BTS SIO

DOCUMENTATION TECHNIQUE FICHE n°1

Épreuve E6

Alexandre BONGRAND
07/05/2025

Suivi des modifications

Version	*Référence*	*Auteur*	*Date*	*Commentaires*
A	Documentation technique fiche n°1	Alexandre	25/04/2025	Création

Objet :

Documentation technique fiche n°1 - Projet E6

Diffusion :

BTS SIO – Étudiants BTS SIO.

Développement :

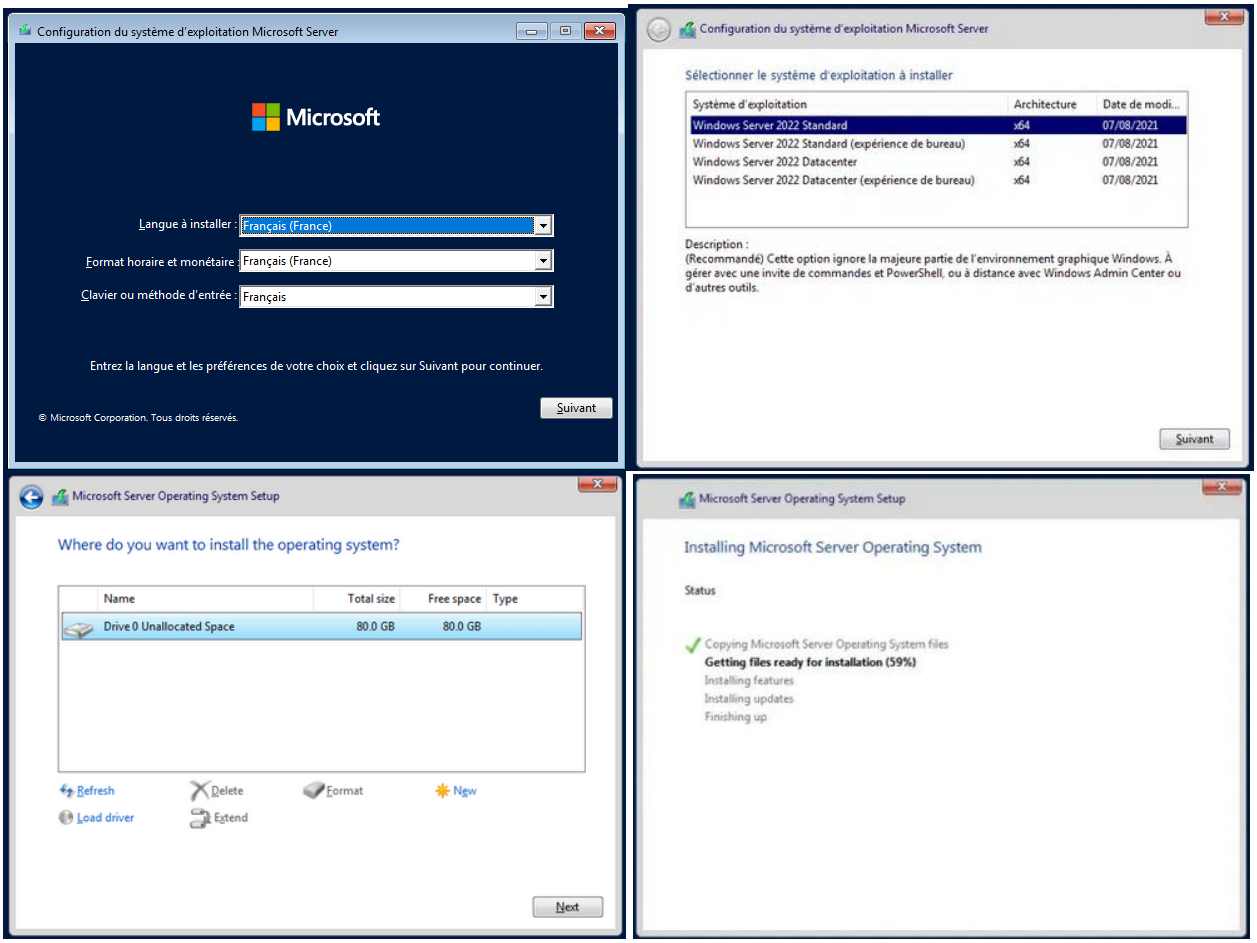
Table des matières

WINDOWS SERVEUR

Dans un premier temps, j'ai installé Windows Server 2022 afin de mettre en place un contrôleur de domaine (AD DS), un serveur DNS, ainsi qu'un partage de fichiers.

Ces services permettront aux utilisateurs de l'entreprise BONGRAND de travailler de manière collaborative.

L'installation de Windows Server 2022 s'effectue de la même manière que celle d'un poste client sous Windows 10.

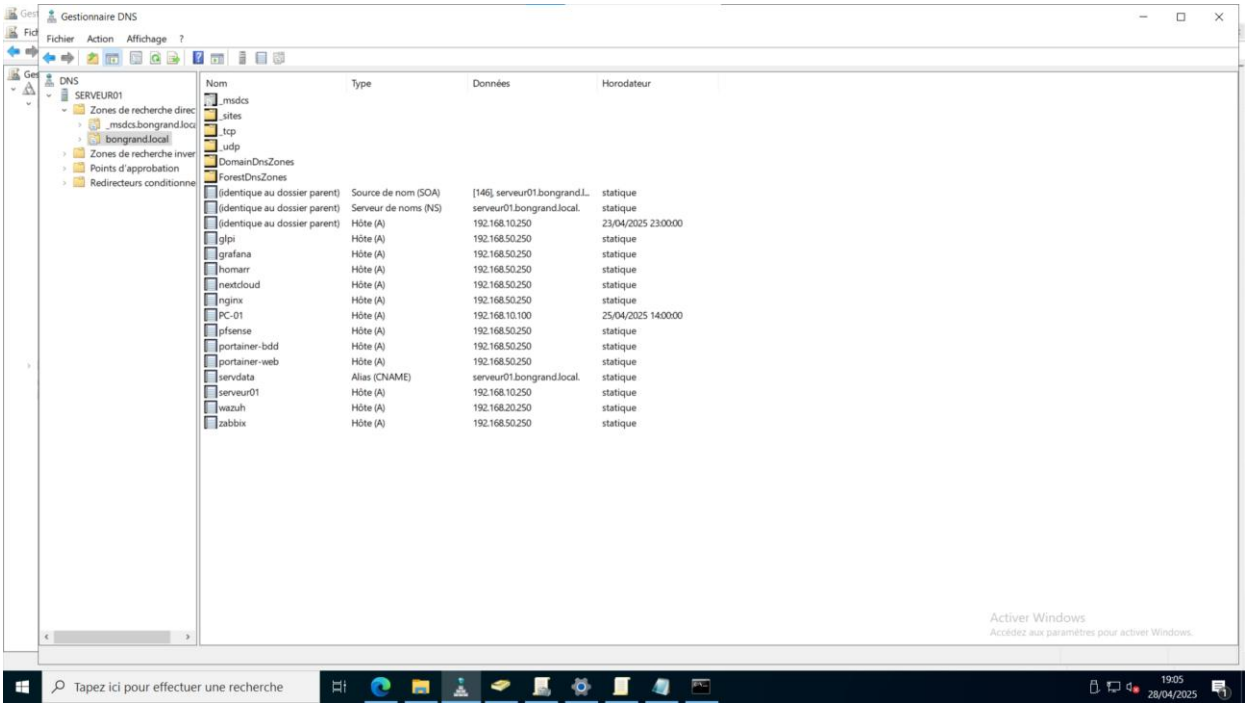


Une fois le système installé, nous pouvons procéder à l'installation des services nécessaires pour répondre aux besoins de l'entreprise BONGRAND.

