



Rapport de Travail Pratique

Infrastructure de Monitoring Centralisé sur AWS

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

Réalisé par :
El Abssi Sami

Encadré par :
Prof. Azeddine KHIAT

Filière : Ingénieur informatique et intelligence artificielle

Année universitaire : 2025/2026

Dépôt GitHub :
<https://github.com/samielabssi/Infrastructure-de-Monitoring-Centralis-sur-AWS>

Table des matières

1 Introduction	2
1.1 Présentation du projet	2
1.2 Outils utilisés	2
1.3 Architecture globale	2
2 Architecture Réseau	3
2.1 Création du VPC	3
2.2 Configuration du sous-réseau et de la passerelle Internet	3
2.3 Table de routage	4
2.4 Groupes de sécurité	4
2.4.1 Groupe de sécurité du serveur Zabbix	4
2.4.2 Groupe de sécurité des clients	4
3 Déploiement des Instances EC2	6
3.1 Types d'instances choisies	6
3.2 Instances en cours d'exécution	6
3.3 Connexion au serveur Zabbix	6
4 Déploiement du Serveur Zabbix en Mode Conteneurisé	8
4.1 Installation de Docker et Docker Compose	8
4.2 Configuration du fichier docker-compose.yml	8
4.3 Lancement des conteneurs	9
4.4 Vérification de l'état des conteneurs	10
4.5 Accès à l'interface web Zabbix	10
5 Configuration des Clients (Agents Zabbix)	12
5.1 Configuration du client Linux	12
5.1.1 Installation de l'agent	12
5.1.2 Configuration de l'agent	12
5.1.3 Démarrage du service	12
5.2 Configuration du client Windows	12
5.2.1 Installation de l'agent	12
5.2.2 Vérification du service	13
6 Monitoring et Tableaux de Bord	14
6.1 Ajout des hôtes dans Zabbix	14
6.2 Statut des hôtes	14
6.3 Visualisation des données	14
6.3.1 Client Linux	14
6.3.2 Client Windows	15
6.4 Données récentes	15
7 Conclusion	16
7.1 Bilan du travail réalisé	16
7.2 Difficultés rencontrées et solutions	16
7.3 Améliorations possibles	16

1 Introduction

1.1 Présentation du projet

Ce travail pratique a pour objectif de déployer une infrastructure de monitoring centralisée sur Amazon Web Services (AWS) en utilisant Zabbix conteneurisé avec Docker. L'infrastructure vise à surveiller un parc hybride composé de machines Linux et Windows, démontrant ainsi la flexibilité et la puissance de Zabbix dans un environnement cloud.

1.2 Outils utilisés

- **AWS (Amazon Web Services)** : Plateforme cloud utilisée pour héberger toute l'infrastructure (instances EC2, VPC, groupes de sécurité).
- **Docker** : Solution de conteneurisation permettant de déployer facilement Zabbix et ses composants (serveur, base de données, interface web).
- **Zabbix** : Solution de monitoring open-source offrant une surveillance complète des ressources système et réseau.

1.3 Architecture globale

L'infrastructure déployée comprend :

- Un VPC avec un sous-réseau public
- Trois instances EC2 : serveur Zabbix, client Linux, client Windows
- Des groupes de sécurité pour contrôler les flux réseau
- Zabbix déployé via Docker Compose
- Des agents Zabbix installés sur les clients

2 Architecture Réseau

2.1 Crédation du VPC

La première étape consiste à créer un VPC (Virtual Private Cloud) isolé pour héberger notre infrastructure. Nous avons opté pour une configuration simple avec un sous-réseau public pour faciliter l'accès sans VPN.

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

VPC only VPC and more

Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.
TP-Zabbix-VPC-Sami

IPv4 CIDR block [Info](#)
 IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block
10.0.0.0/16
CIDR block size must be between /16 and /28.

IPv6 CIDR block [Info](#)
 No IPv6 CIDR block IPAM-allocated IPv6 CIDR block Amazon-provided IPv6 CIDR block IPv6 CIDR owned by me

Tenancy [Info](#)
Default

VPC encryption control (\$) [Info](#)
Monitor mode provides visibility into encryption status without blocking traffic. Enforce mode prevents unencrypted traffic. Additional charges apply ↗
 None Monitor mode See which resources in your VPC are unencrypted but allow the creation of unencrypted resources. Enforce mode Requires all resources, except exclusions, in your VPC to be encryption-capable and blocks creation of unencrypted resources.

FIGURE 1 – Crédation du VPC 'TP-Zabbix-VPC' avec le bloc CIDR 10.0.0.0/16

2.2 Configuration du sous-réseau et de la passerelle Internet

Un sous-réseau public a été créé dans la zone de disponibilité us-east-1a. Une passerelle Internet a été attachée au VPC pour permettre la communication avec l'extérieur.

