Université Hassan II de Casablanca

Ecole Normale Superieure d'enseignement Technique de Mohammedia

Ingénierie Informatique - Big Data et Cloud Computing

Rapport de Projet de Fin de Module

# Déploiement d'une architecture de supervision Cloud distribuée avec Zabbix sur AWS


**Réalisé par :**
Ameziane Mohamed (II-BDCC)

**Sous l'encadrement de :**
Pr. Azeddine KHIAT

Année Universitaire 2024 – 2025

# Table des matières

# 1 Introduction

Ce projet vise à déployer une solution de supervision robuste et distribuée dans un environnement Cloud. L'objectif est de monitorer en temps réel la disponibilité et les performances d'un parc de serveurs hétérogène (Linux et Windows) hébergé sur Amazon Web Services (AWS).

Pour répondre à ce besoin, nous avons choisi **Zabbix 6.0 LTS**, une solution open-source leader sur le marché, couplée à **Docker** pour faciliter le déploiement et la maintenance du serveur de supervision. L'infrastructure repose sur des instances EC2 interconnectées au sein d'un VPC sécurisé.

# 2 Architecture Réseau

La première étape de notre projet a consisté à construire le socle réseau (Network Layer) sur AWS. Nous avons mis en place un **VPC (Virtual Private Cloud)** personnalisé pour isoler nos ressources et maîtriser l'adressage IP.

L'architecture réseau a été configurée étape par étape via la console AWS VPC.

## 2.1 Création du VPC

Nous avons défini un espace d'adressage privé pour notre projet.

1. Dans la console AWS, service **VPC** > **Your VPCs** > **Create VPC**.
2. **Name tag :** `Projet-Zabbix-VPC`.
3. **IPv4 CIDR block :** `10.0.0.0/16` (Ce qui nous offre 65 536 adresses IP privées potentielles).
4. **Tenancy :** Default.

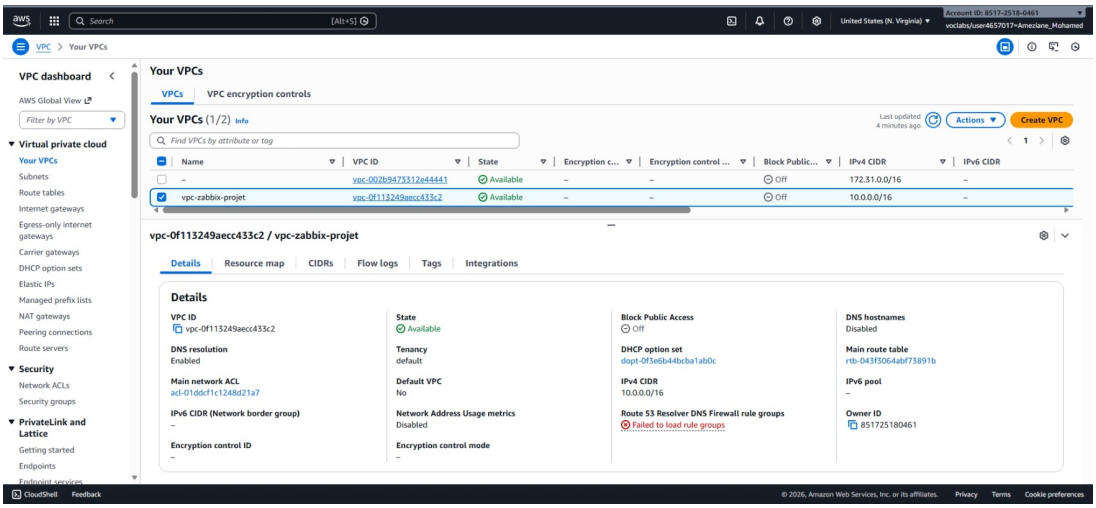


FIGURE 1 – Création du VPC

## 2.2 Configuration de l'accès Internet (Internet Gateway)

Pour que notre serveur Zabbix puisse télécharger les paquets Docker et que nous puissions y accéder à distance, une connexion internet est indispensable.

1. Menu **Internet Gateways** > **Create internet gateway**.
2. **Name :** `Zabbix-IGW`.
3. **Action :** Une fois créée, nous avons cliqué sur **Attach to VPC** et sélectionné notre `Projet-Zabbix-VPC`.

