

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 :

Mohamed BOULAALAM

Encadré par :

Pr. Azeddine KHIAT

Tableau de matière :

Tableau de matière :	3
Table des figures :	6
I. Introduction	8
1.1 Contexte du projet	8
1.2 Objectifs du projet	8
1.3 Outils et Technologies utilisés	8
Dépôt GitHub	9
II. Architecture Réseau	10
2.1 Schéma de l'Architecture	10
2.2 Configuration du Virtual Private Cloud (VPC)	11
2.2.1 Création du VPC et du Sous-réseau	11
2.2.2 Connectivité Internet et Routage	12
2.3 Configuration des Security Groups (Pare-feu)	14
2.3.1 Security Group pour le Serveur Zabbix	14
2.3.2 Security Group pour les Clients (Linux & Windows)	15
III. Architecture des Instances EC2	15
3.1 Présentation des instances déployées	16
3.1.1 Le Serveur Central (Zabbix Server)	16
3.1.2 Le Client Linux	17
3.1.3 Le Client Windows	18
3.2 Justification technique des types d'instances	19
3.3 Vue d'ensemble du parc opérationnel	20
IV. Déploiement du Serveur Zabbix	21
4.1 Préparation et Installation de l'environnement Docker	21
4.1.1 Accès au serveur et mise à jour	21
4.1.2 Installation de Docker Engine et Docker Compose	21
4.2 Configuration de la Stack via Docker-Compose	22
4.2.1 Description des services configurés	22
4.3 Déploiement et Vérification opérationnelle	24
4.3.1 Lancement des conteneurs	24

4.3.2 Contrôle de l'état des services	24
4.4 Accès à la Console de Supervision	25
V. Configuration des Clients (Agents)	26
5.1 Déploiement sur le Client Linux (Ubuntu)	26
5.1.1 Installation de l'Agent	27
5.1.2 Configuration et Sécurisation	27
5.1.3 Activation du Service	28
5.2 Déploiement sur le Client Windows (Windows Server)	29
5.2.1 Téléchargement et Installation MSI	29
5.2.2 Paramétrage de l'Agent Windows	30
5.2.3 Configuration du Pare-feu et Vérification	30
VI. Monitoring et Tableaux de Bord	32
6.1 Intégration du Parc Hybride dans Zabbix	32
6.1.1 Enregistrement du Client Linux	32
6.1.2 Enregistrement du Client Windows	33
6.2 Analyse des Données Collectées (Latest Data)	35
6.3 Visualisation de la Performance (Graphiques)	36
6.4 Gestion des Incidents et Alertes	37
6.4.1 Simulation de Panne (Stress Test)	37
6.4.2 Déclenchement de l'Alerte	37
6.4.3 Configuration d'une Action Corrective	39
6.5 Analyse du Tableau de Bord Final (Console de Pilotage)	40
VII. Difficultés Rencontrées et Solutions	42
7.1 Liste des problèmes et Solutions appliquées	42
7.1.1 Absence de connectivité Internet dans le VPC	42
7.1.2 Limitations de permissions AWS Learner Lab	42
7.1.3 Expiration de session et changement d'adressage IP	43
7.1.4 Délai de déclenchement des alertes (Triggers)	43
7.2 Leçons apprises	43
VIII. Conclusion	44
8.1 Bilan du projet	44
8.2 Compétences acquises	44

8.3 Perspectives d'amélioration	45
Annexes	46
1 Fichiers de configuration complets	46
1.1 Docker Compose (Serveur Zabbix)	46
1.2 Zabbix Agent 2 (Paramètres critiques)	47
2 Scripts et Commandes utilisés	47
2.1 Administration Docker (Serveur)	47
2.2 Gestion de l'Agent Linux	47
2.3 Commandes PowerShell (Windows)	48
2.4 Test de charge (Simulation d'alerte)	48
3 Glossaire technique	48

Table des figures :

Figure 1 : Création du VPC avec le tag identifiant l'étudiant.	11
Figure 2 : Configuration du sous-réseau et activation de l'IP publique.	12
Figure 3 : Internet Gateway attachée avec succès au VPC.	13
Figure 4 : Table de routage configurée avec la route vers l'Internet Gateway.	13
Figure 5 : Détails des règles entrantes pour le groupe de sécurité du serveur.	14
Figure 6 : Configuration du groupe de sécurité pour le parc hybride.	15
Figure 7 : Détails techniques et résumé de l'instance Serveur Zabbix.	17
Figure 8 : Détails techniques et résumé de l'instance Client Linux.	18
Figure 9 : Détails techniques et résumé de l'instance Client Windows.	19
Figure 10 : Tableau de bord EC2 affichant les 3 instances actives avec les tags personnalisés.	20
Figure 11 : Connexion SSH réussie au serveur Zabbix avec affichage des informations système.	21
Figure 12 : Vérification de la version de Docker installée.	
Figure 13 : Vérification de la version de Docker Compose.	22
Figure 14 : Édition du fichier docker-compose.yml avec les variables d'environnement et les volumes de données.	23
Figure 15 : Processus de téléchargement (pull) et de création des conteneurs.	24
Figure 16 : Liste des conteneurs en cours d'exécution prouvant la stabilité de la stack.	24
Figure 17 : Mire de connexion à l'interface Web de Zabbix.	25
Figure 18 : Accueil réussi sur le Dashboard principal de la solution.	26
Figure 19 : Confirmation du téléchargement et de l'installation réussie du paquet Zabbix Agent 2.	27
Figure 20 : Extrait du fichier de configuration montrant les directives de connexion au serveur.	28
Figure 21 : État du service affichant le statut "active (running)".	29
Figure 22 : Page de téléchargement des agents Zabbix sur le client Windows.	29
Figure 23 : Fenêtre de configuration du service Zabbix Agent 2 lors de l'installation MSI.	30
Figure 24 : Commande PowerShell créant la règle de pare-feu entrante pour le port 10050.	31
Figure 25 : Sortie PowerShell affichant le statut "Running" pour Zabbix Agent 2.	31
Figure 26 : Formulaire de configuration pour l'ajout de l'hôte Linux.	32
Figure 27 : Confirmation de la communication avec l'hôte Linux (Statut ZBX vert).	33
Figure 28 : Configuration de l'interface agent pour le serveur Windows.	33
Figure 29 : Validation de la connectivité avec l'hôte Windows.	34
Figure 30 : Vue d'ensemble de la liste des hôtes actifs dans Zabbix.	34
Figure 31 : Flux de données en temps réel provenant du client Linux.	35
Figure 32 : Métriques de performance collectées sur le client Windows.	36

Figure 33 : Historique de la charge système du client Linux.	37
Figure 34 : Analyse graphique de l'utilisation mémoire sur Windows Server.	37
Figure 35 : Alerte active en rouge signalant une surcharge CPU.	38
Figure 37: Vue d'ensemble du Dashboard "Global view" de l'infrastructure BOULAALAM.	40

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 :

- ## Dépôt GitHub

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

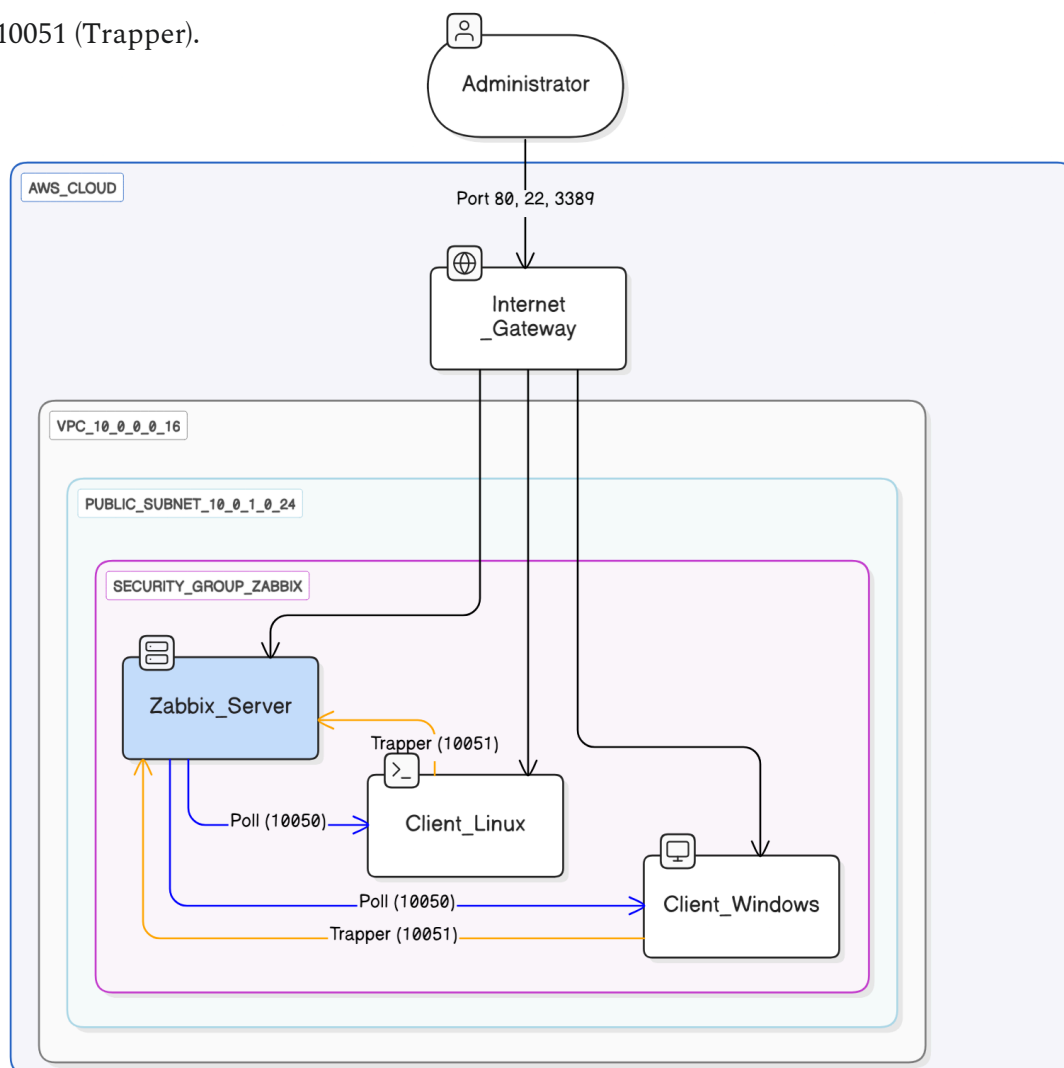
II. Architecture Réseau

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.

