PortailSIO

NOUVELLE FORMULE

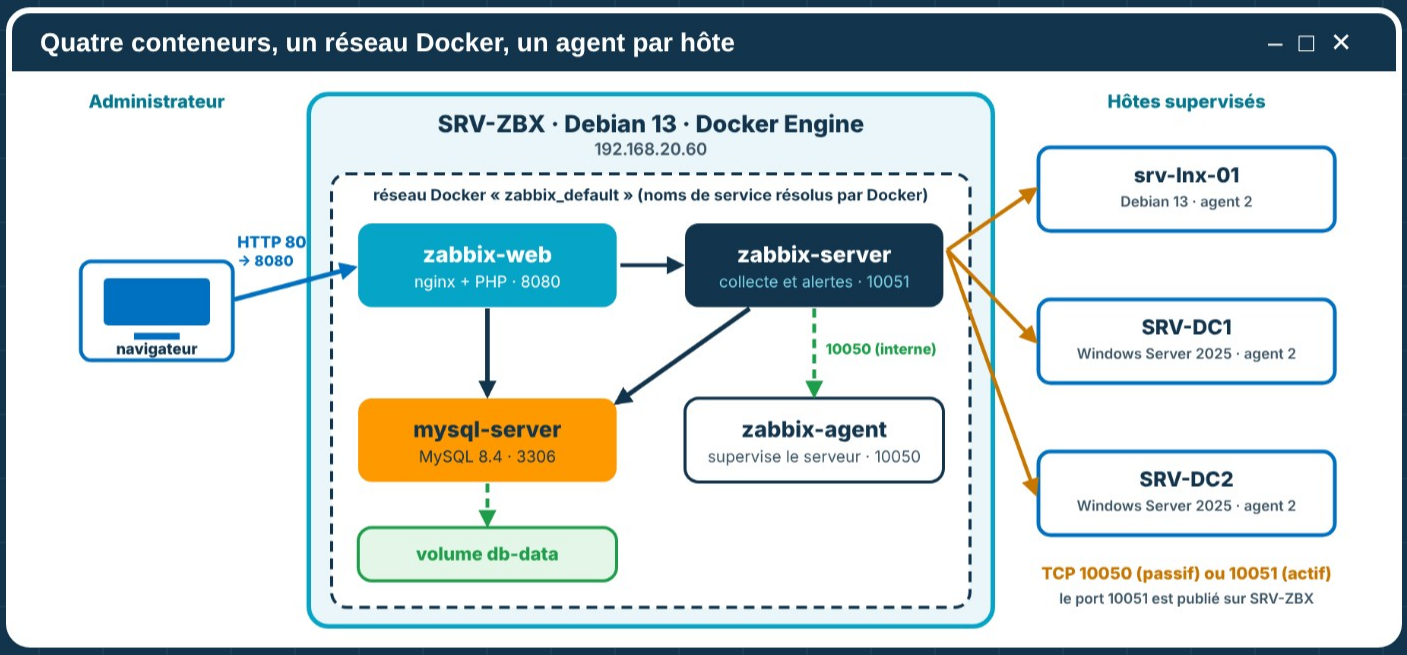
TUTORIEL

DOCKER COMPOSE · ZABBIX 8.0 LTS

Installer avec Docker Compose

ZABBIX





1 Préparer le serveur Debian 13

2 Installer Docker Engine et Docker Compose

3 Écrire le projet Docker Compose de Zabbix

4 Démarrer la pile et se connecter

5 Remonter les hôtes dans Zabbix

6 Exploiter : sauvegarder, mettre à jour, dépanner

CC BY-NC
Utilisation commerciale interdite
© PortailSIO – 11/2026

Ce tutoriel en ligne

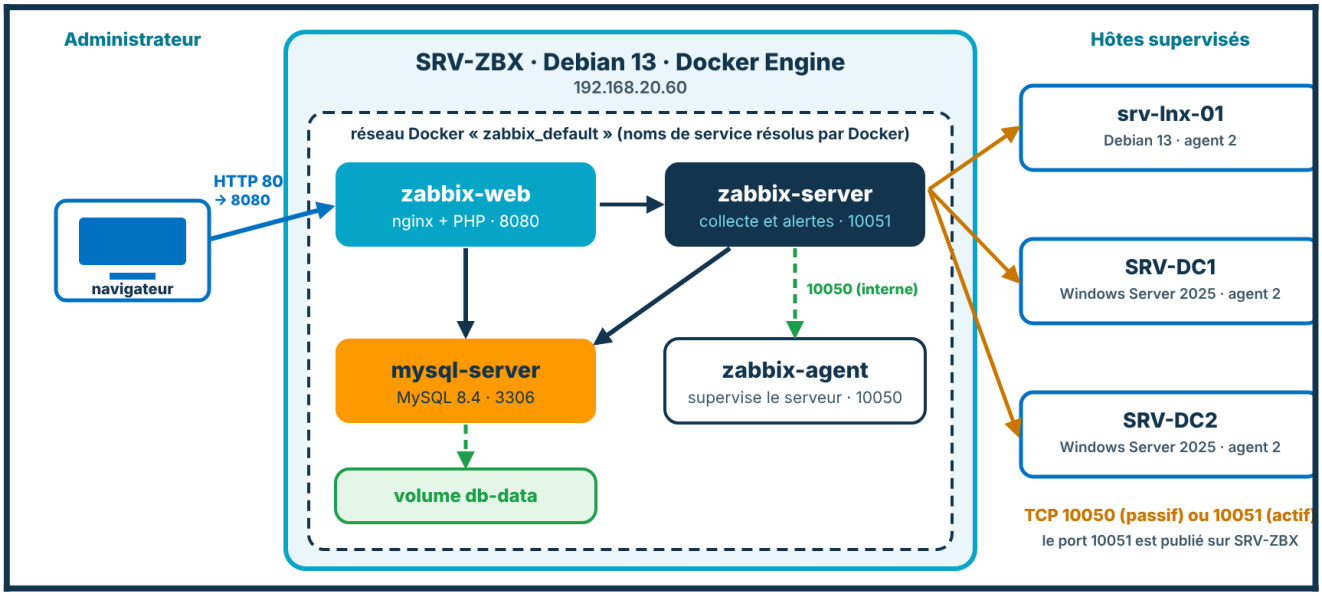
Scannez le QR code. Tous les tutoriels :

portail.sio-ndlp.fr/tutoriels.php

Ce tutoriel a été réalisé avec l'hyperviseur © Proxmox VE. Il peut être réalisé dans un environnement de virtualisation personnel à l'aide des logiciels © VirtualBox ou © VMware Player. Il **suppose que vous disposez d'une machine virtuelle Debian 13** (installation minimale avec serveur SSH) et, pour la partie Windows, du domaine **sio.local** construit dans les tutoriels précédents (contrôleurs **SRV-DC1** et **SRV-DC2** sous Windows Server 2025).

Zabbix est une solution de **supervision** libre : elle collecte à intervalle régulier des mesures sur vos serveurs (processeur, mémoire, disque, services, disponibilité…), les compare à des seuils et **vous alerte** avant que les utilisateurs ne s'aperçoivent de la panne. La version **8.0** est une version **LTS** (Long Term Support) : elle bénéficie d'un support de plusieurs années.

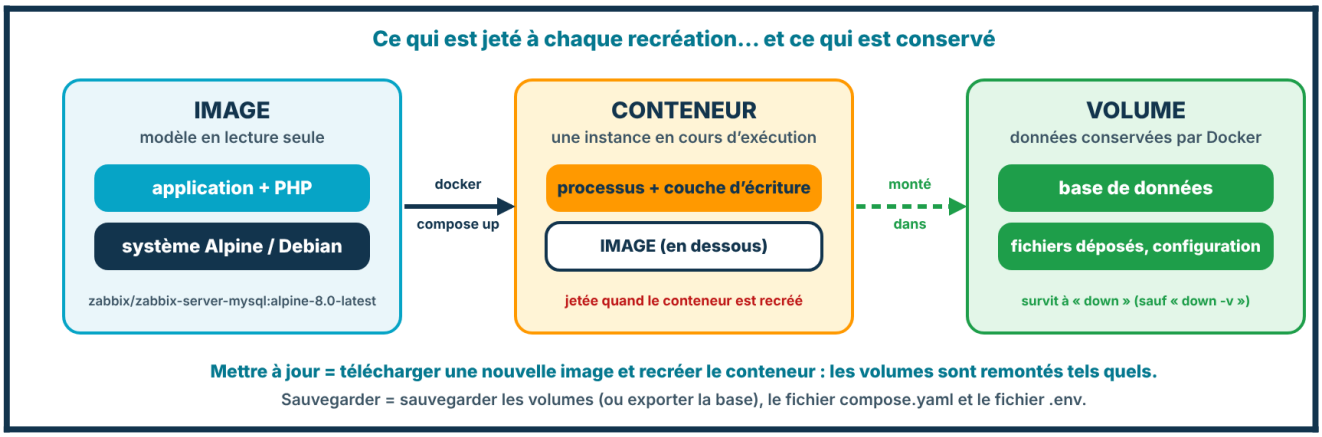
Dans ce tutoriel, nous installons Zabbix **avec Docker Compose** : chaque composant s'exécute dans son propre **conteneur** et toute l'installation se décrit dans **deux fichiers** (`compose.yaml` et `.env`). Pas de dépôt à ajouter ni de paquet à installer sur le serveur : on télécharge des **images** prêtes à l'emploi. Voici l'architecture obtenue :



Pourquoi Docker Compose ?

