Compte rendu – Déploiement d'un logiciel de supervision réseau

LEDIEU Raphaël

Introduction

Ce compte rendu est consacré au déploiement d'une solution de supervision réseau pour l'entreprise Selenia Software. Dans le cadre de ma mission chez Egnom, prestataire de services informatiques, j'ai été mandaté pour concevoir et implémenter une infrastructure de monitoring répondant aux besoins spécifiques de cette PME spécialisée dans le développement de sites e-commerce.

L'objectif principal de ce projet était de déployer une solution de supervision de type Zabbix ou Nagios capable de surveiller en temps réel les équipements critiques du parc informatique de Selenia Software, comprenant une station Linux, une station Windows et un switch réseau. Le système devra notamment détecter et alerter automatiquement en cas de problèmes tels que la saturation d'un disque dur, la modification de l'heure système, ou l'apparition de boucles réseau sur le switch.

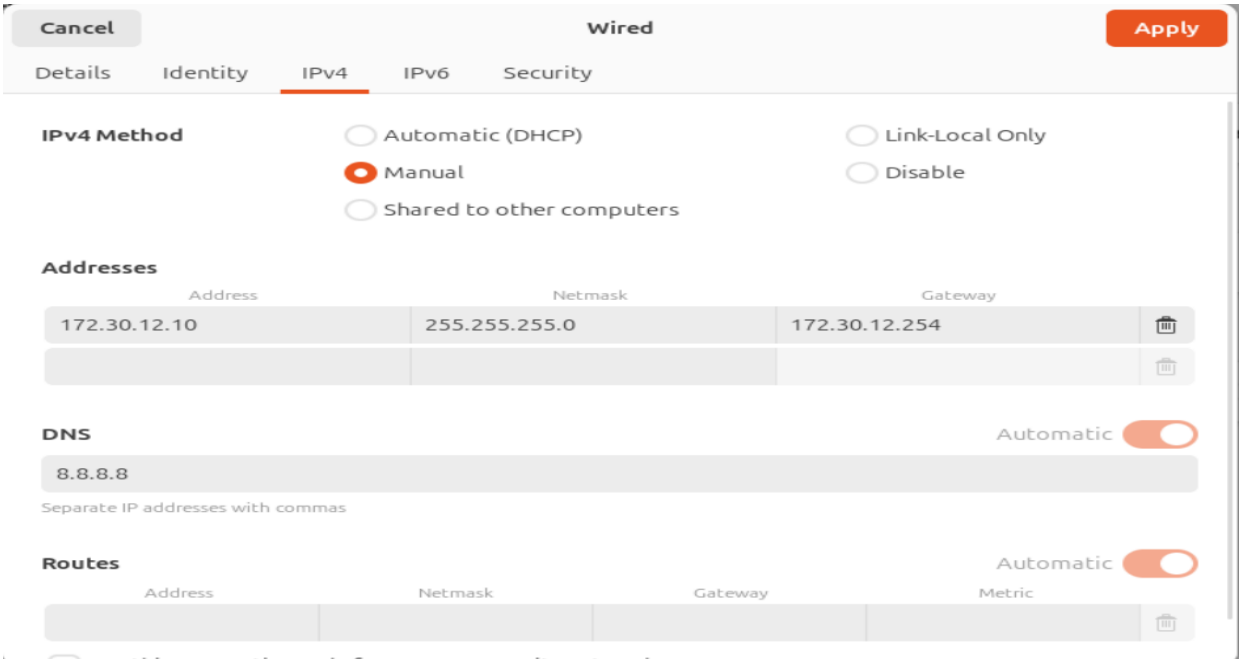
Cette mission s'inscrit dans une démarche complète de gestion de projet, depuis l'analyse des besoins du client jusqu'au déploiement opérationnel de la solution retenue, en passant par l'étude technique et le choix raisonné des technologies. La solution mise en œuvre devra garantir la qualité de service attendue tout en respectant les impératifs de sécurité et de conformité réglementaire propres au traitement des données de supervision.

Ainsi, après évaluation comparative des différentes solutions disponibles, j'ai opté pour Zabbix qui offre une approche centralisée et scalable, particulièrement adaptée à l'environnement technique de Selenia Software et à ses besoins en matière de monitoring proactif.

Configuration Initiale du Serveur de Supervision

Nous avons débuté par la création d'une machine virtuelle Ubuntu dédiée au serveur de supervision. La capture d'écran montre la configuration manuelle de son interface réseau.

L'adresse IP **« 172.30.12.10 »** a été attribuée statiquement pour garantir la permanence et la fiabilité de la connectivité, ce qui est essentiel pour un serveur d'infrastructure. Cette adresse place le serveur dans le même réseau que les autres équipements de Selenia Software, lui permettant de les superviser efficacement.



Installation du dépôt Zabbix

J'ai utilisé la commande complète suivante pour télécharger le paquet de dépôt Zabbix :

« wget https://repo.zabbix.com/zabbix/7.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest_7.0+ubuntu22.04_all.deb »

Cette commande permet de récupérer le paquet d'installation officiel qui contient les dépôts et clés GPG nécessaires pour installer Zabbix 7.0 sur Ubuntu de manière sécurisée et maintenable.

J'ai ensuite installé le paquet de dépôt Zabbix avec la commande :

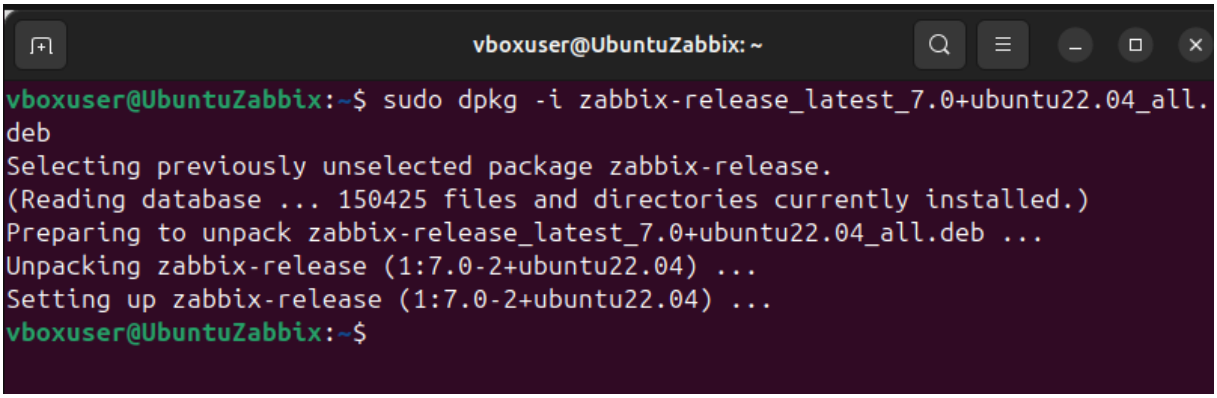
« sudo dpkg -i zabbix-release_latest_7.0+ubuntu22.04_all.deb »

sudo : exécute la commande avec les privilèges administrateur

dpkg -i : installe le paquet Debian (.deb) téléchargé

Le paquet zabbix-release configure automatiquement les dépôts officiels Zabbix dans le système

Cette installation réussie signifie que notre système Ubuntu peut maintenant accéder aux paquets officiels de Zabbix 7.0 via la commande **« apt update »,** préparant ainsi l'installation des composants Zabbix proprement dits.



J'ai installé les composants principaux de Zabbix avec la commande :

« sudo apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-sql-scripts zabbix-agent -y »

Explication des paquets installés :

zabbix-server-mysql : Le serveur Zabbix avec support MySQL/MariaDB

zabbix-frontend-php : L'interface web en PHP

zabbix-apache-conf : La configuration Apache pour l'interface web

zabbix-sql-scripts : Les scripts SQL pour créer la base de données

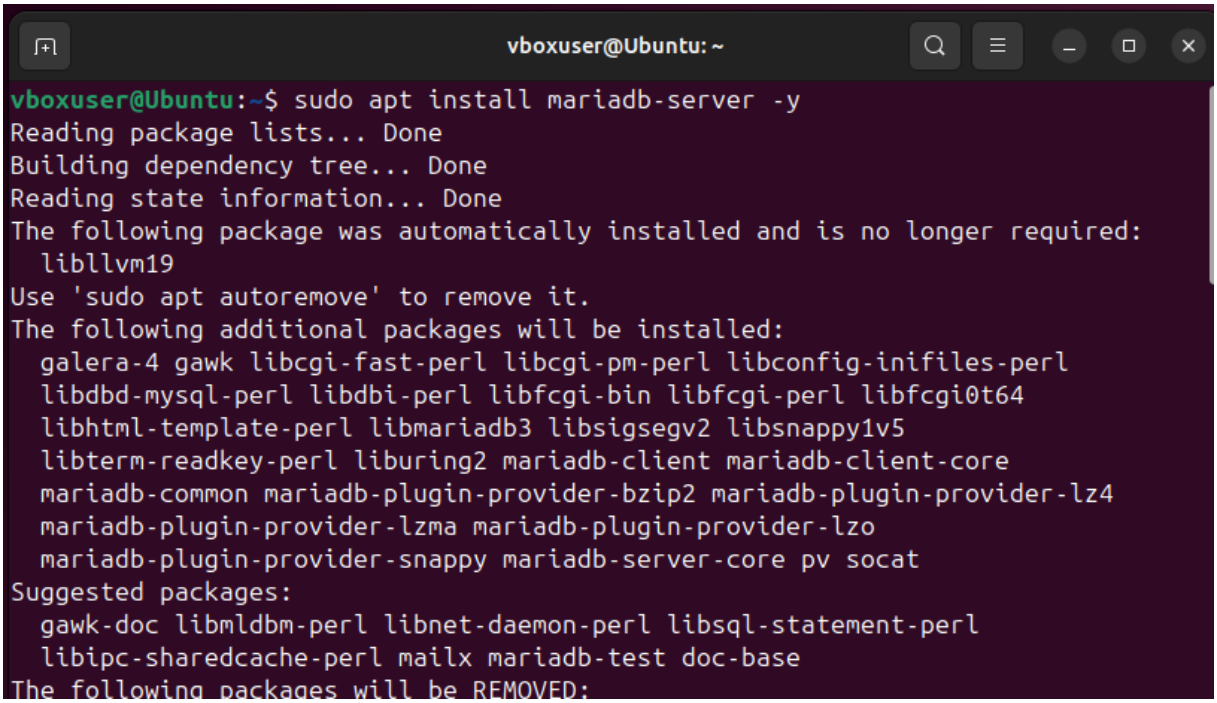
zabbix-agent : L'agent de supervision pour le serveur lui-même

