

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

**Déploiement de Zabbix
conteneurisé (Docker)
pour le monitoring d'un parc hybride
(Linux & Windows)**

Nom étudiant :	Mohamed AASSOU
Encadrant :	Prof. Azeddine KHIAT
Filière :	2ANCI - Génie Informatique
Année universitaire :	2025/2026

Dépôt GitHub :

[https://github.com/mohammedaassou/
zabbix-docker-aws-monitoring.git](https://github.com/mohammedaassou/zabbix-docker-aws-monitoring.git)

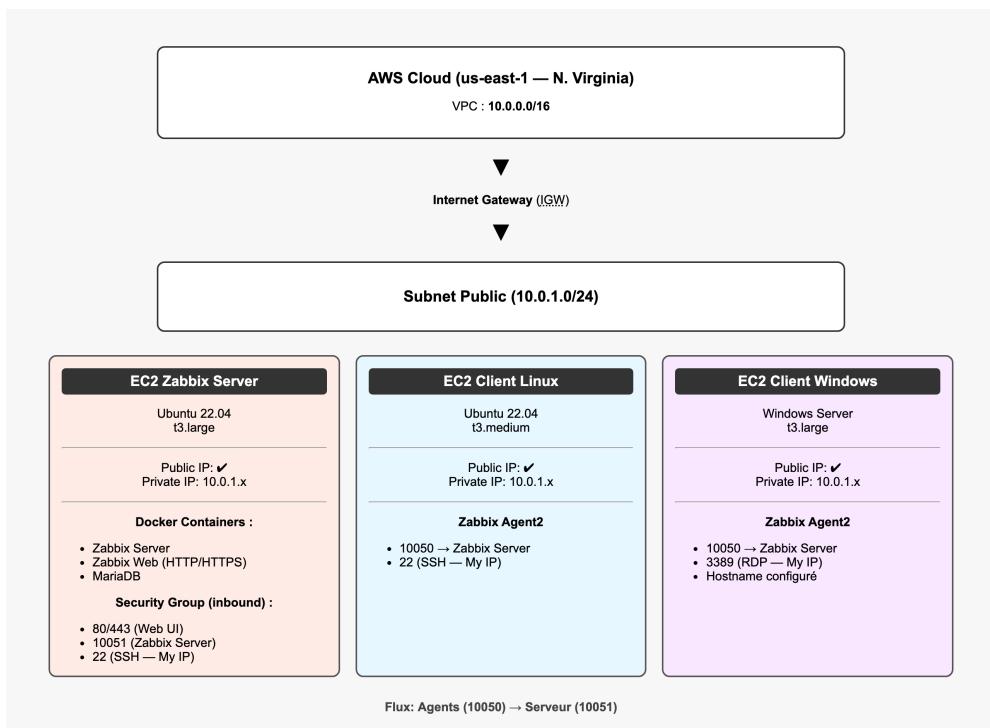


Figure de couverture : aperçu de l'architecture globale du projet.

Rapport de Projet de Fin d'Études

Table des matières

Table des matières	i
Liste des figures	ii
Liste des tableaux	iii
1 Introduction	1
1.1 Contexte	1
1.2 Objectifs techniques	1
1.3 Technologies utilisées	2
2 Architecture réseau	3
2.1 Description du VPC, subnet, IGW et route table	3
2.1.1 Adressage IP	3
2.1.2 Connectivité Internet	3
2.1.3 Captures de configuration réseau	4
2.2 Security Groups : tableau des ports	5
2.2.1 Principe de filtrage	6
2.2.2 Captures des Security Groups	7
2.3 Schéma réseau	8
3 Instances EC2	9
3.1 Tableau récapitulatif des instances	9
3.2 Justification des choix t3.medium / t3.large	9
3.3 Captures d'écran (lancement et état Running)	10
3.4 Configuration réseau des instances (Edit network)	11
4 Déploiement du serveur Zabbix (Docker)	13
4.1 Installation Docker	13
4.2 Installation Docker Compose	13
4.3 Fichier docker-compose.yml expliqué	14
4.4 Lancement des conteneurs	16
4.5 Vérification de l'interface Web	16
5 Configuration des agents	18
5.1 Agent Zabbix sur Linux (Ubuntu)	18
5.1.1 Installation	18

5.1.2 Configuration : zabbix_agent2.conf (extrait)	18
5.2 Agent Zabbix sur Windows Server	20
5.2.1 Installation MSI et paramètres	20
6 Monitoring & Dashboard	22
6.1 Ajout des hôtes	22
6.2 Validation : statut ZBX en vert	23
6.3 Graphiques CPU/RAM et tableaux de bord	23
7 Difficultés rencontrées & solutions	26
7.1 Ports bloqués / accès réseau	26
7.2 Docker indisponible après arrêt/redémarrage du lab	26
7.3 Limitations du Learner Lab	27
8 Conclusion	28
8.1 Résumé des résultats	28
8.2 Bénéfices du monitoring hybride	28
8.3 Améliorations futures possibles	29

Table des figures

2.1	Création du VPC (CIDR 10.0.0.0/16)	4
2.2	Création du subnet public (10.0.1.0/24)	4
2.3	Vérification du subnet (association et paramètres du subnet public)	5
2.4	Route Table : route par défaut 0.0.0.0/0 vers l'Internet Gateway	5
2.5	Security Group du serveur Zabbix (HTTP/HTTPS, 10051, SSH)	7
2.6	Security Group du client Linux (SSH + 10050 depuis le serveur Zabbix)	7
2.7	Security Group du client Windows (RDP + 10050 depuis le serveur Zabbix)	8
2.8	Schéma réseau : VPC (10.0.0.0/16), Subnet public (10.0.1.0/24), IGW et routage	8
3.1	Lancement de l'instance Zabbix Server (Ubuntu 22.04, t3.large)	10
3.2	Lancement de l'instance client Linux (Ubuntu 22.04, t3.medium)	10
3.3	Lancement de l'instance Windows Server (t3.large)	11
3.4	Vue globale des 3 instances EC2 (serveur Zabbix + clients Linux/Windows)	11
3.5	Configuration réseau : vérification/édition réseau de l'instance Zabbix Server (VPC/Subnet)	12
3.6	Configuration réseau : vérification/édition réseau du client Linux (VPC/-Subnet)	12
4.1	Création du fichier docker-compose.yml sur le serveur Zabbix	15
4.2	Téléchargement/initialisation des images Docker (premier lancement des services)	16
4.3	Vérification : accès à l'interface Web Zabbix	17
5.1	Configuration de l'agent Zabbix sur Linux (extrait de configuration)	19
5.2	Connexion SSH au client Linux pour installation et configuration de l'agent	19
5.3	Connexion RDP au serveur Windows pour finaliser l'installation de l'agent	20
5.4	Installation et configuration de l'agent Zabbix sur Windows Server	21
6.1	Ajout d'un hôte dans Zabbix (exemple : client Linux)	22
6.2	Validation de la connectivité : statut ZBX en vert pour Linux et Windows	23
6.3	Monitoring : consultation des dernières données (Latest data)	24
6.4	Exemple de graphique CPU/RAM (preuve de collecte effective)	24
6.5	Tableau de bord Zabbix : vue synthétique de supervision	25
6.6	Dashboard final : état global et indicateurs principaux	25

Liste des tableaux

2.1 Ports autorisés par les Security Groups (synthèse)	6
3.1 Récapitulatif des instances EC2 du projet	9
7.1 Contraintes et Bonnes Pratiques AWS Learner Lab	27
8.1 Pistes d'Amélioration et Évolutions	29

CHAPITRE 1

Introduction

1.1 Contexte

Dans les infrastructures informatiques modernes, les systèmes d'exploitation Linux et Windows cohabitent fréquemment. Cette hétérogénéité complique la supervision : les méthodes d'accès, les services, et les outils de collecte diffèrent. Une solution de monitoring centralisée permet de détecter rapidement les incidents, de suivre l'évolution des ressources (CPU, RAM, disque, réseau) et de garantir un niveau de disponibilité satisfaisant.

Le cloud **AWS** offre un environnement standardisé pour déployer rapidement un serveur de supervision accessible, tout en appliquant des règles de sécurité cohérentes via les **Security Groups**. Dans ce projet, la plateforme Zabbix est déployée sous forme de conteneurs Docker afin de faciliter la reproductibilité et la maintenance.

