

# 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>

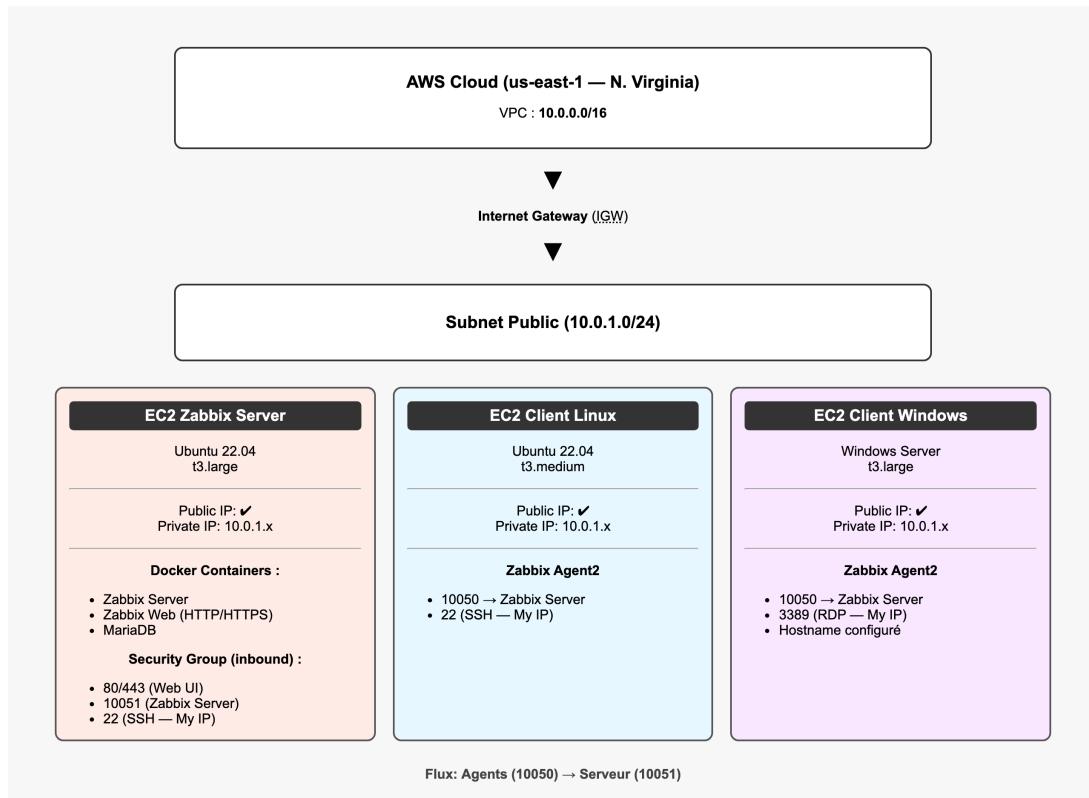


Figure de couverture : aperçu de l'architecture globale du projet.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Contexte . . . . .	1
1.2	Objectifs techniques . . . . .	1
1.3	Technologies utilisées . . . . .	2
<b>2</b>	<b>Architecture réseau</b>	<b>3</b>
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 . . . . .	6
2.2.1	Principe de filtrage . . . . .	6
2.2.2	Captures des Security Groups . . . . .	7
2.3	Schéma réseau . . . . .	8
<b>3</b>	<b>Instances EC2</b>	<b>9</b>
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) . . . . .	12
<b>4</b>	<b>Déploiement du serveur Zabbix (Docker)</b>	<b>14</b>
4.1	Installation Docker . . . . .	14
4.2	Installation Docker Compose . . . . .	15
4.3	Fichier docker-compose.yml expliqué . . . . .	15
4.4	Lancement des conteneurs . . . . .	17
4.5	Vérification de l'interface Web . . . . .	17
<b>5</b>	<b>Configuration des agents</b>	<b>18</b>
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 . . . . .	19
5.2.1	Installation MSI et paramètres . . . . .	19
<b>6</b>	<b>Monitoring &amp; Dashboard</b>	<b>21</b>
6.1	Ajout des hôtes . . . . .	21
6.2	Validation : statut ZBX en vert . . . . .	21
6.3	Graphiques CPU/RAM et tableaux de bord . . . . .	22
<b>7</b>	<b>Difficultés rencontrées &amp; solutions</b>	<b>25</b>
7.1	Ports bloqués / accès réseau . . . . .	25
7.2	Docker indisponible après arrêt/redémarrage du lab . . . . .	25
7.3	Limitations du Learner Lab . . . . .	25
<b>8</b>	<b>Conclusion</b>	<b>26</b>
8.1	Résumé des résultats . . . . .	26
8.2	Bénéfices du monitoring hybride . . . . .	26
8.3	Améliorations futures possibles . . . . .	26

# 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). . . . .	11
3.3	Lancement de l'instance Windows Server (t3.large). . . . .	11
3.4	Vue globale des 3 instances EC2 (serveur Zabbix + clients Linux/Windows). .	12
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). . . . .	13
4.1	Création du fichier <code>docker-compose.yml</code> sur le serveur Zabbix. . . . .	16
4.2	Téléchargement/initialisation des images Docker (premier lancement des services). . . . .	17
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. . . . .	20
6.1	Ajout d'un hôte dans Zabbix (exemple : client Linux). . . . .	21
6.2	Validation de la connectivité : statut ZBX en vert pour Linux et Windows. .	22
6.3	Monitoring : consultation des dernières données (Latest data). . . . .	22
6.4	Exemple de graphique CPU/RAM (preuve de collecte effective). . . . .	23

6.5	Tableau de bord Zabbix : vue synthétique de supervision. . . . .	23
6.6	Dashboard final : état global et indicateurs principaux. . . . .	24

# Liste des tableaux

2.1	Ports autorisés par les Security Groups (synthèse)	6
3.1	Récapitulatif des instances EC2 du projet	9

# 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

- **AWS (Learner Lab)** : VPC, Subnet, Internet Gateway, Route Table, Security Groups, EC2.
- **Docker & Docker Compose** : déploiement reproductible 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

- **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

The screenshot shows the 'Create VPC' wizard in the AWS Management Console. On the left, under 'VPC settings', the 'Resources to create' section is set to 'VPC and more'. The 'Name tag auto-generation' section has 'Auto-generate' checked and the tag value 'MOHAMED\_AASSOU\_VPC'. The 'IPv4 CIDR block' is set to '10.0.0.0/16'. Under 'IPv6 CIDR block', 'No IPv6 CIDR block' is selected. The 'Tenancy' dropdown is set to 'Default'. On the right, a 'Preview' section shows a summary: 'VPC Show details' (Your AWS virtual network) and 'Subnet Subnets us-east MOH...'.