La commande télécharge d'abord la clé GPG officielle de Zabbix pour authentifier les paquets, puis procède à l'installation complète de l'écosystème Zabbix. L'option -y confirme automatiquement l'installation sans demande de validation.

```
sudo apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-sql-scripts zabbix-agent -y
--2025-11-26 08:40:54--  https://repo.zabbix.com/zabbix-official-repo.key
Resolving repo.zabbix.com (repo.zabbix.com)... 178.128.6.101, 2604:a880:2:d0::2062:d001
Connecting to repo.zabbix.com (repo.zabbix.com)|178.128.6.101|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9033 (8.8K) [application/octet-stream]
Saving to: 'STDOUT'

-                   100%[===================>]   8.82K  --.-KB/s    in 0s

2025-11-26 08:40:55 (549 MB/s) - written to stdout [9033/9033]

Hit:1 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
```

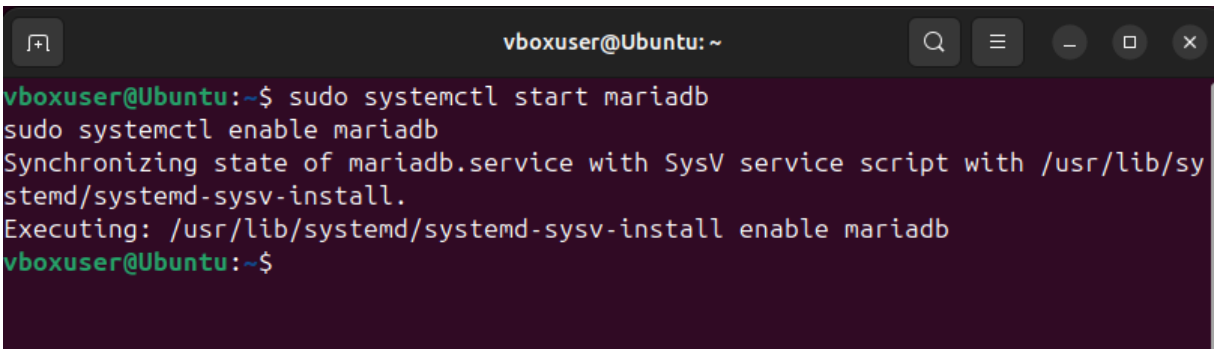
J'ai ensuite procédé à l'installation du serveur de base de données **MariaDB** en exécutant la commande

« sudo apt install mariadb-server -y », ce qui a permis de valider automatiquement l'ajout des dépendances nécessaires.



J'ai ensuite démarré le service de base de données avec la commande

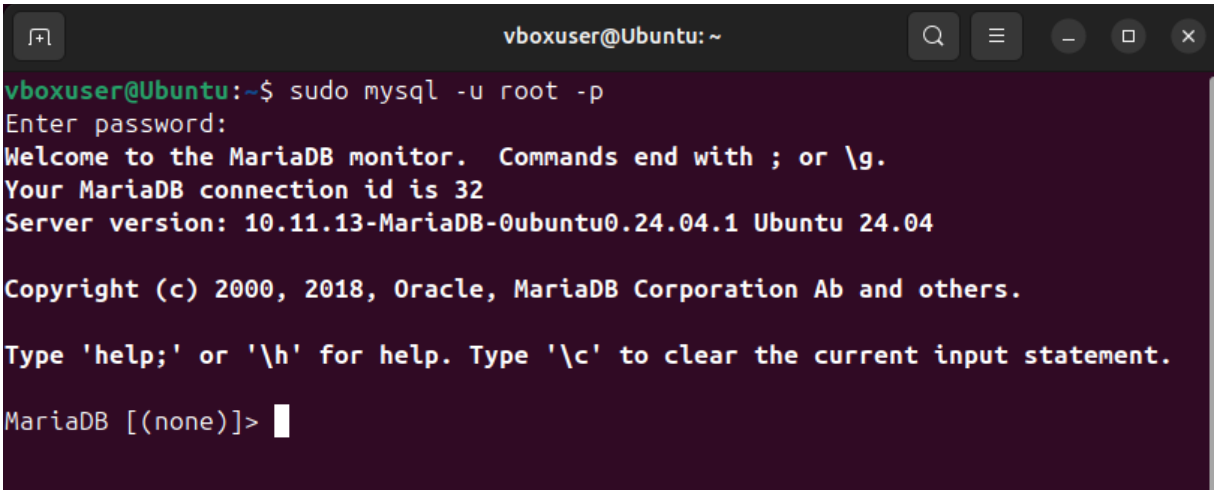
« sudo systemctl start mariadb » puis j'ai configuré son lancement automatique au démarrage du système via la commande **« sudo systemctl enable mariadb »**.



Ensuite j’ai accédé à l'interface d'administration de la base de données en me connectant avec le compte "root" via la commande

« sudo mysql -u root -p »

Ce qui ouvre l'invite de commande MariaDB.



Une fois connecté à l'interface MariaDB, j'ai créé la base de données avec l'encodage approprié en saisissant :

« CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_bin; »

J'ai ensuite créé l'utilisateur dédié avec son mot de passe via la commande :

« CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'MotDePasseZabbix123!'; »

J'ai accordé à cet utilisateur tous les droits sur la base via :

« GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost'; »

Enfin, pour autoriser l'importation future des scripts, j'ai exécuté

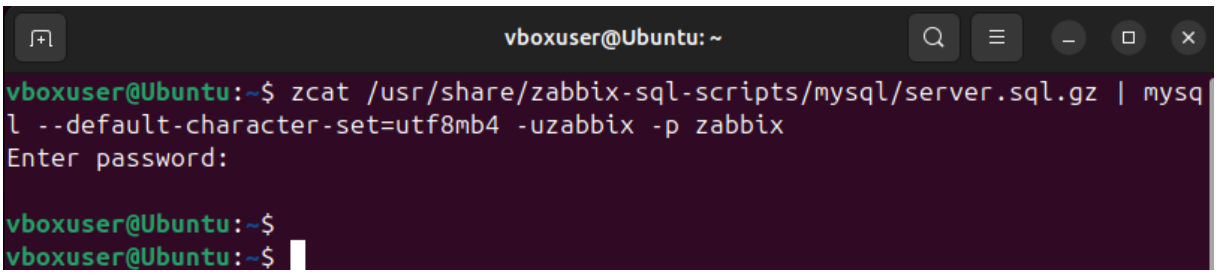
« SET GLOBAL log_bin_trust_function_creators = 1; »

Puis j'ai validé les changements avec **« FLUSH PRIVILEGES; »** avant de quitter l'interface avec **« EXIT; ».**

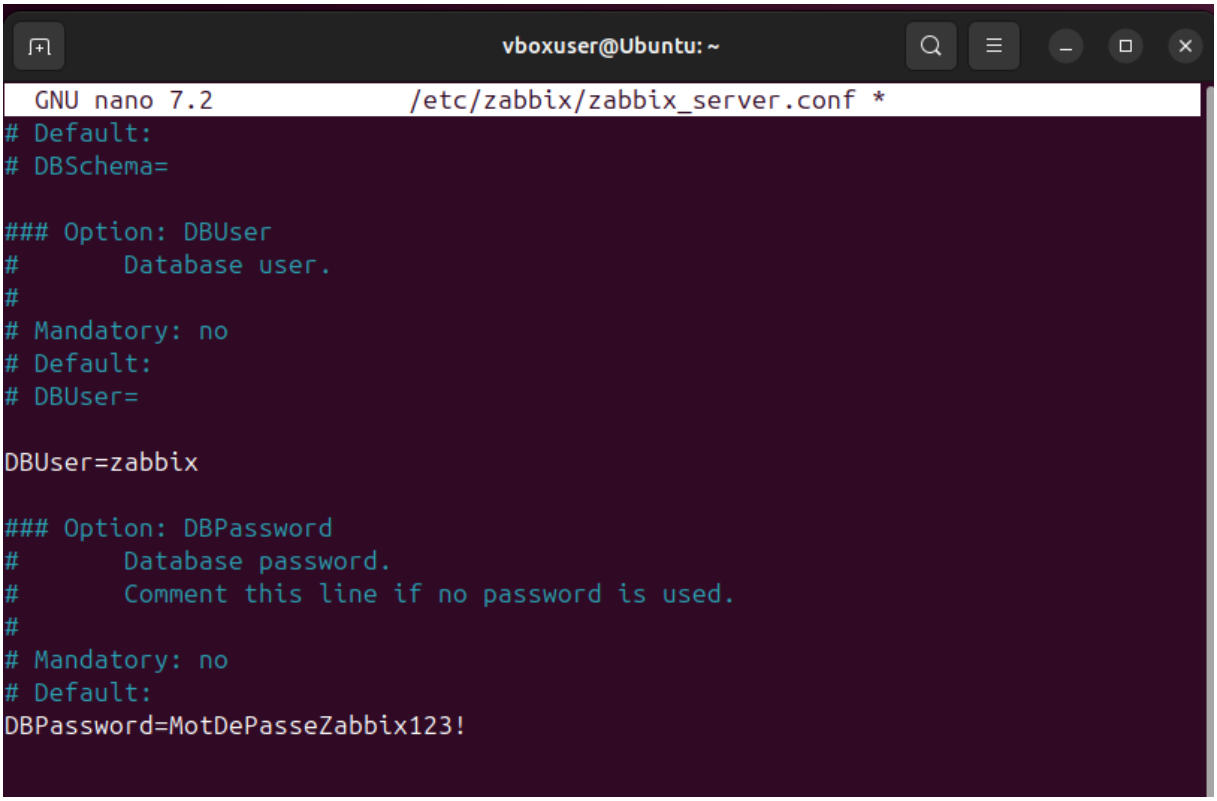
```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_b
in;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'MotDePasseZabb
ix123!';
Query OK, 0 rows affected (0.031 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost';
Query OK, 0 rows affected (0.027 sec)

MariaDB [(none)]> SET GLOBAL log_bin_trust_function_creators = 1;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> EXIT;
Bye
vboxuser@Ubuntu:~$
```

