HESTIM
ENGINEERING & BUSINESS SCHOOL

Ingénierie Informatique, Intelligence Artificielle Et Cybersécurité

Projet Zabbix :

Mise en œuvre d'une infrastructure cloud de supervision centralisée sous AWS : Déploiement de Zabbix conteneurisé pour le monitoring d'un parc hybride (Linux & Windows)

Réalisé par :

MAKOSSO Dylan
Elohim Vianney

Encadré par :

Pr. Azeddine KHIAT

Année universitaire
2025/2026

Table des matières :

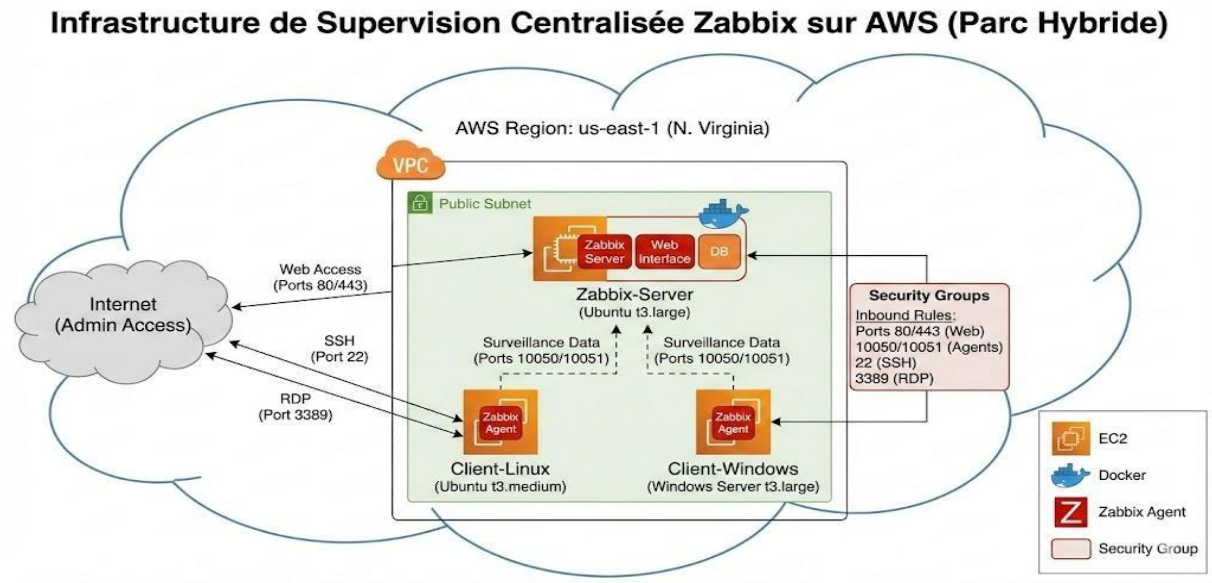
Introduction :

Dans un contexte où les infrastructures informatiques deviennent de plus en plus complexes et critiques, la supervision en temps réel des systèmes représente un élément essentiel pour garantir la disponibilité, la performance et la continuité des services.Ce projet vise ainsi à concevoir et déployer une infrastructure de supervision centralisée au sein d'un environnement Cloud, en s'appuyant sur trois technologies fondamentales qui constituent les bases de son architecture.

Amazon Web Services (AWS) est utilisé comme plateforme Cloud pour héberger l'ensemble de l'infrastructure. À travers des services tels que le VPC (Virtual Private Cloud), qui garantit l'isolation du réseau, et les instances EC2 dédiées à la puissance de calcul, AWS permet de reproduire un environnement d'entreprise flexible, évolutif et sécurisé. Dans le cadre de ce projet, la région us-east-1 est exploitée, avec des types d'instances adaptés aux contraintes du Learner Lab, notamment t3.medium et t3.large.

Docker et la conteneurisation , le déploiement du serveur Zabbix est réalisé à l'aide de Docker et Docker Compose. La conteneurisation assure une portabilité optimale en isolant le serveur Zabbix, son interface Web ainsi que sa base de données dans des environnements légers, indépendants et facilement reproductibles. Cette approche simplifie considérablement l'installation, la configuration et la maintenance de la solution de supervision sur une instance Ubuntu.

Zabbix : supervision en temps réel , zabbix a été retenu comme solution de monitoring en raison de sa robustesse et de sa polyvalence. L'objectif est de mettre en place une supervision centralisée d'un environnement hybride. Grâce à l'installation d'agents sur des machines Linux (Ubuntu) et Windows Server, Zabbix permet la collecte et l'analyse en temps réel de métriques critiques telles que l'utilisation du CPU ou de la mémoire vive (RAM), offrant ainsi une visibilité complète sur l'état de l'infrastructure.



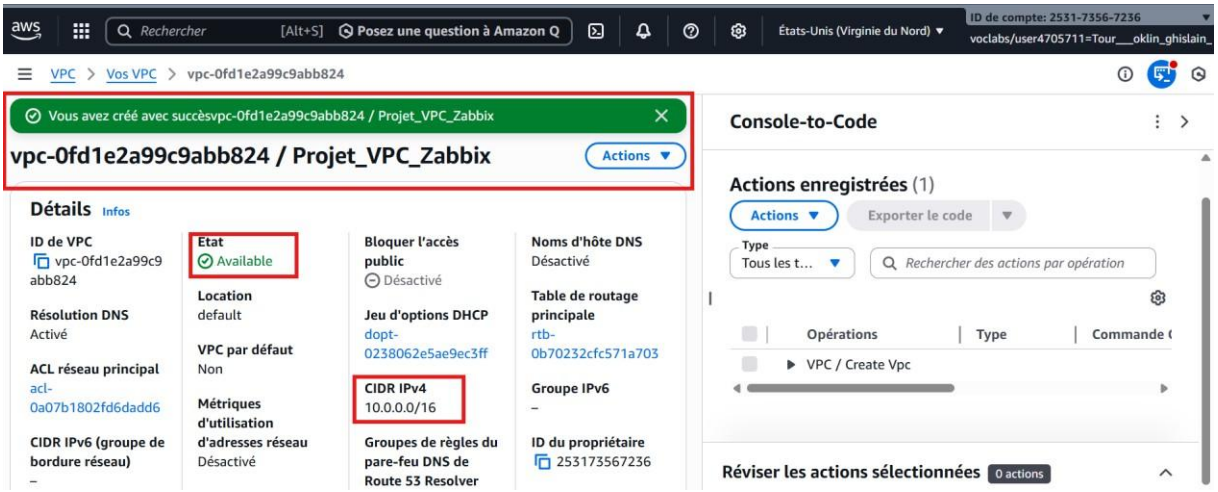
1 Architecture Réseau :

Elle détaille la configuration de l'environnement réseau sécurisé nécessaire au bon fonctionne- ment de la supervision.

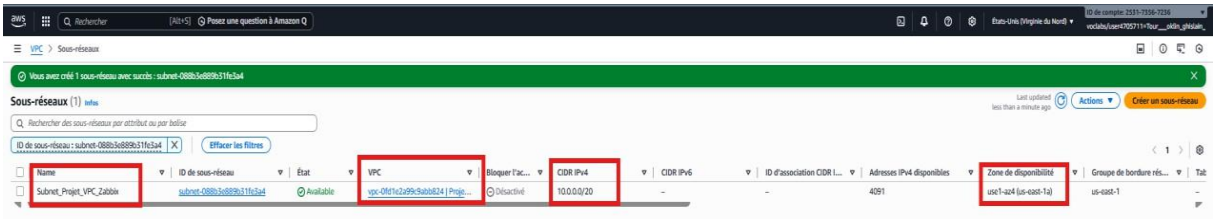
1.1 Configuration du VPC et des Sous-réseaux :

Pour ce projet, un **VPC (Virtual Private Cloud)** dédié a été créé dans la région **us-east-1**. Afin de simplifier l'accès aux interfaces de gestion sans nécessiter de tunnel VPN, un **sous-réseau public** a été configuré, permettant l'attribution d'adresses IP publiques aux instances.