FIGURE 2.1 – Crédit de la capture du VPC (CIDR 10.0.0.0/16).

The screenshot shows the 'Create subnet' wizard in the AWS Management Console. Under 'Subnet settings', it specifies the CIDR blocks and Availability Zone for the subnet. The 'Subnet 1 of 1' section includes a 'Subnet name' field with 'MOHAMED\_AASSOU\_SUBNET', an 'Availability Zone' dropdown set to 'United States (N. Virginia) / use1-az6 (us-east-1a)', and an 'IPv4 VPC CIDR block' dropdown set to '10.0.0.0/16'. The 'IPv4 subnet CIDR block' is set to '10.0.0.0/20'. In the 'Tags - optional' section, there is a single tag 'Name: MOHAMED\_AASSOU\_SUBNET'. The top bar shows the AWS logo, search bar, and account information: Account ID: 2848-9773-8568, voclabs/user4543225=Mohamed\_Aassou.

FIGURE 2.2 – Crédit de la capture du subnet public (10.0.1.0/24).

The screenshot shows the AWS VPC Subnets page. The left sidebar lists various VPC-related services. The main area displays a table for 'Subnets (1/1)'. The subnet listed is 'MOHAMED\_AASSOU\_VPC-subnet-public1-us-east-1a' with the ID 'subnet-0ed7b5aadf0564467'. It is marked as 'Available' and associated with the VPC 'vpc-076b179691201da01'. Below the table, a detailed view for the subnet shows its ARN, state (Available), and other parameters like IPv4 CIDR (10.0.0.0/20) and available IP addresses (4091).

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

The screenshot shows the AWS Route Tables page. The left sidebar lists various VPC-related services. The main area displays a table for 'Route tables (1/1)'. The route table listed is 'MOHAMED\_AASSOU\_VPC-rtb-public' with the ID 'rtb-0d063384ca32daa25'. It has an explicit subnet association to 'subnet-0ed7b5aadf0564467'. Below the table, a detailed view for the route table shows its ID, main status (No), owner ID (284897738568), and edge associations.

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

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 creation interface. The security group name is "Mohamed\_SG-Zabbix-Server". The description is "allow zabbix server". The VPC is set to "vpc-0886a96bf50fca73b". The Inbound rules section contains the following entries:

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Anyw...	HTTP-80-Zabbix
HTTPS	TCP	443	Anyw...	HTTPS-443-Zabbix
Custom TCP	TCP	10050	Custom	
Custom TCP	TCP	10051	Custom	
SSH	TCP	22	My IP	SSH-admin-access

**Add rule** button is present at the bottom left.

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

The screenshot shows the AWS EC2 Security Groups edit inbound rules interface for an instance named "sg-0a6ccf7c1292ce1ba - SG-Linux-Client". The Inbound rules section contains the following entries:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0ff2484d4d7401598	Custom TCP	TCP	10050	Custom	Zabbix-agent
sgr-0cb232bb07e655027	SSH	TCP	22	Custom	SSH-admin-access

**Add rule**, **Cancel**, **Preview changes**, and **Save rules** buttons are at the bottom.

FIGURE 2.6 – Security Group du client Linux (SSH + 10050 depuis le 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-0dfb8dd4db33bb680	Custom TCP	TCP	10050	Custom	Zabbix-agent
sgr-07b06e4e668a5e949	RDP	TCP	3389	Custom	RDP-admin-access

**Add rule** Info

**Cancel** **Preview changes** **Save rules**

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

## 2.3 Schéma réseau

**Create VPC** Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances. Mouse over a resource to highlight the related resources.

**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 auto-generation** Info  
Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.

Auto-generate  
MOHAMED\_AASSOU\_VPC

**IPv4 CIDR block** Info  
Determine the starting IP and the size of your VPC using CIDR notation.

10.0.0.0/16 65,536 IPs

CIDR block size must be between /16 and /28.

**IPv6 CIDR block** Info  
 No IPv6 CIDR block  Amazon-provided IPv6 CIDR block

**Tenancy** Info  
Default

**Encryption settings - optional**

**Preview**

**VPC** Show details  
Your AWS virtual network  
MOHAMED\_AASSOU\_VPC-vpc

**Subnet** Subnets vpc-00000000000000000000000000000000

**us-east** MOHAMED\_AASSOU\_VPC-vpc

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	CPU (vCPU)	RAM (GiB)	Fonction
Serveur Zabbix	t3.large	Ubuntu 22.04	2	8	Héberge Docke bix Server + W
Client Linux	t3.medium	Ubuntu 22.04	2	4	Agent Zabbix triques systèm
Client Windows	t3.large	Windows Server	2	8	Agent Zabbix triques Windo gourmand)

### 3.2 Justification des choix t3.medium / t3.large

- **Serveur Zabbix (t3.large)** : la conteneurisation (serveur + web + base) nécessite une marge de CPU/RAM pour éviter les lenteurs.
- **Client Linux (t3.medium)** : l'agent Zabbix est léger et les charges de test restent modérées.
- **Client Windows (t3.large)** : Windows Server consomme davantage de ressources ; ce type améliore la fluidité (RDP et services).