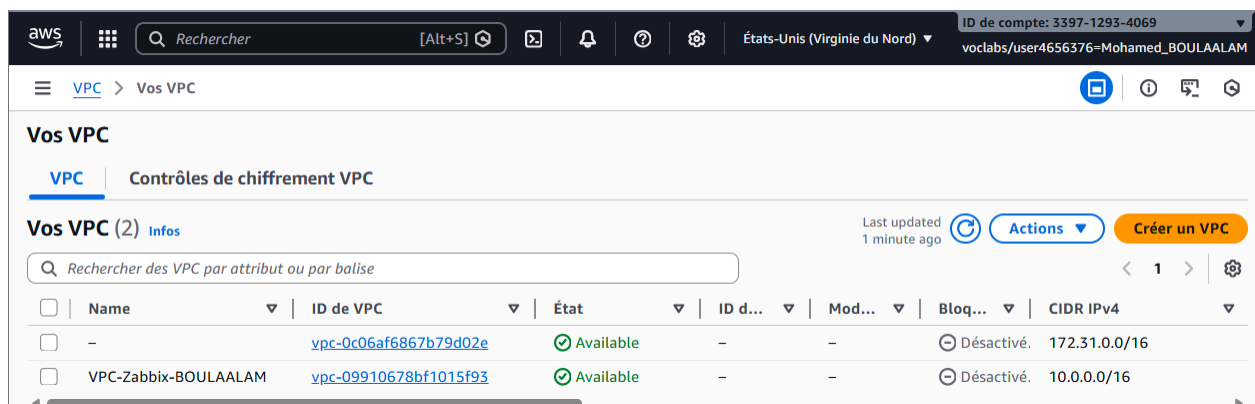


Figure 1 : Création du VPC avec le tag identifiant l'étudiant.

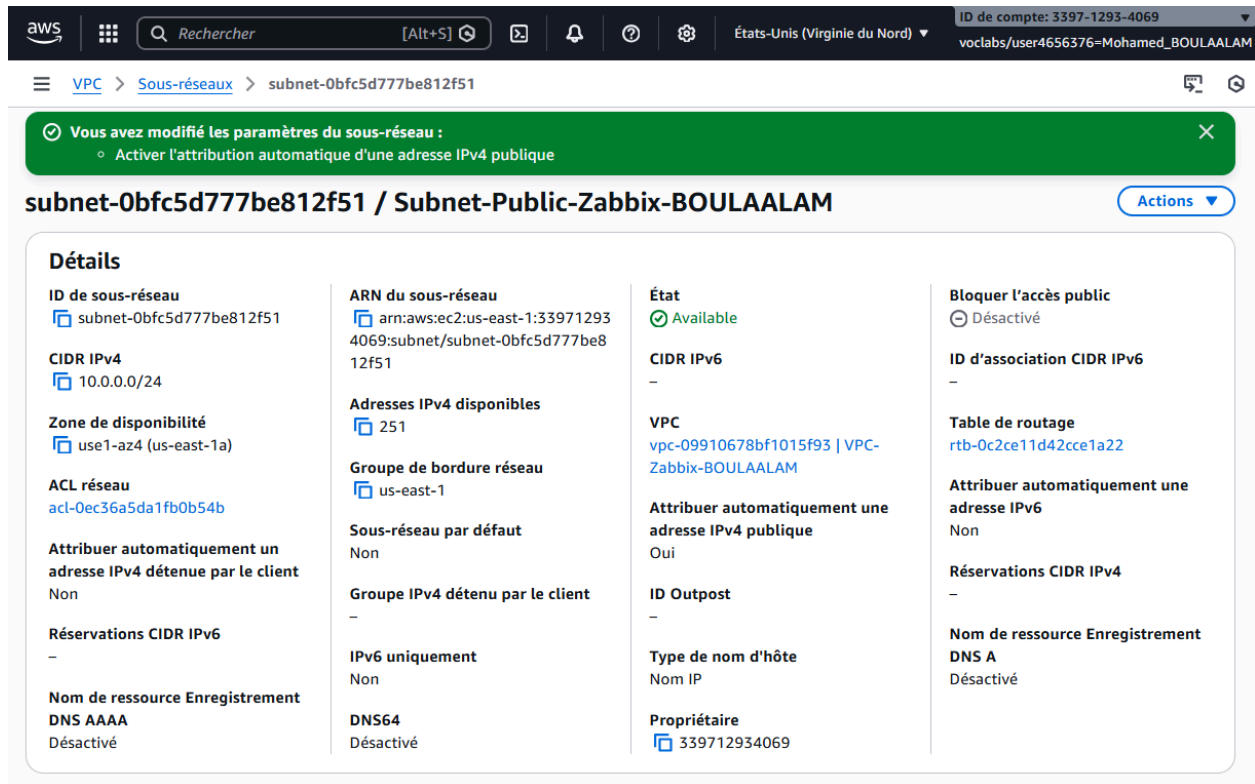


Figure 2 : Configuration du sous-réseau et activation de l'IP publique.

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).

aws [Rechercher] [Alt+S] États-Unis (Virginie du Nord) ID de compte: 3397-1293-4069 voclabs/user4656376=Mohamed_BOULAALAM

VPC > Passerelles Internet > igw-04bf42ab6c083a645

Tableau de bord du VPC

AWS Global View

Filtrer par VPC

▼ Cloud privé virtuel

- Vos VPC
- Sous-réseaux
- Tables de routage
- Passerelles Internet**
- Passerelles Internet de sortie uniquement
- Passerelles de l'opérateur
- Jeux d'options DHCP

La passerelle Internet igw-04bf42ab6c083a645 a bien été attachée à vpc-09910678bf1015f93

Notifications 0 0 2 0 0

igw-04bf42ab6c083a645 / IGW-Zabbix-BOULAALAM Actions

Détails Infos

ID de passerelle Internet igw-04bf42ab6c083a645	État Attached	ID de VPC vpc-09910678bf1015f93 / VPC-Zabbix-BOULAALAM	Propriétaire 339712934069
--	------------------	---	------------------------------

Balises (1) Gérer les balises

Rechercher des balises

Clé	Valeur
Name	IGW-Zabbix-BOULAALAM

Figure 3 : Internet Gateway attachée avec succès au VPC.

aws [Rechercher] [Alt+S] États-Unis (Virginie du Nord) ID de compte: 3397-1293-4069 voclabs/user4656376=Mohamed_BOULAALAM

VPC > Tables de routage > rtb-021c61510af386e98

rtb-021c61510af386e98 / RT-Public-Zabbix-BOULAALAM Actions

Détails Infos

ID de la table de routage rtb-021c61510af386e98	Principal Non	Associations de sous-réseau explicites subnet-0bfc5d777be812f51 / Subnet-Public-Zabbix-BOULAALAM	Associations de périphérie -
VPC vpc-09910678bf1015f93 / VPC-Zabbix-BOULAALAM	ID du propriétaire 339712934069		

Routes Associations de sous-réseau Associations de périphérie Propagation de routage Balises

Routes (2) Les deux Modifier des routes

Filtrer les routes

Destination	Cible	Statut	Propagée	Origine du routage
0.0.0.0/0	igw-04bf42ab6c0...	Actif	Non	Créer une routage
10.0.0.0/16	local	Actif	Non	Créer une table de routage

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.

The screenshot shows the AWS Management Console interface for a security group. At the top, a green banner indicates the group was created successfully. Below this, the group's details are shown in a card format, including its name, ID, description, and VPC ID. A tabbed interface allows switching between 'Règles entrantes' (selected), 'Règles sortantes', 'Partage', 'Associations VPC', and 'Balises'. The 'Règles entrantes' tab displays a table with 3 rules.

Privilège	Version IP	Type	Protocole	Plage d'adresses	Source	Description
	IPv4	TCP person...	TCP	10051	10.0.0.0/16	Communication Agents...
	IPv4	SSH	TCP	22	41.141.27.230/32	-
	IPv4	HTTP	TCP	80	0.0.0.0/0	-

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.

The screenshot displays the AWS Management Console interface for a Security Group. At the top, a green notification bar states: "Le groupe de sécurité (sg-01293663246179f32 | SG-Zabbix-Clients-BOULAALAM) a été créé avec succès." Below this, the title "sg-01293663246179f32 - SG-Zabbix-Clients-BOULAALAM" is shown with an "Actions" button. The "Détails" section provides the following information:

Détails	
Nom du groupe de sécurité SG-Zabbix-Clients-BOULAALAM	ID du groupe de sécurité sg-01293663246179f32
Description Zabbix Clients (Linux & Windows)	ID de VPC vpc-09910678bf1015f93
Propriétaire 339712934069	Nombre de règles entrantes 3 Entrées d'autorisation
	Nombre de règles sortantes 1 Entrée d'autorisation

Below the details, the "Règles entrantes" tab is selected, showing a list of three inbound rules:

Id de groupe de sécurité	Version IP	Type	Protocole	Plage d...	Source	Description
4c7d00867a	IPv4	RDP	TCP	3389	41.141.27.230/32	-
cc027a9d45	IPv4	SSH	TCP	22	41.141.27.230/32	-
997b590b1e	-	TCP person...	TCP	10050	sg-06b1ead3c025e2c96...	-

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

aws [Alt+S] États-Unis (Virginie du Nord) ID de compte: 3397-1293-4069 voclabs/user4656376=Mohamed_BOULAALAM

EC2 > Instances > i-0c3313bcd0dde8e66

Résumé de l'instance pour i-0c3313bcd0dde8e66 (Zabbix-Server-BOULAALAM) Informations

[Se connecter](#) [État de l'instance](#) [Actions](#)

Mis à jour il y a less than a minute

ID d'instance i-0c3313bcd0dde8e66	Adresse IPv4 publique 13.220.84.79 adresse ouverte	Adresses IPv4 privées 10.0.0.195
Adresse IPv6 -	État de l'instance En cours d'exécution	DNS public -
Type de nom d'hôte Nom de l'adresse IP: ip-10-0-0-195.ec2.internal	Nom DNS de l'IP privé (IPv4 uniquement) ip-10-0-0-195.ec2.internal	
Réponse à un nom DNS de ressource privée -	Type d'instance t3.large	Adresses IP élastiques -
Adresse IP attribuée automatiquement 13.220.84.79 [IP publique]	ID de VPC vpc-09910678bf1015f93 (VPC-Zabbix-BOULAALAM)	Recherche d'AWS Compute Optimizer Inscrivez-vous à AWS Compute Optimizer pour obtenir des recommandations. En savoir plus
Rôle IAM -	ID de sous-réseau subnet-0bfc5d777be812f51 (Subnet-Public-Zabbix-BOULAALAM)	Nom du groupe Auto Scaling -
IMDSv2 Required	ARN de l'instance arn:aws:ec2:us-east-1:339712934069:instance/i-0c3313bcd0dde8e66	Géré faux
Opérateur -		

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

aws [Alt+S] États-Unis (Virginie du Nord) ID de compte: 3397-1293-4069 voclabs/user4656376-Mohamed_BOULAALAM

EC2 > Instances > i-06f279a81f39bc689

Résumé de l'instance pour i-06f279a81f39bc689 (Zabbix-Client-Linux-BOULAALAM) Informations

