PortailSIO

NOUVELLE FORMULE

TUTORIEL

DOCKER COMPOSE · GLPI 12.0.1

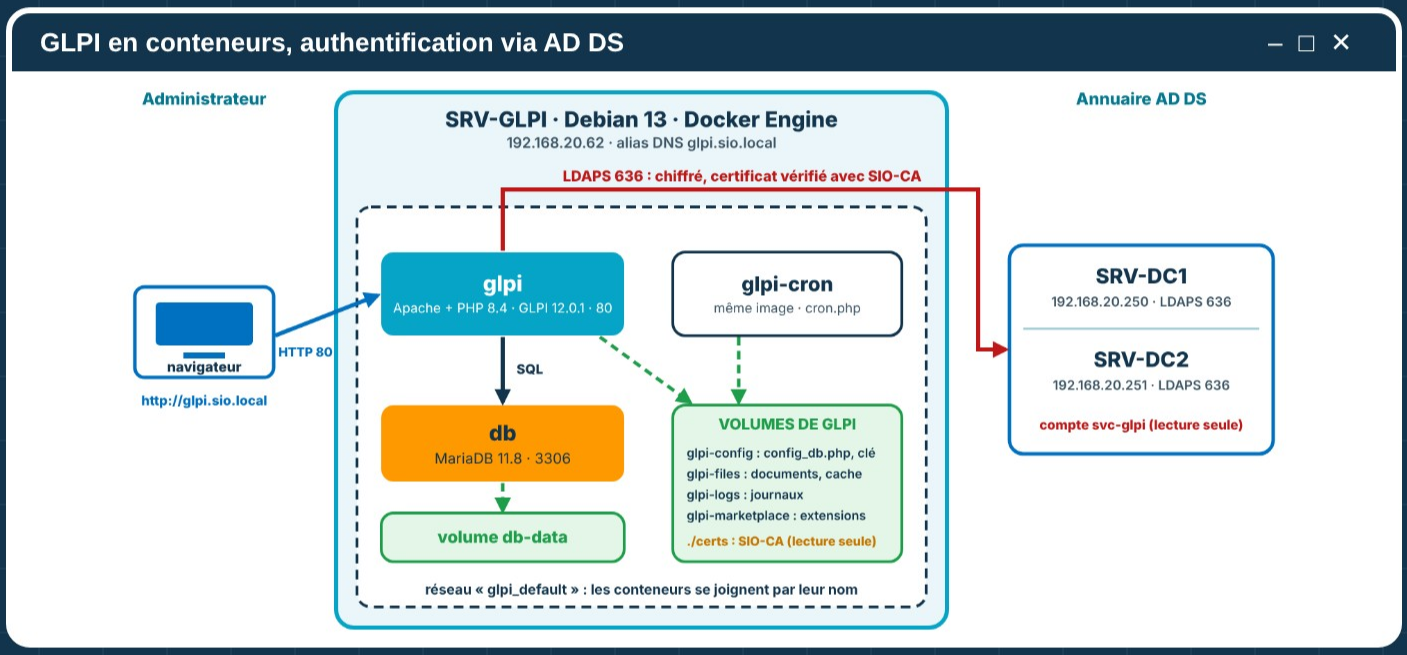
Installer avec Docker Compose

GLPI



NIVEAU

Intermédiaire



1. Préparer le serveur Debian 13 et le DNS
2. Installer Docker Engine et Docker Compose
3. Construire l'image GLPI et écrire le projet Compose
4. Démarrer la pile et terminer l'installation
5. Préparer AD DS : compte de service et autorité de certification
6. Lier GLPI à l'annuaire et importer les utilisateurs

CC BY-NC

Utilisation commerciale interdite

© PortailSIO – 11/2026

Ce tutoriel en ligne

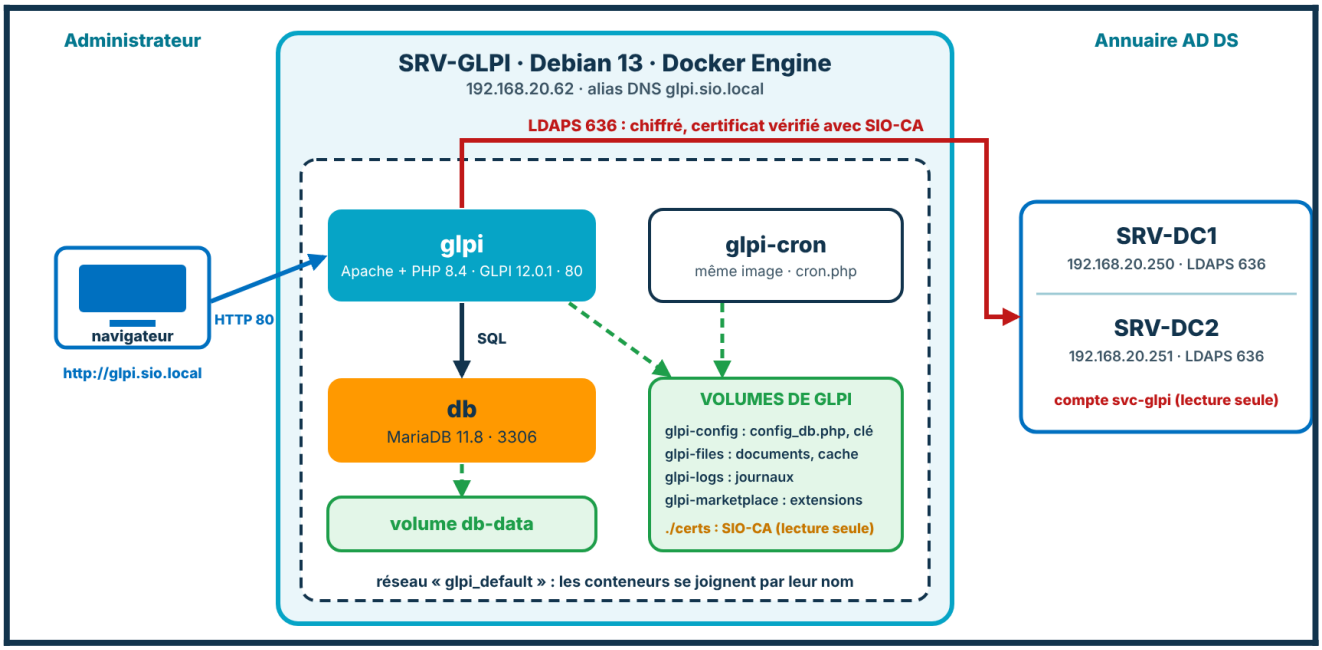
Scannez le QR code. Tous les tutoriels :

