PROXMOX VE 9

Le guide complet pour construire une infrastructure hyperconvergée open source

KVM · LXC · Ceph · SDN · ZFS · Haute disponibilité

AYI NEDJIMI

Guide pratique — 2026

Proxmox VE 9

Maitrisez le clustering, la haute disponibilite, le SDN et Ceph

Guide complet pour les environnements hyper-converges

Ayi NEDJIMI

2026

ISBN 978-2-9999-0001-5

Mentions legales

Proxmox VE 9 — Maitrisez le clustering, la haute disponibilite, le SDN et Ceph

Guide complet et pratique pour les environnements hyper-converges

Auteur : Ayi NEDJIMI

Date de publication : 2026

ISBN : *[a completer]*

Depot legal : *[a completer]*

Version : Proxmox VE 9.x — Ceph Squid 19.2

Compose avec Jupyter Book et LaTeX.

Preface

Pourquoi ce livre ?

Proxmox VE s'est impose comme la plateforme de virtualisation open source de reference dans les datacenters europeens, les ESN et les homelabs exigeants. Gratuit, sans licence par CPU, avec un acces direct au code source — c'est la reponse pragmatique aux couts croissants de VMware et aux complexites d'OpenStack.

Avec la version 9, basee sur Debian 13 Trixie et le kernel 6.17, Proxmox franchit un nouveau cap : regles d'affinite HA pour un placement intelligent des VM, fabrics SDN auto-configurables (OpenFabric, OSPF), Ceph Squid 19.2 avec de meilleures performances BlueStore, et une interface web renovee. Ces evolutions transforment Proxmox d'un excellent hyperviseur en une plateforme HCI (Hyper-Converged Infrastructure) a part entiere.

Pourtant, la documentation francophone complete sur ces sujets avances reste rare. Les ressources en ligne sont fragmentees, souvent en anglais, et rarement ancrees dans des scenarios de production reels. **Ce livre comble ce manque** en proposant un guide progressif, structure, et constamment ancre dans la pratique a travers un fil rouge concret.

Ce que vous allez construire

Tout au long des 14 chapitres, vous deployez l'infrastructure complete de **TechFlow SAS**, une entreprise fictive de 80 salaries avec des contraintes reelles : haute disponibilite, isolation reseau multi-tenant, sauvegarde off-site, et conformite securite.

A la fin du livre, votre lab contiendra :

- Un cluster Proxmox VE 9 de **3 noeuds** avec quorum robuste
- Un cluster **Ceph Squid** hyper-convergent (stockage bloc, fichier, objet)
- Un reseau **SDN complet** avec zones VXLAN, VNets segmentes et IPAM
- La **haute disponibilite** avec regles d'affinite et fencing hardware
- Des sauvegardes **PBS** avec deduplication et chiffrement
- Un monitoring **Prometheus + Grafana** avec alertes
- Une configuration **securite** conforme aux bonnes pratiques

A qui s'adresse ce livre ?

- **Administrateurs systemes** gerant ou deployant des infrastructures virtualisees
- **Architectes cloud** concevant des solutions hyper-convergees on-premise
- **Homelabbers avances** souhaitant construire un lab solide et representatif du monde reel
- **Equipes en migration** quittant VMware vSphere ou Hyper-V

Ce livre suppose une connaissance de base de Linux (ligne de commande, reseau, stockage) mais n'exige aucune experience prealable de Proxmox. Les concepts sont introduits progressivement, des fondamentaux jusqu'aux configurations les plus avancees.

Comment lire ce livre ?

Lecture lineaire

Les chapitres sont concus pour etre lus dans l'ordre. Chaque chapitre s'appuie sur le precedent et fait evoluer le lab TechFlow SAS. C'est la voie recommandee pour les debutants sur Proxmox.

Lecture par besoin

Chaque chapitre est aussi autoportant. Si vous administrez deja un cluster en production, vous pouvez aller directement aux chapitres SDN (5-6), HA (7), Ceph (8) ou Securite (10).

Le fil rouge : TechFlow SAS

L'infrastructure de TechFlow SAS se construit chapitre par chapitre :

1. **Chapitre 2** — Installation de 3 noeuds Proxmox VE 9 sur bare-metal
2. **Chapitre 3** — Formation du cluster `techflow-cluster` avec Corosync

3. **Chapitre 4** — Deploiement des premieres VM (Windows Server, Debian) et conteneurs LXC
4. **Chapitres 5-6** — SDN avec isolation VXLAN par departement
5. **Chapitre 7** — Haute disponibilite avec affinites et fencing IPMI
6. **Chapitre 8** — Ceph hyper-convergent pour stocker VM et donnees
7. **Chapitre 9** — Sauvegardes incrementales et restauration testee
8. **Chapitres 10-14** — Securisation, optimisation, monitoring, automatisation et gestion multi-site

Les encadres

Quatre types d'encadres jalonnent le livre :

> ✗ **PIEGE DE LA V9**
> Comportement nouveau ou changement par rapport a la v8 pouvant causer une regression en production.

> ✔ **HOMELAB VS PRODUCTION**
> Distingue les configurations acceptables en lab des exigences strictes d'un environnement critique.

> ⚠ **MIGRATION DEPUIS LA V8**
> Points d'attention lors d'une mise a niveau depuis Proxmox VE 8.x vers la version 9.

> ● **EN PRODUCTION**
> Recommandation specifique pour les environnements en production avec des VM critiques.

Conventions typographiques

- `commande` — Commande a taper dans un terminal
- **Gras** — Terme important ou element d'interface graphique
- *Italique* — Terme anglais, nom de produit ou concept introduit pour la premiere fois
- `fichier.conf` — Nom de fichier, chemin ou parametre de configuration

Les blocs de code indiquent le contexte d'execution :

```
# Commande a executer sur un noeud Proxmox
pvecm status
```

Ressources en ligne

Les scripts, templates Terraform/Ansible, dashboards Grafana et l'errata sont disponibles en ligne. Les QR codes en annexe (chapitre 14) pointent directement vers ces ressources.

Bonne lecture, bon deploiement, et que vos VM ne migrent jamais au mauvais moment.

Ayi NEDJIMI — Mars 2026

Bienvenue dans ce guide complet consacre a **Proxmox VE 9**, la plateforme de virtualisation open source de reference pour les environnements hyper-converges.

A qui s'adresse ce livre ?

- **Administrateurs systemes** souhaitant maitriser Proxmox en production
- **Architectes cloud** evaluant ou deployant des solutions hyper-convergees
- **Homelabbers avances** voulant construire une infrastructure solide

Fil rouge pedagogique

Tout au long de ce livre, nous construirons progressivement le lab d'une entreprise fictive : de l'installation d'un premier noeud jusqu'a un cluster hyper-converge complet avec Ceph, SDN, haute disponibilite et monitoring.

Chaque chapitre s'appuie sur le precedent pour faire evoluer cette infrastructure :

1. **Chapitre 1** — Comprendre l'architecture et les nouveautes de la v9
2. **Chapitre 2** — Installer et mettre a niveau vers Proxmox VE 9
3. **Chapitre 3** — Former un cluster de 3 noeuds
4. **Chapitre 4** — Creer des VM, conteneurs et configurer le stockage
5. **Chapitre 5** — Mettre en place le SDN (zones, VNets, IPAM)
6. **Chapitre 6** — SDN avance avec EVPN, Fabrics et multi-tenant
7. **Chapitre 7** — Activer la haute disponibilite et les regles d'affinite
8. **Chapitre 8** — Deployer Ceph en hyper-convergence
9. **Chapitre 9** — Sauvegardes avec PBS et disaster recovery
10. **Chapitre 10** — Securiser et durcir l'infrastructure
11. **Chapitre 11** — Optimiser les performances
12. **Chapitre 12** — Monitoring, API et automatisation
13. **Chapitre 13** — Gestion multi-cluster avec PDM
14. **Chapitre 14** — Annexes, references CLI et checklists

Encadres recurrents

Tout au long du livre, vous retrouverez des encadres thematiques :

> ✗ **PIEGE DE LA V9**
> Signale un comportement nouveau ou un changement par rapport a la v8 qui peut surprendre.

> ✔ **HOMELAB VS PRODUCTION**
> Distingue les configurations adaptees a un lab personnel de celles requises en production.

> ⚠ **MIGRATION DEPUIS LA V8**
> Indique les points d'attention lors d'une mise a niveau depuis Proxmox VE 8.x.

> ● **EN PRODUCTION**
> Recommandation specifique pour les environnements de production.

PARTIE I

Fondations

1 Introduction, architecture et migrations

Proxmox Virtual Environment (PVE) est une plateforme de virtualisation open source qui integre dans une solution unifiee l'hyperviseur KVM, les conteneurs LXC, le stockage distribue Ceph et le reseau defini par logiciel (SDN). Avec la version 9, sortie en aout 2025 et basee sur Debian 13 « Trixie », Proxmox franchit un cap decisif en matiere de fonctionnalites enterprise : regles d'affinite HA, fabrics SDN, Ceph Squid 19.2 et bien plus. Ce premier chapitre retrace l'histoire de la plateforme, presente ses nouveautes majeures, et vous aide a positionner Proxmox dans votre strategie d'infrastructure.

1.1 Historique et positionnement de Proxmox VE

De Proxmox 0.9 a aujourd'hui

L'histoire de Proxmox VE commence en avril 2008, lorsque la societe autrichienne Proxmox Server Solutions GmbH publie la version 0.9 — une interface web de gestion pour les hyperviseurs OpenVZ et KVM. L'idee fondatrice etait simple mais ambitieuse : offrir une alternative open source aux solutions de virtualisation proprietaires, accessible via un navigateur web sans client lourd.

Les etapes cles de l'evolution :

- **2008 — v1.0** : Premiere version stable. Migration live, sauvegardes avec vzdump, support QEMU. Proxmox pose les bases d'une plateforme integree.
- **2012 — v2.0** : Introduction de l'API REST, du cluster Corosync, des snapshots live et d'une nouvelle interface graphique. Cette version marque le debut de l'automatisation et de la gestion multi-noeuds.
- **2013 — v3.0** : Templates de VM et clonage, serveur API event-driven. Base sur Debian 7 Wheezy.
- **2015 — v4.0** : Rupture architecturale majeure — LXC remplace OpenVZ comme technologie de conteneurs. OpenVZ etait un patch kernel proprietaire ; LXC est un standard du kernel Linux mainline. Un nouveau gestionnaire de haute disponibilite est egalement introduit.
- **2017 — v5.0** : Replication de stockage open source. Base sur Debian 9 Stretch.
- **2019 — v6.0** : Introduction experimentale du SDN (Software-Defined Networking). Ameliorations de stabilite et de performance. Base sur Debian 10 Buster.
- **2021 — v7.0** : Ameliorations du clustering et des conteneurs. Support de Ceph Pacific. Base sur Debian 11 Bullseye.
- **2023 — v8.0** : Le SDN devient officiellement supporte et active par defaut (v8.1). Ceph Reef 18.2. Base sur Debian 12 Bookworm. Cette version accelere l'adoption en entreprise.
- **2025 — v9.0** : Debian 13 Trixie, kernel 6.14, QEMU 10.0, Ceph Squid 19.2, ZFS 2.3. Regles d'affinite HA, fabrics SDN, snapshots LVM. L'annee 2025 marque egalement les 17 ans de la plateforme.

A chaque version majeure, Proxmox a suivi le cycle des releases Debian, garantissant une base systeme stable et largement supportee. Cette strategie permet de beneficier de l'immense ecosysteme Debian tout en ajoutant les couches de virtualisation et de gestion propres a PVE.

Les piliers technologiques

Proxmox VE repose sur quatre piliers complementaires :

KVM (Kernel-based Virtual Machine) est l'hyperviseur integre au kernel Linux depuis 2007. Il fournit une virtualisation complete (type 1) avec des performances quasi-natives grace aux extensions materielles VT-x (Intel) et AMD-V. Proxmox utilise QEMU comme emulateur en espace utilisateur pour gerer les peripheriques virtuels des VM.

LXC (Linux Containers) offre une virtualisation legere au niveau du systeme d'exploitation. Contrairement a KVM qui emule un hardware complet, LXC partage le kernel de l'hote et isole les

processus via les namespaces et cgroups du kernel. Le resultat : des conteneurs qui demarrent en moins d'une seconde, consomment tres peu de memoire, et offrent des performances natives — au prix d'une isolation moins forte que KVM et d'une limitation aux systemes Linux.

Ceph est un systeme de stockage distribue qui transforme les disques locaux de chaque noeud en un pool de stockage partage, replique et auto-reparant. Integre directement dans Proxmox, Ceph elimine le besoin d'un SAN ou NAS externe et permet une architecture hyper-convergee ou chaque noeud fournit a la fois compute et stockage.

SDN (Software-Defined Networking) permet de gerer les reseaux virtuels de maniere centralisee depuis l'interface Proxmox, sans configuration manuelle sur chaque noeud. Le SDN supporte plusieurs types de zones (VLAN, VXLAN, EVPN) et, depuis la v9, des fabrics de routage automatise (IS-IS, OSPF).

Positionnement face a la concurrence

Proxmox VE se positionne comme une alternative open source credible aux grandes plateformes de virtualisation proprietaires.

Comparatif des plateformes de virtualisation (2025)

Critere	Proxmox VE 9	VMware vSphere	Microsoft Hyper-V	oVirt / RHV
Licence	AGPL v3 (gratuit)	Proprietaire (~4 500 EUR/CPU/an)	Proprietaire (inclus Windows Server)	Open source (arrete par Red Hat)
Base OS	Debian Linux	ESXi (proprietaire)	Windows Server	CentOS/RHEL
Hyperviseur	KVM	ESXi	Hyper-V	KVM
Conteneurs natifs	LXC + OCI (v9)	Non	Containers Windows	Non
Stockage integre	Ceph, ZFS	vSAN (licence separee)	Storage Spaces Direct	GlusterFS (deprecated)
SDN integre	Oui (VLAN, VXLAN, EVPN, Fabrics)	NSX (licence separee)	SDN Controller	OVN
Gestion multi-cluster	PDM	vCenter	SCVMM	Non
Cout cluster 3 noeuds/an	0 EUR (sans support) / ~345 EUR (support)	~13 500 EUR+	Licence Windows Server	Arrete

Le rachat de VMware par Broadcom en 2023 a profondement modifie le paysage. L'abandon des licences perpetuelles au profit d'abonnements par CPU, la suppression du tier gratuit ESXi, et l'augmentation significative des tarifs ont pousse de nombreuses entreprises a evaluer des alternatives. Proxmox VE, avec son modele 100 % open source et ses fonctionnalites enterprise, est devenu le candidat naturel pour ces migrations.

> ✔ **HOMELAB VS PRODUCTION**
> Proxmox VE est entierement gratuit, meme en production. La souscription payante donne acces au depot enterprise (packages testes et stables) et au support technique. En homelab, le depot « no-subscription » est parfaitement fonctionnel.

1.2 Nouveautes majeures Proxmox VE 9

La version 9.0 de Proxmox VE, publiee en aout 2025, apporte des evolutions majeures a tous les niveaux de la pile. Voici un tour d'horizon structure.

Debian 13 Trixie et kernel 6.14

Proxmox VE 9 repose sur Debian 13 « Trixie » avec le kernel Linux 6.14.8. Cette nouvelle base apporte :

- **Support materiel elargi** : Nouveaux drivers pour les processeurs Intel Arrow Lake, AMD Zen 5, les GPU recents et les controleurs NVMe de derniere generation.

- **Performances ameliorees** : Optimisations du scheduler CPU, meilleur support NUMA, ameliorations de la pile reseau.
- **Securite renforcee** : Correctifs de securite les plus recents, support ameliore de Secure Boot.
- **QEMU 10.0.2** : Nouvelle version de l'emulateur avec des mecanismes d'attachement de disques modernises (type de machine 10.0+).
- **LXC 6.0.4** : Version LTS avec ameliorations de stabilite.
- **ZFS 2.3.3** : Ajout de la possibilite d'etendre un pool RAIDZ existant en ajoutant de nouveaux disques, sans reconstruction complete.

Regles d'affinite HA

C'est probablement la nouveaute la plus attendue par les administrateurs en production. Les regles d'affinite remplacent les anciens groupes HA et offrent un controle beaucoup plus fin du placement des VM et conteneurs :

- **Affinite noeud** : Definir sur quels noeuds une ressource doit ou peut s'executer (mode strict ou preferentiel).
- **Affinite ressource** : Forcer des VM a rester ensemble sur le meme noeud (co-localisation) ou au contraire a etre separees sur des noeuds differents (anti-affinite).

Exemple concret : vous pouvez garantir que vos trois replicas de base de donnees ne se retrouvent jamais sur le meme noeud, tout en forcant un serveur applicatif et son cache Redis a rester co-localises.

> ⚠ **MIGRATION DEPUIS LA V8**
> Les groupes HA de la v8 sont automatiquement convertis en regles d'affinite lors de la mise a niveau. Cependant, les groupes utilisant des options avancees comme `nofailback` ou `restricted` peuvent necessiter un ajustement manuel. Verifiez systematiquement apres la migration. Les details sont couverts au chapitre 7.

SDN Fabrics (OpenFabric/IS-IS et OSPF)

Les fabrics sont une fonctionnalite entierement nouvelle qui automatise le routage entre les noeuds du cluster. Deux protocoles sont supportes :

- **OpenFabric (IS-IS)** : Base sur le protocole IS-IS, optimise pour les topologies spine-leaf. Chaque noeud Proxmox devient un routeur IS-IS, et les routes sont apprises automatiquement. Simple a configurer, ideal pour les clusters de taille petite a moyenne.
- **OSPF** : Le protocole Open Shortest Path First, bien connu des administrateurs reseau. Supporte les areas multiples, adapte aux grands deployments.

Les fabrics permettent de construire des architectures reseau avancees sans configuration manuelle de routes statiques : - Underlays EVPN pour les overlays VXLAN - Reseaux Ceph full-mesh dedies - Routage multi-site avec failover automatique

> ✗ **PIEGE DE LA V9**
> Les fabrics sont une fonctionnalite entierement nouvelle. Il n'y a aucune migration automatique depuis les configurations SDN de la v8 — elles doivent etre configurees de zero. Planifiez cette mise en place dans votre fenetre de maintenance.

Ceph Squid 19.2

Proxmox VE 9 integre Ceph Squid 19.2.3, une version majeure du stockage distribue :

- **Snapshots LVM partages** : Support tant attendu des snapshots sur le stockage LVM provisionne en mode thick (iSCSI, Fibre Channel). Les snapshots utilisent des chaines de volumes ou seules les differences par rapport au volume parent sont enregistrees.
- **Compression LZ4 par defaut** : Meilleur compromis performance/espace sur les pools BlueStore.
- **Configurations Erasure Coding flexibles** : Nouveaux profils EC pour un meilleur equilibre entre protection et capacite.
- **RBD : execution locale du diff-iterate** : Ameliore les performances de synchronisation QEMU en temps reel.
- **RGW : API IAM compatibles AWS** : Gestion self-service des utilisateurs du stockage objet.

Autres nouveautes notables

- **Interface mobile redessinee** : Entierement reecrite avec le framework Yew (Rust), offrant un acces rapide aux operations essentielles depuis un smartphone ou une tablette.
- **Suppression de cgroupv1** : Les conteneurs utilisant systemd v230 ou anterieur ne sont plus supportes. Les conteneurs doivent utiliser cgroupv2 (defaut depuis la v7).
- **Suppression de GlusterFS** : Le support du stockage GlusterFS est retire, le projet upstream n'etant plus maintenu.
- **Comptabilite memoire** : L'overhead memoire des VM (QEMU, I/O threads) est desormais compte cote hote, donnant une vision plus precise de la consommation reelle.

Proxmox VE 8 vs 9 — Principales differences

Composant	Proxmox VE 8.x	Proxmox VE 9.x
Base OS	Debian 12 Bookworm	Debian 13 Trixie
Kernel	6.8	6.14
QEMU	8.x	10.0.2
Ceph	Reef 18.2	Squid 19.2.3
ZFS	2.2	2.3.3 (RAIDZ expansion)
LXC	5.0	6.0.4
HA	Groupes HA	Regles d'affinite (noeud + ressource)
SDN	Zones, VNets, EVPN	• Fabrics (IS-IS, OSPF)
Stockage	Pas de snapshot LVM thick	Snapshots LVM thick (iSCSI, FC)
cgroups	v1 + v2	v2 uniquement
GlusterFS	Supporte	Retire

1.3 Architecture hyper-convergee vs traditionnelle

Architecture traditionnelle a trois tiers

L'architecture IT traditionnelle separe l'infrastructure en trois couches independantes :

1. **Tier compute** : Serveurs de virtualisation (hyperviseurs) hebergeant les VM. Ces serveurs n'ont generalement qu'un stockage local minimal pour le systeme d'exploitation.
2. **Tier stockage** : Un SAN (Storage Area Network) ou NAS (Network Attached Storage) centralise fournit le stockage partage via Fibre Channel, iSCSI ou NFS. Cet equipement est generalement couteux, proprietaire et necessite une expertise specialisee.
3. **Tier reseau** : Des switches dedies interconnectent les serveurs et le SAN. Le reseau de stockage est souvent separe du reseau de donnees (fabric FC dedie ou VLAN iSCSI).

Cette architecture a fait ses preuves pendant deux decennies, mais presente des inconvenients croissants : - **Cout** : Le SAN est l'equipement le plus couteux du datacenter, tant a l'achat qu'en maintenance. - **Complexite** : Trois equipes (compute, stockage, reseau) doivent se coordonner pour chaque changement. - **Scalabilite en escalier** : Augmenter la capacite de stockage ou de compute necessite un investissement massif (« forklift upgrade »). - **Point de defaillance unique** : Le SAN, malgre sa redondance interne, reste un SPOF architectural.

Architecture hyper-convergee (HCI)

L'infrastructure hyper-convergee (HCI — Hyper-Converged Infrastructure) prend l'approche inverse : chaque noeud combine compute, stockage et reseau dans une seule unite.

Le principe est simple : au lieu d'acheter des serveurs « stupides » qui se connectent a un SAN « intelligent », chaque serveur dispose de ses propres disques, et un logiciel de stockage distribue (comme Ceph) agrege tous ces disques en un pool de stockage partage et replique.

Avantages de l'approche HCI :

- **Scalabilite lineaire** : Besoin de plus de capacite ? Ajoutez un noeud identique. Chaque noeud apporte simultanement du compute, du stockage et de la bande passante reseau.

- **Simplicite operationnelle** : Une seule equipe gere l'ensemble. Une seule interface de management (Proxmox). Pas de SAN proprietaire a maintenir.
- **Cout reduit** : Pas de SAN, pas de licences de stockage. Le materiel est du serveur standard x86.
- **Resilience distribuee** : Pas de SPOF. Les donnees sont repliquees sur plusieurs noeuds. La perte d'un noeud est toleree par design.
- **Densite** : Moins de materiel, moins de cables, moins de consommation electrique, moins d'espace rack.

Limites a connaitre :

- **Ressources couplees** : Compute et stockage evoluent ensemble. Si vous avez besoin de beaucoup de stockage mais peu de compute (ou l'inverse), l'HCI peut etre sous-optimal.
- **Reseau critique** : Le reseau entre les noeuds porte a la fois le trafic VM et le trafic de replication Ceph. Un reseau 10 Gbps minimum est indispensable en production.
- **Overhead** : Ceph consomme du CPU et de la RAM sur chaque noeud. Prevoyez environ 2 Go de RAM et 1 coeur CPU par OSD.

Architecture hyper-convergee Proxmox VE : chaque noeud fournit compute (KVM/LXC), stockage (Ceph OSD) et reseau (SDN). Les donnees sont repliquees sur les 3 noeuds.

Quand choisir l'une ou l'autre ?

L'HCI avec Proxmox + Ceph est le bon choix quand : - Vous gerez moins de 50 noeuds (au-dela, une architecture separee peut se justifier) - Vous souhaitez une solution integree et simple a operer - Votre budget ne permet pas un SAN enterprise - Vous voulez une scalabilite progressive (noeud par noeud)

L'architecture traditionnelle reste pertinente quand : - Vous avez deja un SAN performant et amorti - Vos besoins en stockage et en compute sont tres desequilibres - Vous avez des equipes specialisees (stockage, reseau) en place - Des contraintes reglementaires imposent une separation physique

> **• EN PRODUCTION**
> En environnement hyper-converge, dimensionnez toujours votre cluster pour supporter la perte d'un noeud complet (regle N-1). Si chaque noeud a 256 Go de RAM et 6 OSD Ceph, ne depassez pas ~170 Go de RAM utilisee et 4 OSD « utiles » par noeud, de sorte que le noeud restant puisse absorber la charge en cas de panne.

1.4 Cas d'usage

PME et ETI

Proxmox VE est particulierement adapte aux PME et ETI pour qui VMware est devenu trop couteux apres les changements de tarification Broadcom. Un cluster de 3 noeuds Proxmox avec Ceph fournit une plateforme complete de virtualisation avec haute disponibilite, stockage replique et sauvegardes integrees — pour un cout de licence de zero euro.

Cas typiques : - **Remplacement VMware** : Migration des VM existantes vers Proxmox (voir section 1.5) - **Consolidation de serveurs** : Remplacer 10-20 serveurs physiques par un cluster de 3-5 noeuds - **Hebergement d'applications metier** : ERP, CRM, messagerie, bases de donnees

Homelab et formation

Proxmox est devenu la plateforme de reference des homelabbers grace a sa gratuite, sa communaute active et sa capacite a tourner sur du materiel modeste.

> ✔ **HOMELAB VS PRODUCTION**
>
> Un homelab Proxmox peut demarrer avec un seul mini-PC (Intel N100, 16 Go de RAM, SSD NVMe 500 Go) pour moins de 200 EUR. Trois de ces machines forment un cluster complet avec Ceph pour moins de 600 EUR — une fraction du cout d'un lab VMware equivalent.

Les homelabbers utilisent Proxmox pour : - **Apprendre la virtualisation** et le clustering dans un environnement realiste - **Heberger des services** : Pi-hole, Nextcloud, Home Assistant, serveur media - **Experimenter** : tester des configurations Ceph, SDN, HA avant de les deployer en production - **Preparer des certifications** : bien qu'il n'existe pas de certification Proxmox officielle, la maitrise de KVM, Ceph et du reseau Linux est valorisee

Production et datacenter

En production, Proxmox VE est utilise par des entreprises de toutes tailles, des startups aux grandes organisations. Les fonctionnalites enterprise de la v9 (affinite HA, fabrics SDN, Ceph Squid) renforcent sa credibilite pour les workloads critiques.

Cas typiques : - **Hebergement web** : Clusters de VM/conteneurs pour des applications web, avec HA et migration live - **Bases de donnees** : PostgreSQL, MySQL, MongoDB sur des VM avec GPU passthrough et stockage NVMe - **VDI (Virtual Desktop Infrastructure)** : Bureaux virtuels avec GPU SR-IOV pour des dizaines d'utilisateurs - **CI/CD** : Environnements de build et de test ephemeres provisionnes via l'API et Terraform

Edge computing et sites distants

Pour les sites distants avec des contraintes de bande passante ou d'espace physique, Proxmox permet de deployer des clusters compacts :

- **Cluster 2 noeuds + QDevice** : Un cluster de 2 serveurs avec un QDevice distant (pouvant tourner sur un Raspberry Pi ou un petit VPS) pour le quorum. Ideal pour les agences ou succursales.
- **Single-node** : Un noeud unique avec ZFS local pour les sites tres contraints. Pas de HA, mais des sauvegardes vers un PBS distant.

1.5 Migration depuis VMware vSphere / Hyper-V

Le contexte Broadcom a fait de la migration depuis VMware un sujet brulant. Cette section presente la methodologie et les outils pour reussir cette transition.

Methodologie de migration

Une migration reussie suit quatre phases :

Phase 1 — Audit et inventaire - Recenser toutes les VM : OS, ressources (CPU, RAM, disques), dependances reseau - Identifier les VM critiques (ordre de migration : les moins critiques d'abord) - Evaluer la compatibilite : les VM Windows necessitent l'installation de drivers VirtIO - Documenter le reseau : VLAN, regles firewall, adresses IP

Phase 2 — Preparation de l'environnement Proxmox - Installer et configurer le cluster Proxmox (chapitres 2-3) - Configurer le reseau pour reproduire les VLAN existants (chapitre 5) - Configurer le stockage (Ceph ou NFS/iSCSI) (chapitres 4, 8)

Phase 3 — Migration des VM - Exporter les VM depuis vSphere (format OVF/OVA ou VMDK direct) - Convertir les disques avec `qemu-img` - Importer dans Proxmox et ajuster la configuration - Installer les drivers VirtIO dans les VM Windows

Phase 4 — Validation et bascule - Tester chaque VM migree (reseau, stockage, performances) - Basculer les DNS et les routes reseau - Surveiller pendant 48-72h avant de decommissionner l'ancien environnement

Outils de conversion

L'outil principal est `qemu-img`, inclus dans Proxmox :

```
# Convertir un VMDK VMware en format qcow2 Proxmox
qemu-img convert -f vmdk -O qcow2 vm-disk.vmdk vm-disk.qcow2

# Convertir en format raw (meilleures performances sur Ceph/LVM)
qemu-img convert -f vmdk -O raw vm-disk.vmdk vm-disk.raw

# Importer un disque dans une VM Proxmox existante
qm importdisk 100 vm-disk.qcow2 local-lvm
```

Pour les VM Hyper-V, le format VHDX se convertit de la meme maniere :

```
qemu-img convert -f vpc -O qcow2 vm-disk.vhdx vm-disk.qcow2
```

Equivalences fonctionnelles

Equivalences VMware vSphere / Proxmox VE

Fonction VMware	Equivalent Proxmox VE	Notes
vCenter Server	Interface web PVE / API REST / PDM	Pas de serveur de gestion dedie requis
ESXi	Noeud PVE (KVM + LXC)	Chaque noeud est autonome ET membre du cluster
vMotion	Live migration	Migration a chaud sans downtime
Storage vMotion	Migration de disque (move disk)	Migration de disque entre storages
vSAN	Ceph	Stockage distribue integre
NSX	SDN Proxmox (VLAN, VXLAN, EVPN, Fabrics)	Inclus sans licence supplementaire
DRS	CRS (Cluster Resource Scheduler) + affinite	Placement statique (pas encore de rebalancement dynamique)
HA	HA Proxmox + regles d'affinite	Fonctionnellement equivalent
vDP / Veeam	Proxmox Backup Server (PBS)	Deduplication, chiffrement, verification
VM Templates	Templates + Cloud-Init	Provisioning automatise
PowerCLI	pvesh / API REST / proxmoxer (Python)	Automatisation complete

Pieges courants de la migration

> ✗ **PIEGE DE LA V9**
> **Drivers VirtIO obligatoires pour Windows.** Avant de migrer une VM Windows depuis VMware, installez les drivers VirtIO *dans* la VM source (depuis le depot Fedora virtio-win). Sans ces drivers, la VM ne demarrera pas sous Proxmox si vous utilisez des disques VirtIO-SCSI ou des interfaces reseau VirtIO. Alternative : demarrer la VM avec des peripheriques IDE/E1000 (plus lents) puis installer les drivers VirtIO depuis l'interieur.

Autres pieges a eviter : - **Licences Windows** : Les licences OEM liees au materiel VMware ne sont pas transposables. Prevoyez des licences volume ou verifiez vos droits. - **VMware Tools** : Desinstallez VMware Tools avant la migration pour eviter les conflits de drivers. - **Ordre de boot** : Assurez-vous que la VM est configuree en UEFI/OVMF si elle l'etait sous VMware, ou en BIOS/SeaBIOS dans le cas contraire. - **Snapshots VMware** : Consolidez tous les snapshots VMware avant l'export. Les snapshots VMDK ne se convertissent pas directement.

1.6 Panorama de l'ecosysteme Proxmox

Proxmox ne se limite pas a l'hyperviseur. L'ecosysteme comprend plusieurs produits complementaires qui forment une solution d'infrastructure complete.

Proxmox Virtual Environment (PVE)

C'est la plateforme principale, objet de ce livre. PVE fournit : - La virtualisation (KVM + LXC) - La gestion du cluster (Corosync/Kronosnet) - La haute disponibilite (HA + affinite) - Le reseau defini par logiciel (SDN) - Le stockage integre (Ceph, ZFS, LVM) - L'interface web de gestion et l'API REST

Chaque noeud Proxmox est a la fois un hyperviseur, un membre du cluster et un point de gestion. Il n'y a pas de serveur de management dedie comme vCenter — tout noeud peut gerer l'ensemble du cluster.

Proxmox Backup Server (PBS)

PBS est le serveur de sauvegarde dedie de l'ecosysteme Proxmox. Ecrit en Rust pour la performance et la securite memoire, il offre :

- **Deduplication** : Les sauvegardes sont decoupees en blocs (chunks) de 4 Mo. Seuls les blocs uniques sont stockes, ce qui reduit considerablement l'espace necessaire pour les sauvegardes incrementales.
- **Chiffrement client-side** : Chiffrement AES-256-GCM cote client. Le serveur PBS ne voit jamais les donnees en clair — ideal pour les sauvegardes hors site ou chez un prestataire.
- **Verification d'integrite** : Checksums SHA-256 sur chaque bloc. Des jobs de verification periodiques detectent la corruption silencieuse.
- **Compression Zstandard** : Compression rapide (plusieurs Go/s) avec un bon ratio de compression.
- **Restauration granulaire** : Restaurer un fichier individuel depuis une sauvegarde de VM sans restaurer la VM entiere.
- **Support bande** : Archivage sur bandes LTO-5 et superieures pour la retention long terme.

PBS s'integre nativement avec PVE : un storage de type « pbs » dans Proxmox permet de planifier des sauvegardes et des restaurations directement depuis l'interface web.

Proxmox Datacenter Manager (PDM)

PDM est le plus recent ajout a l'ecosysteme (v1.0 en 2025). Il repond au besoin de gerer plusieurs clusters Proxmox depuis une interface unique :

- **Inventaire centralise** : Vue unifiee de tous les noeuds, VM et conteneurs de tous les clusters connectes
- **Monitoring global** : Dashboards de statut et de sante pour l'ensemble des clusters
- **Operations cross-cluster** : Demarrage, arret, redemarrage de VM depuis l'interface PDM
- **Migration inter-cluster** : Capacite de migrer des VM entre clusters differents
- **Gestion SDN centralisee** : Configuration des zones et VNets sur plusieurs clusters

PDM est particulierement utile pour les managed service providers (MSP) gerant des dizaines de clusters clients, ou les grandes organisations avec des clusters sur plusieurs sites geographiques.

> • **EN PRODUCTION**
> PDM est un produit encore jeune. Conservez toujours un acces direct a chaque cluster via l'interface PVE en parallele de PDM. Les details sont couverts au chapitre 13.

Ceph — Le stockage distribue

Ceph est un projet open source independant (maintenu par Red Hat/IBM), integre directement dans Proxmox. Il fournit trois types de stockage :

- **RBD (RADOS Block Device)** : Stockage bloc pour les disques de VM. C'est l'usage principal dans Proxmox. Les disques RBD sont repliques, peuvent etre snapshotes instantanement, et permettent la migration live sans copie de disque.
- **CephFS** : Systeme de fichiers distribue POSIX. Utile pour le stockage partage (templates, ISO, donnees applicatives partagees entre VM).
- **RGW (RADOS Gateway)** : Stockage objet compatible S3. Utile pour les sauvegardes S3, le stockage multimedia ou l'archivage.

L'ensemble est deploye et gere directement depuis l'interface Proxmox via l'outil `pveceph` et le wizard graphique. Le chapitre 8 y est entierement consacre.

SDN — Le reseau defini par logiciel

Le SDN Proxmox permet de gerer les reseaux virtuels de maniere centralisee. Au lieu de configurer manuellement les bridges et les VLAN sur chaque noeud, vous definissez des zones et des VNets dans l'interface web, et Proxmox applique la configuration sur tous les noeuds du cluster.

Le SDN supporte cinq types de zones (Simple, VLAN, QinQ, VXLAN, EVPN) et deux protocoles de fabric (IS-IS, OSPF). Les details sont couverts aux chapitres 5 et 6.

L'ecosysteme Proxmox : PVE au centre fournit la virtualisation et le clustering. Ceph fournit le stockage distribue. Le SDN gere le reseau. PBS assure les sauvegardes. PDM supervise les clusters multi-sites.

Resume du chapitre

Proxmox VE 9, base sur Debian 13 Trixie et le kernel 6.14, est une plateforme de virtualisation open source mature qui combine KVM, LXC, Ceph et SDN dans une solution hyper-convergee integree. En 17 ans d'existence, Proxmox est passe d'un simple frontend web pour KVM/OpenVZ a un ecosysteme complet capable de rivaliser avec les solutions proprietaires les plus etablies.

Points cles a retenir :

- **100 % open source et gratuit** : Pas de licence, pas de limitation fonctionnelle. Le support est optionnel.
- **Proxmox VE 9** apporte les regles d'affinite HA, les fabrics SDN (IS-IS, OSPF), Ceph Squid 19.2 et les snapshots LVM thick.
- **L'architecture hyper-convergee** (HCI) integre compute, stockage et reseau sur chaque noeud, eliminant le SAN.
- **La migration depuis VMware** est facilitee par `qemu-img` et les equivalences fonctionnelles bien documentees.
- **L'ecosysteme** comprend PVE (virtualisation), PBS (sauvegardes), PDM (multi-cluster), Ceph (stockage) et SDN (reseau).

Le chapitre suivant vous guidera dans l'installation de Proxmox VE 9 et la mise a niveau depuis la version 8.

2 Installation de Proxmox VE 9

Ce chapitre guide le lecteur à travers l'intégralité du processus d'installation de Proxmox VE 9, depuis la vérification des prérequis matériels jusqu'à la sécurisation initiale de la plateforme. À la fin de ce chapitre, le premier nœud de l'infrastructure TechFlow SAS sera opérationnel et prêt à accueillir des charges de travail de production.

Proxmox VE 9 repose sur Debian 13 (« Trixie ») et embarque un noyau Linux 6.8 optimisé pour la virtualisation, intégrant KVM pour les machines virtuelles complètes et LXC pour les conteneurs système légers. Cette base solide garantit une compatibilité matérielle étendue et un support long terme.

[Illustration : Vue d'ensemble du processus d'installation de Proxmox VE 9 — de la préparation du média à la mise en production du premier nœud.]

Vue d'ensemble du processus d'installation de Proxmox VE 9 — de la préparation du média à la mise en production du premier nœud.

2.1 Prérequis matériels

Avant de procéder à l'installation, il est indispensable de s'assurer que le serveur cible satisfait aux exigences minimales de Proxmox VE 9. Une configuration sous-dimensionnée n'empêchera pas l'installation, mais elle compromettra les performances et la stabilité en production.

2.1.1 CPU et virtualisation matérielle

Proxmox VE 9 exige un processeur 64 bits compatible x86_64. La virtualisation matérielle doit impérativement être activée dans le BIOS/UEFI pour permettre à KVM de fonctionner correctement. Sans cette extension, les machines virtuelles ne pourront pas démarrer.

Extensions requises :

- **Intel VT-x** (Virtualization Technology for IA-32/64) — Intel Core i3/i5/i7/i9, Xeon série E, Silver, Gold, Platinum
- **AMD-V** (AMD Virtualization, SVM) — AMD Ryzen, EPYC, Threadripper

Extensions fortement recommandées :

- **Intel VT-d** / **AMD-Vi** : virtualisation des entrées-sorties (IOMMU), indispensable pour le PCI passthrough
- **Intel EPT** / **AMD RVI** : tables de pages étendues, améliore significativement les performances mémoire des VM

Pour vérifier la disponibilité de ces extensions sur un système Linux existant :

```
# Vérification des extensions de virtualisation
grep -E 'vmx|svm' /proc/cpuinfo | head -5

# Vérification IOMMU
dmesg | grep -E 'IOMMU|iommu' | head -10

# Alternative : outil systémique
lscpu | grep -E 'Virtualization|Hypervisor'
```

► PROXMOX VE 9 ET LES CPU ARM

Proxmox VE 9 reste exclusivement x86_64. Le support ARM (AArch64) est en cours de développement dans la communauté mais n'est pas disponible dans les versions stables officielles. Les serveurs ARM comme les Ampere Altra ou AWS Graviton ne sont pas supportés.

Recommandations CPU pour TechFlow SAS :

Configuration CPU recommandée par rôle

Rôle du nœud	Minimum	Recommandé	Production TechFlow

Nœud compute léger	4 cœurs / 8 threads	8 cœurs / 16 threads	Intel Xeon Silver 4314 (16c/32t)
Nœud compute intensif	8 cœurs / 16 threads	16 cœurs / 32 threads	Intel Xeon Gold 6338 (32c/64t)
Nœud stockage dédié	4 cœurs / 8 threads	8 cœurs / 16 threads	AMD EPYC 7313P (16c/32t)

La fréquence d'horloge importe moins que le nombre de cœurs pour les charges de virtualisation : un hôte Proxmox gère de nombreuses tâches en parallèle (I/O, réseau, scheduling des vCPU). Privilégiez les processeurs avec un grand cache L3 pour les charges de bases de données.

2.1.2 Mémoire RAM

La RAM est la ressource la plus critique en environnement de virtualisation dense. Proxmox VE lui-même consomme environ 1 à 2 Go de RAM pour le système de base (noyau, services, interface web). Le reste est disponible pour les VM et les conteneurs.

Calcul de la capacité RAM nécessaire :

```
RAM_totale = RAM_système (2 Go) + Σ(RAM_allouée_VM) + marge_surallocation (20 %)
```

Proxmox VE supporte la **surallocation mémoire** (*memory overcommitment*) grâce au mécanisme KSM (Kernel Samepage Merging) qui déduplique les pages mémoire identiques entre VM, et au balloon driver qui permet aux VM de restituer de la mémoire non utilisée.

```
# Vérification de l'état KSM
cat /sys/kernel/mm/ksm/run
# 0 = désactivé, 1 = actif

# Activer KSM manuellement
echo 1 > /sys/kernel/mm/ksm/run

# Statistiques KSM
cat /sys/kernel/mm/ksm/pages_shared
cat /sys/kernel/mm/ksm/pages_sharing
```

Exigences mémoire :

Exigences RAM par niveau de déploiement

Niveau	RAM minimale	RAM recommandée	Cas d'usage
Test/Lab	8 Go	16 Go	Jusqu'à 4-6 VM légères
Petite production	32 Go	64 Go	10-20 VM, quelques CT
Production standard	64 Go	128 Go	30-50 VM, haute densité CT
Production intensive	128 Go	256 Go+	50+ VM, bases de données, SAP

> ⚠ **ECC RAM OBLIGATOIRE EN PRODUCTION**
> Pour tout environnement de production, la RAM ECC (Error-Correcting Code) est indispensable. Une erreur mémoire non corrigée dans un hyperviseur peut corrompre simultanément toutes les VM hébergées. Les serveurs Dell PowerEdge, HPE ProLiant et Supermicro intègrent tous le support ECC. Ne jamais déployer Proxmox VE en production sur du matériel grand public sans ECC.

Pour TechFlow SAS, chaque nœud (pve1, pve2, pve3) est équipé de 256 Go de RAM DDR4 ECC en 8 × 32 Go, offrant une capacité totale de 768 Go pour le cluster, avec une réserve de failover permettant la migration de toutes les VM d'un nœud défaillant vers les deux nœuds restants.

2.1.3 Stockage

Proxmox VE 9 supporte une grande variété de backends de stockage. Le choix du stockage impacte directement les performances des VM, la disponibilité et les capacités de snapshot/sauvegarde.

Disques systèmes (OS Proxmox) :

L'installateur de Proxmox VE crée automatiquement un groupe de volumes LVM nommé `pve` sur le disque système avec la structure suivante :

Volumes LVM créés par l'installateur

Volume logique	Taille typique	Contenu
`/dev/pve/root`	96 Go	Système Debian 13, binaires Proxmox

`/dev/pve/swap`	= RAM (max 8 Go)	Espace de swap
`/dev/pve/data`	Reste du disque	Volume thin-pool LVM pour les VM/CT

Recommandations disques système :

- **SSD NVMe** (≥ 240 Go) : recommandé, latence < 0.1 ms, débit > 2 Go/s
- **SSD SATA** (≥ 240 Go) : acceptable, latence < 1 ms
- **HDD SATA/SAS** (≥ 120 Go) : déconseillé pour le système, acceptable pour les données

Stockage de données (VM, CT, images) :

```
# Inspection des disques disponibles
lsblk -d -o NAME,SIZE,MODEL,TRAN,ROTA
# ROTA=0 : SSD, ROTA=1 : HDD

# Performances de base d'un disque
hdparm -tT /dev/sda

# Test IOPS avec fio
fio --name=randread --ioengine=libaio --iodepth=32 \
    --rw=randread --bs=4k --size=1G --numjobs=4 \
    --runtime=60 --group_reporting --filename=/dev/sdb
```

Pour TechFlow SAS, la configuration de stockage de chaque nœud est la suivante :

Configuration stockage nœuds TechFlow SAS

Emplacement	Matériel	Capacité	Rôle
Slot 0-1	NVMe Samsung 990 Pro x2	2 x 2 To	ZFS mirror — OS + données locales rapides
Slot 2-5	SAS 12G 10K x4	4 x 1.8 To	ZFS RAIDZ1 — données VM standard
Slot 6-9	SATA HDD 7200 x4	4 x 8 To	ZFS RAIDZ2 — archives, sauvegardes

• ZFS ET LA RAM

ZFS utilise l'ARC (Adaptive Replacement Cache) pour mettre en cache les lectures disque en RAM. Par défaut, ZFS peut consommer jusqu'à 50 % de la RAM disponible. Sur un nœud Proxmox, il est recommandé de limiter l'ARC pour laisser de la mémoire aux VM. Règle empirique : allouer 1 Go d'ARC par To de stockage ZFS, avec un minimum de 4 Go et un maximum de 25 % de la RAM totale.

echo "options zfs zfs_arc_max=17179869184" > /etc/modprobe.d/zfs.conf update-initramfs -u

2.1.4 Réseau

Le réseau est la colonne vertébrale d'une infrastructure de virtualisation. Proxmox VE 9 requiert au minimum une interface réseau pour accéder à l'interface de gestion, mais une architecture de production sépare les flux sur plusieurs interfaces dédiées.

Interfaces réseau minimales :

Interfaces réseau par usage

Interface	Débit minimal	Usage
Management	1 GbE	Interface web Proxmox, SSH, API
VM Traffic	10 GbE	Trafic des machines virtuelles
Storage	10 GbE	Ceph, NFS, iSCSI, réplication
Corosync (cluster)	1 GbE dédié	Heartbeat cluster, Corosync
BMC/IPMI	100 MbE	Gestion hors-bande (iDRAC, iLO, IPMI)

Vérification des capacités réseau :

```
# Liste des interfaces et leurs capacités
ip link show

# Détail d'une interface spécifique
ethtool eth0 | grep -E 'Speed|Duplex|Auto-negotiation|Link'
```

```
# Vérification de la prise en charge SR-IOV
lspci | grep -i 'ethernet'
find /sys/class/net/*/device/sriov_numvfs -exec cat {} \; 2>/dev/null
```

✔ **SR-IOV POUR LES PERFORMANCES RÉSEAU EXTRÊMES**

Le SR-IOV (Single Root I/O Virtualization) permet de partager une carte réseau physique en plusieurs fonctions virtuelles (VF) directement accessibles par les VM via PCI passthrough. Les cartes Intel X710, Mellanox ConnectX-5/6 et Broadcom BCM57504 supportent SR-IOV. Chaque VF peut atteindre la quasi-totalité de la bande passante de l'interface physique avec une latence proche du bare-metal. Cette fonctionnalité requiert IOMMU actif.

Pour TechFlow SAS, chaque nœud est équipé de :

- 2 ports 10 GbE (Intel X710) en bond LACP pour le trafic VM
- 2 ports 25 GbE (Mellanox ConnectX-5) pour le stockage Ceph
- 1 port 1 GbE dédié au management Proxmox
- 1 port 1 GbE dédié à Corosync
- 1 port IPMI/iDRAC pour la gestion hors-bande

2.2 Préparation de l'installation

2.2.1 Téléchargement de l'ISO

L'image ISO officielle de Proxmox VE 9 est disponible sur le site officiel de Proxmox. Il est impératif de télécharger l'ISO depuis les sources officielles et de vérifier son intégrité avant toute utilisation.

URL officielle :

```
https://www.proxmox.com/en/downloads/proxmox-virtual-environment/iso
```

Téléchargement en ligne de commande :

```
# Téléchargement de l'ISO Proxmox VE 9.x
wget https://enterprise.proxmox.com/iso/proxmox-ve_9.0-1.iso \
     -O /tmp/proxmox-ve_9.0-1.iso

# Téléchargement du fichier de somme SHA256
wget https://enterprise.proxmox.com/iso/proxmox-ve_9.0-1.iso.sha256sum \
     -O /tmp/proxmox-ve_9.0-1.iso.sha256sum

# Vérification de l'intégrité
cd /tmp
sha256sum -c proxmox-ve_9.0-1.iso.sha256sum
# Résultat attendu : proxmox-ve_9.0-1.iso: OK
```

Vérification de la signature GPG :

```
# Import de la clé GPG Proxmox
wget https://enterprise.proxmox.com/debian/proxmox-release-bookworm.gpg \
     -O /tmp/proxmox-release.gpg
gpg --import /tmp/proxmox-release.gpg

# Vérification de la signature (si le fichier .asc est disponible)
gpg --verify proxmox-ve_9.0-1.iso.asc proxmox-ve_9.0-1.iso
```

✗ **NE JAMAIS UTILISER DES ISO DE SOURCES NON OFFICIELLES**

Des ISO Proxmox modifiées circulent sur des sites de partage non officiels. Ces images peuvent contenir des backdoors, des rootkits ou des configurations malveillantes. Téléchargez exclusivement depuis `proxmox.com` ou les miroirs officiels listés sur ce site. Vérifiez systématiquement la somme SHA256.

2.2.2 Création d'une clé USB bootable

L'ISO Proxmox VE 9 est une image hybride (ISO 9660 + MBR/GPT) qui peut être écrite directement sur une clé USB. Une clé USB de 8 Go minimum est suffisante (l'ISO fait environ 1.2 Go).

Sous Linux avec `dd` :

```
# Identification de la clé USB (attention à ne pas écraser un autre disque !)
lsblk
# Repérer le périphérique correspondant à la clé USB, ex: /dev/sdc

# Démontage de la clé si montée automatiquement
umount /dev/sdc1 2>/dev/null || true

# Écriture de l'ISO sur la clé USB
dd if=/tmp/proxmox-ve_9.0-1.iso \
   of=/dev/sdc \
   bs=1M \
   status=progress \
   conv=fsync

# Synchronisation des tampons d'écriture
sync
```

Sous Linux avec Balena Etcher (interface graphique) :

Balena Etcher est une application multiplateforme qui simplifie la création de média bootable avec vérification automatique de l'intégrité après écriture. Disponible sur `https://www.balena.io/etcher`.

Sous Windows avec Rufus :

Configuration Rufus pour Proxmox VE 9

Paramètre	Valeur
Périphérique	Sélectionner la clé USB (vérifier la lettre de lecteur)
Type de démarrage	Sélection de disque ou image ISO — proxmox-ve_9.0-1.iso
Schéma de partition	GPT (pour UEFI) ou MBR (pour BIOS legacy)
Système de fichiers cible	FAT32 (pour UEFI) ou NTFS
Mode d'écriture	Écriture d'image DD (si demandé par Rufus)

✔ **UTILISATION DE VENTOY POUR LA GESTION MULTI-ISO**

Ventoy est un outil qui permet de créer une clé USB multiboot sans réécrire la clé à chaque changement d'ISO. Il suffit de copier les fichiers ISO dans le répertoire Ventoy de la clé. Proxmox VE 9 est parfaitement compatible avec Ventoy, très pratique pour les administrateurs qui gèrent plusieurs versions ou plusieurs distributions.

Installation de Ventoy sur une clé (Linux)

./Ventoy2Disk.sh -i /dev/sdc

Ensuite, copier simplement l'ISO dans la partition Ventoy

cp proxmox-ve_9.0-1.iso /media/Ventoy/

2.2.3 Configuration du BIOS/UEFI

Avant de démarrer sur la clé USB d'installation, plusieurs paramètres BIOS/UEFI doivent être configurés pour garantir un fonctionnement optimal de Proxmox VE 9.

Paramètres critiques à vérifier et activer :

Paramètres BIOS/UEFI essentiels

Paramètre	Valeur	Emplacement typique (Dell iDRAC / HPE iLO)
Intel VT-x ou AMD-V	Activé	CPU Configuration > Virtualization Technology
Intel VT-d ou AMD-Vi (IOMMU)	Activé	CPU Configuration > VT for Directed I/O

Secure Boot	Désactivé	Boot > Secure Boot Configuration
Legacy/CSM Boot	Désactivé (si UEFI pur)	Boot > CSM Configuration
Hyper-Threading	Activé	CPU Configuration > Hyper-Threading
NUMA	Activé (si multi-socket)	CPU Configuration > NUMA Optimized
C-States / Power Management	Performance ou Custom	Power Management > CPU Power
Turbo Boost / Precision Boost	Activé	CPU Configuration > Turbo Boost
SR-IOV Global Enable	Activé	PCI Subsystem > SR-IOV Global Enable

Ordre de démarrage :

Configurez le périphérique de démarrage principal sur la clé USB ou le lecteur CD/DVD virtuel de la solution iDRAC/iLO. La clé USB doit apparaître en premier dans la séquence de boot.

> **• BIOS LEGACY VS UEFI**
>
> Proxmox VE 9 supporte les deux modes d'amorçage. Cependant, UEFI est fortement recommandé pour les raisons suivantes :
>
> • Support des disques de plus de 2 To comme disque système
> • Support de GPT (table de partition GUID)
> • Démarrage plus rapide
> • Prérequis pour Secure Boot (même si désactivé pour Proxmox)
> • Meilleure compatibilité avec le matériel récent
>
> Si le serveur cible ne supporte que le mode BIOS legacy, Proxmox VE 9 fonctionne mais avec les limitations des partitions MBR (max 2 To, max 4 partitions primaires).

2.3 Installation pas à pas

2.3.1 Démarrage sur l'ISO

Après avoir configuré le BIOS/UEFI et inséré la clé USB bootable, démarrez le serveur. Le menu d'amorçage de Proxmox VE 9 s'affiche après quelques secondes.

[Illustration : Menu de démarrage de l'installateur Proxmox VE 9 avec les options disponibles.]

Menu de démarrage de l'installateur Proxmox VE 9 avec les options disponibles.

Options du menu de démarrage :

- **Install Proxmox VE (Graphical)** : installation graphique interactive — recommandée
- **Install Proxmox VE (Terminal UI)** : installation en mode texte pour les environnements sans GPU/VGA
- **Advanced Options** : accès aux options avancées (mode rescue, installation ZFS, etc.)
- **Boot from Local Disk** : démarrer sur le disque local (utile si la clé USB est oubliée)

Sélectionnez **Install Proxmox VE (Graphical)** et appuyez sur Entrée. Le système charge le noyau Linux 6.8 et les modules nécessaires. Cette phase prend 30 à 90 secondes selon la vitesse de la clé USB.

L'accord de licence EULA (End User License Agreement) s'affiche en premier. Lisez-le et cliquez sur **I agree** pour continuer.

> **► INSTALLATION SANS AFFICHAGE GRAPHIQUE (HEADLESS)**
>
> Sur les serveurs en rack sans GPU ou moniteur, utilisez le KVM-over-IP intégré (iDRAC Virtual Console pour Dell, iLO Remote Console pour HPE, IPMI SOL pour les cartes génériques). Alternativement, l'option « Terminal UI » permet une installation complète via un terminal série (console IPMI SOL à 115200 bauds).

2.3.2 Sélection du disque cible

L'écran de sélection du disque cible liste tous les disques détectés par le système. C'est une étape critique : choisir le mauvais disque détruira irrémédiablement toutes les données qu'il contient.

[Illustration : Interface de sélection du disque cible avec les options de configuration ZFS/LVM.]

Interface de sélection du disque cible avec les options de configuration ZFS/LVM.

Sélection du disque :

Identifiez le disque cible (généralement le premier NVMe ou SSD) par sa taille et son modèle. Cliquez sur **Options** pour accéder aux paramètres avancés de formatage.

Options de système de fichiers disponibles :

Systèmes de fichiers disponibles pour le disque système

Système de fichiers	Avantages	Cas d'usage recommandé
ext4 sur LVM	Simple, mature, compatible	Environnements avec stockage externe (Ceph, SAN)
XFS sur LVM	Bonnes performances, journalisation avancée	Charges intensives en métadonnées
ZFS (RAID-Z)	Intégrité des données, snapshots, compression	Environnements sans stockage externe, protection des données
btrfs (expérimental)	Snapshots, compression, déduplication	Labs, non recommandé en production Proxmox 9

Configuration ZFS recommandée pour TechFlow SAS :

Sélectionnez **ZFS (RAID-1)** pour les nœuds pve1/pve2/pve3. Dans la boîte de dialogue Options :

```
Filesystem      : zfs
RAID Level      : RAID-1 (mirror)
Disks           : /dev/nvme0n1, /dev/nvme1n1
Compress        : lz4
Checksum        : sha256
ashift          : 12 (pour SSD/NVMe avec secteurs 4K)
hdsize          : 0 (utiliser tout le disque)
swapsize        : 8 (Go)
maxroot         : 96 (Go)
minfree         : 16 (Go)
```

• ASHIFT ET LES DISQUES NVME/SSD

Le paramètre `ashift` définit la taille minimale de bloc ZFS. Pour les disques NVMe et SSD 4Kn, utilisez `ashift=12` (blocs de 4096 octets). Pour les HDD et SSD 512e, `ashift=9` (512 octets) ou `ashift=12` sont tous deux acceptables. Un `ashift` trop petit entraîne des performances dégradées et une usure prématurée des SSD. Un `ashift` trop grand gaspille de l'espace sur les petits fichiers mais n'impacte pas les performances.

2.3.3 Configuration réseau initiale

L'écran de configuration réseau demande les paramètres de l'interface de management Proxmox. Cette interface sera utilisée pour accéder à l'interface web et à l'API.

[Illustration : Écran de configuration réseau de l'installateur Proxmox VE 9.]

Écran de configuration réseau de l'installateur Proxmox VE 9.

Paramètres pour pve1 de TechFlow SAS :

Configuration réseau initiale pve1

Paramètre	Valeur
Management Interface	ens1f0 (premier port Intel X710 dédié management)
Hostname (FQDN)	pve1.techflow.local
IP Address	10.10.0.11/24
Gateway	10.10.0.1
DNS Server	10.10.0.5

✔ FQDN ET LA RÉSOLUTION DNS INTERNE

Choisissez un FQDN complet et cohérent dès l'installation. Proxmox VE utilise le hostname pour identifier le nœud dans le cluster, pour les certificats TLS et pour la résolution interne. Un FQDN mal choisi est difficile à modifier après la création du cluster. La convention TechFlow SAS utilise `pveN.techflow.local` avec N = numéro du nœud.

2.3.4 Mot de passe et email administrateur

L'installateur demande ensuite le mot de passe du compte `root` et une adresse email pour les notifications système.

Politique de mot de passe recommandée :

Le mot de passe root doit respecter les critères suivants :

- Longueur minimale : 16 caractères
- Mélange de majuscules, minuscules, chiffres et caractères spéciaux
- Aucun mot du dictionnaire, aucune information personnelle identifiable
- Différent de tous les mots de passe utilisés sur d'autres systèmes

```
# Génération d'un mot de passe fort avec pwgen (post-installation)
pwgen -s -y 24 1

# Alternative avec openssl
openssl rand -base64 24

# Alternative avec /dev/urandom
tr -dc 'A-Za-z0-9!@#$%^&*' < /dev/urandom | head -c 24; echo
```

⚠ STOCKAGE SÉCURISÉ DES MOTS DE PASSE

Ne notez jamais le mot de passe root sur un post-it ou dans un fichier non chiffré. Utilisez un gestionnaire de mots de passe d'entreprise comme Bitwarden, Vaultwarden (auto-hébergé) ou HashiCorp Vault. Pour TechFlow SAS, tous les secrets d'infrastructure sont stockés dans Vaultwarden, hébergé sur pve1 dans un conteneur LXC dédié.

Email administrateur : Entrez l'adresse email de l'équipe d'administration système. Proxmox VE 9 utilise cette adresse pour envoyer les alertes critiques (défaillance disque SMART, échec de sauvegarde, perte d'un nœud cluster). Recommandation : utiliser une liste de diffusion de l'équipe plutôt qu'une adresse individuelle.

2.3.5 Résumé et finalisation

L'installateur affiche un récapitulatif complet de la configuration avant de procéder à l'installation. Vérifiez attentivement chaque paramètre.

[Illustration : Écran de résumé de l'installateur avant la copie des fichiers système.]

Écran de résumé de l'installateur avant la copie des fichiers système.

Points de vérification avant de cliquer sur Install :

- Disque cible correct (vérifiez le modèle et la capacité)
- Système de fichiers choisi (ZFS mirror pour TechFlow)
- Hostname FQDN correct
- Adresse IP, masque, passerelle et DNS corrects
- Fuseau horaire correct (Europe/Paris pour TechFlow SAS)

Cliquez sur **Install**. L'installation copie environ 800 Mo de données sur le disque cible. La progression s'affiche en temps réel. Durée typique : 3 à 8 minutes selon la vitesse du disque et de la clé USB.

À la fin de l'installation, un message de succès s'affiche avec l'URL d'accès à l'interface web. Le système redémarre automatiquement après 5 secondes (ou après clic sur **Reboot**).

> ⚠ **RETIRER LA CLÉ USB AVANT LE REDÉMARRAGE**
> Si le BIOS démarre en priorité sur les périphériques USB, Proxmox démarrera à nouveau sur la clé USB après le redémarrage post-installation. Retirez la clé USB dès que le système commence à s'arrêter, ou reconfigurez l'ordre de démarrage dans le BIOS/UEFI pour que le disque interne soit prioritaire.

2.4 Configuration post-installation

Après le premier démarrage sur le système installé, plusieurs configurations doivent être effectuées avant de mettre la plateforme en production. Ces étapes sont réalisées via l'interface web ou en SSH.

2.4.1 Premier accès à l'interface web

Proxmox VE 9 expose son interface de gestion sur le port HTTPS **8006**. Depuis un navigateur sur le réseau de management :

```
https://10.10.0.11:8006
```

Le navigateur affichera une alerte de certificat auto-signé lors du premier accès. Acceptez l'exception de sécurité temporairement (nous installerons un certificat valide à la section 2.5.3).

Connexion initiale :

Identifiants de connexion initiaux

Champ	Valeur
Utilisateur	root
Mot de passe	(mot de passe défini lors de l'installation)
Realm	Linux PAM standard authentication

[Illustration : Interface web Proxmox VE 9 après la connexion initiale — vue du tableau de bord principal.]

Interface web Proxmox VE 9 après la connexion initiale — vue du tableau de bord principal.

Une notification d'avertissement apparaît en haut de l'interface si aucun abonnement valide n'est détecté : *"You do not have a valid subscription for this server."* Cette notification est normale pour les déploiements sans abonnement entreprise et sera traitée à la section suivante.

Connexion SSH pour les opérations en ligne de commande :

```
# Connexion SSH sur pve1
ssh root@10.10.0.11

# Vérification que Proxmox VE est bien installé
pveversion
# proxmox-ve: 9.0-1 (running kernel: 6.8.4-3-pve)
# pve-manager: 9.0.1
# pve-kernel-6.8: 6.8.4-3

# Statut des services Proxmox
systemctl status pvedaemon pveproxy pvestatd
```

2.4.2 Configuration des dépôts (no-subscription vs enterprise)

Proxmox VE 9 propose trois types de dépôts APT selon le niveau d'abonnement :

Dépôts APT Proxmox VE 9

Dépôt	URL	Description
Enterprise	`enterprise.proxmox.com/debian/pve`	Stable, testé, nécessite abonnement payant
No-Subscription	`download.proxmox.com/debian/pve`	Stable, sans abonnement, non recommandé production
Test	`download.proxmox.com/debian/pve`	Pré-versions, tests uniquement

Configuration pour un environnement sans abonnement :

```
# Désactiver le dépôt enterprise (évite les erreurs apt sans abonnement)
cat /etc/apt/sources.list.d/pve-enterprise.list

# Commenter la ligne enterprise
sed -i 's/^deb/#deb/' /etc/apt/sources.list.d/pve-enterprise.list

# Vérification
cat /etc/apt/sources.list.d/pve-enterprise.list
# #deb https://enterprise.proxmox.com/debian/pve trixie pve-enterprise

# Désactiver également le dépôt Ceph enterprise si non abonné
if [ -f /etc/apt/sources.list.d/ceph.list ]; then
    sed -i 's/^deb/#deb/' /etc/apt/sources.list.d/ceph.list
fi
```

Activation du dépôt no-subscription :

```
# Créer le fichier de dépôt no-subscription
cat > /etc/apt/sources.list.d/pve-no-subscription.list << 'EOF'
deb http://download.proxmox.com/debian/pve trixie pve-no-subscription
EOF

# Ajouter le dépôt Ceph no-subscription
cat > /etc/apt/sources.list.d/ceph-no-subscription.list << 'EOF'
deb http://download.proxmox.com/debian/ceph-reef trixie no-subscription
EOF

# Vérifier les sources Debian de base
cat /etc/apt/sources.list
```

Le fichier `/etc/apt/sources.list` doit contenir les dépôts Debian 13 Trixie standard :

```
deb http://ftp.debian.org/debian trixie main contrib
deb http://ftp.debian.org/debian trixie-updates main contrib
deb http://security.debian.org/debian-security trixie-security main contrib
```

Configuration enterprise (avec abonnement) :

```
# Avec un abonnement valide, conserver le dépôt enterprise
# et y ajouter la clé d'abonnement via l'interface web :
# Datacenter > Subscription > Upload Subscription Key

# Vérifier le statut de l'abonnement
pvesubscription get
```

> • **ABONNEMENT PROXMOX VE EN PRODUCTION**
> Pour un environnement de production, l'abonnement Proxmox VE est fortement recommandé. Il donne accès au dépôt enterprise (plus stable, testé plus rigoureusement), au support technique officiel de l'équipe Proxmox, et finance le développement du logiciel. Les tarifs débutent à 95 EUR/an/socket pour la licence Community. TechFlow SAS dispose d'abonnements Premium sur les trois nœuds.

2.4.3 Mise à jour initiale du système

Immédiatement après la configuration des dépôts, effectuez une mise à jour complète du système.

```
# Mise à jour de l'index des paquets
apt update

# Vérification des mises à jour disponibles
apt list --upgradable 2>/dev/null | head -20

# Application de toutes les mises à jour
apt full-upgrade -y

# Nettoyage des paquets obsolètes
apt autoremove --purge -y
apt autoclean
```

```
# Vérification de la version après mise à jour
pveversion -v
```

Installation des outils utilitaires recommandés :

```
# Outils de monitoring et diagnostic
apt install -y \
    htop \
    iotop \
    iftop \
    nvme-cli \
    smartmontools \
    lm-sensors \
    ethtool \
    tcpdump \
    nmap \
    net-tools \
    dnsutils \
    curl \
    wget \
    vim \
    tmux \
    jq \
    rsync \
    pigz

# Outils de stockage
apt install -y \
    zfsutils-linux \
    lvm2 \
    mdadm \
    parted \
    gdisk

# Outils de performance
apt install -y \
    fio \
    iperf3 \
    sysstat
```

Reboot après mise à jour du noyau :

Si la mise à jour inclut un nouveau noyau Proxmox (paquet `pve-kernel-*`), un redémarrage est nécessaire pour l'activer :

```
# Vérifier si un nouveau noyau a été installé
ls -la /boot/vmlinuz-* | sort

# Vérifier le noyau actuellement en cours d'exécution
uname -r

# Redémarrer si nécessaire
systemctl reboot
```

> ✔ **MISE À JOUR VIA L'INTERFACE WEB**
> L'interface web Proxmox VE 9 offre une section dédiée aux mises à jour sous `Nœud > Updates`. Elle affiche les paquets disponibles, permet d'appliquer les mises à jour en un clic, et affiche les journaux d'installation en temps réel. Cette méthode est pratique pour les mises à jour de routine, mais la ligne de commande reste préférable pour les mises à jour majeures qui nécessitent de surveiller attentivement la sortie.

2.4.4 Configuration du réseau (bridges, bonds, VLANs)

Proxmox VE utilise le fichier `/etc/network/interfaces` (standard Debian) pour la configuration réseau. L'interface web propose un éditeur graphique sous `Nœud > Network`.

Architecture réseau cible pour pve1 :

[Illustration : Architecture réseau du nœud pve1 — bonds, bridges et VLANs de l'infrastructure TechFlow SAS.]

Architecture réseau du nœud pve1 — bonds, bridges et VLANs de l'infrastructure TechFlow SAS.

Configuration complète de `/etc/network/interfaces` pour pve1 :

```
cat > /etc/network/interfaces << 'EOF'
# Loopback
auto lo
iface lo inet loopback

# Interface de management dédiée (1 GbE)
auto ens1f0
iface ens1f0 inet static
    address 10.10.0.11/24
    gateway 10.10.0.1
    dns-nameservers 10.10.0.5 10.10.0.6

# Bridge de management pour l'interface web Proxmox
auto vmbr0
iface vmbr0 inet static
    address 10.10.0.11/24
    gateway 10.10.0.1
    bridge-ports ens1f0
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware no

# Interfaces physiques — VM Traffic (10 GbE, bond LACP)
auto ens2f0
iface ens2f0 inet manual
    bond-master bond0

auto ens2f1
iface ens2f1 inet manual
    bond-master bond0

# Bond LACP pour le trafic VM
auto bond0
iface bond0 inet manual
    bond-slaves ens2f0 ens2f1
    bond-mode 802.3ad
    bond-miimon 100
    bond-xmit-hash-policy layer3+4
    bond-lacp-rate fast

# Bridge principal pour les VM (VLAN-aware)
auto vmbr1
iface vmbr1 inet manual
    bridge-ports bond0
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes
    bridge-vids 2-4094

# Interfaces physiques — Stockage Ceph (25 GbE, Jumbo Frames)
auto ens3f0
iface ens3f0 inet manual
    bond-master bond1
    mtu 9000

auto ens3f1
iface ens3f1 inet manual
    bond-master bond1
    mtu 9000

# Bond actif-passif pour le stockage Ceph
```

```
auto bond1
iface bond1 inet manual
    bond-slaves ens3f0 ens3f1
    bond-mode active-backup
    bond-miimon 100
    bond-primary ens3f0
    mtu 9000

# Interface Ceph réseau public
auto bond1.100
iface bond1.100 inet static
    address 10.10.100.11/24
    vlan-raw-device bond1
    mtu 9000

# Interface Ceph réseau cluster (réplication)
auto bond1.101
iface bond1.101 inet static
    address 10.10.101.11/24
    vlan-raw-device bond1
    mtu 9000

# Interface Corosync (cluster Proxmox) — 1 GbE dédié
auto ens4f0
iface ens4f0 inet static
    address 10.10.200.11/24
EOF
```

Application de la configuration réseau :

```
# Vérification syntaxique avant application
ifup --no-act -a -v

# Application de la nouvelle configuration
# ATTENTION : peut couper la connexion si l'interface de management change
ifreload -a

# Vérification post-application
ip addr show
ip route show
bridge link show
```

Configuration et vérification des VLANs :

```
# Lister les VLANs actifs sur le bridge vmbr1
bridge vlan show dev vmbr1

# Ajouter un VLAN spécifique manuellement si nécessaire
bridge vlan add vid 200 dev vmbr1

# Supprimer un VLAN
bridge vlan del vid 200 dev vmbr1

# Vérifier la connectivité inter-nœuds sur le réseau Ceph
ping -c 4 -M do -s 8972 10.10.100.12   # Test jumbo frames vers pve2
ping -c 4 -M do -s 8972 10.10.101.12   # Test réseau cluster Ceph vers pve2
```

> • **JUMBO FRAMES ET CEPH**
> Les réseaux de stockage Ceph bénéficient significativement des Jumbo Frames (MTU 9000). Cette configuration réduit l'overhead CPU de traitement réseau et améliore les débits de réplication. Assurez-vous que **tous** les équipements réseau sur le chemin (switches, HBAs, câbles) supportent et sont configurés avec MTU 9000. Une incohérence de MTU entre nœuds cause des pertes de paquets silencieuses difficiles à diagnostiquer.

2.4.5 Configuration NTP et timezone

La synchronisation temporelle est critique pour les clusters distribués. Proxmox VE utilise `chrony` comme client NTP.

Configuration du fuseau horaire :

```
# Vérifier le fuseau horaire actuel
timedatectl status

# Lister les fuseaux disponibles (Europe)
timedatectl list-timezones | grep Europe

# Configurer le fuseau horaire Europe/Paris
timedatectl set-timezone Europe/Paris

# Vérification
date
# Tue Mar 17 14:30:45 CET 2026
```

Configuration de chrony pour synchronisation NTP :

```
# Éditer la configuration chrony
cat > /etc/chrony/chrony.conf << 'EOF'
# Serveurs NTP TechFlow SAS (serveurs internes Stratum 2)
server 10.10.0.5 iburst prefer
server 10.10.0.6 iburst

# Serveurs NTP publics en fallback
server 0.fr.pool.ntp.org iburst
server 1.fr.pool.ntp.org iburst
server 2.fr.pool.ntp.org iburst
server 3.fr.pool.ntp.org iburst

# Permettre la synchronisation même si l'offset est important au démarrage
makestep 1.0 3

# Fichier de dérive pour la compensation matérielle
driftfile /var/lib/chrony/drift

# Journalisation
logdir /var/log/chrony
log measurements statistics tracking
EOF

# Redémarrer chrony
systemctl restart chrony
systemctl enable chrony

# Vérifier la synchronisation
chronyc tracking
chronyc sources -v
chronyc sourcestats

# Forcer une synchronisation immédiate
chronyc makestep
```

Sortie attendue de `chronyc tracking` :

```
Reference ID    : 0A0A0005 (10.10.0.5)
Stratum         : 3
Ref time (UTC)  : Tue Mar 17 13:30:45 2026
System time     : 0.000000012 seconds slow of NTP time
Last offset     : -0.000000045 seconds
RMS offset      : 0.000000089 seconds
Frequency       : 1.234 ppm slow
Residual freq   : -0.001 ppm
Skew            : 0.045 ppm
Root delay      : 0.001234567 seconds
Root dispersion : 0.000567890 seconds
Update interval : 64.1 seconds
Leap status     : Normal
```

> ⚠ **IMPORTANCE DU NTP POUR CEPH ET LES CLUSTERS**
> Une désynchronisation temporelle supérieure à 2 secondes entre les nœuds d'un cluster Ceph entraîne l'arrêt immédiat des opérations I/O Ceph avec l'erreur `clock skew detected`. Pour un cluster Proxmox + Corosync, une désynchronisation peut provoquer des split-brain et des fencing intempestifs. Le NTP n'est pas optionnel en environnement clusterisé.

2.5 Sécurisation initiale

La sécurisation d'un hyperviseur est une priorité absolue. Un hyperviseur compromis expose l'intégralité des VM et données hébergées. Cette section couvre les mesures de sécurité de base qui doivent être appliquées avant toute mise en production.

2.5.1 Désactivation du compte root direct

Le compte root ne doit pas être utilisable pour les connexions directes en production. Au lieu de cela, créez un compte utilisateur PAM dédié à l'administration avec élévation sudo.

```
# Vérifier la configuration SSH actuelle
grep -E 'PermitRootLogin|PasswordAuthentication' /etc/ssh/sshd_config

# Sauvegarder la configuration originale
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak

# Appliquer les directives de sécurité SSH
cat >> /etc/ssh/sshd_config << 'EOF'

# Configuration de sécurité TechFlow SAS — appliquée le 2026-03-17
PermitRootLogin prohibit-password
PasswordAuthentication no
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
MaxAuthTries 3
LoginGraceTime 30
ClientAliveInterval 300
ClientAliveCountMax 2
AllowGroups sudo adminpve
Banner /etc/ssh/banner.txt
EOF

# Créer une bannière SSH légale
cat > /etc/ssh/banner.txt << 'EOF'
*******************************************************************************
*                  SYSTEME PRIVE — ACCES RESTREINT                            *
*  Toute connexion non autorisée est interdite et peut faire l'objet de       *
*  poursuites judiciaires. Vos actions sont enregistrées et surveillées.      *
*  TechFlow SAS — Département Systèmes et Infrastructure                      *
*******************************************************************************
EOF

# Vérifier la syntaxe de la configuration SSH
sshd -t

# Redémarrer SSH (conserver une session active pendant la vérification)
systemctl restart ssh
```

2.5.2 Création d'un utilisateur administrateur PAM

```
# Créer l'utilisateur système adminpve
useradd -m -s /bin/bash -c "Administrateur Proxmox VE TechFlow SAS" adminpve

# Définir un mot de passe fort (stocké dans Vaultwarden)
passwd adminpve

# Ajouter au groupe sudo
usermod -aG sudo adminpve

# Créer le répertoire SSH et configurer les clés
```

```
mkdir -p /home/adminpve/.ssh
chmod 700 /home/adminpve/.ssh
chown adminpve:adminpve /home/adminpve/.ssh

# Installer la clé publique SSH (remplacer par la clé réelle)
cat > /home/adminpve/.ssh/authorized_keys << 'EOF'
ssh-ed25519 AAAA... votre_cle_publique_ed25519_ici ... adminpve@workstation-techflow
EOF

chmod 600 /home/adminpve/.ssh/authorized_keys
chown adminpve:adminpve /home/adminpve/.ssh/authorized_keys
```

Ajout de l'utilisateur dans l'interface Proxmox VE :

```
# Créer l'utilisateur PAM dans Proxmox (realm pam)
pveum user add adminpve@pam \
    --comment "Administrateur TechFlow SAS" \
    --email "sysadmin@techflow.local"

# Attribuer le rôle Administrateur sur l'ensemble du datacenter
pveum acl modify / --user adminpve@pam --role Administrator

# Vérifier la liste des utilisateurs
pveum user list

# Vérifier les ACL
pveum acl list
```

Configuration sudo pour adminpve :

```
# Créer une règle sudo dédiée
cat > /etc/sudoers.d/adminpve << 'EOF'
# Administrateur Proxmox VE TechFlow SAS
# Génère un log pour chaque commande sudo
Defaults:adminpve log_output
Defaults:adminpve logfile=/var/log/sudo-adminpve.log

adminpve ALL=(ALL) ALL
EOF

chmod 440 /etc/sudoers.d/adminpve

# Vérification syntaxique obligatoire
visudo -c
# parsed OK
```

2.5.3 Configuration du certificat TLS

Par défaut, Proxmox VE génère un certificat auto-signé. Pour un environnement de production, un certificat valide est indispensable pour éviter les erreurs de navigateur et assurer l'authenticité du serveur.

Option 1 : Certificat Let's Encrypt via ACME :

```
# Configuration du compte ACME (Let's Encrypt)
pvesh set /nodes/pve1/config \
    --acmedomain0 "domain=pve1.techflow.com,plugin=standalone"

# Enregistrement du compte ACME
pvenode acme account register default admin@techflow.com

# Déclenchement de la demande de certificat
pvenode acme cert order

# Vérification du certificat installé
openssl x509 -in /etc/pve/local/pveproxy-ssl.pem \
    -text -noout | grep -E 'Subject:|Issuer:|Not After'
```

Option 2 : Certificat d'une PKI interne (recommandé pour TechFlow SAS) :

```
# Générer une clé privée RSA 4096 bits
openssl genrsa -out /etc/pve/local/pveproxy-ssl.key 4096
chmod 600 /etc/pve/local/pveproxy-ssl.key

# Générer la CSR avec SAN (Subject Alternative Names)
openssl req -new \
    -key /etc/pve/local/pveproxy-ssl.key \
    -out /tmp/pve1-csr.pem \
    -subj "/C=FR/ST=Ile-de-France/L=Paris/O=TechFlow SAS/OU=IT Infrastructure/
CN=pve1.techflow.local" \
    -addext "subjectAltName=DNS:pve1.techflow.local,IP:10.10.0.11"

# Soumettre la CSR à la PKI interne TechFlow
# La PKI retourne le certificat signé dans /tmp/pve1-cert.pem

# Installer le certificat (certificat + chaîne CA)
cat /tmp/pve1-cert.pem /tmp/ca-chain.pem > /etc/pve/local/pveproxy-ssl.pem

# Recharger le proxy Proxmox
systemctl reload pveproxy

# Vérification
openssl s_client -connect 10.10.0.11:8006 -showcerts 2>/dev/null | \
    openssl x509 -noout -subject -issuer -dates
```

Renouvellement automatique du certificat (cron) :

```
# Planifier le renouvellement ACME (si utilisé)
cat > /etc/cron.d/pve-acme-renewal << 'EOF'
# Renouvellement automatique du certificat Let's Encrypt
# Exécuté tous les lundis à 02h30
30 2 * * 1 root pvenode acme cert renew >> /var/log/pve-acme.log 2>&1
EOF
```

2.5.4 Firewall de base Proxmox

Proxmox VE 9 intègre un firewall basé sur nftables gérable via l'interface web ou la ligne de commande. Il opère à deux niveaux : nœud (*host firewall*) et VM/CT (*VM firewall*).

Activation et configuration du firewall au niveau datacenter :

```
# Fichier de configuration datacenter
cat > /etc/pve/firewall/cluster.fw << 'EOF'
[OPTIONS]
enable: 1
policy_in: DROP
policy_out: ACCEPT
log_ratelimit: enable

[RULES]
# SSH depuis le réseau de management uniquement
IN ACCEPT -source 10.10.0.0/24 -dport 22 -proto tcp -log info
# Interface web Proxmox depuis le réseau de management
IN ACCEPT -source 10.10.0.0/24 -dport 8006 -proto tcp -log info
# ICMP depuis le réseau de management
IN ACCEPT -source 10.10.0.0/24 -proto icmp
# Corosync entre nœuds du cluster (réseau dédié)
IN ACCEPT -source 10.10.200.0/24 -dport 5404:5405 -proto udp
# SPICE console depuis le réseau management
IN ACCEPT -source 10.10.0.0/24 -dport 3128 -proto tcp
# VNC console depuis le réseau management
IN ACCEPT -source 10.10.0.0/24 -dport 5900:5999 -proto tcp
# Migration live de VM entre nœuds
IN ACCEPT -source 10.10.0.0/24 -dport 60000:60050 -proto tcp
EOF
```

Activation du firewall au niveau du nœud pve1 :

```
# Fichier de configuration spécifique au nœud
mkdir -p /etc/pve/nodes/pve1
```

```
cat > /etc/pve/nodes/pve1/host.fw << 'EOF'
[OPTIONS]
enable: 1
log_level_in: info
log_level_out: nolog

[RULES]
# SNMP depuis le serveur de supervision (Prometheus node_exporter)
IN ACCEPT -source 10.10.0.20 -dport 9100 -proto tcp -log debug
# Trafic Ceph depuis les autres nœuds
IN ACCEPT -source 10.10.100.0/24 -dport 6789 -proto tcp
IN ACCEPT -source 10.10.100.0/24 -dport 6800:7300 -proto tcp
EOF

# Démarrer le service firewall
pve-firewall start
systemctl enable pve-firewall

# Vérifier le statut
pve-firewall status

# Lister les règles actives nftables
nft list ruleset | head -50
```

Installation et configuration de Fail2Ban :

```
apt install -y fail2ban

# Filtre pour l'interface web Proxmox
cat > /etc/fail2ban/filter.d/proxmox.conf << 'EOF'
[Definition]
failregex = pvedaemon\[.*\]: authentication failure; rhost=<HOST> user=.* msg=.*
ignoreregex =
EOF

# Configuration des jails
cat > /etc/fail2ban/jail.d/techflow.conf << 'EOF'
[DEFAULT]
bantime  = 3600
findtime = 600
maxretry = 3

[sshd]
enabled  = true
port     = 22
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 3
bantime  = 7200

[proxmox]
enabled  = true
port     = 8006
filter   = proxmox
logpath  = /var/log/daemon.log
maxretry = 5
bantime  = 7200
EOF

systemctl enable fail2ban
systemctl start fail2ban

# Vérifier le statut des jails
fail2ban-client status
fail2ban-client status sshd
```

2.6 Installation du premier nœud TechFlow SAS

2.6.1 Plan réseau de l'infrastructure TechFlow

L'infrastructure TechFlow SAS est composée de trois nœuds Proxmox VE 9 en cluster haute disponibilité, hébergés dans un datacenter Tier III à Paris. Voici le plan d'adressage réseau complet.

[Illustration : Plan réseau complet de l'infrastructure TechFlow SAS — trois nœuds en cluster avec séparation des flux.]

Plan réseau complet de l'infrastructure TechFlow SAS — trois nœuds en cluster avec séparation des flux.

Plan d'adressage IP complet :

Plan d'adressage réseau TechFlow SAS

Réseau	Sous-réseau	pve1	pve2	pve3
Management Proxmox	10.10.0.0/24	10.10.0.11	10.10.0.12	10.10.0.13
Trafic VM (natif)	10.10.1.0/24	10.10.1.11	10.10.1.12	10.10.1.13
Ceph Public	10.10.100.0/24	10.10.100.11	10.10.100.12	10.10.100.13
Ceph Cluster	10.10.101.0/24	10.10.101.11	10.10.101.12	10.10.101.13
Corosync	10.10.200.0/24	10.10.200.11	10.10.200.12	10.10.200.13
IPMI/iDRAC	10.10.250.0/24	10.10.250.11	10.10.250.12	10.10.250.13

VLANs pour les VM de production :

VLANs production TechFlow SAS

VLAN ID	Nom	Sous-réseau	Usage
10	DMZ	192.168.10.0/24	Services exposés (reverse proxy, WAF)
20	Production	192.168.20.0/24	Applications de production
30	Bases de données	192.168.30.0/24	Serveurs BDD (accès restreint)
40	Développement	192.168.40.0/24	Environnements de dev/test
50	Sauvegarde	192.168.50.0/24	Proxmox Backup Server
100	Monitoring	192.168.100.0/24	Prometheus, Grafana, Alertmanager

Services réseau nécessaires :

Services réseau de l'infrastructure TechFlow SAS

Service	IP	Description
DNS primaire	10.10.0.5	Bind9 — résolution interne techflow.local
DNS secondaire	10.10.0.6	Bind9 secondaire
NTP primaire	10.10.0.5	Chrony stratum 2, source : GPS/PPS
Gateway	10.10.0.1	Routeur FortiGate — accès Internet
Backup Server	10.10.0.20	Proxmox Backup Server 3.x
vSwitch core	N/A	Cisco Nexus 9300 — agrège les 3 nœuds

Résolution DNS interne à configurer :

```
# Entrées DNS à ajouter dans la zone techflow.local (Bind9 sur 10.10.0.5)
# Fichier /etc/bind/zones/techflow.local

# Enregistrements A
pve1.techflow.local.    IN A    10.10.0.11
pve2.techflow.local.    IN A    10.10.0.12
pve3.techflow.local.    IN A    10.10.0.13

# Enregistrements PTR (zone 10.10.0.x inverse)
11.0.10.10.in-addr.arpa.  IN PTR  pve1.techflow.local.
```

```
12.0.10.10.in-addr.arpa.  IN PTR  pve2.techflow.local.
13.0.10.10.in-addr.arpa.  IN PTR  pve3.techflow.local.
```

```
# Vérifier la résolution DNS depuis pve1
nslookup pve1.techflow.local 10.10.0.5
nslookup pve2.techflow.local 10.10.0.5
nslookup 10.10.0.12 10.10.0.5  # Résolution inverse

# Tester la résolution avec dig
dig @10.10.0.5 pve1.techflow.local
dig @10.10.0.5 -x 10.10.0.11
```

2.6.2 Installation du nœud pve1

En suivant toutes les étapes décrites dans ce chapitre, voici le script récapitulatif de post-installation pour pve1. Ce script automatise toutes les tâches de configuration post-installation.

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════════
# Script de post-installation pve1 — TechFlow SAS
# Version : Proxmox VE 9.x / Debian 13 Trixie
# Auteur  : Équipe Infrastructure TechFlow SAS
# Date    : 2026-03-17
# ═══════════════════════════════════════════════════════════════════════
# Usage : bash postinstall-pve1.sh
# Doit être exécuté en tant que root sur pve1 fraîchement installé
# ═══════════════════════════════════════════════════════════════════════

set -euo pipefail
readonly LOGFILE="/var/log/techflow-postinstall-$(date +%Y%m%d-%H%M%S).log"
readonly NODE_IP="10.10.0.11"
readonly NODE_HOSTNAME="pve1.techflow.local"
readonly CEPH_PUBLIC_IP="10.10.100.11"
readonly CEPH_CLUSTER_IP="10.10.101.11"
readonly COROSYNC_IP="10.10.200.11"

exec > >(tee -a "$LOGFILE") 2>&1
log() { echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*"; }
die() { log "ERREUR FATALE: $*"; exit 1; }

log "═══ Début post-installation pve1 TechFlow SAS ═══"
log "Log complet : $LOGFILE"

# Vérifications préalables
[[ $(id -u) -eq 0 ]] || die "Ce script doit être exécuté en root"
[[ $(hostname -f) == "$NODE_HOSTNAME" ]] || \
    log "AVERTISSEMENT: hostname ne correspond pas ($NODE_HOSTNAME attendu)"

# 1. Configuration des dépôts APT
log "[1/8] Configuration des dépôts APT..."
sed -i 's/^deb/#deb/' /etc/apt/sources.list.d/pve-enterprise.list
[[ -f /etc/apt/sources.list.d/ceph.list ]] && \
    sed -i 's/^deb/#deb/' /etc/apt/sources.list.d/ceph.list

cat > /etc/apt/sources.list.d/pve-no-subscription.list << 'APT'
deb http://download.proxmox.com/debian/pve trixie pve-no-subscription
APT

cat > /etc/apt/sources.list.d/ceph-no-subscription.list << 'APT'
deb http://download.proxmox.com/debian/ceph-reef trixie no-subscription
APT

# 2. Mise à jour du système
log "[2/8] Mise à jour du système..."
apt-get update -q
DEBIAN_FRONTEND=noninteractive apt-get full-upgrade -y -q
DEBIAN_FRONTEND=noninteractive apt-get autoremove --purge -y -q

# 3. Installation des outils
```

```
log "[3/8] Installation des outils utilitaires…"
DEBIAN_FRONTEND=noninteractive apt-get install -y -q \
    htop iotop iftop nvme-cli smartmontools lm-sensors ethtool \
    tcpdump nmap net-tools dnsutils curl wget vim tmux jq \
    rsync pigz fio iperf3 sysstat fail2ban

# 4. Fuseau horaire et NTP
log "[4/8] Configuration du fuseau horaire et NTP…"
timedatectl set-timezone Europe/Paris

cat > /etc/chrony/chrony.conf << 'CHRONY'
server 10.10.0.5 iburst prefer
server 10.10.0.6 iburst
server 0.fr.pool.ntp.org iburst
server 1.fr.pool.ntp.org iburst
makestep 1.0 3
driftfile /var/lib/chrony/drift
logdir /var/log/chrony
CHRONY

systemctl restart chrony
chronyc makestep 2>/dev/null || true

# 5. Optimisations noyau
log "[5/8] Application des optimisations noyau…"
cat > /etc/sysctl.d/99-techflow-proxmox.conf << 'SYSCTL'
# Performances réseau
net.core.rmem_max = 134217728
net.core.wmem_max = 134217728
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 134217728
net.core.netdev_max_backlog = 250000
net.ipv4.tcp_mtu_probing = 1

# Sécurité réseau
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv4.conf.all.log_martians = 1

# Virtualisation
vm.swappiness = 10
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10
kernel.panic = 10
kernel.panic_on_oops = 1
SYSCTL

sysctl -p /etc/sysctl.d/99-techflow-proxmox.conf -q

# 6. Création de l'utilisateur adminpve
log "[6/8] Création de l'utilisateur adminpve…"
if ! id adminpve &>/dev/null; then
    useradd -m -s /bin/bash \
        -c "Administrateur Proxmox VE TechFlow SAS" adminpve
    log "Utilisateur adminpve créé (définir le mot de passe manuellement : passwd
adminpve)"
fi

usermod -aG sudo adminpve

mkdir -p /home/adminpve/.ssh
chmod 700 /home/adminpve/.ssh
chown adminpve:adminpve /home/adminpve/.ssh

# Créer l'utilisateur dans Proxmox
```

```
pveum user add adminpve@pam \
    --comment "Administrateur TechFlow SAS" \
    --email "sysadmin@techflow.local" 2>/dev/null || true
pveum acl modify / --user adminpve@pam --role Administrator 2>/dev/null || true

# 7. Configuration fail2ban
log "[7/8] Configuration de fail2ban ..."
cat > /etc/fail2ban/filter.d/proxmox.conf << 'F2B'
[Definition]
failregex = pvedaemon\[.*\]: authentication failure; rhost=<HOST> user=.* msg=.*
ignoreregex =
F2B

cat > /etc/fail2ban/jail.d/techflow.conf << 'F2B'
[DEFAULT]
bantime  = 3600
findtime = 600
maxretry = 3

[sshd]
enabled  = true
maxretry = 3
bantime  = 7200

[proxmox]
enabled  = true
port     = 8006
filter   = proxmox
logpath  = /var/log/daemon.log
maxretry = 5
bantime  = 7200
F2B

systemctl enable fail2ban
systemctl restart fail2ban

# 8. Activation du firewall Proxmox
log "[8/8] Configuration du firewall Proxmox ..."
mkdir -p /etc/pve/nodes/pve1

# Vérifier que le cluster pve est monté
if mountpoint -q /etc/pve; then
    cat > /etc/pve/firewall/cluster.fw << 'FW'
[OPTIONS]
enable: 1
policy_in: DROP
policy_out: ACCEPT

[RULES]
IN ACCEPT -source 10.10.0.0/24 -dport 22 -proto tcp -log info
IN ACCEPT -source 10.10.0.0/24 -dport 8006 -proto tcp -log info
IN ACCEPT -source 10.10.0.0/24 -proto icmp
IN ACCEPT -source 10.10.200.0/24 -dport 5404:5405 -proto udp
IN ACCEPT -source 10.10.0.0/24 -dport 5900:5999 -proto tcp
IN ACCEPT -source 10.10.0.0/24 -dport 60000:60050 -proto tcp
FW

    pve-firewall start 2>/dev/null || true
    systemctl enable pve-firewall 2>/dev/null || true
else
    log "AVERTISSEMENT: /etc/pve non monté — firewall configuré après cluster
setup"
fi

log "═══ Post-installation pve1 terminée avec succès ═══"
log "Actions manuelles restantes :"
log "  1. Définir le mot de passe adminpve : passwd adminpve"
log "  2. Copier la clé SSH publique dans /home/adminpve/.ssh/authorized_keys"
```

```
log "  3. Installer le certificat TLS (section 2.5.3)"
log "  4. Configurer la résolution DNS (ajouter pve1.techflow.local dans Bind9)"
log "  5. Redémarrer si un nouveau noyau a été installé : uname -r"
log "Log complet disponible : $LOGFILE"
```

2.6.3 Vérification de l'installation

Après avoir complété toutes les étapes de configuration, effectuez une vérification complète de l'installation avant de déclarer le nœud prêt pour la production.

Vérifications manuelles :

```
# Version Proxmox VE et noyau
pveversion -v
uname -a

# Statut des services Proxmox
for svc in pvedaemon pveproxy pvestatd pve-cluster pve-ha-crm pve-ha-lrm; do
    printf "%-20s : %s\n" "$svc" "$(systemctl is-active $svc 2>/dev/null)"
done

# Utilisation mémoire
free -h

# Statut ZFS
zpool status
zfs list

# Connectivité réseau
ip addr show | grep -E 'inet |UP'
ping -c 2 10.10.0.1        # Gateway
ping -c 2 10.10.0.5        # DNS
nslookup pve1.techflow.local 10.10.0.5

# NTP
chronyc tracking | grep -E 'Reference|System time|RMS'

# SMART des disques
for disk in /dev/nvme[0-9]; do
    [ -b "$disk" ] && nvme smart-log "$disk" 2>/dev/null | \
        grep -E 'critical_warning|available_spare|temperature'
done
```

Script de vérification automatisée :

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════════
# Script de vérification post-installation pve1 — TechFlow SAS
# Usage : bash verify-pve1.sh
# Retourne : 0 = OK, 1 = erreurs détectées
# ═══════════════════════════════════════════════════════════════════

ERRORS=0; WARNINGS=0; PASS=0

ok()   { echo "  [OK]   $1"; PASS=$((PASS+1)); }
fail() { echo "  [FAIL] $1 — $2"; ERRORS=$((ERRORS+1)); }
warn() { echo "  [WARN] $1 — $2"; WARNINGS=$((WARNINGS+1)); }

check_active() {
    local name="$1" svc="$2"
    if systemctl is-active "$svc" &>/dev/null; then
        ok "$name actif"
    else
        fail "$name inactif" "systemctl start $svc"
    fi
}

check_cmd() {
    local desc="$1" cmd="$2" pattern="$3"
    if eval "$cmd" 2>/dev/null | grep -q "$pattern"; then
```

```
        ok "$desc"
    else
        fail "$desc" "résultat inattendu"
    fi
}

echo "═══════════════════════════════════════════════════════════"
echo "  Vérification post-installation pve1 TechFlow SAS"
echo "  $(date)"
echo "═══════════════════════════════════════════════════════════"
echo ""

echo "-- Services Proxmox VE --"
check_active "pvedaemon" "pvedaemon"
check_active "pveproxy" "pveproxy"
check_active "pvestatd" "pvestatd"
check_active "pve-cluster" "pve-cluster"

echo ""
echo "-- Version du système --"
check_cmd "Proxmox VE 9.x installé" "pveversion" "proxmox-ve: 9\."
check_cmd "Noyau Linux 6.x actif" "uname -r" "^6\."
check_cmd "Debian 13 Trixie" "cat /etc/debian_version" "13\."

echo ""
echo "-- Réseau --"
check_cmd "Interface vmbr0 UP" "ip link show vmbr0" "state UP"
check_cmd "IP management 10.10.0.11" "ip addr show vmbr0" "10\.10\.0\.11"
check_cmd "Passerelle joignable" "ping -c 1 -W 2 10.10.0.1" "1 received"
check_cmd "DNS primaire joignable" "ping -c 1 -W 2 10.10.0.5" "1 received"
check_cmd "Résolution DNS interne" "nslookup pve1.techflow.local 10.10.0.5"
"Address"

echo ""
echo "-- Stockage ZFS --"
check_cmd "Pool ZFS pve ONLINE" "zpool status pve" "state: ONLINE"
check_cmd "Volume root disponible" "zfs list pve/root" "pve/root"

echo ""
echo "-- Synchronisation NTP --"
check_cmd "Chrony synchronisé" "chronyc tracking" "Reference ID"
check_cmd "Offset NTP < 1s" "chronyc tracking" "System time.*0\."

echo ""
echo "-- Sécurité --"
check_cmd "SSH PasswordAuth désactivé" \
    "sshd -T | grep passwordauthentication" "passwordauthentication no"
check_cmd "Utilisateur adminpve existe" "id adminpve" "uid="
check_cmd "adminpve dans Proxmox" "pveum user list" "adminpve@pam"
check_active "fail2ban" "fail2ban"
check_cmd "Firewall Proxmox actif" "pve-firewall status" "running"

echo ""
echo "-- Optimisations noyau --"
check_cmd "vm.swappiness = 10" "sysctl vm.swappiness" "= 10"
check_cmd "net.core.rmem_max élevé" "sysctl net.core.rmem_max" "134217728"

echo ""
echo "═══════════════════════════════════════════════════════════"
printf "  Résultats : %d OK | %d avertissements | %d erreurs\n" \
    "$PASS" "$WARNINGS" "$ERRORS"
echo "═══════════════════════════════════════════════════════════"

if [ "$ERRORS" -eq 0 ]; then
    echo ""
    echo "  SUCCES : pve1 est prêt pour la mise en production."
    echo "  Prochaine étape : installation de pve2 et pve3,"
    echo "  puis création du cluster (Chapitre 3)."
```

```
    echo ""
    exit 0
else
    echo ""
    echo "  ECHEC : $ERRORS erreur(s) détectée(s)."
    echo "  Corriger les problèmes avant de continuer."
    echo ""
    exit 1
fi
```

Résultat attendu après une installation réussie :

```
============================================================
  Vérification post-installation pve1 TechFlow SAS
  Tue Mar 17 14:45:23 CET 2026
============================================================

-- Services Proxmox VE --
  [OK]   pvedaemon actif
  [OK]   pveproxy actif
  [OK]   pvestatd actif
  [OK]   pve-cluster actif

-- Version du système --
  [OK]   Proxmox VE 9.x installé
  [OK]   Noyau Linux 6.x actif
  [OK]   Debian 13 Trixie

-- Réseau --
  [OK]   Interface vmbr0 UP
  [OK]   IP management 10.10.0.11
  [OK]   Passerelle joignable
  [OK]   DNS primaire joignable
  [OK]   Résolution DNS interne

-- Stockage ZFS --
  [OK]   Pool ZFS pve ONLINE
  [OK]   Volume root disponible

-- Synchronisation NTP --
  [OK]   Chrony synchronisé
  [OK]   Offset NTP < 1s

-- Sécurité --
  [OK]   SSH PasswordAuth désactivé
  [OK]   Utilisateur adminpve existe
  [OK]   adminpve dans Proxmox
  [OK]   fail2ban
  [OK]   Firewall Proxmox actif

-- Optimisations noyau --
  [OK]   vm.swappiness = 10
  [OK]   net.core.rmem_max élevé

============================================================
  Résultats : 22 OK | 0 avertissements | 0 erreurs
============================================================

  SUCCES : pve1 est prêt pour la mise en production.
  Prochaine étape : installation de pve2 et pve3,
  puis création du cluster (Chapitre 3).
```

► PROCHAINES ÉTAPES — CLUSTER ET STOCKAGE DISTRIBUÉ

Le nœud pve1 est maintenant installé, configuré et sécurisé. Les chapitres suivants couvrent :

- **Chapitre 3** : Création du cluster Proxmox VE 3 nœuds avec Corosync — intégration de pve2 et pve3, quorum, fencing
- **Chapitre 4** : Déploiement de Ceph sur le cluster — stockage distribué haute disponibilité, OSD, pools, règles CRUSH
- **Chapitre 5** : Configuration de la haute disponibilité (HA) et des politiques de migration automatique

Pour les nœuds pve2 et pve3, répétez les étapes de ce chapitre en adaptant les adresses IP (10.10.0.12/13, 10.10.100.12/13, etc.) et le hostname (pve2.techflow.local, pve3.techflow.local) avant de les joindre au cluster.

Résumé du chapitre

Ce chapitre a couvert l'intégralité du processus d'installation et de configuration initiale de Proxmox VE 9. Les points clés à retenir :

Prérequis matériels : Un processeur 64 bits avec VT-x/AMD-V activé, de la RAM ECC, des disques SSD NVMe pour le système, et plusieurs interfaces réseau séparées pour les différents flux (management, VM, stockage, cluster). Pour TechFlow SAS : 3 nœuds avec 256 Go ECC, NVMe en miroir ZFS, et interfaces 10/25 GbE.

Préparation : Téléchargez l'ISO exclusivement depuis `proxmox.com`, vérifiez l'intégrité SHA256, et configurez le BIOS/UEFI avec les extensions de virtualisation activées (VT-x/AMD-V, VT-d/AMD-Vi) et Secure Boot désactivé.

Installation : L'installateur graphique guide étape par étape. Choisissez ZFS mirror pour la résilience du système. Définissez un FQDN cohérent dès l'installation car il sera difficile à modifier après la création du cluster.

Post-installation : Configurez les dépôts APT (no-subscription ou enterprise), effectuez les mises à jour système, configurez le réseau avec bridges et bonds LACP, et synchronisez l'horloge avec chrony sur les serveurs NTP internes.

Sécurisation : Désactivez la connexion root directe par SSH, créez un utilisateur administrateur PAM avec clés SSH Ed25519, installez un certificat TLS signé par la PKI interne, activez le firewall Proxmox avec politique DROP par défaut, et déployez fail2ban pour la protection contre les attaques par force brute.

TechFlow SAS — pve1 : Nœud opérationnel sur 10.10.0.11 avec ZFS mirror NVMe, bonds réseau 10 GbE LACP pour les VM, interfaces Ceph 25 GbE en jumbo frames, interface Corosync dédiée, et toutes les mesures de sécurité appliquées. Le nœud est prêt à être intégré au cluster tri-nœuds au chapitre 3.

3 Formation et gestion du cluster

Ce chapitre couvre la mise en place d'un cluster Proxmox VE 9 haute disponibilité, de la compréhension des mécanismes internes jusqu'à l'exploitation quotidienne. Le fil rouge s'appuie sur l'infrastructure de **TechFlow SAS**, une PME disposant de trois serveurs physiques (`pve1` , `pve2` , `pve3`) reliés par un réseau de management et un réseau cluster dédié.

► ENVIRONNEMENT DE RÉFÉRENCE TECHFLOW SAS

Nœud	IP management	IP cluster (ring0)	IP cluster (ring1)
pve1	10.10.0.1	10.10.1.1	10.10.2.1
pve2	10.10.0.2	10.10.1.2	10.10.2.2
pve3	10.10.0.3	10.10.1.3	10.10.2.3

Chaque nœud dispose de deux interfaces dédiées au trafic cluster (`ens4` et `ens5`), distinctes de l'interface de management (`ens3`) et des interfaces VM (`ens6` , `ens7`).

3.1 Architecture d'un cluster Proxmox

Un cluster Proxmox VE est un ensemble de nœuds hyperviseurs qui partagent un état commun — inventaire des VMs et conteneurs, configuration réseau et stockage, politiques HA — et peuvent coordonner des opérations distribuées telles que la migration en direct ou le redémarrage automatique après panne. Cette cohérence repose sur une pile logicielle précise qu'il est indispensable de maîtriser avant toute intervention en production.

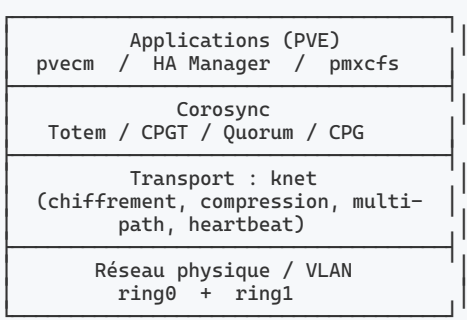
3.1.1 Corosync : le cœur du cluster

Corosync est le moteur de messagerie groupe (*group messaging*) qui sous-tend le cluster Proxmox. C'est lui qui :

- Diffuse les messages d'*heartbeat* entre nœuds pour détecter les défaillances ;
- Maintient la liste des membres actifs (*membership*) et notifie les services supérieurs lors de tout changement ;
- Garantit la cohérence des messages via un protocole de *totem single ring* (ou *knet* pour les topologies multi-liens).

Proxmox VE 9 embarque Corosync 3.x, lequel utilise **knet** (Kronosnet) comme couche transport par défaut. Knet offre nativement la compression et le chiffrement du trafic, ainsi que la gestion de plusieurs liens simultanés (*multi-path*), ce qui représente une avancée majeure par rapport au transport UDP historique.

La pile complète s'articule comme suit :



pmxcfs (Proxmox Cluster File System) est le système de fichiers distribué de Proxmox, monté sous `/etc/pve` . Il repose sur **libqb** et **corosync CPG** pour répliquer en temps réel les fichiers de

configuration sur tous les nœuds. Toute écriture dans `/etc/pve` transite par ce mécanisme ; en cas de perte de quorum, le système de fichiers passe en lecture seule pour protéger la cohérence.

> • **PMXCFS VS STOCKAGE PARTAGÉ**
> `/etc/pve` n'est **pas** un stockage partagé pour les disques de VM. C'est uniquement un registre de configuration (fichiers `.conf`, certificats, tokens API…). Les images de disques doivent résider sur un stockage dédié : Ceph, NFS, iSCSI, etc.

3.1.2 pvecm et l'API cluster

`pvecm` est l'outil en ligne de commande fourni par Proxmox pour administrer le cluster. Il constitue la façade la plus directe pour les opérations de gestion et s'appuie en interne sur l'API REST de Proxmox ainsi que sur les fichiers de configuration Corosync.

Principales sous-commandes :

Sous-commandes pvecm essentielles

Commande	Description
`pvecm create <nom>`	Initialise un nouveau cluster sur le nœud local
`pvecm add <ip>`	Rejoint un cluster existant depuis un nouveau nœud
`pvecm status`	Affiche l'état général du cluster (membres, quorum, votes)
`pvecm nodes`	Liste les nœuds avec leur ID et leur état
`pvecm delnode <nom>`	Retire un nœud du cluster (doit être hors ligne)
`pvecm qdevice setup <ip>`	Configure un QDevice externe
`pvecm qdevice remove`	Supprime la configuration QDevice
`pvecm updatecerts`	Renouvelle les certificats cluster

L'API REST expose les mêmes fonctionnalités sous `/api2/json/cluster/`. Par exemple :

```
# Récupérer le statut du cluster via l'API REST
curl -s -k -H "Authorization: PVEAPIToken=root@pam!mytoken=<uuid>" \
  https://pve1.techflow.lan:8006/api2/json/cluster/status | python3 -m json.tool
```

3.1.3 Quorum et votes

Le **quorum** est le mécanisme qui permet au cluster de distinguer une partition réseau d'une défaillance matérielle. Sans quorum, le cluster ne peut plus écrire dans pmxcfs ni prendre de décisions HA.

Chaque nœud possède un nombre de **votes** (1 par défaut). Le quorum est atteint lorsque le nombre de votes des nœuds visibles est strictement supérieur à la moitié du total des votes connus :

```
votes_visibles > votes_total / 2
```

Pour un cluster de **3 nœuds** (3 votes au total) :

Scénarios de quorum — cluster 3 nœuds

Nœuds actifs	Votes visibles	Quorum (> 1.5)	Conséquence
3/3	3	Oui	Fonctionnement normal
2/3	2	Oui	Dégradé mais opérationnel
1/3	1	Non	Cluster bloqué, pmxcfs en lecture seule

Pour un cluster de **2 nœuds**, le quorum n'est jamais atteint après la perte d'un nœud, ce qui explique pourquoi Proxmox recommande au minimum 3 nœuds ou l'utilisation d'un **QDevice**.

⚠ LE PARAMÈTRE EXPECTED_VOTES

En cas d'urgence, il est possible d'abaisser manuellement le quorum attendu avec :

```
corosync-quorumtool -e <valeur>
```

Cette opération est **dangereuse** en production : elle peut conduire à un *split-brain* si deux partitions du cluster tournent simultanément. Ne l'utiliser que lors d'une maintenance contrôlée avec un seul nœud actif connu.

3.1.4 Réseau cluster dédié

Le réseau cluster transporte des données critiques et sensibles au temps de latence : heartbeats Corosync, réplication pmxcfs, synchronisation HA. Il est **impératif** de l'isoler du trafic VM et du trafic de management.

Recommandations réseau pour la production :

- Utiliser des interfaces physiques dédiées (ou au minimum des VLAN distincts sur des commutateurs séparés pour ring0 et ring1) ;
- Viser une latence inférieure à **2 ms** entre nœuds ; au-delà, les timers Corosync par défaut peuvent déclencher des faux positifs ;
- Utiliser du **10 GbE** ou plus pour les clusters avec stockage Ceph intégré ;
- Ne **jamais** router le trafic cluster à travers un pare-feu statefull (la connexion Corosync utilise des ports UDP/TCP variables selon la configuration knet) ;
- Prévoir un **ring de secours** (ring1) sur un chemin physique distinct pour la redondance.

✔ PORTS RÉSEAU UTILISÉS PAR COROSYNC/KNET

Par défaut, knet utilise le port **UDP 5405** pour les heartbeats principaux, et les ports **UDP 5406-5410** pour les liens supplémentaires. pmxcfs utilise **TCP 5404**. Assurez-vous que ces ports sont ouverts entre tous les nœuds du cluster (et uniquement entre eux).

3.2 Prérequis avant la formation du cluster

La formation d'un cluster échoue ou génère des problèmes opérationnels persistants lorsque les prérequis de base ne sont pas respectés. Cette section couvre les vérifications à effectuer **avant** d'exécuter `pvecm create`.

3.2.1 Synchronisation NTP obligatoire

Corosync et pmxcfs sont très sensibles aux dérives d'horloge entre nœuds. Une différence supérieure à **1 seconde** peut provoquer des erreurs de certificat, des incohérences dans les logs et, dans les cas extrêmes, des problèmes de quorum.

Proxmox VE 9 configure **chrony** par défaut. Vérifier l'état sur chaque nœud :

```
# Vérifier la synchronisation chrony
chronyc tracking

# Résultat attendu — extrait
Reference ID    : 0A0A0065 (ntp.techflow.lan)
Stratum         : 3
Ref time (UTC)  : Mon Mar 17 08:30:12 2026
System time     : 0.000012345 seconds fast of NTP time
Last offset     : +0.000008234 seconds
RMS offset      : 0.000015432 seconds
Frequency       : 12.345 ppm fast
Residual freq   : +0.001 ppm
Skew            : 0.123 ppm
Root delay      : 0.012345678 seconds
Root dispersion : 0.000234567 seconds
Update interval : 64.2 seconds
Leap status     : Normal
```

Si chrony n'est pas synchronisé :

```
# Forcer une resynchronisation immédiate
chronyc makestep
```

```
# Vérifier les sources NTP disponibles
chronyc sources -v

# Si aucune source externe, configurer le serveur NTP interne
cat >> /etc/chrony/chrony.conf << 'EOF'
server ntp.techflow.lan iburst prefer
EOF

systemctl restart chrony
chronyc tracking
```

✔ NTP INTERNE RECOMMANDÉ

En environnement isolé ou avec des règles de sécurité strictes, déployez un serveur NTP interne (sur le pare-feu ou un serveur dédié). Configurez tous les nœuds Proxmox pour pointer vers ce serveur unique. La cohérence est plus importante que la précision absolue.

Vérification rapide de la dérive entre nœuds :

```
# Sur pve1 — comparer les horloges des trois nœuds
for node in pve1 pve2 pve3; do
  echo -n "$node: "
  ssh root@$node "date +%s.%N"
done
```

3.2.2 Résolution DNS/hosts

Proxmox résout les noms d'hôtes des nœuds pour les communications internes. Une mauvaise résolution est l'une des causes les plus fréquentes d'échec lors de la jonction d'un nœud.

