hardware** → **Memory** (options avancées) :

Réglage	Option `qm`	Rôle
Memory	`--memory`	Le **maximum** que la VM peut recevoir
Minimum memory	`--balloon`	Le **minimum garanti** ; `0` désactive le ballon

Entre les deux, c'est le nœud qui décide. Tant que son occupation mémoire reste sous sa cible (80 % par défaut), Proxmox **ajoute** de la mémoire aux VM jusqu'à leur maximum. Quand l'hôte manque de RAM, il la **reprend**, sans descendre sous le minimum ; dans la VM, `free` la voit disparaître.

Le ballon : entre un minimum garanti et un maximum

Cliquez sur le minimum, le maximum ou la valeur mesurée



Pilote VirtIO balloon : le nœud reprend de la mémoire à une VM, ou lui en rend, sans l'arrêter.
Sous sa cible d'occupation (80 % par défaut), il ajoute jusqu'au maximum ; s'il manque de RAM, il reprend,
sans descendre sous le minimum. Réglage : Hardware → Memory (options avancées).

Ce support est la propriété de Dawan France.

Les deux bornes d'une VM réglée entre 1 et 3 Go (`qm set <id> --memory 3072 --balloon 1024`), et la mémoire qu'elle avait réellement peu après le démarrage. Cliquez sur une borne pour son explication.

> **Pas de ballooning pour les services qui dimensionnent leur mémoire au démarrage.** Une base de données ou un cache (PostgreSQL, Redis) calcule ses tampons sur la mémoire qu'il voit ; la lui reprendre en cours de route dégrade ses performances. Pour eux : `--balloon 0`, ou un minimum égal au maximum.

À retenir : VirtIO remplace le matériel émulé par des pilotes paravirtualisés, bien plus rapides (réseau, disque). L'**agent QEMU** donne l'IP, un arrêt propre et des snapshots cohérents.

Le **cache disque** est un choix de risque (`none` par défaut). Le **ballooning** reprend de la mémoire à une VM entre un minimum et un maximum.

Ce support est la propriété de Dawan France.

8. Snapshots et rollback

8.1 Snapshots RAM vs snapshots disque seul

Un snapshot Proxmox capture l'état d'une VM à un instant T. Il est composé de deux parties :

- **L'état du disque** : les données écrites sur le volume LVM-thin ou ZFS au moment du snapshot
- **L'état de la RAM** (facultatif) : le contenu de la mémoire vive — permet de reprendre exactement où la VM en était

Capturer la RAM permet un rollback parfait (même les processus en cours reprennent), mais le snapshot prend plus de temps et consomme autant d'espace que la RAM de la VM.

Via l'interface graphique :

VM → Snapshots → Take Snapshot

Champ	Valeur
Name	`avant-mise-a-jour`
Description	`état stable avant apt upgrade`
Include RAM	décoché (plus rapide)

Via CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Snapshot sans RAM (rapide — VM peut rester allumée)
qm snapshot 100 avant-upgrade --description "état stable avant apt upgrade"

# Snapshot avec RAM (VM figée quelques secondes pendant la capture)
qm snapshot 100 avec-ram --vmstate 1

# Lister les snapshots existants
qm listsnapshot 100

# Revenir à un snapshot : une VM allumée est arrêtée d'office
qm rollback 100 avant-upgrade
```

Ce support est la propriété de Dawan France.

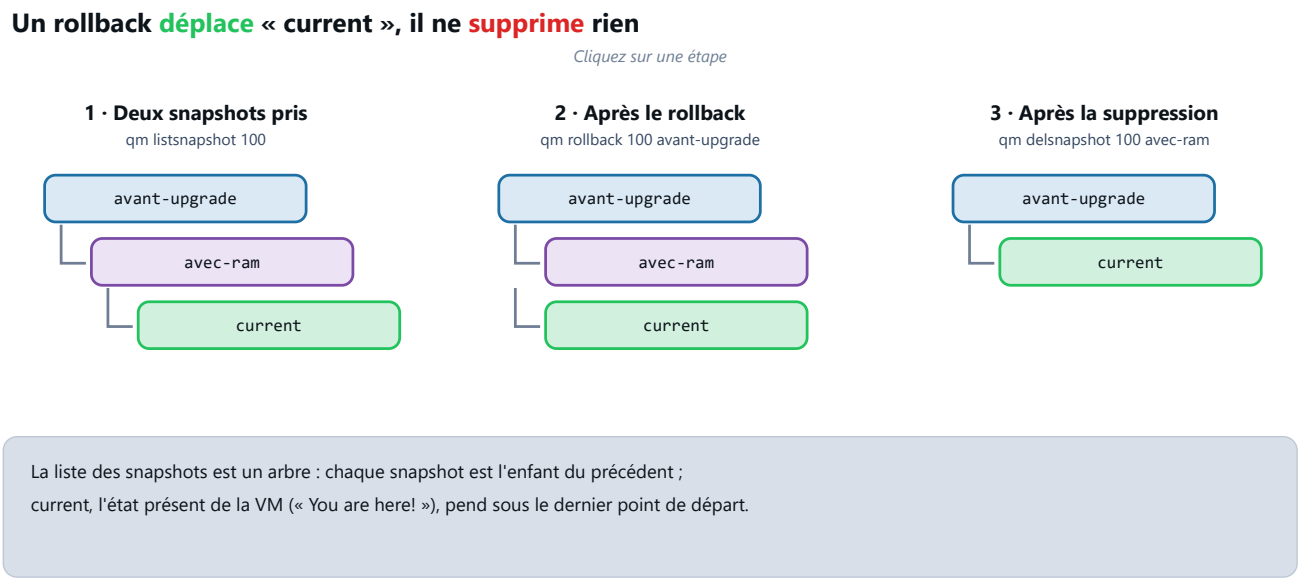
```
# Supprimer un snapshot devenu inutile
qm delsnapshot 100 avant-upgrade
```

Résultat attendu de `qm listsnapshot` (avant le rollback et la suppression), sur la VM 101 du lab, qui portait déjà un snapshot `avant-mise-a-jour` pris lors d'un essai précédent :

```
`-> avant-mise-a-jour          2026-09-25 10:17:35    VM propre avant mise à jour
 `-> avant-upgrade             2026-09-29 13:51:45    état stable avant apt
upgrade
  `-> avec-ram                 2026-09-29 13:51:46    no-description
   `-> current                                       You are here!
```

Chaque prise de snapshot sur `local-lvm` affiche aussi un avertissement de LVM : `WARNING: Sum of all thin volume sizes (<158.51 GiB) exceeds the size of thin pool pve/data`. Ce n'est pas une erreur : la somme des tailles **promises** aux volumes dépasse la place réelle du pool. C'est le principe du provisionnement fin, et la raison de surveiller son remplissage.

La liste est un **arbre** : chaque snapshot est l'enfant du précédent, et `current` — l'état présent de la VM — pend sous le dernier pris, avec la mention `You are here!`. Après `qm rollback 100 avant-upgrade`, `current` remonte sous `avant-upgrade` : le snapshot `avec-ram` existe toujours, mais sur une autre branche. Un rollback ne supprime donc rien ; c'est `qm delsnapshot` qui libère la place.



La liste des snapshots est un arbre : chaque snapshot est l'enfant du précédent ;
current, l'état présent de la VM (« You are here! »), pend sous le dernier point de départ.

Ce support est la propriété de Dawan France.

Trois états successifs de l'arbre de la VM 100. Le rollback déplace `current` sous `avant-upgrade` sans rien effacer ; seul `qm delsnapshot` supprime un snapshot et libère la place. Cliquez sur une étape pour le détail.

Le rollback d'un snapshot **avec RAM** a une particularité : la VM repart allumée, exactement là où elle en était, processus compris. Sans RAM, elle repart éteinte, comme après une coupure de courant à l'instant du snapshot.

- 📖 Documentation Proxmox — les snapshots de VM
- 📖 Snapshots de VM dans Proxmox — Technonagib

> **Piège courant :** les snapshots LVM-thin s'accumulent silencieusement et consomment de l'espace dans le pool. Une VM avec 10 snapshots peut occuper deux fois son espace nominal. Purgez régulièrement les anciens snapshots et surveillez le taux d'utilisation du pool avec `lvs`.

À retenir : un **snapshot** fige l'état d'une VM pour y revenir (avec ou sans sa mémoire vive) ; il s'accumule dans le pool et n'est pas une sauvegarde.

Ce support est la propriété de Dawan France.

9. Sauvegardes locales avec vzdump

9.1 La sauvegarde vzdump — fonctionnement

`vzdump` est l'outil de sauvegarde intégré à Proxmox. Il crée une archive complète d'une VM (configuration + disque) sous forme d'un fichier `.vma.zst` (VM Archive, compressé).

Cette archive peut être restaurée sur n'importe quel Proxmox compatible. Trois modes existent — `stop`, `suspend` et `snapshot` — détaillés au §9.2 ; le mode **snapshot** est le plus utilisé en production, parce que la VM continue de tourner pendant la sauvegarde.

Via l'interface graphique :

VM → Backup → Backup Now

Champ	Valeur
Storage	`local`
Mode	`Snapshot`
Compression	`ZSTD`
Note	`sauvegarde avant migration`

9.2 Créer et restaurer une sauvegarde

Le bouton **Backup Now** lance `vzdump` ; la commande s'utilise aussi directement, pour une sauvegarde ponctuelle ou scriptée, et `qmrestore` restaure depuis une archive existante.

Le paramètre décisif est le **mode**, parce qu'il arbitre entre cohérence des données et disponibilité de la VM :

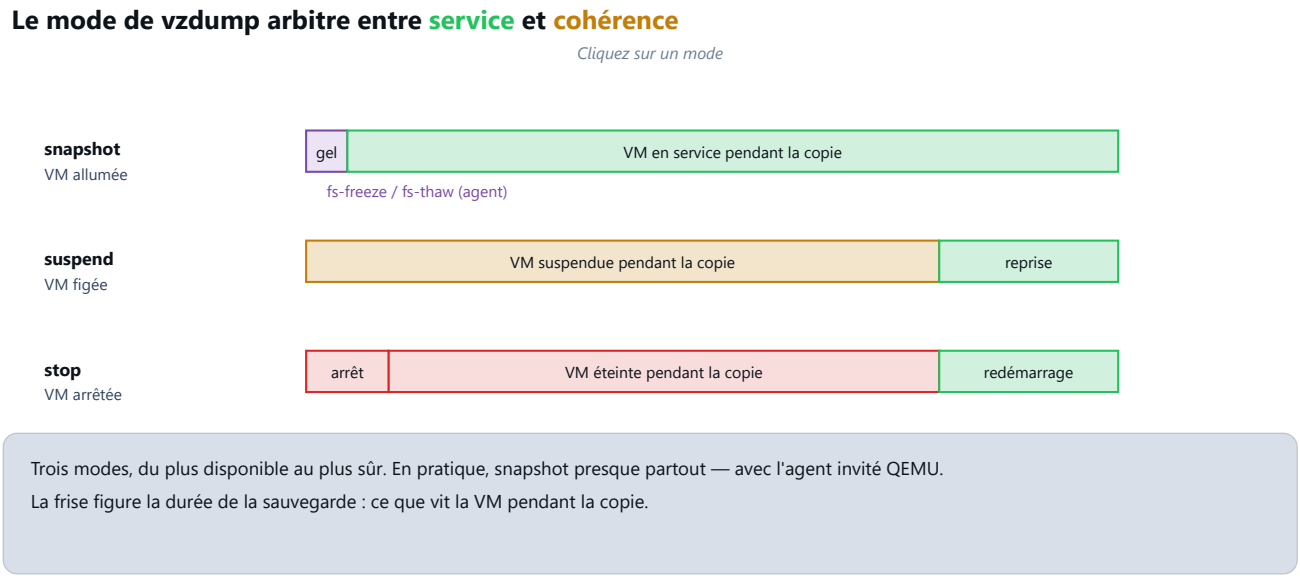
Mode	Ce qu'il fait	Contrepartie
`snapshot`	Sauvegarde à chaud, VM allumée, via un instantané du disque	Cohérence **au niveau bloc** seulement — d'où l'intérêt de l'agent invité
`suspend`	Fige la VM le temps de la sauvegarde	Interruption de service pendant la copie
`stop`	Arrête la VM, sauvegarde, redémarre	Cohérence parfaite, mais coupure franche

En pratique on emploie `snapshot` presque partout, et c'est là que **l'agent invité QEMU prend tout son sens**. Au moment du snapshot, Proxmox lui demande de *geler* le système de fichiers de

Ce support est la propriété de Dawan France.

la VM (`fsfreeze`), le temps de figer le disque. Sans agent, le snapshot capture un disque dans l'état d'une coupure de courant — récupérable, mais pas garanti cohérent pour une base de données en pleine écriture.

Avec agent, la sauvegarde est propre. C'est la raison de fond pour laquelle on installe systématiquement `qemu-guest-agent` dans les VMs.



Ce que vit la VM pendant la sauvegarde, selon le mode. En `snapshot`, elle reste en service et l'agent invité ne gèle le système de fichiers qu'un instant. En `suspend`, elle est figée tout du long.

En `stop`, elle est éteinte puis redémarrée. Cliquez sur un mode pour le détail.

> **Une restauration écrase, elle ne fusionne pas.** `qmrestore` sur un VMID existant **remplace intégralement** la VM par le contenu de l'archive — tout ce qui a été écrit depuis la sauvegarde est perdu. Restaurer vers un **nouveau VMID** est la précaution qui permet de comparer avant de remplacer, et c'est la pratique à enseigner.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Sauvegarder la VM 100 en mode snapshot avec compression ZSTD
vzdump 100 --storage local --mode snapshot --compress zstd

# Sauvegarder toutes les VMs d'un nœud
vzdump --all --storage local --mode snapshot --compress zstd
```

Ce support est la propriété de Dawan France.

```
# Lister les sauvegardes disponibles dans le stockage local
pvesm list local --content backup

# Restaurer depuis une archive (crée une nouvelle VM)
qmrestore /var/lib/vz/dump/vzdump-qemu-100-2026_09_29-13_51_54.vma.zst 199 \
  --storage local-lvm --unique 1

# Restaurer vers la même VM ID (VM doit être éteinte ou inexistante)
qmrestore /var/lib/vz/dump/vzdump-qemu-100-2026_09_29-13_51_54.vma.zst 100 \
  --storage local-lvm --force 1
```

Résultat attendu de `vzdump`, sur la VM 101 du lab (même commande, avec `101` à la place de `100`) :

```
INFO: starting new backup job: vzdump 101 --compress zstd --mode snapshot --storage
local
INFO: Starting Backup of VM 101 (qemu)
INFO: Backup started at 2026-09-29 13:51:54
INFO: status = running
INFO: VM Name: vm-web-01
INFO: include disk 'scsi0' 'local-lvm:vm-101-disk-0' 20G
INFO: backup mode: snapshot
INFO: ionice priority: 7
INFO: snapshots found (not included into backup)
INFO: creating vzdump archive '/var/lib/vz/dump/vzdump-qemu-101-2026_09_29-
13_51_54.vma.zst'
INFO: issuing guest-agent 'fs-freeze' command
INFO: starting backup via QMP command
INFO: issuing guest-agent 'fs-thaw' command
INFO: started backup task '4bd7ddcc-f7ae-4933-8d0d-b04e2c47b046'
INFO: resuming VM again
INFO:   9% (1.9 GiB of 20.0 GiB) in 3s, read: 638.0 MiB/s, write: 358.1 MiB/s
INFO: 100% (20.0 GiB of 20.0 GiB) in 5s, read: 9.1 GiB/s, write: 22.6 MiB/s
INFO: backup is sparse: 18.91 GiB (94%) total zero data
INFO: transferred 20.00 GiB in 5 seconds (4.0 GiB/s)
INFO: archive file size: 402MB
INFO: Finished Backup of VM 101 (00:00:06)
INFO: Backup finished at 2026-09-29 13:52:00
```

Ce support est la propriété de Dawan France.

```
INFO: Backup job finished successfully
INFO: notified via target `mail-to-root`
```

L'archive est créée dans `/var/lib/vz/dump/` ; son nom inclut le VMID, la date et l'heure. Cinq lignes de ce journal méritent qu'on s'y arrête.

Ligne	Ce qu'elle dit
`fs-freeze` puis `fs-thaw`	L'agent invité a gelé puis dégelé les systèmes de fichiers de la VM le temps de poser le snapshot. Sans agent, ces deux lignes manquent, et la sauvegarde n'a que la cohérence d'une coupure de courant
`snapshots found (not included into backup)`	La VM a des snapshots : ils ne sont **pas** sauvegardés. L'archive contient l'état présent de la VM, rien d'autre
`backup is sparse: 18.91 GiB (94%) total zero data`	94 % du disque n'a jamais été écrit, et vzdump ne stocke pas ces zéros
`archive file size: 402MB`	La taille réelle, après compression zstd, pour un disque de 20 Go. C'est ce rapport qu'il faut surveiller pour dimensionner le stockage des sauvegardes
`notified via target mail-to-root`	Le résultat part vers la cible de notification par défaut, `mail-to-root` (le courrier local de root)

> **Planification automatique :** dans l'interface, **Datacenter** → **Backup** → **Add** permet de créer des jobs planifiés (CRON : déclenchement à heures fixes). En production, on planifie des sauvegardes nocturnes avec une rétention configurable (garder les 7 derniers jours, par exemple).

9.3 Rétention automatique des sauvegardes

Sauvegarder chaque nuit sans jamais purger les anciennes archives conduit inévitablement à saturer le stockage de sauvegarde. Proxmox intègre un mécanisme de rétention basé sur des règles de type **grand-père/père/fils**, comme la plupart des outils de sauvegarde professionnels. Il conserve un nombre défini de sauvegardes quotidiennes, hebdomadaires, mensuelles et annuelles, et purge automatiquement le reste.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Sauvegarder avec rétention : 7 quotidiennes, 4 hebdomadaires, 6 mensuelles
vzdump 100 --storage local --mode snapshot --compress zstd \
  --prune-backups keep-daily=7,keep-weekly=4,keep-monthly=6

# Appliquer une politique de rétention à des sauvegardes déjà existantes
# (sans en créer de nouvelle) : d'abord à blanc, pour voir ce qui partirait
pvesm prune-backups local --vmid 100 --keep-daily 7 --keep-weekly 4 --dry-run 1
pvesm prune-backups local --vmid 100 --keep-daily 7 --keep-weekly 4   # puis pour de vrai
```

Résultat attendu de la sauvegarde avec rétention (extrait) :

```
INFO: Finished Backup of VM 100 (00:00:12)
INFO: prune older backups with retention: keep-daily=7, keep-monthly=6, keep-weekly=4
INFO: removing backup 'local:backup/vzdump-qemu-100-2026_05_10-02_00_00.vma.zst'
INFO: pruned 3 backup(s) not covered by keep-retention policy
```

Le passage `--dry-run 1` de `pvesm prune-backups` affiche, pour chaque archive, `keep` ou `remove` et la règle qui la retient. C'est la seule façon de vérifier une politique **avant** qu'elle n'efface quoi que ce soit.

Proxmox conserve les sauvegardes les plus pertinentes selon la politique définie, et supprime automatiquement les archives devenues redondantes. Par exemple, une sauvegarde quotidienne de plus de 7 jours n'est conservée que si elle correspond aussi à une échéance hebdomadaire ou mensuelle encore active.

Ces mêmes options de rétention sont configurables graphiquement dans **Datacenter** → **Backup** → **Add**, sous la section **Retention**, au moment de planifier un job de sauvegarde récurrent. En production, définissez cette politique dès la création du job plutôt que de laisser les sauvegardes s'accumuler sans limite. Un stockage de sauvegarde saturé fait systématiquement échouer la sauvegarde suivante, généralement au pire moment.

- 📖 Documentation Proxmox — Backup and Restore
- 📖 Configurer et automatiser les sauvegardes — pve-admin-guide

À retenir : `vzdump` produit une archive complète de la VM ; le **mode** (`snapshot` , `suspend` , `stop`) arbitre entre continuité de service et cohérence. `qmrestore` **écrase** la VM cible. Une

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

rétention (`keep-daily` , `keep-weekly` …) évite de saturer le stockage.

> **▶ À pratiquer maintenant — Atelier 6**
>
> Prendre un instantané, casser puis revenir en arrière ; sauvegarder la VM, puis la restaurer sous un nouveau numéro sans conflit d'adresse.

Ce support est la propriété de Dawan France.

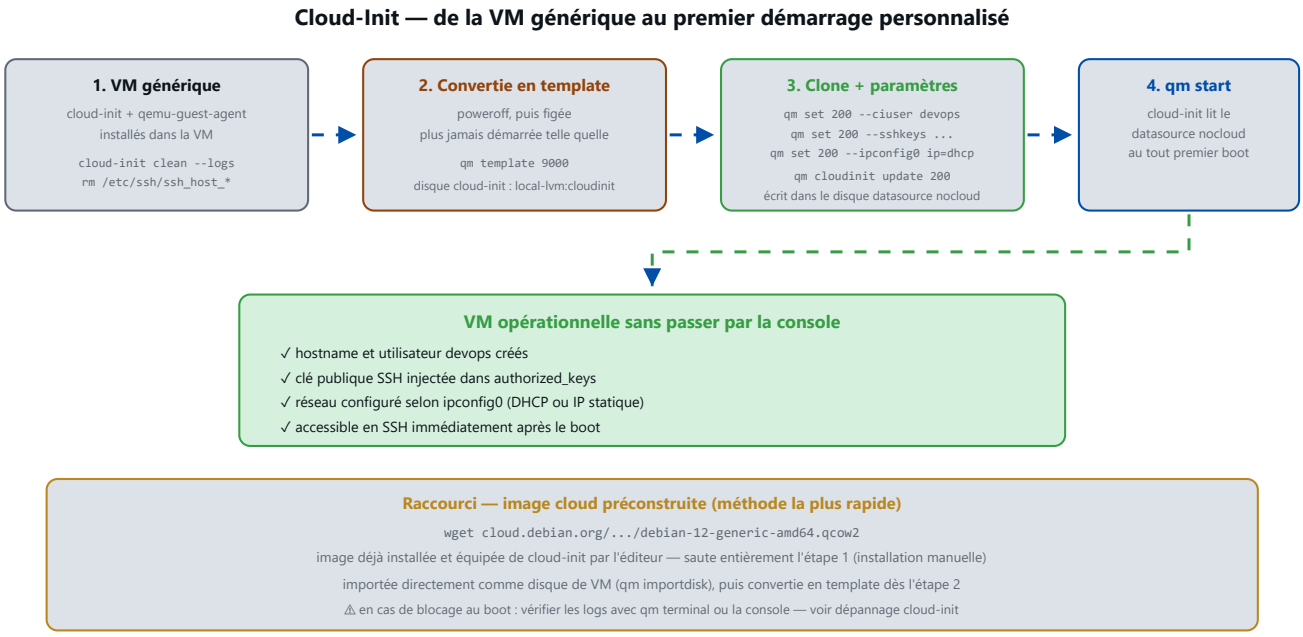
10. Cloud-Init — préparer une VM pour le déploiement automatisé

10.1 Le problème que cloud-init résout

Installer Debian manuellement depuis un ISO prend entre 10 et 20 minutes, et le résultat est une VM unique qu'on ne peut pas reproduire à l'identique. Dans un environnement où l'on déploie régulièrement des VMs (développement, tests, staging), ce processus devient un goulot d'étranglement.

Cloud-init est un outil de personnalisation au premier démarrage, adopté comme standard par la quasi-totalité des images cloud (Debian cloud, Ubuntu Server, Rocky Linux, AlmaLinux). Au lieu d'installer un OS depuis une ISO, vous partez d'une image déjà installée et configurée, et cloud-init injecte vos paramètres — hostname, utilisateur, clé SSH, réseau — lors du premier boot.

Proxmox intègre cloud-init nativement : il génère un disque virtuel spécial (datasource `nocloud`) qui est attaché à la VM et lu par cloud-init au démarrage.



10.2 Préparer une image cloud-init

Ce support est la propriété de Dawan France.

> **Le cours et l'atelier n'utilisent pas les mêmes numéros.** Dans les exemples du cours, le template porte le numéro 9000, et les clones 101 et 102 sont des clones liés.
>
> Dans les ateliers, votre template est la VM **100**, et vos VM 101 et 102 en sont des clones **intégraux**. Suivez les numéros de l'atelier.

La méthode complète part d'une VM Debian installée manuellement, puis la transforme en image générique prête à être clonée.

Comprendre ce que cloud-init fait, avant de le préparer. Une image de template est volontairement **anonyme** : pas de nom d'hôte, pas d'utilisateur, pas de clé, pas d'adresse.

Si on clonait une VM déjà configurée, toutes les copies partageraient la même identité : même hostname, même clé SSH d'hôte, parfois même `machine-id`. Cela casse la résolution et la sécurité. Cloud-init résout cela en **injectant l'identité au premier démarrage**, à partir des paramètres passés par Proxmox.

D'où une étape que les débutants oublient et qui explique la moitié des templates ratés : le **nettoyage** avant conversion en template. Il faut effacer de la VM tout ce qui la rend unique — `machine-id`, clés SSH d'hôte, journaux, baux DHCP — sans quoi les clones en hériteront. C'est l'objet de la commande `cloud-init clean` et de la purge du `machine-id`, détaillées dans les étapes qui suivent.

La règle à retenir : **un template se prépare une fois, se nettoie, puis ne se redémarre plus jamais tel quel**. Le rallumer régénérerait le `machine-id` qu'on vient d'effacer, et le template redeviendrait pollué.

Étape 1 — Dans la VM : installer les prérequis

Dans la VM (sa **Console**, ou en SSH) :

```
# Depuis l'intérieur de la VM Debian
apt update                                    # la liste des paquets
apt install -y cloud-init qemu-guest-agent    # cloud-init et l'agent QEMU

# Rien à activer pour l'agent : son service démarre seul dès que la VM possède
# le port de l'agent (option --agent enabled=1, posée sur le template)

# Nettoyer avant de créer le template
```

Ce support est la propriété de Dawan France.

```
cloud-init clean --logs          # supprimer les données d'init précédentes
truncate -s 0 /etc/machine-id    # réinitialiser l'ID machine (chaque clone aura le sien)
rm -f /var/lib/dbus/machine-id   # retirer la copie D-Bus de l'identifiant
ln -s /etc/machine-id /var/lib/dbus/machine-id   # D-Bus lira le même identifiant que le système
rm -f /etc/ssh/ssh_host_*        # supprimer les clés SSH hôte (régénérées au premier boot)
apt clean   # vider le cache des paquets téléchargés
poweroff   # éteindre la VM avant d'en faire un template
```

Résultat attendu — la VM s'éteint proprement :

```
...
[  OK  ] Stopped Session 1 of User root.
[  OK  ] Reached target System Shutdown.
[  OK  ] Reached target Late Shutdown Services.
         Unmounting /boot...
[  OK  ] Unmounted /boot.
         Stopping D-Bus System Message Bus...
[  OK  ] Stopped D-Bus System Message Bus.
[  OK  ] Reached target Unmount All Filesystems.
Connection to 10.44.1.55 closed.
```

La connexion SSH se ferme automatiquement à l'extinction de la VM. Dans l'interface Proxmox, la VM passe en statut `stopped` — c'est le moment de configurer cloud-init depuis l'hôte.

Étape 2 — Sur l'hôte Proxmox : configurer cloud-init

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Ajouter le lecteur cloud-init (génère l'image datasource nocloud)
# Il prend la place du lecteur CD de l'installation, devenu inutile
qm set 100 --ide2 local-lvm:cloudinit

# Configurer le hostname, l'utilisateur et la clé SSH via cloud-init
qm set 100 --ciuser devops
qm set 100 --cipassword 'dawan'   # mot de passe de cet utilisateur
qm set 100 --sshkeys ~/.ssh/authorized_keys   # clé SSH publique autorisée
```

Ce support est la propriété de Dawan France.

```
# Adresse FIXE, conformement a la regle de l infrastructure de lab :
# chaque stagiaire dispose d un reseau NATe en 10.44.1.x, et toute
# ressource creee y porte une adresse fixe. Prenez celle de votre plan
# d adressage plutot que celle-ci telle quelle.
qm set 100 --ipconfig0 ip=10.44.1.55/24,gw=10.44.1.1

# Le mode DHCP existe (valeur "dhcp" pour ipconfig0), mais il est ecarte ici :
# clonees du meme template obtiendraient un bail imprevisible, et le plan
# d adressage du lab exige une adresse fixe sur chaque ressource creee.

# Régénérer l'image cloud-init après modification
qm cloudinit update 100

# Démarrer la VM — cloud-init applique la configuration au premier boot
qm start 100
```

Résultat attendu au premier démarrage — dans la VM 101 du lab, depuis le nœud, par l'agent invité (`qm guest exec 101 -- cloud-init status` , puis `qm guest exec 101 -- tail -n 1 /var/log/cloud-init-output.log`) :

```
status: done
Cloud-init v. 25.1.4 finished at Tue, 29 Sep 2026 11:53:13 +0000. Datasource
DataSourceNoCloud [seed=/dev/sr0].  Up 10.96 seconds
```

`status: done` : cloud-init a terminé tous ses modules. `DataSourceNoCloud [seed=/dev/sr0]` : la configuration venait du lecteur CD que Proxmox génère (la ligne `generating cloud-init ISO` au `qm start`).

L'utilisateur `devops` est créé avec la clé SSH injectée, le hostname est appliqué, et le réseau configuré selon `ipconfig0` . La VM est immédiatement accessible en SSH sans passer par la console.

Étape 3 — Dans l'interface graphique :

Proxmox expose un onglet dédié cloud-init pour chaque VM : **VM** → **Cloud-Init**.

Champ	Valeur
User	`devops`

Ce support est la propriété de Dawan France.

Champ	Valeur
Password	`dawan`
SSH public key	coller la clé publique
IP Config (net0)	`ip=10.44.1.55/24,gw=10.44.1.1` — **adresse fixe**
DNS domain	*(facultatif)*

Après modification, cliquez **Regenerate Image** pour appliquer les changements avant le prochain démarrage.

- 📖 Documentation Proxmox — Cloud Init Support
- 📖 Documentation cloud-init
- 🎥 Techno Tim — la procédure complète, de l'image au template cloud-init (vidéo en anglais)

10.3 Utiliser des images cloud préconstruites

Les distributions majeures publient des images cloud prêtes à l'emploi, déjà équipées de cloud-init. C'est la méthode la plus rapide pour créer un template, et en pratique celle qu'on utilise presque toujours. La préparation manuelle de la section précédente sert surtout à comprendre ce que l'image cloud fait déjà pour vous.

L'intérêt dépasse la rapidité. Ces images sont **construites et testées par la distribution elle-même** : nettoyées, minimales, sans `machine-id` figé, avec cloud-init déjà configuré.

On évite ainsi toutes les erreurs de préparation vues plus haut. C'est aussi ce qui garantit la reproductibilité — l'image `debian-13-generic` est identique pour tout le monde, à l'octet près, vérifiable par son empreinte.

Le format `.qcow2` de ces images appelle un réflexe : leur **taille de disque est minuscule** (souvent 2 à 3 Go), car elles sont provisionnées au plus juste. Après import, il faut donc **agrandir le disque** avec `qm disk resize` avant de créer des VMs utilisables — sans quoi le clone démarre avec un disque presque plein. C'est l'oubli le plus fréquent avec les images cloud.

Un dernier point : la ligne `qm set ... --ide2 ...cloudinit` de la procédure n'ajoute pas un disque de données mais le **lecteur cloud-init**. C'est un petit disque virtuel qui transporte l'identité (nom, IP, clé) jusqu'à la VM au premier démarrage. Sans lui, cloud-init n'a aucune source de configuration à lire, et le clone démarre anonyme.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Télécharger l'image cloud Debian 13 (Trixie)
wget -P /tmp/ https://cloud.debian.org/images/cloud/trixie/latest/debian-13-generic-amd64.qcow2

# Créer une VM squelette (sans disque) ; processeur compatible avec tout le cluster
qm create 9000 --name debian-13-cloud --memory 2048 --cores 2 \
  --net0 virtio,bridge=vmbr0 --ostype l26 --agent enabled=1 --cpu x86-64-v2-AES

# Importer l'image cloud : elle arrive comme disque non attaché (unused0)
qm disk import 9000 /tmp/debian-13-generic-amd64.qcow2 local-lvm

# Attacher le disque importé et configurer le boot
qm set 9000 --scsi0 local-lvm:vm-9000-disk-0,discard=on,ssd=1
qm set 9000 --boot order=scsi0   # démarrer sur ce disque

# Ajouter le disque cloud-init
qm set 9000 --ide2 local-lvm:cloudinit
qm set 9000 --serial0 socket --vga serial0   # console série, utilisée par les images cloud

# Agrandir le disque à 20 Go
qm disk resize 9000 scsi0 20G

# Convertir en template
qm template 9000
```

Résultat attendu — l'import, puis la conversion en template :

```
importing disk '/tmp/debian-13-generic-amd64.qcow2' to VM 9000 ...
  Logical volume "vm-9000-disk-0" created.
transferred 3.0 GiB of 3.0 GiB (100.00%)
unused0: successfully imported disk 'local-lvm:vm-9000-disk-0'
...
  Renamed "vm-9000-disk-0" to "base-9000-disk-0" in volume group "pve"
  Logical volume pve/base-9000-disk-0 changed.
```

Deux détails de cette sortie éclairent la suite. L'import dépose le disque en `unused0` : il existe, mais n'est branché nulle part, d'où la commande `qm set 9000 --scsi0 …` qui l'attache.

Ce support est la propriété de Dawan France.

Et la conversion **renomme** le disque de `vm-9000-disk-0` en `base-9000-disk-0`, en lecture seule. C'est le signe qu'il est devenu une base, que les clones liés vont partager sans jamais la modifier. La VM 9000 passe en mode template dans l'interface — son icône change et elle n'est plus démarrable directement.

- 📖 Images cloud officielles de Debian — pour Debian 13, le dossier trixie/latest contient `debian-13-generic-amd64.qcow2` et son fichier d'empreintes `SHA512SUMS`
- 📖 Images cloud officielles d'Ubuntu
- 🎥 Jim's Garage — cloner des VM avec cloud-init et Terraform (vidéo en anglais) : coupler le template à Terraform pour automatiser le provisionnement, une pratique courante en environnement DevOps.

> **▶ À pratiquer maintenant — Atelier 7 et Atelier 8**
>
> Passer à l'image cloud : importer le disque Debian prêt à l'emploi, puis le personnaliser par cloud-init, le tester et le nettoyer avant d'en faire un template.

Ce support est la propriété de Dawan France.

11. Templates et clones — déploiement en série

11.1 Convertir une VM en template

Une VM préparée (OS installé, cloud-init configuré, agent QEMU actif) se convertit en template en un clic. Si elle a démarré pendant sa préparation, ne serait-ce que pour un test, elle doit d'abord être **nettoyée** et éteinte (§10.2).

La commande `cloud-init clean --logs --machine-id` efface l'état de cloud-init et remet l'identifiant système à `uninitialized`. Sans cela, tous les clones partagent le même `machine-id`, vérifié sur le lab. Un template n'est plus démarrable directement — il sert uniquement de source de clonage.

Via l'interface graphique :

> **VM → More → Convert to Template**

Une boîte de confirmation s'ouvre. Après validation, l'icône de la VM dans le panneau de gauche change pour indiquer qu'il s'agit d'un template.

Via CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Convertir la VM 100 en template
qm template 100

# Vérifier que la conversion a réussi
qm config 100 | grep template
# → template: 1
```

Résultat attendu de `qm config 100 | grep template` :

```
template: 1
```

Ce support est la propriété de Dawan France.

Deux points de départ pour un template. Le plus courant actuellement est une image cloud préparée par cloud-init (§10) : chaque clone reçoit au premier démarrage son nom, son adresse, son compte et ses clés.

Mais n'importe quelle VM installée peut devenir un template, par exemple la VM 105 installée depuis l'ISO. La différence porte sur ce qui reste à faire, avant la conversion puis sur chaque clone.

	Template cloud-init (atelier 9, partie A)	Template d'une VM installée (atelier 9, partie B)
Avant la conversion	`cloud-init clean --logs --machine-id`	Vider `/etc/machine-id`, supprimer `/var/lib/dbus/machine-id` et les clés `/etc/ssh/ssh_host_*`
Au premier démarrage d'un clone	Nom, adresse, compte et clés SSH injectés par Proxmox	Le nom et l'adresse du modèle : à changer dans la console, carte débranchée
Clés SSH de la machine	Générées par cloud-init	`ssh-keygen -A` à la main ; sans clés, le service SSH ne démarre pas

Deux constats du lab justifient le nettoyage manuel. Sans suppression de `/var/lib/dbus/machine-id`, le clone a repris le `machine-id` de la VM d'origine : systemd recrée `/etc/machine-id` à partir de cette copie. Et le clone démarre avec l'adresse fixe du modèle : branché tel quel, il entre en conflit avec la VM d'origine.

Le template d'une VM installée convient pour quelques machines, ou pour un système qui n'a pas d'image cloud. Dès qu'on déploie en série, cloud-init fait ce travail à la place de l'administrateur.

11.2 Cloner un template — clone lié vs clone intégral

Proxmox propose deux types de clones avec des compromis différents :

Clone lié (Linked Clone) : le clone partage les blocs de données du template grâce au mécanisme CoW (Copy-on-Write). Le disque du clone n'occupe que l'espace des données **modifiées** par rapport au template.

Très rapide à créer (quelques secondes), mais le clone **dépend** du template : ses blocs non modifiés sont lus dans le disque `base-` du template. Proxmox refuse d'ailleurs de supprimer un template tant qu'un clone lié en dépend.

Ce support est la propriété de Dawan France.

Clone intégral (Full Clone) : copie complète et indépendante du disque. Prend plus de temps et d'espace, mais le clone est totalement autonome — le template peut être supprimé sans impact.

Via l'interface graphique :

Template → More → Clone

Champ	Valeur
Mode	`Linked Clone` (atelier) ou `Full Clone` (production)
Target Storage	`local-lvm`
VM ID	`101`
Name	`debian-prod-01`

Via CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Clone lié (rapide — dépend du template 9000)
qm clone 9000 101 --name debian-prod-01 --linked 1

# Clone intégral (indépendant)
qm clone 9000 102 --name debian-prod-02 --full 1 --storage local-lvm

# Personnaliser le clone avant démarrage
qm set 101 --ipconfig0 ip=10.44.1.56/24,gw=10.44.1.1
qm set 101 --ciuser devops --sshkeys ~/.ssh/authorized_keys   # utilisateur et clé SSH du clone
qm cloudinit update 101   # régénérer le lecteur cloud-init

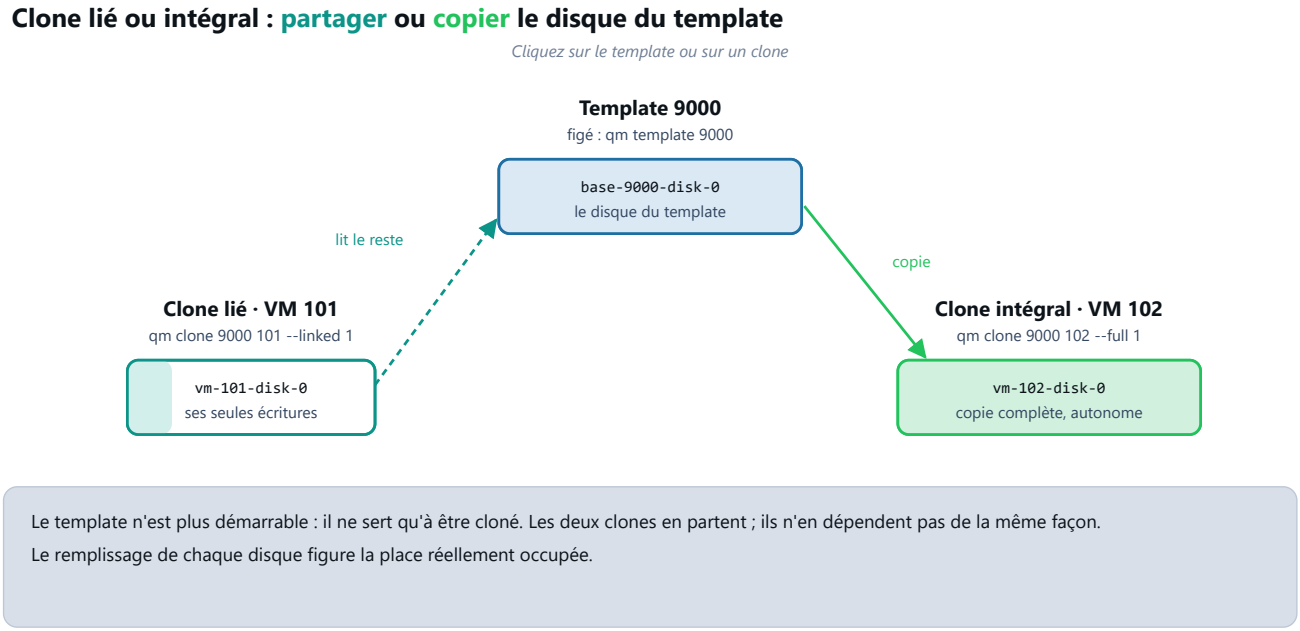
# Démarrer le clone
qm start 101
```

Résultat attendu de `qm clone 9000 101 --name debian-prod-01 --linked 1` :

```
create linked clone of drive scsi0 (local-lvm:base-9000-disk-0)
  Logical volume "vm-101-disk-0" created.
```

Ce support est la propriété de Dawan France.

Le clone est créé en quelques secondes : son disque `vm-101-disk-0` n'est qu'un instantané du disque `base-9000-disk-0`, qui ne contiendra que ce que le clone écrira. Après `qm start 101`, cloud-init applique les paramètres spécifiques au clone (hostname, IP, clé SSH) au premier démarrage.



Ce que chaque type de clone fait du disque du template. Le clone lié n'écrit que ses différences et lit le reste dans `base-9000-disk-0` : il en dépend.

Le clone intégral en a une copie complète : il est autonome. Cliquez sur un élément pour le détail.

Revenons à la définition de base : **cloner**, c'est fabriquer une nouvelle VM à partir d'une autre. La question est : le clone a-t-il besoin de l'original pour vivre ?

Terme	Ce que c'est	Dans le laboratoire
Template	Une VM figée, qui ne démarre plus et ne sert qu'à être clonée	VM 100 `debian13-template` (atelier 9)
Disque `base-`	Le disque du template, en lecture seule	`base-100-disk-0`
Clone lié	Un disque qui ne garde que ses différences et lit le reste dans le disque du template	Le mode proposé par défaut dans la fenêtre **Clone**
Copie à l'écriture (*copy-on-write*)	Un bloc n'est copié dans le clone que lorsqu'il le modifie	En quelques secondes, le clone est prêt
Clone intégral	Une copie autonome du disque ; plus longue, plus lourde	VM 101 et 102 de l'atelier 9 (`--full`)

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
cloud-init	Ce qui donne à chaque clone son nom, son IP et sa clé SSH au premier démarrage	`qm set 101 --ipconfig0 …`

Un clone lié lit ses blocs non modifiés dans le disque du template : c'est pourquoi Proxmox refuse de supprimer un template tant qu'un clone lié en dépend. Un clone intégral, lui, survit à la suppression du template.

Où les voir sur votre nœud :

Notion	Où la voir
Le template	Une icône à part dans l'arborescence ; le menu **More** → **Clone** au lieu de **Start**
Le lien entre un clone lié et son template	Le journal du clonage : `create linked clone of drive scsi0 (local-lvm:base-<template>-disk-0)`. `qm config <clone>` ne le montre pas : il n'affiche que le disque du clone
Le mode de clonage	La fenêtre **Clone** : champ **Mode**, `Linked Clone` ou `Full Clone`
L'espace occupé par chaque clone	`lvs` : le `Data%` d'un clone lié reste faible, celui d'un clone intégral reprend toutes les données du template

Résultat attendu (pxe211, Proxmox VE 9, stockage `local-lvm`) : un modèle d'essai 990, puis un clone lié 991.

```
root@pxe211:~# qm template 990
  Renamed "vm-990-disk-0" to "base-990-disk-0" in volume group "pve"
root@pxe211:~# qm clone 990 991 --name essai-lie
creating a clone of VM 990 with ID 991
create linked clone of drive scsi0 (local-lvm:base-990-disk-0)
  Logical volume "vm-991-disk-0" created.
root@pxe211:~# qm config 991 | grep scsi0
boot: order=scsi0
scsi0: local-lvm:vm-991-disk-0,size=1G
```

Sans `--full`, `qm clone` d'un template fait un clone lié : le journal le dit (`create linked clone`) et nomme le disque du modèle qui sert de base. La configuration du clone, elle, ne cite que son propre disque. Pour savoir si une VM est un clone lié, on lit le journal de la tâche de clonage.

Ce support est la propriété de Dawan France.

▶ À pratiquer maintenant — Atelier 9

Convertir la VM 100 en template et en tirer deux clones à adresse fixe ; puis faire un template d'une VM installée à la main, pour comparer.

Ce support est la propriété de Dawan France.

Ateliers — chapitre 2

> **Infrastructure** : nœud `pxe2<N>1` (même nœud que le Chapitre 1 — les conteneurs LXC restent en place)

État du lab en fin de chapitre :

- VM 105 `vm-debian` installée depuis l'ISO, adresse fixe `.54`, second disque de 6 Gio, un instantané et une sauvegarde (ateliers 2 à 6). VM 110 `vm-debian-restauree`, carte débranchée, arrêtée ou supprimée
- Template `debian13-template` (VMID 100) créé via image cloud + cloud-init, nettoyé, converti en template en lecture seule (ateliers 7 à 9)
- Deux clones de production (VM 101 `vm-web-01`, VM 102 `vm-web-02`) avec IP statiques (ateliers 8 et 9)
- Template 106 `debian-modele`, tiré de la VM 105 et nettoyé à la main, et son clone lié VM 107 `vm-classique-01` en `.58`, personnalisé dans sa console (atelier 9, partie B)

Atelier 1 — Vérifier les prérequis KVM

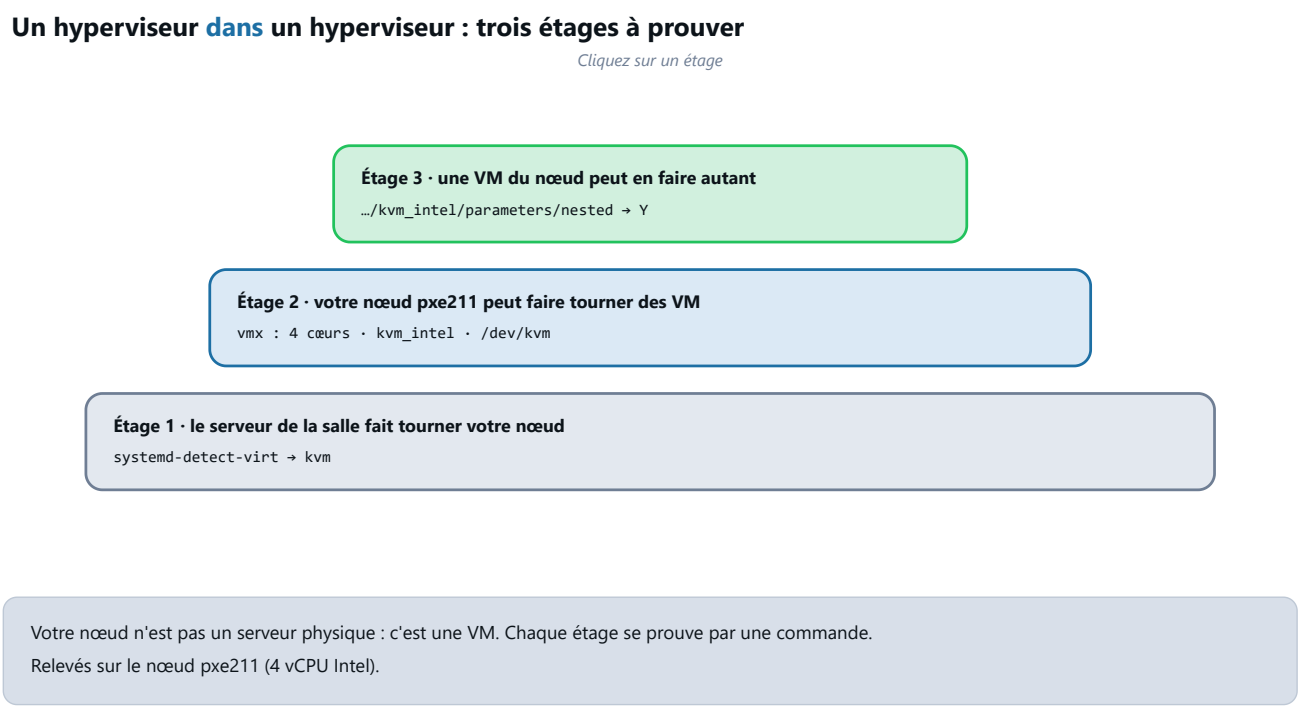
Contexte

Avant de créer la moindre VM KVM, deux vérifications s'imposent — et les sauter est la cause des deux pannes les plus fréquentes du chapitre. D'abord, le processeur du nœud doit exposer ses extensions de virtualisation matérielle (Intel VT-x ou AMD-V) : sans elles, les VMs démarrent en émulation logicielle, lentes au point d'être inutilisables.

Ensuite, le pool de stockage doit avoir de la place : une VM en provisionnement fin qui remplit un pool saturé fige toutes les autres.

Résultat à obtenir : la confirmation que la virtualisation matérielle est active sur le nœud. La preuve que ce nœud est lui-même une machine virtuelle capable d'en héberger d'autres. Et une lecture claire du taux de remplissage du pool `local-lvm`, avant d'y créer des VMs.

Ce support est la propriété de Dawan France.



Les trois étages que l'atelier fait prouver, du serveur de la salle à une VM qui peut elle-même héberger des VM. Cliquez sur un étage pour la commande qui le prouve et le résultat obtenu.

Arborescence attendue

```
Aucun fichier produit : deux vérifications en ligne de commande,
depuis le Shell du nœud.
```

Ce que vous devez faire

1. **Vérifier les extensions CPU** : chercher `vmx` (Intel) ou `svm` (AMD) dans `/proc/cpuinfo`, et confirmer que le module KVM est bien chargé.
2. **Un cran plus loin : votre nœud est lui-même une VM.** Prouvez-le, puis montrez qu'il peut quand même faire tourner des VM KVM, et même laisser ses VM en faire autant (virtualisation imbriquée).
3. **Évaluer le pool LVM-thin** : lire le taux d'occupation des données **et** des métadonnées du pool. Demandez-vous ce qui se passerait si les métadonnées atteignaient 100 % — c'est un piège moins connu que le remplissage des données.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Extensions de virtualisation CPU

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Vérifier VT-x (Intel) ou AMD-V : une ligne « flags » par cœur logique
nproc
grep -c "^flags.*vmx" /proc/cpuinfo   # Intel
grep -c "^flags.*svm" /proc/cpuinfo   # AMD (l'un ou l'autre)

# Vérifier les modules KVM chargés
lsmod | grep ^kvm
```

Résultat attendu (nœud `pxe211`, 4 vCPU Intel) :

```
4
4
0
kvm_intel             507904  0
kvm                  1445888  1 kvm_intel
```

Un nombre supérieur à 0 pour `vmx` ou `svm` suffit : c'est le nombre de cœurs logiques qui annoncent l'extension, ici autant que `nproc`. Ne comptez pas avec un simple `grep -c vmx` : les noyaux récents ajoutent une ligne `vmx flags` par cœur, et le compte double (`8` sur le lab).

`kvm_intel` (ou `kvm_amd`) doit figurer à côté de `kvm` : c'est lui qui parle au processeur. Si seul `kvm` apparaît, l'extension est désactivée dans le BIOS.

Étape 1 bis — Un hyperviseur dans un hyperviseur

Votre nœud n'est pas un serveur physique : c'est une VM, installée sur un serveur de la salle. Pour qu'il héberge à son tour des VM KVM, le serveur de la salle doit lui transmettre les extensions du processeur. C'est la **virtualisation imbriquée** (*nested*).

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Sur quoi tourne ce système ? « none » sur du matériel, le nom de l'hyperviseur sinon
systemd-detect-virt
# Le périphérique par lequel QEMU parle au module KVM
ls -l /dev/kvm
# Ce nœud laisse-t-il ses propres VM utiliser KVM à leur tour ? (Intel, puis AMD)
cat /sys/module/kvm_intel/parameters/nested
cat /sys/module/kvm_amd/parameters/nested   # même question, processeur AMD
```

Résultat attendu (nœud `pxe211`, processeur Intel) :

```
kvm
crw-rw---- 1 root kvm 10, 232 Sep 28 07:46 /dev/kvm
Y
cat: /sys/module/kvm_amd/parameters/nested: No such file or directory
```

Un seul des deux fichiers existe : celui du module du processeur. Sur un nœud Intel, la question AMD répond `No such file or directory`, et inversement : c'est normal.

Trois étages se lisent ici :

Étage	Ce qui le prouve
le serveur de la salle fait tourner votre nœud	`systemd-detect-virt` répond `kvm`
votre nœud peut faire tourner des VM	le drapeau `vmx` / `svm` de l'étape 1 et `/dev/kvm`
une VM de votre nœud pourrait en faire tourner d'autres	`nested` vaut `Y` (ou `1`)

C'est ce troisième étage, la **virtualisation imbriquée**, qui permettrait d'installer un Proxmox dans une VM de votre nœud. Le formateur l'a fait sur son poste à l'atelier 1 du chapitre 1.

Sur un serveur physique, `systemd-detect-virt` répondrait `none`. Et si `/dev/kvm` manquait, Proxmox ne pourrait démarrer une VM qu'en désactivant l'accélération (option **KVM hardware virtualization** à `No`), avec des performances très dégradées.

Étape 2 — État du pool LVM-thin

Le pool LVM-thin `local-lvm` est le stockage par défaut pour les disques VM.

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Remplissage des données ET des métadonnées du pool thin
lvs -o name,size,data_percent,metadata_percent pve/data
```

Résultat attendu (pool presque vide, avec le seul CT 200) :

```
  LV   LSize   Data%  Meta%
  data 129.87g 0.74   1.66
```

`Data%` est le remplissage des blocs de données, `Meta%` celui de la table qui dit à quel volume appartient chaque bloc. Les deux saturent indépendamment.

Un pool rempli à 40 % de données mais à 100 % de métadonnées gèle ses VM tout autant. C'est le piège de la question posée plus haut : on l'évite en surveillant les deux colonnes.

> **Règle des 70% :** ne jamais dépasser 70% de remplissage d'un pool LVM-thin. Le thin provisioning alloue l'espace à la demande.
>
> Au-delà de ce seuil, les performances I/O chutent. Le risque de saturation complète augmente fortement ; elle bloque toutes les écritures sur le pool, VMs comprises.

Atelier 2 — Créer et installer une VM Debian depuis l'ISO

Contexte

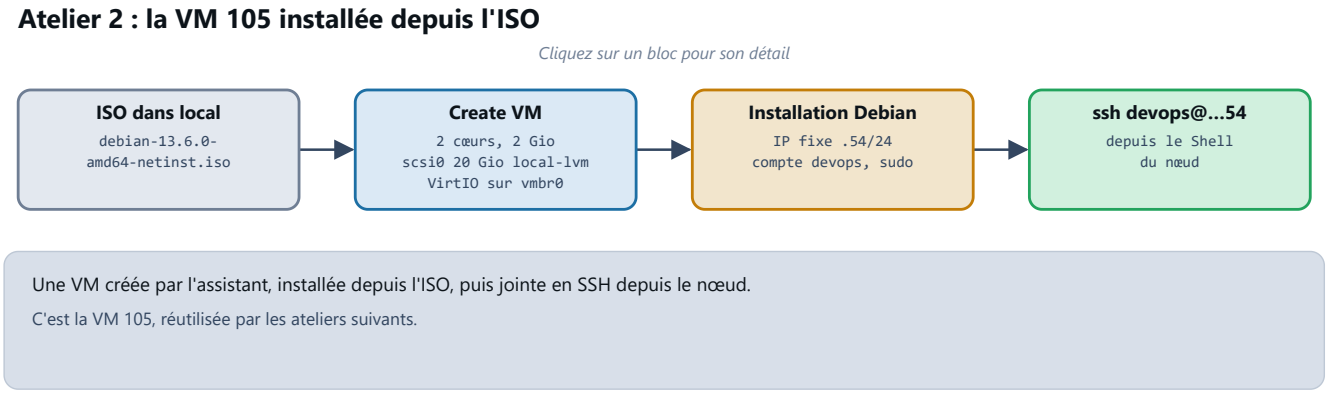
Votre première machine virtuelle, de bout en bout. Vous la créez avec l'assistant **Create VM**, puis vous installez Debian depuis l'ISO avec une adresse IP fixe. Vous vérifiez enfin qu'elle démarre seule sur son disque et qu'on peut s'y connecter.

C'est la VM 105 `vm-debian` : elle sert à tous les ateliers suivants jusqu'à l'atelier 6 (cycle de vie, stockage, agent, instantanés).

Cours : §3 (médias) et §4 (créer une VM).

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Résultat à obtenir : la VM 105 `vm-debian`, 2 cœurs, 2 Gio de mémoire, disque de 20 Gio sur `local-lvm`, carte VirtIO sur `vmbr0`. Elle est installée depuis l'ISO Debian 13 et démarre sur son disque avec l'adresse fixe `10.<site>.<N>.54/24` et un compte `devops` qui a les droits `sudo`.



Le schéma suit la VM 105 de l'ISO à la première connexion SSH. Cliquez sur chaque bloc pour les valeurs à saisir.

Arborescence attendue

```
local → ISO Images
└── debian-13.6.0-amd64-netinst.iso     ← téléchargée au chapitre 1

VM 105 « vm-debian »
├── scsi0 : local-lvm, 20 Gio           ← disque système
├── ide2  : l'ISO, retirée après l'installation (atelier 3)
└── net0  : VirtIO sur vmbr0            ← adresse fixe 10.<site>.<N>.54 dans Debian
```

Ce que vous devez faire

1. **Vérifier** que l'ISO Debian 13 est dans le stockage `local` (chapitre 1).

2. **Créer la VM 105** avec l'assistant **Create VM**, onglet par onglet, avec les valeurs du tableau de la solution. Repérer ensuite l'option `qm create` qui correspond à chaque champ.

   Cochez la case **Advanced** en bas de l'assistant. Relevez, sans les modifier, trois valeurs par défaut : le **Shutdown timeout** (onglet General), la case **Backup** du disque (Disks) et les **CPU units** (CPU).

Ce support est la propriété de Dawan France.

3. **La démarrer** et l'installer depuis sa console : français, adresse **fixe** `10.<site>.<N>.54/24` (jamais de DHCP), utilisateur `devops`, sans environnement de bureau, avec le serveur SSH.

4. **Vérifier**, après le redémarrage, qu'elle démarre sur son disque et qu'elle a bien son adresse IPv4 ; corriger si besoin.
5. **Se connecter** depuis le Shell du nœud : `ssh devops@10.<site>.<N>.54`.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple :** les captures ont été prises sur `pxe212` (Proxmox VE 9.2.21), où la VM 105 était en `192.168.1.154`. Les résultats en texte viennent du lab passé sur le plan des stagiaires (`pxe211`, VM 105 en `10.59.1.54`). Chez vous : `10.<site>.<N>.54`.

Étape 1 — Créer la VM avec l'assistant

En haut à droite, **Create VM**. Remplissez les onglets dans l'ordre :

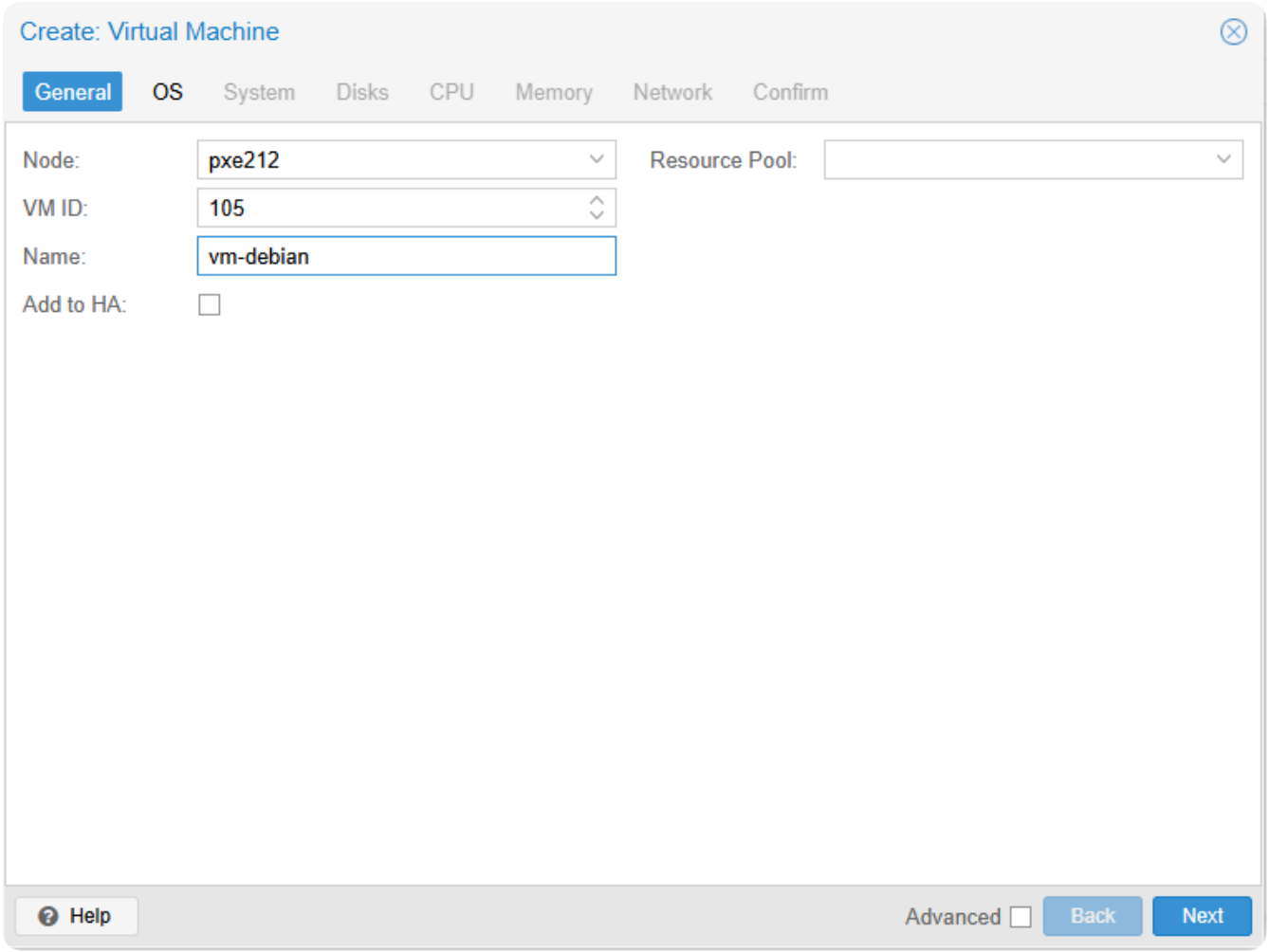
Onglet	Champ	Valeur
General	Node / VM ID / Name	votre nœud / `105` / `vm-debian`
OS	ISO image	`debian-13.6.0-amd64-netinst.iso` (stockage `local`)
OS	Type / Version	Linux / `7.x - 2.6 Kernel` (proposé d'office)
System	Qemu Agent	**coché**
System	Machine	`q35` (le défaut est `i440fx` : à changer)
System	BIOS / SCSI Controller	laissés par défaut : `SeaBIOS`, `VirtIO SCSI single`
Disks	Storage / Disk size	`local-lvm` / `20`
CPU	Sockets / Cores / Type	`1` / `2` / laissé par défaut (`x86-64-v2-AES`)
Memory	Memory (MiB)	`2048`
Network	Bridge / Model	`vmbr0` / `VirtIO (paravirtualized)`
Confirm	Start after created	**décoché**

Les options avancées. Cochez **Advanced**, en bas à droite de l'assistant : chaque onglet affiche des champs de plus, sous un trait. On les laisse tous par défaut.

Les trois valeurs demandées : **Shutdown timeout** `default`, soit 180 secondes laissées à la VM pour s'éteindre avant un arrêt forcé. **Backup** est coché : le disque est inclus dans les sauvegardes. **CPU units** vaut `100`, le poids de la VM quand le processeur du nœud est saturé.

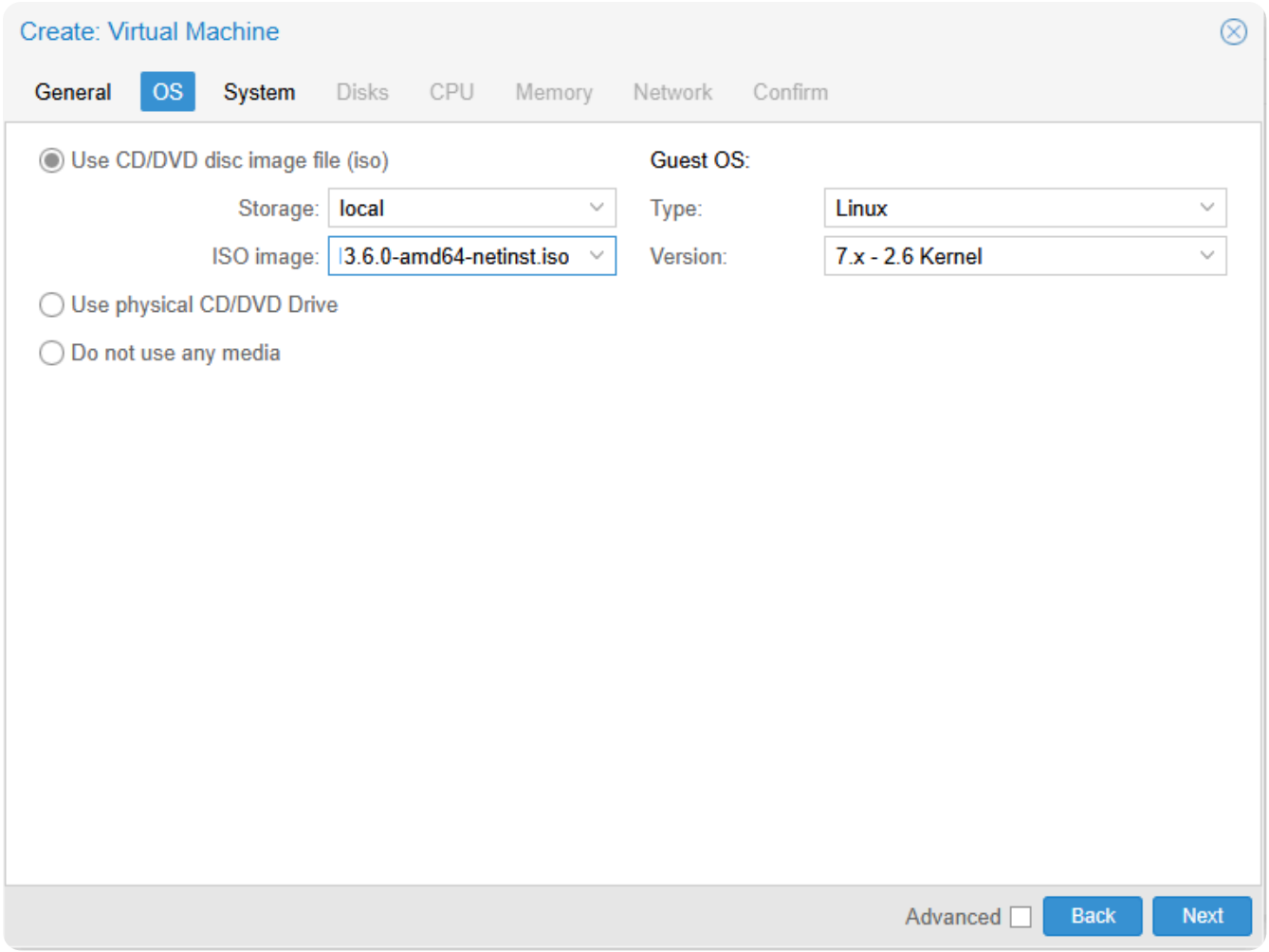
Le rôle de chaque champ avancé est détaillé au cours §4.1.

Ce support est la propriété de Dawan France.



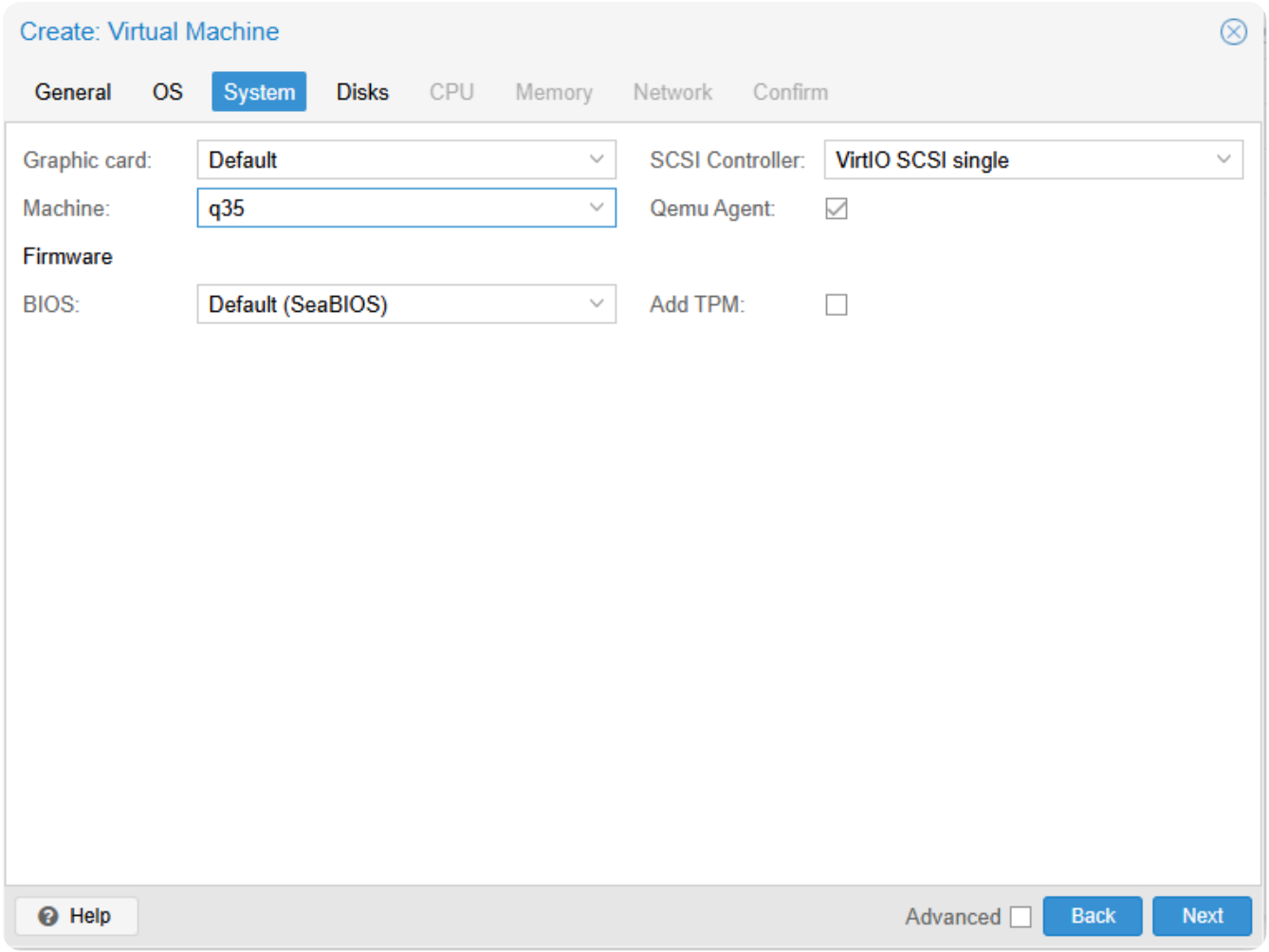
L'onglet **General** : le nœud, le numéro et le nom. Le numéro doit être libre sur tout le cluster.

Ce support est la propriété de Dawan France.



L'onglet **OS** : le lecteur CD de la VM reçoit l'ISO. Le type `Linux`, version `7.x - 2.6 Kernel`, règle des optimisations propres à Linux.

Ce support est la propriété de Dawan France.



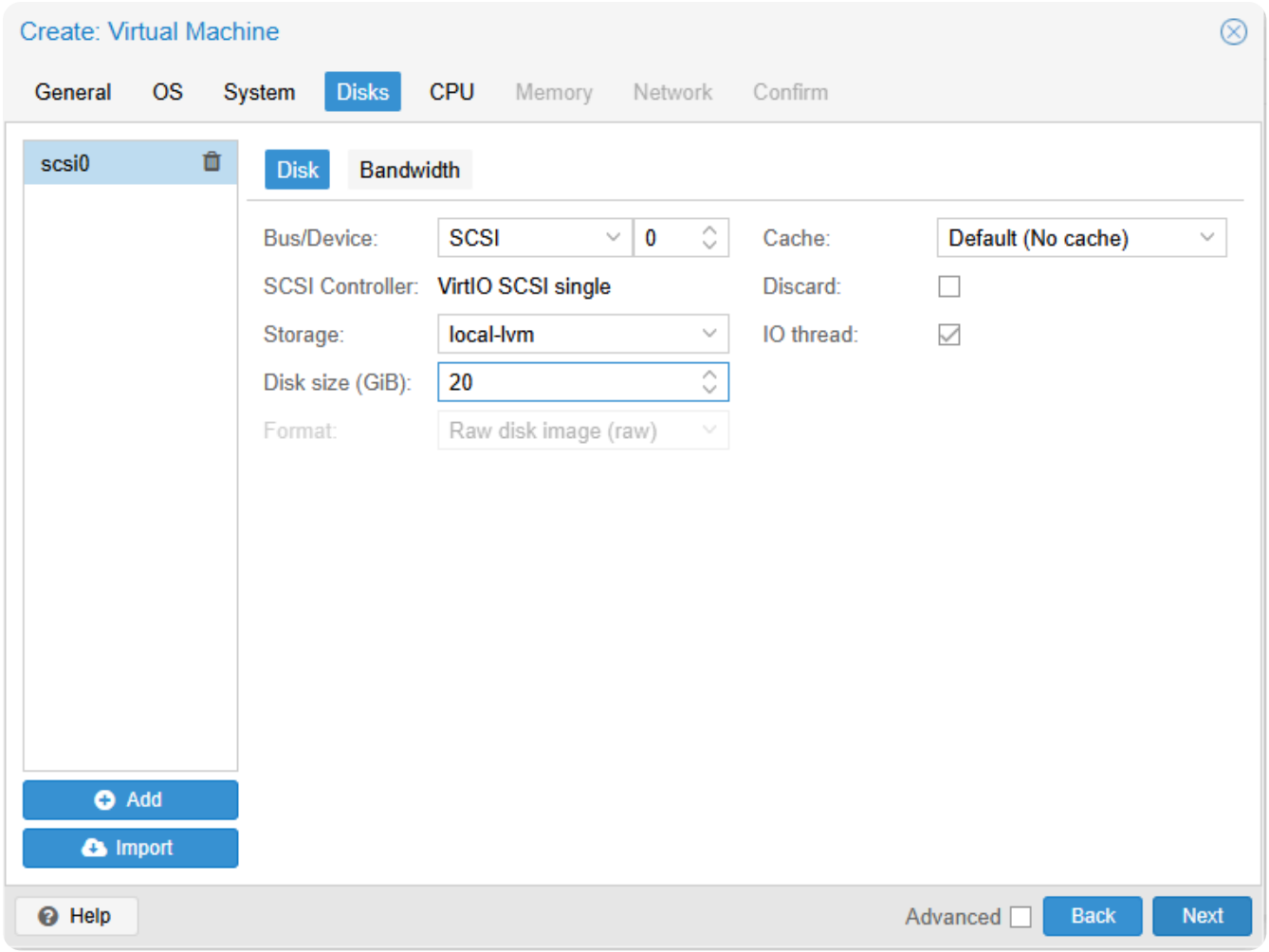
L'onglet **System** : **Machine** passe de `Default (i440fx)` à `q35`, les deux seuls choix de la liste. Cochez **Qemu Agent**, qui prépare le canal avec l'agent invité (atelier 5). BIOS et contrôleur SCSI restent par défaut.

Pourquoi `q35`. Le champ **Machine** choisit la carte mère que QEMU simule. `i440fx` reproduit un chipset Intel de 1996, avec un bus PCI.

`q35` reproduit un chipset Intel plus récent, qui fournit un bus **PCI Express** virtuel. La documentation de Proxmox le recommande quand on veut faire passer une carte PCIe physique à la VM (*passthrough*), et c'est avec `q35` qu'on peut activer un vIOMMU.

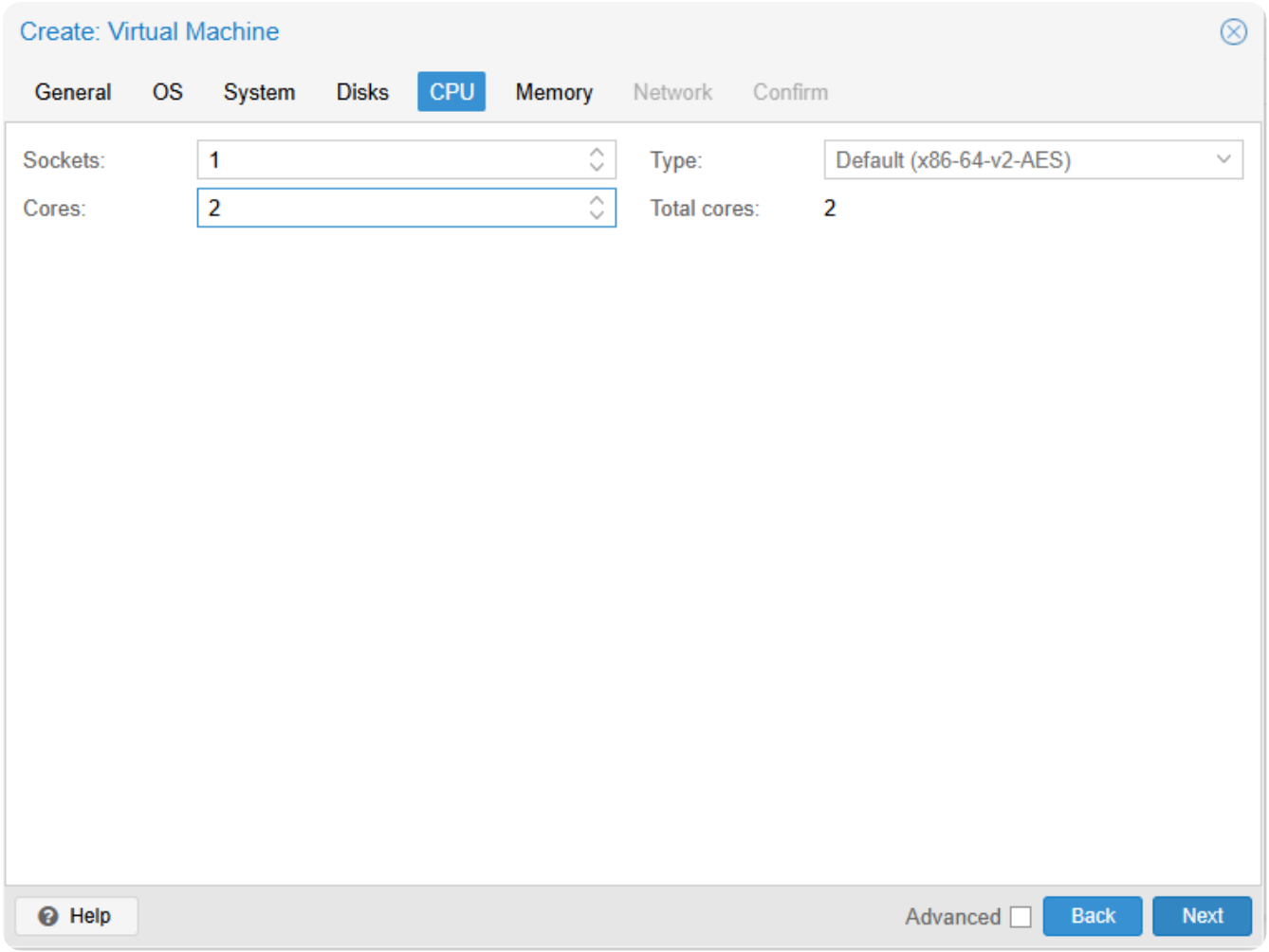
C'est aussi le choix des VM des ateliers suivants : le template de l'atelier 7 est en `q35`. Sur le lab, Debian nomme sa carte `ens18` dans les deux cas : l'adresse fixe de l'installation ne change pas.

Ce support est la propriété de Dawan France.



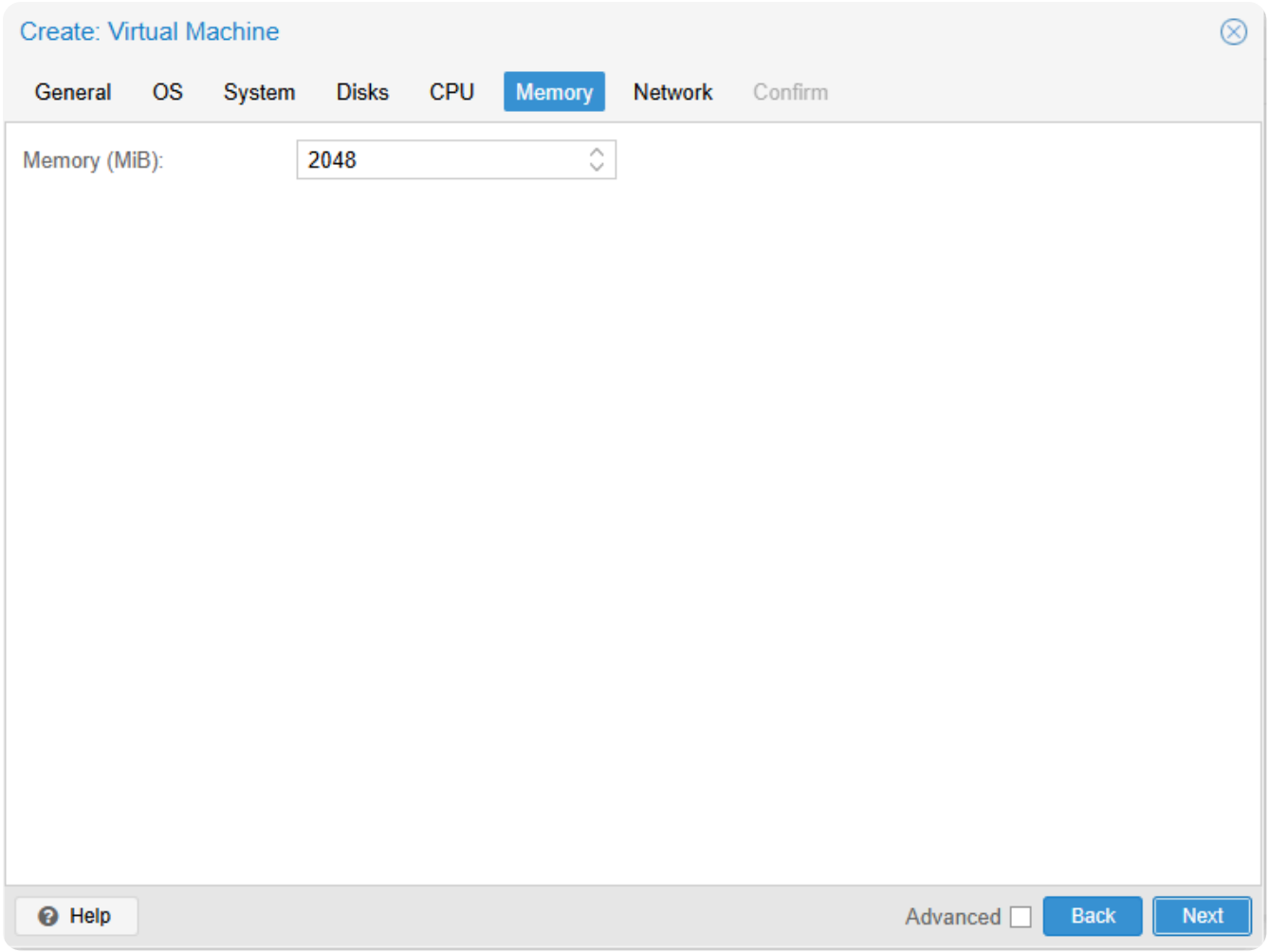
L'onglet **Disks** : un disque SCSI de 20 Gio sur `local-lvm`, avec `IO thread` coché d'office.

Ce support est la propriété de Dawan France.



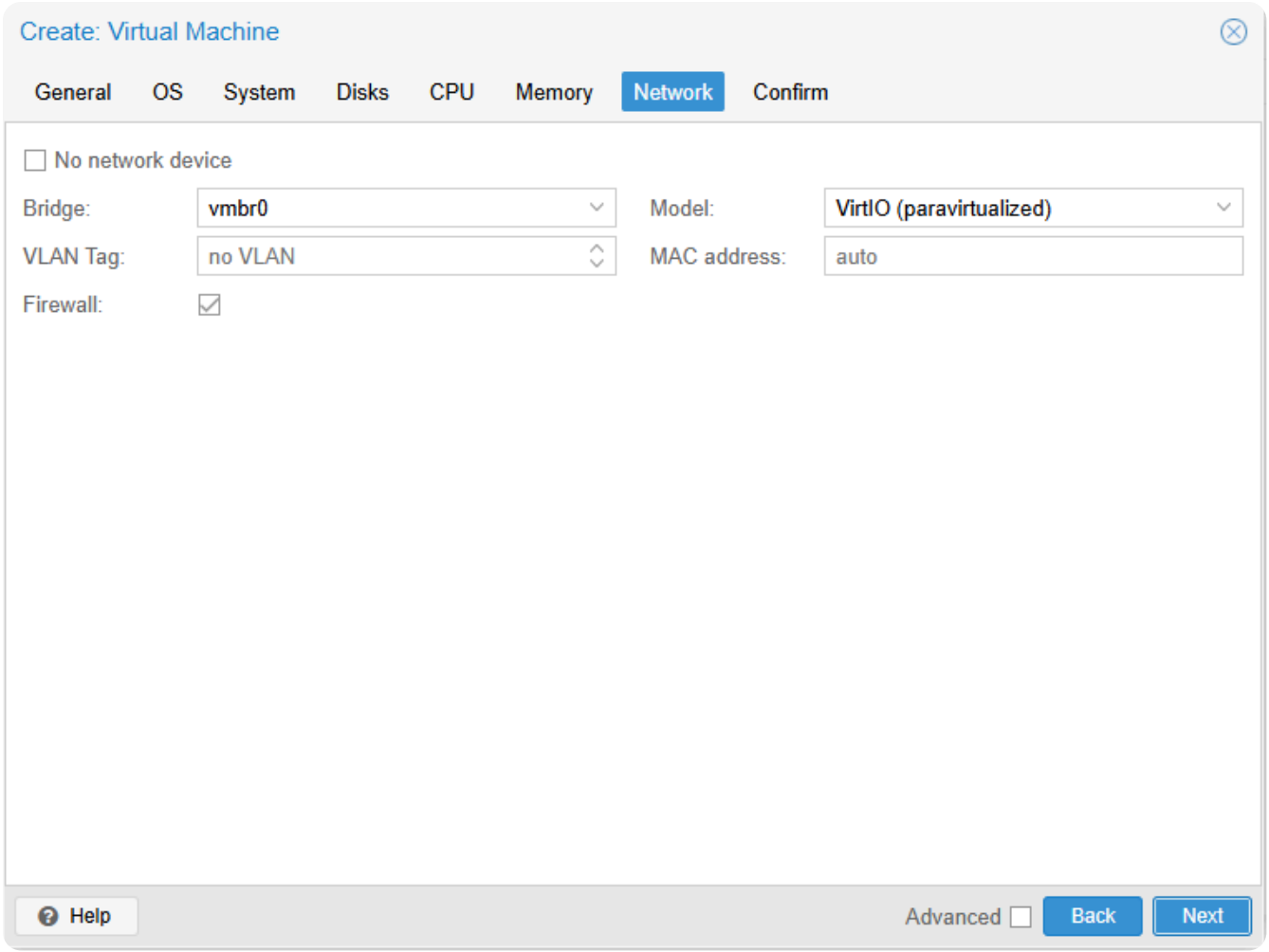
L'onglet **CPU** : 1 socket, 2 cœurs. Le type par défaut, `x86-64-v2-AES`, permet de migrer la VM vers n'importe quel nœud d'un cluster.

Ce support est la propriété de Dawan France.



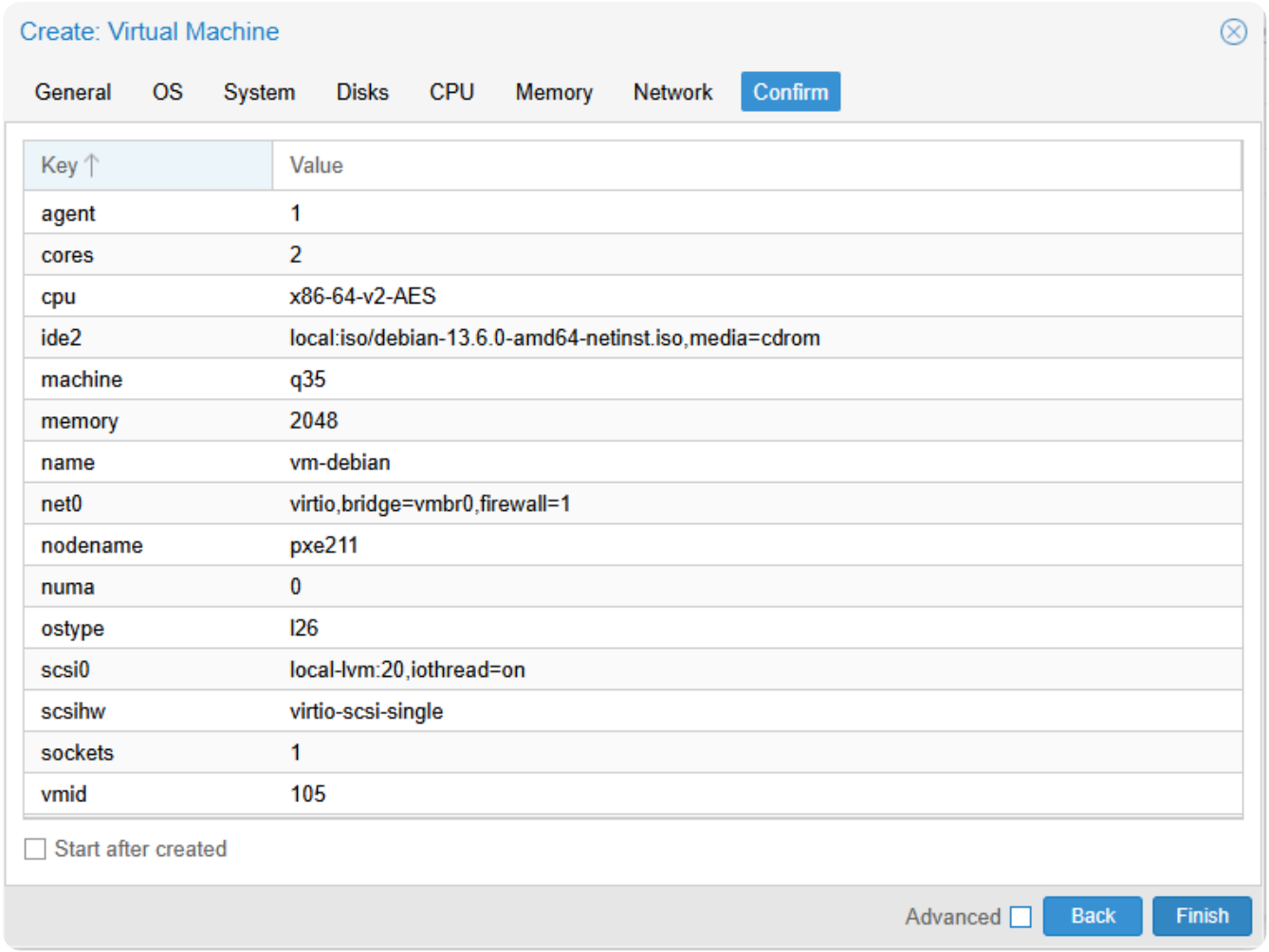
L'onglet **Memory** : 2048 Mio. Le ballooning est présenté au cours, §7.4.

Ce support est la propriété de Dawan France.



L'onglet **Network** : la carte se branche sur `vmbr0` ; le modèle `VirtIO` est le plus rapide (cours §7.1). La case **Firewall** cochée ne filtre rien tant que le pare-feu n'est pas activé.

Ce support est la propriété de Dawan France.



L'onglet **Confirm** liste exactement ce qui part vers Proxmox : `agent 1`, `cores 2`, `cpu x86-64-v2-AES`, `ide2` (l'ISO), `machine q35`, `memory 2048`, `net0 virtio,bridge=vmbr0,firewall=1`, `scsi0 local-lvm:20,iothread=on`. Laissez **Start after created** décoché, puis **Finish**.

La même VM en ligne de commande. Chaque champ de l'assistant correspond à une option de `qm create` :

Onglet	Champ de l'assistant	Option `qm create`
General	VM ID / Name	`105` / `--name vm-debian`
OS	ISO image	`--ide2 local:iso/debian-13.6.0-amd64-netinst.iso,media=cdrom`
OS	Type / Version : Linux, `7.x - 2.6 Kernel`	`--ostype l26`
System	Machine	`--machine q35`
System	Qemu Agent	`--agent 1`

Ce support est la propriété de Dawan France.

Onglet	Champ de l'assistant	Option `qm create`
System	SCSI Controller	`--scsihw virtio-scsi-single`
Disks	Storage / Disk size / IO thread	`--scsi0 local-lvm:20,iothread=1`
CPU	Sockets / Cores / Type	`--sockets 1 --cores 2 --cpu x86-64-v2-AES`
Memory	Memory (MiB)	`--memory 2048`
Network	Bridge / Model / Firewall	`--net0 virtio,bridge=vmbr0,firewall=1`
(ordre fixé par l'assistant)	—	`--boot 'order=scsi0;ide2;net0'`

Si vous créez la VM en ligne de commande **à la place** de l'assistant :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# La même VM que l'assistant, en une commande
qm create 105 \
  --name vm-debian \
  --ostype l26 \
  --machine q35 \
  --ide2 local:iso/debian-13.6.0-amd64-netinst.iso,media=cdrom \
  --agent 1 \
  --scsihw virtio-scsi-single \
  --scsi0 local-lvm:20,iothread=1 \
  --sockets 1 --cores 2 --cpu x86-64-v2-AES \
  --memory 2048 \
  --net0 virtio,bridge=vmbr0,firewall=1 \
  --boot 'order=scsi0;ide2;net0'
```

Résultat attendu (sur une VM de test 9106, supprimée ensuite) : `scsi0: successfully created disk 'local-lvm:vm-9106-disk-0,iothread=1,size=20G'`, précédé d'avertissements de LVM sur le provisionnement fin (cours §6). `qm config` y montre `machine: q35`, `boot: order=scsi0;ide2;net0` et `scsi0: …,iothread=1,size=20G` : la configuration est celle de l'assistant.

Dans les deux cas, vérifiez la configuration créée :

```
qm config 105   # la configuration créée, par l'assistant ou par qm create
```

Ce support est la propriété de Dawan France.

Résultat attendu — l'essentiel, sur la VM 105 du lab, créée en `i440fx`. Avec `q35`, la configuration porte une ligne de plus, `machine: q35` (sur la VM de test 9106) :

```
agent: 1
boot: order=scsi0;ide2;net0
cores: 2
cpu: x86-64-v2-AES
ide2: local:iso/debian-13.6.0-amd64-netinst.iso,media=cdrom,size=755M
memory: 2048
name: vm-debian
net0: virtio=BC:24:11:98:4E:62,bridge=vmbr0,firewall=1
scsi0: local-lvm:vm-105-disk-0,iothread=1,size=20G
scsihw: virtio-scsi-single
```

L'assistant a placé le disque **avant** le CD dans l'ordre de démarrage (`order=scsi0;ide2;net0`). Le disque vide est sauté au premier démarrage, et l'installateur se lance depuis le CD. La VM démarrera ensuite sur son disque une fois Debian installé.

Étape 2 — Installer Debian depuis la console

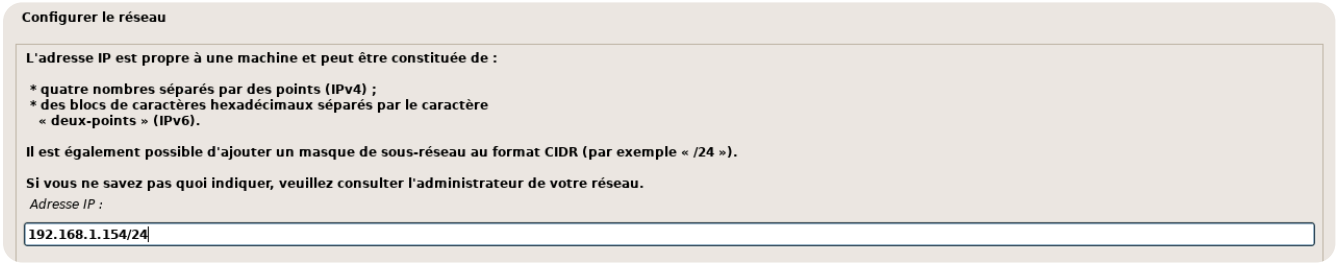
VM 105 → **Start**, puis **Console**. Dans le menu de l'installateur, **Graphical install**. Les écrans s'enchaînent ; voici les choix, dans l'ordre :

Écran	Choix
Langue, pays, clavier	Français, France, Français
Configurer le réseau	**adresse fixe** (voir ci-dessous)
Nom de machine / Domaine	`vm-debian` / champ vidé
Mot de passe du superutilisateur (« root »)	**laissé vide**, deux fois : le compte `root` est bloqué et le premier utilisateur reçoit `sudo`
Nom complet / Identifiant / Mot de passe	`devops` / `devops` / `dawan`
Partitionner les disques	Assisté - utiliser un disque entier → le disque de 20 Gio → Tout dans une seule partition → Terminer → **Oui**
Analyser d'autres supports	Non
Miroir de l'archive Debian	France → `deb.debian.org`

Ce support est la propriété de Dawan France.

Écran	Choix
Mandataire HTTP	**vide** : la VM télécharge ses paquets directement sur Internet. Facultatif : `http://10.<site>.<N>.50:3142`, le cache `apt` du conteneur 200 (cours du chapitre 1, §13) ; la VM dépend alors de ce conteneur pour chaque `apt`
popularity-contest	Non
Sélection des logiciels	**décocher** « environnement de bureau Debian » et le bureau GNOME (environnement graphique, case « GNOME »), **cocher** « serveur SSH »
Programme de démarrage GRUB (*GRand Unified Bootloader*)	Oui → `/dev/sda`

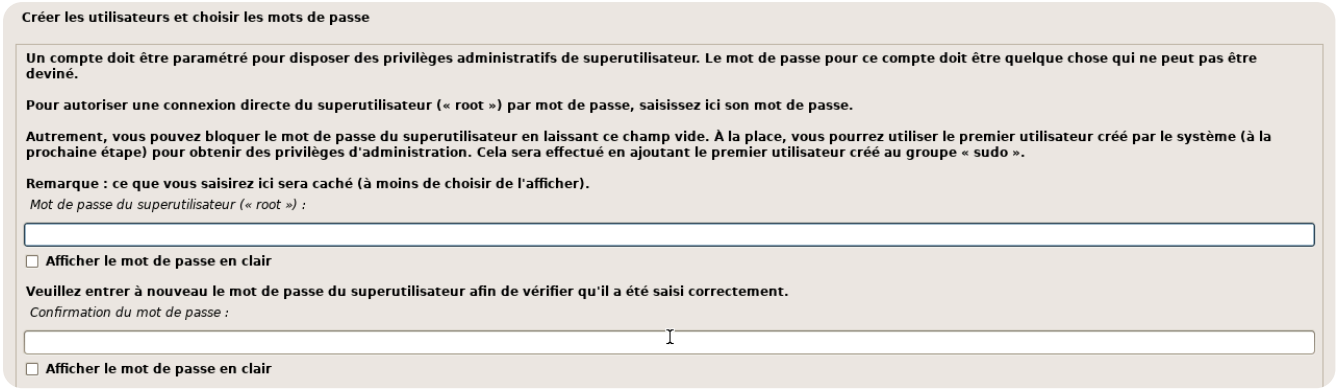
Le réseau, en adresse fixe. Si le réseau a un serveur DHCP, l'installateur prend une adresse seul et saute au nom de machine. Revenez alors en arrière (**Revenir en arrière** ou **Échap**), choisissez **Configurer vous-même le réseau**, puis saisissez l'adresse avec son masque :



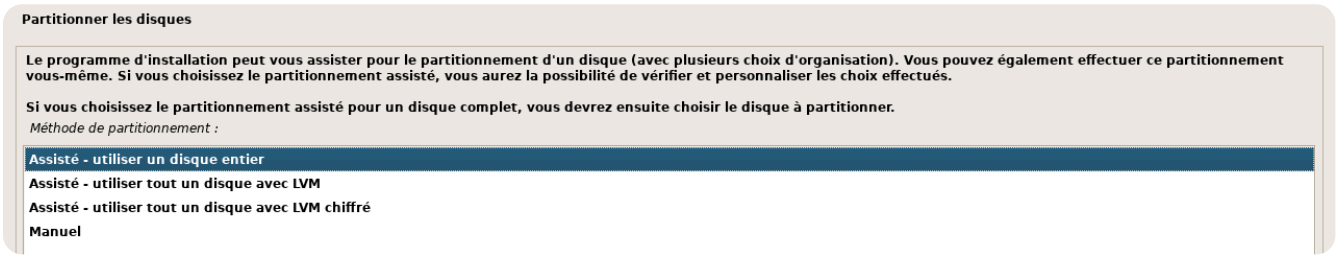
Puis la passerelle et le serveur de noms, proposés d'office en `.1`. Gardez la passerelle `10.<site>.<N>.1`, mais **remplacez le serveur de noms** : sur le lab Dawan, la passerelle du réseau NATé ne répond pas aux requêtes DNS.

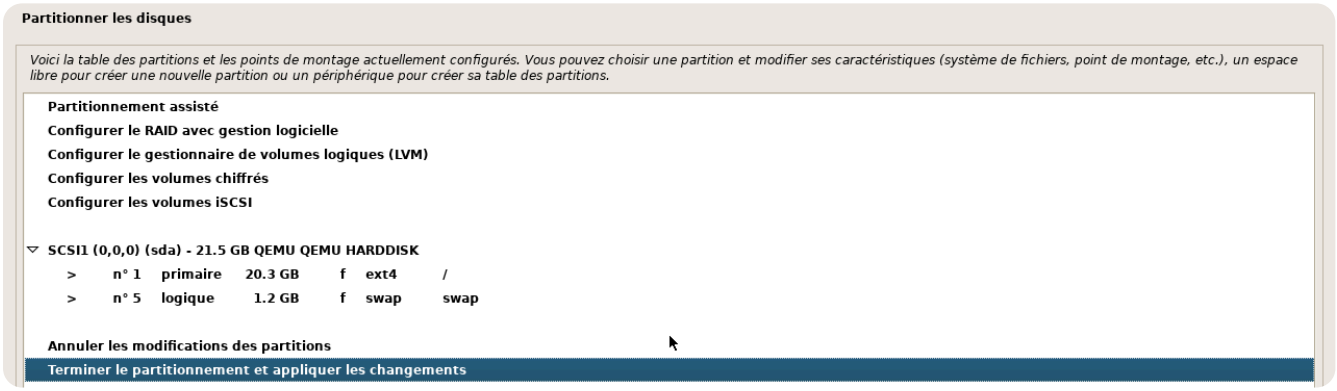
Saisissez le DNS de l'hôte, `192.168.<site>.1` : `192.168.44.1` à Nantes, `192.168.75.1` à Paris, `192.168.67.1` à Strasbourg. Sans cela, l'installateur ne joint pas le miroir Debian, et la VM ne résout aucun nom.

Ce support est la propriété de Dawan France.

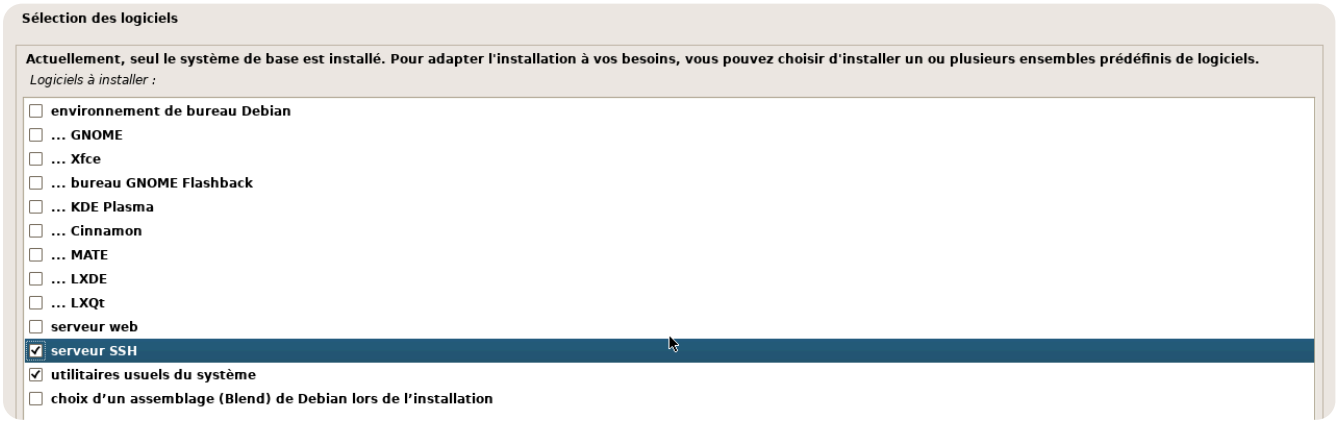


Laisser le mot de passe `root` vide est le choix le plus sûr : on n'ouvre pas de session `root`. L'administration passe alors par `sudo`, comme sur les VM cloud du chapitre.



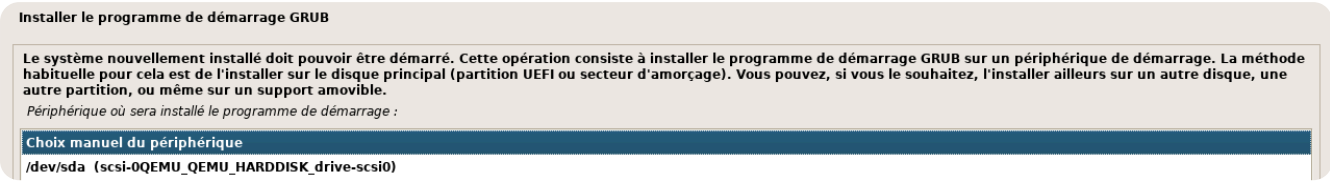


Sur le lab : une partition `ext4` de 20,3 Go montée sur `/` et un swap de 1,2 Go. À la question **Faut-il appliquer les changements sur les disques ?**, la réponse proposée est **Non** : choisissez **Oui**.



Ce support est la propriété de Dawan France.

Un serveur n'a pas besoin de bureau graphique. Cochez la case avec la **barre d'espace** ; la touche **Entrée** coche ou décoche aussi la ligne sélectionnée au lieu de valider : passez sur **Continuer** avec la touche **Tab**.



Piège : la ligne sélectionnée d'office est **Choix manuel du périphérique**. Choisissez `/dev/sda`.

L'installation se termine par **Installation terminée** : **Continuer**, la VM redémarre. Comptez une dizaine de minutes en tout.

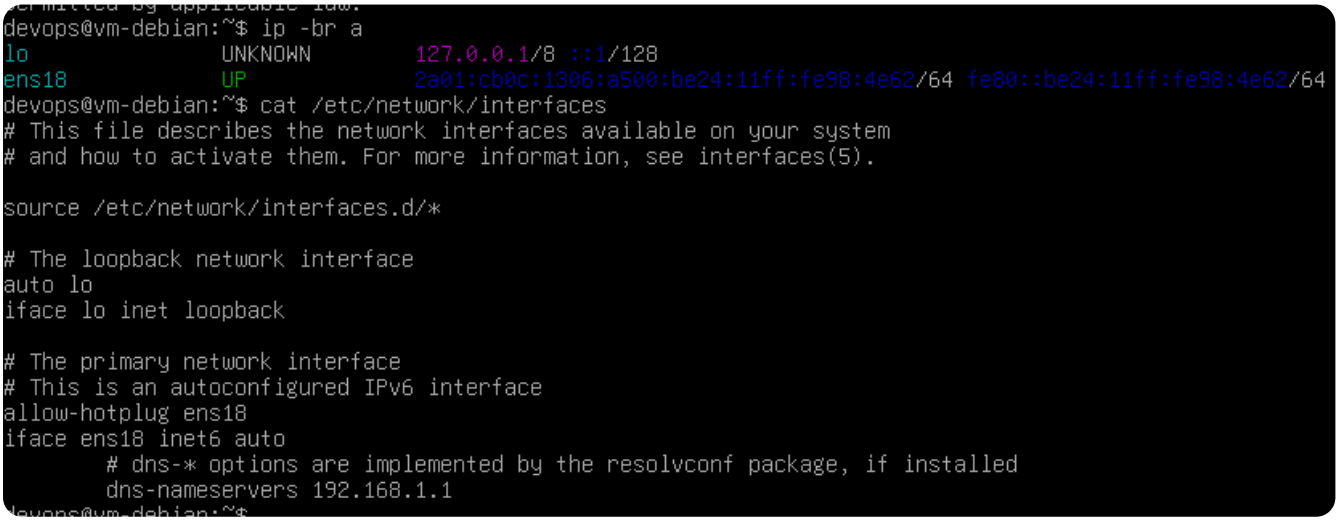
Étape 3 — Vérifier le réseau au premier démarrage

Connectez-vous en `devops`.

Dans la console de la VM :

```
ip -br a   # les adresses de la VM
```

Sur le lab, sur un réseau qui annonce de l'IPv6 : **aucune adresse IPv4**, seulement de l'IPv6, malgré la saisie manuelle de l'adresse.



`cat /etc/network/interfaces` montre pourquoi : l'installateur a écrit `iface ens18 inet6 auto`, sans ligne `inet static`. Si `ip -br a` affiche bien votre `10.<site>.<N>.54/24`, passez

Ce support est la propriété de Dawan France.

à l'étape 4. Sinon, corrigez.

Dans la console de la VM :

```
sudo nano /etc/network/interfaces.d/ens18   # la carte en adresse fixe, dans un fichier à part
```

Le contenu de `ens18`, à taper dans `nano`, puis **Ctrl + O**, Entrée et **Ctrl + X** :

```
auto ens18
iface ens18 inet static
    address 10.<site>.<N>.54/24
    gateway 10.<site>.<N>.1
```

Dans la console de la VM :

```
sudo nano /etc/network/interfaces          # désactiver les deux lignes écrites par l'installateur
```

Dans `nano`, ajoutez un `#` au début des deux lignes `allow-hotplug ens18` et `iface ens18 inet6 auto`, puis enregistrez. Une ligne qui commence par `#` est ignorée.

Dans la console de la VM :

```
sudo reboot                                # redémarrer pour appliquer
```

Résultat attendu : après redémarrage, la VM répond au `ping` depuis le nœud sur son adresse fixe. `/etc/network/interfaces` lit tous les fichiers de `/etc/network/interfaces.d/` grâce à sa ligne `source`.

Étape 4 — Se connecter depuis le nœud

Dans le Shell du nœud (interface web → votre nœud → **Shell**), puis dans la VM :

```
ssh devops@10.<site>.<N>.54                # mot de passe : dawan
hostname                                   # le nom de la VM
ip -br a show ens18                        # son adresse
```

Ce support est la propriété de Dawan France.

```
df -h /                                          # son disque
exit                                             # retour au nœud
```

Résultat attendu :

```
vm-debian
ens18            UP             10.59.1.54/24 fe80::be24:11ff:fe98:4e62/64
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
/dev/sda1           19G    1,3G   17G   8% /
```

Le système installé occupe 1,3 Go. L'adresse IPv6 n'apparaît que si votre réseau en distribue.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm status 105                                    # la VM tourne
qm config 105   # démarrage, disque, carte ; à lire : boot, scsi0, net0
ping -c 2 10.<site>.<N>.54                        # elle répond à son adresse fixe
```

Résultat attendu (les lignes qui comptent) : `status: running`, `boot: order=scsi0;ide2;net0`, le disque de 20 Gio et la carte VirtIO, puis deux réponses au `ping`.

À garder : la VM 105 reste allumée pour l'atelier 3.

Atelier 3 — Console et cycle de vie de la VM

Contexte

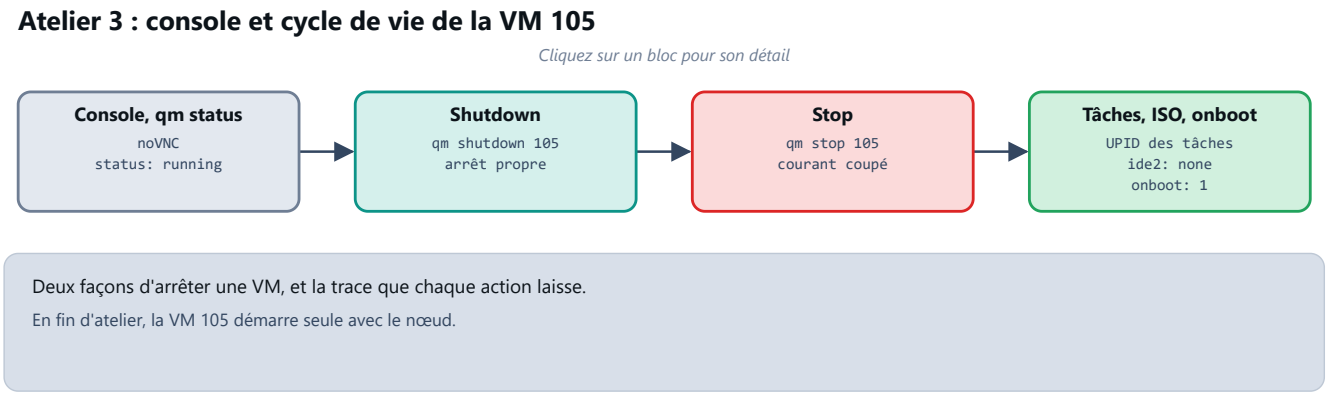
Une VM se démarre, s'arrête proprement ou brutalement, redémarre, et chaque action laisse une trace. Vous pilotez la VM 105 par l'interface puis par `qm`, vous lisez l'historique des tâches, et vous terminez son installation : retirer l'ISO, la faire démarrer avec le nœud.

Cours : §5 (console et cycle de vie).

Résultat à obtenir : la différence entre **Shutdown** et **Stop** constatée sur la VM 105, et ses dernières tâches lues avec leur identifiant (UPID : *Unique Process ID*). L'ISO est retirée du lecteur

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

CD, et la VM est réglée pour démarrer avec le nœud.



Le schéma oppose l'arrêt propre et l'arrêt brutal, entre la lecture de l'état et les réglages finaux. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
VM 105 « vm-debian »
├── ide2   : none, media=cdrom    ← l'ISO retirée
└── onboot : 1                    ← démarre avec le nœud
```

Ce que vous devez faire

1. **Ouvrir la console** noVNC de la VM 105, puis lire son état avec `qm status`.
2. **Arrêter proprement** la VM (**Shutdown** / `qm shutdown`), la redémarrer, puis **l'arrêter brutalement** (**Stop** / `qm stop`) : quelle différence pour le système invité ?
3. **Lire l'historique** des tâches de la VM : dans l'interface (panneau **Tasks**) et en ligne de commande.
4. **Retirer l'ISO** du lecteur CD et **activer le démarrage automatique** (`onboot`).

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe212` , VM 105.

Étape 1 — L'état de la VM

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm status 105              # en marche ou arrêtée
qm status 105 --verbose    # le détail ; à lire : status, qmpstatus, uptime, pid, name, maxmem, cpus
```

Résultat attendu (les lignes qui comptent) :

```
status: running
cpus: 2
maxmem: 2147483648
name: vm-debian
pid: 9151
qmpstatus: running
status: running
uptime: 61
```

`pid` est le processus QEMU de la VM sur le nœud (cours §2.3) ; `uptime` compte les secondes depuis son démarrage.

Étape 2 — Shutdown, puis Stop

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
time qm shutdown 105   # arrêt propre : le système invité ferme ses services
qm status 105          # l'état de la VM
qm start 105           # la redémarrer
time qm stop 105       # arrêt brutal : comme couper le courant
qm start 105           # la redémarrer
```

Ce support est la propriété de Dawan France.

Résultat attendu : `qm shutdown` a rendu la main en `1,08` s puis `status: stopped` ; `qm stop` en `0,60` s.

Les deux durées se ressemblent, mais pas ce qui se passe dans la VM. **Shutdown** demande au système de s'arrêter (signal ACPI, ou l'agent s'il est installé) : Debian ferme ses services et écrit ses fichiers. **Stop** coupe le processus QEMU, comme on débranche une prise : à réserver à une VM qui ne répond plus.

Dans l'interface, ce sont les boutons **Shutdown** et **Stop** de la VM ; **Reboot** (`qm reboot 105`) enchaîne un arrêt propre et un démarrage.

Étape 3 — L'historique des tâches

Chaque action laisse une tâche : panneau **Tasks** en bas de l'interface, ou onglet **Task History** de la VM. En ligne de commande :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesh get /nodes/$(hostname)/tasks --limit 4 --noborder 1   # les 4 dernières tâches du nœud
```

Résultat attendu (extrait) :

```
ID  Node    PID pstart           Starttime Type       UPID
User                 Endtime Status
105 pxe212 9451 225154 2026-10-06 12:56:08 qmstart
UPID:pxe212:000024EB:00036F82:6AC4D3C8:qmstart:105:root@pam:    root@pam 2026-10-
06 12:56:08 OK
105 pxe212 9428 225003 2026-10-06 12:56:06 qmshutdown
UPID:pxe212:000024D4:00036EEB:6AC4D3C6:qmshutdown:105:root@pam: root@pam 2026-10-
06 12:56:07 OK
```

L'**UPID** identifie la tâche : nœud, processus, heure, type (`qmstart` , `qmshutdown`), VM et utilisateur. C'est lui qu'on cite quand on cherche la cause d'un échec (cours §5.3).

Étape 4 — Retirer l'ISO, démarrer avec le nœud

L'installation est faite : l'ISO n'a plus rien à faire dans le lecteur. VM 105 → **Hardware** → **CD/DVD Drive** → **Edit** → **Do not use any media** ; puis **Options** → **Start at boot** → **Yes**. En

Dawan Ce support est la propriété de Dawan France.

ligne de commande :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm set 105 --ide2 none,media=cdrom   # vider le lecteur CD
qm set 105 --onboot 1                # démarrer avec le nœud
qm config 105   # vérifier ; à lire : boot, ide2, onboot
```

Résultat attendu (les lignes qui comptent) :

```
update VM 105: -ide2 none,media=cdrom
update VM 105: -onboot 1
boot: order=scsi0;ide2;net0
ide2: none,media=cdrom
onboot: 1
```

Une ISO laissée dans le lecteur bloque la migration de la VM vers un autre nœud : on la retire dès l'installation finie.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm status 105                              # la VM tourne
qm config 105   # lecteur vide, démarrage avec le nœud ; à lire : ide2, onboot
```

Résultat attendu (les lignes qui comptent) : `status: running`, `ide2: none,media=cdrom`, `onboot: 1`.

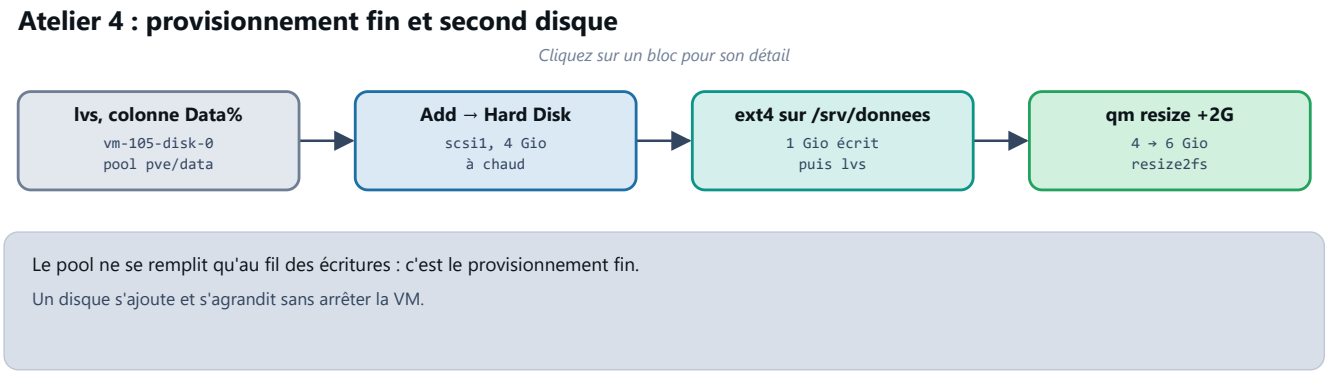
Atelier 4 — Le stockage de la VM : provisionnement fin et second disque

Contexte

Le disque de 20 Gio de la VM 105 n'occupe sur `local-lvm` que ce qu'elle a réellement écrit : c'est le provisionnement fin (*thin provisioning*). Vous le mesurez, vous ajoutez à la VM un second disque pour ses données, vous le remplissez, puis vous l'agrandissez sans arrêter la VM.

Cours : §6 (stockage LVM-thin).

Résultat à obtenir : le remplissage réel des disques de la VM 105 lu avec `lvs`, et un second disque de 4 Gio ajouté, formaté et monté sur `/srv/donnees`. Ce disque est ensuite agrandi à 6 Gio à chaud, système de fichiers compris.



Le schéma suit les disques de la VM 105, de la lecture de `lvs` à l'agrandissement. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
VM 105 « vm-debian »
├── scsi0 : vm-105-disk-0, 20 Gio   ← système
└── scsi1 : vm-105-disk-1, 4 Gio, puis 6 Gio
            └── /srv/donnees (ext4)
```

Ce que vous devez faire

1. **Lire** l'occupation réelle du disque de la VM avec `lvs` (colonne `Data%`).
2. **Ajouter** un disque de 4 Gio à la VM, à chaud (**Hardware** → **Add** → **Hard Disk**).
3. Dans la VM, **le formater** en `ext4`, le monter sur `/srv/donnees`, y écrire 1 Gio, puis relire `lvs`.
4. **L'agrandir** de 2 Gio à chaud, puis étendre son système de fichiers.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe211` , VM 105 en `10.59.1.54` (lab sur le plan des stagiaires) sur `pxe212` .

Étape 1 — Ce que le disque occupe vraiment

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
lvs pve/vm-105-disk-0   # le disque système de la VM
```

Résultat attendu :

```
  LV            VG  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync
Convert
  vm-105-disk-0 pve Vwi-aotz-- 20.00g data        12.75
```

Le disque mesure 20 Gio pour la VM (`LSize`), mais n'occupe que 12,75 % de cette taille dans le pool `data` : les blocs que Debian a réellement écrits.

Étape 2 — Ajouter un second disque

VM 105 → **Hardware** → **Add** → **Hard Disk** : Bus `SCSI` , Storage `local-lvm` , Disk size `4` , **Discard** coché. Ou :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm set 105 --scsi1 local-lvm:4,discard=on,iothread=1   # 4 Gio, à chaud
```

Résultat attendu :

```
update VM 105: -scsi1 local-lvm:4,discard=on,iothread=1
  Logical volume "vm-105-disk-1" created.
scsi1: successfully created disk 'local-lvm:vm-105-disk-
1,discard=on,iothread=1,size=4G'
```

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

`discard=on` permet à la VM de signaler les blocs qu'elle libère, pour que le pool les récupère.

Étape 3 — Le formater, le monter, le remplir

Dans le Shell du nœud (interface web → votre nœud → **Shell**), puis dans la VM :

```
ssh devops@10.<site>.<N>.54                          # mot de passe : dawan
lsblk -o NAME,SIZE,TYPE,MOUNTPOINTS                  # le nouveau disque : sdb
sudo mkfs.ext4 -q -L donnees /dev/sdb                # un système de fichiers sur tout
le disque
sudo mkdir -p /srv/donnees                           # le point de montage
sudo mount /dev/sdb /srv/donnees                     # le monter
sudo dd if=/dev/urandom of=/srv/donnees/essai.bin bs=1M count=1000 status=none
# 1 Gio de données
sync                                                 # tout écrire sur le disque
df -h /srv/donnees                                   # l'espace occupé
exit                                                 # retour au nœud
```

Résultat attendu — `lsblk`, puis `df` :

```
NAME    SIZE TYPE MOUNTPOINTS
sda      20G disk
├─sda1 18,9G part /
├─sda2    1K part
└─sda5  1,1G part [SWAP]
sdb       4G disk
sr0    1024M rom
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
/dev/sdb           3,9G   1002M  2,7G  27% /srv/donnees
```

Puis, sur le nœud, `lvs pve/vm-105-disk-1 pve/data` : le nouveau disque est passé à `26.07` % et le pool entier à `30.29` %. Le pool ne se remplit qu'au fil des écritures des VM : c'est pourquoi on le surveille (cours §6.2).

Étape 4 — Agrandir à chaud

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
qm resize 105 scsi1 +2G   # le disque passe de 4 à 6 Gio, VM en marche
```

Résultat attendu :

```
  Size of logical volume pve/vm-105-disk-1 changed from 4.00 GiB (1024 extents)
to 6.00 GiB (1536 extents).
  Logical volume pve/vm-105-disk-1 successfully resized.
```

Dans la VM, `lsblk /dev/sdb` voit aussitôt 6 Gio, mais `df` affiche toujours 3,9 Gio : seul le disque a grandi, pas le système de fichiers (cours §6.4). On l'étend, toujours à chaud.

Dans le Shell du nœud, puis dans la VM :

```
ssh devops@10.<site>.<N>.54                # mot de passe : dawan
sudo resize2fs /dev/sdb                    # étendre le système de fichiers
df -h /srv/donnees                         # la nouvelle taille
exit                                       # retour au nœud
```

Résultat attendu :

```
resize2fs 1.47.2 (1-Jan-2025)
Filesystem at /dev/sdb is mounted on /srv/donnees; on-line resizing required
The filesystem on /dev/sdb is now 1572864 (4k) blocks long.
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
/dev/sdb           5,9G   1002M  4,6G  18% /srv/donnees
```

Ce disque de données a son système de fichiers directement sur le disque, sans partition : une seule commande suffit. Sur un disque partitionné, il faut d'abord agrandir la partition (`growpart`, cours §6.4).

> **Le montage ne survit pas au redémarrage.** `mount` vaut jusqu'au prochain arrêt. Pour le rendre permanent, on ajoute une ligne dans `/etc/fstab` : `LABEL=donnees /srv/donnees ext4 defaults,nofail 0 2`.

Validation finale

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm config 105 | grep -E "^scsi"            # les deux disques de la VM
lvs pve/vm-105-disk-0 pve/vm-105-disk-1    # leurs volumes LVM-thin
```

Résultat attendu : `scsi0` de 20 Gio et `scsi1` de 6 Gio ; deux volumes dont `Data%` reste bien inférieur à 100 %.

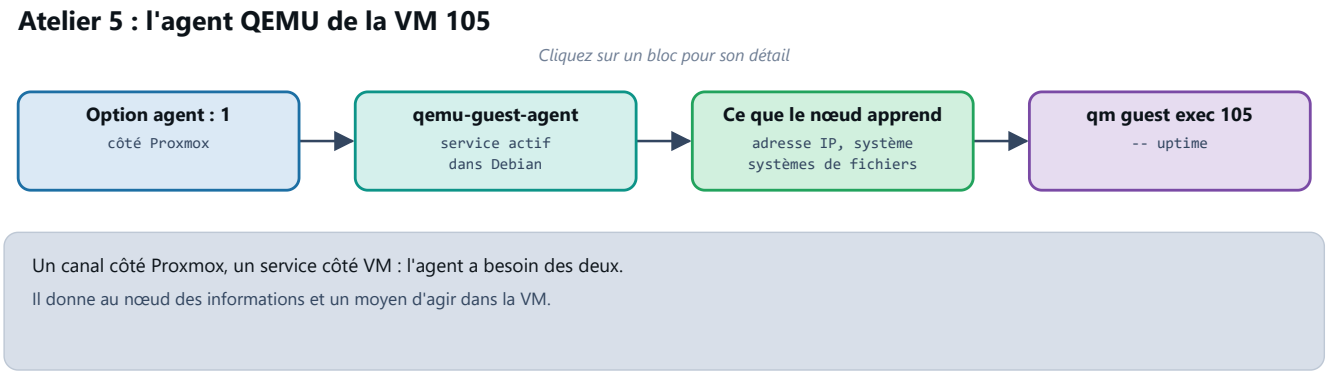
Atelier 5 — L'agent QEMU

Contexte

L'agent invité est un petit programme installé **dans** la VM, qui répond à Proxmox par un canal dédié. Avec lui, Proxmox connaît l'adresse IP de la VM, l'arrête proprement et peut figer ses fichiers pendant une sauvegarde. Sur la VM 105, l'installateur Debian l'a déjà installé : vous vérifiez ce qu'il apporte.

Cours : §7.2 (agent QEMU).

Résultat à obtenir : l'agent de la VM 105 répond ; son adresse IP, son système et l'état de ses fichiers sont lus depuis le nœud. Une commande est exécutée dans la VM sans SSH.



Le schéma part du canal et aboutit à ce que l'agent permet. Cliquez sur chaque bloc pour la commande correspondante.

Arborescence attendue

Ce support est la propriété de Dawan France.

```
VM 105 « vm-debian »
├── option agent : 1                    ← le canal, côté Proxmox (atelier 2)
└── qemu-guest-agent (service actif)    ← le programme, dans Debian
```

Ce que vous devez faire

1. **Vérifier** que l'option agent est active et que l'agent répond (`qm agent 105 ping`).
2. **Vérifier** dans la VM que le paquet `qemu-guest-agent` est installé et son service actif.
3. **Lire** depuis le nœud l'adresse IP de la VM, son système et l'état de ses systèmes de fichiers ; regarder la ligne **IPs** de l'onglet **Summary**.
4. **Exécuter** une commande dans la VM depuis le nœud, avec `qm guest exec`.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe212` , VM 105.

Étape 1 — L'agent répond-il ?

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm config 105 | grep ^agent   # le canal est-il ouvert côté Proxmox ?
qm agent 105 ping             # l'agent répond-il ?
echo "code : $?"              # son code
```

Résultat attendu : `agent: 1` , puis `code : 0` : l'agent répond.

Étape 2 — Dans la VM

Dans le Shell du nœud, puis dans la VM :

```
ssh devops@10.<site>.<N>.54                  # mot de passe : dawan
systemctl is-active qemu-guest-agent         # le service
dpkg -l qemu-guest-agent                     # le paquet
exit                                         # retour au nœud
```

Résultat attendu : `active` , et le paquet `qemu-guest-agent` en version `1:10.0.13+ds-0+deb13u1` , installé par l'installateur Debian : il l'ajoute de lui-même quand il détecte une machine virtuelle QEMU. Vous ne l'avez pas installé. Une **image cloud**, au contraire, n'a pas l'agent : vous le constaterez à l'atelier 8 ; il s'y installe avec `apt install qemu-guest-agent` , comme ici.

Étape 3 — Ce que Proxmox apprend grâce à l'agent

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm guest cmd 105 network-get-interfaces | grep -E '"ip-address"|"name"'   # les adresses de la VM
qm guest cmd 105 get-osinfo | grep pretty                                  # le système invité
```

Dawan Nous formons votre savoir
Ce support est la propriété de Dawan France.

```
qm guest cmd 105 fsfreeze-status                                    #
fichiers figés ou non
```

Résultat attendu :

```
            "ip-address" : "127.0.0.1",
            "ip-address" : "::1",
      "name" : "lo",
            "ip-address" : "10.59.1.54",
            "ip-address" : "fe80::be24:11ff:fe98:4e62",
      "name" : "ens18",
   "pretty-name" : "Debian GNU/Linux 13 (trixie)",
thawed
```

Ce sont ces adresses que l'onglet **Summary** affiche sur la ligne **IPs**. `thawed` : les systèmes de fichiers ne sont pas figés. Une sauvegarde ou un instantané les fige un instant, pour copier un disque cohérent (atelier 6).

Étape 4 — Une commande dans la VM, sans SSH

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm guest exec 105 -- uptime   # exécuté par l'agent, dans la VM
```

Résultat attendu :

```
{
   "exitcode" : 0,
   "exited" : 1,
   "out-data" : " 12:58:19 up 0 min,  0 users,  load average: 0,00, 0,00,
0,00\n",
   "out-truncated" : 0
}
```

Avec l'agent, `qm shutdown` passe aussi par lui plutôt que par le signal ACPI, `1,34` s pour un arrêt propre.

Ce support est la propriété de Dawan France.

Validation finale

Dans le Shell du nœud :

```
qm agent 105 ping              # l'agent répond-il ?
echo "code : $?"               # son code
```

Résultat attendu : `code : 0`, et l'adresse `10.<site>.<N>.54` sur la ligne **IPs** du **Summary**.

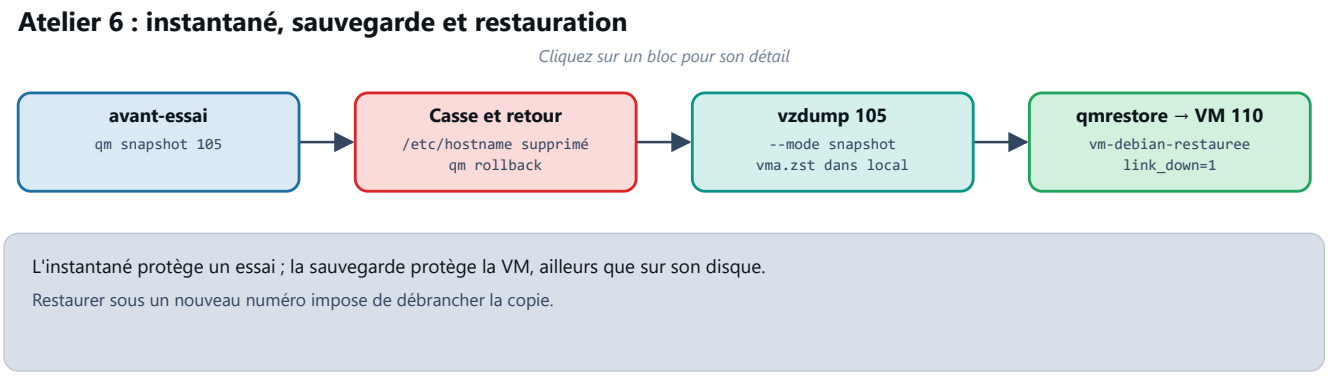
Atelier 6 — Instantanés et sauvegarde de la VM

Contexte

Avant une intervention risquée, on prend un **instantané** (*snapshot*) pour revenir en arrière en quelques secondes. Pour se protéger d'une panne du disque, on fait une **sauvegarde** : une archive complète, ailleurs. Vous faites les deux sur la VM 105, puis vous restaurez la sauvegarde sous un nouveau numéro, sans écraser l'originale.

Cours : §8 (instantanés) et §9 (sauvegardes).

Résultat à obtenir : un instantané de la VM 105, restauré après une casse volontaire. Une sauvegarde `vzdump` de la VM 105 est aussi restaurée en VM 110 avec sa carte réseau débranchée, pour ne pas entrer en conflit d'adresse avec l'originale.



Les deux premiers blocs portent sur l'instantané, les deux suivants sur la sauvegarde. Cliquez sur chaque bloc pour la commande correspondante.

Arborescence attendue

Dawan Ce support est la propriété de Dawan France.

```
VM 105 « vm-debian »
└── instantané « avant-essai »
local → Backups
└── vzdump-qemu-105-AAAA_MM_JJ-HH_MM_SS.vma.zst
VM 110 « vm-debian-restauree »   ← restaurée, carte réseau débranchée
```

Ce que vous devez faire

1. **Prendre un instantané** de la VM 105, nommé `avant-essai`.
2. **Casser** volontairement la VM (supprimer `/etc/hostname`), puis **revenir** à l'instantané et vérifier que le fichier est revenu.
3. **Sauvegarder** la VM 105 avec `vzdump` sur le stockage `local`, en mode `snapshot`.
4. **Restaurer** cette sauvegarde en **VM 110**, la renommer, **débrancher sa carte réseau**, la démarrer et constater pourquoi.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe211` , VM 105 en `10.59.1.54` (lab sur le plan des stagiaires) sur `pxe212` .

Étape 1 — L'instantané

VM 105 → **Snapshots** → **Take Snapshot**, nom `avant-essai` . Ou :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm snapshot 105 avant-essai --description "VM propre avant essai"   # l'instantané
qm listsnapshot 105                                                 # la liste des instantanés
```

Résultat attendu :

```
freeze guest filesystem
snapshotting 'drive-scsi0' (local-lvm:vm-105-disk-0)
  Logical volume "snap_vm-105-disk-0_avant-essai" created.
snapshotting 'drive-scsi1' (local-lvm:vm-105-disk-1)
  Logical volume "snap_vm-105-disk-1_avant-essai" created.
thaw guest filesystem
`-> avant-essai                 2026-10-06 12:59:16     VM propre avant essai
 `-> current                                            You are here!
```

`freeze` puis `thaw` : l'agent (atelier 5) a figé les fichiers le temps de l'instantané. Les deux disques sont pris ensemble.

Étape 2 — Casser, puis revenir

Dans le Shell du nœud :

```
ssh devops@10.<site>.<N>.54     # mot de passe : dawan
sudo rm /etc/hostname           # la casse
cat /etc/hostname               # le fichier a disparu
```

Ce support est la propriété de Dawan France.

```
exit                              # retour au nœud
qm rollback 105 avant-essai       # retour à l'instantané : la VM s'arrête
qm status 105                     # stopped
qm start 105                      # la redémarrer
```

Dans le Shell du nœud, trente secondes plus tard, le temps que la VM démarre :

```
qm guest exec 105 -- cat /etc/hostname   # le fichier est revenu
```

Résultat attendu : `cat: /etc/hostname: Aucun fichier ou dossier de ce nom`, puis `status: stopped` après le retour arrière, et enfin `"out-data" : "vm-debian\n"` : le fichier est revenu.

Étape 3 — La sauvegarde

VM 105 → **Backup** → **Backup now** : Storage `local`, Mode `Snapshot`, Compression `ZSTD`. Ou :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
vzdump 105 --storage local --mode snapshot --compress zstd   # la sauvegarde
pvesm list local --content backup                            # le fichier produit
```

Résultat attendu (extrait) :

```
INFO: Starting Backup of VM 105 (qemu)
INFO: backup mode: snapshot
INFO: creating vzdump archive '/var/lib/vz/dump/vzdump-qemu-105-2026_10_06-
12_59_52.vma.zst'
INFO: transferred 26.00 GiB in 11 seconds (2.4 GiB/s)
INFO: archive file size: 2.09GB
INFO: Finished Backup of VM 105 (00:00:12)
Volid                                                  Format  Type
Size VMID
local:backup/vzdump-qemu-105-2026_10_06-12_59_52.vma.zst vma.zst backup
2249067266 105
```

Ce support est la propriété de Dawan France.

26 Gio de disques (20 + 6) donnent une archive de 2,09 Go : seuls les blocs écrits sont sauvegardés, puis compressés. La VM a tourné pendant toute la sauvegarde.

Étape 4 — Restaurer sous un nouveau numéro

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qmrestore /var/lib/vz/dump/vzdump-qemu-105-AAAA_MM_JJ-HH_MM_SS.vma.zst 110 \
  --storage local-lvm --unique 1     # nouvelle VM 110 ; --unique : nouvelle adresse MAC
qm config 110   # ce qu'elle a hérité ; à lire : name, net0, onboot
```

Résultat attendu (les lignes qui comptent) :

```
progress 100% (read 27917287424 bytes, duration 6 sec)
total bytes read 27917287424, sparse bytes 24502398976 (87.8%)
name: vm-debian
net0: virtio=BC:24:11:C7:0F:53,bridge=vmbr0,firewall=1
onboot: 1
```

La copie est parfaite, et c'est le problème : elle porte le même nom, démarre avec le nœud, et son système contient la **même adresse fixe** que la VM 105. Démarrée telle quelle, elle provoquerait un conflit d'adresse. On la renomme et on débranche sa carte avant de la démarrer.

Dans le Shell du nœud :

```
qm set 110 --name vm-debian-restauree --net0 virtio,bridge=vmbr0,link_down=1 --onboot 0   # renommer, débrancher
qm start 110                                          # la démarrer
```

Dans le Shell du nœud, trente-cinq secondes plus tard :

```
qm guest exec 110 -- ip -br a show ens18              # sa carte, DOWN
```

Résultat attendu :

Ce support est la propriété de Dawan France.

```
update VM 110: -name vm-debian-restauree -net0 virtio,bridge=vmbr0,link_down=1 -
onboot 0
   "out-data" : "ens18            DOWN           10.59.1.54/24 \n",
```

La VM 110 porte bien l'adresse de la VM 105, mais sa carte est `DOWN` : aucun conflit. Pour l'utiliser vraiment, il faudrait d'abord changer son adresse dans le fichier qui la déclare (`/etc/network/interfaces` ou `interfaces.d/ens18`, voir l'atelier 9, étape 9), puis rebrancher la carte (`link_down=0`).

Validation finale

Dans le Shell du nœud :

```
qm listsnapshot 105                 # l'instantané avant-essai
pvesm list local --content backup   # la sauvegarde
qm list   # la VM et sa restauration ; à lire : 105, 110
```

Résultat attendu (les lignes qui comptent) : l'instantané `avant-essai`, l'archive `vzdump-qemu-105-…`, puis les VM 105 `vm-debian` et 110 `vm-debian-restauree`.

Nettoyage : arrêtez la VM 110 (`qm stop 110`) ; supprimez-la si la mémoire du nœud manque (`qm destroy 110`). La VM 105, son instantané et la sauvegarde restent.

Atelier 7 — Créer une VM template depuis une image cloud

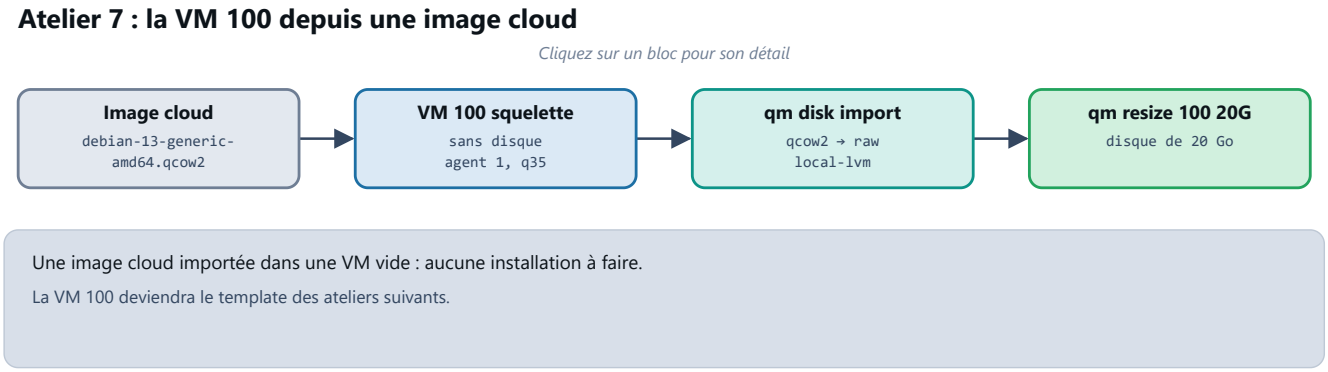
Contexte

Installer un système dans chaque VM à la main est long et source de divergences. La méthode professionnelle part d'une **image cloud** — un disque déjà installé, nettoyé et équipé de cloud-init, publié par la distribution elle-même. On la transforme une fois en template, puis on clone en quelques secondes.

Cet atelier pose la première brique : récupérer l'image cloud Debian 13 et bâtir la VM 100 qui deviendra ce template. Vous ne l'installez pas — vous importez un disque déjà prêt.

Ce support est la propriété de Dawan France.

Résultat à obtenir : une VM 100 nommée `debian13-template`, dotée d'un disque de 20 Go importé depuis l'image cloud Debian 13, prête à être configurée par cloud-init à l'atelier suivant.



Le schéma suit l'image cloud jusqu'au disque de 20 Go de la VM 100. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
Sur le nœud :
/var/lib/vz/template/iso/      (stockage « local »)
└── debian-13-generic-amd64.qcow2        ← image cloud téléchargée

VM 100 « debian13-template »   ← squelette sans disque, puis disque importé
```

Ce que vous devez faire

1. **Télécharger l'image cloud** Debian 13 (format `.qcow2`) sur le nœud.
2. **Créer la VM 100 « squelette »** depuis l'interface — sans disque, avec les réglages adaptés à un template (agent invité activé, contrôleur adéquat).
3. **Importer le disque cloud** dans la VM 100, puis l'attacher. Pensez à l'**agrandir** à 20 Go : les images cloud sont volontairement minuscules et démarreraient sinon presque pleines.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

L'approche cloud-init évite une installation manuelle de 15 minutes : on part d'une image Debian déjà installée, pré-configurée avec cloud-init, personnalisée au premier démarrage.

Étape 1 — Télécharger l'image cloud Debian 13

L'image vient du site officiel de Debian :

- 📖 Images cloud officielles de Debian : toutes les versions et tous les formats ;
- 📖 Debian 13 (trixie), dernière version : le dossier d'où vient `debian-13-generic-amd64.qcow2`, avec son fichier d'empreintes `SHA512SUMS`.

L'image `generic` convient à Proxmox : elle contient cloud-init et les pilotes VirtIO. Copiez l'adresse du fichier `.qcow2` depuis cette page, ou reprenez-la dans la commande ci-dessous.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Télécharger l'image cloud Debian 13 dans le dossier du stockage « local »
wget -O /var/lib/vz/template/iso/debian-13-generic-amd64.qcow2 \
  https://cloud.debian.org/images/cloud/trixie/latest/debian-13-generic-amd64.qcow2

# Vérifier sa présence et sa taille
ls -lh /var/lib/vz/template/iso/debian-13-generic-amd64.qcow2
```

Résultat attendu : `414M`. C'est la taille du **fichier** ; le disque qu'il contient est plus grand, comme le montre `qemu-img` :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qemu-img info /var/lib/vz/template/iso/debian-13-generic-amd64.qcow2   # taille virtuelle et taille réelle de l'image ; à lire : virtual size, disk size
```

```
virtual size: 3 GiB (3221225472 bytes)
disk size: 414 MiB
```

Le format qcow2 ne stocke que les blocs écrits : 414 Mio de données pour un disque de 3 Gio.

Ce support est la propriété de Dawan France.

> **Si le réseau est lent :** l'image a peut-être été téléchargée à l'avance par le formateur. Vérifiez avec `ls /var/lib/vz/template/iso/*.qcow2` avant de relancer le téléchargement.

Étape 2 — Créer la VM squelette depuis l'interface

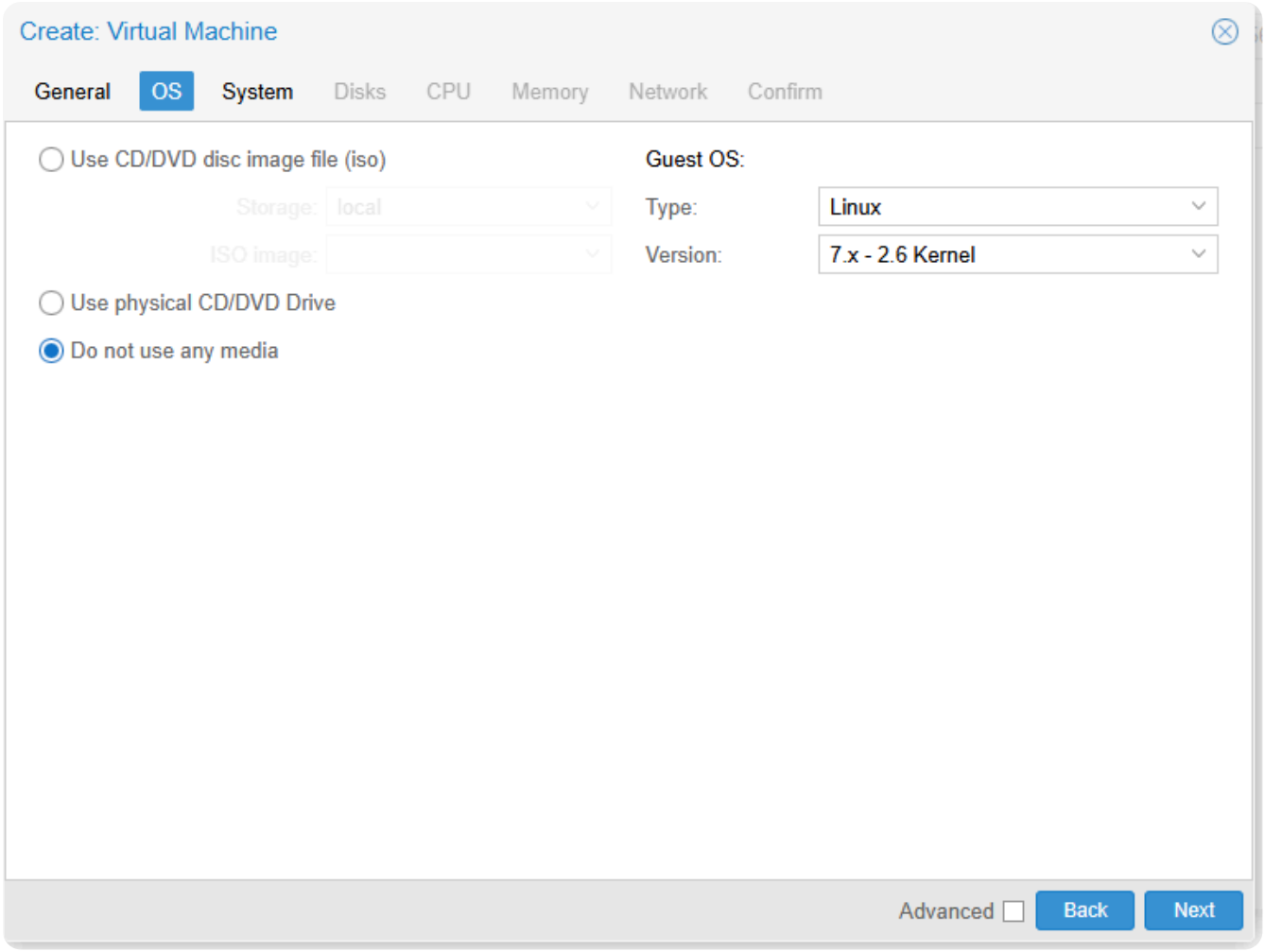
Create VM en haut à droite :

Onglet	Champ	Valeur
General	VM ID / Name	`100` / `debian13-template`
OS	Do not use any media	✓ (on importera le disque après)
System	Machine / SCSI Controller / Qemu Agent	`q35` / `VirtIO SCSI single` / ✓
Disks	—	supprimer le disque proposé par défaut (🗑️)
CPU	Sockets / Cores / Type	`1` / `2` / `x86-64-v2-AES`
Memory	Memory / **Advanced** ✓ → Minimum memory	`2048` Mio / `512` Mio
Network	Bridge / Model	`vmbr0` / `VirtIO (paravirtualized)`

Vérifiez le récapitulatif dans **Confirm** → **Finish**.