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
Public-Subnet-1a
The name can be up to 256 characters long.

Availability Zone [Info](#)
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
United States (N. Virginia) / use1-az6 (us-east-1a)

IPv4 VPC CIDR block [Info](#)
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.
10.0.0.0/16

IPv4 subnet CIDR block
10.0.1.0/24 256 IPs
< > ^ ^

FIGURE 2 – Configuration du sous-réseau public avec le CIDR 10.0.1.0/24

2.3 Table de routage

La table de routage a été configurée pour diriger le trafic Internet (0.0.0.0/0) vers la passerelle Internet.

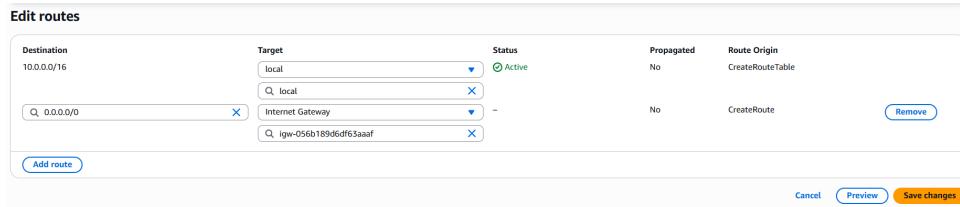


FIGURE 3 – Table de routage avec la route par défaut vers l’Internet Gateway

2.4 Groupes de sécurité

Deux groupes de sécurité ont été créés pour sécuriser les communications :

2.4.1 Groupe de sécurité du serveur Zabbix

Ce groupe contrôle l'accès au serveur Zabbix :

- Port 22 (SSH) : Accès administrateur
- Port 80/443 (HTTP/HTTPS) : Interface web Zabbix
- Ports 10050/10051 : Communications avec les agents Zabbix

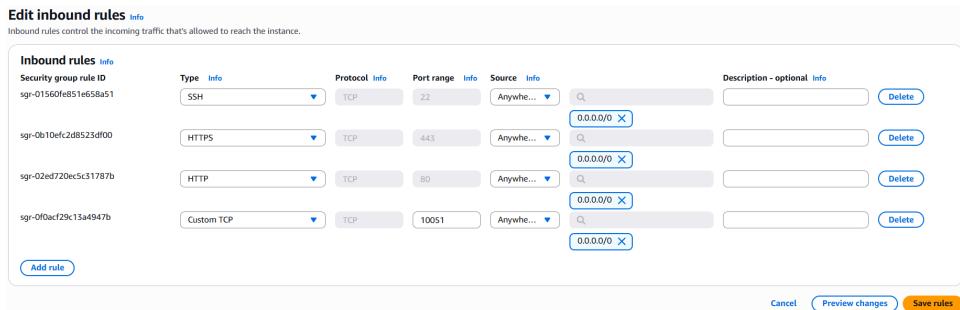


FIGURE 4 – Règles d’entrée du groupe de sécurité 'SG_Serveur_Zabbix'

2.4.2 Groupe de sécurité des clients

Ce groupe sécurise les instances clientes :

- Port 22 (SSH) : Accès au client Linux
- Port 3389 (RDP) : Accès au client Windows
- Ports 10050/10051 : Restreints à l'IP privée du serveur Zabbix

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0c56b7ca989d27dc4	RDP	TCP	3389	Anywhere... <input type="text" value="0.0.0.0/0"/> <input type="button" value="Delete"/>	<input type="text"/> <input type="button" value="0.0.0.0/0 X"/>
sgr-0f23e13410450f62f	Custom TCP	TCP	10051	Anywhere... <input type="text" value="0.0.0.0/0"/> <input type="button" value="Delete"/>	<input type="text"/> <input type="button" value="0.0.0.0/0 X"/>
sgr-0eed847bb74789ed3	SSH	TCP	22	Anywhere... <input type="text" value="0.0.0.0/0"/> <input type="button" value="Delete"/>	<input type="text"/> <input type="button" value="0.0.0.0/0 X"/>

[Add rule](#) [Cancel](#) [Preview changes](#) [Save rules](#)

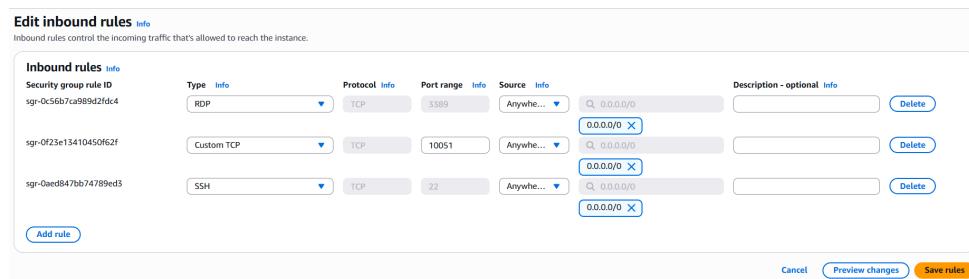


FIGURE 5 – Règles d'entrée du groupe de sécurité 'SG_Clients'