- **Rapide et reproductible** : la même commande installe la même pile sur n'importe quel serveur Debian
- **Isolé** : Zabbix, son interface et sa base ne dépendent pas des paquets du système
- **Simple à mettre à jour** : on change le numéro de version de l'image et on recrée les conteneurs
- **Contrepartie** : il faut comprendre trois notions (image, conteneur, volume) et savoir où sont les données

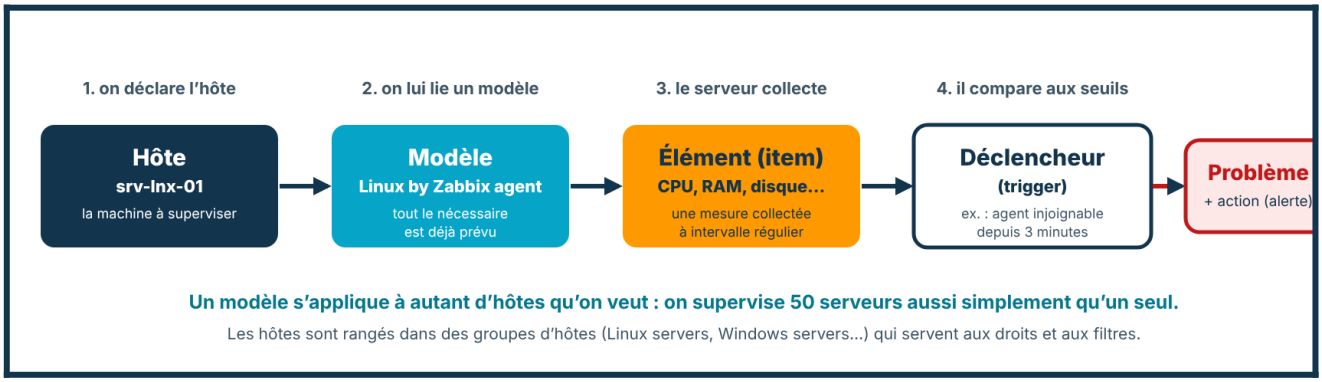
Une **image** est le modèle en lecture seule d'une application ; un **conteneur** en est une instance en cours d'exécution ; un **volume** conserve les données en dehors du conteneur, ce qui permet de supprimer et de recréer le conteneur sans rien perdre :



Les quatre conteneurs

- **zabbix-server** : le cœur de Zabbix, il interroge les hôtes, évalue les seuils et déclenche les alertes
- **mysql-server** : la base de données (MySQL 8.4) qui stocke la configuration et l'historique des mesures ; ses données sont dans le volume `db-data`
- **zabbix-web** : l'interface web (nginx et PHP) pour administrer Zabbix et consulter les tableaux de bord
- **zabbix-agent** : l'agent Zabbix 2 qui supervise… le serveur Zabbix lui-même (chaque hôte supervisé aura aussi son propre agent)

Le vocabulaire de Zabbix suit une logique simple : on déclare un **hôte**, on lui associe un **modèle** (template) qui contient les mesures et les seuils déjà prêts, et Zabbix lève un **problème** quand un seuil est franchi :



Dans le cadre de ce tutoriel, nous utilisons les machines suivantes (VLAN SERVEURS, réseau 192.168.20.0 /24) :

Machine	Rôle	Système	Adresse IP
SRV-ZBX	Serveur Docker exécutant Zabbix (nom `srv-zbx`)	Debian 13	192.168.20.60
srv-lnx-01	Hôte Linux supervisé	Debian 13	192.168.20.61
SRV-DC1	Hôte Windows supervisé (contrôleur de domaine)	Windows Server 2025	192.168.20.250
SRV-DC2	Hôte Windows supervisé (contrôleur de domaine)	Windows Server 2025	192.168.20.251

Pour **SRV-ZBX**, prévoyez au minimum **2 vCPU, 4 Go de mémoire et 30 Go de disque**. Ces valeurs, comme les adresses et les mots de passe d'exemple, sont à adapter à votre environnement. Les commandes sont exécutées avec un compte autorisé à utiliser `sudo` (ou directement en `root`, sans le préfixe `sudo`).

01 ÉTAPE 1 SUR 6 — Préparer le serveur Debian 13

Docker n'a besoin que d'un système à jour, d'un nom, d'une adresse fixe et de l'heure exacte. Ouvrez une session (console Proxmox ou SSH) sur **SRV-ZBX** puis effectuez les manipulations suivantes :

1. Mettez le système à jour : `sudo apt update && sudo apt full-upgrade -y`
2. Donnez son nom au serveur et déclarez-le dans le fichier `hosts` (nom court et nom complet) :

```
TERMINAL
sudo hostnamectl set-hostname srv-zbx
echo "192.168.20.60  srv-zbx.sio.local  srv-zbx" | sudo tee -a /etc/hosts
```

3. Réglez le fuseau horaire : `sudo timedatectl set-timezone Europe/Paris`
4. Donnez au serveur une **adresse IP fixe** (adaptez le nom de la carte, ici `ens18`, avec `ip -br address`) : éditez `/etc/network/interfaces` puis redémarrez le réseau avec `sudo systemctl restart networking` :

```
/etc/network/interfaces
auto ens18
iface ens18 inet static
    address 192.168.20.60/24
    gateway 192.168.20.254
```

5. Indiquez les **DNS du domaine** dans `/etc/resolv.conf` : les conteneurs s'en servent pour résoudre les noms du domaine (par exemple `srv-dc1.sio.local`) :

```
/etc/resolv.conf
search sio.local
nameserver 192.168.20.250
nameserver 192.168.20.251
```

6. Vérifiez le résultat : nom, fuseau, heure synchronisée, adresse IP et résolution d'un nom du domaine :

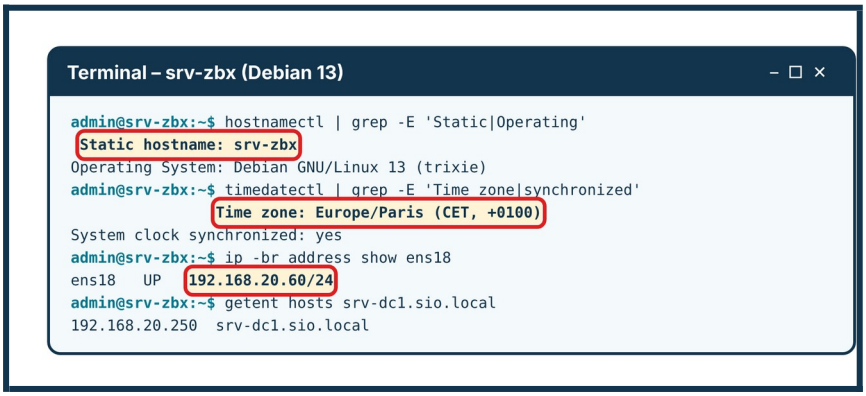


Figure 1 – Contrôle de la préparation du serveur

À SAVOIR

Si « System clock synchronized » indique « no », installez un service de temps (`sudo apt install -y chrony`). Zabbix horodate chaque mesure : une horloge fausse décale les graphiques et les alertes.

✔ POINT DE CONTRÔLE

- `hostnamectl` affiche le nom **srv-zbx** et Debian GNU/Linux **13**
- L'adresse IP **192.168.20.60** est fixe
- Le fuseau horaire est **Europe/Paris** et l'horloge est **synchronisée**
- `getent hosts srv-dc1.sio.local` renvoie l'adresse du contrôleur

ÉTAPE 2 SUR 6

02 Installer Docker Engine et Docker Compose

Docker Engine est le moteur qui exécute les conteneurs ; **Docker Compose** est son extension qui décrit toute une application (plusieurs conteneurs, un réseau, des volumes) dans **un seul fichier** et la pilote avec une seule commande. Docker s'installe depuis le **dépôt officiel de Docker** : les paquets de Debian sont plus anciens et n'incluent pas Compose. Sur le serveur :

A – Ajouter le dépôt officiel de Docker

1 Retirez d'éventuels paquets concurrents (sur une installation neuve, il n'y a rien à retirer) puis installez les outils de téléchargement :

TERMINAL

```
sudo apt remove $(dpkg --get-selections docker.io docker-compose docker-doc \
    podman-docker containerd runc 2>/dev/null | cut -f1)
sudo apt update && sudo apt install -y ca-certificates curl
```

2 Enregistrez la clé de signature du dépôt :

TERMINAL

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg \
    -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

3 Déclarez le dépôt (format « deb822 »). Le nom de la version de Debian, `trixie`, est lu automatiquement dans le système :

TERMINAL

```
sudo tee /etc/apt/sources.list.d/docker.sources > /dev/null <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF
sudo apt update
```

B – Installer Docker Engine et le plugin Compose

1 Installez les cinq paquets de Docker :

TERMINAL

```
sudo apt install -y docker-ce docker-ce-cli containerd.io \
    docker-buildx-plugin docker-compose-plugin
```

Paquet	Rôle
docker-ce	Le moteur (service `docker`) qui crée et exécute les conteneurs
docker-ce-cli	La commande `docker`
containerd.io	Le composant de bas niveau qui exécute réellement les conteneurs
docker-buildx-plugin	L'outil de construction d'images (`docker build`)
docker-compose-plugin	**Docker Compose** : la commande `docker compose`