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 du domaine **sio.local** construit dans les tutoriels précédents (contrôleurs **SRV-DC1** et **SRV-DC2** sous Windows Server 2025, avec l'autorité de certification **SIO-CA** et **LDAPS** activé : voir le tutoriel « Activer LDAPS sur AD DS »).

GLPI est une solution libre de **gestion de parc informatique** et de **gestion des demandes** (ITSM) : inventaire du matériel et des logiciels, tickets d'assistance, contrats, licences… C'est l'outil de référence des services informatiques et des épreuves du BTS SIO. La version **12.0.1** s'exécute sur **PHP 8.3 ou plus**.

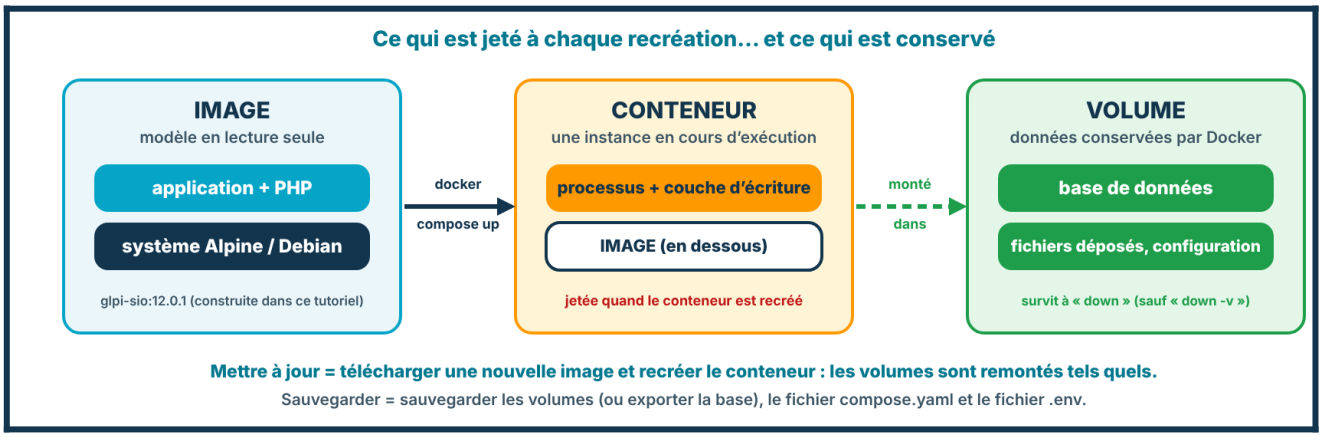
Dans ce tutoriel, nous installons GLPI **avec Docker Compose** : Apache et PHP, la base de données et les tâches automatiques s'exécutent chacun dans leur **conteneur**, et toute l'installation se décrit dans quelques fichiers. Pour comprendre chaque pièce et obtenir une image qui contient l'extension LDAP et les réglages du tutoriel « natif », nous **construisons notre propre image** à partir de l'image officielle de PHP. Nous lions ensuite GLPI à l'annuaire **AD DS** pour que les utilisateurs se connectent avec leur compte Windows. Voici l'architecture obtenue :



Les trois conteneurs

- **glpi** : Apache et PHP 8.4 avec le programme GLPI 12.0.1 ; il publie le port **80** du serveur et ne sert que le sous-dossier `public` de GLPI
- **glpi-cron** : la **même image**, qui exécute `cron.php` chaque minute (notifications, nettoyages, inventaire…) ; le conteneur remplace la tâche `cron` du système
- **db** : MariaDB 11.8, qui stocke toutes les données ; son port 3306 reste **invisible** du réseau, seuls les autres conteneurs l'atteignent par le nom `db`
- **AD DS** (SRV-DC1 et son réplicat SRV-DC2) authentifie les utilisateurs par une connexion **chiffrée LDAPS**, avec un compte de service en lecture seule ; le certificat de l'autorité **SIO-CA** est fourni aux conteneurs par un dossier monté en lecture seule

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. C'est ce qui permet de supprimer et de recréer le conteneur (pour une mise à jour, par exemple) sans rien perdre :



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-GLPI	Serveur Docker exécutant GLPI (nom `srv-glpi`, alias DNS `glpi.sio.local`)	Debian 13	192.168.20.62
SRV-DC1	Contrôleur de domaine, DNS, autorité de certification SIO-CA	Windows Server 2025	192.168.20.250
SRV-DC2	Contrôleur de domaine, DNS, annuaire de secours	Windows Server 2025	192.168.20.251
Poste admin	Navigateur web pour utiliser GLPI (DNS = les deux contrôleurs)	Windows 11	DHCP

Pour **SRV-GLPI**, 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 Debian sont exécutées avec un compte autorisé à utiliser `sudo` (ou directement en `root`, sans le préfixe `sudo`) ; les commandes Windows, dans **PowerShell** ouvert **en tant qu'administrateur** sur **SRV-DC1**.