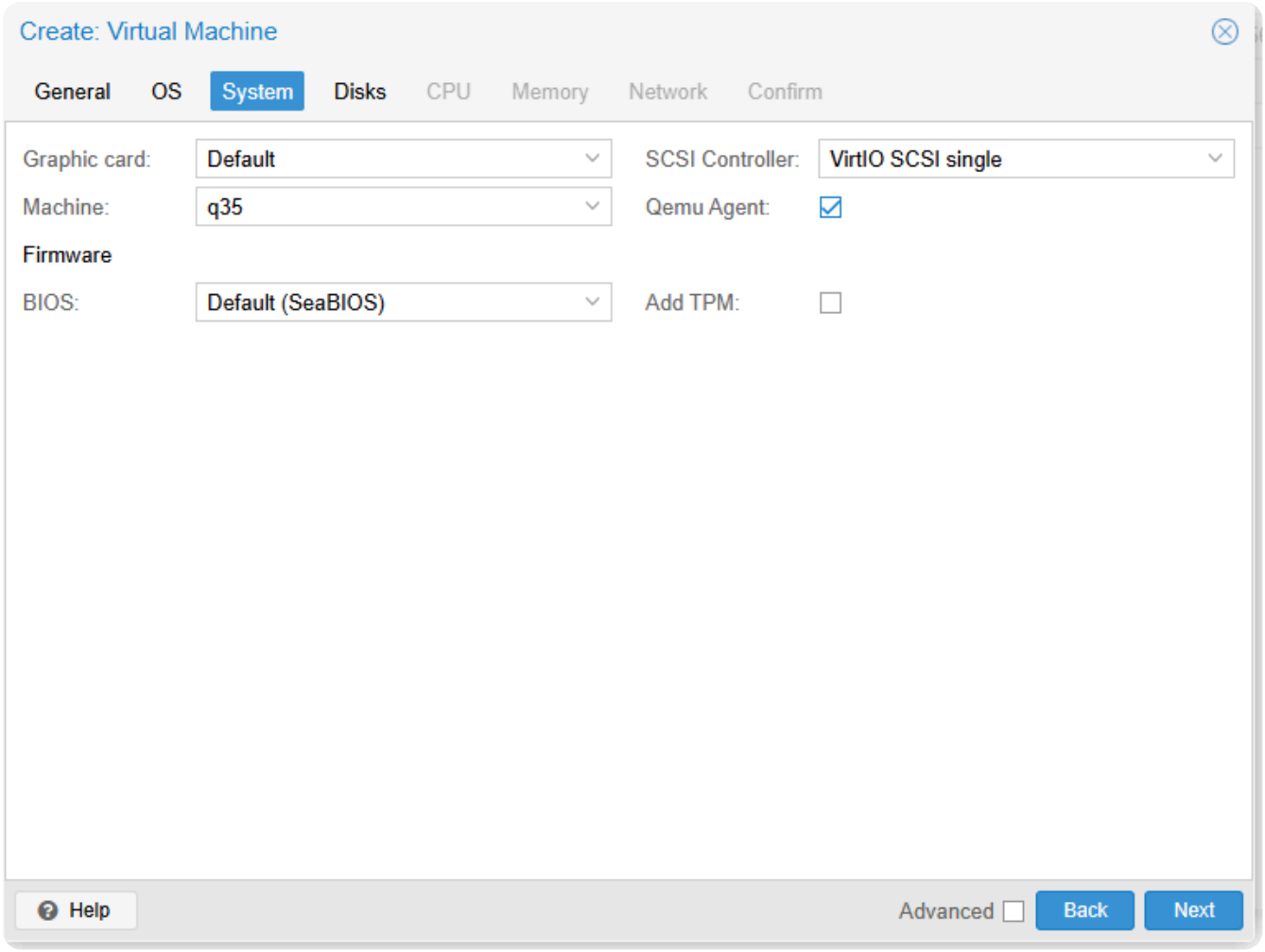
> **Machine d'exemple : nœud** `pxe211` (Proxmox VE 9.2.20). Les captures qui suivent viennent de l'assistant **Create VM** de ce test sur le lab.

Ce support est la propriété de Dawan France.



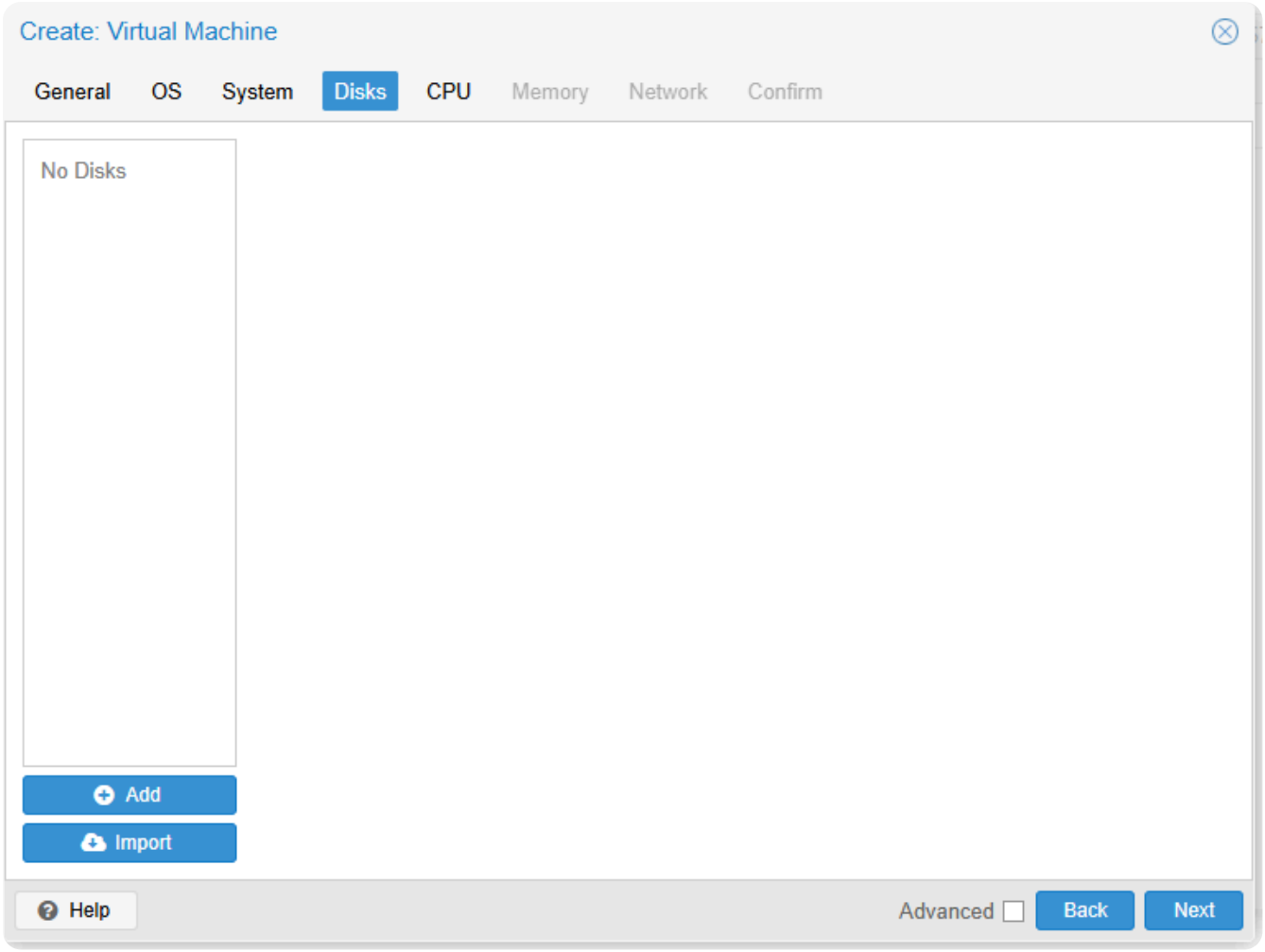
Do not use any media : la VM ne démarrera sur aucune ISO, puisqu'on lui donnera un disque déjà installé. Le type de système reste **Linux 7.x - 2.6 Kernel**, celui des distributions Linux actuelles.

Ce support est la propriété de Dawan France.



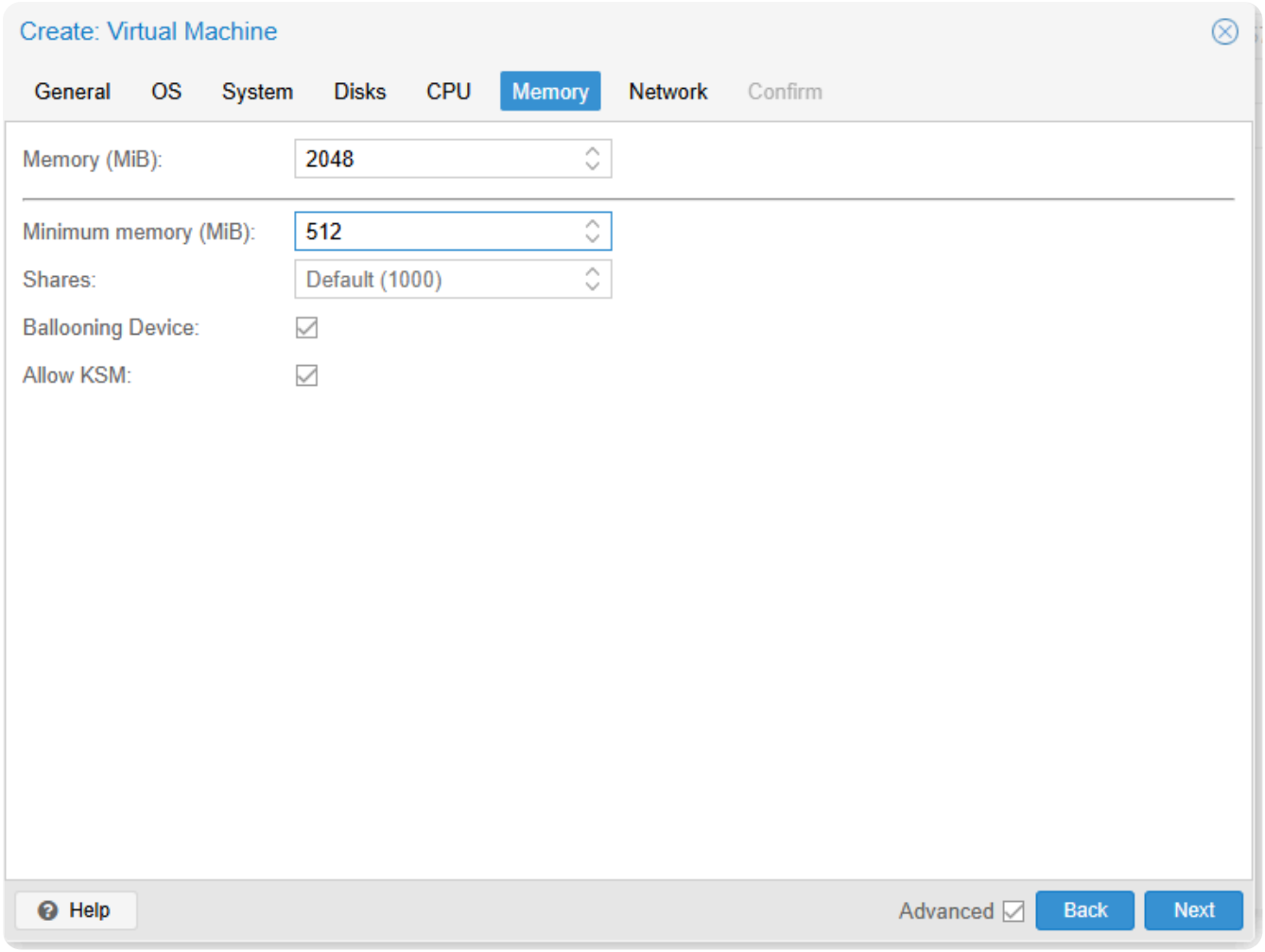
Machine `q35`, le chipset moderne (PCI Express) ; **SCSI Controller** déjà sur **VirtIO SCSI single** ; **Qemu Agent** coché : le canal est prêt pour l'agent. Le BIOS reste **SeaBIOS** : l'image cloud Debian démarre en BIOS comme en UEFI.

Ce support est la propriété de Dawan France.



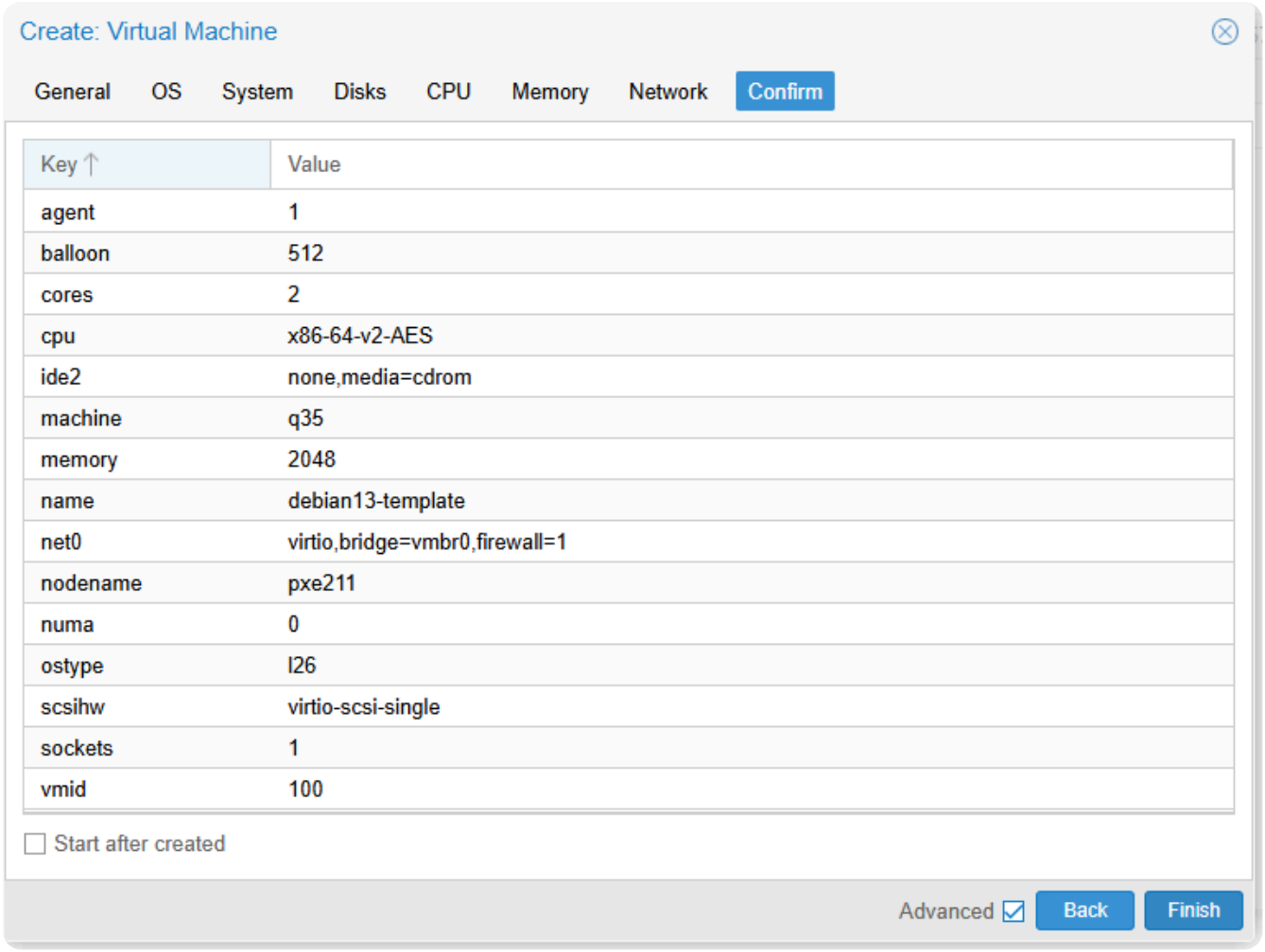
Le disque `scsi0` proposé par défaut a été supprimé (icône corbeille) : la liste est vide. Remarquez le bouton **Import**, nouveau en Proxmox VE 9 : il importe une image disque directement dans l'assistant, à condition qu'elle se trouve dans le contenu *Import* d'un stockage. Cet atelier utilise la méthode en ligne de commande, qui marque mieux chaque étape.

Ce support est la propriété de Dawan France.



Memory `2048` Mio, et, une fois **Advanced** coché, **Minimum memory** `512` Mio : c'est le plancher du ballooning (`balloon` en ligne de commande). **Ballooning Device** est coché par défaut.

Ce support est la propriété de Dawan France.



Le récapitulatif réellement envoyé : `agent 1`, `balloon 512`, `cores 2`, `cpu x86-64-v2-AES`, `machine q35`, `memory 2048`, `scsihw virtio-scsi-single`. Deux lignes viennent de l'assistant et pas de la commande `qm create` ci-dessous : un lecteur `ide2` vide (`none,media=cdrom`) et `firewall=1` sur la carte réseau. **Start after created** est décoché : c'est voulu.

> **Ne démarrez pas la VM** — elle n'a pas encore de disque. Continuez à l'étape suivante.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI complet ; x86-64-v2-AES garde la VM migrable entre nœuds
qm create 100 --name debian13-template --memory 2048 --balloon 512 \
  --cores 2 --cpu x86-64-v2-AES --net0 virtio,bridge=vmbr0 \
  --machine q35 --ostype l26 --agent enabled=1
```

Ce support est la propriété de Dawan France.

Résultat attendu : aucun affichage ; la VM 100 apparaît dans l'arbre, sans disque. `--balloon 512` autorise Proxmox à reprendre de la mémoire à la VM jusqu'à ce plancher quand l'hôte en manque — l'agent et le pilote *balloon* de l'invité s'en chargent.

Étape 3 — Importer le disque cloud dans la VM

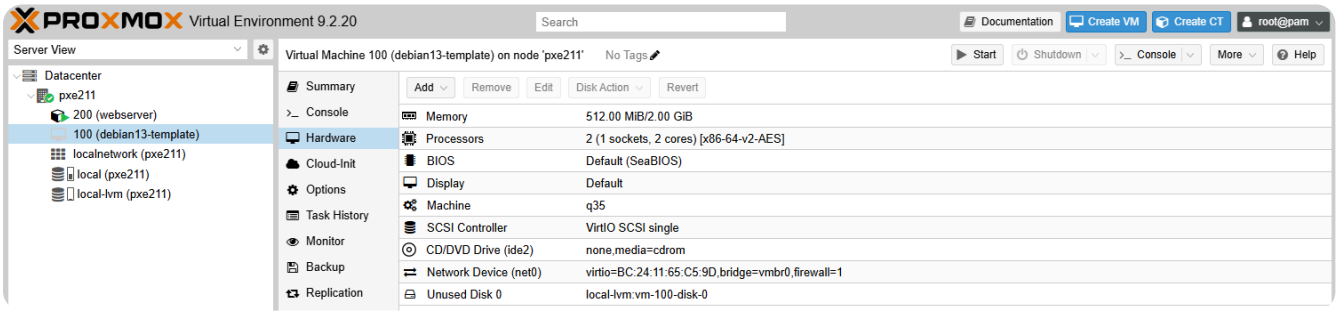
Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Convertir l'image qcow2 en volume du pool local-lvm, rattaché à la VM 100
qm disk import 100 \
  /var/lib/vz/template/iso/debian-13-generic-amd64.qcow2 \
  local-lvm
```

Résultat attendu (fin de la sortie) :

```
transferred 3.0 GiB of 3.0 GiB (100.00%)
unused0: successfully imported disk 'local-lvm:vm-100-disk-0'
```

Le qcow2 est converti en `raw` au passage, seul format que connaît LVM-thin : les 3 Gio sont transférés en entier. `qm importdisk`, l'ancien nom de la commande, fonctionne encore.



Le disque importé apparaît en **Unused Disk 0** : il existe sur `local-lvm`, mais n'est branché sur aucun contrôleur. La VM ne pourrait pas démarrer dessus. Notez aussi la ligne **Memory** `512.00 MiB/2.00 GiB` — le plancher et le maximum du ballooning.

Attachez-le : double-clic sur `Unused Disk 0` → Bus/Device `SCSI 0`, Discard ✓, IO thread ✓ → **Add**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Équivalent CLI : attacher le volume importé comme premier disque SCSI
qm set 100 --scsihw virtio-scsi-single \
  --scsi0 local-lvm:vm-100-disk-0,discard=on,iothread=1
```

Résultat attendu : `update VM 100: -scsi0 local-lvm:vm-100-disk-0,discard=on,iothread=1 -scsihw virtio-scsi-single`. Le contrôleur `virtio-scsi-single` donne un contrôleur par disque : c'est ce qui permet à `iothread=1` de dédier un fil d'exécution à ce disque.

Agrandir le disque (l'image cloud fait 3 Gio, on veut 20 Gio) : sélectionnez `scsi0` → **Disk Action** → **Resize** → Size Increment `17` (Gio).

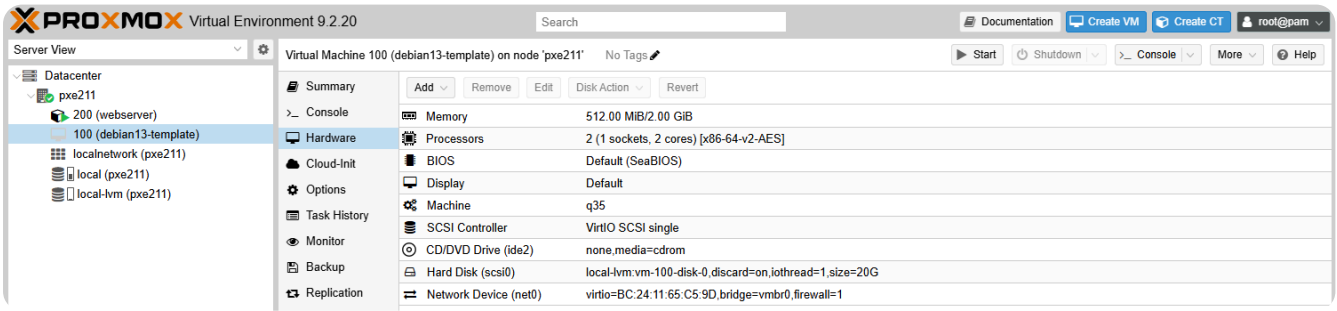
Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Taille absolue : le disque passe à 20 Gio, quelle que soit sa taille de départ
qm resize 100 scsi0 20G
```

Résultat attendu :

```
  Size of logical volume pve/vm-100-disk-0 changed from 3.00 GiB (768 extents) to 20.00 GiB (5120 extents).
  Logical volume pve/vm-100-disk-0 successfully resized.
```

Sans signe `+`, la taille est **absolue**. Avec `+18G`, on **ajouterait** 18 Gio aux 3 Gio de l'image, soit 21 Gio. Dans l'interface, le champ est toujours un **incrément**.



Le disque est devenu **Hard Disk (scsi0)** `local-lvm:vm-100-disk-0,discard=on,iothread=1,size=20G`, et la ligne *Unused Disk* a disparu.

Ce support est la propriété de Dawan France.

Pourquoi aucune commande à l'intérieur de la VM ici ? Parce que vous travaillez sur une **image cloud**.

Ces images embarquent cloud-init et son module de redimensionnement : au démarrage suivant, la VM constate que son disque a grandi et étend seule sa partition et son système de fichiers. Vérifiez-le après le prochain boot avec `df -h /`.

Ce ne sera pas le cas d'une VM installée depuis une ISO. Là, `qm resize` agrandit le disque virtuel et rien d'autre : la partition et le système de fichiers restent à leur taille d'origine. Il faut alors intervenir dans la VM, et l'outil nécessaire n'y est pas installé par défaut :

Dans la VM (sa **Console**, ou en SSH) :

```
# Dans la VM installee depuis une ISO
apt update                                  # la liste des paquets
apt install -y cloud-guest-utils            # fournit growpart
growpart /dev/sda 1                         # etendre la partition
resize2fs /dev/sda1                         # etendre le systeme de
fichiers
df -h /   # vérifier la nouvelle taille
```

Retenez les trois couches : **le disque virtuel** (agrandi par `qm resize`), **la partition** (`growpart`) et **le système de fichiers** (`resize2fs`). L'image cloud fait les deux dernières pour vous ; une installation classique, non.

Et si `growpart` répond `NOCHANGE: partition 1 could only be grown by 2046`, c'est que la partition visée n'est pas la dernière du disque : avec le partitionnement Debian par défaut, le swap s'intercale entre `/` et l'espace libre. Une partition ne grandit que sur l'espace immédiatement contigu. Deux issues.

Créer une partition dans l'espace libre et la monter là où la place manque. Ou supprimer le swap (`swapoff -a`, `sfdisk --delete /dev/sda 5`, puis `2`), agrandir, et remettre du swap sous forme de fichier. C'est exactement le problème que LVM supprime : sur un volume logique, `lvextend` n'a aucune contrainte de contiguïté.

Validation finale

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
ls -lh /var/lib/vz/template/iso/debian-13-generic-amd64.qcow2   # taille du fichier image téléchargé
qm config 100   # disque, contrôleur, chipset et agent de la VM 100 ; à lire : scsi0, scsihw, machine, agent
```

Résultat attendu (les lignes qui comptent) :

```
-rw-r--r-- 1 root root 414M Sep 14 16:30 /var/lib/vz/template/iso/debian-13-generic-amd64.qcow2
agent: 1
machine: q35
scsi0: local-lvm:vm-100-disk-0,discard=on,iothread=1,size=20G
scsihw: virtio-scsi-single
```

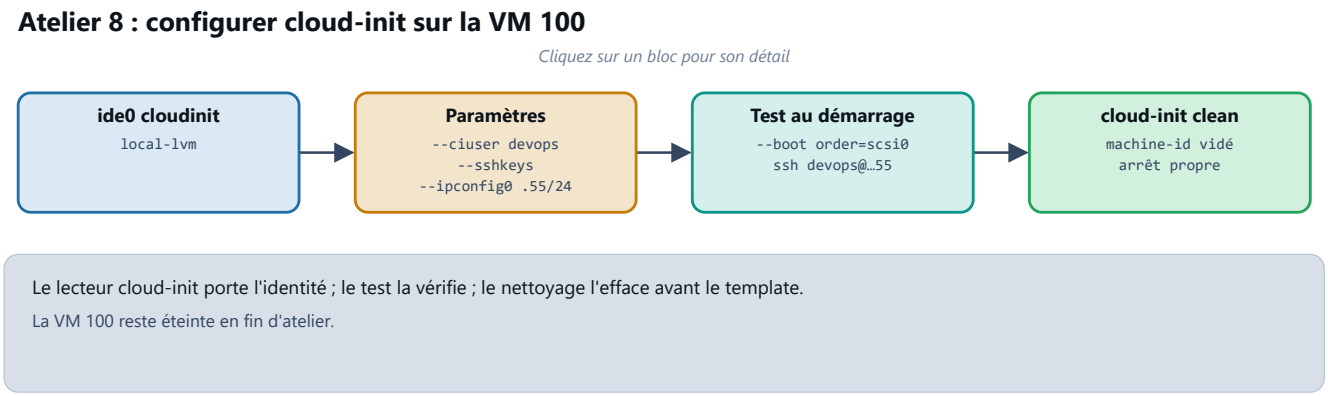
Atelier 8 — Configurer cloud-init

Contexte

Le disque cloud importé est anonyme : ni utilisateur, ni mot de passe, ni réseau configuré. C'est **cloud-init** qui injecte cette identité au premier démarrage, à partir des paramètres que vous fixez dans Proxmox. C'est le mécanisme qui rend un template réellement réutilisable — chaque clone reçoit sa propre configuration sans qu'on touche au disque.

Résultat à obtenir : la VM 100 configurée par cloud-init : utilisateur, mot de passe, clé SSH, adresse **fixe**, ordre de démarrage. Démarrée une fois pour valider la connexion SSH, **nettoyée** de ce premier démarrage, puis éteinte proprement avant d'en faire un template.

Ce support est la propriété de Dawan France.



Le schéma suit la VM 100 du lecteur cloud-init jusqu'au nettoyage final. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
VM 100 « debian13-template »
├── ide0 : lecteur cloud-init   ← porte l'identité injectée au boot
└── cloud-init : user / password / clé SSH / IP FIXE
```

Ce que vous devez faire

1. **Ajouter le lecteur cloud-init** à la VM 100 (un disque `cloudinit` sur `local-lvm`).
2. **Renseigner les paramètres cloud-init** : utilisateur, mot de passe, clé SSH publique, et une **adresse IP fixe**. Jamais de DHCP sur les ressources du lab, conformément à la règle d'adressage de la formation.
3. **Définir l'ordre de démarrage** de la VM.
4. **Tester avant d'en faire un template** : démarrer la VM et vérifier qu'elle répond à son adresse fixe. Se connecter en SSH pour confirmer que cloud-init a bien appliqué la configuration. Passer la console de la VM en clavier français : l'image cloud est en QWERTY (disposition de clavier anglo-saxonne), et chaque clone en hériterait.
5. **Nettoyer, puis éteindre** : effacer ce que ce premier démarrage a laissé (état de cloud-init, identifiant `machine-id`), puis éteindre proprement. Ne la redémarrez plus ensuite.

   ⚠️ On teste **avant** de convertir en template : une fois la VM transformée en template, elle ne se démarre plus, et corriger cloud-init imposerait de tout refaire. Mais ce test laisse une trace : le premier démarrage a donné à la VM un identifiant système, le `machine-id`, que tous les clones recopieraient. D'où l'étape 5, le nettoyage, avant l'arrêt définitif.

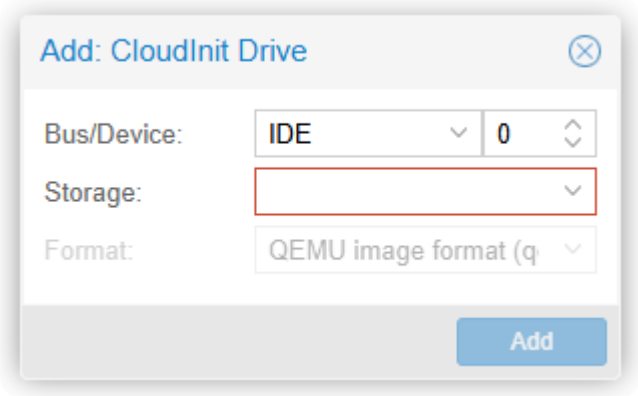
Dawan
Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Ajouter le lecteur cloud-init

Cloud-init lit sa configuration depuis un CD-ROM (*Compact Disc Read-Only Memory*, disque en lecture seule) virtuel : VM 100 → **Hardware** → **Add** → **CloudInit Drive**, Storage `local-lvm` → **Add**.

> **Machine d'exemple : nœud** `pxe211` (Proxmox VE 9.2.21, lab `10.59.1.x`). La VM y a l'adresse `10.59.1.55` (`network`) ; chez vous, c'est `10.<site>.<N>.55`. Les captures datent d'avant le passage du lab sur ce plan.



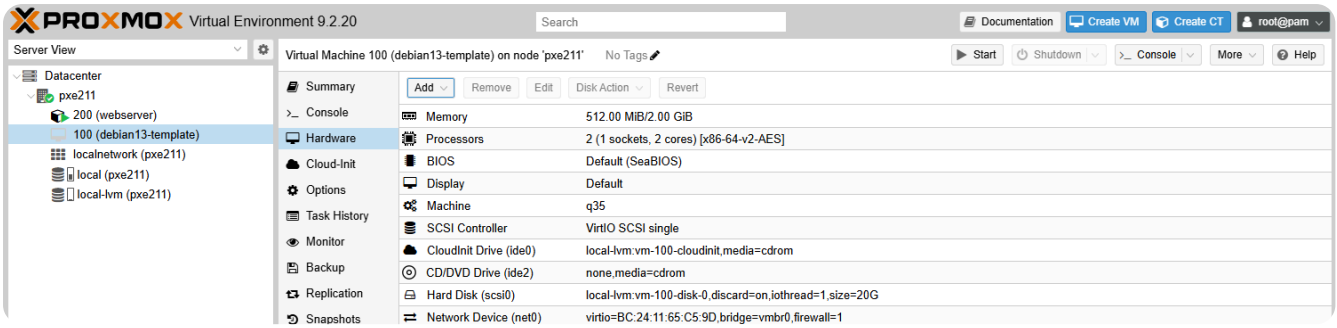
Proxmox propose **IDE 0** : l'emplacement `ide2` est déjà pris par le lecteur CD vide que l'assistant a créé (atelier 7). Gardez IDE 0, et choisissez le stockage `local-lvm`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI : lecteur cloud-init sur ide0, à côté du lecteur CD vide (ide2)
qm set 100 --ide0 local-lvm:cloudinit
```

Résultat attendu : `ide0: local-lvm:vm-100-cloudinit,media=cdrom` dans la configuration, et un volume `vm-100-cloudinit` de 4 Mio sur `local-lvm` — le lecteur est un vrai petit volume.

Ce support est la propriété de Dawan France.



Deux lecteurs CD : **CloudInit Drive (ide0)**, qui portera l'identité de la VM, et le **CD/DVD Drive (ide2)** vide. Le disque système est `scsi0`, en 20 Gio.

Étape 2 — Configurer les paramètres cloud-init

VM 100 → onglet **Cloud-Init** :

Champ	Valeur
User	`devops`
Password	`dawan`
SSH public key	les clés autorisées sur le nœud (`cat ~/.ssh/authorized_keys`) : elles contiennent la clé du nœud lui-même, qui lui permettra de se connecter à la VM
IP Config (net0)	**Static** : `10.<site>.<N>.55/24` , Gateway `10.<site>.<N>.1`

OK puis **Regenerate Image**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Identité injectée au premier démarrage : compte, mot de passe, clé, adresse FIXE
qm set 100 \
  --ciuser devops \
  --cipassword 'dawan' \
  --sshkeys /root/.ssh/authorized_keys \
  --ipconfig0 ip=10.<site>.<N>.55/24,gw=10.<site>.<N>.1

# Relire ce que Proxmox a réellement mis sur le lecteur
qm cloudinit dump 100 user
qm cloudinit dump 100 network   # et la configuration réseau
```

Ce support est la propriété de Dawan France.

Résultat attendu — la partie `user`, puis la partie `network` (clés raccourcies) :

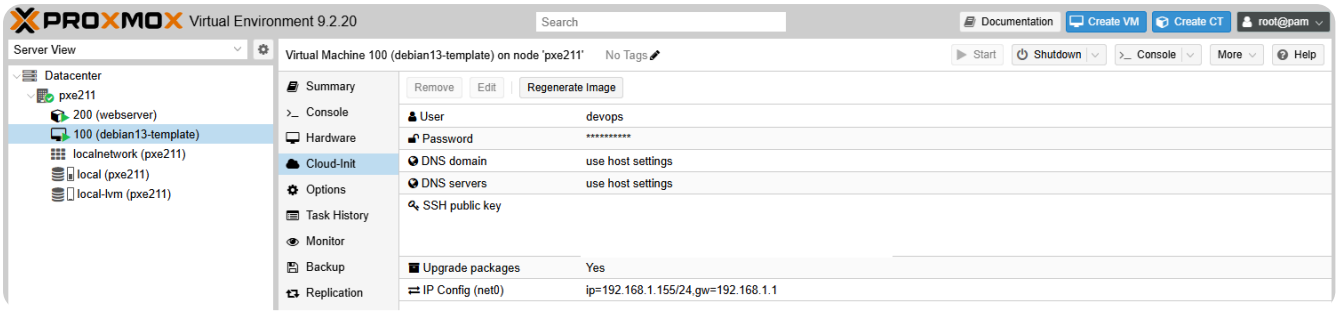
```
#cloud-config
hostname: debian13-template
manage_etc_hosts: true
fqdn: debian13-template
user: devops
password: $5$mNlfP33C$xWwaMz2UVw6hm5bqvXitiKu7bURs5QobWhu2EB.l/d9
ssh_authorized_keys:
  - ssh-rsa AAAAB3NzaC1y... root@pxe211
  - ...
chpasswd:
  expire: False
users:
  - default
package_upgrade: true
```

Résultat attendu, partie `network` :

```
version: 1
config:
    - type: physical
      name: eth0
      mac_address: 'bc:24:11:bf:e1:d7'
      subnets:
      - type: static
        address: '10.59.1.55'
        netmask: '255.255.255.0'
        gateway: '10.59.1.1'
    - type: nameserver
      address:
      - '1.1.1.1'
      search:
      - 'dawan.local'
```

La dernière section n'a pas été saisie : faute de DNS indiqué dans l'onglet Cloud-Init, Proxmox reprend **ceux du nœud** (*use host settings*).

Ce support est la propriété de Dawan France.



Chaque paramètre saisi : **User** `devops`, **Password** masqué, **DNS** `use host settings`, les clés SSH (masquées ici), **Upgrade packages** `Yes` et **IP Config (net0)** en adresse fixe. Après toute modification, **Regenerate Image** reconstruit le lecteur ; Proxmox le fait aussi seul au démarrage (`generating cloud-init ISO`).

Trois lignes méritent une explication. **`hostname`** vient du **nom de la VM** : c'est pour cela qu'un clone nommé `vm-web-01` s'appellera `vm-web-01` sans rien configurer. **`password`** est déjà une empreinte : Proxmox ne stocke jamais le mot de passe en clair, même dans sa configuration.

`users: - default` crée le compte par défaut de la distribution, renommé `devops` par la ligne `user:`. Sur Debian, ce compte par défaut reçoit d'office le droit `sudo` sans mot de passe. `package_upgrade: true` fait mettre à jour les paquets au premier démarrage : c'est pourquoi ce premier démarrage est plus long que les suivants.

Étape 3 — Configurer l'ordre de démarrage

VM 100 → **Options** → **Boot Order** → **Edit** : cochez uniquement `scsi0`, décochez ide0/ide2/net0, faites-le glisser en première position.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Démarrer uniquement sur le disque système
qm set 100 --boot order=scsi0
```

Résultat attendu : `update VM 100: -boot order=scsi0`. L'assistant avait réglé `order=ide2;net0` : sans cette étape, la VM chercherait à démarrer sur le lecteur CD vide, puis sur le réseau.

Étape 4 — Tester la VM avant de créer le template

Démarrez : VM 100 → **Start** → **Console**. Cloud-init s'exécute au premier boot : il applique l'identité, agrandit le disque, puis met à jour les paquets (`package_upgrade: true`).

Ce support est la propriété de Dawan France.

Depuis le **Shell du nœud**, attendez que la VM réponde, puis connectez-vous avec la clé injectée :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
ping -c 2 10.<site>.<N>.55   # la VM répond-elle à son adresse fixe ?
ssh devops@10.<site>.<N>.55  # puis, dans la VM :
hostname                     # son nom
id                           # le compte
sudo -n true                 # sudo, sans mot de passe ?
echo "code : $?"             # 0 : oui
df -h /                      # le disque
cloud-init status            # l'état de cloud-init
exit                         # retour au nœud
```

Résultat attendu : le port SSH de la VM répondait une vingtaine de secondes après le démarrage.

```
debian13-template
uid=1000(devops) gid=1000(devops)
groups=1000(devops),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30
(dip),44(video),46(plugdev)
code : 0
/dev/sda1        20G  950M   18G   5% /
status: done
```

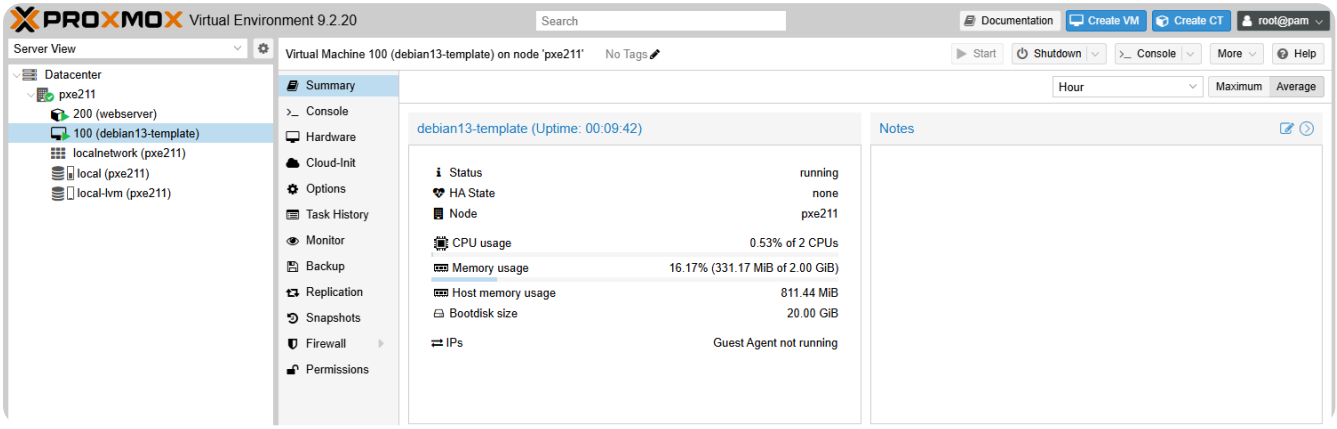
Tout est là. Le nom vient de la VM ; le compte `devops` est dans le groupe `sudo` et l'utilise **sans mot de passe** (`sudo -n` réussit).

Le système de fichiers fait bien **20G** : cloud-init a agrandi la partition seul. Et `cloud-init status` confirme que le premier démarrage est terminé.

> **L'agent QEMU n'est pas dans l'image cloud Debian.** Sur le lab, `qm guest cmd 100 network-get-interfaces` a répondu `QEMU guest agent is not running` pendant plus de six minutes, et pour cause : `dpkg -l qemu-guest-agent` répond `no packages found` .
>
> La case **Qemu Agent** de l'atelier 7 prépare seulement le canal côté Proxmox ; le programme, lui, s'installe dans la VM (`apt install qemu-guest-agent` , comme à l'atelier

Ce support est la propriété de Dawan France.

> 5). Tant qu'il n'y est pas, l'interface affiche **Guest Agent not running** à la place des adresses IP.



La VM tourne (`running` , 2 CPU, disque de 20 Gio), mais la ligne **IPs** affiche `Guest Agent not running` : sans agent, Proxmox ne connaît pas l'adresse de la VM. C'est aussi pourquoi **Shutdown** met du temps : Proxmox essaie d'abord l'agent (`QEMU Guest Agent is not running - VM 100 qga command 'guest-ping' failed - got timeout`), puis passe par le signal ACPI.

Le clavier de la console. Tapez quelques lettres dans la **Console** de la VM : le `a` sort en `q` . L'image cloud Debian n'a aucun outil de clavier, et sa console reste en QWERTY.

Corrigez-le maintenant, avant le template : chaque clone en héritera.

Dans le Shell du nœud (interface web → votre nœud → **Shell**), connectez-vous à la VM :

```
ssh devops@10.<site>.<N>.55   # se connecter à la VM
```

Puis, **dans la VM** :

```
sudo apt-get update   # la liste des paquets
# Les outils du clavier de la console. DEBIAN_FRONTEND=noninteractive : aucune question,
# la disposition par défaut (us) est retenue ; la ligne suivante la remplace
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y console-setup
sudo nano /etc/default/keyboard   # disposition française : la ligne XKBLAYOUT devient XKBLAYOUT="fr"
sudo setupcon --force --save   # appliquer, et enregistrer pour les démarrages suivants
```

Ce support est la propriété de Dawan France.

```
grep XKBLAYOUT /etc/default/keyboard   # vérifier
exit   # quitter la VM
```

Résultat attendu (VM de test sur `pxe212`, même image Debian 13) : `XKBLAYOUT="fr"`. Après un redémarrage, les touches `q w a z m` frappées dans la console ont affiché `azqw,` : la console est en AZERTY (disposition de clavier française).

Étape 5 — Nettoyer le premier démarrage, puis éteindre

Le test a laissé deux traces dans la VM : l'état de cloud-init (`/var/lib/cloud/`), qui marque l'initialisation comme faite, et surtout l'identifiant système `/etc/machine-id`, tiré au hasard au premier démarrage.

Converti tel quel en template, ce disque donnerait **le même `machine-id` à tous les clones** : vérifié sur le lab, deux clones ont répondu le même identifiant, `87703c63bbf841358c1a7eb6912f791b`, celui du template. Or des services s'en servent comme identité unique de la machine : le journal système, ou l'identifiant client d'un DHCP.

`cloud-init clean` efface les deux d'un coup :

Dans le Shell du nœud (interface web → votre nœud → **Shell**), puis dans la VM :

```
ssh devops@10.<site>.<N>.55                          # mot de passe : celui de la clé, ou aucun
# --logs : efface aussi les journaux de cloud-init ; --machine-id : remet l'identifiant système à zéro
sudo cloud-init clean --logs --machine-id
cat /etc/machine-id                                  # l'identifiant, remis à zéro
exit                                                 # retour au nœud
```

Résultat attendu (cloud-init 25.1.4, celui de l'image Debian 13) :

```
uninitialized
```

`uninitialized` dit à systemd de tirer un nouvel identifiant au prochain démarrage, et cloud-init, sans état, s'exécutera de nouveau : chaque clone aura donc son propre identifiant et sa propre initialisation.

Ce support est la propriété de Dawan France.

Éteignez maintenant la VM, sans la redémarrer : VM 100 → **Shutdown**. Un nouveau démarrage referait un `machine-id`, et il faudrait nettoyer de nouveau.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le compte injecté, l'ordre de démarrage, puis le compte et sudo vus de l'intérieur
qm cloudinit dump 100 user   # à lire : user, hostname
qm config 100 | grep "^boot"   # l'ordre de démarrage
qm status 100   # éteinte après le nettoyage
```

Résultat attendu (les lignes qui comptent) : `hostname: debian13-template` et `user: devops`, puis `boot: order=scsi0`, puis `status: stopped`. La connexion SSH et `sudo` ont été vérifiés à l'étape 4, avant le nettoyage.

Atelier 9 — Créer un template et déployer des clones

Contexte

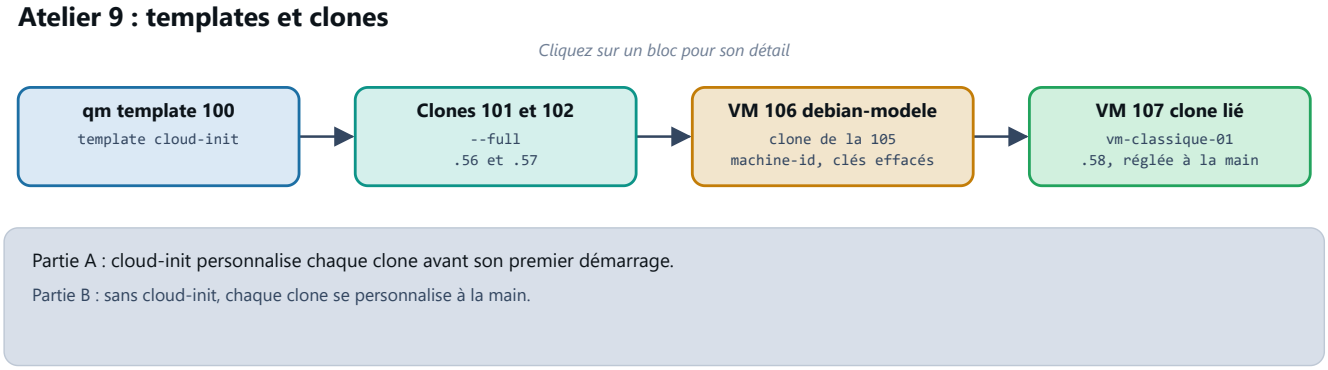
La VM 100 est configurée et validée : il est temps d'en faire un **template**, c'est-à-dire un modèle en lecture seule qu'on ne démarre plus mais qu'on clone. C'est là que tout le travail précédent porte ses fruits — deux VMs de production naissent en quelques secondes, identiques par construction, différenciées seulement par ce que cloud-init leur injecte.

Un template ne demande pas cloud-init : n'importe quelle VM installée peut en devenir un. La partie B le fait avec la VM 105 de l'atelier 2, pour comparer : tout ce que cloud-init fait seul doit alors se faire à la main, clone par clone.

Résultat à obtenir :

- **Partie A** : la VM 100 convertie en template, et deux clones complets — VM 101 et VM 102 — démarrant chacun avec une **adresse IP statique distincte**.
- **Partie B** : un template 106 tiré de la VM 105, nettoyé à la main. Puis un clone 107, personnalisé dans sa console, en `10.<site>.<N>.58`, avec ses propres `machine-id` et clés SSH.

Dawan Ce support est la propriété de Dawan France.



Les deux premiers blocs suivent la partie A, les deux suivants la partie B. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
Partie A
VM 100 « debian13-template »   ← TEMPLATE (ne démarre plus)
├── clone complet → VM 101 « vm-web-01 »   IP fixe 10.<site>.<N>.56
└── clone complet → VM 102 « vm-web-02 »   IP fixe 10.<site>.<N>.57

Partie B
VM 105 « vm-debian » (atelier 2)
└── clone complet → VM 106 « debian-modele », nettoyée → TEMPLATE
    └── clone lié → VM 107 « vm-classique-01 »   IP fixe 10.<site>.<N>.58, réglée à
la main
```

Ce que vous devez faire

Partie A — un template cloud-init

1. **Convertir la VM 100 en template.**
2. **Cloner** le template pour créer la VM 101 (clone **complet**, pas lié — demandez-vous pourquoi le clone complet est préférable ici).
3. **Personnaliser le clone avant de le démarrer** : lui donner son propre nom d'hôte et son adresse IP fixe via cloud-init.
4. **Déployer un second clone**, VM 102, avec une IP différente — et vérifier que les deux VMs coexistent sans collision de nom ni d'adresse.

Partie B — un template sans cloud-init

Ce support est la propriété de Dawan France.

5. **Cloner la VM 105** en VM 106 (clone complet), **carte réseau débranchée** : le clone porte la même adresse que la VM 105.
6. **Effacer son identité** dans sa console : `machine-id` et clés SSH de la machine. Puis l'éteindre et la **convertir en template**.
7. **Cloner le template** en VM 107, la démarrer carte débranchée, et relever ce dont elle a hérité.
8. **La personnaliser à la main** : nom, adresse `.58`, clés SSH. Rebrancher la carte, redémarrer, et prouver depuis le nœud qu'elle ne se confond pas avec la VM 105.

> 💡 Clone lié ou clone complet ? Le clone lié est instantané mais reste dépendant du template ; le clone complet est autonome. Sur des VMs de production, on privilégie l'autonomie.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Convertir la VM en template

VM 100 → clic droit → **Convert to Template** → confirmer.

> **Pas de conversion inverse officielle.** Une fois la VM convertie, elle ne peut plus démarrer directement — seuls les clones peuvent l'être.
>
> Ni l'interface web ni une sous-commande `qm` dédiée n'offrent de "dé-templater" une VM. Le seul contournement (non officiellement documenté) consiste à éditer directement le fichier de configuration `/etc/pve/qemu-server/100.conf` pour retirer la ligne `template: 1`.

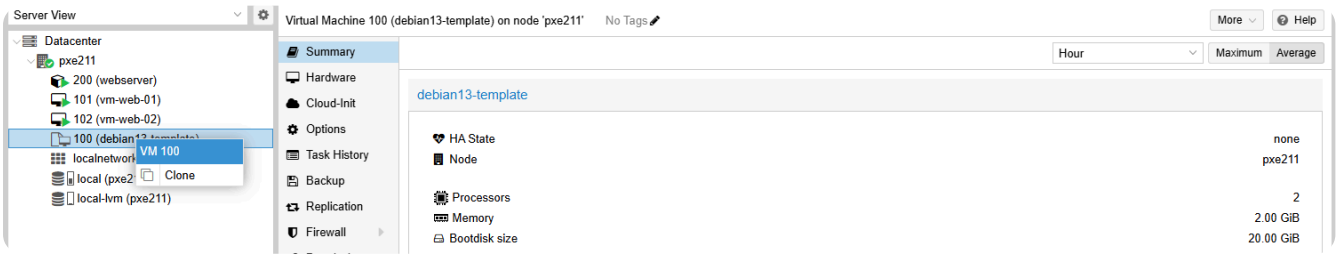
Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Figer la VM 100 en modèle
qm template 100
```

Résultat attendu :

```
Renamed "vm-100-disk-0" to "base-100-disk-0" in volume group "pve"
Logical volume pve/base-100-disk-0 changed.
```

Seul le disque système est renommé : le préfixe `base-` signale un disque devenu la base, en lecture seule, de tous les futurs clones. Le lecteur cloud-init, lui, reste `vm-100-cloudinit` : chaque clone reçoit son propre lecteur, régénéré à son premier démarrage. `qm config 100` montre désormais la ligne `template: 1`.



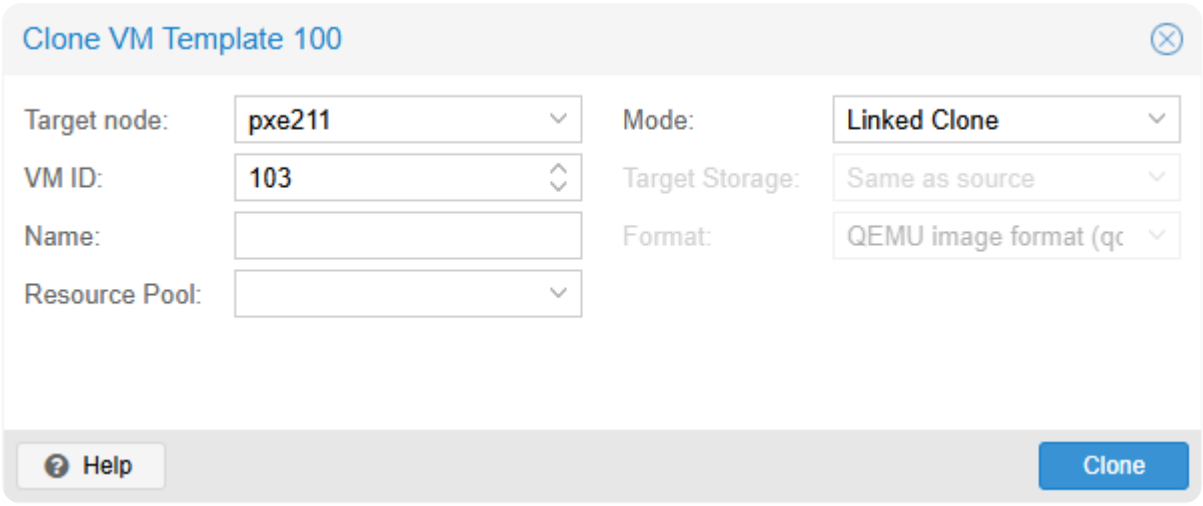
La VM 100 a changé d'icône (une page stylisée) et passé en bas de la liste. Son clic droit ne propose plus que **Clone** : ni *Start*, ni *Console*. Son résumé ne montre que la configuration —

Ce support est la propriété de Dawan France.

processeurs, mémoire, disque de 20 Gio — sans aucune mesure d'activité.

Étape 2 — Cloner le template pour créer une VM de production

VM 100 (template) → clic droit → **Clone** : Target node `pxe2<N>1`, VM ID `101`, Name `vm-web-01`, Mode **Full Clone**, Target Storage `local-lvm` → **Clone**.



La fenêtre telle qu'elle s'ouvre. **Attention au mode : Proxmox propose `Linked Clone` par défaut** sur un template.

Il faut le passer à **Full Clone**, ce qui débloque le choix du **Target Storage**. Le **VM ID** proposé est le premier libre.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Clone complet : un disque indépendant, copié sur local-lvm
qm clone 100 101 --name vm-web-01 --full --storage local-lvm
```

Résultat attendu :

```
create full clone of drive scsi0 (local-lvm:base-100-disk-0)
  Logical volume "vm-101-disk-0" created.
transferred 20.0 GiB of 20.0 GiB (100.00%)
```

La copie des 20 Gio a pris environ 7 secondes : sur un pool thin, seuls les blocs réellement écrits (moins de 1 Gio ici) sont recopiés.

Ce support est la propriété de Dawan France.

Full Clone vs Linked Clone :

- **Full Clone** : copie complète du disque — VM totalement indépendante du template. Prend plus d'espace mais peut être déplacée ou sauvegardée librement.
- **Linked Clone** : partage les blocs communs avec le template (copy-on-write) — démarrage instantané, minimal en espace. La VM ne peut pas migrer sans le template.

Étape 3 — Personnaliser le clone avant démarrage

VM 101 → **Cloud-Init** → IP Config (net0) → Edit : `10.<site>.<N>.56/24`, Gateway `10.<site>.<N>.1` → **OK** → **Regenerate Image**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le clone reçoit SA propre adresse ; le nom d'hôte vient du nom de la VM
qm set 101 \
  --ciuser devops \
  --ipconfig0 ip=10.<site>.<N>.56/24,gw=10.<site>.<N>.1
```

Démarrez : VM 101 → **Start**. Attendez 30 secondes.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Se connecter au clone et vérifier son identité
ssh devops@10.<site>.<N>.56
hostname   # le nom donné par cloud-init
ip addr show eth0   # l'adresse fixe de la VM
```

Résultat attendu (lab `10.59.1.x`) : `vm-web-01`, puis `eth0 UP 10.59.1.56/24` ; `vm-web-02` en `10.59.1.57/24`. Les deux clones répondaient en SSH 15 secondes après leur démarrage.

Ce qui vient de se passer : en moins d'une minute, une VM Debian 13 complète a été déployée avec son hostname, son IP et sa clé SSH. Sans aucune installation manuelle. C'est exactement ce que font les providers cloud (AWS — *Amazon Web Services* —, Azure, GCP — *Google Cloud Platform*) à chaque création d'instance.

Dawan Ce support est la propriété de Dawan France.

Étape 4 — Déployer un second clone avec une IP différente

Répétez l'opération pour la VM 102 : VM 100 → **Clone**, VM ID `102`, Name `vm-web-02`, Full Clone. Puis Cloud-Init : `10.<site>.<N>.57/24`, Gateway `10.<site>.<N>.1`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Second clone, seconde adresse
qm clone 100 102 --name vm-web-02 --full --storage local-lvm
qm set 102 --ipconfig0 ip=10.<site>.<N>.57/24,gw=10.<site>.<N>.1   # son adresse fixe
qm start 102   # démarrer le clone
```

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

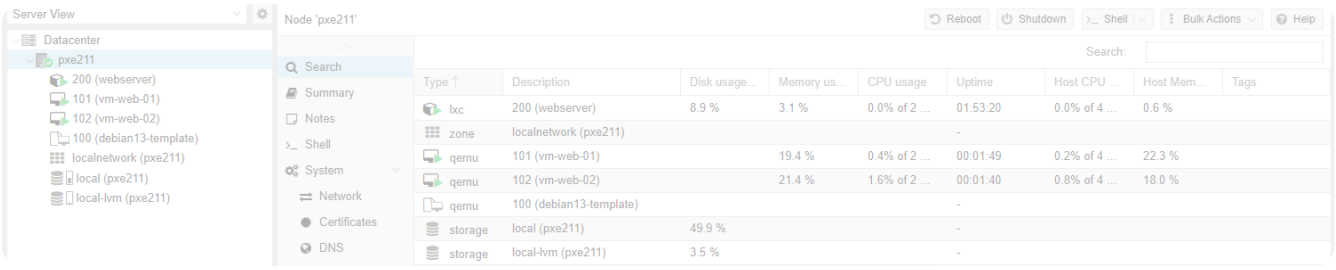
```
# Les trois VM, puis le nom et l'identifiant système de chaque clone
qm list   # à lire : 100, 101, 102
ssh devops@10.<site>.<N>.56 hostname                # le nom du clone 101
ssh devops@10.<site>.<N>.56 cat /etc/machine-id     # son identifiant
ssh devops@10.<site>.<N>.57 hostname                # le nom du clone 102
ssh devops@10.<site>.<N>.57 cat /etc/machine-id     # son identifiant
```

Résultat attendu (les lignes qui comptent) :

```
       100 debian13-template    stopped    2048              20.00 0
       101 vm-web-01            running    2048              20.00 37513
       102 vm-web-02            running    2048              20.00 37681
vm-web-01
<32 caractères hexadécimaux>
vm-web-02
<32 autres caractères hexadécimaux>
```

Le template 100 `stopped`, les VM 101 et 102 `running`, deux noms, deux adresses, aucune collision. Les deux `machine-id` doivent être **différents** : sur le lab, `0ecd8b4b61ab4e26965c29f31bb2679c` et `11e7b2f2090b46f09d50e58ed09d8660`. S'ils sont identiques, le nettoyage de l'atelier 8 (étape 5) a été oublié.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.



Le CT 200 et les deux clones tournent. Regardez la colonne **Host Mem...** : chaque VM occupe environ 20 % de la mémoire du nœud.

> **Surveillez la mémoire du nœud.** Votre nœud dispose de 4 Go. Deux VM de 2 Go plus le CT 200 dépassent cette capacité sur le papier : c'est le ballooning (plancher de 512 Mio, atelier 7) qui rend la cohabitation possible.
>
> Sur le lab, `free -m` affichait `3846` Mo au total, `3240` utilisés et `605` disponibles. Arrêtez les VM dont vous n'avez plus besoin (`qm shutdown`), avant d'en démarrer d'autres.

Partie B — Un template sans cloud-init, à partir de la VM 105

> **Machine d'exemple : nœud** `pxe211` (Proxmox VE 9.2.21, lab `10.59.1.x`). La VM 105 y est en `10.59.1.54` et le clone 107 en `10.59.1.58` ; chez vous, `10.<site>.<N>.54` et `10.<site>.<N>.58` . Les commandes dans la VM se tapent dans sa **Console** : le mot de passe de `devops` est `dawan` .

Étape 6 — Cloner la VM 105, carte débranchée

Via l'interface : VM 105 → **More** → **Clone** : **VM ID** `106` , **Name** `debian-modele` , **Mode** `Full Clone` → **Clone**. Puis VM 106 → **Hardware** → **Network Device** → **Edit** → cocher **Disconnect** → **OK**.

Dans le Shell du nœud :

```
qm clone 105 106 --name debian-modele --full   # clone complet de la VM 105
qm set 106 --net0 virtio,bridge=vmbr0,link_down=1   # la carte, débranchée
qm start 106                                   # démarrer le clone
```

Ce support est la propriété de Dawan France.

Pourquoi débrancher : le clone est la copie exacte du disque de la VM 105, adresse fixe comprise. Branché, il entrerait en conflit avec elle.

Étape 7 — Effacer l'identité, puis faire le template

Dans la Console de la VM 106, connecté en `devops` :

```
hostname                    # ce que le clone a hérité : le nom
cat /etc/machine-id         # l'identifiant système
ls /etc/ssh/ssh_host_*      # les clés SSH de la machine
ip -br a show ens18         # la carte et son adresse
```

Résultat attendu : `vm-debian`, le `machine-id` de la VM 105 (`c85ab6a3cfc54e329cc7f19c6da42284`), six fichiers `ssh_host_*` , et `ens18 DOWN 10.59.1.54/24` . Le clone est la VM 105, à l'identique.

Dans la console de la VM 106 :

```
sudo truncate -s 0 /etc/machine-id                          # vider l'identifiant système
sudo rm -v /var/lib/dbus/machine-id /etc/ssh/ssh_host_*      # sa copie D-Bus, et les clés SSH de la machine
wc -c /etc/machine-id                                       # vérifier : 0 octet
ls /var/lib/dbus/ /etc/ssh/                                 # plus de copie, plus de clés
sudo poweroff                                               # éteindre
```

Résultat attendu : les sept fichiers `supprimé` , puis `0 /etc/machine-id` , un dossier `/var/lib/dbus/` vide, et plus aucune clé `ssh_host_*` dans `/etc/ssh/` .

> **Vider `/etc/machine-id` ne suffit pas sur une Debian installée depuis l'ISO.** Sur le lab, au premier essai, sans supprimer `/var/lib/dbus/machine-id` : le clone a redémarré avec le `machine-id` de la VM 105, `c85ab6a3cfc54e329cc7f19c6da42284` .
>
> Au démarrage, systemd recrée `/etc/machine-id` à partir de cette copie si elle existe. L'image cloud de la partie A n'a pas ce piège : `cloud-init clean --machine-id` s'en charge.

Ce support est la propriété de Dawan France.

Puis, la VM éteinte : VM 106 → **More** → **Convert to template**, ou en ligne de commande.

Dans le Shell du nœud :

```
qm template 106   # convertir en template
```

Étape 8 — Un clone, et ce dont il hérite

Via l'interface : clic droit sur le template 106 → **Clone** : **VM ID** `107`, **Name** `vm-classique-01`, **Mode** `Linked Clone` (proposé d'office) → **Clone**. La carte reste débranchée : le clone hérite du réglage du template.

Dans le Shell du nœud :

```
qm clone 106 107 --name vm-classique-01   # clone lié, par défaut depuis un template
qm start 107                              # le démarrer
```

Dans la Console de la VM 107 :

```
hostname                    # le nom
ip -br a show ens18         # la carte
cat /etc/machine-id         # l'identifiant, neuf
systemctl is-active ssh     # le service SSH
```

Résultat attendu :

```
vm-debian
ens18            DOWN           10.59.1.54/24
f298cc1603ad42aca68c84b1650c883a
failed
```

Le `machine-id` est neuf : le nettoyage a fonctionné. Mais le nom et l'adresse sont ceux de la VM 105, et le service SSH est en échec : il n'a plus de clés de machine. Tout ce que cloud-init règle seul, dans la partie A, reste à faire à la main.

Étape 9 — Personnaliser le clone à la main

Ce support est la propriété de Dawan France.

Dans la Console de la VM 107 :

```
sudo hostnamectl set-hostname vm-classique-01        # son propre nom
grep -rn "address" /etc/network/interfaces /etc/network/interfaces.d/   # dans quel fichier est l'adresse de ens18 ?
sudo nano /etc/network/interfaces                    # ou le fichier que grep a trouvé : remplacer .54 par .58 sur la ligne address
sudo ssh-keygen -A                                   # générer ses propres clés SSH de machine
```

Où est l'adresse ? Cela dépend de l'installation de la VM 105 (atelier 2, étape 3). Si l'installateur a écrit l'adresse fixe, elle est dans `/etc/network/interfaces`.

Si vous l'avez corrigée à la main, elle est dans `/etc/network/interfaces.d/ens18`. Modifiez le fichier existant : n'en créez pas un second, sinon `ens18` serait déclarée deux fois.

Le bloc de `ens18`, une fois modifié (exemple : stagiaire 2 à Paris) :

```
iface ens18 inet static
    address 10.75.2.58/24
    gateway 10.75.2.1
```

La passerelle est `10.<site>.<N>.1`, celle de votre réseau. `192.168.<site>.1` est le **serveur DNS** : il ne va pas sur la ligne `gateway`.

Résultat attendu : après `hostnamectl`, `sudo` affiche `impossible de résoudre l'hôte vm-classique-01` : le nouveau nom n'est connu qu'au prochain démarrage, et `sudo` attend plusieurs secondes avant de répondre. `ssh-keygen -A` répond `generating new host keys: RSA ECDSA ED25519`.

Redémarrez d'abord la VM, carte toujours débranchée (VM 107 → **Reboot**) : c'est au démarrage qu'elle prend sa nouvelle adresse. **Puis rebranchez la carte** : VM 107 → **Hardware** → **Network Device** → **Edit** → décocher **Disconnect**.

> **Rebrancher avant de redémarrer crée un conflit d'adresse.** Sur le lab : la carte rebranchée avant le redémarrage portait encore `.54`. Le nœud a joint la VM 107 à la place de la VM 105, puis a gardé la mauvaise correspondance après le redémarrage.

Ce support est la propriété de Dawan France.

> `ip neigh flush 10.<site>.<N>.54`, dans le Shell du nœud, l'a effacée.

Étape 10 — Vérifier depuis le nœud

Dans le Shell du nœud :

```
ping -c 2 10.<site>.<N>.58                                    # le clone répond à sa nouvelle adresse
ssh-keyscan -t ed25519 10.<site>.<N>.58 | ssh-keygen -lf -    # empreinte de sa clé SSH
ssh-keyscan -t ed25519 10.<site>.<N>.54 | ssh-keygen -lf -    # celle de la VM 105
```

Résultat attendu : `2 received, 0% packet loss`, puis deux empreintes différentes : `SHA256:jbNlPFQIM9zSgk5FQy4CQJmUw7VdpmRqvasFOxrjGyk` pour la VM 107, `SHA256:YYdxFjHDeoTahliKT1HXZ9ZQ5EZlLpT1ZKcUPa3Ue6I` pour la VM 105. Deux machines distinctes, nées du même disque.

	Partie A — template cloud-init	Partie B — template d'une VM installée
Point de départ	Image cloud de la distribution	Une VM installée depuis l'ISO
Nettoyage avant conversion	`cloud-init clean --logs --machine-id`	`machine-id`, sa copie D-Bus et les clés SSH, effacés à la main
Nom, adresse, compte	Injectés par Proxmox au premier démarrage	Réglés à la main dans la console de chaque clone
Clés SSH de la machine	Générées au premier démarrage	`ssh-keygen -A` à la main
Cent clones	Une boucle de `qm clone` et `qm set`	Cent passages dans une console

Le template d'une VM installée convient pour quelques machines, ou pour un système qui n'a pas d'image cloud. Dès qu'on déploie en série, cloud-init fait le travail de l'étape 9 à votre place.

Dépannage — Erreurs fréquentes

Ce support est la propriété de Dawan France.

Console noVNC noire sur une VM cloud Ubuntu. `--vga serial0` redirige la sortie vers le port série. Remède : Utiliser l'onglet **Console** → **xterm.js**, pas noVNC.

`qm cloudinit dump` **n'affiche pas la clé SSH attendue.** Fichier `authorized_keys` non régénéré après modification. Remède : Cliquer **Regenerate Image** dans l'onglet Cloud-Init après toute modification.

Deux clones ont le même `machine-id` **.** Template converti sans nettoyage après un démarrage.

Ou, sur une VM installée depuis l'ISO, `/etc/machine-id` vidé mais `/var/lib/dbus/machine-id` laissé en place : systemd en recopie l'identifiant (atelier 9, partie B). Remède : Refaire la préparation du template : `cloud-init clean --machine-id` pour une image cloud ; `truncate -s 0 /etc/machine-id` **et** `rm /var/lib/dbus/machine-id` pour une VM installée.

Pool `local-lvm` **proche de la saturation.** Multiples VMs/templates non nettoyés, seuil des 70% dépassé. Remède : `lvs -o name,size,data_percent pve/data`, supprimer les VMs de test obsolètes.

Récapitulatif des commandes essentielles

Situation	Commande
Créer une VM / importer un disque cloud	`qm create` / `qm disk import <vmid> <image> <storage>`
Attacher et redimensionner un disque	`qm set <vmid> --scsi0 ...` / `qm resize <vmid> scsi0 +NG`
Configurer cloud-init	`qm set <vmid> --ciuser ... --cipassword ... --ipconfig0 ...`
Convertir en template / cloner	`qm template <vmid>` / `qm clone <vmid> <new> --full --storage ...`
Snapshot / rollback	`qm snapshot <vmid> <nom>` / `qm rollback <vmid> <nom>`
Sauvegarde / restauration	`vzdump <vmid> --mode snapshot` / `qmrestore <archive> <nouveau-vmid>`
Interroger l'agent QEMU	`qm guest cmd <vmid> network-get-interfaces`

Ce support est la propriété de Dawan France.

Chapitre 3 — Réseau et stockage

Jusqu'ici, tout ce que vous avez créé tenait sur le disque système du nœud et sortait par un seul pont, `vmbr0`. C'est suffisant pour apprendre, et insuffisant pour exploiter : un disque unique ne protège de rien, un pont unique mélange l'administration, les machines et les échanges entre serveurs. Ce chapitre donne au nœud ce qui lui manque encore pour la production.

Des réseaux séparés au niveau du **pont Linux**. Un stockage qui détecte et répare ses propres erreurs. Et un stockage partagé par le réseau.

Le réseau se construit ici comme il se construisait avant l'arrivée du SDN : avec les **bridges** (`vmbr`), les **VLAN** portés par un pont VLAN-aware, et le **NAT**. C'est le socle, indépendant de tout cluster, et il suffit à un nœud seul.

Le fil conducteur suit le plan de disques et de cartes réseau de votre lab. Vous commencez par le réseau (ponts, VLAN, NAT) et préparez le pont `vmbr1`, réservé au réseau du cluster. Vous posez ensuite la grille de lecture du stockage : **bloc** ou **fichier**, et ce que chaque modèle permet.

Vous montez alors un pool ZFS en miroir sur `sdc` et `sdd`, puis les stockages réseau NFS et iSCSI, tels qu'on les déclare dans Proxmox. Une conférence porte ensuite sur Ceph, le stockage distribué. Vous terminez par le choix et la surveillance des stockages.

Objectifs du chapitre

À l'issue de ce chapitre, les stagiaires seront capables de :

- **Configurer** le réseau Linux de Proxmox (bridges, VLANs, NAT)
- **Distinguer** stockage bloc et stockage fichier, et en déduire ce que Proxmox autorise (format de disque, snapshots, contenus)
- **Mettre en place** un stockage ZFS et un pool LVM-thin, et en expliquer les mécanismes de protection
- **Connecter** un stockage réseau NFS ou iSCSI à Proxmox
- **Surveiller** les stockages selon les bonnes pratiques opérationnelles

> **Infrastructure disponible à ce stade :** 1 nœud Proxmox opérationnel + disques supplémentaires disponibles pour ZFS.

Dawan
Ce support est la propriété de Dawan France.

1. Le réseau Linux : du câble au bridge

1.1 De l'interface physique au bridge virtuel

Proxmox n'invente pas son propre système réseau — il s'appuie entièrement sur la pile réseau de Linux, avec ses bridges, ses interfaces et ses VLANs.

Quand vous installez Proxmox sur un serveur physique, votre carte réseau physique (`ens18` sur les nœuds du lab) devient le point d'entrée du trafic. Mais cette interface physique ne connecte pas directement vos VMs et vos conteneurs — c'est le rôle du **bridge Linux**, appelé `vmbr0` par convention.

Un bridge Linux est une interface virtuelle qui fonctionne comme un pont logiciel. Il permet de relier plusieurs domaines de broadcast : une interface physique de l'hôte, et les interfaces des machines virtuelles.

Un bridge configuré sur un hôte permet aux machines virtuelles qui y sont connectées de communiquer entre elles. Si une interface physique est liée au bridge, elles communiquent aussi avec le réseau physique.

Un bridge est une interface réseau à part entière. À ce titre, on peut le configurer exactement comme une interface physique : adresse IP, VLAN-aware, MTU, interface physique reliée.

Comment s'appelle votre carte réseau, au juste

La question paraît triviale ; elle est la première source de blocage dès qu'on crée un second bridge.

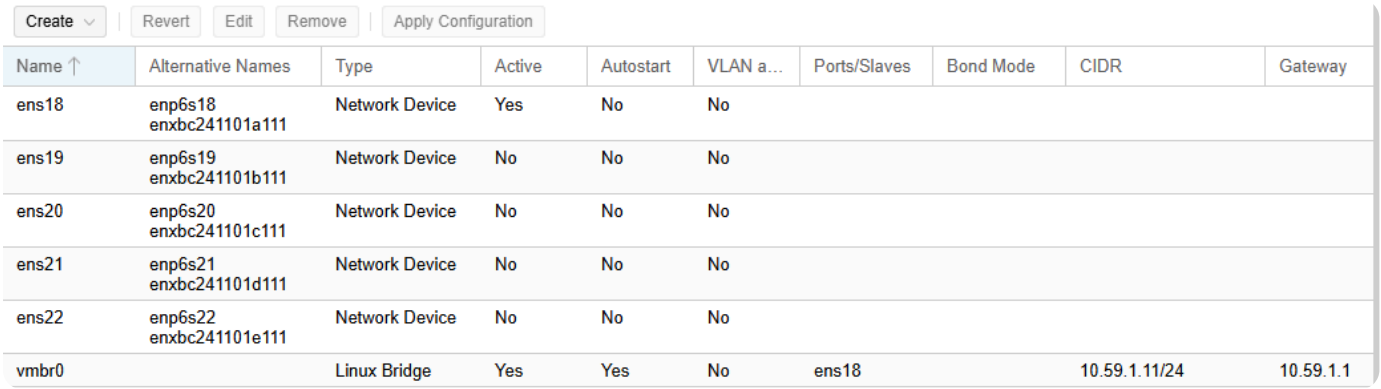
Les deux générations de noms.

`eth0` , `eth1` — la convention historique. Simple, mais instable : l'ordre dépendait de celui où le noyau détectait les cartes, si bien qu'ajouter une carte pouvait renommer les autres et casser la configuration réseau au redémarrage.

`ens18` , `enp6s18` — le nommage **prévisible** de systemd, celui de vos nœuds. Le nom encode l'emplacement matériel de la carte : `ens18` est la carte du slot 18 ; `enp6s18` , son autre nom, se lit « ethernet, bus PCI 6, slot 18 ».

Ce support est la propriété de Dawan France.

Il reste le même d'un démarrage à l'autre, tant que la carte ne change pas d'emplacement. Proxmox VE 9 sait aussi remplacer ces noms par des noms courts (`nic0` , `nic1` ...) : c'est l'épinglage, rare, et vos nœuds ne l'utilisent pas.


Create | Revert | Edit | Remove | Apply Configuration

Name	Alternative Names	Type	Active	Autostart	VLAN a...	Ports/Slaves	Bond Mode	CIDR	Gateway
ens18	enp6s18 enxbc241101a111	Network Device	Yes	No	No				
ens19	enp6s19 enxbc241101b111	Network Device	No	No	No				
ens20	enp6s20 enxbc241101c111	Network Device	No	No	No				
ens21	enp6s21 enxbc241101d111	Network Device	No	No	No				
ens22	enp6s22 enxbc241101e111	Network Device	No	No	No				
vmbr0		Linux Bridge	Yes	Yes	No	ens18		10.59.1.11/24	10.59.1.1

L'onglet **Network** de `pxe211` , votre écran. Cinq cartes, `ens18` à `ens22` . La colonne **Alternative Names** donne les autres noms de chacune (`enp6s18` , `enxbc241101a111` pour `ens18`) : ils désignent la même carte.

En dessous, `vmbr0` , le pont qui porte l'adresse du nœud : `10.59.1.11/24` , passerelle `10.59.1.1` , sur le nœud du lab (le vôtre : `10.<site>.<N>.11`).

La méthode, en trois questions. Quelle carte est réellement utilisée ? Celle dont la colonne `Active` vaut `Yes` : ici `ens18` .

Laquelle est déjà prise ? Celle qui apparaît dans la colonne `Ports/Slaves` d'un pont : `ens18` , sous `vmbr0` .

Laquelle reste disponible ? Toutes les autres, `ens19` à `ens22` — c'est l'une d'elles que vous attacherez à un nouveau pont.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Vue synthetique : nom, etat, adresse MAC
ip -br link

# Une carte, avec ses noms alternatifs
ip link show ens18

# La correspondance carte <-> pont, telle que Proxmox l'a ecrite
grep -E "iface|bridge-ports" /etc/network/interfaces
```

Ce support est la propriété de Dawan France.

Résultat attendu sur `pxe211` (Proxmox VE 9.2.21) :

```
root@pxe211:~# ip -br link
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
ens18            UP             bc:24:11:01:a1:11 <BROADCAST,MULTICAST,UP,LOWER_UP>
ens19            DOWN           bc:24:11:01:b1:11 <BROADCAST,MULTICAST>
ens20            DOWN           bc:24:11:01:c1:11 <BROADCAST,MULTICAST>
ens21            DOWN           bc:24:11:01:d1:11 <BROADCAST,MULTICAST>
ens22            DOWN           bc:24:11:01:e1:11 <BROADCAST,MULTICAST>
vmbr0            UP             bc:24:11:01:a1:11 <BROADCAST,MULTICAST,UP,LOWER_UP>

root@pxe211:~# ip link show ens18
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master vmbr0
state UP mode DEFAULT group default qlen 1000
    link/ether bc:24:11:01:a1:11 brd ff:ff:ff:ff:ff:ff
    altname enp6s18
    altname enxbc241101a111

root@pxe211:~# grep -E "iface|bridge-ports" /etc/network/interfaces
iface lo inet loopback
iface ens18 inet manual
iface vmbr0 inet static
    bridge-ports ens18
iface ens19 inet manual
```

Trois choses se lisent d'un coup. **`vmbr0` porte la même adresse MAC que `ens18`** : un pont Linux adopte celle de son premier port. **`master vmbr0`** dans `ip link show ens18`, et **`bridge-ports ens18`** dans le fichier, disent la même chose : `ens18` alimente `vmbr0`.

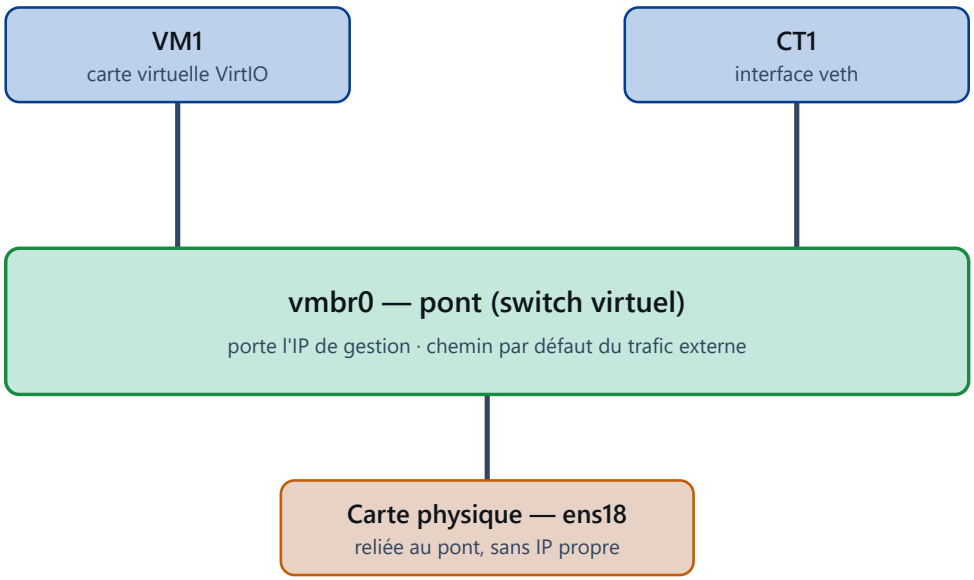
`ens19` à `ens22` sont `DOWN`, sans `UP` dans leurs drapeaux : les cartes existent, mais rien ne les utilise encore. Les noms `altname` sont ceux de la colonne **Alternative Names** de l'interface.

> **Ne renommez pas vos interfaces pour « faire propre ».** La tentation existe devant un nom comme `enxbc241101a111`, l'un des noms alternatifs de `ens18`. Mais le nom de la carte est référencé dans `/etc/network/interfaces`, et parfois dans des règles de pare-feu.
>
> Un renommage incomplet coupe le nœud du réseau. Il faut alors un accès physique ou IPMI (*Intelligent Platform Management Interface*, la carte de gestion à distance du serveur) pour le

Dawan Ce support est la propriété de Dawan France.

récupérer. Si vous devez vraiment le faire, faites-le avant de configurer quoi que ce soit d'autre.

Le bridge vmbr0 — un switch Ethernet virtuel



Créer ou modifier un bridge — via l'interface graphique :

Nœud → Network → Create → Linux Bridge

Ce que vous lisez sur cet écran. Un seul champ décide du comportement du pont : **Bridge ports**. S'il contient une carte physique, le pont communique avec le réseau ; s'il reste vide, le pont isole totalement les machines qui y sont branchées.

IPv4/CIDR n'est renseigné que si l'hôte lui-même doit être joignable sur ce pont. **Autostart** le fait remonter à chaque démarrage, et **Comment** — champ libre — vous évitera six mois plus tard de chercher à quoi sert ce pont.

C'est le même écran dans les trois cas que vous allez rencontrer : `vmbr0` pour l'administration, `vmbr1` pour le cluster, `vmbr9` pour un réseau isolé. Seul le contenu de **Bridge ports** change vraiment d'un cas à l'autre.

Champ	Valeur
Name	`vmbr0`
IPv4/CIDR	`<ip-de-votre-noeud>/24`

Ce support est la propriété de Dawan France.

Champ	Valeur
Gateway (IPv4)	`<ip-de-la-passerelle>`
Bridge ports	`ens18` (interface physique)
Autostart	✓
VLAN aware	*(laisser décoché pour l'instant)*

Cliquez **Create** puis le bouton **Apply Configuration** en haut de la liste d'interfaces — sans ce second clic, les changements sont enregistrés mais pas encore actifs.

Les cinq cartes de votre nœud de lab

Votre serveur n'a pas deux cartes réseau, mais **cinq**. Ce n'est pas une générosité gratuite : chacune correspond à un flux qu'une infrastructure Proxmox sérieuse sépare physiquement, et vous les utiliserez toutes avant la fin de la formation.

Carte	Rattachée à	Le flux qu'elle porte	Vue au chapitre
1re	`vmbr0`	côté hôte Administration : l'interface web, SSH, le trafic des VMs	1 à 5
2e	`cluster`	Corosync — les battements de cœur entre nœuds	3 (`vmbr1`), puis 4
3e	`CephClus`	Réplication interne de Ceph, entre OSD	4
4e	`CephPub`	Accès des clients à Ceph	4
5e	`migra`	Migration à chaud des machines	4

Pourquoi séparer. Les trois derniers flux ont un point commun : ils sont **volumineux et par rafales**. Une réplication Ceph ou une migration à chaud sature un lien pendant quelques minutes.

Si Corosync partage ce lien, ses messages arrivent en retard et les nœuds se croient morts les uns les autres. Le cluster se fracture : un `fencing` en pleine journée, déclenché par une simple sauvegarde. Séparer les flux n'est donc pas une élégance d'architecte : c'est ce qui empêche une opération de routine de faire tomber le cluster.

> **Ce que vous voyez dépend de l'étage où vous regardez.** Votre nœud de formation est lui-même une machine virtuelle, installée sur un serveur Proxmox physique. Les noms `cluster` , `CephClus` , `CephPub` et `migra` sont ceux des ponts de **l'hôte physique** — vous ne les verrez pas sous ces noms.

Ce support est la propriété de Dawan France.

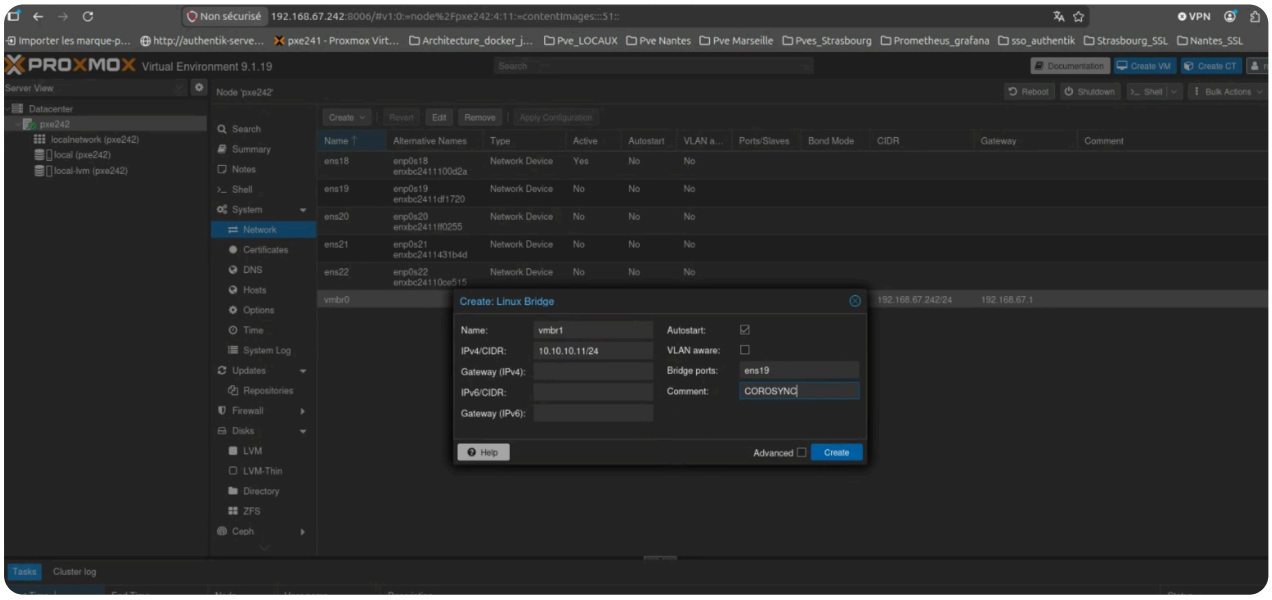
Ce que **vous** voyez, ce sont cinq cartes réseau ordinaires. À vous de les rebaptiser en ponts sur votre propre nœud : `vmbr1` pour le cluster, à l'atelier 1. La correspondance carte ↔ flux vous est donnée par l'ordre : la première porte l'administration, la deuxième le cluster, et ainsi de suite.

N'attachez jamais deux ponts à la même carte. Une interface ne peut appartenir qu'à un seul pont. L'interface web enregistre pourtant la saisie sans protester.

C'est au moment de **Apply Configuration** que l'application échoue, avec un message du type `is already a member of a bridge` dans le journal de la tâche. Ou bien le second pont reste sans port. Avant de créer un pont, vérifiez que la carte visée n'apparaît dans la colonne `Ports/Slaves` d'aucun autre.

Le pont dédié au cluster (`vmbr1`)

Un serveur destiné à rejoindre un cluster reçoit presque toujours un **second pont**, monté sur une carte distincte et réservé aux échanges entre nœuds. C'est le cas sur votre lab : la première carte porte déjà `vmbr0`, la deuxième attend d'être utilisée — c'est celle du réseau `cluster` du tableau ci-dessus. Relevez son nom exact avant de continuer : selon la version, elle s'appelle `nic1`, `ens19` ou autrement.



Le pont `vmbr1` en cours de création : `ens19` dans **Bridge ports**, l'adresse Corosync du nœud dans **IPv4/CIDR**, `COROSYNC` dans **Comment**. Et **Gateway laissé vide** : ce réseau ne route rien vers l'extérieur, il ne fait que relier les serveurs entre eux.

Ce support est la propriété de Dawan France.

Champ	Valeur
Name	`vmbr1`
IPv4/CIDR	**l'adresse Corosync de votre nœud**, en `/24`
Gateway (IPv4)	*(laisser vide)*
Bridge ports	`ens19`
Autostart	✓
Comment	`COROSYNC`

Les trois nœuds partagent le même réseau, une adresse chacun — même préfixe, dernier octet différent :

Nœud	IP Corosync
node1	`10.<site>.<N+10>.11/24`
node2	`10.<site>.<N+10>.12/24`
node3	`10.<site>.<N+10>.13/24`

> **Prenez l'adresse de votre nœud, pas celle du voisin.** Le préfixe est identique pour les trois : c'est ce qui les met sur le même réseau. Seul le dernier octet change ; il est indiqué en face du nom de votre serveur sur le document partagé de la salle.

Pourquoi ce réseau séparé. Corosync échange en permanence de petits messages entre les nœuds et supporte très mal la latence : quelques dizaines de millisecondes suffisent à lui faire croire qu'un nœud est mort.

Une sauvegarde ou une migration à chaud qui sature le réseau d'administration provoquerait alors une fausse détection de panne. C'est l'adresse de ce pont que la création du cluster désigne comme **Link 0**.

> `vmbr1` **est réservé au cluster.** Pour un réseau de lab isolé, prenez un autre numéro — `vmbr9` dans ce support — et laissez **Bridge ports** vide. Vider `vmbr1` de son port physique couperait Corosync.

La même configuration est écrite automatiquement dans `/etc/network/interfaces`, le fichier de configuration réseau standard sous Debian :

Ce support est la propriété de Dawan France.

```
# Configuration typique après installation de Proxmox
auto vmbr0
iface vmbr0 inet static
    address <ip-de-votre-noeud>/24   # IP de l'hyperviseur lui-même
    gateway <ip-de-la-passerelle>    # Passerelle par défaut
    bridge-ports ens18           # Interface physique attachée au bridge
    bridge-stp off               # STP désactivé (inutile sur un seul nœud)
    bridge-fd 0                  # Forward delay à 0 pour éviter les délais de
bascule
```

`bridge-stp off` désactive le Spanning Tree Protocol (STP : le protocole qui évite les boucles entre switchs), inutile ici puisqu'il n'y a qu'un seul switch virtuel et aucun risque de boucle. `bridge-fd 0` supprime le délai de 15 secondes que STP impose normalement avant de faire transiter du trafic. Avec STP actif, ce délai s'applique deux fois (écoute, puis apprentissage) : environ 30 secondes pendant lesquelles les invités du pont sont injoignables au démarrage.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Appliquer un changement réseau sans redémarrer la machine
ifreload -a
```

> **Erreur fréquente :** modifier `/etc/network/interfaces` à la main sans passer par `ifreload -a` ou sans rebooter ne change rien à la configuration active. Le fichier est lu au démarrage ou lors d'un rechargement explicite.

Qui écrit ce fichier : l'interface, `pvesh`, ou vous ?

C'est Proxmox qui gère `/etc/network/interfaces`. Dès qu'il le réécrit, il y place cet en-tête (sur `pxe212`) :

```
# network interface settings; autogenerated
# Please do NOT modify this file directly, unless you know what
# you're doing.
```

La règle : on modifie le réseau par l'interface web, ou par `pvesh`, qui appelle la même API. L'édition à la main reste réservée à ce que l'interface ne sait pas faire, comme les règles de NAT

Dawan Ce support est la propriété de Dawan France.

du §2.4.

Les modifications ne s'appliquent pas tout de suite. Proxmox les écrit d'abord dans `/etc/network/interfaces.new`. Le bouton **Apply Configuration**, ou `pvesh set /nodes/<nœud>/network`, les applique.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Créer un pont sans carte physique (équivalent de Create → Linux Bridge)
pvesh create /nodes/pxe212/network --iface vmbr8 --type bridge --autostart 1 --
comments "essai pvesh"
ip -br link show vmbr8                      # pas encore actif
pvesh set /nodes/pxe212/network             # = Apply Configuration
ip -br link show vmbr8                      # actif
pvesh delete /nodes/pxe212/network/vmbr8    # supprimer le pont
pvesh set /nodes/pxe212/network             # appliquer la suppression
```

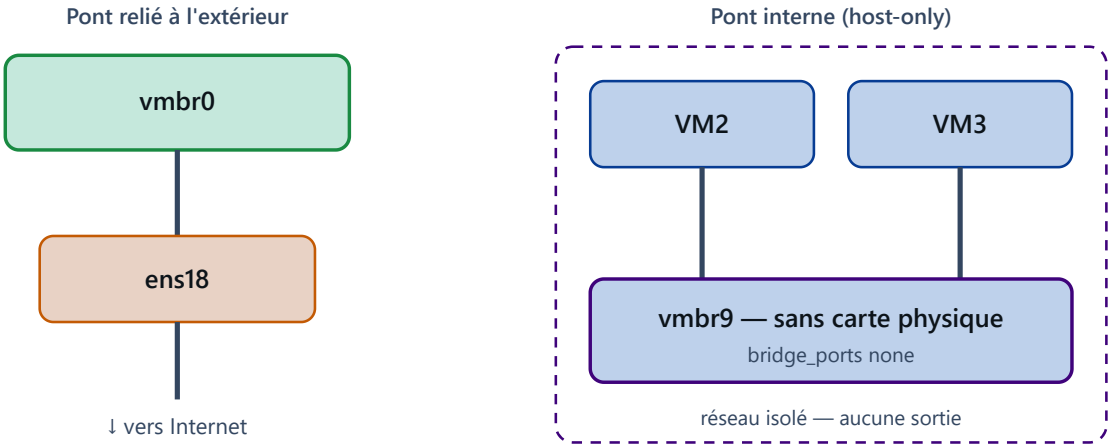
Résultat attendu (nœud `pxe212`, Proxmox VE 9.2.21) :

```
Device "vmbr8" does not exist.
UPID:pxe212:0000EB83:001E055B:6AC517D7:srvreload:networking:root@pam:
vmbr8            UNKNOWN        a2:f5:7a:1c:cb:b5 <BROADCAST,MULTICAST,UP,LOWER_UP>
```

Avant l'application, le pont n'existe que dans `interfaces.new` : `Device "vmbr8" does not exist`. Après `pvesh set`, il est actif. La ligne `UPID…srvreload:networking` est la tâche de rechargement, la même que celle qu'affiche l'interface.

Ce support est la propriété de Dawan France.

Pont avec sortie (vmbr0) vs pont isolé (vmbr9)



Vous créez `vmbr9` sans port physique et y branchez une VM. Elle joint les autres invités de `vmbr9`, mais ni le nœud voisin, ni Internet. Est-ce une panne ?

▼ Réponse

Non, c'est le rôle d'un pont sans carte physique. Il fonctionne comme un commutateur débranché du reste du réseau : ses invités se voient entre eux, rien d'autre. Pour faire sortir ce réseau, il faut donner une adresse au pont et ajouter une règle **NAT** (section 2.4). Pour le relier au réseau physique, il faudrait lui attacher une carte.

Revenons à la définition de base : un **commutateur** (*switch*) relie des machines sur un même réseau et leur fait passer les trames. Un pont Linux est un commutateur en logiciel, dans le noyau du nœud.

Terme	Ce que c'est	Dans le laboratoire
Pont (*bridge*)	Un commutateur logiciel dans le noyau du nœud	`vmbr0`
Carte physique (port du pont)	La carte réelle, qui relie le pont au réseau extérieur	`ens18`, dans `bridge-ports`
Carte d'une VM ou d'un conteneur	L'interface virtuelle de l'invité (`tap` pour une VM, `veth` pour un conteneur)	`net0` de chaque VM et CT
Adresse du pont	L'adresse par laquelle on joint le nœud (interface web, Shell)	`10.<site>.<N>.11/24` sur `vmbr0`

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
Pont sans carte physique	Un réseau isolé : les invités se voient entre eux, rien d'autre	`vmbr9` (atelier 1)
`ifreload -a`	Applique `/etc/network/interfaces` sans redémarrer (**Apply Configuration**)	Après chaque modification du réseau

Un pont sans carte physique n'est pas en panne : c'est un réseau privé. Pour le faire sortir, on lui donne une adresse et une règle de NAT (§2.4).

Où voir le pont sur votre nœud :

Notion	Où la voir
Les ponts et leurs cartes	**Nœud** → **System** → **Network** : colonnes *Ports/Slaves*, *Active*, *CIDR*
Le fichier qui les décrit	`/etc/network/interfaces` : les lignes `iface` et `bridge-ports`
L'état des cartes	`ip -br link`
Le pont d'une VM	La VM → **Hardware** → **Network Device** : `bridge=vmbr0`

Et à l'intérieur des invités ?

Proxmox fournit le câble virtuel et le pont. L'adresse IP, elle, dépend de l'invité :

Invité	Qui configure l'adresse IP
Conteneur	Proxmox, au moment de la création (champ **Network**, ou `--net0 …,ip=`) : il écrit lui-même le fichier réseau du conteneur
VM	Le système de la VM : `/etc/network/interfaces` sous Debian, `/etc/netplan/` sous Ubuntu, ou cloud-init (chapitre 2)

Dans une VM Debian en q35, la carte VirtIO s'appelle en général `ens18`, comme sur le nœud. On le vérifie avec `ip -br link` dans la console de la VM, avant d'écrire sa configuration.

Bonnes pratiques

- **Un segment, un pont, un rôle.** `vmbr0` pour l'administration et la sortie, un pont par réseau interne (applicatif, base de données). Des composants sans lien ne se voient pas, et les règles de pare-feu restent simples.
- **L'adresse IP sur le pont, jamais sur la carte physique.** La carte reste en `inet manual`.
- **Tester aussi ce qui doit être bloqué.** Un réseau isolé se vérifie par un ping qui doit échouer, pas seulement par ceux qui réussissent.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Dans le Shell du nœud :

```
# Exemple : le conteneur 100 ne doit PAS joindre 10.0.1.30
pct exec 100 -- ping -c 2 -W 2 10.0.1.30              # deux essais, deux secondes d'attente chacun
echo "code : $?"                                      # 0 : joint ; 1 : isolé
```

Résultat attendu (les lignes qui comptent) : `2 packets transmitted, 0 received, 100% packet loss`, puis `code : 1`. Le ping échoue : l'isolation tient. Un code `0` signalerait au contraire une isolation défaillante.

À retenir : Proxmox s'appuie sur la pile réseau de Linux. La carte physique est rattachée à un **pont** (`vmbr0`), qui joue le rôle de commutateur virtuel pour les VM et les conteneurs. Un pont sans carte physique fait un réseau isolé.

`vmbr1` est réservé au cluster. Le réseau se modifie par l'interface ou `pvesh`, et s'applique par **Apply Configuration** (`pvesh set /nodes/<nœud>/network`, ou `ifreload -a` après une édition à la main).

> **▶ À pratiquer maintenant — Atelier 1**
>
> Lire la liste des interfaces du nœud, puis construire deux ponts aux rôles opposés : `vmbr9` qui isole, `vmbr1` qui reliera les serveurs du cluster. C'est le moment de vérifier comment vos cartes s'appellent réellement.

Ce support est la propriété de Dawan France.

2. VLAN et NAT : segmenter et faire sortir

Un bridge par réseau physique ne suffit vite plus : on veut séparer plusieurs réseaux sur le même câble, et faire sortir sur Internet des machines placées sur un réseau privé. Ce sont les deux outils de cette section : le **VLAN** et le **NAT**.

2.1 VLAN — segmenter sans multiplier les câbles

Un VLAN permet de **segmenter un réseau physique** en plusieurs réseaux logiques. Chaque VLAN est identifié par un **ID numérique**, de 1 à 4094. Deux notions structurent tout le sujet :

Le **VLAN natif** (non tagué) correspond au trafic envoyé sans balise 802.1Q — c'est souvent le VLAN par défaut d'un port. Le **VLAN tagué** encapsule le trafic avec un identifiant VLAN explicite ; cela nécessite que le switch et l'OS le prennent en charge.

Prenons un exemple concret avec un switch configuré ainsi :

Port	Type de VLAN	ID VLAN	Utilisation
1	Natif (untagged)	1	Administration
2	Tagged	10, 20	VM réseau privé, stockage

Le port 1 reçoit du trafic non tagué (VLAN 1). Le port 2 transporte plusieurs VLANs identifiés par des balises (10, 20) — la machine connectée doit donc gérer le **trunk 802.1Q**.

Chaque bridge de Proxmox peut transporter un ou plusieurs VLANs, et les VMs peuvent être attachées à un bridge avec un VLAN ID explicite. La configuration côté hyperviseur doit toujours correspondre à celle du switch physique.

> **Pourquoi éviter le VLAN 1 comme VLAN natif :** le VLAN 1 est activé par défaut sur presque tous les switchs. Il sert à la gestion, aux protocoles de découverte et à tout trafic non tagué.
>
> Le laisser actif peut entraîner des fuites réseau involontaires ou un accès non souhaité. Choisir un VLAN natif explicite (par exemple VLAN 10 ou 100) rend l'architecture plus claire et plus sécurisée.

Pour que Proxmox comprenne ces tags, le bridge doit être déclaré **VLAN-aware**.

Dawan
Ce support est la propriété de Dawan France.

Via l'interface graphique :

> **Nœud → Network → vmbr0 → Edit** → cocher **VLAN aware** → **OK** → **Apply Configuration**

Une fois le bridge VLAN-aware, assigner un VLAN à une VM ou un conteneur est immédiat : **VM → Hardware → Network Device → Edit → VLAN Tag =** `10` (remplacer `10` par l'ID voulu).

En CLI, c'est l'option `bridge-vlan-aware yes` accompagnée de `bridge-vids 2-4094` qui autorise tous les tags VLAN possibles dans `/etc/network/interfaces` :

```
# Bridge vmbr0 rendu VLAN-aware (modif dans /etc/network/interfaces)
auto vmbr0
iface vmbr0 inet static
    address <ip-de-votre-noeud>/24
    gateway <ip-de-la-passerelle>
    bridge-ports ens18
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes        # Active la compréhension des tags VLAN
    bridge-vids 2-4094           # Autorise tous les VLANs de 2 à 4094
```

Revenons à la définition de base : un **réseau local** est l'ensemble des machines qui reçoivent les diffusions (*broadcast*) les unes des autres. Un VLAN découpe un même réseau physique en plusieurs de ces réseaux.

Terme	Ce que c'est	Dans le laboratoire
VLAN	Un réseau logique, isolé des autres sur le même câble	L'exemple du cours : VLAN 10
ID de VLAN	Un numéro de 1 à 4094	`10`
Étiquette 802.1Q (*tag*)	Quatre octets ajoutés à la trame, qui portent l'ID	Invisible pour la VM
Trunk (port tagué)	Un lien qui transporte plusieurs VLAN, étiquetés	Le port du switch physique relié au nœud
Port access	Un port d'un seul VLAN : l'étiquette est retirée avant la livraison	La carte d'une VM avec **VLAN Tag = 10**
VLAN natif	Le trafic non étiqueté d'un port	VLAN 1 par défaut sur la plupart des switchs

Dawan
Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
Pont `vlan-aware`	Un pont qui pose, lit et retire les étiquettes	`vmbr0` après **VLAN aware** coché

Un même lien trunk transporte tous les VLAN, et chaque invité ne reçoit que le trafic de son VLAN. La VM ne voit jamais l'étiquette : le pont la pose en sortie et la retire en entrée (§2.2).

Où voir un VLAN sur un nœud :

Notion	Où la voir
Le pont `vlan-aware`	**Nœud** → **System** → **Network** → **vmbr0** → **Edit** : case **VLAN aware** ; dans `/etc/network/interfaces` : `bridge-vlan-aware yes` et `bridge-vids 2-4094`
Le VLAN d'une VM	La VM → **Hardware** → **Network Device** : champ **VLAN Tag** ; dans sa configuration : `net0: virtio=…,bridge=vmbr0,tag=10`
Les VLAN autorisés sur chaque port du pont	`bridge vlan show` dans le Shell du nœud

2.2 Comprendre le flux d'une trame taguée

Un bridge Linux configuré `vlan-aware` fonctionne comme un switch manageable. Il comprend les VLANs tagués et peut filtrer par VLAN. Il peut aussi tagger ou détagger les paquets selon les paramètres du port.

Flux entrant — du physique vers la VM. Un paquet arrive sur l'interface physique (`ens18`).

Si le bridge est configuré pour accepter le VLAN correspondant (VLAN 10 par exemple), il laisse passer ce paquet. Le bridge examine ensuite le port virtuel associé à la VM (`tapX`).

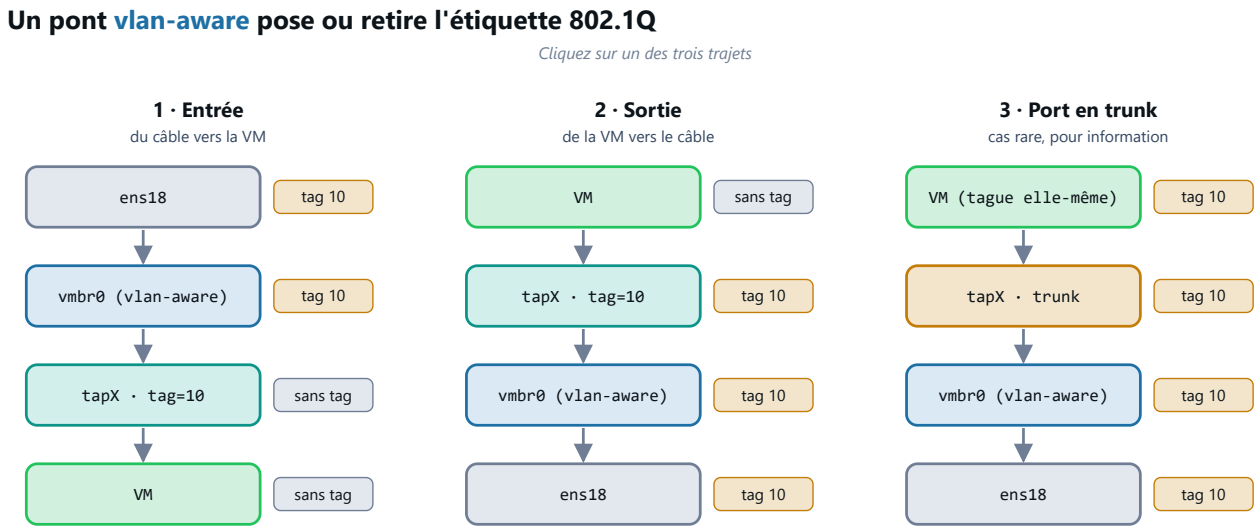
Si ce port est en mode access pour le VLAN 10, le bridge détague le paquet et le livre nu à la VM. S'il est en mode trunk, le tag est conservé et le paquet est remis à la VM avec son tag.

Flux sortant — de la VM vers l'extérieur. Quand une VM envoie un paquet, le bridge le reçoit via son interface `tapX` .

Si le port est en mode access avec PVID (*Port VLAN ID*, le VLAN par défaut du port) 10, l'interface tague elle-même le paquet en VLAN 10. C'est la configuration matérielle de la VM dans Proxmox. Le paquet part ensuite vers l'interface physique avec ce tag.

Ce support est la propriété de Dawan France.

Si le port est en trunk, le bridge n'ajoute aucun tag — il attend que la VM ait déjà tagué le paquet elle-même.



La VM ne voit jamais l'étiquette : c'est le port tapX, réglé par « VLAN Tag = 10 » dans Hardware, qui la gère.
Sur le câble (ens18), la trame porte le tag 10 ; le switch en face doit transporter ce VLAN (port « tagged »).
Condition : le pont est déclaré VLAN aware (bridge-vlan-aware yes, bridge-vids 2-4094).

Les trois trajets d'une trame du VLAN 10. En entrée, le pont retire l'étiquette avant de livrer la VM ; en sortie, le port de la VM (« VLAN Tag = 10 ») l'ajoute. La VM ne voit donc jamais l'étiquette.

Le troisième trajet, en trunk, montre le cas rare où la VM tague elle-même. Cliquez sur un trajet pour le détail.

> **En pratique :** il est très rare de configurer un port de VM en mode trunk pour un seul VLAN. On préfère gérer le tag directement via les paramètres réseau de la VM (`VLAN Tag = 10`) — c'est la méthode utilisée dans ce chapitre et dans les ateliers.

- 🎥 apalrd's adventures — Proxmox networking : VLANs, bridges et bonds (vidéo en anglais, passage sur les VLAN)

2.3 NAT — faire sortir un réseau privé sur Internet

Le **NAT** (Network Address Translation) permet de faire transiter des paquets IP d'un réseau privé vers un réseau public en modifiant l'adresse IP source ou destination. C'est ce qui permet à plusieurs machines privées (`192.168.x.x`) d'accéder à Internet avec une seule adresse IP publique. Trois variantes structurent le sujet :

Ce support est la propriété de Dawan France.

Le **Source NAT** (SNAT : *Source Network Address Translation*) modifie l'adresse IP source des paquets sortants — c'est ce qui fait sortir un réseau privé sur Internet avec une IP publique unique. Exemple : `192.168.1.100` devient `203.0.113.1` pour Internet.

Le **Destination NAT** (DNAT : *Destination Network Address Translation*) modifie l'adresse IP de destination des paquets entrants. Il sert à rediriger des ports vers des machines internes, par exemple pour un hébergement web en DMZ (*Demilitarized Zone*, réseau exposé isolé du réseau interne). Exemple : `203.0.113.1:80` redirige vers `192.168.1.10:80`.

Le **PAT (Port Address Translation)**, aussi appelé NAT overload, est une extension de SNAT ou DNAT qui ajoute la modification des ports pour permettre plusieurs connexions sur une même IP. En sortie, le PAT est généralement transparent : c'est le mécanisme qui permet au firewall d'ouvrir un port dynamique pour savoir vers qui rediriger une réponse — donc du SNAT + PAT.

En entrée, on ouvre un port fixe et non utilisé pour exposer un service en continu (SSH, reverse proxy sur 443) — donc du DNAT + PAT.

Terme	Traduit	Fonction principale
NAT	Traduction d'adresse	Concept global de substitution IP
SNAT	Source NAT	Modifier l'IP source en sortie
DNAT	Destination NAT	Modifier l'IP de destination en entrée
PAT	Traduction de ports	Multiplexer plusieurs machines via une IP

2.4 Mettre en place le NAT pour des VMs isolées

Un cas pratique fréquent : un bridge interne sans accès direct à Internet (`vmbr9` , sans interface physique attachée). Ses VMs doivent malgré tout pouvoir sortir vers Internet pour leurs mises à jour. Le bridge devient alors la passerelle de ce réseau interne, et l'hôte fait le NAT vers l'extérieur.

La méthode que recommande Proxmox attache le routage et la règle NAT **au pont lui-même**, dans `/etc/network/interfaces` . Ils naissent quand le pont monte et disparaissent quand il descend, sans service supplémentaire à installer.

```
# /etc/network/interfaces — pont interne 10.0.0.0/24, NATé derrière vmbr0
auto vmbr9
iface vmbr9 inet static
    address 10.0.0.1/24             # le nœud est la passerelle de ce réseau
    bridge-ports none               # aucune carte physique : réseau interne
    bridge-stp off
```

Ce support est la propriété de Dawan France.

```
    bridge-fd 0
    # Routage activé, puis traduction d'adresse des paquets qui sortent par vmbr0
    post-up   echo 1 > /proc/sys/net/ipv4/ip_forward
    post-up   iptables -t nat -A POSTROUTING -s '10.0.0.0/24' -o vmbr0 -j MASQUERADE
    post-down iptables -t nat -D POSTROUTING -s '10.0.0.0/24' -o vmbr0 -j MASQUERADE
```

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

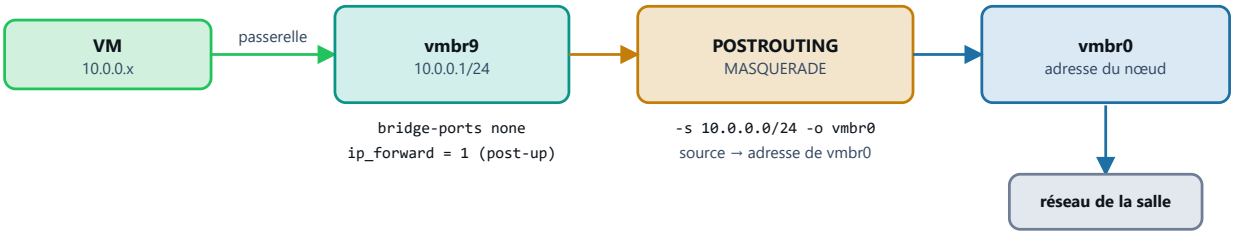
```
# Appliquer sans redémarrer, puis vérifier que la règle est posée
ifreload -a
iptables -t nat -S POSTROUTING   # la règle MASQUERADE doit apparaître
```

La règle se lit de gauche à droite. Table `nat`, chaîne `POSTROUTING` : juste avant que le paquet quitte le nœud.

Tout paquet **venant de** `10.0.0.0/24` (`-s`) et **sortant par** `vmbr0` (`-o`) prend l'adresse de `vmbr0` (`MASQUERADE`). Le `-s` compte : sans lui, la règle traduirait aussi le trafic du nœud lui-même.

Un réseau interne qui sort sous l'adresse du nœud

Cliquez sur une étape du trajet



Le pont vmbr9 n'a aucune carte physique : ses VM ne sortent que parce que le nœud route et traduit.
Routage et règle NAT sont attachés au pont (post-up / post-down) : ils vivent et meurent avec lui.

Le trajet d'un paquet qui quitte une VM du pont interne `vmbr9`. La VM l'envoie à sa passerelle (le nœud, 10.0.0.1), et le nœud le route.

Puis la règle `MASQUERADE` remplace l'adresse source par celle de `vmbr0`, juste avant la sortie. Cliquez sur une étape pour la commande qui la met en place.

Ce support est la propriété de Dawan France.

> **Deux méthodes répandues à éviter sur Proxmox.** Écrire `net.ipv4.ip_forward=1` dans `/etc/sysctl.conf` ne tient plus sur Debian 13 : ce fichier n'est plus relu au démarrage, il faut un fichier dans `/etc/sysctl.d/` — d'où l'intérêt de la ligne `post-up`. Et le paquet `iptables-persistent` recharge au démarrage un jeu de règles figé, qui entre en concurrence avec celles que le pare-feu Proxmox (`pve-firewall`) réécrit lui-même : les deux se marchent dessus.

Résultat attendu de `iptables -t nat -S POSTROUTING` : une ligne `-A POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j MASQUERADE`. Le test se fait ensuite depuis une VM branchée sur `vmbr9`, d'adresse `10.0.0.x/24` et de passerelle `10.0.0.1` : elle doit joindre une machine extérieure au pont.

Le NAT MASQUERADE traduit silencieusement l'adresse interne de la VM en adresse de l'hôte au moment où le paquet sort par `vmbr0`. La VM n'a aucune adresse routable, et pourtant elle atteint internet — c'est tout l'objet de la manœuvre.

> **Sur le lab, ce test dépend du routage en place autour de vous.** Le mécanisme décrit ici est celui du NAT, et il fonctionne : la VM sort bien par `vmbr0` avec l'adresse de l'hôte. Mais si le réseau de la salle ne route pas lui-même vers internet, le `ping` échouera un cran plus loin, sans que la configuration soit en cause.
>
> Pour vérifier que le NAT fait son travail, visez une machine du réseau de la salle plutôt qu'une adresse publique. Observez `iptables -t nat -L POSTROUTING -v` : le compteur de la règle `MASQUERADE` s'incrémente à chaque paquet traduit.

- 📖 Documentation Proxmox — Network Configuration
- 📖 Configuration réseau avancée sous Proxmox — Documentation Resinfo

À retenir : un **VLAN** découpe un même réseau physique en réseaux logiques par une étiquette 802.1Q ; un pont `vlan-aware` pose ou retire l'étiquette selon le `tag` de chaque carte. Le **NAT** (SNAT / masquerade) fait sortir un réseau privé sous l'adresse du nœud ; sur Proxmox, on l'attache au pont dans `/etc/network/interfaces` (`post-up`).

> **▶ À pratiquer maintenant — Atelier 2**

Dawan
Ce support est la propriété de Dawan France.

Le pont `vmbr9` de l'atelier 1 au travail : VLAN-aware, il sépare deux conteneurs selon leur VLAN ; passerelle NATée, il fait sortir un réseau privé sous l'adresse du nœud. Sans jamais toucher à `vmbr0` .

Ce support est la propriété de Dawan France.

3. Le modèle de stockage de Proxmox : bloc, fichier, LVM-thin

Avant de construire un pool ZFS, de déclarer un stockage iSCSI ou de monter un cluster Ceph, il faut une grille de lecture commune. Tous ces stockages ne se ressemblent pas : certains offrent des **disques bruts**, d'autres des **dossiers partagés**. Cette distinction explique presque tout ce que Proxmox autorise ou refuse — quel format de disque, quels snapshots, où ranger une ISO.

3.1 Trois modèles de stockage : bloc, fichier, objet

Il existe trois grandes façons d'organiser des données sur un support, et Proxmox les rencontre toutes :

Modèle	Unité manipulée	Qui met la structure ?	Exemples
Bloc (*block*)	Des **blocs** de taille fixe (512 o ou 4 Kio), adressés par leur numéro	**Le client** : il pose un système de fichiers, une base, un swap par-dessus	Disque SATA (bus de disque classique) ou NVMe, LUN iSCSI, LVM, zvol ZFS, **Ceph RBD**
Fichier (*file*)	Des **fichiers** rangés dans une arborescence de dossiers	**Le serveur** : le système de fichiers existe déjà, on lit et écrit par un chemin	ext4, XFS, **NFS**, SMB (*Server Message Block*, le partage de fichiers de Windows), aussi appelé CIFS (*Common Internet File System*), CephFS
Objet (*object*)	Des **objets** (données + métadonnées + identifiant) dans un espace plat, par API HTTP	Le service objet : pas de hiérarchie, pas de modification partielle	Amazon S3 (*Simple Storage Service*), MinIO, Ceph RGW (la passerelle objet de Ceph)

Le **bloc** fournit un disque nu, que le client formate lui-même. Le **fichier** fournit une arborescence partagée. L'**objet** fournit des objets entiers, adressés par URL.

Critère	Bloc	Fichier	Objet
Granularité	Le bloc (512 o ou 4 Kio)	Le fichier	L'objet entier
Modification partielle	Oui : on réécrit un bloc	Oui	Non : on réécrit tout l'objet

Ce support est la propriété de Dawan France.

Critère	Bloc	Fichier	Objet
Métadonnées	Aucune, au niveau du volume	Attributs POSIX (droits, dates)	Riches, extensibles
Accès simultané par plusieurs hôtes	Non, sauf système conçu pour (cluster, stockage distribué)	Oui, avec des verrous réseau	Oui, nativement
Protocole	SCSI, NVMe, iSCSI, FC (*Fibre Channel*), RBD	NFS, SMB, POSIX	HTTP/REST (S3)
Latence	La plus faible	Moyenne	Plus élevée (HTTP)
Usage type	Disques de VM, bases de données, swap	Partages, ISO, sauvegardes	Archives, médias, gros volumes de données

Les disques de VM relèvent du bloc ; les ISO et les sauvegardes, du fichier ; l'objet n'apparaît dans Proxmox qu'à la marge (cible S3 de PBS).

Revenons à la définition de base : un **stockage** garde des données ; toute la question est de savoir **qui les range**.

Terme	Ce que c'est	Dans le laboratoire
Bloc	Des blocs numérotés ; la VM y pose son propre système de fichiers	`local-lvm` (LVM-thin), le pool ZFS, un LUN iSCSI
Fichier	Une arborescence de fichiers, tenue par le serveur ; on lit par un chemin	`local` (`/var/lib/vz`), un partage NFS
Objet	Un objet entier, rangé à plat, retrouvé par son identifiant, par HTTP	Absent du laboratoire (cible S3 de PBS)
Type de stockage	Comment Proxmox accède aux données : `lvmthin` , `zfspool` , `dir` , `nfs` , `iscsi` …	Colonne *Type* de **Datacenter** → **Storage**
Contenu	`images` (disques de VM), `rootdir` (conteneurs), `iso` , `vztmpl` , `backup` …	`local` : ISO, templates, sauvegardes ; `local-lvm` : disques
Partagé	Le même stockage, vu par tous les nœuds	`Shared Yes` pour NFS

Sur un stockage bloc, c'est la VM qui gère son système de fichiers. Sur un stockage fichier, c'est le serveur : une ISO ou une sauvegarde y est un fichier ordinaire.

Où voir le modèle de chaque stockage :

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Notion	Où la voir
Type, contenus, partage	**Datacenter** → **Storage** : colonnes *Type*, *Content*, *Shared* ; le fichier `/etc/pve/storage.cfg`
Remplissage de chaque stockage	`pvesm status`
Un disque de VM en mode bloc	`pvesm path local-lvm:vm-101-disk-0` → `/dev/pve/vm-101-disk-0`, un périphérique sans système de fichiers visible (§3.2)
Un stockage en mode fichier	Le dossier `/var/lib/vz` (stockage `local`), ou `/mnt/pve/<nom>` pour un NFS

3.2 Le mode bloc : un disque brut, sans structure

Le **stockage en mode bloc** découpe la capacité en volumes présentés à l'hôte comme un **disque dur brut** : une simple suite de blocs numérotés, **sans aucune structure interne**. C'est au système qui le consomme de décider quoi en faire.

Le formater en ext4, y créer un pool ZFS ou un groupe LVM. Ou le donner **directement** à une VM comme disque virtuel.

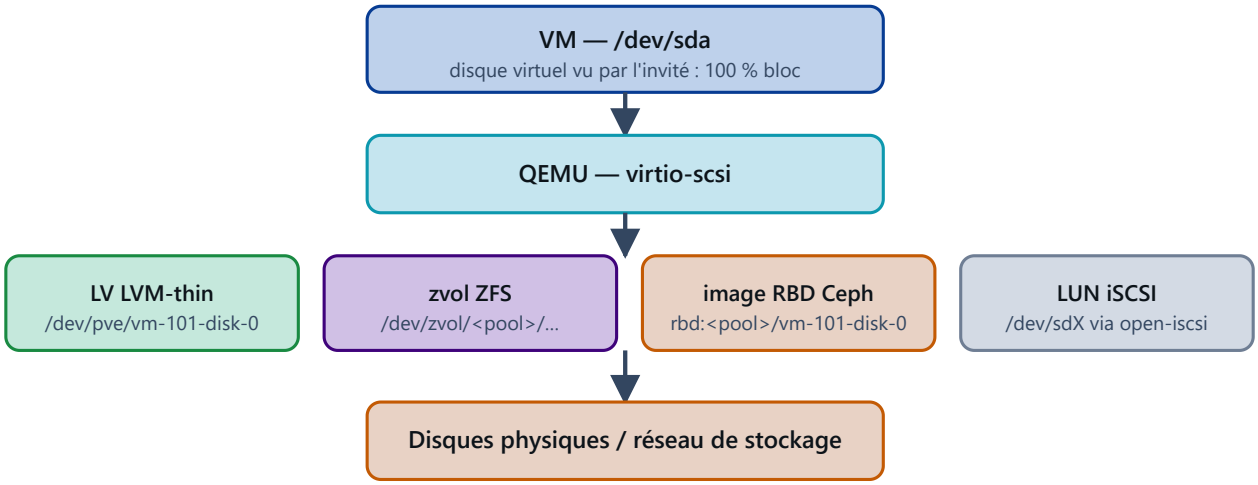
Le stockage bloc ne connaît ni fichiers, ni dossiers, ni permissions. Il ne répond qu'à deux ordres : « lis le bloc n° X » et « écris ceci dans le bloc n° Y ». Cette absence de couche intermédiaire explique ses **performances** — latence faible, IOPS élevés — et sa polyvalence, au prix d'une richesse fonctionnelle nulle au niveau du volume lui-même.

Pour une VM, le chemin d'un bloc traverse plusieurs étages, et **chacun reste en mode bloc**. D'abord le disque `/dev/sda` vu par l'invité, puis QEMU qui transmet les écritures.

Ensuite le volume que Proxmox a attribué : un volume LVM-thin, un zvol ZFS, une image RBD Ceph ou un LUN iSCSI. Enfin, les disques physiques.

Ce support est la propriété de Dawan France.

Comment un bloc descend jusqu'au disque



À chaque étage : toujours des blocs, jamais de système de fichiers intermédiaire — d'où le format raw imposé.

On le constate directement sur le nœud : un disque de VM rangé dans `local-lvm` n'est pas un fichier, c'est un périphérique bloc.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le chemin réel d'un disque de VM stocké sur local-lvm
pvesm path local-lvm:vm-101-disk-0
```

Résultat attendu sur `pxe211` (la VM 101 du chapitre 2) :

```
/dev/pve/vm-101-disk-0
```

`/dev/pve/vm-101-disk-0` est un **volume logique** du groupe LVM `pve` : un périphérique sous `/dev`, sans extension, sans système de fichiers visible de l'hôte. C'est la VM qui, de l'intérieur, y a posé le sien.

3.3 Ce que le mode bloc implique dans Proxmox

Ce modèle a des conséquences très concrètes, que vous retrouverez à chaque création de disque :

- **Format de disque imposé : `raw`.** Un volume bloc *est déjà* un disque brut ; on ne peut pas y déposer un fichier `qcow2`, puisqu'il n'y a pas de système de fichiers pour l'accueillir.

Ce support est la propriété de Dawan France.

Les fonctions qu'apporterait `qcow2` — snapshots, allocation à la demande — sont alors assurées par **la couche bloc elle-même**. C'est le *thin provisioning* pour LVM-thin (assuré par `dm-thin`, un module du *device mapper* du noyau Linux), et la copie sur écriture pour ZFS et Ceph.

- **Snapshots** : possibles **si le backend bloc les fournit** (LVM-thin, ZFS, Ceph RBD) ; **impossibles** sur du LVM « épais » (*thick*) ou sur un LUN iSCSI brut sans surcouche.
- **Performance** : chemin d'accès court, donc latence faible et IOPS élevés. C'est le bon choix pour les **disques de VM**, surtout pour les bases de données.
- **Partage** : un même volume bloc ne peut **pas** être monté en lecture-écriture par plusieurs nœuds à la fois, sous peine de corruption.

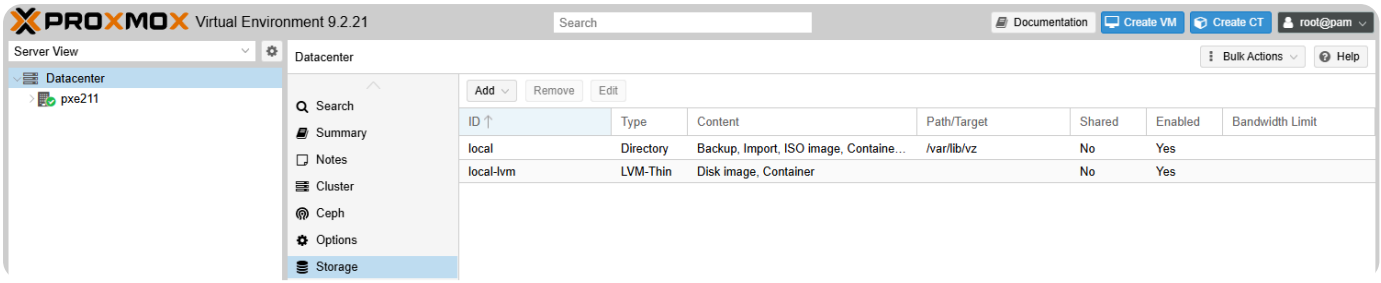
  Il faut alors une couche qui coordonne les nœuds. Ce peut être LVM en mode partagé (§5.2) ou un système de fichiers de cluster (OCFS2 — *Oracle Cluster File System* —, GFS2 — *Global File System* —). Ce peut être aussi un stockage **distribué** conçu pour cela, comme **Ceph RBD**.
- **ISO, templates, sauvegardes** : ce sont des **fichiers**. Ils ne peuvent pas vivre sur un stockage purement bloc ; il leur faut un stockage **fichier** (`dir`, NFS, CIFS) ou l'entrepôt de Proxmox Backup Server.

	Stockage bloc (LVM, LVM-thin, zvol ZFS, Ceph RBD, iSCSI)	**Stockage fichier (dir, NFS, CIFS, CephFS)**
Format de disque	`raw` obligatoire	`raw` ou `qcow2`
Snapshots	Fournis par le backend, s'il les supporte	Par `qcow2`, ou par le backend
Contenus possibles	Disques de VM et de conteneurs uniquement	Disques, **et** ISO, templates, sauvegardes, snippets
Accès	Rapide, sans système de fichiers intermédiaire	Souple, accueille aussi les fichiers d'exploitation

3.4 Un stockage Proxmox : un type, des contenus, un partage

Pour Proxmox, un **stockage** est une entrée de `/etc/pve/storage.cfg` — fichier répliqué par `pmxcfs`, donc identique sur tous les nœuds d'un cluster. Chaque entrée porte trois informations :

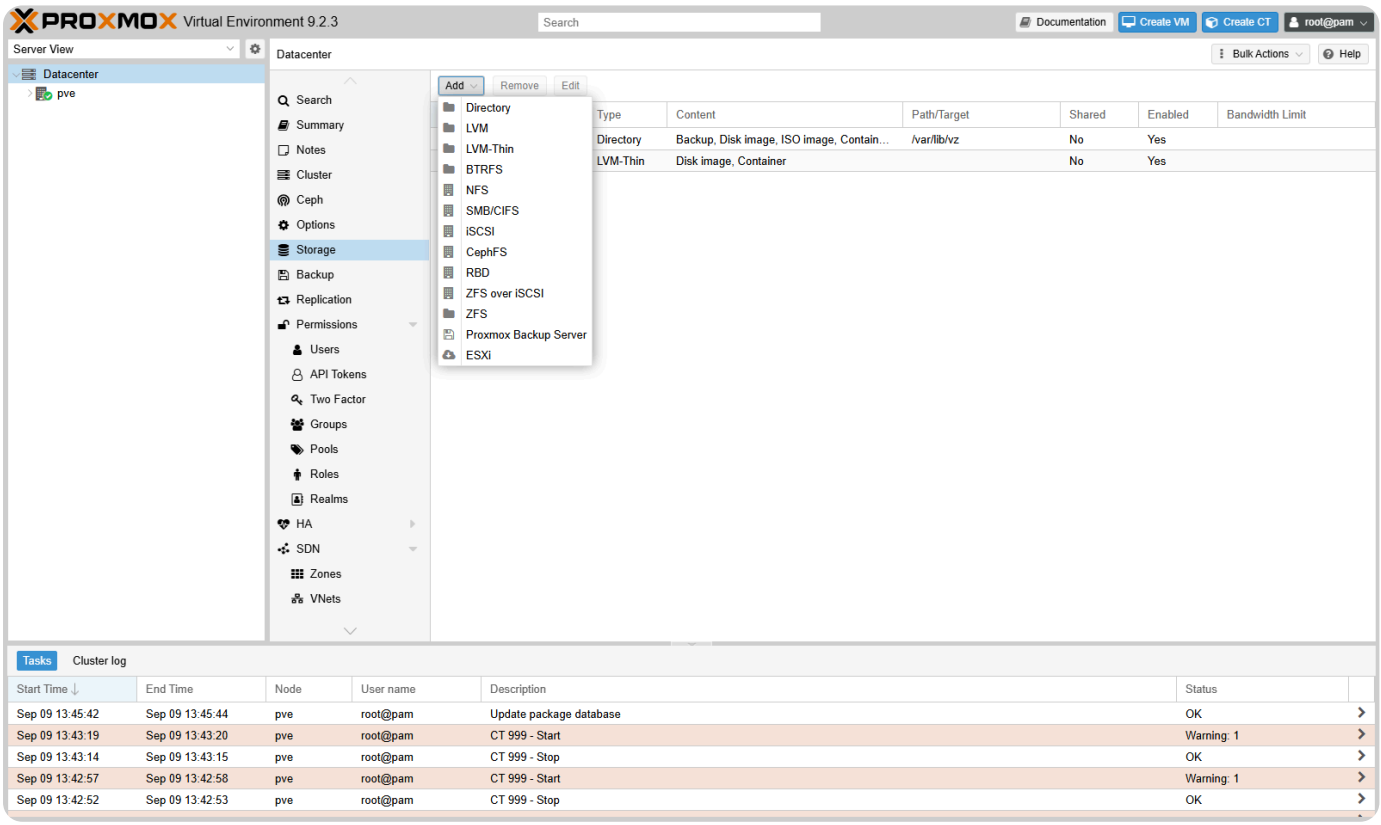
Ce support est la propriété de Dawan France.



Datacenter → Storage sur `pxe211` (Proxmox VE 9.2.21), votre écran à ce stade. C'est la vue graphique de `storage.cfg` : une ligne par stockage, avec son **type** (`Directory`, `LVM-Thin`), ses **contenus** (colonne *Content*), son chemin (*Path/Target*) et la colonne **Shared**.

`local` et `local-lvm` sont les deux stockages créés par l'installation. Ils sont propres au nœud : `Shared No`.

- un **type** : `dir`, `lvm`, `lvmthin`, `zfspool`, `nfs`, `cifs`, `iscsi`, `rbd`, `cephfs`, `pbs` … — il dit *comment* Proxmox accède aux données ;
- une liste de **contenus** acceptés : `images` (disques de VM), `rootdir` (disques de conteneurs), `iso`, `vztmpl` (templates LXC), `backup`, `snippets`, `import` ;
- un attribut **`shared`**, qui indique que le même contenu est visible depuis tous les nœuds (NFS, Ceph) — condition de la migration à chaud et de la haute disponibilité.



Ce support est la propriété de Dawan France.

Le même écran sous Proxmox VE 9.2, bouton **Add** ouvert : la liste des **types** de stockage que Proxmox sait déclarer. En tête, les stockages en fichier ou en bloc locaux : `Directory`, `LVM`, `LVM-Thin`, `BTRFS`.

Puis les stockages réseau : `NFS`, `SMB/CIFS`, `iSCSI`, `CephFS`, `RBD`, `ZFS over iSCSI`. Puis `ZFS`, et deux types particuliers : **Proxmox Backup Server**, l'entrepôt de sauvegardes, et **ESXi**, qui sert à importer des VM depuis VMware.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Les stockages déclarés et leur remplissage
pvesm status

# Leur définition, telle que Proxmox la stocke
cat /etc/pve/storage.cfg
```

Résultat attendu sur `pxe211` (Proxmox VE 9.2.21) :

```
Name             Type     Status     Total (KiB)      Used (KiB) Available (KiB)
%
local             dir     active        24527236         9669064        13695292
39.42%
local-lvm     lvmthin     active        64876544        42040000        22836543
64.80%

dir: local
    path /var/lib/vz
    content import,vztmpl,iso,backup

lvmthin: local-lvm
    thinpool data
    vgname pve
    content images,rootdir
```

Les deux stockages créés à l'installation illustrent parfaitement la distinction bloc/fichier, et c'est une confusion classique :

Ce support est la propriété de Dawan France.

Stockage	Volume sous-jacent	Type	Rôle
`local`	le volume `pve/root`, monté sur `/` ; les données vivent dans le répertoire `/var/lib/vz`	`dir` — **fichier**	ISO, templates, sauvegardes, snippets
`local-lvm`	le volume `pve/data`, un *thin pool* LVM	`lvmthin` — **bloc**	Disques de VM et de conteneurs

`local` est bien posé sur un volume LVM, mais Proxmox ne le voit que comme **un répertoire** : pas de thin provisioning, pas de snapshot au niveau du stockage. Seul `local-lvm` offre l'allocation à la demande, les snapshots par copie sur écriture et l'agrandissement à chaud.

Déclarer, modifier, retirer un stockage en ligne de commande. Les trois boutons **Add**, **Edit** et **Remove** de **Datacenter** → **Storage** ont leur équivalent `pvesm`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesm add dir essai-dir --path /var/tmp/essai-dir --content iso --create-base-path 1
# déclarer
pvesm set essai-dir --content iso,vztmpl
# modifier les contenus
grep -A3 "dir: essai-dir" /etc/pve/storage.cfg
# la déclaration écrite
pvesm remove essai-dir
# retirer la déclaration
ls -d /var/tmp/essai-dir
# le répertoire, lui, reste
```

Résultat attendu (nœud `pxe212`) :

```
dir: essai-dir
    path /var/tmp/essai-dir
    content vztmpl,iso
    create-base-path 1
/var/tmp/essai-dir
```

`pvesm remove` retire seulement la déclaration dans `storage.cfg`. Le dossier, et tout ce qu'il contient, reste sur le disque : `ls` le retrouve. Pour effacer les données, il faut une action séparée.

Dawan Nous formons votre savoir
Ce support est la propriété de Dawan France.

L'ancienne option `--mkdir 1` fonctionne encore, mais Proxmox affiche alors `NOTE: The 'mkdir' option … is deprecated` : on utilise `--create-base-path`.

3.5 LVM-thin : créer un pool sur des disques supplémentaires

Le chapitre 2 a détaillé le principe du *thin provisioning* sur le pool `local-lvm` installé par défaut : allocation à la demande, snapshots quasi instantanés, surveillance par `lvs`. Ce qui change ici : un nœud reçoit souvent des disques supplémentaires dédiés aux VM, séparés du disque système. Il faut alors construire un nouveau pool LVM-thin de toutes pièces.

La pile se lit de bas en haut, et c'est ce que font les trois commandes ci-dessous. Le disque devient un **volume physique** (PV : *Physical Volume*), rangé dans un **groupe de volumes** (VG : *Volume Group*).

Dans ce groupe, on crée un **thin pool**, un volume spécial qui distribue l'espace à la demande. Puis on le déclare à Proxmox comme stockage de type `lvmthin`.

> **À lire seulement : ne le faites pas sur votre nœud de lab.** La raison : chacun de vos disques supplémentaires a déjà un rôle — `sdc` et `sdd` recevront le miroir ZFS (§4), `sdb` est réservé à Ceph.
>
> Les commandes ci-dessous désignent un disque neuf, `sde`, que votre nœud n'a pas. Les lancer sur `sdb`, `sdc` ou `sdd` effacerait le pool ZFS ou bloquerait Ceph.

Via l'interface graphique :

Nœud → Disks → LVM-Thin → Create: Thinpool

Champ	Valeur
Name	`thin`
Disk	`/dev/sde` (un disque vide, dédié)

Add Storage ✓ (ajoute automatiquement le stockage dans Proxmox)

En ligne de commande :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Créer un nouveau groupe de volumes sur le disque dédié
# (vgcreate initialise lui-même le disque comme volume physique LVM)
```

Ce support est la propriété de Dawan France.

```
vgcreate ssd /dev/sde

# Assigner l'ensemble de l'espace disponible du VG au thinpool
lvcreate --type thin-pool -l 100%FREE -n thin ssd

# Ajouter le thinpool comme stockage connu de Proxmox
pvesm add lvmthin thin --vgname ssd --thinpool thin --content rootdir,images
```

Résultat attendu (testé sur PVE 9.2 avec un disque de test) :

```
  Physical volume "/dev/sde" successfully created.
  Volume group "ssd" successfully created
  Thin pool volume with chunk size 64.00 KiB can address at most <15.88 TiB of data.
  Logical volume "thin" created.
```

`vgcreate` a d'abord initialisé le disque comme volume physique, puis créé le groupe. La ligne `chunk size 64.00 KiB` de `lvcreate` n'est pas une erreur : elle donne la granularité d'allocation du pool.

L'espace est distribué aux disques de VM par tranches de 64 Kio, au fur et à mesure des écritures. L'entrée ajoutée à `/etc/pve/storage.cfg` :

```
lvmthin: thin
    thinpool thin
    vgname ssd
    content images,rootdir
```

Proxmox range lui-même la liste des contenus (`images,rootdir`, quel que soit l'ordre saisi). Sans l'option `--content`, aucune ligne `content` n'est écrite : le stockage accepte alors, par défaut, les disques de VM et de conteneurs.

Le pool se surveille ensuite comme `local-lvm`. Voici la lecture du pool installé par défaut, sur le groupe `pve` :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Espace réellement consommé (Data%) et métadonnées (Meta%) du thin pool, volume par volume
```

Ce support est la propriété de Dawan France.

