

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

Program :

«Computer Engineering: Big Data and Cloud Computing»

Projet de fin de module :

**Mise en œuvre d'une infrastructure cloud de
supervision centralisée sous AWS**

Réalisé par ELOUAFI Abderrahmane

Encadré par KHIAT Azzedine

Année Universitaire : 2025-2026

Table des Matières

Table des Matières	2
1. Introduction	3
1.1 Contexte et problématique	3
1.2 Objectifs du projet	3
1.3 Technologies utilisées	3
2. Architecture Globale du Projet	5
2.1 Vue d'ensemble de l'architecture	5
2.2 Schéma d'architecture AWS	5
2.3 Avantages de cette architecture	5
3. Configuration Réseau AWS	7
3.1 Création du VPC	7
3.2 Configuration du Security Group	8
4. Déploiement des Instances EC2	10
4.1 Spécifications des instances	10
5. Déploiement Zabbix avec Docker	11
5.1 Architecture Docker	11
5.2 Installation de Docker	11
5.3 Configuration Docker Compose	11
6. Configuration des Agents Zabbix	13
6.1 Flux de communication Zabbix	13
6.2 Installation de l'agent Linux	13
6.3 Installation de l'agent Windows	14
7. Monitoring et Tableaux de Bord	15
7.1 Ajout des hôtes dans Zabbix	15
7.2 Visualisation des données en temps réel	15
8. Conclusion	17
8.1 Résumé des réalisations	17
8.2 Difficultés rencontrées et solutions	17
8.3 Compétences acquises	17

1. Introduction

1.1 Contexte et problématique

Dans un contexte où les infrastructures informatiques deviennent de plus en plus complexes et distribuées, la supervision centralisée des systèmes est devenue un enjeu majeur pour les entreprises. Les environnements hybrides, mélangeant serveurs Linux et Windows, posent des défis particuliers en termes de monitoring unifié et de détection proactive des incidents.

Amazon Web Services (AWS) offre une plateforme cloud flexible permettant de déployer rapidement des infrastructures scalables et hautement disponibles. Combiné à Docker pour la conteneurisation et Zabbix pour le monitoring, nous pouvons construire une solution de supervision robuste, professionnelle et facilement maintenable.

Ce projet s'inscrit dans le cadre du module Cloud Computing et vise à mettre en pratique les concepts théoriques appris durant la formation, en déployant une infrastructure réelle de monitoring sur le cloud AWS Academy Learner Lab.

1.2 Objectifs du projet

Les objectifs principaux de ce projet couvrent plusieurs aspects techniques essentiels :

- **Infrastructure Cloud** : Maîtriser le déploiement sur Amazon Web Services (VPC, EC2, Security Groups)
- **Conteneurisation** : Utiliser Docker et Docker Compose pour orchestrer les composants Zabbix
- **Monitoring** : Configurer Zabbix Server pour la supervision centralisée de systèmes hétérogènes
- **Multi-plateforme** : Déployer des agents de monitoring sur des clients Linux et Windows
- **Visualisation** : Mettre en place des tableaux de bord et des alertes pour le suivi en temps réel
- **Documentation** : Documenter l'ensemble du processus pour assurer la reproductibilité

1.3 Technologies utilisées

Le projet s'appuie sur un ensemble de technologies modernes et éprouvées dans le domaine du cloud computing et du DevOps :

Technologie	Rôle et justification
AWS (EC2, VPC)	Plateforme cloud leader offrant flexibilité, scalabilité et haute disponibilité pour héberger l'infrastructure de monitoring.

Docker & Compose	Conteneurisation pour isoler les composants, simplifier le déploiement et garantir la portabilité.
Zabbix 6.4 LTS	Solution de monitoring open-source mature, supportant nativement Linux et Windows avec templates prédéfinis.
MySQL 8.0	Base de données relationnelle performante pour stocker l'historique des métriques et la configuration.
Ubuntu 24.04 LTS	Distribution Linux stable et largement supportée, idéale pour les environnements de production.
Windows Server 2022	Système d'exploitation serveur Microsoft pour démontrer le monitoring d'un environnement hétérogène.

 **Lien du dépôt GitHub :**

<https://github.com/elouafi-abderrahmane-2002/infrastructure-cloud-de-supervision-centralise-sous-AWS/tree/main>

2. Architecture Globale du Projet

2.1 Vue d'ensemble de l'architecture

L'architecture déployée suit les bonnes pratiques AWS en matière de sécurité et d'isolation réseau. Elle se compose d'un VPC (Virtual Private Cloud) dédié contenant un sous-réseau public où sont déployées trois instances EC2 : le serveur Zabbix central et deux clients à superviser (Linux et Windows).