3 Déploiement des Instances EC2

3.1 Types d'instances choisies

Conformément aux limitations du Learner Lab, nous avons déployé trois instances :

Instance	Type	OS	Rôle
Serveur Zabbix	t3.large	Ubuntu 22.04	Serveur de monitoring
Client Linux	t3.medium	Ubuntu 22.04	Hôte monitoré Linux
Client Windows	t3.large	Windows Server 2022	Hôte monitoré Windows

TABLE 1 – Configuration des instances EC2 déployées

3.2 Instances en cours d'exécution

Toutes les instances ont été lancées dans la région us-east-1 (N. Virginia) pour assurer la stabilité du Lab.

<input type="checkbox"/>	Serveur Zabbix	i-063ae802cccd6eb1ad	Running ⓘ ⓘ	t3.large	Initializing	View alarms +	us-east-1a
<input type="checkbox"/>	Client Windows	i-0c73ac9b06d035414	Running ⓘ ⓘ	t3.large	Initializing	View alarms +	us-east-1a
<input type="checkbox"/>	Client Linux	i-09dff61e2e2a3a4e0d	Running ⓘ ⓘ	t3.medium	Initializing	View alarms +	us-east-1a

FIGURE 6 – Récapitulatif des trois instances EC2 en cours d'exécution avec leurs IPs

3.3 Connexion au serveur Zabbix

La connexion SSH à l'instance serveur a été établie avec succès.

```
C:\Users\HP VICTUS\Downloads>ssh -i "Zabbix.pem" ubuntu@13.220.220.44
The authenticity of host '13.220.220.44 (13.220.220.44)' can't be established.
ED25519 key fingerprint is SHA256:YaL5YVl5KdCbj9BDzI4ys45Kob9Rdze0IEpFW35K7R4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.220.220.44' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Tue Feb 17 20:09:36 UTC 2026

  System load:  0.0          Temperature:      -273.1 C
  Usage of /:   25.9% of 6.71GB  Processes:        116
  Memory usage: 3%           Users logged in:  0
  Swap usage:   0%           IPv4 address for ens5: 10.0.1.83

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-83:~$ |
```

FIGURE 7 – Connexion SSH réussie à l’instance ‘Serveur Zabbix’

4 Déploiement du Serveur Zabbix en Mode Conteneurisé

4.1 Installation de Docker et Docker Compose

Sur l'instance serveur, nous avons installé Docker et Docker Compose.

```
1 # Installation de Docker
2 sudo apt update && sudo apt upgrade -y
3 sudo apt install docker.io -y
4 sudo systemctl start docker
5 sudo systemctl enable docker
6
7 # Installation de Docker Compose
8 sudo apt install docker-compose -y
```

Listing 1 – Installation de Docker et Docker Compose

4.2 Configuration du fichier docker-compose.yml

Le fichier `docker-compose.yml` définit trois services : MySQL (base de données), Zabbix Server et l'interface web Zabbix.

```
1 version: '3.5'
2 services:
3     mysql-server:
4         image: mysql:8.0
5         container_name: zabbix-mysql
6         command: --character-set-server=utf8mb4 --collation-
7             server=utf8mb4_unicode_ci
8         volumes:
9             - ./zabbix-mysql-data:/var/lib/mysql
10        environment:
11            MYSQL_DATABASE: 'zabbix'
12            MYSQL_USER: 'zabbix'
13            MYSQL_PASSWORD: 'zabbix_pwd'
14            MYSQL_ROOT_PASSWORD: 'root_pwd'
15        restart: always
16        networks:
17            - zbx_network
18
19     zabbix-server:
20         image: zabbix/zabbix-server-mysql:alpine-6.4-latest
21         container_name: zabbix-server
22         environment:
23             DB_SERVER_HOST: 'mysql-server'
24             MYSQL_DATABASE: 'zabbix'
25             MYSQL_USER: 'zabbix'
26             MYSQL_PASSWORD: 'zabbix_pwd'
27             MYSQL_ROOT_PASSWORD: 'root_pwd'
28         ports:
```

```
28      - "10051:10051"
29  depends_on:
30    - mysql-server
31  restart: always
32  networks:
33    - zbx_network
34
35 zabbix-web:
36   image: zabbix/zabbix-web-nginx-mysql:alpine-6.4-latest
37   container_name: zabbix-web
38   environment:
39     ZBX_SERVER_HOST: 'zabbix-server'
40     DB_SERVER_HOST: 'mysql-server'
41     MYSQL_DATABASE: 'zabbix'
42     MYSQL_USER: 'zabbix'
43     MYSQL_PASSWORD: 'zabbix_pwd'
44     MYSQL_ROOT_PASSWORD: 'root_pwd'
45     PHP_TZ: 'Europe/Paris'
46   ports:
47     - "80:8080"
48   depends_on:
49     - mysql-server
50     - zabbix-server
51   restart: always
52   networks:
53     - zbx_network
54
55 networks:
56   zbx_network:
57     driver: bridge
```