1.1.1 Création du réseau virtuel (Projet_VPC_Zabbix) avec bloc CIDR : 10.0.0.0/16

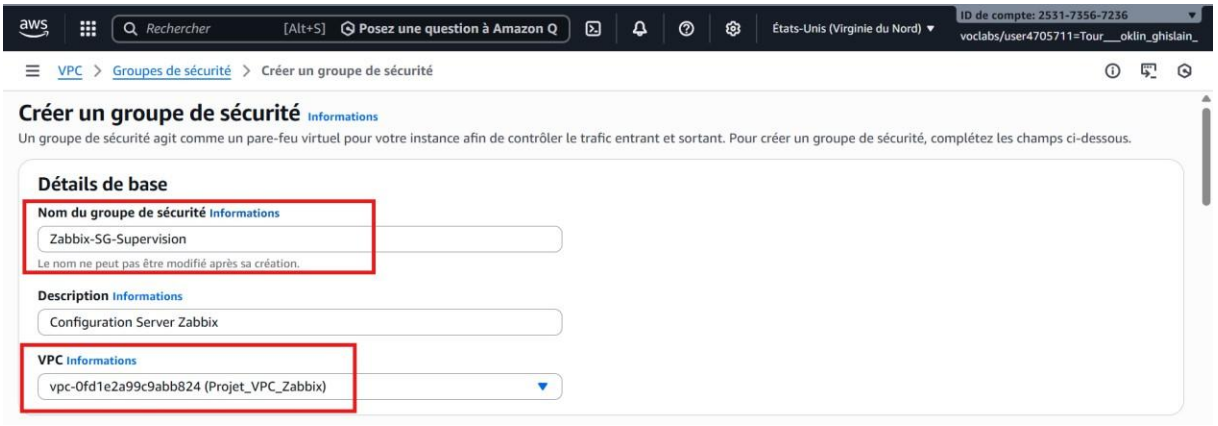


1.1.2 Création du sous réseau virtuel :(Subnet_Projet_VPC_Zabbix) au sein du même VPC avec un CIDR 10.0.0.0/20

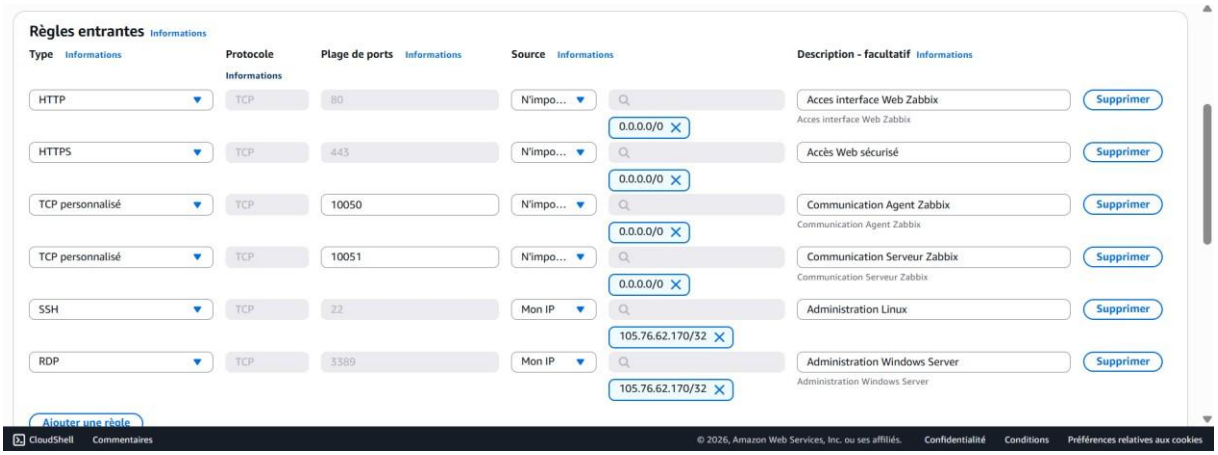


1.2 Configuration de la Sécurité :(Security Groups)

La sécurité de notre infrastructure cloud repose sur la configuration rigoureuse des **Security Groups** (groupes de sécurité). Ils agissent comme un pare-feu virtuel au niveau de l'instance pour contrôler le trafic entrant et sortant.

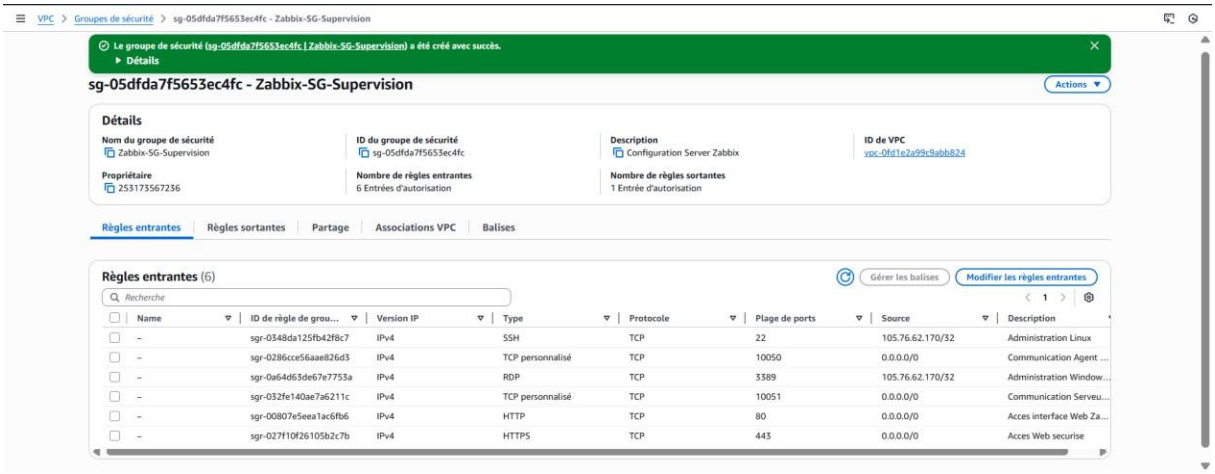


Pour permettre le monitoring centralisé du parc hybride tout en maintenant un accès d'administration sécurisé, nous avons configuré les règles entrantes suivantes sur notre VPC:



1.2.1 Sécurisation des flux : Groupe de sécurité :(Security Group)

Pour garantir l'intégrité de notre infrastructure de supervision, nous avons mis en place un groupe de sécurité nommé Zabbix-SG-Supervision. Ce pare-feu virtuel est configuré selon le principe du moindre privilège pour les accès administratifs tout en permettant l'ouverture des flux nécessaires au monitoring hybride.



1.3 Architecture des Instances EC2 :

Elle détaille le déploiement opérationnel des serveurs au sein du sous-réseau public du VPC. Le choix des types d'instances a été optimisé pour répondre aux exigences de performance de Zabbix et du client Windows tout en respectant les limites du budget AWS Learner Lab.

1.3.1 Tableau Récapitulatif du Parc Informatique :

Le parc est composé de trois instances distinctes formant un environnement de supervision hybride.

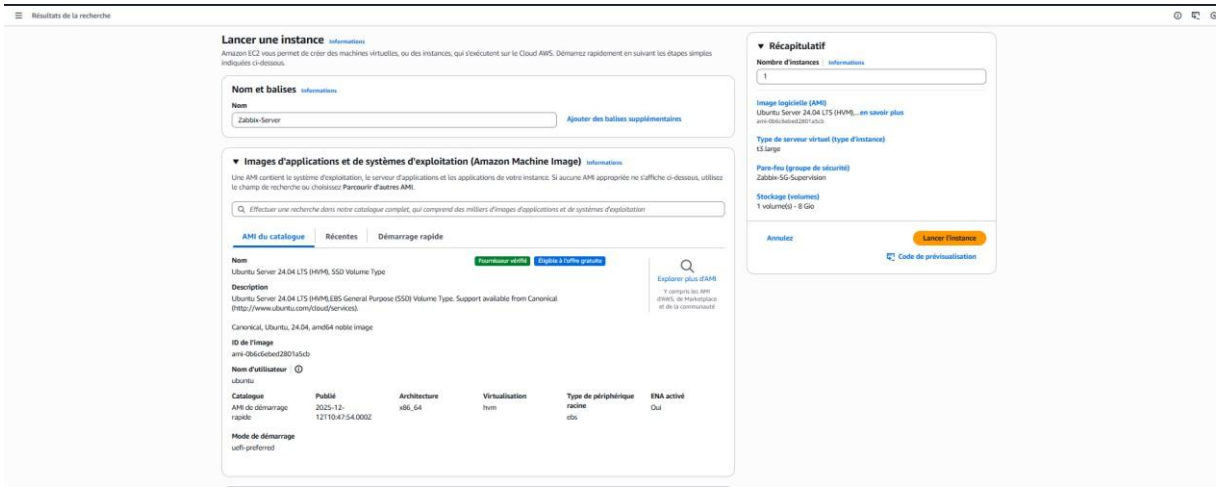
Nom de l'instance	**Système d'Exploitation (AMI)**	**Type d'instance**	**Rôle et Justification**
Zabbix-Server	Ubuntu 22.04 LTS	**t3.large**	Serveur central hébergeant Docker et la stack Zabbix.
Client-Linux	Ubuntu 22.04 LTS	**t3.medium**	Hôte Linux utilisé pour tester la remontée d'agents Zabbix.
Client-Windows	Windows Server 2022	**t3.large**	Hôte Windows nécessitant 8 Go de RAM pour une performance stable.

1.3.2 Procédure de configuration et de lancement :

Le lancement des instances suit une méthodologie rigoureuse pour garantir la connectivité et la sécurité du parc hybride.

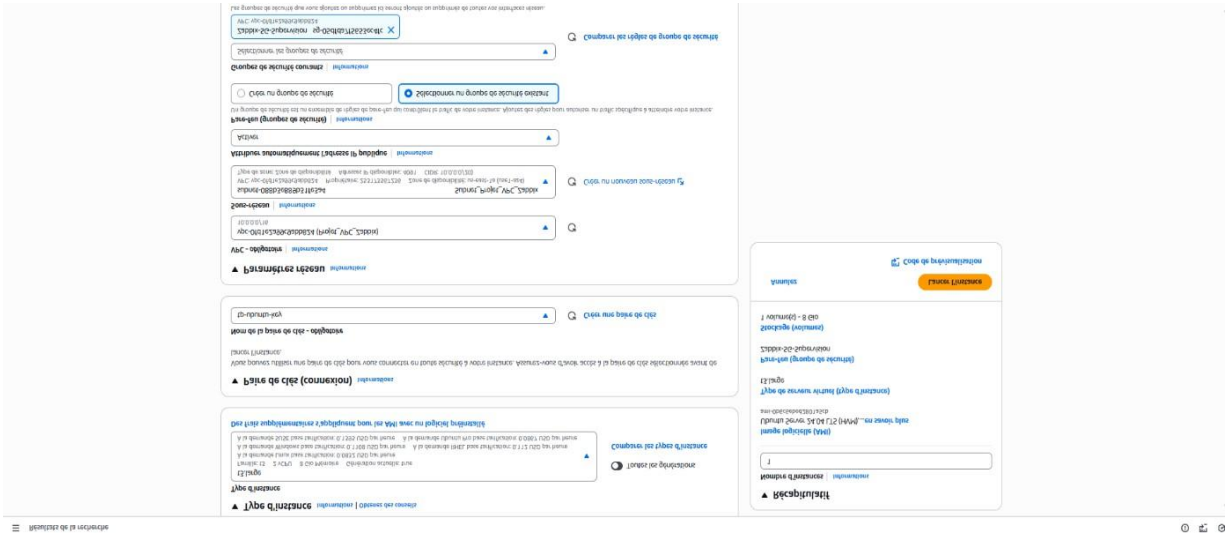
1.3.2.1 Sélection du système et du type d'instance :

Nous avons opté pour des images **Ubuntu 24.04 LTS** pour le serveur de supervision, offrant un support à long terme et une compatibilité optimale avec Docker. Le type d'instance **t3.large** a été choisi pour le serveur Zabbix afin de supporter la charge de la base de données et de l'interface web.



1.3.2.2 Configuration réseau et sécurité :

C'est l'étape critique où l'instance est rattachée à notre infrastructure personnalisée.