ÉTAPE 1 SUR 6

01 Préparer le serveur Debian 13 et le DNS

Docker n'a besoin que d'un système à jour, d'un nom, d'une adresse fixe et de l'heure exacte. On déclare aussi le nom **glpi.sio.local** dans le DNS du domaine. Sur **SRV-GLPI** :

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 complet et nom court) :

TERMINAL

```
sudo hostnamectl set-hostname srv-glpi
echo "192.168.20.62  glpi.sio.local  srv-glpi" | 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 192.168.20.62/24
    gateway 192.168.20.254
```

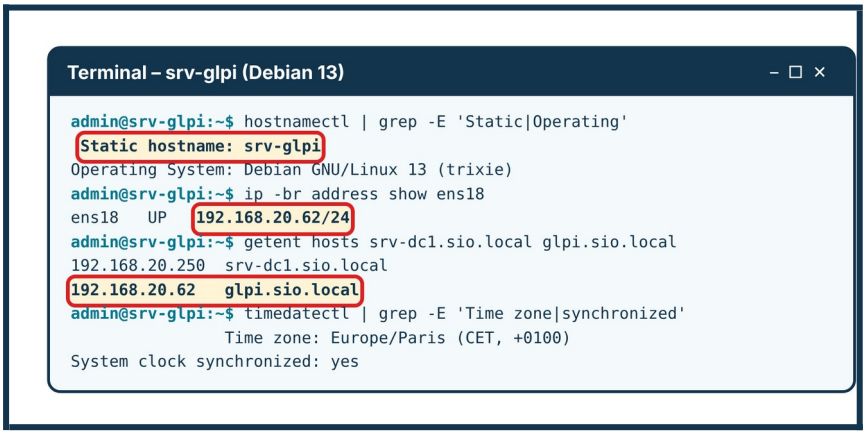
5 Indiquez les **DNS du domaine** dans `/etc/resolv.conf` : les conteneurs s'en servent pour retrouver `srv-dc1.sio.local` et `srv-dc2.sio.local` :

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

6 Sur **SRV-DC1**, déclarez l'alias DNS de GLPI (l'enregistrement est répliqué sur SRV-DC2 avec la zone) :

```
POWERSHELL
Add-DnsServerResourceRecordA -Name "glpi" -ZoneName "sio.local" `
    -IPv4Address "192.168.20.62"
```

7 Vérifiez le résultat : nom, adresse IP, résolution DNS des contrôleurs et de GLPI, heure :



À SAVOIR

Si « System clock synchronized » indique « no », installez un service de temps (`sudo apt install -y chrony`). L'heure doit être juste : le certificat LDAPS de l'étape 5 a des dates de validité.

Figure 1 – Contrôle de la préparation du serveur

✔ **POINT DE CONTRÔLE**

- `hostnamectl` affiche le nom **srv-glpi** et Debian GNU/Linux **13**
- L'adresse IP **192.168.20.62** est fixe
- `getent hosts srv-dc1.sio.local glpi.sio.local` renvoie les deux adresses
- Le fuseau horaire est **Europe/Paris** et l'horloge est **synchronisée**

02 ÉTAPE 2 SUR 6 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
```

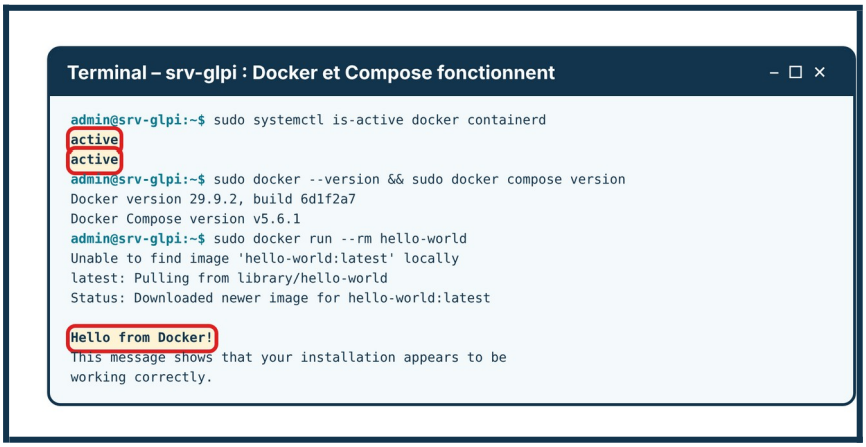


Figure 3 – Docker et Compose en état de marche

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!** »

ÉTAPE 3 SUR 6

03 Construire l'image GLPI et écrire le projet Compose

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). Le sous-dossier `glpi` contient de quoi **construire l'image** de GLPI ; le sous-dossier `initdb`, un script exécuté à la première création de la base.

A – Créer le dossier et le fichier d'environnement

1 Créez l'arborescence du projet (le dossier `certs` recevra le certificat de SIO-CA à l'étape 5), puis le fichier `.env` avec `sudo nano /opt/glpi/.env` :

TERMINAL

```
sudo mkdir -p /opt/glpi/glpi /opt/glpi/initdb /opt/glpi/certs
cd /opt/glpi
```

/opt/glpi/.env