```
lvs -o lv_name,lv_size,pool_lv,data_percent,metadata_percent pve
```

Résultat attendu (sur un nœud PVE 9.2) :

```
  LV               LSize    Pool Data%  Meta%
  data             <816.21g      11.01  0.59
  root               96.00g
  swap                8.00g
  vm-101-cloudinit    4.00m data 9.38
  vm-101-disk-0      20.00g data 10.34
```

La ligne `data` est le thin pool lui-même : `Data%` **est son remplissage physique réel**, toujours inférieur ou égal à la somme des tailles annoncées des disques qu'il héberge. Les lignes `vm-…` sont les disques des VM, rattachés au pool (colonne `Pool`) : `vm-101-disk-0` annonce 20 Go mais n'en occupe réellement que 10 %.

`root` et `swap`, sans pool, sont des volumes classiques — `root` porte le stockage `local`. Surveillez aussi `Meta%` : un thin pool dont les métadonnées sont pleines se bloque exactement comme s'il manquait de données.

> **Rappel du chapitre 2 :** Proxmox n'envoie aucune alerte sur le remplissage d'un pool thin. Alerte à 70 %, action avant 80 % : étendre le pool ou libérer de la place.

- 📖 Documentation Proxmox — Storage
- 📖 Le chapitre Storage du guide d'administration (types, contenus, fonctions par backend)
- 📖 LVM-thin dans Proxmox VE — pve-admin-guide
- 📖 Stockage fichier, bloc et objet — Red Hat

À retenir : trois modèles : le **bloc** (un disque brut, structuré par l'invité : LVM-thin, ZFS, iSCSI, Ceph RBD), le **fichier** (un dossier partagé : répertoire, NFS) et l'**objet**. Un stockage Proxmox est une entrée de `/etc/pve/storage.cfg` : un type, des contenus autorisés, et le fait d'être partagé ou non.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

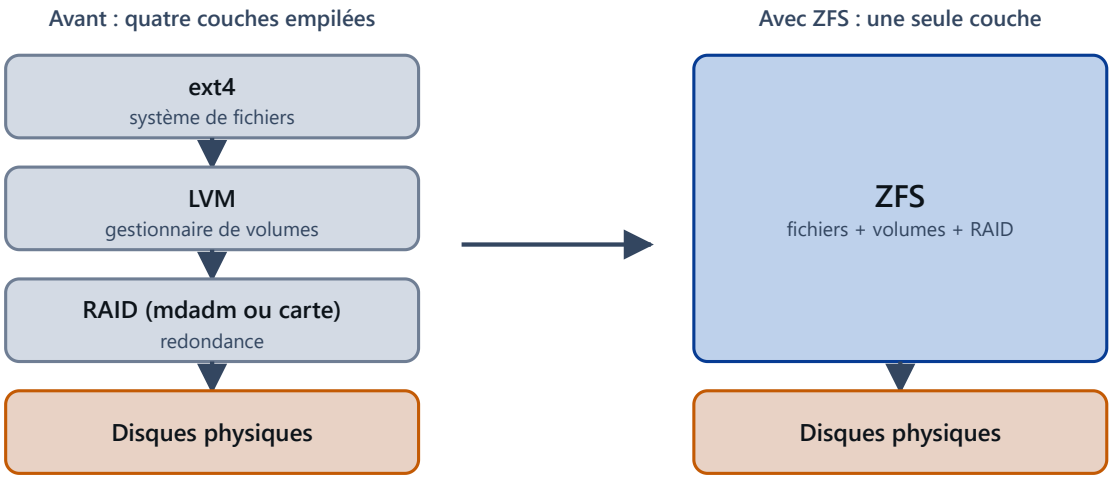
4. ZFS — le système de fichiers qui se souvient de tout

La grande majorité des administrateurs Linux s'appuie historiquement sur ext4, xfs ou LVM pour gérer leurs disques. Ces solutions fonctionnent bien, mais elles partagent une faiblesse commune : elles font confiance aux données qu'elles lisent.

Si un disque retourne silencieusement des données corrompues — le "bit rot" — le système de fichiers ne s'en aperçoit pas. **ZFS** a été conçu précisément pour résoudre ce problème.

ZFS combine en un seul outil ce que LVM, mdadm et ext4 font séparément : la gestion des volumes, le RAID logiciel, les snapshots et le système de fichiers lui-même. Pas besoin de contrôleur RAID matériel coûteux : ZFS s'en charge en logiciel. Des checksums sur chaque bloc de données détectent toute corruption silencieuse ; en mode miroir ou RAIDZ, ZFS la corrige.

Ce que ZFS remplace : une pile de quatre couches



ZFS ne fait jamais confiance au matériel : chaque bloc lu est vérifié par checksum.

Pas de ZFS ni de Ceph sur des SSD grand public — c'est l'avertissement que répètent les contributeurs de Proxmox. ZFS et Ceph demandent au disque de garantir chaque écriture synchrone (`sync`). Un SSD grand public, sans protection contre la coupure de courant, ne peut le faire qu'en vidant son cache à chaque fois.

Ce support est la propriété de Dawan France.

> Les performances s'effondrent, et l'usure s'accélère au point de consommer la durée de vie du disque en quelques mois. Les SSD d'entreprise à condensateurs (*PLP*, *power loss protection*) acquittent ces écritures depuis leur cache protégé, sans ce coût.
>
> Sans ce budget, une installation simple — ext4 et de bonnes sauvegardes — vaut mieux qu'un ZFS sur du matériel inadapté. En atelier, avec des disques virtuels, la question ne se pose pas ; elle se pose avant tout déploiement en production.

4.1 Vocabulaire ZFS et niveaux RAID

ZFS introduit son propre vocabulaire qu'il faut maîtriser avant de taper la première commande. Un **vdev** (virtual device) regroupe un ou plusieurs disques physiques avec une stratégie de redondance donnée.

Un **zpool** est constitué d'un ou plusieurs vdevs — c'est l'unité de stockage visible de l'extérieur. Si vous perdez un vdev entier, vous perdez tout le zpool : la résilience ne se fait jamais d'un vdev à l'autre, uniquement à l'intérieur d'un même vdev.

Un zpool peut ensuite héberger des **datasets** (systèmes de fichiers virtuels) et des **zvols** (volumes logiques en mode bloc). La taille d'un dataset peut être limitée par quota.

Niveau	Équivalent RAID classique	Tolérance de panne	Disques minimum
stripe	RAID 0	Aucune	1
mirror	RAID 1	1 disque par vdev	2
RAIDZ1	RAID 5	1 disque	3
RAIDZ2	RAID 6	2 disques	4
RAIDZ3	—	3 disques	5

En RAIDZ, la capacité utile est celle de N − 1 disques (RAIDZ1), N − 2 (RAIDZ2) ou N − 3 (RAIDZ3). Ainsi, 3 disques de 10 Go en RAIDZ1 donnent environ 20 Go utiles.

RAIDZ face au RAID 5 classique. Le RAID 5 connaît le *write hole* : une coupure en pleine écriture peut laisser données et parité incohérentes. RAIDZ écrit de façon atomique, grâce au copy-on-write (§4.2), et sa largeur de bande s'adapte à chaque écriture.

En cas de panne, il ne reconstruit que les blocs réellement utilisés, là où le RAID 5 reconstruit tout le disque. Et chaque bloc porte une somme de contrôle.

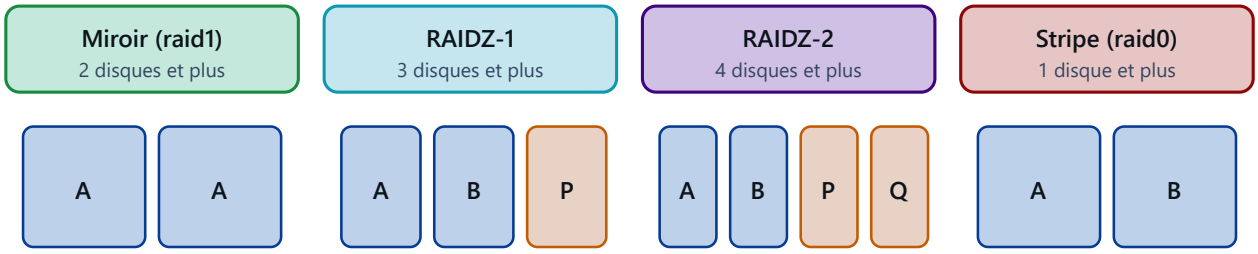
Un pool accepte aussi des vdev **spécialisés**, en plus des vdev de données :

Dawan
Ce support est la propriété de Dawan France.

vdev	Rôle
`spare`	Disque de rechange, prêt à remplacer un disque défaillant du pool
`cache`	**L2ARC** : un SSD qui prolonge le cache de lecture tenu en mémoire
`log`	**SLOG** : un SSD rapide qui reçoit le journal des écritures synchrones
`special`	Les métadonnées et les petits blocs, sur un support rapide

Dans le contexte de l'atelier, les VMs imbriquées disposent de disques supplémentaires (`/dev/sdc` , `/dev/sdd`). Un pool miroir sur ces deux disques est l'exercice de référence pour s'entraîner.

Les niveaux de redondance ZFS



En production pour la virtualisation, le miroir est recommandé : meilleures IOPS aléatoires.

A, B = blocs de données · P, Q = parité · la redondance est purement logicielle, sans contrôleur RAID

4.2 Le Copy-on-Write : pourquoi ZFS ne corrompt jamais un fichier en cours d'écriture

Le **Copy-on-Write** (CoW) est le mécanisme qui rend les snapshots ZFS quasi-instantanés et garantit l'intégrité des données à tout moment. Avec le CoW, toute modification d'un fichier écrit les données modifiées à côté du fichier original, jamais par-dessus.

Imaginons un fichier constitué de blocs A, B, C et D. La plupart des systèmes de fichiers classiques écriraient les nouvelles données par-dessus le bloc C, sur place.

Avec le Copy-on-Write, ZFS copie les nouvelles données dans une nouvelle zone. Une fois la copie terminée, il met à jour les métadonnées du fichier pour pointer vers cette nouvelle zone. Le bloc C original est libéré ultérieurement par un processus en tâche de fond.

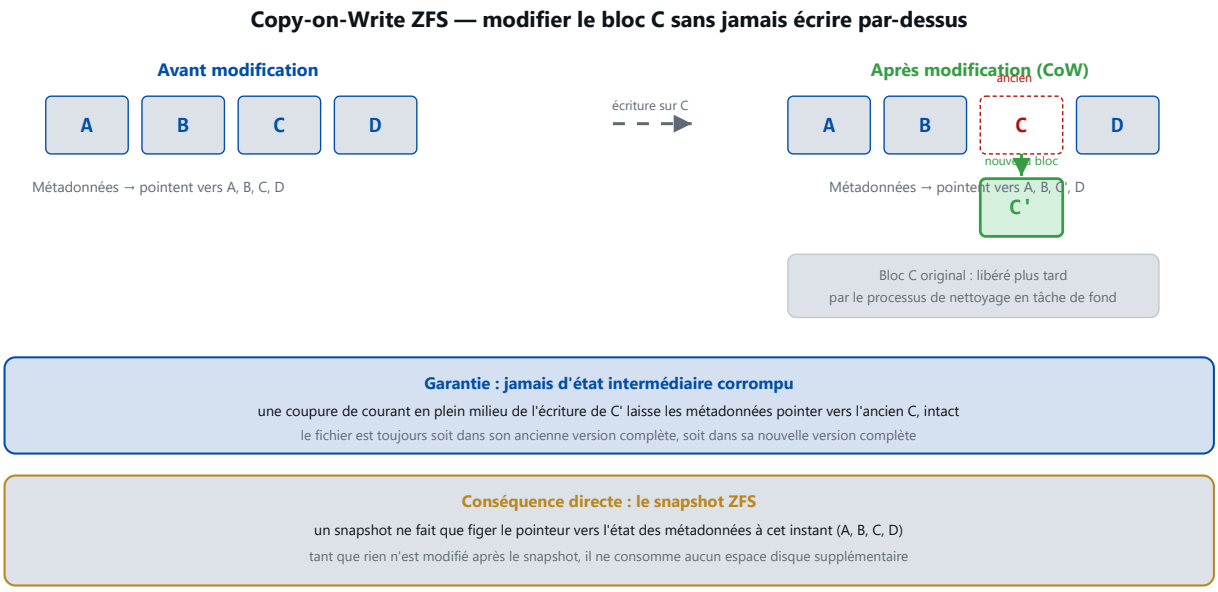
Il ne peut donc jamais exister d'état où les données sont à moitié écrites dans le fichier. Celui-ci est toujours soit dans son ancienne version complète, soit dans la nouvelle version complète. Une

Dawan
Ce support est la propriété de Dawan France.

coupure de courant en plein milieu d'une écriture ne corrompt jamais un fichier ZFS — au pire, l'écriture n'a simplement pas eu lieu.

Un arbre de sommes de contrôle. ZFS range ses blocs dans un **arbre de Merkle** : chaque bloc contient la somme de contrôle des blocs qu'il désigne. Au sommet, l'**uberblock** décrit l'état complet du pool.

Une écriture se termine en remplaçant cet uberblock d'un coup. C'est pourquoi un pool ZFS n'a jamais besoin de `fsck` après une coupure : il repart du dernier uberblock valide.



> **Conséquence directe pour les snapshots :** un snapshot ZFS ne fait que figer un pointeur vers l'état des métadonnées à cet instant. Tant que rien n'est modifié, le snapshot ne consomme aucun espace disque supplémentaire.

Revenons à la définition de base : un **système de fichiers** range des données sur des disques et tient le registre de ce qui est où. ZFS fait en plus le travail du RAID et des volumes. Son vocabulaire, du disque au bloc :

Terme	Ce que c'est	Dans le laboratoire
Disque	Un disque physique	`sdc` et `sdd`
vdev	Un groupe de disques avec sa redondance (miroir, RAIDZ)	`mirror-0`
zpool	L'espace de stockage, fait d'un ou plusieurs vdev ; si un vdev entier tombe, tout le pool est	`datapool`

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
	perdu	
Dataset	Un système de fichiers dans le pool, avec ses propriétés (quota, compression)	`datapool/vms`
zvol	Un volume en mode bloc, pour le disque d'une VM	Les disques des VM sur le stockage ZFS (§4.5)
Copy-on-write	Une donnée modifiée s'écrit ailleurs ; le pointeur bascule une fois l'écriture finie	Une coupure ne laisse jamais un fichier à moitié écrit
Snapshot	Un état figé, instantané, qui ne coûte que les blocs modifiés ensuite	`datapool/vms@avant-migration` (§4.4)
Checksum	Une somme de contrôle par bloc, vérifiée à chaque lecture et par le *scrub*	`zpool status` : colonne `CKSUM`

Avec le copy-on-write, les pointeurs désignent toujours soit l'ancien bloc, soit le nouveau, jamais un bloc à moitié écrit. C'est pourquoi les snapshots sont instantanés, et une coupure de courant sans danger pour la cohérence.

Où voir ZFS sur votre nœud :

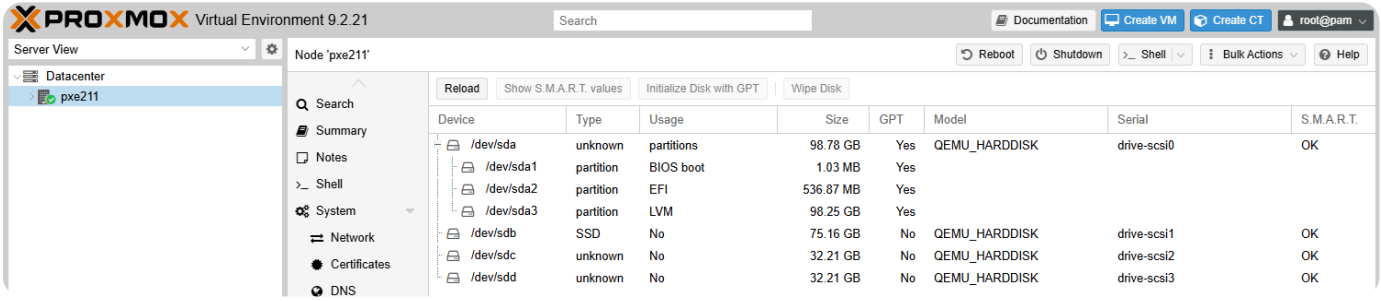
Notion	Où la voir
Le pool, ses vdev et ses disques	**Nœud** → **Disks** → **ZFS** ; `zpool status` (état `ONLINE`, `mirror-0`, `sdc`, `sdd`)
Les datasets et l'espace	`zfs list`
Les snapshots	`zfs list -t snapshot`
Le stockage Proxmox posé dessus	**Datacenter** → **Storage** : type `ZFS` ; `/etc/pve/storage.cfg` : `zfspool`

Ce support est la propriété de Dawan France.

4. ZFS — le système de fichiers qui se souvient de tout — Créer et gérer un pool ZFS

4.3 Créer et gérer un pool ZFS

La gestion ZFS repose sur deux commandes distinctes : `zpool` pour le pool (les disques, la redondance, la santé) et `zfs` pour les datasets (les volumes logiques, leurs propriétés, les snapshots).



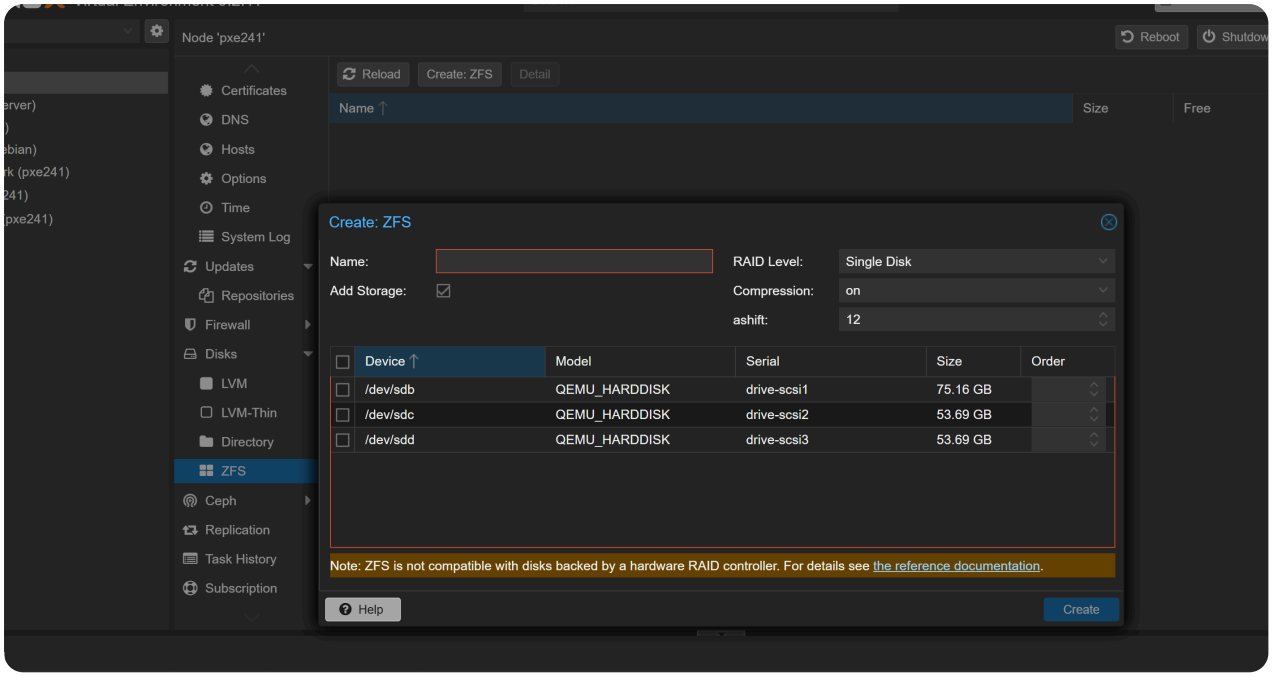
Avant de créer un pool, on identifie les disques. Nœud → Disks liste chaque disque avec son **usage** actuel. Sur `pxe211` (Proxmox VE 9.2.21), avant le pool ZFS.

`/dev/sda` porte le système (partitions `BIOS boot`, `EFI`, `LVM`). `/dev/sdb`, `/dev/sdc` et `/dev/sdd` affichent `No` dans la colonne *Usage* : ils sont libres. `sdc` et `sdd` font la même taille, 32,21 Go : ce sont eux qui recevront le miroir ZFS.

L'interface ne propose un disque à ZFS que si sa colonne *Usage* indique `No` ; sinon, il faut d'abord l'effacer (**Wipe Disk**), après avoir vérifié trois fois son nom. Dans le lab, les disques de ZFS sont `sdc` et `sdd` ; `sdb` est réservé à Ceph.

Depuis l'interface graphique : `Nœud → Disks → ZFS → Create: ZFS`

Ce support est la propriété de Dawan France.



Trois points à relever dans cette fenêtre.

- **La liste des disques.** `sdc` et `sdd` font exactement la même taille : 32,21 Go sur le lab d'exemple, `pxe211`. `sdb` en fait 75,16 (les tailles peuvent différer sur votre nœud).

  C'est la raison technique du plan de disques : **dans un miroir, la capacité du pool est celle du plus petit disque**. Associer `sdb` à `sdc` donnerait 32,21 Go utiles et gaspillerait près de 43 Go. `sdb` reste par ailleurs réservé à Ceph.

- **Le champ RAID Level** affiche `Single Disk` par défaut : aucune redondance. C'est celui qu'il faut basculer sur `Mirror`.

- **La case Add Storage**, cochée, déclare automatiquement le pool comme stockage Proxmox. Sans elle, le pool existerait côté système sans apparaître dans l'interface, et il faudrait l'ajouter avec `pvesm add`.

L'avertissement orange rappelle une règle absolue : **ZFS ne se pose jamais sur un contrôleur RAID matériel**. Il a besoin d'un accès direct aux disques pour vérifier les sommes de contrôle et réparer les blocs — un contrôleur RAID s'interposerait et lui masquerait l'état réel du support.

Champ	Valeur
Name	`datapool`
RAID Level	`Mirror`
Disk 1	`/dev/sdc`
Disk 2	`/dev/sdd`

Ce support est la propriété de Dawan France.

En CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Créer un pool mirror nommé "datapool" sur deux disques
# (l'interface ajoute d'office -o ashift=12 : blocs de 4 Kio, adaptés aux disques actuels)
zpool create -o ashift=12 datapool mirror /dev/sdc /dev/sdd

# Vérifier l'état et la santé du pool
zpool status

# Créer un dataset (sous-volume avec ses propres propriétés)
zfs create datapool/vms

# Activer la compression transparente lz4 (rapide, bon ratio)
zfs set compression=lz4 datapool

# Vérifier la liste des datasets et leur occupation
zfs list
```

Résultat attendu de `zpool status` :

```
  pool: datapool
 state: ONLINE
config:
        NAME        STATE     READ WRITE CKSUM
        datapool    ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            sdc     ONLINE       0     0     0
            sdd     ONLINE       0     0     0

errors: No known data errors
```

Les trois colonnes de droite sont les compteurs qu'on surveille. `READ` et `WRITE` comptent les erreurs d'entrée-sortie remontées par le disque.

`CKSUM` compte les blocs dont la somme de contrôle ne correspondait pas : des corruptions silencieuses, que le miroir a réparées. Tant qu'ils restent à zéro et que chaque ligne est `ONLINE`, le

Ce support est la propriété de Dawan France.

pool est sain.

> **Erreur fréquente :** ne jamais utiliser un disque qui contient déjà des partitions ou des données sans l'avoir effacé avec `wipefs -a /dev/sdX`. ZFS refusera de créer le pool si le disque contient une signature de système de fichiers reconnu.

Les propriétés qui comptent

Une propriété se pose sur un dataset avec `zfs set`, se lit avec `zfs get`, et s'hérite par les datasets enfants.

Propriété	Exemple	Effet
Compression	`zfs set compression=lz4 datapool/vms`	`lz4` : usage général, très rapide. `zstd` : meilleur taux, CPU modéré. `gzip` : archivage, CPU élevé
Quota	`zfs set quota=50G datapool/utilisateurs`	Plafond : le dataset ne dépassera pas 50 Go
Réservation	`zfs set reservation=10G datapool/service`	Garantie : 10 Go restent toujours disponibles pour ce dataset
Déduplication	`dedup=on`	Ne stocke qu'une fois les blocs identiques. Très gourmande en RAM (sa table, la DDT — *DeDuplication Table* —, doit tenir en mémoire) : déconseillée en production, sauf cas très particulier

Le provisionnement fin est natif. Un zvol déclaré « fin » (`-s`, pour *sparse*) n'occupe que les blocs réellement écrits. Sans `-s`, ZFS lui réserve d'emblée toute sa taille, par la propriété `refreservation` : c'est le provisionnement épais.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
zfs create -V 200M essai/zvol-epais        # épais : taille réservée d'emblée
zfs create -s -V 200M essai/zvol-fin       # fin : rien n'est réservé
zfs get -o name,property,value refreservation,volsize essai/zvol-epais essai/zvol-
fin   # réservation et taille
zfs list -o name,used,refer essai          # l'espace réellement compté
```

Résultat attendu (nœud `pxe212`, sur un pool d'essai `essai` monté sur des fichiers, supprimé ensuite) :

Ce support est la propriété de Dawan France.

```
NAME              PROPERTY        VALUE
essai/zvol-epais  refreservation  205M
essai/zvol-epais  volsize         200M
essai/zvol-fin    refreservation  none
essai/zvol-fin    volsize         200M
NAME    USED  REFER
essai   205M  30.6K
```

Le zvol épais réserve 205 Mo (ses 200 Mo, plus les métadonnées), comptés dans `USED` alors que rien n'est écrit. Le zvol fin ne réserve rien : `refreservation none`.

> **Le provisionnement fin permet de promettre plus d'espace qu'on n'en a.** Si toutes les VM écrivent en même temps jusqu'à leur taille, le pool sature : erreurs d'écriture, VM en erreur. On surveille donc l'occupation réelle (`zpool list` , colonne `CAP`) et on pose une alerte vers 75 à 80 %.

4.4 Snapshots ZFS

ZFS intègre les snapshots directement au niveau du système de fichiers, sans passer par LVM comme sur un système classique. Cette intégration change tout, pour une raison technique : ZFS est **copy-on-write**.

Concrètement, ZFS n'écrase jamais un bloc en place — il écrit la modification ailleurs et met à jour ses pointeurs. Un snapshot n'a donc rien à copier : il se contente de **figer les pointeurs actuels**.

C'est pourquoi il est **instantané et sans coût initial**, quelle que soit la taille du dataset. Un snapshot d'un pool de 2 To se prend en une fraction de seconde et n'occupe zéro octet à l'instant où on le crée.

L'espace ne se consomme qu'**ensuite**, au fur et à mesure que les données divergent. Chaque bloc modifié après le snapshot est conservé en double : l'ancien pour le snapshot, le nouveau pour l'état courant. Un snapshot sur un système qui écrit beaucoup grossit donc vite — d'où l'intérêt de ne pas les accumuler.

Avec plusieurs snapshots, l'espace se partage. Un bloc ancien reste tant qu'**un** snapshot au moins le désigne. Supprimer le plus ancien ne libère donc pas forcément de place, si le suivant référence les mêmes blocs.

Dawan Ce support est la propriété de Dawan France.

Pour savoir qui occupe quoi : `zfs list -t all -o name,used,refer,written,clones`. `USED` est ce que la suppression libérerait ; `WRITTEN`, ce qui a été écrit depuis le snapshot précédent ; `CLONES`, les clones qui en dépendent.

Deux points de vigilance qui distinguent ZFS de l'instantané LVM :

`zfs rollback` **est destructif et unidirectionnel.** Revenir à un snapshot détruit **tout ce qui a été écrit depuis**, sans confirmation au-delà de l'option. Et il faut revenir dans l'ordre : ZFS refuse de sauter par-dessus un snapshot plus récent sans le `-r` qui le détruit au passage.

Un snapshot n'est pas une sauvegarde tant qu'il reste dans le pool. Il partage le sort du pool : si le disque meurt, le snapshot meurt avec. C'est `zfs send` qui le transforme en sauvegarde réelle, en l'expédiant vers un autre pool ou une autre machine — la contrepartie ZFS de l'envoi vers PBS.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Créer un snapshot (syntaxe : dataset@nom-du-snapshot)
zfs snapshot datapool/vms@avant-migration

# Lister tous les snapshots
zfs list -t snapshot

# Revenir à l'état du snapshot (le dataset actuel est écrasé)
zfs rollback datapool/vms@avant-migration

# Supprimer un snapshot devenu inutile
zfs destroy datapool/vms@avant-migration
```

Résultat attendu de `zfs list -t snapshot` :

```
NAME                              USED  AVAIL     REFER  MOUNTPOINT
datapool/vms@avant-migration        0B      -     24.5K  -
```

La colonne `USED` à `0B` signifie qu'aucun bloc n'a encore changé depuis le snapshot — c'est la conséquence directe du Copy-on-Write vu plus haut.

4.5 Intégrer ZFS dans Proxmox VE

Ce support est la propriété de Dawan France.

Une fois le pool créé, il faut l'enregistrer auprès de Proxmox pour qu'il apparaisse comme source de stockage dans l'interface.

Via l'interface graphique : `Datacenter → Storage → Add → ZFS`

Champ	Valeur
ID	`zfs-vms`
ZFS Pool	`datapool/vms`
Content	`Disk image`, `Container`

En CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Déclarer le dataset datapool/vms comme stockage Proxmox
# --content images,rootdir : autorise les disques de VMs et les rootfs LXC
pvesm add zfspool zfs-vms --pool datapool/vms --content images,rootdir

# Vérifier que le stockage est bien enregistré
pvesm status
```

Résultat attendu (nœud `pxe211`, disques de 30 Go) :

```
Name             Type     Status     Total (KiB)      Used (KiB) Available (KiB)
%
zfs-vms       zfspool     active        29965928              96        29965832
0.00%
```

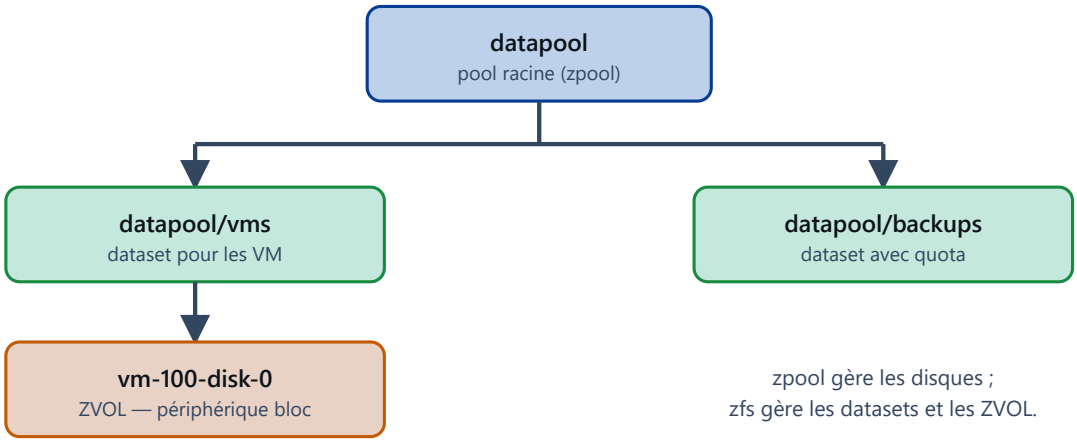
Si vous avez laissé **Add Storage** coché à la création du pool (§4.3), un stockage `datapool` existe déjà et pointe sur la racine du pool. Celui-ci, `zfs-vms`, pointe sur le dataset `vms` : on peut ainsi lui donner ses propres propriétés (quota, compression).

Depuis ce moment, `zfs-vms` est sélectionnable lors de la création d'une VM ou d'un conteneur. Le disque sera stocké directement dans le pool ZFS avec tous les avantages qui en découlent : compression automatique, snapshots intégrés à l'interface Proxmox, vérification d'intégrité continue.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

> **Bonnes pratiques à retenir :** ne jamais mélanger ZFS avec LVM sur les mêmes disques. ZFS consomme de la RAM : la documentation de Proxmox demande « au moins 8 Go pour commencer », et davantage dès que le budget le permet (§4.6).
>
> Et ne placez jamais ZFS derrière un contrôleur RAID dont le cache d'écriture n'est pas protégé par batterie. ZFS croirait ses écritures en sécurité, alors qu'elles dorment dans une mémoire que la moindre coupure efface.

Pool, datasets et ZVOL



Déplacer un disque existant vers ZFS. Une VM déjà installée sur `local-lvm` peut passer sur le stockage ZFS sans être arrêtée.

Dans l'interface : VM → **Hardware** → le disque → **Disk Action** → **Move Storage**, cible `zfs-vms`, case **Delete source** cochée. **En ligne de commande :** `qm disk move <vmid> scsi0 zfs-vms --delete 1`.

QEMU copie le disque pendant que la VM tourne, en répercutant ses écritures sur la copie, puis bascule sur la copie. Sans **Delete source**, l'ancien volume reste attaché à la VM en `unused0` et continue d'occuper `local-lvm`. Le format suit le stockage cible : ZFS n'accepte que `raw`.

4.6 L'ARC — le cache mémoire qui accélère ZFS

ZFS consomme de la RAM : c'est son **ARC** (*Adaptive Replacement Cache*), un cache de lecture tenu en mémoire vive, qui garde les blocs récemment ou fréquemment lus. Il équilibre seul les deux, et s'adapte aux habitudes de lecture.

Ce support est la propriété de Dawan France.

Sur un nœud qui héberge aussi des VM, l'ARC entre en concurrence avec elles pour la mémoire. Depuis Proxmox VE 8.1, l'installateur pose donc une limite prudente : **10 % de la RAM, 16 Gio au plus**, écrite dans `/etc/modprobe.d/zfs.conf` (`options zfs zfs_arc_max=…`).

Un nœud installé avant la 8.1 n'a pas cette limite. `arc_summary` affiche le plafond (`Max target size`) et la mémoire réellement occupée (`Current size`).

- 📖 Documentation Proxmox — ZFS on Linux
- 📖 Documentation OpenZFS

Ce support est la propriété de Dawan France.

4. ZFS — le système de fichiers qui se souvient de tout — Le snapshot, délégué au système de fichiers

4.7 Le snapshot, délégué au système de fichiers

En clair. Dans l'interface, prendre un instantané est toujours le même geste : onglet **Snapshots** de la machine, puis **Take Snapshot**. Que le disque soit sur `local-lvm` ou sur ZFS, l'écran est identique.

Mais Proxmox ne fait pas le snapshot lui-même : il le **délègue** au stockage qui porte le disque. C'est donc le stockage qui décide si le snapshot est possible, et comment il est fait.

Stockage du disque	Qui fait réellement le snapshot	Où le voir, avec l'outil du stockage
LVM-thin (`local-lvm`)	LVM crée un volume léger, qui partage les blocs du disque	`lvs` : un volume `snap_vm-<id>-disk-0_<nom>`
ZFS (`zfs-vms`)	ZFS fige ses pointeurs, sans rien copier (copy-on-write, §4.2)	`zfs list -t snapshot` : `…/vm-<id>-disk-0@<nom>`
Répertoire, disque `raw`	Personne : un répertoire n'a pas de snapshot au niveau du stockage (§3.4)	Nulle part : le snapshot est refusé (`snapshot feature is not available`)

Sur le lab, le snapshot d'une VM posée sur `local-lvm` (une VM 101 de test) s'est traduit par une ligne de LVM, et non de Proxmox. La voici : `Logical volume "snap_vm-101-disk-0_avant-mise-a-jour" created.` C'est la preuve de la délégation.

Le même geste, sur une VM de test qui porte un disque sur ZFS et un disque sur `local-lvm`, montre les deux outils au travail.

Dans le Shell du nœud :

```
qm snapshot 190 essai   # snapshot de la VM 190, ses deux disques
zfs list -t snapshot    # côté ZFS : le snapshot du dataset
```

Ce support est la propriété de Dawan France.

```
lvs | grep 190          # côté LVM : le volume du snapshot
```

Résultat attendu (nœud `pxe212`, Proxmox VE 9.2.21 ; VM 190 de test, supprimée ensuite) :

```
snapshotting 'drive-scsi0' (zfs-vms:vm-190-disk-0)
snapshotting 'drive-scsi1' (local-lvm:vm-190-disk-0)
  WARNING: You have not turned on protection against thin pools running out of
space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger
automatic extension of thin pools before they get full.
  Logical volume "snap_vm-190-disk-0_essai" created.
  WARNING: Sum of all thin volume sizes (80.00 GiB) exceeds the size of thin pool
pve/data (no free space in volume group).

NAME                                USED  AVAIL  REFER  MOUNTPOINT
datapool/vms/vm-190-disk-0@essai     0B      -    56K  -

  snap_vm-190-disk-0_essai       pve Vri---tz-k  1.00g data vm-190-disk-0
  vm-190-disk-0                  pve Vwi-a-tz--  1.00g data
0.00
```

Un seul snapshot Proxmox, deux mécanismes. ZFS a créé `…@essai` (0 octet : rien n'est copié). LVM a créé le volume léger `snap_vm-190-disk-0_essai`.

Les `WARNING` viennent de LVM : la somme des tailles des disques fins (80 Gio) dépasse la taille du pool `pve/data`, et aucune extension automatique n'est réglée. Le snapshot est créé malgré tout.

Il suffit qu'**un seul** disque soit sur un stockage sans snapshot pour que toute la machine soit bloquée. Vérifié en ajoutant à la même VM un disque `raw` sur un stockage de type répertoire.

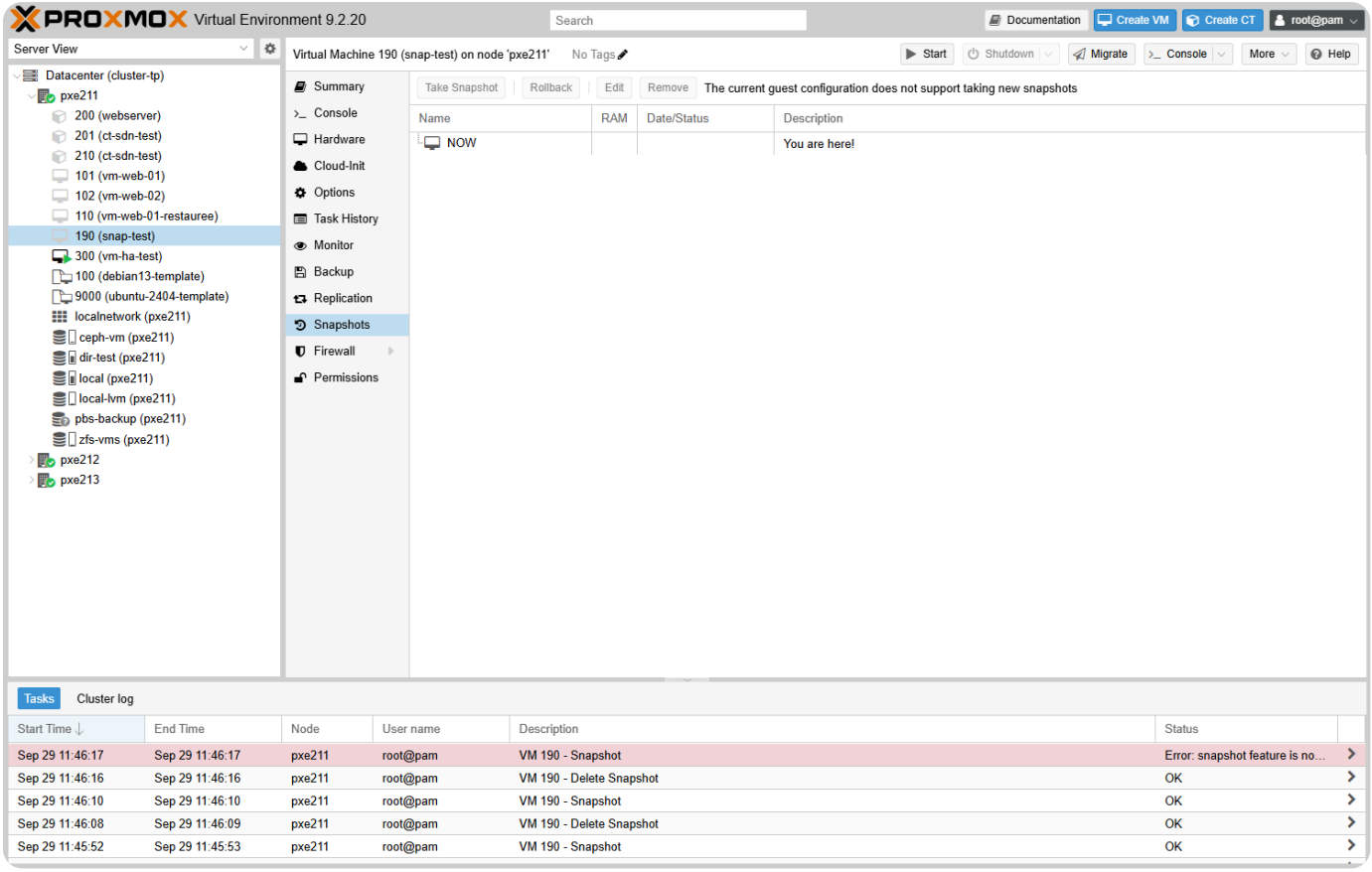
Dans le Shell du nœud :

```
qm snapshot 190 essai-dir   # même geste, avec un disque sur un répertoire
```

Résultat attendu :

```
snapshot feature is not available
```

Ce support est la propriété de Dawan France.



L'onglet **Snapshots** de la VM de test. Le bouton **Take Snapshot** est grisé, et le message à sa droite en donne la raison : *The current guest configuration does not support taking new snapshots*.

L'interface a interrogé le stockage de chaque disque avant d'afficher le bouton. En bas, le journal des tâches garde la trace de la tentative en ligne de commande : *Error: snapshot feature is not available*.

Deux conséquences pratiques :

- **Le choix du stockage décide de ce que vous pourrez faire de vos machines.** Un disque posé sur un stockage sans snapshot ne pourra jamais être figé avant une mise à jour risquée. C'est la colonne *Snapshots* du comparatif du §6.1.
- **Un snapshot vit sur le même stockage que le disque.** Il partage son sort : si le pool ou le groupe de volumes meurt, le snapshot meurt avec. Ce n'est pas une sauvegarde.

À retenir : ZFS réunit volumes, RAID logiciel, snapshots et système de fichiers. Proxmox **délègue** le snapshot au stockage : même bouton, mécanisme propre à chaque stockage.

Le **copy-on-write** protège les données et rend les snapshots instantanés ; les **checksums** détectent la corruption.

Dawan
Ce support est la propriété de Dawan France.

L'**ARC** met en cache en mémoire vive. On vérifie et on ajuste son plafond (`zfs_arc_max`, posé à 10 % de la RAM par l'installateur depuis la version 8.1). Sinon, ZFS peut prendre plus de la moitié de la RAM du nœud.

> **▶ À pratiquer maintenant — Atelier 3 puis Atelier 4**
>
> Créer un pool miroir sur deux disques, y poser un dataset compressé, puis éprouver le snapshot et le rollback — casser un fichier volontairement pour constater qu'il revient.
>
> Enfin, faire passer le disque d'une VM de `local-lvm` à `zfs-vms` sans l'arrêter.

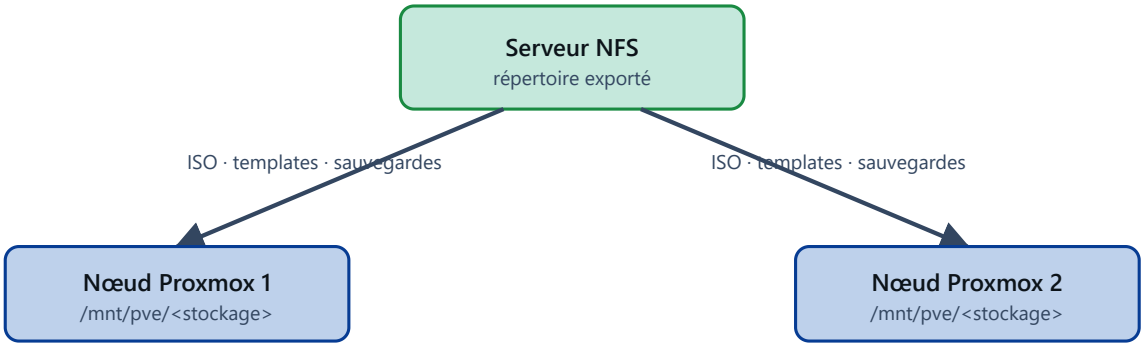
Ce support est la propriété de Dawan France.

5. Le stockage réseau : NFS et iSCSI

ZFS et LVM-thin couvrent le stockage local d'un nœud. Dès que plusieurs nœuds doivent accéder aux mêmes données, ou que vous voulez centraliser ISOs et sauvegardes, il faut du stockage réseau. Deux protocoles classiques structurent ce sujet.

NFS (Network File System) expose un répertoire d'un serveur comme un dossier réseau. C'est la solution la plus simple pour partager des ISOs, des templates ou des sauvegardes entre nœuds. NFS fonctionne bien pour du stockage de fichiers, moins bien pour des disques de VMs en production à cause de la latence et des problèmes de verrouillage en accès concurrent.

NFS : le partage de fichiers

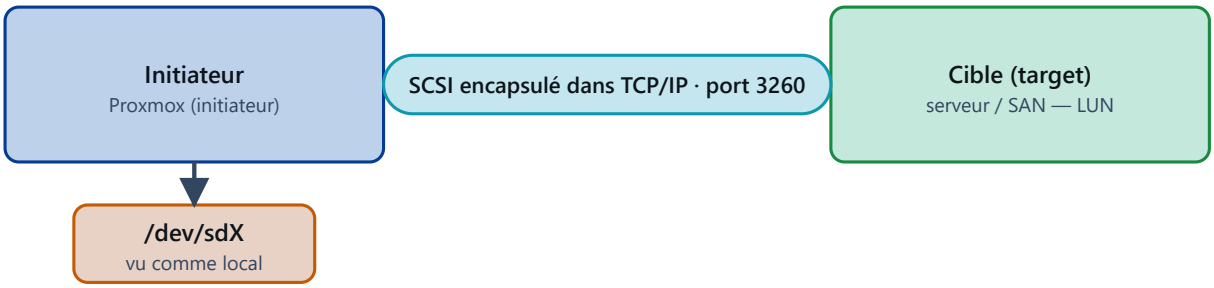


Simple et partagé — mais sensible à la latence : à éviter pour les disques de VM en production.

iSCSI expose des blocs de disque (LUNs) à travers le réseau IP, comme si c'était un disque local. C'est plus performant que NFS pour les disques de VMs car il n'y a aucune couche de système de fichiers entre Proxmox et le stockage.

Ce support est la propriété de Dawan France.

iSCSI : le stockage en mode bloc



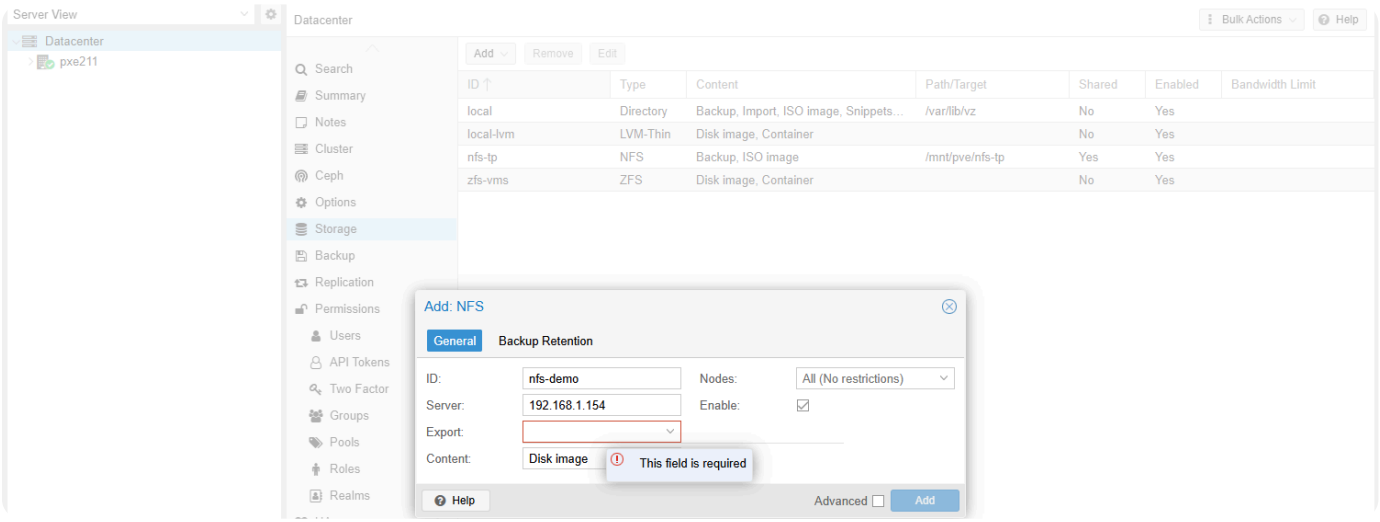
Performant pour les disques de VM · nécessite un réseau dédié

Le serveur NFS ou la baie iSCSI sont fournis par l'infrastructure de stockage de l'entreprise : on ne les construit pas sur Proxmox. Côté Proxmox, le travail consiste à **déclarer** ces stockages.

5.1 Ajouter un partage NFS à Proxmox

Dans l'interface :

Datacenter → Storage → Add → NFS



(nœud `pxe211` du lab d'exemple, devant un serveur NFS de test). Quatre points méritent l'attention.

- **Le champ Export est une liste déroulante, pas un champ libre.** Dès l'adresse du serveur saisie, Proxmox l'interroge et affiche ce qu'il partage réellement.

Ce support est la propriété de Dawan France.

Si la liste reste vide, inutile de remplir le reste : soit le serveur ne répond pas, soit il n'exporte pas vers ce nœud. C'est le test de connectivité le plus rapide dont vous disposiez.

- **Le champ Content** décide de ce que le stockage acceptera : sauvegardes, ISO, disques de machines, modèles de conteneurs. Un contenu non coché ne pourra pas y être écrit.
- **Nodes : All (No restrictions)** déclare le stockage pour tous les nœuds du cluster. Chacun devra pouvoir joindre le serveur, sans quoi le stockage remontera en erreur sur ceux qui n'y arrivent pas.
- **La colonne Shared, dans la liste derrière la fenêtre.** `local` et `local-lvm` affichent `No`.

  Le partage NFS affichera `Yes` : la distinction entre stockage local et stockage partagé, rendue visible en une colonne. C'est elle qui conditionne la migration à chaud et la haute disponibilité.

Laissez **NFS Version** sur `Default` : client et serveur négocient seuls. Ne cochez pas *Allow Snapshots as Volume-Chain*, signalé comme fonction en avant-première.

Champ	Valeur
ID	`nfs-partage`
Server	l'adresse du serveur NFS
Export	`/srv/partage` *(saisir ou sélectionner dans le menu déroulant)*
Content	`ISO Image`, `Container template`, `Disk image`

En CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Côté Proxmox — ajouter le stockage NFS
pvesm add nfs nfs-partage \
  --server <adresse-du-serveur-nfs> \
  --export /srv/partage \
  --content iso,vztmpl,images

# Vérifier l'ajout
pvesm status
```

Résultat attendu d'exemple, où le partage portait l'ID `nfs-tp` :

Ce support est la propriété de Dawan France.

```
nfs-tp           nfs     active         4046848          720896          3099648
17.81%
```

La colonne `Status` doit indiquer `active`. Un `inactive` signale un serveur injoignable ou un export refusé.

> **▶ À pratiquer maintenant — Atelier 5**
>
> L'autre côté du partage : monter vous-même le serveur NFS dans un conteneur, l'exporter, constater `root_squash` et le passage brut des UID, puis y écrire une sauvegarde de Proxmox.

Le stockage NFS apparaît immédiatement dans **Datacenter** → **Storage** avec son point de montage local `/mnt/pve/nfs-partage`, que Proxmox choisit lui-même d'après l'identifiant du stockage.

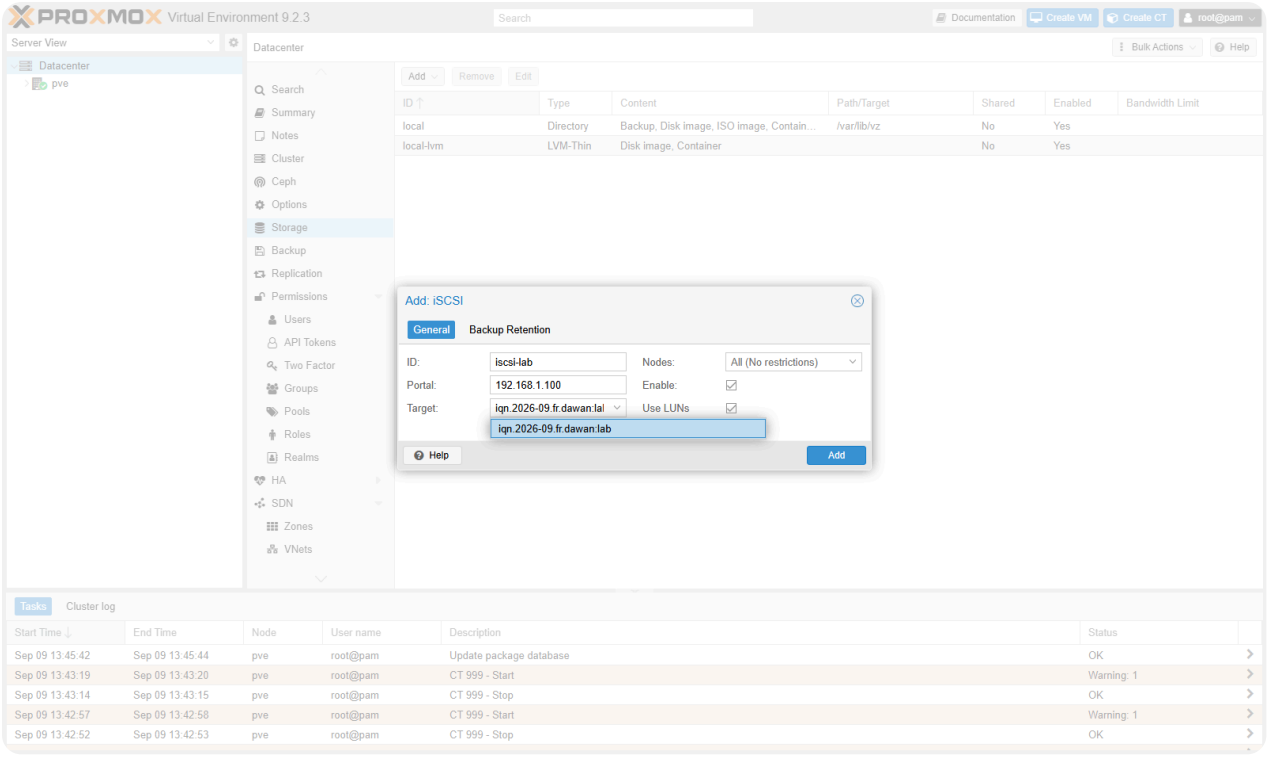
5.2 Ajouter une cible iSCSI, puis LVM par-dessus

Une cible iSCSI expose un disque (un **LUN**) sur le réseau. On la déclare à Proxmox en deux temps : la cible elle-même, puis un groupe LVM posé sur son LUN. C'est le montage « LVM sur iSCSI » des clusters qui disposent déjà d'un SAN.

Datacenter → Storage → Add → iSCSI

Champ	Valeur	Ce qu'il décide
ID	`iscsi-lab`	Le nom du stockage dans Proxmox
Portal	l'adresse de la cible	Proxmox l'interroge dès la saisie
Target	*(liste déroulante)*	Elle se remplit toute seule — c'est le signe que la découverte fonctionne
Use LUNs directly	✓ ou ✗	Voir l'encadré ci-dessous

Ce support est la propriété de Dawan France.



La fenêtre **Add: iSCSI**, au moment précis où la découverte a réussi.

- **Le champ `Target` est une liste déroulante, pas un champ libre.** Dès que le portail est saisi, Proxmox interroge la cible et affiche les IQN (*iSCSI Qualified Name*, nom unique d'une cible) qu'elle propose — ici `iqn.2026-09.fr.dawan:lab`.

  Si cette liste reste vide, inutile de continuer : soit la cible ne répond pas sur le port 3260, soit elle refuse votre nœud. C'est votre meilleur test de connectivité, et il ne coûte rien.

- **`Use LUNs directly` est cochée par défaut.** C'est le réglage à interroger : coché, le LUN entier ira à une seule VM ; décoché, il servira de socle à un groupe LVM. Voir l'encadré plus bas.

- **Il n'y a pas de champ `Content`**, contrairement au dialogue NFS : un LUN iSCSI n'accepte que des images disque, la question ne se pose pas.
- **Regardez la liste derrière la fenêtre.** `local` et `local-lvm` affichent `Shared: No`. Une fois ajouté, `iscsi-lab` y apparaîtra en `Shared: Yes` — la nature du protocole, pas un réglage.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# déclarer la cible iSCSI comme stockage
pvesm add iscsi iscsi-lab \
  --portal <ip-de-la-cible-iscsi> \
```

Ce support est la propriété de Dawan France.

```
  --target iqn.2026-09.fr.dawan:lab \
  --content images

pvesm status   # le nouveau stockage doit être actif
```

Résultat attendu :

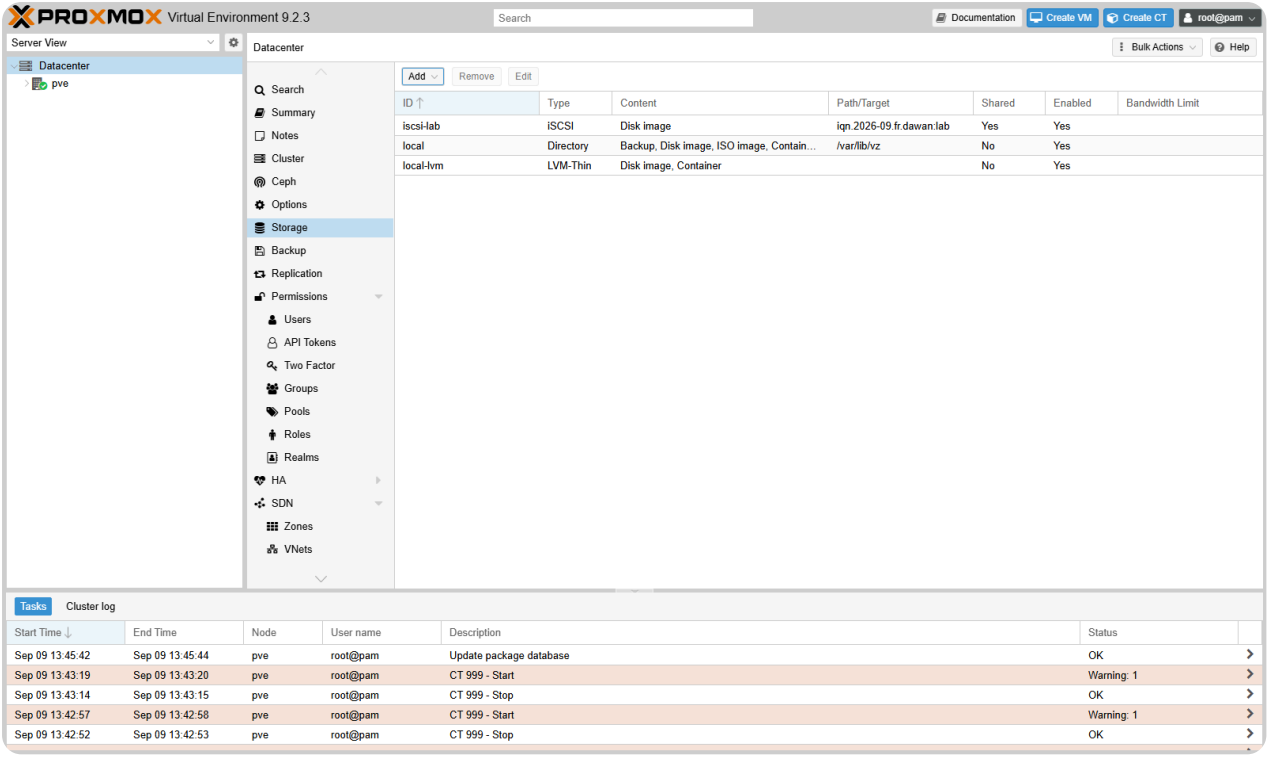
```
root@pxe211:~# pvesm add iscsi iscsi-lab --portal 10.44.1.98 --target iqn.2026-
09.fr.dawan:lab --content images
root@pxe211:~# pvesm status
Name             Type     Status           Total            Used       Available
%
iscsi-lab       iscsi     active               0               0               0
0.00%
local             dir     active        98559220        22417684        71095996
22.75%

root@pxe211:~# lsblk | grep sde
sde    8:64   0    10G  0 disk
```

> **Les trois zéros ne sont pas une erreur.** Un stockage iSCSI brut n'a aucun système de fichiers : Proxmox ne peut donc en calculer ni la taille ni l'occupation. `active` suffit à confirmer que la session est établie.
>
> La vraie vérification est `lsblk` : **un nouveau disque `/dev/sde` de la taille du LUN apparaît sur le nœud.**

En l'état, ce stockage est peu utilisable : Proxmox y placerait les disques de VM en accès direct au LUN, sans gestion de l'espace ni snapshots. **C'est la raison d'être de l'étape suivante** — poser LVM par-dessus, ce qui rend le LUN partageable entre nœuds et lui donne une vraie gestion des volumes.

Ce support est la propriété de Dawan France.



Le stockage apparaît avec `Shared: Yes`, alors que `local` et `local-lvm` sont à `No`. Ce n'est pas un réglage : c'est la nature du protocole. Un LUN iSCSI est joignable par tous les nœuds qui connaissent son portail.

C'est ce qui en fait un stockage de cluster. C'est aussi ce qui le rend dangereux sans une couche de coordination au-dessus.

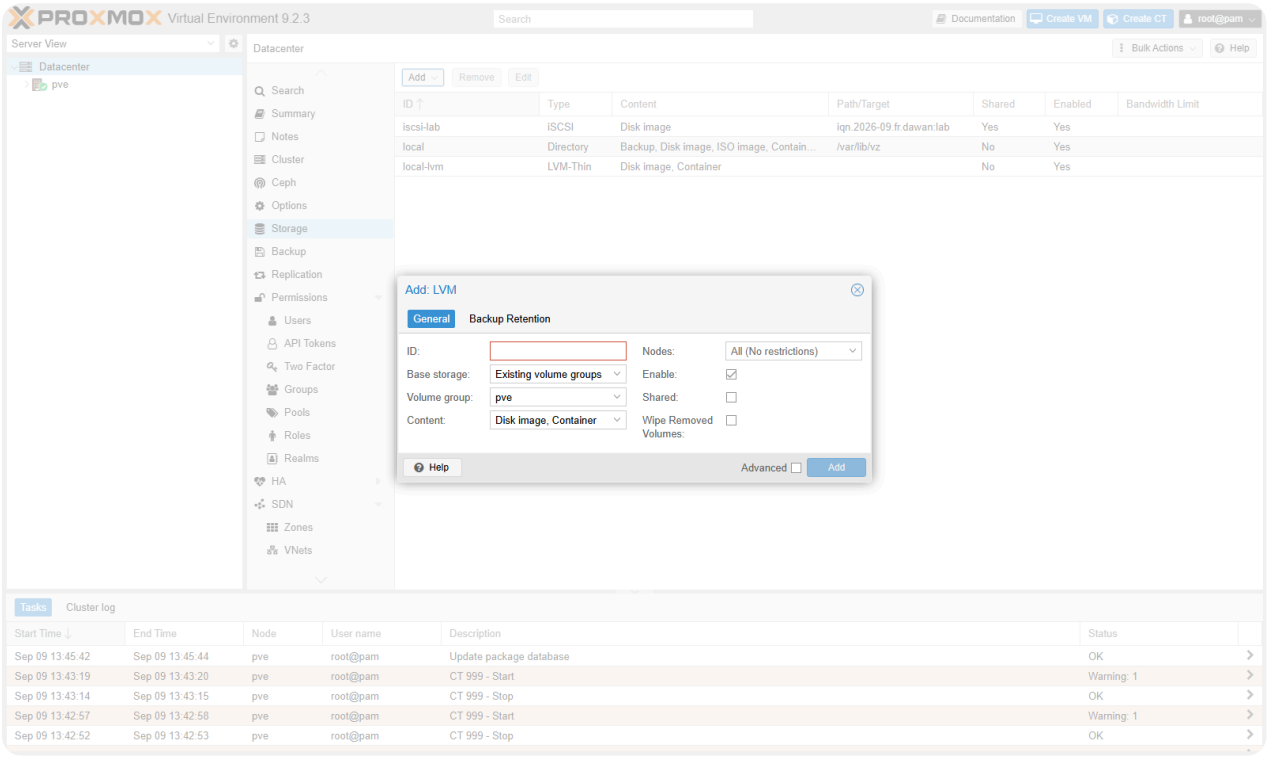
> **« Use LUNs directly », la case qui décide de tout.**
>
> **Cochée**, le LUN entier devient le disque d'**une seule** machine virtuelle. C'est simple, mais un LUN de 2 To consacré à une VM de 20 Go est un gâchis, et aucune seconde VM ne pourra y tenir.
>
> **Décochée**, le LUN reste brut et sert de **socle à un groupe de volumes LVM**. Proxmox y découpe alors autant de volumes logiques que de disques de VM — un LUN, N machines. C'est le montage de production, et c'est celui que décrit la suite.

L'empilement LVM :

Datacenter → Storage → Add → LVM

Ce support est la propriété de Dawan France.

Champ	Valeur	Le piège
ID	`lvm-sur-iscsi`	—
Base storage	`iscsi-lab (iSCSI)`	Le champ est sur « Existing volume groups » par défaut : c'est **lui** qu'il faut changer en premier, sinon la liste des volumes reste vide
Base volume	le LUN proposé	Ne se remplit qu'après avoir choisi le bon *Base storage*
Volume group	`vgiscsi`	Nom du VG à créer
Shared	✓	Indispensable en cluster



La fenêtre **Add: LVM**, telle qu'elle s'ouvre — et telle qu'il ne faut pas la laisser.

- **`Base storage` vaut `Existing volume groups` par défaut.** C'est le piège : tant que ce champ n'est pas changé, `Volume group` ne propose que les VG déjà présents sur le disque local (ici `pve`), et le LUN iSCSI reste invisible. **Changez ce champ en premier**, avant tout le reste.
- **`Content` vaut `Disk image, Container`.** Sur un LUN partagé de cluster, on laisse les deux : les disques de VM comme les rootfs de conteneurs pourront y vivre.
- **`Shared` n'est pas cochée par défaut.** Sur un stockage bâti sur iSCSI, il faut la cocher. C'est elle qui dit à Proxmox que tous les nœuds voient le même espace, et qui autorise la

Ce support est la propriété de Dawan France.

migration à chaud sans copier les disques.

- `Wipe Removed Volumes` **reste décochée.** Cochée, elle écrase chaque volume supprimé — sécurisant, mais chaque suppression de VM prend alors plusieurs minutes.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le meme resultat en ligne de commande, en trois temps :
# le LUN devient un volume physique, puis un groupe de volumes,
# et c'est ce groupe qu'on declare comme stockage Proxmox.
pvcreate /dev/sde
vgcreate vgiscsi /dev/sde   # puis un groupe de volumes
pvesm add lvm lvm-sur-iscsi --vgname vgiscsi --content images --shared 1   # déclaré comme stockage partagé
```

Résultat attendu :

```
root@pxe211:~# pvcreate /dev/sde
  Physical volume "/dev/sde" successfully created.
root@pxe211:~# vgcreate vgiscsi /dev/sde
  Volume group "vgiscsi" successfully created
root@pxe211:~# pvesm add lvm lvm-sur-iscsi --vgname vgiscsi --content images --
shared 1
root@pxe211:~# pvesm status | grep lvm-sur-iscsi
lvm-sur-iscsi     lvm     active        10481664               0        10481664
0.00%
```

Cette fois, les tailles sont renseignées : **LVM apporte la couche qui manquait au LUN brut**, et Proxmox sait enfin compter l'espace disponible.

> `--shared 1` **est indispensable dès que plusieurs nœuds voient le même LUN**, et son oubli est coûteux. Ce drapeau indique à Proxmox que le groupe de volumes est visible de tous. Le cluster coordonne alors les accès, et interdit qu'un même volume logique soit activé sur deux nœuds à la fois.
>
> Sans lui, chaque nœud croit le stockage local, et rien n'empêche deux VM d'écrire simultanément sur le même volume — la corruption est alors certaine.

Ce support est la propriété de Dawan France.

> En contrepartie, **LVM sur iSCSI ne gère pas les snapshots** : les instantanés LVM classiques ne fonctionnent pas sur un groupe de volumes partagé. Pour des snapshots sur stockage partagé, il faut Ceph ou ZFS over iSCSI.
>
> Proxmox VE 9 propose, en avant-première, une alternative : *Allow Snapshots as Volume-Chain*, avec des disques au format qcow2. À tester, pas à mettre en production tant qu'elle n'est pas déclarée stable.

Ces commandes ne s'exécutent qu'une fois, depuis un seul nœud. `pvcreate` et `vgcreate` écrivent sur le LUN lui-même : les autres nœuds voient le groupe de volumes sans rien refaire. Et `pvesm add` écrit dans `/etc/pve/storage.cfg`, commun à tout le cluster : le stockage est déclaré pour tous les nœuds d'un coup.

> `pvcreate` **détruit ce que porte le périphérique visé.** Vérifiez `lsblk -S` juste avant : le disque iSCSI n'est pas forcément `/dev/sde` sur votre machine, et l'ordre des lettres peut changer d'un démarrage à l'autre.
>
> Repérez-le par sa taille et sa colonne `TRAN=iscsi` — jamais par la lettre seule. Cette prudence n'est pas rhétorique : `pvcreate` sur le mauvais disque efface un système en une seconde.

> **Le piège du LUN partagé sans verrou.** Rien, dans le protocole iSCSI, n'empêche deux nœuds d'écrire simultanément au même endroit du même LUN. Deux systèmes de fichiers concurrents sur un même LUN, c'est une corruption certaine — pas probable, certaine.
>
> C'est exactement le rôle de la couche LVM déclarée `shared`. Proxmox y tient le registre des volumes logiques, et garantit qu'un même disque de VM n'est ouvert que par un nœud à la fois.
>
> **Ne formatez jamais un LUN partagé directement en ext4 ou XFS** : ces systèmes de fichiers supposent un accès exclusif. Si vous avez réellement besoin d'un système de fichiers partagé, il faut un système conçu pour cela — OCFS2, GFS2 — ou, dans l'univers Proxmox, Ceph.

À retenir : un partage **NFS** se déclare par **Datacenter** → **Storage** → **Add** → **NFS** : il accepte ISO, modèles, sauvegardes et disques. Une cible **iSCSI** se déclare par **Add** → **iSCSI**, puis on pose **LVM** sur son LUN ; `--shared 1` est indispensable dès que plusieurs nœuds le voient.

Dawan
Ce support est la propriété de Dawan France.

5.3 La conférence : Ceph, le stockage du futur

NFS et iSCSI supposent un serveur ou une baie de stockage, extérieurs aux nœuds. Ceph, le stockage distribué, s'en passe : cette conférence en explique le fonctionnement.

- 🎥 Ceph, le stockage du futur — Grégory Colpart, conférence du Capitole du Libre (2019, 1 h 10) : le fonctionnement interne de Ceph (OSD, groupes de placement, moniteurs). Elle montre aussi ses usages en virtualisation, dont Proxmox.

C'est une bonne introduction à la mise en cluster : elle fait comprendre comment plusieurs nœuds partagent un même stockage.

Ce support est la propriété de Dawan France.

6. Choisir et surveiller ses stockages

Tous les backends du chapitre sont maintenant connus. Cette dernière section les compare pour aider à choisir. Puis elle montre comment ZFS vérifie l'intégrité de ses disques, pour qu'un disque ne se dégrade pas sans bruit.

6.1 Comparatif des backends de stockage

Type	Snapshots	Migration à chaud sans copie des disques	Multi-nœuds	Usage recommandé
local-lvm (LVM-thin)	✅	❌	❌	VMs single-node, dev
ZFS (local)	✅	❌	❌	Single-node, intégrité des données
NFS	✅ (disques qcow2)	✅	✅	ISOs, templates, backups
iSCSI + LVM	❌	✅	✅ (avec lock)	Migration, SAN existant
Ceph RBD	✅	✅	✅	Production cluster HA

La colonne « migration » se lit précisément. Une VM sur stockage **local** peut aussi migrer à chaud. Mais Proxmox copie alors ses disques d'un nœud à l'autre pendant la migration, ce qui prend du temps et du réseau.

Sur un stockage **partagé**, seule la mémoire voyage. Les snapshots sur NFS viennent du format `qcow2`, qui les porte lui-même.

Ce tableau explique pourquoi Ceph est la solution privilégiée dans les clusters Proxmox en production : c'est le seul backend qui combine snapshots, migration en ligne et partage natif multi-nœuds.

Retenez cependant que ce choix n'est pas binaire : un même nœud Proxmox combine souvent plusieurs backends. `local-lvm` pour les VMs de test rapides à recréer ; du NFS pour les ISOs et les sauvegardes centralisées ; Ceph pour les VMs de production qui exigent la haute disponibilité. Le bon réflexe consiste à choisir le backend en fonction du besoin de chaque VM plutôt que d'imposer un backend unique à l'ensemble du cluster.

Dawan
Ce support est la propriété de Dawan France.

- 📖 Documentation Proxmox — Storage: NFS
- 📖 Configurer iSCSI dans Proxmox VE — pve-admin-guide

6.2 ZFS — vérification périodique de l'intégrité

ZFS détecte et corrige les erreurs grâce aux sommes de contrôle de chaque bloc, à condition de tout relire de temps en temps : c'est le **scrub**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Lancer un scrub complet sur le pool ; la commande rend la main aussitôt
zpool scrub datapool

# Une fois terminé : le bilan du scrub et des erreurs
zpool status datapool   # à lire : scan, errors
```

Résultat attendu (les lignes qui comptent) :

```
  scan: scrub repaired 0B in 00:00:01 with 0 errors on Tue Oct  6 19:48:05 2026
errors: No known data errors
```

`repaired 0B with 0 errors` **est le résultat souhaité** : ZFS a relu chaque bloc et n'a rien trouvé à corriger. Un `repaired` non nul signifie que la redondance a réparé une corruption silencieuse ; des erreurs répétées sur le même disque annoncent sa fin.

> **Un scrub sollicite fortement les disques** et ralentit les VM. Debian le planifie déjà la nuit, une fois par mois, le deuxième dimanche (`/etc/cron.d/zfsutils-linux`) : inutile d'en ajouter un.

À retenir : chaque stockage a ses forces : seul **Ceph** réunit snapshots, migration à chaud sans copie et partage natif. Côté surveillance : le **scrub** ZFS relit et vérifie chaque bloc ; Debian le planifie déjà chaque mois.

Ce support est la propriété de Dawan France.

Ateliers — chapitre 3

> **Infrastructure** : nœud `pxe2<N>1` — carte `ens19` pour le pont du cluster ; disques `/dev/sdc`, `/dev/sdd` disponibles pour ZFS.
>
> **Adressage** : votre poste dispose d'un réseau NATé qui lui est propre. **Toutes les ressources que vous créez portent une adresse fixe** — aucune ne dépend d'un DHCP de la salle.

État du lab en fin de chapitre :

- Pont du cluster `vmbr1` sur `ens19` (adresse Corosync `10.<site>.<N+10>.11/24`) et bridge isolé `vmbr9`
- Pool ZFS miroir `datapool` (snapshot/rollback validé) intégré comme stockage `zfs-vms`
- Disque de la VM 140 déplacé à chaud de `local-lvm` vers `zfs-vms` sans perte de ping (atelier 4), puis VM supprimée
- VLAN et NAT éprouvés sur `vmbr9` (atelier 2), puis `vmbr9` rendu isolé et conteneurs 311-312 supprimés
- Serveur NFS dans un conteneur, partage déclaré et sauvegarde écrite (atelier 5) — CT 260 et stockage `nfs-partage` supprimés

---

Atelier 1 — Réseau Linux et bridges

Contexte

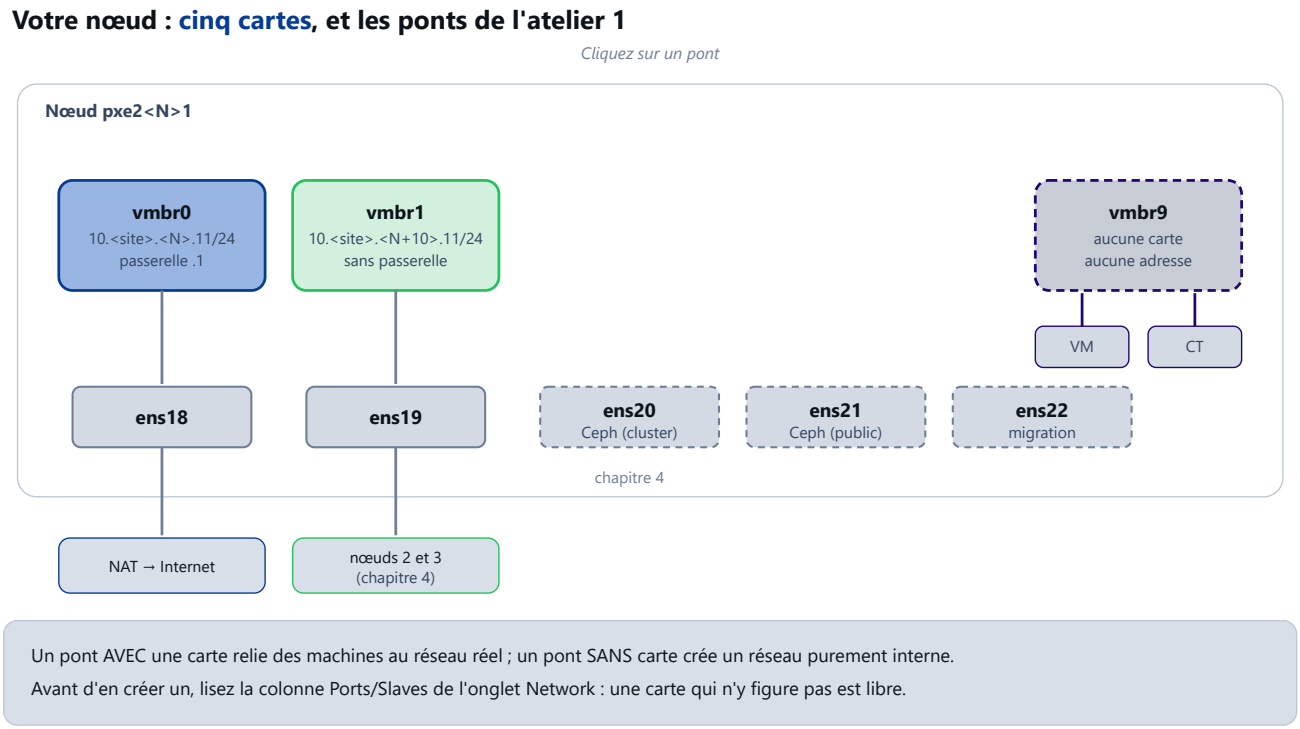
Tout ce qui se connecte au réseau dans Proxmox — VM, conteneur, cluster — passe par un **bridge**, un commutateur virtuel logiciel. Le nœud en a déjà un, `vmbr0`, qui le relie au monde. Savoir lire cette configuration et en créer d'autres est le préalable à tout le reste.

Le réseau isolé pour des VMs entre elles. Et surtout le réseau dédié qu'un cluster exige pour Corosync.

Résultat à obtenir : une lecture claire du réseau existant, et un bridge isolé sans interface physique (pour un réseau interne). Puis `vmbr1` monté sur la seconde carte physique, prêt à

Dawan Ce support est la propriété de Dawan France.

porter le trafic du cluster.



Les cinq cartes de votre nœud, et les trois ponts de l'atelier. `vmbr0` existe déjà sur `ens18`. Vous créez `vmbr9`, sans carte, puis `vmbr1` sur `ens19`.

Les cartes `ens20` à `ens22` restent libres. Cliquez sur un pont pour lire son rôle et le résultat obtenu.

Arborescence attendue

```
/etc/network/interfaces   ← la configuration réseau persistante du nœud
├── vmbr0   ← existant : pont principal, relié à la carte physique + Internet
├── vmbr9   ← à créer : pont ISOLÉ, sans interface physique (réseau interne)
└── vmbr1   ← à créer : pont dédié au cluster, sur ens19, adresse 10.<site>.
<N+10>.11/24
```

Ce que vous devez faire

1. **Explorer l'existant** : lister les interfaces et lire la configuration de `vmbr0` — quelle carte physique il englobe, quelle adresse il porte.
2. **Créer un bridge isolé**, sans interface physique attachée : il servira à faire communiquer des VMs entre elles sans les exposer au réseau.

Ce support est la propriété de Dawan France.

3. **Créer** `vmbr1` sur la carte `ens19`, celle du réseau interne *cluster* — c'est lui qui portera le réseau Corosync du cluster.

   💡 Un bridge sans interface physique = réseau purement interne au nœud. Un bridge avec interface = pont vers le réseau réel. Cette distinction commande toute l'architecture réseau de la formation.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Explorer le réseau existant

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Toutes les interfaces, une ligne chacune
ip -br link
ip -br addr show vmbr0   # l'adresse portée par vmbr0
bridge link show   # les ports rattachés aux ponts
# La configuration persistante, telle que Proxmox l'a écrite
cat /etc/network/interfaces
```

> **Machine d'exemple : nœud `pxe211`** (Proxmox VE 9.2.21), construit comme le vôtre : cinq cartes `ens18` à `ens22`. C'est le nœud du stagiaire 1 du site 59 : son `vmbr0` est en `10.59.1.11` ; le vôtre est en `10.<site>.<N>.11`.

Résultat attendu (extraits) :

```
ens18            UP             bc:24:11:01:a1:11
<BROADCAST,MULTICAST,UP,LOWER_UP>
ens19            DOWN           bc:24:11:01:b1:11 <BROADCAST,MULTICAST>
ens20            DOWN           bc:24:11:01:c1:11 <BROADCAST,MULTICAST>
ens21            DOWN           bc:24:11:01:d1:11 <BROADCAST,MULTICAST>
ens22            DOWN           bc:24:11:01:e1:11 <BROADCAST,MULTICAST>
vmbr0            UP             10.59.1.11/24 fe80::be24:11ff:fe01:a111/64
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master vmbr0 state
forwarding priority 32 cost 100
auto vmbr0
iface vmbr0 inet static
    address 10.59.1.11/24
    gateway 10.59.1.1
    bridge-ports ens18
    bridge-stp off
    bridge-fd 0
```

Ce support est la propriété de Dawan France.

`vmbr0` porte l'adresse du nœud ; `bridge link show` montre sa carte physique, `ens18`, avec `master vmbr0`. Les quatre autres cartes sont `DOWN` : elles mènent aux réseaux internes du lab, encore inutilisés. `bridge link show` liste aussi des interfaces `veth…`, `tap…`, `fwbr…` : ce sont les cartes des conteneurs et des VM en marche, branchées sur `vmbr0` — chaque invité est un port du commutateur virtuel.

Étape 1 bis — Un filet avant de toucher au réseau

Apply Configuration recharge le réseau du nœud à chaud. Une erreur (la carte de `vmbr0` reprise par un autre pont, une passerelle en trop) peut vous couper l'accès.

Votre nœud est distant : sans accès physique, personne ne pourrait le rattraper. Avant de créer quoi que ce soit, gardez une copie de la configuration et programmez son retour automatique.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Copie de la configuration réseau actuelle, qui fonctionne
cp /etc/network/interfaces /root/interfaces.avant-tp3
# Dans 10 minutes, la remettre en place et recharger le réseau, sauf si on annule avant
systemd-run --on-active=10min --unit=filet-reseau sh -c "cp /root/interfaces.avant-tp3 /etc/network/interfaces && ifreload -a"
```

Résultat attendu :

```
Running timer as unit: filet-reseau.timer
Will run service as unit: filet-reseau.service
```

Si un **Apply** vous coupait l'accès, il suffirait d'attendre la fin du délai : la configuration d'origine revient, et les ponts ajoutés disparaissent. Sur le lab, avec un délai d'une minute, le fichier a bien été restauré et le pont de test `vmbr9` supprimé (`Device "vmbr9" does not exist.`).

Étape 2 — Créer un bridge supplémentaire isolé

`Nœud → Network → Create → Linux Bridge` : Name `vmbr9`, Autostart ✓, Bridge ports *(laisser vide — bridge interne isolé)*, Comment `LAB-ISOLE` → **Create** → **Apply Configuration**.

Ce support est la propriété de Dawan France.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le nouveau pont existe-t-il ?
ip -br link show vmbr9
# Combien de ports y sont branchés ?
bridge link show | grep -c 'master vmbr9'
```

Résultat attendu :

```
vmbr9            UNKNOWN        8e:1d:76:ca:3e:4e
<BROADCAST,MULTICAST,UP,LOWER_UP>
0
```

Aucune ligne `master vmbr9` dans `bridge link show`. Le pont existe sans interface physique : un réseau isolé entre les VM et conteneurs qu'on y branchera.

Désactiver, puis réactiver le pont. L'interface n'a pas de bouton pour cela : la colonne **Active** de **Network** ne fait qu'afficher l'état. On passe par le Shell :

Dans le Shell du nœud :

```
ifdown vmbr9                # désactiver le pont : il disparaît du système
ip -br link show vmbr9      # vérifier
ifup vmbr9                  # le réactiver, d'après /etc/network/interfaces
ip -br link show vmbr9      # vérifier
```

Résultat attendu (`pxe212`) :

```
Device "vmbr9" does not exist.
vmbr9            UNKNOWN        6a:09:e0:40:4f:f9
<BROADCAST,MULTICAST,UP,LOWER_UP>
```

Désactivé, le pont n'existe plus, et la colonne **Active** de **Network** passe à `No` ; réactivé, il revient. Sa déclaration, elle, reste dans `/etc/network/interfaces` : c'est **Autostart** (la ligne `auto vmbr9`) qui le remonte à chaque démarrage du nœud.

Étape 3 — Créer `vmbr1` , le pont dédié au cluster

Ce support est la propriété de Dawan France.

Votre serveur a une seconde carte physique inutilisée. Elle portera le réseau réservé aux échanges du cluster.

Chaque carte de votre nœud mène à un réseau du lab :

Carte	Réseau interne	Usage
`ens18`	`10.<site>.<N>.0/24` (NAT)	déjà sous `vmbr0` : gestion, interface web, Internet
`ens19`	`10.<site>.<N+10>.0/24`	**cluster (Corosync)** — cet atelier
`ens20`	`10.<site>.<N+20>.0/24`	Ceph, réseau du cluster
`ens21`	`10.<site>.<N+30>.0/24`	Ceph, réseau public
`ens22`	`10.<site>.<N+40>.0/24`	migration des VM

> **Lisez la liste réelle avant de choisir une carte.** Sur les nœuds du lab, les cartes s'appellent `ens18` à `ens22`. La carte à prendre est celle qui n'apparaît dans la colonne `Ports/Slaves` d'aucun pont.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Lister les cartes et leur etat. Celle qui porte deja vmbr0 est prise ;
# une carte "DOWN" sans bridge associe est celle que vous cherchez.
ip -br link

# La meme lecture, cote Proxmox, avec la correspondance nom <-> bridge
cat /etc/network/interfaces
```

`Nœud → Network → Create → Linux Bridge` :

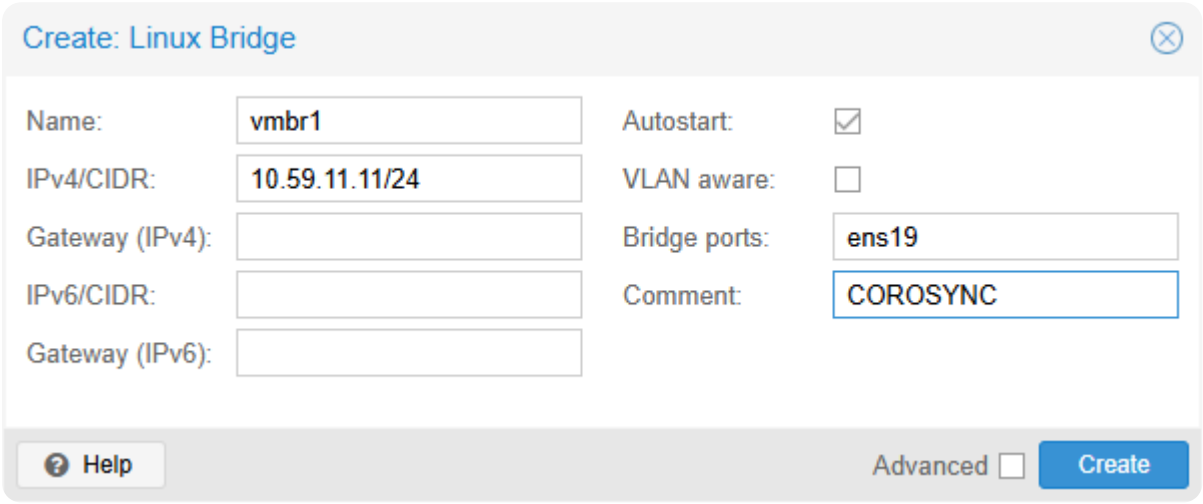
Champ	Valeur
Name	`vmbr1`
IPv4/CIDR	**l'adresse Corosync de votre nœud** : `10.<site>.<N+10>.11/24`
Gateway (IPv4)	*(laisser vide)*
Bridge ports	`ens19`
Autostart	✓
Comment	`COROSYNC`

Ce support est la propriété de Dawan France.

Les trois nœuds partageront ce réseau, une adresse chacun, avec le même dernier octet que sur `vmbr0` :

Nœud	IP Corosync	Exemple : stagiaire 1 du site 59 (le lab d'exemple)
`pxe2<N>1`	`10.<site>.<N+10>.11/24`	`10.59.11.11/24`
`pxe2<N>2`	`10.<site>.<N+10>.12/24`	`10.59.11.12/24`
`pxe2<N>3`	`10.<site>.<N+10>.13/24`	`10.59.11.13/24`

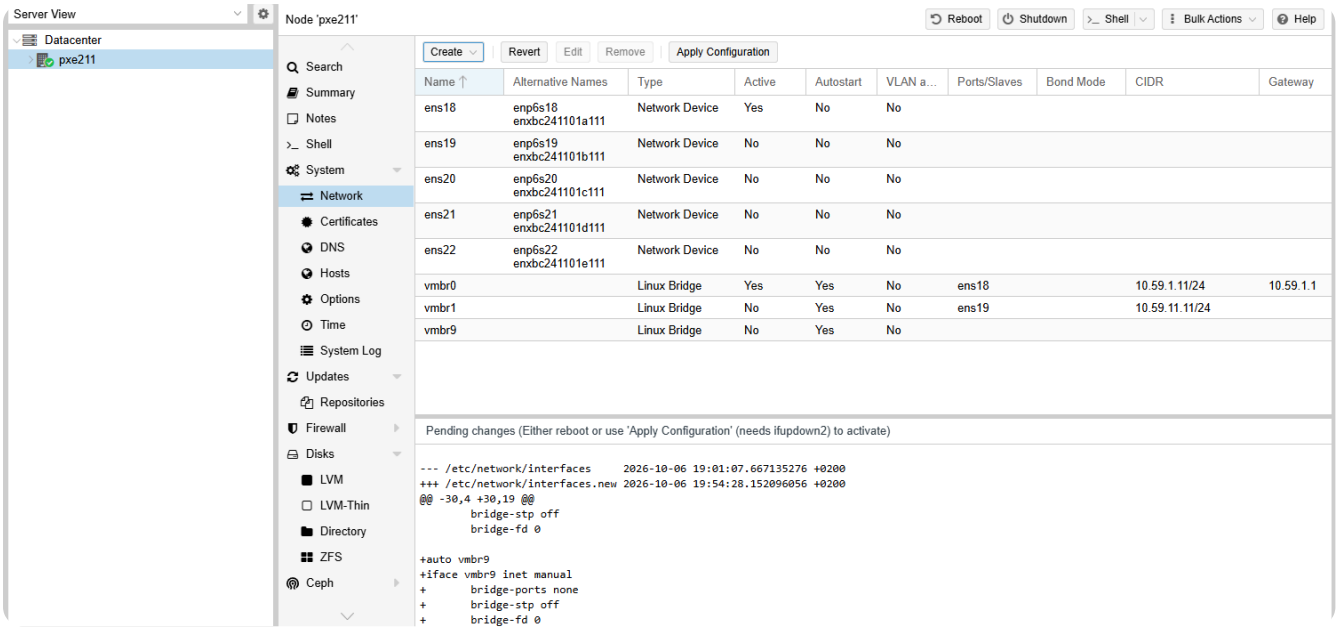
Le troisième octet vaut votre numéro de stagiaire **plus 10** : stagiaire 4 du site 59 → `10.59.14.x`. C'est le même calcul pour les trois nœuds — c'est ce qui les met sur le même réseau ; seul le dernier octet change.



Les valeurs du lab : **Name** `vmbr1`, **IPv4/CIDR** `10.59.11.11/24`, **Bridge ports** `ens19`, **Comment** `COROSYNC`, et **Gateway** laissé **vide**. **Autostart** est coché par défaut.

→ **Create**, puis **Apply Configuration**.

Ce support est la propriété de Dawan France.



Les deux ponts sont créés, mais **pas encore actifs** : colonne **Active** à `No`. Le panneau **Pending changes** montre la différence entre `/etc/network/interfaces` et `/etc/network/interfaces.new`, le fichier que Proxmox a préparé. Rien n'est appliqué tant que vous n'avez pas cliqué **Apply Configuration** (ou redémarré le nœud).

> **Pas de passerelle sur `vmbr1`.** Ce réseau ne sort pas : il ne reliera que les trois serveurs entre eux. Une seconde passerelle sur la machine casserait le routage par défaut, et vous perdriez l'accès à l'interface.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le pont du cluster, son adresse, puis sa configuration persistante
ip -br addr show vmbr1
grep -A 6 vmbr1 /etc/network/interfaces   # les lignes de vmbr1 dans le fichier réseau
bridge link show | grep ens19   # la carte ens19, rattachée à vmbr1
```

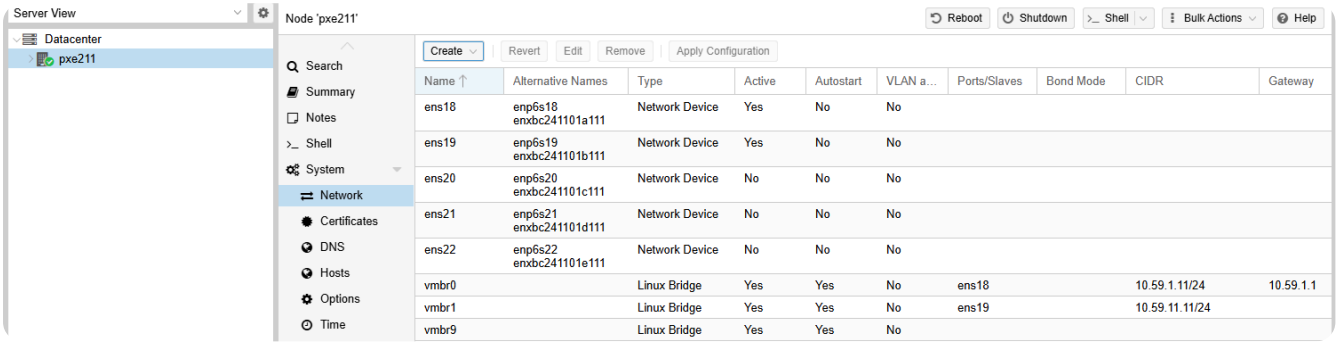
Résultat attendu :

```
vmbr1            UP             10.59.11.11/24 fe80::be24:11ff:fe01:b111/64
auto vmbr1
iface vmbr1 inet static
    address 10.59.11.11/24
    bridge-ports ens19
```

Ce support est la propriété de Dawan France.

```
    bridge-stp off
    bridge-fd 0
#COROSYNC
3: ens19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master vmbr1 state
forwarding priority 32 cost 100
```

`vmbr1` est `UP`, avec son adresse, sur `ens19` ; `bridge link show` montre désormais `ens19 … master vmbr1`. Le pont n'a encore personne à qui parler : votre serveur est seul à ce stade.



Après **Apply Configuration**, le panneau *Pending changes* a disparu. `vmbr0` (sur `ens18`) et `vmbr1` (sur `ens19`, `10.59.11.11/24`) portent chacun une adresse, et la colonne **Active** est passée à `Yes`, pour `ens19` aussi ; `vmbr9` n'a ni port ni adresse.

> **Ne confondez pas les deux ponts que vous venez de créer.** `vmbr9` isole (aucun port physique, aucune adresse) ; `vmbr1` relie les serveurs entre eux (seconde carte + adresse Corosync). Vider `vmbr1` de son port couperait le cluster.

Tout répond : annulez le filet, sinon il remettra l'ancienne configuration et supprimera vos deux ponts à la fin du délai.

Dans le Shell du nœud :

```
# Arrêter le minuteur : la restauration programmée n'aura pas lieu
systemctl stop filet-reseau.timer
# Vérifier qu'il n'est plus armé
systemctl is-active filet-reseau.timer
```

Résultat attendu : `inactive`.

Ce support est la propriété de Dawan France.

Atelier 2 — VLAN et NAT sur le pont isolé

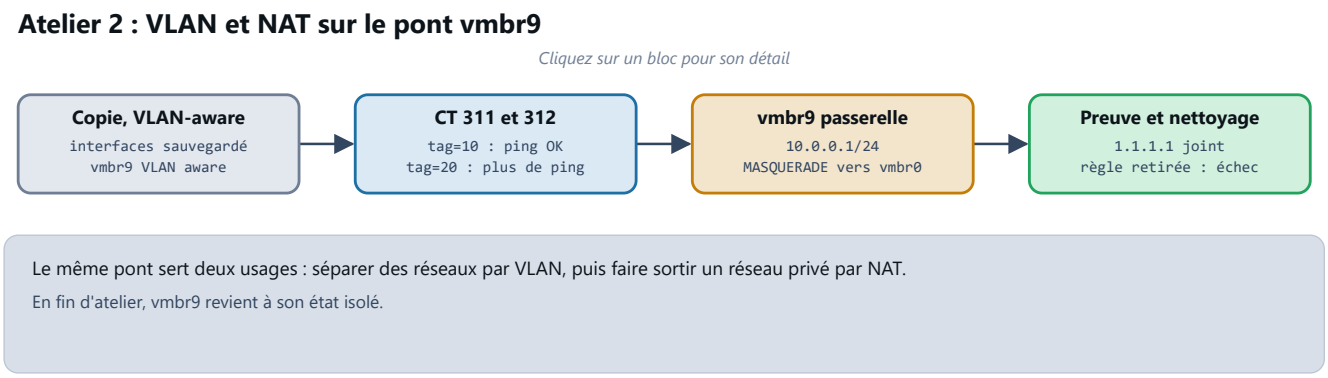
Contexte

Le pont `vmbr9` de l'atelier 1 isole ses invités. Cet atelier le fait travailler de deux façons, celles du cours §2.

D'abord il **segmente** : rendu VLAN-aware, il sépare deux conteneurs branchés sur lui selon leur VLAN. Puis il **fait sortir** : il devient la passerelle d'un réseau privé, que le nœud traduit vers l'extérieur par NAT.

Tout se passe sur `vmbr9`. Le pont `vmbr0`, qui porte votre accès au nœud, n'est jamais modifié.

Résultat à obtenir : deux conteneurs qui se voient dans le même VLAN et ne se voient plus dans deux VLAN différents. Puis un conteneur du réseau privé `10.0.0.0/24` qui sort par le NAT du nœud, avec la preuve que c'est bien la règle `MASQUERADE` qui le permet. Enfin `vmbr9` rendu à son état isolé.



Le schéma suit `vmbr9` de la copie de sécurité au nettoyage. Cliquez sur chaque bloc pour les commandes et ce qu'elles prouvent.

Arborescence attendue

```
Nœud pxe2<N>1
└── vmbr9   ← VLAN-aware, puis passerelle 10.0.0.1/24 avec NAT vers vmbr0
    ├── CT 311 « ct-vlan311 »   10.0.0.11/24   VLAN 10, puis sans VLAN, passerelle
10.0.0.1
    └── CT 312 « ct-vlan312 »   10.0.0.12/24   VLAN 10, puis VLAN 20
(les deux conteneurs et la configuration de l'atelier sont retirés à la fin)
```

Ce support est la propriété de Dawan France.

Ce que vous devez faire

1. **Garder une copie** de `/etc/network/interfaces`, comme à l'atelier 1.
2. **Rendre `vmbr9` VLAN-aware.**
3. **Créer deux conteneurs** sur `vmbr9`, dans le VLAN 10, et vérifier qu'ils se joignent. Puis passer le second dans le VLAN 20 et constater qu'ils ne se joignent plus.
4. **Faire de `vmbr9` une passerelle NATée** : adresse `10.0.0.1/24`, routage et règle `MASQUERADE` attachés au pont (cours §2.4).
5. **Sortir par le NAT** depuis le premier conteneur, sans VLAN : joindre la passerelle de votre `vmbr0`, puis une adresse extérieure. Retirer la règle, refaire le test, la remettre.
6. **Nettoyer** : conteneurs supprimés, `vmbr9` rendu isolé, règle NAT et routage retirés.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe211` (Proxmox VE 9.2.21), avec `vmbr9` dans l'état de fin d'atelier 1. Son `vmbr0` est en `10.59.1.11`, sa passerelle en `10.59.1.1` ; chez vous, `10.<site>.<N>.11` et `10.<site>.<N>.1`. Les adresses `10.0.0.x` sont internes à `vmbr9` : elles sont les mêmes chez vous.

Étape 1 — Une copie de la configuration réseau

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
cp /etc/network/interfaces /root/interfaces.avant-atelier2   # l'état de fin d'atelier 1, pour le nettoyage
cat /proc/sys/net/ipv4/ip_forward                            # le routage du nœud avant l'atelier : notez la valeur
```

Résultat attendu : `0` : le nœud ne route pas. L'étape 4 l'activera, l'étape 6 remettra la valeur notée.

Étape 2 — Rendre `vmbr9` VLAN-aware

Via l'interface : votre nœud → **System** → **Network** → `vmbr9` → **Edit** → cocher **VLAN aware** → **OK** → **Apply Configuration**.

En ligne de commande, ce sont les deux lignes du cours §2.1, ajoutées au bloc de `vmbr9` dans `/etc/network/interfaces` (avec `nano`), puis `ifreload -a` :

```
auto vmbr9
iface vmbr9 inet manual
    bridge-ports none
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes
    bridge-vids 2-4094
#LAB-ISOLE
```

Dans le Shell du nœud :

Dawan Ce support est la propriété de Dawan France.

```
ifreload -a                    # appliquer
bridge vlan show dev vmbr9     # les VLAN que le pont connaît
```

Résultat attendu :

```
port              vlan-id
vmbr9             1 PVID Egress Untagged
```

Le pont lui-même est dans le VLAN 1, sans étiquette : c'est son VLAN natif. Les VLAN 2 à 4094 lui sont permis (`bridge-vids`) ; ils apparaîtront avec les ports qui les utilisent.

Étape 3 — Deux conteneurs, un VLAN, puis deux

Via l'interface : Create CT, deux fois, avec le modèle Debian 13 du chapitre 1 :

Champ	CT 311	CT 312
CT ID / Hostname	`311` / `ct-vlan311`	`312` / `ct-vlan312`
Disk / Memory	`2` Gio / `256` Mio	`2` Gio / `256` Mio
Network → Bridge / VLAN Tag	`vmbr9` / `10`	`vmbr9` / `10`
Network → IPv4	Static, `10.0.0.11/24`, sans passerelle	Static, `10.0.0.12/24`, sans passerelle

Dans le Shell du nœud :

```
# Les deux conteneurs, dans le VLAN 10 de vmbr9 (nesting=1 : requis par le systemd de Debian 13)
pct create 311 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst --hostname ct-vlan311 --password dawan \
  --rootfs local-lvm:2 --memory 256 --unprivileged 1 --features nesting=1 \
  --net0 name=eth0,bridge=vmbr9,ip=10.0.0.11/24,tag=10
# le second, même VLAN
pct create 312 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst --hostname ct-vlan312 --password dawan \
  --rootfs local-lvm:2 --memory 256 --unprivileged 1 --features nesting=1 \
  --net0 name=eth0,bridge=vmbr9,ip=10.0.0.12/24,tag=10
pct start 311                              # démarrer le premier
```

Ce support est la propriété de Dawan France.

```
pct start 312                                  # puis le second
pct exec 311 -- ping -c 2 -W 2 10.0.0.12       # même VLAN
bridge vlan show | grep -A1 veth               # le VLAN de chaque port
```

Résultat attendu :

```
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
veth200i0         1 PVID Egress Untagged
fwbr200i0         1 PVID Egress Untagged
--
veth311i0         10 PVID Egress Untagged
veth312i0         10 PVID Egress Untagged
```

`Failed to create the network` au démarrage : `vmbr9` n'est pas VLAN-aware. Sur le lab, avec un conteneur étiqueté `tag=10` sur un `vmbr9` resté sans l'option :

```
run_buffer: 569 Script exited with status 255
lxc_create_network_priv: 3731 Success - Failed to create network device
lxc_spawn: 2032 Failed to create the network
startup for container '311' failed
```

L'étape 2 n'a pas pris effet : case **VLAN aware** cochée sans **Apply Configuration**, ou fichier modifié sans `ifreload -a`. Vérifiez avec `grep -A7 "iface vmbr9" /etc/network/interfaces` : les lignes `bridge-vlan-aware yes` et `bridge-vids 2-4094` doivent y être.

Les lignes présentes ne suffisent pas : `cat /sys/class/net/vmbr9/bridge/vlan_filtering` dit si le pont en marche les applique (sur le lab : `0` tant que `ifreload -a` n'a pas été lancé, `1` après). Appliquez, puis relancez `pct start 311; pct start 312` : sur le lab, le conteneur a alors démarré.

Si le message est `bridge 'vmbr9' does not exist`, c'est le pont lui-même qui manque : `ifup vmbr9` (atelier 1, étape 2).

Dans le résultat, les deux premières lignes sont le CT 200 du chapitre 1, branché sur `vmbr0` : ses ports sont dans le VLAN 1, celui par défaut. Si votre nœud porte d'autres conteneurs, leurs

Ce support est la propriété de Dawan France.

ports apparaissent de la même façon.

Chaque conteneur de l'atelier est un port du pont, dans le VLAN 10 : ils se joignent. `Egress Untagged` : le pont retire l'étiquette avant de livrer la trame, le conteneur ne la voit jamais (cours §2.2).

Sans `nesting=1`, `pct create` avertit : `WARN: Systemd 257 detected. You may need to enable nesting.` L'assistant de l'interface coche **Nesting** d'office.

Passez maintenant le CT 312 dans le VLAN 20 : CT 312 → **Network** → `net0` → **Edit** → **VLAN Tag** `20` → **OK**. En ligne de commande, recopiez l'adresse MAC (`pct config 312 | grep net0`) : sans `hwaddr=`, `pct set` en tire une nouvelle.

Dans le Shell du nœud :

```
pct set 312 --net0 name=eth0,bridge=vmbr9,hwaddr=<MAC-du-
312>,ip=10.0.0.12/24,tag=20   # le 312 dans le VLAN 20
pct exec 311 -- ping -c 2 -W 2 10.0.0.12    # VLAN 10 vers VLAN 20
bridge vlan show | grep -A1 veth            # le VLAN de chaque port
```

Résultat attendu :

```
2 packets transmitted, 0 received, 100% packet loss, time 1003ms
veth311i0         10 PVID Egress Untagged
veth312i0         20 PVID Egress Untagged
```

Même pont, même sous-réseau, même nœud : seul le VLAN diffère, et plus rien ne passe. C'est la segmentation du cours §2.1, sans un câble de plus.

Étape 4 — `vmbr9` devient une passerelle NATée

Via l'interface : `vmbr9` → **Edit** → **IPv4/CIDR** `10.0.0.1/24` (sans passerelle) → **OK**. Les règles de NAT, elles, n'ont pas de champ dans l'interface : elles s'ajoutent au fichier.

Dans le Shell du nœud, avec `nano /etc/network/interfaces`, le bloc de `vmbr9` devient celui du cours §2.4 :

```
auto vmbr9
iface vmbr9 inet static
```

Ce support est la propriété de Dawan France.

```
    address 10.0.0.1/24
    bridge-ports none
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes
    bridge-vids 2-4094
    post-up   echo 1 > /proc/sys/net/ipv4/ip_forward
    post-up   iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j
MASQUERADE
    post-down iptables -t nat -D POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j
MASQUERADE
#LAB-ISOLE
```

Dans le Shell du nœud :

```
ifreload -a                          # appliquer
ip -br addr show vmbr9               # l'adresse du pont
iptables -t nat -S POSTROUTING       # la règle posée par post-up
cat /proc/sys/net/ipv4/ip_forward    # le routage activé
```

Résultat attendu :

```
vmbr9            UP             10.0.0.1/24 fe80::6c52:eff:feb3:deb3/64
-P POSTROUTING ACCEPT
-A POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j MASQUERADE
1
```

Étape 5 — Sortir par le NAT

Le CT 311 sort du VLAN 10 et prend la passerelle : CT 311 → **Network** → `net0` → **Edit** → **VLAN Tag** vide, **Gateway (IPv4)** `10.0.0.1` → **OK**. L'adresse du pont est portée par son VLAN natif, le VLAN 1 sans étiquette (étape 2). On retire donc l'étiquette du CT 311 pour le mettre sur le même segment que sa passerelle.

Dans le Shell du nœud :

```
pct set 311 --net0 name=eth0,bridge=vmbr9,hwaddr=<MAC-du-
311>,ip=10.0.0.11/24,gw=10.0.0.1   # ou par l'interface
```

Ce support est la propriété de Dawan France.

```
pct exec 311 -- ping -c 2 -W 2 10.0.0.1             # la passerelle : le pont
pct exec 311 -- ping -c 2 -W 2 10.<site>.<N>.1      # la passerelle de votre
vmbr0, de l'autre côté du NAT
pct exec 311 -- ping -c 2 -W 2 1.1.1.1              # une adresse extérieure
iptables -t nat -L POSTROUTING -v -n | grep MASQUERADE   # le compteur de la
règle
```

Résultat attendu : `0% packet loss` pour les trois, puis :

```
    2   168 MASQUERADE  all  --  *      vmbr0   10.0.0.0/24          0.0.0.0/0
```

Le compteur a compté 2 paquets : un par connexion traduite, les deux pings vers l'extérieur. Le premier paquet d'une connexion passe par la règle ; les suivants suivent la traduction déjà établie.

La contre-épreuve : la règle retirée, puis remise.

Dans le Shell du nœud :

```
iptables -t nat -D POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j MASQUERADE   # retirer
la règle
pct exec 311 -- ping -c 2 -W 2 1.1.1.1                                  # sans la
règle
iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j MASQUERADE   # la
remettre
pct exec 311 -- ping -c 2 -W 2 1.1.1.1                                  # avec la
règle
```

Résultat attendu : `2 packets transmitted, 0 received, 100% packet loss`, puis `0% packet loss` : c'est bien la traduction d'adresse qui fait sortir le réseau privé.

> **Si `1.1.1.1` ne répond pas même avec la règle**, le réseau autour de votre nœud ne route peut-être pas vers Internet (cours §2.4). La passerelle de votre `vmbr0` et le compteur de la règle suffisent alors à prouver le NAT.

Étape 6 — Nettoyer

Ce support est la propriété de Dawan France.

Dans le Shell du nœud :

```
pct stop 311                                                    # arrêter le premier
conteneur
pct destroy 311 --purge                                         # le supprimer
pct stop 312                                                    # puis le second
pct destroy 312 --purge                                         # le supprimer
cp /root/interfaces.avant-atelier2 /etc/network/interfaces   # vmbr9 tel qu'à la
fin de l'atelier 1
ifreload -a                                                     # appliquer
iptables -t nat -S POSTROUTING                                  # la règle est-elle
partie ?
cat /proc/sys/net/ipv4/ip_forward                               # et le routage ?
```

Résultat attendu :

```
-P POSTROUTING ACCEPT
-A POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j MASQUERADE
1
```

> `ifreload -a` **n'exécute pas les** `post-down` **d'une configuration qu'on a retirée.** La règle `MASQUERADE` et le routage restent actifs après le retour au fichier d'origine, vu ci-dessus. Retirez-les à la main.
>
> **Dans le Shell du nœud** :
>
> ```
> iptables -t nat -D POSTROUTING -s 10.0.0.0/24 -o vmbr0 -j MASQUERADE   #
> retirer la règle
> echo 0 > /proc/sys/net/ipv4/ip_forward    # la valeur notée à l'étape 1 (0
> sur le lab)
> iptables -t nat -S POSTROUTING            # → -P POSTROUTING ACCEPT, seul
> ```

`ip -br link show vmbr9` répond alors `DOWN` et `NO-CARRIER` : un pont sans port ni invité n'a pas de lien. Il redevient `UP` dès qu'un invité s'y branche.

Ce support est la propriété de Dawan France.

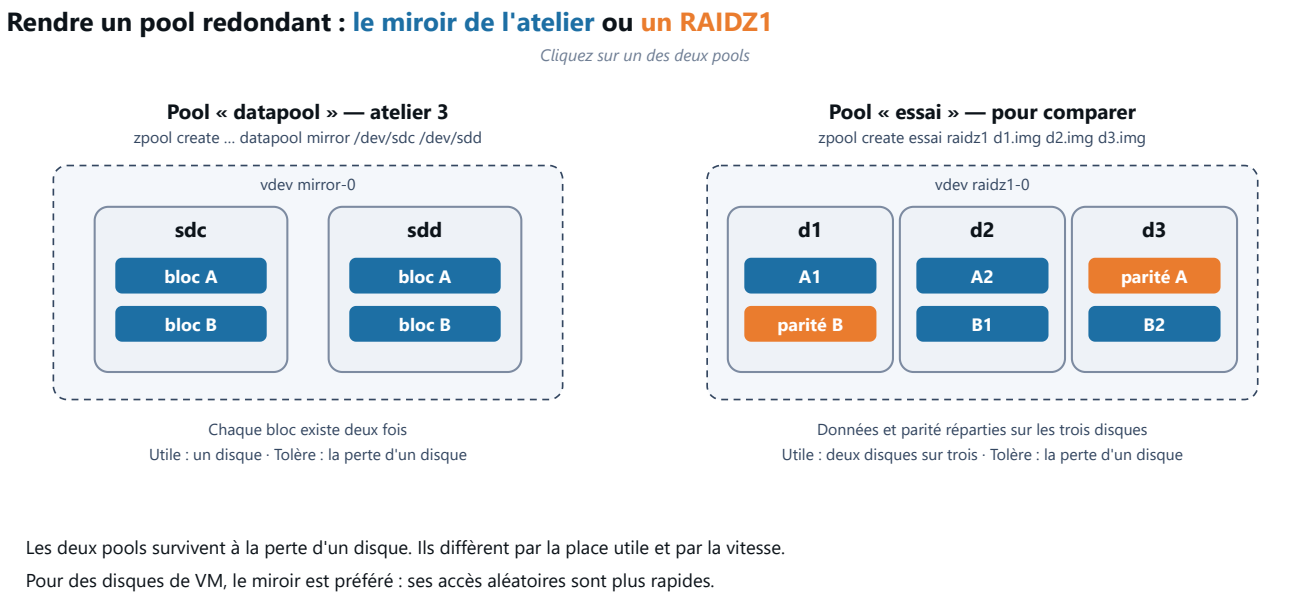
Atelier 3 — Pool ZFS en miroir

Contexte

ZFS est bien plus qu'un système de fichiers : il réunit la gestion des volumes, le RAID logiciel, la compression transparente et les snapshots instantanés en une seule couche. Sur ce nœud, deux disques lui sont réservés (`sdc` et `sdd`) pour un **miroir** — l'équivalent d'un RAID1, où chaque donnée existe en double.

Vous allez le construire de bout en bout : le pool en miroir, un dataset compressé, un snapshot avec retour arrière, puis son intégration comme stockage Proxmox.

Résultat à obtenir : un pool ZFS en miroir sur `sdc` / `sdd` . Deux datasets compressés selon leur usage, et la mesure de ce que chacun fait gagner. Un snapshot restauré avec succès, et le stockage visible et utilisable dans Proxmox.



Deux façons de rendre un pool redondant. Le **miroir** écrit la même donnée sur deux disques (comme RAID 1) : c'est ce que vous construisez ici, avec `sdc` et `sdd` .

RAIDZ1 répartit les données et une parité sur trois disques ou plus (comme RAID 5) et tolère la perte d'un disque. En production pour la virtualisation, le miroir est préféré pour ses performances en accès aléatoire. Cliquez sur un pool pour son détail.

Arborescence attendue

Ce support est la propriété de Dawan France.

```
Pool ZFS « datapool »  (miroir sdc + sdd)
├── dataset « datapool/vms »          ← compression lz4 (disques de VM : la vitesse)
│   └── snapshot @avant-modif          ← créé puis rollback
└── dataset « datapool/sauvegardes »  ← compression zstd (archives : la place),
comparé puis supprimé
Intégré dans Proxmox → stockage de type « ZFS »
```

Ce que vous devez faire

1. **Identifier les disques disponibles** et confirmer que `sdc` et `sdd` sont bien libres — surtout pas `sdb`, réservé à Ceph, ni `sda`, le système.
2. **Créer le pool en miroir** sur les deux disques, puis un **dataset** avec la **compression** activée.
3. **Un cran plus loin : un réglage par usage.** Créez un second dataset, `datapool/sauvegardes`, compressé en `zstd`. Écrivez les mêmes données texte dans les deux datasets et comparez ce qu'elles occupent réellement.

   Supprimez ensuite ce second dataset.
4. **Éprouver les snapshots** : prendre un snapshot du dataset, modifier des données, puis revenir en arrière et constater la restauration.
5. **Intégrer le pool dans Proxmox** comme stockage, et vérifier qu'il accepte bien des disques de VM.

   ⚠️ Attention au disque : monter ZFS sur `sdb` rendrait Ceph impossible (Ceph refuse un disque portant une signature ZFS). Respectez le plan des disques.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Identifier les disques disponibles

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Les disques entiers (-d), puis le contenu des deux disques visés
lsblk -d -o NAME,SIZE,TYPE,MODEL
fdisk -l /dev/sdc   # le disque sdc
fdisk -l /dev/sdd   # le disque sdd
```

Résultat attendu (nœud `pxe211`, construit comme le vôtre) :

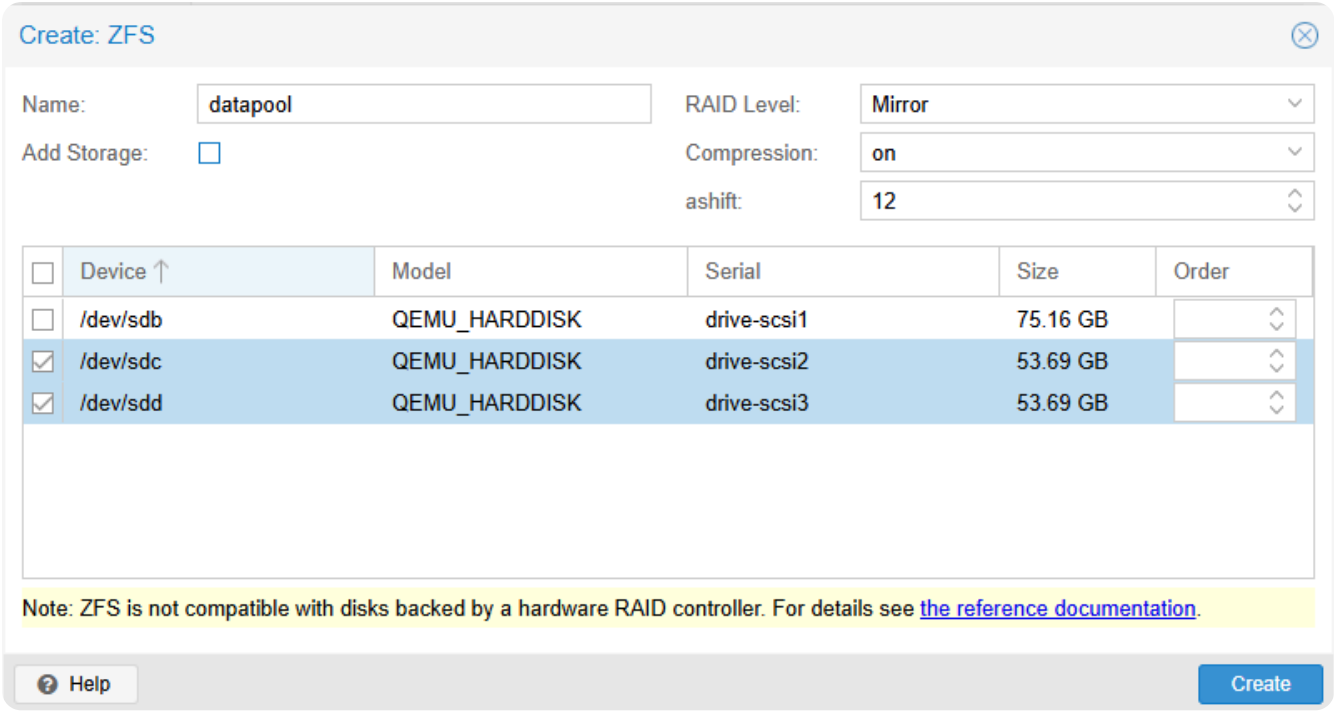
```
NAME  SIZE TYPE MODEL
sda   150G disk QEMU HARDDISK
sdb    70G disk QEMU HARDDISK
sdc    50G disk QEMU HARDDISK
sdd    50G disk QEMU HARDDISK
Disk /dev/sdc: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: QEMU HARDDISK
...
```

Quatre disques : `sda` (système), `sdb` (70 Go, Ceph) et `sdc`, `sdd` de même taille (50 Go). Pour ces deux derniers, `fdisk -l` n'affiche **aucune** ligne de partition — seulement la taille et le modèle.

> **Attention :** vérifiez bien que `sdc` et `sdd` sont des disques vides, sans données. La création du pool est irréversible.

Étape 2 — Créer un pool ZFS miroir

Ce support est la propriété de Dawan France.



La fenêtre remplie sur le lab. Seuls les **deux disques de taille identique** sont cochés — `sdc` et `sdd`, 53.69 GB chacun sur ce nœud (50 Gio exprimés en gigaoctets décimaux).

La taille varie d'un nœud à l'autre : seule compte l'égalité des deux disques. Dans un miroir, la capacité du pool est celle du plus petit disque, et `sdb` reste réservé aux OSD Ceph.

RAID Level est passé de `Single Disk` à `Mirror`, **Add Storage** est décoché. Les valeurs par défaut **Compression** `on` et **ashift** `12` conviennent.

Via l'interface : `pxe2<N>1 → Disks → ZFS → Create: ZFS` — Name `datapool`, RAID Level `Mirror`, Add Storage décoché (on l'ajoute manuellement à l'étape 5), Disk 1 `/dev/sdc`, Disk 2 `/dev/sdd` → **Create**.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Pool en miroir ; -f passe outre un avertissement de signature résiduelle (à
n'utiliser
# qu'après avoir vérifié à l'étape 1 que les disques sont bien vides)
zpool create -f -o ashift=12 datapool mirror /dev/sdc /dev/sdd

zpool status   # état du pool et de ses disques
zpool list   # taille et remplissage du pool
```

Ce support est la propriété de Dawan France.

Résultat attendu :

```
  pool: datapool
 state: ONLINE
config:
        NAME        STATE     READ WRITE CKSUM
        datapool    ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            sdc     ONLINE       0     0     0
            sdd     ONLINE       0     0     0

errors: No known data errors
```

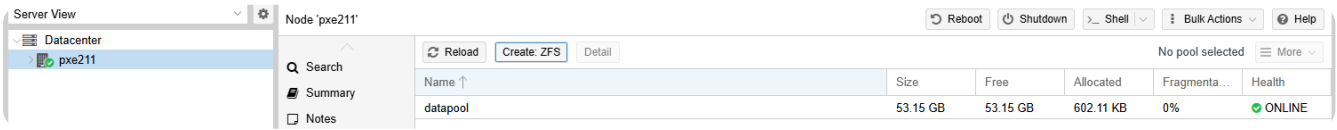
`zpool list` affiche une taille d'environ 49,5 Go : celle d'**un** disque, puisque le miroir écrit tout en double.

Résultat attendu, pool créé par l'interface :

```
    NAME                                      STATE     READ WRITE CKSUM
    datapool                                  ONLINE       0     0     0
      mirror-0                                ONLINE       0     0     0
        scsi-0QEMU_QEMU_HARDDISK_drive-scsi2  ONLINE       0     0     0
        scsi-0QEMU_QEMU_HARDDISK_drive-scsi3  ONLINE       0     0     0
NAME       SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH
ALTROOT
datapool  49.5G   624K  49.5G        -         -     0%     0%  1.00x    ONLINE
-
```

Remarquez les noms des disques. L'interface ne les désigne pas par `sdc` et `sdd`, mais par leur **identifiant stable** (`/dev/disk/by-id/…`). Un nom comme `sdc` peut changer au redémarrage suivant, si l'ordre de détection des disques change ; l'identifiant, jamais.

C'est la bonne pratique en production. En ligne de commande, `zpool create … mirror /dev/sdc /dev/sdd` garde les noms courts du tableau précédent.



Ce support est la propriété de Dawan France.

`datapool` apparaît sous `Disks → ZFS`, **Health** `ONLINE`. La taille affichée, 53.15 GB, est la même capacité que les 49,5 Gio de `zpool list`, exprimée en gigaoctets décimaux.

Étape 3 — Créer un dataset et activer la compression

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Un dataset dédié aux VM, compressé en lz4
zfs create datapool/vms
zfs set compression=lz4 datapool/vms   # compression lz4 sur ce dataset

zfs get compression,used,available datapool/vms   # compression, espace occupé et disponible
```

Résultat attendu :

```
NAME          PROPERTY     VALUE           SOURCE
datapool/vms  compression  lz4             local
datapool/vms  used         96K             -
datapool/vms  available    48.0G           -
```

`SOURCE local` signifie que la propriété est posée **sur ce dataset** ; un dataset enfant afficherait `inherited from datapool/vms`. C'est ainsi qu'on repère d'où vient un réglage.

Étape 3 bis — Un réglage par usage

Tous les datasets n'ont pas le même besoin. Les disques de VM demandent de la **vitesse** : `lz4` compresse peu mais presque sans coût.

Des archives de sauvegarde demandent de la **place** : `zstd` compresse davantage, au prix d'un peu plus de calcul. Sur ZFS, ce choix se fait dataset par dataset.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Un second dataset, compressé en zstd dès sa création
zfs create -o compression=zstd datapool/sauvegardes
# Les mêmes données texte (les modules Perl de Proxmox), écrites séparément dans chaque dataset
tar -cf /datapool/vms/perl.tar -C /usr/share perl5          # dans le premier
```

Ce support est la propriété de Dawan France.

```
dataset
tar -cf /datapool/sauvegardes/perl.tar -C /usr/share perl5   # puis dans le
second
sync   # forcer l'écriture sur disque avant de mesurer
# Taux de compression, place réellement occupée, taille avant compression
zfs get -o name,property,value compressratio,used,logicalused datapool/vms
datapool/sauvegardes
```

Résultat attendu :

```
NAME                  PROPERTY       VALUE
datapool/sauvegardes  compressratio  3.81x
datapool/sauvegardes  used           3.25M
datapool/sauvegardes  logicalused    12.2M
datapool/vms          compressratio  2.56x
datapool/vms          used           4.82M
datapool/vms          logicalused    12.2M
```

Les mêmes 12,2 Mo (`logicalused`) occupent 4,82 Mo en `lz4` et 3,25 Mo en `zstd` : un tiers de place en moins pour les sauvegardes. Deux pièges vus sur le lab :

- **Écrivez les données dans chaque dataset.** Un `cp` d'un dataset à l'autre du même pool ne réécrit pas les blocs : OpenZFS les **clone** (`zpool get feature@block_cloning datapool` répond `enabled`), et la copie garde la compression d'origine. Les deux taux affichaient alors `2.56x` .
- **Des données déjà compressées ne gagnent rien.** Avec une archive de `/usr/share/doc` , faite surtout de fichiers `.gz` , les deux datasets plafonnaient au même taux (`1.54x`), quel que soit l'algorithme.

Supprimez les deux archives de test, puis le dataset `sauvegardes` , qui ne sert plus.

Dans le Shell du nœud :

```
rm /datapool/vms/perl.tar /datapool/sauvegardes/perl.tar   # les deux archives de
test
zfs destroy datapool/sauvegardes                           # le dataset
```

Les ateliers suivants partent d'un pool qui ne contient que `datapool/vms` .

Ce support est la propriété de Dawan France.

Étape 4 — Snapshots ZFS

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Un fichier témoin dans le dataset (monté d'office sur /datapool/vms)
mkdir -p /datapool/vms/test
nano /datapool/vms/test/fichier.txt   # un fichier de test : la ligne « données importantes »

zfs snapshot datapool/vms@avant-modif   # snapshot du dataset
zfs list -t snapshot   # vérifier qu'il existe

# Simuler une erreur
rm /datapool/vms/test/fichier.txt

# Revenir au snapshot
zfs rollback datapool/vms@avant-modif

cat /datapool/vms/test/fichier.txt   # le contenu actuel du fichier
```

Résultat attendu :

```
NAME                       USED  AVAIL  REFER  MOUNTPOINT
datapool/vms@avant-modif     0B      -   104K  -
données importantes
```

Le snapshot n'occupe rien (`USED 0B`) tant que rien n'a changé ; après le `rollback` , `cat` affiche de nouveau `données importantes` — le fichier supprimé est revenu, sans aucune copie préalable.

Étape 5 — Intégrer ZFS dans Proxmox

Via l'interface : `Datacenter → Storage → Add → ZFS` — ID `zfs-vms` , ZFS Pool `datapool/vms` , Content `Disk image` , `Container` , Nodes *(laisser vide = tous les nœuds)* → **Add**.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Déclarer le dataset comme stockage : disques de VM et racines de conteneurs
pvesm add zfspool zfs-vms \
  --pool datapool/vms \
  --content images,rootdir

pvesm status   # le stockage zfs-vms doit apparaître
```

Résultat attendu : `zfs-vms zfspool active 50298460 104 50298356 0.00%`. Dans `Datacenter → Storage`, le stockage `zfs-vms` apparaît avec le type **ZFS**.

`zfspool` en ligne de commande, `ZFS` dans l'interface : c'est le même type. `zfspool` est son identifiant technique, celui de `pvesm` et de `/etc/pve/storage.cfg` ; `ZFS` est son libellé dans l'interface.

Attention au piège inverse : l'identifiant `zfs`, en ligne de commande, désigne un autre type, **ZFS over iSCSI**. Cela se lit dans le code de l'interface de Proxmox VE 9.2.21 : `zfspool` → `ZFS`, `zfs` → `ZFS over iSCSI`.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le pool, la compression, le stockage Proxmox
zpool list -H -o health datapool   # l'état du pool
zfs get compression datapool/vms   # la compression du dataset
pvesm status | grep zfs-vms   # le stockage zfs-vms est-il actif ?
```

Résultat attendu : `ONLINE`, `datapool/vms compression lz4 local`, et la ligne `zfs-vms zfspool active`.

Atelier 4 — Déplacer à chaud un disque de VM de LVM-thin vers ZFS

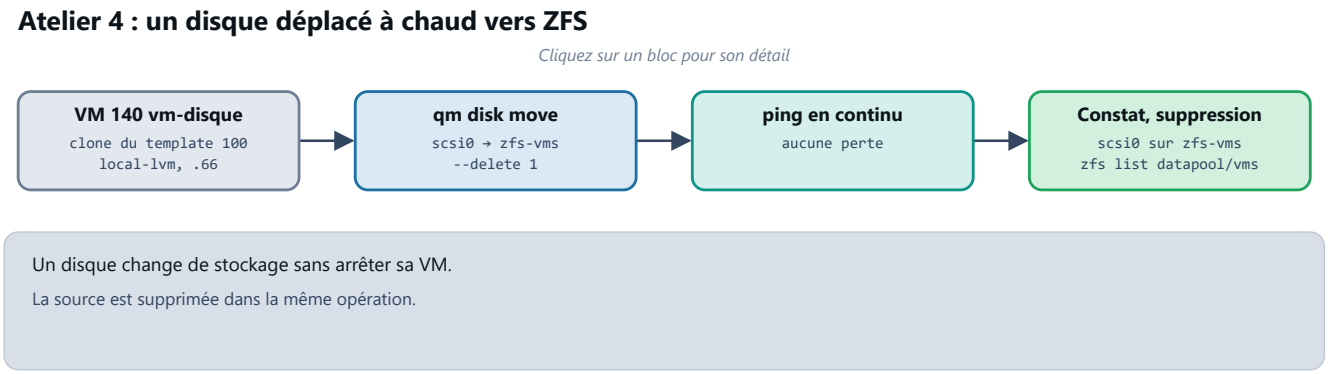
Contexte

Ce support est la propriété de Dawan France.

Votre nœud a maintenant deux stockages de disques : `local-lvm` (LVM-thin) et `zfs-vms` (le miroir de l'atelier 3). Une VM installée sur le premier peut passer sur le second **sans être arrêtée**.

Proxmox recopie le disque pendant que la VM tourne, puis bascule sur la copie (cours, §4.5). C'est le geste qui permet de profiter du miroir ZFS sans réinstaller une machine.

Résultat à obtenir : une VM 140 `vm-disque`, clonée sur `local-lvm`, dont le disque passe sur `zfs-vms` sans perte de ping ; l'ancien volume LVM supprimé ; puis la VM supprimée.



Le schéma suit le disque de la VM 140 de `local-lvm` vers `zfs-vms`. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
Nœud pxe2<N>1
├── Avant : VM 140 « vm-disque » (10.<site>.<N>.66), running   ← étapes 1 et 2
│   └── scsi0 : local-lvm:vm-140-disk-0        → volume LVM  pve/vm-140-disk-0
└── Après : la même VM, toujours running                        ← étapes 3 et 4
    └── scsi0 : zfs-vms:vm-140-disk-0          → zvol        datapool/vms/vm-140-
disk-0
```

Ce que vous devez faire

1. **Cloner** le template 100 du chapitre 2 en VM 140 `vm-disque`, clone complet sur `local-lvm`, adresse `.66`, et la démarrer.
2. **Relever l'état initial** : où est le disque, et quel volume LVM le porte.
3. **Déplacer le disque** vers `zfs-vms`, VM démarrée, en supprimant la source, pendant qu'un ping tourne.
4. **Constater** : où est le disque maintenant, et combien de pings ont été perdus.

Ce support est la propriété de Dawan France.

5. **Supprimer** la VM 140.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe211` (Proxmox VE 9.2.21, lab `10.59.1.x`), avec le template de 20 Go de l'atelier 7 du chapitre 2. La VM 140 y avait l'adresse `10.59.1.66` ; chez vous, `10.<site>.<N>.66`. Les captures, plus anciennes, montrent un disque de 3 Go.

Étape 1 — La VM de l'atelier

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm clone 100 140 --name vm-disque --full --storage local-lvm   # clone intégral de la VM 100
qm set 140 --ipconfig0 ip=10.<site>.<N>.66/24,gw=10.<site>.<N>.1   # son adresse fixe
qm start 140   # la démarrer
```

Attendez une trentaine de secondes, le temps que la VM démarre.

Étape 2 — L'état initial

Dans le Shell du nœud :

```
ping -c2 10.<site>.<N>.66   # la VM répond
qm config 140 | grep scsi0   # son disque, sur local-lvm
lvs pve | grep vm-140   # le volume LVM correspondant
```

Résultat attendu :

```
2 packets transmitted, 2 received, 0% packet loss, time 1014ms
scsi0: local-lvm:vm-140-disk-0,discard=on,iothread=1,size=20G
  vm-140-cloudinit pve Vwi-aotz--  4.00m data        9.38
  vm-140-disk-0    pve Vwi-aotz-- 20.00g data        6.17
```

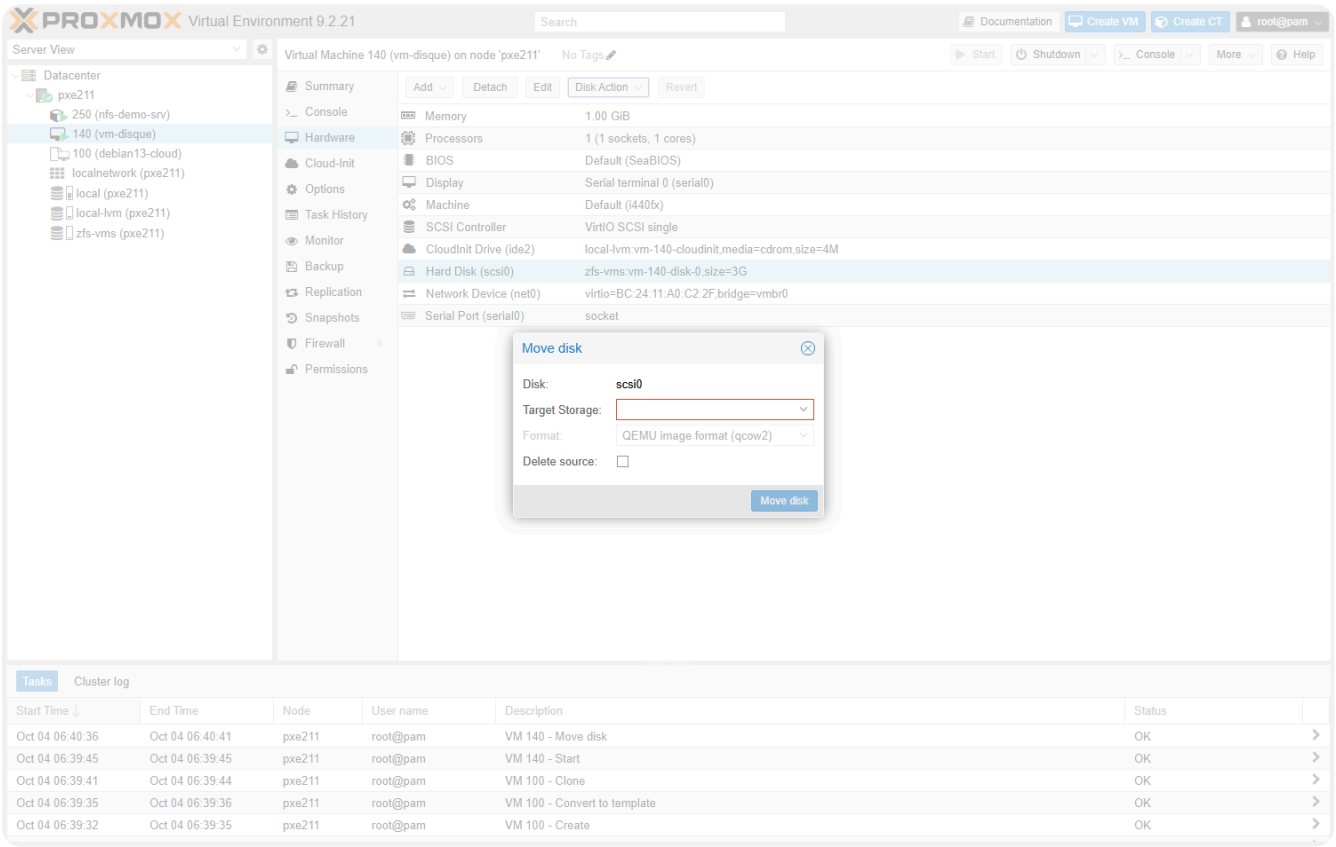
Le disque de 20 Go est un volume du pool LVM-thin `data`, rempli à 6 % : à peine plus d'1 Go réellement écrit.

Ce support est la propriété de Dawan France.

Étape 3 — Déplacer le disque, VM démarrée

Via l'interface : VM 140 → **Hardware** → sélectionner **Hard Disk (scsi0)** → **Disk Action** → **Move Storage**.

Champ	Valeur
Target Storage	`zfs-vms`
Format	grisé : un stockage ZFS n'accepte que le format `raw`
Delete source	coché



La fenêtre **Move disk** telle qu'elle s'ouvre. **Target Storage** est encadré en rouge tant qu'aucune cible n'est choisie. **Format** est grisé : c'est le stockage cible qui impose le format.

Delete source décide du sort de l'ancien volume. Non coché, il reste en `unused0` dans la configuration de la VM et continue d'occuper `local-lvm`. (Capture prise après le déplacement, d'où `zfs-vms` déjà affiché sur la ligne du disque.)

Ou en ligne de commande. **Dans le Shell du nœud**, avec un ping qui tourne pendant le déplacement :

Ce support est la propriété de Dawan France.

```
ping -i 0.2 10.<site>.<N>.66 > /tmp/ping140.txt 2>&1 &   # ping toutes les 0,2 s, en arrière-plan
qm disk move 140 scsi0 zfs-vms --delete 1   # déplacer le disque vers ZFS, VM allumée
kill %1                     # arrêter le ping
tail -2 /tmp/ping140.txt    # lire le bilan
```

Résultat attendu (lignes de progression abrégées) :

```
create full clone of drive scsi0 (local-lvm:vm-140-disk-0)
allocated target volume 'zfs-vms:vm-140-disk-0'
drive mirror is starting for drive-scsi0
mirror-scsi0: transferred 20.0 GiB of 20.0 GiB (100.00%) in 8s, ready
all 'mirror' jobs are ready
mirror-scsi0: Completing block job...
mirror-scsi0: Completed successfully.
mirror-scsi0: mirror-job finished
removing source volume 'local-lvm:vm-140-disk-0' after successful copy
  Logical volume "vm-140-disk-0" successfully removed.
```

Lisez les lignes dans l'ordre. Proxmox **crée** le volume cible, puis lance un **miroir** (*drive mirror*) : QEMU copie le disque et répercute sur la copie chaque écriture de la VM. Quand les deux sont identiques (`ready`), il **bascule** la VM sur la copie (`Completing block job`).

Enfin il **supprime** la source. L'opération a duré 12 secondes sur le lab ; le ping, une réponse toutes les 0,2 s, n'en a manqué aucune (57 reçues sur 57).

Étape 4 — Constater

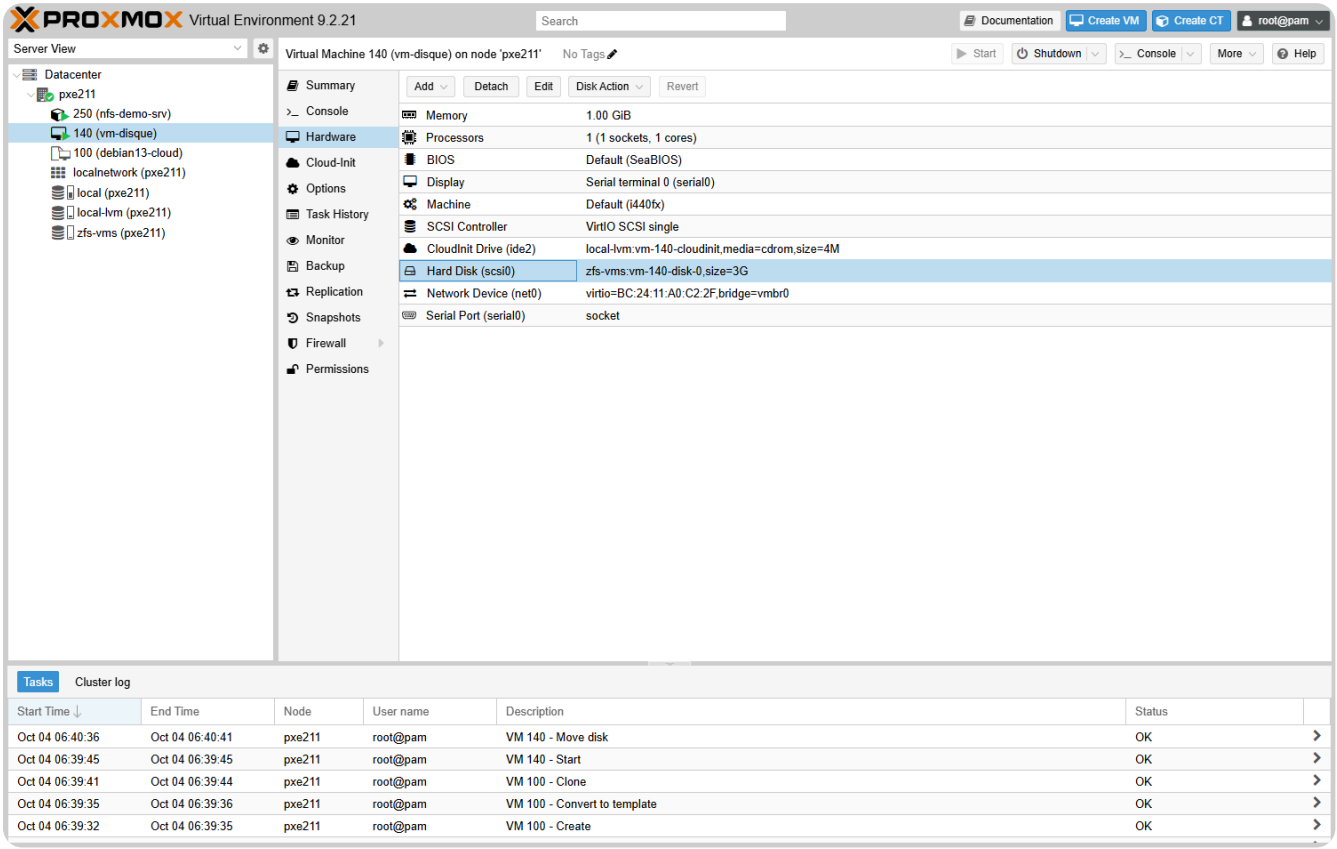
Dans le Shell du nœud :

```
qm config 140   # le disque est maintenant sur zfs-vms ; à lire : scsi0, unused
zfs list -o name,used,refer,volsize -r datapool/vms   # le volume ZFS de la VM
lvs pve | grep vm-140   # plus de volume LVM
```

Résultat attendu (les lignes qui comptent) :

Ce support est la propriété de Dawan France.

```
scsi0: zfs-vms:vm-140-disk-0,discard=on,iothread=1,size=20G
NAME                          USED  REFER  VOLSIZE
datapool/vms                 20.3G    96K        -
datapool/vms/vm-140-disk-0   20.3G   787M      20G
```



L'onglet **Hardware** après le déplacement. **Hard Disk (scsi0)** pointe sur `zfs-vms:vm-140-disk-0`.

Le **CloudInit Drive** (`ide2`) est resté sur `local-lvm` : seul le disque choisi a bougé. En bas, la tâche **VM 140 - Move disk** est `OK`.

Deux constats. Le disque est devenu un **zvol** ZFS de 20 Go, dont 787 Mo réellement écrits (`REFER`). Pourtant `USED` vaut 20,3G : sur un stockage `zfs-vms` qui n'est pas en *Thin provision*, Proxmox réserve d'avance toute la taille du disque.

Il n'y a plus de ligne `unused0` ni de volume `vm-140-disk-0` dans LVM : la source est supprimée. Et aucun ping n'a été perdu : la VM n'a pas été interrompue.

> **Un disque sur `zfs-vms` est local au nœud.** Dans un cluster, une VM dont le disque est sur ce stockage ne peut pas être migrée vers un nœud qui n'a pas de pool `datapool`.

Dawan
Ce support est la propriété de Dawan France.

> C'est pourquoi l'atelier se fait sur une VM dédiée, supprimée à la fin : vos VM 101 et 102 restent sur `local-lvm`.

Validation finale

Contrôle	Attendu
`qm config 140 | grep scsi0`	`zfs-vms:vm-140-disk-0`
Pings perdus pendant le déplacement	`0% packet loss`
`lvs pve | grep vm-140-disk-0`	rien

Nettoyage :

Dans le Shell du nœud :

```
qm stop 140                          # arrêter la VM 140
qm destroy 140 --purge               # puis la supprimer
zfs list -r datapool/vms   # vérifier que son volume a disparu
```

Résultat attendu : plus de `vm-140-disk-0` sous `datapool/vms`.

Atelier 5 — Un serveur NFS dans un conteneur

> Cours : §5.1 (ajouter un partage NFS à Proxmox).

Contexte

Au §5.1, le partage NFS est fourni par l'infrastructure : on ne fait que le **déclarer**. Ici, vous montez vous-même le serveur, dans un conteneur, pour voir l'autre côté : ce qu'il exporte, à qui, et avec quels droits.

Le serveur NFS du noyau Linux (`nfsd`) tourne **dans le noyau**, que le conteneur partage avec l'hôte. Un conteneur non privilégié n'a pas le droit de le piloter : ce conteneur-ci sera **privilégié**, un choix à justifier et à documenter (chapitre 1, conteneurs privilégiés).

Résultat à obtenir : un conteneur `ct-nfs` qui exporte `/srv/partage` vers votre réseau. Le partage monté depuis le nœud, où `root` est ramené à `nobody` (`root_squash`), et où un UID

Ce support est la propriété de Dawan France.

étranger passe tel quel.

Puis ce partage déclaré comme stockage de sauvegardes, et une sauvegarde écrite dessus. Tout est supprimé à la fin.



Le schéma suit le partage du serveur jusqu'au stockage de sauvegardes. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
CT 260 « ct-nfs » — privilégié, nesting, 10.<site>.<N>.81
└── /srv/partage            ← exporté vers 10.<site>.<N>.0/24, root_squash
    ├── test.txt            ← créé dans le conteneur
    ├── depuis-root         ← écrit par root du nœud : propriétaire 65534 (nobody)
    ├── equipe/rapport.txt  ← écrit par l'UID 2000 du nœud : propriétaire 2000
    └── dump/               ← sauvegarde du CT 200, via le stockage nfs-partage
```

Ce que vous devez faire

1. **Créer le conteneur 260** `ct-nfs`, **privilégié**, avec `nesting`, à l'adresse `.81`, depuis le modèle Debian 13. Y installer `nfs-kernel-server`.
2. **Exporter** `/srv/partage` vers votre réseau, avec `root_squash`.
3. **Monter le partage sur le nœud** et y écrire en `root` : expliquer le refus, le corriger.
4. **Écrire avec l'UID 2000** depuis le nœud : quel propriétaire voit le serveur ?
5. **Déclarer le partage** comme stockage `nfs-partage` (contenu : sauvegardes), et y sauvegarder le CT 200.
6. **Nettoyer** dans le bon ordre : le stockage d'abord, le conteneur ensuite.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Machine d'exemple : nœud** `pxe211` (Proxmox VE 9.2.21). Le conteneur 260 y a l'adresse `10.59.1.81` et exporte vers `10.59.1.0/24` ; chez vous, `10.<site>.<N>.81` et `10.<site>.<N>.0/24`. Les heures affichées par le conteneur sont en UTC (*Coordinated Universal Time*, temps universel), celles du nœud en heure de Paris.

Étape 1 — Le conteneur et le serveur

Via l'interface : Create CT — General : CT ID `260`, Hostname `ct-nfs`, **Unprivileged container décoché**, **Nesting** coché, mot de passe. Template : `debian-13-standard` ; Disks : `local-lvm`, `4` Go ; CPU `1` ; Memory `512`. Network : `vmbr0`, IPv4 `10.<site>.<N>.81/24`, passerelle `10.<site>.<N>.1` → **Finish**, puis **Start**.

Équivalent CLI, dans le Shell du nœud :

```
# Conteneur privilégié (--unprivileged 0), nesting activé
pct create 260 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst --hostname
ct-nfs \
  --unprivileged 0 --features nesting=1 --cores 1 --memory 512 --swap 512 --
rootfs local-lvm:4 \
  --net0 name=eth0,bridge=vmbr0,ip=10.<site>.<N>.81/24,gw=10.<site>.<N>.1 --
password proxmoxx
pct start 260   # le démarrer
```

Dans la console du conteneur 260 (CT 260 → **Console**, compte `root`) :

```
apt update                                  # la liste des paquets
apt install -y nfs-kernel-server            # le serveur NFS du noyau
systemctl is-active nfs-server              # le service doit être actif
```

Résultat attendu :

```
● nfs-server.service - NFS server and services
     Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; preset:
```

Ce support est la propriété de Dawan France.

```
enabled)
     Active: active (exited) since Tue 2026-10-06 18:28:55 UTC; 868ms ago
```

Sur Proxmox VE 9, le conteneur privilégié avec `nesting` suffit : aucun réglage AppArmor supplémentaire n'a été nécessaire.

Étape 2 — Exporter le répertoire

Dans la console du conteneur 260 :

Dans la console du conteneur 260 :

```
mkdir -p /srv/partage                                  # le répertoire à partager
nano /srv/partage/test.txt                             # un fichier de test : la
ligne « Fichier de test NFS »
nano /etc/exports                                      # le partage
```

La ligne à écrire dans `/etc/exports` : le partage, pour tout votre réseau, en lecture-écriture, root ramené à nobody.

```
/srv/partage 10.<site>.<N>.0/24(rw,sync,no_subtree_check,root_squash)
```

Dans la console du conteneur 260 :

```
exportfs -ra   # relire /etc/exports
exportfs -v    # ce qui est réellement exporté, options comprises
ls -ld /srv/partage   # propriétaire et droits du répertoire
```

Résultat attendu :

```
/srv/partage
10.59.1.0/24(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,root_squash,no_a
ll_squash)
drwxr-xr-x 2 root root 4096 Oct  6 18:29 /srv/partage
```

Dawan Ce support est la propriété de Dawan France.

Option	Effet
`rw`	lecture et écriture (`ro` : lecture seule)
`sync`	le serveur ne confirme une écriture qu'une fois sur disque
`no_subtree_check`	pas de vérification coûteuse de l'arborescence, recommandé pour un répertoire entier
`root_squash`	le `root` du client devient `nobody` sur le serveur : la protection par défaut

Étape 3 — Le nœud comme client

Dans le Shell du nœud :

```
showmount -e 10.<site>.<N>.81   # ce que le serveur propose
mkdir -p /mnt/essai-nfs         # le point de montage
mount -t nfs4 10.<site>.<N>.81:/srv/partage /mnt/essai-nfs   # monter le partage
cat /mnt/essai-nfs/test.txt     # lire le fichier de test
touch /mnt/essai-nfs/depuis-root   # écrire en root
```

Résultat attendu :

```
Export list for 10.59.1.81:
/srv/partage 10.59.1.0/24
Fichier de test NFS
touch: cannot touch '/mnt/essai-nfs/depuis-root': Permission denied
```

Vous êtes `root` sur le nœud, et pourtant refusé. `root_squash` vous a ramené à `nobody`, et `/srv/partage` appartient à `root`, en `755` : `nobody` n'y écrit pas.

Dans la console du conteneur 260 :

```
chown nobody:nogroup /srv/partage   # le répertoire à nobody
chmod 775 /srv/partage              # écriture pour nobody, traversée pour les autres
```

Dans le Shell du nœud :

Ce support est la propriété de Dawan France.

