SIMPLON.CO

DOSSIER DE PROJET SEAHAWKS

SEAHAWKS MONITORING

Nadia BENBOUALI

AIS 2024-2025

PROXMOX pfsense veeam Prometheus

Grafana Glpi HAPROXY GitLab

Table des matières

Introduction

Ce dossier s'inscrit dans le cadre de l'épreuve de certification du titre professionnel Administrateur d'Infrastructures Sécurisées. (Il vise à présenter en détail un projet complet d'administration et de sécurisation d'une infrastructure informatique, mené de manière autonome en dehors d'un contexte d'entreprise.

Le projet retenu, intitulé **Seahawks Monitoring**, répond à un besoin de centralisation de la supervision, de la maintenance et du pilotage d'un parc informatique réparti sur plusieurs sites distants. Il met en œuvre de nombreuses compétences clés du métier : conception d'architectures virtualisées, automatisation, supervision, gestion des sauvegardes, sécurité réseau et télémaintenance.

À travers ce document, nous détaillerons les choix techniques réalisés, l'organisation de la mise en œuvre, les méthodes de travail collaboratif utilisées, ainsi que l'impact concret du projet sur les problématiques métier simulées. Ce travail a également permis de consolider une approche méthodologique rigoureuse, indispensable à tout administrateur en charge de la continuité et de la sécurité des services.

Fiche projet

Projet Seahawks

Durée: **du 16/06 au 09/09/2025**

Participants: **Trevys, Lyes et Nadia**

Environnement : **Datacenter Proxmox**

Objectif: **Supervision et maintenance sécurisée multisite.**

1.Compétences mises en œuvre

Dans le cadre du projet Seahawks Monitoring, nous avons mis en œuvre un ensemble de compétences techniques couvrant l'administration système, la virtualisation, le réseau, la supervision, le développement et la documentation. L'équipe était composée de trois membres : Trevys, Lyes et Nadia.

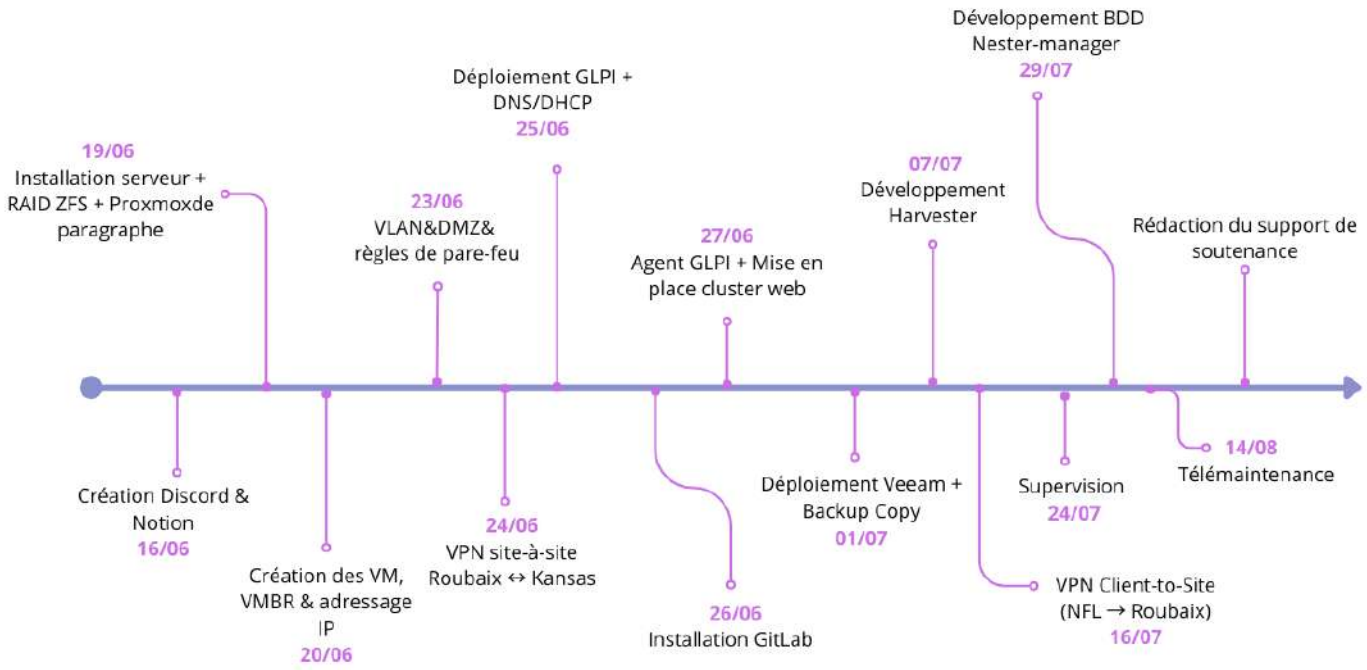

Figure 1 Frise chronologique des étapes clés du projet SeaHawks

Compétences mises en œuvre dans le projet Seahawks Monitoring

- CP1 : Appliquer les bonnes pratiques d'administration d'infrastructures
  → Mise en place d'une architecture virtualisée documentée et structurée (Proxmox, ZFS, plan IP).

- CP2 : Administrer et sécuriser les infrastructures réseaux
  → Configuration des VLANs, DMZ et VPN site-to-site / client-to-site sur pfSense.

- CP3 : Administrer et sécuriser les infrastructures systèmes
  → Déploiement de serveurs Linux (Debian, Alpine, Ubuntu) et Windows pour les rôles DHCP, DNS, GLPI, Veeam, GitLab.

- CP4 : Administrer et sécuriser les infrastructures virtualisées
  → Installation et gestion de Proxmox VE, création de VM et gestion du stockage via ZFS en RAID-Z3.

- CP5 : Concevoir une solution technique répondant à un besoin d'évolution
  → Développement de l'application Harvester/Nester pour la gestion des franchises NFL.

- CP6 : Mettre en production des évolutions de l'infrastructure
  → Intégration de nouveaux services (GLPI, Grafana, Prometheus, Veeam) et mise en service dans l'infra.

- CP7 : Mettre en œuvre et optimiser la supervision
  → Mise en place de Prometheus et Grafana pour le suivi des ressources et services.

- CP8 : Participer à la mesure du niveau de sécurité
  → Utilisation de PingCastle et Purple Knight pour auditer Active Directory.

- CP9 : Participer à l'élaboration et à la mise en œuvre de la politique de sécurité
  → Application de règles firewall sur pfSense et cloisonnement via VLAN/DMZ.

- CP10 : Participer à la détection et au traitement des incidents de sécurité
  → Mise en place d'alertes et centralisation des journaux via les outils de supervision.

2.Cahier des charges / Expression des besoins

2.1 Contexte du projet

Le projet Seahawks Monitoring est né du besoin de centraliser la supervision, la maintenance et la gestion des incidents au sein des 32 franchises de la NFL (National Football League). Trop d'interventions techniques étaient réalisées sur site, alors que la majorité pouvaient être effectuées à distance. Pour remédier à cela, NFL IT a souhaité concevoir une infrastructure complète, déployée dans son datacenter à Roubaix, avec un accès sécurisé depuis son siège social à Kansas City.

2.2 Objectifs du projet (présentation par besoins)

Besoins identifies	Objectifs à atteindre	Services/solutions prévus
-Interventions trop fréquentes sur site	Permettre un support à distance sécurisé	VPN IPsec site-à-site entre Roubaix et Kansas (pfSense) Télémaintenance
-Manque de visibilité sur le parc machine	Avoir un inventaire automatique et à jour	GLPI Inventory + agent sur chaque VM
-Suivi non centralisé des incidents	Traiter et suivre les demandes utilisateurs efficacement	GLPI avec gestion des tickets
-Risque de perte de données critiques	Sauvegarder régulièrement les machines	Veeam Backup & Replication + Backup Copy
-Aucun outil pour superviser les ressources	Obtenir des indicateurs techniques en temps réel	Grafana + Prometheus
-Pas de visualisation des données techniques collectées	Offrir une interface lisible au support technique	Développement de l'application web Nester
-Besoin de récolter automatiquement les données locales	Permettre l'exécution de scans et diagnostics réseau à la demande	Développement du Harvester (app Python + JSON)
-Besoin de garantir la haute disponibilité de l'interface web	Assurer l'accès continu au Nester même en cas de panne serveur	Cluster web (HAProxy + NGINX)
-Besoin de mettre à jour automatiquement les applications	Faciliter la maintenance et les évolutions du Harvester	Plateforme GitLab

3.Gestion de projet

3.1 Équipe projet

Le projet Seahawks Monitoring a été réalisé par une équipe de trois apprenants : Trevys, Lyes et Nadia. Chaque membre s'est vu attribuer des responsabilités spécifiques selon ses compétences et les besoins du projet.

La gestion du projet a été organisée dans l'espace collaboratif Notion, à l'aide d'une base de données structurée par tâches**.** Chaque tâche était associée à un ou plusieurs responsables, un niveau de priorité**,** des dates de début et de fin, ainsi qu'un statut d'avancement**.**

Cette organisation a permis un suivi clair et collaboratif des différentes étapes techniques du projet Seahawks.

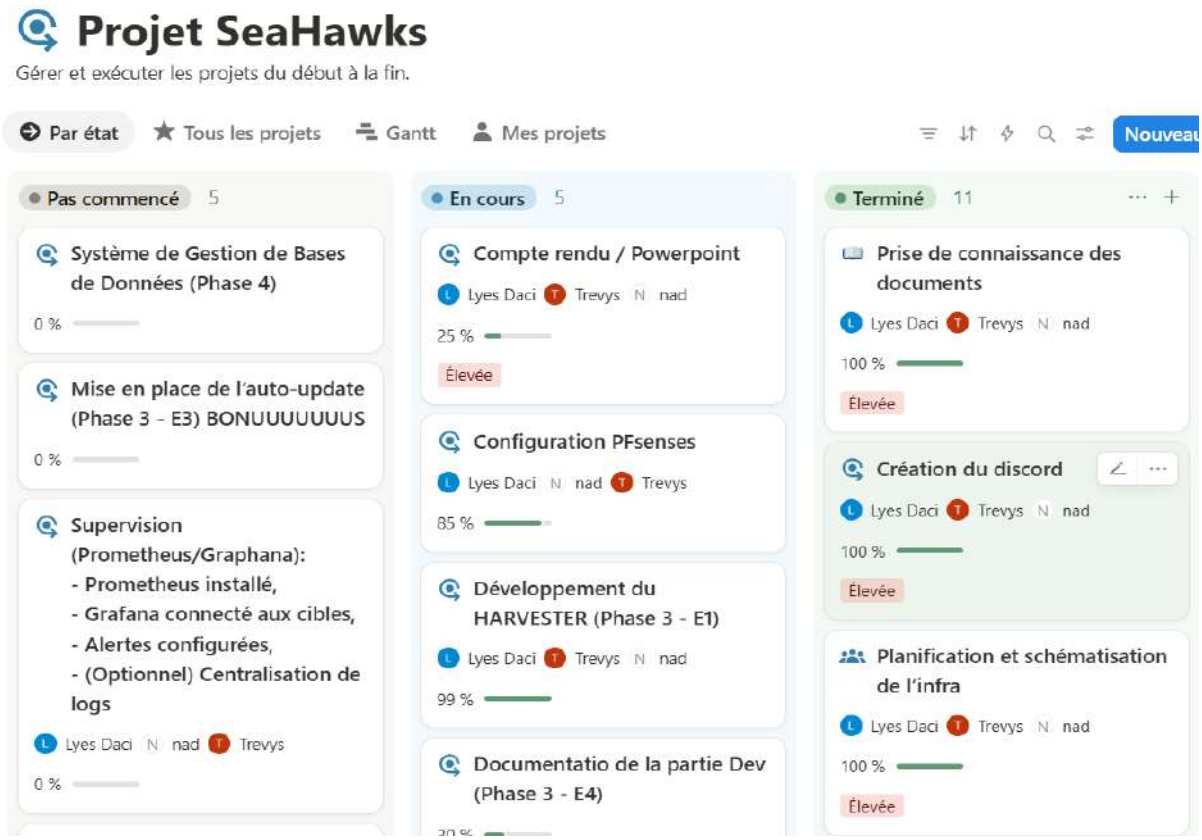

Figure 2 Outil de gestion de projet

3.2 Répartition des rôles

Figure 3 Répartition des tâches

3.3 Environnement technique

Choix de l'hyperviseur et configuration physique :

L'infrastructure mise en place dans le cadre du projet Seahawks Monitoring repose sur un environnement virtualisé basé sur Proxmox VE 8.4.0, installé sur un serveur physique dédié. Le choix de Proxmox s'explique par plusieurs avantages majeurs :

- Prise en charge native de ZFS pour un stockage fiable et résilient,
- Gestion centralisée des machines virtuelles et des containers,
- Intégration native du SDN (Software Defined Network) pour une meilleure flexibilité,
- Compatibilité avec de nombreux systèmes d'exploitation.

Le serveur hôte utilisé pour ce projet dispose des caractéristiques suivantes :

- Processeurs : 2 × Intel Xeon E5-2620 v3 @ 2.4 GHz (24 threads),
- Mémoire vive : 128 Go DDR4 ECC,
- Stockage : 8 disques de 300 Go configurés en RAID-Z3 ZFS, offrant 2,39 To de capacité utile.

Cette infrastructure matérielle et logicielle fournit une base robuste et sécurisée pour héberger l'ensemble des services du projet Seahawks Monitoring.

Sur le plan réseau, l'architecture est organisée autour de plusieurs zones fonctionnelles distinctes :

- Un réseau WAN pour la connexion Internet,
- Un réseau interne Roubaix pour les services critiques,
- Un réseau Kansas représentant le siège social,
- Un réseau NFL simulant les franchises clientes,
- Une DMZ dédiée à l'exposition des serveurs accessibles depuis l'extérieur (ex. HAProxy, Nester).

Cette organisation garantit une séparation claire entre les environnements internes, les zones exposées et les interconnexions avec l'extérieur, assurant ainsi une meilleure sécurité et une administration simplifiée.

4. Argumentaire du choix des solutions retenues

Le tableau ci-dessous présente l'ensemble des solutions techniques retenues dans le cadre du projet Seahawks Monitoring, accompagnées d'un argumentaire clair pour justifier leur intégration dans l'infrastructure.

Service / Solution choisie	Justification du choix
VPN IPsec site-à-site (pfSense)	Assure une communication sécurisée entre les sites Roubaix et Kansas. Facile à déployer via pfSense, open source et robuste.
VPN Client-to-Site (WireGuard)	Garantit un tunnel sécurisé entre les franchises NFL (Harvester) et le serveur central (Nester). Léger, rapide et automatisable avec systemd.
GLPI Inventory	Permet l'inventaire automatique des machines via agents. Solution libre, très utilisée, facile à intégrer.
GLPI (gestion des tickets)	Assure le suivi des incidents, des demandes de support et permet un historique clair. Interface simple et adaptée à un usage en entreprise.
Veeam Backup & Replication	Outil de sauvegarde fiable et complet. Compatible Linux/Windows. Possibilité de planifier des Backup Copy, restauration souple.
Grafana + Prometheus	Couple open source performant pour superviser l'état des machines, métriques réseau et services critiques. Tableaux de bord personnalisés.
Développement de l'application web Nester	Interface interne de visualisation des données envoyées par les Harvesters. Codée en Python (Flask), légère, personnalisable, centralisée.
Développement du Harvester	Agent distant développé en Python permettant d'effectuer des scans réseau, mesurer la latence, remonter les rapports en JSON. Adapté aux franchises clientes.
Cluster Web (HAProxy + NGINX)	3 serveurs web reliés à un load balancer HAProxy en round-robin. NGINX en frontal. Assure la haute disponibilité, la répartition de charge et la sécurité de l'interface web.
Plateforme GitLab	Plateforme GitLab déployée pour centraliser le code source (Harvester/Nester), gérer les dépôts et les utilisateurs. Assure traçabilité et collaboration, avec une organisation pérenne pour l'intégration future de franchises.

5. Mise en œuvre technique

5.1 Fondation de l'infrastructure de base

La mise en œuvre du projet Seahawks Monitoring a nécessité la création d'une fondation technique robuste, assurant à la fois la fiabilité du stockage, la segmentation réseau et la disponibilité des services de base (DHCP et DNS). Cette étape a constitué le socle sur lequel se sont appuyés l'ensemble des déploiements ultérieurs.

5.1.1 Configuration du stockage ZFS

Le serveur Proxmox repose sur un système de fichiers ZFS configuré en RAID-Z3 sur 8 disques de 300 Go, offrant 2,39 To de capacité utile et une tolérance de panne de trois disques simultanés. Afin d'organiser efficacement les données, plusieurs pools ZFS ont été créés :

- Rpool : pool système principal utilisé par Proxmox.
- Zfs-seahawks : pool de production hébergeant les disques des machines virtuelles du projet.
- Zfs-backup : pool séparé dédié aux sauvegardes locales.
- Local-zfs : pool secondaire pour tests et stockage d'images ISO.

Ce choix garantit une meilleure résilience, l'utilisation de snapshots rapides et la vérification automatique de l'intégrité des données.

5.1.2 Architecture réseau sous Proxmox

L'architecture réseau repose sur plusieurs ponts réseau (vmbr) configurés dans Proxmox. Chaque bridge correspond à une zone fonctionnelle distincte :

Bridge	Interface	Réseau IP	Usage
vmbr0	eno1	10.25.3.0/20	Réseau WAN (connexion Internet)
vmbr1	-	10.10.10.0/24	Réseau interne principal (site Roubaix)
vmbr2	-	192.168.60.0/24	Réseau du siège social (Kansas City)
vmbr3	-	172.16.30.0/24	Réseau isolé des franchises NFL
Vmbr4	-	10.10.12.0/24	Réseau DMZ (serveurs exposés : HAProxy, Nester)

Cette segmentation facilite la mise en place des VLANs et permet d'appliquer des règles de sécurité différenciées via pfSense.

5.1.3 Plan d'adressage IP

Un plan d'adressage structuré a été défini afin de distinguer clairement chaque site et chaque service :

- Roubaix (LAN interne) : 10.10.10.0/24, héberge les services centraux (GLPI, Veeam, GitLab, Prometheus, Grafana).
- Kansas (siège social) : 192.168.60.0/24, avec serveur DHCP/DNS et poste de support.
- NFL (franchises simulées) : 172.16.30.0/24.
- DMZ Roubaix : 10.10.12.0/24, isolant les serveurs exposés (HAProxy, Nester).
- VPN WireGuard : 10.10.20.0/24, permettant la connexion sécurisée des Harvesters au datacenter.

Ce plan garantit une gestion claire, simplifie la supervision et limite les risques de conflits d'adresses.

5.1.4 Services réseaux de base (DHCP/DNS)

Dans l'infrastructure Seahawks Monitoring, une machine virtuelle Alpine Linux a été déployée pour assurer les rôles de serveur DHCP et de serveur DNS interne sur le réseau de la franchise Kansas (192.168.60.0/24).

Le service DHCP, installé avec udhcpd, attribue automatiquement les adresses IP aux machines clientes dans la plage 192.168.60.100 à 192.168.60.150. Il définit également le routeur par défaut (192.168.60.1) et le serveur DNS local (192.168.60.2), garantissant une configuration réseau homogène et centralisée.

Le service DNS, assuré par Bind9, a été configuré en tant que résolveur récursif, redirigeant les requêtes vers des serveurs DNS publics tels que Google (8.8.8.8) et Cloudflare (1.1.1.1). Il écoute sur le sous-réseau local 192.168.60.0/24 afin d'être accessible à toutes les machines clientes de Kansas et d'assurer une résolution de noms fiable et rapide.

Grâce à cette configuration, les clients du réseau Kansas bénéficient d'une attribution IP automatisée et d'une résolution de noms interne efficace, renforçant la stabilité et la continuité des services.

5.1.5 Déploiement des machines virtuelles

Au total, 18 machines virtuelles ont été déployées pour couvrir les différents rôles nécessaires au projet (services Roubaix, DMZ, siège Kansas, réseau NFL, bastion, etc.). Chaque VM a été intégrée dans le plan d'adressage IP et affectée à son sous-réseau en fonction de son rôle.