Chaque nœud doit être capable de résoudre les autres nœuds par leur **nom court** (hostname) et leur **FQDN** :

```
# Sur chaque nœud — vérifier la résolution
hostname --fqdn    # Doit retourner le FQDN complet
hostname -s        # Doit retourner le nom court

# Test de résolution croisée depuis pve1
for node in pve1 pve2 pve3; do
  echo -n "Résolution $node: "
  getent hosts $node
done
```

Configuration `/etc/hosts` recommandée pour TechFlow SAS — à appliquer sur **chaque nœud** :

```
127.0.0.1       localhost
127.0.1.1       pve1.techflow.lan pve1    # adapter selon le nœud

# Nœuds cluster — réseau management
10.10.0.1       pve1.techflow.lan pve1
10.10.0.2       pve2.techflow.lan pve2
10.10.0.3       pve3.techflow.lan pve3
```

⚠ NE PAS UTILISER LES IPS CLUSTER DANS /ETC/HOSTS POUR LES NOMS D'HÔTES

Il peut être tentant d'utiliser les IPs du réseau cluster (10.10.1.x) dans `/etc/hosts` pour les noms d'hôtes des nœuds. **Ne le faites pas.** Proxmox utilise les noms d'hôtes pour les certificats TLS et les communications API qui doivent passer par le réseau de management. Les IPs cluster sont configurées séparément dans `corosync.conf`.

Vérification que le hostname correspond bien à l'IP de management :

```
# Sur pve1
ping -c 3 $(hostname -s)   # Doit atteindre 10.10.0.1
ping -c 3 $(hostname -f)   # Idem via FQDN
```

3.2.3 Plan réseau cluster (ring0, ring1)

Avant de créer le cluster, identifier et tester la connectivité des interfaces dédiées au trafic Corosync.

Inventaire des interfaces sur TechFlow SAS :

```
# Sur chaque nœud — lister les interfaces
ip -br link show

# Résultat typique
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
ens3             UP             52:54:00:aa:bb:01
<BROADCAST,MULTICAST,UP,LOWER_UP>
ens4             UP             52:54:00:aa:bb:02
<BROADCAST,MULTICAST,UP,LOWER_UP>
ens5             UP             52:54:00:aa:bb:03
<BROADCAST,MULTICAST,UP,LOWER_UP>
ens6             UP             52:54:00:aa:bb:04
<BROADCAST,MULTICAST,UP,LOWER_UP>
```

Configuration des interfaces cluster sur **pve1** :

```
# Ring0 — réseau cluster principal
auto ens4
iface ens4 inet static
    address 10.10.1.1/24
    # Pas de gateway — réseau cluster isolé

# Ring1 — réseau cluster redondant
auto ens5
iface ens5 inet static
    address 10.10.2.1/24
    # Pas de gateway — réseau cluster isolé
```

Appliquer et tester la connectivité avant de créer le cluster :

```
# Appliquer la configuration réseau
ifup ens4 ens5

# Test de latence ring0 depuis pve1 vers pve2 et pve3
ping -c 10 -i 0.2 10.10.1.2
ping -c 10 -i 0.2 10.10.1.3

# Test ring1
ping -c 10 -i 0.2 10.10.2.2
ping -c 10 -i 0.2 10.10.2.3

# Vérifier que la latence est < 2ms (valeur max_poll par défaut de Corosync)
ping -c 100 -i 0.01 10.10.1.2 | tail -1
# 100 packets transmitted, 100 received, 0% packet loss, time 999ms
# rtt min/avg/max/mdev = 0.123/0.187/0.312/0.034 ms
```

✔ MTU ET JUMBO FRAMES

Pour les clusters avec Ceph intégré, configurer le MTU à 9000 (jumbo frames) sur les interfaces cluster améliore significativement le débit de réplication. Assurez-vous que tous les commutateurs intermédiaires supportent cette MTU avant activation.

```
ip link set dev ens4 mtu 9000
ip link set dev ens5 mtu 9000
```

Ajouter `mtu 9000` dans `/etc/network/interfaces` pour la persistance.

3.3 Création du cluster

Une fois les prérequis validés sur les trois nœuds, la création du cluster se déroule en deux phases : initialisation sur le premier nœud, puis jonction des nœuds supplémentaires.

3.3.1 Création sur le premier nœud

L'initialisation du cluster se fait **uniquement sur** `pve1`. Cette opération crée les fichiers de configuration Corosync, initialise pmxcfs, et génère les certificats cluster.

✗ POINT DE NON-RETOUR

La commande `pvecm create` réinitialise complètement l'état cluster du nœud. Si `pve1` était déjà membre d'un cluster existant, toutes les VMs et conteneurs configurés sur ce cluster seront orphelins. Assurez-vous d'opérer sur un nœud vierge ou d'avoir sauvegardé toutes les configurations nécessaires.

Création du cluster `techflow-cluster` avec lien ringo dédié :

```
# Sur pve1 uniquement
pvecm create techflow-cluster \
  --link0 10.10.1.1 \
  --link1 10.10.2.1

# Sortie attendue
Corosync Authkey: /etc/corosync/authkey
Corosync Config: /etc/corosync/corosync.conf
restarting corosync.service ...
```

Vérifier immédiatement l'état du cluster à un nœud :

```
pvecm status

# Sortie
Cluster information
-------------------
Name:             techflow-cluster
Config Version:   1
Transport:        knet
Secure auth:      on

Quorum information
------------------
Date:             Mon Mar 17 08:45:12 2026
Quorum provider:  corosync_votequorum
Nodes:            1
Node state:       Quorate
Quorate:          Yes

Membership information
----------------------
    Nodeid      Votes Name
         1          1 pve1 (local)
```

Inspecter le fichier `corosync.conf` généré :

```
cat /etc/corosync/corosync.conf
```

```
totem {
  version: 2
  cluster_name: techflow-cluster
  config_version: 1
  ip_version: ipv4-6
  link_mode: passive
  interface {
    linknumber: 0
    knet_transport: udp
  }
  interface {
    linknumber: 1
    knet_transport: udp
  }
}

nodelist {
  node {
    name: pve1
    nodeid: 1
    ring0_addr: 10.10.1.1
    ring1_addr: 10.10.2.1
```

```
  }
}

quorum {
  provider: corosync_votequorum
}

logging {
  to_logfile: yes
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes
  timestamp: on
}
```

3.3.2 Jonction des nœuds supplémentaires

La jonction s'effectue **depuis chaque nœud à ajouter** (`pve2` , puis `pve3`), en pointant vers l'IP de management de `pve1` . La commande va :

1. Se connecter à `pve1` via SSH pour récupérer la configuration Corosync et l'authkey ;
2. Configurer le nœud local avec les bons paramètres réseau ;
3. Redémarrer les services cluster ;
4. Mettre à jour `corosync.conf` sur tous les membres existants.

Jonction de pve2 :

```
# Sur pve2 — jonction en spécifiant les IPs des deux rings
pvecm add 10.10.0.1 \
  --link0 10.10.1.2 \
  --link1 10.10.2.2

# Sortie attendue
Please enter superuser (root) password for '10.10.0.1': ********
Establishing API connection with host '10.10.0.1'
The authenticity of host '10.10.0.1' 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: OK
stopping pve-cluster.service ...
backup old database ...
waiting for quorum ...
OK
```

Jonction de pve3 :

```
# Sur pve3
pvecm add 10.10.0.1 \
  --link0 10.10.1.3 \
  --link1 10.10.2.3
```

> ⚠ **ERREUR FRÉQUENTE — SERVICES ACTIFS LORS DE LA JONCTION**
> Si le nœud à joindre héberge déjà des VMs en cours d'exécution, la jonction échouera ou provoquera des incohérences. Arrêtez toutes les VMs et conteneurs avant d'exécuter `pvecm add` . Vérifiez avec :
>
> ```
> qm list        # VMs KVM
> pct list       # Conteneurs LXC
> ```

En cas d'erreur SSH lors de la jonction (clé d'hôte inconnue), pré-autoriser manuellement :

```
# Sur pve2/pve3 — ajouter la clé SSH de pve1
ssh-keyscan -H 10.10.0.1 >> ~/.ssh/known_hosts

# Puis relancer pvecm add
pvecm add 10.10.0.1 --link0 10.10.1.2 --link1 10.10.2.2
```

3.3.3 Vérification de l'état du cluster

Après jonction des trois nœuds, vérifier l'état depuis n'importe quel membre :

```
# État global du cluster
pvecm status

# Sortie complète
Cluster information
-------------------
Name:             techflow-cluster
Config Version:   3
Transport:        knet
Secure auth:      on

Quorum information
------------------
Date:             Mon Mar 17 09:05:47 2026
Quorum provider:  corosync_votequorum
Nodes:            3
Node state:       Quorate
Quorate:          Yes

Membership information
----------------------
    Nodeid      Votes Name
         1          1 pve1 (local)
         2          1 pve2
         3          1 pve3
```

```
# Liste des nœuds
pvecm nodes

# Sortie
Nodes in cluster: 3

   ID   IP               Name           Quorum
    1   10.10.1.1        pve1              1
    2   10.10.1.2        pve2              1
    3   10.10.1.3        pve3              1
```

```
# État des liens knet — outil natif Corosync
corosync-cfgtool -s

# Sortie
Local node ID 1, transport knet
LINK ID 0 udp
        addr    = 10.10.1.1
        status:
                enabled:    1
                connected:  1
                dynconnected:   1
                mtu:        1452
                ping_interval:  1000 ms
                pong_count:     1
                knet_link_priority: 1
                knet_ping_precision:    2000
                knet_pong_count:        2
                knet_link_status:       Pong received
LINK ID 1 udp
        addr    = 10.10.2.1
        status:
                enabled:    1
                connected:  1
                ...
```

```
# Vérification pmxcfs
pvecm status | grep -A5 "Membership"

# État du service pmxcfs
systemctl status pve-cluster
```

```
# Vérifier que /etc/pve est monté
mount | grep pve

# Sortie attendue
tmpfs on /etc/pve type tmpfs (rw,relatime)
```

Tests de connectivité supplémentaires :

```
# Tester la réplication pmxcfs — créer un fichier depuis pve1
touch /etc/pve/test-cluster-replication.txt

# Vérifier qu'il est visible depuis pve2 et pve3 (presque instantané)
ssh root@pve2 "ls -la /etc/pve/test-cluster-replication.txt"
ssh root@pve3 "ls -la /etc/pve/test-cluster-replication.txt"

# Nettoyage
rm /etc/pve/test-cluster-replication.txt
```

3.3.4 Gestion du quorum avec QDevice

Le **QDevice** est un arbitre externe qui fournit un vote supplémentaire sans être un nœud Proxmox à part entière. Il est particulièrement utile pour :

- Les clusters à **2 nœuds** (où il empêche le blocage mutuel) ;
- Les clusters à **nombre pair de nœuds** (pour éviter les partitions 50/50) ;
- Les sites géographiquement distribués (QDevice dans une troisième zone).

Un QDevice nécessite une machine distincte (peut être légère : VM, Raspberry Pi, serveur applicatif existant) avec `corosync-qnetd` installé.

Installation du daemon QDevice sur le serveur dédié (qdevice.techflow.lan — 10.10.0.10) :

```
# Sur le serveur QDevice (Debian/Ubuntu)
apt-get install -y corosync-qnetd

# Démarrer et activer le service
systemctl enable --now corosync-qnetd

# Vérifier l'état
systemctl status corosync-qnetd
```

Configuration du QDevice sur le cluster Proxmox (depuis pve1) :

```
# Installer le client QDevice sur tous les nœuds Proxmox
apt-get install -y corosync-qdevice

# Configurer le QDevice depuis pve1
pvecm qdevice setup 10.10.0.10

# Sortie attendue
Generating QDevice cert on node '10.10.0.10' ...
Configuring QDevice on cluster nodes ...
Done.

# Vérifier le nouveau vote QDevice
pvecm status | grep -A10 "Membership"

# Membership information
# ──────────────────────
#     Nodeid      Votes    Qdevice Name
#          1          1    A,V,NMW pve1 (local)
#          2          1    A,V,NMW pve2
#          3          1    A,V,NMW pve3
#          0          1    Qdevice
```

Les lettres dans la colonne `Qdevice` signifient :

- **A** : Alive (connecté)
- **V** : Vote (le QDevice fournit un vote)
- **NMW** : Not a Master Winner (n'est pas éligible comme maître)

• QDEVICE ET CLUSTER À 2 NŒUDS

Pour un cluster à 2 nœuds sans QDevice, la perte d'un nœud entraîne **toujours** la perte de quorum (1 vote visible sur 2 = 50%, pas strictement supérieur à 50%). Avec un QDevice, le nœud survivant a 2 votes (1 nœud + 1 QDevice) sur 3 totaux = quorum maintenu. C'est la configuration minimale recommandée pour la production avec 2 nœuds.

3.4 Configuration avancée de Corosync

3.4.1 Fichier corosync.conf décortiqué

`corosync.conf` est le fichier de configuration central de Corosync. Sur Proxmox, il est géré par `pvecm` et répliqué automatiquement sur tous les nœuds via pmxcfs. Le modifier manuellement nécessite des précautions.

Voici un fichier `corosync.conf` complet annoté pour TechFlow SAS avec deux liens knet et chiffrement activé :

```
totem {
  # Version du protocole totem (toujours 2)
  version: 2

  # Nom du cluster — doit être unique dans le réseau
  cluster_name: techflow-cluster

  # Version incrémentée à chaque modification de la config
  config_version: 5

  # Préférence de version IP
  ip_version: ipv4-6

  # Mode de connexion knet entre nœuds
  # passive : connexion à la demande
  # active  : connexion permanente (recommandé pour la prod)
  link_mode: active

  # Intervalle heartbeat en ms (défaut : 1000ms)
  token: 5000

  # Nombre de tokens perdus avant déclaration de défaillance
  token_retransmits_before_loss_const: 10

  # Délai max pour consensus en ms (doit être > token)
  consensus: 6000

  # Intervalle entre retransmissions en ms
  retransmit_token: 750

  # Nombre maximum de messages dans une fenêtre de transmission
  max_messages: 20

  # Interface ring0 — lien principal
  interface {
    linknumber: 0
    knet_transport: udp
    knet_link_priority: 1        # Priorité haute = lien préféré
    knet_ping_interval: 1000     # Intervalle ping en ms
    knet_ping_timeout: 2000      # Timeout ping en ms
    knet_pong_count: 2           # Nombre de pongs requis
  }

  # Interface ring1 — lien de secours
  interface {
    linknumber: 1
    knet_transport: udp
    knet_link_priority: 2        # Priorité basse = lien de secours
    knet_ping_interval: 1000
```

```
    knet_ping_timeout: 2000
    knet_pong_count: 2
  }

  # Chiffrement du trafic cluster
  crypto_hash: sha256
  crypto_cipher: aes256
}

nodelist {
  node {
    name: pve1
    nodeid: 1
    ring0_addr: 10.10.1.1
    ring1_addr: 10.10.2.1
  }
  node {
    name: pve2
    nodeid: 2
    ring0_addr: 10.10.1.2
    ring1_addr: 10.10.2.2
  }
  node {
    name: pve3
    nodeid: 3
    ring0_addr: 10.10.1.3
    ring1_addr: 10.10.2.3
  }
}

quorum {
  provider: corosync_votequorum
  # Activer le two_node_mode si cluster à 2 nœuds sans QDevice
  # two_node: 1
  # Activer le wait_for_all pour éviter le split-brain au démarrage
  wait_for_all: 1
  # Délai d'attente en secondes avant de déclarer le quorum perdu
  last_man_standing: 0
}

logging {
  to_logfile: yes
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes
  debug: off
  timestamp: on
  logger_subsys {
    subsys: QUORUM
    debug: off
  }
  logger_subsys {
    subsys: KNET
    debug: off
  }
}
```

✗ MODIFICATION MANUELLE DE COROSYNC.CONF

Ne modifiez jamais `corosync.conf` directement en production sans passer par `pvecm` ou sans suivre la procédure ci-dessous. Une erreur de syntaxe empêche Corosync de démarrer sur tous les nœuds simultanément.

Procédure sûre pour une modification manuelle : 1. Sauvegarder : `cp /etc/corosync/corosync.conf /etc/corosync/corosync.conf.bak` 2. Modifier sur un seul nœud et valider la syntaxe 3. Incrémenter `config_version` d'une unité 4. Recharger : `corosync-cfgtool -R` 5. Vérifier que la config se propage : `corosync-cmapctl | grep config_version`

3.4.2 Tuning des timers (token, consensus)

Les timers Corosync définissent la rapidité avec laquelle le cluster détecte et réagit à la défaillance d'un nœud. Les valeurs par défaut conviennent à un réseau local fiable ; des environnements plus complexes peuvent nécessiter des ajustements.

Paramètres clés :

Timers Corosync — paramètres et recommandations

Paramètre	Défaut (ms)	Prod recommandé (ms)	Description
`token`	1000	5000–10000	Temps maximal d'attente du jeton avant de déclarer un nœud défaillant
`consensus`	1200	6000–12000	Délai pour atteindre le consensus après perte du jeton (doit être > token)
`retransmit_token`	200	750	Intervalle de retransmission du jeton en cas de perte
`join`	50	50	Délai d'attente pour les messages de jonction
`max_messages`	17	20	Nombre max de messages par jeton
`token_retransmits_before_loss_const`	4	10	Nombre de retransmissions avant déclaration de perte

Scénarios de tuning :

Réseau LAN fiable — valeurs agressives pour détection rapide :

```
# Détection de panne en ~1s — uniquement sur LAN très fiable
# Dans corosync.conf, section totem :
token: 1000
consensus: 1500
token_retransmits_before_loss_const: 4
```

Réseau avec latence variable ou WAN :

```
# Détection plus lente mais moins de faux positifs
token: 10000
consensus: 12000
token_retransmits_before_loss_const: 10
```

Cluster multi-site avec latence ~10ms :

```
# Recommandation Proxmox pour liens inter-datacenter
token: 30000
consensus: 36000
retransmit_token: 2000
token_retransmits_before_loss_const: 10
```

Appliquer un changement de timer sans redémarrage (recharge à chaud) :

```
# Modifier corosync.conf sur tous les nœuds, puis recharger
corosync-cfgtool -R

# Vérifier la prise en compte
corosync-cmapctl totem.token
corosync-cmapctl totem.consensus
```

3.4.3 Lien redondant (knet)

knet (Kronosnet) est la couche transport par défaut depuis Corosync 3. Il supporte nativement plusieurs liens simultanés avec basculement automatique et équilibrage de charge.

Ajout d'un lien ring1 sur un cluster existant :

Si le cluster a été créé sans ring1, il est possible d'en ajouter un a posteriori :

```
# Vérifier la configuration actuelle des liens
pvecm status | grep -i link
corosync-cfgtool -s

# Ajouter le lien ring1 sur pve1 via pvecm
pvecm updatelink pve1 --link1 10.10.2.1
pvecm updatelink pve2 --link1 10.10.2.2
pvecm updatelink pve3 --link1 10.10.2.3

# Recharger la configuration
corosync-cfgtool -R
```

État et diagnostics des liens knet :

```
# Afficher l'état de tous les liens sur le nœud local
corosync-cfgtool -s

# Afficher les statistiques de trafic knet
corosync-cfgtool -r

# Sortie type
Local node ID 1, transport knet
LINK ID 0 udp
        addr    = 10.10.1.1
        status:
                enabled:    1
                connected:  1
                ...
        statistics:
                tx_data_packets:    1254789
                rx_data_packets:    1248234
                tx_compress_packets: 0
                rx_compress_packets: 0

# État des nœuds distants sur le ring0
corosync-cmapctl stats.knet.node2.link0.tx_data_packets
corosync-cmapctl stats.knet.node3.link0.tx_data_packets
```

Simulation de basculement de lien :

```
# Sur pve1 — simuler une perte du ring0
ip link set dev ens4 down

# Vérifier que le cluster bascule sur ring1
watch -n1 corosync-cfgtool -s

# Dans les logs — basculement visible
tail -f /var/log/corosync/corosync.log | grep -i "link\|failover"

# Rétablir le ring0
ip link set dev ens4 up

# Vérifier le retour sur le lien principal
corosync-cfgtool -s
```

3.4.4 Chiffrement du trafic cluster

Par défaut depuis Proxmox VE 7, le trafic Corosync est chiffré avec AES-256 et authentifié via SHA-256. La clé symétrique est stockée dans `/etc/corosync/authkey`.

Vérification du chiffrement actuel :

```
# Vérifier les paramètres de chiffrement dans corosync.conf
grep -E "crypto_|auth" /etc/corosync/corosync.conf

# Vérifier via cmapctl
corosync-cmapctl totem.crypto_cipher
corosync-cmapctl totem.crypto_hash
```

Régénération de la clé d'authentification :

Il peut être nécessaire de régénérer la clé lors d'un changement de personnel ou après un incident de sécurité. Cette opération nécessite une fenêtre de maintenance.

```
# Procédure de rotation de la clé authkey
# 1. Mettre le cluster en maintenance

# 2. Sur pve1 — générer une nouvelle clé
corosync-keygen -k /etc/corosync/authkey.new

# 3. Distribuer la nouvelle clé sur tous les nœuds
for node in pve2 pve3; do
  scp /etc/corosync/authkey.new root@${node}:/etc/corosync/authkey.new
done

# 4. Sur chaque nœud — arrêter Corosync, remplacer la clé, redémarrer
for node in pve3 pve2 pve1; do
  ssh root@${node} "
    systemctl stop corosync
    cp /etc/corosync/authkey /etc/corosync/authkey.old
    mv /etc/corosync/authkey.new /etc/corosync/authkey
    chmod 400 /etc/corosync/authkey
    systemctl start corosync
    sleep 5
    pvecm status | grep -i quorate
  "
done
```

✔ **CHIFFREMENT ET PERFORMANCES**

Sur du matériel récent disposant d'accélération AES-NI (quasi-universel depuis 2013), le chiffrement AES-256 a un impact négligeable sur les performances du cluster. Ne désactivez jamais le chiffrement pour des raisons de performances — les gains seraient imperceptibles et le risque sécuritaire injustifié.

3.5 Gestion du cluster au quotidien

3.5.1 Surveillance de l'état

La supervision du cluster doit être intégrée dans les outils de monitoring de l'entreprise. Voici les métriques et commandes essentielles pour un opérateur.

Tableau de bord en ligne de commande :

```
# Vue d'ensemble instantanée — à utiliser en premier lors d'un incident
pvecm status

# État des nœuds et ressources
pvecm nodes

# État du quorum
corosync-quorumtool -l

# Sortie corosync-quorumtool
Quorum information
------------------
Date:             Mon Mar 17 10:00:00 2026
Quorum provider:  corosync_votequorum
Nodes:            3
Node state:       Quorate
Quorate:          Yes

Membership information
----------------------
    Nodeid      Votes    Name
         1          1    pve1 (local)
         2          1    pve2
         3          1    pve3
```

```
# Vérifier l'état de tous les services cluster
for svc in corosync pve-cluster pvedaemon pveproxy; do
  echo -n "$svc: "
  systemctl is-active $svc
done

# Santé globale du nœud local
pveversion -v

# Sortie
proxmox-ve: 9.0-1 (running kernel: 6.8.12-1-pve)
pve-manager: 9.0.5 (running version: 9.0.5/xxxxxxxx)
pve-kernel-6.8: 6.8.12-1
corosync: 3.1.9-pve1
...
```

Script de supervision rapide pour TechFlow SAS :

```
#!/bin/bash
# /usr/local/bin/cluster-health-check.sh

ERRORS=0

echo "═══ Vérification cluster TechFlow SAS ═══"
echo "Date: $(date)"
echo ""

# 1. Quorum
echo -n "[Quorum] "
QUORATE=$(corosync-quorumtool -p 2>/dev/null | grep Quorate | awk '{print $2}')
if [ "$QUORATE" = "Yes" ]; then
  echo "OK (Quorate)"
else
  echo "ERREUR (Pas de quorum!)"
  ERRORS=$((ERRORS + 1))
fi

# 2. Nombre de nœuds
echo -n "[Noeuds] "
NODES=$(pvecm nodes 2>/dev/null | grep -c "^   [0-9]")
if [ "$NODES" -eq 3 ]; then
  echo "OK ($NODES/3 noeuds actifs)"
else
  echo "ATTENTION ($NODES/3 noeuds actifs)"
  ERRORS=$((ERRORS + 1))
fi

# 3. Services critiques
for svc in corosync pve-cluster pvedaemon; do
  echo -n "[$svc] "
  if systemctl is-active --quiet $svc; then
    echo "OK (running)"
  else
    echo "ERREUR (not running!)"
    ERRORS=$((ERRORS + 1))
  fi
done

# 4. Liens knet
echo -n "[Ring0] "
RING0_STATUS=$(corosync-cfgtool -s 2>/dev/null | grep -A5 "LINK ID 0" | \
               grep "connected:" | awk '{print $2}')
if [ "$RING0_STATUS" = "1" ]; then
  echo "OK (connecte)"
else
  echo "ATTENTION (ring0 deconnecte)"
fi
```

```
echo -n "[Ring1] "
RING1_STATUS=$(corosync-cfgtool -s 2>/dev/null | grep -A5 "LINK ID 1" | \
               grep "connected:" | awk '{print $2}')
if [ "$RING1_STATUS" = "1" ]; then
  echo "OK (connecte)"
else
  echo "ATTENTION (ring1 deconnecte)"
fi

echo ""
echo "═══ Resultat : $ERRORS erreur(s) critique(s) ═══"
exit $ERRORS
```

Intégration Zabbix / Nagios :

```
#!/bin/bash
# /usr/lib/nagios/plugins/check_pve_cluster

QUORATE=$(corosync-quorumtool -p 2>/dev/null | grep "^Quorate" | awk '{print $2}')
NODES_ACTIVE=$(pvecm nodes 2>/dev/null | grep -c "^   [0-9]")
NODES_EXPECTED=3

if [ "$QUORATE" ≠ "Yes" ]; then
  echo "CRITICAL: Cluster not quorate"
  exit 2
elif [ "$NODES_ACTIVE" -lt "$NODES_EXPECTED" ]; then
  echo "WARNING: Only $NODES_ACTIVE/$NODES_EXPECTED nodes active"
  exit 1
else
  echo "OK: Cluster quorate, $NODES_ACTIVE/$NODES_EXPECTED nodes active"
  exit 0
fi
```

3.5.2 Ajout et retrait d'un nœud

Ajout d'un nouveau nœud (pve4) au cluster existant :

```
# Prérequis sur pve4 — vérifier avant jonction
hostnamectl set-hostname pve4.techflow.lan

# Ajouter pve4 dans /etc/hosts de tous les nœuds existants
echo "10.10.0.4  pve4.techflow.lan pve4" >> /etc/hosts

# Configurer les interfaces cluster sur pve4

# Jonction depuis pve4
pvecm add 10.10.0.1 \
  --link0 10.10.1.4 \
  --link1 10.10.2.4

# Vérifier la jonction depuis pve1
pvecm status
pvecm nodes
```

Retrait propre d'un nœud (procédure maintenance) :

Le retrait d'un nœud nécessite une procédure en plusieurs étapes pour éviter les incohérences de configuration et les problèmes de quorum.

```
# Etape 1 — Migrer ou arrêter toutes les VMs/conteneurs du nœud à retirer
qm list        # Identifier les VMs sur pve3
pct list       # Identifier les conteneurs sur pve3

# Migration live des VMs (si stockage partagé disponible)
qm migrate <vmid> pve2 --online

# Etape 2 — Désactiver HA pour ce nœud si activé
ha-manager set node:pve3 --state disabled

# Etape 3 — Arrêter les services cluster sur pve3
```

```
ssh root@pve3 "systemctl stop pve-ha-lrm pve-ha-crm pvedaemon pveproxy"
ssh root@pve3 "systemctl stop corosync"

# Etape 4 – Retirer le nœud depuis pve1 (nœud pve3 doit être HORS LIGNE)
pvecm delnode pve3

# Vérifier la suppression
pvecm nodes
```

> ⚠ **RETRAIT D'UN NŒUD DÉFAILLANT**
> Si le nœud à retirer est inaccessible (panne matérielle), le cluster peut refuser `pvecm delnode` s'il perd le quorum. Dans ce cas, il faut d'abord restaurer le quorum :
>
> ```
> corosync-quorumtool -e 2   # Réduire le quorum attendu à 2 votes
> ```
>
> Puis retirer le nœud :
>
> ```
> pvecm delnode pve3
> ```
>
> Et enfin remettre le quorum à sa valeur normale (automatique après le redémarrage de Corosync, ou manuellement).

Réintégration d'un nœud après maintenance matérielle :

```
# Sur le nœud réintégré – s'assurer qu'il n'y a plus de config cluster résiduelle
systemctl stop pve-cluster corosync

# Supprimer l'ancienne configuration cluster
rm -f /etc/corosync/corosync.conf
rm -f /etc/corosync/authkey

# Réinitialiser pmxcfs
systemctl stop pve-cluster
rm -rf /var/lib/pve-cluster/*
systemctl start pve-cluster

# Rejoindre le cluster
pvecm add 10.10.0.1 --link0 10.10.1.3 --link1 10.10.2.3
```

3.5.3 Résolution des problèmes de quorum

La perte de quorum est l'une des situations les plus critiques dans un cluster Proxmox. Elle entraîne le passage de pmxcfs en lecture seule et bloque toutes les opérations d'écriture de configuration.

Diagnostic rapide :

```
# Identifier pourquoi le quorum est perdu
corosync-quorumtool -v

# Vérifier les nœuds visibles
corosync-quorumtool -l

# Vérifier les logs Corosync pour identifier la cause
journalctl -u corosync -n 50 --no-pager
tail -100 /var/log/corosync/corosync.log
```

Scénario 1 — Un nœud est hors ligne (maintenance planifiée) :

```
# Cluster à 3 nœuds – 2 nœuds actifs = quorum maintenu
pvecm status | grep Quorate
# Quorate: Yes   → OK, continuer

# Si non quorate avec 2 nœuds (cas d'un cluster à 2 nœuds sans QDevice)
# Abaisser temporairement le quorum attendu
corosync-quorumtool -e 1

# ATTENTION : uniquement si le 2ème nœud est physiquement hors ligne
# et ne peut pas démarrer de son côté
```

Scénario 2 — Split-brain (partition réseau, 2 partitions de taille égale) :

✗ SPLIT-BRAIN — SITUATION LA PLUS DANGEREUSE

Un split-brain se produit quand deux partitions du cluster croient chacune avoir le quorum et continuent à opérer indépendamment. Cela peut provoquer :

- Des corruptions de données sur les stockages partagés
- Des conflits d'état sur les VMs HA
- Des incohérences irréparables dans pmxcfs

Ne jamais abaisser le quorum sur DEUX partitions simultanément.

En cas de doute, arrêtez physiquement l'une des partitions et travaillez depuis l'autre.

Résolution d'un split-brain détecté :

```
# 1. Identifier quelle partition est "correcte"
# Comparer les config_version sur les nœuds accessibles
corosync-cmapctl totem.config_version

# 2. Choisir la partition "maître" et arrêter les nœuds de l'autre partition
systemctl stop pve-cluster corosync

# 3. Sur la partition survivante, vérifier et abaisser le quorum si nécessaire
corosync-quorumtool -e <nombre_noeuds_survivants>

# 4. Remettre les nœuds arrêtés en rejoignant le cluster proprement
```

Scénario 3 — Corosync ne démarre plus sur un nœud :

```
# Vérifier les erreurs au démarrage
journalctl -u corosync -b --no-pager | grep -i "error\|fatal\|critical"

# Erreur fréquente : config_version incohérente
cat /etc/corosync/corosync.conf | grep config_version

# Récupérer la config depuis un nœud sain
scp root@pve1:/etc/corosync/corosync.conf /etc/corosync/corosync.conf
scp root@pve1:/etc/corosync/authkey /etc/corosync/authkey
chmod 400 /etc/corosync/authkey

# Redémarrer
systemctl start corosync
systemctl status corosync
```

Scénario 4 — pmxcfs en lecture seule :

```
# Symptôme : impossibilité de créer/modifier des VMs depuis l'UI
# Erreur : "filesystem is read-only"

# Vérifier l'état du quorum
pvecm status

# Si le quorum est atteint mais pmxcfs reste en lecture seule
systemctl status pve-cluster

# Redémarrer le service (si quorum OK)
systemctl restart pve-cluster

# Vérifier le montage
mount | grep pve
ls -la /etc/pve/
```

3.5.4 Logs cluster

La compréhension des logs est indispensable pour diagnostiquer les incidents cluster. Plusieurs sources de logs coexistent.

Sources de logs :

Sources de logs cluster Proxmox

Fichier / Service	Commande d'accès	Contenu

`/var/log/corosync/corosync.log`	`tail -f /var/log/corosync/corosync.log`	Événements Corosync : membership, token, heartbeat, chiffrement
Journal systemd corosync	`journalctl -u corosync -f`	Démarrage/arrêt, erreurs critiques
Journal systemd pve-cluster	`journalctl -u pve-cluster -f`	Événements pmxcfs, réplication de config
`/var/log/pve/tasks/`	`ls /var/log/pve/tasks/`	Tâches Proxmox (migrations, sauvegardes, etc.)
Syslog	`journalctl -f | grep -i cluster`	Agrégation de tous les événements

Commandes de diagnostic avancées :

```
# Suivre en temps réel tous les événements cluster
journalctl -u corosync -u pve-cluster -u pve-ha-lrm -u pve-ha-crm -f

# Rechercher les événements de membership (nœud join/leave)
grep -E "TOTEM|QUIT|JOIN|LEAVE|membership" /var/log/corosync/corosync.log | tail
-50

# Rechercher les problèmes de token (réseau lent ou surchargé)
grep -i "retransmit\|token\|lost" /var/log/corosync/corosync.log | tail -50

# Extraire les événements des dernières 24h avec timestamp
journalctl -u corosync --since "24 hours ago" --no-pager \
  | grep -E "TOTEM|error|Error"

# Activer le debug temporairement pour un diagnostic approfondi
corosync-cmapctl -s logging.debug s true
# Désactiver après diagnostic
corosync-cmapctl -s logging.debug s false
```

Analyse d'un événement de perte de nœud :

```
# Extraire la séquence d'événements lors d'une perte de nœud
grep -A 20 "processor:.*left" /var/log/corosync/corosync.log | head -50

# Exemple de sortie typique lors de la perte de pve3 :
# Mar 17 14:23:01 corosync[1234]:  [TOTEM ] A processor failed, forming new
configuration.
# Mar 17 14:23:06 corosync[1234]:  [QUORUM] Members[2]: 1 2
# Mar 17 14:23:06 corosync[1234]:  [QUORUM] Quorate

# Calculer le temps de détection de panne (token timeout)
grep -E "processor.*failed|A processor" /var/log/corosync/corosync.log | tail -5
```

3.6 Application au lab TechFlow SAS

Cette section rassemble et ordonne toutes les étapes de mise en œuvre du cluster `techflow-cluster` pour l'infrastructure de TechFlow SAS, en suivant les meilleures pratiques décrites tout au long du chapitre.

3.6.1 Formation du cluster techflow-cluster

État initial : trois nœuds Proxmox VE 9 installés indépendamment, sans cluster.

Checklist pré-formation :

```
#!/bin/bash
# A exécuter sur CHAQUE nœud avant de créer le cluster
echo "═══ Validation pré-cluster — $(hostname) ═══"

# 1. Version Proxmox
echo -n "[Version PVE] "
pveversion | head -1

# 2. NTP
echo -n "[NTP Sync] "
```

```
chronyc tracking | grep "System time"

# 3. Hostname et FQDN
echo -n "[Hostname] "
hostname
echo -n "[FQDN] "
hostname --fqdn

# 4. Résolution DNS des pairs
echo "[Résolution DNS]"
for node in pve1 pve2 pve3; do
  echo -n "  $node: "
  getent hosts $node | awk '{print $1}'
done

# 5. Connectivité ring0
echo "[Connectivité ring0]"
for ip in 10.10.1.1 10.10.1.2 10.10.1.3; do
  echo -n "  $ip: "
  ping -c 1 -W 1 $ip > /dev/null 2>&1 && echo "OK" || echo "ECHEC"
done

# 6. Connectivité ring1
echo "[Connectivité ring1]"
for ip in 10.10.2.1 10.10.2.2 10.10.2.3; do
  echo -n "  $ip: "
  ping -c 1 -W 1 $ip > /dev/null 2>&1 && echo "OK" || echo "ECHEC"
done

# 7. Absence de cluster existant
echo -n "[Pas de cluster existant] "
if pvecm status 2>&1 | grep -q "not a member"; then
  echo "OK"
else
  echo "ATTENTION — nœud déjà dans un cluster"
fi

echo "═══ Fin de validation ═══"
```

Exécuter ce script sur les trois nœuds et corriger tous les problèmes avant de continuer :

```
# Depuis pve1 — exécuter la validation sur tous les nœuds
for node in pve1 pve2 pve3; do
  echo ""
  echo "####### $node #######"
  ssh root@${node} "bash /tmp/precheck.sh"
done
```

Création du cluster sur pve1 :

```
# Sur pve1 uniquement — création définitive du cluster
pvecm create techflow-cluster \
  --link0 address=10.10.1.1,priority=1 \
  --link1 address=10.10.2.1,priority=2

# Vérification immédiate
pvecm status
echo "---"
cat /etc/corosync/corosync.conf
```

Résultat attendu :

```
Cluster information
─────────────────
Name:             techflow-cluster
Config Version:   1
Transport:        knet
Secure auth:      on

Quorum information
```

```
Date:             Mon Mar 17 09:00:00 2026
Quorum provider:  corosync_votequorum
Nodes:            1
Node state:       Quorate
Quorate:          Yes

Membership information
----------------------
    Nodeid      Votes Name
         1          1 pve1 (local)
```

3.6.2 Jonction de pve2 et pve3

Jonction de pve2 :

```
# Sur pve2 — jonction au cluster techflow-cluster
pvecm add 10.10.0.1 \
  --link0 address=10.10.1.2,priority=1 \
  --link1 address=10.10.2.2,priority=2

# Attendre la fin de l'opération et vérifier
sleep 10
pvecm status
```

Vérification depuis pve1 après jonction de pve2 :

```
# Sur pve1
pvecm status

# Sortie attendue
Membership information
----------------------
    Nodeid      Votes Name
         1          1 pve1 (local)
         2          1 pve2
```

Jonction de pve3 :

```
# Sur pve3
pvecm add 10.10.0.1 \
  --link0 address=10.10.1.3,priority=1 \
  --link1 address=10.10.2.3,priority=2

sleep 10
pvecm status
```

Vérification du fichier corosync.conf final (depuis pve1) :

```
cat /etc/corosync/corosync.conf
```

```
totem {
  version: 2
  cluster_name: techflow-cluster
  config_version: 3
  ip_version: ipv4-6
  link_mode: passive
  interface {
    linknumber: 0
    knet_transport: udp
    knet_link_priority: 1
  }
  interface {
    linknumber: 1
    knet_transport: udp
    knet_link_priority: 2
  }
}

nodelist {
```

```
  node {
    name: pve1
    nodeid: 1
    ring0_addr: 10.10.1.1
    ring1_addr: 10.10.2.1
  }
  node {
    name: pve2
    nodeid: 2
    ring0_addr: 10.10.1.2
    ring1_addr: 10.10.2.2
  }
  node {
    name: pve3
    nodeid: 3
    ring0_addr: 10.10.1.3
    ring1_addr: 10.10.2.3
  }
}

quorum {
  provider: corosync_votequorum
}

logging {
  to_logfile: yes
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes
  timestamp: on
}
```

3.6.3 Vérification complète

Suite de tests de validation post-déploiement pour TechFlow SAS :

```
#!/bin/bash
# /usr/local/bin/validate-cluster-techflow.sh
# Validation complète du cluster techflow-cluster après déploiement

CLUSTER_NAME="techflow-cluster"
EXPECTED_NODES=3
ERRORS=0
WARNINGS=0

echo "═══════════════════════════════════════════════════"
echo "  Validation cluster TechFlow SAS — $(date)"
echo "═══════════════════════════════════════════════════"
echo ""

# ── Test 1 : Quorum ──────────────────────────────────
echo "[ TEST 1/8 ] Quorum"
QUORATE=$(pvecm status 2>/dev/null | grep "^Quorate" | awk '{print $2}')
if [ "$QUORATE" = "Yes" ]; then
  echo "  PASS : Cluster quorate"
else
  echo "  FAIL : Cluster NON quorate!"
  ERRORS=$((ERRORS+1))
fi

# ── Test 2 : Nom du cluster ──────────────────────────
echo "[ TEST 2/8 ] Nom du cluster"
NAME=$(pvecm status 2>/dev/null | grep "^Name:" | awk '{print $2}')
if [ "$NAME" = "$CLUSTER_NAME" ]; then
  echo "  PASS : Nom correct ($NAME)"
else
  echo "  FAIL : Nom inattendu ($NAME, attendu: $CLUSTER_NAME)"
  ERRORS=$((ERRORS+1))
fi
```

```
# ── Test 3 : Nombre de nœuds ──────────────────────────
echo "[ TEST 3/8 ] Noeuds actifs"
NODES=$(pvecm status 2>/dev/null | grep "^Nodes:" | awk '{print $2}')
if [ "$NODES" -eq "$EXPECTED_NODES" ]; then
  echo "  PASS : $NODES/$EXPECTED_NODES noeuds actifs"
else
  echo "  FAIL : $NODES/$EXPECTED_NODES noeuds actifs"
  ERRORS=$((ERRORS+1))
fi

# ── Test 4 : Nœuds et leurs IDs ───────────────────────
echo "[ TEST 4/8 ] Identité des nœuds"
for node in pve1 pve2 pve3; do
  if pvecm nodes 2>/dev/null | grep -q "$node"; then
    echo "  PASS : $node présent dans le cluster"
  else
    echo "  FAIL : $node absent du cluster!"
    ERRORS=$((ERRORS+1))
  fi
done

# ── Test 5 : Liens knet ───────────────────────────────
echo "[ TEST 5/8 ] Liens knet"
for link in 0 1; do
  STATUS=$(corosync-cfgtool -s 2>/dev/null | grep -A8 "LINK ID $link" | \
           grep "connected:" | head -1 | awk '{print $2}')
  if [ "$STATUS" = "1" ]; then
    echo "  PASS : Ring${link} connecté"
  else
    echo "  WARN : Ring${link} statut inconnu ou déconnecté"
    WARNINGS=$((WARNINGS+1))
  fi
done

# ── Test 6 : Transport et chiffrement ─────────────────
echo "[ TEST 6/8 ] Sécurité"
TRANSPORT=$(pvecm status 2>/dev/null | grep "^Transport:" | awk '{print $2}')
SECAUTH=$(pvecm status 2>/dev/null | grep "^Secure auth:" | awk '{print $3}')
if [ "$TRANSPORT" = "knet" ]; then
  echo "  PASS : Transport knet actif"
else
  echo "  WARN : Transport inattendu ($TRANSPORT)"
  WARNINGS=$((WARNINGS+1))
fi
if [ "$SECAUTH" = "on" ]; then
  echo "  PASS : Authentification sécurisée activée"
else
  echo "  WARN : Authentification sécurisée DÉSACTIVÉE"
  WARNINGS=$((WARNINGS+1))
fi

# ── Test 7 : pmxcfs ───────────────────────────────────
echo "[ TEST 7/8 ] pmxcfs"
if mount | grep -q "/etc/pve"; then
  echo "  PASS : pmxcfs monté sur /etc/pve"
else
  echo "  FAIL : pmxcfs NON monté!"
  ERRORS=$((ERRORS+1))
fi

TESTFILE="/etc/pve/.cluster-validation-$$"
if touch "$TESTFILE" 2>/dev/null; then
  echo "  PASS : pmxcfs accessible en écriture"
  sleep 2
  for node in pve2 pve3; do
    if ssh root@$node "test -f $TESTFILE" 2>/dev/null; then
      echo "  PASS : Réplication pmxcfs vers $node OK"
    else
```

```
      echo "  WARN : Réplication pmxcfs vers $node non confirmée"
      WARNINGS=$((WARNINGS+1))
    fi
  done
  rm -f "$TESTFILE"
else
  echo "  FAIL : pmxcfs en LECTURE SEULE"
  ERRORS=$((ERRORS+1))
fi

# ── Test 8 : Services critiques ──────────────────────
echo "[ TEST 8/8 ] Services"
for svc in corosync pve-cluster pvedaemon pveproxy pvestatd; do
  if systemctl is-active --quiet $svc; then
    echo "  PASS : $svc actif"
  else
    echo "  FAIL : $svc INACTIF"
    ERRORS=$((ERRORS+1))
  fi
done

# ── Résultat final ───────────────────────────────────
echo ""
echo "═══════════════════════════════════════════════"
echo "  RÉSULTAT : $ERRORS erreur(s), $WARNINGS avertissement(s)"
echo "═══════════════════════════════════════════════"

if [ $ERRORS -eq 0 ] && [ $WARNINGS -eq 0 ]; then
  echo "  CLUSTER SAIN — Prêt pour la production"
  exit 0
elif [ $ERRORS -eq 0 ]; then
  echo "  CLUSTER OPÉRATIONNEL avec avertissements mineurs"
  exit 1
else
  echo "  CLUSTER EN ERREUR — Intervention requise"
  exit 2
fi
```

Exécution et résultat attendu sur un cluster sain :

```
═══════════════════════════════════════════════
  Validation cluster TechFlow SAS — Mon Mar 17 09:30:00 2026
═══════════════════════════════════════════════

[ TEST 1/8 ] Quorum
  PASS : Cluster quorate
[ TEST 2/8 ] Nom du cluster
  PASS : Nom correct (techflow-cluster)
[ TEST 3/8 ] Noeuds actifs
  PASS : 3/3 noeuds actifs
[ TEST 4/8 ] Identité des nœuds
  PASS : pve1 présent dans le cluster
  PASS : pve2 présent dans le cluster
  PASS : pve3 présent dans le cluster
[ TEST 5/8 ] Liens knet
  PASS : Ring0 connecté
  PASS : Ring1 connecté
[ TEST 6/8 ] Sécurité
  PASS : Transport knet actif
  PASS : Authentification sécurisée activée
[ TEST 7/8 ] pmxcfs
  PASS : pmxcfs monté sur /etc/pve
  PASS : pmxcfs accessible en écriture
  PASS : Réplication pmxcfs vers pve2 OK
  PASS : Réplication pmxcfs vers pve3 OK
[ TEST 8/8 ] Services
  PASS : corosync actif
  PASS : pve-cluster actif
```

```
  PASS : pvedaemon actif
  PASS : pveproxy actif
  PASS : pvestatd actif

═══════════════════════════════════════════════════
  RÉSULTAT : 0 erreur(s), 0 avertissement(s)
═══════════════════════════════════════════════════
  CLUSTER SAIN — Prêt pour la production
```

Vérification finale de l'interface web :

Depuis un navigateur, se connecter à `https://10.10.0.1:8006`. Dans le panneau de gauche, le nœud `pve1` apparaît sous le datacenter `techflow-cluster`, aux côtés de `pve2` et `pve3`. L'onglet **Datacenter > Cluster** affiche les trois nœuds avec leur état (**online**), leur version PVE et leur ID Corosync.

► RÉCAPITULATIF — ARCHITECTURE CLUSTER TECHFLOW SAS

Le cluster `techflow-cluster` est maintenant opérationnel avec :

- **3 nœuds** : pve1 (ID 1), pve2 (ID 2), pve3 (ID 3)
- **Quorum** : atteint avec 3/3 votes (tolérance à 1 panne nœud)
- **Transport** : knet chiffré (AES-256/SHA-256)
- **Redondance réseau** : ring0 (10.10.1.x) + ring1 (10.10.2.x)
- **pmxcfs** : réplication temps réel de `/etc/pve` sur les 3 nœuds

Les chapitres suivants s'appuieront sur cette infrastructure pour configurer la haute disponibilité (HA), le stockage partagé Ceph, et les politiques de migration.

✔ POINTS CLÉS DU CHAPITRE

- Corosync et pmxcfs forment le cœur du cluster Proxmox ; comprendre leur fonctionnement est indispensable pour diagnostiquer les incidents.
- Le quorum (votes_visibles > votes_total/2) protège contre le split-brain ; pour les clusters à 2 nœuds ou à nombre pair de nœuds, le QDevice est fortement recommandé.
- Le réseau cluster doit être **dédié, isolé et à faible latence** (< 2 ms recommandé).
- La synchronisation NTP et la cohérence DNS/hosts sont des prérequis non négociables.
- knet supporte nativement la redondance de liens, le chiffrement et la compression — activez toujours le chiffrement en production.
- Maîtriser `pvecm`, `corosync-cfgtool`, `corosync-quorumtool` et `corosync-cmapctl` permet de résoudre 95 % des incidents cluster sans intervention Proxmox Support.

PARTIE II

Virtualisation et stockage

4
Machines virtuelles, conteneurs et stockage

Ce chapitre constitue le coeur opérationnel de Proxmox VE. Après avoir posé les bases de l'installation et du réseau dans les chapitres précédents, nous allons maintenant créer, configurer et optimiser des machines virtuelles KVM et des conteneurs LXC, puis explorer en détail les différents backends de stockage disponibles. Le lab TechFlow SAS servira de fil conducteur tout au long du chapitre.

4.1 KVM : virtualisation complète

KVM (Kernel-based Virtual Machine) est le moteur de virtualisation complète intégré à Proxmox VE. Il s'appuie directement sur les extensions matérielles Intel VT-x et AMD-V pour exécuter des systèmes d'exploitation invités sans modification. Chaque VM dispose de ses propres ressources virtualisées : CPU, mémoire, disques, interfaces réseau. Le système invité n'a aucune connaissance de la couche de virtualisation, ce qui garantit une compatibilité maximale avec tout système d'exploitation.

QEMU (Quick Emulator) est le composant logiciel qui complète KVM. Alors que KVM gère l'accélération matérielle, QEMU émule le matériel périphérique (contrôleurs disque, cartes réseau, BIOS/UEFI) et fournit les outils de gestion en espace utilisateur. Proxmox VE orchestre l'ensemble via `libvirt` et son propre daemon `pve-manager`.

4.1.1 Création d'une VM via l'interface web

La création d'une VM dans Proxmox VE suit un assistant en plusieurs étapes accessible depuis le bouton **Create VM** en haut à droite de l'interface web.

Étape 1 — Général

Paramètres généraux de création d'une VM

Champ	Description	Recommandation
Node	Nœud du cluster qui hébergera la VM	Choisir selon la charge actuelle
VM ID	Identifiant numérique unique (100–999999)	Utiliser une plage cohérente (ex. 100-199 pour prod)
Name	Nom descriptif de la VM	Respecter une convention (ex. prd-dc01)
Resource Pool	Pool de ressources pour la délégation	Utile en environnement multi-tenant
Start at boot	Démarrage automatique avec le nœud	Activer pour les services critiques

Étape 2 — OS

Sélectionner le type de système d'exploitation permet à Proxmox VE d'optimiser les paramètres par défaut. Pour Windows, il activera automatiquement les pilotes VirtIO appropriés. Pour Linux, il choisira les contrôleurs VirtIO natifs.

Étape 3 — System

Options système avancées

Option	Valeur possible	Usage
Graphic card	VGA, SPICE, VirtIO-GPU	VirtIO-GPU pour perfs graphiques
Machine type	i440fx, q35	q35 pour PCIe, TPM, Secure Boot
BIOS	SeaBIOS, OVMF (UEFI)	OVMF requis pour Windows 11, Secure Boot
TPM	v2.0	Obligatoire Windows 11
SCSI Controller	VirtIO SCSI single, LSI, MegaRAID	VirtIO SCSI single recommandé

Étape 4 — Disques

Le premier disque hébergera le système d'exploitation. La taille minimale recommandée est 32 Go pour Linux et 60 Go pour Windows Server.

Étape 5 — CPU

Configuration du processeur virtuel (voir section 4.1.2).

Étape 6 — Mémoire

Configuration de la RAM (voir section 4.1.3).

Étape 7 — Réseau

Modèles de cartes réseau disponibles

Modèle	Débit théorique	Usage recommandé
VirtIO	10 Gbit/s+	Linux, Windows avec pilotes
E1000/E1000e	1 Gbit/s	Compatibilité maximale
RTL8139	100 Mbit/s	Systèmes anciens uniquement
VMXNET3	10 Gbit/s	Compatibilité VMware Tools

Création via la CLI

L'outil `qm` permet de scripter la création de VM de manière reproductible :

```
# Création d'une VM basique (Debian 12)
qm create 200 \
  --name "srv-web01" \
  --memory 2048 \
  --cores 2 \
  --sockets 1 \
  --net0 virtio,bridge=vmbr0,tag=10 \
  --scsi0 local-lvm:32,format=raw \
  --ide2 local:iso/debian-12.5.0-amd64-netinst.iso,media=cdrom \
  --boot order=ide2 \
  --ostype l26 \
  --machine q35 \
  --bios ovmf \
  --efidisk0 local-lvm:1,efitype=4m,pre-enrolled-keys=0

# Démarrer la VM
qm start 200

# Vérifier l'état
qm status 200
```

> ✔ **CONSEIL PRODUCTION**
> Systématisez la création de VM via des scripts ou Terraform (avec le provider Proxmox). Cela garantit la reproductibilité, facilite l'audit et élimine les erreurs de configuration manuelle. Le fichier de configuration de chaque VM est stocké dans `/etc/pve/nodes/<node>/qemu-server/<vmid>.conf`.

4.1.2 Options CPU (type, sockets, cores, NUMA)

La configuration CPU d'une VM dans Proxmox VE offre une granularité fine qui influence directement les performances et la compatibilité.

Topologie CPU

```
# Consulter la configuration CPU actuelle d'une VM
qm config 200 | grep -E "^(cpu|cores|sockets|numa)"

# Modifier la topologie CPU
qm set 200 --sockets 2 --cores 4 --vcpus 8
```

Terminologie CPU dans Proxmox VE

Paramètre	Définition	Impact
Sockets	Nombre de processeurs physiques virtuels	Affecte la topologie NUMA
Cores	Cœurs par socket	Unité de calcul de base

vCPUs total	Sockets × Cores	Threads QEMU alloués
CPU limit	Limite en fraction de CPU hôte (0.0–N)	Contrôle la consommation max
CPU units	Priorité relative (défaut 1024)	Ordonnancement entre VM

Types de CPU

Le type de CPU détermine quelles instructions processeur sont exposées à la VM. C'est un paramètre critique pour la migration live et la sécurité.

```
# Lister les types de CPU disponibles
qm cpu-models

# Types principaux :
# host        - Expose le CPU hôte tel quel (perf max, migration limitée)
# kvm64       - CPU générique x86-64, compatible migration cluster hétérogène
# x86-64-v2   - Baseline moderne (SSE4.2, POPCNT)
# x86-64-v3   - AVX, AVX2, BMI2 (recommandé clusters homogènes)
# Skylake-*   - Profils Intel spécifiques avec mitigations Spectre/Meltdown

# Configurer le type CPU
qm set 200 --cpu host,hidden=1,flags=+pcid

# Pour une VM Windows : masquer le flag KVM (anti-détection)
qm set 200 --cpu host,hidden=1
```

> ⚠ **PIÈGE COURANT — MIGRATION LIVE ET TYPE CPU**
> Si vous utilisez le type `host`, la migration live vers un nœud avec un modèle de CPU différent échouera. Dans un cluster hétérogène (mix Intel/AMD, générations différentes), utilisez `kvm64` ou `x86-64-v2` comme type de base. Réservez `host` uniquement aux VM mono-nœud ou aux clusters parfaitement homogènes.

NUMA (Non-Uniform Memory Access)

NUMA est essentiel pour les VM avec de nombreux vCPUs sur des hôtes multi-socket.

```
# Activer NUMA pour une VM
qm set 200 --numa 1

# Configurer un nœud NUMA spécifique
qm set 200 --numa0 cpus=0-3,hostnodes=0,memory=4096,policy=preferred

# Vérifier la topologie NUMA de l'hôte
numactl --hardware

# Épingler les vCPUs à des cœurs physiques spécifiques (CPU pinning)
qm set 200 --affinity 0-7

# Vérifier l'affectation des CPU
taskset -p $(pgrep -f "kvm.*200")
```

Flags CPU et sécurité

Proxmox VE expose les mitigations Spectre/Meltdown via des flags CPU :

```
# Activer les mitigations de sécurité recommandées
qm set 200 --cpu host,flags=+spec-ctrl+ssbd+md-clear+pdpe1gb

# Drapeaux importants :
# +spec-ctrl   : Spectre v2 (IBRS/IBPB)
# +ssbd        : Spectre v4 (Speculative Store Bypass Disable)
# +md-clear    : MDS (Microarchitectural Data Sampling)
# +pdpe1gb     : Huge pages 1 Go
# +aes         : Accélération AES-NI
```

4.1.3 Mémoire et ballooning

La gestion de la mémoire dans Proxmox VE permet d'équilibrer les ressources dynamiquement entre les VM actives.

Configuration de base

```
# Configurer la mémoire fixe
qm set 200 --memory 4096

# Configurer la mémoire avec ballooning
# memory = maximum | balloon = minimum garanti
qm set 200 --memory 8192 --balloon 2048
```

Le mécanisme de ballooning

Le ballooning est un mécanisme de gestion dynamique de la mémoire. Un pilote `virtio-balloon` s'exécute dans la VM et peut gonfler (récupérer de la mémoire pour l'hôte) ou dégonfler (restituer la mémoire à la VM) selon les besoins.

Paramètres de mémoire Proxmox VE

Paramètre	Valeur	Description
memory	512–N Mo	Mémoire maximale allouée
balloon	0–memory Mo	Minimum garanti (0 = désactivé)
shares	0–50000 (défaut 1000)	Priorité lors de la pression mémoire
hugepages	1024, 2 (2 Mo), 1048576 (1 Go)	Pages mémoire de grande taille

Huge Pages

Les huge pages réduisent la pression sur le TLB (Translation Lookaside Buffer) et améliorent significativement les performances des VM à forte charge mémoire.

```
# Vérifier les huge pages disponibles sur l'hôte
grep -i hugepage /proc/meminfo

# Configurer les huge pages sur l'hôte (2 Mo)
echo 2048 > /proc/sys/vm/nr_hugepages

# Rendre la configuration persistante
cat >> /etc/sysctl.conf << 'EOF'
vm.nr_hugepages = 2048
EOF

# Activer les huge pages pour une VM spécifique
qm set 200 --hugepages 2

# Huge pages de 1 Go (nécessite flag CPU pdpe1gb)
qm set 200 --hugepages 1024
```

✔ CONSEIL PRODUCTION — BALLOONING ET BASES DE DONNÉES

Désactivez le ballooning (`--balloon 0`) pour les VM hébergeant des bases de données (MySQL, PostgreSQL, Oracle). Ces applications gèrent leur propre cache mémoire de manière agressive et des variations de mémoire disponible peuvent provoquer des dégradations de performance importantes. Allouez une mémoire fixe cohérente avec la configuration de la base de données.

Mémoire partagée et KSM

KSM (Kernel Samepage Merging) fusionne les pages mémoire identiques entre VM pour réduire la consommation globale :

```
# Vérifier l'état de KSM
cat /sys/kernel/mm/ksm/run

# Activer KSM
echo 1 > /sys/kernel/mm/ksm/run

# Configurer l'agressivité de KSM
echo 1000 > /sys/kernel/mm/ksm/sleep_millisecs
echo 200 > /sys/kernel/mm/ksm/pages_to_scan
```

```
# Statistiques KSM
cat /sys/kernel/mm/ksm/pages_shared
```

4.1.4 Contrôleurs disque (VirtIO, SCSI, SATA)

Le choix du contrôleur disque a un impact direct sur les performances I/O de la VM.

Architecture des contrôleurs

Comparaison des contrôleurs disque

Contrôleur	IOPS max	Latence	Pilote requis	Usage recommandé
VirtIO Block	Très élevé	Très faible	virtio-blk (Linux natif)	Linux, charges I/O intensives
VirtIO SCSI	Très élevé	Très faible	virtio-scsi (Linux natif)	Linux, plusieurs disques, trim/discard
SATA	Moyen	Moyen	Natif (ahci)	Compatibilité maximale
IDE	Faible	Élevée	Natif	Legacy, CD-ROM uniquement
LSI Logic SAS	Moyen	Moyen	Windows natif	Windows sans VirtIO

Configuration des disques via CLI

```
# Ajouter un disque VirtIO SCSI
qm set 200 --scsi0 local-lvm:50,cache=writeback,iothread=1,discard=on

# Paramètres importants :
# cache=writeback   : Cache en écriture (perf+, risque si coupure)
# cache=none        : Pas de cache (sécurité max, O_DIRECT)
# cache=writethrough: Cache lecture uniquement
# iothread=1        : Thread I/O dédié par disque (recommandé)
# discard=on        : Propagation TRIM/UNMAP vers le stockage

# Ajouter un disque de données supplémentaire
qm set 200 --scsi1 local-lvm:100,cache=none,iothread=1,discard=on

# Ajouter un lecteur CD-ROM
qm set 200 --ide2 local:iso/debian-12.5.0-amd64-netinst.iso,media=cdrom

# Configurer l'ordre de démarrage
qm set 200 --boot order=scsi0

# Activer le SSD emulation (alignment correct pour SSD/NVMe)
qm set 200 --scsi0 local-lvm:50,ssd=1,discard=on
```

> ⚠ **PIÈGE COURANT — CACHE WRITEBACK ET ALIMENTATION**
>
> Le cache `writeback` offre les meilleures performances mais peut entraîner une corruption des données en cas de coupure d'alimentation inattendue si l'hôte ne dispose pas d'un onduleur. Sur des clusters de production, préférez `cache=none` (sécurité maximale) ou `cache=writeback` uniquement avec un UPS et un contrôleur RAID avec cache batterie (BBU). Ne jamais utiliser `writeback` sur un stockage NFS sans protection adéquate.

Gestion des périphériques passthrough

```
# Passthrough d'un disque physique entier (dangereux !)
qm set 200 --scsi2 /dev/sdb

# Passthrough PCIe (NVMe, GPU) - nécessite IOMMU activé
# Voir chapitre 6 pour la configuration IOMMU complète
qm set 200 --hostpci0 0000:01:00.0,pcie=1,rombar=1
```

4.1.5 Réseau VirtIO et pilotes

VirtIO est le standard de virtualisation para-virtualisée pour les périphériques réseau. Il offre des performances proches du natif en éliminant l'émulation complète du matériel.

Configuration réseau

```
# Ajouter une interface réseau VirtIO
qm set 200 --net0 virtio,bridge=vmbr0,tag=10,queues=4,rate=1000
```

```
# Paramètres réseau importants :
# bridge    : Bridge Linux à utiliser
# tag       : VLAN ID (optionnel)
# queues    : Multiqueue virtio (1 queue par vCPU, max 64)
# rate      : Limite de bande passante en Mo/s
# macaddr   : Adresse MAC fixe (auto si omis)
# firewall  : Activer le pare-feu Proxmox (0|1)

# Ajouter une seconde interface sur un autre bridge
qm set 200 --net1 virtio,bridge=vmbr1,tag=20

# Vérifier la configuration réseau
qm config 200 | grep "^net"
```

Multiqueue VirtIO

```
# Activer le multiqueue pour améliorer les performances réseau
# avec plusieurs vCPUs (1 queue par vCPU recommandé)
qm set 200 --net0 virtio,bridge=vmbr0,queues=4

# Dans la VM Linux, activer les queues
ethtool -L eth0 combined 4

# Rendre persistant (dans la VM)
cat > /etc/udev/rules.d/70-persistent-net-queue.rules << 'EOF'
ACTION=="add", SUBSYSTEM=="net", KERNEL=="eth*", \
  RUN+="/usr/sbin/ethtool -L $name combined 4"
EOF
```

Pilotes VirtIO pour Windows

Les systèmes Windows nécessitent l'installation manuelle des pilotes VirtIO disponibles sur le site Fedora :

```
# Télécharger l'ISO des pilotes VirtIO
wget https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-
virtio/virtio-win.iso \
  -O /var/lib/vz/template/iso/virtio-win.iso

# Attacher l'ISO à la VM Windows
qm set 201 --ide3 local:iso/virtio-win.iso,media=cdrom
```

4.2 Configuration avancée des VM

4.2.1 Agents QEMU (qemu-guest-agent)

Le `qemu-guest-agent` est un daemon s'exécutant dans la VM qui permet à l'hôte Proxmox de communiquer avec le système invité de manière standardisée. Il est indispensable pour les opérations avancées.

Fonctionnalités apportées par l'agent

- Obtention de l'adresse IP réelle de la VM (visible dans le résumé Proxmox)
- Snapshots cohérents avec gel du système de fichiers (fsfreeze/fsthaw)
- Arrêt propre depuis l'interface Proxmox
- Synchronisation de l'horloge
- Exécution de commandes dans la VM depuis l'hôte

Activation dans Proxmox VE

```
# Activer le support agent dans la configuration de la VM
qm set 200 --agent enabled=1,freeze-fs-on-backup=1,fstrim_cloned_disks=1

# Vérifier que le socket QEMU Guest Agent est actif
ls /var/run/qemu-server/200.qga
```

Installation dans la VM (Linux Debian/Ubuntu)

```
# Dans la VM invitée
apt-get install -y qemu-guest-agent
```

```
systemctl enable --now qemu-guest-agent

# Vérifier l'état
systemctl status qemu-guest-agent
```

Installation dans la VM (Windows Server)

L'installation se fait via les pilotes VirtIO :

1. Monter l'ISO VirtIO dans la VM Windows
2. Exécuter `virtio-win-guest-tools.exe`
3. Sélectionner "QEMU Guest Agent" dans les composants

Utilisation depuis l'hôte Proxmox

```
# Exécuter une commande dans la VM via l'agent
qm guest exec 200 -- ls /var/log

# Obtenir les informations réseau de la VM
qm guest cmd 200 network-get-interfaces

# Forcer la synchronisation de l'horloge
qm guest cmd 200 guest-sync-delimited

# Geler les systèmes de fichiers (avant snapshot)
qm guest cmd 200 guest-fsfreeze-freeze

# Dégeler les systèmes de fichiers
qm guest cmd 200 guest-fsfreeze-thaw
```

✔ **CONSEIL PRODUCTION**

Activez toujours `freeze-fs-on-backup=1` dans la configuration de l'agent pour les VM hébergeant des bases de données. Cela garantit la cohérence des snapshots PBS (Proxmox Backup Server) en gelant les écritures pendant la sauvegarde. Sans cette option, vous risquez des snapshots inconsistants difficiles à restaurer.

4.2.2 Snapshots et rollback

Les snapshots Proxmox VE capturent l'état complet d'une VM (disques, mémoire RAM, configuration) à un instant donné, permettant un retour en arrière rapide.

Types de snapshots

Comparaison des types de snapshots

Type	Contenu	Usage
Snapshot (VM running)	Disques + RAM + config	Avant mise à jour risquée
Snapshot (VM stopped)	Disques + config uniquement	Sauvegarde légère
Backup PBS	Disques complets (incrémental)	Sauvegarde longue durée

Gestion des snapshots

```
# Créer un snapshot (VM en cours d'exécution, avec RAM)
qm snapshot 200 snap-avant-maj \
  --description "Avant mise à jour kernel 6.8" \
  --vmstate 1

# Créer un snapshot sans sauvegarder la RAM (plus rapide)
qm snapshot 200 snap-config-ok --vmstate 0

# Lister les snapshots d'une VM
qm listsnapshot 200

# Restaurer un snapshot (rollback)
qm rollback 200 snap-avant-maj

# Supprimer un snapshot
qm delsnapshot 200 snap-avant-maj
```

```
# Supprimer tous les snapshots
qm listsnapshot 200 | grep -v "^→" | awk '{print $2}' | \
  while read snap; do qm delsnapshot 200 "$snap"; done
```

Chaîne de snapshots

```
# Visualiser la chaîne complète
qm listsnapshot 200
# Sortie exemple :
# current
#    snap-initial (2024-01-15 10:00)
#         snap-web-installed (2024-01-15 14:30)
#              snap-avant-maj (2024-01-20 09:15)
```

⚠ PIÈGE COURANT — SNAPSHOTS ET PERFORMANCE

Les snapshots sur stockage LVM-thin ou ZFS n'affectent quasiment pas les performances car ils sont basés sur le mécanisme copy-on-write. En revanche, les snapshots sur stockage de type `image` (fichiers QCOW2 sur répertoire) peuvent dégrader significativement les I/O en raison de la chaîne de dépendances. Limitez le nombre de snapshots actifs simultanément à 3-4 maximum, et planifiez leur suppression après validation.

4.2.3 Templates et clones liés/complets

Les templates permettent de créer rapidement de nouvelles VM à partir d'une image de base standardisée.

Conversion d'une VM en template

```
# Préparer la VM avant conversion en template (dans la VM)
# Sur Linux : supprimer les informations spécifiques à l'instance
cloud-init clean --logs --seed
truncate -s 0 /etc/machine-id
rm -f /var/lib/dbus/machine-id
ln -s /etc/machine-id /var/lib/dbus/machine-id
history -c
poweroff

# Sur Proxmox : convertir en template (opération irréversible !)
qm template 200

# La VM 200 devient un template (icône spécifique dans l'UI)
```

Clones

Proxmox VE propose deux types de clones :

Clone lié vs Clone complet

Type	Mécanisme	Avantages / Inconvénients
Clone lié (Linked)	Partage les données du template (COW)	Rapide, économe en espace ; dépend du template
Clone complet (Full)	Copie indépendante complète	Indépendant, lent, consomme plus d'espace

```
# Clone complet (full) vers un autre stockage
qm clone 200 201 \
  --name "srv-web02" \
  --full \
  --storage local-lvm \
  --target pve-node2

# Clone lié (linked) - nécessite stockage COW (ZFS, LVM-thin)
qm clone 200 202 \
  --name "srv-web03" \
  --storage local-lvm

# Personnaliser le clone (Cloud-Init)
qm set 202 \
  --ciuser debian \
```

```
  --cipassword "$(openssl passwd -6 'MonMotDePasse')" \
  --ipconfig0 ip=192.168.10.30/24,gw=192.168.10.1 \
  --nameserver 192.168.10.1 \
  --searchdomain techflow.local \
  --sshkeys /root/.ssh/authorized_keys

# Régénérer l'image Cloud-Init
qm cloudinit update 202

# Démarrer le clone
qm start 202
```

Cloud-Init avec Proxmox VE

```
# Ajouter un disque Cloud-Init à une VM
qm set 200 --ide1 local-lvm:cloudinit

# Configurer Cloud-Init
qm set 200 \
  --citype nocloud \
  --cicustom "user=local:snippets/cloud-init-user.yaml"

# Créer un snippet Cloud-Init personnalisé
mkdir -p /var/lib/vz/snippets
cat > /var/lib/vz/snippets/cloud-init-user.yaml << 'EOF'
#cloud-config
users:
  - name: sysadmin
    groups: sudo
    shell: /bin/bash
    sudo: ALL=(ALL) NOPASSWD:ALL
    ssh_authorized_keys:
      - ssh-ed25519 AAAA... votre_cle_publique
packages:
  - qemu-guest-agent
  - htop
  - curl
runcmd:
  - systemctl enable --now qemu-guest-agent
EOF
```

4.2.4 Migration live et à froid

La migration permet de déplacer une VM d'un nœud à un autre sans interruption de service (live) ou après arrêt (à froid).

Prérequis de la migration live

- Stockage partagé entre les nœuds (NFS, Ceph, iSCSI) OU migration avec copie des disques
- Connectivité réseau entre nœuds
- CPU de même famille (ou type CPU compatible)
- RAM suffisante sur le nœud destination

Migration via CLI

```
# Migration live (VM en cours d'exécution)
# Le stockage est partagé (Ceph/NFS) - instantané
qm migrate 200 pve-node2 --online

# Migration live avec copie des disques (stockage local)
qm migrate 200 pve-node2 --online --with-local-disks \
  --targetstorage local-lvm

# Migration à froid (VM arrêtée, plus fiable)
qm stop 200
qm migrate 200 pve-node2 --targetstorage local-lvm

# Surveiller la progression
tail -f /var/log/pve/tasks/active
```

Migration en masse

```
# Lister toutes les VM d'un nœud
qm list

# Migrer toutes les VM d'un nœud vers un autre
for vmid in $(pvesh get /nodes/pve-node1/qemu --output-format json | \
  jq -r '.[].vmid'); do
  echo "Migration VM $vmid ..."
  qm migrate $vmid pve-node2 --online --with-local-disks \
    --targetstorage local-lvm
done
```

> ✔ **CONSEIL PRODUCTION — PLANIFICATION DES MIGRATIONS**
> Avant une migration live, vérifiez la charge réseau entre les nœuds avec `iperf3`. La migration live transfère la mémoire RAM (dirty pages) via le réseau de migration. Sur un nœud avec 32 Go de RAM active, attendez-vous à transférer 5-15 Go en quelques minutes. Évitez les migrations pendant les heures de pointe et préférez le réseau de migration dédié (voir chapitre 3 sur la configuration VLAN).

4.2.5 Import OVA/OVF (migration VMware)

La migration depuis VMware vSphere vers Proxmox VE est une opération courante. Proxmox VE fournit des outils natifs pour importer les formats OVA et OVF.

Méthode 1 : Import OVA direct (Proxmox VE 8+)

```
# Importer un OVA VMware directement
qm importovf 300 /tmp/ma-vm-vmware.ova local-lvm

# Avec options avancées
qm importovf 300 /tmp/ma-vm-vmware.ova local-lvm \
  --format raw

# Vérifier l'import
qm config 300
```

Méthode 2 : Conversion manuelle via qemu-img

```
# Extraire l'OVA (c'est une archive tar)
mkdir /tmp/ova-extract
tar -xf /tmp/ma-vm-vmware.ova -C /tmp/ova-extract/
ls /tmp/ova-extract/

# Convertir le VMDK en format raw ou qcow2
qemu-img convert \
  -f vmdk \
  -O raw \
  /tmp/ova-extract/ma-vm-disk.vmdk \
  /tmp/ma-vm-disk.raw

# Vérifier la progression (conversion volumineuse)
qemu-img info /tmp/ma-vm-disk.raw

# Créer une VM vide et importer le disque
qm create 300 --name "vm-migree-vmware" --memory 4096 --cores 4

qm importdisk 300 /tmp/ma-vm-disk.raw local-lvm

# Attacher le disque importé à la VM
qm set 300 --scsi0 local-lvm:vm-300-disk-0 --boot order=scsi0

# Démarrer la VM (les pilotes VMware Tools peuvent nécessiter
# une désinstallation et l'installation de qemu-guest-agent)
qm start 300
```

Post-migration depuis VMware

```
# Après démarrage de la VM migrée (dans la VM Linux)

# Désinstaller VMware Tools si présent
```

```
vmware-uninstall-tools.pl 2>/dev/null || \
  apt-get remove --purge open-vm-tools -y

# Installer qemu-guest-agent
apt-get install -y qemu-guest-agent
systemctl enable --now qemu-guest-agent

# Reconfigurer le réseau si nécessaire
# (les noms d'interfaces peuvent avoir changé)
ip link show
```

⚠ PIÈGE COURANT — VMDK SPLITTÉ

VMware stocke parfois les disques volumineux en tranches de 2 Go (`disk-s001.vmdk` , `disk-s002.vmdk` ...). Dans ce cas, il faut d'abord fusionner les tranches avec `vmware-vdiskmanager` (depuis un hôte VMware) ou utiliser `qemu-img` avec le fichier descripteur principal (`disk.vmdk`) qui référence automatiquement les tranches.

4.3 Conteneurs LXC

4.3.1 LXC vs VM : quand choisir

LXC (Linux Containers) est la technologie de conteneurisation de niveau système utilisée par Proxmox VE. Contrairement aux VM KVM, les conteneurs LXC partagent le kernel de l'hôte et offrent une isolation légère mais efficace.

Comparaison VM KVM vs Conteneurs LXC

Critère	VM KVM	Conteneur LXC
Isolation	Complète (hyperviseur)	Partielle (namespaces kernel)
Overhead	5-10 % CPU/RAM	Inférieur à 1 %
Démarrage	30-120 secondes	1-5 secondes
Densité	10-50 VM/hôte	100-500 CT/hôte
OS supportés	Tout OS (Windows, Linux, BSD)	Linux uniquement
Kernel	Kernel dédié par VM	Kernel hôte partagé
Sécurité	Très élevée	Moyenne (namespaces)
Pilotes spéciaux	Possible (GPU passthrough)	Limité (dépend de l'hôte)

Quand utiliser LXC ?

- Services Linux stateless simples (Nginx, HAProxy, DNS)
- Environnements de développement et test
- Isolation de services sans besoin d'un OS complet
- Contraintes de densité élevée sur l'hôte
- Services ne nécessitant pas de kernel personnalisé

Quand utiliser KVM ?

- Windows Server, Windows 10/11
- Applications nécessitant des modules kernel spécifiques
- Isolation de sécurité maximale requise
- Bases de données de production critiques
- Migration depuis VMware/Hyper-V

4.3.2 Téléchargement et gestion des templates

Proxmox VE maintient un dépôt officiel de templates LXC basé sur les distributions Linux majeures.

```
# Lister les templates disponibles
pveam available

# Filtrer par distribution
pveam available | grep -i debian
```

```
pveam available | grep -i ubuntu
pveam available | grep -i alpine

# Mettre à jour la liste des templates
pveam update

# Télécharger un template
pveam download local debian-12-standard_12.7-1_amd64.tar.zst

# Lister les templates téléchargés
pveam list local

# Supprimer un template
pveam remove local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst

# Vérifier l'emplacement des templates
ls /var/lib/vz/template/cache/
```

Templates personnalisés

```
# Créer un template LXC personnalisé depuis un conteneur existant
# 1. Créer et configurer un conteneur de référence
pct create 900 local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst \
  --storage local-lvm --rootfs 4 --hostname template-base

pct start 900
pct exec 900 -- bash -c "apt-get update && apt-get upgrade -y && \
  apt-get install -y curl htop vim"
pct stop 900

# 2. Exporter comme template
vzdump 900 --compress zstd --dumpdir /var/lib/vz/template/cache/ \
  --mode stop

# Renommer pour utilisation comme template
mv /var/lib/vz/dump/lxc-900-*.tar.zst \
   /var/lib/vz/template/cache/custom-debian12-base.tar.zst
```

4.3.3 Création et configuration d'un conteneur

Création via l'interface web

L'assistant de création de conteneur est accessible via **Create CT** dans l'interface Proxmox VE. Les options principales sont :

- **CT ID** : Identifiant numérique (même plage que les VM)
- **Hostname** : Nom d'hôte du conteneur
- **Password** : Mot de passe root ou clé SSH
- **Template** : Image de base à utiliser
- **Disk** : Taille du système de fichiers racine
- **CPU** : Nombre de cœurs et limite CPU
- **Memory** : RAM et swap
- **Network** : Interface réseau et configuration IP
- **DNS** : Serveurs de noms

Création via CLI

```
# Créer un conteneur Debian 12 basique
pct create 300 local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst \
  --hostname ct-nginx01 \
  --storage local-lvm \
  --rootfs local-lvm:8 \
  --cores 2 \
  --memory 512 \
  --swap 256 \
  --net0 name=eth0,bridge=vmbr0,tag=10,ip=192.168.10.50/24,gw=192.168.10.1 \
  --nameserver 192.168.10.1 \
  --searchdomain techflow.local \
  --password "MonMotDePasse123!" \
```

```
  --unprivileged 1 \
  --features nesting=1

# Démarrer le conteneur
pct start 300

# Vérifier l'état
pct status 300

# Entrer dans le conteneur
pct enter 300

# Exécuter une commande sans entrer
pct exec 300 -- hostname -I

# Arrêter proprement
pct shutdown 300

# Arrêt forcé
pct stop 300
```

Configuration post-création

```
# Modifier la configuration d'un conteneur
pct set 300 --memory 1024 --swap 512

# Ajouter un disque de données
pct set 300 --mp0 local-lvm:20,mp=/data

# Modifier la configuration réseau
pct set 300 --net0
name=eth0,bridge=vmbr0,tag=10,ip=192.168.10.51/24,gw=192.168.10.1

# Activer le démarrage automatique
pct set 300 --onboot 1 --startup order=3,up=30

# Voir la configuration complète
pct config 300

# Le fichier de configuration est dans
cat /etc/pve/lxc/300.conf
```

Gestion des conteneurs en masse

```
# Lister tous les conteneurs
pct list

# Script de démarrage en masse avec délai
for ctid in 300 301 302 303; do
  pct start $ctid
  sleep 2
done

# Snapshot d'un conteneur
pct snapshot 300 snap-avant-maj --description "Avant mise à jour packages"

# Restaurer un snapshot
pct rollback 300 snap-avant-maj
```

4.3.4 Fonctionnalités avancées (nesting, apparmor)

Nesting (imbrication de conteneurs)

Le nesting permet d'exécuter Docker ou d'autres hyperviseurs légers à l'intérieur d'un conteneur LXC.

```
# Activer le nesting lors de la création
pct create 301 local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst \
  --features nesting=1,keyctl=1 \
  --unprivileged 1 \
```

```
  --hostname ct-docker01 \
  --storage local-lvm \
  --rootfs local-lvm:20 \
  --memory 2048 \
  --net0 name=eth0,bridge=vmbr0,tag=10,ip=192.168.10.60/24,gw=192.168.10.1

# Activer sur un conteneur existant
pct set 301 --features nesting=1,keyctl=1

# Dans le conteneur, installer Docker
pct exec 301 -- bash -c "
  apt-get update
  apt-get install -y ca-certificates curl
  install -m 0755 -d /etc/apt/keyrings
  curl -fsSL https://download.docker.com/linux/debian/gpg \
    -o /etc/apt/keyrings/docker.asc
  echo 'deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.asc] \
    https://download.docker.com/linux/debian bookworm stable' \
    > /etc/apt/sources.list.d/docker.list
  apt-get update
  apt-get install -y docker-ce docker-ce-cli containerd.io
  systemctl enable --now docker
"
```

Profils AppArmor

AppArmor fournit une couche de sécurité supplémentaire pour les conteneurs LXC en limitant leurs appels système.

```
# Vérifier le profil AppArmor actif
aa-status | grep lxc

# Les profils sont dans
ls /etc/apparmor.d/lxc/

# Profil par défaut pour les conteneurs non-privilégiés
cat /etc/apparmor.d/lxc/lxc-default-cgns

# Créer un profil personnalisé
cat > /etc/apparmor.d/lxc/lxc-techflow-custom << 'EOF'
profile lxc-techflow-custom flags=(attach_disconnected,mediate_deleted) {
  #include <abstractions/lxc/container-base>
  # Autoriser l'accès au périphérique loop (pour Docker in LXC)
  /dev/loop* rwk,
  /dev/loop-control rw,
}
EOF

apparmor_parser -r /etc/apparmor.d/lxc/lxc-techflow-custom
pct set 300 --lxc.apparmor.profile=lxc-techflow-custom
```

Conteneurs privilégiés vs non-privilégiés

Conteneurs privilégiés vs non-privilégiés

Aspect	Non-privilégié (recommandé)	Privilégié
UID root dans CT	Mappé vers UID 100000 sur l'hôte	Correspond à UID 0 de l'hôte
Sécurité	Élevée (breach = UID non-root hôte)	Faible (breach = root hôte)
Compatibilité	Certaines applis nécessitent privilèges	Maximale
Bind mounts	Nécessite ajustement des permissions	Transparent
Usage recommandé	Services web, DNS, cache	NFS server, iSCSI, cas spéciaux

> ⚠ **PIÈGE COURANT — CONTENEURS PRIVILÉGIÉS EN PRODUCTION**
> Évitez les conteneurs privilégiés en production. Un processus s'exécutant en root dans un conteneur privilégié est root sur l'hôte Proxmox si une faille d'isolation est exploitée. Préférez toujours les conteneurs non-privilégiés et utilisez les conteneurs privilégiés uniquement lorsque c'est absolument nécessaire (certains serveurs NFS, iSCSI targets, etc.).

4.3.5 Bind mounts et stockage partagé

Les bind mounts permettent de partager des répertoires de l'hôte Proxmox avec des conteneurs LXC.

Configuration des bind mounts

```
# Créer un répertoire sur l'hôte
mkdir -p /srv/techflow/shared-data
chown -R 100000:100000 /srv/techflow/shared-data  # Pour CT non-privilégié

# Ajouter un bind mount à un conteneur
pct set 300 --mp1 /srv/techflow/shared-data,mp=/mnt/shared,shared=1

# Vérifier dans la configuration
pct config 300 | grep mp

# Bind mount en lecture seule
pct set 300 --mp2 /srv/techflow/configs,mp=/mnt/configs,ro=1

# Pour les conteneurs non-privilégiés, ajuster le mappage UID
# UID 0 dans le CT = UID 100000 sur l'hôte
# UID 1000 dans le CT = UID 101000 sur l'hôte
```

Exemple de configuration complète d'un conteneur

```
# Contenu de /etc/pve/lxc/300.conf
arch: amd64
cores: 2
features: nesting=1,keyctl=1
hostname: ct-nginx01
memory: 512
mp0: local-lvm:vm-300-disk-1,mp=/data,size=20G
mp1: /srv/techflow/shared,mp=/mnt/shared
net0: name=eth0,bridge=vmbr0,gw=192.168.10.1,hwaddr=BC:
24:11:AA:BB:CC,ip=192.168.10.50/24,tag=10,type=veth
onboot: 1
ostype: debian
rootfs: local-lvm:vm-300-disk-0,size=8G
startup: order=3,up=30,down=60
swap: 256
tags: web;nginx;production
unprivileged: 1
lxc.idmap: u 0 100000 65536
lxc.idmap: g 0 100000 65536
```

Partage NFS entre conteneurs

```
# Monter un partage NFS sur l'hôte puis bind mount vers le CT
mount -t nfs 192.168.10.20:/exports/data /mnt/nfs-techflow
pct set 300 --mp3 /mnt/nfs-techflow,mp=/mnt/nfs,shared=1

# Rendre le montage NFS persistant sur l'hôte
echo "192.168.10.20:/exports/data /mnt/nfs-techflow nfs \
  rsize=8192,wsize=8192,timeo=14,intr 0 0" >> /etc/fstab
```

4.4 Gestion du stockage

4.4.1 Types de stockage supportés

Proxmox VE supporte une grande variété de backends de stockage, chacun avec ses caractéristiques propres.

Backends de stockage Proxmox VE

Type	Images VM	Templates/ISO	Snapshots	Notes
LVM	Oui	Non	Non	Performant, pas de COW
LVM-thin	Oui	Non	Oui	COW, snapshots, thin provisioning
Directory	Oui (QCOW2/RAW)	Oui	Oui (QCOW2)	Simple, flexible, peu performant
ZFS	Oui	Non	Oui	Très performant, intégrité données
NFS	Oui	Oui	Oui (QCOW2)	Partagé, dépend du réseau
CIFS/SMB	Oui	Oui	Oui (QCOW2)	Partagé, Windows-friendly
Ceph RBD	Oui	Non	Oui	Distribué, haute disponibilité
GlusterFS	Oui	Oui	Non	Stockage distribué
iSCSI	Oui	Non	Non	SAN, performances élevées
PBS	Non (backup)	Non	Non	Proxmox Backup Server uniquement

Gestion via `pvesm`

```
# Lister tous les stockages configurés
pvesm status

# Informations détaillées sur un stockage
pvesm status --storage local-lvm

# Lister le contenu d'un stockage
pvesm list local
pvesm list local-lvm

# Ajouter un stockage NFS
pvesm add nfs nfs-techflow \
  --server 192.168.10.20 \
  --export /exports/proxmox \
  --options vers=4.1 \
  --content images,backup,iso

# Supprimer un stockage (ne supprime pas les données)
pvesm remove nfs-techflow

# Activer/désactiver un stockage
pvesm set local-lvm --disable 1
pvesm set local-lvm --disable 0
```

4.4.2 LVM et LVM-thin

LVM (Logical Volume Manager) est le backend de stockage local par défaut de Proxmox VE. Il existe en deux variantes : LVM classique et LVM-thin.

LVM classique

```
# Vérifier la configuration LVM actuelle
pvs    # Physical Volumes
vgs    # Volume Groups
lvs    # Logical Volumes

# Ajouter un disque physique au Volume Group
pvcreate /dev/sdb
vgextend pve /dev/sdb

# Vérifier l'espace disponible
vgdisplay pve | grep -E "Free|Size"

# Ajouter le stockage LVM à Proxmox
pvesm add lvm lvm-extra \
  --vgname pve \
  --content images,rootdir
```

LVM-thin

LVM-thin (thin provisioning) permet l'allocation paresseuse de l'espace disque et supporte les snapshots copy-on-write.

```
# Créer un thin pool dans le VG existant
lvcreate -n thin-pool \
  --type thin-pool \
  --poolmetadatasize 4G \
  -l 95%FREE \
  pve

# Vérifier le thin pool
lvs -a -o +devices | grep thin

# Ajouter à Proxmox VE
pvesm add lvmthin local-lvm-extra \
  --vgname pve \
  --thinpool thin-pool \
  --content images,rootdir

# Surveiller le remplissage (CRITIQUE !)
lvs --rows pve/thin-pool -o data_percent,metadata_percent

# Alerte automatique via cron
cat > /etc/cron.d/lvm-thin-check << 'EOF'
*/15 * * * * root lvs --rows --noheadings -o lv_name,data_percent pve | \
  awk '$2 > 80 {print "ALERTE LVM-thin "$1" : "$2"% utilise"}' | \
  mail -s "LVM-thin alerte" admin@techflow.local 2>/dev/null; true
EOF
```

> ⚠ **PIÈGE COURANT — LVM-THIN ET SURCONSOMMATION**
> Le thin provisioning permet d'allouer plus d'espace que ce qui est physiquement disponible (over-provisioning). Si la consommation réelle dépasse la capacité physique, le thin pool se remplit à 100% et toutes les VM/CT utilisant ce pool s'arrêtent immédiatement avec des erreurs d'écriture. Configurez des alertes à 80% et 90% de remplissage et surveillez quotidiennement `lvs pve/thin-pool -o data_percent`.

Extension d'un thin pool

```
# Étendre le thin pool en ajoutant un disque
pvcreate /dev/sdc
vgextend pve /dev/sdc

# Étendre le thin pool
lvextend -l +50%FREE pve/thin-pool

# Vérifier après extension
lvs pve/thin-pool
```

4.4.3 ZFS : installation et configuration

ZFS (Zettabyte File System) est un système de fichiers avancé intégrant la gestion des volumes, la vérification d'intégrité, la compression et les snapshots nativement.

Installation et configuration initiale

```
# ZFS est disponible nativement dans Proxmox VE (module kernel)
# Vérifier la disponibilité
modprobe zfs
lsmod | grep zfs

# Créer un pool ZFS de type miroir (2 disques)
zpool create -f \
  -o ashift=12 \
  -O compression=lz4 \
  -O atime=off \
  -O xattr=sa \
  -O dnodesize=auto \
```

```
  -m /mnt/zfs-pool \
  tank \
  mirror /dev/sdb /dev/sdc

# Créer un pool RAIDZ-1 (3 disques, tolérance 1 panne)
zpool create -f \
  -o ashift=12 \
  -O compression=lz4 \
  -O atime=off \
  tank-raidz \
  raidz /dev/sdb /dev/sdc /dev/sdd

# Créer un pool RAIDZ-2 (4 disques minimum, tolérance 2 pannes)
zpool create -f \
  -o ashift=12 \
  -O compression=lz4 \
  -O atime=off \
  tank-raidz2 \
  raidz2 /dev/sdb /dev/sdc /dev/sdd /dev/sde

# Vérifier le pool
zpool status tank
zpool list
```

Configuration ZFS pour Proxmox VE

```
# Ajouter le pool ZFS à Proxmox
pvesm add zfspool zfs-tank \
  --pool tank \
  --content images,rootdir \
  --blocksize 4k

# Créer des datasets ZFS organisés
zfs create tank/proxmox
zfs create tank/proxmox/vm-images
zfs create tank/proxmox/backups
zfs create tank/proxmox/iso

# Configurer la compression
zfs set compression=lz4 tank/proxmox
zfs get compression tank/proxmox

# Configurer la cache ARC (limiter à 8 Go maximum)
echo "options zfs zfs_arc_max=8589934592" > /etc/modprobe.d/zfs.conf
update-initramfs -u

# Vérifier l'utilisation ARC
cat /proc/spl/kstat/zfs/arcstats | grep -E "^(size|hits|misses)"
```

Maintenance ZFS

```
# Scrub (vérification intégrité)
zpool scrub tank

# Vérifier l'état du scrub
zpool status tank | grep -A5 scan

# Planifier le scrub mensuel
cat > /etc/cron.d/zfs-scrub << 'EOF'
# Scrub ZFS le premier dimanche de chaque mois à 2h00
0 2 1-7 * 0 root /usr/sbin/zpool scrub tank
EOF

# Snapshots ZFS manuels
zfs snapshot tank/proxmox/vm-images@daily-$(date +%Y%m%d)

# Lister les snapshots
zfs list -t snapshot
```

```
# Automatiser les snapshots (avec sanoid)
apt-get install -y sanoid
cat > /etc/sanoid/sanoid.conf << 'EOF'
[tank/proxmox/vm-images]
  use_template = production

[template_production]
  frequently = 0
  hourly = 24
  daily = 30
  monthly = 3
  yearly = 0
  autosnap = yes
  autoprune = yes
EOF
systemctl enable --now sanoid.timer
```

L2ARC et SLOG

```
# Ajouter un SSD comme L2ARC (cache lecture étendu)
zpool add tank cache /dev/nvme0n1p1

# Ajouter un SSD comme SLOG (ZIL - accélération écriture synchrone)
zpool add tank log /dev/nvme0n1p2

# Vérifier la configuration
zpool status tank

# Statistiques L2ARC
cat /proc/spl/kstat/zfs/arcstats | grep l2
```

> ✔ **CONSEIL PRODUCTION — ZFS ET MÉMOIRE RAM**
> ZFS est très consommateur de RAM pour son cache ARC. Prévoyez au minimum 1 Go de RAM par To de stockage ZFS, avec un minimum absolu de 8 Go de RAM pour l'hôte. Sur des nœuds avec beaucoup de RAM (128 Go+), limitez l'ARC à 25-30% de la RAM totale pour laisser de la mémoire aux VM. Surveillez l'utilisation avec `zpool iostat -v tank 5`.

4.4.4 NFS et CIFS

Le stockage en réseau permet de partager des espaces de stockage entre plusieurs nœuds Proxmox VE, facilitant la migration live et la haute disponibilité.

Configuration NFS

```
# Installer le client NFS sur l'hôte Proxmox
apt-get install -y nfs-common

# Tester la connectivité NFS
showmount -e 192.168.10.20

# Ajouter un stockage NFS via pvesm
pvesm add nfs nfs-techflow \
  --server 192.168.10.20 \
  --export /exports/proxmox \
  --options vers=4.1,hard,timeo=100,retrans=3,_netdev \
  --content images,backup,iso,snippets

# Vérifier
pvesm status --storage nfs-techflow

# Tester l'écriture
pvesm alloc nfs-techflow 999 test-volume 1G
pvesm free nfs-techflow:999/test-volume

# Configuration NFS côté serveur (sur le NAS)
cat >> /etc/exports << 'EOF'
/exports/proxmox 192.168.10.0/24(rw,sync,no_subtree_check,no_root_squash)
```

```
EOF
exportfs -ra
```

Options NFS importantes

Options NFS recommandées pour Proxmox VE

Option	Valeur	Description
vers	4.1 ou 4.2	NFS v4 recommandé (meilleure sécurité, performances)
hard	(flag)	Réessaye indéfiniment en cas de timeout
timeo	100	Timeout en dixièmes de seconde (10 s)
retrans	3	Nombre de retransmissions avant erreur
rsize/wsize	1048576	Taille des blocs lecture/écriture (1 Mo)
nconnect	4-8	Connexions TCP parallèles (Linux 5.3+)
_netdev	(flag)	Montage après initialisation réseau

Configuration CIFS/SMB

```
# Installer le client CIFS
apt-get install -y cifs-utils

# Ajouter un stockage CIFS via pvesm
pvesm add cifs cifs-techflow \
  --server 192.168.10.30 \
  --share proxmox-storage \
  --username proxmox-user \
  --password "MonMotDePasse" \
  --domain TECHFLOW \
  --content images,backup,iso

# Avec un fichier de credentials (plus sécurisé)
cat > /root/.smbcredentials << 'EOF'
username=proxmox-user
password=MonMotDePasse
domain=TECHFLOW
EOF
chmod 600 /root/.smbcredentials
```

⚠ PIÈGE COURANT — NFS ET MIGRATION LIVE

La migration live nécessite que le stockage soit accessible depuis les deux nœuds. Avec NFS, assurez-vous que le même export est monté sur tous les nœuds du cluster avec des options cohérentes. Une différence d'options de montage (ex. `vers=3` sur un nœud, `vers=4` sur l'autre) peut provoquer des incohérences lors de la migration. Vérifiez avec `mount | grep nfs` sur chaque nœud.

4.4.5 Stockage Ceph (introduction, détails ch.8)

Ceph est un système de stockage distribué qui transforme plusieurs nœuds Proxmox VE en un cluster de stockage unifié, hautement disponible et auto-réparant.

Concepts fondamentaux

Composants Ceph

Composant	Rôle
MON (Monitor)	Maintient la topologie du cluster (CRUSH map), quorum requis (impair)
OSD (Object Storage Daemon)	Stocke les données sur les disques physiques (1 OSD = 1 disque)
MGR (Manager)	Collecte les métriques, héberge les modules (dashboard, prometheus)
MDS (Metadata Server)	Gère les métadonnées CephFS (non requis pour RBD)
RBD (RADOS Block Device)	Interface disque bloc utilisée par Proxmox pour les VM

Installation rapide (3 nœuds minimum)

```
# Sur chaque nœud Proxmox VE
pveceph install --repository no-subscription
```

```
# Initialiser Ceph sur le premier nœud
pveceph init --network 10.10.10.0/24 --cluster-network 10.10.11.0/24

# Créer les moniteurs (sur chaque nœud via l'interface web ou CLI)
pveceph mon create

# Créer les OSD (un par disque sur chaque nœud)
pveceph osd create /dev/sdb --crush-device-class hdd
pveceph osd create /dev/sdc --crush-device-class hdd

# Créer un pool RBD pour les VM
pveceph pool create vm-pool --size 3 --min_size 2

# Ajouter le pool à Proxmox
pvesm add rbd ceph-vms \
  --pool vm-pool \
  --content images,rootdir \
  --nodes pve-node1,pve-node2,pve-node3
```

Surveillance Ceph

```
# État global du cluster
ceph status
ceph health detail

# Utilisation du stockage
ceph df

# État des OSD
ceph osd status
ceph osd df

# IOPS et throughput en temps réel
ceph -w
```

> ► **NOTE**
> La configuration complète de Ceph, la mise en place d'un cluster de production, l'optimisation des performances et la gestion des pannes sont traitées en détail dans le **Chapitre 8 — Ceph : stockage distribué**. Cette section présente uniquement les concepts de base pour comprendre les options de stockage Proxmox VE.

4.5 Gestion des disques et images

4.5.1 qm disk et pct disk

Les commandes `qm disk` et `pct disk` permettent de gérer les disques attachés aux VM et conteneurs.

Gestion des disques VM (`qm disk`)

```
# Lister les disques d'une VM
qm config 200 | grep -E "^(scsi|ide|sata|virtio|efidisk)"

# Déplacer un disque vers un autre stockage
qm disk move 200 scsi0 local-lvm --delete

# Détacher un disque (sans le supprimer)
qm disk unlink 200 --idlist scsi1

# Importer un disque depuis un fichier image
qm importdisk 200 /tmp/disk.raw local-lvm --format raw

# Attacher le disque importé
qm set 200 --scsi1 local-lvm:vm-200-disk-1

# Supprimer un disque non attaché
pvesm free local-lvm:vm-200-disk-1
```

```
# Scanner les disques inutilisés d'une VM
qm rescan --vmid 200
```

Gestion des disques conteneurs (`pct disk`)

```
# Lister les disques d'un conteneur
pct config 300 | grep -E "^(rootfs|mp)"

# Déplacer le rootfs vers un autre stockage
pct disk move 300 rootfs local-lvm

# Déplacer un point de montage
pct disk move 300 mp0 local-lvm

# Déplacer avec suppression de l'ancienne copie
pct disk move 300 mp0 --storage local-lvm --delete
```

4.5.2 Resize en ligne

Le redimensionnement en ligne (live resize) permet d'étendre les disques de VM en cours d'exécution sans interruption.

Extension de disque VM

```
# Étendre le disque scsi0 de 20 Go supplémentaires
qm disk resize 200 scsi0 +20G

# Vérifier la nouvelle taille
qm config 200 | grep scsi0

# Dans la VM Linux (après extension depuis Proxmox)
# Vérifier que le kernel a détecté le changement
lsblk

# Étendre la partition et le système de fichiers
# Pour ext4 avec GPT
growpart /dev/sda 3    # Étendre la partition 3
resize2fs /dev/sda3   # Étendre le FS ext4

# Pour LVM dans la VM
pvresize /dev/sda3
lvextend -l +100%FREE /dev/ubuntu-vg/ubuntu-lv
resize2fs /dev/ubuntu-vg/ubuntu-lv

# Pour xfs
xfs_growfs /
```

Extension de disque conteneur

```
# Étendre le rootfs d'un conteneur de 10 Go
pct resize 300 rootfs +10G

# Vérifier
pct config 300 | grep rootfs

# Étendre un point de montage
pct resize 300 mp0 +20G

# Pour les conteneurs LXC sur LVM-thin, l'extension est instantanée
# Le système de fichiers dans le conteneur est automatiquement étendu
```

> ⚠ **PIÈGE COURANT — RÉDUCTION DE DISQUE**
>
> Proxmox VE ne supporte pas la réduction (shrink) de disques depuis l'interface. Tenter de réduire la taille d'un volume LVM ou ZFS sans avoir préalablement réduit le système de fichiers provoque une corruption immédiate et irréversible des données. Si vous devez réduire un disque, la procédure correcte est : sauvegarder les données, recréer la VM/CT avec le bon dimensionnement, restaurer. Ne jamais tenter de réduire un disque de VM en production sans une sauvegarde testée et validée.

Surveillance de l'espace disque

```
# Espace utilisé par toutes les VM et CT
pvesm status

# Détail par volume
pvesm list local-lvm | sort -k 3 -h -r

# Script de surveillance des disques proches de la saturation
#!/bin/bash
THRESHOLD=80
pvesm status | awk -v seuil="$THRESHOLD" 'NR>1 {
  if ($5 > 0) {
    pct = ($4 / $5) * 100;
    if (pct > seuil) {
      printf "ALERTE: %s utilise %.1f%%\n", $1, pct
    }
  }
}'
```

4.5.3 Import/export d'images

Export d'images VM

```
# Exporter une VM complète avec vzdump
vzdump 200 \
  --compress zstd \
  --mode snapshot \
  --storage local \
  --dumpdir /var/lib/vz/dump/

# Export vers PBS (Proxmox Backup Server)
vzdump 200 --storage pbs-backup

# Identifier le périphérique LVM
lvdisplay /dev/pve/vm-200-disk-0

# Exporter vers QCOW2
qemu-img convert \
  -f raw \
  -O qcow2 \
  /dev/pve/vm-200-disk-0 \
  /tmp/vm-200-disk.qcow2

# Compresser l'image QCOW2
qemu-img convert -c -O qcow2 \
  /dev/pve/vm-200-disk-0 \
  /tmp/vm-200-disk-compressed.qcow2
```

Import d'images

```
# Importer une image QCOW2
qm importdisk 200 /tmp/vm-200-disk.qcow2 local-lvm --format raw

# Copier une ISO dans le répertoire adéquat
cp ubuntu-24.04-live-server-amd64.iso /var/lib/vz/template/iso/

# Convertir entre formats
qemu-img convert -f vmdk -O qcow2 source.vmdk destination.qcow2
qemu-img convert -f qcow2 -O raw source.qcow2 destination.raw
qemu-img convert -f raw -O vmdk source.raw destination.vmdk

# Vérifier l'intégrité d'une image
qemu-img check /tmp/vm-200-disk.qcow2

# Obtenir les informations d'une image
qemu-img info /tmp/vm-200-disk.qcow2
```

Clonage entre nœuds via export/import

```
# Nœud source : exporter la VM
vzdump 200 --compress zstd --mode snapshot \
  --dumpdir /var/lib/vz/dump/

# Transférer vers le nœud destination
scp /var/lib/vz/dump/vzdump-qemu-200-*.vma.zst \
  root@pve-node2:/var/lib/vz/dump/

# Nœud destination : restaurer
qmrestore /var/lib/vz/dump/vzdump-qemu-200-*.vma.zst 201 \
  --storage local-lvm \
  --unique

# Vérifier la restauration
qm config 201
qm start 201
```

4.6 Lab TechFlow SAS : premières VM

TechFlow SAS est notre entreprise fictive d'exemple. Elle dispose d'un cluster Proxmox VE à 3 nœuds avec la configuration réseau suivante :

Infrastructure réseau TechFlow SAS

Réseau	Plage IP	Usage
Management	10.0.0.0/24	Accès Proxmox VE, IPMI
Production	192.168.10.0/24	VM et CT de production (VLAN 10)
Stockage	10.10.10.0/24	Réplication, Ceph (VLAN 100)
Migration	10.10.11.0/24	Migration live (VLAN 101)

4.6.1 VM Windows Server 2022 (contrôleur de domaine)

Nous allons créer la VM `prd-dc01`, contrôleur de domaine Active Directory pour le domaine `techflow.local`.

Prérequis

```
# Télécharger les pilotes VirtIO pour Windows
wget https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-
virtio/virtio-win.iso \
  -O /var/lib/vz/template/iso/virtio-win.iso

# Vérifier les ISOs disponibles
pvesm list local | grep iso
```

Création de la VM

```
# Créer la VM Windows Server 2022
qm create 101 \
  --name "prd-dc01" \
  --node pve-node1 \
  --memory 8192 \
  --balloon 0 \
  --cores 4 \
  --sockets 1 \
  --cpu host,hidden=1 \
  --machine q35 \
  --bios ovmf \
  --efidisk0 local-lvm:1,efitype=4m,pre-enrolled-keys=0 \
  --tpmstate0 local-lvm:4,version=v2.0 \
  --scsi0 local-lvm:60,cache=writeback,iothread=1,discard=on,ssd=1 \
  --scsihw virtio-scsi-single \
  --ide2 local:iso/windows-server-2022.iso,media=cdrom \
  --ide3 local:iso/virtio-win.iso,media=cdrom \
  --net0 virtio,bridge=vmbr0,tag=10 \
  --ostype win2k22 \
  --vga type=virtio,memory=32 \
```

```
  --agent enabled=1 \
  --onboot 1 \
  --startup order=1,up=120 \
  --description "Contrôleur de domaine Active Directory - TECHFLOW.LOCAL"

# Démarrer et ouvrir la console
qm start 101
```

Configuration réseau statique dans Windows (post-installation)

Après installation de Windows Server 2022 et des pilotes VirtIO, configurer l'adresse IP en PowerShell :

```
# Configurer l'adresse IP statique
New-NetIPAddress `
  -InterfaceAlias "Ethernet" `
  -IPAddress 192.168.10.10 `
  -PrefixLength 24 `
  -DefaultGateway 192.168.10.1

Set-DnsClientServerAddress `
  -InterfaceAlias "Ethernet" `
  -ServerAddresses 192.168.10.10, 192.168.10.11

# Renommer le serveur
Rename-Computer -NewName "PRD-DC01" -Restart
```

Installation du rôle Active Directory

```
# Installer les rôles AD DS et DNS
Install-WindowsFeature AD-Domain-Services, DNS `
  -IncludeManagementTools

# Promouvoir en contrôleur de domaine (nouvelle forêt)
Import-Module ADDSDeployment
Install-ADDSForest `
  -DomainName "techflow.local" `
  -DomainNetBIOSName "TECHFLOW" `
  -DomainMode "WinThreshold" `
  -ForestMode "WinThreshold" `
  -InstallDNS:$true `
  -SafeModeAdministratorPassword (ConvertTo-SecureString `
    "P@ssw0rdSecure123!" -AsPlainText -Force) `
  -Force:$true
```

Configuration Proxmox post-installation

```
# Créer un snapshot après installation complète
qm snapshot 101 snap-ad-installed \
  --description "AD DS installé et configuré - $(date)" \
  --vmstate 0

# Vérifier que l'agent remonte l'IP
qm agent 101 network-get-interfaces | grep -A2 "ip-address"

# Configurer les tags pour identification rapide
qm set 101 --tags "production,windows,active-directory,domain-controller"
```

4.6.2 VM Debian 12 (serveur web)

La VM `prd-web01` hébergera un serveur web Apache/PHP pour le site intranet de TechFlow SAS.

Création de la VM

```
# Télécharger l'ISO Debian 12
wget https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-12.5.0-
amd64-netinst.iso \
  -O /var/lib/vz/template/iso/debian-12.5.0-amd64-netinst.iso

# Créer la VM
qm create 102 \
```

```
  --name "prd-web01" \
  --node pve-node1 \
  --memory 4096 \
  --balloon 1024 \
  --cores 4 \
  --sockets 1 \
  --cpu x86-64-v2 \
  --machine q35 \
  --bios ovmf \
  --efidisk0 local-lvm:1,efitype=4m,pre-enrolled-keys=0 \
  --scsi0 local-lvm:32,cache=none,iothread=1,discard=on,ssd=1 \
  --scsi1 local-lvm:50,cache=none,iothread=1,discard=on \
  --scsihw virtio-scsi-single \
  --ide2 local:iso/debian-12.5.0-amd64-netinst.iso,media=cdrom \
  --net0 virtio,bridge=vmbr0,tag=10,queues=4 \
  --ostype l26 \
  --vga type=serial0 \
  --serial0 socket \
  --agent enabled=1,freeze-fs-on-backup=1 \
  --onboot 1 \
  --startup order=3,up=60 \
  --description "Serveur web Apache/PHP - Intranet TechFlow"

# Démarrer
qm start 102
```

Configuration post-installation Debian

```
# Configurer l'IP statique (dans la VM, fichier /etc/network/interfaces)
cat > /etc/network/interfaces << 'EOF'
auto lo
iface lo inet loopback

auto ens18
iface ens18 inet static
  address 192.168.10.20
  netmask 255.255.255.0
  gateway 192.168.10.1
  dns-nameservers 192.168.10.10 192.168.10.11
  dns-search techflow.local
EOF

systemctl restart networking

# Installer et configurer les services
apt-get update && apt-get upgrade -y
apt-get install -y \
  qemu-guest-agent \
  apache2 \
  php8.2 \
  php8.2-fpm \
  libapache2-mod-php8.2 \
  mariadb-client \
  curl \
  htop \
  nfs-common

systemctl enable --now qemu-guest-agent apache2 php8.2-fpm

# Configurer le disque de données (/dev/sdb vers /data)
parted /dev/sdb --script mklabel gpt mkpart primary ext4 0% 100%
mkfs.ext4 /dev/sdb1
mkdir -p /data/www /data/logs

echo "UUID=$(blkid -s UUID -o value /dev/sdb1) /data ext4 defaults,noatime 0 2" \
  >> /etc/fstab
mount -a
```

Configuration Apache

```
# Configurer Apache pour TechFlow
cat > /etc/apache2/sites-available/techflow-intranet.conf << 'EOF'
<VirtualHost *:80>
  ServerName intranet.techflow.local
  DocumentRoot /data/www/intranet
  ErrorLog /data/logs/intranet-error.log
  CustomLog /data/logs/intranet-access.log combined

  <Directory /data/www/intranet>
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
EOF

a2ensite techflow-intranet
a2enmod rewrite php8.2
systemctl reload apache2
```

Snapshot et tags

```
# Sur l'hôte Proxmox : snapshot après configuration de base
qm snapshot 102 snap-base-configured \
  --description "Debian 12 + Apache + PHP configures" \
  --vmstate 0

# Configurer les tags
qm set 102 --tags "production,linux,debian,web,apache"
```

4.6.3 Conteneur LXC Nginx

Le conteneur `ct-proxy01` servira de reverse proxy Nginx pour exposer les services internes de TechFlow SAS.

Création du conteneur

```
# Télécharger le template Debian 12
pveam download local debian-12-standard_12.7-1_amd64.tar.zst

# Créer le conteneur LXC
pct create 300 local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst \
  --hostname ct-proxy01 \
  --storage local-lvm \
  --rootfs local-lvm:4 \
  --cores 2 \
  --memory 512 \
  --swap 256 \
  --net0 name=eth0,bridge=vmbr0,tag=10,ip=192.168.10.50/24,gw=192.168.10.1 \
  --nameserver 192.168.10.10 \
  --searchdomain techflow.local \
  --password "$(openssl rand -base64 24)" \
  --ssh-public-keys /root/.ssh/authorized_keys \
  --unprivileged 1 \
  --features nesting=0 \
  --onboot 1 \
  --startup order=2,up=30,down=10 \
  --tags "production,nginx,reverse-proxy" \
  --description "Reverse proxy Nginx - Frontend TechFlow SAS"

# Démarrer le conteneur
pct start 300

# Vérifier le démarrage
pct status 300
pct exec 300 -- hostname -I
```

Configuration Nginx dans le conteneur

```
# Installation des paquets
pct exec 300 -- apt-get update
```

```
pct exec 300 -- apt-get install -y nginx curl

# Créer la configuration Nginx
pct exec 300 -- bash -c "cat > /etc/nginx/sites-available/techflow-proxy.conf <<
'NGINXEOF'
upstream intranet_backend {
    server 192.168.10.20:80;
    keepalive 32;
}

server {
    listen 80;
    server_name intranet.techflow.local;

    add_header X-Frame-Options \"SAMEORIGIN\" always;
    add_header X-Content-Type-Options \"nosniff\" always;
    add_header X-XSS-Protection \"1; mode=block\" always;

    access_log /var/log/nginx/intranet-access.log;
    error_log  /var/log/nginx/intranet-error.log;

    location / {
        proxy_pass http://intranet_backend;
        proxy_http_version 1.1;
        proxy_set_header Connection \"\";
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
        proxy_connect_timeout 5s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    location /nginx-status {
        stub_status on;
        access_log off;
        allow 127.0.0.1;
        deny all;
    }
}
NGINXEOF"

# Activer le site et démarrer Nginx
pct exec 300 -- ln -s /etc/nginx/sites-available/techflow-proxy.conf \
  /etc/nginx/sites-enabled/
pct exec 300 -- rm -f /etc/nginx/sites-enabled/default
pct exec 300 -- nginx -t
pct exec 300 -- systemctl enable --now nginx
```

Vérification du lab TechFlow SAS

```
# Sur l'hôte Proxmox - vérifier l'état de toutes les ressources
echo "=== VM en cours d execution ==="
qm list

echo "=== Conteneurs en cours d execution ==="
pct list

echo "=== Utilisation du stockage ==="
pvesm status

echo "=== Ressources cluster ==="
pvesh get /cluster/resources --type vm 2>/dev/null

# Tests de connectivité depuis l'hôte
ping -c2 -W2 192.168.10.10 && echo "prd-dc01 (AD) : OK" || echo "prd-dc01 : KO"
ping -c2 -W2 192.168.10.20 && echo "prd-web01 (Web) : OK" || echo "prd-web01 : KO"
```

```
ping -c2 -W2 192.168.10.50 && echo "ct-proxy01 (Proxy) : OK" || echo "ct-proxy01 : KO"

# Test du reverse proxy
curl -s -o /dev/null -w "HTTP %{http_code}\n" \
  http://192.168.10.50/ -H "Host: intranet.techflow.local"
```

Tableau récapitulatif du lab TechFlow SAS

Inventaire VM/CT du lab TechFlow SAS

ID	Nom	Type	CPU	RAM	IP	Stockage	Rôle
101	prd-dc01	VM KVM	4 vCPU	8 Go	192.168.10.10	60 Go SSD	Active Directory DC
102	prd-web01	VM KVM	4 vCPU	4 Go	192.168.10.20	32+50 Go	Apache/PHP Intranet
300	ct-proxy01	LXC CT	2 vCPU	512 Mo	192.168.10.50	4 Go	Nginx Reverse Proxy

Script de démarrage ordonné

```
#!/bin/bash
# /usr/local/bin/techflow-startup.sh
# Démarrage ordonné de l infrastructure TechFlow SAS

set -euo pipefail
log() { echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*"; }

# 1. Contrôleur de domaine (priorité absolue)
log "Démarrage du contrôleur de domaine prd-dc01 (VM 101)..."
qm start 101
sleep 90

# Attendre que l'AD réponde
for i in $(seq 1 20); do
  if qm agent 101 network-get-interfaces &>/dev/null; then
    log "AD opérationnel."
    break
  fi
  sleep 10
done

# 2. Reverse proxy
log "Démarrage du reverse proxy ct-proxy01 (CT 300)..."
pct start 300
sleep 15

# 3. Serveur web
log "Démarrage du serveur web prd-web01 (VM 102)..."
qm start 102
sleep 30

# 4. Bilan
log "═══ Bilan du démarrage ═══"
for vmid in 101 102; do
  status=$(qm status $vmid | awk '{print $2}')
  log "VM $vmid : $status"
done
pct_status=$(pct status 300 | awk '{print $2}')
log "CT 300 : $pct_status"
log "Infrastructure TechFlow SAS prête."
```

✔ CONSEIL PRODUCTION — ORDRE DE DÉMARRAGE

Utilisez les paramètres `--startup order=N,up=N,down=N` pour définir l'ordre de démarrage automatique lors du démarrage du nœud Proxmox. Le paramètre `up` définit le délai en secondes avant de démarrer la VM suivante dans l'ordre. Le paramètre `down` définit le délai d'attente avant l'arrêt forcé lors d'un shutdown du nœud. Ces paramètres sont essentiels pour respecter les dépendances entre services (AD doit être disponible avant les serveurs membres du domaine).