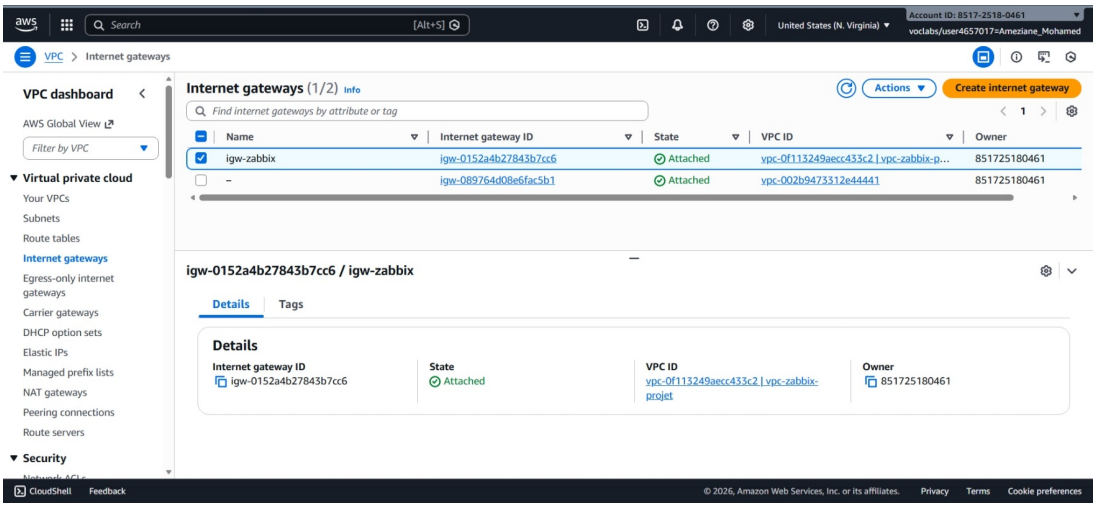


FIGURE 2 – Configuration de Gateway

## 2.3 Création du Sous-réseau Public (Subnet)

Nous avons segmenté le VPC pour placer nos instances.

1. Menu **Subnets** > **Create subnet**.
2. **VPC ID :** Sélection de `Projet-Zabbix-VPC`.
3. **Subnet name :** `Public-Subnet-1`.
4. **Availability Zone :** `us-east-1a` (pour garantir la haute disponibilité dans une zone précise).
5. **IPv4 CIDR block :** `10.0.1.0/24` (256 adresses IP).
6. **Configuration IP :** Nous avons activé l'option *"Enable auto-assign public IPv4 address"* pour que chaque instance reçoive automatiquement une IP publique.

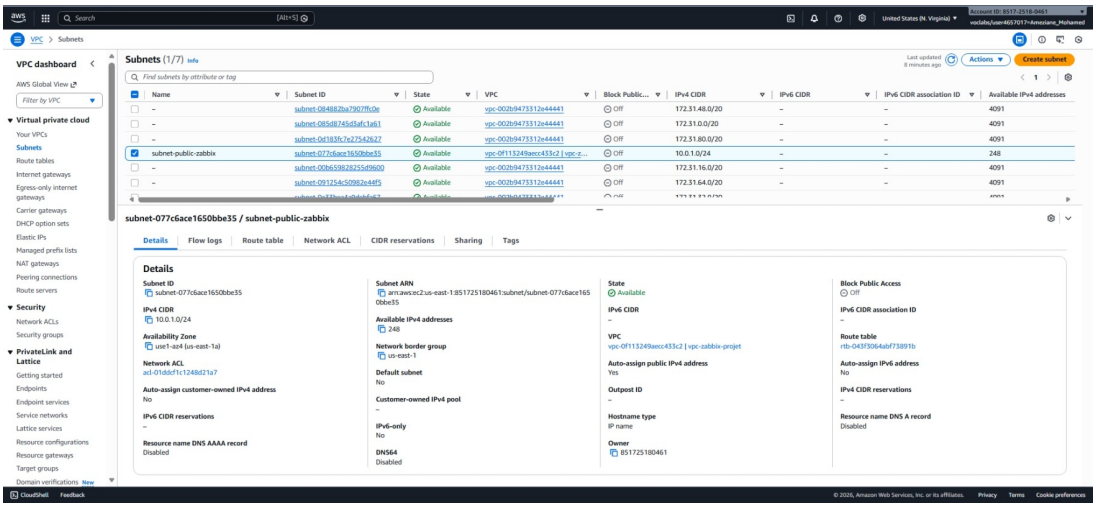


FIGURE 3 – Création de sous-réseau Public

## 2.4 Table de Routage (Routing Table)

C'est l'étape qui rend le sous-réseau véritablement "public".

