ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Filière : **II-BDCC**

Ingénierie Informatique : Big Data et Cloud Computing

PROJET DE FIN DE MODULE

CYBER OPS

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)

Année Universitaire : 2025/2026

Réalisé par :	Encadré par :
Mohamed BOULAALAM	Pr. Azeddine KHIAT

Tableau de matière :

Table des figures :

I. Introduction

1.1 Contexte du projet

Dans le paysage technologique actuel, la disponibilité et la performance des infrastructures informatiques sont des piliers critiques pour toute organisation. Dans le cadre de la dernière année du cycle d'ingénieur, ce projet porte sur la **supervision centralisée** au sein d'environnements Cloud.

Le défi principal réside dans la gestion d'un parc hybride, nécessitant une solution capable d'unifier le monitoring de systèmes d'exploitation hétérogènes (Linux et Windows). Pour répondre à ce besoin, nous avons choisi de déployer une infrastructure robuste sur **Amazon Web Services** (**AWS**) en utilisant le logiciel de supervision **Zabbix**. L'approche technologique repose sur la conteneurisation, garantissant une portabilité et une agilité accrues lors du déploiement des services.

1.2 Objectifs du projet

L'objectif majeur de ce travail est de mettre en œuvre une plateforme de supervision opérationnelle en respectant les exigences techniques suivantes :

- **Déploiement Cloud** : Conception d'un réseau privé virtuel (VPC) sécurisé et provisionnement d'instances EC2 adaptées.
- **Conteneurisation** : Installation et orchestration de la stack Zabbix (Serveur, Interface Web, Base de données) via Docker et Docker-Compose.
- **Interopérabilité Hybride** : Configuration de la remontée des métriques de performance depuis un serveur Ubuntu Linux et un serveur Windows Server 2022.
- **Pilotage et Alerting** : Création de tableaux de bord personnalisés et configuration de déclencheurs (triggers) pour une détection proactive des incidents.

1.3 Outils et Technologies utilisés

La réussite de cette infrastructure repose sur la synergie de trois technologies clés :

- **AWS** (**Amazon Web Services**) : Utilisation de l'environnement *Learner Lab* pour le déploiement des ressources réseau (VPC, Subnets, Internet Gateway) et des ressources de calcul (Instances t3.medium et t3.large).
- **Docker & Docker-Compose** : Choisi pour simplifier le déploiement du serveur Zabbix. Cette technologie permet d'isoler les dépendances système et d'assurer une gestion simplifiée du cycle de vie des applications.
- **Zabbix** (**Version 6.4**) : Solution open-source de classe entreprise utilisée pour la collecte, l'analyse et la visualisation des données de monitoring. L'utilisation du **Zabbix Agent 2** a été privilégiée pour ses performances accrues sur les hôtes surveillés.

Dépôt GitHub

Pour une transparence totale et une valorisation technique du projet, l'ensemble des scripts de déploiement, des fichiers de configuration Docker et de la documentation associée est disponible sur le dépôt public suivant :

Lien du dépôt : https://github.com/MohamedBOULAALAM/Zabbix_AWS_Monitoring_Infrastructure.git

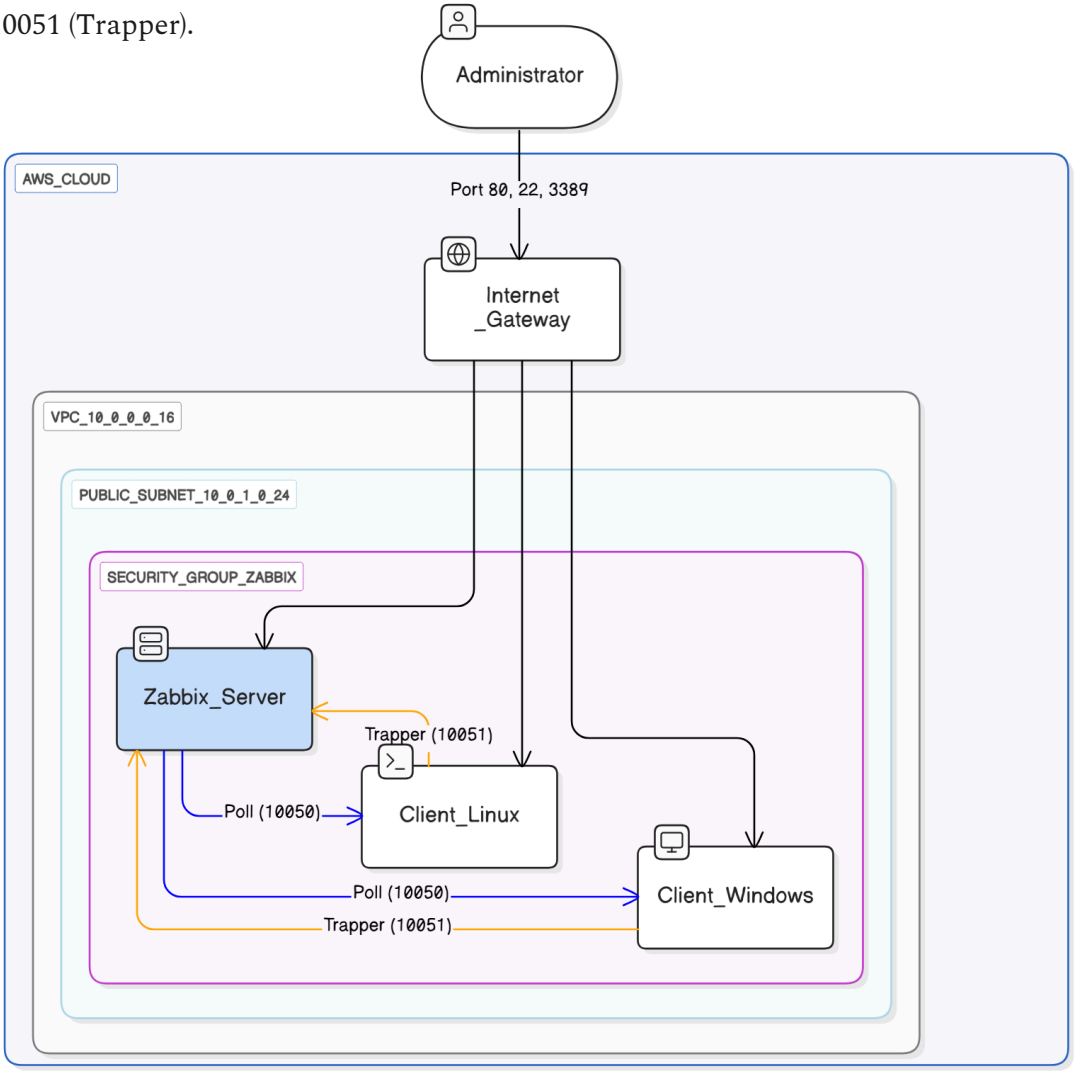
II. Architecture Réseau

Cette section détaille la conception et la mise en œuvre de l'infrastructure réseau sur AWS, constituant le socle de communication sécurisé pour notre système de supervision centralisée.

2.1 Schéma de l'Architecture

L'architecture repose sur une infrastructure Cloud isolée au sein d'un Virtual Private Cloud (VPC). Le schéma ci-dessous illustre l'organisation des composants et les flux de communication prévus :

- **Administrateur** : Accès via les ports 80 (Web), 22 (SSH) et 3389 (RDP).
- **Composants Réseau** : Une Internet Gateway pour l'accès externe et un sous-réseau public.
- **Instances** : Le serveur Zabbix centralise les données provenant des clients Linux et Windows via les agents.
- **Flux de communication** : Communication bidirectionnelle via les ports 10050 (Poll) et 10051 (Trapper).



2.2 Configuration du Virtual Private Cloud (VPC)

2.2.1 Création du VPC et du Sous-réseau

Pour garantir l'isolation de notre parc hybride, nous avons créé un VPC dédié nommé `VPC-Zabbix-BOULAALAM` avec un bloc d'adressage IPv4 `10.0.0.0/16`.

Au sein de ce VPC, un sous-réseau public unique a été déployé pour simplifier l'accès aux instances durant cette phase de projet :

- **Nom** : `Subnet-Public-Zabbix-BOULAALAM`.
- **CIDR** : `10.0.1.0/24`.
- **Zone de disponibilité** : `us-east-1a`.
- **Paramètre critique** : L'auto-assignation d'adresses IPv4 publiques a été activée pour permettre l'administration à distance.

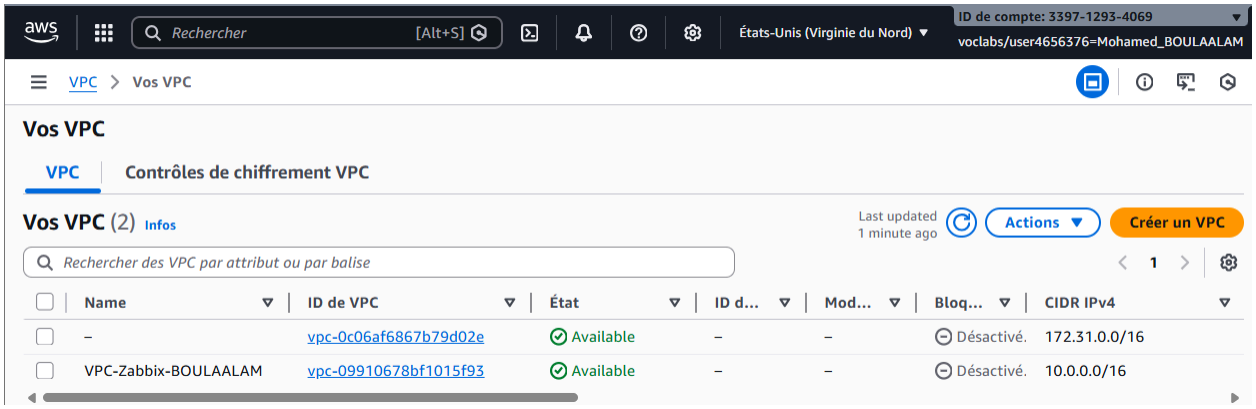


Figure 1 : Création du VPC avec le tag identifiant l'étudiant.

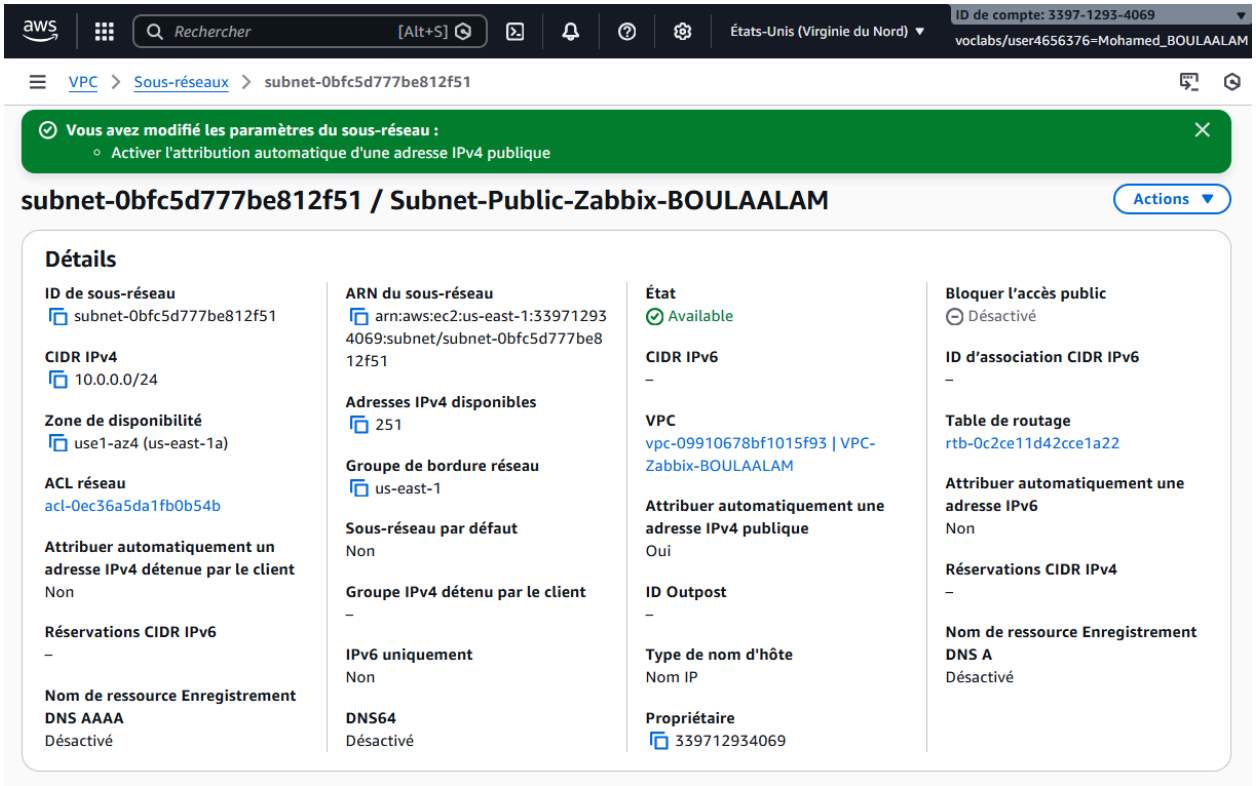


Figure 2 : Configuration du sous-réseau et activation de l'IP publique.

2.2.2 Connectivité Internet et Routage

La connectivité vers l'extérieur est assurée par une passerelle Internet (Internet Gateway) nommée `IGW-Zabbix-BOULAALAM`, rattachée au VPC.

Pour diriger le trafic, une table de routage personnalisée (`RT-Public-Zabbix-BOULAALAM`) a été configurée:

- Une route locale pour le trafic interne au VPC (`10.0.0.0/16`).
- Une route par défaut (`0.0.0.0/0`) pointant vers l'Internet Gateway pour permettre aux instances de télécharger les paquets nécessaires (Docker, Agents).

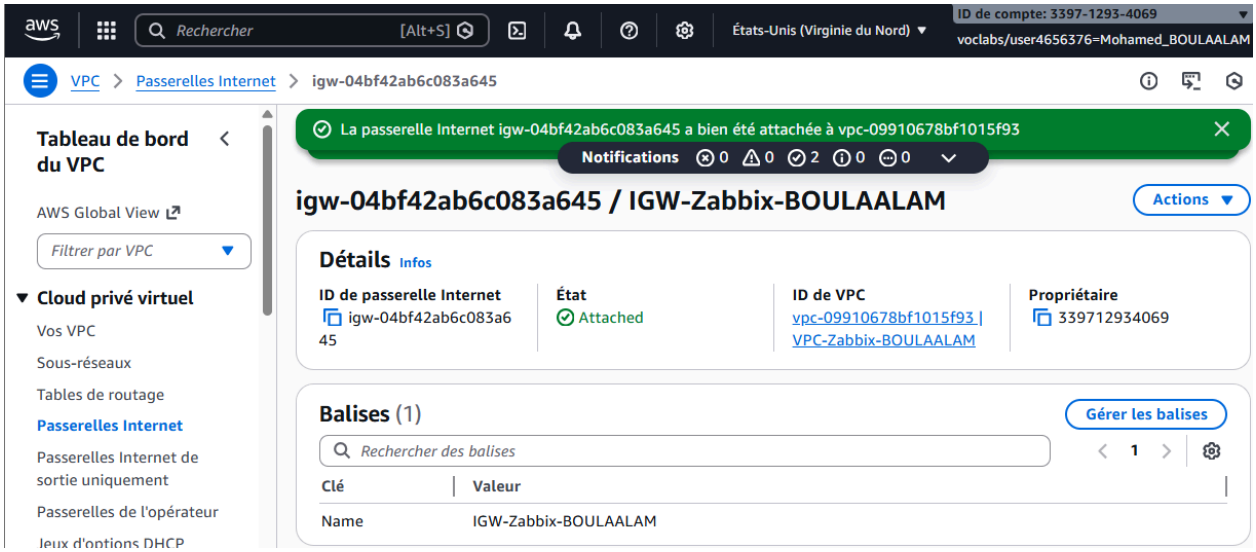


Figure 3 : Internet Gateway attachée avec succès au VPC.