1.2 Objectifs techniques

Les objectifs du projet sont les suivants :

- Concevoir une architecture réseau AWS simple et conforme : un VPC, un subnet public, une Internet Gateway et une route table
- Déployer un serveur Zabbix conteneurisé (Docker) dans AWS (Zabbix Server + interface Web + base de données)
- Superviser un parc hybride : un client Linux (Ubuntu) et un client Windows Server, via l'installation et la configuration d'agents Zabbix
- Vérifier le bon fonctionnement par l'observation du statut des hôtes (ZBX en vert) et l'affichage de graphiques CPU/RAM

1.3 Technologies utilisées

Stack Technologique

- **AWS (Learner Lab)** : VPC, Subnet, Internet Gateway, Route Table, Security Groups, EC2
- **Docker & Docker Compose** : déploiement reproduitible des services Zabbix
- **Zabbix** : collecte de métriques et visualisation (tableaux de bord)
- **Ubuntu 22.04** : OS du serveur Zabbix et du client Linux
- **Windows Server** : OS du client Windows

CHAPITRE 2

Architecture réseau

2.1 Description du VPC, subnet, IGW et route table

L'architecture réseau est construite autour d'un VPC unique, avec un subnet public permettant un accès direct à Internet (sans VPN), conformément aux contraintes du lab.

L'objectif de cette partie est de démontrer :

- la **segmentation réseau** (VPC + subnet)
- la **connectivité Internet** (IGW + route par défaut)
- et la **sécurisation** (Security Groups) en ouvrant uniquement les ports nécessaires

2.1.1 Adressage IP

Configuration Réseau

- **VPC** : CIDR 10.0.0.0/16
- **Subnet public** : CIDR 10.0.1.0/24

2.1.2 Connectivité Internet

- Une **Internet Gateway (IGW)** est attachée au VPC
- Une **Route Table** associée au subnet public contient la route par défaut 0.0.0.0/0 pointant vers l'IGW

2.1.3 Captures de configuration réseau

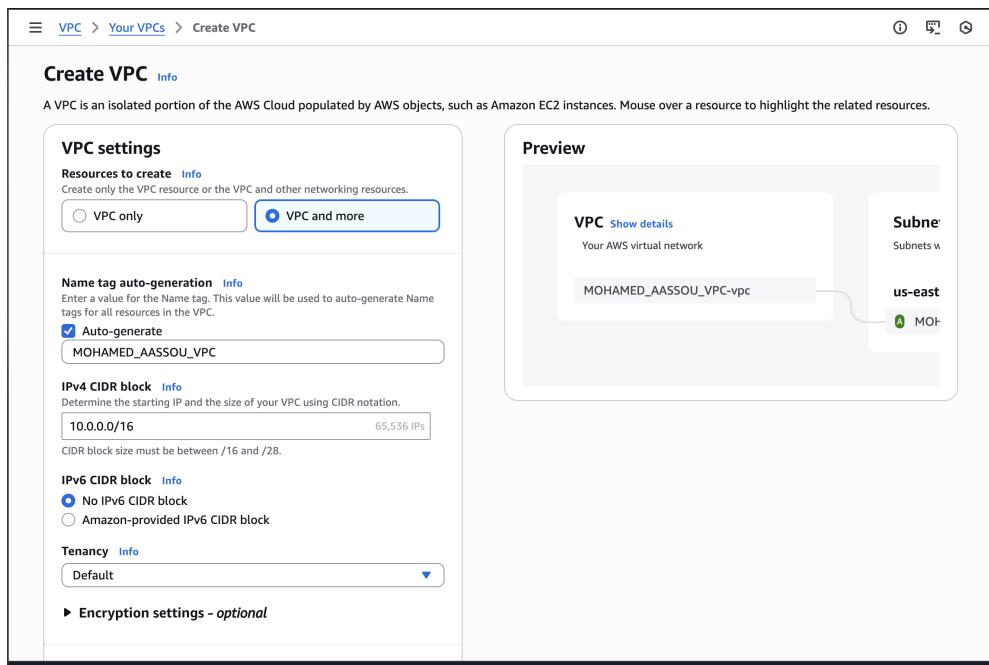


FIGURE 2.1 – Crédit du VPC (CIDR 10.0.0.0/16)

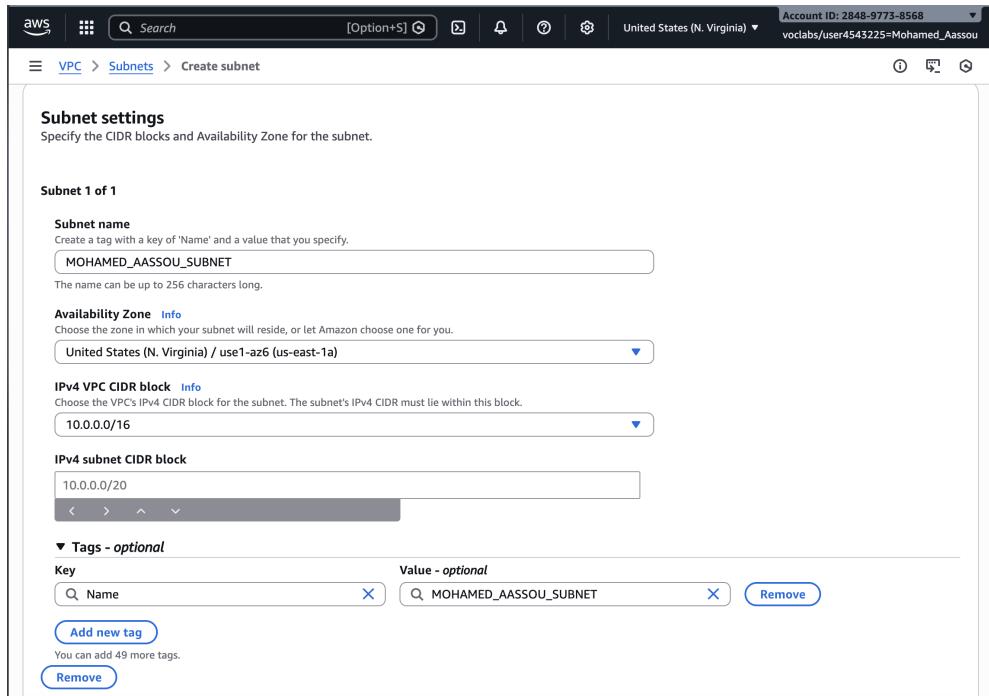


FIGURE 2.2 – Crédit du subnet public (10.0.1.0/24)

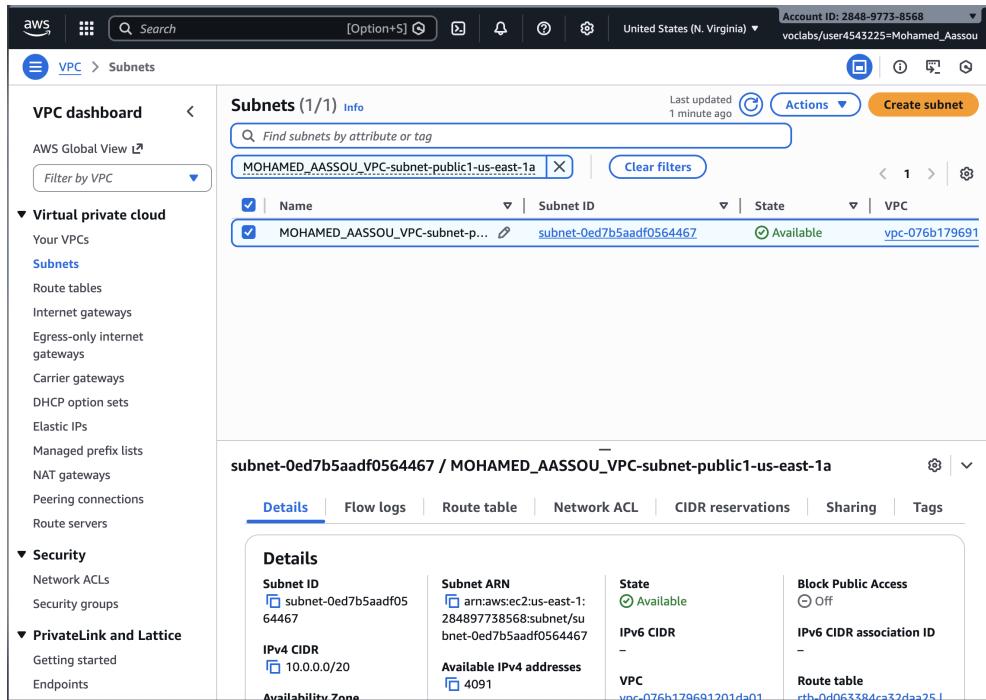


FIGURE 2.3 – Vérification du subnet (association et paramètres du subnet public)