```
touch /mnt/essai-nfs/depuis-root   # écrire en root, de nouveau
ls -ln /mnt/essai-nfs   # -n : les numéros d'UID, pas les noms
```

Résultat attendu :

```
-rw-r--r-- 1 65534 65534  0 Oct  6 20:29 depuis-root
-rw-r--r-- 1     0     0 20 Oct  6 20:29 test.txt
```

Le fichier écrit par `root` appartient à **65534**, l'UID de `nobody` : c'est `root_squash` au travail.

Étape 4 — Un UID étranger

Dans la console du conteneur 260 :

```
mkdir /srv/partage/equipe             # un sous-répertoire
chown 2000:2000 /srv/partage/equipe   # un répertoire à l'UID 2000, qui n'existe pas ici
```

Dans le Shell du nœud :

```
# Écrire sous l'UID 2000 et le GID 2000, sans créer de compte
setpriv --reuid=2000 --regid=2000 --clear-groups touch /mnt/essai-nfs/equipe/rapport.txt
```

Dans la console du conteneur 260 :

```
ls -l /srv/partage/equipe   # le contenu, vu du serveur
getent passwd 2000   # existe-t-il un compte 2000 ?
```

Résultat attendu :

```
-rw-r--r-- 1 2000 2000 0 Oct  6 18:29 rapport.txt
```

`getent` ne répond rien : aucun compte 2000 n'existe dans le conteneur. NFS transmet le **numéro** d'UID tel quel, sans traduction par nom. Deux machines qui partagent des fichiers doivent donc avoir les mêmes UID.

Ce support est la propriété de Dawan France.

> Avec un `chmod 770` sur `/srv/partage` au lieu de `775`, l'UID 2000 ne peut même pas **traverser** le répertoire : `Permission denied`, alors que le sous-répertoire `equipe` lui appartient. Pour atteindre un fichier, il faut le droit `x` sur chaque répertoire du chemin.

Étape 5 — Le partage comme stockage de sauvegardes

Dans le Shell du nœud, démontez d'abord le montage d'essai :

```
umount /mnt/essai-nfs                          # démonter
rmdir /mnt/essai-nfs                           # retirer le point de montage
```

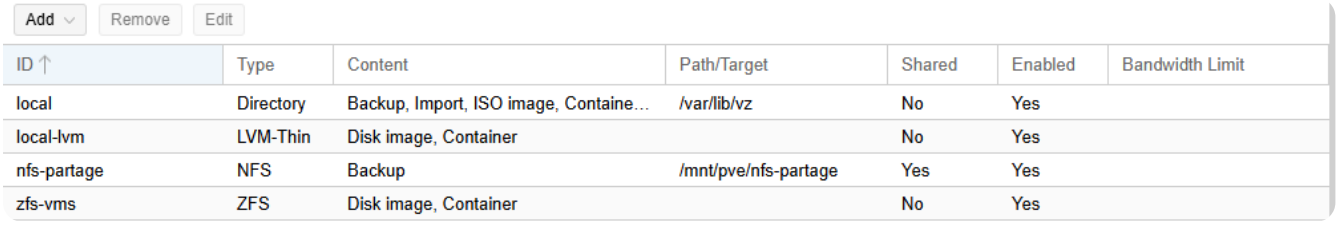
Via l'interface : Datacenter → Storage → Add → NFS — ID `nfs-partage`, Server `10.<site>.<N>.81`, Export `/srv/partage`, Content **Backup** → **Add**. Puis CT 200 → **Backup** → **Backup now** — Storage `nfs-partage`, Mode **Snapshot**, Compression **ZSTD** → **Backup**.

Équivalent CLI, **dans le Shell du nœud** :

```
pvesm add nfs nfs-partage --server 10.<site>.<N>.81 --export /srv/partage --
content backup   # déclarer le partage
pvesm status | grep nfs-partage                                    # actif ?
vzdump 200 --storage nfs-partage --mode snapshot --compress zstd   # une
sauvegarde dessus
```

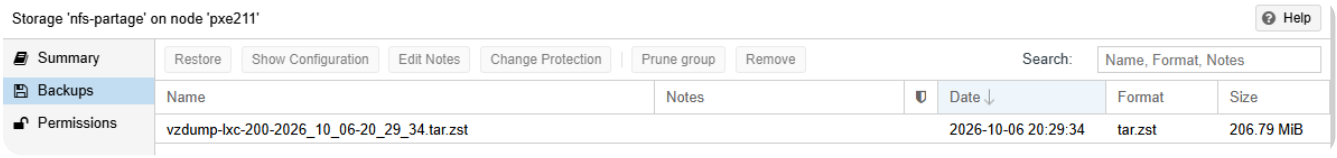
Résultat attendu (extrait) :

```
nfs-partage         nfs     active         4046848          720896
3099648   17.81%
INFO: Finished Backup of VM 200 (00:00:07)
INFO: Backup job finished successfully
```



Ce support est la propriété de Dawan France.

Datacenter → **Storage** : `nfs-partage` est de type **NFS**, contenu **Backup**, monté sous `/mnt/pve/nfs-partage`. Sa colonne **Shared** vaut **Yes**, contrairement aux trois stockages locaux.



Le stockage `nfs-partage` → **Backups** : la sauvegarde du CT 200, 206,79 Mio. Sa capacité (4 Go) est celle du disque du conteneur 260 : un partage n'offre que la place de ce qu'il exporte.

Dans la console du conteneur 260 :

```
ls -ln /srv/partage/dump   # la sauvegarde, vue du serveur
```

Résultat attendu :

```
-rw-r--r-- 1 65534 65534       742 Oct  6 18:29 vzdump-lxc-200-2026_10_06-20_29_34.log
-rw-r--r-- 1 65534 65534 216834678 Oct  6 18:29 vzdump-lxc-200-2026_10_06-20_29_34.tar.zst
```

Proxmox écrit ses sauvegardes en `root` : sur ce partage, `root_squash` les fait appartenir à `nobody`, comme le fichier `depuis-root` de l'étape 3.

Validation finale

Contrôle	Attendu
`exportfs -v` dans le conteneur	`/srv/partage` vers votre réseau, `root_squash`
Propriétaire de `depuis-root`	`65534`
Propriétaire de `equipe/rapport.txt`	`2000`
`pvesm status | grep nfs-partage`	`active`

Nettoyage — dans cet ordre, **dans le Shell du nœud** :

```
pvesm remove nfs-partage        # 1. le stockage d'abord : Proxmox cesse de l'utiliser
```

Ce support est la propriété de Dawan France.

```
umount /mnt/pve/nfs-partage     # 2. pvesm remove ne démonte pas
pct stop 260                            # 3. le serveur ensuite : l'arrêter
pct destroy 260 --purge                 # puis le supprimer
pvesm status                    # nfs-partage a disparu, tout est active
```

Résultat attendu (`pvesm status`) :

```
local             dir     active        24527236         9683680        13680676
39.48%
local-lvm     lvmthin     active        64876544        41929710        22946833
64.63%
zfs-vms       zfspool     active        29965724              96        29965628
0.00%
```

Détruire le conteneur avant de retirer le stockage laisserait un `nfs-partage` déclaré mais injoignable. Sur le lab, serveur arrêté : `pvesm status` le montre `inactive`, avec une capacité de 0. Et `umount -f -l` est resté bloqué plusieurs minutes, jusqu'au redémarrage du serveur.

Ce support est la propriété de Dawan France.

Chapitre 4 — Cluster, Ceph et haute disponibilité

Jusqu'ici, toute votre infrastructure tenait sur un seul serveur : s'il s'arrête, tout s'arrête. Ce chapitre fait sortir Proxmox de ce nœud unique. Deux serveurs vous rejoignent, et les trois deviennent un **cluster** qui partage sa configuration, déplace ses machines d'un nœud à l'autre, et relance seul celles d'un nœud tombé.

Le fil conducteur suit l'ordre dans lequel on construit réellement une telle infrastructure.

D'abord le **cluster** : ce qui le fait tenir (le quorum, un réseau dédié à Corosync, deux liens), puis sa construction sur le pont `vmbr1` préparé au chapitre 3. Vient ensuite sa vie courante, avec la migration. Puis **Ceph**, qui transforme le disque `sdb` de chaque nœud en un stockage partagé.

Vient alors la **haute disponibilité** : elle n'a de sens que si les disques des VM sont visibles de partout, donc sur Ceph. Le chapitre se termine sur la **maintenance** d'un nœud sans interrompre ses machines.

Objectifs du chapitre

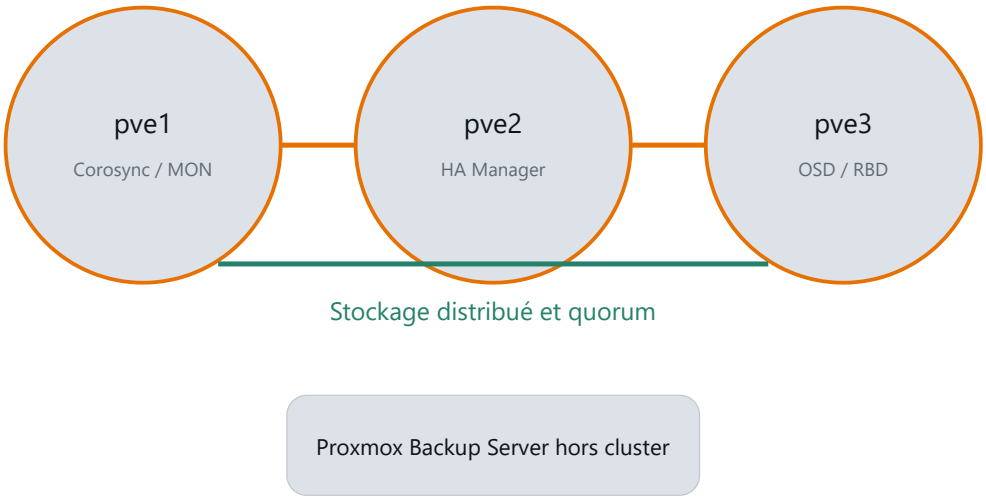
À l'issue de ce chapitre, les stagiaires seront capables de :

- **Constituer** un cluster Proxmox à plusieurs nœuds avec `pvecm`, et en lire l'état (quorum, Corosync)
- **Migrer** une VM à froid et à chaud entre nœuds
- **Déployer** Ceph comme stockage distribué partagé entre les nœuds, et en expliquer le placement des données
- **Configurer** la haute disponibilité (HA) d'une VM posée sur Ceph
- **Mettre** un nœud en maintenance sans interrompre les machines qu'il porte

> **Infrastructure disponible à ce stade :** vous disposez maintenant de 3 nœuds Proxmox (le vôtre depuis le début de la formation, plus 2 nouveaux). Vérifiez que les 3 nœuds se voient sur le réseau avant de commencer l'atelier cluster.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Cluster Proxmox, HA, Ceph et sauvegarde



La vue d'ensemble du chapitre : trois nœuds reliés par **Corosync**, qui forment le quorum. Sur chacun, les briques que le chapitre installe une à une : moniteurs Ceph, gestionnaire de haute disponibilité, OSD et disques RBD. Le **Proxmox Backup Server** reste **hors** du cluster : une sauvegarde ne doit pas tomber avec ce qu'elle protège.

Ce support est la propriété de Dawan France.

1. Constituer le cluster

Avant de cliquer sur **Create Cluster**, il faut savoir ce qu'on construit. Un cluster tient par trois choses : un vote à la majorité, le **quorum** ; un réseau sur lequel les nœuds se parlent ; et un second chemin si ce réseau tombe. Ce sont elles qui dictent la préparation des nœuds (§1.2) et les choix de la fenêtre de création (§1.3).

1.1 Avant d'assembler : ce qu'est un cluster, et ce qui le fait tenir

À quoi sert un cluster

Un cluster Proxmox VE est un groupe de nœuds administrés comme une seule entité. Il apporte trois choses qu'un serveur seul ne peut pas offrir :

Ce qu'il apporte	Concrètement	Dans ce chapitre
Une administration centralisée	Une seule interface web, une seule API, quel que soit le nœud auquel on se connecte. Les utilisateurs, les droits et les stockages se déclarent une fois.	§1.4
La migration	Déplacer une VM d'un nœud à l'autre, à froid ou à chaud, pour vider un nœud avant une intervention.	§2.1, §7.1
La haute disponibilité (HA)	Quand un nœud tombe, ses VM redémarrent seules sur un autre nœud.	§6

Ce qu'un cluster **n'est pas** :

- **un stockage partagé** : chaque nœud garde ses disques. Pour qu'une VM soit visible de partout, il faut une brique de plus, comme Ceph (§3), NFS ou iSCSI (chapitre 3) ;
- **un répartiteur de charge** : il ne répartit pas les requêtes des utilisateurs entre les VM.

Sa limite principale, c'est le réseau. Les nœuds doivent rester **proches** : même salle ou même site, avec un délai de quelques millisecondes au plus.

On ne fait pas un cluster entre deux villes. Pour administrer plusieurs clusters ou des sites éloignés, il existe Proxmox Datacenter Manager, une console web qui les réunit.

Deux composants sur chaque nœud

Ce support est la propriété de Dawan France.

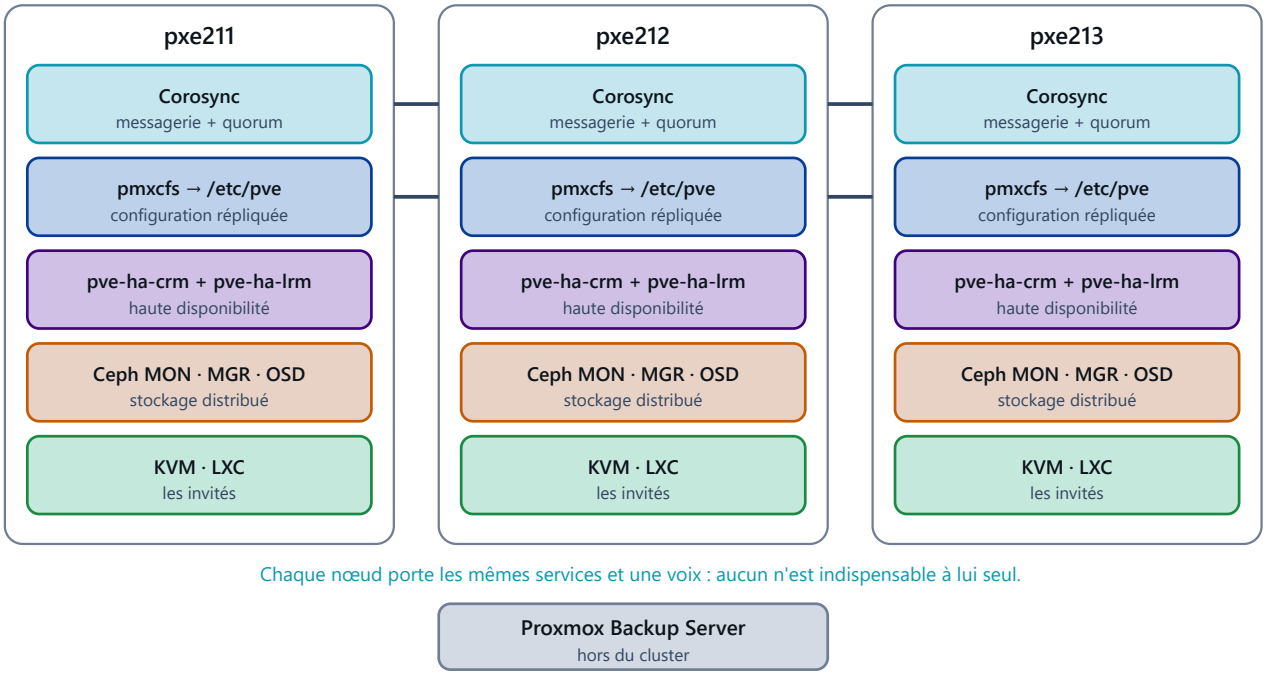
Corosync est le moteur de communication du cluster. Il envoie des messages de type « battement de cœur » (*heartbeat*) entre les nœuds, à intervalle régulier, pour vérifier que chaque membre est toujours vivant. Il compte aussi les nœuds présents : c'est lui qui calcule le quorum.

pmxcfs (Proxmox Cluster File System) est le système de fichiers monté sur `/etc/pve/`. Son contenu vit dans une petite base SQLite locale (`/var/lib/pve-cluster/config.db`), gardée en mémoire pour la vitesse. Chaque modification est répliquée en temps réel sur tous les nœuds du cluster, par l'intermédiaire de Corosync.

C'est là que vivent les configurations des VMs, des conteneurs, des stockages et des utilisateurs. Quand vous modifiez la configuration d'une VM depuis n'importe quel nœud, le changement se propage instantanément sur l'ensemble du cluster.

Dans un cluster, il n'y a pas de nœud « de stockage » ou de nœud « de gestion ». Chaque serveur exécute la même pile de services, et c'est leur accord qui fait le cluster.

Un cluster Proxmox : les mêmes services sur chaque nœud



Chaque nœud porte les mêmes services et une voix : aucun n'est indispensable à lui seul.

Le quorum : décider à la majorité

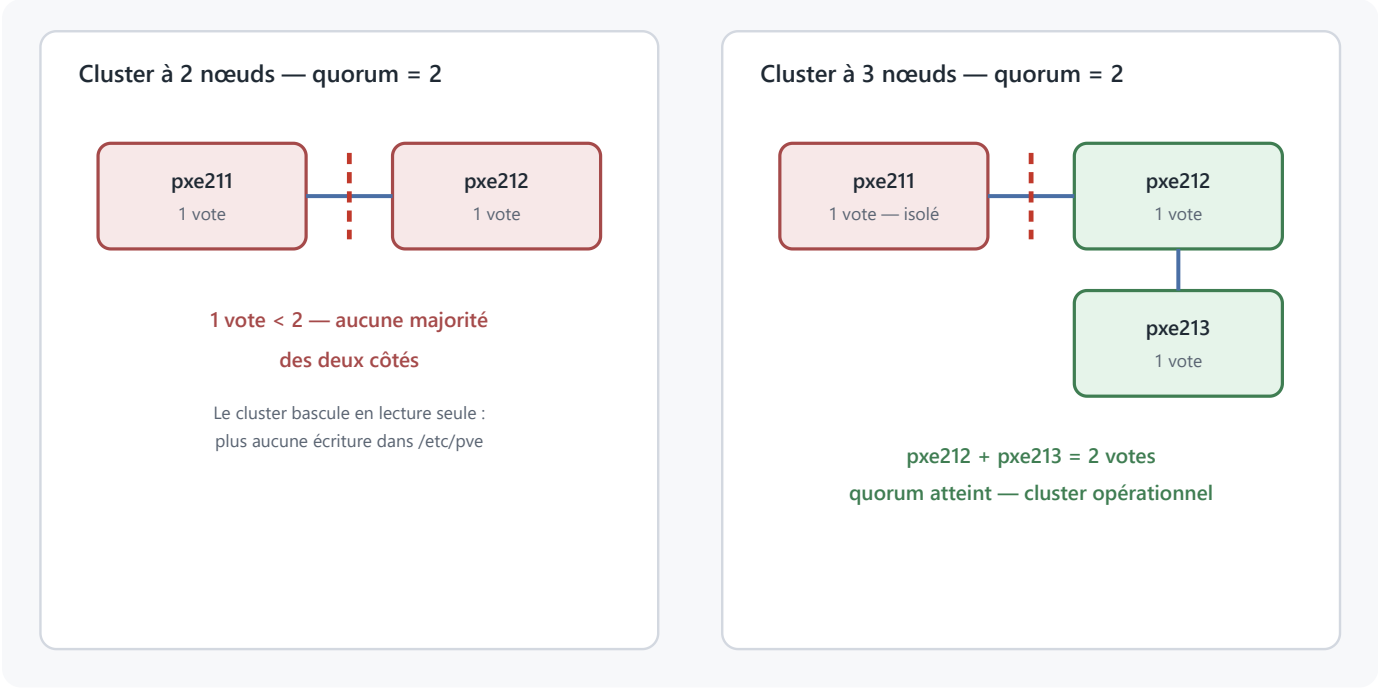
Le scénario le plus redouté dans un cluster est le **split-brain**. Le réseau se coupe en deux. Chaque moitié croit l'autre morte, et prend des décisions contradictoires sur les mêmes ressources.

Ce support est la propriété de Dawan France.

Par exemple, chacune démarre la même VM sur le même disque partagé. Deux systèmes écrivent alors en même temps sur le même disque : ses données sont corrompues, sans retour possible.

Le **quorum** est la protection contre ce scénario. Chaque nœud a **une voix**. Un groupe de nœuds n'a le droit d'agir que s'il réunit **plus de la moitié** des voix du cluster : la moitié, arrondie à l'entier inférieur, plus une.

Le groupe minoritaire, lui, se bloque de lui-même : `/etc/pve` passe en lecture seule, plus rien ne démarre. Mieux vaut un cluster figé et cohérent que deux moitiés qui divergent.



À deux nœuds, une coupure laisse deux groupes d'une voix chacun : aucun n'a la majorité, tout se fige. À trois, le groupe de deux voix continue, et le nœud isolé passe en lecture seule.

Combien de pannes un cluster supporte :

Nœuds	Voix nécessaires (quorum)	Pannes tolérées
2	2	aucune, sans QDevice
3	**2**	**1**
4	3	1 : aucun gain par rapport à 3
5	3	2

D'où la règle : **un nombre impair de nœuds**, et **trois au minimum** en production. Passer de 3 à 4 n'apporte aucune tolérance de plus.

Ce support est la propriété de Dawan France.

Le cas particulier des deux nœuds. Proxmox écrit alors dans `corosync.conf` l'option `two_node: 1`, qui permet de fonctionner avec une voix sur deux.

Elle s'accompagne de `wait_for_all: 1` : au démarrage, les deux nœuds doivent s'être vus au moins une fois avant que le quorum s'établisse. Le fichier du lab, à trois nœuds, n'en contient pas (§2.3).

Un cluster à deux nœuds reste fragile. La solution est un **QDevice** : un petit serveur extérieur, qui n'est pas un nœud Proxmox et n'héberge aucune VM, mais apporte une troisième voix.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Ajouter un QDevice externe pour un cluster à 2 nœuds (non pratiqué : le lab a trois nœuds)
pvecm qdevice setup <IP-QDevice>
```

Avec un QDevice, `pvecm status` compte trois voix attendues pour deux nœuds : si l'un tombe, l'autre garde le quorum (sa voix plus celle du QDevice).

Les termes, un par un :

Terme	Ce que c'est	Dans le laboratoire
Nœud	Un serveur du cluster ; une voix par défaut	`pxe2<N>1`, `pxe2<N>2`, `pxe2<N>3`
Corosync	Le service qui fait dialoguer les nœuds et compte les présents	Sur `vmbr1` (carte `ens19`)
Votes attendus (*Expected votes*)	Le total des voix du cluster	`3`
Quorum	Plus de la moitié des voix : la moitié, arrondie à l'entier inférieur, plus une	`2`
Quorate	Le groupe de nœuds voit assez de voix pour décider	`Quorate: Yes`
Split-brain	Deux moitiés du cluster qui démarrent la même VM et corrompent ses données	Empêché par le quorum
Lecture seule	Hors quorum, `/etc/pve` refuse toute écriture : ni création, ni démarrage	Le nœud isolé au §2.2 et à l'atelier 4

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
QDevice	Un petit serveur qui apporte une voix à un cluster de deux nœuds	Non utilisé (trois nœuds)

Un réseau dédié à Corosync

Corosync ne transporte pas les données des VM. Il transporte les **battements de cœur** : quelques octets, envoyés plusieurs fois par seconde par chaque nœud. C'est sur eux, et uniquement sur eux, que le cluster décide qui est vivant, donc qui a le quorum.

D'où une exigence contre-intuitive : **Corosync a besoin d'une latence faible et régulière, pas de débit.** Quelques kilo-octets par seconde lui suffisent.

Mais un battement en retard fait croire à la disparition d'un nœud. La documentation Proxmox demande **moins de 5 ms** entre les nœuds ; le lab mesure 0,5 ms (§1.2).

Or une sauvegarde ou une migration à chaud peut saturer le réseau d'administration pendant plusieurs minutes. Si Corosync partage ce réseau, ses battements prennent du retard. Le cluster croit un nœud mort, l'exclut, et la haute disponibilité redémarre ses VM ailleurs, alors qu'il fonctionnait.

C'est pourquoi on donne à Corosync **son propre réseau**. Dans cette formation, c'est le pont `vmbr1`, sur la carte `ens19`, préparé au chapitre 3. Le réseau d'administration, `vmbr0`, garde l'interface web, les migrations et les sauvegardes.

Pourquoi doubler le lien

Un réseau dédié peut tomber à son tour : un commutateur en panne, un câble arraché. Avec un seul lien, les nœuds ne se voient plus, et le cluster perd son quorum.

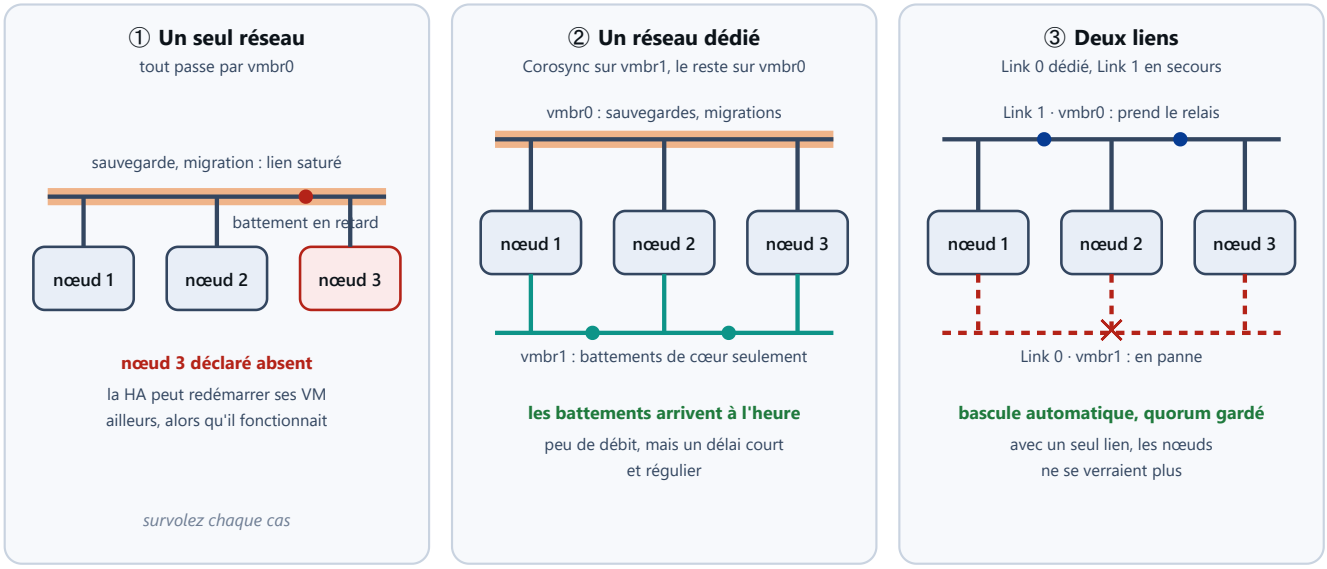
Depuis Proxmox VE 6, Corosync utilise le transport **knet** (Kronosnet). Il apporte deux choses :

- **plusieurs liens** entre les mêmes nœuds : `Link 0`, `Link 1`, jusqu'à huit. Un seul sert à la fois, celui de plus haute priorité ; si ses battements ne passent plus, Corosync **bascule seul** sur le suivant ;
- le **chiffrement** des échanges entre nœuds, activé par défaut (`secauth: on`).

Dans cette formation : **Link 0** sur `vmbr1`, le réseau dédié, prioritaire ; **Link 1** sur `vmbr0`, le réseau d'administration, en secours. Ce dernier n'est pas idéal en temps normal, mais il vaut mieux que rien quand le premier tombe.

Ce support est la propriété de Dawan France.

Corosync : un réseau à lui, et deux chemins



Trois cas, de gauche à droite :

- ① Un seul réseau : la sauvegarde sature `vmbr0`, le battement du nœud 3 arrive en retard, et le nœud est exclu à tort.
- ② Un réseau dédié : les battements circulent seuls sur `vmbr1`, la sauvegarde ne les gêne plus.
- ③ Deux liens : `vmbr1` tombe, Corosync bascule sur `vmbr0`, et le quorum tient.

Survolez chaque cas pour le détail.

Et une agrégation de cartes (bond) ? Doubler les cartes sous un même lien est une autre façon de se protéger. Si le réseau Corosync passe par un bond :

- le mode **active-backup** est sûr : une carte à la fois, l'autre en secours ;
- les modes qui répartissent la charge, dont **LACP (802.3ad)**, peuvent créer une connectivité asymétrique quand une seule carte du bond faiblit : certains nœuds se voient encore, d'autres non ;
- avec LACP en réglage par défaut, la bascule peut prendre **jusqu'à 90 secondes**, plus que le délai au bout duquel la haute disponibilité isole un nœud. Des nœuds sains peuvent alors être redémarrés de force. On règle `bond-lacp-rate fast` des deux côtés, sur le nœud et sur le commutateur.

Les deux liens de knet restent la protection la plus simple : c'est celle du lab.

Ce qu'il faut avant d'assembler

Ce support est la propriété de Dawan France.

Prérequis	Pourquoi	Où
Même version de Proxmox VE sur tous les nœuds	Corosync et pmxcfs doivent parler le même langage	§1.2, point 1
Un nom unique par nœud	Chaque nœud a son dossier dans `/etc/pve/nodes/<nom>`	`pxe2<N>1` à `pxe2<N>3`
Le réseau Corosync en place sur chaque nœud	Le Link 0 se choisit à la création : il doit exister avant	§1.2, point 2
Les nœuds se joignent, avec une latence faible	Moins de 5 ms entre tous les nœuds	§1.2, point 3
L'heure synchronisée	Proxmox VE installe `chrony` par défaut ; il doit rester actif	—
Les ports ouverts entre nœuds	UDP 5405 et suivants (Corosync, un port par lien), TCP 22 (SSH), TCP 8006 (API, pour la jonction)	§1.3, §2.3
Aucune VM sur les nœuds qui rejoignent	La jonction remplace leur configuration par celle du cluster	§1.3

À retenir avant d'assembler : un cluster sert à administrer, migrer et relancer ensemble ; il ne partage pas les disques. Il décide à la majorité, d'où **trois nœuds** au minimum. Corosync veut un **réseau à lui**, à faible latence, et **un second lien** pour survivre à la perte du premier.

Ce support est la propriété de Dawan France.

1. Constituer le cluster — Préparer les trois nœuds avant d'assembler

1.2 Préparer les trois nœuds avant d'assembler

Vous avez travaillé jusqu'ici sur un seul serveur. Les deux autres arrivent dans l'état où était le vôtre au début de la formation : dépôt réservé aux abonnés encore actif, aucune mise à jour appliquée, aucun second pont. **Trois choses sont à faire sur chacun d'eux avant de construire quoi que ce soit.**

1. Le dépôt et les mises à jour — sur les trois nœuds

Un cluster fait dialoguer en permanence les mêmes composants : Corosync, `pmxcfs`, les outils `pve*`. Si un nœud est resté en arrière, il peut refuser de rejoindre, ou pire, rejoindre et se comporter de travers sans que rien ne le signale. **On aligne les versions d'abord, on assemble ensuite.**

Sur chaque nœud, refaites exactement ce que vous avez fait au chapitre 1 :

1. le serveur → **Repositories** → sélectionner **chacune des deux** lignes `enterprise.proxmox.com` (dépôt PVE et dépôt Ceph) → **Disable**
2. **Add** → **No-Subscription** → valider
3. **Updates** → **Refresh**, puis **Upgrade**
4. redémarrer le nœud si le noyau a été mis à jour

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent en ligne de commande, dans Shell, sur chaque noeud
apt update                                          # la liste des paquets
apt full-upgrade -y                                 # la mise à jour

# La verification qui compte
pveversion
```

Résultat attendu : `pveversion` renvoie **la même version sur les trois nœuds**, et `apt update` ne renvoie plus d'erreur 401. Faites ce contrôle avant d'aller plus loin — c'est trente secondes qui

Ce support est la propriété de Dawan France.

vous en économisent trente.

2. Le pont `vmbr1` — sur les trois nœuds

Le réseau Corosync monté au chapitre précédent n'existe que sur votre serveur. Il doit exister à l'identique sur les deux autres, chacun avec **sa** propre adresse :

Nœud	IP Corosync
`pxe2<N>1`	`10.<site>.<N+10>.11/24` — déjà créé au chapitre 3
`pxe2<N>2`	`10.<site>.<N+10>.12/24`
`pxe2<N>3`	`10.<site>.<N+10>.13/24`

Sur chaque nœud : **Network** → **Create** → **Linux Bridge**.

Champ	Valeur
Name	`vmbr1`
IPv4/CIDR	l'adresse Corosync **de ce nœud-là**, en `/24`
Gateway (IPv4)	*(laisser vide)*
Bridge ports	`ens19`, la deuxième carte
Autostart	✓
Comment	`COROSYNC`

→ **Create**, puis **Apply Configuration**. Sans ce second clic, le pont est enregistré mais pas actif.

3. Vérifier que les trois se voient

C'est maintenant que ce test a un sens : les trois ponts existent, ils peuvent se parler.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis le premier nœud, vers les deux autres
ping -c 3 10.<site>.<N+10>.12
ping -c 3 10.<site>.<N+10>.13   # puis vers le troisième
```

Résultat attendu, sur le lab (de `pxe211` vers `10.59.11.12` puis `10.59.11.13`) : `rtt min/avg/max/mdev = 0.394/0.575/0.727/0.137 ms` et `0.393/0.535/0.679/0.116 ms` — moins d'une milliseconde, comme Corosync l'attend.

Ce support est la propriété de Dawan France.

Ne créez pas le cluster tant qu'un ping ne passe pas. Un cluster assemblé sur un réseau incomplet fonctionne en apparence, puis perd son quorum sans raison visible — et le diagnostic devient bien plus difficile une fois les trois nœuds liés. Reprenez le point 2 sur le nœud fautif : adresse correcte, préfixe `/24` identique partout, `ens19` bien présent dans **Bridge ports**, et **Apply Configuration** cliqué.

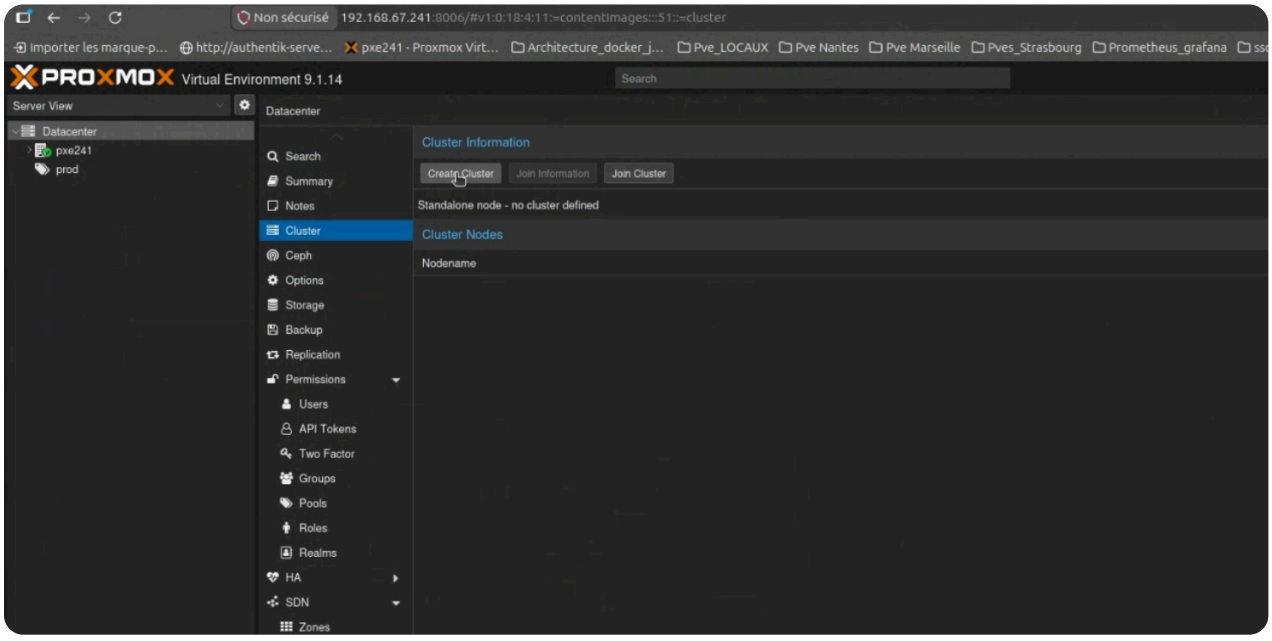
La liste avant d'assembler : même version sur les trois, `vmbr1` sur les trois, ping croisé qui passe, et aucune VM sur les nœuds qui vont rejoindre.

1.3 Créer le cluster et y joindre les nœuds

La création d'un cluster suit toujours le même ordre. On initialise le cluster sur le nœud qui sera le premier membre, puis on ajoute les autres un par un.

Il est crucial de réaliser cette étape **avant** de créer des VMs ou des conteneurs. Rejoindre un cluster efface la configuration locale du nœud entrant, pour adopter celle du cluster.

Créer le cluster — via l'interface graphique :



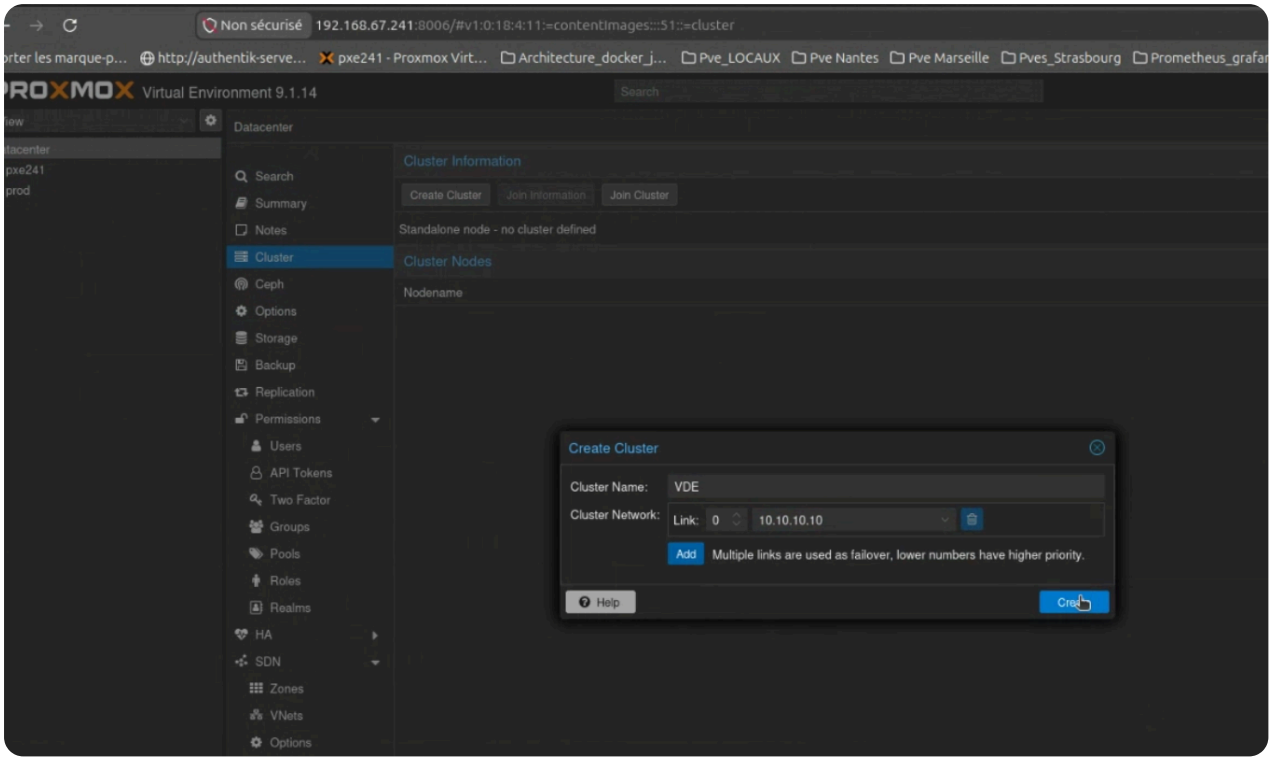
L'onglet `Datacenter → Cluster` d'un nœud encore seul. La ligne `Standalone node - no cluster defined` est l'état de départ de tous vos serveurs.

Trois boutons seulement : **Create Cluster** pour fonder, **Join Cluster** pour rejoindre, et **Join Information** — grisé, parce qu'un nœud sans cluster n'a aucune information de jonction à

Ce support est la propriété de Dawan France.

donner. C'est cet écran qui dit si un nœud a bien rejoint ou non.

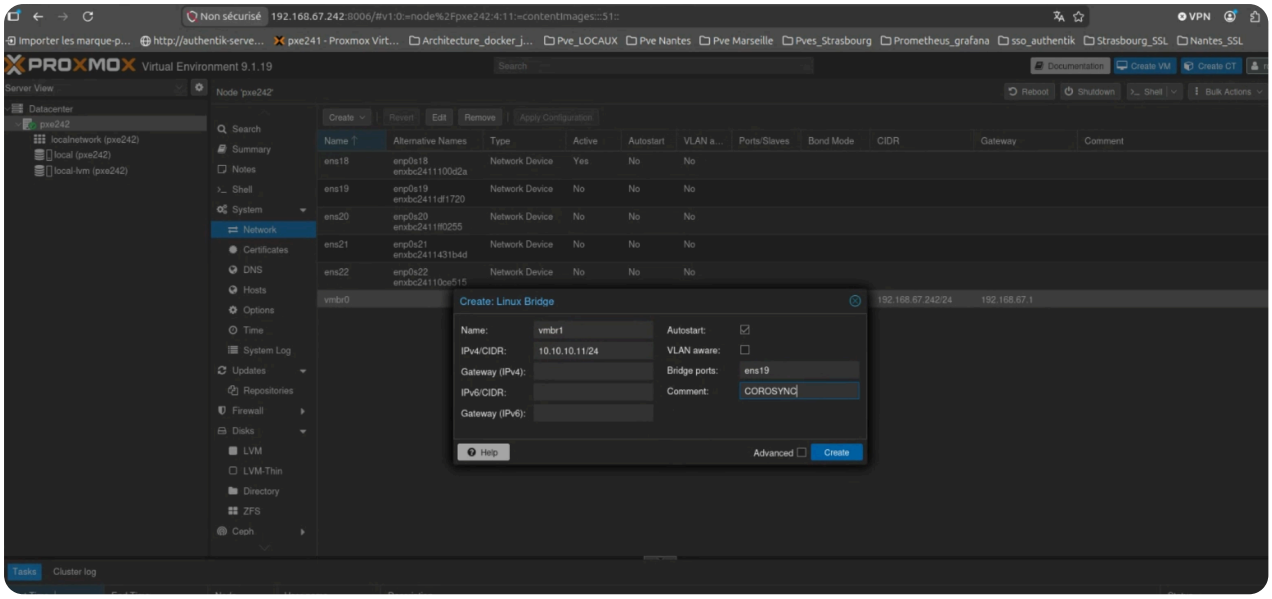
Datacenter → Cluster → Create Cluster



La fenêtre remplie sur `pxe211` (lab). Deux champs, et le second est celui qui compte.

- **Cluster Name** : le nom du cluster, purement identifiant. Ici `cluster-tp`.
- **Cluster Network — Link 0** : **l'adresse par laquelle les nœuds vont se parler**. Ici `10.59.11.11`, l'adresse de `vmbr1` sur le réseau dédié à Corosync, et non celle de l'interface d'administration.
- **Link 1**, ajouté par le bouton **Add** : `10.59.1.11`, l'adresse de `vmbr0`, en secours. La phrase sous les champs le dit : *« Multiple links are used as failover, lower numbers have higher priority »*. Si le premier réseau tombe, les nœuds continuent de se voir par le second et le cluster ne perd pas son quorum.

Ce support est la propriété de Dawan France.



La création du pont `vmbr1` sur le deuxième nœud, `pxe212` : rattaché à une carte réseau physique distincte (`ens19`), portant l'adresse Corosync du nœud (`10.59.11.12/24`) et commenté `COROSYNC` . C'est le réseau que le cluster utilise pour ses échanges, séparé de `vmbr0` qui porte l'administration et le trafic des machines.

C'est ici que se concrétisent les deux choix du §1.1 : le réseau dédié (`vmbr1`) devient le **Link 0**, et le réseau d'administration (`vmbr0`) le **Link 1**, en secours.

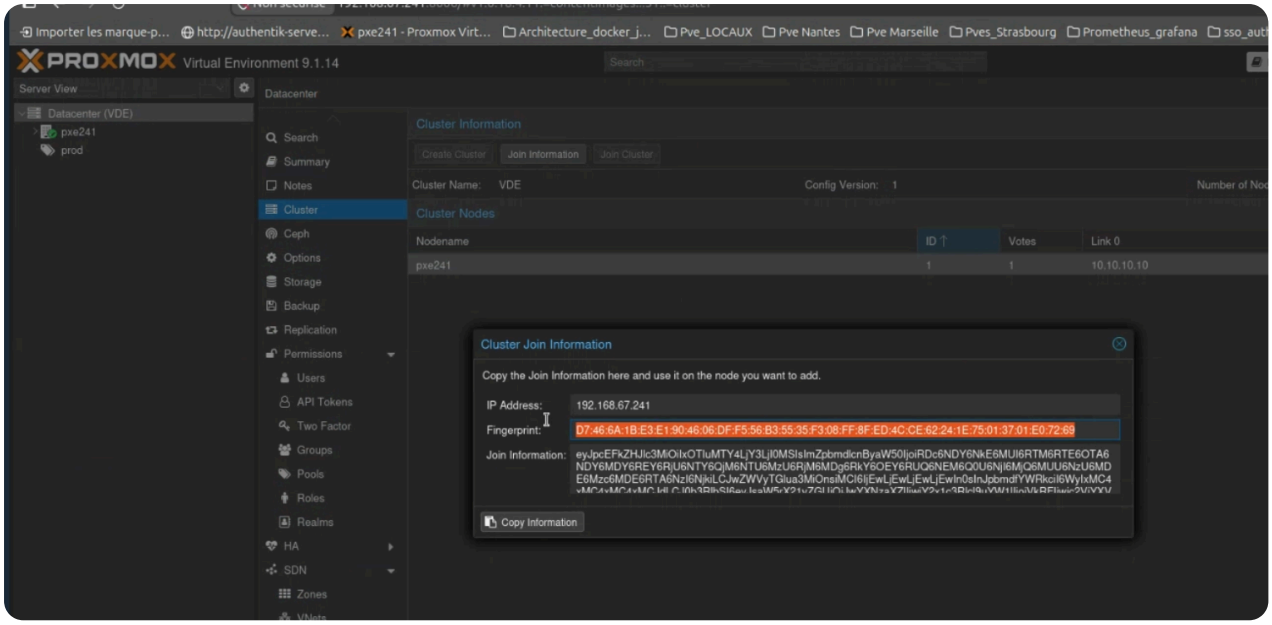
Champ	Valeur
Cluster Name	`cluster-tp`
Cluster Network — Link 0	**l'adresse Corosync de votre nœud** (celle de `vmbr1`), à choisir dans la liste déroulante

Choisissez bien l'adresse de `vmbr1` , pas celle de `vmbr0` . La liste déroulante propose les deux : c'est le seul moment où ce choix se fait, et il conditionne tout le reste du fonctionnement du cluster.

Attendez la coche verte dans la fenêtre de tâche. Ensuite : **Datacenter** → **Cluster** → **Join Information** → bouton **Copy Information** — ce bloc sera collé sur chaque nœud entrant.

Joindre un nœud — sur le nœud entrant :

Ce support est la propriété de Dawan France.

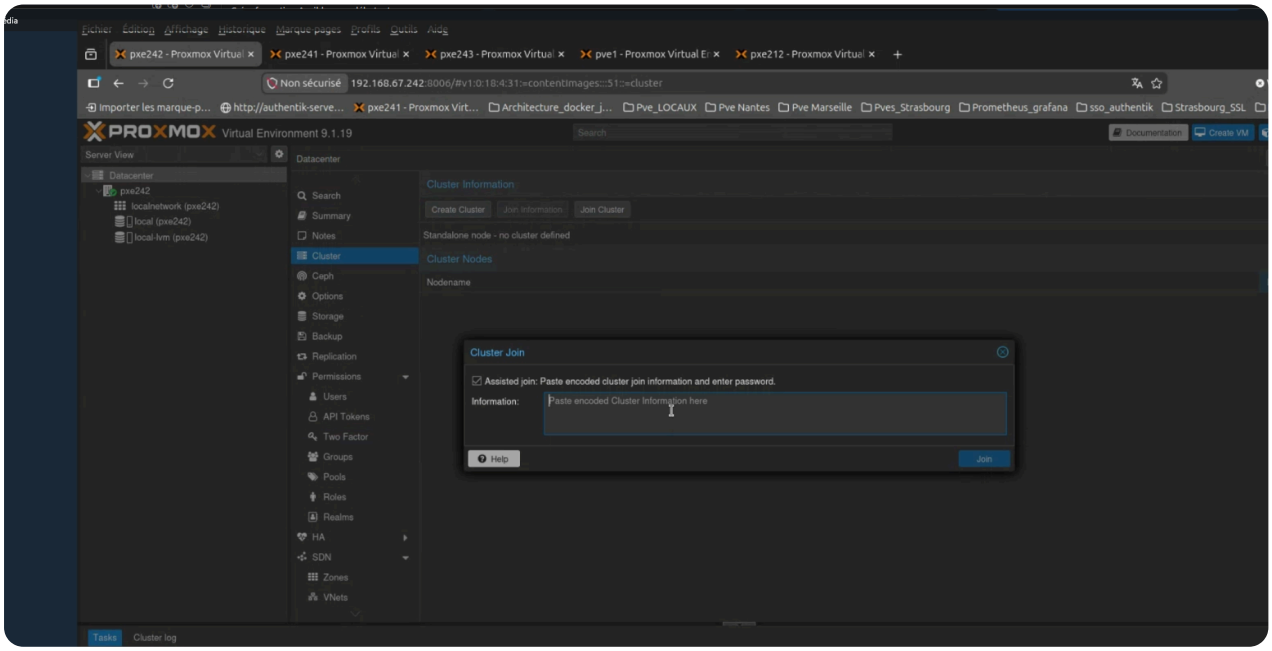


Ce que le nœud fondateur fournit aux autres, une fois le cluster créé :

- son **adresse IP**,
- son **empreinte de certificat**, qui garantit au nœud rejoignant qu'il parle bien au bon serveur,
- et le **bloc encodé** qui contient les deux, à copier d'un clic sur **Copy Information**.

Sur le lab, l'adresse IP de la fenêtre est `10.59.1.11` : celle de `vmbr0` du fondateur. C'est par elle que les autres nœuds le joignent pour adhérer ; Corosync, lui, passera par `vmbr1`.

Datacenter → Cluster → Join Cluster



Ce support est la propriété de Dawan France.

La fenêtre **Cluster Join**, **sur le nœud qui rejoint** (`pxe212`). La case **Assisted join** étant cochée, Proxmox extrait seul du bloc collé l'adresse du pair (`10.59.1.11`) et l'empreinte. Il vous demande le mot de passe `root` du nœud fondateur.

Les deux champs **Link** restent vides, en rouge. À droite, Proxmox rappelle l'adresse du pair sur chaque lien (`peer's link address: 10.59.11.11` , puis `10.59.1.11`) : choisissez en face l'adresse **de ce nœud** sur le même réseau.

Collez le contenu de `Join Information` , saisissez le mot de passe root du nœud fondateur, cliquez **Join**.

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# ===== Sur le nœud fondateur =====
# Nom : lettres, chiffres, tirets — 15 caractères au plus
# --link0 : l'adresse Corosync (vmbr1) ; --link1 : le secours (vmbr0)
pvecm create cluster-tp --link0 10.<site>.<N+10>.11 --link1 10.<site>.<N>.11
pvecm status   # vérifier : le cluster existe, quorum à 1 voix
```

Résultat attendu (`pxe211` , juste après la création, extrait) :

```
Cluster information
-------------------
Name:             cluster-tp
Config Version:   1
Transport:        knet
Secure auth:      on

Quorum information
------------------
Date:             Tue Oct  6 19:28:31 2026
Quorum provider:  corosync_votequorum
Nodes:            1
Node ID:          0x00000001
Ring ID:          1.5
Quorate:          Yes
```

Ce support est la propriété de Dawan France.

Un nœud seul, une voix : le cluster est déjà `Quorate`. La configuration est à sa version 1.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# ===== Sur chaque nœud qui rejoint, l'un après l'autre =====
# jonction au fondateur, avec ses deux liens : pvecm demande le mot de passe root
# du fondateur, puis fait accepter l'empreinte de son certificat (yes)
pvecm add 10.<site>.<N>.11 --link0 address=10.<site>.<N+10>.13 --link1 address=10.
<site>.<N>.13
```

Résultat attendu (jonction du troisième nœud, `pxe213`) ; le mot de passe tapé ne s'affiche pas :

```
Please enter superuser (root) password for '10.59.1.11':
Establishing API connection with host '10.59.1.11'
The authenticity of host '10.59.1.11' can't be established.
X509 SHA256 key fingerprint is …
Are you sure you want to continue connecting (yes/no)? yes
Login succeeded.
check cluster join API version
Request addition of this node
Join request OK, finishing setup locally
stopping pve-cluster service
backup old database to '/var/lib/pve-cluster/backup/config-1791307887.sql.gz'
waiting for quorum...OK
(re)generate node files
generate new node certificate
merge authorized SSH keys
generated new node certificate, restart pveproxy and pvedaemon services
successfully added node 'pxe213' to cluster.
```

Deux adresses, sur deux réseaux. `pvecm add <adresse>` vise le **fondateur**, sur son interface d'administration.

`--link0 address=` et `--link1 address=` sont celles du nœud **qui rejoint**. La jonction prend une dizaine de secondes : échange des clés SSH, nouveau certificat, attente du quorum.

> **Erreur fréquente :** `pvecm add` échoue avec une erreur de connexion. Depuis le nœud qui rejoint, vérifiez que le nœud fondateur répond sur l'API (port **8006**, par lequel passe la

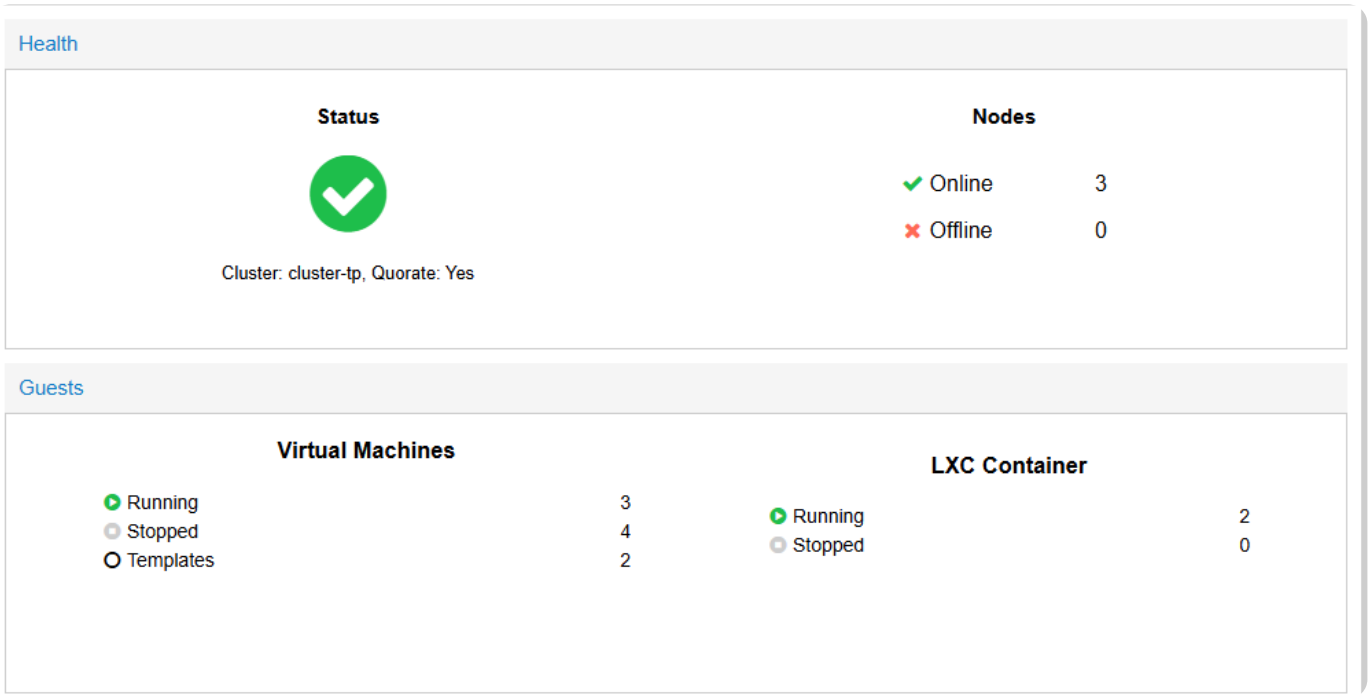
Ce support est la propriété de Dawan France.

> jonction) et en SSH (port 22).
>
> Les commandes : `curl -k https://10.<site>.<N>.11:8006` et `ssh root@10.<site>.<N>.11`. Si le pare-feu Proxmox a été activé sur le nœud fondateur, il doit laisser passer ces deux ports.

- 📖 Documentation Proxmox — Cluster Manager
- 📖 Créer un cluster Proxmox VE pas à pas — Technonagib

1.4 Vérifier l'état du cluster

Une fois les trois nœuds joints, la vérification s'effectue depuis n'importe quel nœud. **C'est en soi le premier contrôle** : un cluster sain donne la même réponse quel que soit le nœud interrogé. Deux nœuds qui décrivent différemment la composition du cluster est le symptôme d'une partition réseau.



(cluster `cluster-tp` du lab, Proxmox VE 9.2.21, juste après la jonction du troisième nœud). **Datacenter → Summary** résume l'état d'un coup d'œil. Le bloc *Status* indique le nom du cluster et `Quorate: Yes`.

Le bloc *Nodes* compte les nœuds en ligne (3) et hors ligne (0). Le bloc *Guests* additionne les machines de tous les nœuds. La même information, avec le détail des votes, est donnée par `pvecm status` ci-dessous.

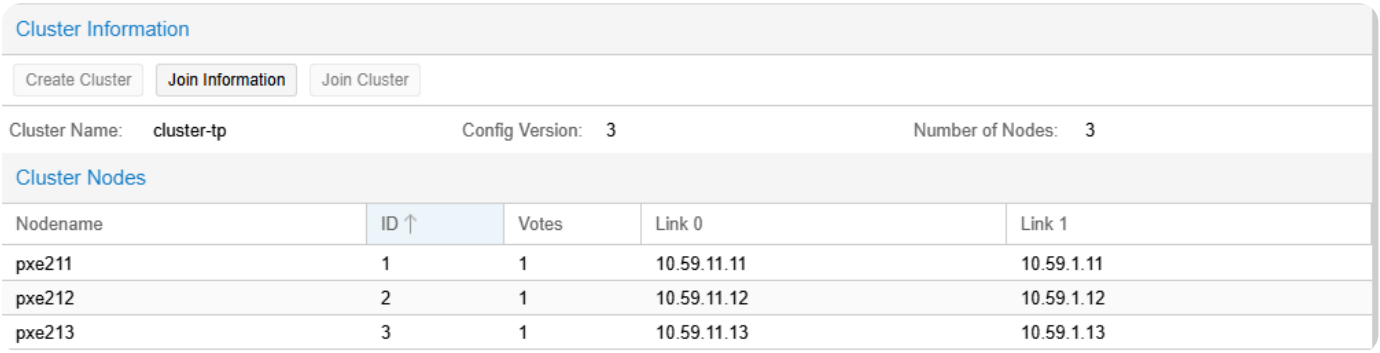
Ce support est la propriété de Dawan France.

Trois lignes de la sortie de `pvecm status` méritent d'être lues avant toutes les autres, et il faut savoir laquelle regarder en premier :

Ligne	Ce qu'elle dit	Ce qui doit vous alerter
`Quorate`	Le cluster a-t-il le quorum, donc le droit d'agir	`No` — le cluster est en lecture seule, plus aucune VM ne démarre
`Expected votes` / `Total votes`	Combien de voix il attend, combien il en compte	Un écart : un nœud ne répond plus
`Quorum`	Le seuil à atteindre	Sur 3 nœuds il vaut 2 : la majorité stricte

Ce sont les grandeurs du §1.1 : trois voix attendues, un quorum de deux. Tant que `Quorate` vaut `Yes`, le cluster a le droit d'agir.

L'interface web rend les mêmes informations, sous **Datacenter** → **Cluster**. Voici la maquette d'exemple de cette formation, un cluster de trois nœuds réellement en fonctionnement :



Le cluster `cluster-tp` du lab vu depuis l'interface (Proxmox VE 9.2.21) — trois nœuds, une voix chacun.

Trois colonnes de ce tableau méritent d'être commentées, parce qu'elles renvoient directement à ce qui précède et à ce qui suit.

La colonne **Votes** vaut 1 pour chaque nœud : trois voix au total, quorum à deux. C'est la traduction visuelle de la majorité stricte.

Le champ **Config Version** compte les versions successives de la configuration du cluster. Il vaut 3 ici : la création, puis deux jonctions. Chaque adhésion produit une nouvelle version de la configuration.

Et surtout, les colonnes **Link 0** et **Link 1** montrent deux adresses par nœud. `10.59.11.1x` sur le réseau dédié du cluster (`vmbr1`, Link 0, prioritaire). `10.59.1.1x` sur le réseau de gestion (`vmbr0`,

Ce support est la propriété de Dawan France.

Link 1, secours).

C'est exactement la **redondance de liens Corosync** présentée au §1.1. Sur vos nœuds, remplacez `59` par votre site et `1` par votre numéro.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Vue d'ensemble du cluster (quorum, membres, transport)
pvecm status

# Liste des nœuds avec leur ID et leur état
pvecm nodes

# Inspecter la configuration Corosync générée automatiquement
cat /etc/pve/corosync.conf
```

Résultat sur `pxe211`, trois nœuds joints (extrait de `pvecm status`) :

```
Cluster information
-------------------
Name:             cluster-tp
Config Version:   3

Quorum information
------------------
Nodes:            3
Quorate:          Yes

Membership information
----------------------
    Nodeid      Votes Name
0x00000001          1 10.59.11.11 (local)
0x00000002          1 10.59.11.12
0x00000003          1 10.59.11.13
```

La section Membership affiche l'adresse du Link 0, pas le nom du nœud : c'est la preuve que Corosync passe par `vmbr1`. Pour les noms, `pvecm nodes`.

Ce support est la propriété de Dawan France.

À retenir : un cluster s'assemble sur des nœuds **prêts** : même version, réseau de cluster (`vmbr1`) en place, ping croisé qui passe, aucune VM sur les nœuds qui rejoignent. On le crée sur un premier nœud (`pvecm create`), puis chaque autre le rejoint (`pvecm add`). `pvecm status` doit donner la même réponse sur tous les nœuds.

Ce support est la propriété de Dawan France.

2. Faire vivre le cluster : migration, quorum, Corosync

Le cluster est formé et répond d'une seule voix. Cette section en montre la vie courante : déplacer des machines d'un nœud à l'autre, voir le quorum à l'œuvre sur un nœud isolé. Elle montre aussi comment lire la configuration de Corosync et de pmxcfs, et diagnostiquer une panne.

2.1 Migration de VMs entre nœuds

La migration est l'une des raisons principales de construire un cluster. Elle permet de déplacer une VM d'un nœud à l'autre, pour des raisons de maintenance, d'équilibrage de charge, ou en cas de panne imminente détectée.

La **migration à chaud** (live migration) déplace la VM sans l'arrêter. La mémoire vive est transférée en temps réel entre les deux nœuds, et la VM continue de fonctionner pendant toute l'opération.

Sur un stockage partagé (comme Ceph), seule la mémoire voyage. Sur un stockage local, c'est possible aussi avec l'option `--with-local-disks`, mais Proxmox copie alors les disques en même temps : c'est beaucoup plus long.

La **migration à froid** s'applique à une VM **arrêtée** : Proxmox déplace sa configuration et, si besoin, copie ses disques sur le nœud cible. C'est le comportement par défaut de `qm migrate` quand on ne demande pas `--online`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Migration à chaud — la VM 300 reste active pendant le déplacement vers pxe212
# Prerequis : un stockage partage — Ceph dans nos ateliers (atelier 5)
qm migrate 300 pxe212 --online

# Migration à froid — la VM doit être arrêtée d'abord (la VM 120 de l'atelier 2)
qm shutdown 120                                    # l'arrêter
qm migrate 120 pxe212                              # puis la migrer

# Les conteneurs n'ont PAS de migration à chaud : --restart les arrête,
```

Ce support est la propriété de Dawan France.

```
# les déplace et les redémarre, avec quelques secondes de coupure
pct migrate 200 pxe213 --restart
```

Sortie réelle d'une migration à chaud. La commande `qm migrate 951 pxe211 --online` est lancée depuis `pxe212`, sur une VM de test de 512 Mo sans disque, avec le réseau de migration par défaut :

```
2026-10-07 18:13:43 conntrack state migration not supported or disabled, active
connections might get dropped
2026-10-07 18:13:44 starting migration of VM 951 to node 'pxe211' (10.59.1.11)
2026-10-07 18:13:44 starting VM 951 on remote node 'pxe211'
2026-10-07 18:13:45 start remote tunnel
2026-10-07 18:13:45 ssh tunnel ver 1
2026-10-07 18:13:45 starting online/live migration on unix:/run/qemu-
server/951.migrate
2026-10-07 18:13:45 set migration capabilities
2026-10-07 18:13:45 migration downtime limit: 100 ms
2026-10-07 18:13:45 migration cachesize: 64.0 MiB
2026-10-07 18:13:45 set migration parameters
2026-10-07 18:13:45 start migrate command to unix:/run/qemu-server/951.migrate
2026-10-07 18:13:46 average migration speed: 528.8 MiB/s - downtime 2 ms
2026-10-07 18:13:46 migration completed, transferred 3.7 MiB VM-state
2026-10-07 18:13:46 migration status: completed
2026-10-07 18:13:48 migration finished successfully (duration 00:00:05)
```

La séquence se lit ligne à ligne. La première ligne prévient que l'état des connexions suivies par le nœud (*conntrack*) n'est pas transféré : des connexions actives de la VM peuvent être coupées.

Puis une VM « coquille » démarre sur le nœud cible et un tunnel s'ouvre entre les deux nœuds. La mémoire de la VM d'origine y est recopiée **pendant qu'elle continue de tourner**.

Les pages modifiées pendant la copie sont renvoyées, jusqu'à ce que le reste à transférer tienne dans la `downtime limit` — 100 ms. La VM est alors figée ce court instant (ici `downtime 2 ms`), les dernières pages passent, et elle repart sur le nœud cible.

Seules les pages de mémoire réellement utilisées voyagent. Cette VM de test, qui n'a démarré aucun système, n'en a transféré que 3,7 Mio.

Ce support est la propriété de Dawan France.

Une VM Debian en service en transfère autant que sa mémoire occupée. Aucune ligne ne parle de disque : sur un stockage partagé, les deux nœuds voient déjà le même.

Et l'adresse entre parenthèses est celle **par laquelle passe la migration**. Ce n'est pas celle du réseau Corosync.

Par quel réseau passe la migration ? Par défaut, par l'adresse principale du nœud cible, celle de `vmbr0` : le réseau d'administration.

Sur le lab, pendant l'atelier, avant tout réglage (atelier 2), le journal indiquait `starting migration of VM 120 to node 'pxe212' (10.59.1.12)`, l'adresse `vmbr0` du lab (`10.<site>.<N>.12` chez vous). Le réseau Corosync (`10.59.11.x`) n'était pas utilisé.

On donne à la migration son propre réseau dans **Datacenter → Options → Migration Settings**. Sur le lab (atelier 5), avec `migration: network=10.59.41.11/24,type=secure`, le journal a changé dès la migration suivante.

Dans le Shell de `pxe211` :

```
# Migration à chaud de la VM 300 vers pxe212 (atelier 5)
qm migrate 300 pxe212 --online
```

```
use dedicated network address for sending migration traffic (10.59.41.12)
starting migration of VM 300 to node 'pxe212' (10.59.41.12)
average migration speed: 2.0 GiB/s - downtime 16 ms
migration completed, transferred 552.6 MiB VM-state
```

Cette VM Debian en service a transféré 553 Mio de mémoire, contre 3,1 Mio pour la VM de test sans système.

La bascule a duré 16 ms. Aucun disque n'a voyagé : il est sur Ceph.

Et un disque local ? Une VM dont un disque est sur `local-lvm` refuse la migration à chaud sans option.

Sur le lab : `can't live migrate attached local disks without with-local-disks option`. Avec `--with-local-disks`, le disque est recopié pendant la migration, ce qui la rallonge d'autant.

Ce support est la propriété de Dawan France.

Pourquoi pas de migration à chaud pour un conteneur ? Une VM, c'est un processus QEMU dont on sait copier la mémoire.

Un conteneur, ce sont des dizaines de processus du noyau de l'hôte, avec leurs fichiers ouverts et leurs connexions : rien qu'on puisse transporter tel quel vers un autre noyau. `pct migrate --online` répond donc par une erreur, et `--restart` est la seule migration « avec le conteneur en marche ».

La migration est également accessible depuis l'interface graphique : clic droit sur la VM → Migrate. L'interface propose automatiquement les nœuds disponibles dans le cluster et indique si une migration à chaud est possible selon le stockage utilisé.

2.2 Le quorum à l'épreuve : un nœud isolé

Le §1.1 a posé la règle : sans majorité, un nœud se bloque de lui-même. Voici ce qu'on observe sur le cluster monté.

Ce que voit un nœud qui a perdu le quorum. Sur le lab, en arrêtant Corosync sur `pxe213` (`systemctl stop corosync`), puis en le relançant :

Dans le Shell de `pxe213` :

```
systemctl stop corosync                # couper Corosync sur ce nœud
pvecm status                           # le nœud ne parle plus à Corosync
touch /etc/pve/essai-lecture-seule     # écrire dans /etc/pve
systemctl start corosync               # retour à la normale
```

Résultat attendu :

```
Cluster information
-------------------
Name:             cluster-tp
Config Version:   3
Transport:        knet
Secure auth:      on

Cannot initialize CMAP service
touch: cannot touch '/etc/pve/essai-lecture-seule': Permission denied
```

Ce support est la propriété de Dawan France.

Pendant ce temps, sur `pxe211`, `pvecm status` affichait `Nodes: 2`, `Total votes: 2`, `Quorate: Yes` : la partition de deux nœuds continue. Le nœud isolé, lui, passe `/etc/pve` en **lecture seule**.

Après `systemctl start corosync`, il retrouve les trois voix et l'écriture. C'est le cas à trois nœuds du schéma du quorum (§1.1) : la majorité continue, le nœud isolé se fige.

Où voir le quorum sur votre cluster :

Notion	Où la voir
L'état d'ensemble	**Datacenter** → **Summary** : bloc *Status*, `Quorate: Yes`, et le nombre de nœuds en ligne
Votes, quorum, partition	`pvecm status` : lignes `Expected votes`, `Total votes`, `Quorum`, `Quorate`
Les membres et leur voix	`pvecm nodes` ; **Datacenter** → **Cluster**
La configuration de Corosync	`/etc/pve/corosync.conf` (lecture seule ici : il ne s'édite pas à la main sur un nœud isolé)

2.3 Corosync et pmxcfs sur le cluster monté

Le §1.1 a expliqué pourquoi Corosync a un réseau à lui et deux liens. Sur le cluster assemblé, on lit maintenant ce que `pvecm` a écrit : la configuration de Corosync, l'état de ses deux liens, ses ports, et le système de fichiers qu'il réplique.

> `corosync.conf` **ne s'édite jamais directement sur un nœud isolé.** Le fichier vit dans `/etc/pve/`, un système de fichiers répliqué : toute modification se propage à l'ensemble du cluster. Et surtout, **le champ** `config_version` **doit être incrémenté à chaque changement** — c'est lui qui indique aux autres nœuds qu'une nouvelle version existe.
>
> L'oublier laisse le cluster avec deux configurations divergentes, chacune se croyant à jour. Une erreur ici peut faire perdre le quorum à tout le cluster d'un coup ; la manipulation se fait toujours sur un cluster sain, jamais en pleine panne.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
cat /etc/pve/corosync.conf   # la configuration Corosync écrite par pvecm
```

Ce support est la propriété de Dawan France.

Résultat attendu, sur `pxe211` (cluster de 3 nœuds, deux liens) ; extrait, les blocs `logging` et `quorum` et les lignes secondaires de `totem` sont omis :

```
nodelist {
  node {
    name: pxe211
    nodeid: 1
    quorum_votes: 1
    ring0_addr: 10.59.11.11
    ring1_addr: 10.59.1.11
  }
  (pxe212 et pxe213 : le même bloc, nodeid 2 et 3, adresses en .12 et .13)
}

totem {
  cluster_name: cluster-tp
  config_version: 3
  link_mode: passive
  secauth: on
}
```

Chaque nœud a une adresse par lien : `ring0_addr` pour le Link 0 (`vmbr1`), `ring1_addr` pour le Link 1 (`vmbr0`). `link_mode: passive` : un seul lien sert à la fois, le plus prioritaire.

`config_version: 3` : la création, puis deux jonctions. C'est ce numéro qu'il faut incrémenter à chaque modification manuelle.

Dans le Shell du nœud :

```
# État des liens Corosync vus par ce nœud
corosync-cfgtool -s
```

Résultat attendu sur `pxe211` :

```
Local node ID 1, transport knet
LINK ID 0 udp
    addr    = 10.59.11.11
    status:
        nodeid:          1: localhost
```

Ce support est la propriété de Dawan France.

```
        nodeid:          2: connected
        nodeid:          3: connected
LINK ID 1 udp
    addr    = 10.59.1.11
    status:
        nodeid:          1: localhost
        nodeid:          2: connected
        nodeid:          3: connected
```

Un bloc par lien, et pour chacun l'état vers chaque autre nœud. `connected` partout : les deux chemins fonctionnent.

Les ports, et ce qui doit passer entre les nœuds. Corosync écoute en UDP, un port par lien.

Dans le Shell du nœud :

```
ss -ulnp | grep corosync   # les ports UDP ouverts par Corosync
```

Résultat attendu sur `pxe211` :

```
UNCONN 0      0         10.59.11.11:5405      0.0.0.0:*    users:
(("corosync",pid=4355,fd=33))
UNCONN 0      0          10.59.1.11:5406      0.0.0.0:*    users:
(("corosync",pid=4355,fd=34))
```

`5405` sur l'adresse du Link 0, `5406` sur celle du Link 1. Entre les nœuds doivent donc passer l'UDP de Corosync (5405 et suivants, un port par lien). Il faut aussi le **SSH (TCP 22)**, que Proxmox utilise pour certaines opérations entre nœuds, comme la migration.

Sous le capot : pmxcfs. `/etc/pve` n'est pas un dossier ordinaire : c'est un système de fichiers en espace utilisateur (FUSE). Sa copie de référence est une petite base SQLite, `/var/lib/pve-cluster/config.db`, gardée en mémoire.

Dans le Shell du nœud :

```
mount | grep /etc/pve                                  # un montage FUSE
ls -la /var/lib/pve-cluster/config.db                  # la base qui le porte
ln -s /etc/pve/storage.cfg /etc/pve/lien-essai         # un lien symbolique :
refusé
```

Ce support est la propriété de Dawan France.

```
touch /etc/pve/essai-chmod                                  # un fichier d'essai
chmod 600 /etc/pve/essai-chmod                              # un changement de droits
: refusé
rm -f /etc/pve/essai-chmod                                  # retirer le fichier
d'essai
```

Résultat attendu sur `pxe211` :

```
/dev/fuse on /etc/pve type fuse
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other)
-rw------- 1 root root 69632 Oct  6 19:32 /var/lib/pve-cluster/config.db
ln: failed to create symbolic link '/etc/pve/lien-essai': Function not implemented
chmod: changing permissions of '/etc/pve/essai-chmod': Operation not permitted
```

Ces restrictions sont voulues : les droits se règlent par les chemins, pas par des attributs de fichier. Elles garantissent aussi qu'un même numéro de VM ne peut pas être créé deux fois dans le cluster.

Chaque écriture attend tous les nœuds. Une modification de configuration est diffusée par Corosync à tous les nœuds, y compris l'émetteur. Chacun l'écrit sur son disque (`fsync`) avant qu'elle soit confirmée.

Un seul nœud au disque système lent ralentit donc tout le cluster. `pveperf` mesure ce débit.

Dans le Shell du nœud :

```
pveperf /var/lib/pve-cluster   # la ligne FSYNCS/SECOND
```

Résultat attendu sur `pxe211` (disque virtuel du lab) : `FSYNCS/SECOND: 511.54`. Quelques dizaines de fsync/s désignent un support trop lent pour un nœud de cluster.

2.4 Diagnostiquer le cluster

Premier réflexe : `pvecm status` , puis les journaux. Mais `Quorate: Yes` ne veut pas dire « cluster en bonne santé » : c'est une mesure binaire, assez de voix ou pas. Un cluster peut être quorate et inutilisable si un service au-dessus est bloqué.

Ce support est la propriété de Dawan France.

Service	Son rôle	Ses journaux
`corosync`	Communication, quorum, liens	`journalctl -u corosync`
`pve-cluster`	`pmxcfs`, le montage `/etc/pve`	`journalctl -u pve-cluster`
`pvestatd`	Les statistiques des VM et des stockages	`journalctl -u pvestatd`
`pve-ha-lrm`, `pve-ha-crm`	La haute disponibilité (§6)	`journalctl -u pve-ha-lrm -u pve-ha-crm`
`pveproxy`, `pvedaemon`	L'interface web et l'API	`journalctl -u pveproxy -u pvedaemon`

Ajoutez `-f` pour suivre en direct, par exemple pendant une jonction.

Dans le Shell du nœud :

```
journalctl -u corosync --no-pager -n 12   # les 12 dernières lignes
```

Résultat sur `pxe211`, au moment où `pxe213` rejoint (extrait) :

```
corosync[4355]:   [QUORUM] Sync joined[1]: 3
corosync[4355]:   [TOTEM ] A new membership (1.d) was formed. Members joined: 3
corosync[4355]:   [QUORUM] Members[3]: 1 2 3
corosync[4355]:   [MAIN  ] Completed service synchronization, ready to provide
service.
corosync[4355]:   [KNET  ] rx: host: 3 link: 1 is up
corosync[4355]:   [KNET  ] pmtud: Global data MTU changed to: 1397
```

Le nœud 3 entre dans l'appartenance (`Members[3]: 1 2 3`), puis ses liens sont vus un par un.

Les pannes courantes :

`pvecm status` affiche `Cannot initialize CMAP service`. Corosync ne tourne pas sur ce nœud (vu au §2.2). Remède : `systemctl status corosync`, `journalctl -u corosync -n 100` ; vérifier `/etc/pve/corosync.conf`.

`/etc/pve/...` : `Permission denied`. Le nœud a perdu le quorum : `/etc/pve` est en lecture seule. Remède : Rétablir le réseau ou Corosync ; ne rien forcer.

Ce support est la propriété de Dawan France.

`'/etc/pve/nodes/<nœud>/pve-ssl.pem' does not exist`. Le nœud n'a pas pu écrire son certificat, faute de quorum. Remède : D'abord le quorum, puis `pvecm updatecerts` (`-f` pour forcer la régénération) et `systemctl restart pveproxy`.

Sur le lab : sur un nœud dont le certificat existe, `pvecm updatecerts` ne dit rien et rend la main (code `0`).

Les pièges :

- **Forcer `pvecm expected` en routine.** La commande abaisse le nombre de voix attendues et contourne la protection contre le *split-brain*. Elle n'a sa place qu'en urgence documentée.
- **Un cluster à deux nœuds sans QDevice** en production : perdre un nœud fige l'autre.
- **Un bond mal réglé** sous Corosync (§1.1).
- **Oublier de sauvegarder `/var/lib/pve-cluster/config.db`** : c'est la base de `/etc/pve`. Une copie régulière permet de la reconstruire.

> Un cluster de **deux** nœuds perd le lien qui les relie. Chacun fonctionne encore. Pourquoi ne peut-on plus créer de VM sur aucun des deux ?
> ▼ Réponse
> **Aucun n'a le quorum.** Il faut plus de la moitié des voix, soit 2 sur 2 : chaque nœud isolé n'en a qu'une. Pour éviter le *split-brain*, `/etc/pve` passe alors en lecture seule des deux côtés, et toute modification de configuration est refusée. Avec trois nœuds, la partie qui en garde deux conserve le quorum : c'est la raison du troisième nœud (ou d'un QDevice).

2.5 Sortir un nœud du cluster

Le cas le plus fréquent en salle : sur le nœud 3, on clique **Create Cluster** au lieu de **Join Cluster**. Le nœud fonde alors un cluster à lui seul : son `pvecm status` a l'air sain (`Quorate: Yes`), mais il affiche `Nodes: 1` et `Expected votes: 1`. Dans cet état, `pvecm add` refuse de rejoindre le vrai cluster.

La réparation suit une procédure officielle de Proxmox, *Separate a Node Without Reinstalling*. Sur le nœud, on arrête le système de fichiers du cluster et Corosync, on efface leur configuration, puis on le joint de nouveau au vrai cluster. Elle demande un nœud sans VM ni conteneur, et elle se fait avec le formateur.

- 📖 Documentation Proxmox — Separate a Node Without Reinstalling

Ce support est la propriété de Dawan France.

> **Ne « réparez » jamais un nœud isolé par `pvecm expected 1`** : il retrouverait le droit d'écrire seul dans `/etc/pve`, au risque du *split-brain* (§2.2).

À retenir : la **migration à chaud** déplace une VM sans l'arrêter, sur un réseau qu'on peut lui dédier. Le **quorum** (plus de la moitié des voix) empêche le *split-brain* : sans lui, `/etc/pve` passe en lecture seule.

Corosync (transport knet) porte ce dialogue ; il supporte plusieurs liens et ne s'édite pas à la main sur un nœud isolé.

> **▶ À pratiquer maintenant — Préparation à la mise en cluster, puis Atelier 1 et Atelier 2, puis Atelier 3 et Atelier 4**
>
> Nettoyer le premier nœud et mettre les deux nouveaux à jour. Puis assembler les trois nœuds en cluster, et déplacer une machine de l'un à l'autre. D'abord à froid, pour voir ce que la migration transporte réellement, puis allumée, avec ses disques locaux, et enfin un conteneur.
>
> Pour finir, priver un nœud de ses deux voisins et observer ce que le quorum lui interdit.

Ce support est la propriété de Dawan France.

3. Ceph : architecture et déploiement

Imaginez un système de stockage sans point de défaillance unique. Il distribue automatiquement les données sur plusieurs disques et plusieurs serveurs.

Il se répare tout seul en cas de panne, et s'intègre directement dans votre hyperviseur. C'est exactement ce que fait Ceph.

Ceph est né de la thèse de doctorat de Sage Weil à l'université de Californie à Santa Cruz (UC : *University of California*) en 2007. Son idée centrale : un algorithme de placement de données appelé CRUSH (Controlled Replication Under Scalable Hashing).

N'importe quel client calcule lui-même où se trouvent ses données, sans passer par un serveur de métadonnées centralisé. Cette approche élimine les goulots d'étranglement et les points uniques de défaillance.

Proxmox VE intègre Ceph depuis la version 3.2 (2014), et le pilote entièrement depuis son interface depuis la version 5. En quelques clics dans l'interface (ou quelques commandes), vous transformez les disques locaux de vos nœuds en un stockage distribué partagé, accessible par l'ensemble du cluster.

- 📽️ Ceph, le stockage du futur — conférence du Capitole du Libre : la conférence présentée au chapitre 3 (§5.3), à revoir pour les principes de distribution, de réplication et l'algorithme CRUSH.

3.1 Architecture Ceph — les composants

Avant de déployer Ceph, il est essentiel de comprendre le rôle de chaque composant, car chacun a une responsabilité précise dans le fonctionnement du cluster :

MON — Monitor : Le MON maintient la "carte" du cluster Ceph — il sait où sont tous les OSDs, leur état de santé, et la topologie de réplication. Sans MON, rien ne fonctionne.

Comme pour Corosync, le quorum des MONs est indispensable : il faut toujours un nombre impair de MONs, avec un minimum de 3 en production. Dans notre atelier, 1 MON par nœud, soit 3 au total.

MGR — Manager : Le MGR collecte les statistiques du cluster (utilisation, IOPS, latence) et expose le dashboard web de Ceph. Les entrées/sorties continuent sans lui pendant un court

Dawan Ce support est la propriété de Dawan France.

moment. Mais Ceph le considère comme un service requis depuis Luminous.

Sans manager : plus de statistiques de placement, plus d'autoscaler, plus de balancer, et `ceph status` se dégrade. On en déploie donc **un par nœud**, comme les moniteurs.

MON et MGR ne fonctionnent pourtant pas de la même façon. Les moniteurs travaillent **tous en même temps** et votent : c'est leur majorité qui fait autorité sur la carte du cluster, exactement la logique de quorum vue pour Corosync.

Les managers fonctionnent en **actif/veille** : un seul est actif à un instant donné, les autres attendent et prennent la main automatiquement s'il disparaît. `ceph -s` le montre directement.

Dans le Shell du nœud :

```
# État du cluster Ceph (extrait : lignes mon et mgr)
ceph -s
```

```
    mon: 3 daemons, quorum pve1,pve2,pve3
    mgr: pve1(active, since 4m), standbys: pve2, pve3
```

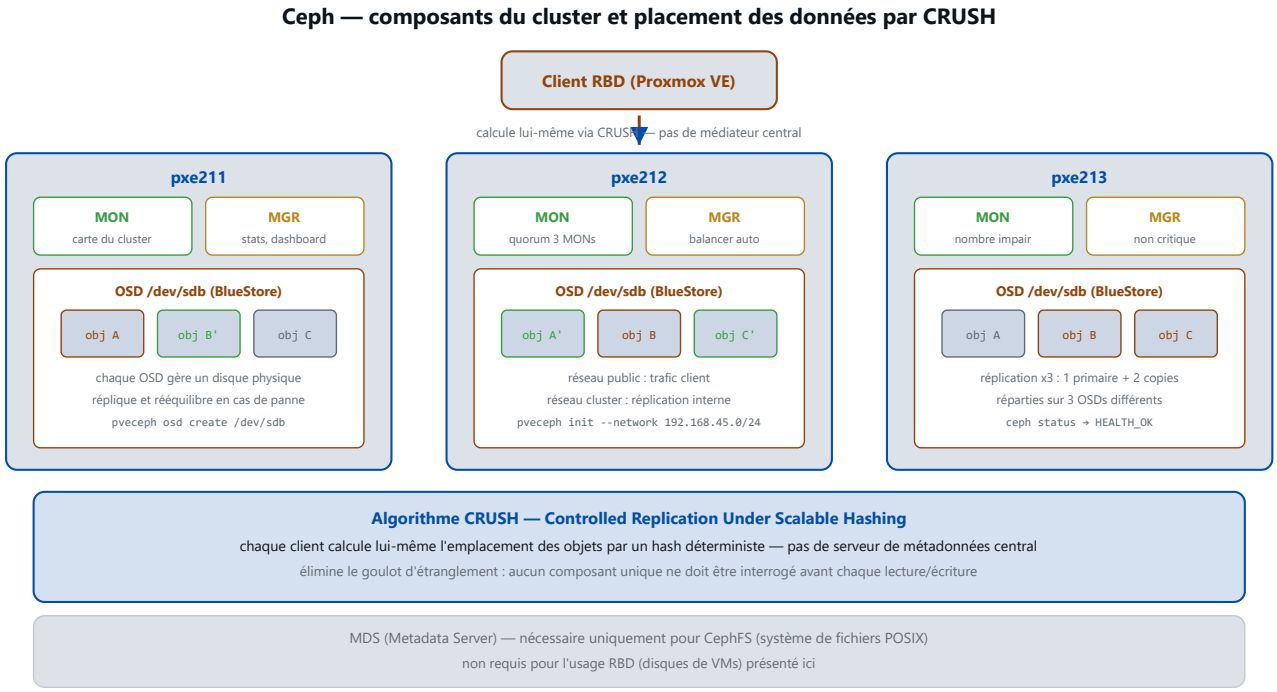
Trois managers ne rendent pas Ceph plus rapide — ils évitent de perdre la supervision le jour où un nœud tombe.

OSD — Object Storage Daemon : C'est le cœur du stockage. Chaque OSD gère un disque physique.

Quand vous écrivez une donnée, Ceph la découpe en objets et les répartit sur plusieurs OSDs selon les règles de réplication définies dans votre pool. Si un OSD tombe, les autres prennent le relais automatiquement et commencent à répliquer les données manquantes.

MDS — Metadata Server : Uniquement nécessaire pour CephFS (le système de fichiers POSIX de Ceph). Pour notre usage (stockage de disques VMs via RBD), les MDS ne sont pas requis.

Ce support est la propriété de Dawan France.



Revenons à la définition de base : un **stockage distribué** range les données sur les disques de plusieurs serveurs, et les garde lisibles si l'un d'eux tombe. Ceph se compose de plusieurs services, chacun avec son rôle :

Terme	Ce que c'est	Dans le laboratoire
OSD	Un démon par disque : il stocke les objets et réplique vers les autres	Un OSD par nœud, sur `/dev/sdb`
MON	Les moniteurs tiennent la carte du cluster ; leur majorité fait autorité	Trois MON, un par nœud
MGR	Supervision, tableau de bord, autoscaler ; actif/veille	Trois MGR, un actif
Pool	Un espace logique, avec son nombre de copies	`ceph-vm` (§4.1)
`size` / `min_size`	Le nombre de copies, et le minimum pour accepter les écritures	`3` / `2`
CRUSH	L'algorithme qui calcule où se trouve chaque objet (§4.2)	Une copie par nœud
RBD	Un disque bloc, découpé en objets dans le pool	Les disques des VM sur `ceph-vm`
MDS	Le serveur de métadonnées, requis seulement pour CephFS	Non déployé

Ce support est la propriété de Dawan France.

Un nœud qui tombe ne fait perdre aucune donnée : les deux autres ont une copie, et les OSD en recréent une troisième. Aucun serveur central n'est consulté pour trouver un objet : chaque client calcule son emplacement avec CRUSH, ce qui évite le point de défaillance unique.

Où voir Ceph sur votre cluster :

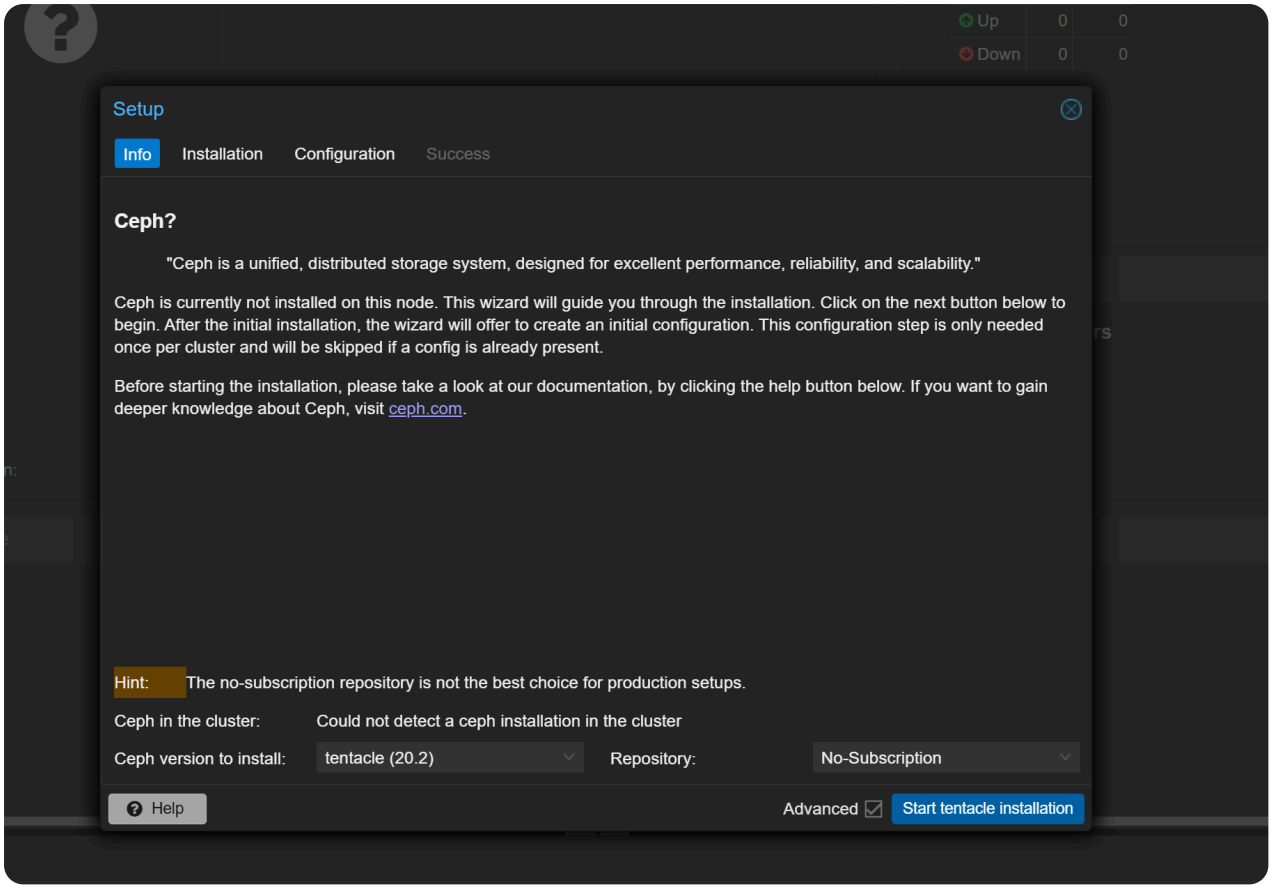
Notion	Où la voir
L'état global	**Nœud** → **Ceph** (tableau de bord : `HEALTH_OK`, MON, OSD) ; `ceph -s`
Les moniteurs et managers	**Nœud** → **Ceph** → **Monitor** ; lignes `mon:` et `mgr:` de `ceph -s`
Les OSD et leur disque	**Nœud** → **Ceph** → **OSD** ; `ceph osd tree`
Les pools, `size` et `min_size`	**Nœud** → **Ceph** → **Pools** ; **Datacenter** → **Storage** : le stockage de type RBD

3.2 Déployer Ceph depuis Proxmox VE

L'installation de Ceph dans Proxmox se fait idéalement depuis l'interface graphique.

Étape 1 — Installer Ceph (sur chaque nœud) :

> **Nœud → Ceph → Install Ceph**

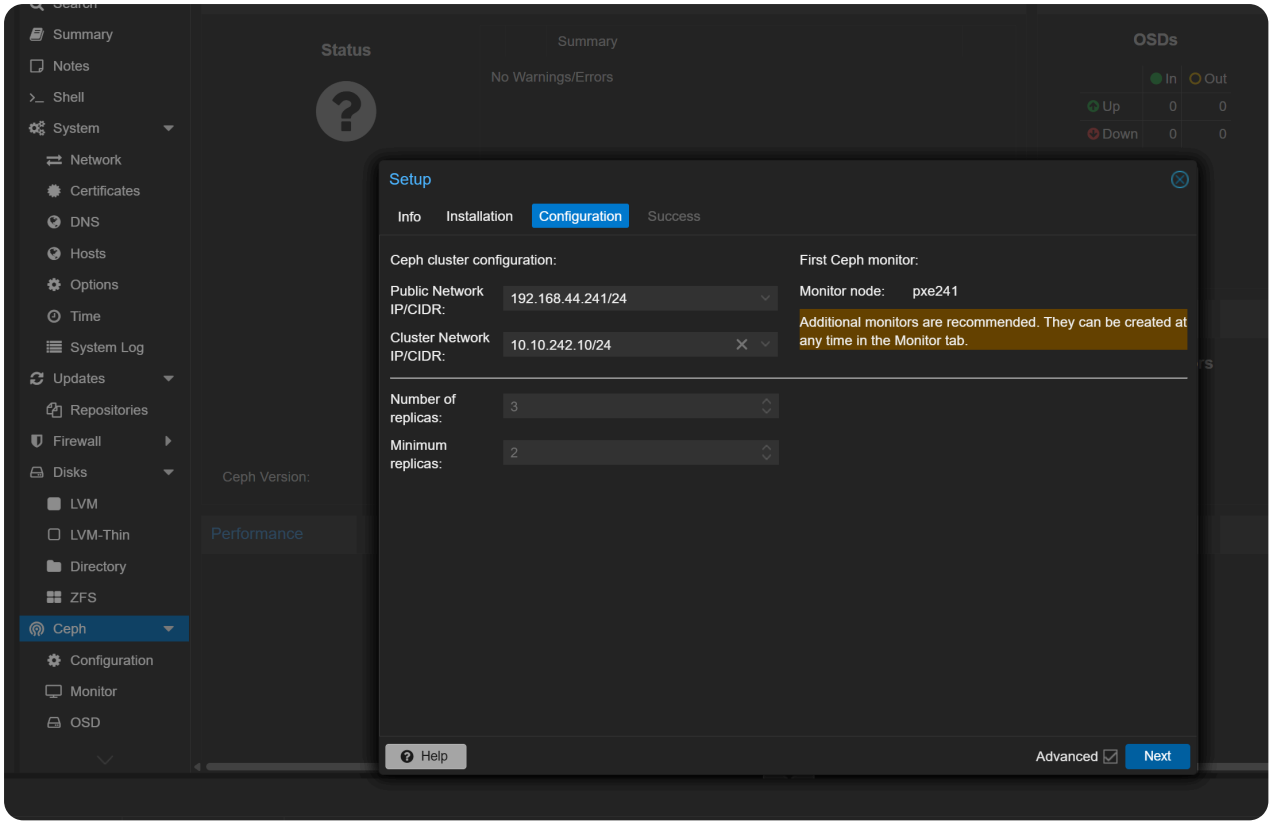


Le premier onglet de l'assistant, avant toute installation. Trois choses à y lire :

- **Ceph version to install** : la version proposée, ici **tentacle (20.2)**. Prenez celle que Proxmox propose — elle est alignée sur votre version de PVE, et mélanger les versions entre nœuds vous coûterait cher.
- **Repository : No-Subscription** : le dépôt gratuit, celui que vous avez configuré au début. Le champ existe parce qu'un abonnement donne accès au dépôt *Enterprise*, plus lentement mis à jour mais davantage testé.
- **Ceph in the cluster : Could not detect a ceph installation** : normal sur le premier nœud. Sur les deux suivants, cette ligne indiquera au contraire la version déjà présente — c'est votre confirmation que vous installez bien la même.

Le bandeau **Hint** dit sans détour que le dépôt sans abonnement n'est pas le meilleur choix en production. C'est vrai, et ça n'empêche rien : les paquets sont les mêmes, ils arrivent simplement plus tôt.

Ce support est la propriété de Dawan France.



C'est l'écran où tout se joue. L'assistant se déroule en quatre onglets (**Info**, **Installation**, **Configuration**, **Success**) ; celui-ci est le troisième, et le seul qui demande de vraies décisions.

- **Public Network IP/CIDR** : le réseau par lequel les **clients** parlent à Ceph : c'est par là que vos VM liront et écriront leurs disques. Sur le lab, le réseau préparé sur `ens21`.
- **Cluster Network IP/CIDR** : le réseau que les **OSD** utilisent entre eux pour se répliquer. C'est le trafic le plus lourd de Ceph — chaque écriture est recopiée deux fois de plus — et c'est pourquoi on le sépare du réseau public quand on le peut. Sur le lab, le réseau dédié préparé sur `ens20`.
- **Number of replicas : 3** — chaque donnée existe en trois exemplaires, sur trois nœuds différents.
- **Minimum replicas : 2** — le nombre de copies encore disponibles en dessous duquel Ceph **refuse d'écrire**. C'est la valeur qui vous protège.

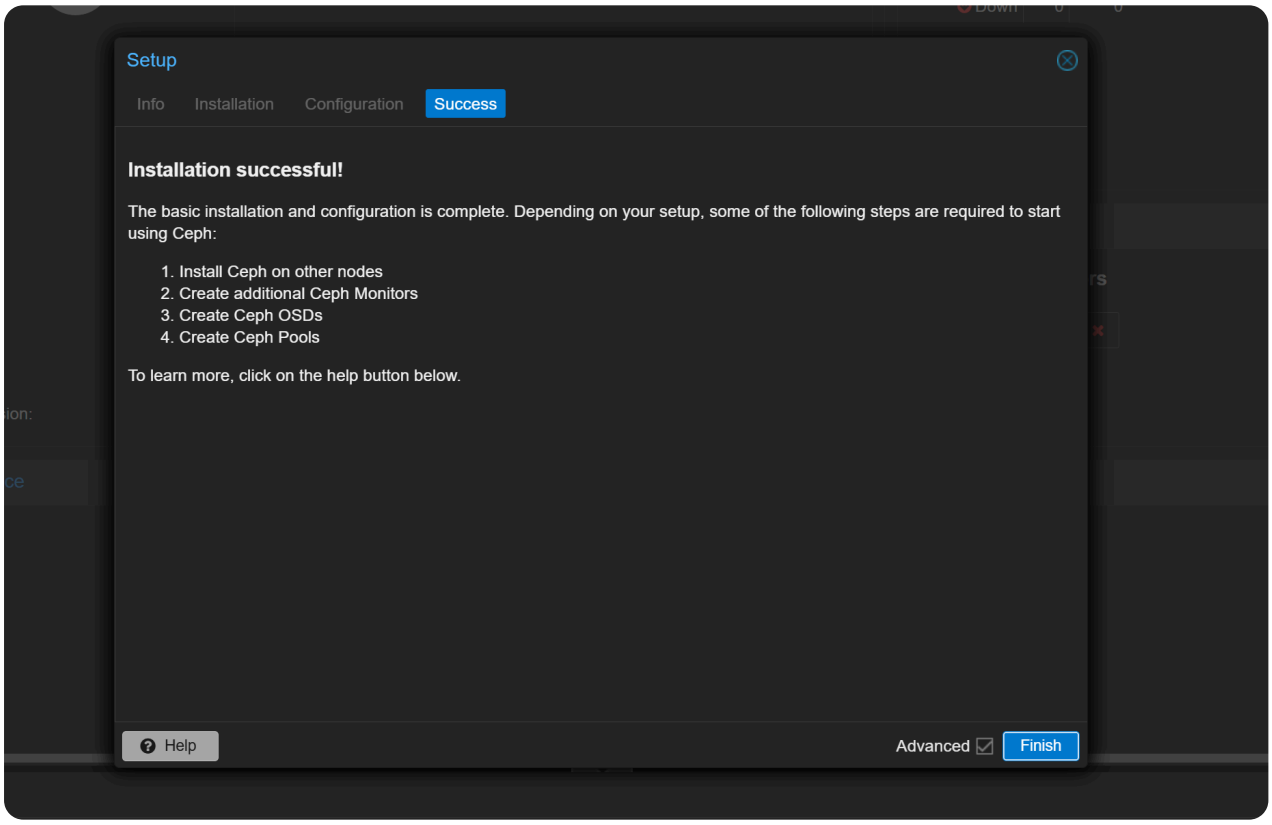
  Avec 3 et 2, un nœud peut tomber sans que rien ne s'arrête. Si un deuxième tombe, Ceph bloque les écritures plutôt que de risquer une donnée en exemplaire unique.
- **First Ceph monitor** : le nœud qui hébergera le premier moniteur. L'encadré jaune le dit clairement : *« Additional monitors are recommended »*.

Ce support est la propriété de Dawan France.

> Un moniteur seul est un point de défaillance. Vous en ajouterez deux autres à l'étape suivante, pour que la carte du cluster tienne, elle aussi, à la majorité.

Remarquez enfin la colonne de gauche : sous **Ceph** apparaissent déjà **Configuration**, **Monitor** et **OSD** — les trois endroits où vous reviendrez. Et en haut à droite, le compteur **OSDs** affiche `0` partout : normal, aucun disque n'a encore été donné à Ceph.

> **Les deux réseaux peuvent être le même**, et Ceph fonctionnera. Mais la réplication passera alors dans le même tuyau que les accès disque de vos VM.
>
> Une reconstruction après panne de disque saturera le réseau au pire moment : exactement quand les machines ont besoin de leurs données. Séparez-les dès que vous avez une seconde carte.



La dernière page de l'assistant : une liste, à garder en tête. L'assistant se termine en énumérant lui-même ce qu'il reste à faire : installer Ceph sur les autres nœuds, créer les moniteurs supplémentaires, créer les OSD, créer les pools.

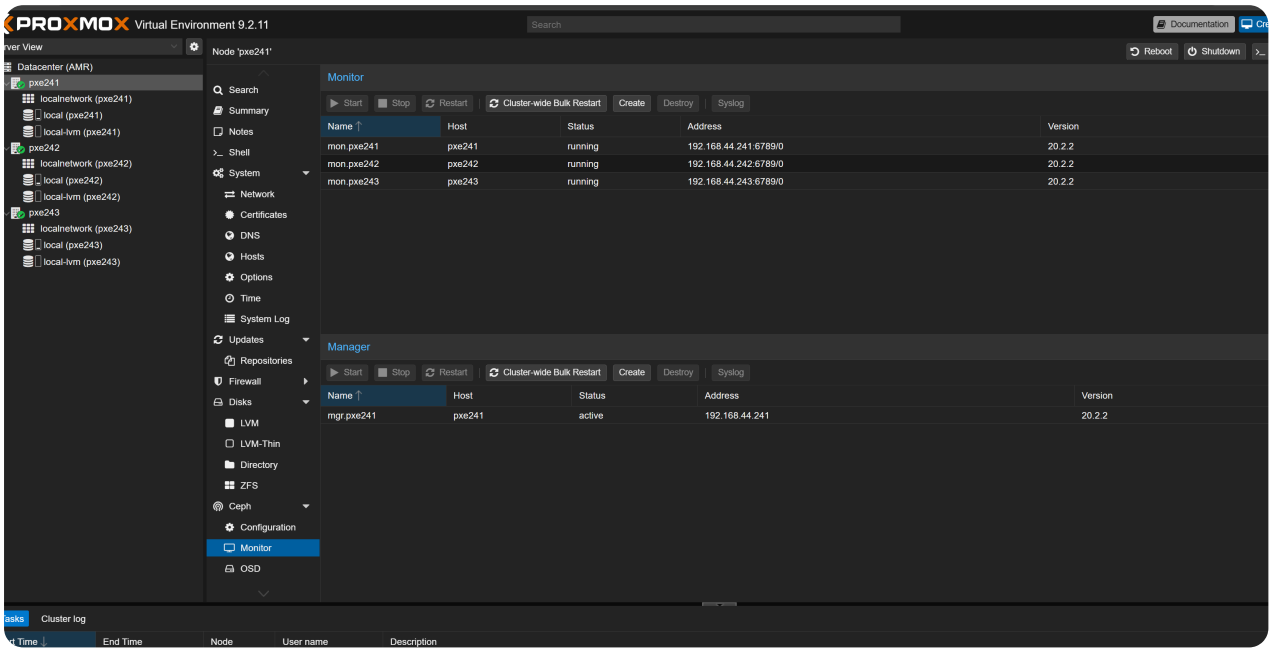
« Installation successful » ne signifie donc pas « Ceph est prêt » : à ce stade, aucun disque n'a encore été donné au cluster et aucune donnée ne peut y être écrite. C'est exactement le programme des sections qui suivent.

Ce support est la propriété de Dawan France.

Sélectionner la version **Tentacle** (version stable actuelle). Choisir le réseau public Ceph. Cliquer **Start installation** et attendre la fin.

Étape 2 — Créer le Monitor :

> **Nœud → Ceph → Monitor → Create** (répéter sur les 3 nœuds)



Regardez les deux tableaux ensemble. Le même écran porte les deux services, et leur état révèle tout ce qu'on vient de dire.

- En haut, **Monitor** : trois moniteurs, `mon.pxe211`, `mon.pxe212` et `mon.pxe213`, tous `running`, un par nœud. Chacun écoute sur le port **6789**, celui de Ceph, à une adresse du réseau public de Ceph (`10.44.31.x` sur le lab).
- En bas, **Manager** : **un seul**, `mgr.pxe211`, en `active`. C'est celui que l'assistant a créé tout seul avec le premier moniteur.

C'est précisément la situation à corriger. Le bouton **Create** de la section *Manager* sert à ajouter les deux autres. Une fois fait, la liste affichera un manager `active` et deux `standby` — et la disparition de `pxe211` ne vous privera plus ni des statistiques ni du tableau de bord.

Notez au passage le bouton **Cluster-wide Bulk Restart**, présent sur les deux sections : il redémarre le service sur tous les nœuds à la fois, utile après un changement de configuration.

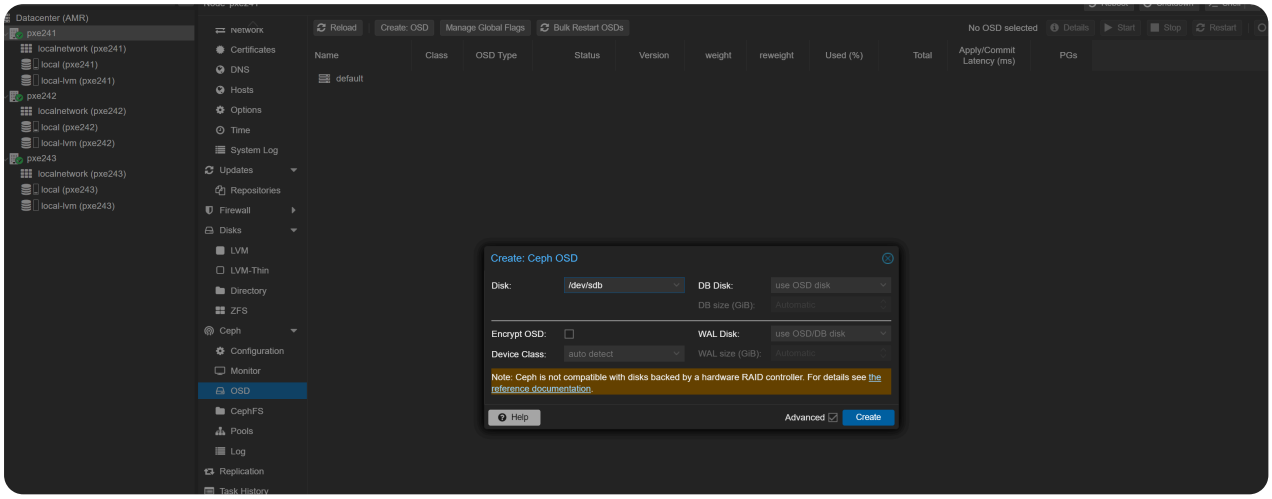
Étape 3 — Créer le Manager :

> **Nœud → Ceph → Monitor**, section **Manager → Create** (répéter sur les 3 nœuds)

Ce support est la propriété de Dawan France.

Étape 4 — Créer les OSDs :

> **Nœud → Ceph → OSD → Create: OSD** — sélectionner `/dev/sdb`, laisser **BlueStore** (le format de stockage des OSD, proposé par défaut), cliquer **Create** (répéter sur les 3 nœuds)



La fenêtre qui transforme un disque en OSD.

- **Disk** : le disque à donner à Ceph, ici `/dev/sdb`. La liste ne propose que les disques **entièrement vierges** : si le vôtre n'apparaît pas, c'est qu'il porte encore une signature de partition, de LVM ou de ZFS.
- **DB Disk** (*database*, base de métadonnées) et **WAL Disk** (*Write-Ahead Log*, journal d'écriture), laissés sur *use OSD disk*. Ceph peut placer sa base de métadonnées et son journal d'écriture sur un **disque plus rapide** que celui des données : un SSD devant des disques mécaniques, par exemple. Avec des disques identiques, on laisse tout sur le même.
- **Device Class : auto detect** : Ceph classe seul le disque en `hdd`, `ssd` ou `nvme`. Cette étiquette sert à créer des règles de placement par type de matériel.
- **Encrypt OSD** : chiffre le disque au repos. À décider **maintenant** — l'activer ensuite impose de détruire l'OSD et de le recréer.

L'encadré orange mérite d'être lu : **Ceph n'est pas compatible avec un disque présenté par une carte RAID matérielle.** Ceph veut voir le disque tel quel, pour gérer lui-même la redondance ; un RAID en dessous lui cacherait les pannes et ferait doublon avec sa propre réplication. En présence d'un contrôleur RAID, on le passe en mode *IT / HBA* (*Initiator Target / Host Bus Adapter* : le contrôleur présente les disques tels quels).

Équivalent CLI (séquence complète) :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# ===== Sur pxe211 (premier nœud) =====
# Installer les paquets Ceph (ce que fait l'onglet Installation de l'assistant).
# « yes | » répond à la question d'apt [Y/n] ; sans lui, apt abandonne.
yes | pveceph install --version tentacle --repository no-subscription

# Initialiser Ceph : --network = réseau PUBLIC (celui des clients),
# --cluster-network = réseau de RÉPLICATION entre OSD
pveceph init --network 10.<site>.<N+30>.0/24 --cluster-network 10.<site>.<N+20>.0/24

# Créer le premier Monitor sur ce nœud
pveceph mon create

# Créer le Manager
pveceph mgr create

# Créer le premier OSD avec le disque dédié Ceph
pveceph osd create /dev/sdb
```

Résultat attendu de `pveceph osd create /dev/sdb` :

```
create OSD on /dev/sdb (bluestore)
wiping block device /dev/sdb
200+0 records in
200+0 records out
209715200 bytes (210 MB, 200 MiB) copied, 0.52 s, 403 MB/s
Running command: /usr/bin/ceph-authtool --gen-print-key
...
--> ceph-volume lvm activate successful for osd ID: 0
--> ceph-volume lvm create successful for: /dev/sdb
```

La sortie, longue, se lit par ses extrémités. En tête, `wiping block device` : Proxmox efface le début du disque — d'où l'exigence d'un disque vierge, qu'il vérifie avant d'écrire quoi que ce soit.

À la fin, les deux lignes `successful` : l'OSD a reçu son numéro (`osd ID: 0`) et il est actif. Entre les deux, `ceph-volume` a fait du disque un volume LVM au format **BlueStore**, le format de stockage par défaut depuis Ceph Luminous.

`pveceph mon create` et `pveceph mgr create` sont plus discrets : quelques lignes de création de clés, puis le service démarre. Vérifiez l'état avec `ceph status` une fois tous les nœuds configurés.

Ce support est la propriété de Dawan France.

Dans le Shell de chaque nœud du cluster :

```
# ===== Répéter sur pxe212 et pxe213 =====
pveceph mon create
pveceph mgr create   # un manager par nœud (un actif, les autres en veille)
pveceph osd create /dev/sdb   # le disque libre devient un OSD
```

Résultat attendu de `pveceph osd create /dev/sdb` sur pxe212/pxe213 :

```
create OSD on /dev/sdb (bluestore)
wiping block device /dev/sdb
...
--> ceph-volume lvm create successful for: /dev/sdb
```

Après ces commandes sur les 3 nœuds, le cluster Ceph dispose de 3 moniteurs, 3 managers et 3 OSDs — une configuration minimale redondante. Un seul OSD par nœud suffit pour un cluster d'atelier ; en production, plusieurs OSDs par nœud (un par disque physique) sont recommandés pour maximiser les performances parallèles.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Vérifier l'état global du cluster Ceph
ceph status

# Voir l'arborescence des OSDs et leur état
ceph osd tree

# Vérifier l'utilisation de l'espace
ceph df
```

Résultat attendu de `ceph status` quand tout est en ordre :

```
  cluster:
    id:     a8f3b2c1-...
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum pxe211,pxe212,pxe213
```

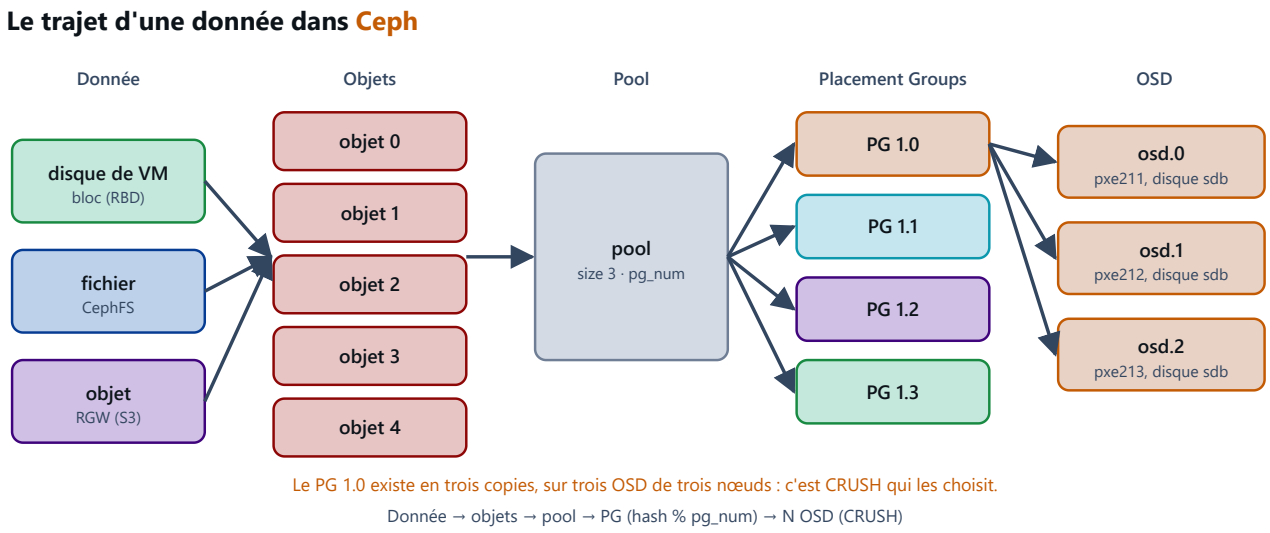
Ce support est la propriété de Dawan France.

```
    mgr: pxe211(active), standbys: pxe212, pxe213
    osd: 3 osds: 3 up, 3 in

  data:
    pools:   1 pools, 1 pgs
    objects: 0 objects, 0 B
    usage:   3.0 GiB used, 147 GiB / 150 GiB avail
```

Le schéma suivant montre le **trajet d'une donnée** dans Ceph, de gauche à droite. Un bloc, un objet ou un fichier est d'abord découpé en **objets**. Ces objets sont versés dans un **pool**.

Le pool les répartit dans ses **Placement Groups** : le `pgid` vaut simplement `hash(nom de l'objet) % pg_num`. Enfin, chaque PG (*Placement Group*) est répliqué sur plusieurs **OSD**, c'est-à-dire sur plusieurs disques de plusieurs serveurs. C'est cette chaîne qui explique qu'aucun disque ne détienne la totalité d'une donnée, et qu'en perdre un ne perde rien :



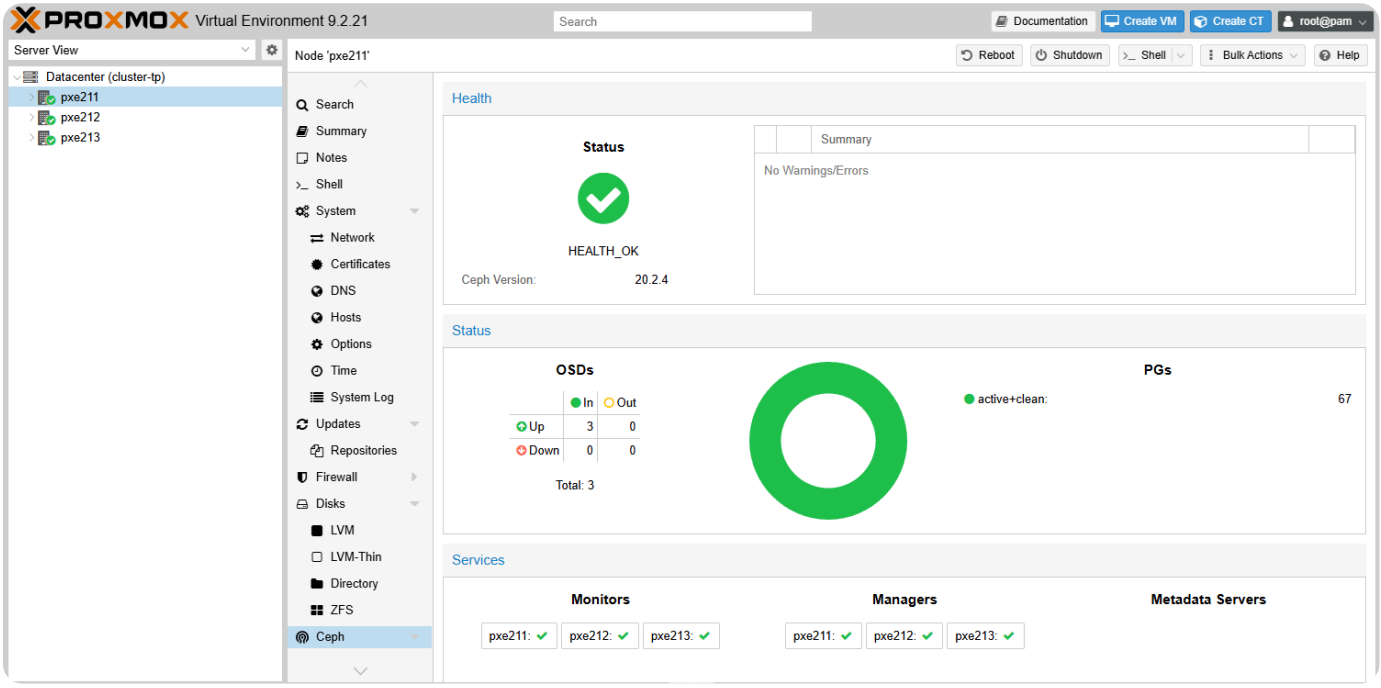
- 📖 Documentation Proxmox — Ceph Server
- 📖 Documentation Ceph

À retenir : Ceph répartit les données sur tous les disques du cluster, sans point de défaillance unique. Les **MON** tiennent la carte du cluster, les **OSD** gèrent chacun un disque, le **MGR** assure supervision et tableau de bord. On l'installe depuis Proxmox, nœud par nœud, avec de préférence un réseau de réplication séparé.

Ce support est la propriété de Dawan France.

4. Ceph : pools, placement des données et résilience

Ceph est installé, ses moniteurs et ses OSD tournent. Il reste à en faire un stockage utilisable par les VM : un pool, et sa déclaration dans Proxmox. Puis à comprendre comment Ceph place et protège les données : l'algorithme CRUSH, le facteur de réplication et ses réseaux.



Le tableau de bord **Nœud → Ceph** une fois l'installation finie (cluster du lab, Ceph 20.2.4). En haut, **HEALTH_OK** et « No Warnings/Errors ». Au milieu, les **OSD** : 3 `In`, 3 `Up`, aucun `Down` ni `Out` ; et les **PG**, tous `active+clean` (l'anneau vert).

En bas, les **Monitors** et les **Managers**, un de chaque sur `pxe211`, `pxe212` et `pxe213`. C'est la même information que `ceph -s`, présentée pour l'œil.

4.1 Créer un pool Ceph et l'intégrer à Proxmox

Un pool Ceph est un espace de stockage logique avec ses propres règles de réplication. Pour héberger des disques de VMs, on utilise le protocole RBD (Rados Block Device), qui expose des disques virtuels directement à partir du stockage objet Ceph.

Via l'interface graphique :

Ce support est la propriété de Dawan France.

Nœud → Ceph → Pools → Create

Champ	Valeur
Name	`ceph-vm`
Size (réplication)	`3`
Min. Size	`2`
PG Autoscale	`on`
Add as Storage	✓

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Créer un pool pour les disques de VMs
# --pg_autoscale_mode on : Ceph ajuste automatiquement le nombre de placement groups
# --application rbd : indique que ce pool sera utilisé pour les disques blocs
pveceph pool create ceph-vm --pg_autoscale_mode on --application rbd

# Ajouter ce pool comme stockage dans Proxmox
pvesm add rbd ceph-vm --pool ceph-vm --content images

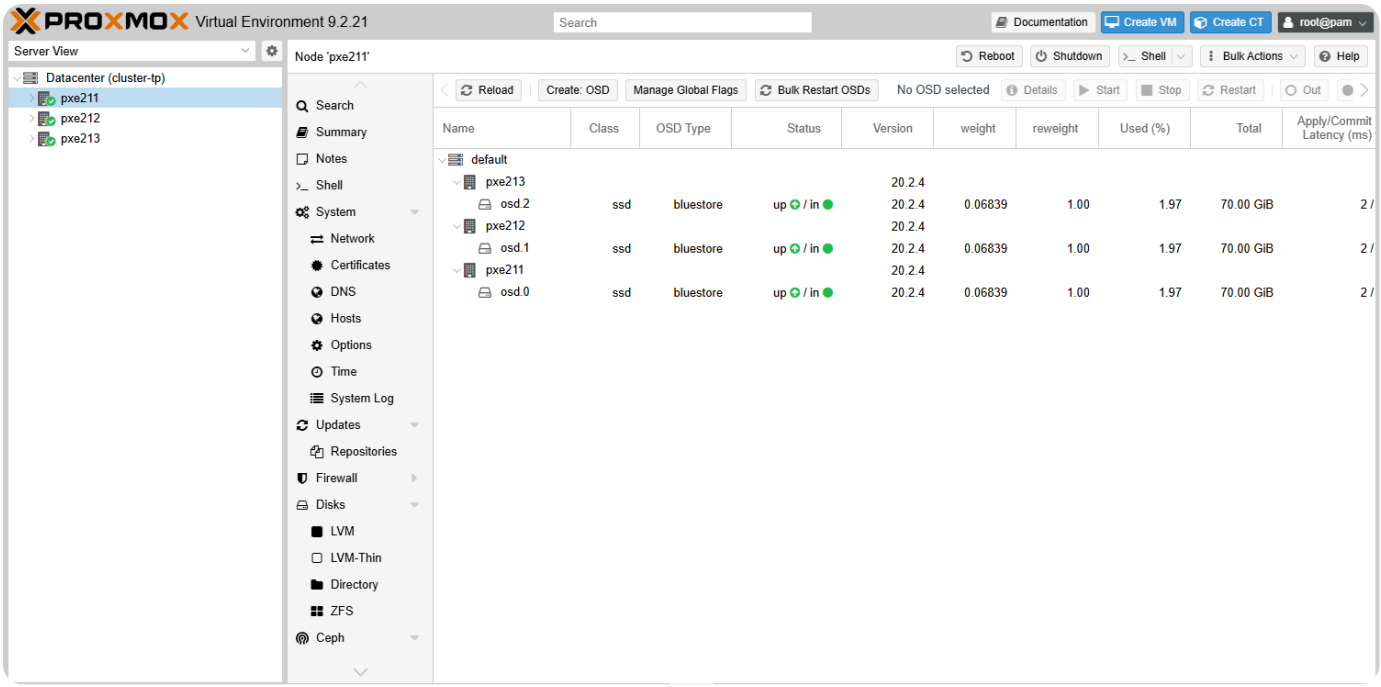
# Vérifier que le stockage est actif
pvesm status
```

Résultat attendu :

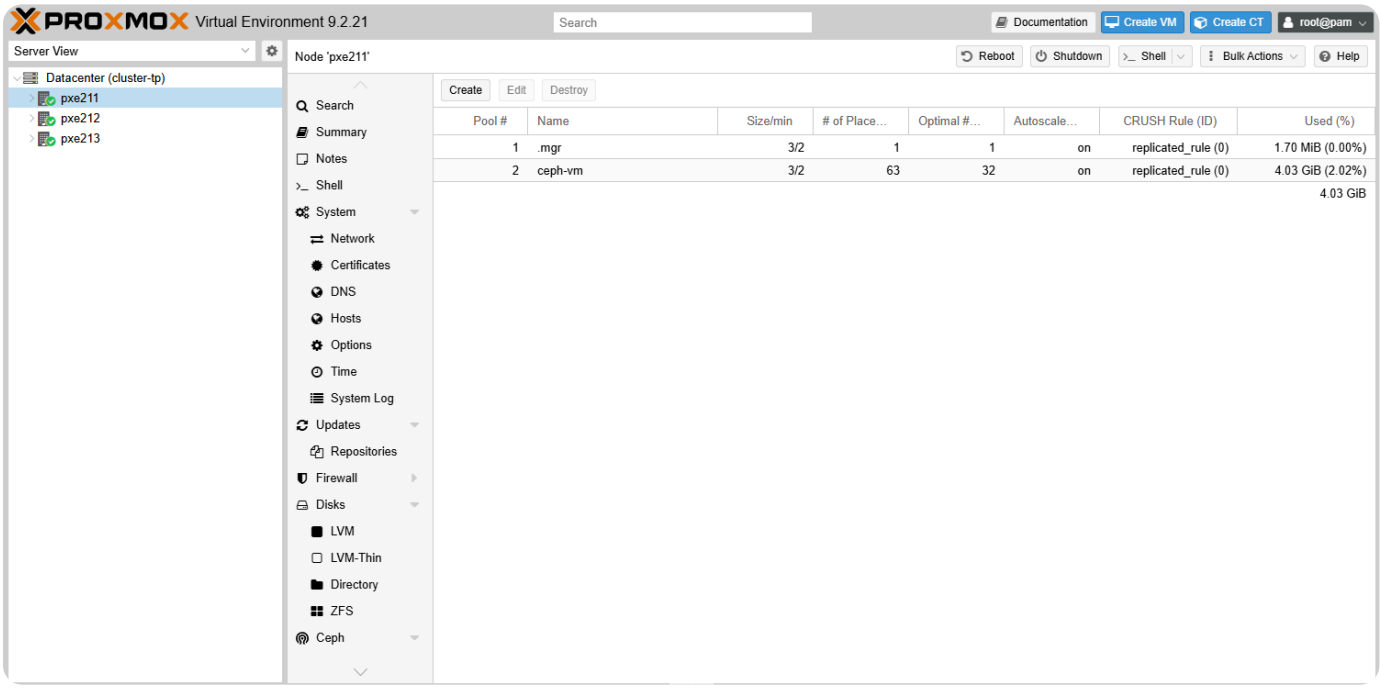
```
Name          Type     Status           Total
ceph-vm      rbd      active     150000000
```

À partir de ce moment, vous pouvez créer des VMs dont les disques sont hébergés dans Ceph. Ces disques sont accessibles depuis tous les nœuds du cluster simultanément, ce qui rend possible la migration à chaud et la haute disponibilité.

Ce support est la propriété de Dawan France.



Nœud → Ceph → OSD range les OSD par nœud : `osd.0` sur `pxe211`, `osd.1` sur `pxe212`, `osd.2` sur `pxe213`. Chacun est de type `bluestore`, marqué `up` et `in`, et dispose de 70 Gio — le disque `sdb` du nœud.



Nœud → Ceph → Pools : `ceph-vm`, créé ci-dessus, en `3/2` (trois copies, deux au minimum pour écrire), autoscaler `on`, règle `replicated_rule`. La ligne `.mgr` est un pool interne, créé par Ceph pour ses managers : on n'y touche pas.

Le pool et le stockage : deux objets différents

Ce support est la propriété de Dawan France.

Créer le pool n'a fait qu'une chose : réserver un espace **côté Ceph**. Proxmox, lui, ne sait pas encore qu'il a le droit d'y déposer des disques.

Il lui faut pour cela un **stockage de type RBD** déclaré dans `Datacenter → Storage`, qui pointe vers ce pool. Sans cette déclaration, le pool existe et reste inutilisable : il n'apparaît nulle part au moment de créer un disque.

La case `Add as Storage` fait ce travail pour vous — comme l'option `--add_storages` en ligne de commande. Vérifiez-le :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesm status | grep rbd   # le stockage RBD doit être actif
grep -A 5 "^rbd:" /etc/pve/storage.cfg   # sa déclaration dans la configuration des stockages
```

Résultat attendu : une section `rbd: ceph-vm` avec la ligne `pool ceph-vm`. C'est elle, le stockage ; le pool est ce qu'elle désigne.

Quand faut-il le déclarer à la main ? Trois situations, et vous rencontrerez la dernière en entreprise :

- la case a été décochée, ou le pool a été créé sans `--add_storages` ;
- vous voulez un autre identifiant, d'autres types de contenu, ou restreindre le stockage à certains nœuds ;
- **le cluster Ceph est externe**, hébergé ailleurs que sur vos nœuds. Dans ce cas il n'y a aucun pool à créer dans l'interface — on déclare seulement le stockage RBD, avec l'adresse des moniteurs et la clé d'accès.

Dans l'interface : `Datacenter` → **Storage** → **Add** → **RBD**.

Champ	Valeur	Ce que ça fait
ID	`ceph-vm`	Le nom du stockage dans Proxmox
Pool	`ceph-vm`	Le pool Ceph visé
Monitor(s)	*(laisser vide)*	Vide = le cluster Ceph local ; à renseigner pour un Ceph externe
User name	`admin`	L'utilisateur Ceph, `admin` en hyperconvergé

Ce support est la propriété de Dawan France.

Champ	Valeur	Ce que ça fait
Content	`Disk image`, `Container`	Ce qu'on autorise à y ranger
Nodes	*(tous)*	Ceph étant partagé, laissez les trois

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Equivalent en ligne de commande
pvesm add rbd ceph-vm --pool ceph-vm --content images,rootdir
pvesm status   # vérifier qu'il est actif
```

Résultat attendu :

```
root@pxe211:~# pvesm add rbd ceph-vm --pool ceph-vm --content images,rootdir
root@pxe211:~# pvesm status
Name             Type     Status           Total            Used       Available
%
ceph-vm           rbd     active       314572800        18874368       295698432
6.00%
local             dir     active        98559220        22417684        71095996
22.75%
local-lvm     lvmthin     active       167772160        39267532       128504628
23.40%
```

Le stockage RBD apparaît sur tous les nœuds du cluster à la fois, sans qu'il faille répéter la commande ailleurs : la déclaration vit dans `/etc/pve/storage.cfg`, répliqué par `pmxcfs`.

La capacité annoncée est l'espace **utile**, après réplication. Avec le facteur 3 par défaut de Ceph, 300 Gio disponibles correspondent à 900 Gio de disques bruts. C'est le coût de la redondance, et il surprend souvent au premier calcul de dimensionnement.

> `--content images,rootdir` **limite ce pool aux disques de VM et de conteneurs.** Un pool RBD ne peut pas héberger d'ISO ni de sauvegardes : ce sont des fichiers, et RBD expose des périphériques de bloc. Pour des fichiers sur Ceph, il faut CephFS, déclaré séparément.

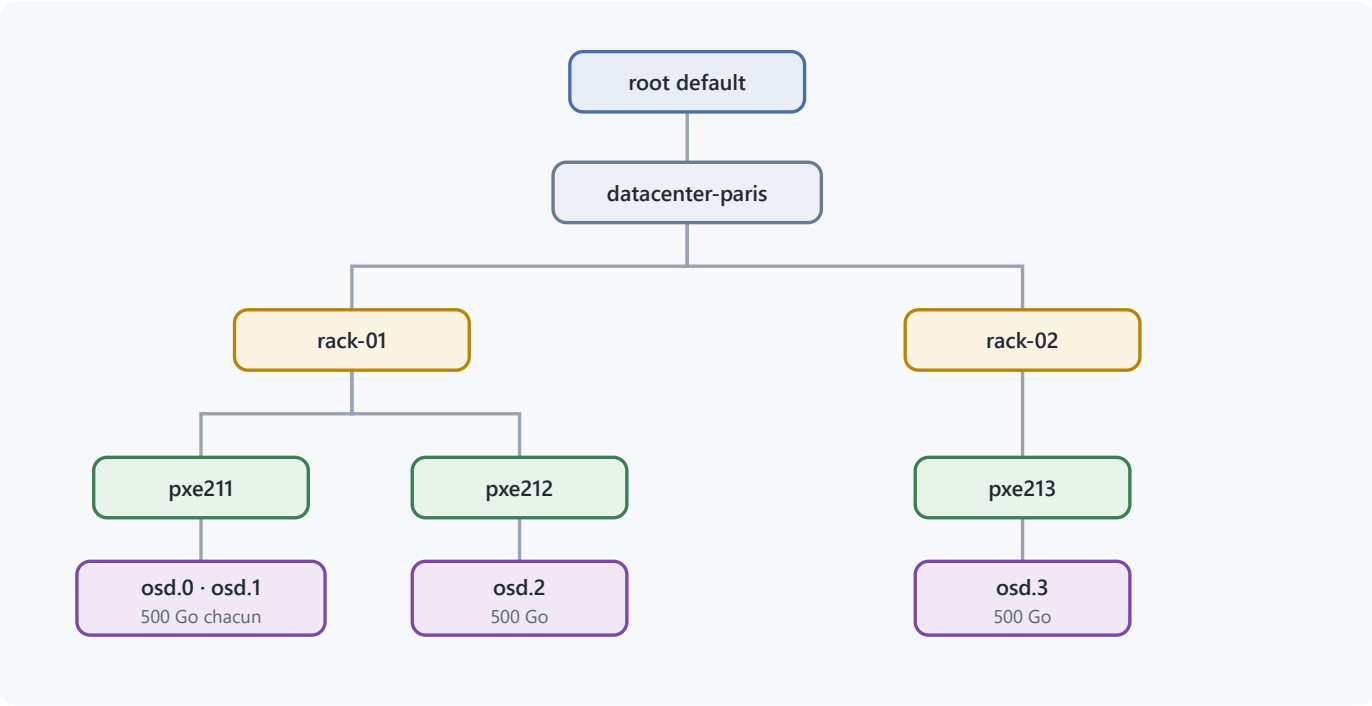
> **Erreur fréquente :** Dans un atelier avec du nested virtualization, les performances Ceph seront faibles car les disques `sdb` sont eux-mêmes des disques virtuels. Les erreurs de

Ce support est la propriété de Dawan France.

> latence élevée (`slow ops`) sont normales dans ce contexte et ne doivent pas alarmer. En production, Ceph s'utilise sur des disques physiques dédiés, idéalement des SSDs NVMe pour les journaux.

4.2 CRUSH, l'algorithme de placement

Ceph n'a pas de serveur de métadonnées central : chaque client **calcule** lui-même où sont ses données, avec l'algorithme **CRUSH** (*Controlled Replication Under Scalable Hashing*). CRUSH s'appuie sur une carte de la topologie physique, la **CRUSH map** : centre de données, baies, hôtes, OSD.



Quand un pool a `size=3`, CRUSH place les trois répliques sur trois hôtes différents ; une règle de baie les placerait sur des baies différentes. `ceph osd tree` affiche cette hiérarchie.

4.3 Facteur de réplication et dimensionnement du cluster Ceph

Par défaut, Ceph crée les pools avec `size=3` (3 répliques) et `min_size=2` (minimum pour accepter les écritures). Chaque objet est stocké en 3 exemplaires, tolérant la perte simultanée d'un OSD.

Impact sur l'espace utile : avec `size=3`, seul 1/3 de l'espace brut est utilisable. 3 nœuds × 500 Go = 1500 Go bruts → 500 Go utiles. C'est le coût de la résilience.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Vérifier le facteur de réplication d'un pool
ceph osd pool get ceph-vm size
# → size: 3

ceph osd pool get ceph-vm min_size
# → min_size: 2

# Voir la capacité totale (RAW vs AVAIL)
ceph df
```

Résultat attendu de `ceph df` :

```
--- RAW STORAGE ---
CLASS  SIZE     AVAIL    USED     RAW USED  %RAW USED
hdd    150 GiB  147 GiB  3.0 GiB   3.0 GiB       2.00
TOTAL  150 GiB  147 GiB  3.0 GiB   3.0 GiB       2.00
```

Pour calculer l'espace utile : `(150 GiB bruts) / 3 (réplication) × 80% (marge opérationnelle) = 40 GiB utilisables`. Les 20% de marge absorbent les rééquilibrages lors des opérations de maintenance.

> **Règle de dimensionnement :** ne jamais remplir un cluster Ceph au-delà de 80% de son espace utile. Les seuils de Ceph sont ensuite rapides : à **85 %** d'un OSD (`nearfull`), le cluster passe en `HEALTH_WARN` ; à **95 %** (`full`), il **bloque toutes les écritures** pour protéger l'intégrité des données.
>
> Et la marge de 20 % n'est pas du confort : quand un nœud tombe, ses données sont reconstruites sur les OSD restants, qui doivent avoir la place de les recevoir.

4.4 Réseau public et réseau cluster

En production, Ceph sépare deux trafics. Le **réseau public** relie les clients (VM, Proxmox) et les OSD ; le **réseau cluster** relie les OSD entre eux, pour la réplication et la recopie après une panne. Les séparer protège les performances des VM pendant une recopie massive.

Les deux se fixent à l'initialisation de Ceph (`--network` et `--cluster-network`), et se lisent dans **Nœud → Ceph → Configuration**.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

À retenir : un **pool RBD** porte les disques de VM et de conteneurs (pas d'ISO ni de sauvegardes). **CRUSH** calcule l'emplacement des données sans serveur de métadonnées.

`size=3` / `min_size=2` : trois copies, écritures acceptées tant que deux restent ; un tiers de l'espace brut est utile, et on ne dépasse pas 80 %. Les réseaux Ceph se fixent dès l'initialisation.

Ce support est la propriété de Dawan France.

5. L'infrastructure hyperconvergée — Proxmox + Ceph

5.1 Du modèle à trois niveaux à l'hyperconvergence

Depuis que vous avez constitué votre cluster Proxmox avec Ceph, vous avez en réalité construit quelque chose qui a un nom dans l'industrie : une **infrastructure hyperconvergée**, ou HCI (Hyper-Converged Infrastructure). Comprendre ce concept aide à positionner Proxmox dans le paysage des plateformes d'infrastructure et à justifier vos choix techniques face à des décideurs ou lors d'appels d'offres.

L'architecture traditionnelle, ou infrastructure dite "à trois niveaux", sépare strictement les rôles. Des serveurs de calcul, sans disque ou avec peu de disque local, se connectent à une baie de stockage centralisée. Ils passent par un réseau de stockage dédié : SAN Fibre Channel ou iSCSI.

Le réseau de production est une troisième couche séparée. Cette architecture est robuste et connue, mais elle implique trois types de matériel différents et, souvent, trois équipes différentes. Elle entraîne aussi un coût d'investissement (CAPEX : *capital expenditure*, dépenses d'investissement) élevé dès le premier nœud.

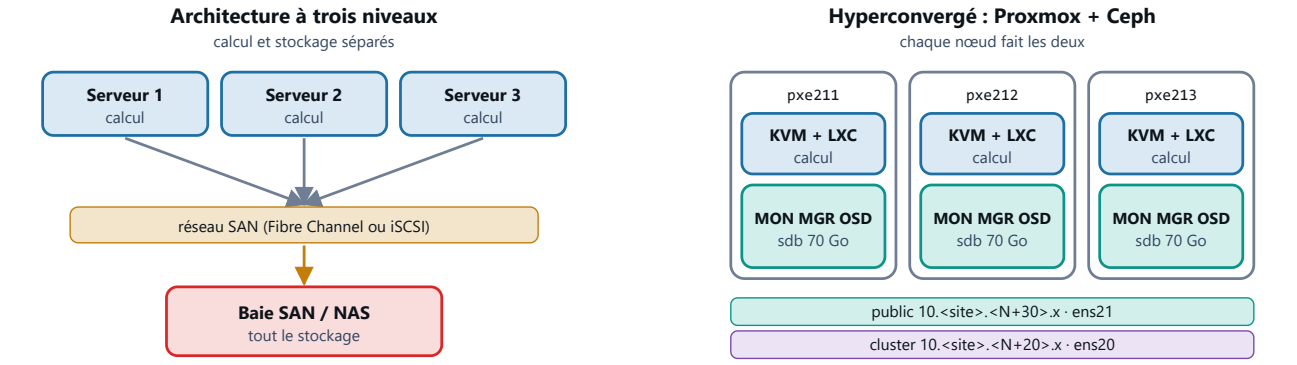
Architecture traditionnelle (3 tiers) : les Serveurs 1, 2 et 3 (calcul uniquement) se connectent via un réseau SAN à une baie centralisée (stockage). Cette baie est de type SAN ou NAS (*Network Attached Storage*, stockage en réseau par fichiers). Les deux couches sont physiquement séparées.

Infrastructure hyperconvergée (HCI) avec Proxmox + Ceph : `pxe211`, `pxe212` et `pxe213` font tourner à la fois KVM + LXC (calcul) et MON+MGR+OSD (stockage Ceph, disque `sdb` 70 Go chacun). Ils sont reliés par deux réseaux Ceph dédiés : public `10.<site>.<N+30>.x` sur `ens21`, cluster `10.<site>.<N+20>.x` sur `ens20`. Calcul et stockage cohabitent sur les mêmes nœuds.

Ce support est la propriété de Dawan France.

De trois niveaux séparés à des nœuds qui font tout

Cliquez sur une des deux architectures



Même service rendu aux VM, deux façons de le construire.
Le cluster du chapitre est déjà une infrastructure hyperconvergée (HCI) : il n'y a plus de baie.

Les deux architectures décrites ci-dessus, côte à côte. À gauche, la baie centrale et son réseau de stockage ; à droite, le cluster du lab, où chaque nœud porte les VM et un morceau du stockage Ceph. Cliquez sur une architecture pour ses forces et ses limites.

Dans une architecture HCI, chaque nœud joue simultanément les deux rôles : calcul (hyperviseur) **et** stockage (OSD Ceph). Il n'y a plus de baie SAN externe.

L'espace de stockage est fourni par les disques présents dans chaque serveur, agrégé et distribué par Ceph. Quand une VM sur pxe211 écrit sur son disque, l'OSD local reçoit la donnée. Il la réplique ensuite sur les OSDs de pxe212 et pxe213, en parallèle, sur le réseau interne Ceph.

Les avantages de l'HCI Proxmox + Ceph :

Le premier avantage est l'élimination du SPOF (*Single Point Of Failure*, point de défaillance unique) lié à la baie de stockage. Dans l'architecture traditionnelle, la perte de la baie SAN met hors service l'ensemble des VMs. Avec Ceph, la perte d'un nœud complet n'entraîne qu'un rééquilibrage automatique — les deux nœuds restants continuent de servir les données tant que `min_size` est respecté.

Le deuxième avantage est la **montée en charge linéaire**. Ajouter un quatrième nœud à votre cluster Proxmox Ceph augmente simultanément la puissance de calcul disponible (un hyperviseur supplémentaire) et la capacité de stockage Ceph (un OSD supplémentaire). Dans une architecture traditionnelle, ces deux dimensions évoluent indépendamment — et les projets d'extension du SAN sont souvent les plus coûteux.

Ce support est la propriété de Dawan France.

Les autres avantages de l'HCI avec Proxmox VE :

Avantage	Ce qu'il veut dire
Faible coût	Proxmox VE est open source et intègre tous les composants nécessaires
Protection des données	La sauvegarde et la reprise après sinistre sont intégrées
Simplicité	Une configuration et une administration centralisées, depuis l'interface du cluster
Open source	Pas de dépendance à un fournisseur unique (*vendor lock-in*)

Le prérequis : le réseau. Chaque écriture d'une VM est répliquée par Ceph sur les autres nœuds, à travers le réseau. Pour un cluster performant, la documentation de Proxmox recommande des réseaux physiques séparés, dont :

Réseau	Débit recommandé	Ce qui y passe
Ceph **cluster** (interne)	25 Gbit/s et plus	La réplication entre OSD
Ceph **public**	10 Gbit/s et plus	Le trafic entre les clients (les VM) et les OSD ; il peut aussi porter le trafic des invités et la migration à chaud

Sur le lab, les deux réseaux Ceph sont `ens20` (cluster) et `ens21` (public) : le §4.4 les détaille.

À retenir : Proxmox + Ceph forment une infrastructure **hyperconvergée** : chaque nœud porte à la fois le calcul et le stockage, au lieu de serveurs, réseau de stockage et baie séparés. Plus de point de défaillance unique, une croissance par ajout de nœuds, et un prérequis : un réseau Ceph rapide et dédié.

> **▶ À pratiquer maintenant — Atelier 5**
>
> Déployer Ceph sur les trois nœuds, créer son pool, y installer une VM, puis la migrer à chaud.

Ce support est la propriété de Dawan France.

6. Haute disponibilité (HA)

La migration à chaud du §2 répond à une maintenance **prévue** : l'administrateur choisit le moment, la VM part pendant qu'elle tourne. La haute disponibilité répond à ce qu'on ne choisit pas : un nœud qui s'éteint, une carte mère qui lâche, un service qui refuse de démarrer.

Dans ces cas, Proxmox VE décide seul de relancer les machines concernées ailleurs, sans intervention humaine. Cette section décrit précisément comment il prend cette décision, ce qu'il lui faut pour la prendre sans risque, et comment l'administrateur garde la main.

6.1 Ce que « haute disponibilité » veut dire

La haute disponibilité désigne la capacité d'un service à rester accessible malgré la défaillance d'un ou plusieurs composants de l'infrastructure. Elle se mesure en **pourcentage de temps de fonctionnement sur une année**. Le guide d'administration de Proxmox VE donne la correspondance suivante :

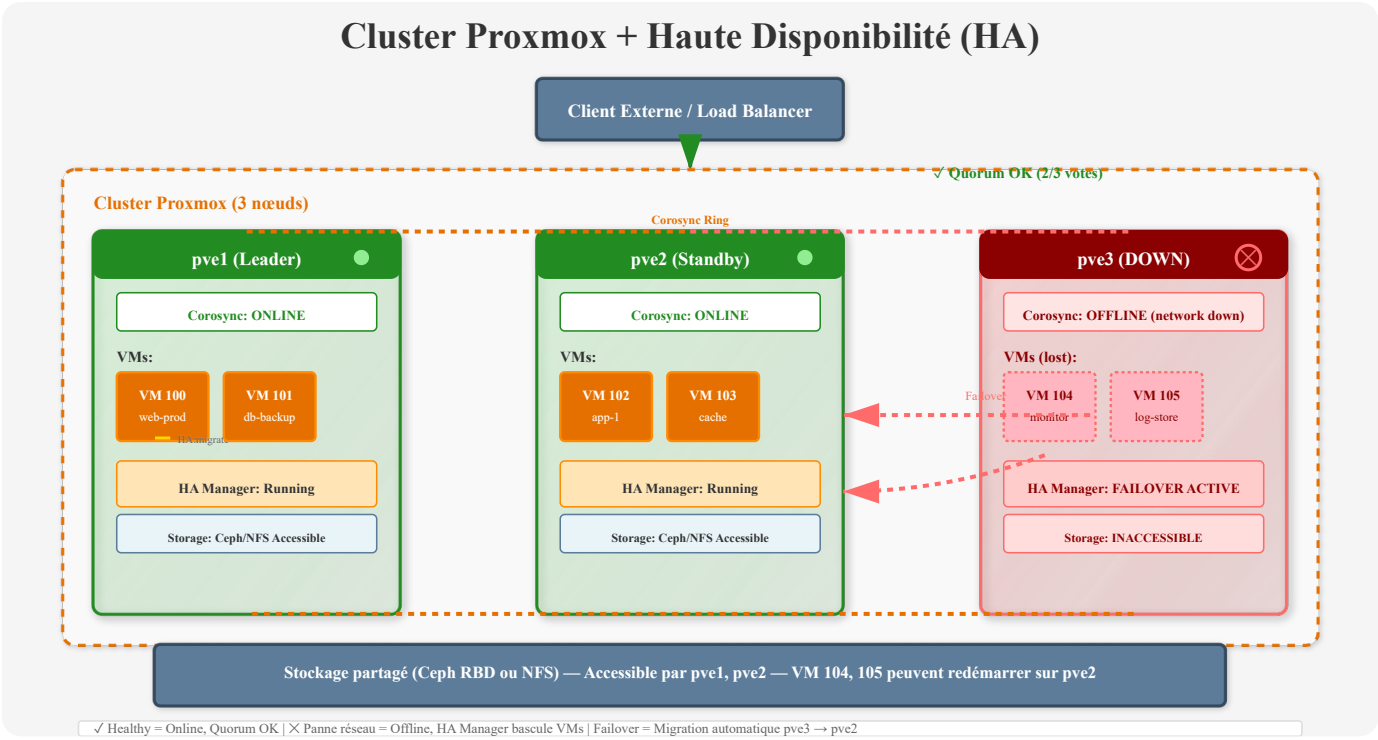
Disponibilité	Temps d'arrêt annuel
99 %	3,65 jours
99,9 %	8,76 heures
99,99 %	52,56 minutes
99,999 %	5,26 minutes
99,9999 %	31,5 secondes
99,99999 %	3,15 secondes

Proxmox VE intègre un gestionnaire de haute disponibilité, `ha-manager`. Il détecte la panne d'un nœud et redémarre les VM et conteneurs qu'il protège sur un autre nœud du cluster.

La documentation officielle donne un ordre de grandeur honnête : **environ 2 minutes** entre la panne et le redémarrage du service. Une seule panne par an consomme donc déjà l'essentiel du budget de 99,999 % — c'est le plafond atteignable avec ce mécanisme.

> **La HA n'est pas de la continuité sans coupure.** Sur une panne de nœud, la VM **redémarre** ailleurs, à froid : ce qui était en mémoire est perdu, les connexions ouvertes sont coupées. Seule une maintenance planifiée (§7.1) déplace les VM à chaud, sans interruption.

Dawan
Ce support est la propriété de Dawan France.



6.2 Les prérequis

Quatre conditions doivent être réunies avant d'activer la HA. Les trois premières sont indispensables.

Prérequis	Pourquoi
Au moins 3 nœuds	Le quorum Corosync exige une majorité absolue (§1.1). Avec deux nœuds, la perte de l'un laisse l'autre minoritaire : il ne peut plus rien décider, donc plus rien relancer.
Un stockage partagé	Relancer une VM sur un autre nœud suppose que ce nœud **voie son disque** : Ceph, NFS ou iSCSI. Sur un stockage local, le disque est resté sur la machine tombée.
De la redondance matérielle	Alimentations et liens réseau doublés, mémoire ECC (*Error-Correcting Code*, qui corrige les erreurs de bits). La HA logicielle relance les services ; elle ne compense pas un matériel qui tombe en cascade.
Un watchdog	Il garantit qu'un nœud isolé s'arrête vraiment avant qu'on relance ses VM ailleurs (§6.4). Proxmox utilise par défaut le watchdog logiciel du noyau, `softdog`.

> **La HA exige un stockage partagé.** C'est Ceph, déployé aux sections 3 à 5, qui le fournit : la VM que l'atelier 6 fait basculer est celle que vous avez posée sur le pool Ceph (§4).

Ce support est la propriété de Dawan France.

La HA repose sur deux démons, le CRM (*Cluster Resource Manager*) et le LRM (*Local Resource Manager*).

6.3 Fonctionnement interne : le CRM décide, le LRM exécute

Deux démons tournent en permanence sur **chaque** nœud du cluster.

Démon	Nom complet	Rôle
`pve-ha-crm`	*Cluster Resource Manager*	Prend les décisions pour tout le cluster : surveille l'état des nœuds, détecte les pannes, choisit où relancer. **Un seul CRM est maître** à un instant donné ; les autres sont en veille.
`pve-ha-lrm`	*Local Resource Manager*	Exécute localement les ordres du CRM : démarrer, arrêter, migrer une VM ou un conteneur de son nœud, puis remonter le résultat.

L'unicité du maître ne repose pas sur une élection réseau, mais sur des **verrous** du système de fichiers de cluster pmxcfs (`/etc/pve`). Le CRM qui détient le verrou `ha_manager_lock` est le maître ; chaque LRM actif détient le verrou de son nœud, `ha_agent_<nœud>_lock` . Ces verrous sont visibles dans `/etc/pve/priv/lock/` .

Dans le Shell du nœud :

```
# Les verrous HA, dans le système de fichiers du cluster
ls /etc/pve/priv/lock/
```

Résultat attendu (cluster de l'atelier 6, VM 300 sous HA) :

```
ha_agent_pxe211_lock
ha_agent_pxe212_lock
ha_manager_lock
```

Ces verrous garantissent qu'un seul CRM décide et qu'un seul LRM agit par nœud. Ils sont aussi la clé du fencing. Si le maître tombe, un autre CRM acquiert `ha_manager_lock` et prend la relève.

Si un nœud tombe, c'est la **prise de son verrou d'agent** par le CRM qui l'autorise à voler ses services. Le journal du CRM montre cette prise de fonction au démarrage.

Dans le Shell du nœud :

Ce support est la propriété de Dawan France.

