Université Mundiapolis

Année universitaire : 2025-2026

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)

Étudiant : Aymane El yamani
Encadrant : Prof. Azeddine KHIAT
Filière : 2ANCI – Génie Informatique

Table des matières

Chapitre 1

Introduction

Ce projet porte sur la mise en place d'une infrastructure de supervision centralisée dans le cloud (AWS), afin de superviser un parc hybride composé de machines Linux et Windows. La solution retenue s'appuie sur **Zabbix** déployé en **conteneurs Docker** sur une instance EC2, et sur des **agents Zabbix** installés sur les machines clientes.

1.1 Contexte et outils

Les outils et services mobilisés sont :
— **AWS (EC2, VPC)** pour l'hébergement et le réseau.
— **Docker / Docker Compose** pour déployer Zabbix de manière reproductible.
— **Zabbix** pour la collecte, l'agrégation et la visualisation des métriques.

Chapitre 2

Objectifs

L'objectif principal est de déployer une infrastructure de monitoring sur AWS avec Zabbix (Docker), permettant de superviser un parc hybride (Linux & Windows).

2.1 Objectifs opérationnels

— Concevoir un réseau (VPC, sous-réseau) adapté à l'accès et à la supervision.
— Mettre en place des règles de sécurité (Security Groups) cohérentes avec les flux Zabbix.
— Déployer un serveur Zabbix conteneurisé (serveur, interface web, base de données).
— Installer et configurer les agents Zabbix sur Linux et Windows.
— Valider la supervision : hôtes visibles, statut "ZBX" au vert, données en temps réel, graphiques.

Chapitre 3

Architecture de la solution

3.1 Architecture réseau (VPC et sous-réseaux)

Afin de simplifier l'accès dans le cadre du Lab, l'architecture repose sur un VPC avec un sous-réseau public.

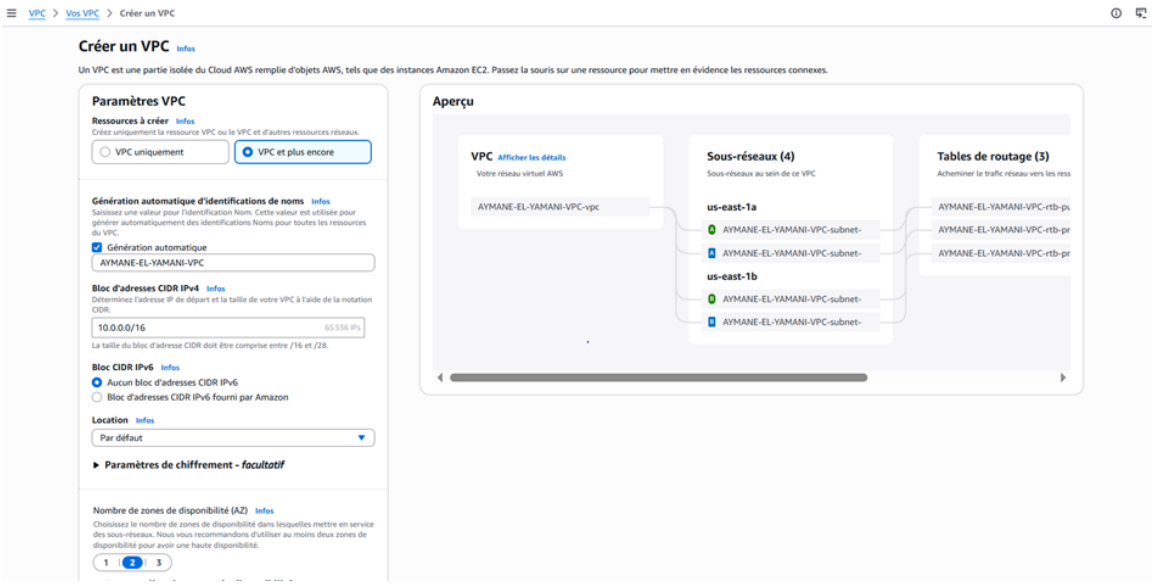

FIGURE 3.1 – Création du VPC

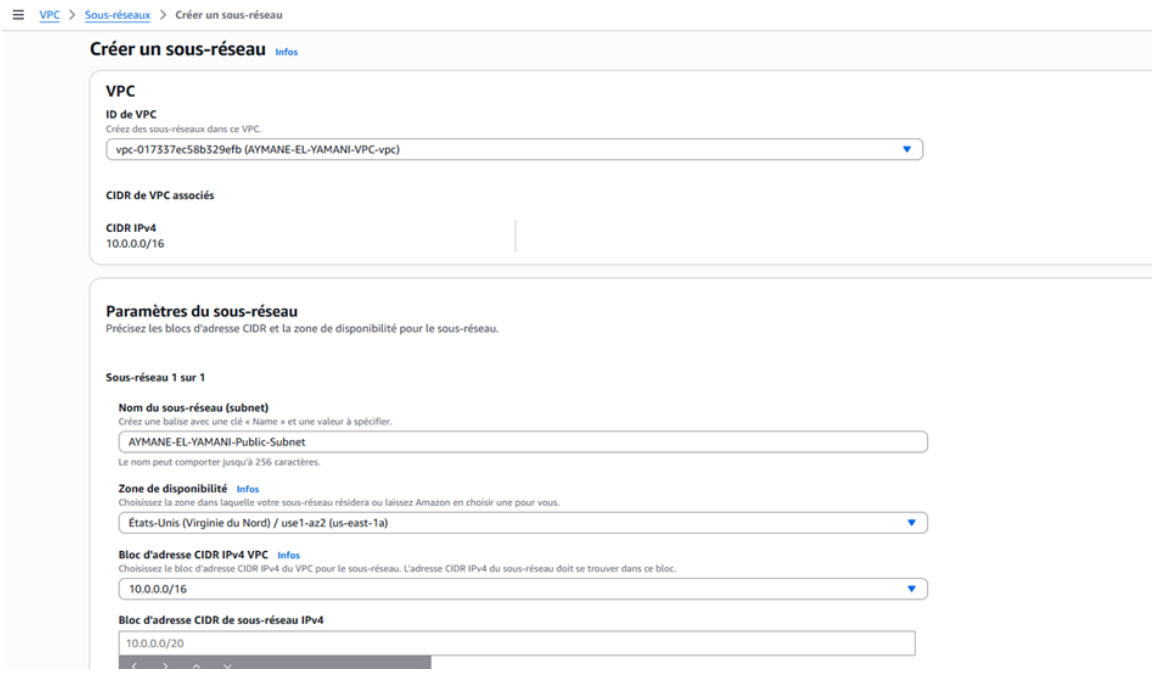

FIGURE 3.2 – Création du sous-réseau

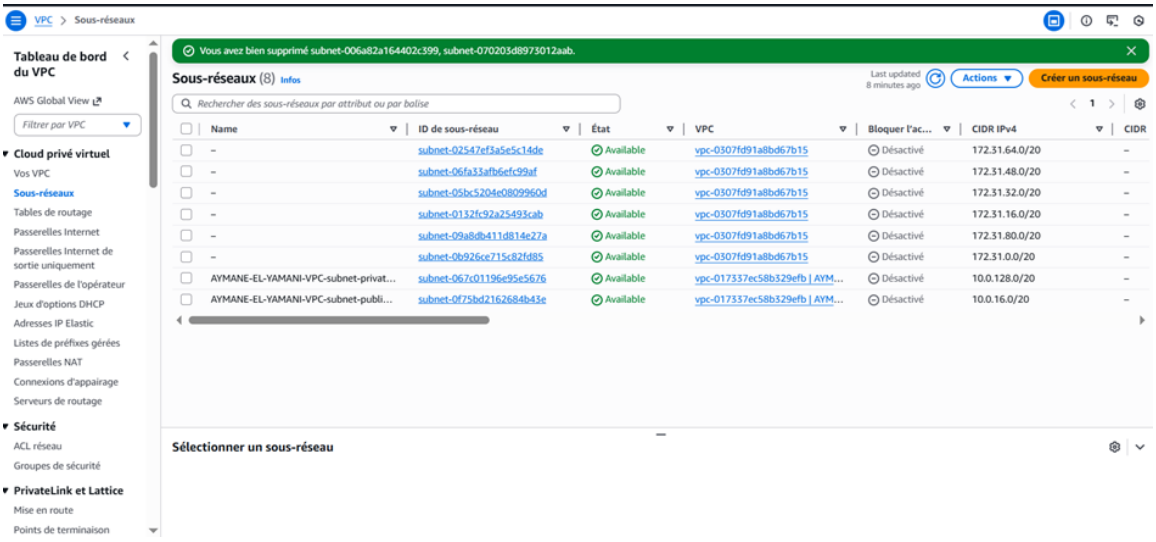

FIGURE 3.3 – Organisation des sous-réseaux (public/privé)

3.2 Sécurité (Security Groups)

Les groupes de sécurité autorisent les ports nécessaires :
— 80/443 (interface Web Zabbix),
— 10050/10051 (agent/serveur Zabbix),
— 22 (SSH) et 3389 (RDP) pour l'administration.

