



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 :
FELLAH Youssef

Encadré par :
Prof. KHIAT Azeddine

Remerciements

Je tiens tout d'abord à remercier mon encadrant, **M. Azeddine KHIAT**, pour ses directives claires et son accompagnement pédagogique tout au long de ce module de Cloud Computing.

Ce projet m'a permis de consolider mes compétences en administration système et en architecture cloud, en mettant en pratique les concepts théoriques abordés en cours, notamment sur les services AWS (EC2, VPC) et la conteneurisation.

Avant-propos

La supervision des infrastructures informatiques est devenue une composante indispensable de la stratégie opérationnelle des entreprises modernes. Avec la migration croissante vers le Cloud, la capacité à surveiller en temps réel la disponibilité et la performance des services est un enjeu critique pour garantir la continuité d'activité.

Ce projet, réalisé dans le cadre du module de **Cloud Computing**, a pour vocation de simuler un environnement de production réel hébergé sur Amazon Web Services (AWS). Il ne s'agit pas uniquement de déployer des instances, mais de construire une architecture résiliente et surveillée, capable d'alerter les administrateurs au moindre incident.

Le choix de la solution **Zabbix**, couplée à la technologie de conteneurisation **Docker**, répond à un besoin d'agilité et de standardisation, compétences clés pour un futur ingénieur en informatique. Ce rapport retrace ainsi la démarche technique adoptée, depuis la conception de l'architecture réseau jusqu'à l'analyse des métriques de performance.

Table des matières

1	Introduction	4
1.1	Outils et Technologies	4
1.2	Ressources du Projet (GitHub)	4
2	Architecture Réseau	4
2.1	Création du VPC	4
2.2	Groupes de Sécurité	5
3	Architecture des Instances EC2	5
4	Déploiement du Serveur Zabbix	6
4.1	Installation de Docker	6
4.2	Configuration de l'Orchestrator	7
4.3	Lancement et Validation	8
4.4	Accès à l'Interface Web	8
5	Configuration des Clients (Agents)	9
5.1	Client Linux (Ubuntu)	9
5.1.1	Installation des paquets	9
5.1.2	Configuration de l'Agent	10
5.1.3	Déclaration de l'hôte dans Zabbix	11
5.2	Client Windows (Windows Server 2022)	11
5.2.1	Accès à l'instance	11
5.2.2	Installation et Configuration de l'Agent	12
6	Monitoring et Tableaux de Bord	13
6.1	Validation de la Connectivité (Disponibilité)	13
6.2	Visualisation des Données (Graphiques)	14
7	Conclusion	15
7.1	Bilan Technique	15
7.2	Difficultés Rencontrées et Solutions	15
8	Webographie	16

Table des figures

1	Aperçu de la topologie du VPC	4
2	Validation de la création des ressources VPC	5
3	Configuration des règles entrantes (Inbound Rules)	5
4	Instances EC2 en cours d'exécution (Running)	6
5	Installation des paquets Docker sur le terminal	7
6	Configuration du fichier docker-compose.yml	8
7	Validation des conteneurs actifs via docker ps	8
8	Page de connexion Zabbix	9
9	Installation réussie du paquet zabbix-agent via apt	10
10	Configuration du Hostname et de l'IP du serveur dans zabbix_agentd.conf	10
11	Redémarrage du service pour appliquer la configuration	11
12	Ajout de l'hôte Linux dans l'interface de gestion Zabbix	11
13	Connexion RDP à l'instance Windows Server	12
14	Configuration de l'agent Zabbix via l'installateur MSI	13
15	Vue globale des hôtes : Le statut "Vert" (ZBX) valide la communication	14
16	Graphiques d'utilisation des disques sur le Client Linux	14

Liste des codes sources

1	Installation des dépendances Docker	6
2	Création du répertoire de travail	7
3	Extrait du fichier docker-compose.yml	7
4	Lancement de la stack Zabbix	8
5	Script d'installation de l'agent Zabbix sur Ubuntu	9

1 Introduction

Ce projet s'inscrit dans le cadre du module de **Cloud Computing** et vise la mise en œuvre d'une infrastructure de supervision centralisée hébergée sur le cloud Amazon Web Services (AWS). L'objectif est de déployer une solution de monitoring pour un parc hybride (Linux et Windows).

1.1 Outils et Technologies

- **AWS** : Infrastructure (EC2, VPC, Security Groups).
- **Docker** : Conteneurisation du serveur Zabbix.
- **Zabbix** : Solution de monitoring open-source.

1.2 Ressources du Projet (GitHub)



2 Architecture Réseau

L'infrastructure réseau repose sur un VPC personnalisé avec un sous-réseau public.

2.1 Crédation du VPC

Le VPC **Fellah-Youssef-VPC-Projet-Zabbix** a été créé avec le bloc CIDR **10.0.0.0/16**.