NB : Par souci de concision dans ce rapport, les captures d'écran ci-dessus illustrent la procédure détaillée pour l'instance **Zabbix-Server**. Cette méthodologie (sélection de l'AMI, choix du type d'instance, rattachage au VPC Projet_VPC_Zabbix et application du Security Group) a été **répétée à l'identique** pour le déploiement des instances **Client-Linux** et **Client-Windows**, en adaptant simplement le type d'instance (**t3.medium** pour Linux et **t3.large** pour Windows) conformément au tableau récapitulatif de la figure 7.

1.3.3 Validation du déploiement :

Une fois les instances lancées, nous vérifions leur état de fonctionnement ("Running") et la bonne attribution des adresses IP publiques dans le tableau de bord EC2.

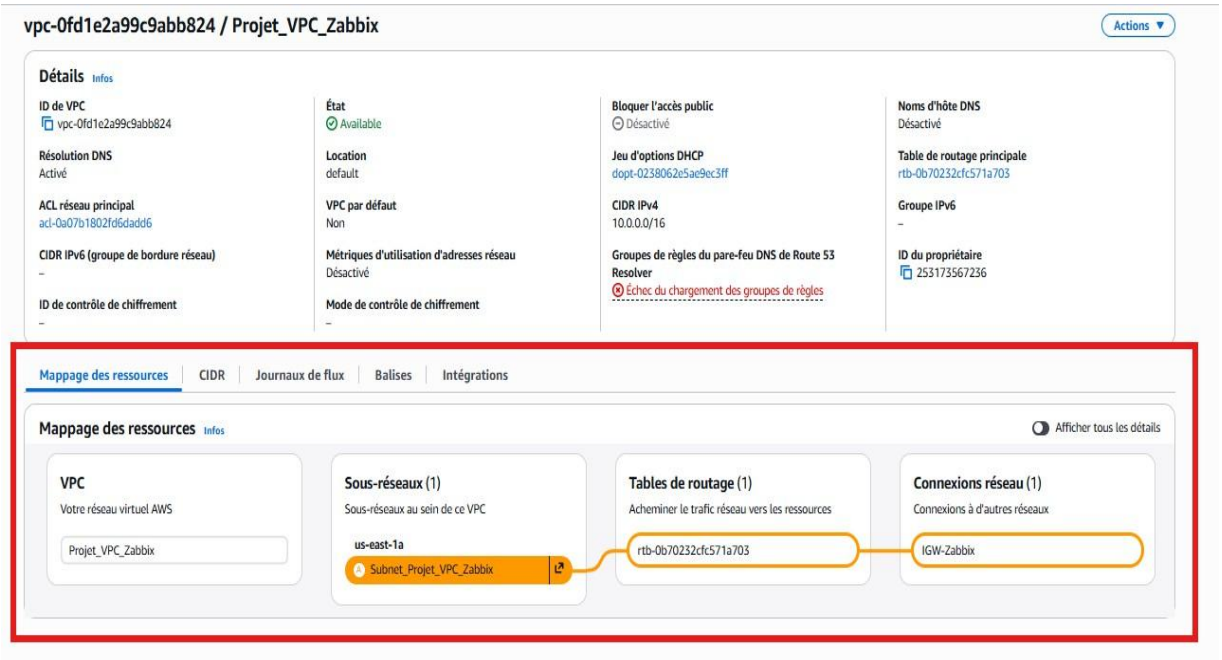
2 Déploiement du Serveur Zabbix : Installation et Configuration du Serveur Zabbix :

Cette phase consiste à transformer notre instance Ubuntu vierge en un serveur de supervision opérationnel grâce à Docker.

2.1 Installation de Docker et Docker-Compose :

Avant de procéder à l'installation de Docker, une phase de diagnostic réseau a été nécessaire pour établir la connexion avec nos instances.
Initialement, la connexion SSH vers l'instance **Zabbix-Server** était impossible (erreur *Connection Timeout*). L'analyse du mappage des ressources a révélé que le VPC était isolé, sans accès vers l'extérieur. Pour rétablir la connectivité, les actions suivantes ont été entreprises :

1. **Création d'une Internet Gateway (IGW) :** Nommée IGW-Zabbix, elle sert de pont entre le VPC et l'Internet.
2. **Mise à jour de la table de routage :** Ajout d'une route par défaut (0.0.0.0/0) pointant vers cette passerelle.



Une fois l'accès SSH rétabli, l'environnement a été préparé pour accueillir la stack Zabbix conteneurisée.

Étape 1 : Mise à jour et installation L'installation a été réalisée via le gestionnaire de paquets apt pour garantir la compatibilité avec Ubuntu 24.04 LTS.

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-10-0-4-89:~$ sudo apt update && sudo apt upgrade -y && sudo apt install docker.io docker-compose -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1408 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1697 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [314 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [16.0 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1519 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [310 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [380 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.6 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2426 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [554 kB]
```

Étape 2 : Vérification des versions Il est crucial de vérifier la bonne installation des outils avant de lancer les conteneurs.