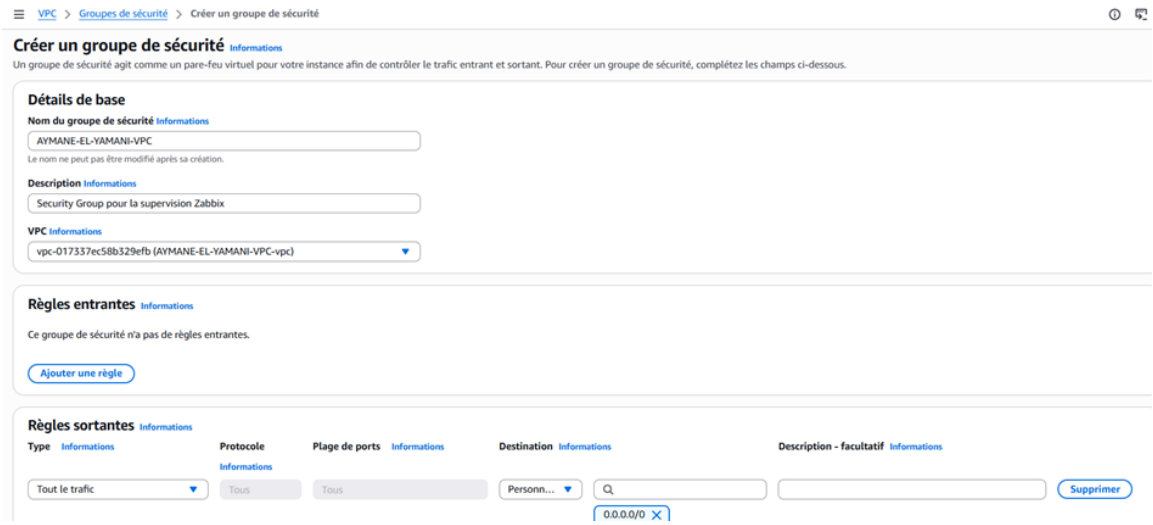

FIGURE 3.4 – Création d'un groupe de sécurité

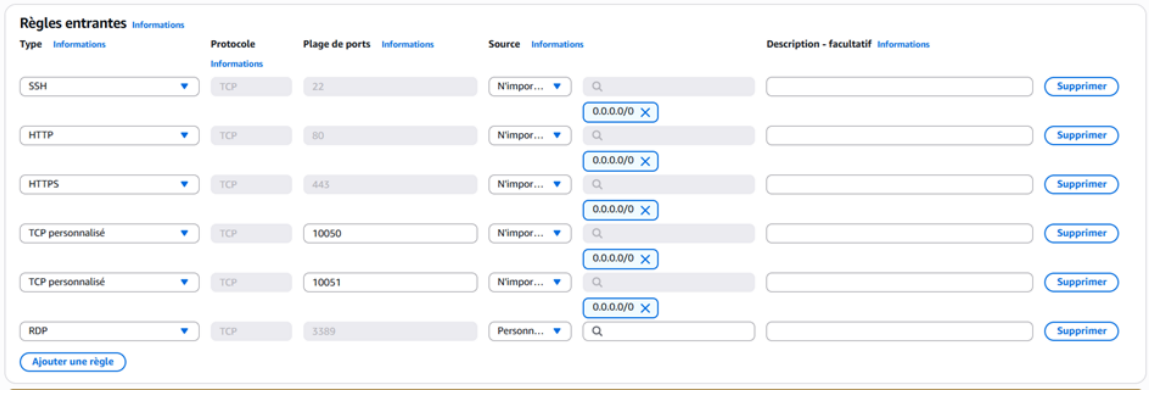

FIGURE 3.5 – Règles entrantes (ports requis)

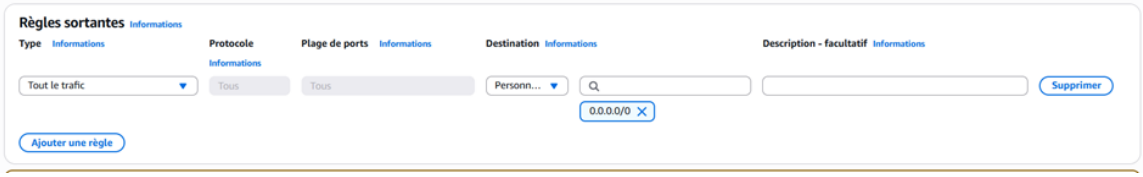

FIGURE 3.6 – Règles sortantes

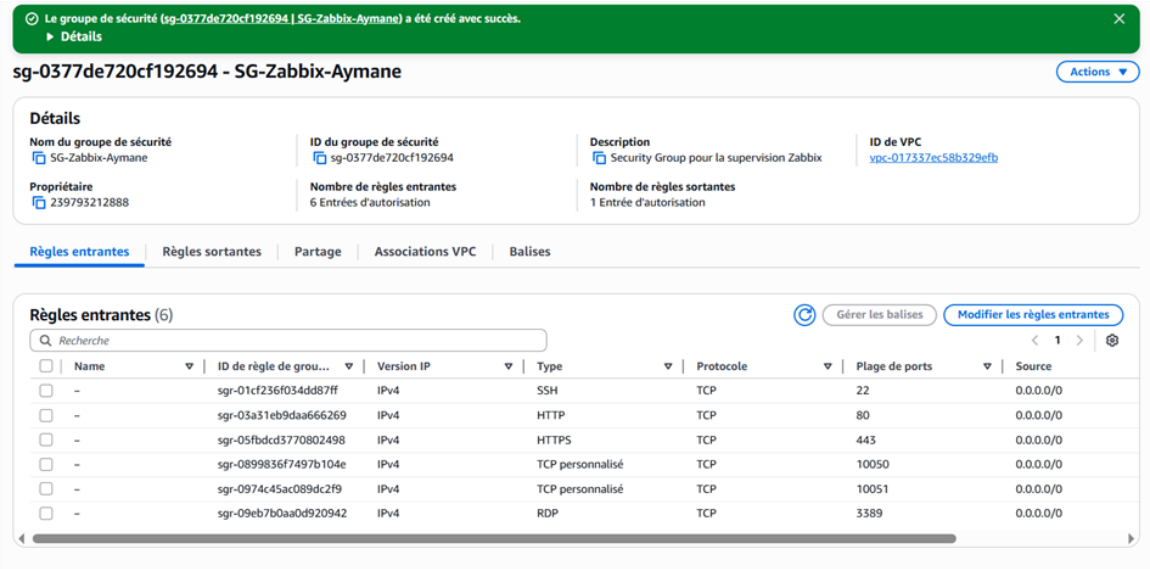

FIGURE 3.7 – Groupe de sécurité créé avec succès

3.3 Instances EC2 (serveur et clients)

L'architecture retient trois instances : un serveur Zabbix, un client Linux et un client Windows. Les tailles d'instances sont choisies pour rester compatibles avec les limitations du Lab.