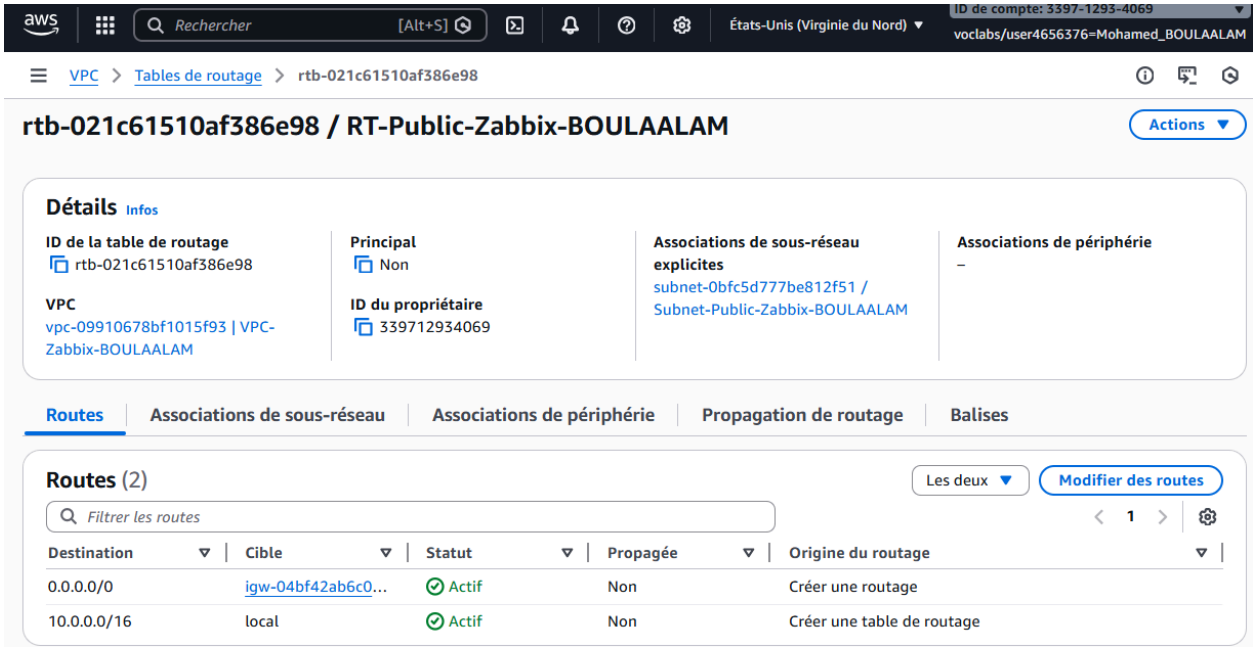


Figure 4 : Table de routage configurée avec la route vers l'Internet Gateway.

2.3 Configuration des Security Groups (Pare-feu)

La sécurité des instances est gérée par des groupes de sécurité agissant comme des pare-feu virtuels au niveau de l'instance.

2.3.1 Security Group pour le Serveur Zabbix

Le groupe `SG-Zabbix-Server-BOULAALAM` est configuré avec les règles entrantes suivantes:

- **SSH (22)** : Autorisé uniquement depuis l'IP de l'administrateur pour la gestion du serveur.
- **HTTP (80)** : Autorisé depuis n'importe quelle source (`0.0.0.0/0`) pour l'accès à l'interface Web Zabbix.
- **Zabbix Trapper (10051)** : Autorisé pour tout le réseau interne (`10.0.0.0/16`) afin de recevoir les données actives envoyées par les agents.

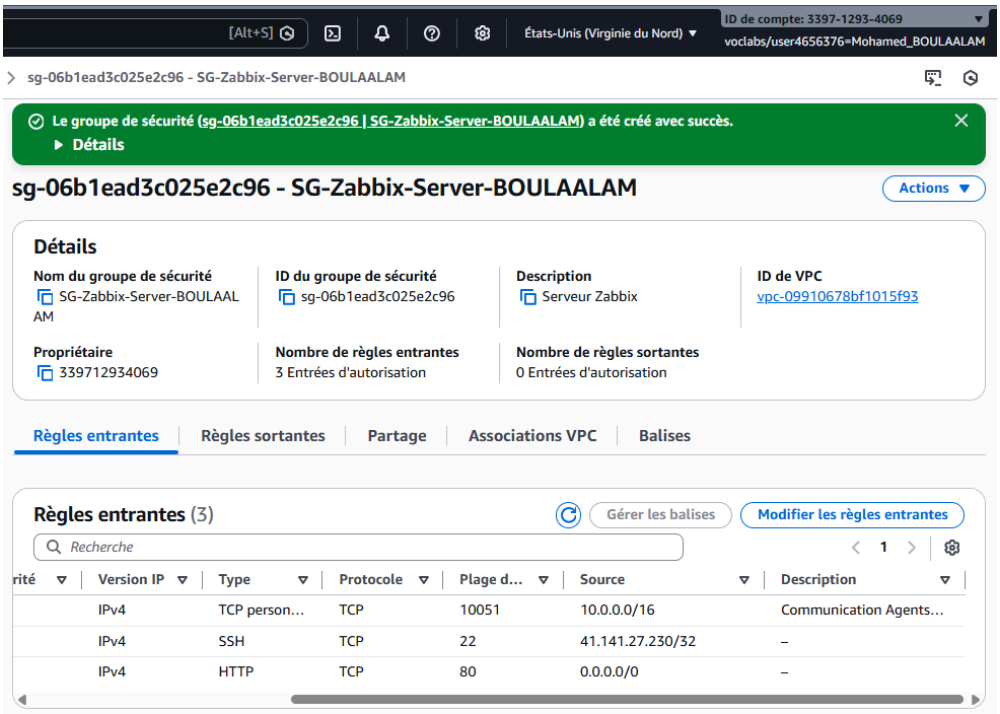


Figure 5: Détails des règles entrantes pour le groupe de sécurité du serveur.

2.3.2 Security Group pour les Clients (Linux & Windows)

Le groupe `SG-Zabbix-Clients-BOULAALAM` protège les hôtes surveillés:

- **SSH** (**22**) et **RDP** (**3389**) : Accès restreint à l'IP de l'administrateur pour la maintenance des clients.
- **Zabbix Agent** (**10050**) : Cette règle cruciale n'autorise que le groupe de sécurité du serveur Zabbix (`sg-06b1ead3c025e2c96`) à interroger les agents sur les clients, garantissant ainsi que personne d'autre ne peut collecter de données sur ces machines.

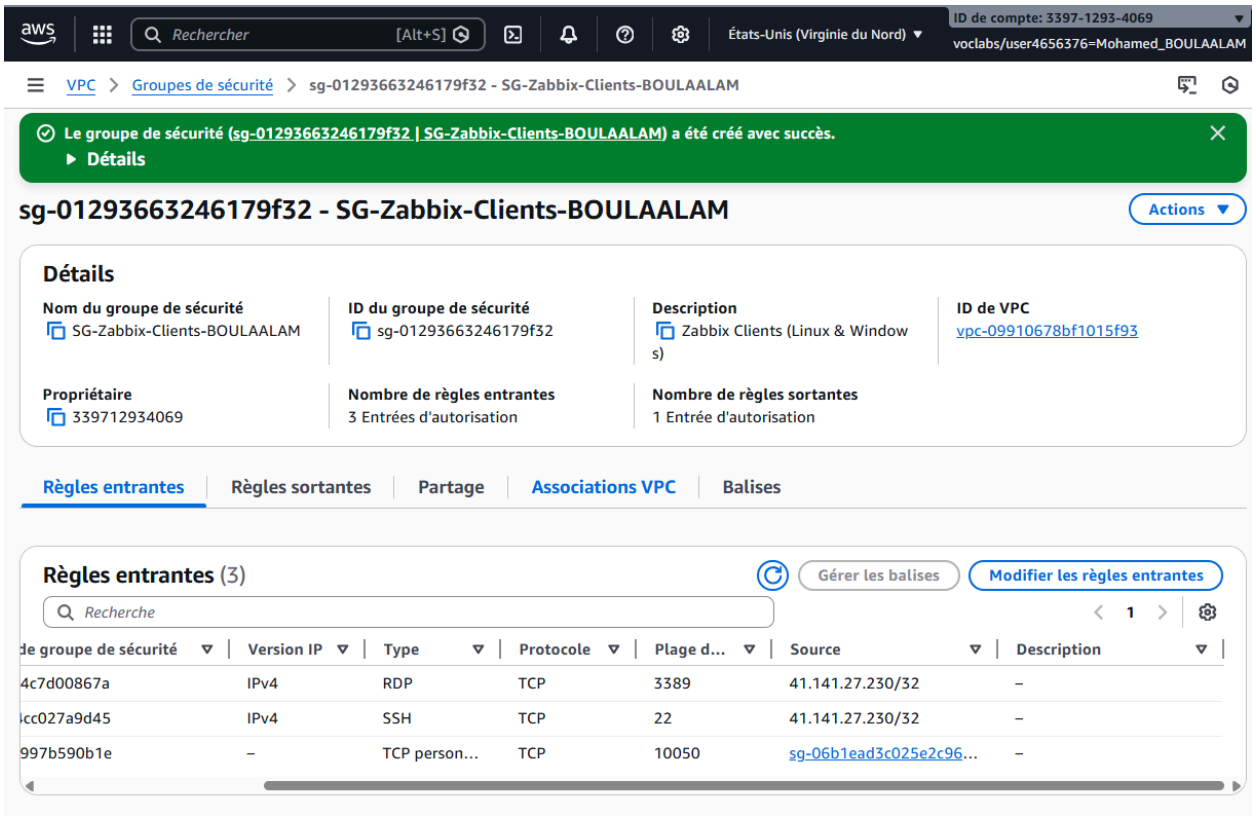


Figure 6 : Configuration du groupe de sécurité pour le parc hybride.

III. Architecture des Instances EC2

Pour simuler un parc informatique hybride représentatif d'un environnement d'entreprise, nous avons déployé trois instances EC2 distinctes sur AWS. Chaque instance remplit un rôle spécifique dans l'écosystème de supervision.

3.1 Présentation des instances déployées

Le tableau suivant récapitule les caractéristiques techniques de chaque machine virtuelle :

Nom de l'instance	**Type d'instance**	**Système d'Exploitation (AMI)**	**Rôle principal**
Zabbix-Server-BOULAALAM	t3.large	Ubuntu Server 24.04 LTS	Hébergement de la stack Docker (Server, DB, Web)
Zabbix-Client-Linux-BOULAALAM	t3.medium	Ubuntu Server 24.04 LTS	Hôte Linux supervisé via Zabbix Agent 2
Zabbix-Client-Windows-BOULAALAM	t3.large	Windows Server 2022 Base	Hôte Windows supervisé via Zabbix Agent 2

3.1.1 Le Serveur Central (Zabbix Server)

L'instance serveur est le cœur de l'infrastructure. Elle exécute le moteur de supervision et la base de données MySQL.

- **Adresse IP Publique** : 13.220.84.79
- **Adresse IP Privée** : 10.0.0.195

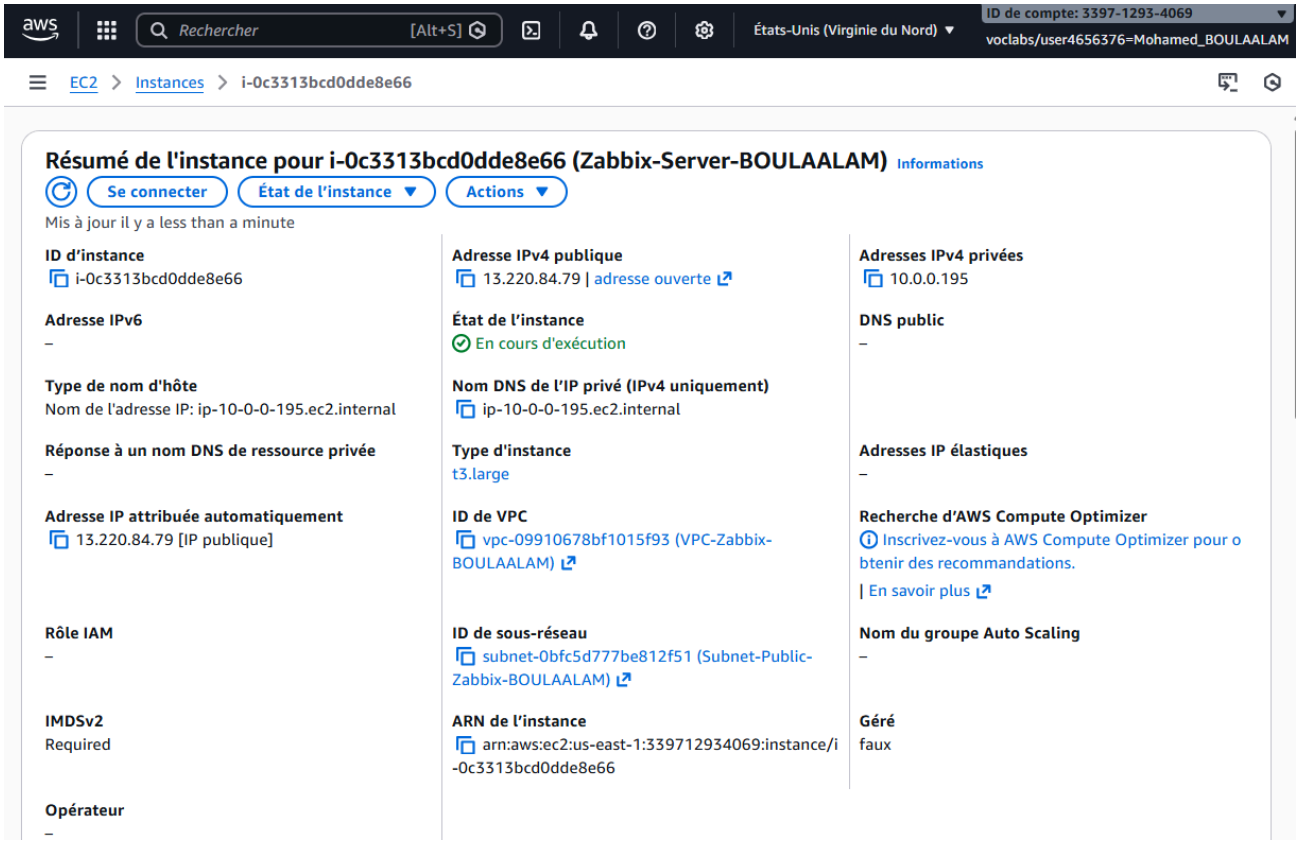


Figure 7 : Détails techniques et résumé de l'instance Serveur Zabbix.

3.1.2 Le Client Linux

Cette instance représente un serveur applicatif sous Ubuntu. Elle communique ses métriques de performance au serveur via le réseau privé d'AWS.

