

# Ingénierie Informatique, Intelligence Artificielle Et Cybersécurité

## Projet Zabbix :

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



Réalisé par :

MAKOSSO Dylan  
Elohim Vianney

Encadré par :

Pr. Azeddine KHIAT

Année universitaire  
2025/2026

## **Table des matières :**

Introduction.....	4
1    Architecture Réseau .....	5
1.1    Configuration du VPC et des Sous-réseaux.....	5
1.1.1    Crédation du réseau virtuel (Projet_VPC_Zabbix) avec bloc CIDR : 10.0.0.0/16.....	5
1.1.2    Crédation du sous réseau virtuel (Subnet_Projet_VPC_Zabbix) au sein du même VPC avec un CIDR 10.0.0.0/20.....	5
1.2    Configuration de la Sécurité (Security Groups) .....	6
1.2.1    Paramétrage du Security Group Zabbix-SG-Supervision.....	6
1.2.2    Sécurisation des flux : Groupe de sécurité (Security Group) .....	6
1.3    Architecture des Instances EC2 .....	7
1.3.1    Tableau Rédactif du Parc Informatique.....	7
1.3.2    Procédure de configuration et de lancement .....	7
1.3.3    Validation du déploiement .....	8
2    Déploiement du Serveur Zabbix : Installation et Configuration du Serveur Zabbix.....	9
2.1    Installation de Docker et Docker-Compose.....	9
2.2    Configuration de l'Orchestration (Docker Compose).....	10
2.2.1    Structure du fichier de déploiement .....	10
2.2.2    Sécurisation et variables d'environnement.....	11
2.3    Lancement et Vérification de la Stack Zabbix .....	12
Cette étape automatise la création de l'ensemble de l'architecture (Base de données, Serveur et Web). .....	12
2.3.1    Commande de déploiement.....	12
2.3.2    Contrôle de l'état des services (Vérification) .....	12
2.3.3    Interface de Connexion Zabbix .....	13
2.3.4    Tableau de bord et Etat du Système .....	13
3    Configuration des Clients (Agents Zabbix) .....	14
3.1    Installation et configuration de l'agent sur le client Linux.....	14
3.1.1    Phase d'installation .....	14
3.1.2    Configuration du fichier zabbix_agentd.conf .....	15
3.1.3    Finalisation et persistance .....	16
3.1.4    Ajout de l'hôte dans l'interface web Zabbix .....	16
3.1.5    Résolution de l'erreur d'accès (Access Permissions) .....	16
3.2    Extension de la supervision : Installation de l'agent sur Windows .....	17
3.2.1    Déploiement de l'agent MSI.....	17
3.2.2    Configuration du flux réseau et du host .....	19

3.2.3	Diagnostic : Résolution de l'erreur de connexion.....	19
3.2.4	Validation du service .....	20
3.3	Auto-supervision du serveur Zabbix .....	21
3.3.1	Identification réseau du conteneur .....	21
3.3.2	État final de la supervision.....	21
4	Monitoring et Tableaux de bord.....	22
4.1	Centralisation des métriques (Global Infrastructure Monitoring).....	22
4.1.1	Graphique de charge CPU.....	22
4.1.2	Graphique d'utilisation RAM .....	22
4.2	Alertes Proactives et Synthèse Visuelle.....	23
4.2.1	Mise en place d'un Trigger (Alerte Proactive).....	23
4.2.2	Synthèse Visuelle du Dashboard Consolidé.....	23
	Conclusion .....	25
	Perspectives .....	25



# Introduction :

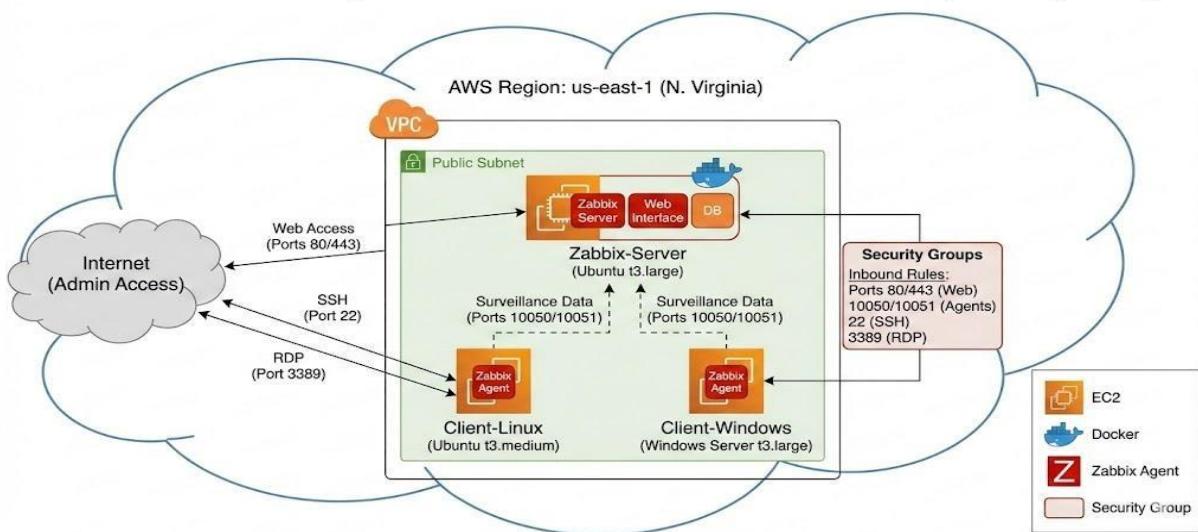
Dans un contexte où les infrastructures informatiques deviennent de plus en plus complexes et critiques, la supervision en temps réel des systèmes représente un élément essentiel pour garantir la disponibilité, la performance et la continuité des services. Ce projet vise ainsi à concevoir et déployer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

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

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

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

**Infrastructure de Supervision Centralisée Zabbix sur AWS (Parc Hybride)**



# 1 Architecture Réseau :

Elle détaille la configuration de l'environnement réseau sécurisé nécessaire au bon fonctionnement de la supervision.