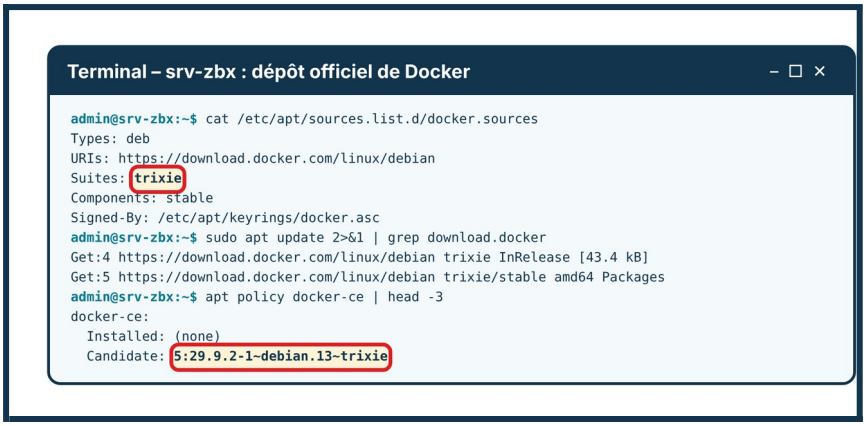


Figure 2 – Dépôt officiel de Docker et version candidate

À SAVOIR

Si la ligne « Candidate » indique une version venant de Debian (`docker.io`), le dépôt n'est pas pris en compte : relisez le fichier `docker.sources` (indentation, nom de la version).

C – Vérifier que Docker fonctionne

1 Le service démarre tout seul à l'installation. Contrôlez-le, affichez les versions, puis lancez le conteneur de test officiel :

TERMINAL

```
sudo systemctl is-active docker containerd
sudo docker --version && sudo docker compose version
sudo docker run --rm hello-world
```

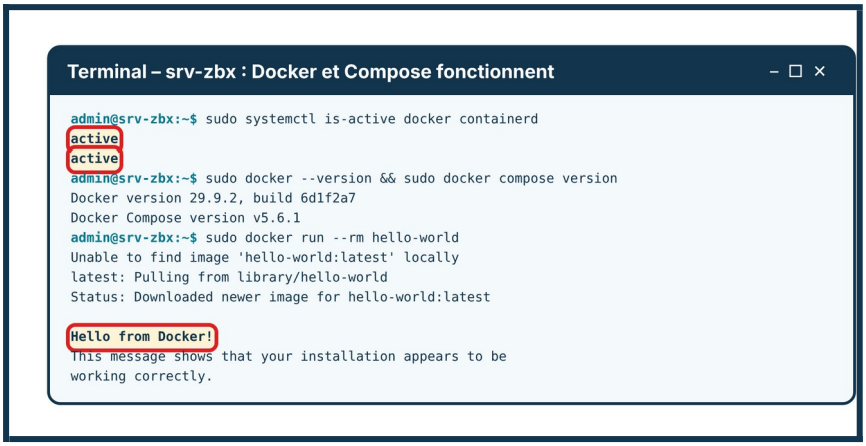


Figure 3 – Docker et Compose en état de marche

À SAVOIR

Le message « Hello from Docker! » prouve que le moteur sait télécharger une image sur Internet et exécuter un conteneur. L'option `--rm` supprime ensuite ce conteneur de test.

À SAVOIR

Les commandes de ce tutoriel utilisent `sudo docker`. Ajouter votre compte au groupe `docker` permettrait de s'en passer, mais ce groupe équivaut à des droits **root** sur le serveur : on l'évite en production. Autre point d'attention : les ports **publiés** par Docker (`ports:`) ne sont pas filtrés par un pare-feu configuré sur l'hôte (ufw, firewalld) ; limitez au besoin leur accès par une règle réseau en amont.

✔ **POINT DE CONTRÔLE**

- Le dépôt `download.docker.com` apparaît dans `apt update` et `apt policy docker-ce` propose une version **5:29** (ou supérieure)
- `systemctl is-active docker` répond **active**
- `docker compose version` affiche une version de **Docker Compose**
- `docker run --rm hello-world` affiche « **Hello from Docker!** »

03

ÉTAPE 3 SUR 6

Écrire le projet Docker Compose de Zabbix

Un **projet Compose** est un dossier qui contient le fichier `compose.yaml` (les conteneurs, leurs réglages, leurs volumes) et, à côté, le fichier `.env` (les valeurs qui changent d'un serveur à l'autre : versions, mots de passe). Séparer les deux permet de **partager** le premier sans divulguer les secrets du second.

A – Créer le dossier et le fichier d'environnement

1 Créez le dossier du projet, puis le fichier `.env` avec `sudo nano /opt/zabbix/.env` :

TERMINAL

```
sudo mkdir -p /opt/zabbix
cd /opt/zabbix
```

/opt/zabbix/.env

```
ZBX_TAG=alpine-8.0-latest
MYSQL_TAG=8.4
TZ=Europe/Paris
DB_PASSWORD=Zbx_SIO_2026
DB_ROOT_PASSWORD=Root_SIO_2026
```

2 Ce fichier contient des mots de passe : réservez-le à `root` :

TERMINAL

```
sudo chmod 600 /opt/zabbix/.env
```

REMARQUE IMPORTANTE

`Zbx_SIO_2026` et `Root_SIO_2026` ne sont que des exemples : **choisissez deux mots de passe solides et différents**. Attention : MySQL ne lit ces valeurs **qu'à la première création** du volume de données ; les modifier ensuite dans `.env` ne change pas les mots de passe déjà enregistrés dans la base (voir le dépannage).

Variable	Rôle
`ZBX_TAG`	Version des images Zabbix. `alpine-8.0-latest` suit les mises à jour de la branche **8.0** ; une valeur précise comme `alpine-8.0.0` fige la version
`MYSQL_TAG`	Version de MySQL (8.4 est la version de maintenance à long terme prise en charge par Zabbix)
`TZ`	Fuseau horaire appliqué aux conteneurs et à l'interface
`DB_PASSWORD`, `DB_ROOT_PASSWORD`	Mot de passe de l'utilisateur `zabbix` de la base et mot de passe de l'administrateur MySQL

B – Écrire le fichier compose.yaml

Le fichier est écrit en **YAML** : l'**indentation** (deux espaces, jamais de tabulation) donne la structure. Créez `sudo nano /opt/zabbix/compose.yaml`. Voici d'abord la base de données et le serveur Zabbix :

/opt/zabbix/compose.yaml (1/2)

```
name: zabbix

services:
  mysql-server:
    image: mysql:${MYSQL_TAG}
    restart: unless-stopped
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_bin
      - --skip-character-set-client-handshake
      - --skip-log-bin
      - --mysql-native-password=ON
    environment:
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: ${DB_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
      TZ: ${TZ}
    volumes:
      - db-data:/var/lib/mysql
    healthcheck:
      test: ["CMD-SHELL", "mysqladmin ping -h localhost -uroot -p$$MYSQL_ROOT_PASSWORD"]
      interval: 10s
      timeout: 5s
      retries: 12
      start_period: 30s

  zabbix-server:
    image: zabbix/zabbix-server-mysql:${ZBX_TAG}
    restart: unless-stopped
    depends_on:
      mysql-server:
        condition: service_healthy
    environment:
      DB_SERVER_HOST: mysql-server
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: ${DB_PASSWORD}
      TZ: ${TZ}
    ports:
      - "10051:10051"
```

Puis, à la suite dans le même fichier (au même niveau que `mysql-server`), l'interface web, l'agent et la déclaration du volume :

/opt/zabbix/compose.yaml (2/2)

```
  zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql:${ZBX_TAG}
    restart: unless-stopped
    depends_on:
      - mysql-server
      - zabbix-server
    environment:
      ZBX_SERVER_HOST: zabbix-server
      ZBX_SERVER_NAME: Zabbix SIO
      DB_SERVER_HOST: mysql-server
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: ${DB_PASSWORD}
      PHP_TZ: ${TZ}
    ports:
      - "80:8080"

  zabbix-agent:
    image: zabbix/zabbix-agent2:${ZBX_TAG}
    restart: unless-stopped
    depends_on:
      - zabbix-server
    environment:
      ZBX_HOSTNAME: Zabbix server
      ZBX_SERVER_HOST: zabbix-server
      TZ: ${TZ}

volumes:
  db-data:
```

Élément	Signification
`image:`	L'image à télécharger, avec sa version (`${ZBX_TAG}` est remplacé par la valeur du fichier `.env`)
`restart: unless-stopped`	Le conteneur redémarre après une panne et au démarrage du serveur, sauf si vous l'avez arrêté vous-même
`environment:`	Les réglages de l'application. Les services se joignent par leur **nom** (`mysql-server`, `zabbix-server`) : Docker fournit un DNS interne
`depends_on` + `healthcheck`	Le serveur Zabbix ne démarre qu'une fois MySQL **prêt** (contrôle `mysqladmin ping`) ; sans cela, il échouerait à la première connexion
`ports: "80:8080"`	Publie le port **8080** du conteneur web sur le port **80** du serveur. Seuls les ports **80** (interface) et **10051** (agents actifs) sont ouverts : la base reste **invisible** du réseau
`volumes: db-data`	Volume nommé, géré par Docker, qui conserve la base de données
`--skip-log-bin`	Désactive le journal binaire de MySQL, inutile ici : cela évite l'erreur d'import « SUPER privilege and binary logging » du schéma