Annexes :

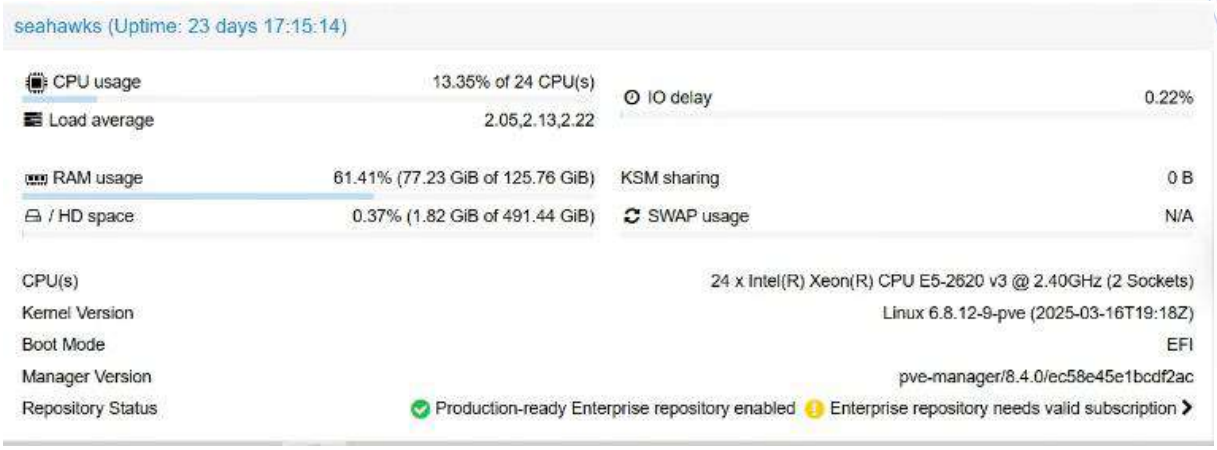

Figure 4 Spécifications techniques du serveur Proxmox (CPU, RAM, stockage ZFS, version ...)

Site	Équipement	Interface	Adresse IP	Masque de	OS	Notes
Roubaix	pfsens-roubaix	WAN	10.25.3.47 (DHCP)	/20	FreeBSD	Connexion vers l'extérieur (Internet/cloud VPN)
		LAN	10.10.10.1	/24		Réseau interne du site de Roubaix (Datacenter)
		DMZ	10.10.12.1	/24		Réseau DMZ pour le cluster web
	vm-lb-nester	Interface	10.10.12.2	/24	Debian12	Répartiteur de charge (Load Balancer) pour le cluster web
	vm-nester1	Interface	10.10.12.7	/24	Ubuntu	Serveur Seahawks Nester 1 (Cluster web)
	vm-nester2	Interface	10.10.12.8	/24	Ubuntu	Serveur Seahawks Nester 2 (Cluster web)
	vm-nester3	Interface	10.10.12.9	/24	Ubuntu	Serveur Seahawks Nester 3 (Cluster web)
	vm-gitlab	Interface	10.10.10.3	/24	RockyLinux	Hébergement et gestion des dépôts du projet Seahawks
	vm-db-nester	Interface	10.10.10.4	/24	AlmaLinux	SGBD pour Nester Seahawks
	vm-graf-prom-loki	Interface	10.10.10.5	/24	ArchLinux	Supervision (Grafana/Prometheus/Loki)
	vm-backupSrv	Interface	10.10.10.10	/24	WinSrv2019	Serveur Veeam Backup et Réplication (Interface de gestion)
	vm-rebond	Interface	10.10.30.13	/24	Win10	Poste Technicien dans le LAN de Roubaix
	vm-dépôt linux	Interface	10.10.10.11	/24	Debian12	Serveur stockage sauvegrades
	vmBackupCopy	interface	10.10.10.20	/24	Debian12	Serveur copie des sauvegardes
Kansas city	pfsens-kansas	WAN	10.25.3.27 (DHCP)	/20	FreeBSD	Connexion vers l'extérieur (Internet/cloud VPN)
		LAN	192.168.60.1	/24		Réseau interne du site de Kansas City (Siège social)
	vm-dns-dhcp	Interface	192.168.60.2	/24	AlpineLinux	Serveur DNS + DHCP siège
	vm-glpi	Interface	192.168.60.20	/24	Debian12	Ticketing/Inventaire GLPI au siège
	vm-wintech	Interface	192.168.60.13	/24	Win10	VM rebond support
NFL	pfsens-NFL	WAN	10.25.3.28 (DHCP)	/20	FreeBSD	Connexion vers l'extérieur (Internet)
		LAN	172.16.30.1	/24		Réseau interne du site de la Franchise NFL
	vm-harvester	Interface	172.16.30.10	/24		Client de supervision (franchise)
VPN WireGuard	pfSenseRbx	Wg0	10.10.20.1	/24		VPN Client nomade- côté routeur
VPN WireGuard	vm-harvester	Wg0	10.10.20.2	/24		VPN Client nomade-côté client

Figure 5 Plan d'adressage IP structuré selon les sites et les services

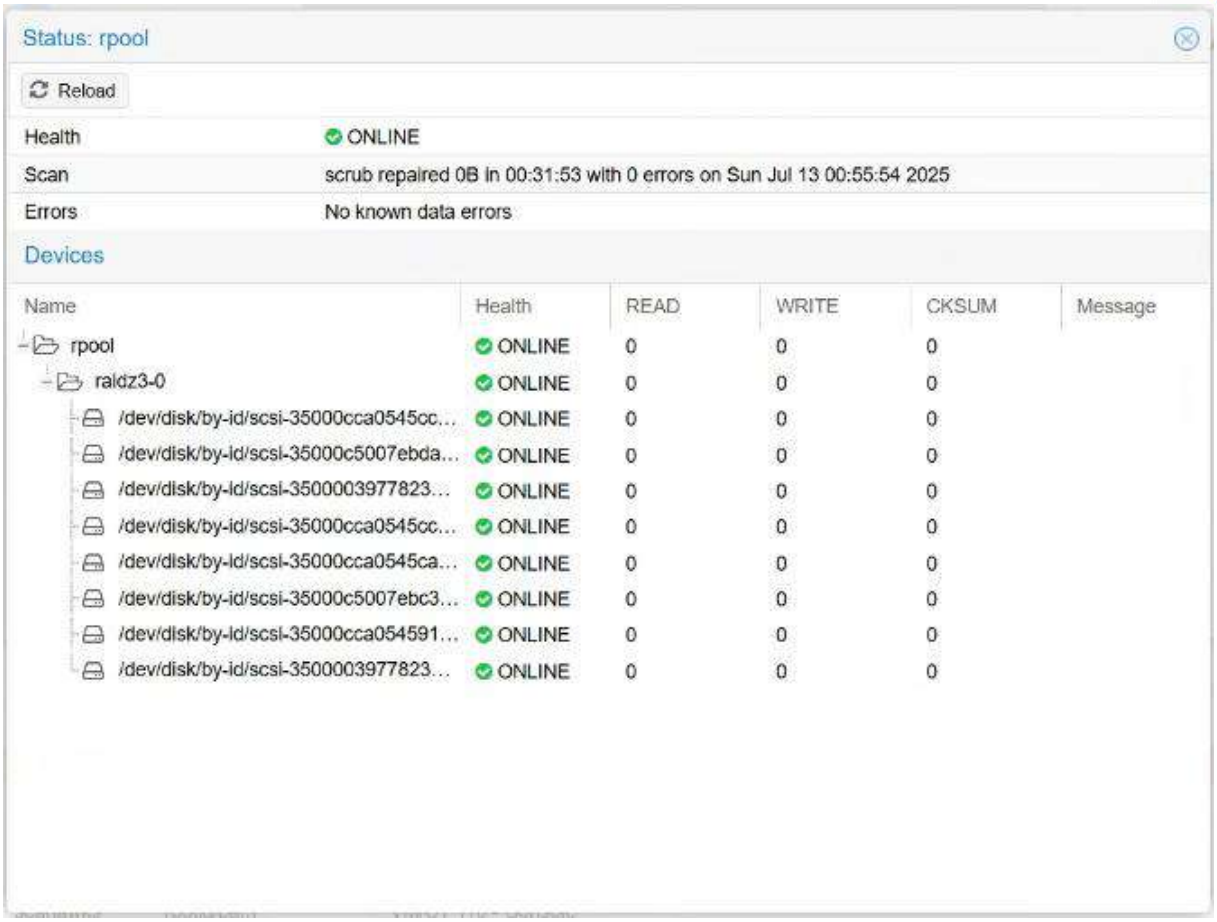

Figure 7 Configuration du pool ZFS (RAID-Z3) sur 8 disques

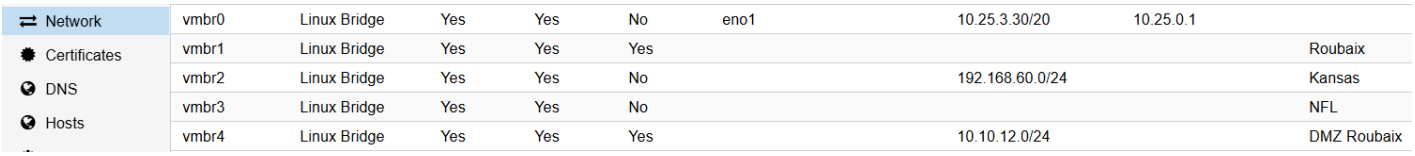

Figure 6 Topologie réseaux sous Proxmox

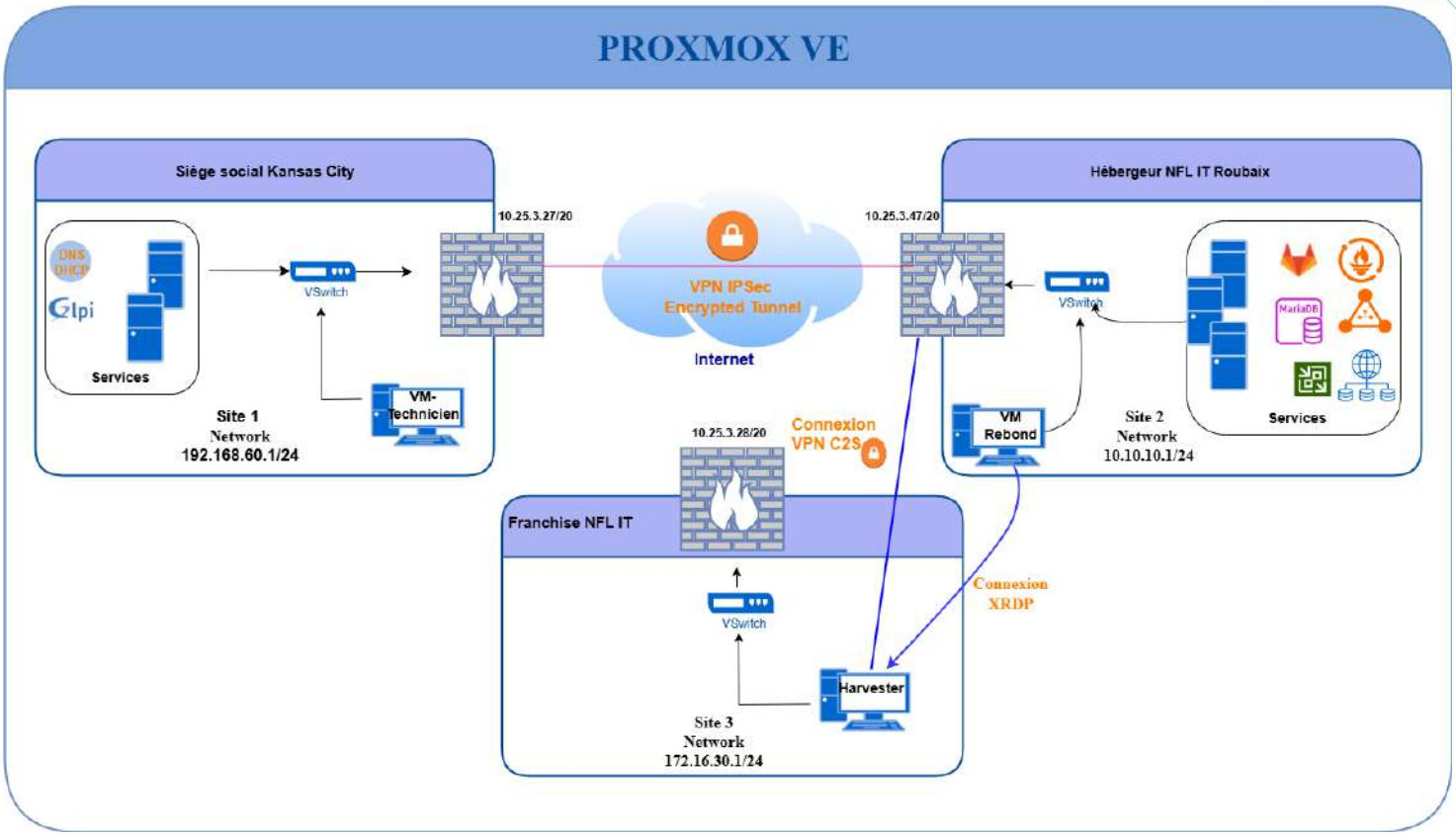

Figure 8 Schéma de l'infrastructure SeaHawks Monitoring

5.2 – Sécurisation de l'infrastructure avec pfSense

Dans le projet SeaHawks Monitoring, pfSense constitue la brique de sécurité réseau centrale, déployée sur chaque site (Roubaix, Kansas et franchises). Il assure à la fois les fonctions de pare-feu, de routage, de segmentation réseau et de mise en place de tunnels VPN sécurisés. Son utilisation a permis de concevoir une infrastructure résiliente, cloisonnée et évolutive.

5.2.1 Pare-feu et filtrage réseau

Des règles strictes de filtrage ont été définies sur chaque pfSense, selon le principe du moindre privilège :
- Blocage par défaut de tout trafic non autorisé.
- Ports critiques (SSH 22, HTTP/HTTPS 80/443) restreints aux seules IP ou VLAN d'administration.
- Flux spécifiques (Prometheus, Grafana, GLPI, Veeam…) autorisés uniquement sur des plages IP précises.
- Segmentation par interface (LAN, VPN, DMZ) pour isoler les flux internes des flux exposés.

5.2.2 Sécurisation des accès aux services (HTTP / SSH)

Afin de limiter les attaques externes et internes :
- Accès SSH renforcés par l'usage d'utilisateurs non-root, clés SSH et filtrage IP.
- Politique de mots de passe complexes pour les comptes d'administration.
- Interfaces web protégées (pfSense, GLPI, Grafana, Veeam) accessibles uniquement via le VLAN 10 et le VPN S2S.

5.2.3 VPN site-to-site IPsec (Roubaix ↔ Kansas)

Un tunnel IPsec permanent a été établi entre les sites principaux Roubaix et Kansas, interconnectant les sous-réseaux 10.10.10.0/24 et 192.168.60.0/24.
- Chiffrement robuste (AES-256, SHA-256).
- Authentification par clé partagée.
- Reconnexion automatique en cas de coupure.
- Autorisation des flux nécessaires (GLPI, Grafana, Veeam, supervision…).

5.2.4 VPN client-to-site WireGuard (Franchises ↔ Roubaix)

Chaque Harvester déployé en franchise établit un tunnel chiffré vers Roubaix à l'aide de WireGuard.
- Interface dédiée wg0 attribuant une IP VPN (10.10.20.0/24).
- Connexion automatisée au démarrage des sondes Harvester.
- Règles spécifiques sur pfSense (port 51820/UDP ouvert, filtrage par IP et ports).
- Isolation des flux VPN via une interface dédiée WG_VPN.

5.2.5 Intérêt de cette architecture

Cette approche apporte plusieurs bénéfices :
- Cloisonnement strict entre les différents environnements (LAN, VPN, DMZ).
- Sécurisation des échanges inter-sites et franchises.
- Administration centralisée et homogène via pfSense.
- Résilience : continuité de service même en cas de coupure de tunnel (reconnexion auto).
- Extensibilité : ajout de nouvelles franchises via WireGuard sans remettre en cause la configuration globale.

5.2.6 Sécurisation de l'accès aux services internes (NAT/PAT sur pfSense)

Dans le cadre du projet Seahawks Monitoring, il était nécessaire de rendre l'application Nester, hébergée sur le réseau interne (10.10.10.0/24), accessible depuis l'extérieur tout en maintenant un niveau de sécurité élevé.

Pour cela, le pare-feu pfSense a été configuré afin de contrôler et rediriger le trafic réseau grâce à deux mécanismes complémentaires :

- NAT sortant (Outbound NAT) : permet aux machines internes d'accéder à Internet en utilisant l'adresse IP publique du pare-feu. Cela masque les adresses IP privées et renforce la sécurité en évitant d'exposer directement le réseau interne.

- Port Forwarding (PAT) : permet de rediriger uniquement certains ports de l'IP publique vers des services internes. Dans notre cas, le port 80 (HTTP) a été redirigé vers le serveur Nester (10.10.12.2).

Ces configurations garantissent que :
- le trafic sortant est centralisé et filtré par le pare-feu,
- seuls les services strictement nécessaires (HTTP) sont exposés depuis l'extérieur,
- l'architecture reste cloisonnée entre WAN et LAN.

Afin de limiter les risques, plusieurs mesures de sécurité complémentaires ont été mises en place :
- l'exposition est réduite au strict minimum (port 80),
- les interfaces d'administration (pfSense, SSH, etc.) ne sont accessibles qu'en interne via LAN ou VPN,
- les journaux pfSense sont régulièrement analysés afin de détecter toute tentative d'accès non autorisée.

Cette approche illustre l'équilibre recherché entre accessibilité des services pour les utilisateurs externes et sécurisation de l'infrastructure interne grâce au rôle central du pare-feu pfSense.

5.2.7 – Mise en place des VLANs sur Roubaix

Afin de renforcer la segmentation logique du réseau interne et d'appliquer une politique de moindre privilège, plusieurs VLANs ont été déployés sur le site de Roubaix. L'interface LAN physique a été configurée en mode VLAN Aware, permettant de définir des réseaux virtuels isolés tout en conservant la même infrastructure physique.

Trois VLANs ont été créés :
- VLAN 10 – Services (10.10.10.0/24) : il regroupe les services critiques de l'infrastructure, tels que la supervision (Prometheus, Grafana), la sauvegarde (Veeam), la base de données, GitLab et GLPI.
- VLAN 30 – Bastion (10.10.30.0/24) : ce réseau héberge la VM rebond (Jump Server) qui constitue le point d'accès intermédiaire pour la télémaintenance.
- VLAN 40 – IT (10.10.40.0/24) : réservé aux postes techniques des administrateurs et techniciens, ce VLAN permet de séparer clairement l'espace de travail des utilisateurs de l'infrastructure des autres zones sensibles.

Cette organisation garantit un cloisonnement strict entre les environnements de production, d'administration et de support. Elle permet également d'appliquer des règles de firewall granulaires sur pfSense, limitant la surface d'attaque et réduisant les risques de compromission transversale entre les services.

5.2.8 – Mise en place d'une DMZ

Dans le cadre de la sécurisation de l'accès aux services exposés à Internet, une zone démilitarisée (DMZ) a été mise en place. Cette DMZ, adressée en 10.10.12.0/24, héberge les composants accessibles depuis l'extérieur :

- le serveur HAProxy (10.10.12.2) jouant le rôle d'équilibreur de charge,
- les serveurs Nester01 à Nester03 (10.10.12.7-9) qui assurent l'hébergement de l'application web Seahawks.

Le pare-feu pfSense assure l'isolation de cette DMZ par une interface réseau dédiée. Les règles de sécurité définissent :
- une ouverture limitée au seul port nécessaire (80 depuis Internet vers HAProxy),
- des flux internes contrôlés (ex. accès des Nester vers la base de données en VLAN Services),
- une interdiction stricte des communications directes vers les VLANs d'administration et techniques.

Cette architecture offre plusieurs bénéfices :
- Réduction de l'exposition : seuls les serveurs nécessaires à la publication web sont accessibles depuis l'extérieur.
- Confinement en cas de compromission : une éventuelle attaque sur un serveur de la DMZ ne permet pas d'accéder directement aux réseaux internes.
- Contrôle des flux : les communications entre la DMZ et le reste du système sont surveillées et limitées à des besoins précis.

La mise en place de cette DMZ s'inscrit dans la logique de défense en profondeur déjà amorcée par le cloisonnement en VLANs et la configuration de règles de firewall restrictives.