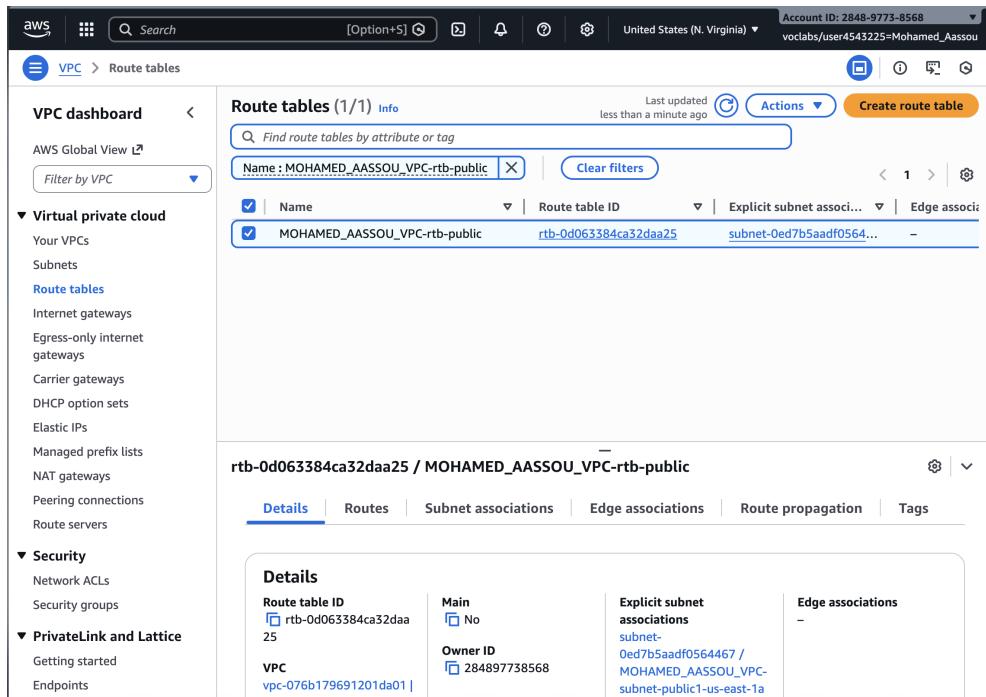


FIGURE 2.4 – Route Table : route par défaut 0.0.0.0/0 vers l’Internet Gateway

2.2 Security Groups : tableau des ports

Les règles de sécurité ouvrent uniquement les ports requis par le projet.

2.2.1 Principe de filtrage

Sécurité - Principe du moindre privilège

Les règles sont définies selon le principe du moindre privilège :

- **Administration** (SSH/RDP/Web) limitée à *My IP*
- **Monitoring** : l'agent (10050) n'accepte que les requêtes venant du serveur Zabbix

TABLE 2.1 – Ports autorisés par les Security Groups (synthèse)

Service / Usage	Port	Cible	Rôle
Interface Web Zabbix (HTTP/HTTPS)	80 / 443	Zabbix Server	Accès à l'UI depuis l'extérieur (My IP)
Zabbix Server	10051	Zabbix Server	Réception / échanges côté serveur
Zabbix Agent	10050	Clients Linux/Windows	Collecte des métriques par le serveur Zabbix
Administration SSH	22	Instances Ubuntu	Administration distante (My IP)
Administration RDP	3389	Instance Windows	Administration distante (My IP)

2.2.2 Captures des Security Groups

The screenshot shows the AWS VPC Security Groups interface for creating a new security group named "Mohamed_5G-Zabbix-Server". The "Inbound rules" section contains one rule:

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	My IP 41.140.152.223/32	SSH-admin-access

FIGURE 2.5 – Security Group du serveur Zabbix (HTTP/HTTPS, 10051, SSH)

The screenshot shows the AWS EC2 Security Groups interface for editing the inbound rules of a security group associated with a Linux client. The "Inbound rules" section contains two rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-Off2484d4d7401598	Custom TCP	TCP	10050	Custom 10.0.0.0/16	Zabbix-agent
sgr-0cb232bb07e655027	SSH	TCP	22	Custom 41.140.152.223/32	SSH-admin-access

FIGURE 2.6 – Security Group du client Linux (SSH + 10050 depuis le serveur Zabbix)

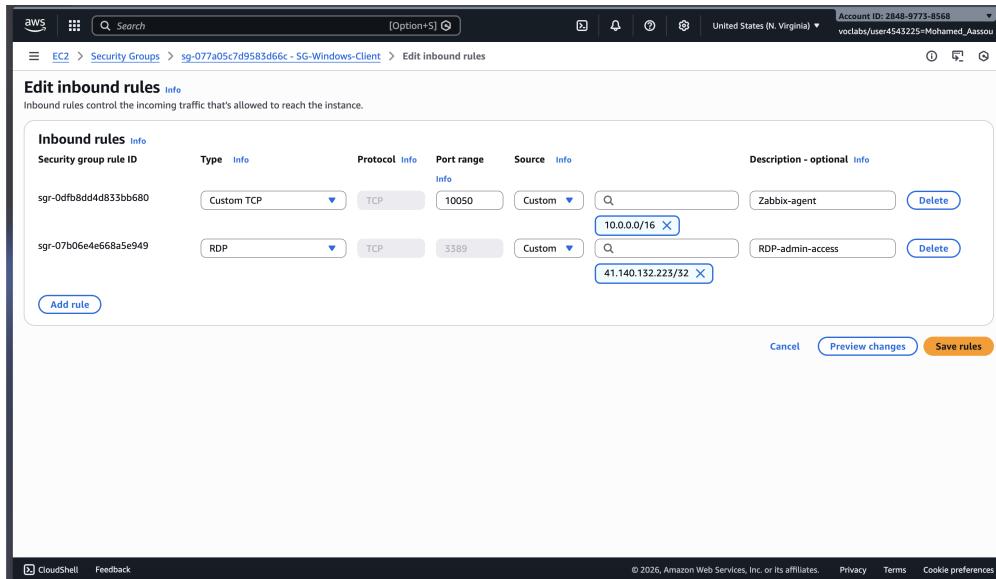


FIGURE 2.7 – Security Group du client Windows (RDP + 10050 depuis le serveur Zabbix)

2.3 Schéma réseau

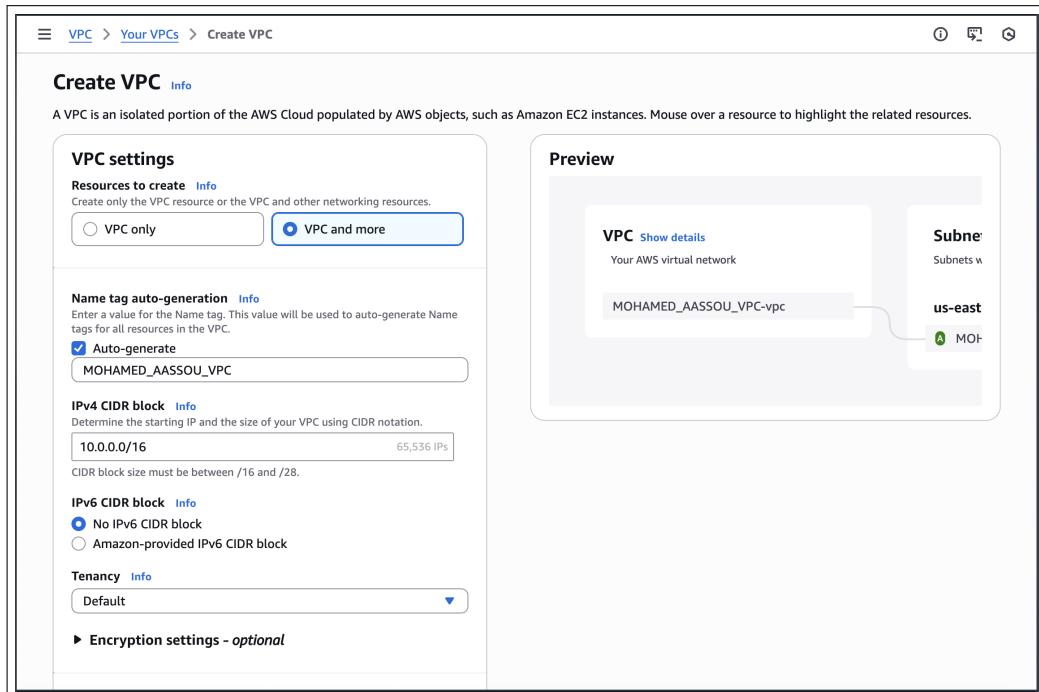


FIGURE 2.8 – Schéma réseau : VPC (10.0.0.0/16), Subnet public (10.0.1.0/24), IGW et routage

CHAPITRE 3

Instances EC2

3.1 Tableau récapitulatif des instances

Les instances EC2 déployées sont conformes aux types autorisés et recommandés par le lab.