REMARQUE IMPORTANTE

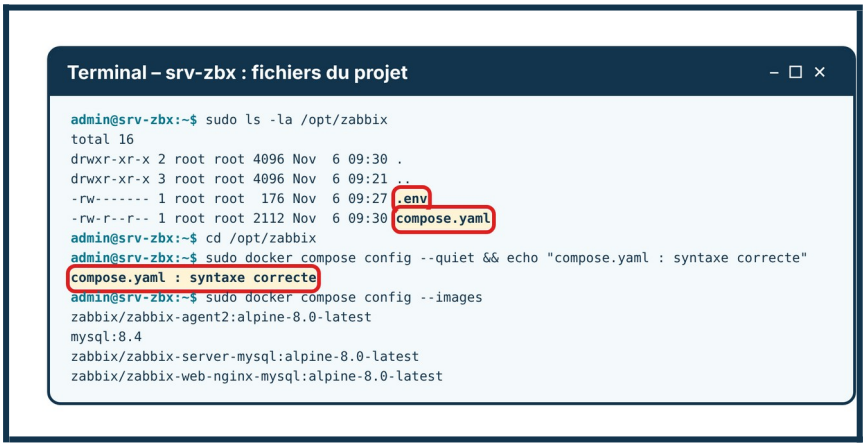
Dans un fichier YAML, une **erreur d'indentation** change le sens du fichier ou l'invalide. Évitez de copier-coller depuis un document qui remplace les espaces par des caractères spéciaux : la commande de validation de l'étape suivante repère la plupart des erreurs.

C – Valider le projet et télécharger les images

1 Contrôlez la syntaxe, puis affichez les images qui seront utilisées (les variables du fichier `.env` doivent y être remplacées) :

TERMINAL

```
cd /opt/zabbix
sudo docker compose config --quiet && echo "compose.yaml : syntaxe correcte"
sudo docker compose config --images
```



À SAVOIR

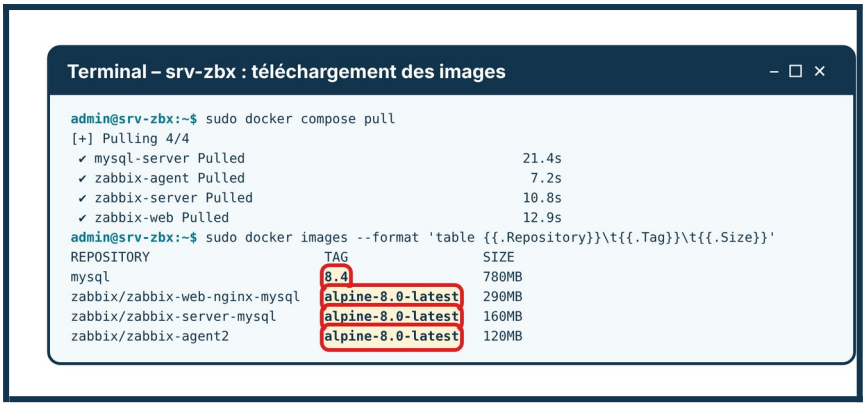
`docker compose config` affiche le fichier tel que Compose le comprend, variables remplacées : sans l'option `--quiet`, il montrerait aussi les mots de passe en clair. Une erreur indique la ligne en cause.

Figure 4 – Fichiers du projet et validation de la syntaxe

2 Téléchargez les quatre images (environ 1,4 Go au total) et vérifiez-les :

TERMINAL

```
sudo docker compose pull
sudo docker images
```



À SAVOIR

Le téléchargement est fait une seule fois : un démarrage ultérieur réutilise les images locales. Une erreur « pull access denied » ou « not found » signale une faute dans le nom ou la version de l'image (fichier `.env`).

Figure 5 – Téléchargement des images

✔ POINT DE CONTRÔLE

- `/opt/zabbix/.env` n'est lisible que par **root** (droits 600) et contient des mots de passe personnalisés
- `docker compose config --quiet` ne renvoie **aucune erreur**
- `docker images` liste **mysql 8.4** et les trois images **zabbix** en `alpine-8.0-latest`

04 ÉTAPE 4 SUR 6 Démarrer la pile et se connecter

A – Démarrer les conteneurs

1 Lancez toute la pile **en arrière-plan** (`-d`), puis affichez l'état des services :

TERMINAL

```
cd /opt/zabbix
sudo docker compose up -d
sudo docker compose ps
```

Compose crée le réseau et le volume, puis démarre **MySQL en premier** ; quand celui-ci est « healthy », il démarre les autres conteneurs. Le premier démarrage dure **une à deux minutes** : le serveur Zabbix **crée les tables** de la base (le schéma) avant de se lancer.

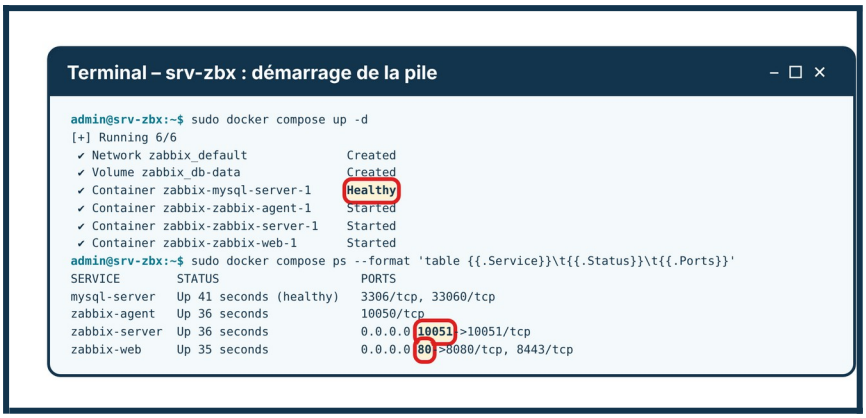


Figure 6 – Démarrage de la pile et état des services

À SAVOIR

Les ports 80 et 10051 sont publiés ; 3306 (MySQL) et 10050 (agent) ne sont accessibles qu'aux autres conteneurs. Si un conteneur indique « Restarting », lisez son journal (point suivant).

1 Suivez le journal du serveur : il indique la création du schéma puis le démarrage. Quittez avec **Ctrl + C** :

TERMINAL

```
sudo docker compose logs -f --tail 20 zabbix-server
```

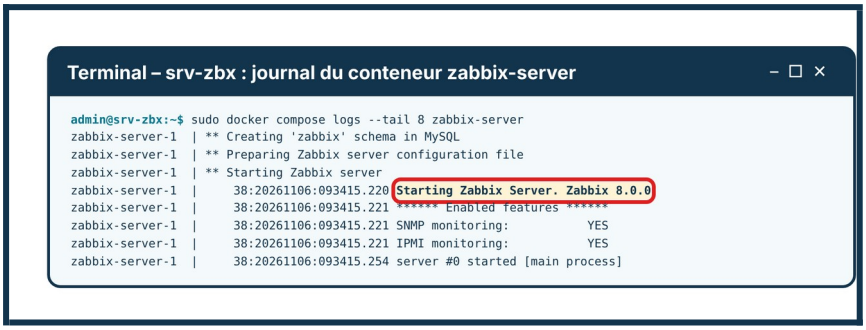


Figure 7 – Journal du conteneur zabbix-server

À SAVOIR

La ligne « Starting Zabbix Server » puis « server #0 started » confirment que le serveur fonctionne. Une erreur de connexion à la base se lit ici (mot de passe, base non prête).

B – Première connexion et mot de passe administrateur

À SAVOIR

Dans les conteneurs, il n'y a pas d'assistant d'installation dans le navigateur : l'image de l'interface écrit elle-même sa configuration à partir des variables d'environnement. On arrive directement sur la page de connexion.

1 Depuis un poste d'administration, ouvrez `http://192.168.20.60`, saisissez l'utilisateur `Admin` et le mot de passe `zabbix` (valeurs par défaut) puis cliquez « **Sign in** » :



Figure 8 – Page de connexion de Zabbix

À SAVOIR

Attention à la casse : le compte s'appelle bien « Admin » avec un A majuscule. Si la page affiche une erreur de base de données, attendez la fin de l'initialisation (point A).

Le tableau de bord « **Global view** » s'affiche. Dans « **System information** », la ligne « **Zabbix server is running** » doit indiquer « **Yes** » (le serveur est joint sur `zabbix-server:10051`) :



Figure 9 – Tableau de bord Global view

À SAVOIR

Si la ligne indique « No », le serveur Zabbix est arrêté ou encore en cours d'initialisation : consultez son journal.

REMARQUE IMPORTANTE

Changez immédiatement le mot de passe du compte Admin : « zabbix » est le mot de passe par défaut, connu de tous.