Annexes :

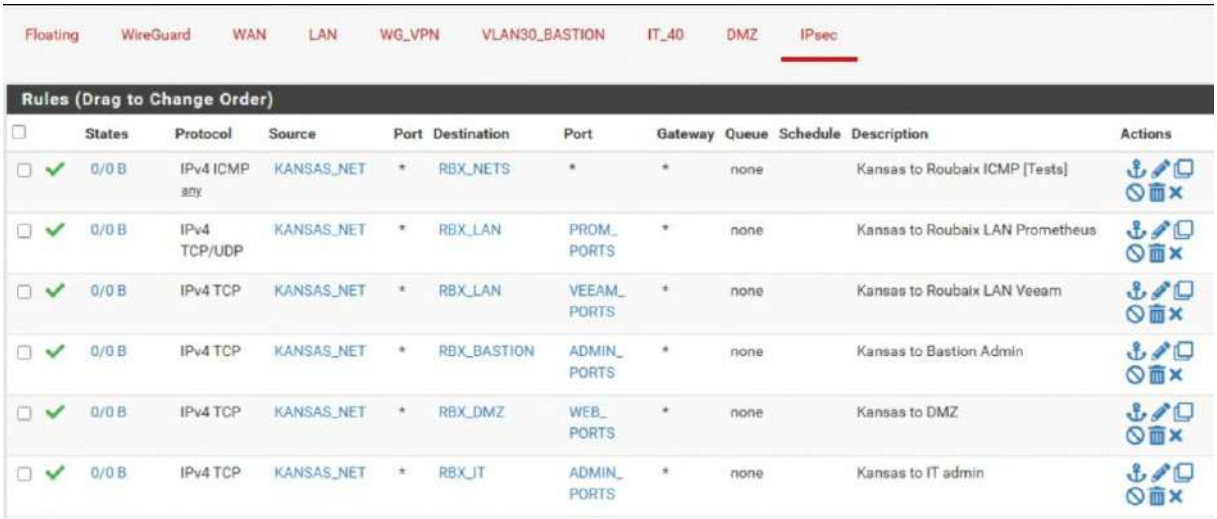

Figure 9 Application du principe du moindre privilège : seuls les flux nécessaires au fonctionnement de l'infrastructure sont autorisés

Figure 10 Affichage d'un client Harvester connecté via VPN client -to-site (interface tun_wg0, clé publique, IP attribuée)

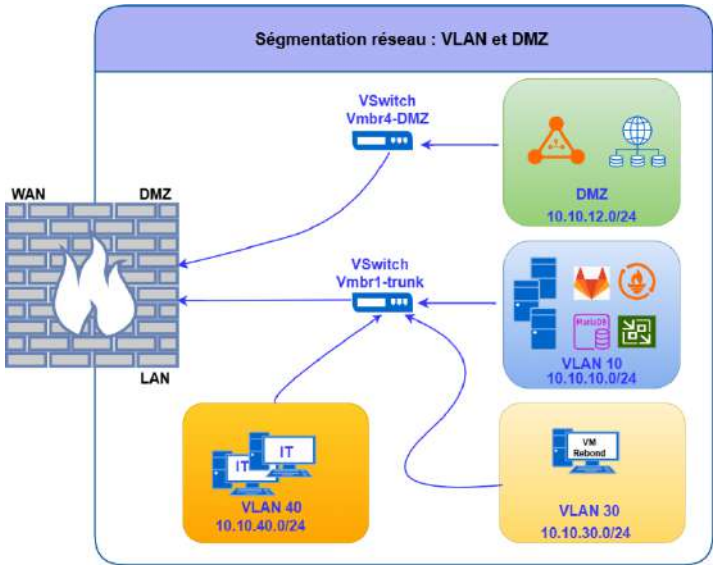

Figure 11 Schéma des VLANs

Firewall / Aliases / IP

IP Ports URLs All

Firewall Aliases IP

Name	Type	Values	Description	Actions
KANSAS_NET	Network(s)	192.168.60.0/24		
RBX_BASTION	Network(s)	10.10.30.0/24		
RBX_DMZ	Network(s)	10.10.12.0/24		
RBX_IT	Network(s)	10.10.40.0/24		
RBX_LAN	Network(s)	10.10.10.0/24		
RBX_NETS	Network(s)	10.10.10.0/24, 10.10.12.0/24, 10.10.30.0/24, 10.10.40.0/24	All of network in Roubaix	

Figure 12 Définition des alias IP sur pfSense

Firewall / Aliases / Ports

IP Ports URLs All

Firewall Aliases Ports

Name	Type	Values	Description	Actions
ADMIN_PORTS	Port(s)	22, 3389		
PROM_PORTS	Port(s)	9100, 9184		
VEEAM_PORTS	Port(s)	2500, 6160:6200, 10001:10099, 10006, 111:2049		
WEB_PORTS	Port(s)	80, 443		

Figure 13 Définition ds alias de ports applicatifs

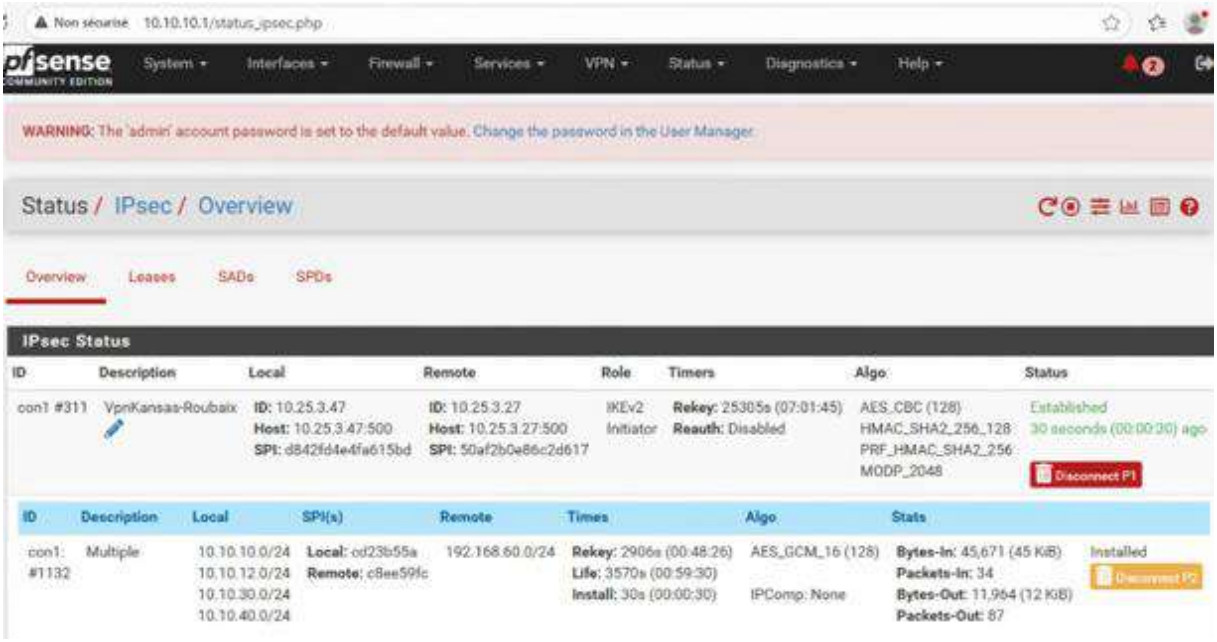

Figure 14 Configuration visuelle de l'établissement du tunnel sécurisé entre Roubaix et Kensas (indicateur vert actif)

```
(harvester) admin12@debHARVESTER:~$ ping 10.10.20.1
PING 10.10.20.1 (10.10.20.1) 56(84) bytes of data.
64 bytes from 10.10.20.1: icmp_seq=1 ttl=64 time=2.06 ms
64 bytes from 10.10.20.1: icmp_seq=2 ttl=64 time=2.36 ms
64 bytes from 10.10.20.1: icmp_seq=3 ttl=64 time=2.18 ms
64 bytes from 10.10.20.1: icmp_seq=4 ttl=64 time=1.83 ms
64 bytes from 10.10.20.1: icmp_seq=5 ttl=64 time=2.32 ms
64 bytes from 10.10.20.1: icmp_seq=6 ttl=64 time=2.30 ms
^C
--- 10.10.20.1 ping statistics ---
7 packets transmitted, 6 received, 14.2857% packet loss, time 6010ms
t min/avg/max/mdev = 1.832/2.174/2.359/0.183 ms
arvester) admin12@debHARVESTER:~$
```

Figure 15 Ping réussi entre un Harvester en franchise et le datacenter de Roubaix à travers le tunnel WireGuard

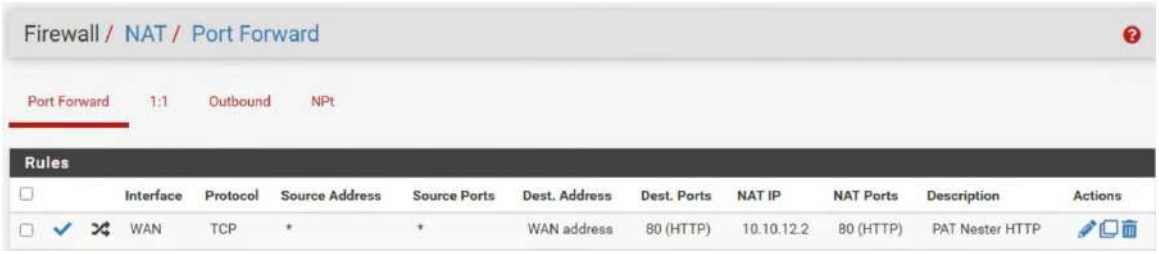

Figure 16 Configuration de la régle NAT/PAT sur pfSense : redirection du port HTTP du WAN vers le serveur interne Nester

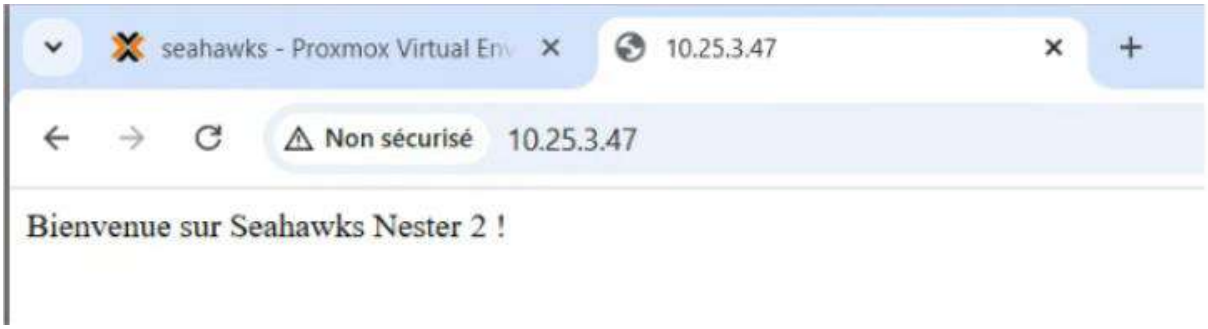

Figure 17 Résultat de la redirection : accès externe au serveur Nester via pfSense

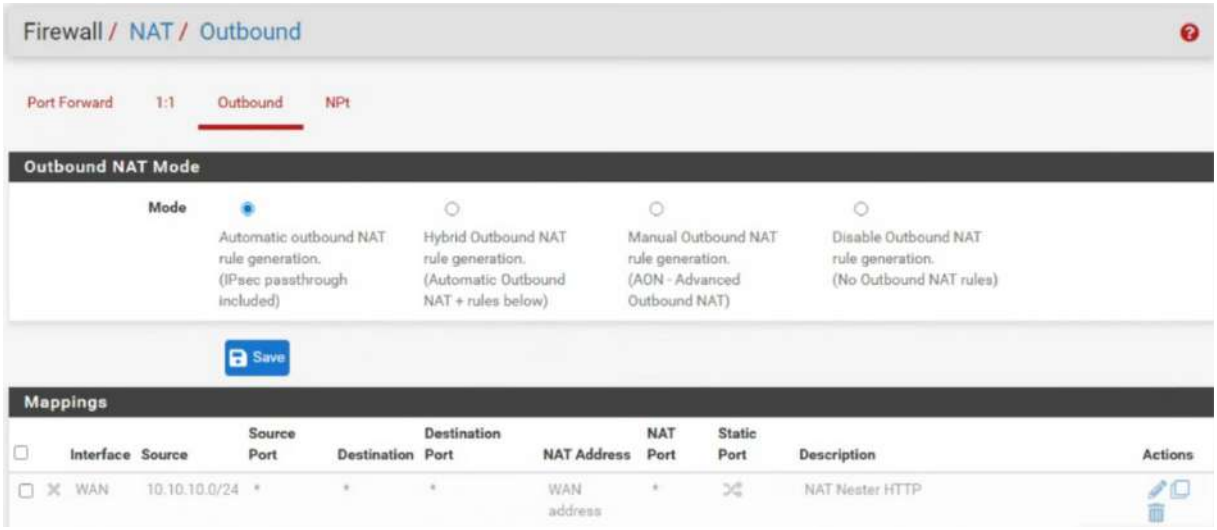

Figure 18 Paramètres du NAT sortant de pfSense en mode automatique, réseau local 10.10.10.0/24 vers Ineternet

5.3 – Sauvegarde avec Veeam Backup & Replication

La mise en place du système de sauvegarde a été réalisée à l'aide de la solution Veeam Backup & Replication (Community Edition), installée sur un serveur Windows Server 2019. Ce choix s'est imposé pour sa souplesse, sa compatibilité Linux/Windows, et sa capacité à fonctionner sans agent préinstallé.

Objectif

Assurer la sauvegarde fiable, automatisée et redondante des machines virtuelles critiques de l'infrastructure Seahawks, y compris celles situées à distance ou derrière des tunnels VPN.

5.3.1 Déploiement de l'environnement Veeam

L'infrastructure de sauvegarde repose sur une architecture à deux niveaux :
- Un dépôt principal (VM depot-linux) utilisé pour les sauvegardes quotidiennes,
- Un dépôt secondaire (VM depot-backup-copy) dédié aux copies de sauvegarde (Backup Copy) pour garantir la redondance 3-2-1.

Un disque de 80 Go a été ajouté à la VM depot-linux, partitionné, formaté en ext4 et monté de manière persistante. Ce volume a été déclaré comme dépôt Veeam via SSH.

5.3.2 Gestion des sauvegardes et de la redondance

Chaque machine virtuelle est associée à un job de sauvegarde planifié, combinant une sauvegarde complète (Full) suivie de sauvegardes incrémentales.
Un job de type Backup Copy a été configuré pour répliquer automatiquement les points de restauration sur le dépôt secondaire, conformément aux bonnes pratiques de résilience.

Cas particulier :
- La VM glpi, située derrière un tunnel VPN IPsec, a été dirigée vers le dépôt local de VBR pour des raisons de performance, puis recopiée vers le dépôt secondaire pour redondance.

5.3.3 Traitement des cas spécifiques

Certaines VM comme Arch Linux ou Alpine, non compatibles avec les agents Veeam, ont été sauvegardées via une VM intermédiaire (nester) avec un partage NFS monté en local, puis incluse dans un job VBR standard.

Ce processus a permis de :
- Ne pas installer d'agent Veeam manuellement,
- Respecter les contraintes de compatibilité des systèmes non supportés,
- Conserver un niveau de sauvegarde stable et reproductible.

5.3.4 Backup sans agent préinstallé

Un avantage clé de Veeam réside dans la possibilité d'ajouter des machines en mode "Managed by backup server", permettant à VBR de déployer temporairement l'agent au moment de l'exécution du job, sans intervention manuelle.

Ce mécanisme a permis de simplifier la gestion et d'éviter les conflits de version ou de dépendances entre les systèmes.

5.3.5 Résilience, optimisation et choix stratégiques

- Les sauvegardes sont planifiées intelligemment pour éviter les surcharges nocturnes.
- Le Backup Copy n'est activé que pour les VMs critiques (GLPI, Grafana…), afin d'optimiser l'espace disque.
- La configuration de pfSense n'est pas sauvegardée via Veeam, mais via un script automatisé qui exporte chaque jour le fichier .xml contenant toute la configuration du pare-feu.

Conclusion

La solution Veeam a permis de centraliser et sécuriser efficacement les sauvegardes des différentes VMs du projet Seahawks, tout en garantissant une stratégie de redondance intelligente, adaptée aux contraintes réseau, aux OS hétérogènes, et à la faible consommation de ressources.
Elle constitue un socle fiable et maîtrisé de l'infrastructure.

Annexes :

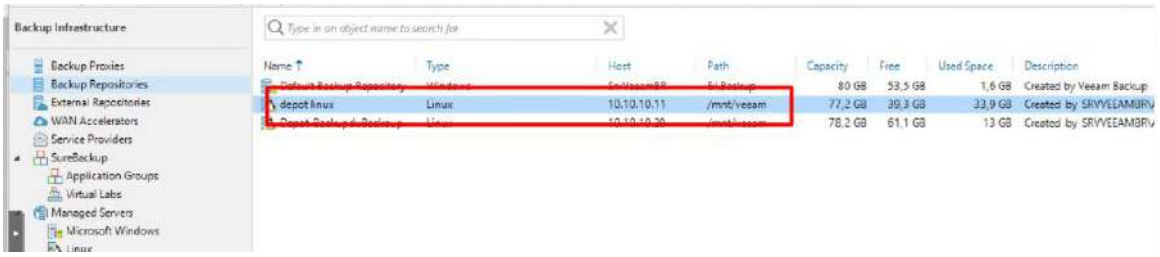

Figure 19 Intégration de la machine Linux 10.10.10.11 (dépôt Linux dédié)

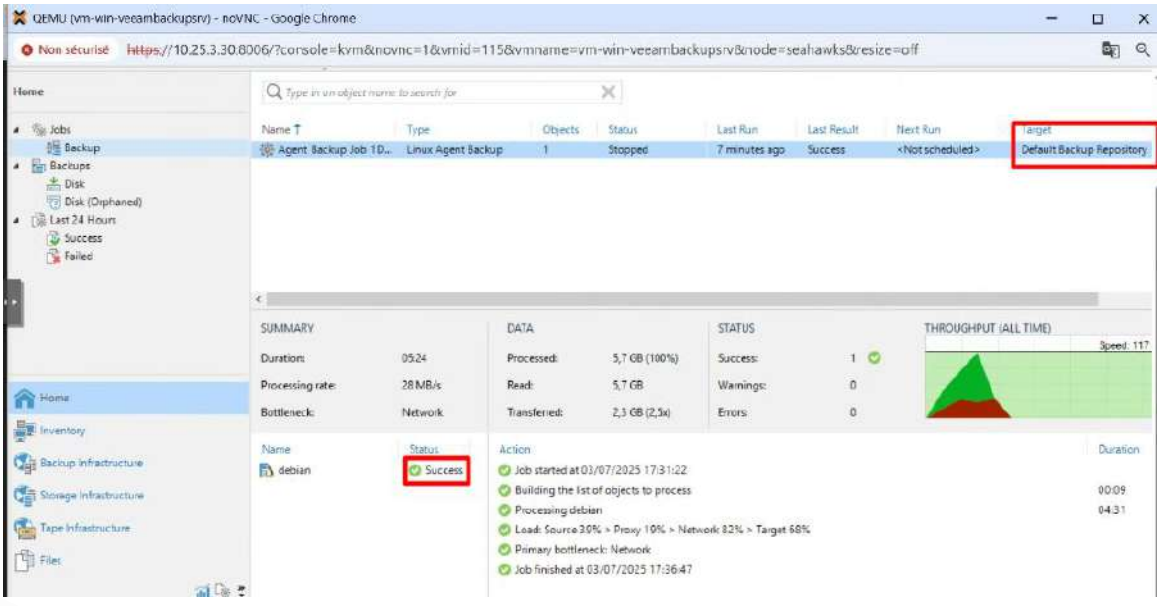

Figure 20 Déploiement du premier job de sauvegarde Veeam, stocké sur le dépôt par défaut de VBR