Listing 2 – Fichier docker-compose.yml

4.3 Lancement des conteneurs

Les conteneurs ont été lancés avec Docker Compose.

```
ubuntu@ip-10-0-1-83:~/zabbix-docker$ sudo docker-compose up -d
Creating network "zabbix-docker_zbx_network" with driver "bridge"
Pulling mysql-server (mysql:8.0)...
8.0: Pulling from library/mysql
4f37333d1be6: Pull complete
bde62e757594: Pull complete
f508d7fab5b3: Pull complete
d442b2c1726e: Pull complete
a9a9deeee02a: Pull complete
23fbf4028535: Pull complete
2e2c1f6f8d57: Pull complete
ce98f3559366: Pull complete
bae900376130: Pull complete
e7a04c019bde: Pull complete
e05db5310ebc: Pull complete
Digest: sha256:99d774bf02a48a1bb1c599920d2571946d31e5940b854b02737d5e95c184358f
Status: Downloaded newer image for mysql:8.0
Pulling zabbix-server (zabbix/zabbix-server-mysql:alpine-6.4-latest)...
alpine-6.4-latest: Pulling from zabbix/zabbix-server-mysql
1f3e46996e29: Pull complete
1772a90c6a5c: Pull complete
f803498c2d79: Pull complete
152fedaa2c3af: Pull complete
fe17bf02638c: Pull complete
e266bac3ed30: Pull complete
4f4fb700ef54: Pull complete
2d84b546e482: Pull complete
Digest: sha256:cda6ddbda3fc0c5d18294fa2a5d6df13e2decae692e33458847e5bf9dadfd9cb
Status: Downloaded newer image for zabbix/zabbix-server-mysql:alpine-6.4-latest
Pulling zabbix-web (zabbix/zabbix-web-nginx-mysql:alpine-6.4-latest)...
alpine-6.4-latest: Pulling from zabbix/zabbix-web-nginx-mysql
1f3e46996e29: Already exists
f22e0f609cf2: Pull complete
447f8b7a49df: Pull complete
17c1c2cf73ef: Pull complete
4f4fb700ef54: Pull complete
47f2a23549dd: Pull complete
Digest: sha256:002c4d7da19d3de020c278b1b442eb8e413f319c1f00e2a7c9b9e8e9b6e30013
Status: Downloaded newer image for zabbix/zabbix-web-nginx-mysql:alpine-6.4-latest
Creating zabbix-mysql ... done
Creating zabbix-server ... done
Creating zabbix-web ... done
ubuntu@ip-10-0-1-83:~/zabbix-docker$ |
```

FIGURE 8 – Exécution de la commande `sudo docker-compose up -d` sur le serveur

4.4 Vérification de l'état des conteneurs

Les trois conteneurs sont bien en état 'Up', indiquant un fonctionnement correct.

Name	Command	State	Ports
zabbix-mysql	docker-entrypoint.sh --cha ...	Up	3306/tcp, 33060/tcp
zabbix-server	/sbin/tini -- /usr/bin/doc ...	Up	0.0.0.0:10051->10051/tcp,:::10051->10051/tcp
zabbix-web	docker-entrypoint.sh	Up (healthy)	0.0.0.0:80->8080/tcp,:::80->8080/tcp, 8443/tcp

FIGURE 9 – Résultat de la commande `sudo docker-compose ps` montrant les conteneurs en état 'Up'

4.5 Accès à l'interface web Zabbix

L'interface web est accessible via l'adresse IP publique du serveur.