```
# Journal du CRM (extrait)
journalctl -u pve-ha-crm
```

Résultat attendu (`pxe213` , devenu maître au redémarrage du cluster) :

```
pve-ha-crm[1412]: successfully acquired lock 'ha_manager_lock'
pve-ha-crm[1412]: watchdog active
pve-ha-crm[1412]: status change slave => master
```

`ha-manager status` résume l'ensemble. Sortie du lab, la VM 300 de l'atelier 6 placée sous HA.

Dans le Shell du nœud :

```
# Résumé de la HA : quorum, maître, LRM, ressources
ha-manager status
```

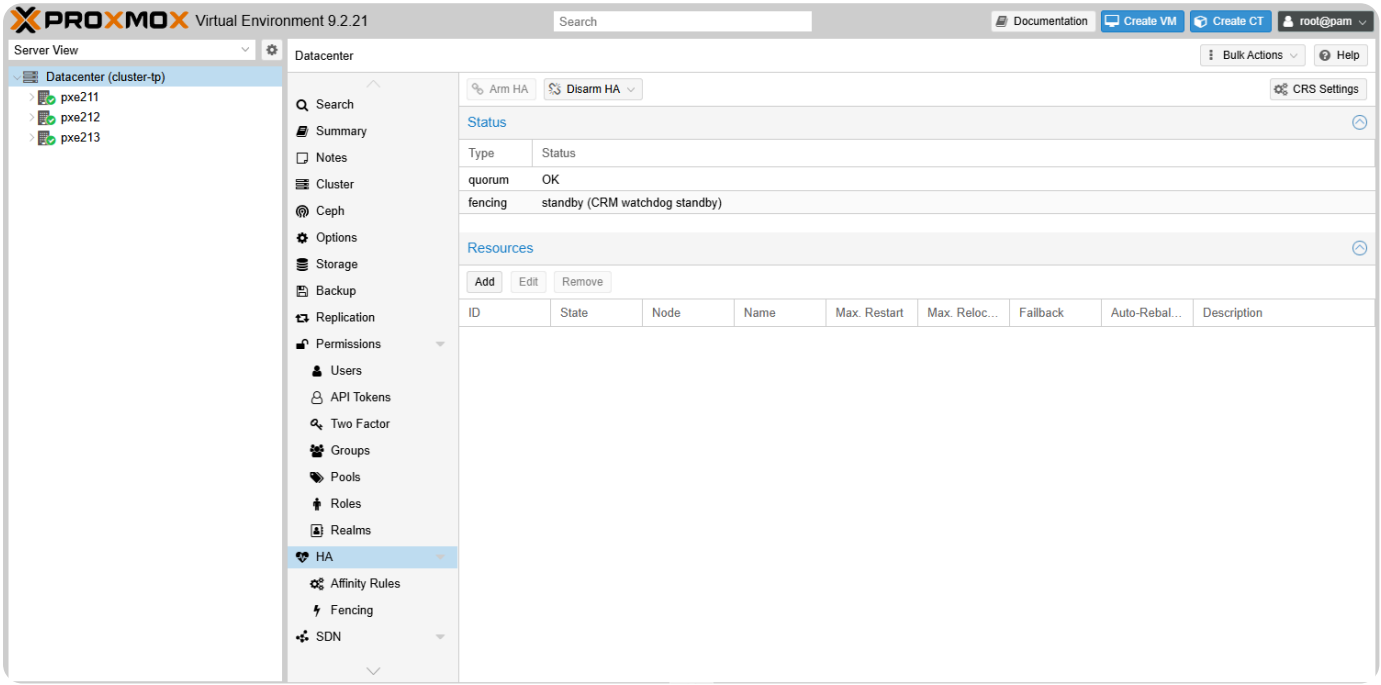
```
quorum OK
master pxe213 (active, Wed Oct  7 18:13:02 2026)
fencing armed (CRM watchdog active)
lrm pxe211 (active, watchdog active, Wed Oct  7 18:13:07 2026)
lrm pxe212 (idle, watchdog standby, Wed Oct  7 18:13:09 2026)
lrm pxe213 (idle, watchdog standby, Wed Oct  7 18:13:08 2026)
service vm:300 (pxe211, started)
```

Ligne	Lecture
`quorum OK`	Le nœud interrogé fait partie de la partition majoritaire : les décisions HA sont possibles.
`master pxe213 (active, …)`	Le CRM maître, et l'heure de son dernier signe de vie.
`fencing armed (CRM watchdog active)`	Un maître gère au moins une ressource : son watchdog est armé.
`lrm … (active, watchdog active, …)`	Le LRM a pris son verrou et arme son watchdog : le nœud a (ou a eu récemment) une ressource HA.
`lrm … (idle, watchdog standby, …)`	Le LRM n'a aucune ressource HA à gérer : pas de watchdog armé.

Dawan
Ce support est la propriété de Dawan France.

Ligne	Lecture
`service vm:300 (pxe211, started)`	La ressource, le nœud qui l'héberge, son état courant (§6.6).

> **La ligne `fencing` est apparue avec Proxmox VE 9.2.** Sans ressource HA, elle vaut `fencing standby (CRM watchdog standby)` : aucun CRM ne gère activement de service. Elle passe à `armed` dès qu'un maître prend en charge une ressource, comme ci-dessus.



La même chose dans **Datacenter → HA**, sur le cluster du lab avant l'atelier 6 : `quorum OK`, puis `fencing standby (CRM watchdog standby)`, car aucune ressource n'est encore sous HA (la liste **Resources** est vide). Le bouton **Add** de cette liste est celui du § 6.5.

6.4 Le fencing et le watchdog

Le **fencing** (« mise à l'écart ») est le mécanisme qui **isole définitivement un nœud défaillant avant de relancer ses services ailleurs**. Il existe pour empêcher le pire scénario d'un cluster : deux instances de la même VM qui écrivent en même temps sur le même disque partagé. Le résultat est une corruption garantie du système de fichiers de la VM.

Le cas difficile n'est pas le nœud mort, mais le nœud **à moitié vivant**. Il ne répond plus sur le réseau du cluster : Corosync ne le voit plus. Mais son processeur tourne toujours, et ses VM aussi.

Ce support est la propriété de Dawan France.

Le CRM voudrait relancer ces VM ailleurs ; s'il le fait alors que le nœud continue d'écrire sur Ceph, c'est le split-brain. Ce nœud ne peut pas libérer son verrou lui-même, et personne ne peut lui demander de s'arrêter : il n'est plus joignable.

C'est le rôle du **watchdog**, un minuteur que le nœud doit réarmer régulièrement faute de quoi il redémarre de force. Le guide d'administration le décrit ainsi. Un nœud qui constate qu'il n'est plus dans la partition quorate ne peut plus réarmer son watchdog.

S'il héberge des services actifs, le watchdog expire et **redémarre le nœud au bout de 60 secondes**. Le CRM, de son côté, sait qu'au-delà de ce délai le nœud isolé est forcément arrêté : il peut prendre son verrou et relancer ses services sans risque.

La séquence, dans l'ordre :

1. Le nœud est isolé : réseau du cluster coupé, processeur toujours actif.
2. Hors quorum, son LRM ne peut plus réarmer le watchdog.
3. Le watchdog expire au bout de 60 secondes : le nœud redémarre de force.
4. Le CRM acquiert le verrou du nœud : ses services passent en `recovery`.
5. Les VM redémarrent sur un nœud sain.

Sur le lab, pendant l'atelier 6, ces étapes se lisaient dans le journal du nouveau maître : `fencing: acknowledged - got agent lock for node 'pxe212'`, puis `recover service 'vm:300' from fenced node 'pxe212' to node 'pxe213'`.

Deux familles de watchdog existent :

	`softdog` **(logiciel, par défaut)**	**Watchdog matériel (IPMI, iDRAC, iTCO…)**
Disponible sans configuration	Oui	Non : module à déclarer
Résiste à un *kernel panic*	Non	Oui
Résiste à un gel complet du processeur	Non	Oui
Usage	Laboratoire, infrastructure non critique	Production avec stockage partagé

`softdog` tourne dans le noyau qu'il surveille : si ce noyau est figé, il ne peut plus rien déclencher. Un watchdog matériel est un circuit indépendant de la carte mère, qui coupe le serveur quoi qu'il arrive au système. Tous les modules de watchdog matériel sont bloqués par défaut, car un watchdog mal initialisé peut redémarrer un serveur sain.

Ce support est la propriété de Dawan France.

Pour en activer un, on déclare son module dans `/etc/default/pve-ha-manager`, lu par le service `watchdog-mux`. Le fichier livré avec Proxmox VE 9.2 contient ceci.

Dans le Shell du nœud :

```
# Le fichier lu par watchdog-mux
cat /etc/default/pve-ha-manager
```

```
# select watchdog module (default is softdog)
#WATCHDOG_MODULE=ipmi_watchdog
```

> **Ne jamais arrêter `pve-ha-crm`, `pve-ha-lrm` ou `watchdog-mux` à la main** sur un nœud qui héberge des ressources HA. Ces services tiennent le watchdog ouvert : les tuer revient à cesser de le réarmer, et le nœud redémarre brutalement. Le guide d'administration le signale explicitement.

Le mode de fencing du cluster se règle dans `datacenter.cfg` (`fencing: watchdog`, valeur par défaut). Les valeurs `hardware` et `both` pilotent des équipements de coupure externes, décrits dans `/etc/pve/ha/fence.cfg`. La documentation de la version 9.2 les marque **expérimentales** : on s'en tient au watchdog.

Revenons à la définition de base : la **haute disponibilité** relance un service ailleurs quand son serveur tombe, sans intervention humaine. Le danger est de le relancer alors que l'original tourne encore. Les composants du mécanisme :

Terme	Ce que c'est	Dans le laboratoire
Ressource HA	Une VM ou un conteneur placé sous la surveillance du gestionnaire HA	`vm:300` (`vm-ha-test`)
CRM	`pve-ha-crm` : décide pour tout le cluster ; un maître, les autres en veille	La ligne `master` de `ha-manager status`
LRM	`pve-ha-lrm` : exécute sur son nœud les ordres du CRM	Une ligne `lrm` par nœud
Verrou	Un verrou de `/etc/pve` : `ha_manager_lock` pour le maître, `ha_agent_<nœud>_lock` pour chaque LRM	`/etc/pve/priv/lock/`

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
Watchdog	Un minuteur que le nœud réarme ; hors quorum, il ne peut plus, et le nœud redémarre au bout de 60 secondes	`softdog`, par défaut
Fencing	Mettre le nœud défaillant hors jeu avant de relancer ses services	`fencing: acknowledged - got agent lock for node 'pxe212'`
Recovery	La relance des services sur un nœud sain	`recover service 'vm:300' … to node 'pxe213'`
Watchdog matériel	Un circuit de la carte mère, qui coupe même un système figé	Non utilisé au laboratoire

Le cas délicat est le nœud coupé du réseau du cluster, dont les VM tournent toujours. Le CRM attend que le watchdog ait redémarré ce nœud, puis prend son verrou d'agent : seulement alors, il relance ses services sur un nœud sain.

Où voir la haute disponibilité sur votre cluster :

Notion	Où la voir
Le maître, les LRM, les ressources	**Datacenter** → **HA** ; `ha-manager status`
Les verrous	`ls /etc/pve/priv/lock/`
Les décisions du CRM (fencing, recovery)	`journalctl -u pve-ha-crm` sur le nœud maître
Le watchdog	`/etc/default/pve-ha-manager` (module), service `watchdog-mux`
Le mode de fencing	`/etc/pve/datacenter.cfg` (`fencing: watchdog` par défaut)

6.5 Les ressources HA

Une **ressource HA** est une VM ou un conteneur confié au `ha-manager`. Elle est identifiée par un **SID** (*Service ID*) de la forme `type:id` : `vm:100` pour la VM 100, `ct:200` pour le conteneur 200. Les ressources sont stockées dans `/etc/pve/ha/resources.cfg`, répliqué sur tous les nœuds par pmxcfs.

Activer la HA pour une VM — via l'interface graphique :

> **Datacenter → HA → Resources → Add**

Ce support est la propriété de Dawan France.

Champ	Valeur
VM	`300` (sur Ceph, atelier 5, placée sous HA à l'atelier 6)
Max. Restart	`1`
Max. Relocate	`1`
Request State	`started`

Équivalent CLI :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Placer la VM 300 sous la protection de la HA : "started" = toujours la maintenir démarrée
ha-manager add vm:300 --state started --max_restart 1 --max_relocate 1
```

Après `ha-manager add vm:950 --state started --max_restart 1 --max_relocate 1` sur le cluster d'exemple, le fichier contient exactement :

```
vm: 950
    max_relocate 1
    max_restart 1
    state started
```

Les propriétés d'une ressource, telles que les décrit `ha-manager help add` sur Proxmox VE 9.2 :

Propriété	Rôle	Défaut
`state`	État demandé : `started`, `stopped`, `disabled`, `ignored` (`enabled` est un alias de `started`)	`started`
`max_restart`	Nombre de nouvelles tentatives de démarrage **sur le même nœud** après un échec	`1`
`max_relocate`	Nombre de déplacements **vers un autre nœud** après échec des redémarrages	`1`
`failback`	Ramener automatiquement la ressource sur le nœud de plus haute priorité de sa règle d'affinité quand il revient	`1` (activé)
`auto-rebalance`	Autoriser la ressource à être déplacée par le rééquilibrage automatique	`1`
`comment`	Description libre	—

Ce support est la propriété de Dawan France.

Propriété	Rôle	Défaut
`group`	Ancien groupe HA — **déprécié** au profit des règles d'affinité	—

:::tip **Une ressource HA se pilote comme une VM ordinaire.** Les commandes `qm start` et `qm stop`, comme les boutons de l'interface, sont transmises au gestionnaire HA quand la VM en est une ressource. Sur le cluster d'exemple, `qm start 950` répond simplement `Requesting HA start for VM 950` : c'est le CRM qui décide ensuite du nœud et lance le démarrage. :::

6.6 Les états d'une ressource

Le CRM tient pour chaque ressource un **état courant**, distinct de l'état demandé. C'est lui qu'affichent `ha-manager status` et **Datacenter → HA**. La liste est celle du guide d'administration :

État	Signification
`queued`	Ressource tout juste ajoutée, que le CRM n'a pas encore vue
`started`	Active : le LRM la démarre si elle ne tourne pas, la redémarre si elle s'arrête
`starting`	Démarrage demandé, le CRM attend la confirmation du LRM
`request_stop`	Arrêt demandé, le CRM attend la confirmation du LRM
`stopping`	Arrêt en attente, pas encore transmis
`stopped`	Arrêtée, confirmé par le LRM ; si elle redémarre seule, le LRM l'arrête de nouveau
`migrate`	Migration à chaud en cours vers un autre nœud
`fence`	Le nœud hôte est hors de la partition quorate : attente de son fencing
`recovery`	Nœud isolé avec succès : recherche d'un nouveau nœud
`freeze`	Ne pas toucher : nœud en cours de redémarrage ou LRM en cours de mise à jour
`ignored`	Traitée comme si elle n'était pas sous HA, sans la retirer de la configuration
`error`	Échec de la politique de récupération : **intervention manuelle requise**
`disabled`	Arrêtée et désactivée

Un état supplémentaire, `deleting`, apparaît brièvement quand une ressource est retirée par `ha-manager remove`, `service vm:950 (pxe213, deleting)`, affiché pendant plusieurs secondes.

6.7 Commandes et fichiers à connaître

Action	Commande
Placer une VM sous HA	`ha-manager add vm:300 --state started`

Ce support est la propriété de Dawan France.

Action	Commande
Changer l'état demandé	`ha-manager set vm:300 --state stopped`
Lister les ressources	`ha-manager config`
État global	`ha-manager status` (`--verbose 1` pour le détail JSON)
Migrer à chaud, VM démarrée	`ha-manager crm-command migrate vm:300 pxe212` (alias : `ha-manager migrate`)
Déplacer par arrêt puis redémarrage	`ha-manager crm-command relocate vm:300 pxe212` (alias : `ha-manager relocate`)
Gérer les règles	`ha-manager rules add / list / set / remove`
Sortir une ressource de la HA	`ha-manager remove vm:300`
Suivre les décisions	`journalctl -u pve-ha-crm -f` (maître) et `journalctl -u pve-ha-lrm -f` (chaque nœud)

Fichier	Contenu
`/etc/pve/ha/resources.cfg`	Les ressources HA et leurs propriétés
`/etc/pve/ha/rules.cfg`	Les règles d'affinité (Proxmox VE 9)
`/etc/pve/ha/groups.cfg`	Les anciens groupes (version 8, convertis à la mise à niveau)
`/etc/pve/ha/manager_status`	L'état courant tenu par le CRM, lu par les LRM
`/etc/pve/nodes/<nœud>/lrm_status`	Ce que chaque LRM remonte au CRM
`/etc/default/pve-ha-manager`	Le module de watchdog à charger

> **Points clés à retenir.** La HA exige au moins trois nœuds, un stockage partagé et un watchdog. Le CRM décide, le LRM exécute ; les verrous de pmxcfs garantissent un seul maître.
>
> Le fencing, par le watchdog, isole un nœud avant de relancer ses services — compter environ 2 minutes. Une ressource qui échoue partout passe en `error` et attend l'administrateur : `disabled` , corriger, `started` . Depuis Proxmox VE 9, les groupes sont remplacés par les règles d'affinité.

- 📖 Documentation Proxmox — High Availability
- 📖 Les règles HA dans le guide d'administration
- 📖 Options du fichier datacenter.cfg (CRS — *Cluster Resource Scheduling* —, fencing, politique d'arrêt)

Dawan
Ce support est la propriété de Dawan France.

- 🎥 Simulation de panne d'un nœud et basculement automatique des VM HA (vidéo)

> **▶ À pratiquer maintenant — Atelier 6**
>
> Inscrire la VM 300, posée sur Ceph, comme ressource HA, couper brutalement son nœud, et chronométrer le cluster qui la relance ailleurs. Puis mettre un nœud en maintenance et voir la VM partir et revenir à chaud.

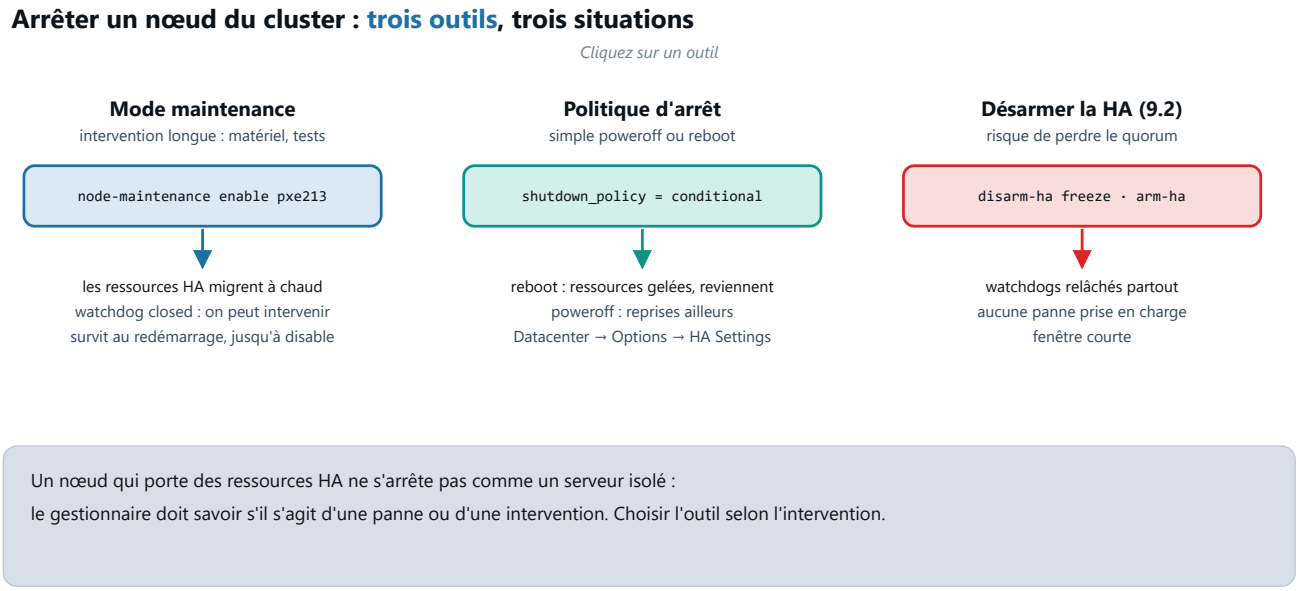
Ce support est la propriété de Dawan France.

7. Maintenance d'un nœud et mises à jour

Un nœud de cluster doit parfois s'arrêter : changer une pièce, appliquer une mise à jour. Tant qu'il porte des machines, l'arrêter les interromprait. Proxmox sait le vider d'abord, en migrant ses machines à chaud vers les autres nœuds.

7.1 Mode maintenance et politique d'arrêt

Un nœud qui porte des ressources HA ne s'arrête pas comme un serveur isolé : le gestionnaire doit savoir s'il s'agit d'une panne ou d'une intervention. Trois outils répondent à trois situations.



Un nœud qui porte des ressources HA ne s'arrête pas comme un serveur isolé :
le gestionnaire doit savoir s'il s'agit d'une panne ou d'une intervention. Choisir l'outil selon l'intervention.

Les trois outils de cette sous-section, avec la situation à laquelle chacun répond et son effet. Cliquez sur un outil pour la commande et ce qu'elle affiche.

Le mode maintenance vide un nœud de ses ressources HA pour une intervention qui ne se résume pas à un seul redémarrage (remplacement de matériel, tests). La commande peut être lancée depuis **n'importe quel nœud** : elle est mise en file d'attente et traitée par le CRM.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
ha-manager crm-command node-maintenance enable pxe213   # mettre pxe213 en
maintenance : ses ressources HA partent
```

Ce support est la propriété de Dawan France.

```
ha-manager crm-command node-maintenance disable pxe213   # sortir pxe213 de
maintenance
```

Sur le lab, avec la VM de test 950 sous HA sur `pxe213`. Après `enable`, le CRM migre la ressource **à chaud** :

```
pve-ha-crm[1412]: node 'pxe213': state changed from 'online' => 'maintenance'
pve-ha-crm[1412]: migrate service 'vm:950' to node 'pxe212' (running)
pve-ha-crm[1412]: service 'vm:950': state changed from 'started' to 'migrate'  (node
= pxe213, target = pxe212)
pve-ha-crm[1412]: service 'vm:950': state changed from 'migrate' to 'started'  (node
= pxe212)
```

Le LRM du nœud signale qu'il a fini de se vider en fermant son watchdog — c'est la ligne à guetter avant d'intervenir : `pve-ha-lrm[1931]: watchdog closed (disabled)`, 30 secondes après la commande. `ha-manager status` affiche alors `lrm pxe213 (maintenance mode, watchdog active, …)`. Après `disable`, la ressource **revient d'elle-même** sur son nœud d'origine :

```
pve-ha-crm[1412]: node 'pxe213': state changed from 'maintenance' => 'online'
pve-ha-crm[1412]: moving service 'vm:950' back to 'pxe213', node came back from
maintenance.
pve-ha-crm[1412]: migrate service 'vm:950' to node 'pxe213' (running)
pve-ha-crm[1412]: service 'vm:950': state changed from 'migrate' to 'started'  (node
= pxe213)
```

Sur le lab, le retour a pris 39 secondes après la commande `disable` : le LRM doit d'abord reprendre son verrou d'agent (`successfully acquired lock 'ha_agent_pxe213_lock'`, 9 secondes après). Il faut attendre, pas relancer la commande. Le mode maintenance **survit à un redémarrage** du nœud : il ne disparaît que par `disable`.

La politique d'arrêt (`shutdown_policy`) dit ce que fait le gestionnaire quand on éteint ou redémarre un nœud sans mode maintenance. Elle se règle dans **Datacenter → Options → HA Settings** :

Valeur	Comportement
`conditional` (défaut)	Sur `poweroff` : les ressources sont arrêtées, puis reprises par d'autres nœuds. Sur `reboot` : elles sont gelées (`freeze`) et redémarrent sur le même nœud à

Ce support est la propriété de Dawan France.

Valeur	Comportement
	son retour
`migrate`	Le LRM se déclare indisponible et fait migrer toutes ses ressources avant de s'éteindre ; elles reviennent au retour du nœud
`failover`	Les ressources sont arrêtées, et reprises ailleurs si le nœud ne revient pas rapidement
`freeze`	Les ressources sont arrêtées et gelées jusqu'au retour du nœud

C'est pourquoi l'atelier de HA (§6, atelier 6) simule la panne par une coupure brutale, et **non** par `poweroff`. Un arrêt propre est traité par cette politique, pas par le mécanisme de reprise sur panne.

Désarmer la HA (Proxmox VE 9.2) sert aux interventions qui risquent de faire perdre le quorum : une reconfiguration de Corosync ou du réseau du cluster, par exemple. `ha-manager crm-command disarm-ha freeze` (ou `ignore`) relâche tous les watchdogs du cluster.

`ha-manager crm-command arm-ha` les réarme. Pendant ce temps, **aucune panne n'est prise en charge** : la fenêtre doit rester courte.

7.2 Les mises à jour en cluster

En cluster, on met à jour un nœud à la fois. On le vide de ses VM (migration, mode maintenance HA), on le met à jour, on le redémarre, puis on passe au suivant. C'est le *rolling update*.

Avec Ceph, on pose d'abord `ceph osd set noout`, pour que Ceph ne recopie pas les données d'un OSD absent quelques minutes.

> **Ne jamais redémarrer deux nœuds simultanément** dans un cluster à 3 nœuds. Avec deux nœuds en cours de redémarrage, le troisième n'a plus le quorum (1 vote sur 3) et passe en lecture seule : toutes les VM sont figées.

À retenir : on applique les mises à jour de sécurité sans attendre. En cluster, **un nœud à la fois**, vidé de ses VM, mis à jour, redémarré, puis le suivant. Jamais deux nœuds redémarrés en même temps sur trois ; avec Ceph, on pose `noout`.

Ce support est la propriété de Dawan France.

Ateliers — chapitre 4

Infrastructure : 3 nœuds — `pxe2<N>1` (le vôtre, depuis le chapitre 1), `pxe2<N>2` et `pxe2<N>3`, joignables par leurs URL, `https://pxe2<N>1.<ville>.pm.dawan.ovh` et suivantes, données par le formateur. Sur chacun : disque `/dev/sdb` (70 Go) réservé à Ceph, et cinq cartes réseau.

Réseaux internes (stagiaire N ; site 44 Nantes, 75 Paris, 67 Strasbourg) : `ens19` → cluster Corosync `10.<site>.<N+10>.x` ; `ens20` → Ceph cluster `10.<site>.<N+20>.x` ; `ens21` → Ceph public `10.<site>.<N+30>.x` ; `ens22` → migration `10.<site>.<N+40>.x`. Nœud 1 en `.11`, nœud 2 en `.12`, nœud 3 en `.13`. Ces réseaux ne sortent pas : ni passerelle, ni NAT.

Les captures et les sorties proviennent du lab d'exemple, monté comme le vôtre : site `59`, stagiaire `1`, Proxmox VE 9.2.21. Ses adresses suivent donc le même plan : `vmbr0` en `10.59.1.x`, Corosync en `10.59.11.x`, et ainsi de suite. Remplacez `59` par votre site et `1` par votre numéro.

État du lab en fin de chapitre :

- Cluster `cluster-tp` de 3 nœuds, quorate, Link 0 sur `vmbr1` et Link 1 sur `vmbr0`. VM 120 migrée à froid, puis à chaud avec ses disques locaux, et éteinte sur `pxe2<N>2` (ateliers 2 et 3)
- Migration d'un conteneur en `--restart` mesurée, perte et retour du quorum éprouvés (ateliers 3 et 4) ; CT 330 supprimé
- Réseaux internes adressés : `ens20` (Ceph cluster), `ens21` (Ceph public), `ens22` (migration, déclaré dans Datacenter → Options)
- Ceph Tentacle : 3 MON, 3 MGR, 3 OSD sur `sdb`, pool `ceph-vm` (3/2) déclaré en stockage ; VM 300 `vm-ha-test` dessus, migrée à chaud
- (Atelier 6) VM 300 en ressource HA, basculement automatique constaté après une panne brutale ; maintenance du nœud 1 éprouvée

---

Préparation à la mise en cluster

Ce support est la propriété de Dawan France.

Contexte

Votre nœud `pxe2<N>1` va **créer** le cluster ; `pxe2<N>2` et `pxe2<N>3`, livrés pour ce chapitre, vont le **rejoindre**. Avant de les assembler, deux préparations :

- **Nettoyer `pxe2<N>1` .** À la création du cluster, la configuration de `/etc/pve` devient commune aux trois nœuds : chaque stockage déclaré sur le vôtre sera proposé aux deux autres. Les objets de test des chapitres précédents occupent aussi la mémoire et le disque dont Ceph aura besoin.
- **Mettre `pxe2<N>2` et `pxe2<N>3` à jour.** Ils sortent d'installation, comme votre premier nœud au chapitre 1 : dépôts **enterprise** actifs, `apt update` en erreur, aucune mise à jour. Les trois nœuds d'un cluster doivent tourner sur la même version.

Résultat à obtenir : `pxe2<N>1` sans objet de test, `zfs-vms` limité à ce nœud, tous ses stockages `active`. `pxe2<N>2` et `pxe2<N>3` sur le dépôt no-subscription, à jour, vides de machines. Et la même version de Proxmox VE sur les trois.

Ce que vous devez faire

1. **Sur `pxe2<N>1`** : supprimer les objets de test encore présents, limiter le stockage `zfs-vms` à ce nœud, vérifier que tous les stockages sont `active`.
2. **Sur `pxe2<N>2`, puis sur `pxe2<N>3`**, chacun par sa propre interface web : désactiver les deux dépôts enterprise, ajouter le dépôt no-subscription, mettre à jour, redémarrer si le noyau a changé.
3. **Vérifier** : même version sur les trois nœuds, aucune machine sur `pxe2<N>2` et `pxe2<N>3`.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Nettoyer `pxe2<N>1`

Ce qui doit avoir disparu	Pourquoi
Les objets de test : VM 140 (chapitre 3, atelier 4), CT 260 (atelier 5), CT 311 et 312 (atelier 2)	Ils occupent de la mémoire et du disque dont Ceph aura besoin
Tout stockage dont le serveur n'existe plus (`nfs-partage`, chapitre 3, atelier 5)	Il reste `inactive` dans `pvesm status`, et sa déclaration deviendrait commune aux trois nœuds
Les snapshots de test sur les machines qui restent	Ils retiennent de l'espace dans le stockage (chapitre 3, §4.7)

Ce qui reste	Pourquoi
Le pont `vmbr1` sur `ens19`, avec son adresse	C'est le lien Corosync du cluster (atelier 1)
Le CT 200 et la VM vide 290 du chapitre 1, les VM 100 à 110 du chapitre 2	Elles servent encore ; la VM 290 porte le droit du comptable
Le pool `datapool` et le stockage `zfs-vms`	Ils peuvent rester, **limités à** `pxe2<N>1`
Le pont `vmbr9`, isolé	Sans port ni adresse, il ne gêne rien

Dans le Shell de `pxe2<N>1` (interface web de `pxe2<N>1` → le nœud `pxe2<N>1` → **Shell**) :

```
qm list          # les VM du nœud : il ne doit rester que 100 à 110, et la 290
pct list         # les conteneurs : il ne doit rester que le CT 200
pvesm status     # tous les stockages en active, plus de nfs-partage
```

Supprimez ce qui dépasse : la machine → **More** → **Remove**, ou `qm destroy <vmid> --purge` et `pct destroy <ctid> --purge`. Un stockage en trop : **Datacenter** → **Storage** → le stockage → **Remove**, ou `pvesm remove <id>`.

Limiter `zfs-vms` à votre nœud. Le pool `datapool` n'existe que sur `pxe2<N>1`. Déclaré pour tous les nœuds (**Nodes** : *All (No restrictions)*), `zfs-vms` serait proposé à `pxe2<N>2` et `pxe2<N>3`, qui n'ont pas ce pool.

Via l'interface : Datacenter → **Storage** → `zfs-vms` → **Edit** → **Nodes** : `pxe2<N>1` → **OK**.

Équivalent CLI, dans le Shell de `pxe2<N>1` :

Ce support est la propriété de Dawan France.

```
pvesm set zfs-vms --nodes pxe2<N>1                # le stockage n'existe que sur ce nœud
grep -A6 "zfspool: zfs-vms" /etc/pve/storage.cfg   # la ligne nodes apparaît
```

Résultat attendu (`pxe211`) :

```
zfspool: zfs-vms
    pool datapool/vms
    content rootdir,images
    mountpoint /datapool/vms
    nodes pxe211
    sparse 1
```

La ligne `nodes pxe211` limite le stockage à ce nœud. `sparse 1` est la case **Thin provision** du stockage ; elle est absente si elle n'a pas été cochée.

Étape 2 — Dépôts et mise à jour de `pxe2<N>2` , puis de `pxe2<N>3`

C'est la procédure du chapitre 1, atelier 3, sur chacun des deux nouveaux nœuds. Ils ne sont pas encore en cluster : chacun s'administre par **sa propre interface web**, avec son URL.

> Les sorties ci-dessous sont celles du chapitre 1, sur `pxe211` juste après son installation : les nœuds `pxe2<N>2` et `pxe2<N>3` sortent de la même installation. Le nombre de paquets à mettre à jour et les versions dépendent du jour.

1. Lire l'état de départ. Interface web de `pxe2<N>2` → le nœud → **Updates** → **Repositories**.

Le bandeau **Status** avertit : *The enterprise repository is enabled, but there is no active subscription!* Deux dépôts Proxmox sont cochés : `pve-enterprise` (fichier `pve-enterprise.sources`) et `ceph-squid` en `enterprise` (fichier `ceph.sources`). Tous deux pointent vers `enterprise.proxmox.com` , réservé aux abonnés.

2. Désactiver les deux dépôts enterprise. Sélectionnez la ligne `https://enterprise.proxmox.com/debian/pve` → **Disable**.

Puis la ligne `https://enterprise.proxmox.com/debian/ceph-squid` → **Disable**. Oublier la seconde laisse `apt update` en erreur.

Ce support est la propriété de Dawan France.

Dans le Shell de `pxe2<N>2` :

```
apt update   # relire les dépôts
```

Résultat attendu, si `ceph-squid` est resté en `enterprise` :

```
Err:6 https://enterprise.proxmox.com/debian/ceph-squid trixie InRelease
  401  Unauthorized [IP: 51.91.38.34 443]
```

`401 Unauthorized` : le serveur refuse l'accès, faute d'abonnement. **Disable** ne supprime rien : il ajoute la ligne `Enabled: false` au fichier du dépôt.

3. Ajouter le dépôt no-subscription. Même page → **Add** → **Repository** : `No-Subscription` → **Add**.

Équivalent CLI, dans le Shell de `pxe2<N>2` :

```
# Dépôt no-subscription de PVE 9, au format DEB822
nano /etc/apt/sources.list.d/proxmox.sources   # un fichier neuf
```

Le contenu de `proxmox.sources`, à taper dans `nano`, puis **Ctrl + O**, Entrée et **Ctrl + X** :

```
Types: deb
URIs: http://download.proxmox.com/debian/pve
Suites: trixie
Components: pve-no-subscription
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

Dans le Shell de `pxe2<N>2` :

```
cat /etc/apt/sources.list.d/proxmox.sources   # relire le fichier créé
```

Le bandeau passe à *You get updates for Proxmox VE*. Un avertissement jaune s'ajoute : *The no-subscription repository is not recommended for production use!* Ce dépôt convient au lab ; en production, on prend un abonnement.

4. Mettre à jour. Dans le Shell de `pxe2<N>2` :

Dawan Ce support est la propriété de Dawan France.

```
apt update            # relire la liste des paquets : plus aucune erreur 401
apt full-upgrade -y   # appliquer toutes les mises à jour
pveversion            # version de Proxmox VE et noyau en cours
ls /boot/vmlinuz-*    # les noyaux installés
```

Résultat attendu au chapitre 1 (fin de `apt update`) :

```
Get:5 http://download.proxmox.com/debian/pve trixie/pve-no-subscription amd64
Packages [390 kB]
181 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Si `pveversion` affiche un `running kernel` plus ancien que le dernier `vmlinuz` installé, un nouveau noyau attend. Redémarrez alors le nœud (bouton **Reboot** de son interface, ou `reboot` dans son Shell), puis reconnectez-vous.

Recommencez les points 1 à 4 sur `pxe2<N>3`, par son interface web.

> Le dépôt Ceph enterprise reste désactivé, sans dépôt Ceph de remplacement : c'est l'installation de Ceph, à l'atelier 5, qui écrit le bon dépôt.

Étape 3 — Vérifier

Dans le Shell de chacun des trois nœuds :

```
pveversion   # la même version partout
```

Dans le Shell de `pxe2<N>2`, puis de `pxe2<N>3` :

```
qm list    # rien : un nœud qui rejoint un cluster doit être vide de VM
pct list   # et de conteneurs
apt update 2>&1 | grep -i "401\|error"   # rien : plus d'erreur de dépôt
```

Résultat attendu (sur `pxe211`, `pxe212` et `pxe213`) : `pve-manager/9.2.21/4f6e0ac86f9e8c7f (running kernel: 7.0.14-20-pve)` partout.

Ce support est la propriété de Dawan France.

Atelier 1 — Constitution du cluster 3 nœuds

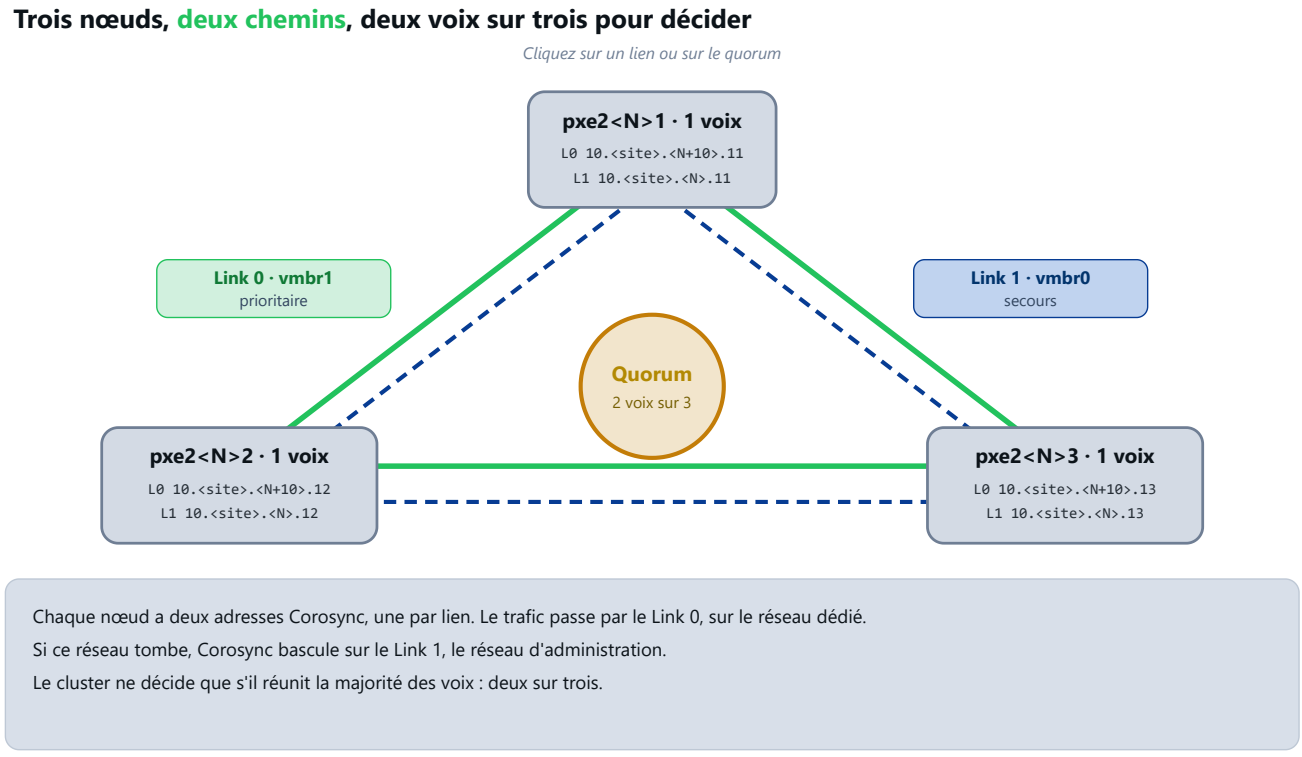
Contexte

Jusqu'ici, chaque nœud vivait seul. Le **cluster** les réunit en une seule entité, administrée depuis n'importe lequel d'entre eux. C'est le socle du chapitre : sans cluster, ni Ceph, ni haute disponibilité, ni migration.

Vous le montez à trois nœuds. C'est le minimum sérieux : à trois, la perte d'un nœud laisse deux voix sur trois, et le quorum tient.

Les nœuds se parlent par **Corosync**. Ce trafic est court mais sensible au délai : on lui donne son propre réseau, `vmbr1`, créé au chapitre 3. Le réseau d'administration, `vmbr0`, sert de lien de secours.

Résultat à obtenir : un cluster `cluster-tp` de trois nœuds, quorate, dont le lien principal passe par `vmbr1`.



Ce que vous allez construire. Chaque nœud a deux adresses Corosync : une sur `vmbr1` (Link 0, prioritaire), une sur `vmbr0` (Link 1, secours).

Ce support est la propriété de Dawan France.

Chaque nœud a une voix ; le cluster décide avec deux voix sur trois. Cliquez sur un lien ou sur le quorum pour lire le résultat obtenu.

Arborescence attendue

```
Cluster « cluster-tp »   (quorum : 2 voix sur 3)
├── pxe2<N>1   ← nœud fondateur, garde ses VM et conteneurs
│   ├── Link 0 : 10.<site>.<N+10>.11  (vmbr1, réseau Corosync)
│   └── Link 1 : 10.<site>.<N>.11     (vmbr0, secours)
├── pxe2<N>2   ← rejoint le cluster, vierge de machines
│   ├── Link 0 : 10.<site>.<N+10>.12
│   └── Link 1 : 10.<site>.<N>.12
└── pxe2<N>3   ← rejoint ensuite, vierge de machines
    ├── Link 0 : 10.<site>.<N+10>.13
    └── Link 1 : 10.<site>.<N>.13
```

Ce que vous devez faire

1. **Préparer le réseau des deux nouveaux nœuds** (la préparation ci-dessus est faite) : pont `vmbr1`. Vérifier que les trois se joignent sur le réseau Corosync.
2. **Créer le cluster** depuis `pxe2<N>1`, avec le Link 0 sur `vmbr1` et le Link 1 sur `vmbr0`.
3. **Faire adhérer** `pxe2<N>2`, puis `pxe2<N>3`, **l'un après l'autre**.
4. **Vérifier** : trois nœuds, trois voix, `Quorate: Yes`, deux liens connectés.

> ⚠️ On joint les nœuds un par un. On attend que le premier soit stable avant de joindre le second.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 0 — Le réseau Corosync des trois nœuds

Les dépôts et les mises à jour sont faits (préparation à la mise en cluster). Il reste à donner aux deux nouveaux nœuds leur accès au réseau Corosync.

1. Pont `vmbr1`, sur `pxe2<N>2` et `pxe2<N>3`. Celui du chapitre 3 n'existe que sur votre premier nœud.

Nœud	`vmbr0` (administration)	`vmbr1` (Corosync, sur `ens19`)
`pxe2<N>1`	`10.<site>.<N>.11`	`10.<site>.<N+10>.11/24` — déjà créé au chapitre 3
`pxe2<N>2`	`10.<site>.<N>.12`	`10.<site>.<N+10>.12/24`
`pxe2<N>3`	`10.<site>.<N>.13`	`10.<site>.<N+10>.13/24`

Avant tout Apply, le filet du chapitre 3 (atelier 1, étape 1 bis).

Dans le Shell de chaque nœud dont vous modifiez le réseau :

```
# Copie de la configuration réseau qui fonctionne, puis retour automatique dans 10 minutes
cp /etc/network/interfaces /root/interfaces.avant
systemd-run --on-active=10min --unit=filet-reseau sh -c "cp /root/interfaces.avant /etc/network/interfaces && ifreload -a"   # si l'accès est perdu, restaurer la copie dans 10 min
```

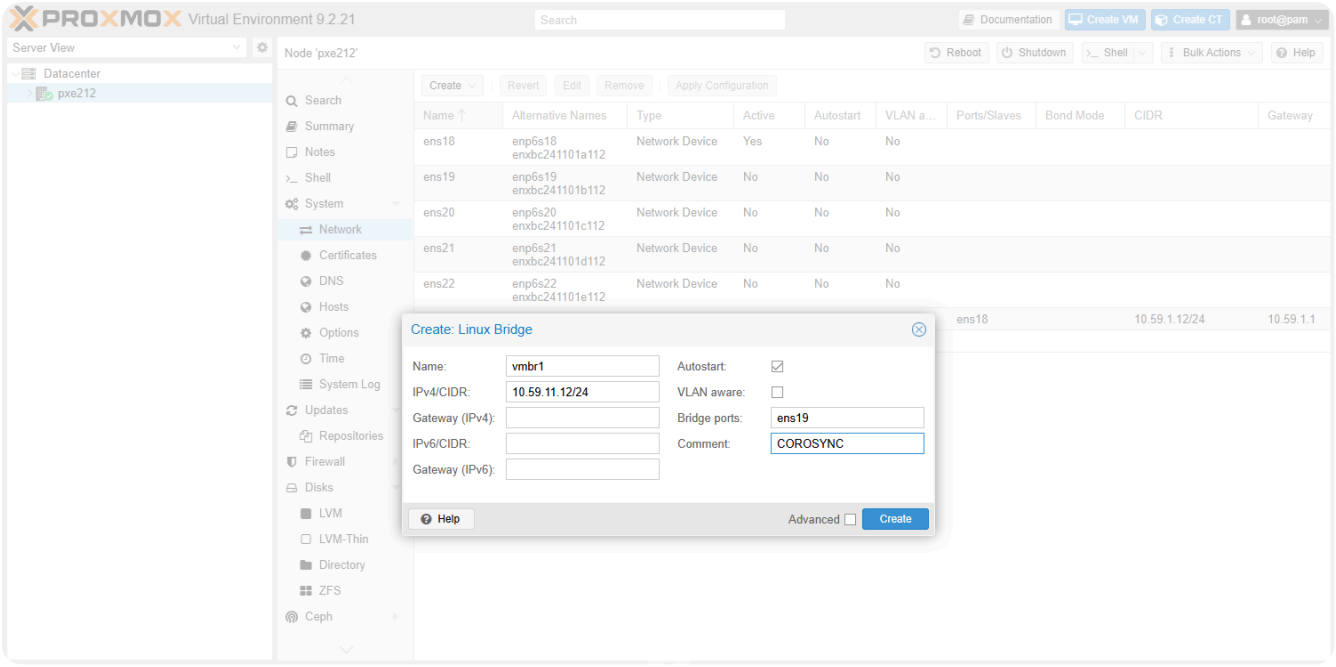
Résultat attendu :

```
Running timer as unit: filet-reseau.timer
Will run service as unit: filet-reseau.service
```

Une fois l'accès et les pings vérifiés, annulez-le sur chacun : `systemctl stop filet-reseau.timer`.

Network → **Create** → **Linux Bridge** : Name `vmbr1`, IPv4/CIDR l'adresse **de ce nœud-là**, Gateway vide, Bridge ports `ens19`, Comment `COROSYNC` → **Create** → **Apply Configuration**.

Ce support est la propriété de Dawan France.



(`pxe212` , lab d'exemple). La fenêtre **Create: Linux Bridge** remplie : `vmbr1` , `10.59.11.12/24` , `ens19` , `COROSYNC` . Pas de passerelle : ce réseau ne sort pas.

Derrière, les cinq cartes du nœud : `ens19` est libre, `ens18` porte déjà `vmbr0` . Les cartes `ens20` à `ens22` serviront à Ceph et à la migration, à l'atelier 5.

2. Vérifier que les trois se voient, depuis `pxe2<N>1` :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Les deux autres nœuds, sur le réseau Corosync
ping -c 3 10.<site>.<N+10>.12
ping -c 3 10.<site>.<N+10>.13   # puis le troisième
```

Résultat attendu (`pxe211` vers `10.59.11.12` , puis `10.59.11.13`) :

```
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.394/0.575/0.727/0.137 ms
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.393/0.535/0.679/0.116 ms
```

Moins d'une milliseconde : c'est ce que Corosync attend.

Ce support est la propriété de Dawan France.

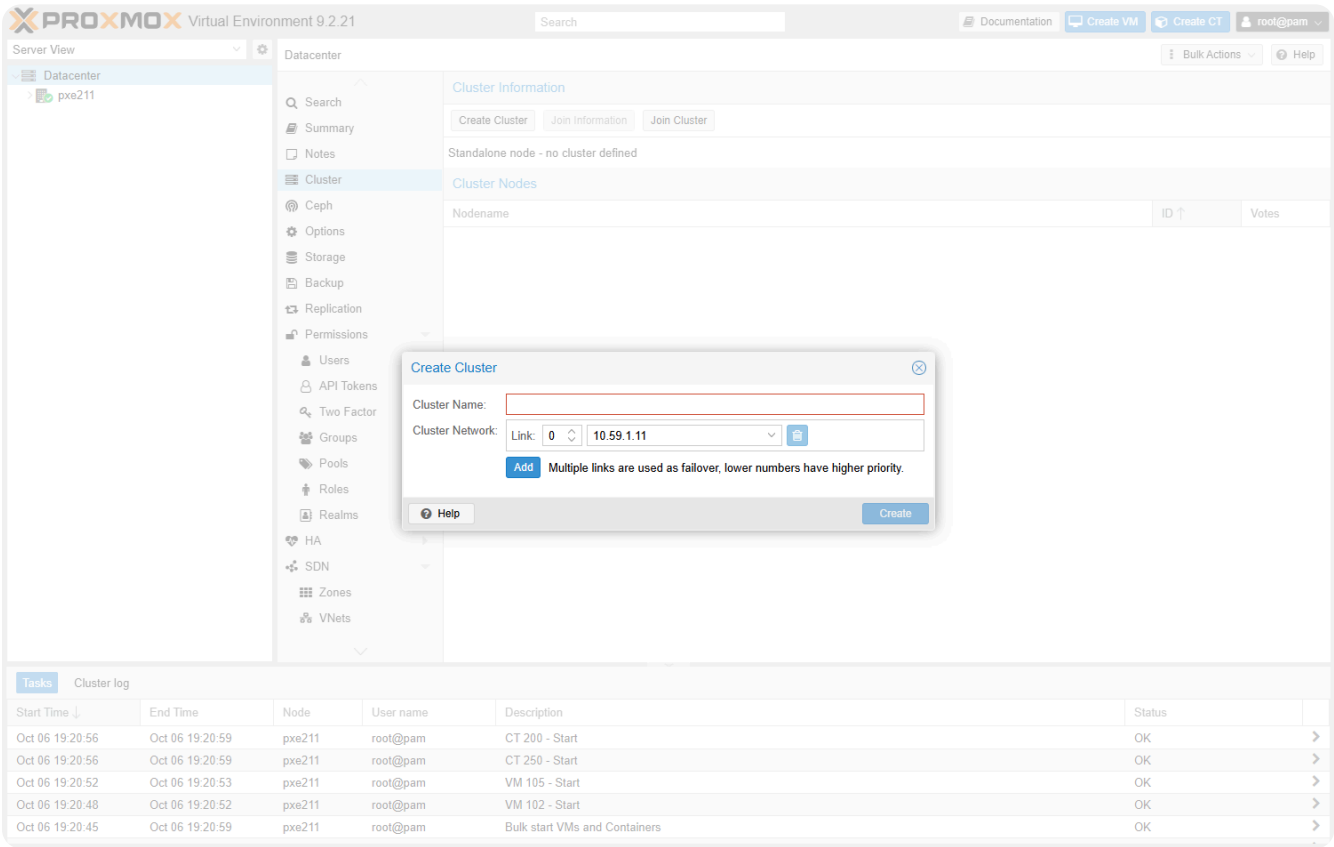
N'allez pas plus loin tant qu'un ping échoue. Un cluster monté sur un réseau incomplet perd son quorum sans raison visible. Le diagnostic est bien plus pénible une fois les nœuds liés.

Les nœuds qui rejoignent doivent être vierges de machines. Une VM ou un conteneur présent sur `pxe2<N>2` ou `pxe2<N>3` bloque la jonction.

Vérifiez avec `qm list` et `pct list`. Le nœud **fondateur**, lui, garde ses machines : elles font partie du cluster dès sa création.

Étape 1 — Créer le cluster depuis pxe21

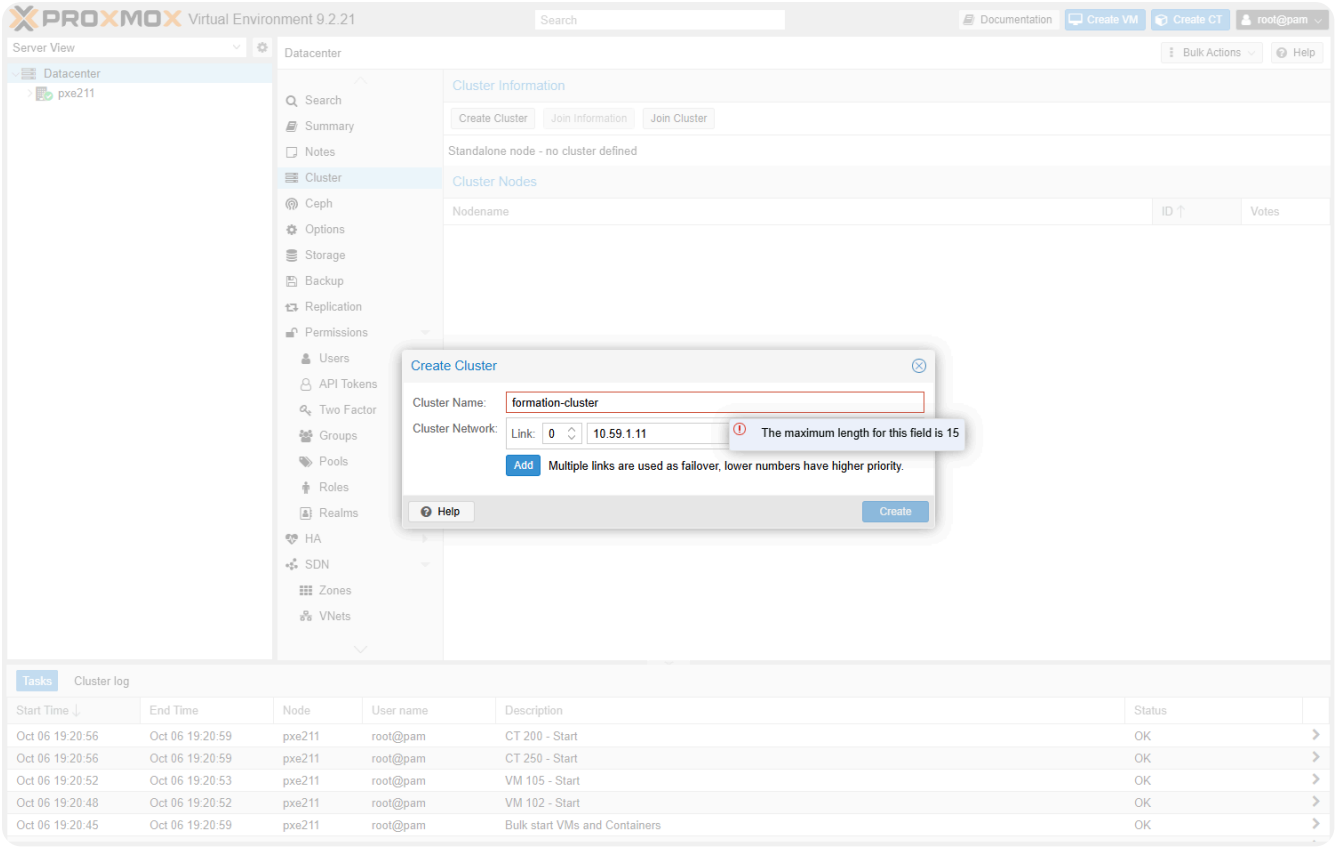
Sur `pxe2<N>1` : **Datacenter → Cluster → Create Cluster**.



À l'ouverture, **Link 0 est pré-rempli avec l'adresse de `vmbr0`**. Si vous validez tel quel, le cluster dialogue par le réseau d'administration. Il faut changer ce champ.

1. Le nom : 15 caractères au plus.

Ce support est la propriété de Dawan France.

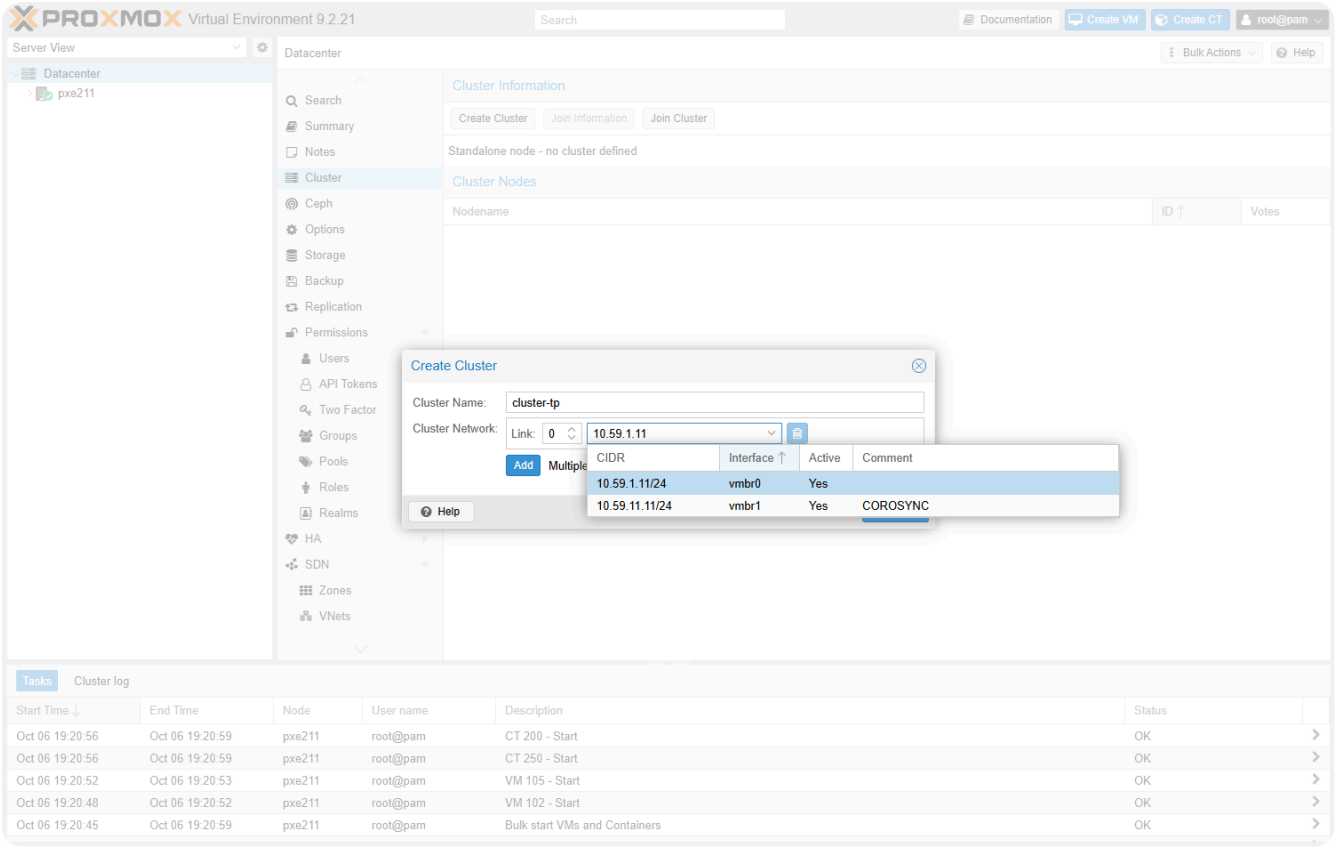


Le nom `formation-cluster` (17 caractères) est refusé : le champ passe en rouge, **Create** reste grisé. L'infobulle indique : *The maximum length for this field is 15*.

La commande `pvecm create` applique la même limite. Nommez le cluster `cluster-tp`.

2. Link 0 : l'adresse de `vmbr1`. Ouvrez la liste du champ.

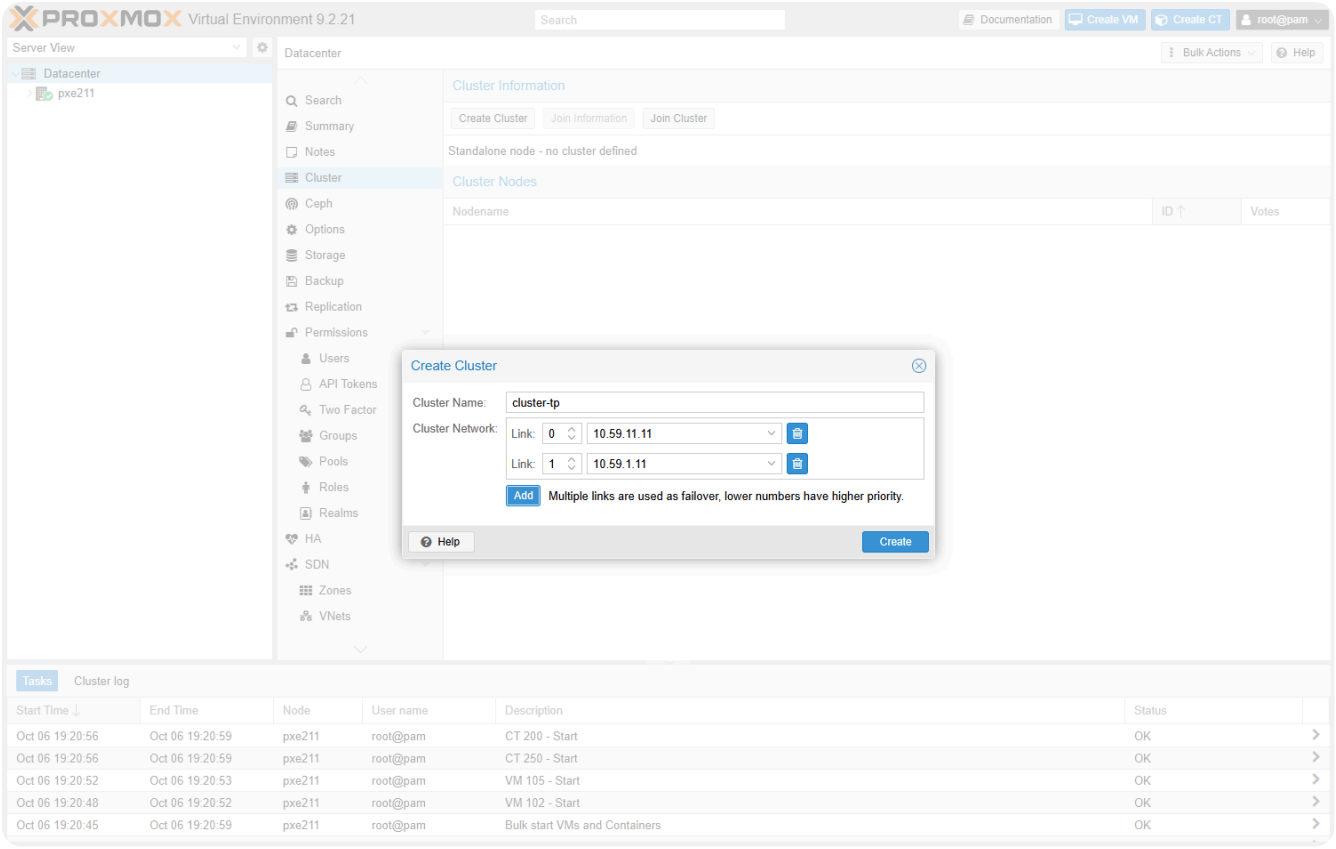
Ce support est la propriété de Dawan France.



La liste affiche chaque adresse avec son interface et son commentaire. Choisissez la ligne `vmbr1`, commentaire `COROSYNC`. Le commentaire saisi à la création du pont sert ici à ne pas se tromper.

3. Link 1 : le secours. Cliquez **Add**.

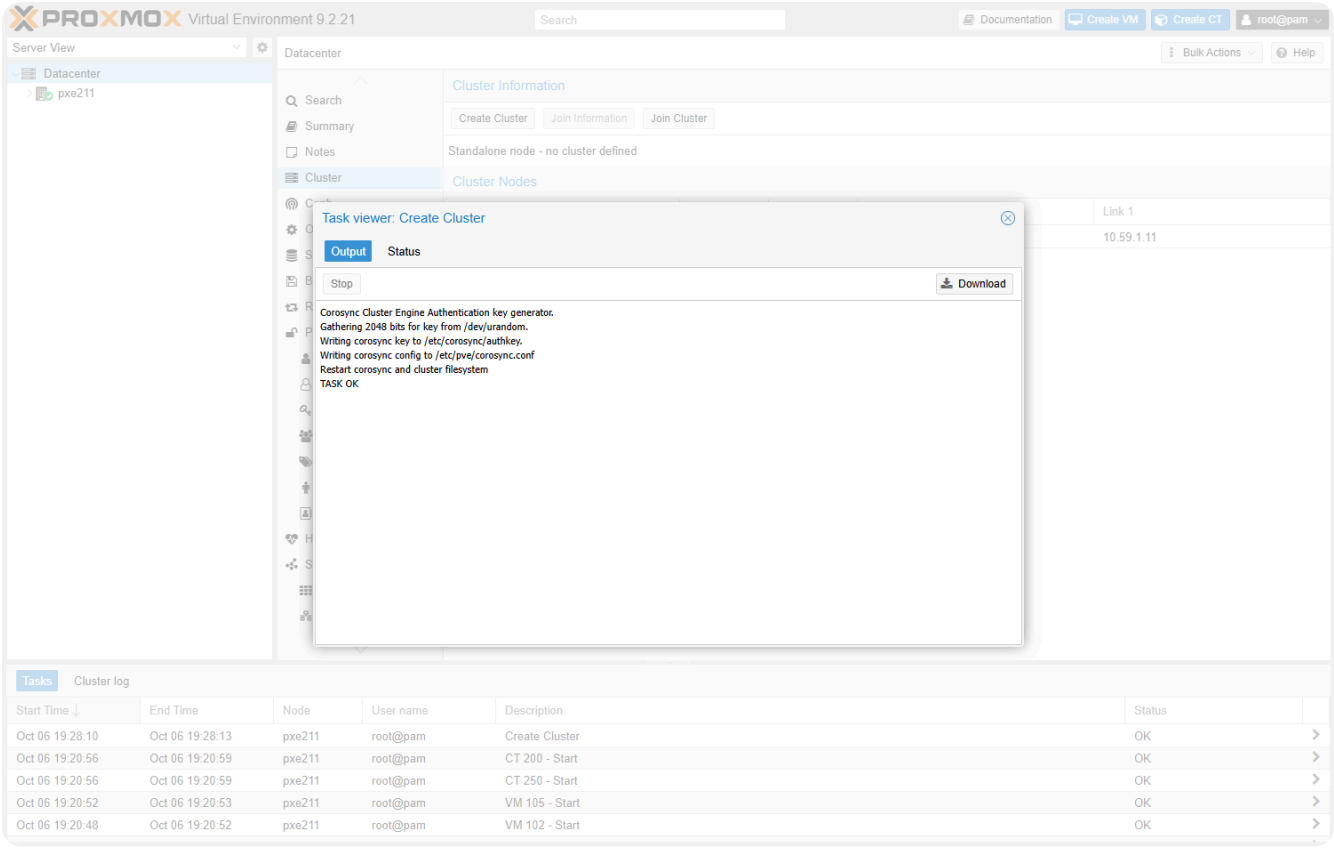
Ce support est la propriété de Dawan France.



Add crée le Link 1 et le remplit seul avec l'adresse de `vmbr0`. Le texte le rappelle : *lower numbers have higher priority*. Corosync passe par le Link 0 ; il bascule sur le Link 1 si le premier tombe.

Cliquez **Create**.

Ce support est la propriété de Dawan France.



Le journal de la tâche : génération de la clé Corosync (`/etc/corosync/authkey`), écriture de `/etc/pve/corosync.conf` , redémarrage de Corosync et du système de fichiers du cluster, puis `TASK OK` .

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI, pour référence
pvecm create cluster-tp --link0 10.<site>.<N+10>.11 --link1 10.<site>.<N>.11
```

Vérifiez que le cluster dialogue par le bon réseau :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Les adresses de chaque lien, telles qu'écrites dans la configuration
grep -E "name|ring" /etc/pve/corosync.conf
```

Résultat attendu :

```
    name: pxe211
    ring0_addr: 10.59.11.11
```

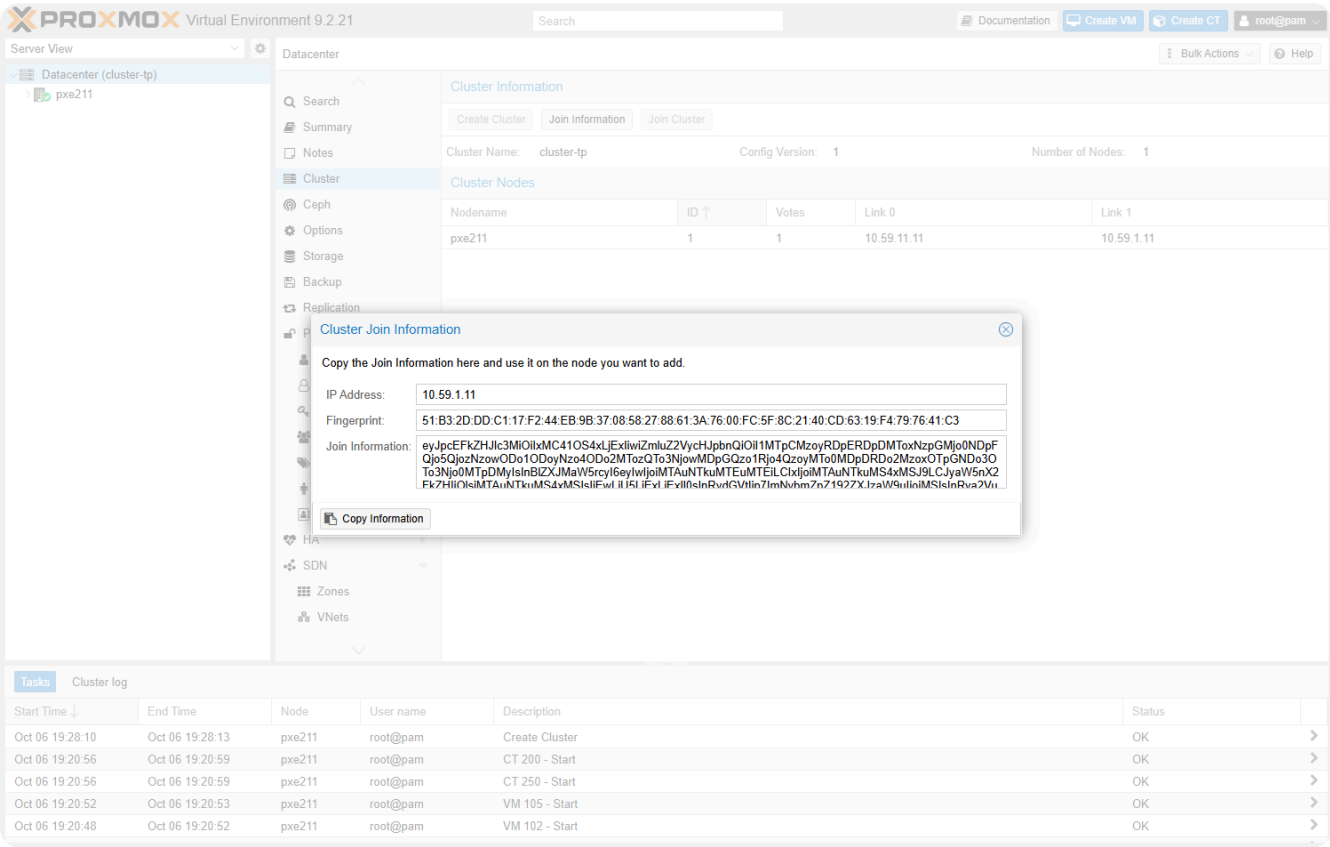
Ce support est la propriété de Dawan France.

```
    ring1_addr: 10.59.1.11
  cluster_name: cluster-tp
```

`ring0_addr` est l'adresse du Link 0 : elle doit être sur `10.<site>.<N+10>.x`. `ring1_addr`, celle du Link 1, est l'adresse de `vmbr0`.

Étape 2 — Récupérer les informations de jointure

Sur `pxe2<N>1` : **Datacenter → Cluster → Join Information → Copy Information**.



Le nœud fondateur fournit trois choses : son adresse, l'empreinte de son certificat, et un bloc encodé qui contient les deux. Derrière, la liste des nœuds montre déjà les colonnes **Link 0** et **Link 1**. Gardez le texte copié : il sert pour les deux autres nœuds.

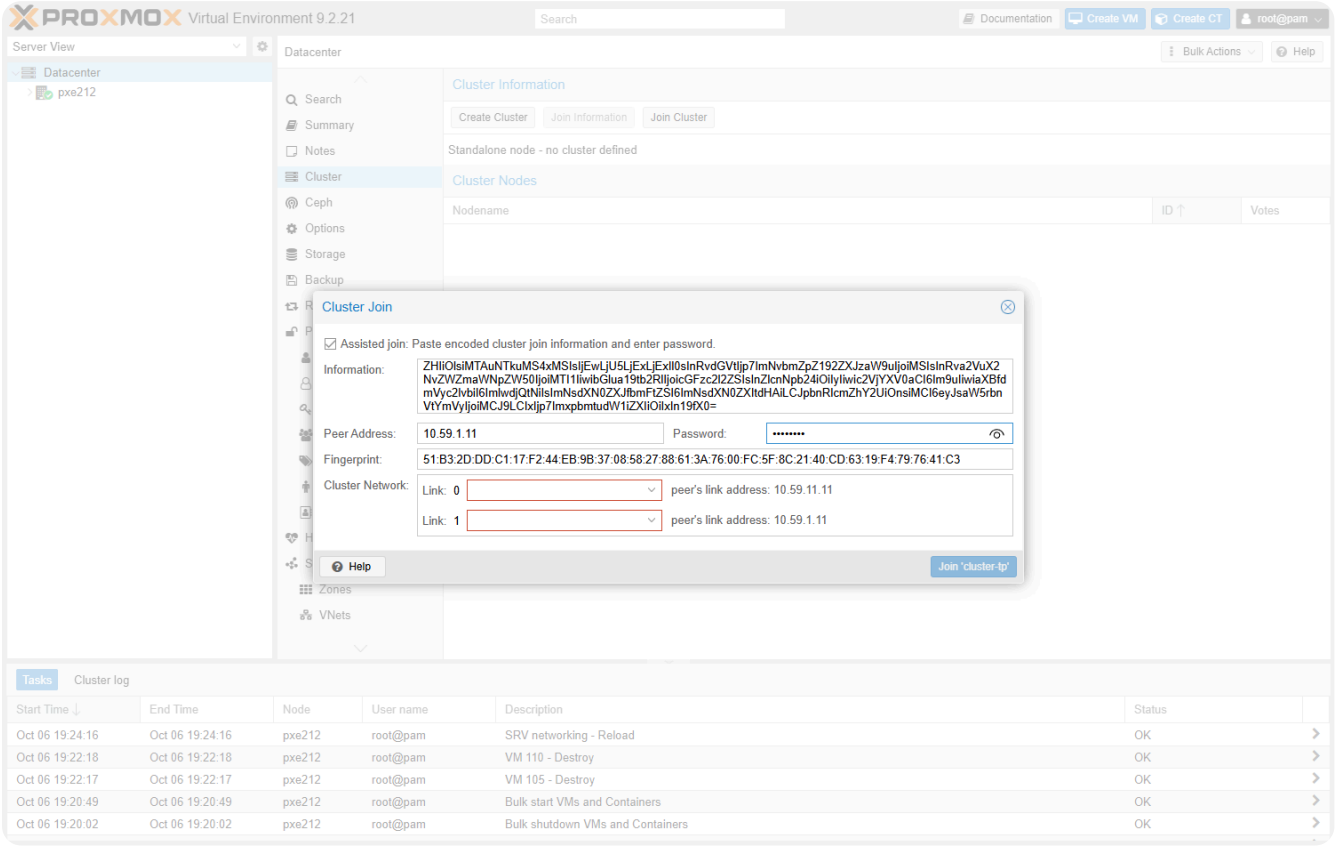
Étape 3 — Joindre pxe22 au cluster

Sur `pxe2<N>2` : **Datacenter → Cluster → Join Cluster**. Collez le bloc dans **Information**, puis saisissez le mot de passe root de `pxe2<N>1` (`proxmoxx`).

> **Join Cluster, pas Create Cluster.** Sur les nœuds 2 et 3, un clic sur **Create Cluster** fonde un second cluster, à un seul nœud. Son `pvecm status` affiche alors `Nodes: 1`, `Config`

Ce support est la propriété de Dawan France.

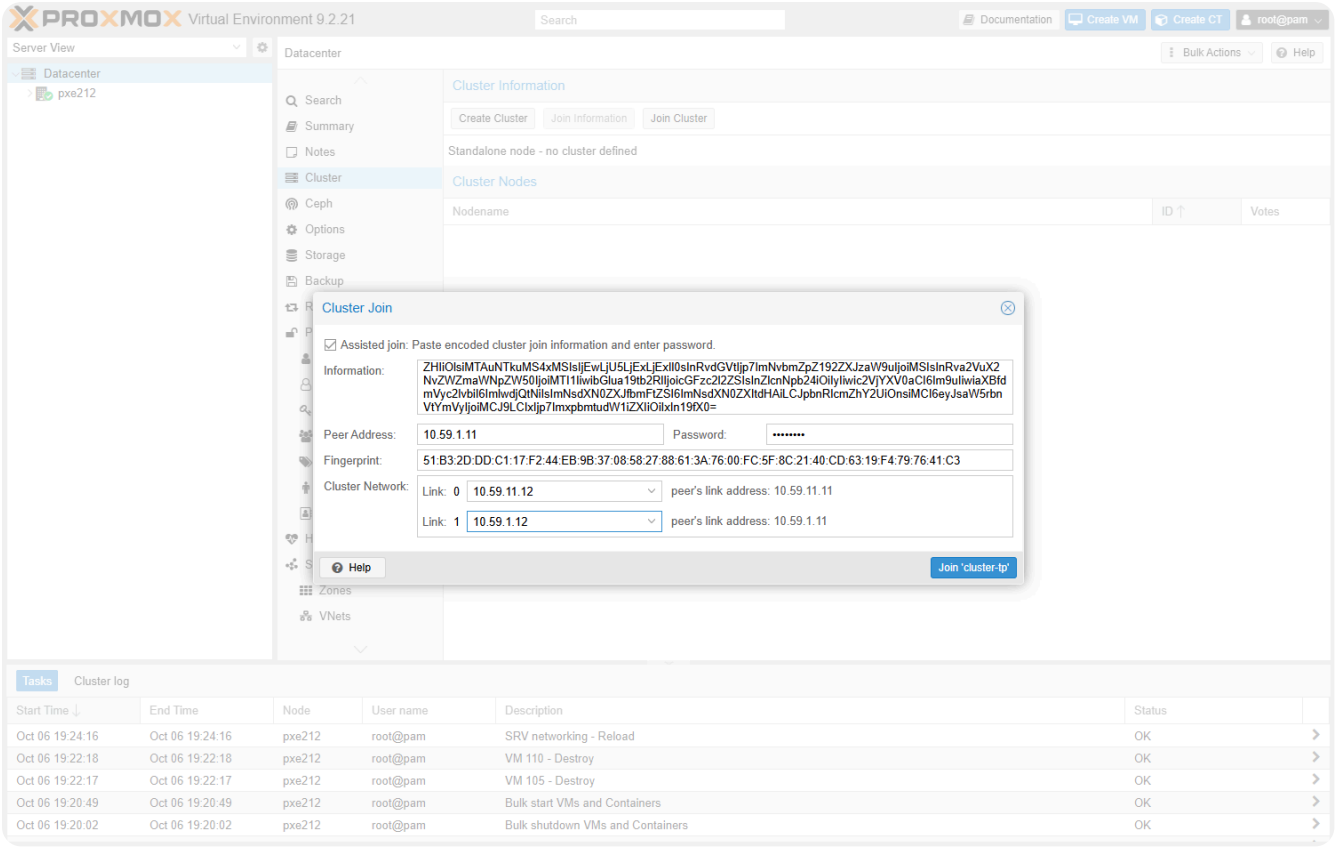
> `Version: 1`, `Expected votes: 1`, et la jonction devient impossible (`authentication key '/etc/corosync/authkey' already exists`).
>
> La réparation, sans réinstaller : cours, §2.5.



Peer Address et **Fingerprint** se remplissent seuls à partir du bloc collé. Mais **les champs Link 0 et Link 1 restent vides** et passent en rouge.

À droite, Proxmox rappelle l'adresse du pair pour chaque lien : `peer's link address`. Choisissez pour chaque lien l'adresse **de ce nœud** sur le même réseau.

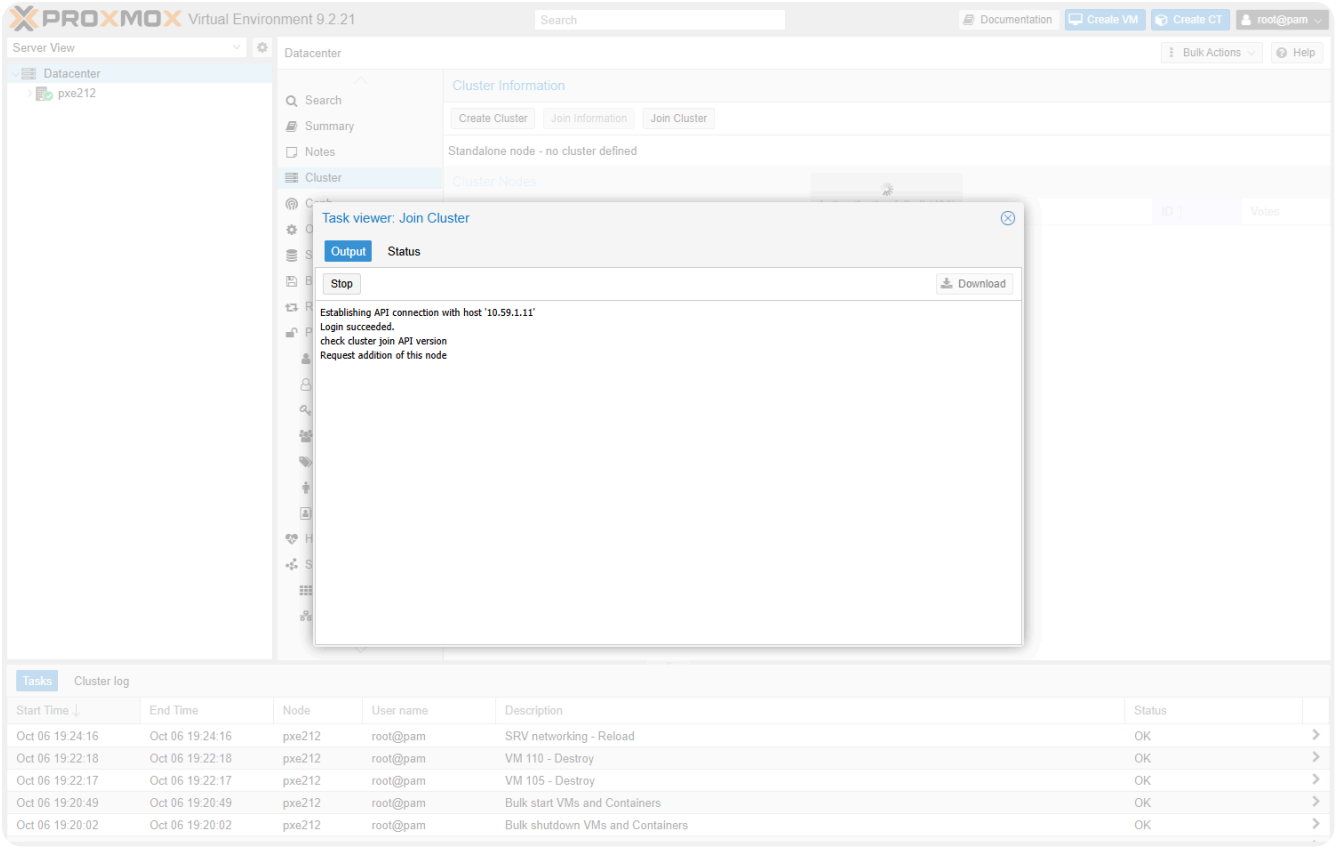
Ce support est la propriété de Dawan France.



Link 0 : `10.<site>.<N+10>.12`, en face du pair `…11`. Link 1 : l'adresse `vmbr0` du nœud. Le bouton s'intitule **Join 'cluster-tp'** : il confirme le cluster visé.

Cliquez **Join 'cluster-tp'**.

Ce support est la propriété de Dawan France.



Le journal s'arrête sur `stopping pve-cluster service`, sans `TASK OK`. C'est normal : le service qui sert l'interface redémarre, et la page perd sa connexion.

Attendez une minute, puis rechargez la page. Le nœud apparaît alors dans le cluster.

> **Pour la ligne de commande, deux adresses sur deux réseaux différents.** `pvecm add <adresse>` désigne le nœud **fondateur**, sur son interface d'administration.
>
> `--link0 address=` désigne le nœud **qui rejoint**, sur le réseau Corosync. Ce n'est pas le même réseau, et c'est normal.

Étape 4 — Joindre pxe23 au cluster

Recommencez l'étape 3 sur `pxe2<N>3`, avec les mêmes informations de jointure. Sur le lab, ce nœud a été joint en ligne de commande, pour vérifier la méthode :

Dans le Shell du nœud 3 (`pxe2<N>3`) :

```
# Sur pxe2<N>3 : la jonction au fondateur, avec ses deux liens. pvecm demande le mot de passe
# root du fondateur (proxmoxx), puis fait accepter l'empreinte de son certificat
```

Ce support est la propriété de Dawan France.

```
(yes)
pvecm add 10.<site>.<N>.11 --link0 address=10.<site>.<N+10>.13 --link1
address=10.<site>.<N>.13
```

Résultat attendu ; le mot de passe tapé ne s'affiche pas :

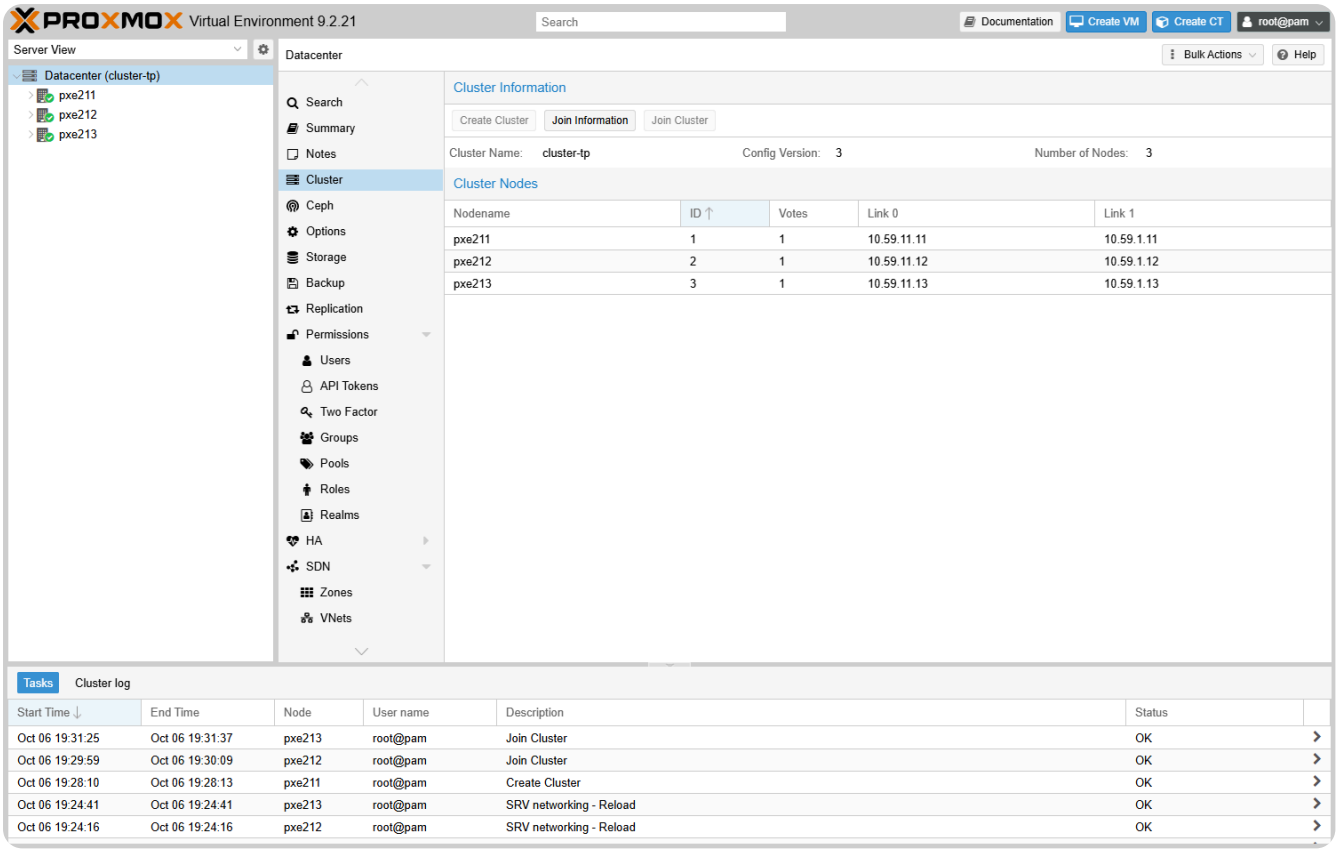
```
Please enter superuser (root) password for '10.59.1.11':
Establishing API connection with host '10.59.1.11'
The authenticity of host '10.59.1.11' can't be established.
X509 SHA256 key fingerprint is …
Are you sure you want to continue connecting (yes/no)? yes
Login succeeded.
check cluster join API version
Request addition of this node
Join request OK, finishing setup locally
stopping pve-cluster service
backup old database to '/var/lib/pve-cluster/backup/config-1791307887.sql.gz'
waiting for quorum...OK
(re)generate node files
generate new node certificate
merge authorized SSH keys
generated new node certificate, restart pveproxy and pvedaemon services
successfully added node 'pxe213' to cluster.
```

C'est la suite du journal que l'interface n'a pas pu afficher à l'étape 3 : attente du quorum, nouveau certificat, échange des clés SSH entre nœuds.

Étape 5 — Vérifier l'état du cluster

Rechargez l'interface : les trois nœuds apparaissent sous **Datacenter (cluster-tp)**.

Ce support est la propriété de Dawan France.



Config Version: 3 : la configuration a changé trois fois (création, puis deux jonctions). Chaque nœud a une voix, et ses deux adresses : Link 0 sur `10.59.11.x` (`vmbr1`), Link 1 sur `10.59.1.x` (`vmbr0`). En bas, les trois tâches : **Create Cluster**, puis deux **Join Cluster**, toutes `OK` .

Dans le Shell de chaque nœud du cluster :

```
# Le quorum, puis l'état de chaque lien Corosync
pvecm status
corosync-cfgtool -s   # l'état de chaque lien Corosync
```

Résultat sur `pxe211` (`pvecm status` , extrait) :

```
Name:             cluster-tp
Config Version:   3
Transport:        knet
Secure auth:      on

Quorum information
------------------
Nodes:            3
```

Ce support est la propriété de Dawan France.

```
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate

Membership information
----------------------
    Nodeid      Votes Name
0x00000001          1 10.59.11.11 (local)
0x00000002          1 10.59.11.12
0x00000003          1 10.59.11.13
```

`Quorum: 2` : il faut deux voix sur trois pour que le cluster décide. La section Membership affiche les **adresses du Link 0**, pas les noms : c'est la preuve que Corosync passe bien par `vmbr1`. Pour les noms, `pvecm nodes`.

Résultat sur `pxe211` (`corosync-cfgtool -s`) :

```
Local node ID 1, transport knet
LINK ID 0 udp
    addr    = 10.59.11.11
    status:
        nodeid:          1: localhost
        nodeid:          2: connected
        nodeid:          3: connected
LINK ID 1 udp
    addr    = 10.59.1.11
    status:
        nodeid:          1: localhost
        nodeid:          2: connected
        nodeid:          3: connected
```

Les deux liens sont `connected` vers les deux autres nœuds. Si le Link 0 tombe, le cluster continue par le Link 1.

Ce support est la propriété de Dawan France.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le quorum, puis le nombre de membres
pvecm status | grep Quorate   # le quorum
pvecm nodes | grep -c pxe   # nombre de nœuds membres
```

Résultat attendu : `Quorate: Yes`, puis `3`.

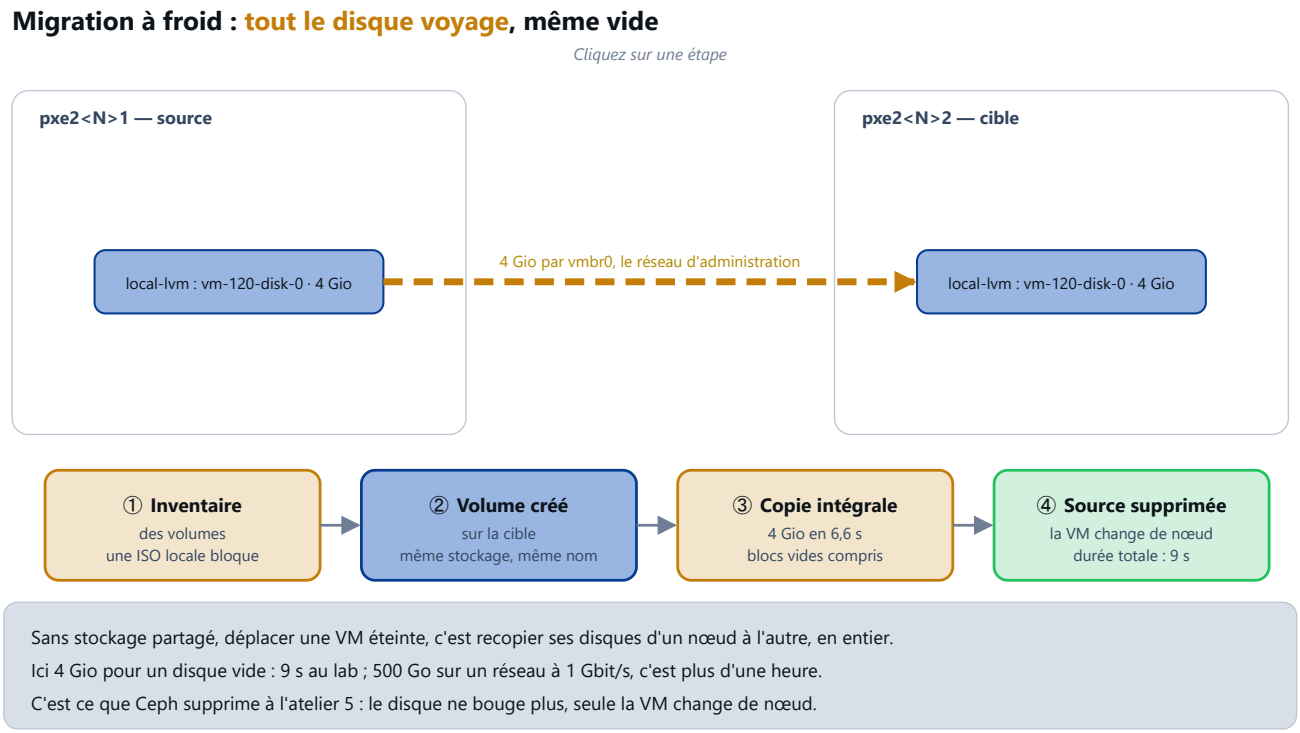
Atelier 2 — Migration à froid

Contexte

Le premier bénéfice tangible d'un cluster, c'est de pouvoir **déplacer une VM d'un nœud à l'autre**. À froid — VM éteinte — l'opération est simple et sans risque : Proxmox copie le disque vers le nœud cible puis y recrée la VM. C'est la brique que Ceph rendra ensuite possible **à chaud**, sans arrêt.

Résultat à obtenir : une VM éteinte migrée avec succès d'un nœud vers un autre, et la compréhension de ce que Proxmox a dû copier faute de stockage partagé.

Ce support est la propriété de Dawan France.



Ce que Proxmox fait quand vous cliquez **Migrate** sur une VM éteinte. Il inventorie les disques, crée le volume sur la cible, copie le disque en entier, puis supprime l'original. Cliquez sur une étape pour lire la ligne de journal correspondante.

Arborescence attendue

```
VM de test (éteinte)
pxe2<N>1  ──[ migration à froid ]──▶  pxe2<N>2
        (copie du disque, puis recréation sur la cible)   ← 4 Gio copiés, même vide
```

Ce que vous devez faire

1. **Disposer d'une VM de test** éteinte sur l'un des nœuds (en recréer une au besoin).
2. **Lancer la migration à froid** depuis l'interface, vers un autre nœud du cluster.
3. **Observer ce qui se passe** : sans stockage partagé, Proxmox recopie l'intégralité du disque sur le nœud cible. Notez le temps que cela prend — c'est exactement ce que Ceph supprimera à l'atelier suivant.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 0 — Créer une VM de test

Les VM 101 et 102 du chapitre 2 sont toujours là, sur votre nœud — mais elles servent aux ateliers suivants, et les deux nœuds qui vous ont rejoint sont vides. Créez donc une VM dédiée à l'exercice, la **120**. Deux façons de faire, selon le temps dont vous disposez — **les deux permettent de faire l'atelier en entier**.

	Option A — VM sans système	**Option B — VM avec Debian**
Durée	environ une minute	une vingtaine de minutes
Ce que vous pourrez montrer	le disque et la mémoire changés de serveur, la VM jamais redémarrée	la même chose, **plus** un service qui ne s'interrompt pas pendant la bascule
Il vous faut	rien	l'ISO Debian, encore présente dans `local` depuis le chapitre 1

Si vous hésitez, prenez l'option B. Refaire une installation n'est pas du temps perdu : c'est le geste que vous répéterez le plus souvent en production.

Et la migration à chaud d'une machine qui rend réellement un service est autrement plus convaincante qu'un écran de firmware. L'option A, elle, va droit à la migration.

Deux règles valables dans les deux cas. Elles ne sont pas des détails : chacune empêche la migration si on la néglige.

- **Le lecteur d'ISO doit être vide au moment de migrer.** Le stockage `local` n'est pas partagé : chaque nœud a le sien.

  Une VM dont le lecteur pointe vers une ISO de `local` sur `pxe2<N>1` refusera de partir, puisque le fichier n'existe pas sur la cible. En option A on ne monte rien ; en option B on détache l'ISO une fois l'installation finie.

- **Le type de CPU doit rester générique.** Le type `host` expose le processeur réel du nœud, et la VM ne peut alors migrer que vers un serveur au processeur identique. Le type par défaut `x86-64-v2-AES` présente un CPU commun aux trois nœuds — c'est lui qui rend la migration possible.

Option A — une VM sans système, en une minute

Ce support est la propriété de Dawan France.

Pourquoi ça suffit. Ce que la migration déplace, c'est le **disque** quand la VM est éteinte et la **mémoire** quand elle tourne.

Une VM sans système a bien les deux : elle démarre sur son firmware, affiche `No bootable device`, et reste un processus vivant avec sa mémoire. Le mécanisme démontré est exactement le même.

En interface graphique. Sur `pxe2<N>1`, bouton **Create VM** en haut à droite :

Onglet	Champ	Valeur
General	Node	`pxe2<N>1`
	VM ID	`120`
	Name	`vm-migration`
OS	Storage / ISO image	**Do not use any media**
	Guest OS — Type	`Linux`
	Guest OS — Version	`7.x - 2.6 Kernel`
System	Machine	`Default (i440fx)`
	BIOS	`Default (SeaBIOS)`
	SCSI Controller	`VirtIO SCSI single`
	Qemu Agent	✓
Disks	Bus/Device	`SCSI` / `0`
	Storage	`local-lvm`
	Disk size (GiB)	`4`
CPU	Sockets / Cores	`1` / `2`
	Type	`x86-64-v2-AES` *(surtout pas* `host` *)*
Memory	Memory (MiB)	`1024`
Network	Bridge	`vmbr0`
	Model	`VirtIO (paravirtualized)`
Confirm	Start after created	☐

→ **Finish**

En ligne de commande :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# créer la VM 120, sans système
qm create 120 \
  --name vm-migration \
  --memory 1024 \
  --cores 2 \
  --cpu x86-64-v2-AES \
  --ostype l26 \
  --scsihw virtio-scsi-single \
  --scsi0 local-lvm:4 \
  --net0 virtio,bridge=vmbr0 \
  --agent 1

qm start 120   # la démarrer
qm status 120   # vérifier qu'elle tourne
```

Résultat attendu : `status: running`. La console affiche `No bootable device` ou l'invite du firmware — c'est normal, la VM occupe bien un disque et de la mémoire.

> **Attention à l'** `uptime` **de Proxmox.** `qm status 120 --verbose | grep uptime` compte la vie du **processus QEMU**, pas celle du système. Après une migration à chaud, la VM tourne dans un nouveau processus, sur le nœud cible : le compteur repart de zéro, sans qu'il y ait eu redémarrage.
>
> Sur le lab : `uptime: 20` avant la bascule, `uptime: 16` juste après. Pour prouver qu'une VM n'a pas redémarré, il faut le demander **à son système** (`uptime -s` dans la VM), ce que seule l'option B permet.

Option B — une VM avec Debian, comme en production

C'est l'installation du chapitre 2, en version minimale : ni bureau graphique ni serveur web, juste le système et de quoi répondre au réseau.

1. Vérifier que l'ISO est toujours là

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
ls -lh /var/lib/vz/template/iso/   # les ISO présentes sur le stockage local
```

Ce support est la propriété de Dawan France.

Résultat attendu : l'ISO Debian téléchargée au chapitre 2. Si elle a disparu, reprenez-la : votre serveur → **local** → **ISO Images** → **Download from URL**.

2. Créer la VM — mêmes valeurs que l'option A, avec **deux différences** : l'ISO dans le lecteur, et un disque de 12 Go pour loger le système.

Onglet	Champ	Valeur
OS	Storage / ISO image	`local` / l'ISO Debian
Disks	Disk size (GiB)	`12`

Tout le reste est identique — `x86-64-v2-AES` compris.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Relevez d'abord le nom exact de l'ISO avec la commande ci-dessus
qm create 120 \
  --name vm-migration \
  --memory 2048 \
  --cores 2 \
  --cpu x86-64-v2-AES \
  --ostype l26 \
  --scsihw virtio-scsi-single \
  --scsi0 local-lvm:12 \
  --ide2 local:iso/<nom-exact-de-l-iso>,media=cdrom \
  --net0 virtio,bridge=vmbr0 \
  --boot 'order=ide2;scsi0' \
  --agent 1

qm start 120   # démarrer sur l'ISO pour installer Debian
```

3. Installer Debian depuis la console.

Allez au plus simple : langue et clavier, nom de machine `vm-migration`, mot de passe root, un utilisateur. Partitionnement **assisté — tout dans une seule partition**.

Et surtout, à l'écran **Sélection des logiciels** : **décochez tout sauf** `Utilitaires usuels du système`. Sans environnement de bureau, l'installation tient en une dizaine de minutes.

4. Après le redémarrage, dans la VM, installer l'agent invité — c'est lui qui permettra à Proxmox d'afficher l'adresse IP de la machine :

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Dans la VM (sa **Console**, ou en SSH) :

```
apt update                          # la liste des paquets
apt install -y qemu-guest-agent     # installer l'agent invité
# Lancer l'agent maintenant ; il redémarrera seul à chaque démarrage (VM créée
avec --agent enabled=1)
systemctl start qemu-guest-agent
ip -br addr   # l'adresse IP de la VM
```

5. Détacher l'ISO — à ne pas oublier

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qm set 120 --ide2 none,media=cdrom   # vider le lecteur CD de la VM
qm config 120 | grep ide2   # vérifier : plus d'ISO
```

Par l'interface : `VM 120` → **Hardware** → **CD/DVD Drive** → **Edit** → **Do not use any media** → **OK**.

> **C'est l'oubli le plus fréquent de cet atelier.** Tant que le lecteur pointe vers `local:iso/...`, la fenêtre Migrate affiche *Cannot migrate VM with local CD/DVD* et refuse de partir. Le fichier ISO existe sur `pxe2<N>1`, pas sur la cible : `local` n'est pas partagé.

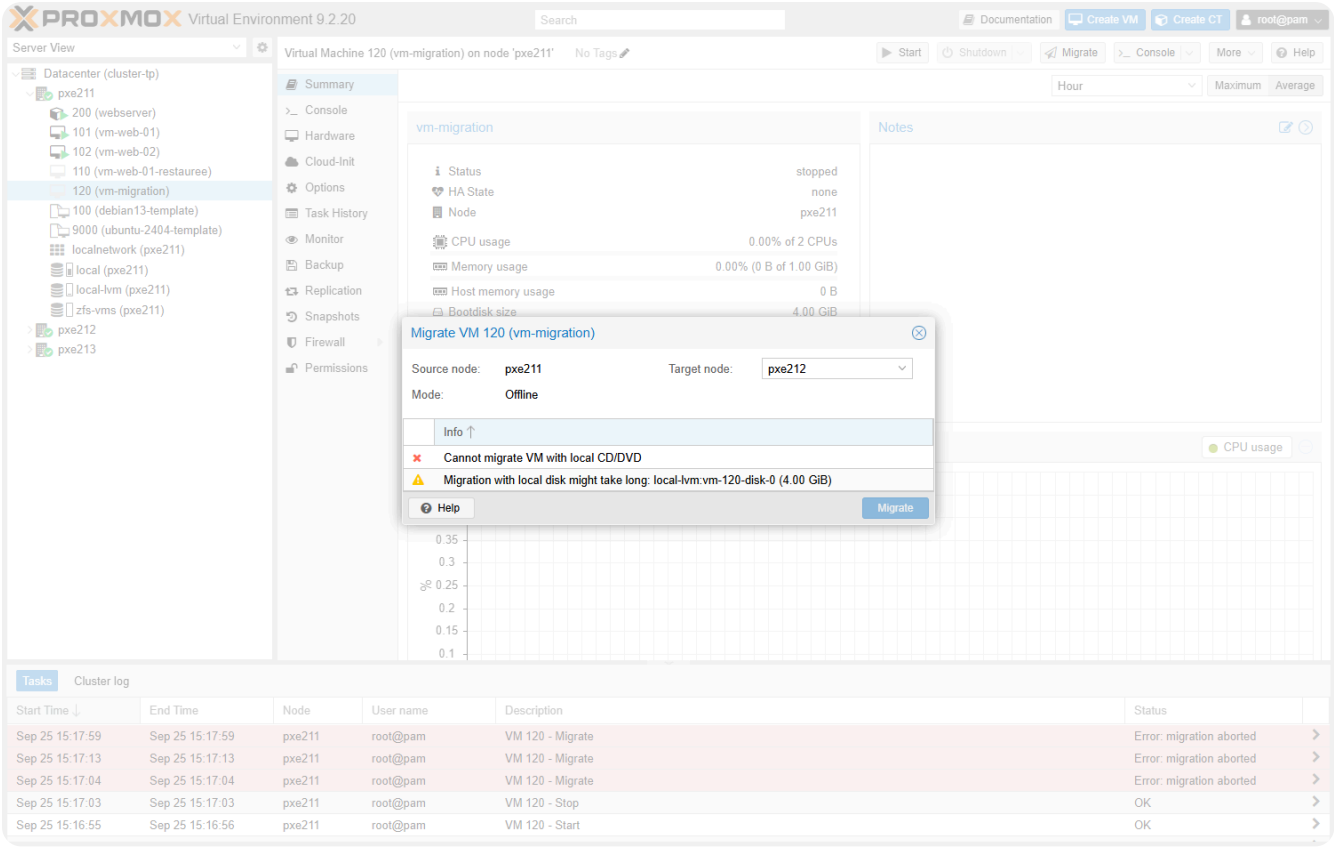
Résultat attendu : `qm agent 120 ping` ne renvoie rien (c'est le signe qu'il répond), et la ligne **IPs** de l'onglet **Summary** affiche l'adresse de la VM. Notez-la : elle vous servira pour le `ping` continu de l'étape 2.

Étape 1 — Migration à froid depuis l'interface

Si la VM est allumée, éteignez-la d'abord : VM 120 → **Shutdown**.

VM 120 → **Migrate** (bouton en haut à droite).

Ce support est la propriété de Dawan France.



L'ISO Debian est restée dans le lecteur. La fenêtre le détecte avant toute tentative : **Cannot migrate VM with local CD/DVD**, et le bouton **Migrate** reste grisé. Détachez l'ISO (étape 0, option B, point 5), puis rouvrez la fenêtre.

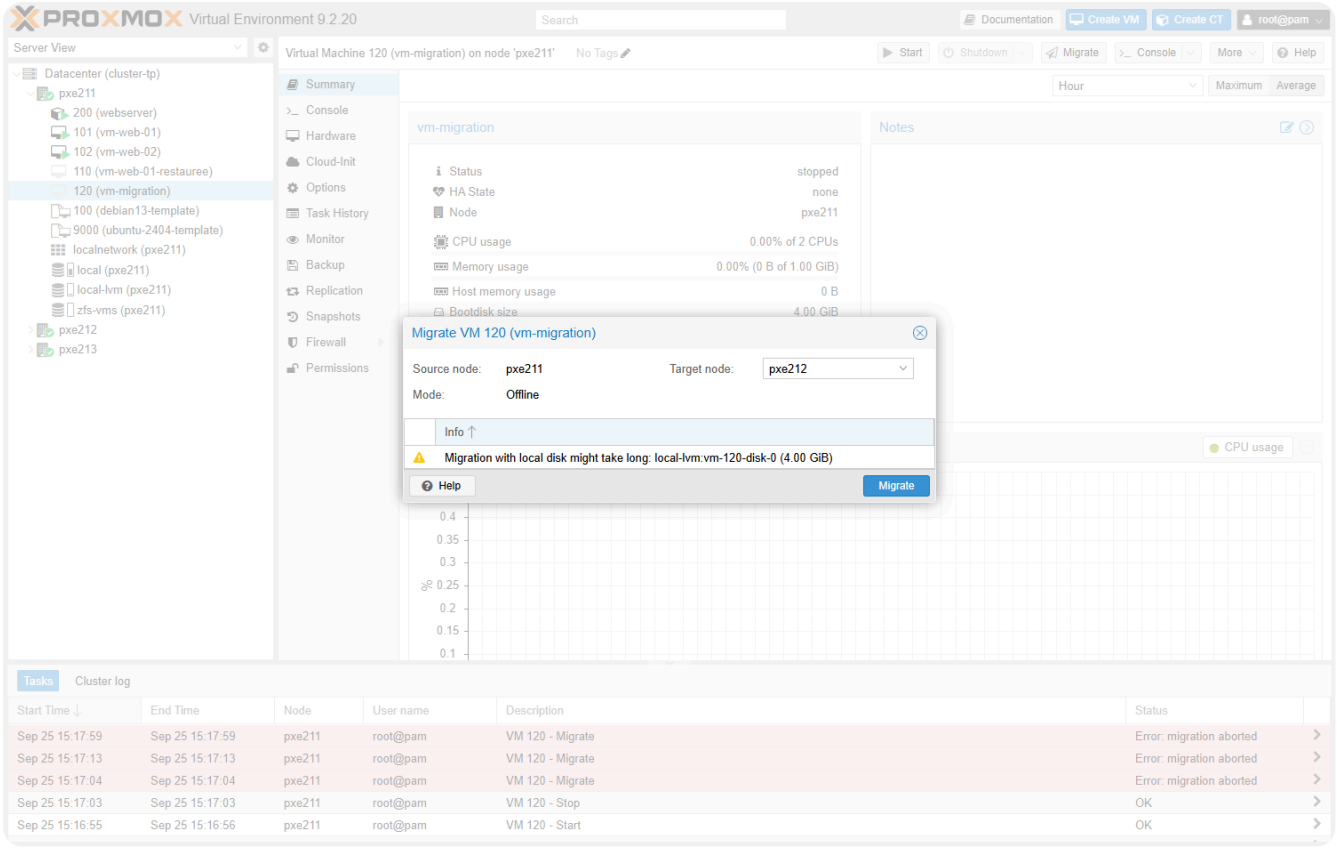
En ligne de commande, la même VM est refusée avec un message plus précis.

Dans le Shell du nœud 1 :

```
# Tenter la migration vers le nœud 2, ISO encore attachée
qm migrate 120 pxe2<N>2
```

```
found local disk 'local:iso/debian-13.6.0-amd64-netinst.iso' (attached)
can't migrate local disk 'local:iso/debian-13.6.0-amd64-netinst.iso': local cdrom
image
ERROR: migration aborted (duration 00:00:00): Problem found while scanning
volumes - can't migrate VM - check log
```

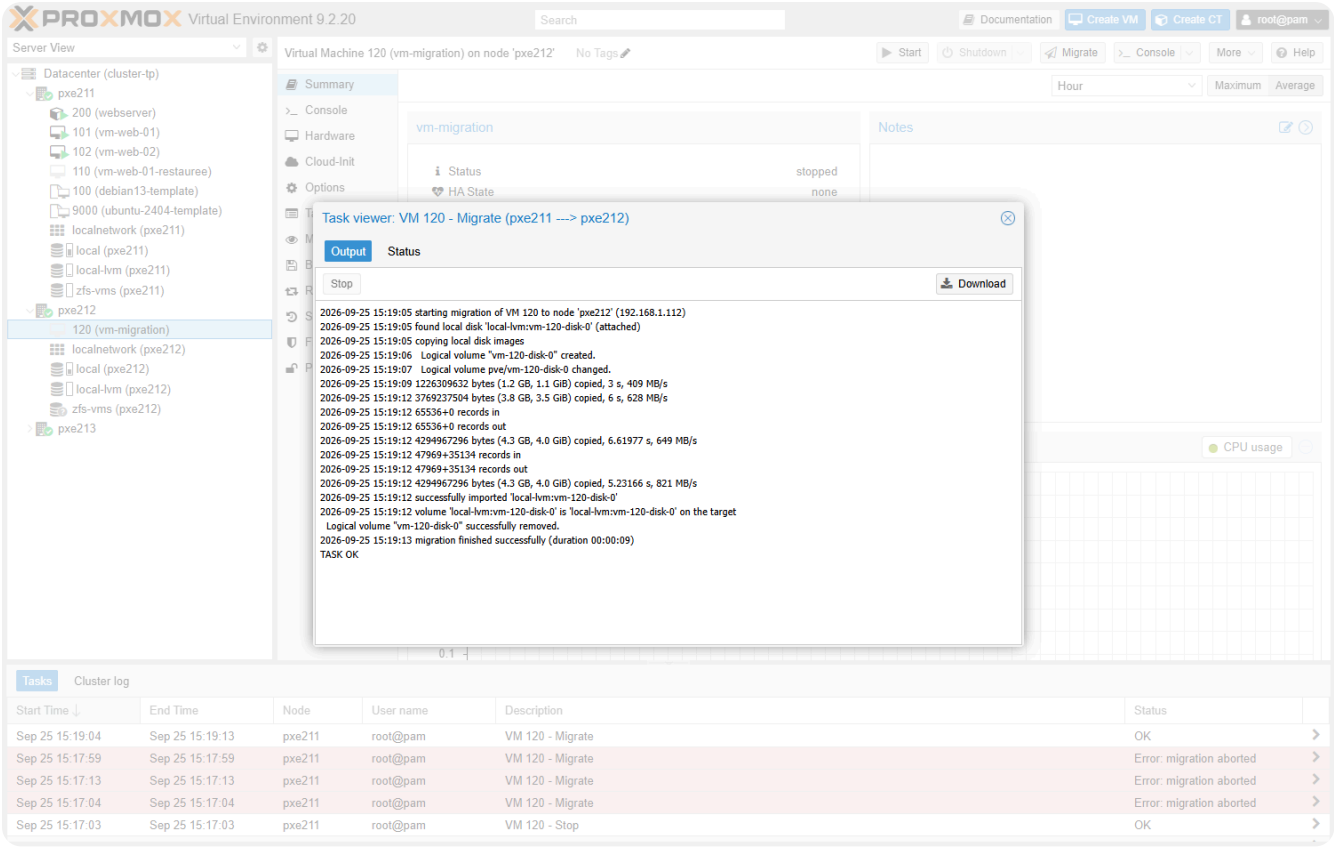
Ce support est la propriété de Dawan France.



Lecteur vide, la fenêtre ne garde qu'un avertissement : *Migration with local disk might take long*, suivi du disque et de sa taille. **Mode : Offline**, puisque la VM est éteinte.

Choisissez **Target node** `pxe2<N>2`, puis **Migrate**. Le disque atterrit sur le stockage de même nom du nœud cible : `local-lvm`.

Ce support est la propriété de Dawan France.



Le journal complet : disque local repéré, volume créé sur la cible, copie, puis suppression du volume source.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI
qm migrate 120 pxe2<N>2
```

Résultat attendu (option A, disque de 4 Gio) :

```
starting migration of VM 120 to node 'pxe212' (10.59.1.12)
found local disk 'local-lvm:vm-120-disk-0' (attached)
copying local disk images
4294967296 bytes (4.3 GB, 4.0 GiB) copied, 4.97538 s, 863 MB/s
successfully imported 'local-lvm:vm-120-disk-0'
  Logical volume "vm-120-disk-0" successfully removed.
migration finished successfully (duration 00:00:06)
```

Sur le lab (`10.59.1.x` : remplacez par votre réseau). Deux constats. **Les 4 Gio ont traversé le réseau**, alors que le disque est vide : sans stockage partagé, Proxmox copie le volume entier.

Ce support est la propriété de Dawan France.

Et la copie est passée par `10.59.1.12`, l'adresse `vmbr0` de la cible : le réseau d'administration. L'atelier 5 supprime la copie grâce à Ceph, et envoie la migration sur son propre réseau.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# qm list ne montre que les VM du nœud local : on interroge pxe2<N>2 par l'API
pvesh get /nodes/pxe2<N>2/qemu --output-format text | grep 120
```

Résultat attendu : une ligne pour la VM 120, `stopped`, `vm-migration`. Dans l'arbre de gauche, la VM apparaît sous `pxe2<N>2`.

Atelier 3 — Migrer sans stockage partagé : VM allumée et conteneur

Contexte

À l'atelier 2, la VM 120 a changé de nœud **éteinte**. En production, on déplace plutôt des machines **en service**, pour vider un nœud avant une intervention. Sans stockage partagé, deux cas se présentent, et ils ne coûtent pas la même chose :

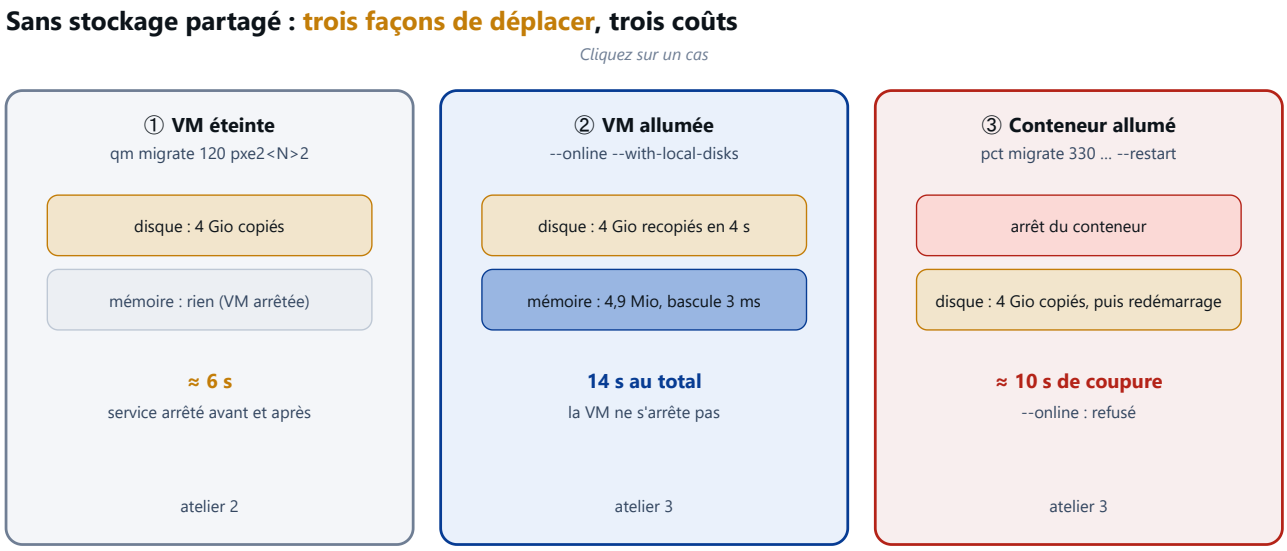
- **une VM allumée** peut migrer à chaud, à condition de demander la copie de ses disques locaux (`--with-local-disks`) ;
- **un conteneur** ne migre jamais à chaud : il est arrêté, déplacé, puis redémarré (`--restart`). Son service est coupé pendant l'opération.

On mesure les deux, et on les compare à la migration à froid de l'atelier 2.

Résultat à obtenir : la VM 120, allumée, migrée de `pxe2<N>2` vers `pxe2<N>3` sans être arrêtée. Un conteneur neuf, le CT 330, est migré de `pxe2<N>1` vers `pxe2<N>2` avec `--restart`, et la durée de sa coupure est mesurée au `ping`.

Les deux refus sont expliqués (VM sans `--with-local-disks`, conteneur avec `--online`). En fin d'atelier, la VM 120 est éteinte sur `pxe2<N>2` et le CT 330 supprimé.

Dawan
Ce support est la propriété de Dawan France.



Sans Ceph, le disque voyage toujours. Ce qui change d'un cas à l'autre, c'est ce qui reste en service pendant le voyage.
Cliquez sur un cas pour lire la ligne de journal obtenue sur le lab.

Les trois cas côte à côte, avec ce qui voyage et ce qui reste en service. Sans Ceph, le disque voyage toujours. Cliquez sur un cas pour lire la ligne de journal obtenue sur le lab.

Arborescence attendue

```
pxe2<N>2  ── VM 120 allumée ──[ --online --with-local-disks ]──▶  pxe2<N>3
          (disque recopié pendant que la VM tourne, puis mémoire)

pxe2<N>1  ── CT 330 « ct-migration » ──[ --restart ]──▶  pxe2<N>2
          (arrêt, copie du disque, redémarrage)          ← mesuré par un ping depuis
pxe2<N>3

Fin d'atelier : VM 120 éteinte, de retour sur pxe2<N>2 ; CT 330 supprimé
```

Ce que vous devez faire

1. **Démarrer** la VM 120 sur `pxe2<N>2`, puis tenter de la migrer à chaud vers `pxe2<N>3`, **sans** option. Lire le refus.
2. **Relancer** la migration avec la copie des disques locaux. Lire le journal : qu'est-ce qui voyage, dans quel ordre, et combien de temps la VM est-elle figée ?
3. **Créer** sur `pxe2<N>1` un petit conteneur Debian, le CT 330, en adresse fixe `10.<site>.<N>.60`, et le démarrer.

Ce support est la propriété de Dawan France.

4. **Lancer** depuis `pxe2<N>3` un `ping` continu vers ce conteneur.
5. **Tenter** une migration à chaud du conteneur vers `pxe2<N>2`, puis le migrer avec `--restart`. Compter les réponses perdues au `ping`.
6. **Remettre en état** : supprimer le CT 330, éteindre la VM 120 et la ramener sur `pxe2<N>2`.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> Sur le lab (`pxe211` à `pxe213` , réseau `10.59.1.x`). Chez vous, les nœuds s'appellent `pxe2<N>1` à `pxe2<N>3` et le réseau est `10.<site>.<N>.x` .

Étape 1 — La VM allumée refuse de partir

En ligne de commande.

Dans le Shell de `pxe2<N>2` :

```
qm start 120                          # démarrer la VM
qm migrate 120 pxe2<N>3 --online      # migration à chaud, sans autre option
```

Résultat attendu :

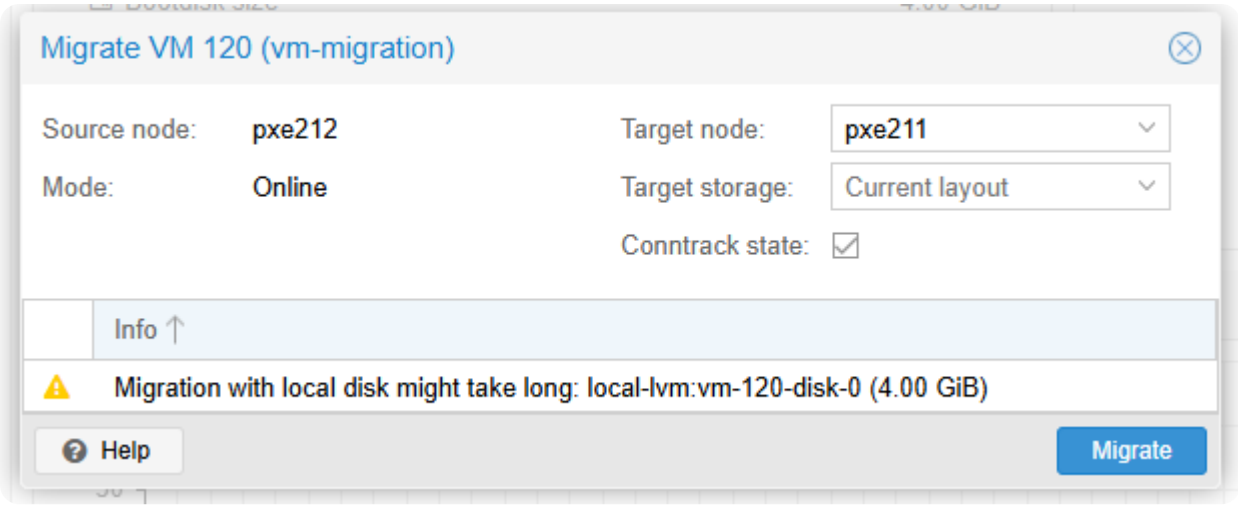
```
found local disk 'local-lvm:vm-120-disk-0' (attached)
can't migrate local disk 'local-lvm:vm-120-disk-0': can't live migrate attached
local disks without with-local-disks option
ERROR: Problem found while scanning volumes - can't migrate VM - check log
```

Le disque est sur `local-lvm`, que chaque nœud possède en propre : la cible ne le voit pas. En ligne de commande, Proxmox refuse de déplacer une VM en service sans qu'on lui demande explicitement de recopier ce disque.

Étape 2 — Migration à chaud avec les disques locaux

En interface graphique. VM 120 → **Migrate**.

Ce support est la propriété de Dawan France.



La fenêtre indique **Mode : Online**, puisque la VM tourne. Elle ajoute un champ **Target storage** (`Current layout` : le disque garde le stockage de même nom sur la cible) et prévient : *Migration with local disk might take long*.

L'interface ajoute d'elle-même la copie du disque : elle ne propose pas de case à cocher. Choisissez **Target node** `pxe2<N>3` , puis **Migrate**.

En ligne de commande, la copie du disque se demande avec l'option `--with-local-disks` .

Dans le Shell de `pxe2<N>2` :

```
qm migrate 120 pxe2<N>3 --online --with-local-disks   # recopier le disque, puis la mémoire
```

Résultat attendu (extrait) :

```
starting migration of VM 120 to node 'pxe213' (10.59.1.13)
found local disk 'local-lvm:vm-120-disk-0' (attached)
starting VM 120 on remote node 'pxe213'
starting storage migration
mirror-scsi0: transferred 1.0 GiB of 4.0 GiB (25.12%) in 1s
mirror-scsi0: transferred 4.0 GiB of 4.0 GiB (100.00%) in 4s, ready
switching mirror jobs to actively synced mode
starting online/live migration on unix:/run/qemu-server/120.migrate
average migration speed: 1.0 GiB/s - downtime 3 ms
migration completed, transferred 4.9 MiB VM-state
mirror-scsi0: Completed successfully.
migration finished successfully (duration 00:00:14)
```

Ce support est la propriété de Dawan France.

Le journal se lit en trois temps :

Temps	Lignes	Ce qui se passe
① Le disque	`starting storage migration` , `mirror-scsi0 …`	Le disque est recopié vers la cible (*drive mirror*) pendant que la VM tourne ; les écritures en cours sont recopiées aussi (`actively synced mode`)
② La mémoire	`starting online/live migration` , `downtime 3 ms`	Comme à chaud sur un stockage partagé : la mémoire passe, puis la VM est figée 3 ms et bascule
③ La fin	`Completed successfully` , `duration 00:00:14`	La copie du disque est close ; l'ancien volume est supprimé sur la source

14 secondes au lieu de 6 pour la migration à froid de l'atelier 2, mais la VM n'a jamais été arrêtée. Avec un vrai disque de 100 Gio, c'est la copie du disque qui dicterait la durée : c'est ce que Ceph supprimera à l'atelier 5.

> L' `uptime` affiché par Proxmox repart de zéro après la bascule (`uptime: 16` sur `pxe2<N>3`). Ce n'est pas un redémarrage : ce compteur suit le processus QEMU, et la VM tourne désormais dans un nouveau processus, sur la cible.

Étape 3 — Créer un conteneur à déplacer

En interface graphique. Sur `pxe2<N>1` , **Create CT** :

Onglet	Champ	Valeur
General	CT ID / Hostname	`330` / `ct-migration`
	Unprivileged container / Nesting	✓ / ✓
	Password	celui du lab
Template	Storage / Template	`local` / `debian-13-standard_…`
Disks	Storage / Disk size	`local-lvm` / `4`
CPU / **Memory**	Cores / Memory	`1` / `512`
Network	Bridge / IPv4 / Gateway	`vmbr0` / `10.<site>.<N>.60/24` / `10.<site>.<N>.1`

Ce support est la propriété de Dawan France.

Onglet	Champ	Valeur
Confirm	Start after created	✓

En ligne de commande.

Dans le Shell de `pxe2<N>1` :

```
# Un conteneur Debian minimal, en adresse fixe
pct create 330 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst \
  --hostname ct-migration --memory 512 --cores 1 \
  --rootfs local-lvm:4 \
  --net0 name=eth0,bridge=vmbr0,ip=10.<site>.<N>.60/24,gw=10.<site>.<N>.1 \
  --unprivileged 1 --features nesting=1 \
  --password
pct start 330     # le démarrer
pct status 330    # status: running
```

`--password` sans valeur demande le mot de passe au clavier. `--features nesting=1` évite l'avertissement `Systemd 257 detected. You may need to enable nesting.`, obtenu sans cette option : le système Debian 13 du conteneur en a besoin.

Étape 4 — Un ping continu pour mesurer la coupure

Dans le Shell de `pxe2<N>3` (interface web de `pxe2<N>3` → **Shell**) :

```
ping -i 0.5 10.<site>.<N>.60    # une requête toutes les 0,5 s ; laisser tourner
```

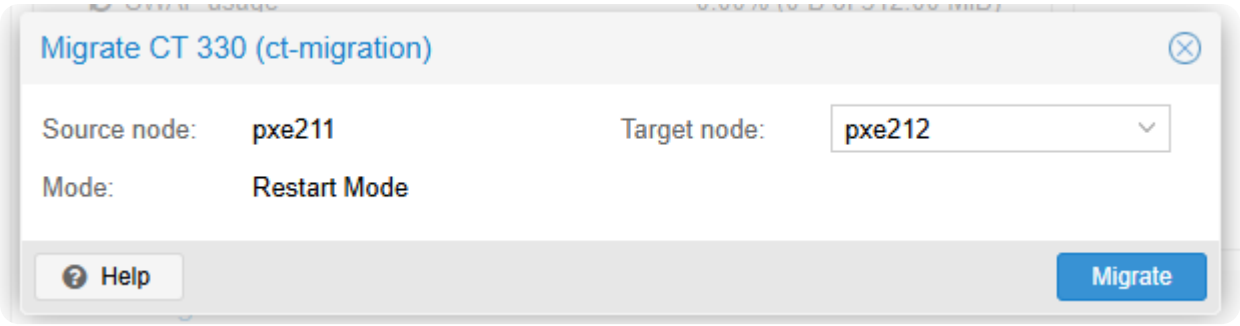
Étape 5 — Migrer le conteneur

Dans le Shell de `pxe2<N>1` :

```
pct migrate 330 pxe2<N>2 --online     # essai de migration à chaud
pct migrate 330 pxe2<N>2 --restart    # arrêt, déplacement, redémarrage
```

En interface graphique : CT 330 → **Migrate** → **Target node** `pxe2<N>2`.

Ce support est la propriété de Dawan France.



Pour un conteneur allumé, la fenêtre impose **Mode : Restart Mode** : aucune option ne permet de le déplacer sans l'arrêter.

Résultat attendu :

```
ERROR: migration aborted (duration 00:00:00): lxc live migration is currently not
implemented
```

```
shutdown CT 330
starting migration of CT 330 to node 'pxe212' (10.59.1.12)
found local volume 'local-lvm:vm-330-disk-0' (in current VM config)
4294967296 bytes (4.3 GB, 4.0 GiB) copied, 5.95972 s, 721 MB/s
successfully imported 'local-lvm:vm-330-disk-0'
start container on target node
migration finished successfully (duration 00:00:11)
```

Arrêtez le `ping` (**Ctrl+C**) et lisez sa fin. Sur le lab, les numéros `icmp_seq` sautaient de 9 à 31 : 21 réponses perdues à 0,5 s d'intervalle, soit **environ 10 secondes sans service**. C'est le temps d'arrêter le conteneur, de copier ses 4 Gio et de le redémarrer.

Pourquoi pas à chaud ? Une VM est un seul processus, dont on sait copier la mémoire. Un conteneur, ce sont les processus du noyau de l'hôte, avec leurs fichiers ouverts et leurs connexions : rien qu'on puisse transporter vers un autre noyau (cours, §2.1).

Étape 6 — Remettre en état

Dans le Shell de `pxe2<N>2` :

```
pct stop 330                # arrêter le conteneur
pct destroy 330 --purge     # le supprimer
```

Ce support est la propriété de Dawan France.

Dans le Shell de `pxe2<N>3` :

```
qm stop 120                  # la VM 120 n'a pas de système : un arrêt brutal suffit
qm migrate 120 pxe2<N>2      # migration à froid, retour sur pxe2<N>2
```

Si vous avez installé Debian dans la VM 120 (option B de l'atelier 2), préférez `qm shutdown 120`. Sans système, `qm shutdown` attend une réponse qui ne vient pas, puis échoue sur `got timeout`.

Validation finale

Contrôle	Résultat attendu
Migration à chaud sans option	refusée : `can't live migrate attached local disks without with-local-disks option`
Migration à chaud avec `--with-local-disks`	réussie, `downtime` de quelques millisecondes, disque recopié d'abord
Conteneur avec `--online`	refusé : `lxc live migration is currently not implemented`
Conteneur avec `--restart`	réussi, une dizaine de secondes de coupure au `ping`
État final	VM 120 éteinte sur `pxe2<N>2` ; plus de CT 330

Atelier 4 — Le quorum à l'épreuve

Contexte

Sur le lab, un nœud isolé s'est figé tandis que les deux autres continuaient (cours, §2.2). On va plus loin : on prive votre nœud `pxe2<N>1` de ses **deux** voisins, l'un après l'autre, et l'on observe ce qu'il peut encore faire.

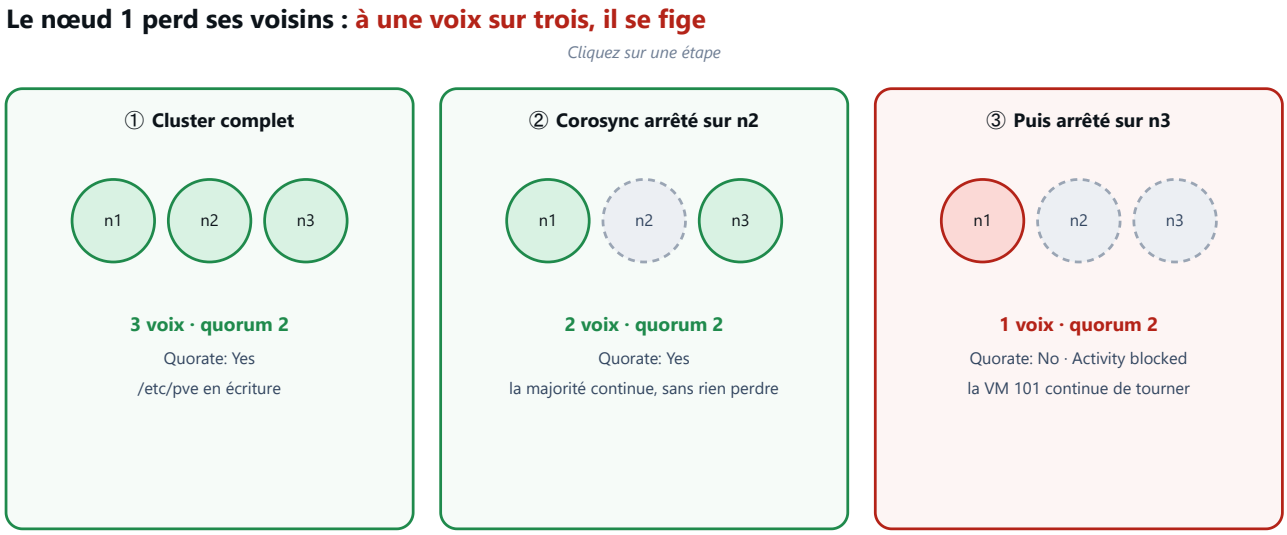
La panne est simulée en arrêtant Corosync sur les voisins, depuis leur propre Shell. Elle est donc **réversible** : on relance Corosync, et tout revient. On ne coupe aucun nœud et l'on ne touche à aucun réseau : un nœud éteint ne se rallume pas depuis l'interface web.

Résultat à obtenir : l'état du quorum sur `pxe2<N>1` avec trois, deux, puis une seule voix. La preuve qu'à deux voix sur trois, le cluster écrit encore. À une voix, `/etc/pve` passe en lecture

Ce support est la propriété de Dawan France.

seule, la création et le démarrage de VM sont refusés, mais la VM 101 continue de tourner.

Les événements correspondants sont lus dans le journal de Corosync. Enfin, le quorum revient sans rien forcer.



Les trois étapes de l'atelier, vues depuis `pxe2<N>1`. Le quorum reste fixé à deux voix ; seul change le nombre de voix présentes. Cliquez sur une étape pour lire le résultat.

Arborescence attendue

```
Cluster « cluster-tp »   (3 voix attendues, quorum = 2)
├── pxe2<N>1   ← on observe ici : la VM 101 y tourne
├── pxe2<N>2   ← Corosync arrêté en premier   → 2 voix sur 3 : quorate
└── pxe2<N>3   ← Corosync arrêté ensuite      → 1 voix sur 3 : figé
Retour : Corosync relancé sur pxe2<N>2, puis pxe2<N>3 → 3 voix
```

Ce que vous devez faire

1. **Relever** l'état de départ sur `pxe2<N>1` : `pvecm status`, et l'`uptime` de la VM 101 (démarrez-la si besoin).
2. **Arrêter Corosync** sur `pxe2<N>2`. Sur `pxe2<N>1` : relever le quorum, puis tenter une écriture dans `/etc/pve`.

Ce support est la propriété de Dawan France.

3. **Arrêter Corosync** sur `pxe2<N>3`. Sur `pxe2<N>1` : relever le quorum, tenter une écriture, créer une VM, démarrer une VM éteinte, et vérifier que la VM 101 tourne toujours.
4. **Lire** le journal de Corosync sur `pxe2<N>1`.
5. **Relancer Corosync** sur `pxe2<N>2`, puis sur `pxe2<N>3`, et vérifier le retour au quorum.

> **Ne jamais utiliser `pvecm expected 1` pour « débloquer » le nœud resté seul.** La commande abaisse le nombre de voix attendues : le nœud isolé se croit majoritaire.
>
> Si les autres nœuds tournent toujours de leur côté, deux parties du cluster écrivent en même temps : c'est le *split-brain* (cours, §2.4). Ici, la bonne réponse est de rétablir Corosync.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> Sur le lab (`pxe211` à `pxe213`). Chez vous, les nœuds s'appellent `pxe2<N>1` à `pxe2<N>3` . Ouvrez trois onglets, un Shell par nœud : l'interface web de `pxe2<N>1` restera utilisable pour lire, pas pour modifier.

Étape 1 — L'état de départ

Dans le Shell de `pxe2<N>1` :

```
qm start 101                                          # la VM 101 doit tourner ; si elle tourne déjà, qm le dit, sans conséquence
qm status 101 --verbose   # état et durée de fonctionnement ; à lire : status, uptime
pvecm status   # les voix du cluster ; à lire : Nodes, Expected votes, Total votes, Quorum, Quorate
```

Résultat attendu (les lignes qui comptent) :

```
status: running
uptime: 271
Nodes:            3
Quorate:          Yes
Expected votes:   3
Total votes:      3
Quorum:           2
```

Trois voix attendues, trois présentes ; il en faut deux (`Quorum: 2`) pour décider.

Étape 2 — Un voisin perdu : deux voix sur trois

Dans le Shell de `pxe2<N>2` :

```
systemctl stop corosync    # ce nœud ne parle plus au cluster
```

Dans le Shell de `pxe2<N>1` :

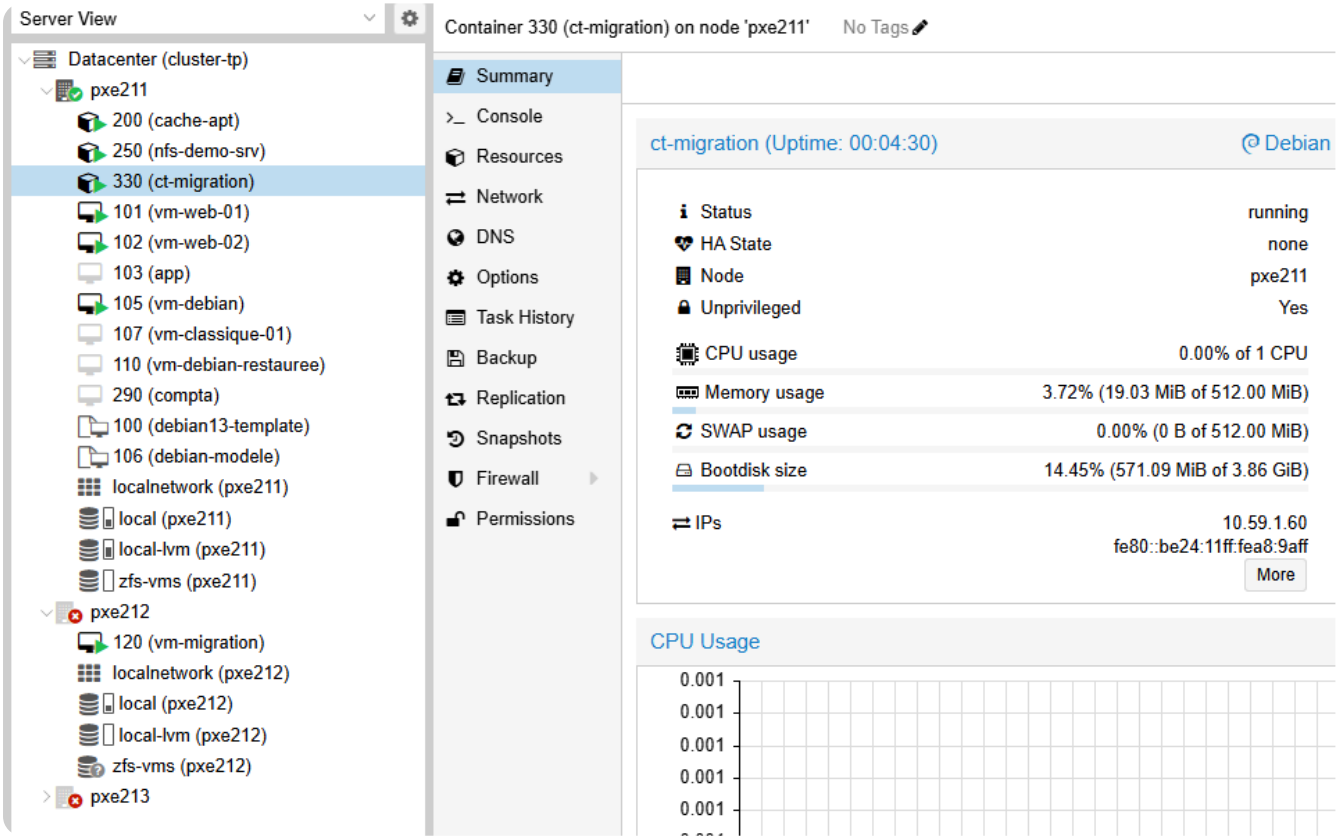
Dawan Ce support est la propriété de Dawan France.

