RAPPORT DE STAGE

Administration Systèmes et Réseaux

GADONNAUD Ewen

BTS SIO — Promotion 2025-2026

Établissement d'accueil : Lycée Paul-Louis Courier

Section : BTS SIO (Services Informatiques aux Organisations)

Durée : 18 mai 2026 - 24 juin 2026

Tuteur de stage : M. BALNY David

Remerciements

Je tiens à remercier **M. Balny David**, mon tuteur de stage, pour son encadrement et sa disponibilité tout au long de ces cinq semaines. Son accompagnement au quotidien, ses conseils techniques et sa pédagogie m'ont permis d'acquérir une approche rigoureuse et méthodique face aux problématiques rencontrées.

Je remercie également **M. Demoulière Quentin**, enseignant pédagogique, pour son suivi et son expertise dans les domaines de l'administration systèmes et réseaux, ainsi que **Mme Ferreira Mercier Christèle** pour sa bienveillance tout au long de ce stage.

Enfin, je remercie l'ensemble des personnes qui ont contribué, de près ou de loin, à la réussite de ce stage. Cette expérience au sein de l'infrastructure réseau de l'établissement m'a offert une vision concrète des responsabilités liées au métier d'administrateur systèmes et réseaux, et a renforcé mon projet professionnel.

Sommaire

1. Introduction

Dans le cadre de la formation BTS SIO (Services Informatiques aux Organisations) du lycée Paul-Louis Courier, la réalisation d'un stage en milieu professionnel constitue une étape obligatoire du cursus. Ce stage, d'une durée de cinq semaines, a pour objectif d'approfondir les compétences techniques et organisationnelles acquises en formation, en les confrontant aux réalités du terrain.

La particularité de ce stage réside dans le fait qu'il se déroule au sein de l'établissement scolaire lui-même, le lycée Paul-Louis Courier, dont je suis également élève. Cette configuration offre une perspective à la fois technique et institutionnelle car il s'agit d'intervenir sur l'infrastructure réelle qui supporte quotidiennement les activités pédagogiques et administratives de l'établissement.

Ce rapport de stage a pour objectif de présenter l'environnement dans lequel le stage s'est déroulé, puis d'exposer les différentes missions techniques réalisées. Il s'organise autour de trois grandes parties : une introduction à l'établissement et à son infrastructure réseau, la présentation des missions menées (déploiement d'images machines avec FOG, mise en place d'une solution de supervision), et une conclusion synthétisant les apports de cette expérience.

1.1. Présentation de l'établissement

Le lycée Paul-Louis Courier est un lycée public situé à Tours (Indre-et-Loire). Il propose des formations générales, technologiques et professionnelles. La section BTS SIO (Services Informatiques aux Organisations) y est implantée depuis plusieurs années et forme des techniciens spécialisés dans les infrastructures réseaux, les systèmes d'information et le développement applicatif.

L'établissement dispose d'une infrastructure informatique significative, comprenant plusieurs salles de travaux pratiques dédiées à la section SIO (salles 406, 407, 408, 409), un réseau segmenté en VLANs, une ferme de serveurs virtualisés sous Proxmox et Nutanix, ainsi que des équipements actifs (commutateurs, routeurs) administrés par l'équipe pédagogique.

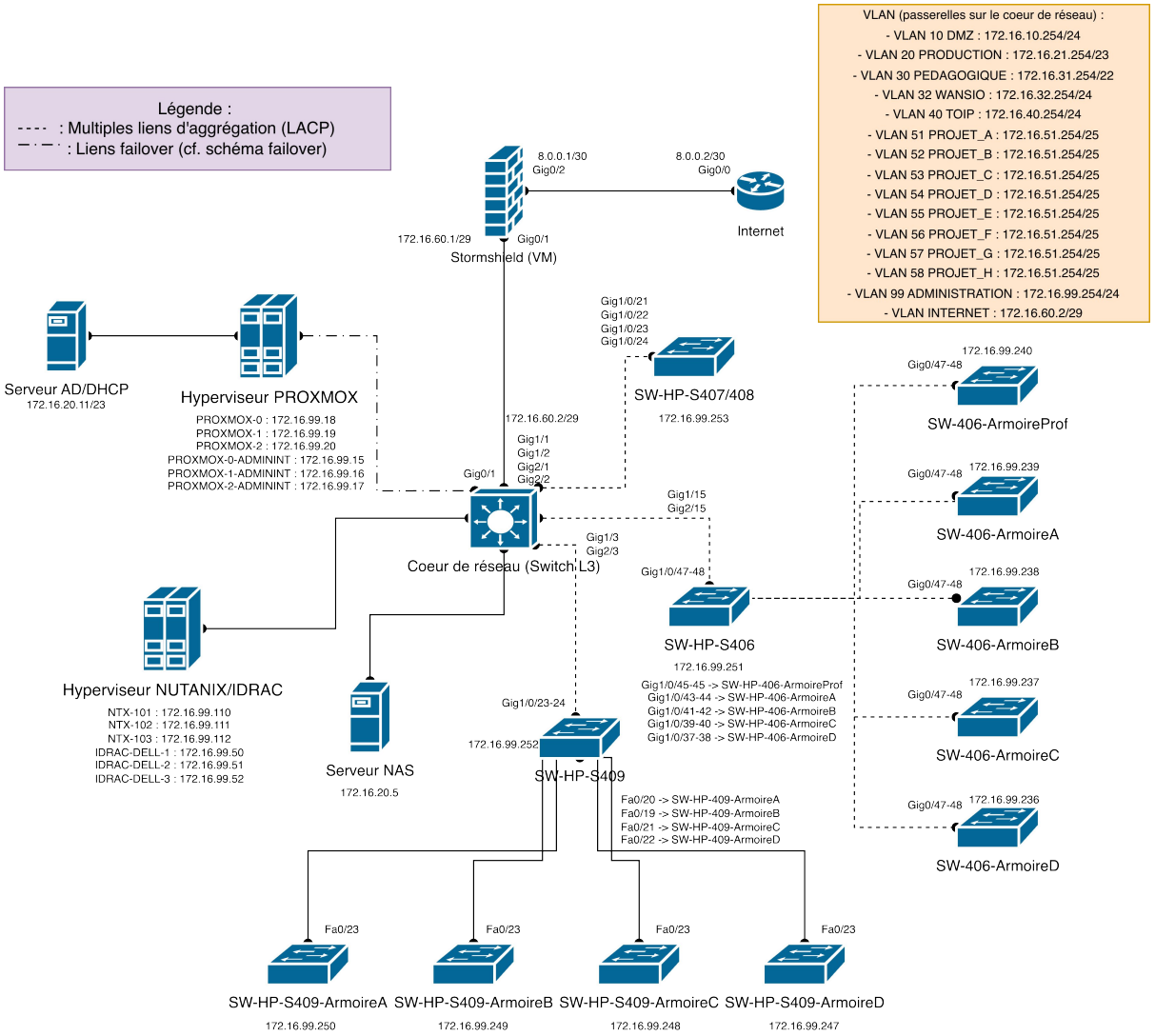
1.2. Présentation de la section BTS SIO

Le BTS SIO (Services Informatiques aux Organisations) est un diplôme d'État de niveau Bac+2 qui forme des techniciens supérieurs capables de gérer, maintenir et faire évoluer le système d'information d'une organisation. Il se décline en deux options :

- Option SISR (Solutions d'Infrastructure, Systèmes et Réseaux) : administration des réseaux, sécurité, virtualisation, supervision.
- Option SLAM (Solutions Logicielles et Applications Métiers) : développement d'applications, bases de données, intégration.

Dans le cadre de ce stage, les missions confiées s'inscrivent dans l'option SISR : déploiement d'images systèmes, administration réseau, et mise en place d'outils de supervision.

1.3. Schéma réseau physique de la section SIO



L'infrastructure réseau de la section SIO repose sur une architecture hiérarchique à trois niveaux : un cœur de réseau (commutateur L3) assurant le routage inter-VLAN, des commutateurs de distribution par salle (SW-HP-S406, SW-HP-S407/408, SW-HP-S409), et des commutateurs d'accès par armoire connectés aux postes de travail.

La segmentation en VLANs permet d'isoler les différents usages : VLAN de production (serveurs, hyperviseurs), VLAN pédagogique (postes élèves), VLANs projets (51 à 58, un par groupe), VLAN d'administration (99, dédié à la gestion des équipements réseau) et VLAN DMZ. Chaque VLAN dispose de sa propre passerelle configurée en SVI sur le cœur de réseau L3.

L'accès Internet est assuré par un Stormshield faisant office de pare-feu et de routeur NAT/PAT, relié au cœur de réseau via un lien point-à-point (172.16.60.0/29). Les hyperviseurs Proxmox et Nutanix/IDRAC sont accessibles via le VLAN d'administration.

1.4. Infrastructure physique

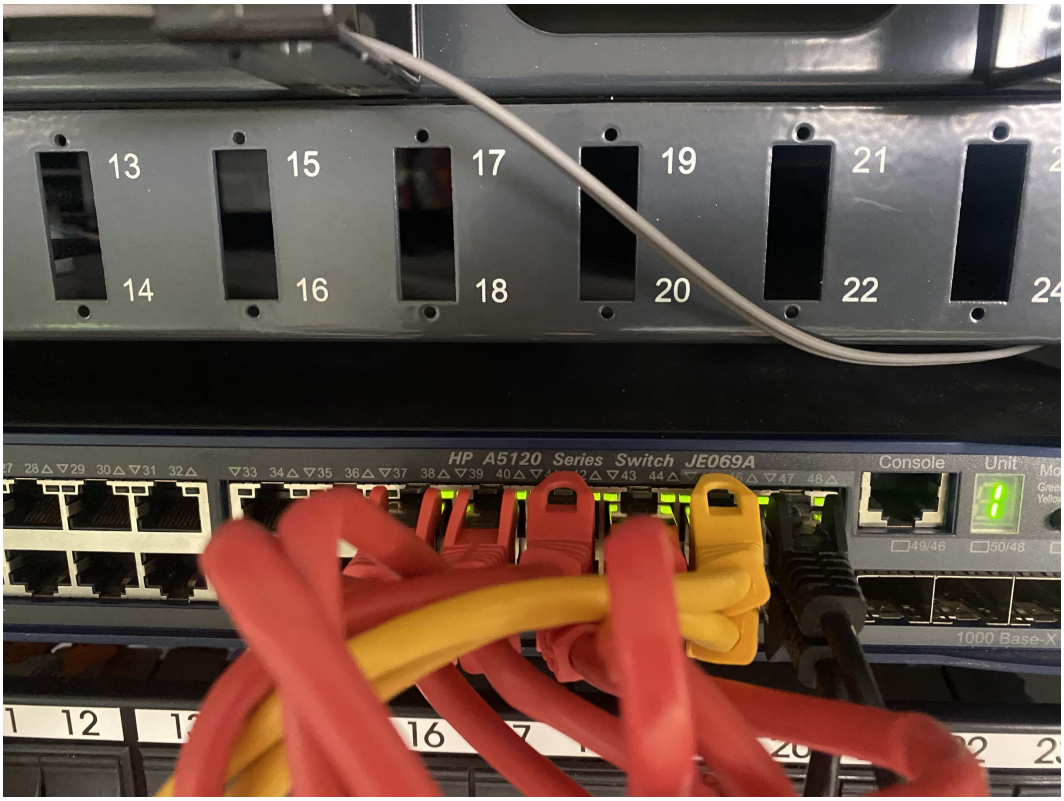
L'armoire rack principale de la section SIO concentre l'ensemble des équipements cœur de l'infrastructure. On y trouve notamment les deux commutateurs Aruba formant le cœur de réseau L3 en configuration redondante (présentés comme un équipement unique sur le schéma réseau par souci de simplification), les trois nœuds Proxmox constituant un cluster de virtualisation, les trois nœuds Nutanix/IDRAC formant également un cluster, ainsi que le serveur NAS de la section. L'ensemble est alimenté par des onduleurs Eaton assurant la continuité de service en cas de coupure électrique.

Armoire rack principale — cœur de réseau, cluster Proxmox, cluster Nutanix, NAS, onduleurs Eaton

Une baie de brassage (patch panel) est un panneau passif monté en rack qui centralise les arrivées de câbles réseau provenant des prises murales d'une salle. Elle ne traite pas les données mais permet de relier physiquement, via des cordons de brassage, les prises de salle aux ports du commutateur correspondant. Cela facilite grandement la gestion du câblage : modifier une connexion revient à déplacer un cordon sur la baie, sans toucher aux câbles installés dans les murs. La baie ci-dessous est celle utilisée pour la salle 406 ; les cordons rouges et jaunes redirigent le flux vers les commutateurs Huawei présents dans les armoires de la salle.

Baie de brassage salle 406 — redirection du flux vers les commutateurs Huawei des armoires

Le commutateur visible ci-dessous est un **HP A5120 Series Switch JE069A**. Dans le cadre de la réalisation de la maquette réseau, j'ai relevé les numéros de ports de ce commutateur en consultant les connexions physiques afin d'intégrer les correspondances exactes entre ports physiques et connexions logiques. Ce processus a été répété pour chaque commutateur du réseau SIO afin d'obtenir une maquette fidèle à l'infrastructure réelle.



HP A5120 Series Switch JE069A — salle 406, relevé des numéros de ports pour la maquette

Chacun des trois nœuds Proxmox dispose de quatre interfaces réseau : deux ports Ethernet 1 Gbit/s et deux ports fibre 10 Gbit/s. Ces interfaces sont regroupées en deux bonds Linux (bond0 et bond1), chacun contenant un port Ethernet et un port fibre, fonctionnant en mode *failover*. Le port fibre est utilisé en priorité pour bénéficier du haut débit 10 Gbit/s ; si ce port tombe, le relais est automatiquement passé au port Ethernet 1 Gbit/s sans interruption de service.

La redondance est également assurée au niveau du cœur de réseau : sur chaque bond, un port est branché sur le premier commutateur Aruba et l'autre sur le second, de sorte que la panne d'un Aruba n'entraîne aucune interruption. Sur les deux ports fibre, l'un est dédié à l'accès SSH d'administration (VLAN 99) et l'autre est configuré en lien trunk autorisant l'ensemble des VLANs du réseau SIO. Cette architecture garantit à la fois la haute disponibilité (HA) et le débit nécessaire aux migrations de machines virtuelles entre nœuds du cluster.

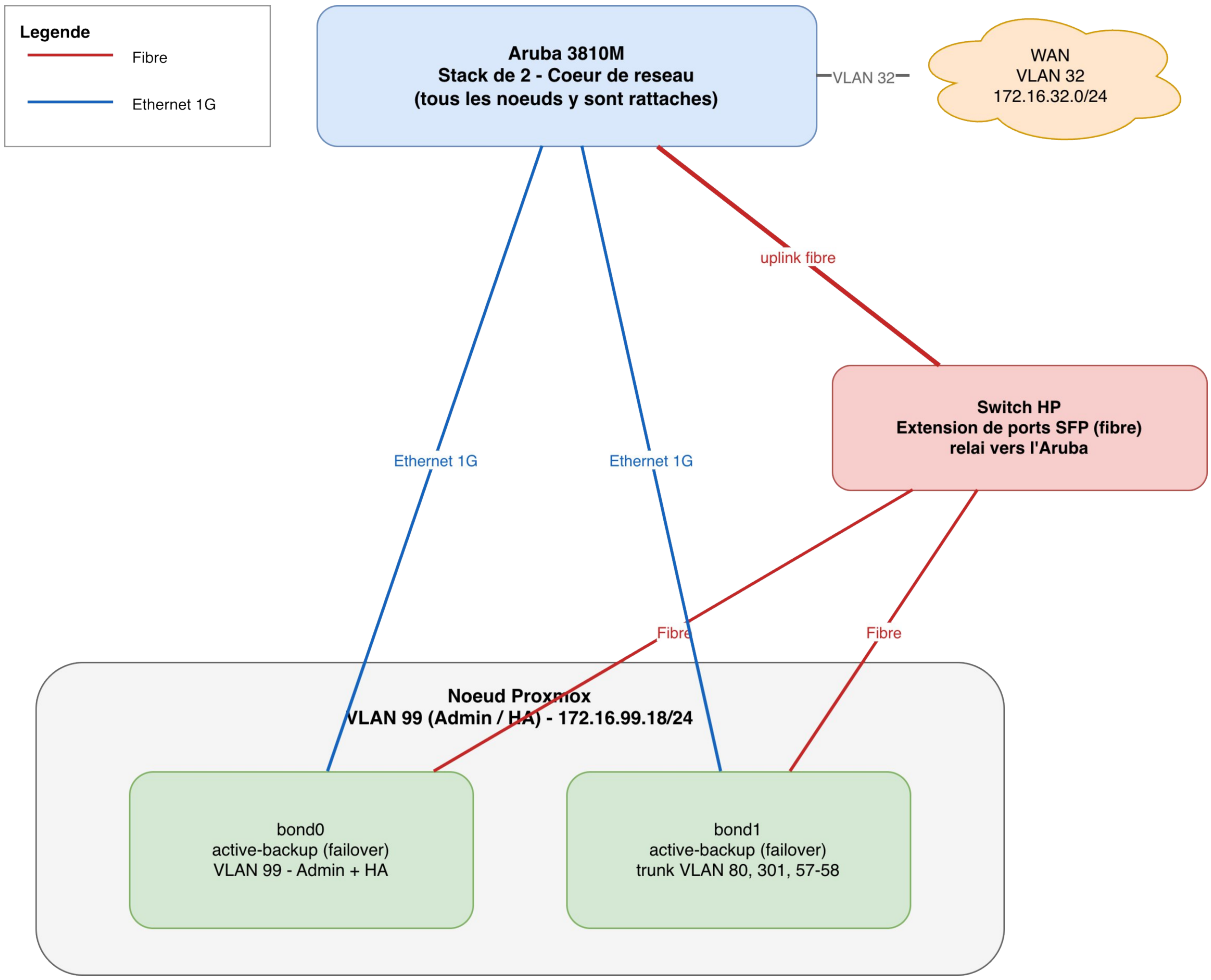


Schéma de principe du bonding Linux en mode failover sur les nœuds Proxmox

2. Mission 1 : Déploiement Master + FOG

La deuxième et troisième semaine de stage sont consacrées au déploiement d'images système sur les postes de travail des salles de la section SIO. L'objectif est de mettre en place une solution permettant de déployer rapidement et de manière reproductible des images Windows et Linux sur l'ensemble du parc, salle par salle.