Installation AD DS & DNS

Pour installer l'Active Directory, nous allons dans fonctionnalité de notre serveur et ajout d'une fonctionnalité.

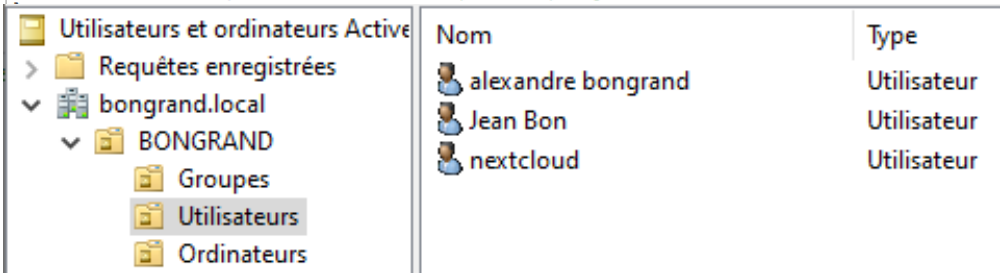
Pour installer un AD, certain prérequis sont obligatoire, comme une adresse IP statique et installer le service DNS (Domaine Name Service).



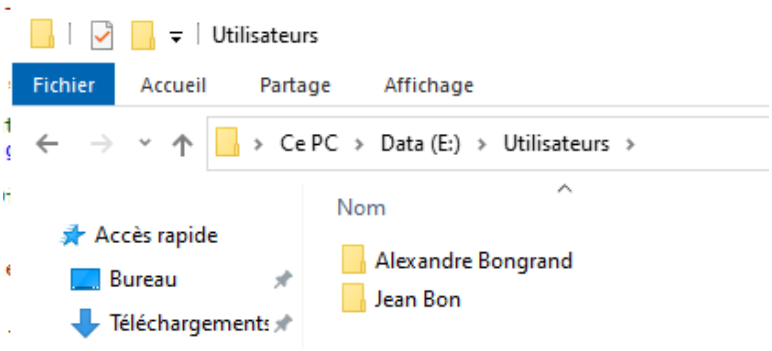
En installant l'AD, nous créerons un domaine (une nouvelle forêt), notre domaine s'appellera bongrand.local

Une fois notre domaine créé, nous pouvons ajouter nos utilisateurs. Pour ajouter les utilisateurs à l'Active Directory, j'ai d'abord fait un excel avec tous les utilisateurs de l'organigramme.
J'ai enregistré cet Excel en « .csv (point-virgule) », ce qui va nous permettre de faire un script d'ajout d'utilisateur.
Grâce à mon fichier.csv, j'ai pu créer un script powershell, qui va ajouter chaque utilisateur dans l'AD dans un groupe que j'ai nommé « Utilisateurs ».

```powershell
# Chemin vers le fichier CSV contenant les informations des utilisateurs
$cheminCSV = "C:\Users\Administrateur\Desktop\user.csv"

# Lire les utilisateurs à partir du fichier CSV
$utilisateurs = Import-Csv -Path $cheminCSV

foreach ($utilisateur in $utilisateurs) {
    # Construire le nom d'utilisateur (login) au format "prenom.p"
    $nomUtilisateur = ($utilisateur.Prenom + "." + $utilisateur.Nom.Substring(0,1)).ToLower()

    # Construction du nom complet
    $nomComplet = $utilisateur.Prenom + " " + $utilisateur.Nom

    # Définir l'OU (Unité d'Organisation) cible où les utilisateurs seront créés
    $ou = "OU=Utilisateurs,DC=lanlrt,DC=monEpicerie,DC=eu"

    # Chemin d'accès à l'AD
    $cheminAD = "CN=" + $nomComplet + "," + $ou

    # Définir un mot de passe par défaut (peut être modifié selon vos besoins)
    $motDePasse = ConvertTo-SecureString "Kirikou202409!" -AsPlainText -Force

    # Créer l'utilisateur dans Active Directory
    New-ADUser `
        -SamAccountName $nomUtilisateur `
        -UserPrincipalName $nomUtilisateur `
        -Name $nomComplet `
        -DisplayName $nomComplet `
        -GivenName $utilisateur.Prenom `
        -Surname $utilisateur.Nom `
        -Path $ou `
        -AccountPassword $motDePasse `
        -Enabled $true `
        -PasswordNeverExpires $true ` # Le mot de passe n'expire jamais
```

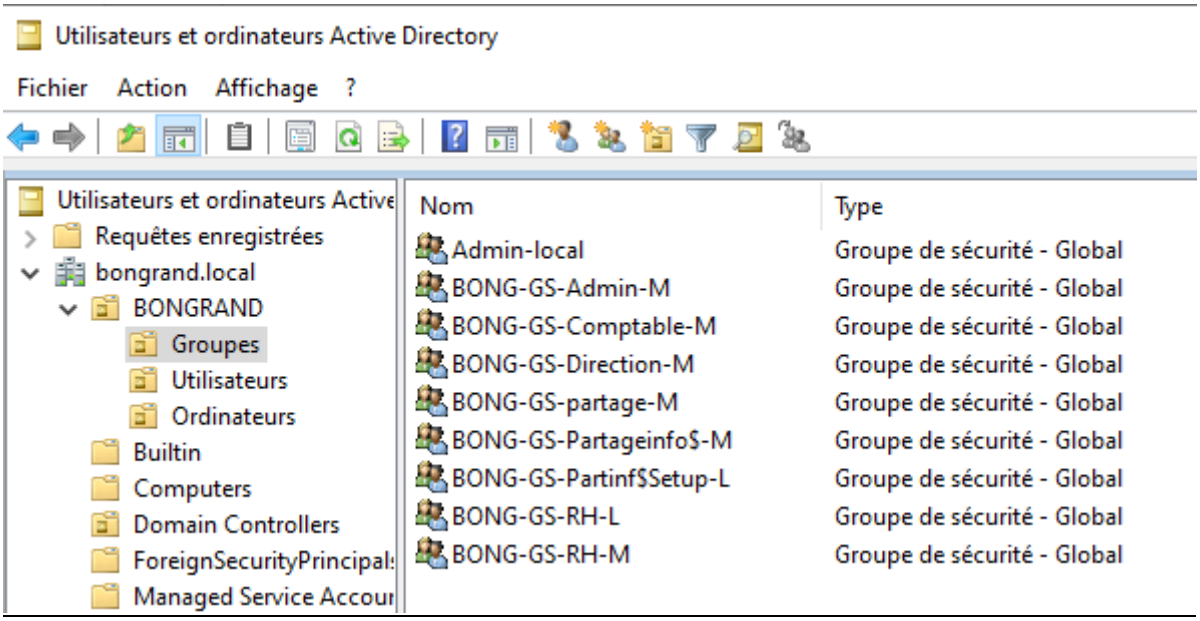


Une fois les utilisateurs crées, j'ai crée un dossier personnel pour chaque utilisateurs.



Pour les ordinateurs, lorsque l'on va ajouter un ordinateur à notre domaine, il va automatiquement s'ajouter à computers, je les déplace dans le groupe " Ordinateurs" que j'ai créé afin de pouvoir lier des GPOs à ce groupe.