## 1.1 Configuration du VPC et des Sous-réseaux :

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

### 1.1.1 Crédit du réseau virtuel (Projet\_VPC\_Zabbix) avec bloc CIDR : 10.0.0.0/16

The screenshot shows the AWS VPC console with the following details:

- VPC Details:**
  - ID de VPC: vpc-0fd1e2a99c9abb824
  - Etat: Available
  - Résolution DNS: Activé
  - ACL réseau principal: acl-0a07b1802fd6dadd6
  - CIDR IPv6 (groupe de bordure réseau): -
- Network Settings:**
  - Bloquer l'accès public: Désactivé
  - Location: default
  - VPC par défaut: Non
  - Métriques d'utilisation d'adresses réseau: Désactivé
- Networking Options:**
  - Jeu d'options DHCP: dopt-0238062e5ae9ec3ff
  - Noms d'hôte DNS: Désactivé
  - Table de routage principale: rtb-0b70232cf571a703
  - Groupe IPv6: -
  - ID du propriétaire: 253173567236

**Actions enregistrées (1):**

- Type: Tous les t...
- Opérations: VPC / Create Vpc

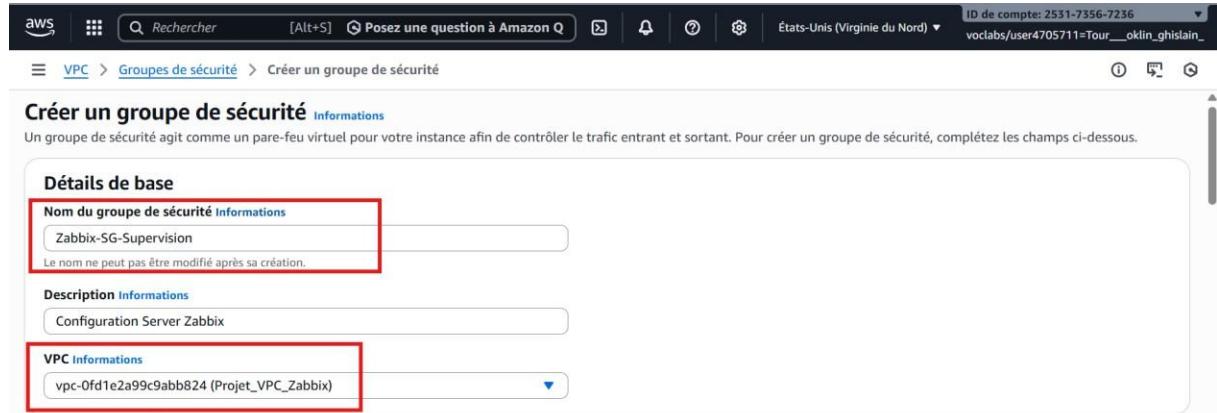
### 1.1.2 Crédit du sous réseau virtuel :(Subnet\_Projet\_VPC\_Zabbix) au sein du même VPC avec un CIDR 10.0.0.0/20

The screenshot shows the AWS VPC console with the following details:

ID de sous-réseau	Name	ID de VPC	Bloquer l'ac...	CIDR IPv4	Zone de disponibilité
subnet-08803e89821fe3d4	Subnet_Projet_VPC_Zabbix	vpc-0fd1e2a99c9abb824	Désactivé	10.0.0.0/20	use1-az2 (us-east-1a)

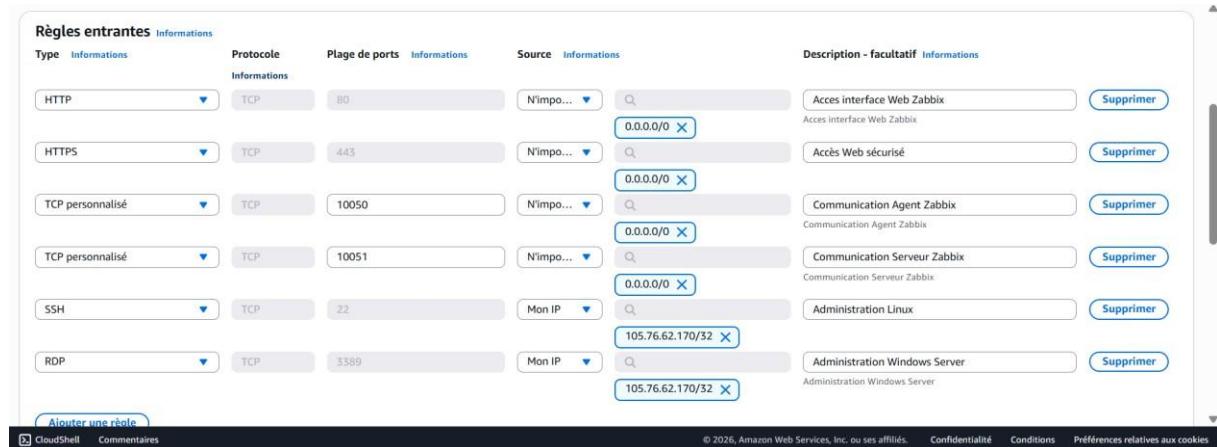
## 1.2 Configuration de la Sécurité :(Security Groups)

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



The screenshot shows the AWS VPC Security Groups creation interface. The 'Nom du groupe de sécurité' (Name) field is highlighted with a red box. The field contains 'Zabbix-SG-Supervision'. Below it, a note says 'Le nom ne peut pas être modifié après sa création.' (The name cannot be changed after creation.). The 'Description' field contains 'Configuration Server Zabbix'. The 'VPC Informations' section shows 'vpc-0fd1e2a99c9abb824 (Projet\_VPC\_Zabbix)' selected. The top navigation bar shows 'ID de compte: 2531-7356-7236' and 'États-Unis (Virginie du Nord)'.

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



The screenshot shows the AWS VPC Security Group rules interface. It lists several inbound rules:

Type	Protocole	Plage de ports	Source	Description - facultatif
HTTP	TCP	80	N'importe qui	Accès interface Web Zabbix
HTTPS	TCP	443	N'importe qui	Accès Web sécurisé
TCP personnalisé	TCP	10050	N'importe qui	Communication Agent Zabbix
TCP personnalisé	TCP	10051	N'importe qui	Communication Serveur Zabbix
SSH	TCP	22	Mon IP	Administration Linux
RDP	TCP	3389	Mon IP	Administration Windows Server

At the bottom, there are buttons for 'Ajouter une règle' (Add rule), 'CloudShell', 'Commentaires' (Comments), and links to '© 2026, Amazon Web Services, Inc. ou ses affiliés.', 'Confidentialité', 'Conditions', and 'Préférences relatives aux cookies'.