Les agents Zabbix installés sur les clients communiquent avec le serveur via le port 10050, permettant la collecte des métriques système en temps réel. Cette communication s'effectue via les adresses IP privées du VPC, assurant une sécurité optimale.

2.2 Schéma d'architecture AWS

Le schéma ci-dessous illustre l'architecture complète de la solution avec les différents composants et leurs interconnexions :

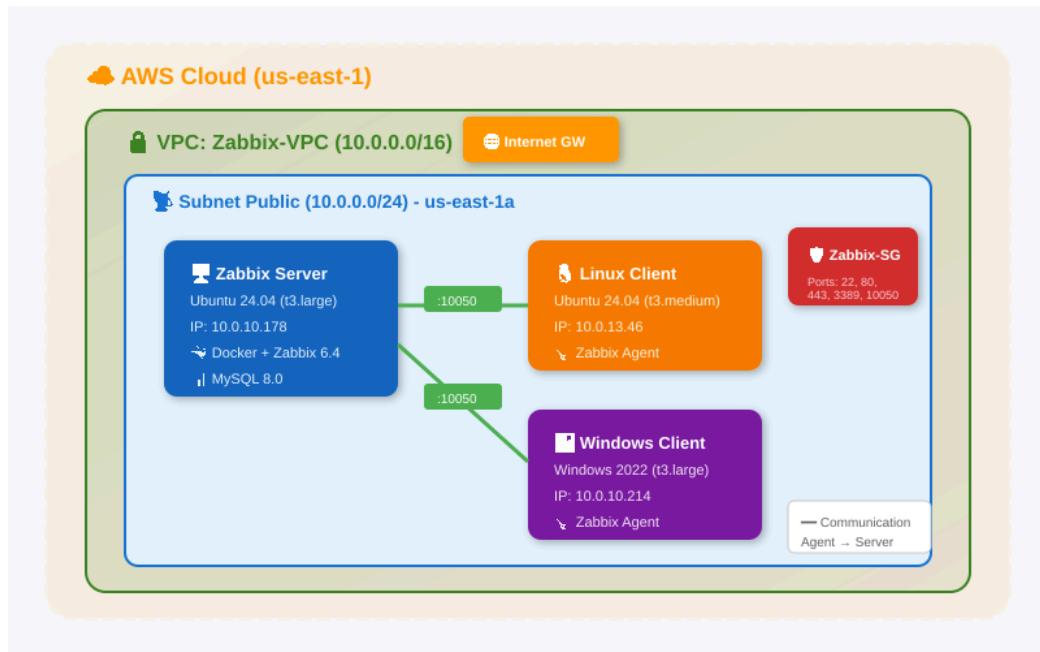


Figure 1 : Architecture globale de l'infrastructure AWS

2.3 Avantages de cette architecture

Cette architecture présente plusieurs avantages clés :

- **Isolation réseau** : Le VPC isole complètement notre infrastructure du reste d'AWS et d'Internet
- **Sécurité** : Le Security Group contrôle précisément les flux réseau autorisés

- **Scalabilité** : L'architecture permet d'ajouter facilement de nouveaux clients à superviser
- **Communication interne** : Les instances communiquent via leurs adresses IP privées

3. Configuration Réseau AWS

3.1 Création du VPC

La première étape du déploiement consiste à créer le VPC (Virtual Private Cloud) qui servira de fondation réseau pour toute l'infrastructure. Le VPC a été configuré via l'assistant AWS "VPC and more" qui crée automatiquement les ressources associées.

Paramètres de configuration du VPC :

- Nom : Zabbix-VPC
- Bloc CIDR IPv4 : 10.0.0.0/16 (65,536 adresses IP disponibles)
- Région : us-east-1 (Virginie du Nord)
- Zone de disponibilité : us-east-1a
- Sous-réseau public : 1 (pour l'accès Internet)
- Passerelle Internet : Créeée et attachée automatiquement