Mis à jour il y a less than a minute

ID d'instance i-06f279a81f39bc689	Adresse IPv4 publique 13.220.187.148 adresse ouverte	Adresses IPv4 privées 10.0.0.13
Adresse IPv6 -	État de l'instance ✔ En cours d'exécution	DNS public -
Type de nom d'hôte Nom de l'adresse IP: ip-10-0-0-13.ec2.internal	Nom DNS de l'IP privé (IPv4 uniquement) ip-10-0-0-13.ec2.internal	Adresses IP élastiques -
Réponse à un nom DNS de ressource privée -	Type d'instance t3.medium	Recherche d'AWS Compute Optimizer Inscrivez-vous à AWS Compute Optimizer pour obtenir des recommandations. En savoir plus
Adresse IP attribuée automatiquement 13.220.187.148 [IP publique]	ID de VPC vpc-09910678bf1015f93 (VPC-Zabbix-BOULAALAM)	Nom du groupe Auto Scaling -
Rôle IAM -	ID de sous-réseau subnet-0bfc5d777be812f51 (Subnet-Public-Zabbix-BOULAALAM)	Géré faux
IMDSv2 Required	ARN de l'instance arn:aws:ec2:us-east-1:339712934069:instance/i-06f279a81f39bc689	
Opérateur -		

Détails | Statut et alarmes | Surveillance | Sécurité | Mise en réseau | Stockage | Billing

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

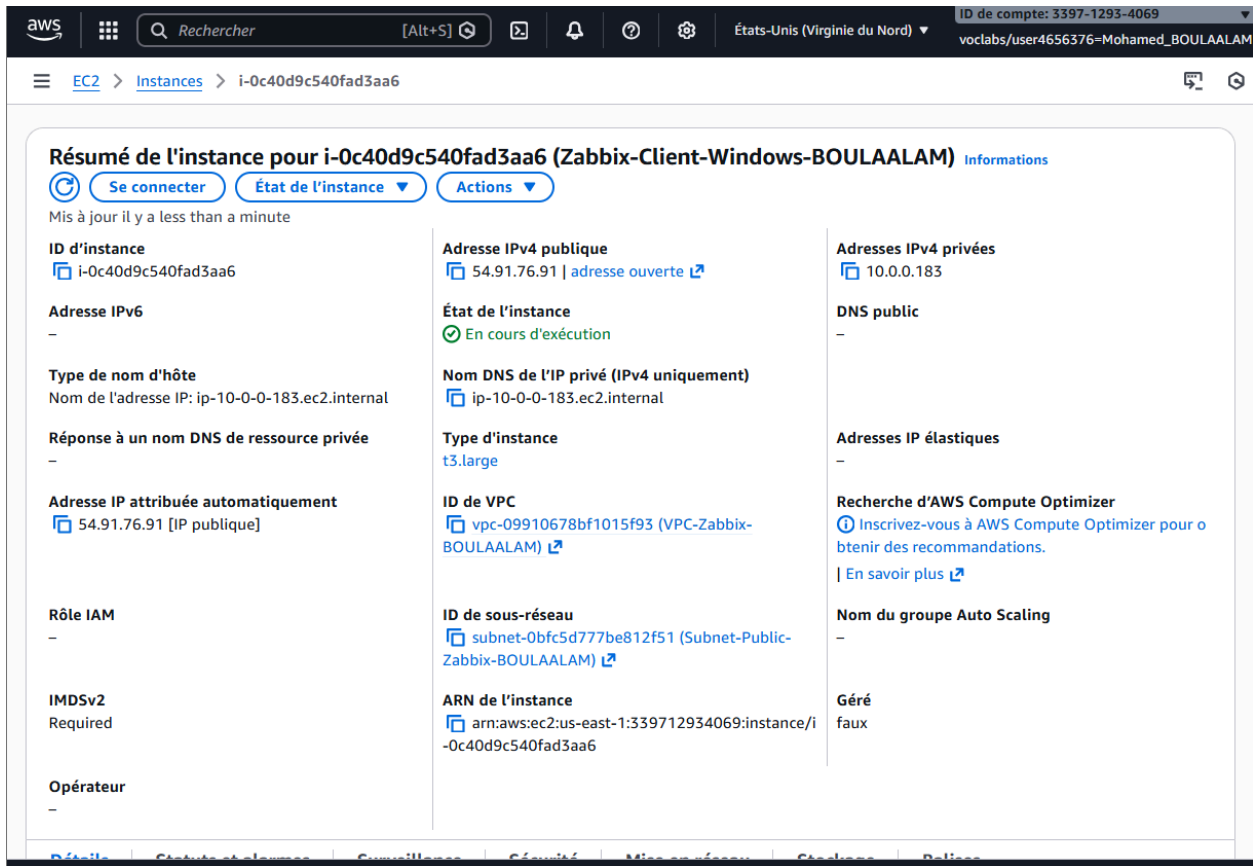


Figure 9 : Détails techniques et résumé de l'instance Client Windows.

3.2 Justification technique des types d'instances

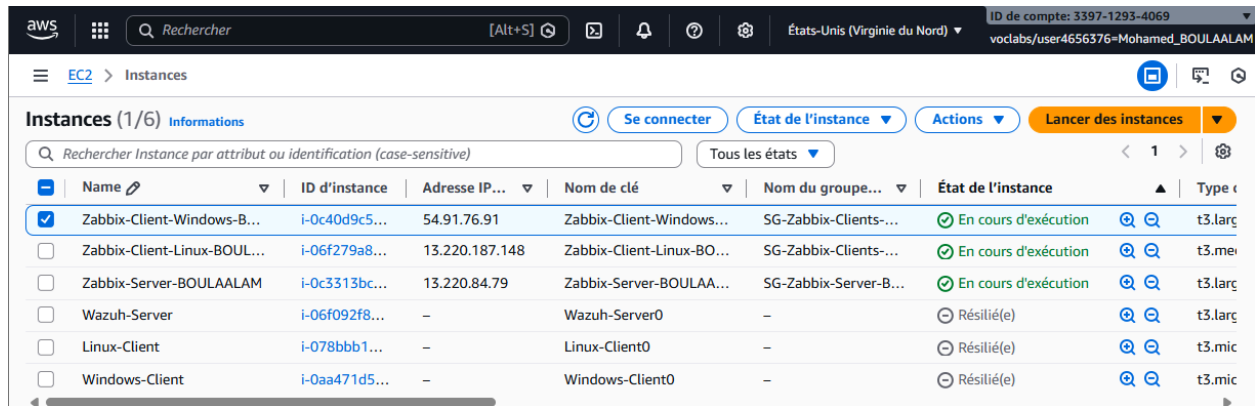
Le choix des familles et tailles d'instances répond à des exigences de performance spécifiques :

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.



	Name	ID d'instance	Adresse IP...	Nom de clé	Nom du groupe...	État de l'instance	Type
<input checked="" type="checkbox"/>	Zabbix-Client-Windows-B...	i-0c40d9c5...	54.91.76.91	Zabbix-Client-Windows...	SG-Zabbix-Clients-...	En cours d'exécution	t3.larg
<input type="checkbox"/>	Zabbix-Client-Linux-BOUL...	i-06f279a8...	13.220.187.148	Zabbix-Client-Linux-BO...	SG-Zabbix-Clients-...	En cours d'exécution	t3.me
<input type="checkbox"/>	Zabbix-Server-BOULAALAM	i-0c3313bc...	13.220.84.79	Zabbix-Server-BOULAA...	SG-Zabbix-Server-B...	En cours d'exécution	t3.larg
<input type="checkbox"/>	Wazuh-Server	i-06f092f8...	-	Wazuh-Server0	-	Résilié(e)	t3.larg
<input type="checkbox"/>	Linux-Client	i-078bbb1...	-	Linux-Client0	-	Résilié(e)	t3.mic
<input type="checkbox"/>	Windows-Client	i-0aa471d5...	-	Windows-Client0	-	Résilié(e)	t3.mic

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

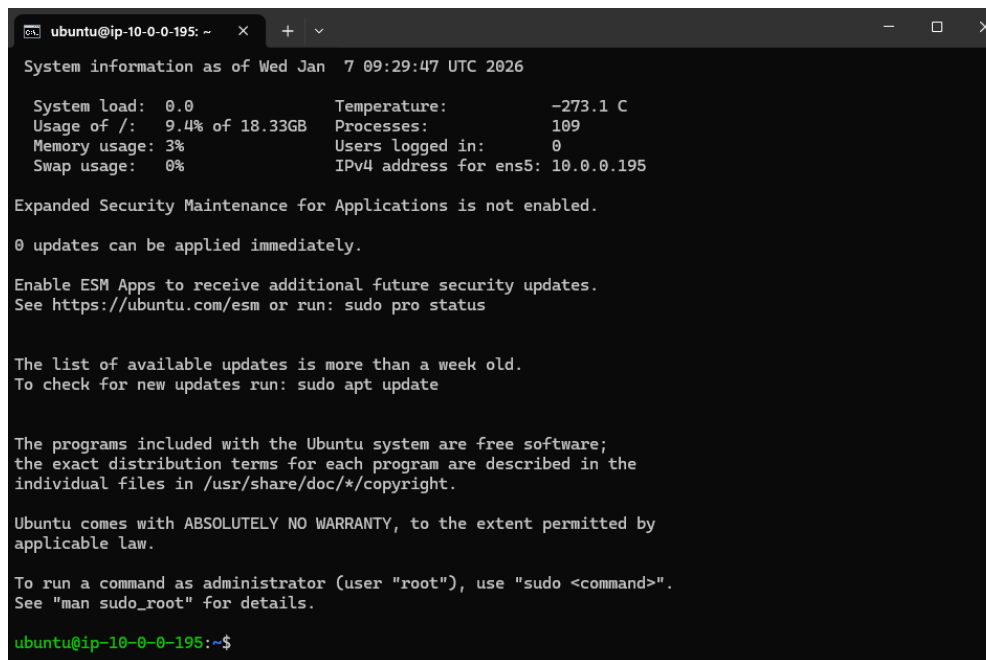
IV. Déploiement du Serveur Zabbix

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.



```
ubuntu@ip-10-0-0-195: ~  
System information as of Wed Jan 7 09:29:47 UTC 2026  
  
System load: 0.0      Temperature: -273.1 C  
Usage of /: 9.4% of 18.33GB  Processes: 109  
Memory usage: 3%      Users logged in: 0  
Swap usage: 0%        IPv4 address for ens5: 10.0.0.195  
  
Expanded Security Maintenance for Applications is not enabled.  
  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-10-0-0-195:~$
```

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.

```
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-10-0-0-195:~$ sudo curl -L "https://github.com/docker/compose/releases/latest/download
/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed
  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--    0
  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--    0
100 29.8M 100 29.8M    0     0  91.8M      0  --:--:-- --:--:-- --:--:--  91.8M
ubuntu@ip-10-0-0-195:~$ sudo chmod +x /usr/local/bin/docker-compose
ubuntu@ip-10-0-0-195:~$ docker --version
Docker version 28.2.2, build 28.2.2-0ubuntu1~24.04.1
ubuntu@ip-10-0-0-195:~$ docker-compose --version
Docker Compose version v5.0.1
ubuntu@ip-10-0-0-195:~$
```

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).

```

GNU nano 7.2                                docker-compose.yml
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-p
    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-server

```

[^]G Help [^]O Write Out [^]W Where Is [^]K Cut [^]T Execute [^]C Location
[^]X Exit [^]R Read File [^]\ Replace [^]U Paste [^]J Justify [^]/ Go To Line