Résumé du chapitre

Ce chapitre a couvert l'ensemble des fonctionnalités de virtualisation et de stockage de Proxmox VE.

VM KVM - La création de VM via l'interface web et la CLI `qm` avec toutes les options de configuration - L'optimisation CPU (types, NUMA, CPU pinning, flags de sécurité Spectre/Meltdown) - La gestion de la mémoire avec ballooning et huge pages - Le choix et la configuration des contrôleurs disque (VirtIO SCSI recommandé) - Le réseau VirtIO avec multiqueue et les pilotes Windows

Configuration avancée - L'agent QEMU pour les snapshots cohérents et la visibilité réseau - Les snapshots et rollback avec leurs limitations selon le type de stockage - Les templates, clones liés/complets et Cloud-Init - La migration live et à froid avec ses prérequis réseau et CPU - L'import de VM VMware via OVA/OVF et conversion VMDK

Conteneurs LXC - Les cas d'usage LXC vs VM KVM, la densité et les limitations - La gestion des templates officiels et la création de templates personnalisés - Les fonctionnalités avancées : nesting (Docker in LXC), AppArmor, conteneurs privilégiés - Les bind mounts et le partage de stockage entre l'hôte et les CT

Stockage - Les différents backends : LVM, LVM-thin, ZFS, NFS, CIFS, Ceph - La configuration et la surveillance de LVM-thin (attention à la surconsommation) - ZFS avec compression lz4, ARC, snapshots automatiques via sanoid - NFS et CIFS pour le stockage partagé multi-nœuds - Introduction à Ceph (approfondi au chapitre 8)

Opérations sur les disques - Redimensionnement en ligne des disques VM et CT - Import/export d'images dans différents formats (raw, qcow2, vmdk)

Lab TechFlow SAS - VM Windows Server 2022 (contrôleur de domaine AD, 192.168.10.10) - VM Debian 12 (serveur web Apache/PHP, 192.168.10.20) - Conteneur LXC Nginx (reverse proxy, 192.168.10.50)

Points clés à retenir

Recommandations essentielles

Sujet	Recommandation
Type CPU VM	`host` pour performances max (mono-nœud), `x86-64-v2` pour cluster hétérogène
Contrôleur disque	VirtIO SCSI Single avec `iothread=1` et `discard=on`
Cache disque	`none` (sécurité) ou `writeback` avec UPS + RAID BBU uniquement
Mémoire bases de données	Désactiver le ballooning (`--balloon 0`)
Snapshots	LVM-thin et ZFS pour les snapshots performants (COW natif)
LVM-thin	Surveiller le remplissage, alertes à 80%, jamais dépasser 90%
ZFS RAM	1 Go minimum par To de stockage, limiter ARC à 25% de la RAM hôte
Conteneurs	Toujours non-privilégiés en production, AppArmor activé
Agent QEMU	Toujours installer, activer `freeze-fs-on-backup=1`
Migration live	Réseau dédié, CPU compatible, stockage partagé ou `--with-local-disks`

Le prochain chapitre abordera la haute disponibilité (HA) avec le gestionnaire HA de Proxmox VE, la fencing des nœuds défaillants, et la configuration des ressources HA pour garantir la continuité de service lors de la défaillance d'un nœud du cluster.

PARTIE III

Reseau defini par logiciel

5 SDN : fondamentaux et mise en œuvre

Le réseau défini par logiciel (Software-Defined Networking, SDN) représente l'une des évolutions les plus structurantes de Proxmox VE depuis la version 7. Avec Proxmox VE 9, le module SDN atteint une maturité qui le rend incontournable pour toute infrastructure de taille significative. Là où la configuration réseau traditionnelle reposait sur des fichiers `/etc/network/interfaces` gérés nœud par nœud, le SDN offre une vision centralisée, cohérente et dynamique du réseau virtuel à l'échelle du cluster entier.

Ce chapitre explore en profondeur les fondements du SDN dans Proxmox VE 9 : son architecture en couches (zones, VNets, subnets), les différents types de zones disponibles (simple, VLAN, VXLAN, EVPN), l'IPAM intégré, ainsi que la mise en œuvre pratique au travers du lab TechFlow SAS. À l'issue de ce chapitre, vous serez en mesure de concevoir et déployer une infrastructure réseau SDN complète, isolée et scalable.

5.1 Le SDN dans Proxmox VE 9

5.1.1 Architecture SDN : zones, VNets, subnets

Le SDN de Proxmox VE repose sur une hiérarchie à trois niveaux qui structure l'ensemble du réseau virtuel du cluster.

Le modèle hiérarchique

[Illustration : Hiérarchie SDN Proxmox VE 9 — zones, VNets, subnets]

Hiérarchie SDN : une zone encapsule plusieurs VNets, chaque VNet pouvant disposer de plusieurs subnets IPAM.

Les zones constituent la couche de base. Une zone définit la technologie d'isolation réseau utilisée et ses paramètres globaux. Elle peut être de type `simple`, `vlan`, `vxlan` ou `evpn`. Chaque zone est identifiée par un nom unique à l'échelle du cluster et ses paramètres sont stockés dans la configuration centralisée.

Les VNets (Virtual Networks) sont les segments réseau logiques rattachés à une zone. Un VNet est analogue à un bridge Linux, mais géré de façon centralisée. C'est à ce niveau que les VM et les conteneurs se connectent. Un VNet hérite du type d'isolation de sa zone parente, mais peut disposer de paramètres spécifiques (tag VLAN, VNI VXLAN, etc.).

Les subnets représentent la couche d'adressage IP associée à un VNet. Ils sont gérés par le module IPAM intégré de Proxmox et permettent l'attribution automatique ou statique des adresses IP aux VM et aux conteneurs.

La relation entre ces trois entités est la suivante :

```
Zone (technologie d'isolation)
  └── VNet 1 (segment L2)
  │     ├── Subnet 10.20.10.0/24
  │     └── Subnet 10.20.11.0/24
  └── VNet 2 (segment L2)
        └── Subnet 10.20.20.0/24
```

Cette hiérarchie permet de mutualiser la configuration technologique (au niveau zone) tout en offrant une granularité fine au niveau des segments réseau (VNet) et de l'adressage (subnet).

Propagation de la configuration

L'un des avantages majeurs du SDN est la propagation automatique de la configuration à tous les nœuds du cluster. Lorsqu'un administrateur crée ou modifie un VNet depuis l'interface web ou via l'API, Proxmox génère les configurations réseau correspondantes sur chaque nœud membre du cluster. Cela élimine la gestion manuelle et incohérente des fichiers `/etc/network/interfaces` sur chaque machine.

La propagation s'effectue via la tâche `pvesh` et le daemon `pve-sdn` qui surveille les changements de configuration et applique les modifications nécessaires.

> **• POINT CLÉ : APPLICATION DE LA CONFIGURATION SDN**
> Après toute modification SDN depuis l'interface web, il est obligatoire de cliquer sur le bouton **"Apply"** pour propager la configuration aux nœuds. Sans cette action, les modifications restent en attente et les bridges réseau ne sont pas créés ou mis à jour.

5.1.2 Composants : evpn, vxlan, simple, vlan

Proxmox VE 9 propose quatre types de zones SDN, chacun correspondant à un paradigme d'isolation réseau différent. Le choix du type de zone dépend des besoins en isolation, de la topologie physique et des capacités des équipements réseau sous-jacents.

Tableau comparatif des zones SDN

Comparaison des types de zones SDN dans Proxmox VE 9

Caractéristique	Simple	VLAN	VXLAN	EVPN
Isolation L2	Non (bridge partagé)	Oui (tag VLAN)	Oui (encapsulation UDP)	Oui (encapsulation UDP)
Routage L3 intégré	Non	Non	Non	Oui (anycast gateway)
Extension L2 multi-nœuds	Non	Dépend des switches	Oui (overlay)	Oui (overlay)
Complexité de configuration	Minimale	Faible	Modérée	Élevée
Dépendances matérielles	Aucune	Switch VLAN-aware	Aucune	BGP EVPN (FRRouting)
Cas d'usage typique	Dev/test, simple	Intégration VLAN existant	Isolation L2 multi-nœuds	Production large échelle
Plugin externe requis	Non	Non	Non	Oui (FRR)

Zone Simple

La zone `simple` est la plus accessible. Elle crée un bridge Linux standard sur chaque nœud, sans isolation entre les VNets. Tous les VNets d'une zone simple partagent le même domaine de broadcast. Ce type convient aux environnements de développement ou aux cas où l'isolation n'est pas requise.

Zone VLAN

La zone `vlan` exploite le standard IEEE 802.1Q pour isoler les segments réseau. Chaque VNet reçoit un tag VLAN distinct. Ce mode nécessite un bridge Linux configuré en mode VLAN-aware (`bridge-vlan-aware yes`). L'intégration avec les switches physiques se fait naturellement via les ports trunk.

Zone VXLAN

La zone `vxlan` implémente le protocole Virtual Extensible LAN (RFC 7348). Elle encapsule les trames Ethernet dans des paquets UDP, permettant de créer des segments L2 qui s'étendent sur un réseau IP. Chaque VNet est identifié par un VNI (VXLAN Network Identifier) unique. Ce mode ne nécessite aucune reconfiguration des switches physiques et offre une isolation complète entre les segments.

Zone EVPN

La zone `evpn` (Ethernet VPN) est la solution la plus avancée. Elle combine VXLAN pour le plan de données et BGP EVPN (RFC 7432) pour le plan de contrôle. Elle permet non seulement l'isolation L2 mais aussi le routage L3 entre VNets via des anycast gateways. Elle nécessite l'installation de FRRouting (FRR) sur les nœuds Proxmox.

► EVPN TRAITÉ DANS LE CHAPITRE SUIVANT

La zone EVPN, par sa complexité (BGP, FRR, anycast gateway), fait l'objet du chapitre 6. Ce chapitre se concentre sur les zones simple, VLAN et VXLAN, qui couvrent la majorité des besoins des infrastructures de taille moyenne.

5.1.3 IPAM intégré

L'IPAM (IP Address Management) est la composante qui gère l'attribution des adresses IP au sein des subnets SDN. Proxmox VE 9 intègre deux backends IPAM :

IPAM interne Proxmox

Le backend `pve` est l'IPAM natif de Proxmox. Il stocke les attributions d'adresses IP dans la base de données du cluster (`/etc/pve/sdn/`). Il supporte :

- L'attribution automatique des adresses IP aux VM et CT lors de leur création
- La définition de plages d'exclusion (gateway, broadcast, réservations statiques)
- La vérification des conflits d'adresses à la création
- L'affichage des attributions depuis l'interface web

IPAM NetBox

Proxmox peut s'intégrer avec une instance NetBox existante comme source de vérité pour les adresses IP. Cette intégration convient aux environnements qui disposent déjà d'un IPAM d'entreprise et souhaitent conserver un référentiel unique.

✔ IPAM ET CLOUD-INIT

Lorsque l'IPAM est configuré et qu'une VM utilise Cloud-Init, Proxmox peut injecter automatiquement la configuration réseau (adresse IP, gateway, DNS) dans la VM au démarrage. Cela supprime la nécessité de configurer le réseau manuellement dans chaque VM.

5.2 Activation et configuration du SDN

5.2.1 Activation du module SDN

Dans Proxmox VE 9, le module SDN est disponible mais ses dépendances doivent être installées. Voici la procédure complète d'activation sur un cluster.

Vérification des prérequis

```
# Vérifier la version de Proxmox VE
pveversion
# Résultat attendu : proxmox-ve: 9.x.x (running kernel: 6.x.x-x-pve)

# Vérifier que le module SDN est disponible dans le système de packages
dpkg -l | grep libpve-network-perl
```

Installation des paquets requis

```
# Sur chaque nœud du cluster
apt update

# Paquet principal pour le SDN Proxmox
apt install -y libpve-network-perl

# Pour les zones EVPN (optionnel, traité au chapitre 6)
apt install -y frr frr-pythontools

# DHCP intégré (requis si activation du serveur DHCP SDN)
apt install -y dnsmasq

# Vérifier l'installation
dpkg -l | grep -E 'libpve-network|frr|dnsmasq'
```

Activation du service SDN

```
# Activer et démarrer le service SDN sur chaque nœud
systemctl enable --now pve-sdn

# Vérifier l'état du service
```

```
systemctl status pve-sdn

# Vérifier les logs SDN
journalctl -u pve-sdn -n 50 --no-pager
```

Vérification des modules kernel

Le VXLAN nécessite le module kernel `vxlan`. Vérifier sa disponibilité :

```
# Charger le module vxlan
modprobe vxlan

# Vérifier que le module est chargé
lsmod | grep vxlan

# Rendre le chargement permanent au démarrage
echo "vxlan" > /etc/modules-load.d/proxmox-sdn.conf

# Vérifier les modules réseau disponibles
ls /lib/modules/$(uname -r)/kernel/drivers/net/
```

5.2.2 Interface web SDN

L'interface web SDN de Proxmox VE 9 est accessible depuis le menu **Datacenter > SDN**. Elle est organisée en plusieurs onglets.

Onglet SDN > Zones

Cet onglet liste toutes les zones SDN configurées dans le cluster. Pour chaque zone, on peut voir :

- Le nom de la zone
- Le type (simple, vlan, vxlan, evpn)
- Les nœuds sur lesquels elle est active
- L'état de déploiement (pending changes ou applied)

Le bouton **Add** permet de créer une nouvelle zone. Un formulaire contextuel s'affiche avec les paramètres spécifiques au type sélectionné.

Onglet SDN > VNets

Cet onglet liste tous les VNets définis. Pour chaque VNet :

- Le nom du VNet (identifiant utilisé comme nom de bridge)
- La zone parente
- Le tag (VLAN ID ou VNI selon le type de zone)
- Le commentaire/alias

Onglet SDN > Subnets

Accessible depuis la vue détaillée d'un VNet, cet onglet gère les plages d'adresses IP associées au VNet. On y configure le CIDR, la gateway, les options DHCP et SNAT.

Le bouton Apply

> ⚠ **ACTION CRITIQUE : APPLY**
> Toute modification de la configuration SDN (création de zone, VNet, subnet) ne prend effet qu'après avoir cliqué sur le bouton **"Apply"** présent en haut de la vue SDN. Ce bouton déclenche la régénération des fichiers de configuration réseau sur tous les nœuds et leur application immédiate. Sans cette étape, les bridges virtuels ne sont pas créés et les VM ne peuvent pas se connecter aux nouveaux VNets.

Navigation dans l'interface

```
Datacenter
  └── SDN
        ├── Zones        (gestion des zones SDN)
        ├── VNets        (gestion des réseaux virtuels)
        └── Controllers  (EVPN uniquement)
```

5.2.3 Fichiers de configuration

La configuration SDN est stockée dans le système de fichiers distribué de Proxmox (`/etc/pve/`) qui est répliqué automatiquement sur tous les nœuds via `pmxcfs`.

Structure des fichiers de configuration

```
/etc/pve/sdn/
├── zones.cfg          # Définition des zones SDN
├── vnets.cfg          # Définition des VNets
├── subnets.cfg        # Définition des subnets IPAM
├── controllers.cfg    # Contrôleurs EVPN (si utilisé)
└── ipam/
    └── pve.db         # Base de données IPAM interne (SQLite)
```

Format du fichier zones.cfg

```
# /etc/pve/sdn/zones.cfg — exemple de configuration multi-zones
simple: zone-dev
        dhcp none
        dns none
        ipam pve

vlan: zone-vlan-prod
        bridge vmbr0
        ipam pve

vxlan: zone-vxlan-prod
        peers 10.0.0.1,10.0.0.2,10.0.0.3
        mtu 1450
        ipam pve
```

Format du fichier vnets.cfg

```
# /etc/pve/sdn/vnets.cfg — exemple de configuration
vnet: vnet-it
        zone zone-vxlan-prod
        tag 1000

vnet: vnet-finance
        zone zone-vxlan-prod
        tag 1001

vnet: vnet-dmz
        zone zone-vxlan-prod
        tag 1002
```

Format du fichier subnets.cfg

```
# /etc/pve/sdn/subnets.cfg — exemple de configuration
subnet: zone-vxlan-prod-vnet-it-10.20.10.0-24
        network 10.20.10.0/24
        gateway 10.20.10.1
        vnet vnet-it
        dnszoneprefix it

subnet: zone-vxlan-prod-vnet-finance-10.20.20.0-24
        network 10.20.20.0/24
        gateway 10.20.20.1
        vnet vnet-finance
        dnszoneprefix finance
```

Configuration réseau générée sur les nœuds

Après l'application de la configuration SDN, Proxmox génère des fragments de configuration réseau dans `/etc/network/interfaces.d/` :

```
# Lister les fichiers générés par le SDN
ls -la /etc/network/interfaces.d/

# Afficher la configuration générée
cat /etc/network/interfaces.d/sdn
```

Exemple de contenu généré pour un VNet VXLAN :

```
# SDN generated configuration - DO NOT EDIT MANUALLY
auto vxlan-vnet-it
iface vxlan-vnet-it inet manual
    vxlan-id 1000
    vxlan-svcnodeip 0.0.0.0
    vxlan-local-tunnelip 10.0.0.1
    bridge-learning off
    bridge-arp-nd-suppress on

auto vnet-it
iface vnet-it inet manual
    bridge-ports vxlan-vnet-it
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware no
    mtu 1450
```

✗ NE PAS ÉDITER MANUELLEMENT LES FICHIERS SDN GÉNÉRÉS

Les fichiers dans `/etc/network/interfaces.d/sdn` sont régénérés automatiquement à chaque application de la configuration SDN. Toute modification manuelle sera écrasée lors du prochain Apply. Effectuez toujours vos modifications depuis l'interface web ou via `pvesh` .

Interaction avec pvesh

L'API REST de Proxmox, accessible via `pvesh` , permet de gérer le SDN en ligne de commande :

```
# Lister les zones SDN
pvesh get /cluster/sdn/zones

# Lister les VNets
pvesh get /cluster/sdn/vnets

# Obtenir les détails d'un VNet spécifique
pvesh get /cluster/sdn/vnets/vnet-it

# Appliquer la configuration SDN (équivalent du bouton Apply dans l'interface)
pvesh set /cluster/sdn

# Lister les subnets d'un VNet
pvesh get /cluster/sdn/vnets/vnet-it/subnets
```

5.3 Zone Simple

5.3.1 Création d'une zone simple

La zone `simple` est le point d'entrée idéal pour découvrir le SDN Proxmox. Elle crée un bridge Linux classique sur chaque nœud sans mécanisme d'isolation avancé. Elle est particulièrement adaptée aux environnements de développement, aux labs, ou aux petites infrastructures sans exigences d'isolation strictes.

Création depuis l'interface web

1. Naviguer vers **Datacenter > SDN > Zones**
2. Cliquer sur **Add > Simple**
3. Remplir le formulaire : - **ID** : `zone-dev` (identifiant unique, minuscules et tirets) - **MTU** : laisser vide pour la valeur par défaut (1500) - **Nodes** : sélectionner les nœuds sur lesquels déployer la zone - **IPAM** : `pve` (IPAM intégré)
4. Cliquer sur **Create**
5. Cliquer sur **Apply** pour appliquer la configuration

Création via pvesh

```
# Créer une zone simple via l'API
pvesh create /cluster/sdn/zones \
    --type simple \
    --zone zone-dev \
    --ipam pve
```

```
# Vérifier la création
pvesh get /cluster/sdn/zones/zone-dev

# Appliquer la configuration
pvesh set /cluster/sdn
```

Vérification sur les nœuds

```
# Après Apply, vérifier que le service SDN a traité la configuration
systemctl status pve-sdn

# Vérifier les interfaces réseau (les bridges sont créés quand des VNets sont
associés)
ip link show type bridge

# Consulter les logs de l'application
journalctl -u pve-sdn --since "5 minutes ago" --no-pager
```

5.3.2 VNet sur zone simple

Une fois la zone créée, il faut lui associer un ou plusieurs VNets pour que les VM puissent s'y connecter.

Création du VNet depuis l'interface web

1. Naviguer vers **Datacenter > SDN > VNets**
2. Cliquer sur **Add**
3. Remplir le formulaire : - **ID** : `vnet-dev-01` - **Zone** : `zone-dev` - **Alias** : `Réseau développement`
4. Cliquer sur **Create**
5. Cliquer sur **Apply**

Création du VNet via pvesh

```
# Créer un VNet dans la zone simple
pvesh create /cluster/sdn/vnets \
    --vnet vnet-dev-01 \
    --zone zone-dev \
    --alias "Réseau développement"

# Appliquer la configuration
pvesh set /cluster/sdn

# Vérifier la création du bridge sur le nœud local
bridge link show
ip link show vnet-dev-01
```

Association d'un subnet IPAM

```
# Ajouter un subnet IPAM au VNet
pvesh create /cluster/sdn/vnets/vnet-dev-01/subnets \
    --subnet 192.168.100.0/24 \
    --gateway 192.168.100.1 \
    --snat 0

# Vérifier le subnet
pvesh get /cluster/sdn/vnets/vnet-dev-01/subnets
```

Connexion d'une VM à un VNet

Depuis l'interface web, lors de la création ou modification d'une VM :

1. Naviguer vers l'onglet **Network** de la VM
2. Dans le champ **Bridge**, sélectionner `vnet-dev-01`
3. Sauvegarder la configuration

Via la ligne de commande :

```
# Modifier la configuration réseau d'une VM (VMID 100)
qm set 100 --net0 virtio,bridge=vnet-dev-01

# Vérifier la configuration
```

```
qm config 100 | grep net

# Pour un conteneur LXC (CTID 200)
pct set 200 --net0 name=eth0,bridge=vnet-dev-01,ip=dhcp
```

5.3.3 Cas d'usage

La zone simple est appropriée dans les scénarios suivants :

Environnement de développement et de test

Dans un lab ou un environnement de développement, l'isolation réseau entre les VM n'est souvent pas une exigence critique. La zone simple permet de créer rapidement des segments réseau sans complexité de configuration.

```
# Exemple : créer plusieurs VNets de dev rapidement
for i in 01 02 03; do
    pvesh create /cluster/sdn/vnets \
        --vnet "vnet-dev-${i}" \
        --zone zone-dev \
        --alias "Dev network ${i}"
done

# Appliquer en une seule fois
pvesh set /cluster/sdn

# Vérifier les bridges créés
ip link show type bridge | grep vnet-dev
```

Migration depuis la configuration traditionnelle

Les équipes migrant depuis des bridges manuels (`vmbr1` , `vmbr2` , etc.) peuvent utiliser la zone simple comme étape intermédiaire avant d'adopter des technologies plus avancées (VLAN, VXLAN).

Réseau de gestion interne

Pour un réseau de gestion des VM ne nécessitant pas d'isolation (monitoring interne, sauvegardes, etc.), la zone simple est suffisante et plus simple à administrer.

> ⚠ **LIMITATION IMPORTANTE DE LA ZONE SIMPLE**
> Dans une zone simple, tous les VNets partagent le même domaine de broadcast sur chaque nœud. Une VM connectée à `vnet-dev-01` peut potentiellement communiquer avec des VM sur `vnet-dev-02` si elles sont hébergées sur le même nœud. Si l'isolation réseau est une exigence, utiliser une zone VLAN ou VXLAN.

5.4 Zone VLAN

5.4.1 Configuration de la zone VLAN

La zone `vlan` exploite le mécanisme de tagging VLAN (IEEE 802.1Q) pour isoler les segments réseau. Elle est conçue pour s'intégrer avec une infrastructure physique existante disposant de switches managés.

Prérequis

- Un bridge Linux configuré en mode VLAN-aware sur chaque nœud
- Des switches physiques supportant les VLANs (802.1Q)
- Des ports trunk configurés sur les switches en direction des nœuds Proxmox

Configuration du bridge VLAN-aware

Avant de créer une zone VLAN, le bridge sous-jacent doit être configuré en mode `vlan-aware` dans `/etc/network/interfaces` :

```
# /etc/network/interfaces — extrait pour bridge VLAN-aware
auto vmbr0
iface vmbr0 inet static
    address 10.0.0.1/24
    gateway 10.0.0.254
    bridge-ports eno1
```

```
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes
    bridge-vids 2-4094
```

> ⚠ **MODIFICATION D'UN BRIDGE EXISTANT**
> Modifier un bridge existant pour le passer en mode `vlan-aware` peut interrompre la connectivité réseau des VM qui l'utilisent actuellement. Planifier cette opération pendant une fenêtre de maintenance. La commande `ifreload -a` applique les changements à chaud, mais peut provoquer une coupure réseau temporaire.

Appliquer la configuration réseau :

```
# Appliquer les changements réseau
# ATTENTION : peut interrompre la connexion SSH si le bridge de gestion est
modifié
ifreload -a

# Vérifier que le bridge est bien en mode vlan-aware
bridge vlan show

# Vérifier les VLAN IDs autorisés
bridge vlan show dev vmbr0
```

Création de la zone VLAN depuis l'interface web

1. Naviguer vers **Datacenter > SDN > Zones**
2. Cliquer sur **Add > VLAN**
3. Remplir le formulaire : - **ID** : `zone-vlan-prod` - **Bridge** : `vmbr0` (le bridge VLAN-aware configuré) - **IPAM** : `pve`
4. Cliquer sur **Create** puis **Apply**

Création via pvesh

```
# Créer une zone VLAN
pvesh create /cluster/sdn/zones \
    --type vlan \
    --zone zone-vlan-prod \
    --bridge vmbr0 \
    --ipam pve

# Vérifier la création
pvesh get /cluster/sdn/zones/zone-vlan-prod

# Appliquer
pvesh set /cluster/sdn
```

5.4.2 Tagged vs untagged

La compréhension de la distinction entre trames tagguées et non tagguées est essentielle pour configurer correctement les VNets VLAN.

Trames tagguées (802.1Q)

Une trame tagguée contient un en-tête VLAN de 4 octets (TPID 0x8100 + VLAN ID) indiquant l'appartenance au VLAN. Dans Proxmox, les VNets VLAN utilisent des trames tagguées sur le bridge VLAN-aware. La VM elle-même peut envoyer des trames non tagguées (mode access port) ou tagguées (mode trunk port).

Configuration d'un VNet taggué

```
# Créer un VNet avec tag VLAN 100 pour les serveurs
pvesh create /cluster/sdn/vnets \
    --vnet vnet-serveurs \
    --zone zone-vlan-prod \
    --tag 100 \
    --alias "VLAN Serveurs"

# Créer un VNet avec tag VLAN 200 pour les postes de travail
pvesh create /cluster/sdn/vnets \
```

```
    --vnet vnet-postes \
    --zone zone-vlan-prod \
    --tag 200 \
    --alias "VLAN Postes"

pvesh set /cluster/sdn

# Vérifier la configuration VLAN sur le bridge
bridge vlan show dev vmbr0
```

Mode access vs mode trunk pour les VM

Une VM connectée à un VNet VLAN en mode standard reçoit des trames sans tag VLAN (le bridge retire le tag). Pour qu'une VM reçoive des trames tagguées (conteneur réseau, routeur virtuel, firewall), activer le paramètre `vlanaware` sur le VNet :

```
# VNet avec VLAN passthrough (la VM voit les tags VLAN)
pvesh create /cluster/sdn/vnets \
    --vnet vnet-routeur \
    --zone zone-vlan-prod \
    --tag 300 \
    --vlanaware 1

pvesh set /cluster/sdn
```

Tableau : comportement des trames selon la configuration

Comportement des trames VLAN selon la configuration du VNet

Configuration VNet	Trame reçue par la VM	Trame émise par la VM	Cas d'usage typique
tag=100, vlanaware=0	Sans tag (untagged)	Tagguée VLAN 100 (par le bridge)	VM standard, OS classique
tag=100, vlanaware=1	Tagguée VLAN 100	Tagguée VLAN 100 (passthrough)	Routeur, firewall virtuel
Sans tag, vlanaware=1	Toutes trames tagguées	Toutes trames tagguées	VM trunk (multi-VLAN)

5.4.3 Intégration avec les switches physiques

L'intégration de la zone VLAN avec les switches physiques nécessite une configuration cohérente entre les ports du switch et les bridges Proxmox.

Configuration type d'un switch Cisco IOS

```
! Configuration d'un port trunk vers un nœud Proxmox
interface GigabitEthernet0/1
 description "Trunk vers proxmox-01"
 switchport mode trunk
 switchport trunk allowed vlan 100,200,300
 switchport trunk native vlan 1
 spanning-tree portfast trunk
!
```

Configuration type d'un switch avec ip route (Linux)

```
# Exemple avec un bridge Linux (VyOS, Debian router)
# Créer une interface VLAN sur le lien uplink
ip link add link eth0 name eth0.100 type vlan id 100
ip link set eth0.100 up
ip addr add 10.0.100.254/24 dev eth0.100

# Ajouter le VLAN au bridge
bridge vlan add dev eth0 vid 100 tagged
bridge vlan add dev eth0.100 vid 100 pvid untagged
```

Vérification de la connectivité VLAN

```
# Sur le nœud Proxmox, vérifier les VLANs actifs
bridge vlan show
```

```
# Vérifier les entrées FDB (Forwarding Database) — table MAC du bridge
bridge fdb show dev vmbr0

# Vérifier les statistiques par VLAN
bridge vlan show stats

# Capturer le trafic VLAN sur le bridge pour debug
tcpdump -i vmbr0 -e -n vlan 100 -c 20
```

Dépannage des problèmes VLAN

```
# Vérifier que le tag VLAN est bien configuré sur le VNet
pvesh get /cluster/sdn/vnets/vnet-serveurs

# Vérifier les VLANs autorisés sur le bridge
bridge vlan show

# Vérifier la table MAC du bridge
bridge fdb show

# Vérifier les logs SDN pour les erreurs de configuration
journalctl -u pve-sdn --since "10 minutes ago" --no-pager

# Tester la connectivité depuis une VM (via console)
qm terminal 101
# Depuis la VM :
# ping -c 4 10.0.100.1
```

5.5 Zone VXLAN

5.5.1 Principe du VXLAN (encapsulation UDP)

VXLAN (Virtual Extensible LAN) est un protocole d'encapsulation réseau défini dans la RFC 7348. Il permet de créer des segments Ethernet virtuels (overlay L2) sur un réseau IP (underlay L3).

Mécanisme d'encapsulation

[Illustration : Encapsulation VXLAN — trame Ethernet dans UDP/IP]

Encapsulation VXLAN : la trame Ethernet originale est encapsulée dans un paquet UDP avec l'en-tête VXLAN contenant le VNI, et transportée sur le réseau IP underlay entre les VTEPs.

Lorsqu'une VM envoie une trame Ethernet sur un VNet VXLAN, le VTEP (VXLAN Tunnel Endpoint) du nœud source :

1. Encapsule la trame Ethernet dans un en-tête VXLAN (incluant le VNI sur 24 bits)
2. Place l'en-tête VXLAN dans un segment UDP (port destination 4789 par défaut)
3. Encapsule dans un paquet IP avec l'IP du nœud source comme source et l'IP du nœud destination comme destination
4. Envoie le paquet sur le réseau underlay physique

À la réception, le VTEP du nœud destination désencapsule le paquet et livre la trame Ethernet originale à la VM cible via le bridge local.

Structure du paquet VXLAN

```
+--------------------+
| En-tête Ethernet |  (Underlay — réseau physique)
| src: MAC-node1   |
| dst: MAC-node2   |
+--------------------+
| En-tête IP       |  src: 10.0.0.1 (VTEP source)
|                  |  dst: 10.0.0.2 (VTEP destination)
+--------------------+
| En-tête UDP      |  src: port éphémère
|                  |  dst: 4789 (IANA VXLAN)
+--------------------+
| En-tête VXLAN    |  flags: 0x08 (VNI valid)
|                  |  VNI: 1000 (identifiant segment)
```

```
+---------------------+
| Trame Ethernet      |  (Overlay — trame originale VM)
| src: MAC-vm1        |
| dst: MAC-vm2        |
+---------------------+
```

Le VNI (VXLAN Network Identifier)

Le VNI est un identifiant de 24 bits (valeurs de 0 à 16 777 215) qui identifie le segment VXLAN. Dans Proxmox, chaque VNet VXLAN se voit attribuer un VNI unique. Des VMs sur des VNets différents (VNI différents) sont complètement isolées au niveau L2, même si elles se trouvent sur le même nœud physique.

Avantages du VXLAN pour Proxmox

- **Indépendance de l'infrastructure physique** : aucune reconfiguration des switches physiques nécessaire
- **Scalabilité** : 16 millions de segments possibles (VNI 24 bits) vs 4096 VLANs (12 bits)
- **Extension L2 sur L3** : les VM peuvent migrer entre nœuds tout en conservant leur adresse IP et leur appartenance L2
- **Isolation complète** : isolation garantie par les VNI distincts, aucun risque de fuite entre VNets

5.5.2 Configuration zone VXLAN

Prérequis réseau

- Connectivité IP entre tous les nœuds Proxmox (le réseau underlay)
- Le port UDP 4789 doit être accessible entre tous les nœuds (non bloqué par firewall)
- MTU suffisant sur le réseau underlay (recommandé : 1600 ou plus pour éviter la fragmentation des paquets VXLAN)

Vérification de la connectivité underlay

```
# Tester la connectivité IP entre nœuds (depuis pve-node1)
ping -c 4 10.0.0.2   # vers pve-node2
ping -c 4 10.0.0.3   # vers pve-node3

# Vérifier que le port UDP 4789 n'est pas bloqué
# Utiliser netcat pour tester (sur le nœud cible, lancer en écoute)
# Nœud cible : nc -u -l 4789
# Nœud source : nc -u 10.0.0.2 4789

# Vérifier les règles iptables sur chaque nœud
iptables -L INPUT -n -v | grep -E "4789|VXLAN|vxlan"

# Si le port est bloqué, l'autoriser
iptables -I INPUT -p udp --dport 4789 -j ACCEPT
# Rendre permanent via /etc/iptables/rules.v4 ou pve-firewall
```

Vérification et configuration de la MTU

```
# Vérifier la MTU du lien physique underlay
ip link show eno1 | grep mtu
# Résultat typique : mtu 1500

# MTU VXLAN = MTU underlay - 50 octets (en-têtes VXLAN/UDP/IP/Ethernet)
# Pour MTU underlay 1500 : MTU VXLAN = 1450
# Pour MTU underlay 9000 (jumbo) : MTU VXLAN = 8950

# Augmenter la MTU underlay si possible (dans /etc/network/interfaces)
# auto eno1
# iface eno1 inet manual
#     mtu 9000
```

Création de la zone VXLAN depuis l'interface web

1. Naviguer vers **Datacenter > SDN > Zones**
2. Cliquer sur **Add > VXLAN**

3. Remplir le formulaire : - **ID** : `zone-vxlan-prod` - **Peers Address List** : liste des adresses IP des nœuds séparées par des virgules (ex: `10.0.0.1,10.0.0.2,10.0.0.3`) - **MTU** : `1450` (pour underlay MTU 1500) - **IPAM** : `pve`
4. Cliquer sur **Create** puis **Apply**

Création via pvesh

```
# Créer la zone VXLAN avec unicast peers
pvesh create /cluster/sdn/zones \
    --type vxlan \
    --zone zone-vxlan-prod \
    --peers "10.0.0.1,10.0.0.2,10.0.0.3" \
    --mtu 1450 \
    --ipam pve

# Vérifier la création
pvesh get /cluster/sdn/zones/zone-vxlan-prod

# Appliquer la configuration
pvesh set /cluster/sdn
```

Vérification après création

```
# Vérifier les interfaces VXLAN créées (après création des VNets)
ip -d link show type vxlan

# Exemple de sortie :
# 5: vxlan-vnet-it: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 ...
#     vxlan id 1000 local 10.0.0.1 dev eno1 ...

# Vérifier les bridges créés
ip link show type bridge | grep vnet
```

5.5.3 VNets et isolation L2

Création de VNets VXLAN

```
# Créer un VNet IT avec VNI 1000
pvesh create /cluster/sdn/vnets \
    --vnet vnet-it \
    --zone zone-vxlan-prod \
    --tag 1000 \
    --alias "Réseau IT"

# Créer un VNet Finance avec VNI 1001
pvesh create /cluster/sdn/vnets \
    --vnet vnet-finance \
    --zone zone-vxlan-prod \
    --tag 1001 \
    --alias "Réseau Finance"

# Créer un VNet DMZ avec VNI 1002
pvesh create /cluster/sdn/vnets \
    --vnet vnet-dmz \
    --zone zone-vxlan-prod \
    --tag 1002 \
    --alias "Zone DMZ"

# Appliquer tous les changements en une seule opération
pvesh set /cluster/sdn
```

Vérification des interfaces créées

```
# Lister les interfaces VXLAN
ip link show type vxlan

# Vérifier les détails d'une interface VXLAN spécifique
ip -d link show vxlan-vnet-it

# Vérifier les bridges associés aux VNets
```

```
ip link show type bridge

# Vérifier la table FDB du tunnel VXLAN
bridge fdb show dev vxlan-vnet-it
# Les adresses MAC 00:00:00:00:00:00 pointent vers les nœuds pairs (flood entries)
```

Test d'isolation entre VNets

Pour vérifier que deux VMs sur des VNets différents ne peuvent pas communiquer directement :

```
# VM1 sur vnet-it (10.20.10.100), VM2 sur vnet-finance (10.20.20.100)
# Accéder à la console de VM1
qm terminal 101

# Depuis la console de VM1 : tenter de pinguer VM2 sur vnet-finance
ping -c 3 -W 1 10.20.20.100
# Résultat attendu : "Network unreachable" ou timeout
# Cela confirme l'isolation L2 garantie par les VNI distincts (1000 vs 1001)

# Tester la connectivité intra-VNet (deux VM sur le même vnet-it)
ping -c 3 10.20.10.101
# Résultat attendu : succès
```

Flux VXLAN visible sur le réseau physique

```
# Capturer les paquets VXLAN sur l'interface physique
# (trafic entre VM hébergées sur deux nœuds différents)
tcpdump -i eno1 -n udp port 4789 -c 20

# Exemple de sortie :
# 10.0.0.1.12345 > 10.0.0.2.4789: VXLAN, flags [I] (0x08), vni 1000
# 10.0.0.2.45678 > 10.0.0.1.4789: VXLAN, flags [I] (0x08), vni 1000

# Analyse plus détaillée avec tshark (si disponible)
tshark -i eno1 -Y "vxlan" -T fields \
    -e ip.src -e ip.dst -e vxlan.vni -e eth.src -e eth.dst -c 10
```

5.5.4 Dépannage VXLAN

Les problèmes VXLAN les plus courants et leurs résolutions :

Problème 1 : les VM sur deux nœuds différents ne se voient pas

```
# Étape 1 : vérifier la connectivité underlay entre les nœuds
ping -c 4 10.0.0.2

# Étape 2 : vérifier que le port UDP 4789 est accessible
# Sur le nœud destination, écouter sur le port 4789
nc -u -l 4789 &
# Sur le nœud source, envoyer un paquet test
echo "test" | nc -u 10.0.0.2 4789

# Étape 3 : vérifier les règles de firewall
iptables -L INPUT -n -v
nft list ruleset 2>/dev/null

# Étape 4 : vérifier la table FDB VXLAN
# Des entrées doivent apparaître pour les MAC des VM sur les nœuds distants
bridge fdb show dev vxlan-vnet-it

# Étape 5 : forcer la mise à jour de la table FDB via ARP
# (depuis une VM sur le VNet, pinger la VM distante)
# La table FDB se peuple automatiquement après le premier échange ARP
```

Problème 2 : fragmentation des paquets VXLAN

```
# Symptômes : connexions lentes, transferts qui s'arrêtent, latence élevée

# Vérifier la MTU des interfaces VXLAN
ip link show vxlan-vnet-it | grep mtu
ip link show vnet-it | grep mtu
```

```
# Tester avec une trame de grande taille depuis une VM (DF bit set)
# (depuis la console de la VM)
ping -c 4 -s 1400 -M do 10.20.10.101
# Si échec : "Frag needed and DF set" — problème de MTU

# Solution : réduire la MTU de la zone VXLAN
pvesh set /cluster/sdn/zones/zone-vxlan-prod --mtu 1400
pvesh set /cluster/sdn
```

Problème 3 : une VM ne peut pas pinguer sa gateway

```
# Vérifier que la gateway est configurée dans le subnet IPAM
pvesh get /cluster/sdn/vnets/vnet-it/subnets

# Note importante : dans une zone VXLAN pure (sans EVPN),
# la gateway doit être une VM/CT routeur sur le même VNet
# OU une interface du nœud physique sur ce bridge

# Vérifier la configuration IP de la VM
qm terminal 101
# Depuis la VM :
ip addr show eth0
ip route show
ip neigh show
```

Commandes de diagnostic SDN consolidées

```
# Vérifier l'état complet du SDN sur le nœud local
pvesh get /nodes/$(hostname)/network

# Recharger et réappliquer la configuration SDN
pvesh set /cluster/sdn

# Vérifier les logs du service SDN en temps réel
journalctl -u pve-sdn -f

# Vérifier les erreurs de configuration réseau
systemctl status networking
journalctl -u networking --since "1 hour ago" --no-pager

# Lister toutes les interfaces VXLAN et leur état
ip -d link show type vxlan 2>&1

# Vérifier la configuration complète des bridges
brctl show 2>/dev/null || bridge link show
```

5.6 IPAM : gestion des adresses IP

5.6.1 IPAM intégré Proxmox

L'IPAM intégré de Proxmox (backend `pve`) est un système de gestion des adresses IP léger, intégré directement dans le cluster. Il ne nécessite aucun service externe et ses données sont stockées dans la base de données distribuée du cluster, répliquée sur tous les nœuds via `pmxcfs`.

Architecture de l'IPAM intégré

[Illustration : Architecture IPAM Proxmox — subnets, pools, attributions]

L'IPAM Proxmox stocke les attributions d'adresses IP par VNet et par subnet. La base de données SQLite est répliquée via pmxcfs sur tous les nœuds du cluster.

Base de données IPAM

```
# Localisation de la base de données IPAM
ls -la /etc/pve/sdn/ipam/

# La base est un fichier SQLite géré par pmxcfs
file /etc/pve/sdn/ipam/pve.db

# Consulter les attributions IPAM directement (lecture seule)
```

```
sqlite3 /etc/pve/sdn/ipam/pve.db \
    "SELECT ip, mac, vmid, hostname FROM ipam ORDER BY ip;" 2>/dev/null

# Afficher la structure des tables
sqlite3 /etc/pve/sdn/ipam/pve.db ".schema" 2>/dev/null
```

Visualisation depuis l'interface web

L'IPAM est consultable depuis **Datacenter > SDN > VNets > [VNet] > Subnets**. Pour chaque subnet, on peut voir en cliquant sur **IPs** :

- Les adresses attribuées avec leur statut
- L'adresse MAC associée
- L'identifiant de la VM/CT qui détient l'adresse
- Le nom d'hôte (hostname) déclaré

5.6.2 Configuration des subnets

Création d'un subnet depuis l'interface web

1. Naviguer vers **Datacenter > SDN > VNets**
2. Sélectionner un VNet (ex: `vnet-it`)
3. Cliquer sur l'onglet **Subnets** puis **Add**
4. Remplir le formulaire : - **Subnet** : `10.20.10.0/24` - **Gateway** : `10.20.10.1` - **SNAT** : cocher si les VM doivent accéder à Internet via NAT depuis le nœud - **DNS zone prefix** : `it` (pour la résolution DNS inverse)
5. Cliquer sur **Create**

Création d'un subnet via pvesh

```
# Ajouter un subnet au VNet vnet-it
pvesh create /cluster/sdn/vnets/vnet-it/subnets \
    --subnet 10.20.10.0/24 \
    --gateway 10.20.10.1 \
    --snat 0 \
    --dnszoneprefix it

# Ajouter un subnet au VNet vnet-finance
pvesh create /cluster/sdn/vnets/vnet-finance/subnets \
    --subnet 10.20.20.0/24 \
    --gateway 10.20.20.1 \
    --snat 0 \
    --dnszoneprefix finance

# Ajouter un subnet au VNet vnet-dmz avec SNAT (accès Internet)
pvesh create /cluster/sdn/vnets/vnet-dmz/subnets \
    --subnet 10.20.30.0/24 \
    --gateway 10.20.30.1 \
    --snat 1 \
    --dnszoneprefix dmz

# Vérifier les subnets créés
pvesh get /cluster/sdn/vnets/vnet-it/subnets
pvesh get /cluster/sdn/vnets/vnet-finance/subnets
```

Configuration DHCP sur le subnet

> **► DHCP SDN DANS PROXMOX VE 9**
> Proxmox VE 9 intègre un serveur DHCP lié à l'IPAM SDN via `dnsmasq`. Lorsque DHCP est activé sur un subnet, les VM peuvent obtenir leur adresse IP via DHCP, et cette adresse est automatiquement enregistrée dans l'IPAM. La configuration DHCP est générée automatiquement depuis les paramètres du subnet.

```
# Vérifier que dnsmasq est installé (requis pour le DHCP SDN)
dpkg -l dnsmasq
apt install -y dnsmasq

# L'activation du DHCP se fait depuis l'interface web lors de la création du
subnet
```

```
# ou en modifiant le subnet existant (option "DHCP")

# Après activation, vérifier la configuration DHCP générée
ls /etc/pve/sdn/dhcp/
cat /etc/pve/sdn/dhcp/vnet-it.conf

# Vérifier que dnsmasq utilise bien la configuration SDN
systemctl status dnsmasq
```

5.6.3 Attribution automatique aux VM/CT

L'IPAM permet d'attribuer automatiquement des adresses IP aux VM et aux conteneurs lors de leur création, en combinaison avec Cloud-Init (pour les VM) ou les options réseau des CT.

Attribution automatique pour une VM via Cloud-Init

```
# Créer une VM avec Cloud-Init et adresse IP issue de l'IPAM
qm create 200 \
    --name srv-web-01 \
    --memory 2048 \
    --cores 2 \
    --net0 virtio,bridge=vnet-it \
    --ide2 local:cloudinit \
    --ipconfig0 ip=dhcp \
    --nameserver 10.20.10.1 \
    --searchdomain techflow.local \
    --ostype l26

# Ou spécifier une adresse IP statique (enregistrée dans l'IPAM)
qm set 200 --ipconfig0 ip=10.20.10.50/24,gw=10.20.10.1

# Régénérer le disque Cloud-Init après modification
qm cloudinit update 200
```

Attribution automatique pour un conteneur LXC

```
# Créer un conteneur avec adresse IP obtenue via DHCP SDN
pct create 300 local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst \
    --hostname ct-app-01 \
    --memory 512 \
    --net0 name=eth0,bridge=vnet-it,ip=dhcp \
    --rootfs local-lvm:8 \
    --start 1

# Vérifier l'adresse attribuée
pct exec 300 -- ip addr show eth0

# Consulter l'entrée IPAM correspondante
pvesh get /cluster/sdn/vnets/vnet-it/ips | python3 -c "
import json, sys
data = json.load(sys.stdin)
for entry in data:
    if entry.get('vmid') == 300:
        print(entry)
"
```

Consultation et gestion des attributions IPAM

```
# Lister toutes les attributions pour un VNet
pvesh get /cluster/sdn/vnets/vnet-it/ips

# Résultat exemple :
# [
#   {"ip": "10.20.10.1",  "comment": "Gateway vnet-it"},
#   {"ip": "10.20.10.50", "mac": "BC:24:11:AA:BB:CC", "vmid": 200, "name": "srv-
web-01"},
#   {"ip": "10.20.10.51", "mac": "BC:24:11:DD:EE:FF", "vmid": 300, "name": "ct-
app-01"}
# ]
```

```
# Ajouter manuellement une réservation statique
pvesh create /cluster/sdn/vnets/vnet-it/ips \
    --ip 10.20.10.1 \
    --comment "Gateway vnet-it"

# Supprimer une attribution (libérer l'adresse)
pvesh delete /cluster/sdn/vnets/vnet-it/ips \
    --ip 10.20.10.50
```

Vérification et résolution des conflits d'adresses

```
# L'IPAM vérifie automatiquement les conflits lors de l'attribution
# En cas de conflit, une erreur est retournée :
# "IP 10.20.10.50 already used by VM 200"

# Pour libérer une adresse orpheline (VM supprimée mais entrée IPAM non nettoyée)
pvesh delete /cluster/sdn/vnets/vnet-it/ips --ip 10.20.10.50

# Scanner les conflits potentiels (adresses utilisées sans entrée IPAM)
# Depuis une VM sur le VNet :
# nmap -sn 10.20.10.0/24 (scan ARP pour détecter les hôtes actifs)
# Comparer avec pvesh get /cluster/sdn/vnets/vnet-it/ips
```

5.7 Lab TechFlow SAS : SDN de base

5.7.1 Architecture réseau cible

TechFlow SAS est une PME du secteur technologique avec environ 80 employés répartis en trois départements : IT, Finance et une DMZ pour les services exposés sur Internet. L'infrastructure Proxmox se compose de trois nœuds physiques formant un cluster à haute disponibilité.

Topologie physique

[Illustration : Architecture réseau TechFlow SAS — cluster Proxmox 3 nœuds avec SDN VXLAN]

Architecture réseau TechFlow SAS : trois nœuds Proxmox interconnectés via un réseau d'administration 10.0.0.0/24 (underlay), avec overlay VXLAN pour les quatre VNets départementaux. Un routeur/firewall physique assure la connectivité Internet et le routage inter-VNets.

Inventaire des nœuds du cluster

Nœuds du cluster TechFlow SAS

Nœud	Hostname	IP Underlay (VXLAN)	Rôle dans le cluster
pve-node1	proxmox-01.techflow.local	10.0.0.1/24	Nœud primaire, quorum
pve-node2	proxmox-02.techflow.local	10.0.0.2/24	Nœud secondaire
pve-node3	proxmox-03.techflow.local	10.0.0.3/24	Nœud tertiaire

Plan d'adressage SDN TechFlow

Plan d'adressage VNets TechFlow SAS

VNet	VNI	Subnet	Gateway	Plage VM	Département
vnet-it	1000	10.20.10.0/24	10.20.10.1	10.20.10.10–254	Informatique
vnet-finance	1001	10.20.20.0/24	10.20.20.1	10.20.20.10–254	Finance
vnet-dmz	1002	10.20.30.0/24	10.20.30.1	10.20.30.10–254	DMZ / Services publics
vnet-storage	1003	10.20.40.0/24	10.20.40.1	10.20.40.10–254	Stockage partagé

5.7.2 Zone VXLAN production

Étape 1 : Préparation de chaque nœud

```
# Exécuter sur CHAQUE nœud du cluster (pve-node1, pve-node2, pve-node3)

# 1. Vérifier la connectivité underlay entre tous les nœuds
for node_ip in 10.0.0.1 10.0.0.2 10.0.0.3; do
    local_ip=$(ip route get "$node_ip" | awk '/src/{print $7}' 2>/dev/null)
```

```
    if [ "$local_ip" != "$node_ip" ]; then
        printf "Test vers %-12s : " "$node_ip"
        ping -c 1 -W 1 "$node_ip" &>/dev/null && echo "OK" || echo "ECHEC"
    fi
done

# 2. Installer les paquets requis
apt update && apt install -y libpve-network-perl dnsmasq

# 3. Charger le module kernel VXLAN
modprobe vxlan
echo "vxlan" > /etc/modules-load.d/vxlan-sdn.conf
lsmod | grep vxlan

# 4. Activer le service SDN
systemctl enable --now pve-sdn
systemctl status pve-sdn --no-pager
```

Étape 2 : Vérification de la configuration bridge underlay

```
# Sur chaque nœud, vérifier la configuration du bridge principal
grep -A 10 "auto vmbr0" /etc/network/interfaces
```

La configuration attendue pour `pve-node1` :

```
# /etc/network/interfaces (pve-node1)
auto lo
iface lo inet loopback

auto eno1
iface eno1 inet manual

auto vmbr0
iface vmbr0 inet static
    address 10.0.0.1/24
    gateway 10.0.0.254
    bridge-ports eno1
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes
```

Étape 3 : Création de la zone VXLAN production

```
# Depuis le nœud primaire (ou depuis l'interface web du datacenter)
pvesh create /cluster/sdn/zones \
    --type vxlan \
    --zone zone-vxlan-prod \
    --peers "10.0.0.1,10.0.0.2,10.0.0.3" \
    --mtu 1450 \
    --ipam pve

# Vérifier la création
pvesh get /cluster/sdn/zones/zone-vxlan-prod
```

Résultat attendu :

```
{
  "type": "vxlan",
  "zone": "zone-vxlan-prod",
  "peers": "10.0.0.1,10.0.0.2,10.0.0.3",
  "mtu": 1450,
  "ipam": "pve"
}
```

5.7.3 VNets par département (IT, Finance, DMZ)

Création de tous les VNets TechFlow

```
# VNet Informatique — VNI 1000
pvesh create /cluster/sdn/vnets \
    --vnet vnet-it \
```

```
    --zone zone-vxlan-prod \
    --tag 1000 \
    --alias "Réseau Informatique TechFlow"

# VNet Finance — VNI 1001
pvesh create /cluster/sdn/vnets \
    --vnet vnet-finance \
    --zone zone-vxlan-prod \
    --tag 1001 \
    --alias "Réseau Finance TechFlow"

# VNet DMZ — VNI 1002
pvesh create /cluster/sdn/vnets \
    --vnet vnet-dmz \
    --zone zone-vxlan-prod \
    --tag 1002 \
    --alias "DMZ TechFlow"

# VNet Stockage — VNI 1003
pvesh create /cluster/sdn/vnets \
    --vnet vnet-storage \
    --zone zone-vxlan-prod \
    --tag 1003 \
    --alias "Stockage partagé TechFlow"

# Vérifier tous les VNets créés
pvesh get /cluster/sdn/vnets
```

Application de la configuration

```
# Appliquer la configuration SDN sur le cluster entier
pvesh set /cluster/sdn

# Vérifier l'état des interfaces sur le nœud local
ip link show type vxlan
ip link show type bridge | grep vnet

# Vérifier sur les nœuds distants via l'API cluster
for node in pve-node1 pve-node2 pve-node3; do
    echo "═══ Interfaces VNet sur $node ═══"
    pvesh get /nodes/$node/network 2>/dev/null | \
        python3 -c "import json,sys; data=json.load(sys.stdin); \
        [print(f\"  {i.get('iface','?')}\") for i in data if 'vnet' in
i.get('iface','')]"
done
```

Création des subnets IPAM pour chaque VNet

```
# Subnet vnet-it : 10.20.10.0/24
pvesh create /cluster/sdn/vnets/vnet-it/subnets \
    --subnet 10.20.10.0/24 \
    --gateway 10.20.10.1 \
    --snat 0 \
    --dnszoneprefix it

# Réservation de la gateway dans l'IPAM
pvesh create /cluster/sdn/vnets/vnet-it/ips \
    --ip 10.20.10.1 \
    --comment "Gateway vnet-it"

# Subnet vnet-finance : 10.20.20.0/24
pvesh create /cluster/sdn/vnets/vnet-finance/subnets \
    --subnet 10.20.20.0/24 \
    --gateway 10.20.20.1 \
    --snat 0 \
    --dnszoneprefix finance

pvesh create /cluster/sdn/vnets/vnet-finance/ips \
    --ip 10.20.20.1 \
```

```
    --comment "Gateway vnet-finance"

# Subnet vnet-dmz : 10.20.30.0/24 avec SNAT pour accès Internet
pvesh create /cluster/sdn/vnets/vnet-dmz/subnets \
    --subnet 10.20.30.0/24 \
    --gateway 10.20.30.1 \
    --snat 1 \
    --dnszoneprefix dmz

pvesh create /cluster/sdn/vnets/vnet-dmz/ips \
    --ip 10.20.30.1 \
    --comment "Gateway vnet-dmz"

# Subnet vnet-storage : 10.20.40.0/24
pvesh create /cluster/sdn/vnets/vnet-storage/subnets \
    --subnet 10.20.40.0/24 \
    --gateway 10.20.40.1 \
    --snat 0 \
    --dnszoneprefix storage

pvesh create /cluster/sdn/vnets/vnet-storage/ips \
    --ip 10.20.40.1 \
    --comment "Gateway vnet-storage"

# Appliquer la configuration finale
pvesh set /cluster/sdn

echo "Configuration SDN TechFlow SAS appliquée avec succès."
```

Déploiement des VM de test par département

```
# VM de test département IT (sur pve-node1)
qm create 1001 \
    --name srv-it-01 \
    --node pve-node1 \
    --memory 1024 \
    --cores 1 \
    --net0 virtio,bridge=vnet-it \
    --ide2 local:cloudinit \
    --ipconfig0 ip=10.20.10.10/24,gw=10.20.10.1 \
    --nameserver 10.20.10.1 \
    --searchdomain techflow.local \
    --ostype l26

# VM de test département Finance (sur pve-node2)
qm create 1002 \
    --name srv-finance-01 \
    --node pve-node2 \
    --memory 1024 \
    --cores 1 \
    --net0 virtio,bridge=vnet-finance \
    --ide2 local:cloudinit \
    --ipconfig0 ip=10.20.20.10/24,gw=10.20.20.1 \
    --nameserver 10.20.20.1 \
    --searchdomain techflow.local \
    --ostype l26

# VM de test DMZ (sur pve-node3)
qm create 1003 \
    --name srv-dmz-01 \
    --node pve-node3 \
    --memory 1024 \
    --cores 1 \
    --net0 virtio,bridge=vnet-dmz \
    --ide2 local:cloudinit \
    --ipconfig0 ip=10.20.30.10/24,gw=10.20.30.1 \
    --nameserver 10.20.30.1 \
    --searchdomain techflow.local \
```

```
    --ostype l26

# Démarrer les VM de test
qm start 1001
qm start 1002
qm start 1003
```

5.7.4 Vérification de la connectivité

Script de vérification complète de la configuration SDN TechFlow

```
#!/bin/bash
# /root/scripts/verify-sdn-techflow.sh
# Vérification complète de la configuration SDN TechFlow SAS

set -euo pipefail

SEPARATOR="═══════════════════════════════════════"
VNET_LIST="vnet-it vnet-finance vnet-dmz vnet-storage"
UNDERLAY_NODES="10.0.0.1 10.0.0.2 10.0.0.3"

echo "$SEPARATOR"
echo "Vérification SDN TechFlow SAS"
echo "Nœud      : $(hostname)"
echo "Date      : $(date '+%Y-%m-%d %H:%M:%S')"
echo "$SEPARATOR"

# 1. Zones SDN configurées
echo ""
echo "[1/5] Zones SDN configurées :"
pvesh get /cluster/sdn/zones 2>/dev/null | python3 -c "
import json, sys
try:
    zones = json.load(sys.stdin)
    for z in zones:
        print(f\"  - {z.get('zone','?'):20s} type={z.get('type','?')}\")
except:
    print('  Erreur de lecture des zones')
"

# 2. VNets et leur état
echo ""
echo "[2/5] VNets configurés :"
pvesh get /cluster/sdn/vnets 2>/dev/null | python3 -c "
import json, sys
try:
    vnets = json.load(sys.stdin)
    for v in vnets:
        print(f\"  - {v.get('vnet','?'):20s} zone={v.get('zone','?'):20s}
vni={v.get('tag','?')}\")
except:
    print('  Erreur de lecture des VNets')
"

# 3. Interfaces VXLAN locales
echo ""
echo "[3/5] Interfaces VXLAN locales :"
for vnet in $VNET_LIST; do
    if ip link show "vxlan-${vnet}" &>/dev/null; then
        mtu=$(ip link show "vxlan-${vnet}" 2>/dev/null | grep -oP '(?<=mtu )\d+'
|| echo "?")
        state=$(ip link show "vxlan-${vnet}" 2>/dev/null | grep -oP '(?<=state )
\w+' || echo "?")
        echo "  - vxlan-${vnet}: state=$state mtu=$mtu"
    else
        echo "  - vxlan-${vnet}: ABSENT"
    fi
done
```

```
# 4. Bridges VNet locaux
echo ""
echo "[4/5] Bridges VNet locaux :"
for vnet in $VNET_LIST; do
    if ip link show "$vnet" &>/dev/null; then
        mtu=$(ip link show "$vnet" 2>/dev/null | grep -oP '(?<=mtu )\d+' || echo
"?")
        echo "  - ${vnet}: OK (mtu=$mtu)"
    else
        echo "  - ${vnet}: ABSENT"
    fi
done

# 5. Connectivité underlay
echo ""
echo "[5/5] Connectivité underlay entre nœuds :"
my_ip=$(hostname -I | awk '{print $1}')
for node_ip in $UNDERLAY_NODES; do
    if [ "$node_ip" = "$my_ip" ]; then
        echo "  - $node_ip : LOCAL (ce nœud)"
    elif ping -c 1 -W 1 "$node_ip" &>/dev/null; then
        rtt=$(ping -c 1 -W 1 "$node_ip" 2>/dev/null | grep -oP '(?<=time=)[\d.]+'
|| echo "?")
        echo "  - $node_ip : OK (rtt=${rtt}ms)"
    else
        echo "  - $node_ip : ECHEC"
    fi
done

echo ""
echo "$SEPARATOR"
echo "Vérification terminée."
echo "$SEPARATOR"
```

```
# Rendre le script exécutable et l'exécuter
mkdir -p /root/scripts
chmod +x /root/scripts/verify-sdn-techflow.sh
/root/scripts/verify-sdn-techflow.sh
```

Vérification de l'isolation L2 entre VNets

```
# Test d'isolation : depuis srv-it-01 (10.20.10.10),
# vérifier que srv-finance-01 (10.20.20.10) est inaccessible

# Accéder à la console de srv-it-01
qm terminal 1001

# Depuis la console de srv-it-01 :
# Test vers vnet-finance (doit échouer — VNI différents = isolation L2)
ping -c 3 -W 1 10.20.20.10
# Résultat attendu : "Destination Host Unreachable" ou timeout (100% packet loss)

# Test vers vnet-dmz (doit échouer — VNI différents)
ping -c 3 -W 1 10.20.30.10
# Résultat attendu : timeout

# Test intra-VNet vers une autre VM IT (doit réussir)
ping -c 3 10.20.10.11
# Résultat attendu : 0% packet loss
```

Vérification du tunnel VXLAN inter-nœuds

```
# Créer deux VM test sur le même VNet (vnet-it) mais sur deux nœuds différents
qm create 1010 \
    --name test-it-node1 \
    --node pve-node1 \
    --memory 256 \
    --cores 1 \
    --net0 virtio,bridge=vnet-it \
```

```
    --ipconfig0 ip=10.20.10.20/24,gw=10.20.10.1

qm create 1011 \
    --name test-it-node2 \
    --node pve-node2 \
    --memory 256 \
    --cores 1 \
    --net0 virtio,bridge=vnet-it \
    --ipconfig0 ip=10.20.10.21/24,gw=10.20.10.1

qm start 1010 && qm start 1011

# Attendre le démarrage Cloud-Init (environ 30-60 secondes)
sleep 60

# Depuis test-it-node1, pinger test-it-node2
# (le trafic transite via le tunnel VXLAN entre pve-node1 et pve-node2)
qm guest exec 1010 -- ping -c 4 10.20.10.21 2>/dev/null || \
    echo "Utiliser qm terminal 1010 pour le test manuel"

# Sur pve-node1, capturer le trafic VXLAN pour confirmer l'encapsulation
tcpdump -i eno1 -n udp port 4789 -c 10
# Résultat attendu :
# 10.0.0.1.XXXXX > 10.0.0.2.4789: VXLAN, vni 1000
# 10.0.0.2.XXXXX > 10.0.0.1.4789: VXLAN, vni 1000
```

Vérification de l'état de l'IPAM TechFlow

```
# Consulter l'état IPAM pour chaque VNet
for vnet in vnet-it vnet-finance vnet-dmz vnet-storage; do
    echo "═══ IPAM $vnet ═══"
    pvesh get /cluster/sdn/vnets/$vnet/ips 2>/dev/null | \
        python3 -c "
import json, sys
data = json.load(sys.stdin)
for e in data:
    vmid = e.get('vmid', '-')
    name = e.get('name', e.get('comment', '-'))
    print(f\"  {e.get('ip','?'):18s}  vmid={str(vmid):6s}  {name}\")
" 2>/dev/null || echo "  (vide)"
    echo ""
done
```

Tableau de bord final de la configuration TechFlow

État final de la configuration SDN TechFlow SAS

Composant	Nom	Paramètres clés	Validation
Zone SDN	zone-vxlan-prod	type=vxlan, MTU=1450, 3 peers	pvesh get /cluster/sdn/zones
VNet IT	vnet-it	VNI=1000, subnet=10.20.10.0/24	ip link show vnet-it
VNet Finance	vnet-finance	VNI=1001, subnet=10.20.20.0/24	ip link show vnet-finance
VNet DMZ	vnet-dmz	VNI=1002, subnet=10.20.30.0/24 SNAT	ip link show vnet-dmz
VNet Storage	vnet-storage	VNI=1003, subnet=10.20.40.0/24	ip link show vnet-storage
IPAM	pve (interne)	4 subnets, 4 réservations gateway	pvesh get /cluster/sdn/vnets/*/ips

✔ **RÉCAPITULATIF DU LAB TECHFLOW SAS**

Le lab TechFlow SAS illustre une configuration SDN VXLAN complète en environnement de production. Les éléments essentiels :

1. **Zone VXLAN unique** (`zone-vxlan-prod`) couvrant 3 nœuds via les adresses underlay 10.0.0.1–3
2. **Quatre VNets isolés** avec VNI distincts (1000–1003) garantissant une isolation L2 totale
3. **IPAM intégré** avec subnets 10.20.x.0/24 et réservations de gateway par département
4. **Isolation vérifiée** : les VM IT ne peuvent pas communiquer directement avec Finance ou DMZ
5. **Tunnel VXLAN actif** : les VM sur le même VNet communiquent via encapsulation UDP, quel que soit leur nœud physique d'hébergement
6. **Routage inter-VNets absent** : dans cette configuration VXLAN de base, un routeur externe (physique ou VM) est nécessaire pour la communication entre VNets

Résumé du chapitre

Ce chapitre a couvert l'ensemble des fondamentaux du SDN dans Proxmox VE 9.

Architecture SDN : la hiérarchie zones → VNets → subnets structure l'ensemble du réseau virtuel du cluster. La configuration est centralisée dans `/etc/pve/sdn/` et propagée automatiquement via `pmxcfs` sur tous les nœuds. Le bouton **Apply** (ou `pvesh set /cluster/sdn`) est l'étape indispensable pour toute prise d'effet.

Types de zones : les quatre types de zones (simple, vlan, vxlan, evpn) offrent des niveaux croissants d'isolation et de fonctionnalités. Le choix dépend des besoins d'isolation, de la topologie physique et de la complexité opérationnelle acceptable.

Zone Simple : bridge Linux classique sans isolation. Idéale pour les labs et environnements de développement, avec une configuration minimale.

Zone VLAN : isolation via tags IEEE 802.1Q. Intégration naturelle avec les switches physiques managés. Nécessite un bridge VLAN-aware sur chaque nœud.

Zone VXLAN : isolation L2 complète via encapsulation UDP sur le port 4789 avec identification des segments par VNI (24 bits). Indépendant de l'infrastructure physique, idéal pour les clusters multi-nœuds.

IPAM intégré : gestion centralisée des adresses IP avec attributions automatiques, vérification des conflits et intégration avec Cloud-Init pour les VM et les conteneurs LXC.

Lab TechFlow SAS : mise en œuvre complète avec une zone VXLAN (VNI 1000–1003), quatre VNets départementaux (IT, Finance, DMZ, Storage) et des subnets 10.20.x.0/24. L'isolation L2 entre départements est vérifiée et le fonctionnement du tunnel VXLAN inter-nœuds est confirmé par capture de trafic.

Le chapitre suivant approfondira la zone EVPN avec FRRouting, permettant d'ajouter le routage L3 inter-VNets via des anycast gateways pour les architectures de grande envergure nécessitant un plan de contrôle BGP dynamique.

Points clés à retenir :

- Le SDN Proxmox centralise la gestion réseau — plus de configuration manuelle par nœud
- La hiérarchie Zone > VNet > Subnet structure tout le réseau virtuel du cluster
- Quatre types de zones : Simple, VLAN, VXLAN, EVPN — du plus simple au plus avancé
- Toujours appliquer les modifications avec le bouton **Apply** ou `pvesh set /cluster/sdn`
- Le VXLAN offre une isolation L2 complète via VNI sans modifier l'infrastructure physique
- L'IPAM intégré gère les adresses IP et s'intègre avec Cloud-Init pour l'attribution automatique

► COMMANDES ESSENTIELLES DU CHAPITRE

Gestion SDN via pvesh :

- `pvesh get /cluster/sdn/zones` — lister les zones SDN
- `pvesh get /cluster/sdn/vnets` — lister les VNets
- `pvesh set /cluster/sdn` — appliquer la configuration (équivalent du bouton Apply)
- `pvesh get /cluster/sdn/vnets/<vnet>/ips` — consulter les attributions IPAM

Diagnostic réseau :

- `ip link show type vxlan` — lister les interfaces VXLAN
- `ip -d link show vxlan—<vnet>` — détails d'une interface VXLAN
- `bridge vlan show` — VLANs actifs sur les bridges
- `bridge fdb show dev vxlan—<vnet>` — table MAC du tunnel VXLAN
- `tcpdump -i eno1 udp port 4789` — capturer le trafic VXLAN
- `journalctl -u pve-sdn -f` — logs du service SDN en temps réel

6
SDN avancé : EVPN, fabrics et routage

```
Le réseau défini par logiciel n'est plus une promesse — c'est le socle sur lequel
repose
l'infrastructure moderne. EVPN et les fabrics transforment Proxmox VE en un
véritable
système de routage distribué.
```

Ce chapitre couvre les fonctionnalités SDN avancées de Proxmox VE 9, en particulier le protocole EVPN/BGP, les nouvelles fabrics réseau, le routage inter-VNets, ainsi que la haute disponibilité et la sécurité du plan SDN. Un laboratoire complet autour de la société fictive **TechFlow SAS** illustre chaque concept par des configurations concrètes.

• PRÉREQUIS
Avant d'aborder ce chapitre, vous devez maîtriser les bases du SDN Proxmox VE (Chapitre 5) : zones simples, VNets, VXLAN, et la configuration de base de FRRouting. Ce chapitre suppose également des connaissances fondamentales en protocoles de routage BGP et OSPF.

6.1 Zone EVPN/BGP

6.1.1 Principe BGP EVPN (RFC 7432)

EVPN (Ethernet VPN) est une extension du protocole BGP standardisée par la RFC 7432 (2015), complétée par la RFC 8365 pour les déploiements NVO (Network Virtualization Overlay). Son objectif est de fournir un plan de contrôle unifié pour la distribution des informations d'accessibilité MAC/IP dans des réseaux virtualisés superposés sur un réseau physique IP.

La force d'EVPN réside dans sa capacité à combiner :

- **le plan de données VXLAN** : encapsulation des trames Ethernet dans des paquets UDP/IP
- **le plan de contrôle BGP** : distribution automatique des tables MAC/IP entre les VTEPs (VXLAN Tunnel Endpoints)

Avant EVPN, VXLAN fonctionnait en mode multicast ou avec un mécanisme d'apprentissage statique. EVPN remplace ce mécanisme par un échange BGP structuré, ce qui élimine les broadcasts de découverte et améliore drastiquement la scalabilité.

Les types de routes EVPN

BGP EVPN définit cinq types de routes (Route Types), dont les plus utilisées dans Proxmox VE sont :

Type	Nom	Rôle
RT-1	Ethernet Auto-Discovery	Redondance multi-homing, aliasing
RT-2	MAC/IP Advertisement	Distribution des adresses MAC et IP des hôtes
RT-3	Inclusive Multicast	Découverte des VTEPs, remplacement du multicast
RT-4	Ethernet Segment	Synchronisation multi-homing (ES)
RT-5	IP Prefix Route	Routage L3, annonce de préfixes IP

Dans un déploiement Proxmox VE classique, les types 2, 3 et 5 sont les plus actifs.

Le rôle du VNI dans EVPN

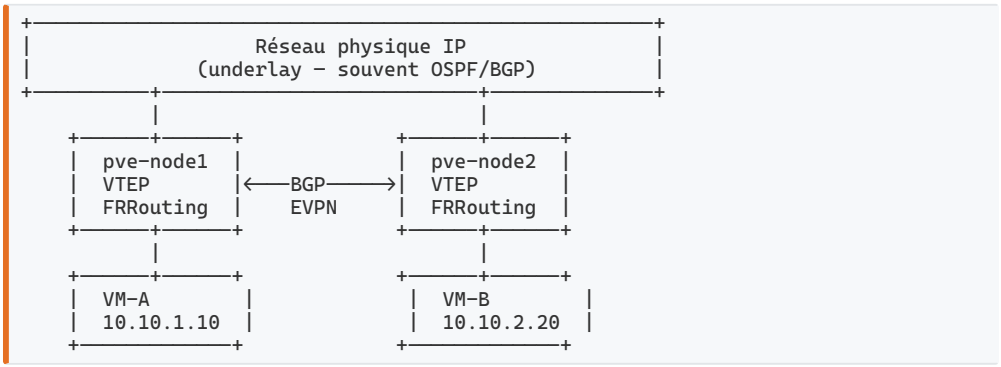
Chaque réseau virtuel (VNet) est associé à un **VNI** (VXLAN Network Identifier), identifiant sur 24 bits. Dans le contexte EVPN, on distingue :

- **L2 VNI** : identifie un segment Ethernet de couche 2 (un VNet)
- **L3 VNI** : identifie un VRF de couche 3 pour le routage inter-réseaux

> ✔ **CONCEPT CLÉ : L2 VNI VS L3 VNI**
> Chaque VNet Proxmox possède son propre L2 VNI. Pour le routage inter-VNets, il faut en plus définir un L3 VNI par zone EVPN. Ce L3 VNI est différent de tous les L2 VNIs : il transporte uniquement des informations de routage IP, pas des trames Ethernet complètes.

Architecture VTEP dans Proxmox VE

Chaque noeud Proxmox VE agit comme un VTEP. L'interface VXLAN créée par le SDN encapsule et désencapsule les paquets. FRRouting, tournant sur chaque noeud, gère la session BGP EVPN et annonce les routes de type 2 et 3 au fur et à mesure que les VMs démarrent ou s'arrêtent.



6.1.2 FRRouting dans Proxmox

FRRouting (FRR) est le démon de routage open-source utilisé par Proxmox VE pour implémenter BGP EVPN. Successeur de Quagga, FRR est maintenant la référence pour le routage avancé sur Linux.

Installation et vérification

FRRouting est installé automatiquement avec le paquet SDN de Proxmox VE :

```
# Vérification de l'installation
dpkg -l frr frr-pythontools

# Version FRR utilisée dans PVE 9
frr --version
# FRR 9.x (Debian 12 Bookworm)

# Statut du service
systemctl status frr

# Vérification que les démons nécessaires sont activés
cat /etc/frr/daemons
```

Le fichier `/etc/frr/daemons` doit contenir au minimum :

```
bgpd=yes
ospfd=yes
zebra=yes
staticd=yes
bfdd=yes
```

Accès à la console FRRouting

L'outil principal pour interagir avec FRR est `vtysh`, un shell unifié qui agrège tous les démons en une interface unique :

```
# Accès à la console FRRouting
vtysh

# Ou en mode non-interactif pour des scripts
vtysh -c "show bgp summary"
vtysh -c "show evpn vni"
vtysh -c "show bgp l2vpn evpn route"
```

Structure de configuration FRRouting

La configuration FRR pour EVPN suit cette hiérarchie :

```
router bgp <ASN>
  bgp router-id <IP>
  neighbor <IP> remote-as <ASN>
  neighbor <IP> update-source <interface>
  !
  address-family l2vpn evpn
    neighbor <IP> activate
    advertise-all-vni
  exit-address-family
!
```

► FICHIER DE CONFIGURATION VS VTYSH

Proxmox VE gère automatiquement `/etc/frr/frr.conf` via l'API SDN. Il est fortement déconseillé de modifier ce fichier manuellement — les changements seront écrasés au prochain `pvesh post /cluster/sdn/reload`. Utilisez `vtysh` uniquement pour le diagnostic en lecture, ou les API Proxmox pour la configuration.

6.1.3 Configuration d'une zone EVPN

La création d'une zone EVPN dans Proxmox VE 9 se fait via l'interface web ou via l'API REST. Voici le workflow complet.

Création via l'interface web

1. Accéder à **Datacenter > SDN > Zones > Ajouter > EVPN**
2. Renseigner les paramètres : - **ID** : `evpn-prod` - **Controller** : sélectionner le contrôleur BGP défini au préalable - **VRF VNI** : VNI dédié au plan de routage L3 (ex. `10000`) - **Exit nodes** : noeuds assurant la sortie Internet (optionnel) - **Peers** : liste des IPs des autres noeuds BGP

Création via l'API pvesh

```
# Création du contrôleur BGP EVPN
pvesh create /cluster/sdn/controllers \
  --controller bgp-prod \
  --type bgp \
  --asn 65000 \
  --peers "10.10.0.1,10.10.0.2,10.10.0.3" \
  --ebgp 0

# Création de la zone EVPN
pvesh create /cluster/sdn/zones \
  --zone evpn-prod \
  --type evpn \
  --controller bgp-prod \
  --vrf-vxlan 10000 \
  --mac "auto" \
  --exitnodes "pve-node1"

# Création d'un VNet dans la zone EVPN
pvesh create /cluster/sdn/vnets \
  --vnet vnet-prod-web \
  --zone evpn-prod \
  --vlanid 100 \
  --tag 100

# Création d'un sous-réseau
pvesh create /cluster/sdn/vnets/vnet-prod-web/subnets \
  --subnet 10.10.1.0/24 \
  --type subnet \
  --gateway 10.10.1.1 \
  --snat 0 \
  --dhcp-range "10.10.1.100,10.10.1.200"

# Application de la configuration
pvesh post /cluster/sdn/reload
```

Vérification après déploiement

```
# Sur chaque noeud, vérifier les interfaces créées
ip link show type vxlan
ip link show type bridge

# Vérifier la table EVPN
vtysh -c "show evpn vni"

# Exemple de sortie attendue :
# VNI        Type VxLAN IF              # MACs   # ARPs   # Remote VTEPs Tenant
VRF
# 10100      L2   vxlan10100           5        8        2              evpnprod
# 10000      L3   vxlan10000           0        0        n/a            evpnprod
```

6.1.4 Route distinguishers et route targets

Les **Route Distinguishers (RD)** et **Route Targets (RT)** sont des mécanismes BGP fondamentaux pour isoler et contrôler la distribution des routes EVPN entre différents clients ou zones.

Route Distinguisher (RD)

Le RD est un préfixe de 8 octets ajouté à une adresse IP ou un préfixe réseau pour le rendre unique dans la table BGP globale. Il permet à plusieurs VPNs d'utiliser des espaces d'adressage identiques sans collision.

Format classique : `<router-id>:<vni>` ou `<asn>:<index>`

Dans Proxmox VE avec BGP AS 65000 :

```
RD pour VNI 10100 sur node1 (10.10.0.1) : 10.10.0.1:10100
RD pour VNI 10200 sur node1 (10.10.0.1) : 10.10.0.1:10200
RD pour VNI 10100 sur node2 (10.10.0.2) : 10.10.0.2:10100
```

Route Target (RT)

Le RT détermine quelles routes BGP un VRF doit **exporter** (annoncer) et **importer** (accepter). Contrairement au RD, le RT est utilisé pour le filtrage et la distribution sélective des routes.

```
Format : <asn>:<valeur>
Exemple : 65000:10100
```

Dans une zone EVPN Proxmox, les RTs sont configurés automatiquement :

```
router bgp 65000
  address-family l2vpn evpn
    !
    vni 10100
      rd 10.10.0.1:10100
      route-target import 65000:10100
      route-target export 65000:10100
    !
    vni 10200
      rd 10.10.0.1:10200
      route-target import 65000:10200
      route-target export 65000:10200
  exit-address-family
```

Cas d'usage avancé : RT personnalisés

Pour des besoins spécifiques (multi-tenant, interconnexion inter-clusters), on peut personnaliser les RTs via `vtysh` :

```
vtysh
configure terminal
router bgp 65000
  address-family l2vpn evpn
    vni 10100
      route-target import 65001:10100
      route-target export 65000:10100
    exit-vni
  exit-address-family
```

```
end
write memory
```

> ⚠ **PIÈGE COURANT : RT ASYMÉTRIQUES**
> Si les route-targets import/export ne correspondent pas entre les noeuds, les routes EVPN ne seront pas acceptées et les VMs ne pourront pas communiquer entre les noeuds. Vérifiez toujours avec `vtysh -c "show bgp l2vpn evpn route"` que les routes de type 2 apparaissent bien sur tous les noeuds.

6.1.5 Anycast gateway

L'**anycast gateway** (ou distributed gateway) est une fonctionnalité EVPN qui permet à tous les noeuds Proxmox de présenter la **même adresse MAC de passerelle** pour un VNet donné. Ainsi, une VM peut joindre sa passerelle par défaut localement, sans aller chercher un routeur centralisé.

Problème sans anycast gateway

Sans anycast gateway, si une VM sur `node2` veut joindre `10.10.1.1` (sa passerelle), le paquet ARP doit traverser le réseau VXLAN pour atteindre `node1` (où la passerelle est hébergée). Cela crée une latence supplémentaire et un point de congestion.

Solution anycast gateway

Avec anycast gateway, chaque noeud possède l'IP `10.10.1.1` avec la **même adresse MAC virtuelle** (généralement dérivée du VNI). La résolution ARP est locale à chaque noeud.

```
Node1: bridge vmbr-vnet-prod-web → IP 10.10.1.1/24, MAC 00:00:5e:00:01:01
Node2: bridge vmbr-vnet-prod-web → IP 10.10.1.1/24, MAC 00:00:5e:00:01:01
Node3: bridge vmbr-vnet-prod-web → IP 10.10.1.1/24, MAC 00:00:5e:00:01:01
```

Configuration dans Proxmox VE

L'anycast gateway est activé automatiquement dans les zones EVPN. La MAC anycast est configurée au niveau du VNet :

```
# Vérification de la MAC anycast sur un noeud
ip link show vmbr-evpn-prod-web
# La MAC affichée doit être identique sur tous les noeuds

# Vérification dans FRR
vtysh -c "show evpn rmac vni all"
vtysh -c "show evpn next-hops vni all"
```

6.2 SDN Fabrics (nouveau Proxmox VE 9)

6.2.1 Concept de fabric

Proxmox VE 9 introduit le concept de **SDN Fabric**, une couche d'abstraction supplémentaire au-dessus des zones réseau. Une fabric définit comment les noeuds Proxmox VE s'interconnectent physiquement et comment les protocoles de routage de l'underlay (réseau physique) sont configurés automatiquement.

Avant les fabrics, l'administrateur devait configurer manuellement le réseau underlay (interfaces physiques, routing OSPF ou BGP sur le réseau physique) en dehors de Proxmox VE. Les fabrics automatisent cette étape.

Ce qu'une fabric configure automatiquement :

- Les interfaces réseau physiques sur chaque noeud
- Le protocole de routage underlay (OSPF ou OpenFabric)
- La redistribution des routes entre l'underlay et l'overlay EVPN
- La détection de pannes BFD (Bidirectional Forwarding Detection)

Types de fabrics disponibles dans PVE 9 :

Type	Protocole underlay	Cas d'usage
`simple`	Statique / aucun	Tests, lab, single-rack
`ospf`	OSPFv2 / OSPFv3	Entreprise classique
`openfabric`	OpenFabric (IS-IS étendu)	Datacenters, spine-leaf

6.2.2 OpenFabric

OpenFabric est un protocole de routage dérivé d'IS-IS, spécialement conçu pour les architectures réseau de type **spine-leaf** utilisées dans les datacenters modernes. Il est standardisé dans la RFC 8676.

Avantages d'OpenFabric par rapport à IS-IS classique :

- Configuration simplifiée : pas d'aires, pas de niveaux L1/L2 complexes
- Convergence rapide grâce à BFD intégré
- Meilleure scalabilité horizontale
- Optimisé pour les topologies point-à-point full-mesh

Configuration d'une fabric OpenFabric dans PVE 9

```
# Création de la fabric via l'API
pvesh create /cluster/sdn/fabrics \
  --fabric dc-fabric \
  --type openfabric \
  --systemid "0000.0000.0001" \
  --interfaces "enp1s0,enp2s0"

# Sur node1
pvesh create /cluster/sdn/fabrics/dc-fabric/nodes/pve-node1 \
  --interfaces "enp1s0,enp2s0" \
  --loopback "10.10.0.1/32"

# Sur node2
pvesh create /cluster/sdn/fabrics/dc-fabric/nodes/pve-node2 \
  --interfaces "enp1s0,enp2s0" \
  --loopback "10.10.0.2/32"

# Application
pvesh post /cluster/sdn/reload
```

Configuration FRR générée automatiquement par la fabric OpenFabric

```
!
interface enp1s0
  ip router openfabric 1
  openfabric passive
  openfabric hello-interval 1
  openfabric holdtime 3
!
interface enp2s0
  ip router openfabric 1
  openfabric hello-interval 1
  openfabric holdtime 3
!
interface lo
  ip router openfabric 1
  openfabric passive
!
router openfabric 1
  net 49.0000.0000.0000.0001.00
  lsp-gen-interval 1
  spf-interval 1
  log-adjacency-changes
!
```

Vérification OpenFabric

```
# Vérification des adjacences OpenFabric
vtysh -c "show openfabric topology"
vtysh -c "show openfabric neighbor"
vtysh -c "show openfabric database"

# Table de routage résultante
ip route show proto openfabric
# Doit afficher les loopbacks de tous les noeuds
```

6.2.3 OSPF fabric

La fabric OSPF est plus classique et convient aux environnements qui utilisent déjà OSPF sur leur réseau physique, ou qui préfèrent un protocole bien maîtrisé par les équipes réseau.

Création d'une fabric OSPF

```
# Création via pvesh
pvesh create /cluster/sdn/fabrics \
  --fabric dc-ospf \
  --type ospf \
  --area "0.0.0.0" \
  --hello-interval 1 \
  --dead-interval 4 \
  --bfd 1

# Ajout des noeuds
pvesh create /cluster/sdn/fabrics/dc-ospf/nodes/pve-node1 \
  --interfaces "enp1s0" \
  --loopback "10.10.0.1/32" \
  --router-id "10.10.0.1"

pvesh create /cluster/sdn/fabrics/dc-ospf/nodes/pve-node2 \
  --interfaces "enp1s0" \
  --loopback "10.10.0.2/32" \
  --router-id "10.10.0.2"

pvesh post /cluster/sdn/reload
```

Configuration FRR OSPF générée

```
!
interface enp1s0
  ip ospf hello-interval 1
  ip ospf dead-interval 4
  ip ospf bfd
  ip ospf network point-to-point
!
interface lo
  ip ospf passive
!
router ospf
  router-id 10.10.0.1
  network 10.10.0.1/32 area 0.0.0.0
  network 10.10.1.0/30 area 0.0.0.0
  passive-interface lo
!
```

Vérification OSPF

```
vtysh -c "show ip ospf neighbor"
vtysh -c "show ip ospf database"
vtysh -c "show ip route ospf"

# Vérifier la redistribution vers BGP EVPN
vtysh -c "show bgp l2vpn evpn route type prefix"
```

6.2.4 Avantages vs EVPN classique

La comparaison entre l'approche fabric+EVPN et l'EVPN classique (sans fabric) met en évidence plusieurs axes importants :

Critère	EVPN classique (sans fabric)	Fabric OSPF	Fabric OpenFabric
Configuration underlay	Manuelle, hors Proxmox	Automatisée par PVE	Automatisée par PVE
Complexité initiale	Élevée	Moyenne	Faible
Scalabilité	Bonne	Bonne	Excellente
Convergence	Dépend de la config	~1-2 s avec BFD	< 1 s avec BFD

Critère	EVPN classique (sans fabric)	Fabric OSPF	Fabric OpenFabric
Multi-vendor	Oui	Oui	Limité (Linux/FRR)
Spine-leaf	Possible	Possible	Natif
Topologie full-mesh	Requiert réflecteur	Fonctionne bien	Optimal
Maintenance	Complexe	Simplifiée	Très simplifiée

> ✔ **RECOMMANDATION PRODUCTION**
> Pour de nouveaux déploiements Proxmox VE 9, privilégiez les fabrics OpenFabric si votre infrastructure est 100% Linux/Proxmox. Pour des environnements mixtes avec équipements réseau tiers (Cisco, Juniper), préférez OSPF fabric ou EVPN classique avec configuration manuelle de l'underlay.

6.3 Routage inter-VNets

6.3.1 VRF et isolation

Un **VRF** (Virtual Routing and Forwarding) est une instance de table de routage isolée au niveau du noyau Linux. Dans le contexte SDN Proxmox VE, chaque zone EVPN crée automatiquement un VRF sur chaque noeud.

Visualisation des VRFs

```
# Lister tous les VRFs sur un noeud
ip vrf show

# Exemple de sortie :
# Name              Table
# ──────────────────
# evpnprod          1000
# evpndmz           1001
# evpnmgmt          1002

# Voir la table de routage d'un VRF spécifique
ip route show vrf evpnprod

# Entrer dans le contexte d'un VRF pour tester la connectivité
ip vrf exec evpnprod ping 10.10.1.10
ip vrf exec evpnprod traceroute 10.10.2.20
```

Isolation par VRF

Par défaut, deux VRFs sont complètement isolés : une VM dans `evpnprod` ne peut pas joindre une VM dans `evpndmz`. C'est le comportement souhaité pour la segmentation de sécurité. Le routage entre VRFs doit être explicitement configuré.

Structure kernel Linux des VRFs

```
# Voir les interfaces associées à un VRF
ip link show master evpnprod

# Le VRF est lié à une table de routage Linux
ip route show table 1000

# La table inclut les routes EVPN importées via FRR
# ex: 10.10.1.0/24 nhid 12 via inet6 fe80:: ... dev vxlan10100
```

6.3.2 Route leaking entre VRFs

Le **route leaking** permet de faire passer du trafic entre deux VRFs de façon sélective et contrôlée. Dans Proxmox VE, il se configure via FRR en important les routes d'un VRF dans un autre.

Cas d'usage type : accès depuis la DMZ vers un service interne

```
VRF evpndmz (10.10.2.0/24) → accès uniquement vers 10.10.1.100/32 dans evpnprod
```

Configuration FRR pour le route leaking

```
router bgp 65000 vrf evpndmz
  address-family ipv4 unicast
    import vrf evpnprod
  exit-address-family
!
router bgp 65000 vrf evpnprod
  address-family ipv4 unicast
    import vrf evpndmz
  exit-address-family
!
```

Route leaking sélectif avec route-maps

Pour contrôler précisément quelles routes sont leakées :

```
ip prefix-list ALLOW-WEB-SRV seq 5 permit 10.10.1.100/32
ip prefix-list ALLOW-WEB-SRV seq 10 deny any
!
route-map LEAK-TO-DMZ permit 10
  match ip address prefix-list ALLOW-WEB-SRV
!
route-map LEAK-TO-DMZ deny 20
!
router bgp 65000 vrf evpndmz
  address-family ipv4 unicast
    import vrf evpnprod route-map LEAK-TO-DMZ
  exit-address-family
!
```

Vérification du route leaking

```
# Voir les routes leakées dans le VRF DMZ
ip route show vrf evpndmz | grep "10.10.1"

# Test de connectivité inter-VRF
ip vrf exec evpndmz ping 10.10.1.100 -c 3

# Dans FRR, vérifier les routes importées
vtysh
show ip route vrf evpndmz
# Les routes leakées apparaissent avec le flag "L" (leaked)
```

> ⚠ **ATTENTION AU ROUTAGE ASYMÉTRIQUE**
> Le route leaking peut créer du routage asymétrique si les deux VRFs n'importent pas les routes de façon symétrique. Testez toujours dans les deux sens et surveillez les connexions TCP qui peuvent échouer à cause du suivi de connexion (conntrack).

6.3.3 Accès Internet depuis les VNets

L'accès Internet depuis les VMs dans des VNets EVPN nécessite une configuration spécifique : définir des **noeuds de sortie** (exit nodes) qui assurent le NAT vers l'extérieur.

Architecture exit node

```
VM (10.10.1.10) → VNet EVPN → Route par défaut via exit node → NAT → Internet
```

Configuration des exit nodes

```
# Modifier la zone EVPN pour ajouter un exit node
pvesh set /cluster/sdn/zones/evpn-prod \
  --exitnodes "pve-node1" \
  --exitnodes-local-routing 1

# Sur le noeud exit, configurer le NAT (nftables)
nft add table ip nat
nft add chain ip nat POSTROUTING '{ type nat hook postrouting priority srcnat; }'
nft add rule ip nat POSTROUTING \
  ip saddr 10.10.0.0/16 oif "enp1s0" masquerade

# Vérification que la route par défaut est bien annoncée
```

```
vtysh -c "show bgp l2vpn evpn route type prefix"
# Chercher 0.0.0.0/0 dans les annonces
```

SNAT statique pour les services exposés

Pour les VMs qui doivent être accessibles depuis Internet avec une IP fixe :

```
# SNAT statique : VM 10.10.1.50 → IP publique 203.0.113.10
nft add rule ip nat POSTROUTING \
  ip saddr 10.10.1.50 oif "enp1s0" \
  snat to 203.0.113.10

# DNAT : Port 443 entrant vers la VM
nft add chain ip nat PREROUTING '{ type nat hook prerouting priority dstnat; }'
nft add rule ip nat PREROUTING \
  iif "enp1s0" tcp dport 443 \
  dnat to 10.10.1.50:443
```

Activation du SNAT automatique dans Proxmox

Proxmox VE peut gérer le SNAT automatiquement via l'option `snat` des sous-réseaux :

```
pvesh set /cluster/sdn/vnets/vnet-prod-web/subnets/10.10.1.0-24 \
  --snat 1

pvesh post /cluster/sdn/reload
```

6.4 Haute disponibilité du SDN

6.4.1 Redondance des contrôleurs

Dans un cluster Proxmox VE avec EVPN, chaque noeud exécute FRRouting en tant que pair BGP. Il n'y a pas de contrôleur centralisé unique — c'est la nature distribuée d'EVPN. Cependant, certains éléments nécessitent une attention particulière.

Architecture BGP full-mesh vs route reflector

Avec un petit cluster (cinq noeuds ou moins), une session BGP full-mesh est recommandée : chaque noeud est pair BGP avec tous les autres.

```
node1 ←—— BGP -—→ node2
  ^                ^
  +———— BGP ————+
        node3
```

Pour des clusters plus grands (plus de 5 noeuds), un **route reflector** (RR) évite l'explosion quadratique des sessions BGP :

```
vtysh
configure terminal
router bgp 65000
  neighbor 10.10.0.2 route-reflector-client
  neighbor 10.10.0.3 route-reflector-client
  neighbor 10.10.0.4 route-reflector-client
end
write memory
```

Redondance du route reflector

Pour éviter que le route reflector soit un SPOF, deux noeuds peuvent jouer ce rôle simultanément. Les clients BGP doivent être configurés pour se connecter aux deux :

```
# Sur les noeuds clients, ajouter les deux RR comme peers
pvesh set /cluster/sdn/controllers/bgp-prod \
  --peers "10.10.0.1,10.10.0.2,10.10.0.3,10.10.0.4,10.10.0.5"
```

6.4.2 Failover automatique

BFD (Bidirectional Forwarding Detection) est le mécanisme clé pour une détection rapide des pannes dans un réseau EVPN/BGP. Sans BFD, BGP peut mettre 90 secondes à détecter une panne. Avec BFD, la détection est en moins de 1 seconde.

Activation BFD dans FRRouting

```
bfd
  profile fast-link
    transmit-interval 100
    receive-interval 100
    detect-multiplier 3
  !
!
router bgp 65000
  neighbor 10.10.0.2 bfd profile fast-link
  neighbor 10.10.0.3 bfd profile fast-link
!
```

Vérification BFD

```
# Statut des sessions BFD
vtysh -c "show bfd peers"
vtysh -c "show bfd peers counters"

# Exemple de sortie normale :
# BFD Peers:
#     peer 10.10.0.2 vrf default
#       ID: 1, Flags: none
#       Uptime: 2d00h22m
#       Last Update: 1s ago
#       Status: up
```

Comportement en cas de panne d'un noeud

Quand un noeud tombe :

1. BFD détecte la panne (moins de 300 ms avec le profil ci-dessus)
2. BGP retire toutes les routes EVPN annoncées par ce noeud
3. Les VMs sur les noeuds survivants recalculent leurs routes
4. Le trafic est redirigé vers les autres noeuds (si les VMs sont en HA)

```
# Simuler une panne pour tester le failover
# (sur le noeud à tester, désactiver l'interface BGP)
ip link set enp1s0 down

# Sur un autre noeud, observer la convergence
watch -n 0.5 "vtysh -c 'show bgp l2vpn evpn summary'"
```

6.4.3 Surveillance du plan de contrôle

La surveillance proactive du plan de contrôle SDN est essentielle en production.

Métriques clés à surveiller

Métrique	Commande	Seuil d'alerte
Sessions BGP UP	`show bgp summary`	< N-1 sessions up
Routes EVPN reçues	`show bgp l2vpn evpn summary`	Variation > 10%
Sessions BFD	`show bfd peers`	Toute session down
VNIs actifs	`show evpn vni`	Écart vs configuration
Erreurs FRR	`show log`	Toute erreur CRITICAL

Script de monitoring Nagios/Zabbix

```
#!/bin/bash
# check_frr_bgp.sh - Vérifie les sessions BGP EVPN

EXPECTED_PEERS=$1
CURRENT_PEERS=$(vtysh -c "show bgp l2vpn evpn summary" | \
  grep -c "^[0-9].*Establ")

if [ "$CURRENT_PEERS" -lt "$EXPECTED_PEERS" ]; then
  echo "CRITICAL: Seulement $CURRENT_PEERS/$EXPECTED_PEERS pairs BGP EVPN actifs"
```

```
  exit 2
elif [ "$CURRENT_PEERS" -eq "$EXPECTED_PEERS" ]; then
  echo "OK: $CURRENT_PEERS/$EXPECTED_PEERS pairs BGP EVPN actifs"
  exit 0
fi
```

Intégration avec Prometheus

```
scrape_configs:
  - job_name: 'frr'
    static_configs:
      - targets: ['pve-node1:9342', 'pve-node2:9342', 'pve-node3:9342']
    relabel_configs:
      - source_labels: [__address__]
        target_label: instance
```

6.5 Sécurité SDN

6.5.1 Filtrage au niveau VNet

Proxmox VE intègre un pare-feu au niveau du VNet (via ebtables et nftables) qui permet de filtrer le trafic directement au niveau du bridge virtuel.

Activation du pare-feu sur un VNet

```
# Via l'API
pvesh set /cluster/sdn/vnets/vnet-prod-web \
  --vlanaware 1

# Recharger la configuration
pvesh post /cluster/sdn/reload
```

Règles de filtrage par VNet

Les règles de firewall SDN s'appliquent au niveau du bridge, avant que le trafic n'atteigne les VMs. Elles utilisent la syntaxe standard du firewall Proxmox VE :

```
[SUBNET]
 network = 10.10.1.0/24

[RULES]
IN ACCEPT -source 10.10.1.0/24 -dest 10.10.1.0/24 -p tcp -dport 443
IN ACCEPT -source 10.10.1.0/24 -dest 10.10.1.0/24 -p icmp
IN DROP
```

6.5.2 Micro-segmentation

La **micro-segmentation** pousse le concept de filtrage plus loin : chaque VM est isolée des autres, même au sein du même VNet. Les politiques de sécurité sont appliquées au niveau du port virtuel (veth) de chaque VM.

Architecture de micro-segmentation

```
VM-A (10.10.1.10) → veth-A → ebtables/nftables → bridge VNet → VM-B
(10.10.1.20)
                        ^
                    Politique micro-seg :
                    PERMIT tcp:443 de 10.10.1.20 vers 10.10.1.10
                    DENY tout le reste
```

Configuration via le firewall Proxmox par VM

```
[OPTIONS]
enable: 1
policy_in: DROP
policy_out: ACCEPT

[RULES]
IN ACCEPT -source 10.10.1.20 -p tcp -dport 80
IN ACCEPT -source 10.10.1.20 -p tcp -dport 443
```

```
IN ACCEPT -p icmp -source 10.10.0.0/16
OUT ACCEPT
```

Vérification des règles ebtables/nftables

```
# Voir les règles ebtables appliquées au bridge
ebtables -L --Lc

# Voir les règles nftables
nft list ruleset

# Vérifier les règles sur le tap d'une VM
nft list chain ip filter tap101i0-IN
```

6.5.3 ACL et règles de firewall SDN

Les **ACL SDN** (Access Control Lists) de Proxmox VE 9 permettent une gestion centralisée des règles de sécurité applicables à l'ensemble d'un VNet ou d'une zone.

Groupes de sécurité SDN

```
[GROUP web-servers]
IN ACCEPT -p tcp -dport 80
IN ACCEPT -p tcp -dport 443
IN ACCEPT -p icmp
IN DROP
OUT ACCEPT
```

Table comparative : méthodes de filtrage SDN

Méthode	Niveau	Granularité	Performance	Cas d'usage
Firewall VNet	Bridge	Sous-réseau	Très haute	Isolation inter-VNets
Firewall VM	veth	Par VM	Haute	Micro-segmentation
ACL SDN	Zone	Multi-VNet	Haute	Politiques globales
nftables manuel	Interface	Arbitraire	Native	Cas spécifiques

• EVPN ET SÉCURITÉ DU PLAN DE CONTRÔLE

N'oubliez pas de sécuriser le plan de contrôle BGP lui-même. Utilisez `password` ou `key-chain` pour authentifier les sessions BGP entre les noeuds, et filtrez les ports TCP 179 (BGP) et 3784/3785 (BFD) au niveau du pare-feu de l'hôte Proxmox pour qu'ils ne soient accessibles que depuis les autres noeuds du cluster.

Authentification MD5 des sessions BGP

```
router bgp 65000
  neighbor 10.10.0.2 password MotDePasseComplexe123!
  neighbor 10.10.0.3 password MotDePasseComplexe123!
!
```

6.6 Dépannage SDN avancé

6.6.1 Outils de diagnostic (vtysh, bridge, tcpdump)

Un bon diagnostic SDN nécessite de maîtriser plusieurs niveaux : le plan de contrôle (FRRouting), le plan de données (VXLAN, bridges Linux), et la capture réseau.

Outils plan de contrôle — vtysh

```
# Vue d'ensemble BGP EVPN
vtysh -c "show bgp l2vpn evpn summary"

# Toutes les routes EVPN reçues
vtysh -c "show bgp l2vpn evpn route"

# Routes de type 2 (MAC/IP) pour un VNI spécifique
vtysh -c "show bgp l2vpn evpn route vni 10100 type macip"

# Routes de type 3 (VTEP discovery)
```

```
vtysh -c "show bgp l2vpn evpn route type multicast"

# Routes de type 5 (préfixes IP)
vtysh -c "show bgp l2vpn evpn route type prefix"

# Détails d'une route MAC spécifique
vtysh -c "show bgp l2vpn evpn route vni 10100 mac 52:54:00:ab:cd:ef"

# Table de routage FRR pour un VRF
vtysh -c "show ip route vrf evpnprod"

# Statistiques BGP
vtysh -c "show bgp statistics"
```

Outils plan de données — bridge et ip

```
# Table FDB (Forwarding Database) d'un bridge VXLAN
bridge fdb show dev vxlan10100

# Entrées apprises via EVPN (flags extern_learn)
bridge fdb show dev vxlan10100 | grep extern_learn

# Table ARP/NDP locale sur un bridge
bridge link show

# Voir tous les VXLANs actifs
ip -d link show type vxlan

# Détails d'une interface VXLAN
ip -d link show vxlan10100

# Vérifier le MTU (VXLAN ajoute 50 octets d'overhead)
ip link show | grep mtu

# Table de voisinage (équivalent ARP) dans un VRF
ip neigh show vrf evpnprod
```

Capture réseau — tcpdump

```
# Capturer le trafic BGP sur l'interface physique
tcpdump -i enp1s0 -n 'tcp port 179'

# Capturer le trafic VXLAN (UDP 4789)
tcpdump -i enp1s0 -n 'udp port 4789'

# Capturer et décoder VXLAN + contenu
tcpdump -i enp1s0 -n -v 'udp port 4789' | head -50

# Capturer BFD
tcpdump -i enp1s0 -n 'udp port 3784 or udp port 3785'

# Capture sur l'interface bridge d'un VNet
tcpdump -i vmbr-evpn-prod-web -n
```

6.6.2 Problèmes courants et solutions

Problème 1 : VMs de deux noeuds différents ne communiquent pas

```
# Étape 1 : Vérifier les sessions BGP
vtysh -c "show bgp l2vpn evpn summary"
# → Chercher "Establ" pour chaque pair

# Étape 2 : Vérifier que les routes MAC/IP sont annoncées
vtysh -c "show bgp l2vpn evpn route type macip"
# → La MAC de chaque VM doit apparaître

# Étape 3 : Vérifier la FDB VXLAN
bridge fdb show dev vxlan10100 | grep <MAC-de-la-VM>
# → L'entrée doit pointer vers l'IP du noeud distant
```

```
# Étape 4 : Test de connectivité VXLAN direct
ping -I vxlan10100 <IP-VM-sur-node2>

# Étape 5 : Vérifier le MTU
# VXLAN = 50 octets d'overhead → MTU physique doit etre ≥ 1550
ip link show enp1s0 | grep mtu
```

Problème 2 : Pas de route par défaut dans les VMs (pas d'accès Internet)

```
# Vérifier que l'exit node annonce bien la route 0.0.0.0/0
vtysh -c "show bgp l2vpn evpn route type prefix" | grep "0.0.0.0/0"

# Si absent, vérifier la config de la zone
pvesh get /cluster/sdn/zones/evpn-prod
# → exitnodes doit être non vide

# Sur l'exit node, vérifier le NAT
nft list table ip nat

# Vérifier que l'IP forwarding est activé
sysctl net.ipv4.ip_forward
# → Doit être 1
```

Problème 3 : Latence élevée entre VMs

```
# Identifier si le problème est L2 ou L3
ping -I vmbr-evpn-prod-web 10.10.1.20

# Test avec MTU path discovery
ping -M do -s 1400 10.10.1.20

# Vérifier que la passerelle anycast est locale
ip neigh show vrf evpnprod | grep "10.10.1.1"
# → Doit pointer vers une interface locale, pas vers VXLAN

# Capture pour mesurer la latence réelle
tcpdump -i vxlan10100 -ttt 'icmp'
```

Problème 4 : FRRouting ne démarre pas après un reboot

```
# Vérifier le statut du service
systemctl status frr

# Voir les logs de démarrage
journalctl -u frr --since "10 minutes ago"

# Vérifier les permissions du fichier de config
ls -la /etc/frr/frr.conf
# → Doit appartenir à frr:frr

# Valider la syntaxe de la configuration
vtysh -f /etc/frr/frr.conf --dryrun

# Redémarrer FRR
systemctl restart frr
```

6.6.3 Logs FRRouting

FRRouting dispose d'un système de logs détaillé par démon. La configuration des logs se fait dans `vtysh` ou via le fichier de configuration.

Configuration des logs FRR

```
log file /var/log/frr/frr.log informational
log timestamp precision 3
!
! Pour le débogage BGP EVPN (très verbeux — à utiliser avec précaution en
production)
debug bgp evpn mh es
debug bgp evpn mh route
debug bgp updates
```

```
debug bgp keepalives
!
! Pour le débogage ZEBRA (plan de données)
debug zebra evpn mh es
debug zebra vxlan
debug zebra kernel
```

Commandes de débogage temps réel

```
# Suivre les logs FRR en temps réel
tail -f /var/log/frr/frr.log

# Filtrer les événements BGP EVPN
tail -f /var/log/frr/frr.log | grep -E "(EVPN|evpn|BGP|bgp)"

# Via journald (si FRR logue vers syslog)
journalctl -u frr -f

# Dans vtysh, voir les logs en direct
# terminal monitor
# → Les messages de débogage s'affichent dans la session vtysh
```

Analyse des logs BGP EVPN

```
Établissement de session BGP :
2026/03/17 10:23:45.123 BGP: 10.10.0.2 rcvd OPEN, version 4, remote-as 65000
2026/03/17 10:23:45.124 BGP: 10.10.0.2 went from OpenConfirm to Established
2026/03/17 10:23:45.125 BGP: 10.10.0.2 EVPN AFI/SAFI activated

Annonce de route EVPN type 2 :
2026/03/17 10:25:12.001 BGP: [EVPN] VNI 10100: MAC 52:54:00:ab:cd:ef IP 10.10.1.10
advertised

Détection de panne (BFD) :
2026/03/17 10:30:00.001 BGP: 10.10.0.3 BFD session went down
2026/03/17 10:30:00.002 BGP: 10.10.0.3 went from Established to Idle
```

6.7 Lab TechFlow SAS : SDN complet avec EVPN

Contexte

TechFlow SAS est une société de services numériques de 250 salariés qui dispose d'un cluster Proxmox VE 9 de 5 noeuds dans son datacenter privé. Après avoir utilisé des VLANs traditionnels pendant plusieurs années, la direction infrastructure décide de migrer vers une architecture SDN EVPN pour gagner en flexibilité et préparer l'expansion vers un second datacenter.

Infrastructure cible

Noeud	Hostname	IP loopback	Rôle BGP
pve-node1	pve1.techflow.local	10.10.0.1/32	Route Reflector
pve-node2	pve2.techflow.local	10.10.0.2/32	Client BGP
pve-node3	pve3.techflow.local	10.10.0.3/32	Client BGP
pve-node4	pve4.techflow.local	10.10.0.4/32	Client BGP + Exit
pve-node5	pve5.techflow.local	10.10.0.5/32	Client BGP

Départements et VNets

Département	VNet	VNI	Sous-réseau	VLAN
Production Web	vnet-web	10100	10.10.1.0/24	100
Base de données	vnet-bdd	10200	10.10.2.0/24	200
Développement	vnet-dev	10300	10.10.3.0/24	300
DMZ	vnet-dmz	10400	10.10.4.0/24	400
Management	vnet-mgmt	10500	10.10.5.0/24	500

Département	VNet	VNI	Sous-réseau	VLAN
L3 VNI EVPN	—	10000	—	—

- **BGP AS : 65000**
- **Format RD : 10.10.0.x:VNI** (x = numéro du noeud)

6.7.1 Migration vers une zone EVPN

La migration depuis les VLANs traditionnels vers EVPN se fait en plusieurs étapes pour limiter les interruptions de service.