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

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

## **1.3 Architecture des Instances EC2 :**

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

### **1.3.1 Tableau Récapitulatif du Parc Informatique :**

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

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

### **1.3.2 Procédure de configuration et de lancement :**

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

#### **1.3.2.1 Sélection du système et du type d'instance :**

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

### 1.3.2.2 Configuration réseau et sécurité :

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

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

### 1.3.3 Validation du déploiement :

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

Instances (4) informations														
État de l'instance		État de l'instance par attribut ou identification (case-sensible)		Tous les états		Actions		État de l'instance		Actions		Gérer des instances		
État de l'instance = running	X	Effacer les filtres												
TP-Sécurité-R...	v	ID d'instance	En cours d... 12.micro	Type d'insta... t2.micro	Contrôle des stats 2/2 vérifications	Statut d'alarme Afficher les alarm	Zone de dispon... us-east-1a	DNS IPv4 public ec2-98-93-197-201.co...	Adresse IPv4... 98.93.197.201	IP élastique -	Adresses IP1... -	Surveillance disabled	Nom du groupe de s... launch-wizard-1	Nom de clé tp-ubuntu-key
Client-Windo...	v	i-0309d9593c52fbf804	En cours d... 13.large	t2.micro	3/3 vérifications	Afficher les alarm	us-east-1a	-	35.173.249.16	-	-	disabled	Zabbix-SG-Supervision	tp-ubuntu-key
Zabbix-Server	v	i-05c6d45d9865c652	En cours d... 13.large	t2.micro	3/3 vérifications	Afficher les alarm	us-east-1a	-	3.91.194.29	-	-	disabled	Zabbix-SG-Supervision	tp-ubuntu-key
Client-linux-Z...	v	i-094156c23f06d3b42	En cours d... 13.medium	t2.micro	3/3 vérifications	Afficher les alarm	us-east-1a	-	98.84.137.145	-	-	disabled	Zabbix-SG-Supervision	tp-ubuntu-key

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

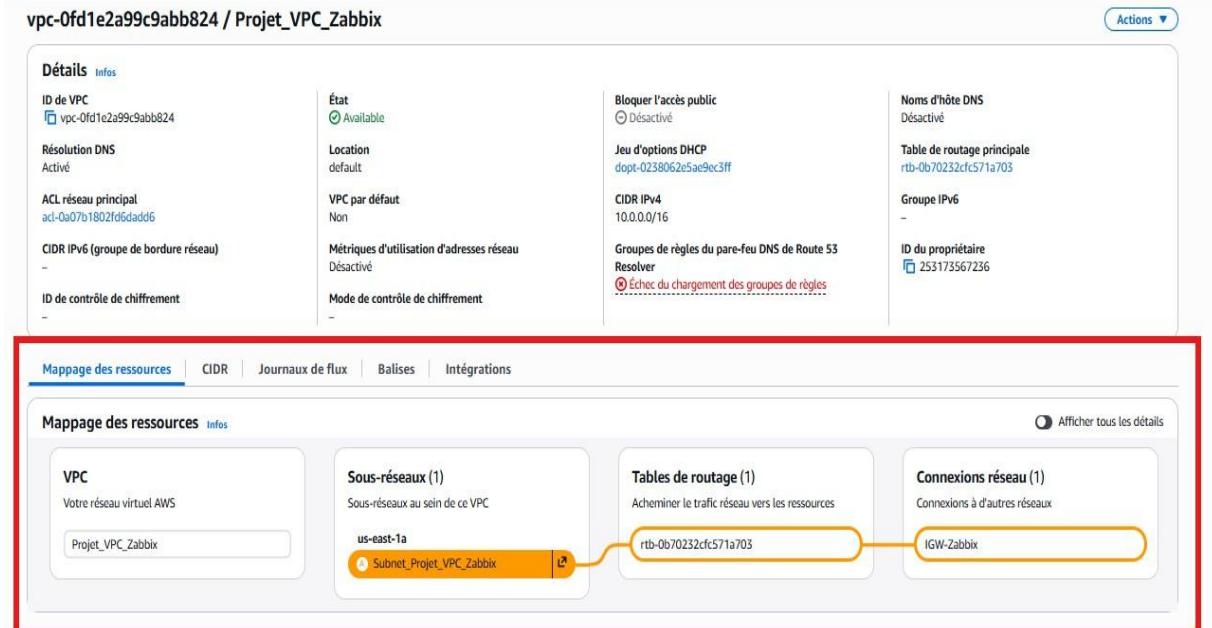
Cette phase consiste à transformer notre instance Ubuntu vierge en un serveur de supervision opérationnel grâce à Docker.

### 2.1 Installation de Docker et Docker-Compose :

Avant de procéder à l'installation de Docker, une phase de diagnostic réseau a été nécessaire pour établir la connexion avec nos instances.

Initialement, la connexion SSH vers l'instance **Zabbix-Server** était impossible (erreur *Connection Timeout*). L'analyse du mappage des ressources a révélé que le VPC était isolé, sans accès vers l'extérieur. Pour rétablir la connectivité, les actions suivantes ont été entreprises :

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



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

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

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

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

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

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

## 2.2 Configuration de l'Orchestration (Docker Compose)

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

### 2.2.1 Structure du fichier de déploiement :

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

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

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

```

ubuntu@ip-10-0-4-89: ~/zabt  X  +  -
GNU nano 7.2
services:
  zabbix-db:
    image: mysql:8.0
    container_name: zabbix-db
    restart: always
    command: [mysqld,]
      --character-set-server=utf8,
      --collation-server=utf8_bin,
      --default-authentication-plugin=mysql_native_password,
      --log-bin_trust_function_creators=1
    environment:
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - MYSQL_DATABASE=${MYSQL_DATABASE}
    volumes:
      - ./zabbix-db-data:/var/lib/mysql
  zabbix-server:
    image: zabbix/zabbix-server-mysql:ubuntu-6.4-latest
    container_name: zabbix-server
    restart: always
    ports:
      - "10051:10051"
    environment:
      - DB_SERVER_HOST=zabbix-db
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
      - MYSQL_DATABASE=${MYSQL_DATABASE}
    depends_on:
      - zabbix-db
  zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql:ubuntu-6.4-latest
    container_name: zabbix-web
    restart: always
    ports:
      - "80:8080"
    environment:
      - ZBX_SERVER_HOST=zabbix-server
      - DB_SERVER_HOST=zabbix-db
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
      - MYSQL_DATABASE=${MYSQL_DATABASE}
      - ZBX_PHP_TZ=${ZBX_PHP_TZ}
    depends_on:
      - zabbix-db
      - zabbix-server

```

## 2.2.2 Sécurisation et variables d'environnement :

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

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

```

GNU nano 7.2
# Configuration Base de données
MYSQL_ROOT_PASSWORD=password-root
MYSQL_USER=zabbix
MYSQL_PASSWORD=password-user
MYSQL_DATABASE=zabbix

# Configuration System
ZBX_PHP_TZ=Africa/Casablanca