TP Monitoring DÉMOSOITEMENT DU SERVEUR ZABBIX EN MODE CONTENEURISÉ

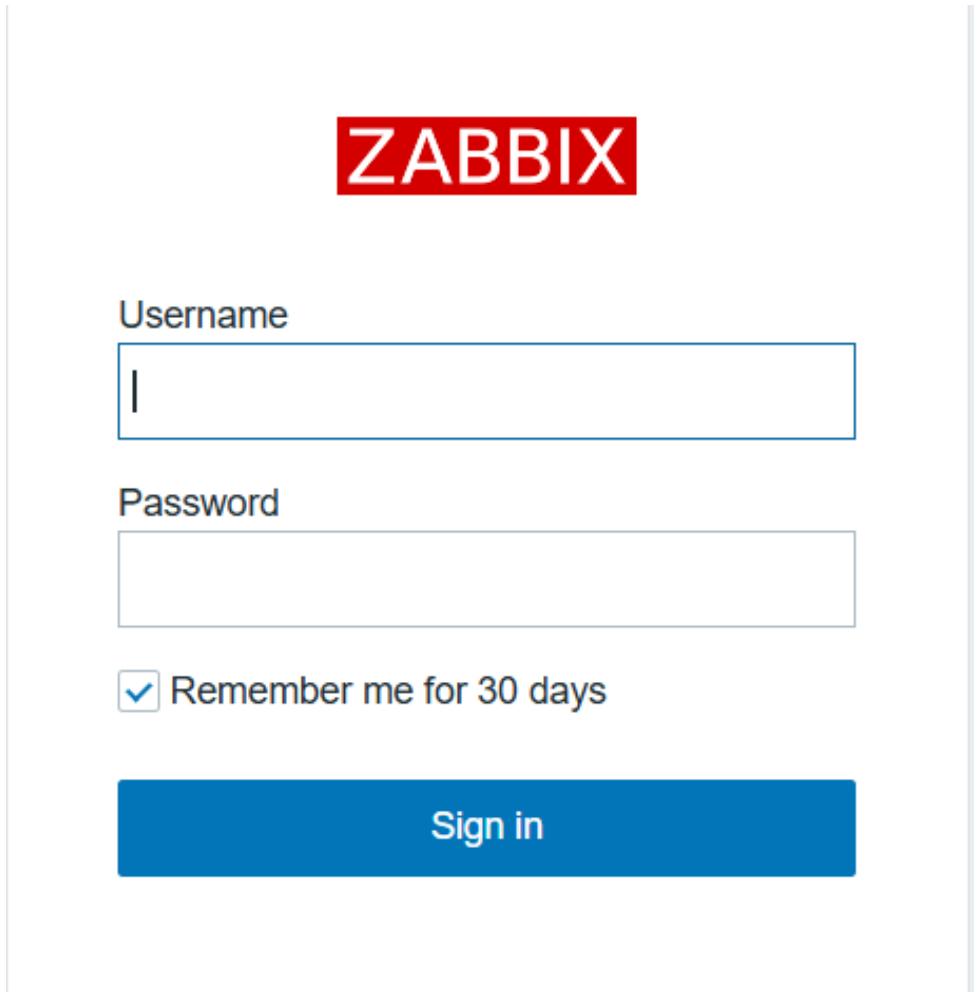


FIGURE 10 – Page de connexion à l'interface web de Zabbix

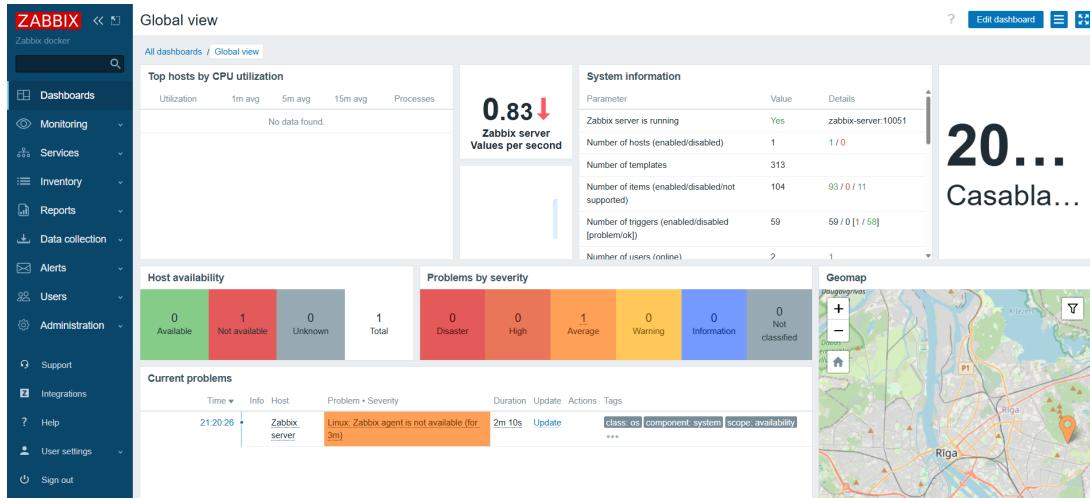


FIGURE 11 – Interface principale de Zabbix après connexion réussie

5 Configuration des Clients (Agents Zabbix)

5.1 Configuration du client Linux

5.1.1 Installation de l'agent

Sur l'instance client Linux, nous avons installé l'agent Zabbix.

```

1 wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix
      -release/zabbix-release_6.4-1+ubuntu22.04_all.deb
2 sudo dpkg -i zabbix-release_6.4-1+ubuntu22.04_all.deb
3 sudo apt update
4 sudo apt install zabbix-agent -y