1. Menu **Route Tables**.
2. Sélection de la table de routage associée à notre VPC.
3. Onglet **Routes** > **Edit routes** > **Add route**.
4. **Destination :** `0.0.0.0/0` (Tout le trafic internet).
5. **Target :** Sélection de l'Internet Gateway créé (`Zabbix-IGW`).
6. **Association :** Dans l'onglet *Subnet associations*, nous avons lié cette table à notre `Public-Subnet-1`.

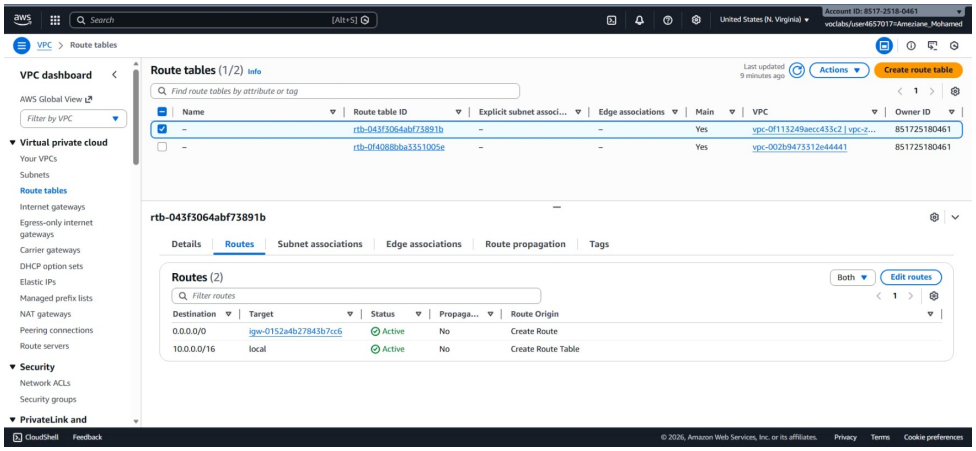


FIGURE 4 – Table de Routage

## 2.5 Sécurité et Filtrage (Security Groups)

Enfin, nous avons créé un pare-feu virtuel (**Security Group**) appliqué à nos instances pour filtrer les connexions entrantes.

**Configuration via le menu "EC2 > Security Groups" :** Nous avons créé le groupe `Zabbix-SG` avec les règles entrantes (Inbound Rules) suivantes :

1. **Administration (SSH/RDP) :** Ports 22 et 3389 ouvert sur `0.0.0.0/0`.
2. **Interface Web :** Port 80 ouvert sur `0.0.0.0/0` pour l'accès HTTP.
3. **Communication Zabbix (Critique) :**
   — **Port 10050 (TCP) :** Autorisé pour permettre au serveur de contacter les agents. Sans cette règle explicite, nous avions rencontré une erreur "Connection Refused".
   — **Port 10051 (TCP) :** Autorisé pour les remontées d'alertes actives.