```

## **2.3 Lancement et Vérification de la Stack Zabbix :**

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

### **2.3.1 Commande de déploiement :**

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

```
ubuntu@ip-10-0-4-89:~/zabbix$ sudo docker-compose up -d
Creating network "zabbix_default" with the default driver
Pulling zabbix-db (mysql:8.0)...
8.0: Pulling from library/mysql
658e67031dba: Pull complete
c6664f4b: Pull complete
c9ae3e76dfc9: Pull complete
cc3a161f487a: Pull complete
af8a12fa4e1: Pull complete
Digest: sha256:9c3380eac945af0736031b200027f581925927c81e010056214a4bd6b6693714
Status: Downloaded newer image for mysql:8.0
Pulling zabbix-server (zabbix/zabbix-server-mysql:ubuntu-6.4-latest)...
ubuntu-6.4-latest: Pulling from zabbix/zabbix-server-mysql
de44b265507a: Pull complete
c15ab064eeaa3: Pull complete
a2ab853cdf6c: Pull complete
5f562f77cda: Pull complete
66367a2dd2d: Pull complete
210f7f100111: Pull complete
4f4fb700ef54: Pull complete
037d1acdde55: Pull complete
Digest: sha256:55d074b6b031c8fb274e6f9321344b41839284e618940c90c51a1b4883523932
Status: Downloaded newer image for zabbix/zabbix-server-mysql:ubuntu-6.4-latest
Pulling zabbix-web (zabbix/zabbix-web-nginx-mysql:ubuntu-6.4-latest)...
ubuntu-6.4-latest: Pulling from zabbix/zabbix-web-nginx-mysql
de44b265507a: Already exists
64acbf469e79: Pull complete
8ea225815046: Pull complete
ef724024751d: Pull complete
4f4fb700ef54: Pull complete
```

#### **Analyse de la commande :**

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

### **2.3.2 Contrôle de l'état des services (Vérification)**

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

```
ubuntu@ip-10-0-4-89:~/zabbix$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
74bbe66e632b        zabbix/zabbix-web-nginx-mysql:ubuntu-6.4-latest   "docker-entrypoint.sh"   5 minutes ago      Up 5 minutes (healthy)   84
43/tcp, 0.0.0.0:80->8080/tcp, [::]:80->8080/tcp   zabbix-web
8b91c19ddea        zabbix/zabbix-server-mysql:ubuntu-6.4-latest   "/usr/bin/tini -- /u..."  5 minutes ago      Up 5 minutes           0.
0.0.0.0:10051->10051/tcp, [::]:10051->10051/tcp   zabbix-server
c60fd2eb7314        mysql:8.0          "docker-entrypoint.s..."  5 minutes ago      Up 5 minutes           33
06/tcp, 33060/tcp,  zabbix-db
```

#### **Conformité validés :**

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

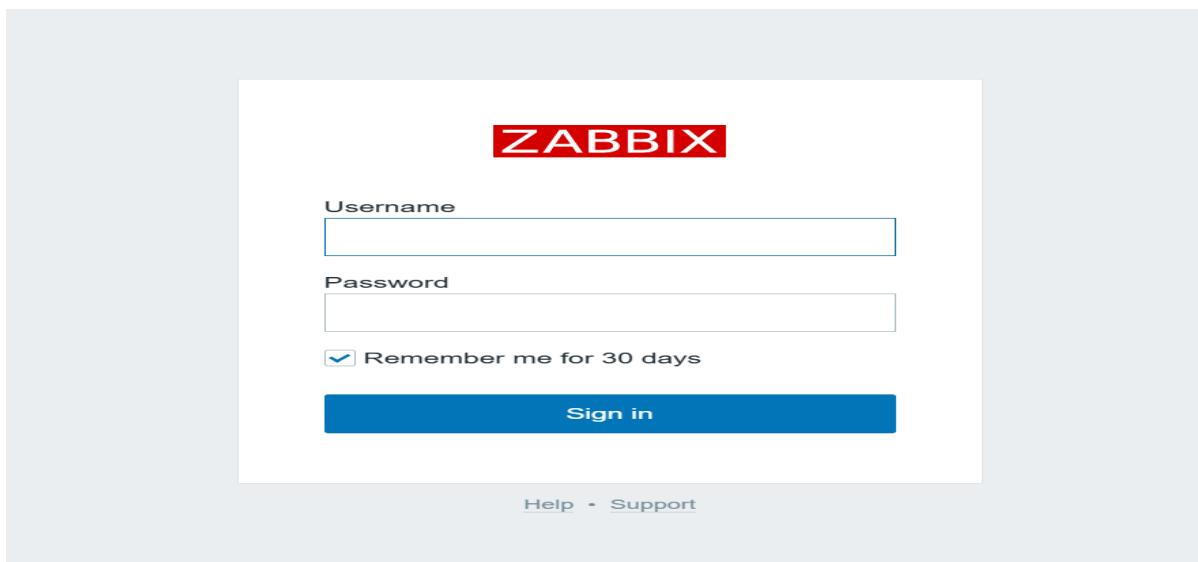
## 2. Mappage des Ports :

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

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

### 2.3.3 Interface de Connexion Zabbix :

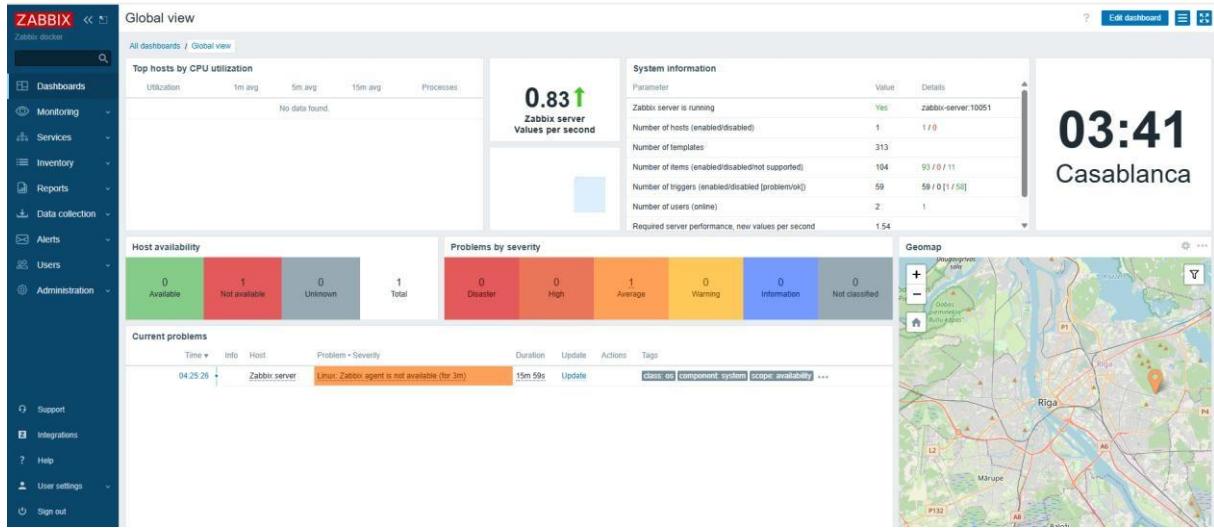
Une fois le déploiement Docker terminé, l'interface est devenue accessible via l'IP publique. Pour cette première phase, nous avons utilisé les identifiants d'administration par défaut.