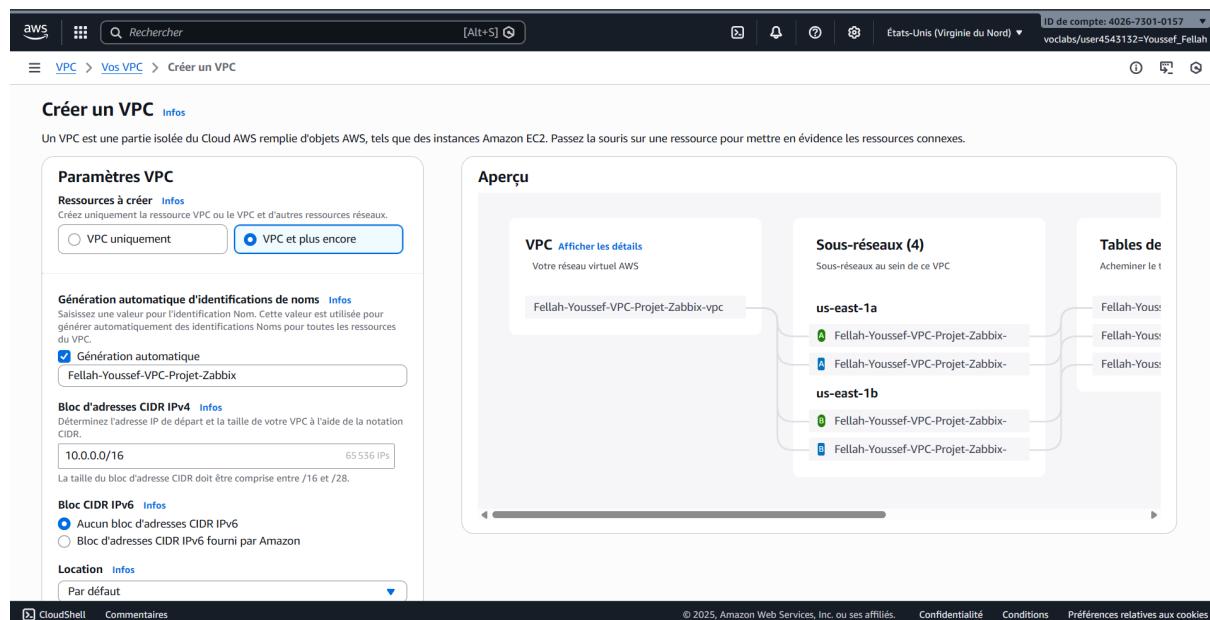


FIGURE 1 – Aperçu de la topologie du VPC

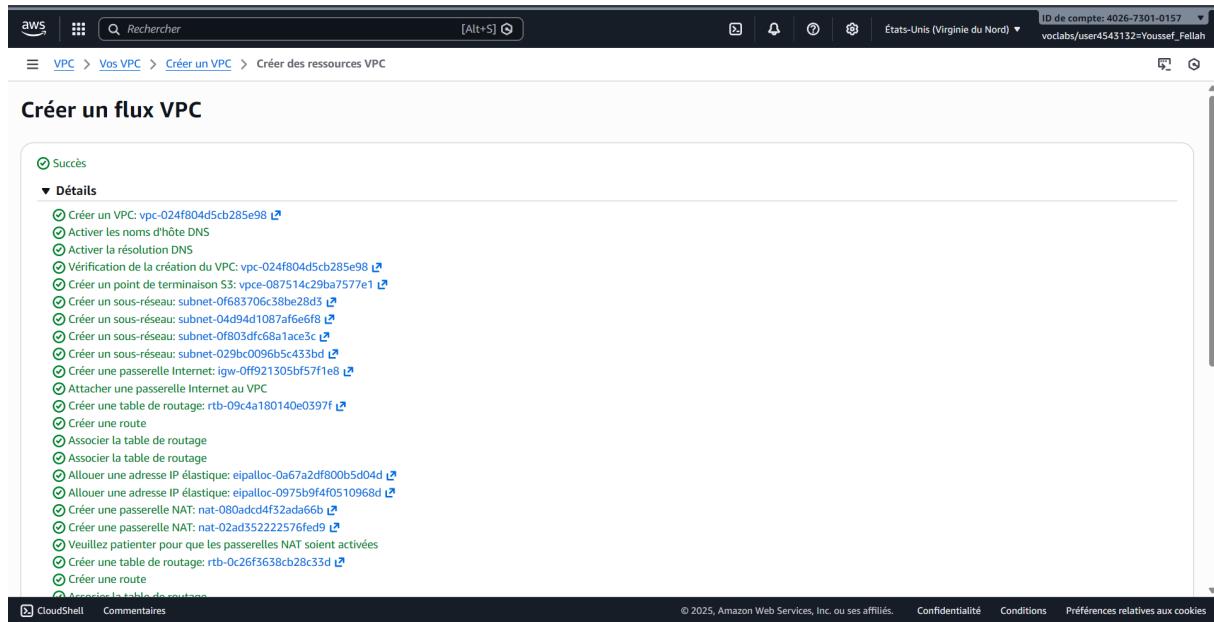


FIGURE 2 – Validation de la création des ressources VPC

2.2 Groupes de Sécurité

Le Security Group autorise les flux nécessaires au monitoring (Ports 10050, 10051) et à l'administration (22, 3389).