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).

```
ubuntu@ip-10-0-0-195:~/zabbix-stack$ sudo docker-compose up -d
WARN[0000] /home/ubuntu/zabbix-stack/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential con
[+] up 3/26
[+] up 5/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.0s
[+] up 5/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.1s
[+] up 6/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.2s
[+] up 6/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.3s
[+] up 7/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.4s
[+] up 7/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.5s
[+] up 8/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.6s
[+] up 10/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.7s
[+] up 11/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.8s
[+] up 11/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 1.9s
[+] up 12/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.0s
[+] up 12/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.1s
[+] up 13/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.2s
[+] up 13/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.3s
[+] up 13/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.4s
[+] up 14/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.5s
[+] up 15/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.6s
[+] up 15/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.7s
[+] up 17/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.8s
[+] up 17/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 2.9s
[+] up 17/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 3.0s
[+] up 17/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 3.1s
[+] up 17/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 3.2s
[+] up 17/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 3.3s
[+] up 18/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 3.4s
[+] up 18/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 3.5s
[+] up 19/26bix/zabbix-server-mysql:ubuntu-6.4-latest [██] ] Pulling 3.6s
[+] up 19/26bix/zabbix-server-mysql:ubuntu-6.4-latest [███████] 56.57MB / 107.2MB Pulling 3.7s
[+] up 20/26bix/zabbix-server-mysql:ubuntu-6.4-latest [███████] 61.96MB / 107.2MB Pulling 3.8s
[+] up 20/26bix/zabbix-server-mysql:ubuntu-6.4-latest [███████] 69.52MB / 107.2MB Pulling 3.9s
[+] up 20/26bix/zabbix-server-mysql:ubuntu-6.4-latest [███████] 76.53MB / 107.2MB Pulling 4.0s
[+] up 20/26bix/zabbix-server-mysql:ubuntu-6.4-latest [███████] 82.47MB / 107.2MB Pulling 4.1s
[+] up 20/26bix/zabbix-server-mysql:ubuntu-6.4-latest [███████] 82.47MB / 107.2MB Pulling 4.2s
```

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

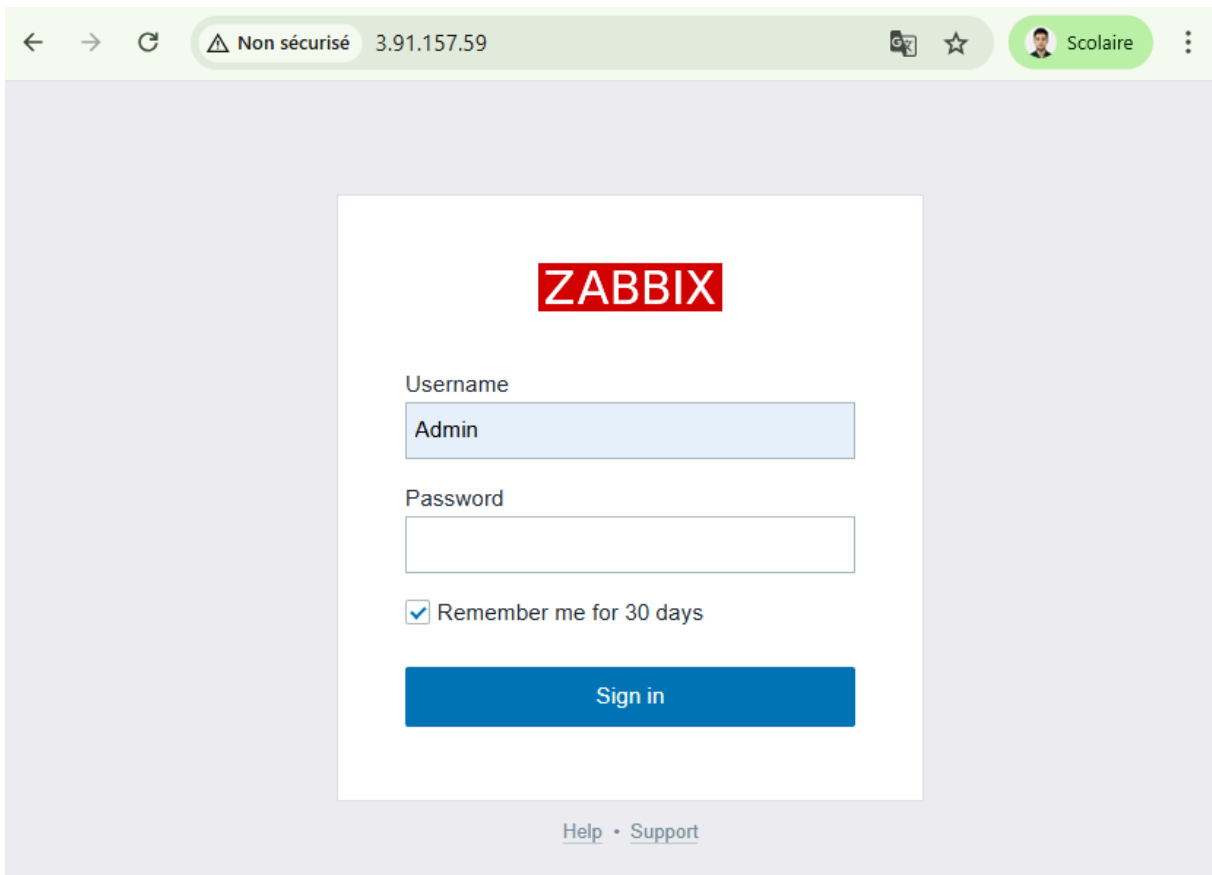
```
ubuntu@ip-10-0-0-195:~/zabbix-stack$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                                NAMES
98a85d9dbf3   zabbix/zabbix-web-nginx-mysql:ubun  "/docker-entrypoint.sh"  33 minutes ago Up 33 minutes 8003/tcp, 0.0.0.0:80->8080/tcp, [::]:80->8080/tcp  zabbix-web-ui
98a85d9dbf3   zabbix/zabbix-server-mysql:ubuntu-  "docker-entrypoint.s..."  33 minutes ago Up 33 minutes 0.0.0.0:10051->10051/tcp, [::]:10051->10051/tcp  zabbix-server
99fc8f1db41c   mysql:8.0                          "docker-entrypoint.s..."  33 minutes ago Up 33 minutes 3306/tcp, 33060/tcp  zabbix-mysql
```

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").



The screenshot shows a web browser window with the address bar displaying "Non sécurisé 3.91.157.59". The page features the ZABBIX logo at the top. Below the logo, there is a login form with the following elements:

- A "Username" label above a text input field containing "Admin".
- A "Password" label above an empty password input field.
- A checkbox labeled "Remember me for 30 days" which is checked.
- A blue "Sign in" button.
- At the bottom of the form, there are links for "Help" and "Support".

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

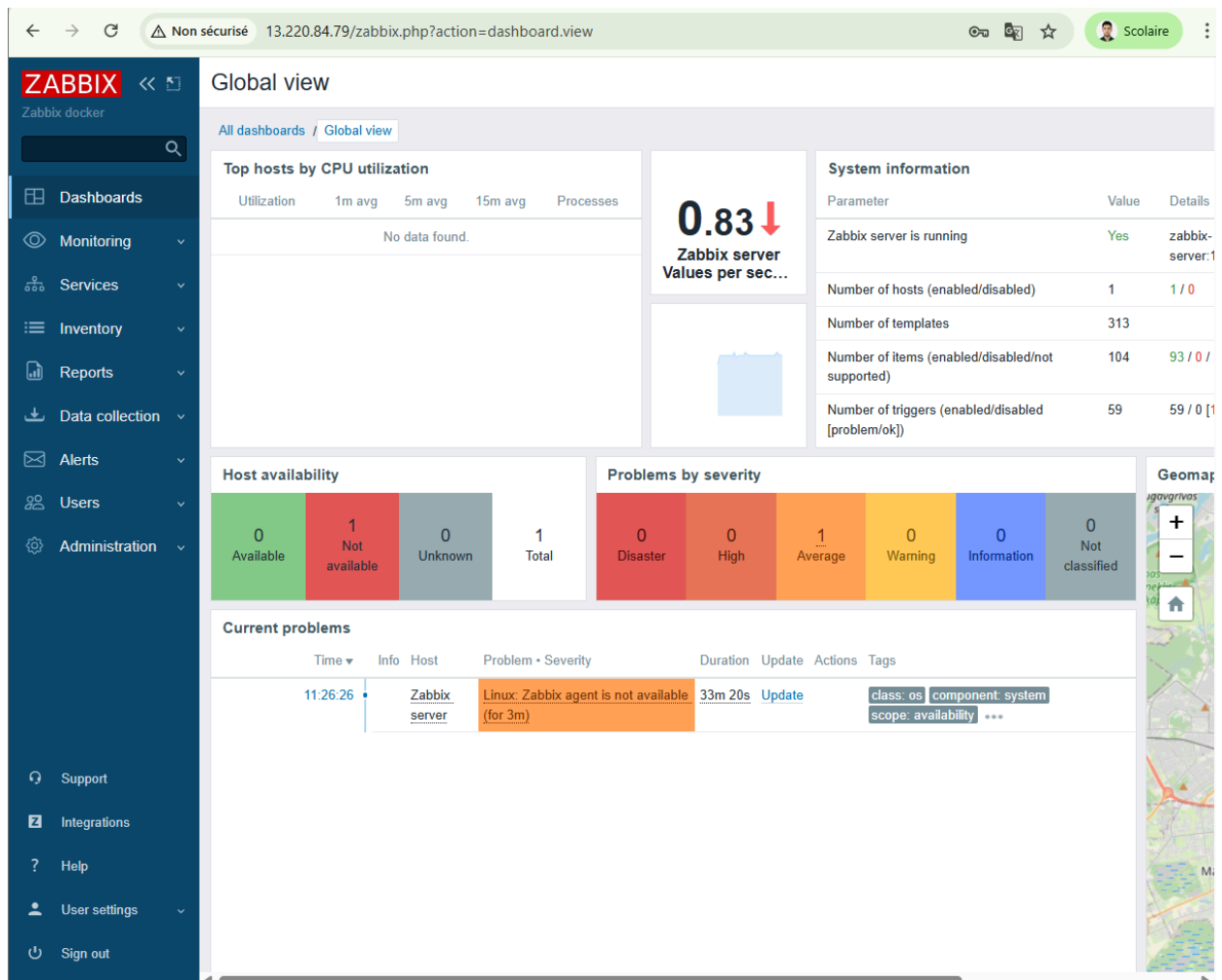


Figure 18 : Accueil réussi sur le Dashboard principal de la solution.