```

Listing 3 – Installation de l'agent sur Linux

5.1.2 Configuration de l'agent

Le fichier de configuration a été modifié pour pointer vers le serveur Zabbix.

```

ServerActive=10.0.1.83

### 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-Ubuntu

```

FIGURE 12 – Configuration du fichier /etc/zabbix/zabbix_agentd.conf sur le client Linux

5.1.3 Démarrage du service

Le service a été démarré et configuré pour se lancer automatiquement.

```

ubuntu@ip-10-0-1-212:/etc/zabbix$ sudo systemctl restart zabbix-agent
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 13 – Démarrage du service zabbix-agent avec vérification de son statut

5.2 Configuration du client Windows

5.2.1 Installation de l'agent

Sur l'instance Windows, l'agent a été installé via l'installateur MSI.

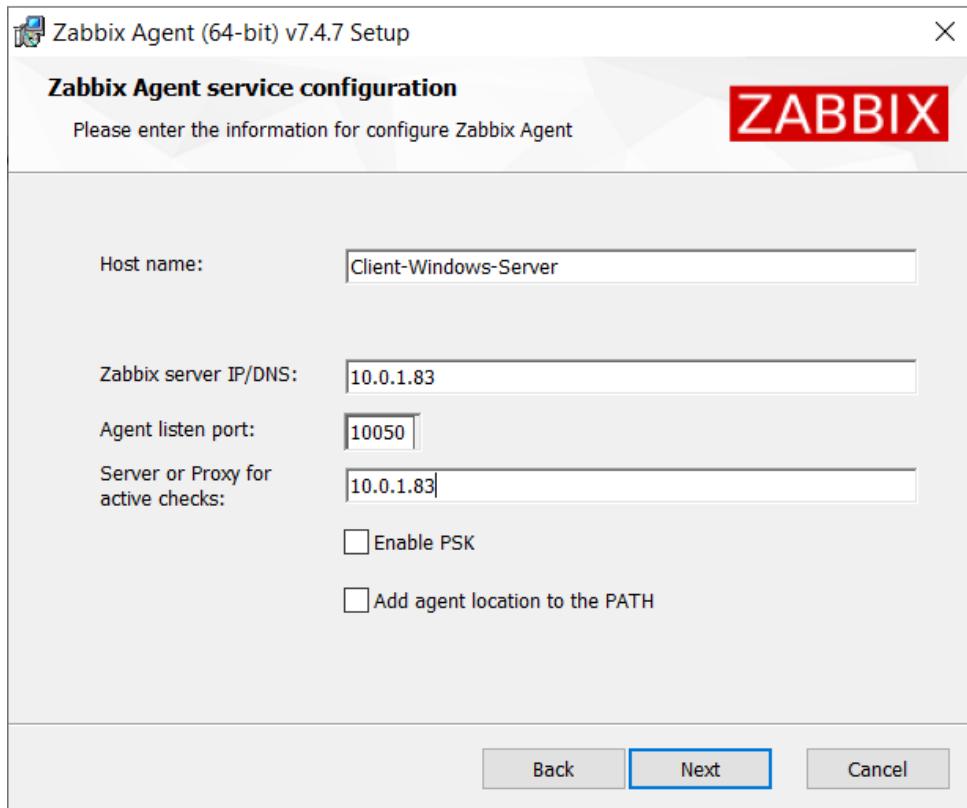


FIGURE 14 – Assistant d’installation de l’agent Zabbix pour Windows

5.2.2 Vérification du service

Le service Windows a été vérifié dans le gestionnaire de services.

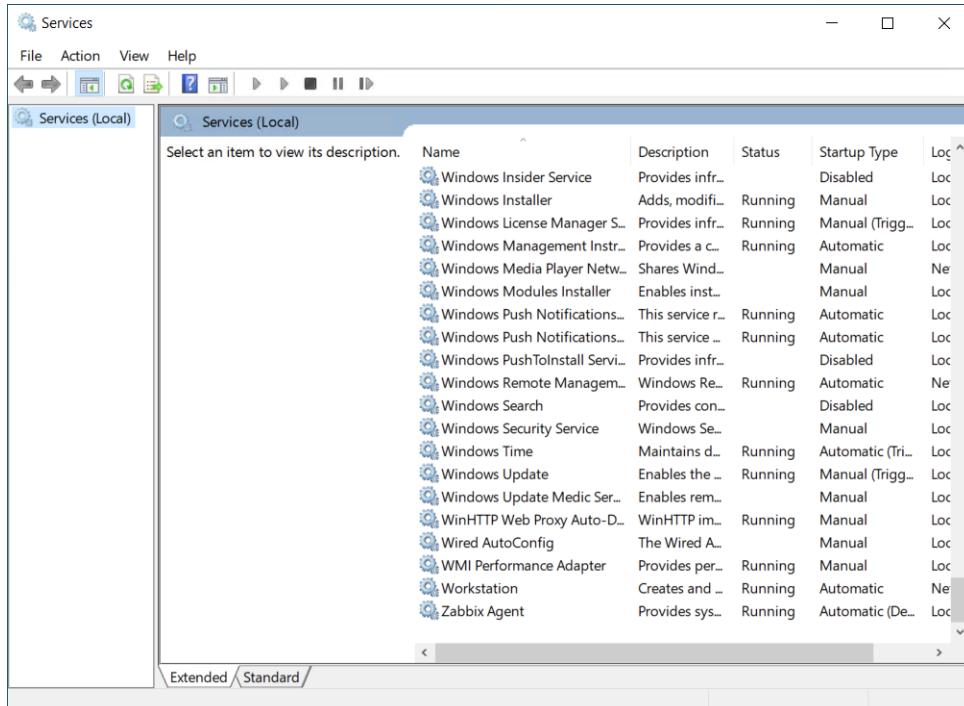


FIGURE 15 – Vérification du service 'Zabbix Agent' dans le gestionnaire de services Windows

6 Monitoring et Tableaux de Bord

6.1 Ajout des hôtes dans Zabbix

Dans l'interface web de Zabbix, nous avons ajouté les deux clients comme nouveaux hôtes.

New host

Host IPMI Tags Macros Inventory Encryption Value mapping

* Host name: Client-Linux-Ubuntu

Visible name: Client-Linux-Ubuntu

Templates: Linux by Zabbix agent

* Host groups: Linux servers

Interfaces

Type	IP address	DNS name	Connect to	Port	Default
Agent	10.0.1.212		IP	10050	<input checked="" type="radio"/> Remove

Add

Description:

Monitored by proxy: (no proxy)

Add Cancel