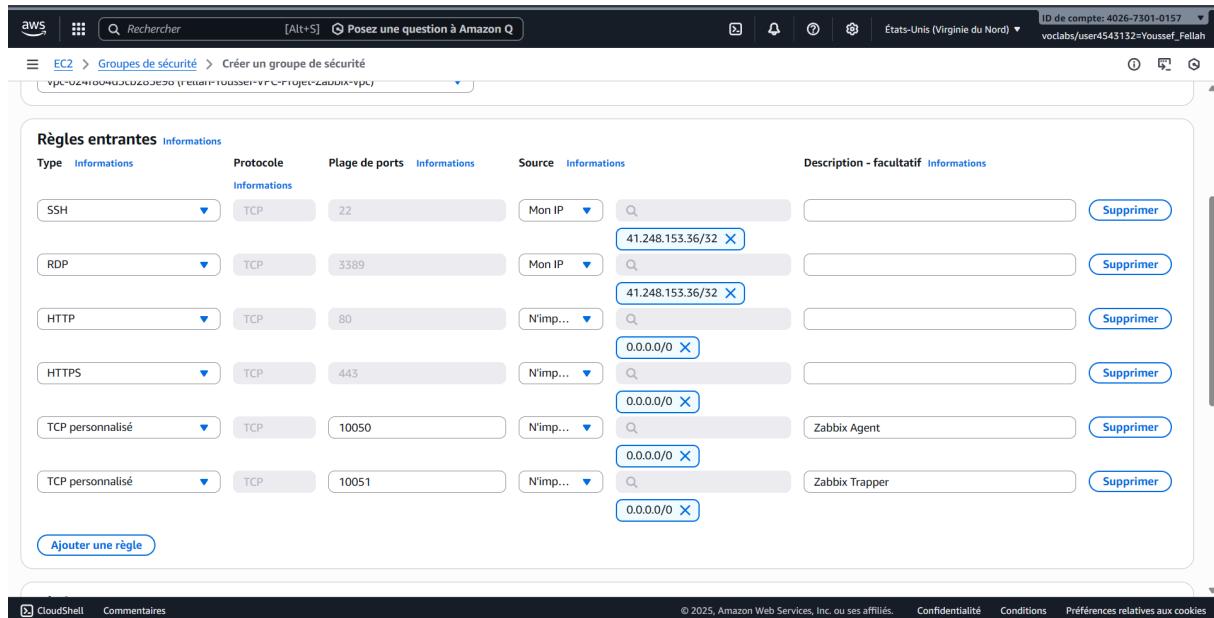


FIGURE 3 – Configuration des règles entrantes (Inbound Rules)

3 Architecture des Instances EC2

Trois instances ont été déployées pour simuler l'environnement de production.

- **Serveur Zabbix :** Ubuntu 22.04, t2.medium (4Go RAM).

- **Client Linux :** Ubuntu 22.04, t3.micro.
- **Client Windows :** Windows Server 2022, t3.medium.

The screenshot shows the AWS Management Console interface for the EC2 service. The left sidebar is collapsed, and the main area displays a table of instances. The table has columns for Name, ID d'instance, État de l'instance, Type d'insta..., Contrôle des statuts, and Statut d'alarme. Three instances are listed:

Name	ID d'instance	État de l'instance	Type d'insta...	Contrôle des statuts	Statut d'alarme
Fellah-Youssef-Zabbix-Server	i-0b8a9dd945cd98c3d	En cours d'exécution	t2.medium	2/2 vérifications réussies	Afficher les alarmes
Fellah-Youssef-Client-Linux	i-06d480eb98e475441	En cours d'exécution	t3.micro	3/3 vérifications réussies	Afficher les alarmes
Fellah-Youssef-Client-Windows	i-0aaeee12ac165b0ed	En cours d'exécution	t3.medium	Initialisation en cours	Afficher les alarmes

FIGURE 4 – Instances EC2 en cours d'exécution (Running)

4 Déploiement du Serveur Zabbix

Le déploiement du serveur de supervision a été réalisé de manière conteneurisée sur l'instance Ubuntu dédiée.

4.1 Installation de Docker

Nous avons d'abord préparé l'environnement en installant les paquets nécessaires et en configurant les droits utilisateurs.

```

1 # 1. Mise à jour du système
2 sudo apt update && sudo apt upgrade -y
3
4 # 2. Installation de Docker et Docker Compose
5 sudo apt install docker.io -y
6 sudo apt install docker-compose -y
7
8 # 3. Démarrage et activation du service Docker
9 sudo systemctl start docker
10 sudo systemctl enable docker
11
12 # 4. Ajout de l'utilisateur actuel au groupe docker
13 sudo usermod -aG docker $USER

```

Listing 1 – Installation des dépendances Docker

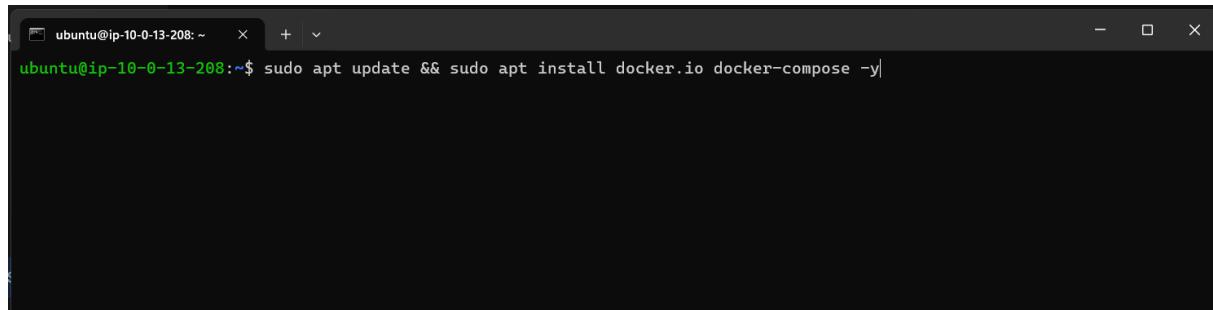


FIGURE 5 – Installation des paquets Docker sur le terminal

4.2 Configuration de l'Orchestrator

Nous avons créé l'arborescence du projet dans le dossier zabbix-docker.