### 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.

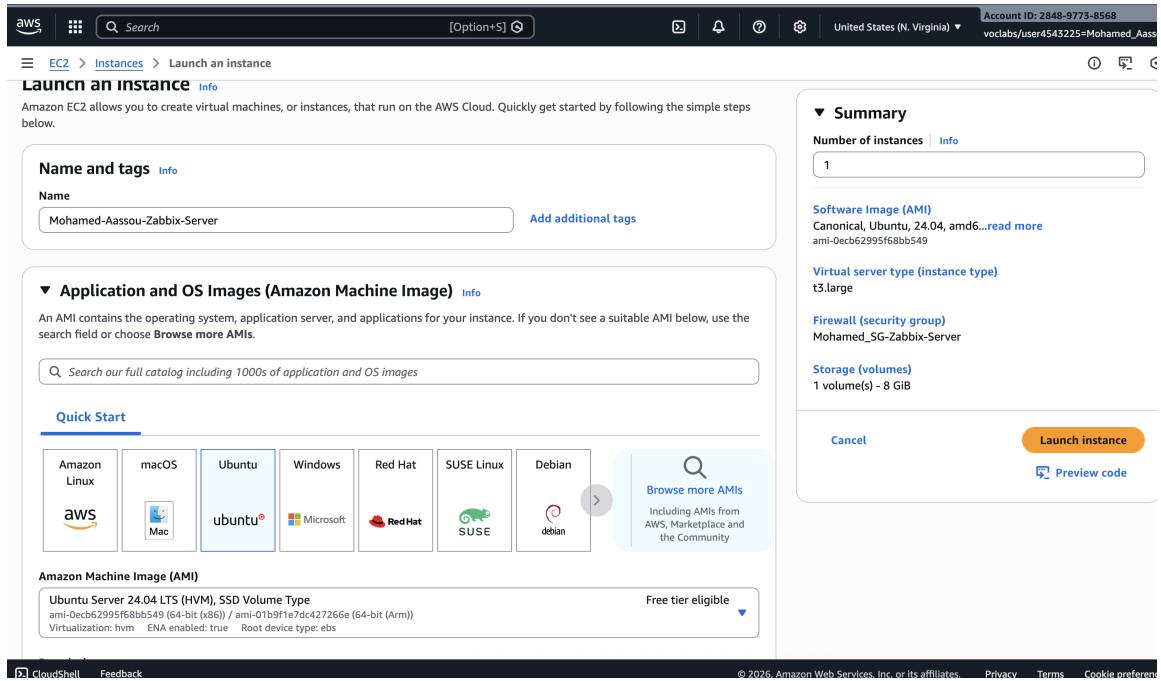


FIGURE 3.1 – Lancement de l'instance Zabbix Server (Ubuntu 22.04, t3.large).

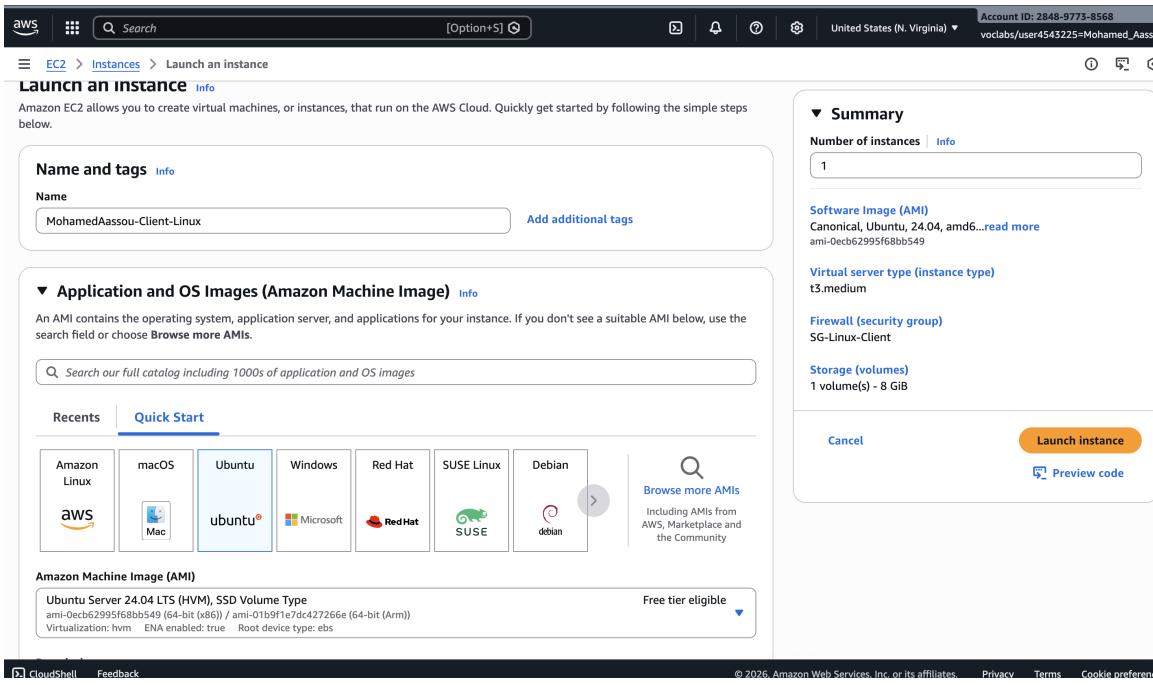


FIGURE 3.2 – Lancement de l’instance client Linux (Ubuntu 22.04, t3.medium).

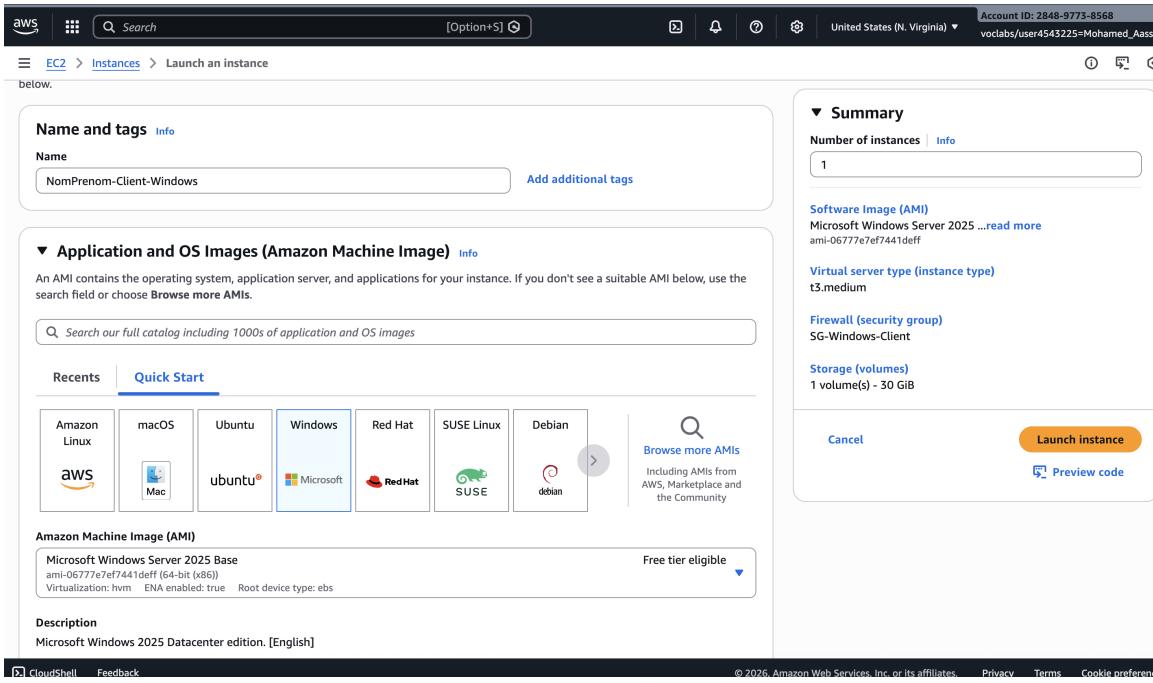


FIGURE 3.3 – Lancement de l’instance Windows Server (t3.large).