1. Cliquez le menu « **Users** » puis « **Users** » et ouvrez le compte « **Admin** »
2. Cliquez « **Change password** », saisissez le mot de passe actuel puis deux fois le nouveau mot de passe, et cliquez « **Update** » :

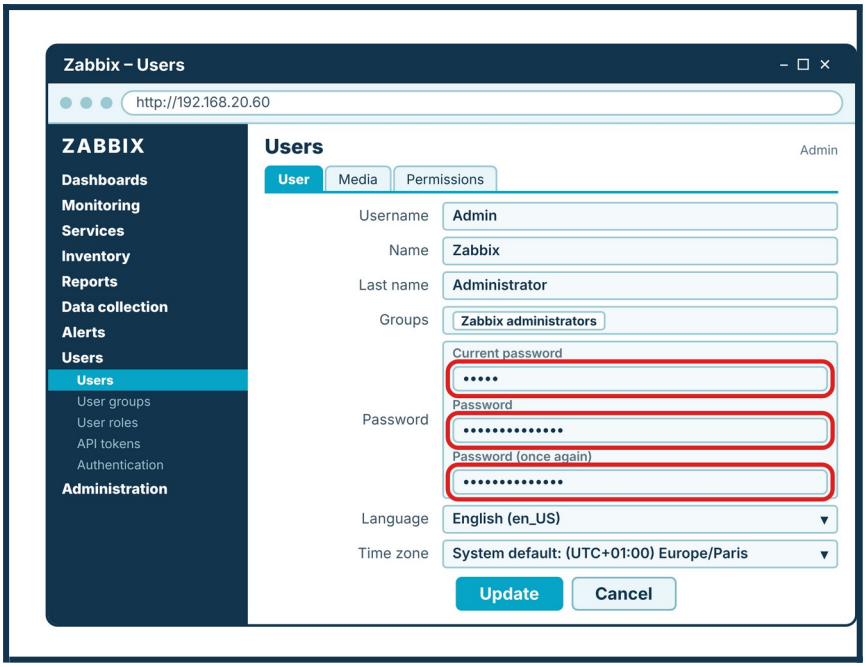


Figure 10 – Users > Users > Admin : changement du mot de passe

À SAVOIR

Les noms de menus de ce tutoriel sont ceux de l'interface en anglais. La langue peut être changée dans la même fiche (champ « Language »), si elle est proposée par l'image.

C – Rattacher l'agent du serveur Zabbix

Zabbix crée automatiquement un hôte « **Zabbix server** », configuré pour joindre son agent à l'adresse `127.0.0.1`. Or, ici, l'agent est dans **un autre conteneur** : `127.0.0.1` désigne le conteneur du serveur lui-même, où aucun agent n'écoute. Il faut donc indiquer le **nom du service** :

1. Cliquez le menu « **Data collection** » puis « **Hosts** » et ouvrez l'hôte « **Zabbix server** »
2. Dans « **Interfaces** », videz l'adresse IP, saisissez le nom DNS `zabbix-agent`, choisissez « **Connect to** » : « **DNS** » (port `10050`) puis cliquez « **Update** » :

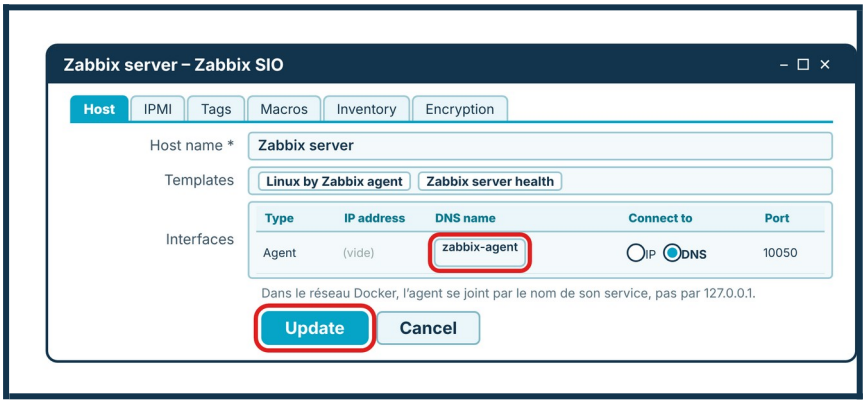


Figure 11 – Hôte Zabbix server : interface de l'agent

À SAVOIR

Après une à deux minutes, la pastille « ZBX » de l'hôte passe au vert dans « Monitoring > Hosts ». Tant que ce n'est pas fait, un problème « Zabbix agent is not available » est affiché pour l'hôte « Zabbix server ».

À SAVOIR

En production, publiez l'interface en **HTTPS** (certificat délivré par votre autorité interne) : sans cela, le mot de passe circule en clair sur le réseau. Le tutoriel « Installer Zabbix derrière Traefik » montre comment le faire avec un proxy inverse.

✔ **POINT DE CONTRÔLE**

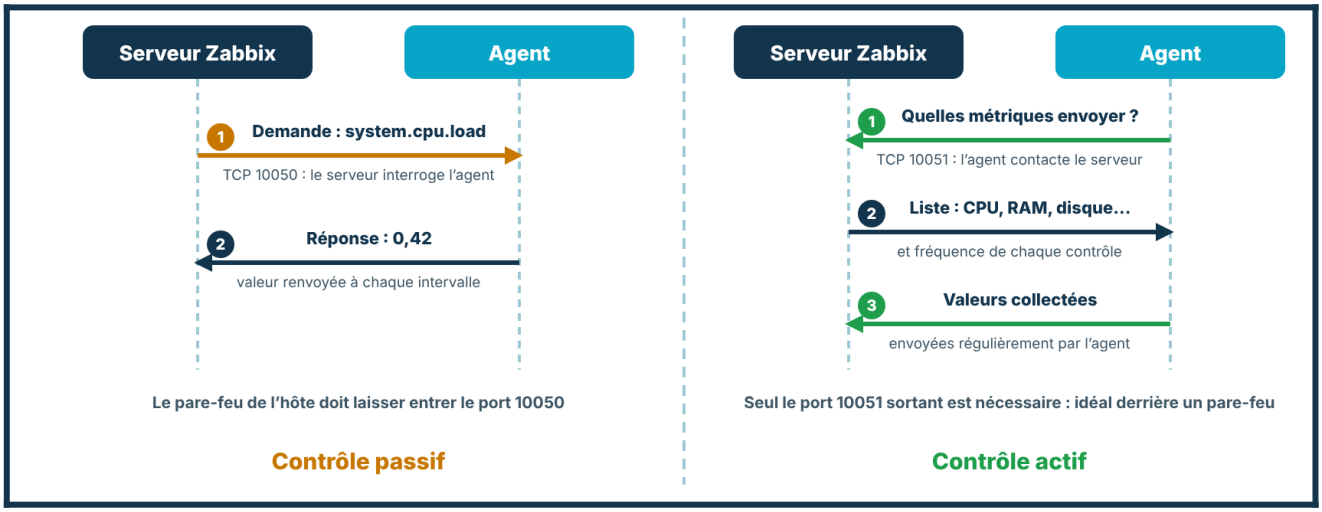
- `docker compose ps` indique **4 services** démarrés (MySQL « healthy »)
- « **Zabbix server is running** » indique « **Yes** » dans le tableau de bord
- Le mot de passe du compte **Admin** a été **changé**
- L'hôte « Zabbix server » est **disponible** (pastille ZBX verte) avec l'interface DNS `zabbix-agent`

05 ÉTAPE 5 SUR 6 Remonter les hôtes dans Zabbix

« Remonter » un hôte, c'est le faire apparaître dans Zabbix pour le superviser. Cela demande toujours trois choses : **installer l'agent** sur la machine, **déclarer l'hôte** dans Zabbix (à la main ou automatiquement) et **lui associer un modèle**. Les hôtes à superviser sont des machines **ordinaires** : leur agent s'installe comme d'habitude, seul le serveur Zabbix est conteneurisé. Nous allons procéder de trois manières : un hôte Linux (manuellement), un serveur Windows (manuellement) puis un second serveur Windows qui s'enregistre **tout seul**.

A – Passif ou actif : deux façons de collecter

L'agent peut fonctionner de deux manières, qui déterminent **qui contacte qui** :



	Contrôle passif	Contrôle actif
Qui initie ?	Le serveur interroge l'agent	L'agent contacte le serveur
Port	10050/TCP, entrant sur l'hôte	10051/TCP, sortant de l'hôte (vers SRV-ZBX)
Paramètre de l'agent	`Server=` (adresse du serveur autorisé)	`ServerActive=` et `Hostname=`
Modèles	« Linux by Zabbix agent », « Windows by Zabbix agent »	« … by Zabbix agent active »
À privilégier	Réseau simple, hôtes joignables depuis le serveur	Hôtes derrière un pare-feu ou NAT, auto-enregistrement

Avec Docker, le serveur joint les agents depuis son conteneur : le trafic sort par SRV-ZBX (contrôles passifs). Les agents actifs, eux, contactent le port **10051 publié** de SRV-ZBX : c'est la raison de la ligne `ports` du serveur dans `compose.yaml`.

B – Hôte Linux (srv-lnx-01) : déclaration manuelle

Connectez-vous à la machine Debian **srv-lnx-01** (192.168.20.61) :

1. Ajoutez le dépôt officiel de Zabbix (paquet `zabbix-release` pour **8.0** et Debian 13), puis installez l'agent :

```
TERMINAL (srv-lnx-01)
sudo apt install -y wget
wget https://repo.zabbix.com/zabbix/8.0/release/debian/pool/main/z/\
zabbix-release/zabbix-release_latest_8.0+debian13_all.deb
sudo dpkg -i zabbix-release_latest_8.0+debian13_all.deb
sudo apt update && sudo apt install -y zabbix-agent2
```

2. Renseignez l'adresse du serveur et le nom de l'hôte dans le fichier de configuration de l'agent, puis redémarrez-le :

```
TERMINAL (srv-lnx-01)
sudo sed -i -e 's/^Server=127.0.0.1/Server=192.168.20.60/' \
  -e 's/^ServerActive=127.0.0.1/ServerActive=192.168.20.60/' \
  -e 's/^Hostname=Zabbix server/Hostname=srv-lnx-01/' \
  /etc/zabbix/zabbix_agent2.conf
grep -E '^(Server|ServerActive|Hostname)=' /etc/zabbix/zabbix_agent2.conf
sudo systemctl enable --now zabbix-agent2
sudo systemctl restart zabbix-agent2
```

REMARQUE IMPORTANTE

La valeur de `Hostname` doit être **identique, à la majuscule près**, au « Host name » saisi dans Zabbix : c'est ainsi que le serveur reconnaît l'hôte en mode actif. Une différence est la première cause d'hôte qui ne remonte pas.