```
ubuntu@ip-10-0-4-89:~$ docker --version
Docker version 28.2.2, build 28.2.2-0ubuntu1~24.04.1
ubuntu@ip-10-0-4-89:~$ docker-compose --version
docker-compose version 1.29.2, build unknown
ubuntu@ip-10-0-4-89:~$
```

2.2 Configuration de l'Orchestration (Docker Compose)

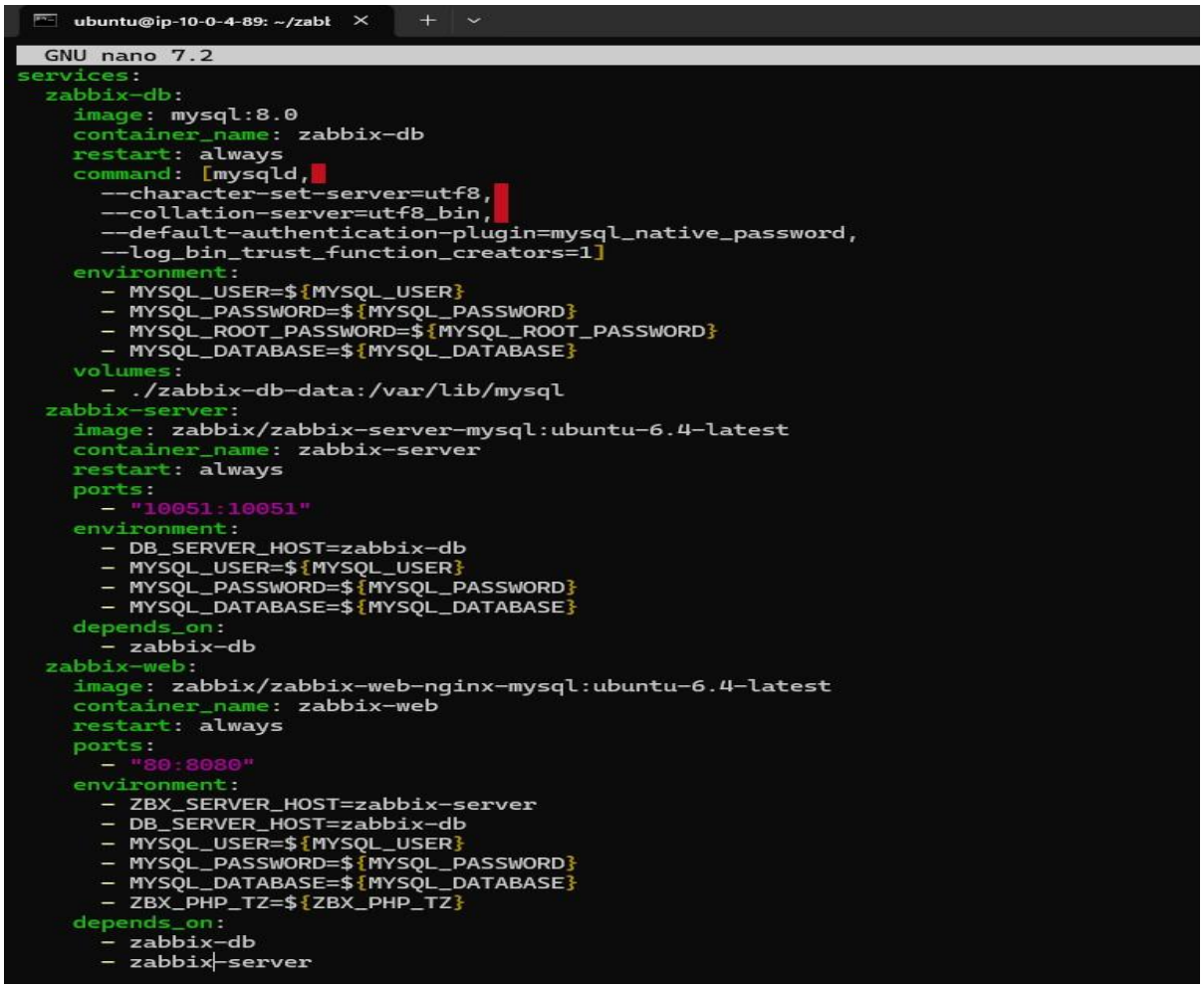
Après avoir préparé l'environnement Docker, nous structurons notre infrastructure de supervision à l'aide d'un fichier docker-compose.yml. Cette méthode permet de gérer le cycle de vie des trois services essentiels de manière atomique et cohérente.

2.2.1 Structure du ficher de déploiement :

Le fichier définit trois conteneurs distincts qui collaborent sur un réseau virtuel interne :

1. **zabbix-db :** Un moteur MySQL 8.0 pour le stockage persistant des données.
2. **zabbix-server :** Le cœur du système qui traite les données de supervision.
3. **zabbix-web :** L'interface utilisateur utilisant **Nginx** comme serveur web haute performance.

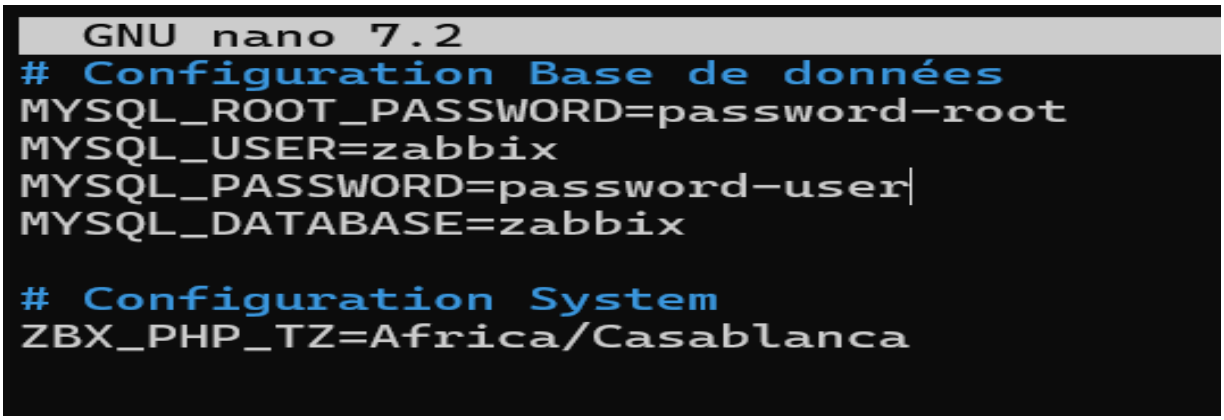
```
ubuntu@ip-10-0-4-89:~$ mkdir zabbix
ubuntu@ip-10-0-4-89:~$ cd zabbix
ubuntu@ip-10-0-4-89:~/zabbix$ nano docker-compose.yml
ubuntu@ip-10-0-4-89:~/zabbix$
```



2.2.2 Sécurisation et variables d'environnement :

Pour des raisons de **sécurité** et de **maintenabilité**, les informations sensibles (mots de passe, utilisateurs, fuseaux horaires) ne sont pas inscrites "en dur" dans le code principal.

- Nous utilisons un fichier **.env** séparé.
- Ce fichier permet de modifier la configuration globale du projet sans avoir à retoucher à la logique structurelle du fichier YAML.

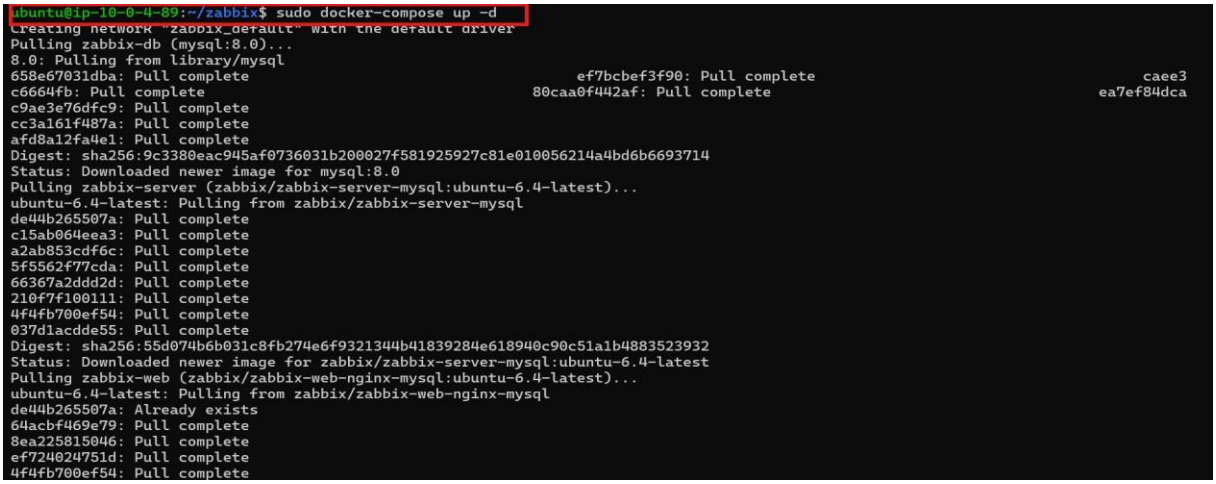


2.3 Lancement et Vérification de la Stack Zabbix :

Cette étape automatise la création de l'ensemble de l'architecture (Base de données, Serveur et Web).

2.3.1 Commande de déploiement :

Pour lancer l'infrastructure, nous utilisons la commande de déploiement de Docker Compose au sein du répertoire du projet :

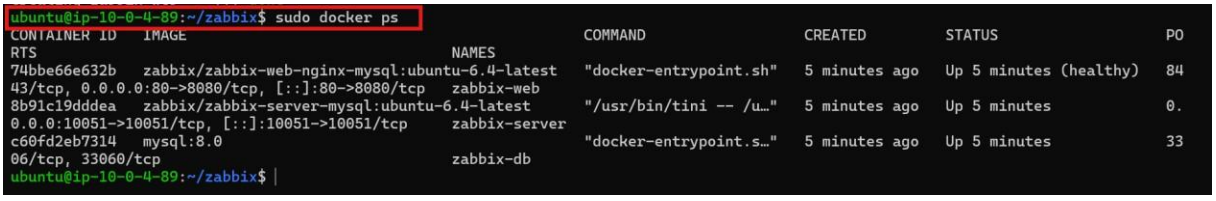


Analyse de la commande :

- **up** : Indique à Docker de construire, créer et démarrer les conteneurs.
- **-d (Detached mode)** : Cette option est cruciale ; elle permet d'exécuter les conteneurs en arrière-plan. Cela libère le terminal pour d'autres tâches d'administration tout en maintenant les services actifs.
- **Processus automatique** : Docker procède au "Pulling" (téléchargement) des images officielles (MySQL, Zabbix Server, Nginx) depuis le Docker Hub, crée un réseau virtuel isolé et respecte l'ordre de démarrage (la base de données avant le serveur).