2.1. Comparatif des solutions de déploiement

Avant de retenir FOG Project comme solution, plusieurs alternatives ont été étudiées selon plusieurs critères : licence, dépendance cloud, interface d'administration, support du multicast, et adéquation avec un environnement scolaire on-premises.

Solution	Licence	Interface web	Multicast	Dépendance cloud	Retenu
FOG Project	GPL (gratuit)	Oui	Oui	Non (on-premises)	✔ Oui
Clonezilla	GPL (gratuit)	Non	Oui (SE uniquement)	Non	Non
MDT (Microsoft)	Gratuit (abandonné)	Non	Non	Non (fin de vie jan. 2026)	Non
WDS (Windows)	Inclus Windows Server	Non	Oui	Non (nécessite licence)	Non
Windows Autopilot	Payant (M365)	Oui (Azure)	Non	Oui (Azure requis)	Non

Le choix s'est porté sur **FOG Project** pour plusieurs raisons : c'est une solution libre sous licence GPL, sans dépendance cloud, déployable sur un serveur Linux léger hébergé sur Proxmox. Elle dispose d'une interface web centralisant la gestion du parc, supporte le déploiement multicast (déploiement simultané sur plusieurs machines) et est éprouvée en environnement scolaire. MDT, son principal concurrent gratuit, a été mis hors service par Microsoft en janvier 2026 et n'est plus maintenu.

2.2. Création du serveur FOG

Le serveur FOG est déployé sous forme de machine virtuelle sur l'hyperviseur Proxmox de l'établissement. La version installée est FOG Project 1.5.10.1826, sur une distribution Debian 13 « Trixie ». Son interface réseau est rattachée au VLAN 32 (WAN SIO), ce qui lui permet d'être joignable par les postes des salles lors du boot PXE.

Le plan d'adressage retenu pour le serveur est le suivant : adresse IP 172.16.32.250/24, passerelle 172.16.32.254. Une fois installé et configuré, le serveur a été validé sur les trois fonctionnalités clés du processus de déploiement : enregistrement d'hôtes dans la console d'administration, capture d'image à partir d'un poste master, et déploiement d'une image sur un hôte préalablement enregistré.

2.3. Préparation et validation du master Windows 11 en VM

Avant tout déploiement sur des postes physiques, le processus complet a été validé sur deux machines virtuelles VirtualBox. L'objectif est d'isoler les éventuels problèmes liés à la chaîne FOG (capture, multicast, boot PXE, redéploiement) avant d'intervenir sur les postes en salle. La VM master, nommée Windows11Master, héberge une installation Windows 11 Professionnel sur laquelle ont été installés les logiciels destinés au poste type (Firefox, VLC, LibreOffice, 7-Zip, Foxit PDF Reader, déployés via Ninite). Une seconde VM, Windows11Deploy, est créée avec un disque dur vierge de 80 Go, sans système installé : elle sert de cible pour valider le déploiement.

Une fois le master prêt, un sysprep en mode OOBE est exécuté avant capture. Ce point est revenu sur lors du déploiement (voir difficultés ci-après). La capture est ensuite lancée depuis l'interface FOG, suivie du déploiement sur la VM Windows11Deploy. La procédure détaillée de chaque étape est consignée en annexe, le corps du rapport se concentrant ici sur le déroulement global et les difficultés rencontrées.

Les captures d'écran illustrant le déroulement nominal de la validation en VM (image capturée dans la console FOG, hôtes enregistrés, capture Partclone en cours, écran OOBE et session étudiant après déploiement) sont reportées en annexe A1, afin de ne conserver dans le corps du rapport que l'analyse du déroulement et des difficultés rencontrées.

Difficultés rencontrées

Première tentative sur Windows 10 boucle de démarrage au déploiement. Le processus a d'abord été tenté avec une VM master sous Windows 10. La capture s'est correctement effectuée, mais le poste cible restait coincé dans une boucle de démarrage signalant un échec et invitant à appliquer une mise à jour. L'hypothèse retenue est l'activation du démarrage rapide (« fast boot ») sur le master : Windows écrit alors un état du noyau et des pilotes au moment de l'extinction, état qui devient incohérent après redéploiement sur un matériel virtuel ré-initialisé. Le passage à Windows 11 avec désactivation explicite du démarrage rapide avant capture a permis de résoudre le problème.

Incompatibilité UEFI / VirtualBox lors du boot PXE. FOG gère mal le boot PXE en mode UEFI sur les VM VirtualBox, ce qui est une limitation connue de la combinaison des deux outils. Le contournement appliqué a consisté à désactiver l'EFI dans la configuration de la VM pour les opérations FOG (enregistrement, capture, déploiement), puis à le réactiver une fois le déploiement terminé. Aucun effet de bord n'a été observé après réactivation de l'UEFI sur la VM déployée.

Débit de capture anormalement bas. Lors de la première capture du master Windows 11, Partclone a affiché un débit de l'ordre de 2,9 Mo/min, sans cause apparente, alors que les captures nominales tournent autour de 2 Go/s. L'arrêt et la relance de la tâche de capture ont suffi à revenir à un débit normal. L'hypothèse la plus probable est une dégradation du mode d'accès disque côté VirtualBox (mécanismes de cache ou bascule en mode de compatibilité), réinitialisée par la relance de la session FOG. Le problème n'a pas réapparu sur les captures suivantes.

Utilité limitée du sysprep dans le contexte salle. Le sysprep en mode OOBE/Generalize a été appliqué sur le master Windows 11. Conformément à son

fonctionnement, le poste déployé a redémarré sur l'assistant de configuration initiale (OOBE), exigeant la création d'un nouveau compte utilisateur, ce qui n'est pas adapté au contexte des salles, où tous les postes doivent partager exactement le même compte « étudiant » avec les mêmes logiciels installés. Pour le déploiement réel en salle, le sysprep en mode OOBE ne sera pas conservé, afin que l'image déployée embarque directement le compte étudiant prêt à l'emploi.

2.4. Interface dédiée sur les commutateurs pour FOG

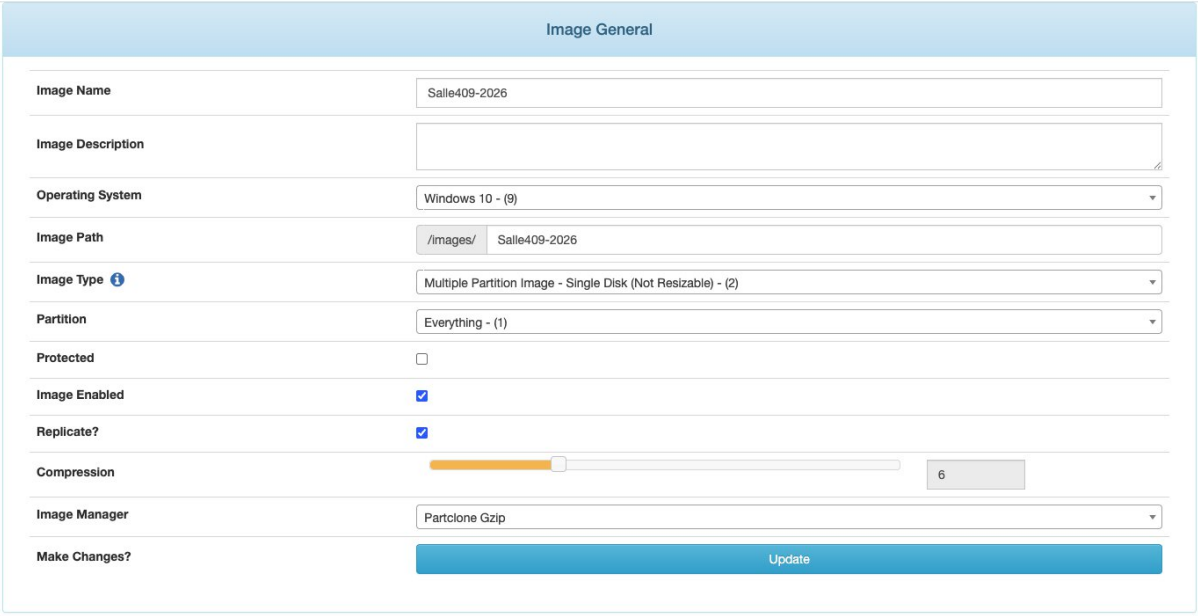
Afin d'isoler le trafic de déploiement FOG du reste du réseau pédagogique, une interface dédiée est configurée sur chaque commutateur HP des armoires de salle. En salle 409, les postes sont branchés sur les ports 19 et 20 du commutateur HP (sorties Proxmox), rattachés au VLAN 32 pour permettre aux postes en boot PXE de joindre directement le serveur FOG (172.16.32.250) sans interférer avec les autres VLANs.

Pour le déploiement simultané d'une rangée de la salle 409 (4 postes), un commutateur d'appoint Cisco SG 300-10 est utilisé. Les 4 postes de la rangée sont raccordés sur les ports d'accès du SG 300-10, ainsi que les sorties WANSIO correspondants au port 19 et 20 du serveur FOG. Cette topologie permet de bénéficier du mode multicast de FOG : une seule image est transmise par le serveur et distribuée simultanément aux 4 postes, ce qui réduit la durée totale du déploiement et la charge réseau côté serveur. Le déroulement prévu est de procéder rangée par rangée (4 × 4 postes), jusqu'à couvrir l'ensemble de la salle.

2.5. Déploiement salle 409 — Préparation du master Windows 11 de production

Pour le déploiement en salle 409, une image Windows 11 existante sur le serveur FOG de production du lycée a servi de base. Cette image a été mise à jour avec les dernières versions disponibles de Windows 11. Parallèlement, un nouveau serveur FOG de production a été instancié sur Proxmox en suivant la même procédure que celle employée lors des tests en VM. Le détail de l'installation est consigné en annexe.

Lors de la configuration de l'image dans l'interface FOG, le type **Image à partitions multiples sur disque seul** a été sélectionné, avec le gestionnaire **Gzip**. Ce choix est imposé par l'architecture du disque Windows 11 : le système crée plusieurs partitions réservées (partition EFI, partition de récupération, partition principale NTFS) qu'il est indispensable de capturer intégralement. Sans ce paramètre, Partclone interrompt la procédure de capture car il ne gère pas un disque multi-partitions en mode image simple.



Paramètres utilisés — Type d'image : Image à partitions multiples sur disque seul ; Gestionnaire d'image : Gzip

Une fois les paramètres validés, la tâche de capture est créée depuis l'interface FOG et la machine master est bootée en PXE. Partclone prend en charge la capture partition par partition et transfère l'image compressée vers le stockage du serveur FOG.

2.6. Déploiement physique — Salle 409

Après validation du processus en environnement virtuel, le déploiement est réalisé sur les postes physiques de la salle 409. L'organisation rigoureuse est ici primordiale : une nomenclature précise est adoptée pour les groupes FOG, avec la convention **PC409-TableX** (TableA, TableB, TableC, TableD), correspondant aux 4 rangées de 4 postes de la salle. Chaque groupe contient exactement les 4 hôtes de sa rangée, ce qui permet de retrouver rapidement un poste, de relancer un déploiement ciblé et d'éviter toute confusion lors des opérations.

La procédure adoptée pour chaque rangée est la suivante : les 4 postes sont branchés sur le commutateur Cisco SG 300-10 (ports 1 à 4), dont le port 8 est relié au serveur FOG (port 19 du commutateur HP). Un *Quick Inventory* est effectué sur chaque poste au boot PXE pour les enregistrer dans FOG, puis ils sont ajoutés à leur groupe respectif. Depuis l'onglet **Group** → **Basic Tasks** → **Multicast**, la tâche est lancée, puis les 4 postes sont allumés en PXE simultanément. FOG distribue l'image en multicast à toute la rangée en une seule passe. Le débit observé est de **14,61 GB/min**.

Switch Cisco SG 300-10 utilisé pour le déploiement multicast — port 8 relié au serveur FOG, ports 1–4 aux postes de la rangée

Les 4 postes d'une rangée de la salle 409 en cours de boot PXE — menu FOG affiché sur chaque écran

Déploiement multicast en cours — Partclone s'exécute simultanément sur les 4 postes de la rangée

Bureau Windows 11 après déploiement sur un poste de la salle 409 — compte etudiant, logiciels préinstallés visibles (Wireshark, Packet Tracer, Chrome, Firefox, LibreOffice, VirtualBox, etc.)

Une difficulté a été rencontrée lors de l'enchaînement des sessions multicast entre deux rangées successives : les postes de la deuxième rangée bootaient en PXE et

rejoignaient le menu FOG, mais le déploiement ne démarrait pas — Partclone s'affichait sans barre de progression. La cause est un processus **udp-sender** résiduel côté serveur FOG, qui reste actif quelques secondes après la fin de la session précédente et entre en conflit avec la nouvelle session multicast. Il s'agit d'un problème connu de FOG. La solution est de redémarrer le service **FOGMulticastManager** entre chaque rangée :

```
sudo systemctl restart FOGMulticastManager
```

Après ce redémarrage, la session multicast suivante démarre normalement. Cette opération est à répéter systématiquement entre chaque rangée pour garantir un déploiement fluide. Le processus a été répété pour chaque rangée de la salle 409, couvrant l'ensemble des 16 postes.

Intégration au domaine btssio.lan. Une tentative d'intégration automatique au domaine Active Directory *btssio.lan* via la fonctionnalité FOG (paramètres Active Directory du groupe, champ Organizational Unit en format LDAP) a été tentée après le déploiement. Cette tentative n'a pas abouti : aucun poste n'a rejoint le domaine automatiquement. L'hypothèse retenue est une problématique de routage inter-VLAN : le serveur FOG étant dans le VLAN 32 (WAN SIO) et le serveur AD dans le VLAN 20 (Production), les requêtes d'intégration au domaine émises par le FOG Client après déploiement sont vraisemblablement bloquées par les règles de filtrage ACL ou par le pare-feu. L'intégration au domaine a donc été réalisée manuellement poste par poste, ce qui n'affecte pas le résultat final mais élimine le bénéfice d'automatisation apporté par cette fonctionnalité de FOG.

2.7. Déploiement salle 406 — Master Debian 13

Pour la salle 406, le master est une installation **Debian 13 avec l'environnement de bureau Cinnamon**, configuré avec les outils nécessaires aux travaux pratiques de la section : Firefox, LibreOffice, VSCode, Notepadqq, VirtualBox, Cisco Packet Tracer 8.2.1, Wireshark, PuTTY et Minicom. Une fois le master validé, il est capturé via FOG puis déployé poste par poste sur l'ensemble de la salle 406, en mode **unicast** (tâche individuelle par poste), le multicast ayant présenté des instabilités sur ce déploiement. L'unicast s'est avéré aussi efficace pour cette configuration.

Difficulté 1 — Conflit de déploiement entre NVMe et HDD. Les postes de la salle 406 disposent de deux disques : un SSD NVMe (système) et un disque dur SATA 2 To (stockage). Lors des premières tentatives de capture et de déploiement, FOG détectait les deux disques et écrivait l'image simultanément sur le NVMe et sur le HDD, provoquant des conflits au niveau du chargeur d'amorçage et rendant le poste non démarrable sur NVMe. La solution a consisté à débrancher physiquement le câble d'alimentation et le câble de données du disque dur SATA avant chaque opération FOG (capture et déploiement), afin que FOG ne voie que le NVMe. Les disques ont ensuite été rebranchés après déploiement.