TABLE 3.1 – Récapitulatif des instances EC2 du projet

Rôle	Type	OS	vCPU	RAM (GiB)	Fonction
Serveur Zabbix	t3.large	Ubuntu 22.04	2	8	Héberge Docker : Zabbix Server + Web + DB
Client Linux	t3.medium	Ubuntu 22.04	2	4	Agent Zabbix pour métriques système Linux
Client Windows	t3.large	Windows Server	2	8	Agent Zabbix pour métriques Windows

3.2 Justification des choix t3.medium / t3.large

Dimensionnement des Instances

- **Serveur Zabbix (t3.large)** : La conteneurisation (serveur + web + base) nécessite une marge de CPU/RAM pour éviter les lenteurs. Les 8 GiB de RAM permettent d'exécuter PostgreSQL, Zabbix Server et l'interface web simultanément sans saturation mémoire.
- **Client Linux (t3.medium)** : L'agent Zabbix est léger et les charges de test restent modérées. Les 4 GiB de RAM sont largement suffisants pour Ubuntu 22.04 + agent de monitoring.
- **Client Windows (t3.large)** : Windows Server consomme davantage de ressources au repos (\approx 2–3 GiB). Ce type améliore la fluidité (RDP et services) et garantit des performances acceptables pour la collecte de métriques.

3.3 Captures d'écran (lancement et état Running)

Cette partie illustre la création des instances, puis la vérification de leur état. Dans un contexte de lab, il est important de contrôler :

- le **type d'instance** (t3.medium/t3.large)
- la **région** (us-east-1)
- l'**adresse IP publique** (pour l'accès Web/SSH/RDP)
- et le **Security Group** associé

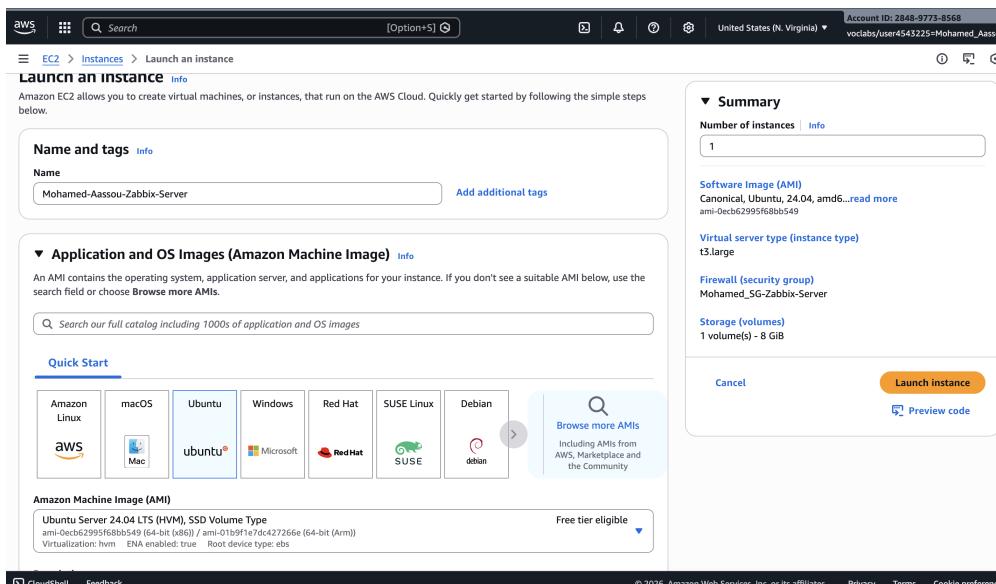


FIGURE 3.1 – Lancement de l'instance Zabbix Server (Ubuntu 22.04, t3.large)

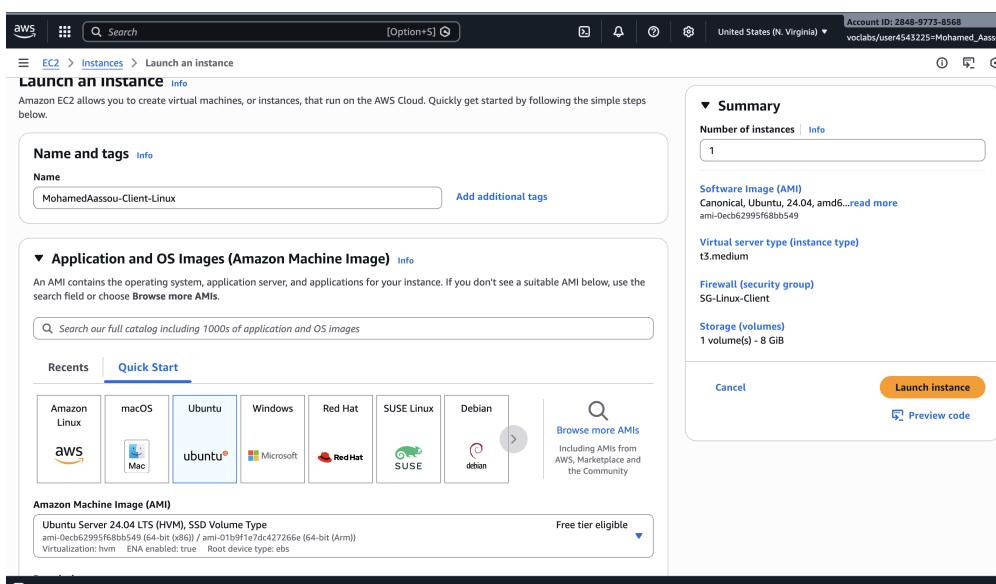


FIGURE 3.2 – Lancement de l'instance client Linux (Ubuntu 22.04, t3.medium)

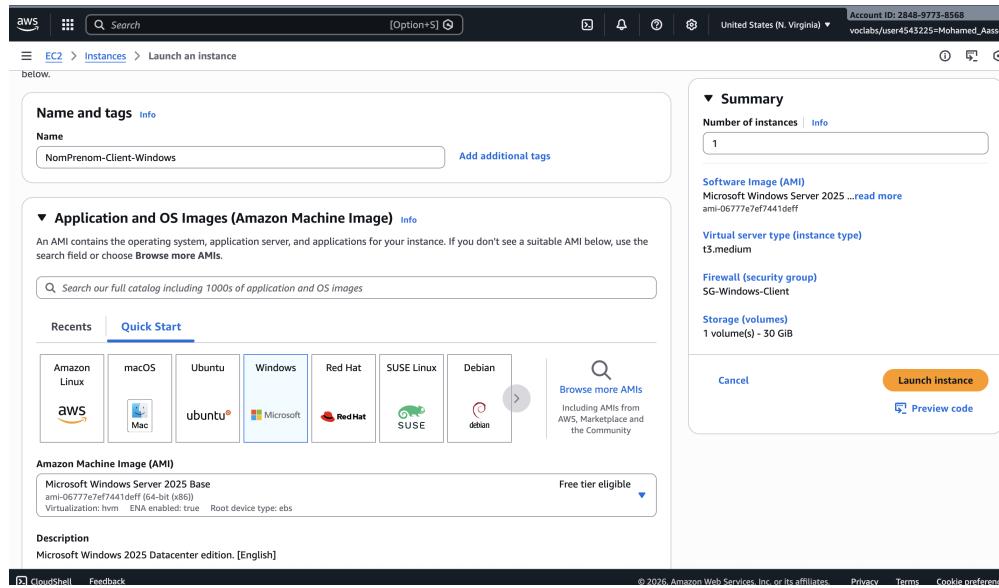


FIGURE 3.3 – Lancement de l’instance Windows Server (t3.large)

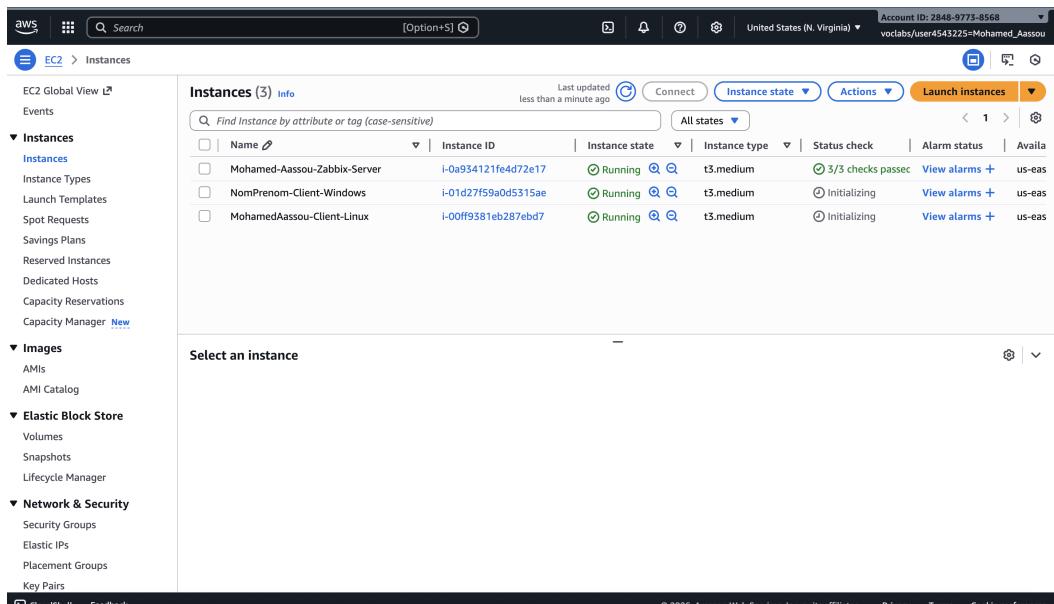


FIGURE 3.4 – Vue globale des 3 instances EC2 (serveur Zabbix + clients Linux/Windows)

3.4 Configuration réseau des instances (Edit network)

Après le lancement, il peut être nécessaire de vérifier (ou ajuster) la configuration réseau de chaque instance : VPC, subnet, et paramètres d’interface réseau. Les captures suivantes illustrent ce contrôle *après création*.