```
pvecm status   # deux voix sur trois ; à lire : Nodes, Total votes, Quorum,
Quorate
touch /etc/pve/essai-2-voix        # écrire dans /etc/pve
echo "code : $?"                   # 0 : écriture acceptée
rm /etc/pve/essai-2-voix           # retirer le fichier d'essai
```

Résultat attendu (les lignes qui comptent) :

```
Nodes:            2
Quorate:          Yes
Total votes:      2
Quorum:           2
code : 0
```

Deux voix sur trois, c'est encore la majorité : le cluster continue sans rien perdre. Dans l'interface web, `pxe2<N>2` apparaît avec une croix rouge.



L'arbre de l'interface de `pxe2<N>1` après l'étape 3 : les deux voisins portent une croix rouge, leurs machines restent listées. Ce que l'interface ne montre pas d'un coup d'œil, c'est que

Ce support est la propriété de Dawan France.

`pxe2<N>1` a perdu le quorum : il faut le lire dans `pvecm status`, ou dans **Datacenter** → **Summary**.

Étape 3 — Le second voisin perdu : une voix sur trois

Dans le Shell de `pxe2<N>3` :

```
systemctl stop corosync   # le second voisin quitte le cluster
```

Dans le Shell de `pxe2<N>1` :

```
pvecm status   # une voix sur trois ; à lire : Nodes, Expected votes, Total votes, Quorum, Quorate
touch /etc/pve/essai-1-voix                              # écrire dans /etc/pve
qm create 199 --name essai-sans-quorum --memory 256      # créer une VM
qm start 105                                             # démarrer une VM éteinte
qm status 101 --verbose   # la VM qui tournait ; à lire : status, uptime
```

Résultat attendu (les lignes qui comptent) :

```
Nodes:            1
Quorate:          No
Expected votes:   3
Total votes:      1
Quorum:           2 Activity blocked
touch: cannot touch '/etc/pve/essai-1-voix': Permission denied
cluster not ready - no quorum?
cluster not ready - no quorum?
status: running
uptime: 286
```

Ce qu'on tente	Réponse	Pourquoi
Écrire dans `/etc/pve`	`Permission denied`	Le nœud minoritaire passe la configuration commune en lecture seule
Créer une VM	`cluster not ready - no quorum?`	Une création écrit une configuration dans `/etc/pve`

Ce support est la propriété de Dawan France.

Ce qu'on tente	Réponse	Pourquoi
Démarrer une VM	`cluster not ready - no quorum?`	Le démarrage est refusé : la même VM pourrait être lancée ailleurs, par la partie majoritaire
La VM 101 déjà lancée	`status: running`, `uptime` qui continue	Rien ne l'arrête : le quorum protège la configuration, pas les machines qui tournent

Étape 4 — Ce que dit le journal

Dans le Shell de `pxe2<N>1` :

```
journalctl -u corosync --no-pager -n 8    # les 8 derniers événements de Corosync
```

Résultat attendu :

```
[QUORUM] This node is within the non-primary component and will NOT provide any
services.
[QUORUM] Members[1]: 1
[MAIN  ] Completed service synchronization, ready to provide service.
[KNET  ] link: host: 3 link: 0 is down
[KNET  ] host: host: 3 (passive) best link: 1 (pri: 1)
[KNET  ] link: host: 3 link: 1 is down
[KNET  ] host: host: 3 (passive) best link: 1 (pri: 1)
[KNET  ] host: host: 3 has no active links
```

- **`non-primary component`** : ce nœud est du côté minoritaire, il ne rendra plus de service de cluster.
- **`Members[1]: 1`** : il ne voit plus que lui-même (le nœud n°1).
- **`link: 0 is down`, puis `link: 1 is down`** : les deux liens vers le nœud 3 tombent l'un après l'autre. Avec un seul lien perdu, Corosync aurait basculé sur l'autre (`best link: 1`) sans perdre le nœud.

Étape 5 — Revenir au quorum

Dans le Shell de `pxe2<N>2`, puis dans celui de `pxe2<N>3` :

```
systemctl start corosync   # le nœud revient dans le cluster
```

Ce support est la propriété de Dawan France.

Dans le Shell de `pxe2<N>1`, après le premier redémarrage puis après le second :

```
pvecm status   # les voix revenues ; à lire : Nodes, Total votes, Quorate
```

Résultat attendu (les lignes qui comptent) :

```
Nodes:            2
Quorate:          Yes
Total votes:      2
[…]
Nodes:            3
Quorate:          Yes
Total votes:      3
```

Dès que `pxe2<N>2` revient, le journal annonce `This node is within the primary component and will provide service.` : avec deux voix, `pxe2<N>1` retrouve l'écriture. Aucune VM 199 n'a été créée pendant la panne (`ls /etc/pve/qemu-server/` ne la montre pas).

Validation finale

Dans le Shell de `pxe2<N>1` :

```
pvecm status | grep Quorate                    # Quorate: Yes
corosync-cfgtool -s | grep -c connected        # 4 : deux liens vers deux voisins
touch /etc/pve/essai                           # /etc/pve accessible en écriture ?
echo "code : $?"                               # 0 : oui
rm /etc/pve/essai                              # retirer le fichier d'essai
```

Résultat attendu : `Quorate: Yes` , `4` , `code : 0` . Le cluster est revenu à son état de départ, sans rien forcer.

Atelier 5 — Déploiement de Ceph Tentacle

Contexte

Ce support est la propriété de Dawan France.

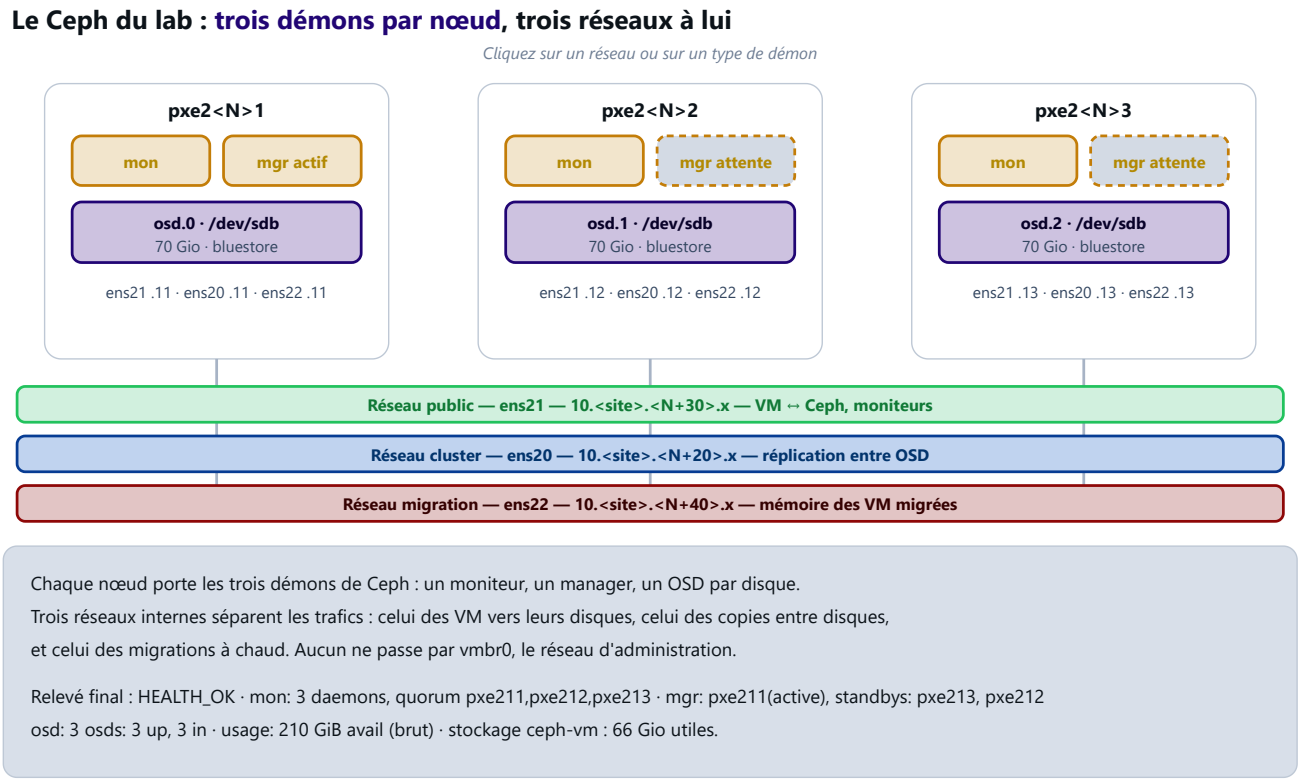
À l'atelier 2, migrer une VM a coûté la copie de tout son disque. **Ceph** supprime cette copie.

C'est un stockage distribué : chaque donnée existe en trois exemplaires, un par nœud. Le disque d'une VM est donc déjà partout, et seule sa mémoire doit voyager.

Ceph a besoin de deux réseaux à lui. Le **réseau public** transporte le trafic entre les VM et Ceph. Le **réseau cluster** transporte la réplication entre les disques.

Vos cartes `ens20` et `ens21` sont là pour eux. La carte `ens22`, elle, portera la migration à chaud.

Résultat à obtenir : Ceph sain (`HEALTH_OK`) sur les trois nœuds, un pool répliqué déclaré en stockage, et une VM migrée **à chaud** par le réseau de migration.



Ce que vous allez déployer. Sur chaque nœud, un moniteur, un manager et un OSD.

En dessous, les trois réseaux internes, chacun sur sa carte. Cliquez sur un réseau ou sur un type de démon pour lire son rôle et le résultat obtenu.

Arborescence attendue

```
Réseaux internes (une adresse par nœud : .11, .12, .13)
├── ens20 → Ceph cluster  10.<site>.<N+20>.x   réplication entre OSD
```

Ce support est la propriété de Dawan France.

```
├── ens21 → Ceph public   10.<site>.<N+30>.x   VM ⇔ Ceph, moniteurs
└── ens22 → migration     10.<site>.<N+40>.x   mémoire des VM migrées
Ceph Tentacle (20.2)
├── Monitors : 1 par nœud   (quorum de Ceph, distinct de celui de Corosync)
├── Managers : 1 par nœud   (1 actif, 2 en attente)
├── OSD      : 1 par nœud, sur /dev/sdb
└── pool « ceph-vm » (3 copies, 2 minimum) → stockage RBD partagé
VM 300 « vm-ha-test » sur ceph-vm  ──[ migration à chaud ]──▶  pxe2<N>2   ← seule
la mémoire voyage
```

Ce que vous devez faire

1. **Adresser** `ens20`, `ens21` et `ens22` sur les trois nœuds.
2. **Installer Ceph** sur les trois nœuds, puis l'**initialiser** sur les réseaux dédiés.
3. **Créer un Monitor et un Manager par nœud**, puis **un OSD par nœud** sur `/dev/sdb`.
4. **Vérifier** `HEALTH_OK`, puis **créer un pool répliqué** déclaré en stockage.
5. **Désigner le réseau de migration**, puis **migrer à chaud** une VM posée sur Ceph. Comparez avec l'atelier 2.

   ⚠️ Ceph exige un disque totalement vierge pour ses OSD. Si un pool ZFS occupe `sdb`, il faudra le détruire d'abord (étape 4 bis).

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 0 — Adresser les réseaux de Ceph et de migration

Ces trois réseaux n'ont pas besoin de pont : aucune VM ne s'y branche. L'adresse se pose directement sur la carte.

Carte	Rôle	`pxe2<N>1`	`pxe2<N>2`	`pxe2<N>3`	Comment
`ens20`	Ceph cluster	`10.<site>.<N+20>.11/24`	`….12/24`	`….13/24`	`CEPH-CLUSTER`
`ens21`	Ceph public	`10.<site>.<N+30>.11/24`	`….12/24`	`….13/24`	`CEPH-PUBLIC`
`ens22`	Migration	`10.<site>.<N+40>.11/24`	`….12/24`	`….13/24`	`MIGRATION`

Avant tout Apply, le filet du chapitre 3 (atelier 1, étape 1 bis).

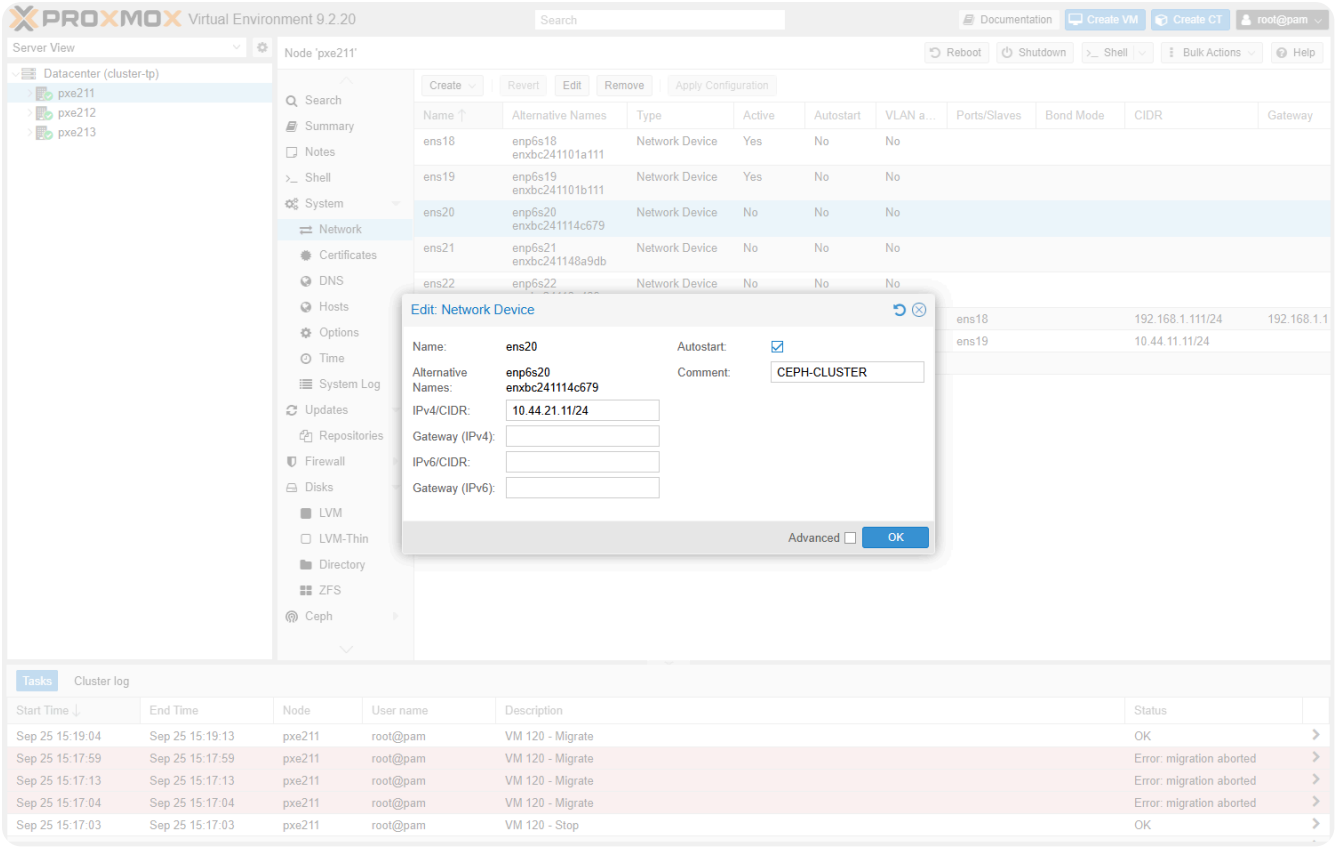
Dans le Shell de chaque nœud dont vous modifiez le réseau :

```
# Copie de la configuration réseau qui fonctionne, puis retour automatique dans 10 minutes
cp /etc/network/interfaces /root/interfaces.avant
systemd-run --on-active=10min --unit=filet-reseau sh -c "cp /root/interfaces.avant /etc/network/interfaces && ifreload -a"   # si l'accès est perdu, restaurer la copie dans 10 min
```

Une fois l'accès et les pings vérifiés, annulez-le sur chacun : `systemctl stop filet-reseau.timer`.

Sur chaque nœud : **Network** → double-clic sur la carte → IPv4/CIDR, Autostart coché, Comment → **OK**. Une fois les trois cartes faites : **Apply Configuration**.

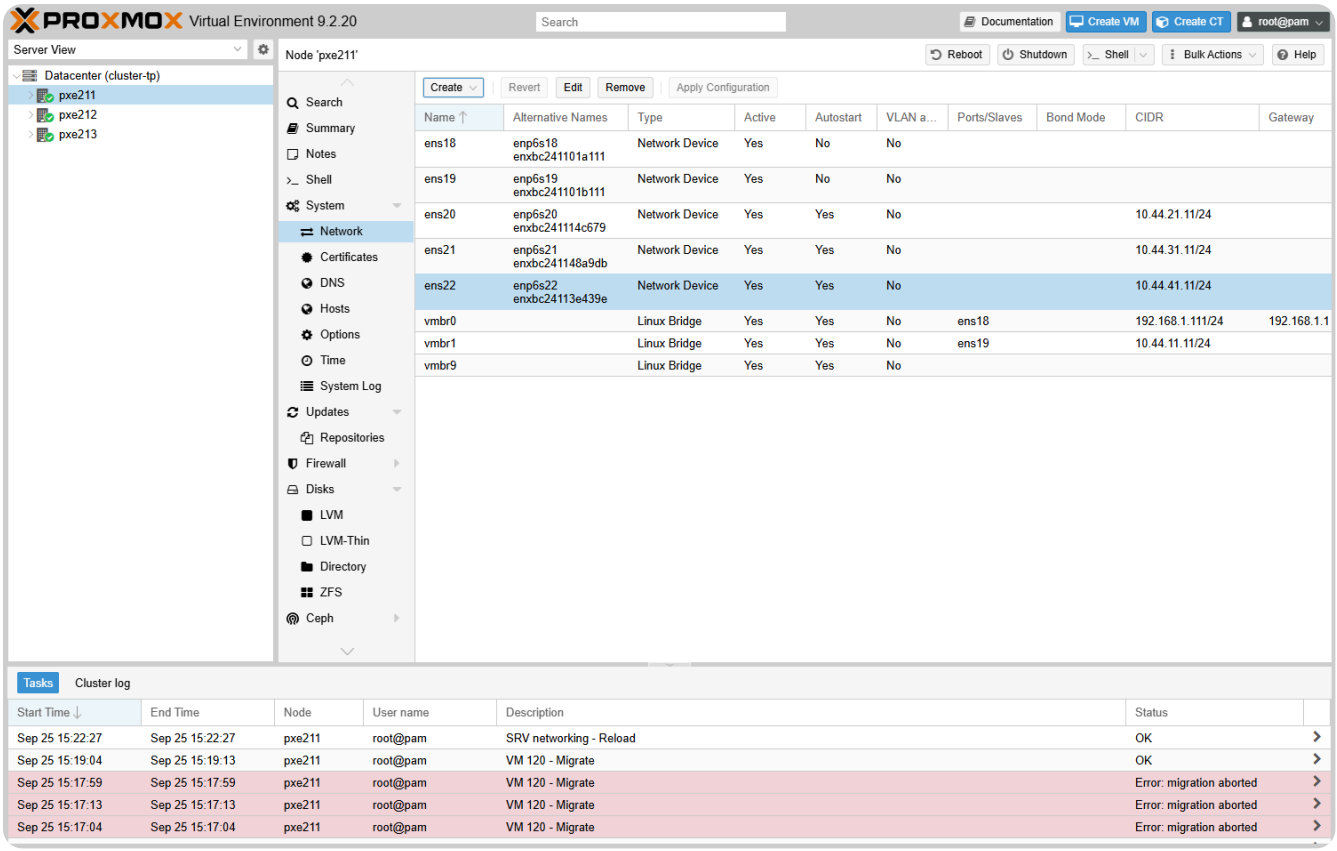
Ce support est la propriété de Dawan France.



(`pxe211`). La fenêtre **Edit: Network Device** de `ens20` .

Pas de passerelle : ces réseaux ne sortent pas. **Autostart** coché : sans lui, la carte resterait éteinte au prochain démarrage.

Ce support est la propriété de Dawan France.



Après **Apply Configuration**, les cinq cartes sont actives. `ens20`, `ens21` et `ens22` portent leur adresse en direct ; `ens18` et `ens19` la portent par leur pont (`vmbr0`, `vmbr1`).

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Depuis pxe2<N>1 : les deux autres nœuds, sur chacun des trois réseaux
ping -c 2 -q 10.<site>.<N+20>.12   # le nœud 2, réseau Ceph cluster
ping -c 2 -q 10.<site>.<N+20>.13   # le nœud 3
ping -c 2 -q 10.<site>.<N+30>.12   # le nœud 2, réseau Ceph public
ping -c 2 -q 10.<site>.<N+30>.13   # le nœud 3
ping -c 2 -q 10.<site>.<N+40>.12   # le nœud 2, réseau de migration
ping -c 2 -q 10.<site>.<N+40>.13   # le nœud 3
```

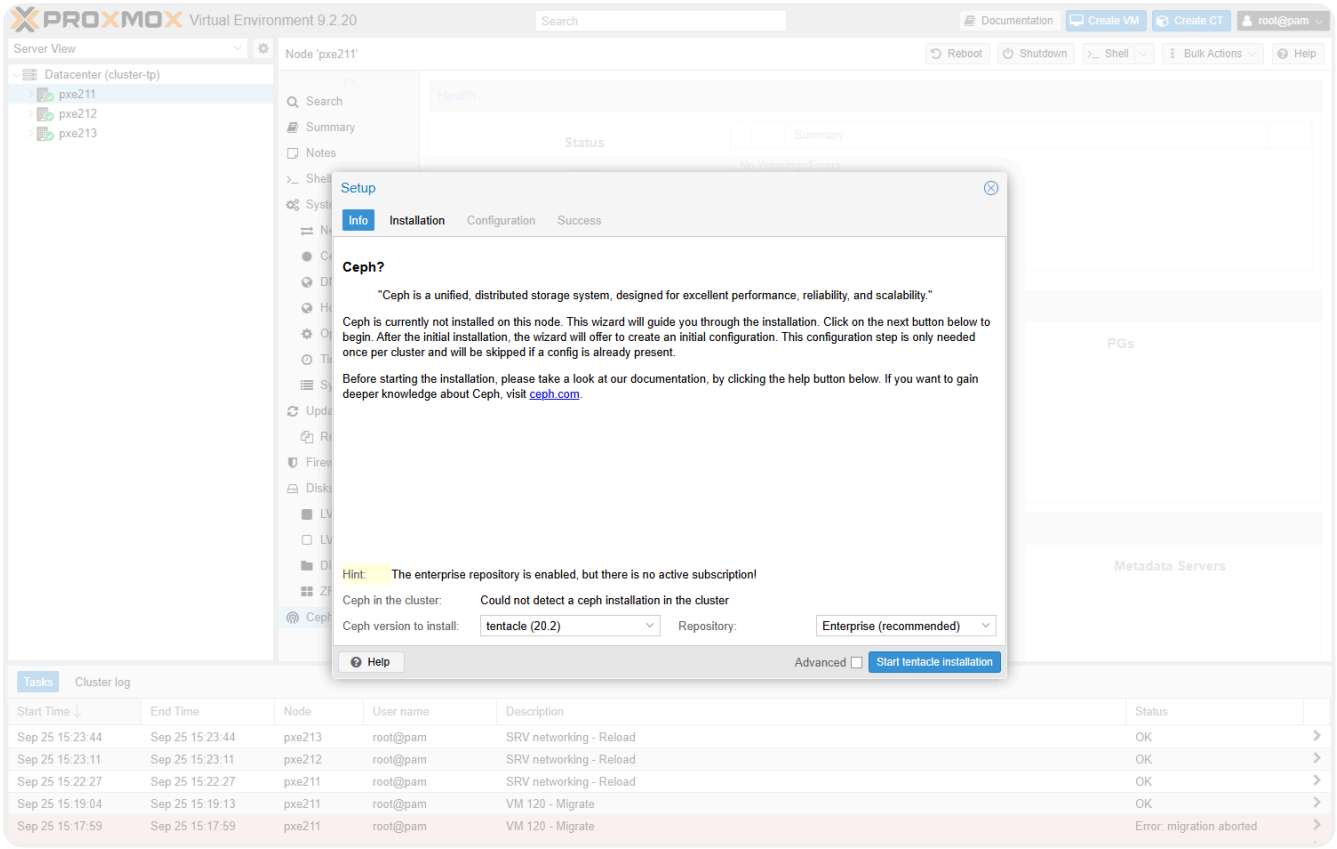
Remplacez `<N+20>`, `<N+30>` et `<N+40>` par leurs valeurs (21, 31 et 41 pour le stagiaire 1).

Résultat attendu : pour chaque ping, `2 packets transmitted, 2 received, 0% packet loss`, puis une ligne `rtt min/avg/max/mdev` sous la milliseconde (de 0,36 à 0,71 ms en moyenne).

Étape 1 — Installer Ceph

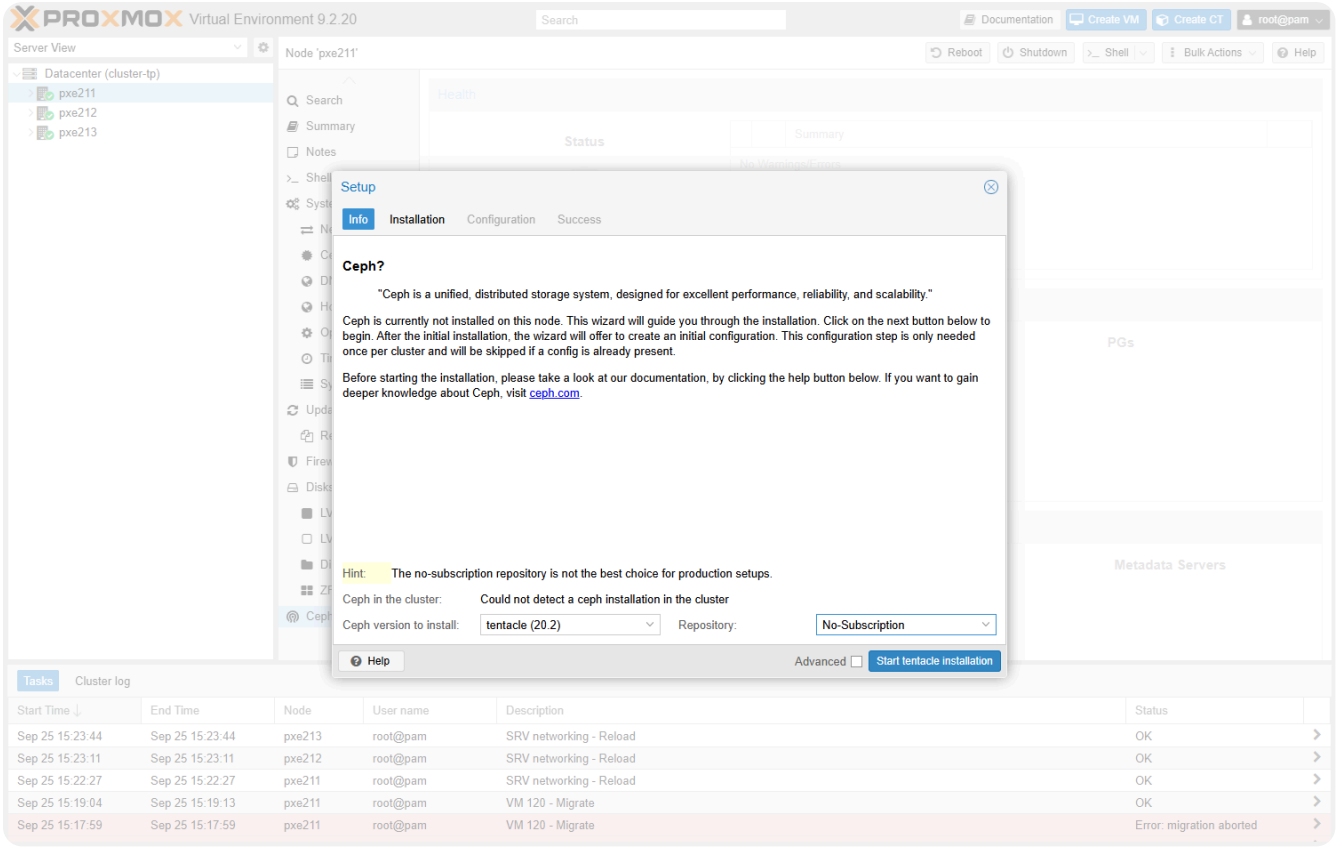
Ce support est la propriété de Dawan France.

Sur `pxe2<N>1`, par l'assistant. Le serveur → **Ceph**. Le panneau affiche *Ceph is not installed on this node* → **Install Ceph**.



L'assistant propose **tentacle (20.2)**, mais le dépôt **Enterprise (recommended)**. Sans abonnement, ce dépôt répond 401 et l'installation échoue. L'indice en bas le signale : *The enterprise repository is enabled, but there is no active subscription!*

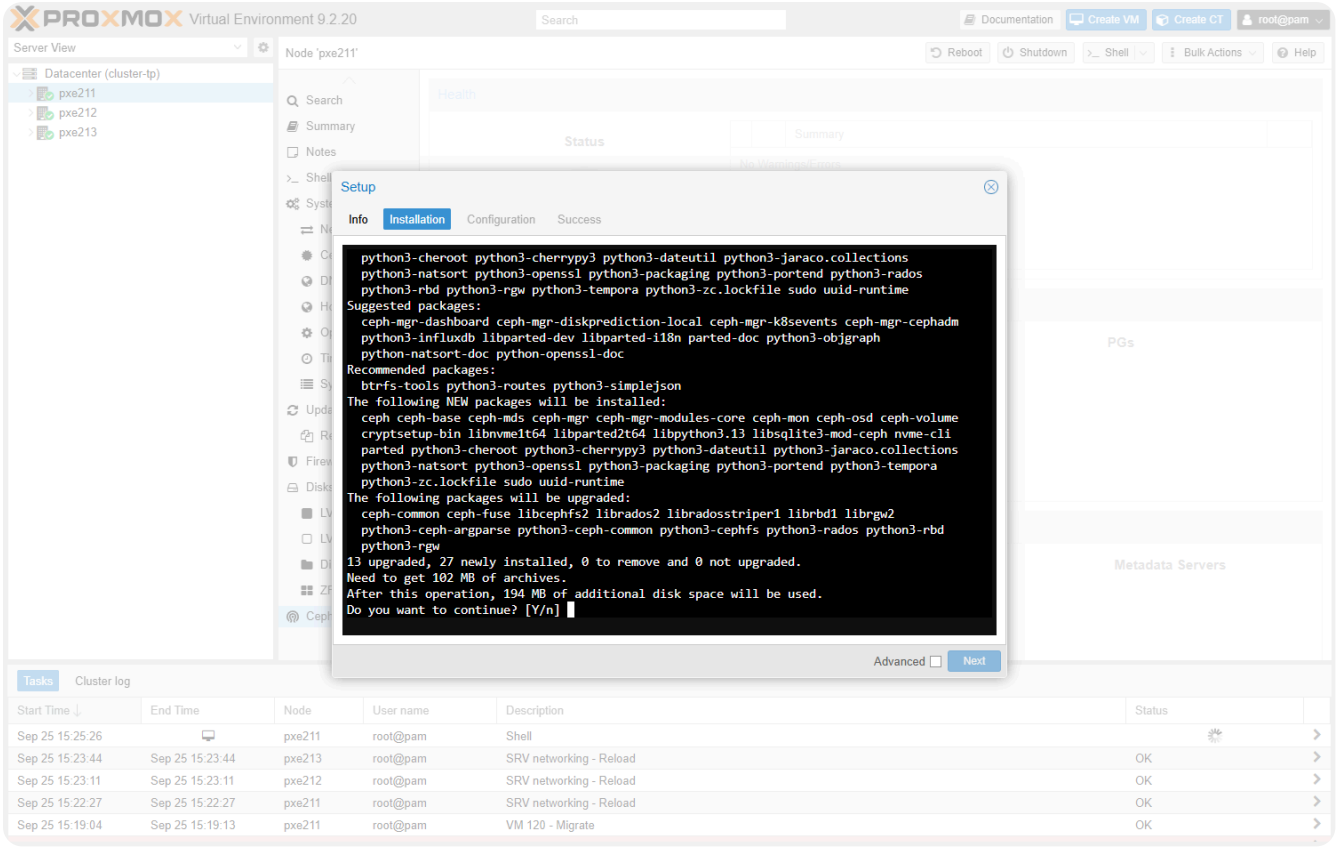
Ce support est la propriété de Dawan France.



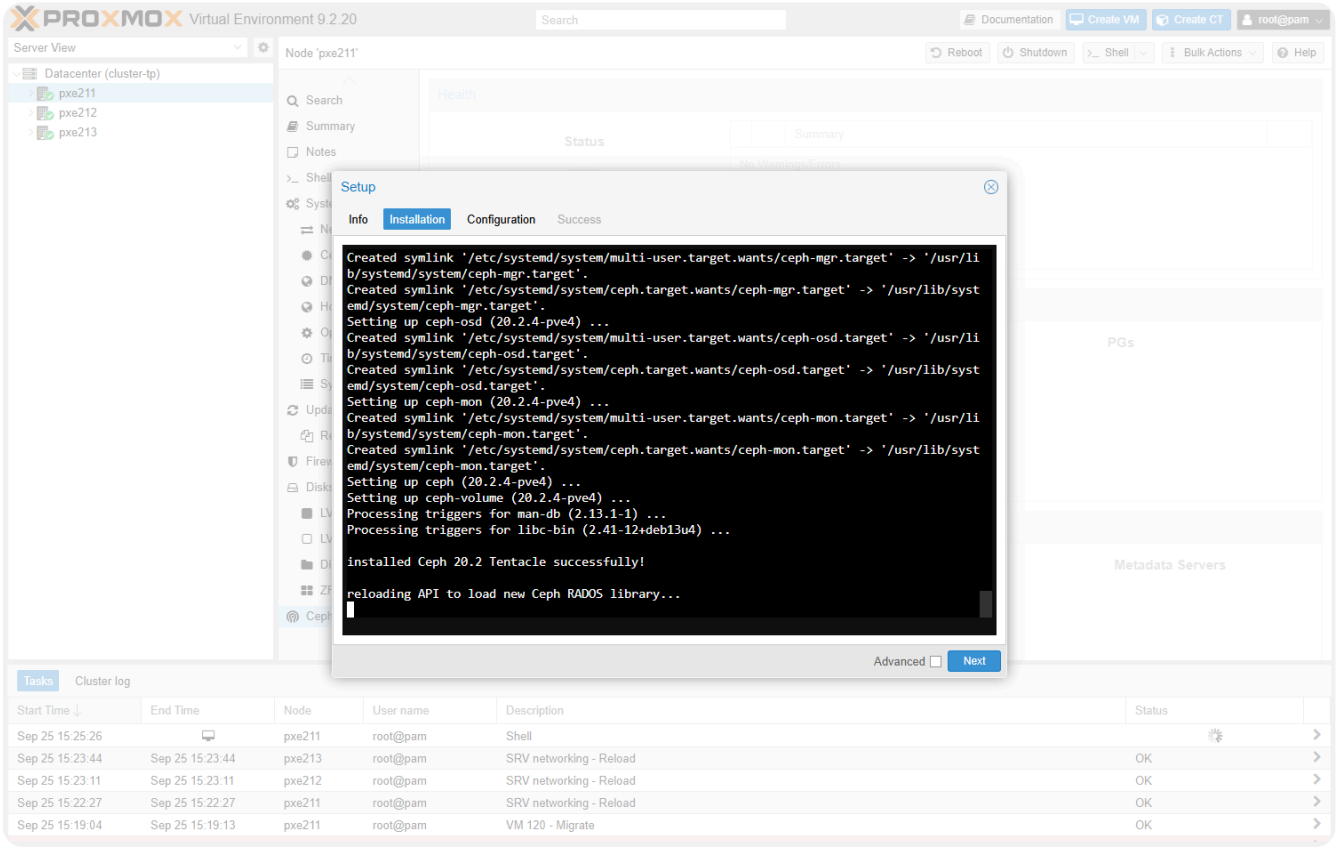
Repository passé sur **No-Subscription**. L'indice change : ce dépôt *n'est pas le meilleur choix en production*.

Pour le lab, c'est le bon. Cliquez **Start tentacle installation**.

Ce support est la propriété de Dawan France.



L'onglet **Installation** est un vrai terminal. apt liste les paquets (27 à installer, 13 à mettre à jour), puis **attend une réponse** : `Do you want to continue? [Y/n]`. Cliquez dans le terminal, tapez `y`, puis Entrée.



Ce support est la propriété de Dawan France.

Les paquets en version `20.2.4-pve4`, puis `installed Ceph 20.2 Tentacle successfully!`. Le bouton **Next** devient actif.

Sur `pxe2<N>2` et `pxe2<N>3`, en ligne de commande — dans le Shell de chaque nœud :

```
# La question d'apt reçoit « y » par le tube
yes | pveceph install --version tentacle --repository no-subscription
ceph --version   # version de Ceph installée
```

Résultat attendu :

```
installed Ceph 20.2 Tentacle successfully!

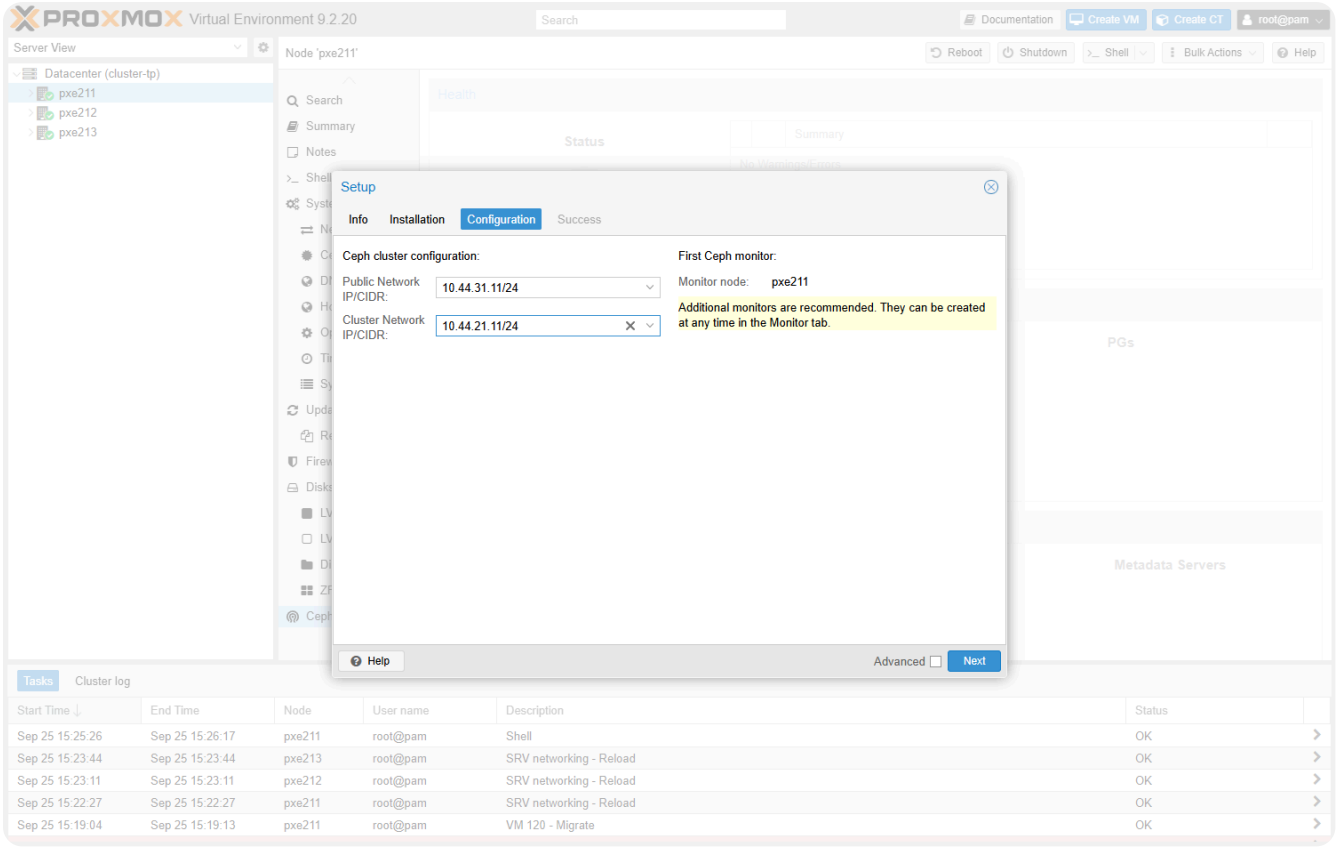
reloading API to load new Ceph RADOS library...
ceph version 20.2.4 (a12e197e4613a9d6be90e8fb073ac4fc3656aee2) tentacle (stable)
```

> **`apt failed during ceph installation (256)` : lisez la ligne au-dessus.** Sans le `yes |`, et sans terminal pour répondre, apt reçoit une réponse vide. Il abandonne : `Do you want to continue? [Y/n] Abort. Ce n'est pas le dépôt : relancez avec yes | . Si la ligne au-dessus parle de 401 ou de dépôt introuvable, c'est bien le dépôt : relancez avec --repository no-subscription . pveceph réécrit lui-même /etc/apt/sources.list.d/ceph.sources ; n'ajoutez pas de fichier .list` à côté.

Étape 2 — Initialiser Ceph sur ses réseaux

Avec l'assistant, cette étape suit l'installation : onglet **Configuration**. Si vous l'avez fermé, rouvrez **Ceph** sur `pxe2<N>1` : le panneau affiche *Ceph is not initialized* → **Configure Ceph**.

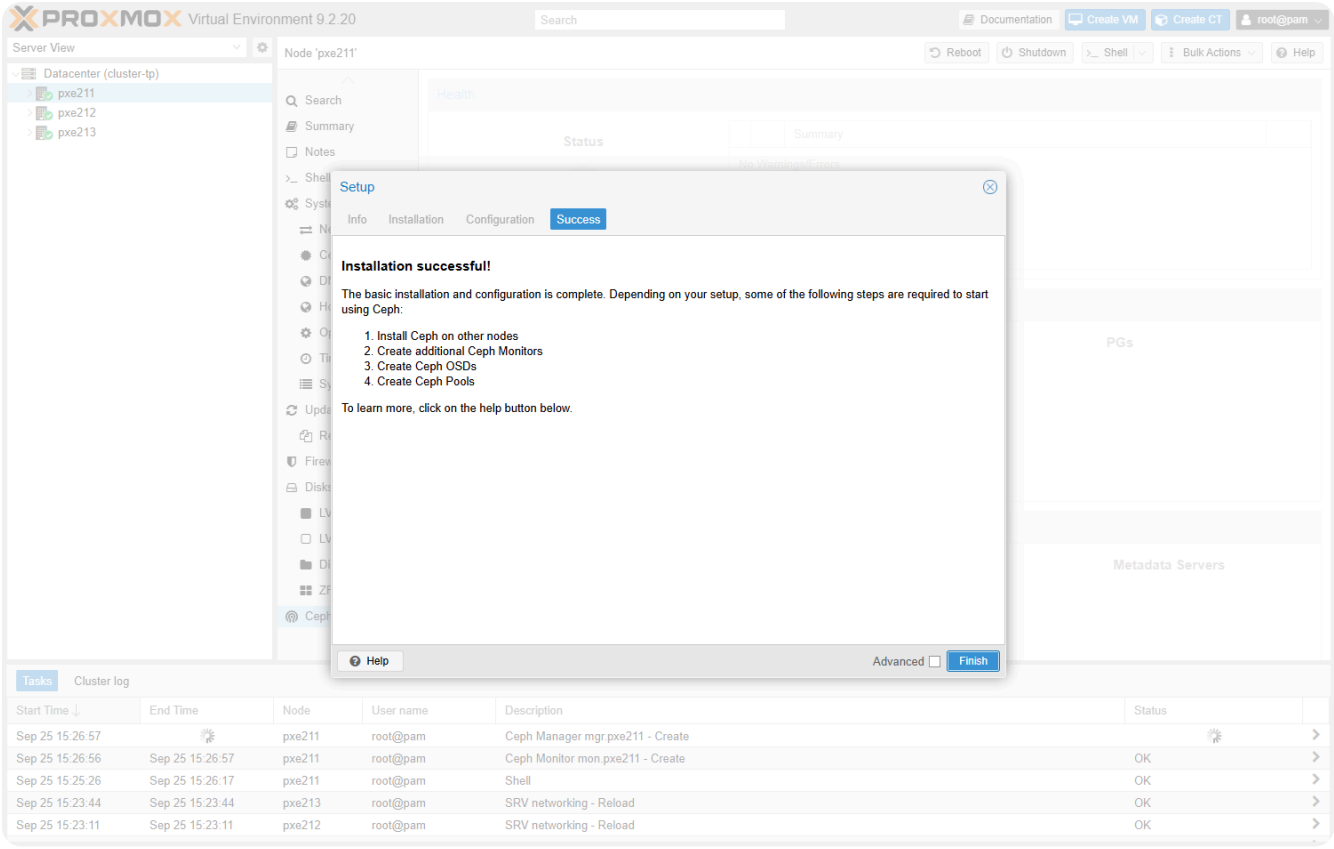
Ce support est la propriété de Dawan France.



Public Network : l'adresse de `ens21` (`10.44.31.11/24` sur le lab). **Cluster Network** : celle de `ens20` (`10.44.21.11/24`).

Par défaut, le second vaut *Same as Public Network* : laissé ainsi, la réplication partagerait le réseau des VM. À droite, le premier moniteur sera `pxe211` . → **Next**.

Ce support est la propriété de Dawan France.



Installation successful! et la suite à faire : installer Ceph sur les autres nœuds, créer les moniteurs, les OSD, les pools. → **Finish**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI, si vous n'avez pas utilisé l'assistant
pveceph init --network 10.<site>.<N+30>.0/24 --cluster-network 10.<site>.
<N+20>.0/24
pveceph mon create   # le premier moniteur
```

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Ce que l'assistant a écrit, et ce qui tourne déjà
grep -E "network|mon_host|pool_default" /etc/pve/ceph.conf
ceph -s   # état du cluster Ceph
```

Résultat attendu (lab, `pveceph init` en ligne de commande, réseaux `10.59.x`) :

```
    cluster_network = 10.59.21.0/24
    mon_host =  10.59.31.11
```

Ce support est la propriété de Dawan France.

```
    osd_pool_default_min_size = 2
    osd_pool_default_size = 3
    public_network = 10.59.31.0/24
...
    health: HEALTH_OK
    mon: 1 daemons, quorum pxe211 (age 6s) [leader: pxe211]
    mgr: pxe211(active, since 5s)
    osd: 0 osds: 0 up, 0 in
```

Par l'assistant, les deux réseaux s'écrivent avec l'adresse du nœud (`10.<site>.<N+20>.11/24`) ; en ligne de commande, avec l'adresse du réseau (`.0/24`). Ceph comprend les deux de la même façon : le réseau entier.

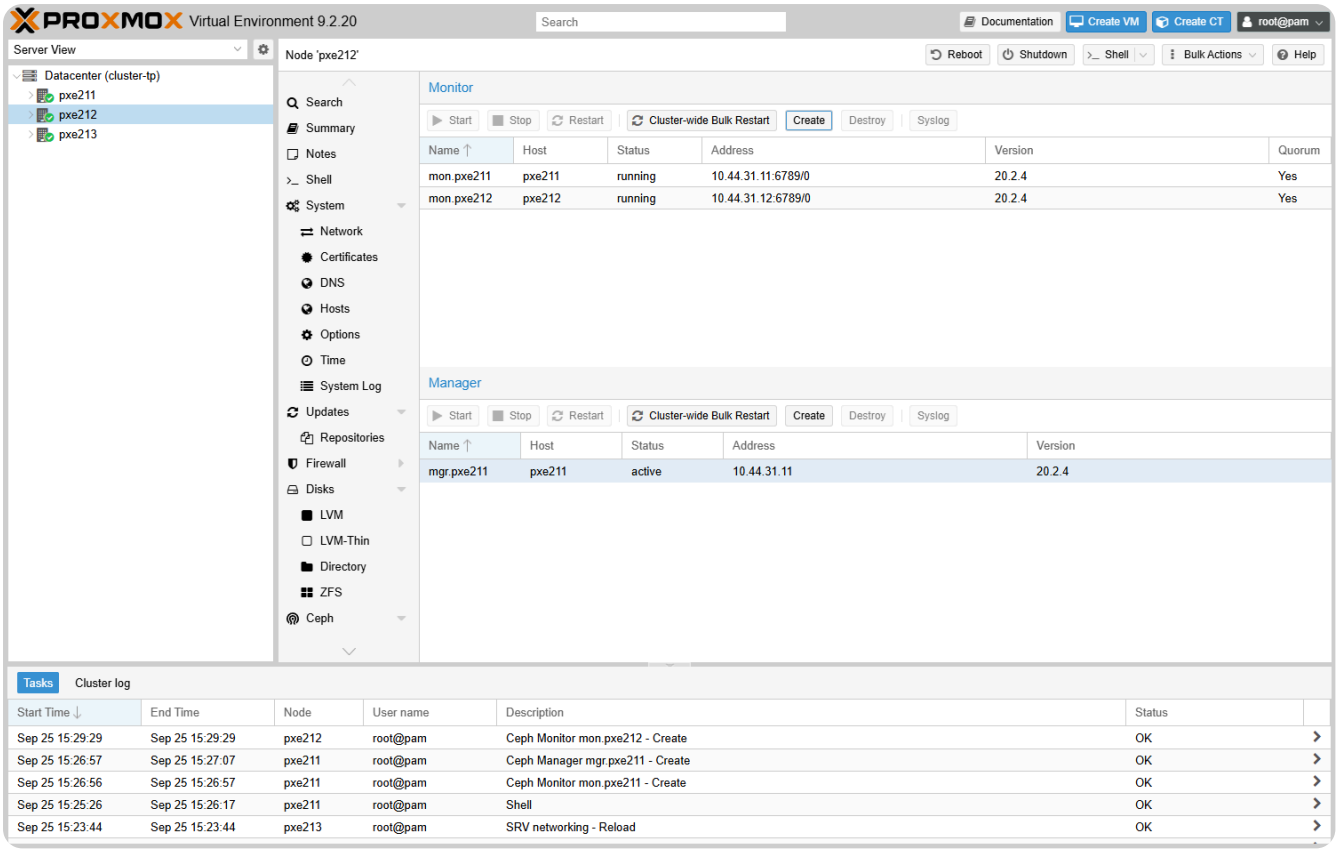
L'assistant, comme `pveceph mon create`, a créé d'un coup le **premier Monitor et le premier Manager**. Le moniteur écoute sur le réseau public. Les valeurs par défaut des pools sont fixées : 3 copies, 2 au minimum.

Étape 3 — Monitors et Managers sur les deux autres nœuds

Le **Monitor** tient la carte du cluster Ceph ; il en faut une majorité pour que Ceph fonctionne. Le **Manager** collecte les statistiques ; un seul est actif, les autres attendent.

Sur `pxe2<N>2` : **Ceph** → **Monitor** → section **Monitor** → **Create** → **Create**. Puis section **Manager** → **Create** → **Create**. Même chose sur `pxe2<N>3`.

Ce support est la propriété de Dawan France.



(`pxe212`). Deux moniteurs, `running` , **Quorum : Yes**, à l'écoute sur le port `6789` du réseau public.

Un seul manager pour l'instant, celui créé par l'assistant. La page a deux boutons **Create** : le premier pour les moniteurs, le second pour les managers.

Dans le Shell de chacun des deux autres nœuds (`pxe2<N>2` , puis `pxe2<N>3`) :

```
# Équivalent CLI, sur chaque nœud qui n'en a pas encore
pveceph mon create
pveceph mgr create   # un manager
ceph -s   # état du cluster Ceph
```

Résultat attendu, une fois les trois nœuds faits :

```
    health: HEALTH_WARN
            OSD count 0 < osd_pool_default_size 3

  services:
    mon: 3 daemons, quorum pxe211,pxe212,pxe213 (age 11s) [leader: pxe211]
```

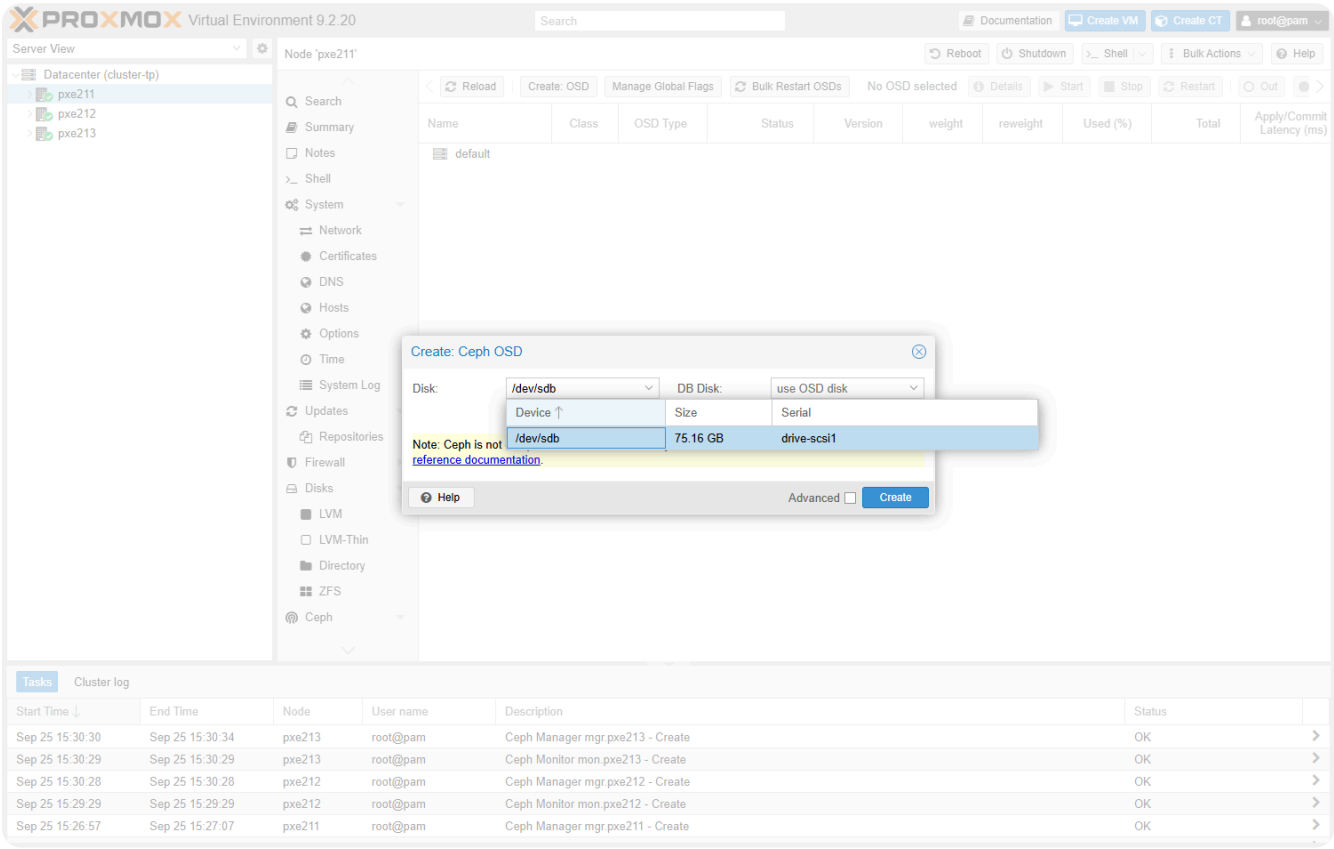
Ce support est la propriété de Dawan France.

```
    mgr: pxe211(active, since 34s), standbys: pxe212, pxe213
    osd: 0 osds: 0 up, 0 in
```

Ce `HEALTH_WARN` est attendu : les pools veulent 3 copies, et il n'y a encore aucun disque.

Étape 4 — Un OSD par nœud

L'**OSD** est le démon qui gère un disque. Sur chaque nœud : **Ceph** → **OSD** → **Create: OSD**.



(`pxe211`). La liste ne propose que les disques libres. Sur ce nœud, `sdc` et `sdd` portent le pool ZFS du chapitre 3 : seul `/dev/sdb` reste.

Sa taille, **75.16 GB**, correspond aux 70 Gio du disque, exprimés en Go. **DB Disk** : `use OSD disk` — tout sur le même disque, ce qui suffit au lab. → **Create**.

La liste des OSD ne se rafraîchit pas seule : cliquez **Reload** après la création.

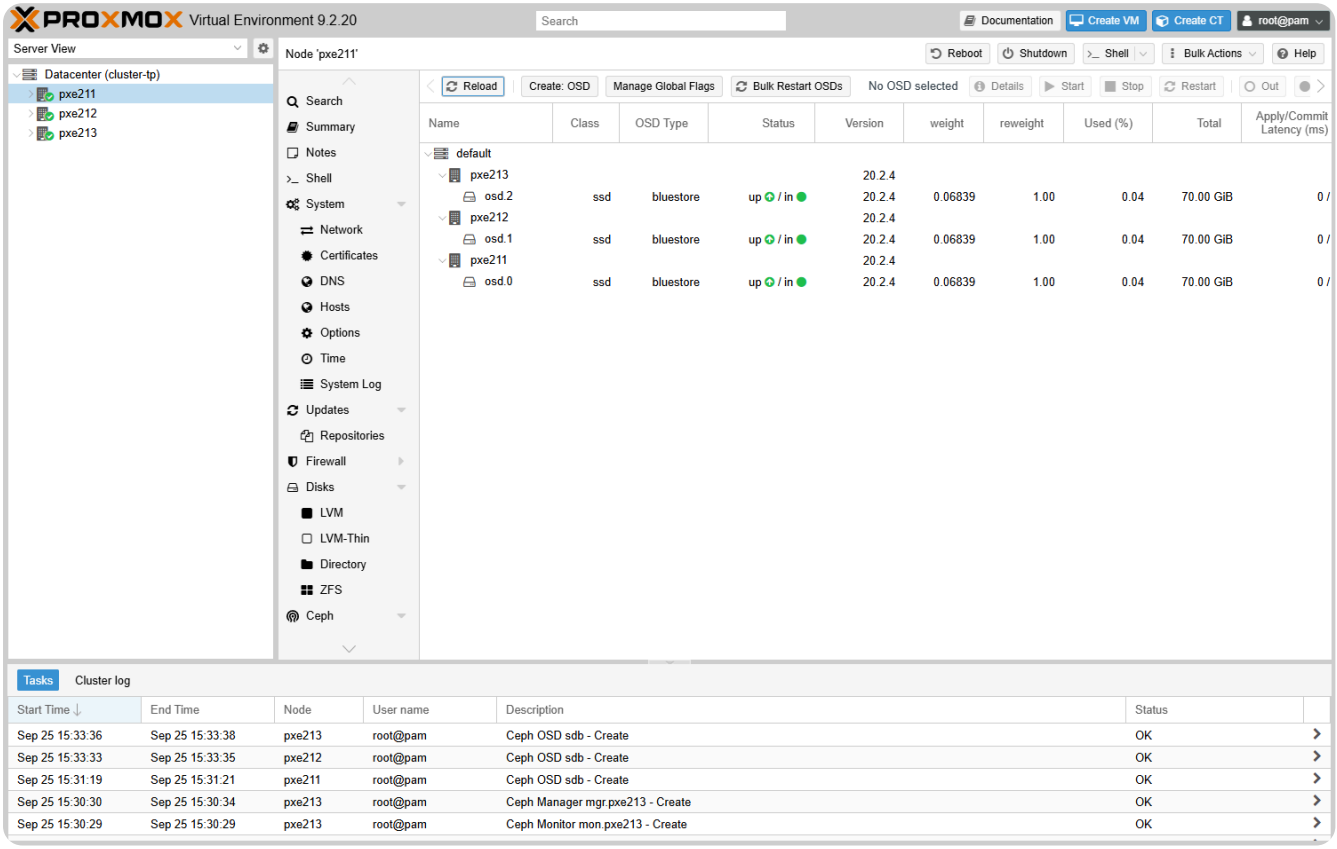
Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI, sur chaque nœud
pveceph osd create /dev/sdb
```

Ce support est la propriété de Dawan France.

Résultat attendu (`pxe212` , fin) :

```
Running command: /bin/systemctl start ceph-osd@1
--> ceph-volume lvm activate successful for osd ID: 1
--> ceph-volume lvm create successful for: /dev/sdb
```



Un OSD par nœud, `osd.0` à `osd.2` , tous **up / in**, type **bluestore**, 70 Gio chacun. La classe **ssd** vient du disque virtuel, que Ceph détecte comme non rotatif.

Étape 4 bis — Si `pveceph osd create` refuse le disque

Ceph exige un disque **totalement vierge**. La moindre signature — table de partition, LVM, ZFS, ancien OSD — fait échouer la création.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le disque destiné à Ceph porte-t-il encore une signature ?
lsblk /dev/sdb
wipefs /dev/sdb   # signatures présentes sur le disque
```

Ce support est la propriété de Dawan France.

Résultat attendu, disque vierge : `lsblk` ne montre que la ligne `sdb 8:16 0 70G 0 disk`, sans partition dessous, et `wipefs` n'affiche rien.

Si `wipefs` affiche des lignes, deux cas.

Cas A — un pool ZFS occupe le disque. Cela arrive si le pool du chapitre 3 a été monté sur `sdb` au lieu de `sdc` et `sdd`. On le démonte dans cet ordre :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# 1. Retirer le stockage côté Proxmox
pvesm remove zfs-vms

# 2. Détruire le pool — les données qu'il contient sont perdues
zpool destroy datapool

# 3. Vérifier
zpool list
```

> **Vérifiez d'abord que ce pool ne contient plus rien d'utile** : `zfs list` et `pvesm status` disent ce qui y vit encore. Un disque de VM hébergé dessus serait perdu avec le pool. Déplacez-le d'abord avec **Move Storage**.

Cas B — le disque porte de vieilles signatures, d'une session précédente ou d'un autre usage :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
wipefs -a /dev/sdb   # effacer les signatures du disque
wipefs /dev/sdb   # vérifier : plus aucune signature
lsblk /dev/sdb   # le disque est libre
```

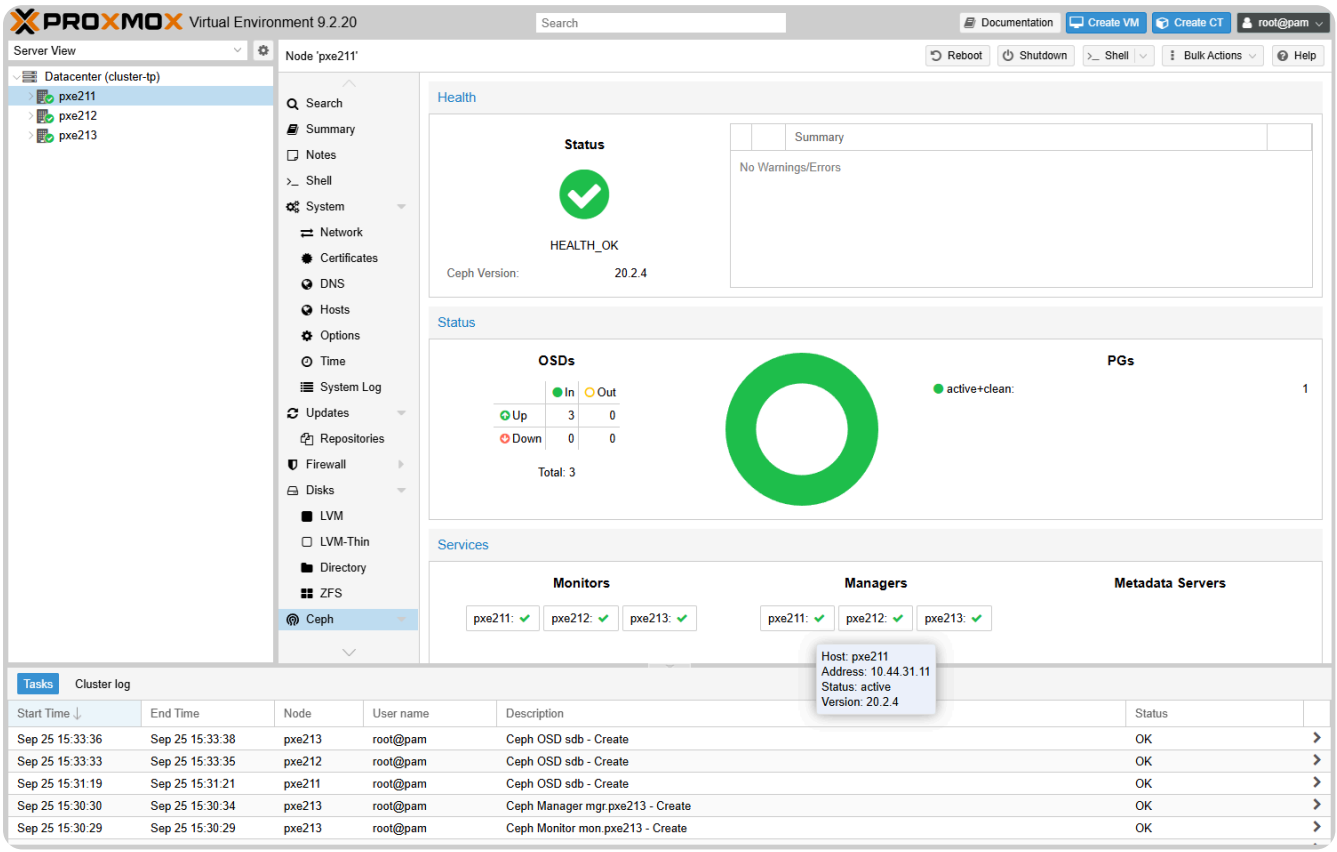
> **Relisez le nom du disque avant de valider `wipefs -a`.** La commande efface sans confirmation ni retour possible. Si `/dev/sda` apparaît dans votre ligne, arrêtez-vous : c'est le système Proxmox.

Reprenez ensuite l'étape 4.

Ce support est la propriété de Dawan France.

Étape 5 — Vérifier l'état de Ceph

Sur un nœud : **Ceph** (la page d'accueil du panneau).



HEALTH_OK, *No Warnings/Errors*, version 20.2.4. Trois OSD **Up** et **In**. Tous les PG `active+clean`.

Trois moniteurs et trois managers, cochés. L'infobulle d'un manager montre son adresse : le réseau public.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
ceph -s    # état du cluster Ceph
ceph osd tree    # les OSD, par nœud
```

Résultat attendu :

```
  cluster:
    id:     e21ecf15-c9ac-4c0f-8621-bf4281df1e24
    health: HEALTH_OK
```

Ce support est la propriété de Dawan France.

```
  services:
    mon: 3 daemons, quorum pxe211,pxe212,pxe213 (age 44s) [leader: pxe211]
    mgr: pxe211(active, since 67s), standbys: pxe212, pxe213
    osd: 3 osds: 3 up (since 17s), 3 in (since 22s)

  data:
    pools:   1 pools, 1 pgs
    objects: 2 objects, 577 KiB
    usage:   81 MiB used, 210 GiB / 210 GiB avail
    pgs:     1 active+clean

ID  CLASS  WEIGHT   TYPE NAME        STATUS  REWEIGHT  PRI-AFF
-1         0.20517  root default
-3         0.06839      host pxe211
 0    ssd  0.06839          osd.0        up   1.00000  1.00000
-5         0.06839      host pxe212
 1    ssd  0.06839          osd.1        up   1.00000  1.00000
-7         0.06839      host pxe213
 2    ssd  0.06839          osd.2        up   1.00000  1.00000
```

Le pool déjà présent (`1 pools`) s'appelle `.mgr` : le manager le crée pour ses propres données. `210 GiB` : l'espace **brut** des trois disques, avant réplication.

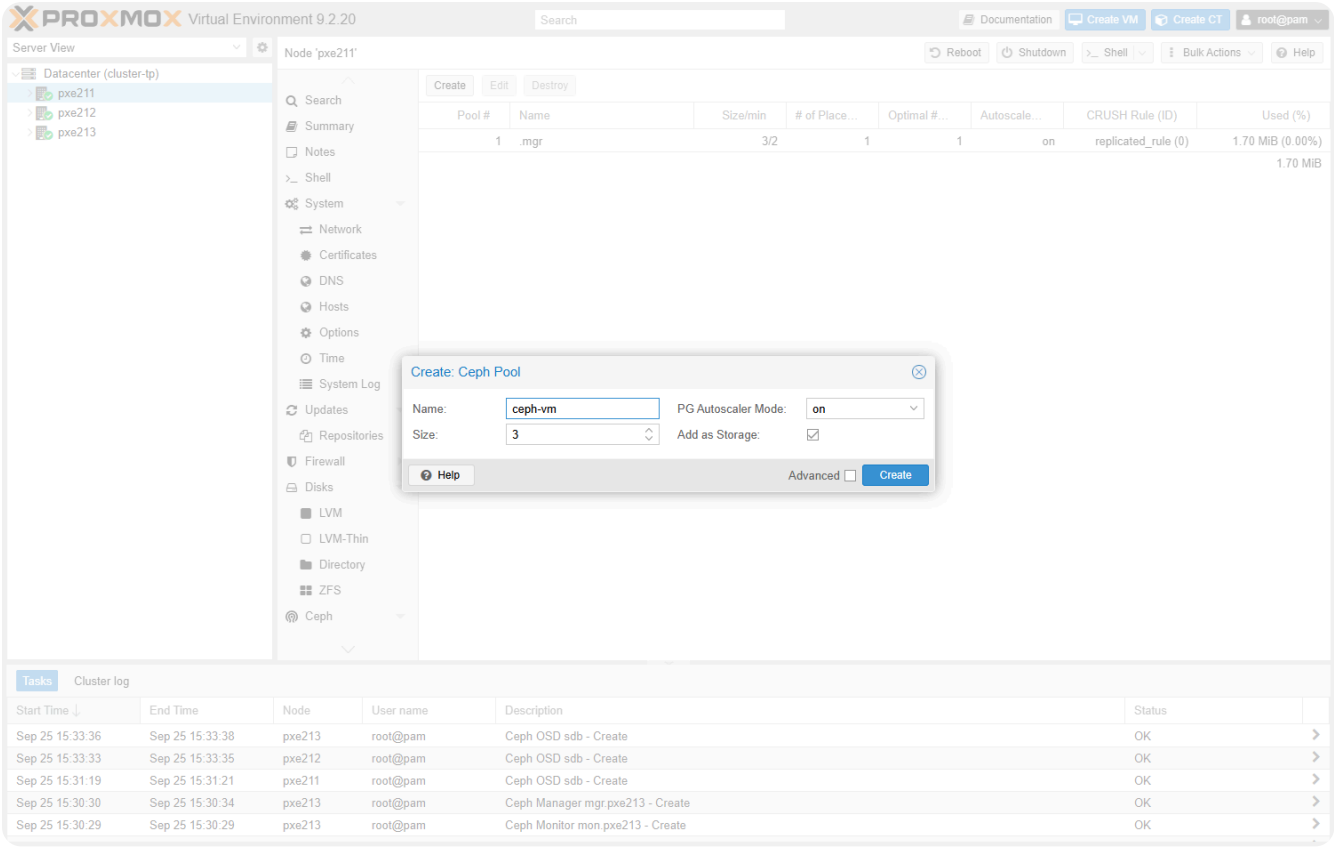
> **Un `HEALTH_WARN` passager est normal** pendant les premières minutes : moniteurs qui se synchronisent, OSD qui démarrent. `ceph health detail` en donne la raison exacte.
>
> S'il persiste au-delà de cinq minutes, lisez-la avant d'aller plus loin. C'est souvent une horloge décalée entre nœuds (`clock skew detected`), que corrige la synchronisation NTP.

Étape 6 — Créer un pool Ceph pour les VM

> **Les écrans Ceph sont sous le nœud, pas sous le Datacenter.** Sélectionnez un serveur, puis **Ceph** : vous y trouvez **Configuration**, **Monitor**, **OSD**, **CephFS**, **Pools** et **Log**. Un pool créé depuis un nœud existe pour les trois.

Sur `pxe2<N>1` : **Ceph** → **Pools** → **Create**.

Ce support est la propriété de Dawan France.



Name `ceph-vm`. **Size 3** : trois copies. **PG Autoscaler Mode on** : Ceph ajuste seul le découpage du pool.

Add as Storage coché : le pool devient aussitôt un stockage Proxmox. **Min.

Size **n'apparaît qu'en cochant** Advanced **; il vaut 2. Derrière, le pool `.mgr` existe déjà.** → Create**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI : pool répliqué, déclaré aussitôt comme stockage
pveceph pool create ceph-vm --pg_autoscale_mode on --add_storages 1
pvesm status | grep ceph-vm   # le stockage ceph-vm est-il actif ?
ceph osd pool ls detail | grep ceph-vm   # taille, taille minimale et règle du pool
```

Résultat attendu :

```
ceph-vm            rbd     active        69698728               0        69698728
0.00%
```

Ce support est la propriété de Dawan France.

```
pool 2 'ceph-vm' replicated size 3 min_size 2 crush_rule 0 object_hash rjenkins
pg_num 128 pgp_num 128 autoscale_mode on ...
```

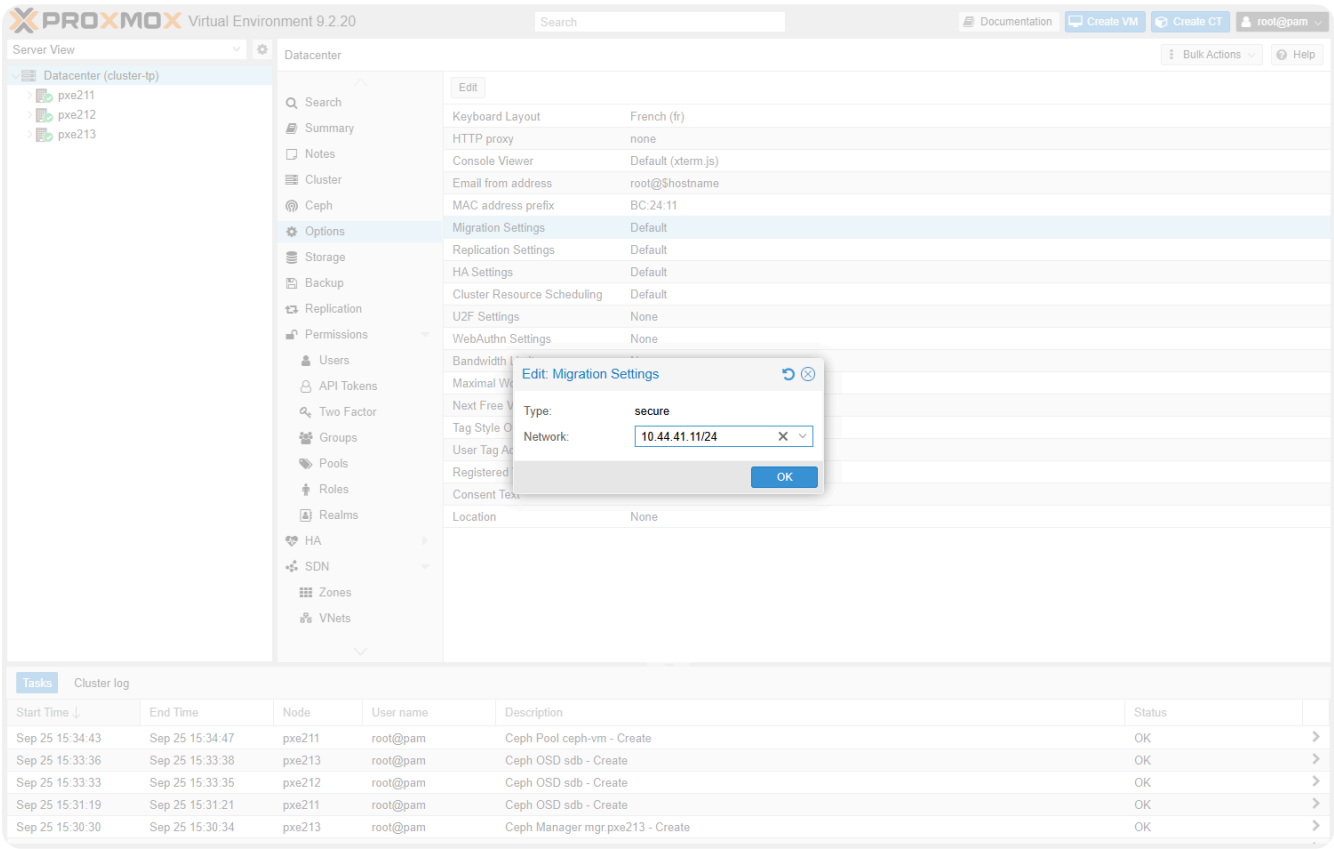
Environ 66 Gio utiles pour 210 Gio bruts : un tiers, puisque chaque donnée existe en trois exemplaires. Le pool part avec `pg_num 128`.

Relisez la même ligne quelques minutes plus tard : sur le lab, elle portait `pg_num 100 pgp_num 100 pg_num_target 32`. L'autoscaler a jugé 128 trop pour un pool presque vide, et réduit pas à pas vers 32 (`ceph osd pool autoscale-status`, colonne `PG_NUM`).

Étape 7 — Désigner le réseau de migration

Par défaut, une migration passe par le réseau d'administration : c'est ce qu'a montré l'atelier 2. On lui donne `ens22`.

Datacenter → **Options** → double-clic sur **Migration Settings** → **Network** : choisir l'adresse de `ens22` → **OK**.



Type : secure — la mémoire voyage chiffrée dans un tunnel SSH. **Network** : la liste propose les adresses du nœud ; on prend celle de `ens22`. Le réglage vaut pour tout le cluster : chaque

Ce support est la propriété de Dawan France.

nœud utilisera son adresse dans ce réseau.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le réglage, tel qu'enregistré pour tout le cluster
grep migration /etc/pve/datacenter.cfg
```

Résultat attendu : `migration: network=10.59.41.11/24,type=secure` (lab `10.59` : chez vous, `10.<site>.<N+40>.11/24`).

Étape 8 — Migration à chaud via Ceph

1. Une VM sur Ceph. Clonez le template 100 du chapitre 2 directement sur le pool.

Template 100 → clic droit → **Clone** : VM ID `300`, Name `vm-ha-test`, Mode **Full Clone**, Target Storage `ceph-vm` → **Clone**. Puis VM 300 → **Cloud-Init** : IP `10.<site>.<N>.85/24`, Gateway `10.<site>.<N>.1` → **Regenerate Image** → **Start**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Clone complet du template, son disque posé sur le pool Ceph
qm clone 100 300 --name vm-ha-test --full --storage ceph-vm
qm set 300 --ipconfig0 ip=10.<site>.<N>.85/24,gw=10.<site>.<N>.1   # son adresse fixe
qm start 300   # la démarrer
```

Résultat attendu : `transferred 20.0 GiB of 20.0 GiB (100.00%)` en une dizaine de secondes (9,6 s sur le lab), puis `scsi0: ceph-vm:vm-300-disk-0` dans `qm config 300`.

2. L'agent invité.

Le template vient de l'image cloud Debian, qui **n'a pas l'agent QEMU** : `qm agent 300 ping` répond `QEMU guest agent is not running`. Installez-le comme au chapitre 2, depuis le Shell du nœud :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Dans la VM, par SSH avec l'utilisateur cloud-init du template
ssh devops@10.<site>.<N>.85                    # puis, dans la VM :
```

Ce support est la propriété de Dawan France.

```
sudo apt-get update                              # la liste des paquets
sudo apt-get install -y qemu-guest-agent         # l'agent invité
sudo systemctl start qemu-guest-agent            # le lancer
exit                                             # retour au nœud
qm agent 300 ping                                # l'agent répond-il ?
echo "code : $?"                                 # 0 : oui
```

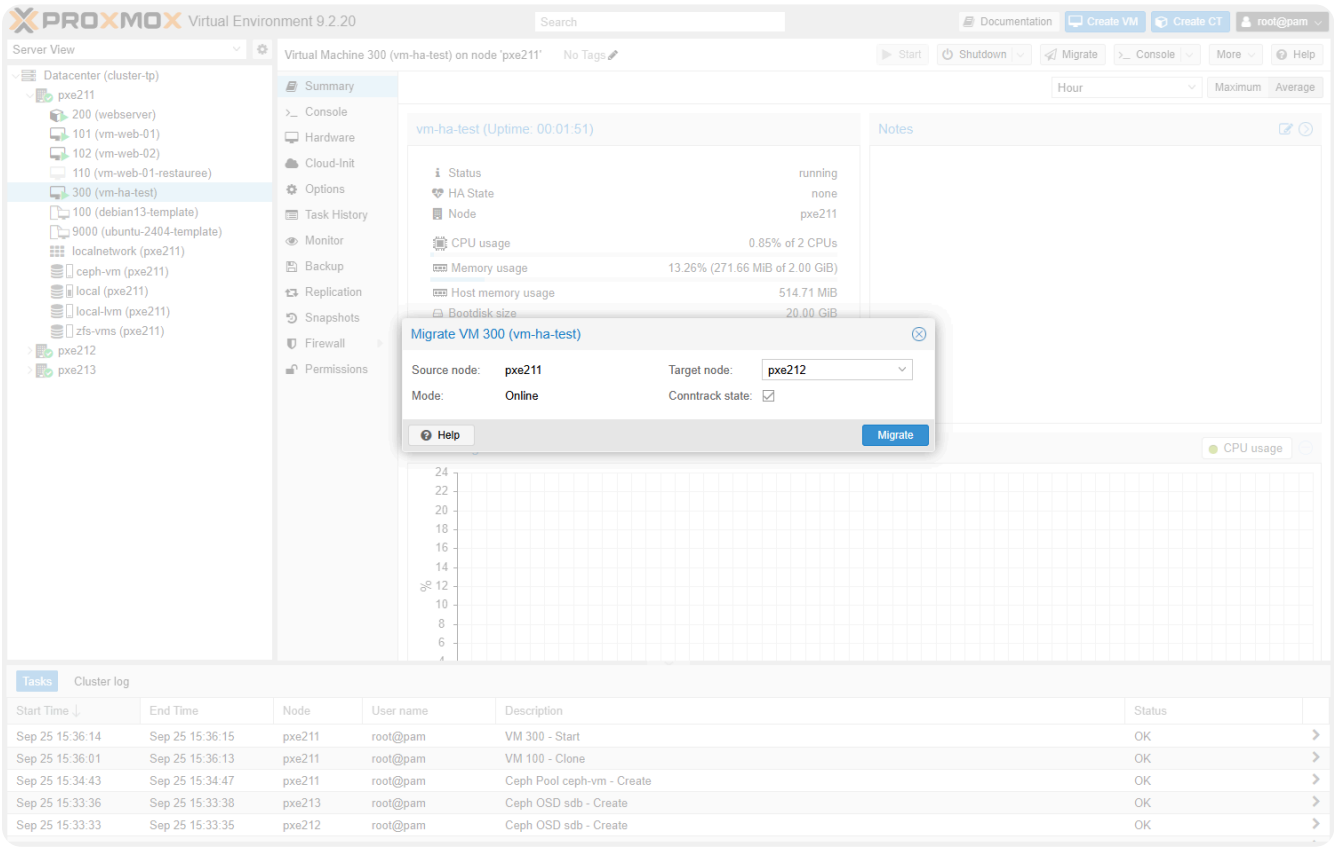
Résultat attendu : `code : 0`. L'atelier 6 en a besoin.

3. La migration. Depuis un **autre** nœud, lancez un ping continu vers la VM : il mesurera la coupure.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Sur pxe2<N>3 : 300 pings, un toutes les 0,2 s
ping -i 0.2 -c 300 10.<site>.<N>.85
```

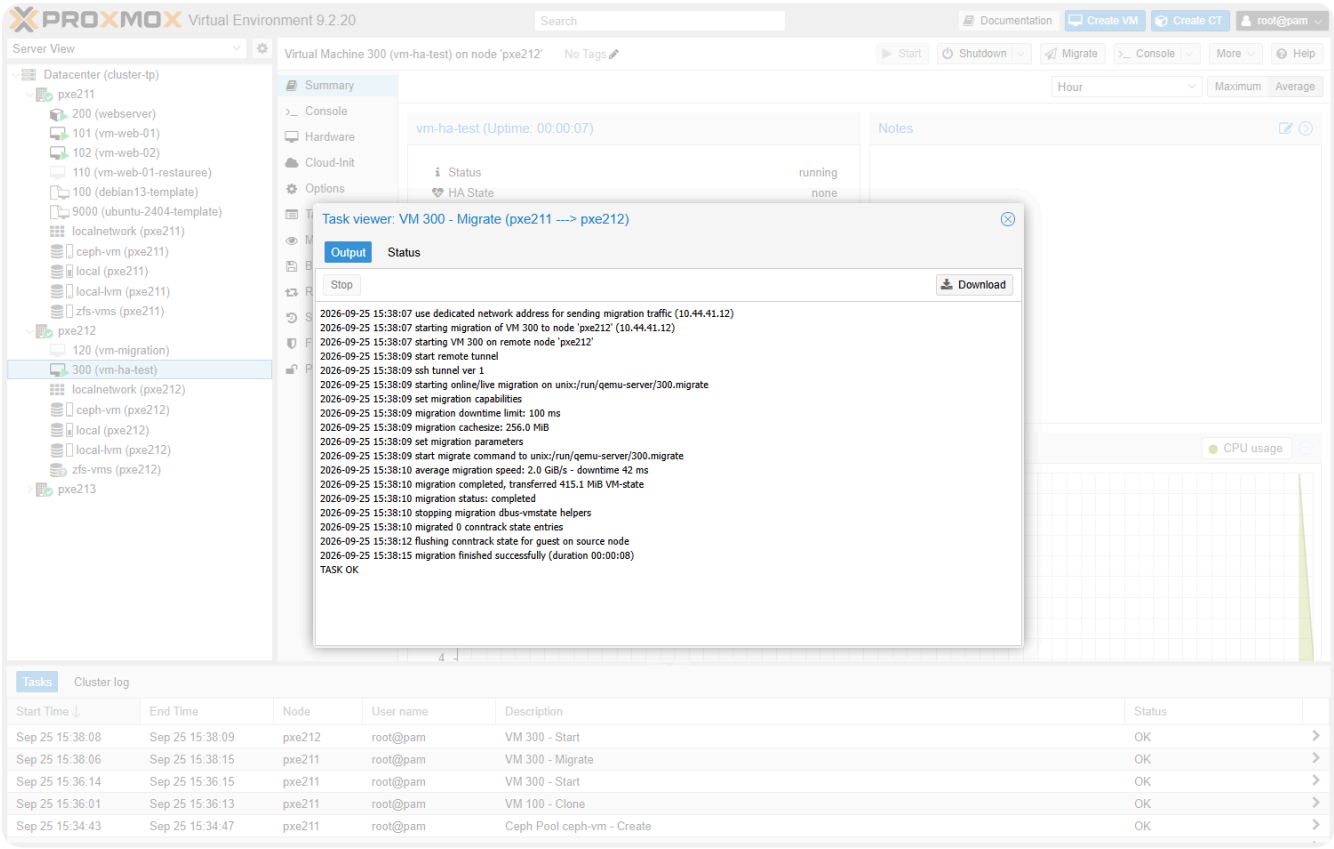
Puis VM 300 → **Migrate**.



Mode : Online : la VM tourne, la migration sera à chaud. Il n'y a rien à cocher pour cela. Aucun avertissement de disque local : le disque est sur Ceph.

Ce support est la propriété de Dawan France.

Conntrack state transporte l'état des connexions suivies par le pare-feu. **Target node** `pxe2<N>2` → **Migrate**.



La première ligne confirme le réseau : *use dedicated network address for sending migration traffic*. Puis la VM démarre sur la cible, la mémoire est copiée, et la bascule ne dure que quelques dizaines de millisecondes (42 ms sur cette capture).

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI
qm migrate 300 pxe2<N>2 --online
```

Résultat attendu (réseau de migration `10.59.41.x`) :

```
use dedicated network address for sending migration traffic (10.59.41.12)
starting migration of VM 300 to node 'pxe212' (10.59.41.12)
starting VM 300 on remote node 'pxe212'
migration downtime limit: 100 ms
average migration speed: 2.0 GiB/s - downtime 16 ms
```

Ce support est la propriété de Dawan France.

```
migration completed, transferred 552.6 MiB VM-state
migration finished successfully (duration 00:00:06)
```

Comparez avec l'atelier 2 : **aucun disque copié**, seulement 553 Mio de mémoire, par le réseau de migration. Côté ping :

```
300 packets transmitted, 299 received, 0.333333% packet loss, time 61926ms
```

Un paquet perdu sur 300 : 0,2 seconde de silence, au pire.

> **L'uptime affiché par Proxmox repart de zéro, et ce n'est pas un redémarrage.** Il compte la vie du processus QEMU. Or la VM tourne désormais dans un nouveau processus, sur la cible.
>
> Sur le lab : `uptime: 42` avant, `uptime: 8` après. Pour prouver que le système n'a pas redémarré, demandez-le **à la VM** : `ssh devops@10.<site>.<N>.85 uptime -s`. Sur le lab, elle répondait `2026-10-07 12:06:19` avant, et `2026-10-07 12:06:20` après (heure UTC) : la même date de démarrage, à l'arrondi près.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Ceph sain, stockage actif, VM 300 en marche sur pxe2<N>2
ceph health                          # la santé de Ceph
pvesm status | grep ceph-vm   # le stockage ceph-vm est actif
pvesh get /nodes/pxe2<N>2/qemu/300/status/current --output-format json-pretty |
grep '"status"'   # la VM 300 tourne-t-elle sur le nœud 2 ?
ssh devops@10.<site>.<N>.85 uptime -s   # depuis quand la VM tourne
```

Résultat attendu : `HEALTH_OK` ; `ceph-vm rbd active` ; `"status" : "running"` ; une date de démarrage antérieure à la migration.

Atelier 6 — Haute disponibilité

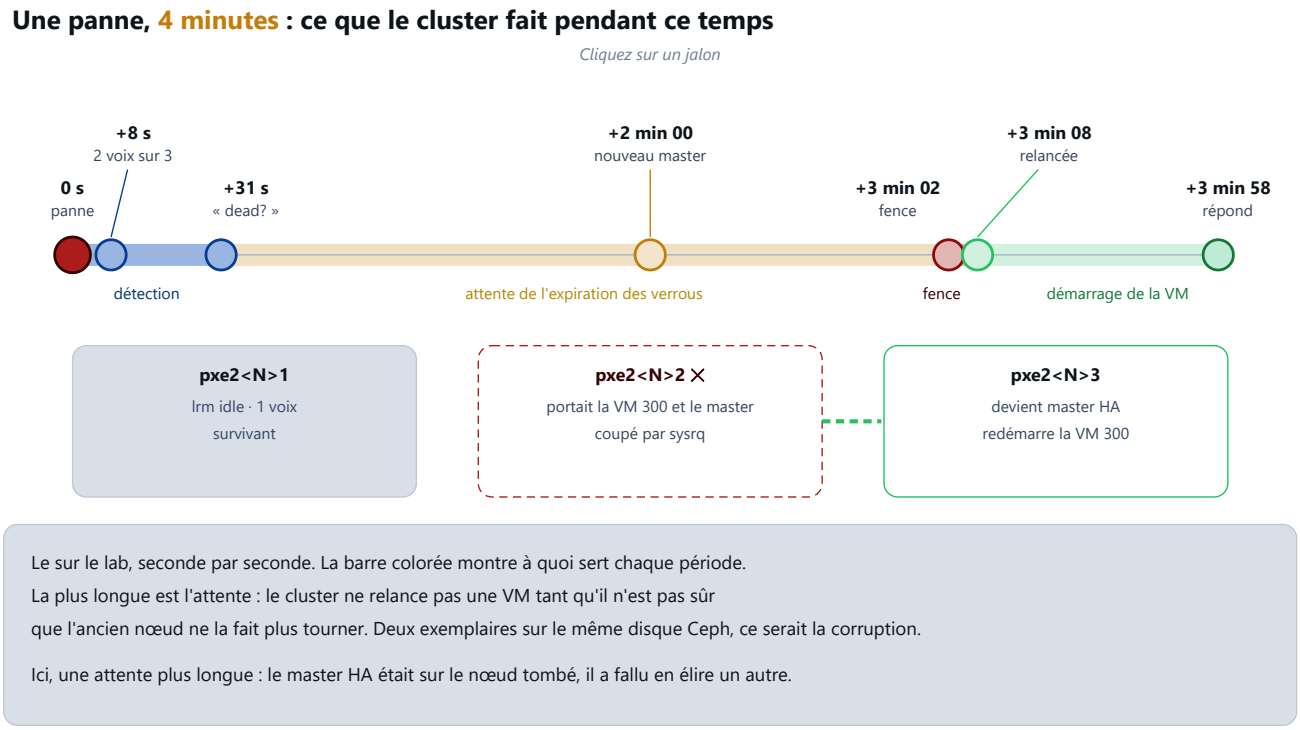
Ce support est la propriété de Dawan France.

Contexte

La migration à chaud protège d'une maintenance **planifiée**. La **haute disponibilité** (HA) protège d'une panne **subie**.

Une VM déclarée ressource HA redémarre seule sur un autre nœud si le sien tombe. C'est possible parce que son disque est sur Ceph, donc accessible depuis les autres nœuds.

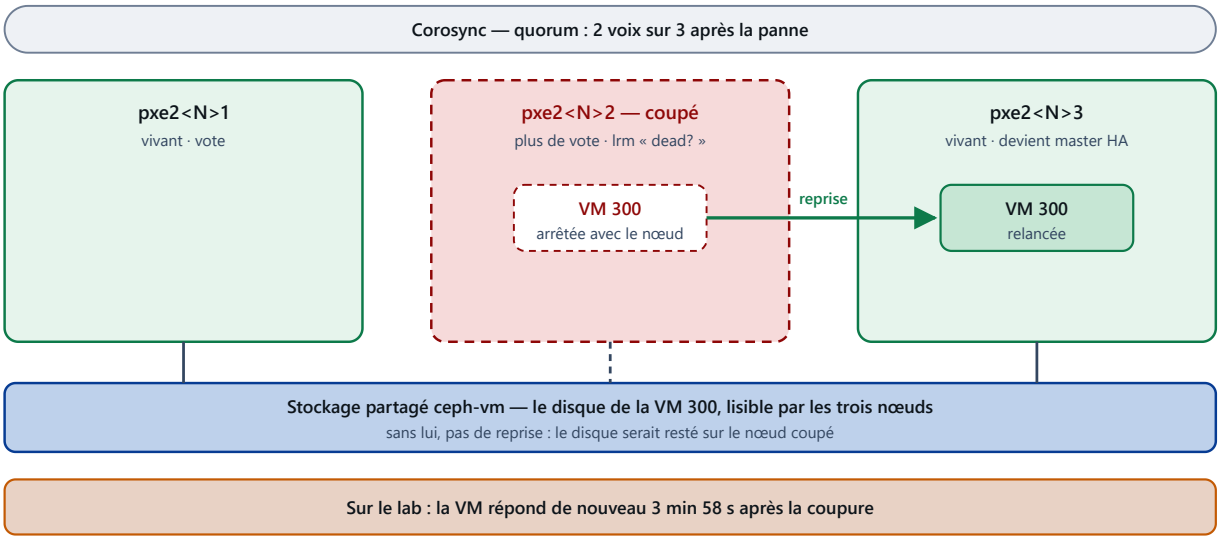
Résultat à obtenir : la VM 300 déclarée ressource HA, puis son redémarrage automatique sur un autre nœud après une panne brutale du sien. Ensuite, une mise en maintenance chronométrée du nœud 1, puis le retour.



La panne de l'étape 2, telle qu'elle s'est déroulée sur le lab. La barre montre à quoi sert chaque période ; la plus longue est une attente volontaire. Cliquez sur un jalon pour lire la ligne de journal correspondante.

Ce support est la propriété de Dawan France.

Haute disponibilité : un nœud tombe, la VM repart ailleurs



Survolez un élément pour lire le résultat correspondant.

Le scénario de l'atelier : le nœud qui porte la VM 300 est coupé. Les deux autres gardent le quorum (2 voix sur 3) et la VM est relancée sur l'un d'eux.

C'est possible parce que son disque est dans Ceph, lisible depuis les trois nœuds. Survolez un élément pour lire le résultat du lab.

Arborescence attendue

```
VM 300 « vm-ha-test » → ressource HA (état souhaité : started)
Panne brutale du nœud qui l'héberge
  ├── Corosync : 2 voix sur 3, le quorum tient
  ├── le nœud est déclaré mort, puis isolé (fence)
  └── la VM redémarre sur un nœud survivant   ← environ 4 min sur le lab
```

Ce que vous devez faire

1. **Déclarer la VM 300 en ressource HA**, état souhaité `started`.
2. **Simuler une panne brutale** du nœud qui l'héberge, et **chronométrer** : quand le nœud est-il déclaré mort, quand la VM redémarre-t-elle ?
3. **Comprendre le délai** : pourquoi le cluster attend-il avant de relancer la VM ?
4. **Mettre le nœud 1 en maintenance**, chronométrer la bascule et la coupure du ping, puis le **sortir de maintenance** et observer le retour.

Ce support est la propriété de Dawan France.

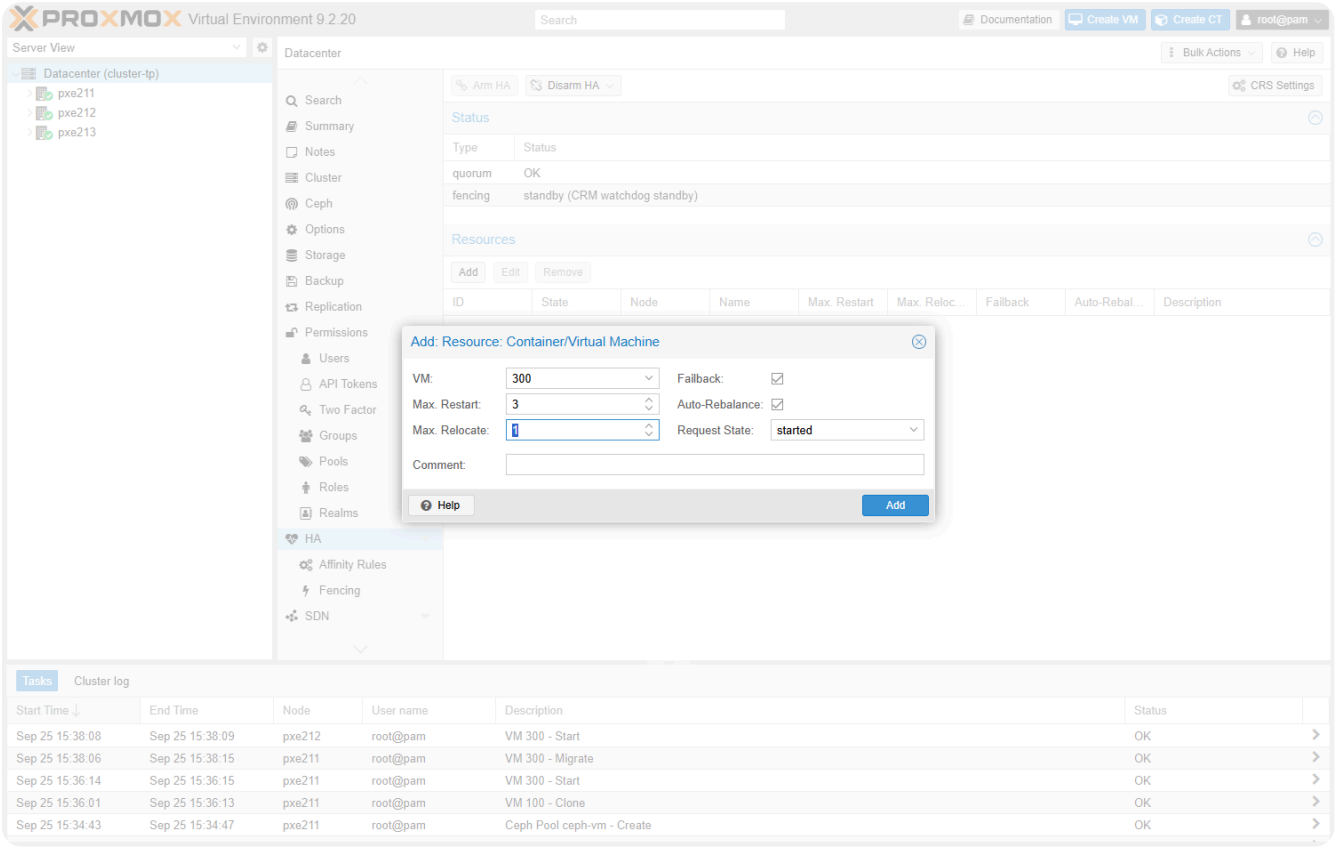
💡 La HA n'est pas une migration sans coupure : la VM **redémarre** ailleurs. Il y a une interruption de quelques minutes. C'est le prix d'une panne que personne n'a prévue.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Déclarer la VM 300 en ressource HA

Datacenter → **HA** → section **Resources** → **Add**.



VM `300` : la liste propose toutes les VM et tous les conteneurs du cluster, avec leur nœud.
Max. Restart `3` : trois redémarrages **sur place** avant d'abandonner (1 par défaut).

Max. Relocate `1` : un déplacement vers un autre nœud. **Request State** `started`.

Failback et **Auto-Rebalance** sont cochés d'office ; ils n'agissent qu'avec des règles d'affinité.
En haut, **Arm HA / Disarm HA** permet de suspendre toute la HA pendant une maintenance.
→ **Add**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI
ha-manager add vm:300 --state started --max_restart 3 --max_relocate 1
ha-manager status   # état de la HA
```

Ce support est la propriété de Dawan France.

Résultat attendu :

```
quorum OK
master pxe212 (active, Wed Oct  7 14:13:21 2026)
fencing armed (CRM watchdog active)
lrm pxe211 (idle, watchdog standby, Wed Oct  7 14:13:24 2026)
lrm pxe212 (active, watchdog active, Wed Oct  7 14:13:21 2026)
lrm pxe213 (idle, watchdog standby, Wed Oct  7 14:13:24 2026)
service vm:300 (pxe212, started)
```

Lisez les lignes. Le **master** décide pour tout le cluster. Chaque nœud a un **lrm** qui exécute ses décisions.

Le lrm du nœud qui porte une ressource passe `active`, et son **chien de garde** (*watchdog*) s'arme : si ce nœud perd le cluster, le chien de garde le redémarre. Sur le lab, le master était lui aussi sur `pxe212`.

Étape 2 — Panne brutale du nœud de la VM

> **Moment clé de l'atelier.** Lisez l'étape en entier avant d'agir. Le nœud que vous allez couper **ne se rallumera pas seul** : c'est le formateur qui le relancera, depuis l'hôte qui l'héberge.

Repérez le nœud de la VM 300 : celui de la ligne `service vm:300 (…, started)`. Sur le lab, `pxe212`, où l'atelier 5 l'avait migrée.

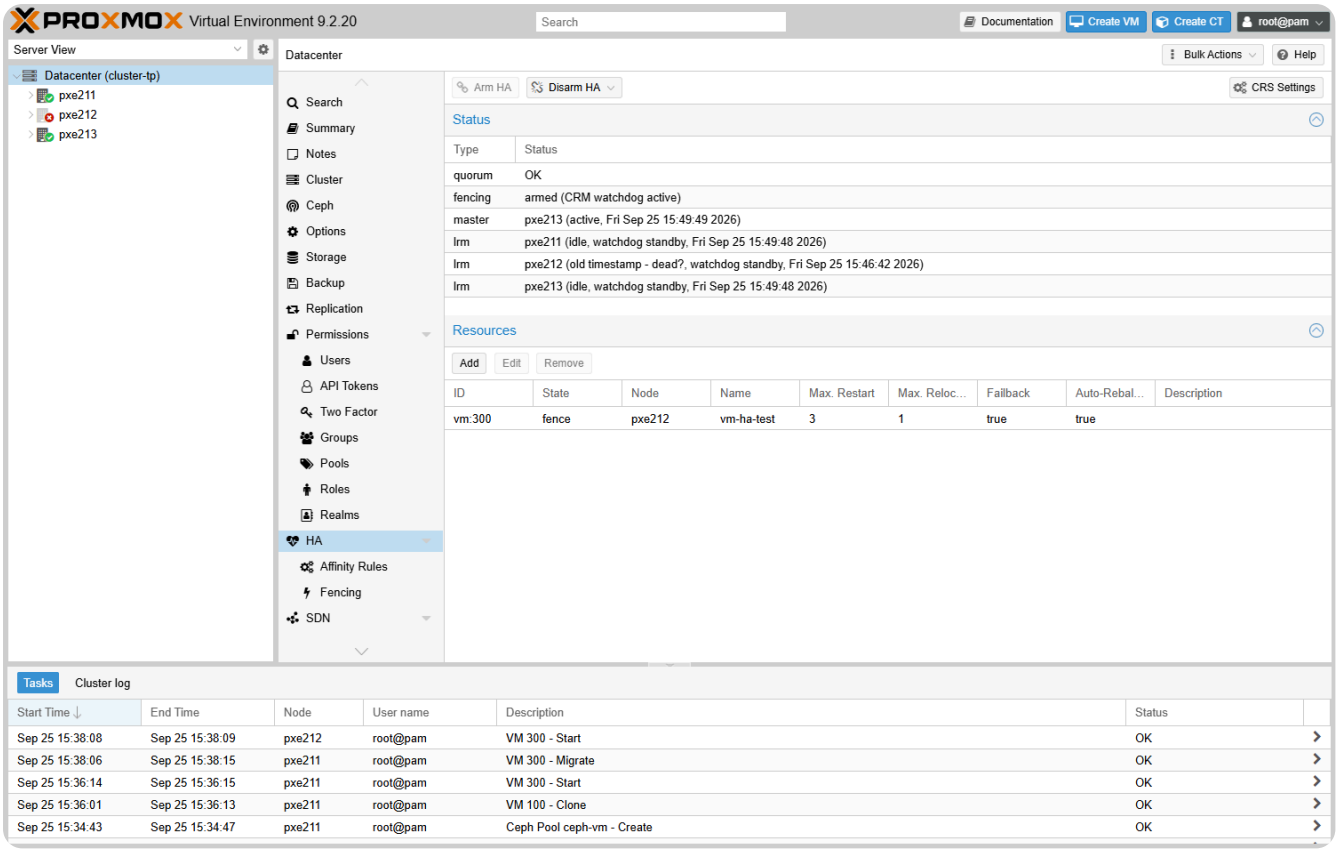
Gardez **Datacenter** → **HA** ouvert sur **un autre nœud**. Puis, dans le Shell du nœud de la VM :

Dans le Shell du nœud qui porte la VM 300 :

```
# Coupure immédiate, sans arrêt propre : l'équivalent d'une panne de courant
echo o > /proc/sysrq-trigger
```

> **Pourquoi pas `poweroff` ?** Un arrêt propre est une maintenance : le gestionnaire HA en est prévenu, et il applique sa politique d'arrêt (`shutdown_policy`).
>
> On ne verrait pas la reprise sur panne. La coupure par `sysrq` ne prévient personne.

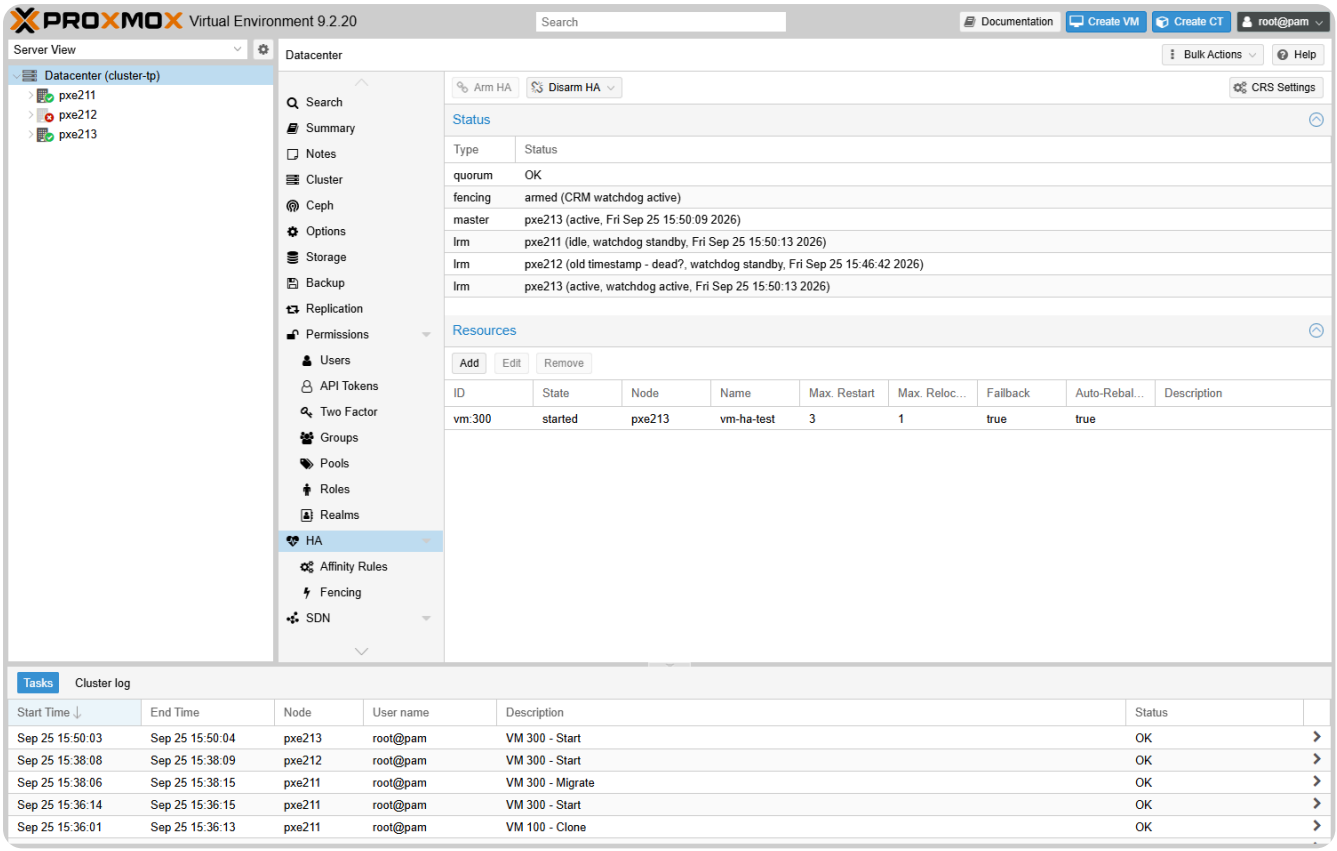
Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.



(3 minutes après la panne). `pxe212` porte une croix rouge dans l'arbre. Son lrm affiche `old timestamp - dead?`.

Le **master est passé sur** `pxe213`. La ressource `vm:300` est en **fence** : le cluster s'assure que `pxe212` ne fait plus tourner la VM.

Ce support est la propriété de Dawan France.



`vm:300` **started** sur `pxe213`. En bas, la tâche **VM 300 - Start** sur `pxe213`, lancée par la HA, sans personne.

Chronologie sur le lab (panne à 14:13:35, `ha-manager status` toutes les 5 secondes depuis `pxe213`, ping toutes les secondes) :

Heure	Délai	Ce qui se passe
14:13:43	+8 s	`Total votes: 2` — le quorum tient (2 voix sur 3)
14:14:06	+31 s	`lrm pxe212 (old timestamp - dead?…)`, et le master aussi : il était sur `pxe212`
14:15:35	+2 min 00	`pxe213` devient master
14:16:37	+3 min 02	`service vm:300 (pxe212, fence)`
14:16:43	+3 min 08	`fencing: acknowledged` → `recovery` → `service vm:300 (pxe213, starting)`
14:17:05	+3 min 30	`service vm:300 (pxe213, started)`

Ce support est la propriété de Dawan France.

Heure	Délai	Ce qui se passe
14:17:33	+3 min 58	la VM répond de nouveau au ping

Le journal du nouveau master détaille la fin de la séquence :

Dans le Shell du nœud devenu maître HA (celui qu'indique `ha-manager status`) :

```
# Sur le nouveau master
journalctl -u pve-ha-crm --since "-10min" --no-pager -n 12   # les 12 dernières lignes
```

Résultat attendu (`pxe213`, extrait) :

```
successfully acquired lock 'ha_manager_lock'
status change slave => master
node 'pxe212': state changed from 'online' => 'unknown'
service 'vm:300': state changed from 'started' to 'fence'
node 'pxe212': state changed from 'unknown' => 'fence'
successfully acquired lock 'ha_agent_pxe212_lock'
fencing: acknowledged - got agent lock for node 'pxe212'
node 'pxe212': state changed from 'fence' => 'unknown'
service 'vm:300': state changed from 'fence' to 'recovery'
recover service 'vm:300' from fenced node 'pxe212' to node 'pxe213'
service 'vm:300': state changed from 'recovery' to 'started'  (node = pxe213)
```

Pourquoi attendre ? Le nœud perdu est peut-être seulement coupé du réseau, VM toujours en marche.

Relancer la VM ailleurs tout de suite, ce serait deux exemplaires qui écrivent sur le même disque Ceph. Le cluster attend donc que le verrou du nœud perdu expire : à ce moment-là, son chien de garde l'a forcément redémarré.

`got agent lock` signifie exactement cela. Ici, l'attente a été plus longue d'une minute, parce que le master était sur le nœud mort : il a d'abord fallu en élire un nouveau.

La VM, elle, a **redémarré** : `ssh devops@10.<site>.<N>.85 uptime -s` répondait `2026-10-07 12:17:04` (UTC, soit 14:17:04 à Paris), l'heure de la reprise. Comparez avec l'atelier 5, où la migration à chaud gardait la VM en vie.

Dawan Nous donnons vos talents
Ce support est la propriété de Dawan France.

Le retour du nœud. Le formateur rallume la VM qui porte `pxe212`.

Sur le lab, le cluster avait retrouvé ses 3 voix moins d'une minute après, et Ceph son `HEALTH_OK` dans la foulée (3 OSD `up`). La VM 300 **reste sur `pxe213`** : sans règle d'affinité, Failback n'a pas de nœud préféré où la ramener.

Étape 3 — Mettre le nœud 1 en maintenance, et chronométrer

On vide le nœud 1 pour une intervention. Avant : un ping continu vers la VM 300, **depuis le Shell du nœud 2**, pour mesurer la coupure.

Dans le Shell du nœud 2 :

```
ping -D -i 0.5 10.<site>.<N>.85   # ping horodaté toutes les 0,5 s
```

Dans le Shell du nœud 1 :

```
date +%T                                                    # l'heure
ha-manager crm-command node-maintenance enable pxe2<N>1     # puis le nœud 1 en maintenance
watch -n 2 "ha-manager status | grep -E 'lrm|service'"   # suivre les LRM et les services toutes les 2 s
```

Quand la VM 300 est `started` ailleurs, quittez `watch` (`Ctrl+C`) et relisez le journal.

Dans le Shell du nœud 1 :

```
journalctl -u pve-ha-crm -u pve-ha-lrm --since "-5min"   # le journal HA des 5 dernières minutes ; à lire : maintenance, vm:300, watchdog
```

Résultat attendu (les lignes qui comptent) :

```
14:24:20  (commande envoyée)
14:24:28 pve-ha-lrm: status change active => maintenance
14:24:33 pve-ha-crm: node 'pxe211': state changed from 'online' => 'maintenance'
14:24:33 pve-ha-crm: migrate service 'vm:300' to node 'pxe213' (running)
14:24:43 pve-ha-crm: service 'vm:300': state changed from 'migrate' to 'started'
```

Dawan
Ce support est la propriété de Dawan France.

```
(node = pxe213)
14:24:44 pve-ha-lrm: watchdog closed (disabled)
```

Mesure	Sur le lab
Commande → VM 300 démarrée ailleurs	23 s
Commande → `watchdog closed (disabled)` sur le nœud 1	24 s
Coupure du ping	une réponse manquée, environ 1 s

Deux constats. La maintenance **migre à chaud** (`running`) : la VM ne redémarre pas, contrairement à la panne de l'étape 2.

`watchdog closed (disabled)` signale que le nœud 1 n'a plus rien à protéger : on peut l'arrêter.

Étape 4 — Sortir de maintenance : le retour

Dans le Shell du nœud 1 :

```
date +%T                                                    # l'heure
ha-manager crm-command node-maintenance disable pxe2<N>1    # puis sortie de
maintenance
watch -n 2 "ha-manager status | grep service"   # suivre les services toutes les
2 s
```

Résultat attendu :

```
14:25:14  (commande envoyée)
14:25:24 pve-ha-lrm: status change maintenance => active
14:25:33 pve-ha-crm: node 'pxe211': state changed from 'maintenance' => 'online'
14:25:33 pve-ha-crm: moving service 'vm:300' back to 'pxe211', node came back
from maintenance.
14:25:53 pve-ha-crm: service 'vm:300': state changed from 'migrate' to 'started'
(node = pxe211)
```

Ce support est la propriété de Dawan France.

La VM 300 est **revenue** sur le nœud 1, en 39 s, de nouveau à chaud : une réponse au ping manquée, comme à l'aller. C'est le comportement annoncé par la documentation. Le ping lancé sur le nœud 2 s'arrête par `Ctrl+C` : sur le lab, `240 packets transmitted, 238 received` sur l'ensemble, les deux réponses manquées étant les deux bascules.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Où tourne la VM 300, et le cluster a-t-il gardé son quorum ?
ha-manager status | grep vm:300
pvecm status | grep Quorate   # le cluster a-t-il gardé son quorum ?
```

Résultat attendu : `service vm:300 (pxe2<N>1, started)`, puis `Quorate: Yes`.

Dépannage — Erreurs fréquentes

`Create Cluster` **reste grisé, champ du nom en rouge.** Nom de cluster de plus de 15 caractères. Remède : Un nom court : `cluster-tp` (atelier 1).

Le cluster dialogue par `vmbr0`**.** Link 0 laissé à sa valeur proposée. Remède : Choisir l'adresse de `vmbr1` dans la liste du Link 0 (atelier 1).

`pvecm add` **échoue** `EOF while reading password`**.** Commande interactive lancée sans terminal. Remède : `echo 'motdepasse' | pvecm add … --fingerprint "$FP"`.

Un nœud affiche `Nodes: 1`**,** `Config Version: 1`**,** `Expected votes: 1`**, et** `pvecm add` **répond** `authentication key '/etc/corosync/authkey' already exists`**. Create Cluster** cliqué sur ce nœud au lieu de **Join Cluster** : il a fondé son propre cluster. Remède : Le séparer sans le réinstaller, puis le joindre de nouveau (cours, §2.5).

`apt failed during ceph installation (256)`**, précédé de** `Abort.`**.** Apt n'a pas eu de réponse à `[Y/n]`. Remède : `yes | pveceph install --version tentacle --repository no-subscription`.

`apt failed during ceph installation`**, précédé d'une erreur 401.** Dépôt Ceph enterprise sans abonnement. Remède : `pveceph install --version tentacle --repository no-`

Dawan Ce support est la propriété de Dawan France.

`subscription`.

Migration refusée « Cannot migrate VM with local CD/DVD ». ISO locale restée dans le lecteur. Remède : Détacher l'ISO : `qm set <vmid> --ide2 none,media=cdrom`.

La VM HA ne bascule pas après un `poweroff`. Arrêt propre traité comme une maintenance. Remède : Simuler la panne par `echo o > /proc/sysrq-trigger`.

`HEALTH_WARN … POOL_TOO_MANY_PGS`. Pool créé en ligne de commande, autoscaler en `warn`. Remède : `pveceph pool set <pool> --pg_autoscale_mode on`.

`HEALTH_WARN` avec le seul message `noout flag(s) set`. Le drapeau `noout`, posé pour une maintenance, est resté. Remède : `ceph osd unset noout` : `HEALTH_OK` une vingtaine de secondes plus tard.

`systemctl start ceph-osd@0` échoue ; le journal dit `OSD data directory /var/lib/ceph/osd/ceph-0 does not exist; bailing out`. Cet OSD n'est pas sur ce nœud : les numéros suivent l'ordre de création des OSD, pas celui des nœuds. Le `stop` précédent n'a donc rien arrêté.

Remède : `ceph osd tree` : prendre le numéro affiché sous `host <ce nœud>`. Effacer l'échec avec `systemctl reset-failed ceph-osd@0`.

`Start request repeated too quickly` sur un service. Systemd bloque un service après plusieurs échecs rapprochés. Remède : Corriger la cause, puis `systemctl reset-failed <service>` avant de relancer.

Récapitulatif des commandes essentielles

Situation	Commande
Créer / rejoindre un cluster	`pvecm create <nom> --link0 <ip-vmbr1> --link1 <ip-vmbr0>` / `pvecm add <ip-fondateur> --link0 address=<ip-vmbr1-local>`
État du cluster / des liens	`pvecm status` / `corosync-cfgtool -s`
Sortir un nœud du cluster	sur le nœud : `systemctl stop pve-cluster corosync`, `pmxcfs -l`, effacer `/etc/pve/corosync.conf` et `/etc/corosync/*`, `killall pmxcfs`, `systemctl start pve-cluster` ; sur un autre nœud : `pvecm delnode <nœud>` (cours, §2.5)

Ce support est la propriété de Dawan France.

Situation	Commande
Migrer une VM (froid / chaud)	`qm migrate <vmid> <nœud>` / `qm migrate <vmid> <nœud> --online`
Installer / initialiser Ceph	`yes | pveceph install --version tentacle --repository no-subscription` / `pveceph init --network … --cluster-network …`
Créer MON / MGR / OSD	`pveceph mon create` / `pveceph mgr create` / `pveceph osd create /dev/sdX`
État Ceph	`ceph -s` / `ceph osd tree` / `ceph df`
Ajouter une ressource HA	`ha-manager add vm:<vmid> --state started`

Ce support est la propriété de Dawan France.

Chapitre 5 — SDN, pare-feu et sauvegarde

Le chapitre 4 a rendu votre infrastructure résistante aux pannes de matériel : un nœud peut tomber, Ceph garde les données et la haute disponibilité relance les machines. Il lui manque trois choses. Un **réseau commun** aux trois nœuds, pour qu'une machine garde son réseau quel que soit le nœud qui la porte.

Une **protection** : pour l'instant, tout peut parler à tout. Et un moyen de **revenir en arrière** après une erreur — une suppression malheureuse, un rançongiciel —, que Ceph réplique fidèlement sur trois disques.

Le fil conducteur suit cet ordre. D'abord une présentation du **SDN** (*Software-Defined Networking*) : déclarer un réseau une seule fois pour les trois nœuds.

Puis le **pare-feu** intégré, en mode cluster : il décide qui a le droit de parler à qui. Vous connectez enfin le cluster à **Proxmox Backup Server**, y sauvegardez vos machines, les restaurez, en entier puis fichier par fichier, et répliquez les sauvegardes vers un second site.

Objectifs du chapitre

À l'issue de ce chapitre, les stagiaires seront capables de :

- **Expliquer** le SDN Proxmox : zones, VNets et subnets, déclarés une fois pour tout le cluster
- **Configurer** le pare-feu Proxmox dans ses zones (Datacenter, nœud, invité) sans perdre l'accès
- **Expliquer** l'architecture de PBS (déduplication par chunks), le connecter à Proxmox VE, sauvegarder et restaurer une VM ou un conteneur
- **Répliquer** les sauvegardes vers un second PBS, un namespace par source

> **Infrastructure disponible à ce stade :** le cluster de 3 nœuds avec Ceph (chapitre 4). Chaque stagiaire dispose aussi de son Proxmox Backup Server, en `10.<site>.<N>.14`.

Ce support est la propriété de Dawan France.

1. SDN : pourquoi, et comment il s'organise

Au chapitre 3, le réseau se construisait pont par pont : un `vmbr` sur chaque nœud, écrit à la main dans `/etc/network/interfaces`. Cela suffit sur un serveur seul. Mais maintenant que les trois nœuds forment un cluster, une nouvelle contrainte apparaît.

Un réseau virtuel utile doit exister **à l'identique sur les trois nœuds**. Sinon, une VM qui migre perd sa connectivité en changeant d'hôte.

C'est exactement ce que résout le **SDN** (*Software-Defined Networking*) de Proxmox VE. On déclare zones, VNets et subnets **une seule fois**, au niveau du Datacenter ; `pmxcfs` réplique la configuration sur tous les nœuds, et chaque *Apply* la transcrit en interfaces réseau réelles. C'est la raison pour laquelle le SDN est traité **ici, une fois le cluster monté** : c'est à plusieurs nœuds qu'il prend tout son sens.

Ce chapitre **présente** le SDN : à quoi il sert, comment il s'organise. Il détaille les deux zones les plus utiles, Simple et VXLAN (*Virtual Extensible LAN*, réseau de niveau 2 transporté sur IP). VXLAN est normalisé par une RFC (*Request for Comments*, document de norme de l'Internet), la 7348.

1.1 Pourquoi le SDN existe dans Proxmox

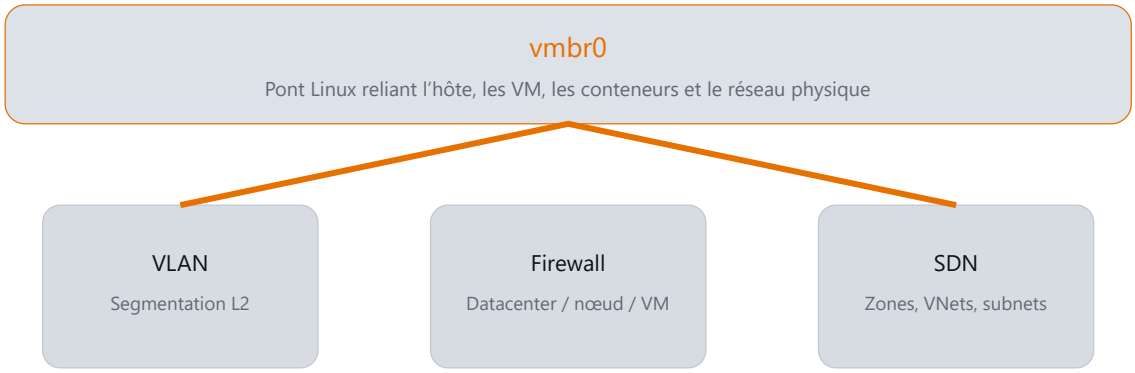
Les ponts classiques suffisent pour un seul hyperviseur. Mais une VM du nœud 1 doit parfois parler à une VM du nœud 3 comme sur un même commutateur, sans VLAN configuré à la main sur chaque nœud.

Le SDN déclare ce réseau une seule fois, au niveau du Datacenter, et Proxmox le reproduit sur tous les nœuds. Même sur un seul nœud, il crée simplement des réseaux isolés.

Dans la pile réseau de Proxmox, le pont `vmbr0` reste le socle. La segmentation VLAN (chapitre 3) et le pare-feu (section 4) s'y ajoutent selon le besoin, et le SDN les orchestre.

Dawan
Ce support est la propriété de Dawan France.

Réseau Proxmox : bridge, VLAN, firewall et SDN



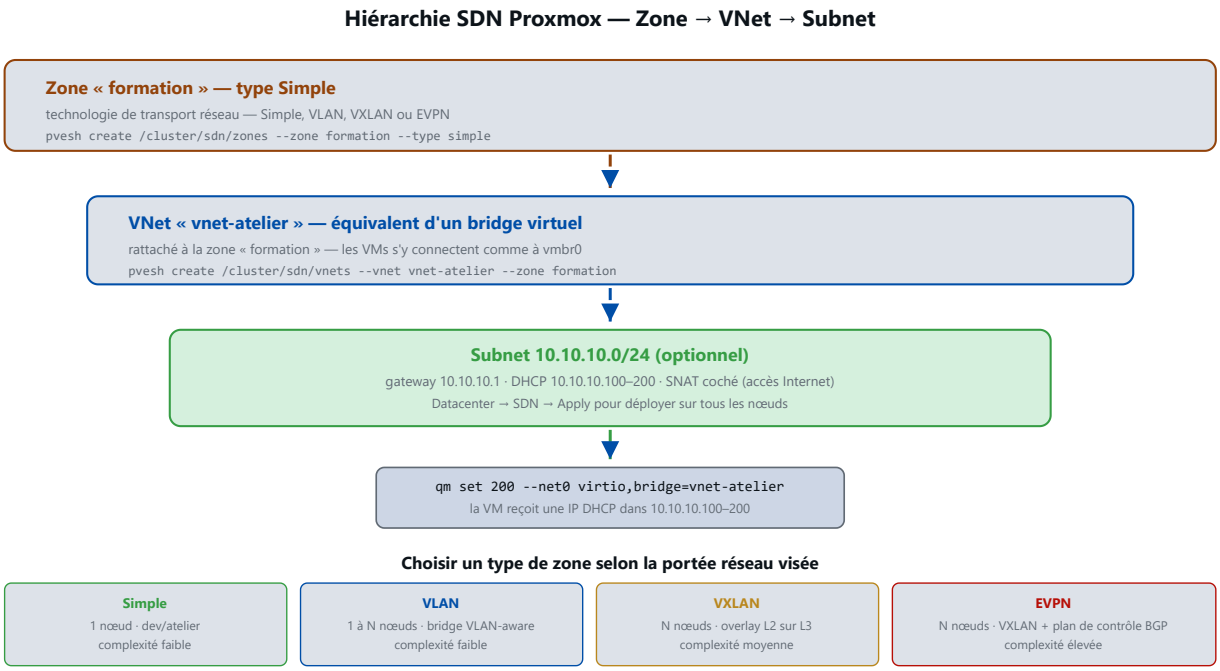
Le SDN est inclus dans Proxmox VE depuis la version 8 : rien à installer. Sa configuration est rangée dans `/etc/pve/sdn/` (`zones.cfg` , `vnets.cfg` , `subnets.cfg`), répliquée sur tous les nœuds par `pmxcfs` . **Ces fichiers se lisent, ils ne s'éditent pas** : on passe par l'interface ou par `pvesh` , sinon le prochain *Apply* écrase la modification.

1.2 Les trois objets fondamentaux du SDN

Le SDN Proxmox s'organise en trois niveaux hiérarchiques, du plus général au plus précis :

Composant	Rôle
Zone	Technologie de transport réseau utilisée (Simple, VLAN, VXLAN, EVPN — *Ethernet Virtual Private Network*)
VNet	Réseau virtuel rattaché à une zone — équivalent d'un bridge classique
Subnet	Plage d'adresses IP associée à un VNet (facultatif)

Ce support est la propriété de Dawan France.



Les trois niveaux sont **imbriqués** : la zone contient le VNet, le VNet contient le subnet. Les invités se branchent sur un VNet comme sur un pont classique.

Les noms (`zonedemo` , `vnetdemo`) font huit caractères au maximum, sans tiret. Les objets, du contenant au contenu :

Terme	Ce que c'est	Exemple
Zone	La technologie de transport : Simple (un nœud), VLAN, VXLAN (entre nœuds), EVPN	`zonetp` , de type Simple
VNet	Le réseau que voient les invités ; sur chaque nœud, un pont Linux à son nom	`vnettp`
Subnet	La plage d'adresses du VNet	`10.10.100.0/24`
Passerelle	L'adresse du nœud dans le subnet, qui mène dehors	`10.10.100.1`
SNAT	La traduction d'adresse qui fait sortir le subnet sous l'adresse du nœud	Case **SNAT** du subnet
Invité	Une VM ou un CT dont la carte est branchée sur le VNet	CT 201 `ct-sdn-test` , en `10.10.100.50`
Apply	La commande qui transforme la configuration en ponts et en règles sur chaque nœud	**Datacenter** → **SDN** → **Apply** ; `pvesh set /cluster/sdn`

Ce support est la propriété de Dawan France.

Un VNet n'existe que dans une zone, un subnet que dans un VNet. D'où l'ordre zone, VNet, subnet à la création, et l'ordre inverse à la suppression. Tant que la configuration n'est pas appliquée, rien n'est créé sur les nœuds : c'est l'état `pending`.

Où voir le SDN sur votre cluster :

Notion	Où la voir
Zones, VNets, subnets	**Datacenter** → **SDN** : **Zones**, **VNets** (et leurs subnets) ; colonne *State* (`pending` avant l'Apply)
La configuration partagée	`/etc/pve/sdn/` : `zones.cfg`, `vnets.cfg`, `subnets.cfg` (se lisent, ne s'éditent pas)
Ce que l'Apply a construit	`/etc/network/interfaces.d/sdn` ; le pont `vnettp` dans `ip link`
Le VNet d'un invité	Sa carte réseau : **Bridge** = `vnettp`

1.3 Types de zones SDN : panorama

Le **type** d'une zone dit comment ses réseaux virtuels sont séparés les uns des autres, et comment ils passent d'un nœud à l'autre. Quatre types sont présentés ici.

Trois notions servent à les lire :

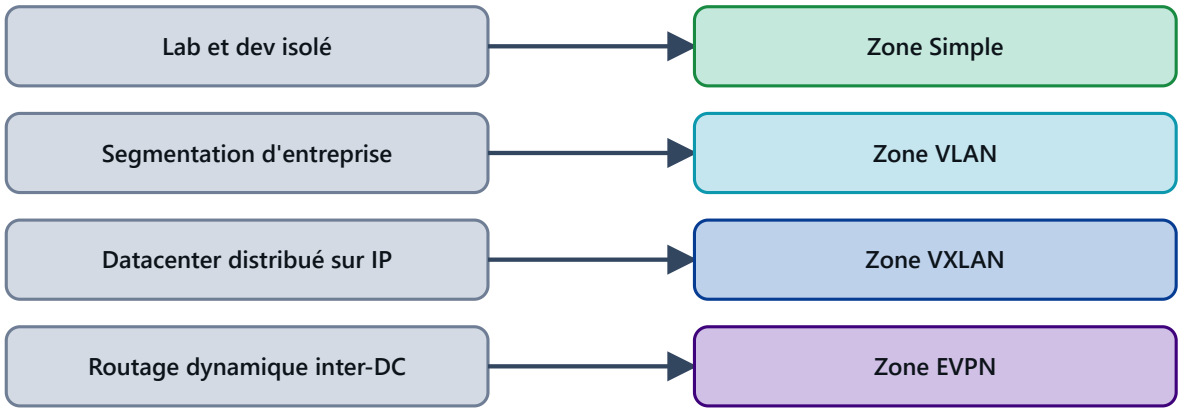
- **Niveau 2** (L2 : *Layer 2*) : le réseau Ethernet. Les machines s'y trouvent par leur adresse MAC, comme branchées sur un même commutateur. **Niveau 3** (L3 : *Layer 3*) : le réseau IP, où des routeurs relient des réseaux différents.
- **Étiquette VLAN** (*tag*) : le numéro ajouté à chaque trame Ethernet pour dire à quel réseau virtuel elle appartient. C'est la norme 802.1Q, vue au chapitre 3 (§2.1) : de 1 à 4094.
- **Encapsuler** : emballer une trame dans un autre paquet pour la transporter, puis la déballer à l'arrivée.

Type	Ce que c'est	Ce qu'il exige du réseau physique	Nœuds
Simple	Un pont isolé, propre à chaque nœud. Les invités d'un même nœud se parlent ; rien ne sort du nœud.	Rien	1
VLAN	Chaque réseau virtuel reçoit son étiquette VLAN, posée sur un pont existant du nœud.	Des commutateurs qui laissent passer ces VLAN	1 à N

Ce support est la propriété de Dawan France.

Type	Ce que c'est	Ce qu'il exige du réseau physique	Nœuds
VXLAN	La trame Ethernet de l'invité est encapsulée dans un paquet IP (UDP) entre les nœuds : un réseau de niveau 2 tendu au-dessus d'un réseau de niveau 3. On parle de réseau **superposé** (*overlay*).	Seulement que les nœuds se joignent en IP	2 et plus
EVPN	Du VXLAN, dont les nœuds s'échangent automatiquement les adresses des invités par **BGP**, le protocole de routage qu'utilisent les opérateurs d'Internet, au lieu d'une liste de pairs écrite à la main.	Un contrôleur BGP déclaré dans le SDN	2 et plus

SDN : quel type de zone pour quel besoin ?



Le type de zone selon le besoin. Un lab ou un réseau de développement isolé : **Simple**. La segmentation d'un réseau d'entreprise : **VLAN**.

Un réseau étendu entre des nœuds reliés en IP : **VXLAN**. Et quand il faut router entre plusieurs datacenters (DC : *datacenter*, centre de données), avec BGP (*Border Gateway Protocol*, protocole de routage entre réseaux) : **EVPN**, l'étape suivante après VXLAN.

La zone **Simple** est la seule qui ne demande ni pair ni réglage du réseau physique : c'est par elle que commence la pratique.

- 📖 RFC 7348 — la norme VXLAN, pour qui veut lire le format de l'encapsulation avant la partie VXLAN

1.4 En pratique : créer, appliquer, étendre

Dawan Ce support est la propriété de Dawan France.

Tout se fait dans **Datacenter** → **SDN**. On crée d'abord la **zone** (**Zones** → **Add**, type *Simple*), puis un **VNet** dans cette zone (**VNets** → **Create**), puis éventuellement un **subnet** avec sa passerelle et la case **SNAT**.

Rien n'existe sur les nœuds tant qu'on n'a pas cliqué sur **Apply** : les objets restent à l'état `pending`. Une fois appliqué, le VNet apparaît sur chaque nœud comme un pont, et une VM s'y branche en choisissant ce VNet comme **Bridge** de sa carte réseau.

À faire sur votre cluster : une zone Simple, puis sa suppression.

Dans l'interface web, menu **Datacenter** → **SDN** :

1. **Zones** → **Add** → **Simple** : ID `zonetp`, IPAM `pve`, puis **Add**. L'IPAM (*IP Address Management*, gestion des adresses IP) `pve` est celle de Proxmox : elle tient la liste des adresses attribuées dans les subnets.
2. **VNets** → **Create** : Name `vnettp`, Zone `zonetp`. Le champ **Tag** est grisé : une zone Simple n'a ni VLAN ni VXLAN.
3. Sélectionnez `vnettp`, puis **Subnets** → **Create** : Subnet `10.10.200.0/24`, Gateway `10.10.200.1`, case **SNAT** cochée.
4. **Datacenter** → **SDN** → **Apply**. Le tableau **Status** montre `zonetp` sur chacun des trois nœuds.

Dans le Shell du nœud 1 (interface web → votre nœud 1 → **Shell**) :

```
ip -br link show vnettp   # le pont du VNet, créé sur le nœud par l'Apply
```

Résultat attendu : `vnettp UNKNOWN …`. Le pont existe ; il est `UNKNOWN` tant qu'aucun invité n'y est branché. Avant l'Apply, la même commande répond `Device "vnettp" does not exist.`

On supprime ensuite dans l'**ordre inverse** de la création, car un subnet vit dans un VNet et un VNet dans une zone :

1. sélectionnez `vnettp` → panneau **Subnets** → le subnet → **Remove** ;
2. **VNets** → `vnettp` → **Remove** ;
3. **Zones** → `zonetp` → **Remove** ;
4. **Datacenter** → **SDN** → **Apply**.

Dans le Shell du nœud 1 :

Dawan
Ce support est la propriété de Dawan France.

```
ip -br link show vnettp   # le pont a-t-il disparu ?
```

Résultat attendu : `Device "vnettp" does not exist.` Le pont a disparu des nœuds.

Une zone **VXLAN** se crée de la même façon, avec la liste des adresses des nœuds (*Peers*). Un même réseau de niveau 2 est alors tendu entre les trois nœuds, et une VM qui migre garde sa connectivité. Les noms des zones et des VNets font huit caractères au plus, sans tiret.

- 📖 Documentation Proxmox — Software-Defined Network

À retenir : le **SDN** déclare les réseaux une seule fois, au niveau du Datacenter, et Proxmox les déploie sur tous les nœuds. Trois objets imbriqués : la **zone** (la technologie de transport), le **VNet** (le réseau des invités), le **subnet** (sa plage d'adresses) ; rien n'est créé avant l'**Apply**. **Simple** pour un réseau isolé par nœud, **VXLAN** pour un réseau tendu entre les nœuds.

Ce support est la propriété de Dawan France.

2. Le pare-feu Proxmox

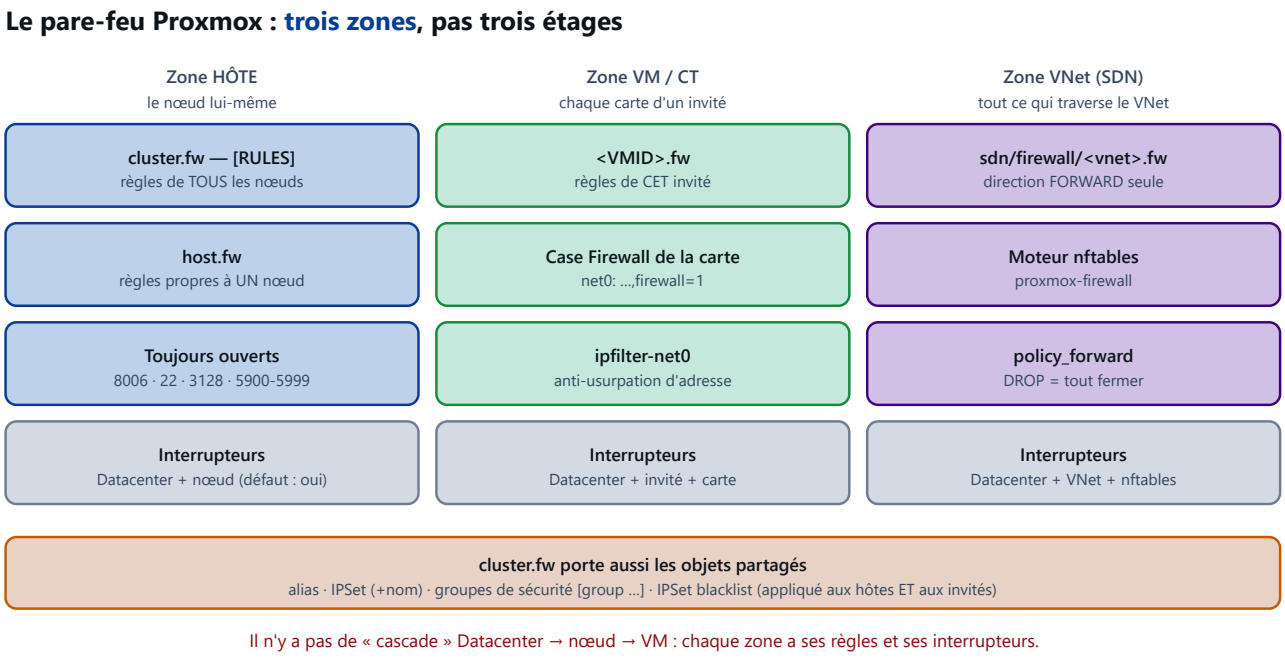
Jusqu'ici, tout peut parler à tout : une VM web atteint la base de données, un conteneur de test joint l'interface d'administration. Le **pare-feu intégré** décide *qui a le droit de parler à qui*. Ce chapitre en présente le modèle et les pièges ; l'atelier 1 le met en œuvre.

2.1 Un pare-feu distribué, organisé en zones

La configuration du pare-feu est rangée dans `/etc/pve`, donc répliquée sur tous les nœuds, mais **chaque nœud applique lui-même les règles** qui le concernent. Il le fait par `iptables` (service classique `pve-firewall`) ou par `nftables` (nouveau service `proxmox-firewall`). Trois **zones**, chacune avec ses règles :

Zone	Ce qu'elle filtre	Directions possibles	Où se trouvent ses règles
Hôte	Le trafic à destination ou en provenance **du nœud** lui-même	`IN`, `OUT` (et `FORWARD` pour le trafic routé par l'hôte, moteur nftables uniquement)	`/etc/pve/firewall/cluster.fw` (tous les nœuds) et `/etc/pve/nodes/<nœud>/host.fw` (un nœud, prioritaire)
VM / CT	Le trafic de **chaque carte réseau** d'un invité	`IN`, `OUT`	`/etc/pve/firewall/<VMID>.fw`
VNet	Tout le trafic qui **traverse un VNet** du SDN, d'invité à invité ou d'invité à hôte	`FORWARD` uniquement	`/etc/pve/sdn/firewall/<vnet>.fw` — moteur nftables uniquement

Ce support est la propriété de Dawan France.



Les règles du Datacenter ne filtrent pas les VM. On lit souvent « Datacenter, puis nœud, puis VM, en cascade ». Faux : les règles de `cluster.fw` sont celles **des nœuds**.

Vérifié sur le lab : avec `IN Ping(DROP)` au Datacenter, le nœud ne répond plus au ping, un conteneur si. Le Datacenter partage avec les invités des **objets** : alias, IP sets, groupes de sécurité (§2.4).

2.2 Ce qui doit être activé pour qu'une règle s'applique

Le pare-feu est désactivé à l'installation : `pve-firewall status` répond `Status: disabled/running`. Le service tourne mais n'applique rien. Pour qu'une règle agisse, plusieurs interrupteurs doivent être levés, différents selon la zone (chacun vérifié sur le lab en l'éteignant seul) :

Pour filtrer…	Il faut	Vérification sur le lab
Le nœud	`enable: 1` au Datacenter **et** l'option `enable` du nœud (vaut `1` par défaut)	—
Un invité	1. `enable: 1` au **Datacenter**	Invité configuré, Datacenter désactivé → le ping et SSH passent
	2. `enable: 1` dans les **options de l'invité**	—

Ce support est la propriété de Dawan France.

Pour filtrer…	Il faut	Vérification sur le lab
	3. La case **Firewall** de la **carte** (`net0: …,firewall=1`)	Carte à `firewall=0` → le trafic passe, règles ou pas
Un VNet	Datacenter, options du VNet, et moteur **nftables** sur les nœuds	Moteur classique → règles du VNet ignorées

Le pare-feu **du nœud n'intervient pas** dans le filtrage des invités, contrairement à ce que demandent beaucoup de supports.

Les politiques décident du sort de ce qu'aucune règle ne nomme : `policy_in`, `policy_out` (`ACCEPT`, `DROP`, `REJECT`) et `policy_forward` pour le transit. `DROP` jette sans réponse ; `REJECT` renvoie un refus, mais révèle la machine. La règle d'exploitation : **politique `DROP`, puis une règle `ACCEPT` par flux nécessaire**.

Où voir le pare-feu sur votre cluster :

Notion	Où la voir
L'interrupteur général et les règles des nœuds	**Datacenter** → **Firewall** (règles, **Options**) ; `/etc/pve/firewall/cluster.fw`
Le pare-feu d'un nœud	**Nœud** → **Firewall** ; `/etc/pve/nodes/<nœud>/host.fw`
Le pare-feu d'un invité	La VM ou le CT → **Firewall** (règles, **Options**) ; la case **Firewall** de sa carte dans **Hardware** / **Network**
L'état du service	`pve-firewall status`
Les règles réellement appliquées	`iptables-save` (moteur classique) ou `nft list ruleset` (nftables)

2.3 Écrire les règles : syntaxe, macros, ordre

Une règle s'écrit `DIRECTION ACTION [OPTIONS]`, par exemple `IN ACCEPT -p tcp -dport 8006 -source 10.44.1.0/24`. Un `|` en tête la désactive.

Les **macros** regroupent protocole et ports sous un nom (une centaine, `man pve-firewall`) : `IN SSH(ACCEPT)`. Quelques-unes :

Macro	Équivalent
`SSH`	tcp 22

Ce support est la propriété de Dawan France.

Macro	Équivalent
`Web`	tcp 80 **et** tcp 443
`HTTPS`	tcp 443
`Ping`	icmp `echo-request`
`DNS`	udp 53 et tcp 53
`PostgreSQL`	tcp 5432

Deux comportements doivent être connus, parce qu'ils trompent à coup sûr :

- **L'ordre compte, et les nouvelles règles arrivent en tête.** Les règles sont évaluées de haut en bas, la première qui correspond décide. Une règle créée par l'API ou l'interface est insérée en **position 0** : après deux créations successives, la seconde apparaît au-dessus de la première dans le fichier.
- **Une règle créée par `pvesh` sans `--enable 1` est désactivée.** Vérifié sur le lab : `pvesh create /cluster/firewall/rules --type in --action DROP --macro Ping` écrit `|IN Ping(DROP)` dans `cluster.fw` — le `|` initial signifie « désactivée », et la règle n'apparaît dans aucune chaîne. L'interface, elle, coche **Enable** par défaut.

Le pare-feu est à états (*stateful*) : accepter une connexion entrante accepte aussi ses réponses. Aucune journalisation par défaut : on l'active par zone (`log_level_in`) ou par règle (`-log`), et on la lit dans **Firewall** → **Log**.

2.4 Nommer les adresses et réutiliser les règles

Trois objets, définis au Datacenter (**Datacenter** → **Firewall**), évitent de répéter une adresse dans vingt règles :

Objet	Contenu	Utilisation dans une règle
Alias	**Une** adresse ou un réseau, nommé	`-source poste_admin`
IP set	Une **liste** d'adresses, de réseaux ou d'alias	`-source +admins` (le `+` désigne un IP set)
Groupe de sécurité	Une liste de **règles**, nommée	Dans un invité : `GROUP web`

Un groupe de sécurité se définit une fois dans `cluster.fw` et s'applique à autant d'invités qu'on veut ; le modifier modifie la protection de tous :

Ce support est la propriété de Dawan France.

```
# /etc/pve/firewall/cluster.fw
[group web]
IN Web(ACCEPT)
IN Ping(ACCEPT) -source +admins

# /etc/pve/firewall/101.fw
[OPTIONS]
enable: 1
policy_in: DROP

[RULES]
GROUP web
IN SSH(ACCEPT) -source +admins
```

Le groupe `web` ouvre le web à tous et le ping aux seuls membres de l'IP set `admins` ; la VM 101 l'applique par une ligne, `GROUP web`. Si l'équipe d'administration change d'adresse, on modifie l'IP set : toutes les règles suivent.

Deux IP sets ont un rôle réservé. **`management`** regroupe les adresses autorisées à administrer les **hôtes** (le réseau local du nœud y est mis d'office, rien d'autre). **`blacklist`** regroupe les adresses jetées par chaque hôte et chaque invité.