FIGURE 16 – Formulaire de création de l'hôte pour le client Linux

6.2 Statut des hôtes

Après quelques minutes, les deux hôtes apparaissent avec une disponibilité verte.

<input type="checkbox"/> Client-Linux-Ubuntu	Items 75	Triggers 30	Graphs 16	Discovery 3	Web	10.0.1.212:10050	Linux by Zabbix agent	Enabled	ZBX	None
<input type="checkbox"/> Client-Windows-Server	Items 48	Triggers 19	Graphs 8	Discovery 4	Web	10.0.1.192:10050	Windows by Zabbix agent	Enabled	ZBX	None

FIGURE 17 – Liste des hôtes dans Zabbix avec disponibilité VERTE (ZBX)

6.3 Visualisation des données

6.3.1 Client Linux

Le graphique suivant montre l'évolution de la charge CPU du client Linux.

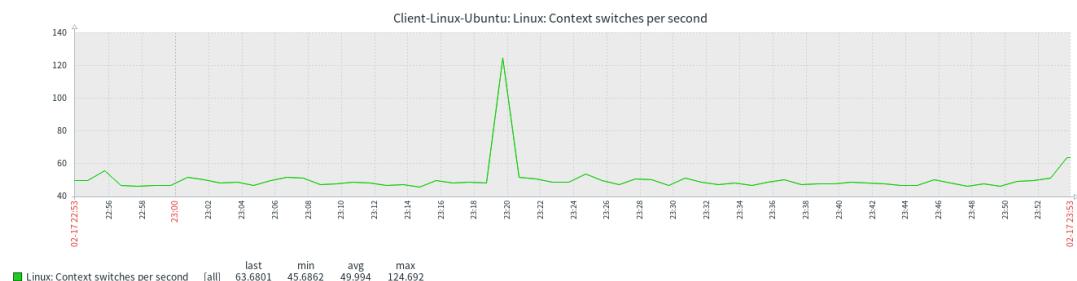


FIGURE 18 – Graphique de la charge CPU du client Linux sur les dernières minutes

6.3.2 Client Windows

Le graphique suivant montre l'utilisation mémoire du client Windows.

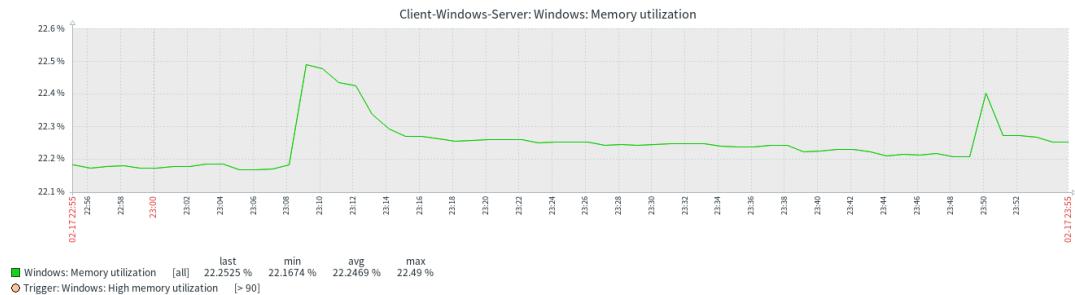


FIGURE 19 – Graphique de l'utilisation mémoire du client Windows

6.4 Données récentes

La vue des données récentes permet de vérifier tous les indicateurs collectés.

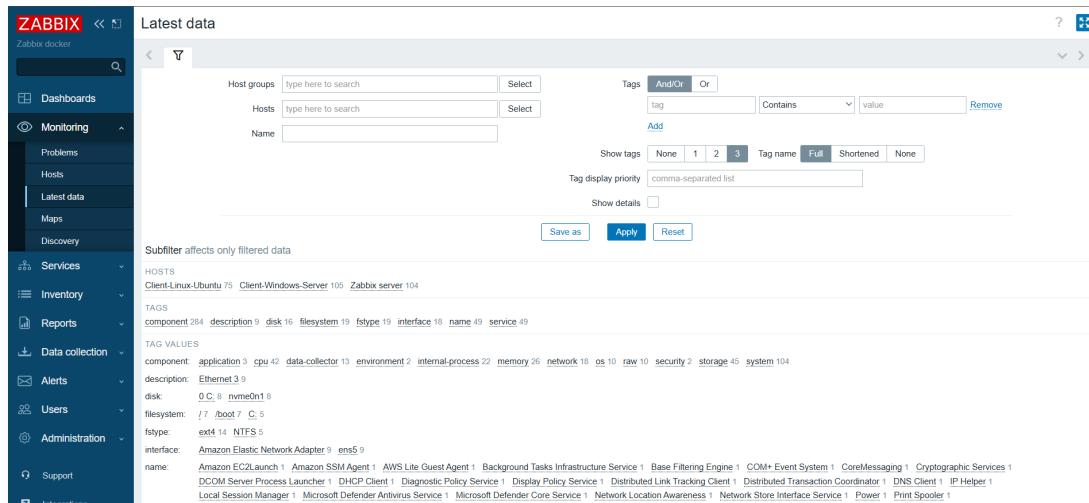


FIGURE 20 – Vue des données récentes pour le client Linux

7 Conclusion

7.1 Bilan du travail réalisé

Ce travail pratique a permis de déployer avec succès une infrastructure de monitoring centralisée sur AWS. L'objectif principal a été atteint : la mise en place d'un serveur Zabbix conteneurisé capable de surveiller un parc hybride composé de machines Linux et Windows.

Les principales réalisations incluent :

- Configuration complète d'un VPC avec sous-réseau public et groupes de sécurité
- Déploiement de trois instances EC2 avec les configurations appropriées