```
GLPI_VERSION=12.0.1
MARIADB_TAG=11.8
TZ=Europe/Paris
DB_PASSWORD=Glpi_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/glpi/.env
```

REMARQUE IMPORTANTE

`Glpi_SIO_2026` et `Root_SIO_2026` ne sont que des exemples : **choisissez deux mots de passe solides et différents**. Attention : MariaDB 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
`GLPI_VERSION`	Version de GLPI téléchargée à la construction de l'image ; elle donne aussi son nom à l'image (`glpi-sio:12.0.1`)
`MARIADB_TAG`	Version de MariaDB (11.8 est la version fournie par Debian 13 et prise en charge par GLPI 12)
`TZ`	Fuseau horaire appliqué aux conteneurs
`DB_PASSWORD`, `DB_ROOT_PASSWORD`	Mot de passe de l'utilisateur `glpi` de la base et mot de passe de l'administrateur MariaDB

B – Écrire le Dockerfile : la recette de l'image

Un **Dockerfile** décrit, ligne après ligne, comment fabriquer une image : on part d'une image de base, on installe ce qu'il faut, on copie ses fichiers. C'est l'équivalent, en une fois et de façon reproductible, des étapes 2 et 3 du tutoriel « natif » (extensions PHP, téléchargement de GLPI, séparation du code, de la configuration et des données). Créez `sudo nano /opt/glpi/glpi/Dockerfile` :

/opt/glpi/glpi/Dockerfile

```
FROM php:8.4-apache
ARG GLPI_VERSION
ARG GLPI_URL=https://github.com/glpi-project/glpi/releases/download

# 1. Bibliothèques et extensions PHP demandées par GLPI
RUN apt-get update && apt-get install -y --no-install-recommends \
      libicu-dev libldap-dev libzip-dev libbz2-dev libpng-dev \
      libjpeg-dev libfreetype6-dev wget ca-certificates \
 && docker-php-ext-configure gd --with-jpeg --with-freetype \
 && docker-php-ext-install -j"$(nproc)" \
      mysqli gd intl ldap zip bz2 exif opcache bcmath \
 && a2enmod rewrite \
 && rm -rf /var/lib/apt/lists/*

# 2. Programme GLPI, séparé de sa configuration et de ses données
RUN wget -q -O /tmp/glpi.tgz \
      ${GLPI_URL}/${GLPI_VERSION}/glpi-${GLPI_VERSION}.tgz \
 && tar -xzf /tmp/glpi.tgz -C /var/www && rm /tmp/glpi.tgz \
 && mv /var/www/glpi/config /etc/glpi \
 && mv /var/www/glpi/files /var/lib/glpi \
 && mkdir /var/log/glpi

# 3. Réglages : emplacement des dossiers, Apache et PHP
COPY downstream.php /var/www/glpi/inc/downstream.php
COPY local_define.php /etc/glpi/local_define.php
COPY glpi.conf /etc/apache2/sites-available/glpi.conf
COPY php.ini /usr/local/etc/php/conf.d/90-glpi.ini

# 4. Droits (code à root, données à www-data) et site Apache
RUN chown -R root:root /var/www/glpi \
 && chown -R www-data:www-data /etc/glpi /var/lib/glpi /var/log/glpi \
      /var/www/glpi/marketplace \
 && a2dissite 000-default && a2ensite glpi
```

Instruction	Rôle
`FROM php:8.4-apache`	Image de base officielle : Debian, Apache et **PHP 8.4** (le module PHP est intégré à Apache, pas de PHP-FPM)
`ARG`	Valeurs fournies à la construction : la version de GLPI vient du fichier `.env`
`docker-php-ext-install`	Compile les extensions PHP de GLPI : base (`mysqli`), images (`gd`), langues (`intl`), **annuaire (`ldap`)**, archives, calculs. Les autres (curl, mbstring, xml…) sont déjà dans l'image
`RUN wget … && mv …`	Télécharge GLPI puis déplace `config` et `files` **hors du site web** : ils deviendront des volumes
`COPY`	Copie dans l'image les quatre petits fichiers de réglage écrits au point suivant
`chown`	Le code appartient à **root** (le site ne peut pas se modifier lui-même) ; configuration, données, journaux et extensions à **www-data**

C – Écrire les fichiers de réglage de l'image

Ces quatre fichiers sont les mêmes que ceux du tutoriel natif, placés dans le dossier `/opt/glpi/glpi`. Le premier indique à GLPI où est sa configuration (`sudo nano /opt/glpi/glpi/downstream.php`) :

/opt/glpi/glpi/downstream.php

```
<?php
define('GLPI_CONFIG_DIR', '/etc/glpi/');
if (file_exists(GLPI_CONFIG_DIR . '/local_define.php')) {
    require_once GLPI_CONFIG_DIR . '/local_define.php';
}
```

Le deuxième indique l'emplacement des fichiers déposés et des journaux (`sudo nano /opt/glpi/glpi/local_define.php`) :

/opt/glpi/glpi/local_define.php

```
<?php
define('GLPI_VAR_DIR', '/var/lib/glpi');
define('GLPI_LOG_DIR', '/var/log/glpi');
```

Le troisième est le site Apache : il ne publie que le dossier `public` de GLPI. Les journaux sont envoyés vers la **sortie standard** du conteneur, que Docker rassemble (`docker compose logs`) :

/opt/glpi/glpi/glpi.conf