J'ai ensuite importé le schéma initial et les données nécessaires au fonctionnement de Zabbix dans la base de données. Pour cela, j'ai exécuté la commande suivante qui extrait et injecte directement le script SQL fourni :

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



J'ai ensuite édité le fichier de configuration principal du serveur Zabbix **« /etc/zabbix/zabbix_server.conf ».** J'ai recherché la directive **« DBPassword »** pour y renseigner le mot de passe défini précédemment (MotDePasseZabbix123!), permettant ainsi au logiciel Zabbix de s'authentifier et de communiquer avec la base de données.

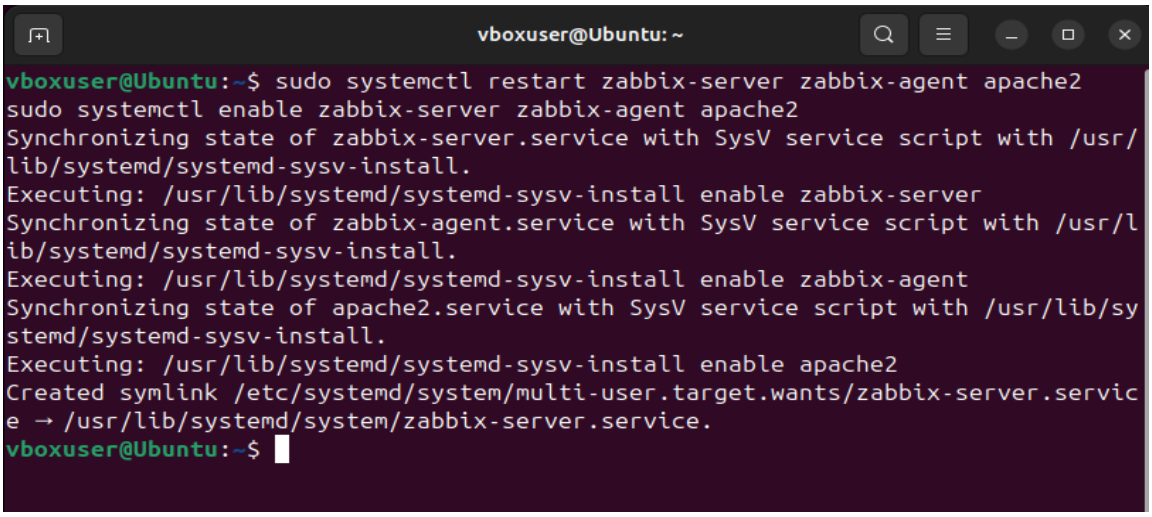


Pour finaliser l'installation et appliquer les configurations précédentes, j'ai redémarré l'ensemble des services nécessaires (le serveur Zabbix, l'agent de surveillance et le serveur web Apache) via la commande :

« sudo systemctl restart zabbix-server zabbix-agent apache2 ».

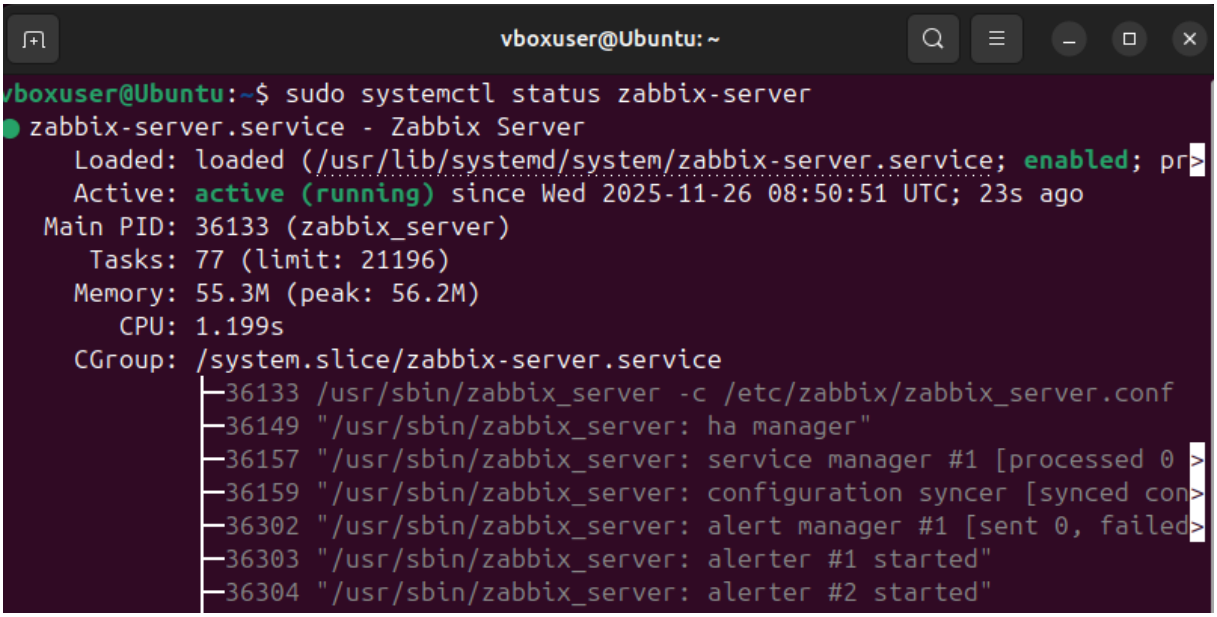
Dans la foulée, j'ai configuré ces services pour qu'ils se lancent automatiquement au démarrage de la machine grâce à la commande :

« sudo systemctl enable zabbix-server zabbix-agent apache2. »



Pour valider l'ensemble de l'installation, j'ai vérifié l'état du service principal via la commande :

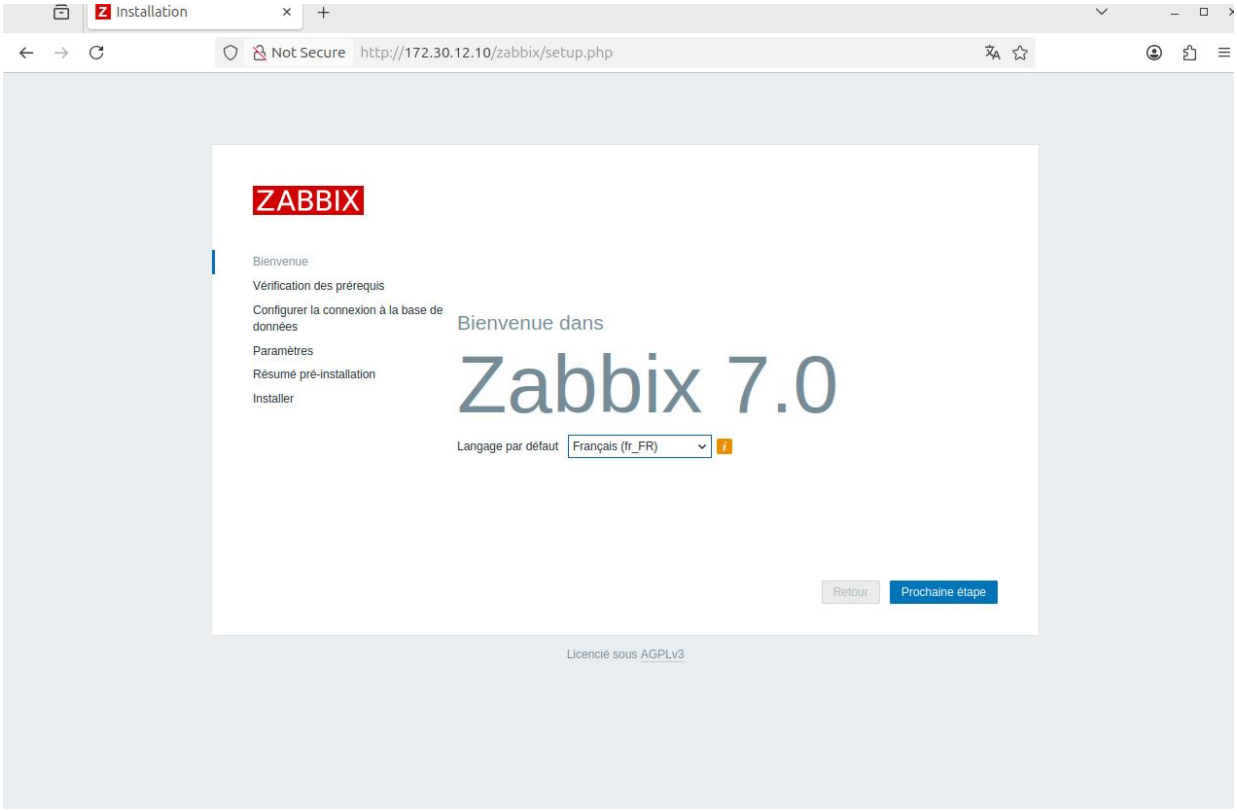
« sudo systemctl status zabbix-server » Le retour indiquant un statut **"active (running)"** en vert confirme que le serveur Zabbix a démarré correctement et qu'il est désormais opérationnel.