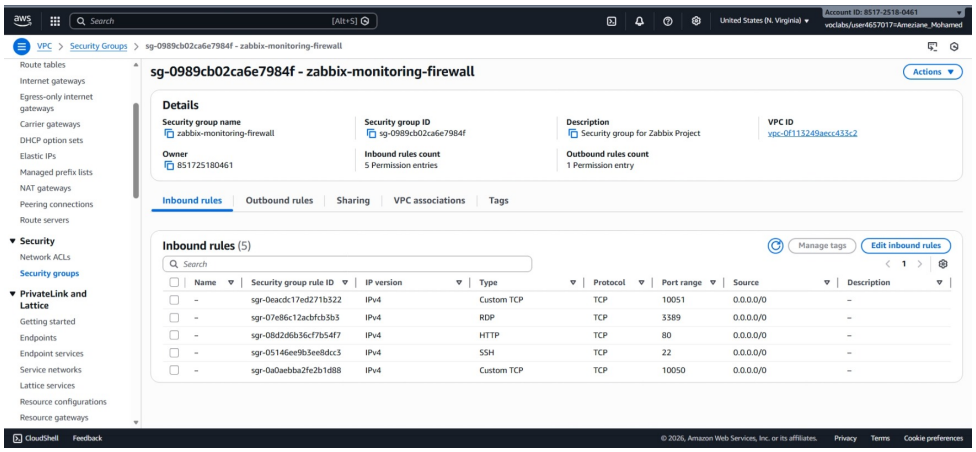


FIGURE 5 – Configuration des règles de sécurité (Inbound Rules)

# 3 Architecture des Instances EC2

Une fois le réseau configuré, nous avons procédé au déploiement des serveurs virtuels. Cette étape a nécessité la création préalable d'une clé de sécurité, puis le lancement successif de trois instances via la console EC2.

## 3.1 Création de la Paire de Clés (Key Pair)

Pour sécuriser l'accès SSH (Linux) et le décryptage du mot de passe Administrateur (Windows), une paire de clés est requise.

1. Dans le menu de gauche de la console EC2, sous *Network & Security*, nous avons cliqué sur **Key Pairs**.
2. Clic sur le bouton orange **Create key pair**.
3. **Name :** `kp-projet-zabbix`.
4. **Key pair type :** RSA.
5. **Private key file format :** `.pem` (Format standard pour OpenSSH).
6. Après validation, le fichier `kp-projet-zabbix.pem` a été téléchargé automatiquement. Ce fichier est critique et a été stocké précieusement sur notre machine locale.

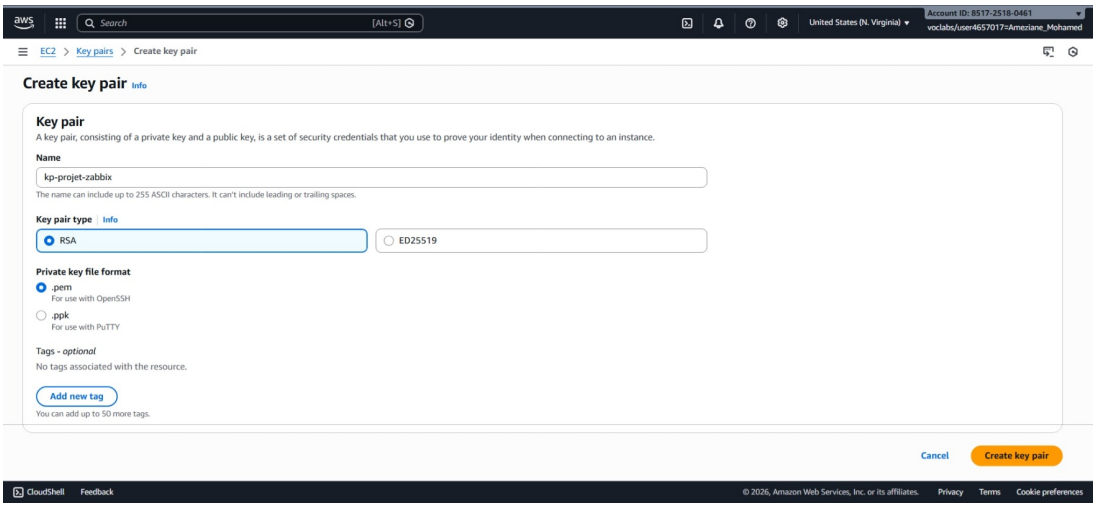


FIGURE 6 – Création de la Paire de Clés

## 3.2 Procédure de Lancement : Serveur Zabbix

Le serveur central nécessite des ressources suffisantes pour exécuter Docker et la base de données.