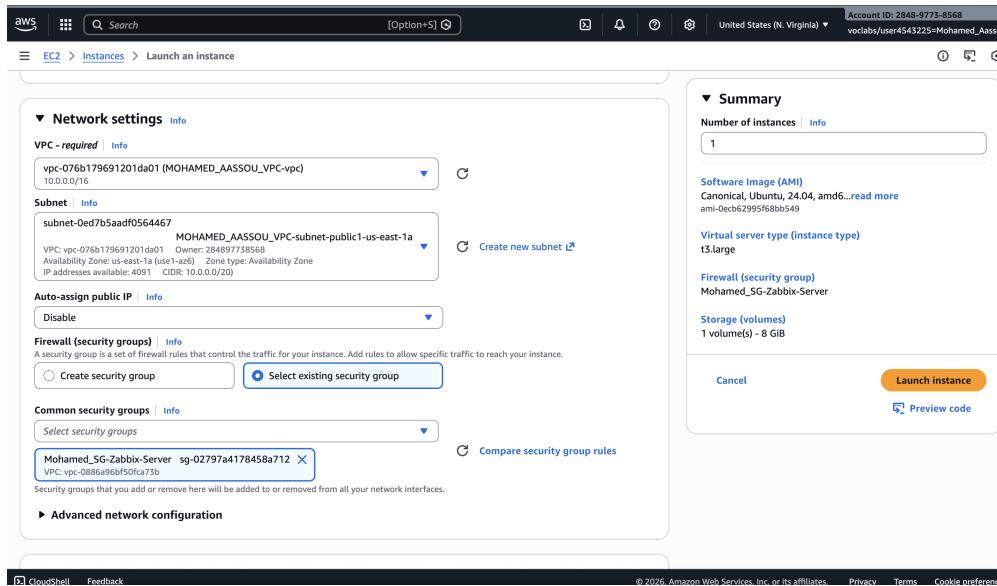


FIGURE 3.5 – Configuration réseau : vérification/édition réseau de l’instance Zabbix Server (VPC/Subnet)

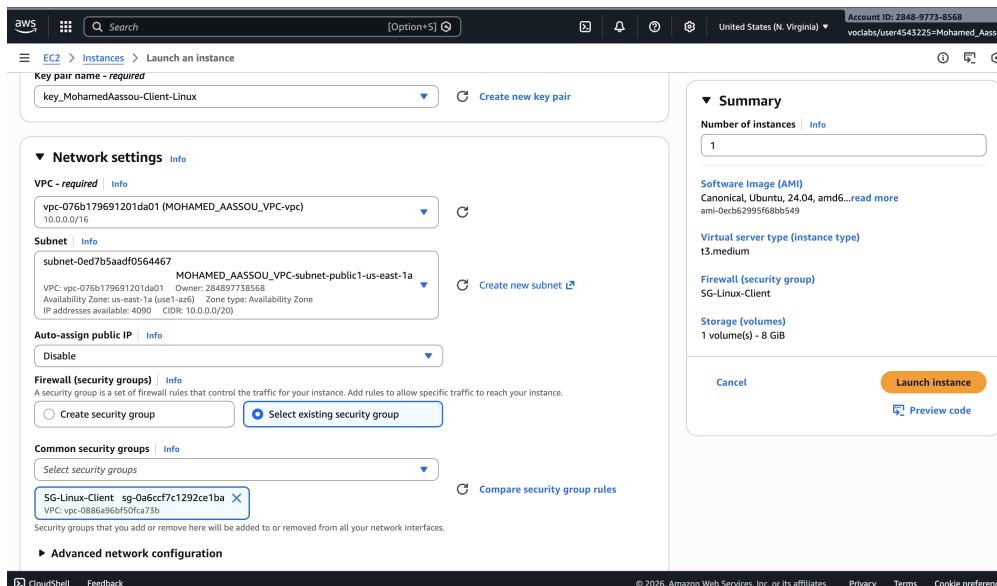


FIGURE 3.6 – Configuration réseau : vérification/édition réseau du client Linux (VPC/Subnet)

CHAPITRE 4

Déploiement du serveur Zabbix (Docker)

4.1 Installation Docker

Sur Ubuntu 22.04, Docker est installé via le dépôt officiel. Les commandes suivantes illustrent une procédure type.

```
1 sudo apt update
2 sudo apt install -y ca-certificates curl gnupg
3
4 sudo install -m 0755 -d /etc/apt/keyrings
5 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
   -o /etc/apt/keyrings/docker.gpg
6 sudo chmod a+r /etc/apt/keyrings/docker.gpg
7
8 echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/
   docker.gpg] \
9 https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo
   $VERSION_CODENAME) stable" \
10 | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
11
12 sudo apt update
13 sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-
   plugin docker-compose-plugin
14 sudo systemctl enable --now docker
```

Listing 4.1 – Installation de Docker sur Ubuntu (serveur Zabbix)

4.2 Installation Docker Compose

Docker Compose est utilisé via le plugin officiel `docker compose`. Cela évite d'ajouter une dépendance externe supplémentaire.

4.3 Fichier docker-compose.yml expliqué

Le déploiement est réalisé via un fichier Compose contenant :

- **PostgreSQL** : base de données Zabbix
- **Zabbix Server** : service principal (port 10051)
- **Zabbix Web** : interface Web (exposée sur le port 80 de l'instance)

```
1 version: '3.8'
2
3 services:
4   postgres:
5     image: postgres:16-alpine
6     container_name: zabbix-postgres
7     environment:
8       POSTGRES_DB: zabbix
9       POSTGRES_USER: zabbix
10      POSTGRES_PASSWORD: zabbixpass
11     volumes:
12       - pgdata:/var/lib/postgresql/data
13     restart: unless-stopped
14   networks:
15     - zabbix-net
16
17 zabbix-server:
18   image: zabbix/zabbix-server-pgsql:alpine-7.0-latest
19   container_name: zabbix-server
20   depends_on:
21     - postgres
22   environment:
23     DB_SERVER_HOST: postgres
24     POSTGRES_DB: zabbix
25     POSTGRES_USER: zabbix
26     POSTGRES_PASSWORD: zabbixpass
27   ports:
28     - "10051:10051"
29   restart: unless-stopped
30   networks:
31     - zabbix-net
32
33 zabbix-web:
```

```

34   image: zabbix/zabbix-web-nginx-pgsql:alpine-7.0-latest
35   container_name: zabbix-web
36   depends_on:
37     - postgres
38     - zabbix-server
39   environment:
40     DB_SERVER_HOST: postgres
41     POSTGRES_DB: zabbix
42     POSTGRES_USER: zabbix
43     POSTGRES_PASSWORD: zabbixpass
44     ZBX_SERVER_HOST: zabbix-server
45     PHP_TZ: Africa/Casablanca
46   ports:
47     - "80:8080"
48   restart: unless-stopped
49   networks:
50     - zabbix-net
51
52 volumes:
53   pgdata:
54
55 networks:
56   zabbix-net:
57     driver: bridge

```

Listing 4.2 – Fichier docker-compose.yml (extrait du projet)

```

services:
  postgres:
    image: postgres:16-alpine
    container_name: zabbix-postgres
    environment:
      POSTGRES_DB: zabbix
      POSTGRES_USER: zabbix
      POSTGRES_PASSWORD: zabbixpass
    volumes:
      - pgdata:/var/lib/postgresql/data
    restart: unless-stopped

  zabbix-server:
    image: zabbix/zabbix-server-pgsql:alpine-7.0-latest
    container_name: zabbix-server
    depends_on:
      - postgres
    environment:
      DB_SERVER_HOST: postgres
      DB_NAME: zabbix
      POSTGRES_USER: zabbix
      POSTGRES_PASSWORD: zabbixpass
    ports:
      - "10051:10051"
    restart: unless-stopped

  zabbix-web:
    image: zabbix/zabbix-web-nginx-pgsql:alpine-7.0-latest
    container_name: zabbix-web
    depends_on:
      - postgres
      - zabbix-server
    environment:
      DB_SERVER_HOST: postgres
      POSTGRES_DB: zabbix
      POSTGRES_USER: zabbix
      POSTGRES_PASSWORD: zabbixpass
      ZBX_SERVER_HOST: zabbix-server
      PHP_TZ: Africa/Casablanca
    ports:
      - "80:8080"
    restart: unless-stopped

volumes:
  pgdata:

```

FIGURE 4.1 – Création du fichier docker-compose.yml sur le serveur Zabbix

```
Last login: Thu Jan  1 11:58:21 2026 from 41.140.132.223
ubuntu@ip-10-0-4-148:~$ nano docker-compose.yml
ubuntu@ip-10-0-4-148:~$ ls
docker-compose.yml
ubuntu@ip-10-0-4-148:~$ docker compose up -d
[*] up 35/36
  ✓ Image postgres:13-alpine          Pulled
  ✓ Image zabbix/zabbix-server-pgsql:alpine-7.0-latest Pulled
  ✓ Image zabbix/zabbix-web-nginx-pgsql:alpine-7.8-latest Pulled
  ✓ Network ubuntu_default             Created
  ✓ Volume ubuntu_pgdata              Created
  ✓ Container zabbix-postgres        Created
  ✓ Container zabbix-server          Created
  ✓ Container zabbix-web            Created
ubuntu@ip-10-0-4-148:~$
```

FIGURE 4.2 – Téléchargement/initialisation des images Docker (premier lancement des services)

4.4 Lancement des conteneurs

```
1 cd /chemin/vers/le/projet
2 docker compose up -d
3 docker compose ps
```

Listing 4.3 – Démarrage des conteneurs Zabbix

4.5 Vérification de l'interface Web

Une fois les conteneurs démarrés, l'interface est accessible via : <http://<IP-Publique-Serveur-Zabbix>>

Points de Contrôle

Le point de contrôle attendu ici est simple :

- Le navigateur affiche la page d'authentification Zabbix
- Les conteneurs `zabbix-server` et `zabbix-web` sont **Up**
- Le port 80 est bien autorisé dans le Security Group du serveur

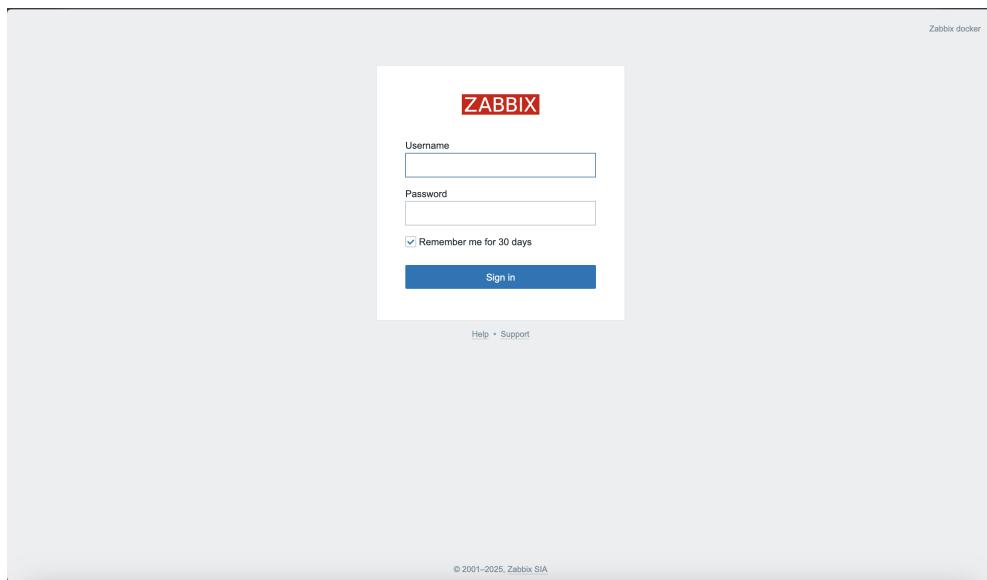


FIGURE 4.3 – Vérification : accès à l'interface Web Zabbix

CHAPITRE 5

Configuration des agents

5.1 Agent Zabbix sur Linux (Ubuntu)

L'agent Linux a pour rôle d'exposer des métriques système (CPU, mémoire, disque, services). Le serveur Zabbix interroge l'agent sur le port 10050.

5.1.1 Installation

```
1 sudo apt update
2 sudo apt install -y zabbix-agent2
3 sudo systemctl enable --now zabbix-agent2
4 sudo systemctl status zabbix-agent2 --no-pager
```

Listing 5.1 – Installation de l'agent Zabbix sur Ubuntu

5.1.2 Configuration : zabbix_agent2.conf (extrait)

Le paramétrage essentiel consiste à déclarer le serveur Zabbix autorisé et le nom d'hôte.

```
1 Server=10.0.1.<IP_PRISEE_ZABBIX_SERVER>
2 ServerActive=10.0.1.<IP_PRISEE_ZABBIX_SERVER>
3 Hostname=MohamedAASSOU-Client-Linux
```

Listing 5.2 – Extrait de configuration agent Linux (Server/ServerActive/Hostname)

Configuration Agent Linux

Paramètres essentiels :

- **Server** : IP privée du serveur Zabbix (communications passives)
- **ServerActive** : IP privée du serveur Zabbix (communications actives)
- **Hostname** : doit correspondre exactement au nom déclaré dans l'interface Zabbix

```

# Mandatory: no
# Range: 1024-32767
# Default:
# StatusPort=10051

##### Active checks related

## Option: ServerActive
# Zabbix server/proxy address or cluster configuration to get active checks from.
# Server/proxy address is IP address or DNS name and optional port separated by colon.
# Cluster configuration is one or more server addresses separated by semicolon.
# More than one Zabbix proxy should not be specified from each Zabbix server/cluster.
# If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.
# It is recommended 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.
# IP ports can be specified in range notation, colons are optional.
# If this parameter is not specified, active checks are disabled.
# Example for Zabbix proxy:
#     ServerActive127.0.0.1:10051
# Example for multiple servers:
#     ServerActive127.0.0.1:20051;zabbix.domain,[::1]:30051,::1,[12fc::1]
# Example for high availability:
#     ServerActive127.0.0.1:20051;zabbix.cluster.node1:20051;zabbix.cluster.node3
# Example for high availability with two clusters and one server:
#     ServerActive1zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node2,zabbix.domain

# Mandatory: no
# Default:
# ServerActive=
ServerActive=10.0.4.1:10051

##### 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=MohamedAassou-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

## Option: HostMetadata
# Optional parameter that defines host metadata.
# Host metadata is used at host auto-registration process.
# An agent will issue an error and not start if the value is over limit of 2034 bytes.
# If not defined, value will be acquired from HostmetadataItem

```

FIGURE 5.1 – Configuration de l'agent Zabbix sur Linux (extrait de configuration)

```

Downloads ssh -i "key_MohamedAassou-Client-Linux.pem" ubuntu@ec2-98-89-227-174.compute-1.amazonaws.com
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

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

System information as of Thu Jan  1 12:10:25 UTC 2026

 System load:  0.0              Temperature:        -273.1 C
 Usage of /:   25.8% of 6.71GB  Processes:          108
 Memory usage: 5%               Users logged in:    0
 Swap usage:   0%               IPV4 address for ens5: 10.0.11.27

Xpanded Security Maintenance for Applications is not enabled.

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-10-0-11-27:~$ 

```

FIGURE 5.2 – Connexion SSH au client Linux pour installation et configuration de l'agent

5.2 Agent Zabbix sur Windows Server

5.2.1 Installation MSI et paramètres

L'agent Zabbix peut être installé via un exécutable MSI. Lors de l'installation, il faut renseigner :

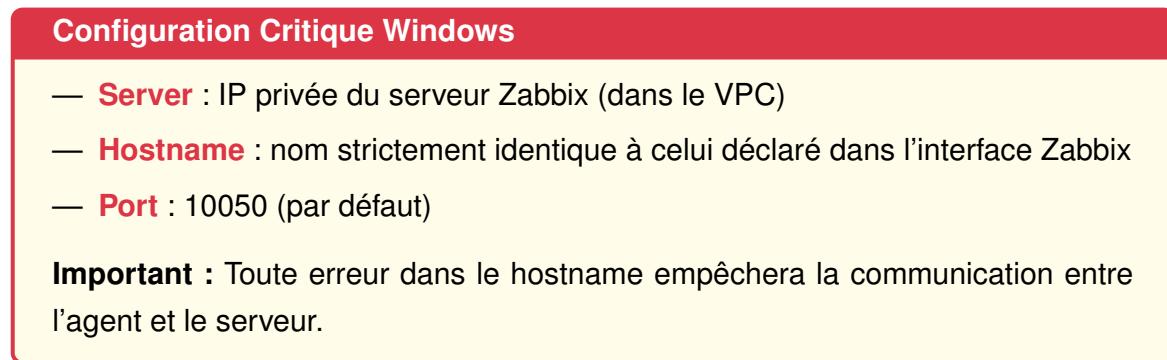


FIGURE 5.3 – Connexion RDP au serveur Windows pour finaliser l'installation de l'agent