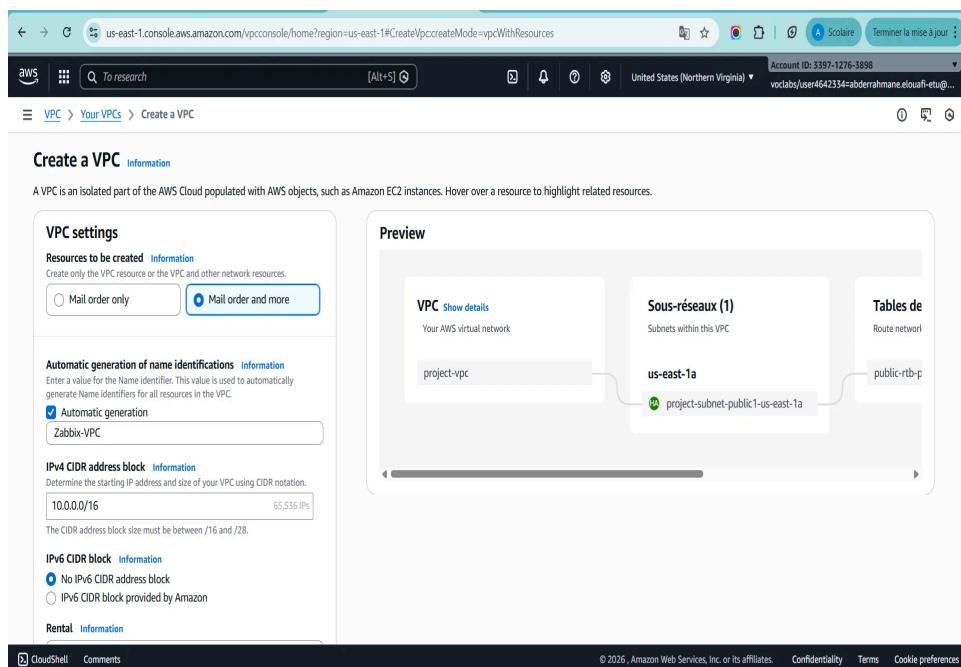


Figure 2 : Interface de configuration du VPC avec l'assistant AWS

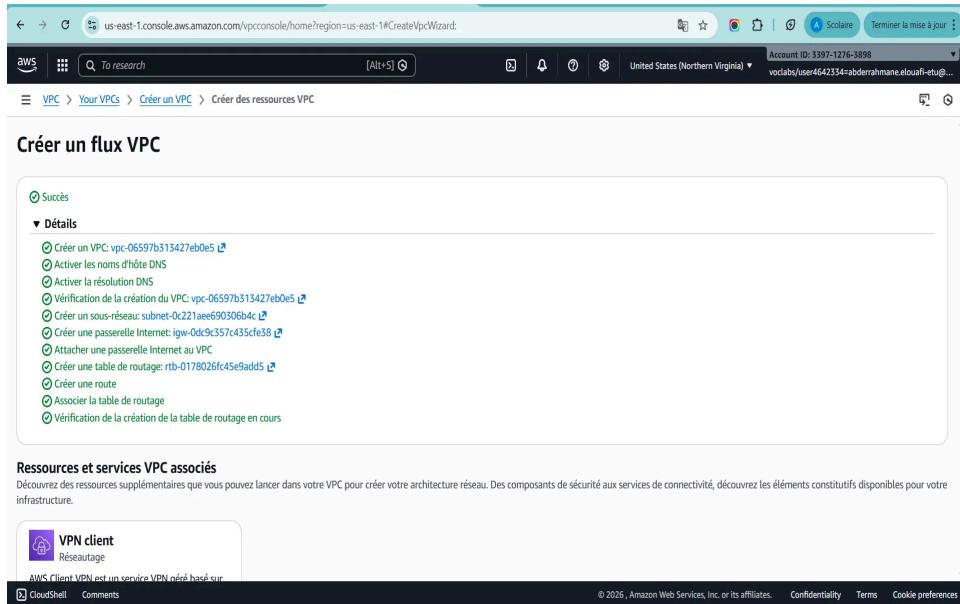


Figure 3 : Confirmation de la création réussie du VPC

3.2 Configuration du Security Group

Le Security Group agit comme un pare-feu virtuel contrôlant le trafic entrant et sortant des instances EC2. Une configuration précise est essentielle pour assurer à la fois la sécurité et le bon fonctionnement de la communication entre les composants Zabbix.

Type	Port	Source	Description
SSH	22	0.0.0.0/0	Accès SSH pour l'administration Linux
HTTP	80	0.0.0.0/0	Interface web Zabbix
HTTPS	443	0.0.0.0/0	Accès SSL sécurisé
Custom TCP	10050	0.0.0.0/0	Communication Zabbix Agent (passif)
Custom TCP	10051	0.0.0.0/0	Zabbix Trapper (actif)
RDP	3389	0.0.0.0/0	Bureau à distance Windows

The screenshot shows the AWS CloudFormation console with a success message: "Le groupe de sécurité (sg-05ecf791ada3be099 | Zabbix-SG) a été créé avec succès." Below this, the security group details are shown:

- Détails**
- Nom du groupe de sécurité**: Zabbix-SG
- ID du groupe de sécurité**: sg-05ecf791ada3be099
- Description**: Security group for Zabbix monitoring
- ID de VPC**: vpc-06597b313427eb0e5
- Propriétaire**: 339712763898
- Nombre de règles entrantes**: 6 Entrées d'autorisation
- Nombre de règles sortantes**: 1 Entrée d'autorisation

Below the details, there are tabs for "Règles entrantes", "Règles sortantes", "Partage", "Associations VPC", and "Balises". The "Règles entrantes" tab is selected, showing a table with 6 rows of incoming rules:

Name	ID de règle de groupe	Version IP	Type	Protocole	Plage de ports	Source	Description
-	sgr-0ea65f736ac82c6a1	IPv4	RDP	TCP	3389	0.0.0.0/0	Windows RDP
-	sgr-07b72de4594811d5c	IPv4	HTTPS	TCP	443	0.0.0.0/0	Zabbix Web SSL
-	sgr-08ad851dcfb94549b	IPv4	TCP personnalisé	TCP	10050	0.0.0.0/0	Zabbix Agent
-	sgr-03e8f71be72eb0777	IPv4	SSH	TCP	22	0.0.0.0/0	SSH acces
-	sgr-048e5baeb0522b23e	IPv4	HTTP	TCP	80	0.0.0.0/0	zabbix web
-	sgr-0d451f064767acab	IPv4	TCP personnalisé	TCP	10051	0.0.0.0/0	Zabbix Server

Figure 4 : Règles entrantes du Security Group Zabbix-SG

4. Déploiement des Instances EC2

4.1 Spécifications des instances

Trois instances EC2 ont été déployées avec des configurations adaptées à leurs rôles respectifs. Le choix des types d'instances tient compte des exigences en ressources de chaque composant et des contraintes du Learner Lab AWS Academy.

Instance	Type	OS	IP Privée	Rôle
Elouafi-Zabbix-Server	t3.large	Ubuntu 24.04	10.0.10.178	Serveur Zabbix
Linux-Client-elouafi	t3.medium	Ubuntu 24.04	10.0.13.46	Client Linux
Windows-Client-elouafi	t3.large	Windows 2022	10.0.10.214	Client Windows

⚠ Note : Le type t3.large a été choisi pour le serveur Zabbix car Docker, Zabbix Server, l'interface web et MySQL nécessitent des ressources conséquentes. Windows Server requiert également 8 Go de RAM minimum.

Figure 5 : Les trois instances EC2 en cours d'exécution

Figure 6 : Confirmation du lancement réussi de l'instance Zabbix Server

5. Déploiement Zabbix avec Docker

5.1 Architecture Docker

Zabbix est déployé sous forme de conteneurs Docker pour faciliter l'installation, la maintenance et les mises à jour. L'architecture Docker comprend trois conteneurs interconnectés qui communiquent via un réseau Docker interne dédié.

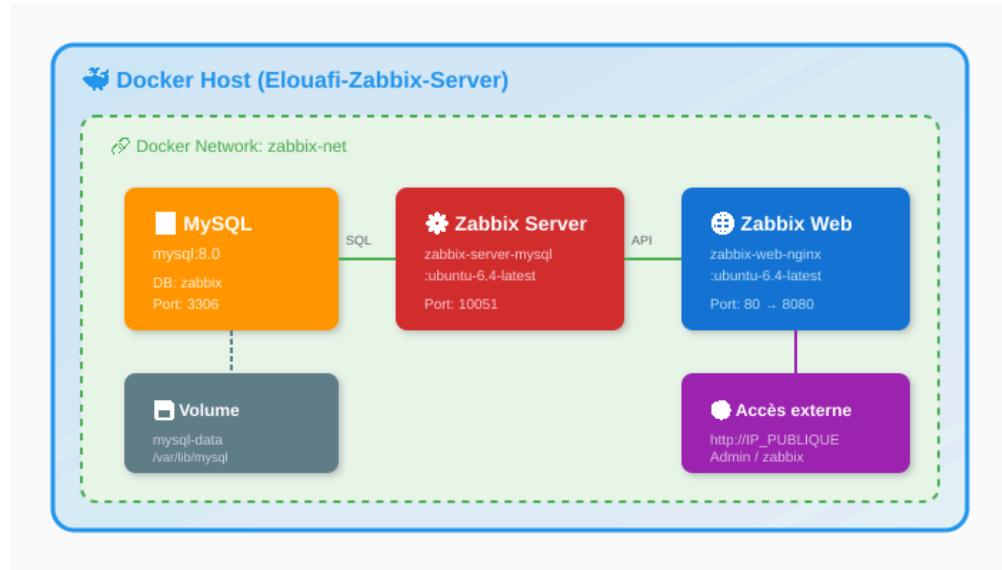


Figure 7 : Architecture des conteneurs Docker pour Zabbix

Les trois conteneurs déployés sont :