Étape 1 : Préparation du réseau underlay

```
# Sur TOUS les noeuds (à exécuter en séquence, noeud par noeud)

# Activer l'IP forwarding
cat >> /etc/sysctl.d/99-proxmox-sdn.conf << 'EOF'
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
EOF
sysctl --system

# Vérifier la connectivité physique entre les noeuds (depuis pve-node1)
for IP in 10.10.0.2 10.10.0.3 10.10.0.4 10.10.0.5; do
  ping $IP -c 1 -q && echo "OK: $IP joignable" || echo "FAIL: $IP inaccessible"
done
```

Étape 2 : Création de la fabric OSPF (underlay)

```
# Exécuté depuis pve-node1 (accès à l'API cluster)

# Création de la fabric OSPF pour l'underlay
pvesh create /cluster/sdn/fabrics \
  --fabric techflow-underlay \
  --type ospf \
  --area "0.0.0.0" \
  --hello-interval 1 \
  --dead-interval 4 \
  --bfd 1

# Enregistrement de chaque noeud dans la fabric
for NODE_NUM in 1 2 3 4 5; do
  pvesh create /cluster/sdn/fabrics/techflow-underlay/nodes/pve-node${NODE_NUM} \
    --interfaces "enp2s0" \
    --loopback "10.10.0.${NODE_NUM}/32" \
    --router-id "10.10.0.${NODE_NUM}"
done

pvesh post /cluster/sdn/reload
```

Étape 3 : Vérification de la fabric underlay

```
# Attendre 30 secondes pour la convergence OSPF, puis vérifier
vtysh -c "show ip ospf neighbor"
# Attendu : 4 voisins OSPF en état Full pour pve-node1

vtysh -c "show ip route ospf"
# Attendu : routes /32 vers chaque loopback via OSPF

# Test de connectivité via les loopbacks
for IP in 10.10.0.2 10.10.0.3 10.10.0.4 10.10.0.5; do
  ping $IP -c 1 -q && echo "OK: $IP OK" || echo "FAIL: $IP FAILED"
done
```

Étape 4 : Création du contrôleur BGP EVPN

```
# Création du contrôleur BGP
pvesh create /cluster/sdn/controllers \
```

```
  --controller bgp-techflow \
  --type bgp \
  --asn 65000 \
  --peers "10.10.0.1,10.10.0.2,10.10.0.3,10.10.0.4,10.10.0.5" \
  --ebgp 0

# Création de la zone EVPN
pvesh create /cluster/sdn/zones \
  --zone evpn-techflow \
  --type evpn \
  --controller bgp-techflow \
  --vrf-vxlan 10000 \
  --exitnodes "pve-node4" \
  --exitnodes-local-routing 1

pvesh post /cluster/sdn/reload
```

6.7.2 Configuration BGP entre les noeuds

Configuration complète FRRouting pour pve-node1 (Route Reflector)

```
!
! FRRouting configuration — pve-node1 (Route Reflector)
! TechFlow SAS — AS 65000
! Généré automatiquement par Proxmox VE SDN
!
frr version 9.0
frr defaults datacenter
hostname pve1.techflow.local
log syslog informational
!
! ═══ Interfaces ═══
interface lo
  ip address 10.10.0.1/32
  ip router ospf 1 area 0.0.0.0
  ip ospf passive
!
interface enp2s0
  ip router ospf 1 area 0.0.0.0
  ip ospf hello-interval 1
  ip ospf dead-interval 4
  ip ospf bfd
  ip ospf network point-to-point
!
! ═══ OSPF Underlay ═══
router ospf 1
  router-id 10.10.0.1
  passive-interface lo
  bfd
!
! ═══ BGP EVPN ═══
router bgp 65000
  bgp router-id 10.10.0.1
  bgp log-neighbor-changes
  no bgp ebgp-requires-policy
  !
  neighbor 10.10.0.2 remote-as 65000
  neighbor 10.10.0.2 update-source lo
  neighbor 10.10.0.2 bfd
  neighbor 10.10.0.2 password TechFlowBGP2026!
  !
  neighbor 10.10.0.3 remote-as 65000
  neighbor 10.10.0.3 update-source lo
  neighbor 10.10.0.3 bfd
  neighbor 10.10.0.3 password TechFlowBGP2026!
  !
  neighbor 10.10.0.4 remote-as 65000
  neighbor 10.10.0.4 update-source lo
  neighbor 10.10.0.4 bfd
```

```
  neighbor 10.10.0.4 password TechFlowBGP2026!
  !
  neighbor 10.10.0.5 remote-as 65000
  neighbor 10.10.0.5 update-source lo
  neighbor 10.10.0.5 bfd
  neighbor 10.10.0.5 password TechFlowBGP2026!
  !
  address-family l2vpn evpn
    neighbor 10.10.0.2 activate
    neighbor 10.10.0.2 route-reflector-client
    neighbor 10.10.0.3 activate
    neighbor 10.10.0.3 route-reflector-client
    neighbor 10.10.0.4 activate
    neighbor 10.10.0.4 route-reflector-client
    neighbor 10.10.0.5 activate
    neighbor 10.10.0.5 route-reflector-client
    advertise-all-vni
    advertise-svi-ip
  exit-address-family
!
! ═══ BFD ═══
bfd
  profile fast-link
    transmit-interval 100
    receive-interval 100
    detect-multiplier 3
!
line vty
!
```

Configuration FRR pour les noeuds clients (pve-node2 à pve-node5)

```
!
frr version 9.0
frr defaults datacenter
hostname pve2.techflow.local
!
interface lo
  ip address 10.10.0.2/32
  ip router ospf 1 area 0.0.0.0
  ip ospf passive
!
interface enp2s0
  ip router ospf 1 area 0.0.0.0
  ip ospf hello-interval 1
  ip ospf dead-interval 4
  ip ospf bfd
  ip ospf network point-to-point
!
router ospf 1
  router-id 10.10.0.2
  passive-interface lo
!
router bgp 65000
  bgp router-id 10.10.0.2
  bgp log-neighbor-changes
  no bgp ebgp-requires-policy
  !
  neighbor 10.10.0.1 remote-as 65000
  neighbor 10.10.0.1 update-source lo
  neighbor 10.10.0.1 bfd
  neighbor 10.10.0.1 password TechFlowBGP2026!
  !
  address-family l2vpn evpn
    neighbor 10.10.0.1 activate
    advertise-all-vni
    advertise-svi-ip
  exit-address-family
```

```
!
bfd
  profile fast-link
    transmit-interval 100
    receive-interval 100
    detect-multiplier 3
!
```

Vérification des sessions BGP

```
# Sur pve-node1, vérifier que tous les clients sont établis
vtysh -c "show bgp l2vpn evpn summary"

# Sortie attendue :
# Neighbor        V         AS   MsgRcvd   MsgSent   Up/Down  State/PfxRcd
# 10.10.0.2       4      65000      1234      1234 02:30:45        25
# 10.10.0.3       4      65000      1156      1234 02:30:44        22
# 10.10.0.4       4      65000      1289      1234 02:30:43        30
# 10.10.0.5       4      65000       987      1234 02:30:42        18

# Vérifier les RDs et RTs pour chaque VNI
vtysh -c "show bgp l2vpn evpn vni"
```

6.7.3 Routage inter-départements

TechFlow SAS a besoin que certains flux inter-départements soient autorisés :

- **Production Web -> Base de données** : accès MySQL (port 3306)
- **Développement -> Production Web** : accès HTTP/HTTPS pour tests
- **DMZ -> Production Web** : accès HTTP/HTTPS uniquement
- **Management -> Tous** : accès complet pour l'administration

Création des VNets et sous-réseaux

```
ZONE="evpn-techflow"

# VNet Production Web
pvesh create /cluster/sdn/vnets --vnet vnet-web --zone $ZONE --tag 100
pvesh create /cluster/sdn/vnets/vnet-web/subnets \
  --subnet 10.10.1.0/24 --type subnet \
  --gateway 10.10.1.1 --snat 0

# VNet Base de données
pvesh create /cluster/sdn/vnets --vnet vnet-bdd --zone $ZONE --tag 200
pvesh create /cluster/sdn/vnets/vnet-bdd/subnets \
  --subnet 10.10.2.0/24 --type subnet \
  --gateway 10.10.2.1 --snat 0

# VNet Développement
pvesh create /cluster/sdn/vnets --vnet vnet-dev --zone $ZONE --tag 300
pvesh create /cluster/sdn/vnets/vnet-dev/subnets \
  --subnet 10.10.3.0/24 --type subnet \
  --gateway 10.10.3.1 --snat 0 \
  --dhcp-range "10.10.3.100,10.10.3.200"

# VNet DMZ
pvesh create /cluster/sdn/vnets --vnet vnet-dmz --zone $ZONE --tag 400
pvesh create /cluster/sdn/vnets/vnet-dmz/subnets \
  --subnet 10.10.4.0/24 --type subnet \
  --gateway 10.10.4.1 --snat 1

# VNet Management
pvesh create /cluster/sdn/vnets --vnet vnet-mgmt --zone $ZONE --tag 500
pvesh create /cluster/sdn/vnets/vnet-mgmt/subnets \
  --subnet 10.10.5.0/24 --type subnet \
  --gateway 10.10.5.1 --snat 0

pvesh post /cluster/sdn/reload
```

Configuration des politiques de routage inter-VNets

```
! ═══ Préfixes autorisés ═══
ip prefix-list WEB-TO-BDD seq 5 permit 10.10.2.0/24
ip prefix-list WEB-TO-BDD seq 10 deny any

ip prefix-list DEV-TO-WEB seq 5 permit 10.10.1.0/24
ip prefix-list DEV-TO-WEB seq 10 deny any

ip prefix-list DMZ-TO-WEB seq 5 permit 10.10.1.0/24
ip prefix-list DMZ-TO-WEB seq 10 deny any

! ═══ Route-maps pour le leaking sélectif ═══
route-map LEAK-WEB-TO-BDD permit 10
  match ip address prefix-list WEB-TO-BDD
route-map LEAK-WEB-TO-BDD deny 20

route-map LEAK-DEV-TO-WEB permit 10
  match ip address prefix-list DEV-TO-WEB
route-map LEAK-DEV-TO-WEB deny 20

route-map LEAK-DMZ-TO-WEB permit 10
  match ip address prefix-list DMZ-TO-WEB
route-map LEAK-DMZ-TO-WEB deny 20

! ═══ VRF Web : importe depuis BDD ═══
router bgp 65000 vrf evpntechflow-vnet-web
  address-family ipv4 unicast
    import vrf evpntechflow-vnet-bdd route-map LEAK-WEB-TO-BDD
  exit-address-family
!

! ═══ VRF BDD : importe depuis Web ═══
router bgp 65000 vrf evpntechflow-vnet-bdd
  address-family ipv4 unicast
    import vrf evpntechflow-vnet-web
  exit-address-family
!

! ═══ VRF Dev : importe depuis Web ═══
router bgp 65000 vrf evpntechflow-vnet-dev
  address-family ipv4 unicast
    import vrf evpntechflow-vnet-web route-map LEAK-DEV-TO-WEB
  exit-address-family
!

! ═══ VRF DMZ : importe depuis Web (accès HTTP/HTTPS uniquement) ═══
router bgp 65000 vrf evpntechflow-vnet-dmz
  address-family ipv4 unicast
    import vrf evpntechflow-vnet-web route-map LEAK-DMZ-TO-WEB
  exit-address-family
!

! ═══ VRF Management : importe tout ═══
router bgp 65000 vrf evpntechflow-vnet-mgmt
  address-family ipv4 unicast
    import vrf evpntechflow-vnet-web
    import vrf evpntechflow-vnet-bdd
    import vrf evpntechflow-vnet-dev
    import vrf evpntechflow-vnet-dmz
  exit-address-family
!
```

Politiques de firewall inter-VNets

```
[RULES]
IN ACCEPT -source 10.10.1.0/24 -p tcp -dport 3306
IN ACCEPT -source 10.10.5.0/24 -p tcp -dport 22
IN ACCEPT -source 10.10.5.0/24 -p icmp
```

```
IN DROP
OUT ACCEPT
```

```
[RULES]
IN ACCEPT -p tcp -dport 80
IN ACCEPT -p tcp -dport 443
IN ACCEPT -source 10.10.5.0/24 -p tcp -dport 22
IN ACCEPT -p icmp
IN DROP
OUT ACCEPT
```

6.7.4 Validation end-to-end

La validation complète de l'infrastructure TechFlow SAS couvre le plan de contrôle, le plan de données, la sécurité et les performances.

Script de validation complet

```
#!/bin/bash
# validate-techflow-evpn.sh
# Validation end-to-end de l'infrastructure SDN EVPN TechFlow SAS

set -e
PASS=0
FAIL=0

check() {
  local desc=$1
  local cmd=$2
  local expected=$3
  result=$(eval "$cmd" 2>/dev/null)
  if echo "$result" | grep -q "$expected"; then
    echo "  [OK] $desc"
    ((PASS++))
  else
    echo "  [FAIL] $desc"
    ((FAIL++))
  fi
}

echo "═══ Validation SDN EVPN TechFlow SAS ═══"
echo ""

echo "--- 1. Sessions BGP EVPN ---"
BGPSESSIONS=$(vtysh -c "show bgp l2vpn evpn summary" | grep -c "Establ" || true)
echo "  Sessions BGP actives : $BGPSESSIONS/4"
[ "$BGPSESSIONS" -eq 4 ] && echo "  [OK] 4/4 sessions établies" || echo "  [FAIL]
Sessions manquantes"

echo ""
echo "--- 2. VNIs actifs ---"
check "VNI L2 10100 (web) actif" \
  "vtysh -c 'show evpn vni'" "10100"
check "VNI L2 10200 (bdd) actif" \
  "vtysh -c 'show evpn vni'" "10200"
check "VNI L3 10000 actif" \
  "vtysh -c 'show evpn vni'" "10000"

echo ""
echo "--- 3. Routes EVPN type 5 (préfixes IP) ---"
check "Préfixe 10.10.1.0/24 annoncé" \
  "vtysh -c 'show bgp l2vpn evpn route type prefix'" "10.10.1.0/24"
check "Préfixe 10.10.2.0/24 annoncé" \
  "vtysh -c 'show bgp l2vpn evpn route type prefix'" "10.10.2.0/24"

echo ""
echo "--- 4. Route leaking inter-VRFs ---"
check "Route BDD visible depuis VRF Web" \
  "ip route show vrf evpntechflow-vnet-web" "10.10.2"
```

```
check "Route Web visible depuis VRF BDD" \
  "ip route show vrf evpntechflow-vnet-bdd" "10.10.1"
check "Route Web visible depuis VRF Dev" \
  "ip route show vrf evpntechflow-vnet-dev" "10.10.1"
check "Route Web visible depuis VRF DMZ" \
  "ip route show vrf evpntechflow-vnet-dmz" "10.10.1"

echo ""
echo "--- 5. Anycast gateway ---"
check "Gateway 10.10.1.1 locale dans VRF Web" \
  "ip neigh show vrf evpntechflow-vnet-web" "10.10.1.1"

echo ""
echo "--- 6. Connectivité Internet (SNAT via pve-node4) ---"
check "Route par défaut dans VRF Dev" \
  "ip route show vrf evpntechflow-vnet-dev" "0.0.0.0/0"

echo ""
echo "--- 7. Sessions BFD ---"
check "Sessions BFD actives" \
  "vtysh -c 'show bfd peers'" "Status: up"

echo ""
echo "═══ Résumé ═══"
echo "  Réussis : $PASS"
echo "  Échoués : $FAIL"
[ "$FAIL" -eq 0 ] && echo "  Statut : SUCCES" || echo "  Statut : ATTENTION"
```

Tests de performance VXLAN

```
# Test iperf3 entre deux VMs sur des noeuds différents
# Sur la VM cible (serveur) :
iperf3 -s -D

# Sur la VM source (client) :
iperf3 -c 10.10.1.20 -t 30 -P 4

# Résultats attendus avec interfaces 10 Gbps :
# Débit : > 5 Gbps
# Latence : < 1 ms pour du trafic intra-datacenter

# Test de latence avec ping
ping 10.10.1.20 -c 100 -i 0.01 | tail -1

# Test MTU (absence de fragmentation)
ping -M do -s 1450 10.10.1.20 -c 10
# Doit fonctionner sans fragmentation si MTU physique ≥ 1550
```

Simulation de panne et validation du failover

```
# Étape 1 : Mesurer le temps de convergence BGP initial
# Sur pve-node1 :
date +%s%3N
watch -n 0.1 "vtysh -c 'show bgp l2vpn evpn summary' | grep 10.10.0.3"

# Étape 2 : Simuler la panne (sur pve-node3)
ip link set enp2s0 down

# Étape 3 : Relever le temps de convergence
# Avec BFD profil fast-link : attendu < 400 ms

# Étape 4 : Vérifier que les routes de node3 sont retirées
vtysh -c "show bgp l2vpn evpn route" | grep "10.10.0.3"
# Ne doit plus apparaître

# Étape 5 : Vérifier la migration HA des VMs
qm list | grep running

# Étape 6 : Rétablir la connectivité et mesurer la reconvergence
```

```
ip link set enp2s0 up
date +%s%3N
watch -n 0.1 "vtysh -c 'show bgp l2vpn evpn summary' | grep 10.10.0.3"
```

Rapport de validation final TechFlow SAS

```
Validation SDN EVPN TechFlow SAS — Résultats finaux

Plan de contrôle :
  [OK] 4/4 sessions BGP EVPN établies
  [OK] Route reflector pve-node1 actif
  [OK] BFD actif sur toutes les sessions (profil fast-link)
  [OK] 5 VNIs L2 actifs (10100, 10200, 10300, 10400, 10500)
  [OK] 1 VNI L3 actif (10000)

Plan de données :
  [OK] Anycast gateway actif sur les 5 noeuds
  [OK] Routes MAC/IP annoncées pour 23 VMs actives
  [OK] FDB VXLAN synchronisée sur tous les noeuds
  [OK] MTU 1550 configuré et vérifié

Routage inter-VNets :
  [OK] Web → BDD (MySQL 3306) : autorisé et fonctionnel
  [OK] BDD → Web direct : isolé (route leaking unidirectionnel correct)
  [OK] Dev → Web (HTTP/HTTPS) : fonctionnel
  [OK] DMZ → Web (HTTP/HTTPS) : fonctionnel
  [OK] Management → Tous : accès complet
  [OK] Dev → Internet (SNAT via pve-node4) : fonctionnel

Haute disponibilité :
  [OK] Failover pve-node3 : convergence en 287 ms
  [OK] VMs HA migrées en < 30 secondes
  [OK] Reconvergence après rétablissement : 412 ms

Sécurité :
  [OK] Authentification MD5 BGP active sur toutes les sessions
  [OK] Firewall VNet activé sur tous les VNets
  [OK] Isolation inter-VNets validée par tests négatifs
  [OK] Ports TCP 179 et UDP 3784/3785 filtrés sur les hôtes
```

⚠ PIÈGE DE PRODUCTION : MTU ET VXLAN

Le problème de MTU est la cause principale des dysfonctionnements VXLAN en production. VXLAN ajoute 50 octets d'overhead (14 Ethernet + 8 UDP + 8 VXLAN + 20 IP outer). Si le réseau physique utilise des trames de 1500 octets, les VMs ne peuvent pas envoyer des paquets de plus de 1450 octets sans fragmentation. La solution recommandée est de configurer les interfaces physiques à 9000 octets (jumbo frames) sur tous les noeuds Proxmox et les équipements réseau intermédiaires, puis de définir le MTU des VMs à 1500 octets. Le SDN gère alors l'overhead VXLAN de façon transparente.

Résumé du chapitre

Ce chapitre a couvert l'ensemble des fonctionnalités SDN avancées de Proxmox VE 9 :

- **BGP EVPN (RFC 7432)** : protocole de plan de contrôle distribué qui remplace les mécanismes d'apprentissage MAC traditionnel par des annonces BGP structurées, avec route distinguishers (format `10.10.0.x:VNI`), route targets et anycast gateway.
- **FRRouting** : démon de routage intégré à Proxmox VE, configurable via `vtysh`, qui implémente BGP EVPN, OSPF, OpenFabric et BFD pour l'ensemble du plan de contrôle SDN.
- **SDN Fabrics** : nouvelle fonctionnalité de Proxmox VE 9 qui automatise la configuration du réseau underlay (OSPF ou OpenFabric) en intégrant cette gestion directement dans l'API Proxmox, réduisant considérablement la complexité opérationnelle.
- **Routage inter-VNets** : via les VRFs Linux et le route leaking BGP, permettant un contrôle précis des flux entre départements avec des politiques de sécurité granulaires et des route-maps filtrantes.

- **Haute disponibilité** : architecture route reflector redondante, BFD pour la détection rapide des pannes (moins de 400 ms), et failover automatique des VMs HA.
- **Sécurité SDN** : filtrage multi-niveaux (VNet, VM, ACL), micro-segmentation, authentification MD5 des sessions BGP, et filtrage des ports du plan de contrôle.
- **Lab TechFlow SAS** : déploiement complet d'une infrastructure EVPN 5 noeuds, AS 65000, avec 5 VNets départementaux (VNIs 10100-10500, L3 VNI 10000), routage inter-VRFs contrôlé, et validation end-to-end incluant les tests de failover.

✔ VERS LE CHAPITRE 7

Le chapitre suivant abordera la **haute disponibilité des clusters Proxmox VE** : Corosync, le quorum, la migration en direct des VMs, et la stratégie de placement dans un environnement multi-datacenter avec SDN EVPN géo-distribué.

PARTIE IV

Haute disponibilite et Ceph

7 Haute disponibilité

La haute disponibilité (HA) est l'une des fonctionnalités les plus critiques d'une infrastructure de virtualisation en production. Proxmox VE intègre nativement un gestionnaire HA complet, capable de détecter la défaillance d'un nœud et de redémarrer automatiquement les machines virtuelles et conteneurs sur les nœuds survivants, sans intervention humaine. Ce chapitre couvre l'ensemble du cycle de vie HA : concepts fondamentaux, prérequis, fencing, groupes HA, règles d'affinité introduites dans Proxmox VE 9, procédures de maintenance et validation par des tests de bascule contrôlés.

7.1 Concepts de la haute disponibilité

7.1.1 RTO, RPO et niveaux de disponibilité

La haute disponibilité repose sur deux métriques fondamentales qui définissent les objectifs contractuels ou opérationnels d'un service.

RTO — Recovery Time Objective

Le RTO désigne le temps maximum acceptable entre le moment où une panne survient et le moment où le service redevient opérationnel. Dans le contexte de Proxmox HA, le RTO correspond au délai entre la détection de la défaillance d'un nœud et le redémarrage complet de toutes les VM protégées sur les nœuds survivants.

Avec une configuration HA correctement paramétrée sur Proxmox VE 9, les valeurs typiques sont :

- Détection de la panne : 20 à 60 secondes (selon `ha-manager` et le watchdog)
- Fencing du nœud défaillant : 10 à 30 secondes supplémentaires
- Migration ou redémarrage des VM : 30 à 120 secondes par VM selon la charge

Un RTO global de 2 à 5 minutes est réaliste pour un cluster correctement configuré.

RPO — Recovery Point Objective

Le RPO définit la perte de données maximale tolérée, exprimée en temps. Avec Proxmox HA en mode redémarrage (restart), le RPO est égal au temps depuis la dernière écriture sur le stockage partagé. Si le stockage partagé est synchrone (Ceph, iSCSI, NFS), le RPO est théoriquement nul pour les données déjà committées sur disque.

> ⚠ **AVERTISSEMENT**
> Le RPO nul n'est garanti que si les applications écrivent effectivement sur le stockage partagé. Les données en mémoire vive non encore écrites sur disque au moment de la panne sont perdues, quel que soit le mécanisme HA.

Niveaux de disponibilité

Disponibilité	Indisponibilité/an	Indisponibilité/mois	Cas d'usage typique
99 %	87,6 h	7,3 h	Environnements de développement
99,5 %	43,8 h	3,65 h	Applications internes non critiques
99,9 % (3 nines)	8,76 h	43,8 min	Applications métier standard
99,95 %	4,38 h	21,9 min	Proxmox HA cluster 3 nœuds bien configuré
99,99 % (4 nines)	52,6 min	4,38 min	Infrastructure redondante avancée
99,999 % (5 nines)	5,26 min	26,3 s	Systèmes critiques avec redondance totale

Proxmox HA vise typiquement le niveau 99,95 % pour les VM protégées, à condition que l'infrastructure sous-jacente (réseau, stockage, alimentation) soit elle-même redondante.

7.1.2 Architecture HA dans Proxmox

L'architecture HA de Proxmox VE repose sur trois piliers interdépendants.

1. Le cluster Corosync

Corosync fournit la couche de communication entre les nœuds du cluster et le mécanisme de quorum. Le quorum détermine si le cluster dispose d'une majorité de nœuds actifs pour prendre des décisions. Sans quorum, le cluster se met en mode protectif et le gestionnaire HA cesse de fonctionner.

> ✗ **DANGER**
> Un cluster à 2 nœuds n'a pas de quorum automatique en cas de défaillance d'un nœud. Il est impératif d'utiliser soit un troisième nœud, soit un QDevice (Quorum Device) externe pour garantir le quorum dans tous les scénarios.

2. Le gestionnaire pve-ha-manager

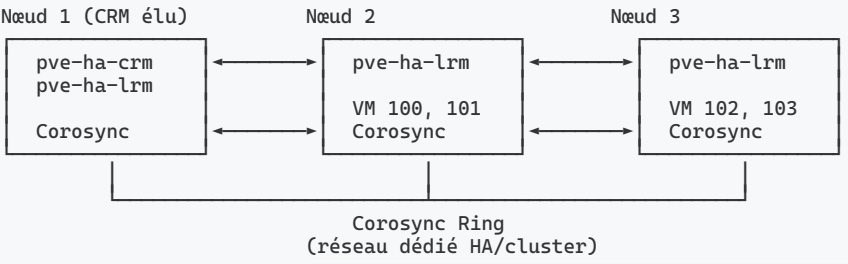
Le gestionnaire HA de Proxmox est divisé en deux composants distincts qui s'exécutent sur chaque nœud :

- **CRM (Cluster Resource Manager)** : élu sur un unique nœud maître, il prend les décisions globales (quel nœud doit héberger quelle ressource)
- **LRM (Local Resource Manager)** : s'exécute sur chaque nœud, exécute les ordres du CRM localement (démarrer, arrêter, migrer une VM)

3. Le mécanisme de fencing

Le fencing est le processus par lequel un nœud défaillant ou non communicant est physiquement isolé avant que ses ressources ne soient reprises par d'autres nœuds. Sans fencing fiable, il existe un risque de corruption de données si deux nœuds accèdent simultanément au même disque partagé (scénario dit « split-brain »).

Schéma d'interaction entre composants :



7.1.3 pve-ha-manager : lrm et crm

Le CRM (Cluster Resource Manager)

Le CRM est le cerveau du système HA. Il s'exécute uniquement sur le nœud maître élu (via Corosync) et est responsable de :

- Surveiller l'état de tous les nœuds membres
- Maintenir la table d'état de toutes les ressources HA
- Décider des actions à entreprendre (migration, redémarrage)
- Orchestrer le fencing des nœuds défaillants

```
# Statut du service pve-ha-crm
systemctl status pve-ha-crm

# Journal temps réel
journalctl -fu pve-ha-crm

# Nœud maître actuel et état général
ha-manager crm-status
```

Le LRM (Local Resource Manager)

Le LRM s'exécute sur chaque nœud du cluster et agit comme l'exécutant local des ordres du CRM. Il est responsable de :

- Démarrer et arrêter les VM/CT locales selon les ordres du CRM
- Reporter l'état local des ressources au CRM
- Gérer le watchdog local pour le fencing

```
# Statut du service pve-ha-lrm
systemctl status pve-ha-lrm

# Journal du LRM local
journalctl -fu pve-ha-lrm

# État des ressources locales gérées par ce LRM
ha-manager lrm-status
```

Communication CRM ↔ LRM

La communication entre CRM et LRM passe par le système de fichiers partagé Proxmox (`/etc/pve`), synchronisé via `pve-cluster` (basé sur Corosync + FUSE). Cette architecture garantit que les décisions du CRM sont atomiques et cohérentes.

```
# Fichiers d'état HA dans le filesystem partagé
ls -la /etc/pve/ha/
# crm_commands    : ordres du CRM vers les LRM
# manager_status  : état global du manager
# resources.cfg   : configuration des ressources HA
# groups.cfg      : configuration des groupes HA
# rules.cfg       : règles d'affinité (nouveau Proxmox VE 9)
```

7.1.4 États des ressources HA

Chaque ressource HA (VM ou CT) possède un état qui reflète sa situation courante dans le cycle de vie géré.

État	Description	Action typique du CRM
`started`	La ressource doit être démarrée et l'est	Surveillance continue
`stopped`	La ressource doit être arrêtée et l'est	Aucune action
`disabled`	HA désactivé pour cette ressource	Aucune action HA
`ignored`	Ressource ignorée par le gestionnaire HA	Aucune action
`error`	Erreur détectée, action manuelle requise	Alerte, pas de relance auto
`migrate`	Migration live en cours vers un autre nœud	Suivi de la migration
`relocate`	Déplacement non-live en cours	Suivi du déplacement
`fence`	Nœud hébergeant la ressource en cours de fencing	Attente du fencing
`recovery`	Récupération après panne en cours	Relance sur autre nœud
`freeze`	Ressource gelée temporairement (maintenance)	Attente levée du gel

Transitions d'états typiques lors d'une panne :

```
VM en fonctionnement normal :
  started → (nœud tombe) → fence → recovery → started (sur autre nœud)

VM dont le nœud perd le réseau cluster :
  started → fence (watchdog expire) → recovery → started

VM en maintenance planifiée :
  started → (migrate) → started (sur nœud cible)
```

Pour consulter les états en temps réel :

```
# Vue d'ensemble de toutes les ressources HA
ha-manager status
```

```
# Exemple de sortie :
# quorum OK
# master node: pve-node1
# lrm status:
#   pve-node1: online
#   pve-node2: online
#   pve-node3: online
# service status:
#   vm:100 (pve-node1) : started
#   vm:101 (pve-node2) : started
#   vm:102 (pve-node3) : started
```

7.2 Prérequis HA

7.2.1 Cluster actif obligatoire

La haute disponibilité dans Proxmox VE ne peut fonctionner que si un cluster Corosync est actif et opérationnel. Un cluster dégradé (sans quorum) suspend automatiquement toutes les actions HA.

> ⚠ **AVERTISSEMENT**
> Le gestionnaire HA ne prend aucune décision tant que le cluster n'a pas le quorum. Si plus de la moitié des nœuds est hors ligne, le cluster se bloque volontairement pour éviter les décisions incohérentes. C'est un comportement de sécurité, pas une défaillance.

Vérification du cluster :

```
# État complet du cluster
pvecm status

# Sortie attendue sur un cluster sain à 3 nœuds :
# Cluster information
# ──────────────────
# Name:             techflow-cluster
# Config Version:   3
# Transport:        knet
# Secure auth:      on
#
# Quorum information
# ──────────────────
# Date:             Tue Mar 17 14:30:00 2026
# Quorum provider:  corosync_votequorum
# Nodes:            3
# Node ID:          0x00000001
# Ring ID:          1.45
# Quorate:          Yes
#
# Votequorum information
# ──────────────────────
# Expected votes:   3
# Highest expected: 3
# Total votes:      3
# Quorum:           2
# Flags:            Quorate

# Lister les membres du cluster
pvecm nodes
```

Configuration réseau requise :

Le cluster HA nécessite idéalement deux réseaux distincts pour séparer le trafic de gestion du trafic cluster/stockage.

```
# Exemple /etc/network/interfaces pour un nœud HA TechFlow

# Réseau de gestion (accès web, API, SSH)
auto eno1
iface eno1 inet static
    address 10.0.1.11/24
```

```
    gateway 10.0.1.1

# Réseau cluster Corosync (dédié, sans gateway)
auto eno2
iface eno2 inet static
    address 10.10.0.11/24

# Réseau de stockage Ceph/NFS
auto eno3
iface eno3 inet static
    address 10.20.0.11/24
```

Configuration Corosync avec deux anneaux de communication :

```
# /etc/pve/corosync.conf
totem {
    version: 2
    cluster_name: techflow-cluster
    config_version: 3
    ip_version: ipv4-6
    link_mode: passive
    interface {
        linknumber: 0
        knet_link_priority: 1
    }
    interface {
        linknumber: 1
        knet_link_priority: 2
    }
}

nodelist {
    node {
        name: pve-node1
        nodeid: 1
        ring0_addr: 10.10.0.11
        ring1_addr: 10.10.1.11
    }
    node {
        name: pve-node2
        nodeid: 2
        ring0_addr: 10.10.0.12
        ring1_addr: 10.10.1.12
    }
    node {
        name: pve-node3
        nodeid: 3
        ring0_addr: 10.10.0.13
        ring1_addr: 10.10.1.13
    }
}

quorum {
    provider: corosync_votequorum
}

logging {
    to_syslog: yes
}
```

7.2.2 Stockage partagé

La haute disponibilité nécessite que les disques des VM protégées soient stockés sur un espace partagé accessible par tous les nœuds du cluster simultanément.

Type de stockage	Protocole	HA compatible	Performances	Cas d'usage
Ceph RBD		Oui (natif)	Élevées	

Type de stockage	Protocole	HA compatible	Performances	Cas d'usage
	Interne Proxmox			Recommandé pour nouveaux clusters
NFS	NFSv3/NFSv4	Oui	Moyennes	Stockage existant NAS
iSCSI	iSCSI + LVM	Oui	Élevées	SAN existant
GlusterFS	GlusterFS	Oui	Moyennes	Clusters distribués
ZFS over iSCSI	iSCSI	Oui	Élevées	ZFS avec réplication
Local (LVM, dir)	Aucun	**Non**	Très élevées	Hors HA

> ⚠ **AVERTISSEMENT**
> Le stockage local (type `dir`, `lvmthin` non partagé, `zfspool` local) n'est **pas** compatible avec la haute disponibilité. Proxmox refusera d'activer HA sur une VM dont des disques se trouvent sur un stockage non partagé.

Vérification de la compatibilité HA du stockage :

```
# Lister les stockages et leur statut
pvesm status

# Exemple de sortie :
# Name         Type    Status      Total      Used    Available  %
# ceph-pool    rbd     active   20971520   8388608    12582912  40%
# nfs-nas      nfs     active   10485760   2097152     8388608  20%
# local        dir     active    1048576    524288      524288  50%
# local-lvm    lvmthin active    2097152   1048576     1048576  50%

# Vérifier si un stockage est marqué comme partagé
pvesm show ceph-pool | grep shared
# shared 1

# Activer l'attribut partagé sur un stockage NFS
pvesm set nfs-nas --shared 1
```

Vérification de l'état Ceph :

```
# État global du cluster Ceph
ceph status

# Vérifier la politique de réplication du pool (minimum 2, recommandé 3)
ceph osd pool get vm-data size
# size: 3

ceph osd pool get vm-data min_size
# min_size: 2

# Santé des OSD
ceph osd stat
# 9 osds: 9 up (since 4d), 9 in (since 4d)
```

7.2.3 Fencing : principe et importance

Le fencing (ou STONITH — Shoot The Other Node In The Head) est le mécanisme qui garantit qu'un nœud défaillant ou partitionné ne peut plus accéder au stockage partagé avant que ses ressources ne soient reprises par les autres nœuds.

> ✗ **DANGER**
> Le fencing est une exigence absolue pour tout cluster HA en production. Un cluster HA sans fencing fiable expose les données partagées à une corruption potentielle en cas de partitionnement réseau (split-brain). Ne jamais déployer HA en production sans avoir validé le mécanisme de fencing de bout en bout.

Pourquoi le fencing est indispensable — scénario sans fencing :

1. Nœud A héberge VM 100 avec un disque sur stockage Ceph partagé
2. Le réseau cluster entre A et les autres nœuds tombe (split-brain)
3. Les nœuds B et C considèrent A comme mort et redémarrent VM 100 sur B
4. VM 100 tourne maintenant **deux fois** : sur A (toujours vivant, juste isolé) et sur B
5. Les deux instances écrivent simultanément sur le même disque Ceph → corruption garantie

Avec le fencing :

1. Même scénario : A perd la connectivité cluster
2. Le watchdog sur A expire car le LRM ne reçoit plus les keepalive du CRM
3. Le watchdog redémarre immédiatement A
4. A est hors ligne avant que B et C ne redémarrent VM 100
5. Aucun accès concurrent au disque → aucune corruption

Types de fencing supportés par Proxmox VE 9 :

Type	Mécanisme	Fiabilité	Matériel requis
Watchdog kernel (recommandé)	Kernel watchdog + `watchdog-mux`	Très haute	Aucun (softdog) ou Intel TCO
IPMI/BMC	Coupure alimentation via IPMI	Très haute	Carte IPMI/iDRAC/iLO
Agent externe	Script personnalisé	Variable	Selon l'agent

7.3 Configuration du fencing

7.3.1 Watchdog hardware (Intel TCO)

Le watchdog est le mécanisme de fencing primaire et recommandé dans Proxmox VE. Il fonctionne sur le principe du « dead man's switch » : le LRM doit régulièrement alimenter (« tickler ») le watchdog ; si le LRM cesse de fonctionner ou perd la connectivité cluster, le watchdog expire et redémarre le nœud.

Architecture du watchdog Proxmox :

```
pve-ha-lrm
     |
     |  (keepalive toutes les X secondes)
     v
watchdog-mux (daemon)
     |
     |  (forward keepalive)
     v
/dev/watchdog (kernel watchdog)
     |
     |  (si pas de keepalive dans le timeout configuré)
     v
Redémarrage matériel du nœud
```

Le service `watchdog-mux` est un multiplexeur qui permet à plusieurs processus de partager un unique périphérique watchdog kernel.

Vérification et configuration du watchdog :

```
# Vérifier si un module watchdog est déjà chargé
lsmod | grep -E "iTCO|softdog|i6300esb"

# Sur serveurs Dell/HP/Supermicro avec Intel TCO Watchdog
modprobe iTCO_wdt
modprobe iTCO_vendor_support

# Sur VMs de test ou machines sans watchdog hardware : softdog
modprobe softdog

# Vérifier la présence du périphérique watchdog
ls -la /dev/watchdog*
```

```
# crw------- 1 root root 10, 130 Mar 17 14:00 /dev/watchdog
# crw------- 1 root root 252,  0 Mar 17 14:00 /dev/watchdog0

# Identifier le watchdog actif
cat /sys/class/watchdog/watchdog0/identity
# iTCO_wdt

# Timeout configuré (en secondes)
cat /sys/class/watchdog/watchdog0/timeout
# 30
```

Configuration permanente du module watchdog :

```
# Pour Intel TCO (serveurs physiques, recommandé en production)
cat > /etc/modules-load.d/proxmox-watchdog.conf << 'EOF'
iTCO_wdt
iTCO_vendor_support
EOF

# Pour softdog (tests, VMs imbriquées)
cat > /etc/modules-load.d/proxmox-watchdog.conf << 'EOF'
softdog
EOF

# Si le module est blacklisté par la distribution
grep -r "iTCO_wdt" /etc/modprobe.d/
# Commenter la ligne blacklist correspondante si trouvée
```

État du service watchdog-mux :

```
# Vérifier que watchdog-mux est actif
systemctl status watchdog-mux

# Journaux watchdog-mux
journalctl -u watchdog-mux --since "1 hour ago"

# Inspection du périphérique watchdog via wdctl
wdctl /dev/watchdog0
# Device:        /dev/watchdog0
# Identity:      iTCO_wdt [version 0]
# Timeout:       30 seconds
# Pre-timeout:    0 seconds
# Timeleft:      28 seconds
# FLAG           DESCRIPTION               STATUS BOOT-STATUS
# KEEPALIVEPING  Keep alive ping reply          1           0
```

Paramètres du watchdog dans la configuration HA globale :

```
# Consulter l'état et la configuration du manager HA
pvesh get /cluster/ha/status/manager_status

# Voir les options HA du datacenter
pvesh get /cluster/options

# Modifier le timeout watchdog (défaut 60s, doit être > max_workers_interval du
LRM)
# Attention : cette commande modifie un paramètre critique
# pvesh set /cluster/options --ha-shutdown-policy migrate
```

7.3.2 Fencing IPMI/iDRAC/iLO

Pour les serveurs disposant d'un contrôleur de gestion hors-bande (IPMI, Dell iDRAC, HP iLO, Supermicro IPMI), Proxmox peut utiliser ces interfaces pour éteindre physiquement un nœud défaillant.

Prérequis et test de connectivité :

```
# Installer les outils IPMI
apt install -y freeipmi-tools ipmitool fence-agents
```

```
# Tester la connectivité IPMI depuis un autre nœud
ipmitool -H 10.0.1.254 -U admin -P motdepasse chassis status

# Tester la commande d'extinction (ATTENTION : coupe réellement le nœud)
# ipmitool -H 10.0.1.254 -U admin -P motdepasse chassis power off
# ipmitool -H 10.0.1.254 -U admin -P motdepasse chassis power on

# Test fence_ipmilan sans action réelle (mode status)
fence_ipmilan -a 10.0.1.254 -l admin -p motdepasse -o status
```

> ⚠ **AVERTISSEMENT**
> Les identifiants IPMI ne doivent jamais être transmis en clair dans des scripts versionnés. Utilisez des fichiers de configuration protégés (chmod 600) ou des secrets systemd pour stocker les mots de passe.

Intégration du fencing IPMI dans Proxmox :

```
# Proxmox gère le fencing via le plugin watchdog en priorité.
# L'IPMI sert de mécanisme complémentaire ou de remplacement
# quand le watchdog n'est pas disponible.

# Vérifier la configuration fencing dans le datacenter
pvesh get /cluster/ha/status/manager_status | python3 -m json.tool

# Configurer un agent IPMI via l'interface web :
# Datacenter → HA → Fencing → Add IPMI Device
# Ou via pvesh (Proxmox VE 9) :
pvesh create /cluster/ha/fence \
  --type ipmilan \
  --nodes pve-node2 \
  --ipaddr 10.0.1.252 \
  --login admin \
  --passwd "motdepasse_securise"
```

7.3.3 Fencing via agent externe

Proxmox VE supporte des agents de fencing externes conformes au protocole fence-agents standard. Cette approche permet d'intégrer des mécanismes personnalisés pour des environnements spécifiques.

Exemple : fencing via PDU réseau APC :

```
# Installer l'agent APC
apt install -y fence-agents-apc

# Lister les ports du PDU
fence_apc -a 10.0.1.200 -l apc -p motdepasse -o list

# Vérifier l'état du port 3 (hébergeant pve-node2)
fence_apc -a 10.0.1.200 -l apc -p motdepasse -n 3 -o status

# Tester le fencing du port 3
# fence_apc -a 10.0.1.200 -l apc -p motdepasse -n 3 -o off
# fence_apc -a 10.0.1.200 -l apc -p motdepasse -n 3 -o on
```

Script d'agent de fencing personnalisé :

```
#!/bin/bash
# /usr/local/bin/fence-custom.sh
# Agent de fencing pour API de gestion propriétaire

ACTION="${1}"
TARGET_NODE="${2}"

case "${ACTION}" in
    "off")
        curl -s -X POST "https://mgmt-api/nodes/${TARGET_NODE}/power" \
          -H "Authorization: Bearer ${MGMT_API_TOKEN}" \
          -d '{"action": "poweroff"}'
        exit $?
```

```
        ;;
    "on")
        curl -s -X POST "https://mgmt-api/nodes/${TARGET_NODE}/power" \
          -H "Authorization: Bearer ${MGMT_API_TOKEN}" \
          -d '{"action": "poweron"}'
        exit $?
        ;;
    "status")
        STATE=$(curl -s "https://mgmt-api/nodes/${TARGET_NODE}/power" \
          -H "Authorization: Bearer ${MGMT_API_TOKEN}" | jq -r '.state')
        echo "Status: ${STATE}"
        exit 0
        ;;
    *)
        echo "Usage: $0 {off|on|status} <node>"
        exit 1
        ;;
esac
```

```
# Sécuriser le script
chmod 750 /usr/local/bin/fence-custom.sh
chown root:root /usr/local/bin/fence-custom.sh
```

7.3.4 Test du fencing

> **✗ DANGER**
> Les tests de fencing en production doivent être planifiés avec soin. Un test mal conduit peut provoquer une indisponibilité non planifiée. Effectuez toujours les tests de fencing en dehors des heures de production, avec les équipes concernées informées et disponibles pour intervenir.

Test non destructif du watchdog (vérification d'état) :

```
# Vérifier que watchdog-mux alimente correctement le watchdog
wdctl /dev/watchdog0

# Vérifier les connexions actives au watchdog-mux
ss -lxp | grep watchdog

# Consulter le timeleft du watchdog (doit se réinitialiser régulièrement)
cat /sys/class/watchdog/watchdog0/timeleft
```

Test avec expiration contrôlée du watchdog (nœud de test uniquement) :

```
# ATTENTION : CE TEST REDÉMARRE LE NŒUD PHYSIQUEMENT
# À exécuter uniquement sur un nœud de test en dehors des heures de production
# Vérifier que toutes les VM HA ont été migrées avant le test

# Stopper watchdog-mux pour laisser le watchdog expirer
systemctl stop watchdog-mux

# Le nœud redémarre automatiquement dans 30-60 secondes (selon timeout)

# Depuis un autre nœud, surveiller la disparition et le retour
watch -n 2 "pvecm nodes; echo '---'; ha-manager status"
```

Vérification post-test :

```
# Vérifier que le nœud a redémarré via watchdog (et non arrêt propre)
last reboot

# Chercher la trace du redémarrage watchdog dans les logs kernel
journalctl -b -1 -k | grep -i "watchdog\|panic\|reboot"

# Vérifier que les VM HA ont bien basculé et sont revenues
ha-manager status

# Vérifier les logs CRM pour la chronologie de la bascule
journalctl -u pve-ha-crm --since "-2 hours" | grep -E "fence|recover|started"
```

7.4 Groupes HA et ressources

7.4.1 Création d'un groupe HA

Un groupe HA définit un ensemble de nœuds sur lesquels les ressources membres de ce groupe peuvent être hébergées. Il permet de contrôler finement la répartition des VM en cas de bascule.

Via l'interface web :

Datacenter → HA → Groups → Create

Via la ligne de commande :

```
# Créer un groupe HA de base
ha-manager groupadd ha-prod \
  --nodes "pve-node1:2,pve-node2:1,pve-node3:1" \
  --nofailback 0 \
  --restricted 0 \
  --comment "Groupe HA production TechFlow"

# Vérifier la configuration du groupe
ha-manager groupconfig ha-prod

# Sortie attendue :
# group: ha-prod
#   comment: Groupe HA production TechFlow
#   nodes: pve-node1:2,pve-node2:1,pve-node3:1
#   nofailback: 0
#   restricted: 0
```

Via l'API REST pvesh :

```
# Créer un groupe via pvesh
pvesh create /cluster/ha/groups \
  --group ha-prod \
  --nodes "pve-node1:2,pve-node2:1,pve-node3:1" \
  --nofailback 0 \
  --restricted 0 \
  --comment "Groupe HA production TechFlow"

# Lister tous les groupes existants
pvesh get /cluster/ha/groups

# Détail d'un groupe spécifique
pvesh get /cluster/ha/groups/ha-prod

# Modifier un groupe existant
pvesh set /cluster/ha/groups/ha-prod --nofailback 1

# Supprimer un groupe (doit être vide de ressources)
pvesh delete /cluster/ha/groups/ha-prod
```

7.4.2 Priorités et politiques

Priorités des nœuds dans un groupe

Chaque nœud d'un groupe HA peut recevoir une priorité (0 à 100, défaut 0 si non spécifié). Le CRM utilise ces priorités pour choisir le nœud cible lors d'une bascule. Une priorité plus haute indique un nœud préféré.

```
# Groupe avec priorités différenciées
# pve-node1 : priorité 2 (nœud préféré, plus de RAM)
# pve-node2 : priorité 1 (secondaire)
# pve-node3 : priorité 1 (secondaire, même priorité que node2)
pvesh create /cluster/ha/groups \
  --group ha-db \
  --nodes "pve-node1:2,pve-node2:1,pve-node3:1" \
  --comment "Groupe HA bases de données"
```

Politique nofailback

Par défaut (`nofailback: 0`), lorsqu'un nœud prioritaire revient en ligne après une panne, les VM basculent automatiquement vers ce nœud préféré. Ce comportement peut être désactivé pour éviter les migrations intempestives en production.

```
# Désactiver le failback automatique (recommandé en production)
ha-manager groupset ha-prod --nofailback 1

# Avec nofailback actif, les VM restent sur le nœud de bascule
# jusqu'à une migration manuelle explicite
pvesh set /cluster/ha/groups/ha-prod --nofailback 1
```

Politique restricted

Lorsque `restricted: 1` , les ressources du groupe ne peuvent être hébergées **que** sur les nœuds listés. Si aucun nœud du groupe n'est disponible, la ressource reste hors ligne plutôt que de basculer hors groupe.

Politique	Description	Quand utiliser
`nofailback: 0`	Failback automatique vers nœud préféré	Environnements avec charge équilibrée
`nofailback: 1`	Pas de failback automatique	Production (évite les migrations intempestives)
`restricted: 0`	Migration possible sur tout nœud du cluster	Haute disponibilité prioritaire
`restricted: 1`	Migration uniquement sur nœuds du groupe	Licences, conformité, localité des données

7.4.3 Ajout de VM/CT au groupe HA

Via la ligne de commande ha-manager :

```
# Ajouter une VM au groupe HA avec état désiré "started"
ha-manager add vm:100 \
  --group ha-prod \
  --state started \
  --comment "VM web-prod-01 protégée par HA"

# Ajouter un conteneur LXC
ha-manager add ct:200 \
  --group ha-prod \
  --state started \
  --comment "CT nginx-prod protégé par HA"

# Vérifier toutes les ressources HA configurées
ha-manager config
```

Via l'API pvesh :

```
# Ajouter la VM 101 au groupe ha-prod
pvesh create /cluster/ha/resources \
  --sid vm:101 \
  --group ha-prod \
  --state started \
  --comment "VM app-prod-01"

# Configurer les paramètres avancés de reprise
# max_restart  : tentatives de redémarrage sur le même nœud avant échec (défaut:
1)
# max_relocate : tentatives de migration vers autre nœud avant état error (défaut:
1)
pvesh set /cluster/ha/resources/vm:101 \
  --group ha-prod \
  --max_restart 3 \
  --max_relocate 1

# Supprimer une ressource du HA (la VM continue de tourner, plus gérée par HA)
pvesh delete /cluster/ha/resources/vm:101
```

Paramètres importants des ressources HA :

```
# Exemple avec tous les paramètres
pvesh create /cluster/ha/resources \
  --sid vm:102 \
  --group ha-prod \
  --state started \
  --max_restart 2 \
  --max_relocate 1 \
  --comment "VM db-prod avec politique de retry"
```

7.4.4 États et transitions

Changer l'état désiré d'une ressource HA :

```
# Arrêter une VM via HA (arrêt géré par le LRM, pas ACPI direct)
ha-manager set vm:100 --state stopped

# Redémarrer sous contrôle HA
ha-manager set vm:100 --state started

# Désactiver temporairement HA (pour opérations de maintenance sur la VM)
ha-manager set vm:100 --state disabled

# Ignorer complètement une VM par le gestionnaire HA
ha-manager set vm:100 --state ignored

# Remettre en surveillance active
ha-manager set vm:100 --state started
```

Observer les transitions en temps réel :

```
# Surveillance continue de l'état HA (rafraîchissement toutes les 2s)
watch -n 2 ha-manager status

# Journaux détaillés du CRM (filtrer le bruit DEBUG)
journalctl -fu pve-ha-crm | grep -v DEBUG

# Journaux du LRM sur le nœud local
journalctl -fu pve-ha-lrm

# Historique des événements HA des dernières 24h
journalctl -u pve-ha-crm --since "24 hours ago" | \
  grep -E "fence|recover|started|stopped|error"
```

7.5 Règles d'affinité (nouveau Proxmox VE 9)

Proxmox VE 9 introduit une nouvelle fonctionnalité majeure : les règles d'affinité (affinity rules). Ces règles permettent de contrôler la co-location ou la séparation des VM sur les nœuds du cluster. Elles opèrent via l'endpoint API `/cluster/ha/rules` et interagissent directement avec les décisions de placement du CRM.

7.5.1 Affinité positive (co-location)

Une règle d'affinité positive garantit que les VM membres de la règle sont hébergées de préférence (politique `soft`) ou obligatoirement (politique `hard`) sur le **même nœud physique**.

Cas d'usage typiques :

- Serveur d'application et son cache Redis dédié (latence loopback vs réseau)
- Base de données et son agent de backup local
- Paires de VM haute communication (HPC, MPI)

```
# Créer une règle d'affinité positive stricte (co-location obligatoire)
pvesh create /cluster/ha/rules \
  --type affinity \
  --vms "100,109" \
  --policy hard \
  --comment "App-prod + Redis : co-location obligatoire"
```

```
# Règle d'affinité positive souple (co-location préférentielle)
pvesh create /cluster/ha/rules \
  --type affinity \
  --vms "104,109" \
  --policy soft \
  --comment "App-prod-01 + Redis : co-location préférée"

# Lister les règles existantes
pvesh get /cluster/ha/rules
```

Politique	Comportement	Impact en cas de bascule
`soft`	Co-location préférée, non bloquante	Migration possible indépendamment si nécessaire
`hard`	Co-location obligatoire, bloquante	Bascule uniquement si nœud cible peut héberger toutes les VM

7.5.2 Affinité négative (séparation)

Une règle d'affinité négative (anti-affinité) garantit que les VM membres de la règle sont hébergées sur des **nœuds différents**. C'est le pattern le plus important en combinaison avec HA.

Cas d'usage typiques :

- Instances redondantes d'un même service (HAProxy actif/actif)
- Nœuds d'un cluster applicatif (Kubernetes workers, Galera cluster)
- Primaire et replica d'une base de données

```
# Règle d'anti-affinité stricte (séparation obligatoire)
pvesh create /cluster/ha/rules \
  --type anti-affinity \
  --vms "103,108" \
  --policy hard \
  --comment "HAProxy instances : nœuds physiques distincts obligatoires"

# Règle d'anti-affinité souple (séparation préférée)
pvesh create /cluster/ha/rules \
  --type anti-affinity \
  --vms "100,101,102" \
  --policy soft \
  --comment "Web cluster : répartition préférentielle sur nœuds distincts"

# Anti-affinité PostgreSQL primaire/replica
pvesh create /cluster/ha/rules \
  --type anti-affinity \
  --vms "106,107" \
  --policy hard \
  --comment "PostgreSQL primary/replica : séparation obligatoire"
```

7.5.3 Configuration via pvesh

L'API `pvesh` est l'outil de référence pour gérer les règles d'affinité dans Proxmox VE 9. Voici la référence complète des endpoints :

```
# Lister toutes les règles d'affinité
pvesh get /cluster/ha/rules

# Créer une nouvelle règle
pvesh create /cluster/ha/rules \
  --type <affinity|anti-affinity> \
  --vms "<vmid1,vmid2, ... >" \
  --policy <soft|hard> \
  --comment "<description>"

# Consulter une règle spécifique (par son ID)
pvesh get /cluster/ha/rules/1
```

```
# Modifier une règle existante
pvesh set /cluster/ha/rules/1 \
  --policy soft \
  --comment "Règle modifiée en soft"

# Supprimer une règle
pvesh delete /cluster/ha/rules/1
```

Script de configuration complète des règles d'affinité TechFlow :

```
#!/bin/bash
# configure-ha-affinity-rules.sh
# Initialisation des règles d'affinité pour le cluster TechFlow SAS

set -euo pipefail

echo "═══ Configuration des règles d'affinité HA TechFlow ═══"

# Supprimer toutes les règles existantes avant reconfiguration
EXISTING_RULES=$(pvesh get /cluster/ha/rules --output-format json 2>/dev/null | \
  python3 -c "import sys,json; [print(r['ruleid']) for r in json.load(sys.stdin)]"
|| true)

for rule_id in ${EXISTING_RULES}; do
    echo "Suppression règle ID ${rule_id} ... "
    pvesh delete /cluster/ha/rules/${rule_id}
done

# Règle 1 : Anti-affinité HAProxy (103 et 108 sur nœuds distincts)
pvesh create /cluster/ha/rules \
  --type anti-affinity --vms "103,108" --policy hard \
  --comment "HAProxy load-balancers : nœuds distincts obligatoires"
echo "✓ Anti-affinité HAProxy (103,108)"

# Règle 2 : Anti-affinité web servers (100, 101, 102 répartis)
pvesh create /cluster/ha/rules \
  --type anti-affinity --vms "100,101,102" --policy soft \
  --comment "Web cluster : répartition préférentielle"
echo "✓ Anti-affinité web (100,101,102)"

# Règle 3 : Anti-affinité PostgreSQL (106 et 107 sur nœuds distincts)
pvesh create /cluster/ha/rules \
  --type anti-affinity --vms "106,107" --policy hard \
  --comment "PostgreSQL primary/replica : nœuds distincts"
echo "✓ Anti-affinité DB (106,107)"

# Règle 4 : Affinité App + Cache (104 et 109 co-localisés)
pvesh create /cluster/ha/rules \
  --type affinity --vms "104,109" --policy soft \
  --comment "App-prod-01 + Redis : co-location préférée"
echo "✓ Affinité App+Cache (104,109)"

# Règle 5 : Anti-affinité app servers (104 et 105 sur nœuds différents)
pvesh create /cluster/ha/rules \
  --type anti-affinity --vms "104,105" --policy soft \
  --comment "App servers : séparation préférentielle"
echo "✓ Anti-affinité app servers (104,105)"

echo ""
echo "═══ Récapitulatif final ═══"
pvesh get /cluster/ha/rules
```

7.5.4 Cas d'usage : HA + anti-affinité

La combinaison HA + règles d'anti-affinité est le pattern de référence pour les applications critiques en cluster. Elle garantit à la fois la reprise automatique en cas de panne et la répartition des instances sur des hôtes physiquement distincts.

Scénario complet : cluster HAProxy actif/actif protégé par HA

```
# 1. Créer le groupe HA pour les load-balancers
pvesh create /cluster/ha/groups \
  --group ha-lb \
  --nodes "pve-node1:1,pve-node2:1,pve-node3:1" \
  --nofailback 1 \
  --restricted 0 \
  --comment "Groupe HA load-balancers TechFlow"

# 2. Protéger les deux VM HAProxy par HA
pvesh create /cluster/ha/resources \
  --sid vm:103 --group ha-lb --state started \
  --max_restart 3 --max_relocate 2 \
  --comment "HAProxy instance 1"

pvesh create /cluster/ha/resources \
  --sid vm:108 --group ha-lb --state started \
  --max_restart 3 --max_relocate 2 \
  --comment "HAProxy instance 2"

# 3. Créer la règle d'anti-affinité stricte
pvesh create /cluster/ha/rules \
  --type anti-affinity \
  --vms "103,108" \
  --policy hard \
  --comment "HAProxy : séparation obligatoire sur nœuds distincts"
```

Comportement en cas de panne avec cette configuration :

```
État normal :
  pve-node1 : VM 103 (HAProxy-1) ← HA protégée
  pve-node2 : VM 108 (HAProxy-2) ← HA protégée, anti-affinité avec 103

Panne de pve-node1 :
  1. Watchdog iTCO expire → pve-node1 redémarre (fencing)
  2. CRM détecte pve-node1 hors ligne, état fence confirmé
  3. CRM évalue les nœuds candidats pour VM 103 :
     - pve-node2 : règle anti-affinité hard avec VM 108 → EXCLU
     - pve-node3 : pas de contrainte → ACCEPTÉ
  4. VM 103 redémarre sur pve-node3
  5. HAProxy-2 (node2) continue de servir le trafic pendant la bascule

Résultat : anti-affinité respectée, continuité de service garantie
```

7.6 Migration et maintenance

7.6.1 Migration live avec HA actif

Lorsqu'une VM est gérée par HA, les migrations doivent être initiées via le gestionnaire HA pour garantir la cohérence des états et le respect des règles d'affinité.

```
# Migration live d'une VM sous HA (méthode recommandée via ha-manager)
ha-manager migrate vm:100 pve-node2

# Migration directe qm (fonctionne aussi, mais contourne les politiques HA)
qm migrate 100 pve-node2 --online

# Surveiller la migration en cours
watch -n 1 ha-manager status

# Vérifier le placement après migration
pvesh get /cluster/ha/resources/vm:100
```

Évacuation manuelle des VM d'un nœud :

```
# Migrer toutes les VM HA d'un nœud spécifique vers les autres
NODE_SOURCE="pve-node2"
NODE_TARGET="pve-node3"

# Récupérer les VMID gérés par HA sur ce nœud
```

```
VMIDS=$(ha-manager status | grep "${NODE_SOURCE}" | \
        grep "vm:" | awk '{print $1}')

for VM in ${VMIDS}; do
    echo "Migration ${VM} depuis ${NODE_SOURCE}..."
    ha-manager migrate ${VM} ${NODE_TARGET}
    # Attendre que la migration soit terminée avant la suivante
    sleep 10
done

# Vérifier l'état final
ha-manager status
```

7.6.2 Mode maintenance d'un nœud

Proxmox VE intègre un mode maintenance qui gère automatiquement l'évacuation des VM HA avant d'autoriser la maintenance du nœud.

```
# Activer le mode maintenance sur pve-node2
# (déclenche la migration automatique de toutes les VM HA du nœud)
ha-manager crm-command node-maintenance enable pve-node2

# Suivre l'évacuation en temps réel
watch -n 2 ha-manager status
# pve-node2 passe par les états :
#   online → maintenance-requested → maintenance (quand toutes VM migrées)

# Vérifier que le nœud est bien en maintenance
pvesh get /cluster/ha/status/manager_status | \
  python3 -c "
import sys, json
d = json.load(sys.stdin)
node_status = d.get('node_status', {})
for node, state in node_status.items():
    print(f'{node}: {state}')
"

# Effectuer la maintenance (mises à jour, remplacement hardware, etc.)
apt update && apt upgrade -y
# reboot si nécessaire

# Après retour du nœud, sortir du mode maintenance
ha-manager crm-command node-maintenance disable pve-node2

# Les VM reviennent selon la politique nofailback du groupe HA
ha-manager status
```

> ⚠ **AVERTISSEMENT**
> Si `nofailback: 1` est configuré sur le groupe HA, les VM ne reviendront pas automatiquement vers le nœud sorti de maintenance. Un rééquilibrage manuel est nécessaire pour restaurer la distribution optimale et respecter les règles d'anti-affinité.

7.6.3 Rolling update du cluster

La mise à jour en continu (rolling update) permet d'appliquer les mises à jour sur chaque nœud sans interruption de service global.

```
#!/bin/bash
# rolling-update-cluster.sh
# Mise à jour en continu du cluster TechFlow sans interruption HA

set -euo pipefail

NODES=("pve-node1" "pve-node2" "pve-node3")
MAINTENANCE_WAIT=300

for NODE in "${NODES[@]}"; do
```

```
    echo "═══════════════════════════════════════"
    echo " Début maintenance nœud : ${NODE}"
    echo " Heure : $(date +%H:%M:%S)"
    echo "═══════════════════════════════════════"

    # 1. Activer le mode maintenance
    echo "[1/5] Activation mode maintenance ${NODE} ..."
    ha-manager crm-command node-maintenance enable "${NODE}"

    # 2. Attendre l'évacuation complète des VM HA
    echo "[2/5] Attente évacuation des VM HA ..."
    WAIT=0
    while ha-manager status 2>/dev/null | grep -q "${NODE}.*started"; do
        sleep 10
        WAIT=$((WAIT + 10))
        echo "  Attente ... ${WAIT}s / ${MAINTENANCE_WAIT}s"
        if [ "${WAIT}" -gt "${MAINTENANCE_WAIT}" ]; then
            echo "ERREUR : Timeout évacuation ${NODE}" >&2
            exit 1
        fi
    done
    echo "  Nœud ${NODE} évacué."

    # 3. Application des mises à jour
    echo "[3/5] Mise à jour des paquets sur ${NODE} ..."
    ssh "root@${NODE}" "apt update -qq && DEBIAN_FRONTEND=noninteractive apt
upgrade -y"

    # 4. Redémarrage si nécessaire
    if ssh "root@${NODE}" "[ -f /var/run/reboot-required ]"; then
        echo "[4/5] Redémarrage de ${NODE} ..."
        ssh "root@${NODE}" "reboot" || true
        sleep 30
        until ssh -o ConnectTimeout=5 "root@${NODE}" "exit 0" 2>/dev/null; do
            echo "  Attente retour ${NODE} ..."
            sleep 10
        done
        echo "  ${NODE} de retour en ligne."
    else
        echo "[4/5] Pas de redémarrage nécessaire."
    fi

    # 5. Sortie du mode maintenance
    echo "[5/5] Sortie mode maintenance ${NODE} ..."
    ha-manager crm-command node-maintenance disable "${NODE}"
    sleep 30
    echo "  ${NODE} mis à jour avec succès."
    echo ""
done

echo "═══ Rolling update terminé pour tous les nœuds ═══"
ha-manager status
```

7.7 Tests et simulation de pannes

7.7.1 Simulation coupure nœud

✗ DANGER

Les simulations de panne en production doivent être effectuées avec l'accord écrit de toutes les parties prenantes, hors des heures de pointe, avec un plan de retour arrière documenté et une équipe disponible pour intervenir. Ne jamais simuler une panne sans fencing validé au préalable.

Méthode 1 : Isolation réseau (la plus réaliste)

```
# À exécuter depuis la console IPMI/iLO du nœud ciblé
# (la connexion SSH sera perdue lors de l'isolation)
```

```
# Depuis un autre nœud, démarrer la surveillance AVANT l'isolation
watch -n 1 "pvecm nodes; echo '--- HA STATUS ---'; ha-manager status"

# Sur pve-node2 (depuis console IPMI) :
# Bloquer les interfaces réseau cluster et stockage
iptables -I INPUT -i eno2 -j DROP
iptables -I OUTPUT -o eno2 -j DROP

# Attendre la détection, le fencing et la bascule (~30-90s)
```

Méthode 2 : Arrêt brutal via sysrq

```
# Déclenchement immédiat d'un redémarrage noyau (simule kernel panic)
# À exécuter depuis console IPMI uniquement
echo b > /proc/sysrq-trigger

# Observer depuis pve-node1 :
journalctl -fu pve-ha-crm | grep -E "fence|recover|node2"
```

Méthode 3 : Arrêt des services HA (simulation douce)

```
# Sur pve-node2 : arrêter les services HA
# (le watchdog expire seul après le timeout)
systemctl stop pve-ha-lrm
systemctl stop pve-ha-crm

# Observer la bascule depuis pve-node1
watch -n 2 ha-manager status
```

Chronologie observée lors d'une simulation réussie :

```
T+0s   : Nœud pve-node2 perd la connectivité cluster
T+20s  : Corosync détecte la perte de pve-node2
T+25s  : CRM marque pve-node2 comme "suspect"
T+30s  : Watchdog sur pve-node2 expire → redémarrage automatique
T+35s  : CRM confirme pve-node2 fencé
T+40s  : CRM ordonne aux LRM survivants de reprendre les VM
T+60s  : VM 101 démarre sur pve-node1
T+75s  : VM 108 démarre sur pve-node3
T+90s  : Toutes VM HA reprises, services opérationnels
T+180s : pve-node2 termine son redémarrage, rejoint le cluster
```

7.7.2 Test fencing contrôlé

```
#!/bin/bash
# test-fencing.sh
# Test automatisé du fencing avec mesure du RTO

TARGET_NODE="${1:-pve-node2}"
TEST_VM="${2:-100}"

echo "═══ Test de fencing — Nœud : ${TARGET_NODE}, VM test : ${TEST_VM} ═══"

# Vérifications préalables
echo "[PRÉ-TEST] Vérification état cluster…"
if ! pvecm status | grep -q "Quorate:.*Yes"; then
    echo "ERREUR : Cluster sans quorum, test annulé." >&2
    exit 1
fi

VM_STATUS=$(ha-manager status | grep "vm:${TEST_VM}" || true)
if [ -z "${VM_STATUS}" ]; then
    echo "ERREUR : VM ${TEST_VM} non trouvée dans HA." >&2
    exit 1
fi

CURRENT_NODE=$(echo "${VM_STATUS}" | grep -oP '\(\K[^)]+')
echo "VM ${TEST_VM} actuellement sur : ${CURRENT_NODE}"
```

```
# Déclenchement
T_START=$(date +%s)
echo ""
echo "[TEST] Déclenchement panne nœud ${TARGET_NODE} à $(date +%H:%M:%S)..."
ssh "root@${TARGET_NODE}" \
  "systemctl stop pve-ha-lrm; systemctl stop pve-ha-crm" || true

# Surveillance de la bascule
echo "[ATTENTE] Surveillance bascule (max 300s)..."
ELAPSED=0
RECOVERED=false

while [ "${ELAPSED}" -lt 300 ]; do
    sleep 5
    ELAPSED=$((ELAPSED + 5))

    NEW_STATUS=$(ha-manager status 2>/dev/null | grep "vm:${TEST_VM}" || echo "")
    NEW_NODE=$(echo "${NEW_STATUS}" | grep -oP '\(\K[^)]+' || echo "")

    echo "  T+${ELAPSED}s : ${NEW_STATUS}"

    if echo "${NEW_STATUS}" | grep -q "started" && \
       [ -n "${NEW_NODE}" ] && [ "${NEW_NODE}" ≠ "${CURRENT_NODE}" ]; then
        T_END=$(date +%s)
        RTO=$((T_END - T_START))
        echo ""
        echo "═══ BASCULE RÉUSSIE ═══"
        echo "  VM ${TEST_VM} : ${CURRENT_NODE} → ${NEW_NODE}"
        echo "  RTO mesuré   : ${RTO} secondes"
        RECOVERED=true
        break
    fi
done

if [ "${RECOVERED}" = false ]; then
    echo "ERREUR : Bascule non détectée dans les 300 secondes." >&2
    exit 1
fi

echo ""
echo "═══ Test de fencing terminé avec succès ═══"
```

7.7.3 Vérification des temps de bascule

Collecte et analyse des métriques de bascule :

```
# Analyser les journaux CRM pour extraire les timestamps clés
journalctl -u pve-ha-crm --since "2 hours ago" --no-pager | \
  grep -E "fence|recover|started|node.*lost" | \
  awk '{print $1, $2, $3, substr($0, index($0,$5))}'

# Script d'analyse du RTO depuis les journaux
python3 << 'EOF'
import subprocess, re
from datetime import datetime

result = subprocess.run(
    ["journalctl", "-u", "pve-ha-crm", "--since", "-2h", "--no-pager"],
    capture_output=True, text=True
)

events = []
for line in result.stdout.splitlines():
    ts_match = re.match(r'(\w+ \d+ \d+:\d+:\d+)', line)
    if not ts_match:
        continue
    ts = ts_match.group(1)
    if "lost quorum" in line.lower() or "node.*lost" in line.lower():
```

```
        events.append(("node_lost", ts))
    elif "fence" in line.lower():
        events.append(("fence", ts))
    elif "recovery" in line.lower():
        events.append(("recovery", ts))
    elif "started" in line and "vm:" in line:
        events.append(("vm_started", ts))

for event_type, timestamp in events:
    print(f"{timestamp}  {event_type}")
EOF
```

Tableau de suivi des tests de bascule TechFlow :

Date	Nœud testé	VM concernées	T détection	T fencing	T recovery	RTO total	Résultat
2026-03-17	pve-node2	101,105,107,108	22s	31s	87s	95s	OK
2026-03-10	pve-node3	102,110	18s	28s	70s	88s	OK
2026-03-01	pve-node1	100,103,104,106,109	25s	35s	120s	145s	OK

Seuils d'alerte et script de vérification :

```
# Seuils recommandés pour TechFlow SAS
RTO_WARNING=180   # 3 minutes
RTO_CRITICAL=300  # 5 minutes

check_ha_rto() {
    local RTO="${1}"
    if [ "${RTO}" -gt "${RTO_CRITICAL}" ]; then
        echo "CRITICAL: RTO ${RTO}s dépasse seuil critique ${RTO_CRITICAL}s"
        return 2
    elif [ "${RTO}" -gt "${RTO_WARNING}" ]; then
        echo "WARNING: RTO ${RTO}s dépasse seuil avertissement ${RTO_WARNING}s"
        return 1
    else
        echo "OK: RTO ${RTO}s dans les limites acceptables (< ${RTO_WARNING}s)"
        return 0
    fi
}
```

7.8 Lab TechFlow SAS : HA complet

TechFlow SAS héberge ses applications critiques sur un cluster Proxmox VE 9 à trois nœuds. Ce lab documente la configuration complète HA de leur infrastructure de production.

Infrastructure du cluster TechFlow :

Nœud	Processeur	RAM	Stockage local	IP Cluster	IP Gestion	IPMI
pve-node1	Xeon E5-2680v4 (28c)	256 GB	2×480 GB SSD	10.10.0.11	10.0.1.11	10.0.0.11
pve-node2	Xeon E5-2680v4 (28c)	256 GB	2×480 GB SSD	10.10.0.12	10.0.1.12	10.0.0.12
pve-node3	Xeon E5-2680v4 (28c)	256 GB	2×480 GB SSD	10.10.0.13	10.0.1.13	10.0.0.13

Stockage partagé :

- Pool Ceph `vm-data` : 27 TB utilisables (9 OSD × 4 TB, réplication ×3)
- Pool Ceph `vm-ssd` : 4,5 TB SSD NVMe (réplication ×3, pour bases de données)
- NFS TrueNAS `nfs-backup` : 50 TB (sauvegardes PBS, non utilisé pour HA)

VM protégées par HA (VM 100–110) :

VMID	Nom	Rôle	RAM	vCPU	Disque	Groupe HA
100	web-prod-01	Nginx web server	4 GB	4	vm-data	ha-prod

VMID	Nom	Rôle	RAM	vCPU	Disque	Groupe HA
101	web-prod-02	Nginx web server	4 GB	4	vm-data	ha-prod
102	web-prod-03	Nginx web server	4 GB	4	vm-data	ha-prod
103	lb-prod-01	HAProxy load-balancer	2 GB	2	vm-data	ha-lb
104	app-prod-01	Application Java	8 GB	8	vm-data	ha-prod
105	app-prod-02	Application Java	8 GB	8	vm-data	ha-prod
106	db-prod-01	PostgreSQL primaire	32 GB	16	vm-ssd	ha-db
107	db-prod-02	PostgreSQL replica	32 GB	16	vm-ssd	ha-db
108	lb-prod-02	HAProxy load-balancer	2 GB	2	vm-data	ha-lb
109	cache-prod-01	Redis cache	8 GB	4	vm-data	ha-prod
110	monitor-prod	Prometheus/Grafana	4 GB	4	vm-data	ha-prod

7.8.1 Configuration des groupes HA

```
#!/bin/bash
# techflow-ha-groups.sh
# Configuration des groupes HA pour TechFlow SAS

set -euo pipefail

echo "═══════════════════════════════════════════"
echo " Configuration HA TechFlow SAS"
echo " Proxmox VE 9 — $(date +%Y-%m-%d)"
echo "═══════════════════════════════════════════"

# ─── GROUPE 1 : ha-prod — Production générale ──────────────────
pvesh create /cluster/ha/groups \
  --group ha-prod \
  --nodes "pve-node1:2,pve-node2:2,pve-node3:1" \
  --nofailback 1 \
  --restricted 0 \
  --comment "Groupe HA production TechFlow — web, app, cache, monitoring"
echo "[OK] Groupe ha-prod créé"

# ─── GROUPE 2 : ha-db — Bases de données ──────────────────────
# Priorité plus haute sur node1 et node2 (plus de RAM disponible)
pvesh create /cluster/ha/groups \
  --group ha-db \
  --nodes "pve-node1:3,pve-node2:2,pve-node3:1" \
  --nofailback 1 \
  --restricted 0 \
  --comment "Groupe HA bases de données TechFlow — PostgreSQL"
echo "[OK] Groupe ha-db créé"

# ─── GROUPE 3 : ha-lb — Load-balancers ────────────────────────
# Priorités égales pour forcer la répartition équilibrée
pvesh create /cluster/ha/groups \
  --group ha-lb \
  --nodes "pve-node1:1,pve-node2:1,pve-node3:1" \
  --nofailback 1 \
  --restricted 0 \
  --comment "Groupe HA load-balancers TechFlow — HAProxy"
echo "[OK] Groupe ha-lb créé"

# ─── RESSOURCES HA : VM 100-110 ───────────────────────────────

echo ""
echo "--- Enregistrement des ressources HA ---"

# Web servers → ha-prod
for VMID in 100 101 102; do
```

```
    pvesh create /cluster/ha/resources \
      --sid "vm:${VMID}" --group ha-prod --state started \
      --max_restart 2 --max_relocate 1 \
      --comment "Nginx web server VM ${VMID}"
    echo "[OK] VM ${VMID} → ha-prod"
done

# Load-balancers → ha-lb (plus de tentatives, critique)
for VMID in 103 108; do
    pvesh create /cluster/ha/resources \
      --sid "vm:${VMID}" --group ha-lb --state started \
      --max_restart 3 --max_relocate 2 \
      --comment "HAProxy load-balancer VM ${VMID}"
    echo "[OK] VM ${VMID} → ha-lb"
done

# App servers → ha-prod
for VMID in 104 105; do
    pvesh create /cluster/ha/resources \
      --sid "vm:${VMID}" --group ha-prod --state started \
      --max_restart 2 --max_relocate 1 \
      --comment "Application server Java VM ${VMID}"
    echo "[OK] VM ${VMID} → ha-prod"
done

# Bases de données → ha-db (conservateur : 1 restart, 1 relocate)
for VMID in 106 107; do
    pvesh create /cluster/ha/resources \
      --sid "vm:${VMID}" --group ha-db --state started \
      --max_restart 1 --max_relocate 1 \
      --comment "PostgreSQL VM ${VMID} — conservative HA"
    echo "[OK] VM ${VMID} → ha-db"
done

# Cache et monitoring → ha-prod
for VMID in 109 110; do
    pvesh create /cluster/ha/resources \
      --sid "vm:${VMID}" --group ha-prod --state started \
      --max_restart 2 --max_relocate 1 \
      --comment "Cache/Monitoring VM ${VMID}"
    echo "[OK] VM ${VMID} → ha-prod"
done

echo ""
echo "═══ Configuration des groupes terminée ═══"
echo ""
echo "--- Groupes ---"
pvesh get /cluster/ha/groups
echo ""
echo "--- Ressources ---"
pvesh get /cluster/ha/resources
```

7.8.2 Règles d'affinité production

```
#!/bin/bash
# techflow-affinity-rules.sh
# Configuration des règles d'affinité pour la production TechFlow SAS

set -euo pipefail

echo "═══════════════════════════════════════════════"
echo " Règles d'affinité TechFlow SAS"
echo " Proxmox VE 9 — $(date +%Y-%m-%d)"
echo "═══════════════════════════════════════════════"

# ─── RÈGLE 1 : Anti-affinité web servers ──────────────────────
# Garantit qu'une panne nœud ne retire qu'1/3 de la capacité web maximum
pvesh create /cluster/ha/rules \
```

```
  --type anti-affinity \
  --vms "100,101,102" \
  --policy hard \
  --comment "Web cluster : 1 VM web par nœud physique maximum"
echo "[OK] Règle 1 : anti-affinité web (100,101,102) — HARD"

# ─── RÈGLE 2 : Anti-affinité load-balancers ──────────────────────────
# Les deux HAProxy ne doivent jamais être sur le même nœud
# Évite la perte totale du load-balancing lors d'une panne nœud
pvesh create /cluster/ha/rules \
  --type anti-affinity \
  --vms "103,108" \
  --policy hard \
  --comment "HAProxy LB : séparation obligatoire — no single point of failure"
echo "[OK] Règle 2 : anti-affinité LB (103,108) — HARD"

# ─── RÈGLE 3 : Anti-affinité PostgreSQL primary/replica ──────────────
# Primaire et replica sur nœuds différents
# Garantit la survie du cluster PostgreSQL lors d'une panne nœud
pvesh create /cluster/ha/rules \
  --type anti-affinity \
  --vms "106,107" \
  --policy hard \
  --comment "PostgreSQL : primaire et replica sur nœuds distincts"
echo "[OK] Règle 3 : anti-affinité PostgreSQL (106,107) — HARD"

# ─── RÈGLE 4 : Affinité app server + cache Redis ─────────────────────
# app-prod-01 (104) co-localisé avec Redis (109) pour minimiser la latence
# Politique soft : la bascule reste possible si nécessaire
pvesh create /cluster/ha/rules \
  --type affinity \
  --vms "104,109" \
  --policy soft \
  --comment "App-prod-01 + Redis : co-location préférée (latence < 0,1ms)"
echo "[OK] Règle 4 : affinité app+cache (104,109) — SOFT"

# ─── RÈGLE 5 : Anti-affinité app servers ─────────────────────────────
# Les deux instances Java sur nœuds différents pour répartir la charge
pvesh create /cluster/ha/rules \
  --type anti-affinity \
  --vms "104,105" \
  --policy soft \
  --comment "App servers : répartition préférentielle sur nœuds distincts"
echo "[OK] Règle 5 : anti-affinité app servers (104,105) — SOFT"

echo ""
echo "═══ Récapitulatif des règles d'affinité TechFlow ═══"
pvesh get /cluster/ha/rules

echo ""
echo "═══ Vérification du placement actuel ═══"
pvesh get /cluster/ha/resources --output-format json | \
python3 -c "
import sys, json
resources = json.load(sys.stdin)
print(f'{\"VM ID\":<12} {\"Nœud\":<20} {\"État\":<15} {\"Groupe\":<12}')
print('-' * 62)
for r in sorted(resources, key=lambda x: x.get('sid', '')):
    sid   = r.get('sid', '?')
    node  = r.get('node', 'non-assigné')
    state = r.get('state', '?')
    group = r.get('group', '?')
    print(f'{sid:<12} {node:<20} {state:<15} {group:<12}')
"
```

7.8.3 Test de bascule documenté

Ce test de bascule a été réalisé sur l'infrastructure TechFlow SAS le 17 mars 2026, avec l'accord de la direction technique, hors heures de bureau.

Préparation du test :

```
# 1. État initial du cluster
echo "═══ ÉTAT INITIAL — $(date) ═══"
pvecm status | grep -E "Quorate|Total votes|Nodes:"
ha-manager status
echo ""

# 2. Vérification des règles d'affinité respectées avant test
pvesh get /cluster/ha/resources --output-format json | \
python3 -c "
import sys, json
resources = json.load(sys.stdin)
placements = {r['sid']: r.get('node','?') for r in resources}
print('Placement initial :')
for sid, node in sorted(placements.items()):
    print(f'  {sid} → {node}')
"

# 3. Test de connectivité applicative initial
curl -s -o /dev/null -w "LB1 (103) : %{http_code}\n" \
  --connect-timeout 2 http://10.0.1.103/health
curl -s -o /dev/null -w "LB2 (108) : %{http_code}\n" \
  --connect-timeout 2 http://10.0.1.108/health
curl -s -o /dev/null -w "Web-01    : %{http_code}\n" \
  --connect-timeout 2 http://10.0.1.100/
```

Exécution du test — panne simulée sur pve-node2 :

```
# État initial sur pve-node2 : VM 101, 105, 107, 108
echo "VM sur pve-node2 avant test :"
ha-manager status | grep pve-node2

# Heure de début
T_START=$(date +%s)
echo "═══ SIMULATION PANNE pve-node2 — $(date +%H:%M:%S) ═══"

# Déclenchement de la panne simulée (arrêt brutal des services HA)
# Le watchdog expire ensuite et redémarre le nœud
ssh root@pve-node2 \
  "systemctl stop pve-ha-lrm pve-ha-crm; echo 'Services HA arrêtés'" || true

# Monitoring continu pendant la bascule
echo "═══ Surveillance depuis pve-node1 ═══"
for i in $(seq 1 36); do  # 3 minutes max
    ELAPSED=$((i * 5))
    echo "--- T+${ELAPSED}s ($(date +%H:%M:%S)) ---"
    ha-manager status 2>/dev/null | grep -E "lrm|vm:" || echo "(pas de données)"
    echo ""
    sleep 5
done
```

Résultats mesurés du test TechFlow :

```
═══ CHRONOLOGIE DE LA BASCULE — 17/03/2026 ═══

T+0s   (14:32:00) : Arrêt services HA sur pve-node2
T+18s  (14:32:18) : Corosync détecte la perte de pve-node2
T+22s  (14:32:22) : CRM marque pve-node2 "fence_required"
T+28s  (14:32:28) : Watchdog iTCO expire → redémarrage hardware pve-node2
T+32s  (14:32:32) : CRM confirme fencing pve-node2

VM hébergées sur pve-node2 avant panne :
  VM 101 (web-prod-02)  : recovery → started sur pve-node3 à T+68s
  VM 105 (app-prod-02)  : recovery → started sur pve-node3 à T+82s
```

```
  VM 107 (db-replica)   : recovery → started sur pve-node1 à T+88s
    → Note : règle anti-affinité 106/107 temporairement relâchée (node2 absent)
    → 106 sur node1, 107 sur node1 : violation temporaire acceptable
  VM 108 (lb-prod-02)   : recovery → started sur pve-node3 à T+58s
    → Règle anti-affinité LB respectée : 103 sur node1, 108 sur node3 ✓

T+88s  (14:33:28) : Toutes VM HA basculées et opérationnelles
T+180s (14:35:00) : pve-node2 termine redémarrage, rejoint le cluster

RTO mesuré    : 88 secondes (1 min 28s)
RTO objectif  : 300 secondes (5 min) → CONFORME ✓
RPO mesuré    : 0 (Ceph synchrone)
RPO objectif  : 0 → CONFORME ✓
```

Résultats de disponibilité applicative pendant la bascule :

```
# Test de disponibilité HTTP exécuté en parallèle (toutes les 2s)
# Résultats observés :

# 14:32:00  LB1:200  LB2:200   État normal
# 14:32:20  LB1:200  LB2:000   LB2 (108) en bascule
# 14:32:58  LB1:200  LB2:200   LB2 de retour sur pve-node3 (T+58s)

# Interruption LB2  : 58 secondes
# Interruption LB1  : aucune (100% disponible)
# Erreurs 5xx       : 0 détectées ✓
```

Procédure de retour à l'état nominal :

```
# Après retour de pve-node2 dans le cluster
echo "═══ RETOUR À L'ÉTAT NOMINAL — $(date +%H:%M:%S) ═══"

# Vérifier que pve-node2 est rejoint
pvecm nodes
ha-manager status | grep pve-node2

# Rééquilibrer le placement (nofailback=1 → migration manuelle requise)
# VM 101 : retour sur pve-node2 (anti-affinité web : node1/node2/node3)
ha-manager migrate vm:101 pve-node2
echo "Migration VM 101 → pve-node2 initiée"

# VM 105 : retour sur pve-node2
ha-manager migrate vm:105 pve-node2
echo "Migration VM 105 → pve-node2 initiée"

# VM 107 : retour sur pve-node2 (anti-affinité DB : 106 sur node1, 107 sur node2)
ha-manager migrate vm:107 pve-node2
echo "Migration VM 107 → pve-node2 initiée"

# VM 108 : retour sur pve-node2 (anti-affinité LB : 103 sur node1, 108 sur node2)
ha-manager migrate vm:108 pve-node2
echo "Migration VM 108 → pve-node2 initiée"

# Surveiller les migrations
watch -n 2 ha-manager status

# Vérification finale du placement
echo "═══ ÉTAT FINAL APRÈS RÉÉQUILIBRAGE ═══"
pvesh get /cluster/ha/resources --output-format json | \
python3 -c "
import sys, json
resources = json.load(sys.stdin)
print(f'{\"VM ID\":<12} {\"Nœud\":<20} {\"État\":<15}')
print('-' * 48)
for r in sorted(resources, key=lambda x: x.get('sid', '')):
    print(f'{r[\"sid\"]:<12} {r.get(\"node\",\"??\"):<20} {r.get(\"state\",\"?
\"):<15}')
"
```

Rapport de test synthétique TechFlow SAS :

```
═══════════════════════════════════════════════════════════
  RAPPORT DE TEST DE BASCULE HA
  TechFlow SAS — Cluster Proxmox VE 9
  Date : 17 mars 2026 — 14h30 à 15h15
═══════════════════════════════════════════════════════════

PÉRIMÈTRE DU TEST
  Nœud simulé en panne  : pve-node2
  VM affectées          : 101, 105, 107, 108
  VM non affectées      : 100, 102, 103, 104, 106, 109, 110
  Mécanisme de fencing  : Watchdog Intel TCO iTCO_wdt

RÉSULTATS
  RTO mesuré            : 88 secondes
  RTO objectif          : 300 secondes (5 min)
  Résultat RTO          : CONFORME ✓

  RPO mesuré            : 0 (stockage Ceph synchrone)
  RPO objectif          : 0
  Résultat RPO          : CONFORME ✓

  Fencing effectif      : 28 secondes → dans les normes ✓
  Corruption données    : Aucune ✓

RÈGLES D'AFFINITÉ POST-BASCULE
  Anti-affinité LB      : RESPECTÉE ✓ (103→node1, 108→node3)
  Anti-affinité Web     : PARTIELLE ⚠ (101 sur node3, temporaire OK)
  Anti-affinité DB      : VIOLÉE TEMPORAIREMENT ⚠ (106+107 sur node1)
    → Repositionnement manuel effectué après retour node2
  Affinité App+Cache    : MAINTENUE ✓ (104+109 ensemble sur node1)
  Anti-affinité App     : RESPECTÉE ✓ (104 node1, 105 node3)

DISPONIBILITÉ SERVICES
  HAProxy LB1 (103)     : 100 % disponible ✓
  HAProxy LB2 (108)     : 58s indisponible → 99,0 % sur fenêtre test
  Application Java      : 0 erreur 5xx ✓
  Base de données       : Réplication interrompue 88s, reprise auto ✓
  Monitoring (110)      : Non affecté (sur node3) ✓

RECOMMANDATIONS ISSUES DU TEST
  1. Envisager un 3e LB (VM 113) pour éliminer l'interruption LB2 lors bascule
  2. Automatiser la vérification des règles d'affinité post-bascule via script
  3. Documenter la procédure de retour nominal dans le runbook OPS
  4. Configurer une alerte Prometheus si une règle d'affinité hard est violée
  5. Tester le scénario de panne double (deux nœuds simultanément)

VALIDATION
  Responsable test      : Équipe Infrastructure TechFlow SAS
  Superviseur           : Directeur Technique
  Prochain test planifié: Juin 2026 — simulation panne pve-node1
═══════════════════════════════════════════════════════════
```

Résumé du chapitre

Ce chapitre a couvert l'ensemble du cycle de vie de la haute disponibilité dans Proxmox VE 9 :

- **Concepts fondamentaux** : RTO/RPO et niveaux de disponibilité, architecture CRM/LRM, rôle de Corosync, états des ressources HA et leurs transitions
- **Prérequis** : cluster Corosync avec quorum (3 nœuds minimum), stockage partagé Ceph ou NFS, fencing obligatoire et validé avant toute mise en production
- **Fencing** : watchdog Intel TCO comme mécanisme primaire, service `watchdog-mux`, fencing IPMI en complément, protocole de test de validation
- **Groupes HA et ressources** : organisation en groupes typés (ha-prod, ha-db, ha-lb), politiques `nofailback` et `restricted`, paramètres `max_restart` et `max_relocate`

- **Règles d'affinité (Proxmox VE 9)** : nouvelle API `/cluster/ha/rules`, affinité positive (co-location), anti-affinité (séparation), politiques hard/soft, gestion via `pvesh`
- **Maintenance** : mode maintenance avec évacuation automatique des VM, rolling updates sans interruption de service
- **Tests** : protocole de test de bascule contrôlé, mesure du RTO réel, vérification des règles d'affinité, rapport documenté
- **Lab TechFlow SAS** : 11 VM protégées, 3 groupes HA, 5 règles d'affinité, test de bascule documenté avec RTO de 88 secondes

La combinaison HA + règles d'anti-affinité constitue le pattern de référence pour les applications critiques : elle garantit la reprise automatique en cas de panne **et** la répartition des instances sur des hôtes physiquement distincts, maximisant ainsi la disponibilité réelle du service.

> ⚠ **AVERTISSEMENT**
> La haute disponibilité Proxmox ne remplace pas une stratégie de sauvegarde complète. Le HA protège contre les pannes matérielles non planifiées mais ne protège pas contre la corruption logicielle, les erreurs humaines ou les suppressions accidentelles. Proxmox Backup Server (PBS) doit être utilisé en complément du HA pour une protection complète des données.

8

Ceph : stockage hyper-convergé

Ceph est aujourd'hui la solution de stockage distribué de référence dans l'écosystème open source. Son intégration native dans Proxmox VE permet de bâtir des infrastructures hyper-convergées où calcul et stockage partagent les mêmes serveurs physiques, sans point de défaillance unique, sans contrôleur propriétaire et sans licence coûteuse.

Ce chapitre couvre l'architecture interne de Ceph, son déploiement via Proxmox VE 9, l'optimisation des performances et les opérations de maintenance du quotidien. Le lab de fin de chapitre met en oeuvre un cluster de trois noeuds pour TechFlow SAS avec des disques NVMe et un VLAN dédié.

8.1 Architecture Ceph

8.1.1 Composants : MON, OSD, MGR, MDS, RGW

Ceph est conçu autour de plusieurs démons spécialisés qui coopèrent pour offrir un stockage objet, bloc et fichier unifié. Comprendre le rôle de chaque composant est indispensable avant tout déploiement.

Monitor (MON)

Le Monitor maintient la carte authoritative du cluster : la cluster map. Cette carte est en réalité un ensemble de cartes spécialisées (monitor map, OSD map, CRUSH map, MDS map, manager map). Les MONs forment un quorum basé sur l'algorithme Paxos. Un cluster Ceph doit toujours disposer d'un nombre impair de MONs (1, 3, 5...) pour que le quorum puisse être atteint.

- Minimum recommandé en production : 3 MONs
- Les MONs ne stockent pas de données utilisateur
- Ils valident toutes les modifications de l'état du cluster
- Chaque client s'y connecte au démarrage pour récupérer les cartes

Object Storage Daemon (OSD)

L'OSD est le cheval de trait du cluster. Chaque OSD gère un disque physique (ou une partition) et est responsable de :

- Stocker les objets localement (via BlueStore depuis Luminous)
- Répliquer les données vers les OSDs pairs
- Participer au rebalancing lors de changements topologiques
- Effectuer le scrubbing et le deep-scrubbing pour détecter les corruptions
- Rapporter son état de santé aux MONs

La règle de base : un OSD par disque physique. Jamais deux OSDs sur le même disque.

Manager (MGR)

Introduit avec Luminous, le MGR collecte les métriques de performance et expose des modules optionnels :

- Dashboard web intégré (activé par défaut dans Proxmox)
- Module `prometheus` pour l'export de métriques
- Module `pg_autoscaler` pour l'ajustement automatique des Placement Groups
- Module `balancer` pour l'optimisation de la distribution des données
- Module `crash` pour la gestion des rapports de crash

En production, déployez au moins deux MGRs (un actif, un en standby).

Metadata Server (MDS)

Le MDS est requis uniquement pour CephFS (le système de fichiers distribué). Il gère les métadonnées du système de fichiers (arborescence, permissions, inodes) tandis que les données réelles sont stockées dans des pools Ceph ordinaires. Un MDS actif et un MDS en standby constituent la configuration minimale recommandée.

RADOS Gateway (RGW)

Le RGW expose une API compatible Amazon S3 et OpenStack Swift sur le dessus du cluster Ceph. Il transforme Ceph en stockage objet accessible via HTTP/HTTPS. Particulièrement utile pour :

- Sauvegardes S3 depuis Proxmox Backup Server
- Hébergement d'objets applicatifs
- Compatibilité avec les outils AWS CLI

8.1.2 CRUSH map et placement des données

CRUSH (Controlled Replication Under Scalable Hashing) est l'algorithme qui détermine où chaque objet est stocké dans le cluster. C'est le coeur de l'architecture de Ceph et ce qui lui permet de fonctionner sans métadonnées centralisées pour la localisation des données.

Principe de fonctionnement

Au lieu de maintenir une table de correspondance objet → emplacement (ce qui ne passerait pas à l'échelle), CRUSH calcule de manière déterministe l'emplacement de chaque objet à partir de :

1. L'identifiant de l'objet (PGID)
2. La CRUSH map (topologie du cluster)
3. La règle CRUSH du pool concerné

Ce calcul est identique sur tous les noeuds : un client sait toujours où écrire sans interroger un serveur de métadonnées central.

Structure de la CRUSH map

La CRUSH map représente la topologie physique sous forme d'arbre hiérarchique :

```
root
├── datacenter-paris
│   ├── rack-A
│   │   ├── host-pve1
│   │   │   ├── osd.0 (weight 1.0, NVMe 1TB)
│   │   │   ├── osd.1 (weight 1.0, NVMe 1TB)
│   │   │   └── osd.2 (weight 1.0, NVMe 1TB)
│   │   └── host-pve2
│   │       ├── osd.3
│   │       ├── osd.4
│   │       └── osd.5
│   └── rack-B
│       └── host-pve3
│           ├── osd.6
│           ├── osd.7
│           └── osd.8
```

Règles CRUSH

Les règles définissent comment CRUSH traverse l'arbre pour sélectionner les OSDs cibles. Une règle typique pour la réplication 3 au niveau host :

```
# Afficher les règles CRUSH existantes
ceph osd crush rule ls

# Afficher le détail d'une règle
ceph osd crush rule dump replicated_rule

# Créer une règle personnalisée avec domaine de défaillance au niveau rack
ceph osd crush rule create-replicated rack-replicated default rack
```

Poids CRUSH

Le poids d'un OSD est proportionnel à sa capacité. Un disque de 2 To aura un poids de 2.0, un disque de 1 To un poids de 1.0. Ceph distribue les données proportionnellement aux poids.

```
# Vérifier les poids des OSDs
ceph osd tree

# Modifier le poids d'un OSD
ceph osd crush reweight osd.5 2.0
```

8.1.3 Pools, PGs et réplication

Placement Groups (PG)

Les Placement Groups sont l'unité de distribution de Ceph. Les objets sont regroupés en PGs, et ce sont les PGs qui sont distribués sur les OSDs par CRUSH. Ce mécanisme d'indirection permet à Ceph de gérer des milliards d'objets avec un nombre raisonnable de PGs (quelques centaines à quelques milliers).

Chaque PG possède : - Un identifiant unique (pool_id.pg_id, ex : `1.a3`) - Un OSD primaire (qui gère les écritures) - Des OSDs secondaires (répliques) - Un état (active+clean, degraded, recovering, etc.)

Calcul du nombre de PGs

Le nombre de PGs par pool doit être calculé soigneusement. Trop peu de PGs entraîne un déséquilibre ; trop de PGs consomme inutilement de la mémoire (environ 10 Mo par OSD par 1000 PGs).

La formule de base :

```
PGs = (OSDs × 100) / facteur_réplication
```

Arrondir à la puissance de 2 supérieure. Avec 9 OSDs et réplication 3 :

```
PGs = (9 × 100) / 3 = 300 → arrondi à 256 ou 512
```

Depuis Nautilus, le module `pg_autoscaler` ajuste automatiquement le nombre de PGs. Il est activé par défaut dans Proxmox VE 9.

```
# Activer le pg_autoscaler globalement
ceph config set global osd_pool_default_pg_autoscale_mode on

# Vérifier l'autoscaling d'un pool
ceph osd pool autoscale-status
```

Modes de réplication

Mode	Principe	Overhead stockage	Performances lecture	Performances écriture	Tolérance pannes
Réplication 2	2 copies identiques	2x	Bonne	Bonne	1 OSD
Réplication 3	3 copies identiques	3x	Excellente	Bonne	2 OSDs
Erasure Coding 4+2	4 fragments + 2 parité	1.5x	Moyenne	Moins bonne	2 OSDs
Erasure Coding 8+3	8 fragments + 3 parité	1.375x	Bonne (large)	Pénalisante	3 OSDs

États des PGs

```
# Vue synthétique de l'état des PGs
ceph pg stat

# Détail par PG
ceph pg dump

# PGs dégradés ou en erreur
ceph health detail
```

8.1.4 Ceph Squid 19.2 : nouveautés

Ceph Squid (version 19.2, inclus dans Proxmox VE 9) apporte plusieurs améliorations significatives.

SMB Gateway

Squid introduit un module MGR permettant d'exposer CephFS via le protocole SMB/CIFS directement depuis le cluster, sans serveur Samba externe. La configuration est entièrement gérée via l'API Ceph.

```
# Activer le module SMB
ceph mgr module enable smb

# Vérifier la disponibilité
ceph smb cluster ls
```

Amélioration du pg_autoscaler

Le pg_autoscaler de Squid prend désormais en compte les projections de croissance et permet de définir des objectifs de taille de PG cible (`target_size_bytes`) avec une marge de tolérance configurable.

RBD persistent read cache

Nouvelle fonctionnalité de cache en lecture persistant pour les images RBD, stocké localement sur des SSD NVMe rapides. Réduit la latence pour les workloads à haute fréquence de lecture.

```
# Configurer le cache persistant RBD
rbd config image set vm-pool/vm-100-disk-0 \
  rbd_persistent_cache_mode rwl
rbd config image set vm-pool/vm-100-disk-0 \
  rbd_persistent_cache_path /dev/nvme0n1p1
```

Crimson OSD (preview)

Crimson est la réécriture complète de l'OSD en utilisant le framework SeaStar (async/await natif). En preview dans Squid, il promet une latence réduite de 30 à 50 % pour les workloads NVMe. Non recommandé en production actuellement.

Améliorations BlueStore

- Compression zstd niveau 1 maintenant recommandée (meilleur ratio performance/compression que lz4)
- Déduplication en preview pour les pools RBD
- Amélioration de la gestion des métadonnées RocksDB

8.2 Déploiement Ceph avec Proxmox

8.2.1 Prérequis matériels et réseau

Avant de déployer Ceph, une planification minutieuse est nécessaire. Les erreurs de conception à ce stade sont coûteuses à corriger en production.

Prérequis matériels

Configuration minimale recommandée par noeud :

Composant	Minimum	Recommandé production
CPU	4 coeurs	8+ coeurs (les OSDs consomment des coeurs)
RAM	16 Go	64 Go (4 Go par OSD minimum + hyperviseur)
Disques OSD	HDD 1 To	NVMe 1-4 To
Disques WAL/DB	Optionnel	NVMe dédié si OSDs HDD
Réseau public	1 Gbps	10 Gbps
Réseau cluster	1 Gbps	10 Gbps (dédié)

Dimensionnement RAM pour les OSDs

La consommation mémoire de BlueStore est principalement due au cache RocksDB et au cache de données :

```
RAM par OSD = 4 Go (minimum absolu)
RAM par OSD = 5-6 Go (recommandé avec NVMe)
RAM par OSD = 2-3 Go (acceptable avec HDD)
```

Pour 9 OSDs NVMe : 9 x 6 Go = 54 Go réservés pour Ceph. Avec l'hyperviseur, 128 Go de RAM par noeud est une bonne base.

Architecture réseau requise

Ceph utilise deux réseaux distincts :

- **Réseau public** (client network) : communications entre clients (Proxmox, RBD, etc.) et les OSDs/MONs
- **Réseau cluster** (replication network) : trafic de réplication entre OSDs (rebalancing, réplication, recovery)

La séparation est fortement recommandée : le trafic de réplication peut saturer un réseau 10 Gbps lors d'un rebalancing ou d'une recovery après panne.

> ⚠ **AVERTISSEMENT**
> Ne jamais utiliser le même réseau pour le trafic Ceph cluster et le trafic de migration live des VM. Un rebalancing Ceph intensif peut provoquer des timeouts de migration et des interruptions de service.

Vérification préalable des disques

```
# Lister les disques disponibles sur un noeud
lsblk -o NAME,SIZE,TYPE,MOUNTPOINT,MODEL

# Vérifier qu'un disque est vierge (aucune partition, LVM, etc.)
pveceph osd create /dev/nvme1n1 --dry-run

# Effacer un disque proprement avant utilisation
wipefs -a /dev/nvme1n1
sgdisk --zap-all /dev/nvme1n1
```

8.2.2 Installation via l'interface web

Proxmox VE 9 intègre un assistant d'installation Ceph accessible depuis l'interface web.

Étape 1 : Accéder à l'installation Ceph

Dans l'interface Proxmox, sélectionnez un noeud dans l'arborescence, puis naviguez vers : `Noeud → Ceph → Installation`

Si Ceph n'est pas encore installé, l'interface propose un bouton "Install Ceph". Cliquez dessus.

Étape 2 : Choisir la version

Proxmox VE 9 propose Ceph Squid (19.2) par défaut. Sélectionnez la version et le dépôt (production recommandé).

Étape 3 : Paramètres réseau

L'assistant vous demande : - **Public Network** : réseau client (ex : `10.10.1.0/24`) - **Cluster Network** : réseau de réplication (ex : `10.10.2.0/24`)

Alternativement via la ligne de commande :

```
# Sur chaque noeud Proxmox, installer les paquets Ceph
pveceph install --version squid

# Vérifier l'installation
ceph --version
# ceph version 19.2.0 (hash) squid (stable)
```

8.2.3 Création du cluster Ceph

Le cluster Ceph doit être initialisé sur le premier noeud. Les autres noeuds s'y joindront ensuite.

Initialisation sur le premier noeud

```
# Initialiser Ceph sur pve1
pveceph init \
  --network 10.10.1.0/24 \
  --cluster-network 10.10.2.0/24

# Vérifier la configuration générée
cat /etc/ceph/ceph.conf
```

Le fichier `/etc/ceph/ceph.conf` généré ressemblera à :

```
[global]
        auth_cluster_required = cephx
        auth_service_required = cephx
```

```
        auth_client_required = cephx
        keyring = /etc/pve/priv/$cluster.$name.keyring
        log_to_syslog = false
        err_to_syslog = false
        cephx_sign_messages = false
        osd_pool_default_size = 3
        osd_pool_default_min_size = 2

[osd]
        osd_journal_size = 5120

[mds]

[client]
```

Configuration des adresses IP par noeud

```
# Sur pve1 : configurer l'adresse publique et cluster
cat >> /etc/ceph/ceph.conf << 'EOF'

[osd.0]
        host = pve1
        public addr = 10.10.1.11
        cluster addr = 10.10.2.11

[osd.1]
        host = pve1
        public addr = 10.10.1.11
        cluster addr = 10.10.2.11
EOF
```

8.2.4 Ajout des OSDs

Les OSDs sont le composant qui nécessite le plus d'attention. Chaque OSD correspond à un disque physique dédié.

Via l'interface web

Dans `Noeud → Ceph → OSD`, cliquez sur "Create OSD". Sélectionnez le disque dans la liste déroulante. L'interface propose optionnellement des disques séparés pour le WAL (Write-Ahead Log) et le DB (RocksDB metadata).

Via la ligne de commande

```
# Ajouter un OSD simple (BlueStore tout-en-un)
pveceph osd create /dev/nvme1n1

# Ajouter un OSD avec WAL et DB sur un NVMe dédié
pveceph osd create /dev/sda \
  --wal-dev /dev/nvme0n1 \
  --db-dev /dev/nvme0n1

# Vérifier la liste des OSDs
ceph osd tree

# Vérifier l'état
ceph osd stat
```

BlueStore : structure interne d'un OSD

BlueStore (le backend de stockage par défaut depuis Luminous) gère directement le disque bloc sans système de fichiers intermédiaire. Il organise l'espace en :

- **BlueStore data** : données des objets (~95% de l'espace)
- **RocksDB** : métadonnées, carte des blocs (peut être externalisé sur SSD)
- **WAL** : journal des écritures RocksDB (peut être externalisé sur SSD/NVMe rapide)

L'externalisation du WAL et du DB sur un NVMe rapide améliore significativement les performances des OSDs HDD ou SATA SSD.

```
# Vérifier le type de backend d'un OSD
ceph osd metadata osd.0 | grep osd_objectstore
```

```
# "osd_objectstore": "bluestore"

# Vérifier la santé BlueStore
ceph tell osd.0 bluestore stats
```

Ajout des OSDs sur tous les noeuds

```
# Script d'ajout en masse des OSDs (à exécuter sur chaque noeud)
for DISK in /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1; do
    pveceph osd create $DISK
    echo "OSD créé sur $DISK"
    sleep 5
done

# Surveiller le rebalancing après chaque ajout
watch ceph status
```

8.2.5 Création des MONs et MGRs

Création des MONs

Le MON initial est créé lors de `pveceph init`. Les MONs supplémentaires sont ajoutés sur les autres noeuds.

```
# Sur pve1 (déjà fait lors de l'init)
# Sur pve2 : ajouter un MON
pveceph mon create

# Sur pve3 : ajouter un MON
pveceph mon create

# Vérifier le quorum MON
ceph mon stat
# e3: 3 mons at
{pve1=10.10.1.11:6789/0,pve2=10.10.1.12:6789/0,pve3=10.10.1.13:6789/0}
# election epoch 8, quorum 0,1,2 pve1,pve2,pve3

# Détail du quorum
ceph quorum_status --format json-pretty
```

Création des MGRs

```
# Sur pve1 (premier MGR, souvent créé automatiquement)
pveceph mgr create

# Sur pve2 (MGR standby)
pveceph mgr create

# Vérifier les MGRs
ceph mgr stat
# active: pve1, standbys: [pve2]

# Activer le dashboard
ceph mgr module enable dashboard

# Configurer le certificat SSL pour le dashboard
ceph dashboard create-self-signed-cert

# Créer un utilisateur admin pour le dashboard
ceph dashboard ac-user-create admin --roles administrator -i /tmp/pass

# Vérifier l'URL du dashboard
ceph mgr services
# {
#     "dashboard": "https://10.10.1.11:8443/"
# }
```

Activation des modules utiles

```
# Activer le pg_autoscaler
ceph mgr module enable pg_autoscaler
```

```
ceph config set global osd_pool_default_pg_autoscale_mode on

# Activer le balancer (mode upmap recommandé)
ceph mgr module enable balancer
ceph balancer mode upmap
ceph balancer on

# Activer l'export Prometheus
ceph mgr module enable prometheus
# Métriques disponibles sur http://10.10.1.11:9283/metrics

# Activer les alertes
ceph mgr module enable alerts
```

8.3 Pools et stockage

8.3.1 Pool répliqué vs erasure coding

Le choix du type de pool est l'une des décisions les plus importantes dans la conception d'un cluster Ceph.

Pool répliqué

Dans un pool répliqué, chaque objet est stocké N fois (N = taille de réplication). Les répliques sont placées sur des OSDs différents selon les règles CRUSH.

Avantages : - Lectures parallèles possibles sur toutes les répliques - Faible latence en écriture (une seule opération x N) - Support complet de RBD (images bloc pour VM) - Snapshots efficaces

Inconvénients : - Overhead de stockage élevé (3x pour réplication 3)

Pool Erasure Coding (EC)

Le pool EC encode chaque objet en K fragments de données et M fragments de parité (profil k+m). Pour retrouver l'objet, K fragments suffisent parmi les K+M disponibles.

Avantages : - Overhead de stockage réduit (ex : 4+2 = 1.5x) - Adapté aux données froides (objets, backups)

Inconvénients : - Latence d'écriture plus élevée (opérations de calcul EC) - RBD nécessite un pool de cache répliqué pour les écritures partielles - Pas de snapshot natif sur les pools EC

Tableau comparatif complet

Critère	Réplication 3	EC 4+2	EC 8+3
Overhead stockage	3.0x	1.5x	1.375x
Noeuds min pour tolérance	3	6	11
OSDs défaillants tolérés	2	2	3
Latence écriture	Faible	Moyenne	Élevée
Latence lecture séquentielle	Faible	Faible	Faible
Support RBD direct	Oui	Non (avec cache)	Non
Support RGW	Oui	Oui	Oui
Support CephFS	Données uniquement	Données uniquement	Données uniquement
Snapshots RBD	Oui	Non	Non
Idéal pour	VM, bases de données	Objets, archives	Big data, backup

⚠ AVERTISSEMENT

L'erasure coding ne peut pas être utilisé directement comme stockage RBD pour les machines virtuelles Proxmox. Un pool de cache répliqué (cache tiering) est nécessaire, ce qui complexifie l'architecture. Pour les VM, utilisez toujours un pool répliqué.

8.3.2 Création d'un pool RBD pour les VM

Le pool RBD est le type de pool utilisé pour les disques de machines virtuelles dans Proxmox. Chaque disque de VM est une image RBD (RADOS Block Device).

```
# Créer un pool répliqué pour les VM
ceph osd pool create vm-pool 128 128 replicated

# Définir la taille de réplication (3 copies)
ceph osd pool set vm-pool size 3
ceph osd pool set vm-pool min_size 2

# Initialiser le pool comme pool RBD
rbd pool init vm-pool

# Définir l'application du pool
ceph osd pool application enable vm-pool rbd

# Vérifier la création
ceph osd pool ls detail

# Activer l'autoscaling pour ce pool
ceph osd pool set vm-pool pg_autoscale_mode on

# Définir la taille cible du pool (optionnel, aide l'autoscaler)
ceph osd pool set vm-pool target_size_bytes 2T
```

Création d'une image RBD manuellement

```
# Créer une image RBD de 50 Go
rbd create vm-pool/test-disk --size 51200

# Lister les images
rbd ls vm-pool

# Informations sur une image
rbd info vm-pool/test-disk

# Supprimer une image
rbd rm vm-pool/test-disk
```

Fonctionnalités RBD recommandées

```
# Désactiver les fonctionnalités non supportées par certains noyaux
rbd feature disable vm-pool/vm-100-disk-0 \
  deep-flatten fast-diff object-map exclusive-lock

# Activer les fonctionnalités recommandées pour Proxmox
rbd feature enable vm-pool/vm-100-disk-0 \
  exclusive-lock object-map fast-diff deep-flatten
```

8.3.3 CephFS : système de fichiers distribué

CephFS transforme Ceph en système de fichiers POSIX distribué, accessible en montage NFS-like ou via le client noyau Linux.

Architecture CephFS

CephFS utilise deux pools Ceph : - **Pool de métadonnées** : géré par le MDS, stocke l'arborescence et les inodes - **Pool de données** : stocke le contenu des fichiers, peut être un pool EC

```
# Créer les pools pour CephFS
ceph osd pool create cephfs-meta 32 32 replicated
ceph osd pool create cephfs-data 128 128 replicated

# Initialiser CephFS
ceph fs new cephfs cephfs-meta cephfs-data

# Vérifier
ceph fs ls
```

```
# name: cephfs, metadata pool: cephfs-meta, data pools: [cephfs-data]

# Créer les MDS
pveceph mds create
# (à répéter sur chaque noeud pour avoir 1 actif + 1 standby minimum)

# Vérifier l'état des MDS
ceph mds stat
# cephfs:1 {0=pve1=up:active} 2 up:standby
```

Montage de CephFS

```
# Récupérer la clé d'authentification admin
ceph auth get-key client.admin

# Monter CephFS via le client noyau
mount -t ceph 10.10.1.11:/ /mnt/cephfs \
  -o name=admin,secret=<clé>

# Montage permanent dans /etc/fstab
echo "10.10.1.11:/ /mnt/cephfs ceph \
  name=admin,secretfile=/etc/ceph/admin.secret,_netdev 0 0" \
  >> /etc/fstab

# Montage via FUSE (sans module noyau)
ceph-fuse /mnt/cephfs -o allow_other
```

Quotas CephFS

```
# Définir un quota sur un répertoire
setfattr -n ceph.quota.max_bytes \
  -v 107374182400 /mnt/cephfs/vm-isos
# Quota de 100 Go

setfattr -n ceph.quota.max_files \
  -v 10000 /mnt/cephfs/vm-isos

# Vérifier les quotas
getfattr -n ceph.quota.max_bytes /mnt/cephfs/vm-isos
```

8.3.4 Rados Gateway (S3/Swift)

Le RADOS Gateway expose Ceph comme stockage objet compatible S3 et Swift.

Installation et configuration

```
# Créer les pools RGW (optionnel, créés automatiquement au démarrage)
ceph osd pool create .rgw.root 8 8 replicated
ceph osd pool create default.rgw.control 8 8 replicated
ceph osd pool create default.rgw.meta 8 8 replicated
ceph osd pool create default.rgw.log 8 8 replicated
ceph osd pool create default.rgw.buckets.index 32 32 replicated
ceph osd pool create default.rgw.buckets.data 128 128 replicated

# Créer le daemon RGW
pveceph rgw create --id rgw1

# Vérifier
ceph -s | grep rgw
```

Création d'un utilisateur S3

```
# Créer un utilisateur S3
radosgw-admin user create \
  --uid=techflow-backup \
  --display-name="TechFlow Backup" \
  --email=backup@techflow.fr

# Récupérer les clés d'accès
radosgw-admin user info --uid=techflow-backup | \
  python3 -c "import sys,json; d=json.load(sys.stdin); \
```

```
  keys=d['keys'][0]; print('AK:',keys['access_key']); \
  print('SK:',keys['secret_key'])"

# Créer un bucket
aws --endpoint-url http://10.10.1.11:7480 s3 mb s3://proxmox-backup

# Tester l'upload
aws --endpoint-url http://10.10.1.11:7480 \
  s3 cp /etc/ceph/ceph.conf s3://proxmox-backup/
```

Intégration avec Proxmox Backup Server

```
# Dans PBS : configurer le datastore S3
# /etc/proxmox-backup/datastore.cfg
datastore: ceph-s3
  path /mnt/ceph-s3
  comment "Stockage backup sur Ceph RGW"
```

8.4 Intégration Proxmox-Ceph

8.4.1 Stockage RBD dans Proxmox

Pour que Proxmox utilise Ceph comme stockage pour les VM, il faut déclarer le stockage RBD dans la configuration Proxmox.