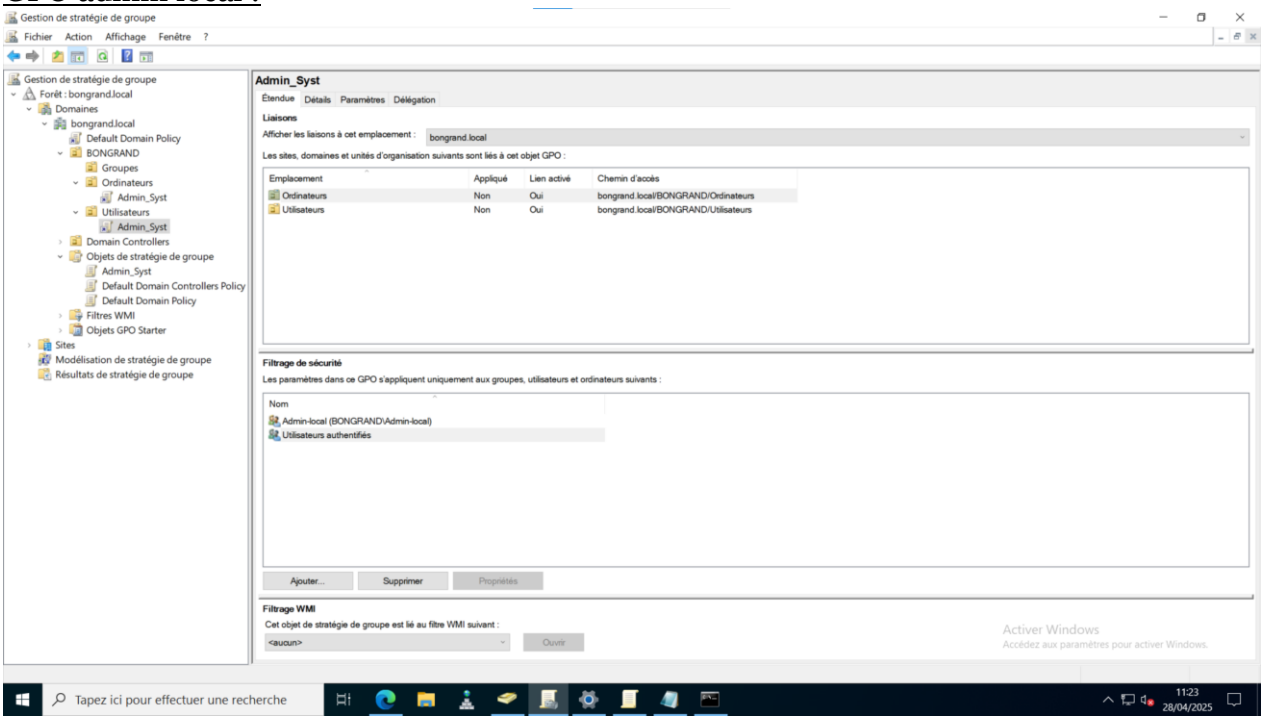
<u>Groupes pour les droits :</u>

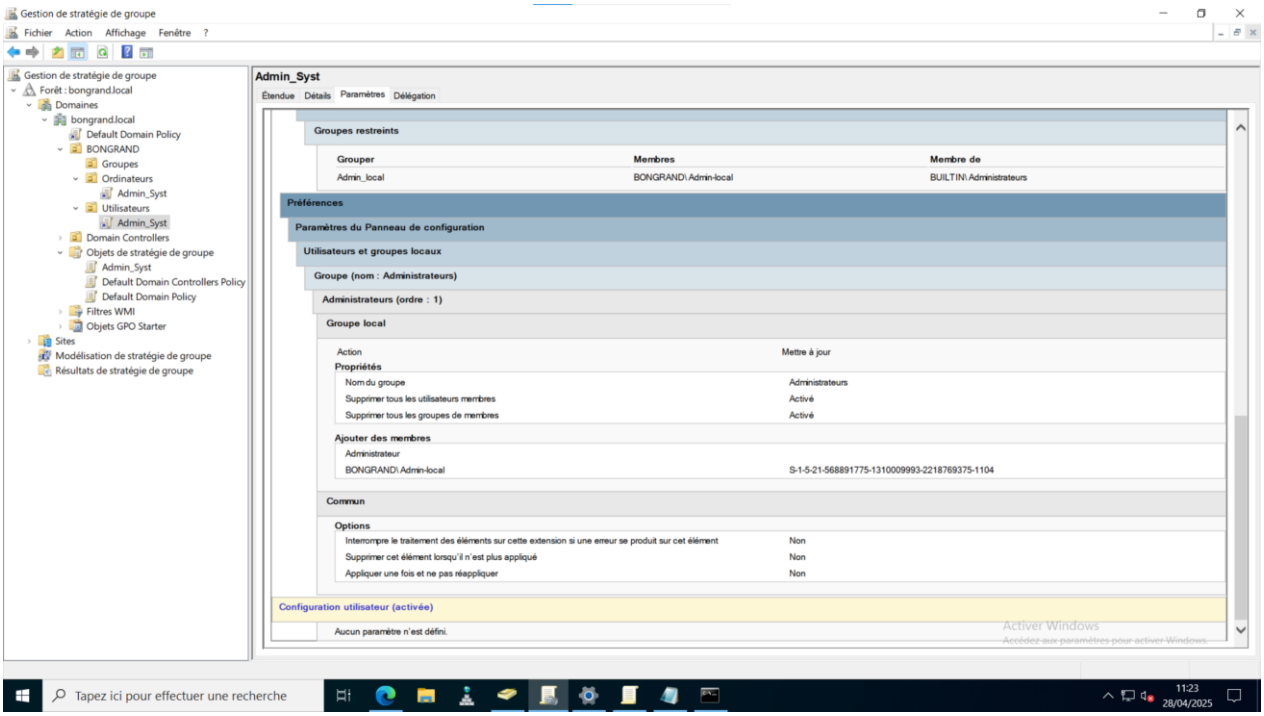


Mise en place des GPOs

J'ai créé des GPO afin de rendre l'utilisateur "Alexandre Bongrand" administrateur des postes qu'il utilise. Cela a pour but de lui permettre d'effectuer des manipulations d'administration sur les postes (installation de logiciels, paramétrage de Windows, etc.).

GPO admin local :