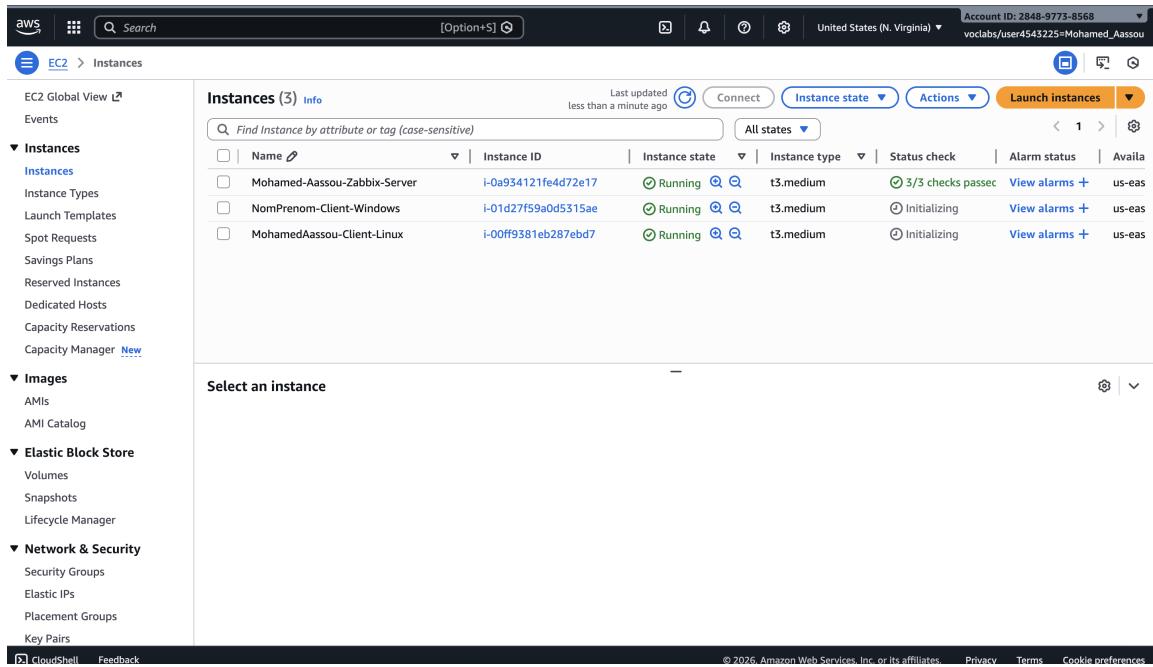


FIGURE 3.4 – Vue globale des 3 instances EC2 (serveur Zabbix + clients Linux/Windows).

## 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*.

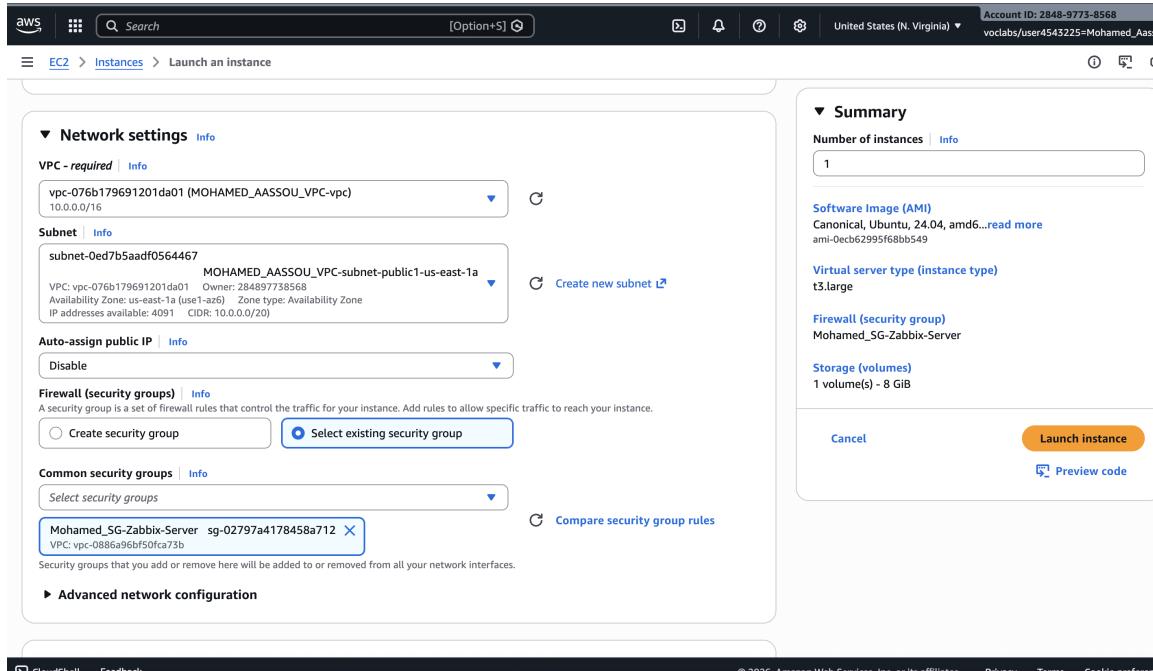


FIGURE 3.5 – Configuration réseau : vérification/édition réseau de l'instance Zabbix Server (VPC/Subnet).