J'ai accédé à l'interface web via **« http://172.30.12.10/zabbix »** pour finaliser la configuration.

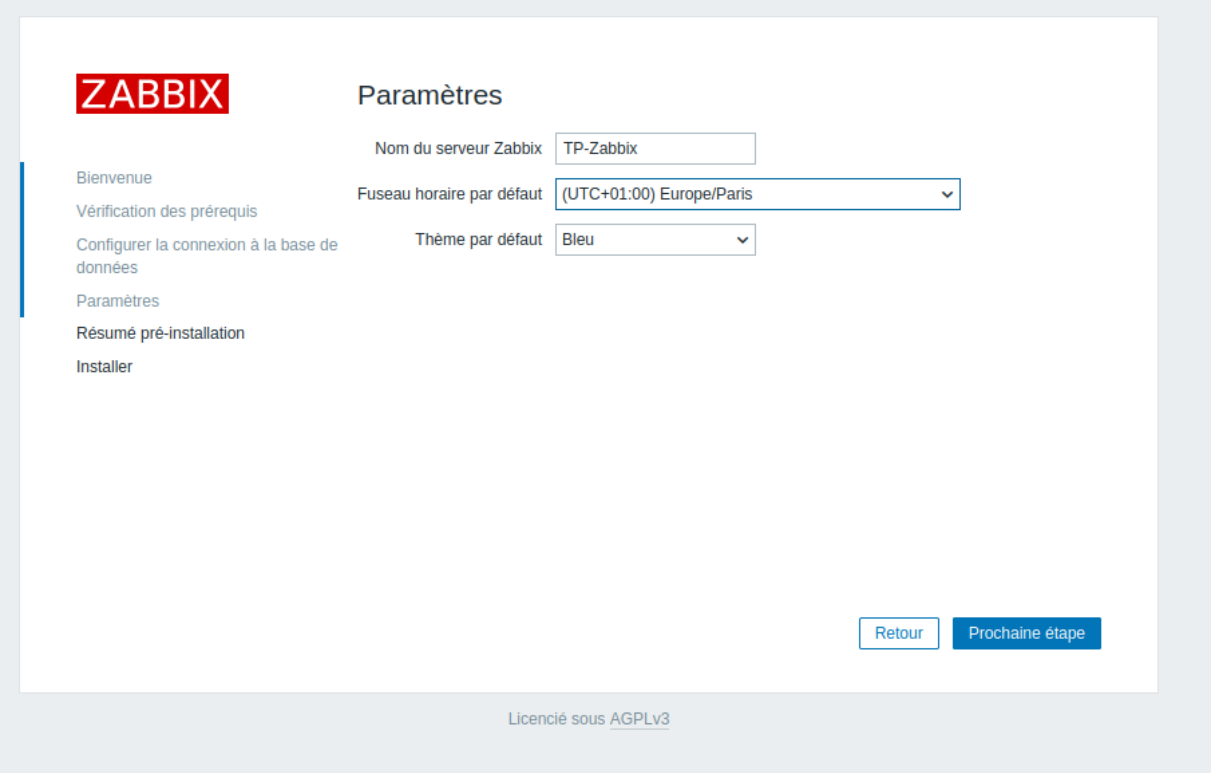
Note technique

Si la langue française n'est pas proposée par défaut dans la liste, cela indique que le pack de langue est absent du système. Pour corriger cela, il est nécessaire de générer les locales françaises sur le serveur en exécutant la commande **« sudo dpkg-reconfigure locales »**, en cochant l'option « **fr_FR.UTF-8 »** puis en redémarrant le service web Apache **« sudo systemctl restart apache2 »** pour que le changement soit pris en compte.

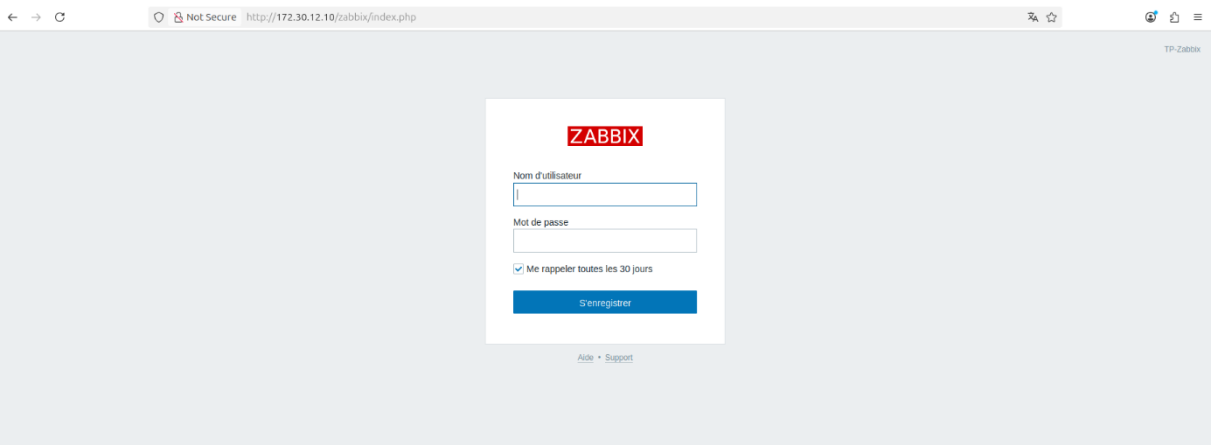


J'ai poursuivi l'assistant d'installation jusqu'à l'étape de configuration de la base de données. Ici, j'ai renseigné les identifiants créés précédemment en ligne de commande : le nom de la base (zabbix), l'utilisateur (zabbix) et le mot de passe associé. En conservant l'hôte sur localhost, j'ai validé cette étape pour permettre à l'interface web de communiquer avec le serveur MariaDB.

J'ai ensuite défini les paramètres généraux de l'instance. J'ai nommé le serveur **« TP-Zabbix »** pour l'identifier clairement et j'ai configuré le fuseau horaire sur **« Europe/Paris »**. Ce réglage est essentiel pour garantir que les horodatages des futures alertes et des graphiques correspondent bien à l'heure locale.



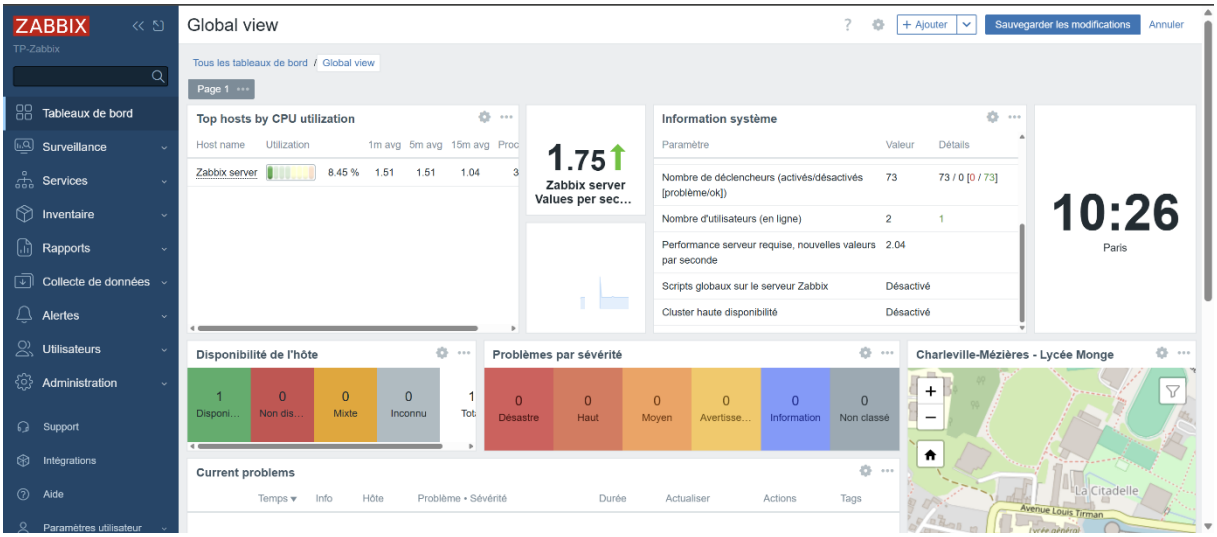
L'assistant de configuration s'est terminé avec succès, finalisant ainsi l'installation complète de la solution. J'ai été redirigé vers l'écran d'authentification où j'ai pu effectuer ma première connexion en utilisant les identifiants par défaut : l'utilisateur Admin (avec une majuscule) et le mot de passe zabbix.



