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

Réalisé par : **Aymen BENBOUHIA**

Encadré par : **Prof. Azeddine KHIAT**

Année universitaire : 2025/2026

Filière: 2ACI INFO GB

1. Introduction

Dans un contexte marqué par la complexité croissante et la distribution des infrastructures informatiques, la supervision joue un rôle stratégique pour assurer la **disponibilité**, la **performance** et la **sécurité** des systèmes.
Ce projet consiste à concevoir et déployer une **solution de supervision centralisée dans le cloud AWS**, en s'appuyant sur l'outil open source **Zabbix**.

L'objectif principal est de superviser des environnements **hétérogènes**, incluant des systèmes **Linux et Windows**, à partir d'un **serveur de supervision centralisé** hébergé sur AWS. La solution permet la **collecte continue des métriques**, la **visualisation des performances en temps réel** ainsi que la **génération automatique d'alertes** en cas d'incident ou de dépassement de seuils.

2. Objectifs du projet

Les objectifs de ce projet sont les suivants :

- Concevoir et mettre en œuvre une **infrastructure de supervision centralisée** dans un environnement cloud.
- Déployer un **serveur de supervision Zabbix** sur la plateforme **AWS**, en s'appuyant sur la technologie **Docker**.
- Assurer la supervision de **deux machines clientes distinctes**, l'une sous **Linux** et l'autre sous **Windows**.
- Collecter et analyser les **métriques système essentiels**, notamment l'utilisation du processeur, de la mémoire, les processus actifs et la disponibilité des services.
- Identifier et visualiser les **incidents et alertes** à travers les **tableaux de bord Zabbix**, afin de faciliter le suivi et la prise de décision.

3. Architecture de l'infrastructure

L'architecture du système repose sur plusieurs **instances EC2** déployées au sein d'un **même VPC AWS**, garantissant une communication sécurisée entre les composants grâce à l'utilisation d'**adresses IP privées**. Cette approche permet d'isoler l'infrastructure tout en assurant des échanges fiables et performants.

Les principaux composants de l'architecture sont les suivants :

- **Serveur Zabbix** : une instance **EC2 sous Ubuntu**, hébergeant la plateforme de supervision Zabbix déployée à l'aide de **Docker**, assurant la collecte, le traitement et la centralisation des données de supervision.

- **Client Linux** : une instance **EC2 Ubuntu**, supervisée par le **Zabbix Agent**, permettant la collecte des métriques système et l'envoi des données vers le serveur central.
- **Client Windows** : une instance **EC2 Windows Server**, supervisée par **Zabbix Agent** , dédiée à la surveillance des performances et de la disponibilité du système Windows.

4. Mise en place de l'infrastructure AWS

4.1 Création des instances EC2

Trois instances EC2 ont été créées :

- Une instance dédiée au serveur Zabbix.
- Une instance client Linux.
- Une instance client Windows.

Chaque instance est déployée dans le même VPC et le même sous-réseau afin de faciliter la communication interne.