- **Adresse IP Publique** : 13.220.187.148
- **Adresse IP Privée** : 10.0.0.13

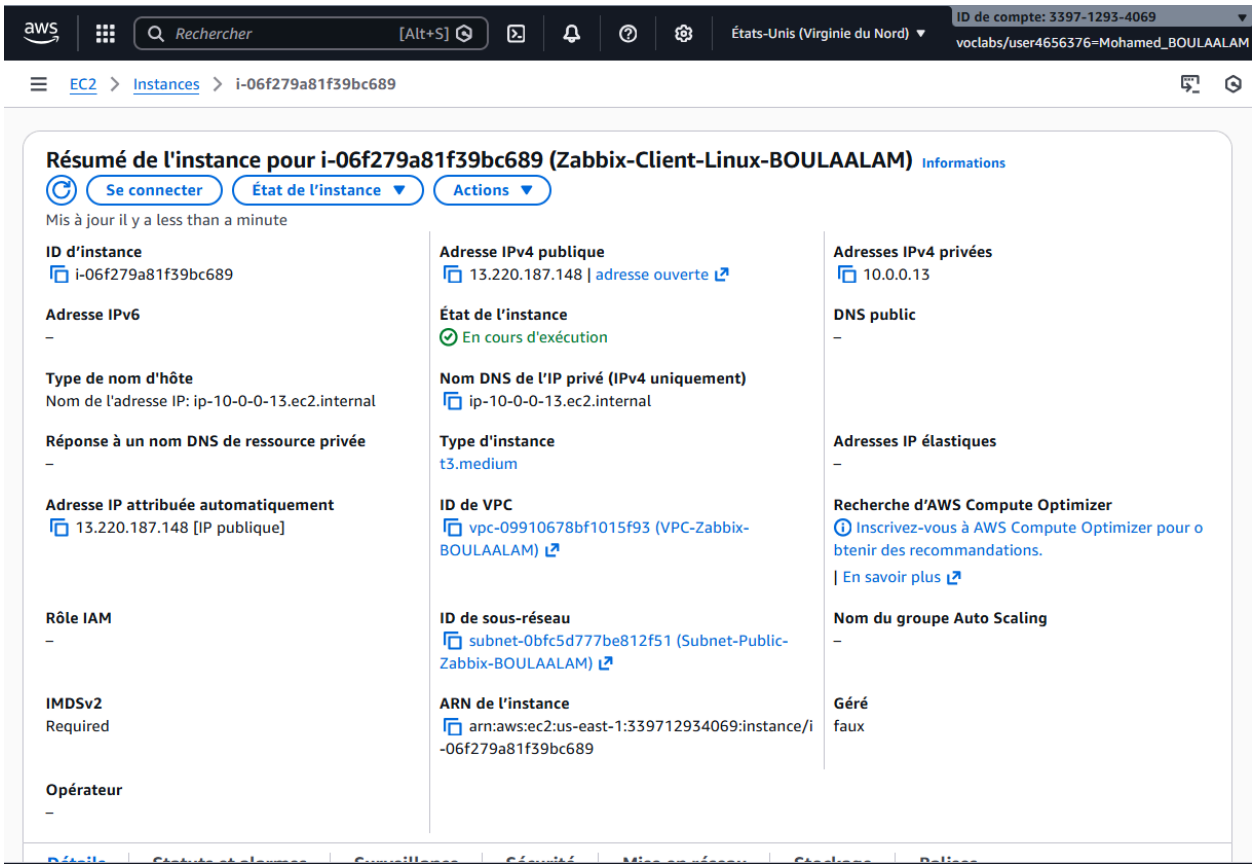


Figure 8 : Détails techniques et résumé de l'instance Client Linux.

3.1.3 Le Client Windows

L'intégration d'une machine Windows permet de valider la capacité de Zabbix à gérer un parc hétérogène.

- **Adresse IP Publique** : 54.91.76.91
- **Adresse IP Privée** : 10.0.0.183

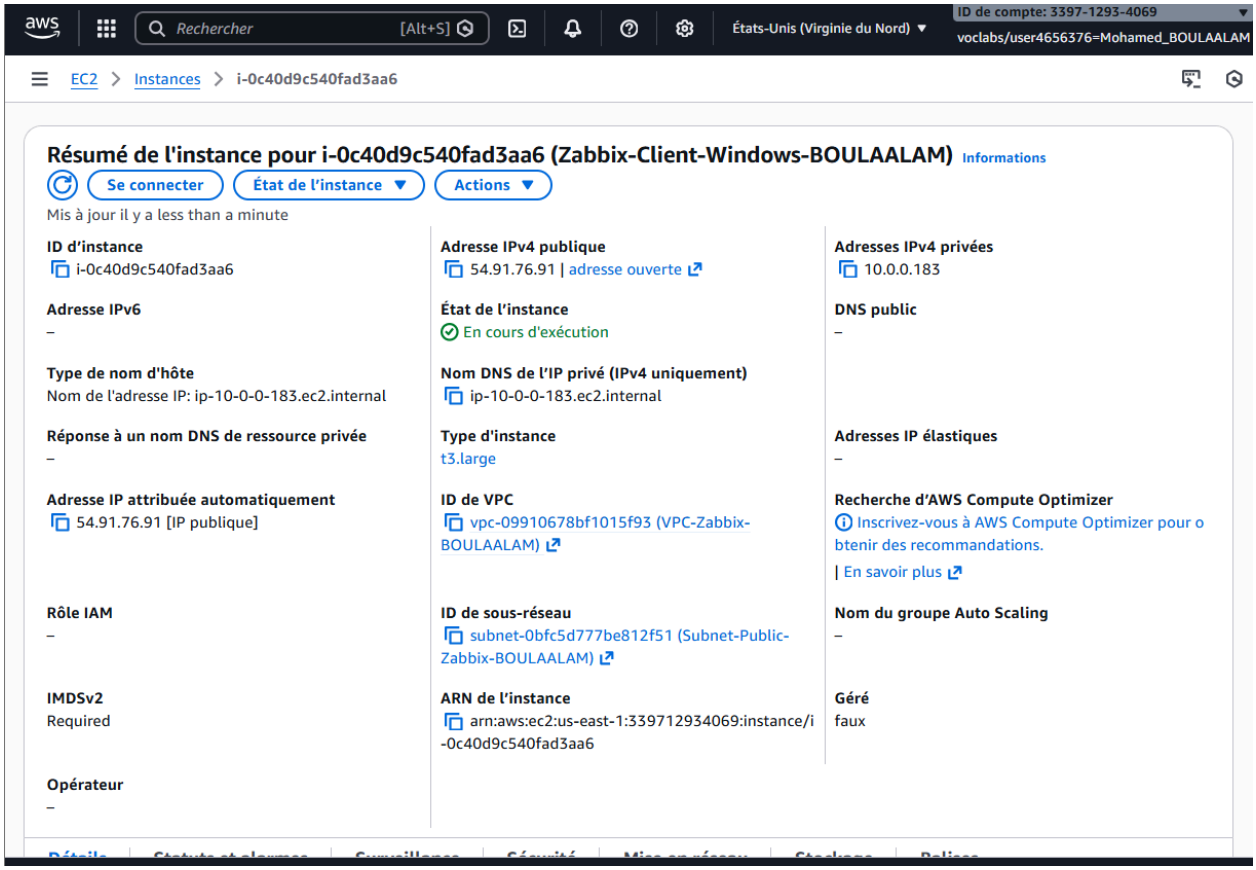


Figure 9 : Détails techniques et résumé de l'instance Client Windows.

3.2 Justification technique des types d'instances

Le choix des familles et tailles d'instances répond à des exigences de performance spécifiques :

1. **Utilisation de la famille t3** : Nous avons opté pour les instances t3 (Burstable Performance) car elles offrent un excellent équilibre entre coût et performance. Elles permettent d'absorber les pics de charge lors du lancement des conteneurs Docker ou de la génération de graphiques complexes dans Zabbix.
2. **Choix du type t3.large pour le Serveur** : Le serveur Zabbix nécessite au minimum 8 Go de RAM pour faire fonctionner simultanément les conteneurs du serveur, de l'interface web (Nginx/PHP) et de la base de données MySQL sans ralentissement.
3. **Choix du type t3.large pour Windows** : Windows Server 2022 possède une empreinte mémoire importante. Un type d'instance inférieur aurait provoqué des lenteurs excessives lors de l'accès via Bureau à Distance (RDP) et de l'installation de l'agent.

4. **Choix du type t3.medium pour le Client Linux** : Le système Ubuntu étant plus léger, un type t3.medium (4 Go RAM) est largement suffisant pour faire fonctionner l'Agent Zabbix 2 tout en maintenant des performances système optimales.

3.3 Vue d'ensemble du parc opérationnel

L'ensemble des instances a été correctement tagué avec le nom de l'étudiant pour garantir l'authenticité du travail réalisé. La console AWS confirme que toutes les machines sont dans l'état "En cours d'exécution" et ont passé les vérifications d'état.

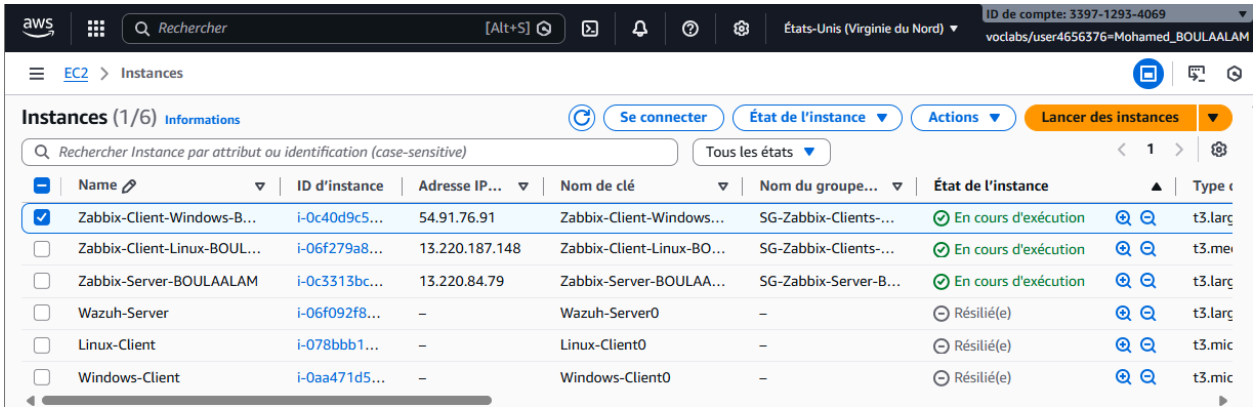


Figure 10 : Tableau de bord EC2 affichant les 3 instances actives avec les tags personnalisés.

IV. Déploiement du Serveur Zabbix

Cette phase détaille la mise en place de la stack de supervision centralisée sur l'instance Ubuntu Server. Le choix de la conteneurisation via Docker a été privilégié pour garantir une isolation optimale des services et faciliter la portabilité de l'infrastructure.

4.1 Préparation et Installation de l'environnement Docker

4.1.1 Accès au serveur et mise à jour

La première étape consiste à établir une connexion sécurisée via le protocole SSH à l'instance **Zabbix-Server-BOULAALAM** en utilisant l'adresse IP publique `13.220.84.79`. Une fois connecté, le système est mis à jour pour assurer la compatibilité des paquets.

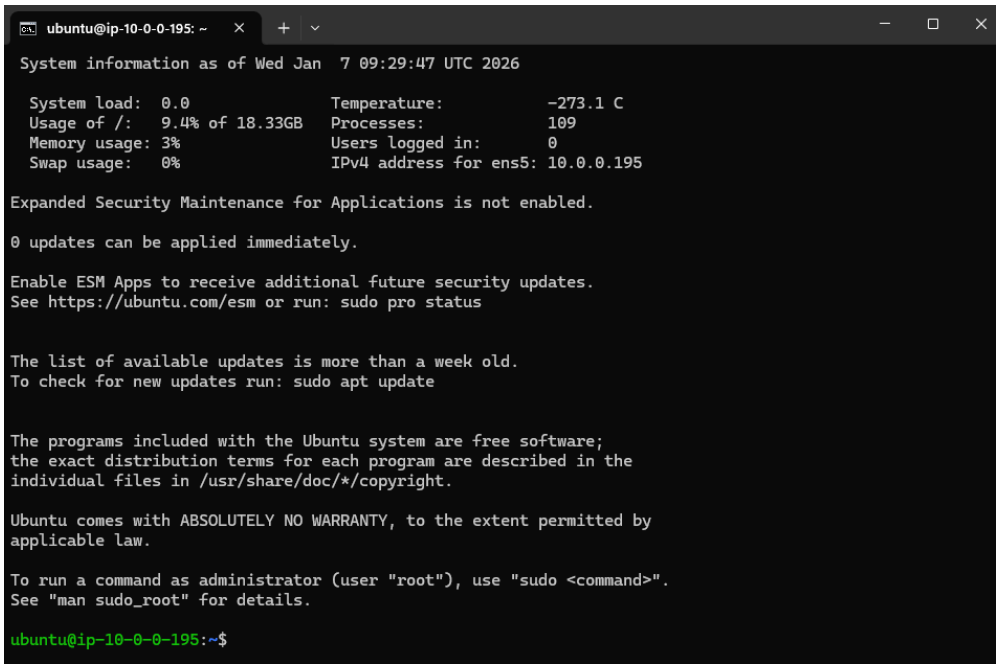


Figure 11 : Connexion SSH réussie au serveur Zabbix avec affichage des informations système.

4.1.2 Installation de Docker Engine et Docker Compose

L'installation de Docker nécessite au préalable la mise en place de plusieurs prérequis, notamment `ca-certificates`, `curl` et `software-properties-common`.

La procédure a suivi les étapes officielles :

1. Ajout de la clé GPG officielle de Docker pour la vérification des paquets.
2. Configuration du dépôt stable Docker pour Ubuntu.
3. Installation du moteur Docker (`docker-ce`) et de l'utilitaire Docker Compose .

Les vérifications de version confirment le succès de l'installation :

- **Docker version** : 28.2.2.
- **Docker Compose version** : v5.0.1.

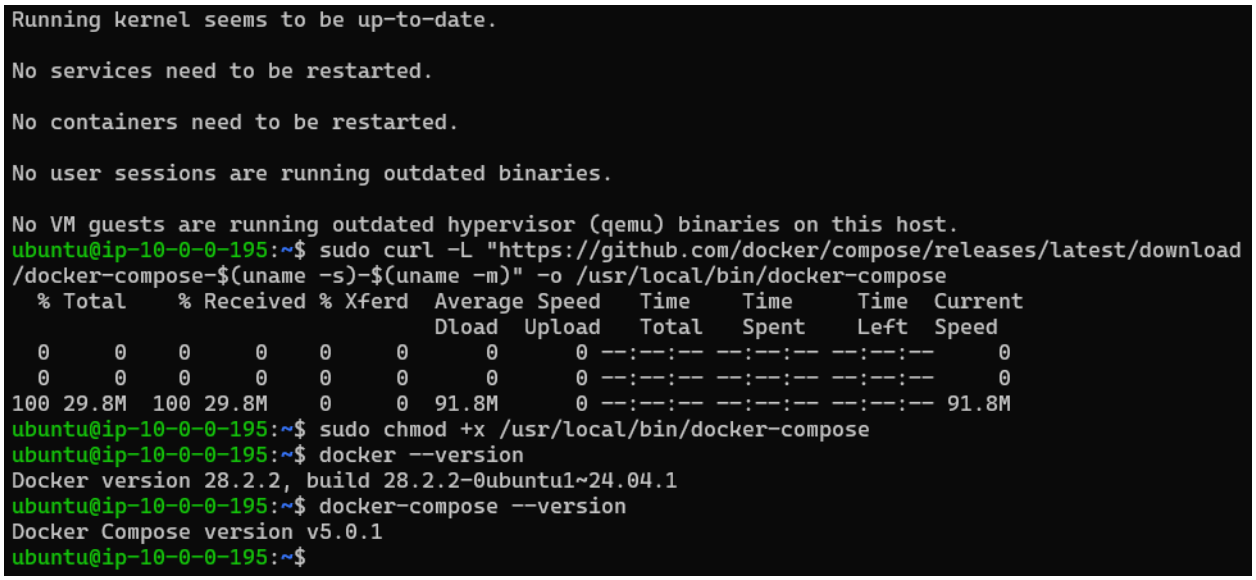


Figure 12 : Vérification de la version de Docker installée.
Figure 13 : Vérification de la version de Docker Compose.

4.2 Configuration de la Stack via Docker-Compose

Pour orchestrer les services, un répertoire dédié nommé `zabbix-stack` a été créé. Nous y avons configuré le fichier `docker-compose.yml` qui définit les trois composants essentiels de notre serveur de monitoring.

4.2.1 Description des services configurés

Le fichier de configuration orchestre les services suivants:

- **zabbix-db** : Une base de données MySQL 8.0 utilisée pour stocker les configurations et les données historiques de supervision.