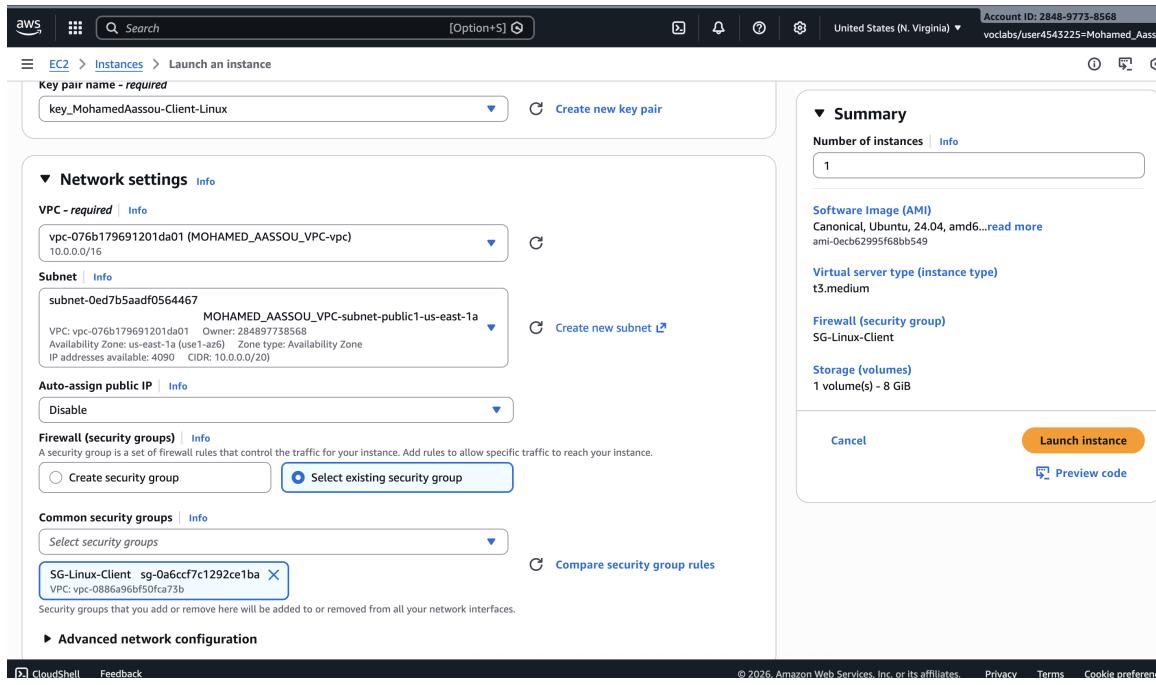


FIGURE 3.6 – Configuration réseau : vérification/édition réseau du client Linux (VPC/-Subnet).

# 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.

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

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.
gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo
$VERSION_CODENAME) stable" \
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
sudo systemctl enable --now docker
```

## 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).

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  
      POSTGRES_DB: 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:

```

```

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
      POSTGRES_DB: zabbix
      POSTGRES_USER: zabbix
      POSTGRES_PASSWORD: zabbixpass
    ports:
      - "10851:10851"
    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 – Crédit de la création du fichier docker-compose.yml sur le serveur Zabbix.

```
Last login: Thu Jan  1 11:50:21 2026 from 41.140.132.223
ubuntu@ip-10-0-4-140:~$ nano docker-compose.yml
ubuntu@ip-10-0-4-140:~$ ls
docker-compose.yml
ubuntu@ip-10-0-4-140:~$ docker compose up -d
[*] up 35/36
✓ Image postgres:16-alpine          Pulled
✓ Image zabbix/zabbix-server-pgsql:alpine-7.0-latest Pulled
✓ Image zabbix/zabbix-web-nginx-pgsql:alpine-7.0-latest Pulled
✓ Network ubuntu_default             Created
✓ Container ubuntu_pgdata            Created
✓ Container zabbix-postgres         Created
✓ Container zabbix-server           Created
✓ Container zabbix-web              Created
ubuntu@ip-10-0-4-140:~$
```

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

## 4.4 Lancement des conteneurs

Listing 4.3 – Démarrage des conteneurs Zabbix

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

## 4.5 Vérification de l’interface Web

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

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*,
- et le port 80 est bien autorisé dans le Security Group du serveur.

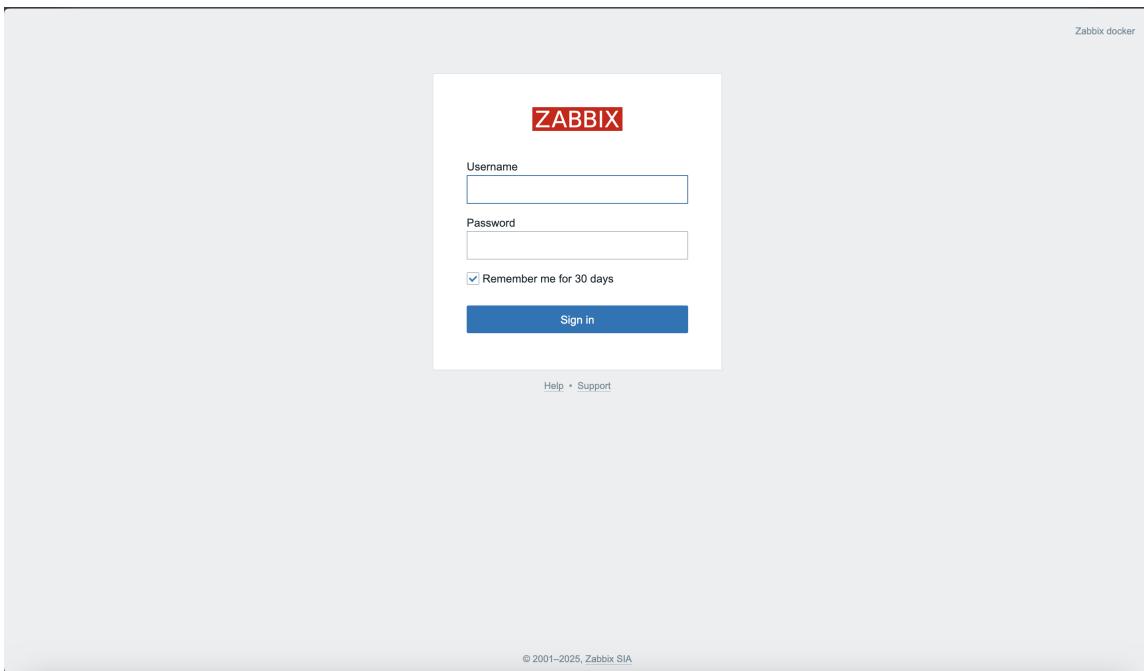


FIGURE 4.3 – Vérification : accès à l’interface Web Zabbix.

# 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

Listing 5.1 – Installation de l’agent Zabbix sur Ubuntu

```
sudo apt update
sudo apt install -y zabbix-agent2
sudo systemctl enable --now zabbix-agent2
sudo systemctl status zabbix-agent2 --no-pager
```

#### 5.1.2 Configuration : zabbix\_agent2.conf (extrait)

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

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

```
Server=10.0.1.<IP_PRIVÉE_ZABBIX_SERVER>
ServerActive=10.0.1.<IP_PRIVÉE_ZABBIX_SERVER>
Hostname=MohamedAASSOU-Client-Linux
```

```

# Mandatory: no
# Range: 1924-32767
# Default:
# StatusPort

##### 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.
# Multiple Zabbix servers/proxies 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.
# Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.
# If port is not specified, square brackets for IPv6 addresses are optional.
# If port is not specified, square brackets for IPv6 addresses are optional.
# If this parameter is not specified, active checks are disabled.
# Example for multiple servers:
#   ServerActive=zabbix.domain[::]:10051,zabbix.domain[::]:10051,zabbix.domain[::]:10051
# Example for high availability with two clusters and one server:
#   ServerActive=zabbix.cluster.node1;zabbix.cluster.node2;zabbix.cluster.node3
#   ServerActive=zabbix.cluster.node1;zabbix.cluster.node2;zabbix.cluster.node3;zabbix.domain
# Mandatory: no
# Default:
# ServerActive

ServerActive=10.0.4.140

### 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.
# HostMetadata 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%
Swap usage: 0% Users logged in: 0
IPv4 address for ens5: 10.0.11.27

expanded 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 :

- **Server** : IP privée du serveur Zabbix (dans le VPC).