1. Depuis le tableau de bord EC2, nous avons cliqué sur **Launch Instances**.
2. **Name and tags :** `Zabbix-Server`.
3. **Application and OS Images (AMI) :** Sélection de **Ubuntu Server 24.04 LTS** (Free tier eligible).
4. **Instance type :** Nous avons sélectionné **t3.large** (2 vCPU, 8 Gio Mémoire).
5. **Key pair :** Sélection de `kp-projet-zabbix`.
6. **Network settings :**
   — **VPC :** `Projet-Zabbix-VPC` (créé précédemment).
   — **Subnet :** `Public-Subnet-1`.
   — **Auto-assign Public IP :** Enable (Pour l'accès SSH et Web).
   — **Security Group :** Sélection de `Zabbix-SG` (créé précédemment).
7. **Storage :** 20 Gio gp3 (Pour stocker les logs et la base de données Docker).

## 3.3 Procédure de Lancement : Clients de Supervision

Nous avons répété l'opération pour les deux machines cibles, avec des configurations adaptées à leur OS.

**A. Client Linux (Ubuntu)**
— **Name :** `Ameziane-Client-Linux`.
— **AMI :** Ubuntu Server 24.04 LTS.
— **Instance type : t3.medium** (2 vCPU, 4 Gio RAM).
**B. Client Windows Server**
— **Name :** `Ameziane-Client-Windows`.
— **AMI :** Microsoft Windows Server 2022 Base.
— **Instance type : t3.large** (2 vCPU, 8 Gio RAM).

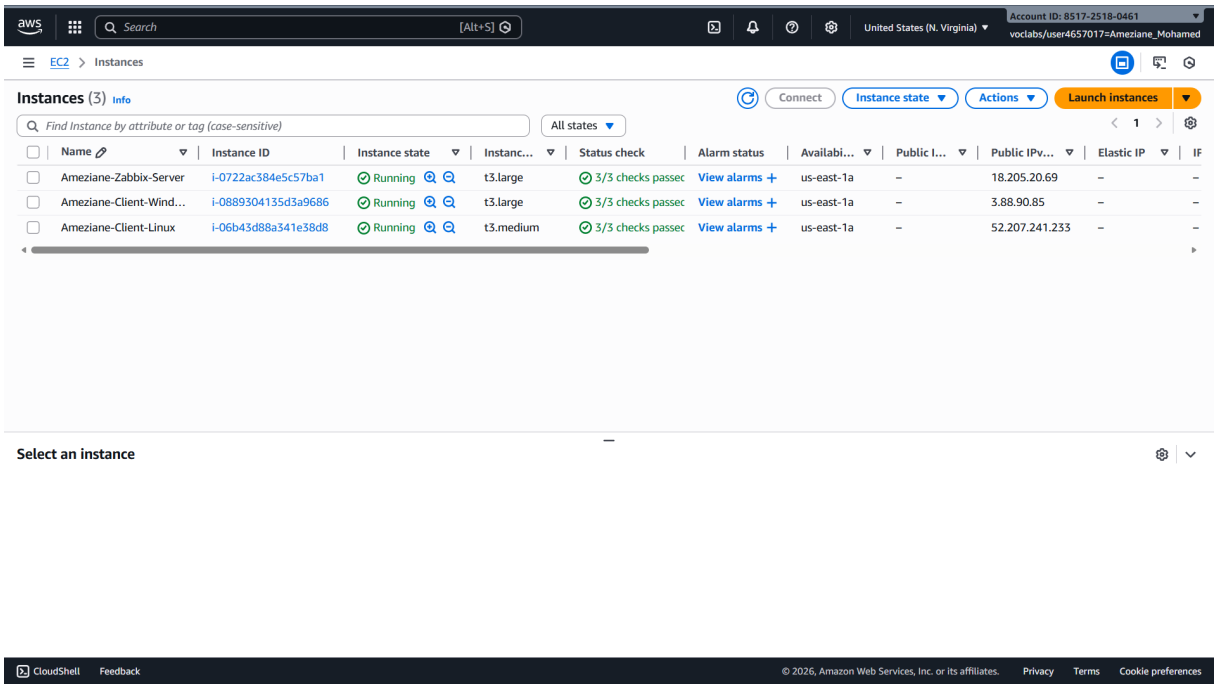


FIGURE 7 – Vue d'ensemble des 3 instances EC2 en cours d'exécution (Status check : 2/2)

# 4 Déploiement du Serveur Zabbix

Le serveur de supervision a été déployé en utilisant la conteneurisation pour garantir l'isolation des services.