- **zabbix-server** : Le moteur principal Zabbix (version 6.4) qui traite les données reçues et gère les alertes. Il écoute sur le port **10051**.
- **zabbix-web** : L'interface utilisateur basée sur Nginx et PHP. Elle est accessible sur le port **80** de l'hôte (redirigé vers le port 8080 du conteneur).

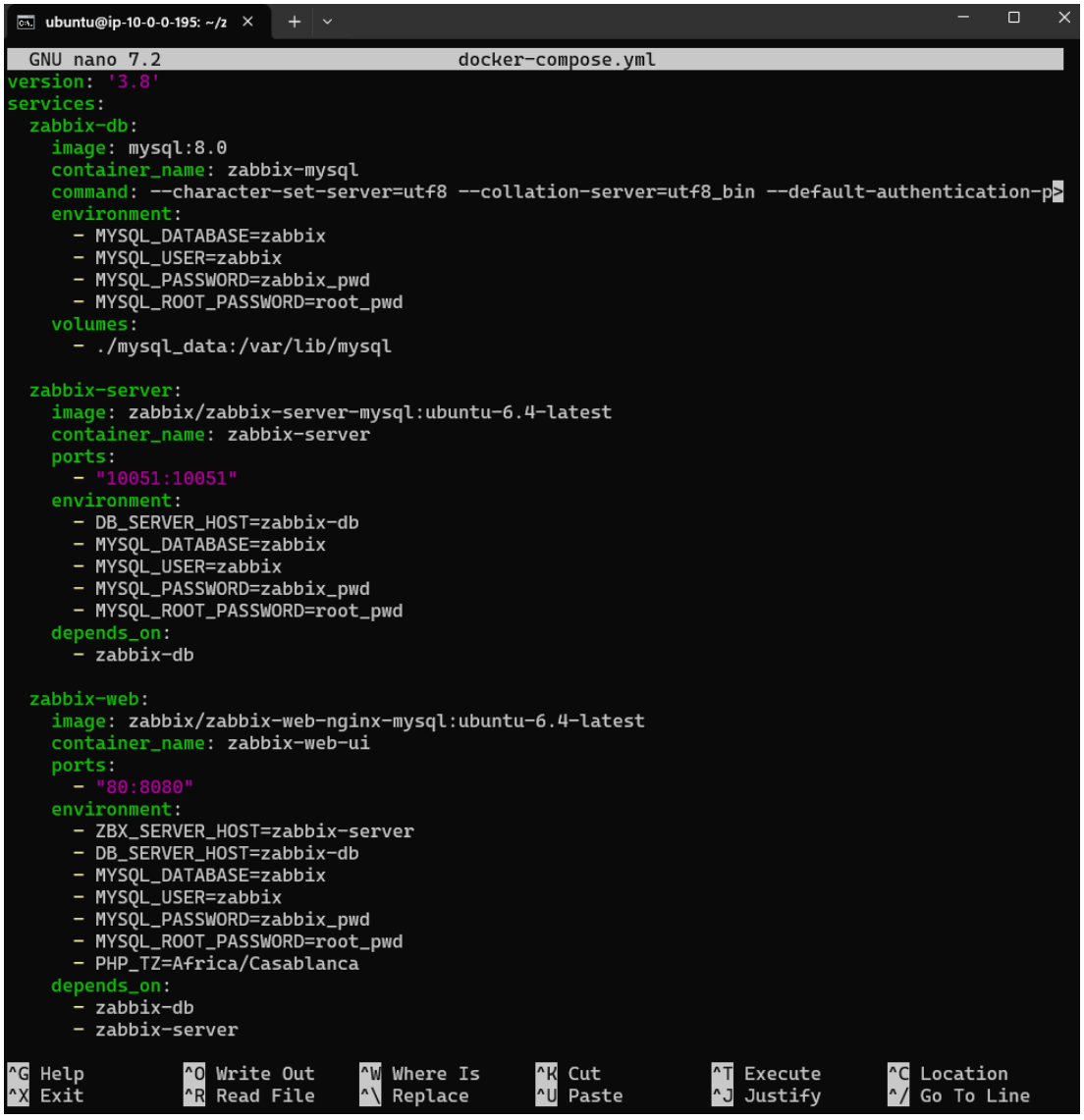


Figure 14 : Édition du fichier docker-compose.yml avec les variables d'environnement et les volumes de données.

4.3 Déploiement et Vérification opérationnelle

4.3.1 Lancement des conteneurs

Le déploiement est exécuté par la commande `sudo docker-compose up -d`, qui télécharge les images nécessaires et lance les conteneurs en mode arrière-plan (detached).

4.3.2 Contrôle de l'état des services

Une vérification systématique via `sudo docker ps` permet de confirmer que les trois conteneurs sont actifs et opérationnels:

1. **zabbix-web-ui** : Up (en ligne).
2. **zabbix-server** : Up (en ligne).
3. **zabbix-mysql** : Up (en ligne).

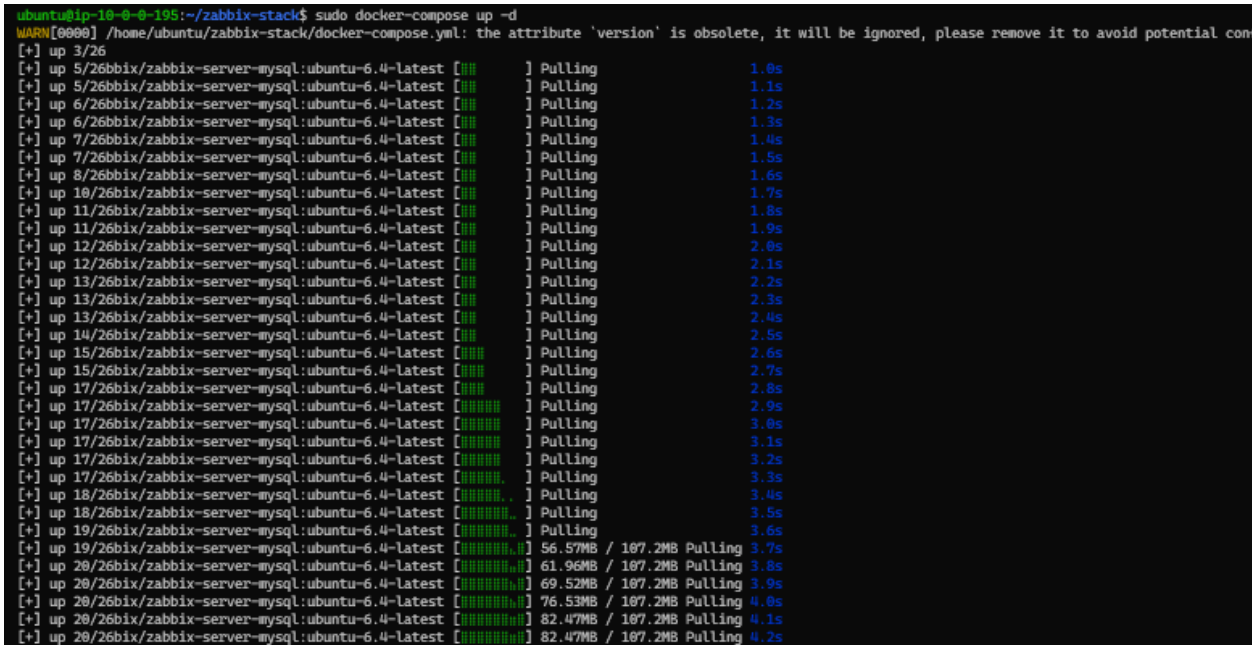


Figure 15 : Processus de téléchargement (pull) et de création des conteneurs.

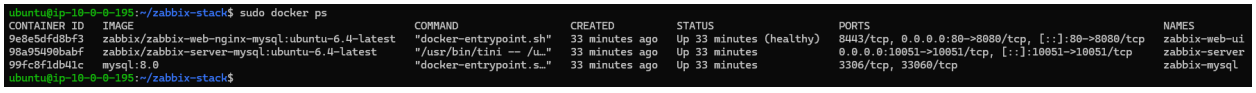


Figure 16 : Liste des conteneurs en cours d'exécution prouvant la stabilité de la stack.

4.4 Accès à la Console de Supervision

L'interface web est accessible depuis n'importe quel navigateur via l'adresse publique de l'instance AWS : `http://13.220.84.79`.

- **Authentification** : La connexion initiale s'effectue avec les identifiants administrateur par défaut (Username: `Admin`, Password: `zabbix`) .
- **Tableau de bord** : Une fois connecté, le Dashboard "Global view" s'affiche, confirmant que le serveur Zabbix est opérationnel ("Zabbix server is running: Yes").

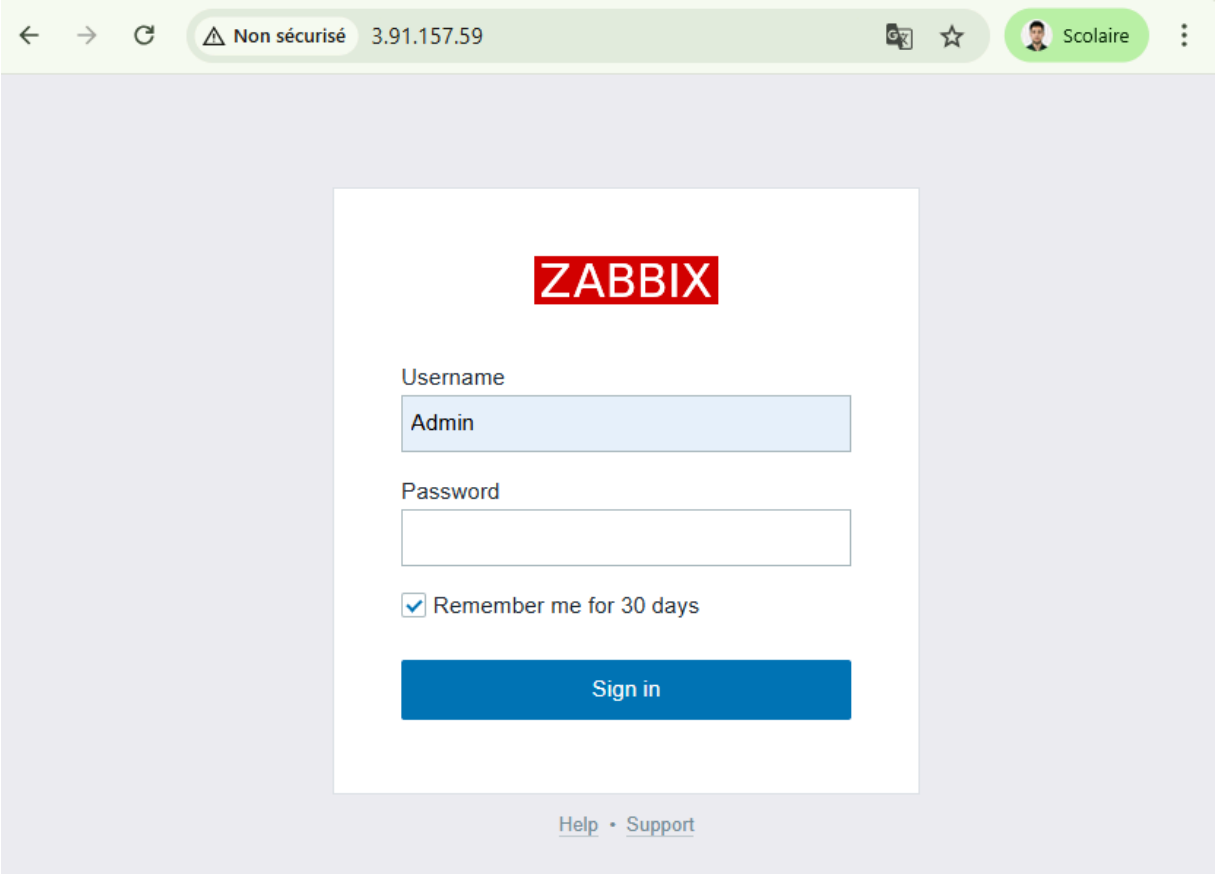


Figure 17 : Mire de connexion à l'interface Web de Zabbix.

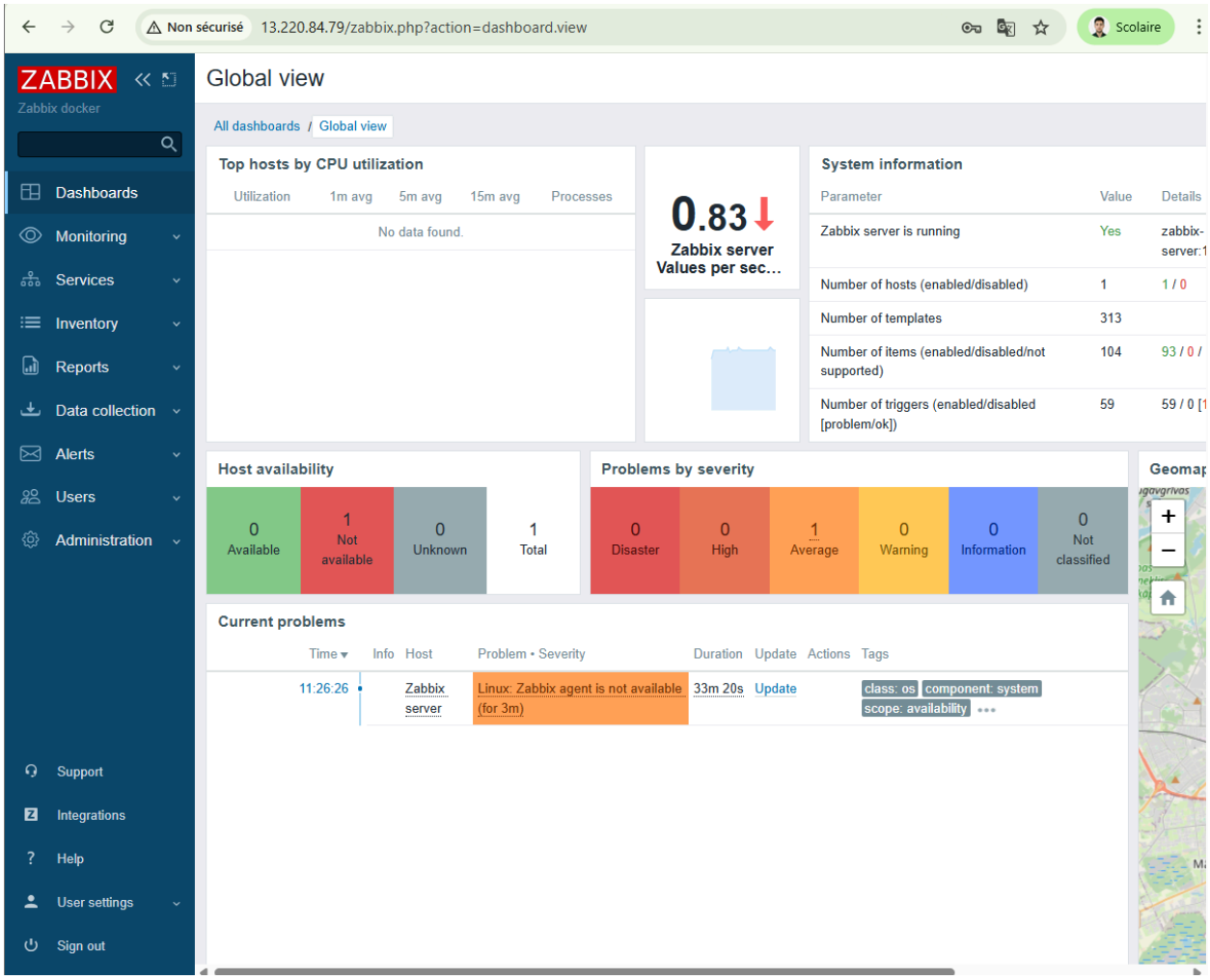


Figure 18 : Accueil réussi sur le Dashboard principal de la solution.