2.5 Activer le pare-feu sans se couper l'accès

Avec une politique d'entrée `DROP` au Datacenter, Proxmox ouvre d'office certains ports, **depuis l'IP set `management` seulement**. Ce sont l'interface web (8006), SSH (22), SPICE (protocole de console à distance, 3128), les consoles (5900-5999), la migration (60000-60050), et Corosync sur le réseau du cluster.

Si vous administrez depuis un autre réseau (VPN — *Virtual Private Network*, réseau privé virtuel —, autre site), l'activation vous coupe. `pve-firewall localnet` affiche le réseau que Proxmox considère comme local : celui de votre `vmbr0`.

> **Ordre absolu : les règles d'accès d'abord, l'activation ensuite.** Si votre poste n'est pas dans ce réseau, ajoutez son adresse à l'IP set `management`, ou créez les règles `ACCEPT` correspondantes, **avant** de cocher **Firewall**.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Gardez une session ouverte pendant l'activation. En cas d'erreur, la récupération se fait par la console (IPMI, iDRAC, ou l'écran du serveur) : `enable: 0` dans `/etc/pve/firewall/cluster.fw`, ou `pve-firewall stop`.

L'IP set `management` ne couvre pas Ceph, ni le VXLAN. Vu sur le lab : pare-feu activé sans règle pour les réseaux Ceph, le cluster est passé en `HEALTH_WARN` (`8 pgs inactive, 8 pgs peering`) et les écritures se sont figées.

Deux règles `IN Ceph(ACCEPT)`, une par réseau Ceph, l'ont rétabli en trois minutes.

Une zone VXLAN demande de même `IN ACCEPT -p udp -dport 4789` depuis le réseau de `vmbr0`. Posez ces règles **avant** d'activer.

Via l'interface : Datacenter → Firewall → Options → Firewall → Yes, après avoir vérifié **Datacenter → Firewall → IPSet → management**.

Équivalent CLI (**Dans le Shell du nœud**) :

```
# Une ouverture explicite d'abord — toujours --enable 1, sinon la règle est créée désactivée
pvesh create /cluster/firewall/rules --type in --action ACCEPT --macro SSH \
  --source 10.99.0.0/24 --enable 1 --comment "SSH depuis le VPN d'administration"
pvesh set /cluster/firewall/options --enable 1   # puis seulement : activer
pve-firewall status   # l'état du pare-feu
```

Résultat attendu : aucune sortie pour les deux premières commandes, puis `Status: enabled/running`.

Pour un invité, même principe dans son propre fichier, et le troisième interrupteur : la case **Firewall** de sa carte.

```
# /etc/pve/firewall/101.fw — un serveur web, administré depuis un seul réseau
[OPTIONS]
enable: 1
policy_in: DROP

[RULES]
```

Ce support est la propriété de Dawan France.

```
IN Web(ACCEPT)
IN SSH(ACCEPT) -source 10.44.1.0/24
```

2.6 Diagnostiquer

`pve-firewall status` (actif ? configuration valide ?), `pve-firewall localnet` (réseau local), `pve-firewall simulate` (tel flux passerait-il ?, sans envoyer de paquet), et **Firewall** → **Log**.

Une règle « ne fait rien ». Elle est désactivée (`.

Règle posée au Datacenter pour protéger une VM, sans effet. Les règles du Datacenter filtrent les **nœuds** : la règle doit aller dans le pare-feu de l'invité, ou dans un groupe de sécurité.

Règles de VNet sans effet. Moteur nftables non activé sur le nœud, ou invités non redémarrés depuis.

Plus d'accès à l'interface après activation. Poste hors du réseau local et absent de l'IP set `management` → console, `enable: 0`.

Un service répond « refusé » au lieu de ne pas répondre. Ce n'est pas le pare-feu : le paquet est passé et rien n'écoute. Un **silence** (délai dépassé) signe un `DROP`.

- 📖 Documentation Proxmox — Firewall
- 📖 Guide d'administration — pare-feu, zones, macros, nftables
- 📖 Proxmox VE : configurer le pare-feu — IT-Connect
- 🎥 Configuration complète du firewall Proxmox en atelier, avec le SDN (vidéo)

> Vous ajoutez au Datacenter une règle qui refuse le SSH vers la VM 300, mais la VM répond toujours sur le port 22. Le pare-feu du Datacenter est pourtant activé. Qu'est-ce qui manque ?
> ▼ Réponse
> **La règle n'est pas au bon endroit, et les interrupteurs de la VM ne sont pas levés.** Les règles du Datacenter protègent les hôtes, pas les invités. Pour filtrer une VM, la règle se pose dans le pare-feu **de la VM**. Trois conditions doivent être réunies : le pare-feu du Datacenter activé, celui de la VM activé dans ses options, et la case `firewall=1` sur sa carte réseau.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

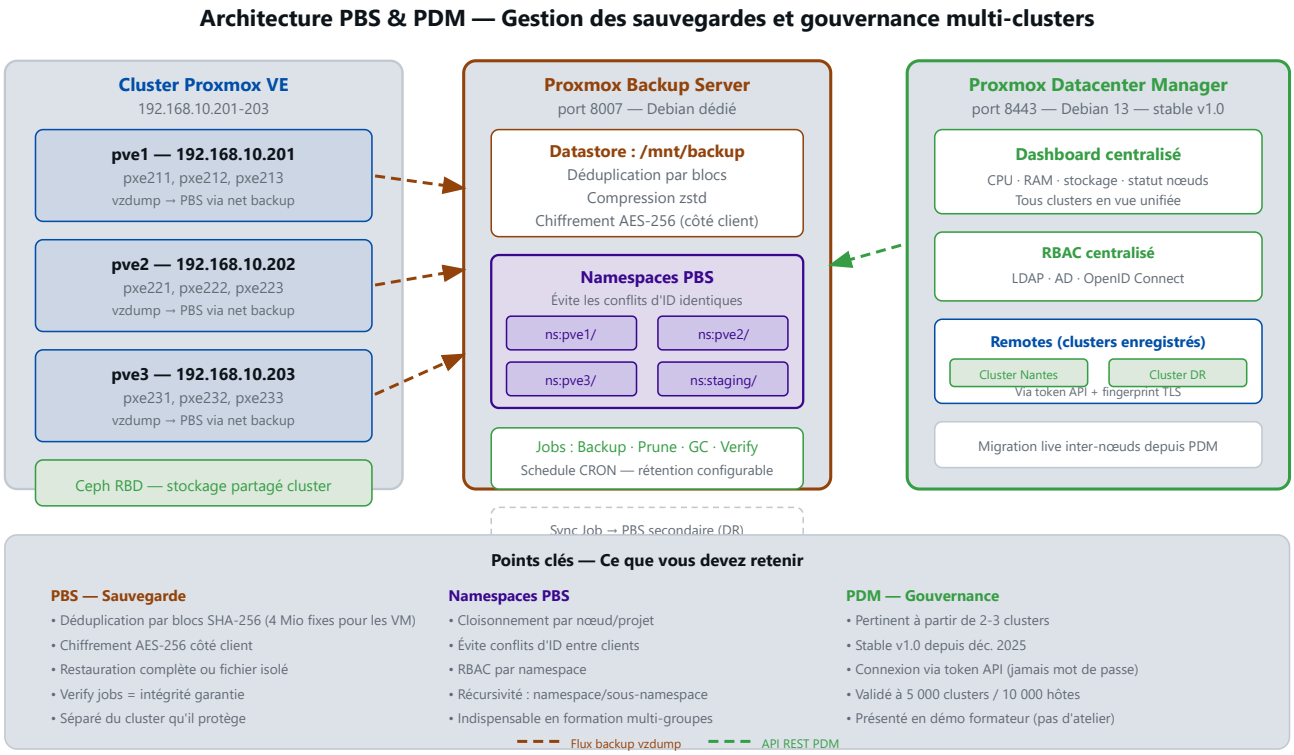
À retenir : le pare-feu Proxmox est **distribué** (configuration dans `/etc/pve`) et **désactivé à l'installation**. Il agit par zones : l'hôte, les invités, les VNets.

Une règle ne s'applique que si tous ses interrupteurs sont levés (Datacenter, invité, case `firewall=1` sur la carte). On écrit les règles d'accès **avant** d'activer une politique `DROP`, et on nomme les adresses (alias, IP sets, groupes de sécurité).

> **▶ À pratiquer maintenant — Atelier 1**
>
> D'abord activer le pare-feu sans se couper l'accès. Puis protéger trois machines web, applicative et base de données, avec un groupe de sécurité par rôle.

Ce support est la propriété de Dawan France.

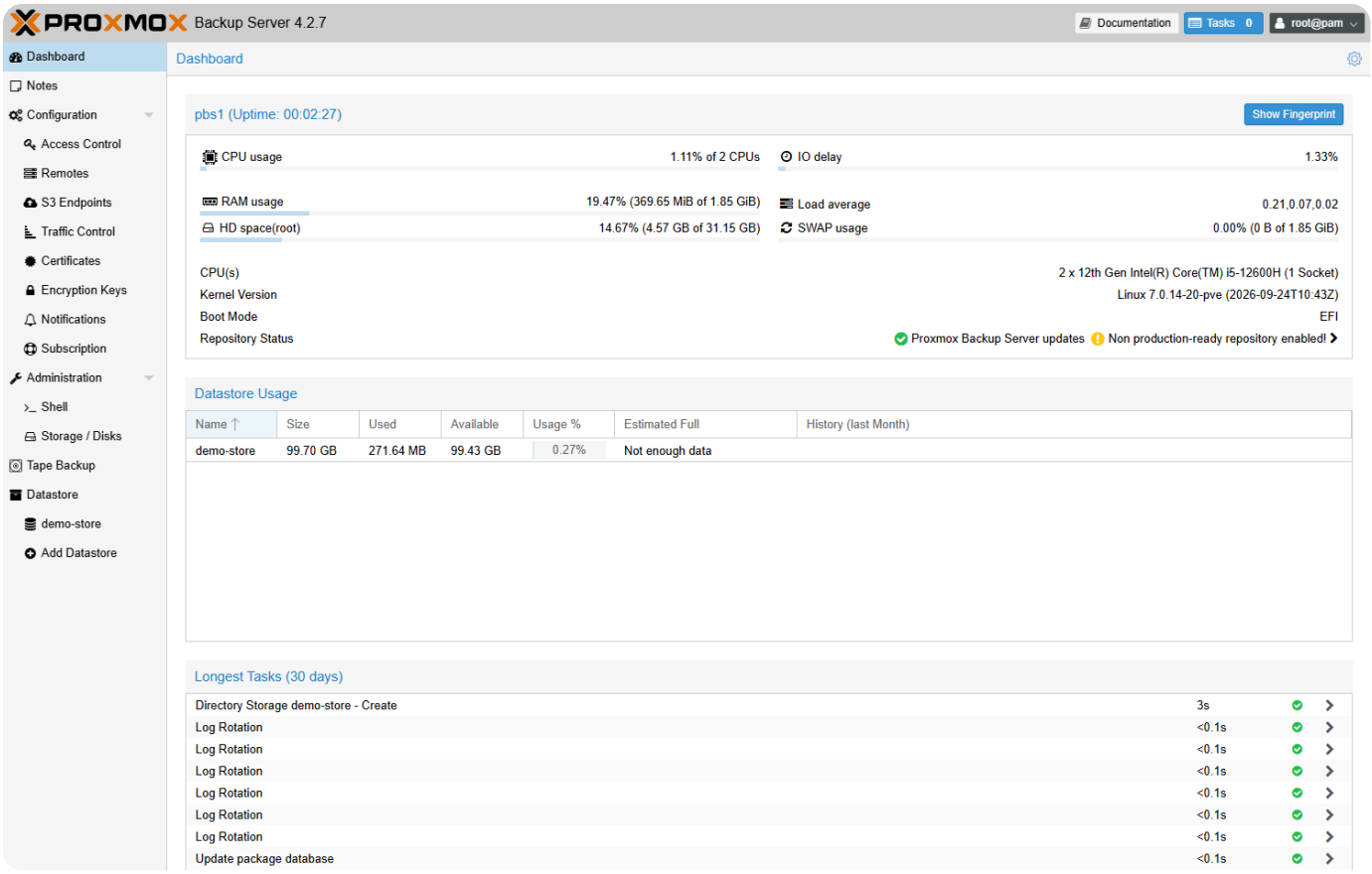
3. Proxmox Backup Server : pourquoi, et comment ça marche



PBS est un **produit distinct de Proxmox VE**, avec sa propre installation, sa propre interface web et son propre port. C'est le **8007**, à ne pas confondre avec le 8006 de PVE. C'est le premier point à poser, parce qu'il explique la logique de tout le chapitre : la sauvegarde ne vit pas sur l'hyperviseur qu'elle protège.

Voici l'interface du PBS du lab, tel qu'il se présente à la connexion, juste après la création de son datastore :

Ce support est la propriété de Dawan France.



Le tableau de bord de PBS 4.2.7 (lab) — l'essentiel se lit dans le tableau du milieu.

Trois zones à repérer dès la première connexion.

Le panneau `Datastore Usage`, sous les informations du serveur, est celui qu'on regarde en premier au quotidien. Il donne l'espace consommé et, surtout, la colonne **Estimated Full** : la date projetée de saturation, calculée à partir de la tendance.

Sur un datastore neuf, elle affiche *Not enough data* : il faut quelques jours d'historique. C'est l'indicateur qui prévient avant la panne, plutôt qu'après.

Le menu `Datastore`, en bas à gauche, liste les magasins de sauvegarde. Un PBS peut en héberger plusieurs, chacun sur un stockage différent — ici un seul, `demo-store`.

Le `Task Summary (30 days)`, plus bas sur la page, compte les sauvegardes, les élagages, les collectes d'ordures et les vérifications des trente derniers jours. Une sauvegarde qui n'est jamais vérifiée n'est pas une sauvegarde. La ligne `Verify` doit rester à zéro échec ; c'est l'objet de la section sur la vérification d'intégrité.

> **« No valid subscription » n'est pas une erreur.** Comme pour PVE, PBS fonctionne sans abonnement ; le bandeau signale seulement l'usage du dépôt sans souscription. La ligne

Ce support est la propriété de Dawan France.

> `Repository Status` du panneau du serveur l'indique d'ailleurs explicitement (*Non production-ready repository enabled!*) — un rappel utile avant une mise en production, où l'on préférera le dépôt entreprise.

3.1 Le problème de vzdump et pourquoi PBS existe

Depuis les débuts de Proxmox VE, `vzdump` a assuré la sauvegarde des VMs et conteneurs. Son principe est simple : à chaque sauvegarde, il produit une archive complète de la VM — un fichier `.vma.zst` pour les VMs KVM, un `.tar.zst` pour les conteneurs LXC.

C'est fiable et facile à comprendre. Mais c'est aussi inefficace.

Prenons un exemple concret : une VM Debian avec 30 Go de données. Vous la sauvegardez tous les jours. Chaque jour, `vzdump` copie les 30 Go en entier — même si seuls 200 Mo ont changé depuis la veille.

Une semaine de rétention consomme 210 Go. Un mois : 900 Go. Pour une infrastructure de 20 VMs, le dimensionnement du stockage de backup devient problématique.

La sauvegarde incrémentale existe depuis des décennies dans des outils comme Bacula, Amanda ou Veeam. Le principe : ne transférer et ne stocker que ce qui a changé depuis la dernière sauvegarde. Proxmox Server Solutions a implémenté ce principe dans **Proxmox Backup Server** (PBS), lancé en 2020, écrit entièrement en **Rust**.

Le choix de Rust est délibéré : c'est un langage compilé, sans garbage collector, qui garantit des performances proches du C avec une sécurité mémoire par construction. PBS partage cette implémentation avec les outils de plus en plus nombreux de l'écosystème Proxmox.

- 📖 Documentation Proxmox Backup Server
- 📖 PBS — introduction et concepts
- 🎥 apalrd's adventures — le fonctionnement interne de PBS, la déduplication et l'architecture (vidéo en anglais)

3.2 Déduplication par chunks — le mécanisme clé

La déduplication de PBS découpe chaque sauvegarde en **blocs** (*chunks*), calcule l'empreinte SHA-256 de chacun, et ne stocke qu'une fois les blocs identiques. La façon de découper dépend de ce qu'on sauvegarde, et la différence explique les performances de PBS.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

Pour un disque de VM : des blocs fixes de 4 Mio. Un disque virtuel est un grand fichier dont les blocs ne se déplacent pas : le secteur 1 000 reste le secteur 1 000.

Un découpage à taille fixe est donc idéal. Il se combine avec le **dirty bitmap** de QEMU, qui note les blocs modifiés depuis la dernière sauvegarde : PBS n'a même pas besoin de relire les autres.

Pour un conteneur ou des fichiers : des blocs de taille variable. Une archive de fichiers pose un problème que le découpage fixe gère mal.

Insérer un octet au début d'un fichier décale toutes les frontières suivantes, et tous les blocs changent d'empreinte. C'est le problème du *boundary shift*.

PBS découpe alors selon le **contenu** : un hachage glissant (*Buzhash*) parcourt les données et place une frontière quand sa valeur remplit une condition. Une insertion ne déplace que les frontières voisines ; le reste de l'archive produit exactement les mêmes blocs qu'avant.

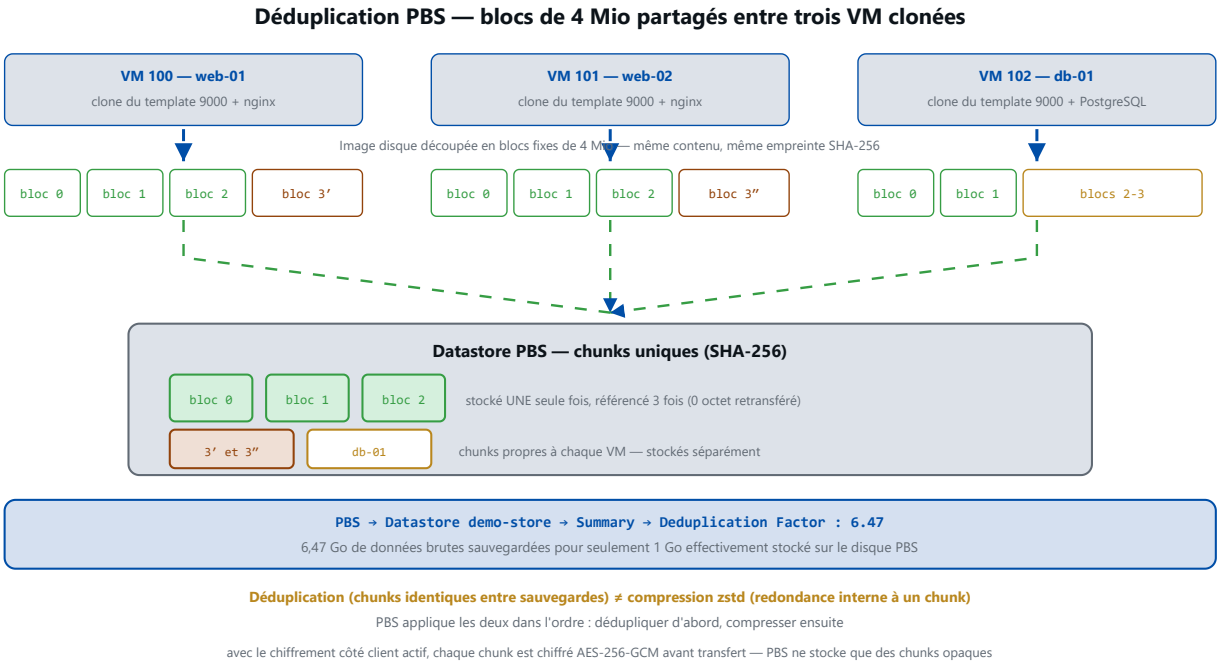
Le parcours d'une sauvegarde (disque de VM ou archive de conteneur) :

1. Découpage en blocs : 4 Mio fixes (VM) ou taille variable (conteneur, fichiers).
2. Pour chaque bloc : calcul de l'empreinte SHA-256.
3. Le bloc existe-t-il dans le datastore PBS ? - **oui** → seule la référence est enregistrée (0 octet transféré) ; - **non** → compression zstd + stockage du bloc.

En pratique, dix VM **clonées du même template** partagent la plupart de leurs blocs de 4 Mio. À la même position dans l'image disque se trouve le même contenu, donc la même empreinte.

Ces blocs ne sont stockés qu'une fois dans le datastore ; seuls les blocs que chaque VM a modifiés depuis le clonage (données applicatives, journaux, configuration) sont stockés à part. Dix VM installées séparément depuis l'ISO, au contraire, rangent leurs fichiers à des positions différentes et partagent beaucoup moins de blocs.

Dawan
Nous formons votre avenir
Ce support est la propriété de Dawan France.



Le facteur de déduplication s'affiche dans l'interface de PBS, **Datastore → demo-store → Summary**, et dans le journal de chaque ramasse-miettes (*garbage collection*, §5.1), qui fait le compte exact :

Dans le Shell de PBS (interface web de PBS → **Administration → Shell**) :

```
# Sur PBS : lancer un ramasse-miettes, puis lire la fin de son journal
proxmox-backup-manager garbage-collection start demo-store
```

Résultat attendu (fin du journal) :

```
Original data usage: 79.00 GiB
On-Disk usage: 12.20 GiB (15.44%)
On-Disk chunks: 20510
Deduplication factor: 6.47
Average chunk size: 624.18 KiB
TASK OK
```

Un facteur de 6,47 signifie que 79 Gio de données sauvegardées tiennent dans 12,2 Gio. Ce n'est pas de la compression : ce sont les blocs identiques, stockés une seule fois, qui font la différence.

Dawan
Ce support est la propriété de Dawan France.

L'effet est spectaculaire sur un parc homogène. Dix VM Debian partagent l'essentiel de leur système : le second système d'exploitation identique ne coûte presque rien. C'est ce qui rend une rétention longue économiquement tenable — trente sauvegardes quotidiennes ne pèsent pas trente fois la première.

> **Le ratio se dégrade sur des données déjà compressées ou chiffrées** : des vidéos, des archives, ou des disques de VM chiffrés n'ont aucun bloc en commun. Un ratio proche de 1 sur un datastore n'est donc pas un dysfonctionnement, mais le signe que les données ne s'y prêtent pas.

> **Déduplication vs compression — ne pas confondre :** La compression (zstd) réduit la taille d'un chunk en éliminant les redondances *internes* à ce chunk. La déduplication élimine les chunks *identiques entre différentes sauvegardes*. PBS applique les deux : chaque chunk unique est d'abord dédupliqué, puis compressé avant d'être écrit sur disque.

Revenons à la définition de base : une **sauvegarde** est une copie des données, gardée ailleurs, pour les restaurer. Garder trente copies d'un même système coûterait trente fois sa taille, sauf si l'on ne garde qu'une fois ce qui se répète. C'est le principe de PBS :

Terme	Ce que c'est	Dans le laboratoire
Chunk	Un bloc de la sauvegarde : 4 Mio fixes pour une VM, taille variable pour un conteneur	—
Empreinte SHA-256	Un identifiant calculé sur le contenu : même contenu, même empreinte	—
Datastore	Le stockage de PBS, où chaque chunk n'existe qu'une fois	`demo-store`
Index	La liste ordonnée des empreintes qui reconstituent une sauvegarde	Une par sauvegarde, dans **Content**
Déduplication	Un bloc déjà présent n'est ni transféré ni stocké : seule sa référence est enregistrée	Dix clones du même template partagent leurs chunks
Facteur de déduplication	Données sauvegardées ÷ place occupée	`Deduplication factor` du ramasse-miettes

Ce support est la propriété de Dawan France.

Terme	Ce que c'est	Dans le laboratoire
Dirty bitmap	Les blocs d'un disque de VM modifiés depuis la dernière sauvegarde : seuls ceux-là sont relus	Sauvegarde incrémentale rapide d'une VM allumée
Ramasse-miettes (*garbage collection*)	Libère les blocs des sauvegardes supprimées par la rétention (§5.1)	`proxmox-backup-manager garbage-collection start demo-store`

Une sauvegarde ne contient que son index ; le datastore garde les chunks. C'est pourquoi trente sauvegardes quotidiennes ne pèsent pas trente fois la première. Un chunk de données déjà compressées ou chiffrées ne ressemble à aucun autre : sur ces données, le facteur reste proche de 1.

Où voir la déduplication :

Notion	Où la voir
Le facteur et l'espace occupé	Interface de PBS : **Datastore** → **demo-store** → **Summary**
Le compte exact	Le journal du ramasse-miettes : `Original data usage`, `On-Disk usage`, `Deduplication factor`
Les sauvegardes (les notices)	**Datastore** → **demo-store** → **Content** ; côté Proxmox VE, le stockage PBS → **Backups**
La part transférée à chaque sauvegarde	Le journal de la tâche de sauvegarde, côté Proxmox VE

3.3 Chiffrement côté client — un modèle de sécurité fort

PBS intègre un mécanisme de chiffrement **AES-256 en mode GCM** avec une particularité importante : le chiffrement est **côté client**. Cela signifie que les données sont chiffrées sur le nœud Proxmox VE *avant* d'être envoyées vers PBS.

Pourquoi est-ce important ? Dans un modèle de chiffrement côté serveur (comme la plupart des solutions de backup traditionnelles), l'administrateur du serveur de backup peut techniquement accéder aux données déchiffrées. Avec le chiffrement côté client de PBS :

- La clé de chiffrement ne quitte jamais le nœud PVE
- PBS stocke uniquement des chunks chiffrés opaques
- Même un accès physique au serveur PBS ne permet pas de lire les données

Ce support est la propriété de Dawan France.

- Ce modèle est conforme aux exigences de confidentialité les plus strictes. Il répond au RGPD (Règlement général sur la protection des données), à HDS (Hébergement de données de santé) et à PCI-DSS (*Payment Card Industry Data Security Standard*)

Dans Proxmox VE, le chiffrement est une propriété **du stockage PBS**, pas de chaque VM : toutes les sauvegardes envoyées par ce stockage sont chiffrées avec la même clé. Il s'active dans **Datacenter → Storage → pbs-backup → Edit → onglet Encryption → Auto-generate a client encryption key**, ou en ligne de commande :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Générer la clé et l'attacher au stockage PBS (clé stockée côté PVE, sans phrase de passe)
pvesm set pbs-backup --encryption-key autogen

# Relire les métadonnées de la clé : son empreinte, jamais le secret
proxmox-backup-client key show /etc/pve/priv/storage/pbs-backup.enc
```

Résultat attendu de `key show` — sur `pxe211` avec une clé d'essai créée au même format (`proxmox-backup-client key create /tmp/demo-cle.enc --kdf none`, supprimée ensuite) :

```
┌─────────────┬──────────────────────────────────────────────────────────────────────────────┐
│ Name        │ Value                                                                        │
╞═════════════╪══════════════════════════════════════════════════════════════════════════════╡
│ path        │ "/tmp/demo-cle.enc"                                                          │
├─────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ kdf         │ none                                                                         │
├─────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ created     │ Tue Sep 29 14:01:27 2026                                                     │
├─────────────┼──────────────────────────────────────────────────────────────────────────────┤
```

Ce support est la propriété de Dawan France.

```
│ modified    │ Tue Sep 29 14:01:27 2026
│
├─────────────┼──────────────────────────────────────────────────────────────
──────────────────────────┤
│ fingerprint │
12:37:94:df:f6:ea:64:4b:9f:1e:c1:27:05:e8:c6:fe:10:ca:30:ac:eb:02:2e:4a:7a:0c:be:f7:
5c:f0:b9:3e │
└─────────────┴──────────────────────────────────────────────────────────────
──────────────────────────┘
```

`kdf none` n'est pas un oubli : Proxmox VE doit pouvoir lire la clé seul, à 3 h du matin, sans personne pour taper une phrase de passe. La clé vit donc en clair dans `/etc/pve/priv/`, lisible par root seulement.

L'empreinte est ce que PBS enregistre avec chaque sauvegarde : c'est elle qui dira, le jour de la restauration, si vous avez la bonne clé. Le client autonome, lui, peut protéger sa clé par une phrase de passe (`proxmox-backup-client key create --kdf scrypt`).

> **Avertissement critique :** si vous perdez la clé de chiffrement, vos sauvegardes chiffrées deviennent **définitivement irrécupérables** — même pour l'équipe Proxmox. Or la clé vit sur le cluster : si le cluster brûle, elle brûle avec lui.
>
> Conservez-en toujours une copie **hors du cluster**, dans un endroit sûr et physiquement séparé : coffre-fort d'entreprise, gestionnaire de secrets (HashiCorp Vault), ou **papier**. L'onglet Encryption propose de la télécharger et de l'imprimer sous forme de QR code (`proxmox-backup-client key paperkey` en ligne de commande).

À retenir : PBS est un produit à part (port **8007**). Il découpe les sauvegardes en **chunks**, identifiés par leur empreinte SHA-256, et ne stocke qu'une fois chaque bloc identique : les sauvegardes suivantes ne coûtent que ce qui a changé. Le **chiffrement côté client** empêche le serveur de lire les données ; la clé perdue, les sauvegardes le sont aussi.

Dawan
Ce support est la propriété de Dawan France.

4. PBS : déployer et connecter

Le principe est posé. Cette section installe PBS là où il doit être — jamais sur l'infrastructure qu'il protège — puis le relie aux nœuds Proxmox.

4.1 Architecture de déploiement PBS

PBS s'installe sur sa propre machine, séparée du cluster Proxmox VE. Cette séparation est indispensable : un serveur de backup ne doit jamais être hébergé sur l'infrastructure qu'il protège. Si un hyperviseur tombe en panne, PBS doit rester accessible pour restaurer les VMs.

Architecture recommandée : `pve-node1` et `pve-node2`, sur le réseau de production (VLAN 10), sauvegardent vers PBS (Debian 13, port 8007, datastore ZFS pool). Ils passent par un réseau dédié séparé (VLAN 20). PBS est physiquement séparé du cluster, idéalement dans un autre datacenter.

L'installation de PBS sur Debian 13 :

Sur la machine Debian 13 qui deviendra le PBS :

```
# Sur la machine dédiée PBS
# 1. La clé qui signe les dépôts Proxmox : absente d'une Debian neuve
wget https://enterprise.proxmox.com/debian/proxmox-archive-keyring-trixie.gpg \
  -O /usr/share/keyrings/proxmox-archive-keyring.gpg
sha256sum /usr/share/keyrings/proxmox-archive-keyring.gpg   # à comparer à la valeur de la documentation

# 2. Le dépôt PBS (format DEB822 pour Debian 13)
nano /etc/apt/sources.list.d/pbs.sources                    # un fichier neuf
```

Le contenu de `pbs.sources`, à taper dans `nano`, puis **Ctrl + O**, Entrée et **Ctrl + X** :

```
Types: deb
URIs: http://download.proxmox.com/debian/pbs
Suites: trixie
Components: pbs-no-subscription
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

Ce support est la propriété de Dawan France.

Sur la machine dédiée PBS :

```
# 3. Installation
apt update                                          # la liste des paquets
apt install -y proxmox-backup-server                # le serveur de sauvegarde
```

Résultat attendu — sur une VM Debian 13.7 neuve (image cloud officielle). Sans l'étape 1, `apt update` refuse le dépôt :

```
Err:7 http://download.proxmox.com/debian/pbs trixie InRelease
  Sub-process /usr/bin/sqv returned an error code (1), error message is: Error:
Failed to parse keyring "/usr/share/keyrings/proxmox-archive-keyring.gpg" [...] No
such file or directory (os error 2)
Error: The repository 'http://download.proxmox.com/debian/pbs trixie InRelease' is
not signed.
```

Avec la clé :

```
136673be77aba35dcce385b28737689ad64fd785a797e57897589aed08db6e45
/usr/share/keyrings/proxmox-archive-keyring.gpg
Get:7 http://download.proxmox.com/debian/pbs trixie InRelease [2785 B]
Get:8 http://download.proxmox.com/debian/pbs trixie/pbs-no-subscription amd64
Packages [216 kB]
[...]
Setting up proxmox-backup-server (4.2.6-1) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/proxmox-backup-
proxy.service' → '/usr/lib/systemd/system/proxmox-backup-proxy.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/proxmox-backup.service'
→ '/usr/lib/systemd/system/proxmox-backup.service'.
```

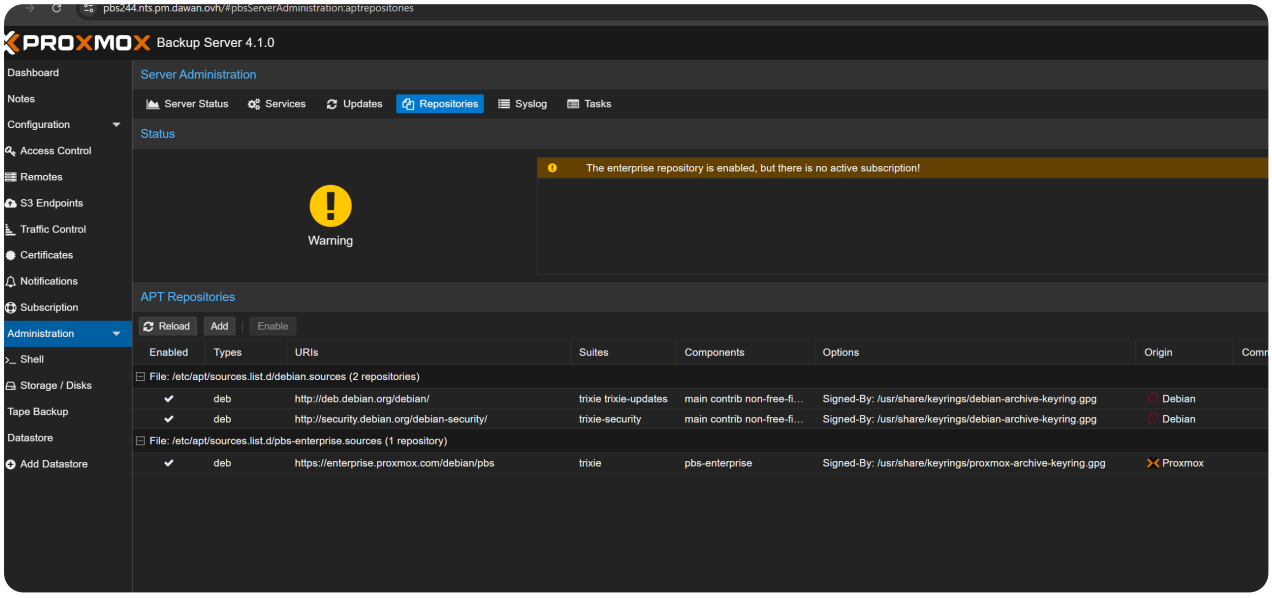
Deux services sont installés, et leur distinction compte pour le diagnostic. `proxmox-backup` fait le travail de sauvegarde ; `proxmox-backup-proxy` sert l'interface web sur le port 8007. Une interface inaccessible alors que les sauvegardes fonctionnent désigne le second.

> **Le format DEB822 — le fichier `.sources` — est celui de Debian 13.** L'ancien format sur une ligne, dans `sources.list.d/*.list`, fonctionne encore mais est déconseillé : il ne permet pas de désigner la clé de signature par dépôt, ce que fait ici `Signed-By`.

Ce support est la propriété de Dawan France.

> `pbs-no-subscription` est le composant sans abonnement, adapté au lab et aux tests. En production, `pbs-enterprise` donne accès aux paquets validés et au support.

> **Si PBS a été installé depuis l'ISO officielle**, il arrive au contraire avec le dépôt *enterprise* actif. Il faut le basculer avant toute mise à jour : voir la capture ci-dessous.



L'onglet `Administration → Repositories` d'un PBS sortant d'installation. Le bandeau jaune ne laisse aucun doute : *« The enterprise repository is enabled, but there is no active subscription! »*. Trois dépôts déclarés, répartis en deux fichiers :

- `/etc/apt/sources.list.d/debian.sources` — les deux dépôts Debian, `trixie` et `trixie-security`. Ils n'ont rien à voir avec l'abonnement et restent en place.
- `/etc/apt/sources.list.d/pbs-enterprise.sources` — le dépôt `pbs-enterprise`, coché, d'origine Proxmox. C'est lui qui déclenche l'avertissement, et c'est lui qu'on désactive.

Regardez la colonne **Components** : elle affiche `pbs-enterprise`, et affichera `pbs-no-subscription` après la bascule. Le nom du composant n'est pas le même que sur Proxmox VE (`pve-enterprise` / `pve-no-subscription`) : c'est le même mécanisme, décliné produit par produit.

`Administration` → **Repositories** → sélectionner la ligne `enterprise.proxmox.com` → **Disable**, puis **Add** → **No-Subscription**. Enfin `Administration` → **Updates** → **Refresh** et **Upgrade**.

État après installation : Sur la machine Debian 13 qui deviendra le PBS :

Ce support est la propriété de Dawan France.

```
# Le service de sauvegarde, puis le port de l'interface web
systemctl status proxmox-backup
ss -tlnp | grep 8007   # PBS doit écouter sur le port 8007
```

Résultat attendu :

```
● proxmox-backup.service - Proxmox Backup API Server
     Loaded: loaded (/usr/lib/systemd/system/proxmox-backup.service; enabled;
preset: enabled)
     Active: active (running) since Tue 2026-09-29 13:56:53 CEST; 2s ago
 Invocation: 4f4e0acc43ac49dd8ff40cc80323f017
   Main PID: 548 (proxmox-backup-)
LISTEN 0      1024               *:8007            *:*    users:(("proxmox-backup-
",pid=617,fd=17),("proxmox-backup-",pid=617,fd=16))
```

Sur `pbs1`, le PBS du lab. `active (running)` et un processus à l'écoute sur le port 8007 : le serveur est prêt. `ss -tlnp` **est la vérification la plus directe** — si rien n'écoute sur 8007, inutile d'essayer l'interface web.

> **L'accès web se fait en HTTPS sur le port 8007**, avec un certificat auto-signé que le navigateur signale comme non fiable. C'est attendu à l'installation ; l'avertissement s'accepte en lab, et se remplace par un certificat Let's Encrypt en production.
>
> Ne confondez pas les ports : **8006 pour Proxmox VE, 8007 pour Proxmox Backup Server.** Ce sont deux machines distinctes ici.
>
> L'identifiant est `root@pam` — le realm `pam` désigne l'authentification par les comptes Linux de la machine. Le mot de passe est donc celui du root système, pas un mot de passe propre à PBS.

Sur le serveur PBS, on crée ensuite un **datastore** — un répertoire local qui héberge les sauvegardes. **C'est un objet côté PBS, et il doit exister avant que Proxmox VE puisse le désigner :** dans la fenêtre d'ajout du stockage côté PVE, le champ **Datastore** propose une liste.

Cette liste est vide tant qu'aucun datastore n'a été créé ici. Par l'interface, c'est le bouton **Add Datastore** du panneau de gauche.

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Creer le datastore "demo-store" dans /mnt/backup-zfs/demo-store
proxmox-backup-manager datastore create demo-store /mnt/backup-zfs/demo-store

# Verifier
proxmox-backup-manager datastore list
# -> Name         Path                          Comment
# -> demo-store   /mnt/backup-zfs/demo-store
```

Résultat attendu — sur `pbs1`, le PBS du lab, avec un datastore d'essai `essai-store` (supprimé ensuite) ; la liste montre aussi les deux datastores des ateliers :

```
Chunkstore create: 1%
[...]
Chunkstore create: 99%
Access time update check successful.
TASK OK

┌──────────────┬──────────────────────────────┬─────────┐
│ name         │ path                         │ comment │
╞══════════════╪══════════════════════════════╪═════════╡
│ backup-store │ /var/lib/backup-store        │         │
├──────────────┼──────────────────────────────┼─────────┤
│ demo-store   │ /mnt/datastore/demo-store    │         │
├──────────────┼──────────────────────────────┼─────────┤
│ essai-store  │ /var/tmp/essai-store         │         │
└──────────────┴──────────────────────────────┴─────────┘
```

La progression du *chunkstore* n'est pas décorative : PBS crée **65 536 sous-répertoires** destinés à répartir les blocs de données. C'est ce qui évite d'avoir des millions de fichiers dans un seul répertoire, situation qu'aucun système de fichiers ne gère bien.

`Access time update check successful` : PBS a vérifié que le système de fichiers met bien à jour la date d'accès des fichiers ; le ramasse-miettes s'en sert pour savoir quels blocs sont encore utilisés. `TASK OK` est le vocabulaire des tâches Proxmox : la même mention apparaît dans l'interface web pour toute opération terminée avec succès.

> **Le répertoire doit exister et être vide, sur un système de fichiers dédié.** Créer un datastore sur la partition racine remplit `/` au premier grand transfert et bloque le serveur.

Dawan
Ce support est la propriété de Dawan France.

ZFS est le socle recommandé, pour ses sommes de contrôle : une sauvegarde silencieusement corrompue est pire que pas de sauvegarde du tout, et c'est exactement ce que ZFS détecte.

ZFS comme système de fichiers pour PBS : ZFS est vivement recommandé pour les datastores PBS. Ses checksums natifs s'ajoutent aux checksums SHA-256 de PBS, créant une double couche de protection contre la corruption silencieuse. La commande `zpool scrub` + les verify jobs PBS offrent une couverture d'intégrité quasi complète.

4.2 Connexion PBS ↔ Proxmox VE

Pour que Proxmox VE puisse sauvegarder vers PBS, il faut l'ajouter comme stockage. Trois informations sont nécessaires avant de commencer.

1 — Le fingerprint TLS de PBS : PBS génère un certificat TLS auto-signé à l'installation. Son empreinte SHA-256 permet à PVE de vérifier l'identité de PBS sans autorité de certification externe.

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
# Sur le serveur PBS
proxmox-backup-manager cert info   # à lire : la ligne Fingerprint et les suivantes
# → Fingerprint (sha256):
# → AA:BB:CC:DD:EE:FF:00:11:22:33:44:55:66:77:88:99:...
```

2 — Un token API PBS : évitez le compte root. Créez un utilisateur et un token dédiés :

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
# Sur PBS : créer un utilisateur de backup
proxmox-backup-manager user create backup@pbs --password 'BackupSecure2026!'

# Créer le token API — son secret ne s'affiche qu'une fois : copiez-le
proxmox-backup-manager user generate-token backup@pbs pve-token

# Accorder les droits sur le datastore AU JETON (un jeton n'hérite pas de son compte)
```

Ce support est la propriété de Dawan France.

```
proxmox-backup-manager acl update /datastore/demo-store DatastoreBackup --auth-id
'backup@pbs!pve-token'
```

Résultat attendu de `generate-token` :

```
Result: {
  "tokenid": "backup@pbs!pve-token",
  "value": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

Dans PBS, un jeton a toujours **ses propres droits** : l'ACL doit viser `backup@pbs!pve-token`, et non le compte `backup@pbs`. Oublier ce détail donne un stockage déclaré côté PVE, mais des sauvegardes refusées avec `permission check failed`. Le rôle `DatastoreBackup` autorise à créer des sauvegardes et à lire les siennes, sans pouvoir effacer celles des autres.

3 — Ajouter dans Proxmox VE :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Déclarer le serveur PBS comme stockage de sauvegarde du cluster.
pvesm add pbs pbs-backup \
  --server <ip-du-serveur-pbs> \
  --datastore demo-store \
  --username 'backup@pbs!pve-token' \
  --password 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx' \
  --fingerprint 'AA:BB:CC:DD:...'
# --username : la forme utilisateur@realm!nom-du-jeton, entre apostrophes :
#   sans elles, bash interprète le « ! » comme un rappel d'historique.
# --password : pour un jeton, c'est son secret, affiché UNE SEULE FOIS à sa
création.
# --fingerprint : empreinte du certificat auto-signé de PBS. Sans elle, la
#   connexion est refusée ; on la relève sur le serveur PBS par
#   « proxmox-backup-manager cert info | grep Fingerprint ».
```

Résultat attendu :

```
Name             Type     Status     Total (KiB)      Used (KiB) Available (KiB)
%
```

Ce support est la propriété de Dawan France.

```
ceph-vm             rbd     active        68741579         1617667        67123912
2.35%
local               dir     active        14210064         9104968         4361468
64.07%
local-lvm       lvmthin     active       136179712        10009208       126170503
7.35%
pbs-backup          pbs     active       102625188         2529284        94836696
2.46%
zfs-vms         zfspool     active        50298076             160        50297916
0.00%
```

Sur `pxe211` au chapitre 5 (le Ceph et le ZFS des chapitres précédents sont là aussi). Le statut `active` confirme trois choses d'un coup : **le serveur PBS est joignable, l'empreinte du certificat a été acceptée, et les identifiants sont bons.**

Les tailles sont en kibioctets, comme partout dans `pvesm`. La colonne `Used` reflète l'occupation **après déduplication** — c'est-à-dire l'espace réellement consommé sur le datastore, non le volume des sauvegardes.

> **Un statut `inactive` juste après l'ajout vient presque toujours de l'empreinte du certificat.** PBS présente un certificat auto-signé. Proxmox VE refuse la connexion tant que son empreinte n'a pas été enregistrée avec le stockage : champ *Fingerprint* dans l'interface, ou `--fingerprint` en ligne de commande.
>
> L'empreinte se relève sur le serveur PBS par `proxmox-backup-manager cert info | grep Fingerprint`.

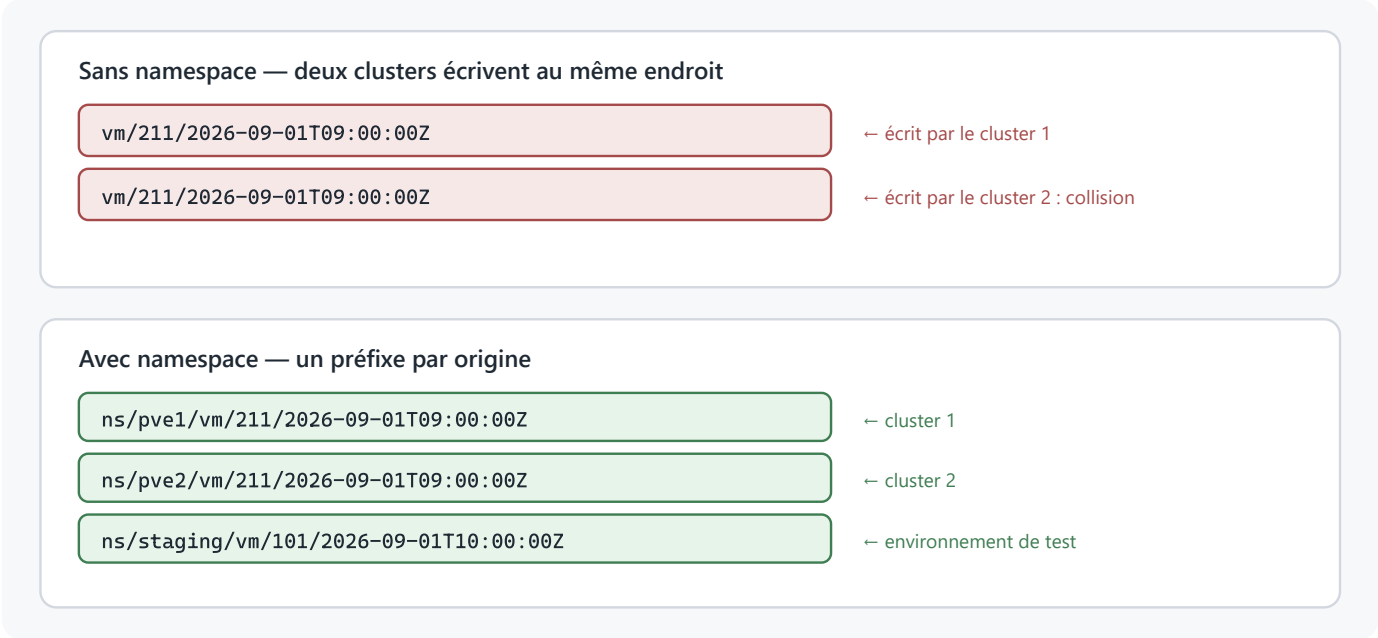
Le stockage apparaît également dans Datacenter → Storage avec le type "Proxmox Backup Server".

> **▶ À pratiquer maintenant — Atelier 2**
>
> Déclarer le serveur de sauvegarde auprès de Proxmox et vérifier son empreinte : c'est le préalable de tout le reste du chapitre.

4.3 Les namespaces PBS

Ce support est la propriété de Dawan France.

Un **namespace** est un dossier logique à l'intérieur d'un datastore. Il sert quand plusieurs clusters sauvegardent dans le même datastore : chacun a sa VM 101, et sans namespace leurs sauvegardes `vm/101` se mélangent. Avec un namespace par cluster (`pve1` , `pve2`), chacun range les siennes à part.



Deux clusters ont chacun leur VM 211 : sans namespace, leurs sauvegardes portent le même chemin `vm/211/…` et se mélangent. Avec un namespace par cluster, le préfixe `ns/<nom>/` les sépare.

Dans PBS : **Datastore** → votre datastore → **Content** → **Add Namespace**. Dans Proxmox VE, le stockage PBS indique ensuite dans quel namespace écrire (champ **Namespace**). Un utilisateur PBS peut être limité à son seul namespace.

On les crée **avant la première sauvegarde** : un groupe de sauvegardes déjà écrit à la racine y reste. C'est aussi là qu'on range la copie d'un autre PBS (§ 6.1) : un namespace par source, et rien ne se mélange.

> **▶ À pratiquer maintenant — Atelier 3**
>
> Un namespace par nœud dans le même datastore, et un utilisateur PBS qui ne voit que le sien.

À retenir : PBS s'installe **hors** de l'infrastructure qu'il protège, avec un datastore sur un système de fichiers dédié. On le déclare dans Proxmox VE comme stockage, avec son adresse et un **jeton** dédié plutôt que root.

Ce support est la propriété de Dawan France.

Il ne faut pas oublier l'**empreinte** de son certificat, dont l'absence est la première cause d'un statut `inactive`. Les **namespaces** rangent séparément les sauvegardes de plusieurs sources dans un même datastore : on les crée avant la première sauvegarde.

Ce support est la propriété de Dawan France.

5. PBS : sauvegarder, restaurer, vérifier

PBS est connecté. On en vient à ce qui compte : planifier les sauvegardes et leur rétention, restaurer une machine entière ou un simple fichier. Et s'assurer, par des vérifications régulières, que les sauvegardes sont réellement restaurables.

5.1 Jobs de backup et politique de rétention

Une fois PBS déclaré comme stockage, les backups se planifient dans **Datacenter** → **Backup** → **Add**. On y définit le planning (expression CRON), les VMs concernées, le mode et la politique de rétention.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Déclencher un backup manuel (mode snapshot = sans arrêter la VM)
# Pas de --compress : PBS compresse lui-même chaque bloc en zstd
vzdump 101 --storage pbs-backup --mode snapshot

# Backup de toutes les VMs et conteneurs d'un nœud
vzdump --all --storage pbs-backup --mode snapshot
```

Résultat attendu :

```
INFO: starting new backup job: vzdump 101 --mode snapshot --storage pbs-backup
INFO: Starting Backup of VM 101 (qemu)
INFO: Backup started at 2026-09-29 13:59:55
INFO: status = running
INFO: VM Name: vm-web-01
INFO: include disk 'scsi0' 'local-lvm:vm-101-disk-0' 20G
INFO: backup mode: snapshot
INFO: ionice priority: 7
INFO: snapshots found (not included into backup)
INFO: creating Proxmox Backup Server archive 'vm/101/2026-09-29T11:59:55Z'
INFO: issuing guest-agent 'fs-freeze' command
INFO: starting backup via QMP command
INFO: issuing guest-agent 'fs-thaw' command
INFO: started backup task '1acbee3f-34f1-47d0-8a08-e31e8f84c121'
```

Ce support est la propriété de Dawan France.

```
INFO: resuming VM again
INFO: scsi0: dirty-bitmap status: created new
INFO:  63% (12.7 GiB of 20.0 GiB) in 3s, read: 4.2 GiB/s, write: 97.3 MiB/s
INFO: 100% (20.0 GiB of 20.0 GiB) in 5s, read: 3.7 GiB/s, write: 0 B/s
INFO: backup is sparse: 18.57 GiB (92%) total zero data
INFO: backup was done incrementally, reused 19.71 GiB (98%)
INFO: transferred 20.00 GiB in 5 seconds (4.0 GiB/s)
INFO: Finished Backup of VM 101 (00:00:06)
INFO: Backup finished at 2026-09-29 14:00:01
INFO: Backup job finished successfully
INFO: notified via target `mail-to-root`
```

La même commande, relancée juste après :

```
INFO: scsi0: dirty-bitmap status: OK (24.0 MiB of 20.0 GiB dirty)
INFO: using fast incremental mode (dirty-bitmap), 24.0 MiB dirty of 20.0 GiB total
INFO: backup was done incrementally, reused 19.98 GiB (99%)
INFO: transferred 24.00 MiB in 1 seconds (24.0 MiB/s)
INFO: Finished Backup of VM 101 (00:00:01)
```

`dirty-bitmap status: created new` **annonce la première sauvegarde depuis le démarrage de la VM** : QEMU n'a pas encore de carte des blocs modifiés, tout est relu. À la sauvegarde suivante, la mention devient `OK (24.0 MiB of 20.0 GiB dirty)` : seuls les 24 Mo modifiés sont lus et envoyés, en une seconde.

`reused 19.71 GiB (98%)` **dès la première sauvegarde** : le datastore contenait déjà ces blocs, venus des sauvegardes précédentes de la même VM. C'est la déduplication de PBS : un bloc déjà stocké n'est jamais renvoyé.

`backup is sparse` mesure les blocs vides du disque virtuel, ignorés plutôt que transférés.

> **Le mode `snapshot` sauvegarde sans arrêter la VM**, mais l'état obtenu est celui d'un arrêt brutal du point de vue de l'invité. Les bases de données peuvent en souffrir. Le **QEMU Guest Agent**, installé dans la VM, permet à Proxmox de figer les systèmes de fichiers pendant l'instantané : la sauvegarde devient cohérente.

Dawan
Ce support est la propriété de Dawan France.

> Les deux autres modes ont leur place : `suspend` met en pause pendant la copie, `stop` arrête la VM. Le premier est un compromis rarement satisfaisant, le second la seule garantie absolue.

La **politique de rétention (pruning)** évite que les sauvegardes n'occupent un espace croissant indéfiniment. PBS utilise un système à niveaux hiérarchiques :

Paramètre	Signification
`keep-last=7`	Conserver les 7 dernières sauvegardes, quelle que soit leur date
`keep-daily=30`	Conserver 1 sauvegarde par jour sur les 30 derniers jours
`keep-weekly=12`	Conserver 1 sauvegarde par semaine sur les 12 dernières semaines
`keep-monthly=6`	Conserver 1 sauvegarde par mois sur les 6 derniers mois

Après le pruning, les chunks qui ne sont plus référencés par aucune sauvegarde restent sur disque jusqu'au **garbage collection** (GC : le nettoyage des chunks orphelins) :

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
# Sur le serveur PBS — forcer le GC manuellement ; il affiche son journal en direct
proxmox-backup-manager garbage-collection start demo-store
```

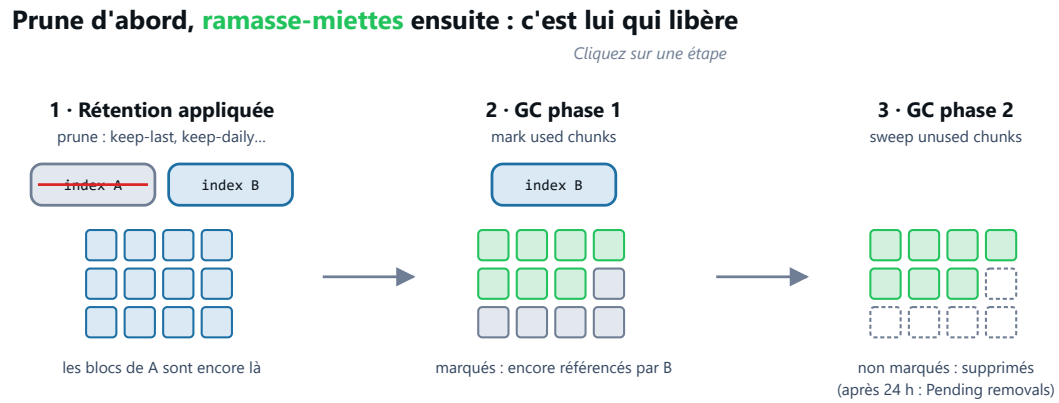
Résultat attendu (extrait du journal) :

```
starting garbage collection on store demo-store
Start GC phase1 (mark used chunks)
marked 100% (12 of 12 index files)
Start GC phase2 (sweep unused chunks)
Removed garbage: 12.30 GiB
Removed chunks: 4821
Pending removals: 0 B (in 0 chunks)
Original data usage: 78.40 GiB
On-Disk usage: 12.20 GiB (15.56%)
Deduplication factor: 6.43
TASK OK
```

Les deux phases sont le mécanisme lui-même. La **phase 1** parcourt tous les index de sauvegarde et **marque** chaque bloc encore utilisé. La **phase 2** balaie le datastore et supprime les blocs non

Dawan Ce support est la propriété de Dawan France.

marqués.



Une sauvegarde PBS = un index qui pointe vers des blocs (chunks) partagés avec les autres sauvegardes.
Supprimer une sauvegarde ne libère rien tout de suite : seul le ramasse-miettes (GC) rend l'espace.

Pourquoi l'espace ne revient qu'après le ramasse-miettes. Le prune ne retire que des index ; la phase 1 marque les blocs encore utilisés, la phase 2 supprime les autres. Cliquez sur une étape pour la ligne de journal correspondante.

Supprimer une sauvegarde ne libère rien tout de suite. PBS retire l'index, mais les blocs restent sur le disque tant qu'un autre index peut encore y faire référence. C'est le ramasse-miettes qui vérifie et libère.

`Pending removals` mérite d'être compris : il compte les blocs devenus orphelins mais **conservés pendant 24 h par sécurité**, au cas où une sauvegarde en cours viendrait à les référencer. Une valeur non nulle est donc normale ; elle passera à zéro au passage suivant.

Un datastore qui se remplit alors que la rétention semble respectée est presque toujours un ramasse-miettes qui ne tourne pas. Vérifiez qu'il est planifié : *Datastore → Prune & GC → GC Schedule*.

L'interface propose un planning quotidien à la création. Mais un datastore créé en ligne de commande, ou repris d'une ancienne installation, peut en être dépourvu.

L'ordre compte : **prune d'abord, GC ensuite.** Le prune retire les index selon la politique de rétention, le GC libère les blocs devenus inutiles. L'inverse ne libère rien.

Ce support est la propriété de Dawan France.

> **Séquence à comprendre :** pruning → GC → espace libéré. Le pruning supprime les *références* aux vieilles sauvegardes dans les métadonnées de PBS.
>
> Le GC supprime ensuite les *chunks* qui ne sont plus référencés. Ces deux étapes peuvent être planifiées automatiquement dans l'interface PBS (Datastore → Prune & GC).

> **▶ À pratiquer maintenant — Atelier 4**
>
> Sauvegarder le CT 201 et la VM du chapitre 4, puis constater ce que la déduplication économise dès la deuxième passe.

5.2 Restauration complète et file-level restore

PBS offre deux niveaux de restauration que vzdump seul ne permettait pas facilement.

Restauration complète d'une VM depuis PVE :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Via l'interface : VM → Backup → sélectionner une sauvegarde → Restore
# Via la CLI :
# (196 : un identifiant libre — la VM 110 existe déjà depuis le chapitre 2)
qmrestore "pbs-backup:backup/vm/101/2026-10-07T13:19:23Z" 196 \
  --storage local-lvm \
  --unique
# --unique génère de nouvelles adresses MAC, pour cohabiter avec l'original.
# L'adresse IP, elle, vient de cloud-init : la restauration reprend celle de la VM 101.
# Avant tout démarrage, lui donner une adresse libre (onglet Cloud-Init, ou qm set --ipconfig0).
```

Résultat attendu :

```
new volume ID is 'local-lvm:vm-196-cloudinit'
new volume ID is 'local-lvm:vm-196-disk-0'
restore proxmox backup image: /usr/bin/pbs-restore --repository
root@pam@10.59.1.14:demo-store --ns pxe211 vm/101/2026-10-07T13:19:23Z drive-
scsi0.img.fidx /dev/pve/vm-196-disk-0 --verbose --format raw --skip-zero
```

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

```
connecting to repository 'root@pam@10.59.1.14:demo-store'
using up to 4 threads
open block backend for target '/dev/pve/vm-196-disk-0'
starting to restore snapshot 'vm/101/2026-10-07T13:19:23Z'
download and verify backup index
fetching up to 16 chunks in parallel
[...]
progress 100% (read 21474836480 bytes, zeroes = 92% (19822280704 bytes), duration 3
sec)
restore image complete (bytes=21474836480, duration=3.57s, speed=5729.46MB/s)
rescan volumes...
```

Sur le lab (PBS en `10.59.1.14`), vers un identifiant d'essai (196, supprimé ensuite) ; les lignes d'avertissement de LVM sur le provisionnement fin sont omises. Le stockage du lab se connecte en `root@pam` et range les sauvegardes dans le namespace `pxe211` (`--ns pxe211`, § 4.3) ; avec un jeton, le dépôt s'écrit `utilisateur@realm!jeton@serveur:datastore`.

La commande sous-jacente, `pbs-restore`, est visible en deuxième ligne : c'est elle qui lit les blocs sur PBS et les écrit dans le nouveau volume. `--skip-zero` et le compteur `zeroes = 92%` montrent que les zones vides du disque ne sont pas écrites — c'est ce qui rend la restauration plus rapide que la sauvegarde initiale.

Restauration de fichier individuel (file-level restore) : c'est la fonctionnalité qui justifie PBS dans de nombreuses infrastructures. Récupérer un fichier `/etc/nginx/nginx.conf` accidentellement supprimé sans avoir à restaurer toute la VM.

La voie la plus simple est l'interface : sélectionnez la sauvegarde (VM → **Backup**, ou le stockage `pbs-backup` → **Backups**) → **File Restore**. Proxmox ouvre une arborescence navigable du disque sauvegardé, et un bouton **Download** récupère le fichier ou le dossier voulu.

En coulisses, il démarre une toute petite machine virtuelle de lecture, qui monte le disque sauvegardé **sans jamais toucher au nœud**. C'est vrai même si le système de fichiers de la sauvegarde est corrompu, ou d'un type que le nœud ne connaît pas.

En ligne de commande, le principe est le même, mais en deux temps. Une sauvegarde de VM est une **image de disque**, pas une arborescence de fichiers. On la présente d'abord comme un disque local, puis on monte sa partition.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Ce support est la propriété de Dawan France.

```
# Définir les variables de connexion (le secret du jeton va dans PBS_PASSWORD)
export PBS_REPOSITORY="backup@pbs!pve-token@<ip-du-serveur-pbs>:demo-store"

# Lister les sauvegardes disponibles
proxmox-backup-client snapshot list --repository "$PBS_REPOSITORY"

# Présenter l'image disque sauvegardée comme un périphérique bloc local (lecture seule)
proxmox-backup-client map "vm/101/2026-10-07T13:19:23Z" drive-scsi0.img --repository "$PBS_REPOSITORY"

# Repérer la partition qui porte le système
lsblk /dev/loop0

# Monter sa première partition, en lecture seule
mkdir -p /mnt/restore-tmp
mount -o ro /dev/loop0p1 /mnt/restore-tmp   # montage en lecture seule

# Explorer puis copier le fichier souhaité
ls /mnt/restore-tmp/etc/ssh/
cp /mnt/restore-tmp/etc/ssh/sshd_config ./sshd_config.restaure   # copier le fichier hors de la sauvegarde

# Démonter, puis libérer le périphérique
umount /mnt/restore-tmp
proxmox-backup-client unmap /dev/loop0   # libérer le périphérique
```

Résultat attendu, sur le lab — sur `pxe211`, (dépôt du lab : `root@pam@10.59.1.14:demo-store`, avec `--ns pxe211` puisque ses sauvegardes sont rangées dans ce namespace, § 4.3). `snapshot list` (extrait) :

```
┌──────────────────────────────┬─────────────┬──────────────────────────────────────────────────────────────────┐
│ snapshot                     │        size │ files                                                            │
╞══════════════════════════════╪═════════════╪══════════════════════════════════════════════════════════════════╡
│ ct/200/2026-10-07T13:19:45Z │ 663.231 MiB │ catalog.pcat1 client.log index.json pct.conf root.pxar           │
```

Ce support est la propriété de Dawan France.

```
├──────────────────────────┼────────────┼──────────────────────────────────────
──────────────┤
│ ct/201/2026-10-07T13:19:30Z │ 524.485 MiB │ catalog.pcat1 client.log index.json
pct.conf root.pxar │
```

Sortie de `map`, puis de `lsblk /dev/loop0` :

```
Image 'root@pam@10.59.1.14:8007:demo-store:vm/101/2026-10-07T13:19:23Z/drive-
scsi0.img' mapped on /dev/loop0
NAME       MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0        7:0    0   20G  1 loop
├─loop0p1  259:0    0 19.9G  1 part
├─loop0p14 259:1    0    3M  1 part
└─loop0p15 259:2    0  124M  1 part
```

La colonne `RO` à `1` : le disque présenté est en lecture seule. `loop0p1` (19,9 Go) porte le système ; les deux petites partitions sont celles du démarrage de l'image cloud. Après `mount`, `ls /mnt/restore-tmp/etc/ssh/` liste `moduli`, `ssh_config`, `ssh_config.d`, `sshd_config`…

`map` rend un périphérique `/dev/loop0` dont les partitions apparaissent comme `/dev/loop0p1`, `/dev/loop0p2`… — `lsblk /dev/loop0` les liste si l'on ne sait pas laquelle porte `/`. Le montage en lecture seule (`-o ro`) n'est pas une précaution de style : la sauvegarde est immuable, et le système refuserait de toute façon d'y écrire. Pour une sauvegarde de **conteneur**, archive de fichiers et non image disque, `proxmox-backup-client mount` monte directement l'arborescence, sans passer par `map`.

> **Prérequis de l'interface :** le bouton **File Restore** a besoin du paquet `proxmox-backup-file-restore` sur le nœud PVE. Il est installé d'office sur les versions récentes ; s'il manque, `apt install proxmox-backup-file-restore`.

> **▶ À pratiquer maintenant — Atelier 5**
>
> Restaurer sous un autre identifiant, puis récupérer un seul fichier sans toucher à la machine d'origine — c'est ce second geste qui sert le plus souvent en production.

5.3 Vérification d'intégrité — verify jobs

Ce support est la propriété de Dawan France.

Avoir une sauvegarde ne suffit pas — encore faut-il qu'elle soit restaurable. La corruption silencieuse (bitrot), les pannes de stockage partielles, ou même des bugs logiciels peuvent rendre une sauvegarde inutilisable sans déclencher d'alerte visible.

Un **verify job** PBS relit tous les chunks d'un ensemble de sauvegardes, recalcule leur SHA-256 et compare avec les métadonnées stockées. Si un chunk est corrompu ou manquant, PBS génère une alerte.

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
# Sur PBS : planifier une vérification hebdomadaire du datastore
proxmox-backup-manager verify-job create verify-weekly \
  --store demo-store \
  --schedule "sun 03:00"
proxmox-backup-manager verify-job list   # vérifier que le job est enregistré

# Vérifier tout de suite le datastore, en sautant ce qui l'a déjà été
proxmox-backup-manager verify demo-store --ignore-verified true
```

Résultat attendu :

```
┌──────────────┬────────────┬────────────┬──────────────────┬─────────────────┬─────────┐
│ id           │ store      │ schedule   │ ignore-verified  │ outdated-after  │ comment │
╞══════════════╪════════════╪════════════╪══════════════════╪═════════════════╪═════════╡
│ essai-verif  │ demo-store │ sun 03:00  │                  │                 │         │
└──────────────┴────────────┴────────────┴──────────────────┴─────────────────┴─────────┘


verify datastore demo-store
found 19 groups
using 1 read and 4 verify thread(s)
verify group demo-store:ct/200 (1 snapshots)
verify demo-store:ct/200/2026-09-25T15:30:13Z
  check pct.conf.blob
  check root.pxar.didx
```

Ce support est la propriété de Dawan France.

```
  verified 225.69/659.51 MiB in 0.69 seconds, speed 327.37/956.61 MiB/s (0 errors)
[...]
percentage done: 100.00% (19/19 groups)
TASK OK
```

Sur `pbs1`, avec une tâche d'essai nommée `essai-verif` (supprimée ensuite) ; le datastore des ateliers contient 19 groupes de sauvegardes.

`(0 errors)` et `TASK OK` signifient que chaque bloc a été relu et comparé à sa somme de contrôle. Les deux nombres de `verified` se lisent ainsi, pour le conteneur 200. 225,69 Mio réellement lus sur le disque de PBS : les blocs stockés, compressés. Pour 659,51 Mio de données une fois décompressées.

Une sauvegarde vérifiée porte ensuite une coche verte dans l'onglet **Content** du datastore, et un bloc corrompu fait marquer la sauvegarde en échec — à refaire. C'est la seule preuve qu'une sauvegarde est restaurable ; le fait qu'elle apparaisse dans la liste ne prouve rien.

Une sauvegarde jamais vérifiée est une sauvegarde dont on ignore l'état. La corruption silencieuse existe : un bloc altéré sur le disque ne se signale pas, et ne se découvre qu'au moment de la restauration — c'est-à-dire au pire moment.

La vérification est coûteuse en entrées-sorties : d'où la planification de nuit, le dimanche, hors des fenêtres de sauvegarde. `--ignore-verified` permet de ne revérifier que ce qui ne l'a jamais été, ce qui rend l'opération beaucoup plus légère sur un gros datastore.

Bonne pratique : testez régulièrement la restauration. Par exemple une fois par trimestre, restaurez une VM de test depuis PBS et vérifiez qu'elle démarre correctement. La vérification automatique des chunks (verify job) contrôle l'intégrité des données, mais seul un test de restauration réel valide le processus complet.

La rétention de vos sauvegardes est réglée à 7 jours, les anciennes sauvegardes disparaissent bien de la liste, et pourtant le datastore PBS continue de se remplir. Pourquoi ?

Ce support est la propriété de Dawan France.

▼ Réponse

Le ramasse-miettes ne tourne pas. Le *pruning* retire les références aux vieilles sauvegardes, mais les chunks restent sur le disque tant que le **garbage collection** ne les a pas supprimés. La séquence est toujours : pruning, puis GC, puis espace libéré. On vérifie donc qu'une tâche GC est planifiée sur le datastore et qu'elle aboutit.

À retenir : un **job** planifie les sauvegardes, la **rétention** (`keep-daily`, `keep-weekly`…) décide lesquelles garder ; le **ramasse-miettes** (GC) libère réellement l'espace. On restaure une VM entière ou un seul fichier (**File Restore**). Un **verify job** relit les chunks : une sauvegarde jamais vérifiée est une sauvegarde dont on ignore l'état.

Ce support est la propriété de Dawan France.

6. PBS : réplication 3-2-1

Un seul PBS reste un point de défaillance unique. On garde donc une copie des sauvegardes ailleurs : c'est la réplication entre PBS.

6.1 Réplication PBS inter-sites — règle 3-2-1

La règle fondamentale du backup est la **règle 3-2-1** : 3 copies des données, sur 2 supports différents, dont 1 hors site. Un seul PBS dans le datacenter primaire ne respecte que les deux premiers critères. Un sinistre (incendie, coupure réseau totale) emporterait à la fois l'infrastructure et les sauvegardes.

PBS résout ce problème avec les **sync jobs** : un PBS peut synchroniser ses datastores vers un PBS distant en ne transférant que les chunks manquants. La déduplication s'applique aussi à la synchronisation — si les deux PBS partagent 80% des données, seuls les 20% manquants transitent sur le réseau.

PBS-Paris (datastore `demo-store`) est la source ; PBS-Lyon (datastore `production-dr`), sur le site de reprise, en garde une copie. Un **sync job** s'exécute **sur le PBS qui reçoit** : il va chercher chez un *remote* les sauvegardes qui lui manquent.

C'est donc sur **Lyon** qu'on le configure, et c'est une force. Le site de reprise tire les données, sans que Paris ait le moindre droit d'écriture ni de suppression chez lui. Un attaquant qui prendrait Paris ne pourrait pas effacer la copie de Lyon.

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
# Sur PBS-Lyon — déclarer PBS-Paris comme remote (le secret est celui d'un jeton de Paris)
proxmox-backup-manager remote create pbs-paris \
  --host <ip-pbs-paris> \
  --auth-id 'sync@pbs!sync-token' \
  --password '<secret-du-jeton>' \
  --fingerprint '<empreinte-de-pbs-paris>'

# Sur PBS-Lyon — tirer chaque jour à 4 h ce qui manque, depuis demo-store de Paris
proxmox-backup-manager sync-job create sync-dr \
  --store production-dr \
```

Ce support est la propriété de Dawan France.

```
  --remote pbs-paris \
  --remote-store demo-store \
  --schedule "04:00" \
  --remove-vanished false

# Vérifier que le remote et le sync-job sont enregistrés
proxmox-backup-manager remote list
proxmox-backup-manager sync-job list   # puis le sync-job
```

Résultat attendu de `proxmox-backup-manager remote list` :

```
┌────────────┬──────────────────┬──────────────────────┬──────────────────────┐
│ name       │ host             │ auth-id              │ fingerprint          │
╞════════════╪══════════════════╪══════════════════════╪══════════════════════╡
│ pbs-paris  │ <ip-pbs-paris>   │ sync@pbs!sync-token  │ AA:BB:CC:…           │
└────────────┴──────────────────┴──────────────────────┴──────────────────────┘
```

Côté Paris, le jeton `sync@pbs!sync-token` n'a besoin que du rôle `DatastoreReader` sur `demo-store` : il lit, il n'écrit jamais. Depuis PBS 3.3, un sync job peut aussi **pousser** (*push*) depuis la source ; le mode par défaut, qui tire, reste le plus sûr pour un site de reprise.

> **`remove-vanished: false` :** si vous l'activez, PBS-Lyon supprimera les sauvegardes prunées sur Paris, reflétant exactement la même rétention. Désactivé (recommandé), le DR (*Disaster Recovery*, le site de reprise, ici PBS-Lyon) conserve les sauvegardes plus longtemps. Il offre ainsi un filet de sécurité en cas de purge accidentelle sur le PBS primaire.

Le job peut ranger la copie dans un **namespace local** (option `--ns`, champ **Local Namespace**) : un namespace par source, et les sauvegardes de plusieurs PBS ne se mélangent pas.

> **▶ À pratiquer maintenant — Atelier 6**
>
> Répliquer vers un second PBS pour satisfaire la règle 3-2-1 — la seule protection qui tienne quand le site principal disparaît.

À retenir : la règle **3-2-1** : trois copies, deux supports, une hors site. Un **sync job**, lancé par le PBS qui reçoit, copie les sauvegardes d'un PBS distant en ne transférant que les chunks manquants, dans un namespace par source.

Ce support est la propriété de Dawan France.

7. Synthèse — l'écosystème Proxmox au complet

Au terme de ces cinq chapitres, votre vision de l'écosystème Proxmox est complète. Chaque produit a son rôle clairement défini :

Produit	Rôle	Port	Langage	Depuis
Proxmox VE	Hyperviseur — KVM + LXC	8006	Perl + JS (JavaScript)	2008
Proxmox Backup Server	Backups dédupliqués + chiffrés	8007	Rust	2020
Proxmox Datacenter Manager	Gestion multi-clusters	8443	Rust	2025

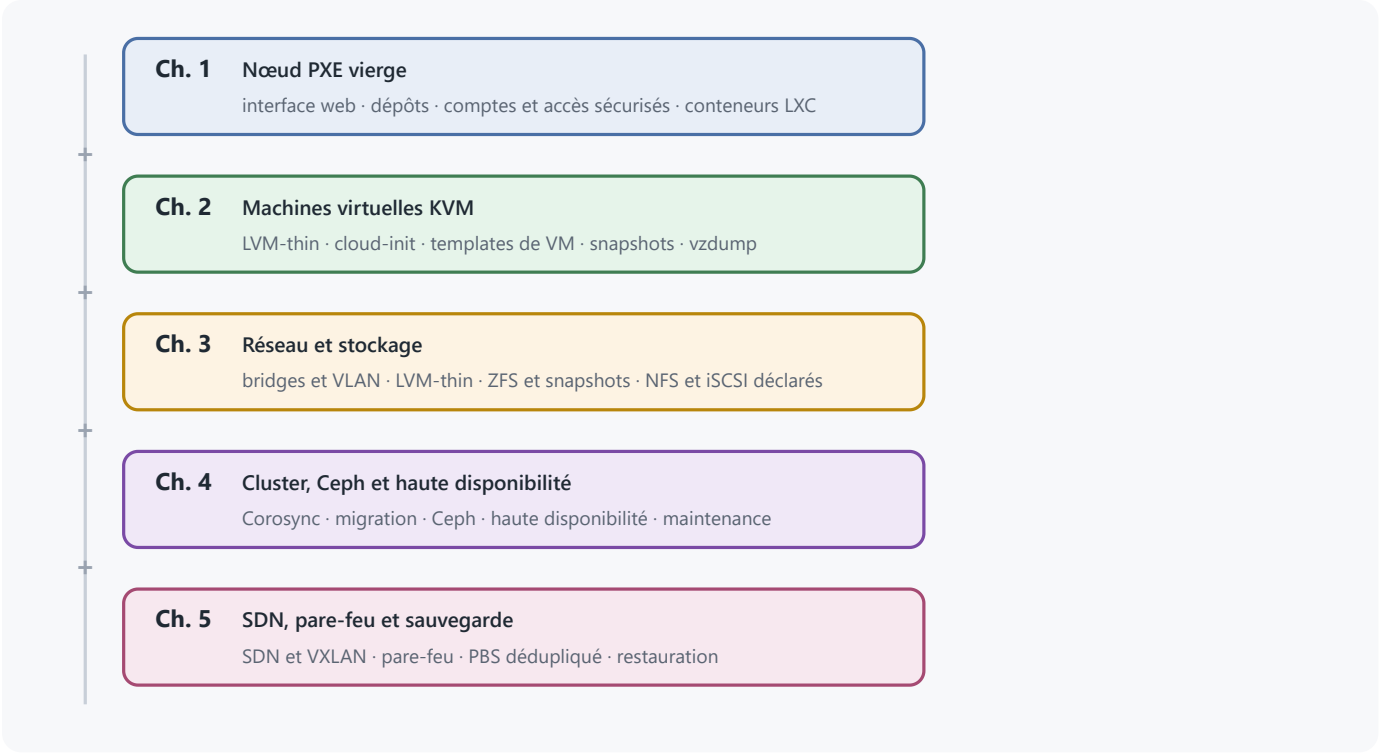
Les technologies de stockage s'articulent autour de leurs points forts respectifs :

Technologie	Usage idéal	Prérequis
LVM-thin	Stockage local par défaut, snapshots rapides	Disques locaux
ZFS	Intégrité maximale, RAIDZ, snapshots	RAM suffisante
Ceph	Stockage partagé distribué pour HA	≥ 3 nœuds, réseau dédié
NFS	Partage fichiers léger (ISOs, templates)	Serveur NFS
PBS	Backups incrémentaux dédupliqués	VM/serveur dédié

La progression de votre infrastructure, chapitre après chapitre

Le schéma suivant récapitule, chapitre par chapitre, l'empilement progressif des briques abordées depuis le Chapitre 1 :

Dawan
Ce support est la propriété de Dawan France.



Félicitations ! Au fil de ces cinq chapitres, vous êtes passés d'un seul hyperviseur installé depuis l'ISO à une infrastructure cluster haute disponibilité. Elle a un stockage Ceph distribué, un réseau SDN, une sauvegarde dédupliquée PBS, et suit les bonnes pratiques de production.

À retenir : Proxmox VE héberge les machines, **Proxmox Backup Server** les sauvegarde, **Proxmox Datacenter Manager** réunit plusieurs clusters dans une seule interface.

Cinq chapitres, de l'installation d'un premier nœud jusqu'à la sauvegarde. Cette formation couvre l'ensemble du cycle de vie d'une infrastructure Proxmox VE. Conteneurs et VM, stockage et réseau SDN, cluster et haute disponibilité, sauvegarde et bonnes pratiques de production.

▶ À pratiquer maintenant — Atelier 7

Le fil rouge de la formation : mettre un serveur web en production avec les outils des cinq chapitres (compte, template, Ceph, HA, pare-feu, PBS). Il s'agit ensuite de passer en revue toute l'infrastructure construite, comme pour la livrer à un client.

Dawan
Ce support est la propriété de Dawan France.

Ateliers — chapitre 5

> **Infrastructure** : le cluster `cluster-tp` du chapitre 4 (3 nœuds, Ceph `HEALTH_OK`). À côté, **votre** Proxmox Backup Server, déjà installé.

Service	URL (navigateur)	Identifiants	Adresse interne
PVE	`https://pxe2<N>1.<ville>.pm.dawan.ovh` (nœuds 2 et 3 : `pxe2<N>2`, `pxe2<N>3`)	`root` / `proxmoxx`, domaine Linux PAM	`10.<site>.<N>.11` à `.13`
PBS	`https://pbs2<N>4.<ville>.pm.dawan.ovh`	`root` / `proxmoxx`, domaine Linux PAM	`10.<site>.<N>.14` — livré sans datastore (atelier 2)

Le formateur vous donne vos URL exactes (`<ville>` : `nts`, `prs` ou `sbg`). Elles sont votre **seul** accès depuis votre poste.

Les adresses internes `10.<site>.<N>.x` ne servent qu'entre vos machines : déclarer PBS dans Proxmox VE.

Les captures proviennent du lab d'exemple (PVE 9.2.20, PBS 4.2.6), dont `vmbr0` était alors en `192.168.1.x`. Les résultats en texte viennent du même lab, passé sur le plan des stagiaires : `10.59.1.x` (site 59, stagiaire 1). Versions : PVE 9.2.21, PBS 4.2.7, PDM (Proxmox Datacenter Manager) 1.1.7.

État du lab en fin de chapitre :

- Pare-feu du Datacenter activé à l'atelier 1 (règles 22, 8006 et Ceph), puis désactivé en fin d'atelier, ses règles conservées
- PBS : datastore `demo-store-s<N>` (atelier 2) ; namespaces `pxe2<N>1` à `pxe2<N>3`, utilisateur `pxe2<N>1-user@pbs` limité au sien, stockage `pbs-backup` dans le namespace `pxe2<N>1` (atelier 3)
- PBS : job nocturne 02:00 avec rétention 7 + 4, facteur de déduplication noté (atelier 4)
- CT 210 restauré depuis PBS, en `.52` ; fichier `/tmp/hostname-restore` extrait (atelier 5)

Dawan
Ce support est la propriété de Dawan France.

- (Atelier 6) copie des PBS voisins dans `backup-store`, un namespace `voisin-s<N>` par voisin, par job de synchronisation
- Fil rouge (atelier 7) : VM 400 `web-prod` en `.90` sur `ceph-vm`, sous HA, filtrée par son pare-feu ; groupe `exploitation` ; pare-feu du Datacenter **réactivé** ; checklist de livraison déroulée, chaque machine décrite

---

Atelier 1 — Activer le pare-feu Proxmox

Contexte

Le pare-feu Proxmox a une règle d'or : **on écrit les règles d'autorisation avant d'activer le pare-feu.** Activer d'abord, c'est risquer de se couper soi-même l'accès à l'interface web.

Sur votre lab, un filet de sécurité existe : Proxmox autorise d'office l'interface web et SSH depuis le **réseau local** du nœud. Or l'accès Dawan arrive par la passerelle de ce réseau.

Vous le vérifierez. Ce filet ne dispense pas du bon ordre : sur un serveur joint depuis un autre réseau, il ne protège rien.

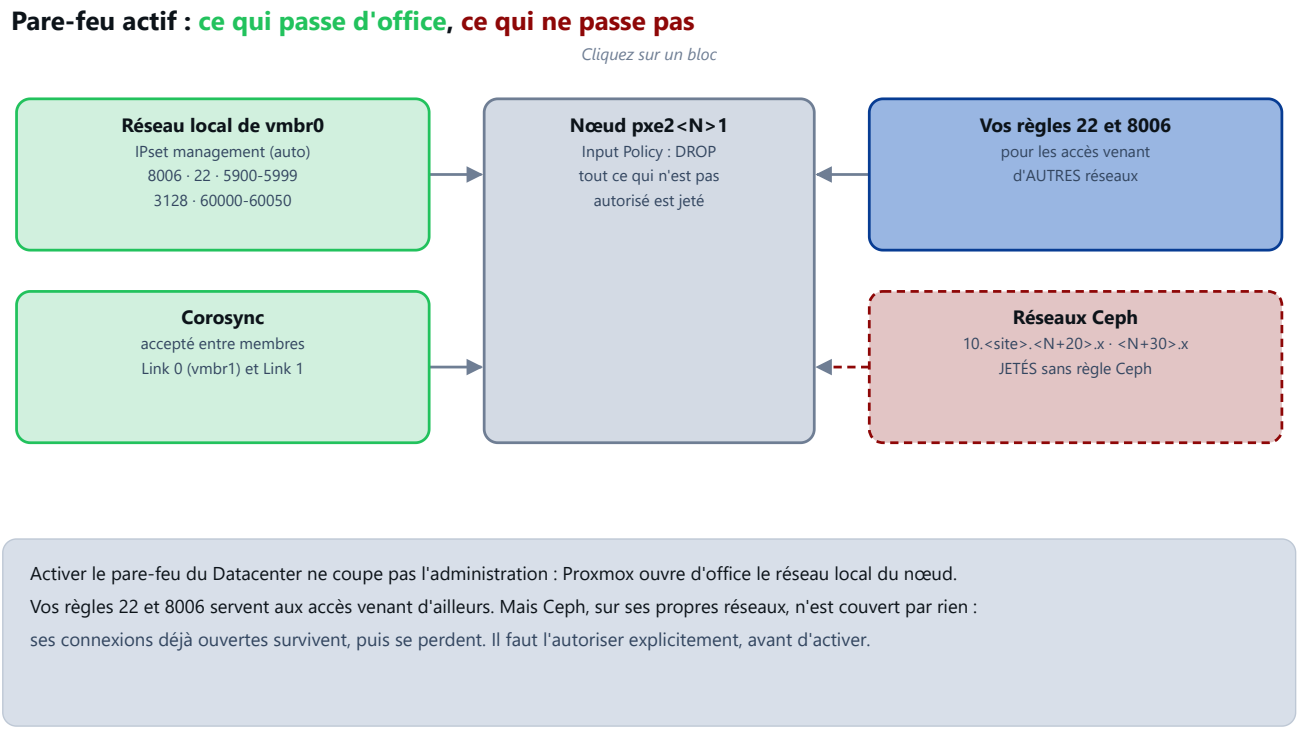
Votre nœud est une machine distante : vous n'y avez **aucun accès physique**. Si le pare-feu vous coupe, personne ne pourra le rattraper depuis un clavier.

Vous allez donc ajouter une seconde précaution, que le cours ne montre pas : un **filet automatique**. Il désactive le pare-feu tout seul au bout de cinq minutes, si vous ne l'annulez pas.

Une dernière précaution concerne Ceph. Le filet de Proxmox ne couvre pas ses réseaux : sans règle, Ceph se bloque quelque temps après l'activation. Ses règles s'écrivent donc **avec** les autres, avant d'activer.

Résultat à obtenir : les règles d'accès (interface web, SSH, Ceph) créées et relues, et un filet automatique posé. **Puis** le pare-feu activé au niveau Datacenter, l'accès vérifié, Ceph toujours en bonne santé, et le filet annulé.

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.



Ce qui passe d'office quand le pare-feu s'active : le réseau local de `vmbr0` et Corosync. Vos règles servent aux autres réseaux.

Et les réseaux de Ceph ne sont couverts par rien. Cliquez sur un bloc pour lire le résultat obtenu.

Arborescence attendue

```
Datacenter → Firewall
├── Règles ACCEPT créées D'ABORD :
│   ├── tcp/8006 (interface web)   ← position 0, la dernière créée
│   ├── tcp/22   (SSH)
│   └── macro Ceph depuis les deux réseaux Ceph   ← sinon Ceph se bloque
└── Options → Firewall : Yes      ← APRÈS les règles
```

Ce que vous devez faire

1. **Créer les règles ACCEPT** au niveau Datacenter, avant toute activation : interface web (8006), SSH (22), et la macro `Ceph` depuis les deux réseaux Ceph.
2. **Relire** ces règles.
3. **Poser un filet automatique** : programmer la désactivation du pare-feu dans cinq minutes (`systemd-run`).

Ce support est la propriété de Dawan France.

4. **Activer le pare-feu** du Datacenter, confirmer que l'accès est conservé et que Ceph reste `HEALTH_OK`, **puis annuler le filet**.
5. **Lire le filet de sécurité de Proxmox** : le réseau qu'il considère comme local.

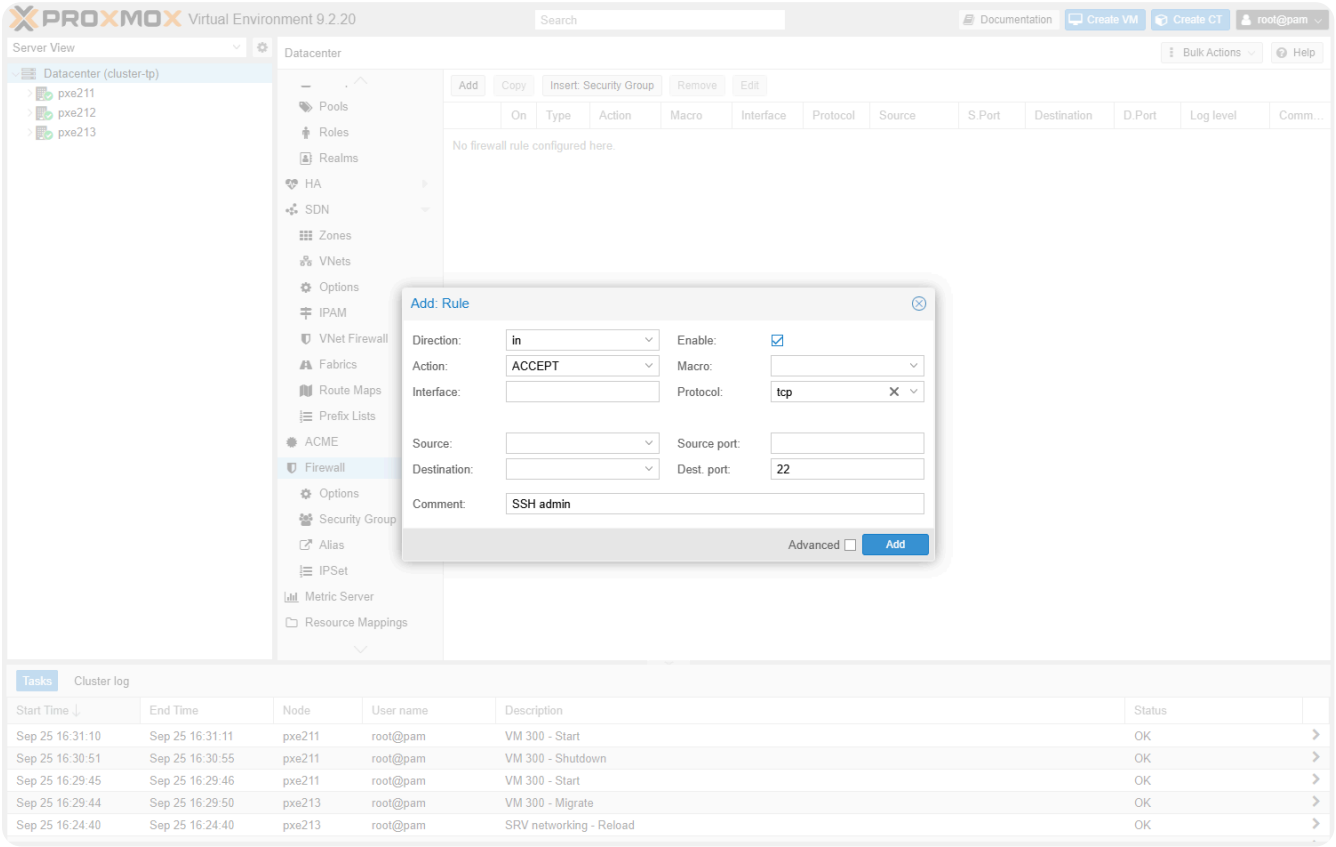
⚠️ L'ordre n'est pas négociable : règles d'abord, activation ensuite.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Créer les règles ACCEPT au niveau Datacenter

Datacenter → **Firewall** (l'entrée seule, pas *VNet Firewall*) → **Add**.



Direction `in`, **Action** `ACCEPT`, **Protocol** `tcp`, **Dest. port** `22`, **Comment** `SSH admin`, **Enable** coché. **Source** vide : la règle vaut pour toute provenance. → **Add**.

Même chose pour l'interface web : **Dest. port** `8006`, **Comment** `Interface web Proxmox VE`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI, dans le Shell d'un nœud
pvesh create /cluster/firewall/rules \
  --action ACCEPT --type in --proto tcp --dport 22 --enable 1 --comment "SSH
admin"
# règle 8006 : l'interface web
pvesh create /cluster/firewall/rules \
  --action ACCEPT --type in --proto tcp --dport 8006 --enable 1 --comment
```

Ce support est la propriété de Dawan France.

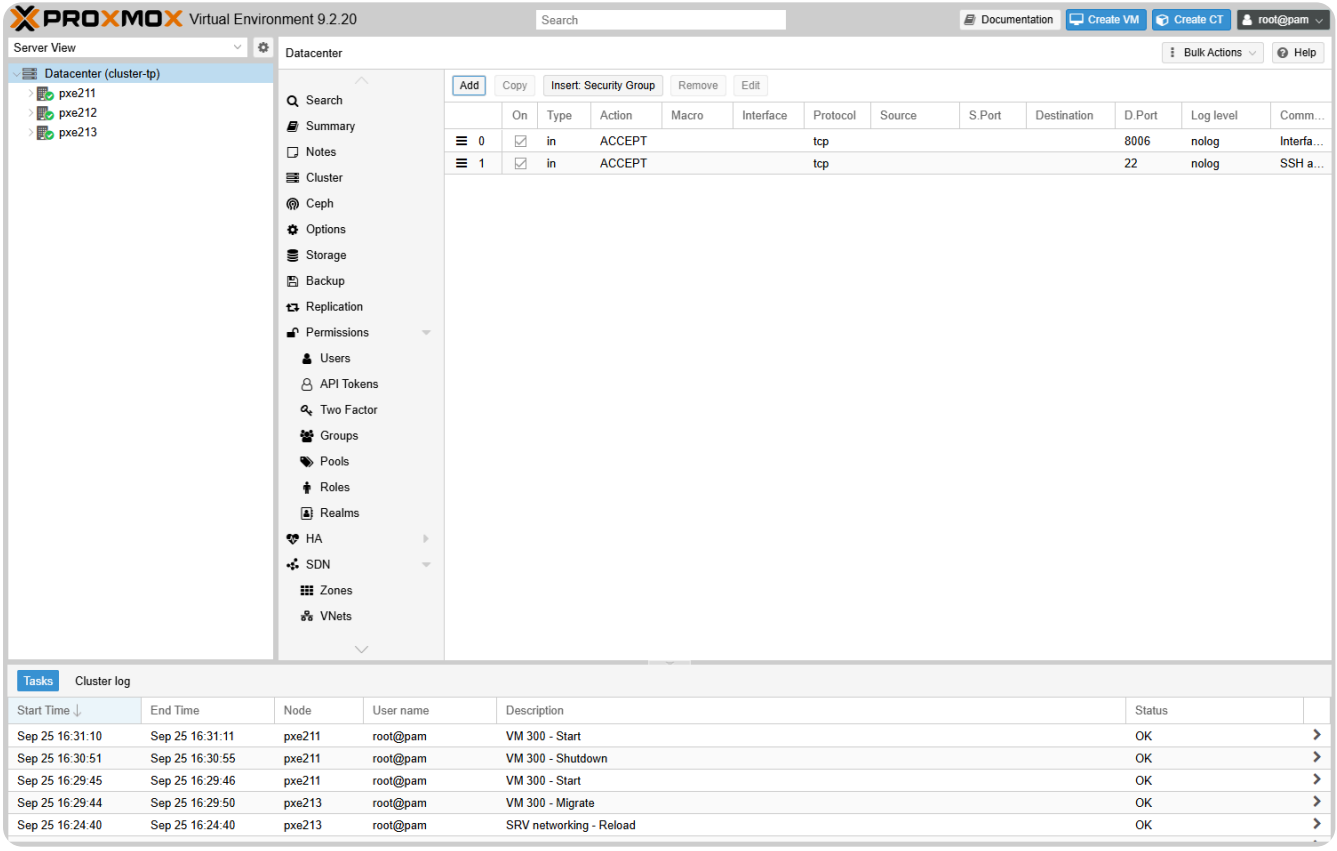
```
"Interface web Proxmox VE"
cat /etc/pve/firewall/cluster.fw   # les règles écrites dans cluster.fw
```

Résultat attendu :

```
[RULES]

IN ACCEPT -p tcp -dport 8006 -log nolog # Interface web Proxmox VE
IN ACCEPT -p tcp -dport 22 -log nolog # SSH admin
```

`-log nolog` est écrit par l'interface ; créées par `pvesh`, les mêmes règles n'ont pas cette mention (`IN ACCEPT -p tcp -dport 8006 # Interface web Proxmox VE`). Le comportement est le même : pas de journalisation.



Deux règles, **8006 en position 0**, 22 en position 1 : chaque nouvelle règle s'insère **en tête**. Les règles se lisent de haut en bas, la première qui correspond décide.

Puis les règles de Ceph. Proxmox fournit une **macro** `Ceph` qui regroupe ses ports.

Ce support est la propriété de Dawan France.

Datacenter → **Firewall** → **Add** : Direction `in`, Action `ACCEPT`, **Macro** `Ceph`, **Source** le réseau Ceph public (`10.<site>.<N+30>.0/24`). Puis une seconde règle identique pour le réseau Ceph cluster (`10.<site>.<N+20>.0/24`).

Dans le Shell du nœud :

```
pvesh create /cluster/firewall/rules --type in --action ACCEPT --macro Ceph \
  --source 10.<site>.<N+30>.0/24 --enable 1 --comment "Ceph depuis 10.<site>.
<N+30>.0/24"   # le réseau Ceph public
pvesh create /cluster/firewall/rules --type in --action ACCEPT --macro Ceph \
  --source 10.<site>.<N+20>.0/24 --enable 1 --comment "Ceph depuis 10.<site>.
<N+20>.0/24"   # le réseau Ceph cluster
grep -E "^IN" /etc/pve/firewall/cluster.fw   # les quatre règles
```

Résultat attendu : quatre lignes, dont `IN Ceph(ACCEPT) -source 10.<site>.<N+20>.0/24` et `IN Ceph(ACCEPT) -source 10.<site>.<N+30>.0/24`, au-dessus des règles 8006 et 22.

> **Pourquoi Ceph avant l'activation.** Le filet de Proxmox (étape 3) couvre le réseau de `vmbr0` et Corosync, **pas** les réseaux de Ceph. Les connexions déjà ouvertes survivent à l'activation ; les nouvelles sont jetées.
>
> Sur le lab, sans ces règles, Ceph a semblé aller bien une vingtaine de minutes. Puis ses démons se sont reconnectés : `HEALTH_WARN`, 8 PG bloqués en `peering`, et une sauvegarde de VM figée 11 minutes. Les deux règles ont tout débloqué en moins de trois minutes.

Ne passez à l'étape 2 qu'après avoir vu vos quatre règles.

Étape 1 bis — Poser un filet automatique

Avant d'activer quoi que ce soit, programmez l'annulation. `systemd-run` crée un minuteur temporaire : dans cinq minutes, il lancera la commande qui désactive le pare-feu du Datacenter.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Dans 5 minutes, désactiver le pare-feu du Datacenter, sauf si on annule avant
systemd-run --on-active=5min --unit=filet-parefeu pvesh set
```

Ce support est la propriété de Dawan France.

```
/cluster/firewall/options --enable 0
# Vérifier que le minuteur est armé
systemctl is-active filet-parefeu.timer
```

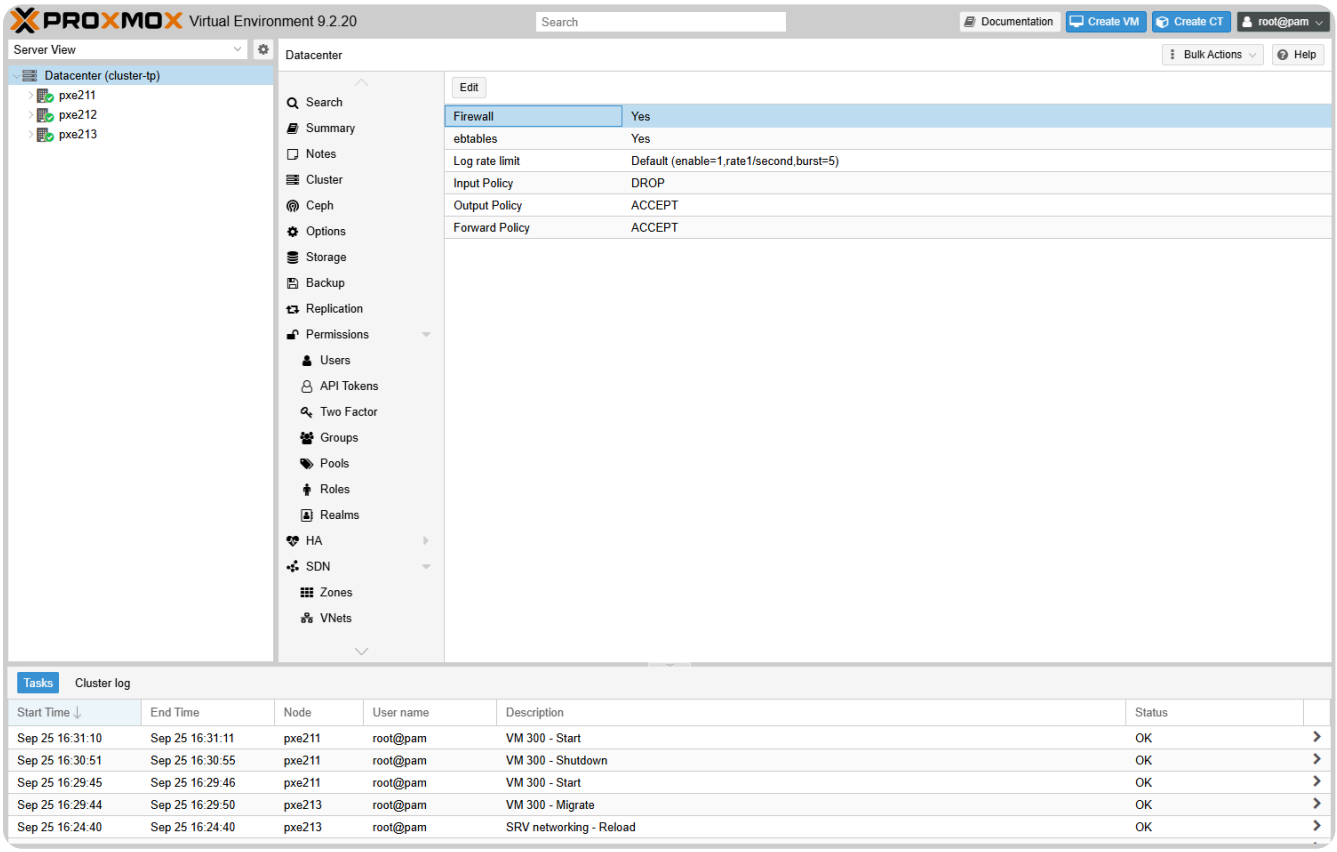
Résultat attendu :

```
Running timer as unit: filet-parefeu.timer
Will run service as unit: filet-parefeu.service
active
```

Si l'activation de l'étape 2 vous coupait l'accès, il suffirait d'attendre : cinq minutes plus tard, le pare-feu repasse à `enable: 0` et l'interface répond de nouveau. Sur le lab, avec un minuteur d'une minute, `pve-firewall status` est bien revenu à `disabled/running`, et `cluster.fw` à `enable: 0`.

Étape 2 — Activer le pare-feu

Datacenter → **Firewall** → **Options** → double-clic sur **Firewall** → cocher → **OK**.



Ce support est la propriété de Dawan France.

Firewall : Yes. **Input Policy : DROP** par défaut : tout ce qu'aucune règle n'autorise est jeté. **Output Policy** et **Forward Policy** restent à `ACCEPT`.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI
pvesh set /cluster/firewall/options --enable 1
pve-firewall status   # état du service de pare-feu
ceph health           # Ceph doit rester en bonne santé
```

Résultat attendu : `Status: enabled/running`, puis `ceph health` toujours à `HEALTH_OK`.

Vérifiez aussitôt, dans un **nouvel** onglet, sans fermer l'ancien : l'interface web se recharge normalement. Sur le lab, `curl -sk -o /dev/null -w "%{http_code}" https://<nœud>:8006/` répondait `200`.

L'accès est bon : annulez le filet, sinon il désactivera le pare-feu dans quelques minutes.

Dans le Shell du nœud :

```
# Arrêter le minuteur : la désactivation programmée n'aura pas lieu
systemctl stop filet-parefeu.timer
# Vérifier qu'il n'est plus armé
systemctl is-active filet-parefeu.timer
```

Résultat attendu : `inactive`.

Si vous ne retrouvez **pas** l'interface dans le nouvel onglet, ne touchez à rien : attendez la fin des cinq minutes, rechargez, puis corrigez vos règles avant de recommencer.

Étape 3 — Le filet de sécurité de Proxmox

Dans le Shell d'un nœud :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pve-firewall localnet   # le réseau que Proxmox considère comme local
```

Résultat attendu (lab `10.59.x`) :

Ce support est la propriété de Dawan France.

```
local hostname: pxe211
local IP address: 10.59.1.11
network auto detect: 10.59.1.0/24
using detected local_network: 10.59.1.0/24

accepting corosync traffic from/to:
 - pxe212: 10.59.11.12 (link: 0)
 - pxe212: 10.59.1.12 (link: 1)
 - pxe213: 10.59.11.13 (link: 0)
 - pxe213: 10.59.1.13 (link: 1)
```

Proxmox détecte le réseau local du nœud. Chez vous, c'est `10.<site>.<N>.0/24`, celui de `vmbr0`. Il y autorise, **sans aucune règle**, ce dont l'administration a besoin : l'interface web (8006), les consoles, SSH (22) et la migration.

Sur le lab, avec **nos règles 22 et 8006 désactivées**, l'interface web répondait toujours `200` depuis ce réseau. Corosync, lui, est accepté entre les membres du cluster, sur les deux liens.

Retenez-le : vos règles 22 et 8006 servent aux accès **hors** du réseau local. Le filet ne couvre que ce réseau-là et Corosync : ni Ceph (d'où l'étape 1), ni rien d'autre.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesh get /cluster/firewall/options --output-format json | grep -o '"enable":1'
# le pare-feu du Datacenter est-il activé ?
curl -sk -o /dev/null -w "%{http_code}\n" https://10.<site>.<N>.11:8006   # l'interface web répond-elle encore ?
ceph health   # Ceph n'a pas souffert
```

Résultat attendu : `"enable":1`, puis `200`, puis `HEALTH_OK`.

Nettoyage — désactiver le pare-feu du Datacenter

En production, il resterait actif. Ici, un cluster filtré n'apporte plus rien aux ateliers suivants (sauvegarde), et une règle oubliée suffirait à les bloquer. Ses règles restent en place : le réactiver les remet en service.

Ce support est la propriété de Dawan France.

Datacenter → **Firewall** → **Options** → **Firewall** → décocher → **OK**.

Dans le Shell du nœud :

```
pvesh set /cluster/firewall/options --enable 0   # désactiver, sans toucher aux règles
pve-firewall status                              # → Status: disabled/running
```

Atelier 2 — Connexion de PBS à Proxmox VE

Contexte

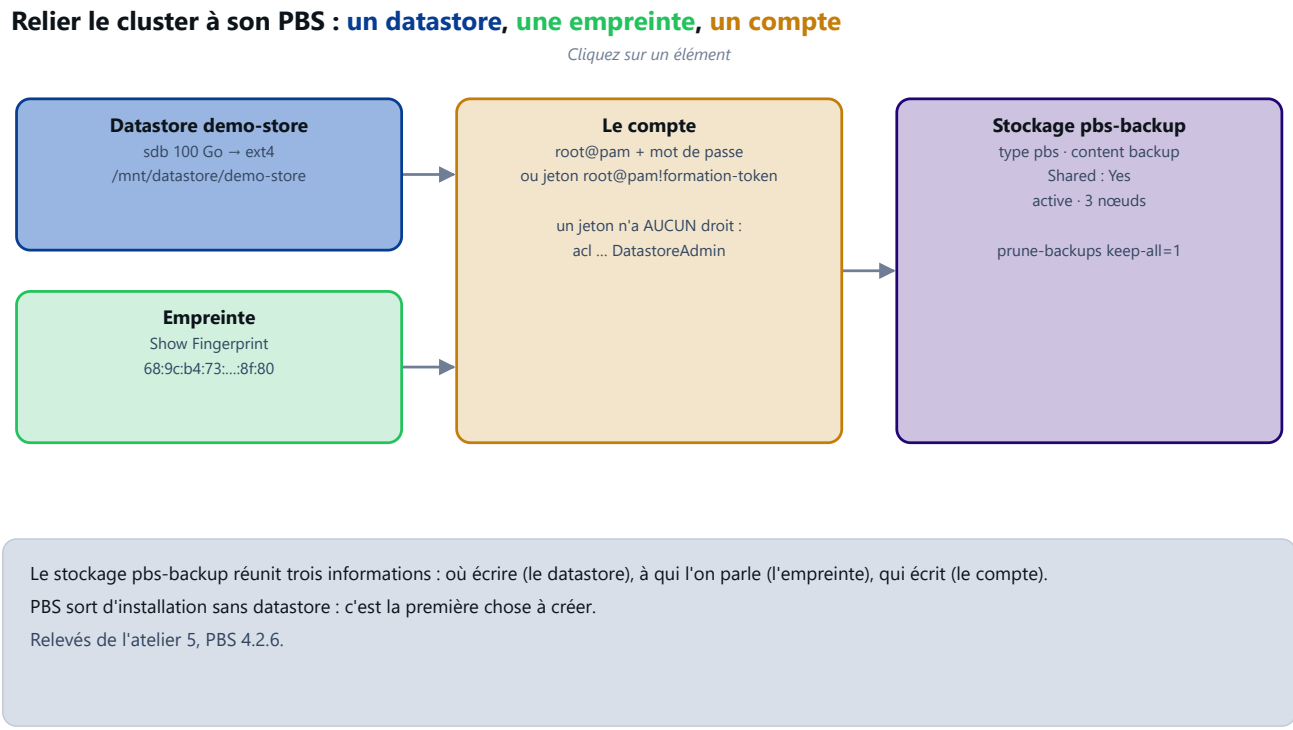
Jusqu'ici, vos sauvegardes restaient sur le nœud qui les produisait : le disque qui tombe emporte la machine **et** sa sauvegarde. Chaque stagiaire dispose de **son propre Proxmox Backup Server**, installé à côté de ses trois nœuds, en `10.<site>.<N>.14`.

Il sort d'installation : pas encore de datastore. Vous allez le préparer, puis le déclarer comme stockage du cluster.

Résultat à obtenir : un datastore `demo-store-s<N>` sur le disque de données de PBS, et un stockage `pbs-backup` actif sur les trois nœuds, qui pointe vers lui.

> **Les ateliers PBS sauvegardent la VM 101 et le CT 201.** La VM 101 vient des chapitres précédents. Créez le CT 201 sur `vmbr0`, avec la commande ci-dessous.
>
> **Dans le Shell du nœud 1** (interface web → votre nœud → **Shell**) :
>
> ```
> # Le CT 201 du fil rouge, sur vmbr0, en .51 (nesting=1 : sinon avertissement systemd)
> pct create 201 local:vztmpl/debian-13-standard_13.6-1_amd64.tar.zst \
>   --hostname ct-sdn-test --storage local-lvm --rootfs local-lvm:4 \
>   --cores 1 --memory 512 --unprivileged 1 --features nesting=1 \
>   --net0 name=eth0,bridge=vmbr0,ip=10.<site>.<N>.51/24,gw=10.<site>.<N>.1
> pct start 201   # démarrer le conteneur
> ```
>
> Si le modèle manque (`pveam list local`), téléchargez-le comme au chapitre 1.

Ce support est la propriété de Dawan France.



Les trois informations que réunit le stockage `pbs-backup`. Cliquez sur un élément pour lire le résultat obtenu.

Arborescence attendue

```
PBS (https://pbs2<N>4.<ville>.pm.dawan.ovh, root / Linux PAM)
├── dépôt pbs-no-subscription      ← à la place du dépôt enterprise
├── disque sdb (100 Go) → Directory ext4 → datastore demo-store
└── jeton root@pam!formation-token + droit DatastoreAdmin sur demo-store
Votre cluster → Datacenter → Storage
└── pbs-backup   (type pbs, active)   ← vu par les trois nœuds
```

Ce que vous devez faire

1. **Se connecter à PBS** et parcourir son interface.
2. **Passer PBS sur le dépôt sans abonnement** et le mettre à jour.
3. **Créer le datastore** `demo-store-s<N>` sur le disque de données (`<N>` = votre numéro de stagiaire).
4. **Relever l'empreinte** du certificat de PBS.
5. **Ajouter PBS comme stockage** du cluster, d'abord par mot de passe, puis par jeton API.

Ce support est la propriété de Dawan France.

Personnalisez le nom de votre datastore : `demo-store-s<N>` (stagiaire 3 : `demo-store-s3`). À l'atelier 6, vos PBS échangent leurs sauvegardes : avec un nom personnel, chacun sait d'où vient chaque copie.

Dans la suite du chapitre, remplacez partout `demo-store` par votre nom. Les sorties du lab gardent `demo-store`.

L'atelier suivant (6) range vos sauvegardes dans des namespaces, avant la toute première : faites-le avant de sauvegarder quoi que ce soit.

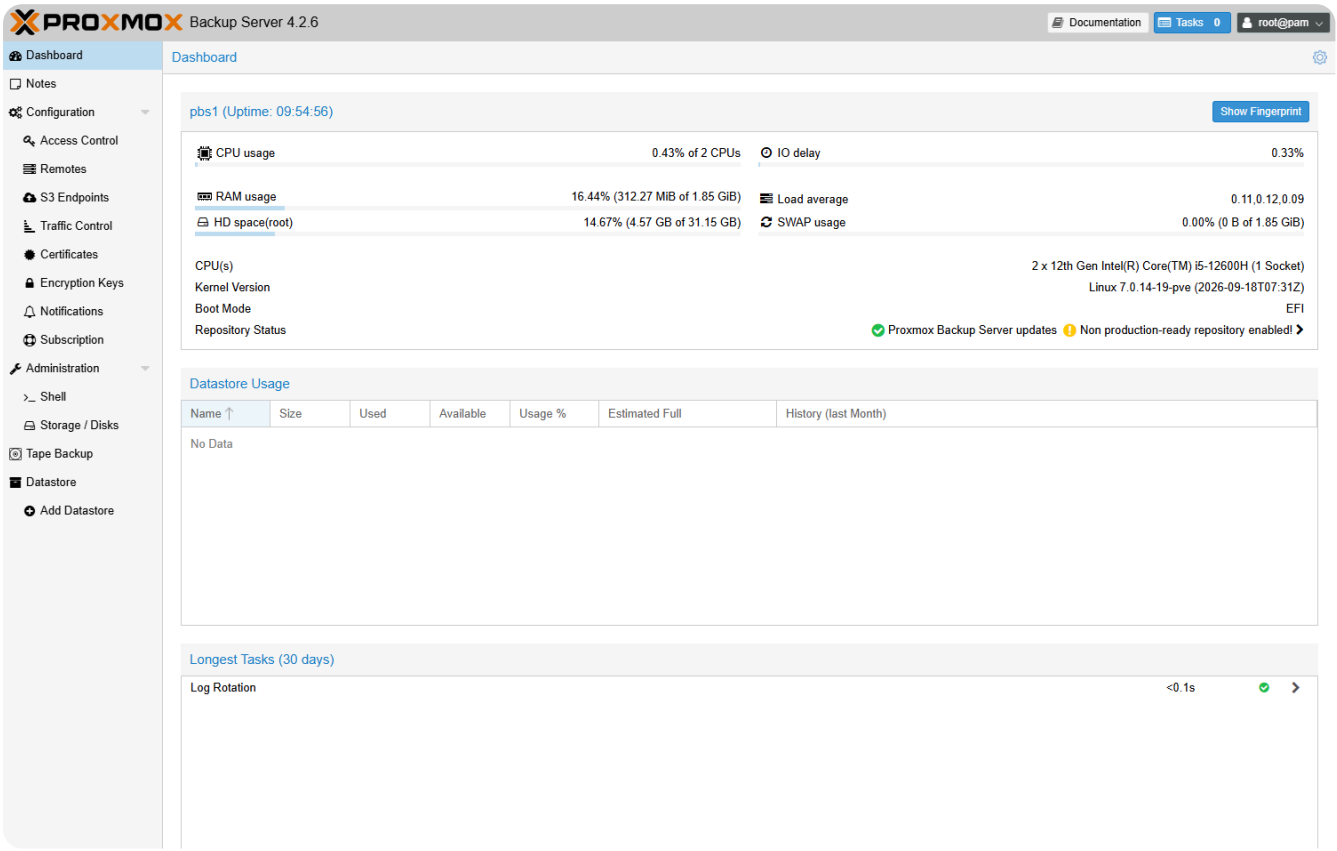
- 🎥 Fonctionnement interne de PBS — apalrd's adventures *(anglais, une quinzaine de minutes)* : le découpage en blocs, et pourquoi la deuxième sauvegarde ne transfère presque rien.
- 📖 Documentation Proxmox Backup Server — Introduction

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Se connecter à PBS

L'URL de votre PBS, `https://pbs2<N>4.<ville>.pm.dawan.ovh` : utilisateur `root`, domaine **Linux PAM standard authentication**, mot de passe `proxmoxx`.



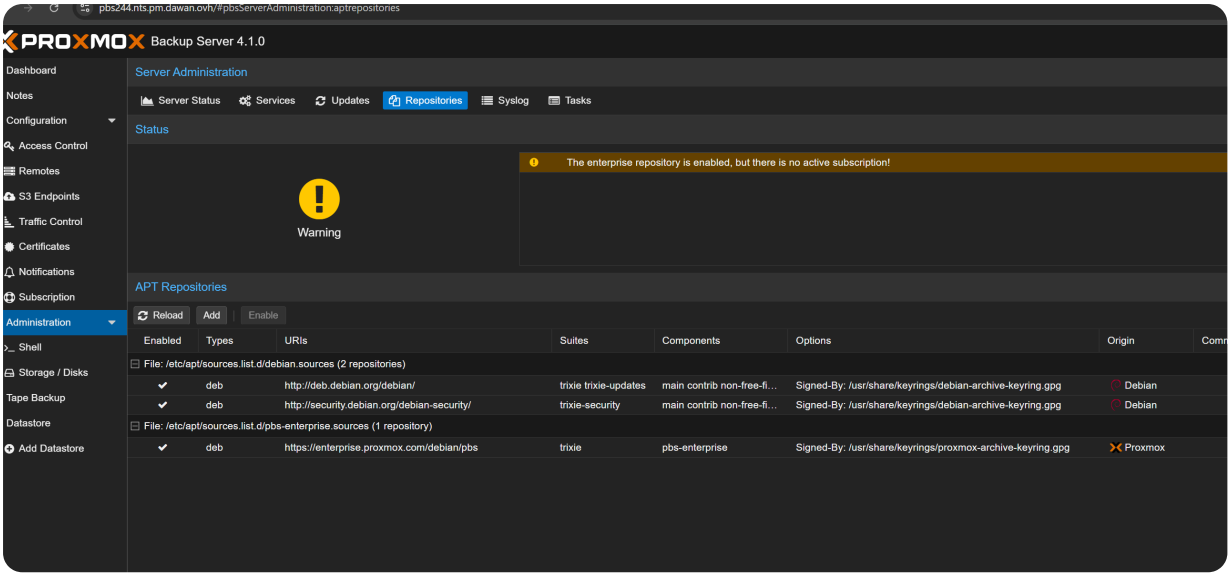
(PBS 4.2.6 du lab). Le tableau de bord : processeur, mémoire, disque système. **Datastore Usage** : *No Data*, aucun datastore n'existe encore.

À gauche, **Configuration** (dont **Access Control** pour les utilisateurs et les jetons) et **Administration** (dont **Storage / Disks**). Le bouton **Show Fingerprint** servira à l'étape 4.

Étape 2 — Passer PBS sur le dépôt sans abonnement

PBS sort d'installation comme Proxmox VE au premier jour : le dépôt réservé aux abonnés est actif, et toute mise à jour échoue sur une **erreur 401**.

Ce support est la propriété de Dawan France.



L'onglet **Administration** → **Repositories**, avec l'avertissement « *The enterprise repository is enabled, but there is no active subscription!* ». Les deux dépôts Debian restent en place. Seul `pbs-enterprise` se désactive.

1. **Administration** → **Repositories** → la ligne `https://enterprise.proxmox.com` → **Disable**
2. **Add** → **No-Subscription** → valider
3. **Administration** → **Updates** → **Refresh**, puis **Upgrade**

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

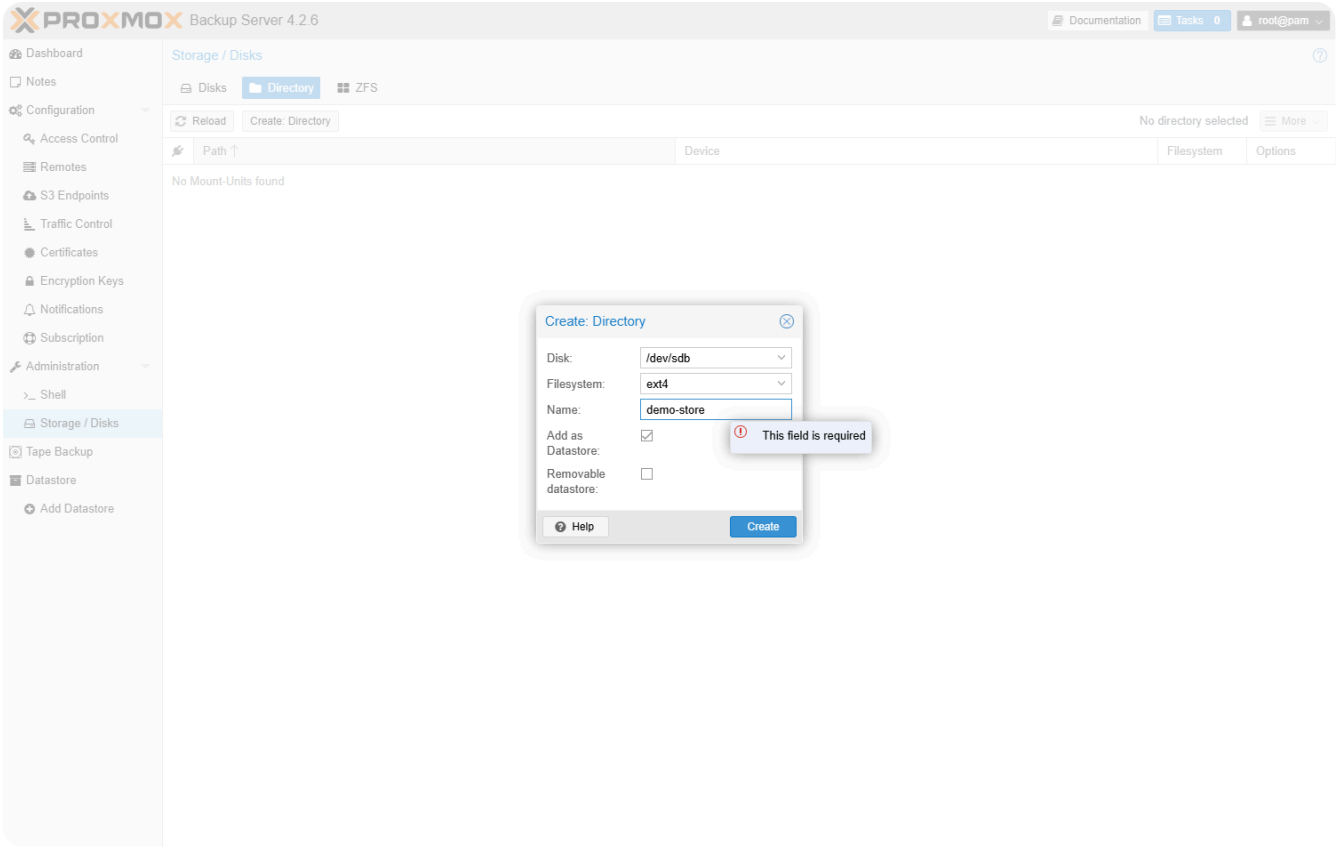
```
# Même chose en ligne de commande, depuis Administration → Shell
apt update                                  # la liste des paquets
apt full-upgrade -y                         # la mise à jour
```

Résultat attendu : plus d'erreur 401. Le tableau de bord indique alors, en face de **Repository Status**, *Non production-ready repository enabled!* : c'est le dépôt sans abonnement, adapté au lab.

Étape 3 — Créer le datastore

Administration → **Storage / Disks** → onglet **Directory** → **Create: Directory**.

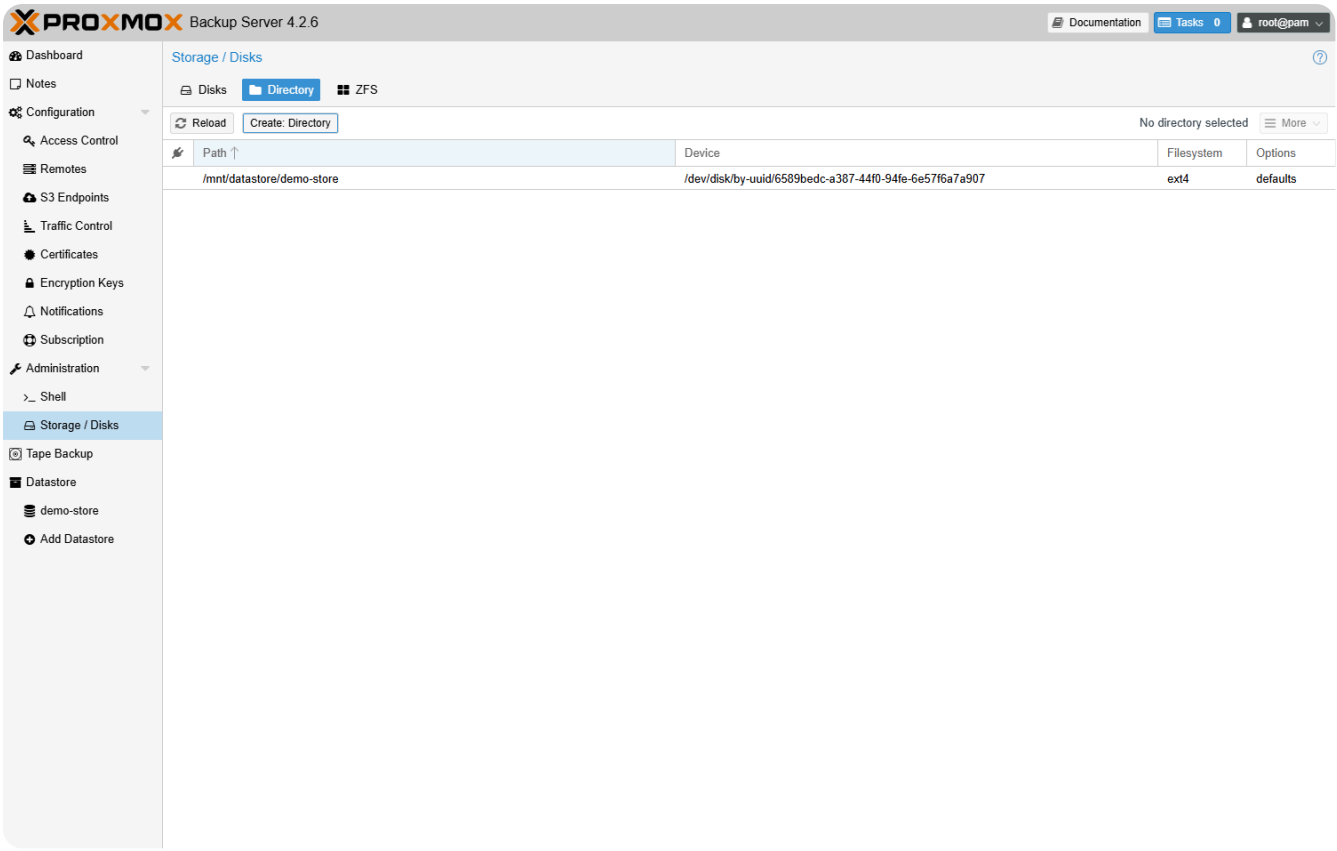
Ce support est la propriété de Dawan France.



Disk `/dev/sdb`, le disque de données de 100 Go, seul disque libre. **Filesystem** `ext4` — le champ est **obligatoire** et vide par défaut : sans lui, **Create** reste grisé.

Name `demo-store-s<N>` (votre numéro ; `demo-store` sur le lab d'exemple). **Add as Datastore** coché : le répertoire devient aussitôt un datastore. → **Create**.

Ce support est la propriété de Dawan France.



Le répertoire `/mnt/datastore/demo-store`, en `ext4`. À gauche, sous **Datastore**, `demo-store` est apparu.

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
# Vérification, dans le Shell de PBS
proxmox-backup-manager datastore list
df -h /mnt/datastore/demo-store   # espace du disque du datastore
```

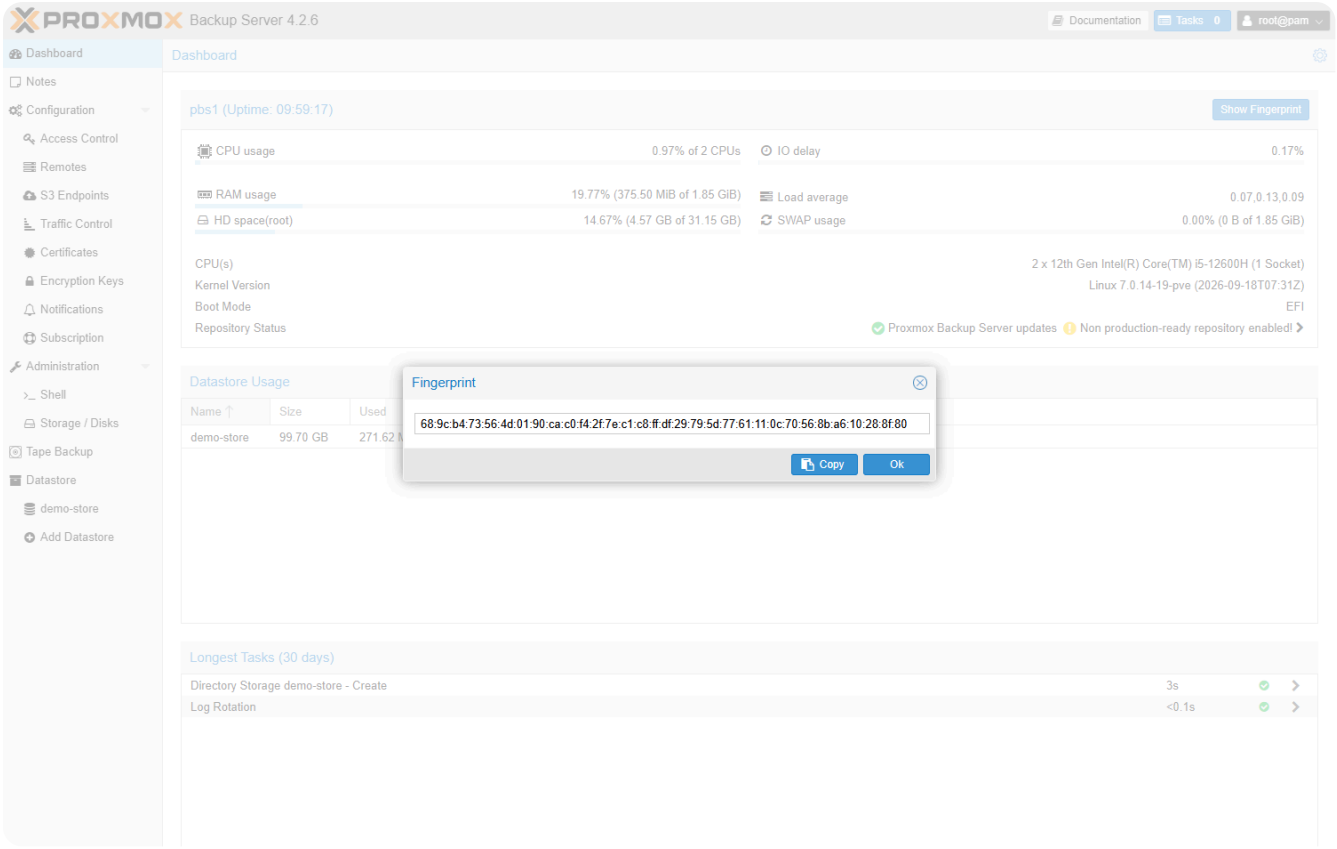
Résultat attendu :

```
| demo-store | /mnt/datastore/demo-store |         |
/dev/sdb1              98G  260M   93G   1% /mnt/datastore/demo-store
```

Étape 4 — Relever l'empreinte du certificat

Tableau de bord → **Show Fingerprint** → **Copy**.

Ce support est la propriété de Dawan France.



L'empreinte SHA-256 du certificat de PBS. Proxmox VE l'exige pour s'assurer qu'il parle au bon serveur. Derrière, **Datastore Usage** montre désormais `demo-store`, 99,70 GB.

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

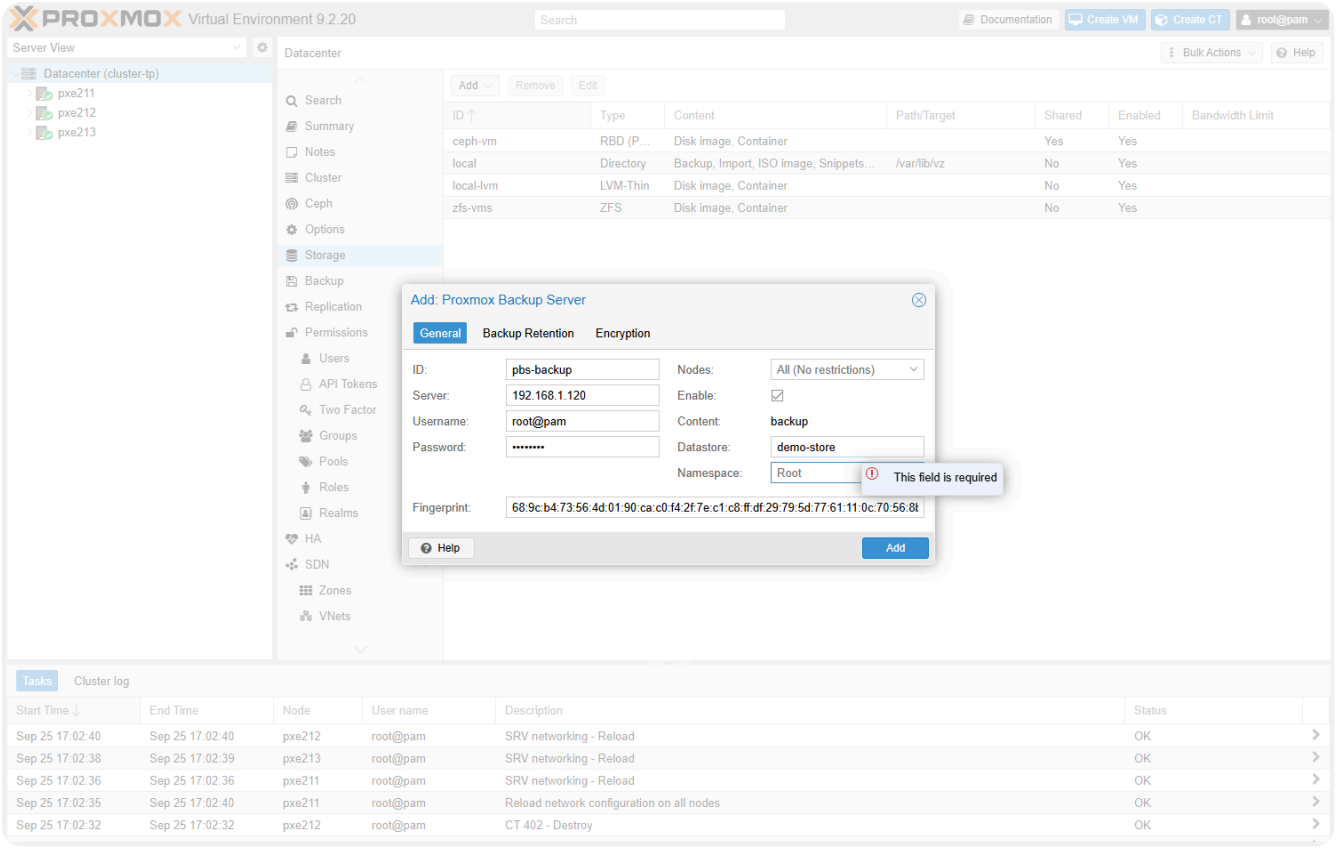
```
# Même chose dans le Shell de PBS
proxmox-backup-manager cert info | grep -i fingerprint
```

Résultat attendu (PBS 4.2.7 du lab) : `Fingerprint (sha256): cd:d6:5b:92:…:a5:6b`. Chaque PBS a la sienne : c'est celle de votre serveur qu'il faut recopier.

Étape 5 — Ajouter PBS comme stockage du cluster

Méthode 1 — par mot de passe (la plus simple en formation). Sur un nœud : **Datacenter** → **Storage** → **Add** → **Proxmox Backup Server**.

Ce support est la propriété de Dawan France.



ID `pbs-backup`. **Server** `10.<site>.<N>.14`. **Username** `root@pam`, **Password** `proxmoxx`.

Datastore `demo-store`. **Fingerprint** : celle de l'étape 4.

Namespace `Root` pour l'instant (atelier 3). **Content** `backup` : un stockage PBS ne reçoit que des sauvegardes. → **Add**.

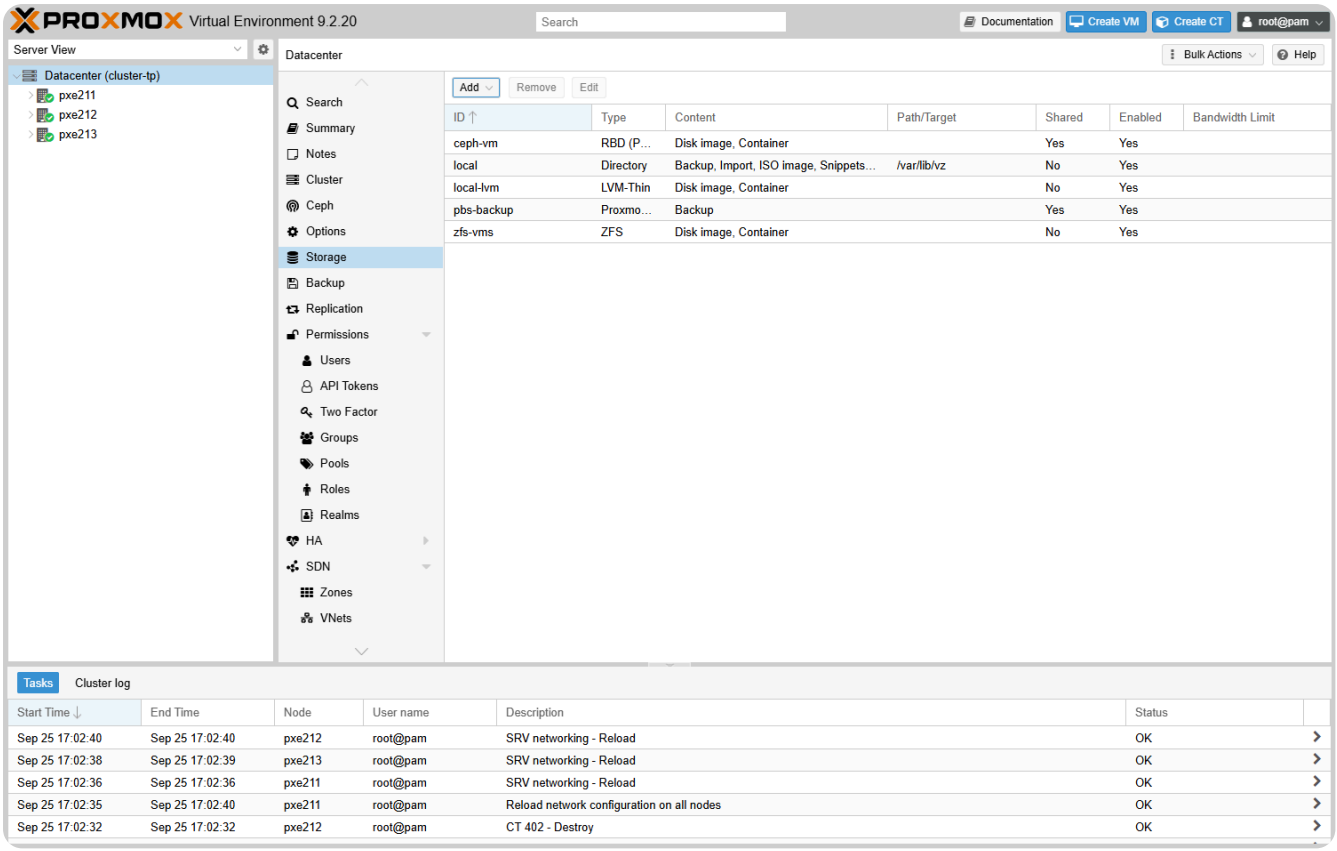
Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Équivalent CLI
pvesm add pbs pbs-backup --server 10.<site>.<N>.14 --username root@pam --password
proxmoxx \
  --datastore demo-store --fingerprint '<empreinte>'
pvesm status | grep pbs-backup   # le stockage pbs-backup est-il actif ?
```

Résultat attendu :

```
pbs-backup         pbs     active       102625188          265256        97100724
0.26%
```

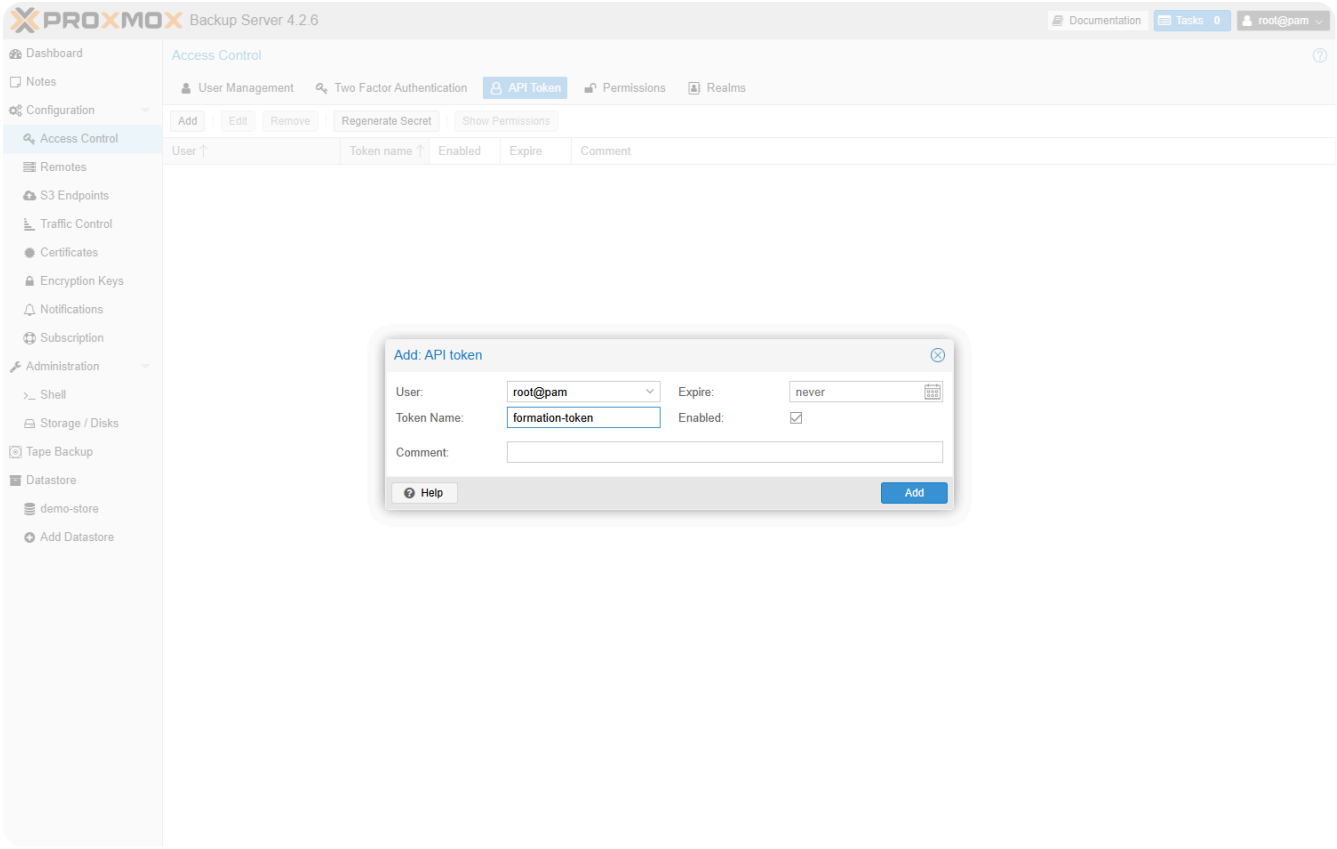
Ce support est la propriété de Dawan France.



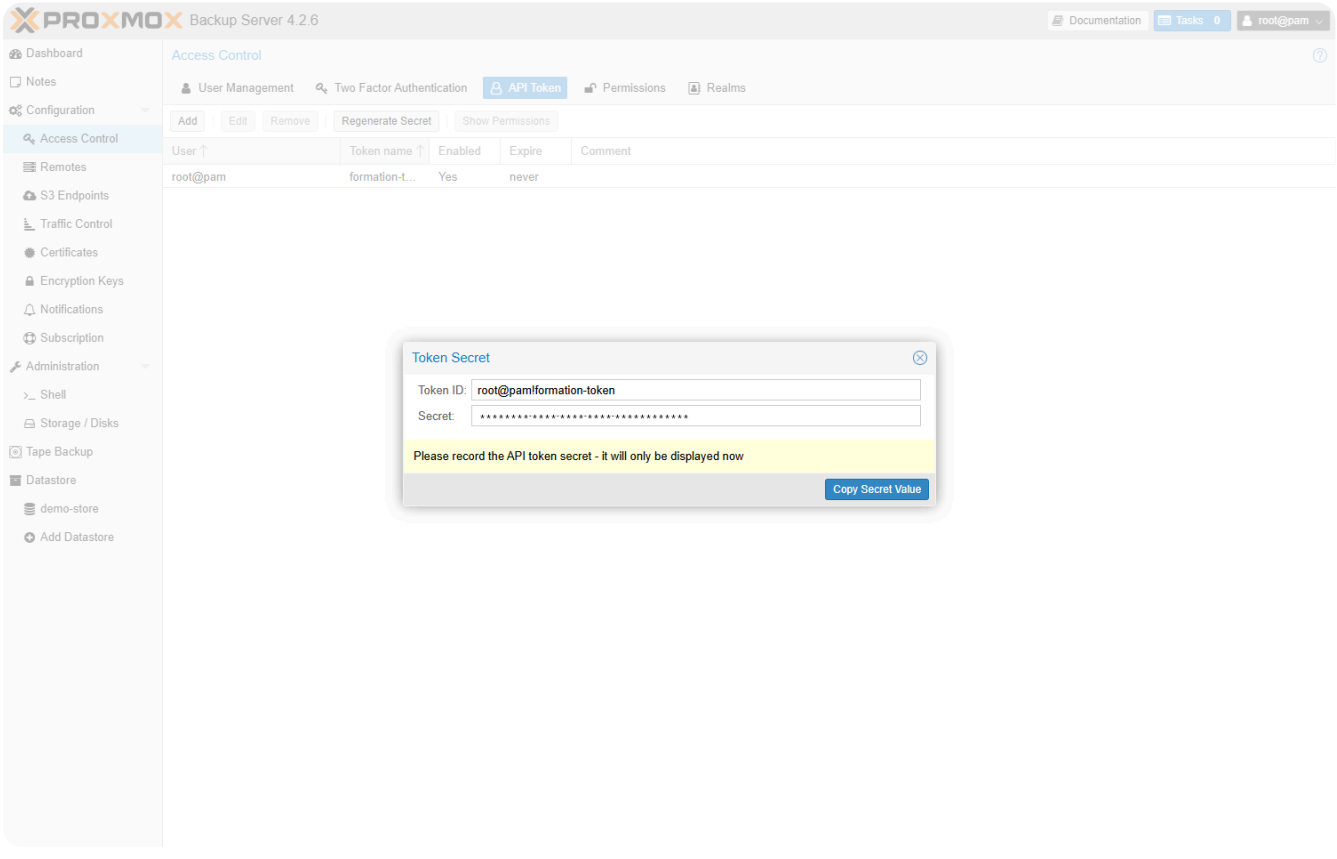
`pbs-backup`, type **Proxmox Backup Server**, contenu *Backup*, **Shared : Yes** : les trois nœuds l'utilisent.

Méthode 2 — par jeton API (la méthode de production). Dans PBS : **Configuration** → **Access Control** → onglet **API Token** → **Add** : **User** `root@pam`, **Token Name** `formation-token` → **Add**.

Ce support est la propriété de Dawan France.



Dans PBS 4.2, **Configuration → Access Control → API Token → Add** : l'utilisateur `root@pam`, le nom du jeton `formation-token`, sans date d'expiration (`never`). Contrairement à Proxmox VE, aucune case *Privilege Separation* : le jeton naît sans aucun droit, d'où l'étape suivante.



Ce support est la propriété de Dawan France.

Le **Token ID** `root@pam!formation-token` et son **Secret**, masqué ici. Il ne s'affiche **qu'une fois** : copiez-le.

> **Un jeton PBS n'a aucun droit par défaut.** La fenêtre de PBS n'a pas de case *Privilege Separation* : les droits d'un jeton se donnent à part. Sur le lab, sans droit, l'ajout du stockage a échoué.
>
> **Dans le Shell du nœud** :
>
> ```
> # Ajouter le stockage avec le jeton, avant de lui donner un rôle
> pvesm add pbs pbs-token --server 10.<site>.<N>.14 --username
> 'root@pam!formation-token' \
>   --password '<secret-du-jeton>' --datastore demo-store --fingerprint
> '<empreinte>'
> ```
>
> ```
> create storage failed: pbs-token: Cannot find datastore 'demo-store', check
> permissions and existence!
> ```
>
> Donnez-lui un rôle sur le datastore — **Configuration** → **Access Control** → **Permissions** → **Add**, ou dans le Shell de PBS :
>
> **Dans le Shell de PBS** (interface web de PBS → **Administration** → **Shell**) :
>
> ```
> proxmox-backup-manager acl update /datastore/demo-store DatastoreAdmin --
> auth-id 'root@pam!formation-token'   # rôle DatastoreAdmin du jeton sur le
> datastore
> ```

Ensuite, sur un nœud :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# ajouter le stockage avec le jeton
pvesm add pbs pbs-token --server 10.<site>.<N>.14 --username 'root@pam!formation-
token' \
  --password '<secret-du-jeton>' --datastore demo-store --fingerprint
'<empreinte>'
```

Ce support est la propriété de Dawan France.

```
pvesm status | grep pbs-token   # le stockage est-il actif ?
# Un seul stockage suffit pour la suite : retirez celui-ci
pvesm remove pbs-token
```

Résultat attendu : `pbs-token pbs active 102625188 …`, identique à `pbs-backup`.