## 4.1 Connexion et Préparation

L'accès au serveur nécessite une clé privée sécurisée.

1. Sécurisation de la clé privée :

```
$ chmod 400 kp-projet-zabbix.pem
```

2. Connexion SSH :

```
$ ssh -i "kp-projet-zabbix.pem" ubuntu@18.205.20.69
```

## 4.2 Installation de Docker et Correction de Version

Lors de l'installation par défaut via `apt`, nous avons rencontré une erreur `KeyError: ContainerConfig` due à une version obsolète de `docker-compose` (v1.29) sur les dépôts Ubuntu.

**Procédure de correction appliquée :**

1. Désinstallation de la version obsolète :

```
$ sudo apt remove docker-compose -y
```

2. Installation manuelle du binaire officiel **Docker Compose v2** depuis GitHub :

```
$ sudo curl -L "https://github.com/docker/compose/releases/
   download/v2.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /
   usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

3. Vérification de la version :

```
$ docker-compose --version
```

## 4.3 Déploiement des Services

Nous avons créé le fichier d'orchestration `docker-compose.yml` à la racine du dossier utilisateur.

1. **Contenu du fichier de configuration :**

```
version: '3.5'
services:
  zabbix-db:
    image: mariadb:10.6
    container_name: zabbix-db
    restart: always
    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-6.0-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-6.0-latest
    container_name: zabbix-web
    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
```

Listing 1 – docker-compose.yml

2. **Lancement de conteneur :**