V. Configuration des Clients (Agents)

L'infrastructure de supervision nécessite l'installation d'un agent sur chaque machine cible pour collecter et transmettre les données de performance au serveur central. Nous avons opté pour le **Zabbix Agent 2**, une version moderne écrite en Go, offrant une meilleure gestion des plugins et des contrôles actifs.

5.1 Déploiement sur le Client Linux (Ubuntu)

L'instance **Zabbix-Client-Linux-BOULAALAM** a été configurée via une session SSH.

5.1.1 Installation de l'Agent

Pour garantir l'accès à la version 6.4, nous avons ajouté le dépôt officiel de Zabbix avant de procéder à l'installation du paquet `zabbix-agent2`.

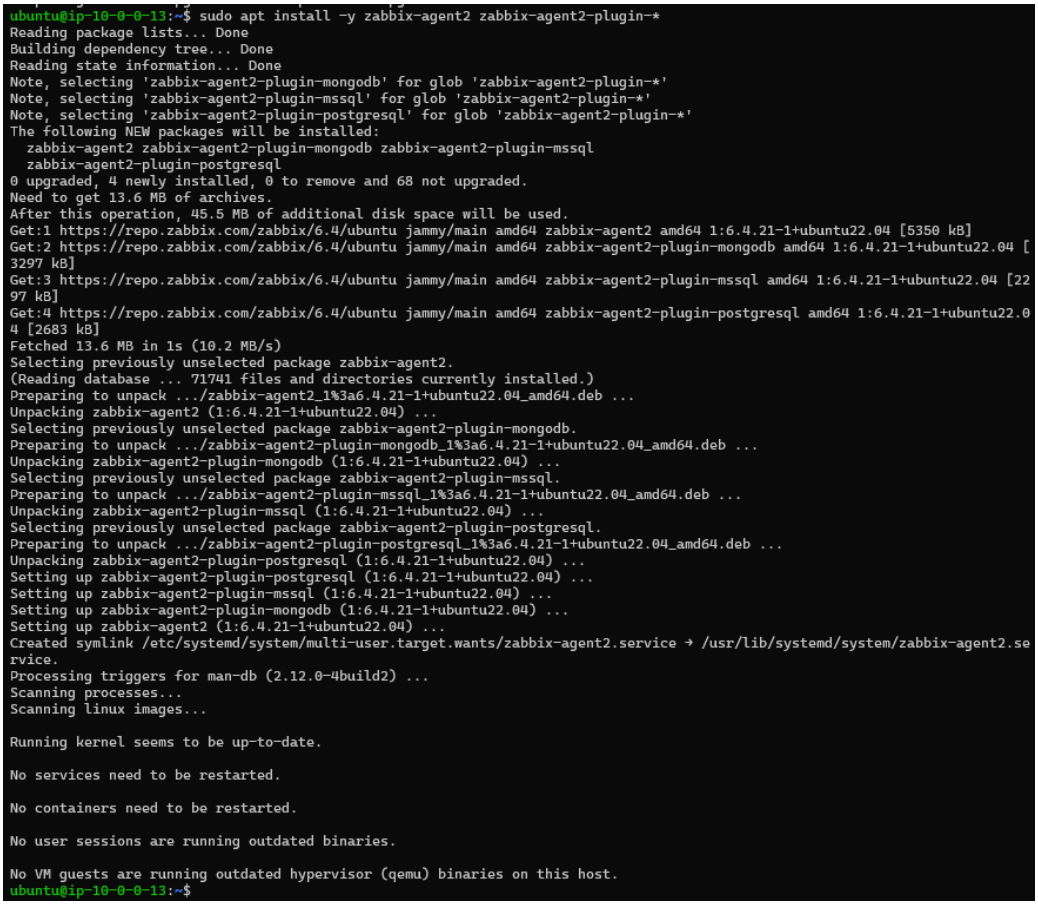


Figure 19 : Confirmation du téléchargement et de l'installation réussie du paquet Zabbix Agent 2.

5.1.2 Configuration et Sécurisation

La configuration s'effectue dans le fichier `/etc/zabbix/zabbix_agent2.conf`. Les paramètres clés ont été renseignés pour pointer vers l'adresse IP privée du serveur Zabbix (`10.0.0.195`) afin de sécuriser les flux au sein du VPC AWS:

- **Server** : `10.0.0.195` (IP privée du serveur).
- **ServerActive** : `10.0.0.195`.
- **Hostname** : `Zabbix-Client-Linux-BOULAALAM`.

Figure 20 : Extrait du fichier de configuration montrant les directives de connexion au serveur.

5.1.3 Activation du Service

Après modification, le service est redémarré et activé pour se lancer automatiquement au démarrage de l'instance. La commande `systemctl status` confirme que l'agent est opérationnel.

```
ubuntu@ip-10-0-0-13:~$ sudo systemctl restart zabbix-agent2
ubuntu@ip-10-0-0-13:~$ sudo systemctl enable zabbix-agent2
Synchronizing state of zabbix-agent2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable zabbix-agent2
ubuntu@ip-10-0-0-13:~$ sudo systemctl status zabbix-agent2
● zabbix-agent2.service - Zabbix Agent 2
     Loaded: loaded (/usr/lib/systemd/system/zabbix-agent2.service; enabled; preset: enabled)
     Active: active (running) since Wed 2026-01-07 11:19:08 UTC; 44s ago
   Main PID: 2131 (zabbix_agent2)
      Tasks: 7 (limit: 4515)
     Memory: 7.2M (peak: 7.8M)
        CPU: 76ms
     CGroup: /system.slice/zabbix-agent2.service
             └─2131 /usr/sbin/zabbix_agent2 -c /etc/zabbix/zabbix_agent2.conf

Jan 07 11:19:08 ip-10-0-0-13 systemd[1]: Started zabbix-agent2.service - Zabbix Agent 2.
Jan 07 11:19:08 ip-10-0-0-13 zabbix_agent2[2131]: 2026/01/07 11:19:08.486127 [MSSQL] plugin "/usr/sbin/zabbix-agent2-plugin>
Jan 07 11:19:08 ip-10-0-0-13 zabbix_agent2[2131]: 2026/01/07 11:19:08.491580 [PostgreSQL] plugin "/usr/sbin/zabbix-agent2-p>
Jan 07 11:19:08 ip-10-0-0-13 zabbix_agent2[2131]: 2026/01/07 11:19:08.497539 [MongoDB] plugin "/usr/sbin/zabbix-agent2-plug>
Jan 07 11:19:08 ip-10-0-0-13 zabbix_agent2[2131]: Starting Zabbix Agent 2 (6.4.21)
Jan 07 11:19:08 ip-10-0-0-13 zabbix_agent2[2131]: Zabbix Agent2 hostname: [Zabbix-Client-Linux-BOULAALAM]
Jan 07 11:19:08 ip-10-0-0-13 zabbix_agent2[2131]: Press Ctrl+C to exit.

ubuntu@ip-10-0-0-13:~$
```

Figure 21 : État du service affichant le statut "active (running)".

5.2 Déploiement sur le Client Windows (Windows Server)

L'administration de l'instance **Zabbix-Client-Windows-BOULAALAM** s'est faite via le protocole RDP.

5.2.1 Téléchargement et Installation MSI

L'installation sur Windows utilise un assistant graphique (MSI). Nous avons téléchargé l'agent 2 version 7.4.6 pour Windows 64-bit directement sur l'instance.

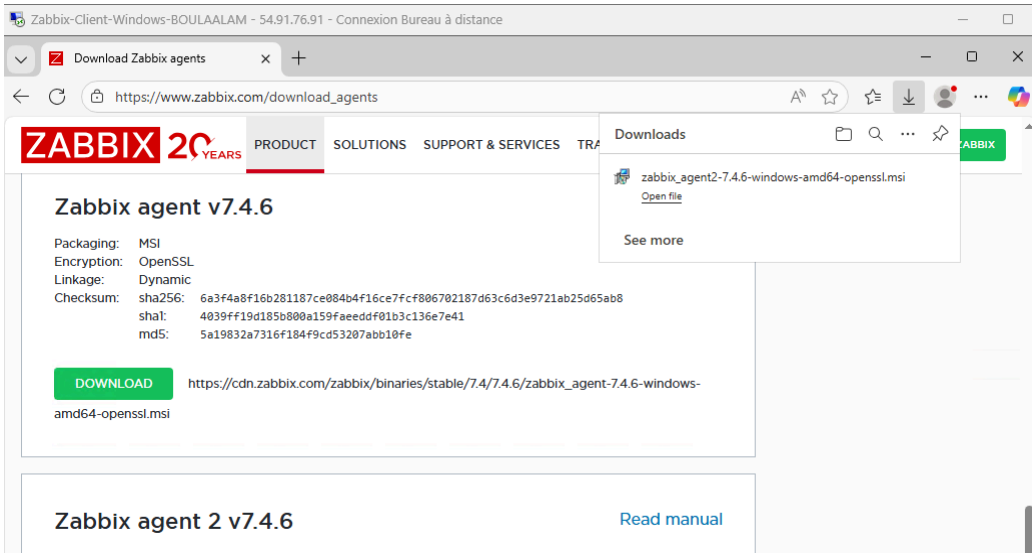


Figure 22 : Page de téléchargement des agents Zabbix sur le client Windows.

5.2.2 Paramétrage de l'Agent Windows

Lors de l'installation, les informations de connexion au serveur ont été saisies manuellement:

- **Host name** : `Zabbix-Client-Windows-BOULAALAM`.
- **Zabbix server IP** : `10.0.0.195`.
- **Agent listen port** : `10050`.

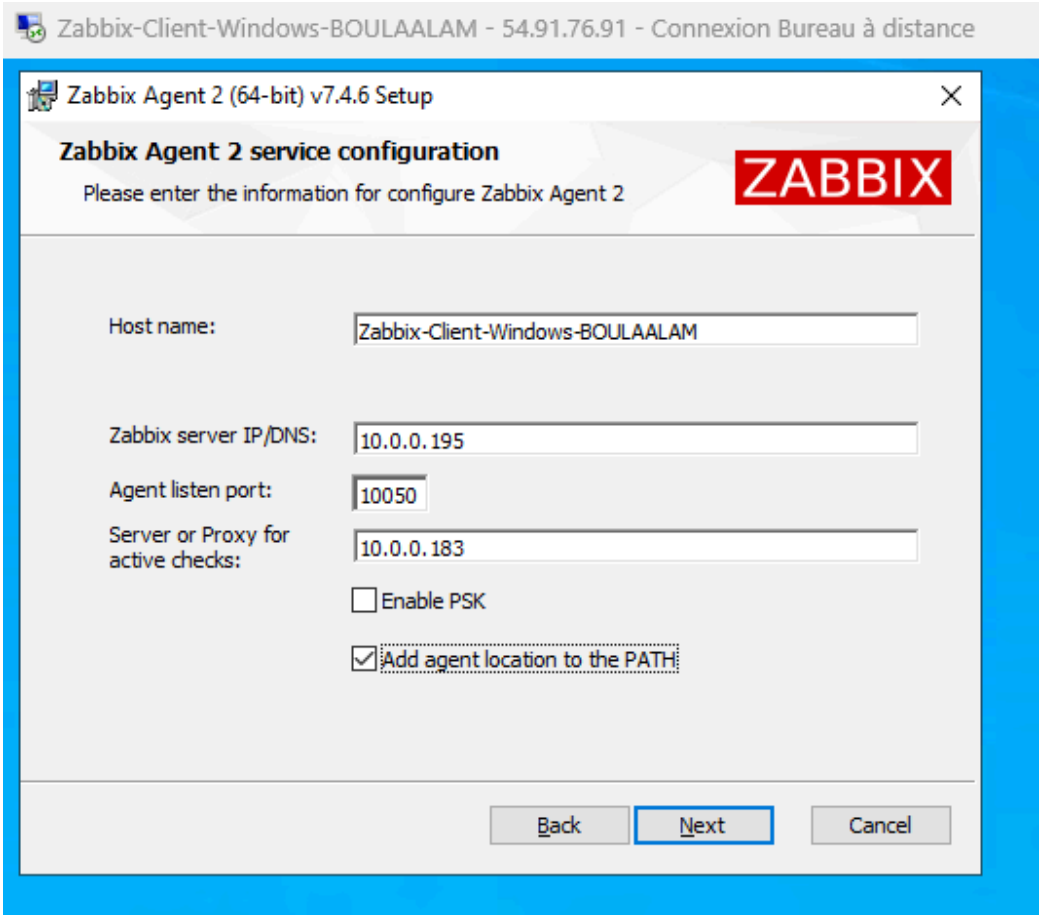


Figure 23 : Fenêtre de configuration du service Zabbix Agent 2 lors de l'installation MSI.

5.2.3 Configuration du Pare-feu et Vérification

Contrairement à Linux, Windows nécessite l'ouverture explicite du port **10050** dans le pare-feu local pour permettre au serveur d'interroger l'agent. Cette action a été réalisée via une commande PowerShell en tant qu'administrateur.

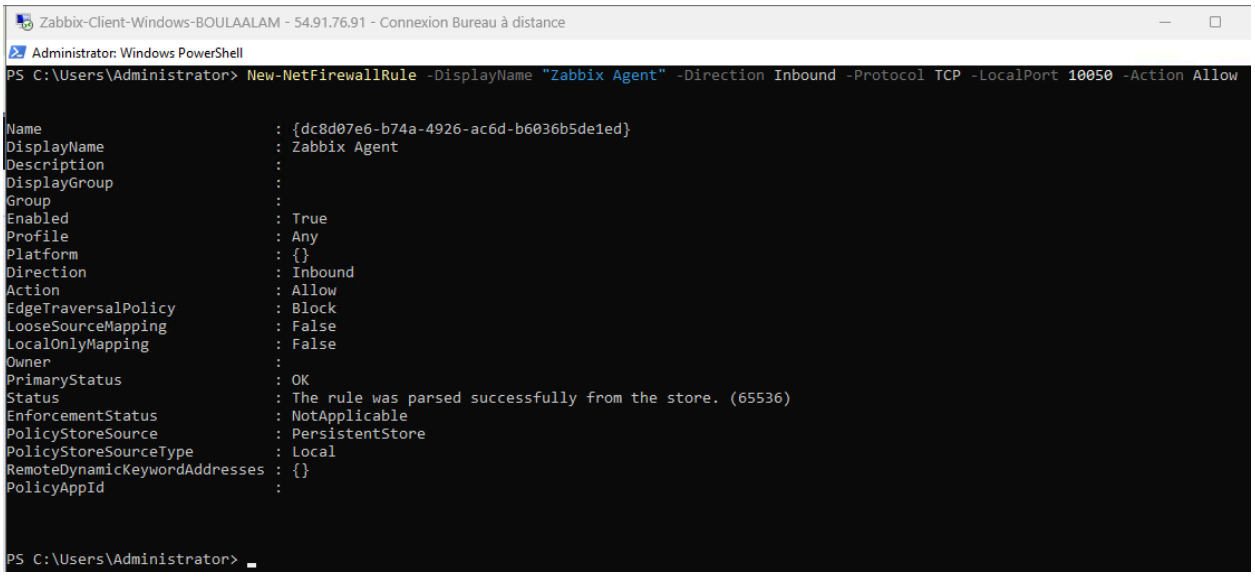


Figure 24 : Commande PowerShell créant la règle de pare-feu entrante pour le port 10050.

Enfin, le gestionnaire de services Windows confirme que l'agent fonctionne correctement en arrière-plan.