— **Hostname** : nom strictement identique à celui déclaré dans l'interface Zabbix.

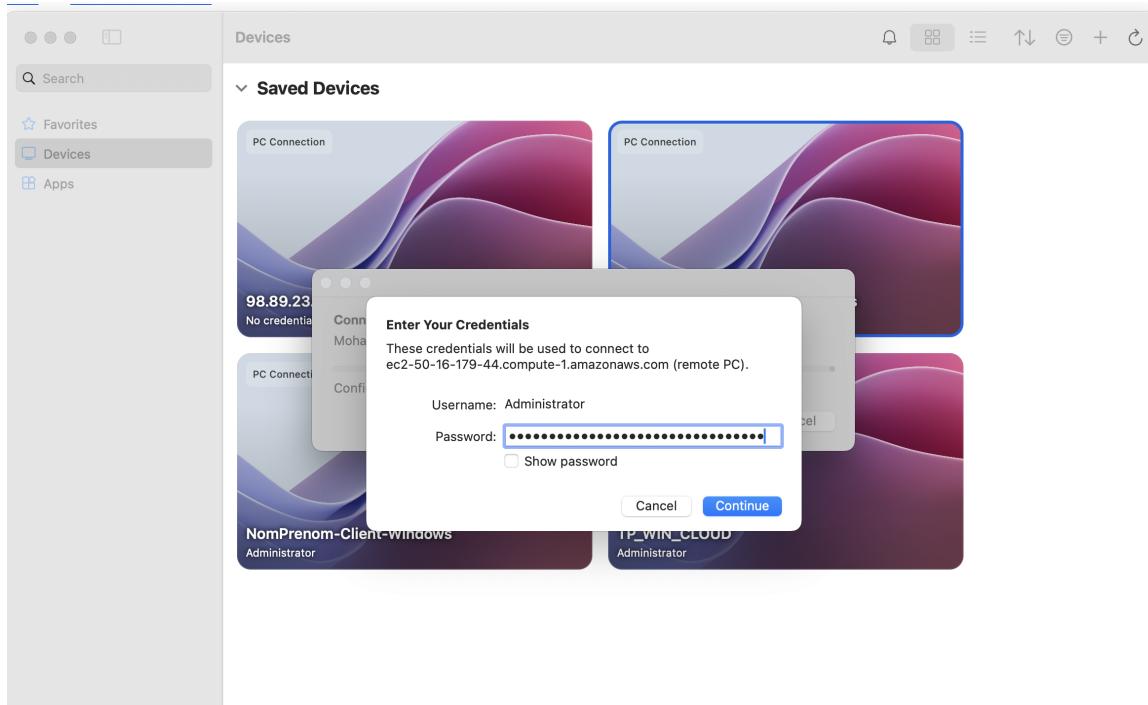


FIGURE 5.3 – Connexion RDP au serveur Windows pour finaliser l'installation de l'agent.

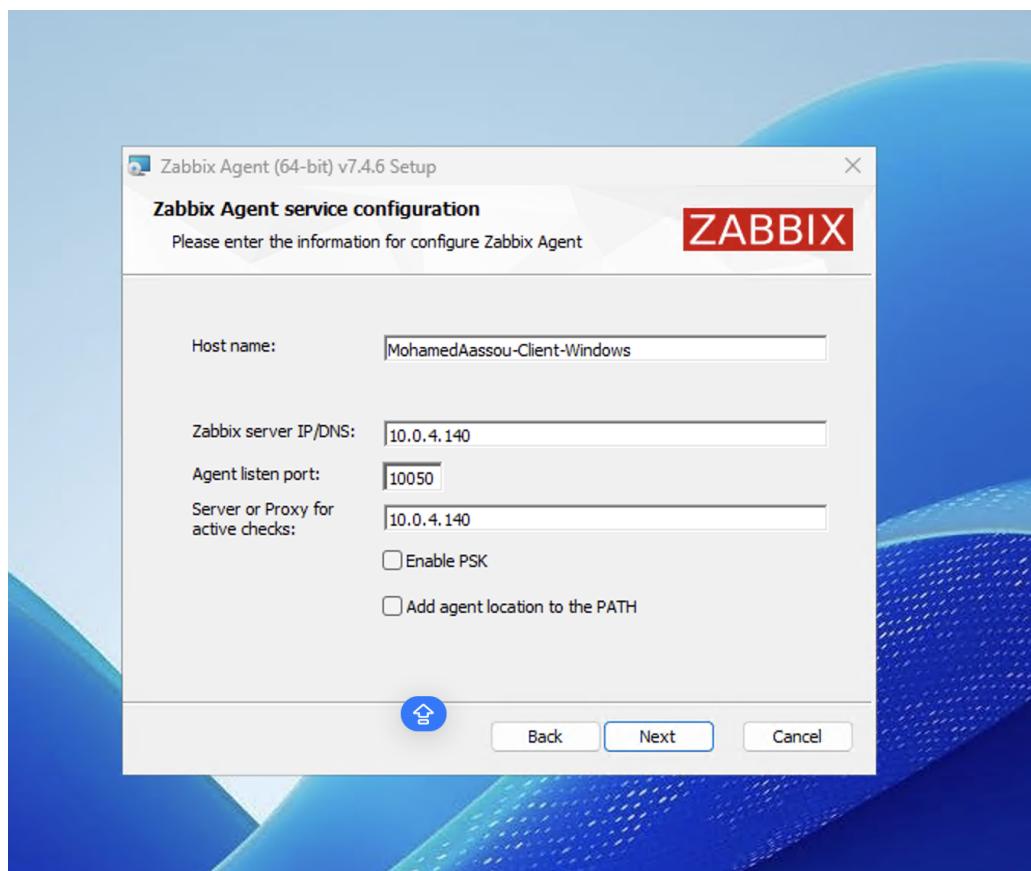


FIGURE 5.4 – Installation et configuration de l'agent Zabbix sur Windows Server.