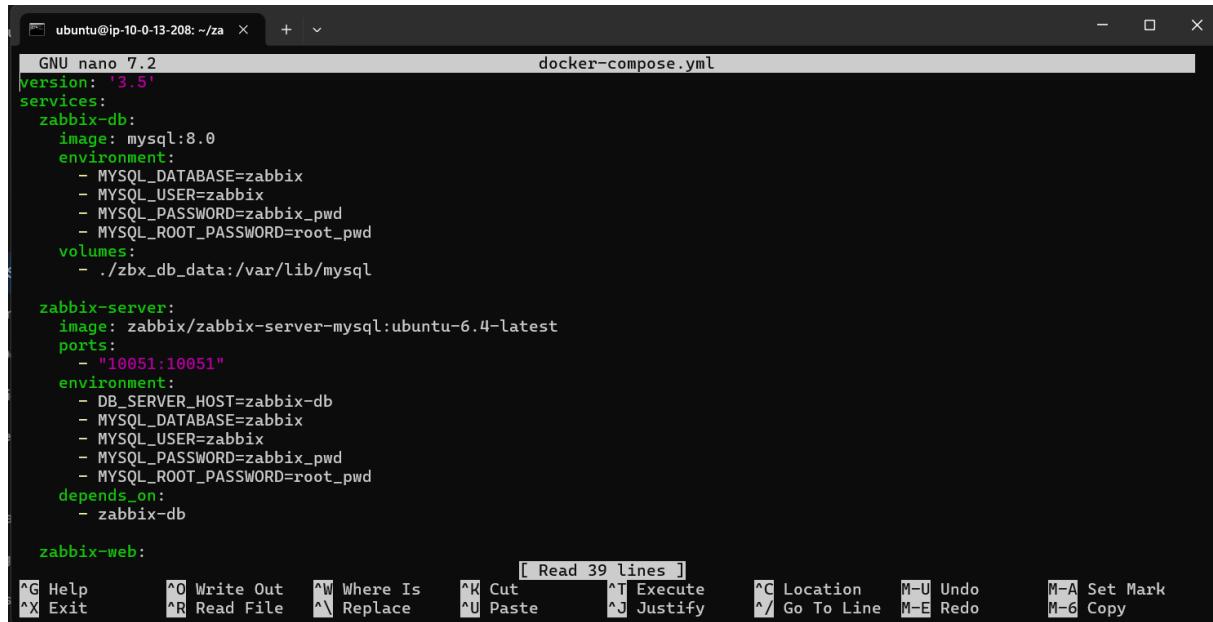
```
1 mkdir zabbix-docker && cd zabbix-docker
2 nano docker-compose.yml
```

Listing 2 – Création du répertoire de travail

Le fichier docker-compose.yml a été configuré pour utiliser des fichiers d'environnement externes (.env_db_mysql, etc.) afin de sécuriser les identifiants.

```
1 version: '3.5'
2 services:
3     zabbix-db:
4         image: mysql:8.0
5         command: --character-set-server=utf8 --collation-server=utf8_bin --
6         default-authentication-plugin=mysql_native_password
7         volumes:
8             - ./zbx_db_data:/var/lib/mysql
9         env_file:
10            - .env_db_mysql
11
12     zabbix-server:
13         image: zabbix/zabbix-server-mysql:ubuntu-6.4-latest
14         ports:
15             - "10051:10051"
16         env_file:
17             - .env_db_mysql
18             - .env_srv
19         depends_on:
20             - zabbix-db
21
22     zabbix-web:
23         image: zabbix/zabbix-web-apache-mysql:ubuntu-6.4-latest
24         ports:
25             - "80:8080"
26             - "443:8443"
27         env_file:
28             - .env_db_mysql
29             - .env_web
30         depends_on:
31             - zabbix-server
```

Listing 3 – Extrait du fichier docker-compose.yml



```

ubuntu@ip-10-0-13-208:~/za ~ + 
GNU nano 7.2                                            docker-compose.yml
version: '3.5'
services:
  zabbix-db:
    image: mysql:8.0
    environment:
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=zabbix_pwd
      - MYSQL_ROOT_PASSWORD=root_pwd
    volumes:
      - ./zbx_db_data:/var/lib/mysql

  zabbix-server:
    image: zabbix/zabbix-server-mysql:ubuntu-6.4-latest
    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:
    [ Read 39 lines ]
^G Help          ^O Write Out   ^W Where Is   ^K Cut           ^T Execute   ^C Location   M-U Undo   M-A Set Mark
^X Exit          ^R Read File   ^\ Replace    ^U Paste        ^J Justify   ^/ Go To Line M-E Redo   M-6 Copy

```

FIGURE 6 – Configuration du fichier docker-compose.yml

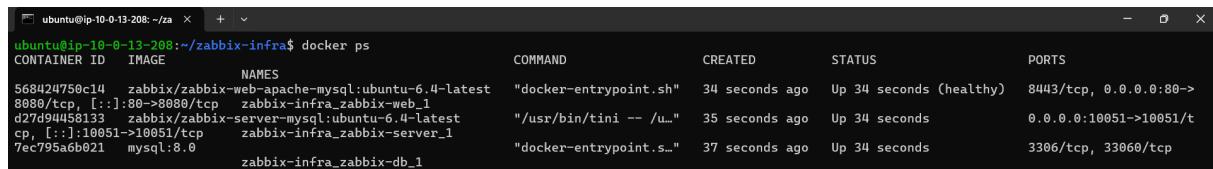
4.3 Lancement et Validation

Le service est démarré en mode détaché.

```
1 docker-compose up -d
```

Listing 4 – Lancement de la stack Zabbix

La vérification de l'état des conteneurs confirme que le déploiement est un succès :



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
568424750c14	zabbix/zabbix-web:apache-mysql:ubuntu-6.4-latest	"docker-entrypoint.sh"	34 seconds ago	Up 34 seconds (healthy)	8443/tcp, 0.0.0.0:80->8080/tcp, [::]:80->8080/tcp
d27d94458133	zabbix/zabbix-server-mysql:ubuntu-6.4-latest	"#/usr/bin/tini -- /u..."	35 seconds ago	Up 34 seconds	0.0.0.0:10051->10051/tcp
7ec795a6b021	mysql:8.0	"docker-entrypoint.s..."	37 seconds ago	Up 34 seconds	3306/tcp, 33060/tcp
	zabbix-infra_zabbix-db_1				

FIGURE 7 – Validation des conteneurs actifs via docker ps

4.4 Accès à l'Interface Web

L'interface est accessible via l'adresse IP publique de l'instance.