```
<VirtualHost *:80>
    ServerName glpi.sio.local
    DocumentRoot /var/www/glpi/public
    <Directory /var/www/glpi/public>
        Require all granted
        RewriteEngine On
        RewriteCond %{HTTP:Authorization} ^(.+)$
        RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteRule ^(.*)$ index.php [QSA,L]
    </Directory>
    ErrorLog /proc/self/fd/2
    CustomLog /proc/self/fd/1 combined
</VirtualHost>
```

Le dernier règle PHP : cookies de session protégés et limites confortables (documents joints jusqu'à 20 Mo) :

/opt/glpi/glpi/php.ini

```
session.cookie_httponly = On
session.cookie_samesite = Lax
memory_limit = 256M
upload_max_filesize = 20M
post_max_size = 20M
max_execution_time = 60
```

D – Écrire le script d'initialisation de la base

L'image de MariaDB exécute, **une seule fois** (à la création de la base), les scripts du dossier `/docker-entrypoint-initdb.d`. Le nôtre charge les **fuseaux horaires** (GLPI les utilise pour afficher les dates dans le bon fuseau) et donne à l'utilisateur `glpi` le droit de les lire. Créez `sudo nano /opt/glpi/initdb/10-glpi.sh` :

/opt/glpi/initdb/10-glpi.sh

```
#!/bin/sh
# Exécuté une seule fois, à la première création de la base
mariadb-tzinfo-to-sql /usr/share/zoneinfo \
    | mariadb -uroot -p"$MARIADB_ROOT_PASSWORD" mysql
mariadb -uroot -p"$MARIADB_ROOT_PASSWORD" <<SQL
GRANT SELECT ON mysql.time_zone_name TO '$MARIADB_USER'@'%';
FLUSH PRIVILEGES;
SQL
```

E – Écrire le fichier compose.yaml

Le fichier est écrit en **YAML** : l'**indentation** (deux espaces, jamais de tabulation) donne la structure. Un **bloc d'ancrage** (`x-glpi`) rassemble ce que partagent les deux conteneurs GLPI : l'image, l'environnement et les volumes. Créez `sudo nano /opt/glpi/compose.yaml` et commencez par ce bloc :

/opt/glpi/compose.yaml (1/3)

```
name: glpi

x-glpi: &glpi
  image: glpi-sio:${GLPI_VERSION}
  restart: unless-stopped
  environment:
    TZ: ${TZ}
    LDAPTLS_CACERT: /certs/sio-ca.crt
  volumes:
    - glpi-config:/etc/glpi
    - glpi-files:/var/lib/glpi
    - glpi-logs:/var/log/glpi
    - glpi-marketplace:/var/www/glpi/marketplace
    - ./certs:/certs:ro
```

Ajoutez à la suite la section `services:` avec la **base de données**. L'image de MariaDB crée la base et l'utilisateur à partir des variables `MARIADB_*` et exécute le script `initdb` une seule fois :

/opt/glpi/compose.yaml (2/3)

```
services:
  db:
    image: mariadb:${MARIADB_TAG}
    restart: unless-stopped
    environment:
      MARIADB_DATABASE: glpi
      MARIADB_USER: glpi
      MARIADB_PASSWORD: ${DB_PASSWORD}
      MARIADB_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
      TZ: ${TZ}
    volumes:
      - db-data:/var/lib/mysql
      - ./initdb:/docker-entrypoint-initdb.d:ro
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]
      interval: 10s
      timeout: 5s
      retries: 12
      start_period: 20s
```

Puis, à la suite dans le même fichier (au même niveau que `db`), les deux conteneurs GLPI et la déclaration des volumes :

/opt/glpi/compose.yaml (3/3)

```
  glpi:
    <<: *glpi
    build:
      context: ./glpi
      args:
        GLPI_VERSION: ${GLPI_VERSION}
    depends_on:
      db:
        condition: service_healthy
    ports:
      - "80:80"

  glpi-cron:
    <<: *glpi
    user: www-data
    entrypoint:
      - sh
      - -c
      - while true; do php /var/www/glpi/front/cron.php; sleep 60; done
    depends_on:
      - glpi

volumes:
  db-data:
  glpi-config:
  glpi-files:
  glpi-logs:
  glpi-marketplace:
```

Élément	Signification
`&glpi` et `<<: *glpi`	Ancre YAML : les deux conteneurs `glpi` et `glpi-cron` reprennent le même bloc (image, volumes, environnement)
`build:` et `image: glpi-sio:…`	Compose **construit** l'image avec le Dockerfile du dossier `glpi` et la nomme `glpi-sio:12.0.1` ; `glpi-cron` réutilise cette image sans la reconstruire
`depends_on` + `healthcheck`	GLPI ne démarre qu'une fois MariaDB **prête** (`healthcheck.sh` fait partie de l'image MariaDB)
`ports: "80:80"`	Publie le port **80** du conteneur sur le port **80** du serveur : c'est le **seul** port ouvert, la base reste invisible du réseau
`volumes:`	Cinq volumes nommés, gérés par Docker : base de données, configuration (avec la clé de chiffrement), fichiers déposés, journaux, extensions
`./certs:/certs:ro`	Dossier de l'hôte monté en **lecture seule** ; il contiendra `sio-ca.crt`, le certificat de l'autorité SIO-CA
`LDAPTLS_CACERT`	Variable lue par la bibliothèque LDAP de PHP : elle désigne le fichier des autorités de confiance pour les connexions LDAPS
`user: www-data` + `entrypoint`	Le conteneur `glpi-cron` n'exécute pas Apache : il lance `cron.php` toutes les 60 secondes, avec les droits de `www-data`

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 du point suivant repère la plupart des erreurs.

F – Valider le projet et construire l'image

1. Contrôlez la syntaxe, puis affichez les images utilisées (l'image construite `glpi-sio` apparaît deux fois, pour `glpi` et `glpi-cron`) :

TERMINAL