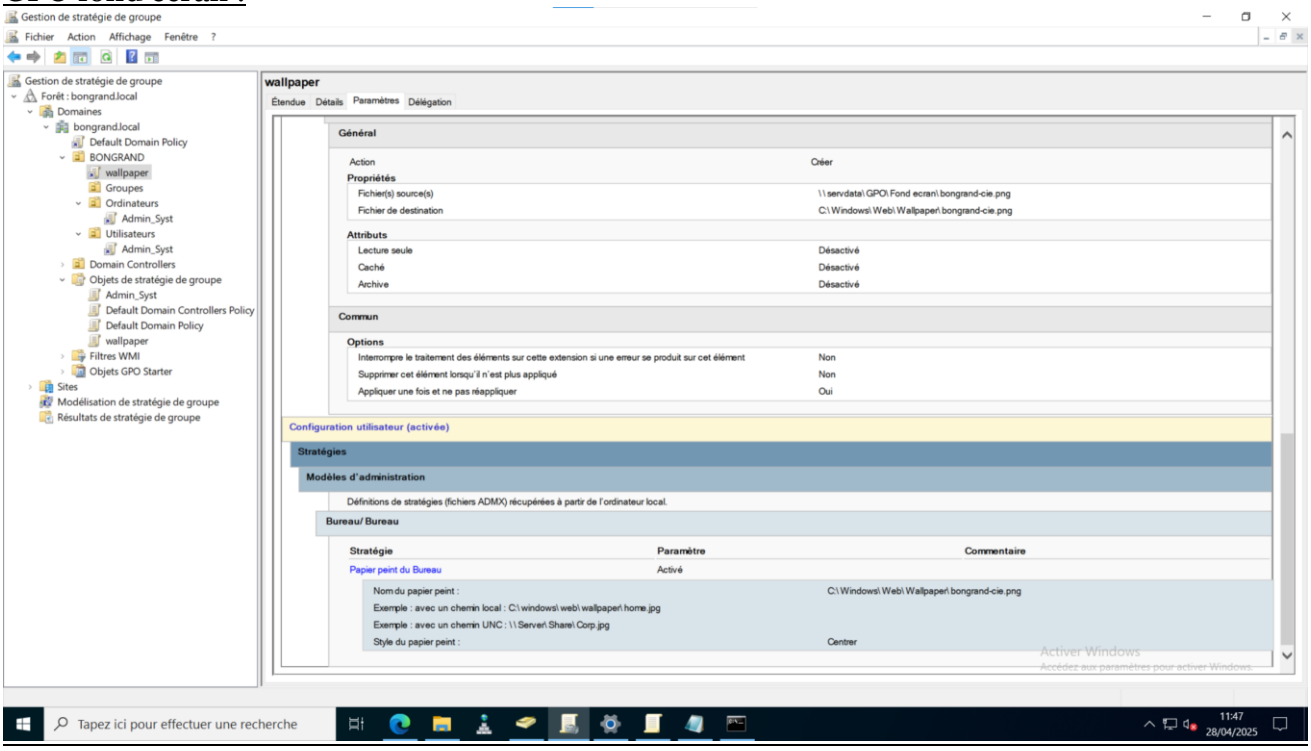




Afin d'uniformiser l'ensemble des postes de l'entreprise BONGRAND, j'ai mis en place une GPO imposant un fond d'écran identique pour tous les utilisateurs.

Le fond d'écran est téléchargé depuis le dossier partagé "GPO" du serveur, puis il est déployé directement sur le poste.

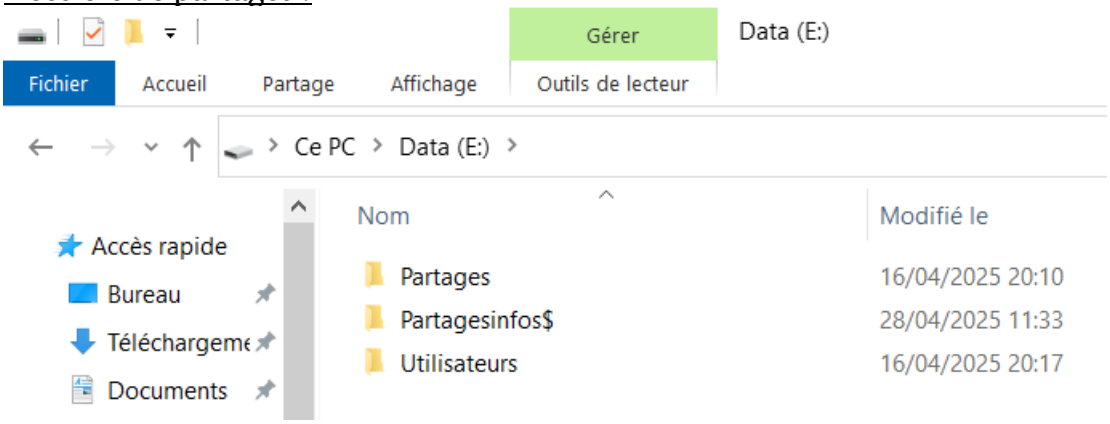
GPO fond écran :



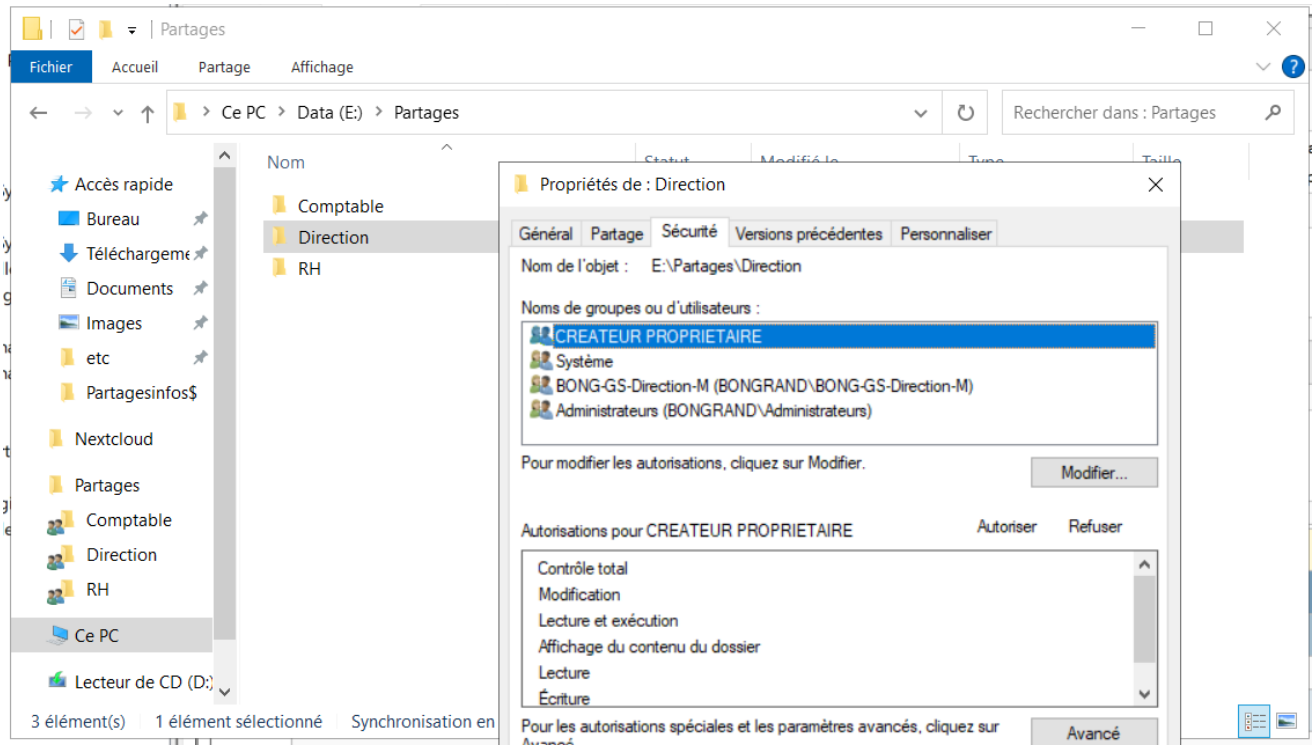
Partage de fichiers

Afin de favoriser le travail collaboratif, j'ai mis en place un partage de fichiers sous Windows. Cela permet aux salariés de partager des documents, de travailler sur des dossiers communs et de gérer les droits d'accès (ACL) pour attribuer à chaque utilisateur uniquement les permissions nécessaires.

Dossiers de partages :



Droits sur un dossier :



Pour la gestion des ACL, seuls des groupes sont utilisés afin d'éviter de mélanger utilisateurs et groupes sur un même dossier. La nomenclature des groupes suit la structure suivante : "BONG" pour le domaine, "GS" pour Global Security, "Direction" pour le nom du dossier, et "M" pour le niveau de permission (Modification).