Rôle	Système	Type (exemple)
Serveur Zabbix	Ubuntu	t3.large
Client Linux	Ubuntu	t3.medium
Client Windows	Windows Server	t3.large

TABLE 3.1 – Instances EC2 utilisées

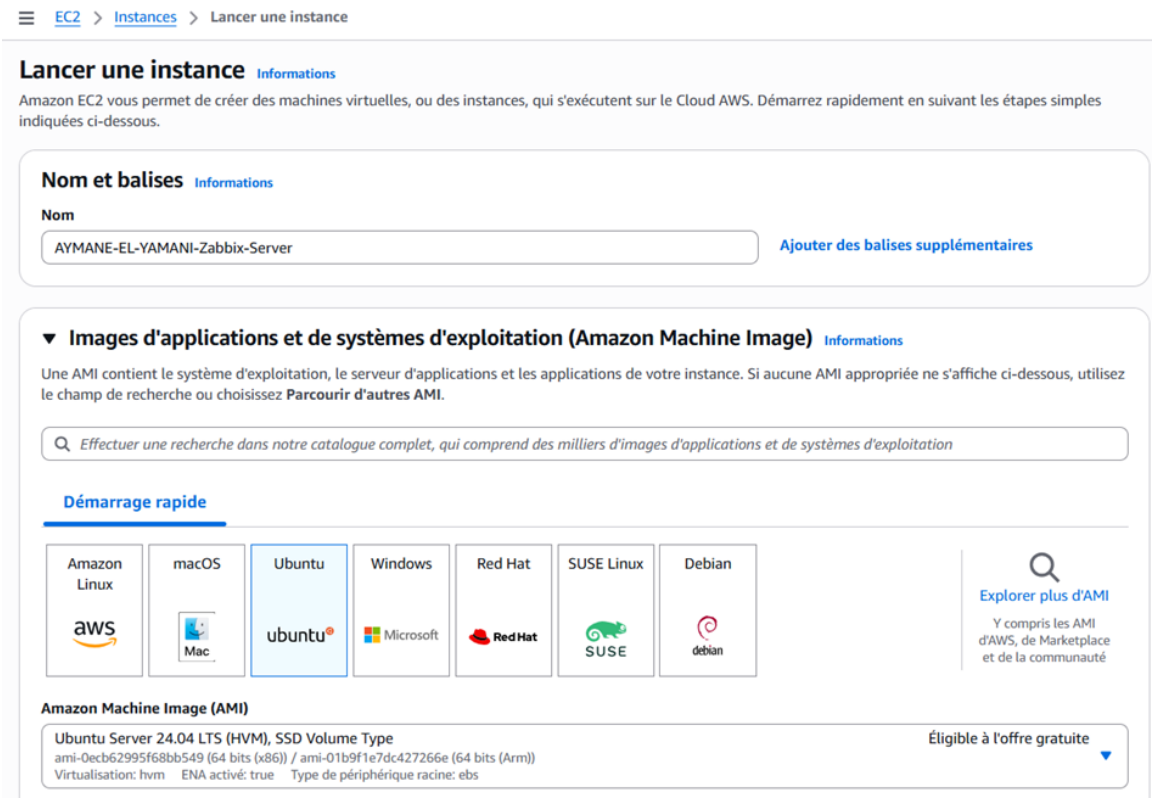

FIGURE 3.8 – Lancement de l'instance EC2 (serveur Zabbix)

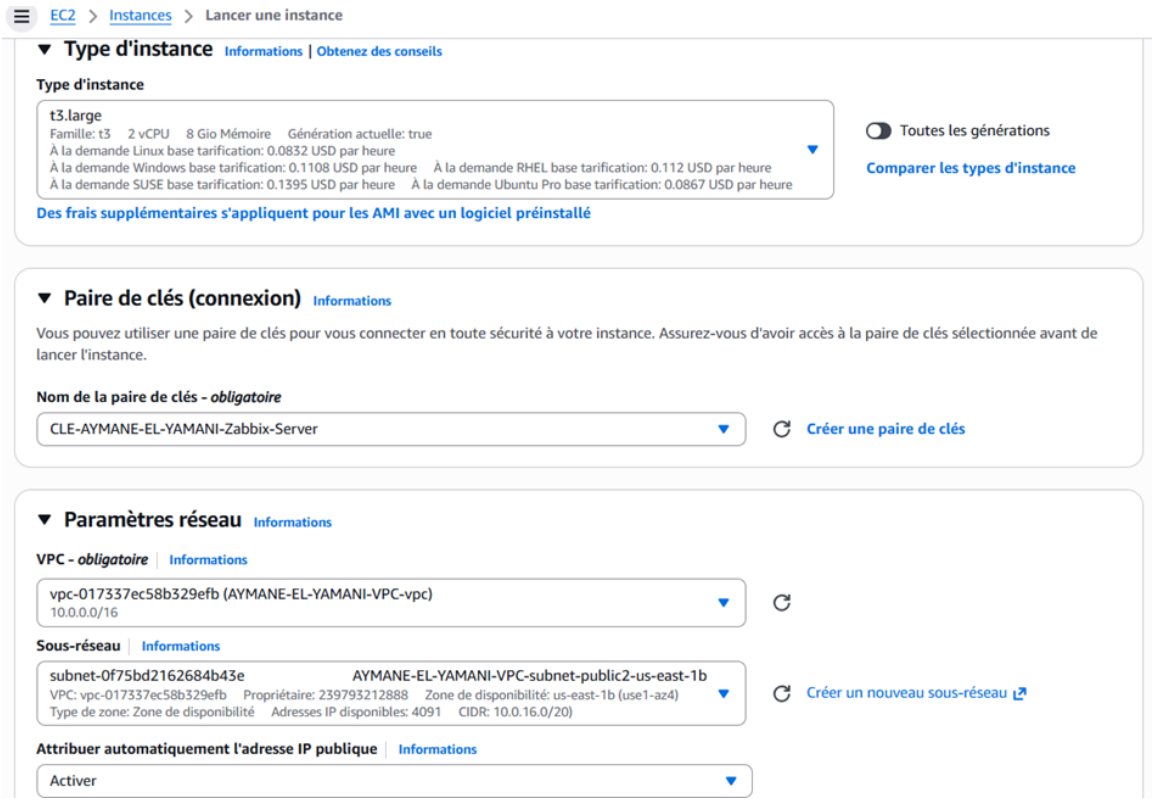

FIGURE 3.9 – Création de la clé de connexion (Key Pair)