```
cd /opt/glpi
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 Construisez l'image (le téléchargement de GLPI et la compilation des extensions durent quelques minutes), puis vérifiez que l'extension **ldap** est bien présente :

TERMINAL

```
sudo docker compose build
sudo docker run --rm --entrypoint php glpi-sio:12.0.1 -m | grep -xE 'ldap|intl'
```

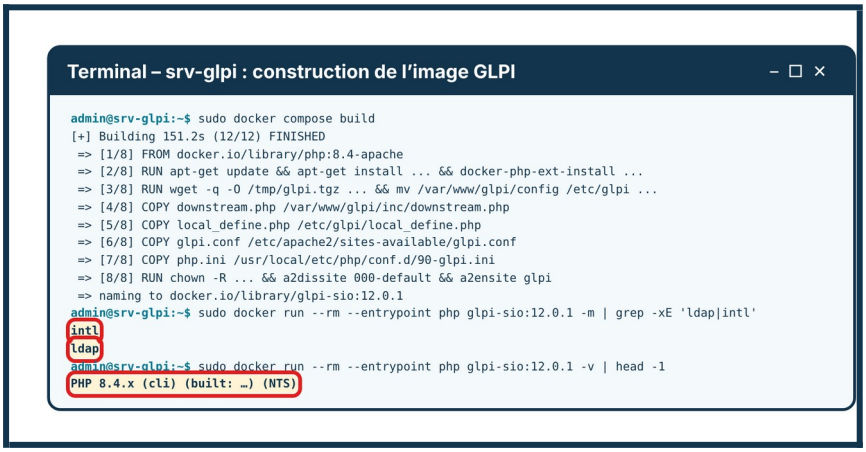


Figure 5 – Construction de l'image GLPI

À SAVOIR

Chaque ligne du Dockerfile devient une étape, mise en **cache** : si vous modifiez `php.ini`, seules les étapes suivantes sont refaites. Une erreur dans l'étape 3 (téléchargement) indique le plus souvent un DNS qui ne relaie pas vers Internet (voir le dépannage).

✔ **POINT DE CONTRÔLE**

- `/opt/glpi/.env` n'est lisible que par **root** (droits 600) et contient des mots de passe personnalisés
- Le dossier `/opt/glpi` contient `compose.yaml`, `glpi/` (cinq fichiers), `initdb/` et `certs/`
- `docker compose config --quiet` ne renvoie **aucune erreur**
- `docker compose build` se termine sans erreur et l'extension **ldap** est listée

04 ÉTAPE 4 SUR 6 Démarrer la pile et terminer l'installation

A – Démarrer les conteneurs

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

TERMINAL

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

Compose crée le réseau et les cinq volumes, démarre **MariaDB en premier** ; quand celle-ci est « healthy » (la première fois, elle charge aussi les fuseaux horaires, une trentaine de secondes), il démarre `glpi` puis `glpi-cron`.

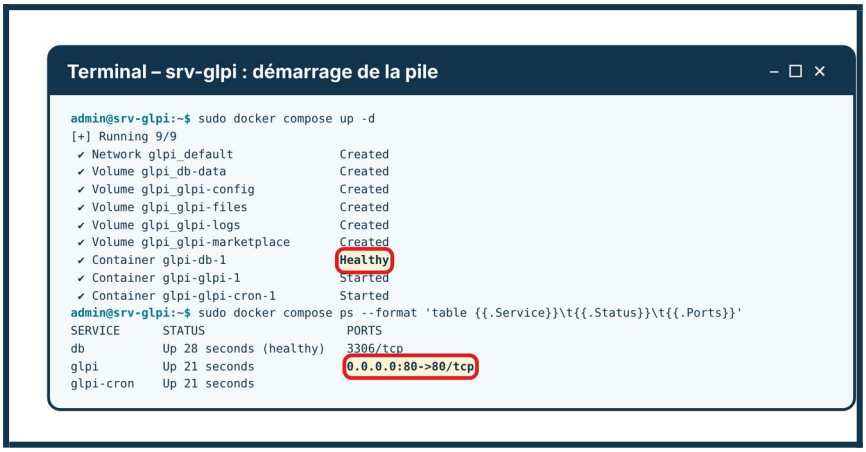


Figure 6 – Démarrage de la pile et état des services

À SAVOIR

Seul le port 80 est publié ; 3306 (MariaDB) n'est accessible qu'aux autres conteneurs. Si un conteneur indique « Restarting », lisez son journal : `sudo docker compose logs --tail 30 glpi`.

B – Dérouler l'assistant d'installation

Depuis le **poste d'administration**, ouvrez un navigateur et saisissez `http://glpi.sio.local`. L'assistant d'installation se lance :

1 Choisissez la langue « **Français** » puis cliquez « **OK** » ; acceptez la licence en cliquant « **Continuer** » :



Figure 7 – Assistant GLPI : choix de la langue

À SAVOIR

Si la page ne s'affiche pas, vérifiez que `glpi.sio.local` se résout bien en 192.168.20.62 (étape 1) et que le conteneur `glpi` est démarré.