2.3.2 Contrôle de l'état des services (Vérification)

Une fois le déploiement terminé, une vérification rigoureuse du statut des processus est effectuée avec la commande :



Conformité validés :

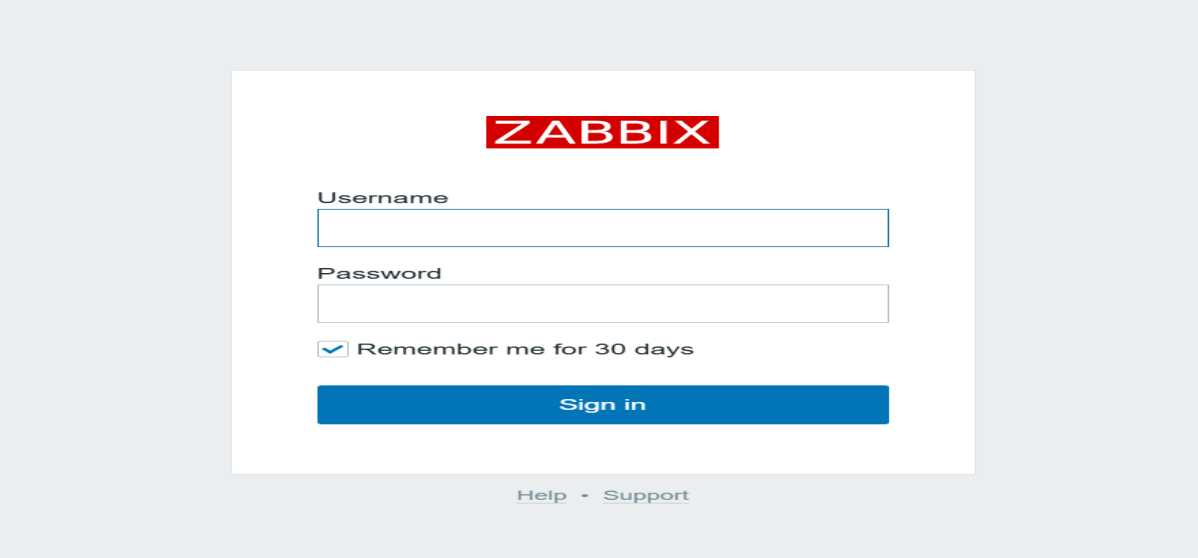
1. **Status (État)** : Les trois conteneurs (zabbix-web, zabbix-server, zabbix-db) affichent le statut **"Up"**, confirmant qu'aucune erreur critique n'est survenue au lancement.

2. **Mappage des Ports** :
    - Le port **80** (HTTP) de l'hôte est correctement redirigé vers le port **8080** du conteneur Web.
    - Le port **10051** est ouvert pour la communication interne des agents de supervision.

Conclusion de l'étape : La réussite de cette vérification confirme que l'interconnexion entre la base de données MySQL et les composants Zabbix est opérationnelle. L'interface de supervision est désormais accessible via l'adresse IP publique de l'instance.

2.3.3 Interface de Connexion Zabbix :

Une fois le déploiement Docker terminé, l'interface est devenue accessible via l'IP publique. Pour cette première phase, nous avons utilisé les identifiants d'administration par défaut.



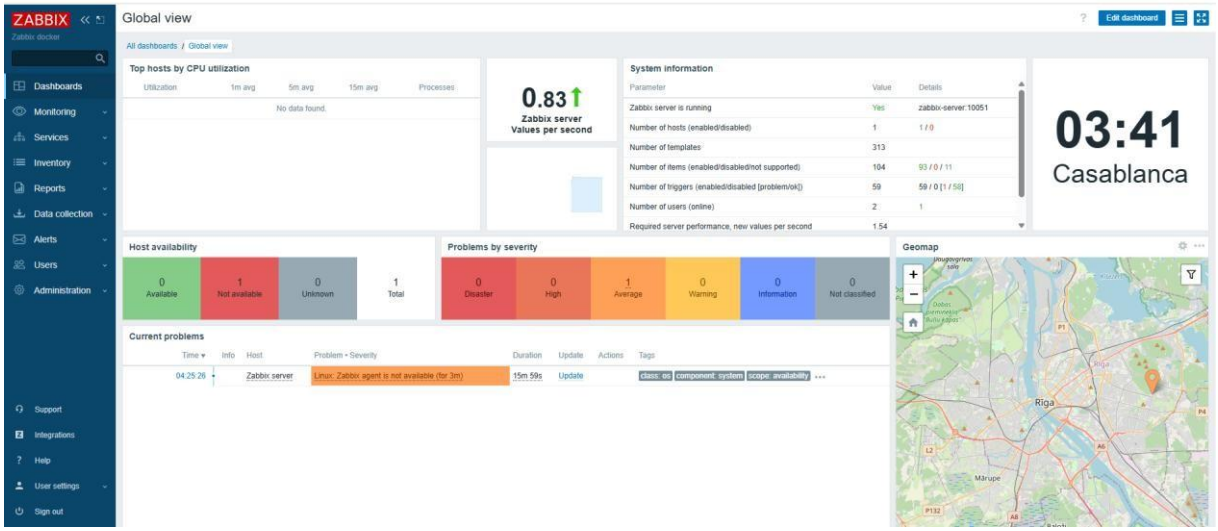
2.3.4 Tableau de bord et Etat du Système :

Après connexion, nous accédons à la **Global View**. Cette interface confirme la réussite totale du projet :

- **Disponibilité du Serveur :** L'indicateur "Zabbix server is running" affiche **Yes**, prouvant que le conteneur zabbix-server communique correctement avec l'interface Web et la base de données MySQL.

- **Localisation et Heure :** Le fuseau horaire Africa/Casablanca configuré dans le fichier .env est correctement appliqué (03:41).

- **Performance :** Le système traite déjà des valeurs par seconde (0.83), indiquant que l'auto-supervision du serveur est active.



3 Configuration des Clients (Agents Zabbix)

L'agent Zabbix est un service léger installé sur les machines cibles (hôtes) pour collecter des données de performance (CPU, RAM, Disque) et les envoyer au serveur.

3.1 Installation et configuration de l'agent sur le client Linux :

Le déploiement sur l'instance Linux-client-zabbix s'effectue en téléchargeant les dépôts officiels de l'éditeur pour garantir une version de l'agent synchronisée avec le serveur.

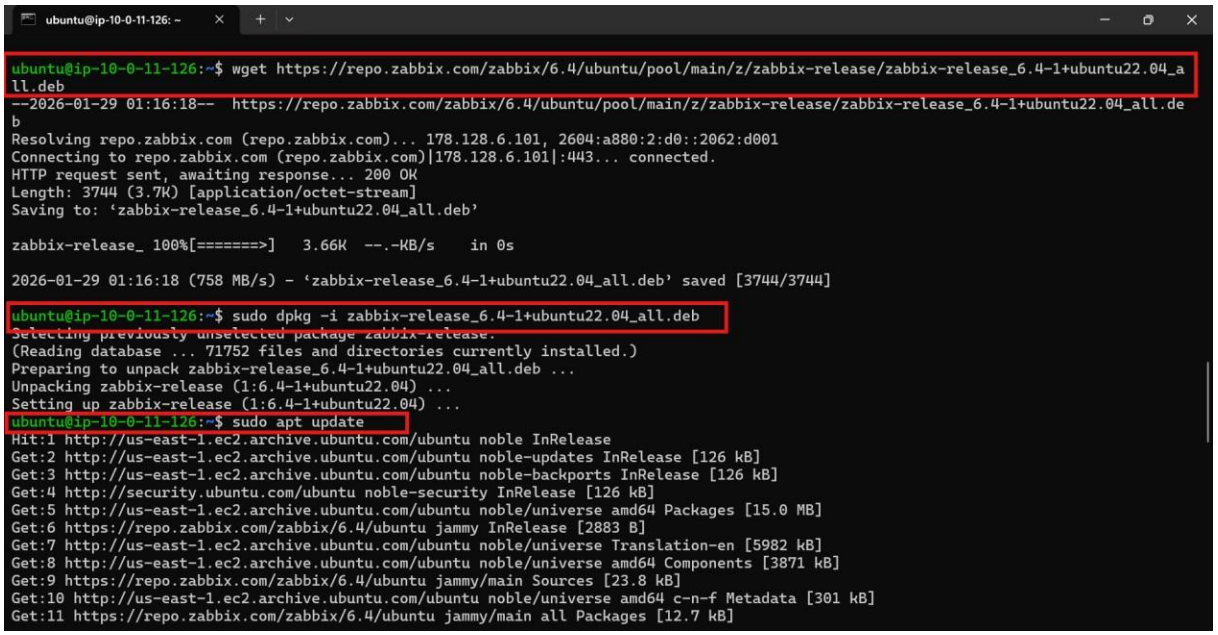
3.1.1 Phase d'installation

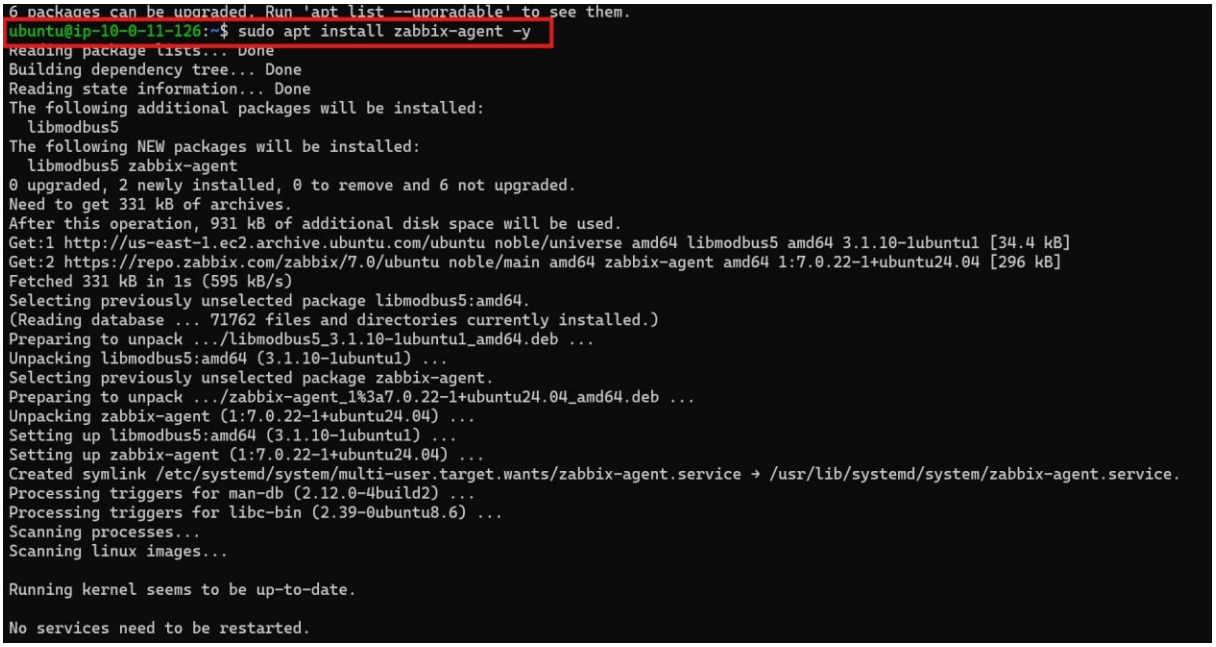
Nous utilisons les commandes suivantes pour préparer l'environnement et installer le service :