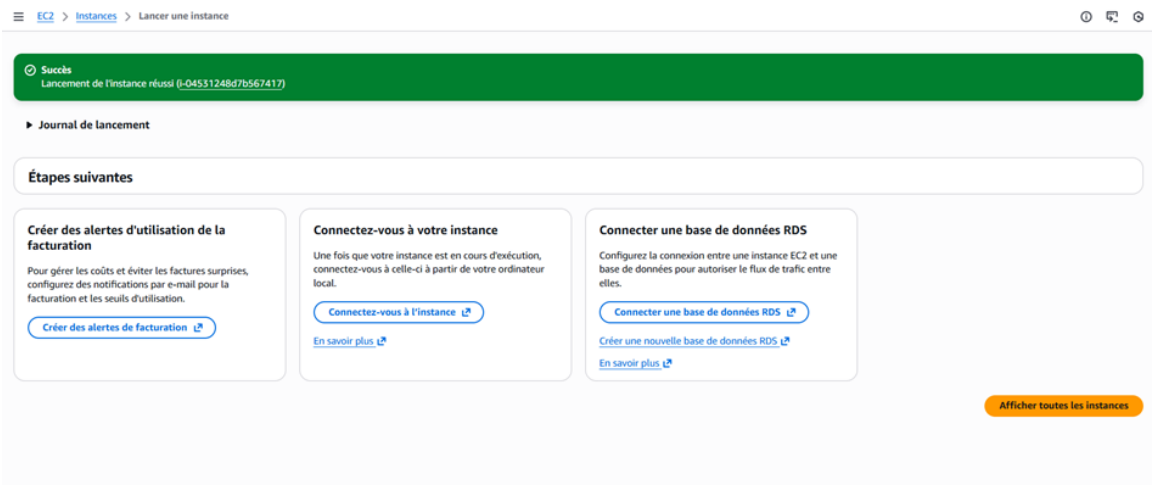

FIGURE 3.10 – Instance créée avec succès

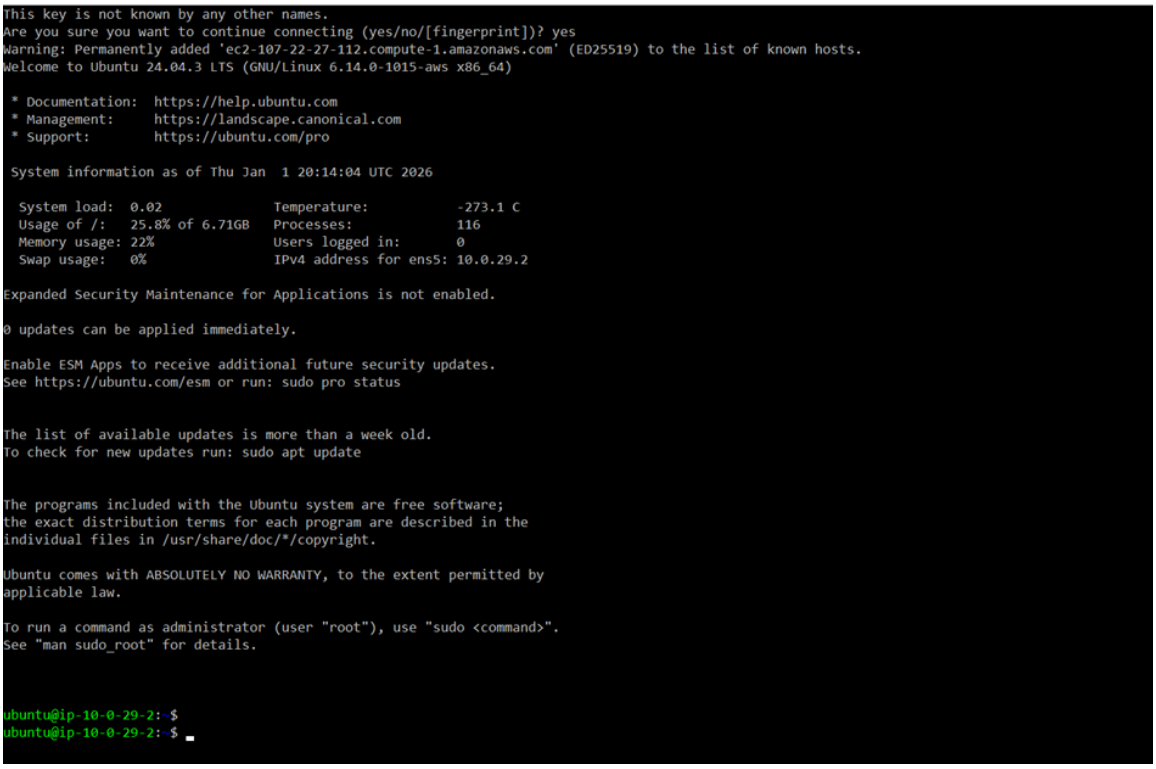

FIGURE 3.11 – Connexion SSH réussie au serveur

Chapitre 4

Étapes d'installation et de configuration

4.1 Préparation du serveur Zabbix (Docker)

Cette section décrit l'installation de Docker et Docker Compose sur Ubuntu, puis la vérification du bon fonctionnement des conteneurs.

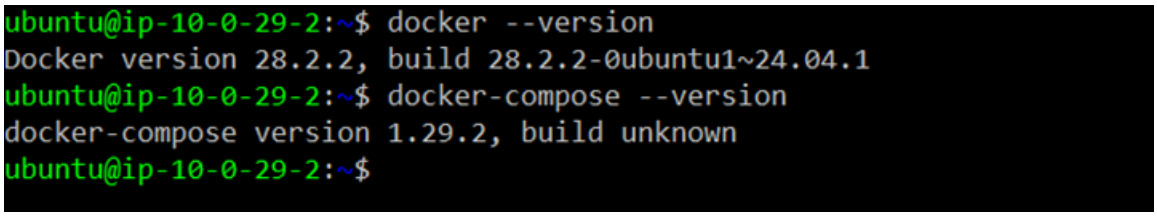

FIGURE 4.1 – Installation de Docker et Docker Compose

4.1.1 Commandes d'installation (exemple)

Listing 4.1 – Installation Docker (exemple Ubuntu)

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
    /etc/apt/keyrings/docker.gpg

echo \
      "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/
   docker.gpg] https://download.docker.com/linux/ubuntu \
      $(. /etc/os-release && echo $VERSION_CODENAME) stable" \
      | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
   docker-compose-plugin
```

```
sudo systemctl enable --now docker
docker --version
docker compose version
```

FIGURE 4.2 – Vérification des conteneurs Docker

4.2 Déploiement du serveur Zabbix conteneurisé

Le serveur Zabbix est déployé via Docker Compose (serveur, interface web et base de données).

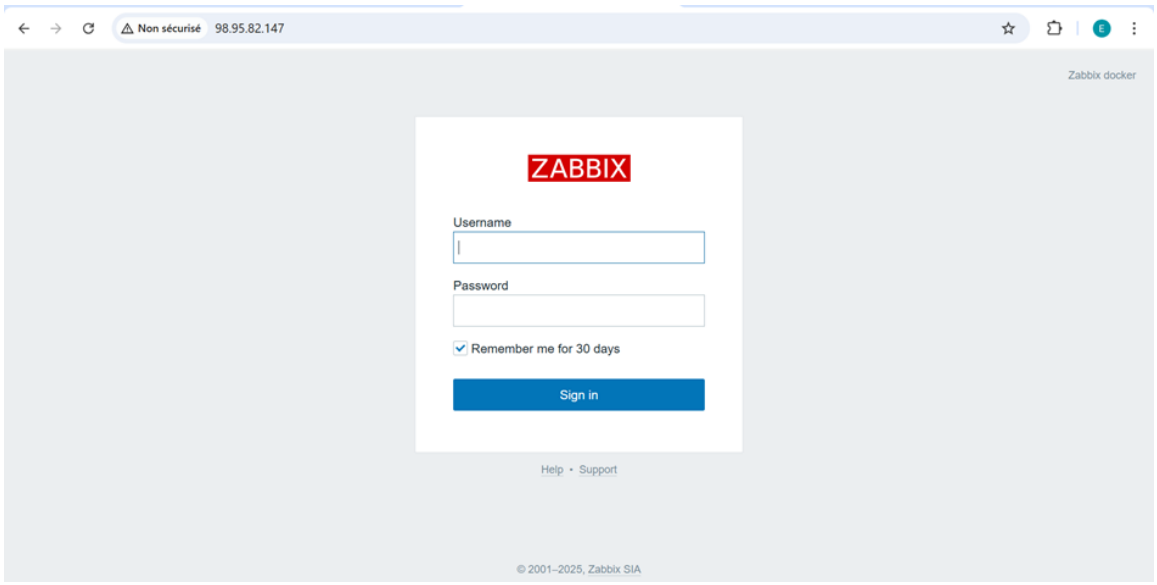

FIGURE 4.3 – Lancement des services Zabbix via Docker

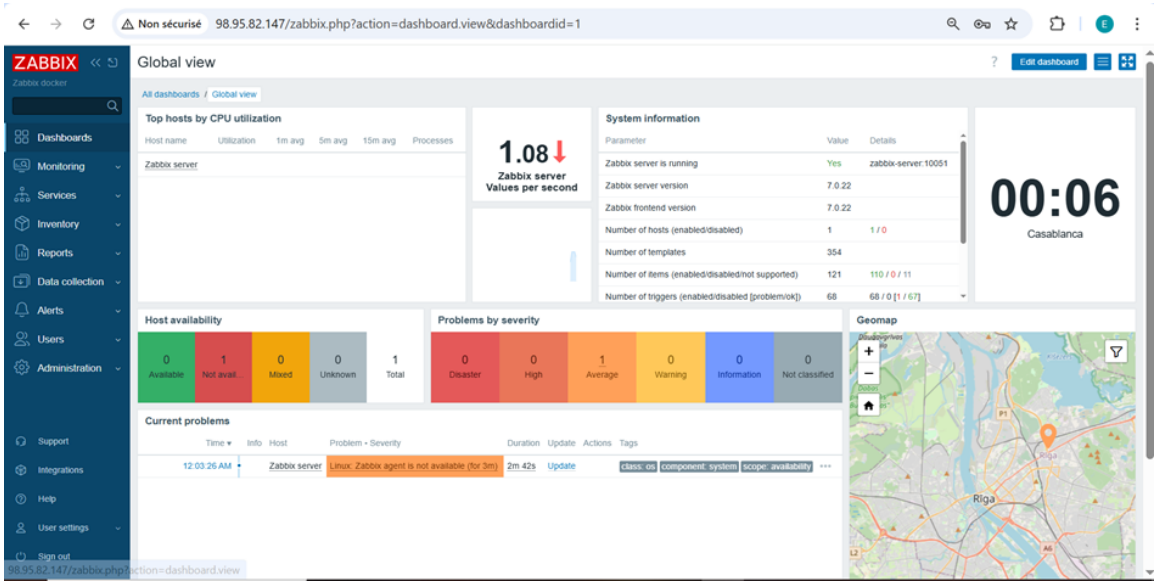

FIGURE 4.4 – Accès au tableau de bord Zabbix

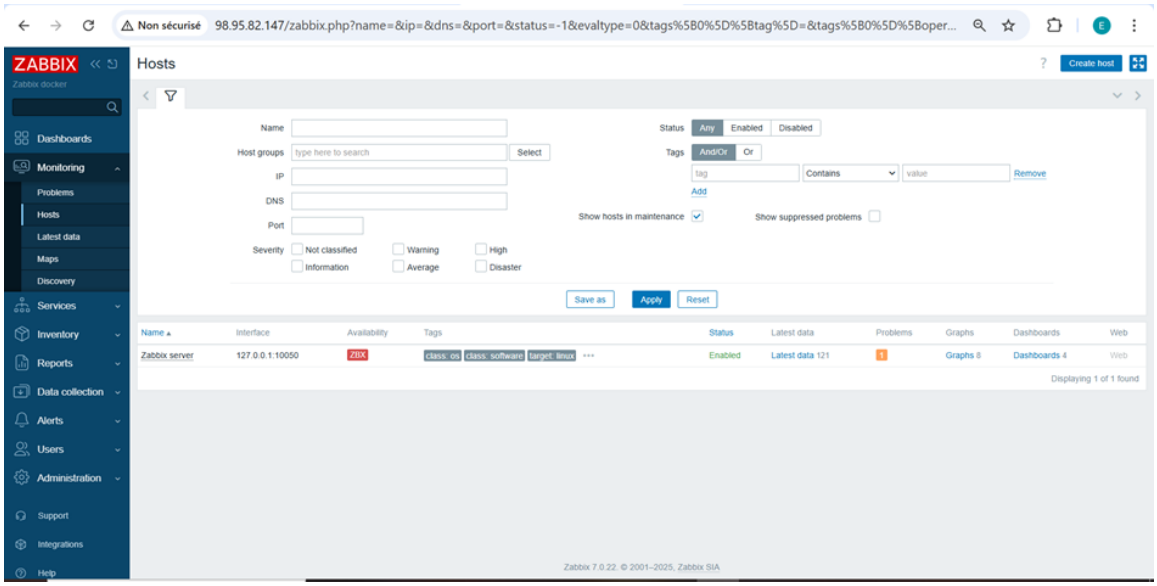

FIGURE 4.5 – Vue “Hosts” (supervision des hôtes)

4.3 Configuration du client Linux (agent Zabbix)

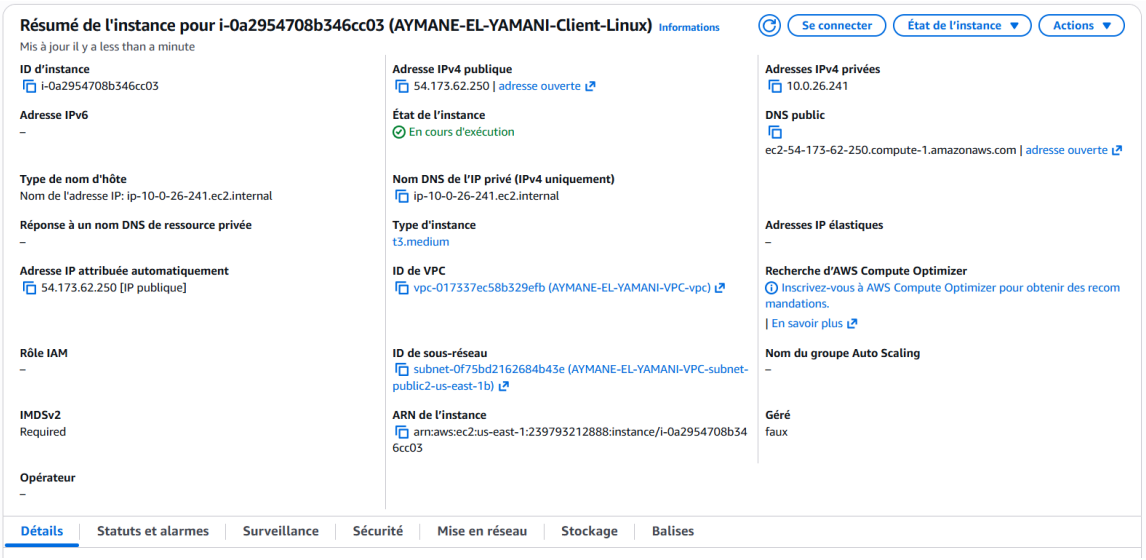

FIGURE 4.6 – Lancement de l'instance EC2 (client Linux)

4.3.1 Installation et paramétrage

Listing 4.2 – Installation de l'agent Zabbix sur Ubuntu (exemple)