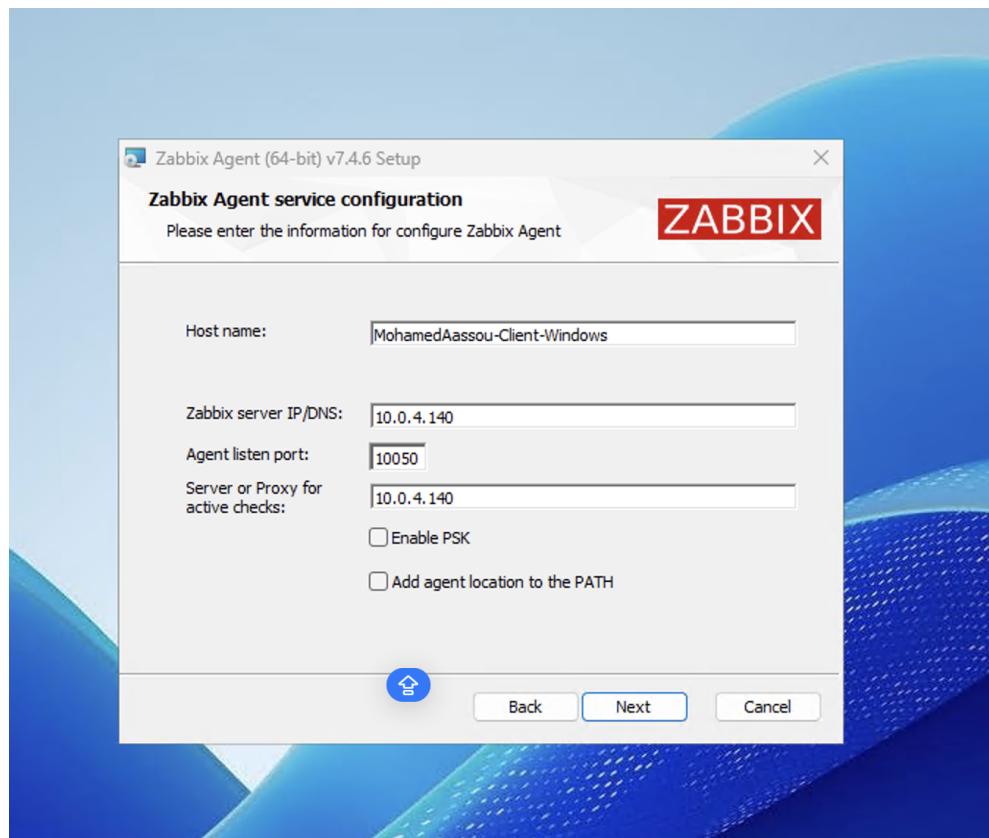


FIGURE 5.4 – Installation et configuration de l'agent Zabbix sur Windows Server

CHAPITRE 6

Monitoring & Dashboard

6.1 Ajout des hôtes

Dans l'interface Zabbix, l'ajout des hôtes suit une logique identique : définir un *Host name*, associer une interface *Agent* (IP privée du client, port 10050) et sélectionner un template correspondant (Linux/Windows).

Processus d'Ajout d'Hôte

Étapes principales :

1. Naviguer vers Configuration → Hosts → Create host
2. Renseigner le **Host name** (doit correspondre au hostname de l'agent)
3. Ajouter une interface **Agent** avec l'IP privée du client
4. Sélectionner le template approprié :
 - [Linux by Zabbix agent](#) pour Ubuntu
 - [Windows by Zabbix agent](#) pour Windows Server
5. Vérifier que le port 10050 est bien ouvert dans le Security Group

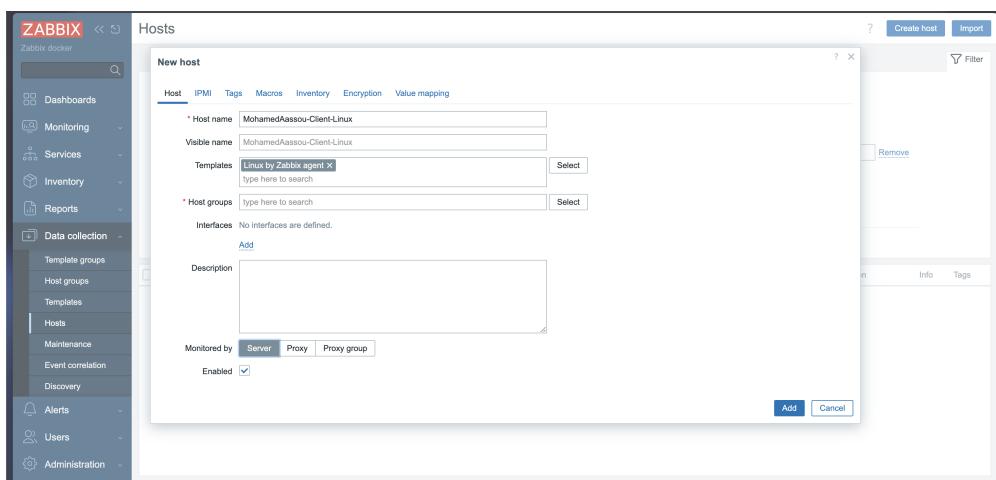


FIGURE 6.1 – Ajout d'un hôte dans Zabbix (exemple : client Linux)

6.2 Validation : statut ZBX en vert

Lorsque les ports sont correctement ouverts et l'agent opérationnel, Zabbix indique la disponibilité en vert (ZBX).

Indicateurs de Succès

- Icône **ZBX verte** : l'agent répond correctement
- Données récentes visibles dans *Latest data*
- Aucun message d'erreur dans les logs de l'agent

Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent encryption	Info	Tags
MohamedAassou-Client-Linux	Items 75	Triggers 30	Graphs 16	Discovery 3	Web 10.0.11.27:10050		Linux by Zabbix agent		Enabled	ZBX	None		
MohamedAassou-Client-Windows	Items 111	Triggers 77	Graphs 12	Discovery 4	Web 10.0.11.125:10050		Windows by Zabbix agent		Enabled	ZBX	None		
Zabbix server	Items 121	Triggers 68	Graphs 8	Discovery 6	Web 127.0.0.1:10050		Linux by Zabbix agent, Zabbix server health		Enabled	ZBX	None		

FIGURE 6.2 – Validation de la connectivité : statut ZBX en vert pour Linux et Windows

6.3 Graphiques CPU/RAM et tableaux de bord

Les graphiques permettent de suivre l'évolution des ressources et d'illustrer la collecte effective des métriques.

FIGURE 6.3 – Monitoring : consultation des dernières données (Latest data)

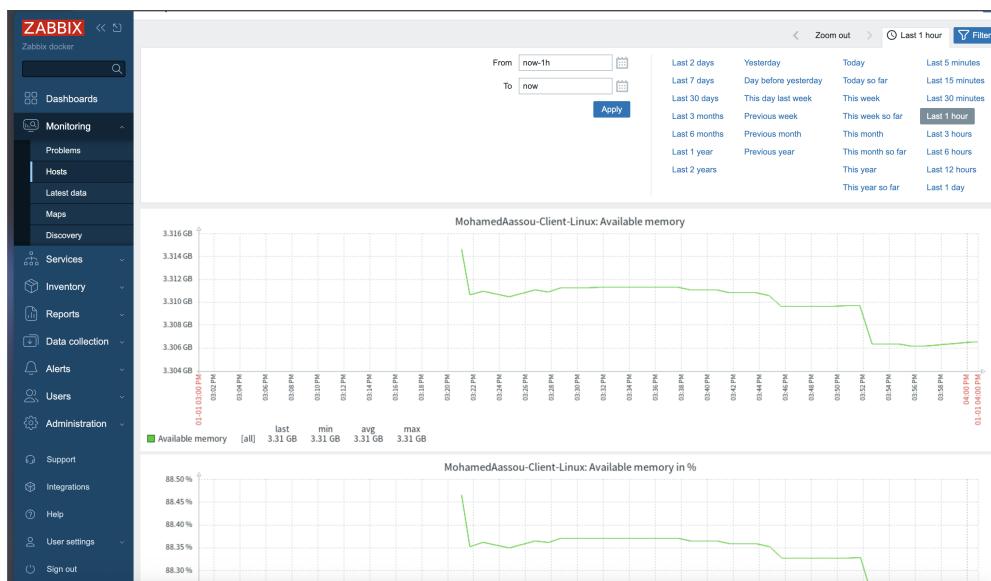


FIGURE 6.4 – Exemple de graphique CPU/RAM (preuve de collecte effective)