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesm status | grep pbs-backup   # le stockage pbs-backup est-il actif ?
```

Résultat attendu : `pbs-backup pbs active`.

Atelier 3 — Namespaces PBS

Contexte

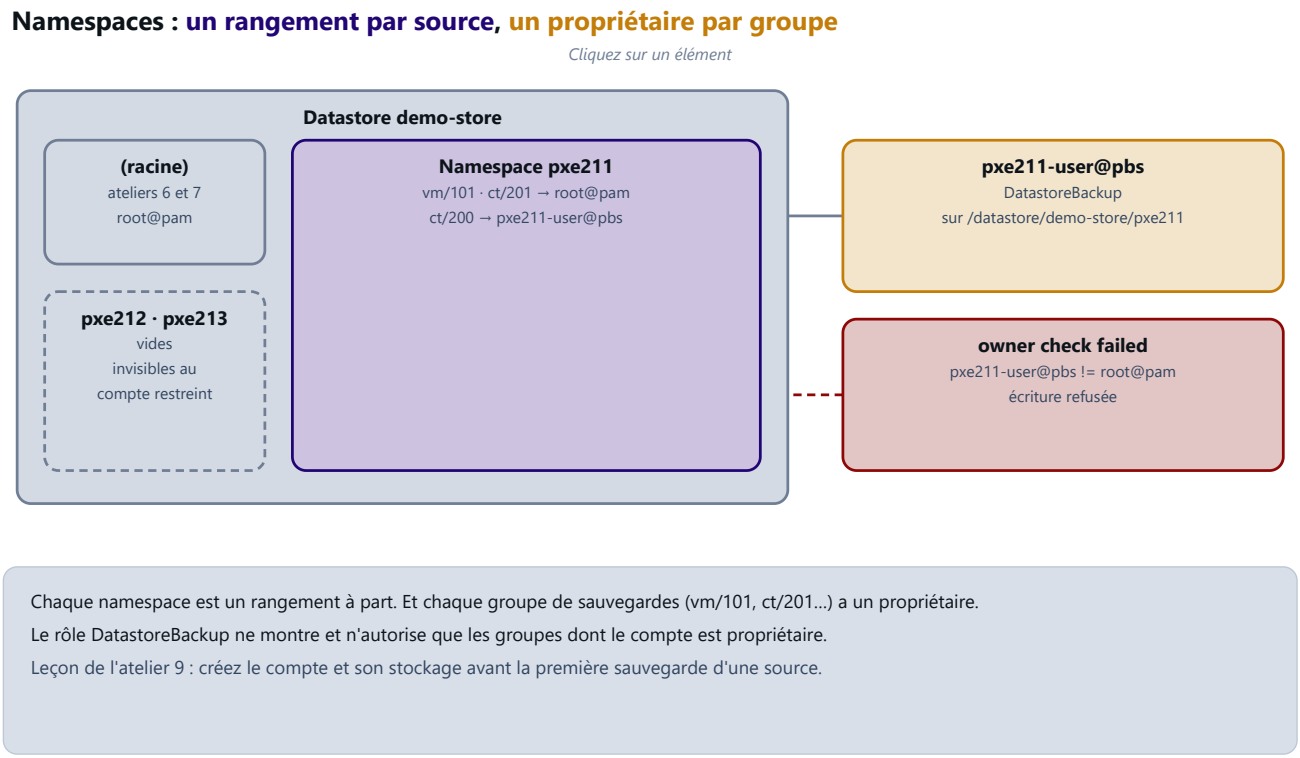
Un PBS sert rarement un seul cluster. Or deux clusters ont chacun leur VM 101 : sans précaution, leurs sauvegardes `vm/101` se retrouvent dans le même groupe.

Les **namespaces** donnent à chaque source son propre rangement. Un utilisateur PBS limité à son namespace garantit ensuite que chacun ne touche qu'à ses sauvegardes.

Vous les créez tout de suite, avant la première sauvegarde. Vos sauvegardes des ateliers 4 et 5 iront dans votre namespace, et à l'atelier 6, la copie de chaque voisin ira dans le sien : rien ne se mélange. Vous simulez plusieurs sources avec vos trois nœuds : un namespace chacun.

Résultat à obtenir : un namespace par nœud, et `pbs-backup` qui écrit dans celui de votre nœud 1. Un utilisateur PBS qui ne voit que ce namespace, et seulement les sauvegardes qui sont les siennes.

Ce support est la propriété de Dawan France.



Les namespaces rangent, les propriétaires décident de qui voit et écrit quoi. Cliquez sur un élément pour lire le résultat obtenu.

Arborescence attendue

```
PBS → demo-store-s<N>
├── (racine)            ← vide : vos sauvegardes vont dans votre namespace
├── pxe2<N>1/           ← vm/101, ct/201 (root@pam) ; ct/200 (pxe2<N>1-user@pbs)
├── pxe2<N>2/
└── pxe2<N>3/
PBS → Access Control
└── pxe2<N>1-user@pbs   ← DatastoreBackup sur /datastore/demo-store-s<N>/pxe2<N>1
seulement
```

Ce que vous devez faire

1. **Créer un namespace par nœud** dans votre datastore.
2. **Faire pointer `pbs-backup` sur le namespace de votre nœud 1**, puis sauvegarder la VM 101 et le CT 201.
3. **Créer un utilisateur PBS** limité à ce namespace, et constater ce qu'il voit.
4. **Sauvegarder avec ce compte**, et comprendre la notion de propriétaire.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> Les commandes et les sorties gardent le nom `demo-store` du lab d'exemple : remplacez-le par votre datastore, `demo-store-s<N>` .

Étape 1 — Créer les namespaces

Dans PBS : **Datastore** → votre datastore → **Content** → **Add Namespace**, une fois par nœud.

En ligne de commande, avec `proxmox-backup-client` , installé sur PBS comme sur vos nœuds. (`proxmox-backup-manager` , l'outil du serveur, n'a pas de sous-commande `namespace` .)

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
export PBS_REPOSITORY='root@pam@10.<site>.<N>.14:demo-store'   # le dépôt : utilisateur@serveur:datastore
export PBS_PASSWORD='proxmoxx'   # le mot de passe
export PBS_FINGERPRINT='<empreinte>'   # l'empreinte du certificat de PBS
proxmox-backup-client namespace create pxe2<N>1   # le namespace du nœud 1
proxmox-backup-client namespace create pxe2<N>2   # celui du nœud 2
proxmox-backup-client namespace create pxe2<N>3   # celui du nœud 3
proxmox-backup-client namespace list   # les namespaces créés
```

Résultat attendu : les trois namespaces, `pxe211` , `pxe212` et `pxe213` (l'ordre d'affichage peut varier).

Étape 2 — Faire écrire `pbs-backup` dans votre namespace

Datacenter → **Storage** → `pbs-backup` → **Edit** → **Namespace** : `pxe2<N>1` → **OK**.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesm set pbs-backup --namespace pxe2<N>1   # le stockage écrit dans le namespace du nœud 1
grep -A 8 '^pbs: pbs-backup' /etc/pve/storage.cfg   # la déclaration du stockage
```

Ce support est la propriété de Dawan France.

```
ls -l /etc/pve/priv/storage/pbs-backup.pw   # le fichier du mot de passe du stockage
```

Résultat attendu (PBS en `10.59.1.14`) :

```
pbs: pbs-backup
    datastore demo-store
    server 10.59.1.14
    content backup
    fingerprint cd:d6:5b:92:…:a5:6b
    namespace pxe211
    username root@pam

-rw------- 1 root www-data 9 Oct  7 15:14 /etc/pve/priv/storage/pbs-backup.pw
```

Un stockage ajouté par l'interface porte en plus la ligne `prune-backups keep-all=1` (la rétention par défaut du stockage : tout garder).

Le mot de passe n'est pas dans `storage.cfg` : il est rangé à part, lisible par root seul.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
vzdump 101 --storage pbs-backup --mode snapshot   # sauvegarder la VM 101
vzdump 201 --storage pbs-backup --mode snapshot   # puis le CT 201
```

Résultat attendu : `creating Proxmox Backup Server archive 'vm/101/…'`, puis pour le CT 201 : `had to backup 524.068 MiB of 524.068 MiB`. Tout est envoyé : c'est la première sauvegarde de ce conteneur dans ce namespace.

Désormais, tout ce que vous sauvegardez par `pbs-backup` va dans `pxe2<N>1`, y compris aux ateliers 4 et 5.

Étape 3 — Un utilisateur limité à son namespace

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

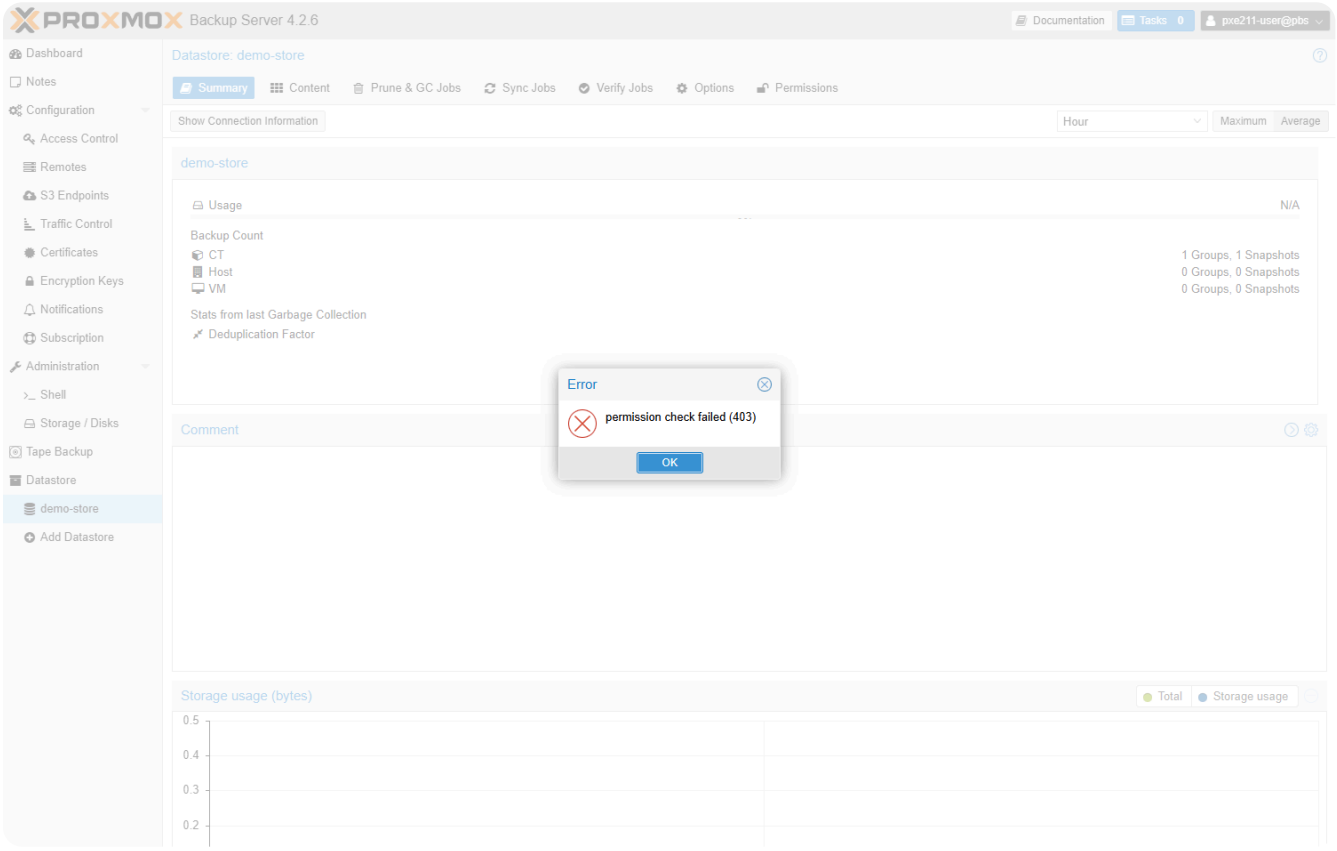
```
# un utilisateur PBS pour le nœud 1
proxmox-backup-manager user create pxe2<N>1-user@pbs \
```

Ce support est la propriété de Dawan France.

```
  --comment "Accès restreint au namespace pxe2<N>1" --password
'FormationPbs2026!'
# Le chemin d'un namespace : celui du datastore, suivi du nom du namespace
proxmox-backup-manager acl update /datastore/demo-store/pxe2<N>1 DatastoreBackup
--auth-id pxe2<N>1-user@pbs
proxmox-backup-manager acl list   # Les ACL de PBS
```

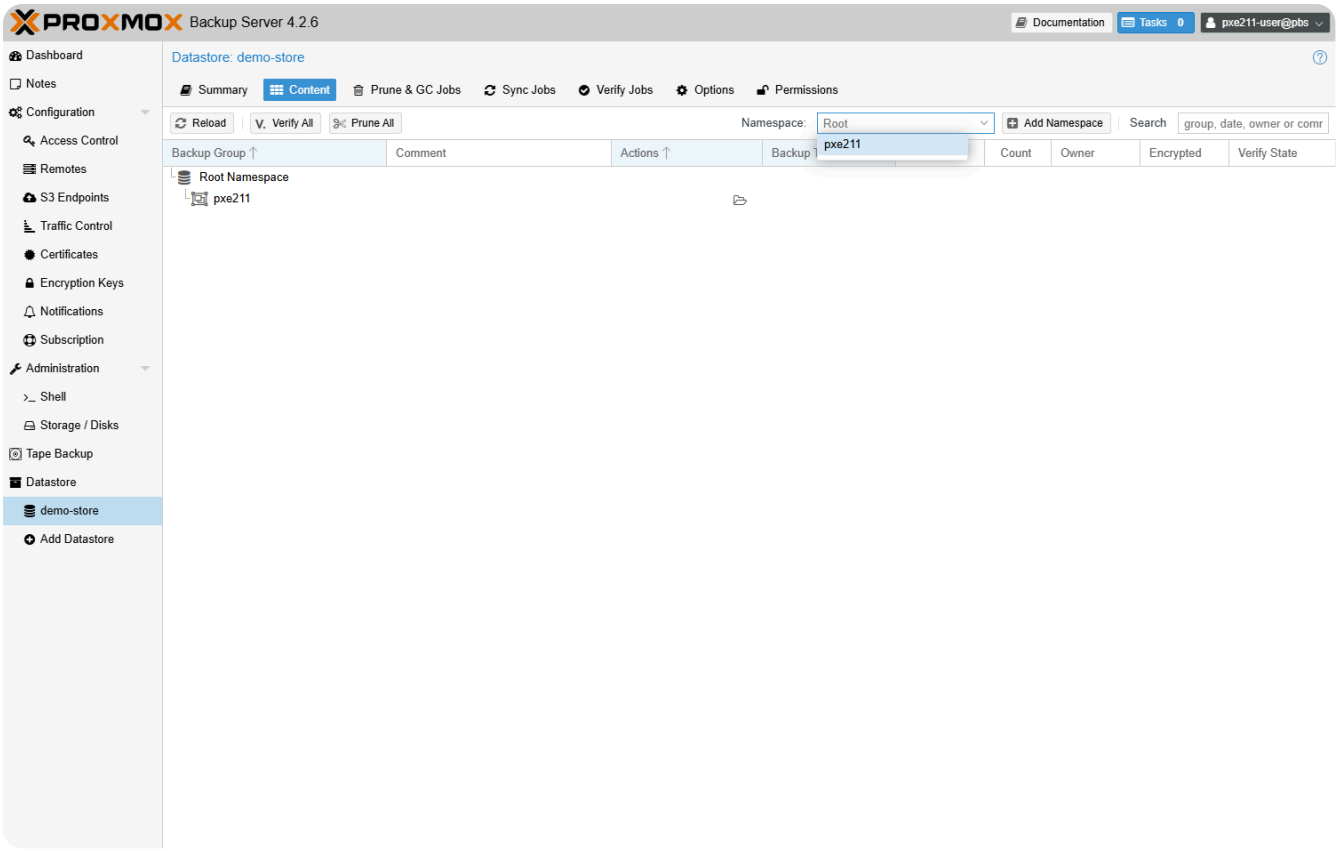
Résultat attendu : une ligne `pxe211-user@pbs | /datastore/demo-store/pxe211 | 1 | DatastoreBackup`.

Connectez-vous à PBS avec ce compte : utilisateur `pxe2<N>1-user`, domaine **Proxmox Backup authentication server**.



En ouvrant le datastore, le compte reçoit **`permission check failed (403)`** : il n'a aucun droit sur la racine du datastore. C'est voulu. Fermez, puis onglet **Content**.

Ce support est la propriété de Dawan France.



La liste **Namespace** ne propose que `pxe211` : ni `pxe212`, ni `pxe213`, ni les groupes de la racine. Sélectionnez-le.

Surprise : le namespace paraît **vide**. La VM 101 et le CT 201 y sont pourtant.

Le rôle `DatastoreBackup` ne montre que les sauvegardes **dont le compte est propriétaire**. Celles-ci ont été faites par `root@pam`.

Étape 4 — Sauvegarder avec le compte restreint

C'est la bonne pratique : chaque source écrit avec son propre compte. Déclarez un second stockage, limité au nœud 1 :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# stockage limité au namespace du nœud 1, avec cet utilisateur
pvesm add pbs pbs-pxe2<N>1 --server 10.<site>.<N>.14 --datastore demo-store --
namespace pxe2<N>1 \
  --username pxe2<N>1-user@pbs --password 'FormationPbs2026!' --fingerprint
'<empreinte>' --nodes pxe2<N>1
```

Ce support est la propriété de Dawan France.

```
vzdump 101 --storage pbs-pxe2<N>1 --mode snapshot   # sauvegarder la VM 101 par ce stockage
```

Résultat attendu :

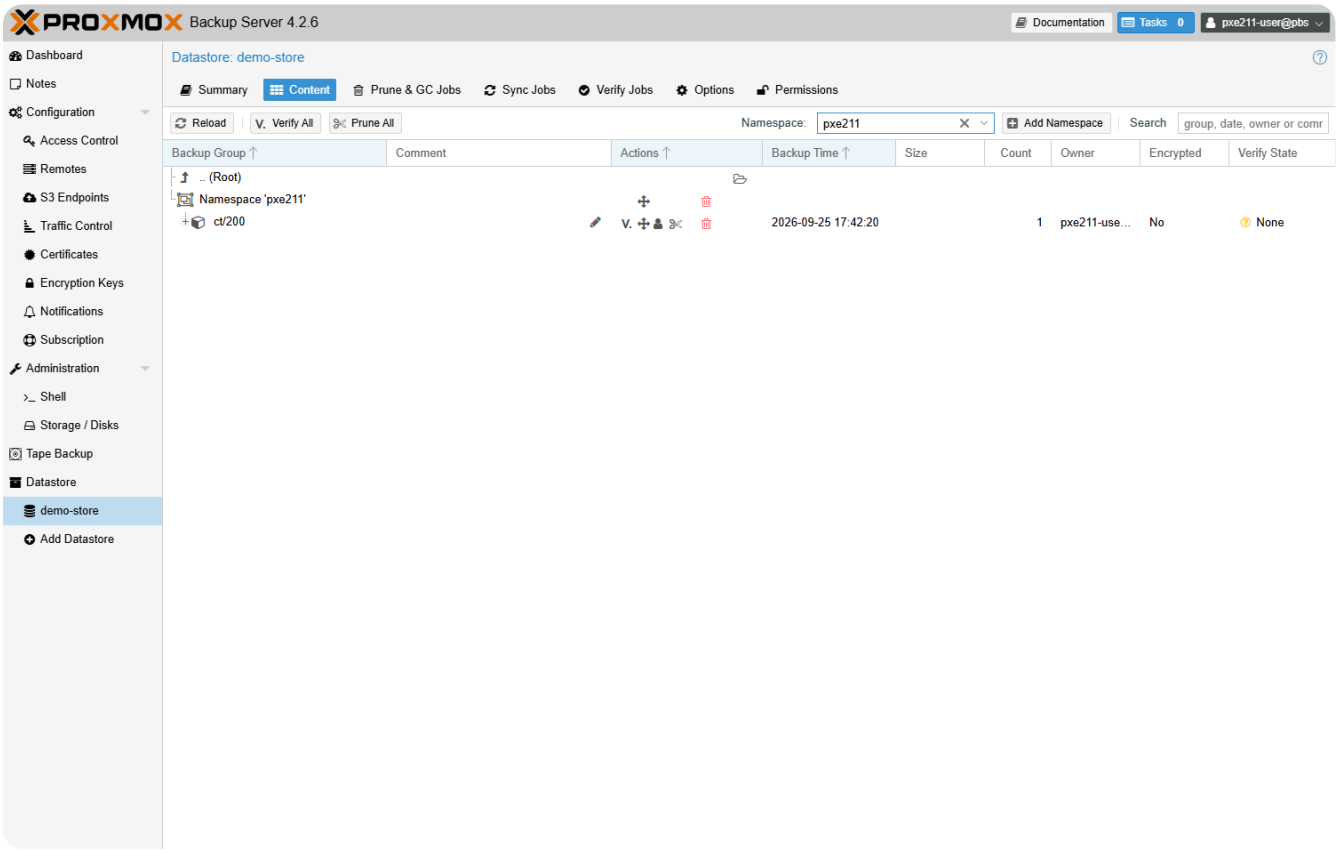
```
ERROR: VM 101 qmp command 'backup' failed - backup connect failed: command error: backup owner check failed (pxe211-user@pbs != root@pam)
```

Chaque groupe de sauvegardes a un **propriétaire**. Le groupe `vm/101` de ce namespace appartient à `root@pam` : un autre compte ne peut pas y ajouter de sauvegarde. Essayez avec une machine jamais sauvegardée dans ce namespace, le CT 200 :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
vzdump 200 --storage pbs-pxe2<N>1 --mode snapshot   # sauvegarder le CT 200 par ce stockage
```

Résultat attendu : `creating Proxmox Backup Server archive 'ct/200/2026-10-07T16:51:17Z'`, puis `Finished Backup of VM 200 (00:00:05)`.



Rechargez **Content** avec le compte restreint : `ct/200` apparaît, **Owner** `pxe211-use…`. Les sauvegardes de `root@pam` restent invisibles. Leçon : **créez le compte et son stockage avant la première sauvegarde** d'une source.

Retirez le stockage de test : `pvesm remove pbs-pxe2<N>1`. Les sauvegardes restent dans PBS.

Validation finale

Dans le Shell de PBS, avec les variables `PBS_*` de l'étape 1 :

```
proxmox-backup-client namespace list   # les namespaces du datastore
```

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
grep -A 8 '^pbs: pbs-backup' /etc/pve/storage.cfg | grep namespace   # le
namespace du stockage pbs-backup
```

Résultat attendu : les trois namespaces, puis `namespace pxe211`.

Atelier 4 — Sauvegardes avec PBS

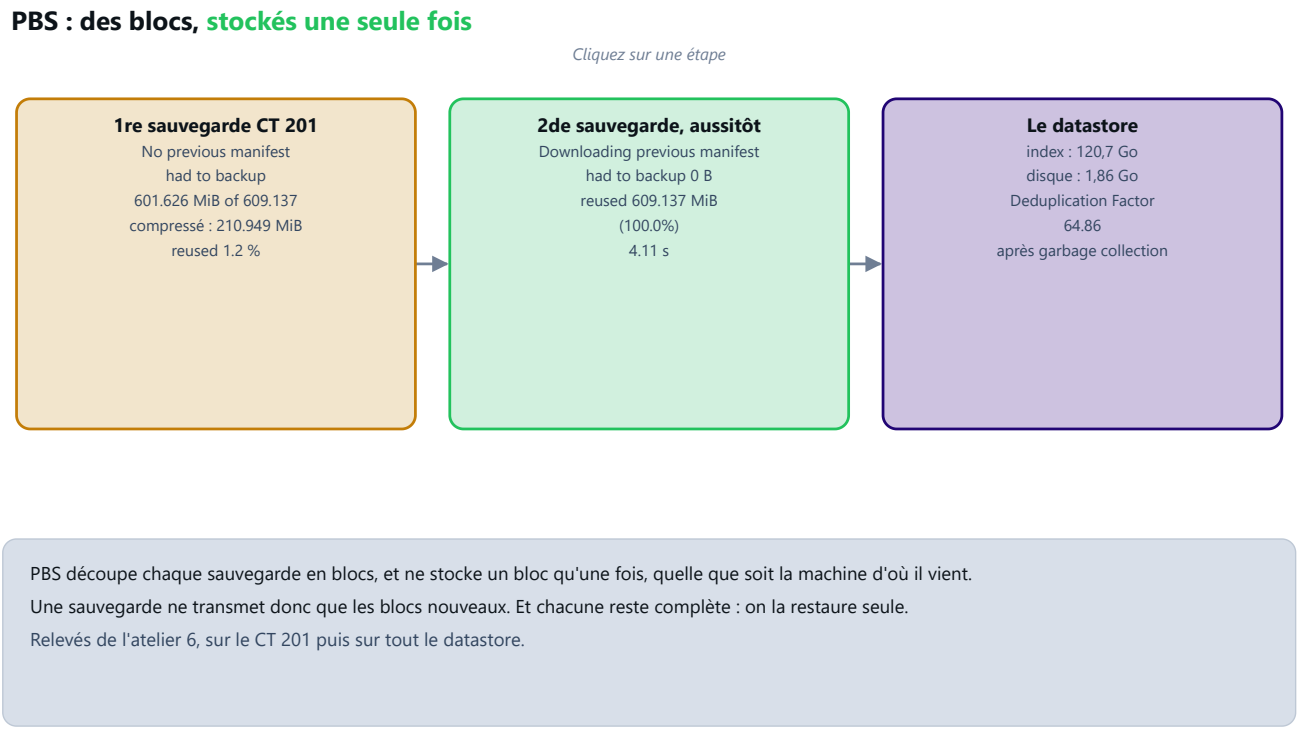
Contexte

Le stockage est en place ; il faut s'en servir. Une sauvegarde lancée à la main prouve que la chaîne fonctionne.

C'est le **job planifié** qui protège réellement : personne ne pense à sauvegarder la veille de la panne. La **rétention** décide combien de versions on garde.

Résultat à obtenir : le CT 201 sauvegardé deux fois, pour voir l'effet incrémental. Toutes les machines de votre nœud sauvegardées, un job nocturne avec rétention, et le facteur de déduplication noté dans PBS.

Ce support est la propriété de Dawan France.



La première sauvegarde, la seconde, puis le datastore entier. Cliquez sur une étape pour lire les chiffres obtenus.

Arborescence attendue

```
PBS → Datastore demo-store-s<N> → Content → namespace pxe2<N>1
├── ct/201   ← 2 sauvegardes : la seconde n'envoie rien
├── vm/101, vm/102, vm/300 …   ← vzdump --all
Votre cluster → Datacenter → Backup
└── job 02:00 · pbs-backup · All · keep-last=7, keep-weekly=4
```

Ce que vous devez faire

1. **Sauvegarder le CT 201** vers `pbs-backup`, **deux fois de suite**, et comparer les journaux.
2. **Sauvegarder toutes les machines de votre nœud** en une commande.
3. **Créer un job planifié** chaque nuit à 2 h : 7 sauvegardes récentes, 4 hebdomadaires.
4. **Relever dans PBS** le nombre de sauvegardes et le facteur de déduplication.

Dawan
Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

Étape 1 — Sauvegarde manuelle, deux fois

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Mode snapshot : le CT reste démarré. Pas de --compress : PBS compresse lui-même.
vzdump 201 --storage pbs-backup --mode snapshot
```

Ou : CT 201 → **Backup** → **Backup now** → Storage `pbs-backup`.

Résultat attendu (première sauvegarde, extrait) :

```
INFO: Starting backup: ct/201/2026-10-07T13:15:07Z
INFO: No previous manifest available.
INFO: root.pxar: had to backup 524.068 MiB of 524.068 MiB (compressed 178.918 MiB) in 3.91 s (average 133.99 MiB/s)
INFO: Finished Backup of VM 201 (00:00:05)
```

Relancez **la même commande** aussitôt.

Résultat attendu (seconde sauvegarde) :

```
INFO: Downloading previous manifest (Wed Oct  7 15:15:07 2026)
INFO: root.pxar: had to backup 0 B of 524.068 MiB (compressed 0 B) in 3.51 s (average 0 B/s)
INFO: root.pxar: backup was done incrementally, reused 524.068 MiB (100.0%)
```

0 octet envoyé. PBS découpe les données en blocs et ne transmet que les blocs qu'il n'a pas. Rien n'a changé dans le conteneur : rien ne voyage, et pourtant la seconde sauvegarde est complète et restaurable seule.

Étape 2 — Sauvegarder toutes les machines du nœud

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

```
# --all : toutes les VM et tous les CT du nœud où la commande est lancée
vzdump --all --storage pbs-backup --mode snapshot
pvesh get /nodes/<nœud>/storage/pbs-backup/content --output-format json | grep -o
'"volid":"[^"]*"'   # les sauvegardes présentes sur le stockage
```

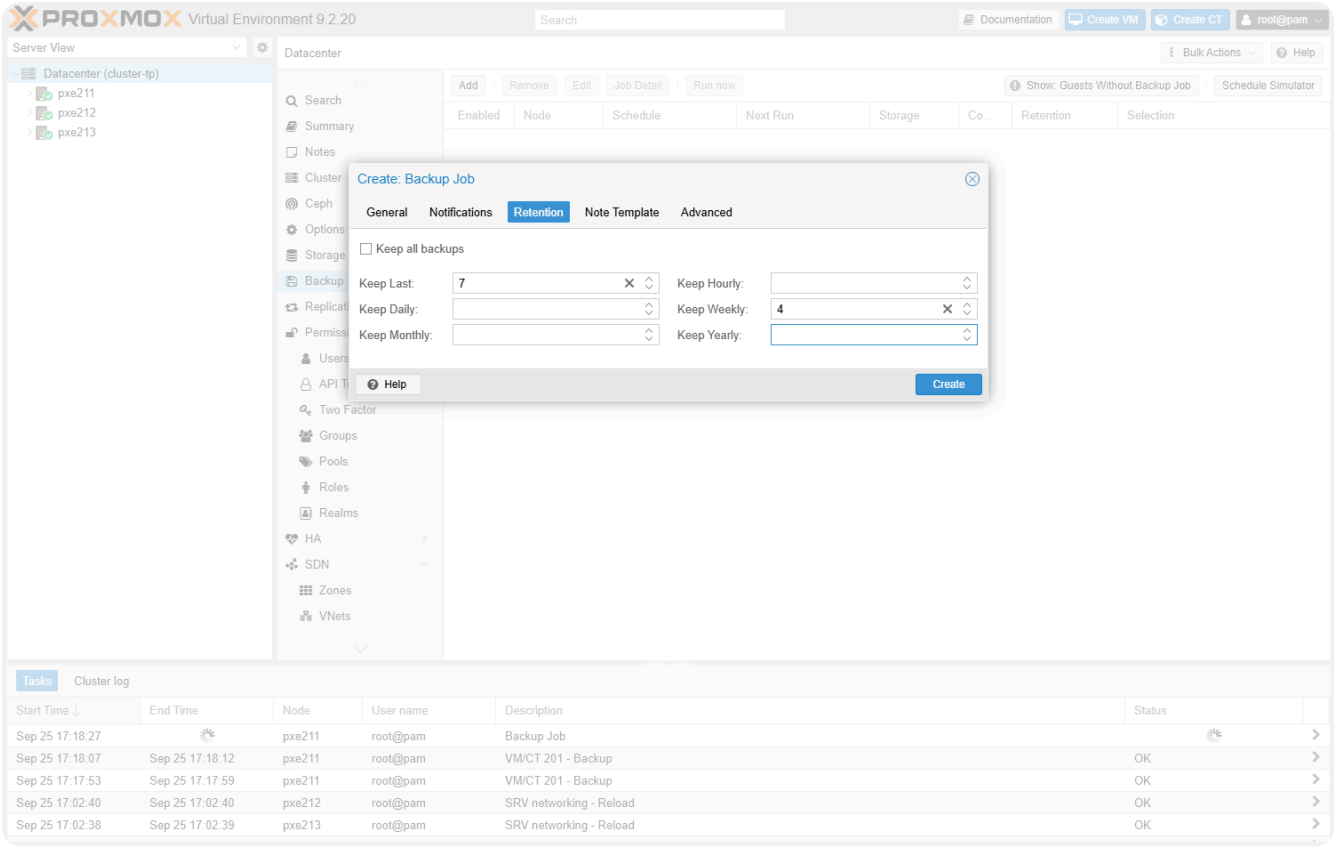
Résultat attendu (extrait, `pxe211`) : `skip external VMs: 102, 120` (les VM 102 et 120 sont sur un autre nœud). Suit une sauvegarde par machine : `Finished Backup of VM 100 (00:00:08)`, `VM 101 (00:00:06)` … et un disque de 20 Gio sauvegardé en quelques secondes. Douze machines en 1 min 48 sur le lab.

Étape 3 — Configurer un job planifié

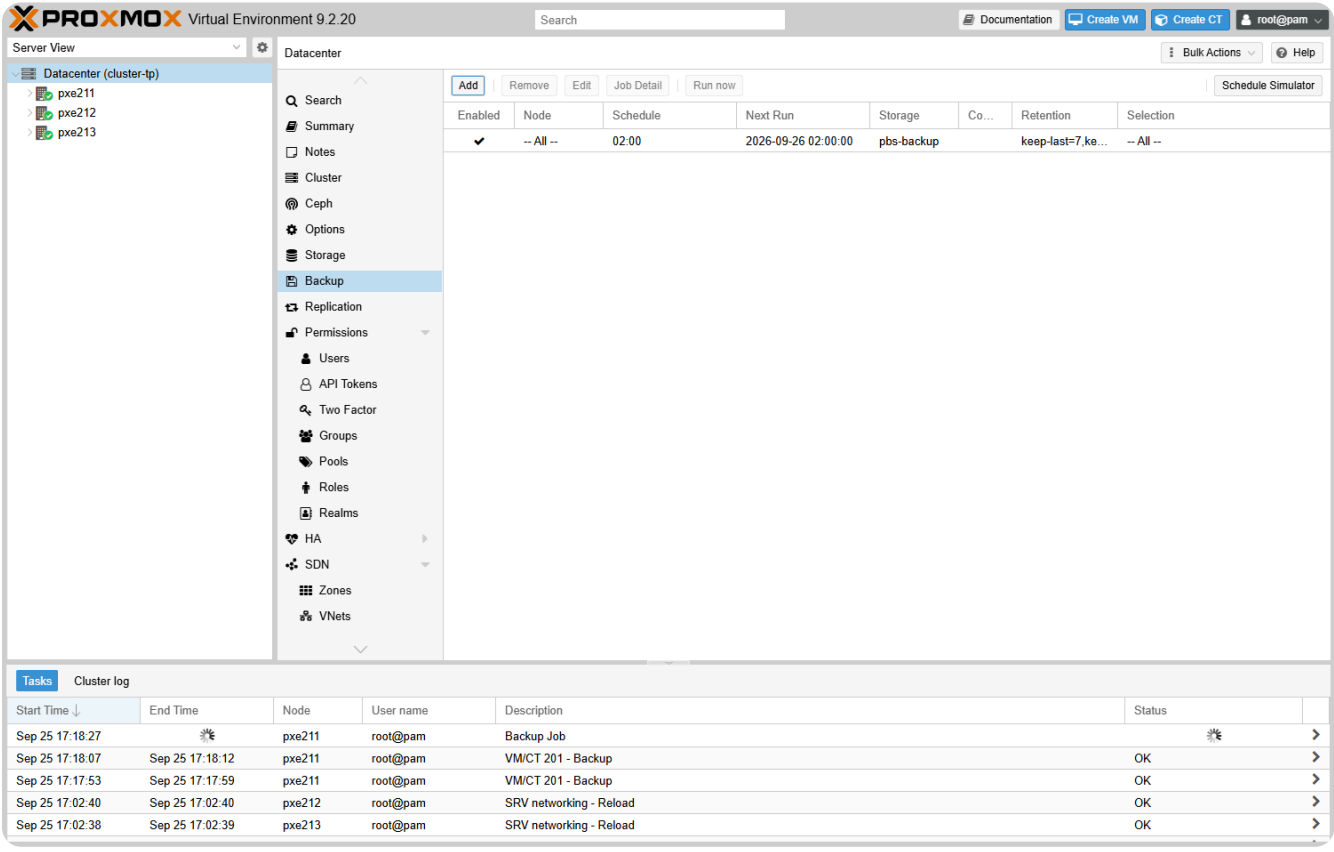
Datacenter → Backup → Add :

Onglet	Champ	Valeur
General	Storage	`pbs-backup`
General	Schedule	`02:00`
General	Selection mode	`All`
General	Mode	`Snapshot`
Retention	Keep Last	`7`
Retention	Keep Weekly	`4`

Ce support est la propriété de Dawan France.



Keep Last `7`, **Keep Weekly** `4`. Le bandeau jaune rappelle que, sans option ici, c'est la rétention du stockage qui s'applique. Or celle de `pbs-backup` vaut `keep-all=1` : on garderait tout.



Dawan
Ce support est la propriété de Dawan France.

Le job : tous les nœuds, `02:00`, **Next Run** le lendemain à 2 h, stockage `pbs-backup`, rétention `keep-last=7,ke…`, sélection **All**. Le bouton **Schedule Simulator**, à droite, montre les prochaines exécutions d'une planification.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le job, tel qu'enregistré pour tout le cluster
cat /etc/pve/jobs.cfg
```

Résultat attendu (extrait) :

```
vzdump: backup-9d5e8c9c-eff8
    schedule 02:00
    all 1
    mode snapshot
    prune-backups keep-last=7,keep-weekly=4
    storage pbs-backup
```

Créé en ligne de commande (`pvesh create /cluster/backup --schedule 02:00 --all 1 --mode snapshot --storage pbs-backup --prune-backups keep-last=7,keep-weekly=4`), le job porte un identifiant sans préfixe, `vzdump: 9b4d9c25-840d-4c46-9e65-26cadae22f9e` sur le lab, et une ligne `enabled 1` en plus.

> **`keep-last=7`** garde les 7 dernières sauvegardes ; **`keep-weekly=4`** en garde une par semaine sur 4 semaines. Les sauvegardes au-delà sont supprimées à chaque exécution du job.

Étape 4 — Explorer PBS après sauvegarde

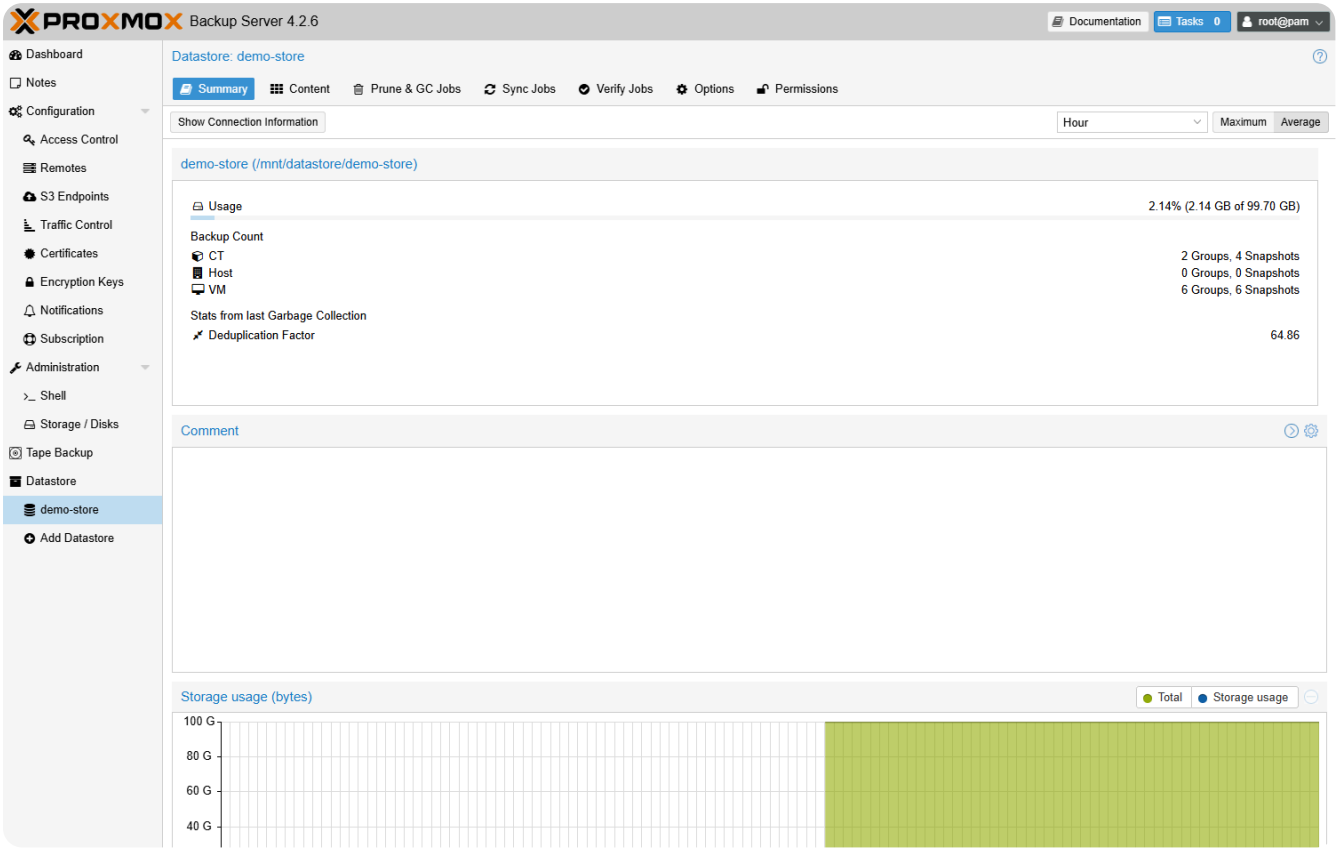
Le facteur de déduplication se calcule au passage du **ramasse-miettes** (*garbage collection*). Il se planifie dans **Datastore** → `demo-store` → onglet **Prune & GC Jobs**. Pour le lancer tout de suite, dans le Shell de PBS :

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
proxmox-backup-manager garbage-collection start demo-store   # lancer le ramasse-
```

Ce support est la propriété de Dawan France.

```
miettes du datastore
```



Usage : 2,14 GB sur 99,70 GB. **Backup Count** : 2 groupes CT et 4 sauvegardes, 6 groupes VM et 6 sauvegardes. **Deduplication Factor : 64.86**.

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

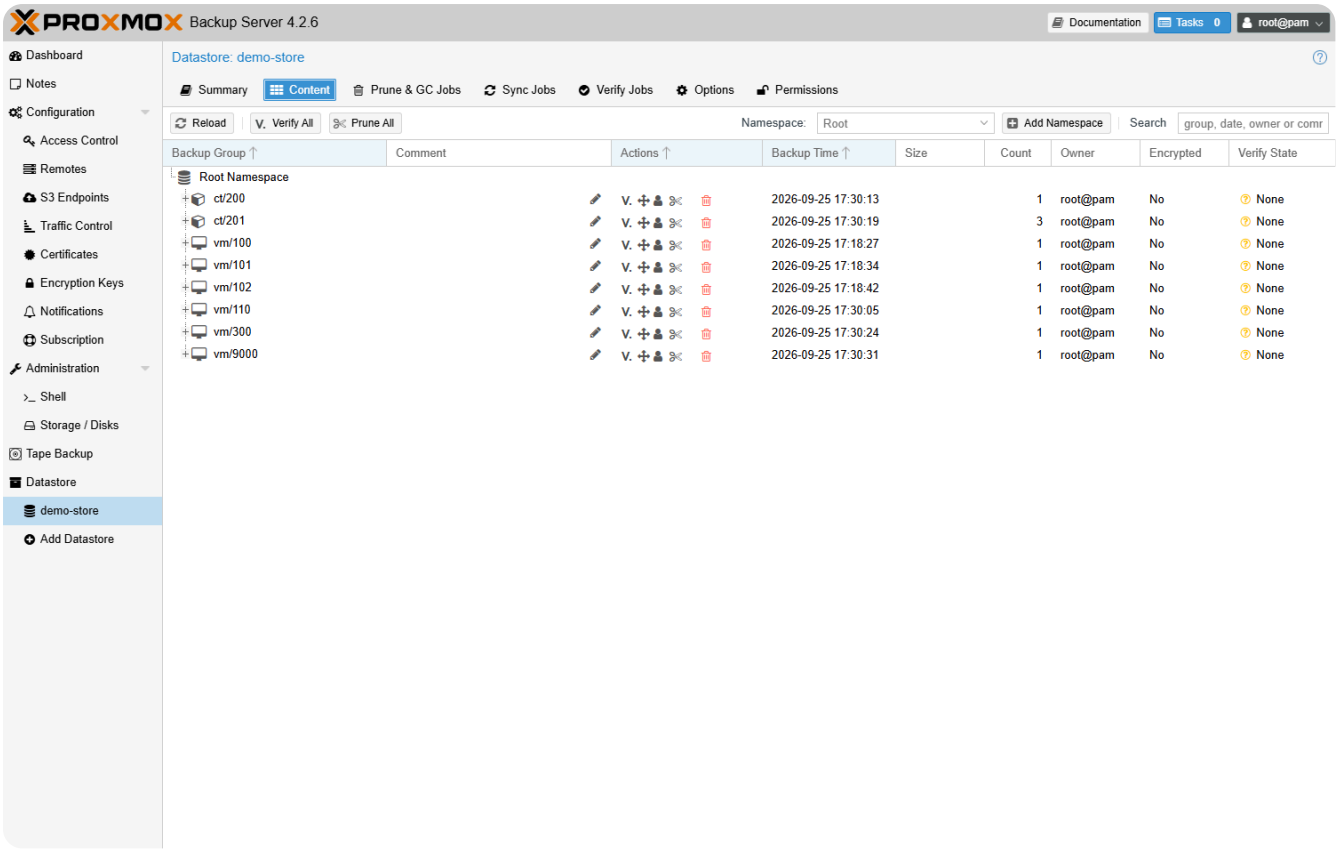
```
proxmox-backup-manager garbage-collection status demo-store --output-format json-pretty   # volume logique et volume réellement stocké ; à lire : index-data-bytes, disk-bytes
```

Résultat attendu (les lignes qui comptent) :

```
  "disk-bytes": 3210698747,
  "index-data-bytes": 200611205470,
```

200,6 Go de sauvegardes occupent 3,2 Go de disque : le ramasse-miettes annonçait `Deduplication factor: 62.48`. Les disques de VM sont en grande partie vides, et les systèmes Debian se ressemblent : leurs blocs ne sont stockés qu'une fois.

Ce support est la propriété de Dawan France.



Un groupe par machine (`ct/201`, `vm/101`…), la date de la dernière sauvegarde, et leur nombre : **3** pour le CT 201. Propriétaire `root@pam`, non chiffrées, pas encore vérifiées (**Verify State** `None`).

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pvesh get /nodes/<nœud>/storage/pbs-backup/content --output-format json | grep -c volid   # nombre de sauvegardes
grep prune-backups /etc/pve/jobs.cfg   # la rétention du job
```

Résultat attendu : une sauvegarde par machine de votre nœud, plus les deux du CT 201 (14 sur le nœud du lab, qui porte aussi des machines d'exemple), puis `prune-backups keep-last=7,keep-weekly=4`.

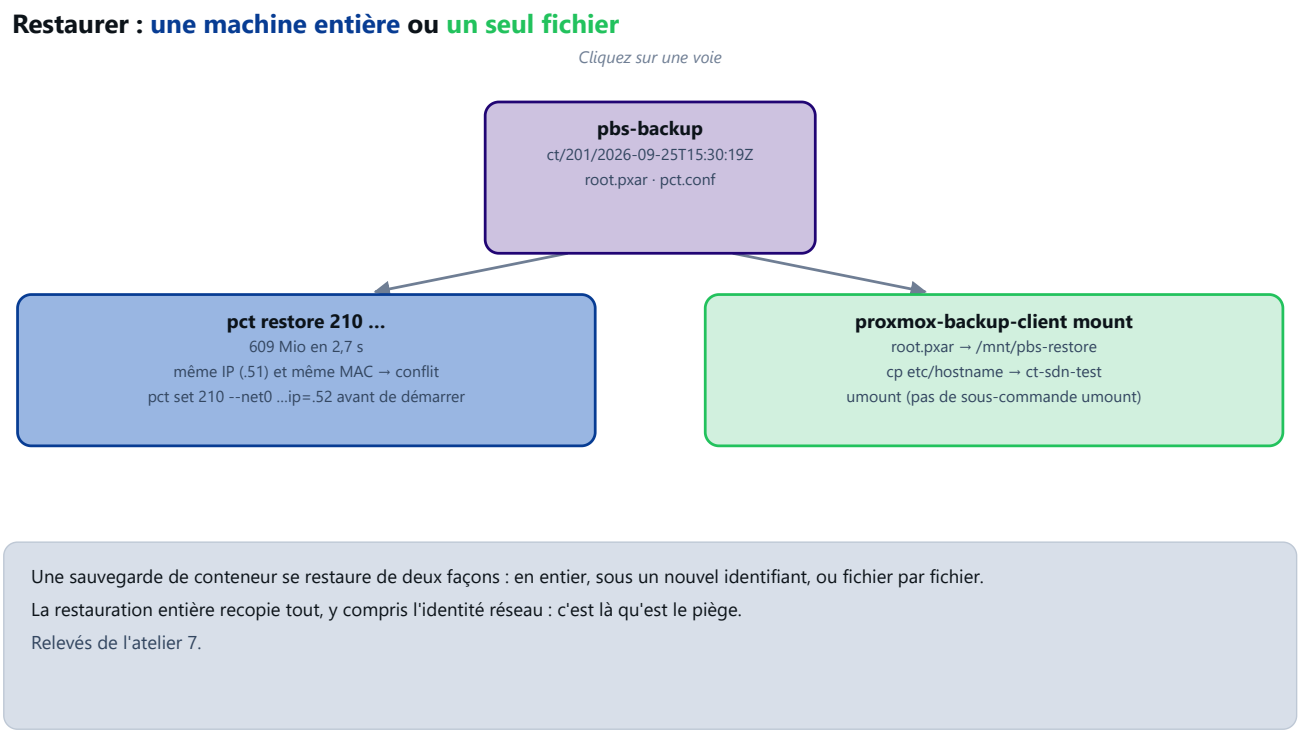
Atelier 5 — Restauration depuis PBS

Contexte

Ce support est la propriété de Dawan France.

Une sauvegarde jamais restaurée n'est qu'une hypothèse. Cet atelier l'éprouve de deux façons : recréer une machine entière **à côté** de l'originale, puis récupérer un seul fichier. C'est le cas le plus fréquent : « j'ai supprimé un fichier de configuration hier ».

Résultat à obtenir : le CT 210, copie du CT 201, démarré à côté de l'original **sans conflit d'adresse** ; et le fichier `/etc/hostname` du CT 201 extrait de la sauvegarde.



Une sauvegarde, deux usages. La restauration entière recopie aussi l'identité réseau : c'est le piège de l'étape 1. Cliquez sur une voie pour lire le résultat.

Arborescence attendue

```
Votre nœud
├── CT 201   ← l'original, intact, en .51
├── CT 210   ← restauré depuis PBS, nouvelle MAC, en .52
└── /tmp/hostname-restore   ← un seul fichier, extrait de la sauvegarde
```

Ce que vous devez faire

1. **Restaurer la sauvegarde du CT 201 sous l'identifiant 210**, lui donner une autre adresse, puis le démarrer.

Ce support est la propriété de Dawan France.

2. **Monter la sauvegarde en lecture seule** sur le nœud, en extraire `/etc/hostname`, puis démonter.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **La restauration se pilote depuis Proxmox VE, pas depuis PBS.** Dans PBS, l'onglet **Content** d'un datastore sert à consulter. Le bouton **Restore** est côté Proxmox VE : votre nœud → stockage `pbs-backup` → onglet **Backups**.

Étape 1 — Restaurer une machine sous un nouvel identifiant

Une sauvegarde de conteneur (`ct/`) se restaure avec `pct restore` . Une sauvegarde de VM (`vm/`), avec `qmrestore` .

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Les sauvegardes du CT 201 : l'identifiant complet est en première colonne
pvesm list pbs-backup | grep 'ct/201/'

pct restore 210 'pbs-backup:backup/ct/201/<date>' --storage local-lvm --
unprivileged 1   # restaurer le CT 201 sous l'ID 210
pct config 210   # nom et réseau de la copie ; à lire : hostname, net0
```

Résultat attendu (les lignes qui comptent) :

```
restoring 'pbs-backup:backup/ct/201/2026-10-07T13:15:07Z' now..
restore complete (524.068 MiB processed in 2.1s, average 251.724 MiB/s)
hostname: ct-sdn-test
net0:
name=eth0,bridge=vmbr0,gw=10.59.1.1,hwaddr=BC:24:11:8A:92:EF,ip=10.59.1.51/24,typ
e=veth
```

> **Ne démarrez pas le CT 210 tel quel.** La restauration reprend **la même adresse IP et la même adresse MAC** que le CT 201, qui tourne toujours. Deux machines, une seule identité réseau : conflit.
>
> Changez sa carte avant de le démarrer. Sans `hwaddr` , Proxmox tire une nouvelle MAC :
>
> **Dans le Shell du nœud** (interface web → votre nœud → **Shell**) :

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

```
pct set 210 --net0 name=eth0,bridge=vmbr0,ip=10.<site>.<N>.52/24,gw=10.
<site>.<N>.1,firewall=1   # une autre adresse fixe pour la copie
pct start 210   # la démarrer
pct exec 210 -- ip -4 -br addr show eth0   # son adresse
```

Résultat attendu : `net0: …,hwaddr=BC:24:11:11:40:D4,ip=10.59.1.52/24…` (une MAC neuve), puis `eth0@if66 UP 10.59.1.52/24`. Le CT 201 garde `.51` ; les deux répondent au ping.

Pour une machine virtuelle, l'ordre des arguments s'inverse, et `--unique` régénère la MAC :

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
qmrestore 'pbs-backup:backup/vm/101/<date>' 151 --storage local-lvm --unique 1
# restaurer la VM 101 sous l'ID 151, nouvelle adresse MAC
```

Résultat attendu : `progress 100% (read 21474836480 bytes, zeroes = 92% …, duration 4 sec)`, puis une adresse MAC neuve sur `net0` (`virtio=BC:24:11:C1:0B:A6`, contre `BC:24:11:0D:6D:68` pour l'original). Les zones vides du disque (92 %) ne sont pas réécrites : 20 Gio restaurés en 4,2 s. L'adresse IP, elle, vient de cloud-init : à changer aussi avant le démarrage.

Ou : votre nœud → `pbs-backup` → onglet **Backups** → la sauvegarde → **Restore**, en changeant l'ID et en cochant **Unique**.

Étape 2 — Restaurer un seul fichier

Pour un conteneur, la sauvegarde est une archive de fichiers, `root.pxar`. `proxmox-backup-client`, présent sur chaque nœud, la monte comme un répertoire.

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
# Le dépôt : utilisateur@serveur:datastore. Mot de passe et empreinte par
variables.
export PBS_REPOSITORY='root@pam@10.<site>.<N>.14:demo-store'
export PBS_PASSWORD='proxmoxx'   # le mot de passe
export PBS_FINGERPRINT='<empreinte>'   # empreinte du certificat de PBS
```

Ce support est la propriété de Dawan France.

```
# --ns : vos sauvegardes sont dans le namespace de votre nœud 1 (atelier 3)
proxmox-backup-client snapshot list --ns pxe2<N>1   # les sauvegardes du
namespace

mkdir -p /mnt/pbs-restore   # le point de montage
proxmox-backup-client mount --ns pxe2<N>1 'ct/201/<date>' root.pxar /mnt/pbs-
restore   # monter l'archive du conteneur
ls /mnt/pbs-restore/   # son contenu
cp /mnt/pbs-restore/etc/hostname /tmp/hostname-restore   # copier un fichier
cat /tmp/hostname-restore   # le fichier restauré

# Un umount ordinaire : le client n'a pas de sous-commande umount
umount /mnt/pbs-restore
```

Résultat attendu : la liste des sauvegardes (`ct/200/…` , `ct/201/…`), puis `FUSE library version: 3.17.2` , la racine du conteneur (`bin boot dev etc home …`), et `ct-sdn-test` . `proxmox-backup-client umount` répond bien `Error: no such command 'umount'` .

Sans `--ns` , la liste ne montre que la racine du datastore : vide chez vous, puisque tout est dans `pxe2<N>1` . Sur le lab : `snapshot list` seul affichait la sauvegarde de la racine, `snapshot list --ns pxe221` celle du namespace, et `mount --ns pxe221` a monté son `root.pxar` .

Pour une **machine virtuelle**, la sauvegarde est une image disque : on passe par `proxmox-backup-client map` (cours, chapitre 5).

Validation finale

Dans le Shell du nœud (interface web → votre nœud → **Shell**) :

```
pct status 210   # le conteneur restauré
cat /tmp/hostname-restore   # le fichier restauré
```

Résultat attendu : `status: running` , puis `ct-sdn-test` .

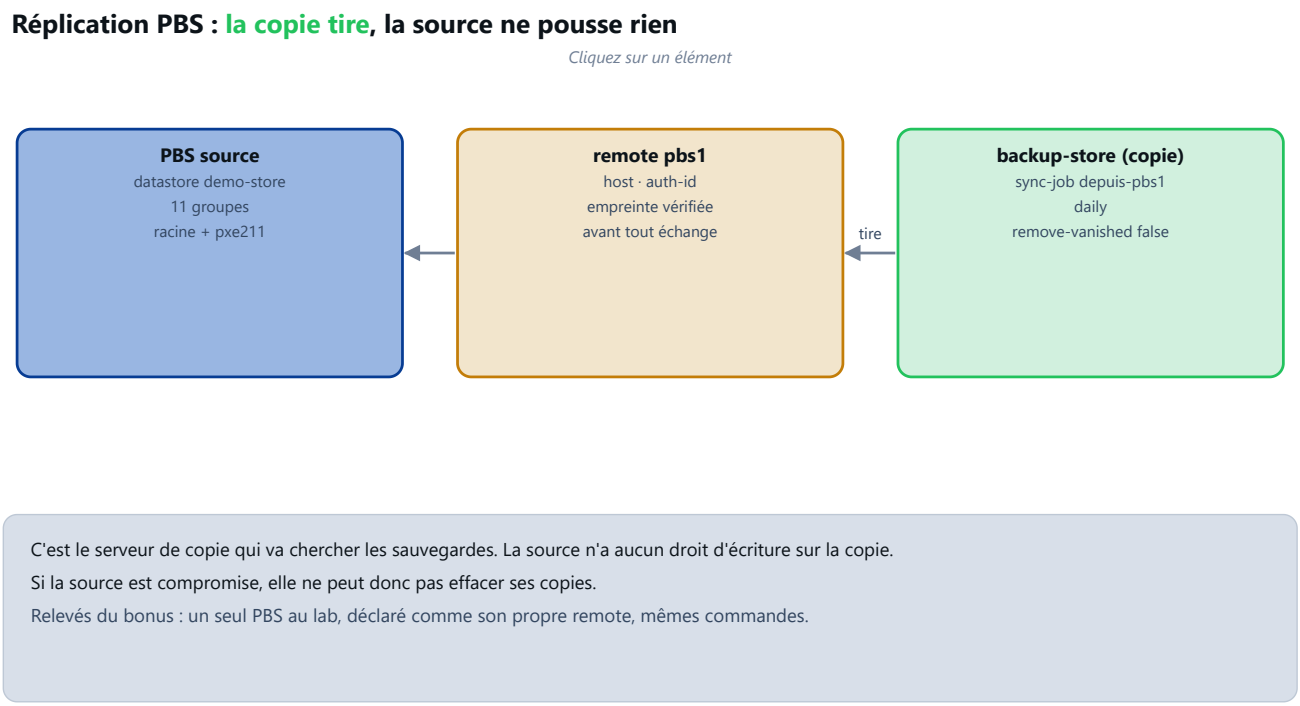
Atelier 6 — Réplication inter-PBS

Ce support est la propriété de Dawan France.

Contexte

Un seul PBS reste un point de défaillance unique : son disque tombe, toutes les sauvegardes partent avec lui. La réplication recopie les sauvegardes sur un second PBS, dans une autre salle ou un autre site. C'est le **second serveur qui tire** les sauvegardes du premier : le serveur principal n'a aucun droit d'écriture sur la copie, ce qui la protège s'il est compromis.

Résultat à obtenir : les sauvegardes de `demo-store` recopiées dans un datastore `backup-store`, par un job de synchronisation quotidien.



Qui tire quoi, et pourquoi la source n'a aucun droit sur la copie. Cliquez sur un élément pour lire le résultat obtenu.

Arborescence attendue

```
PBS source                          PBS de copie
└── demo-store   ─── tiré par ──►   ├── remote « pbs1 »        ← connexion vers la source
                                    │
                                    ├── sync-job depuis-pbs1    (quotidien)
                                    └── backup-store            ← la copie
```

Ce que vous devez faire

Ce support est la propriété de Dawan France.

1. **Sur le PBS de copie, déclarer la source comme *remote***, avec son empreinte.
2. **Créer un job de synchronisation** qui tire le datastore du voisin (`demo-store-s<N>`) vers `backup-store` , **dans un namespace au nom du voisin** (`voisin-s<N>`), sans supprimer les sauvegardes disparues.
3. **Lancer une première synchronisation** et vérifier la copie.

Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> **Avec un seul PBS par stagiaire**, chacun déclare en remote, sur son PBS, **les PBS des stagiaires de la même ville de lab**. Le réseau interne d'un autre lab n'est pas joignable : `--host` est l'adresse de l'hyperviseur du voisin et `--port` son port `82M4`, qui mène à son PBS.
>
> Les adresses sont dans l'annexe « Accès au lab ». Seul dans votre ville, déclarez **votre propre PBS** (son adresse `.14`, sans `--port`) et tirez vers un second datastore. C'est ce qui a été fait sur le lab, avec exactement les mêmes commandes.

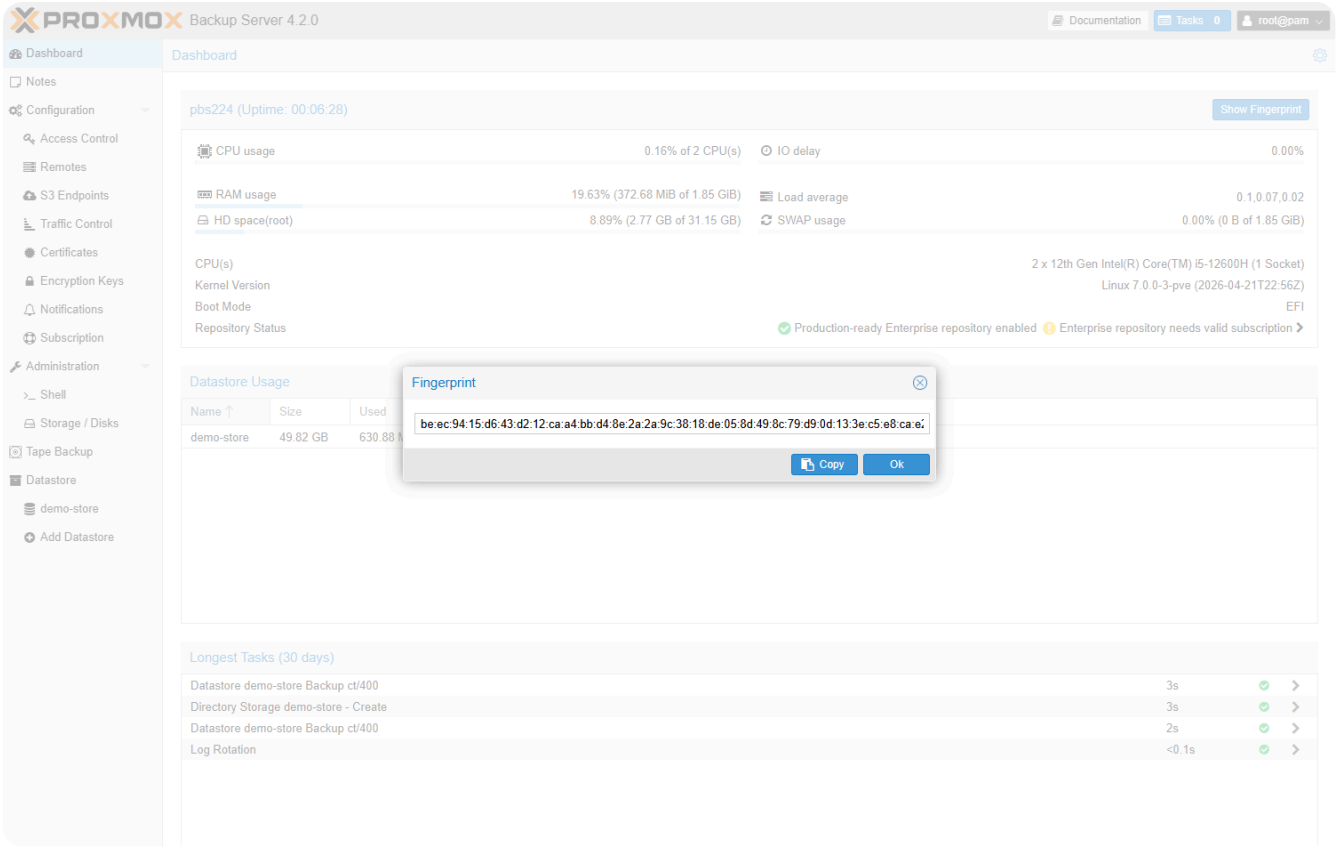
> **L'empreinte est celle du PBS source** (le voisin), lue dans son PBS : **Dashboard** → **Show Fingerprint**. Collez la vraie valeur : `<empreinte-source>` tel quel est refusé (`value does not match the regex pattern`). `remote create` ne teste pas la connexion : une adresse injoignable n'apparaît qu'au `pull`.

Le PBS d'un voisin, par l'interface

Entre deux labs isolés comme ceux des stagiaires. Le PBS du lab 1 (`10.59.1.14`) tire celui du lab 2 (`10.59.2.14`), joignable seulement par l'hôte `192.168.1.100`, port `8224`. L'accès direct à `10.59.2.14` échoue (`curl` : délai dépassé), l'accès par `192.168.1.100:8224` répond.

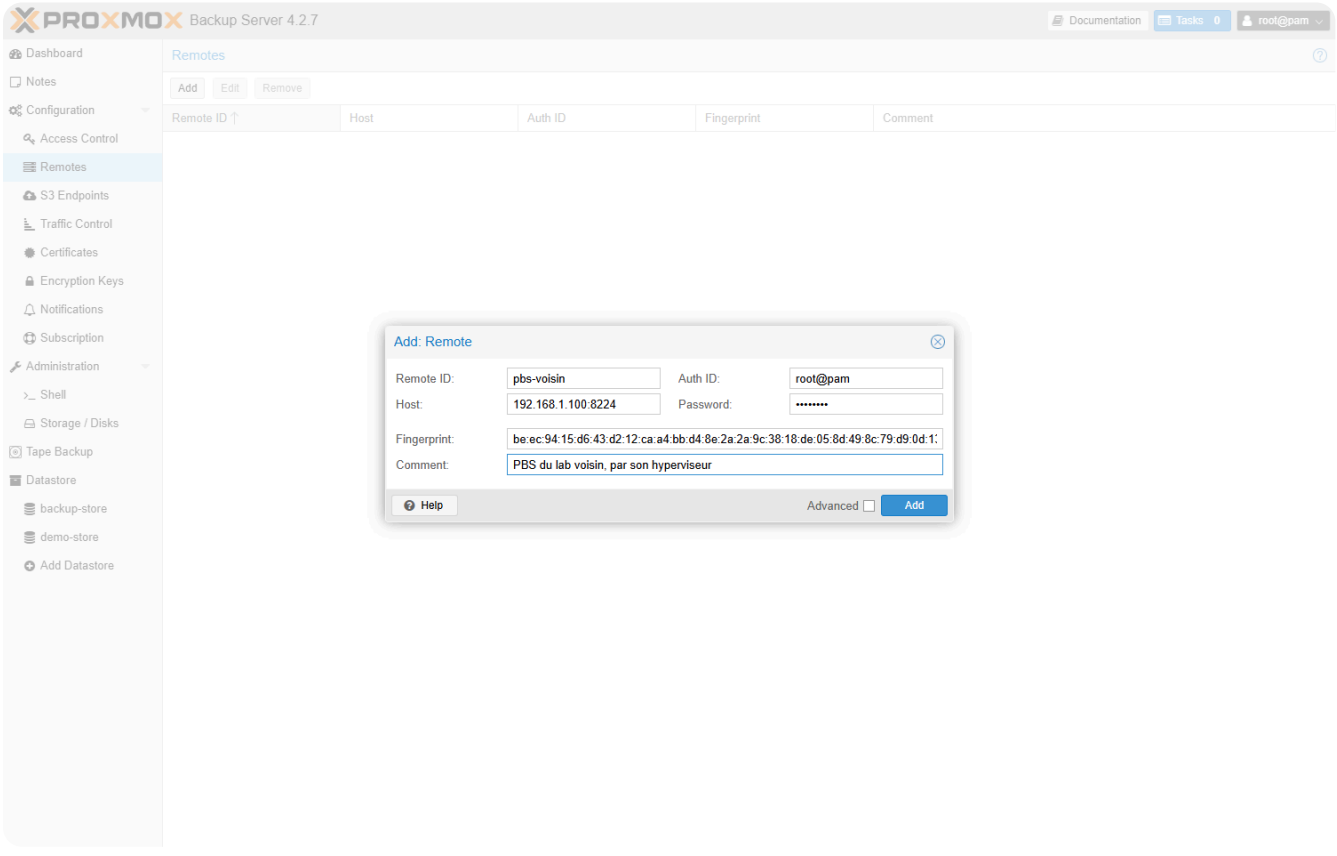
Chez le voisin : **Dashboard** → **Show Fingerprint** → **Copy**. Il vous envoie la valeur.

Ce support est la propriété de Dawan France.



Le datastore du voisin contient la sauvegarde `ct/400` (tâches en bas).

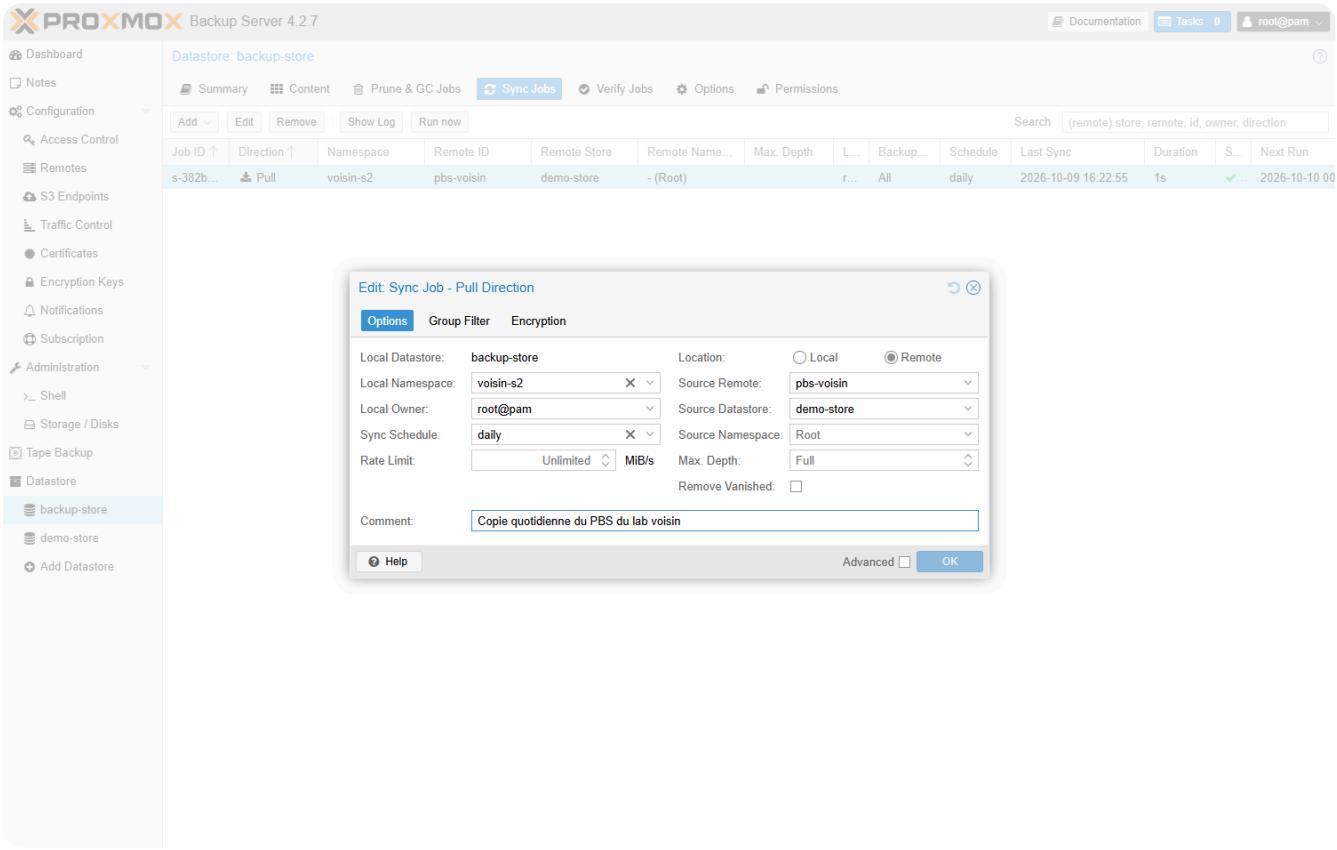
Sur votre PBS : **Configuration → Remotes → Add**.



Ce support est la propriété de Dawan France.

Champ	Valeur
Remote ID	un nom au choix : `pbs-voisin` (ou `pbs-s<N>`)
Host	**l'hyperviseur et le port du voisin dans le même champ** : `192.168.1.100:8224` sur le lab, `192.168.44.203:8234` pour le stagiaire 2 de Nantes
Auth ID / Password	`root@pam` / `proxmoxx`
Fingerprint	celle du voisin

Datastore → `backup-store` → onglet **Sync Jobs** → **Add** → **Add Pull Sync Job**. La liste **Source Datastore** est lue chez le voisin : si elle se remplit, la connexion passe.



Champ	Valeur
Local Namespace	**un namespace au nom du voisin** (`voisin-s2`) : la copie de chaque voisin reste à part
Sync Schedule	`daily`
Source Remote / Source Datastore	`pbs-voisin` / le datastore du voisin
Remove Vanished	décoché

Ce support est la propriété de Dawan France.

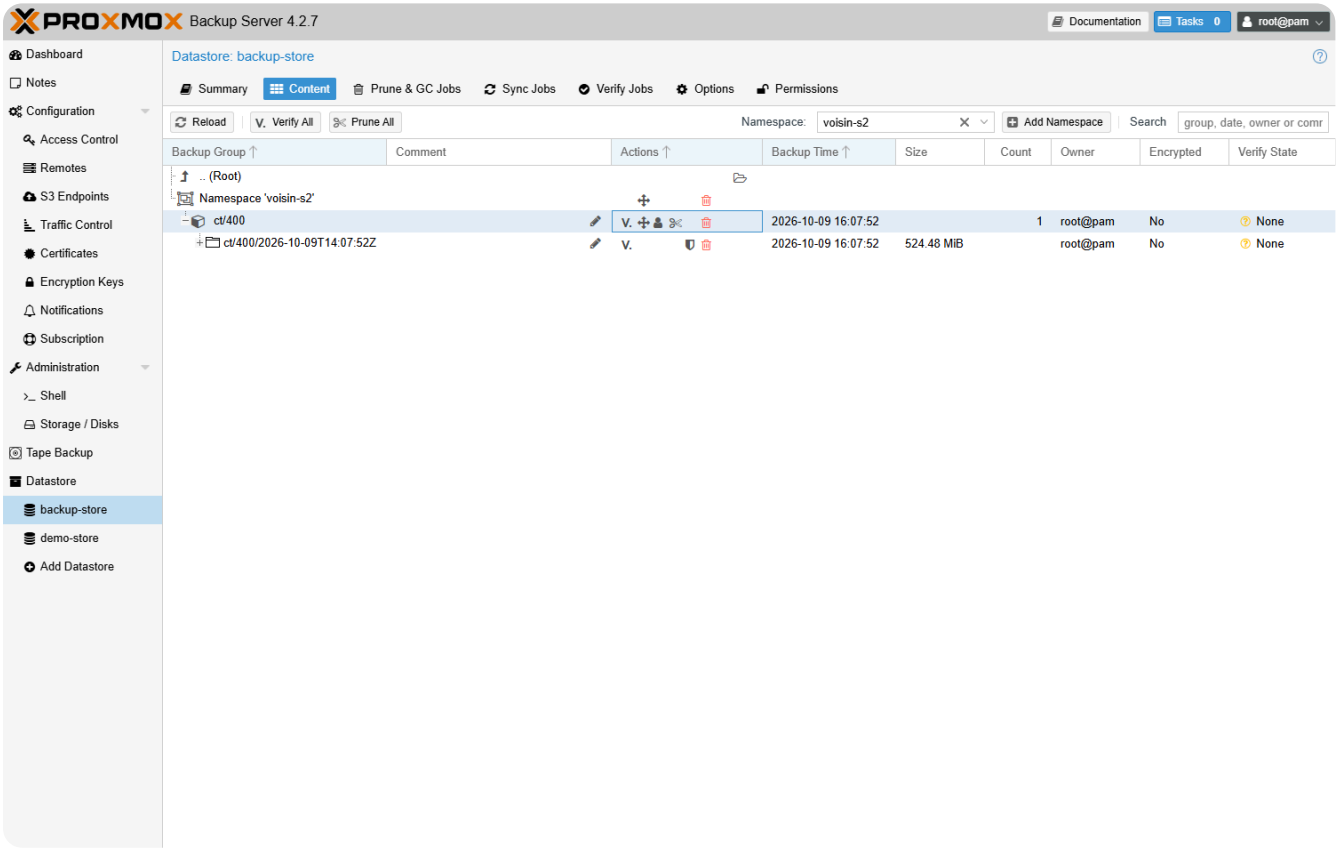
Sélectionnez le job → **Run now**. L'équivalent en ligne de commande :

Dans le Shell de votre PBS (interface web de PBS → **Administration** → **Shell**) :

```
# --ns : le namespace local qui reçoit la copie (créé s'il n'existe pas)
proxmox-backup-manager pull pbs-voisin demo-store backup-store --ns voisin-s2
```

Résultat attendu :

```
Syncing datastore 'demo-store', root namespace into datastore 'backup-store',
namespace 'voisin-s2'
Created namespace voisin-s2
Found 1 groups to sync (out of 1 total)
[ct/400]: 2026-10-09T14:07:52Z: sync done
pull datastore 'backup-store' end
TASK OK
```



Dans **Content**, la copie est rangée sous **Namespace 'voisin-s2'**, à part de vos propres sauvegardes.

Ce support est la propriété de Dawan France.

Étape 1 — Préparer la copie et déclarer la source

Sur le PBS de copie, un datastore de destination. Sur le lab, le datastore va sur le disque système, faute de second disque :

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
proxmox-backup-manager datastore create backup-store /var/lib/backup-store   # le datastore qui recevra la copie
```

Résultat attendu : `Access time update check successful.` puis `TASK OK`.

Configuration → **Remotes** → **Add** : ID `pbs1`, Host (l'adresse de la source), Auth ID `root@pam`, Password, Fingerprint (celle de la source).

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
# Sur la source : son empreinte
proxmox-backup-manager cert info | grep -i fingerprint
# Sur la copie : déclarer la source
# Voisin : --host <hyperviseur du voisin> --port <son port 82M4> ; seul : --host <votre adresse .14>
proxmox-backup-manager remote create pbs1 --host <adresse-source> --port <port-source> --auth-id root@pam \
  --password '<mot-de-passe>' --fingerprint '<empreinte-source>'
proxmox-backup-manager remote list   # le serveur distant enregistré
```

Résultat attendu (le PBS du lab déclaré comme sa propre source) : une ligne `pbs1 | 10.59.1.14 | root@pam | cd:d6:5b:92:…`.

L'empreinte protège de l'homme du milieu : la copie vérifie le certificat de la source avant tout échange. En production, on utilise un jeton limité au rôle `DatastoreReader` plutôt que `root@pam`.

Étape 2 — Le job de synchronisation

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

Ce support est la propriété de Dawan France.

```
# --store : le datastore local (la copie) ; --remote-store : celui de la source
# --remove-vanished false : une sauvegarde supprimée à la source reste dans la
copie
proxmox-backup-manager sync-job create depuis-pbs1 --store backup-store --remote
pbs1 \
  --remote-store demo-store --remove-vanished false --schedule daily
proxmox-backup-manager sync-job list   # le sync-job enregistré

# Première synchronisation, sans attendre la planification
proxmox-backup-manager pull pbs1 demo-store backup-store
```

Dans l'interface : **Datastore** → `backup-store` → **Sync Jobs** → **Run now**.

Résultat attendu (fin de la tâche, 30 secondes) :

```
[vm/300]: 2026-10-07T13:17:03Z: sync done
[vm/300]: percentage done: 100.00% (12/12 groups)
Finished syncing root namespace, current progress: 12 groups, 0 snapshots
pull datastore 'backup-store' end
TASK OK
```

12 groupes recopiés : `ls /var/lib/backup-store/vm` listait `100 101 103 105 106 107 110 290 300`, et `ls /var/lib/backup-store/ct` listait `200 201 250`. Le nombre de groupes dépend des machines sauvegardées aux ateliers 3 et 4. Le job copie aussi les namespaces de la source (`pxe2<N>1` ...), sous le namespace local choisi.

> **`--remove-vanished false`** pour débuter : une suppression accidentelle à la source laisse la sauvegarde intacte dans la copie. Passez à `true` seulement une fois votre rétention maîtrisée.

Validation finale

Dans le Shell de PBS (interface web de PBS → **Administration** → **Shell**) :

```
proxmox-backup-manager remote list   # le serveur distant
proxmox-backup-manager sync-job list   # le sync-job
```

Ce support est la propriété de Dawan France.

Résultat attendu : le remote `pbs1`, puis le job `depuis-pbs1` : `backup-store`, `pbs1`, `demo-store`, `daily`.

Atelier 7 — Fil rouge : mettre un service en production, puis livrer l'infrastructure

Contexte

Cinq chapitres ont construit, pièce par pièce, une infrastructure complète. On y trouve un nœud installé et mis à jour, des comptes nominatifs, un conteneur de service, des VM tirées d'un template, du stockage ZFS. S'y ajoutent un cluster de trois nœuds avec Ceph et la haute disponibilité, un pare-feu, des sauvegardes sur PBS et leur copie chez un voisin.

Ce dernier atelier s'en sert deux fois.

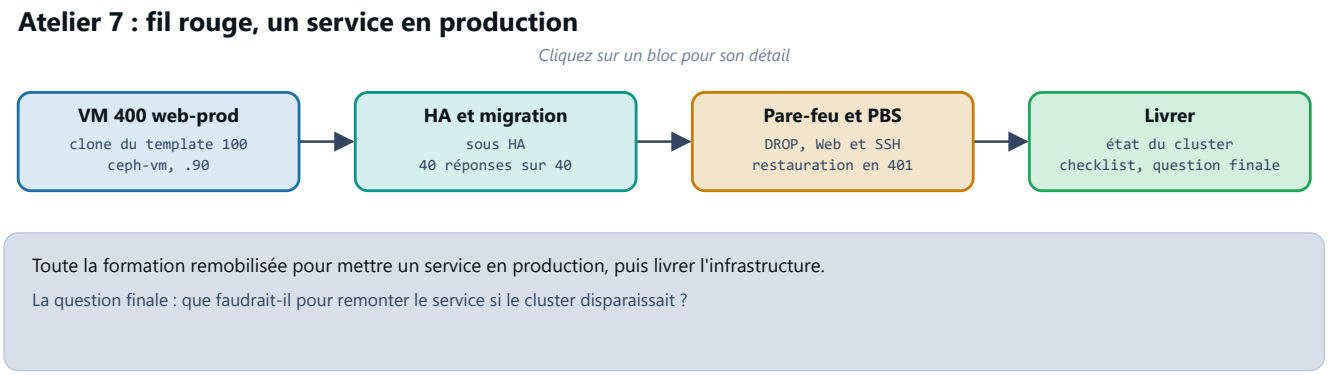
Partie A : une demande arrive, un serveur web à mettre en production. Vous la traitez de bout en bout, avec un outil de chaque chapitre. Il faut un compte pour l'équipe qui l'exploitera et une VM tirée du template, posée sur Ceph, sous HA, filtrée par le pare-feu, sauvegardée, et dont la restauration est prouvée.

Partie B : vous passez l'ensemble en revue **comme si vous le livriez à un client**. Chaque brique est vérifiée par une commande, chaque machine documentée, et les points faibles sont nommés avant la remise.

Résultat à obtenir : la VM 400 `web-prod` qui sert sa page en `10.<site>.<N>.90`, sur `ceph-vm`, sous HA, filtrée (Web et SSH seulement), sauvegardée et restaurée à l'essai. Un groupe `exploitation` n'a de droits que sur elle.

Puis les dix points de la checklist sont vérifiés, avec une description sur chaque machine qui reste, les tâches en échec comprises. Une réponse écrite est donnée à la question de l'étape 11.

Ce support est la propriété de Dawan France.



Le schéma suit la VM 400, de sa création jusqu'à la livraison. Cliquez sur chaque bloc pour son détail.

Arborescence attendue

```
Infrastructure livrée
├── Nœuds pxe2<N>1 à pxe2<N>3   ← cluster-tp, quorate ; dépôt no-subscription, à jour
├── Ceph                        ← HEALTH_OK ; pool ceph-vm en 3/2, sous 80 %
├── Machines                    ← CT 200 cache-apt, VM 101 et 102, VM 300 sous HA ; chacune décrite
│   └── VM 400 web-prod         ← 10.<site>.<N>.90, nginx + agent QEMU, disque sur ceph-vm, sous HA (étapes 2 à 5)
├── Comptes                     ← groupe exploitation (exploitant@pve) : PVEVMUser sur /vms/400 seulement (étapes 1 et 2)
├── Pare-feu                    ← Datacenter réactivé ; VM 400 : Web et SSH acceptés, le reste jeté (étape 6)
└── PBS                         ← job planifié, sauvegarde de la VM 400, restauration prouvée sur une copie 401 supprimée (étape 7)
```

Ce que vous devez faire

Partie A — mettre un service en production

1. **Préparer l'accès de l'équipe d'exploitation** : un groupe `exploitation`, un compte `exploitant@pve` qui en est membre ; le droit `PVEVMUser` sera donné au groupe, sur la seule VM 400.
2. **Créer la VM 400** `web-prod` : clone complet du template 100, disque sur `ceph-vm`, adresse fixe `10.<site>.<N>.90` par cloud-init ; la faire porter par le nœud 3 avant de la démarrer.

Ce support est la propriété de Dawan France.

3. **Installer le service** dans la VM : `nginx` et l'agent QEMU ; vérifier depuis le nœud que la page répond et que l'agent est joignable.
4. **Placer la VM sous haute disponibilité**, avec les mêmes limites que la VM 300.
5. **La déplacer à chaud** vers le nœud 2 par la HA, pendant qu'une boucle interroge la page chaque seconde : le service ne doit pas s'interrompre.
6. **La filtrer** : pare-feu de la VM avec une politique d'entrée `DROP`, le Web et SSH acceptés, la case **Firewall** sur sa carte. Puis **réactiver le pare-feu du Datacenter**, avec un filet automatique comme à l'atelier 1. Constater ce qui passe et ce qui ne passe plus.
7. **La sauvegarder sur PBS et prouver la restauration** : restaurer la sauvegarde sous le numéro 401, carte débranchée, vérifier que le service y tourne, puis supprimer la copie.

Partie B — livrer l'infrastructure

8. **Relever l'état général** du cluster : versions, quorum, Ceph, stockages, HA.
9. **Chercher les tâches en échec** parmi les trente dernières du nœud 1, et ouvrir le journal de chacune.
10. **Documenter chaque machine** qui reste, puis **dérouler la checklist**, point par point, avec la commande ou l'écran qui le prouve.
11. **Répondre par écrit** à la question finale : si le cluster disparaissait cette nuit, que faudrait-il pour remonter le service demain matin ?

> 💡 Une vérification qui échoue n'est pas un échec de l'atelier : c'est exactement ce qu'une revue de livraison sert à trouver. Notez-la, corrigez-la, puis recommencez la vérification.

Dawan
Nous formons votre avenir
Ce support est la propriété de Dawan France.

▼ **Cliquez pour vous aider**

> Les sorties viennent du lab d'exemple (`10.59.1.x` : site 59, stagiaire 1 ; nœuds `pxe211` à `pxe213`). Chez vous, l'adresse de la VM 400 est `10.<site>.<N>.90` et les nœuds `pxe2<N>1` à `pxe2<N>3` .

Partie A — Mettre un service en production

Étape 1 — Le groupe et le compte de l'équipe d'exploitation

On prépare l'accès **avant** de créer la machine : le jour de la livraison, l'équipe d'exploitation se connecte avec son propre compte, jamais avec `root` . Comme au chapitre 1, le droit va au **groupe** : un nouvel arrivant n'aura qu'à y être ajouté.

Par l'interface : Datacenter → Permissions → Groups → Create : `exploitation` . Puis **Users → Add** : `exploitant` , realm **Proxmox VE authentication server**, groupe `exploitation` , mot de passe `proxmoxx` .

Dans le Shell du nœud 1 :

```
pveum group add exploitation --comment "Exploitation des services livrés"   # le groupe
pveum user add exploitant@pve --password 'proxmoxx' --groups exploitation \
  --comment "Exploitation du serveur web"                                    # le compte, membre du groupe
pveum group list | grep exploitation                                         # vérifier
```

Résultat attendu : aucune sortie pour les deux créations, puis une ligne `exploitation …` `exploitant@pve` . Le droit sera posé à l'étape 2, une fois la VM 400 créée.

Étape 2 — Créer la VM 400 sur Ceph, et la confier au nœud 3

Le template 100 est sur le `local-lvm` du nœud 1. Un clone **complet** peut écrire son disque ailleurs : sur `ceph-vm` , la VM devient migrable à chaud et peut être relancée par la HA sur n'importe quel nœud.

Ce support est la propriété de Dawan France.

Par l'interface : clic droit sur le template 100 → **Clone** : **VM ID** `400`, **Name** `web-prod`, **Mode** `Full Clone`, **Target Storage** `ceph-vm` → **Clone**. Puis VM 400 → **Cloud-Init** → **IP Config (net0)** : `10.<site>.<N>.90/24`, passerelle `10.<site>.<N>.1`. Puis **Migrate** vers `pxe2<N>3`, et **Start**.

Le droit : VM 400 → **Permissions** → **Add** → **Group Permission** : groupe `exploitation`, rôle `PVEVMUser`.

Dans le Shell du nœud 1 :

```
qm clone 100 400 --name web-prod --full --storage ceph-vm        # clone complet, disque sur Ceph
qm set 400 --ipconfig0 ip=10.<site>.<N>.90/24,gw=10.<site>.<N>.1 \
  --description "Serveur web nginx du fil rouge - chapitre 5"     # adresse fixe et description
qm migrate 400 pxe2<N>3                                          # VM éteinte : migration à froid
pveum acl modify /vms/400 --groups exploitation --roles PVEVMUser   # le droit du groupe de l'étape 1
```

Puis **dans le Shell du nœud 3** (la VM y est désormais) :

```
qm start 400    # premier démarrage : cloud-init applique l'adresse
qm status 400   # la VM tourne-t-elle ?
```

Résultat attendu : le clone recopie les 20 Gio du disque (`transferred 20.0 GiB of 20.0 GiB (100.00%)`), puis `update VM 400: -description … -ipconfig0 ip=10.59.1.90/24,gw=10.59.1.1` , puis :

```
use dedicated network address for sending migration traffic (10.59.41.13)
starting migration of VM 400 to node 'pxe213' (10.59.41.13)
migration finished successfully (duration 00:00:01)
```

Une seconde de migration pour une VM de 20 Gio : rien n'a été copié. Le disque est sur Ceph, que les trois nœuds voient ; seule la configuration a changé de dossier dans `/etc/pve`. Au démarrage, `generating cloud-init ISO`, puis `status: running`.

Vérifier le droit du groupe. Dans le Shell du nœud 1 :

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

```
pveum user permissions exploitant@pve --path /vms/400   # ce que le compte peut faire sur la VM 400
pveum user permissions exploitant@pve --path /vms/101   # et sur une autre VM
```

Résultat attendu : sur `/vms/400`, dix privilèges, de `VM.Audit` à `VM.PowerMgmt` (console, démarrage et arrêt, sauvegarde, lecture par l'agent) ; sur `/vms/101`, une ligne vide : aucun droit. Le compte voit et pilote son serveur, et rien d'autre.

Étape 3 — Installer le service et l'agent

L'image cloud Debian n'a ni serveur web ni agent QEMU (chapitre 2, atelier 8). On les installe par SSH : la clé du nœud 1 a été déposée dans le template par cloud-init.

Dans le Shell du nœud 1 :

```
ssh devops@10.<site>.<N>.90    # accepter l'empreinte (yes) au premier essai
```

Dans la VM 400 :

```
cloud-init status --wait                                    # attendre la fin du premier démarrage (sinon apt est verrouillé)
sudo apt-get update                                         # la liste des paquets
sudo DEBIAN_FRONTEND=noninteractive apt-get install -y nginx qemu-guest-agent   # le service et l'agent
sudo systemctl start qemu-guest-agent                       # l'agent ne démarre pas seul à l'installation
systemctl is-active nginx qemu-guest-agent                  # deux services actifs ?
exit                                                        # revenir sur le nœud
```

Résultat attendu : `status: done`, `Setting up nginx (1.26.3-3+deb13u9) ...`, puis `active` et `active`.

De retour dans le Shell du nœud 1, la page, puis l'agent (sur le nœud 3, qui porte la VM) :

```
curl -s -o /dev/null -w "%{http_code}\n" http://10.<site>.<N>.90/   # le code de réponse HTTP
curl -s http://10.<site>.<N>.90/ | grep -o "<title>.*</title>"      # le titre de
```

Ce support est la propriété de Dawan France.

```
la page servie
ssh 10.<site>.<N>.13 qm agent 400 ping                                  # l'agent
répond-il ?
echo "code : $?"                                                          # 0 : oui
```

Résultat attendu : `200`, `<title>Welcome to nginx!</title>`, puis `code : 0`. Dans l'interface, l'onglet **Summary** de la VM 400 affiche maintenant son adresse sur la ligne **IPs**.

Étape 4 — Sous haute disponibilité

Par l'interface : Datacenter → HA → Resources → Add : **VM** `400`, **Max. Restart** `3`, **Max. Relocate** `1`, **Request State** `started`.

Dans le Shell du nœud 1 :

```
ha-manager add vm:400 --state started --max_restart 3 --max_relocate 1   # comme
la VM 300 au chapitre 4
ha-manager status   # les LRM et les ressources ; à lire : lrm, service
```

Résultat attendu, une quinzaine de secondes après :

```
lrm pxe211 (active, watchdog active, …)
lrm pxe212 (idle, watchdog standby, …)
lrm pxe213 (active, watchdog active, …)
service vm:300 (pxe211, started)
service vm:400 (pxe213, starting)
```

Le LRM de `pxe213` est `active` : il porte une ressource HA, son chien de garde est armé. Quelques secondes plus tard, `service vm:400 (pxe213, started)`.

Étape 5 — Un déplacement à chaud, sans coupure du service

Une VM sous HA se déplace **par la HA**, pas par `qm migrate` : c'est le gestionnaire qui décide, et il garde la trace de l'opération. Pour prouver que la VM ne s'arrête pas, un `ping` tourne en arrière-plan pendant le déplacement : une réponse par seconde, quarante en tout.

Dans le Shell du nœud 1 :

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

```
ping -c 40 -i 1 10.<site>.<N>.90 > /tmp/ping400.txt &   # 40 pings, un par seconde, en arrière-plan
ha-manager crm-command migrate vm:400 pxe2<N>2          # demander à la HA de la déplacer vers le nœud 2
```

Dans le Shell du nœud 1, quarante-cinq secondes plus tard, une fois les pings terminés :

```
tail -2 /tmp/ping400.txt                          # le bilan du ping
ha-manager status | grep vm:400                   # où est la VM maintenant ?
```

Résultat attendu : `40 packets transmitted, 40 received, 0% packet loss`, puis `service vm:400 (pxe212, started)`. Le journal de la migration, côté nœud 3 (**Datacenter** → **Tasks**, ligne `VM 400 - Migrate`) :

```
starting migration of VM 400 to node 'pxe212' (10.59.41.12)
migration downtime limit: 100 ms
average migration speed: 2.0 GiB/s - downtime 12 ms
migration finished successfully (duration 00:00:06)
```

Douze millisecondes d'arrêt : moins que l'intervalle entre deux pings. C'est ce que permettent le disque sur Ceph et le réseau de migration dédié (`10.59.41.x`) posé au chapitre 4.

Étape 6 — Filtrer la VM, puis réactiver le pare-feu du Datacenter

À l'atelier 1, le pare-feu du Datacenter a protégé les **nœuds**. Ici, c'est la **VM** qu'on filtre : la règle se pose dans le pare-feu de l'invité. Trois interrupteurs doivent être levés (cours §2.2) : le pare-feu du Datacenter, celui de la VM, et la case **Firewall** de sa carte.

Par l'interface : VM 400 → **Firewall** → **Add** : **Direction** `in`, **Action** `ACCEPT`, **Macro** `Web`, **Enable** coché ; idem avec la macro `SSH`. Puis **Firewall** → **Options** : **Firewall** `Yes`, **Input Policy** `DROP`. Puis **Hardware** → **Network Device (net0)** → cocher **Firewall**.

Dans le Shell du nœud 1 (l'API est la même depuis n'importe quel nœud ; le chemin nomme le nœud qui porte la VM) :

```
# Les règles d'abord, puis la politique : on n'écrit jamais un DROP avant ses exceptions
```

Ce support est la propriété de Dawan France.

```
pvesh create /nodes/pxe2<N>2/qemu/400/firewall/rules --type in --action ACCEPT --
macro Web --enable 1   # accepter le Web (80, 443)
pvesh create /nodes/pxe2<N>2/qemu/400/firewall/rules --type in --action ACCEPT --
macro SSH --enable 1   # accepter SSH (22)
pvesh set /nodes/pxe2<N>2/qemu/400/firewall/options --enable 1 --policy_in DROP
# pare-feu de la VM actif, le reste jeté
cat /etc/pve/firewall/400.fw                    # le fichier écrit par ces trois
commandes
ping -c 2 -W 1 -q 10.<site>.<N>.90             # le ping passe-t-il ? -q : le
bilan seul
```

Résultat attendu :

```
[OPTIONS]

policy_in: DROP
enable: 1

[RULES]

IN SSH(ACCEPT)
IN Web(ACCEPT)

2 packets transmitted, 2 received, 0% packet loss, time 1041ms
```

La règle `SSH`, créée en second, est arrivée **en tête** (cours §2.3). Et **le ping passe** : rien n'est encore filtré, deux interrupteurs sur trois manquent.

La case Firewall de la carte, en gardant son adresse MAC (même méthode qu'au chapitre 2, atelier 9). **Dans le Shell du nœud 2**, qui porte la VM :

```
qm config 400 | grep ^net0                  # la carte actuelle, adresse MAC
comprise
qm set 400 --net0 virtio=BC:24:11:D1:C6:7E,bridge=vmbr0,firewall=1   # la même
carte, case cochée : l'adresse MAC recopiée de la ligne précédente
qm config 400 | grep ^net0                  # vérifier
```

Ce support est la propriété de Dawan France.

Résultat attendu : `update VM 400: -net0 virtio=BC:24:11:D1:C6:7E,bridge=vmbr0,firewall=1`, puis la même ligne dans la configuration, appliquée à chaud (`qm pending 400` montre `cur net0: …,firewall=1`, sans valeur en attente). Le ping passe **toujours** : le pare-feu du Datacenter est désactivé depuis la fin de l'atelier 1, et sans lui, aucune règle d'invité ne s'applique.

Le pare-feu du Datacenter, avec le filet de l'atelier 1. Ses règles (22, 8006, Ceph) sont toujours en place. **Dans le Shell du nœud 1** :

```
grep -E "^IN" /etc/pve/firewall/cluster.fw   # les règles de l'atelier 1 sont-elles toujours là ?
systemd-run --on-active=5min --unit=filet-parefeu pvesh set /cluster/firewall/options --enable 0   # le filet
pvesh set /cluster/firewall/options --enable 1   # activer
pve-firewall status                              # l'état du pare-feu
```

Résultat attendu : les quatre règles `IN Ceph(ACCEPT) -source 10.59.21.0/24`, `IN Ceph(ACCEPT) -source 10.59.31.0/24`, `IN ACCEPT -p tcp -dport 8006`, `IN ACCEPT -p tcp -dport 22`, puis `Running timer as unit: filet-parefeu.timer` et `Status: enabled/running`.

Constater, puis annuler le filet. **Dans le Shell du nœud 1** :

```
ping -c 2 -W 1 -q 10.<site>.<N>.90                                  # le ping, son bilan
curl -s -m 3 -o /dev/null -w "http %{http_code}\n" http://10.<site>.<N>.90/  # le Web
ssh -o ConnectTimeout=5 devops@10.<site>.<N>.90 hostname            # SSH
curl -s -m 5 http://10.<site>.<N>.90:3142                           # un port sans règle
echo "code $?"                                                      # 28 : délai dépassé
curl -sk -m 3 -o /dev/null -w "gui %{http_code}\n" https://10.<site>.<N>.11:8006
# l'interface du nœud
ceph health                                                         # Ceph a-t-il souffert ?
```

Ce support est la propriété de Dawan France.

```
systemctl stop filet-parefeu.timer                                    #
tout va bien : annuler le filet
```

Résultat attendu :

```
2 packets transmitted, 0 received, 100% packet loss, time 1023ms
http 200
web-prod
code 28
gui 200
HEALTH_OK
```

Le ping et le port 3142 ne reçoivent **aucune réponse** (`code 28` : délai dépassé pour `curl`) : c'est la signature d'un `DROP` , qui jette sans répondre. Le Web et SSH passent, l'interface du nœud reste joignable et Ceph est en bonne santé : les règles posées à l'atelier 1 font leur travail. Cette fois, **le pare-feu reste actif** : c'est l'état d'une infrastructure livrée.

Étape 7 — Sauvegarder, puis prouver la restauration

Le job nocturne de l'atelier 4 sauvegarde **toutes** les machines (`all 1` dans `/etc/pve/jobs.cfg`) : la VM 400 y entre d'elle-même. Mais une sauvegarde dont on n'a jamais testé la restauration n'est qu'une hypothèse. On en fait une tout de suite, et on la restaure sur une copie.

Dans le Shell du nœud 2, qui porte la VM :

```
vzdump 400 --storage pbs-backup --mode snapshot   # sauvegarde à chaud, vers PBS
pvesm list pbs-backup --vmid 400                  # la sauvegarde, et son nom
exact
```

Résultat attendu :

```
INFO: creating Proxmox Backup Server archive 'vm/400/2026-10-16T13:12:34Z'
INFO: backup is sparse: 18.45 GiB (92%) total zero data
INFO: transferred 20.00 GiB in 6 seconds (3.3 GiB/s)
INFO: Finished Backup of VM 400 (00:00:07)
INFO: Backup job finished successfully
```

Ce support est la propriété de Dawan France.

```
Volid                                              Format  Type             Size VMID
pbs-backup:backup/vm/400/2026-10-16T13:12:34Z pbs-vm  backup    21474839418 400
```

92 % du disque est vide : PBS ne stocke que les morceaux qui contiennent des données (cours §3.2).

La restauration, sous le numéro 401, avec une nouvelle adresse MAC (`--unique 1`) et la **carte débranchée** avant le démarrage. La copie porte en effet la même adresse IP que la VM 400, qui tourne. Remplacez `<date>` par celle de votre sauvegarde.

Dans le Shell du nœud 2 :

```
qmrestore pbs-backup:backup/vm/400/<date> 401 --storage ceph-vm --unique 1   # restaurer sous l'ID 401
qm config 401 | grep ^net0                                                   # la carte de la copie, sa nouvelle adresse MAC
qm set 401 --name web-prod-test --net0 virtio=BC:24:11:99:06:56,bridge=vmbr0,link_down=1   # renommer, débrancher : l'adresse MAC recopiée
qm start 401                                                                 # démarrer la copie
```

Dans le Shell du nœud 2, une minute et demie plus tard, le temps que la copie démarre :

```
qm agent 401 ping                                                            # l'agent répond-il ?
echo "code : $?"                                                             # 0 : oui
qm guest exec 401 -- systemctl is-active nginx                               # le service tourne-t-il dans la copie ?
```

Résultat attendu : `restore image complete (bytes=21474836480, duration=4.48s, speed=4574.32MB/s)` , une carte `virtio=BC:24:11:99:06:56,…` (la VM 400 a `BC:24:11:D1:C6:7E`), `update VM 401: -name web-prod-test -net0 …,link_down=1` , l'agent qui répond au bout d'environ une minute et quinze secondes, puis :

```
{
   "exitcode" : 0,
```

Ce support est la propriété de Dawan France.

```
   "exited" : 1,
   "out-data" : "active\n",
   "out-truncated" : 0
}
```

La copie démarre, et nginx y tourne : la sauvegarde est **utilisable**, pas seulement présente. La restauration a aussi recréé le pare-feu de la VM (`/etc/pve/firewall/401.fw`) : les règles voyagent avec la sauvegarde.

Supprimer la copie. Dans le Shell du nœud 2 :

```
qm stop 401                # arrêter la copie
qm destroy 401 --purge     # la supprimer, avec ses références
qm list | grep -c " 401 "  # reste-t-il une VM 401 ?
```

Résultat attendu : `Removing image: 100% complete...done.` , `purging VM 401 from related configurations..` , puis `0` .

Partie B — Livrer l'infrastructure

Étape 8 — L'état général

Dans le Shell du nœud 1 (interface web → `pxe2<N>1` → **Shell**) :

```
pveversion            # la version de Proxmox VE du nœud
pvecm status          # le cluster : nombre de nœuds, quorum
ceph status           # la santé de Ceph
pvesm status          # tous les stockages déclarés et leur remplissage
ha-manager status     # les ressources sous haute disponibilité
```

Résultat attendu : la même version sur les trois nœuds (relancez `pveversion` sur `pxe2<N>2` et `pxe2<N>3` ; sur le lab, `pve-manager/9.2.21/4f6e0ac86f9e8c7f`). On attend aussi `Quorate: Yes` avec `Nodes: 3` , `HEALTH_OK` , tous les stockages `active` et sous 80 %, et deux ressources HA : `service vm:300 (…, started)` et `service vm:400 (…, started)` .

Étape 9 — Les tâches en échec

Dans le Shell du nœud 1 :

Ce support est la propriété de Dawan France.

```
# Les tâches du nœud qui ne se sont pas terminées par OK (erreurs et avertissements)
pvesh get /nodes/$(hostname)/tasks --limit 30 --errors 1
```

Résultat attendu : seize lignes, triées par numéro de machine. La dernière colonne, **Status**, dit pourquoi chacune n'est pas `OK` :

Status (extrait)	Ce que c'est
`WARNINGS: 1` (`vzcreate`, `vzstart` des CT 200 et 201)	Un avertissement à la création ou au démarrage d'un conteneur ; la tâche a abouti
`job errors` (`vzdump` de la VM 101)	Une sauvegarde manuelle qui a échoué
`broken pipe`, puis `VM is locked` (VM 196)	Une restauration d'essai interrompue, puis une suppression refusée tant que la VM était verrouillée
`command 'apt-get update' failed` (`aptupdate`)	La mise à jour de la liste des paquets du premier jour, avant le changement de dépôts du chapitre 1

La liste n'est donc **pas vide** sur un nœud qui a servi : `--errors 1` garde tout ce qui n'est pas `OK`, avertissements compris. Chaque ligne s'ouvre dans **Datacenter** → **Tasks** (double clic) : son journal dit ce qui s'est passé. Le travail de livraison consiste à **les comprendre une par une**, et à vérifier qu'aucune ne touche un service livré.

Étape 10 — Documenter chaque machine, puis dérouler la checklist

Par l'interface : la machine → **Summary** → bloc **Notes** → icône du crayon → saisir → **OK**.

Dans le Shell du nœud qui porte la machine :

```
# Une description par machine : ce qu'elle fait, d'où elle vient, depuis quand
pct set 200 --description "Cache de paquets apt-cacher-ng, port 3142 - chapitre 1"   # le conteneur du chapitre 1
qm set 101 --description "Serveur web, clone du template 100 (cloud-init) - chapitre 2"   # un clone du chapitre 2
qm set 300 --description "VM de test sur Ceph, ressource HA - chapitre 4"   # la VM sous HA

# Vérifier
```

Ce support est la propriété de Dawan France.

```
pct config 200 | grep description   # la description du conteneur
qm config 101 | grep description    # celle de la VM
```

Résultat attendu : `pct set` ne répond rien ; chaque `qm set` répond par une ligne `update VM … -description …`. Puis `description: Cache de paquets apt-cacher-ng, port 3142 - chapitre 1%0A` (le `%0A` final est le saut de ligne, encodé dans la configuration du conteneur) et `description: Serveur web, clone du template 100 (cloud-init) - chapitre 2`. La VM 400 a reçu la sienne à l'étape 2.

Une commande `qm set` sur une VM portée par un autre nœud répond qu'elle n'existe pas : `Configuration file 'nodes/pxe211/qemu-server/400.conf' does not exist`. Cela a été sur le lab, depuis le nœud 1, pour la VM 400 alors sur le nœud 2. Lancez-la sur le bon nœud, ou passez par l'interface, qui voit tout le cluster.

La checklist :

#	Point	Votre vérification	OK ?
1	Dépôt no-subscription, système à jour, sur les trois nœuds	`apt update` sans erreur 401 ; `apt list --upgradable` vide	
2	Aucun compte partagé ; droits donnés à des groupes	`pveum user list`, `pveum acl list` : le groupe `exploitation` sur `/vms/400`	
3	Double authentification sur les comptes d'administration	**Datacenter → Permissions → Two Factor**	
4	Cluster en quorum, trois nœuds	`pvecm status`	
5	Ceph `HEALTH_OK`, pool sous 80 %	`ceph status`, `ceph df`	
6	Les VM critiques sous HA	`ha-manager status` : `vm:300` et `vm:400`	
7	Pare-feu actif : Datacenter et VM 400	`pve-firewall status`, `cat /etc/pve/firewall/400.fw`	
8	Sauvegardes planifiées vers PBS, avec rétention	**Datacenter → Backup**	
9	Une restauration a été testée	étape 7 : la copie 401 a démarré, nginx `active`	
10	Une copie des sauvegardes hors du site	atelier 6 : le job de synchronisation	

Ce support est la propriété de Dawan France.

Le point 3 est souvent celui qui manque : au chapitre 1, le second facteur a été testé sur un compte d'essai, puis retiré. Sur une infrastructure livrée, chaque compte d'administration en porte un. Nommez l'écart dans votre réponse finale.

Étape 11 — La question qui conclut la formation

Si votre cluster disparaissait cette nuit, sauriez-vous remonter le service demain matin ?
Les éléments dont vous auriez besoin :

Élément	Où il est
Les sauvegardes	Sur PBS, machine séparée, et leur copie chez le voisin (atelier 6)
La procédure de restauration	Atelier 5, et l'étape 7 de cet atelier
La configuration du cluster	À reconstruire : installation, cluster, Ceph (chapitres 1 et 4), puis restauration des machines
Les règles du pare-feu des VM	Dans leurs sauvegardes : la restauration les recrée (étape 7)
Les mots de passe et l'empreinte du certificat de PBS	**Hors de l'infrastructure**, dans un gestionnaire de mots de passe

C'est le dernier point qui manque le plus souvent en entreprise : la sauvegarde est parfaite, mais les identifiants pour y accéder étaient sur le serveur disparu.

Validation finale

Contrôle	Attendu
Étapes 1 et 2	`pveum acl list` : `/vms/400 | PVEVMUser | group | exploitation` ; aucun droit ailleurs pour `exploitant@pve`
Étapes 2 et 3	`curl http://10.<site>.<N>.90/` → `200` ; `qm agent 400 ping` sans erreur
Étapes 4 et 5	`ha-manager status` : `service vm:400 (pxe2<N>2, started)` ; la boucle : quarante `200`
Étape 6	`pve-firewall status` → `enabled/running` ; ping vers la VM 400 jeté, Web et SSH acceptés ; `HEALTH_OK`
Étape 7	Une sauvegarde `vm/400` dans `pbs-backup` ; plus de VM 401
Étape 8	`Quorate: Yes`, `HEALTH_OK`, stockages `active`, `vm:300` et `vm:400` `started`

Ce support est la propriété de Dawan France.

Contrôle	Attendu
Étape 9	Chaque tâche de la liste comprise
Étape 10	Une description dans **Notes** pour le CT 200, les VM 101, 300 et 400 ; dix lignes cochées, ou les écarts nommés
Étape 11	Une réponse écrite, qui dit où sont rangés les mots de passe

---

Dépannage — Erreurs fréquentes

Ceph en `HEALTH_WARN`, PG `peering` bloqués, écritures figées après l'activation du pare-feu. Le trafic Ceph n'est pas couvert par le réseau local autorisé d'office. Remède : Règles `ACCEPT` avec la macro `Ceph` depuis les deux réseaux Ceph (atelier 1, étape 1).

Connexion TCP refusée qui devient un silence (`code 124`). Pare-feu actif : les refus (`RST`) sont jetés comme `INVALID`. Remède : Tester avec un service réellement à l'écoute.

`Create` grisé en créant un datastore PBS. Champ **Filesystem** vide. Remède : Choisir `ext4` ou `xfs`.

`Cannot find datastore … check permissions` avec un jeton PBS. Un jeton PBS n'a aucun droit par défaut. Remède : `proxmox-backup-manager acl update /datastore/<store> DatastoreAdmin --auth-id '<jeton>'`.

`backup owner check failed (<compte> != root@pam)`. Le groupe de sauvegardes appartient déjà à un autre compte. Remède : Créer le compte et son stockage avant la première sauvegarde de la source.

Namespace vide pour un compte restreint. `DatastoreBackup` ne montre que les sauvegardes du compte. Remède : Faire écrire la source avec ce compte (atelier 3).

CT restauré en conflit d'adresse avec l'original. La restauration reprend l'IP et la MAC. Remède : Changer `net0` avant de démarrer ; `--unique` pour une VM.

Règle IP Set inefficace après un moment. La VM a changé d'adresse. Remède : Mettre à jour l'IP Set, et fixer l'adresse de la VM.

Règles pare-feu d'un conteneur sans effet. Carte créée en CLI sans `firewall=1`. Remède : `pct set <id> --net0 …,firewall=1`, ou cocher **Firewall** sur la carte.

Ce support est la propriété de Dawan France.

`HEALTH_WARN` : `1/3 mons down`, **sans lien avec le pare-feu.** Le moniteur de ce nœud est désactivé et n'a pas démarré (`systemctl is-enabled ceph-mon@<nœud>` → `disabled` ; `journalctl -b -u ceph-mon@<nœud>` → `-- No entries --`), sur le lab. Remède : Sur ce nœud : `systemctl enable --now ceph-mon@<nœud>` ; `HEALTH_OK` une vingtaine de secondes plus tard.

Récapitulatif des commandes essentielles

Situation	Commande
Règle pare-feu Datacenter / VM	`pvesh create /cluster/firewall/rules --enable 1 …` / `pvesh create /nodes/<nœud>/qemu/<vmid>/firewall/rules --enable 1 …`
Réseau autorisé d'office par le pare-feu	`pve-firewall localnet`
Créer un datastore PBS (Shell de PBS)	interface : Storage / Disks → Directory → Create ; ou `proxmox-backup-manager datastore create <nom> <chemin>`
Ajouter PBS comme stockage	`pvesm add pbs <id> --server … --datastore … --fingerprint …`
Sauvegarder / restaurer (VM / CT)	`vzdump <id> --storage pbs-backup` / `qmrestore` ou `pct restore`
Monter un backup en lecture seule	`proxmox-backup-client mount <sauvegarde> root.pxar <point-de-montage>`
Créer / lister les namespaces PBS	`proxmox-backup-client namespace create <nom>` / `namespace list` (depuis un nœud)
Créer une remote / un sync job PBS	`proxmox-backup-manager remote create` / `sync-job create`
Créer un jeton API Proxmox VE	`pveum user token add <user>@pve <nom> --privsep 0`
Empreinte du certificat d'un nœud	`openssl x509 -in /etc/pve/local/pve-ssl.pem -noout -fingerprint -sha256`

Ce support est la propriété de Dawan France.

Aide-mémoire — Proxmox VE

> Aide-mémoire des commandes CLI essentielles. Toutes les commandes s'exécutent en `root` sur le nœud Proxmox sauf mention contraire.

VMs KVM — qm

```
# Créer une VM
qm create <vmid> --name <nom> --memory <Mo> --cores <n> \
  --net0 virtio,bridge=vmbr0 \
  --scsi0 local-lvm:<Go> --scsihw virtio-scsi-single \
  --ostype l26 --agent enabled=1

# Cycle de vie
qm start <vmid>
qm shutdown <vmid>      # arrêt propre (signal ACPI)
qm stop <vmid>          # arrêt forcé (coupure alimentation)
qm reboot <vmid>
qm suspend <vmid>

# Informations
qm list
qm status <vmid>
qm config <vmid>

# Modifier à chaud
qm set <vmid> --memory 4096
qm set <vmid> --cores 4
qm set <vmid> --net1 virtio,bridge=vmbr0

# Cloud-Init
qm set <vmid> --ide2 local-lvm:cloudinit
qm set <vmid> --ciuser devops --cipassword dawan --ipconfig0
ip=10.44.1.50/24,gw=10.44.1.1
qm set <vmid> --boot order=scsi0
qm cloudinit dump <vmid> user
```

Dawan
Ce support est la propriété de Dawan France.

```
# Templates et clones
qm template <vmid>
qm clone <vmid> <newid> --name <nom> --full --storage local-lvm

# Snapshots
qm snapshot <vmid> <nom> [--description "texte"] [--vmstate]
qm listsnapshot <vmid>
qm rollback <vmid> <nom>
qm delsnapshot <vmid> <nom>

# Migration
qm migrate <vmid> <nœud-cible> --targetstorage local-lvm   # à froid
qm migrate <vmid> <nœud-cible> --online --targetstorage ceph-vm  # à chaud

# Import d'image cloud (le disque arrive en unused0, à attacher ensuite avec qm set)
qm disk import <vmid> <image.qcow2> <stockage>

# Sauvegarde
vzdump <vmid> --storage <stockage> --mode snapshot --compress zstd

# Restauration
qmrestore <fichier.vma.zst> <newid> --storage local-lvm

# Dépannage
qm monitor <vmid>                          # moniteur QEMU (diagnostic avancé)
pvenode task list --vmid <vmid> --errors 1 # tâches en échec de cette VM
qm guest cmd <vmid> ping                   # l'agent invité répond-il ?
```

Conteneurs LXC — pct

```
# Télécharger un template
pveam update
pveam available --section system
pveam download local <template.tar.zst>
pveam list local

# Créer un conteneur
pct create <ctid> local:vztmpl/<template.tar.zst> \
```

Ce support est la propriété de Dawan France.

```
  --hostname <nom> --memory 512 --swap 512 \
  --rootfs local-lvm:8 \
  --net0 name=eth0,bridge=vmbr0,ip=10.44.1.50/24,gw=10.44.1.1 \
  --unprivileged 1

# Cycle de vie
pct start <ctid>
pct stop <ctid>
pct shutdown <ctid>
pct reboot <ctid>

# Informations
pct list
pct status <ctid>
pct config <ctid>

# Accès
pct enter <ctid>                    # shell root interactif dans le conteneur
pct exec <ctid> -- <commande>       # exécuter une commande

# Modifier à chaud
pct set <ctid> --memory 1024
pct set <ctid> --cores 2

# Snapshots
pct snapshot <ctid> <nom> [--description "texte"]
pct listsnapshot <ctid>
pct rollback <ctid> <nom>
pct delsnapshot <ctid> <nom>

# Sauvegarde
vzdump <ctid> --storage local --mode snapshot --compress zstd
```

Stockages — pvesm

```
# Lister et état
pvesm status
pvesm list <stockage>
```

Dawan Nous formons votre avenir
Ce support est la propriété de Dawan France.

```
# Ajouter des stockages
pvesm add zfspool <id> --pool <pool>/<dataset> --content images,rootdir
pvesm add nfs <id> --server <IP> --export <path> --content iso,backup
pvesm add iscsi <id> --portal <IP> --target <iqn.xxx:target>
pvesm add pbs <id> --server <IP> --datastore <nom> \
  --username '<user>@pbs!<jeton>' --password '<secret-du-jeton>' --fingerprint <fp>
pvesm add rbd <id> --pool <pool> --content images,rootdir  # Ceph

# Supprimer
pvesm remove <id>
```

ZFS — zpool & zfs

```
# Pools
# Désigner les disques par /dev/disk/by-id/ en production ; -f efface les données présentes
zpool create -f -o ashift=12 <nom> mirror /dev/sdb /dev/sdc
zpool create -f -o ashift=12 <nom> raidz /dev/sdb /dev/sdc /dev/sdd
zpool status
zpool list
zpool scrub <nom>           # vérification d'intégrité
zpool destroy <nom>

# Datasets
zfs create <pool>/<dataset>
zfs list
zfs list -t snapshot

# Propriétés
zfs set compression=lz4 <pool>/<dataset>
zfs set quota=100G <pool>/<dataset>
zfs get compression,used,available <pool>/<dataset>

# Snapshots
zfs snapshot <pool>/<dataset>@<nom>
zfs rollback <pool>/<dataset>@<nom>
zfs destroy <pool>/<dataset>@<nom>
```

Dawan
Ce support est la propriété de Dawan France.

```
zfs diff <pool>/<dataset>@<nom> <pool>/<dataset>

# Intégration Proxmox
pvesm add zfspool <id> --pool <pool>/<dataset> --content images,rootdir
```

Réseau — bridges et VLANs

```
# Voir les interfaces et bridges
ip link show
ip addr show
bridge link show

# Appliquer les changements réseau sans redémarrer
ifreload -a

# Voir la configuration Proxmox
cat /etc/network/interfaces
```

SDN — pvesh /cluster/sdn

```
# Lister
pvesh get /cluster/sdn/zones
pvesh get /cluster/sdn/vnets

# Appliquer la configuration (obligatoire après toute modification)
pvesh set /cluster/sdn

# Vérifier le bridge VNet créé
ip link show <vnet-id>
```

Cluster — pvecm

Ce support est la propriété de Dawan France.

```
# Créer un cluster (sur le 1er nœud)
pvecm create <nom-cluster>

# Rejoindre un cluster (sur les nœuds suivants)
pvecm add <IP-nœud1>

# État et diagnostics
pvecm status
pvecm nodes

# Dans l'interface : Datacenter → Cluster → Join Information (à copier sur le nœud qui rejoint)
```

Ceph — pveceph

```
# Installation (sur chaque nœud ; tentacle est la version par défaut de Proxmox VE 9)
pveceph install --repository no-subscription
pveceph init --network <CIDR-réseau-ceph>

# Composants (répéter sur chaque nœud)
pveceph mon create
pveceph mgr create
pveceph osd create /dev/sdX

# Pools (--add_storages déclare aussitôt le stockage RBD dans Proxmox)
pveceph pool create <nom> --size 3 --min_size 2 --add_storages

# État
ceph status
ceph osd tree
ceph df
ceph health detail

# Intégration Proxmox, si le pool a été créé sans --add_storages
pvesm add rbd <id> --pool <nom-pool> --content images,rootdir
```

Dawan
Ce support est la propriété de Dawan France.

Utilisateurs et droits — pveum

```
# Utilisateurs
pveum user list
pveum user add <user>@pve --password <mdp> [--comment "texte"]
pveum user modify <user>@pve --groups <groupe>

# Groupes
pveum group list
pveum group add <groupe>

# Rôles
pveum role list
pveum role add <role> --privs "VM.Console,VM.PowerMgmt"

# Permissions (ACL)
pveum acl list
pveum acl modify <chemin> --users <user>@pve --roles <role>
pveum acl modify <chemin> --groups <groupe> --roles <role>

# API Tokens
pveum user token add <user>@pve <token-name> --privsep 0
```

Firewall — pvesh

```
# Lister les règles datacenter
pvesh get /cluster/firewall/rules

# Ajouter une règle ACCEPT (ex: SSH)
pvesh create /cluster/firewall/rules \
  --action ACCEPT --type in --dport 22 --proto tcp --enable 1 --comment "SSH"

# Activer le firewall
pvesh set /cluster/firewall/options --enable 1

# Security Groups
```

Ce support est la propriété de Dawan France.

```
pvesh create /cluster/firewall/groups --group <nom>
pvesh get /cluster/firewall/groups/<nom>
```

Sauvegardes — vzdump

```
# Sauvegarde unique
vzdump <vmid> --storage <stockage> --compress zstd \
  --mode snapshot       # sauvegarde à chaud par QEMU ; l'agent fige les écritures pour la cohérence
  # --mode suspend      # VM suspendue le temps du backup
  # --mode stop         # VM éteinte

# Sauvegarder tout le nœud
vzdump --all --storage <stockage> --compress zstd --mode snapshot

# Lister les sauvegardes
pvesh get /nodes/<nœud>/storage/<stockage>/content?content=backup

# Restaurer une VM
qmrestore <fichier.vma.zst> <newid> --storage local-lvm --unique

# Restaurer un conteneur
pct restore <newid> <fichier.tar.zst> --storage local-lvm
```

Diagnostic et état général

```
# Version de Proxmox
pveversion -v

# État du nœud
pvesh get /nodes/<nœud>/status

# Services critiques
systemctl status pveproxy pve-cluster pvedaemon

# Tâches récentes
```

Ce support est la propriété de Dawan France.

```
pvesh get /nodes/<nœud>/tasks --limit 20

# Journaux
journalctl -u pve-cluster --since "1 hour ago"
journalctl -u pvedaemon -n 50

# Fichiers de configuration importants
/etc/pve/                          # configuration cluster (pmxcfs)
/etc/pve/qemu-server/<vmid>.conf   # configuration VM
/etc/pve/lxc/<ctid>.conf           # configuration conteneur
/etc/pve/storage.cfg               # configuration stockages
/etc/pve/corosync.conf             # configuration cluster Corosync
/etc/pve/firewall/cluster.fw       # règles firewall datacenter
/etc/network/interfaces            # configuration réseau
```

Concepts clés — aide-mémoire

Tableaux de synthèse des notions à retenir de chaque chapitre, à consulter en révision ou pendant les ateliers.

Virtualisation, installation et stockage de base

Concept	Point clé
Type d'hyperviseur	Proxmox VE est un hyperviseur de type 1, supportant à la fois KVM (VMs) et LXC (conteneurs)
KVM / QEMU / Proxmox	KVM = accélération CPU (module noyau) · QEMU = émulation matérielle · Proxmox VE = orchestration (API, stockage, réseau)
`/etc/pve/`	Répertoire central de la configuration cluster (pmxcfs) — tout ce qui est configuré dans Proxmox y est stocké
`local` vs `local-lvm`	`local` = stockage fichier (ISOs, templates) · `local-lvm` = stockage bloc (disques de VMs et conteneurs)
NFS / iSCSI / Ceph	NFS = simple, ISOs et backups · iSCSI = bloc réseau, performances dépendantes du réseau · Ceph = distribué haute disponibilité, intégré nativement
Dépôts APT	Enterprise = mises à jour validées, réservé aux abonnements · no-subscription = lab ou sans abonnement

Ce support est la propriété de Dawan France.

Concept	Point clé
RBAC	Les permissions Proxmox s'appliquent par chemin (`/vms/100`), pas globalement
namespaces vs cgroups (LXC)	namespaces = isolation (ce que le conteneur voit) · cgroups = limitation (ce qu'il peut consommer)
LXC vs Docker	LXC = alternative légère à une VM, administrée comme un serveur (`systemctl` , `apt`) · Docker = emballage applicatif, jetable
ISO vs template LXC	Une ISO installe un OS · un template LXC est déjà installé et prêt à démarrer
Suppression d'un CT	`pct destroy` exige que le conteneur soit arrêté au préalable
Panne réseau d'un CT	Vérifier d'abord `vmbr0` sur l'hôte (`ip link show vmbr0`), puis l'IP et le sous-réseau du conteneur
Snapshot vs sauvegarde	Snapshot = rapide, local, avant une modification · `vzdump` = plus lent, produit un fichier portable pour archivage/migration

VMs KVM

Concept	Point clé
Fichier `.conf` d'une VM	`qm create` génère `/etc/pve/qemu-server/<vmid>.conf` — modifiable directement, mais `qm set` reste préférable pour éviter les erreurs de syntaxe
QEMU Guest Agent	Indispensable en production : snapshots cohérents (gel des écritures) et visibilité des IPs de la VM sans SSH
Image cloud vs ISO	Une image cloud est plus légère (~300-500 Mo vs 700 Mo+) et arrive avec cloud-init préinstallé — déploiement en ~30 secondes
Déploiement en série	1 template cloud-init → clones liés → cloud-init différencie chaque clone par IP et hostname

Réseau et SDN

Concept	Point clé
Rôle de `vmbr0`	Fournit la connectivité IP à l'hôte Proxmox lui-même ET sert de switch virtuel pour les VMs/CT qui lui sont rattachés
Activer le SDN	Intégré à Proxmox VE (paquet `libpve-network-perl`) : **Datacenter** → **SDN** est présent d'office ; chaque modification ne prend effet qu'après **Apply** (`pvesh set /cluster/sdn`)

Ce support est la propriété de Dawan France.

Concept	Point clé
VNet sans Subnet	Fonctionne en L2 pur, sans IP assignée automatiquement — un Subnet avec DHCP est nécessaire pour l'attribution automatique
Isolation entre VNets	Isolation totale par défaut sans routage explicite entre eux
Scrub ZFS	`0 errors` confirme l'intégrité des données · un `DEGRADED` ou des erreurs de checksum imposent de remplacer le disque immédiatement

Cluster, Ceph et sécurité

Concept	Point clé
Quorum	Avec 3 nœuds/3 votes, 2 nœuds actifs suffisent (majorité simple) · si deux nœuds tombent, le nœud restant perd le quorum : `/etc/pve` passe en lecture seule, les VM en cours continuent mais aucune ne peut être démarrée ni modifiée
`ring0_addr` / `ring1_addr`	Adresses IP Corosync des nœuds · `ring1_addr` ajoute un second lien réseau redondant
État Ceph	`HEALTH_OK` = normal · `HEALTH_WARN` = souvent temporaire (rééquilibrage) · `HEALTH_ERR` = intervention requise
Réseau Ceph lab vs production	Un seul réseau suffit en lab · une interface dédiée 10 Gbps est recommandée en production
Security Groups	Modifier les règles d'un groupe les applique automatiquement à toutes les VMs qui l'utilisent
Héritage des ACL	Combinaison typique : un rôle large sur `/` (ex. `PVEAuditor`) + un rôle restreint sur un sous-chemin (ex. `PVEVMAdmin` sur `/pool/production`)

PBS

Concept	Point clé
Namespaces PBS	Un dossier logique dans un datastore, un par cluster ou par client · côté Proxmox VE, le champ **Namespace** du stockage PBS

Raccourcis interface web

Ce support est la propriété de Dawan France.

Action	Raccourci
Ouvrir la console d'une VM/CT	Clic sur **Console**
Créer une VM	Bouton **Create VM** (haut droite)
Créer un CT	Bouton **Create CT** (haut droite)
Voir les tâches en cours	Onglet **Tasks** du panneau en bas de page
Rechercher une VM/CT	Champ **Search** en haut de l'interface
Actualiser l'affichage	`F5`

Ce support est la propriété de Dawan France.

Glossaire — Proxmox VE

> Référence alphabétique des termes techniques utilisés dans la formation.

A

ACL (Access Control List) Liste de contrôle d'accès. Dans Proxmox, une ACL associe un utilisateur ou groupe, un chemin de ressource (`/` , `/vms/` , `/nodes/`) et un rôle. C'est le mécanisme de base du RBAC Proxmox.

AMD-V (ou SVM) Extension de virtualisation matérielle des processeurs AMD. Équivalent AMD du VT-x Intel. Permet à KVM d'exécuter des VMs avec des performances quasi-natives.

API Token Jeton d'authentification Proxmox alternatif au mot de passe. Utilisé pour les scripts et les intégrations (PBS, PDM, Ansible). Format : `user@realm!token-name` .

B

Bare-metal Installation directement sur le matériel physique, sans système d'exploitation hôte. Proxmox VE est un hyperviseur bare-metal (type 1).

Bonding Agrégation de plusieurs interfaces réseau physiques en une seule interface logique. Permet d'augmenter la bande passante ou d'assurer la redondance réseau. Modes courants : `active-backup` (redondance), `802.3ad` (LACP, performance).

Bridge Linux (vmbr) Switch virtuel implémenté dans le noyau Linux. `vmbr0` est le bridge principal créé par Proxmox à l'installation. Connecte les interfaces physiques, les VMs et les conteneurs dans le même domaine de broadcast.

C

Ceph Solution de stockage distribué open source. Aucun point central de défaillance — les données sont réparties sur l'ensemble des nœuds. Composants principaux : MON (quorum), MGR (métriques), OSD (disques), MDS (métadonnées). Intégré nativement dans Proxmox VE.

Ce support est la propriété de Dawan France.

cgroup (Control Group) Mécanisme du noyau Linux qui limite et isole les ressources (CPU, mémoire, I/O) d'un groupe de processus. Utilisé par LXC pour isoler les conteneurs. Cgroups v2 est le standard depuis Debian Bullseye.

Cloud-Init Outil de personnalisation au premier démarrage d'une VM. Injecte hostname, utilisateur, clé SSH, configuration réseau. Utilisé avec les images cloud pré-installées (Debian cloud, Ubuntu Server). Proxmox génère un disque cloud-init (`nocloud` datasource) attaché à la VM.

Cluster Proxmox Ensemble de nœuds Proxmox reliés par Corosync. Partage la configuration via pmxcfs. Permet la migration de VMs, la haute disponibilité et la gestion centralisée depuis une seule interface.

Corosync Daemon de communication de cluster. Gère le heartbeat entre les nœuds et le quorum. Utilise le protocole Kronosnet (knet) par défaut depuis Proxmox VE 6.

CT (Container) Abréviation de conteneur LXC dans la nomenclature Proxmox. Un CT a un CT ID (entier positif), un hostname, et une configuration stockée dans `/etc/pve/lxc/<ctid>.conf`.

D

Datacenter Dans Proxmox, le niveau hiérarchique le plus haut dans l'interface web. Regroupe tous les nœuds du cluster. Les paramètres définis au niveau Datacenter s'appliquent à tous les nœuds.

Dataset ZFS Sous-volume logique dans un pool ZFS. Possède ses propres propriétés (compression, quota, snapshot). Analogue à un répertoire avec comportement héritable.

Déduplication (PBS) Technique de stockage qui identifie et élimine les données en double. PBS découpe les sauvegardes en blocs (*chunks*) — de taille fixe (4 Mio) pour les disques de VM, de taille variable pour les fichiers des conteneurs — et ne stocke chaque bloc unique qu'une fois. Un ratio de 3:1 signifie que 3 Go de données occupent 1 Go de stockage réel.

E

EVPN (Ethernet VPN) Type de zone SDN : l'étape suivante après VXLAN, pour router entre VNets au niveau 3 avec BGP.

Ce support est la propriété de Dawan France.

F

Fingerprint TLS Empreinte cryptographique du certificat TLS d'un serveur. Utilisée par Proxmox pour vérifier l'identité de PBS sans autorité de certification externe. Format : `AA:BB:CC:...:FF`.

G

Guest Agent (qemu-guest-agent) Agent léger installé à l'intérieur d'une VM KVM. Permet à Proxmox d'obtenir des informations depuis la VM (adresses IP, état du système de fichiers) et d'effectuer des opérations (arrêt propre, gel des systèmes de fichiers avant une sauvegarde ou un instantané). Il n'est pas nécessaire à la migration à chaud. Son service démarre seul quand l'option **QEMU Guest Agent** de la VM est cochée.

H

HA (Haute Disponibilité) Capacité d'un système à redémarrer automatiquement des services si un nœud tombe. Proxmox HA repose sur Corosync (quorum), le HA Manager (`ha-manager`) et, depuis Proxmox VE 9, des **règles HA** (affinité de nœuds, affinité entre ressources) qui remplacent les anciens groupes HA. Nécessite un stockage partagé (Ceph, NFS, iSCSI) ou la réplication ZFS, et au moins 3 votes (3 nœuds, ou 2 nœuds et un QDevice) pour éviter le split-brain.

I

iSCSI (Internet Small Computer System Interface) Protocole qui transporte des commandes SCSI sur un réseau IP. Permet d'exposer des blocs de disque (LUNs) depuis un serveur (target) vers un client (initiator). Proxmox peut utiliser un stockage iSCSI pour les disques de VMs.

K

KVM (Kernel-based Virtual Machine) Module du noyau Linux qui transforme Linux en hyperviseur de type 1. Exploite les extensions de virtualisation matérielle (Intel VT-x / AMD-V) pour exécuter des VMs avec des performances quasi-natives. Intégré dans le noyau Linux depuis la version 2.6.20 (2007).

Dawan
Ce support est la propriété de Dawan France.

L

LUN (Logical Unit Number) Unité logique exposée par un target iSCSI ou Fibre Channel. Un LUN est vu par l'initiateur comme un disque bloc brut.

LVM (Logical Volume Manager) Gestionnaire de volumes logiques Linux. Découple les volumes logiques (visibles par le système) des partitions physiques. Proxmox utilise LVM pour le groupe de volumes `pve`.

LVM-thin (Thin Provisioning LVM) Extension de LVM avec thin provisioning. Les volumes alloués ne consomment de l'espace physique qu'au fur et à mesure des écritures réelles. Permet les snapshots instantanés et l'over-provisioning. Stockage `local-lvm` par défaut dans Proxmox.

LXC (Linux Containers) Technologie de virtualisation par isolation du noyau Linux. Un conteneur LXC partage le noyau de l'hôte mais possède son propre système de fichiers racine, ses interfaces réseau et ses processus. Plus léger qu'une VM KVM mais limité aux OS Linux.

M

Migration à chaud (live migration) Déplacement d'une VM en cours d'exécution d'un nœud à un autre sans interruption de service. Avec un stockage partagé (Ceph, NFS, iSCSI), seule la mémoire est transférée : c'est quasi immédiat. Avec des disques locaux, `qm migrate --online --with-local-disks` copie aussi les disques, ce qui prend plus de temps. L'agent invité n'est pas nécessaire.

Migration à froid Déplacement d'une VM éteinte entre nœuds. Ne nécessite pas de stockage partagé : Proxmox copie les disques d'un nœud à l'autre par le réseau du cluster. Plus lent, mais sans contrainte de stockage.

MON (Ceph Monitor) Daemon Ceph responsable du quorum et de la carte du cluster (CRUSH map). Un cluster Ceph nécessite un nombre impair de MONs (minimum 1, recommandé 3 pour la haute disponibilité).

MDS (Ceph Metadata Server) Daemon Ceph gérant les métadonnées du système de fichiers CephFS. Non requis pour un usage Proxmox en RBD (blocs).

MGR (Ceph Manager) Daemon Ceph qui collecte les métriques, expose l'API REST et héberge les modules d'extension (dashboard, Prometheus exporter…).

Dawan
Ce support est la propriété de Dawan France.

N

Namespace Linux Mécanisme d'isolation du noyau. LXC utilise plusieurs types de namespaces : PID (processus), NET (réseau), MNT (système de fichiers), UTS (hostname), IPC, USER. Chaque conteneur vit dans son propre ensemble de namespaces.

NFS (Network File System) Protocole de partage de fichiers sur réseau. Proxmox peut monter des partages NFS pour stocker des ISOs, templates ou sauvegardes partagés entre nœuds.

noVNC Implémentation VNC dans le navigateur web, sans plugin. Utilisée par Proxmox pour exposer la console graphique des VMs et conteneurs.

O

OSD (Object Storage Daemon) Daemon Ceph responsable du stockage physique des données. Chaque OSD gère un disque (ou une partition). La réplication des données se fait directement entre OSDs.

Over-provisioning Allocation de plus d'espace virtuel que l'espace physique disponible. Possible avec LVM-thin et Ceph. Acceptable si les VMs n'utilisent pas simultanément tout l'espace alloué, mais risqué sans surveillance.

P

PBS (Proxmox Backup Server) Produit Proxmox dédié aux sauvegardes. Écrit en Rust. Fonctionnalités clés : déduplication par chunks, compression zstd, chiffrement côté client, vérification d'intégrité, interface web sur le port 8007.

pct Commande CLI Proxmox pour gérer les conteneurs LXC. Équivalent de `qm` pour les VMs. Principales sous-commandes : `create`, `start`, `stop`, `enter`, `exec`, `set`, `snapshot`, `restore`.

PDM (Proxmox Datacenter Manager) Produit Proxmox qui rassemble plusieurs clusters et nœuds dans une seule interface, et permet de migrer une VM d'un cluster à l'autre. Produit récent (première préversion fin 2024). Interface web sur le port 8443. Présenté au chapitre 5, §7 (l'écosystème Proxmox au complet).

Ce support est la propriété de Dawan France.

pmxcfs (Proxmox Cluster File System) Système de fichiers en cluster propre à Proxmox. Basé sur SQLite et répliqué via Corosync. Stocke toute la configuration de la plateforme dans `/etc/pve/`. Présent même sur un nœud standalone.

Pool ZFS Unité de stockage de base de ZFS. Un pool (`zpool`) regroupe un ou plusieurs vdevs et expose un espace de stockage unifié. Analogue à un groupe de volumes LVM.

pveceph Commande CLI Proxmox pour gérer Ceph. Permet d'initialiser, déployer MON/MGR/OSD et créer des pools depuis le shell sans passer par l'interface web.

pvecm Commande CLI pour gérer le cluster Proxmox (Corosync). `pvecm create` crée un cluster, `pvecm add` joint un nœud, `pvecm status` affiche l'état.

pvesh /cluster/sdn Point d'entrée CLI de la configuration SDN. `pvesh set /cluster/sdn` déclenche l'Apply, obligatoire après toute modification de zone, de VNet ou de subnet — sans lui, la configuration reste `pending` et rien n'existe sur le nœud. (La commande `pvesdn`, citée par d'anciens supports, n'existe pas : elle n'a jamais été livrée.)

pvesm Commande CLI pour gérer les stockages Proxmox. `pvesm add` enregistre un nouveau stockage, `pvesm status` affiche l'état de tous les stockages.

pveum Commande CLI pour gérer les utilisateurs, groupes, rôles et ACL Proxmox.

Q

qm Commande CLI Proxmox pour gérer les VMs KVM. Principales sous-commandes : `create`, `start`, `stop`, `shutdown`, `set`, `config`, `snapshot`, `rollback`, `clone`, `template`, `migrate`.

QEMU Quick Emulator. Émulateur de matériel open source. Dans Proxmox, QEMU émule les périphériques de chaque VM (disque, réseau, BIOS). Travaille toujours en tandem avec KVM pour l'accélération CPU.

Quorum Mécanisme de prise de décision collective dans un cluster. Un cluster Proxmox a le quorum si plus de la moitié des nœuds configurés sont joignables. Sans quorum, le cluster refuse de modifier la configuration pour éviter le split-brain.

R

Ce support est la propriété de Dawan France.

RBAC (Role-Based Access Control) Contrôle d'accès basé sur les rôles. Dans Proxmox : utilisateurs → groupes → rôles → ACL sur des chemins de ressources. Les rôles prédéfinis incluent `PVEAdmin`, `PVEVMAdmin`, `PVEAuditor`, `PVEDatastoreAdmin`.

RBD (RADOS Block Device) Interface de stockage bloc de Ceph. Expose des volumes bloc (disques de VMs) directement depuis le cluster Ceph. Proxmox utilise RBD pour les disques de VMs stockés dans Ceph.

RAIDZ Niveau de redondance ZFS. RAIDZ1 tolère 1 panne de disque, RAIDZ2 tolère 2 pannes, RAIDZ3 tolère 3 pannes. Contrairement au RAID matériel, RAIDZ recalcule les checksums à chaque lecture pour détecter la corruption silencieuse.

Realm (domaine d'authentification) Dans Proxmox, source d'authentification des utilisateurs. `pam` = comptes Linux locaux, `pve` = comptes internes Proxmox, `ldap` = annuaire LDAP/AD.

S

SDN (Software-Defined Networking) Réseau défini par logiciel. Dans Proxmox, le SDN permet de créer des réseaux virtuels (VNets) indépendants de la configuration physique des switches. Objets : Zones → VNets → Subnets.

Snapshot Capture instantanée de l'état d'une VM, d'un conteneur ou d'un dataset ZFS. Avec LVM-thin ou ZFS, la création est quasi-instantanée car seules les différences ultérieures sont stockées.

Storage ID (identifiant de stockage) Nom unique d'un stockage dans Proxmox (`local`, `local-lvm`, `ceph-vm`...), déclaré dans `/etc/pve/storage.cfg`. C'est lui qui précède les deux-points dans l'adresse d'un volume : `local-lvm:vm-150-disk-0`.

Subnet SDN Sous-réseau associé à un VNet Proxmox. Définit la plage d'adresses IP, la passerelle, les plages DHCP et le mode SNAT. C'est le niveau où se configure l'attribution IP automatique.

T

Template Dans Proxmox, une VM convertie en modèle non démarrable. Sert de base pour créer des clones. Une VM devient template avec `qm template <vmid>` — opération irréversible depuis la CLI.

Ce support est la propriété de Dawan France.

Thin provisioning Voir LVM-thin. Allocation d'espace à la demande plutôt qu'en avance. Permet l'over-provisioning et les snapshots instantanés.

TOTP (Time-based One-Time Password) Authentification à deux facteurs basée sur un code généré toutes les 30 secondes. Proxmox supporte TOTP pour renforcer l'authentification des comptes.

V

vdev (Virtual Device) Unité de base d'un pool ZFS. Un vdev peut être un disque seul, un miroir, ou un groupe RAIDZ. Les vdevs sont combinés dans un pool pour former l'espace de stockage total.

VirtIO Standard de paravirtualisation pour les IO des VMs. Les pilotes VirtIO (réseau, disque, mémoire balloon) permettent à une VM de communiquer directement avec l'hyperviseur sans émulation de matériel physique. Performances supérieures aux pilotes émulés.

VM ID (vmid) Identifiant numérique unique d'une VM ou d'un conteneur dans le datacenter Proxmox. Entier positif entre 100 et 999999999. Unique par cluster, pas seulement par nœud.

VNet (Virtual Network) Réseau virtuel SDN dans Proxmox. Un VNet appartient à une zone et peut avoir plusieurs subnets. Se matérialise comme un bridge Linux sur le nœud après `pvesh set /cluster/sdn`.

VT-x Extension de virtualisation matérielle des processeurs Intel (Virtualization Technology for IA-32/64). Nécessaire pour KVM. Vérifiable avec `grep vmx /proc/cpuinfo`.

vzdump Outil de sauvegarde unifié de Proxmox pour VM et conteneurs. Modes : `stop` (machine arrêtée le temps de la sauvegarde), `suspend` (machine suspendue), `snapshot` (sauvegarde à chaud : pour une VM, QEMU copie les disques pendant qu'elle tourne, l'agent invité figeant les écritures au départ ; pour un conteneur, un instantané du stockage est utilisé).

VXLAN (Virtual eXtensible LAN) Protocole d'encapsulation réseau qui transporte des trames Ethernet de couche 2 sur UDP/IP. Permet d'étendre des segments L2 sur un réseau routable L3. Utilisé dans le SDN Proxmox pour les zones VXLAN et EVPN.

Z

Ce support est la propriété de Dawan France.

ZFS Zettabyte File System. Système de fichiers avancé combinant gestion de volumes, RAID logiciel, snapshots et checksums d'intégrité. Né chez Sun Microsystems (2001), maintenu par le projet OpenZFS. Intégré nativement dans Proxmox VE depuis la version 3.4.

Zone SDN Niveau d'isolation réseau dans Proxmox SDN. Types disponibles : Simple (un pont isolé sur chaque nœud), VLAN (isolation par étiquettes 802.1Q), VXLAN (réseau de niveau 2 transporté sur IP), EVPN (VXLAN avec routage dynamique BGP).

Dawan
Ce support est la propriété de Dawan France.