2 Cliquez « **Installer** » (et non « Mettre à jour », réservé aux montées de version) :



Figure 8 – Assistant GLPI : début de l'installation

3 L'assistant vérifie l'environnement. Toutes les lignes « **Requis** » doivent être vertes, ainsi que les lignes « Suggéré ». Cliquez « **Continuer** » :



Figure 9 – Assistant GLPI : vérification de la compatibilité

À SAVOIR

Une ligne rouge indique l'élément à corriger : dans un conteneur, il s'agit presque toujours d'une extension PHP absente du Dockerfile. Corrigez-le, reconstruisez l'image (`docker compose build`) et recréez le conteneur (`docker compose up -d`).

4 Étape 1 : saisissez le serveur SQL **db** (le **nom du service** MariaDB dans `compose.yaml`, et non `localhost`), l'utilisateur **glpi** et le mot de passe `DB_PASSWORD` du fichier `.env`, puis cliquez « **Continuer** » :



Figure 10 – Assistant GLPI : connexion à la base

À SAVOIR

Docker fournit un DNS interne : le nom `db` désigne le conteneur de la base. Avec `localhost`, GLPI chercherait la base dans son propre conteneur et échouerait (« Connection refused »).

5 Étape 2 : sélectionnez la base existante **glpi** puis cliquez « **Continuer** ». L'étape 3 crée alors les tables (une minute environ) ; cliquez « **Continuer** » à chaque fin d'étape, y compris à l'écran « Une dernière chose avant de démarrer » (collecte de statistiques facultative, à votre convenance) :



Figure 11 – Assistant GLPI : choix de la base de données

6 L'installation est terminée. L'écran rappelle les **quatre comptes par défaut** (mot de passe identique à l'identifiant). Cliquez « **Utiliser GLPI** » puis connectez-vous avec **glpi** / **glpi** :

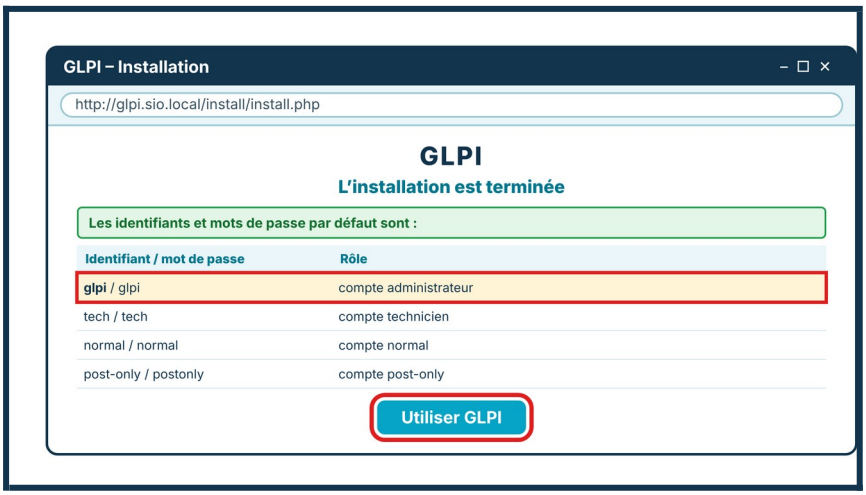


Figure 12 – Assistant GLPI : installation terminée

À SAVOIR

Ces comptes et mots de passe sont publics : le point suivant les neutralise. GLPI vient d'écrire `config_db.php` et la clé `glpicrypt.key` dans le volume `glpi-config`.

C – Sécuriser immédiatement

Dès la connexion, GLPI affiche des **alertes de sécurité** en rouge. Ce sont deux vraies failles à corriger tout de suite : les mots de passe par défaut (ci-dessous) et le fichier d'installation (point E) :



Figure 13 – Page d'accueil : alertes de sécurité

À SAVOIR

Les alertes disparaissent d'elles-mêmes une fois les deux corrections effectuées (rechargez la page).

1 **Mot de passe de glpi** : menu « **Administration** » puis « **Utilisateurs** », cliquez **glpi**, saisissez un mot de passe solide (deux fois) et enregistrez

2 **Autres comptes par défaut** (`tech`, `normal`, `post-only`) : cochez-les dans la liste des utilisateurs puis, dans le menu « **Actions** », supprimez-les ; à défaut, changez leur mot de passe

D – Passer les tâches automatiques en mode CLI

Par défaut, GLPI exécute ses tâches lorsque quelqu'un navigue sur le site. Comme le conteneur **glpi-cron** s'en charge chaque minute, on le déclare à GLPI : menu « **Configuration** » puis « **Actions automatiques** » :

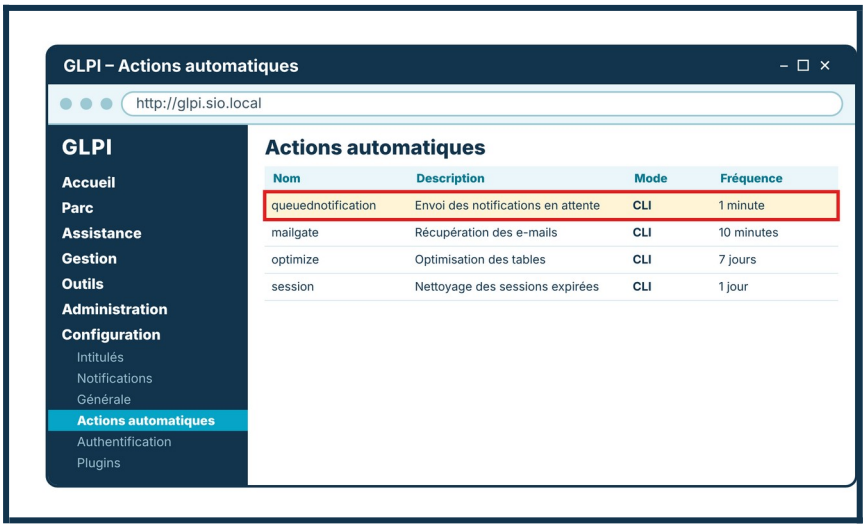


Figure 14 – Configuration > Actions automatiques

À SAVOIR

Ouvrez chaque action concernée (au minimum `queuednotification` et `mailgate`) et réglez son **mode d'exécution** sur « CLI » avant d'enregistrer. Vous pouvez aussi cocher toutes les lignes et utiliser le menu « Actions » de la liste, s'il propose cette modification en masse.