```
$ docker compose up -d
```

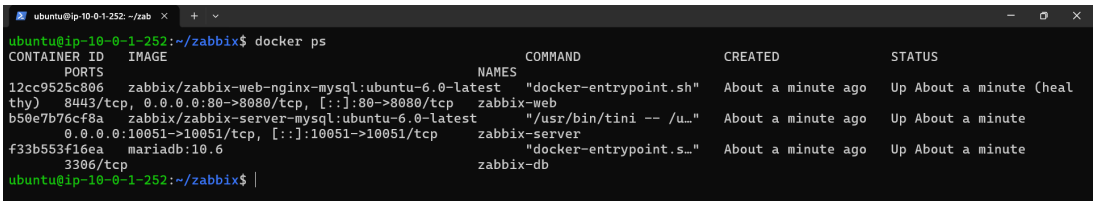


FIGURE 8 – Vérification des conteneurs actifs

Une fois les conteneurs stables, l'interface Web est devenue accessible sur l'IP Publique.

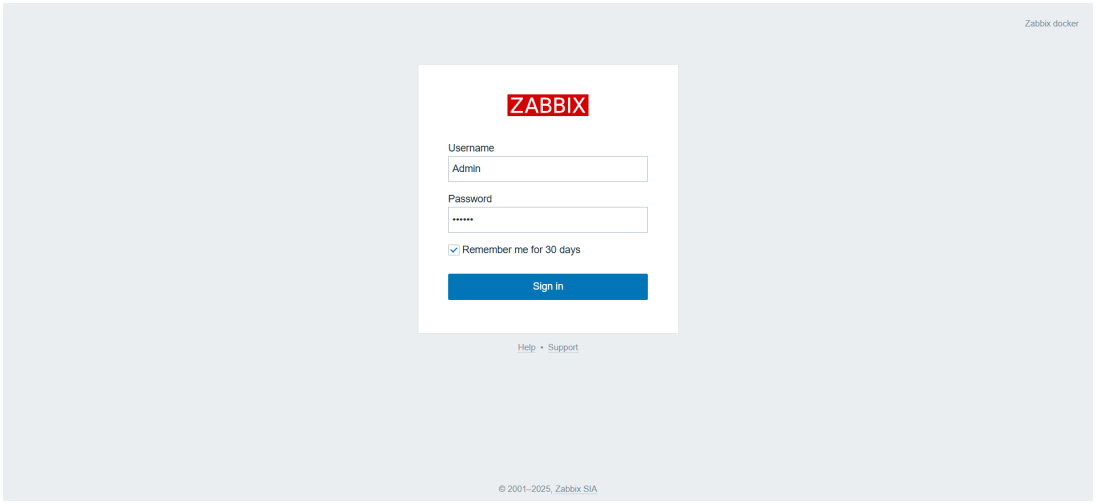


FIGURE 9 – Interface de Connexion Zabbix

En se connectant via Utilisateur : Admin et Mot de pass : zappix

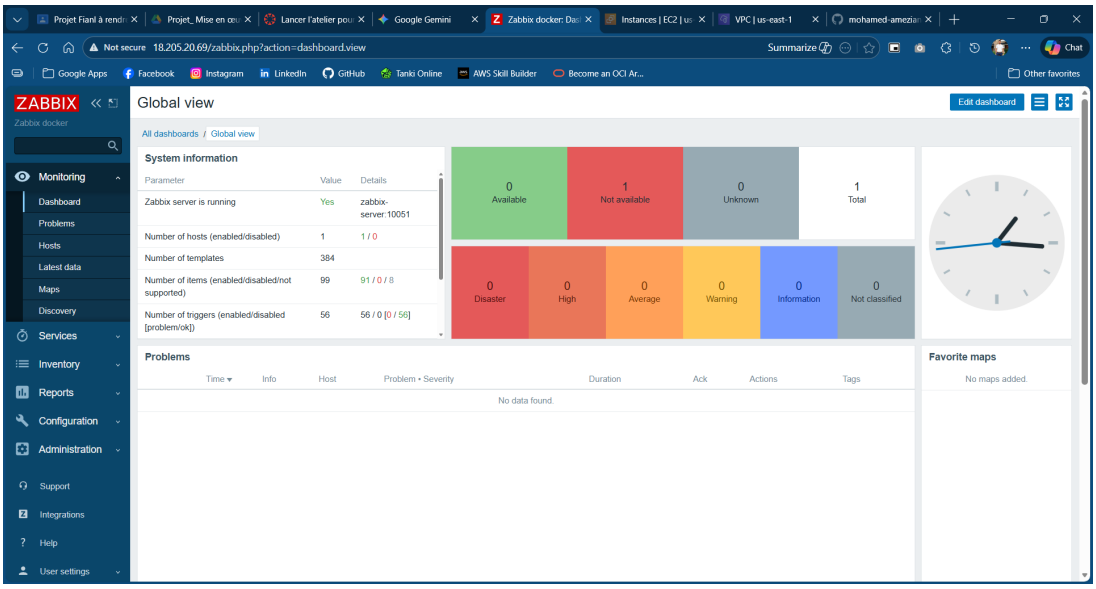


FIGURE 10 – Interface de Connexion Zabbix

# 5 Configuration des Clients (Agents)

## 5.1 Client Linux (Ubuntu)

Sur la machine Ubuntu 24.04, le paquet `zabbix-agent` n'était pas disponible dans les dépôts par défaut.

1. **Ajout du dépôt officiel Zabbix :**

```
$ wget https://repo.zabbix.com/zabbix/6.0/ubuntu/pool/main/z/
   zabbix-release/zabbix-release_6.0-6+ubuntu24.04_all.deb
$ sudo dpkg -i zabbix-release_6.0-6+ubuntu24.04_all.deb
$ sudo apt update
```

2. **Installation et Configuration :**

```
$ sudo apt install zabbix-agent -y
$ sudo nano /etc/zabbix/zabbix_agentd.conf
```

Modification des paramètres pour pointer vers l'IP Privée du Serveur (`10.0.1.252`) :

```
# /etc/zabbix/zabbix_agentd.conf
Server=10.0.1.252
ServerActive=10.0.1.252
Hostname=Ameziane-Client-Linux
```

3. **Redémarrage :**