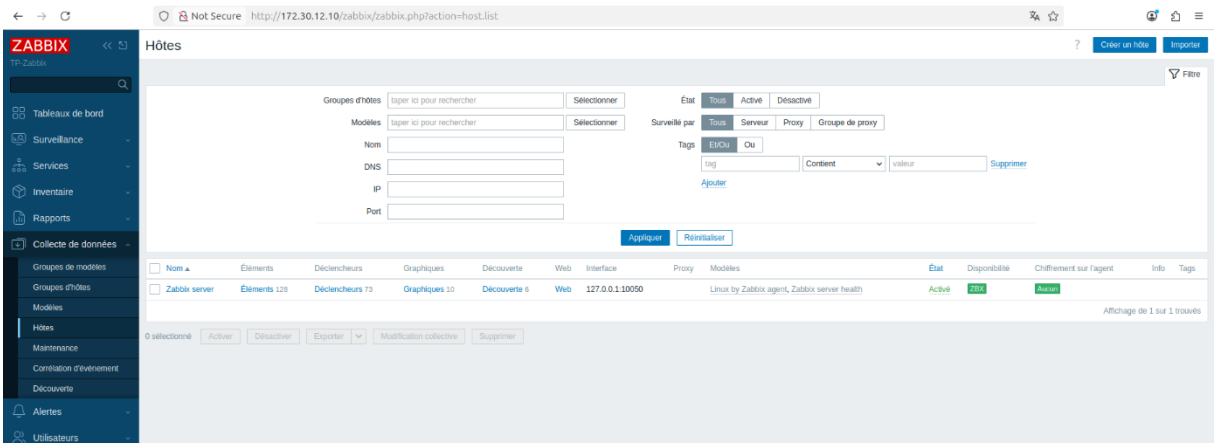
Une fois authentifié, j'ai accédé au **Tableau de bord global** (Global view). Cette interface centrale confirme la réussite de l'installation :

L'encart « Disponibilité de l'hôte » affiche un indicateur vert (ZBX), prouvant que le serveur Zabbix communique correctement avec son propre agent local.

Les widgets de performance (comme l'utilisation CPU) remontent déjà des données en temps réel, validant que la chaîne de supervision est pleinement opérationnelle.

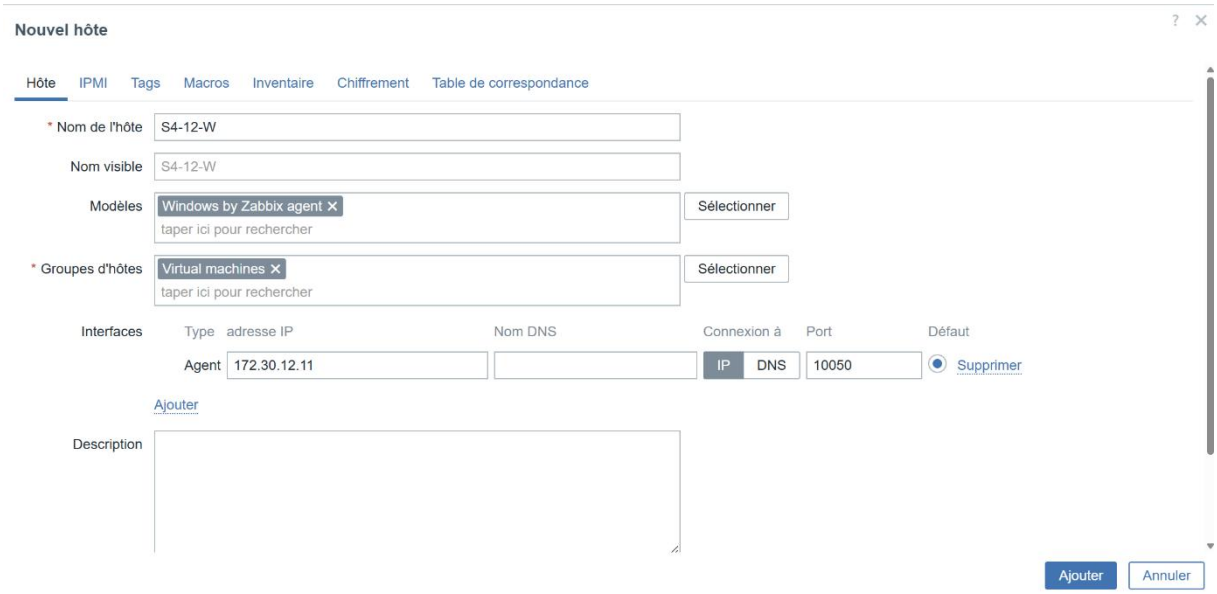


Je me suis ensuite rendu dans le menu **« Collecte de données > Hôtes »**. On y constate la présence par défaut de notre serveur Zabbix (sous Ubuntu) qui est déjà opérationnel. Je l'utiliserai d'ailleurs ultérieurement comme base de test pour simuler des pannes et vérifier la bonne remontée des erreurs.



Pour l'heure, j'ai procédé à l'ajout de la machine client en cliquant sur le bouton

« Créer un hôte » afin de configurer la supervision de l'environnement Windows.



Dans le formulaire de création du nouvel hôte, j'ai renseigné les informations essentielles pour établir la connexion avec la machine virtuelle Windows :

Nom de l'hôte : J'ai nommé la machine **« S4-12-W »** pour l'identifier unique dans l'inventaire.

Modèles (Templates) : J'ai appliqué le modèle **« Windows by Zabbix agent »**. Ce choix est crucial car il pré-configure automatiquement tous les éléments à surveiller spécifiques à Windows (CPU, mémoire, disque, services, etc.) sans avoir à les créer un par un.

Interfaces : J'ai défini l'adresse IP de l'agent **« 172.30.12.11 »** sur le port standard **« 10050 »**, permettant au serveur Zabbix de contacter la machine cible.

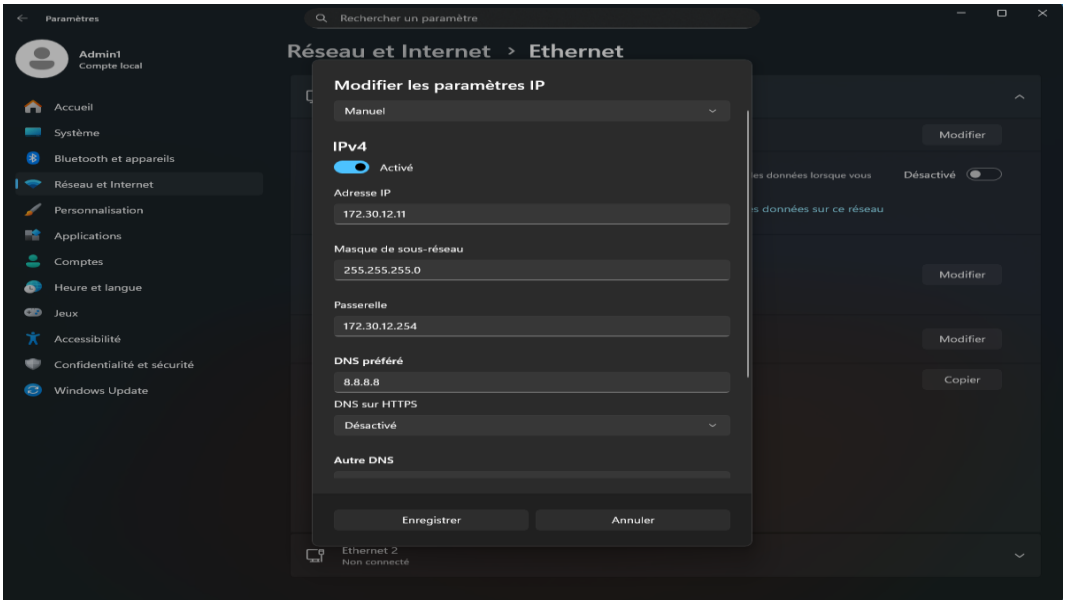
Partie Client : Installation de l'agent Zabbix sur Windows

Pour permettre la remontée des informations vers le serveur, il est nécessaire d'installer l'agent Zabbix sur la machine cible (Windows 10/11).

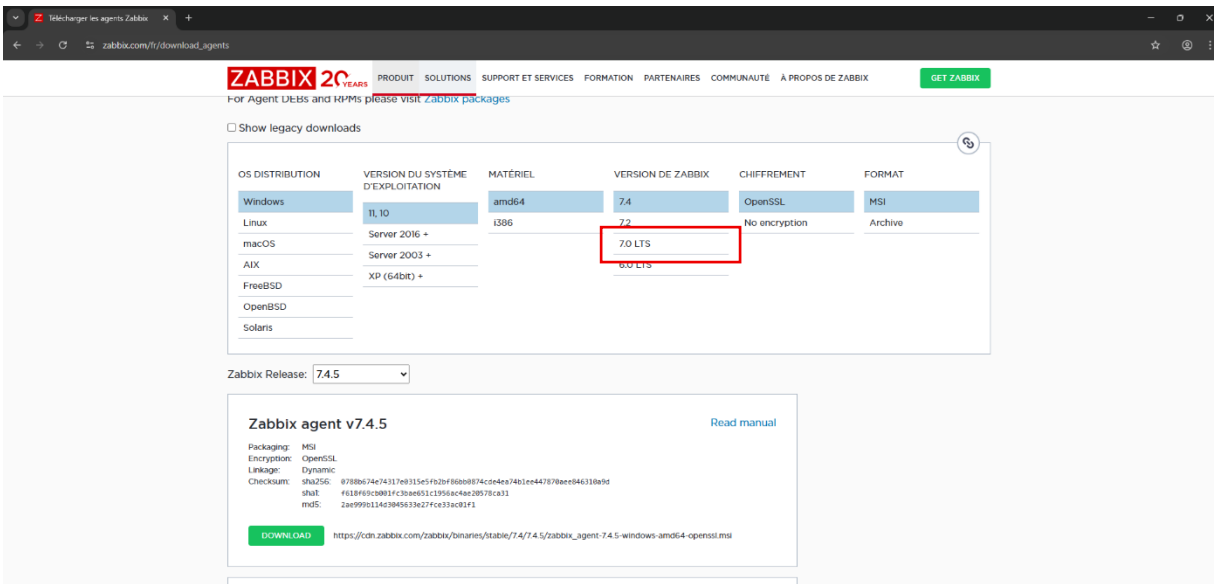
Avant de procéder à l'installation de l'agent, j'ai d'abord dû configurer la connectivité réseau du poste client. Pour garantir une communication stable et permanente avec le serveur de supervision, il est impératif que les deux machines se trouvent sur le même réseau logique.