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`.

```
ubuntu@ip-10-0-0-13:~$ sudo apt install -y zabbix-agent2 zabbix-agent2-plugin-*
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'zabbix-agent2-plugin-mongodb' for glob 'zabbix-agent2-plugin-*'
Note, selecting 'zabbix-agent2-plugin-mssql' for glob 'zabbix-agent2-plugin-*'
Note, selecting 'zabbix-agent2-plugin-postgresql' for glob 'zabbix-agent2-plugin-*'
The following NEW packages will be installed:
  zabbix-agent2 zabbix-agent2-plugin-mongodb zabbix-agent2-plugin-mssql
  zabbix-agent2-plugin-postgresql
0 upgraded, 4 newly installed, 0 to remove and 68 not upgraded.
Need to get 13.6 MB of archives.
After this operation, 45.5 MB of additional disk space will be used.
Get:1 https://repo.zabbix.com/zabbix/6.4/ubuntu jammy/main amd64 zabbix-agent2 amd64 1:6.4.21-1+ubuntu22.04 [5350 kB]
Get:2 https://repo.zabbix.com/zabbix/6.4/ubuntu jammy/main amd64 zabbix-agent2-plugin-mongodb amd64 1:6.4.21-1+ubuntu22.04 [3297 kB]
Get:3 https://repo.zabbix.com/zabbix/6.4/ubuntu jammy/main amd64 zabbix-agent2-plugin-mssql amd64 1:6.4.21-1+ubuntu22.04 [2297 kB]
Get:4 https://repo.zabbix.com/zabbix/6.4/ubuntu jammy/main amd64 zabbix-agent2-plugin-postgresql amd64 1:6.4.21-1+ubuntu22.04 [2683 kB]
Fetched 13.6 MB in 1s (10.2 MB/s)
Selecting previously unselected package zabbix-agent2.
(Reading database ... 71741 files and directories currently installed.)
Preparing to unpack .../zabbix-agent2_1%3a6.4.21-1+ubuntu22.04_amd64.deb ...
Unpacking zabbix-agent2 (1:6.4.21-1+ubuntu22.04) ...
Selecting previously unselected package zabbix-agent2-plugin-mongodb.
Preparing to unpack .../zabbix-agent2-plugin-mongodb_1%3a6.4.21-1+ubuntu22.04_amd64.deb ...
Unpacking zabbix-agent2-plugin-mongodb (1:6.4.21-1+ubuntu22.04) ...
Selecting previously unselected package zabbix-agent2-plugin-mssql.
Preparing to unpack .../zabbix-agent2-plugin-mssql_1%3a6.4.21-1+ubuntu22.04_amd64.deb ...
Unpacking zabbix-agent2-plugin-mssql (1:6.4.21-1+ubuntu22.04) ...
Selecting previously unselected package zabbix-agent2-plugin-postgresql.
Preparing to unpack .../zabbix-agent2-plugin-postgresql_1%3a6.4.21-1+ubuntu22.04_amd64.deb ...
Unpacking zabbix-agent2-plugin-postgresql (1:6.4.21-1+ubuntu22.04) ...
Setting up zabbix-agent2-plugin-postgresql (1:6.4.21-1+ubuntu22.04) ...
Setting up zabbix-agent2-plugin-mssql (1:6.4.21-1+ubuntu22.04) ...
Setting up zabbix-agent2-plugin-mongodb (1:6.4.21-1+ubuntu22.04) ...
Setting up zabbix-agent2 (1:6.4.21-1+ubuntu22.04) ...
Created symlink /etc/systemd/system/multi-user.target.wants/zabbix-agent2.service → /usr/lib/systemd/system/zabbix-agent2.service.
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-10-0-0-13:~$
```

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`.

```

GNU nano 7.2 /etc/zabbix/zabbix_agent2.conf *
Server=10.0.0.195

### Option: ListenPort
# Agent will listen on this port for connections from the server.
#
# Mandatory: no
# Range: 1024-32767
# Default:
# ListenPort=10050

### Option: ListenIP
# List of comma delimited IP addresses that the agent should listen on.
# First IP address is sent to Zabbix server if connecting to it to retrieve list of active checks.
#
# Mandatory: no
# Default:
# ListenIP=0.0.0.0

### Option: StatusPort
# Agent will listen on this port for HTTP status requests.
#
# Mandatory: no
# Range: 1024-32767
# Default:
# StatusPort=

##### Active checks related

### Option: ServerActive
# Zabbix server/proxy address or cluster configuration to get active checks from.
# Server/proxy address is IP address or DNS name and optional port separated by colon.
# Cluster configuration is one or more server addresses separated by semicolon.
# Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma.
# More than one Zabbix proxy should not be specified from each Zabbix server/cluster.
# If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.
# Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are
# If port is not specified, default port is used.
# IPv6 addresses must be enclosed in square brackets if port for that host is specified.
# If port is not specified, square brackets for IPv6 addresses are optional.
# If this parameter is not specified, active checks are disabled.
# Example for Zabbix proxy:
# ServerActive=127.0.0.1:10051
# Example for multiple servers:
# ServerActive=127.0.0.1:20051;zabbix.domain,[::1]:30051,::1,[12fc::1]
# Example for high availability:
# ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
# Example for high availability with two clusters and one server:
# ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster2.node1;zabbix.cluster2.node2,za
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=10.0.0.195

### Option: Hostname
# List of comma delimited unique, case sensitive hostnames.
# Required for active checks and must match hostnames as configured on the server.
# Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

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 [MySQL] 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.

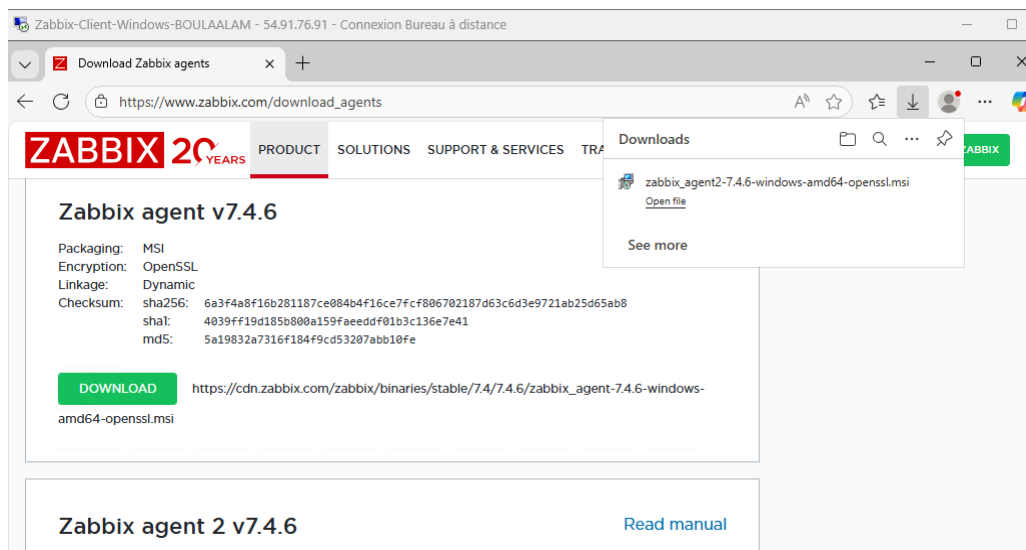


Figure 22 : Page de téléchargement des agents Zabbix sur le client Windows.

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.

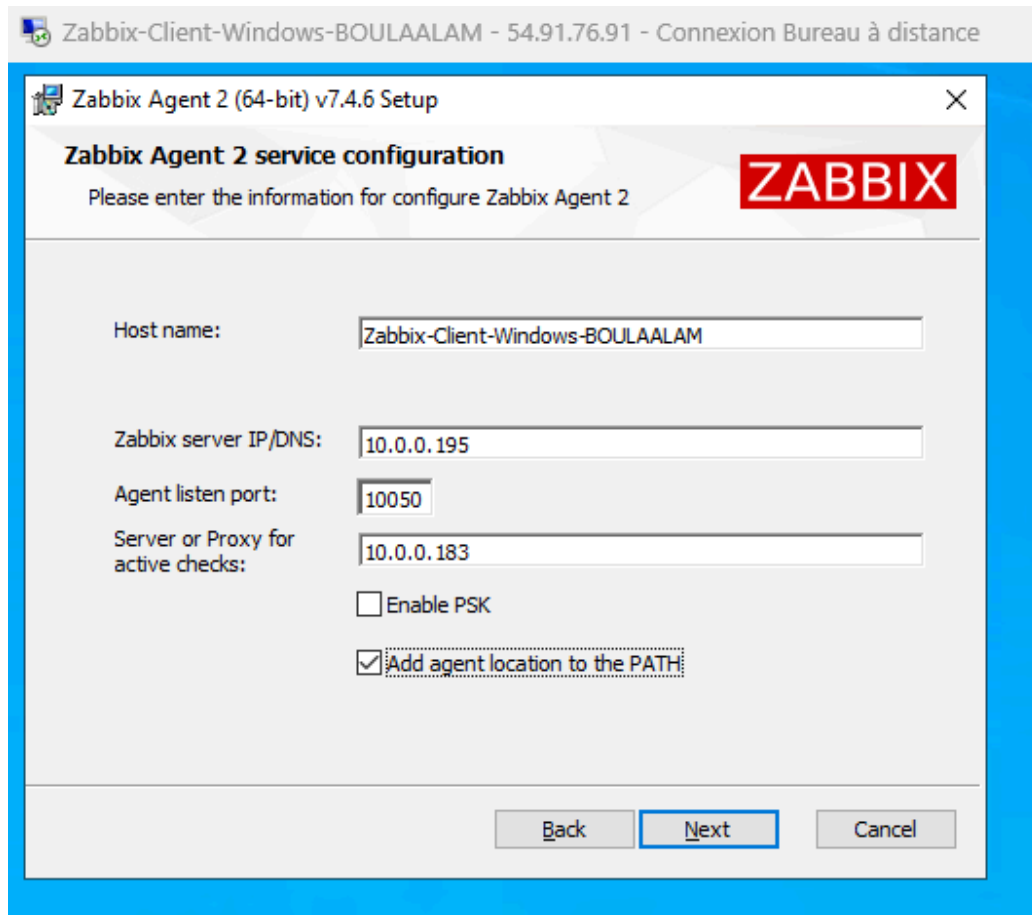
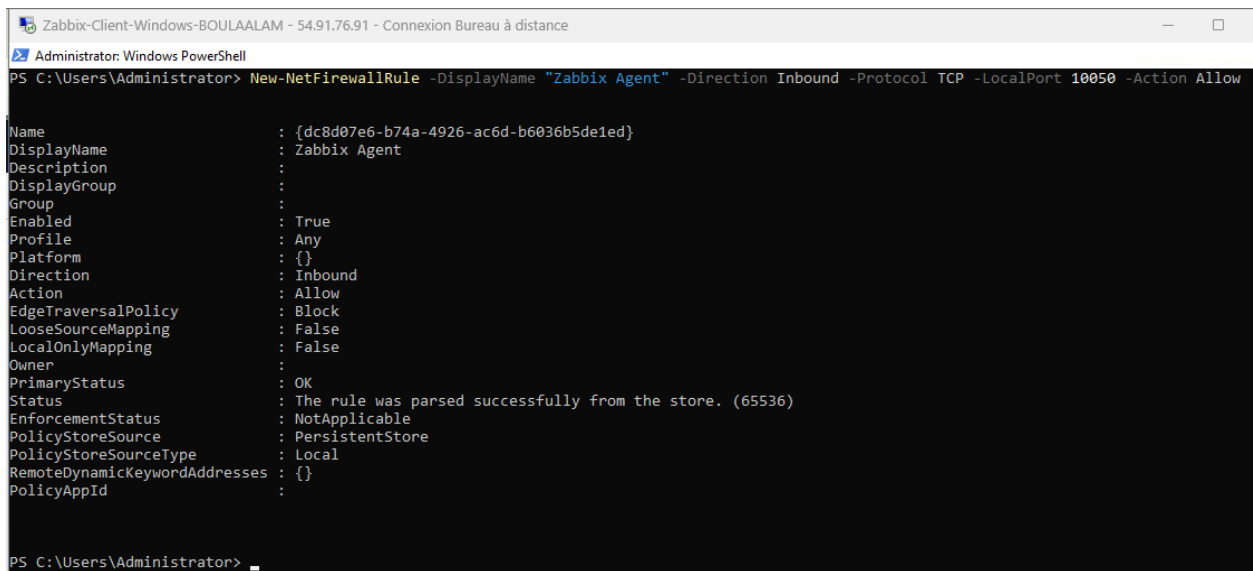


Figure 23 : Fenêtre de configuration du service Zabbix Agent 2 lors de l'installation MSI.

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.



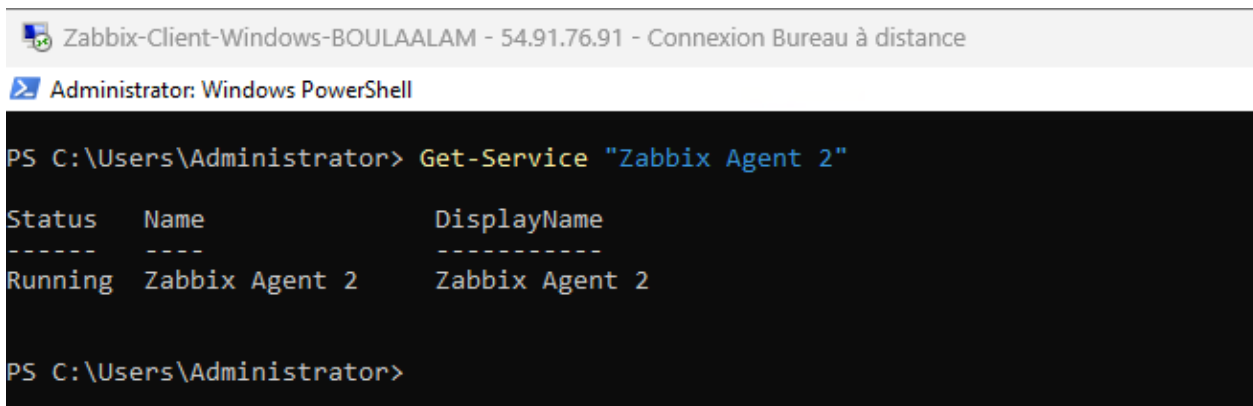
```
Zabbix-Client-Windows-BOULAALAM - 54.91.76.91 - Connexion Bureau à distance
Administrator: Windows PowerShell
PS C:\Users\Administrator> New-NetFirewallRule -DisplayName "Zabbix Agent" -Direction Inbound -Protocol TCP -LocalPort 10050 -Action Allow