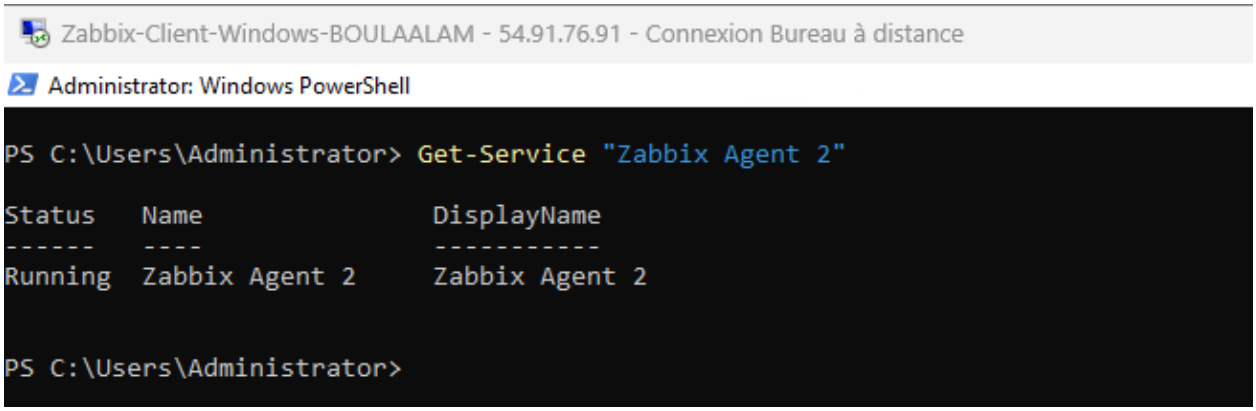


Figure 25 : Sortie PowerShell affichant le statut "Running" pour Zabbix Agent 2.

VI. Monitoring et Tableaux de Bord

Cette section présente l'intégration finale des hôtes dans la console Zabbix et la mise en œuvre des outils de visualisation nécessaires au pilotage de l'infrastructure hybride.

6.1 Intégration du Parc Hybride dans Zabbix

L'ajout des hôtes constitue l'étape de liaison entre les agents déployés et le serveur de supervision.

6.1.1 Enregistrement du Client Linux

Pour l'hôte **Zabbix-Client-Linux-BOULAALAM**, nous avons utilisé l'interface d'agent pointant vers l'adresse IP privée `10.0.0.13` sur le port `10050`. L'utilisation du template officiel "*Linux by Zabbix agent*" a permis de déployer instantanément 75 items de collecte et 30 déclencheurs d'alerte.

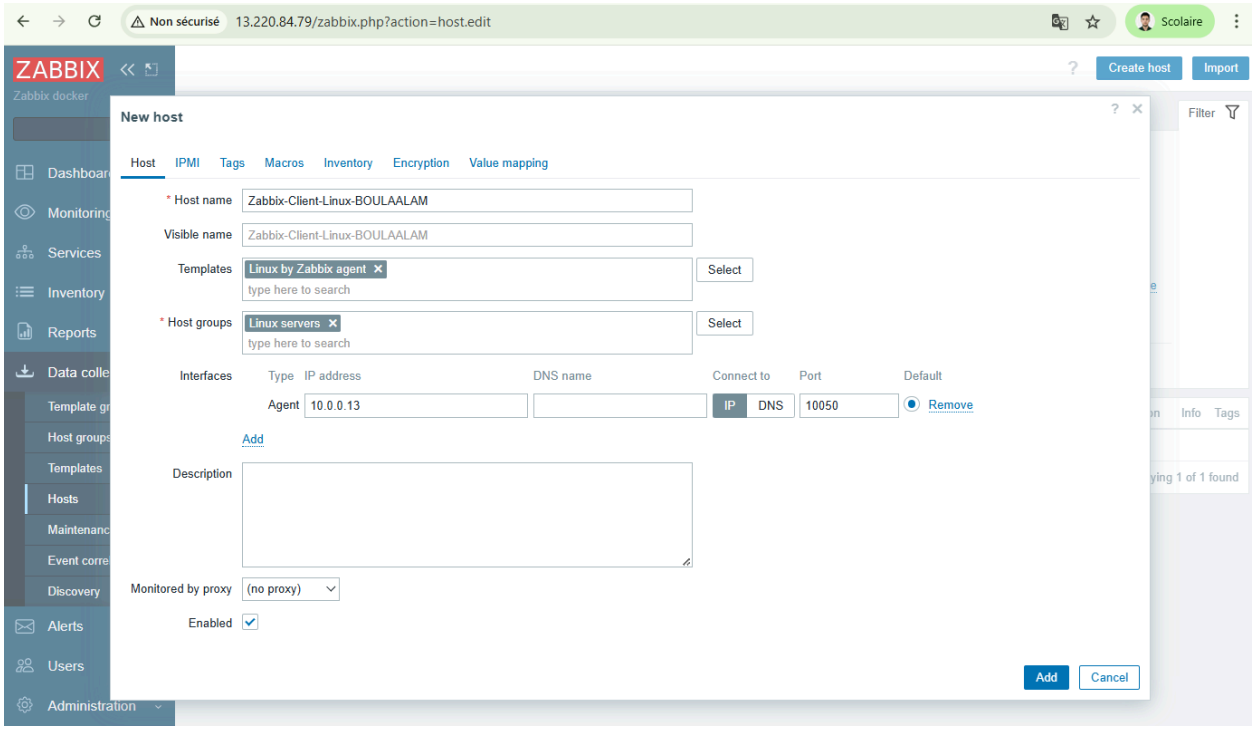


Figure 26 : Formulaire de configuration pour l'ajout de l'hôte Linux.

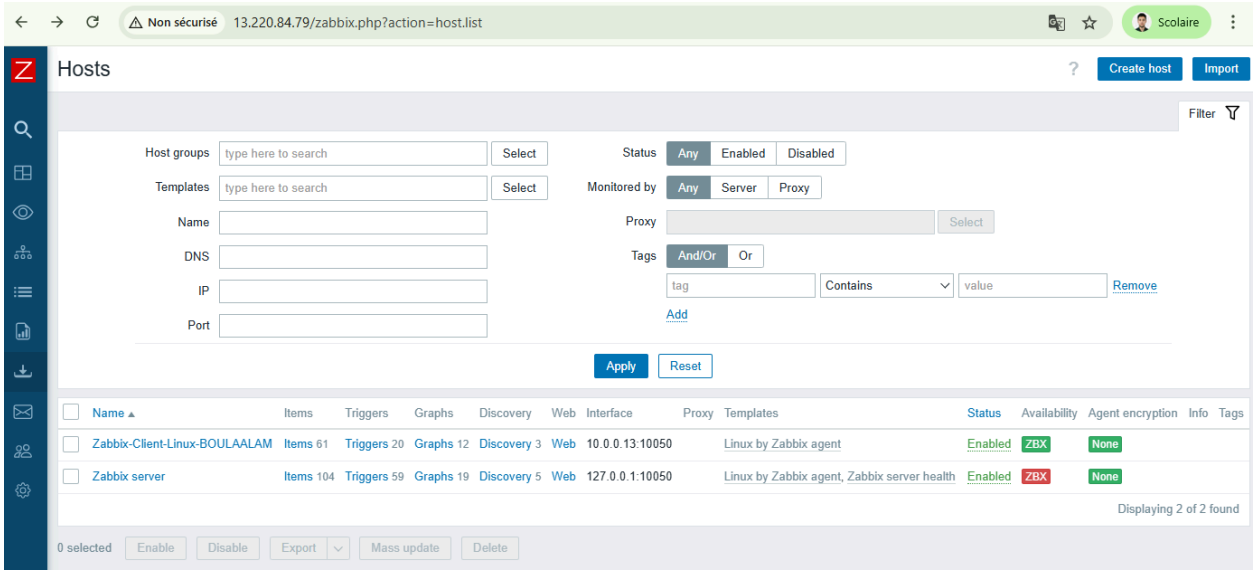


Figure 27 : Confirmation de la communication avec l'hôte Linux (Statut ZBX vert).

6.1.2 Enregistrement du Client Windows

De manière similaire, l'hôte **Zabbix-Client-Windows-BOULAALAM** a été intégré en renseignant son IP privée `10.0.0.183`. Le template "*Windows by Zabbix agent*" a été appliqué pour collecter les métriques spécifiques à l'écosystème Microsoft.

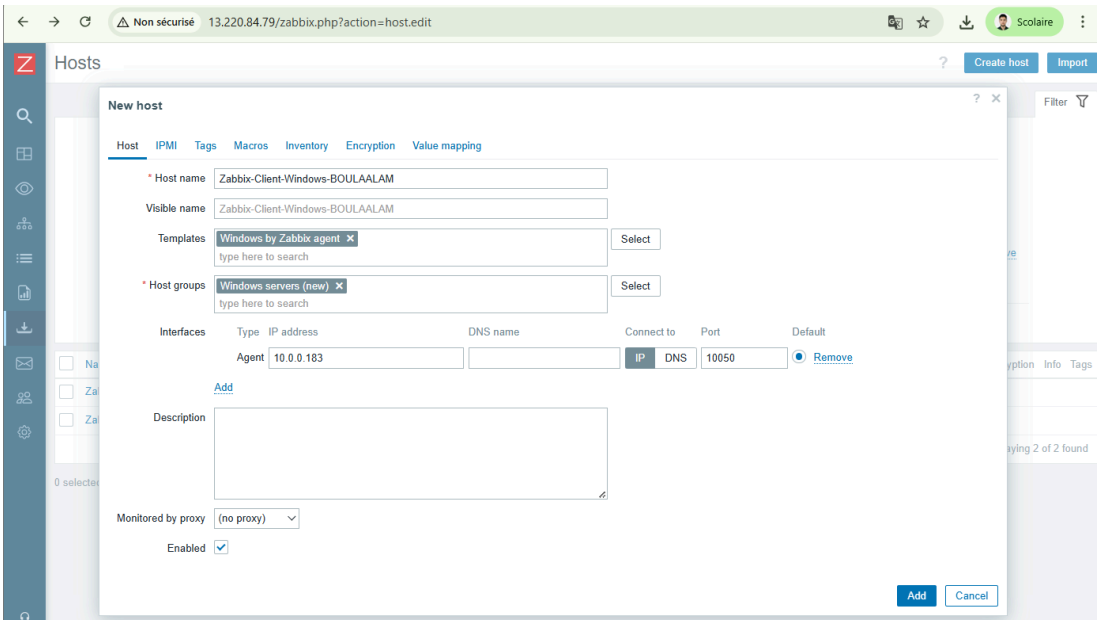


Figure 28 : Configuration de l'interface agent pour le serveur Windows.

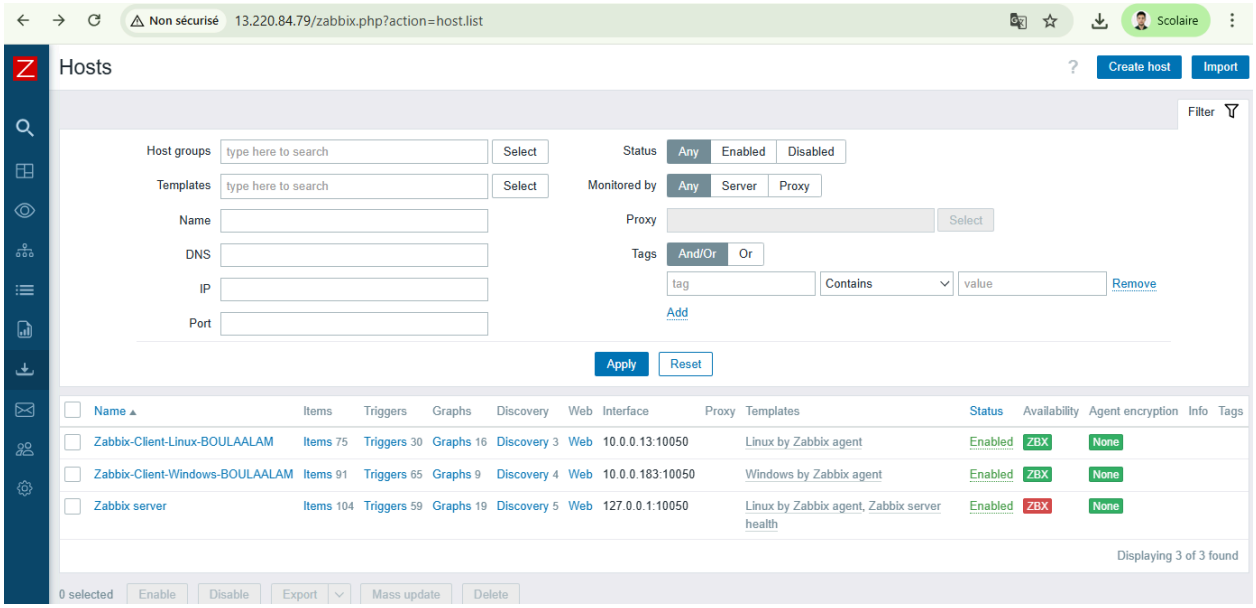


Figure 29 : Validation de la connectivité avec l'hôte Windows.

Une fois ces étapes terminées, la console affiche un état de santé global "vert" pour l'ensemble du parc supervisé.

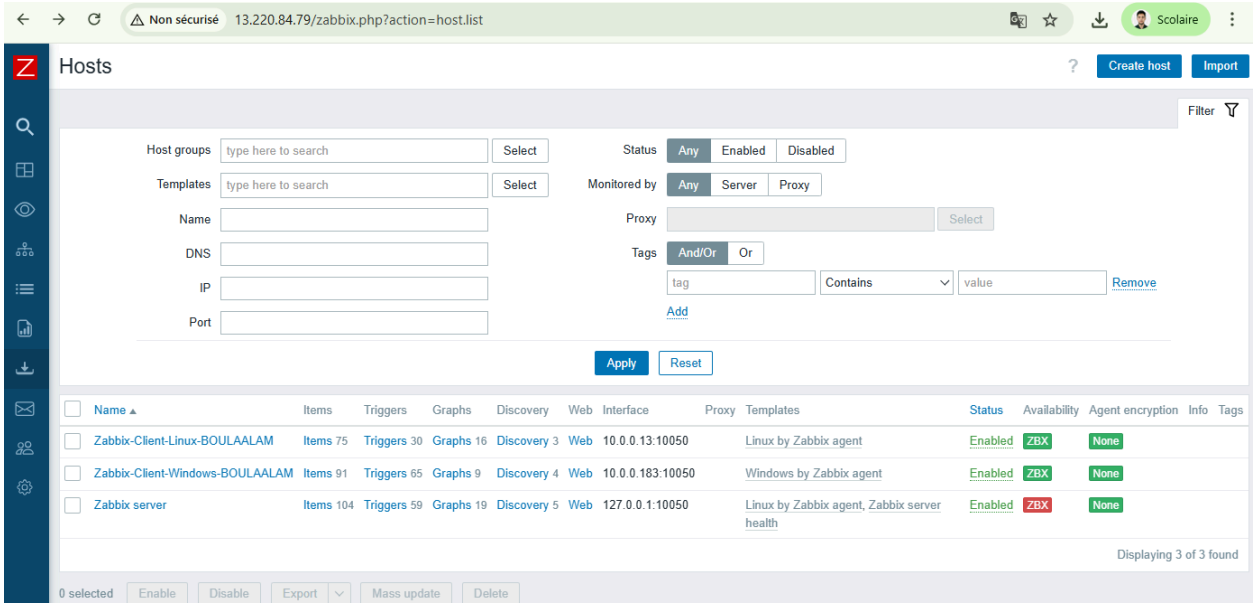


Figure 30 : Vue d'ensemble de la liste des hôtes actifs dans Zabbix.

6.2 Analyse des Données Collectées (Latest Data)

La fonctionnalité "*Latest data*" permet de vérifier en temps réel la remontée des informations brutes. Nous avons pu confirmer la réception des données critiques pour les deux systèmes :

- **Linux** : Utilisation CPU, charge système (load), et état des systèmes de fichiers.
- **Windows** : Utilisation de la mémoire vive (RAM), espace disque NTFS, et état des services système essentiels comme l'agent Amazon SSM.