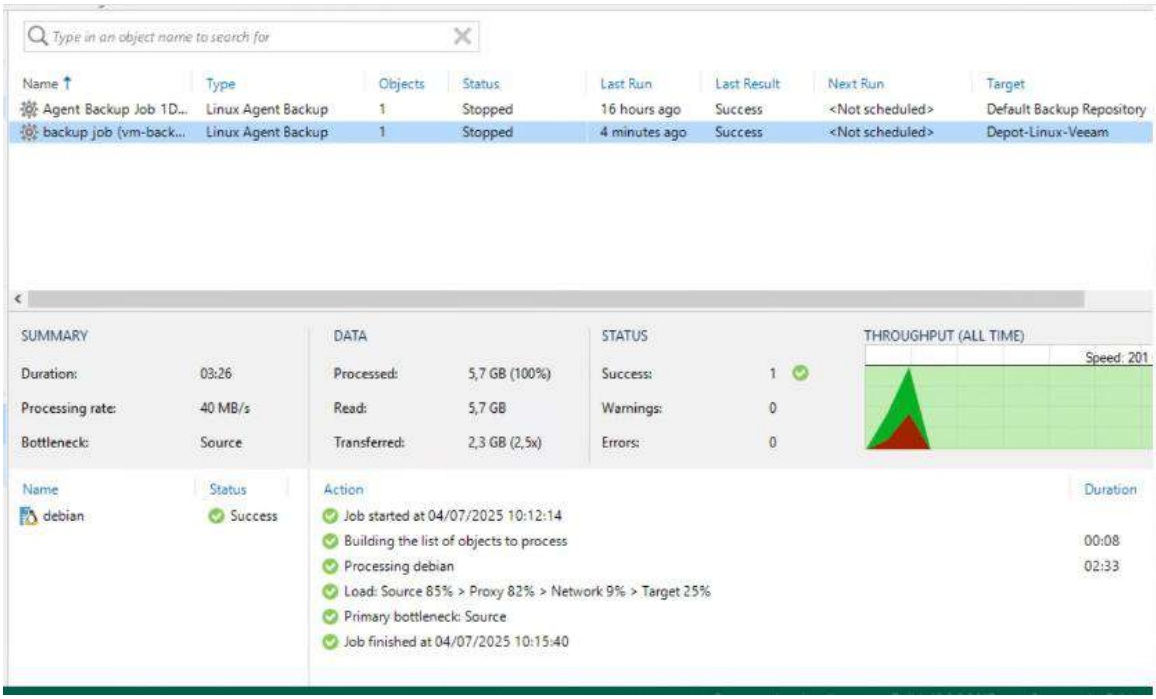

Figure 21 Job Veeam configuré pour sauvegarder une VM spécifique vers un dépôt isolé pour renforcer la résilience

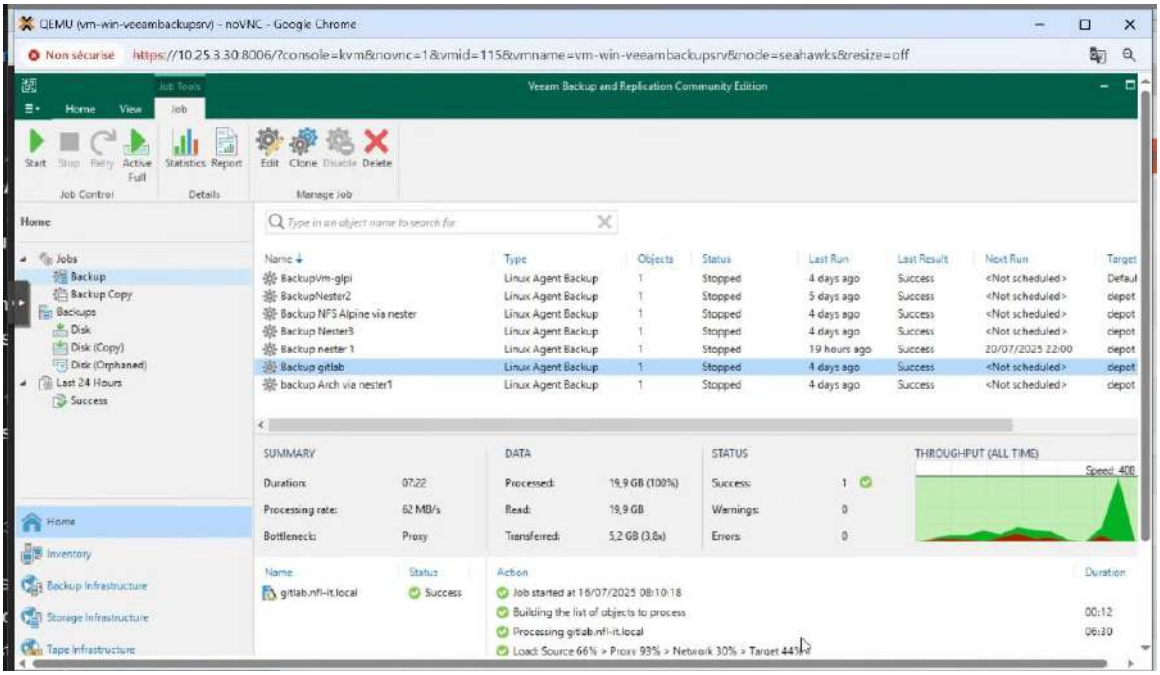

Figure 22 Vue d'ensemble des tâches de sauvegardes configurées pour protéger les services critiques de l'infra

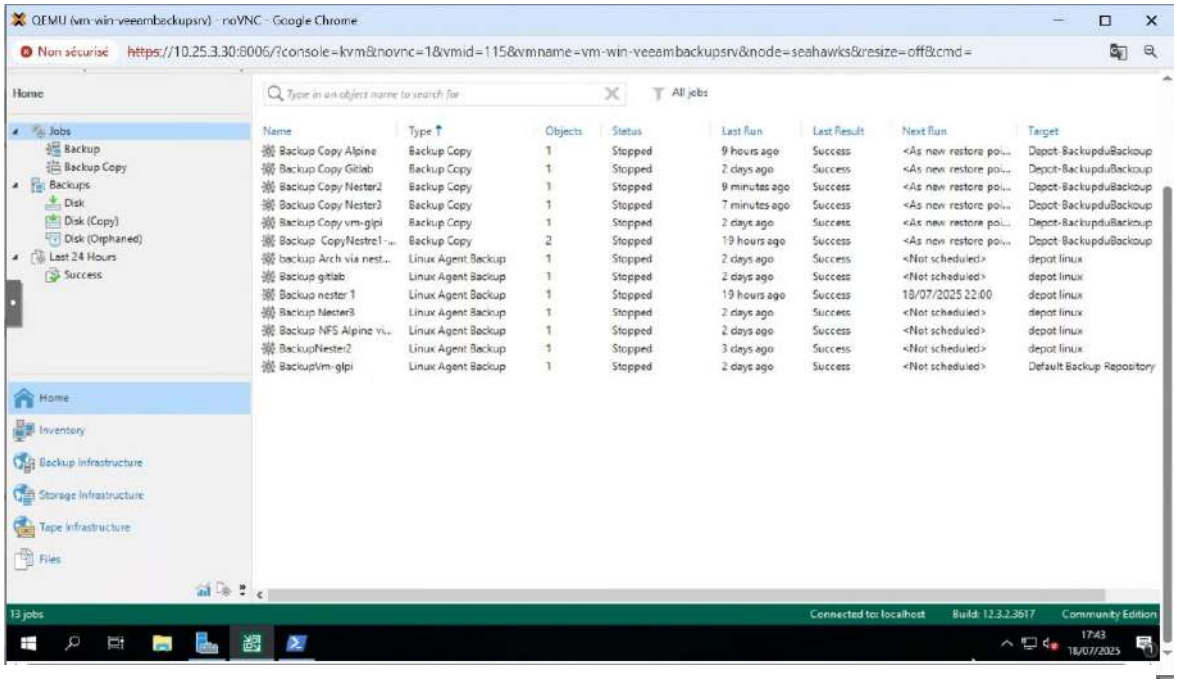

Figure 23 Vue globale des jobs Veeam

5.4 – Mise en œuvre de la Gestion des tickets avec GLPI

Le système de ticketing a été mis en place pour centraliser la gestion des incidents et des demandes internes au cours du projet Seahawks Monitoring.

5.4.1 Structure de la solution de ticketing

Le système de ticketing a été configuré pour répondre aux besoins internes de suivi : chaque anomalie, suggestion ou évolution a été consignée sous forme de ticket GLPI. Des règles ont été définies pour affecter les tickets aux bonnes entités, en fonction du type de problème (infrastructure, sauvegarde, supervision, etc.).

5.4.2 Profils et utilisateurs

Plusieurs profils ont été créés :
- Administrateur : accès total (création d'entités, modification des règles, consultation de tous les tickets).
- Technicien : traitement des tickets affectés.
- Utilisateur standard : déclaration d'un incident via l'interface simplifiée.

5.4.3 Cycle de vie d'un ticket

Chaque ticket suit un cycle classique : création > prise en charge > résolution > clôture. Des commentaires et des pièces jointes peuvent être ajoutés tout au long du processus. Une base de connaissance interne a également été alimentée à partir des tickets résolus pour servir de support documentaire.

5.4.4 Intérêt pour le projet

Le module de ticketing GLPI a permis de structurer la gestion de projet, de centraliser les remontées techniques et de faciliter la traçabilité des incidents rencontrés. Il s'est avéré être un outil d'organisation aussi bien technique qu'humain.

Annexes :

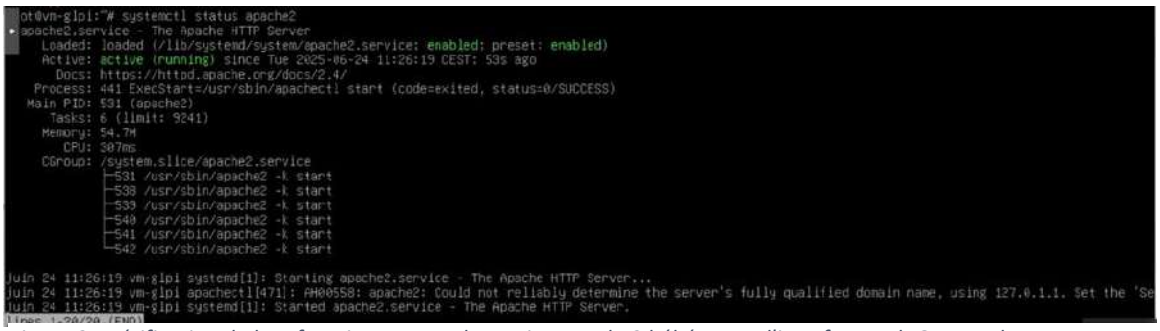

Figure 24 Vérification du bon fonctionnement du service Apache2 hébergeant l'interface web GLPI sur le serveur

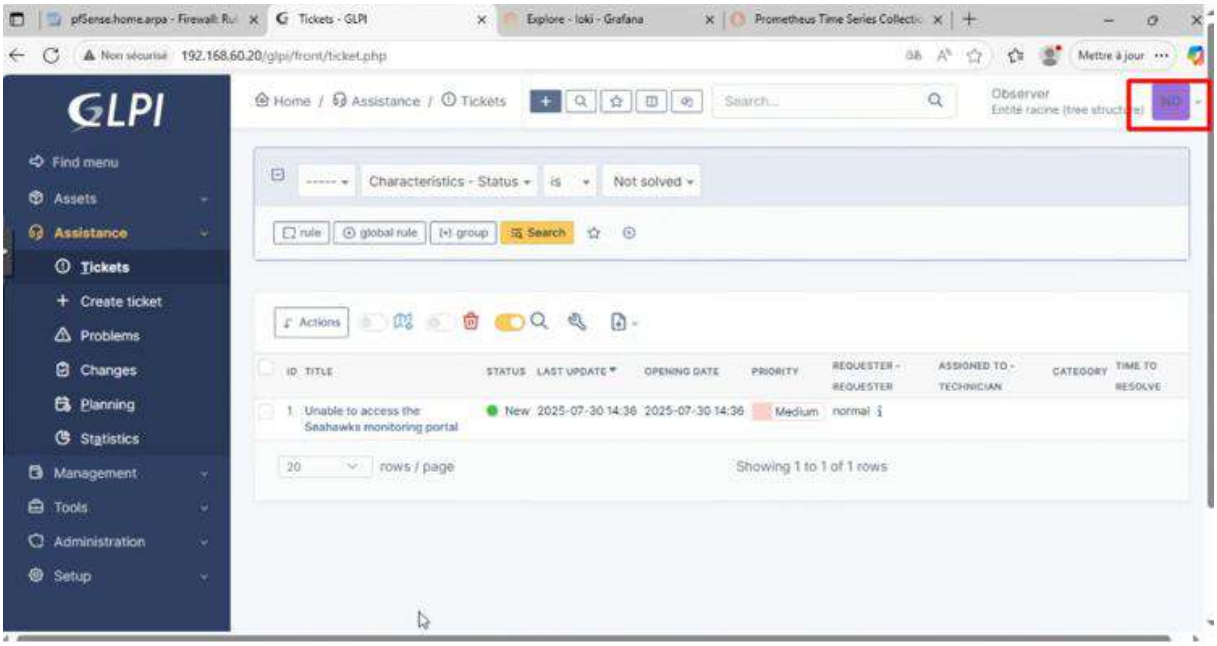

Figure 25 Interface de l'utilisateur, montrant son propre ticket en statut "New" non encore pris en charge

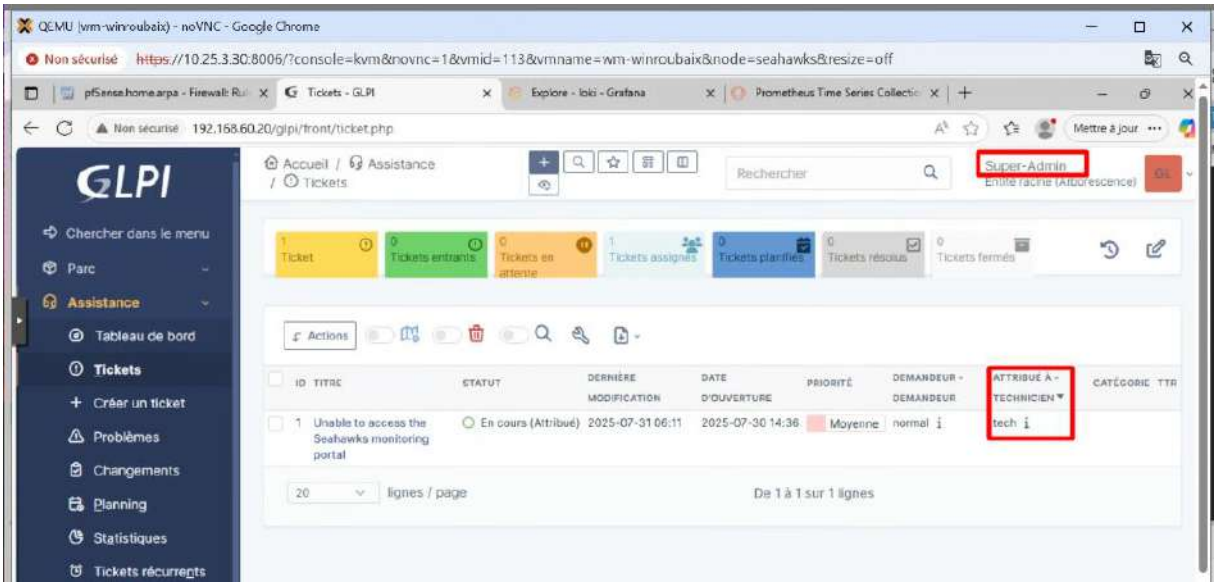

Figure 26 Vue du super-admin qui a attribué le ticket à un technicien pour traitement

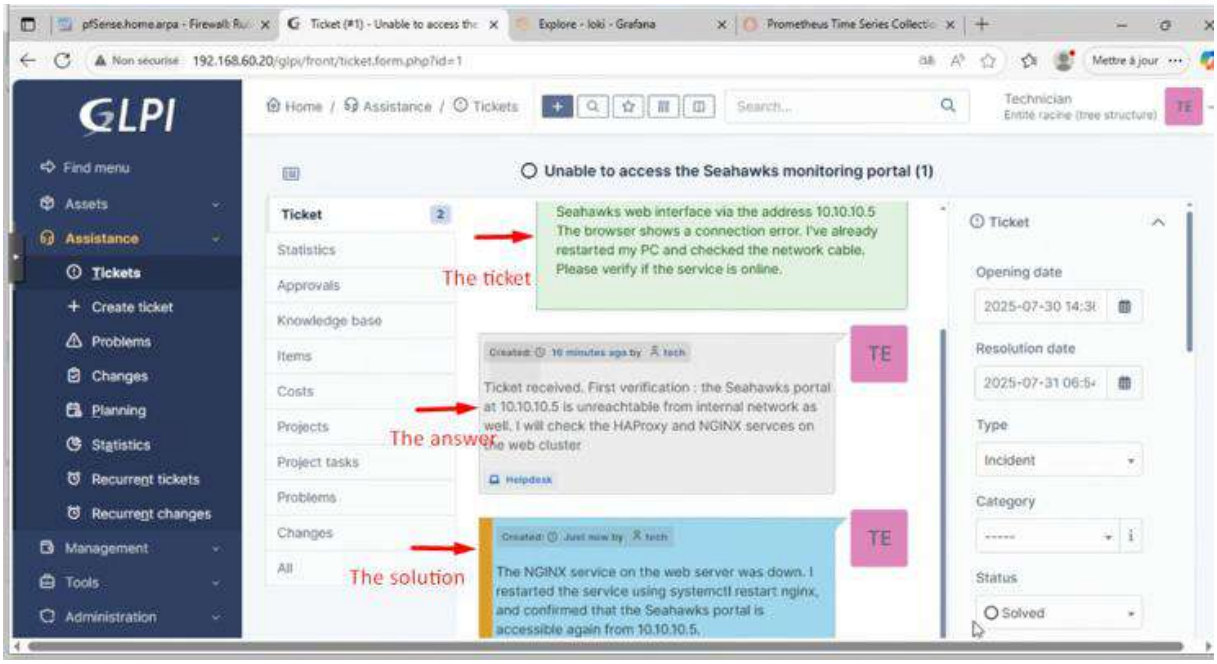

Figure 27 Détail du ticket côté technicien : on voit le message initial, la réponse technique, puis la solution apportée après diagnostic

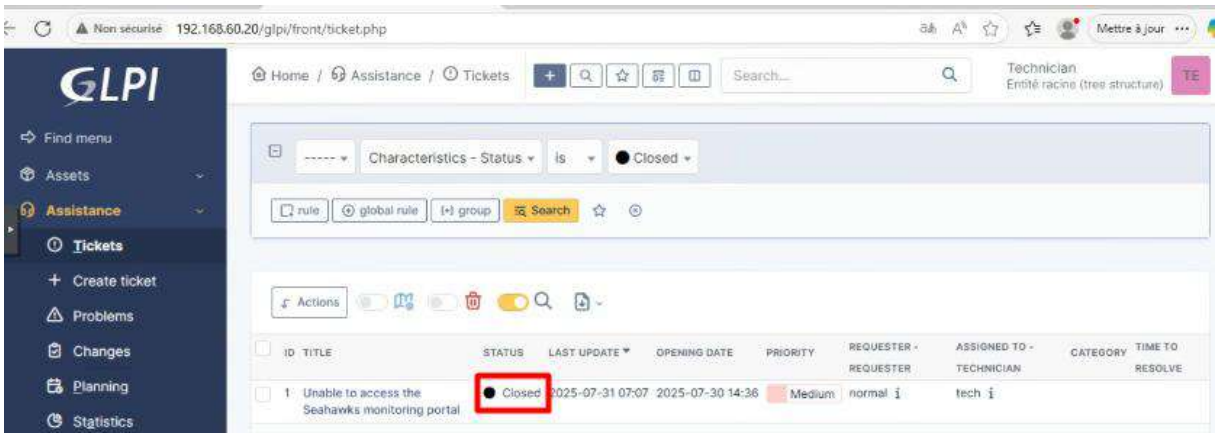

Figure 28 Ticket passé au statut Closed après validation de la solution : le cycle de traitement du ticket est terminé

5.5 – Mise en œuvre de GLPI inventaire automatisé

GLPI (Gestionnaire Libre de Parc Informatique) est une solution libre de gestion de services informatiques (ITSM). Dans le cadre du projet Seahawks Monitoring, il a été utilisé pour centraliser les informations relatives au parc matériel, organiser l'inventaire automatique des équipements, structurer le support et améliorer la visibilité de l'infrastructure.

5.5.1 Installation de GLPI et du plugin Inventory

Le serveur GLPI a été installé sur une machine Debian 12. Le plugin officiel GLPI Inventory (version 1.5.3) a été ajouté manuellement et activé via l'interface d'administration. Ce plugin permet de recevoir automatiquement les rapports d'inventaire envoyés par les agents GLPI installés sur les postes clients.

L'activation de l'inventaire s'effectue depuis l'interface : Administration > Inventaire > Configuration > Activer l'inventaire.

5.5.2 Déploiement des agents GLPI

Deux méthodes ont été utilisées selon le système d'exploitation des machines :

- Sur Debian 12 : l'agent GLPI (version 1.6.1) a été installé via un script officiel téléchargé depuis GitHub. Une tâche cron a été ajoutée pour exécuter un inventaire toutes les 30 minutes.

- Sur Alpine Linux : l'installation a nécessité plus d'adaptations, notamment la gestion des dépendances Perl et la configuration manuelle du fichier agent.cfg en JSON. Un script de démarrage automatique a été ajouté pour lancer l'agent au démarrage de la VM.