Name : {dc8d07e6-b74a-4926-ac6d-b6036b5de1ed}
DisplayName : Zabbix Agent
Description :
DisplayGroup :
Group :
Enabled : True
Profile : Any
Platform : {}
Direction : Inbound
Action : Allow
EdgeTraversalPolicy : Block
LooseSourceMapping : False
LocalOnlyMapping : False
Owner :
PrimaryStatus : OK
Status : The rule was parsed successfully from the store. (65536)
EnforcementStatus : NotApplicable
PolicyStoreSource : PersistentStore
PolicyStoreSourceType : Local
RemoteDynamicKeywordAddresses : {}
PolicyAppId :

PS C:\Users\Administrator>
```

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.



```
Zabbix-Client-Windows-BOULAALAM - 54.91.76.91 - Connexion Bureau à distance
Administrator: Windows PowerShell

PS C:\Users\Administrator> Get-Service "Zabbix Agent 2"

Status      Name              DisplayName
-----
Running     Zabbix Agent 2    Zabbix Agent 2

PS C:\Users\Administrator>
```

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.

The screenshot shows the Zabbix web interface with the 'New host' form. The browser address bar shows '13.220.84.79/zabbix.php?action=host.edit'. The Zabbix logo and 'Zabbix docker' are in the top left. The left sidebar contains navigation links: Dashboard, Monitoring, Services, Inventory, Reports, Data collection, Template groups, Host groups, Templates, Hosts, Maintenance, Event correlation, Discovery, Alerts, Users, and Administration. The 'New host' form has tabs for Host, IPMI, Tags, Macros, Inventory, Encryption, and Value mapping. The 'Host' tab is selected. The form fields are: Host name (Zabbix-Client-Linux-BOULAALAM), Visible name (Zabbix-Client-Linux-BOULAALAM), Templates (Linux by Zabbix agent), Host groups (Linux servers), Interfaces (Agent, 10.0.0.13, Connect to IP, Port 10050, Default Remove), Description (empty), Monitored by proxy (no proxy), and Enabled (checked). Buttons for 'Add' and 'Cancel' are at the bottom right.

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

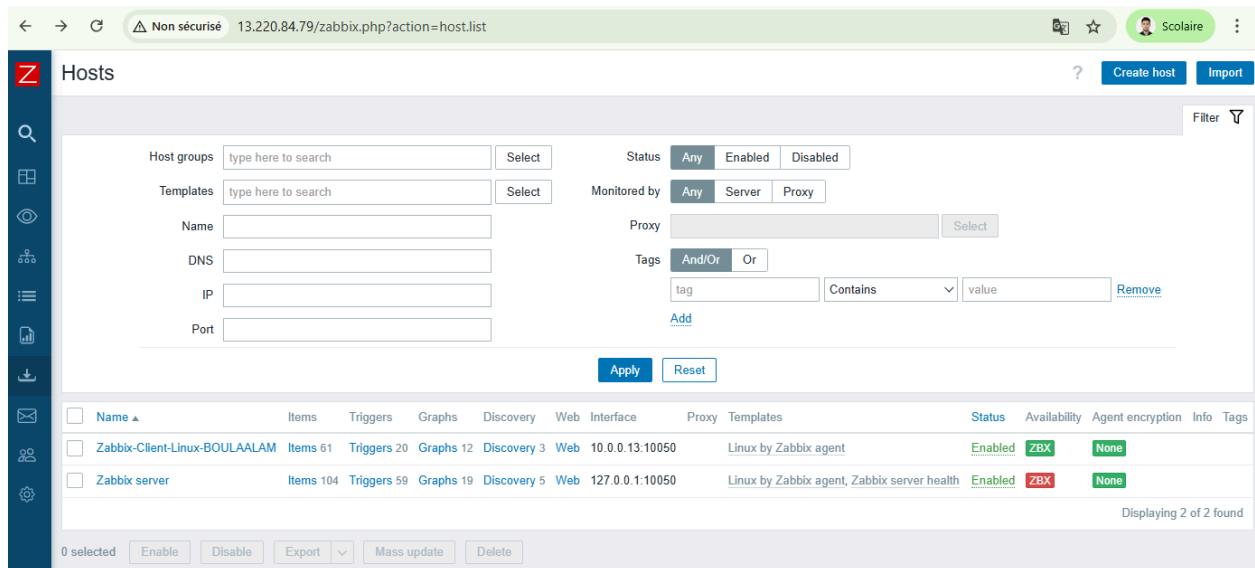


Figure 27 : Confirmation de la communication avec l'hôte Linux (Statut ZBX vert).

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.

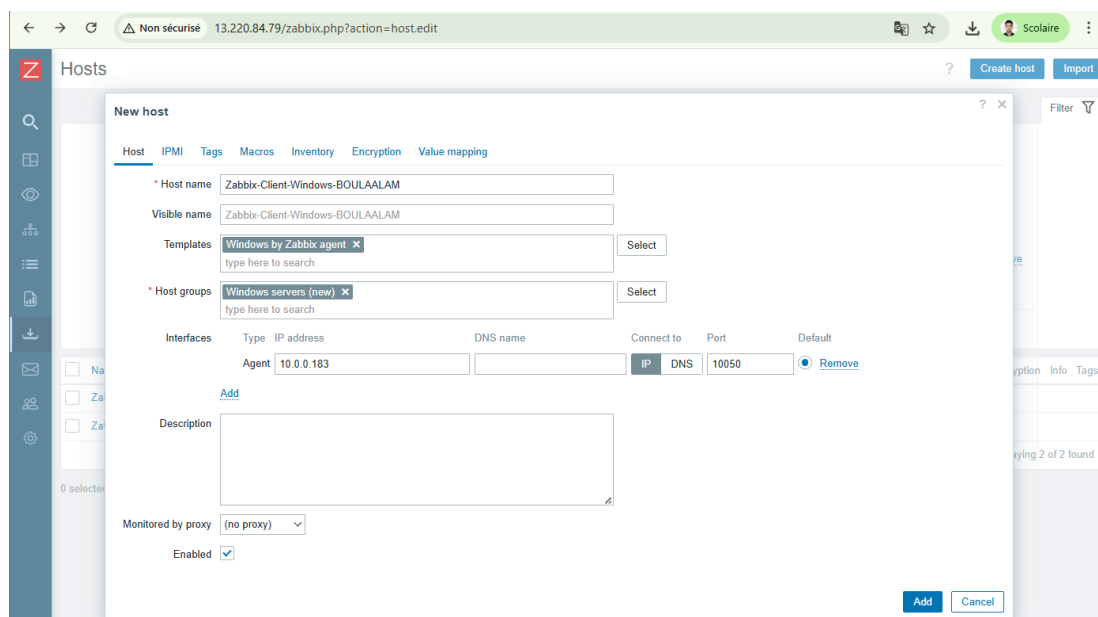


Figure 28 : Configuration de l'interface agent pour le serveur Windows.

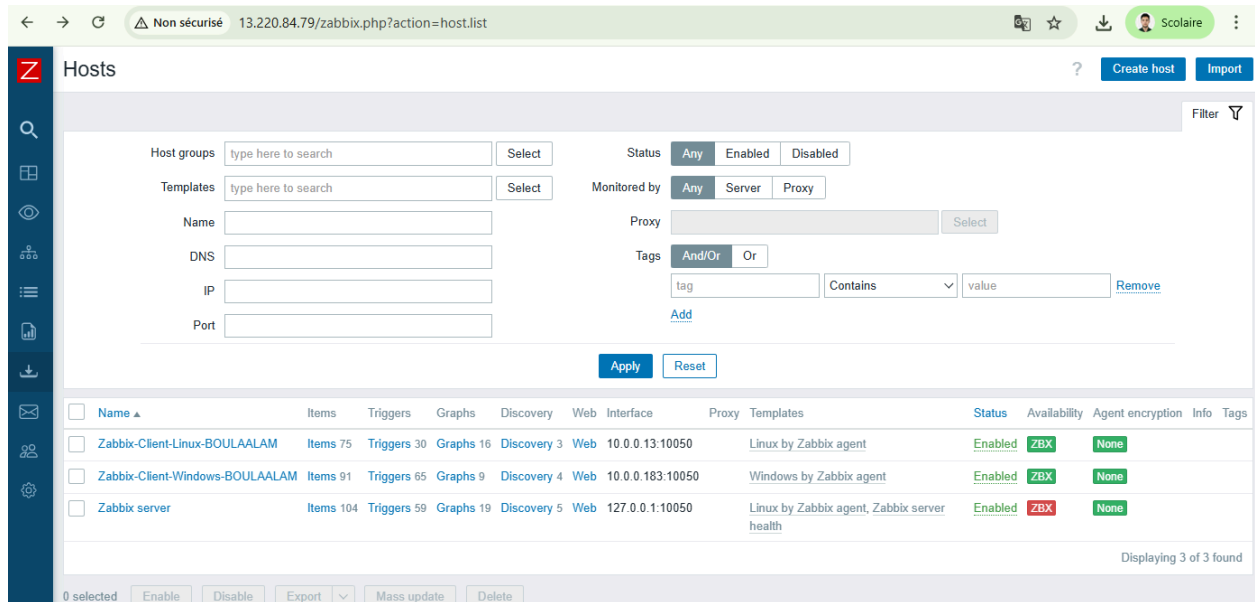


Figure 29 : Validation de la connectivité avec l'hôte Windows.

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