### 2.3.4 Tableau de bord et Etat du Système :

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

- **Disponibilité du Serveur :** L'indicateur "Zabbix server is running" affiche **Yes**, prouvant que le conteneur zabbix-server communique correctement avec l'interface Web et la base de données MySQL.
- **Localisation et Heure :** Le fuseau horaire Africa/Casablanca configuré dans le fichier .env est correctement appliqué (03:41).
- **Performance :** Le système traite déjà des valeurs par seconde (0.83), indiquant que l'auto-supervision du serveur est active.



### 3 Configuration des Clients (Agents Zabbix)

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

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

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

##### 3.1.1 Phase d'installation

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

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

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

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

```

ubuntu@ip-10-0-11-126:~$ wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.4-1+ubuntu22.04_all.deb
--2026-01-29 01:16:18-- https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.4-1+ubuntu22.04_all.deb
Resolving repo.zabbix.com (repo.zabbix.com)... 178.128.6.101, 2604:a880:2:d0::2062:d001
Connecting to repo.zabbix.com[178.128.6.101]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3744 (3.7K) [application/octet-stream]
Saving to: 'zabbix-release_6.4-1+ubuntu22.04_all.deb'

zabbix-release_6.4-1+ubuntu22.04_all.deb 100%[=====] 3.66K --.-KB/s in 0s

2026-01-29 01:16:18 (758 MB/s) - 'zabbix-release_6.4-1+ubuntu22.04_all.deb' saved [3744/3744]

ubuntu@ip-10-0-11-126:~$ sudo dpkg -i zabbix-release_6.4-1+ubuntu22.04_all.deb
Selecting previously unselected package zabbix-release.
(Reading database ... 71752 files and directories currently installed.)
Preparing to unpack zabbix-release_6.4-1+ubuntu22.04_all.deb ...
Unpacking zabbix-release (1:6.4-1+ubuntu22.04) ...
Setting up zabbix-release (1:6.4-1+ubuntu22.04) ...
ubuntu@ip-10-0-11-126:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 https://repo.zabbix.com/zabbix/6.4/ubuntu jammy InRelease [2883 B]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 https://repo.zabbix.com/zabbix/6.4/ubuntu jammy/main Sources [23.8 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:11 https://repo.zabbix.com/zabbix/6.4/ubuntu jammy/main all Packages [12.7 kB]

```

```

6 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-10-0-11-126:~$ sudo apt install zabbix-agent -y
Reading package lists... done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libmodbus5
The following NEW packages will be installed:
  libmodbus5 zabbix-agent
0 upgraded, 2 newly installed, 0 to remove and 6 not upgraded.
Need to get 331 kB of archives.
After this operation, 931 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 libmodbus5 amd64 3.1.10-1ubuntu1 [34.4 kB]
Get:2 https://repo.zabbix.com/zabbix/7.0/ubuntu noble/main amd64 zabbix-agent amd64 1:7.0.22-1+ubuntu24.04 [296 kB]
Fetched 331 kB in 1s (595 kB/s)
Selecting previously unselected package libmodbus5:amd64.
(Reading database ... 71762 files and directories currently installed.)
Preparing to unpack .../libmodbus5_3.1.10-1ubuntu1_amd64.deb ...
Unpacking libmodbus5:amd64 (3.1.10-1ubuntu1) ...
Selecting previously unselected package zabbix-agent.
Preparing to unpack .../zabbix-agent_1%3a7.0.22-1+ubuntu24.04_amd64.deb ...
Unpacking zabbix-agent (1:7.0.22-1+ubuntu24.04) ...
Setting up libmodbus5:amd64 (3.1.10-1ubuntu1) ...
Setting up zabbix-agent (1:7.0.22-1+ubuntu24.04) ...
Created symlink /etc/systemd/system/multi-user.target.wants/zabbix-agent.service → /usr/lib/systemd/system/zabbix-agent.service.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

```

### **3.1.2 Configuration du fichier zabbix\_agentd.conf**

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

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

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

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

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

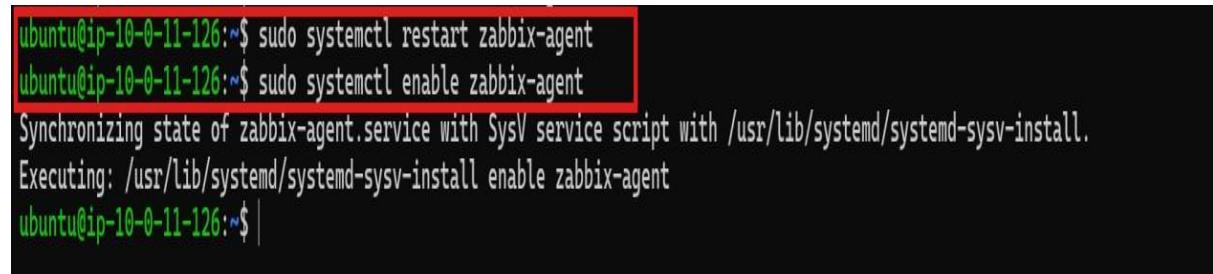
### **3.1.3 Finalisation et persistance :**

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

Bash

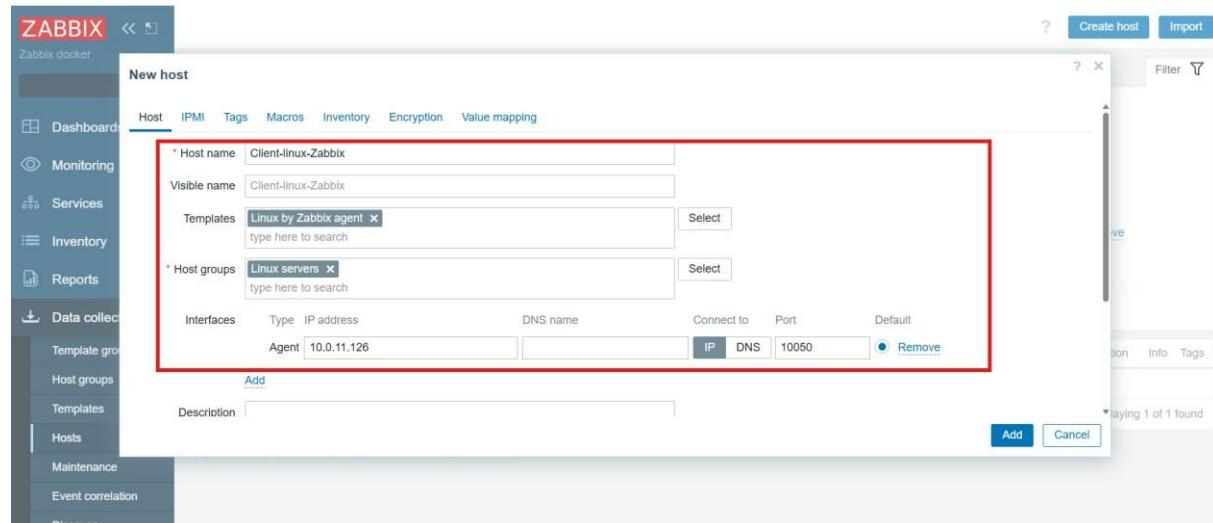
```
sudo systemctl restart zabbix-agent
```

```
sudo systemctl enable zabbix-agent
```



```
ubuntu@ip-10-0-11-126:~$ sudo systemctl restart zabbix-agent
ubuntu@ip-10-0-11-126:~$ sudo systemctl enable zabbix-agent
Synchronizing state of zabbix-agent.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable zabbix-agent
ubuntu@ip-10-0-11-126:~$
```