- **mysql-server** : Base de données MySQL 8.0 stockant configuration et historique Zabbix
- **zabbix-server** : Moteur principal collectant les données et évaluant les triggers
- **zabbix-web** : Interface web Nginx pour visualisation et configuration

5.2 Installation de Docker

Docker et Docker Compose ont été installés sur le serveur avec les commandes suivantes :

```
# Mise à jour du système
sudo apt update && sudo apt upgrade -y
# Installation de Docker
sudo apt install docker.io docker-compose -y
sudo systemctl enable docker && sudo systemctl start docker
```

5.3 Configuration Docker Compose

Le fichier docker-compose.yml définit l'ensemble de l'infrastructure Zabbix :

```
version: '3.5'
services:
```

```

mysql-server:
  image: mysql:8.0
  environment: [MYSQL_DATABASE, MYSQL_USER, ...]
zabbix-server:
  image: zabbix/zabbix-server-mysql:ubuntu-6.4-latest
  ports: ["10051:10051"]
zabbix-web:
  image: zabbix/zabbix-web-nginx-mysql:ubuntu-6.4-latest
  ports: ["80:8080"]

```

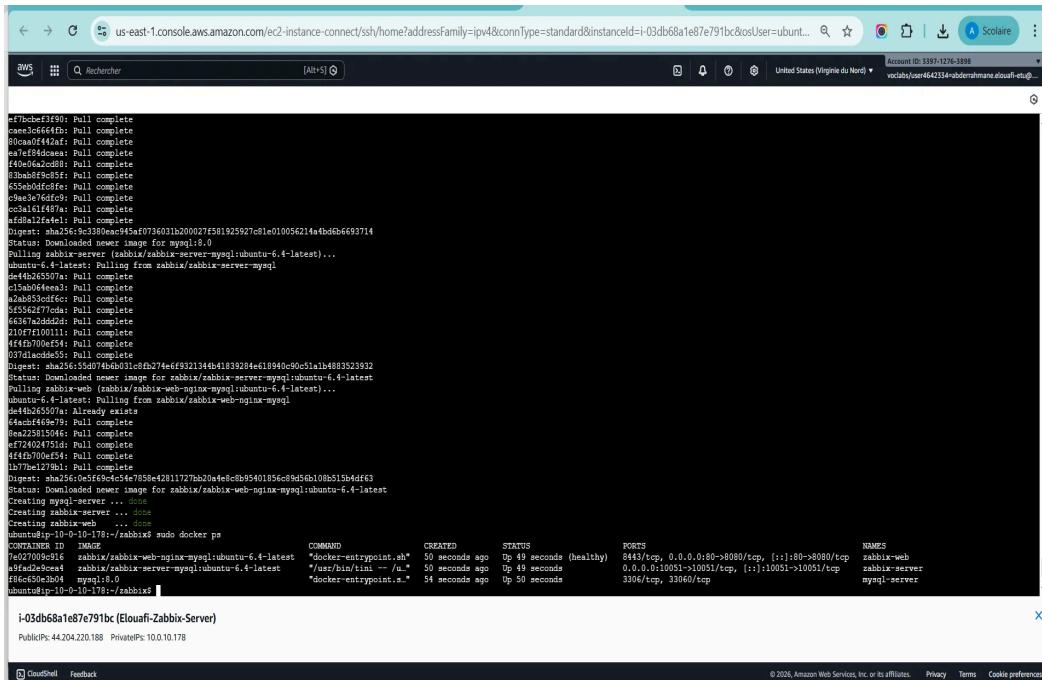


Figure 8 : Déploiement des conteneurs avec Docker Compose

6. Configuration des Agents Zabbix

6.1 Flux de communication Zabbix

Avant de configurer les agents, il est important de comprendre le flux de données dans l'architecture Zabbix. Les agents collectent les métriques système et les transmettent au serveur qui les stocke en base de données pour affichage dans l'interface web.