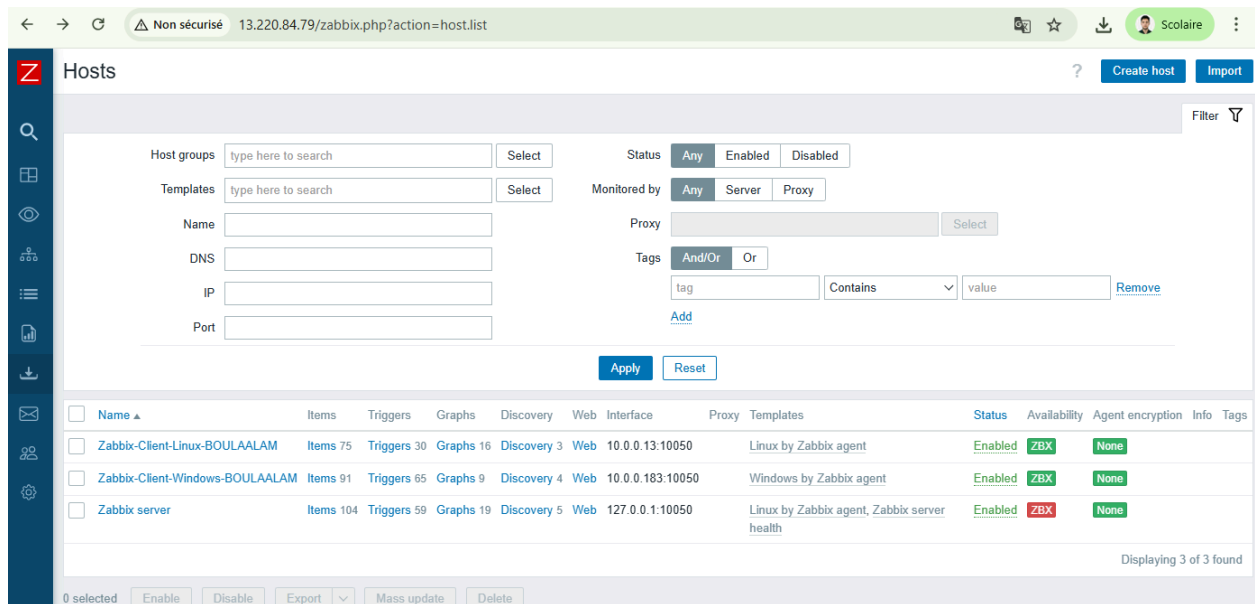


Figure 30 : Vue d'ensemble de la liste des hôtes actifs dans Zabbix.

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.

The screenshot shows the Zabbix web interface with the 'Latest data' view selected for the host 'Zabbix-Client-Linux-BOULAALAM'. The interface includes a sidebar with navigation options like Dashboards, Monitoring, Problems, Hosts, Latest data, Maps, Discovery, Services, Inventory, Reports, Data collection, Alerts, Users, Administration, Support, Integrations, Help, User settings, and Sign out.

The main content area displays the following information:

- Host groups:** type here to search
- Hosts:** Zabbix-Client-Linux-BOULAALAM
- Name:** type here to search
- Tags:** And/Or Or tag Contains value
- Show tags:** None 1 2 3 Tag name Full Shortened None
- Tag display priority:** comma-separated list
- Show details:** ☐
- Subfilter affects only filtered data**
- HOSTS:** Zabbix-Client-Linux-BOULAALAM 18
- TAGS:** component 18
- TAG VALUES:** component: application +1 cpu 17 environment +1 memory +7 network +9 os +3 raw +4 security 1 storage +25 system +12
- disk:** nvme0n1 +8
- filesystem:** / +7 /boot +7
- fstype:** ext4 +14
- interface:** ens5 +9
- DATA:** With data Without data

The table below shows the latest data points for the host:

Host	Name	Last check	Last value	Change	Tags
Zabbix-Client-Linux-B...	Linux: Checksum of /etc/passwd	31m 27s	0d1c138ff32e87e...		component: security
Zabbix-Client-Linux-B...	Linux: Context switches per second	49s	112.9834	+2.2259	component: cpu
Zabbix-Client-Linux-B...	Linux: CPU guest nice time	47s	0 %		component: cpu
Zabbix-Client-Linux-B...	Linux: CPU guest time	48s	0 %		component: cpu
Zabbix-Client-Linux-B...	Linux: CPU idle time	46s	99.8833 %	-0.008301 %	component: cpu
Zabbix-Client-Linux-B...	Linux: CPU interrupt time	45s	0 %		component: cpu
Zabbix-Client-Linux-B...	Linux: CPU iowait time	44s	0 %		component: cpu

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

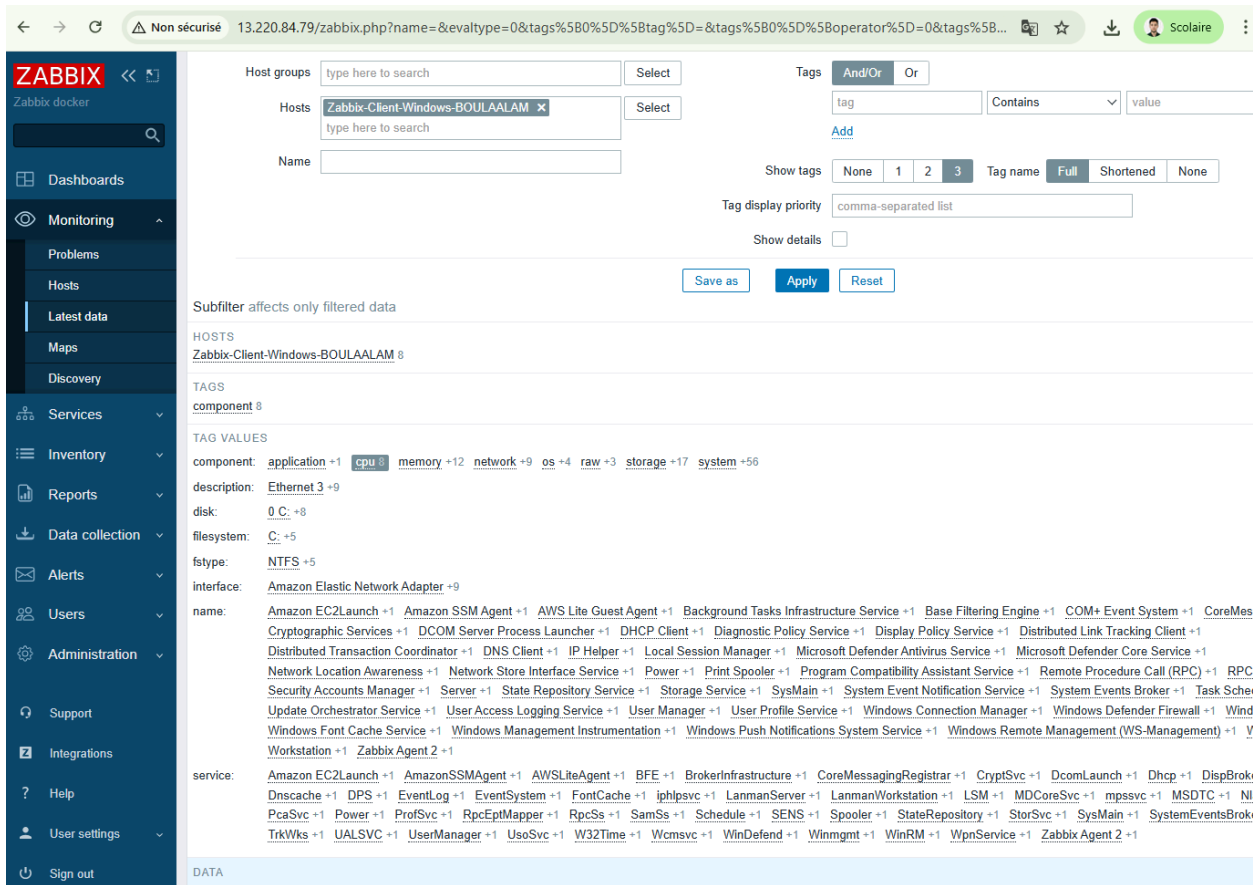


Figure 32 : Métriques de performance collectées sur le client Windows.

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.

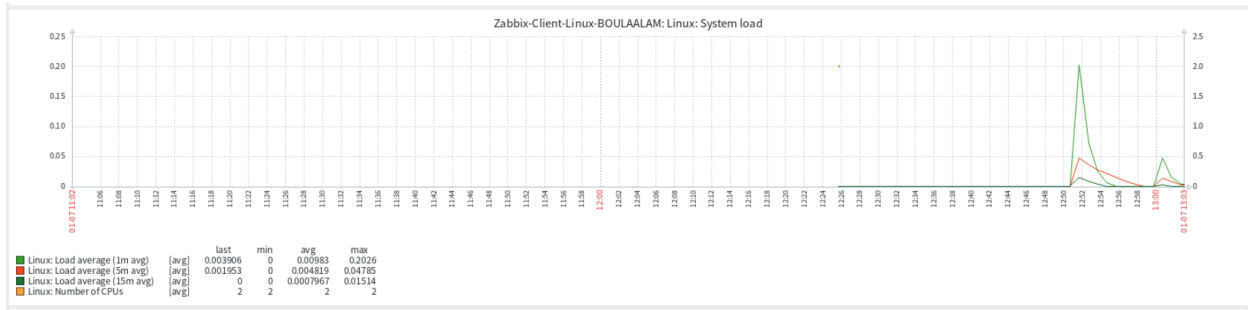


Figure 33 : Historique de la charge système du client Linux.

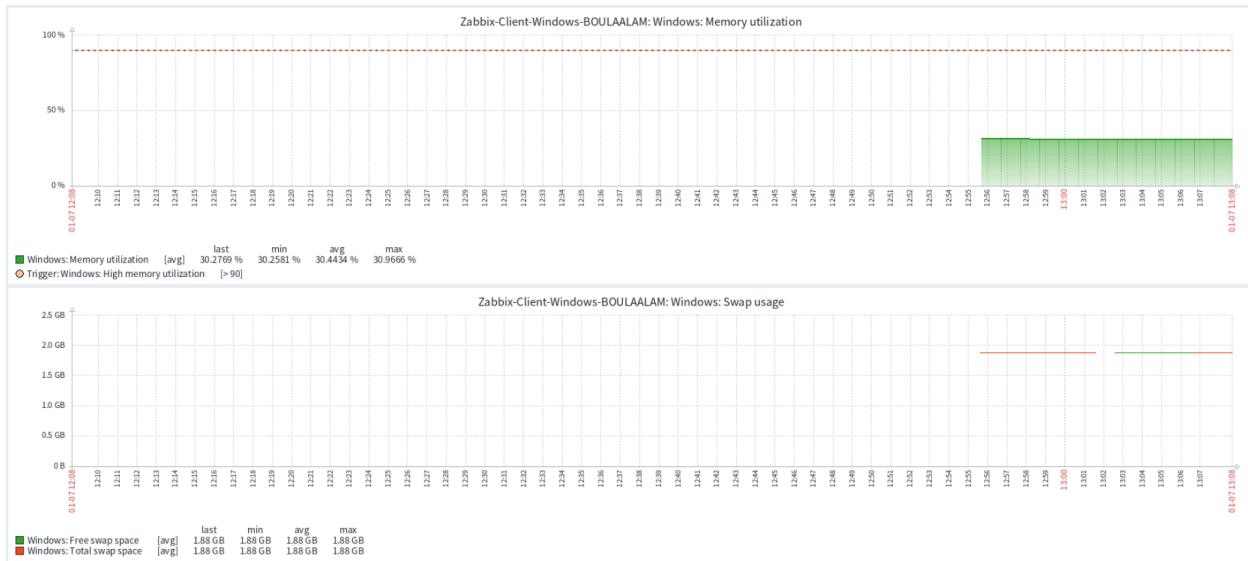


Figure 34 : Analyse graphique de l'utilisation mémoire sur Windows Server.

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.