# 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).

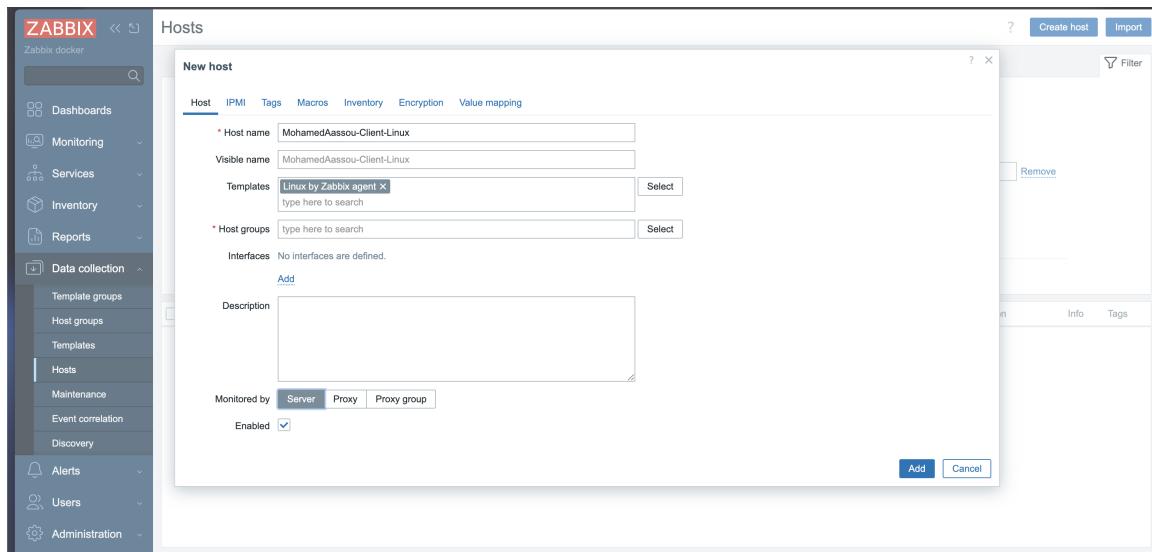


FIGURE 6.1 – Ajout d'un hôte dans Zabbix (exemple : client Linux).

### 6.2 Validation : statut ZBX en vert

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

Zabbix 7.0.22, © 2001–2025, Zabbix SIA

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.

Host	Name	Last check	Last value	Change	Tags	Info
MohamedAassou-Cl...	Available memory	14s	3.31 GB	-216 KB	component: memory	Graph
MohamedAassou-Cl...	Available memory in %	13s	88.2334 %	-0.005497 %	component: memory	Graph
MohamedAassou-Cl...	Checksum of /etc/passwd	34m 53s	0dc138ff32e87...		component: security	History
MohamedAassou-Cl...	Context switches per second	57s	84.4296	-1.5529	component: cpu	Graph
MohamedAassou-Cl...	CPU guest nice time	55s	0 %		component: cpu	Graph
MohamedAassou-Cl...	CPU guest time	56s	0 %		component: cpu	Graph
MohamedAassou-Cl...	CPU idle time	54s	99.8916 %	-0.000009 %	component: cpu	Graph
MohamedAassou-Cl...	CPU interrupt time	53s	0 %		component: cpu	Graph

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

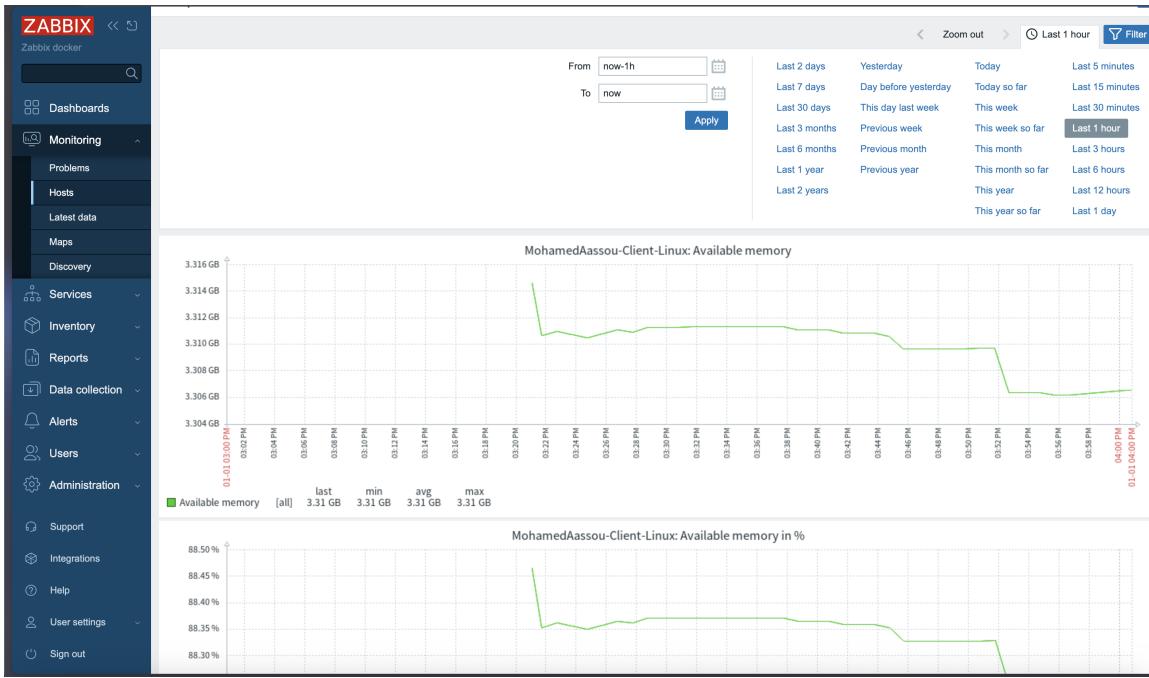


FIGURE 6.4 – Exemple de graphique CPU/RAM (preuve de collecte effective).