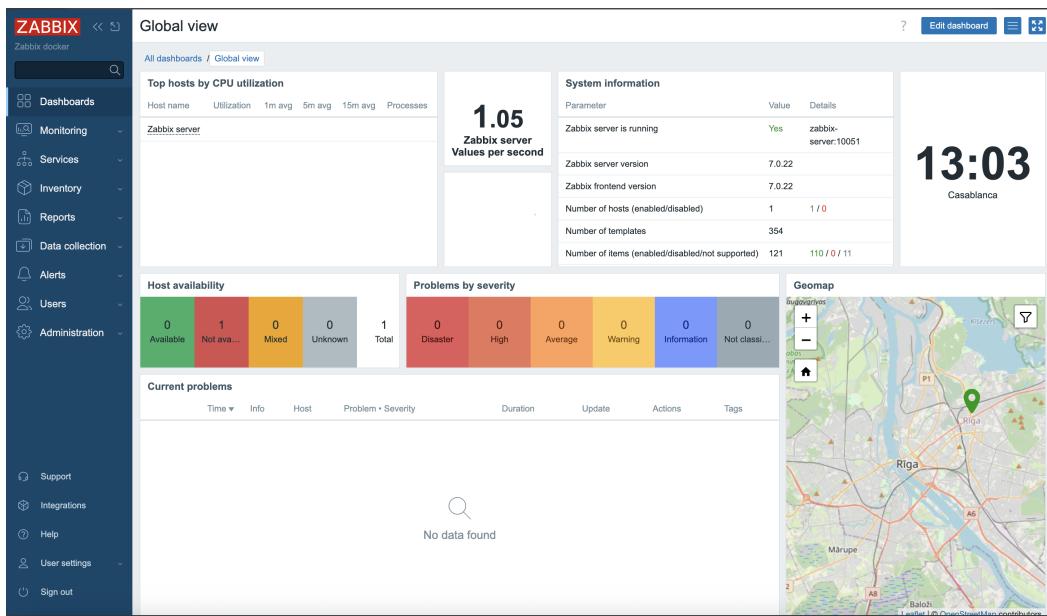


FIGURE 6.5 – Tableau de bord Zabbix : vue synthétique de supervision

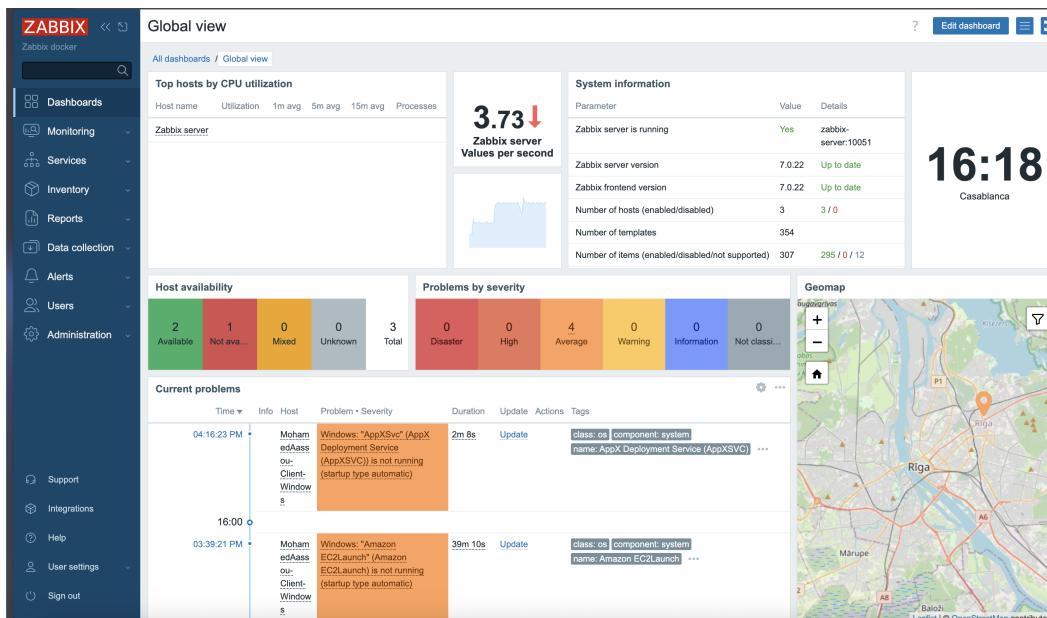


FIGURE 6.6 – Dashboard final : état global et indicateurs principaux

CHAPITRE 7

Difficultés rencontrées & solutions

7.1 Ports bloqués / accès réseau

Problème Rencontré

Symptôme : Les hôtes apparaissent indisponibles (pas de données, ZBX rouge) si les ports 10050/10051 ne sont pas correctement autorisés.

Solution Appliquée

Actions correctives :

- Vérifier les règles des Security Groups dans la console AWS
- Limiter les sources (My IP pour SSH/RDP/Web)
- Autoriser le port 10050 depuis l'IP privée du serveur Zabbix vers les clients
- Tester la connectivité avec `telnet <IP_CLIENT> 10050`

7.2 Docker indisponible après arrêt/redémarrage du lab

Problème Rencontré

Symptôme : Après un *Stop* des instances ou un arrêt automatique du Learner Lab, les conteneurs peuvent ne pas être actifs au redémarrage.

Solution Appliquée

```

1 cd /chemin/vers/le/projet
2 docker compose up -d
3 docker compose ps

```

Listing 7.1 – Relance des conteneurs après redémarrage

Vérifications complémentaires :

- S'assurer que le service Docker est actif : `sudo systemctl status docker`
- Vérifier les logs : `docker compose logs`

7.3 Limitations du Learner Lab

TABLE 7.1 – Contraintes et Bonnes Pratiques AWS Learner Lab

Contrainte	Impact	Bonne Pratique
Région imposée	Certains services peuvent être indisponibles hors us-east-1	Privilégier <code>us-east-1</code> pour tous les déploiements
Types d'instances	Limitation aux types t2/t3	Rester sur <code>t3.medium</code> et <code>t3.large</code>
Budget limité	Environ 50\$ de crédit disponible	Surveiller la consommation et arrêter les instances hors sessions
Arrêt automatique	Les instances s'arrêtent après 4h d'inactivité	Anticiper les redémarrages et relancer Docker Compose
Pas de VPC Peering	Impossibilité de connecter plusieurs VPC	Utiliser un VPC unique avec plusieurs subnets

CHAPITRE 8

Conclusion

Le projet a abouti à la mise en place d'une infrastructure de supervision centralisée sur AWS, capable de monitorer un environnement hybride Linux et Windows via Zabbix.

8.1 Résumé des résultats

Objectifs Atteints

- **Architecture réseau opérationnelle** : VPC + subnet public + IGW + routage configurés et validés
- **Déploiement Zabbix conteneurisé** : Solution reproductible via Docker Compose avec PostgreSQL, Zabbix Server et interface Web
- **Agents Linux et Windows intégrés** : Supervision validée par les dashboards et métriques temps réel
- **Sécurisation appliquée** : Security Groups configurés selon le principe du moindre privilège

8.2 Bénéfices du monitoring hybride

Une supervision unique permet d'unifier les indicateurs, de réduire le temps de diagnostic et d'améliorer la disponibilité globale des services.

Avantages Opérationnels

- **Visibilité centralisée** : Un seul point d'accès pour superviser l'ensemble du parc
- **Détection proactive** : Alertes en temps réel sur les anomalies système
- **Analyse historique** : Graphiques et tendances pour anticiper les besoins
- **Réduction du MTTR** : Diagnostic plus rapide grâce à la centralisation des métriques

8.3 Améliorations futures possibles

TABLE 8.1 – Pistes d'Amélioration et Évolutions

Domaine	Amélioration	Priorité
Sécurité réseau	Segmentation avancée : subnets privés, bastion, VPN site-to-site	Haute
Gestion des secrets	Utilisation d'AWS Secrets Manager ou Systems Manager Parameter Store	Haute
Haute disponibilité	Base de données managée (RDS PostgreSQL), multi-AZ, Load Balancer	Moyenne
Sauvegarde	Snapshots automatiques des volumes EBS, backup base de données	Haute
Alerting	Notifications multi-canal (mail, Teams, Slack, SMS)	Moyenne
Monitoring avancé	Intégration CloudWatch, métriques custom, dashboards consolidés	Basse
Automatisation	Infrastructure as Code (Terraform/CloudFormation)	Moyenne
Scalabilité	Auto Scaling Groups pour les agents, conteneurs orchestrés (ECS/EKS)	Basse

Ce projet démontre la faisabilité et l'efficacité d'une solution de monitoring cloud hybride sous AWS avec Zabbix. Il constitue une base solide pour des déploiements en production.