PFSENSE

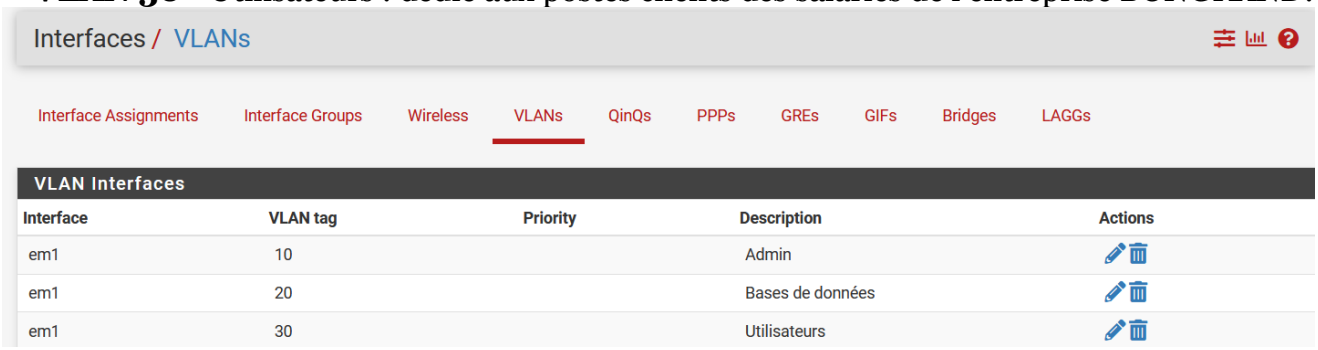
Mise en place des VLANs

Afin de sécuriser le réseau de l'entreprise BONGRAND, j'ai mis en place trois VLAN :

VLAN 10 - Administration : regroupe le serveur Windows Server ainsi que le poste de l'administrateur.

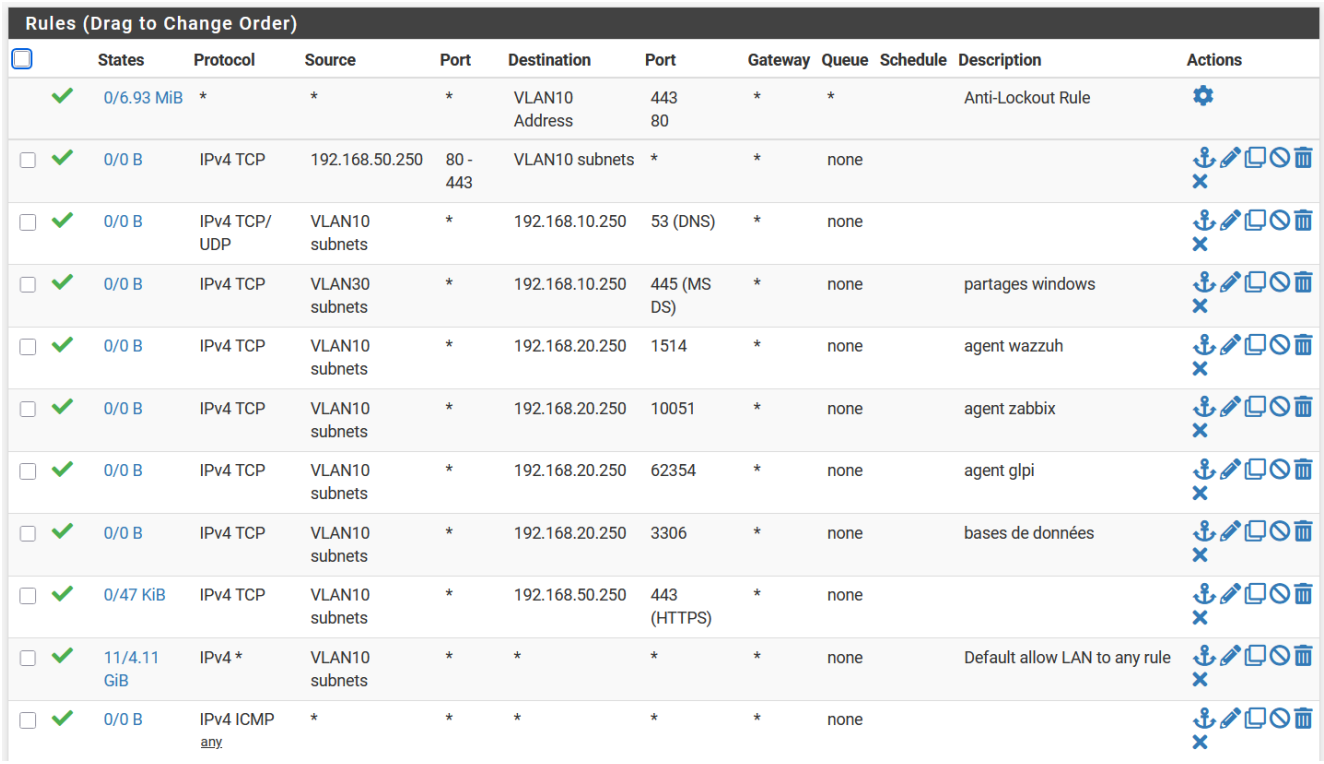
VLAN 20 - Bases de données : contient le serveur Debian hébergeant les bases de données ainsi que les services internes tels que Wazuh et Zabbix.

VLAN 30 - Utilisateurs : dédié aux postes clients des salariés de l'entreprise BONGRAND.



Mise en place des règles de Pare-feu

VLAN 10



0/0 B	IPv6 *	VLAN10 subnets	*	*	*	*	none	Default allow LAN IPv6 to any rule		
0/640 B	IPv4 *	*	*	*	*	*	none			

VLAN 20

Rules (Drag to Change Order)

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0/121.12 MiB	IPv4 TCP	192.168.20.250	*	*	80 - 443	*	none			
0/0 B	IPv4 TCP	VLAN30 subnets	*	192.168.20.250	3306	*	none		bases de données	
0/0 B	IPv4 TCP	192.168.50.250	*	192.168.20.250	3306	*	none			
0/0 B	IPv4 TCP	192.168.20.250	*	192.168.50.250	3306	*	none			
0/227 KiB	IPv4 TCP	192.168.20.250	*	*	10050 - 10051	*	none			
0/0 B	IPv4 TCP	VLAN30 subnets	1514 - 1515	192.168.20.250	1514 - 1515	*	none			
0/0 B	IPv4 TCP	192.168.50.250	1514 - 1515	192.168.20.250	1514 - 1515	*	none			
0/0 B	IPv4 TCP	VLAN10 subnets	1514 - 1515	192.168.20.250	1514 - 1515	*	none			
0/428 KiB	IPv4 TCP/ UDP	192.168.20.250	*	This Firewall (self)	53 (DNS)	*	none			
0/0 B	IPv4 ICMP any	*	*	*	*	*	none			
0/1.12 MiB	IPv4 *	*	*	*	*	*	none			

VLAN 30

Floating WAN VLAN10 VLAN20 VLAN30 VLAN50

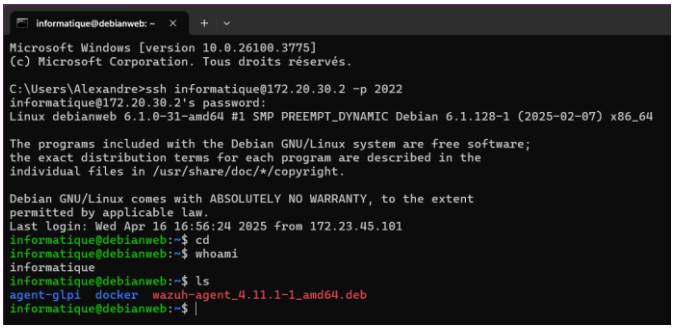
Rules (Drag to Change Order)