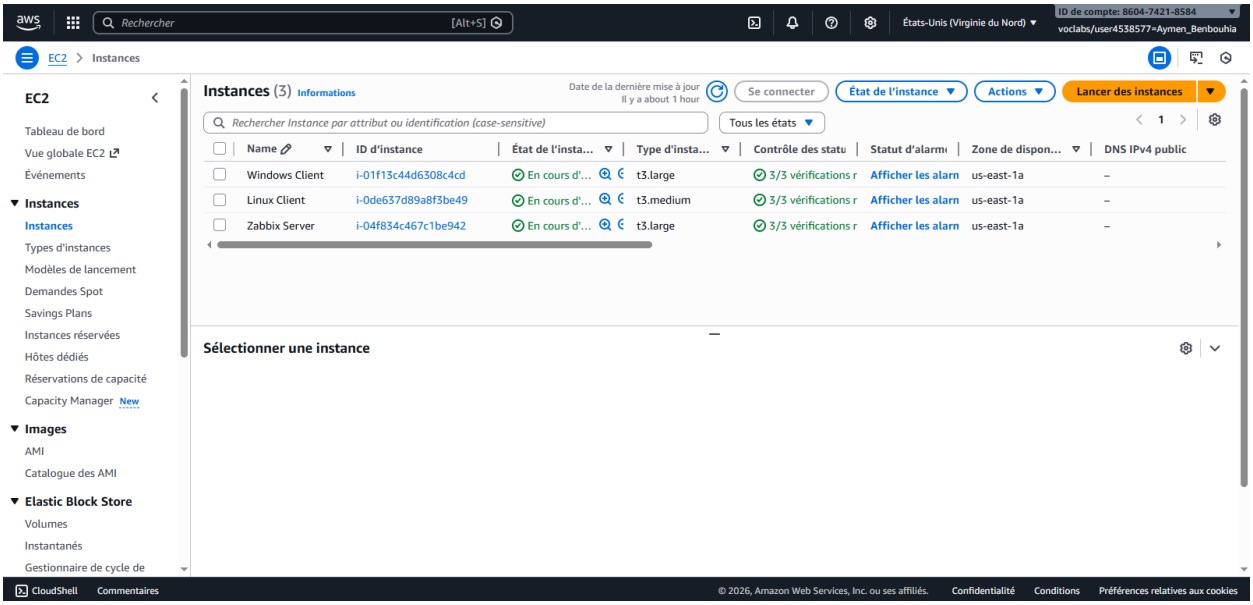


Figure 1: Liste des instances EC2 déployées sur AWS.

4.2 Configuration réseau et sécurité

Les règles de sécurité (Security Groups) ont été configurées comme suit :

- Port 22 (SSH) pour l'administration Linux.
- Port 3389 (RDP) pour l'accès à l'instance Windows.
- Port 80 pour l'accès à l'interface web Zabbix.
- Port 10050 pour la communication des agents Zabbix.
- Port 10051 pour le serveur Zabbix.

Les communications entre le serveur et les clients utilisent exclusivement les adresses IP privées AWS.

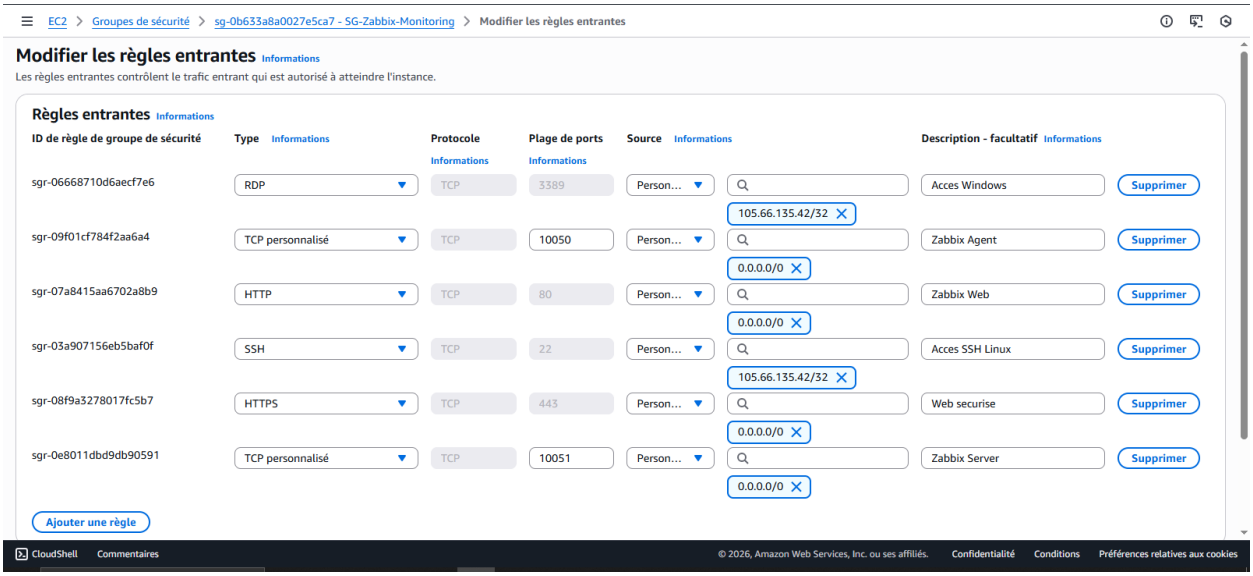


Figure 2 : Configuration des règles entrantes du groupe de sécurité AWS pour l'infrastructure Zabbix

5. Déploiement du serveur Zabbix

5.1 Installation de Docker

Le serveur Zabbix est déployé à l'aide de Docker afin de simplifier le déploiement et la gestion des services.

```
sudo apt update sudo apt install -y
docker.io docker-compose sudo systemctl
enable docker sudo systemctl start docker
```

5.2 Déploiement avec Docker Compose

Un fichier docker-compose.yml a permis de déployer :

- Zabbix Server
- Zabbix Web (Nginx)
- Base de données MySQL

Après le déploiement, les conteneurs sont opérationnels.