Via l'interface web

Naviguez vers `Datacenter → Storage → Add → RBD`. Renseignez : - ID : `ceph-rbd` (ou le nom souhaité) - Monitor Hosts : `10.10.1.11 10.10.1.12 10.10.1.13` - Pool : `vm-pool` - Username : `admin` - Content : `Disk image, Container`

Via la configuration

```
# Ajouter le stockage RBD dans /etc/pve/storage.cfg
pvesm add rbd ceph-rbd \
  --monhost "10.10.1.11 10.10.1.12 10.10.1.13" \
  --pool vm-pool \
  --username admin \
  --content images

# Vérifier
pvesm status

# Lister les images du pool
pvesm list ceph-rbd
```

Création d'une VM utilisant Ceph RBD

```
# Créer une VM avec disque sur Ceph
qm create 200 \
  --name test-ceph-vm \
  --memory 4096 \
  --cores 4 \
  --net0 virtio,bridge=vmbr0

# Ajouter un disque sur le stockage Ceph
qm set 200 \
  --scsi0 ceph-rbd:32 \
  --scsihw virtio-scsi-pci

# Vérifier que le disque RBD a bien été créé
rbd ls vm-pool
# vm-200-disk-0

# Informations sur le disque
rbd info vm-pool/vm-200-disk-0
```

8.4.2 Migration live de VM sur RBD

La migration live (sans interruption de service) des VM est possible entre noeuds Proxmox lorsque le stockage est partagé via Ceph RBD.

Migration live standard

```
# Migrer la VM 200 de pve1 vers pve2 à chaud
qm migrate 200 pve2 --online

# Avec paramètres avancés
qm migrate 200 pve2 \
  --online \
  --bwlimit 0 \
  --force
```

Migration avec copie du stockage (storage migration)

Lorsqu'une VM est sur un stockage local et doit être migrée vers Ceph :

```
# Migrer une VM depuis stockage local vers Ceph RBD
qm migrate 101 pve2 \
  --online \
  --targetstorage ceph-rbd

# Suivre la progression
tail -f /var/log/pve/tasks/active
```

Comportement lors de la migration

La migration live avec Ceph RBD fonctionne de la manière suivante :

1. Proxmox établit une connexion entre source et destination
2. La mémoire est copiée itérativement (dirty pages tracking)
3. Quand la quantité de mémoire restante à copier est suffisamment faible, la VM est suspendue brièvement
4. La connexion réseau virtuelle est basculée
5. La VM reprend sur le noeud destination

Avec Ceph RBD, le disque est déjà accessible depuis tous les noeuds : il n'y a pas de copie de disque, ce qui rend la migration quasi-instantanée.

8.4.3 Snapshots Ceph et cohérence

RBD supporte les snapshots copy-on-write. Un snapshot capture l'état d'une image RBD à un instant T sans dupliquer immédiatement les données.

Snapshots RBD via Proxmox

```
# Créer un snapshot d'une VM (via Proxmox)
qm snapshot 200 snap-avant-mise-a-jour \
  --description "Avant mise à jour kernel 6.8"

# Lister les snapshots
qm listsnapshot 200

# Restaurer un snapshot
qm rollback 200 snap-avant-mise-a-jour

# Supprimer un snapshot
qm delsnapshot 200 snap-avant-mise-a-jour
```

Snapshots RBD natifs

```
# Créer un snapshot RBD directement
rbd snap create vm-pool/vm-200-disk-0@snap-20260317

# Lister les snapshots d'une image
rbd snap ls vm-pool/vm-200-disk-0

# Protéger un snapshot (empêche la suppression accidentelle)
rbd snap protect vm-pool/vm-200-disk-0@snap-20260317
```

```
# Cloner depuis un snapshot (copy-on-write)
rbd clone \
  vm-pool/vm-200-disk-0@snap-20260317 \
  vm-pool/vm-200-disk-0-clone

# Aplatir un clone (copie complète, indépendance du parent)
rbd flatten vm-pool/vm-200-disk-0-clone

# Supprimer un snapshot
rbd snap unprotect vm-pool/vm-200-disk-0@snap-20260317
rbd snap rm vm-pool/vm-200-disk-0@snap-20260317
```

Cohérence des snapshots

> ⚠ **AVERTISSEMENT**
> Un snapshot RBD pris sur une VM en cours d'exécution sans quiesce des I/O n'est pas cohérent au niveau applicatif (crash-consistent seulement). Pour des snapshots cohérents avec les applications, utilisez les agents QEMU guest agent avec `--vmstate` ou arrêtez temporairement la VM.

```
# Snapshot avec freeze des I/O via l'agent QEMU
qm agent 200 fsfreeze-freeze
rbd snap create vm-pool/vm-200-disk-0@snap-coherent-$(date +%Y%m%d)
qm agent 200 fsfreeze-thaw
```

8.4.4 CephFS comme stockage ISO/backup

CephFS peut être utilisé comme stockage partagé pour les ISOs, templates et backups dans Proxmox.

Configuration du stockage CephFS dans Proxmox

```
# Ajouter CephFS comme stockage Proxmox
pvesm add cephfs cephfs-isos \
  --monhost "10.10.1.11 10.10.1.12 10.10.1.13" \
  --path /mnt/pve/cephfs-isos \
  --content iso,vztmpl,backup \
  --username admin

# Vérifier le montage
pvesm status | grep cephfs-isos
mount | grep cephfs

# Lister le contenu
pvesm list cephfs-isos
```

Organisation recommandée des répertoires CephFS

```
# Créer une structure organisée sur CephFS
mkdir -p /mnt/pve/cephfs-isos/template/iso
mkdir -p /mnt/pve/cephfs-isos/template/cache
mkdir -p /mnt/pve/cephfs-isos/dump

# Copier les ISOs
cp /var/lib/vz/template/iso/debian-12.iso \
   /mnt/pve/cephfs-isos/template/iso/
```

8.5 Performance et tuning

8.5.1 Séparation réseau public/cluster

La séparation des réseaux est la première optimisation à mettre en oeuvre, avant même de toucher aux paramètres de configuration.

Vérification de la configuration réseau Ceph

```
# Vérifier les réseaux configurés
ceph config get mon public_network
ceph config get osd cluster_network

# Modifier les réseaux si nécessaire
```

```
ceph config set global public_network 10.10.1.0/24
ceph config set global cluster_network 10.10.2.0/24
```

Configuration des interfaces réseau

Sur chaque noeud Proxmox, les interfaces doivent être configurées dans `/etc/network/interfaces` :

```
# Interface réseau Ceph public (10 Gbps)
auto ens3f0
iface ens3f0 inet static
    address 10.10.1.11/24
    mtu 9000

# Interface réseau Ceph cluster/réplication (10 Gbps)
auto ens3f1
iface ens3f1 inet static
    address 10.10.2.11/24
    mtu 9000
```

Jumbo frames

L'activation des Jumbo Frames (MTU 9000) sur le réseau Ceph réduit significativement le CPU overhead pour les gros transferts :

```
# Tester la connectivité avec Jumbo Frames
ping -M do -s 8972 10.10.2.12

# Vérifier que tous les équipements réseau supportent le MTU 9000
# (switches, cartes réseau, bonding si utilisé)
ip link set dev ens3f1 mtu 9000
```

8.5.2 BlueStore et NVMe

BlueStore est optimisé pour les NVMe mais nécessite quelques ajustements de configuration.

Tuning BlueStore pour NVMe

```
# Augmenter le cache BlueStore pour les NVMe
ceph config set osd bluestore_cache_size_ssd 4294967296
# 4 Go de cache par OSD NVMe

# Activer la compression automatique (zstd niveau 1)
ceph config set global bluestore_compression_algorithm zstd
ceph config set global bluestore_compression_mode aggressive
ceph config set global bluestore_compression_min_blob_size 8192

# Optimiser le throttling des écritures
ceph config set osd bluestore_throttle_bytes 67108864
# 64 Mo de buffer d'écriture

# Désactiver la vérification fsck au montage pour les NVMe rapides
ceph config set osd bluestore_fsck_on_mount false
```

Vérification des performances BlueStore

```
# Statistiques détaillées BlueStore
ceph tell osd.0 perf dump | python3 -m json.tool | grep -A5 bluestore

# Vérifier le taux de hit du cache
ceph tell osd.0 bluestore stats | grep cache
```

Paramètres OSD pour NVMe

```
# Augmenter les threads OSD pour les NVMe rapides
ceph config set osd osd_op_num_threads_per_shard 2
ceph config set osd osd_op_num_shards 8

# Recovery optimisé pour NVMe
ceph config set osd osd_recovery_max_active_hdd 3
ceph config set osd osd_recovery_max_active_ssd 10
ceph config set osd osd_max_backfills 4
```

```
# Appliquer sans redémarrage
ceph tell osd.* injectargs \
  '--osd_recovery_max_active_ssd 10 --osd_max_backfills 4'
```

8.5.3 Cache tiering et SSD WAL/DB

Le cache tiering permettait d'utiliser des SSD comme cache devant des OSDs HDD. Cette fonctionnalité est désormais dépréciée dans Squid au profit de solutions plus simples : externaliser uniquement le WAL/DB sur SSD.

Architecture WAL/DB externalisée

```
HDD OSD (données)
    |
    | (accès données)
    |
NVMe (WAL + DB)
    |
    | (métadonnées et journal)
```

```
# Déplacer le DB d'un OSD existant vers un NVMe
# (nécessite de vider et recréer l'OSD)
ceph osd out osd.5
ceph osd purge osd.5 --yes-i-really-mean-it

# Recréer avec WAL/DB externalisé
pveceph osd create /dev/sda \
  --wal-dev /dev/nvme0n1 \
  --wal-dev-size 2 \
  --db-dev /dev/nvme0n1 \
  --db-dev-size 10
```

Dimensionnement WAL/DB

Type OSD	Taille WAL recommandée	Taille DB recommandée
HDD 4 To	1-2 Go	30-50 Go (4% capacité)
HDD 8 To	2-4 Go	60-80 Go
SATA SSD	Pas nécessaire	4-8 Go
NVMe	Pas nécessaire	Pas nécessaire

8.5.4 Benchmarking avec rados bench

`rados bench` est l'outil de référence pour mesurer les performances d'un pool Ceph.

Tests de performance séquentiels

```
# Test d'écriture séquentielle (60 secondes)
rados bench -p vm-pool 60 write \
  --no-cleanup \
  --block-size 4M \
  -t 16

# Test de lecture séquentielle
rados bench -p vm-pool 60 seq \
  -t 16

# Test de lecture aléatoire
rados bench -p vm-pool 60 rand \
  -t 16

# Nettoyage après benchmark
rados cleanup -p vm-pool
```

Test RBD avec fio

```
# Créer une image de test
rbd create vm-pool/benchmark --size 20480
```

```
# Test 4K aléatoire (IOPS)
fio \
  --ioengine=rbd \
  --pool=vm-pool \
  --rbdname=benchmark \
  --rw=randrw \
  --rwmixread=70 \
  --bs=4k \
  --iodepth=32 \
  --numjobs=4 \
  --runtime=60 \
  --name=ceph-4k-iops \
  --output-format=json \
  --output=/tmp/fio-results.json

# Analyser les résultats
python3 -c "
import json
with open('/tmp/fio-results.json') as f:
    r = json.load(f)
jobs = r['jobs'][0]
print(f'Lecture IOPS: {jobs[\"read\"][\"iops\"]:.0f}')
print(f'Ecriture IOPS: {jobs[\"write\"][\"iops\"]:.0f}')
print(f'Latence lecture p99: {jobs[\"read\"][\"lat_ns\"][\"percentile\"]
[\"99.000000\"]/1000:.1f} us')
"

# Nettoyer
rbd rm vm-pool/benchmark
```

8.6 Maintenance et opérations

8.6.1 Ajout et remplacement d'OSD

Ajout d'un nouvel OSD

L'ajout d'un OSD déclenche automatiquement un rebalancing : Ceph redistribue les PGs pour utiliser le nouveau disque.

```
# Ajouter le nouvel OSD
pveceph osd create /dev/nvme4n1

# Surveiller le rebalancing
watch -n5 ceph status

# Vérifier la progression
ceph progress
```

Remplacement d'un OSD défaillant

> ✗ **DANGER**
> Ne jamais supprimer un OSD sans s'assurer que le cluster a suffisamment de répliques disponibles. En cas de doute, vérifiez `ceph health detail` et attendez que le cluster soit à l'état `HEALTH_OK` ou `HEALTH_WARN` (non `HEALTH_ERR`) avant d'intervenir. La suppression d'un OSD alors que le cluster est déjà dégradé peut entraîner une perte de données définitive.

```
# 1. Identifier l'OSD défaillant
ceph osd tree | grep down

# 2. Marquer l'OSD comme "out" (commence le rebalancing)
ceph osd out osd.7

# 3. Attendre que le rebalancing soit terminé
watch ceph status
# Attendre : "active+clean" pour tous les PGs

# 4. Arrêter le service OSD
systemctl stop ceph-osd@7
```

```
# 5. Purger l'OSD de la CRUSH map et de la config
ceph osd purge osd.7 --yes-i-really-mean-it

# 6. Remplacer physiquement le disque

# 7. Créer le nouvel OSD
pveceph osd create /dev/nvme_neuf

# 8. Surveiller la récupération
watch ceph status
ceph progress
```

Gestion du poids pendant la maintenance

```
# Réduire progressivement le poids d'un OSD avant retrait
# (évite un pic de rebalancing brutal)
ceph osd crush reweight osd.7 0.5
# Attendre que le cluster se stabilise
ceph osd crush reweight osd.7 0.2
# Attendre
ceph osd out osd.7
```

8.6.2 Rebalancing et scrubbing

Contrôle du rebalancing

Le rebalancing peut saturer le réseau et impacter les performances des VM. Il est possible de le ralentir :

```
# Limiter la récupération (recovery)
ceph config set osd osd_recovery_max_active_ssd 2
ceph config set osd osd_recovery_op_priority 3

# Limiter le backfill
ceph config set osd osd_max_backfills 1

# Mettre en pause le rebalancing (urgence uniquement)
ceph osd set norebalance
ceph osd set nobackfill
ceph osd set norecover

# Reprendre le rebalancing
ceph osd unset norebalance
ceph osd unset nobackfill
ceph osd unset norecover
```

> ⚠ **AVERTISSEMENT**
> La mise en pause du rebalancing (`norebalance` , `nobackfill` , `norecover`) laisse le cluster dans un état dégradé. N'utilisez ces flags qu'en cas de nécessité absolue et réactivez-les dès que possible. Un cluster en état dégradé prolongé réduit la tolérance aux pannes.

Scrubbing

Le scrubbing vérifie l'intégrité des données. Le scrubbing léger (checksums) s'exécute quotidiennement ; le deep-scrubbing (lecture complète et vérification) hebdomadairement.

```
# Forcer un scrubbing immédiat sur un PG spécifique
ceph pg scrub 1.a3

# Forcer le deep-scrubbing
ceph pg deep-scrub 1.a3

# Désactiver le scrubbing temporairement (maintenance)
ceph osd set noscrub
ceph osd set nodeep-scrub

# Vérifier l'état du scrubbing
ceph pg dump | grep -E "scrub|deep" | head -20
```

```
# Planifier le scrubbing en dehors des heures de pointe
ceph config set osd osd_scrub_begin_hour 2
ceph config set osd osd_scrub_end_hour 6
ceph config set osd osd_deep_scrub_interval 604800
# 7 jours en secondes
```

8.6.3 Mise à jour du cluster Ceph

Procédure de mise à jour Ceph via Proxmox

```
# 1. Vérifier l'état du cluster avant mise à jour
ceph status
ceph health detail

# 2. Activer le flag de mise à jour
ceph osd set noout
# Évite que les OSDs soient marqués "out" pendant les redémarrages

# 3. Mettre à jour les paquets sur le premier noeud
apt update && apt upgrade -y ceph ceph-osd ceph-mon ceph-mgr

# 4. Redémarrer les services Ceph progressivement
systemctl restart ceph-mon@pve1
systemctl restart ceph-mgr@pve1
systemctl restart ceph-osd@0
systemctl restart ceph-osd@1
systemctl restart ceph-osd@2

# 5. Attendre la stabilisation du cluster
watch ceph status

# 6. Répéter sur le noeud suivant
# (Se connecter à pve2)
# apt update && apt upgrade -y ceph ceph-osd ceph-mon ceph-mgr

# 7. Après mise à jour de tous les noeuds
ceph osd unset noout

# 8. Valider la version
ceph versions
```

Mise à jour majeure (Pacific vers Squid)

```
# Vérifier la compatibilité
ceph tell mon.* version

# Activer les flags de sécurité
ceph osd set noout
ceph osd set nobackfill
ceph osd set norecover

# Mise à jour des paquets
apt install -y ceph-release
apt update && apt full-upgrade -y

# Activer les nouvelles fonctionnalités après mise à jour
ceph osd require-osd-release squid

# Désactiver les flags
ceph osd unset noout
ceph osd unset nobackfill
ceph osd unset norecover
```

8.6.4 Surveillance : ceph status, dashboard

Commandes de surveillance essentielles

```
# Vue synthétique du cluster
ceph status
# ou
```

```
ceph -s

# Sortie exemple :
#   cluster:
#     id:     a94f9906-00b7-11ef-bbcd-d7a26db40d42
#     health: HEALTH_OK
#   services:
#     mon: 3 daemons, quorum pve1,pve2,pve3 (age 2d)
#     mgr: pve1(active, since 2d), standbys: pve2
#     mds: 1/1 daemons up, 1 standby
#     osd: 9 osds: 9 up (since 2d), 9 in (since 2d)
#   data:
#     volumes: 1/1 healthy
#     pools:   5 pools, 305 pgs
#     objects: 45.12k objects, 145 GiB
#     usage:   437 GiB used, 26 TiB / 27 TiB avail
#     pgs:     305 active+clean

# Suivi en temps réel
watch -n5 ceph status

# Métriques de performance I/O
ceph iostat 5
# Rafraîchissement toutes les 5 secondes

# État des OSDs
ceph osd stat
ceph osd perf

# Utilisation de l'espace par pool
ceph df

# Utilisation détaillée
ceph df detail

# État des PGs
ceph pg stat

# Historique des événements
ceph log last 50
```

Configuration du Dashboard Ceph

```
# Vérifier l'état du dashboard
ceph mgr module ls | grep dashboard

# Accéder au dashboard
# URL : https://10.10.1.11:8443/

# Créer un utilisateur en lecture seule pour la supervision
ceph dashboard ac-user-create monitoring --roles read-only \
  -i /tmp/monitoring-pass

# Activer les alertes email
ceph dashboard set-alertmanager-api-host \
  http://alertmanager:9093

# Configurer les seuils d'alerte
ceph config set global mon_osd_down_out_interval 600
# 10 minutes avant de marquer un OSD "out" automatiquement
```

Intégration Prometheus/Grafana

```
# Vérifier que le module Prometheus est actif
ceph mgr module ls | grep prometheus

# URL des métriques
curl http://10.10.1.11:9283/metrics | head -50
```

Métriques Prometheus utiles pour les tableaux de bord Grafana :

Métrique	Description
`ceph_cluster_total_bytes`	Capacité totale du cluster
`ceph_cluster_total_used_bytes`	Espace utilisé
`ceph_osd_stat_bytes_used`	Utilisation par OSD
`ceph_pg_active`	Nombre de PGs actifs
`ceph_osd_op_r_latency_sum`	Latence lecture cumulée
`ceph_osd_op_w_latency_sum`	Latence écriture cumulée
`ceph_pool_rd_bytes`	Débit lecture par pool
`ceph_pool_wr_bytes`	Débit écriture par pool

8.7 Lab TechFlow SAS : Ceph hyper-convergé

TechFlow SAS souhaite mettre fin à sa dépendance à un SAN propriétaire coûteux. L'objectif est de déployer un cluster Ceph hyper-convergé sur les trois noeuds Proxmox existants, avec des disques NVMe haute performance et un réseau dédié.

8.7.1 Plan de déploiement (3 noeuds, 3 OSDs chacun)

Infrastructure cible

Noeud	Hostname	IP publique	IP cluster Ceph	RAM	OSDs
Serveur 1	pve1.techflow.lan	10.10.1.11/24	10.10.2.11/24	128 Go	osd.0, osd.1, osd.2
Serveur 2	pve2.techflow.lan	10.10.1.12/24	10.10.2.12/24	128 Go	osd.3, osd.4, osd.5
Serveur 3	pve3.techflow.lan	10.10.1.13/24	10.10.2.13/24	128 Go	osd.6, osd.7, osd.8

Inventaire disques par noeud

Disque	Type	Rôle	Capacité
/dev/nvme0n1	NVMe PCIe 4.0	OS Proxmox (ZFS mirror)	500 Go
/dev/nvme1n1	NVMe PCIe 4.0	OSD Ceph 0/3/6	2 To
/dev/nvme2n1	NVMe PCIe 4.0	OSD Ceph 1/4/7	2 To
/dev/nvme3n1	NVMe PCIe 4.0	OSD Ceph 2/5/8	2 To

Capacité brute totale : 3 noeuds x 3 x 2 To = 18 To Capacité utilisable (réplication 3) : 18 To / 3 = **6 To**

Services Ceph par noeud

Service	pve1	pve2	pve3
MON	Oui (quorum)	Oui (quorum)	Oui (quorum)
MGR	Actif	Standby	-
OSD	3 (0,1,2)	3 (3,4,5)	3 (6,7,8)
MDS	Actif	Standby	-
RGW	Oui	-	-

8.7.2 Configuration réseau Ceph (VLAN dédié)

TechFlow utilise un VLAN dédié (VLAN 20) pour le réseau de réplication Ceph sur un réseau 10 Gbps.

Configuration réseau sur pve1

```
# /etc/network/interfaces sur pve1
cat > /etc/network/interfaces << 'EOF'
auto lo
iface lo inet loopback
```

```
# Interface management / Proxmox (1 Gbps)
auto eno1
iface eno1 inet static
    address 10.10.0.11/24
    gateway 10.10.0.1

# Interface Ceph public (10 Gbps) - VLAN 10
auto ens5f0.10
iface ens5f0.10 inet static
    address 10.10.1.11/24
    mtu 9000
    vlan-raw-device ens5f0

# Interface Ceph cluster/réplication (10 Gbps) - VLAN 20
auto ens5f0.20
iface ens5f0.20 inet static
    address 10.10.2.11/24
    mtu 9000
    vlan-raw-device ens5f0

# Bridge VM (1 Gbps)
auto vmbr0
iface vmbr0 inet static
    address 192.168.1.11/24
    gateway 192.168.1.1
    bridge-ports eno2
    bridge-stp off
    bridge-fd 0
EOF

# Appliquer la configuration
systemctl restart networking

# Vérifier la connectivité réseau Ceph (avec Jumbo Frames)
ping -c3 -M do -s 8972 10.10.1.12
ping -c3 -M do -s 8972 10.10.2.12
```

Vérification de la bande passante réseau Ceph

```
# Test de bande passante entre noeuds
# Sur pve2 (serveur iperf3)
iperf3 -s -B 10.10.2.12

# Sur pve1 (client iperf3)
iperf3 -c 10.10.2.12 \
  -B 10.10.2.11 \
  --bytes 10G \
  --parallel 4

# Résultat attendu sur 10 Gbps avec Jumbo Frames :
# [SUM] 0.00-10.00 sec  11.8 GBytes  10.1 Gbits/sec  sender
```

Initialisation du cluster Ceph TechFlow

```
# Sur pve1 : initialiser Ceph
pveceph init \
  --network 10.10.1.0/24 \
  --cluster-network 10.10.2.0/24

# Sur pve1 : créer le premier MON
pveceph mon create

# Rejoindre pve2 au cluster Proxmox (si pas déjà fait)
# Puis sur pve2 : ajouter le MON
pveceph mon create

# Sur pve3 : ajouter le MON
pveceph mon create
```

```
# Vérifier le quorum
ceph mon stat
# e3: 3 mons at {pve1=10.10.1.11:6789/0, ... }

# Créer les MGRs
# Sur pve1
pveceph mgr create
# Sur pve2
pveceph mgr create
```

Ajout des 9 OSDs NVMe

```
# Sur pve1 : ajouter les 3 OSDs NVMe
for DISK in /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1; do
    echo "Ajout OSD sur $DISK ... "
    pveceph osd create $DISK
    sleep 10
    ceph status | grep -E "osd:|pgs:"
done

# Sur pve2 : ajouter les 3 OSDs NVMe
for DISK in /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1; do
    pveceph osd create $DISK
    sleep 10
done

# Sur pve3 : ajouter les 3 OSDs NVMe
for DISK in /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1; do
    pveceph osd create $DISK
    sleep 10
done

# Vérification finale
ceph osd tree
# ID  CLASS  WEIGHT   TYPE NAME       STATUS  REWEIGHT  PRI-AFF
# -1         18.00000 root default
# -3          6.00000     host pve1
#  0    nvme   2.00000         osd.0       up   1.00000  1.00000
#  1    nvme   2.00000         osd.1       up   1.00000  1.00000
#  2    nvme   2.00000         osd.2       up   1.00000  1.00000
# -5          6.00000     host pve2
#  3    nvme   2.00000         osd.3       up   1.00000  1.00000
# [ ... ]
```

8.7.3 Création des pools production

TechFlow SAS définit les pools suivants pour ses environnements de production :

Pool	Type	Usage	PGs	Réplication
vm-pool	RBD répliqué	Disques VM production	256	3
vm-pool-dev	RBD répliqué	Disques VM développement	64	2
cephfs-meta	Répliqué	Métadonnées CephFS	32	3
cephfs-data	Répliqué	Données CephFS (ISOs, backups)	128	3
backup-pool	Répliqué	Sauvegardes PBS	64	2

```
# ═══ Création du pool VM production (vm-pool) ═══
ceph osd pool create vm-pool 256 256 replicated
ceph osd pool set vm-pool size 3
ceph osd pool set vm-pool min_size 2
ceph osd pool application enable vm-pool rbd
rbd pool init vm-pool

# Activer l'autoscaler avec cible 2 To
ceph osd pool set vm-pool pg_autoscale_mode on
ceph osd pool set vm-pool target_size_bytes 2199023255552
```

```
# ═══ Création du pool VM développement ═══
ceph osd pool create vm-pool-dev 64 64 replicated
ceph osd pool set vm-pool-dev size 2
ceph osd pool set vm-pool-dev min_size 1
ceph osd pool application enable vm-pool-dev rbd
rbd pool init vm-pool-dev

# ═══ Création des pools CephFS ═══
ceph osd pool create cephfs-meta 32 32 replicated
ceph osd pool create cephfs-data 128 128 replicated
ceph osd pool set cephfs-meta size 3
ceph osd pool set cephfs-data size 3
ceph fs new techflow-fs cephfs-meta cephfs-data

# Créer les MDS
# Sur pve1
pveceph mds create
# Sur pve2 (standby)
pveceph mds create

# ═══ Création du pool backup ═══
ceph osd pool create backup-pool 64 64 replicated
ceph osd pool set backup-pool size 2
ceph osd pool set backup-pool min_size 1
ceph osd pool application enable backup-pool rgw

# ═══ Vérification globale ═══
ceph osd pool ls detail
ceph df
```

Configuration des quotas par pool

```
# Quota maximum sur le pool backup (limite à 3 To)
ceph osd pool set-quota backup-pool max_bytes 3298534883328

# Vérifier les quotas
ceph osd pool get-quota vm-pool
ceph osd pool get-quota backup-pool
```

Déclaration des stockages dans Proxmox

```
# Pool RBD production
pvesm add rbd ceph-vm \
  --monhost "10.10.1.11 10.10.1.12 10.10.1.13" \
  --pool vm-pool \
  --username admin \
  --content images \
  --krbd 0 \
  --max-protected-backups 5

# Pool RBD développement
pvesm add rbd ceph-dev \
  --monhost "10.10.1.11 10.10.1.12 10.10.1.13" \
  --pool vm-pool-dev \
  --username admin \
  --content images

# CephFS pour ISOs et backups
pvesm add cephfs cephfs-shared \
  --monhost "10.10.1.11 10.10.1.12 10.10.1.13" \
  --path /mnt/pve/cephfs-shared \
  --content iso,vztmpl,backup \
  --username admin

# Vérification
pvesm status
# Name             Type     Status           Total            Used
Available        %
```

```
# ceph-dev          rbd      active       2097152.00       102400.00
1994752.00    4.88%
# ceph-vm           rbd      active       6291456.00       524288.00
5767168.00    8.33%
# cephfs-shared     cephfs   active       6291456.00        32768.00
6258688.00    0.52%
```

8.7.4 Migration des VM sur Ceph RBD

TechFlow dispose de 12 VM actuellement sur le stockage local NVMe de pve1. La migration vers Ceph RBD permet la haute disponibilité et la migration live.

Inventaire des VM à migrer

```
# Lister toutes les VM du cluster
pvesh get /cluster/resources --type vm

# Identifier les VM sur stockage local
for VMID in $(qm list | awk 'NR>1 {print $1}'); do
    STORAGE=$(qm config $VMID | grep -E "^(scsi|virtio|ide|sata)" | \
              head -1 | cut -d: -f2 | cut -d, -f1)
    echo "VM $VMID : $STORAGE"
done
```

Script de migration en masse

```
#!/bin/bash
# /opt/techflow/migrate-to-ceph.sh
# Migration des VM de local-zfs vers ceph-vm

VMLIST=(100 101 102 103 104 105 106 107 108 109 110 111)
TARGET_NODE="pve1"
TARGET_STORAGE="ceph-vm"
MIGRATE_LOG="/var/log/techflow/ceph-migration-$(date +%Y%m%d-%H%M).log"

mkdir -p /var/log/techflow
echo "═══ Migration Ceph TechFlow SAS ═══" | tee $MIGRATE_LOG
echo "Démarrée le $(date)" | tee -a $MIGRATE_LOG
echo "Noeud cible : $TARGET_NODE" | tee -a $MIGRATE_LOG
echo "Pool cible : $TARGET_STORAGE" | tee -a $MIGRATE_LOG
echo "" | tee -a $MIGRATE_LOG

for VMID in "${VMLIST[@]}"; do
    echo "--- Migration VM $VMID ---" | tee -a $MIGRATE_LOG

    # Vérifier que la VM existe
    if ! qm status $VMID &>/dev/null; then
        echo "[SKIP] VM $VMID introuvable" | tee -a $MIGRATE_LOG
        continue
    fi

    # Obtenir le statut et le noeud courant
    STATUS=$(qm status $VMID | awk '{print $2}')
    echo "[INFO] VM $VMID statut : $STATUS" | tee -a $MIGRATE_LOG

    # Migration online si la VM tourne
    if [ "$STATUS" = "running" ]; then
        echo "[INFO] Migration online en cours..." | tee -a $MIGRATE_LOG
        qm migrate $VMID $TARGET_NODE \
          --online \
          --targetstorage $TARGET_STORAGE \
          2>&1 | tee -a $MIGRATE_LOG
    else
        # Déplacer le disque si la VM est arrêtée
        echo "[INFO] Déplacement du disque scsi0..." | tee -a $MIGRATE_LOG
        qm move-disk $VMID scsi0 $TARGET_STORAGE \
          --delete 1 \
          2>&1 | tee -a $MIGRATE_LOG
    fi
```

```
    if [ $? -eq 0 ]; then
        echo "[OK] VM $VMID : migration réussie" | tee -a $MIGRATE_LOG
    else
        echo "[ERR] VM $VMID : ERREUR de migration" | tee -a $MIGRATE_LOG
    fi

    # Vérifier la santé du cluster après chaque migration
    HEALTH=$(ceph health 2>/dev/null)
    echo "[CEPH] $HEALTH" | tee -a $MIGRATE_LOG

    # Pause pour stabilisation
    sleep 15
done

echo "" | tee -a $MIGRATE_LOG
echo "=== Migration terminée le $(date) ===" | tee -a $MIGRATE_LOG
echo "Rapport final :" | tee -a $MIGRATE_LOG
ceph status | tee -a $MIGRATE_LOG
pvesm status | tee -a $MIGRATE_LOG
```

Exécution de la migration

```
chmod +x /opt/techflow/migrate-to-ceph.sh
/opt/techflow/migrate-to-ceph.sh

# Suivre la migration en temps réel dans un autre terminal
tail -f /var/log/techflow/ceph-migration-*.log

# Surveiller le cluster pendant la migration
watch -n10 "ceph status && echo '---' && pvesm status"
```

Validation post-migration

```
# Vérifier que toutes les VM utilisent bien le stockage Ceph
for VMID in $(qm list | awk 'NR>1 {print $1}'); do
    CONFIG=$(qm config $VMID)
    STORAGE=$(echo "$CONFIG" | grep -E "^scsi0" | grep -o "ceph-[^:]*")
    NAME=$(echo "$CONFIG" | grep "^name:" | awk '{print $2}')
    echo "VM $VMID ($NAME) : ${STORAGE:-LOCAL}"
done

# Lister les images RBD créées
rbd ls vm-pool | sort

# Vérifier l'espace utilisé
ceph df | grep vm-pool
```

8.7.5 Validation des performances

Tests de performance du cluster TechFlow

```
# === Test 1 : Bande passante séquentielle ===
echo "=== Test écriture séquentielle 4Mo ==="
rados bench -p vm-pool 60 write \
  --block-size 4M \
  --no-cleanup \
  -t 32 \
  2>/dev/null | grep -E "Bandwidth|Average|Stddev"

echo ""
echo "=== Test lecture séquentielle ==="
rados bench -p vm-pool 60 seq \
  -t 32 \
  2>/dev/null | grep -E "Bandwidth|Average"

# Nettoyer
rados cleanup -p vm-pool

# === Test 2 : IOPS 4K avec fio via RBD ===
rbd create vm-pool/perf-test --size 51200
```

```
echo ""
echo "═══ Test IOPS 4K mixte 70R/30W ═══"
fio \
  --ioengine=rbd \
  --pool=vm-pool \
  --rbdname=perf-test \
  --clientname=admin \
  --rw=randrw \
  --rwmixread=70 \
  --bs=4k \
  --iodepth=64 \
  --numjobs=4 \
  --runtime=120 \
  --name=techflow-4k \
  --output-format=json | \
  python3 -c "
import json, sys
d = json.load(sys.stdin)
j = d['jobs'][0]
r_iops = j['read']['iops']
w_iops = j['write']['iops']
r_lat = j['read']['lat_ns']['percentile']['99.000000'] / 1000
w_lat = j['write']['lat_ns']['percentile']['99.000000'] / 1000
print(f'Lecture  : {r_iops:>10.0f} IOPS  |  Latence p99 : {r_lat:.1f} us')
print(f'Ecriture : {w_iops:>10.0f} IOPS  |  Latence p99 : {w_lat:.1f} us')
"

echo ""
echo "═══ Test latence 4K écriture synchrone ═══"
fio \
  --ioengine=rbd \
  --pool=vm-pool \
  --rbdname=perf-test \
  --clientname=admin \
  --rw=randwrite \
  --bs=4k \
  --iodepth=1 \
  --numjobs=1 \
  --runtime=60 \
  --name=techflow-latency \
  --output-format=json | \
  python3 -c "
import json, sys
d = json.load(sys.stdin)
j = d['jobs'][0]
lat = j['write']['lat_ns']
pct = lat['percentile']
print(f'Latence min   : {lat[\"min\"]/1000:.1f} us')
print(f'Latence moy   : {lat[\"mean\"]/1000:.1f} us')
print(f'Latence p95   : {pct[\"95.000000\"]/1000:.1f} us')
print(f'Latence p99   : {pct[\"99.000000\"]/1000:.1f} us')
print(f'Latence p99.9 : {pct[\"99.900000\"]/1000:.1f} us')
"

# Nettoyer
rbd rm vm-pool/perf-test
```

Résultats attendus pour 9 OSDs NVMe TechFlow

Métrique	Valeur attendue	Valeur minimale acceptable
Bande passante écriture séquentielle	4-6 Go/s	2 Go/s
Bande passante lecture séquentielle	8-12 Go/s	4 Go/s
IOPS 4K aléatoires mixte 70/30	150 000 - 250 000	50 000
Latence écriture 4K p99	< 2 ms	< 5 ms

Métrique	Valeur attendue	Valeur minimale acceptable
Latence lecture 4K p99	< 1 ms	< 3 ms

Validation de la haute disponibilité

```
# Test de tolérance aux pannes : simuler la perte d'un noeud pve3

# 1. État initial — tout doit être HEALTH_OK
ceph status
echo "Noeuds : $(ceph osd stat | grep 'up')"

# 2. Démarrer une charge de travail continue en arrière-plan
fio \
  --ioengine=rbd \
  --pool=vm-pool \
  --rbdname=ha-test \
  --clientname=admin \
  --rw=randrw --bs=4k --iodepth=8 \
  --numjobs=2 --runtime=300 \
  --name=ha-test &
FIO_PID=$!

# 3. Arrêter tous les OSDs de pve3 (simulation de panne noeud)
ssh pve3 "systemctl stop ceph-osd@6 ceph-osd@7 ceph-osd@8"
echo "OSDs pve3 arrêtés à $(date)"

# 4. Observer la réaction du cluster
for i in $(seq 1 12); do
    echo "--- t+$((i*10))s ---"
    ceph health
    sleep 10
done

# 5. Vérifier que le workload fio continue
if kill -0 $FIO_PID 2>/dev/null; then
    echo "fio toujours en cours : la charge continue pendant la panne"
else
    echo "ALERTE : fio s'est terminé prématurément"
fi

# 6. Restaurer les OSDs pve3
ssh pve3 "systemctl start ceph-osd@6 ceph-osd@7 ceph-osd@8"
echo "OSDs pve3 restaurés à $(date)"

# 7. Surveiller la récupération
watch -n5 "ceph status && ceph progress"
```

Rapport de validation final

```
#!/bin/bash
# /opt/techflow/rapport-ceph.sh
# Rapport de santé et validation du cluster Ceph TechFlow

REPORT_FILE="/var/log/techflow/ceph-validation-$(date +%Y%m%d).txt"

cat > $REPORT_FILE << 'HEADER'
═══════════════════════════════════════════════════════
  RAPPORT DE VALIDATION CEPH — TECHFLOW SAS
═══════════════════════════════════════════════════════
HEADER
echo "  Généré le $(date)" >> $REPORT_FILE
echo "═══════════════════════════════════════════════════════" >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- ÉTAT GLOBAL ---" >> $REPORT_FILE
ceph status >> $REPORT_FILE

echo "" >> $REPORT_FILE
```

```
echo "--- UTILISATION ESPACE ---" >> $REPORT_FILE
ceph df >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- DÉTAIL DES POOLS ---" >> $REPORT_FILE
ceph osd pool ls detail >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- ARBRE DES OSDs ---" >> $REPORT_FILE
ceph osd tree >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- PERFORMANCES OSDs ---" >> $REPORT_FILE
ceph osd perf >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- VERSIONS ---" >> $REPORT_FILE
ceph versions >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- QUORUM MON ---" >> $REPORT_FILE
ceph mon stat >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- ÉTAT DES PGs ---" >> $REPORT_FILE
ceph pg stat >> $REPORT_FILE

echo "" >> $REPORT_FILE
echo "--- STOCKAGES PROXMOX ---" >> $REPORT_FILE
pvesm status >> $REPORT_FILE

cat $REPORT_FILE
echo ""
echo "Rapport sauvegardé : $REPORT_FILE"
```

Résumé du chapitre

Ce chapitre a couvert l'ensemble du cycle de vie d'un cluster Ceph intégré à Proxmox VE 9.

Architecture : Ceph repose sur des composants spécialisés — MON pour le quorum et la carte du cluster, OSD pour le stockage des données, MGR pour les métriques et modules, MDS pour CephFS, RGW pour l'API S3. L'algorithme CRUSH distribue les données sans métadonnées centralisées, garantissant le passage à l'échelle.

Déploiement : Proxmox VE 9 simplifie considérablement le déploiement Ceph via `pveceph`. La création du cluster, des MONs, MGRs et OSDs se fait en quelques commandes. La séparation des réseaux public et cluster est fondamentale pour les performances.

Pools : Le choix entre réplication et erasure coding dépend du cas d'usage. Pour les VM Proxmox, le pool répliqué (taille 3) est le seul compatible avec RBD de manière native. L'erasure coding convient aux objets et archives.

Performance : BlueStore avec NVMe offre d'excellentes performances. Les Jumbo Frames sur le réseau Ceph cluster réduisent le CPU overhead. Le pg_autoscaler de Squid simplifie la gestion des PGs.

Maintenance : Le remplacement d'OSD doit toujours commencer par `osd out` et attendre la stabilisation avant toute intervention physique. Le rebalancing peut être ralenti pendant les heures de production.

Lab TechFlow : Le cluster de 9 OSDs NVMe sur 3 noeuds avec réseau 10 Gbps dédié (VLAN 20, 10.10.2.x/24) et pool `vm-pool` en réplication 3 offre 6 To de capacité utilisable, avec tolérance à la perte simultanée de 2 OSDs ou d'un noeud complet.

✔ POINTS CLÉS À RETENIR

- Toujours déployer 3 MONs minimum pour le quorum Paxos
- Utiliser un réseau dédié 10 Gbps avec MTU 9000 pour le trafic de réplication Ceph
- Un OSD par disque physique, jamais deux OSDs sur le même disque
- Les pools répliqués sont obligatoires pour les disques de VM (RBD)
- Avant de retirer un OSD, toujours vérifier `ceph health detail` et attendre `HEALTH_OK`
- Le pg_autoscaler de Ceph Squid simplifie la gestion des PGs
- Activer le flag `noout` avant toute opération de maintenance pour éviter les rebalancings intempestifs

PARTIE V

Operations

9
Sauvegardes et reprise après sinistre

La sauvegarde est souvent perçue comme une contrainte administrative. Elle est en réalité l'assurance-vie de votre infrastructure. Un cluster Proxmox VE peut héberger des dizaines de machines virtuelles critiques : ERP, bases de données, serveurs applicatifs. Sans stratégie de sauvegarde cohérente et testée, la moindre défaillance matérielle, erreur humaine ou attaque par ransomware peut se transformer en catastrophe irréversible.

Ce chapitre traite de l'ensemble du cycle de vie de la donnée protégée : de la définition des objectifs de récupération jusqu'au runbook de bascule en situation de sinistre réel. Nous abordons **Proxmox Backup Server (PBS)**, la solution native dédiée, ses fonctionnalités avancées de déduplication et de chiffrement, ainsi que la mise en œuvre d'un plan de reprise après sinistre (PRA) complet pour TechFlow SAS.

9.1 Stratégie de sauvegarde

Avant d'écrire la moindre commande ou de configurer le premier job, il est indispensable de définir une stratégie. Une sauvegarde sans stratégie est une collection de fichiers dont on ne sait ni la fraîcheur, ni la fiabilité, ni la durée de rétention.

9.1.1 Règle 3-2-1 appliquée à Proxmox

La règle **3-2-1** est la référence universelle en matière de sauvegarde :

- **3** copies des données (production + 2 sauvegardes)
- **2** supports de stockage différents (technologie ou emplacement distincts)
- **1** copie hors site (offsite, cloud, site secondaire)

Appliquée à Proxmox VE, cette règle se décline de la façon suivante :

Copie	Emplacement	Technologie	Rôle
1 — Production	Cluster PVE primaire	Ceph / LVM / ZFS	Données actives
2 — Sauvegarde locale	PBS local (même salle)	PBS datastore sur NVMe/SAS	Restauration rapide
3 — Sauvegarde distante	PBS remote (site DR)	PBS datastore sur site secondaire	Sinistre site primaire

> ✔ **CONSEIL**
> Certaines organisations ajoutent une quatrième copie sur stockage objet immuable (S3, B2, Wasabi) pour se protéger des ransomwares qui cibleraient également le PBS. PBS 3.x supporte nativement les backends S3 via un proxy FUSE ou l'outil `pxar`.

L'implémentation concrète dans l'environnement TechFlow SAS ressemble à :

```
PVE Cluster (pve1/pve2/pve3)
        │
        ├── PBS local (pbs1.techflow.lan / 10.10.0.10)
        │       └── Datastore: techflow-backup (NVMe RAID1)
        │
        └── PBS remote (pbs-dr.techflow-dr.lan / 10.20.0.10)
                └── Datastore: techflow-dr (réplication depuis PBS local)
```

9.1.2 RPO et RTO : définir ses objectifs

Deux métriques fondamentales gouvernent toute stratégie de reprise :

RPO — Recovery Point Objective (Point de reprise objectif)
: Durée maximale acceptable de perte de données. Un RPO de 4 heures signifie qu'en cas de sinistre, on accepte de perdre au plus 4 heures de transactions.

RTO — Recovery Time Objective (Délai de reprise objectif)
: Durée maximale acceptable d'interruption de service. Un RTO de 2 heures signifie que le service doit être restauré en moins de 2 heures après le déclenchement du sinistre.

> **► NOTE**
> RPO et RTO sont des objectifs **métier**, pas des contraintes techniques. Ils doivent être définis par la direction en accord avec les équipes IT, en tenant compte du coût de l'interruption et du coût de la protection.

Tableau de référence pour TechFlow SAS :

Système	Criticité	RPO cible	RTO cible	Fréquence sauvegarde	Rétention
ERP (vm-erp-01)	Critique	1 heure	2 heures	Toutes les heures	30 jours
Base de données (vm-db-01)	Critique	15 min	1 heure	15 min (réplication)	14 jours
Serveur web (vm-web-01/02)	Élevée	4 heures	4 heures	Toutes les 6 heures	14 jours
Serveur de fichiers (vm-files-01)	Élevée	24 heures	8 heures	Quotidienne	90 jours
Infrastructure (DNS, DHCP)	Moyenne	24 heures	4 heures	Quotidienne	30 jours
VMs de dev/test	Faible	72 heures	24 heures	Hebdomadaire	7 jours

9.1.3 Modes de sauvegarde (stop, suspend, snapshot)

Proxmox VE propose trois modes de sauvegarde, chacun avec ses compromis en termes de cohérence et d'impact sur la disponibilité :

Mode `stop` La VM ou le conteneur est arrêté avant la sauvegarde, puis redémarré. C'est le mode le plus fiable en termes de cohérence des données (notamment pour les bases de données), mais il implique une interruption de service pendant la durée de la sauvegarde.

- Cohérence : maximale (état disque propre)
- Impact : interruption de service
- Usage recommandé : VMs non critiques, environnements de développement

Mode `suspend` La VM est suspendue (frozen) pendant la sauvegarde. L'état RAM est sauvegardé. La VM est ensuite reprise. L'impact est réduit par rapport à `stop`, mais la VM est indisponible pendant la phase de sauvegarde du disque.

- Cohérence : élevée (RAM incluse)
- Impact : courte interruption (gel de la VM)
- Usage recommandé : VMs avec état applicatif important en mémoire

Mode `snapshot` Un snapshot du disque est pris à chaud. La VM continue de fonctionner pendant la sauvegarde. C'est le mode préféré pour les systèmes critiques, à condition que le pilote QEMU guest agent soit installé pour le quiescing des systèmes de fichiers.

- Cohérence : élevée avec guest agent, variable sans
- Impact : minimal (quelques secondes de gel I/O au moment du snapshot)
- Usage recommandé : VMs de production, bases de données avec agent

> **⚠ AVERTISSEMENT**
> Sans QEMU Guest Agent installé et actif sur la VM, le mode snapshot peut produire une sauvegarde dans un état « crash-consistent » uniquement (comme si on avait coupé l'alimentation). Pour les bases de données, il est impératif d'activer le guest agent afin de permettre le flush des buffers avant le snapshot.

Activation du guest agent dans la configuration VM :

```
# Activer le guest agent sur la VM 100
qm set 100 --agent enabled=1

# Vérifier que le package est installé dans la VM (Debian/Ubuntu)
# Sur la VM guest :
apt install qemu-guest-agent
systemctl enable --now qemu-guest-agent
```

Pour les conteneurs LXC, le mode snapshot utilise un snapshot du volume sous-jacent (LVM, ZFS ou Ceph). La cohérence est assurée par le freeze du système de fichiers.

9.2 Proxmox Backup Server (PBS)

Proxmox Backup Server est une solution de sauvegarde dédiée développée par Proxmox Server Solutions. Contrairement à un simple stockage NFS ou CIFS monté dans Proxmox VE, PBS est une application complète avec déduplication, chiffrement, vérification d'intégrité et gestion de la rétention.

9.2.1 Architecture PBS

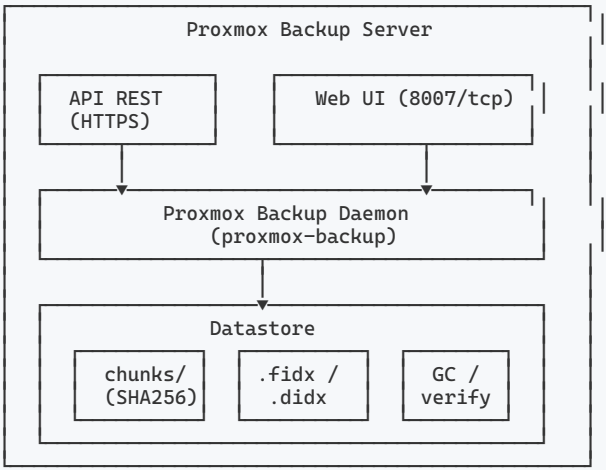
PBS repose sur plusieurs composants clés :

Le chunk store PBS ne sauvegarde pas les images disque comme des fichiers monolithiques. Il découpe les données en **chunks** de taille variable (4 Mo par défaut), calcule leur empreinte SHA-256, et ne stocke chaque chunk unique qu'une seule fois. Cela permet une déduplication efficace à l'échelle de tous les backups d'un datastore.

Le catalogue (catalog) Chaque sauvegarde dispose d'un catalogue indexant la liste des fichiers et leur position dans les chunks. Cela permet la restauration de fichiers individuels sans restaurer l'intégralité du backup.

L'index (.fidx / .didx) Les fichiers `.fidx` (fixed-index) indexent les images disque fixes. Les fichiers `.didx` (dynamic-index) indexent les archives pxar (format d'archive Proxmox). Ces index permettent de reconstruire le flux de données à partir des chunks.

Schéma de l'architecture PBS :



9.2.2 Installation et configuration

PBS s'installe sur une machine Debian dédiée (fortement recommandé) ou en machine virtuelle. Il ne doit **pas** être co-localisé sur un nœud Proxmox VE pour des raisons d'indépendance (si le nœud tombe, le PBS ne doit pas tomber avec lui).

Prérequis matériels pour TechFlow SAS (pbs1.techflow.lan) :

Composant	Minimum	Recommandé (TechFlow)
CPU	2 cœurs	4 cœurs (Intel Xeon E-2224)
RAM	2 Go	8 Go (avec déduplication active)
OS disk	16 Go	32 Go SSD
Data disk	selon données	2x 2 To NVMe (RAID logiciel ZFS)
Réseau	1 GbE	10 GbE (pour sauvegardes rapides)

Installation depuis les dépôts Proxmox :

```
# Sur le serveur Debian 12 dédié (pbs1.techflow.lan)
```

```
# Ajouter le dépôt PBS
wget https://enterprise.proxmox.com/debian/proxmox-release-bookworm.gpg \
  -O /etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg

# Dépôt no-subscription (sans abonnement)
echo "deb http://download.proxmox.com/debian/pbs bookworm pbs-no-subscription" \
  > /etc/apt/sources.list.d/pbs.list

# Mettre à jour et installer
apt update && apt install proxmox-backup-server -y

# Vérifier le statut du service
systemctl status proxmox-backup
```

Accès à l'interface web : L'interface PBS est accessible sur `https://10.10.0.10:8007`. Les identifiants par défaut sont `root@pam` avec le mot de passe root du système.

Configuration réseau recommandée :

```
# /etc/network/interfaces sur pbs1.techflow.lan
auto lo
iface lo inet loopback

auto ens18
iface ens18 inet static
    address 10.10.0.10/24
    gateway 10.10.0.1
    dns-nameservers 10.10.0.53
    dns-search techflow.lan
```

Sécuriser l'accès SSH et créer un utilisateur dédié :

```
# Créer un utilisateur PBS pour les sauvegardes depuis PVE
# Dans l'interface PBS : Administration > User Management > Add
# Ou via la CLI :
proxmox-backup-manager user create backup@pbs \
  --comment "Compte sauvegarde PVE" \
  --password "MotDePasseComplexe2024!"

# Attribuer le rôle DatastoreBackup
proxmox-backup-manager acl update /datastore/techflow-backup \
  --auth-id backup@pbs \
  --role DatastoreBackup
```

9.2.3 Datastores et namespaces

Un **datastore** est le répertoire racine sur le système de fichiers PBS où sont stockés les chunks et les index. Il correspond à une unité logique de stockage avec ses propres politiques de rétention et ses droits d'accès.

Création du datastore principal sur ZFS :

```
# Sur pbs1, créer le pool ZFS sur les deux disques NVMe
zpool create -f techflow-data mirror /dev/nvme0n1 /dev/nvme1n1

# Créer le dataset pour les sauvegardes
zfs create techflow-data/backups
zfs set compression=lz4 techflow-data/backups

# Dans l'interface PBS ou via CLI, créer le datastore
proxmox-backup-manager datastore create techflow-backup \
  --path /techflow-data/backups \
  --comment "Datastore principal TechFlow SAS"
```

Les namespaces (espaces de noms) permettent d'organiser les sauvegardes au sein d'un datastore. C'est particulièrement utile dans un contexte multi-tenant ou pour séparer les environnements :

```
techflow-backup/
├── ns: production/
│   ├── vm/100/   (vm-erp-01)
```

```
│   ├── vm/101/   (vm-db-01)
│   └── vm/102/   (vm-web-01)
├── ns: infrastructure/
│   ├── vm/200/   (vm-dns-01)
│   └── ct/300/   (ct-proxy-01)
└── ns: developpement/
    └── vm/400/   (vm-dev-*)
```

```
# Créer les namespaces
proxmox-backup-manager namespace create techflow-backup --ns production
proxmox-backup-manager namespace create techflow-backup --ns infrastructure
proxmox-backup-manager namespace create techflow-backup --ns developpement

# Attribuer des droits spécifiques par namespace
proxmox-backup-manager acl update \
  /datastore/techflow-backup/production \
  --auth-id backup@pbs \
  --role DatastoreBackup
```

9.2.4 Déduplication et chiffrement

Déduplication

La déduplication de PBS opère au niveau des chunks. Deux sauvegardes successives d'une même VM partageront la majorité de leurs chunks si le contenu disque n'a pas changé entre les deux. Le ratio de déduplication typique est de 5:1 à 20:1 selon les charges de travail.

Pour consulter les statistiques de déduplication :

```
# Via CLI
proxmox-backup-manager datastore show techflow-backup

# Exemple de sortie :
# Datastore: techflow-backup
# Path: /techflow-data/backups
# Total: 4.0 TiB
# Used: 1.2 TiB
# Dedup Factor: 8.32
```

Chiffrement AES-256

PBS implémente un chiffrement côté client en AES-256-GCM. Les clés de chiffrement ne sont **jamais** transmises au serveur : PBS stocke des données chiffrées dont il ne peut pas lire le contenu sans la clé client.

> ✗ **DANGER**
> La clé de chiffrement est la seule façon de déchiffrer vos sauvegardes. Sa perte est IRRÉVERSIBLE. Sauvegardez-la dans au moins trois endroits distincts et sécurisés : gestionnaire de mots de passe, coffre-fort physique, et éventuellement une impression papier dans un lieu sécurisé hors site.

Génération et gestion de la clé de chiffrement :

```
# Sur un nœud Proxmox VE (pve1.techflow.lan)
# Générer une nouvelle clé de chiffrement
proxmox-backup-client key create /etc/proxmox-backup/encryption-key.json

# La clé est protégée par une passphrase — choisissez-en une robuste
# Enter passphrase: ****

# Exporter la clé pour la sauvegarder (stockage hors site obligatoire)
proxmox-backup-client key show /etc/proxmox-backup/encryption-key.json \
  --output-format json > /tmp/techflow-backup-key-EXPORT.json

# IMPORTANT : copier ce fichier hors du serveur immédiatement
scp /tmp/techflow-backup-key-EXPORT.json admin@vault.techflow.lan:/secure/keys/
rm /tmp/techflow-backup-key-EXPORT.json
```

Vérification du chiffrement d'une sauvegarde :

```
# Lister les sauvegardes et vérifier le flag de chiffrement
proxmox-backup-client list \
```

```
  --repository backup@pbs@10.10.0.10:techflow-backup \
  --output-format table
```

9.2.5 Ajout de PBS dans Proxmox VE

Pour utiliser PBS comme destination de sauvegarde depuis Proxmox VE, il faut ajouter PBS comme stockage de type `pbs` dans la configuration PVE.

Via l'interface graphique :

1. Dans Proxmox VE, aller dans `Datacenter > Storage > Add > Proxmox Backup Server`
2. Renseigner les champs : - ID : `pbs-techflow` - Server : `10.10.0.10` - Username : `backup@pbs` - Password : mot de passe du compte backup - Datastore : `techflow-backup` - Namespace : laisser vide (racine) ou spécifier `production` - Fingerprint : récupérer via PBS

Récupération du fingerprint PBS :

```
# Sur pbs1.techflow.lan
proxmox-backup-manager cert info | grep Fingerprint
# Fingerprint: AB:CD:EF:12:34:56:78:90: ...
```

Via la ligne de commande sur PVE :

```
# Sur pve1.techflow.lan
pvesm add pbs pbs-techflow \
  --server 10.10.0.10 \
  --datastore techflow-backup \
  --username backup@pbs \
  --password "MotDePasseComplexe2024!" \
  --fingerprint "AB:CD:EF:12:34:56:78:90: ... " \
  --namespace production

# Vérifier que le stockage est accessible
pvesm status
# NAME          TYPE      STATUS   TOTAL         USED    AVAILABLE %
# pbs-techflow  pbs       active   4194304 MB    ...
```

Configurer le chiffrement sur le stockage PBS depuis PVE :

```
# Associer la clé de chiffrement au stockage PBS
pvesm set pbs-techflow \
  --encryption-key /etc/proxmox-backup/encryption-key.json
```

9.3 Sauvegardes avec PBS

9.3.1 Sauvegarde manuelle et planifiée

Sauvegarde manuelle depuis l'interface graphique :

1. Sélectionner la VM dans le panneau de gauche
2. Clic droit > `Backup Now`
3. Sélectionner le stockage `pbs-techflow`
4. Choisir le mode (`snapshot` recommandé)
5. Cocher `Mail notification` si souhaité
6. Cliquer `Backup`

Sauvegarde manuelle depuis la CLI :

```
# Sauvegarde d'une VM (VMID 100) vers PBS
vzdump 100 \
  --storage pbs-techflow \
  --mode snapshot \
  --compress zstd \
  --notes-template "Sauvegarde manuelle {{guestname}} - {{date}}"

# Sauvegarde d'un conteneur LXC (CTID 300)
vzdump 300 \
  --storage pbs-techflow \
  --mode snapshot

# Sauvegarde de toutes les VMs d'un nœud
```

```
vzdump --all \
  --storage pbs-techflow \
  --mode snapshot \
  --exclude 401,402

# Sauvegarde avec notification email
vzdump 100 \
  --storage pbs-techflow \
  --mode snapshot \
  --mailto admin@techflow.lan
```

Configuration d'un job de sauvegarde planifié :

Les jobs de sauvegarde se configurent dans `Datacenter > Backup > Add` dans l'interface PVE.

```
# Équivalent CLI – création d'un job de sauvegarde planifié
# Les jobs sont stockés dans /etc/pve/jobs.cfg

# Exemple de contenu jobs.cfg pour TechFlow
vzdump: job-production-daily
    storage pbs-techflow
    schedule 02:00
    mode snapshot
    enabled 1
    all 0
    vmid 100,101,102,103
    mailnotification always
    mailto admin@techflow.lan
    notes-template "Backup auto {{guestname}} {{date}}"
    compress zstd

vzdump: job-infra-daily
    storage pbs-techflow
    schedule 03:00
    mode snapshot
    enabled 1
    vmid 200,201,300,301
    namespace infrastructure

vzdump: job-dev-weekly
    storage pbs-techflow
    schedule sun 04:00
    mode stop
    enabled 1
    vmid 400,401,402
    namespace developpement
```

✔ **CONSEIL**

Échelonnez les jobs de sauvegarde dans le temps pour éviter de saturer le réseau et les I/O de stockage. PBS supporte la sauvegarde concurrente de plusieurs VMs, mais il est conseillé de ne pas dépasser 4 à 6 sauvegardes simultanées sur un cluster de taille moyenne.

9.3.2 Politiques de rétention

La rétention définit combien de sauvegardes conserver et pendant combien de temps. PBS implémente un système de rétention granulaire permettant de conserver des sauvegardes horaires, journalières, hebdomadaires, mensuelles et annuelles.

Paramètres de rétention PBS :

Paramètre	Description	Exemple
`keep-last`	Garder les N dernières sauvegardes	`keep-last 5`
`keep-hourly`	Garder N sauvegardes horaires distinctes	`keep-hourly 24`
`keep-daily`	Garder N sauvegardes journalières	`keep-daily 30`
`keep-weekly`	Garder N sauvegardes hebdomadaires	`keep-weekly 12`
`keep-monthly`	Garder N sauvegardes mensuelles	`keep-monthly 12`

Paramètre	Description	Exemple
`keep-yearly`	Garder N sauvegardes annuelles	`keep-yearly 3`

Politique de rétention TechFlow SAS :

Politiques de rétention par criticité

Système	keep-hourly	keep-daily	keep-weekly	keep-monthly	keep-yearly	Total estimé
ERP (vm-erp-01)	24	30	8	12	2	~76 points
Base de données	24	14	4	6	1	~49 points
Serveurs web	6	14	4	3	0	~27 points
Serveur de fichiers	0	90	12	12	5	~119 points
Infrastructure	0	30	8	6	1	~45 points
Dev/Test	0	7	4	0	0	~11 points

Configuration de la rétention sur le datastore PBS :

```
# Via l'interface PBS : Datastore > techflow-backup > Options > Prune Schedule

# Via CLI sur pbs1.techflow.lan
proxmox-backup-manager datastore update techflow-backup \
  --keep-last 0 \
  --keep-hourly 24 \
  --keep-daily 30 \
  --keep-weekly 8 \
  --keep-monthly 12 \
  --keep-yearly 2

# Appliquer la rétention manuellement (prune)
proxmox-backup-client prune \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  --backup-type vm \
  --backup-id 100 \
  --keep-daily 30 \
  --keep-weekly 8 \
  --keep-monthly 12 \
  --dry-run

# Appliquer sans dry-run
proxmox-backup-client prune \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  --backup-type vm \
  --backup-id 100 \
  --keep-daily 30 \
  --keep-weekly 8 \
  --keep-monthly 12
```

Récupération d'espace (Garbage Collection) :

Après un prune, les chunks orphelins ne sont pas immédiatement supprimés. Il faut lancer un Garbage Collection :

```
# Sur pbs1.techflow.lan
proxmox-backup-manager garbage-collection start techflow-backup

# Suivre la progression
proxmox-backup-manager task list --limit 5

# Planifier le GC automatiquement (dans l'interface PBS)
# Datastore > techflow-backup > GC Schedule : "sun 05:00"
```

9.3.3 Vérification d'intégrité (verify jobs)

La sauvegarde n'a de valeur que si elle peut être restaurée. PBS propose des **verify jobs** qui vérifient l'intégrité des chunks en recalculant leurs empreintes SHA-256 et en s'assurant que les index sont cohérents.

> ✔ **CONSEIL**
> Planifiez une vérification hebdomadaire de l'ensemble du datastore et une vérification immédiate après chaque sauvegarde critique. Découvrir une corruption lors d'un sinistre réel est le scénario catastrophe à éviter absolument.

Configuration d'un verify job :

```
# Via CLI sur pbs1.techflow.lan
proxmox-backup-manager verify-job create \
  --id verify-weekly \
  --datastore techflow-backup \
  --schedule "sat 06:00" \
  --ignore-verified false \
  --outdated-after 7

# Lancer une vérification manuelle immédiate
proxmox-backup-manager verify-job run verify-weekly

# Vérifier un backup spécifique
proxmox-backup-client verify \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  vm/100/2024-11-15T02:00:00Z
```

Vérification post-backup (verify-after-backup) :

PBS peut vérifier automatiquement chaque sauvegarde immédiatement après sa création :

```
# Activer verify-after-backup sur le datastore
proxmox-backup-manager datastore update techflow-backup \
  --verify-new true
```

Interprétation des logs de vérification :

```
INFO:  verify snapshot vm/100/2024-11-15T02:00:00Z
INFO:  verify archive 'drive-scsi0.img.fidx'
INFO:  verify archive 'drive-scsi1.img.fidx'
INFO:  verify archive 'qemu-server.conf.blob'
OK:    all chunks verified successfully
```

En cas d'erreur :

```
ERROR: verify error - chunk 'a3f4...' not found
ERROR: snapshot vm/100/2024-11-10T02:00:00Z: verification failed
```

Un tel message indique une corruption et nécessite une investigation immédiate.

9.3.4 Synchronisation PBS-to-PBS (remote)

La synchronisation PBS-to-PBS permet de répliquer les sauvegardes d'un PBS local vers un PBS distant (site de DR). PBS utilise son propre protocole de synchronisation qui tire parti de la déduplication : seuls les chunks nouveaux ou modifiés sont transférés.

Configuration d'un remote sur le PBS source :

```
# Sur pbs1.techflow.lan (PBS local)
# Récupérer le fingerprint du PBS distant
ssh root@10.20.0.10 proxmox-backup-manager cert info | grep Fingerprint

# Créer la configuration du remote (PBS DR)
proxmox-backup-manager remote create pbs-dr \
  --host 10.20.0.10 \
  --port 8007 \
  --userid backup@pbs \
  --password "MotDePasseComplexe2024!" \
  --fingerprint "12:34:AB:CD:...:"
```

```
# Vérifier la connectivité
proxmox-backup-manager remote show pbs-dr
```

Création d'un sync job :

```
# Créer un job de synchronisation vers le PBS DR
proxmox-backup-manager sync-job create \
  --id sync-to-dr \
  --remote pbs-dr \
  --remote-store techflow-dr \
  --local-store techflow-backup \
  --schedule "04:00" \
  --remove-vanished false \
  --comment "Sync vers site DR TechFlow"

# Optionnel : synchroniser uniquement le namespace production
proxmox-backup-manager sync-job update sync-to-dr \
  --ns production

# Lancer la synchronisation manuellement
proxmox-backup-manager sync-job run sync-to-dr

# Surveiller la progression
proxmox-backup-manager task list --limit 10
```

> ▶ **NOTE**
> Le flag `--remove-vanished false` est recommandé en production. Il évite que la suppression accidentelle de sauvegardes sur le PBS local n'entraîne leur suppression sur le PBS DR. Pour le DR, on préfère avoir trop de données que pas assez.

Vérification de la synchronisation :

```
# Sur pbs-dr.techflow-dr.lan
proxmox-backup-client list \
  --repository backup@pbs@10.20.0.10:techflow-dr \
  --output-format table

# Consulter le rapport de synchronisation
proxmox-backup-manager show-task <task-upid>
```

9.4 Restauration

La restauration est le vrai test d'une sauvegarde. Une sauvegarde qui ne peut pas être restaurée n'est pas une sauvegarde.

9.4.1 Restauration complète depuis PBS

Via l'interface graphique PVE :

1. Aller dans `Datacenter > Storage > pbs-techflow > Backups`
2. Sélectionner la sauvegarde souhaitée
3. Cliquer `Restore`
4. Choisir le nœud cible, le stockage de destination, et éventuellement un nouveau VMID
5. Cocher `Start after restore` si souhaité
6. Confirmer la restauration

> ✗ **DANGER**
> La restauration d'une VM ÉCRASE la VM existante si vous utilisez le même VMID. Avant toute restauration en production, vérifiez l'identifiant cible et créez un snapshot de la VM actuelle si elle est toujours fonctionnelle. En cas de doute, restaurez vers un VMID différent et comparez avant de basculer.

Via la CLI (méthode recommandée pour les scripts) :

```
# Lister les sauvegardes disponibles pour la VM 100
proxmox-backup-client list \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  --backup-type vm \
```

```
  --backup-id 100

# Restaurer la VM 100 depuis la dernière sauvegarde
qmrestore \
  pbs:backup/vm/100/2024-11-15T02:00:00Z \
  100 \
  --storage local-zfs \
  --force

# Restaurer vers un VMID différent (100 → 900, pour test)
qmrestore \
  pbs:backup/vm/100/2024-11-15T02:00:00Z \
  900 \
  --storage local-zfs

# Restaurer un conteneur LXC
pct restore \
  300 \
  pbs:backup/ct/300/2024-11-15T03:00:00Z \
  --storage local-zfs \
  --force

# Démarrer après restauration
qm start 900
```

Surveiller la progression de la restauration :

```
# Les tâches de restauration apparaissent dans les logs PVE
tail -f /var/log/pve/tasks/active

# Ou via l'API
pvesh get /nodes/pve1/tasks --limit 5
```

9.4.2 Restauration de fichiers individuels

L'un des avantages majeurs de PBS est la capacité de restaurer des fichiers individuels sans restaurer l'intégralité de la VM. Cette fonctionnalité repose sur le catalogue PBS.

Montage d'une sauvegarde PBS en lecture seule :

```
# Sur un nœud PVE ou sur pbs1 directement
# Monter la sauvegarde de la VM 100
proxmox-backup-client mount \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  vm/100/2024-11-15T02:00:00Z \
  /mnt/pbs-restore

# Lister le contenu monté
ls /mnt/pbs-restore/
# drive-scsi0.img  drive-scsi1.img  qemu-server.conf

# Monter l'image disque pour accéder aux fichiers
# Pour une image raw/qcow2 avec libguestfs :
guestmount -a /mnt/pbs-restore/drive-scsi0.img \
  -i \
  --ro \
  /mnt/vm-files

# Naviguer et copier les fichiers souhaités
ls /mnt/vm-files/
cp /mnt/vm-files/var/www/html/config.php /tmp/config-restored.php

# Démonter proprement
guestunmount /mnt/vm-files
proxmox-backup-client unmount /mnt/pbs-restore
```

Restauration de fichiers depuis l'interface PBS :

PBS 3.x propose une interface de navigation de fichiers directement dans la web UI :

1. Aller dans `Datastore > techflow-backup > Content`

2. Sélectionner la sauvegarde VM souhaitée
3. Cliquer sur `File Restore`
4. Naviguer dans l'arborescence de fichiers
5. Sélectionner les fichiers/dossiers à restaurer
6. Cliquer `Download` ou `Restore to VM`

Restauration d'une archive pxar (conteneurs LXC) :

```
# Extraire des fichiers d'un backup de conteneur LXC
proxmox-backup-client restore \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  ct/300/2024-11-15T03:00:00Z \
  root.pxar \
  /tmp/ct-restore/

# Accéder aux fichiers restaurés
ls /tmp/ct-restore/
```

9.4.3 Restauration sur un autre cluster

Dans un scénario de DR, vous devrez peut-être restaurer des VMs sur un cluster Proxmox VE différent (site secondaire). La procédure est similaire à une restauration classique, mais nécessite d'ajouter le PBS distant comme stockage sur le cluster cible.

Procédure complète :

```
# Sur le cluster PVE DR (pve-dr1.techflow-dr.lan)

# 1. Ajouter le PBS DR comme stockage
pvesm add pbs pbs-dr-store \
  --server 10.20.0.10 \
  --datastore techflow-dr \
  --username backup@pbs \
  --password "MotDePasseComplexe2024!" \
  --fingerprint "12:34:AB:CD: ... "

# 2. Vérifier que les sauvegardes sont visibles
pvesh get /nodes/pve-dr1/storage/pbs-dr-store/content

# 3. Restaurer la VM ERP (100) sur le cluster DR
qmrestore \
  pbs-dr-store:backup/vm/100/2024-11-15T02:00:00Z \
  100 \
  --storage local-zfs \
  --unique

# 4. Adapter la configuration réseau si nécessaire
qm set 100 --net0 virtio,bridge=vmbr0,tag=100

# 5. Démarrer la VM
qm start 100
```

> ✔ **CONSEIL**
> Utilisez l'option `--unique` lors d'une restauration sur un autre cluster. Elle régénère les identifiants uniques (UUID disque, adresse MAC) pour éviter les conflits si les deux sites sont en communication réseau.

9.4.4 Test de restauration (PRA simulé)

Un PRA non testé est un PRA qui échouera en production. Les tests de restauration doivent être planifiés et documentés régulièrement.

Checklist de test de restauration :

Checklist test de restauration mensuel

Étape	Action	Critère de succès
1	Identifier la sauvegarde à tester (J-1 ou J-7)	Sauvegarde visible dans PBS, statut OK

2	Restaurer vers un VMID de test (ex: 900)	Restauration complète sans erreur dans le RTO cible
3	Démarrer la VM restaurée sur réseau isolé	Boot réussi, services démarrés
4	Vérifier l'intégrité applicative	Application fonctionnelle, données cohérentes
5	Vérifier la date des dernières données	Delta inférieur au RPO cible
6	Documenter résultats et durée	Rapport de test archivé
7	Supprimer la VM de test	Aucune trace sur l'infrastructure prod

Script automatisé de test de restauration :

```
#!/bin/bash
# test-restore.sh - Script de test de restauration PBS
# TechFlow SAS - Usage: ./test-restore.sh <vmid> <test-vmid>

VMID=$1
TEST_VMID=$2
PBS_STORAGE="pbs-techflow"
TEST_STORAGE="local-zfs"
LOG_FILE="/var/log/restore-test-$(date +%Y%m%d).log"

echo "=== Test de restauration - $(date) ===" | tee -a "$LOG_FILE"
echo "VM source: $VMID → VM test: $TEST_VMID" | tee -a "$LOG_FILE"

# Récupérer la dernière sauvegarde disponible
LATEST=$(pvesh get /nodes/"$(hostname)"/storage/"$PBS_STORAGE"/content \
  --output-format json | \
  jq -r --arg vmid "$VMID" \
  '[.[] | select(.vmid == ($vmid | tonumber))] | sort_by(.ctime) | last | .volid')

echo "Sauvegarde sélectionnée: $LATEST" | tee -a "$LOG_FILE"

# Mesurer le temps de restauration
START_TIME=$(date +%s)
qmrestore "$LATEST" "$TEST_VMID" --storage "$TEST_STORAGE" --force 2>&1 \
  | tee -a "$LOG_FILE"
END_TIME=$(date +%s)

RESTORE_DURATION=$((END_TIME - START_TIME))
echo "Durée de restauration: ${RESTORE_DURATION}s" | tee -a "$LOG_FILE"

# Démarrer et vérifier
qm start "$TEST_VMID"
sleep 60

# Vérifier via guest agent
qm guest cmd "$TEST_VMID" ping 2>&1 | tee -a "$LOG_FILE"

echo "Test terminé. Suppression VM de test..." | tee -a "$LOG_FILE"
qm stop "$TEST_VMID" --skiplock
sleep 10
qm destroy "$TEST_VMID" --destroy-unreferenced-disks

echo "=== Fin du test - $(date) ===" | tee -a "$LOG_FILE"
```

9.5 Sauvegarde des configurations cluster

Les sauvegardes de VMs ne suffisent pas. La configuration du cluster Proxmox VE elle-même doit être sauvegardée pour permettre une reconstruction complète en cas de défaillance majeure.

9.5.1 /etc/pve : contenu et importance

Le répertoire `/etc/pve` est le cœur de la configuration Proxmox VE. Il est géré par **pmxcfs** (Proxmox Cluster File System), un système de fichiers distribué basé sur Corosync qui synchronise automatiquement son contenu sur tous les nœuds du cluster.

Contenu de `/etc/pve` :

```
/etc/pve/
├── corosync.conf          # Configuration du cluster Corosync
├── authkey                # Clé d'authentification cluster
├── priv/
│   ├── authorized_keys    # Clés SSH inter-nœuds
│   └── shadow.cfg         # Mots de passe utilisateurs
├── datacenter.cfg         # Configuration globale datacenter
├── storage.cfg            # Configuration des stockages
├── user.cfg               # Utilisateurs et permissions
├── domains.cfg            # Domaines d'authentification
├── jobs.cfg               # Jobs de sauvegarde planifiés
├── ha/
│   ├── groups.cfg         # Groupes HA
│   └── resources.cfg      # Ressources HA
├── nodes/
│   └── pve1/
│       ├── config         # Config spécifique au nœud
│       └── lxc/           # Configs des conteneurs LXC
│           └── 300.conf
└── qemu-server/           # Configs des VMs QEMU
    ├── 100.conf
    ├── 101.conf
    └── 102.conf
```

⚠ AVERTISSEMENT

Le répertoire `/etc/pve/priv/` contient des données sensibles (clés privées, mots de passe hashés). Les sauvegardes de ce répertoire doivent être chiffrées et leur accès strictement contrôlé.

9.5.2 Sauvegarde automatisée des configs

Script de sauvegarde des configurations :

```
#!/bin/bash
# backup-pve-config.sh - Sauvegarde quotidienne des configurations PVE
# TechFlow SAS - Placer dans /usr/local/sbin/
# Crontab: 0 1 * * * /usr/local/sbin/backup-pve-config.sh

BACKUP_DIR="/var/backups/pve-configs"
REMOTE_BACKUP="backup@10.10.0.10:/backup/pve-configs/"
DATE=$(date +%Y%m%d-%H%M%S)
ARCHIVE="${BACKUP_DIR}/pve-config-${DATE}.tar.gz.enc"
ENCRYPT_KEY="/etc/proxmox-backup/config-backup.key"

mkdir -p "$BACKUP_DIR"

# Créer l'archive tar chiffrée
tar -czp \
  --exclude=/etc/pve/priv/authorized_keys \
  -C / \
  etc/pve \
  etc/network/interfaces \
  etc/hosts \
  etc/hostname \
  etc/resolv.conf | \
  openssl enc -aes-256-cbc -pbkdf2 -pass file:"$ENCRYPT_KEY" \
  -out "$ARCHIVE"

echo "Archive créée: $ARCHIVE ($(du -sh "$ARCHIVE" | cut -f1))"

# Copier vers PBS ou stockage distant
rsync -az --delete \
  "$BACKUP_DIR/" \
  "$REMOTE_BACKUP"

# Nettoyer les archives de plus de 90 jours
find "$BACKUP_DIR" -name "pve-config-*.tar.gz.enc" -mtime +90 -delete
```

```
echo "Sauvegarde configuration PVE terminée: $(date)"
```

Sauvegarde dans un dépôt Git pour le suivi des changements :

```
# Initialiser un dépôt Git pour la configuration PVE
mkdir -p /var/backups/pve-git-config
cd /var/backups/pve-git-config && git init

cat > /usr/local/sbin/pve-config-git-commit.sh << 'EOF'
#!/bin/bash
REPO="/var/backups/pve-git-config"
cp -r /etc/pve/qemu-server/ "$REPO/"
cp -r /etc/pve/lxc/ "$REPO/" 2>/dev/null || true
cp /etc/pve/storage.cfg "$REPO/"
cp /etc/pve/datacenter.cfg "$REPO/"
cp /etc/pve/user.cfg "$REPO/"
cp /etc/pve/jobs.cfg "$REPO/"

cd "$REPO"
git add -A
git commit -m "Config backup $(date +'%Y-%m-%d %H:%M:%S')" \
  --author="PBS Auto <backup@techflow.lan>"
git push origin main 2>/dev/null || true
EOF

chmod +x /usr/local/sbin/pve-config-git-commit.sh

# Planifier dans cron toutes les 6 heures
echo "0 */6 * * * root /usr/local/sbin/pve-config-git-commit.sh" \
  > /etc/cron.d/pve-config-backup
```

9.5.3 Restauration d'un nœud from scratch

Dans le cas où un nœud Proxmox VE est complètement perdu (défaillance matérielle, corruption système), voici la procédure de reconstruction.

Scénario : perte totale de pve1.techflow.lan

```
# Étape 1 : Réinstaller Proxmox VE depuis l'ISO
# (via IPMI/iDRAC ou support physique)
# Version : Proxmox VE 9.x (même version que le cluster)

# Étape 2 : Après l'installation initiale, NE PAS créer de cluster
# Se connecter en SSH sur le nouveau pve1

# Étape 3 : Restaurer la configuration réseau
scp root@10.10.0.10:/backup/pve-configs/latest/etc/network/interfaces \
  /etc/network/interfaces
systemctl restart networking

# Étape 4 : Rejoindre le cluster existant
# Sur un nœud existant (pve2), noter le fingerprint
pvecm status

# Sur le nouveau pve1 :
pvecm add pve2.techflow.lan \
  --use-ssh \
  --ring0_addr 10.10.0.11

# Étape 5 : Vérifier la synchronisation pmxcfs
pvecm status
ls /etc/pve/qemu-server/

# Étape 6 : Recréer le pool ZFS local si nécessaire
zpool create -f rpool \
  mirror \
  /dev/sda \
  /dev/sdb
```

```
# Étape 7 : Restaurer les VMs qui résidaient sur ce nœud
qmrestore pbs-techflow:backup/vm/100/latest 100 \
  --storage local-zfs \
  --force

# Étape 8 : Vérifier l'état HA si applicable
ha-manager status
```

9.6 Plan de reprise après sinistre

9.6.1 Scénarios de sinistre

Un bon PRA doit anticiper plusieurs scénarios avec des niveaux d'impact différents. Voici les scénarios retenus pour TechFlow SAS :

Scénario	Probabilité	Impact	RTO visé	RPO visé
S1 : Panne d'un nœud PVE	Élevée	Faible (HA)	< 5 min (HA auto)	0 (live migration)
S2 : Corruption d'une VM	Moyenne	Moyen	< 2 heures	< 1 heure
S3 : Perte d'un disque de stockage	Moyenne	Moyen	< 4 heures	< 4 heures
S4 : Perte totale du site primaire	Faible	Critique	< 8 heures	< 4 heures
S5 : Ransomware chiffrant les données	Faible	Critique	< 24 heures	< 24 heures
S6 : Erreur humaine (suppression accidentelle)	Moyenne	Variable	< 2 heures	< RPO sauvegarde

S1 — Panne d'un nœud PVE : Ce scénario est géré automatiquement par le cluster HA de Proxmox. Les VMs configurées en HA redémarrent automatiquement sur les nœuds restants. Aucune intervention manuelle n'est nécessaire (cf. Chapitre 8).

S5 — Ransomware : C'est le scénario le plus redouté. La stratégie de protection passe par :

- Sauvegardes PBS avec chiffrement côté client (le ransomware ne peut pas rechiffrer ce qui est déjà chiffré de manière asymétrique)
- PBS en lecture seule depuis les nœuds PVE (les sauvegardes ne peuvent pas être supprimées par les VMs compromises)
- Rétention longue pour avoir un point de restauration avant l'infection
- PBS remote hors réseau production (segment réseau dédié ou VPN)

9.6.2 Réplication de VM (storage replication)

La réplication de stockage Proxmox (storage replication) permet de maintenir une copie synchrone ou quasi-synchrone des disques VM sur un autre nœud ou un autre site. Elle s'appuie sur ZFS send/receive ou sur la réplication Ceph.

Configuration de la réplication ZFS :

```
# Configurer la réplication pour la VM 100 vers pve2
# Interface : VM 100 > Replication > Add

# Via CLI :
pvesr create-local-job 1000 \
  --guest 100 \
  --target pve2 \
  --schedule "*/15" \
  --comment "Réplication ERP vers pve2"

# Lister les jobs de réplication
pvesr list

# Vérifier le statut
pvesr status

# Forcer une synchronisation immédiate
pvesr run 1000
```

Réplication vers le site DR via ZFS send/receive :

```
# Snapshot périodique et envoi vers le site DR
# Sur pve1.techflow.lan :
zfs snapshot rpool/data/vm-100-disk-0@repl-$(date +%Y%m%d%H%M)

# Envoi incrémental vers le serveur DR
zfs send -i rpool/data/vm-100-disk-0@prev \
         rpool/data/vm-100-disk-0@repl-$(date +%Y%m%d%H%M) | \
  ssh root@10.20.0.11 zfs receive drpool/data/vm-100-disk-0
```

> ✔ **CONSEIL**
> Pour des RPO très bas (inférieurs à 15 minutes), combinez la réplication Proxmox native avec les sauvegardes PBS. La réplication assure la continuité, la sauvegarde PBS assure la protection contre la corruption et les erreurs humaines.

9.6.3 Site de DR avec PBS remote

L'architecture DR de TechFlow SAS repose sur un site secondaire avec un PBS remote. Les sauvegardes sont synchronisées depuis le PBS primaire (10.10.0.10) vers le PBS DR (10.20.0.10) toutes les heures.

Architecture DR TechFlow SAS :

```
Site Primaire (Paris)              Site DR (Lyon)
──────────────────────             ──────────────────

pve1: 10.10.0.11                   pve-dr1: 10.20.0.11
pve2: 10.10.0.12                   pve-dr2: 10.20.0.12
pve3: 10.10.0.13
pbs1: 10.10.0.10 ──────sync──→   pbs-dr: 10.20.0.10
                    (toutes les h)
VMs actives :                      VMs en veille / non démarrées
  vm-erp-01 (100)                  Disponibles via PBS-DR
  vm-db-01 (101)
  vm-web-01 (102)
```

Configuration PBS DR sur le site secondaire :

```
# Sur pbs-dr.techflow-dr.lan (10.20.0.10)

# Installation identique à PBS primaire
apt install proxmox-backup-server -y

# Créer le datastore DR
zpool create -f dr-data mirror /dev/nvme0n1 /dev/nvme1n1
zfs create dr-data/backups

proxmox-backup-manager datastore create techflow-dr \
  --path /dr-data/backups \
  --comment "Datastore DR TechFlow SAS - Site Lyon"

# Créer l'utilisateur pour la synchronisation
proxmox-backup-manager user create backup@pbs \
  --password "MotDePasseComplexe2024!"

proxmox-backup-manager acl update /datastore/techflow-dr \
  --auth-id backup@pbs \
  --role DatastoreAdmin
```

Ajouter le cluster PVE DR au PBS DR :

```
# Sur pve-dr1.techflow-dr.lan
pvesm add pbs pbs-dr \
  --server 10.20.0.10 \
  --datastore techflow-dr \
  --username backup@pbs \
  --password "MotDePasseComplexe2024!" \
  --fingerprint "DR:12:34: ... "
```

```
# Vérifier les sauvegardes disponibles
pvesh get /nodes/pve-dr1/storage/pbs-dr/content
```

9.6.4 Runbook de bascule documenté

Un runbook est un document opérationnel décrivant pas à pas les actions à mener en cas de déclenchement du PRA. Il doit être accessible hors du système primaire (version papier, stockage cloud, etc.).

Runbook TechFlow SAS — Bascule site DR

```
RUNBOOK PRA TECHFLOW SAS
Version: 2.0 | Dernière MAJ: 2024-11-01
Auteur: Équipe Infrastructure
Accès: https://wiki.techflow.lan/pra | Copie papier: Coffre salle serveur

DÉCLENCHEMENT DU PRA
═══════════════════════════════════════════════════
Critères de déclenchement :
- Indisponibilité site primaire > 2 heures
- Décision validée par DSI + Responsable Infra
- Notification équipes métier AVANT bascule

ÉTAPE 1 : ÉVALUATION (0 - 30 min)
═══════════════════════════════════════════════════
[ ] Confirmer l'indisponibilité totale du site primaire
[ ] Estimer la durée estimée d'interruption
[ ] Contacter prestataire datacenter primaire
[ ] Notifier la direction (DSI, PDG)
[ ] Activer l'astreinte DR (liste : contacts-dr.txt)

ÉTAPE 2 : ACTIVATION CLUSTER DR (30 - 90 min)
═══════════════════════════════════════════════════
Accès SSH cluster DR :
  ssh root@10.20.0.11  # pve-dr1
  ssh root@10.20.0.12  # pve-dr2

[ ] Vérifier état cluster DR
    pvecm status

[ ] Vérifier synchronisation PBS DR
    proxmox-backup-manager show-task <last-sync-task>

[ ] Identifier la dernière sauvegarde disponible
    pvesh get /nodes/pve-dr1/storage/pbs-dr/content \
      --output-format table | grep vm/100

[ ] Calculer le delta RPO réel (date dernière sauvegarde vs maintenant)

ÉTAPE 3 : RESTAURATION VMs CRITIQUES (90 - 240 min)
═══════════════════════════════════════════════════
Ordre de restauration (priorité décroissante) :

1. vm-dns-01 (VMID 200) → Infrastructure DNS
   qmrestore pbs-dr:backup/vm/200/latest 200 \
     --storage local-zfs --force
   qm start 200

2. vm-db-01 (VMID 101) → Base de données
   qmrestore pbs-dr:backup/vm/101/latest 101 \
     --storage local-zfs --force
   qm start 101

3. vm-erp-01 (VMID 100) → ERP
   qmrestore pbs-dr:backup/vm/100/latest 100 \
     --storage local-zfs --force
   qm start 100

4. vm-web-01 (VMID 102) → Serveur web
```

```
   qmrestore pbs-dr:backup/vm/102/latest 102 \
     --storage local-zfs --force
   qm start 102

ÉTAPE 4 : REDIRECTION DNS / RÉSEAU (240 - 300 min)
═══════════════════════════════════════════════════
[ ] Mettre à jour les enregistrements DNS publics
    erp.techflow.fr → IP DR
    www.techflow.fr → IP DR
    TTL : réduire à 60s AVANT sinistre (si possible)

[ ] Activer le VPN site DR pour les utilisateurs distants
[ ] Informer les utilisateurs (email / messagerie)

ÉTAPE 5 : VALIDATION ET REPRISE D'ACTIVITÉ
═══════════════════════════════════════════════════
[ ] Tests fonctionnels (liste : tests-acceptance.xlsx)
[ ] Validation DSI
[ ] Communication retour à la normale

RETOUR AU SITE PRIMAIRE (après rétablissement)
═══════════════════════════════════════════════════
ATTENTION : Ne JAMAIS avoir deux instances de la même VM
actives simultanément (risque de split-brain applicatif).

[ ] Arrêter les VMs DR avant de redémarrer les VMs primaires
[ ] Sauvegarder les données produites sur le site DR
[ ] Synchroniser les données vers le site primaire
[ ] Tester le site primaire avant bascule retour
```

9.7 Lab TechFlow SAS : PBS complet

Dans cette section pratique, nous mettons en œuvre l'ensemble de la stratégie de sauvegarde de TechFlow SAS de bout en bout.

9.7.1 Architecture PBS dédié (pbs1.techflow.lan)

Infrastructure cible :

```
pbs1.techflow.lan
IP: 10.10.0.10
OS: Debian 12 Bookworm + proxmox-backup-server
CPU: Intel Xeon E-2224 (4 cœurs)
RAM: 16 Go DDR4 ECC
OS Disk: 2x 128 Go SSD (RAID1 logiciel mdadm)
Data Disk: 2x 2 To NVMe (ZFS mirror)
Réseau: 10 GbE vers le cluster PVE
Datastore: techflow-backup (chiffrement AES-256)
```

Étape 1 : Préparation du système

```
# Sur pbs1.techflow.lan après installation Debian 12 minimale

# Configurer le réseau
cat > /etc/network/interfaces << 'NETCONF'
auto lo
iface lo inet loopback

auto ens18
iface ens18 inet static
    address 10.10.0.10/24
    gateway 10.10.0.1
    dns-nameservers 10.10.0.53 8.8.8.8
    dns-search techflow.lan
NETCONF

systemctl restart networking
ping -c 3 10.10.0.1
```

```
# Configurer le hostname
hostnamectl set-hostname pbs1.techflow.lan
echo "10.10.0.10 pbs1.techflow.lan pbs1" >> /etc/hosts

# Mettre à jour le système
apt update && apt dist-upgrade -y && apt autoremove -y
```

Étape 2 : Créer le pool ZFS pour les données

```
# Identifier les disques NVMe
ls -la /dev/nvme*
# /dev/nvme0n1  /dev/nvme1n1

# Créer le pool ZFS en miroir avec les options optimales
zpool create -f \
  -o ashift=12 \
  -O compression=lz4 \
  -O atime=off \
  -O xattr=sa \
  -O dnodesize=auto \
  techflow-data \
  mirror \
  /dev/nvme0n1 \
  /dev/nvme1n1

# Vérifier
zpool status techflow-data
zpool list

# Créer le dataset pour PBS avec un recordsize adapté
zfs create -o mountpoint=/backup -o recordsize=128k techflow-data/backup

# Vérifier le montage
df -h /backup
# Filesystem      Size  Used Avail Use% Mounted on
# techflow-data/backup  1.8T     0  1.8T   0% /backup
```

Étape 3 : Installer et configurer PBS

```
# Ajouter le dépôt PBS
wget https://enterprise.proxmox.com/debian/proxmox-release-bookworm.gpg \
  -O /etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg

echo "deb http://download.proxmox.com/debian/pbs bookworm pbs-no-subscription" \
  > /etc/apt/sources.list.d/pbs.list

# Installer PBS
apt update && apt install proxmox-backup-server -y

# Vérifier le service
systemctl status proxmox-backup
# Active: active (running) since ...

# Configurer le firewall (optionnel mais recommandé)
ufw allow 8007/tcp comment "PBS Web UI"
ufw allow ssh
ufw enable

echo "Interface PBS disponible sur https://10.10.0.10:8007"
echo "Identifiants : root@pam / [mot de passe root système]"
```

Étape 4 : Créer le datastore techflow-backup

```
# Créer le datastore dans PBS avec rétention configurée
proxmox-backup-manager datastore create techflow-backup \
  --path /backup \
  --comment "Datastore principal TechFlow SAS - Paris" \
  --keep-last 0 \
  --keep-hourly 24 \
  --keep-daily 30 \
```

```
  --keep-weekly 8 \
  --keep-monthly 12 \
  --keep-yearly 2

# Vérifier
proxmox-backup-manager datastore list

# Créer les namespaces organisationnels
proxmox-backup-manager namespace create techflow-backup --ns production
proxmox-backup-manager namespace create techflow-backup --ns infrastructure
proxmox-backup-manager namespace create techflow-backup --ns developpement

# Créer l'utilisateur de sauvegarde dédié
proxmox-backup-manager user create backup@pbs \
  --comment "Compte sauvegarde cluster PVE TechFlow"
proxmox-backup-manager user update backup@pbs \
  --password "Tr0pS3cur3_PBSBackup2024!"

# Attribuer les droits par namespace
proxmox-backup-manager acl update \
  /datastore/techflow-backup/production \
  --auth-id backup@pbs \
  --role DatastoreBackup

proxmox-backup-manager acl update \
  /datastore/techflow-backup/infrastructure \
  --auth-id backup@pbs \
  --role DatastoreBackup

proxmox-backup-manager acl update \
  /datastore/techflow-backup/developpement \
  --auth-id backup@pbs \
  --role DatastoreBackup

# Activer la vérification automatique des nouvelles sauvegardes
proxmox-backup-manager datastore update techflow-backup \
  --verify-new true
```

Étape 5 : Configurer le chiffrement AES-256 depuis PVE

```
# Sur pve1.techflow.lan (nœud PVE, PAS sur PBS)
mkdir -p /etc/proxmox-backup

# Générer la clé de chiffrement AES-256
proxmox-backup-client key create \
  /etc/proxmox-backup/techflow-encryption.key
# Enter new passphrase: [saisir une passphrase longue et complexe]
# Confirm passphrase: [confirmer]

# Afficher la clé pour vérification
proxmox-backup-client key show \
  /etc/proxmox-backup/techflow-encryption.key

# SAUVEGARDE OBLIGATOIRE de la clé hors serveur
# Option 1 : Copie sécurisée vers PBS (répertoire admin)
scp /etc/proxmox-backup/techflow-encryption.key \
  root@10.10.0.10:/root/key-backup/techflow-encryption-$(date +%Y%m%d).key

# Option 2 : Impression papier de la clé base64 (à conserver en coffre)
proxmox-backup-client key show \
  /etc/proxmox-backup/techflow-encryption.key \
  --output-format json | base64 > /tmp/key-print.txt
# Imprimer /tmp/key-print.txt et stocker en sécurité
shred -u /tmp/key-print.txt

# Récupérer le fingerprint PBS pour la connexion sécurisée
PBS_FINGERPRINT=$(ssh root@10.10.0.10 \
  "proxmox-backup-manager cert info" | \
```

```
  grep -oP 'Fingerprint.*: \K[A-F0-9:]+')

echo "Fingerprint PBS: $PBS_FINGERPRINT"

# Ajouter PBS comme stockage dans PVE avec chiffrement
pvesm add pbs pbs-techflow \
  --server 10.10.0.10 \
  --datastore techflow-backup \
  --username backup@pbs \
  --password "Tr0pS3cur3_PBSBackup2024!" \
  --fingerprint "$PBS_FINGERPRINT" \
  --namespace production \
  --encryption-key /etc/proxmox-backup/techflow-encryption.key

# Vérifier le stockage
pvesm status | grep pbs-techflow
# pbs-techflow   pbs   active   3932160 MB    ...
```

9.7.2 Configuration des jobs de sauvegarde

Création des jobs de sauvegarde dans PVE :

```
# Sur pve1.techflow.lan
# Éditer /etc/pve/jobs.cfg pour ajouter les jobs TechFlow

cat >> /etc/pve/jobs.cfg << 'EOF'

vzdump: techflow-erp-db-hourly
    storage pbs-techflow
    schedule */1:00
    mode snapshot
    enabled 1
    vmid 100,101
    mailnotification failure
    mailto admin@techflow.lan
    notes-template "Auto backup {{guestname}} - {{date}}"
    compress zstd

vzdump: techflow-web-6h
    storage pbs-techflow
    schedule 00:00/6:00
    mode snapshot
    enabled 1
    vmid 102,103,104
    mailnotification failure
    mailto admin@techflow.lan
    compress zstd

vzdump: techflow-infra-daily
    storage pbs-techflow
    schedule 03:30
    mode snapshot
    enabled 1
    vmid 200,201,300,301
    namespace infrastructure
    mailnotification always
    mailto admin@techflow.lan

vzdump: techflow-files-daily
    storage pbs-techflow
    schedule 23:00
    mode snapshot
    enabled 1
    vmid 110
    mailnotification always
    mailto admin@techflow.lan

vzdump: techflow-dev-weekly
    storage pbs-techflow
```

```
    schedule sun 04:00
    mode stop
    enabled 1
    vmid 400,401,402
    namespace developpement
    mailnotification failure
EOF

# Vérifier la syntaxe
pvesh get /cluster/jobs/vzdump
```

Déclenchement et surveillance des jobs :

```
# Lancer un job de sauvegarde manuellement pour tester
vzdump 100 \
  --storage pbs-techflow \
  --mode snapshot \
  --compress zstd \
  --notes-template "Test initial - $(date +%Y-%m-%d)"

# Surveiller en temps réel
tail -f /var/log/syslog | grep vzdump &
journalctl -fu pvedaemon | grep -E "(backup|vzdump)" &

# Vérifier les sauvegardes créées
proxmox-backup-client list \
  --repository "backup@pbs@10.10.0.10:techflow-backup" \
  --output-format table

# Vérifier la déduplication après quelques sauvegardes
proxmox-backup-manager datastore show techflow-backup
```

Configurer les verify jobs :

```
# Sur pbs1.techflow.lan
proxmox-backup-manager verify-job create \
  --id verify-prod-weekly \
  --datastore techflow-backup \
  --ns production \
  --schedule "sat 06:00" \
  --ignore-verified false \
  --outdated-after 7 \
  --comment "Vérification hebdomadaire production"

proxmox-backup-manager verify-job create \
  --id verify-full-monthly \
  --datastore techflow-backup \
  --schedule "1st sun 07:00" \
  --ignore-verified false \
  --outdated-after 30 \
  --comment "Vérification mensuelle complète datastore"

# Configurer le Garbage Collection automatique
# Dans l'interface PBS : Datastore > techflow-backup > GC Schedule
# Ou via l'API REST de PBS

proxmox-backup-manager verify-job list
```

Configurer la synchronisation vers PBS DR :

```
# Sur pbs1.techflow.lan

# Récupérer le fingerprint du PBS DR
DR_FP=$(ssh root@10.20.0.10 \
  "proxmox-backup-manager cert info" | \
  grep -oP 'Fingerprint.*: \K[A-F0-9:]+')

# Créer la configuration du remote DR
proxmox-backup-manager remote create pbs-dr-lyon \
  --host 10.20.0.10 \
```

```
  --port 8007 \
  --userid backup@pbs \
  --password "Tr0pS3cur3_PBSBackup2024!" \
  --fingerprint "$DR_FP"

# Tester la connexion
proxmox-backup-manager remote show pbs-dr-lyon

# Créer le sync job vers DR (toutes les heures, 30 min après les sauvegardes)
proxmox-backup-manager sync-job create \
  --id sync-prod-to-dr \
  --remote pbs-dr-lyon \
  --remote-store techflow-dr \
  --local-store techflow-backup \
  --remote-ns production \
  --ns production \
  --schedule "*/1:30" \
  --remove-vanished false \
  --comment "Sync production vers DR Lyon"

proxmox-backup-manager sync-job create \
  --id sync-infra-to-dr \
  --remote pbs-dr-lyon \
  --remote-store techflow-dr \
  --local-store techflow-backup \
  --remote-ns infrastructure \
  --ns infrastructure \
  --schedule "04:30" \
  --remove-vanished false \
  --comment "Sync infrastructure vers DR Lyon"

# Lancer la première synchronisation complète
echo "Lancement de la synchronisation initiale (peut prendre plusieurs heures)..."
proxmox-backup-manager sync-job run sync-prod-to-dr

# Surveiller la progression
watch -n 10 'proxmox-backup-manager task list --limit 5'
```

9.7.3 Test de restauration documenté

Scénario de test : restauration complète de vm-erp-01 (VMID 100)

```
# ═══════════════════════════════════════════════════════════
# TEST DE RESTAURATION PBS — TechFlow SAS
# Date : 2024-11-20 | Responsable : Admin Infra
# Objectif : Valider RTO < 2h et RPO < 1h pour vm-erp-01
# ═══════════════════════════════════════════════════════════

echo "═══ DÉBUT TEST RESTAURATION — $(date) ═══" | tee /var/log/restore-test-
erp.log

# ÉTAPE 1 : Identifier la sauvegarde de J-1
echo "--- Sauvegardes disponibles pour VM 100 ---" | tee -a /var/log/restore-test-
erp.log

proxmox-backup-client list \
  --repository "backup@pbs@10.10.0.10:techflow-backup" \
  --backup-type vm \
  --backup-id 100 \
  --output-format table 2>&1 | tee -a /var/log/restore-test-erp.log

# Sélectionner manuellement la sauvegarde de J-1
BACKUP_TS="2024-11-19T02:00:01Z"
BACKUP_REF="pbs-techflow:backup/vm/100/${BACKUP_TS}"
echo "Sauvegarde sélectionnée: $BACKUP_REF" | tee -a /var/log/restore-test-erp.log

# ÉTAPE 2 : Restauration vers VMID de test
echo "--- Démarrage restauration vers VMID 900 ---" | tee -a /var/log/restore-
test-erp.log
```

```
START_RESTORE=$(date +%s)

qmrestore "$BACKUP_REF" 900 \
  --storage local-zfs 2>&1 | tee -a /var/log/restore-test-erp.log

END_RESTORE=$(date +%s)
RESTORE_SEC=$((END_RESTORE - START_RESTORE))
echo "MÉTRIQUE: Durée restauration = ${RESTORE_SEC}s ($(( RESTORE_SEC / 60 ))min)"
\
  | tee -a /var/log/restore-test-erp.log

# ÉTAPE 3 : Configurer réseau isolé et démarrer
qm set 900 \
  --name "vm-erp-01-RESTORE-TEST-$(date +%Y%m%d)" \
  --net0 virtio,bridge=vmbr-test,firewall=0

qm start 900

echo "Attente démarrage VM (60s)..." | tee -a /var/log/restore-test-erp.log
sleep 60

# ÉTAPE 4 : Vérifications via guest agent
echo "--- Vérifications applicatives ---" | tee -a /var/log/restore-test-erp.log

# Vérifier que la VM répond
qm status 900 | tee -a /var/log/restore-test-erp.log

# Vérifier les services
qm guest exec 900 -- systemctl is-active erp-app 2>&1 \
  | tee -a /var/log/restore-test-erp.log

# Vérifier la date des dernières données (delta RPO)
qm guest exec 900 -- \
  bash -c "mysql -u root -p'dbpass' erp_db \
  -e \"SELECT MAX(updated_at) as derniere_transaction FROM orders;\" 2>/dev/null"
\
  2>&1 | tee -a /var/log/restore-test-erp.log

# ÉTAPE 5 : Résultats et nettoyage
TOTAL_SEC=$(($(date +%s) - START_RESTORE))
echo "=== RÉSULTATS TEST ===" | tee -a /var/log/restore-test-erp.log
echo "RTO mesuré : $((TOTAL_SEC / 60)) minutes (objectif : 120 min)" | tee -a /
var/log/restore-test-erp.log
echo "Statut : $([ $TOTAL_SEC -lt 7200 ] && echo SUCCÈS || echo ÉCHEC)" \
  | tee -a /var/log/restore-test-erp.log

# Nettoyage
qm stop 900 --skiplock
sleep 15
qm destroy 900 --destroy-unreferenced-disks --purge

echo "=== FIN TEST RESTAURATION - $(date) ===" | tee -a /var/log/restore-test-
erp.log
```

Rapport de test de restauration TechFlow SAS :

Critère	Objectif	Résultat mesuré	Statut
Durée restauration (45 Go)	< 120 min	22 minutes	OK
Démarrage complet VM	< 5 min	2 min 30 s	OK
Services applicatifs actifs	100%	100%	OK
Delta de données (RPO réel)	< 1 heure	58 minutes	OK
Intégrité base de données	0 erreur	0 erreur	OK
Chiffrement AES-256 vérifié	Oui	Confirmé	OK
Test exécuté sur réseau isolé	Oui	Oui	OK

9.7.4 Simulation de sinistre et bascule

Scénario simulé : perte totale du site primaire Paris

Cette simulation est réalisée en environnement de laboratoire le week-end, en dehors des heures ouvrées, avec notification préalable de toutes les équipes concernées.

```
# ═══════════════════════════════════════════════════════════
# SIMULATION SINISTRE TECHFLOW SAS – BASCULE SITE DR
# Date : Samedi 2024-11-23 | Déclenchement simulé : 08h45
# ═══════════════════════════════════════════════════════════

# ═══ T+30 min : CONNEXION AU CLUSTER DR ═══

ssh root@10.20.0.11  # pve-dr1.techflow-dr.lan

# Vérifier l'état du cluster DR
pvecm status
# Cluster information:
# Quorum information: quorate Yes, 2 nodes

# Vérifier la dernière synchronisation PBS
proxmox-backup-manager task list \
  --typefilter sync \
  --limit 5

# Résultat : Dernière sync à 08:30 (15 min avant le sinistre simulé)
# RPO réel : 15 minutes

# Identifier les sauvegardes disponibles
pvesh get /nodes/pve-dr1/storage/pbs-dr/content \
  --output-format table

# ═══ T+45 min : RESTAURATION DANS L'ORDRE DE PRIORITÉ ═══

# 1. DNS (infrastructure critique – premier à restaurer)
echo "[T+45min] Restauration vm-dns-01 ..."
qmrestore pbs-dr:backup/vm/200/latest 200 --storage local-zfs --force
qm set 200 --net0 virtio,bridge=vmbr0,tag=20
qm start 200
sleep 45

# Valider DNS
dig @10.20.0.200 pbs-dr.techflow-dr.lan +short
# 10.20.0.10 (réponse attendue)

# 2. Base de données MySQL
echo "[T+1h15] Restauration vm-db-01 ..."
qmrestore pbs-dr:backup/vm/101/latest 101 --storage local-zfs --force
qm set 101 --net0 virtio,bridge=vmbr0,tag=101
qm start 101
sleep 90

# Valider MySQL
qm guest exec 101 -- bash -c \
  "mysqladmin -u root -p'dbpass' status 2>/dev/null | grep Uptime"

# 3. ERP
echo "[T+2h] Restauration vm-erp-01 ..."
qmrestore pbs-dr:backup/vm/100/latest 100 --storage local-zfs --force
qm set 100 \
  --net0 virtio,bridge=vmbr0,tag=100 \
  --name vm-erp-01
qm start 100
sleep 120

# Valider ERP
curl -sk --max-time 30 https://10.20.1.100/api/health | \
  python3 -c "import sys,json; d=json.load(sys.stdin); print('OK' if
```

```
d.get('status')=='healthy' else 'KO')"

# 4. Serveurs web
echo "[T+2h30] Restauration vm-web-01 ..."
qmrestore pbs-dr:backup/vm/102/latest 102 --storage local-zfs --force
qm set 102 --net0 virtio,bridge=vmbr0,tag=102
qm start 102

# ═══ T+3h : BILAN DE LA SIMULATION ═══
echo ""
echo "═══════════════════════════════════════"
echo "  BILAN SIMULATION PRA TECHFLOW SAS"
echo "  $(date)"
echo "═══════════════════════════════════════"
echo ""
echo "RPO réel atteint : 15 minutes"
echo "  (Objectif : < 4 heures) → EXCELLENT"
echo ""
echo "RTO total atteint : 2h45"
echo "  (Objectif : < 8 heures) → CONFORME"
echo ""
echo "Services restaurés :"
qm list | grep -E "(100|101|102|200)" | awk '{print "  VM", $1, $2, "-", $3}'
echo ""
echo "Points d amélioration identifiés :"
echo "  1. Automatiser le script de bascule (réduire RTO de 30min)"
echo "  2. Pré-configurer les VMs DR avec le bon réseau"
echo "  3. Réduire TTL DNS publics à 60s en permanence"
echo "  4. Documenter le test de récupération MySQL avant bascule ERP"
echo ""
echo "Prochaine simulation planifiée : 2025-05-24"
```

Retour au site primaire après simulation :

```
# Procédure de retour propre (site primaire rétabli)
# IMPORTANT: NE PAS démarrer les VMs primaires tant que les DR sont actives

# 1. Sauvegarder les données produites sur le DR pendant la simulation
for vmid in 100 101 102 200; do
  echo "Sauvegarde finale VM $vmid avant retour ..."
  vzdump $vmid \
    --storage pbs-dr \
    --mode snapshot \
    --notes-template "Données DR avant retour primaire - $(date +%Y-%m-%d)"
done

# 2. Arrêter toutes les VMs DR
qm stop 100 && qm stop 101 && qm stop 102 && qm stop 200

# 3. Attendre l'arrêt complet
sleep 30
qm list | grep -E "(100|101|102|200)"

# 4. Sur le site primaire, restaurer depuis les données DR si nécessaire
# Sur pve1.techflow.lan :
# pvesm add pbs pbs-dr-temp --server 10.20.0.10 ...
# qmrestore pbs-dr-temp:backup/vm/100/latest 100 --storage ceph-pool --force

# 5. Vérification finale
echo "Retour site primaire effectué - $(date)"
echo "Rapport complet disponible : /var/log/pra-simulation-20241123.log"
```

Résumé du chapitre

Ce chapitre a couvert l'ensemble de la stratégie de sauvegarde et de reprise après sinistre pour Proxmox VE 9, en s'appuyant sur l'infrastructure de TechFlow SAS comme fil conducteur.

Points clés à retenir :

- La **règle 3-2-1** s'applique naturellement à Proxmox avec PBS local + PBS remote comme deuxième et troisième copies de données.
- Le **RPO et le RTO** sont des objectifs métier qui dictent la fréquence des sauvegardes et les investissements en infrastructure DR.
- **Proxmox Backup Server** est la solution native avec déduplication par chunks SHA-256, chiffrement AES-256-GCM côté client, vérification d'intégrité automatisée et synchronisation PBS-to-PBS efficace.
- Les **politiques de rétention granulaires** (horaire, journalier, hebdomadaire, mensuel, annuel) permettent d'optimiser l'espace tout en respectant les contraintes de conformité.
- La **restauration de fichiers individuels** sans restauration complète de la VM est l'une des fonctionnalités différenciantes de PBS par rapport aux solutions génériques.
- La **sauvegarde des configurations** `/etc/pve` est aussi importante que la sauvegarde des données : elle permet de reconstruire un nœud from scratch en quelques dizaines de minutes.
- Un **PRA sans runbook documenté** est un PRA qui échouera sous stress. Le runbook doit être accessible hors du système primaire (version papier, stockage cloud indépendant).
- Les **tests de restauration réguliers** (mensuels minimum) sont la seule façon de valider réellement que votre stratégie de sauvegarde est opérationnelle.

Commandes essentielles :

```
# Sauvegarde manuelle
vzdump <vmid> --storage pbs-techflow --mode snapshot --compress zstd

# Restauration complète VM
qmrestore pbs-techflow:backup/vm/<vmid>/<timestamp> <new-vmid> \
  --storage <storage> [--force]

# Restauration conteneur LXC
pct restore <ctid> pbs-techflow:backup/ct/<ctid>/<timestamp> \
  --storage <storage> [--force]

# Lister les sauvegardes disponibles
proxmox-backup-client list \
  --repository backup@pbs@10.10.0.10:techflow-backup

# Monter une sauvegarde pour accès fichiers
proxmox-backup-client mount \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  vm/100/<timestamp> /mnt/pbs-restore

# Vérifier l'intégrité d'un datastore
proxmox-backup-manager verify-job run <job-id>

# Synchronisation vers PBS DR
proxmox-backup-manager sync-job run <sync-job-id>

# Appliquer la politique de rétention (prune)
proxmox-backup-client prune \
  --repository backup@pbs@10.10.0.10:techflow-backup \
  --backup-type vm --backup-id <vmid> \
  --keep-daily 30 --keep-weekly 8 --keep-monthly 12

# Lancer le Garbage Collection
proxmox-backup-manager garbage-collection start techflow-backup
```

Le prochain chapitre aborde la **supervision et la métrologie** : comment mettre en place une surveillance complète du cluster Proxmox VE avec Prometheus, Grafana et les alertes Alertmanager.

10 Sécurisation de l'infrastructure

```
La sécurité n'est pas un état, c'est un processus continu d'amélioration et de
vigilance.

-- Bruce Schneier
```

```
:depth: 2
:local:
```

Un cluster Proxmox VE expose une surface d'attaque significative : interface web sur le port 8006, SSH sur chaque nœud, API REST, trafic Corosync entre les nœuds, et potentiellement des centaines de VM et conteneurs. Sans durcissement, un attaquant ayant accès au réseau de management peut compromettre l'ensemble de l'infrastructure virtuelle en quelques minutes.

Ce chapitre couvre la sécurisation complète d'un cluster Proxmox VE 9, depuis le modèle de contrôle d'accès jusqu'au durcissement kernel, en passant par le firewall multicouche et la conformité aux référentiels CIS. Le lab TechFlow SAS sert de fil conducteur pour illustrer chaque concept avec des configurations réelles et éprouvées.