Avant de passer à l'interface, testez l'agent **depuis SRV-ZBX** avec l'outil `zabbix_get` (le paquet de Debian suffit pour ce test) :

```
TERMINAL (SRV-ZBX)
sudo apt install -y zabbix-get
zabbix_get -s 192.168.20.61 -k agent.ping
```

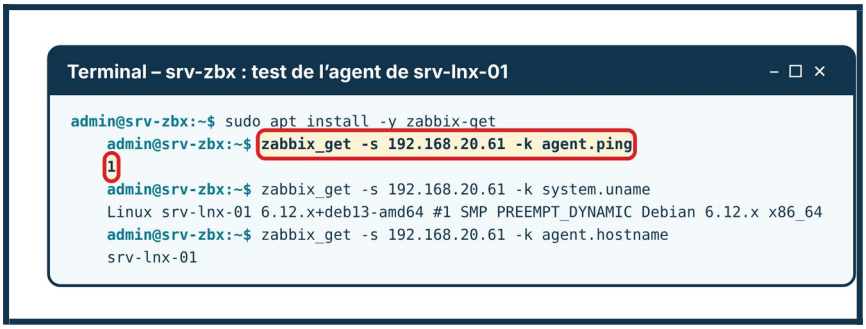


Figure 12 – Test de l'agent de srv-lnx-01 depuis SRV-ZBX

À SAVOIR

La réponse 1 prouve que le réseau, le pare-feu et le paramètre Server sont corrects. « Connection refused » : agent arrêté ou pare-feu ; « Timeout » : réseau ; « Not permitted » : le paramètre Server ne contient pas 192.168.20.60.

Déclarons maintenant l'hôte dans l'interface web de Zabbix :

1. Cliquez le menu « **Data collection** » puis « **Hosts** » et le bouton « **Create host** » (en haut à droite)
2. Saisissez le nom `srv-lnx-01`, choisissez le modèle « **Linux by Zabbix agent** », le groupe d'hôtes « **Linux servers** » et ajoutez l'interface « **Agent** » avec l'adresse `192.168.20.61` et le port `10050`, puis cliquez « **Add** » :



Figure 13 – Data collection > Hosts : création de l'hôte srv-lnx-01

À SAVOIR

Le groupe d'hôtes sert à ranger les machines et à filtrer les droits. Pour en créer un nouveau, tapez son nom dans le champ « Host groups ». Le modèle apporte à lui seul les mesures et les seuils.

3. Cliquez le menu « **Monitoring** » puis « **Hosts** » : après une à deux minutes, la pastille « **ZBX** » de **srv-lnx-01** passe au **vert** :

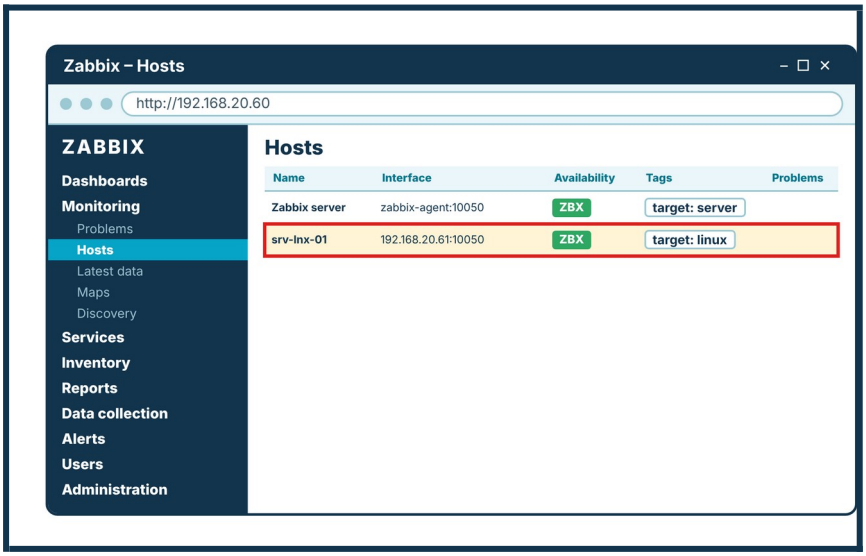


Figure 14 – Monitoring > Hosts : srv-lnx-01 est disponible

À SAVOIR

Pastille grise : aucune réponse encore reçue ; rouge : l'agent est injoignable (survolez-la pour lire l'erreur).

C – Serveur Windows (SRV-DC1) : déclaration manuelle

Sur le serveur **SRV-DC1** (192.168.20.250), ouvrez une session avec le compte `SIO\Administrateur` :

1. Téléchargez l'**agent Zabbix 2 pour Windows** (installateur **MSI**, 64 bits, version 8.0) depuis la page « Download Zabbix agents » du site `zabbix.com`, puis lancez le fichier `zabbix_agent2-8.0.x-windows-amd64-openssl.msi`
2. Acceptez la licence puis, à la page de configuration, saisissez le nom de l'hôte `SRV-DC1`, l'adresse du serveur `192.168.20.60` (champ « Zabbix server IP/DNS ») et la même adresse pour les contrôles actifs :

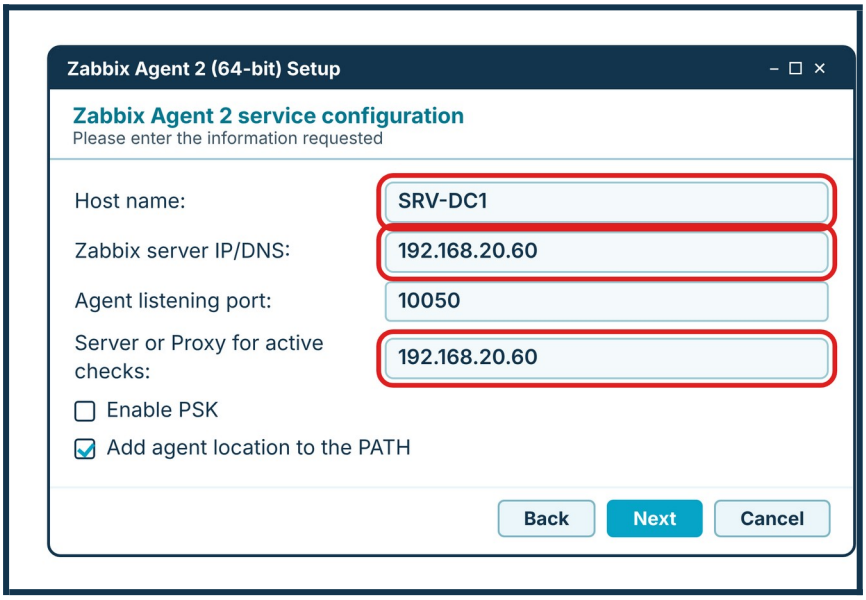


Figure 15 – Installateur de l'agent Zabbix 2 sous Windows

À SAVOIR

Gardez le port d'écoute 10050. L'installateur crée le service « Zabbix Agent 2 » (démarrage automatique) et enregistre ces valeurs dans `C:\Program Files\Zabbix Agent 2\zabbix_agent2.conf`.

3. Cliquez « **Next** » puis « **Install** » et « **Finish** ». Ouvrez PowerShell en administrateur, vérifiez le service et autorisez le port 10050 **depuis le serveur Zabbix uniquement** :

```powershell
Get-Service "Zabbix Agent 2" | Select-Object Name, Status, StartType
New-NetFirewallRule -DisplayName "Zabbix Agent 2 (10050)" -Direction Inbound `
    -Protocol TCP -LocalPort 10050 -RemoteAddress 192.168.20.60 -Action Allow
```

4. Sur **SRV-ZBX**, testez l'agent : `zabbix_get -s 192.168.20.250 -k agent.ping` doit répondre `1`
5. Dans Zabbix, « **Data collection** » puis « **Hosts** » puis « **Create host** » : renseignez les valeurs du tableau ci-dessous, puis cliquez « **Add** » :

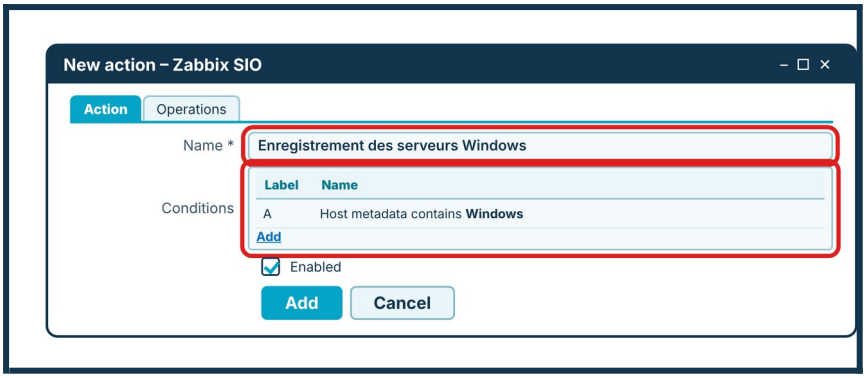
Champ	Valeur pour SRV-DC1
Host name	`SRV-DC1` (identique au nom saisi dans l'installateur)
Templates	« **Windows by Zabbix agent** » (contrôle passif)
Host groups	`Windows servers` (nouveau groupe : tapez son nom)
Interfaces	Agent, adresse IP `192.168.20.250`, port `10050`