5.5.3 Inventaire des pare-feu pfSense

pfSense étant basé sur FreeBSD, il n'est pas compatible avec l'agent GLPI standard. Pour intégrer ses informations dans GLPI, une méthode alternative a été mise en place :

- Un script shell exécuté depuis une VM Alpine interroge pfSense via SSH et génère un rapport HTML.
- Ce rapport est ensuite transféré via WinSCP et rattaché manuellement à la fiche de l'équipement dans GLPI.
- Des fiches spécifiques ont été créées dans GLPI pour chaque pfSense (Roubaix, Kensas), incluant leurs adresses IP et rôles réseau.
- L'exécution peut être planifiée par cron (semi-automatique).

Script d'inventaire (extrait)

```
#!/bin/sh
# Inventaire pfSense -> rapport HTML
# VM Alpine (clé SSH déjà déployée côté pfSense)

PFSENSE_HOST="10.10.10.1"        # IP pfSense
PFSENSE_USER="admin"             # utilisateur pfSense (clé SSH recommandée)
OUTDIR="/home/inventory"
STAMP=$(date +%F_%H%M)
OUTHTML="$OUTDIR/pfsense_${PFSENSE_HOST}_${STAMP}.html"

mkdir -p "$OUTDIR"

# Collecte d'infos système & réseau
HOSTNAME=$(ssh ${PFSENSE_USER}@${PFSENSE_HOST} "hostname")
PFSENSE_VER=$(ssh ${PFSENSE_USER}@${PFSENSE_HOST} "grep product_version
/etc/version")
UNAME=$(ssh ${PFSENSE_USER}@${PFSENSE_HOST} "uname -a")
IFCONFIG=$(ssh ${PFSENSE_USER}@${PFSENSE_HOST} "ifconfig -a")
ROUTES=$(ssh ${PFSENSE_USER}@${PFSENSE_HOST} "netstat -rn")

# Export de la configuration complète
ssh ${PFSENSE_USER}@${PFSENSE_HOST} "cat /cf/conf/config.xml" >
"$OUTDIR/config_${STAMP}.xml"

# Génération d'un rapport HTML minimal lisible
cat > "$OUTHTML" <<EOF
<!DOCTYPE html>
<html lang="fr"><meta charset="utf-8">
<head><title>Inventaire pfSense - ${PFSENSE_HOST} - ${STAMP}</title>
<style>body{font-family:Arial, sans-serif}
pre{background:#f7f7f7;padding:10px;border:1px solid #ddd;}</style>
</head><body>
<h1>Inventaire pfSense</h1>
<p><strong>Date :</strong> ${STAMP}</p>
<p><strong>Adresse IP :</strong> ${PFSENSE_HOST}</p>
<h2>Informations générales</h2>
<pre>${HOSTNAME}
${PFSENSE_VER}
${UNAME}</pre>
<h2>Interfaces réseau (ifconfig)</h2>
<pre>${IFCONFIG}</pre>
<h2>Table de routage</h2>
<pre>${ROUTES}</pre>
<h2>Configuration (config.xml)</h2>
<p>Copie enregistrée : config_${STAMP}.xml</p>
</body></html>
EOF

echo "Rapport généré : $OUTHTML"
```

Planification (cron) – exemple

Exécuter le script chaque nuit à 02:00 :

```
0 2 * * * /home/inventory/inventaire_pfsense.sh
```

5.5.4 Résultats obtenus et intérêts pour l'administration

- L'ensemble des VM inventoriées sont visibles dans GLPI avec leurs caractéristiques techniques (RAM, CPU, disques, OS…)
- Les agents remontent automatiquement les mises à jour matérielles ou réseau
- Les rapports pfSense sont archivés dans GLPI, permettant une traçabilité complète
- L'inventaire est organisé par site et par rôle, facilitant l'administration centralisée

Annexes :

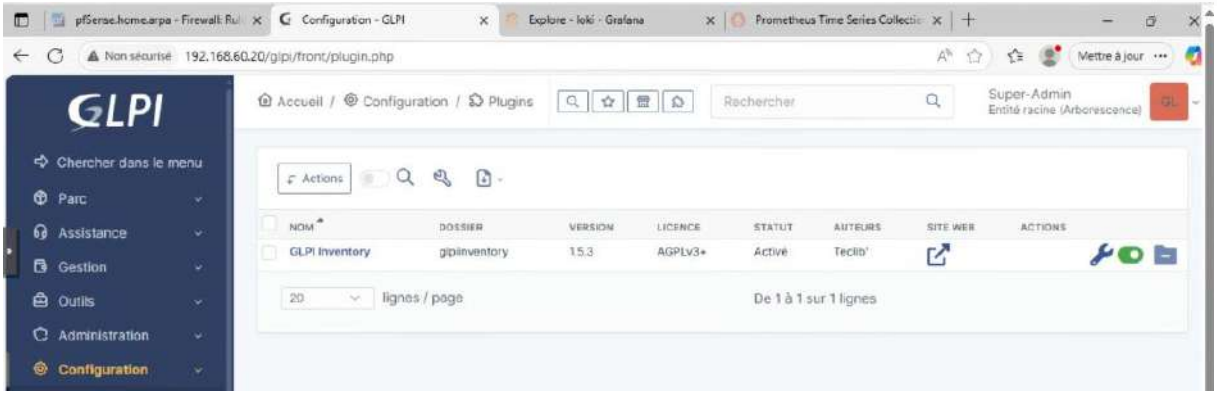

Figure 29 Activation du plugin GLPI Inventory pour la collecte automatique des agents

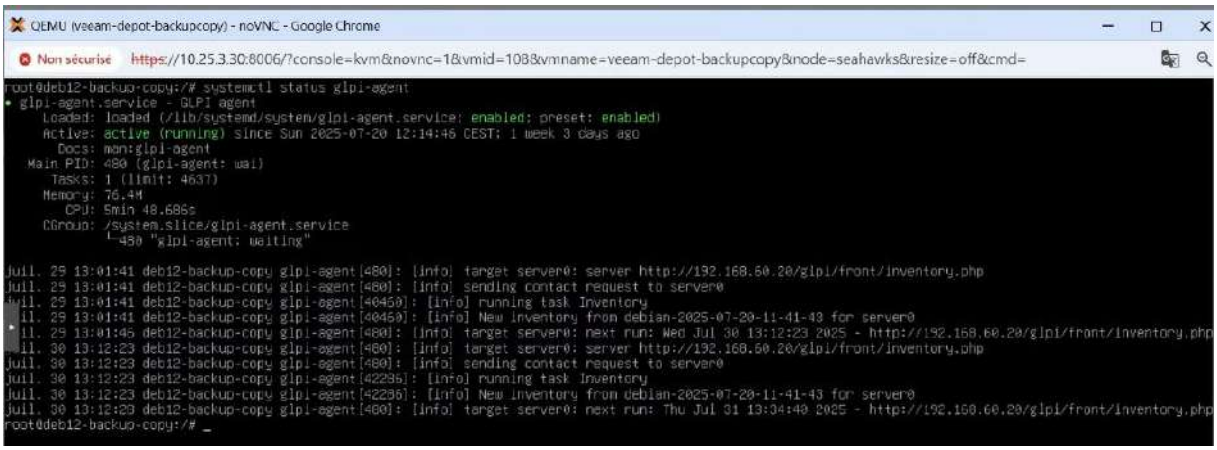

Figure 30 Installation de l'agent GLPI sur une machine cliente

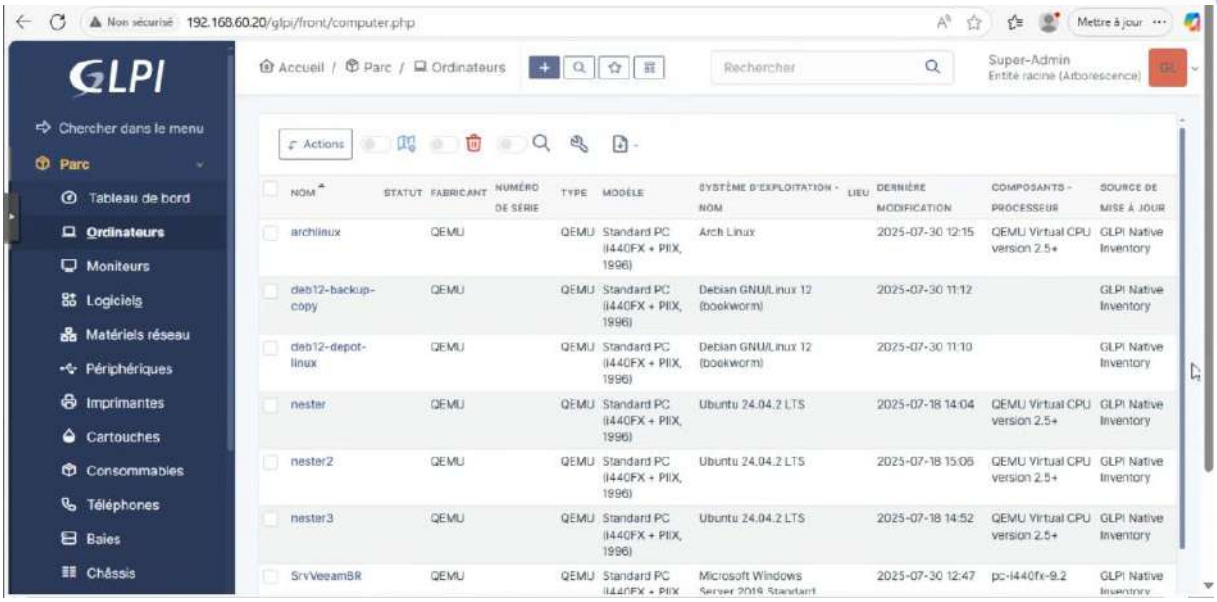

Figure 31 Liste des machines inventoriées automatiquement dans GLPI

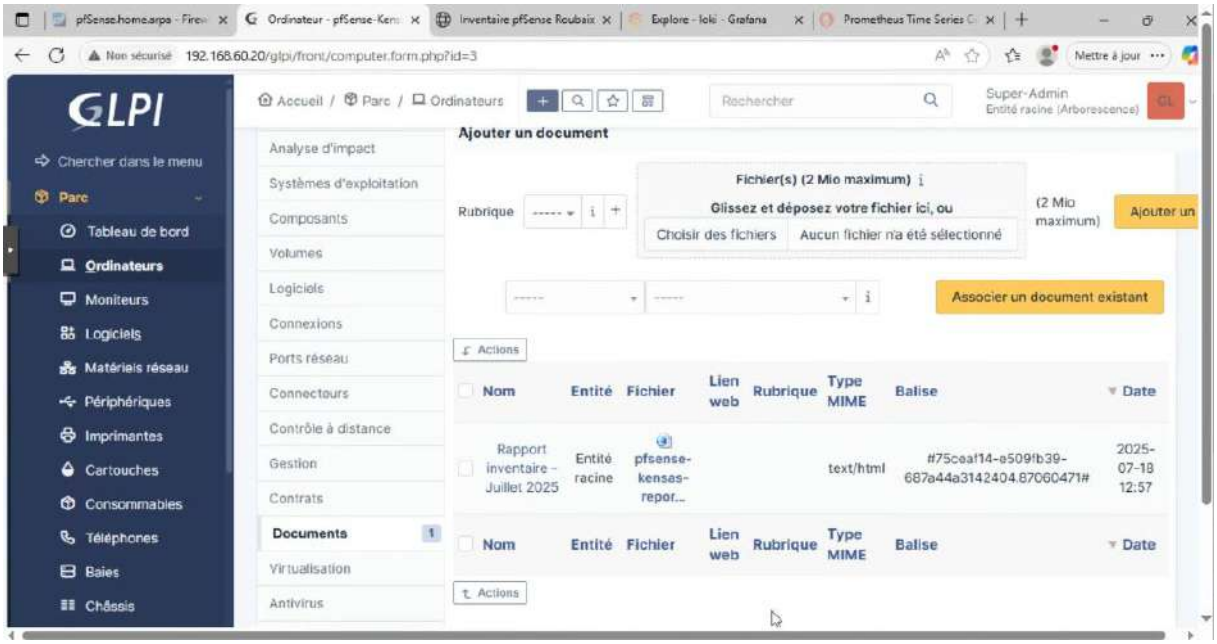

Figure 32 Fiche GLPI de pfSense du site Kensas avec rapport d'inventaire HTML intégré

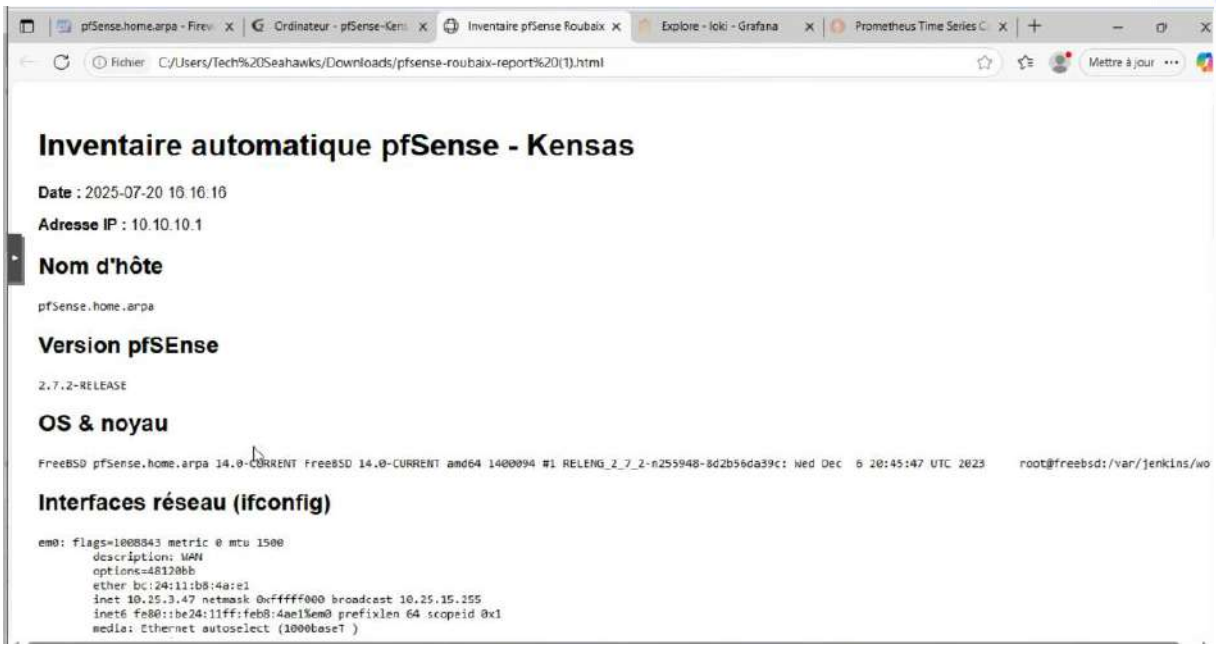

Figure 33 Extrait du rapport HTML généré par script depuis pfSense, joint à la fiche GLPI

5.6 – Mise en œuvre du cluster web

Dans le projet Seahawks Monitoring, un cluster web a été mis en place afin d'assurer une haute disponibilité, une répartition de charge, et une sécurité renforcée de l'application Seahawks Nester . Cette application permet de visualiser et centraliser les données remontées par les agents Harvester déployés dans les franchises distantes. Le cluster repose sur trois nœuds (Nester1, Nester2, Nester3), un équilibreur de charge HAProxy et une architecture VPN sécurisée via WireGuard.

5.6.1 Composants de l'architecture

- HAProxy : serveur d'équilibrage de charge (IP : 10.10.12.2) qui reçoit les requêtes entrantes et les répartit vers les serveurs Nester selon une stratégie round-robin.
- Nester1, Nester2, Nester3 : serveurs Debian 12 hébergeant chacun l'application Flask via Gunicorn, servie localement par NGINX.
- NGINX : reverse proxy local sur chaque Nester qui gère les connexions HTTP et fait le lien entre HAProxy et Gunicorn.
- Gunicorn : serveur exécutant l'application Flask sur le port local 127.0.0.1:8000.
- WireGuard : VPN client-to-site sécurisé utilisé par les agents Harvester pour transmettre leurs données au cluster de manière chiffrée.

5.6.2 Circuit d'une requête typique

Lorsqu'un utilisateur accède à l'interface Nester via http://10.10.12.2 :

1. La requête arrive sur le serveur HAProxy.
2. HAProxy la redirige vers l'un des serveurs Nester disponibles.
3. NGINX reçoit la requête HTTP et la transmet localement à Gunicorn.
4. Gunicorn exécute le code Flask et retourne une réponse HTML.
5. La réponse remonte vers l'utilisateur en suivant le même chemin inversé.

Ce fonctionnement permet une redondance continue et un accès centralisé.

5.6.3 Sécurité et cloisonnement applicatif

Le cluster web a été conçu selon une approche de défense en profondeur :

- NGINX agit comme une barrière entre le réseau et l'application Flask, en filtrant les requêtes HTTP, en interdisant certaines méthodes et en limitant les accès.
- Gunicorn n'est accessible que localement, empêchant tout accès direct non autorisé à l'application.
- HAProxy réalise des vérifications d'état (health checks) pour rediriger les requêtes uniquement vers les serveurs actifs.
- WireGuard chiffre les données en transit entre les Harvester et HAProxy.

Chaque niveau du cluster joue un rôle de filtre et de cloisonnement, renforçant la robustesse de l'ensemble.

5.6.4 Bénéfices techniques et organisationnels

- Tolérance aux pannes : si un serveur Nester est hors service, les autres prennent le relais automatiquement.
- Point d'entrée unique : toutes les connexions passent par 10.10.12.2 (HAProxy), simplifiant l'accès.
- Meilleure répartition de la charge : round-robin pour éviter les saturations.
- Sécurité renforcée à chaque niveau (NGINX, Gunicorn, VPN).
- Modularité : chaque composant peut être redémarré ou mis à jour indépendamment, sans interruption du service.

Annexes :

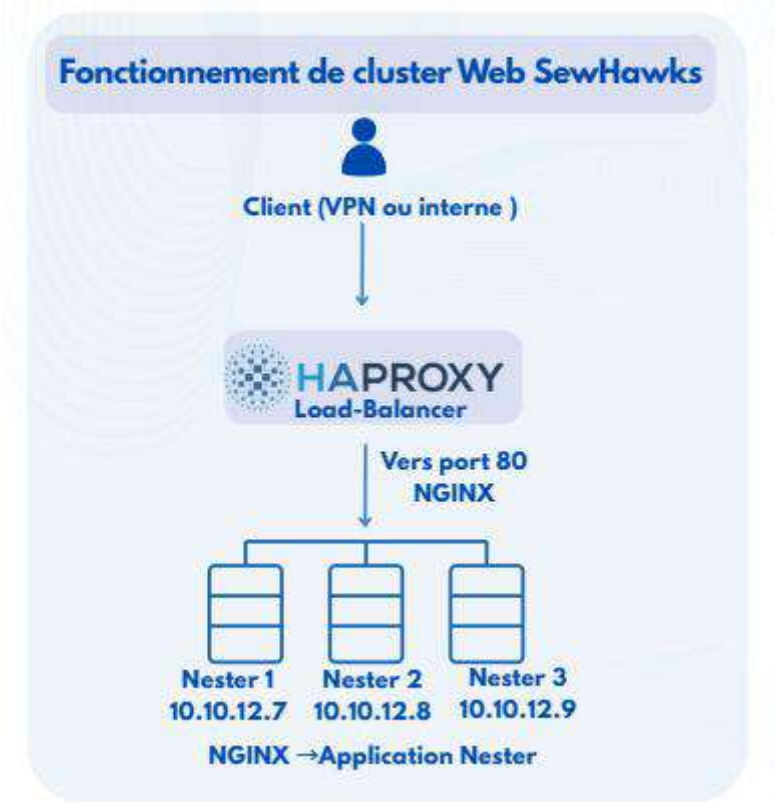

Figure 34 Fonctionnement du cluster web SeaHawks

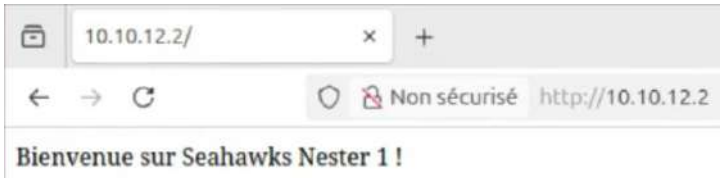

Figure 35 Affichage de l'application SeaHawks Nester via le reverse proxy NGINX à l'adresse http://10.10.12.2

Figure 36 Illustration du Load Blancing HAProxy : les requêtes sont distribuées automatiquement entre les 3 noeuds

```
defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

frontend http_front
        bind *:80
        default_backend nester_servers

backend nester_servers
        balance roundrobin
        option httpchk
        server nester1 10.10.12.7 8000 check
        server nester2 10.10.12.8 8000 check
        server nester3 10.10.12.9 8000 check
```

Figure 37 Configuration HAProxy du cluster web, avec statégie round-robin vers les serveurs Nester

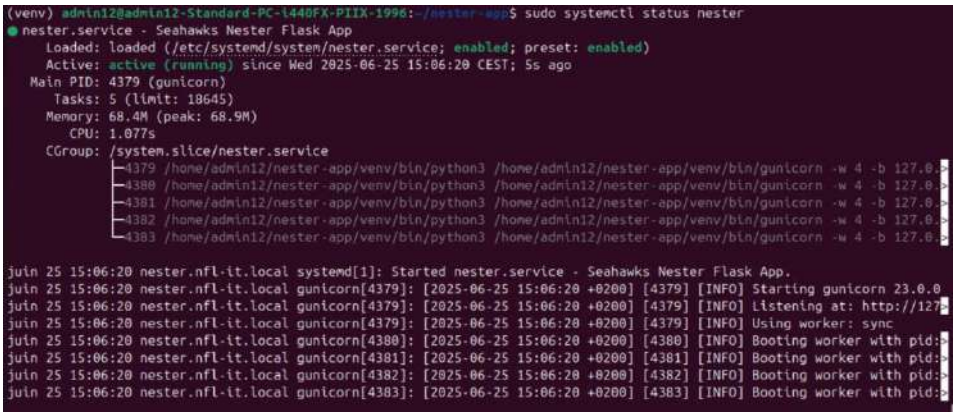

Figure 38 Le serveur Gunicorn "nester" est actif et permet l'éxécution continue de l'application Flask

5.7– Mise en place de la supervision

Dans le cadre du projet Seahawks Monitoring, une supervision complète des machines a été mise en place à l'aide de Prometheus, Grafana, Loki et Promtail. Cette architecture permet de surveiller en temps réel l'état des machines Linux et Windows, de recevoir des alertes en cas de surcharge, et de centraliser les journaux système pour faciliter le diagnostic.