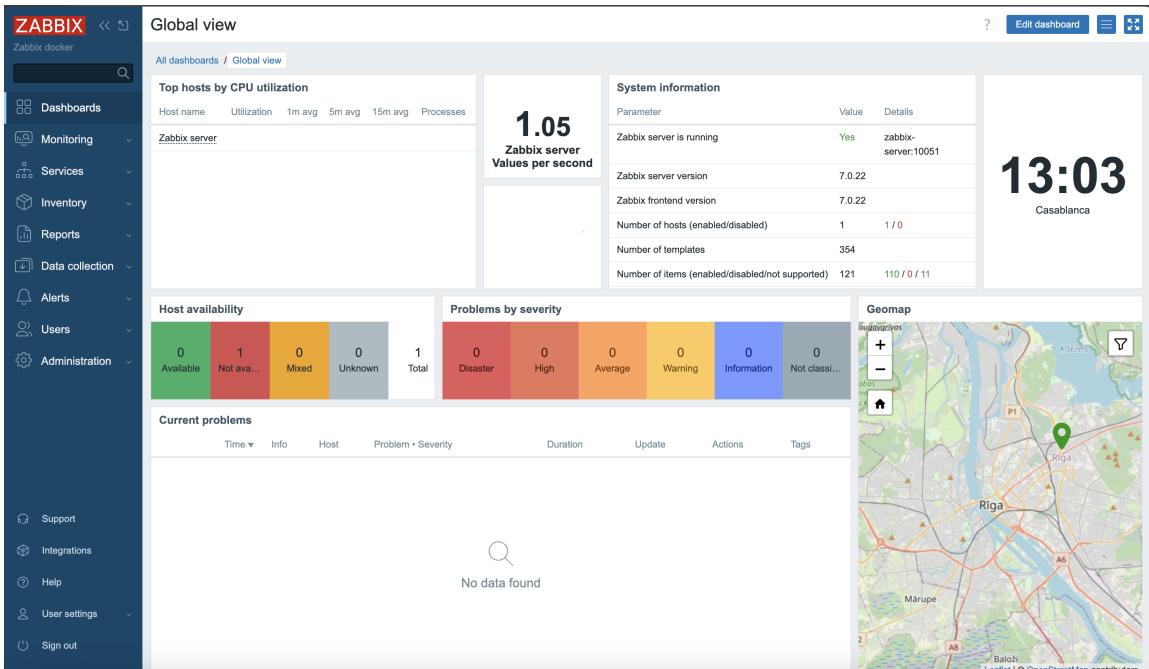


FIGURE 6.5 – Tableau de bord Zabbix : vue synthétique de supervision.

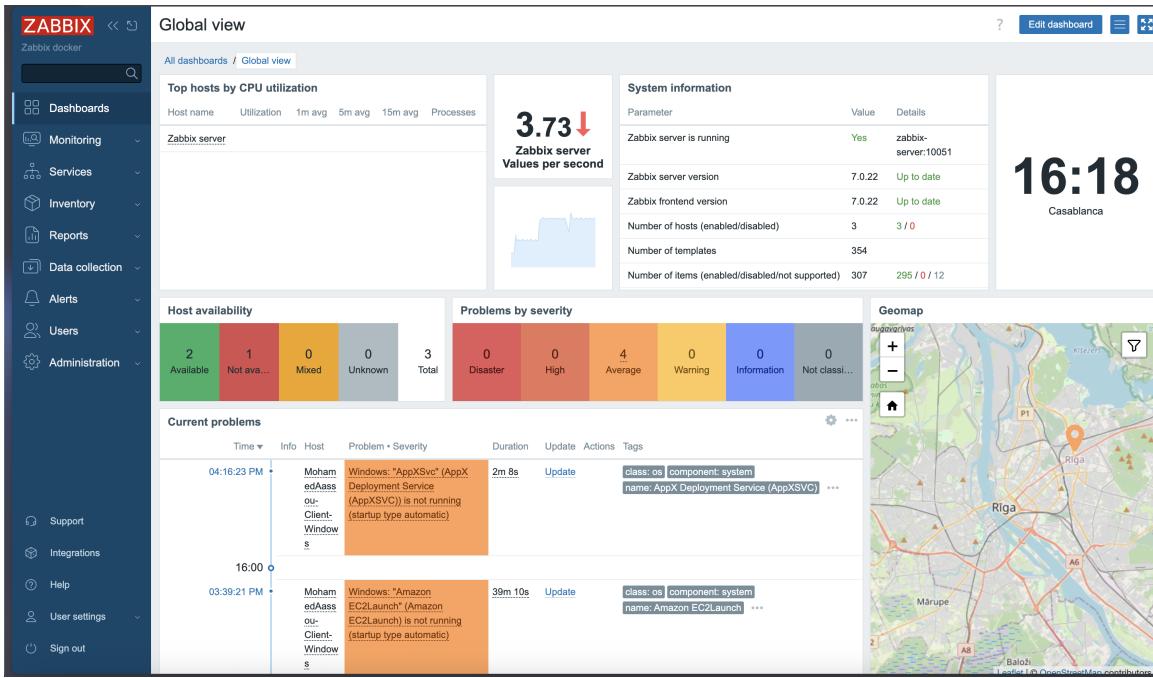


FIGURE 6.6 – Dashboard final : état global et indicateurs principaux.

# Chapitre 7

## Difficultés rencontrées & solutions

### 7.1 Ports bloqués / accès réseau

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

**Solution :** vérifier les règles des Security Groups, limiter les sources (My IP pour SSH/RDP/Web) et autoriser 10050 depuis le serveur Zabbix vers les clients.

### 7.2 Docker indisponible après arrêt/redémarrage du lab

**Problème :** après un *Stop* des instances ou un arrêt automatique, les conteneurs peuvent ne pas être actifs.

**Solution :**

Listing 7.1 – Relance des conteneurs après redémarrage

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

### 7.3 Limitations du Learner Lab

- **Région :** privilégier us-east-1 (N. Virginia).
- **Types d'instances :** rester sur t3.medium et t3.large.
- **Budget :** surveiller la consommation (environ 50\$) et arrêter les instances hors sessions.
- **Arrêt automatique :** anticiper les redémarrages et relancer Docker Compose.

# Chapitre 8

## Conclusion

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

### 8.1 Résumé des résultats

- Architecture réseau opérationnelle (VPC + subnet public + IGW + routage).
- Déploiement Zabbix conteneurisé reproductible via Docker Compose.
- Agents Linux et Windows intégrés, supervision validée par les dashboards et métriques.

### 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.

### 8.3 Améliorations futures possibles

- Segmentation plus avancée : subnets privés, bastion, VPN.
- Gestion sécurisée des secrets (SSM / Secrets Manager).
- Haute disponibilité (base managée, sauvegardes, redondance).
- Notifications (mail, Teams, Slack) et règles d'alerting plus fines.