L'interface web Zabbix est accessible via un navigateur.

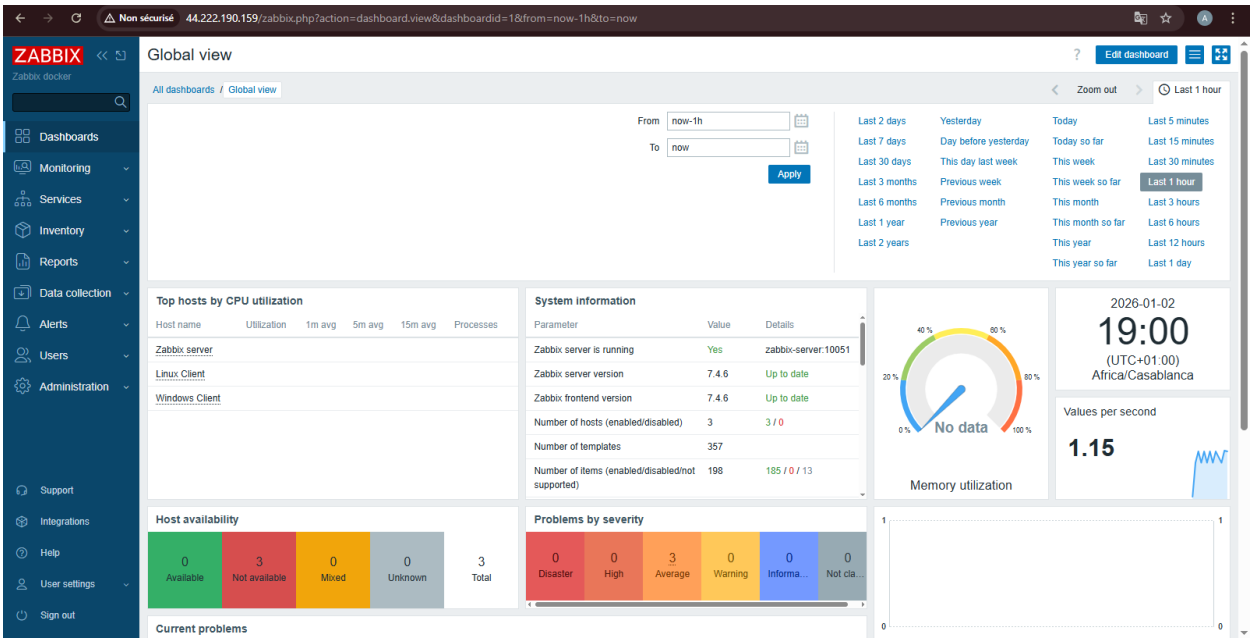


Figure 3 : Interface web Zabbix après déploiement.

6. Supervision du client Linux

6.1 Installation de l'agent Zabbix

Sur le client Linux, l'agent Zabbix a été installé afin de collecter les métriques système.

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

6.2 Configuration de l'agent

Le fichier de configuration /etc/zabbix/zabbix_agentd.conf a été modifié pour définir le serveur Zabbix et le nom de l'hôte.

```
Server=<IP_privée_Zabbix>
ServerActive=<IP_privée_Zabbix>
Hostname=Linux-Client
```

```
### Option: Server
#       List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbi>
#       Incoming connections will be accepted only from the hosts listed here.
#       If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treat>
#       and '::/0' will allow any IPv4 or IPv6 address.
#       '0.0.0.0/0' can be used to allow any IPv4 address.
#       Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
#
# Mandatory: yes, if StartAgents is not explicitly set to 0
# Default:
# Server=

Server=10.0.1.149
```