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



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

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

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

The screenshot shows the Zabbix interface for monitoring hosts. At the top, there are tabs for Items, Triggers, Graphs, Discovery, Web, Interface, Proxy, Templates, Status, Availability, Agent encryption, Info, and Tags. Below the tabs, there are two rows of host statistics: one for 'Items 43' and 'Triggers 15' with IP '10.0.11.126:10050' and another for 'Items 104' and 'Triggers 59' with IP '127.0.0.1:100'. A tooltip for the first host states: 'Received empty response from Zabbix Agent at [10.0.11.126]. Assuming that agent dropped connection because of access permissions.' At the bottom, there are buttons for Disable, Export, Mass update, and Delete.

The screenshot shows the Zabbix 'Hosts' search interface. It includes search filters for Host groups, Status (Any, Enabled, Disabled), Monitored by (Any, Server, Proxy), Name, DNS, IP, Port, and Tags (And/Or, Or). Below the filters is an 'Apply' button. The results table lists a host named 'Client-linux-Zabbix' with the following details: Status: Enabled, Availability: ZBX, Agent enc: None. The 'Status' column for this host is highlighted with a red box.

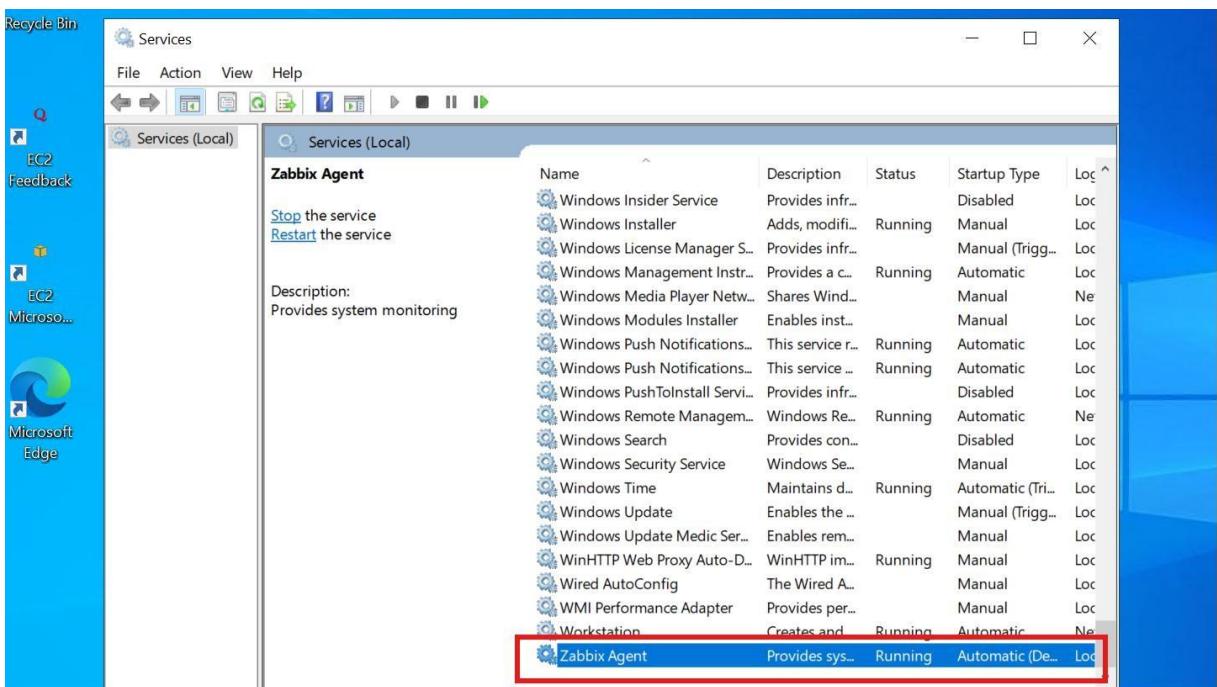
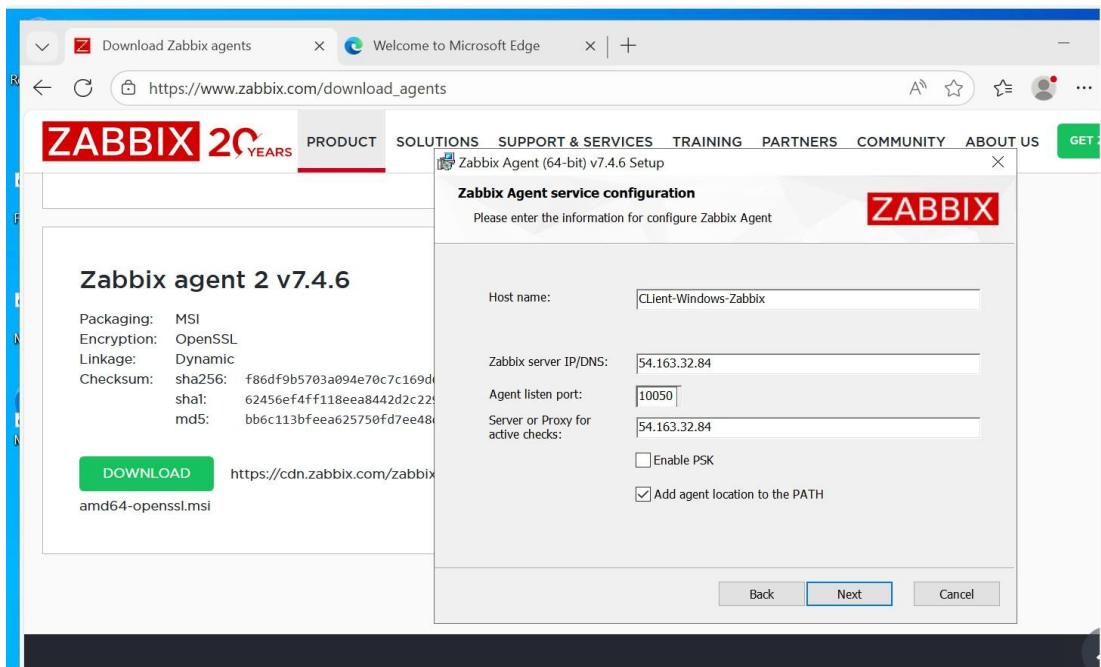
## **3.2 Extension de la supervision :**