J'ai donc désactivé l'attribution automatique (DHCP) pour définir une adresse IP statique. J'ai attribué l'adresse **« 172.30.12.11 »** au poste Windows, ce qui le place dans la même plage

d'adresses que le serveur Zabbix (qui est en .10). Cette étape est indispensable pour que le serveur puisse retrouver et interroger l'agent sans interruption.



J'ai ensuite récupéré l'exécutable d'installation (format .msi) sur le site officiel de Zabbix.



Point de vigilance technique

J'ai spécifiquement choisi de télécharger et d'installer la version **« Zabbix Agent 7.0 LTS. »** En effet, j'ai écarté la toute dernière version disponible car elle présentait des instabilités notables (bugs empêchant la remontée correcte des données). La version 7.0 assure une stabilité optimale pour notre supervision.

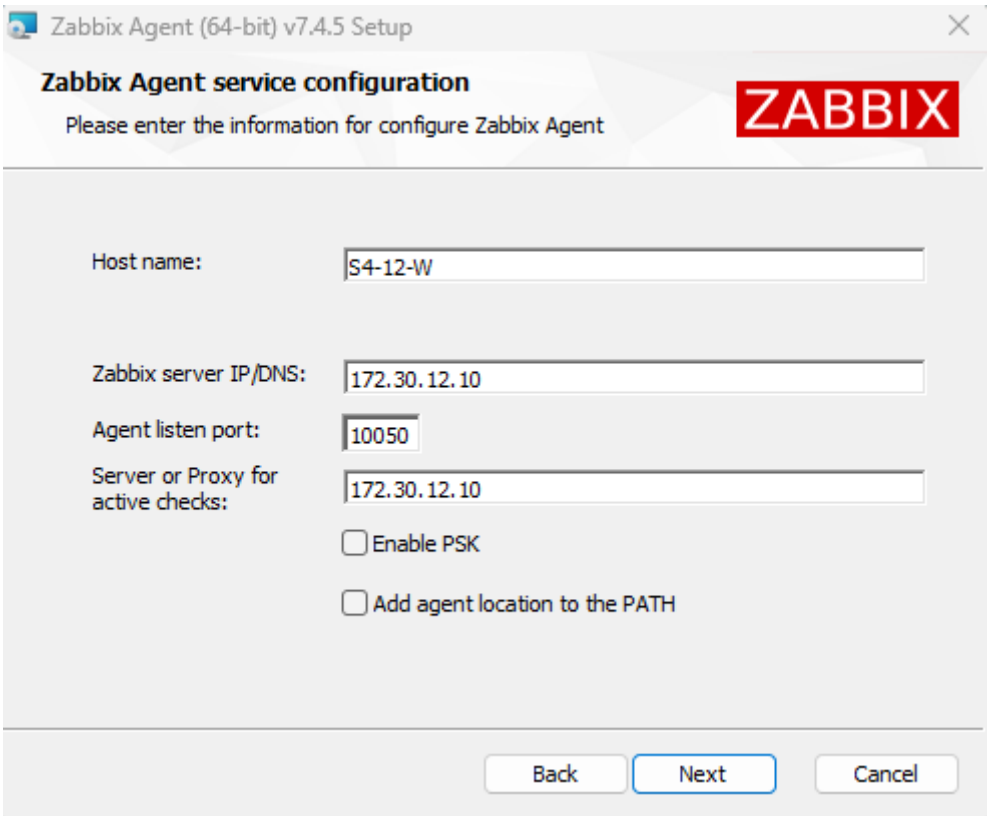
Une fois l'installation lancée, l'étape la plus critique concerne la configuration de la communication avec le serveur. J'ai renseigné les champs suivants :

Host name : J'ai saisi **« S4-12-W ».** Ce nom agit comme un identifiant unique. Il est impératif qu'il corresponde à la lettre près au nom d'hôte que je déclarerai plus tard sur l'interface web du serveur Zabbix. Une différence, même de majuscule, empêcherait le serveur de reconnaître cette machine.

Zabbix server IP/DNS : J'ai indiqué l'adresse IP de notre serveur de supervision Ubuntu, soit **« 172.30.12.10 »**. Ce champ agit comme une liste blanche (whitelist) de sécurité : l'agent n'acceptera de recevoir des instructions (mode passif) que si elles proviennent de cette adresse IP spécifique.

Server or Proxy for active checks : J'ai reporté la même adresse IP **« 172.30.12.10 »**. Cela permet à l'agent d'envoyer lui-même spontanément des données au serveur (mode actif), par exemple pour des journaux d'événements.

Agent listen port : J'ai conservé le port par défaut **« 10050 »,** qui est le standard pour l'écoute de l'agent Zabbix.

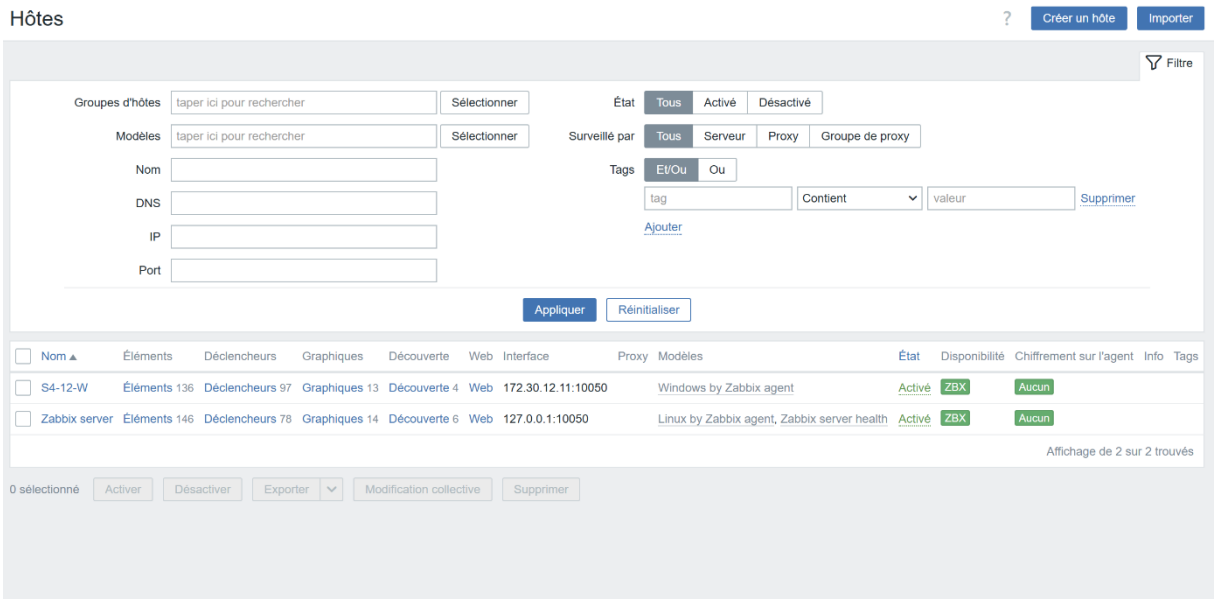


J'ai finalisé l'installation en laissant les autres paramètres par défaut. Le service

« Zabbix Agent » s'est lancé automatiquement en arrière-plan. J'ai ensuite vérifié dans la console des services Windows qu'il était bien en cours d'exécution, afin de garantir qu'il transmettrait correctement les données au serveur.

Grâce à la correspondance stricte des noms d'hôtes (S4-12-W) et à la bonne configuration réseau (IP statiques), la communication entre le serveur et l'agent s'est établie automatiquement après quelques instants.

En actualisant la liste des hôtes dans l'interface Zabbix, j'ai constaté que l'indicateur de disponibilité **« ZBX »** est passé au vert pour notre machine cliente. Cela confirme que l'agent remonte correctement les métriques de performance.

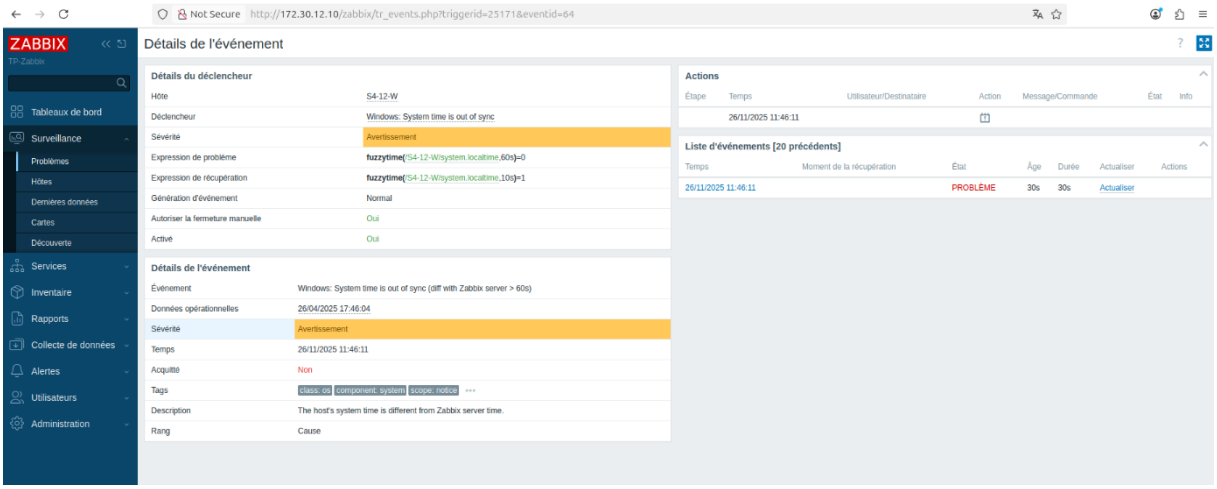


L'infrastructure de supervision étant désormais pleinement opérationnelle, je vais pouvoir passer à la phase de test pratique, la simulation d'incidents. Je vais provoquer volontairement des pannes pour vérifier que Zabbix détecte bien les anomalies et déclenche les alertes attendues.