5.7.1 Installation de Prometheus et Grafana

Prometheus a été installé sur une VM Arch Linux, avec Grafana dans le même environnement. Les services sont actifs en tant que daemons systemd. Prometheus collecte les métriques système via Node Exporter ou Windows Exporter. Grafana se connecte à Prometheus pour afficher les données en dashboards dynamiques.

Des tableaux de bord prédéfinis ont été utilisés (ex : Node Exporter Full, Windows Exporter) pour visualiser CPU, mémoire, disques, charge système, etc.

5.7.2 Supervision de machines distantes

Chaque machine supervisée intègre un agent d'export (Node Exporter ou Windows Exporter), exposant les données sur un port spécifique (9100 ou 9184). Prometheus les interroge à intervalle régulier (15 secondes). (Cas particulier pfSense)

Dans le cas des machines situées derrière le VPN site-à-site, des règles pfSense ont été ajoutées pour autoriser les flux sur les ports nécessaires via IPsec.

5.7.3 Mise en place d'alertes (exemple CPU)

Des règles d'alerte Prometheus ont été ajoutées pour détecter certaines anomalies, comme une surcharge CPU persistante. Les règles sont définies dans le fichier alert.rules.yml, chargé automatiquement par Prometheus.

Exemple : alerte si le CPU dépasse 80% pendant plus de 60 secondes. Cette configuration permet de passer d'un état INACTIVE → PENDING → FIRING, avec une résolution automatique lorsque la situation revient à la normale.

5.7.4 Centralisation des logs avec Loki & Promtail

Loki a été installé en complément sur la même VM que Prometheus et Grafana. Promtail est configuré sur les machines distantes (Linux) pour envoyer les logs système vers Loki, centralisés dans Grafana.

Des dashboards permettent une lecture en temps réel (mode Live) ou sur historique, filtrée par label (ex : job=varlogs).
Cette solution complète l'analyse métrique par une visibilité sur les événements système.

Injection de logs pour test Grafana Loki

Pour vérifier que l'agent Promtail fonctionnait correctement et que les logs étaient bien remontés dans Grafana via Loki, il a été nécessaire de générer manuellement une ligne de log dans le fichier surveillé par Promtail.

Commande utilisée :

```
echo "test alpine $(date)" >> /var/log/test.log
```

Remarque : cette commande est utile en phase de test, lorsque peu de logs sont générés. En production, elle n'est généralement pas nécessaire, car les services actifs produisent déjà des journaux exploités automatiquement par Promtail.

5.7.5 Supervision de pfSense via SNMP

Le pare-feu pfSense ne disposant pas de Node Exporter natif, la supervision a été mise en place via SNMP. Pour cela, un schéma intermédiaire a été déployé :

Activation du service SNMP sur pfSense (v2c, community 'public').
Installation de Telegraf sur la VM lb-nester. Telegraf est configuré pour interroger pfSense en SNMP (OIDs ifHCInOctets, ifHCOutOctets, ifHighSpeed, etc.), puis exposer ces métriques au format Prometheus via un endpoint HTTP (:9273/metrics).
Prometheus collecte ces données grâce à un job dédié (telegraf-pfsense).
Grafana affiche ensuite des dashboards construits à partir de ces métriques, en particulier :

- Débit entrant / sortant de l'interface WAN (enc0) :

```
rate(pfsense_snmp_ifHCInOctets{ifName="enc0"}[5m]) * 8
rate(pfsense_snmp_ifHCOutOctets{ifName="enc0"}[5m]) * 8
```

- Statut de l'interface WAN (ifOperStatus) :

```
max_over_time(pfsense_snmp_ifOperStatus{ifName="enc0"}[5m])
```

Cette intégration permet de superviser en temps réel le trafic réseau du pare-feu, tout en homogénéisant la collecte (Prometheus comme source unique de vérité).

5.7.6 Bénéfices pour le monitoring

- Visualisation centralisée des performances systèmes
- Alertes proactives sur les charges excessives
- Centralisation des logs système pour audit
- Supervision étendue aux machines distantes (VPN)
- Dashboards riches pour Linux et Windows
- Meilleure réactivité face aux incidents
- Promtail : prend les logs du serveur et les envoie.
- Loki : reçoit, stocke et indexe ces logs pour qu'ils soient exploitables dans Grafana.

Annexes :

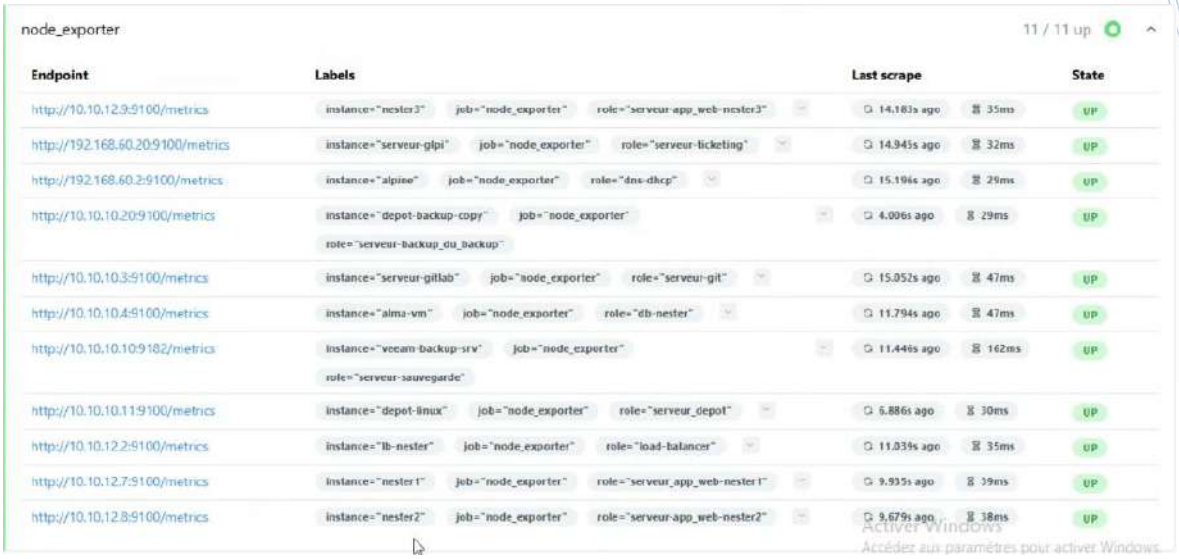

Figure 39 Liste des machines supervisées par Prometheus

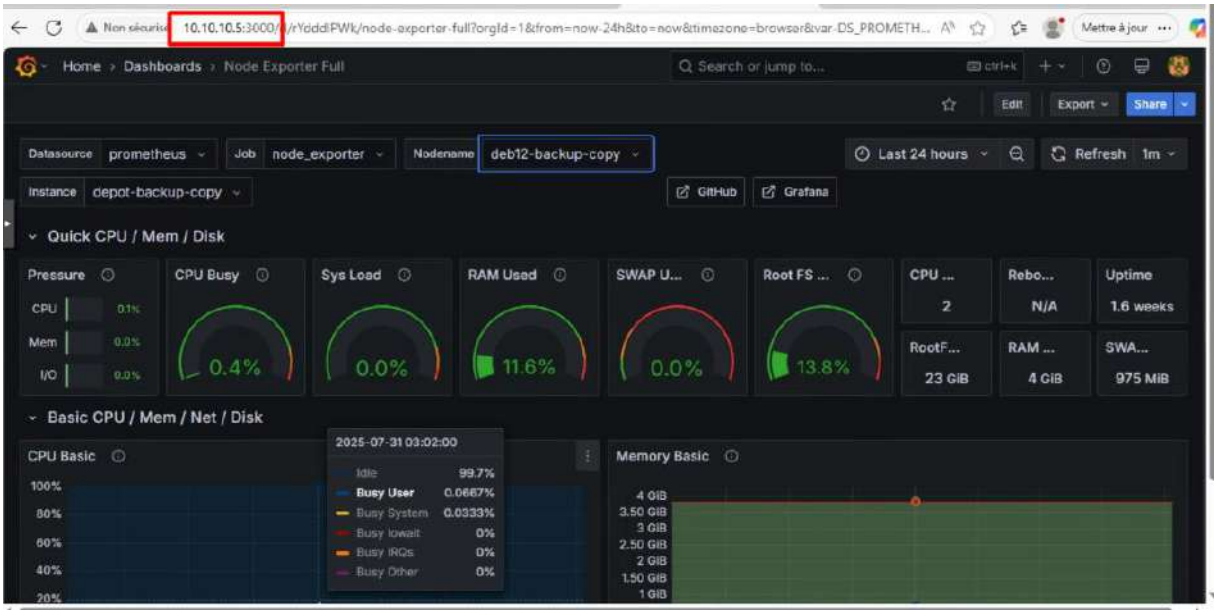

Figure 40 Vue des métriques systèmes collectées depuis les VM

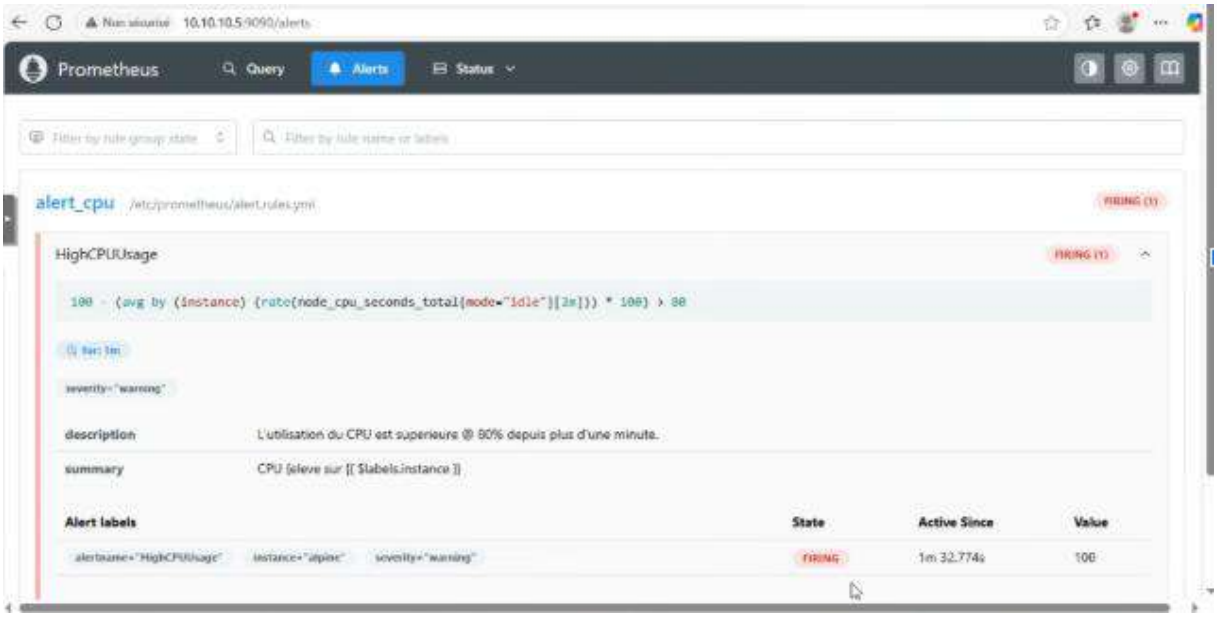

Figure 41 Alerte sur utilisation élevée délenchée via Prometheus

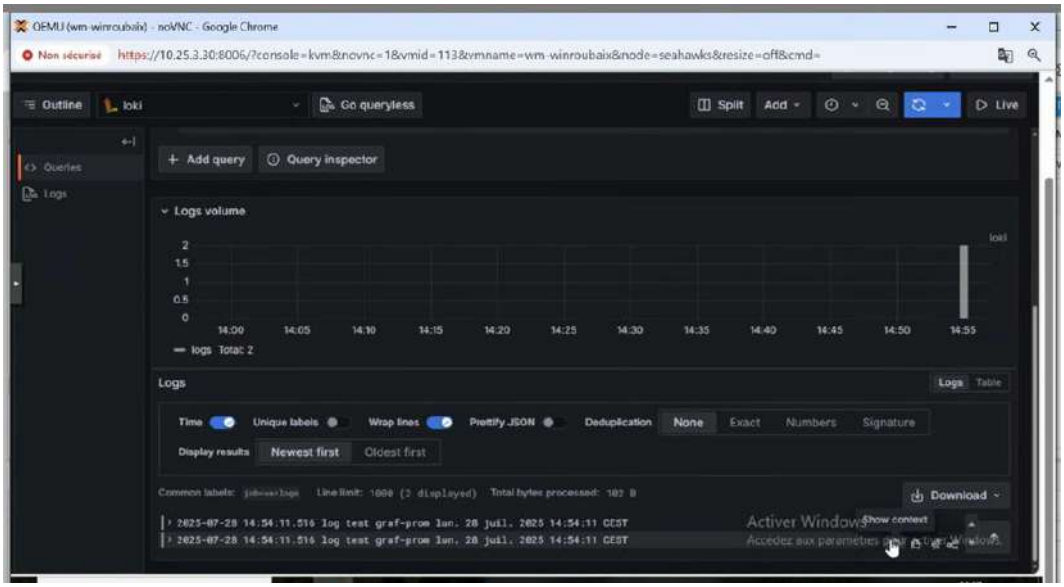

Figure 42 Consultation des logs systèmes grâce à Grafana+Loki

```
vm-dns-dhcp:/opt# cat /opt/promtail/promtail-config.yaml

server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://10.10.10.5:3100/loki/api/v1/push

scrape_configs:
  - job_name: alpine-logs
    static_configs:
      - targets:
          - localhost
        labels:
          job: alpine
          host: vm-alpine
          __path__: /var/log/*.log
vm-dns-dhcp:/opt#
```

Figure 43 Fichier de configuration de Promtail définissant la collecte des logs et l'envoi à l'instance Loki

Figure 44 Job pfSense, Vérification que Prometheus collecte bien les métriques de pfSense via Telegraf

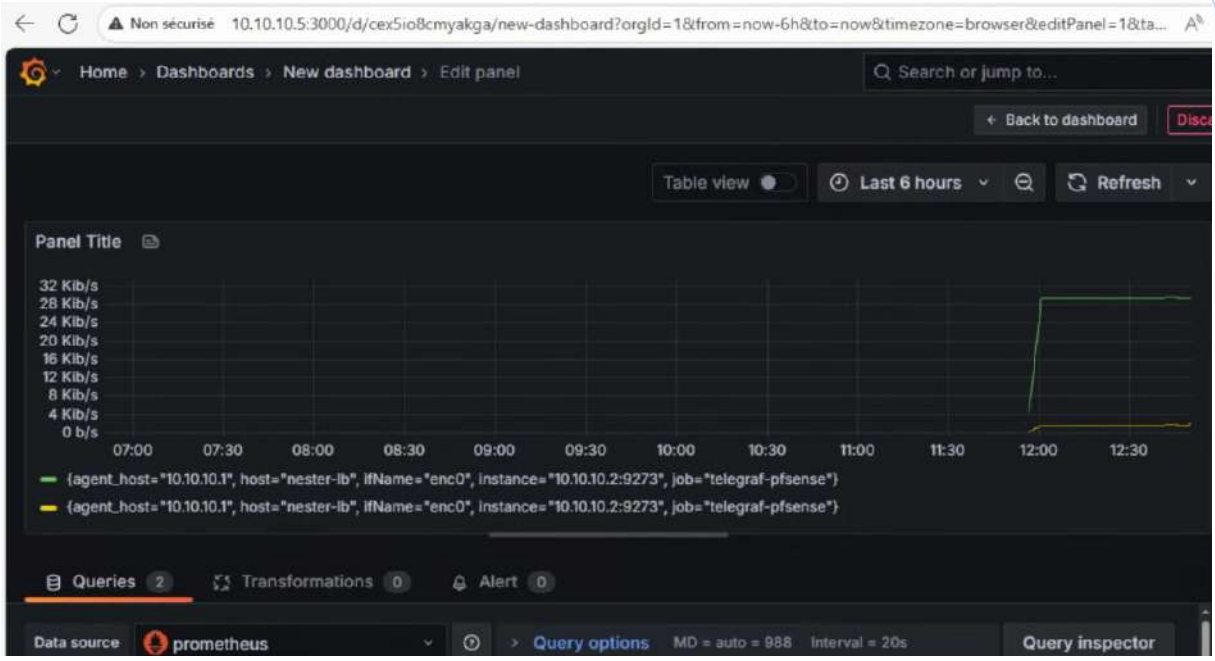

Figure 45 Courbes du trafic entrant et sortant de l'interface WAN de pfSense

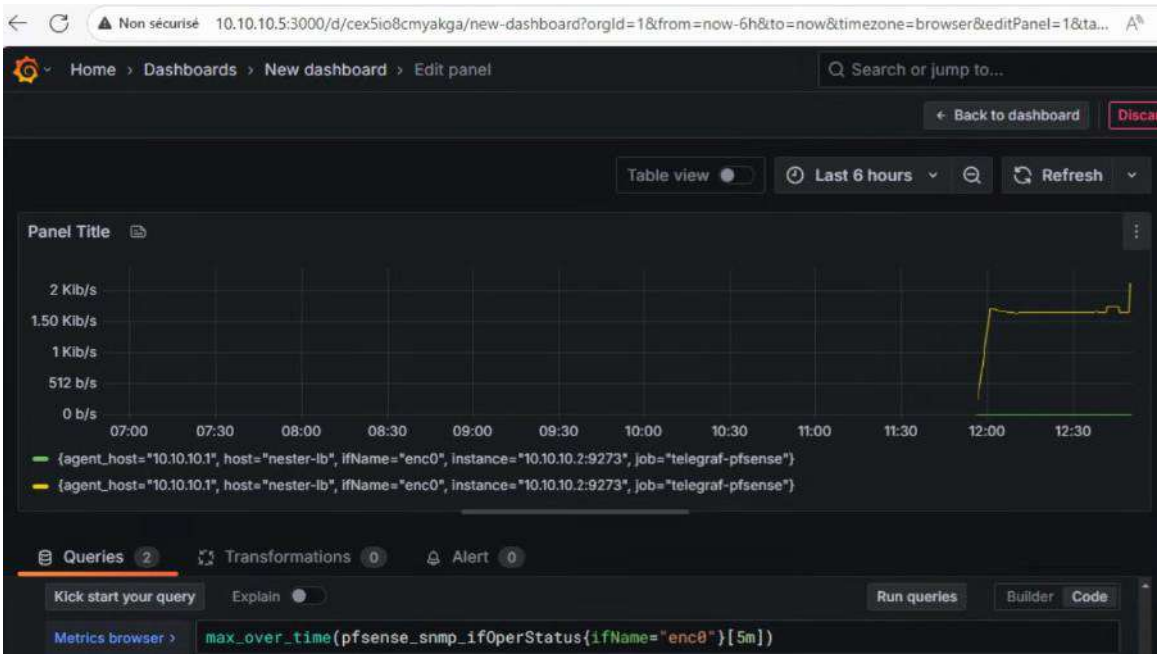

Figure 46 Suivi de l'état de l'interface WAN (active sur la période observée)

5.8 – Mise en œuvre de la base de données Nester Manager

Dans le cadre du projet Seahawks Monitoring, une base de données dédiée, nommée nester_manager, a été mise en place afin de stocker et organiser toutes les données nécessaires au fonctionnement de l'application Nester Manager. Cette application repose sur MariaDB (SGBD) et permet la gestion centralisée de l'ensemble des Harvesters déployés dans l'infrastructure, notamment le suivi des instances, la remontée des incidents, la gestion des prestataires et l'historique des opérations.