```
ubuntu@ip-10-0-0-13:~$ dd if=/dev/zero of=/dev/null &
[2] 3039
ubuntu@ip-10-0-0-13:~$
```

```
ubuntu@ip-10-0-0-13:~$ stress --cpu 4 --timeout 120
stress: info: [3016] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

```
ubuntu@ip-10-0-0-13:~$ stress --cpu 4 --timeout 600
stress: info: [3055] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

Problems

Host groups: Linux servers x

Hosts: Zabbix-Client-Linux-BOULAALAM x

Triggers: type here to search

Severity: ☐ Not classified ☐ Warning ☐ High ☐ Information ☐ Average ☐ Disaster

Age less than: 14 days

Show symptoms: ☐

Show suppressed problems: ☐

Show unacknowledged only: ☐

Host inventory: Type Remove

Tags: Does not contain value Remove Contains value Remove

Show tags: None 1 2 3 Tag name: Full Shortened None

Tag display priority: comma-separated list

Show operational data: None Separately With problem name

Compact view: ☐ Show timeline: ☒

Show details: ☐ Highlight whole row: ☐

Save as Apply Reset

Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Update	Actions	Tags
13:22:40	Average		PROBLEM		Zabbix-Client-Linux-BOULAALAM	Linux: Load average is too high (per CPU load over 1.5 for 5m)	3m 39s	Update		class: os component: cpu scope: capacity

Displaying 1 of 1 found

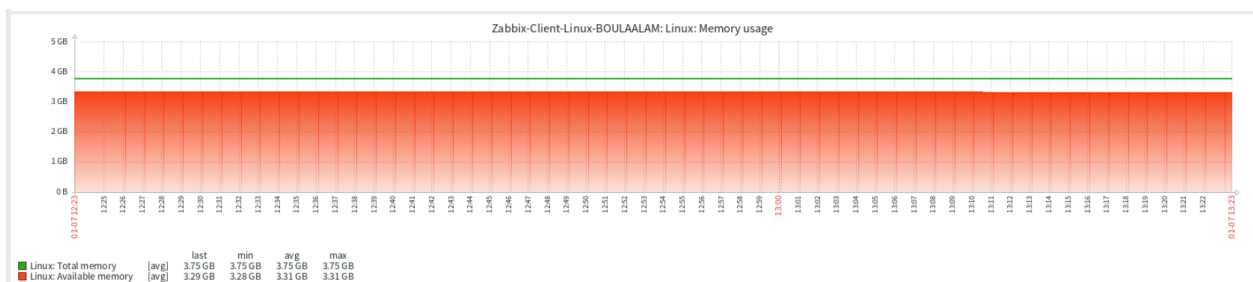
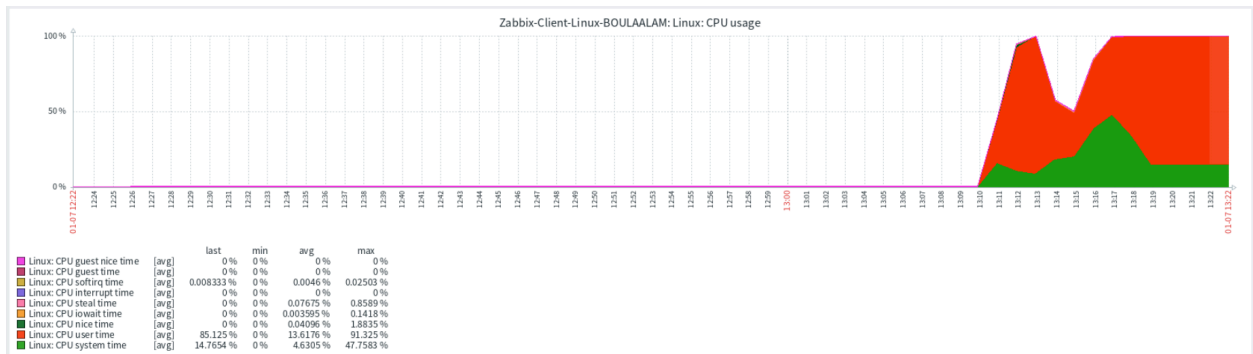


Figure 35 : Alerte active en rouge signalant une surcharge CPU.

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.

The image shows the Zabbix configuration interface for creating a new action. The top section, titled "Operation details", includes fields for "Operation" (Send message), "Steps" (1 - 1), "Step duration" (0), and "Send to user groups" (type here to search). It also has a "Send to users" field with a dropdown menu showing "Admin (Zabbix Administrator)". Below this is a "Send only to" dropdown set to "- All -". There are checkboxes for "Custom message" and "Conditions" (Label, Name). The bottom section, titled "New action", includes a "Name" field with the value "Alerte Critique - BOULAALAM", a "Conditions" field with a dropdown menu showing "Add", and an "Enabled" checkbox checked. There are also checkboxes for "Pause operations for symptom problems", "Pause operations for suppressed problems", and "Notify about canceled escalations". The bottom right corner has "Add" and "Cancel" buttons.

Operation details

Operation: Send message

Steps: 1 - 1 (0 - infinitely)

Step duration: 0 (0 - use action default)

* At least one user or user group must be selected.

Send to user groups: type here to search

Send to users: Admin (Zabbix Administrator) x

Send only to: - All -

Custom message: ☐

Conditions: Label Name

New action

Action: Operations 1

* Name: Alerte Critique - BOULAALAM

Conditions: Label Name Action

Enabled: ☒

* At least one operation must exist.

Buttons: Add Cancel

Trigger actions

0 selected Enable Disable Delete

Name	Conditions	Operations	Status
Alerte Critique - BOULAALAM		Send message to users: Admin (Zabbix Administrator) via all media	Enabled
Report problems to Zabbix administrators		Send message to user groups: Zabbix administrators via all media	Disabled

Displaying 2 of 2 found

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.

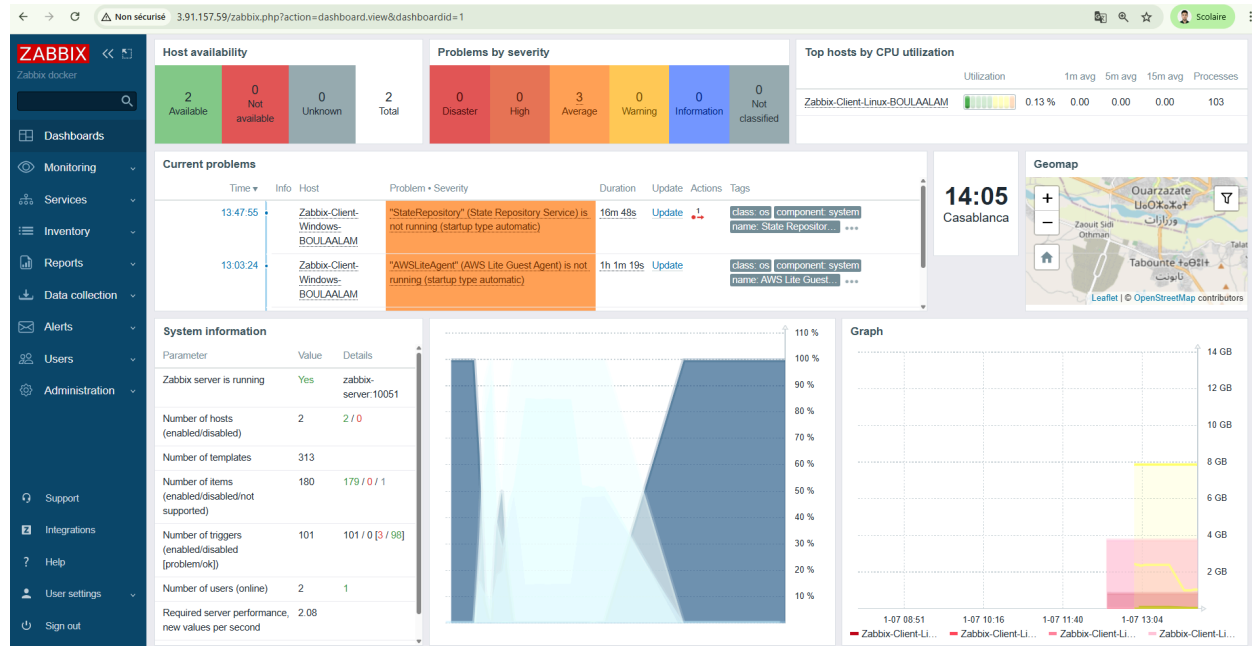


Figure 37: Vue d'ensemble du Dashboard "Global view" de l'infrastructure BOULAALAM.

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.