Difficulté 2 — Ordre de boot BIOS après déploiement. Après déploiement, les postes bootaient en priorité sur le PXE plutôt que sur le NVMe. Les postes n'étant plus dans le VLAN 32 après déploiement, ils ne recevaient aucune réponse du serveur FOG et restaient bloqués sur le menu PXE, obligeant l'utilisateur à appuyer sur Échap

à chaque démarrage. La solution a été de reconfigurer l'ordre de boot dans le BIOS de chaque poste (accès avec le mot de passe fourni par le tuteur de stage) pour placer le NVMe en première position, devant le boot PXE.

2.8. Déploiement salles 407 et 408

Le master des salles 407 et 408 devait intégrer un ensemble d'outils orientés développement, définis en concertation avec Mme Ferreira pour répondre aux besoins pédagogiques spécifiques de ces salles. Les logiciels déployés sont les suivants :

- **Android Studio** et le **SDK Android** (développement mobile)
- **Eclipse** (environnement de développement Java)
- **Looping** (modélisation de bases de données)
- **pgAdmin** (administration PostgreSQL) et un **client SQL** (MariaDB et PostgreSQL)
- **Visual Studio** (développement .NET / applications métiers)
- **Visual Studio Code** ou un éditeur de code léger équivalent
- **PuTTY** (client SSH/Telnet)
- Un logiciel de virtualisation (**VirtualBox**, ou alternative équivalente)

La constitution finale du master a été validée avec Mme Ferreira, puis l'image a été déployée sur les postes des salles 407 et 408. Le processus FOG étant désormais éprouvé sur les salles déployées précédemment, cette dernière vague s'est déroulée sans difficulté particulière, achevant le déploiement de l'ensemble des salles de la section dans le nouveau domaine AD (sio.lan).

3. Mission 2 : Mise en place d'une solution de supervision

La deuxième mission du stage consiste à mettre en place une solution de supervision pour l'infrastructure de la section SIO. L'objectif est de disposer d'un tableau de bord centralisé affichant en temps réel les métriques des équipements supervisés (hyperviseurs, switches, pare-feu), ainsi que les alertes de sécurité générées par le pare-feu. La solution doit être libre, déployable on-premises sur le cluster Proxmox, et produire un rendu visuel adapté à un affichage permanent sur écran.

3.1. Comparatif des solutions de supervision

Avant de retenir une solution, plusieurs outils de supervision open source ont été étudiés selon les critères suivants : licence, qualité de la visualisation, support des équipements réseau (SNMP), gestion des logs, complexité de déploiement et adéquation avec un affichage temps réel sur écran.

Solution	Licence	Visualisation	SNMP	Complexité	Retenu
Zabbix	AGPL v3	Très bonne avec Grafana	Oui (natif)	Faible (tout-en-un)	✔ **Oui**
Prometheus + Grafana	Apache 2.0 / AGPL v3	Très bonne	Oui (snmp_exporter)	Modérée (stack modulaire)	Non
Nagios / Centreon	GPL (core)	Faible (interface old)	Oui (plugins)	Élevée	Non
Checkmk Raw	GPL (éd. Raw)	Bonne	Oui (natif)	Modérée	Non
LibreNMS	GPL v3	Bonne (orientée réseau)	Excellent (natif)	Faible	Non

3.2. Justification du choix — Zabbix + Grafana

Le choix s'est porté sur **Zabbix** comme moteur de collecte, associé à **Grafana** pour la visualisation, pour deux raisons principales. Premièrement, Zabbix est une solution tout-en-un : un seul composant gère la collecte, le stockage et les alertes, ce qui simplifie considérablement le déploiement et la maintenance par rapport à la stack Prometheus qui nécessite d'empiler plusieurs composants distincts (Prometheus, Alertmanager, exporters). Dans un contexte scolaire, cette simplicité architecturale garantit la pérennité de la solution après la fin du stage.

Deuxièmement, l'équipe pédagogique de la section est familiarisée avec Zabbix, ce qui facilite la reprise en main de la solution à l'issue du stage. Grafana, connecté à Zabbix via le plugin datasource officiel, apportera la qualité de visualisation nécessaire à un affichage temps réel sur écran. Prometheus reste néanmoins une alternative pertinente, notamment grâce à son écosystème d'exporters riche (en particulier le

proxmox_exporter), mais la priorité donnée à la maintenabilité et à la continuité de service a guidé ce choix.

3.3. Architecture de supervision envisagée

L'ensemble de la stack sera déployé sur une machine virtuelle hébergée sur le cluster Proxmox. La collecte des métriques reposera sur le serveur Zabbix et ses agents, ainsi que sur le module SNMP natif pour les équipements réseau. Grafana centralisera l'affichage en interrogeant Zabbix comme datasource via le plugin officiel, dans un dashboard unifié.

Composant	Rôle	Cible supervisée
Zabbix Server	Collecte, stockage des métriques, alertes et escalades	Tous les agents et sources SNMP
Grafana	Affichage des dashboards temps réel (métriques)	Zabbix Server (source de données via plugin officiel)
Zabbix Agent	Agent léger installé sur les hôtes Linux supervisés	Nœuds Proxmox, serveurs AD, VM supervision
Zabbix SNMP	Supervision des équipements réseau via SNMP (natif dans Zabbix)	Switches HP / Aruba, switches de salle
Templates Zabbix officiels	Templates pré-configurés pour Linux, SNMP, Proxmox	Tous les équipements supervisés

3.4. Éléments supervisés

Nœuds Proxmox et infrastructure de virtualisation. Les trois nœuds du cluster Proxmox seront supervisés via l'agent Zabbix et les templates officiels Linux et Proxmox (CPU, RAM, disque, trafic réseau, état des VMs, stockage partagé, état du cluster). Un dashboard dédié permettra de visualiser la charge de chaque nœud et de détecter rapidement une ressource saturée.

Switches et équipements réseau. Les switches HP et Aruba (cœur de réseau L3) ainsi que les switches de salle seront interrogés via le module SNMP natif de Zabbix. Les métriques collectées incluront le trafic entrant et sortant par interface (en bits/s), le taux d'erreurs, et l'état des ports (up/down) ; notamment ceux des liens trunk entre les différents équipements. Ces données permettront d'afficher des graphes de consommation réseau en temps réel et de détecter des anomalies de trafic.

Disponibilité des services réseau. Zabbix permettra également de vérifier régulièrement la joignabilité des passerelles VLANs, du serveur DNS/AD et des services HTTP internes via des checks ICMP et HTTP natifs, afin de détecter rapidement une panne de service sans attendre un signalement manuel.

3.5 Protocoles exploités par les services

La stack Zabbix + Grafana repose sur plusieurs protocoles distincts selon les flux concernés. La communication entre le serveur Zabbix et les agents installés sur les

hôtes Linux (nœuds Proxmox, serveur FOG) utilise le protocole propriétaire Zabbix sur TCP port 10050 (mode actif) ou 10051 (mode passif).

La supervision des équipements réseau — switches HP et Aruba — passe par SNMP (Simple Network Management Protocol) sur UDP port 161, en version SNMPv2c ou SNMPv3, permettant d'interroger les MIBs (base de données explicitant les configuration réseaux pouvant être exposées via SNMP) pour récupérer le trafic par interface, l'état des ports et la charge des équipements.

Choix de la version du protocole SNMP

Trois versions du protocole coexistent. SNMPv1 est aujourd'hui obsolète et n'est pas retenue. SNMPv2c reste très répandue : elle apporte des mécanismes de récupération de données plus efficaces (requêtes GetBulk), mais sa sécurité repose sur une simple « community string » transmise en clair sur le réseau, équivalent d'un mot de passe non chiffré susceptible d'être intercepté. SNMPv3 ajoute un modèle de sécurité complet : authentification de l'émetteur des requêtes (HMAC-SHA) et chiffrement du contenu des échanges (AES). C'est donc SNMPv3 qui est retenu pour la supervision du réseau SIO, le niveau de sécurité « authPriv » (authentification et chiffrement) étant privilégié.

Ce choix a été validé en conditions réelles sur un commutateur Cisco Catalyst 2960 et un routeur Cisco 1921, tous deux compatibles SNMPv3. La configuration retenue définit une vue SNMP (périmètre des OID autorisés en lecture), un groupe et un utilisateur dédié à Zabbix, avec authentification SHA et chiffrement AES. La validation a été effectuée côté serveur Zabbix à l'aide de la commande « snmpget » avant l'intégration dans l'interface, afin d'isoler un éventuel problème de configuration de l'équipement d'un problème de paramétrage de Zabbix.

Accessibilité des MIB sur les équipements

Une MIB (Management Information Base) est l'arborescence normalisée des informations qu'un équipement expose via SNMP ; chaque valeur y est identifiée par un OID (Object Identifier) unique. Tous les équipements n'implémentent toutefois pas l'intégralité des MIB : il est donc indispensable de vérifier, pour chaque élément à superviser, que les OID réellement utiles sont accessibles avant de s'appuyer dessus.

Cette vérification a révélé des différences concrètes selon le matériel. Sur le Catalyst 2960 testé (modèle d'entrée de gamme doté d'une version d'IOS ancienne), les métriques reposant sur la MIB standard IF-MIB—trafic par interface, état des ports (up/down), nombre d'erreurs—remontent correctement, de même que la charge processeur et la mémoire. En revanche, certains OID présents dans le modèle générique « Cisco IOS » ne sont pas implémentés sur cet équipement : les capteurs de température et les OID d'inventaire matériel (ENTITY-MIB, numéro de série, modèle) renvoient une réponse « No Such Instance ». Ces remontées non supportées n'empêchent pas la supervision : elles sont simplement marquées comme indisponibles côté Zabbix et peuvent être désactivées pour alléger l'affichage. Cette cartographie des MIB par type d'équipement est à reconduire pour les switches HP et Aruba, dont les MIB diffèrent de celles des équipements Cisco.

Supervision active et supervision passive

Zabbix distingue deux modes de collecte. En supervision active, c'est l'agent ou l'équipement supervisé qui prend l'initiative d'envoyer ses données au serveur :

l'agent récupère la liste des métriques à collecter, puis transmet les valeurs de lui-même. En supervision passive, c'est le serveur Zabbix qui interroge périodiquement l'équipement et attend sa réponse. Pour les agents Zabbix installés sur les hôtes Linux, le mode actif (l'agent émet vers le port 10051 du serveur) allège la charge du serveur et facilite la traversée des pare-feux ; le mode passif (le serveur interroge le port 10050 de l'agent) offre un contrôle centralisé du rythme de collecte. La supervision SNMP des équipements réseau relève par nature d'une logique passive : le serveur interroge régulièrement chaque équipement par requêtes SNMP « Get ».

Notion de trap SNMP

Le mode passif repose sur de l'interrogation périodique (polling) : un événement survenu entre deux interrogations n'est détecté qu'au cycle suivant. Le trap SNMP répond à cette limite : il s'agit d'une notification émise spontanément par l'équipement vers le collecteur (port UDP 162) dès qu'un événement significatif se produit—par exemple le passage d'un port en état down (linkDown), un redémarrage de l'équipement ou le franchissement d'un seuil. Le trap apporte ainsi une détection quasi immédiate des incidents, là où le polling introduit un délai égal à l'intervalle de collecte. Dans l'architecture envisagée, le polling assure la collecte régulière des métriques (graphes de trafic, taux d'utilisation) tandis que les traps pourront être activés sur les événements critiques, comme la chute d'un lien trunk, pour une remontée d'alerte sans délai. La réception des traps côté Zabbix nécessite le service « snmptrapd » en complément du serveur.

Zabbix stocke l'ensemble de ses métriques dans une base de données relationnelle (MySQL/MariaDB) sur TCP port 3306. Grafana interroge Zabbix via son API JSON-RPC en HTTP/HTTPS sur le port 443, grâce au plugin datasource officiel.

Enfin, l'interface web Grafana est accessible depuis un navigateur sur TCP port 3000 par défaut.

3.6. Dashboards réalisés

La stack **Zabbix + Grafana** a été déployée et trois dashboards ont été réalisés, couvrant chacun un périmètre fonctionnel distinct. Construits sur une maquette prototype (cluster Proxmox de test, switches Cisco, VM de services), ils constituent une démonstration de faisabilité (POC) destinée à être transposée sur l'infrastructure de production.

Dashboard Proxmox (réalisé, validé). Ce dashboard offre une vue synthétique du cluster de virtualisation. Il affiche le nombre de machines virtuelles actives, ainsi que les métriques CPU, RAM et stockage pour chacun des trois nœuds. Un panneau dédié présente les températures mesurées en entrée et en sortie (bezel) de chaque nœud, permettant de détecter une montée en température anormale. Un tableau d'alertes Zabbix, filtré sur les événements critiques, complète l'ensemble : chute d'un nœud, surcharge de stockage, inaccessibilité d'un IDRAC.

Capture du dashboard Proxmox (vue d'ensemble du cluster

Dashboard Réseau (réalisé, validé). Ce dashboard présente l'état de chaque lien d'agrégation du cœur de réseau et des commutateurs de distribution de manière exhaustive et individualisée (non agrégée), grâce à des panneaux Stat affichant le nom de chaque interface avec une couleur de fond pilotée par son état. Le débit de chaque agrégation est tracé sous forme de séries temporelles, de même que les débits réseau des nœuds Proxmox. Un panneau de synthèse en tête de dashboard affiche d'un coup d'œil l'état des port-channels et du lien WAN. La corrélation entre la chute d'un lien membre et l'effondrement du débit de l'agrégation correspondante a été vérifiée en conditions réelles en débranchant physiquement des câbles.

Capture du dashboard Réseau (état des agrégations et débits)

Dashboard Services (réalisé, validé). Ce dashboard supervise la disponibilité des services critiques sous forme d'indicateurs UP/DOWN reposant sur des tests TCP : FOG, Grafana, Proxmox et Zabbix sur le périmètre prototype. Il affiche également l'uptime de chaque service avec un code couleur (rouge en deçà de 5 minutes de fonctionnement, signalant un redémarrage récent, orange jusqu'à une heure, vert au-delà), le nombre de baux DHCP attribués et libres rapportés à la taille totale du pool, ainsi que l'utilisation disque et la consommation mémoire des VM de supervision, afin de détecter une saturation avant qu'elle n'affecte le service. Sur l'infrastructure de production, ce périmètre sera étendu aux services réels de la section (Active Directory, DNS, DHCP, contrôleur Wi-Fi).*)*

Capture du dashboard Services (états UP/DOWN, baux DHCP, uptime, disques)

3.7. Réalisation et difficultés rencontrées

Le déploiement de la stack Zabbix + Grafana a nécessité de résoudre plusieurs difficultés techniques, documentées ci-après.

Wizard d'installation Zabbix bloqué. Lors de la première installation du serveur Zabbix, l'assistant de configuration web s'est interrompu à l'étape de génération du fichier de configuration, sans message d'erreur explicite. Après analyse, la cause identifiée était un problème de permissions sur le répertoire cible : le processus web n'avait pas les droits d'écriture nécessaires pour créer le fichier. La solution a consisté à créer manuellement le fichier `zabbix.conf.php` avec les paramètres de connexion à la base de données, en contournant l'assistant. L'interface Zabbix s'est ensuite chargée normalement.

Locales manquantes sur la VM. L'interface Zabbix affichait des erreurs de rendu et des caractères incorrects liés à l'absence des locales système sur la VM Debian. La génération des locales manquantes (`locale-gen` et configuration via `dpkg-reconfigure locales`) a résolu le problème.

Cache du datasource Grafana masquant les tags. Lors de la construction des dashboards, certains tags et groupes d'hôtes Zabbix n'apparaissaient pas dans les menus de sélection Grafana alors qu'ils étaient correctement configurés côté Zabbix. La cause était un cache de métadonnées conservé par le plugin datasource Grafana-Zabbix. La solution a consisté à invalider manuellement ce cache depuis les paramètres du datasource, ce qui a immédiatement rendu les tags disponibles.

Choix du type de visualisation pour l'état des ports. La première tentative d'affichage de l'état des ports réseau (up/down) utilisait un panneau de type Time series, inadapté à des données discrètes binaires : le rendu était illisible et ne permettait pas de visualiser clairement les transitions d'état. Le remplacement par un panneau de type State timeline, conçu pour ce cas d'usage, a permis d'obtenir un affichage clair et immédiatement exploitable, avec une couleur par état et une lecture chronologique des changements de liens.

Privilege Separation des tokens API Proxmox. L'intégration de la sonde Proxmox via le template « Proxmox VE by HTTP » repose sur un token API. Les premières tentatives ne remontaient aucune donnée malgré l'attribution du rôle PVEAuditor à l'utilisateur dédié. La cause est le mécanisme de « Privilege Separation » activé par défaut sur les tokens Proxmox : un token possède ses propres permissions, indépendantes de celles de l'utilisateur, et n'hérite pas automatiquement de ses droits. Les permissions effectives sont calculées en croisant celles de l'utilisateur et celles du token. La résolution a consisté à attribuer explicitement le rôle PVEAuditor également au token lui-même, et non au seul utilisateur.