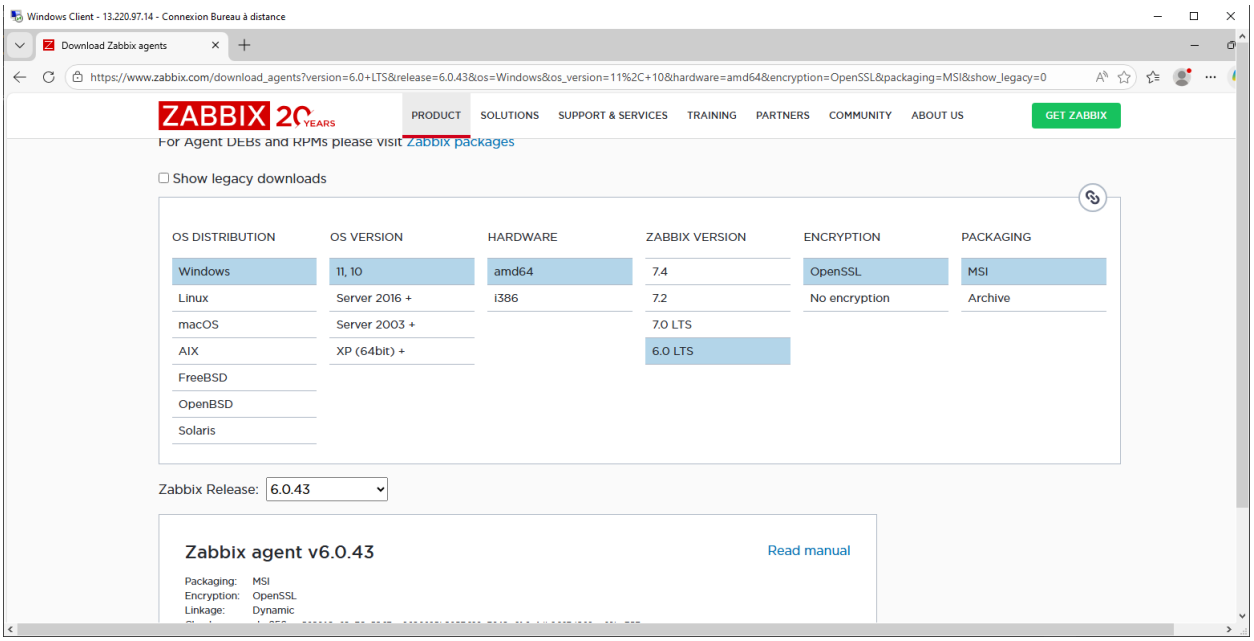
```
### Option: ServerActive
#       List of comma delimited IP:port (or DNS name:port) pairs of Zabbix servers and Zabbix pr>
#       If port is not specified, default port is used.
#       IPv6 addresses must be enclosed in square brackets if port for that host is specified.
#       If port is not specified, square brackets for IPv6 addresses are optional.
#       If this parameter is not specified, active checks are disabled.
#       Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=10.0.1.149
```

```
### Option: Hostname
#       Unique, case sensitive hostname.
#       Required for active checks and must match hostname as configured on the server.
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default
Hostname=Client Linux
```

Figure 4 : Configuration de l'agent Zabbix sur le client Linux.

7. Supervision du client Windows

Sur le client Windows, Zabbix Agent a été installé via le web Download Zabbix agents.




```
### Option: Server
#       List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
#       Incoming connections will be accepted only from the hosts listed here.
#       If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6
#       '0.0.0.0/0' can be used to allow any IPv4 address.
#       Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.domain
#
# Mandatory: yes, if StartAgents is not explicitly set to 0
# Default:
# Server=

Server=10.0.1.149
```

```
ServerActive=10.0.1.149

### 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=Windows Client

# Hostname=EC2AMAZ-E9V7SQT
```

8. Tableaux de bord et alertes

Zabbix permet de visualiser l'état global de l'infrastructure à travers des tableaux de bord personnalisés.

Le tableau de bord global affiche :

- La disponibilité des hôtes.
- L'utilisation CPU des clients Linux et Windows.
- Les problèmes détectés par sévérité.