Afin de valider la capacité de détection d'incidents de Zabbix, j'ai réalisé un premier test pratique en modifiant volontairement l'heure système de la machine Windows. J'ai introduit un décalage de plusieurs heures par rapport à l'heure réelle du serveur.

Cette manipulation a pour but de vérifier si la règle de surveillance

« System time is out of sync » s'active correctement. La synchronisation horaire est en effet un paramètre critique (notamment pour les logs et l'authentification), et Zabbix est configuré pour alerter dès que la différence dépasse 60 secondes.



Le test s'est révélé concluant quasi instantanément. Quelques instants après la modification manuelle de l'horloge sur le poste client, une nouvelle entrée est apparue dans la liste des problèmes Zabbix.

Comme en témoigne la capture d'écran des détails de l'événement :

Le déclencheur **« Windows: System time is out of sync »** s'est activé automatiquement.

L'incident est correctement identifié sur l'hôte **« S4-12-W »**

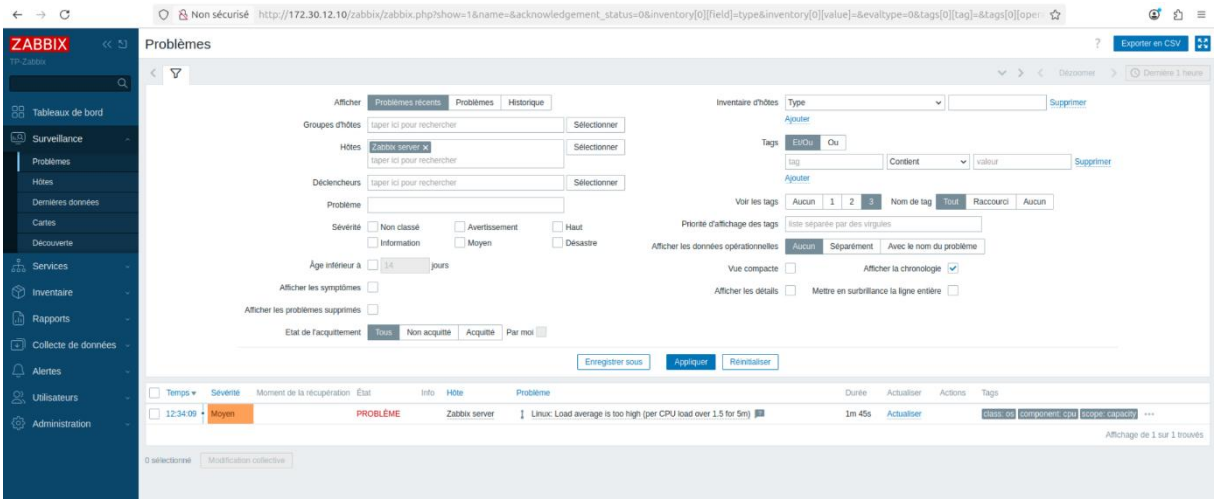
La sévérité a été classée en **« Avertissement »** (bandeau jaune), conformément à la configuration par défaut du modèle Windows.

Cette alerte confirme que l'agent interroge régulièrement le système (toutes les minutes par défaut pour ce paramètre) et que le serveur traite correctement les anomalies remontées.

Simulation d'une surcharge CPU sur le serveur Linux

Après avoir validé la remontée d'alertes sur Windows, j'ai souhaité vérifier la surveillance des performances sur le serveur Linux lui-même. Pour ce scénario, j'ai volontairement provoqué une surcharge du processeur (stress test) afin de saturer les ressources de la machine.

L'objectif était de déclencher le déclencheur (trigger) qui surveille la charge moyenne du système. Comme le montre la capture d'écran, le test a fonctionné parfaitement :



Une alerte de sévérité **« Moyen »** (Orange) s'est déclenchée sur l'hôte **« Zabbix server »**.

Le motif du problème est explicitement indiqué : **« Linux: Load average is too high (per CPU load over 1.5 for 5m) »**.

Cela signifie que Zabbix a bien détecté que la charge par cœur a dépassé le seuil critique de 1.5 pendant plus de 5 minutes, validant ainsi la bonne configuration des modèles de performance Linux.

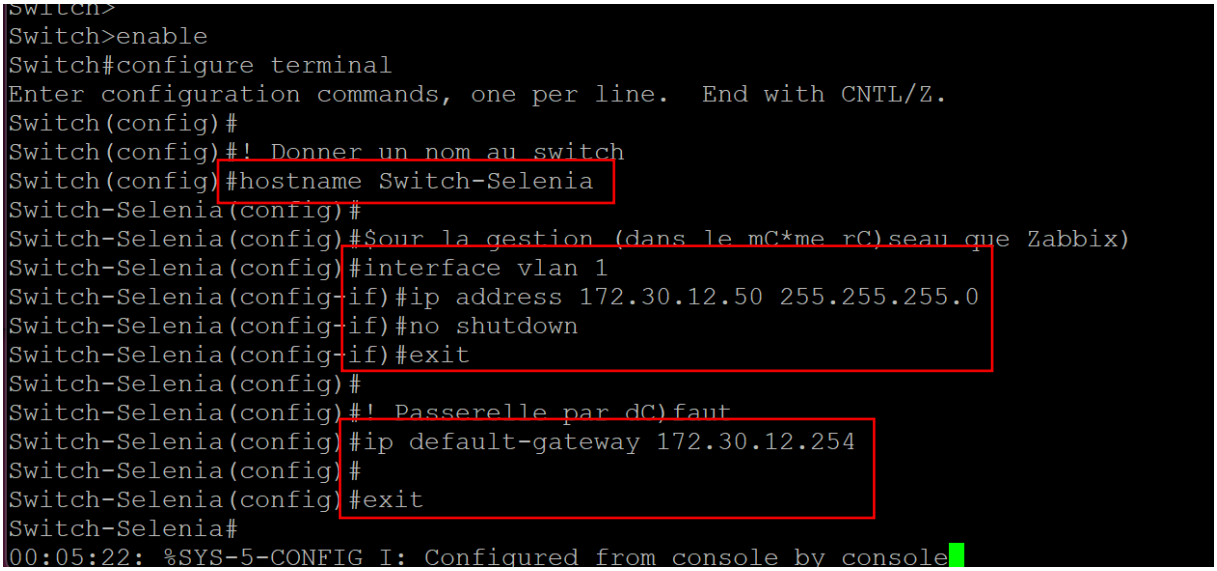
Après avoir validé la supervision des systèmes (Serveur Linux et Client Windows), j'ai étendu le périmètre de surveillance aux équipements réseau. Pour ce test, j'ai utilisé un switch Cisco que j'ai préalablement réinitialisé à ses paramètres d'usine (commande write erase puis redémarrage) afin de partir sur une configuration totalement vierge.

La première étape a consisté à rendre le switch accessible sur le réseau de supervision. Comme le montre la première capture de configuration :

Identité : J'ai nommé l'équipement **« Switch-Selenia »** pour l'identifier clairement.

Adressage IP : J'ai configuré l'interface de gestion (Vlan 1) avec l'adresse IP statique **« 172.30.12.50 »**, située dans le même sous-réseau que notre serveur Zabbix.

Routage : J'ai défini la passerelle par défaut **« 172.30.12.254 »** pour permettre la communication avec les autres réseaux si nécessaire.



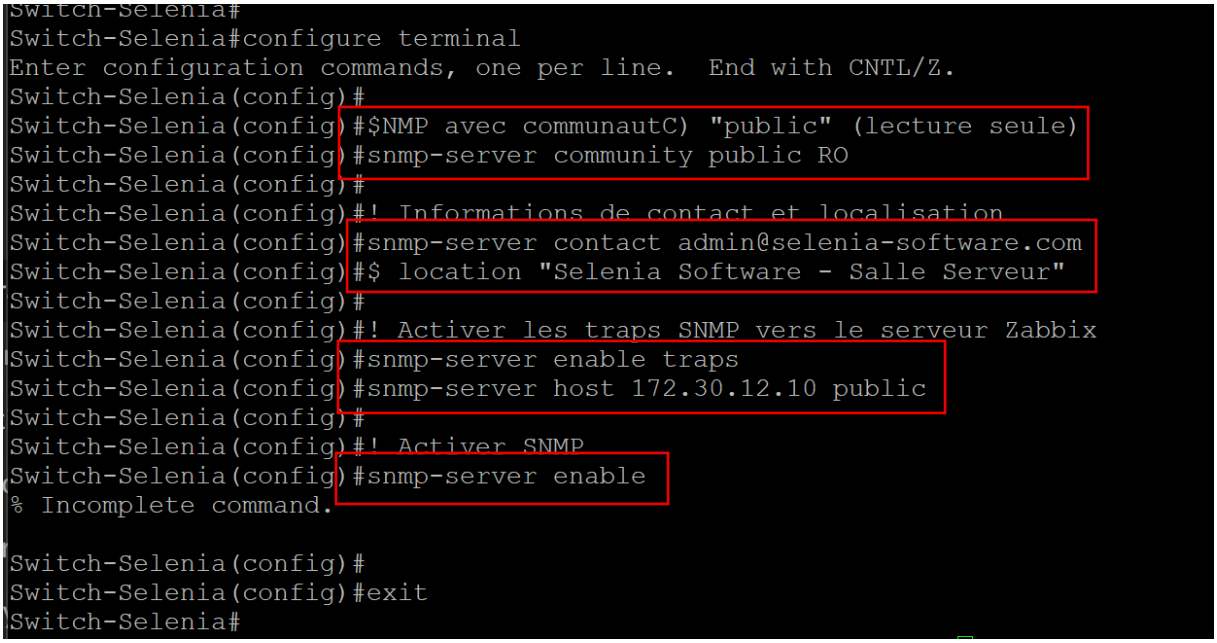
L'adressage IP étant configuré, il a fallu activer le service **SNMP** (Simple Network Management Protocol) sur le commutateur. C'est ce protocole standard qui permet à Zabbix de récupérer les métriques (trafic, état des ports, CPU) à distance.