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0/15 KiB	IPv4 TCP/UDP	VLAN30 subnets	*	VLAN10 subnets	88	*	none			
0/0 B	IPv4 TCP	192.168.50.250	80 - 443	VLAN30 subnets	*	*	none			
0/5.91 MiB	IPv4 TCP/UDP	VLAN30 subnets	*	192.168.50.250	*	*	none			
0/325 KiB	IPv4 TCP	VLAN30 subnets	*	192.168.10.250	389 (LDAP)	*	none			
4/131 KiB	IPv4 UDP	VLAN30 subnets	*	192.168.10.250	53 (DNS)	*	none			
4/68.30 MiB	IPv4 TCP/UDP	VLAN30 subnets	*	*	443 (HTTPS)	*	none			
0/0 B	IPv4 TCP	VLAN30 subnets	*	192.168.50.250	443 (HTTPS)	*	none			
0/0 B	IPv4 UDP	VLAN30 subnets	*	This Firewall (self)	67 - 68	*	none			
0/0 B	IPv4 TCP	VLAN30 subnets	*	192.168.20.250	3306	*	none			
0/0 B	IPv4 TCP	VLAN30 subnets	*	192.168.20.250	10051	*	none			
0/0 B	IPv4 TCP	VLAN30 subnets	*	192.168.20.250	62354	*	none			
0/108 KiB	IPv4 TCP	VLAN30 subnets	*	192.168.10.250	445 (MS DS)	*	none			
0/0 B	IPv4 TCP	VLAN30 subnets	*	192.168.20.250	1514	*	none			
0/3 KiB	IPv4 ICMP any	*	*	*	*	*	none			
0/621 KiB	IPv4 *	*	*	*	*	*	none			

SERVEUR DEBIAN

J'ai décidé d'installer un serveur Debian 12 afin d'y héberger les services de bases de données, Wazuh et Zabbix. J'ai choisi Debian car il s'agit d'un système d'exploitation open source, reconnu pour sa légèreté et sa fiabilité, ce qui en fait un excellent choix pour un serveur. L'installation est simple et rapide.

Ce serveur est nommé "debiandatabases", il est accessible via son adresse IP en ssh avec le mot de passe : debianalexandre



Une fois notre serveur en fonctionnement, je peux installer les services nécessaires pour l'entreprise BONGRAND en commençant par Docker.

Installation Docker et Portainer

L'installation de Docker se fait directement en ligne de commande sur notre machine hôte. C'est une installation simple qui nécessite que quelques lignes de commandes.

Une fois notre Debian mise à jour et curl d'installé, nous allons installer GPG de Docker à fin de permettre aux utilisateurs de communiqué de manière sûre avec le dépôt.

```
install -m 0755 -d /etc/apt/keyrings
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Nous ajoutons l'adresse du dépôt Docker dans les sources.

Nous devons refaire un `apt update` car nous avons ajouté un dépôt.

Une fois cette update faite, nous pouvons lancer l'installation de Docker et Docker-compose en plugin, il nous sera utile pour créer des multi-conteneurs, tel qu'un WordPress et sa base de données.

```
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Ajouter notre utilisateur dans le groupe Docker.

```
usermod -aG docker « utilisateur »
```

Portainer

À présent que nous avons Docker d'installé, nous pouvons installer Portainer, ce qui va nous permettre de gérer nos machines virtuelles plus facilement et grâce à une interface web.

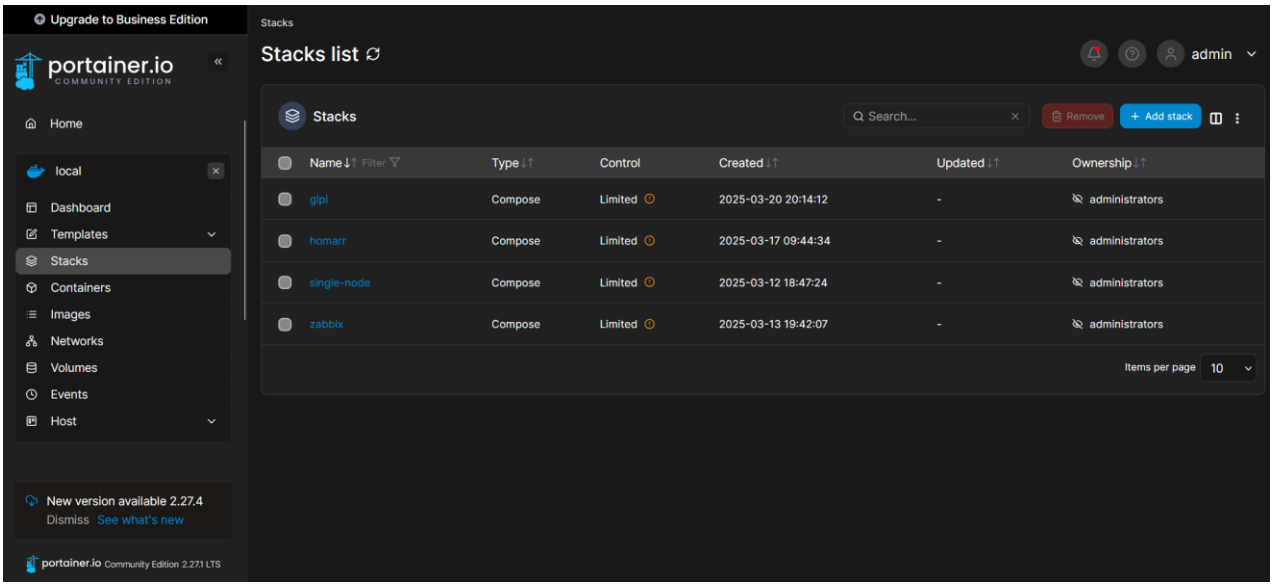
Nous allons créer un volume Docker nommé "portainer_data" pour stocker les données de Portainer.

```
Docker volume create portainer_data
```

Lancer le conteneur Portainer.

```
docker run -d --name=portainer --restart=always -p 8000:8000 -p 9443:9443 -v /var/run/docker.sock:/var/run/docker.sock -v
 portainer_data:/data portainer/portainer-ce:latest
```

Maintenant que nous avons Portainer d'installé, nous pouvons déployer des services plus simplement grâce à des stacks