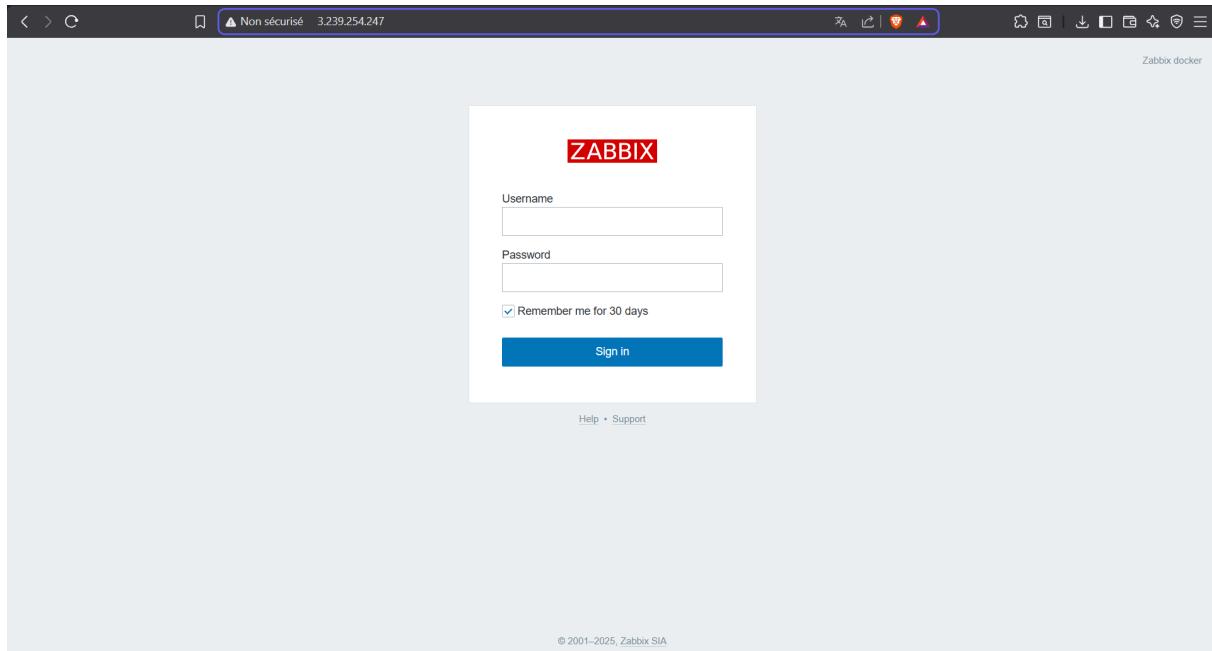


FIGURE 8 – Page de connexion Zabbix

5 Configuration des Clients (Agents)

Pour permettre la remontée des métriques vers le serveur central, l’agent Zabbix doit être installé et configuré sur chaque machine cible.

5.1 Client Linux (Ubuntu)

Sur l’instance Client-Linux, nous avons procédé à l’installation manuelle de l’agent en utilisant le dépôt officiel Zabbix pour garantir la compatibilité avec la version 6.4 du serveur.

5.1.1 Installation des paquets

Les commandes suivantes ont été exécutées pour ajouter le dépôt et installer l’agent :

```

1 # 1. Récupération et installation du dépôt officiel
2 wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/
      zabbix-release_latest+ubuntu24.04_all.deb
3 sudo dpkg -i zabbix-release_latest+ubuntu24.04_all.deb
4 sudo apt update
5
6 # 2. Installation du paquet de l’agent
7 sudo apt install zabbix-agent -y

```

Listing 5 – Script d’installation de l’agent Zabbix sur Ubuntu

```

2025-12-27 19:04:58 (791 MB/s) - 'zabbix-release_latest+ubuntu24.04_all.deb' saved [4612/4612]

ubuntu@ip-172-31-26-230:~$ sudo dpkg -i zabbix-release_latest+ubuntu24.04_all.deb
Selecting previously unselected package zabbix-release.
(Reading database ... 71735 files and directories currently installed.)
Preparing to unpack zabbix-release_latest+ubuntu24.04_all.deb ...
Unpacking zabbix-release (1:6.4-1+ubuntu24.04) ...
Setting up zabbix-release (1:6.4-1+ubuntu24.04) ...
ubuntu@ip-172-31-26-230:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://repo.zabbix.com/zabbix/6.4/ubuntu noble InRelease [3220 B]
Get:6 https://repo.zabbix.com/zabbix/6.4/ubuntu noble/main Sources [10.9 kB]
Get:7 https://repo.zabbix.com/zabbix/6.4/ubuntu noble/main Packages [5384 B]
Get:8 https://repo.zabbix.com/zabbix/6.4/ubuntu noble/main amd64 Packages [24.6 kB]
Fetched 44.1 kB in 1s (46.7 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
68 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-26-230:~$ 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:
  zabbix-agent

```

FIGURE 9 – Installation réussie du paquet zabbix-agent via apt

5.1.2 Configuration de l'Agent

L'étape critique consiste à modifier le fichier `/etc/zabbix/zabbix_agentd.conf` pour autoriser le serveur de supervision à communiquer avec cet agent.