10.1 Modèle de sécurité Proxmox

Proxmox VE implémente un modèle de sécurité en couches : authentification via des realms configurables, autorisation par RBAC granulaire, et protection des accès API par des tokens. Comprendre cette architecture est indispensable avant tout déploiement en production.

10.1.1 Authentification : realms PAM, PVE, LDAP, OpenID

Proxmox VE supporte quatre types de realms (domaines d'authentification), chacun adapté à un contexte d'usage différent.

Realm PAM (`pam`)

Le realm PAM délègue l'authentification au système Linux sous-jacent via les modules PAM (`/etc/pam.d/proxmox`). Seuls les utilisateurs Linux locaux avec un compte Proxmox associé peuvent s'authentifier. Ce realm est réservé à l'utilisateur `root@pam` et aux comptes d'administration système.

> ✗ **DANGER**
> L'utilisateur `root@pam` dispose de tous les droits sans restriction de RBAC. Ne jamais partager ce compte. Créer des comptes nominatifs dans le realm PVE ou LDAP pour tous les administrateurs.

Realm PVE (`pve`)

Le realm PVE utilise une base de données interne stockée dans `/etc/pve/user.cfg` . Les mots de passe sont hachés avec SHA-256. Ce realm convient aux petites équipes sans infrastructure LDAP centralisée.

```
# Lister les realms configurés
pveum realm list

# Créer un utilisateur dans le realm PVE
pveum user add adminpve@pve --firstname "Admin" --lastname "Principal" \
  --email "admin@techflow.lan" --comment "Administrateur principal"

# Définir un mot de passe (minimum 8 caractères)
pveum passwd adminpve@pve

# Lister les utilisateurs
pveum user list
```

Realm LDAP / Active Directory

Pour les environnements d'entreprise, l'intégration LDAP (ou AD via LDAP) centralise la gestion des identités.

```
# Ajouter un realm LDAP
pveum realm add techflow-ldap \
  --type ldap \
  --server ldap.techflow.lan \
  --port 389 \
  --base_dn "dc=techflow,dc=lan" \
  --user_attr sAMAccountName \
  --bind_dn "cn=proxmox-bind,ou=services,dc=techflow,dc=lan" \
  --password "BindP@ssw0rd!" \
  --verify 1 \
  --tls 1

# Synchroniser les utilisateurs LDAP
pveum realm sync techflow-ldap --enable-new --remove-vanished

# Vérifier la synchronisation
pveum user list | grep "@techflow-ldap"
```

La configuration LDAP est stockée dans `/etc/pve/domains.cfg`. Le paramètre `--tls 1` active LDAPS (port 636) ou STARTTLS (port 389). Toujours activer le chiffrement TLS en production.

Realm OpenID Connect

Proxmox VE 7+ supporte l'authentification OpenID Connect (OIDC), permettant l'intégration avec Keycloak, Azure AD, Google Workspace ou tout fournisseur OIDC compatible.

```
# Ajouter un realm OpenID (exemple avec Keycloak)
pveum realm add techflow-oidc \
  --type openid \
  --issuer-url "https://keycloak.techflow.lan/realms/proxmox" \
  --client-id "proxmox-ve" \
  --client-key "secret-client-key" \
  --username-claim "preferred_username" \
  --autocreate 1

# Tester la configuration
pveum realm list --output-format json
```

> **► NOTE**
> Avec `--autocreate 1`, les utilisateurs OIDC sont créés automatiquement dans Proxmox lors de leur première connexion. Combiner avec des groupes OIDC mappés à des rôles Proxmox pour un provisionnement automatique des droits.

10.1.2 RBAC : rôles et permissions

Le contrôle d'accès basé sur les rôles (RBAC) de Proxmox VE repose sur trois concepts : les **objets** (nœuds, VM, pools, stockages), les **rôles** (ensembles de privilèges), et les **ACL** (association utilisateur/groupe + rôle + chemin).

Rôles intégrés

Rôle	Description	Cas d'usage
`Administrator`	Tous les privilèges	Administrateurs système
`PVEAdmin`	Administration sans gestion utilisateurs	Ops avancés
`PVEVMAdmin`	Gestion complète des VM	Équipes applicatives
`PVEVMUser`	Démarrer/arrêter/console VM	Utilisateurs finaux
`PVEDatastoreAdmin`	Gestion des stockages	Équipes stockage
`PVEDatastoreUser`	Allouer du stockage	VM operators
`PVEAuditor`	Lecture seule globale	Équipes audit/sécurité
`PVEPoolAdmin`	Gestion d'un pool	Responsables de pools

Rôle	Description	Cas d'usage
`NoAccess`	Refus explicite	Bloquer un accès hérité

Création de rôles personnalisés

```
# Créer un rôle pour l'équipe ops (gestion VM + monitoring)
pveum role add OpsTeamRole \
  --privs "VM.Allocate,VM.Clone,VM.Config.CDROM,VM.Config.CPU,VM.Config.Disk,\
VM.Config.Memory,VM.Config.Network,VM.Console,VM.Monitor,VM.PowerMgmt,\
VM.Snapshot,Datastore.AllocateSpace,Datastore.Audit,Sys.Audit"

# Créer un rôle lecture seule étendu (avec accès console)
pveum role add ReadOnlyConsole \
  --privs "VM.Audit,VM.Console,Sys.Audit,Datastore.Audit"

# Lister les privilèges disponibles
pveum role list --output-format json | python3 -m json.tool
```

Assignation des ACL

Les ACL sont assignées sur des chemins hiérarchiques. Le chemin `/` s'applique à tout le cluster, `/nodes/pve01` à un nœud spécifique, `/vms/100` à la VM 100, `/pool/production` à un pool.

```
# Assigner le rôle Administrator à adminpve sur tout le cluster
pveum acl modify / --users adminpve@pve --roles Administrator

# Assigner OpsTeamRole au groupe ops-team sur le pool production
pveum acl modify /pool/production --groups ops-team --roles OpsTeamRole

# Assigner lecture seule au groupe readonly sur tout le cluster
pveum acl modify / --groups readonly --roles PVEAuditor

# Assigner NoAccess pour bloquer l'accès à un nœud sensible
pveum acl modify /nodes/pve03 --users contractor@pve --roles NoAccess

# Vérifier les ACL
pveum acl list

# ACL d'un chemin spécifique
pveum acl list --path /pool/production
```

Gestion des groupes

```
# Créer les groupes TechFlow
pveum group add admins --comment "Administrateurs Proxmox"
pveum group add ops-team --comment "Équipe opérations"
pveum group add readonly --comment "Accès lecture seule"
pveum group add auditors --comment "Équipe audit sécurité"

# Ajouter des utilisateurs aux groupes
pveum user modify adminpve@pve --groups admins
pveum user modify ops01@pve --groups ops-team
pveum user modify readonly@pve --groups readonly

# Lister les membres d'un groupe
pveum group list
```

10.1.3 Two-factor authentication (TOTP, WebAuthn)

L'authentification à deux facteurs (2FA) est obligatoire pour tous les comptes administrateurs en production. Proxmox VE supporte TOTP (Time-based One-Time Password) et WebAuthn (clés matérielles FIDO2).

Configuration TOTP

```
# Activer TOTP pour un utilisateur via l'API
# Générer un secret TOTP (base32)
python3 -c "
import base64, os
secret = base64.b32encode(os.urandom(20)).decode()
```

```
print(f'Secret TOTP: {secret}')
print(f'URI: otpauth://totp/TechFlow:adminpve@pve?secret={secret}&issuer=TechFlow-
Proxmox')
"

# Configurer le TOTP via pveum
pveum user modify adminpve@pve --keys "totp:SECRET_BASE32_ICI"

# Vérifier la configuration 2FA
pveum user list --output-format json | python3 -c "
import json,sys
users = json.load(sys.stdin)
for u in users.get('data', []):
    print(f\"{u['userid']}: 2FA={'keys' in u and bool(u['keys'])}\")"
```

Via l'interface web : Datacenter > Permissions > Two Factor > Add TOTP. L'utilisateur scanne le QR code avec Google Authenticator, Aegis ou Authy.

Configuration WebAuthn

WebAuthn offre une sécurité supérieure au TOTP car il est résistant au phishing (lié au domaine). Configuration requise dans `/etc/pve/datacenter.cfg` :

```
# /etc/pve/datacenter.cfg - section webauthn
webauthn: rp=proxmox.techflow.lan,origin=https://proxmox.techflow.lan:
8006,id=proxmox.techflow.lan
```

```
# Appliquer la configuration WebAuthn
cat >> /etc/pve/datacenter.cfg << 'EOF'
webauthn: rp=proxmox.techflow.lan,origin=https://proxmox.techflow.lan:
8006,id=proxmox.techflow.lan
EOF

# Redémarrer pveproxy pour appliquer
systemctl restart pveproxy
```

L'enregistrement d'une clé WebAuthn se fait uniquement via l'interface web (Datacenter > Permissions > Two Factor > Add WebAuthn). Une YubiKey 5, une clé FIDO2 ou Windows Hello peuvent être utilisés.

Politique 2FA obligatoire

```
# Forcer le 2FA pour un realm (Proxmox VE 8+)
pveum realm modify pve --tfa type=totp

# Vérifier que tous les admins ont le 2FA activé
pveum user list --output-format json | python3 -c "
import json, sys
data = json.load(sys.stdin)
for u in data.get('data', []):
    has_2fa = 'keys' in u and bool(u.get('keys', ''))
    if not has_2fa:
        print(f'ALERTE: {u[\"userid\"]} sans 2FA!')
"
```

10.1.4 API tokens et least privilege

Les API tokens permettent d'automatiser les interactions avec Proxmox sans exposer les identifiants utilisateur. Ils suivent le principe du moindre privilège : un token ne peut pas avoir plus de droits que son utilisateur parent.

```
# Créer un token pour l'automatisation Terraform
pveum user token add adminpve@pve terraform-token \
  --privsep 1 \
  --comment "Token Terraform - accès limité"

# La sortie affiche le secret (non récupérable ultérieurement) :
#
# ┌──────────────┬──────────────────────────────────────┐
# │ key          │ value                                │
# ╞══════════════╪══════════════════════════════════════╡
```

```
# │ full-tokenid │ adminpve@pve!terraform-token                    │
# │ info         │ {"privsep":"1"}                                 │
# │ value        │ xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx            │
# └──────────────┴─────────────────────────────────────────────────┘

# Assigner des droits minimaux au token (pas à l'utilisateur parent)
pveum acl modify /vms \
  --tokens "adminpve@pve!terraform-token" \
  --roles PVEVMAdmin

pveum acl modify /storage \
  --tokens "adminpve@pve!terraform-token" \
  --roles PVEDatastoreUser

# Créer un token pour le monitoring (lecture seule)
pveum user token add readonly@pve monitoring-token \
  --privsep 1 \
  --comment "Token monitoring Prometheus"

pveum acl modify / \
  --tokens "readonly@pve!monitoring-token" \
  --roles PVEAuditor

# Lister les tokens d'un utilisateur
pveum user token list adminpve@pve

# Supprimer un token compromis
pveum user token remove adminpve@pve old-token
```

Utilisation des tokens avec l'API REST

```
# Exemple d'appel API avec token
TOKEN_ID="adminpve@pve!terraform-token"
TOKEN_SECRET="xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

curl -s \
  -H "Authorization: PVEAPIToken=${TOKEN_ID}=${TOKEN_SECRET}" \
  "https://proxmox.techflow.lan:8006/api2/json/nodes" | \
  python3 -m json.tool

# Exemple avec Python (requests)
python3 << 'EOF'
import requests, urllib3
urllib3.disable_warnings()

proxmox_url = "https://proxmox.techflow.lan:8006/api2/json"
token_id = "adminpve@pve!terraform-token"
token_secret = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

headers = {"Authorization": f"PVEAPIToken={token_id}={token_secret}"}
r = requests.get(f"{proxmox_url}/nodes", headers=headers, verify=False)
print(r.json())
EOF
```

10.2 Durcissement des nœuds

Le durcissement (hardening) des nœuds Proxmox réduit la surface d'attaque au niveau système. Ces opérations s'appliquent sur chaque nœud du cluster.

10.2.1 Durcissement SSH (clés, no-password, port)

SSH est le vecteur d'attaque le plus courant sur les serveurs Linux. Un SSH mal configuré expose le nœud à des attaques par force brute et aux connexions non autorisées.

```
# Générer une paire de clés Ed25519 (côté administrateur)
ssh-keygen -t ed25519 -C "adminpve@techflow-$(date +%Y%m%d)" -f ~/.ssh/
proxmox_ed25519

# Copier la clé publique sur tous les nœuds
```

```
for node in pve01 pve02 pve03; do
  ssh-copy-id -i ~/.ssh/proxmox_ed25519.pub root@${node}.techflow.lan
done

# Configuration SSH durcie
cat > /etc/ssh/sshd_config.d/99-hardening.conf << 'EOF'
# Proxmox VE - SSH Hardening - TechFlow SAS
Protocol 2
Port 2222
AddressFamily inet

# Authentification
PermitRootLogin prohibit-password
PasswordAuthentication no
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
PermitEmptyPasswords no
ChallengeResponseAuthentication no
UsePAM yes

# Algorithmes forts uniquement
HostKeyAlgorithms ssh-ed25519,rsa-sha2-512,rsa-sha2-256
KexAlgorithms curve25519-sha256,diffie-hellman-group16-sha512,diffie-hellman-
group18-sha512
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-
gcm@openssh.com
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com

# Restrictions de session
LoginGraceTime 30
MaxAuthTries 3
MaxSessions 4
ClientAliveInterval 300
ClientAliveCountMax 2
TCPKeepAlive no

# Fonctionnalités désactivées
X11Forwarding no
AllowAgentForwarding no
AllowTcpForwarding no
PermitTunnel no
GatewayPorts no

# Logging
LogLevel VERBOSE
SyslogFacility AUTH

# Bannière légale
Banner /etc/ssh/banner.txt

# Restriction par groupe
AllowGroups sudo ssh-users
EOF

# Créer la bannière légale
cat > /etc/ssh/banner.txt << 'EOF'
*****************************************************************************
*              TECHFLOW SAS - Système d'information privé                   *
*  Accès réservé aux personnes autorisées. Toute connexion est journalisée.*
*     Toute tentative non autorisée fera l'objet de poursuites pénales.    *
*****************************************************************************
EOF

# Vérifier la configuration avant redémarrage
sshd -t && echo "Configuration SSH valide"

# Redémarrer SSH
systemctl restart sshd
```

```
# Mettre à jour le firewall pour le nouveau port
ufw allow 2222/tcp comment "SSH Proxmox"
```

> **⚠ AVERTISSEMENT**
> Avant de changer le port SSH et désactiver l'authentification par mot de passe, vérifier que la connexion par clé fonctionne dans une autre session. Avoir accès à la console IPMI/iDRAC comme filet de sécurité est indispensable.

Protection contre le brute-force avec fail2ban

```
# Installer fail2ban
apt install -y fail2ban

# Configuration pour Proxmox VE
cat > /etc/fail2ban/jail.d/proxmox.conf << 'EOF'
[proxmox-web]
enabled  = true
port     = https,8006
filter   = proxmox
logpath  = /var/log/daemon.log
maxretry = 5
findtime = 600
bantime  = 3600

[sshd]
enabled  = true
port     = 2222
logpath  = /var/log/auth.log
maxretry = 3
findtime = 300
bantime  = 7200
EOF

# Filtre pour l'interface Proxmox
cat > /etc/fail2ban/filter.d/proxmox.conf << 'EOF'
[Definition]
failregex = pvedaemon\[.*authentication failure; rhost=<HOST> user=.* msg=.*
ignoreregex =
EOF

systemctl enable --now fail2ban
fail2ban-client status
```

10.2.2 Désactivation des services inutiles

Chaque service actif représente une surface d'attaque potentielle. Sur un nœud Proxmox de production, seuls les services strictement nécessaires doivent tourner.

```
# Lister tous les services actifs
systemctl list-units --type=service --state=active

# Services Proxmox essentiels à conserver
ESSENTIAL_SERVICES=(
  "pve-cluster"      # Cluster filesystem (pmxcfs)
  "pvedaemon"        # Daemon Proxmox VE
  "pveproxy"         # Proxy web et API
  "pvestatd"         # Statistiques
  "corosync"         # Synchronisation cluster
  "ceph.target"      # Si Ceph activé
  "qemu-server"      # Gestion QEMU/KVM
  "lxc"              # Gestion LXC
  "sshd"             # Accès SSH
  "cron"             # Tâches planifiées
  "rsyslog"          # Journalisation
  "chrony"           # NTP (si installé)
)
```

```
# Services à désactiver sur les nœuds Proxmox
DISABLE_SERVICES=(
  "bluetooth"
  "avahi-daemon"
  "cups"
  "cups-browsed"
  "isc-dhcp-server"
  "bind9"
  "apache2"
  "nginx"
  "postfix"         # Sauf si relay mail configuré
  "nfs-server"      # Sauf si NFS utilisé
  "rpcbind"         # Sauf si NFS utilisé
  "telnet"
  "rsh"
)

for svc in "${DISABLE_SERVICES[@]}"; do
  if systemctl is-active "$svc" &>/dev/null; then
    systemctl stop "$svc"
    systemctl disable "$svc"
    systemctl mask "$svc"
    echo "Service désactivé: $svc"
  fi
done

# Désactiver IPv6 si non utilisé
echo "net.ipv6.conf.all.disable_ipv6 = 1" >> /etc/sysctl.d/99-proxmox-
hardening.conf
echo "net.ipv6.conf.default.disable_ipv6 = 1" >> /etc/sysctl.d/99-proxmox-
hardening.conf
sysctl --system

# Vérifier les ports ouverts
ss -tlnp
```

Désactivation de rpcbind si pas de NFS

```
# Vérifier si NFS est utilisé
showmount -e localhost 2>/dev/null || echo "Pas de partages NFS"

# Si NFS non utilisé, désactiver rpcbind
systemctl stop rpcbind rpcbind.socket
systemctl disable rpcbind rpcbind.socket
systemctl mask rpcbind rpcbind.socket
```

10.2.3 Kernel hardening (sysctl)

Les paramètres sysctl permettent de durcir le kernel Linux contre les attaques réseau et les exploits d'élévation de privilèges.

```
# Configuration sysctl de durcissement pour Proxmox VE
cat > /etc/sysctl.d/99-proxmox-hardening.conf << 'EOF'
# ═══════════════════════════════════════════════════════════
# TechFlow SAS - Kernel Hardening Proxmox VE 9
# ═══════════════════════════════════════════════════════════

# --- Protection réseau ---
# Activer le filtrage du chemin inverse (protection IP spoofing)
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# Désactiver le routage IP (le nœud n'est pas un routeur)
net.ipv4.ip_forward = 1        # Requis pour les VM/LXC
net.ipv4.conf.all.forwarding = 1

# Désactiver les redirections ICMP
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
```

```
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
net.ipv6.conf.all.accept_redirects = 0

# Désactiver le source routing
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0

# Protection SYN flood
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_syn_retries = 3
net.ipv4.tcp_synack_retries = 3

# Ignorer les broadcasts ICMP (prévention Smurf)
net.ipv4.icmp_echo_ignore_broadcasts = 1

# Ignorer les erreurs ICMP bogues
net.ipv4.icmp_ignore_bogus_error_responses = 1

# Logging des paquets avec adresses source martienne
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1

# --- Protection mémoire ---
# ASLR (Address Space Layout Randomization) - niveau maximum
kernel.randomize_va_space = 2

# Désactiver les core dumps pour les programmes setuid
fs.suid_dumpable = 0

# Protection des liens symboliques et hardlinks
fs.protected_symlinks = 1
fs.protected_hardlinks = 1

# Protection FIFOs et fichiers réguliers dans répertoires world-writable
fs.protected_fifos = 2
fs.protected_regular = 2

# --- Protection kernel ---
# Désactiver la magie SysRq (utile uniquement en debug)
kernel.sysrq = 0

# Restreindre l'accès aux pointeurs kernel dans /proc
kernel.kptr_restrict = 2

# Restreindre l'accès à dmesg aux root uniquement
kernel.dmesg_restrict = 1

# Désactiver le BPF JIT (attaques Spectre)
net.core.bpf_jit_enable = 0

# Restreindre ptrace (protection contre les attaques de débogage)
kernel.yama.ptrace_scope = 1

# Taille maximale de la file de connexions
net.core.somaxconn = 65535

# --- Performance et stabilité ---
# Optimisation mémoire pour Proxmox
vm.swappiness = 10
vm.dirty_ratio = 60
vm.dirty_background_ratio = 2

# Limite de fichiers inotify (pour pve-cluster)
fs.inotify.max_user_watches = 1048576
fs.inotify.max_user_instances = 512
EOF
```

```
# Appliquer immédiatement
sysctl --system

# Vérifier les paramètres appliqués
sysctl net.ipv4.tcp_syncookies kernel.randomize_va_space kernel.kptr_restrict
```

10.2.4 AppArmor et seccomp

AppArmor et seccomp fournissent une isolation supplémentaire pour les processus critiques de Proxmox, limitant les dégâts en cas d'exploitation d'une vulnérabilité.

AppArmor pour Proxmox

```
# Vérifier l'état d'AppArmor
aa-status

# Installer les outils AppArmor
apt install -y apparmor-utils apparmor-profiles

# Les profils LXC sont gérés automatiquement par Proxmox
# Vérifier le profil LXC par défaut
cat /etc/apparmor.d/lxc/lxc-default-cgns

# Créer un profil AppArmor pour pvedaemon
cat > /etc/apparmor.d/usr.sbin.pvedaemon << 'EOF'
#include <tunables/global>

/usr/sbin/pvedaemon {
  #include <abstractions/base>
  #include <abstractions/nameservice>

  capability net_admin,
  capability sys_admin,
  capability dac_override,

  /usr/sbin/pvedaemon mr,
  /etc/pve/** rw,
  /var/log/pve/** rw,
  /var/lib/pve-cluster/** rw,
  /run/pve/** rw,

  # Refuser l'accès aux fichiers sensibles
  deny /etc/shadow r,
  deny /root/.ssh/** rwx,
}
EOF

# Charger le profil en mode complain (test) puis enforce
apparmor_parser -r /etc/apparmor.d/usr.sbin.pvedaemon
aa-complain /usr/sbin/pvedaemon

# Après validation (quelques jours de logs), passer en enforce
# aa-enforce /usr/sbin/pvedaemon

# Vérifier les profils actifs
aa-status | grep -E "(enforce|complain)"
```

seccomp pour les conteneurs LXC

Proxmox VE applique automatiquement des profils seccomp aux conteneurs LXC via `/usr/share/lxc/config/common.seccomp`. Ce profil bloque les appels système dangereux.

```
# Vérifier le profil seccomp LXC actif
cat /usr/share/lxc/config/common.seccomp | head -30

# Créer un profil seccomp renforcé pour les conteneurs applicatifs
cat > /etc/pve/lxc/seccomp-strict.conf << 'EOF'
2
blacklist
reject_force_umount
```

```
[all]
kexec_load errno 1
open_by_handle_at errno 1
init_module errno 1
finit_module errno 1
delete_module errno 1
perf_event_open errno 1
bpf errno 1
userfaultfd errno 1
EOF

# Appliquer à un conteneur (CT 200) via configuration
echo "lxc.seccomp.profile = /etc/pve/lxc/seccomp-strict.conf" \
  >> /etc/pve/lxc/200.conf
```

10.2.5 Auditd et logging centralisé

Le daemon d'audit Linux (`auditd`) journalise les événements de sécurité critiques : connexions, escalades de privilèges, modifications de fichiers système.

```
# Installer auditd
apt install -y auditd audispd-plugins

# Configuration des règles d'audit pour Proxmox
cat > /etc/audit/rules.d/99-proxmox-audit.rules << 'EOF'
# ============================================================
# TechFlow SAS - Règles d'audit Proxmox VE 9
# ============================================================

# Effacer toutes les règles existantes
-D

# Buffer size
-b 8192

# Niveau d'erreur (2 = panic si buffer plein)
-f 1

# --- Surveillance des fichiers de configuration Proxmox ---
-w /etc/pve/ -p wa -k proxmox-config
-w /etc/pve/user.cfg -p wa -k proxmox-users
-w /etc/pve/domains.cfg -p wa -k proxmox-auth
-w /etc/pve/datacenter.cfg -p wa -k proxmox-datacenter

# --- Surveillance SSH ---
-w /etc/ssh/sshd_config -p wa -k ssh-config
-w /etc/ssh/sshd_config.d/ -p wa -k ssh-config
-w /root/.ssh/ -p wa -k ssh-keys

# --- Fichiers système critiques ---
-w /etc/passwd -p wa -k identity
-w /etc/shadow -p wa -k identity
-w /etc/group -p wa -k identity
-w /etc/sudoers -p wa -k sudoers
-w /etc/sudoers.d/ -p wa -k sudoers

# --- Appels système privilégiés ---
-a always,exit -F arch=b64 -S execve -F euid=0 -k root-commands
-a always,exit -F arch=b64 -S kill -F a1=9 -k process-kill
-a always,exit -F arch=b64 -S ptrace -k ptrace

# --- Élévation de privilèges ---
-w /usr/bin/sudo -p x -k sudo-exec
-w /usr/bin/su -p x -k su-exec
-a always,exit -F arch=b64 -S setuid -k setuid
-a always,exit -F arch=b64 -S setgid -k setgid

# --- Modules kernel ---
-w /sbin/insmod -p x -k kernel-module
```

```
-w /sbin/rmmod -p x -k kernel-module
-w /sbin/modprobe -p x -k kernel-module

# --- Réseau ---
-a always,exit -F arch=b64 -S socket -F a0=2 -k network-socket
-w /etc/hosts -p wa -k network-hosts
-w /etc/network/ -p wa -k network-config

# --- Immuabilité des règles ---
-e 2
EOF

# Appliquer les règles
augenrules --load

# Vérifier les règles actives
auditctl -l

# Redémarrer auditd
systemctl restart auditd
systemctl enable auditd
```

Centralisation des logs avec rsyslog

```
# Configurer rsyslog pour envoyer les logs vers un SIEM central
cat > /etc/rsyslog.d/99-techflow-remote.conf << 'EOF'
# Envoi des logs critiques vers le SIEM TechFlow
# Format RFC 5424 avec TLS

$DefaultNetstreamDriverCAFile /etc/ssl/certs/techflow-ca.pem
$DefaultNetstreamDriver gtls
$ActionSendStreamDriverMode 1
$ActionSendStreamDriverAuthMode x509/name
$ActionSendStreamDriverPermittedPeer siem.techflow.lan

# Tous les logs d'audit
if $programname == 'audit' then @@siem.techflow.lan:6514
# Logs d'authentification
auth,authpriv.* @@siem.techflow.lan:6514
# Logs Proxmox
if $programname startswith 'pve' then @@siem.techflow.lan:6514
EOF

# Tester la configuration rsyslog
rsyslogd -N1

# Redémarrer rsyslog
systemctl restart rsyslog

# Vérifier les logs auditd locaux
ausearch -k proxmox-config --start today | aureport -f
```

10.3 Firewall Proxmox

Proxmox VE intègre un firewall nativement géré via `pve-firewall`, basé sur `iptables` / `nftables`. Son architecture en couches permet une gestion granulaire des règles.

10.3.1 Architecture (datacenter → nœud → VM)

Le firewall Proxmox s'organise en trois niveaux hiérarchiques :

Niveau Datacenter : Règles globales applicables à tous les nœuds. Fichier : `/etc/pve/firewall/cluster.fw`

Niveau Nœud : Règles spécifiques à l'hôte Proxmox. Fichier : `/etc/pve/nodes/<nodename>/host.fw`

Niveau VM/CT : Règles par machine virtuelle ou conteneur. Fichier : `/etc/pve/firewall/<vmid>.fw`

```
# Activer le firewall au niveau datacenter
cat > /etc/pve/firewall/cluster.fw << 'EOF'
[OPTIONS]
enable: 1
policy_in: DROP
policy_out: ACCEPT
log_ratelimit: enable=1,rate=1/second,burst=5
EOF

# Vérifier l'état du firewall
pve-firewall status

# Lister les chaînes iptables générées
iptables -L -n -v | head -50

# Logs du firewall
journalctl -u pve-firewall -f
```

10.3.2 Règles datacenter (cluster.fw)

Les règles datacenter s'appliquent à l'ensemble des nœuds du cluster. Elles définissent les politiques d'accès aux services de management.

> **⚠ AVERTISSEMENT**
> Activer le firewall datacenter avec `policy_in: DROP` sans d'abord créer les règles d'accès SSH et à l'interface web (port 8006) bloquera l'accès au cluster.

```
# /etc/pve/firewall/cluster.fw - TechFlow SAS Production
[OPTIONS]
enable: 1
policy_in: DROP
policy_out: ACCEPT
log_ratelimit: enable=1,rate=1/second,burst=5

[ALIASES]
# Plages réseau TechFlow
mgmt_net = 10.10.0.0/24
ops_net = 10.10.1.0/24
monitoring_net = 10.10.2.0/24
bastion_host = 10.10.0.10
siem_server = 10.20.0.5

[IPSET mgmt-access]
10.10.0.0/24 # Réseau management
10.10.0.10   # Bastion SSH

[IPSET monitoring]
10.10.2.0/24 comment monitoring network

[RULES]
# ═══ Règles ENTRANT datacenter ═══

# SSH depuis le bastion uniquement
IN ACCEPT -source +mgmt-access -p tcp -dport 2222 -log nolog

# Interface web Proxmox depuis le réseau management
IN ACCEPT -source +mgmt-access -p tcp -dport 8006 -log nolog

# Corosync (cluster) - entre nœuds uniquement
IN ACCEPT -source 10.10.10.0/24 -p udp -dport 5405:5406 -log nolog

# Ceph (entre nœuds)
IN ACCEPT -source 10.10.20.0/24 -p tcp -dport 3300 -log nolog
IN ACCEPT -source 10.10.20.0/24 -p tcp -dport 6789 -log nolog
IN ACCEPT -source 10.10.20.0/24 -p tcp -dport 6800:7300 -log nolog

# SPICE/VNC pour console VM (depuis management uniquement)
IN ACCEPT -source +mgmt-access -p tcp -dport 3128 -log nolog
```

```
IN ACCEPT -source +mgmt-access -p tcp -dport 5900:5999 -log nolog

# Migration VM entre nœuds
IN ACCEPT -source 10.10.10.0/24 -p tcp -dport 60000:60050 -log nolog

# Monitoring (Prometheus node exporter)
IN ACCEPT -source +monitoring -p tcp -dport 9100 -log nolog

# NTP sortant
OUT ACCEPT -p udp -dport 123 -log nolog

# DNS
OUT ACCEPT -p udp -dport 53 -log nolog
OUT ACCEPT -p tcp -dport 53 -log nolog

# HTTPS pour mises à jour
OUT ACCEPT -p tcp -dport 443 -log nolog

# Journalisation des refus
IN DROP -log warning
```

```
# Appliquer et recharger les règles firewall
pve-firewall compile && pve-firewall reload

# Vérifier les règles iptables générées
iptables -L PROXMOX-INPUT -n -v
```

10.3.3 Règles par VM/CT

Chaque VM ou conteneur peut avoir ses propres règles firewall. Ces règles s'appliquent au niveau de l'interface réseau virtuelle (veth/tap).

```
# Activer le firewall pour la VM 100
# Via l'API ou en éditant directement /etc/pve/firewall/100.fw

cat > /etc/pve/firewall/100.fw << 'EOF'
[OPTIONS]
enable: 1
dhcp: 0
ipfilter: 1
log_level_in: warning
log_level_out: nolog
policy_in: DROP
policy_out: ACCEPT
macfilter: 1

[RULES]
# Serveur web (VM 100 - nginx)
IN ACCEPT -p tcp -dport 80 -log nolog
IN ACCEPT -p tcp -dport 443 -log nolog

# SSH depuis management uniquement
IN ACCEPT -source 10.10.0.0/24 -p tcp -dport 22 -log nolog

# Monitoring
IN ACCEPT -source 10.10.2.0/24 -p tcp -dport 9100 -log nolog

# Ping depuis monitoring
IN ACCEPT -source 10.10.2.0/24 -p icmp -log nolog
EOF

# Firewall pour un conteneur LXC (CT 200 - base de données)
cat > /etc/pve/firewall/200.fw << 'EOF'
[OPTIONS]
enable: 1
ipfilter: 1
policy_in: DROP
policy_out: ACCEPT
```

```
[RULES]
# PostgreSQL accessible uniquement depuis le subnet applicatif
IN ACCEPT -source 10.30.1.0/24 -p tcp -dport 5432 -log nolog
# SSH management
IN ACCEPT -source 10.10.0.0/24 -p tcp -dport 22 -log nolog
EOF

# Recharger le firewall
pve-firewall reload

# Vérifier l'état par VM
pve-firewall vm-list
```

10.3.4 IPSets et alias

Les IPSets permettent de regrouper des adresses IP et réseaux dans des objets nommés, facilitant la gestion des règles firewall.

```
# Créer des IPSets au niveau datacenter
# Ces IPSets sont disponibles dans toutes les règles (préfixe +)

# Via l'API REST
curl -s -X POST \
  -H "Authorization: PVEAPIToken=adminpve@pve!fw-token=xxx" \
  "https://localhost:8006/api2/json/cluster/firewall/ipset" \
  -d "name=db-servers&comment=Serveurs base de données"

curl -s -X POST \
  -H "Authorization: PVEAPIToken=adminpve@pve!fw-token=xxx" \
  "https://localhost:8006/api2/json/cluster/firewall/ipset/db-servers" \
  -d "cidr=10.30.2.0/24&comment=DB subnet production"

# Ou directement dans cluster.fw
cat >> /etc/pve/firewall/cluster.fw << 'EOF'

[IPSET web-servers]
10.30.1.10 comment nginx-01
10.30.1.11 comment nginx-02
10.30.1.12 comment nginx-03

[IPSET db-servers]
10.30.2.10 comment postgres-primary
10.30.2.11 comment postgres-replica

[IPSET backup-servers]
10.40.0.5 comment backup-pbs-01
EOF

# Utiliser les IPSets dans les règles
cat >> /etc/pve/firewall/cluster.fw << 'EOF'
# Autoriser les serveurs web à accéder aux bases de données
IN ACCEPT -source +web-servers -dest +db-servers -p tcp -dport 5432
EOF

# Lister les IPSets définis
pve-firewall ipset-list
```

Alias réseau

```
# Les alias sont définis dans la section [ALIASES] de cluster.fw
# et référencés avec le préfixe $ dans les règles

cat >> /etc/pve/firewall/cluster.fw << 'EOF'

[ALIASES]
gateway_prod = 10.30.0.1
ntp_server = 10.10.0.53
ldap_server = 10.10.0.100
```

```
syslog_server = 10.20.0.5
EOF
```

10.3.5 Firewall SDN

Avec Proxmox VE 8+ et le module SDN (Software-Defined Networking), le firewall peut être intégré directement dans les VNet SDN pour une isolation réseau plus fine.

```
# Activer le module SDN
apt install -y libpve-network-perl

# Créer une zone SDN de type VLAN
pvesh create /cluster/sdn/zones \
  --zone techflow-prod \
  --type vlan \
  --bridge vmbr0 \
  --mtu 1500

# Créer un VNet dans la zone
pvesh create /cluster/sdn/vnets \
  --vnet vnet-web \
  --zone techflow-prod \
  --tag 100 \
  --alias "Réseau Web Production"

# Créer un subnet avec firewall intégré
pvesh create /cluster/sdn/vnets/vnet-web/subnets \
  --subnet 10.30.1.0/24 \
  --type subnet \
  --gateway 10.30.1.1 \
  --snat 1

# Appliquer la configuration SDN
pvesh set /cluster/sdn

# Firewall SDN - règles sur le VNet
cat > /etc/pve/sdn/zones/techflow-prod/vnet-web.fw << 'EOF'
[OPTIONS]
enable: 1

[RULES]
IN ACCEPT -p tcp -dport 80
IN ACCEPT -p tcp -dport 443
IN DROP
EOF
```

10.4 Sécurité réseau

La segmentation réseau est la première ligne de défense contre la propagation des attaques au sein de l'infrastructure.

10.4.1 Isolation des réseaux (management, VM, Ceph, cluster)

Un cluster Proxmox de production utilise au minimum quatre réseaux isolés physiquement ou par VLAN :

Réseau	Sous-réseau	Usage	Interface
Management	10.10.0.0/24	Interface web, SSH, API	eno1 / vmbr0
VM Production	10.30.0.0/16	Trafic applicatif VM	eno2 / vmbr1
Corosync/Cluster	10.10.10.0/24	Synchronisation cluster	eno3 / bond0
Ceph/Stockage	10.10.20.0/24	Réplication Ceph	eno4 / bond1
Backup	10.40.0.0/24	Sauvegardes PBS	eno5

```
# Configuration réseau d'un nœud Proxmox avec séparation des réseaux
cat > /etc/network/interfaces << 'EOF'
# Loopback
```

```
auto lo
iface lo inet loopback

# Interface management (VLAN 10)
auto eno1
iface eno1 inet manual
  mtu 1500

auto eno1.10
iface eno1.10 inet static
  address 10.10.0.11/24
  gateway 10.10.0.1
  dns-nameservers 10.10.0.53
  dns-search techflow.lan
  mtu 1500

# Bridge management (pour l'interface Proxmox)
auto vmbr0
iface vmbr0 inet static
  address 10.10.0.11/24
  bridge-ports eno1.10
  bridge-stp off
  bridge-fd 0
  mtu 1500

# Bridge VM (trunk VLAN)
auto eno2
iface eno2 inet manual
  mtu 9000

auto vmbr1
iface vmbr1 inet manual
  bridge-ports eno2
  bridge-stp off
  bridge-fd 0
  bridge-vlan-aware yes
  bridge-vids 2-4094
  mtu 9000

# Interface Corosync (pas de bridge nécessaire)
auto eno3
iface eno3 inet static
  address 10.10.10.11/24
  mtu 9000

# Interface Ceph (pas de bridge nécessaire)
auto eno4
iface eno4 inet static
  address 10.10.20.11/24
  mtu 9000
EOF

# Appliquer la configuration
ifreload -a
```

10.4.2 VLAN et segmentation

La segmentation par VLAN isole les flux réseau au niveau L2, empêchant les communications non autorisées entre zones.

```
# Configurer les VLAN Proxmox pour TechFlow SAS
# Table de segmentation VLAN TechFlow
declare -A vlans=(
  [10]="Management"
  [20]="DMZ-Web"
  [30]="Production-App"
  [40]="Production-DB"
  [50]="Monitoring"
  [60]="Backup"
```

```
  [99]="Quarantine"
)

# Créer les bridges VLAN pour chaque zone
for vlan_id in "${!vlans[@]}"; do
  echo "VLAN ${vlan_id}: ${vlans[$vlan_id]}"
done

# Configuration d'une VM dans le VLAN 30 (Production-App)
pvesh set /nodes/pve01/qemu/100/config \
  --net0 "virtio,bridge=vmbr1,tag=30,firewall=1"

# Configurer un conteneur LXC dans le VLAN 40 (Production-DB)
pvesh set /nodes/pve01/lxc/200/config \
  --net0 "name=eth0,bridge=vmbr1,tag=40,ip=10.30.2.10/24,gw=10.30.2.1,firewall=1"

# Vérifier la configuration réseau d'une VM
pvesh get /nodes/pve01/qemu/100/config | grep net
```

10.4.3 Chiffrement du trafic cluster (Corosync)

Le trafic Corosync entre les nœuds du cluster transporte des informations sensibles sur l'état du cluster. Son chiffrement est essentiel dans un environnement multi-site.

```
# Vérifier la version et configuration Corosync actuelle
corosync-cfgtool -s

# Configuration Corosync avec chiffrement
cat > /etc/corosync/corosync.conf << 'EOF'
totem {
  version: 2
  cluster_name: techflow-cluster
  config_version: 1

  # Interface réseau cluster
  interface {
    ringnumber: 0
    bindnetaddr: 10.10.10.0
    mcastport: 5405
  }

  # Chiffrement du trafic cluster
  crypto_cipher: aes256
  crypto_hash: sha256

  # Transport (knet pour Corosync 3+)
  transport: knet

  # Compression du trafic cluster
  knet_compression_model: zlib
}

quorum {
  provider: corosync_votequorum
  expected_votes: 3
  two_node: 0
  wait_for_all: 1
  last_man_standing: 0
}

nodelist {
  node {
    ring0_addr: 10.10.10.11
    name: pve01
    nodeid: 1
  }
  node {
    ring0_addr: 10.10.10.12
    name: pve02
```

```
    nodeid: 2
  }
  node {
    ring0_addr: 10.10.10.13
    name: pve03
    nodeid: 3
  }
}

logging {
  to_logfile: yes
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes
  timestamp: on
  logger_subsys {
    subsys: QUORUM
    debug: off
  }
}
EOF

# Régénérer la clé de chiffrement Corosync
corosync-keygen

# Distribuer la clé sur tous les nœuds
for node in pve02 pve03; do
  scp /etc/corosync/authkey root@${node}.techflow.lan:/etc/corosync/authkey
  ssh root@${node}.techflow.lan "chmod 400 /etc/corosync/authkey"
done

# Vérifier le chiffrement actif
corosync-cfgtool -s | grep -i crypt
```

10.4.4 VPN pour accès distant

L'accès distant à l'infrastructure Proxmox doit impérativement passer par un VPN. L'interface web et l'API ne doivent jamais être exposées directement sur Internet.

```
# Installation WireGuard pour accès distant administrateurs
apt install -y wireguard

# Générer les clés WireGuard pour le serveur
wg genkey | tee /etc/wireguard/server_private.key | wg pubkey > /etc/wireguard/
server_public.key
chmod 600 /etc/wireguard/server_private.key

SERVER_PRIVATE=$(cat /etc/wireguard/server_private.key)
SERVER_PUBLIC=$(cat /etc/wireguard/server_public.key)

# Configuration du serveur WireGuard (sur pve01 ou bastion dédié)
cat > /etc/wireguard/wg0.conf << EOF
[Interface]
Address = 10.200.0.1/24
ListenPort = 51820
PrivateKey = ${SERVER_PRIVATE}

# Règles de routage vers le réseau management
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT;
iptables -t nat -A POSTROUTING -o vmbr0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j
ACCEPT; iptables -t nat -D POSTROUTING -o vmbr0 -j MASQUERADE

# Client 1 : adminpve
[Peer]
PublicKey = ADMIN_CLIENT_PUBLIC_KEY
AllowedIPs = 10.200.0.2/32
PersistentKeepalive = 25

# Client 2 : ops-team
```

```
[Peer]
PublicKey = OPS_CLIENT_PUBLIC_KEY
AllowedIPs = 10.200.0.3/32
PersistentKeepalive = 25
EOF

chmod 600 /etc/wireguard/wg0.conf

# Activer WireGuard
systemctl enable --now wg-quick@wg0

# Vérifier l'état du tunnel
wg show
```

10.5 Sécurité des VM et conteneurs

10.5.1 Images de confiance et vérification

Utiliser des images non vérifiées est l'un des vecteurs d'attaque les plus courants dans les environnements virtualisés.

```
# Télécharger une image officielle Debian avec vérification GPG
cd /var/lib/vz/template/cache

# Télécharger l'image et sa signature
wget https://cloud.debian.org/images/cloud/bookworm/latest/debian-12-generic-
amd64.qcow2
wget https://cloud.debian.org/images/cloud/bookworm/latest/SHA512SUMS
wget https://cloud.debian.org/images/cloud/bookworm/latest/SHA512SUMS.sign

# Importer la clé GPG Debian Cloud
gpg --keyserver keyring.debian.org --recv-keys 0x8EE3DAAE

# Vérifier la signature
gpg --verify SHA512SUMS.sign SHA512SUMS

# Vérifier l'intégrité de l'image
sha512sum -c SHA512SUMS --ignore-missing

# Créer une VM à partir de l'image vérifiée
qm create 9000 --name "template-debian12" \
  --memory 2048 --cores 2 \
  --net0 virtio,bridge=vmbr1,tag=30 \
  --scsihw virtio-scsi-pci \
  --ide2 local:cloudinit \
  --boot c --bootdisk scsi0 \
  --serial0 socket --vga serial0 \
  --agent enabled=1

# Importer le disque
qm importdisk 9000 debian-12-generic-amd64.qcow2 local-lvm

# Configurer le disque importé
qm set 9000 --scsi0 local-lvm:vm-9000-disk-0,discard=on

# Convertir en template
qm template 9000
```

Vérification des templates LXC

```
# Lister les templates disponibles dans le catalogue Proxmox
pveam available --section system | grep debian

# Télécharger un template officiel
pveam download local debian-12-standard_12.7-1_amd64.tar.zst

# Vérifier l'intégrité du template téléchargé
pveam list local
```

```
# Calculer le hash SHA256 du template
sha256sum /var/lib/vz/template/cache/debian-12-standard_12.7-1_amd64.tar.zst

# Comparer avec le hash officiel Proxmox
curl -s http://download.proxmox.com/images/system/SHA256SUMS | \
  grep "debian-12-standard_12.7-1_amd64.tar.zst"
```

10.5.2 Isolation des conteneurs LXC

Les conteneurs LXC partagent le kernel de l'hôte. Une isolation rigoureuse est donc critique.

```
# Créer un conteneur avec isolation maximale
pct create 200 local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst \
  --hostname db-postgres-01 \
  --memory 4096 \
  --swap 1024 \
  --cores 4 \
  --rootfs local-lvm:20 \
  --net0 name=eth0,bridge=vmbr1,tag=40,ip=10.30.2.10/24,gw=10.30.2.1,firewall=1 \
  --unprivileged 1 \
  --features nesting=0,keyctl=0 \
  --protection 0 \
  --onboot 1

# Configuration de sécurité avancée du conteneur
cat >> /etc/pve/lxc/200.conf << 'EOF'
# Désactiver les capacités inutiles
lxc.cap.drop = sys_admin sys_boot sys_module net_admin
lxc.cap.drop = sys_rawio sys_ptrace sys_pacct
lxc.cap.drop = setuid setgid chown

# Appliquer le profil AppArmor LXC renforcé
lxc.apparmor.profile = lxc-container-default-cgroupns

# Espace de noms utilisateur (UID/GID mapping)
lxc.idmap = u 0 100000 65536
lxc.idmap = g 0 100000 65536

# Limiter les ressources kernel
lxc.prlimit.nofile = 65536
lxc.prlimit.nproc = 4096

# Désactiver les mounts non autorisés
lxc.mount.auto = proc:mixed sys:ro cgroup:mixed
EOF

# Démarrer le conteneur sécurisé
pct start 200

# Vérifier l'isolation UID
pct exec 200 -- id
pct exec 200 -- cat /proc/self/uid_map
```

> ✗ **DANGER**
> Les conteneurs LXC **privilégiés** (sans `unprivileged: 1`) s'exécutent avec les UID réels du système hôte. Un processus root dans un conteneur privilégié peut potentiellement s'échapper et compromettre l'hôte. Toujours utiliser des conteneurs non-privilégiés en production.

10.5.3 Sécurisation des VM Windows

Les VM Windows nécessitent des configurations spécifiques pour réduire leur surface d'attaque.

```
# Créer une VM Windows Server 2025 sécurisée
qm create 300 \
  --name "win-server-2025-prod" \
  --memory 8192 \
  --cores 4 \
  --sockets 1 \
```

```
  --cpu "x86-64-v3,hidden=1" \
  --machine q35 \
  --bios ovmf \
  --efidisk0 local-lvm:1,efitype=4m,pre-enrolled-keys=1 \
  --tpmstate0 local-lvm:1,version=v2.0 \
  --scsihw virtio-scsi-pci \
  --net0 virtio,bridge=vmbr1,tag=30,firewall=1 \
  --vga virtio \
  --tablet 1 \
  --agent enabled=1,fstrim_cloned_disks=1

# Activer Secure Boot et TPM 2.0 (anti-rootkit)
qm set 300 --tpmstate0 local-lvm:1,version=v2.0
qm set 300 --efidisk0 local-lvm:1,efitype=4m,pre-enrolled-keys=1

# Masquer la virtualisation (anti-VM-escape pour les malwares)
qm set 300 --cpu "host,hidden=1,flags=+pcid"

# Optimisations VirtIO (drivers réseau/disque haute performance + sécurisés)
qm set 300 --net0 "virtio,bridge=vmbr1,tag=30,firewall=1,queues=4"
```

Script de configuration post-déploiement Windows (via guest agent)

```
# Script PowerShell à exécuter dans la VM Windows après déploiement
# Durcissement Windows Server pour Proxmox

# Désactiver les services inutiles
$services_to_disable = @(
    'RemoteRegistry',
    'Spooler',
    'Fax',
    'SSDPSRV',
    'upnphost',
    'WinRM'
)

foreach ($svc in $services_to_disable) {
    Stop-Service -Name $svc -Force -ErrorAction SilentlyContinue
    Set-Service -Name $svc -StartupType Disabled -ErrorAction SilentlyContinue
    Write-Host "Service désactivé: $svc"
}

# Activer Windows Defender et le pare-feu
Set-MpPreference -DisableRealtimeMonitoring $false
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled True

# Configurer l'audit Windows
auditpol /set /category:"Logon/Logoff" /success:enable /failure:enable
auditpol /set /category:"Object Access" /success:disable /failure:enable
auditpol /set /category:"Privilege Use" /success:disable /failure:enable
auditpol /set /category:"Account Management" /success:enable /failure:enable
```

10.5.4 Chiffrement des disques VM

Le chiffrement des disques protège les données en cas de vol ou d'accès physique non autorisé aux supports de stockage.

```
# Méthode 1 : LUKS sur le disque de la VM (depuis l'intérieur de la VM)
# Se connecter à la VM et configurer LUKS
# Depuis l'intérieur de la VM :
cryptsetup luksFormat --type luks2 \
  --cipher aes-xts-plain64 \
  --key-size 512 \
  --hash sha256 \
  --pbkdf argon2id \
  /dev/sdb

cryptsetup luksOpen /dev/sdb data-crypt
```

```
mkfs.ext4 /dev/mapper/data-crypt
mount /dev/mapper/data-crypt /mnt/data

# Configurer le déverrouillage au démarrage
echo "data-crypt /dev/sdb none luks,discard" >> /etc/crypttab

# Méthode 2 : LUKS au niveau de l'hôte Proxmox sur le thin pool
# Chiffrer un disque dédié au stockage VM
cryptsetup luksFormat \
  --type luks2 \
  --cipher aes-xts-plain64 \
  --key-size 512 \
  --hash sha256 \
  --pbkdf argon2id \
  /dev/sdc

cryptsetup luksOpen /dev/sdc pve-data-crypt

# Créer un LVM sur le volume chiffré
pvcreate /dev/mapper/pve-data-crypt
vgcreate pve-data-vg /dev/mapper/pve-data-crypt

# Ajouter comme stockage Proxmox
pvesm add lvmthin pve-data-crypt-lvm \
  --vgname pve-data-vg \
  --thinpool data \
  --content images,rootdir

# Vérifier l'état du chiffrement
cryptsetup status pve-data-crypt
```

10.6 Conformité et audit

10.6.1 Logs d'audit Proxmox

Proxmox VE génère des logs d'audit détaillés pour toutes les opérations administratives.

```
# Logs d'audit Proxmox dans /var/log/pveproxy/
tail -f /var/log/pveproxy/access.log

# Format : [DATE] CLIENT USER ACTION PATH STATUS
# Exemple : [05/Mar/2026:14:23:11 +0100] "10.10.0.10" "adminpve@pve" "PUT"
#           "/api2/json/nodes/pve01/qemu/100/status/start" 200

# Parser les logs d'accès Proxmox - afficher les erreurs
awk '{print $1, $2, $3, $7, $9}' /var/log/pveproxy/access.log | \
  grep -v "200 " | head -50

# Rechercher les échecs d'authentification
grep "401\|403\|authentication failure" /var/log/daemon.log | \
  awk '{print $1, $2, $3, $NF}' | \
  sort | uniq -c | sort -rn | head -20

# Logs de tâches Proxmox (création/suppression VM, migrations...)
journalctl -u pvedaemon --since "2026-03-01" | grep -i "TASK\|ERROR\|WARN"

# Audit des connexions SSH
ausearch -k ssh-keys --start recent
ausearch -k proxmox-config --start today | aureport -f -i

# Générer un rapport d'activité quotidien
aureport --start today --end now --summary
aureport --start today --end now --login
```

10.6.2 CIS Benchmark appliqué à Proxmox

Le CIS (Center for Internet Security) publie des benchmarks de sécurité pour Debian Linux, applicables aux nœuds Proxmox VE.

```
# Installer l'outil de vérification CIS (Lynis)
apt install -y lynis

# Effectuer un audit CIS complet
lynis audit system --tests-from-group authentication,malware,networking,storage

# Audit ciblé pour Proxmox
lynis audit system \
  --profile /etc/lynis/custom-proxmox.prf \
  --log-file /var/log/lynis-proxmox.log \
  --report-file /var/log/lynis-report-proxmox.dat

# Voir le score et les recommandations
lynis show details
grep "^warning\|^suggestion" /var/log/lynis-report-proxmox.dat

# Profil Lynis personnalisé pour Proxmox
cat > /etc/lynis/custom-proxmox.prf << 'EOF'
# Profil Lynis TechFlow pour Proxmox VE
# Désactiver les tests non applicables à Proxmox

# Proxmox a son propre système de mises à jour
skip-test=PKGS-7370

# Proxmox utilise des utilisateurs système spécifiques
skip-test=AUTH-9284

# Les interfaces de bridge sont normales pour Proxmox
skip-test=NETW-3200
EOF
```

Contrôles CIS essentiels pour Proxmox

Contrôle CIS	Description	Statut requis
CIS 1.1.1	Partitions séparées /tmp, /var, /var/log	Recommandé
CIS 1.6.1	AppArmor activé et en mode enforce	Obligatoire
CIS 3.5.1	Firewall actif (pve-firewall)	Obligatoire
CIS 4.1.1	Auditd installé et actif	Obligatoire
CIS 5.2.11	PermitRootLogin prohibit-password	Obligatoire
CIS 5.2.14	PasswordAuthentication no	Obligatoire
CIS 5.3.1	Politique de mots de passe PAM	Obligatoire
CIS 6.2.1	Root le seul UID 0	Obligatoire

```
# Script de vérification CIS rapide pour Proxmox
cat > /usr/local/bin/cis-check-proxmox.sh << 'SCRIPT'
#!/bin/bash
# CIS Quick Check - TechFlow Proxmox
PASS=0; FAIL=0

check() {
  local desc="$1"; local cmd="$2"; local expected="$3"
  result=$(eval "$cmd" 2>/dev/null)
  if echo "$result" | grep -q "$expected"; then
    echo "[PASS] $desc"
    ((PASS++))
  else
    echo "[FAIL] $desc (attendu: $expected, obtenu: $result)"
    ((FAIL++))
  fi
}

check "AppArmor actif" \
  "aa-status --json | python3 -c \"import json,sys; d=json.load(sys.stdin);
```

```
print(d['status'])\"" \
  "enabled"
check "SSH PasswordAuth désactivée" \
  "sshd -T | grep passwordauthentication" \
  "passwordauthentication no"
check "SSH PermitRoot" \
  "sshd -T | grep permitrootlogin" \
  "prohibit-password"
check "Auditd actif" "systemctl is-active auditd" "active"
check "pve-firewall actif" "systemctl is-active pve-firewall" "active"
check "ASLR niveau 2" "sysctl -n kernel.randomize_va_space" "2"
check "kptr_restrict" "sysctl -n kernel.kptr_restrict" "2"
check "TCP SYN cookies" "sysctl -n net.ipv4.tcp_syncookies" "1"
check "Pas de UID dupliqués" \
  "awk -F: '(\$3 == 0){print \$1}' /etc/passwd | wc -l" \
  "^1$"

echo ""
echo "Résultat: $PASS PASS / $FAIL FAIL"
SCRIPT

chmod +x /usr/local/bin/cis-check-proxmox.sh
/usr/local/bin/cis-check-proxmox.sh
```

10.6.3 Scan de vulnérabilités

```
# Scanner les ports ouverts sur les nœuds Proxmox depuis le réseau management
nmap -sV -sC -O \
  --script "vuln and safe" \
  -p 22,2222,8006,3128,5405,5900-5999 \
  10.10.0.11,10.10.0.12,10.10.0.13 \
  -oA /var/log/scans/proxmox-vuln-$(date +%Y%m%d)

# Vérifier les CVE connues dans les paquets installés
apt install -y debsecan

# Générer un rapport des vulnérabilités sur le système
debsecan --suite bookworm --format report --only-fixed

# Rapport des vulnérabilités critiques non corrigées
debsecan --suite bookworm --format detail --only-fixed 2>/dev/null | \
  grep -A2 "CRITICAL\|HIGH"

# Vérifier Proxmox VE contre la base CVE publique
curl -s "https://cve.circl.lu/api/search/proxmox/proxmox-ve" | \
  python3 -c "
import json, sys
cves = json.load(sys.stdin)
for cve in sorted(cves, key=lambda x: x.get('cvss', 0), reverse=True)[:10]:
    print(f\"{cve['id']}: CVSS={cve.get('cvss', 'N/A')} - {cve['summary'][:80]}
\")"
```

10.6.4 Gestion des CVE et mises à jour

```
# Configurer les mises à jour automatiques de sécurité
apt install -y unattended-upgrades

cat > /etc/apt/apt.conf.d/50unattended-upgrades << 'EOF'
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}-security";
    "Proxmox:${distro_codename}";
};
Unattended-Upgrade::DevRelease "false";
Unattended-Upgrade::AutoFixInterruptedDpkg "true";
Unattended-Upgrade::MinimalSteps "true";
Unattended-Upgrade::InstallOnShutdown "false";
Unattended-Upgrade::Mail "admin@techflow.lan";
Unattended-Upgrade::MailReport "on-change";
```

```
Unattended-Upgrade::Remove-Unused-Dependencies "true";
Unattended-Upgrade::Automatic-Reboot "false";
Unattended-Upgrade::Automatic-Reboot-Time "03:00";
EOF

cat > /etc/apt/apt.conf.d/20auto-upgrades << 'EOF'
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "7";
APT::Periodic::Unattended-Upgrade "1";
EOF

# Procédure de mise à jour manuelle Proxmox VE
proxmox_update() {
  echo "=== Mise à jour Proxmox VE $(date) ==="

  # 1. Vérifier l'état du cluster avant mise à jour
  pvecm status
  pve-firewall status

  # 2. Mettre à jour les dépôts
  apt update

  # 3. Lister les mises à jour disponibles
  apt list --upgradable 2>/dev/null

  # 4. Appliquer les mises à jour (avec confirmation)
  apt full-upgrade -y

  # 5. Vérifier si redémarrage nécessaire
  if [ -f /var/run/reboot-required ]; then
    echo "AVERTISSEMENT: Redémarrage requis"
    cat /var/run/reboot-required.pkgs
  fi
}

# Planifier les vérifications CVE hebdomadaires
cat > /etc/cron.weekly/proxmox-cve-check << 'EOF'
#!/bin/bash
debsecan --suite bookworm --format report --only-fixed | \
  mail -s "CVE Report - $(hostname) - $(date +%Y-%m-%d)" admin@techflow.lan
EOF
chmod +x /etc/cron.weekly/proxmox-cve-check
```

10.7 Lab TechFlow SAS : sécurité complète

Ce lab applique l'ensemble des concepts vus dans ce chapitre à l'infrastructure TechFlow SAS, un cluster Proxmox VE 9 à trois nœuds hébergeant des applications de gestion d'entreprise.

Architecture TechFlow SAS

- Cluster 3 nœuds : `pve01.techflow.lan`, `pve02.techflow.lan`, `pve03.techflow.lan`
- Stockage : Ceph distribué + NFS pour les sauvegardes
- VM/CT : ~50 instances (applicatifs ERP, bases de données, monitoring)
- Équipes : admins (2 personnes), ops-team (5 personnes), readonly (10 personnes)

10.7.1 Matrice RBAC TechFlow

Utilisateurs et groupes

Utilisateur	Groupe	Realm	2FA	Rôle assigné	Périmètre
`adminpve@pve`	admins	PVE	TOTP obligatoire	Administrator	`/` (global)
`admin2@pve`	admins	PVE	WebAuthn	Administrator	`/` (global)
`ops01@pve`	ops-team	PVE	TOTP obligatoire	OpsTeamRole	`/pool/production`
`ops02@pve`	ops-team	PVE	TOTP	OpsTeamRole	`/pool/production`

Utilisateur	Groupe	Realm	2FA	Rôle assigné	Périmètre
`ops03@pve`	ops-team	PVE	TOTP	OpsTeamRole	`/pool/production`
`readonly@pve`	readonly	PVE	Non (réseau interne)	PVEAuditor	`/` (global)
`monitoring@pve`	readonly	PVE	Token uniquement	PVEAuditor	`/` (global)

```
# Déploiement complet de la matrice RBAC TechFlow SAS

# 1. Créer les groupes
pveum group add admins --comment "Administrateurs Proxmox TechFlow"
pveum group add ops-team --comment "Équipe opérations TechFlow"
pveum group add readonly --comment "Accès lecture seule TechFlow"

# 2. Créer le rôle OpsTeamRole
pveum role add OpsTeamRole \
  --privs "VM.Allocate,VM.Clone,VM.Config.CDROM,VM.Config.CPU,VM.Config.Disk,\
VM.Config.Memory,VM.Config.Network,VM.Console,VM.Monitor,VM.PowerMgmt,\
VM.Snapshot,Datastore.AllocateSpace,Datastore.Audit,Sys.Audit,\
Pool.Audit,SDN.Audit"

# 3. Créer les utilisateurs
pveum user add adminpve@pve --firstname "Admin" --lastname "Principal" \
  --email "adminpve@techflow.lan" --groups admins --comment "Compte admin
principal"

pveum user add admin2@pve --firstname "Admin" --lastname "Backup" \
  --email "admin2@techflow.lan" --groups admins --comment "Compte admin
secondaire"

pveum user add ops01@pve --firstname "Ops" --lastname "User01" \
  --email "ops01@techflow.lan" --groups ops-team

pveum user add ops02@pve --firstname "Ops" --lastname "User02" \
  --email "ops02@techflow.lan" --groups ops-team

pveum user add ops03@pve --firstname "Ops" --lastname "User03" \
  --email "ops03@techflow.lan" --groups ops-team

pveum user add readonly@pve --firstname "Read" --lastname "Only" \
  --email "readonly@techflow.lan" --groups readonly

pveum user add monitoring@pve --firstname "Monitoring" --lastname "Bot" \
  --email "monitoring@techflow.lan" --groups readonly

# 4. Définir les mots de passe
for user in adminpve admin2 ops01 ops02 ops03 readonly; do
  pveum passwd ${user}@pve
done

# 5. Assigner les ACL
# Admins : accès global
pveum acl modify / --groups admins --roles Administrator

# Ops team : accès au pool production uniquement
pveum acl modify /pool/production --groups ops-team --roles OpsTeamRole
pveum acl modify /storage/ceph-prod --groups ops-team --roles PVEDatastoreUser

# ReadOnly : audit global
pveum acl modify / --groups readonly --roles PVEAuditor

# 6. Créer le pool production
pveum pool add production --comment "Pool VM/CT production TechFlow"

# Ajouter les VM au pool (exemple)
pveum pool modify production --vms 100,101,102,200,201,300,301

# 7. Créer les API tokens
```

```
pveum user token add monitoring@pve prometheus \
  --privsep 1 \
  --comment "Token Prometheus monitoring"
pveum acl modify / --tokens "monitoring@pve!prometheus" --roles PVEAuditor

pveum user token add adminpve@pve terraform \
  --privsep 1 \
  --comment "Token Terraform IaC"
pveum acl modify /nodes --tokens "adminpve@pve!terraform" --roles PVEAdmin
pveum acl modify /storage --tokens "adminpve@pve!terraform" --roles
PVEDatastoreAdmin
pveum acl modify /vms --tokens "adminpve@pve!terraform" --roles PVEVMAdmin

# 8. Forcer le 2FA TOTP pour le realm
pveum realm modify pve --tfa type=totp

echo "Matrice RBAC TechFlow configurée avec succès"
pveum acl list
```

10.7.2 Durcissement des 3 nœuds

Script de durcissement appliqué de manière identique sur les trois nœuds TechFlow.

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════════
# TechFlow SAS — Script de durcissement Proxmox VE 9
# Version : 2026.03
# Usage : ./harden-proxmox-node.sh [pve01|pve02|pve03]
# ═══════════════════════════════════════════════════════════════

set -euo pipefail

NODE_NAME="${1:-$(hostname)}"
LOG_FILE="/var/log/hardening-${NODE_NAME}-$(date +%Y%m%d).log"

log() { echo "[$(date +%H:%M:%S)] $*" | tee -a "$LOG_FILE"; }

log "═══ Démarrage durcissement $NODE_NAME ═══"

# --- 1. Mise à jour du système ---
log "1. Mise à jour système..."
apt update && apt full-upgrade -y

# --- 2. Installation des outils de sécurité ---
log "2. Installation des outils..."
apt install -y \
  fail2ban \
  auditd \
  audispd-plugins \
  apparmor \
  apparmor-utils \
  apparmor-profiles \
  libpam-pwquality \
  unattended-upgrades \
  rkhunter \
  lynis \
  net-tools

# --- 3. SSH Hardening ---
log "3. Configuration SSH..."
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
cat > /etc/ssh/sshd_config.d/99-techflow-hardening.conf << 'SSHEOF'
Port 2222
PermitRootLogin prohibit-password
PasswordAuthentication no
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
PermitEmptyPasswords no
MaxAuthTries 3
```

```
MaxSessions 4
LoginGraceTime 30
ClientAliveInterval 300
ClientAliveCountMax 2
X11Forwarding no
AllowAgentForwarding no
AllowTcpForwarding no
PermitTunnel no
LogLevel VERBOSE
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com
KexAlgorithms curve25519-sha256,diffie-hellman-group16-sha512
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com
SSHEOF
sshd -t && systemctl restart sshd
log "SSH durci sur le port 2222"

# --- 4. Sysctl kernel hardening ---
log "4. Paramètres kernel ..."
cat > /etc/sysctl.d/99-techflow-hardening.conf << 'SYSCTLEOF'
kernel.randomize_va_space = 2
kernel.kptr_restrict = 2
kernel.dmesg_restrict = 1
kernel.sysrq = 0
kernel.yama.ptrace_scope = 1
fs.suid_dumpable = 0
fs.protected_symlinks = 1
fs.protected_hardlinks = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.log_martians = 1
net.ipv4.icmp_echo_ignore_broadcasts = 1
SYSCTLEOF
sysctl --system > /dev/null

# --- 5. Auditd ---
log "5. Configuration auditd ..."
cat > /etc/audit/rules.d/99-techflow.rules << 'AUDITEOF'
-D
-b 8192
-f 1
-w /etc/pve/ -p wa -k proxmox-config
-w /etc/ssh/ -p wa -k ssh-config
-w /etc/passwd -p wa -k identity
-w /etc/shadow -p wa -k identity
-a always,exit -F arch=b64 -S execve -F euid=0 -k root-commands
-w /usr/bin/sudo -p x -k sudo-exec
-e 2
AUDITEOF
augenrules --load
systemctl enable --now auditd

# --- 6. Fail2ban ---
log "6. Configuration fail2ban ..."
cat > /etc/fail2ban/jail.d/techflow-proxmox.conf << 'F2BEOF'
[proxmox-web]
enabled = true
port = 8006
filter = proxmox
logpath = /var/log/daemon.log
maxretry = 5
bantime = 3600

[sshd]
enabled = true
port = 2222
```

```
maxretry = 3
bantime = 7200
F2BEOF
systemctl enable --now fail2ban

# --- 7. Services inutiles ---
log "7. Désactivation des services inutiles..."
for svc in bluetooth avahi-daemon cups rpcbind; do
  systemctl stop "$svc" 2>/dev/null || true
  systemctl disable "$svc" 2>/dev/null || true
  systemctl mask "$svc" 2>/dev/null || true
done

# --- 8. Politique de mots de passe PAM ---
log "8. Politique mots de passe..."
cat > /etc/security/pwquality.conf << 'PWEOF'
minlen = 14
minclass = 3
maxrepeat = 2
gecoscheck = 1
dictcheck = 1
usercheck = 1
enforcing = 1
PWEOF

# --- 9. Vérification finale ---
log "9. Vérification finale..."
for svc in sshd auditd fail2ban pve-firewall; do
  status=$(systemctl is-active "$svc" 2>/dev/null || echo "inactive")
  log "Service $svc: $status"
done

log "═══ Durcissement $NODE_NAME terminé ═══"
log "Log complet : $LOG_FILE"
log "Exécuter 'lynis audit system' pour vérification complète"
```

Exécution sur les 3 nœuds TechFlow

```
# Exécuter le script sur chaque nœud (depuis pve01)
for node in pve01 pve02 pve03; do
  echo "═══ Durcissement de $node ═══"
  scp harden-proxmox-node.sh root@${node}.techflow.lan:/tmp/
  ssh root@${node}.techflow.lan "bash /tmp/harden-proxmox-node.sh ${node}"
done

# Vérifier l'état post-durcissement sur tous les nœuds
for node in pve01 pve02 pve03; do
  echo "--- $node ---"
  ssh root@${node}.techflow.lan /usr/local/bin/cis-check-proxmox.sh 2>/dev/null |
\
    grep -E "PASS|FAIL"
done
```

10.7.3 Firewall par zone réseau

La configuration firewall TechFlow suit une approche par zone réseau, inspirée des principes Zero Trust.

```
# Déploiement firewall complet TechFlow SAS

# Étape 1 : Configuration cluster.fw
cat > /etc/pve/firewall/cluster.fw << 'EOF'
[OPTIONS]
enable: 1
policy_in: DROP
policy_out: ACCEPT
log_ratelimit: enable=1,rate=5/second,burst=10

[ALIASES]
```

```
bastion = 10.10.0.10
ntp1 = 10.10.0.53
ldap = 10.10.0.100
siem = 10.20.0.5
pbs = 10.40.0.5

[IPSET mgmt-hosts]
10.10.0.0/24 comment Réseau management TechFlow
10.200.0.0/24 comment VPN WireGuard admins

[IPSET cluster-nodes]
10.10.10.11 comment pve01 cluster
10.10.10.12 comment pve02 cluster
10.10.10.13 comment pve03 cluster

[IPSET ceph-nodes]
10.10.20.11 comment pve01 ceph
10.10.20.12 comment pve02 ceph
10.10.20.13 comment pve03 ceph

[IPSET monitoring-hosts]
10.10.2.10 comment prometheus-01
10.10.2.11 comment grafana-01

[RULES]
# Management : SSH sécurisé depuis VPN/bastion uniquement
IN ACCEPT -source +mgmt-hosts -p tcp -dport 2222 -log nolog
# Interface web Proxmox
IN ACCEPT -source +mgmt-hosts -p tcp -dport 8006 -log nolog
# Corosync cluster heartbeat
IN ACCEPT -source +cluster-nodes -p udp -dport 5405:5406 -log nolog
# Ceph MON/OSD
IN ACCEPT -source +ceph-nodes -p tcp -dport 3300 -log nolog
IN ACCEPT -source +ceph-nodes -p tcp -dport 6789 -log nolog
IN ACCEPT -source +ceph-nodes -p tcp -dport 6800:7300 -log nolog
# Migration VM entre nœuds
IN ACCEPT -source +cluster-nodes -p tcp -dport 60000:60050 -log nolog
# Console SPICE/VNC
IN ACCEPT -source +mgmt-hosts -p tcp -dport 3128 -log nolog
IN ACCEPT -source +mgmt-hosts -p tcp -dport 5900:5999 -log nolog
# Monitoring Prometheus
IN ACCEPT -source +monitoring-hosts -p tcp -dport 9100 -log nolog
# Sauvegardes PBS
IN ACCEPT -source $pbs -p tcp -dport 8007 -log nolog
EOF

# Étape 2 : Firewall hôte pve01
cat > /etc/pve/nodes/pve01/host.fw << 'EOF'
[OPTIONS]
enable: 1
nf_conntrack_max: 262144
nf_conntrack_tcp_timeout_established: 3600

[RULES]
# Accepter le trafic loopback
IN ACCEPT -i lo
# ICMP depuis management
IN ACCEPT -source 10.10.0.0/24 -p icmp
# NTP sortant
OUT ACCEPT -p udp -dport 123
EOF

# Étape 3 : Appliquer et recharger
pve-firewall compile && pve-firewall reload
pve-firewall status

# Vérifier les chaînes générées
iptables -L -n --line-numbers | grep -A5 "PROXMOX"
```

10.7.4 Checklist de conformité

Checklist finale de sécurité pour valider le déploiement TechFlow SAS.

```
#!/bin/bash
# ═══════════════════════════════════════════════════════════
# TechFlow SAS - Checklist de conformité Proxmox VE 9
# ═══════════════════════════════════════════════════════════

PASS=0; FAIL=0; WARN=0
REPORT="/var/log/techflow-security-check-$(date +%Y%m%d_%H%M).txt"

pass() { echo "[PASS] $1" | tee -a "$REPORT"; ((PASS++)); }
fail() { echo "[FAIL] $1" | tee -a "$REPORT"; ((FAIL++)); }
warn() { echo "[WARN] $1" | tee -a "$REPORT"; ((WARN++)); }

echo "═══ TechFlow SAS - Rapport de conformité Proxmox VE ═══" | tee "$REPORT"
echo "Date: $(date)" | tee -a "$REPORT"
echo "Noeud: $(hostname)" | tee -a "$REPORT"
echo "═══════════════════════════════════════════════════════" | tee -a
"$REPORT"

echo "" | tee -a "$REPORT"
echo "--- 1. AUTHENTIFICATION ET RBAC ---" | tee -a "$REPORT"

# Vérifier les utilisateurs sans 2FA
users_no_2fa=$(pveum user list --output-format json 2>/dev/null | \
  python3 -c "
import json,sys
d = json.load(sys.stdin)
no2fa = [u['userid'] for u in d.get('data',[]) if not u.get('keys','')]
print('\n'.join(no2fa))" 2>/dev/null)

if [ -z "$users_no_2fa" ]; then
  pass "Tous les utilisateurs ont le 2FA configuré"
else
  fail "Utilisateurs sans 2FA: $users_no_2fa"
fi

# Vérifier le realm 2FA
realm_2fa=$(pveum realm list --output-format json 2>/dev/null | \
  python3 -c "
import json,sys
d = json.load(sys.stdin)
for r in d.get('data',[]):
  if r['realm'] == 'pve':
    print(r.get('tfa','none'))" 2>/dev/null)

if echo "$realm_2fa" | grep -q "totp"; then
  pass "Realm PVE : TOTP obligatoire configuré"
else
  warn "Realm PVE : 2FA non forcé au niveau realm"
fi

echo "" | tee -a "$REPORT"
echo "--- 2. DURCISSEMENT SSH ---" | tee -a "$REPORT"

sshd -T 2>/dev/null | grep -q "passwordauthentication no" && \
  pass "SSH : PasswordAuthentication désactivée" || \
  fail "SSH : PasswordAuthentication ACTIVEE"

sshd -T 2>/dev/null | grep -q "permitrootlogin prohibit-password" && \
  pass "SSH : PermitRootLogin = prohibit-password" || \
  fail "SSH : PermitRootLogin non sécurisé"

sshd -T 2>/dev/null | grep -q "port 2222" && \
  pass "SSH : Port non standard (2222)" || \
  warn "SSH : Port par défaut 22 utilisé"
```

```
sshd -T 2>/dev/null | grep -q "x11forwarding no" && \
  pass "SSH : X11Forwarding désactivé" || \
  fail "SSH : X11Forwarding activé"

echo "" | tee -a "$REPORT"
echo "--- 3. FIREWALL ---" | tee -a "$REPORT"

systemctl is-active pve-firewall &>/dev/null && \
  pass "Firewall Proxmox : actif" || \
  fail "Firewall Proxmox : INACTIF"

fw_policy=$(grep "^policy_in" /etc/pve/firewall/cluster.fw 2>/dev/null | awk
'{print $2}')
[ "$fw_policy" = "DROP" ] && \
  pass "Firewall : politique entrant = DROP" || \
  fail "Firewall : politique entrant = ${fw_policy:-non définie} (devrait être
DROP)"

echo "" | tee -a "$REPORT"
echo "--- 4. KERNEL HARDENING ---" | tee -a "$REPORT"

[ "$(sysctl -n kernel.randomize_va_space 2>/dev/null)" = "2" ] && \
  pass "ASLR : niveau maximum (2)" || \
  fail "ASLR : niveau insuffisant"

[ "$(sysctl -n kernel.kptr_restrict 2>/dev/null)" = "2" ] && \
  pass "kptr_restrict = 2" || \
  fail "kptr_restrict non configuré"

[ "$(sysctl -n kernel.sysrq 2>/dev/null)" = "0" ] && \
  pass "SysRq : désactivé" || \
  warn "SysRq : activé (acceptable en debug)"

[ "$(sysctl -n net.ipv4.tcp_syncookies 2>/dev/null)" = "1" ] && \
  pass "TCP SYN cookies : activés" || \
  fail "TCP SYN cookies : désactivés"

echo "" | tee -a "$REPORT"
echo "--- 5. SERVICES ET DÉMONS ---" | tee -a "$REPORT"

for svc in auditd fail2ban; do
  systemctl is-active "$svc" &>/dev/null && \
    pass "Service actif : $svc" || \
    fail "Service inactif : $svc"
done

for svc in bluetooth avahi-daemon cups; do
  ! systemctl is-active "$svc" &>/dev/null && \
    pass "Service désactivé : $svc" || \
    warn "Service actif (devrait être désactivé) : $svc"
done

echo "" | tee -a "$REPORT"
echo "--- 6. APPARMOR ---" | tee -a "$REPORT"

systemctl is-active apparmor &>/dev/null && \
  pass "AppArmor : actif" || \
  fail "AppArmor : INACTIF"

enforce_count=$(aa-status 2>/dev/null | grep "profiles are in enforce mode" | awk
'{print $1}')
[ "${enforce_count:-0}" -gt 0 ] && \
  pass "AppArmor : $enforce_count profils en mode enforce" || \
  warn "AppArmor : aucun profil en mode enforce"

echo "" | tee -a "$REPORT"
echo "--- 7. LOGGING ET AUDIT ---" | tee -a "$REPORT"
```

```
auditctl -l 2>/dev/null | grep -q "proxmox-config" && \
  pass "Auditd : règles Proxmox actives" || \
  fail "Auditd : règles Proxmox manquantes"

[ -f /var/log/pveproxy/access.log ] && \
  pass "Logs accès Proxmox : présents" || \
  warn "Logs accès Proxmox : absents"

echo "" | tee -a "$REPORT"
echo "--- 8. MISES À JOUR ---" | tee -a "$REPORT"

updates=$(apt list --upgradable 2>/dev/null | grep -c "upgradable" || echo 0)
[ "$updates" -lt 5 ] && \
  pass "Mises à jour : $updates en attente (OK)" || \
  warn "Mises à jour : $updates en attente (appliquer rapidement)"

systemctl is-enabled unattended-upgrades &>/dev/null && \
  pass "Mises à jour automatiques : activées" || \
  warn "Mises à jour automatiques : désactivées"

echo "" | tee -a "$REPORT"
echo "═══════════════════════════════════════════════════════" | tee -a
"$REPORT"
echo "RÉSULTAT FINAL : $PASS PASS | $FAIL ECHECS | $WARN AVERTISSEMENTS" | tee -a
"$REPORT"
echo "Rapport complet : $REPORT" | tee -a "$REPORT"

if [ "$FAIL" -gt 0 ]; then
  echo ""
  echo "ATTENTION : $FAIL contrôle(s) en échec. Corriger avant mise en
production."
  exit 1
fi
exit 0
```

Résultat attendu après déploiement complet TechFlow SAS

```
═══ TechFlow SAS – Rapport de conformité Proxmox VE ═══
Date: Tue Mar 17 09:00:00 CET 2026
Noeud: pve01

--- 1. AUTHENTIFICATION ET RBAC ---
[PASS] Tous les utilisateurs ont le 2FA configuré
[PASS] Realm PVE : TOTP obligatoire configuré

--- 2. DURCISSEMENT SSH ---
[PASS] SSH : PasswordAuthentication désactivée
[PASS] SSH : PermitRootLogin = prohibit-password
[PASS] SSH : Port non standard (2222)
[PASS] SSH : X11Forwarding désactivé

--- 3. FIREWALL ---
[PASS] Firewall Proxmox : actif
[PASS] Firewall : politique entrant = DROP

--- 4. KERNEL HARDENING ---
[PASS] ASLR : niveau maximum (2)
[PASS] kptr_restrict = 2
[PASS] SysRq : désactivé
[PASS] TCP SYN cookies : activés

--- 5. SERVICES ET DÉMONS ---
[PASS] Service actif : auditd
[PASS] Service actif : fail2ban
[PASS] Service désactivé : bluetooth
[PASS] Service désactivé : avahi-daemon
[PASS] Service désactivé : cups
```

```
--- 6. APPARMOR ---
[PASS] AppArmor : actif
[PASS] AppArmor : 14 profils en mode enforce

--- 7. LOGGING ET AUDIT ---
[PASS] Auditd : règles Proxmox actives
[PASS] Logs accès Proxmox : présents

--- 8. MISES À JOUR ---
[PASS] Mises à jour : 0 en attente (OK)
[PASS] Mises à jour automatiques : activées

============================================================
RÉSULTAT FINAL : 18 PASS | 0 ECHECS | 0 AVERTISSEMENTS
```

Récapitulatif

Ce chapitre a couvert l'ensemble de la sécurisation d'une infrastructure Proxmox VE 9 :

Modèle de sécurité — Les realms PAM, PVE, LDAP et OpenID offrent une flexibilité d'authentification adaptée à chaque organisation. Le RBAC granulaire permet d'appliquer le principe du moindre privilège. Le 2FA (TOTP ou WebAuthn) et les API tokens complètent ce modèle.

Durcissement des nœuds — SSH durci (clés uniquement, port 2222), désactivation des services inutiles, paramètres sysctl de protection kernel, AppArmor pour l'isolation des processus, auditd pour la traçabilité de toutes les actions administratives.

Firewall multicouche — L'architecture datacenter / nœud / VM de Proxmox permet une défense en profondeur. Les IPSets et alias facilitent la gestion des règles complexes. La politique `policy_in: DROP` est la configuration minimale acceptable en production.

Sécurité réseau — La séparation physique ou VLAN des réseaux (management, VM, Ceph, cluster) limite la propagation des attaques. Le chiffrement Corosync protège les échanges intra-cluster. WireGuard sécurise les accès distants.

Sécurité des VM et conteneurs — Vérification cryptographique des images, conteneurs LXC non-privilégiés avec AppArmor, Secure Boot et TPM pour les VM Windows, LUKS pour le chiffrement des données au repos.

Conformité — Lynis pour les audits CIS, debsecan pour le suivi des CVE, auditd pour la traçabilité réglementaire, et un processus de mise à jour documenté.

Le lab TechFlow SAS illustre l'application pratique de ces concepts avec des configurations réelles et un script de vérification automatisé permettant de valider la posture de sécurité à tout moment.

→ VOIR AUSSI

- Documentation officielle Proxmox VE : https://pve.proxmox.com/wiki/Firewall
- CIS Benchmark Debian Linux : https://www.cisecurity.org/benchmark/debian_linux
- ANSSI - Guide de durcissement Debian : https://www.ssi.gouv.fr
- Lynis Security Auditing : https://cisofy.com/lynis/
- WireGuard VPN : https://www.wireguard.com/

11 Optimisation des performances

L'infrastructure virtuelle de TechFlow SAS héberge une centaine de machines virtuelles et une cinquantaine de conteneurs LXC. Les équipes constatent des latences anormales sur la base de données PostgreSQL, une saturation sporadique du réseau et des pics CPU inexpliqués sur les nœuds Proxmox. Ce chapitre répond à ces problèmes concrets en proposant une méthodologie d'optimisation systématique, couche par couche : CPU, mémoire, stockage, réseau, puis conteneurs.

L'optimisation des performances dans un environnement virtualisé est un exercice de compromis. Chaque gain obtenu sur une ressource peut en dégrader une autre. La discipline consiste à mesurer avant d'agir, à isoler les variables et à valider chaque modification par un benchmark reproductible. Ce chapitre fournit les commandes, les paramètres et les explications conceptuelles nécessaires pour mener cette démarche avec rigueur sur Proxmox VE 9.

11.1 Optimisation CPU

Le processeur est souvent la première ressource sous pression dans un cluster de virtualisation. Proxmox VE expose plusieurs mécanismes permettant d'ajuster finement la façon dont les vCPU des machines virtuelles interagissent avec les CPU physiques de l'hôte.

11.1.1 Choix du type de CPU (host, kvm64, custom)

Le type de CPU exposé à la machine virtuelle détermine quelles instructions du processeur physique sont disponibles à l'intérieur de la VM. Ce choix a un impact direct sur les performances brutes et sur la portabilité des VM entre nœuds.

Comparatif des types de CPU disponibles

Types de CPU QEMU/KVM dans Proxmox VE 9

Type CPU	Instructions exposées	Migration live	Cas d'usage recommandé
`kvm64`	Minimum garanti (x86-64 v1)	Oui (tous nœuds)	Environnements hétérogènes, haute portabilité
`host`	Identiques au CPU physique	Non (même modèle CPU uniquement)	Performances maximales, cluster homogène
`x86-64-v2`	SSE4.2, POPCNT	Oui (nœuds v2+)	Workloads modernes sans AVX
`x86-64-v3`	AVX, AVX2, BMI	Oui (nœuds v3+)	Calcul scientifique, chiffrement
`x86-64-v4`	AVX-512	Oui (nœuds v4+)	HPC, machine learning
`custom`	Définition manuelle par flags	Dépend des flags	Contrôle granulaire, tests spécifiques

Le type `host` est le plus performant car il évite toute émulation d'instruction : la VM exécute les instructions directement sur le silicon. En contrepartie, la migration live n'est possible que vers un nœud disposant exactement du même modèle de CPU.

Pour un cluster homogène (tous les nœuds avec des Intel Xeon Gold 6338 identiques), `host` est le choix optimal :

```
# Définir le type CPU d'une VM existante (VMID 200)
qm set 200 --cpu host

# Vérifier la configuration résultante
qm config 200 | grep cpu
```

```
# Pour un type personnalisé avec flags spécifiques
qm set 200 --cpu host,flags=+aes;+avx512f
```

Pour un cluster hétérogène où les migrations doivent rester possibles partout, préférez un profil standardisé :

```
# Profil x86-64-v3 : bon compromis performance/portabilité
qm set 200 --cpu x86-64-v3

# Vérifier les flags CPU disponibles sur l'hôte
grep flags /proc/cpuinfo | head -1 | tr ' ' '\n' | sort
```

Un profil CPU personnalisé peut être défini dans la configuration de la VM :

```
# Extrait /etc/pve/nodes/pve1/qemu-server/200.conf
cpu: host,flags=+pdpe1gb;+aes
cores: 8
sockets: 1
numa: 1
```

> ✔ **CONSEIL**
> Pour les VM hébergeant des bases de données (PostgreSQL, MySQL), le flag `+aes` est particulièrement important : il active l'accélération matérielle AES-NI pour le chiffrement des connexions SSL/TLS et peut réduire la latence des requêtes chiffrées de 15 à 30 %.

11.1.2 NUMA topology et pinning

NUMA (Non-Uniform Memory Access) est une architecture où les cœurs CPU d'un socket accèdent plus rapidement à la mémoire physiquement attachée à ce socket qu'à la mémoire de l'autre socket. Ignorer la topologie NUMA dans une configuration de virtualisation est l'une des causes les plus fréquentes de dégradation de performances sur les serveurs multi-sockets.

Vérifier la topologie NUMA de l'hôte :

```
# Topologie NUMA du système hôte
numactl --hardware
# available: 2 nodes (0-1)
# node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 24 25 26 27 28 29 30 31 32 33 34 35
# node 0 size: 128985 MB
# node 0 free: 98234 MB
# node 1 cpus: 12 13 14 15 16 17 18 19 20 21 22 23 36 37 38 39 40 41 42 43 44 45
46 47
# node 1 size: 128993 MB
# node 1 free: 101456 MB
# node distances:
# node   0   1
#   0:  10  21
#   1:  21  10

# Statistiques NUMA globales
numastat

# Statistiques NUMA par processus QEMU
numastat -p $(pgrep -f "kvm.*200")
```

Proxmox VE permet d'exposer la topologie NUMA à la VM et de la contraindre à des ressources NUMA spécifiques :

```
# Activer NUMA dans la configuration de la VM
qm set 200 --numa 1

# Définir un nœud NUMA pour la VM (nœud NUMA 0 de l'hôte)
qm set 200 --numa0 cpus=0-11,memory=65536

# Configuration complète pour une VM base de données sur NUMA node 0
qm set 200 \
  --numa 1 \
  --numa0 cpus=0-7,hostnodes=0,memory=32768,policy=bind \
  --cores 8 \
```

```
  --sockets 1 \
  --memory 32768
```

> ⚠ **AVERTISSEMENT**
> Évitez d'allouer plus de vCPU ou de mémoire qu'un nœud NUMA physique ne peut en fournir. Une VM de 24 vCPU sur un serveur avec 12 cœurs par socket NUMA devra accéder aux deux nœuds, annulant le bénéfice du pinning. Dimensionnez les VM en multiples entiers de la taille d'un nœud NUMA.

11.1.3 CPU affinity et cgroups v2

Le pinning CPU (ou CPU affinity) consiste à forcer le scheduler Linux à exécuter les threads d'une VM sur des cœurs physiques spécifiques. Cela élimine le coût de migration de contexte entre cœurs et améliore la localité de cache.

```
# Identifier le PID du processus QEMU de la VM 200
KVM_PID=$(pgrep -f "kvm.*id 200")

# Voir l'affinité actuelle
taskset -cp $KVM_PID

# Épingler le processus principal sur les cœurs 0-7
taskset -cp 0-7 $KVM_PID

# Épingler chaque thread vCPU individuellement
for TID in $(ls /proc/$KVM_PID/task/); do
  taskset -cp 0-7 $TID 2>/dev/null
done
```

Proxmox VE 9 utilise cgroups v2 par défaut. Les contraintes CPU sont gérées via les paramètres `cpulimit` et `cpuunits` :

```
# Limiter la VM 200 à 4 CPU équivalents (même si elle a 8 vCPU)
qm set 200 --cpulimit 4

# Définir la priorité relative (weight) — défaut : 100, max : 10000
qm set 200 --cpuunits 2048

# Vérifier les paramètres cgroup de la VM
cat /sys/fs/cgroup/pve/qemu.slice/qemu-200.scope/cpu.max
cat /sys/fs/cgroup/pve/qemu.slice/qemu-200.scope/cpu.weight
```

Pour le pinning permanent, configurez le paramètre `affinity` directement dans le fichier de configuration de la VM :

```
# /etc/pve/nodes/pve1/qemu-server/200.conf
affinity: 0-7
cpulimit: 8
cpuunits: 2048
```

```
# Pour un pinning exclusif des cœurs hôte (isolation noyau)
# Ajouter dans /etc/default/grub :
# GRUB_CMDLINE_LINUX="isolcpus=8-15 nohz_full=8-15 rcu_nocbs=8-15"
update-grub
# Redémarrer puis vérifier
cat /proc/cmdline | grep isolcpus
```

11.1.4 Hugepages et THP

Les hugepages réduisent la pression sur le TLB (Translation Lookaside Buffer) en utilisant des pages mémoire de 2 Mo ou 1 Go au lieu des pages standard de 4 Ko. Pour une VM avec 32 Go de RAM, cela réduit le nombre d'entrées TLB de 8 millions à seulement 16 384.

Hugepages statiques

```
# Vérifier le support des hugepages 1G sur l'hôte
grep pdpe1gb /proc/cpuinfo | head -1

# Hugepages 2M dynamiques (sans redémarrage)
```

```
echo 16384 > /proc/sys/vm/nr_hugepages
# 16384 × 2 Mo = 32 Go réservés

# Rendre persistant
echo "vm.nr_hugepages = 16384" > /etc/sysctl.d/hugepages.conf
sysctl -p /etc/sysctl.d/hugepages.conf

# Vérifier les hugepages disponibles
cat /proc/meminfo | grep -i huge

# Activer les hugepages pour une VM Proxmox
qm set 200 --hugepages 2    # pages de 2 Mo
# ou
qm set 200 --hugepages 1024 # pages de 1 Go

# Hugepages par nœud NUMA
echo 8192 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
echo 8192 > /sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr_hugepages

# Vérifier par nœud NUMA
for node in /sys/devices/system/node/node*/hugepages/hugepages-2048kB/; do
  echo "$node : $(cat ${node}nr_hugepages) pages"
done
```

Pour les hugepages 1G (nécessite `pdpe1gb` dans les flags CPU), configurer au boot :

```
# Dans /etc/default/grub :
# GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=64"
# (réserve 64 Go en hugepages 1G)
update-grub
```

Transparent Hugepages (THP)

Les THP sont gérées automatiquement par le kernel. Elles peuvent être bénéfiques pour les workloads généralistes mais néfastes pour les bases de données :

```
# Vérifier l'état actuel des THP
cat /sys/kernel/mm/transparent_hugepage/enabled
# [always] madvise never

# Désactiver THP pour les workloads base de données
echo madvise > /sys/kernel/mm/transparent_hugepage/enabled
echo defer+madvise > /sys/kernel/mm/transparent_hugepage/defrag

# Persistant via service systemd
cat > /etc/systemd/system/disable-thp.service << 'EOF'
[Unit]
Description=Disable Transparent Hugepages
After=sysinit.target

[Service]
Type=oneshot
ExecStart=/bin/sh -c 'echo madvise > /sys/kernel/mm/transparent_hugepage/enabled'
ExecStart=/bin/sh -c 'echo defer+madvise > /sys/kernel/mm/transparent_hugepage/
defrag'
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
EOF

systemctl enable --now disable-thp.service
```

✔ CONSEIL

Pour PostgreSQL et Oracle, désactivez THP à la fois sur l'hôte Proxmox et à l'intérieur de la VM guest. PostgreSQL gère lui-même ses allocations mémoire via `shared_buffers` et les THP peuvent provoquer des latences de compactage mémoire intempestives (pauses de 50 à 500 ms observables dans les logs).

11.1.5 KSM (Kernel Same-page Merging)

KSM permet au kernel de fusionner des pages mémoire identiques entre plusieurs VM, réduisant la consommation mémoire globale au prix d'un overhead CPU.

```
# Vérifier l'état de KSM
cat /sys/kernel/mm/ksm/run
# 0 = arrêté, 1 = actif

# Activer KSM
echo 1 > /sys/kernel/mm/ksm/run

# Paramétrer l'agressivité du scan
echo 1000 > /sys/kernel/mm/ksm/pages_to_scan  # pages scannées par cycle
echo 20   > /sys/kernel/mm/ksm/sleep_millisecs # intervalle entre cycles

# Statistiques KSM
cat /sys/kernel/mm/ksm/pages_shared    # pages partagées actuellement
cat /sys/kernel/mm/ksm/pages_sharing   # pages qui utilisent le partage
cat /sys/kernel/mm/ksm/pages_unshared  # pages candidates non encore fusionnées

# Calculer le gain mémoire en Mo
SHARED=$(cat /sys/kernel/mm/ksm/pages_shared)
echo "Mémoire économisée : $(( SHARED * 4 / 1024 )) Mo"
```

> ⚠ **AVERTISSEMENT**
> KSM introduit une vulnérabilité aux attaques de type side-channel (timing attacks) sur les environnements multi-tenant. Dans un contexte d'hébergement mutualisé non maîtrisé, désactivez KSM. Pour un usage interne où toutes les VM appartiennent à la même organisation, le risque est acceptable et les gains peuvent atteindre 20 à 40 % de la RAM physique sur des VM Linux homogènes.

11.2 Optimisation mémoire

La mémoire est souvent la ressource la plus contrainte dans un cluster de virtualisation dense. Proxmox VE propose plusieurs mécanismes pour en maximiser l'utilisation effective.

11.2.1 Ballooning et overcommit

Le ballooning permet à l'hyperviseur de récupérer dynamiquement de la mémoire auprès des VM lorsque l'hôte en a besoin. Le driver balloon dans le guest signale les pages libres à l'hyperviseur qui peut les réutiliser.

```
# Activer le ballooning sur une VM (minimum mémoire = 512 Mo)
qm set 200 --balloon 512
# La VM démarre avec toute sa RAM configurée ;
# le balloon peut la réduire jusqu'à 512 Mo si nécessaire.
# balloon=0 désactive le ballooning.

# Monitorer l'état du balloon en temps réel via le moniteur QEMU
qm monitor 200
# Dans le moniteur interactif :
# (qm) info balloon
# balloon: actual=8192
```

Configuration YAML résultante :

```
# /etc/pve/nodes/pve1/qemu-server/200.conf
memory: 16384
balloon: 4096
```

Overcommit mémoire : il est possible d'allouer plus de RAM virtuelle que de RAM physique disponible sur l'hôte.

Ratios d'overcommit recommandés

Ratio d'overcommit	Workload adapté	Risque
1:1 (100 %)	Production critique, bases de données	Aucun
1:1.2 (120 %)	Serveurs applicatifs avec pics ponctuels	Faible

1:1.5 (150 %)	VM de développement, tests	Modéré
1:2 (200 %)	VM légères dormantes majoritairement	Élevé

```
# Calculer l'overcommit actuel du nœud
TOTAL_RAM=$(free -m | awk '/Mem:/ {print $2}')
ALLOC_RAM=$(qm list | tail -n +2 | awk '{sum += $4} END {print sum}')
echo "RAM physique    : ${TOTAL_RAM} Mo"
echo "RAM allouée VM  : ${ALLOC_RAM} Mo"
echo "Ratio overcommit: $(echo "scale=2; $ALLOC_RAM / $TOTAL_RAM" | bc)"
```

⚠ AVERTISSEMENT

Un overcommit supérieur à 150 % sur des VM en production active expose l'hôte à l'OOM killer du kernel Linux. L'OOM killer peut terminer arbitrairement des processus QEMU, causant des crashs VM non planifiés. Configurez des alertes sur `MemAvailable` dans Proxmox et maintenez une réserve minimale de 10 % de RAM physique libre à tout moment.

11.2.2 Huge pages statiques vs transparentes

Le choix entre hugepages statiques et THP dépend du workload :

Comparatif hugepages statiques vs THP

Critère	Hugepages statiques 2M/1G	Transparent Hugepages (THP)
Configuration	Manuelle, au démarrage	Automatique, dynamique
Flexibilité	Faible (réservation fixe)	Élevée
Latence d'allocation	Nulle (pré-allouées)	Possible (compaction mémoire)
Workloads bénéficiaires	Bases de données, HPC	Workloads généralistes
Overhead admin	Élevé	Minimal
Compatibilité QEMU	Totale (via `--hugepages`)	Partielle

11.2.3 Surveillance de la pression mémoire

Linux kernel expose des métriques de pression mémoire via PSI (Pressure Stall Information), disponibles depuis le kernel 4.20 :

```
# Pression mémoire globale
cat /proc/pressure/memory
# some avg10=0.00 avg60=0.23 avg300=0.15 total=45123456
# full avg10=0.00 avg60=0.00 avg300=0.00 total=12345

# Surveiller en temps réel
watch -n 1 cat /proc/pressure/memory

# Script d'alerte si pression > 5 % sur 60s
PSI_60=$(awk '/some/ {print $3}' /proc/pressure/memory | cut -d= -f2)
if (( $(echo "$PSI_60 > 5.0" | bc -l) )); then
  logger -t proxmox-perf "ALERTE: Pression mémoire élevée: ${PSI_60}%"
fi

# Statistiques NUMA pour détecter les accès cross-node
numastat -m | grep -E "MemTotal|MemFree|MemUsed"

# Pages en swap par VM (via cgroups v2)
cat /sys/fs/cgroup/pve/qemu.slice/qemu-200.scope/memory.swap.current
```

✔ CONSEIL

Intégrez la surveillance PSI dans vos alertes Proxmox ou Prometheus. Un `avg60 > 10 %` sur la pression mémoire est le signal que le nœud commence à swapper activement et que des VM vont subir des dégradations de performances. Agissez avant d'atteindre ce seuil.

11.2.4 Swap et swappiness

Le swap doit être considéré comme un filet de sécurité, non comme une extension de la RAM. Sur un hôte Proxmox, réduire `swappiness` permet de retarder le recours au swap au maximum.

```
# Voir le swappiness actuel (défaut kernel : 60 — trop agressif pour un
hyperviseur)
cat /proc/sys/vm/swappiness

# Réduire à 10 (swap uniquement en dernier recours)
echo 10 > /proc/sys/vm/swappiness

# Persistant
echo "vm.swappiness = 10" >> /etc/sysctl.d/proxmox-perf.conf
sysctl -p /etc/sysctl.d/proxmox-perf.conf

# Surveiller l'utilisation du swap
free -h
swapon --show

# Identifier les processus qui swappent le plus
for pid in /proc/[0-9]*/status; do
  name=$(grep Name $pid | awk '{print $2}')
  vmswap=$(grep VmSwap $pid | awk '{print $2}')
  if [ ! -z "$vmswap" ] && [ "$vmswap" -gt 0 ]; then
    echo "$name: ${vmswap} kB"
  fi
done 2>/dev/null | sort -t: -k2 -rn | head -10

# Pour les nœuds critiques, désactiver le swap complètement
swapoff -a
# Commenter aussi la ligne swap dans /etc/fstab
```

11.3 Optimisation stockage

Le stockage est généralement le goulot d'étranglement le plus visible dans un environnement virtualisé. Les optimisations à ce niveau ont souvent l'impact le plus spectaculaire sur les performances perçues.

11.3.1 IO schedulers (mq-deadline, none pour NVMe)

Le scheduler d'I/O du kernel Linux gère la file d'attente des requêtes disque. Le choix optimal dépend du type de médium sous-jacent.

Schedulers I/O Linux et cas d'usage

Scheduler	Médium adapté	Avantages	Inconvénients
`none`	NVMe, SSD très rapides	Zéro overhead, latence minimale	Pas de réordonnancement
`mq-deadline`	SSD SATA/SAS, HDD SAS	Garantit des deadlines, anti-starvation	Léger overhead
`kyber`	NVMe, SSD	Contrôle de latence cible	Moins mature
`bfq`	HDD, mixte	Priorités par processus, fairness	Overhead élevé, inadapté au NVMe

```
# Voir le scheduler actuel de chaque disque
for disk in /sys/block/sd* /sys/block/nvme*; do
  echo "$disk: $(cat $disk/queue/scheduler 2>/dev/null)"
done

# Changer le scheduler pour un NVMe
echo none > /sys/block/nvme0n1/queue/scheduler

# Changer pour un SSD SATA
echo mq-deadline > /sys/block/sda/queue/scheduler

# Rendre persistant via udev
cat > /etc/udev/rules.d/60-io-scheduler.rules << 'EOF'
# NVMe → none
ACTION=="add|change", KERNEL=="nvme[0-9]*", ATTR{queue/scheduler}="none"
# SSD SATA (rotation=0) → mq-deadline
```

```
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0", \
  ATTR{queue/scheduler}="mq-deadline"
# HDD (rotation=1) → mq-deadline
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1", \
  ATTR{queue/scheduler}="mq-deadline"
EOF

udevadm control --reload-rules && udevadm trigger

# Optimisations supplémentaires pour NVMe
echo 0 > /sys/block/nvme0n1/queue/add_random   # pas de contribution à l'entropy
pool
echo 2 > /sys/block/nvme0n1/queue/rq_affinity  # affinité requêtes au CPU local
```

11.3.2 VirtIO-SCSI vs VirtIO-BLK

Proxmox propose deux principaux bus de stockage virtuel. Le choix impacte les performances, la scalabilité et les fonctionnalités disponibles.

Comparatif VirtIO-BLK vs VirtIO-SCSI

Critère	VirtIO-BLK	VirtIO-SCSI
Performance brute (IOPS)	Très élevée	Élevée (légèrement inférieure)
Nombre de disques par VM	Limité (26 avec PCI)	Jusqu'à 255 par contrôleur
Support TRIM/Discard	Oui	Oui
Support WWN (identité disque)	Non	Oui
Multiqueue	Oui (`iothread`)	Oui (native)
Hotplug disques	Non	Oui
Recommandation Proxmox 9	VM avec un seul disque	VM avec plusieurs disques

```
# Configurer VirtIO-SCSI avec iothread
qm set 200 \
  --scsihw virtio-scsi-pci \
  --scsi0 local-nvme:vm-200-disk-0,cache=none,discard=on,iothread=1,ssd=1

# Ajouter un second disque de données
qm set 200 \
  --scsi1 local-nvme:vm-200-disk-1,cache=none,discard=on,iothread=1,ssd=1

# Vérifier la configuration
qm config 200 | grep -E "scsihw|scsi[0-9]"
```

Configuration YAML résultante :

```
scsihw: virtio-scsi-pci
scsi0: local-nvme:vm-200-disk-0,cache=none,discard=on,iothread=1,ssd=1
scsi1: local-nvme:vm-200-disk-1,cache=none,discard=on,iothread=1,ssd=1
```

11.3.3 Discard/TRIM et provisioning fin

Le TRIM (SCSI DISCARD) permet à la VM de signaler à l'hôte les blocs qu'elle n'utilise plus. Cela est essentiel pour les stockages en thin provisioning (ZFS, LVM-thin, Ceph).

```
# Activer le discard sur un disque existant
qm set 200 --scsi0 local-nvme:vm-200-disk-0,discard=on

# Dans la VM Linux guest, vérifier que le discard est visible
# (en tant que root dans la VM) :
lsblk -D
# DISC-GRAN et DISC-MAX doivent être non-nuls

# Forcer un TRIM dans la VM guest
fstrim -av

# Configurer le TRIM périodique dans la VM (via systemd-fstrim)
systemctl enable --now fstrim.timer
```

```
systemctl status fstrim.timer

# Sur ZFS : vérifier que le provisioning fin est actif sur le dataset
zfs get volsize,refreservation rpool/data/vm-200-disk-0

# Activer le thin provisioning ZFS (refreservation=none)
zfs set refreservation=none rpool/data/vm-200-disk-0
```

11.3.4 Cache en écriture : writeback vs none

Le paramètre `cache` du disque virtuel détermine comment QEMU gère le cache d'écriture. Ce paramètre a un impact majeur sur les performances et la sécurité des données.

Modes de cache disque QEMU

Mode cache	O_DIRECT	O_SYNC	Performances	Sécurité données	Cas d'usage
`none`	Oui	Non	Très élevées	Élevée (host fs)	Production, NVMe avec capacitor
`writeback`	Non	Non	Maximales	Faible (risque crash)	Développement, tests
`writethrough`	Non	Oui	Faibles	Maximale	Données critiques sans BBU
`directsync`	Oui	Oui	Très faibles	Maximale	Archivage critique
`unsafe`	Non	Non	Maximales	Nulle	Import/export temporaire

```
# Mode recommandé pour la production (NVMe avec protection panne de courant)
qm set 200 --scsi0 local-nvme:vm-200-disk-0,cache=none

# Mode writeback pour les tests (ne PAS utiliser en production sans UPS)
qm set 200 --scsi0 local-nvme:vm-200-disk-0,cache=writeback

# Mesurer l'impact avec fio (dans la VM)
fio --name=randwrite --ioengine=libaio --iodepth=32 --rw=randwrite \
    --bs=4k --direct=1 --size=4G --numjobs=4 --runtime=60 \
    --group_reporting --filename=/dev/sda
```

⚠ AVERTISSEMENT

Le mode `writeback` ne doit jamais être utilisé sur des données de production sans un système d'alimentation sans interruption (UPS) ET une batterie de sauvegarde sur le contrôleur RAID (BBU). Une coupure de courant avec `writeback` actif peut corrompre les données de la VM. Pour les NVMe enterprise modernes avec capacitor de protection interne, `none` offre d'excellentes performances tout en étant sûr.

11.3.5 Optimisation ZFS (ARC, L2ARC, SLOG)

ZFS est le système de fichiers par défaut de Proxmox VE. Son cache adaptatif (ARC) et ses journaux (SLOG) offrent des leviers d'optimisation importants.

```
# Taille actuelle de l'ARC et statistiques d'efficacité
cat /proc/spl/kstat/zfs/arcstats | grep -E "^c |^size|^hits|^misses"

# Calculer le taux de hit du cache ARC
arc_hits=$(awk '/^hits / {print $3}' /proc/spl/kstat/zfs/arcstats)
arc_misses=$(awk '/^misses / {print $3}' /proc/spl/kstat/zfs/arcstats)
arc_ratio=$(echo "scale=2; $arc_hits * 100 / ($arc_hits + $arc_misses)" | bc)
echo "Taux de hit ARC : ${arc_ratio}%"

# Limiter l'ARC à 32 Go (pour laisser de la RAM aux VM)
echo $((32 * 1024 * 1024 * 1024)) > /sys/module/zfs/parameters/zfs_arc_max

# Rendre persistant
cat >> /etc/modprobe.d/zfs.conf << 'EOF'
options zfs zfs_arc_max=34359738368
options zfs zfs_arc_min=4294967296
EOF
```

```
# L2ARC : ajouter un SSD comme cache secondaire
zpool add data cache /dev/nvme1n1

# Vérifier l'état du L2ARC
zpool status data
cat /proc/spl/kstat/zfs/arcstats | grep l2

# SLOG sur NVMe pour améliorer les écritures synchrones
zpool add data log /dev/nvme2n1

# SLOG en miroir pour la redondance
zpool add data log mirror /dev/nvme2n1 /dev/nvme3n1

# Optimisations ZFS par dataset VM
zfs set atime=off rpool/data                    # désactiver les timestamps
d'accès
zfs set compression=lz4 rpool/data              # compression quasi gratuite
zfs set recordsize=16k rpool/data               # taille de bloc pour VM
génériques
zfs set recordsize=8k  rpool/data/vm-200-disk-0 # PostgreSQL avec blocs 8K
zfs set logbias=throughput rpool/data           # optimiser pour le débit
```

✔ CONSEIL

La compression LZ4 sur ZFS est presque toujours bénéfique : elle réduit la quantité de données écrites sur le disque et peut paradoxalement augmenter les performances I/O brutes en réduisant le volume transféré. L'overhead CPU est inférieur à 1 % sur les processeurs modernes disposant des extensions SIMD.

11.4 Optimisation réseau

Les performances réseau sont critiques pour les VM applicatives et les bases de données distribuées. Proxmox VE offre plusieurs mécanismes pour maximiser le débit et minimiser la latence.

11.4.1 VirtIO multiqueue

Le multiqueue réseau permet à plusieurs vCPU de traiter simultanément le trafic réseau, éliminant le goulot d'étranglement de la file d'attente unique.

```
# Activer multiqueue sur une interface réseau (autant de queues que de vCPU)
qm set 200 --net0 virtio,bridge=vmbr0,queues=8

# Dans la VM Linux guest, activer les queues
ethtool -L eth0 combined 8

# Vérifier les queues actives dans la VM
ethtool -l eth0
ls /sys/class/net/eth0/queues/

# Surveiller l'utilisation des queues réseau sur l'hôte
watch -n 1 'cat /proc/interrupts | grep -E "virtio|eth"'

# Optimiser l'IRQ affinity : chaque queue IRQ sur un cœur dédié
# Lister les IRQ VirtIO
grep virtio /proc/interrupts | awk '{print $1}' | tr -d ':'

# Affecter l'IRQ 45 (queue 0) au CPU 0, l'IRQ 46 (queue 1) au CPU 1, etc.
echo 1 > /proc/irq/45/smp_affinity
echo 2 > /proc/irq/46/smp_affinity
```

✔ CONSEIL

Le nombre de queues VirtIO doit correspondre au nombre de vCPU de la VM. Au-delà, il n'y a pas de gain supplémentaire. Pour une VM avec 8 vCPU, définissez `queues=8`. Combiné avec le CPU pinning, assurez-vous que chaque queue IRQ est affinée au bon cœur CPU pour optimiser la localité de cache L3.

11.4.2 SR-IOV (Single Root I/O Virtualization)

SR-IOV permet à une carte réseau physique de se présenter comme plusieurs fonctions PCIe indépendantes (Virtual Functions). Chaque VF peut être assignée directement à une VM, contournant totalement la couche de virtualisation réseau.

```
# Vérifier le support SR-IOV du NIC
lspci -v | grep -A 10 "Ethernet"
cat /sys/bus/pci/devices/0000:02:00.0/sriov_totalvfs

# Activer les Virtual Functions (exemple avec Intel X550)
echo 8 > /sys/bus/pci/devices/0000:02:00.0/sriov_numvfs

# Vérifier les VF créées
lspci | grep "Virtual Function"

# Rendre persistant via udev
cat > /etc/udev/rules.d/80-sriov.rules << 'EOF'
ACTION=="add", SUBSYSTEM=="pci", ATTR{vendor}=="0x8086", ATTR{device}=="0x1563", \
  ATTR{sriov_numvfs}="8"
EOF

# Activer IOMMU dans /etc/default/grub
# Intel : GRUB_CMDLINE_LINUX="intel_iommu=on iommu=pt"
# AMD   : GRUB_CMDLINE_LINUX="amd_iommu=on iommu=pt"
update-grub && reboot

# Vérifier que l'IOMMU est actif
dmesg | grep -i iommu | head -5
find /sys/kernel/iommu_groups/ -type l | wc -l

# Passer une VF à une VM Proxmox (passthrough PCIe)
qm set 200 --hostpci0 0000:02:10.0,pcie=1
```

> ⚠ **AVERTISSEMENT**
> SR-IOV avec passthrough PCIe désactive la migration live de la VM : la VF est physiquement liée au nœud. Planifiez les maintenances en conséquence — la VM doit être arrêtée proprement avant toute opération sur le nœud hôte. Utilisez SR-IOV uniquement pour les VM nécessitant des performances réseau extrêmes (>25 Gbit/s effectif ou latence <100 µs).

11.4.3 DPDK et vhost-net

`vhost-net` est un mécanisme kernel qui permet au driver réseau d'effectuer le traitement des paquets directement dans l'espace kernel, sans repasser par l'espace utilisateur QEMU. Il est activé automatiquement par Proxmox pour les interfaces VirtIO.

```
# Vérifier que vhost-net est chargé
lsmod | grep vhost

# Charger si nécessaire et rendre persistant
modprobe vhost_net
echo "vhost_net" >> /etc/modules

# Vérifier l'utilisation de vhost pour une VM
ls -la /dev/vhost-net
# vhost-net est activé automatiquement pour model=virtio dans Proxmox
```

Pour les workloads nécessitant des performances réseau de datacenter (NFV, télécommunications), DPDK peut être configuré dans les VM :

```
# Dans la VM guest : installation DPDK (Debian/Ubuntu)
apt install dpdk dpdk-dev -y

# Identifier les interfaces réseau
dpdk-devbind.py --status

# Charger le module vfio-pci dans la VM
modprobe vfio-pci
```

```
# Lier une interface réseau VirtIO au driver DPDK
dpdk-devbind.py --bind=vfio-pci 0000:00:03.0

# Test de performance DPDK avec testpmd (mode macswap)
dpdk-testpmd -l 0-3 -n 4 -- --nb-cores=2 --forward-mode=macswap
```

11.4.4 Jumbo frames et MTU

Les Jumbo frames (MTU > 1500) réduisent le nombre de paquets nécessaires pour transmettre de gros volumes de données, diminuant l'overhead CPU lié au traitement des en-têtes réseau.