Objectif

Fournir un stockage fiable, structuré et accessible des données collectées et utilisées par Nester Manager, tout en assurant leur sauvegarde régulière pour éviter toute perte en cas d'incident.

5.8.1 Mise en place et structure

La conception de la base s'est articulée en plusieurs étapes :
- Élaboration d'un schéma conceptuel (MCD) décrivant les entités (Prestataire, ClientFinal, Instance, Technicien, Évènement, Licence, Script, etc.) et leurs relations.
- Traduction en schéma physique (MPD) avec définition des tables, clés primaires et étrangères, types de données et contraintes.
- Déploiement de la structure dans MariaDB via un script SQL créant l'ensemble des tables et relations prévues.
- Création d'utilisateurs SQL avec des droits adaptés, afin de sécuriser les accès et limiter les risques d'erreur ou d'intrusion.

5.8.2 Sauvegarde et résilience

Pour protéger les données, un script de sauvegarde MariaDB a été prévu. Il exporte la base nester_manager toutes les deux heures dans un répertoire dédié, avec une conservation des sauvegardes pendant sept jours. Ce mécanisme assure la possibilité de restaurer rapidement la base en cas de corruption, de suppression accidentelle ou de panne serveur.

5.8.3 Supervision

La machine hébergeant la base est intégrée à la supervision Grafana/Prometheus. Les indicateurs CPU, mémoire, disque et disponibilité sont suivis en temps réel, garantissant une réactivité en cas d'anomalie.

5.8.4 Intérêt pour le projet

La mise en place de cette base permet :
- La centralisation et l'organisation des données nécessaires à la gestion des Harvesters.
- Une intégration fluide avec l'application Nester pour la visualisation et l'exploitation.

- La sécurisation des informations grâce à une politique de sauvegarde régulière.
- Un suivi continu de l'état de santé du serveur hébergeant la base.

Annexes :

```
MariaDB [(none)]> SELECT user, host FROM mysql.user;
+-------------+-----------+
| User        | Host      |
+-------------+-----------+
| mariadb.sys | localhost |
| mysql       | localhost |
| root        | localhost |
+-------------+-----------+
3 rows in set (0,013 sec)

MariaDB [(none)]>
```