Nous avons édité les paramètres suivants :

- **Server=10.0.13.208** : Adresse IP privée de notre serveur Zabbix (autorisé à faire du polling).
- **ServerActive=10.0.13.208** : Adresse IP pour les vérifications actives.
- **Hostname=Client-Linux** : Nom unique identifiant la machine.

```

/etc/zabbix/zabbix_agentd.conf
GNU nano 7.2

# If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.
# Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.
# If port is not specified, default port is used.
# IPv6 addresses must be enclosed in square brackets if port for that host is specified.
# If port is not specified, square brackets for IPv6 addresses are optional.
# If this parameter is not specified, active checks are disabled.
# Example for Zabbix proxy:
#   ServerActive=127.0.0.1:10051
# Example for multiple servers:
#   ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
# Example for high availability:
#   ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
# Example for high availability with two clusters and one server:
#   ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,zabbix.domain
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=10.0.13.208

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

Hostname=Client-Linux

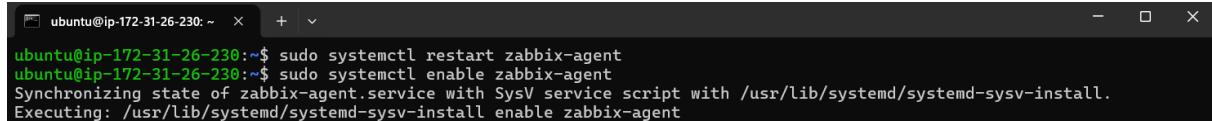
### Option: HostnameItem
#   Item used for generating Hostname if it is undefined. Ignored if Hostname is defined.
#   Does not support UserParameters or aliases.
#
# Mandatory: no
# Default:
# HostnameItem=system.hostname

```

FIGURE 10 – Configuration du Hostname et de l'IP du serveur dans zabbix_agentd.conf

Une fois la configuration sauvegardée, le service a été redémarré et activé au démarrage :

```
1 sudo systemctl restart zabbix-agent
2 sudo systemctl enable zabbix-agent
```



```
ubuntu@ip-172-31-26-230:~$ sudo systemctl restart zabbix-agent
ubuntu@ip-172-31-26-230:~$ 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
```

FIGURE 11 – Redémarrage du service pour appliquer la configuration

5.1.3 Déclaration de l'hôte dans Zabbix

Enfin, pour finaliser l'appairage, nous avons ajouté l'hôte dans l'interface Web de Zabbix (*Configuration > Hosts > Create host*) en veillant à ce que le "Host name" corresponde exactement à celui défini dans le fichier de configuration (Client-Linux).

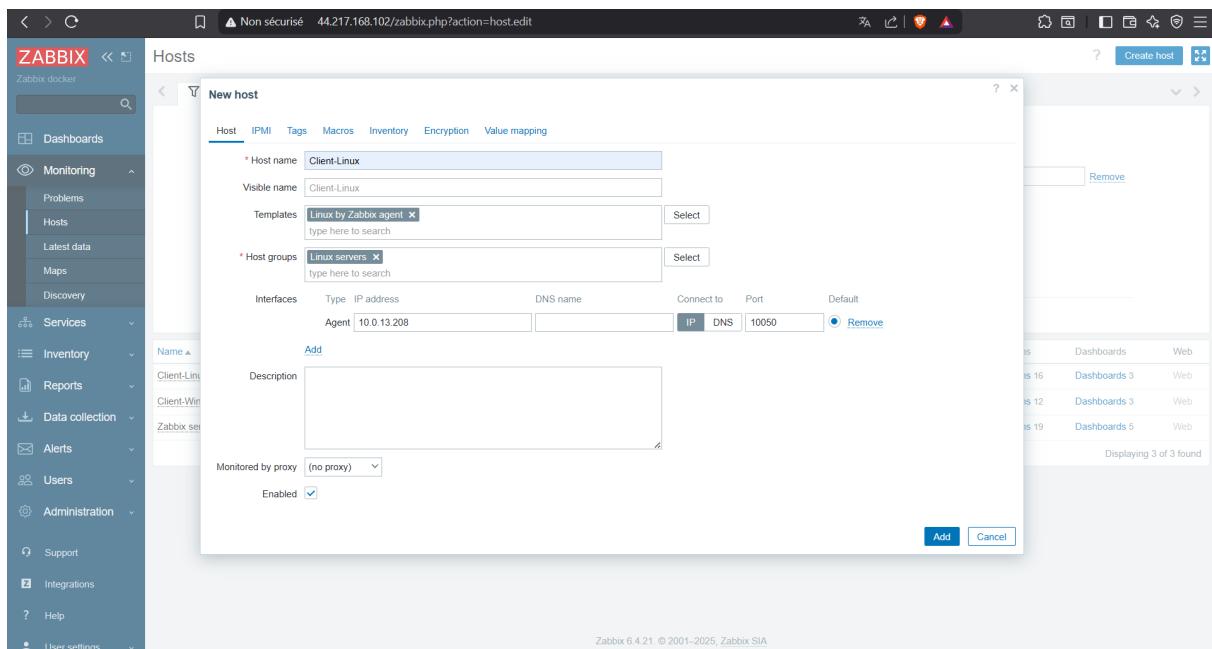


FIGURE 12 – Ajout de l'hôte Linux dans l'interface de gestion Zabbix

5.2 Client Windows (Windows Server 2022)

Pour le client Windows, la procédure diffère légèrement, s'appuyant sur une interface graphique via une connexion Bureau à Distance (RDP).

5.2.1 Accès à l'instance

Nous nous sommes connectés à l'instance Fellah-Youssef-Client-Windows en utilisant le client RDP et les identifiants administrateur déchiffrés via la clé privée AWS.