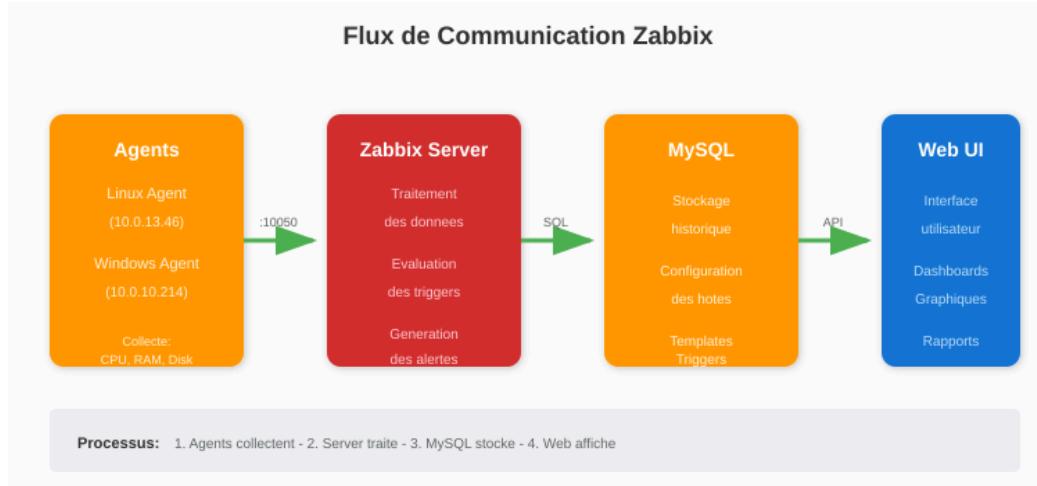


Figure 9 : Flux de communication entre les composants Zabbix

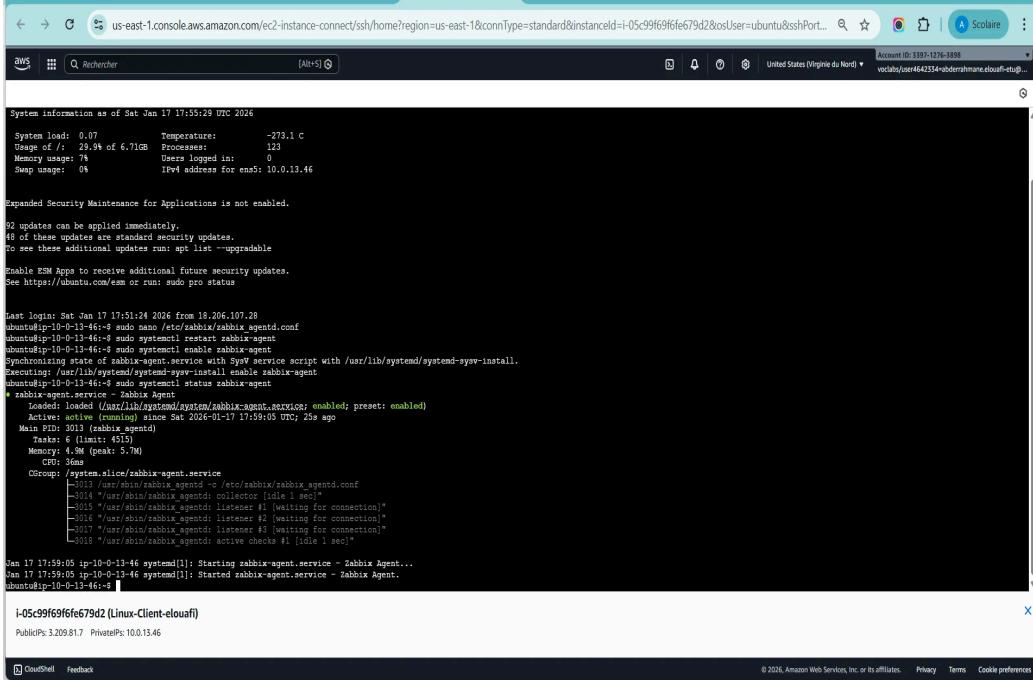
6.2 Installation de l'agent Linux

L'agent Zabbix a été installé sur le client Linux en utilisant le dépôt officiel :

```
# Télécharger le dépôt Zabbix
wget https://repo.zabbix.com/.../zabbix-release_6.4-1+ubuntu24.04_all.deb
sudo dpkg -i zabbix-release_6.4-1+ubuntu24.04_all.deb
sudo apt update && sudo apt install zabbix-agent -y
```