D – Serveur Windows (SRV-DC2) : auto-enregistrement

Déclarer chaque hôte à la main devient pénible dès qu'on a des dizaines de machines. L'**auto-enregistrement** inverse le principe : c'est **l'agent actif qui se présente** au serveur, et une **action** crée l'hôte, le range dans un groupe et lui

associe un modèle. L'agent indique « qui il est » grâce à une valeur libre, la **métadonnée d'hôte** (`HostMetadata`). Commençons par créer l'action dans Zabbix :

1. Cliquez le menu « **Alerts** » puis « **Actions** » puis « **Autoregistration actions** » et le bouton « **Create action** »
2. Onglet « **Action** » : saisissez le nom `Enregistrement des serveurs Windows` puis, sous « **Conditions** », cliquez « **Add** » et choisissez « **Host metadata** », l'opérateur « **contains** » et la valeur `Windows` :

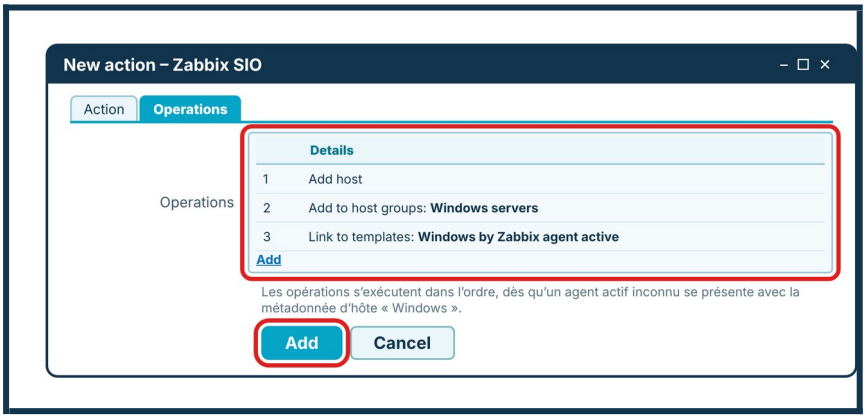


À SAVOIR

La condition filtre les agents : seuls ceux dont la métadonnée contient « Windows » déclenchent cette action. On peut créer une seconde action pour les hôtes Linux (métadonnée « Linux », modèle « Linux by Zabbix agent active »).

Figure 16 – Action d'auto-enregistrement : nom et condition

3. Onglet « **Operations** » : ajoutez trois opérations (bouton « **Add** » de la liste) : « **Add host** », « **Add to host groups** » avec le groupe `Windows servers`, et « **Link to templates** » avec « **Windows by Zabbix agent active** ». Cliquez « **Add** » pour enregistrer l'action :



À SAVOIR

Le modèle « active » convient ici : l'agent actif est justement celui qui se présente. Aucune règle de pare-feu entrante n'est nécessaire sur l'hôte.

Figure 17 – Action d'auto-enregistrement : opérations

Installons maintenant l'agent sur **SRV-DC2** (192.168.20.251) **en ligne de commande**, sans assistant : copiez le fichier MSI dans `C:\Temp` puis, dans PowerShell en administrateur, indiquez les paramètres de l'agent, **dont la métadonnée** :

POWERSHELL

```
msiexec /i "C:\Temp\zabbix_agent2-8.0.x-windows-amd64-openssl.msi" /qn `
    SERVER=192.168.20.60 SERVERACTIVE=192.168.20.60 `
    HOSTNAME=SRV-DC2 HOSTMETADATA=Windows
Get-Service "Zabbix Agent 2" | Select-Object Name, Status
```

1. Patientez une à deux minutes : l'agent contacte le port **10051** de SRV-ZBX (publié par Docker), l'action s'exécute et l'hôte **SRV-DC2** apparaît tout seul dans « **Monitoring** » puis « **Hosts** » :

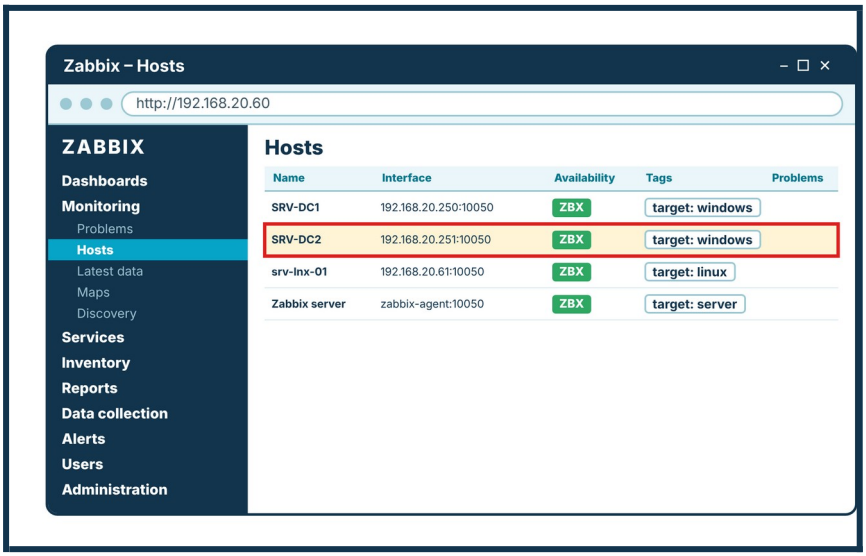


Figure 18 – Monitoring > Hosts : SRV-DC2 s'est enregistré automatiquement

À SAVOIR

L'hôte est créé avec l'adresse d'où provient l'agent et le port 10050 : vous n'avez rien saisi dans l'interface. La même commande, répétée sur 50 serveurs (script, GPO, outil de déploiement), les fait tous remonter.

À SAVOIR

Sous Linux, le principe est identique : ajoutez `HostMetadata=Linux` dans `/etc/zabbix/zabbix_agent2.conf`, redémarrez l'agent et créez l'action correspondante. Pour un même hôte, ne mélangez pas déclaration manuelle et auto-enregistrement : le nom de l'hôte doit être unique dans Zabbix.

✔ **POINT DE CONTRÔLE**

- **srv-lnx-01** (déclaré à la main) et **SRV-DC1** (déclaré à la main) ont une pastille **ZBX verte**
- **SRV-DC2** est apparu **sans** création manuelle, avec le groupe et le modèle de l'action
- Le `Hostname` de chaque agent est **identique** au nom de l'hôte dans Zabbix

06 ÉTAPE 6 SUR 6
Exploiter : sauvegarder, mettre à jour, dépanner

A – Consulter les mesures collectées

1 Cliquez le menu « **Monitoring** » puis « **Latest data** » et, dans le filtre « **Hosts** », sélectionnez **SRV-DC1** puis cliquez « **Apply** » :

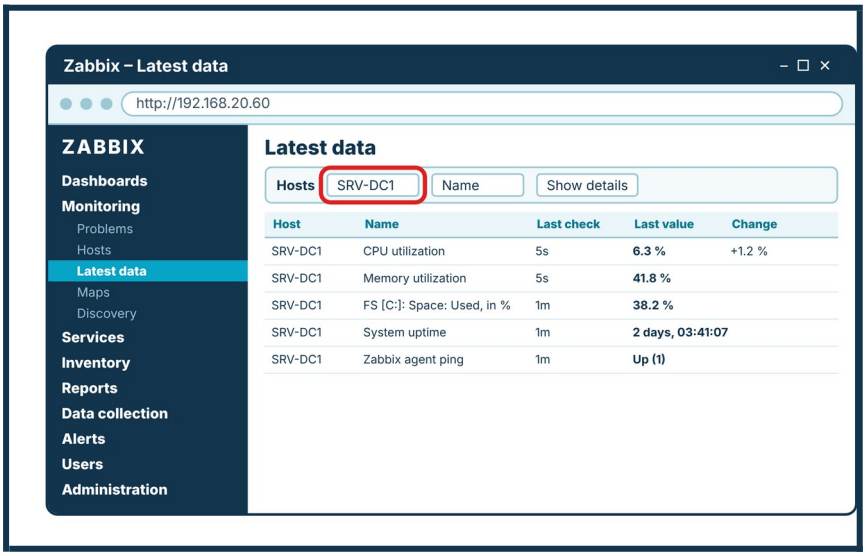


Figure 19 – Monitoring > Latest data : dernières valeurs de SRV-DC1

À SAVOIR

Les premières valeurs arrivent en quelques secondes. Les éléments découverts automatiquement (disques, cartes réseau, services) peuvent mettre quelques minutes à apparaître. Cliquez sur « Graph » en face d'une mesure pour la voir évoluer.