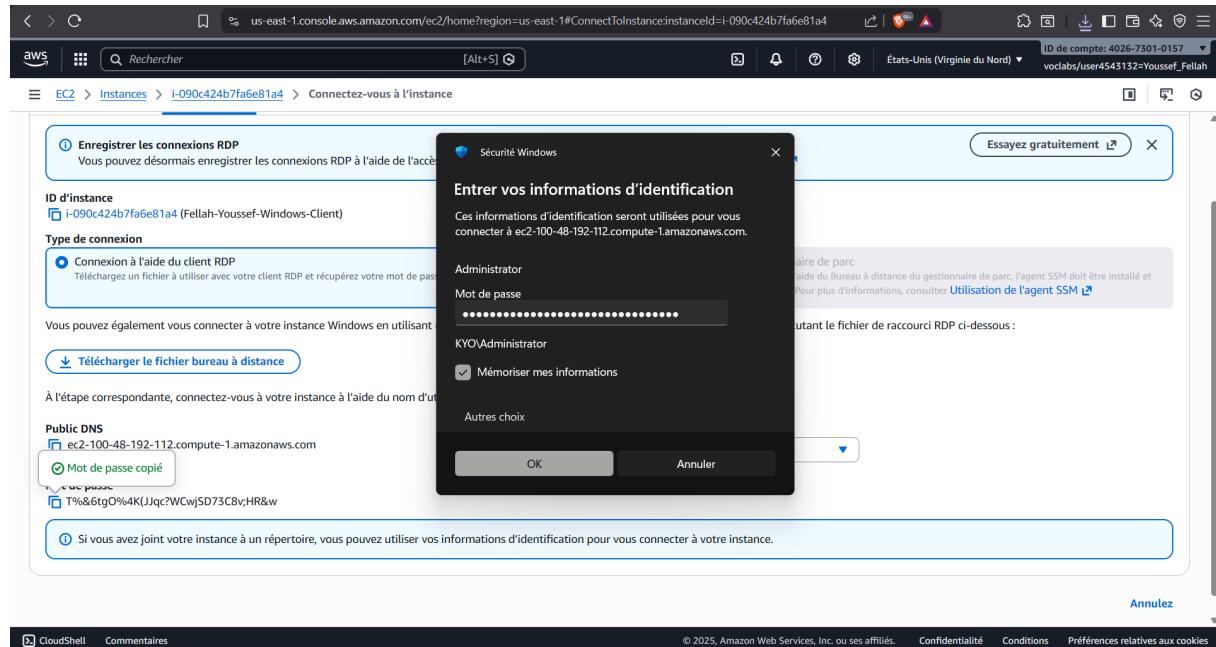


FIGURE 13 – Connexion RDP à l’instance Windows Server

5.2.2 Installation et Configuration de l’Agent

Depuis le navigateur de la VM, nous avons téléchargé l’installateur MSI officiel de l’agent Zabbix (version 7.4 LTS, compatible avec notre serveur).

Durant l’installation, nous avons configuré les paramètres de connexion pour lier l’agent à notre serveur de supervision :

- **Host name :** Client-Windows (Doit être identique au nom déclaré dans l’interface Web Zabbix).
- **Zabbix server IP :** 10.0.13.208 (L’IP privée de notre serveur Zabbix).
- **Agent listen port :** 10050 (Port par défaut).
- **Add agent location to the PATH :** Coché pour faciliter le débogage en ligne de commande.

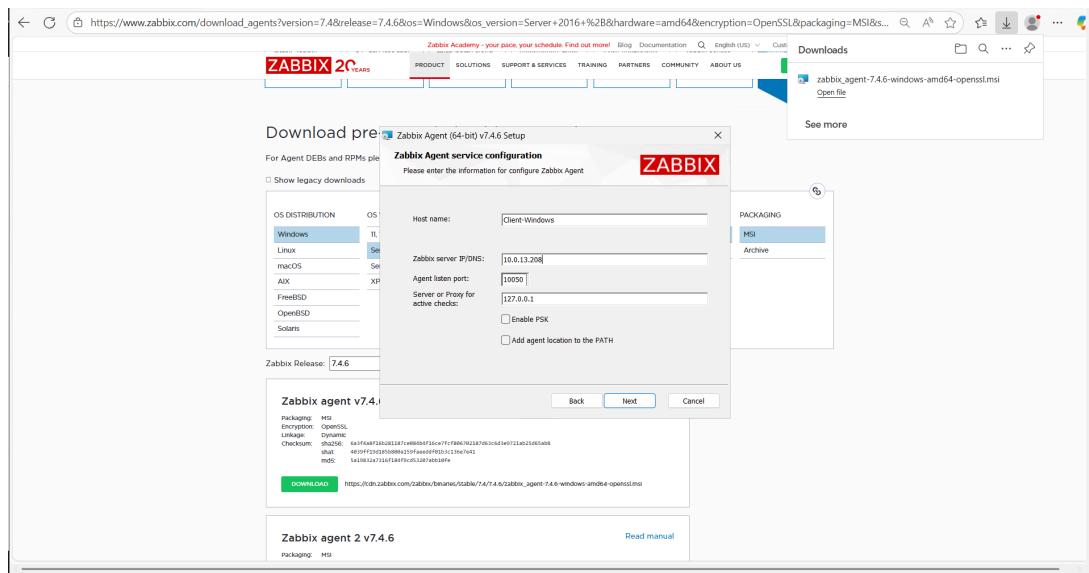


FIGURE 14 – Configuration de l’agent Zabbix via l’installateur MSI

L’installation MSI configure automatiquement le Pare-feu Windows (Windows Defender Firewall) pour autoriser le trafic entrant sur le port TCP 10050, rendant l’agent immédiatement opérationnel.