Configuration dans /etc/zabbix/zabbix_agentd.conf :

```
Server=10.0.10.178
ServerActive=10.0.10.178
Hostname=Linux-Client-elouafi
```



```

System information as of Sat Jan 17 17:55:28 UTC 2026
System load: 0.07      Temperature:          -273.1 C
Usage of /: 29.9% of 6.71GB  Processes:           123
Memory usage: 7%        Users logged in:       0
Swap usage: 0%          IPv4 address for ena5: 10.0.13.46

Expanded Security Maintenance for Applications is not enabled.
92 updates can be applied immediately.
48 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat Jan 17 17:51:24 2026 from 18.206.107.28
ubuntu@ip-10-0-13-46:~$ /etc/init.d/zabbix-agent start
ubuntu@ip-10-0-13-46:~$ sudo systemctl restart zabbix-agent
ubuntu@ip-10-0-13-46:~$ 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-10-0-13-46:~$ sudo systemctl status zabbix-agent
● zabbix-agent.service - Zabbix Agent
   Loaded: loaded (/usr/lib/systemd/system/zabbix-agent.service; enabled; preset: enabled)
     Active: active (running) since Sat 2026-01-17 17:59:05 UTC; 25s ago
       Main PID: 3013 (zabbix-agent)
         Tasks: 1 (limit: 4515)
        Memory: 4.5M (peak: 5.7M)
        CPU: 36ms
      CGroup: /system.slice/zabbix-agent.service
              └─3013 /usr/sbin/zabbix_agend -> /etc/zabbix/zabbix_agend.conf

Jan 17 17:59:05 ip-10-0-13-46 systemd[1]: Starting zabbix-agent.service - Zabbix Agent...
Jan 17 17:59:05 ip-10-0-13-46 systemd[1]: Started zabbix-agent.service - Zabbix Agent.

ubuntu@ip-10-0-13-46:~$ i-05c9f69f6fe679d2 (Linux-Client-elouafi)
PublicIP: 3.209.81.7 PrivateIP: 10.0.13.46

```

Figure 10 : Agent Zabbix actif (running) sur le client Linux

6.3 Installation de l'agent Windows

L'agent Zabbix pour Windows a été téléchargé depuis le site officiel (zabbix_agent-6.4.11-windows-amd64-openssl.msi) et installé via l'assistant graphique. Les paramètres configurés lors de l'installation sont l'adresse IP du serveur Zabbix (10.0.10.178) et le hostname (Windows-Client-elouafi).

7. Monitoring et Tableaux de Bord

7.1 Ajout des hôtes dans Zabbix

Les trois hôtes ont été ajoutés dans l'interface web Zabbix via le menu Data Collection > Hosts. Chaque hôte a été configuré avec son template approprié (Linux by Zabbix agent ou Windows by Zabbix agent) et son adresse IP privée.

Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent encryption	Info	Tags
Linux-Client-elouafi	Items 75	Triggers 30	Graphs 16	Discovery 3	Web	10.0.13.46:10050		Linux by Zabbix agent	Enabled	ZBX	None		
Windows-Client-elouafi	Items 105	Triggers 71	Graphs 12	Discovery 4	Web	10.0.10.214:10050		Windows by Zabbix agent	Enabled	ZBX	None		
Zabbix server	Items 136	Triggers 74	Graphs 27	Discovery 5	Web	10.0.10.178:10050		Linux by Zabbix agent, Zabbix server health	Enabled	ZBX	None		

Figure 11 : Les trois hôtes avec le statut ZBX vert (agents connectés)

L'indicateur ZBX en vert confirme que les agents sont correctement connectés et communiquent avec le serveur Zabbix. Cet indicateur passe au rouge en cas de problème de connexion.

7.2 Visualisation des données en temps réel

Une fois les agents connectés, Zabbix collecte automatiquement les métriques système grâce aux templates prédéfinis. Les données sont visibles dans le menu Monitoring > Latest Data et incluent : utilisation CPU, mémoire RAM, espace disque, trafic réseau, uptime, etc.

The screenshot shows the Zabbix 'Latest data' interface. On the left, a sidebar navigation menu includes 'Dashboards', 'Monitoring' (selected), 'Hosts', 'Latest data' (selected), 'Maps', 'Discovery', 'Services', 'Inventory', 'Reports', 'Data collection', 'Alerts', 'Users', 'Administration', 'Support', and 'Integrations'. The main panel displays a search bar at the top with 'Host groups' and 'Tags' filters. Under 'Hosts', 'Linux-Client-elouafi' is selected. Below this, sections for 'TAGS', 'TAG VALUES', and 'DATA' are shown. The 'DATA' section lists metrics for 'Linux-Client-elouafi' with columns for Host, Name, Last check, Last value, Change, Tags, and Info. One metric is highlighted: 'FS [/boot]: Get data' with a value of '1s' and tags 'component: raw', 'component: storage', 'filesystem: /boot', 'fstype: ext4', and 'component: raw'.