```
# Téléchargement du paquet de dépôt Zabbix 6.4
wget  https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.4-1+ubuntu22.04_all.deb

# Installation du dépôt
sudo dpkg -i zabbix-release_6.4-1+ubuntu22.04_all.deb

# Mise à jour de la liste des paquets et installation
sudo apt update
sudo apt install zabbix-agent -y
```





3.1.2 Configuration du fichier zabbix_agentd.conf

Pour établir la liaison avec le serveur distant sur AWS, nous éditons le fichier de configuration principal : sudo nano /etc/zabbix/zabbix_agentd.conf

Nous modifions les variables clés pour autoriser les flux de données :

Server : On renseigne l'adresse IP publique actuelle de l'instance Zabbix-Server (vérifier l'IP dans la console AWS si l'instance a été redémarrée).

ServerActive : On utilise la même IP pour permettre l'envoi de données à l'initiative de l'agent.

Hostname : On définit le nom exact de l'instance cliente (ex: Linux-client-zabbix), ce nom sera utilisé lors de la déclaration dans l'interface Web.

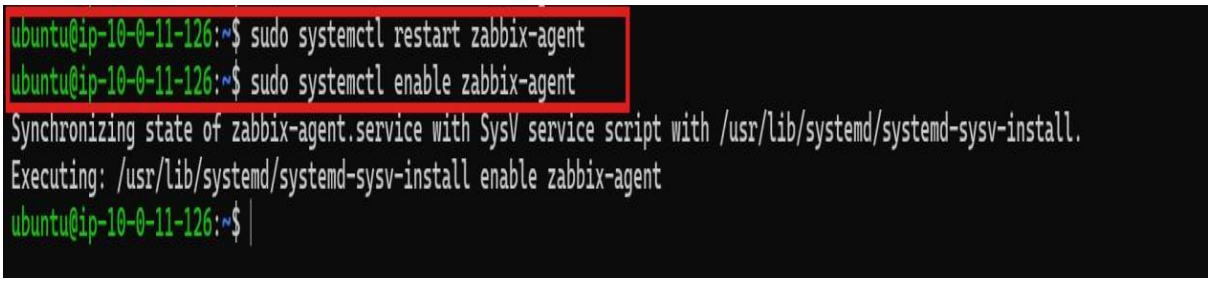
3.1.3 Finalisation et persistance :

Enfin, nous redémarrons le service pour appliquer les changements et l'activons au démarrage du système :

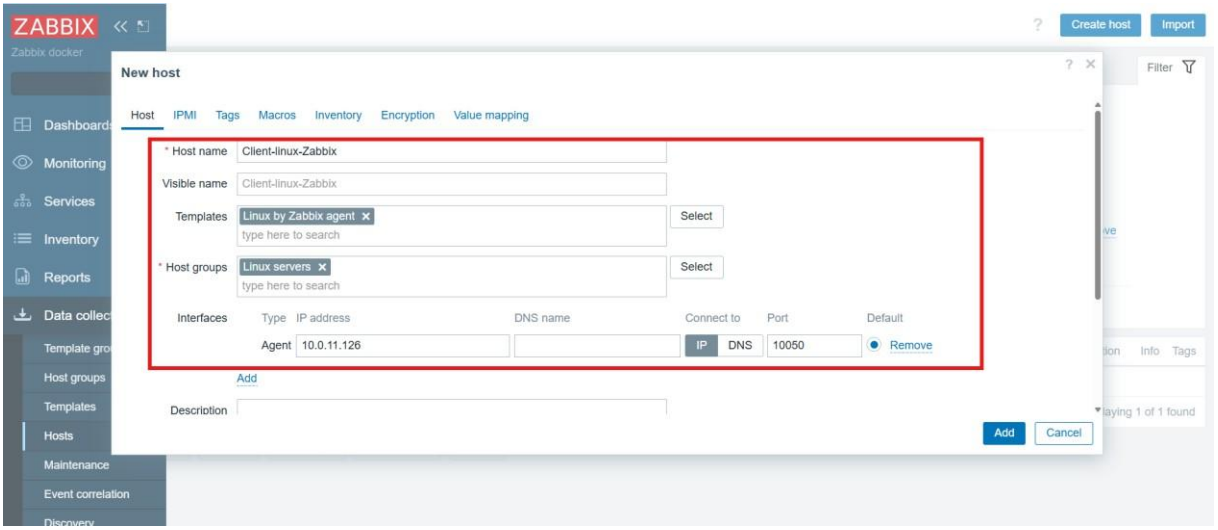
Bash

sudo systemctl restart zabbix-agent

sudo systemctl enable zabbix-agent



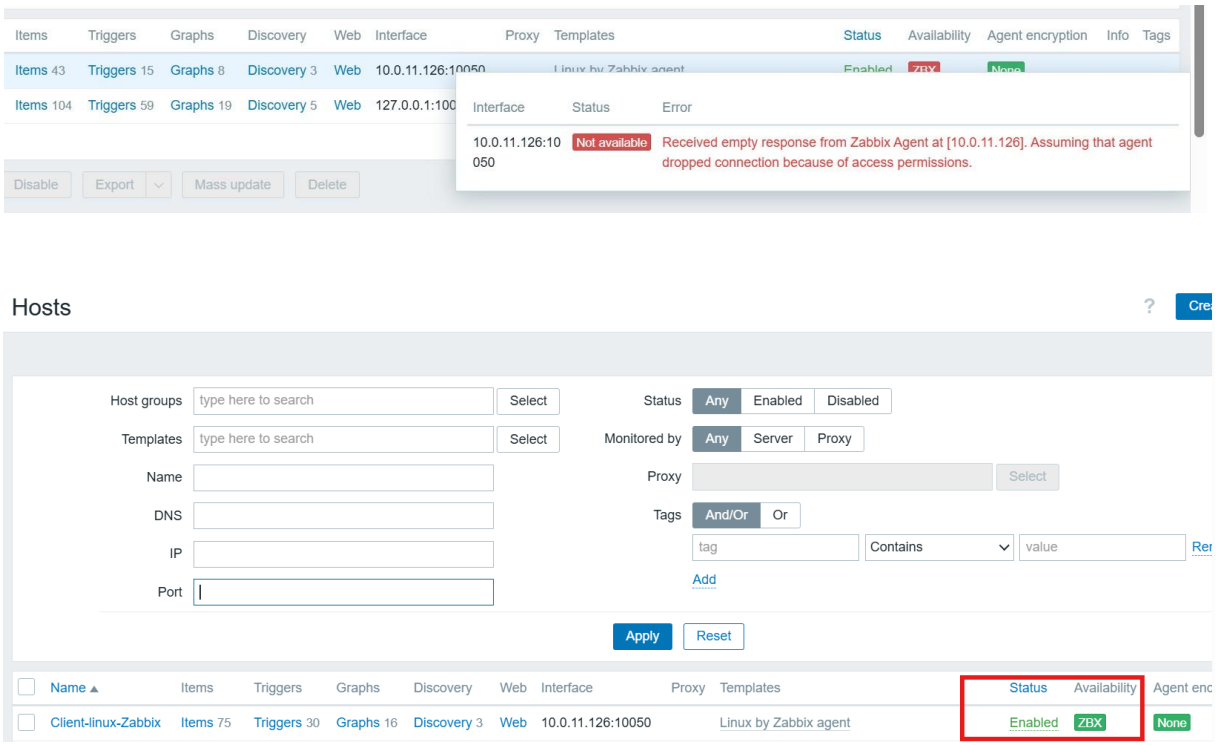
3.1.4 Ajout de l'hôte dans l'interface web Zabbix :



3.1.5 Résolution de l'erreur d'accès (Access Permissions)

Malgré l'ouverture des ports 10050/10051 dans le Security Group, l'hôte affichait une erreur de permissions.

Diagnostic : L'agent rejetait les requêtes car il était configuré pour accepter l'IP publique du serveur, alors que les flux au sein du VPC AWS transitent par les interfaces privées.



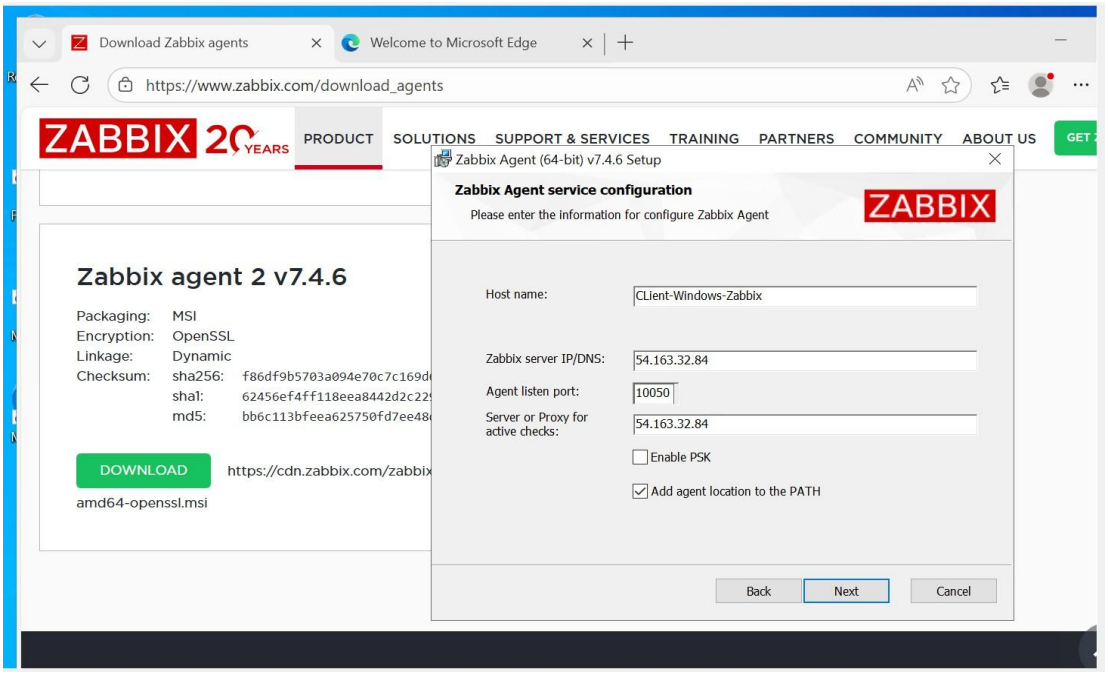
3.2 Extension de la supervision :

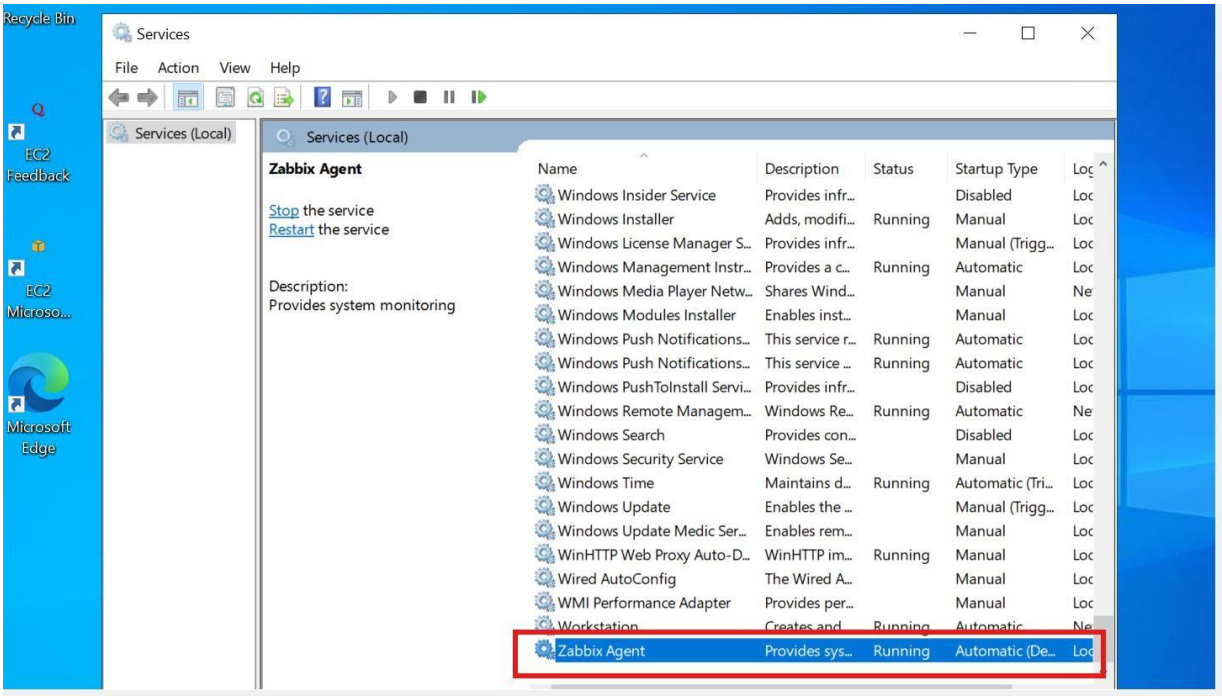
3.2.1 Déploiement de l'agent MSI :

L'installation sur l'instance CLient-Windows-Zabbix (10.0.2.40) a été réalisée via le package officiel MSI. Contrairement à l'installation Linux en ligne de commande, Windows utilise un assistant graphique pour configurer les paramètres de communication.

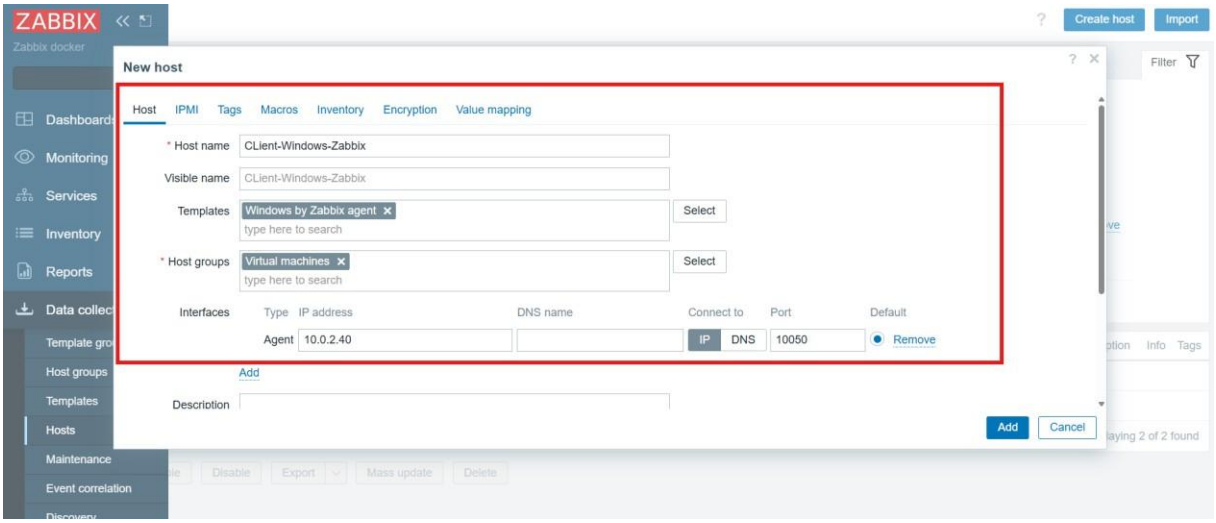
Configuration appliquée lors de l'installation :

- Host name : CLient-Windows-Zabbix (doit correspondre strictement au nom dans l'interface Web).
- Zabbix server IP : 54.163.32.84 (IP publique du serveur pour permettre la liaison via le réseau AWS).
- Options de sécurité : La case "Enable PSK" a été laissée décochée pour simplifier la phase de test initiale et éviter les erreurs de chiffrement de clés.
- Environnement : L'option "Add agent location to the PATH" a été activée pour permettre l'administration de l'agent en ligne de commande (PowerShell).


Download Zabbix agents
Welcome to Microsoft Edge
https://www.zabbix.com/download_agents
ZABBIX 20 YEARS
PRODUCT
SOLUTIONS
SUPPORT & SERVICES
TRAINING
PARTNERS
COMMUNITY
ABOUT US
Zabbix agent 2 v7.4.6
Packaging: MSI
Encryption: OpenSSL
Linkage: Dynamic
Checksum: sha256:
sha1:
md5:
DOWNLOAD
https://cdn.zabbix.com/zabbix
amd64-openssl.msi
Zabbix Agent (64-bit) v7.4.6 Setup
Zabbix Agent service configuration
Please enter the information for configure Zabbix Agent
ZABBIX
Host name:
CLient-Windows-Zabbix
Zabbix server IP/DNS:
54.163.32.84
Agent listen port:
10050
Server or Proxy for active checks:
54.163.32.84
Enable PSK
Add agent location to the PATH
Back
Next
Cancel

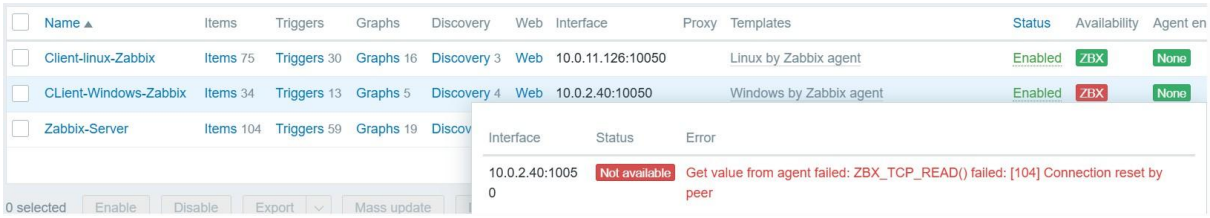