Chemins ACL invalides sur Proxmox VE 9. La documentation du template indique d'accorder les droits sur les chemins /cluster/resources et /cluster/status. Or ces chemins, qui sont des points d'accès de l'API, ne sont pas des chemins ACL valides : Proxmox VE 9 renvoie une erreur « invalid path » à leur saisie. Les chemins ACL réellement valides se limitent à /, /nodes, /vms, /storage, /pool, /sdn et /access. Les premières configurations, limitées à /nodes, /vms et /storage, remontaient correctement les métriques propres à chaque nœud, mais le cluster apparaissait réduit à un seul nœud et le décompte des machines virtuelles restait vide. En effet, les données à l'échelle du cluster sont exposées par les points d'accès /cluster/status

et /cluster/resources, dont la lecture exige le privilège Sys.Audit au niveau de la racine. La configuration finale attribue donc le rôle PVEAuditor sur / avec propagation, ce rôle incluant les privilèges d'audit nécessaires (Sys.Audit, VM.Audit, Datastore.Audit) sur l'ensemble de l'arborescence.

Affichage « No data » dans Grafana. Certains panneaux Grafana affichaient « No data » alors que les métriques étaient correctement collectées côté Zabbix. La cause était une fenêtre temporelle du dashboard plus courte que l'intervalle de collecte de l'item : plusieurs métriques étant collectées à un intervalle d'une minute ou plus, aucun point n'est disponible sur une fenêtre de quelques minutes. L'élargissement de la fenêtre d'affichage à au moins une heure a résolu le problème.

Supervision de la température CPU sur nœuds bi-processeurs. Les nœuds Proxmox étant équipés de deux processeurs, la commande sensors renvoie deux valeurs de température distinctes pour le cœur 0, une par socket. Un item Zabbix de type « Zabbix agent » ne pouvant recevoir qu'une valeur numérique unique, la moyenne des deux températures a dû être calculée directement dans la commande du UserParameter, sans recourir à un script externe. La procédure complète figure en annexe A9.

Disparition des jauges de disque après inactivité. Les jauges d'utilisation des espaces de stockage des nœuds disparaissaient de Grafana après plusieurs heures, alors même que la valeur restait visible sur une fenêtre temporelle large. La cause était l'étape de prétraitement « Discard unchanged with heartbeat » appliquée par le template Proxmox aux items de stockage : Zabbix ne réenregistre une valeur que si elle change, ou au plus toutes les douze heures (heartbeat). Une utilisation disque stable ne générant aucun nouveau point pendant cet intervalle, Grafana ne disposait plus de donnée récente à afficher. La résolution a consisté à réduire le heartbeat à dix minutes sur le prototype d'item de la règle de découverte concernée. Un point d'attention a été relevé : la modification d'un prototype ne met pas à jour les items déjà découverts ; il a fallu supprimer ces derniers pour qu'ils soient recréés avec le nouveau paramètre.

3.8. Passage en production

Après validation de la chaîne de supervision sur un environnement prototype, l'ensemble a été reconstruit sur des machines virtuelles dédiées à la production. Cette étape distingue la phase de maquettage de l'exploitation réelle : les adresses IP de production sont fixes et propres à cet environnement, indépendantes de celles utilisées durant les essais (mentionnées dans les annexes prototype).

Les serveurs de supervision de production disposent des adresses suivantes : le serveur Zabbix sur 172.16.32.107 et le serveur Grafana sur 172.16.32.106. Les agents Zabbix déployés sur les nœuds Proxmox ont été repointés vers cette nouvelle adresse de serveur (directives Server et ServerActive), et les dashboards ont été réimportés depuis l'environnement prototype au format JSON avant reconfiguration de la datasource Zabbix sur la nouvelle instance Grafana.

La supervision réseau a complété cette mise en production : les commutateurs de la section, soit le cœur de réseau Aruba 3810M et les commutateurs Comware, ont été intégrés via SNMPv3 au niveau de sécurité authPriv. La configuration détaillée de

chaque équipement, ainsi que le choix des templates Zabbix associés, figurent en annexe A3.7.

À la suite de cette mise en place, mon tuteur a demandé que les machines virtuelles de supervision soient déplacées dans le VLAN dédié à la production. Ce changement de VLAN s'est accompagné d'un ré-adressage : le serveur Zabbix est désormais joignable sur 172.16.20.7 et le serveur Grafana sur 172.16.20.6. Les directives Server et ServerActive des agents, ainsi que la configuration de la datasource Grafana, ont été mises à jour en conséquence.

La supervision a enfin été étendue aux deux serveurs Active Directory redondants de l'établissement (172.16.20.21 et 172.16.20.22). Ces hôtes ont été ajoutés à Zabbix via le template « Windows by Zabbix agent », qui remonte notamment l'occupation de leurs disques — affichée sur le dashboard des services — ainsi que l'état des services Windows critiques, en particulier DHCPServer et DNS, surveillés au moyen de la clé service.info. La remontée du nombre de baux DHCP, qui ne repose sur aucun template standard, fait l'objet d'une procédure d'adaptation spécifique détaillée en annexe A6.

3.9. Cartographie réseau et affichage en kiosque

Pour offrir une vue synthétique de l'état du réseau, une cartographie a été réalisée à l'aide des cartes natives de Zabbix (Network maps). Les équipements — cœur de réseau Aruba, commutateurs Comware de chaque salle, nœuds Proxmox et serveurs Active Directory — y sont représentés regroupés par salle et reliés par des liens reflétant la topologie physique. L'état de chaque équipement est déduit automatiquement de ses déclencheurs ; celui de chaque lien est associé au déclencheur d'accessibilité ICMP de l'équipement de destination, de sorte qu'un lien passe au rouge dès que l'hôte correspondant cesse de répondre.

Cette carte, comme les dashboards Grafana, est destinée à être affichée en continu sur un écran de supervision. L'affichage retenu repose sur une rotation automatique des onglets du navigateur : chaque ressource, dashboards Grafana et carte Zabbix, est ouverte dans son propre onglet, le navigateur les faisant défiler à intervalle régulier. La carte Zabbix est appelée directement via l'URL du compte invité en lecture seule, en mode plein écran (paramètre kiosk), ce qui évite toute authentification manuelle. Le compte invité a été activé et restreint en lecture seule sur les groupes d'hôtes de production, et la carte partagée au groupe Guests.

En complément, les interfaces Zabbix et Grafana ont été basculées en HTTPS afin de chiffrer le trafic de supervision, identifiants du compte invité compris. En l'absence d'accès à une autorité de certification publique sur ce réseau interne, une autorité de certification interne (PKI) a été mise en place : une CA racine signe les certificats des deux serveurs, émis avec leur adresse IP déclarée en SubjectAltName comme l'exigent les navigateurs récents. Le certificat racine de la CA doit être importé dans les navigateurs clients pour que la connexion soit reconnue comme sûre. La procédure complète de création de la PKI et d'import du certificat figure en annexe A10.

Difficulté — intégration en iframe abandonnée. L'affichage de la carte Zabbix dans une iframe au sein d'un dashboard Grafana, d'abord envisagé pour réunir l'ensemble dans une playlist unique, s'est heurté à plusieurs restrictions de sécurité des

navigateurs en contexte inter-sites. Le cookie de session du compte invité, soumis à la politique SameSite, n'était pas transmis à l'intérieur de l'iframe, ce qui empêchait l'authentification ; par ailleurs, certaines ressources de l'interface Zabbix (polices, scripts) étaient bloquées par la politique CORS. Le passage en HTTPS et l'ajout d'en-têtes adaptés (Content-Security-Policy avec frame-ancestors, suppression de X-Frame-Options) ont levé une partie des blocages, mais pas la restriction de cookie inter-sites. La solution de rotation d'onglets, où chaque page est chargée dans son contexte natif, contourne entièrement le problème et a donc été préférée.

4. Mission annexe — Préparation du parcours de formation Stormshield

4.1. Contexte

Les 28 et 29 mai 2026, M. Demoulière Quentin, tuteur pédagogique et formateur certifié, a animé un parcours de formation Stormshield destiné aux étudiants de deuxième année de BTS SIO, en préparation à leurs examens de certification. J'ai été associé à la préparation de cette session sur deux journées : une journée complète de formation théorique le jeudi, et une demi-journée de mise en place de la salle le vendredi matin.

4.2. Tâches réalisées

La préparation physique de la salle 409 a consisté à installer les pare-feux **Stormshield SN210** en armoire rack et à effectuer le câblage requis selon le guide officiel remis aux formateurs. Chaque équipement a été vérifié et remis à l'état usine avant utilisation. Lorsque la procédure de réinitialisation n'avait pas été effectuée par les étudiants précédents, un reset du mot de passe était nécessaire : celui-ci s'effectuait en interrompant le démarrage du firmware du pare-feu à l'aide de commandes spécifiques saisies en console série, permettant d'accéder au système sans authentification et de réinitialiser les identifiants. Lorsque les équipements étaient accessibles via l'interface d'administration, la réinitialisation était réalisée directement depuis cette interface.

J'ai également assisté à la mise en route d'un pare-feu **Stormshield SN310** neuf : installation des mises à jour firmware et enregistrement de la licence auprès du portail Stormshield.

4.3. Apports techniques

La journée du jeudi a été consacrée à une formation théorique dispensée par M. Demoulière sur le fonctionnement des pare-feux Stormshield. Les concepts abordés incluaient le **bridging** (mode transparent), les **règles de filtrage**, la **translation d'adresses (NAT)** et la **prévention d'intrusion (IPS)**. Cette formation m'a permis d'élargir mes connaissances en sécurité réseau au-delà du périmètre de mes missions principales.

Conclusion

Ce stage de cinq semaines, réalisé au sein même du lycée Paul-Louis Courier, m'a permis de mener à bien deux missions techniques sur une infrastructure de production, dans des conditions réelles d'exploitation.

La première mission a abouti au déploiement, via la solution FOG, de masters systèmes sur l'ensemble des salles de la section : Windows 11 en multicast pour la salle 409, Debian 13 pour la salle 406, et un master orienté développement pour les salles 407 et 408. Au-delà de la mise en œuvre, cette mission m'a confronté à des contraintes concrètes — compatibilité PXE/UEFI, préparation des masters par sysprep, gestion du multicast — dont la résolution a consolidé ma maîtrise du déploiement d'images à grande échelle.

La seconde mission a conduit à la mise en production d'une solution de supervision complète reposant sur Zabbix et Grafana. J'y ai supervisé les équipements réseau multi-constructeurs en SNMPv3 (Aruba, HPE/Comware, Ubiquiti), les nœuds du cluster Proxmox via son API, ainsi que les serveurs et services de la section, en y ajoutant des métriques personnalisées (baux DHCP, température processeur) au moyen de scripts et de UserParameters. J'ai complété ce dispositif par une cartographie réseau, un affichage en kiosque sur écran, et la sécurisation des interfaces en HTTPS au travers d'une autorité de certification interne.

En marge de ces missions, j'ai participé à la préparation d'un parcours de formation Stormshield, qui a élargi mes connaissances en sécurité réseau. De manière transversale, ce stage m'a surtout appris à découvrir, comprendre et documenter une infrastructure existante de bout en bout, une démarche que j'ai concrétisée par la rédaction d'une documentation technique à destination des futurs membres de la section.

Plusieurs pistes restent ouvertes pour prolonger ce travail, notamment l'extension du périmètre de supervision aux journaux des pare-feu et la généralisation de l'autorité de certification interne à l'ensemble des postes clients. Ce stage a confirmé mon intérêt pour le domaine SISR et constituera un socle solide pour mon stage de deuxième année.

Annexes : Procédures détaillées

Dans cette section, diverses annexes sont présentes afin de démontrer une démarche précise, des tests, des méthodologies ou autres sujets qui n'ont pas leur place au cœur du rapport.

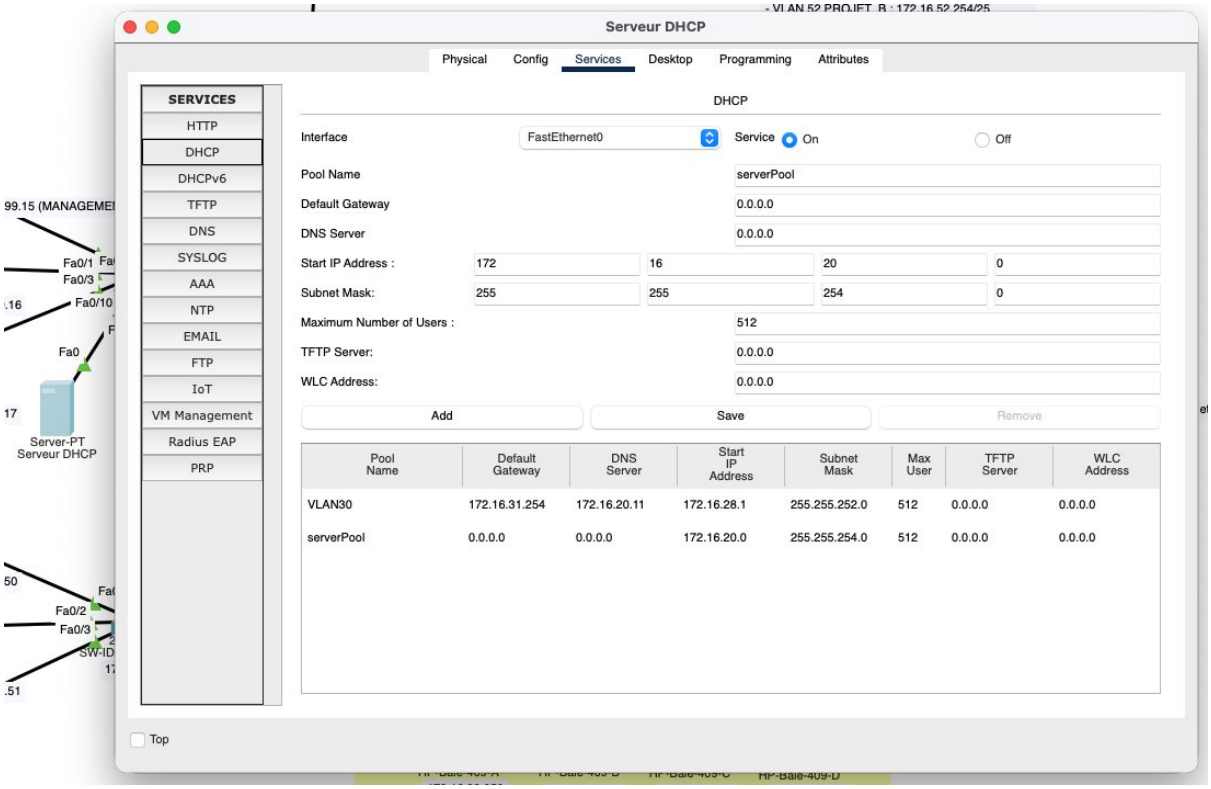
A0. Documentation de la maquette Packet Tracer du réseau BTS SIO de l'établissement

Cette documentation couvre différents tests concernant la maquette Packet Tracer du réseau SIO du lycée Paul-Louis Courier, dans le contexte d'un stage de 1ère année de BTS SIO.

A0.1 Attribution dynamique des informations réseaux aux équipements (Serveur DHCP)

Notre serveur DHCP (qui est en réalité un serveur AD/DHCP/DNS dans l'infrastructure réelle du lycée) est configuré de telle sorte à fournir des adresses IP de façon dynamique dans le VLAN30 (PEDAGOGIQUE) avec différents paramètres convenu ci-dessous :

Plage d'adresse utilisable : de 172.16.28.1 à 172.16.29.254 (les adresses IP seront distribuées dans cette plage)
Masque de sous-réseau : 255.255.252.0
Passerelle par défaut : 172.16.31.254 (qui correspond à l'interface SVI de notre commutateur L3 Cœur de Réseau, pour le VLAN30 PEDAGOGIQUE)
DNS : 172.16.20.11 (serveur AD/DNS du réseau)



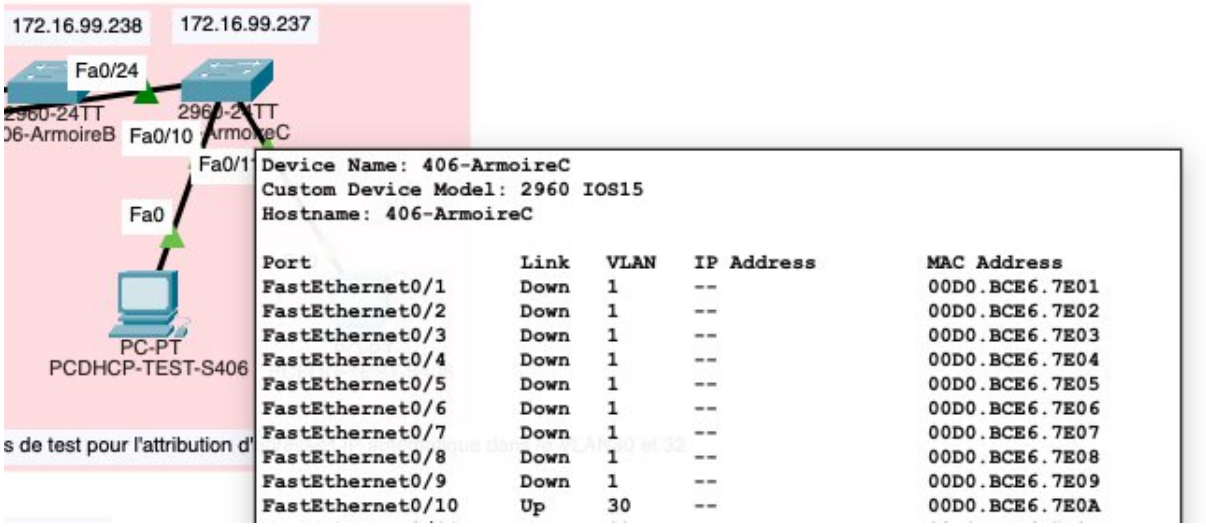
Capture d'écran montrant le pool VLAN30, et le service DHCP qui est actif sur le serveur.