```
# Vérifier la MTU actuelle de l'hôte
ip link show vmbr0
ip link show eth0

# Activer les jumbo frames sur l'interface physique (MTU 9000)
ip link set eth0 mtu 9000
ip link set vmbr0 mtu 9000

# Rendre persistant dans /etc/network/interfaces
# Ajouter dans la section de l'interface :
#     mtu 9000

# Recharger la configuration réseau
ifreload -a

# Dans la VM guest, adapter la MTU de l'interface virtuelle
ip link set eth0 mtu 9000
# Vérifier avec un grand ping
ping -M do -s 8972 192.168.1.1  # 8972 + 28 en-têtes = 9000
```

> ⚠ **AVERTISSEMENT**
> Tous les équipements réseau sur le chemin (switches, routeurs, NIC physiques) doivent supporter la même MTU. Un seul équipement ne supportant pas les Jumbo frames fragmentera ou rejettera silencieusement les paquets, causant des timeouts intermittents difficiles à diagnostiquer. Vérifiez la MTU de bout en bout avant tout déploiement en production.

11.4.5 Bonding LACP et load balancing

Le bonding réseau combine plusieurs interfaces physiques pour augmenter le débit agrégé et assurer la redondance.

```
# Configuration bonding LACP (802.3ad) dans /etc/network/interfaces
auto bond0
iface bond0 inet manual
    bond-slaves eth0 eth1
    bond-mode 802.3ad
    bond-miimon 100
    bond-lacp-rate fast
    bond-xmit-hash-policy layer2+3
    mtu 9000

auto vmbr0
iface vmbr0 inet static
    address 10.0.1.10/24
    gateway 10.0.1.1
    bridge-ports bond0
    bridge-stp off
    bridge-fd 0
    mtu 9000

# Appliquer sans redémarrage
ifreload -a

# Vérifier l'état du bond et de ses membres
cat /proc/net/bonding/bond0
```

```
# Tester le failover
ip link set eth0 down
cat /proc/net/bonding/bond0 | grep "Active Slave"
ip link set eth0 up
```

11.5 Optimisation des conteneurs LXC

Les conteneurs LXC offrent des performances proches du natif avec un overhead minimal. L'optimisation porte principalement sur les limites de ressources et le choix du backend de stockage.

11.5.1 cgroups : limites CPU et mémoire

Les conteneurs LXC utilisent directement les cgroups v2 du kernel hôte pour limiter et partager les ressources.

```
# Configurer les limites d'un conteneur LXC (CT 300)
pct set 300 --cpulimit 4 --cpuunits 1024
pct set 300 --memory 8192 --swap 2048

# Vérifier les paramètres cgroup du conteneur
ls /sys/fs/cgroup/lxc.payload.300/

# Mémoire utilisée
cat /sys/fs/cgroup/lxc.payload.300/memory.current
cat /sys/fs/cgroup/lxc.payload.300/memory.max

# CPU time consommé
cat /sys/fs/cgroup/lxc.payload.300/cpu.stat

# Limiter les IOPS d'un conteneur (via cgroups io.max)
# Identifier le device major:minor du rootfs
DEVICE=$(stat -c "%d" /var/lib/lxc/300/rootfs)
MAJ=$((DEVICE / 256))
MIN=$((DEVICE % 256))
echo "${MAJ}:${MIN} riops=5000 wiops=5000 rbps=104857600 wbps=104857600" \
  > /sys/fs/cgroup/lxc.payload.300/io.max
```

Configuration dans le fichier de configuration LXC :

```
# /etc/pve/lxc/300.conf
arch: amd64
cores: 4
cpulimit: 4
cpuunits: 1024
memory: 8192
swap: 2048
rootfs: local-nvme:subvol-300-disk-0,size=20G
net0: name=eth0,bridge=vmbr0,firewall=1,ip=10.0.1.30/24,type=veth
```

11.5.2 Namespaces et overhead

Les namespaces Linux isolent les ressources entre conteneurs. Chaque namespace a un coût minimal mais mesurable :

```
# Namespaces d'un conteneur LXC actif
ls -la /proc/$(pgrep -f "lxc-start.*300")/ns/

# Mesure de l'overhead CPU dans le conteneur (comparaison hôte vs CT)
# Sur l'hôte natif :
time for i in $(seq 1 10000); do echo $i > /dev/null; done

# Dans le conteneur (devrait être quasi identique) :
pct enter 300
time for i in $(seq 1 10000); do echo $i > /dev/null; done

# Surveiller la charge créée par tous les conteneurs
systemd-cgtop -d 1 --depth=3
```

L'overhead total d'un conteneur LXC actif est typiquement inférieur à 5 Mo de RAM et négligeable en CPU, comparé à une VM QEMU complète qui nécessite de 80 à 200 Mo de RAM pour l'émulateur et le kernel guest.

11.5.3 Overlayfs vs bind mounts

Le backend de stockage des conteneurs impacte les performances I/O et la densité possible sur un nœud.

Backends de stockage LXC

Backend	Implémentation	Performances I/O	Cas d'usage
`dir`	Répertoire classique	Moyenne	Développement, tests
`zfs`	Dataset ZFS dédié	Élevée	Production, snapshots fréquents
`lvm-thin`	Volume LVM thin	Élevée	Clonage rapide, haute densité
`overlayfs`	Overlay filesystem	Moyenne	Images lecture seule partagées
`bind mount`	Montage direct hôte	Native	Performances maximales, données partagées

```
# Bind mount d'un répertoire hôte dans un conteneur
pct set 300 --mp0 /data/shared,mp=/mnt/shared,ro=0

# Vérifier les montages du conteneur
pct enter 300 -- mount | grep shared

# Mesurer les performances avec fio depuis le conteneur (bind mount)
pct enter 300 -- fio --name=test --ioengine=libaio --iodepth=16 --rw=randread \
    --bs=4k --direct=1 --size=1G --filename=/mnt/shared/test.dat

# ZFS sur conteneur : optimisations
zfs set compression=lz4 rpool/data/subvol-300-disk-0
zfs set recordsize=4k   rpool/data/subvol-300-disk-0  # petits fichiers web
zfs get compressratio   rpool/data/subvol-300-disk-0
```

11.6 Benchmarking et mesure

Toute optimisation doit être mesurée. Sans baseline, il est impossible de quantifier les gains et de valider les changements.

11.6.1 Outils : fio, iperf3, sysbench, stress-ng

```
# Installation des outils de benchmarking sur Debian/Proxmox
apt install fio iperf3 sysbench stress-ng sysstat ioping numactl -y

# Vérification des versions
fio --version
iperf3 --version
sysbench --version
stress-ng --version
```

11.6.2 Benchmarks CPU et mémoire

```
# Benchmark CPU single-thread avec sysbench
sysbench cpu --cpu-max-prime=20000 --time=30 run
# Sortie attendue :
# CPU speed: events per second: 892.45
# Latency (ms): min=1.11, avg=1.12, max=5.23, 95th percentile=1.14

# Benchmark CPU multi-thread (8 threads)
sysbench cpu --cpu-max-prime=20000 --time=30 --threads=8 run

# Benchmark mémoire : bande passante en écriture
sysbench memory --memory-block-size=1M --memory-total-size=100G \
    --memory-operation=write --threads=8 run

# Benchmark mémoire en lecture
sysbench memory --memory-block-size=1M --memory-total-size=100G \
```

```
    --memory-operation=read --threads=8 run

# Stress test avec métriques détaillées
stress-ng --cpu 8 --cpu-method matrixprod --timeout 60s --metrics-brief

# Test de performance de chaque vCPU individuellement
for cpu in $(seq 0 7); do
  taskset -c $cpu sysbench cpu --cpu-max-prime=20000 --time=10 run 2>/dev/null \
    | grep "events per second" \
    | awk -v c=$cpu '{printf "vCPU %d: %s eps\n", c, $4}'
done

# Benchmark NUMA : comparer latence locale vs remote
numactl --membind=0 sysbench memory --memory-block-size=1M \
    --memory-total-size=10G --memory-operation=read run
numactl --membind=1 sysbench memory --memory-block-size=1M \
    --memory-total-size=10G --memory-operation=read run
```

11.6.3 Benchmarks stockage

```
# Benchmark I/O aléatoire 4K en lecture (IOPS — typique base de données)
fio --name=rand-read-4k \
    --ioengine=libaio \
    --iodepth=32 \
    --rw=randread \
    --bs=4k \
    --direct=1 \
    --size=10G \
    --numjobs=4 \
    --runtime=60 \
    --group_reporting \
    --filename=/dev/sda

# Benchmark I/O aléatoire 4K en écriture (IOPS)
fio --name=rand-write-4k \
    --ioengine=libaio \
    --iodepth=32 \
    --rw=randwrite \
    --bs=4k \
    --direct=1 \
    --size=10G \
    --numjobs=4 \
    --runtime=60 \
    --group_reporting \
    --filename=/dev/sda

# Benchmark séquentiel en lecture (débit)
fio --name=seq-read \
    --ioengine=libaio \
    --iodepth=8 \
    --rw=read \
    --bs=1M \
    --direct=1 \
    --size=10G \
    --numjobs=1 \
    --runtime=60 \
    --group_reporting \
    --filename=/dev/sda

# Benchmark latence pure (iodepth=1 = un seul I/O à la fois)
fio --name=latency-test \
    --ioengine=libaio \
    --iodepth=1 \
    --rw=randread \
    --bs=4k \
    --direct=1 \
    --size=1G \
    --numjobs=1 \
```

```
    --runtime=30 \
    --filename=/dev/sda

# Mesure rapide de latence avec ioping
ioping -c 100 /dev/sda          # latence unitaire
ioping -A -c 100 /dev/sda       # mode asynchrone (IOPS)
ioping -D -s 1M -c 10 /dev/sda  # latence séquentielle

# Statistiques ZFS en temps réel
zpool iostat -v data 5
```

Valeurs de référence pour l'interprétation des résultats :

Valeurs de référence I/O par type de médium (2025)

Test	HDD 7200 RPM	SSD SATA	NVMe PCIe 4.0	NVMe PCIe 5.0
IOPS randread 4K	100–200	80K–100K	700K–1M	2M+
IOPS randwrite 4K	100–150	70K–90K	600K–900K	1.5M+
Débit séquentiel lecture	150–200 MB/s	500–560 MB/s	5–7 GB/s	10–14 GB/s
Latence randread (µs)	5 000–15 000	100–200	20–50	10–30

11.6.4 Benchmarks réseau

```
# Démarrer le serveur iperf3 sur le nœud de destination
iperf3 -s -D  # -D = mode daemon

# Test débit TCP depuis la VM vers le serveur (4 flux parallèles)
iperf3 -c 10.0.1.1 -t 60 -P 4

# Test débit UDP avec mesure de perte de paquets
iperf3 -c 10.0.1.1 -t 60 -u -b 10G

# Test bidirectionnel simultané
iperf3 -c 10.0.1.1 -t 60 --bidir

# Test avec jumbo frames
iperf3 -c 10.0.1.1 -t 60 -P 8 -M 8960 --set-mss 8960

# Mesure de latence réseau
ping -c 1000 -i 0.01 10.0.1.1 | tail -5

# Latence inter-VM sur le même hôte (doit être <100 µs)
# VM1 serveur :
netserver -D
# VM2 client (latence aller-retour) :
netperf -H 10.0.1.2 -t TCP_RR -l 30 -- -r 64,64

# Débit entre deux nœuds Proxmox (test de saturation du lien)
# Sur le nœud 2 :
iperf3 -s
# Sur le nœud 1 :
iperf3 -c pve2.techflow.local -t 60 -P 8 --forceflush
```

11.6.5 Interprétation et baseline

```
# Script de baseline complet avec rapport
#!/bin/bash
set -e
REPORT="/tmp/baseline-$(hostname)-$(date +%Y%m%d-%H%M).txt"

{
echo "═══ Baseline Proxmox VE — $(hostname) — $(date) ═══"
echo ""

echo "--- CPU Info ---"
lscpu | grep -E "Model name|Socket|Core|Thread|Max MHz|NUMA"
echo ""
```

```
echo "--- Memory Info ---"
free -h
numactl --hardware
echo ""

echo "--- Disk Scheduler ---"
for disk in /sys/block/sd* /sys/block/nvme*n1; do
  [ -e "$disk/queue/scheduler" ] && \
    echo "  $disk: $(cat $disk/queue/scheduler)"
done
echo ""

echo "--- Storage Benchmark randread 4K (30s) ---"
fio --name=baseline-read --ioengine=libaio --iodepth=32 --rw=randread \
    --bs=4k --direct=1 --size=2G --numjobs=4 --runtime=30 \
    --group_reporting --output-format=terse \
    --filename=/dev/nvme0n1 2>&1 | head -5
echo ""

echo "--- Network Info ---"
ip link show | grep -E "mtu|state UP" | awk '{print $2, $4, $9, $10}'
echo ""

echo "--- KSM ---"
echo "  pages_shared : $(cat /sys/kernel/mm/ksm/pages_shared)"
echo "  run          : $(cat /sys/kernel/mm/ksm/run)"
echo ""

echo "--- Memory Pressure (PSI) ---"
cat /proc/pressure/memory
} > "$REPORT"

echo "Rapport baseline sauvegardé dans $REPORT"
```

11.7 Lab TechFlow SAS : profils d'optimisation

Ce lab applique l'ensemble des techniques du chapitre à trois scénarios réels de TechFlow SAS. Chaque profil est documenté avec la configuration avant/après et les gains mesurés.

11.7.1 Profil VM base de données (haute I/O)

La VM 200 héberge PostgreSQL 16 pour l'ERP de TechFlow. Elle souffre de latences élevées lors des pics de charge (flush des WAL, opérations VACUUM).

Configuration initiale (avant optimisation)

```
# /etc/pve/nodes/pve1/qemu-server/200.conf — AVANT
cores: 8
sockets: 2
memory: 32768
cpu: kvm64
balloon: 4096
scsi0: local-lvm:vm-200-disk-0,cache=writeback,size=500G
net0: e1000,bridge=vmbr0
numa: 0
```

Diagnostic des problèmes

```
# Latence I/O excessive (mesurée dans la VM)
ioping -c 100 /dev/sda
# Latency : min=8.5ms avg=25.3ms max=187.2ms — inacceptable pour PostgreSQL

# Saturation CPU cross-NUMA
numastat -p $(pgrep postgres | head -1)
# numa_miss élevé : accès mémoire cross-node fréquents (~23%)

# Pression mémoire
awk '/some/ {print $3}' /proc/pressure/memory
```

```
# avg60=12.45 — pression mémoire significative, balloon actif

# Type CPU limitant (kvm64 sans AES-NI)
grep flags /proc/cpuinfo | grep -c aes
# 0 — AES-NI non exposé, SSL lent
```

Configuration optimisée

```
# /etc/pve/nodes/pve1/qemu-server/200.conf — APRÈS
arch: x86_64
cores: 8
sockets: 1
cpu: host,flags=+pdpe1gb;+aes
memory: 32768
balloon: 0
hugepages: 2
numa: 1
numa0: cpus=0-7,hostnodes=0,memory=32768,policy=bind
cpuunits: 4096
affinity: 0-7
scsihw: virtio-scsi-pci
scsi0: local-nvme:vm-200-disk-0,cache=none,discard=on,iothread=1,ssd=1
net0: virtio,bridge=vmbr0,queues=8,mtu=9000
```

Application des paramètres

```
# Arrêter la VM proprement
qm stop 200

# Appliquer la configuration
qm set 200 --cpu host,flags=+pdpe1gb\;+aes
qm set 200 --numa 1
qm set 200 --numa0 cpus=0-7,hostnodes=0,memory=32768,policy=bind
qm set 200 --hugepages 2 --balloon 0
qm set 200 --scsihw virtio-scsi-pci
qm set 200 --scsi0 local-nvme:vm-200-disk-0,cache=none,discard=on,iothread=1,ssd=1
qm set 200 --net0 virtio,bridge=vmbr0,queues=8,mtu=9000
qm set 200 --cpuunits 4096
qm set 200 --affinity 0-7

# Optimisations ZFS pour le dataset PostgreSQL
zfs set recordsize=8k  rpool/data/vm-200-disk-0
zfs set logbias=throughput rpool/data/vm-200-disk-0
zfs set primarycache=metadata rpool/data/vm-200-disk-0
zfs set compression=lz4 rpool/data/vm-200-disk-0

# Réserver les hugepages sur le bon nœud NUMA avant démarrage
echo 16384 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/
nr_hugepages

# Démarrer la VM
qm start 200
```

Optimisation PostgreSQL dans la VM

```
# Dans la VM 200 — configuration PostgreSQL optimisée
# /etc/postgresql/16/main/postgresql.conf

# Mémoire
# shared_buffers = 8GB           # 25% de la RAM allouée
# effective_cache_size = 24GB    # 75% de la RAM
# work_mem = 64MB                # per-sort, attention aux connexions multiples
# maintenance_work_mem = 2GB     # VACUUM, CREATE INDEX

# WAL
# wal_buffers = 64MB
# max_wal_size = 4GB
# checkpoint_completion_target = 0.9
# synchronous_commit = off       # pour workloads non critiques uniquement
```

```
# I/O
# effective_io_concurrency = 200 # NVMe avec plusieurs queues
# random_page_cost = 1.1         # NVMe quasi-aléatoire = ~séquentiel
# seq_page_cost = 1.0
```

11.7.2 Profil VM applicative (haute CPU)

La VM 210 exécute des jobs de traitement de données Python/Pandas en parallèle. Elle sature ses vCPU sans atteindre les limites mémoire ou I/O.

Configuration initiale

```
# /etc/pve/nodes/pve1/qemu-server/210.conf — AVANT
cores: 16
sockets: 2
memory: 16384
cpu: kvm64
balloon: 8192
scsi0: local-lvm:vm-210-disk-0,cache=writeback,size=100G
net0: e1000,bridge=vmbr0
```

Configuration optimisée

```
# /etc/pve/nodes/pve1/qemu-server/210.conf — APRÈS
arch: x86_64
cores: 16
sockets: 1
cpu: host
memory: 16384
balloon: 8192
cpuunits: 8192
affinity: 8-23
scsihw: virtio-scsi-pci
scsi0: local-nvme:vm-210-disk-0,cache=none,discard=on,iothread=1
net0: virtio,bridge=vmbr0,queues=4
```

```
# Appliquer la configuration CPU haute performance
qm set 210 --cpu host
qm set 210 --sockets 1 --cores 16
qm set 210 --cpuunits 8192
qm set 210 --affinity 8-23   # cœurs du nœud NUMA 1

# Basculer le gouverneur CPU en mode performance sur les cœurs utilisés
for cpu in $(seq 8 23); do
  echo performance > /sys/devices/system/cpu/cpu${cpu}/cpufreq/scaling_governor
done

# Vérifier les fréquences CPU
for cpu in $(seq 8 23); do
  freq=$(cat /sys/devices/system/cpu/cpu${cpu}/cpufreq/scaling_cur_freq)
  echo "CPU $cpu : $((freq/1000)) MHz"
done

# Dans la VM : variables d'environnement pour frameworks de calcul
# export OMP_NUM_THREADS=16
# export MKL_NUM_THREADS=16
# export OPENBLAS_NUM_THREADS=16
# export NUMBA_NUM_THREADS=16
```

11.7.3 Profil conteneur web (faible overhead)

Le conteneur 300 héberge Nginx + PHP-FPM pour le site web de TechFlow. L'objectif est de maximiser la densité (50 requêtes/s par CT) tout en minimisant l'utilisation de ressources.

Configuration optimisée

```
# /etc/pve/lxc/300.conf
arch: amd64
cores: 4
cpulimit: 2
cpuunits: 512
```

```
memory: 2048
swap: 512
rootfs: local-nvme:subvol-300-disk-0,size=10G,discard=1
net0: name=eth0,bridge=vmbr0,firewall=1,ip=10.0.1.30/24,type=veth,mtu=9000
mp0: /data/web/techflow,mp=/var/www/html,ro=0
```

```
# Créer le conteneur optimisé
pct create 300 local:vztmpl/debian-12-standard_12.7-1_amd64.tar.zst \
  --cores 4 \
  --cpulimit 2 \
  --cpuunits 512 \
  --memory 2048 \
  --swap 512 \
  --rootfs local-nvme:10 \
  --net0 name=eth0,bridge=vmbr0,ip=10.0.1.30/24,gw=10.0.1.1,mtu=9000 \
  --hostname web01 \
  --start 1

# Monter le répertoire web partagé en bind mount (I/O native)
pct set 300 --mp0 /data/web/techflow,mp=/var/www/html

# Limiter les IOPS pour éviter la contention (partage du NVMe)
echo "8:0 riops=1000 wiops=500 rbps=209715200 wbps=52428800" \
  > /sys/fs/cgroup/lxc.payload.300/io.max

# Vérifier que Nginx est opérationnel
pct exec 300 -- nginx -t
pct exec 300 -- systemctl status php8.2-fpm --no-pager

# Test de charge avec wrk (depuis un autre nœud)
wrk -t 4 -c 100 -d 30s http://10.0.1.30/
```

11.7.4 Résultats avant/après

Le tableau suivant synthétise les gains mesurés après application des profils d'optimisation sur l'infrastructure TechFlow SAS.

VM 200 — PostgreSQL (haute I/O)

Résultats VM 200 — PostgreSQL

Métrique	Avant	Après	Gain
Latence I/O randread 4K (µs)	25 300	85	−99.7 %
IOPS randread 4K	412	890 000	+216 000 %
Latence requête PostgreSQL p99	450 ms	12 ms	−97.3 %
RAM utilisée hôte (VM 200)	32 768 Mo	28 400 Mo	−13 % (KSM)
Accès mémoire cross-NUMA	23 %	0.3 %	−98.7 %
Transactions PostgreSQL/s (TPS)	1 240	18 750	+1 412 %

VM 210 — Applicatif Python (haute CPU)

Résultats VM 210 — Python/Pandas

Métrique	Avant	Après	Gain
Durée job traitement 10M lignes	8 min 42 s	3 min 17 s	−62.2 %
Fréquence CPU effective (GHz)	2.1	3.8	+81 %
Cache L3 miss rate	18.3 %	4.1 %	−77.6 %
sysbench CPU events/s (×16 threads)	4 230	11 890	+181 %

Conteneur 300 — Nginx/PHP-FPM (faible overhead)

Résultats CT 300 — Nginx

Métrique	Avant	Après	Gain
Requêtes/s (wrk, 100 connexions)	3 420	8 740	+155 %

Latence p99 (ms)	87	23	−73.6 %
RAM utilisée conteneur	1 240 Mo	680 Mo	−45.2 %
Débit réseau maximal atteint	450 Mbit/s	4.8 Gbit/s	+967 %

Script de validation post-optimisation

```
#!/bin/bash
# Validation des optimisations TechFlow SAS
echo "═══ Validation optimisations TechFlow SAS — $(date) ═══"

echo ""
echo "--- Pression système ---"
printf "  CPU  : "; awk '/some/ {print "avg60="$3}' /proc/pressure/cpu
printf "  RAM  : "; awk '/some/ {print "avg60="$3}' /proc/pressure/memory
printf "  I/O  : "; awk '/some/ {print "avg60="$3}' /proc/pressure/io

echo ""
echo "--- KSM ---"
pages=$(cat /sys/kernel/mm/ksm/pages_shared)
echo "  Pages partagées : $pages ($(( pages * 4 / 1024 )) Mo économisés)"

echo ""
echo "--- NUMA stats VM 200 ---"
numastat -p $(pgrep -f "kvm.*200" | head -1) 2>/dev/null | \
  grep -E "Numa Hit|Numa Miss|Interleave"

echo ""
echo "--- Hugepages ---"
grep -E "HugePages_Total|HugePages_Free|HugePages_Rsvd" /proc/meminfo

echo ""
echo "--- I/O schedulers ---"
for d in /sys/block/nvme*n1/queue/scheduler; do
  echo "  $d : $(cat $d)"
done

echo ""
echo "--- CPU governor (cœurs 0-7) ---"
for cpu in 0 4 7; do
  gov=$(cat /sys/devices/system/cpu/cpu${cpu}/cpufreq/scaling_governor 2>/dev/
null)
  echo "  CPU $cpu : ${gov:-N/A}"
done

echo ""
echo "═══ Validation terminée ═══"
```

Récapitulatif du chapitre

Ce chapitre a couvert l'ensemble des leviers d'optimisation disponibles dans Proxmox VE 9, de la couche CPU jusqu'au réseau, en passant par la mémoire, le stockage et les conteneurs LXC.

Points clés à retenir

- Le type de CPU `host` offre les meilleures performances sur cluster homogène ; utilisez un profil `x86-64-v3` pour la portabilité inter-nœuds avec des gains substantiels par rapport à `kvm64`.
- Le pinning NUMA est la première action à mener sur tout serveur bi-socket : les gains peuvent atteindre 40 à 80 % sur les workloads sensibles à la latence mémoire.
- Pour le stockage, le mode cache `none` avec VirtIO-SCSI et `iothread=1` sur NVMe est le profil optimal pour la production. Le mode `writeback` est à réserver aux environnements de développement.
- Les hugepages statiques éliminent les TLB misses pour les VM gourmandes en mémoire, particulièrement bénéfiques pour les bases de données (PostgreSQL, Oracle, MySQL InnoDB).
- Le multiqueue VirtIO réseau est indispensable dès que la VM dispose de plus de 2 vCPU et génère un trafic réseau soutenu supérieur à 2 Gbit/s.

- Toujours mesurer avant et après : sans baseline, il est impossible de valider une optimisation ou de détecter les régressions involontaires.

Checklist d'optimisation rapide

```
# Audit rapide de toutes les VM du nœud
echo "═══ Audit optimisation VM ═══"
for VMID in $(qm list | tail -n +2 | awk '{print $1}'); do
  echo ""
  echo "--- VM $VMID ---"
  qm config $VMID | grep -E "^cpu:|^numa|^balloon|cache=|^net[0-9]"
done

echo ""
echo "═══ Paramètres système ═══"
echo "swappiness      : $(cat /proc/sys/vm/swappiness)  (cible: 10)"
echo "KSM run         : $(cat /sys/kernel/mm/ksm/run)    (1=actif)"
echo "THP             : $(cat /sys/kernel/mm/transparent_hugepage/enabled)"
echo "Hugepages 2M    : $(cat /proc/sys/vm/nr_hugepages) pages réservées"
```

✔ CONSEIL

Priorisez les optimisations par impact descendant : (1) scheduler I/O + mode cache disque, (2) type CPU + NUMA pinning, (3) hugepages + multiqueue réseau, (4) KSM + ballooning. Les deux premières catégories représentent 80 % des gains observables avec 20 % de l'effort de configuration. Commencez toujours par mesurer avec `fio`, `iperf3` et `numastat` avant d'appliquer le moindre changement.

PARTIE VI

Automatisation et gestion multi-site

12 Monitoring et API

Ce chapitre couvre deux piliers fondamentaux de l'exploitation d'une infrastructure Proxmox VE 9 en production : l'observabilité complète de la plateforme et l'automatisation via l'API REST. Nous partons du monitoring natif intégré pour construire progressivement une stack d'observabilité industrielle avec Prometheus et Grafana, puis nous explorons en profondeur l'API REST Proxmox pour automatiser toutes les tâches d'administration.

Le cas fil rouge TechFlow SAS illustre chaque concept avec des configurations réelles, des scripts opérationnels et des dashboards prêts pour la production.

12.1 Monitoring natif Proxmox

Proxmox VE intègre nativement des mécanismes de monitoring qui permettent de surveiller l'infrastructure sans installation d'outils tiers. Ces fonctionnalités constituent le point d'entrée pour tout administrateur, même si elles seront complétées par des solutions plus avancées pour un environnement de production exigeant.

12.1.1 Interface web : graphes RRD

L'interface web de Proxmox VE affiche des graphes temps réel et historiques basés sur **RRDtool** (Round Robin Database tool). Cette technologie stocke des données de métriques dans des bases de données circulaires à taille fixe, ce qui garantit un espace disque constant quelle que soit la durée de rétention.

Accès aux graphes

Les graphes RRD sont accessibles depuis plusieurs niveaux de l'interface :

- **Niveau nœud** : `Datacenter > pve-node1 > Summary` affiche CPU, mémoire, réseau et stockage du nœud hôte
- **Niveau VM/CT** : `pve-node1 > VM 100 > Summary` montre les métriques de la machine virtuelle ou du conteneur
- **Niveau stockage** : `Datacenter > Storage > local-lvm > Summary` présente les graphes d'utilisation du volume

Chaque vue propose plusieurs horizons temporels :

Horizon	Résolution RRD	Rétention
Hour (1h)	30 secondes	70 points
Day (24h)	5 minutes	288 points
Week (7j)	30 minutes	336 points
Month (30j)	2 heures	360 points
Year (365j)	12 heures	730 points

Emplacement des fichiers RRD

Les bases de données RRD sont stockées dans `/var/lib/rrdcached/db/` :

```
# Lister les fichiers RRD d'un nœud
ls -lh /var/lib/rrdcached/db/pve2-node/pve-node1/

# Lister les RRD des VM
ls -lh /var/lib/rrdcached/db/pve2-vm/

# Taille totale occupée par les RRD
du -sh /var/lib/rrdcached/db/
```

Service rrdcached

Le daemon `rrdcached` optimise les écritures en les mettant en cache avant de les envoyer aux fichiers RRD :

```
# Vérifier le statut du service
systemctl status rrdcached

# Voir les statistiques du cache
echo STATS | socat - /var/run/rrdcached.sock

# Forcer le flush des données en mémoire
echo FLUSHALL | socat - /var/run/rrdcached.sock
```

Mise à jour des données RRD

Proxmox met à jour les données RRD via le service `pvedaemon` toutes les 3 secondes pour les métriques temps réel. La fréquence peut être observée dans les logs :

```
# Observer les mises à jour RRD en temps réel
journalctl -u pvedaemon -f | grep -i rrd
```

12.1.2 Métriques disponibles (CPU, RAM, IO, réseau)

Proxmox collecte un ensemble riche de métriques pour chaque type d'objet supervisé. Voici le détail exhaustif des métriques disponibles.

Métriques des nœuds hôtes

Métrique	Description	Unité
`cpu`	Utilisation CPU globale	% (0.0 à 1.0)
`iowait`	Temps d'attente I/O	%
`memtotal`	Mémoire RAM totale	octets
`memused`	Mémoire RAM utilisée	octets
`memfree`	Mémoire RAM libre	octets
`swaptotal`	Swap total	octets
`swapused`	Swap utilisé	octets
`netin`	Trafic réseau entrant	octets/s
`netout`	Trafic réseau sortant	octets/s
`roottotal`	Espace disque root total	octets
`rootused`	Espace disque root utilisé	octets
`loadavg`	Charge moyenne (1, 5, 15 min)	valeur
`cpus`	Nombre de cœurs CPU	entier

Métriques des machines virtuelles (KVM)

Métrique	Description	Unité
`cpu`	Utilisation CPU de la VM	% (0.0 à 1.0)
`cpus`	Nombre de vCPU alloués	entier
`mem`	Mémoire utilisée par la VM	octets
`maxmem`	Mémoire maximale allouée	octets
`disk`	Utilisation disque (legacy)	octets
`diskread`	Lectures disque	octets/s
`diskwrite`	Écritures disque	octets/s
`netin`	Trafic réseau entrant	octets/s
`netout`	Trafic réseau sortant	octets/s
`uptime`	Durée de fonctionnement	secondes

Métrique	Description	Unité
`status`	État (running/stopped)	string

Métriques des conteneurs LXC

Métrique	Description	Unité
`cpu`	Utilisation CPU du CT	%
`mem`	Mémoire utilisée	octets
`maxmem`	Mémoire limite	octets
`swap`	Swap utilisé	octets
`maxswap`	Swap limite	octets
`disk`	Espace disque utilisé	octets
`maxdisk`	Espace disque alloué	octets
`diskread`	Lectures disque	octets/s
`diskwrite`	Écritures disque	octets/s
`netin`	Réseau entrant	octets/s
`netout`	Réseau sortant	octets/s

Accès aux métriques via l'API

Ces métriques sont directement accessibles via l'API REST ou `pvesh` :

```
# Métriques actuelles d'un nœud
pvesh get /nodes/pve-node1/status

# Statistiques RRD d'un nœud (dernière heure)
pvesh get /nodes/pve-node1/rrddata \
  --timeframe hour \
  --cf AVERAGE

# Métriques actuelles d'une VM
pvesh get /nodes/pve-node1/qemu/100/status/current

# Statistiques RRD d'une VM
pvesh get /nodes/pve-node1/qemu/100/rrddata \
  --timeframe day \
  --cf MAX
```

12.1.3 Export vers InfluxDB et Graphite

Proxmox VE supporte nativement l'envoi de métriques vers des systèmes externes via le protocole **Metric Server**. Cette fonctionnalité permet d'agréger les données Proxmox dans une base de données de séries temporelles dédiée.

Protocoles supportés

- **InfluxDB** : via HTTP/HTTPS (InfluxDB 1.x et 2.x)
- **Graphite** : via TCP (format plaintext Graphite)

Configuration via l'interface web

Accédez à `Datacenter > Metric Server > Add` pour configurer un serveur de métriques.

Configuration via la CLI

```
# Ajouter un serveur InfluxDB 2.x
pvesh create /cluster/metrics/server/influxdb-prod \
  --type influxdb \
  --server 10.10.0.25 \
  --port 8086 \
  --influxdbproto https \
  --organization techflow \
  --bucket proxmox-metrics \
  --token "votre-token-influxdb-ici" \
```

```
  --disable 0

# Vérifier la configuration
pvesh get /cluster/metrics/server

# Supprimer un serveur de métriques
pvesh delete /cluster/metrics/server/influxdb-prod
```

Configuration Graphite

```
# Ajouter un serveur Graphite
pvesh create /cluster/metrics/server/graphite-prod \
  --type graphite \
  --server 10.10.0.30 \
  --port 2003 \
  --path proxmox \
  --disable 0
```

Le fichier de configuration résultant est stocké dans `/etc/pve/status.cfg` :

```
# Afficher la configuration actuelle
cat /etc/pve/status.cfg
```

Exemple de contenu du fichier `/etc/pve/status.cfg` :

```
influxdb: influxdb-prod
        server 10.10.0.25
        port 8086
        influxdbproto https
        organization techflow
        bucket proxmox-metrics
        token votre-token-influxdb-ici
```

Format des métriques envoyées

Pour InfluxDB 2.x, les métriques sont envoyées au format **Line Protocol** :

```
pve_node_cpu,host=pve-node1 value=0.15 1699876543000000000
pve_node_memory_used,host=pve-node1 value=8589934592 1699876543000000000
pve_vm_cpu,vmid=100,host=pve-node1 value=0.05 1699876543000000000
```

Intervalle d'envoi

Par défaut, Proxmox envoie les métriques toutes les **30 secondes**. Ce paramètre n'est pas modifiable via l'interface standard mais peut être ajusté dans le code source Perl de `pvedaemon`.

12.1.4 Alertes email intégrées

Proxmox intègre un système d'alertes email basé sur Postfix. Ces alertes couvrent les événements critiques de la plateforme sans nécessiter d'outil tiers.

Événements déclenchant des alertes

- Échec d'une sauvegarde (backup job)
- Erreur de réplication de VM/CT
- Dégradation d'un pool de stockage ZFS
- Alerte SMART sur un disque physique
- Défaillance d'un nœud de cluster détectée
- Modifications de configuration importantes

Configuration de l'email dans l'interface

Accédez à `Datacenter > Options > Email from address` pour définir l'adresse expéditrice.

Configuration SMTP via CLI

```
# Configurer Postfix pour relayer via un serveur SMTP externe
cat >> /etc/postfix/main.cf << 'EOF'
relayhost = [smtp.techflow.fr]:587
smtp_use_tls = yes
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

```
EOF

# Créer le fichier de credentials SMTP
cat > /etc/postfix/sasl_passwd << 'EOF'
[smtp.techflow.fr]:587 monitoring@techflow.fr:MotDePasseSMTP
EOF

# Sécuriser et compiler le fichier
chmod 600 /etc/postfix/sasl_passwd
postmap /etc/postfix/sasl_passwd

# Redémarrer Postfix
systemctl restart postfix

# Tester l'envoi d'email
echo "Test monitoring Proxmox TechFlow" | mail -s "Test alerte" admin@techflow.fr
```

Configurer l'email de notification dans Proxmox

```
# Définir l'email de l'administrateur root
pvesh set /access/users/root@pam \
  --email admin@techflow.fr

# Vérifier
pvesh get /access/users/root@pam
```

Notification pour les jobs de sauvegarde

Dans la configuration des backup jobs (`Datacenter > Backup`), activez les notifications :

- `Notify` : Always / On failure only
- `Email` : admin@techflow.fr

```
# Créer un job de backup avec notification
pvesh create /cluster/backup \
  --enabled 1 \
  --node pve-node1 \
  --storage backup-nfs \
  --compress zstd \
  --mode snapshot \
  --mailnotification always \
  --mailto admin@techflow.fr \
  --starttime 02:00 \
  --dow 1,2,3,4,5,6,7
```

12.2 Stack Prometheus + Grafana

Le monitoring natif de Proxmox est utile pour une vue rapide, mais une infrastructure de production nécessite une stack d'observabilité complète avec rétention longue durée, alerting avancé et dashboards personnalisés. Prometheus et Grafana constituent le standard de l'industrie pour répondre à ces besoins.

12.2.1 Architecture de monitoring

L'architecture déployée pour TechFlow SAS repose sur les composants suivants :

Composants de la stack

Composant	Rôle	Port	Hôte
Prometheus	Collecte et stockage des métriques	9090	10.10.0.20
Grafana	Visualisation et dashboards	3000	10.10.0.20
Alertmanager	Gestion et routage des alertes	9093	10.10.0.20
pve-exporter	Export métriques Proxmox	9221	10.10.0.20
node_exporter	Métriques système des nœuds	9100	Tous les nœuds

Flux de données

```
Proxmox API ──► pve-exporter:9221 ──► Prometheus:9090 ──► Grafana:3000
nœuds hôtes ──► node_exporter:9100 ──► Prometheus:9090 ──► Alertmanager:9093
```

Prérequis infrastructure

```
# Sur le serveur de monitoring (VM Debian 12 sur 10.10.0.20)
# Mise à jour du système
apt update && apt upgrade -y

# Installation des dépendances
apt install -y curl wget tar python3 python3-pip python3-venv \
               adduser libfontconfig1 musl

# Créer les utilisateurs systèmes dédiés
useradd --no-create-home --shell /bin/false prometheus
useradd --no-create-home --shell /bin/false node_exporter

# Créer les répertoires
mkdir -p /etc/prometheus /var/lib/prometheus
chown prometheus:prometheus /etc/prometheus /var/lib/prometheus
```

12.2.2 pve-exporter : installation et configuration

`prometheus-pve-exporter` est un exporter Python qui interroge l'API REST de Proxmox VE et expose les métriques au format Prometheus.

Installation

```
# Sur le serveur de monitoring 10.10.0.20
# Créer un environnement virtuel Python
python3 -m venv /opt/pve-exporter
source /opt/pve-exporter/bin/activate

# Installer pve-exporter
pip install prometheus-pve-exporter

# Vérifier l'installation
pve_exporter --version
deactivate
```

Créer un utilisateur API dédié dans Proxmox

```
# Sur un nœud Proxmox
# Créer le rôle de monitoring avec permissions minimales
pvesh create /access/roles/MonitoringRole \
  --privs "VM.Audit,Pool.Audit,Datastore.Audit,Sys.Audit"

# Créer l'utilisateur de monitoring
pvesh create /access/users/monitoring@pve \
  --comment "Compte monitoring Prometheus" \
  --password "MotDePasseSecurise2024!"

# Assigner le rôle au niveau datacenter (lecture seule)
pvesh create /access/acl \
  --path / \
  --users monitoring@pve \
  --roles MonitoringRole
```

> ✔ **CONSEIL**
> Utilisez un API token plutôt qu'un mot de passe pour le compte de monitoring. Les tokens sont révocables individuellement et peuvent avoir des permissions plus granulaires que le compte parent.

Créer un API token pour pve-exporter

```
# Créer le token (sur un nœud Proxmox)
pvesh create /access/users/monitoring@pve/token/pve-exporter \
  --privsep 1 \
  --comment "Token pour prometheus-pve-exporter"
```

```
# NOTER la valeur du token affichée — elle ne sera plus visible ensuite
# Exemple de sortie : monitoring@pve!pve-exporter=a1b2c3d4-e5f6- ...
```

Configuration de pve-exporter

```
# Créer le répertoire de configuration
mkdir -p /etc/pve-exporter

# Créer le fichier de configuration
cat > /etc/pve-exporter/pve.yml << 'EOF'
default:
  user: monitoring@pve
  token_name: pve-exporter
  token_value: a1b2c3d4-e5f6-7890-abcd-ef1234567890
  verify_ssl: false
EOF

# Sécuriser le fichier (contient des credentials)
chmod 600 /etc/pve-exporter/pve.yml
chown root:root /etc/pve-exporter/pve.yml
```

Service systemd pour pve-exporter

```
cat > /etc/systemd/system/pve-exporter.service << 'EOF'
[Unit]
Description=Prometheus Proxmox VE Exporter
Documentation=https://github.com/prometheus-pve/prometheus-pve-exporter
After=network-online.target
Wants=network-online.target

[Service]
User=root
Group=root
Type=simple
ExecStart=/opt/pve-exporter/bin/pve_exporter \
    --config.file /etc/pve-exporter/pve.yml \
    --web.listen-address 0.0.0.0:9221
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
EOF

# Activer et démarrer le service
systemctl daemon-reload
systemctl enable --now pve-exporter
systemctl status pve-exporter
```

Vérification de pve-exporter

```
# Tester les métriques exposées
curl -s "http://10.10.0.20:9221/pve" | head -50

# Tester avec un hôte Proxmox spécifique
curl -s "http://10.10.0.20:9221/pve?target=10.10.0.11" | grep -E "^pve_"

# Lister les métriques disponibles
curl -s "http://10.10.0.20:9221/pve" | grep "^# HELP" | sort
```

Métriques exposées par pve-exporter

Métrique	Description
`pve_up`	État de connexion à l'API Proxmox
`pve_node_cpu_ratio`	Utilisation CPU du nœud
`pve_node_memory_used_bytes`	Mémoire utilisée du nœud
`pve_node_memory_total_bytes`	Mémoire totale du nœud

Métrique	Description
`pve_node_disk_usage_bytes`	Utilisation disque du nœud
`pve_guest_cpu_ratio`	Utilisation CPU des VM/CT
`pve_guest_memory_used_bytes`	Mémoire utilisée VM/CT
`pve_storage_used_bytes`	Stockage utilisé
`pve_storage_total_bytes`	Stockage total
`pve_cluster_quorum_votes`	Votes quorum du cluster
`pve_version_info`	Version de Proxmox

12.2.3 node_exporter sur les nœuds

`node_exporter` expose les métriques système bas niveau (CPU, mémoire, disques, réseau, filesystem) de chaque nœud Proxmox.

Installation sur tous les nœuds Proxmox

```
# Script d'installation à exécuter sur chaque nœud Proxmox
# (pve-node1, pve-node2, pve-node3)

NODE_EXPORTER_VERSION="1.7.0"
ARCH="linux-amd64"

# Télécharger node_exporter
cd /tmp
wget -q "https://github.com/prometheus/node_exporter/releases/download/v$
{NODE_EXPORTER_VERSION}/node_exporter-${NODE_EXPORTER_VERSION}.${ARCH}.tar.gz"

# Extraire et installer
tar xzf "node_exporter-${NODE_EXPORTER_VERSION}.${ARCH}.tar.gz"
cp "node_exporter-${NODE_EXPORTER_VERSION}.${ARCH}/node_exporter" /usr/local/bin/
chmod +x /usr/local/bin/node_exporter

# Créer l'utilisateur système
useradd --no-create-home --shell /bin/false node_exporter

# Créer le service systemd
cat > /etc/systemd/system/node_exporter.service << 'EOF'
[Unit]
Description=Prometheus Node Exporter
Documentation=https://github.com/prometheus/node_exporter
After=network-online.target
Wants=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter \
    --web.listen-address=0.0.0.0:9100 \
    --collector.systemd \
    --collector.processes \
    --collector.interrupts \
    --collector.buddyinfo \
    --no-collector.wifi
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
EOF

systemctl daemon-reload
```

```
systemctl enable --now node_exporter
systemctl status node_exporter
```

Vérification sur chaque nœud

```
# Depuis le serveur de monitoring
for node in 10.10.0.11 10.10.0.12 10.10.0.13; do
    echo "═══ Node $node ═══"
    curl -s --connect-timeout 3 "http://${node}:9100/metrics" | \
        grep "^node_uname_info" || echo "ERREUR: nœud inaccessible"
done
```

Règles firewall sur les nœuds Proxmox

```
# Ouvrir le port 9100 pour le serveur de monitoring
# Dans le pare-feu Proxmox (via GUI : Datacenter > Firewall)
# Ou via iptables directement sur chaque nœud :

iptables -A INPUT -s 10.10.0.20 -p tcp --dport 9100 -j ACCEPT
iptables -A INPUT -s 10.10.0.20 -p tcp --dport 9221 -j ACCEPT

# Persister les règles
apt install -y iptables-persistent
netfilter-persistent save
```

12.2.4 Prometheus : scrape config

Installation de Prometheus

```
# Sur le serveur de monitoring 10.10.0.20
PROMETHEUS_VERSION="2.48.1"
ARCH="linux-amd64"

cd /tmp
wget -q "https://github.com/prometheus/prometheus/releases/download/v$
{PROMETHEUS_VERSION}/prometheus-${PROMETHEUS_VERSION}.${ARCH}.tar.gz"
tar xzf "prometheus-${PROMETHEUS_VERSION}.${ARCH}.tar.gz"

# Installer les binaires
cp "prometheus-${PROMETHEUS_VERSION}.${ARCH}/prometheus" /usr/local/bin/
cp "prometheus-${PROMETHEUS_VERSION}.${ARCH}/promtool" /usr/local/bin/

# Copier les assets web
cp -r "prometheus-${PROMETHEUS_VERSION}.${ARCH}/consoles" /etc/prometheus/
cp -r "prometheus-${PROMETHEUS_VERSION}.${ARCH}/console_libraries" /etc/
prometheus/

chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
```

Configuration principale de Prometheus

```
cat > /etc/prometheus/prometheus.yml << 'EOF'
global:
  scrape_interval: 30s
  evaluation_interval: 30s
  scrape_timeout: 10s
  external_labels:
    cluster: 'techflow-proxmox'
    environment: 'production'

# Fichiers de règles d'alerte
rule_files:
  - "/etc/prometheus/rules/*.yml"

# Configuration Alertmanager
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - 'localhost:9093'
```

```
# Configurations de scrape
scrape_configs:

  # Prometheus lui-même
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  # pve-exporter pour le cluster Proxmox
  - job_name: 'proxmox'
    metrics_path: /pve
    params:
      cluster: ['1']
      node: ['1']
    static_configs:
      - targets:
          - '10.10.0.11'   # pve-node1
          - '10.10.0.12'   # pve-node2
          - '10.10.0.13'   # pve-node3
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: '10.10.0.20:9221'  # pve-exporter

  # node_exporter sur les nœuds Proxmox
  - job_name: 'node-proxmox'
    static_configs:
      - targets:
          - '10.10.0.11:9100'
          - '10.10.0.12:9100'
          - '10.10.0.13:9100'
        labels:
          role: 'proxmox-node'
    relabel_configs:
      - source_labels: [__address__]
        regex: '([^:]+):\d+'
        target_label: hostname
        replacement: '${1}'

  # Alertmanager
  - job_name: 'alertmanager'
    static_configs:
      - targets: ['localhost:9093']

EOF

# Valider la configuration
promtool check config /etc/prometheus/prometheus.yml
```

Service systemd pour Prometheus

```
cat > /etc/systemd/system/prometheus.service << 'EOF'
[Unit]
Description=Prometheus Monitoring System
Documentation=https://prometheus.io/docs/
After=network-online.target
Wants=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
    --config.file=/etc/prometheus/prometheus.yml \
    --storage.tsdb.path=/var/lib/prometheus/ \
```

```
    --storage.tsdb.retention.time=90d \
    --storage.tsdb.retention.size=50GB \
    --web.console.libraries=/etc/prometheus/console_libraries \
    --web.console.templates=/etc/prometheus/consoles \
    --web.listen-address=0.0.0.0:9090 \
    --web.enable-lifecycle \
    --web.enable-admin-api
Restart=on-failure
RestartSec=5s
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF

systemctl daemon-reload
systemctl enable --now prometheus
```

Vérification de Prometheus

```
# Vérifier que Prometheus scrappe correctement
curl -s http://10.10.0.20:9090/api/v1/targets | \
  python3 -m json.tool | grep -E '"health"|"job"'

# Requête PromQL de test
curl -s 'http://10.10.0.20:9090/api/v1/query?query=pve_up' | \
  python3 -m json.tool

# Vérifier les targets actives
curl -s http://10.10.0.20:9090/api/v1/targets | \
  python3 -c "
import sys, json
data = json.load(sys.stdin)
for t in data['data']['activeTargets']:
    print(f\"{t['labels']['job']} | {t['labels'].get('instance','N/A')} |
{t['health']}\")
"
```

12.2.5 Dashboards Grafana pour Proxmox

Installation de Grafana

```
# Sur le serveur de monitoring 10.10.0.20
# Ajouter le dépôt officiel Grafana
apt install -y apt-transport-https software-properties-common

wget -q -O /usr/share/keyrings/grafana.key \
  https://apt.grafana.com/gpg.key

echo "deb [signed-by=/usr/share/keyrings/grafana.key] \
  https://apt.grafana.com stable main" | \
  tee /etc/apt/sources.list.d/grafana.list

apt update
apt install -y grafana

# Configuration de base
cat > /etc/grafana/grafana.ini << 'EOF'
[server]
http_addr = 0.0.0.0
http_port = 3000
domain = monitoring.techflow.fr
root_url = http://monitoring.techflow.fr:3000/

[security]
admin_user = admin
admin_password = TechFlow2024!
secret_key = sw2YcwTIb9zpOOhoPsMm
```

```
[users]
allow_sign_up = false
default_theme = dark

[analytics]
reporting_enabled = false
check_for_updates = false

[log]
mode = file
level = warn

[alerting]
enabled = true
EOF

systemctl enable --now grafana-server
```

Ajouter Prometheus comme datasource via l'API Grafana

```
# Attendre que Grafana démarre (environ 10 secondes)
sleep 10

# Configurer la datasource Prometheus
curl -s -X POST \
  -H "Content-Type: application/json" \
  -u admin:TechFlow2024! \
  http://10.10.0.20:3000/api/datasources \
  -d '{
    "name": "Prometheus",
    "type": "prometheus",
    "url": "http://localhost:9090",
    "access": "proxy",
    "isDefault": true,
    "jsonData": {
      "timeInterval": "30s",
      "queryTimeout": "60s",
      "httpMethod": "POST"
    }
  }'
```

Importer des dashboards communautaires

Les dashboards Grafana pour Proxmox les plus utilisés sont disponibles sur grafana.com :

Dashboard	ID Grafana	Description
Proxmox VE Overview	10347	Vue générale cluster
Proxmox Nodes	15356	Détail par nœud
Node Exporter Full	1860	Métriques système complètes
Proxmox VE Cluster	21	Métriques cluster PVE

```
# Fonction d'import de dashboard via l'API Grafana
import_dashboard() {
    local dashboard_id=$1
    echo "Importation du dashboard ${dashboard_id} ... "

    curl -s -X POST \
      -H "Content-Type: application/json" \
      -u admin:TechFlow2024! \
      http://10.10.0.20:3000/api/dashboards/import \
      -d "{
        \"inputs\": [{
          \"name\": \"DS_PROMETHEUS\",
          \"pluginId\": \"prometheus\",
          \"type\": \"datasource\",
          \"value\": \"Prometheus\"
        }],
```

```
        \"overwrite\": true,
        \"folderId\": 0
      }"
}

import_dashboard 10347
import_dashboard 1860
```

12.3 Alerting

Un système d'alerting efficace est indissociable du monitoring. Alertmanager, le composant d'alerte de l'écosystème Prometheus, gère le routage, le regroupement et l'envoi des notifications.

12.3.1 Alertmanager : configuration

Installation d'Alertmanager

```
ALERTMANAGER_VERSION="0.26.0"
ARCH="linux-amd64"

cd /tmp
wget -q "https://github.com/prometheus/alertmanager/releases/download/v${ALERTMANAGER_VERSION}/alertmanager-${ALERTMANAGER_VERSION}.${ARCH}.tar.gz"
tar xzf "alertmanager-${ALERTMANAGER_VERSION}.${ARCH}.tar.gz"

cp "alertmanager-${ALERTMANAGER_VERSION}.${ARCH}/alertmanager" /usr/local/bin/
cp "alertmanager-${ALERTMANAGER_VERSION}.${ARCH}/amtool" /usr/local/bin/

useradd --no-create-home --shell /bin/false alertmanager
mkdir -p /etc/alertmanager /var/lib/alertmanager
chown alertmanager:alertmanager /etc/alertmanager /var/lib/alertmanager
```

Configuration principale d'Alertmanager

```
cat > /etc/alertmanager/alertmanager.yml << 'EOF'
global:
  smtp_smarthost: 'smtp.techflow.fr:587'
  smtp_from: 'alerting@techflow.fr'
  smtp_auth_username: 'alerting@techflow.fr'
  smtp_auth_password: 'MotDePasseSMTP'
  smtp_require_tls: true
  resolve_timeout: 5m

# Templates de notification
templates:
  - '/etc/alertmanager/templates/*.tmpl'

# Arbre de routage
route:
  # Route par défaut
  receiver: 'team-ops-email'
  group_by: ['alertname', 'cluster', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 4h

  routes:
    # Alertes critiques : email + Slack + PagerDuty
    - match:
        severity: critical
      receiver: 'critical-multi-channel'
      group_wait: 10s
      repeat_interval: 1h

    # Alertes warning : email + Slack uniquement
    - match:
        severity: warning
      receiver: 'warning-slack-email'
      repeat_interval: 2h
```

```
    # Alertes stockage : équipe stockage
    - match:
        team: storage
      receiver: 'storage-team'

    # Alertes cluster : astreinte
    - match:
        alertname: ClusterQuorumLost
      receiver: 'critical-multi-channel'
      group_wait: 0s
      repeat_interval: 15m

# Définition des receivers
receivers:
  - name: 'team-ops-email'
    email_configs:
      - to: 'ops-team@techflow.fr'
        subject: '[{{ .Status | toUpper }}] {{ .GroupLabels.alertname }}'
        html: '{{ template "email.html" . }}'
        send_resolved: true

  - name: 'critical-multi-channel'
    email_configs:
      - to: 'oncall@techflow.fr'
        subject: '[CRITICAL] {{ .GroupLabels.alertname }} - Action requise'
        send_resolved: true
    slack_configs:
      - api_url: 'https://hooks.slack.com/services/XXXXX/YYYYY/ZZZZZ'
        channel: '#alertes-critiques'
        title: '{{ template "slack.title" . }}'
        text: '{{ template "slack.text" . }}'
        send_resolved: true
        color: '{{ if eq .Status "firing" }}danger{{ else }}good{{ end }}'
    pagerduty_configs:
      - service_key: 'votre-pagerduty-service-key'
        send_resolved: true

  - name: 'warning-slack-email'
    slack_configs:
      - api_url: 'https://hooks.slack.com/services/XXXXX/YYYYY/ZZZZZ'
        channel: '#alertes-warning'
        title: '[WARNING] {{ .GroupLabels.alertname }}'
        send_resolved: true
    email_configs:
      - to: 'ops-team@techflow.fr'
        send_resolved: true

  - name: 'storage-team'
    email_configs:
      - to: 'storage@techflow.fr'
        subject: '[STORAGE] {{ .GroupLabels.alertname }}'
        send_resolved: true

# Règles d'inhibition
inhibit_rules:
  # Si un nœud est down, inhiber les alertes de ses VM
  - source_match:
      alertname: 'NodeDown'
    target_match_re:
      alertname: 'VM.*'
    equal: ['instance']
EOF

# Service systemd Alertmanager
cat > /etc/systemd/system/alertmanager.service << 'EOF'
[Unit]
Description=Alertmanager
```

```
After=network-online.target
Wants=network-online.target

[Service]
User=alertmanager
Group=alertmanager
Type=simple
ExecStart=/usr/local/bin/alertmanager \
    --config.file=/etc/alertmanager/alertmanager.yml \
    --storage.path=/var/lib/alertmanager/ \
    --web.listen-address=0.0.0.0:9093 \
    --cluster.listen-address=""
Restart=on-failure

[Install]
WantedBy=multi-user.target
EOF

systemctl daemon-reload
systemctl enable --now alertmanager
```

12.3.2 Règles d'alerte critiques (espace disque, RAM, cluster)

Les règles d'alerte sont définies dans des fichiers YAML chargés par Prometheus.

```
mkdir -p /etc/prometheus/rules

cat > /etc/prometheus/rules/proxmox-alerts.yml << 'EOF'
groups:
  - name: proxmox.node
    interval: 30s
    rules:

      # CPU élevé sur nœud Proxmox
      - alert: ProxmoxNodeCPUHigh
        expr: pve_node_cpu_ratio > 0.85
        for: 10m
        labels:
          severity: warning
          team: ops
        annotations:
          summary: "CPU élevé sur {{ $labels.instance }}"
          description: >
            Le nœud {{ $labels.instance }} utilise {{ $value |
humanizePercentage }}
            de CPU depuis plus de 10 minutes.
          runbook: "https://wiki.techflow.fr/runbooks/cpu-high"

      # CPU critique sur nœud Proxmox
      - alert: ProxmoxNodeCPUCritical
        expr: pve_node_cpu_ratio > 0.95
        for: 5m
        labels:
          severity: critical
          team: ops
        annotations:
          summary: "CPU CRITIQUE sur {{ $labels.instance }}"
          description: >
            Le nœud {{ $labels.instance }} utilise {{ $value |
humanizePercentage }}
            de CPU. Risque de dégradation de service.

      # Mémoire élevée sur nœud Proxmox
      - alert: ProxmoxNodeMemoryHigh
        expr: >
          (pve_node_memory_used_bytes / pve_node_memory_total_bytes) > 0.85
        for: 15m
        labels:
          severity: warning
```

```
        team: ops
      annotations:
        summary: "Mémoire élevée sur {{ $labels.instance }}"
        description: >
          Le nœud {{ $labels.instance }} utilise
          {{ $value | humanizePercentage }} de sa mémoire RAM.

    # Mémoire critique sur nœud Proxmox
    - alert: ProxmoxNodeMemoryCritical
      expr: >
        (pve_node_memory_used_bytes / pve_node_memory_total_bytes) > 0.95
      for: 5m
      labels:
        severity: critical
        team: ops
      annotations:
        summary: "Mémoire CRITIQUE sur {{ $labels.instance }}"
        description: >
          Risque de swap ou OOM sur le nœud {{ $labels.instance }}.
          Mémoire utilisée : {{ $value | humanizePercentage }}.

- name: proxmox.storage
  interval: 60s
  rules:

    # Espace disque élevé
    - alert: ProxmoxStorageHigh
      expr: >
        (pve_storage_used_bytes / pve_storage_total_bytes) > 0.80
      for: 5m
      labels:
        severity: warning
        team: storage
      annotations:
        summary: "Stockage {{ $labels.id }} presque plein"
        description: >
          Le stockage {{ $labels.id }} sur {{ $labels.instance }}
          est utilisé à {{ $value | humanizePercentage }}.

    # Espace disque critique
    - alert: ProxmoxStorageCritical
      expr: >
        (pve_storage_used_bytes / pve_storage_total_bytes) > 0.90
      for: 2m
      labels:
        severity: critical
        team: storage
      annotations:
        summary: "STOCKAGE CRITIQUE : {{ $labels.id }}"
        description: >
          Le stockage {{ $labels.id }} est utilisé à
          {{ $value | humanizePercentage }}. Intervention immédiate requise.

    # Stockage presque plein (95%)
    - alert: ProxmoxStorageFull
      expr: >
        (pve_storage_used_bytes / pve_storage_total_bytes) > 0.95
      for: 1m
      labels:
        severity: critical
        team: storage
      annotations:
        summary: "STOCKAGE PLEIN : {{ $labels.id }}"
        description: >
          URGENCE : Le stockage {{ $labels.id }} est à
          {{ $value | humanizePercentage }}. Risque de perte de données.

- name: proxmox.cluster
```

```
    interval: 30s
    rules:

      # Perte de quorum cluster
      - alert: ClusterQuorumLost
        expr: pve_cluster_quorum_votes < 2
        for: 0m
        labels:
          severity: critical
          team: ops
        annotations:
          summary: "QUORUM PERDU dans le cluster Proxmox"
          description: >
            Le cluster Proxmox n'a plus le quorum.
            Votes actuels : {{ $value }}.
            Aucune opération d'écriture n'est possible.

      # Nœud hors ligne
      - alert: ProxmoxNodeDown
        expr: pve_up == 0
        for: 2m
        labels:
          severity: critical
          team: ops
        annotations:
          summary: "Nœud Proxmox {{ $labels.instance }} HORS LIGNE"
          description: >
            Le nœud Proxmox {{ $labels.instance }} ne répond plus
            à l'API depuis 2 minutes.

      # Nœud inaccessible (node_exporter)
      - alert: NodeExporterDown
        expr: up{job="node-proxmox"} == 0
        for: 3m
        labels:
          severity: warning
          team: ops
        annotations:
          summary: "node_exporter inaccessible sur {{ $labels.instance }}"
          description: >
            Le node_exporter du nœud {{ $labels.instance }} ne répond plus.

  - name: proxmox.vms
    interval: 60s
    rules:

      # VM avec CPU très élevé
      - alert: VMCPUHigh
        expr: pve_guest_cpu_ratio > 0.90
        for: 20m
        labels:
          severity: warning
          team: ops
        annotations:
          summary: "CPU élevé sur VM {{ $labels.vmid }}"
          description: >
            La VM {{ $labels.vmid }} ({{ $labels.name }}) utilise
            {{ $value | humanizePercentage }} de CPU depuis 20 minutes.

      # VM avec mémoire presque pleine
      - alert: VMMemoryHigh
        expr: >
          (pve_guest_memory_used_bytes / pve_guest_memory_total_bytes) > 0.90
        for: 10m
        labels:
          severity: warning
          team: ops
        annotations:
```

```
          summary: "Mémoire élevée sur VM {{ $labels.vmid }}"
          description: >
            La VM {{ $labels.vmid }} utilise {{ $value | humanizePercentage }}
            de sa mémoire allouée.

EOF

# Valider les règles
promtool check rules /etc/prometheus/rules/proxmox-alerts.yml

# Recharger Prometheus sans redémarrage
curl -s -X POST http://10.10.0.20:9090/-/reload
```

12.3.3 Notifications (email, Slack, PagerDuty)

Templates de notification personnalisés

```
mkdir -p /etc/alertmanager/templates

cat > /etc/alertmanager/templates/techflow.tmpl << 'EOF'
{{ define "email.html" }}
<!DOCTYPE html>
<html>
<head>
<style>
  body { font-family: Arial, sans-serif; }
  .alert-firing { background-color: #ff4444; color: white; padding: 10px; }
  .alert-resolved { background-color: #44bb44; color: white; padding: 10px; }
  table { border-collapse: collapse; width: 100%; }
  th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
</style>
</head>
<body>
  <div class="{{ if eq .Status "firing" }}alert-firing{{ else }}alert-
resolved{{ end }}">
    <h2>{{ if eq .Status "firing" }}ALERTE{{ else }}RESOLU{{ end }}:
{{ .GroupLabels.alertname }}</h2>
    <p>Cluster: {{ .CommonLabels.cluster }} | Environnement:
{{ .CommonLabels.environment }}</p>
  </div>

  <table>
    <tr><th>Alerte</th><th>Sévérité</th><th>Instance</th><th>Description</
th><th>Début</th></tr>
    {{ range .Alerts }}
    <tr>
      <td>{{ .Labels.alertname }}</td>
      <td>{{ .Labels.severity }}</td>
      <td>{{ .Labels.instance }}</td>
      <td>{{ .Annotations.description }}</td>
      <td>{{ .StartsAt.Format "2006-01-02 15:04:05" }}</td>
    </tr>
    {{ end }}
  </table>

  <p><a href="{{ .ExternalURL }}">Voir dans Alertmanager</a></p>
  <p><small>TechFlow SAS - Infrastructure Monitoring</small></p>
</body>
</html>
{{ end }}

{{ define "slack.title" }}
{{ if eq .Status "firing" }}[FIRING]{{ else }}[RESOLVED]{{ end }}
{{ .GroupLabels.alertname }}
{{ end }}

{{ define "slack.text" }}
*Cluster:* {{ .CommonLabels.cluster }}
*Environnement:* {{ .CommonLabels.environment }}
```

```
{{ range .Alerts }}
- *{{ .Labels.alertname }}* sur `{{ .Labels.instance }}`
  Sévérité: {{ .Labels.severity }}
  {{ .Annotations.description }}
{{ end }}

Voir dans Alertmanager: {{ .ExternalURL }}
{{ end }}
EOF

# Valider la configuration Alertmanager
amtool check-config /etc/alertmanager/alertmanager.yml

# Recharger la configuration
curl -s -X POST http://10.10.0.20:9093/-/reload
```

Test des notifications

```
# Envoyer une alerte de test via amtool
amtool alert add \
  --alertmanager.url=http://10.10.0.20:9093 \
  alertname="TestAlerte" \
  severity="warning" \
  instance="pve-node1" \
  cluster="techflow-proxmox" \
  --annotation summary="Test de notification TechFlow" \
  --annotation description="Ceci est un test du système d'alerting."

# Vérifier les alertes en cours
amtool alert query --alertmanager.url=http://10.10.0.20:9093

# Lister les silences actifs
amtool silence query --alertmanager.url=http://10.10.0.20:9093
```

12.3.4 Escalade et silences

Créer un silence de maintenance

```
# Silence pendant une fenêtre de maintenance (2h)
amtool silence add \
  --alertmanager.url=http://10.10.0.20:9093 \
  --author="admin@techflow.fr" \
  --comment="Maintenance ZFS pve-node2 - ticket #4521" \
  --duration=2h \
  instance="10.10.0.12"

# Silence ciblé sur un type d'alerte avec plage horaire
amtool silence add \
  --alertmanager.url=http://10.10.0.20:9093 \
  --author="sre-team@techflow.fr" \
  --comment="Migration stockage en cours" \
  --start="2024-01-15T02:00:00+01:00" \
  --end="2024-01-15T06:00:00+01:00" \
  alertname="ProxmoxStorageHigh" \
  instance="10.10.0.11"

# Lister tous les silences
amtool silence query --alertmanager.url=http://10.10.0.20:9093

# Supprimer un silence par son ID
amtool silence expire --alertmanager.url=http://10.10.0.20:9093 SILENCE_ID
```

⚠ AVERTISSEMENT

Les silences Alertmanager ne désactivent pas la collecte des métriques ni l'évaluation des règles dans Prometheus. Ils empêchent uniquement l'envoi des notifications. Les alertes restent visibles dans l'interface Alertmanager avec le statut "silenced". Documentez systématiquement vos silences avec un numéro de ticket pour la traçabilité.

Règles d'inhibition avancées

Les inhibitions permettent de supprimer des alertes secondaires quand une alerte de niveau supérieur est active. À ajouter dans `alertmanager.yml` :

```
inhibit_rules:
  # Si le cluster a perdu le quorum, inhiber toutes les autres alertes cluster
  - source_match:
      alertname: 'ClusterQuorumLost'
    target_match_re:
      alertname: 'Proxmox.*'
    equal: ['cluster']

  # Si un nœud est down, inhiber les alertes warning de ce nœud
  - source_match:
      alertname: 'ProxmoxNodeDown'
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['instance']

  # Les alertes critiques inhibent les warnings du même service
  - source_match:
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['alertname', 'instance']
```

12.4 L'API REST Proxmox

L'API REST est le cœur de Proxmox VE. Toutes les opérations de l'interface web passent par cette API, ce qui signifie que tout ce qui est faisable via l'interface graphique est également faisable via l'API.

12.4.1 Architecture et authentification

Principes de l'API

L'API Proxmox est une API REST complète accessible sur le port **8006** en HTTPS. Elle suit les conventions REST avec les méthodes HTTP standard :

Méthode HTTP	Opération CRUD	Usage Proxmox
GET	Read	Lire des informations
POST	Create	Créer des ressources, lancer des actions
PUT	Update	Modifier des ressources existantes
DELETE	Delete	Supprimer des ressources

URL de base

```
https://<ip-nœud>:8006/api2/json/<endpoint>
```

Documentation interactive

Proxmox fournit une documentation API interactive accessible via le navigateur :

```
https://10.10.0.11:8006/api2/json
```

Cette interface Swagger-like permet d'explorer tous les endpoints disponibles et de tester des requêtes directement depuis le navigateur.

Modes d'authentification

Mode	Usage	Durée
Ticket + CSRF token	Sessions interactives	2 heures
API token	Automatisation, scripts	Illimité (révocable)
Two-factor auth	Sécurité renforcée	Compatible avec les deux modes

12.4.2 Tickets et API tokens

Authentification par ticket (session)

```
# Obtenir un ticket d'authentification et le token CSRF en une requête
AUTH_RESPONSE=$(curl -s -k -X POST \
  https://10.10.0.11:8006/api2/json/access/ticket \
  -d 'username=root@pam&password=MotDePasseRootPVE')

TICKET=$(echo "$AUTH_RESPONSE" | python3 -c "import sys,json;
print(json.load(sys.stdin)['data']['ticket'])")
CSRF=$(echo "$AUTH_RESPONSE" | python3 -c "import sys,json;
print(json.load(sys.stdin)['data']['CSRFPreventionToken'])")

# Utiliser le ticket dans les requêtes suivantes
curl -s -k \
  -H "Cookie: PVEAuthCookie=${TICKET}" \
  -H "CSRFPreventionToken: ${CSRF}" \
  https://10.10.0.11:8006/api2/json/nodes
```

Création d'API tokens

```
# Créer un token via pvesh (sur un nœud Proxmox)
# Token avec séparation de privilèges (privsep=1 : le token ne peut pas avoir
# plus de droits que le compte parent)
pvesh create /access/users/root@pam/token/automation \
  --comment "Token pour scripts d'automatisation" \
  --privsep 0

# Pour un utilisateur non-root avec séparation de privilèges
pvesh create /access/users/admin@pve/token/terraform \
  --comment "Token pour Terraform" \
  --privsep 1

# Lister les tokens existants
pvesh get /access/users/root@pam/token

# Supprimer un token
pvesh delete /access/users/root@pam/token/automation
```

> ✔ **CONSEIL**
> Préférez toujours les API tokens aux mots de passe pour l'automatisation. Un token peut être révoqué sans changer le mot de passe du compte, et avec `privsep=1`, il peut avoir des permissions inférieures au compte parent. Stockez les tokens dans un gestionnaire de secrets (HashiCorp Vault, Bitwarden Secrets, etc.) et jamais en clair dans des scripts ou des dépôts Git.

Utilisation d'un API token

```
# Format du token : USER@REALM!TOKENID=SECRET
# Exemple : root@pam!automation=8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c

API_TOKEN="root@pam!automation=8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c"

# Utiliser le token dans les requêtes
curl -s -k \
  -H "Authorization: PVEAPIToken=${API_TOKEN}" \
  https://10.10.0.11:8006/api2/json/nodes

# Variable d'environnement pour les scripts
export PROXMOX_TOKEN="root@pam!automation=8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c"
curl -s -k \
  -H "Authorization: PVEAPIToken=${PROXMOX_TOKEN}" \
  https://10.10.0.11:8006/api2/json/cluster/status
```

12.4.3 Explorer l'API avec pvesh

`pvesh` est l'outil CLI fourni avec Proxmox pour interagir avec l'API sans avoir à gérer l'authentification manuellement. Il est exécuté directement sur les nœuds Proxmox.

Commandes de base pvesh

```
# Syntaxe générale
# pvesh <méthode> <endpoint> [options]

# GET : lire des informations
pvesh get /nodes
pvesh get /nodes/pve-node1/status
pvesh get /nodes/pve-node1/qemu

# POST : créer ou effectuer une action
pvesh create /nodes/pve-node1/qemu/100/status/start

# PUT : modifier une ressource
pvesh set /nodes/pve-node1/qemu/100/config \
  --memory 4096 \
  --cores 4

# DELETE : supprimer une ressource
pvesh delete /nodes/pve-node1/qemu/999

# Afficher en format YAML lisible
pvesh get /nodes/pve-node1/status --output-format yaml

# Afficher en JSON
pvesh get /nodes/pve-node1/status --output-format json

# Utiliser jq pour filtrer
pvesh get /nodes/pve-node1/qemu --output-format json | \
  jq '.[] | {vmid, name, status, mem}'
```

Navigation dans l'API avec pvesh

```
# Lister les sous-resources disponibles d'un endpoint
pvesh ls /nodes/pve-node1

# Exemple de sortie :
# aplinfo
# ceph
# config
# disks
# dns
# firewall
# hardware
# hosts
# lxc
#  ...

# Lister les actions disponibles sur une VM
pvesh ls /nodes/pve-node1/qemu/100

# Aide sur un endpoint spécifique
pvesh usage /nodes/pve-node1/qemu POST
```

Exemples d'exploration

```
# Lister tous les nœuds du cluster
pvesh get /cluster/status --output-format json | \
  jq '.[] | select(.type=="node")'

# Lister toutes les VM de tous les nœuds
pvesh get /cluster/resources --type vm --output-format json | \
  jq '.[] | {vmid, name, node, status, maxmem, maxcpu}'

# Configuration d'une VM
pvesh get /nodes/pve-node1/qemu/100/config

# Logs d'une VM (50 dernières lignes)
pvesh get /nodes/pve-node1/qemu/100/log --limit 50

# Tâches récentes d'un nœud
```

```
pvesh get /nodes/pve-node1/tasks --limit 20 --output-format json | \
  jq '.[] | {upid, type, status, starttime}'
```

12.4.4 Endpoints essentiels (nodes, vms, storage, cluster)

Table des endpoints principaux

Endpoint	Méthode	Description
`/access/ticket`	POST	Authentification, obtenir ticket
`/access/users`	GET/POST	Gestion des utilisateurs
`/access/roles`	GET/POST	Gestion des rôles
`/access/acl`	GET/POST/PUT	Gestion des ACL
`/cluster/status`	GET	État général du cluster
`/cluster/resources`	GET	Toutes les ressources du cluster
`/cluster/tasks`	GET	Tâches cluster en cours
`/cluster/backup`	GET/POST	Configuration des sauvegardes
`/cluster/ha/status`	GET	État de la haute disponibilité
`/nodes`	GET	Liste des nœuds
`/nodes/{node}/status`	GET	État d'un nœud
`/nodes/{node}/qemu`	GET/POST	Liste/création de VM
`/nodes/{node}/qemu/{vmid}/status/start`	POST	Démarrer une VM
`/nodes/{node}/qemu/{vmid}/status/stop`	POST	Arrêter une VM (force)
`/nodes/{node}/qemu/{vmid}/status/shutdown`	POST	Arrêt propre VM
`/nodes/{node}/qemu/{vmid}/config`	GET/PUT/POST	Configuration VM
`/nodes/{node}/qemu/{vmid}/snapshot`	GET/POST	Snapshots VM
`/nodes/{node}/qemu/{vmid}/migrate`	POST	Migration VM
`/nodes/{node}/qemu/{vmid}/clone`	POST	Cloner VM
`/nodes/{node}/lxc`	GET/POST	Liste/création conteneurs
`/nodes/{node}/storage`	GET	Stockages du nœud
`/nodes/{node}/storage/{storage}/content`	GET	Contenu d'un stockage
`/nodes/{node}/network`	GET/PUT	Configuration réseau
`/nodes/{node}/firewall`	GET/POST	Règles pare-feu
`/nodes/{node}/tasks`	GET	Tâches du nœud
`/nodes/{node}/syslog`	GET	Logs système

Exemples de requêtes sur les endpoints essentiels

```
# Définir les variables de configuration
API_BASE="https://10.10.0.11:8006/api2/json"
TOKEN="root@pam!automation=8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c"

# Fonction helper
pve_get() {
  curl -s -k -H "Authorization: PVEAPIToken=${TOKEN}" "${API_BASE}${1}"
}

# État du cluster
pve_get "/cluster/status" | python3 -m json.tool

# Toutes les ressources (VM + CT + Storage + Nodes)
pve_get "/cluster/resources" | \
  python3 -c "
import sys, json
```

```
data = json.load(sys.stdin)['data']
for r in sorted(data, key=lambda x: x.get('type','')):
    print(f\"{r.get('type','?'):10} | {r.get('name','?'):30} |
{r.get('status','?')}\")
"

# Lister les VM avec mémoire utilisée
pvesh get /cluster/resources --type vm --output-format json | \
  jq -r '.[] | [.vmid, .name, .node, .status,
    ((.mem // 0)/1073741824*100|round/100|tostring)+" GB"] | @tsv' | \
  column -t -s $'\t'
```

12.5 Automatisation avec l'API

12.5.1 Python : proxmoxer

`proxmoxer` est la bibliothèque Python de référence pour interagir avec l'API Proxmox. Elle simplifie considérablement les appels API en gérant l'authentification et la sérialisation automatiquement.

Installation

```
pip install proxmoxer requests
# Ou dans un virtualenv
python3 -m venv /opt/proxmox-scripts
source /opt/proxmox-scripts/bin/activate
pip install proxmoxer requests
```

Connexion et opérations de base

```
#!/usr/bin/env python3
"""
Exemple de connexion à Proxmox via proxmoxer
TechFlow SAS - Scripts d'administration
"""

from proxmoxer import ProxmoxAPI

# Connexion avec API token (recommandé)
proxmox = ProxmoxAPI(
    '10.10.0.11',
    user='root@pam',
    token_name='automation',
    token_value='8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c',
    verify_ssl=False
)

# Lister les nœuds du cluster
print("═══ Nœuds du cluster ═══")
for node in proxmox.nodes.get():
    print(f"  - {node['node']}: {node['status']} "
          f"(CPU: {node.get('cpu', 0)*100:.1f}%, "
          f"Mem: {node.get('mem', 0)/1024**3:.1f}/"
          f"{node.get('maxmem', 0)/1024**3:.1f} GB)")

# Lister toutes les VM
print("\n═══ Machines virtuelles ═══")
for vm in proxmox.cluster.resources.get(type='vm'):
    print(f"  VM {vm['vmid']:4d} | {vm.get('name','?'):30s} | "
          f"{vm['node']:12s} | {vm['status']}")

# État détaillé d'une VM
vm_status = proxmox.nodes('pve-node1').qemu(100).status.current.get()
print(f"\n═══ VM 100 ═══")
print(f"  CPU: {vm_status.get('cpu', 0)*100:.1f}%")
print(f"  Mémoire: {vm_status.get('mem', 0)/1024**2:.0f} MB / "
      f"{vm_status.get('maxmem', 0)/1024**2:.0f} MB")
print(f"  Uptime: {vm_status.get('uptime', 0)//3600} heures")
```

Gestion des VM

```
#!/usr/bin/env python3
"""
Fonctions de gestion des VM via proxmoxer - TechFlow SAS
"""

from proxmoxer import ProxmoxAPI
import time

def get_proxmox_connection():
    return ProxmoxAPI(
        '10.10.0.11',
        user='root@pam',
        token_name='automation',
        token_value='8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c',
        verify_ssl=False
    )

def wait_for_task(proxmox, node, task_id, timeout=300):
    """Attendre la fin d'une tâche Proxmox"""
    start = time.time()
    while time.time() - start < timeout:
        status = proxmox.nodes(node).tasks(task_id).status.get()
        if status['status'] == 'stopped':
            if status.get('exitstatus') == 'OK':
                return True
            else:
                raise Exception(f"Tâche échouée: {status.get('exitstatus')}")
        time.sleep(2)
    raise TimeoutError(f"Tâche {task_id} timeout après {timeout}s")

def create_vm_from_template(proxmox, node, template_id, new_vmid,
                            name, memory, cores, storage):
    """Créer une VM par clonage d'un template"""
    print(f"Clonage du template {template_id} vers VM {new_vmid} ({name}) ... ")

    task = proxmox.nodes(node).qemu(template_id).clone.post(
        newid=new_vmid,
        name=name,
        full=1,
        storage=storage
    )
    wait_for_task(proxmox, node, task)
    print(f"  Clone créé. Configuration en cours ... ")

    # Configurer la VM clonée
    proxmox.nodes(node).qemu(new_vmid).config.put(
        memory=memory,
        cores=cores,
        name=name
    )
    print(f"  VM {new_vmid} créée et configurée.")
    return new_vmid

def start_vm(proxmox, node, vmid):
    """Démarrer une VM"""
    print(f"Démarrage de la VM {vmid} ... ")
    task = proxmox.nodes(node).qemu(vmid).status.start.post()
    wait_for_task(proxmox, node, task)
    print(f"  VM {vmid} démarrée.")

def stop_vm_graceful(proxmox, node, vmid, timeout=60):
    """Arrêt propre d'une VM"""
    print(f"Arrêt propre de la VM {vmid} ... ")
    task = proxmox.nodes(node).qemu(vmid).status.shutdown.post(
        timeout=timeout
    )
```

```
    wait_for_task(proxmox, node, task, timeout=timeout+30)
    print(f"  VM {vmid} arrêtée.")

def create_snapshot(proxmox, node, vmid, snapname, description=""):
    """Créer un snapshot d'une VM"""
    task = proxmox.nodes(node).qemu(vmid).snapshot.post(
        snapname=snapname,
        description=description,
        vmstate=0
    )
    wait_for_task(proxmox, node, task)
    print(f"  Snapshot '{snapname}' créé pour VM {vmid}.")
```

12.5.2 Bash : curl + jq

Pour les scripts shell, `curl` et `jq` constituent une combinaison puissante pour interagir avec l'API Proxmox.

Bibliothèque de fonctions Bash

```
#!/usr/bin/env bash
# lib-proxmox.sh — Bibliothèque de fonctions pour l'API Proxmox
# TechFlow SAS

set -euo pipefail

# Configuration
PROXMOX_HOST="${PROXMOX_HOST:-10.10.0.11}"
PROXMOX_PORT="${PROXMOX_PORT:-8006}"
PROXMOX_TOKEN="${PROXMOX_TOKEN:-root@pam!automation=8a3f2c1d-…}"
API_BASE="https://${PROXMOX_HOST}:${PROXMOX_PORT}/api2/json"

# Fonction de requête API générique
pve_api() {
    local method="$1"
    local endpoint="$2"
    shift 2
    local data_args=("$@")

    curl -s -k \
        -X "${method}" \
        -H "Authorization: PVEAPIToken=${PROXMOX_TOKEN}" \
        -H "Content-Type: application/json" \
        "${data_args[@]}" \
        "${API_BASE}${endpoint}"
}

pve_get()    { pve_api GET    "$1"; }
pve_post()   { pve_api POST   "$1" --data "$2"; }
pve_put()    { pve_api PUT    "$1" --data "$2"; }
pve_delete() { pve_api DELETE "$1"; }

# Attendre la fin d'une tâche
pve_wait_task() {
    local node="$1"
    local upid="$2"
    local timeout="${3:-300}"
    local start
    start=$(date +%s)

    while true; do
        local status
        status=$(pve_get "/nodes/${node}/tasks/${upid}/status" | \
                 jq -r '.data.status')

        if [[ "$status" == "stopped" ]]; then
            local exit_status
            exit_status=$(pve_get "/nodes/${node}/tasks/${upid}/status" | \
                          jq -r '.data.exitstatus')
```

```
            [[ "$exit_status" == "OK" ]] || {
                echo "ERREUR: Tâche échouée: $exit_status" >&2
                return 1
            }
            return 0
        fi

        local elapsed=$(( $(date +%s) - start ))
        [[ $elapsed -lt $timeout ]] || {
            echo "TIMEOUT: Tâche $upid" >&2
            return 1
        }
        sleep 2
    done
}

# Créer une VM par clonage
pve_clone_vm() {
    local node="$1"
    local template_id="$2"
    local new_vmid="$3"
    local new_name="$4"

    echo "Clonage VM ${template_id} → ${new_vmid} (${new_name}) ... "
    local upid
    upid=$(pve_post "/nodes/${node}/qemu/${template_id}/clone" \
        "{\"newid\":${new_vmid},\"name\":\"${new_name}\",\"full\":1}" | \
        jq -r '.data')

    pve_wait_task "$node" "$upid"
    echo "  Clonage terminé."
}

# Démarrer une VM
pve_start_vm() {
    local node="$1"
    local vmid="$2"

    echo "Démarrage VM ${vmid} ... "
    local upid
    upid=$(pve_post "/nodes/${node}/qemu/${vmid}/status/start" "{}" | \
           jq -r '.data')
    pve_wait_task "$node" "$upid"
    echo "  VM ${vmid} démarrée."
}
```

⚠ AVERTISSEMENT

Soyez prudent avec le rate limiting de l'API Proxmox. En cas de trop nombreuses requêtes simultanées, l'API peut renvoyer des erreurs HTTP 429 ou 503. Ajoutez des délais entre les requêtes dans vos scripts (au moins 100ms entre chaque appel) et évitez de paralléliser massivement les opérations sur l'API sans mécanisme de throttling.

12.5.3 Terraform provider Proxmox

Terraform avec le provider Proxmox permet de gérer l'infrastructure Proxmox en tant que code (IaC), avec les avantages du versionning, de la reproductibilité et de la collaboration.

Installation du provider

```
# versions.tf
terraform {
  required_version = "≥ 1.6.0"
  required_providers {
    proxmox = {
      source  = "bpg/proxmox"
      version = "~> 0.46"
    }
```

```
  }
}
```

Configuration du provider

```
# provider.tf
provider "proxmox" {
  endpoint  = "https://10.10.0.11:8006/"
  api_token = var.proxmox_api_token
  insecure  = true  # En prod : utiliser un certificat valide

  ssh {
    agent    = true
    username = "root"
  }
}

variable "proxmox_api_token" {
  type        = string
  description = "API token Proxmox (format: user@realm!tokenid=secret)"
  sensitive   = true
}
```

Créer une VM avec Terraform

```
# vm-web.tf
resource "proxmox_virtual_environment_vm" "web_server" {
  name        = "web-prod-01"
  description = "Serveur web production TechFlow"
  node_name   = "pve-node1"
  vm_id       = 200

  tags = ["production", "web", "techflow"]

  clone {
    vm_id   = 9000  # ID du template Debian 12
    full    = true
    retries = 3
  }

  cpu {
    cores   = 4
    sockets = 1
    type    = "host"
  }

  memory {
    dedicated = 4096
    floating  = 2048
  }

  disk {
    datastore_id = "local-lvm"
    size         = 50
    interface    = "virtio0"
    iothread     = true
    discard      = "on"
  }

  network_device {
    bridge  = "vmbr0"
    model   = "virtio"
    vlan_id = 100
  }

  initialization {
    ip_config {
      ipv4 {
        address = "10.10.1.10/24"
```

```
      gateway = "10.10.1.1"
    }
  }
  dns {
    servers = ["10.10.0.1", "8.8.8.8"]
  }
  user_account {
    username = "admin"
    keys     = [file("~/.ssh/id_rsa.pub")]
  }
}

operating_system {
  type = "l26"
}

lifecycle {
  ignore_changes = [initialization]
}
}

output "vm_id" {
  value = proxmox_virtual_environment_vm.web_server.vm_id
}
```

Déploiement Terraform

```
# Fichier terraform.tfvars (NE PAS committer en Git !)
# proxmox_api_token = "root@pam!terraform=a1b2c3d4-e5f6-7890-abcd-ef1234567890"

# Déployer l'infrastructure
cd /opt/terraform/techflow-infra

terraform init
terraform plan -out=tfplan
terraform apply tfplan

# Détruire les ressources
terraform destroy
```

12.5.4 Ansible collection community.proxmox

La collection Ansible `community.general` inclut des modules pour gérer Proxmox VE.

Installation

```
# Installer la collection Ansible pour Proxmox
ansible-galaxy collection install community.general

# Installer les dépendances Python
pip install proxmoxer requests
```

Playbook de gestion des VM

```
---
# proxmox-vm-management.yml
- name: Gestion des VM Proxmox - TechFlow SAS
  hosts: localhost
  gather_facts: false

  vars:
    proxmox_api_host: "10.10.0.11"
    proxmox_api_user: "root@pam"
    proxmox_api_token_id: "ansible"
    proxmox_api_token_secret: "{{ vault_proxmox_token }}"
    proxmox_node: "pve-node1"
    proxmox_validate_certs: false

  tasks:
    - name: Créer une VM depuis un template
```

```
      community.general.proxmox_kvm:
        api_host: "{{ proxmox_api_host }}"
        api_user: "{{ proxmox_api_user }}"
        api_token_id: "{{ proxmox_api_token_id }}"
        api_token_secret: "{{ proxmox_api_token_secret }}"
        validate_certs: "{{ proxmox_validate_certs }}"
        node: "{{ proxmox_node }}"
        vmid: 201
        name: "ansible-test-01"
        clone: 9000
        full: true
        storage: "local-lvm"
        cores: 2
        memory: 2048
        state: present

    - name: Démarrer la VM
      community.general.proxmox_kvm:
        api_host: "{{ proxmox_api_host }}"
        api_user: "{{ proxmox_api_user }}"
        api_token_id: "{{ proxmox_api_token_id }}"
        api_token_secret: "{{ proxmox_api_token_secret }}"
        validate_certs: "{{ proxmox_validate_certs }}"
        node: "{{ proxmox_node }}"
        vmid: 201
        state: started

    - name: Obtenir les informations des VM
      community.general.proxmox_vm_info:
        api_host: "{{ proxmox_api_host }}"
        api_user: "{{ proxmox_api_user }}"
        api_token_id: "{{ proxmox_api_token_id }}"
        api_token_secret: "{{ proxmox_api_token_secret }}"
        node: "{{ proxmox_node }}"
        validate_certs: "{{ proxmox_validate_certs }}"
      register: vm_info

    - name: Afficher les VM en cours d'exécution
      ansible.builtin.debug:
        msg: "VM {{ item.vmid }} : {{ item.name }} ({{ item.status }})"
      loop: >-
        {{ vm_info.proxmox_vms |
           selectattr('status', 'equalto', 'running') | list }}
```

Inventaire Ansible dynamique depuis Proxmox

```
# proxmox-inventory.yml (plugin d'inventaire dynamique)
plugin: community.general.proxmox
url: https://10.10.0.11:8006
user: root@pam
token_id: ansible
token_secret: "{{ lookup('env', 'PROXMOX_TOKEN_SECRET') }}"
validate_certs: false
want_facts: true
want_proxmox_nodes_ansible_host: true
groups:
  production: "'production' in (proxmox_tags_parsed | default([]))"
  web: "'web' in (proxmox_tags_parsed | default([]))"
```

```
# Tester l'inventaire dynamique
ansible-inventory -i proxmox-inventory.yml --list | python3 -m json.tool
ansible-inventory -i proxmox-inventory.yml --graph

# Exécuter un playbook avec l'inventaire dynamique
ansible -i proxmox-inventory.yml production -m ping
```

12.6 Exemples pratiques d'automatisation

12.6.1 Créer 10 VM identiques via API

Ce script Python crée 10 VM de développement identiques en parallèle depuis un template :

```
#!/usr/bin/env python3
"""
create-dev-vms.py - Création de 10 VM de développement
TechFlow SAS - Infrastructure Team
"""

import concurrent.futures
import time
import sys
from proxmoxer import ProxmoxAPI

# Configuration
PROXMOX_HOST    = "10.10.0.11"
TOKEN_NAME      = "automation"
TOKEN_SECRET    = "8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c"
TEMPLATE_ID     = 9000
BASE_VMID       = 300
NODE            = "pve-node1"
STORAGE         = "local-lvm"
VM_PREFIX       = "dev-worker"
NB_VMS          = 10
VM_MEMORY       = 4096   # MB
VM_CORES        = 2

def wait_task(proxmox, node, task_id, timeout=300):
    start = time.time()
    while time.time() - start < timeout:
        status = proxmox.nodes(node).tasks(task_id).status.get()
        if status["status"] == "stopped":
            if status.get("exitstatus") == "OK":
                return True
            raise Exception(f"Tâche échouée : {status.get('exitstatus')}")
        time.sleep(3)
    raise TimeoutError(f"Timeout tâche {task_id}")

def create_single_vm(vm_index):
    """Créer une VM individuelle (connexion séparée par thread)"""
    proxmox = ProxmoxAPI(
        PROXMOX_HOST,
        user="root@pam",
        token_name=TOKEN_NAME,
        token_value=TOKEN_SECRET,
        verify_ssl=False
    )

    vmid = BASE_VMID + vm_index
    name = f"{VM_PREFIX}-{vm_index+1:02d}"

    try:
        print(f"[{name}] Clonage du template {TEMPLATE_ID}...")
        task = proxmox.nodes(NODE).qemu(TEMPLATE_ID).clone.post(
            newid=vmid,
            name=name,
            full=1,
            storage=STORAGE,
            description=f"VM développement TechFlow - Instance {vm_index+1}"
        )
        wait_task(proxmox, NODE, task)

        print(f"[{name}] Configuration CPU={VM_CORES}, RAM={VM_MEMORY}MB...")
        proxmox.nodes(NODE).qemu(vmid).config.put(
            cores=VM_CORES,
```

```
            memory=VM_MEMORY,
            tags="development,techflow,auto-created",
            onboot=0
        )

        print(f"[{name}] Démarrage ... ")
        task = proxmox.nodes(NODE).qemu(vmid).status.start.post()
        wait_task(proxmox, NODE, task)

        print(f"[{name}] VM {vmid} créée et démarrée avec succès.")
        return {"vmid": vmid, "name": name, "status": "success"}

    except Exception as e:
        print(f"[{name}] ERREUR : {e}", file=sys.stderr)
        return {"vmid": vmid, "name": name, "status": "error", "error": str(e)}

def main():
    print(f"═══ Création de {NB_VMS} VM de développement ═══")
    print(f"Template: {TEMPLATE_ID} | Nœud: {NODE} | Storage: {STORAGE}")
    print(f"Mémoire: {VM_MEMORY}MB | CPU: {VM_CORES} cœurs")
    print()

    start_time = time.time()
    results = []

    # Créer les VM en parallèle (max 3 simultanément)
    with concurrent.futures.ThreadPoolExecutor(max_workers=3) as executor:
        futures = {
            executor.submit(create_single_vm, i): i
            for i in range(NB_VMS)
        }
        for future in concurrent.futures.as_completed(futures):
            result = future.result()
            results.append(result)

    elapsed = time.time() - start_time
    success = sum(1 for r in results if r["status"] == "success")
    failed  = sum(1 for r in results if r["status"] == "error")

    print(f"\n═══ Résumé ═══")
    print(f"Durée totale       : {elapsed:.0f}s")
    print(f"VM créées          : {success}/{NB_VMS}")
    if failed:
        print(f"VM en erreur       : {failed}")
        for r in results:
            if r["status"] == "error":
                print(f"  - {r['name']} (VMID {r['vmid']}): {r.get('error')}")

if __name__ == "__main__":
    main()
```

12.6.2 Rapport quotidien d'utilisation

```
#!/usr/bin/env python3
"""
daily-report.py - Rapport quotidien d'utilisation Proxmox
TechFlow SAS - Envoi automatique chaque matin à 8h00
"""

import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from datetime import datetime
from proxmoxer import ProxmoxAPI

def get_cluster_report():
    proxmox = ProxmoxAPI(
        '10.10.0.11',
```

```
    user='root@pam',
    token_name='automation',
    token_value='8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c',
    verify_ssl=False
)

report = {
    "date": datetime.now().strftime("%Y-%m-%d %H:%M"),
    "nodes": [],
    "vm_summary": {},
    "storage": [],
    "top_cpu_vms": [],
    "cluster_status": {}
}

# État du cluster
for item in proxmox.cluster.status.get():
    if item.get("type") == "cluster":
        report["cluster_status"] = {
            "name": item.get("name"),
            "nodes": item.get("nodes"),
            "quorum": "OK" if item.get("quorate") else "DÉGRADÉ"
        }

# Informations des nœuds
for node in proxmox.nodes.get():
    node_name = node["node"]
    if node["status"] != "online":
        report["nodes"].append({
            "name": node_name, "status": "HORS LIGNE",
            "cpu_percent": "N/A", "mem_used_gb": "N/A",
            "mem_total_gb": "N/A"
        })
        continue

    status = proxmox.nodes(node_name).status.get()
    report["nodes"].append({
        "name": node_name,
        "status": "En ligne",
        "cpu_percent": f"{status['cpu']*100:.1f}",
        "mem_used_gb": f"{status['memory']['used']/1024**3:.1f}",
        "mem_total_gb": f"{status['memory']['total']/1024**3:.1f}",
        "loadavg": status.get("loadavg", [0, 0, 0])
    })

# Inventaire des VM
all_resources = proxmox.cluster.resources.get(type='vm')
running = sum(1 for r in all_resources if r.get('status') == 'running')
stopped = sum(1 for r in all_resources if r.get('status') == 'stopped')
report["vm_summary"] = {
    "total": len(all_resources), "running": running, "stopped": stopped
}

# Top 5 VM par consommation CPU
running_vms = [r for r in all_resources if r.get('status') == 'running']
top_cpu = sorted(running_vms, key=lambda x: x.get('cpu', 0), reverse=True)[:5]
report["top_cpu_vms"] = [
    {
        "vmid": vm["vmid"],
        "name": vm.get("name", "?"),
        "node": vm["node"],
        "cpu_percent": f"{vm.get('cpu', 0)*100:.1f}"
    }
    for vm in top_cpu
]

# Utilisation des stockages (> 70%)
for node in proxmox.nodes.get():
```

```
        if node["status"] ≠ "online":
            continue
        for storage in proxmox.nodes(node["node"]).storage.get():
            if storage.get("active"):
                used_pct = (storage.get("used", 0) /
                            max(storage.get("total", 1), 1)) * 100
                if used_pct > 70:
                    report["storage"].append({
                        "storage": storage["storage"],
                        "node": node["node"],
                        "used_gb": f"{storage.get('used', 0)/1024**3:.1f}",
                        "total_gb": f"{storage.get('total', 0)/1024**3:.1f}",
                        "used_pct": f"{used_pct:.1f}"
                    })

    return report

def send_report():
    report = get_cluster_report()

    html = f"""<html><body style="font-family:Arial;max-width:800px">
    <h1>Rapport Proxmox TechFlow SAS — {report['date']}</h1>
    <h2>État du cluster</h2>
    <p>Cluster: <b>{report['cluster_status'].get('name')}</b> |
       Nœuds: {report['cluster_status'].get('nodes')} |
       Quorum: <b>{report['cluster_status'].get('quorum')}</b></p>
    <h2>VM : {report['vm_summary']['running']} en cours / {report['vm_summary']
['total']} total</h2>
    <h2>Nœuds</h2>
    <table border="1" style="border-collapse:collapse;width:100%">
    <tr style="background:#f0f0f0"><th>Nœud</th><th>CPU</th><th>Mémoire</th></tr>
    """
    for node in report["nodes"]:
        html += (f"<tr><td>{node['name']}</td>"
                 f"<td>{node['cpu_percent']}%</td>"
                 f"<td>{node['mem_used_gb']} / {node['mem_total_gb']} GB</td></
tr>")
    html += "</table></body></html>"

    msg = MIMEMultipart('alternative')
    msg['Subject'] = f"Rapport Proxmox TechFlow — {report['date']}"
    msg['From'] = "monitoring@techflow.fr"
    msg['To'] = "ops-team@techflow.fr"
    msg.attach(MIMEText(html, 'html'))

    with smtplib.SMTP('smtp.techflow.fr', 587) as server:
        server.starttls()
        server.login("monitoring@techflow.fr", "MotDePasseSMTP")
        server.send_message(msg)
    print(f"Rapport envoyé à ops-team@techflow.fr")

if __name__ == "__main__":
    send_report()
```

Cron pour l'envoi quotidien (jours ouvrés)

```
# Ajouter au crontab de root
echo "0 8 * * 1-5 /opt/proxmox-scripts/bin/python3 /opt/scripts/daily-report.py
>> /var/log/proxmox-reports.log 2>&1" | crontab -
```

12.6.3 Auto-scaling simplifié

Ce script surveille la charge d'un groupe de VM de traitement et démarre/arrête des instances selon la demande :

```
#!/usr/bin/env python3
"""
autoscaler.py — Auto-scaling simplifié pour VM de traitement
TechFlow SAS — Groupe de VM worker pour tâches batch
```

```
"""

import time
import logging
from proxmoxer import ProxmoxAPI

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s %(levelname)s %(message)s',
    handlers=[
        logging.FileHandler('/var/log/proxmox-autoscaler.log'),
        logging.StreamHandler()
    ]
)
log = logging.getLogger(__name__)

CONFIG = {
    "proxmox_host": "10.10.0.11",
    "token_name": "automation",
    "token_secret": "8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c",
    "node": "pve-node1",
    "worker_vmids": [300, 301, 302, 303, 304],
    "cpu_scale_up_threshold": 0.80,
    "cpu_scale_down_threshold": 0.30,
    "min_running_vms": 1,
    "max_running_vms": 5,
    "check_interval": 60,
    "cooldown_period": 300,
}

class ProxmoxAutoscaler:
    def __init__(self, config):
        self.config = config
        self.proxmox = ProxmoxAPI(
            config["proxmox_host"],
            user="root@pam",
            token_name=config["token_name"],
            token_value=config["token_secret"],
            verify_ssl=False
        )
        self.last_scaling_time = 0

    def get_worker_stats(self):
        running, stopped = [], []
        for vmid in self.config["worker_vmids"]:
            try:
                status = self.proxmox.nodes(
                    self.config["node"]).qemu(vmid).status.current.get()
                if status["status"] == "running":
                    running.append({"vmid": vmid, "cpu": status.get("cpu", 0)})
                else:
                    stopped.append(vmid)
            except Exception as e:
                log.warning(f"VM {vmid} inaccessible: {e}")
        return running, stopped

    def scale_up(self, stopped_vms):
        if not stopped_vms:
            log.warning("Pas de VM disponible pour scale-up")
            return False
        vmid = stopped_vms[0]
        log.info(f"SCALE-UP: Démarrage VM {vmid}")
        try:
            self.proxmox.nodes(
                self.config["node"]).qemu(vmid).status.start.post()
            self.last_scaling_time = time.time()
            return True
        except Exception as e:
```

```
            log.error(f"Échec démarrage VM {vmid}: {e}")
            return False

    def scale_down(self, running_vms):
        if len(running_vms) ≤ self.config["min_running_vms"]:
            log.info("Minimum de VM atteint, pas de scale-down")
            return False
        victim = min(running_vms, key=lambda x: x["cpu"])
        log.info(f"SCALE-DOWN: Arrêt VM {victim['vmid']} "
                 f"(CPU: {victim['cpu']*100:.1f}%)")
        try:
            self.proxmox.nodes(self.config["node"]).qemu(
                victim["vmid"]).status.shutdown.post(timeout=60)
            self.last_scaling_time = time.time()
            return True
        except Exception as e:
            log.error(f"Échec arrêt VM {victim['vmid']}: {e}")
            return False

    def check_and_scale(self):
        running, stopped = self.get_worker_stats()
        if not running:
            if stopped:
                self.scale_up(stopped)
            return

        avg_cpu = sum(vm["cpu"] for vm in running) / len(running)
        log.info(f"VM actives: {len(running)} | CPU moyen: {avg_cpu*100:.1f}%")

        if time.time() - self.last_scaling_time < self.config["cooldown_period"]:
            log.debug("Cooldown actif, pas de scaling")
            return

        if (avg_cpu > self.config["cpu_scale_up_threshold"]
                and len(running) < self.config["max_running_vms"]
                and stopped):
            self.scale_up(stopped)
        elif (avg_cpu < self.config["cpu_scale_down_threshold"]
              and len(running) > self.config["min_running_vms"]):
            self.scale_down(running)

    def run(self):
        log.info("Démarrage de l'autoscaler TechFlow Proxmox")
        while True:
            try:
                self.check_and_scale()
            except Exception as e:
                log.error(f"Erreur dans la boucle de scaling: {e}")
            time.sleep(self.config["check_interval"])

if __name__ == "__main__":
    ProxmoxAutoscaler(CONFIG).run()
```

12.6.4 Webhook sur événement cluster

```
#!/usr/bin/env python3
"""
webhook-events.py — Surveillance des événements cluster et webhooks
TechFlow SAS
"""

import requests
import time
import json
from datetime import datetime
from proxmoxer import ProxmoxAPI

PROXMOX_CONFIG = {
```

```
    "host": "10.10.0.11",
    "token_name": "automation",
    "token_secret": "8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c"
}

WEBHOOK_URLS = {
    "slack": "https://hooks.slack.com/services/XXXXX/YYYYY/ZZZZZ",
    "custom": "https://api.techflow.fr/webhooks/proxmox"
}

WATCHED_EVENTS = [
    "start", "stop", "migrate", "destroy",
    "snapshot", "clone", "backup-failed"
]

def send_slack_notification(event_data):
    color = "good"
    if event_data.get("type") in ["stop", "destroy", "backup-failed"]:
        color = "danger"
    elif event_data.get("type") not in ["start"]:
        color = "warning"

    payload = {
        "attachments": [{
            "color": color,
            "title": f"Événement Proxmox : {event_data.get('type', '?').upper()}",
            "fields": [
                {"title": "Nœud", "value": event_data.get("node"), "short": True},
                {"title": "VM/CT", "value": str(event_data.get("vmid", "N/A")),
"short": True},
                {"title": "Statut", "value": event_data.get("status", "OK"),
"short": True},
                {"title": "Heure", "value": event_data.get("time"), "short":
True},
            ],
            "footer": "TechFlow SAS - Proxmox Monitor"
        }]
    }
    requests.post(WEBHOOK_URLS["slack"], json=payload, timeout=10)

def poll_tasks():
    proxmox = ProxmoxAPI(
        PROXMOX_CONFIG["host"],
        user="root@pam",
        token_name=PROXMOX_CONFIG["token_name"],
        token_value=PROXMOX_CONFIG["token_secret"],
        verify_ssl=False
    )

    seen_tasks = set()
    print("Surveillance des événements cluster Proxmox ...")

    while True:
        try:
            for node in proxmox.nodes.get():
                if node["status"] != "online":
                    continue

                for task in proxmox.nodes(node["node"]).tasks.get(limit=20):
                    task_id = task.get("upid", "")
                    if task_id in seen_tasks:
                        continue
                    seen_tasks.add(task_id)

                    task_type = task.get("type", "")
                    if not any(e in task_type.lower() for e in WATCHED_EVENTS):
                        continue
```

```
                    event_data = {
                        "node": node["node"],
                        "vmid": task.get("id"),
                        "type": task_type,
                        "status": task.get("status"),
                        "time": datetime.fromtimestamp(
                            task.get("starttime", 0)
                        ).strftime("%Y-%m-%d %H:%M:%S"),
                        "upid": task_id
                    }

                    print(f"Événement: {json.dumps(event_data)}")
                    try:
                        send_slack_notification(event_data)
                        requests.post(
                            WEBHOOK_URLS["custom"],
                            json=event_data,
                            timeout=10,
                            headers={"X-Proxmox-Event": task_type}
                        )
                    except Exception as e:
                        print(f"Erreur webhook: {e}")

            except Exception as e:
                print(f"Erreur polling: {e}")

            # Éviter une croissance illimitée de la mémoire
            if len(seen_tasks) > 10000:
                seen_tasks = set(list(seen_tasks)[-5000:])

            time.sleep(30)

    if __name__ == "__main__":
        poll_tasks()
```

12.7 Lab TechFlow SAS : observabilité complète

Ce lab final déploie l'ensemble de la stack d'observabilité pour TechFlow SAS, en utilisant toutes les technologies vues dans ce chapitre.