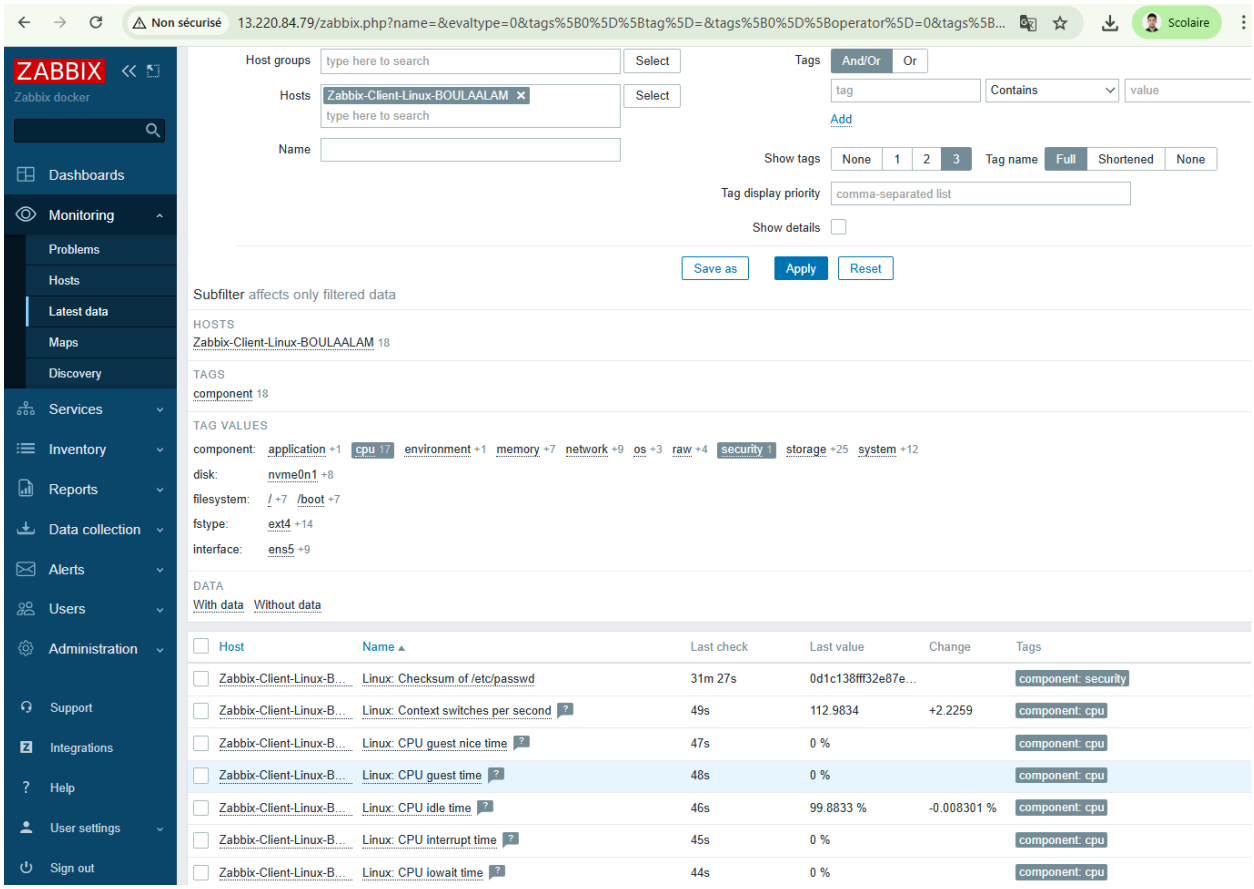


Figure 31 : Flux de données en temps réel provenant du client Linux.

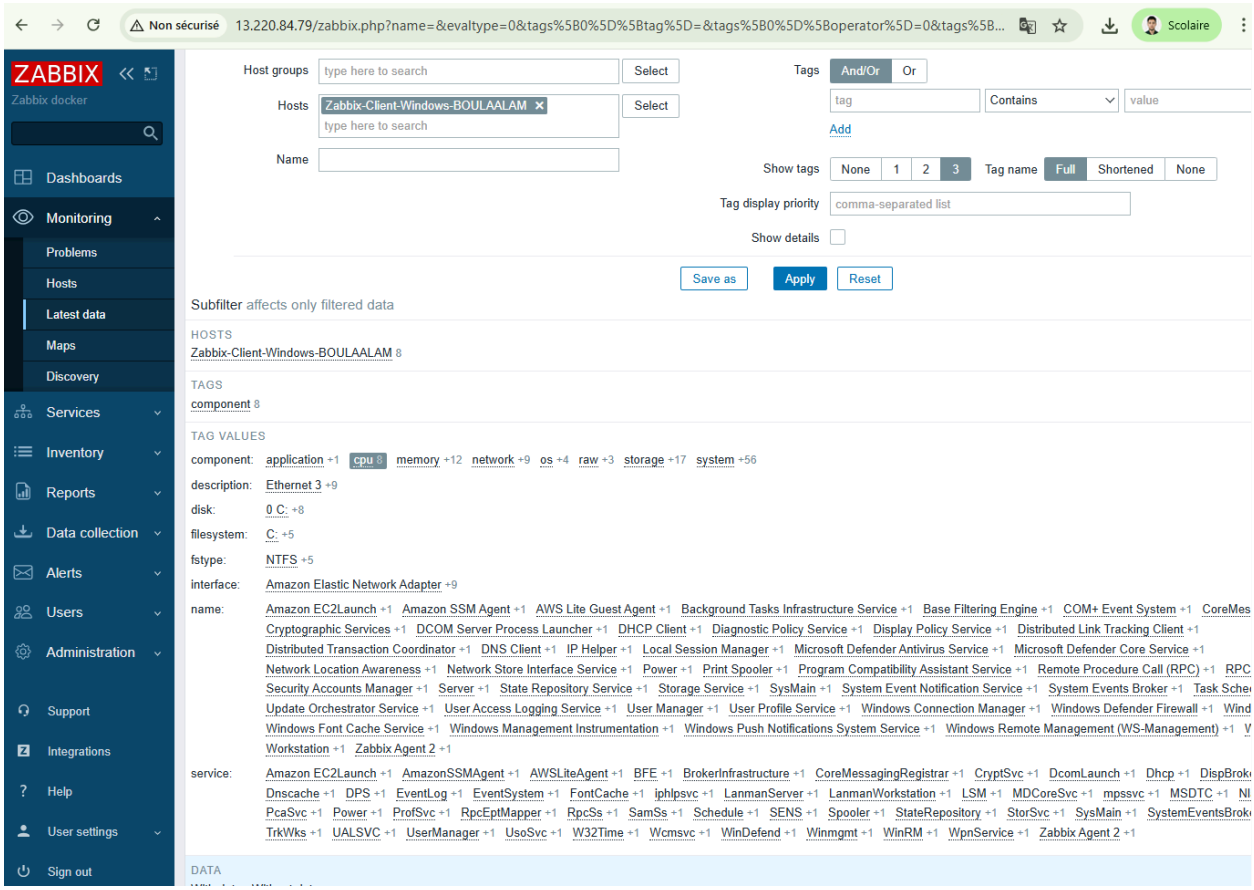


Figure 32 : Métriques de performance collectées sur le client Windows.

6.3 Visualisation de la Performance (Graphiques)

Pour faciliter l'analyse des tendances, des graphiques personnalisés ont été générés. Ces visuels permettent d'identifier rapidement des anomalies de consommation de ressources sur une période donnée.

- **Graphique CPU Linux** : Affiche la charge système moyenne sur la dernière heure.
- **Graphique RAM Windows** : Illustre le taux d'utilisation de la mémoire physique.

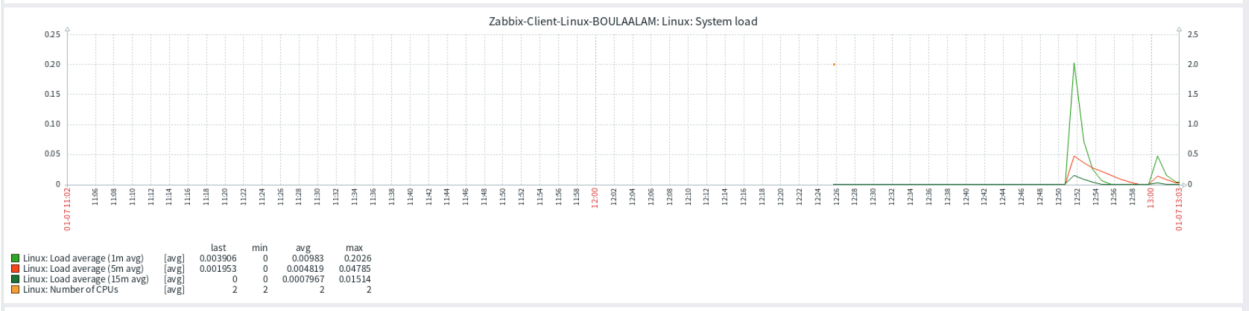


Figure 33 : Historique de la charge système du client Linux.

Figure 34 : Analyse graphique de l'utilisation mémoire sur Windows Server.

6.4 Gestion des Incidents et Alertes

La valeur ajoutée de Zabbix réside dans sa capacité à alerter l'administrateur en cas de dépassement de seuils critiques.

6.4.1 Simulation de Panne (Stress Test)

Pour valider le fonctionnement des déclencheurs (triggers), nous avons simulé une charge CPU intensive sur le client Linux via la commande `stress --cpu 4 --timeout 600`.

6.4.2 Déclenchement de l'Alerte

Le serveur a détecté une hausse anormale de la charge, faisant passer l'état du déclencheur en "PROBLEM" dans l'interface de gestion.

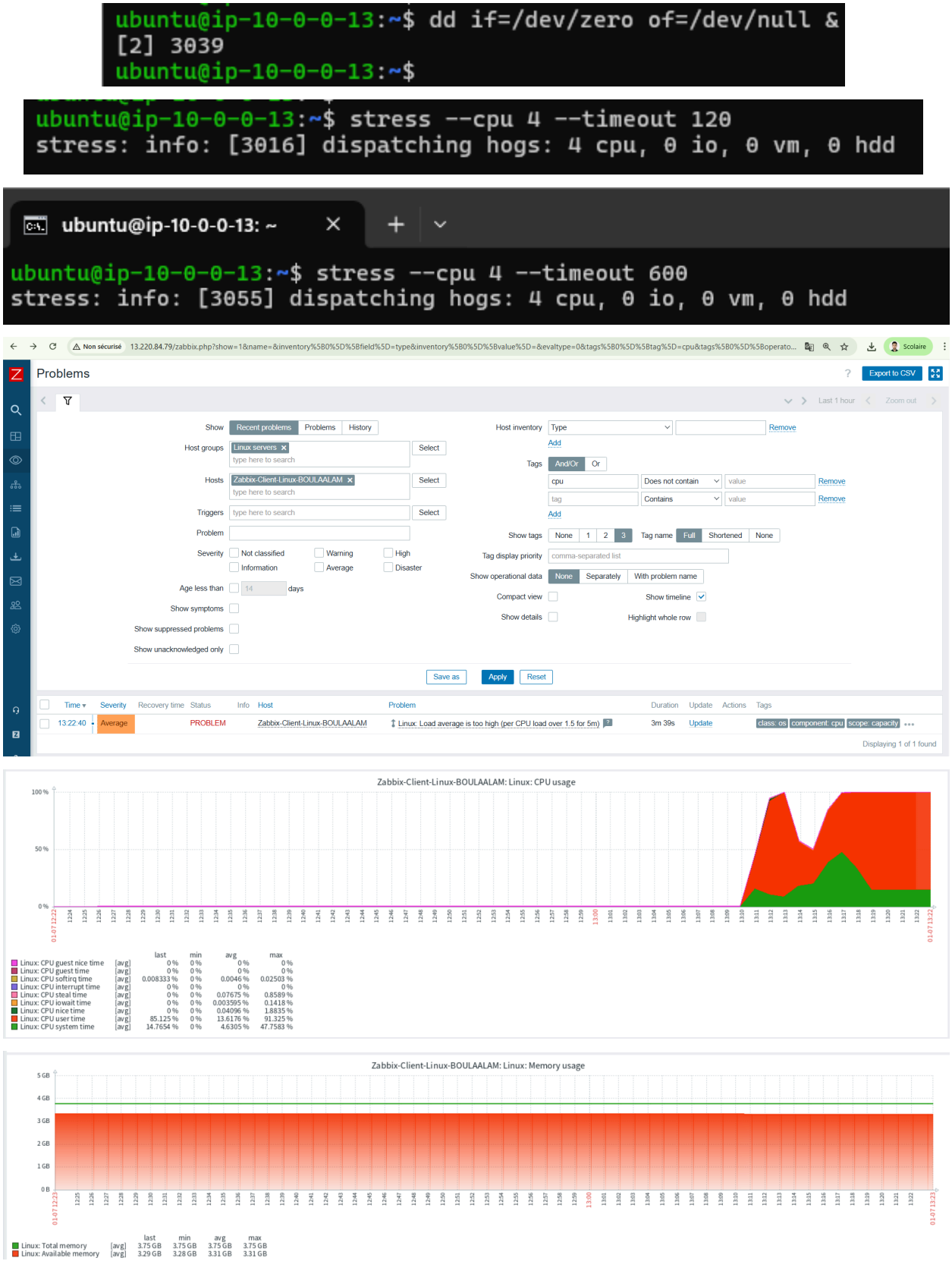


Figure 35 : Alerte active en rouge signalant une surcharge CPU.

6.4.3 Configuration d'une Action Corrective

Afin d'automatiser la réponse aux incidents, une action nommée "*Alerte Critique - BOULAALAM*" a été configurée. Elle a pour but d'envoyer immédiatement une notification au groupe d'administrateurs dès qu'une alerte de sévérité supérieure ou égale à "Warning" est détectée.

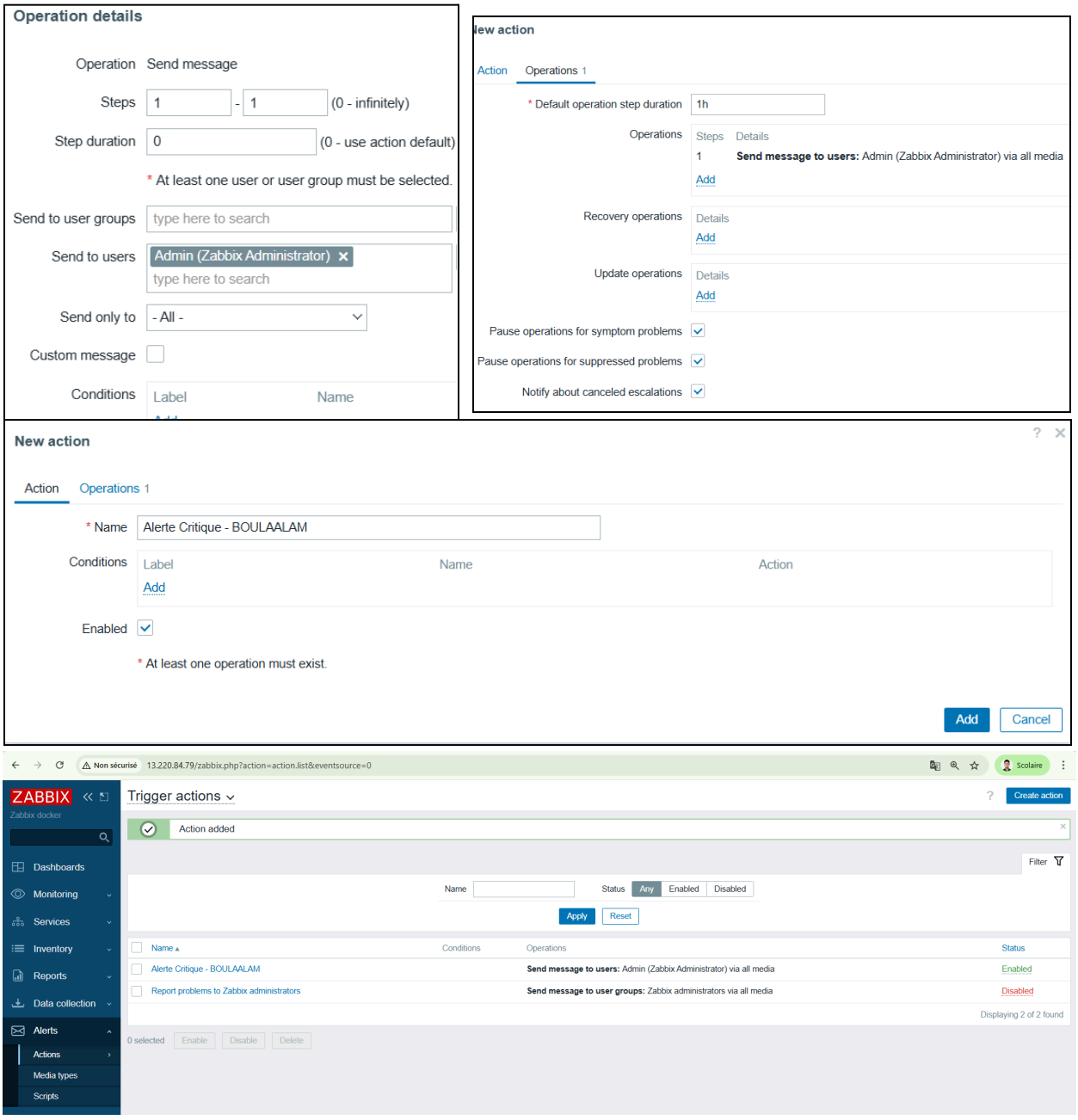


Figure 36 : Paramétrage de l'action automatique pour l'envoi de messages d'alerte.