Ensuite, nous devons établir un relai DHCP au niveau du commutateur Cœur de Réseau, afin que les requêtes DHCP puissent traverser le commutateur sur un VLAN spécifique, ici, ce seront les flux DHCP correspondant au VLAN30 qui seront diffusés, la commande Cisco IOS à renseigner ici est la suivante :

```
SW-CORE>en
SW-CORE#conf t
SW-CORE#(config)int vlan 30
SW-CORE#(config-if)ip helper-address 172.16.20.11
```

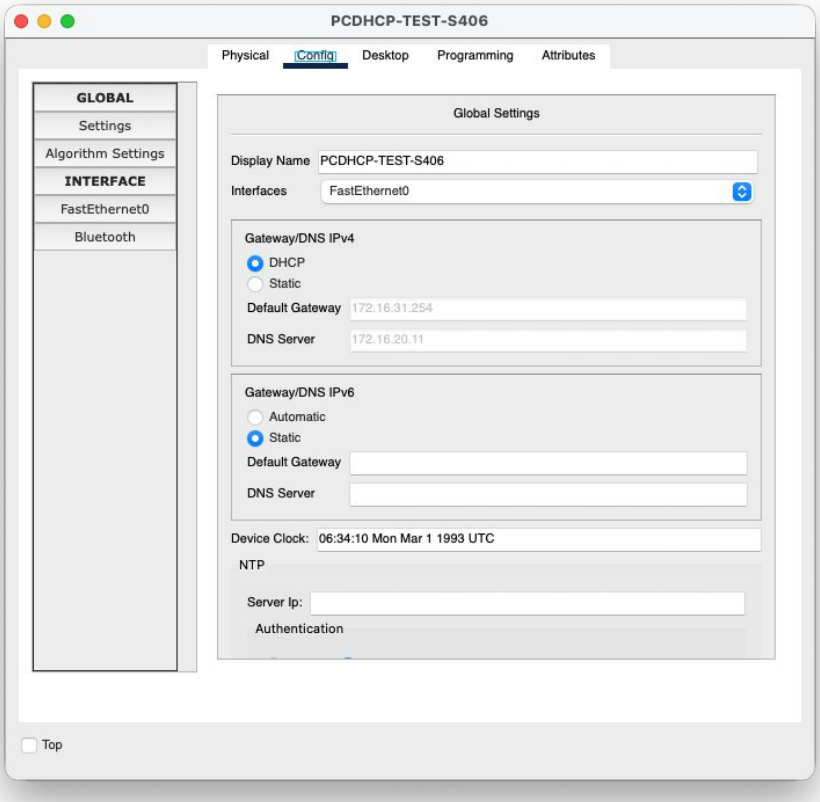
Afin de vérifier si l'attribution dynamique d'adresse fonctionne, nous positionnons un poste sur un des commutateur de la salle 406 (sur la maquette), sur un port attribué au VLAN30 au préalable avec les commandes ci-dessous :

```
406-ArmoireC>en
406-ArmoireC#conf t
406-ArmoireC#(config)int fa0/10
406-ArmoireC#(config-int)sw mode access
406-ArmoireC#(config-int)sw access vlan 30
```



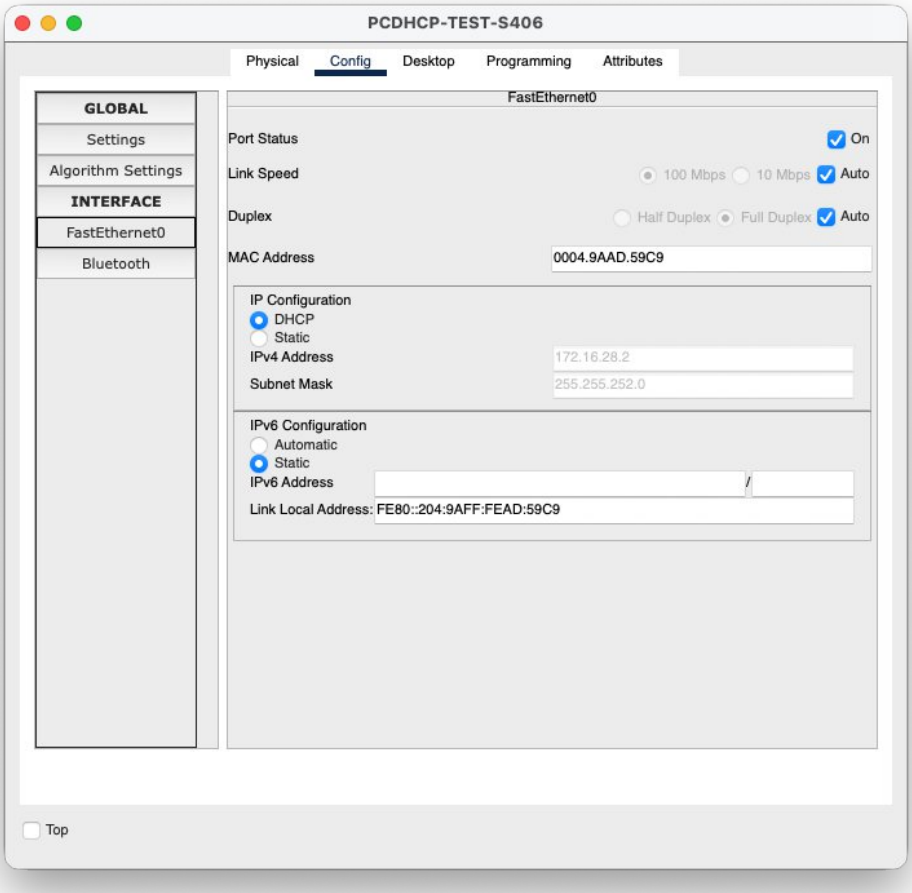
Capture d'écran montrant l'affectation réussie du port FastEthernet0/10 au VLAN30.

Ensuite, nous sélectionnons l'attribution automatique (DHCP) dans les paramètres de l'ordinateur de test :



Capture d'écran montrant que la configuration réseau reçue par le serveur DHCP est correcte (passerelle et serveur DNS)

Nous pouvons voir que nous recevons les bonnes informations, la passerelle est correcte, le serveur DNS également, vérifions ensuite que nous obtenons les bonnes informations réseau au niveau de notre interface réseau :

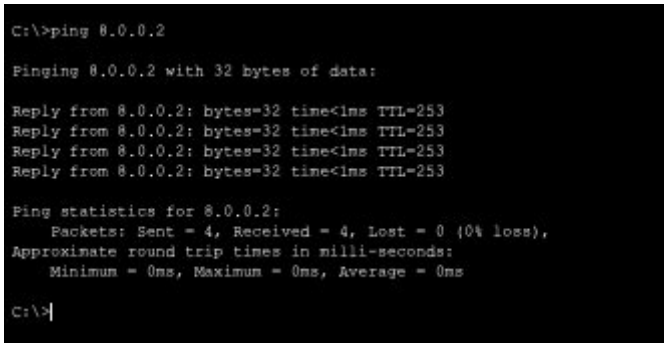


Capture d'écran montrant que la configuration réseau reçue par le serveur DHCP est correcte (adresse IP et masque de sous réseau)

Notre adresse IP est dans la plage définie au préalable, le masque est également correct.

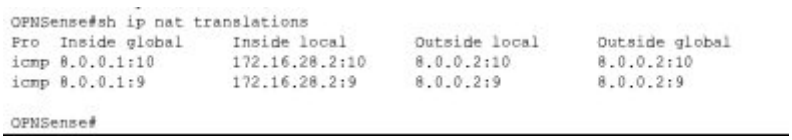
A0.2 Translation d'adresses et accès à internet

Afin de vérifier si la translation d'adresse fonctionne convenablement, nous essayons de contacter l'adresse 8.0.0.2 (qui correspond au routeur Internet sur la maquette) avec un poste présent dans un VLAN, essayons par exemple avec l'ordinateur que nous avons utilisé pour la vérification de l'attribution d'adresses dynamiquement. Il est donc positionné dans le VLAN30 PEDAGOGIQUE.



Capture d'écran montrant un ping réussi vers le routeur Internet (8.0.0.2)

Comme nous pouvons le voir, notre ordinateur atteint bien le routeur Internet, vérifions à présent la table de translation d'adresses sur notre pare-feu OPNSense avec la commande **'sh ip nat translations' :**



Capture d'écran montrant la translation d'adresses au niveau des ports (PAT)

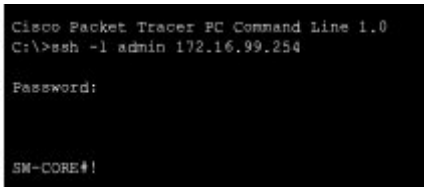
Nous pouvons voir que notre poste à l'adresse 172.16.28.2 est translatée au niveau des ports, confirmant bien le fonctionnement du PAT sur le pare-feu.

A0.3 Accès SSH aux équipements intermédiaires

Depuis un ordinateur présent dans le VLAN99 (ADMINISTRATION), nous essayons d'initialiser une connexion SSH vers un équipement distant, prenons par exemple le poste PC-Admin409, qui est branché sur une interface du commutateur HP-Baie-409-A présent dans l'armoire A de la salle 409.

Nous essayons de nous connecter en SSH au commutateur L3 Coeur de Réseau, pour ce faire, nous utilisons la commande suivante sur l'ordinateur :

'ssh -l admin 172.16.99.254' : ici, nous renseignons le login associé au commutateur, ce login est défini lors de la création des accès SSH sur le commutateur, en parallèle d'un mot de passe qui nous sera demandé juste après (etudiant_007):



Capture d'écran montrant une connexion à distance réussie en SSH

Nous voici alors connecté au commutateur coeur de réseau (dont le hostname est SW-CORE sur la maquette) via SSH, démontrant un accès à distance sécurisé.

A1. Démarche détaillées de la mission 1 (FOG)

A1.1. Préparation du master Windows 11 en VM

Création d'une VM VirtualBox avec EFI activé, support TPM 2.0 (requis par Windows 11) et carte réseau attachée au VLAN 32 pour atteindre le serveur FOG. Installation d'une Windows 11 Professionnel à partir d'une ISO officielle. La création de compte Microsoft à l'OOBE a été contournée en déconnectant temporairement la carte réseau et en utilisant la commande OOBE\BYPASSNRO (Shift+F10 puis cmd) afin de créer un compte local.

Installation des logiciels via Ninite (Firefox, VLC, LibreOffice, 7-Zip, Foxit PDF Reader) pour disposer d'un master conforme au poste type. Puis, avant capture, désactivation du démarrage rapide : Panneau de configuration → Options d'alimentation → Choisir l'action des boutons d'alimentation → Modifier des paramètres actuellement non disponibles → décocher « Activer le démarrage rapide ». Cette étape est critique : c'est l'omission de cette désactivation qui avait causé la boucle de démarrage observée sur la première tentative Windows 10.

Exécution du sysprep depuis C:\Windows\System32\Sysprep\sysprep.exe avec les options OOBE + Generalize + Shutdown. La VM s'éteint d'elle-même à la fin du sysprep, dans un état prêt à être capturé. Aucun redémarrage n'est effectué entre le sysprep et la capture, afin de ne pas réinitialiser le contexte préparé par sysprep.

A1.2. Capture d'image avec FOG

Désactivation de l'EFI dans la configuration de la VM (Système → Carte mère → décocher « Activer EFI ») pour permettre le boot PXE sous VirtualBox, contournement de l'incompatibilité connue avec FOG. Démarrage de la VM en mode boot réseau : la console FOG iPXE s'affiche et propose les options d'enregistrement, capture, déploiement.

Côté serveur FOG (interface web) : enregistrement de l'hôte (Windows11Master), création d'une définition d'image associée, puis création d'une tâche de type « Capture ». Au prochain boot PXE de la VM master, la capture démarre automatiquement. L'outil Partclone affiche en temps réel le débit, le volume traité et le temps estimé. Le débit nominal observé est de l'ordre de 2 Go/s sur le stockage Proxmox.

A1.3. Déploiement FOG sur poste vierge

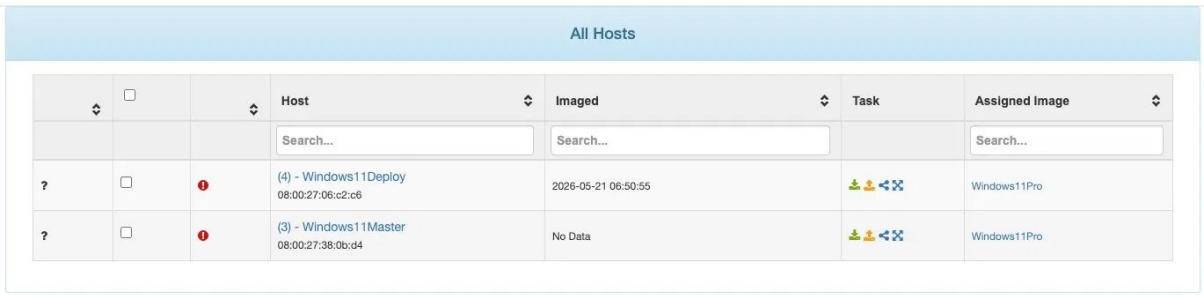
Création d'une VM cible Windows11Deploy avec un disque dur vierge de 80 Go, EFI désactivé pour le boot PXE, carte réseau sur le VLAN 32. Enregistrement de l'hôte dans FOG (host registration au boot PXE), puis création d'une tâche « Déploiement » à partir de l'image préalablement capturée.

Au boot PXE suivant, le déploiement démarre automatiquement. À la fin du déploiement, la VM redémarre : Windows 11 affiche l'écran OOBE (effet attendu du sysprep en mode OOBE), un compte utilisateur local est créé pour la validation. Après finalisation de l'OOBE, la VM est éteinte, l'EFI est réactivé dans la configuration VirtualBox, et la VM redémarre normalement sans effet de bord. La session ouverte

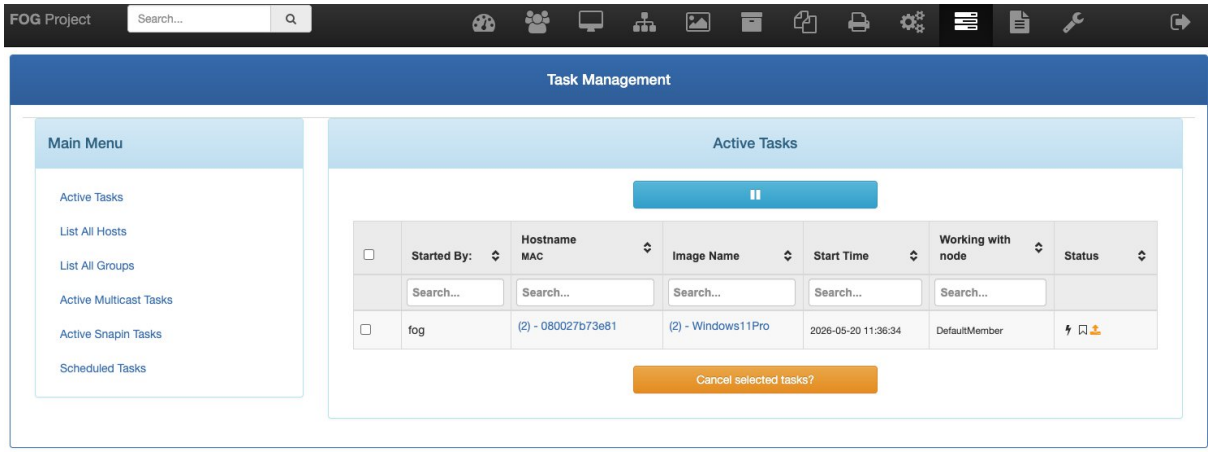
sur le compte créé montre l'ensemble des logiciels installés sur le master (Firefox, VLC, LibreOffice, 7-Zip, Foxit), confirmant l'intégrité du déploiement.