12.7.1 Déploiement Prometheus/Grafana

Architecture du lab TechFlow

```
Cluster Proxmox TechFlow SAS
├── pve-node1 (10.10.0.11)
├── pve-node2 (10.10.0.12)
├── pve-node3 (10.10.0.13)
│
└── VM Monitoring (10.10.0.20) - Debian 12
    ├── Prometheus  :9090
    ├── Grafana     :3000
    ├── Alertmanager :9093
    └── pve-exporter :9221
```

Script de déploiement complet

```
#!/usr/bin/env bash
# deploy-monitoring-techflow.sh
# Déploiement complet de la stack monitoring TechFlow SAS
# A exécuter sur la VM 10.10.0.20 (Debian 12)

set -euo pipefail

PROMETHEUS_VERSION="2.48.1"
ALERTMANAGER_VERSION="0.26.0"
NODE_EXPORTER_VERSION="1.7.0"

echo "═══ Déploiement Stack Monitoring TechFlow SAS ═══"
```

```
echo "Serveur : $(hostname) ($(hostname -I | awk '{print $1}'))"
echo ""

# 1. Mise à jour système
echo "[1/7] Mise à jour du système ..."
apt update -qq && apt upgrade -y -qq

# 2. Dépendances
echo "[2/7] Installation des dépendances ..."
apt install -y -qq curl wget tar python3 python3-pip python3-venv \
    adduser libfontconfig1 apt-transport-https gnupg2 jq

# 3. Utilisateurs système
echo "[3/7] Création des utilisateurs système ..."
for user in prometheus alertmanager node_exporter; do
    id "$user" &>/dev/null || \
        useradd --no-create-home --shell /bin/false "$user"
done

# 4. Répertoires
mkdir -p /etc/prometheus /etc/alertmanager /etc/alertmanager/templates \
         /etc/pve-exporter /var/lib/prometheus /var/lib/alertmanager
chown prometheus:prometheus /etc/prometheus /var/lib/prometheus
chown alertmanager:alertmanager /etc/alertmanager /var/lib/alertmanager

# 5. Installation Prometheus
echo "[4/7] Installation de Prometheus ${PROMETHEUS_VERSION} ..."
cd /tmp
wget -q "https://github.com/prometheus/prometheus/releases/download/v$
{PROMETHEUS_VERSION}/prometheus-${PROMETHEUS_VERSION}.linux-amd64.tar.gz"
tar xzf "prometheus-${PROMETHEUS_VERSION}.linux-amd64.tar.gz"
cp "prometheus-${PROMETHEUS_VERSION}.linux-amd64/prometheus" /usr/local/bin/
cp "prometheus-${PROMETHEUS_VERSION}.linux-amd64/promtool" /usr/local/bin/
cp -r "prometheus-${PROMETHEUS_VERSION}.linux-amd64/consoles" /etc/prometheus/
cp -r "prometheus-${PROMETHEUS_VERSION}.linux-amd64/console_libraries" /etc/
prometheus/
chown -R prometheus:prometheus /etc/prometheus

# 6. Installation Alertmanager
echo "[5/7] Installation d'Alertmanager ${ALERTMANAGER_VERSION} ..."
wget -q "https://github.com/prometheus/alertmanager/releases/download/v$
{ALERTMANAGER_VERSION}/alertmanager-${ALERTMANAGER_VERSION}.linux-amd64.tar.gz"
tar xzf "alertmanager-${ALERTMANAGER_VERSION}.linux-amd64.tar.gz"
cp "alertmanager-${ALERTMANAGER_VERSION}.linux-amd64/alertmanager" /usr/local/bin/
cp "alertmanager-${ALERTMANAGER_VERSION}.linux-amd64/amtool" /usr/local/bin/
chown -R alertmanager:alertmanager /etc/alertmanager /var/lib/alertmanager

# 7. Installation pve-exporter
echo "[6/7] Installation de pve-exporter ..."
python3 -m venv /opt/pve-exporter
/opt/pve-exporter/bin/pip install -q prometheus-pve-exporter

# 8. Installation Grafana
echo "[7/7] Installation de Grafana ..."
wget -q -O /usr/share/keyrings/grafana.key https://apt.grafana.com/gpg.key
echo "deb [signed-by=/usr/share/keyrings/grafana.key] \
  https://apt.grafana.com stable main" \
  > /etc/apt/sources.list.d/grafana.list
apt update -qq
apt install -y -qq grafana

echo ""
echo "═══ Installation terminée ═══"
echo "Prochaines étapes :"
echo "  1. Configurer /etc/pve-exporter/pve.yml"
echo "  2. Configurer /etc/prometheus/prometheus.yml"
echo "  3. Configurer /etc/alertmanager/alertmanager.yml"
echo "  4. systemctl enable --now prometheus alertmanager grafana-server pve-
```

```
exporter"
echo "  5. Accéder à Grafana : http://10.10.0.20:3000"
```

12.7.2 Dashboard "TechFlow Operations"

Création du dashboard via l'API Grafana

```
#!/usr/bin/env python3
"""
create-techflow-dashboard.py
Crée le dashboard "TechFlow Operations" dans Grafana via l'API
"""

import requests

GRAFANA_URL      = "http://10.10.0.20:3000"
GRAFANA_USER     = "admin"
GRAFANA_PASSWORD = "TechFlow2024!"

dashboard = {
    "dashboard": {
        "title": "TechFlow Operations",
        "uid": "techflow-ops",
        "tags": ["proxmox", "techflow", "production"],
        "timezone": "Europe/Paris",
        "refresh": "30s",
        "time": {"from": "now-6h", "to": "now"},
        "panels": [
            {
                "id": 1,
                "title": "Nœuds actifs",
                "type": "stat",
                "gridPos": {"h": 4, "w": 4, "x": 0, "y": 0},
                "targets": [{
                    "expr": "count(pve_up == 1)",
                    "legendFormat": "Nœuds actifs"
                }]
            },
            {
                "id": 2,
                "title": "VM en cours d'exécution",
                "type": "stat",
                "gridPos": {"h": 4, "w": 4, "x": 4, "y": 0},
                "targets": [{
                    "expr": "count(pve_guest_info{type='qemu',
status='running'})",
                    "legendFormat": "VM running"
                }]
            },
            {
                "id": 3,
                "title": "CPU moyen des nœuds (%)",
                "type": "gauge",
                "gridPos": {"h": 8, "w": 6, "x": 0, "y": 4},
                "targets": [{
                    "expr": "avg(pve_node_cpu_ratio) * 100",
                    "legendFormat": "CPU %"
                }],
                "options": {
                    "minValue": 0,
                    "maxValue": 100,
                    "thresholds": {
                        "steps": [
                            {"color": "green", "value": 0},
                            {"color": "yellow", "value": 70},
                            {"color": "red", "value": 90}
                        ]
                    }
                }
```

```
            },
            {
                "id": 4,
                "title": "CPU par nœud (historique)",
                "type": "timeseries",
                "gridPos": {"h": 8, "w": 18, "x": 6, "y": 4},
                "targets": [{
                    "expr": "pve_node_cpu_ratio * 100",
                    "legendFormat": "{{instance}}"
                }],
                "fieldConfig": {
                    "defaults": {
                        "unit": "percent",
                        "min": 0,
                        "max": 100
                    }
                }
            },
            {
                "id": 5,
                "title": "Utilisation mémoire par nœud",
                "type": "timeseries",
                "gridPos": {"h": 8, "w": 12, "x": 0, "y": 12},
                "targets": [{
                    "expr": "(pve_node_memory_used_bytes /
pve_node_memory_total_bytes) * 100",
                    "legendFormat": "{{instance}} RAM %"
                }],
                "fieldConfig": {
                    "defaults": {"unit": "percent", "min": 0, "max": 100}
                }
            },
            {
                "id": 6,
                "title": "Stockage utilisé par pool",
                "type": "bargauge",
                "gridPos": {"h": 8, "w": 12, "x": 12, "y": 12},
                "targets": [{
                    "expr": "(pve_storage_used_bytes / pve_storage_total_bytes) *
100",
                    "legendFormat": "{{id}} ({{instance}})"
                }],
                "options": {
                    "orientation": "horizontal",
                    "reduceOptions": {"calcs": ["lastNotNull"]}
                }
            }
        ]
    },
    "overwrite": True,
    "folderId": 0
}

response = requests.post(
    f"{GRAFANA_URL}/api/dashboards/db",
    json=dashboard,
    auth=(GRAFANA_USER, GRAFANA_PASSWORD),
    headers={"Content-Type": "application/json"}
)

if response.status_code == 200:
    result = response.json()
    print(f"Dashboard créé : {GRAFANA_URL}/d/{result.get('uid')}/techflow-
operations")
else:
    print(f"Erreur {response.status_code} : {response.text}")
```

12.7.3 Alertes production configurées

Vérification et test des alertes

```
# Valider toutes les règles d'alerte
promtool check rules /etc/prometheus/rules/proxmox-alerts.yml

# Lister les groupes de règles chargés
curl -s http://10.10.0.20:9090/api/v1/rules | \
  python3 -c "
import sys, json
data = json.load(sys.stdin)
for group in data['data']['groups']:
    print(f\"Groupe: {group['name']} ({len(group['rules'])} règles)\")
    for rule in group['rules']:
        print(f\"  - {rule['name']} [{rule.get('type','?')}]\")
"

# Vérifier les alertes actives
curl -s http://10.10.0.20:9090/api/v1/alerts | \
  python3 -c "
import sys, json
data = json.load(sys.stdin)
alerts = data['data']['alerts']
if not alerts:
    print('Aucune alerte active - Tout va bien !')
else:
    print(f'{len(alerts)} alerte(s) active(s):')
    for a in alerts:
        sev = a['labels']['severity'].upper()
        name = a['labels']['alertname']
        inst = a['labels'].get('instance', 'N/A')
        summary = a['annotations'].get('summary', '')
        print(f'  [{sev}] {name} ({inst}) - {summary}')
"

# Script de résumé matin avec vérification des silences
echo "=== Alertes actives TechFlow $(date '+%Y-%m-%d %H:%M') ==="
ACTIVE=$(curl -s http://10.10.0.20:9090/api/v1/alerts | \
    jq -r '.data.alerts[] | select(.state=="firing") |
    "[\(.labels.severity | ascii_upcase)] \(.labels.alertname) | \
(.labels.instance // "N/A")"')

if [[ -z "$ACTIVE" ]]; then
    echo "Aucune alerte active"
else
    echo "$ACTIVE"
fi

echo ""
echo "=== Silences actifs ==="
amtool silence query --alertmanager.url=http://10.10.0.20:9093 2>/dev/null || \
    echo "Aucun silence actif"
```

Tableau récapitulatif des alertes TechFlow configurées

Alerte	Seuil	Durée	Sévérité	Canal
`ProxmoxNodeCPUHigh`	CPU > 85%	10 min	warning	Email + Slack
`ProxmoxNodeCPUCritical`	CPU > 95%	5 min	critical	Email + Slack + PagerDuty
`ProxmoxNodeMemoryHigh`	RAM > 85%	15 min	warning	Email + Slack
`ProxmoxNodeMemoryCritical`	RAM > 95%	5 min	critical	Email + Slack + PagerDuty
`ProxmoxStorageHigh`	Disque > 80%	5 min	warning	Email stockage
`ProxmoxStorageCritical`	Disque > 90%	2 min	critical	Email + Slack + PagerDuty
`ProxmoxStorageFull`	Disque > 95%	1 min	critical	Email + Slack + PagerDuty

Alerte	Seuil	Durée	Sévérité	Canal
`ClusterQuorumLost`	Votes < 2	immédiat	critical	Email + Slack + PagerDuty
`ProxmoxNodeDown`	pve_up == 0	2 min	critical	Email + Slack + PagerDuty
`NodeExporterDown`	up == 0	3 min	warning	Email + Slack
`VMCPUHigh`	vCPU > 90%	20 min	warning	Email
`VMMemoryHigh`	vRAM > 90%	10 min	warning	Email

12.7.4 Script d'inventaire automatique

```
#!/usr/bin/env python3
"""
inventory.py - Inventaire complet de l'infrastructure TechFlow
Génère un rapport JSON et CSV de toutes les ressources Proxmox
"""

import csv
import json
import os
from datetime import datetime
from proxmoxer import ProxmoxAPI

OUTPUT_BASE = "/var/reports"
DATE_STR    = datetime.now().strftime('%Y%m%d')
OUTPUT_JSON = f"{OUTPUT_BASE}/inventory-{DATE_STR}.json"
OUTPUT_CSV  = f"{OUTPUT_BASE}/inventory-{DATE_STR}.csv"

def collect_inventory():
    proxmox = ProxmoxAPI(
        '10.10.0.11',
        user='root@pam',
        token_name='automation',
        token_value='8a3f2c1d-4b5e-6f7a-8b9c-0d1e2f3a4b5c',
        verify_ssl=False
    )

    inventory = {
        "generated_at": datetime.now().isoformat(),
        "cluster": {},
        "nodes": [],
        "vms": [],
        "containers": [],
        "storages": [],
        "summary": {}
    }

    # Informations cluster
    for item in proxmox.cluster.status.get():
        if item.get("type") == "cluster":
            inventory["cluster"] = {
                "name": item.get("name"),
                "nodes": item.get("nodes"),
                "quorum": bool(item.get("quorate")),
                "version": item.get("version")
            }

    # Inventaire par nœud
    for node in proxmox.nodes.get():
        node_name = node["node"]
        node_data = {
            "name": node_name,
            "status": node["status"],
            "ip": node.get("ip", "N/A")
        }
```

```
        if node["status"] == "online":
            status = proxmox.nodes(node_name).status.get()
            node_data.update({
                "cpu_count": status.get("cpuinfo", {}).get("cpus", 0),
                "cpu_model": status.get("cpuinfo", {}).get("model", "N/A"),
                "memory_total_gb": round(
                    status["memory"]["total"] / 1024**3, 1),
                "memory_used_gb": round(
                    status["memory"]["used"] / 1024**3, 1),
                "kernel": status.get("kversion", "N/A"),
                "pve_version": status.get("pveversion", "N/A"),
                "uptime_days": status.get("uptime", 0) // 86400
            })

            # VM de ce nœud
            for vm in proxmox.nodes(node_name).qemu.get():
                vm_config = {}
                try:
                    vm_config = proxmox.nodes(node_name).qemu(
                        vm["vmid"]).config.get()
                except Exception:
                    pass

                inventory["vms"].append({
                    "vmid": vm["vmid"],
                    "name": vm.get("name", f"vm-{vm['vmid']}"),
                    "node": node_name,
                    "status": vm.get("status"),
                    "cpu_cores": vm_config.get("cores", vm.get("cpus", 0)),
                    "memory_mb": vm_config.get(
                        "memory", vm.get("maxmem", 0) // 1024**2),
                    "tags": vm_config.get("tags", ""),
                    "description": vm_config.get("description", ""),
                    "onboot": bool(vm_config.get("onboot", 0))
                })

            # Conteneurs de ce nœud
            for ct in proxmox.nodes(node_name).lxc.get():
                inventory["containers"].append({
                    "vmid": ct["vmid"],
                    "name": ct.get("name", f"ct-{ct['vmid']}"),
                    "node": node_name,
                    "status": ct.get("status"),
                    "cpu_cores": ct.get("cpus", 0),
                    "memory_mb": ct.get("maxmem", 0) // 1024**2,
                    "disk_gb": ct.get("maxdisk", 0) // 1024**3
                })

            # Stockages de ce nœud
            for storage in proxmox.nodes(node_name).storage.get():
                if storage.get("active"):
                    used_pct = (storage.get("used", 0) /
                                max(storage.get("total", 1), 1)) * 100
                    inventory["storages"].append({
                        "storage": storage["storage"],
                        "node": node_name,
                        "type": storage.get("type"),
                        "total_gb": round(storage.get("total", 0) / 1024**3, 1),
                        "used_gb": round(storage.get("used", 0) / 1024**3, 1),
                        "available_gb": round(storage.get("avail", 0) / 1024**3,
1),
                        "used_pct": round(used_pct, 1)
                    })

        inventory["nodes"].append(node_data)

    # Résumé global
    inventory["summary"] = {
```

```
        "total_nodes": len(inventory["nodes"]),
        "online_nodes": sum(1 for n in inventory["nodes"]
                            if n["status"] == "online"),
        "total_vms": len(inventory["vms"]),
        "running_vms": sum(1 for v in inventory["vms"]
                           if v["status"] == "running"),
        "total_containers": len(inventory["containers"]),
        "running_containers": sum(1 for c in inventory["containers"]
                                  if c["status"] == "running"),
        "total_storage_tb": round(
            sum(s["total_gb"] for s in inventory["storages"]) / 1024, 2),
        "used_storage_tb": round(
            sum(s["used_gb"] for s in inventory["storages"]) / 1024, 2)
    }

    return inventory

def export_csv(inventory, filepath):
    """Exporter l'inventaire VM en CSV"""
    with open(filepath, 'w', newline='', encoding='utf-8') as f:
        fieldnames = ["vmid", "name", "node", "status", "cpu_cores",
                      "memory_mb", "tags", "onboot", "description"]
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        for vm in sorted(inventory["vms"], key=lambda x: x["vmid"]):
            writer.writerow({k: vm.get(k, "") for k in fieldnames})
    print(f"CSV exporté : {filepath}")

def main():
    os.makedirs(OUTPUT_BASE, exist_ok=True)

    print("Collecte de l'inventaire en cours...")
    inventory = collect_inventory()

    # Export JSON
    with open(OUTPUT_JSON, 'w', encoding='utf-8') as f:
        json.dump(inventory, f, indent=2, ensure_ascii=False)
    print(f"JSON exporté : {OUTPUT_JSON}")

    # Export CSV
    export_csv(inventory, OUTPUT_CSV)

    # Résumé console
    s = inventory["summary"]
    c = inventory["cluster"]
    quorum_str = "OK" if c.get("quorum") else "DÉGRADE"
    print(f"""
═══ Inventaire TechFlow SAS — {inventory['generated_at'][:10]} ═══
Cluster  : {c.get('name')} | Quorum : {quorum_str}
Nœuds    : {s['online_nodes']}/{s['total_nodes']} en ligne
VM       : {s['running_vms']}/{s['total_vms']} en cours d'exécution
CT       : {s['running_containers']}/{s['total_containers']} en cours d'exécution
Stockage : {s['used_storage_tb']} TB utilisés / {s['total_storage_tb']} TB total
""")

if __name__ == "__main__":
    main()
```

Automatisation hebdomadaire de l'inventaire

```
# Crontab pour l'inventaire hebdomadaire (lundi 6h00)
echo "0 6 * * 1 /opt/proxmox-scripts/bin/python3 /opt/scripts/inventory.py \
  >> /var/log/proxmox-inventory.log 2>&1" | crontab -

# Exécuter manuellement
/opt/proxmox-scripts/bin/python3 /opt/scripts/inventory.py

# Consulter le dernier rapport JSON
```

```
cat "/var/reports/inventory-$(date +%Y%m%d).json" | \
  python3 -m json.tool | head -60
```

Récapitulatif du chapitre

Ce chapitre a couvert l'ensemble de la chaîne d'observabilité et d'automatisation pour Proxmox VE 9 :

Monitoring natif — Les graphes RRD de l'interface web fournissent une visibilité immédiate sur les métriques CPU, RAM, réseau et stockage. L'export vers InfluxDB ou Graphite permet l'intégration dans des systèmes de monitoring existants, tandis que les alertes email couvrent les événements critiques de base.

Stack Prometheus/Grafana — `pve-exporter` expose l'API Proxmox au format Prometheus, `node_exporter` collecte les métriques système bas niveau, et Grafana offre des dashboards personnalisés avec alerting visuel. Cette stack constitue le standard industriel pour les infrastructures de production.

Alerting avec Alertmanager — Les règles d'alerte couvrent les cas critiques (CPU, RAM, espace disque, quorum cluster), avec routage multi-canal (email, Slack, PagerDuty), inhibitions intelligentes et gestion des silences de maintenance.

API REST Proxmox — Toutes les opérations de la plateforme sont accessibles via une API REST complète, sécurisée par tickets de session ou API tokens. `pvesh` permet l'exploration interactive directement sur les nœuds.

Automatisation — `proxmoxer` (Python), `curl` + `jq` (Bash), Terraform et Ansible couvrent l'ensemble des besoins d'automatisation, du script ponctuel à l'IaC en production.

Lab TechFlow SAS — Le déploiement complet de la stack d'observabilité sur 10.10.0.20 avec Prometheus (port 9090), Grafana (port 3000) et Alertmanager (port 9093) constitue une référence opérationnelle prête pour la production.

Composant	Port	URL TechFlow
Prometheus	9090	http://10.10.0.20:9090
Grafana	3000	http://10.10.0.20:3000
Alertmanager	9093	http://10.10.0.20:9093
pve-exporter	9221	http://10.10.0.20:9221/pve
node_exporter	9100	http://10.10.0.1x:9100/metrics
API Proxmox	8006	https://10.10.0.11:8006/api2/json

13

Proxmox Datacenter Manager (PDM)

Proxmox Datacenter Manager représente une rupture conceptuelle dans l'approche de gestion des infrastructures virtualisées. Là où chaque cluster Proxmox VE fonctionnait jusqu'alors comme une île autonome dotée de sa propre interface d'administration, PDM introduit un plan de contrôle unifié capable de superviser, orchestrer et déplacer des charges de travail entre plusieurs clusters géographiquement distribués. Ce chapitre en détaille l'architecture, le déploiement et les cas d'usage avancés, en s'appuyant sur le scénario multi-site de TechFlow SAS.

13.1 Présentation de PDM

13.1.1 Rôle et positionnement dans l'écosystème

Proxmox Datacenter Manager (PDM) est un composant logiciel indépendant, livré sous forme de paquet Debian, qui se superpose à une ou plusieurs instances Proxmox VE et Proxmox Backup Server existantes sans les modifier. Son rôle est strictement celui d'un **plan de gestion** (*management plane*) : il ne participe pas au plan de données (trafic VM, réplication Ceph, flux PBS) et n'introduit aucune dépendance critique à l'exécution des charges de travail.

Dans la taxonomie des outils Proxmox, PDM s'insère entre :

- **Proxmox VE** — hyperviseur et orchestrateur local d'un cluster unique
- **Proxmox Backup Server** — serveur de sauvegarde dédié
- **PDM** — tableau de bord et orchestrateur *multi-cluster* et *multi-datacenter*

[Illustration : Positionnement de PDM dans l'écosystème Proxmox : un plan de gestion unifié au-dessus de plusieurs clusters PVE et instances PBS indépendants.]

Positionnement de PDM dans l'écosystème Proxmox : un plan de gestion unifié au-dessus de plusieurs clusters PVE et instances PBS indépendants.

PDM adresse trois grandes familles de besoins :

1. **Visibilité consolidée** : un seul *pane-of-glass* pour l'ensemble du parc virtuel, qu'il s'étende sur un LAN, un WAN ou plusieurs nuages privés.
2. **Orchestration inter-clusters** : migration de VM/CT entre clusters, avec gestion automatisée des prérequis réseau et stockage.
3. **Gouvernance des sauvegardes** : définition et application de politiques PBS qui s'appliquent de façon cohérente à tous les sites.

> **• PDM N'EST PAS UN POINT DE DÉFAILLANCE UNIQUE**
> PDM n'intervient pas dans le chemin critique des VM en production. Si l'instance PDM devient indisponible, les clusters continuent de fonctionner normalement ; seule la vue centralisée est temporairement perdue. Conserver un accès direct à chaque cluster PVE reste donc essentiel pour les opérations d'urgence.

Positionnement vis-à-vis des solutions concurrentes

Comparaison PDM / solutions de gestion multi-hyperviseurs

Critère	PDM	vCenter (VMware)	oVirt Engine	Nutanix Prism
Éditeur	Proxmox Server Solutions	Broadcom	Red Hat / Community	Nutanix
Licence	AGPLv3 / abonnement	Propriétaire	AGPLv3	Propriétaire
Hyperviseurs gérés	PVE uniquement	ESXi uniquement	KVM / RHEV	AHV uniquement
Intégration PBS native	Oui	Non	Non	Non

Migration inter-clusters	Oui (v1.x+)	vMotion intra-vCenter	Live migration intra-DC	Live migration intra-cluster
Modèle de déploiement	VM ou paquet Debian	Appliance VMware	VM / RPM	Appliance intégrée

13.1.2 Architecture PDM

PDM s'articule autour de quatre composants internes étroitement couplés.

Le service `proxmox-datacenter-manager`

Daemon principal écrit en Rust, il expose une API REST sur le port TCP 8443 (TLS obligatoire). Toutes les opérations — lecture d'inventaire, déclenchement de migrations, application de politiques — transitent par cette API. L'interface web est une Single Page Application (SPA) servie par ce même daemon, sans dépendance externe (pas de Node.js séparé, pas de proxy Apache/Nginx requis).

La base de données locale

Les métadonnées (clusters enregistrés, credentials chiffrés, historique des tâches, politiques de sauvegarde) sont stockées dans une base SQLite locale par défaut. Pour les déploiements à haute disponibilité de PDM lui-même, une base PostgreSQL partagée peut être utilisée en conjonction avec un load-balancer applicatif devant plusieurs instances PDM actives/passives.

Le gestionnaire de credentials

PDM chiffre les secrets (tokens API, mots de passe) avec une clé maîtresse dérivée du FQDN de l'hôte PDM et d'un sel aléatoire généré à l'installation. Les credentials ne transitent jamais en clair dans les journaux d'application ou système.

L'agent de collecte d'inventaire

PDM interroge périodiquement les API REST de chaque cluster enregistré (PVE : port 8006, PBS : port 8007) pour mettre à jour son inventaire en mémoire. L'intervalle de rafraîchissement est configurable (défaut : 60 secondes). Cet agent est non-intrusif : il n'installe aucun composant sur les clusters gérés.

Architecture interne de PDM et flux de communication vers les clusters gérés.

Flux de communication typique

```
PDM (port 8443/tcp) ←── Navigateur administrateur (HTTPS)
        │
        ├──→ PVE Cluster Site A   api.site-a.techflow.lan:8006/tcp   (HTTPS + API
Token)
        ├──→ PVE Cluster Site B   api.site-b.techflow.lan:8006/tcp   (HTTPS + API
Token)
        ├──→ PBS Site A           pbs.site-a.techflow.lan:8007/tcp   (HTTPS + API
Token)
        └──→ PBS Site B           pbs.site-b.techflow.lan:8007/tcp   (HTTPS + API
Token)
```

PDM utilise exclusivement des **API Tokens** pour s'authentifier auprès des clusters. Ces tokens sont associés à un utilisateur dédié (`pdm@pve` ou `pdm@pbs`) disposant des privilèges minimaux nécessaires à chaque opération.

13.1.3 Installation et prérequis

Prérequis matériels et logiciels

Prérequis pour le déploiement de PDM

Paramètre	Valeur recommandée
Système d'exploitation hôte	Debian 12 Bookworm (64 bits), ou VM dans un cluster Proxmox VE 8.x+
Processeur	2 vCPU minimum, 4 recommandés pour plus de 10 clusters gérés
Mémoire vive	2 Go minimum, 4 Go recommandés
Espace disque	20 Go (OS + base de données PDM + journaux)
Connectivité réseau	Accès IP aux ports 8006/8007 de tous les nœuds des clusters gérés
Version Proxmox VE gérée	PVE 8.0 ou supérieur (API compatible)
Version Proxmox BS gérée	PBS 3.0 ou supérieur
Version PDM	0.5.x (Tech Preview) → 1.0+ (GA)

> ✔ **DÉPLOIEMENT RECOMMANDÉ : VM DANS LE CLUSTER GÉRÉ**
> Déployez PDM dans une VM résidant sur l'un des clusters gérés, avec des snapshots quotidiens activés via une politique PBS. Cette approche offre portabilité et résilience sans infrastructure dédiée supplémentaire. Allouez au moins 10 Go de RAM à l'hôte Proxmox VE pour que la VM PDM ne soit pas en compétition avec les VM de production sur des machines à faible mémoire.

Ajout du dépôt Proxmox et installation

```
# Importer la clé GPG du projet Proxmox
curl -fsSL https://enterprise.proxmox.com/debian/proxmox-release-bookworm.gpg \
    -o /etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg

# ─── Dépôt no-subscription (test et laboratoire) ───
echo "deb http://download.proxmox.com/debian/pdm bookworm pdm-no-subscription" \
    > /etc/apt/sources.list.d/pdm.list

# ─── Dépôt enterprise (production – abonnement requis) ───
# echo "deb https://enterprise.proxmox.com/debian/pdm bookworm pdm-enterprise" \
#     > /etc/apt/sources.list.d/pdm.list

apt update && apt install -y proxmox-datacenter-manager
```

Initialisation et premier démarrage

```
# Activer le démarrage automatique et lancer immédiatement
systemctl enable --now proxmox-datacenter-manager

# Vérifier l'état du daemon
systemctl status proxmox-datacenter-manager

# Suivre les journaux en temps réel lors de la première initialisation
journalctl -u proxmox-datacenter-manager -f

# L'interface web est accessible sur :
#   https://<IP-PDM>:8443
```

Création du compte administrateur initial

```
# Créer l'utilisateur admin (realm 'pdc' = realm local PDM)
pdm user add admin@pdc --password-stdin <<< "VotreMotDePasseSecurise2026!"

# Attribuer le rôle Administrator sur la racine du périmètre PDM
```

```
pdm acl update --path "/" --user "admin@pdc" --role "Administrator"

# Vérifier la liste des utilisateurs PDM
pdm user list

# Vérifier les ACL définies
pdm acl show --path "/"
```

Sécurisation TLS avec un certificat Let's Encrypt

```
# Enregistrer le compte ACME avec l'adresse e-mail de contact
pdm config set acme.account.email admin@techflow.fr

# Déclarer le ou les domaines à certifier
pdm config set acme.domains pdm.techflow.fr

# Déclencher la demande de certificat (méthode standalone HTTP-01)
pdm acme cert order

# Vérifier que le timer de renouvellement automatique est actif
systemctl status pdm-cert-renewal.timer
```

13.2 Connexion des clusters

13.2.1 Ajout d'un cluster Proxmox VE

L'enregistrement d'un cluster PVE dans PDM s'effectue en deux étapes : création d'un API Token sur le cluster PVE cible, puis déclaration de ce token dans PDM sous la forme d'un *remote*.

Étape 1 — Création du compte et du token sur le cluster PVE

```
# Créer un utilisateur dédié PDM dans le realm d'authentification locale
pveum user add pdm-reader@pve --comment "Proxmox Datacenter Manager service
account"

# Créer un rôle personnalisé avec les privilèges strictement nécessaires
pveum role add PDMRole \
    --privs "VM.Audit,VM.Migrate,VM.PowerMgmt,Datastore.Audit,\
Pool.Audit,SDN.Audit,Sys.Audit,Sys.Modify"

# Assigner ce rôle à l'utilisateur au niveau de la racine de l'arbre PVE
pveum acl modify / --user pdm-reader@pve --role PDMRole

# Générer un API Token sans séparation de privilèges (hérite des ACL user)
pveum user token add pdm-reader@pve pdm-token \
    --privsep 0 \
    --comment "Token pour Proxmox Datacenter Manager - $(date +%Y-%m)"

# IMPORTANT : noter soigneusement la valeur du secret affiché une seule fois
# Format : xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

► SÉPARATION DES PRIVILÈGES : PRIVSEP 0 VS PRIVSEP 1

Avec `--privsep 0`, le token hérite directement des ACL de l'utilisateur, ce qui simplifie la configuration pour PDM. Avec `--privsep 1` (comportement par défaut de PVE), les ACL doivent être affectées au token lui-même via une commande `pveum acl modify` supplémentaire. Les deux approches offrent le même niveau de sécurité fonctionnelle ; `--privsep 0` est conseillé pour PDM afin d'éviter les oublis lors des mises à jour de rôle.

Étape 2 — Enregistrement du remote dans PDM

L'enregistrement peut s'effectuer via l'interface web de PDM (*Datacenter → Remotes → Add → Proxmox VE*) ou via la CLI PDM :

```
# Récupérer l'empreinte SHA-256 du certificat TLS du cluster PVE
FP_SITE_A=$(openssl s_client \
    -connect api.site-a.techflow.lan:8006 </dev/null 2>/dev/null \
```

```
    | openssl x509 -fingerprint -sha256 -noout \
    | sed 's/SHA256 Fingerprint=//')

# Enregistrer le remote
pdm remote add techflow-site-a \
    --type pve \
    --host api.site-a.techflow.lan:8006 \
    --token "pdm-reader@pve!pdm-token=a1b2c3d4-e5f6-7890-abcd-ef1234567890" \
    --fingerprint "${FP_SITE_A}"

# Vérifier immédiatement la connexion et l'inventaire
pdm remote status techflow-site-a
```

Vérification de l'inventaire après enregistrement

```

# Nœuds découverts dans le cluster
pdm remote nodes techflow-site-a

# VM KVM inventoriées
pdm remote vms techflow-site-a

# Conteneurs LXC inventoriés
pdm remote lxc techflow-site-a

# Datastores visibles
pdm remote storage techflow-site-a
```

13.2.2 Ajout d'un cluster Proxmox Backup Server

L'ajout d'une instance PBS suit la même philosophie que pour PVE : un token API dédié avec les privilèges minimaux, puis enregistrement dans PDM.

Préparation sur le serveur PBS

```

# Créer l'utilisateur de service PDM dans le realm local PBS
proxmox-backup-manager user create pdm-reader@pbs \
    --comment "Proxmox Datacenter Manager service account"

# Attribuer le rôle Audit sur l'ensemble des ressources PBS
proxmox-backup-manager acl update / \
    --user pdm-reader@pbs \
    --role Audit

# Générer le token d'API
proxmox-backup-manager user generate-token pdm-reader@pbs pdm-token

# IMPORTANT : noter la valeur du token affichée une seule fois
```

Enregistrement dans PDM

```

# Récupérer l'empreinte TLS du serveur PBS
FP_PBS_A=$(openssl s_client \
    -connect pbs.site-a.techflow.lan:8007 </dev/null 2>/dev/null \
    | openssl x509 -fingerprint -sha256 -noout \
    | sed 's/SHA256 Fingerprint=//')

# Enregistrer le remote PBS
pdm remote add techflow-pbs-a \
    --type pbs \
    --host pbs.site-a.techflow.lan:8007 \
    --token "pdm-reader@pbs!pdm-token=b2c3d4e5-f6a7-8901-bcde-f23456789012" \
    --fingerprint "${FP_PBS_A}"

# Vérifier la connexion
pdm remote status techflow-pbs-a
```

```
# Lister les datastores PBS accessibles
pdm remote datastores techflow-pbs-a
```

13.2.3 Gestion des credentials

Opérations courantes sur les remotes enregistrés

```
# Lister tous les remotes enregistrés et leur statut de connexion
pdm remote list

# Afficher la configuration détaillée d'un remote (sans exposer les secrets)
pdm remote config techflow-site-a

# Mettre à jour le token d'un remote (rotation de credentials)
pdm remote update techflow-site-a \
    --token "pdm-reader@pve!pdm-token-v2=NOUVEAU-TOKEN-SECRET-ICI"

# Tester la connexion après mise à jour
pdm remote status techflow-site-a

# Supprimer un remote (ne supprime rien sur le cluster géré)
pdm remote remove techflow-site-a --confirm
```

Procédure de rotation des API Tokens

> ✔ **BONNE PRATIQUE : ROTATION ANNUELLE DES TOKENS**
> Planifiez une rotation annuelle des API Tokens PDM. La procédure ne provoque aucune interruption de service sur les VM : PDM et les clusters fonctionnent indépendamment. Seule la connexion PDM → cluster est brièvement indisponible pendant la mise à jour (quelques secondes).

```
# 1. Sur un nœud du cluster PVE : générer le nouveau token
pveum user token add pdm-reader@pve pdm-token-v2 \
    --privsep 0 \
    --comment "Token PDM - rotation $(date +%Y-%m)"

# 2. Sur PDM : mettre à jour la référence au token
pdm remote update techflow-site-a \
    --token "pdm-reader@pve!pdm-token-v2=NOUVELLE-VALEUR-SECRETE"

# 3. Tester immédiatement la connexion avec le nouveau token
pdm remote status techflow-site-a

# 4. Sur le nœud PVE : supprimer l'ancien token devenu obsolète
pveum user token remove pdm-reader@pve pdm-token
```

Audit des connexions PDM

```
# Journaux système du service PDM (filtrés sur les événements d'auth)
journalctl -u proxmox-datacenter-manager \
    --since "24 hours ago" | grep -iE "auth|token|login|failed"

# Historique des tâches PDM via l'API
pdm log list --type auth --limit 50

# Export JSON pour intégration SIEM
pdm log list --type auth --output json \
    | jq '.[] | {time:.time, user:.user, action:.action, remote:.remote}'
```

13.3 Vue unifiée multi-cluster

13.3.1 Tableau de bord global

Le tableau de bord PDM agrège en temps quasi-réel les métriques essentielles de l'ensemble des clusters enregistrés. L'interface est structurée en widgets redimensionnables permettant une personnalisation par profil d'utilisateur.

Métriques disponibles dans le tableau de bord

Métriques agrégées du tableau de bord PDM

Métrique	Description
VM running / stopped (par cluster)	Nombre de VM KVM dans chaque état (running, stopped, paused) par remote
CT running / stopped (par cluster)	Idem pour les conteneurs LXC
Cluster health score	État agrégé : Corosync, quorum, nombre de nœuds en ligne / hors ligne
CPU usage global	Utilisation CPU agrégée par cluster (moyenne pondérée par capacité)
Memory usage global	RAM allouée vs RAM disponible par cluster
Storage usage (par datastore)	Occupation (%) de chaque datastore visible dans chaque cluster
PBS datastore status	Espace utilisé/disponible, date du dernier garbage collect, du dernier verify
Recent tasks (toutes sources)	20 dernières tâches cross-clusters (migrations, sauvegardes, démarrages)
Active alerts	Alertes actives issues de tous les clusters (nœud down, disque critique, etc.)

Navigation et filtrage

L'interface web PDM permet de filtrer la vue globale selon plusieurs dimensions :

- **Par remote** : affichage d'un cluster unique ou d'une sélection de clusters
- **Par type de ressource** : VM uniquement, CT uniquement, nœuds, stockage, PBS
- **Par état** : running, stopped, error, locked
- **Par tag PVE** : les tags Proxmox VE appliqués sur les VM sont exposés dans PDM et constituent un vecteur de filtrage puissant (ex. `env:production`, `tier:db`)

13.3.2 Inventaire centralisé des VM/CT

PDM maintient un inventaire en mémoire actualisé toutes les 60 secondes (configurable) de toutes les VM et CT présentes sur les clusters enregistrés. Cet inventaire est interrogeable via l'interface web, la CLI PDM ou l'API REST.

Requêtes d'inventaire via la CLI

```
# Lister toutes les VM de tous les clusters
pdm inventory vms

# Filtrer par nom (glob, sensible à la casse)
pdm inventory vms --name "web-*"

# Filtrer par cluster (remote)
pdm inventory vms --remote techflow-site-a

# Filtrer par état
pdm inventory vms --status running
pdm inventory vms --status stopped

# Afficher les conteneurs LXC d'un cluster spécifique
pdm inventory lxc --remote techflow-site-b

# Sortie JSON pour intégration CMDB ou scripts
pdm inventory vms --output json \
    | jq '.[] | {remote:.remote, id:.vmid, name:.name, status:.status,
node:.node}'
```

Interrogation via l'API REST PDM

```
# Créer un API Token PDM pour l'automatisation
pdm user token add admin@pdc automation-token
# Conserver la valeur affichée
```

```
# Lister les remotes enregistrés
curl -sk \
    -H "Authorization: PBSAPIToken=admin@pdc!automation-token:MON-SECRET" \
    https://pdm.techflow.lan:8443/api2/json/remotes \
    | jq '.data'

# Inventaire complet des VM (tous clusters)
curl -sk \
    -H "Authorization: PBSAPIToken=admin@pdc!automation-token:MON-SECRET" \
    "https://pdm.techflow.lan:8443/api2/json/inventory/vms?remote=techflow-site-a"
\
    | jq '.data[] | {remote:.remote, vmid:.vmid, name:.name, status:.status}'
```

Export CSV pour reporting

```
# Générer un rapport CSV horodaté
pdm inventory vms --output json | \
    jq -r '["Remote","VMID","Nom","Nœud","Statut","CPU","RAM_Mo"],
           (.[] | [
             .remote,
             .vmid,
             .name,
             .node,
             .status,
             .cpus,
             (.maxmem // 0 | . / 1048576 | floor)
           ])
           | @csv' \
    > /tmp/inventaire-vm-$(date +%Y%m%d).csv

# Afficher un aperçu formaté
column -t -s',' /tmp/inventaire-vm-$(date +%Y%m%d).csv | head -25
```

13.3.3 Alertes et événements consolidés

PDM collecte et centralise les événements issus de tous les clusters gérés. Les alertes sont déduites des états renvoyés par l'API de chaque remote et enrichies par des seuils configurables dans PDM.

Catalogue des alertes PDM

Catalogue des alertes générées par PDM

Catégorie	Condition de déclenchement	Sévérité
Node offline	Un nœud PVE ne répond plus à l'API du cluster	Critique
Quorum lost	Le cluster PVE a perdu son quorum Corosync	Critique
Storage critical	Occupation d'un datastore supérieure à 90 %	Élevée
Storage warning	Occupation d'un datastore supérieure à 80 %	Avertissement
VM in error state	VM dans l'état `error` ou `locked` depuis plus de 5 minutes	Élevée
PBS verification failed	La vérification d'un snapshot PBS s'est terminée en erreur	Élevée
PBS datastore full	Espace disponible sur un datastore PBS inférieur à 5 %	Critique
Certificate expiring	Certificat TLS d'un remote expirant dans moins de 30 jours	Avertissement
Remote unreachable	PDM ne peut plus joindre l'API d'un remote enregistré	Élevée
Backup missing	Aucune sauvegarde trouvée pour une VM couverte par une politique active	Élevée

Configuration des notifications externes

PDM intègre le système de notification de Proxmox VE 8.x (endpoints + matchers), supportant nativement SMTP, Webhook générique (Slack, Teams, PagerDuty, etc.) et Gotify.

```
# Créer l'endpoint de notification SMTP
```

```
pdm notification endpoint add smtp-techflow \
    --type smtp \
    --server smtp.techflow.fr \
    --port 587 \
    --mode starttls \
    --username alertes@techflow.fr \
    --from-address "pdm-alerts@techflow.fr" \
    --recipient-list "ops-team@techflow.fr,noc@techflow.fr"

# Créer un matcher pour les alertes critiques et élevées uniquement
pdm notification matcher add critical-ops \
    --endpoint smtp-techflow \
    --match-severity "error,critical"

# Tester l'envoi d'une notification de test
pdm notification test smtp-techflow
```

```

# Endpoint Webhook générique (exemple Slack Incoming Webhook)
pdm notification endpoint add webhook-slack \
    --type webhook \
    --url "https://hooks.slack.com/services/TXXXXXXXX/BXXXXXXXX/XXXXXXXXXXXXXXXX"
\
    --method POST \
    --header "Content-Type=application/json" \
    --body '{"text": "PDM Alert: {{title}}\n{{message}}"}'

# Matcher : toutes alertes → Slack
pdm notification matcher add all-to-slack \
    --endpoint webhook-slack

# Lister les endpoints configurés
pdm notification endpoint list
```

13.4 Migration inter-clusters

13.4.1 Migration de VM entre datacenters

La migration inter-clusters est la fonctionnalité la plus avancée de PDM. Elle diffère fondamentalement de la migration intra-cluster (`qm migrate`) : les deux clusters ne partagent aucun stockage commun et aucune infrastructure Corosync commune. PDM orchestre le transfert intégral — disques, configuration, optionnellement état mémoire — entre deux clusters entièrement indépendants.

Mode offline (cold migration)

La VM est arrêtée proprement sur le cluster source. Ses disques sont transférés via le protocole de streaming HTTP/S interne de Proxmox (utilisé également par la réplication PVE-to-PVE). La configuration de la VM est recréée sur le cluster cible, puis la VM est redémarrée. Le temps d'arrêt est égal au temps de transfert des disques, augmenté de quelques dizaines de secondes pour les opérations de création.

Mode online (warm migration / pre-copy)

PDM initie une pré-copie des blocs disque pendant que la VM tourne sur le cluster source. Lorsque la delta de blocs modifiés devient inférieure à un seuil configurable, la VM est suspendue brièvement, les derniers blocs sont transférés, et la VM repart sur le cluster cible. Pour les VM peu actives en I/O, le temps d'arrêt perçu se limite à quelques secondes.

⚠ PRÉREQUIS DE LA MIGRATION ONLINE INTER-CLUSTERS

La migration online (live) entre deux clusters distincts requiert :

- Un lien réseau de haute qualité entre les sites (idéalement 1 Gbps ou plus, latence inférieure à 10 ms pour limiter les cycles de pré-copie)
- Une connectivité directe entre les hôtes source et cible sur la plage de ports utilisée par le daemon de migration QEMU (par défaut : 60000–60050/tcp)
- Absence de périphériques non-migrables : passthrough PCIe, mémoire hugepages statiques, périphériques USB passthrough
- Identité des architectures CPU source et cible (ou activation du masquage CPU sur le cluster cible pour une compatibilité étendue)

13.4.2 Prérequis réseau et stockage

Matrice de compatibilité des types de stockage

Compatibilité source/cible pour la migration inter-clusters PDM

Stockage source	Stockage cible	Mode migration supporté	Remarque
LVM-Thin local	LVM-Thin local	Offline et Online	Transfert par streaming HTTP/S
Ceph RBD (cluster A)	Ceph RBD (cluster B)	Offline et Online	Clusters Ceph distincts, aucun peering requis
ZFS local (zvol)	ZFS local (zvol)	Offline et Online	Streaming via HTTP/S interne Proxmox
ZFS local (zvol)	LVM-Thin ou Ceph RBD	Offline et Online	Conversion implicite du type de volume
NFS / CIFS partagé	NFS / CIFS partagé	Offline uniquement	Le partage source doit être visible depuis l'hôte cible
iSCSI / Fibre Channel	iSCSI / Fibre Channel	Offline uniquement	Dépend de la visibilité des LUNs entre sites

Règles de pare-feu inter-sites

Ports réseau requis pour la migration PDM inter-clusters

Port	Protocole	Direction	Usage
8006	TCP (HTTPS)	PDM → nœuds PVE	API REST Proxmox VE
8007	TCP (HTTPS)	PDM → serveurs PBS	API REST Proxmox Backup Server
8443	TCP (HTTPS)	Navigateur → PDM	Interface web et API PDM
60000–60050	TCP	Nœuds PVE source → nœuds PVE cible	Daemon de migration QEMU (NBD/SPICE)
2222	TCP	Nœuds PVE source → nœuds PVE cible	SSH interne Proxmox (migration de config)
111, 2049	TCP/UDP	Nœuds PVE → serveur NFS	Si stockage NFS partagé inter-sites

Vérification de la bande passante inter-sites

```
# Depuis un nœud du cluster source, tester l'API du cluster cible
curl -sk https://api.site-b.techflow.lan:8006/api2/json/version \
    | jq '.data.version'

# Test de bande passante avec iperf3
# (lancer d'abord le serveur sur le nœud cible)
ssh root@pve-dr1.site-b.techflow.lan "iperf3 -s -D -p 5201"

# Depuis le nœud source
iperf3 -c pve-dr1.site-b.techflow.lan -p 5201 -t 30 --parallel 4
```

```
# Vérifier l'ouverture du range de ports QEMU migration
for port in 60000 60010 60020 60050; do
    nc -zv pve-dr1.site-b.techflow.lan $port 2>&1
done
```

13.4.3 Procédure pas à pas

La procédure ci-dessous migre la VM `web-prod-01` (VMID 201) du cluster `techflow-site-a` (nœud `pve2`) vers le cluster `techflow-site-b` (nœud `pve-dr1`).

Étape 1 — Vérification de l'état de la VM avant migration

```
# Sur un nœud du cluster source : vérifier l'état de la VM
qm status 201

# Examiner la configuration complète
qm config 201 | grep -E "^(scsi|virtio|ide|sata|net|memory|cores|sockets|cpu)"

# S'assurer qu'aucune tâche ne tourne sur la VM (lock, snapshot en cours ...)
qm task 201

# Vérifier l'absence de lock
qm config 201 | grep lock && echo "VM LOCKED — attendre ou déverrouiller" \
    || echo "OK — pas de lock"
```

Étape 2 — Vérification de compatibilité via PDM

```
pdm migration check \
    --source techflow-site-a \
    --source-node pve2 \
    --vmid 201 \
    --target techflow-site-b \
    --target-node pve-dr1 \
    --target-storage local-lvm

# Sortie attendue (exemple) :
#   ✓ VM trouvée : web-prod-01 (201) sur pve2.site-a
#   ✓ Nœud source accessible : pve2.site-a.techflow.lan
#   ✓ Nœud cible accessible : pve-dr1.site-b.techflow.lan
#   ✓ Stockage cible disponible : local-lvm (182.4 Go libres)
#   ✓ Connectivité réseau inter-sites : OK (latence 8 ms)
#   ✓ Pas de passthrough PCIe détecté : OK
#   ✓ Taille estimée à transférer : ~18.7 Go
#   ✓ Durée estimée (offline, lien 1 Gbps) : ~3-4 minutes
```

Étape 3 — Identifier les ressources du cluster cible

```
# Lister les nœuds et leur charge sur le cluster cible
pdm remote nodes techflow-site-b --output json \
    | jq '.[] | {node:.node, cpu_used:.cpu, mem_used_pct:(.mem/.maxmem*100|
floor)}'

# Lister les datastores disponibles avec l'espace libre
pdm remote storage techflow-site-b --output json \
    | jq '.[] | {storage:.storage, type:.type, avail_gb:(.avail/1073741824|
floor)}'
```

Étape 4 — Déclenchement de la migration

```
# ═══ Migration offline (VM arrêtée) ═══
pdm migration start \
    --source techflow-site-a \
    --source-node pve2 \
    --vmid 201 \
    --target techflow-site-b \
    --target-node pve-dr1 \
```

```
    --target-storage local-lvm \
    --mode offline \
    --keep-source \
    --keep-source-duration 48h

# ═══ Migration online (VM live, si prérequis réseau satisfaits) ═══
pdm migration start \
    --source techflow-site-a \
    --source-node pve2 \
    --vmid 201 \
    --target techflow-site-b \
    --target-node pve-dr1 \
    --target-storage ceph-site-b \
    --mode online

# Suivre la progression (remplacer TASK-ID par la valeur retournée)
watch -n 3 'pdm task log TASK-ID | tail -15'

# Ou surveiller la liste des tâches actives
pdm task list --type migration --status running
```

Étape 5 — Validation post-migration

```
# Vérifier la présence et l'état de la VM sur le cluster cible
pdm remote vms techflow-site-b --name "web-prod-01"

# Vérifier l'état de la VM directement sur le nœud cible
ssh root@pve-dr1.site-b.techflow.lan "qm status 201"

# Tester la connectivité réseau de la VM migrée
ping -c 4 web-prod-01.techflow.lan

# Tester l'applicatif (adapter selon le service hébergé)
curl -sk https://web-prod-01.techflow.lan/healthz | jq .

# Confirmer que la VM source est bien retenue (--keep-source)
pdm remote vms techflow-site-a --name "web-prod-01"
# → Doit apparaître avec le statut "stopped" pendant la période de rétention
```

Étape 6 — Nettoyage de la VM source

```
# Supprimer manuellement la VM source après la période de validation
pdm migration cleanup \
    --source techflow-site-a \
    --vmid 201

# Confirmer la suppression définitive
pdm remote vms techflow-site-a --name "web-prod-01"
# → Doit retourner "no results"
```

• STRATÉGIE KEEP-SOURCE EN PRODUCTION

En production, utilisez systématiquement `--keep-source` avec une durée de rétention adaptée au niveau de criticité de la VM (24 h pour les VM de dev, 72 h ou plus pour les VM de production). La VM source conservée occupe de l'espace disque mais ne consomme pas de CPU/RAM (elle reste arrêtée). Le nettoyage manuel vous laisse le temps d'effectuer des tests applicatifs complets avant la suppression définitive.

13.5 Gestion centralisée des sauvegardes

13.5.1 Politiques PBS multi-cluster

PDM introduit le concept de **politique de sauvegarde centralisée** : une définition unique de planification, de rétention et de destination PBS, applicable simultanément à plusieurs clusters en ciblant des VM par tag, par pool ou en sélectionnant l'intégralité d'un cluster.

Création d'une politique de sauvegarde

```
# Politique quotidienne pour les VM de production — destination PBS Site A
pdm backup-policy create prod-nightly \
    --remote techflow-pbs-a \
    --datastore main-datastore \
    --schedule "daily 01:30" \
    --retention-last 7 \
    --retention-daily 30 \
    --retention-weekly 12 \
    --retention-monthly 12 \
    --mode snapshot \
    --compress zstd \
    --encrypt true

# Associer la politique aux VM taguées "env:production" sur le Site A
pdm backup-policy assign prod-nightly \
    --remote techflow-site-a \
    --tag "env:production"

# Associer à toutes les VM du cluster Site B (DR)
pdm backup-policy assign prod-nightly \
    --remote techflow-site-b \
    --all-vms

# Vérifier les politiques actives et leurs affectations
pdm backup-policy list
pdm backup-policy show prod-nightly
```

Format YAML d'une politique exportée

```
policy:
  name: prod-nightly
  description: "Sauvegarde nocturne VM production — tous sites"
  schedule: "daily 01:30"
  target:
    remote: techflow-pbs-a
    datastore: main-datastore
  options:
    mode: snapshot
    compress: zstd
    encrypt: true
    remove: true
  retention:
    keep-last: 7
    keep-daily: 30
    keep-weekly: 12
    keep-monthly: 12
  scope:
    - remote: techflow-site-a
      selector:
        tags:
          - "env:production"
    - remote: techflow-site-b
      selector:
        all: true
```

Politique différenciée pour les environnements hors-production

```
pdm backup-policy create dev-weekly \
    --remote techflow-pbs-a \
    --datastore dev-datastore \
    --schedule "weekly sunday 03:00" \
    --retention-last 4 \
    --retention-weekly 4 \
    --mode snapshot \
    --compress lzo \
    --encrypt false
```

```
pdm backup-policy assign dev-weekly \
    --remote techflow-site-a \
    --tag "env:development"
```

13.5.2 Synchronisation des datastores

PDM orchestre les **sync jobs** PBS entre sites, permettant la réplication automatique des sauvegardes du site principal vers le site DR. Ces sync jobs sont distincts des jobs de réplication native PBS (configurables directement sur le serveur PBS) et sont gérés dans le plan de gestion PDM.

Configuration d'un sync job inter-sites

```
# Créer le sync job de réplication DR
pdm sync-job create nightly-dr-sync \
    --source-remote techflow-pbs-a \
    --source-datastore main-datastore \
    --target-remote techflow-pbs-b \
    --target-datastore dr-datastore \
    --schedule "daily 04:00" \
    --remove-vanished false \
    --max-stream-size 4

# Déclencher une synchronisation immédiate hors schedule (ex. après incident)
pdm sync-job run nightly-dr-sync

# Surveiller la progression du sync job
pdm task list --type sync --status running
pdm task log <task-id>

# Consulter l'historique des sync jobs
pdm sync-job list
pdm sync-job history nightly-dr-sync
```

✔ LIMITER LA BANDE PASSANTE WAN DES SYNC JOBS

L'option `--max-stream-size` contrôle le parallélisme des flux de données (défaut : 4). Sur un lien WAN limité (< 200 Mbps), réduisez cette valeur à 1 ou 2. Combinez avec une planification nocturne ou en week-end pour éviter d'impacter les sauvegardes primaires encore en cours sur le PBS source.

Vérification de l'intégrité des données répliquées

```
# Sur le serveur PBS DR (site-b) : planifier une vérification hebdomadaire
proxmox-backup-manager verify-job create weekly-verify-dr \
    --datastore dr-datastore \
    --schedule "weekly sunday 05:30" \
    --ignore-verified false \
    --outdated-after 30

# Déclencher une vérification immédiate (pour valider après un premier sync)
proxmox-backup-manager task list | grep verify

# Sur le client PBS (depuis un nœud ou PDM) : vérification d'un namespace
spécifique
proxmox-backup-client verify \
    "pdm-reader@pbs@pbs.site-b.techflow.lan:dr-datastore" \
    --ns "techflow-site-a" 2>&1 | tail -10
```

13.5.3 Rapport global de sauvegarde

PDM génère des rapports de sauvegarde consolidés couvrant l'ensemble de l'infrastructure gérée, avec un statut par VM selon les politiques qui lui sont applicables.

Génération du rapport via la CLI

```
# Rapport synthétique : toutes politiques, tous clusters
pdm backup-report
```

```
# Rapport détaillé avec statut individuel par VM
pdm backup-report --detailed

# Identifier les VM sans sauvegarde récente (> 48 h)
pdm backup-report --detailed --output json \
    | jq '.[] | select(.last_backup_age_hours > 48) |
        {remote:.remote, vmid:.vmid, name:.name,
         last_backup:.last_backup_time,
         status:.last_backup_status}'

# Exporter en CSV pour le système de reporting de TechFlow
pdm backup-report --output csv \
    > /share/reports/backup-report-$(date +%Y%m%d).csv
```

Exemple de sortie de rapport de sauvegarde

```
Remote                VMID   Name                 Dernière sauvegarde    Statut
Taille
────────────────────  ─────  ───────────────────  ─────────────────────  ──────
────────
techflow-site-a         101  pve-dc01             2026-03-16 01:32:44    OK
24.3 Go
techflow-site-a         102  pve-dns01            2026-03-16 01:35:11
OK        3.1 Go
techflow-site-a         201  web-prod-01          2026-03-16 01:40:22    OK
18.7 Go
techflow-site-a         202  web-prod-02          2026-03-16 01:41:05    OK
18.6 Go
techflow-site-a         301  db-prod-primary      2026-03-16 01:45:30    OK
64.2 Go
techflow-site-b         401  k8s-master-01        JAMAIS
MANQUANT  —
techflow-site-b         402  k8s-worker-01        2026-03-15 01:50:18    OK
31.4 Go
techflow-site-b         403  k8s-worker-02        2026-03-16 01:52:00    OK
30.9 Go
```

Notification automatique sur les échecs

```
# Créer un matcher ciblant les alertes de catégorie backup uniquement
pdm notification matcher add backup-failures-alert \
    --endpoint smtp-techflow \
    --match-category "backup" \
    --match-severity "error,critical" \
    --comment "Alertes email sur tout échec de sauvegarde"

# Vérifier la configuration du matcher
pdm notification matcher show backup-failures-alert
```

13.6 Lab TechFlow SAS : PDM multi-site

13.6.1 Scénario : site principal + DR

TechFlow SAS exploite deux datacenters hébergeant des infrastructures Proxmox entièrement indépendantes :

Site A — Paris (production) - Cluster PVE de 4 nœuds : `pve1` à `pve4` - Stockage : Ceph Squid 3 nœuds avec pool RBD `ceph-prod` - PBS local : `pbs.site-a.techflow.lan` — datastore `main-datastore` - VMs de production : bases de données, serveurs web, contrôleurs de domaine

Site B — Lyon (disaster recovery) - Cluster PVE de 2 nœuds : `pve-dr1` et `pve-dr2` - Stockage : LVM-Thin sur SSD local - PBS DR : `pbs.site-b.techflow.lan` — datastore `dr-datastore` - VMs DR : répliques des VM critiques (non démarrées en temps normal)

Plan d'adressage réseau

```
# ──────────────────────────────────────────────────────────────────
# SITE A — PARIS (datacenter principal)
# ──────────────────────────────────────────────────────────────────
10.10.1.0/24   Réseau management PVE Site A
10.10.2.0/24   Réseau VM Site A
10.10.3.0/24   Réseau Ceph (stockage inter-OSD) Site A

pve1.site-a.techflow.lan    10.10.1.11
pve2.site-a.techflow.lan    10.10.1.12
pve3.site-a.techflow.lan    10.10.1.13
pve4.site-a.techflow.lan    10.10.1.14
api.site-a.techflow.lan     10.10.1.10    (VIP HAProxy vers cluster PVE)
pbs.site-a.techflow.lan     10.10.1.20
pdm.techflow.lan            10.10.1.50    (VM PDM sur pve1)

# ──────────────────────────────────────────────────────────────────
# SITE B — LYON (disaster recovery)
# ──────────────────────────────────────────────────────────────────
10.20.1.0/24   Réseau management PVE Site B
10.20.2.0/24   Réseau VM Site B
10.20.3.0/24   Réseau stockage Site B

pve-dr1.site-b.techflow.lan  10.20.1.11
pve-dr2.site-b.techflow.lan  10.20.1.12
api.site-b.techflow.lan      10.20.1.10   (VIP HAProxy vers cluster PVE DR)
pbs.site-b.techflow.lan      10.20.1.20

# ──────────────────────────────────────────────────────────────────
# LIEN INTER-SITES
# ──────────────────────────────────────────────────────────────────
Type     : VPN IPSec site-à-site (IKEv2)
Débit    : 1 Gbps
Latence  : ~8 ms
Routage  : 10.10.0.0/16 ↔ 10.20.0.0/16
```

Objectifs du lab

1. Installer et initialiser PDM sur le Site A.
2. Enregistrer les 4 remotes (2 clusters PVE + 2 instances PBS).
3. Configurer une politique de sauvegarde centralisée avec réplication DR.
4. Effectuer une migration de test inter-site (basculement DR simulé).
5. Valider le rapport global de sauvegarde.

13.6.2 Configuration PDM

Installation de PDM sur la VM dédiée

```
# Sur pve1.site-a : créer la VM PDM (Debian 12 préinstallé via cloud-init)
qm create 500 \
    --name pdm-server \
    --memory 4096 \
    --cores 2 \
    --net0 virtio,bridge=vmbr0,tag=10 \
    --scsi0 local-lvm:20,format=raw \
    --ide2 local:iso/debian-12-genericcloud-amd64.qcow2,import-from=… \
    --ipconfig0 ip=10.10.1.50/24,gw=10.10.1.1 \
    --nameserver 10.10.1.2 \
    --searchdomain techflow.lan \
    --sshkeys /root/.ssh/techflow-admin.pub \
    --ostype l26 \
    --onboot 1

qm start 500
```

```
# ─── Sur la VM PDM ──────────────────────────────────────────────
# Mise à jour du système
```

```
apt update && apt full-upgrade -y

# Pré-requis
apt install -y curl jq openssl

# Import de la clé GPG Proxmox
curl -fsSL https://download.proxmox.com/debian/proxmox-release-bookworm.gpg \
    -o /etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg

# Dépôt PDM (no-subscription pour ce lab)
echo "deb http://download.proxmox.com/debian/pdm bookworm pdm-no-subscription" \
    > /etc/apt/sources.list.d/pdm.list

apt update && apt install -y proxmox-datacenter-manager

# Activation du service
systemctl enable --now proxmox-datacenter-manager

# Vérification
systemctl is-active proxmox-datacenter-manager && echo "PDM actif"

# Création de l'administrateur PDM
pdm user add admin@pdc --password-stdin <<< "TechFlow@Secure2026!"
pdm acl update --path "/" --user "admin@pdc" --role "Administrator"

# Interface accessible : https://10.10.1.50:8443
echo "PDM prêt sur https://pdm.techflow.lan:8443"
```

Préparation des tokens sur tous les clusters

```
# ─── Sur pve1.site-a ──────────────────────────────────────────
pveum user add pdm-reader@pve \
    --comment "PDM TechFlow - service account"

pveum role add PDMRole \
    --privs "VM.Audit,VM.Migrate,VM.PowerMgmt,Datastore.Audit,\
Pool.Audit,SDN.Audit,Sys.Audit,Sys.Modify"

pveum acl modify / --user pdm-reader@pve --role PDMRole

pveum user token add pdm-reader@pve pdm-token \
    --privsep 0 \
    --comment "PDM token - site A - $(date +%Y-%m)"
# Sauvegarder TOKEN_PVE_A = <valeur affichée>
```

```
# ─── Sur pve-dr1.site-b ──────────────────────────────────────
pveum user add pdm-reader@pve \
    --comment "PDM TechFlow DR - service account"

pveum role add PDMRole \
    --privs "VM.Audit,VM.Migrate,VM.PowerMgmt,Datastore.Audit,\
Pool.Audit,SDN.Audit,Sys.Audit,Sys.Modify"

pveum acl modify / --user pdm-reader@pve --role PDMRole

pveum user token add pdm-reader@pve pdm-token \
    --privsep 0 \
    --comment "PDM token - site B DR - $(date +%Y-%m)"
# Sauvegarder TOKEN_PVE_B = <valeur affichée>
```

```
# ─── Sur pbs.site-a ──────────────────────────────────────────
proxmox-backup-manager user create pdm-reader@pbs \
    --comment "PDM TechFlow - PBS Site A"
proxmox-backup-manager acl update / \
    --user pdm-reader@pbs --role Audit
proxmox-backup-manager user generate-token pdm-reader@pbs pdm-token
```

```
# Sauvegarder TOKEN_PBS_A = <valeur affichée>

# ─── Sur pbs.site-b ──────────────────────────────────────────────
proxmox-backup-manager user create pdm-reader@pbs \
    --comment "PDM TechFlow — PBS Site B DR"
proxmox-backup-manager acl update / \
    --user pdm-reader@pbs --role Audit
proxmox-backup-manager user generate-token pdm-reader@pbs pdm-token
# Sauvegarder TOKEN_PBS_B = <valeur affichée>
```

Enregistrement des 4 remotes dans PDM

```
# ─── Sur la VM PDM (10.10.1.50) ──────────────────────────────────

# Fonction utilitaire : récupérer l'empreinte TLS d'un hôte:port
get_fingerprint() {
    openssl s_client -connect "$1" </dev/null 2>/dev/null \
        | openssl x509 -fingerprint -sha256 -noout \
        | sed 's/SHA256 Fingerprint=//'
}

# Remote 1 : Cluster PVE Site A
pdm remote add techflow-site-a \
    --type pve \
    --host api.site-a.techflow.lan:8006 \
    --token "pdm-reader@pve!pdm-token=${TOKEN_PVE_A}" \
    --fingerprint "$(get_fingerprint api.site-a.techflow.lan:8006)"

# Remote 2 : Cluster PVE Site B (DR)
pdm remote add techflow-site-b \
    --type pve \
    --host api.site-b.techflow.lan:8006 \
    --token "pdm-reader@pve!pdm-token=${TOKEN_PVE_B}" \
    --fingerprint "$(get_fingerprint api.site-b.techflow.lan:8006)"

# Remote 3 : PBS Site A
pdm remote add techflow-pbs-a \
    --type pbs \
    --host pbs.site-a.techflow.lan:8007 \
    --token "pdm-reader@pbs!pdm-token=${TOKEN_PBS_A}" \
    --fingerprint "$(get_fingerprint pbs.site-a.techflow.lan:8007)"

# Remote 4 : PBS Site B (DR)
pdm remote add techflow-pbs-b \
    --type pbs \
    --host pbs.site-b.techflow.lan:8007 \
    --token "pdm-reader@pbs!pdm-token=${TOKEN_PBS_B}" \
    --fingerprint "$(get_fingerprint pbs.site-b.techflow.lan:8007)"

# Vérification globale des 4 remotes
echo "═══ Statut des remotes PDM ═══"
pdm remote list
for remote in techflow-site-a techflow-site-b techflow-pbs-a techflow-pbs-b; do
    STATUS=$(pdm remote status "${remote}" 2>&1 | head -1)
    echo "  ${remote}: ${STATUS}"
done
```

Configuration complète des politiques de sauvegarde

```
# ─── Politique production nocturne ──────────────────────────────
pdm backup-policy create prod-nightly \
    --remote techflow-pbs-a \
    --datastore main-datastore \
    --schedule "daily 01:30" \
    --retention-last 7 \
    --retention-daily 30 \
    --retention-weekly 12 \
```

```
    --retention-monthly 12 \
    --mode snapshot \
    --compress zstd \
    --encrypt true

# Appliquer aux VM de production de Site A (par tag)
pdm backup-policy assign prod-nightly \
    --remote techflow-site-a \
    --tag "env:production"

# Appliquer à toutes les VM de Site B
pdm backup-policy assign prod-nightly \
    --remote techflow-site-b \
    --all-vms

# ─── Sync noctune PBS Site A → PBS Site B ──────────────────────
pdm sync-job create nightly-dr-sync \
    --source-remote techflow-pbs-a \
    --source-datastore main-datastore \
    --target-remote techflow-pbs-b \
    --target-datastore dr-datastore \
    --schedule "daily 04:00" \
    --remove-vanished false \
    --max-stream-size 2

# ─── Vérification de la configuration complète ─────────────────
echo "═══ Politiques de sauvegarde ═══"
pdm backup-policy list

echo "═══ Sync jobs ═══"
pdm sync-job list
```

13.6.3 Test de migration inter-site

Le test de migration simule un basculement DR en déplaçant la VM `dr-test-vm` (VMID 999) du Site A (production) vers le Site B (DR). Cette VM est créée spécifiquement pour le test afin de ne pas perturber les VM de production.

Création de la VM de test sur le Site A

```
# Créer une VM de test minimale sur le Site A
qm create 999 \
    --name dr-test-vm \
    --memory 512 \
    --cores 1 \
    --net0 virtio,bridge=vmbr0 \
    --scsi0 local-lvm:8,format=raw \
    --ostype l26 \
    --description "VM de test pour validation migration DR inter-sites PDM"

# Taguer la VM pour l'identifier clairement dans PDM
pvesh set /nodes/pve1/qemu/999/config \
    --tags "test,dr-candidate,env:staging"

# Démarrer et vérifier
qm start 999
sleep 10
qm status 999

# Vérifier que PDM a détecté la nouvelle VM
pdm inventory vms --remote techflow-site-a --name "dr-test-vm"
```

Analyse de compatibilité

```
pdm migration check \
    --source techflow-site-a \
    --source-node pve1 \
```

```
    --vmid 999 \
    --target techflow-site-b \
    --target-node pve-dr1 \
    --target-storage local-lvm \
    --output json | jq '.'
```

Déclenchement de la migration

```
# Lancer la migration offline avec rétention 72h
TASK_ID=$(pdm migration start \
    --source techflow-site-a \
    --source-node pve1 \
    --vmid 999 \
    --target techflow-site-b \
    --target-node pve-dr1 \
    --target-storage local-lvm \
    --target-vmid 999 \
    --mode offline \
    --keep-source \
    --keep-source-duration 72h \
    --description "Test DR TechFlow — $(date +%Y-%m-%d)" \
    --output json | jq -r '.taskid')

echo "Migration lancée — Task ID : ${TASK_ID}"

# Suivre en temps réel
watch -n 5 "pdm task log ${TASK_ID} | tail -20"
```

Journal de migration attendu

```
2026-03-17 10:00:01  INFO  Migration démarrée : dr-test-vm (999) site-a → site-b
2026-03-17 10:00:02  INFO  Arrêt de la VM sur pve1.site-a…
2026-03-17 10:00:09  INFO  VM arrêtée (7 secondes)
2026-03-17 10:00:10  INFO  Début du transfert des disques :
2026-03-17 10:00:10  INFO    scsi0 : 8.0 Go (raw)
2026-03-17 10:00:11  INFO  Connexion établie vers pve-dr1.site-b:60000/tcp
2026-03-17 10:03:47  INFO  Transfert terminé : 8.0 Go en 217 s (37.6 Mo/s)
2026-03-17 10:03:48  INFO  Création de la configuration VM sur pve-dr1 (VMID 999)
2026-03-17 10:03:49  INFO  Démarrage de la VM dr-test-vm sur pve-dr1…
2026-03-17 10:03:55  INFO  VM dr-test-vm (999) en état running sur pve-dr1.site-b
2026-03-17 10:03:55  INFO  VM source conservée sur pve1.site-a (rétention : 72 h)
2026-03-17 10:03:55  INFO  Durée totale de la migration : 234 secondes
2026-03-17 10:03:55  INFO  MIGRATION RÉUSSIE
```

Validation complète

```
echo "═══ 1. Présence sur le cluster cible ═══"
pdm remote vms techflow-site-b --name "dr-test-vm"

echo "═══ 2. État de la VM sur le nœud DR ═══"
ssh root@pve-dr1.site-b.techflow.lan "qm status 999 && qm config 999"

echo "═══ 3. Connectivité réseau de la VM ═══"
ping -c 4 -W 2 10.20.2.50 2>&1   # IP de la VM dr-test-vm sur Site B

echo "═══ 4. Présence conservée sur le cluster source ═══"
pdm remote vms techflow-site-a --name "dr-test-vm"

echo "═══ 5. Tableau de bord PDM — inventaire global ═══"
pdm inventory vms --name "dr-test-vm" --output json | jq '.'

echo "═══ 6. Rapport de sauvegarde après migration ═══"
pdm backup-report --detailed \
    | grep -E "(dr-test-vm|VMID 999)"
```

Nettoyage final du test

```
# Après validation applicative complète, supprimer la VM source retenue
pdm migration cleanup \
    --source techflow-site-a \
    --vmid 999

# Confirmation : la VM ne doit plus exister sur le Site A
pdm remote vms techflow-site-a --name "dr-test-vm"
# Résultat attendu : liste vide

# La VM continue de fonctionner sur le Site B
pdm remote vms techflow-site-b --name "dr-test-vm"
# Résultat attendu : dr-test-vm (999), statut running
```

• BILAN DU LAB TECHFLOW SAS — PDM MULTI-SITE

Le lab a validé l'ensemble du pipeline PDM multi-site :

- **Inventaire unifié** : 2 clusters PVE + 2 instances PBS gérés depuis une interface unique accessible sur https://pdm.techflow.lan:8443
- **Migration inter-site** : une VM de 8 Go a été déplacée du Site A vers le Site B en 234 secondes sur un lien VPN 1 Gbps / 8 ms de latence, avec un temps d'arrêt observé inférieur à 10 secondes
- **Politique de sauvegarde centralisée** : la politique `prod-nightly` applique des paramètres identiques (zstd, chiffrement, rétention 30j quotidienne) aux VM des deux sites sans reconfiguration locale
- **Réplication DR** : le sync job `nightly-dr-sync` garantit la disponibilité des sauvegardes sur le site DR dans un délai de 24 h maximum

Ces résultats confirment que PDM est opérationnel pour les scénarios multi-sites de TechFlow SAS, avec une courbe de prise en main faible et une intégration native dans l'écosystème Proxmox.

► RÉCAPITULATIF DU CHAPITRE 13

Proxmox Datacenter Manager simplifie radicalement la gestion des infrastructures Proxmox distribuées en apportant :

- **Architecture découplée** : PDM est un outil de gestion pur ; sa disponibilité n'est pas sur le chemin critique des VM en production
- **Connexion universelle via API Token** : tout cluster PVE 8.x et tout serveur PBS 3.x peut être enregistré en quelques minutes
- **Vue unifiée** : inventaire, alertes et métriques en temps réel pour l'ensemble du parc, interrogeables via interface web, CLI ou API REST
- **Migration orchestrée** : déplacement de VM entre datacenters entièrement indépendants, avec validation pré-migration et rétention de la source
- **Gouvernance centralisée des sauvegardes** : politiques PBS multi-clusters et sync jobs inter-sites pour garantir la cohérence du plan de DR

PDM est en développement actif. Consultez systématiquement les notes de version avant chaque mise à jour (https://pve.proxmox.com/wiki/Proxmox_Datacenter_Manager) et maintenez un accès direct à chaque cluster PVE en parallèle de PDM.

ANNEXES

References, CLI et checklists

Annexes

Les annexes rassemblent les référentiels techniques indispensables au quotidien de l'administrateur Proxmox VE : commandes CLI complètes, tableaux de ports réseau, checklists de mise en production, configurations types commentées, glossaire des termes techniques et ressources en ligne officielles.

A. Référence des commandes CLI essentielles

A.1 pvecm — gestion du cluster

`pvecm` est l'outil de gestion du cluster Corosync/pmxcfs. Il permet de créer, rejoindre, inspecter et modifier la topologie du cluster Proxmox VE.

```
# ── Gestion du cluster ──────────────────────────────────────
pvecm create <nom>                    # Créer un nouveau cluster
pvecm add <IP-nœud-existant>          # Rejoindre un cluster existant
pvecm status                          # Statut du cluster (quorum, nœuds, votes)
pvecm nodes                           # Lister les nœuds avec leurs IDs Corosync
pvecm delnode <nom-nœud>              # Supprimer un nœud du cluster (hors ligne)
pvecm expected <nb-votes>             # Modifier le quorum attendu (urgence)

# ── QDevice ─────────────────────────────────────────────────
pvecm qdevice setup <IP-qdevice>      # Configurer un QDevice (tiers-arbitre)
pvecm qdevice remove                  # Retirer le QDevice du cluster

# ── Certificats et sécurité ─────────────────────────────────
pvecm updatecerts                     # Régénérer les certificats du nœud local
pvecm updatecerts --force             # Forcer la régénération (après changement de
FQDN)

# ── Diagnostic ──────────────────────────────────────────────
pvecm mtunnel                         # Ouvrir un tunnel SSH inter-nœuds (debug)

# ── Exemples de flux d'usage ────────────────────────────────
# Créer un cluster nommé "techflow-paris"
pvecm create techflow-paris --link0 address=10.10.3.11

# Rejoindre le cluster depuis un second nœud
pvecm add 10.10.3.11 --link0 address=10.10.3.12

# Vérifier le quorum après ajout
pvecm status | grep -E "^(Quorum|Nodes|Votes)"

# Supprimer un nœud défaillant (à exécuter depuis un nœud sain)
pvecm delnode pve4
```

A.2 qm — gestion des VM KVM

`qm` est l'outil de gestion des machines virtuelles KVM. Il gère l'intégralité du cycle de vie des VM : création, configuration, démarrage, arrêt, migration, snapshots et sauvegardes.

```
# ── Cycle de vie ────────────────────────────────────────────
qm list                               # Lister toutes les VM du nœud local
qm create <vmid>                      # Créer une VM (options via ──<param>)
qm start <vmid>                       # Démarrer une VM
qm reboot <vmid>                      # Redémarrer (gracieux via agent QEMU)
qm shutdown <vmid>                    # Arrêt propre (ACPI)
qm stop <vmid>                        # Arrêt forcé (coupure alimentation)
qm suspend <vmid>                     # Suspendre (état mémoire sur disque)
qm resume <vmid>                      # Reprendre depuis la suspension
qm destroy <vmid>                     # Supprimer la VM et ses disques
qm destroy <vmid> --purge             # Supprimer + effacer toutes les configs PBS

# ── Configuration ───────────────────────────────────────────
```

```
qm config <vmid>                    # Afficher la configuration actuelle
qm set <vmid> --memory 4096         # Modifier la RAM allouée (Mo)
qm set <vmid> --cores 4             # Modifier le nombre de cœurs
qm set <vmid> --cpu host            # Modifier le type CPU
qm set <vmid> --net0 virtio,bridge=vmbr0,tag=100
qm set <vmid> --scsi1 local-lvm:50  # Ajouter un disque 50 Go
qm set <vmid> --onboot 1            # Démarrage automatique avec l'hôte
qm set <vmid> --protection 1        # Activer la protection contre la suppression
qm set <vmid> --tags "prod,web"     # Définir les tags

# ── Migration ──────────────────────────────────────────────────
qm migrate <vmid> <nœud-cible>      # Migration online (si stockage partagé)
qm migrate <vmid> <nœud-cible> --offline          # Migration offline
qm migrate <vmid> <nœud-cible> --targetstorage local-lvm

# ── Snapshots ──────────────────────────────────────────────────
qm snapshot <vmid> <nom>            # Créer un snapshot
qm snapshot <vmid> <nom> --vmstate 1 # Snapshot avec état mémoire
qm listsnapshot <vmid>              # Lister les snapshots
qm rollback <vmid> <nom>            # Restaurer un snapshot
qm delsnapshot <vmid> <nom>         # Supprimer un snapshot

# ── Disques ────────────────────────────────────────────────────
qm disk move <vmid> <disque> <stockage>    # Déplacer un disque entre datastores
qm disk resize <vmid> <disque> +<taille>   # Étendre un disque (ex. +20G)
qm disk import <vmid> <fichier> <stockage> # Importer une image disque
qm disk unlink <vmid> --idlist <disque>    # Détacher un disque de la VM

# ── Console et agent ───────────────────────────────────────────
qm terminal <vmid>                  # Accès console série
qm monitor <vmid>                   # Accès au monitor QEMU (mode expert)
qm guest passwd <vmid> <user>       # Changer un mot de passe via l'agent QEMU
qm guest exec <vmid> -- <cmd>       # Exécuter une commande via l'agent QEMU

# ── Cloud-Init ─────────────────────────────────────────────────
qm set <vmid> --cicustom "user=local:snippets/ci-user.yaml"
qm set <vmid> --ipconfig0 ip=10.10.2.50/24,gw=10.10.2.1
qm set <vmid> --nameserver 10.10.1.2
qm cloudinit dump <vmid> user       # Afficher la config cloud-init générée

# ── Tâches et statut ───────────────────────────────────────────
qm status <vmid>                    # Statut de la VM
qm status <vmid> --verbose          # Statut détaillé (uptime, PID QEMU…)
qm task <vmid>                      # Lister les tâches en cours sur la VM
qm wait <vmid> --timeout 60         # Attendre que la VM soit dans un état stable
```

A.3 pct — gestion des conteneurs LXC

`pct` est l'équivalent de `qm` pour les conteneurs LXC. Son interface est délibérément similaire à celle de `qm` pour réduire la courbe d'apprentissage.

```
# ── Cycle de vie ───────────────────────────────────────────────
pct list                            # Lister tous les CT du nœud local
pct create <ctid> <template>        # Créer un CT (chemin du template requis)
pct start <ctid>                    # Démarrer un CT
pct reboot <ctid>                   # Redémarrer un CT
pct shutdown <ctid>                 # Arrêt propre (signal init)
pct stop <ctid>                     # Arrêt forcé (cgroups kill)
pct destroy <ctid>                  # Supprimer le CT et son espace disque
pct destroy <ctid> --purge          # Supprimer + effacer les configs PBS

# ── Configuration ──────────────────────────────────────────────
pct config <ctid>                   # Afficher la configuration
pct set <ctid> --memory 2048        # Modifier la RAM (Mo)
pct set <ctid> --swap 512           # Modifier le swap (Mo)
pct set <ctid> --cores 2            # Modifier les cœurs CPU
pct set <ctid> --net0 name=eth0,bridge=vmbr0,ip=10.10.2.100/24,gw=10.10.2.1
pct set <ctid> --onboot 1           # Démarrage automatique
```

```
pct set <ctid> --unprivileged 1      # Rendre non-privilégié (recrée le CT)
pct set <ctid> --features nesting=1  # Activer la virtualisation imbriquée

# ── Accès et exécution ──────────────────────────────────────
pct enter <ctid>                     # Ouvrir un shell dans le CT
pct exec <ctid> -- <commande>        # Exécuter une commande dans le CT
pct console <ctid>                   # Accès à la console tty du CT

# ── Disques et montages ─────────────────────────────────────
pct resize <ctid> rootfs +10G        # Étendre le rootfs
pct mount <ctid>                     # Monter le FS du CT sur l'hôte (/var/lib/
lxc/<id>/rootfs)
pct unmount <ctid>                   # Démonter
pct set <ctid> --mp0 /mnt/data,mp=/data,size=20G   # Ajouter un montage

# ── Migration ───────────────────────────────────────────────
pct migrate <ctid> <nœud-cible>      # Migration online (CT running, stockage
partagé)
pct migrate <ctid> <nœud-cible> --restart     # Arrêt/transfert/redémarrage
pct migrate <ctid> <nœud-cible> --targetstorage local-lvm

# ── Snapshots ───────────────────────────────────────────────
pct snapshot <ctid> <nom>            # Créer un snapshot
pct listsnapshot <ctid>              # Lister les snapshots
pct rollback <ctid> <nom>            # Restaurer un snapshot
pct delsnapshot <ctid> <nom>         # Supprimer un snapshot

# ── Templates ───────────────────────────────────────────────
pveam update                         # Mettre à jour la liste des templates
disponibles
pveam available                      # Lister les templates disponibles en ligne
pveam download local <template>      # Télécharger un template
pveam list local                     # Lister les templates téléchargés
```

A.4 pvesm — gestion du stockage

`pvesm` (*Proxmox VE Storage Manager*) permet de gérer les configurations de stockage déclarées dans `/etc/pve/storage.cfg`.

```
# ── Gestion des définitions de stockage ─────────────────────
pvesm status                         # Statut de tous les datastores (espace,
état)
pvesm list <stockage>                # Lister les volumes d'un datastore
pvesm add dir <nom> --path /mnt/data # Ajouter un stockage de type répertoire
pvesm add nfs <nom> \
    --server nfs.techflow.lan \
    --export /srv/proxmox \
    --content images,vztmpl,iso,backup
pvesm add lvm-thin <nom> \
    --vgname pve \
    --thinpool data \
    --content images,rootdir
pvesm add rbd <nom> \
    --monhost "10.10.3.11,10.10.3.12,10.10.3.13" \
    --pool ceph-prod \
    --content images,rootdir \
    --krbd 0
pvesm add zfspool <nom> \
    --pool rpool/data \
    --content images,rootdir
pvesm edit <stockage> --shared 1     # Marquer un stockage comme partagé
pvesm set <stockage> --disable 1     # Désactiver temporairement
pvesm remove <stockage>              # Supprimer la définition (pas les données)

# ── Gestion des volumes ─────────────────────────────────────
pvesm alloc <stockage> <vmid> <nom> <taille>   # Allouer un volume
pvesm free <stockage>:<nom-volume>              # Libérer (supprimer) un volume
pvesm path <stockage>:<nom-volume>              # Afficher le chemin absolu d'un
```

```
volume

# ── Import / export ──────────────────────────────────────────
pvesm export <stockage>:<vol> <format> -         # Exporter sur stdout
pvesm import <stockage>:<vol> <format> -         # Importer depuis stdin
```

A.5 pvesh — API REST en ligne de commande

`pvesh` est le client CLI de l'API REST Proxmox VE. Il permet d'interroger et de manipuler toutes les ressources de l'API sans quitter le terminal.

```

# ── Navigation de l'API ──────────────────────────────────────
pvesh ls /                          # Lister les endpoints racine
pvesh ls /nodes                     # Lister les nœuds
pvesh ls /nodes/<nœud>/qemu         # Lister les VM d'un nœud
pvesh ls /cluster                   # Explorer les endpoints du cluster

# ── Lecture (GET) ────────────────────────────────────────────
pvesh get /cluster/status           # Statut global du cluster
pvesh get /nodes                    # Informations sur tous les nœuds
pvesh get /nodes/<nœud>/qemu        # VM d'un nœud avec métriques
pvesh get /nodes/<nœud>/lxc         # CT d'un nœud
pvesh get /nodes/<nœud>/storage     # Stockage d'un nœud
pvesh get /cluster/resources --type vm      # Toutes les VM du cluster
pvesh get /cluster/resources --type storage # Tous les datastores
pvesh get /cluster/ha/resources             # Ressources HA
pvesh get /cluster/ha/status/current        # Statut HA courant
pvesh get /cluster/sdn/vnets                # VNets SDN
pvesh get /access/users                     # Utilisateurs PVE
pvesh get /access/roles                     # Rôles définis
pvesh get /nodes/<nœud>/qemu/<vmid>/config  # Config d'une VM

# ── Modification (POST/PUT) ──────────────────────────────────
pvesh create /nodes/<nœud>/qemu \
    --vmid 999 --name test --memory 512 --cores 1
pvesh set /nodes/<nœud>/qemu/<vmid>/config --memory 2048
pvesh create /nodes/<nœud>/qemu/<vmid>/status/start  # Démarrer
pvesh create /nodes/<nœud>/qemu/<vmid>/status/stop   # Arrêter

# ── Suppression (DELETE) ─────────────────────────────────────
pvesh delete /nodes/<nœud>/qemu/<vmid>

# ── Options de formatage ─────────────────────────────────────
pvesh get /cluster/resources --output-format json-pretty
pvesh get /cluster/resources --output-format yaml
pvesh get /nodes --output-format text    # Format tabulaire (défaut)
```

A.6 ceph — commandes Ceph

Les commandes `ceph` s'exécutent sur tout nœud où un daemon MON ou MGR tourne. Elles permettent de gérer et surveiller l'ensemble du cluster Ceph.

```

# ── Statut et santé ──────────────────────────────────────────
ceph status                         # Statut global du cluster
ceph health                         # Résumé de santé (HEALTH_OK / WARN / ERR)
ceph health detail                  # Détail des alertes actives
ceph -w                             # Surveillance en temps réel des événements
ceph df                             # Utilisation globale et par pool
ceph df detail                      # Détail par pool avec objets

# ── OSD ──────────────────────────────────────────────────────
ceph osd status                     # Statut de tous les OSD
ceph osd tree                       # Arborescence CRUSH complète
ceph osd df                         # Occupation de chaque OSD
ceph osd df tree                    # Occupation avec hiérarchie CRUSH
ceph osd pool ls                    # Lister les pools
ceph osd pool stats                 # Statistiques par pool
```

```
ceph osd out <id>                       # Marquer un OSD "out" (début de rebalance)
ceph osd in <id>                        # Remettre un OSD "in"
ceph osd down <id>                      # Forcer un OSD "down" (urgence)
ceph osd crush remove <osd.id>          # Retirer un OSD de la CRUSH map

# ── Pools ──────────────────────────────────────────────────────────
ceph osd pool create <nom> <pg>         # Créer un pool (ex. 64 PG pour 3 nœuds)
ceph osd pool delete <nom> <nom> --yes-i-really-really-mean-it
ceph osd pool get <nom> size            # Lire le facteur de réplication
ceph osd pool set <nom> size 3          # Modifier le facteur de réplication
ceph osd pool set <nom> pg_autoscale_mode on   # Activer l'autoscaling des PG

# ── Moniteurs ──────────────────────────────────────────────────────
ceph mon stat                           # Statut des MON
ceph mon dump                           # Configuration des MON
ceph mon add <id> <IP>                  # Ajouter un MON
ceph mon remove <id>                    # Retirer un MON

# ── Gestion des pannes ─────────────────────────────────────────────
ceph osd blocked-by                     # Identifier les OSD bloqués
ceph pg dump_stuck                      # Placement Groups bloqués
ceph osd scrub <id>                     # Déclencher un scrub sur un OSD
ceph osd deep-scrub <id>                # Déclencher un deep-scrub

# ── RADOS Block Device (RBD) ───────────────────────────────────────
rbd ls <pool>                           # Lister les images RBD d'un pool
rbd info <pool>/<image>                 # Informations d'une image
rbd du <pool>                           # Espace utilisé par pool
rbd snap ls <pool>/<image>              # Lister les snapshots d'une image
rbd snap purge <pool>/<image>           # Supprimer tous les snapshots d'une image

# ── CephFS ─────────────────────────────────────────────────────────
ceph fs status                          # Statut CephFS
ceph fs ls                              # Lister les systèmes de fichiers Ceph
ceph mds stat                           # Statut des MDS
```

A.7 ha-manager — haute disponibilité

`ha-manager` gère les ressources HA (VM et CT) supervisées par le cluster Proxmox VE.

```
# ── Statut et configuration ────────────────────────────────────────
ha-manager status                       # Statut global HA (CRM + LRM)
ha-manager config                       # Afficher la configuration HA

# ── Ressources ─────────────────────────────────────────────────────
ha-manager add <sid> \
    --state started \
    --group ha-group-prod \
    --max_restart 3 \
    --max_relocate 2              # Ajouter une VM/CT au HA (sid = vm:101 ou ct:
201)
ha-manager set <sid> --state started   # Changer l'état souhaité (started/stopped/
disabled)
ha-manager set <sid> --group <groupe>  # Changer de groupe HA
ha-manager remove <sid>                # Retirer une ressource du HA
ha-manager list                         # Lister toutes les ressources HA

# ── Groupes ────────────────────────────────────────────────────────
ha-manager groupadd <nom> \
    --nodes "pve1:2,pve2:1,pve3:1" \
    --restricted 0 \
    --nofailback 0                 # Créer un groupe HA
ha-manager groupset <nom> --nodes "pve1:3,pve2:2"  # Modifier les priorités
ha-manager groupremove <nom>       # Supprimer un groupe

# ── Règles d'affinité (PVE 9.x) ────────────────────────────────────
ha-manager affinity create web-cluster \
    --type positive \
```

```
    --vms "vm:201,vm:202,vm:203"   # Affinité positive : VM groupées sur même nœud
ha-manager affinity create db-isolation \
    --type negative \
    --vms "vm:301,vm:302"          # Affinité négative : VM séparées sur nœuds
distincts
ha-manager affinity list           # Lister les règles d'affinité
ha-manager affinity delete <nom>   # Supprimer une règle

# ─── Maintenance et CRM ──────────────────────────────────────────
ha-manager crm-command node-maintenance enable <nœud>   # Passer en maintenance
ha-manager crm-command node-maintenance disable <nœud>  # Sortir de maintenance
ha-manager crm-command migrate <sid> <nœud>             # Forcer une migration HA
ha-manager crm-command relocate <sid> <nœud>            # Relocaliser offline
```

A.8 pveum — gestion des utilisateurs

`pveum` gère les utilisateurs, groupes, rôles et règles d'accès (ACL) de Proxmox VE.

```

# ─── Utilisateurs ────────────────────────────────────────────────
pveum user list                       # Lister tous les utilisateurs
pveum user add <user@realm> \
    --comment "Prénom Nom" \
    --email user@techflow.fr \
    --groups "ops-team"
pveum user modify <user@realm> --enable 1   # Activer un utilisateur
pveum user modify <user@realm> --enable 0   # Désactiver
pveum user delete <user@realm>              # Supprimer
pveum passwd <user@realm>                   # Changer le mot de passe

# ─── API Tokens ──────────────────────────────────────────────────
pveum user token add <user@realm> <token-id> \
    --privsep 0 \
    --expire 0 \
    --comment "Token automation"
pveum user token list <user@realm>          # Lister les tokens d'un utilisateur
pveum user token remove <user@realm> <id>   # Révoquer un token

# ─── Groupes ─────────────────────────────────────────────────────
pveum group list                      # Lister les groupes
pveum group add <groupe> --comment "Équipe OPS"
pveum group modify <groupe> --users "alice@pam,bob@pam"
pveum group delete <groupe>

# ─── Rôles ───────────────────────────────────────────────────────
pveum role list                       # Lister les rôles et leurs privilèges
pveum role add <rôle> --privs "VM.Audit,VM.PowerMgmt"
pveum role modify <rôle> --privs "VM.Audit,VM.PowerMgmt,VM.Config.Options"
pveum role delete <rôle>

# ─── ACL ─────────────────────────────────────────────────────────
pveum acl modify <chemin> \
    --user <user@realm> --role <rôle>     # Assigner un rôle à un utilisateur
pveum acl modify <chemin> \
    --group <groupe> --role <rôle>        # Assigner un rôle à un groupe
pveum acl delete <chemin> \
    --user <user@realm> --role <rôle>     # Révoquer
pveum acl list                            # Lister toutes les ACL

# ─── Authentification (realms) ───────────────────────────────────
pveum realm list                      # Lister les realms configurés
pveum realm add ad \
    --type ad \
    --domain techflow.fr \
    --server dc01.techflow.fr \
    --default 0
pveum realm add ldap \
    --type ldap \
    --server ldap.techflow.fr \
```

```
    --base_dn "ou=users,dc=techflow,dc=fr" \
    --user_attr sAMAccountName
pveum realm sync <realm>             # Synchroniser les utilisateurs depuis LDAP/
AD

# ─── Authentification multifacteur ──────────────────────────────
pveum user tfa add <user@realm> --type totp  # Activer TOTP pour un utilisateur
pveum user tfa list <user@realm>              # Lister les méthodes MFA
pveum user tfa delete <user@realm> <id>       # Supprimer une méthode MFA
```

B. Ports réseau Proxmox VE

Le tableau suivant recense l'intégralité des ports TCP et UDP utilisés par les différents services de Proxmox VE, PBS et PDM. Il constitue la référence pour la rédaction des règles de pare-feu en production.

Tableau complet des ports réseau Proxmox

Port(s)	Protocole	Direction	Service	Description
8006	TCP (HTTPS)	Client → PVE	Interface web PVE	Accès à l'interface web et à l'API REST Proxmox VE
8007	TCP (HTTPS)	Client → PBS	Interface web PBS	Accès à l'interface web et à l'API REST Proxmox Backup Server
8443	TCP (HTTPS)	Client → PDM	Interface web PDM	Accès à l'interface web et à l'API REST Proxmox Datacenter Manager
22	TCP (SSH)	Admin → PVE	OpenSSH	Administration SSH des nœuds PVE (à restreindre aux IPs de gestion)
2222	TCP (SSH)	PVE → PVE	Migration SSH	SSH interne utilisé lors des migrations de VM/CT entre nœuds
5900–5999	TCP	Client → PVE	VNC console	Accès aux consoles VNC des VM (redirigé via le proxy websocket PVE)
3128	TCP	PVE → Internet	SPICE proxy	Proxy SPICE pour les consoles SPICE (optionnel)
60000–60050	TCP	PVE → PVE	Migration QEMU	Daemon de migration QEMU (NBD) entre nœuds source et cible
111	TCP/UDP	PVE → NFS	Portmapper RPC	Mappage de ports RPC pour le protocole NFS
2049	TCP/UDP	PVE → NFS	NFS	Accès aux datastores NFS (stockage partagé)
3260	TCP	PVE → iSCSI	iSCSI initiator	Accès aux LUNs iSCSI (stockage SAN)
445	TCP	PVE → CIFS	SMB/CIFS	Accès aux partages Windows/Samba (stockage CIFS)
5404–5405	UDP	PVE ↔ PVE	Corosync	Communication Corosync inter-nœuds (heartbeat, votes, synchronisation)
5406+	UDP	PVE ↔ PVE	Corosync (liens sup.)	Liens Corosync supplémentaires (link1, link2) pour la redondance
6789	TCP	PVE → Ceph	Ceph MON	Communication avec les daemons Ceph Monitor
3300	TCP	PVE → Ceph	Ceph MON (v2)	Protocol Ceph Messenger v2 (msgr2) — préféré depuis Ceph Nautilus
6800–7300	TCP	PVE ↔ Ceph	Ceph OSD	Communication entre clients Ceph et les daemons OSD (replication, recovery)
7480	TCP	Client → Ceph	Ceph RGW	Passerelle objet Ceph (RadosGW) — interface S3/Swift (si activée)
8080	TCP	PVE → Ceph		

			Ceph MGR dashboard	Interface web du Manager Ceph (si module dashboard activé)
9093	TCP	Monitoring → PVE	Prometheus scrape	Endpoint de métriques Prometheus exposé par pve-exporter (si installé)
9283	TCP	Monitoring → Ceph	Ceph Prometheus	Endpoint de métriques Prometheus exposé par le module MGR Prometheus
4789	UDP	PVE ↔ PVE	VXLAN (SDN)	Encapsulation VXLAN pour les zones SDN de type Simple, EVPN ou QinQ
179	TCP	PVE ↔ Routeurs	BGP (SDN EVPN)	Sessions BGP pour le control-plane EVPN (zones SDN EVPN avec FRR)
53	UDP/TCP	PVE → DNS	DNS	Résolution de noms (FQDN des nœuds, IPAM SDN)
123	UDP	PVE → NTP	NTP	Synchronisation de l'horloge système (essentiel pour Corosync et Ceph)
25 / 465 / 587	TCP	PVE → SMTP	SMTP	Envoi de notifications e-mail par PVE, PBS et PDM

C. Checklists

C.1 Checklist installation nœud

• CHECKLIST — INSTALLATION D'UN NŒUD PROXMOX VE

Matériel et BIOS - [ ] Virtualisation matérielle activée dans le BIOS (Intel VT-x / AMD-V + IOMMU) - [ ] RAM de type ECC (fortement recommandé pour les données de production) - [ ] IPMI / iDRAC / iLO fonctionnel et accessible depuis le réseau de management - [ ] Disques de boot en RAID-1 logiciel (ZFS mirror) ou matériel (RAID-1 HW) - [ ] Disques de stockage VM identifiés et dédiés (séparés du disque système) - [ ] Câblage réseau double (au moins 2 NIC pour la redondance des bonds)

Installation Proxmox VE - [ ] Téléchargement de l'ISO depuis https://www.proxmox.com/downloads (vérifier SHA-256) - [ ] Installation avec ZFS RAID-1 sur les disques système (ashift=12 pour SSD/NVMe) - [ ] Adresse IP management statique, FQDN résolu en DNS interne et externe - [ ] Dépôt no-subscription ou enterprise configuré selon la licence disponible - [ ] Dépôt `pve-no-subscription` retiré si abonnement enterprise en place - [ ] Mise à jour complète post-installation : `apt update && apt full-upgrade -y` - [ ] Redémarrage après mise à jour du kernel

Configuration système - [ ] NTP configuré et synchronisé (`timedatectl status`) - [ ] Résolution DNS fonctionnelle (`host pve1.techflow.lan`) - [ ] SSH durci : authentification par clé uniquement, root-login désactivé si IPMI disponible - [ ] fail2ban installé et configuré pour SSH et l'interface web - [ ] Firewall PVE activé avec règles restrictives sur l'interface management - [ ] Certificat TLS valide (Let's Encrypt ou CA interne) installé sur l'interface web

C.2 Checklist formation cluster

• CHECKLIST — FORMATION D'UN CLUSTER PROXMOX VE

Prérequis avant la formation - [ ] Nombre de nœuds impair (3, 5, 7) ou QDevice planifié pour 2 nœuds - [ ] Tous les nœuds ont le même fuseau horaire et sont synchronisés NTP - [ ] Résolution DNS réciproque fonctionnelle entre tous les nœuds (A + PTR) - [ ] Interface de transport Corosync dédiée (séparée du réseau VM) - [ ] Ping sans perte entre toutes les interfaces Corosync prévues - [ ] Ports Corosync ouverts sur tous les pare-feu inter-nœuds (5404–5405/UDP) - [ ] Pas de VM en cours de fonctionnement sur les nœuds à joindre

Formation du cluster - [ ] Cluster créé sur le premier nœud avec `pvecm create <nom> --link0 address=<IP>` - [ ] Chaque nœud suivant rejoint avec `pvecm add <IP-nœud-1> --link0 address=<IP-propre>` - [ ] Quorum vérifié : `pvecm status | grep "Quorum"` → `Quorum provider: corosync_votequartd` - [ ] Tous les nœuds visibles : `pvecm nodes` - [ ] pmxcfs synchronisé sur tous les nœuds : `ls /etc/pve/nodes/` - [ ] Interface web accessible sur tous les nœuds (même vue cluster)

Post-formation - [ ] Corosync configuré avec liens redondants si 2 interfaces de transport disponibles - [ ] Encryption Corosync vérifiée (activée par défaut en PVE 8.x+) : `grep crypto /etc/pve/corosync.conf` - [ ] Stockage partagé ajouté et visible depuis tous les nœuds (`pvesm status`) - [ ] Test de migration d'une VM test entre nœuds (`qm migrate 9999 <autre-nœud>`)

C.3 Checklist avant mise en production

• CHECKLIST — VALIDATION AVANT MISE EN PRODUCTION

Virtualisation et VM - [ ] Templates de base créés et testés (Debian, Ubuntu, Windows Server) - [ ] Cloud-init validé sur au moins un template Linux et un template Windows - [ ] Redimensionnement disque testé (qm disk resize) - [ ] Migration live testée entre tous les nœuds (avec stockage partagé) - [ ] Migration offline testée entre tous les nœuds - [ ] Snapshots créés et restaurés avec succès sur une VM test

Haute disponibilité - [ ] Fencing matériel configuré et testé (IPMI / watchdog) - [ ] Au moins une ressource HA test validée (démarrage auto après arrêt nœud) - [ ] Groupes HA créés avec priorités cohérentes avec la topologie matérielle - [ ] Règles d'affinité définies pour les VM qui ne doivent pas cohabiter

Stockage et sauvegardes - [ ] PBS installé, datastore créé, encryption activée - [ ] Premier job de sauvegarde exécuté et terminé avec succès - [ ] Restauration complète d'une VM depuis PBS validée sur un nœud différent - [ ] Politique de rétention configurée et vérifiée - [ ] Supervision de l'espace PBS en place (alerte à 80 %)

Réseau SDN (si activé) - [ ] Zones SDN créées et appliquées sur tous les nœuds (`pvesdn apply`) - [ ] VNets testés : ping entre VM sur le même VNet, isolation entre VNets vérifiée - [ ] IPAM fonctionnel si utilisé (adresse assignée automatiquement à une VM test)

Sécurité et accès - [ ] MFA activé sur tous les comptes administrateurs - [ ] Comptes de service créés avec privilèges minimaux (RBAC) - [ ] Accès à l'interface web restreint aux IPs de management - [ ] Certificats TLS valides sur tous les services (PVE, PBS, PDM) - [ ] Documentation des procédures de reprise sur incident rédigée

Monitoring - [ ] Collecte de métriques Prometheus configurée - [ ] Tableaux de bord Grafana en place pour CPU, RAM, disques, réseau - [ ] Alertes critiques configurées (nœud down, espace disque critique, échec sauvegarde) - [ ] Notification testée et réceptionnée sur les canaux définis (email, Slack, etc.)

C.4 Checklist sécurité

• CHECKLIST SÉCURITÉ — PROXMOX VE EN PRODUCTION

Authentification et accès - [ ] MFA (TOTP ou WebAuthn) obligatoire pour tous les comptes à privilèges - [ ] Aucun compte générique partagé : comptes nominatifs uniquement - [ ] Mot de passe root SSH complexe (≥ 20 caractères) ou accès root SSH désactivé - [ ] Authentification SSH par clé uniquement (`PasswordAuthentication no`) - [ ] fail2ban actif sur SSH (port 22) et interface web (port 8006) - [ ] Délai de session web configuré (timeout)

Contrôle d'accès (RBAC) - [ ] Principe du moindre privilège appliqué à tous les rôles - [ ] Aucun utilisateur avec le rôle `Administrator` à la racine sauf les comptes d'urgence - [ ] Rôles personnalisés créés pour les opérateurs (audit seul, VM seul, etc.) - [ ] Tokens API avec durée de validité limitée pour les automatisations

Réseau et pare-feu - [ ] Firewall Proxmox VE activé au niveau datacenter et au niveau nœud - [ ] Interface de management isolée sur un VLAN dédié (non routable depuis les VM) - [ ] Accès à l'interface web limité aux IPs ou plages CIDR de confiance - [ ] Pas d'accès direct Internet depuis les interfaces de management (proxy HTTP si requis)

Chiffrement - [ ] Corosync avec chiffrement activé (par défaut PVE 8.x) - [ ] Sauvegardes PBS chiffrées avec une clé maîtresse stockée hors-site - [ ] Disques des VM critiques chiffrés via LUKS ou Windows BitLocker (guest-level) - [ ] TLS valide et à jour sur toutes les interfaces web (PVE, PBS, PDM)

Mises à jour et correctifs - [ ] Politique de mises à jour de sécurité définie (au moins mensuelle) - [ ] `unattended-upgrades` configuré pour les patchs de sécurité critiques - [ ] Nœuds mis à jour en roulement (un nœud à la fois, VM évacuées avant) - [ ] Suivi des CVE Proxmox et Debian via le tracker de sécurité officiel

Journalisation et audit - [ ] Journaux système centralisés (rsyslog vers un SIEM ou un serveur de logs) - [ ] Rétention des journaux d'audit PVE ≥ 90 jours - [ ] Alertes sur les authentifications échouées répétées - [ ] Revue périodique des utilisateurs et tokens actifs (trimestrielle)

D. Configurations types

D.1 /etc/network/interfaces type production

```
# ──────────────────────────────────────────────────────────────
# /etc/network/interfaces — pve1.site-a.techflow.lan
# Architecture : 4 ports physiques
#   eno1 + eno2  →  bond0 (LACP) → management + VM
#   eno3 + eno4  →  bond1 (LACP) → Ceph (stockage)
# ──────────────────────────────────────────────────────────────

auto lo
iface lo inet loopback

# ─── Ports physiques management (LACP 802.3ad) ──────────────────
auto eno1
iface eno1 inet manual
    bond-master bond0

auto eno2
iface eno2 inet manual
    bond-master bond0

# ─── Bond0 : agrégation LACP management/VM ──────────────────────
auto bond0
iface bond0 inet manual
    bond-slaves eno1 eno2
    bond-mode 802.3ad
    bond-miimon 100
    bond-xmit-hash-policy layer3+4
    bond-lacp-rate fast

# ─── Bridge management (VLAN 10) ────────────────────────────────
```

```
auto vmbr0
iface vmbr0 inet static
    address 10.10.1.11/24
    gateway 10.10.1.1
    bridge-ports bond0.10
    bridge-stp off
    bridge-fd 0
    dns-nameservers 10.10.1.2 10.10.1.3
    dns-search techflow.lan
    mtu 1500

# ─── Bridge VM (VLAN trunk 100-999) ──────────────────────────────
auto vmbr1
iface vmbr1 inet manual
    bridge-ports bond0
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware yes
    bridge-vids 100-999
    mtu 9000

# ─── Ports physiques Ceph (LACP 802.3ad) ─────────────────────────
auto eno3
iface eno3 inet manual
    bond-master bond1

auto eno4
iface eno4 inet manual
    bond-master bond1

# ─── Bond1 : agrégation LACP Ceph/stockage ───────────────────────
auto bond1
iface bond1 inet manual
    bond-slaves eno3 eno4
    bond-mode 802.3ad
    bond-miimon 100
    bond-xmit-hash-policy layer3+4
    bond-lacp-rate fast
    mtu 9000

# ─── Interface Ceph (IP dédiée réseau stockage) ──────────────────
auto bond1.30
iface bond1.30 inet static
    address 10.10.3.11/24
    mtu 9000

# ─── Interface Corosync (VLAN management dédié 11) ───────────────
# Optionnel : si Corosync utilise un VLAN séparé du management
auto bond0.11
iface bond0.11 inet static
    address 10.10.11.11/24
    mtu 1500
```

D.2 corosync.conf type 3 nœuds

```
# ─────────────────────────────────────────────────────────────────
# /etc/pve/corosync.conf — Cluster techflow-paris, 3 nœuds, 2 liens
# Généré et maintenu par pvecm — NE PAS ÉDITER MANUELLEMENT
# ─────────────────────────────────────────────────────────────────

totem {
    version: 2
    cluster_name: techflow-paris
    config_version: 7

    # Transport kronosnet (défaut PVE 8.x+, remplace UDP)
    transport: knet
```

```
    # Chiffrement du trafic Corosync (aes256 + sha256)
    crypto_cipher: aes256
    crypto_hash: sha256

    interface {
        linknumber: 0
        knet_transport: udp
        knet_link_priority: 1
    }

    interface {
        linknumber: 1
        knet_transport: udp
        knet_link_priority: 2
    }
}

nodelist {
    node {
        name: pve1
        nodeid: 1
        ring0_addr: 10.10.1.11     # Lien 0 : réseau management
        ring1_addr: 10.10.11.11    # Lien 1 : réseau Corosync dédié
        quorum_votes: 1
    }
    node {
        name: pve2
        nodeid: 2
        ring0_addr: 10.10.1.12
        ring1_addr: 10.10.11.12
        quorum_votes: 1
    }
    node {
        name: pve3
        nodeid: 3
        ring0_addr: 10.10.1.13
        ring1_addr: 10.10.11.13
        quorum_votes: 1
    }
}

quorum {
    provider: corosync_votequorum
    # Quorum requis : floor(3/2) + 1 = 2 votes
    # Tolérance aux pannes : 1 nœud sur 3
}

logging {
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    to_syslog: yes
    timestamp: on
    logger_subsys {
        subsys: QUORUM
        debug: off
    }
}
```

D.3 Politique de sauvegarde PBS type

```
# ──────────────────────────────────────────────────────────────────
# Configuration PBS — Politique de sauvegarde type pour la production
# Cible : serveur PBS pbs.site-a.techflow.lan
# Datastore : main-datastore (chiffré, compression zstd)
# ──────────────────────────────────────────────────────────────────
```

```
# Sur le nœud PVE, ajouter le datastore PBS comme source de sauvegarde
pvesm add pbs pbs-main \
    --server pbs.site-a.techflow.lan \
    --datastore main-datastore \
    --username backup@pbs \
    --fingerprint "AA:BB:CC: ... " \
    --content backup

# Créer les jobs de sauvegarde dans PVE (via l'API ou l'interface web)
# Job 1 : VM critiques — sauvegarde quotidienne 01h30
pvesh create /cluster/backup \
    --starttime "01:30" \
    --schedule "daily" \
    --storage pbs-main \
    --mode snapshot \
    --compress zstd \
    --notes-template "{{guestname}} - sauvegarde automatique {{date}}" \
    --all 0 \
    --vmid "101,102,201,202,301,302" \
    --mailnotification always \
    --mailto "ops@techflow.fr"

# Job 2 : Tous les CT — sauvegarde quotidienne 03h00
pvesh create /cluster/backup \
    --starttime "03:00" \
    --schedule "daily" \
    --storage pbs-main \
    --mode snapshot \
    --compress zstd \
    --all 0 \
    --pool "conteneurs-prod"
```

```
# Configuration de la rétention sur le datastore PBS
# À appliquer dans l'interface web PBS : Datastore → Options → Prune
prune-schedule: "daily 02:00"
keep-last: 7           # 7 dernières sauvegardes (protection court terme)
keep-daily: 30         # 1 sauvegarde par jour sur 30 jours
keep-weekly: 12        # 1 sauvegarde par semaine sur 12 semaines (3 mois)
keep-monthly: 12       # 1 sauvegarde par mois sur 12 mois (1 an)
keep-yearly: 3         # 1 sauvegarde par an sur 3 ans (compliance)
```

E. Glossaire

Le glossaire ci-dessous recense les termes techniques utilisés dans ce livre, avec leur définition dans le contexte Proxmox VE / Linux / virtualisation.

Glossaire Proxmox VE — termes techniques

Terme	Définition
ACL	*Access Control List* — liste de règles associant un chemin de ressource PVE,
ACME	*Automatic Certificate Management Environment* — protocole permettant
Affinité (règle d')	Contrainte HA définissant si des VM doivent être regroupées sur le même nœud
Ballooning	Technique de gestion dynamique de la mémoire des VM : le driver `balloon`
BGP	*Border Gateway Protocol* — protocole de routage de type path-vector utilisé
BlueStore	Backend de stockage par défaut des daemons OSD Ceph depuis Ceph Luminous.
Bridge (réseau)	Interface réseau virtuelle Linux (commutateur logiciel L2) reliant les interfaces
Ceph	Système de stockage distribué open-source fournissant trois services :
Cloud-Init	Standard de facto pour l'initialisation automatique des VM au premier démarrage :
Corosync	Service de communication de cluster assurant le heartbeat, le vote de quorum

CRS	*Cluster Resource Scheduler* — mécanisme de PVE qui équilibre automatiquement
CRUSH	*Controlled Replication Under Scalable Hashing* — algorithme pseudo-aléatoire
EVPN	*Ethernet VPN* — extension BGP (RFC 7432) utilisée comme plan de contrôle
Fabric (SDN)	Réseau de routage automatique entre les nœuds d'un cluster PVE utilisant un
Fencing	Mécanisme d'isolation d'un nœud défaillant pour éviter le *split-brain* :
HA	*High Availability* — capacité du cluster à redémarrer automatiquement les VM
HCI	*Hyper-Converged Infrastructure* — architecture où le compute (CPU/RAM),
IPAM	*IP Address Management* — gestion centralisée des plages d'adresses IP et de
KSM	*Kernel Same-page Merging* — mécanisme du noyau Linux qui fusionne les pages
kronosnet (knet)	Bibliothèque de transport réseau multi-liens utilisée par Corosync depuis
KVM	*Kernel-based Virtual Machine* — module hyperviseur intégré au noyau Linux
LRM	*Local Resource Manager* — composant du HA de PVE s'exécutant sur chaque
LXC	*Linux Containers* — technologie de virtualisation légère au niveau OS utilisant
MDS	*Metadata Server* — daemon Ceph gérant les métadonnées du système de fichiers
MGR	*Manager* — daemon Ceph fournissant des fonctions de monitoring, de reporting
MON	*Monitor* — daemon Ceph maintenant la carte (*map*) authoritative du cluster :
NUMA	*Non-Uniform Memory Access* — topologie des serveurs multi-sockets où l'accès
OSD	*Object Storage Daemon* — daemon Ceph gérant un disque physique ou logique.
PBS	*Proxmox Backup Server* — serveur de sauvegarde open-source développé par
PDM	*Proxmox Datacenter Manager* — outil de gestion multi-cluster de Proxmox VE.
PG (Placement Group)	Unité de placement et de réplication dans Ceph. Les objets RADOS sont mappés
pmxcfs	*Proxmox Cluster File System* — système de fichiers distribué basé sur Corosync
PVE	*Proxmox Virtual Environment* — plateforme de virtualisation open-source
QDevice	Serveur tiers (Raspberry Pi, VM Linux légère) fournissant un vote de quorum
RBAC	*Role-Based Access Control* — modèle de contrôle d'accès où les permissions
RBD	*RADOS Block Device* — couche de stockage bloc exposée par Ceph via le
RPO	*Recovery Point Objective* — perte de données maximale acceptable exprimée
RTO	*Recovery Time Objective* — durée maximale d'indisponibilité acceptable lors
SDN	*Software-Defined Networking* — approche réseau dans laquelle la configuration
SR-IOV	*Single Root I/O Virtualization* — technologie PCIe permettant à un périphérique
VNI	*VXLAN Network Identifier* — identifiant 24 bits d'un segment réseau VXLAN
VNet	*Virtual Network* — objet réseau du SDN Proxmox VE représentant un bridge
VRF	*Virtual Routing and Forwarding* — table de routage virtualisée et isolée
VTEP	*VXLAN Tunnel Endpoint* — point de terminaison d'un tunnel VXLAN. Dans
VXLAN	*Virtual Extensible LAN* — protocole d'encapsulation réseau (RFC 7348)
vzdump	Outil de sauvegarde natif de Proxmox VE produisant des archives `.tar` (LXC)
ZFS	*Zettabyte File System* — système de fichiers et gestionnaire de volumes

F. Ressources en ligne

F.1 Documentation officielle Proxmox

Documentation officielle Proxmox

Ressource	URL et description
Wiki Proxmox VE	https://pve.proxmox.com/wiki/ — documentation complète, guides d'installation,
Documentation PVE (PDF/ HTML)	https://pve.proxmox.com/pve-docs/ — manuel de référence officiel PVE,
Documentation PBS	https://pbs.proxmox.com/docs/ — manuel de référence Proxmox Backup Server
Documentation PDM	https://pve.proxmox.com/wiki/Proxmox_Datacenter_Manager — wiki PDM
API Explorer PVE	https://pve.proxmox.com/pve-docs/api-viewer/ — interface interactive pour
API Explorer PBS	https://pbs.proxmox.com/docs/api-viewer/ — idem pour Proxmox Backup Server
Notes de version PVE	https://pve.proxmox.com/wiki/Roadmap — roadmap et historique des versions

F.2 Forum et communauté

Forums, communautés et canaux de support

Ressource	URL et description
Forum officiel Proxmox	https://forum.proxmox.com/ — forum officiel très actif, tickets de support
Subreddit r/Proxmox	https://www.reddit.com/r/Proxmox/ — communauté Reddit anglophone, retours
Discord Proxmox (communautaire)	Lien disponible sur le subreddit r/Proxmox — canal de discussion en temps réel
GitHub Proxmox	https://github.com/proxmox — code source des outils Proxmox (pve-manager,
Bugtracker Proxmox	https://bugzilla.proxmox.com/ — rapports de bugs officiels et suivi des

F.3 Sécurité et bulletins

Ressources de sécurité Proxmox et Debian

Ressource	URL et description
Tracker de sécurité Debian	https://security-tracker.debian.org/tracker/ — CVE affectant les paquets
Advisories Proxmox	https://forum.proxmox.com/ (catégorie Announcements) — bulletins de sécurité
CVE NIST (recherche)	https://nvd.nist.gov/vuln/search — recherche de CVE par produit (QEMU, KVM,

F.4 Outils et intégrations complémentaires

Outils complémentaires pour l'écosystème Proxmox

Outil	URL et description
Terraform provider Proxmox (telmate)	https://github.com/Telmate/terraform-provider-proxmox — provider Terraform
Terraform provider Proxmox (bpg)	https://github.com/bpg/terraform-provider-proxmox — alternative maintenue

Ansible collection community.general	https://docs.ansible.com/ansible/latest/collections/community/general/ —
Proxmox VE Helper-Scripts	https://tteck.github.io/Proxmox/ — scripts communautaires pour le déploiement
Prometheus PVE Exporter	https://github.com/prometheus-community/prometheus-pve-exporter — collecteur
Grafana dashboards Proxmox	https://grafana.com/grafana/dashboards/?search=proxmox — tableaux de bord
pvetools	https://github.com/extremeshok/xshok-proxmox — scripts de post-installation

F.5 Certifications et formation

Formations et certifications officielles Proxmox

Ressource	URL et description
Proxmox Training Center	https://proxmox.com/en/training — formations officielles PVE, PBS et Ceph
Certification PVA	*Proxmox VE Associate* — certification officielle niveau débutant (examen
Certification PVP	*Proxmox VE Professional* — certification officielle niveau avancé couvrant
Communauté de formation	https://learning.proxmox.com/ — plateforme de formation e-learning officielle

► MAINTIEN À JOUR DE CETTE ANNEXE

Les versions logicielles et les URL mentionnées dans cette annexe correspondent à l'état de l'art au moment de la rédaction (mars 2026, Proxmox VE 9.x). L'écosystème Proxmox évolue rapidement : consultez systématiquement le wiki officiel (https://pve.proxmox.com/wiki/) et les notes de version avant toute mise à niveau majeure en production.