Figure 12 : Interface Latest Data avec les filtres par tags

Host	Name	Last check	Last value	Change	Tags	Info
Linux-Client-elouafi	FS [/boot]: Get data	33s	{"\$name": "/boot", ...}		component: raw component: storage filesystem: /boot ...	History
Linux-Client-elouafi	FS [/boot]: Inodes Free, in %	33s	98.9726 %		component: storage filesystem: /boot fstype: ext4	Graph
Linux-Client-elouafi	FS [/boot]: Option: Read-only	33s	0		component: storage filesystem: /boot fstype: ext4	Graph
Linux-Client-elouafi	FS [/boot]: Space: Available	33s	729.84 MB		component: storage filesystem: /boot fstype: ext4	Graph
Linux-Client-elouafi	FS [/boot]: Space: Total	33s	880.39 MB		component: storage filesystem: /boot fstype: ext4	Graph
Linux-Client-elouafi	FS [/boot]: Space: Used	33s	88.9 MB		component: storage filesystem: /boot fstype: ext4	Graph
Linux-Client-elouafi	FS [/boot]: Space: Used, in %	33s	10.8579 %		component: storage filesystem: /boot fstype: ext4	Graph
Linux-Client-elouafi	FS [/]: Get data	33s	{"\$name": "/", "optio...}		component: raw component: storage filesystem: / ...	History
Linux-Client-elouafi	FS [/]: Inodes: Free, in %	33s	91.0118 %		component: storage filesystem: / fstype: ext4	Graph
Linux-Client-elouafi	FS [/]: Option: Read-only	33s	0		component: storage filesystem: / fstype: ext4	Graph
Linux-Client-elouafi	FS [/]: Space: Available	33s	4.6 GB		component: storage filesystem: / fstype: ext4	Graph
Linux-Client-elouafi	FS [/]: Space: Total	33s	6.71 GB		component: storage filesystem: / fstype: ext4	Graph
Linux-Client-elouafi	FS [/]: Space: Used	33s	2.09 GB		component: storage filesystem: / fstype: ext4	Graph
Linux-Client-elouafi	FS [/]: Space: Used, in %	33s	31.2146 %		component: storage filesystem: / fstype: ext4	Graph
Linux-Client-elouafi	Interface ens5: Bits received	1m 32s	2.17 Kbps	+136 bps	component: network interface: ens5	Graph
Linux-Client-elouafi	Interface ens5: Bits sent	1m 29s	3.89 Kbps	+152 bps	component: network interface: ens5	Graph
Linux-Client-elouafi	Interface ens5: Inbound packets discarded	1m 34s	0		component: network interface: ens5	Graph
Linux-Client-elouafi	Interface ens5: Inbound packets with errors	1m 33s	0		component: network interface: ens5	Graph
Linux-Client-elouafi	Interface ens5: Interface type	1h 45m 26s	Ethernet (1)		component: network interface: ens5	Graph

Figure 13 : Détail des métriques collectées (filesystem, réseau, CPU)

8. Conclusion

8.1 Résumé des réalisations

Ce projet a permis de mettre en œuvre avec succès une infrastructure de monitoring centralisée sur AWS. Les objectifs initiaux ont été atteints et l'ensemble du système est opérationnel.

- ✓ Déploiement d'une infrastructure cloud AWS complète (VPC, EC2, Security Groups)
- ✓ Conteneurisation de Zabbix Server avec Docker Compose (3 conteneurs)
- ✓ Configuration des agents sur un parc hybride Linux et Windows
- ✓ Mise en place du monitoring en temps réel avec collecte automatique

8.2 Difficultés rencontrées et solutions

Difficulté rencontrée	Solution appliquée
Agent non trouvé dans dépôts Ubuntu	Ajout du dépôt officiel Zabbix
Erreur "access restrictions" agent	Ajout réseau Docker dans paramètre Server=
IP publique change après redémarrage Lab	Utilisation des IPs privées pour communication

8.3 Compétences acquises

Ce projet a permis de développer des compétences pratiques essentielles dans le domaine du Cloud Computing et du DevOps :

- **AWS** : Création et gestion de VPC, EC2, Security Groups
- **Docker** : Conteneurisation, Docker Compose, gestion volumes et réseaux
- **Monitoring** : Configuration Zabbix, déploiement agents multi-plateformes
- **Administration** : Linux/Windows, configuration réseau, gestion services

 **Lien du dépôt GitHub :**

<https://github.com/elouafi-abderrahmane-2002/infrastructure-cloud-de-supervision-centralisee-sous-AWS/tree/main>