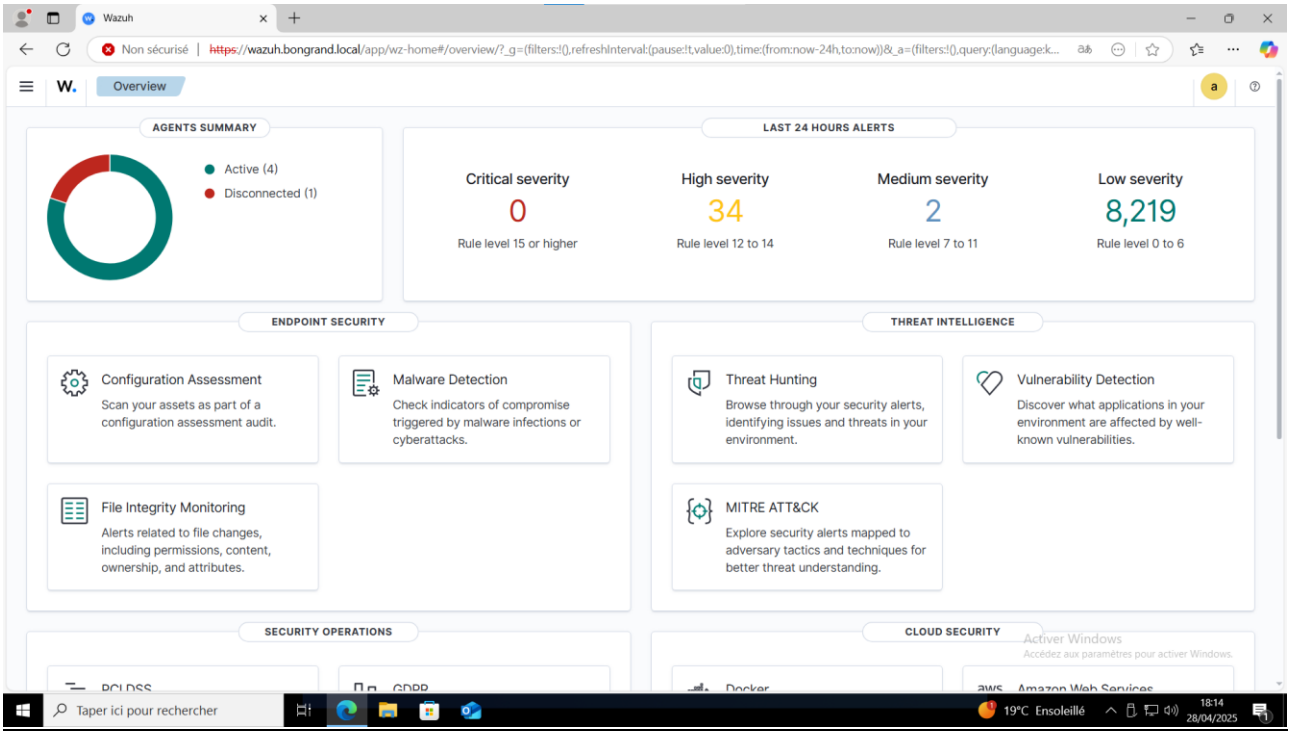
Wazuh

L'installation de Wazuh a été réalisée à l'aide d'un fichier Docker-compose.yml, ce qui permet de déployer la solution en seulement cinq minutes, le temps nécessaire pour télécharger l'image.

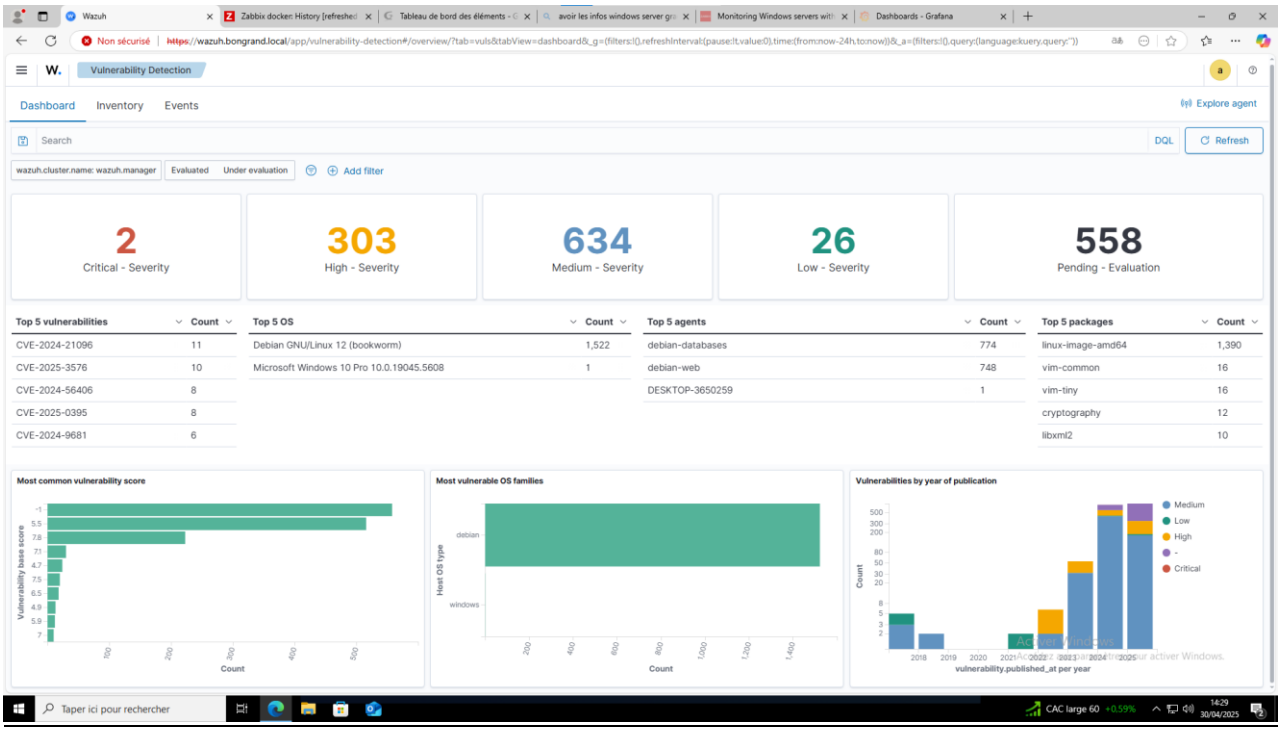
Docker-compose.yml :

```
  GNU nano 7.2                  docker-compose.yml
# Wazuh App Copyright (C) 2017, Wazuh Inc. (License GPLv2)
version: '3.7'

services:
  wazuh.manager:
    image: wazuh/wazuh-manager:4.11.1
    hostname: wazuh.manager
    restart: always
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 655360
        hard: 655360
    ports:
      - "1514:1514"
      - "1515:1515"
      - "514:514/udp"
      - "55000:55000"
    environment:
      - INDEXER_URL=https://wazuh.indexer:9200
      - INDEXER_USERNAME=admin
      - INDEXER_PASSWORD=SecretPassword
      - FILEBEAT_SSL_VERIFICATION_MODE=full
      - SSL_CERTIFICATE_AUTHORITIES=/etc/ssl/root-ca.pem
      - SSL_CERTIFICATE=/etc/ssl/filebeat.pem
      - SSL_KEY=/etc/ssl/filebeat.key
      - API_USERNAME=wazuh-wui
      - API_PASSWORD=MyS3cr37P450r.*-
    volumes:
      - wazuh_api_configuration:/var/ossec/api/configuration
      - wazuh_etc:/var/ossec/etc
      - wazuh_logs:/var/ossec/logs
      - wazuh_queue:/var/ossec/queue
      - wazuh_var_multigroups:/var/ossec/var/multigroups
      - wazuh_integrations:/var/ossec/integrations
```

Interface web :



Configuration Wazuh :



J'ai ajouté une configuration pour que Wazuh bloque les tentatives de Brute-Force sur SSH.

Configuration bloque SSH :