B – Provoquer une panne et voir l'alerte

Le meilleur test est une panne volontaire. Sur **srv-lnx-01**, arrêtez l'agent :

1 Saisissez `sudo systemctl stop zabbix-agent2`

2 Patientez environ **quatre minutes** : le seuil du modèle est de trois minutes sans réponse. Cliquez le menu « **Monitoring** » puis « **Problems** » : un problème de sévérité « **Average** » apparaît :

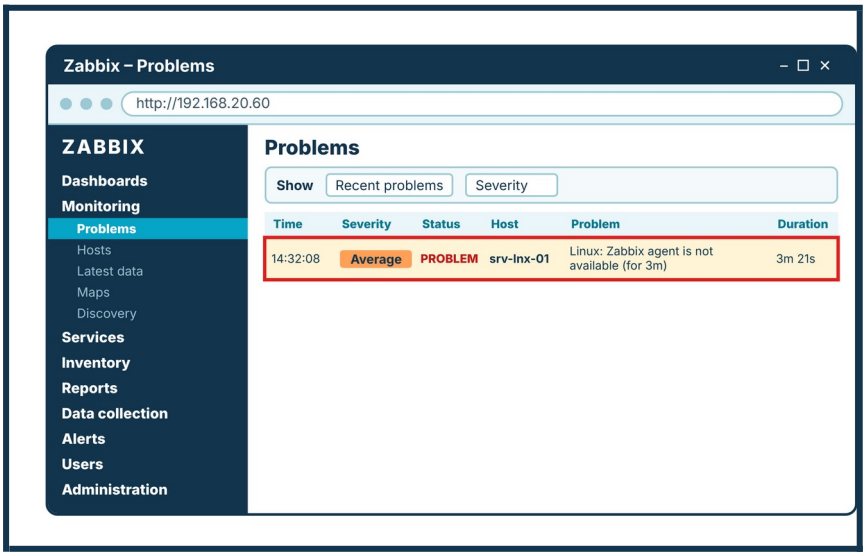


Figure 20 – Monitoring > Problems : agent injoignable

À SAVOIR

Le nom du problème vient du déclencheur du modèle. La couleur indique la sévérité (Information, Warning, Average, High, Disaster). Dans « Monitoring > Hosts », la pastille ZBX de l'hôte est devenue rouge.

3. Redémarrez l'agent : `sudo systemctl start zabbix-agent2` ; une à deux minutes plus tard, le problème passe à « **RESOLVED** »

C – Sauvegarder

Avec Docker, **trois choses** sont à sauvegarder : la **base de données** (dans le volume `db-data`), et le dossier du projet (`compose.yaml` et `.env`, qui contient les mots de passe). Les images, elles, se retéléchargent. Exportez la base **à chaud** (sans arrêter Zabbix) puis archivez le projet :

TERMINAL

```
cd /opt/zabbix
sudo docker compose exec -T mysql-server sh -c \
  'mysqldump --single-transaction --routines --triggers -uroot \
   -p"$MYSQL_ROOT_PASSWORD" zabbix' | gzip | \
  sudo tee /var/backups/zabbix-$(date +%F).sql.gz > /dev/null
sudo tar czf /var/backups/zabbix-projet-$(date +%F).tar.gz -C /opt zabbix
```

Pour **restaurer** sur un serveur neuf : installez Docker, recréez le dossier `/opt/zabbix` depuis l'archive, démarrez **uniquement la base** (`sudo docker compose up -d mysql-server`), importez le fichier (`gunzip -c fichier.sql.gz | sudo docker compose exec -T mysql-server sh -c 'mysql -uroot -p"$MYSQL_ROOT_PASSWORD" zabbix'`) puis démarrez le reste (`sudo docker compose up -d`). Copiez les sauvegardes **hors du serveur** (partage réseau, serveur de sauvegarde).

D – Mettre à jour

1. Avec le suffixe `latest` de la branche 8.0 (`alpine-8.0-latest`), les corrections de la **8.0.x** s'appliquent en téléchargeant les nouvelles images et en recréant les conteneurs ; les volumes sont conservés. **Sauvegardez d'abord**, puis :

TERMINAL

```
cd /opt/zabbix
sudo docker compose pull
sudo docker compose up -d
sudo docker image prune -f
```

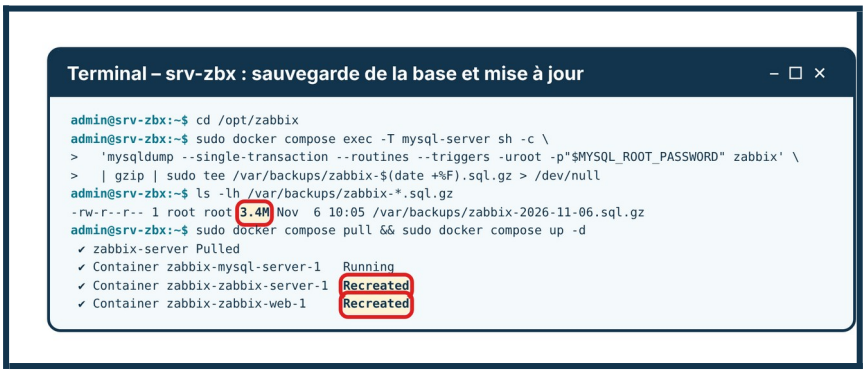


Figure 21 – Sauvegarde de la base puis mise à jour des conteneurs

À SAVOIR

Seuls les conteneurs dont l'image a changé sont recréés. Le serveur Zabbix met la base à jour tout seul à son démarrage : suivez son journal. Pour un changement de version **majeure** (par exemple 8.0 vers 8.2), lisez d'abord les notes de version puis modifiez `ZBX_TAG` dans `.env`.

2. Arrêt et suppression : `sudo docker compose down` supprime les **conteneurs** et le réseau mais **garde les volumes** (la base). N'ajoutez l'option `-v` que pour **effacer définitivement** les données

✔ POINT DE CONTRÔLE

- Les valeurs de **SRV-DC1** apparaissent dans « Latest data »
- L'arrêt de l'agent de **srv-lnx-01** crée un **problème** visible dans « Problems », puis il est résolu
- Une **sauvegarde** de la base et du dossier `/opt/zabbix` existe
- `docker compose pull` puis `up -d` met à jour la pile **sans perte de données**

Notre supervision est **opérationnelle** : Zabbix 8.0 tourne dans quatre conteneurs sur Debian 13, les agents remontent les mesures des serveurs Linux et Windows, et la pile se sauvegarde et se met à jour en quelques commandes.

Dépannage : les symptômes les plus fréquents

Symptôme	Cause probable	Solution
« permission denied … Docker daemon socket »	Commande `docker` lancée sans `sudo`	Utiliser `sudo docker …` (ou se connecter en root)
« Bind for 0.0.0.0:80 failed: port is already allocated »	Un autre service (Apache, nginx…) occupe déjà le port 80	`sudo ss -tlnp | grep ':80 '` puis arrêter ce service, ou publier un autre port (`"8080:8080"`)
Un conteneur est en « Restarting »	Erreur de configuration ou base inaccessible	`sudo docker compose logs --tail 30 <service>` ; vérifier `.env` et l'indentation de `compose.yaml`
L'interface affiche « Zabbix server is not running »	Serveur encore en initialisation, arrêté, ou base non prête	Attendre 1 à 2 minutes ; `sudo docker compose ps` ; journal de `zabbix-server`
Mot de passe de la base modifié dans `.env` sans effet	MySQL ne relit ces variables qu'à la création du volume	Changer le mot de passe dans MySQL (`ALTER USER`) puis `.env` ; sans données à garder : `docker compose down -v` et redémarrer
L'hôte « Zabbix server » est indisponible (ZBX rouge)	L'interface de l'agent pointe encore sur 127.0.0.1	Étape 4-C : nom DNS `zabbix-agent`, mode « DNS »
Pastille ZBX grise ou rouge (mode passif)	Agent arrêté, port 10050 bloqué ou `Server=` incorrect	`zabbix_get -s <IP> -k agent.ping` depuis SRV-ZBX ; contrôler le pare-feu et le paramètre `Server`
Hôte actif qui ne remonte pas, ou qui ne s'enregistre pas	Port 10051 inaccessible, `Hostname` différent (casse), condition d'action fausse	Vérifier `0.0.0.0:10051` dans `docker compose ps` ; lire le journal de l'agent ; contrôler la métadonnée et l'état « Enabled » de l'action
Le disque se remplit	Anciennes images, historique des mesures	`sudo docker system df` ; `sudo docker image prune -f` ; ajuster la durée de conservation de l'historique

Pense-bête : commandes et emplacements

Commande / emplacement	Utilité
`sudo docker compose ps`	État des services du projet et ports publiés
`sudo docker compose logs -f <service>`	Journal d'un conteneur en direct (Ctrl + C pour quitter)
`sudo docker compose up -d`	Crée ou met à jour les conteneurs d'après `compose.yaml`
`sudo docker compose restart <service>`	Redémarre un conteneur sans le recréer
`sudo docker compose exec <service> sh`	Ouvre un terminal dans un conteneur en marche
`sudo docker compose down`	Supprime conteneurs et réseau (les volumes sont conservés)
`/opt/zabbix/compose.yaml` **et** `.env`	Configuration du projet : à sauvegarder, `.env` reste confidentiel
`sudo docker volume ls`	Volumes existants (`zabbix_db-data` contient la base)
Monitoring > Hosts / Problems	Disponibilité des agents et problèmes en cours
Alerts > Actions	Auto-enregistrement, notifications et actions sur déclencheur