Figure 49 Liste des utilisateurs et hôtes autorisées à se connecter à MariaDB

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| nester_manager     |
| performance_schema |
| seahawks_db        |
+--------------------+
5 rows in set (0,001 sec)
```

Figure 50 Affichage des bases disponbles dans MariaDB, dont nester_manager et seahawks_db

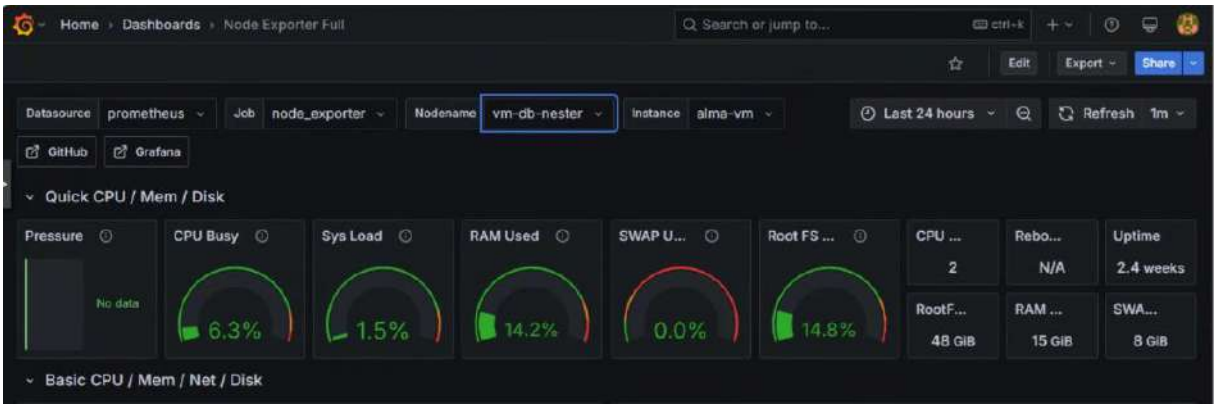

Figure 48 Supervision de la VM db-nester avec les indicateurs CPU, memoire, swap et disque

Figure 47 Interface phpMyadmin : aperçu de la gestion des utilisateurs et droits d'accès à la base MariaDB (Nester Manager)

5.9 –Mise en œuvre de l'application Harvester

Dans le cadre du projet Seahawks Monitoring mené pour la société fictive NFL IT, l'application Harvester représente la composante cliente installée dans chacune des 32 franchises NFL.

Objectif principal :
Fournir à l'équipe support un moyen automatisé et normalisé de collecter des informations techniques et de diagnostic sur le réseau local de chaque franchise, afin de réduire les déplacements et d'accélérer la résolution d'incidents.

5.9.1 Fonctionnement de l'application Harvester :

Chaque Harvester déployé en franchise établit un tunnel chiffré vers Roubaix à l'aide de WireGuard.

L'application Harvester fonctionne de manière autonome dans le LAN de la franchise. Elle agit comme une sonde logicielle, capable de :
- Détecter les équipements actifs grâce à un scan réseau réalisé avec la bibliothèque nmap (identification des adresses IP et des ports ouverts).
- Mesurer la latence WAN par envoi de requêtes ping.
- Collecter des informations système comme l'adresse IP locale, le nom d'hôte et la version de l'application.
- Générer un raport JSON contenant les données collectées.

Les données collectées peuvent être transmises vers Nester, l'application web hébergée au datacenter de Roubaix, qui centralise la visualisation de l'ensemble des franchises. Cette intégration permet au support de disposer d'une vue globale sur l'état des réseaux et d'intervenir de manière ciblée.

5.9.2 Intérêt de cette architecture :

- Une vue centralisée des performances et de l'état réseau de chaque franchise via Nester.
- En cas d'échec d'exportation du rapport vers le cluster web, il reste stocké localement .
- Une intégration future avec Nester Manager, l'outil administratif destiné à gérer l'inventaire, le cycle de vie et les licences de toutes les instances Harvester.

Le Harvester est conçu pour être :
- Facilement déployable sous forme de machine virtuelle prête à l'emploi.
- Simple d'utilisation grâce à une interface graphique qui masque la complexité technique.
- Évolutif grâce à l'utilisation de GitLab pour gérer les versions, avec la possibilité d'automatiser les mises à jour dans le futur.

En pratique, les techniciens support peuvent :
- Lancer un scan réseau complet afin d'identifier les hôtes et services actifs.
- Mesurer la latence et diagnostiquer les problèmes de connectivité.
- Consulter à distance les données collectées via Nester.

- Exploiter les informations pour anticiper et résoudre rapidement les incidents.

5.9.3 Stockage des rapports Harvesters dans la BDD

Les Harvesters collectent des informations dans les franchises et les transmettent automatiquement au serveur Nester.
Celui-ci assure ensuite leur enregistrement dans la base de données MariaDB Nester Manager, hébergée dans le VLAN 10 – Services**.**
Cette organisation garantit une centralisation des rapports et facilite leur consultation par les administrateurs via l'interface Nester.
Par mesure de sécurité, un utilisateur spécifique *nester_app* a été créé ; il s'agit du seul compte autorisé à se connecter à la base *seahawks_db*, qui centralise l'ensemble des rapports transmis par les Harvesters

5.9.4 Bénéfices pour le projet :

- Réduction significative des interventions sur site.
- Amélioration de la réactivité et de la qualité du support.
- Structuration des échanges d'informations techniques entre le terrain et le datacenter.

Annexes :

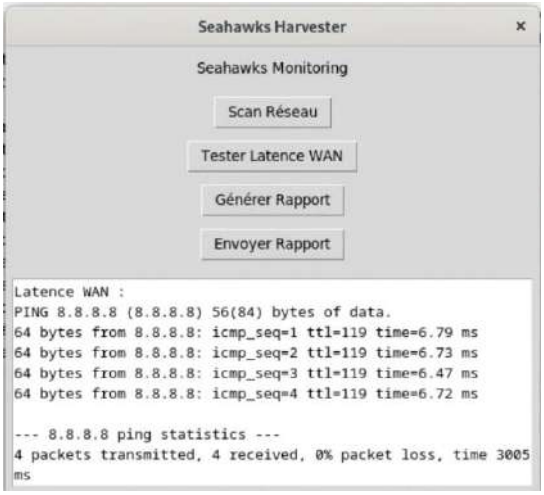

Figure 51 Fenêtre Tkinter affichant le résultat d'un test de latence vers l'adresse 8.8.8.8

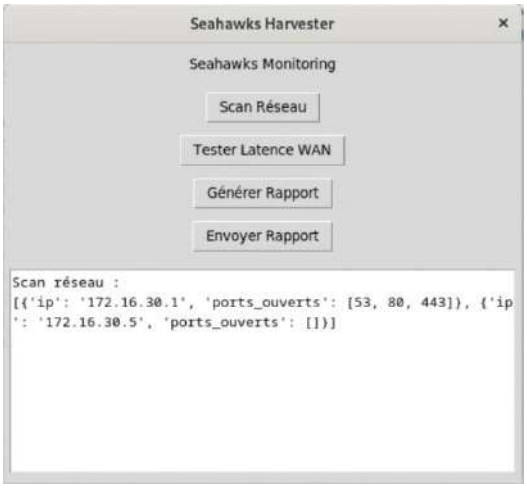

Figure 52 Fenêtre tkinter affichant les hôtes détectés et les ports ouverts après un scan LAN

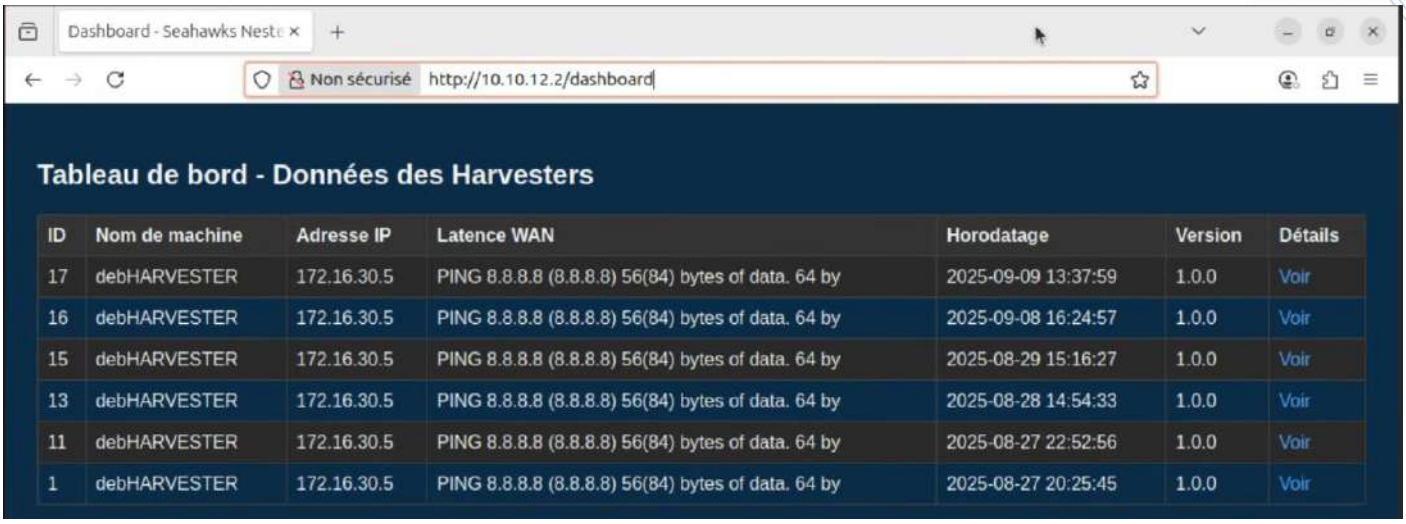

Figure 54 Exemlpe de rapport structuré contenant la latence WAN, le nom de machine et les résultats du scan réseau

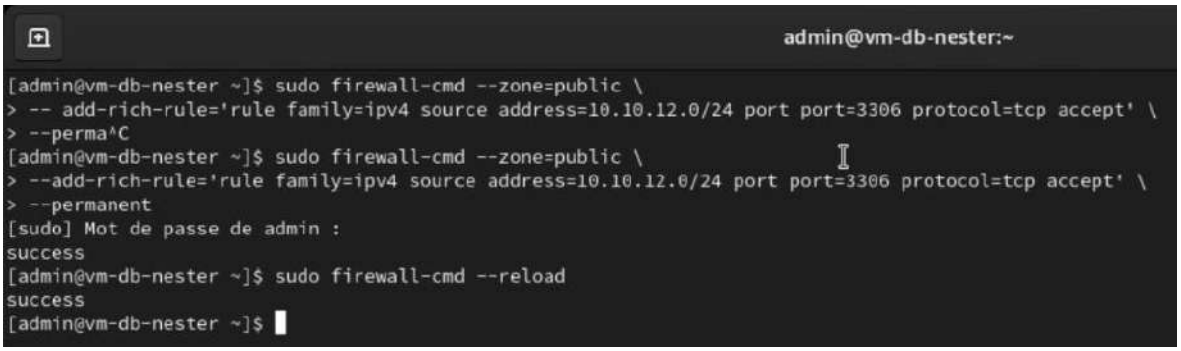

Figure 53 Régle pfSense autorisant le cluster web Nester à accéder à la BDD MariaDB située dans le VLAN 10

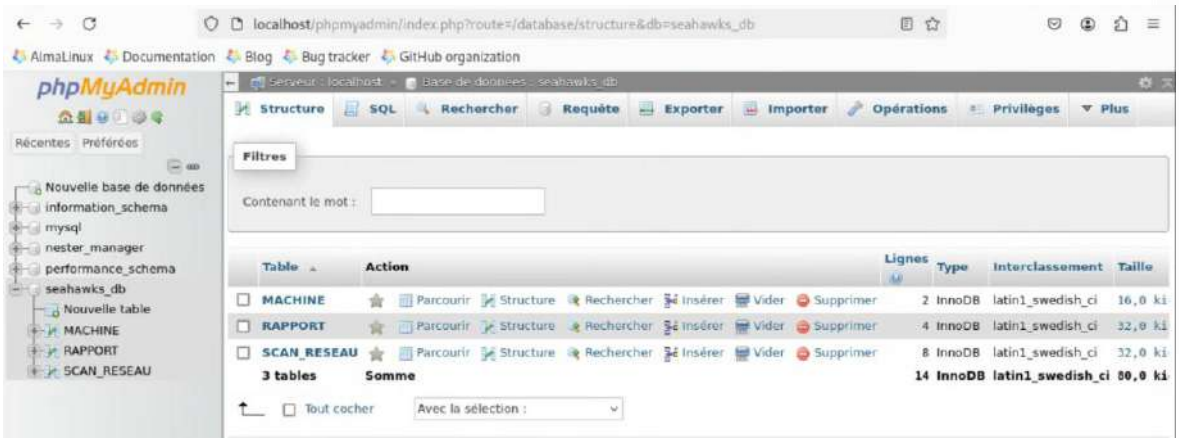

Figure 55 Interface phpMyAdmin : aperçu des BDD nester_manager et seahawks_db, cette dernière contenant les tables de stockage des rapports générés par les Harvesters

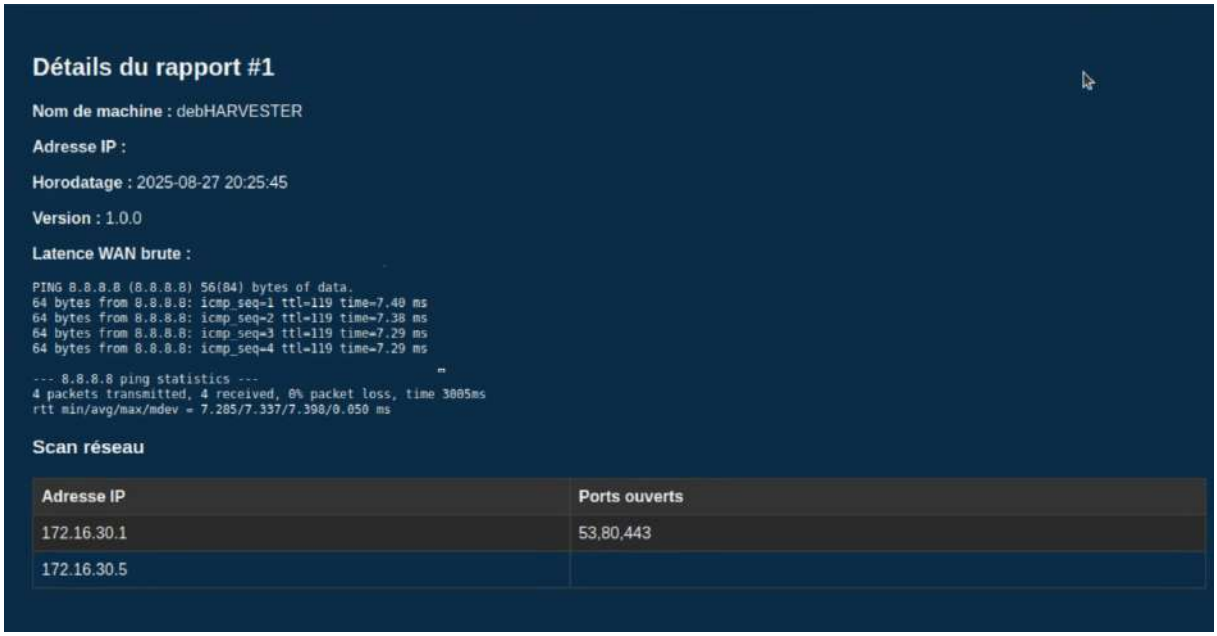

Figure 56 Exemple de rapport structuré contenant la latence WAN, le nom de machine et les résultats du scan réseau

5.10 – Mise en œuvre de la plateforme GitLab

Dans le cadre du projet Seahawks Monitoring, GitLab a été déployé comme plateforme centrale de gestion du code source et de collaboration pour les applications Harvester et Nester. L'outil permet de centraliser les dépôts, d'assurer la traçabilité des modifications et de gérer les contributions des différents membres du projet.

5.10.1 Fonctionnement de la plateforme GitLab

GitLab est utilisé pour héberger le code source, assurer la gestion des versions et faciliter la collaboration. Les dépôts des applications Harvester et Nester sont regroupés dans un espace unique, ce qui permet un suivi clair de l'historique et une gestion structurée des utilisateurs, des groupes et des droits d'accès. La plateforme intègre également la gestion des issues et la documentation liée au développement.

5.10.2 Intérêt de cette architecture

L'utilisation de GitLab apporte plusieurs avantages :
- Centralisation des projets et du code source.
- Traçabilité des évolutions et des auteurs.
- Gestion sécurisée des accès et des rôles.
- Amélioration de la collaboration entre les membres de l'équipe.
- Pérennité de la solution : l'architecture est conçue pour accueillir facilement de nouvelles franchises sans remettre en cause l'organisation technique existante.

5.10.3 En pratique, GitLab a permis de :

- Héberger et organiser les dépôts du Harvester et du Nester.
- Gérer les utilisateurs et leurs permissions.
- Centraliser la documentation liée au développement.
- Suivre les activités et les modifications de manière partagée et sécurisée.

5.10.4 Bénéfices pour le projet

GitLab a contribué à structurer le développement collaboratif et à renforcer la visibilité des évolutions. Cette organisation garantit une meilleure maintenabilité et une évolutivité à long terme, en particulier dans la perspective d'intégrer progressivement de nouvelles franchises au projet Seahawks Monitoring sans complexifier l'infrastructure existante.

Annexes :

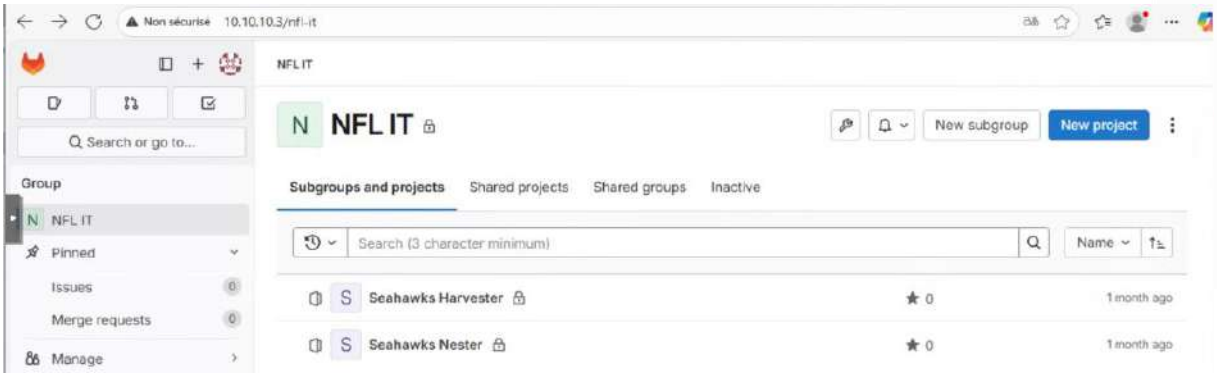

Figure 57 Interface Gitlab - Vue des dépôts de code SeaHawks Harvester et SeaHawks Nester centralisés dans Gitlab

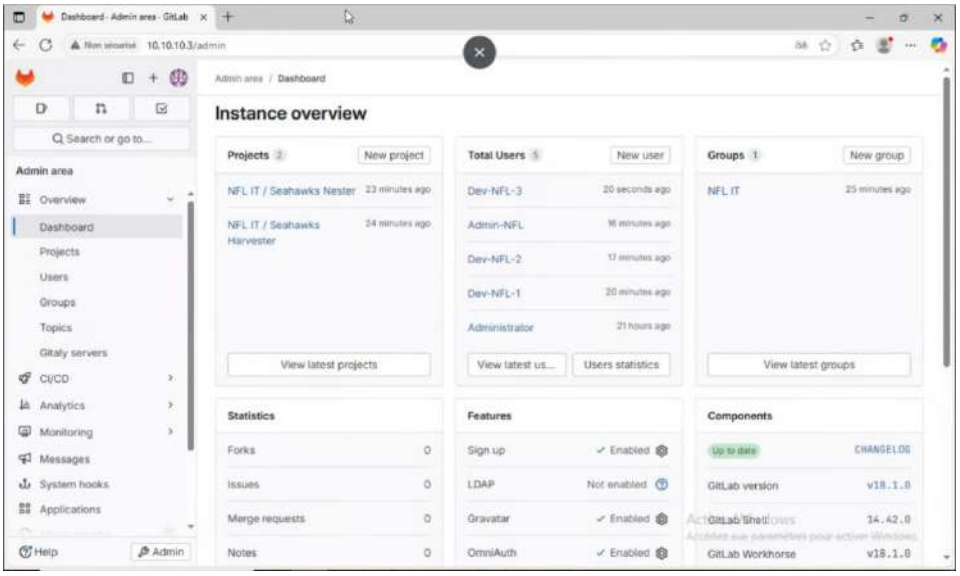

Figure 58 Tableau de bord GitLab : gestion des projets SeaHawks Harvester et Nester

5.11 – Télémaintenance des Harvesters

La mise en place de la télémaintenance avait pour objectif de permettre aux équipes du siège Seahawks d'administrer et de superviser à distance les Harvesters déployés dans les différentes franchises. Pour cela, une solution sécurisée a été conçue en s'appuyant sur une machine virtuelle Windows dédiée, jouant le rôle de bastion intermédiaire. Cette organisation garantit la centralisation des accès tout en assurant une traçabilité et une isolation conformes aux bonnes pratiques de cybersécurité.

5.11.1 Déploiement de l'environnement de télémaintenance

L'infrastructure repose sur une VM Windows Server installée à Roubaix, derrière pfSense et connectée au VPN site-à-site avec le siège.

- Cette VM a été configurée comme poste rebond, en activant le protocole RDP (Remote Desktop Protocol).
- Un compte utilisateur dédié, remoteops, a été créé pour les opérations de télémaintenance.
- L'accès RDP est restreint uniquement aux adresses IP issues du VPN du siège (filtrage Windows Firewall et pfSense).

Une fois connectés à cette VM, les administrateurs peuvent accéder aux Harvesters Debian via le VPN client-à-site en WireGuard.

5.11.2 Gestion des accès et des connexions distantes

Depuis la VM bastion, différents outils d'administration (mRemoteNG, PuTTY, VNC, RDP) permettent de se connecter aux Harvesters.

Pour les environnements Linux (Debian 12), un service XRDP a été installé afin de rendre les machines accessibles via RDP. XRDP implémente le protocole RDP sur Linux et s'appuie sur XFCE, un environnement de bureau léger, utilisé comme interface graphique minimale.

Cette combinaison permet aux techniciens de retrouver une interface familière et fluide depuis Windows, tout en maintenant une faible consommation de ressources sur les serveurs.

5.11.3 Cas particuliers et justifications

La mise en place d'un bastion intermédiaire peut sembler ajouter une étape supplémentaire. Cependant, elle se justifie pleinement dans le contexte Seahawks :

- La présence de 32 franchises rendait impossible la gestion de tunnels directs entre chaque Harvester et le siège.
- Centraliser les connexions via une seule VM simplifie l'administration, limite la complexité réseau et réduit la surface d'attaque.
- Le bastion permet une meilleure traçabilité (logs Windows centralisés) et un contrôle des accès.

5.11.4 Résilience et optimisation

- Les accès sont strictement limités aux adresses IP du VPN du siège, renforçant la sécurité.
- Les mots de passe utilisés respectent les règles de complexité et sont renouvelés régulièrement.
- La VM bastion est durcie : seuls les outils nécessaires à la télémaintenance sont installés.
- XRDP + XFCE assure une interface distante fonctionnelle, même avec une faible bande passante.

Bénéfices pour le projet

La solution de télémaintenance mise en œuvre permet d'assurer un accès centralisé, sécurisé et maîtrisé aux Harvesters distants. En utilisant une VM Windows bastion associée à XRDP/XFCE pour les serveurs Linux, l'infrastructure répond aux besoins opérationnels de supervision et d'administration, tout en respectant les contraintes de sécurité et de simplicité de gestion. Cette approche constitue un compromis équilibré entre accessibilité et cybersécurité dans un contexte multi-sites.

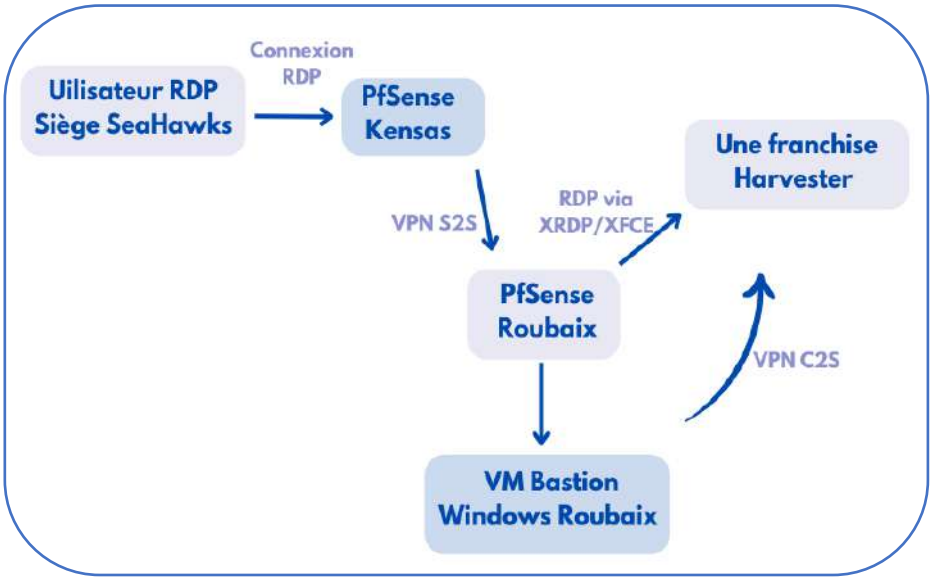

Figure 59 Chaîne de télémaintenance : accès RDP du siège vers le Bastion Roubaix via VPN S2S, puis connexion aux Harvesters en RDP via VPN C2S

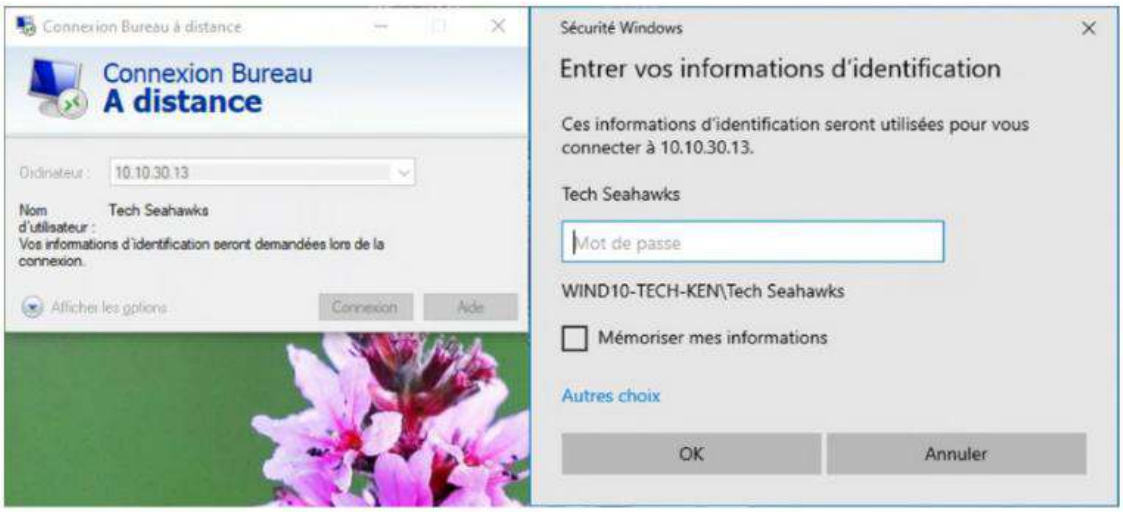

Figure 60 Accès distant sécurisé via le bastion (conexion RDP avec authentification Tech SeaHawks)

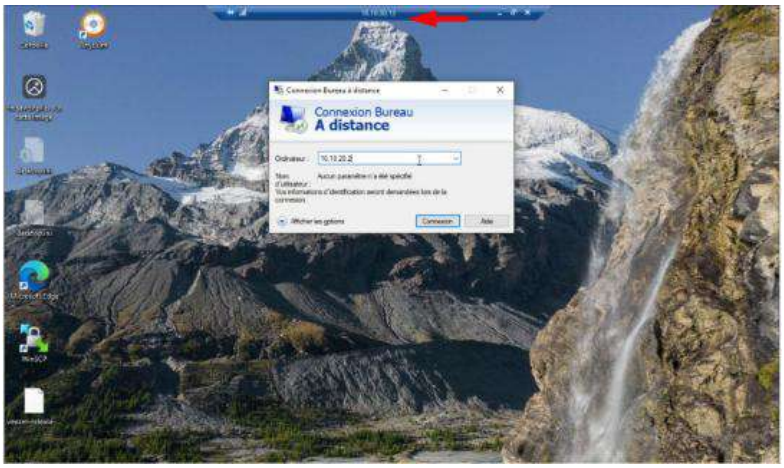

Figure 61 Accès distant au bastion Windows, depuis le siège, via protocole RDP (10.10.20.2)

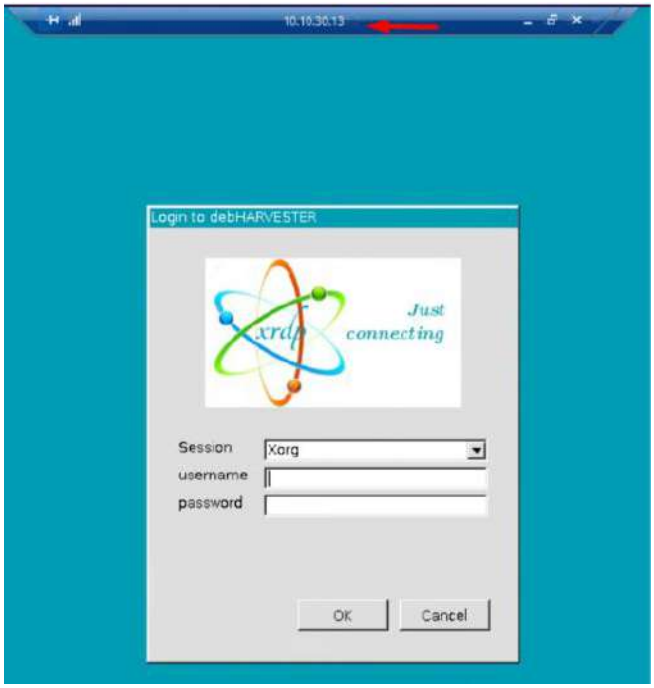

Figure 62 Connexion graphique à distance sur un Harvester Dabian via XRDP

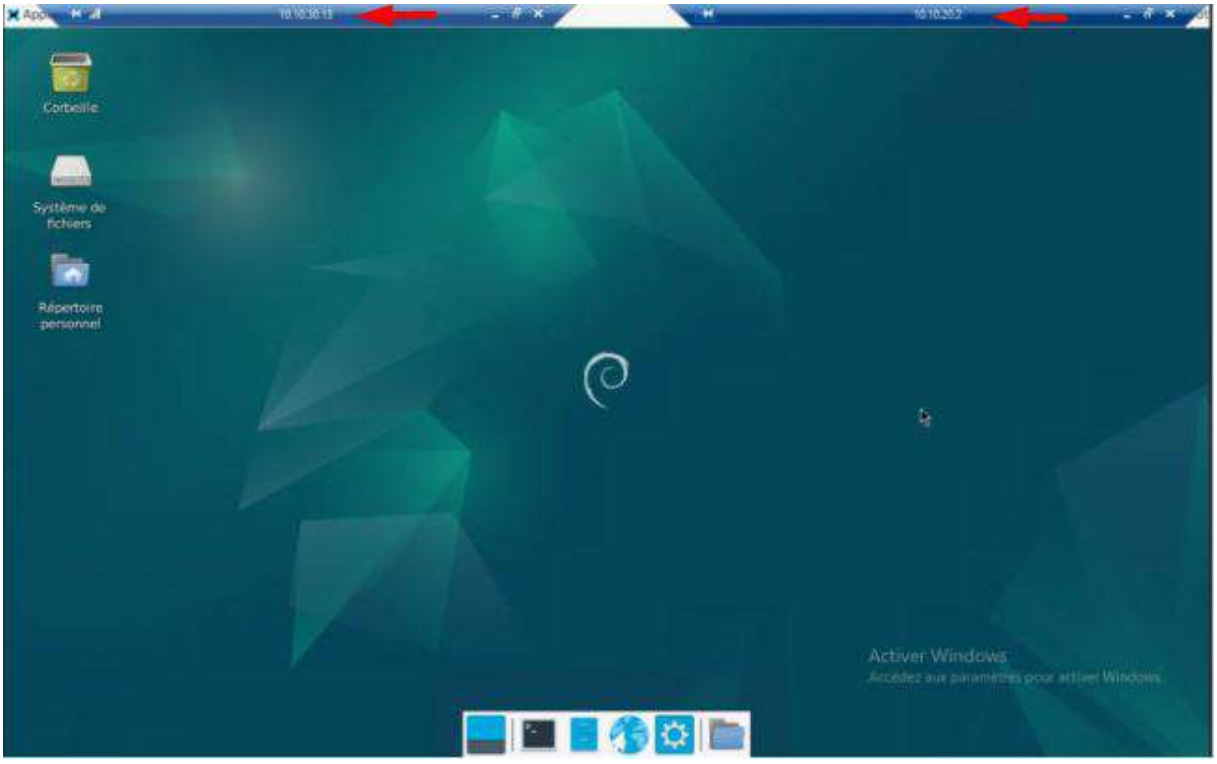

Figure 63 Session distante active entre le bastion (10.10.10.13) et un Harvester (10.10.20.2)

Conclusion finale

Le projet Seahawks Monitoring nous a permis de concevoir et de déployer une infrastructure complète, sécurisée et évolutive, répondant aux besoins d'une organisation multisite. Grâce à l'intégration de solutions éprouvées telles que Proxmox, pfSense, Veeam, GLPI, Grafana/Prometheus et GitLab, nous avons assuré la supervision, la sauvegarde, la maintenance et la gestion centralisée des services essentiels.

Ce projet a été l'occasion de mettre en pratique l'ensemble des compétences attendues d'un Administrateur d'Infrastructures Sécurisées : conception et déploiement d'architectures virtualisées, segmentation réseau et sécurité renforcée, mise en place de mécanismes de supervision et de sauvegarde, et gestion d'outils collaboratifs pour le support et l'inventaire.

En définitive, Seahawks Monitoring illustre notre capacité à proposer une solution technique fiable, sécurisée et durable, mais aussi à travailler en équipe sur un projet structuré de bout en bout. Il constitue une démonstration concrète des compétences acquises au cours de notre formation et une étape significative dans notre parcours vers le titre professionnel d'Administrateur d'Infrastructures Sécurisées.

Bilan et limites du projet

La mise en œuvre du projet Seahawks Monitoring a permis de déployer une infrastructure virtualisée complète et fonctionnelle, intégrant à la fois les aspects systèmes, réseaux et sécurité.

Réalisations marquantes :

Mise en place d'un environnement virtualisé basé sur Proxmox et ZFS, garantissant robustesse et souplesse.
Déploiement de services réseaux essentiels (DNS, DHCP) et d'outils métiers clés (GLPI, Veeam, Prometheus, Grafana, GitLab).
Segmentation du réseau avec VLANs et DMZ, associée à une politique de filtrage via pfSense.
Déploiement d'un mécanisme de maintenance à distance permettant l'accès sécurisé aux infrastructures des franchises via VPN et bastion.
Intégration de l'application Harvester/Nester, qui bien que ne relevant pas directement des compétences AIS, constitue une brique essentielle pour la supervision et l'exploitation des environnements distants.

Difficultés rencontrées :

Contraintes matérielles liées au travail sur une seule machine physique, limitant certaines possibilités (par exemple, la mise en place d'une haute disponibilité réelle).

Limites identifiées :

Absence de cluster pfSense en haute disponibilité.

Pas de plan de reprise d'activité complet défini.

Projet réalisé en environnement de formation, sans exposition en conditions réelles de production.

Perspectives d'évolution

Plusieurs axes d'évolution permettraient d'enrichir la solution et de la rapprocher d'un usage en production :

Haute disponibilité : mise en place d'un cluster PfSense HA pour renforcer la continuité de service.
Automatisation : intégration d'outils comme Ansible ou Terraform pour standardiser et accélérer le déploiement des configurations.
Sécurité renforcée : ajout d'une brique de type SIEM (Wazuh, ELK) pour centraliser et corréler les événements de sécurité.
Plan de reprise d'activité (PRA) : formalisation de procédures complètes de redémarrage après sinistre.
Extension aux franchises : l'architecture a été pensée pour permettre l'intégration de nouvelles franchises NFL sans remise en cause de l'organisation technique existante, garantissant évolutivité et facilité de gestion.
Sécurisation des accès web : passage de l'application Nester en HTTPS avec certificats TLS (Let's Encrypt ou auto-signés), accompagné d'une redirection HTTP → HTTPS afin de garantir la confidentialité des échanges et de se conformer aux bonnes pratiques de cybersécurité.

Remerciement :

Je tiens tout d'abord à remercier chaleureusement nos **formateurs** pour leur accompagnement, leurs conseils et leur disponibilité tout au long de cette formation. Leur expertise et leur engagement ont été déterminants pour la réussite de cette formation et pour ma progression vers le métier d'Administrateur d'Infrastructures Sécurisées.

Je souhaite également exprimer ma gratitude à mes **camarades de formation**, avec qui j'ai partagé cette année d'apprentissage. Leurs échanges, leur soutien et leur esprit collectif ont rendu ce parcours plus enrichissant et motivant.

Enfin, un grand merci à ma **famille**, pour ses encouragements constants et son soutien moral durant toute cette période. Leur présence et leur confiance m'ont permis d'avancer avec motivation et détermination.