```
$ sudo systemctl restart zabbix-agent
$ sudo systemctl enable zabbix-agent
```

## 5.2 Client Windows

Pour Windows, l'installation est graphique.

1. **Récupération du mot de passe :** Dans la console AWS > Connect > RDP Client > "Get Password" en utilisant le fichier `.pem`.
2. **Connexion RDP :** Utilisation de l'outil "Bureau à distance".
3. **Installation :** Téléchargement de l'agent MSI Zabbix 6.0 LTS.
4. **Configuration :** Lors de l'installation, nous avons renseigné :
   — Zabbix Server IP : `10.0.1.252`
   — Agent Listen Port : `10050`
   — Hostname : `Ameziane-Client-Windows`
5. **Pare-feu :** Création d'une règle entrante sur le port 10050 dans le Pare-feu Windows Defender.

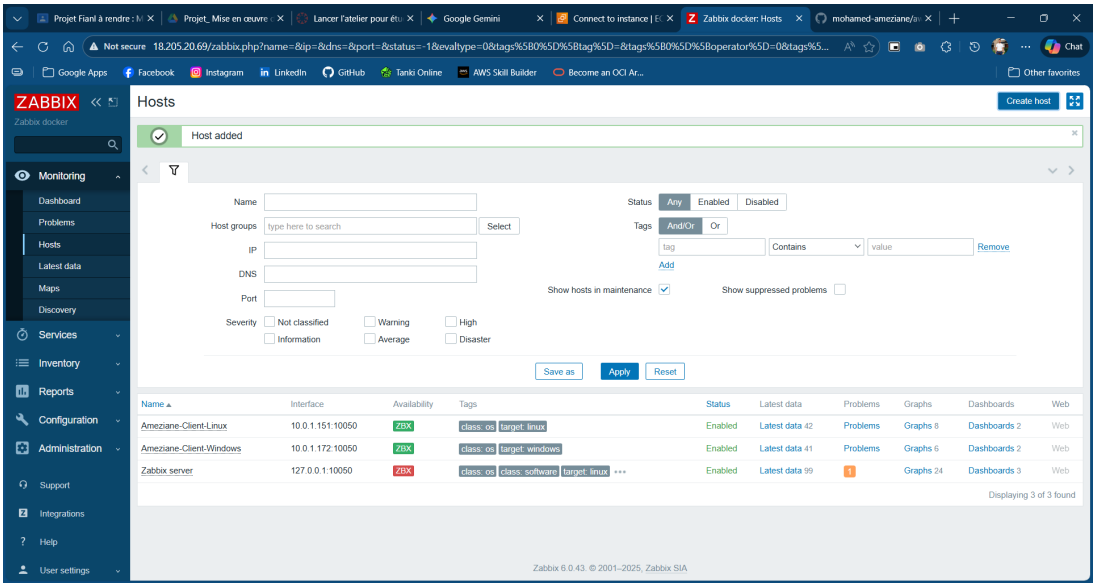


FIGURE 11 – Configuration de l'agent Linux via Nano

# 6 Monitoring et Analyse des Métriques

Une fois l'infrastructure en place, la dernière étape a consisté à relier nos instances EC2 à l'interface de supervision pour visualiser les données collectées.

## 6.1 Visualisation des Graphiques

Grâce aux templates appliqués, Zabbix génère automatiquement des graphiques historiques sans configuration supplémentaire. Nous n'avons pas eu besoin de créer de tableau de bord complexe pour accéder aux données essentielles.

Pour visualiser l'état de santé des machines :

1. Nous sommes allés dans le menu **Monitoring > Hosts**.
2. Nous avons cliqué sur le lien **Graphs** correspondant à la machine `Ameziane-Client-Linux`.

Les graphiques ci-dessous illustrent la charge processeur (CPU Load) et la consommation de memoire (RAM) collectée en temps réel. Ils prouvent que la chaîne de supervision est fonctionnelle de bout en bout.


FIGURE 12 – Visualisation de la charge CPU du client Linux


FIGURE 13 – Visualisation de la charge RAM du client Linux

# 7 Conclusion

Ce projet a permis de valider des compétences clés en Cloud Computing et Administration Système. Nous avons réussi à déployer une architecture complète malgré plusieurs défis techniques :

- La gestion fine des **Security Groups AWS** pour permettre la communication inter-instances.
- La résolution de conflits de versions logicielles (**Docker Compose v1 vs v2**) sur Ubuntu 24.04.
- Le débogage de conteneurs Docker (Migration **MySQL vers MariaDB** et nettoyage de volumes) pour assurer la stabilité du backend.

L'infrastructure Zabbix est désormais opérationnelle et permet une surveillance proactive de l'ensemble du parc.