### **3.2.1 Déploiement de l'agent MSI :**

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

Configuration appliquée lors de l'installation :

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



### 3.2.2 Configuration du flux réseau et du host :

The screenshot shows the 'New host' configuration dialog in the Zabbix web interface. The 'Host' tab is selected. The host name is set to 'CLIENT-Windows-Zabbix', and the visible name is 'Client-Windows-Zabbix'. The template is 'Windows by Zabbix agent', and the host group is 'Virtual machines'. An interface is defined with IP '10.0.2.40' and port '10050'. The 'Add' button is visible at the bottom.

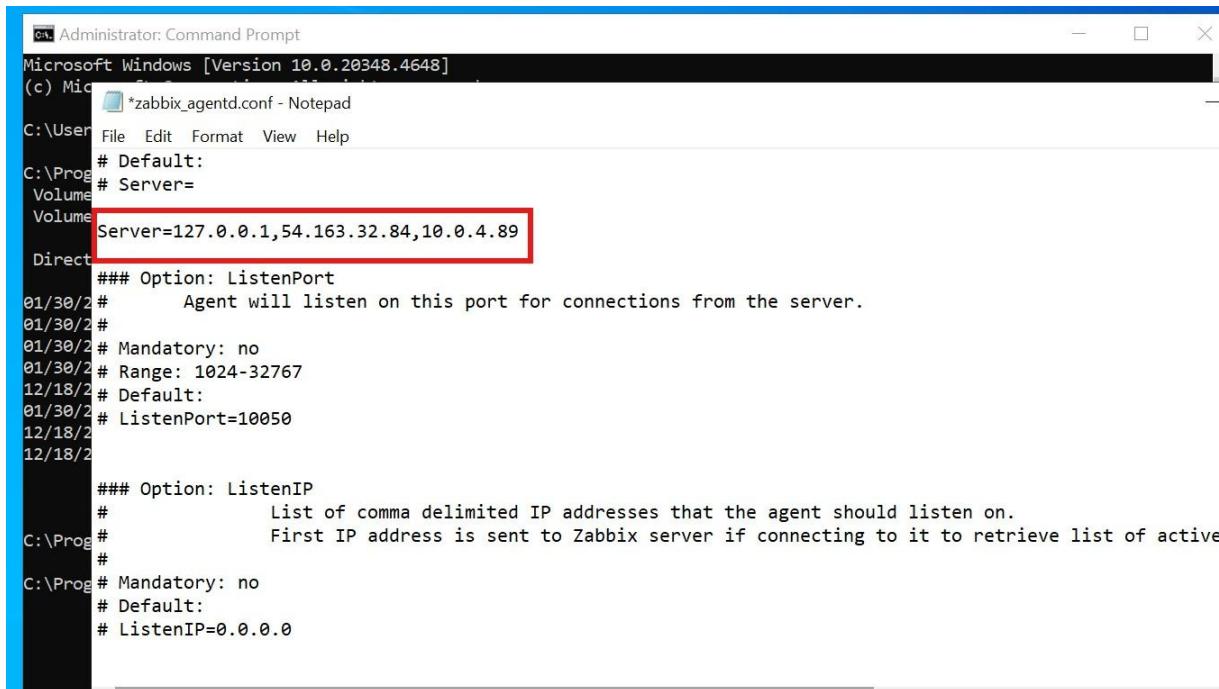
### 3.2.3 Diagnostic : Résolution de l'erreur de connexion :

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

Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent en
Client-linux-Zabbix	Items 75	Triggers 30	Graphs 16	Discovery 3	Web	10.0.11.126:10050		Linux by Zabbix agent	Enabled	ZBX	None
Client-Windows-Zabbix	Items 34	Triggers 13	Graphs 5	Discovery 4	Web	10.0.2.40:10050		Windows by Zabbix agent	Enabled	ZBX	None
Zabbix-Server	Items 104	Triggers 59	Graphs 19	Discovery	Interface	10.0.2.40:10050	Not available	Get value from agent failed: ZBX_TCP_READ() failed: [104] Connection reset by peer			

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

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



```

Administrator: Command Prompt
Microsoft Windows [Version 10.0.20348.4648]
(c) Microsoft Corporation. All rights reserved.

C:\Users\... File Edit Format View Help
*zabbix_agentd.conf - Notepad

# Default:
# Server=
Server=127.0.0.1,54.163.32.84,10.0.4.89

Direct...
    ### Option: ListenPort
01/30/2#             Agent will listen on this port for connections from the server.
01/30/2#
01/30/2# Mandatory: no
01/30/2# Range: 1024-32767
12/18/2# Default:
01/30/2# ListenPort=10050
12/18/2#
12/18/2

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

```

### **3.2.4 Validation du service :**

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

<input type="checkbox"/>	Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent en
<input type="checkbox"/>	Client-linux-Zabbix	Items 75	Triggers 30	Graphs 16	Discovery 3	Web 10.0.11.126:10050			Linux by Zabbix agent	Enabled	ZBX	None
<input type="checkbox"/>	Client-Windows-Zabbix	Items 56	Triggers 22	Graphs 12	Discovery 4	Web 10.0.2.40:10050			Windows by Zabbix agent	Enabled	ZBX	None
<input type="checkbox"/>	Zabbix-Server	Items 104	Triggers 59	Graphs 19	Discovery 5	Web localhost:10050			Linux by Zabbix agent, Zabbix server health	Enabled	ZBX	None

### **3.3 Auto-supervision du serveur Zabbix :**

#### **3.3.1 Identification réseau du conteneur**

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

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

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

```
ubuntu@ip-10-0-4-89:~$ sudo docker inspect -f '{range .NetworkSettings.Networks}{{.IPAddress}}' zabbix-server
172.18.0.3
ubuntu@ip-10-0-4-89:~$
```

#### **3.3.2 État final de la supervision**

Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent e
Client-linux-Zabbix	75	30	16	3	Web	10.0.11.126:10050		Linux by Zabbix agent	Enabled	ZBX	None
Client-Windows-Zabbix	56	22	12	4	Web	10.0.2.40:10050		Windows by Zabbix agent	Enabled	ZBX	None
<b>zabbix-server</b>	136	74	27	5	Web	172.18.0.3:10050		Linux by Zabbix agent, Zabbix server health	Enabled	ZBX	None