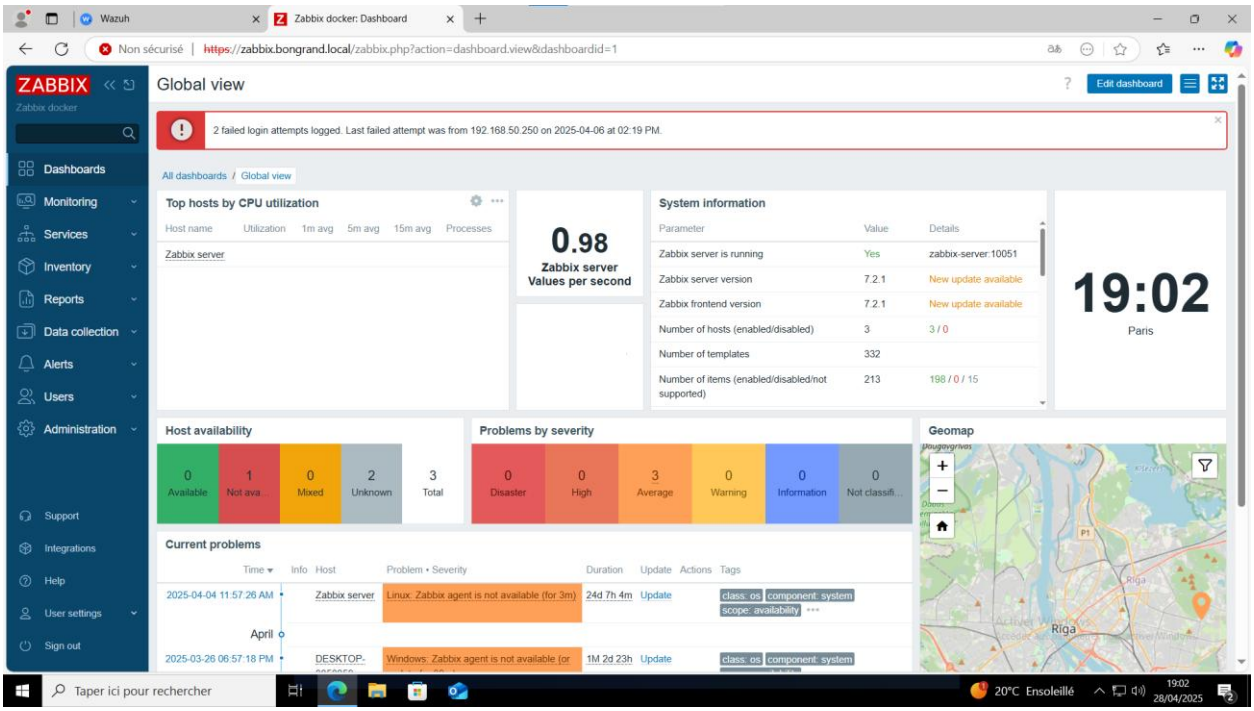
Zabbix

Idem que pour l'installation de Wazuh, Zabbix a été installé grâce à un docker-compose.yml ce qui facilite et améliore la gestion de ce service.

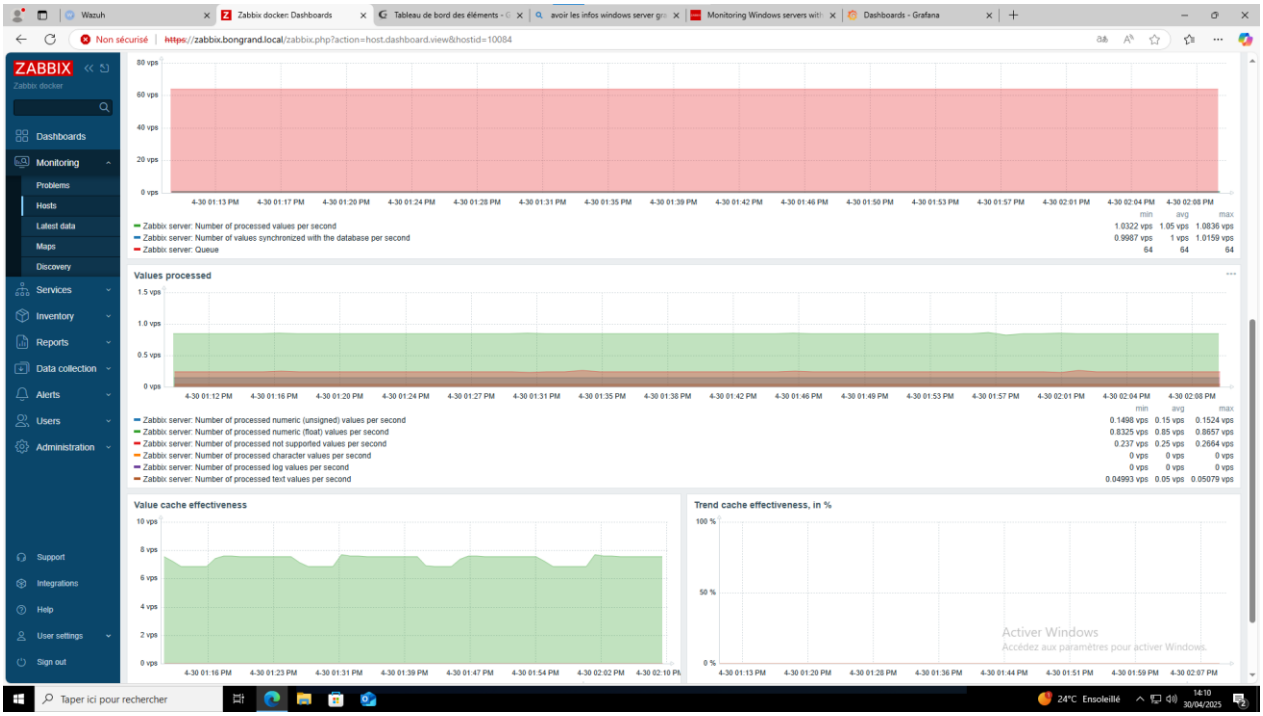
Docker-compose.yml :

```
 GNU nano 7.2                    docker-compose.yml
services:
  zabbix-server:
    image: ${ZABBIX_SERVER_IMAGE:-zabbix/zabbix-server-mysql:ubuntu-7.2.1}
    container_name: zabbix-server
    restart: unless-stopped
    ports:
      - "10051:10051"
    environment:
      DB_SERVER_HOST: ${DB_SERVER_HOST}
      DB_SERVER_PORT: 3306
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
    networks:
      - network-zabbix

  zabbix-frontend:
    image: ${ZABBIX_FRONTEND_IMAGE:-zabbix/zabbix-web-nginx-mysql:ubuntu-7.>
    restart: unless-stopped
    container_name: frontend
    ports:
      - "8880:8080"
      - "8443:8443"
    environment:
      DB_SERVER_HOST: ${DB_SERVER_HOST}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      PHP_TZ: ${TZ}
      ZBX_SERVER_HOST: zabbix-server
      ZBX_SERVER_PORT: 10051
    depends_on:
      - zabbix-server
    networks:
      - network-zabbix

  zabbix-agent:
```

Interface web :



Configuration Zabbix :



Installation Base de données

Pour utiliser tous ces services, une base de données est nécessaire. J'ai choisi d'installer MariaDB directement sur le serveur, sans utiliser de conteneur Docker.

J'ai sécurisé l'installation en utilisant la commande mysql_secure_installation, ce qui m'a permis de définir un mot de passe pour accéder à la base de données.

Après l'installation de MariaDB, j'ai créé la base de données pour le service Zabbix ainsi que l'utilisateur associé. Cet utilisateur a un caractère " % " dans le champ "host", ce qui lui permet de se connecter à la base de données depuis n'importe quelle adresse IP.

```
+--------------------+
| Database           |
+--------------------+
| glpi               |
| information_schema |
| mysql              |
| nextcloud          |
| nginx_proxy        |
| performance_schema |
| presta             |
| sys                |
| wordpress          |
| zabbix             |
+--------------------+
```

```
+--------------+------+
| user         | host |
+--------------+------+
| zabbix_user  | %    |
+--------------+------+
```

```
Database changed
MariaDB [zabbix]> show tables;
+----------------------------+
| Tables_in_zabbix           |
+----------------------------+
| acknowledges               |
| actions                    |
| alerts                     |
| auditlog                   |
| autoreg_host               |
| changelog                  |
| conditions                 |
| config                     |
| config_autoreg_tls         |
| connector                  |
| connector_tag              |
| corr_condition             |
| corr_condition_group       |
| corr_condition_tag         |
| corr_condition_tagpair     |
| corr_condition_tagvalue    |
| corr_operation             |
| correlation                |
| dashboard                  |
| dashboard_page             |
| dashboard_user             |
| dashboard_usrgrp           |
| dbversion                  |
| dchecks                    |
```