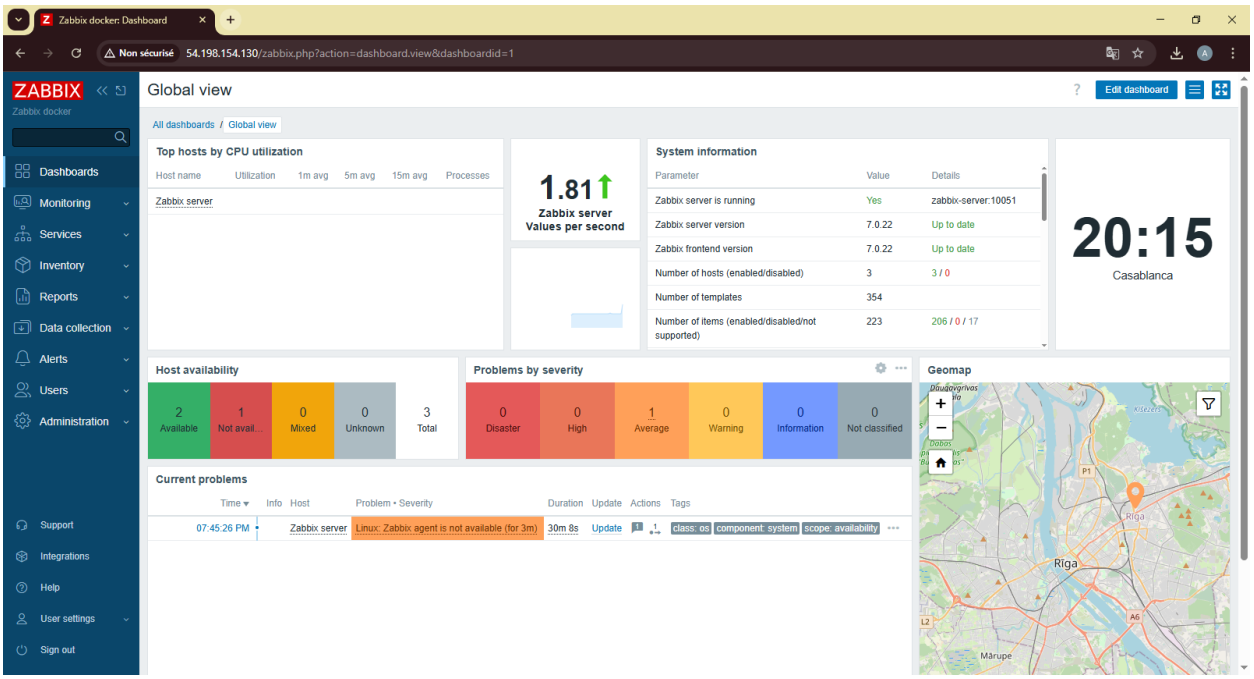


Figure 5 : Tableau de bord global de supervision.

9. Résultats obtenus

Les résultats obtenus montrent que :

- Les deux clients sont correctement supervisés.
- Les métriques système sont collectées en temps réel.
- Les alertes sont générées automatiquement en cas de problème.
- Le tableau de bord offre une vue claire et centralisée de l'infrastructure.

Top hosts by CPU utilization

Host name	Utilization		1m avg	5m avg	15m avg	Processes
Windows-Client		0.65 %				100
Linux-Client		0.10 %	0.00	0.00	0.00	106
Zabbix server						

Figure 6 : Données de supervision CPU et mémoire.

10. Difficultés rencontrées

Plusieurs difficultés ont été rencontrées lors de la mise en œuvre de l'infrastructure de supervision. Parmi celles-ci figurent la **configuration des règles de sécurité AWS**, notamment l'ouverture et la restriction des ports nécessaires, ainsi que des **problèmes de communication entre le serveur Zabbix et les agents** installés sur les machines clientes.
L'adaptation de Zabbix à un **environnement conteneurisé avec Docker** a également constitué un défi technique.

En particulier, un problème de disponibilité du **serveur Zabbix** a été identifié, lié à l'absence d'un **agent Zabbix dans l'environnement Docker** et à une **configuration réseau incomplète**. L'ajout d'un conteneur **Zabbix Agent 2**, accompagné de la définition d'un **réseau Docker dédié**, a permis de rétablir la communication entre le serveur et l'agent et d'assurer la supervision correcte du serveur lui-même.

L'ensemble de ces difficultés a été résolu grâce à une **configuration rigoureuse des ports**, à l'utilisation des **adresses IP privées au sein du VPC**, ainsi qu'à l'application appropriée des **templates Zabbix**, garantissant ainsi le bon fonctionnement global de la solution de supervision.

11. Conclusion

Ce projet a permis de mettre en place une infrastructure de supervision centralisée performante en s'appuyant sur AWS, Docker et Zabbix. La solution déployée permet de superviser efficacement des systèmes Linux et Windows, de visualiser les métriques clés et de détecter rapidement les incidents.

Cette infrastructure constitue une base solide pouvant être étendue à des environnements plus complexes et professionnels.

12. GitHub Link

Ce dépôt contient l'ensemble des fichiers et configurations utilisés pour la mise en place de l'infrastructure cloud, ainsi que les scripts, la documentation et les instructions nécessaires pour reproduire l'environnement et tester les fonctionnalités implémentées.

https://github.com/AymnBen/infraCloud_projet.git