## 4 Monitoring et Tableaux de bord :

### 4.1 Centralisation des métriques (Global Infrastructure Monitoring)

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

#### 4.1.1 Graphique de charge CPU

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



#### 4.1.2 Graphique d'utilisation RAM

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

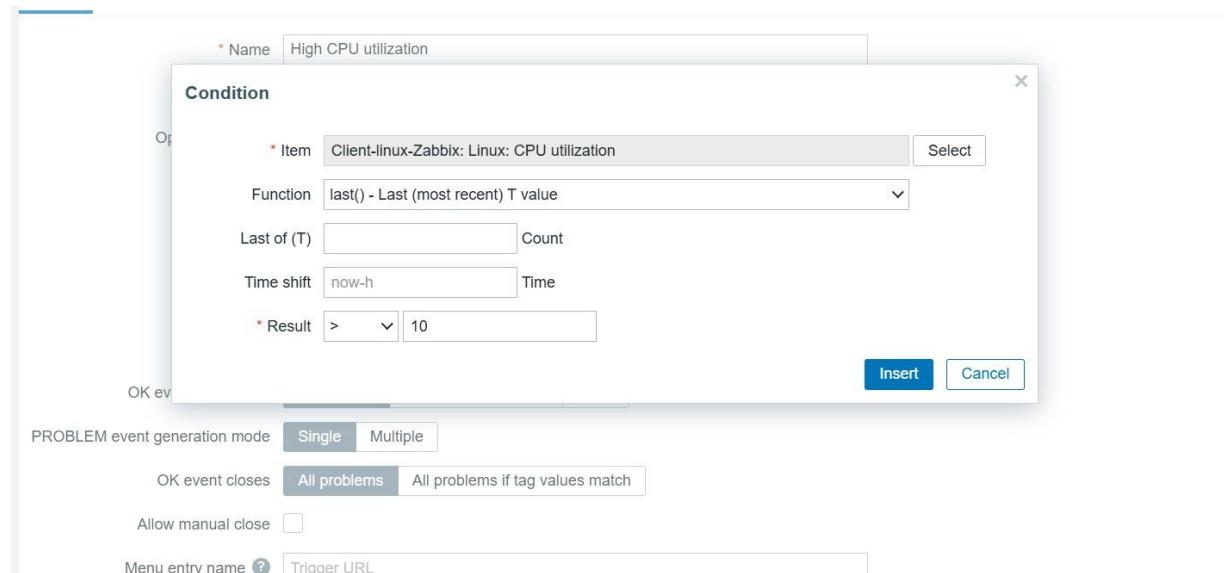


## **4.2 Alertes Proactives et Synthèse Visuelle :**

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

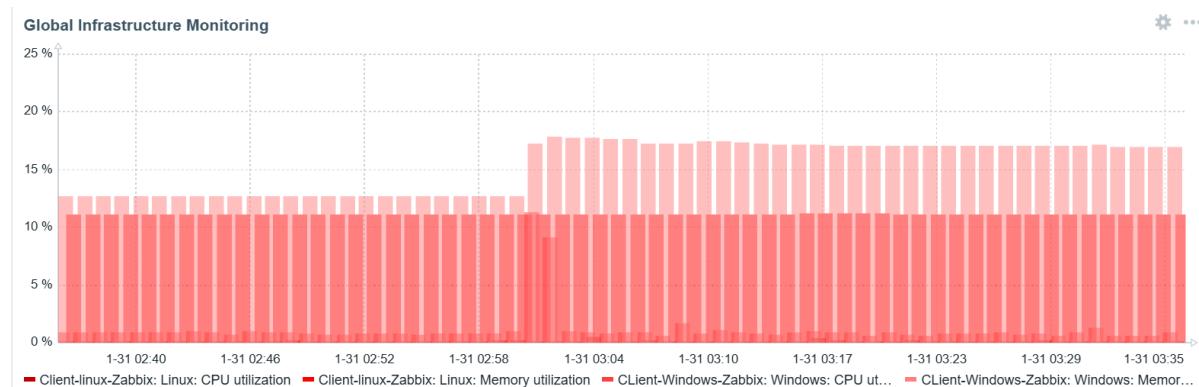
Pour démontrer la réactivité de Zabbix, un déclencheur a été configuré pour l'hôte Linux.

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



### **4.2.2 Synthèse Visuelle du Dashboard Consolidé**

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



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

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

## **Conclusion**

Le projet de déploiement d'une solution de monitoring avec Zabbix a permis de valider la mise en place d'une architecture de supervision fiable et performante, capable de surveiller un parc informatique hybride et distribué au sein du Cloud **Amazon Web Services**.

La réalisation de ce travail pratique a mobilisé un ensemble de compétences techniques en administration système et réseau. Elle a notamment permis de consolider la maîtrise de l'écosystème Cloud à travers la configuration des instances EC2, la gestion des adresses IP publiques et privées ainsi que la sécurisation des flux via les Security Groups AWS. Le projet a également nécessité des compétences en ingénierie Docker, avec le déploiement du serveur Zabbix dans un environnement conteneurisé, impliquant une compréhension approfondie de l'isolation réseau et des mécanismes de communication entre l'hôte et ses conteneurs. Enfin, l'installation et la configuration des agents sur des systèmes d'exploitation différents, notamment Linux Ubuntu et Windows Server, ont permis d'assurer une collecte de données homogène malgré l'hétérogénéité des environnements.

Le succès du projet repose également sur la résolution de plusieurs défis techniques rencontrés lors de la phase d'implémentation. L'instabilité liée à l'adressage IP dynamique des conteneurs Docker (changement d'adresse de 172.18.0.3 à 172.18.0.4) a été corrigée par la mise en place d'une configuration basée sur l'adresse locale 127.0.0.1 ainsi que par l'autorisation de segments réseau spécifiques (172.16.0.0/12), garantissant ainsi la stabilité de l'auto-supervision. Par ailleurs, les erreurs de type « Connection refused » ou « Reset by peer » ont conduit à un travail approfondi sur la configuration des fichiers zabbix\_agentd.conf et sur l'ajustement des pare-feu locaux, notamment Windows Firewall, afin d'assurer une communication sécurisée et fonctionnelle entre le serveur et les agents.

En conclusion, ce projet démontre qu'une supervision efficace constitue un élément fondamental d'une infrastructure Cloud bien administrée. Elle permet de transformer des métriques techniques brutes en un véritable outil d'aide à la décision pour l'administrateur système, contribuant ainsi à la fiabilité, à la performance et à la sécurité globale de l'environnement informatique.