- Installation et configuration de Zabbix via Docker Compose
- Mise en place des agents sur les systèmes Linux et Windows
- Intégration réussie des hôtes dans l'interface Zabbix avec collecte de données

7.2 Difficultés rencontrées et solutions

Plusieurs difficultés ont été rencontrées durant ce TP :

- **Limitations du Learner Lab** : Les instances s'arrêtent automatiquement après une période d'inactivité. La solution a été de redémarrer systématiquement les conteneurs Docker avec `docker-compose up -d` après chaque reprise du Lab.
- **Correspondance des noms d'hôtes** : Un décalage entre le nom configuré dans l'agent et celui déclaré dans Zabbix empêchait la communication. La solution a été de vérifier rigoureusement que les noms correspondent exactement.
- **Flux réseau bloqués** : Initialement, les agents ne parvenaient pas à communiquer avec le serveur. La résolution a consisté à vérifier et ajuster les règles des groupes de sécurité pour autoriser les ports Zabbix depuis l'IP privée du serveur.

7.3 Améliorations possibles

Pour aller plus loin, plusieurs améliorations pourraient être apportées :

- Mise en place d'un Zabbix Proxy pour surveiller des réseaux distants
- Configuration d'alertes par email ou par d'autres canaux (Slack, Teams)
- Création de tableaux de bord personnalisés avec les métriques les plus critiques
- Surveillance d'autres services AWS via des templates spécifiques
- Automatisation du déploiement avec des outils comme Terraform ou CloudFormation

Ce TP a permis de démontrer la puissance et la flexibilité de la combinaison AWS/Docker/Zabbix pour la mise en place d'une solution de monitoring professionnelle, scalable et adaptée aux environnements hybrides.