6 Monitoring et Tableaux de Bord

Une fois les agents installés et configurés, la dernière étape consistait à valider la remontée d’informations dans l’interface Zabbix et à visualiser les métriques.

6.1 Validation de la Connectivité (Disponibilité)

Nous avons ajouté nos deux clients ("Client-Linux" et "Client-Windows") dans la section *Configuration > Hosts*.

Le tableau ci-dessous présente l’état final du parc. La colonne **Availability** affiche l’icône **ZBX en vert** pour l’ensemble des hôtes, confirmant que le serveur reçoit correctement les données via le port 10050.

FIGURE 15 – Vue globale des hôtes : Le statut "Vert" (ZBX) valide la communication

6.2 Visualisation des Données (Graphiques)

Zabbix collecte désormais des métriques en temps réel (CPU, RAM, Disque, Réseau). Nous avons configuré un tableau de bord pour visualiser l’occupation des systèmes de fichiers du client Linux.

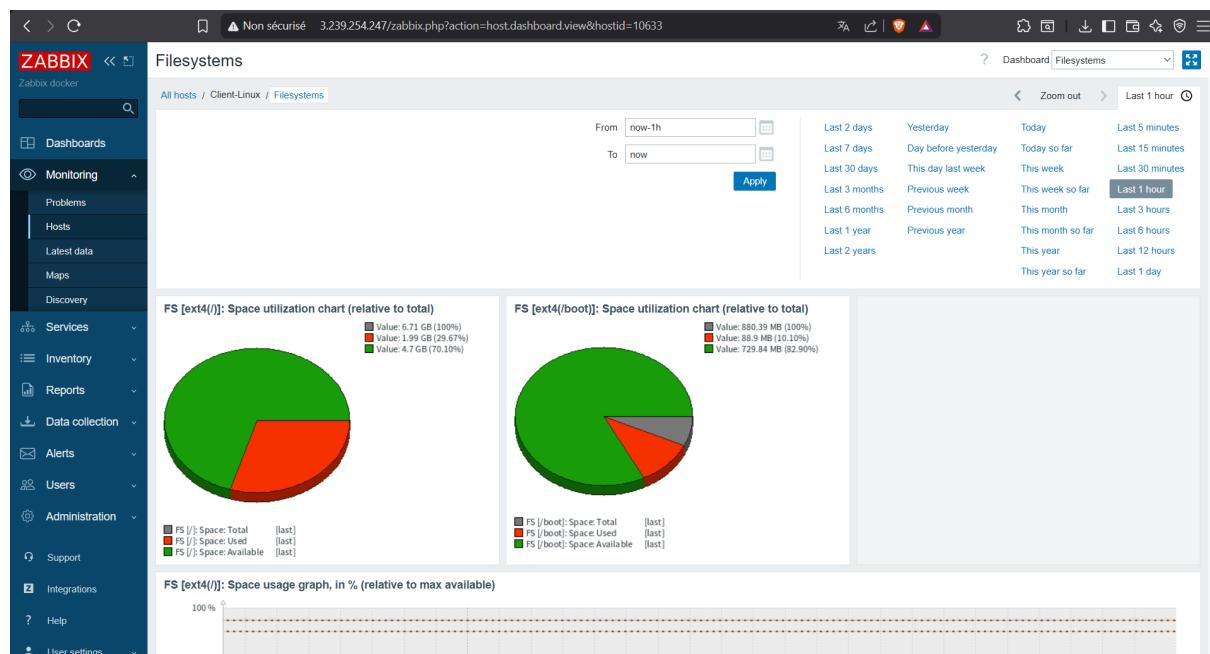


FIGURE 16 – Graphiques d'utilisation des disques sur le Client Linux

7 Conclusion

Ce projet nous a permis de déployer avec succès une infrastructure de supervision centralisée sur le cloud AWS, répondant aux exigences d'un environnement hybride moderne.

7.1 Bilan Technique

Nous avons réussi à :

- Mettre en œuvre une architecture réseau sécurisée (VPC, Security Groups) isolant les flux d'administration et de monitoring.
- Conteneuriser une application complexe (Zabbix) via Docker pour en faciliter le déploiement.
- Fédérer des systèmes hétérogènes (Ubuntu et Windows Server) au sein d'une même console de gestion.

7.2 Difficultés Rencontrées et Solutions

Durant la réalisation, nous avons fait face à quelques obstacles techniques :

1. **Configuration des Agents** : Une erreur de saisie lors de la déclaration de l'hôte Linux (doublon dans l'adresse IP, voir capture d'erreur lors des essais) a initialement empêché la connexion. Cela nous a appris l'importance de la validation rigoureuse des fichiers de configuration.
2. **Règles de Sécurité AWS** : La communication sur le port 10050 était initialement bloquée. Il a fallu modifier les *Security Groups* pour autoriser explicitement le trafic TCP sur ce port spécifique entre les instances du VPC.

En conclusion, cette architecture offre une base solide et évolutive pour la supervision d'un parc informatique d'entreprise.

8 Webographie

Pour la réalisation de ce projet, les documentations techniques suivantes ont été consultées :

- **Documentation AWS officielle** (VPC, Security Groups, EC2) :
<https://docs.aws.amazon.com/>
- **Documentation Zabbix 6.4** (Installation Agents, Docker) :
<https://www.zabbix.com/documentation/6.4/en/manual>
- **Docker Hub** (Images officielles Zabbix) :
<https://hub.docker.com/u/zabbix>