Comme le montre la capture d'écran, j'ai exécuté les commandes suivantes pour sécuriser et diriger les échanges :

Définition de la communauté (snmp-server community public RO) : J'ai défini le mot de passe d'accès (appelé "chaîne de communauté") sur **« public »**. J'ai restreint les droits en **RO** (Read-Only / Lecture Seule). Cela permet au serveur Zabbix de lire les informations sans pouvoir modifier la configuration du switch, ce qui est une bonne pratique de sécurité.

Métadonnées administratives : J'ai renseigné l'email de l'administrateur et l'emplacement physique « Salle Serveur » pour faciliter la gestion de l'inventaire dans l'interface Zabbix.

Activation des alertes (Traps) : J'ai activé l'envoi de « Traps » (snmp-server enable traps) et désigné notre serveur Zabbix comme destinataire via la commande **« snmp-server host 172.30.12.10 public ».** Ainsi, en cas de panne majeure, le switch enverra spontanément une alerte au serveur sans attendre d'être interrogé.



La configuration du switch étant terminée, je suis retourné sur l'interface web du serveur pour l'ajouter officiellement à la supervision. J'ai créé un nouvel hôte avec des paramètres spécifiques au matériel réseau :

Nom de l'hôte : J'ai saisi **« Switch-Selenia ».**

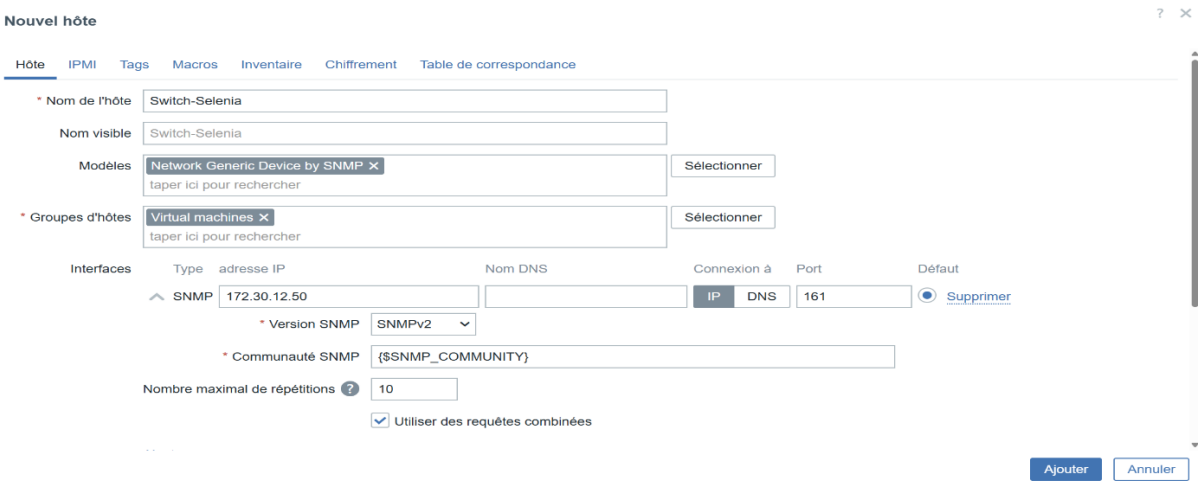
Modèles (Templates) : Au lieu d'utiliser un agent (impossible sur un switch Cisco standard), j'ai sélectionné le modèle **« Network Generic Device by SNMP ».** Ce modèle est conçu pour interroger les équipements via le protocole SNMP et remonter automatiquement l'état des ports, le trafic et la charge CPU.

Interfaces : J'ai cliqué sur "Ajouter" puis **SNMP** (et non "Agent"). J'y ai renseigné l'IP du switch **« 172.30.12.50 »** sur le port standard **« 161 ».**

Configuration SNMP :

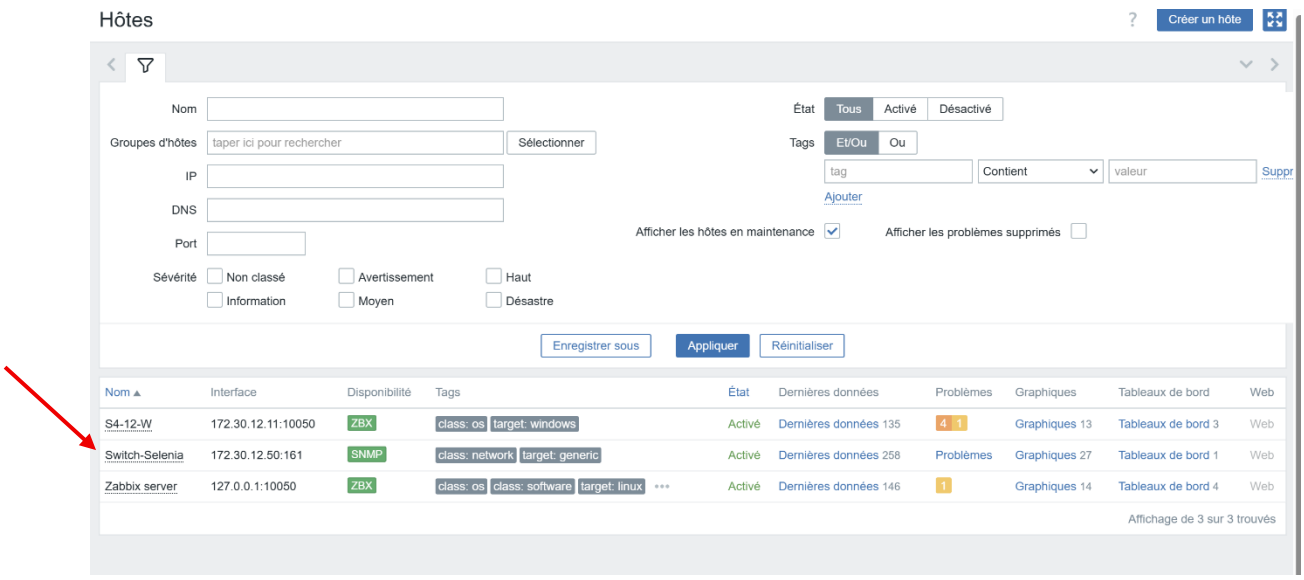
Version : J'ai laissé **SNMPv2**, le standard le plus courant.

Communauté : Le champ indique {$SNMP_COMMUNITY}. C'est une macro (variable) par défaut de Zabbix qui correspond à la valeur **« public ».** Cela tombe bien, car c'est exactement le mot de passe que j'ai configuré sur le switch juste avant.



Après avoir validé l'ajout, j'ai patienté quelques instants le temps que le serveur Zabbix effectue sa première interrogation (polling).

En consultant la liste des hôtes, j'ai eu la confirmation que tout fonctionnait : L'étiquette **SNMP** est apparue en vert sur la ligne du Switch-Selenia. Cela prouve que le serveur arrive à dialoguer avec le switch, à s'authentifier avec la communauté "public" et à récupérer des données.



Test d'incident réseau : Simulation d'une boucle (Loop)

Pour valider la supervision de l'équipement réseau, j'ai réalisé un test physique en provoquant volontairement une boucle réseau sur le port **Fa0/9** du switch.

Observation du comportement : Lors du branchement du câble (boucle active), j'ai constaté qu'aucune alerte ne remontait dans Zabbix. Bien que je n'aie pas pu isoler la cause exacte de ce comportement durant la séance, il est probable que le switch ait géré la boucle via le protocole **Spanning Tree (STP)** en bloquant le trafic logiquement, tout en maintenant le statut physique du port en "Up". Zabbix voyant le port allumé, il n'a pas considéré cela comme une panne.

Déclenchement de l'alerte : L'alerte s'est déclenchée uniquement lorsque j'ai débranché le câble pour mettre fin au test. À cet instant précis, l'état de l'interface a physiquement changé (passage en "Down"). Zabbix a alors immédiatement détecté la coupure du lien et a généré l'alerte visible sur la capture : **« Network Generic Device: Interface Fa0/9(): Link down »**.

Sources :

Pour la réalisation de ce projet, je me suis appuyé sur la documentation officielle Zabbix. J'ai également utilisé l'IA Gemini pour m'aider à rédiger, corriger et reformuler ce compte-rendu. Enfin, je remercie mes camarades pour l'entraide apportée lors de la résolution des bugs rencontrés.

Conclusion :

Ce projet de déploiement de Zabbix s'est révélé être une expérience enrichissante et globalement accessible sur le plan technique. La mise en place du serveur et des agents sur les différents systèmes (Windows et Linux) s'est déroulée sans encombre, me permettant de valider rapidement les objectifs principaux de la mission.

Cependant, j'ai pu constater certaines contraintes inhérentes à la supervision, notamment l'inertie dans la remontée des alertes. Les délais de rafraîchissement (polling) rendent parfois les phases de test fastidieuses, obligeant à patienter plusieurs minutes pour valider une configuration.

Enfin, une difficulté technique a persisté concernant la supervision du switch Cisco. Malgré mes tentatives de configuration je n'ai pas réussi à obtenir le déclenchement immédiat de l'alerte lors de la création de la boucle réseau. L'incident n'a été remonté par Zabbix qu'au moment de la déconnexion du câble, un comportement que je n'ai pas pu corriger.

En définitive, ce TP m'a permis de comprendre l'importance d'une infrastructure de monitoring, tout en mettant en lumière la complexité du réglage fin des équipements réseau pour une détection d'incidents en temps réel.