Captures de validation du processus en VM

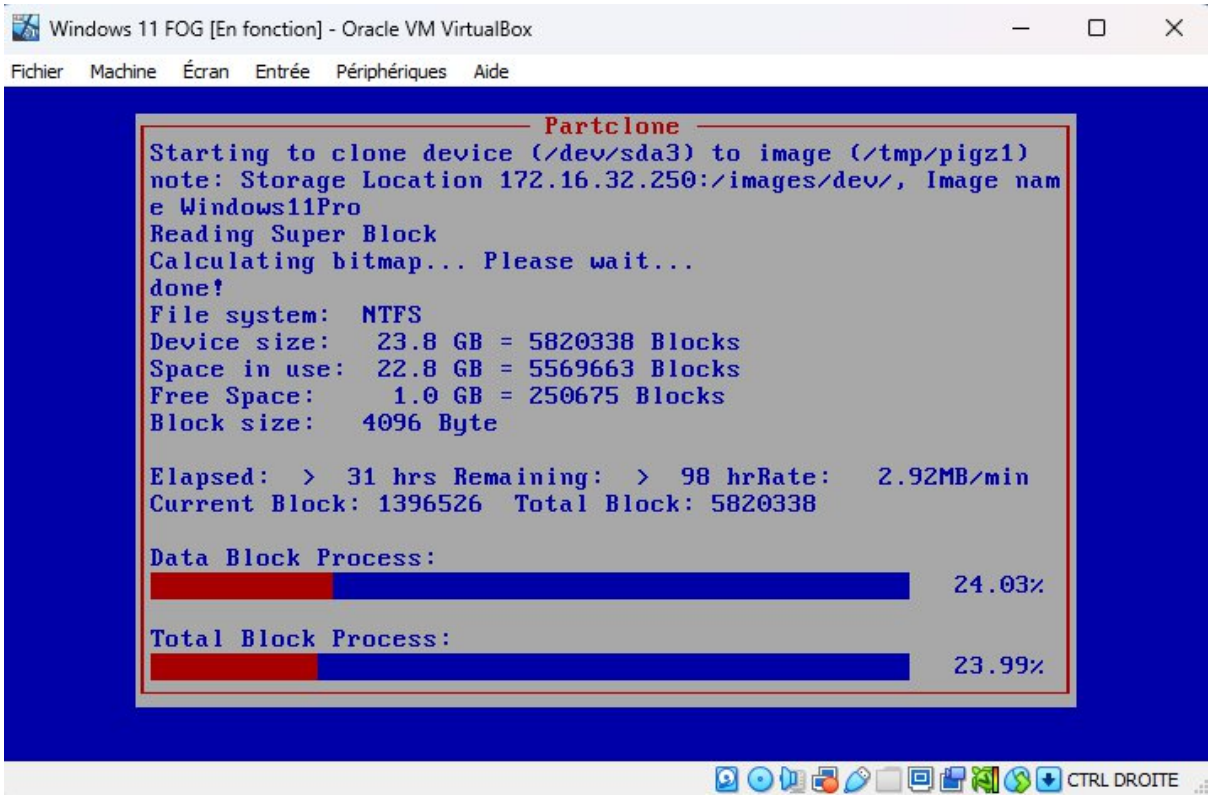
Image Windows 11 Pro capturée — visible dans la liste des images de la console FOG.



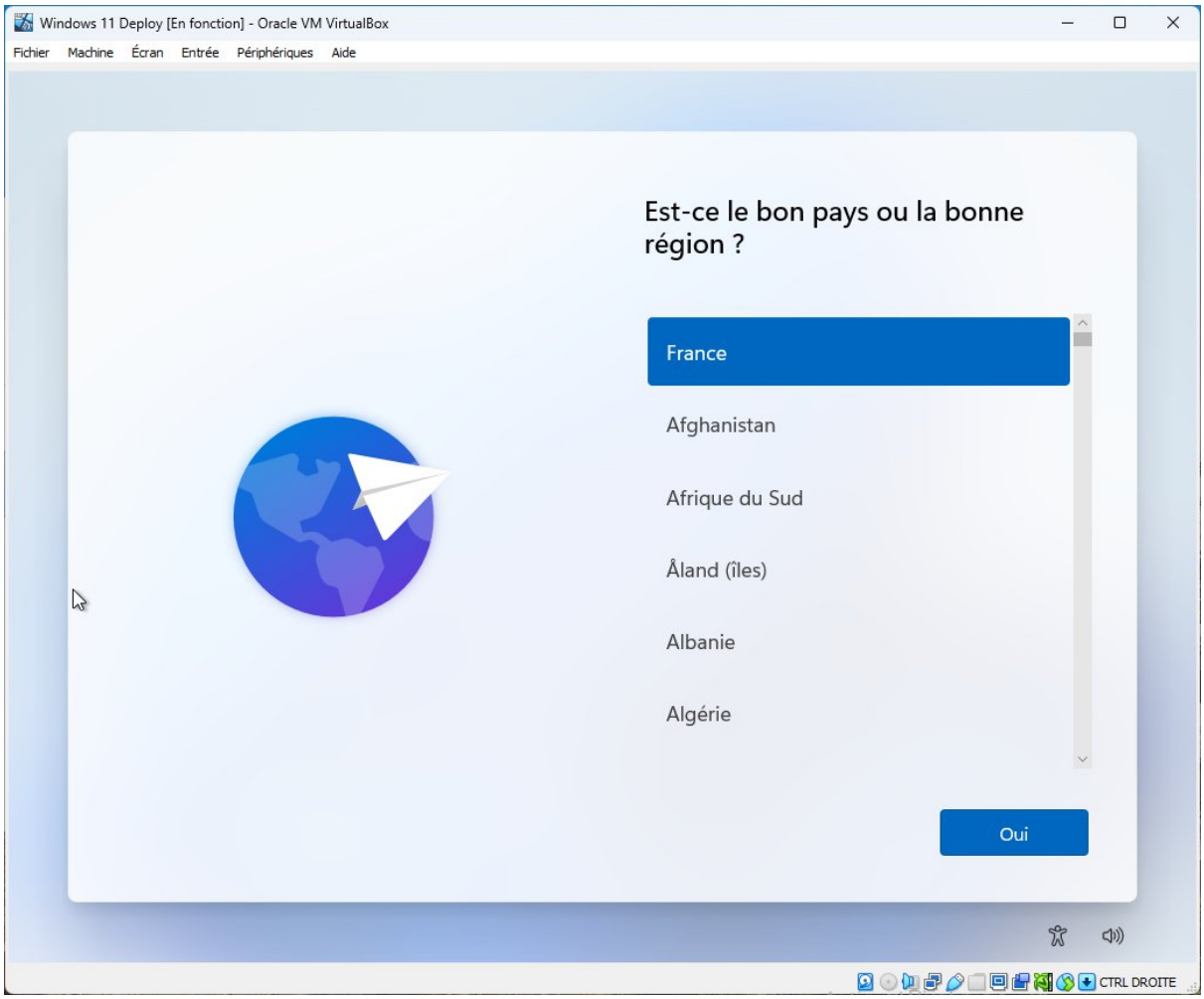
Hôtes enregistrés dans la console FOG — Windows11Master (source de l'image) et Windows11Deploy (cible vierge de validation).



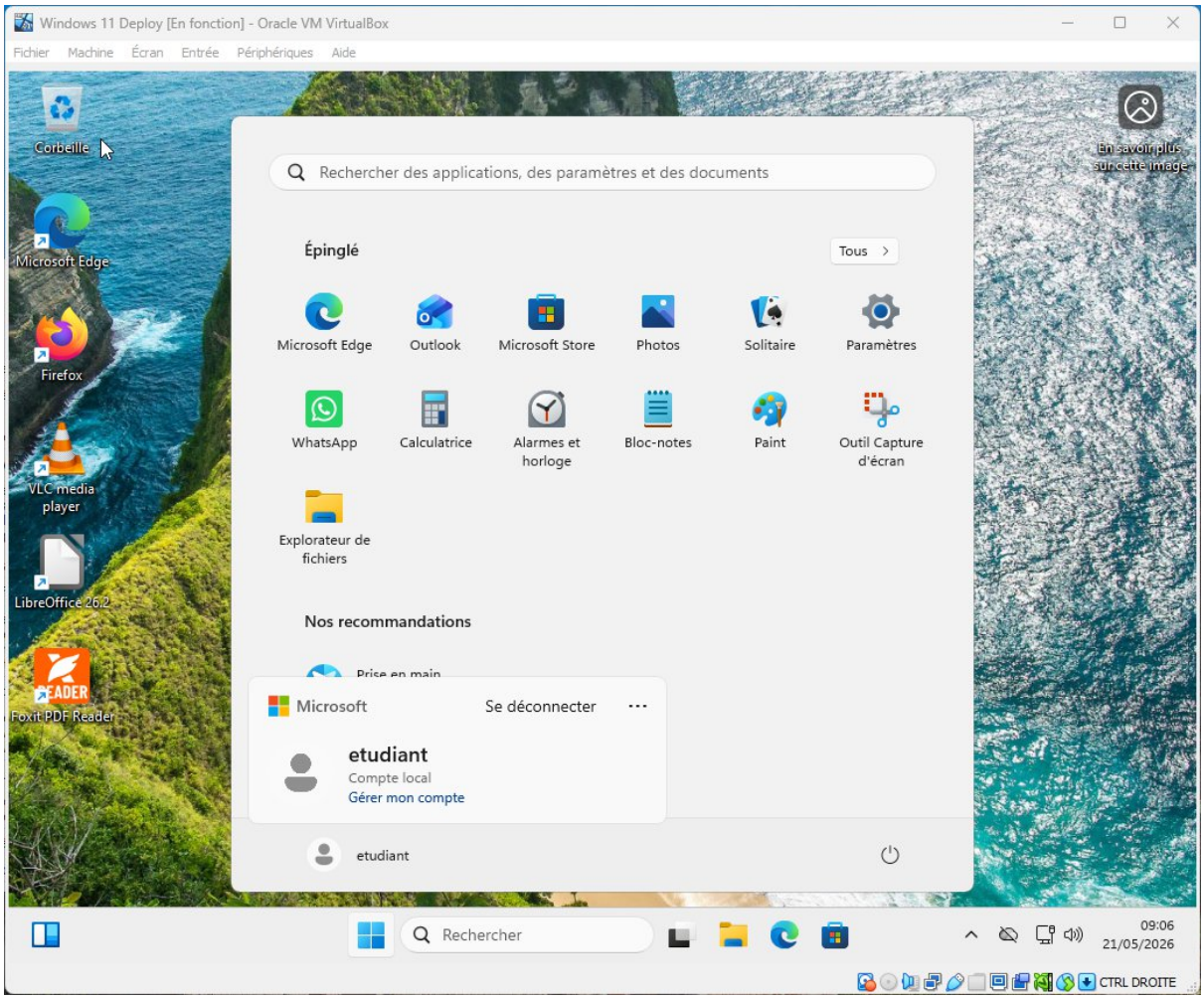
Capture en cours d'une image sur l'hôte Windows11Master dans l'onglet des tâches FOG.



Détail Partclone lors de la capture — débit, volume traité, temps estimé.



Premier démarrage de la VM Windows11Deploy après déploiement — Windows 11 affiche l'écran OOBE, conséquence du sysprep en mode OOBE/Generalize.



Session ouverte sur le compte étudiant après déploiement — les logiciels installés sur le master sont bien présents, confirmant le succès du déploiement.

A2. Installation de Grafana et connexion de la datasource Zabbix

Grafana est installé sur une machine virtuelle dédiée sous Debian 13, dans le même VLAN que le serveur Zabbix (VLAN 32, IP 172.16.32.60). Les commandes suivantes sont exécutées en root.

A2.1. Installation de Grafana OSS

Ajout du dépôt APT officiel Grafana et installation du paquet :

```
apt install -y apt-transport-https software-properties-common wget
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 && apt install -y grafana
systemctl enable --now grafana-server
```

L'interface Grafana est ensuite accessible sur `http://172.16.32.60:3000` avec les identifiants par défaut `admin / admin` (un changement de mot de passe est imposé à la première connexion).

A2.2. Installation du plugin datasource Zabbix

L'installation du plugin nécessite les droits root (la commande n'est pas accessible en mode utilisateur normal). Sur la VM Grafana :

```
sudo grafana cli plugins install alexanderzobnin-zabbix-app
systemctl restart grafana-server
```

Note : la commande `grafana-cli` a été dépréciée depuis Grafana v10.0 et supprimée dans Grafana v13.0. La syntaxe à utiliser sur les versions actuelles est `sudo grafana cli plugins install alexanderzobnin-zabbix-app` (sans tiret entre grafana et cli).

A2.3. Configuration de la datasource dans Grafana

Dans l'interface Grafana, activer le plugin : Administration → Plugins → rechercher « Zabbix » → Enable.

Ajouter ensuite la datasource : Connections → Data sources → Add new data source → Zabbix. Renseigner les paramètres suivants :

```
URL : http://172.16.32.50/zabbix/api_jsonrpc.php
Username : Admin
Password : <mot de passe Zabbix>
```

Cliquer sur « Save & test ». Un message de confirmation indique que la connexion à l'API Zabbix est établie.

A3. Installation de Zabbix 7.4 sur Debian 13

Zabbix 7.4 est installé sur une machine virtuelle dédiée sous Debian 13 (IP 172.16.32.50, VLAN 32), avec Apache comme serveur web et MariaDB comme base de données. Les commandes sont exécutées en root.

A3.1. Ajout du dépôt et installation des paquets

```
wget https://repo.zabbix.com/zabbix/7.4/release/debian/pool/main/z/zabbix-
release/zabbix-release_latest_7.4+debian13_all.deb
dpkg -i zabbix-release_latest_7.4+debian13_all.deb
apt update
apt install -y zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-
sql-scripts zabbix-agent2 mariadb-server
```

A3.2. Création de la base de données

```
mysql
CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
CREATE USER zabbix@localhost IDENTIFIED BY 'motdepasse';
GRANT ALL PRIVILEGES ON zabbix.* TO zabbix@localhost;
FLUSH PRIVILEGES;
QUIT;
```

Import du schéma Zabbix dans la base :

```
zcat /usr/share/zabbix/sql-scripts/mysql/server.sql.gz | mysql --default-character-
set=utf8mb4 -uzabbix -p zabbix
```

Renseigner ensuite le mot de passe de la base dans `/etc/zabbix/zabbix_server.conf` (paramètre `DBPassword`), puis démarrer les services :

```
systemctl enable --now zabbix-server zabbix-agent2 apache2
```

A3.3. Contournement du wizard web bloqué

Si le wizard de configuration web (`http://IP/zabbix/setup.php`) se bloque à l'étape de sélection du fuseau horaire (dropdown qui ne se charge pas), le contournement le plus efficace est de créer manuellement le fichier de configuration PHP. Cela bypasse intégralement le wizard, qui n'est qu'un outil de génération de ce fichier :

```
cat > /etc/zabbix/web/zabbix.conf.php << 'EOF'
<?php
global $DB;
$DB['TYPE']           = 'MYSQL';
$DB['SERVER']         = 'localhost';
$DB['PORT']           = '0';
$DB['DATABASE']       = 'zabbix';
$DB['USER']           = 'zabbix';
$DB['PASSWORD']       = 'MOT_DE_PASSE_DB';
$DB['SCHEMA']         = '';
$DB['ENCRYPTION']     = false;
$DB['DOUBLE_IEEE754'] = true;
$ZBX_SERVER           = 'localhost';
$ZBX_SERVER_PORT      = '10051';
$ZBX_SERVER_NAME      = 'PROTOTYPE-ZABBIX';
$IMAGE_FORMAT_DEFAULT = IMAGE_FORMAT_PNG;
EOF
```

Une fois le fichier créé, accéder directement à `http://172.16.32.50/zabbix/` — le wizard est bypassé et la page de login s'affiche directement. Identifiants par défaut : `Admin / zabbix` (à changer immédiatement après la première connexion).

A3.4. Correction des locales manquantes

Si l'interface Zabbix affiche des problèmes de rendu liés aux locales, générer la locale `en_US.UTF-8` :

```
sed -i 's/# en_US.UTF-8 UTF-8/en_US.UTF-8 UTF-8/' /etc/locale.gen
locale-gen
systemctl restart apache2
```

A3.5. Concepts clés de Zabbix : hôtes, templates, triggers et macros

Hôte. Un hôte (host) est la représentation dans Zabbix d'un équipement à superviser : un serveur Linux, un switch, un routeur. On lui associe une ou plusieurs interfaces (agent, SNMP, ICMP) qui définissent comment Zabbix communique avec lui. Pour ajouter un hôte : Data collection → Hosts → Create host. On renseigne le nom, le groupe, l'interface (type, IP, port), puis on attache un ou plusieurs templates.