Recycle Bin
EC2 Feedback
EC2 Microso...
Microsoft Edge
Services
File Action View Help
Services (Local)
Zabbix Agent
Stop the service
Restart the service
Description:
Provides system monitoring
Name
Description
Status
Startup Type
Windows Insider Service
Windows Installer
Windows License Manager S...
Windows Management Instr...
Windows Media Player Netw...
Windows Modules Installer
Windows Push Notifications...
Windows Push Notifications...
Windows PushToInstall Servi...
Windows Remote Managem...
Windows Search
Windows Security Service
Windows Time
Windows Update
Windows Update Medic Ser...
WinHTTP Web Proxy Auto-D...
Wired AutoConfig
WMI Performance Adapter
Workstation
Zabbix Agent
Provides sys...
Running
Automatic (De...

3.2.2 Configuration du flux réseau et du host :



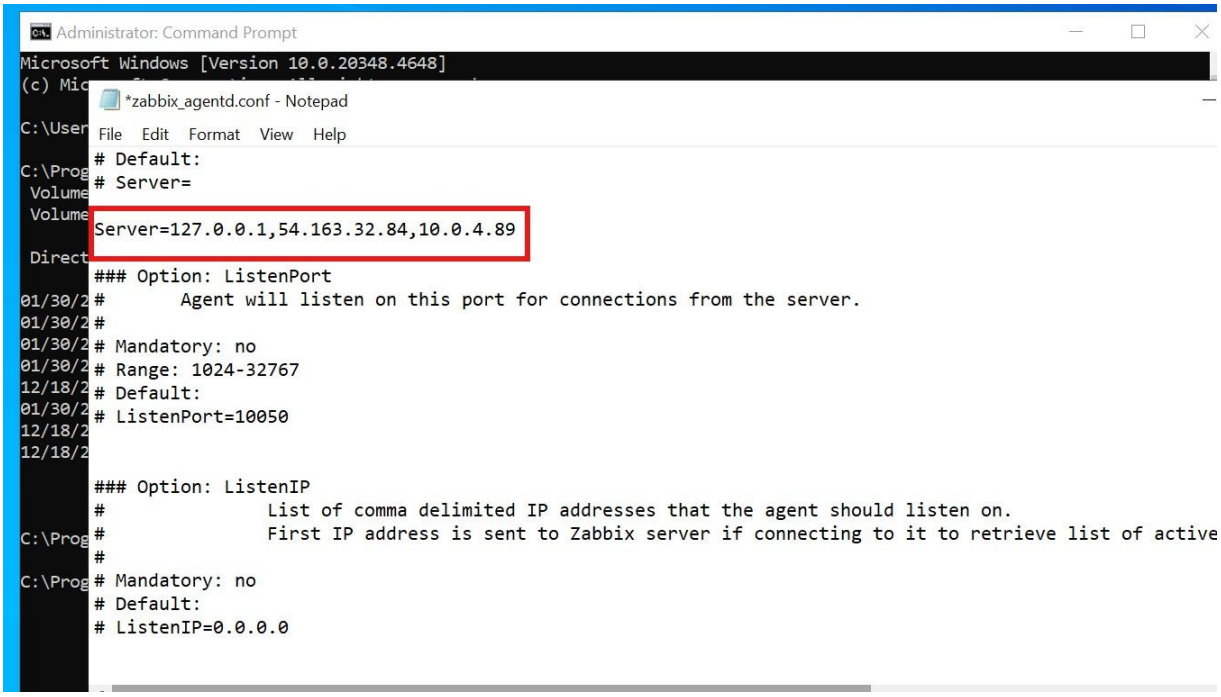
3.2.3 Diagnostic : Résolution de l'erreur de connexion :

Dès l'activation, une erreur de disponibilité ZBX est apparue : Connection reset by peer.



Ce problème survient car l'agent n'autorisait pas l'IP privée du serveur pour les requêtes de collecte de données. Pour y remédier, le fichier zabbix_agentd.conf a été modifié manuellement.

- **Action** : Ajout de l'IP privée 10.0.4.89 dans la directive Server.



3.2.4 Validation du service :

Après modification, le service "Zabbix Agent" a été redémarré dans le gestionnaire des services Windows pour appliquer les paramètres.

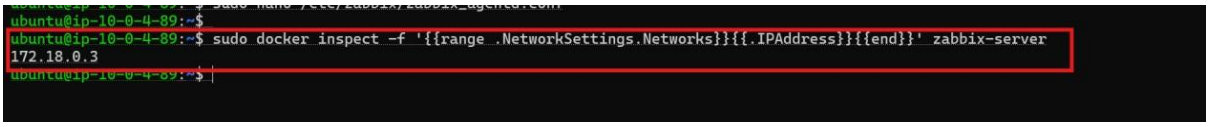
3.3 Auto-supervision du serveur Zabbix :

3.3.1 Identification réseau du conteneur

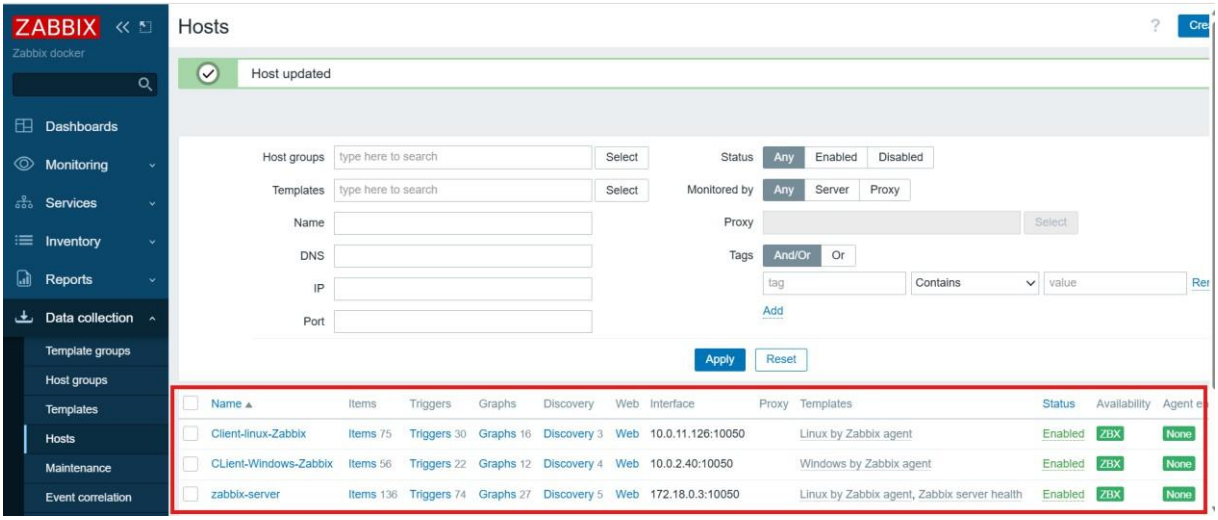
Le serveur tournant sous Docker, il a fallu identifier son adresse IP interne pour permettre l'auto-supervision sans conflit de permissions.

Une première tentative via docker inspect a échoué en raison de privilèges insuffisants et d'une erreur de syntaxe sur le nom de l'objet. Après avoir vérifié le nom exact via sudo docker ps, l'IP correcte a été extraite.

- **Commande** : sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' zabbix-server.
- **IP identifiée** : 172.18.0.3.



3.3.2 État final de la supervision



4 Monitoring et Tableaux de bord :

4.1 Centralisation des métriques (Global Infrastructure Monitoring)

Nous avons conçu un tableau de bord unique pour visualiser les performances critiques sans changer de page.

4.1.1 Graphique de charge CPU

- **Widget** : Type "Graph".
- **Pattern** : Utilisation du motif CPU utilization.
- **Multi-hôtes** : Sélection simultanée de Client-linux-Zabbix et CLient-Windows-Zabbix pour comparer les performances en temps réel.

4.1.2 Graphique d'utilisation RAM

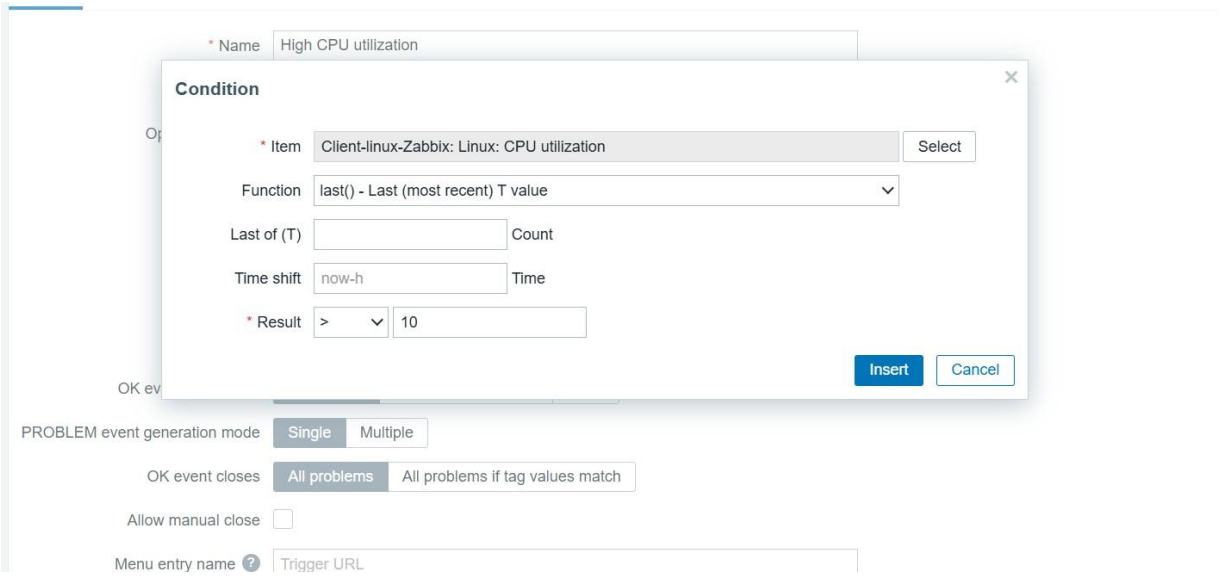
Un second widget a été ajouté pour surveiller la mémoire vive (item Memory utilization). Ce graphique permet d'anticiper les saturations sur les instances EC2.

4.2 Alertes Proactives et Synthèse Visuelle :

4.2.1 Mise en place d'un Trigger (Alerte Proactive)

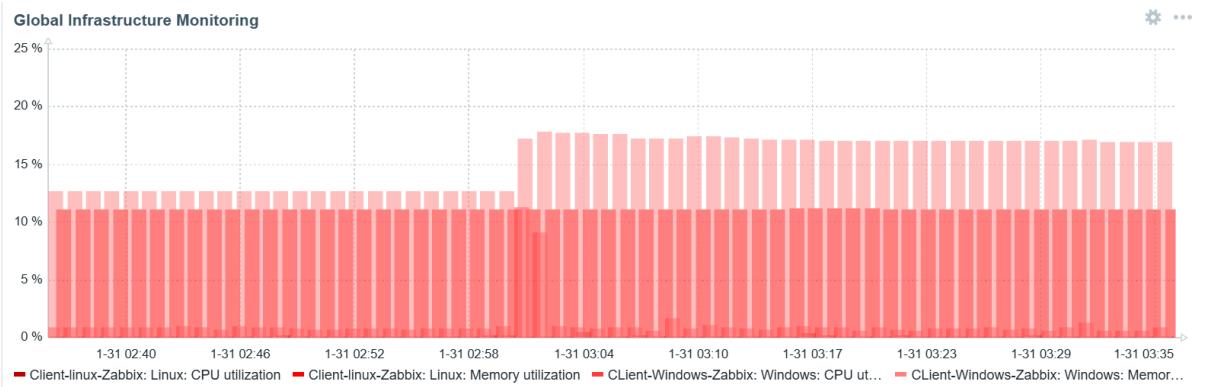
Pour démontrer la réactivité de Zabbix, un déclencheur a été configuré pour l'hôte Linux.

- **Seuil de test** : Configuration de la fonction last() > 10 pour l'item CPU.
- **Sévérité** : "High" (Rouge) pour garantir une visibilité immédiate en cas d'anomalie.



4.2.2 Synthèse Visuelle du Dashboard Consolidé

Le résultat final offre une vue corrélée sur l'état de santé des infrastructures Linux et Windows. Cette interface permet à un administrateur système de détecter en un coup d'œil un problème de performance ou une rupture de service sur l'ensemble du VPC AWS.