```
sudo apt update
sudo apt install -y zabbix-agent

# Adapter l'IP du serveur Zabbix
sudo sed -i 's/^Server=.*/Server=<IP_SERVEUR_ZABBIX>/' /etc/zabbix/
    zabbix_agentd.conf
sudo sed -i 's/^ServerActive=.*/ServerActive=<IP_SERVEUR_ZABBIX>/' /etc/zabbix/
    zabbix_agentd.conf
sudo sed -i 's/^Hostname=.*/Hostname=client-linux/' /etc/zabbix/zabbix_agentd.
    conf

sudo systemctl enable --now zabbix-agent
sudo systemctl status zabbix-agent
```

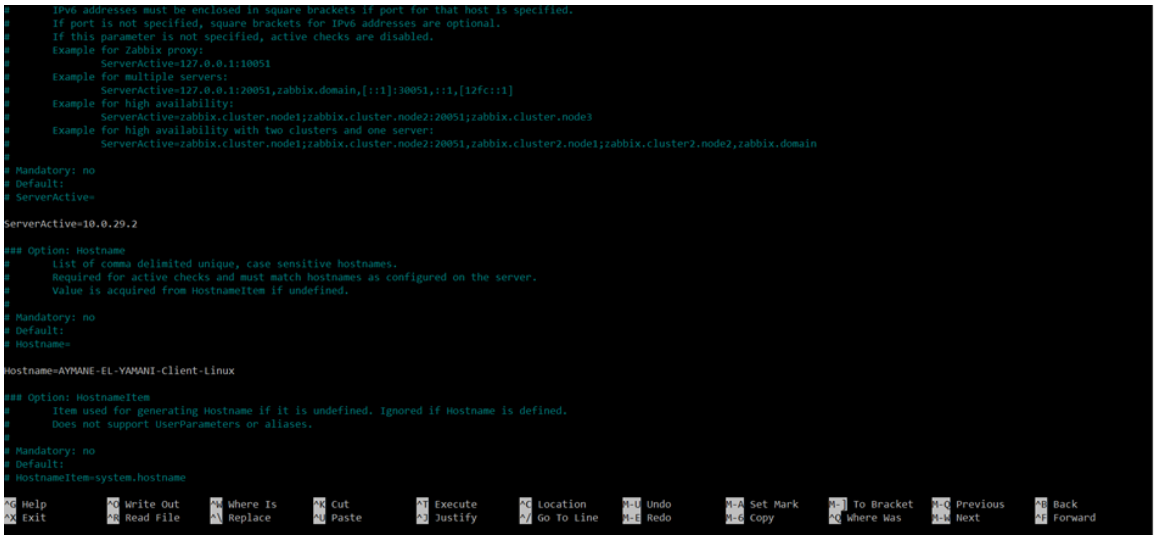

FIGURE 4.7 – Configuration “server / client / agent”

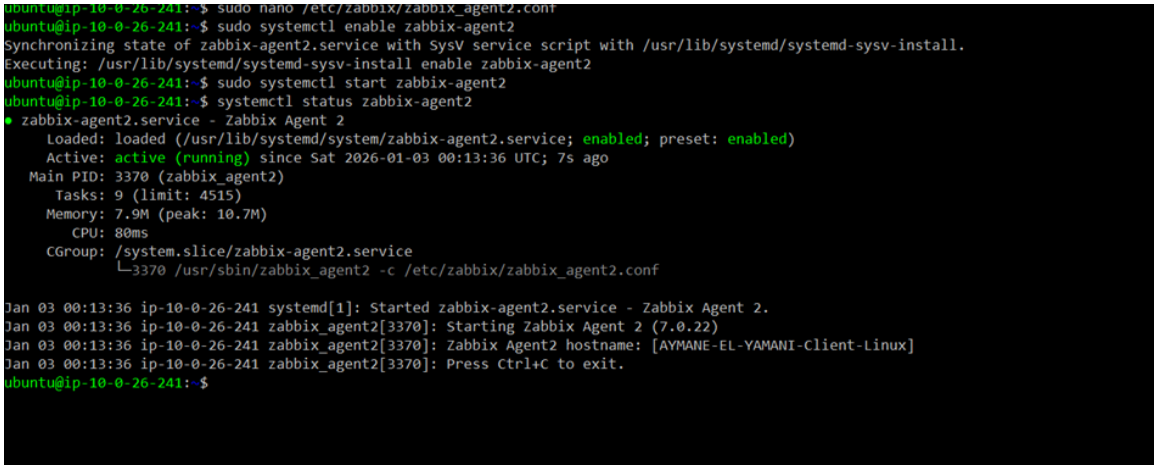

FIGURE 4.8 – Activation du service agent côté client Linux

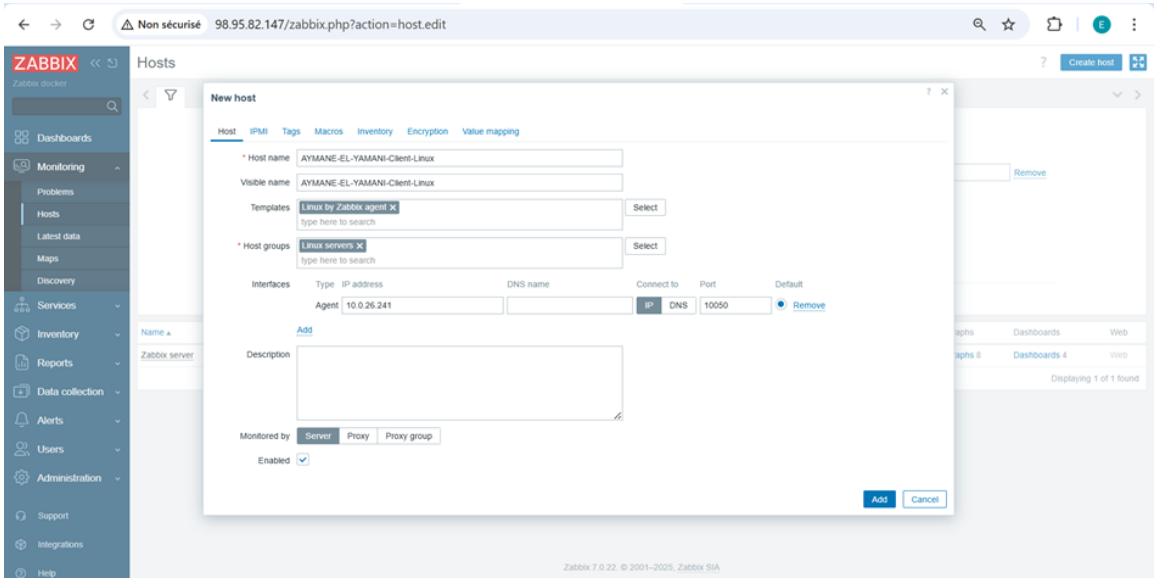

FIGURE 4.9 – Modification/validation des paramètres de l'hôte Linux

4.4 Configuration du client Windows (agent Zabbix)

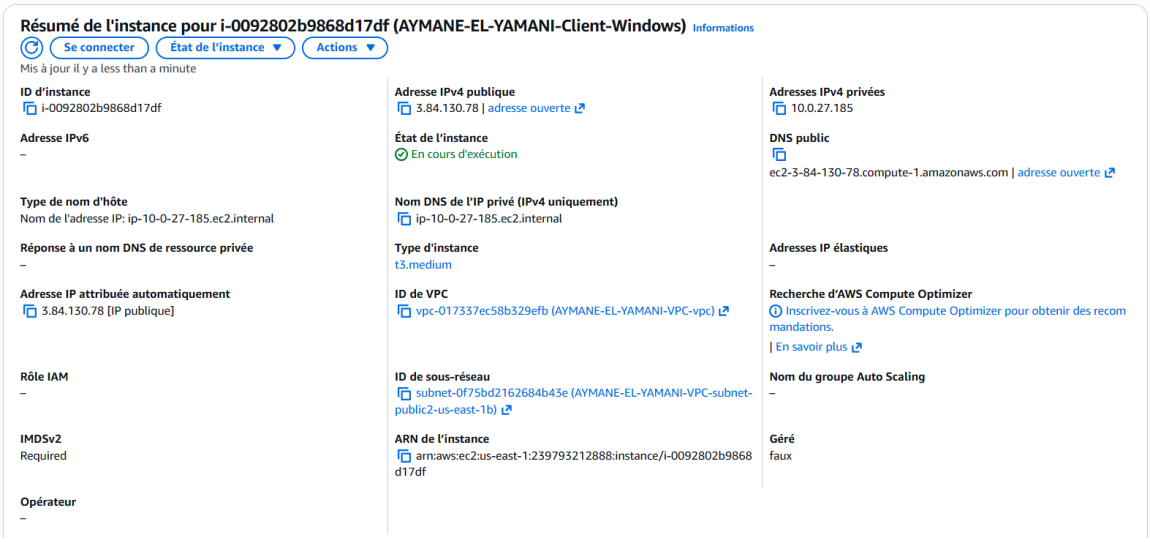

FIGURE 4.10 – Lancement de l'instance EC2 (client Windows)

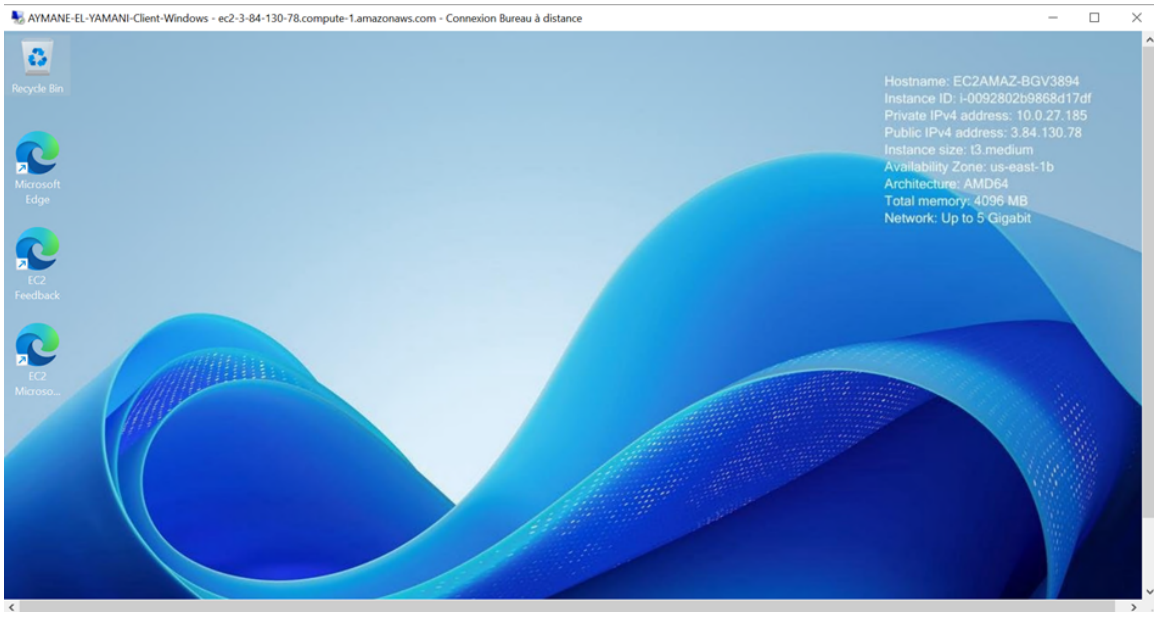

FIGURE 4.11 – Affichage et accès au client Windows

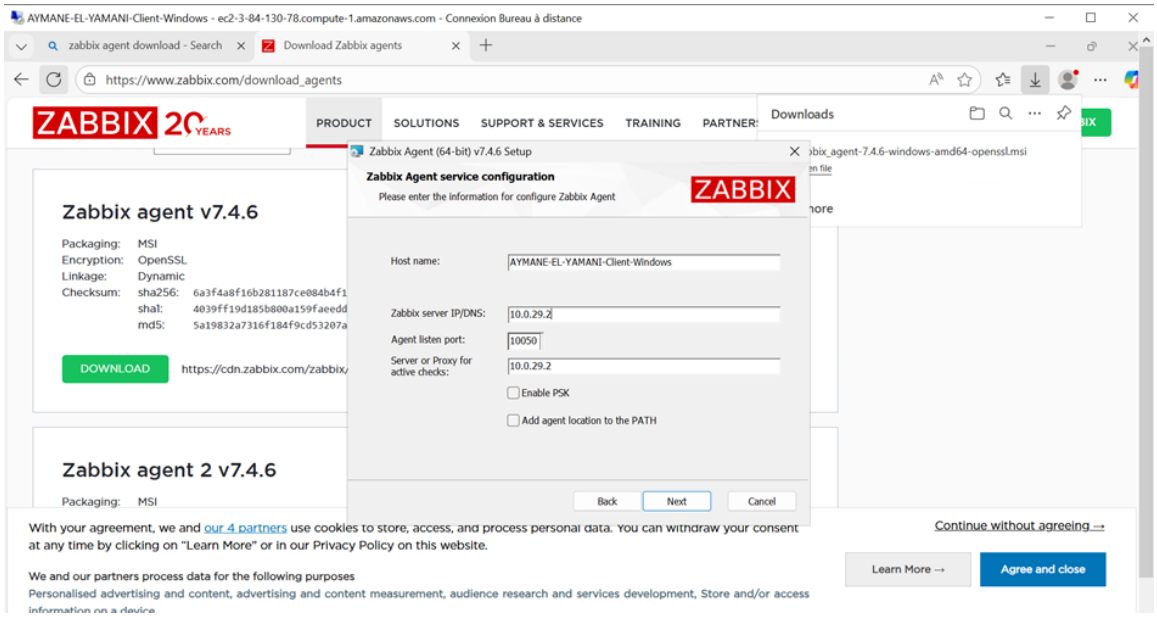

FIGURE 4.12 – Installation de l'agent Zabbix sur Windows

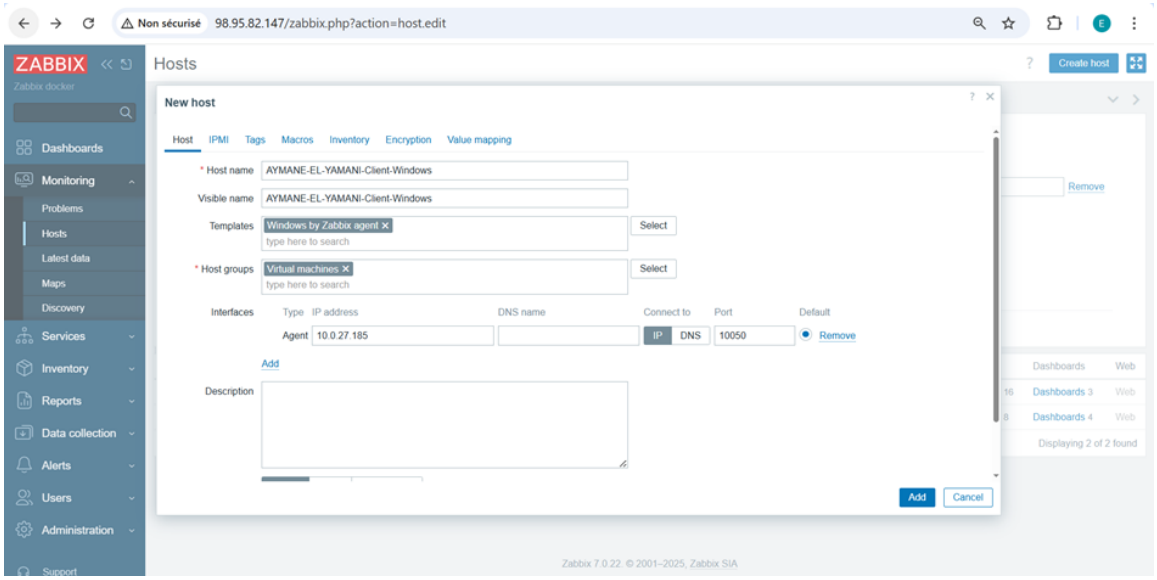