Template. Un template est un ensemble préconfiguré d'items (métriques à collecter), de triggers (seuils d'alerte) et de graphes, conçu pour un type d'équipement donné. Zabbix fournit des templates officiels pour Linux, Windows, Cisco IOS, Proxmox VE, etc. Appliquer un template à un hôte crée automatiquement tous les items correspondants, sans configuration manuelle item par item. Un même hôte peut recevoir plusieurs templates (par exemple « Linux by Zabbix agent » et « ICMP Ping » simultanément).

Trigger. Un trigger est une expression logique évaluée en temps réel sur les valeurs collectées. Quand l'expression devient vraie (par exemple : CPU > 90 % pendant 5 minutes, ou host injoignable en ICMP), Zabbix déclenche un événement et peut envoyer une alerte. La sévérité d'un trigger va de « Not classified » à « Disaster ». Les templates officiels incluent déjà des triggers préconfigurés, il est possible d'en ajouter de custom directement sur un hôte ou dans un template.

Macro. Une macro est une variable réutilisable définie sous la forme `{$NOM_MACRO}`. Elle peut être définie globalement, au niveau d'un template, ou au niveau d'un hôte (ce qui permet de surcharger la valeur du template pour un équipement spécifique). Les macros sont très utilisées dans les templates pour paramétrer les seuils de trigger (ex. `{$CPU.UTIL.CRIT}` = 90) ou les identifiants SNMP (`{$SNMP_COMMUNITY}`). Cela permet d'adapter le comportement d'un template sans le modifier : il suffit de redéfinir la macro sur l'hôte concerné.

A3.6. Configuration SNMPv3 sur un switch Cisco IOS et intégration dans Zabbix

A3.6.1. Configuration côté switch (Cisco IOS)

Les commandes suivantes sont à saisir en mode configuration globale. Elles créent une vue MIB, un groupe SNMPv3 avec authentification et chiffrement, et l'utilisateur que Zabbix utilisera pour interroger l'équipement :

```
enable
conf t
snmp-server view ZBX-VIEW iso included
snmp-server group ZBX-GROUP v3 priv read ZBX-VIEW
snmp-server user zbxuser ZBX-GROUP v3 auth sha MOT_DE_PASSE_AUTH priv aes 128
MOT_DE_PASSE_PRIV
end
```

Vérification de la configuration appliquée :

```
show snmp user
show snmp group
```

Test de connectivité SNMP depuis la VM Zabbix (installer le paquet `snmp` si absent) :

```
snmpget -v3 -l authPriv -u zbxuser -a SHA -A "MOT_DE_PASSE_AUTH" -x AES -X
"MOT_DE_PASSE_PRIV" IP_DU_SWITCH 1.3.6.1.2.1.1.1.0
```

Une réponse de type `STRING: Cisco IOS...` confirme que SNMPv3 est opérationnel sur l'équipement et que la VM Zabbix peut l'interroger correctement avant de procéder à la configuration dans l'interface.

A3.6.2. Ajout de l'hôte dans Zabbix

Dans l'interface Zabbix, accéder à `Data collection → Hosts → Create host`. Renseigner les champs suivants :

```
Host name       : SW-CISCO-2960 (ou nom de votre choix)
Host groups     : groupe de votre choix (ex. PROTOTYPES)
Templates       : Cisco IOS by SNMP
Interface       : SNMP — IP du switch, port 161
```

Dans la section SNMP de l'interface, sélectionner `SNMPv3` et renseigner les paramètres suivants. Attention : le champ **Security name** correspond au nom d'utilisateur SNMP (`zbxuser`) et non au mot de passe, erreur fréquente qui empêche l'ouverture de session SNMP :

```
Security name           : zbxuser
Security level          : authPriv
Authentication protocol : SHA1
Authentication passphrase : MOT_DE_PASSE_AUTH
Privacy protocol        : AES128
Privacy passphrase      : MOT_DE_PASSE_PRIV
```

Cliquer sur `Add` pour enregistrer l'hôte. Après quelques secondes, l'indicateur de disponibilité de l'hôte doit passer au vert dans la liste des hôtes. Les items du template commencent à collecter des données ; ils sont visibles dans `Monitoring → Latest data` en filtrant sur le nom de l'hôte.

A3.7. Configuration SNMPv3 sur les commutateurs de production (Aruba et Comware)

Contrairement à la maquette Packet Tracer (annexe A3.6, équipements Cisco IOS), les commutateurs réels de la section relèvent de deux familles de systèmes d'exploitation imposant des syntaxes distinctes : ArubaOS-Switch pour le cœur de réseau Aruba 3810M, et Comware (HP/H3C) pour les commutateurs HP. Le niveau de sécurité retenu est authPriv, garantissant à la fois l'authenticité et l'intégrité (authentification SHA) et la confidentialité (chiffrement) des échanges de supervision.

A3.7.1. Cœur de réseau — Aruba 3810M (ArubaOS-Switch)

Le stack de deux commutateurs étant administré comme un équipement logique unique, la configuration n'est saisie qu'une seule fois sur le commander. Le chiffrement retenu sur cet équipement est DES :

```
configure terminal
snmpv3 enable
snmpv3 user zabbix_mon auth sha MOT_DE_PASSE_AUTH priv des MOT_DE_PASSE_PRIV
snmpv3 group managerpriv user "zabbix_mon" sec-model ver3
write memory
```

Vérification : `show snmpv3 user`. ArubaOS-Switch ne supporte de manière fiable que SHA-1 pour l'authentification.

A3.7.2. Commutateurs Comware (HPE 5140 EI, FlexNetwork 5120 EI, HP A5120 EI)

Les commutateurs Comware partagent la même logique de configuration, mais la version de Comware impose une nuance : le mot-clé `simple`, présent sur Comware 7 (HPE 5140 EI), n'existe pas sur Comware 5 (FlexNetwork 5120 EI et HP A5120 EI), où le mot de passe est saisi en clair par défaut. Le chiffrement retenu sur ces équipements est AES-128. Le groupe doit impérativement être créé avant l'utilisateur.

Comware 7 — HPE 5140 EI :

```
system-view
snmp-agent
snmp-agent sys-info version v3
snmp-agent group v3 zabbix_grp privacy
snmp-agent usm-user v3 zabbix_mon zabbix_grp simple authentication-mode sha
MOT_DE_PASSE_AUTH privacy-mode aes128 MOT_DE_PASSE_PRIV
save
```

Comware 5 — FlexNetwork 5120 EI et HP A5120 EI (sans le mot-clé simple) :

```
system-view
snmp-agent
snmp-agent sys-info version v3
snmp-agent group v3 zabbix_grp privacy
snmp-agent usm-user v3 zabbix_mon zabbix_grp authentication-mode sha
MOT_DE_PASSE_AUTH privacy-mode aes128 MOT_DE_PASSE_PRIV
save
```

Vérification : `display snmp-agent usm-user`.

A3.7.3. Choix des templates Zabbix

Le template officiel Aruba de Zabbix concerne la gamme ArubaOS-CX et ne convient pas au 3810M (ArubaOS-Switch). Le template retenu est donc « HP Enterprise Switch by SNMP » pour le cœur Aruba, et « HP Comware HH3C by SNMP » pour les trois commutateurs Comware. L'ajout des hôtes dans Zabbix suit la procédure de l'annexe

A3.6.2 (interface SNMPv3, le champ Security name correspondant au nom d'utilisateur zabbix_mon) ; le protocole de chiffrement renseigné dans Zabbix doit correspondre à celui configuré sur l'équipement, soit DES pour l'Aruba et AES128 pour les Comware.

A4. Déploiement de l'agent Zabbix sur les hôtes Linux

L'agent zabbix-agent2 est déployé sur les hôtes Linux supervisés (nœuds Proxmox, VM FOG) afin de remonter les métriques système (CPU, RAM, disque) ainsi que les métriques personnalisées via UserParameters. Les commandes sont exécutées en root.

A4.1. Installation et configuration de l'agent

Après ajout du dépôt Zabbix, installation du paquet :

```
apt install -y zabbix-agent2
```

Le fichier de configuration principal est `/etc/zabbix/zabbix_agent2.conf`. Trois directives sont essentielles :

```
Server=172.16.32.50
ServerActive=172.16.32.50
Hostname=VM FOG
```

`Server` désigne l'adresse du serveur Zabbix autorisé à interroger l'agent en mode passif. `ServerActive` indique le serveur vers lequel l'agent émet ses données en mode actif (et non 127.0.0.1 par défaut). `Hostname` doit correspondre exactement au nom de l'hôte tel qu'il est déclaré dans l'interface Zabbix : dans le cas contraire, les items en mode actif ne remontent pas, le serveur ne sachant pas à quel hôte associer les données. Après modification, redémarrer l'agent :

```
systemctl restart zabbix-agent2
```

A4.2. Validation depuis le serveur Zabbix

Le test de récupération d'une valeur s'effectue depuis le serveur Zabbix (et non depuis l'agent lui-même, qui n'est pas dans la liste des serveurs autorisés), à l'aide de l'outil zabbix_get (paquet zabbix-get) :

```
zabbix_get -s 172.16.32.250 -p 10050 -k system.cpu.load
```

Une valeur numérique en retour confirme que le serveur peut interroger l'agent.

A5. Supervision de la disponibilité des services (tests TCP)

Pour superviser le service lui-même et non seulement la machine qui l'héberge, des tests TCP sont créés. Un simple ping ICMP confirmerait que la VM répond, mais pas que le service applicatif est fonctionnel : tester le port applicatif est donc plus pertinent.

Dans Data collection → Hosts → (hôte) → Items → Create item, un item de type Simple check est créé avec une clé de la forme :

```
net.tcp.service[http,172.16.20.250,80]
```

La clé `net.tcp.service` retourne 1 si le port répond, 0 sinon — format idéal pour un indicateur UP/DOWN. Pour un Simple check, le champ « Host interface » n'est pas utilisé : la cible est définie directement dans la clé, et le test est exécuté par le serveur Zabbix. Les quatre services du périmètre prototype utilisent les clés suivantes :

```
FOG     : net.tcp.service[http,172.16.20.7,80]
Grafana : net.tcp.service[tcp,172.16.20.6,3000]
Proxmox : net.tcp.service[tcp,172.16.99.18,8006]
Zabbix  : net.tcp.service[tcp,172.16.32.50,10051]
```

Le mot-clé `http` effectue un test applicatif léger ; `tcp` se limite à vérifier l'ouverture du port (utilisé pour les services dont le protocole applicatif n'est pas reconnu nativement).

A6. Supervision des baux DHCP (script personnalisé)

Sur le prototype, la VM FOG assure le service DHCP via isc-dhcp-server. L'objectif est de remonter dans Zabbix le nombre de baux attribués, libres, et la taille totale du pool.

Adaptation à l'infrastructure réelle : sur l'infrastructure du lycée, le service DHCP n'est pas assuré par FOG mais par le serveur Active Directory Windows. La méthode décrite ci-dessous (script bash lisant le fichier de baux isc-dhcp) ne s'y applique donc pas directement : il faudra recourir à une approche Windows, par exemple via PowerShell (Get-DhcpServerv4ScopeStatistics) ou WMI. La présente procédure établit le principe sur le prototype, l'adaptation restant à réaliser.

A6.1. Analyse du fichier de baux

isc-dhcp stocke les baux dans `/var/lib/dhcp/dhcpd.leases`. Chaque bail est un bloc dont la ligne `binding state` indique l'état (active pour un bail attribué, free pour un bail relâché). La plage du pool est définie dans `/etc/dhcp/dhcpd.conf` par la directive range, soit 245 adresses (de .10 à .254).

A6.2. Script de comptage

Le script suivant est créé dans `/usr/local/bin/dhcp_leases.sh` :

```
#!/bin/bash
LEASE_FILE="/var/lib/dhcp/dhcpd.leases"
POOL_TOTAL=245

case "$1" in
  active)
    grep -c "binding state active" "$LEASE_FILE"
    ;;
  free)
    ACTIVE=$(grep -c "binding state active" "$LEASE_FILE")
    echo $((POOL_TOTAL - ACTIVE))
    ;;
  total)
    echo "$POOL_TOTAL"
    ;;
  *)
    echo "Usage: $0 {active|free|total}"
    exit 1
    ;;
esac
```

Le script prend un argument (`active`, `free` ou `total`). La structure `case` aiguille vers le bloc correspondant : `grep -c` compte les lignes « binding state active » (baux attribués) ; le bloc `free` calcule la différence entre le total et les actifs via la syntaxe arithmétique $((...)). Le script est ensuite rendu exécutable avec chmod +x.

Limite connue : grep -c compte les occurrences dans le fichier. Or isc-dhcp conserve un historique, et un même bail peut apparaître plusieurs fois. Pour un prototype, cette approximation est acceptable ; en production, il faudrait dédoublonner par adresse IP pour un comptage exact.

A6.3. Intégration via UserParameter

Un UserParameter relie une clé Zabbix à l'exécution du script. La directive Include de la configuration de l'agent (`Include=/etc/zabbix/zabbix_agent2.d/*.conf`) charge tous les fichiers .conf placés directement dans ce répertoire (le motif ne descend pas dans les sous-dossiers). Le fichier `/etc/zabbix/zabbix_agent2.d/dhcp.conf` est créé avec :

```
UserParameter=dhcp.active,/usr/local/bin/dhcp_leases.sh active
UserParameter=dhcp.free,/usr/local/bin/dhcp_leases.sh free
UserParameter=dhcp.total,/usr/local/bin/dhcp_leases.sh total
```

Après redémarrage de l'agent (systemctl restart zabbix-agent2), la validation s'effectue depuis le serveur Zabbix :

```
zabbix_get -s 172.16.32.250 -p 10050 -k dhcp.active
```

Point d'attention sur les permissions : l'agent Zabbix s'exécute sous l'utilisateur zabbix. Le script doit donc être exécutable par cet utilisateur et le fichier de baux lisible par lui. Le test sudo -u zabbix /usr/local/bin/dhcp_leases.sh active permet de le vérifier.

Enfin, trois items de type Zabbix agent (clés dhcp.active, dhcp.free, dhcp.total) sont créés côté serveur. Le type doit être Zabbix agent et non Simple check, les UserParameters étant exécutés par l'agent (port 10050).

A6.4. Adaptation à l'AD Windows (procédure)

Sur l'infrastructure réelle, le service DHCP est porté par les serveurs Active Directory Windows ; la méthode bash décrite ci-dessus, qui lit le fichier de baux isc-dhcp, ne s'y applique pas. La remontée des baux y est obtenue selon la même logique (un script appelé par un UserParameter), mais réécrite en PowerShell à l'aide du cmdlet Get-DhcpServerv4ScopeStatistics. Le serveur Zabbix de production étant joignable sur 172.16.20.7, c'est cette adresse qui est renseignée dans la configuration de l'agent.

Le ScopeId du scope à superviser (son adresse réseau) est obtenu par la commande Get-DhcpServerv4Scope. Le cmdlet Get-DhcpServerv4ScopeStatistics retourne, pour un scope donné, les champs InUse (baux attribués), Free (baux libres) et Reserved (réservations). Le script PowerShell suivant est créé dans C:\zabbix_scripts\dhcp_stats.ps1, SCOPE_ID étant remplacé par la valeur relevée :

```
param([string]$metric)
$stats = Get-DhcpServerv4ScopeStatistics -ScopeId "SCOPE_ID"
switch ($metric) {
    "active" { Write-Output $stats.InUse }
    "free"   { Write-Output $stats.Free }
    "total"  { Write-Output ($stats.InUse + $stats.Free + $stats.Reserved) }
}
```

Le fichier de configuration de l'agent (C:\Program Files\Zabbix Agent 2\zabbix_agent2.conf) est renseigné avec l'adresse du serveur Zabbix de production et le nom d'hôte déclaré dans Zabbix :

```
Server=172.16.20.7
ServerActive=172.16.20.7
Hostname=SRV-AD-DHCP1
```

Une règle de pare-feu Windows autorise le trafic entrant sur le port 10050/TCP depuis le seul serveur Zabbix :

```
New-NetFirewallRule -DisplayName "Zabbix Agent 2 (10050)" -Direction Inbound -
Protocol TCP -LocalPort 10050 -RemoteAddress 172.16.20.7 -Action Allow
```

Les UserParameters sont ajoutés au fichier de configuration de l'agent. Le flag -ExecutionPolicy Bypass est nécessaire, la politique d'exécution Windows bloquant sinon les scripts lancés par le service de l'agent :

```
UserParameter=dhcp.active,powershell -NonInteractive -NoProfile -ExecutionPolicy
Bypass -File "C:\zabbix_scripts\dhcp_stats.ps1" active
UserParameter=dhcp.free,powershell -NonInteractive -NoProfile -ExecutionPolicy
Bypass -File "C:\zabbix_scripts\dhcp_stats.ps1" free
UserParameter=dhcp.total,powershell -NonInteractive -NoProfile -ExecutionPolicy
Bypass -File "C:\zabbix_scripts\dhcp_stats.ps1" total
```

Après redémarrage du service (Restart-Service « Zabbix Agent 2 »), la validation s'effectue depuis le serveur Zabbix :

```
zabbix_get -s 172.16.20.21 -k dhcp.active
```

Point d'attention sur les permissions : le service Zabbix Agent 2 s'exécute sous le compte SYSTEM, qui dispose localement d'un accès suffisant aux données DHCP. En cas d'erreur « access denied » à l'exécution par l'agent, le compte de service doit être ajouté au groupe local « DHCP Administrators ». Enfin, comme sur le prototype, trois items de type Zabbix agent (clés dhcp.active, dhcp.free, dhcp.total, Numeric unsigned) sont créés côté serveur Zabbix.