Le graphique présente une vue corrélée de l'utilisation des ressources sur une période d'environ une heure (de 02:40 à 03:35).

- Comparaison des environnements (Linux vs Windows) :
    - Utilisation CPU : Les courbes de charge CPU (en bas du graphique) restent très faibles et stables, généralement en dessous de 1% pour les deux systèmes. Cela indique que les instances EC2 sont actuellement au repos (idle).
    - Utilisation Mémoire (RAM) : On observe une différence nette entre les deux systèmes. L'instance Client-Windows-Zabbix (en rose clair) présente une consommation RAM supérieure (environ 17%) par rapport à l'instance Client-linux-Zabbix (en rouge foncé), qui se stabilise autour de 11%.
- Observation d'un pic de charge (vers 03:00) :
    - À partir de 03:00, on note une augmentation soudaine de l'utilisation mémoire sur l'hôte Windows, passant de 13% à 17%.
    - Cette hausse peut être interprétée comme le démarrage d'un service système ou la mise en cache de données par l'OS Windows Server, démontrant la capacité de Zabbix à détecter des variations de quelques points de pourcentage en temps réel.

Conclusion

Le projet de déploiement d'une solution de monitoring avec Zabbix a permis de valider la mise en place d'une architecture de supervision fiable et performante, capable de surveiller un parc informatique hybride et distribué au sein du Cloud **Amazon Web Services**.
La réalisation de ce travail pratique a mobilisé un ensemble de compétences techniques en administration système et réseau. Elle a notamment permis de consolider la maîtrise de l'écosystème Cloud à travers la configuration des instances EC2, la gestion des adresses IP publiques et privées ainsi que la sécurisation des flux via les Security Groups AWS. Le projet a également nécessité des compétences en ingénierie Docker, avec le déploiement du serveur Zabbix dans un environnement conteneurisé, impliquant une compréhension approfondie de l'isolation réseau et des mécanismes de communication entre l'hôte et ses conteneurs. Enfin, l'installation et la configuration des agents sur des systèmes d'exploitation différents, notamment Linux Ubuntu et Windows Server, ont permis d'assurer une collecte de données homogène malgré l'hétérogénéité des environnements.
Le succès du projet repose également sur la résolution de plusieurs défis techniques rencontrés lors de la phase d'implémentation. L'instabilité liée à l'adressage IP dynamique des conteneurs Docker (changement d'adresse de 172.18.0.3 à 172.18.0.4) a été corrigée par la mise en place d'une configuration basée sur l'adresse locale 127.0.0.1 ainsi que par l'autorisation de segments réseau spécifiques (172.16.0.0/12), garantissant ainsi la stabilité de l'auto-supervision. Par ailleurs, les erreurs de type « Connection refused » ou « Reset by peer » ont conduit à un travail approfondi sur la configuration des fichiers zabbix_agentd.conf et sur l'ajustement des pare-feu locaux, notamment Windows Firewall, afin d'assurer une communication sécurisée et fonctionnelle entre le serveur et les agents.
En conclusion, ce projet démontre qu'une supervision efficace constitue un élément fondamental d'une infrastructure Cloud bien administrée. Elle permet de transformer des métriques techniques brutes en un véritable outil d'aide à la décision pour l'administrateur système, contribuant ainsi à la fiabilité, à la performance et à la sécurité globale de l'environnement informatique.