6.5 Analyse du Tableau de Bord Final (Console de Pilotage)

Le point d'orgue de ce déploiement est la mise en place d'un tableau de bord centralisé, offrant une visibilité immédiate sur l'état de santé du parc hybride.

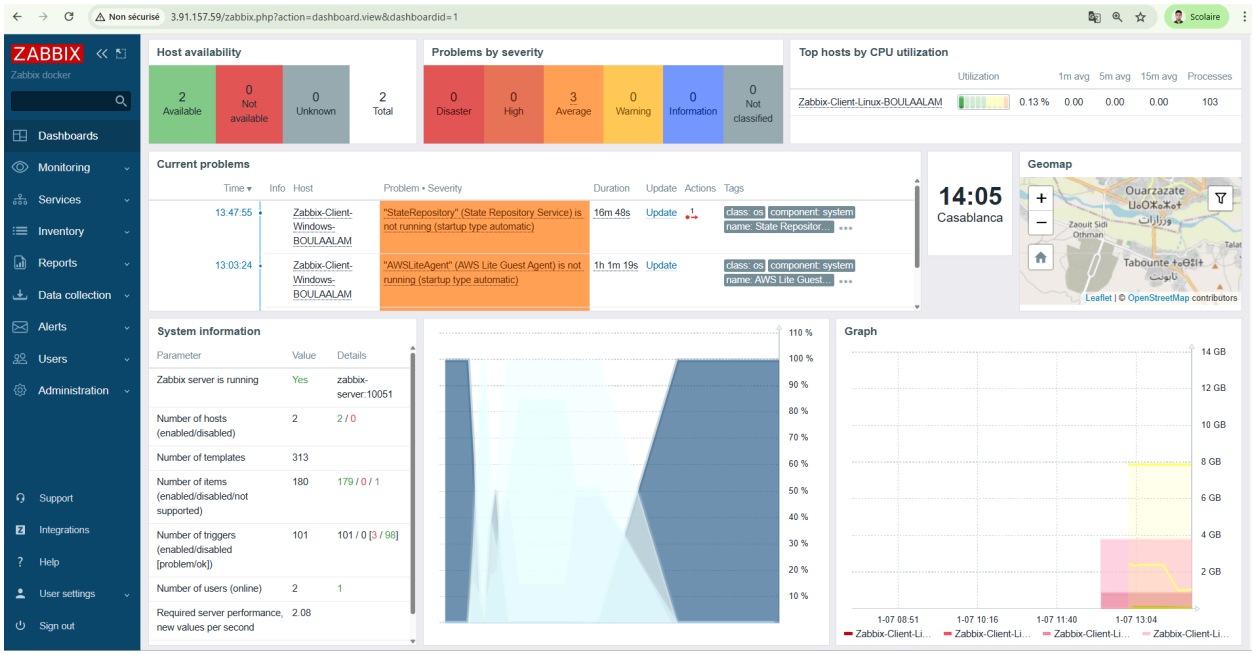


Figure 37: Vue d'ensemble du Dashboard "Global view" de l'infrastructure BOULAALAM.

Cette interface permet une gestion proactive grâce aux widgets suivants visibles sur la capture :

- **Disponibilité des Hôtes** (**Host availability**) : On observe deux indicateurs **verts** dans la section "Available". Cela confirme que le serveur Zabbix communique avec succès via le réseau privé AWS avec l'instance Linux et l'instance Windows simultanément.
- **Problèmes Actuels** (**Current problems**) : Le widget central démontre la sensibilité du système. Il identifie en temps réel deux alertes de sévérité "Average" sur le client Windows (services `StateRepository` et `AWSLiteAgent` non démarrés). Cela prouve que les déclencheurs (triggers) configurés sont opérationnels.
- **Utilisation CPU** (**Top hosts by CPU utilization**) : Le graphique en barres montre une consommation stable de **0.13%** pour le client Linux, confirmant que l'agent collecte les métriques avec précision.
- **Informations Système** : Le bloc de gauche confirme que le moteur Zabbix est "Running" et traite actuellement **2.08 valeurs par seconde**, témoignant de la fluidité de la stack conteneurisée sur l'instance `t3.large`.

- **Visualisation Graphique** : Les courbes en bas à droite permettent de suivre l'historique des ressources, facilitant ainsi la corrélation entre les événements système et la consommation de bande passante ou de mémoire.

Ce tableau de bord transforme des données brutes en informations exploitables, permettant à l'administrateur de superviser un parc hétérogène depuis une interface unique, sécurisée et hébergée sur le cloud.

VII. Difficultés Rencontrées et Solutions

Le déploiement d'une infrastructure cloud hybride présente des défis techniques liés à la connectivité réseau, aux permissions de l'environnement de laboratoire et à la persistance des données. Cette section recense les principaux problèmes identifiés et les mesures correctives appliquées.

7.1 Liste des problèmes et Solutions appliquées

7.1.1 Absence de connectivité Internet dans le VPC

Lors de la phase initiale d'installation de Docker sur l'instance Serveur, les commandes `sudo apt update` et `curl` ont échoué systématiquement avec l'erreur `Network is unreachable`.

- **Cause :** L'instance était située dans un sous-réseau sans route vers la sortie. La table de routage principale ne contenait que la route locale (`10.0.0.0/16`).
- **Solution :** Une nouvelle table de routage (`rtb-021c61510af386e98`) a été créée. Nous avons configuré manuellement une route vers la destination `0.0.0.0/0` pointant vers l'Internet Gateway (`igw-04bf42ab6c083a645`). Enfin, le sous-réseau public a été explicitement associé à cette table.

7.1.2 Limitations de permissions AWS Learner Lab

Durant la navigation dans la console AWS VPC, des messages d'erreur de type `Not authorized to perform: route53resolver` sont apparus.

- **Cause :** L'environnement AWS Academy impose des restrictions strictes sur les politiques IAM (Identity and Access Management) pour les comptes étudiants.
- **Solution :** Après analyse, il a été déterminé que ces erreurs concernaient des services DNS secondaires n'impactant pas le cœur du projet (EC2 et VPC). Nous avons ignoré ces alertes pour nous concentrer sur la configuration réseau fondamentale.

7.1.3 Expiration de session et changement d'adressage IP

La session du Learner Lab est limitée dans le temps. À chaque expiration, les instances sont arrêtées et les adresses IP publiques sont régénérées.

- **Cause :** Comportement standard des environnements de laboratoire éphémères d'AWS.
- **Solution :** Pour assurer la continuité, nous avons utilisé l'adressage **IPv4 privé** (`10.0.0.195` pour le serveur et `10.0.0.13` pour le client) pour la configuration des agents Zabbix. Cela a permis aux agents de se reconnecter automatiquement au serveur Docker après chaque redémarrage, malgré le changement d'IP publique.

7.1.4 Délai de déclenchement des alertes (Triggers)

Suite à l'exécution de la commande `stress --cpu 4`, aucune alerte n'apparaissait immédiatement dans l'onglet **Problems**.

- **Cause :** Le modèle standard Zabbix attend que le CPU dépasse 90% pendant 5 minutes consécutives avant de déclencher l'alerte.
- **Solution :** Nous avons modifié temporairement la configuration du déclencheur (trigger) `High CPU utilization` en abaissant le seuil à **20%** et le temps d'analyse à **1 minute**. Cette modification a permis de valider instantanément le flux d'alerte.

7.2 Leçons apprises

La réalisation de ce projet a permis d'acquérir des réflexes techniques cruciaux pour un futur ingénieur :

1. **Priorité au routage :** Une instance cloud n'est fonctionnelle que si sa table de routage est explicitement configurée pour le trafic sortant.
2. **Sécurité par segmentation :** L'utilisation d'IP privées pour la communication entre agents et serveur est une bonne pratique indispensable pour la sécurité et la stabilité (persistance).
3. **Flexibilité de la conteneurisation :** Docker a prouvé sa valeur en permettant de relancer une stack complexe de supervision en une seule commande (`docker-compose up -d`) après une coupure d'infrastructure.

VIII. Conclusion

8.1 Bilan du projet

La réalisation de ce projet de fin de module a permis de concevoir et de déployer une solution complète de supervision centralisée au sein d'un environnement Cloud AWS. L'objectif principal, qui consistait à surveiller un parc hybride composé d'instances Ubuntu Linux et Windows Server, a été pleinement atteint.

L'infrastructure repose sur un socle réseau robuste (VPC, IGW, Tables de routage) et une stack logicielle moderne entièrement conteneurisée grâce à Docker et Docker-Compose. Le déploiement de l'**Agent Zabbix 2** sur les clients a permis une remontée de données précise et en temps réel, comme en témoigne le tableau de bord final présentant des indicateurs de performance critiques et des alertes actives. Malgré les contraintes liées aux limitations du *AWS Learner Lab* et aux défis de routage initialement rencontrés, la persévérance technique a permis de livrer une infrastructure opérationnelle et sécurisée.

8.2 Compétences acquises

Ce projet a été un catalyseur pour consolider mes compétences en ingénierie système et réseau :

- **Administration Cloud** (**AWS**) : Maîtrise de la gestion des ressources EC2, de la segmentation réseau via les VPC et de la sécurisation des flux par les Groupes de Sécurité.
- **Conteneurisation et Orchestration** : Capacité à déployer et à interconnecter des micro-services (Serveur, Web, Base de données) via Docker-Compose pour assurer la portabilité et la résilience de l'application.
- **Supervision Hybride** : Expertise dans la configuration d'outils de monitoring (Zabbix) pour des environnements hétérogènes, incluant la gestion des templates, des triggers et de la visualisation de données.
- **Résolution de problèmes** (**Troubleshooting**) : Aptitude à diagnostiquer des pannes réseau complexes (routage, pare-feu) et à adapter des configurations logicielles en environnement restreint.

8.3 Perspectives d'amélioration

Bien que fonctionnelle, cette infrastructure constitue une base pouvant être enrichie par plusieurs axes d'amélioration pour répondre à des besoins de niveau entreprise :

1. **Automatisation** (**Infrastructure as Code**) : L'utilisation d'outils comme Terraform ou Ansible permettrait de provisionner le VPC et d'installer les agents Zabbix de manière totalement automatisée et reproductible.
2. **Sécurisation des Accès** : L'implémentation de certificats SSL/TLS pour l'interface Web Zabbix et le chiffrement des communications entre les agents et le serveur via des clés pré-partagées (PSK) renforceraient la confidentialité des données de supervision.
3. **Notifications Externes** : La configuration d'actions pour envoyer des alertes vers des canaux externes (Slack, Telegram ou Emails) permettrait une réactivité immédiate sans consultation permanente de la console.
4. **Haute Disponibilité** : Pour un parc de plus grande envergure, le déploiement de **Zabbix Proxies** permettrait de décharger le serveur principal et d'assurer une supervision plus scalable sur plusieurs régions AWS.

En conclusion, ce travail démontre la puissance de l'alliance entre le Cloud et la conteneurisation pour répondre aux enjeux modernes de la supervision des infrastructures informatiques.

Annexes

1 Fichiers de configuration complets

1.1 Docker Compose (Serveur Zabbix)

Ce fichier, localisé dans `~/zabbix-stack/docker-compose.yml`, orchestre les trois conteneurs du serveur central.

```
version: '3.8'
services:
  zabbix-db:
    image: mysql:8.0
    container_name: zabbix-mysql
     command: --character-set-server=utf8 --collation-server=utf8_bin
--default-authentication-plugin=mysql_native_password
    environment:
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=zabbix_pwd
      - MYSQL_ROOT_PASSWORD=root_pwd
    volumes:
      - ./mysql_data:/var/lib/mysql

  zabbix-server:
    image: zabbix/zabbix-server-mysql:ubuntu-6.4-latest
    container_name: zabbix-server
    ports:
      - "10051:10051"
    environment:
      - DB_SERVER_HOST=zabbix-db
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=zabbix_pwd
      - MYSQL_ROOT_PASSWORD=root_pwd
    depends_on:
      - zabbix-db

  zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql:ubuntu-6.4-latest
    container_name: zabbix-web-ui
    ports:
      - "80:8080"
    environment:
      - ZBX_SERVER_HOST=zabbix-server
      - DB_SERVER_HOST=zabbix-db
      - MYSQL_DATABASE=zabbix
```

```
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=zabbix_pwd
      - MYSQL_ROOT_PASSWORD=root_pwd
      - PHP_TZ=Africa/Casablanca
    depends_on:
      - zabbix-db
      - zabbix-
```

1.2 Zabbix Agent 2 (Paramètres critiques)

Les fichiers de configuration ont été adaptés pour la communication au sein du VPC AWS.

- **Configuration Linux** (`/etc/zabbix/zabbix_agent2.conf`) :
  - `Server=10.0.0.195`
  - `ServerActive=10.0.0.195`
  - `Hostname=Zabbix-Client-Linux-BOULAALAM`
- **Configuration Windows** (`C:\Program Files\Zabbix Agent 2\zabbix_agent2.conf`) :
  - `Server=10.0.0.195`
  - `ServerActive=10.0.0.183`
  - `Hostname=Zabbix-Client-Windows-BOULAALAM`

2 Scripts et Commandes utilisés

2.1 Administration Docker (Serveur)

- **Lancement des services** : `sudo docker-compose up -d`
- **Vérification de l'état** : `sudo docker ps`
- **Lecture des logs** : `sudo docker logs -f zabbix-server`

2.2 Gestion de l'Agent Linux

- **Installation des dépôts** : `sudo dpkg -i zabbix-release_6.4-1+ubuntu22.04_all.deb`
- **Redémarrage du service** : `sudo systemctl restart zabbix-agent2`
- **Activation au démarrage** : `sudo systemctl enable zabbix-agent2`

2.3 Commandes PowerShell (Windows)

- **Ouverture du port 10050** :
  `New-NetFirewallRule -DisplayName "Zabbix Agent" -Direction Inbound -Protocol TCP -LocalPort 10050 -Action Allow`
- **Vérification du service** : `Get-Service "Zabbix Agent 2"`

2.4 Test de charge (Simulation d'alerte)

- **Stress CPU** : `stress --cpu 4 --timeout 600`
- **Alternative (boucle infinie)** : `dd if=/dev/zero of=/dev/null &`

3 Glossaire technique

- **AMI** : Amazon Machine Image.
- **IGW** : Internet Gateway (Passerelle Internet).
- **Trigger** : Déclencheur logique dans Zabbix pour définir une alerte.
- **VPC** : Virtual Private Cloud (Réseau privé virtuel).
- **ZBX** : Indicateur de disponibilité de l'agent Zabbix dans l'interface web.