A7. Intégration de la supervision Proxmox via l'API

La supervision du cluster Proxmox repose sur le template officiel « Proxmox VE by HTTP ». Contrairement à l'agent Zabbix, ce template interroge directement l'API REST de Proxmox en HTTP, sans script externe ni agent installé sur les nœuds. L'authentification s'effectue au moyen d'un token API Proxmox.

A7.1. Création du token API et attribution des droits

Un utilisateur dédié à la supervision est créé (compte monitoring), auquel est associé un token API via Datacenter → Permissions → API Tokens. Le rôle PVEAuditor (lecture seule, tous les droits .Audit) est attribué selon le principe de moindre privilège.

Privilege Separation du token : par défaut, un token Proxmox possède ses propres permissions, indépendantes de celles de l'utilisateur. Il ne suffit donc pas d'attribuer le rôle à l'utilisateur : il faut également l'attribuer explicitement au token lui-même, sans quoi l'API renvoie des données vides ou des erreurs d'autorisation.

Chemins ACL : les chemins /cluster/resources et /cluster/status mentionnés dans certaines documentations ne sont pas des chemins ACL valides sur Proxmox VE 9 (erreur « invalid path »). Le rôle PVEAuditor est donc attribué, pour l'utilisateur comme pour le token, sur les chemins /, /nodes, /vms et /storage avec propagation, ce qui couvre les ressources interrogées par le template.

A7.2. Configuration de l'hôte dans Zabbix

Un hôte est créé dans Zabbix et le template « Proxmox VE by HTTP » lui est attaché. Les macros suivantes sont renseignées (la valeur du secret est conservée séparément et ne figure pas dans ce document) :

```
{$PVE.URL.HOST}      : adresse de l'API Proxmox
{$PVE.URL.PORT}      : 8006
{$PVE.TOKEN.ID}      : monitoring@pam!zabbix
{$PVE.TOKEN.SECRET} : <SECRET_DU_TOKEN>
```

Le test du token, indépendamment de Zabbix, s'effectue par une requête directe à l'API :

```
curl -k -H "Authorization: PVEAPIToken=monitoring@pam!zabbix=<SECRET>" \
  https://IP_PROXMOX:8006/api2/json/cluster/resources
```

Un retour JSON contenant les ressources confirme que le token est correctement autorisé. Le format du header est PVEAPIToken=USER@REALM!TOKENID=SECRET.

Comportement sur nœud isolé : sur un nœud Proxmox standalone (hors cluster), les compteurs préfixés « Cluster: » renvoient 0. Ce comportement est normal ; ces métriques n'ont de sens que sur un véritable cluster multi-nœuds.

A8. Techniques de visualisation Grafana

A8.1. Affichage de l'état des ports et services (panneau Stat)

Pour afficher l'état d'un port ou d'un service sous forme de bloc coloré portant le nom de l'élément (et non la valeur brute 1/0), trois réglages sont combinés : le `Text mode` est réglé sur « Name » (le panneau affiche le nom de la série) ; la fonction Zabbix `setAlias` est appliquée sur chaque requête pour forcer un nom distinct (sans quoi plusieurs requêtes afficheraient le même libellé) ; une value mapping (1 → vert, 0 → rouge) associée au mode de couleur « Background » pilote la couleur de fond. Le nom reste lisible tandis que l'état est indiqué par la couleur, sans texte « UP/DOWN » redondant.

A8.2. Seuils colorés sur l'uptime

Pour colorer l'uptime des services selon leur durée de fonctionnement, on utilise les Thresholds (et non les value mappings, réservées aux valeurs discrètes). L'uptime étant exprimé en secondes, les seuils sont : rouge à 0 (uptime inférieur à 5 min, redémarrage récent), orange à 300 (entre 5 min et 1 h), vert à 3600 (supérieur à 1 h). Le mode de couleur « Background » applique la couleur au fond du bloc. Cette logique met en évidence un redémarrage récent, signe d'un incident potentiel.

A8.3. Baux DHCP, utilisation disque et mémoire

Les baux DHCP sont représentés par un panneau Bar gauge horizontal à trois séries (attribués, libres, total du pool), avec des bornes fixées (0 à 245) pour une lecture immédiate des proportions. L'utilisation disque des VM (item vfs.fs.size[/,pused]) et la consommation mémoire sont représentées par des panneaux Gauge avec seuils colorés, format adapté à la détection rapide d'une saturation sur grand écran.

Note : les nouveaux tags ou items Zabbix n'apparaissent dans les menus Grafana qu'après un redémarrage du service grafana-server, en raison du cache du plugin datasource.

A9. Supervision de la température CPU (UserParameter personnalisé)

A9.1. Le UserParameter

Sur les nœuds Proxmox équipés de deux processeurs, la commande sensors renvoie deux lignes « Core 0 » (une par socket) au format `+47.0°C  (high = +100.0°C, crit = +100.0°C)`. Zabbix nécessitant une valeur numérique unique pour un item de type float, la moyenne des deux températures est calculée directement dans la commande associée au UserParameter, sans script externe :

```
UserParameter=system.cpu.temp.avg,sensors | awk '/Core 0/{gsub(/[^0-9.]/,"",$3);
s+=$3; n++} END{printf "%.1f\n", s/n}'
```

La commande awk filtre les lignes contenant « Core 0 », supprime tout caractère non numérique du troisième champ (qui vaut par exemple `+47.0°C`), accumule les valeurs et divise par le nombre de lignes trouvées (n). Les seuils high et crit présents en fin de ligne sont ignorés naturellement car ils se trouvent dans les champs suivants. L'usage de n plutôt qu'une constante fixée garantit le fonctionnement quel que soit le nombre de processeurs du nœud.

A9.2. Déploiement sur l'agent

Sur chaque nœud supervisé, le fichier de configuration UserParameter est créé dans le répertoire dédié à zabbix-agent2. L'utilisation d'un heredoc avec `'EOF'` (guillemets simples) neutralise toute interprétation du caractère `$3` par le shell, ce qui garantit que le fichier contiendra la commande exacte attendue par l'agent :

```
sudo tee /etc/zabbix/zabbix_agent2.d/userparameter_temperature.conf << 'EOF'
UserParameter=system.cpu.temp.avg,sensors | awk '/Core 0/{gsub(/[^0-9.]/,"",$3);
s+=$3; n++} END{printf "%.1f\n", s/n}'
EOF
sudo systemctl restart zabbix-agent2
```

Validation depuis le serveur Zabbix (la commande est à exécuter depuis le serveur, non depuis l'agent) :

```
zabbix_get -s <IP_DU_NOEUD> -k system.cpu.temp.avg
```

A9.3. Création de l'item dans Zabbix

Aucun template standard ne propose la clé `system.cpu.temp.avg` ; l'item est donc créé manuellement sur chaque hôte Proxmox via Data collection → Hosts → [hôte] → Items → Create item, avec les paramètres suivants :

```
Name            : Température CPU moyenne
Type            : Zabbix agent
Key             : system.cpu.temp.avg
Type of information : Numeric (float)
```

Après quelques cycles de collecte, la valeur apparaît dans Monitoring → Latest data. L'item peut ensuite être ajouté à un panneau Grafana de type Time series pour visualiser l'évolution thermique du nœud.

A10. Sécurisation HTTPS de la supervision (PKI interne)

A10.1. Création de la CA et des certificats

La supervision étant déployée sur un réseau interne dépourvu d'accès à une autorité de certification publique, une autorité de certification (CA) propre à l'établissement a été créée pour signer les certificats des serveurs Zabbix et Grafana. Les commandes sont exécutées sur le serveur Zabbix, qui héberge la CA :

```
sudo mkdir -p /etc/ssl/ca-lycee && cd /etc/ssl/ca-lycee
openssl genrsa -out ca.key 4096
openssl req -new -x509 -days 3650 -key ca.key -out ca.crt \
  -subj "/C=FR/ST=Centre-Val de Loire/L=Tours/O=Lycee Paul-Louis
Courier/OU=SIO/CN=CA-Lycee-PLC"
```

Un certificat est ensuite généré pour chaque service. L'adresse IP du serveur doit impérativement figurer dans le champ SubjectAltName, faute de quoi les navigateurs récents rejettent le certificat. Exemple pour Zabbix (172.16.20.7) :

```
openssl genrsa -out zabbix.key 2048
openssl req -new -key zabbix.key -out zabbix.csr -subj "/CN=172.16.20.7"
echo "subjectAltName=IP:172.16.20.7" > zabbix.ext
openssl x509 -req -days 1095 -in zabbix.csr -CA ca.crt -CAkey ca.key \
  -CAcreateserial -out zabbix.crt -extfile zabbix.ext
```

La même procédure est répétée pour Grafana en remplaçant l'adresse par 172.16.20.6 (fichiers grafana.key, grafana.crt). La clé privée de la CA (ca.key) ne quitte jamais le serveur ; seul le certificat public ca.crt est diffusé.

A10.2. Activation du HTTPS sur Zabbix et Grafana

Côté Zabbix, un hôte virtuel Apache en écoute sur le port 443 référence les certificats. Les en-têtes nécessaires à un éventuel affichage en iframe y figurent également :

```
SSLEngine on
SSLCertificateFile /etc/ssl/ca-lycee/zabbix.crt
SSLCertificateKeyFile /etc/ssl/ca-lycee/zabbix.key
Header always unset X-Frame-Options
Header always set Content-Security-Policy "frame-ancestors
https://172.16.20.6:3000"
```

Côté Grafana, le protocole HTTPS est activé dans la section [server] du fichier grafana.ini :

```
protocol = https
cert_file = /etc/ssl/grafana.crt
cert_key = /etc/ssl/grafana.key
```

Les services sont ensuite rechargés (systemctl reload apache2 et systemctl restart grafana-server). Les deux interfaces répondent alors en HTTPS.

A10.3. Import du certificat de la CA dans Firefox

Tant que le certificat racine de la CA n'est pas connu du navigateur, celui-ci signale la connexion comme non sécurisée. Le certificat ca.crt doit donc être importé dans

chaque navigateur client. Sur le poste d'affichage (Linux, Firefox), il est d'abord transféré depuis le serveur :

```
scp zabbix@172.16.20.7:/etc/ssl/ca-lycee/ca.crt ~/ca-lycee.crt
```

L'import s'effectue ensuite dans Firefox : ouvrir Paramètres puis la rubrique « Vie privée et sécurité » ; dans la section « Certificats », cliquer sur « Afficher les certificats… » ; dans l'onglet « Autorités », cliquer sur « Importer… » et sélectionner le fichier ca-lycee.crt ; cocher « Confirmer cette AC pour identifier des sites web » puis valider. Après redémarrage du navigateur, les interfaces Zabbix et Grafana sont reconnues comme sécurisées et la rotation d'onglets s'effectue sans avertissement.

Glossaire

Définition des principaux sigles et termes techniques employés dans ce rapport.

AD — Active Directory : service d'annuaire de Microsoft assurant la gestion centralisée des utilisateurs, des ordinateurs et des ressources d'un domaine Windows.

CA — Certificate Authority (autorité de certification) : entité qui émet et signe des certificats numériques, fondant la confiance dans une chaîne TLS.

CORS — Cross-Origin Resource Sharing : mécanisme par lequel un navigateur autorise ou bloque l'accès à des ressources servies par un domaine différent de celui de la page.

CSP — Content Security Policy : en-tête HTTP de sécurité contrôlant les sources de contenu qu'un navigateur peut charger ou intégrer (dont la directive frame-ancestors).

DHCP — Dynamic Host Configuration Protocol : protocole attribuant automatiquement aux machines leur configuration réseau (adresse IP, passerelle, DNS).

DNS — Domain Name System : système de résolution des noms de domaine en adresses IP.

FOG — Free Open-source Ghost : solution libre de capture et de déploiement d'images systèmes sur le réseau.

ICMP — Internet Control Message Protocol : protocole utilisé notamment par la commande ping pour tester l'accessibilité d'un hôte.

LLD — Low-Level Discovery : mécanisme de Zabbix découvrant automatiquement des entités (interfaces, systèmes de fichiers, espaces de stockage) pour créer des items sans configuration manuelle.

MIB — Management Information Base : base décrivant de façon hiérarchique les objets supervisables d'un équipement SNMP.

OID — Object Identifier : identifiant numérique désignant un objet précis au sein d'une MIB SNMP.

PKI — Public Key Infrastructure (infrastructure à clés publiques) : ensemble des composants (CA, certificats, clés) permettant d'émettre et de vérifier des certificats.

PXE — Preboot Execution Environment : environnement d'amorçage réseau permettant à une machine de démarrer depuis un serveur distant (utilisé par FOG).

SAN — Subject Alternative Name : champ d'un certificat listant les noms ou adresses IP pour lesquels il est valide, requis par les navigateurs récents (à ne pas confondre avec Storage Area Network).

SNMP — Simple Network Management Protocol : protocole de supervision des équipements réseau tels que commutateurs, routeurs ou onduleurs.

SNMPv3 — Troisième version de SNMP, ajoutant l'authentification et le chiffrement des échanges (modèle USM).

TLS — Transport Layer Security : protocole de chiffrement des communications réseau, successeur de SSL et fondement du HTTPS.

USM — User-based Security Model : modèle de sécurité de SNMPv3 reposant sur des utilisateurs, avec authentification (SHA) et chiffrement (AES).

VLAN — Virtual Local Area Network : réseau local virtuel segmentant logiquement un même réseau physique.

Webographie

Sources consultées dans le cadre de la rédaction de ce rapport et de la réalisation des missions.

FOG Project

[1] Documentation officielle FOG Project — Multicasting
https://wiki.fogproject.org/wiki/index.php/Multicasting

Microsoft — Outils de déploiement

[2] Microsoft — Annonce de fin de vie de MDT (Microsoft Deployment Toolkit), janvier 2026
https://learn.microsoft.com/en-us/troubleshoot/mem/configmgr/mdt/mdt-retirement

[3] Microsoft — Documentation Sysprep (préparation d'un master Windows pour déploiement) https://learn.microsoft.com/fr-fr/windows-hardware/manufacture/desktop/sysprep--generalize--a-windows-installation

Cisco — Packet Tracer et équipements réseau

[4] Cisco Networking Academy — Téléchargement de Packet Tracer
https://www.netacad.com/resources/lab-downloads

[5] Cisco — Instructions d'installation de Packet Tracer (procédure détaillée Linux)
https://www.netacad.com/skillsforall/files/Cisco_Packet_Tracer_Download_and_Installation_Instructions.pdf

Problèmes connus — VirtualBox et compatibilité

[6] FOG Project Forums — Incompatibilité UEFI / boot PXE sous VirtualBox (problème connu) http://forums.fogproject.org/post/124147

Linux — Bonding, Proxmox et infrastructure

[7] Documentation Proxmox — Network Configuration, Linux Bonding (mode active-backup / failover) https://pve.proxmox.com/wiki/Network_Configuration#_bonding

Supervision — Zabbix, Grafana et sécurisation

[8] Documentation Zabbix — Items de type agent SNMP
https://www.zabbix.com/documentation/current/en/manual/config/items/itemtypes/snmp

[9] Documentation Zabbix — Configuration d'une carte réseau (Network maps)
https://www.zabbix.com/documentation/current/en/manual/config/visualization/maps/map

[10] Documentation Zabbix — User parameters (extension de l'agent)
https://www.zabbix.com/documentation/current/en/manual/config/items/userparameters

[11] Grafana Labs — Plugin Zabbix pour Grafana (data source)
https://grafana.com/grafana/plugins/alexanderzobnin-zabbix-app/

[12] Documentation Grafana — Mise en place du HTTPS (Set up HTTPS)
https://grafana.com/docs/grafana/latest/setup-grafana/set-up-https/

[13] Wiki Proxmox VE — Proxmox VE API (authentification par token)
https://pve.proxmox.com/wiki/Proxmox_VE_API

[14] Documentation OpenSSL — Extensions de certificat x509v3 (champ subjectAltName)
https://www.openssl.org/docs/manmaster/man5/x509v3_config.html