FIGURE 4.13 – Paramétrage de l'hôte Windows dans Zabbix

4.5 Monitoring et tableaux de bord

Après ajout des hôtes, la réception des données est vérifiée via le statut et les écrans "Latest data" et "Graphs".

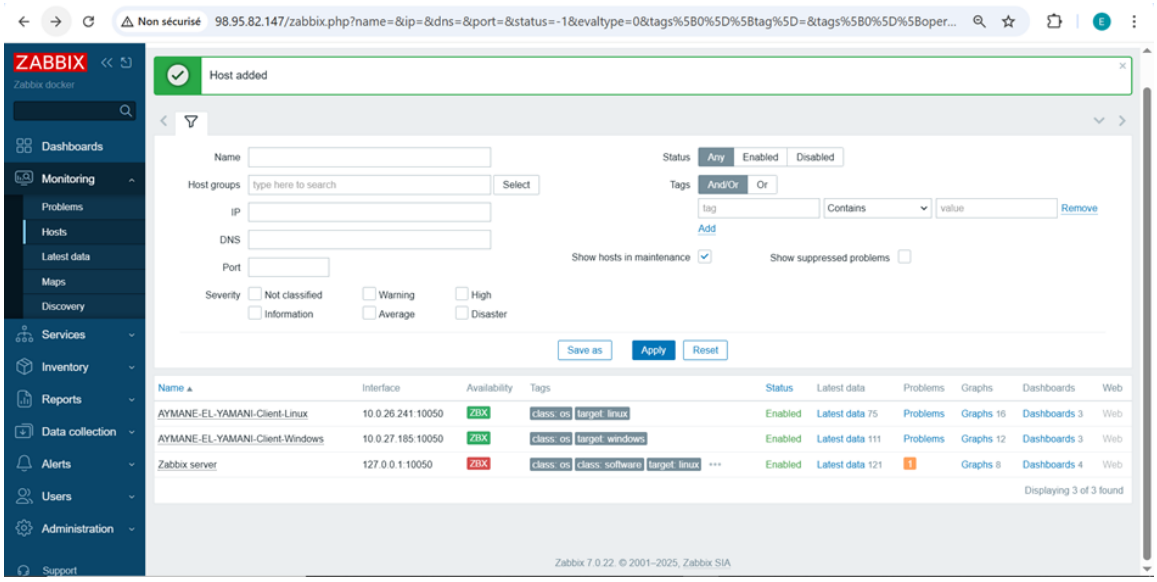

FIGURE 4.14 – Ajout des hôtes réussi

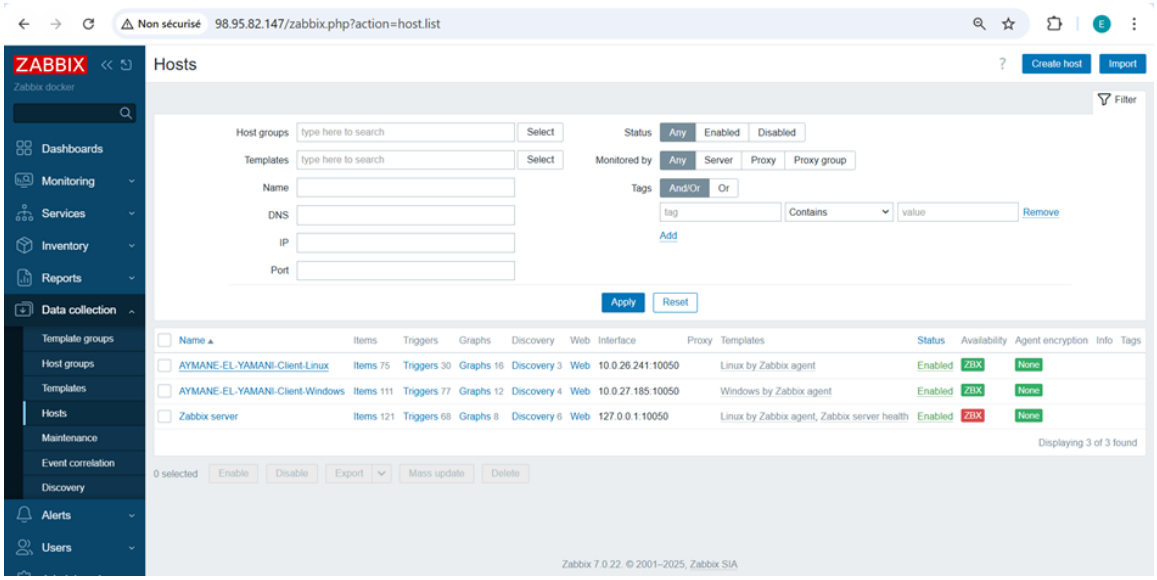

FIGURE 4.15 – Liste des hôtes (statut de connexion)

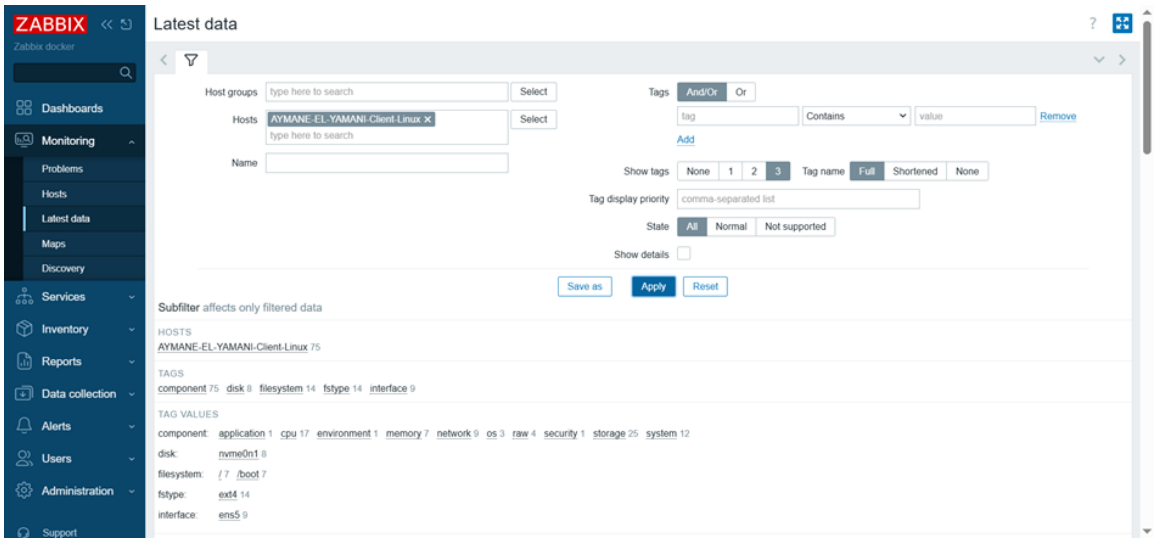

FIGURE 4.16 – “Latest data” pour le client Linux

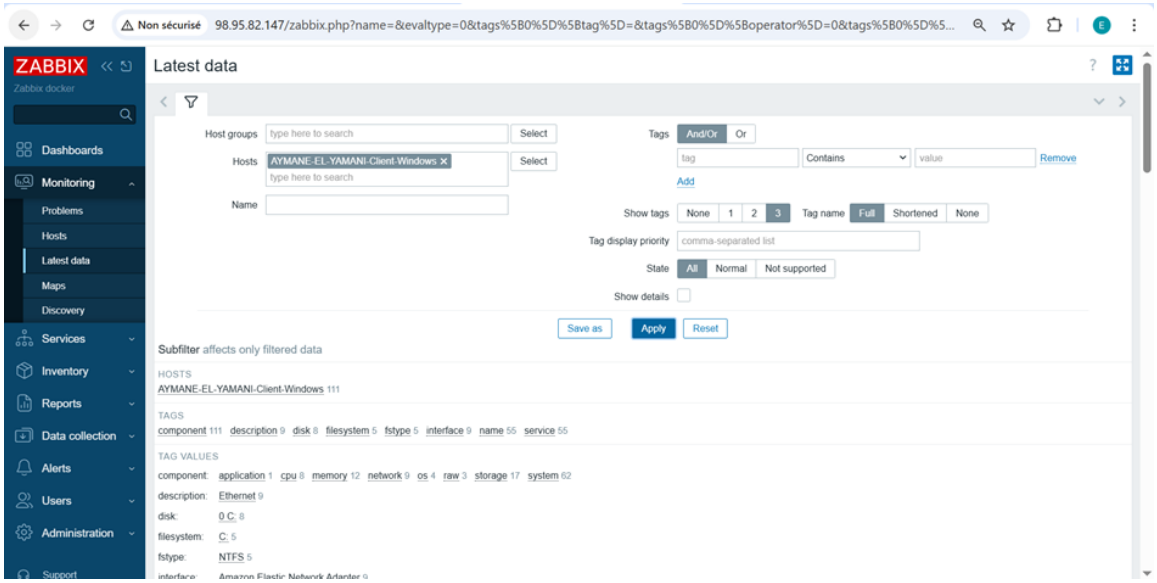

FIGURE 4.17 – “Latest data” pour le client Windows

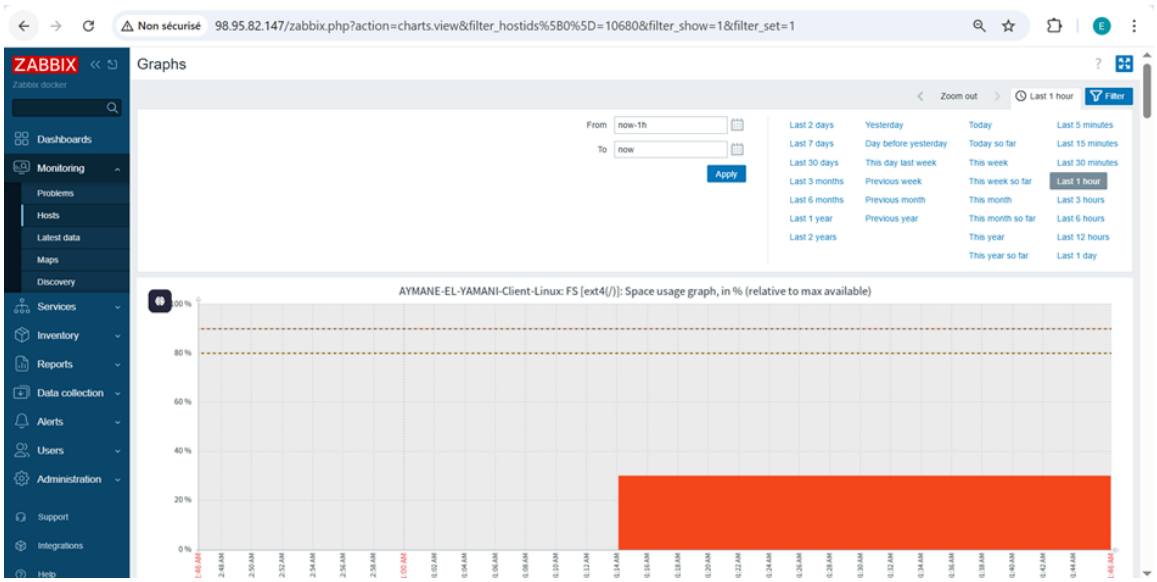

FIGURE 4.18 – Graphiques de supervision (CPU/RAM, etc.)

Chapitre 5

Problèmes rencontrés et solutions

5.1 Problèmes réseau et accès

— **Ports bloqués (10050/10051)** : vérification des règles du Security Group et des pare-feux locaux.
— **Accès web Zabbix** : s'assurer que le port 80/443 est autorisé et que l'IP publique de l'instance est correcte.

5.2 Contraintes du Learner Lab

Dans le cadre du Lab, certaines limitations sont à respecter :
— Types d'instances limités (ex. t3.medium, t3.large).
— Région recommandée : `us-east-1` (N. Virginia).
— Arrêt automatique du Lab : redémarrage nécessaire des services.

Listing 5.1 – Relance des services Docker (exemple)

```
docker compose up -d
```

— Suivi du budget : arrêt des instances hors usage.

Chapitre 6

Résultats

Les résultats attendus sont atteints lorsque :
— les hôtes Linux et Windows apparaissent dans Zabbix avec un statut sain (“ZBX” actif),
— les métriques remontent en temps réel (“Latest data”),
— des graphiques sont disponibles pour les ressources (CPU, RAM, réseau),
— une alerte peut être déclenchée et observée en supervision.

Chapitre 7

Conclusion

Ce projet démontre la mise en place d'une supervision centralisée sur AWS, basée sur Zabbix déployé via Docker. L'approche conteneurisée facilite le déploiement et la maintenance, tandis que l'installation des agents sur Linux et Windows permet d'obtenir une visibilité opérationnelle sur un parc hybride.

7.1 Perspectives

— Sécurisation renforcée (accès privé via VPN/bastion, moindre exposition publique).
— Automatisation (Infrastructure as Code : Terraform/CloudFormation).
— Haute disponibilité (multi-AZ, base de données managée, sauvegardes).

7.2 Dépôt GitHub

https://github.com/aymaneElyamani/zabbix-docker-aws-monitoring