E – Supprimer le fichier d'installation et contrôler le cron

1. Supprimez le fichier d'installation (il permettrait de relancer l'assistant), puis lancez la tâche à la main pour contrôler que `cron.php` s'exécute sans erreur :

TERMINAL

```
cd /opt/glpi
sudo docker compose exec glpi rm /var/www/glpi/install/install.php
sudo docker compose exec glpi test ! -e /var/www/glpi/install/install.php \
    && echo "install.php supprimé"
sudo docker compose exec --user www-data glpi php /var/www/glpi/front/cron.php
echo $?
```

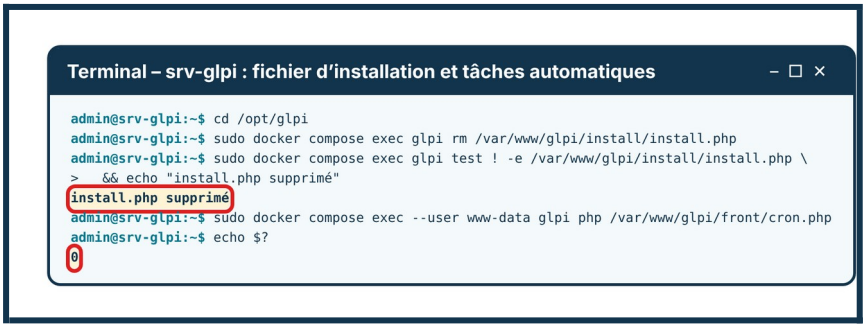


Figure 15 – Fichier d'installation supprimé et tâche exécutée

À SAVOIR

Le code de retour 0 signifie que la tâche s'est exécutée sans erreur. Après quelques minutes, la colonne « Dernière exécution » de la liste des actions automatiques se renseigne.

REMARQUE IMPORTANTE

Le fichier supprimé fait partie de l'**image** : il réapparaît lorsque le conteneur `glpi` est **recréé** (mise à jour de l'image, `up -d --force-recreate`). Refaites alors la commande `rm`. Les données, elles, ne sont jamais touchées : elles sont dans les volumes.

F – Sauvegarder et mettre à jour

Avec Docker, **quatre éléments** sont à sauvegarder : la **base de données**, les volumes `glpi-config` (avec la clé de chiffrement `glpicrypt.key`, sans elle les mots de passe stockés deviennent illisibles), `glpi-files` et `glpi-marketplace`, ainsi que le dossier du projet (`compose.yaml` et `.env`, qui contient les mots de passe). Les images, elles, se reconstruisent. Exportez la base **à chaud**, puis archivez les volumes et le projet :

TERMINAL

```
cd /opt/glpi
sudo docker compose exec -T db sh -c 'mariadb-dump --single-transaction \
    -uroot -p"$MARIADB_ROOT_PASSWORD" glpi' | gzip \
    | sudo tee /var/backups/glpi-$(date +%F).sql.gz > /dev/null
sudo tar czf /var/backups/glpi-donnees-$(date +%F).tar.gz \
    -C /var/lib/docker/volumes \
    glpi_glpi-config glpi_glpi-files glpi_glpi-marketplace
sudo tar czf /var/backups/glpi-projet-$(date +%F).tar.gz -C /opt glpi
```

Pour **mettre à jour** GLPI : sauvegardez d'abord, puis changez `GLPI_VERSION` dans `.env`, reconstruisez l'image, recréez les conteneurs et mettez la base à jour avec la console de GLPI. Copiez les sauvegardes **hors du serveur** (partage réseau, serveur de sauvegarde).

TERMINAL

```
sudo nano /opt/glpi/.env      # GLPI_VERSION=<nouvelle version>
sudo docker compose up -d --build
sudo docker compose exec --user www-data glpi php bin/console glpi:database:update
```

À SAVOIR

En production, publiez GLPI en **HTTPS** (certificat délivré par l'autorité de certification de l'entreprise) : les identifiants des utilisateurs ne doivent pas circuler en clair sur le réseau. Le tutoriel « Installer GLPI derrière Traefik et le lier à AD DS » montre comment le faire avec un proxy inverse ; dans ce tutoriel, le site reste en HTTP pour se concentrer sur Docker et sur l'annuaire.

✔ **POINT DE CONTRÔLE**

- `docker compose ps` indique **3 services** démarrés (base « healthy »)
- L'assistant s'est terminé sur « **Utiliser GLPI** » et la connexion fonctionne
- Le mot de passe de **glpi** est changé, les comptes `tech`, `normal` et `post-only` sont supprimés, et le fichier `install.php` est supprimé
- Le mode d'exécution des **actions automatiques** est **CLI** et `cron.php` renvoie le code **0**
- Une **sauvegarde** de la base, des volumes et du dossier `/opt/glpi` existe

ÉTAPE 5 SUR 6

05 Préparer AD DS : compte de service et autorité de certification

Pour authentifier les utilisateurs, GLPI interroge l'annuaire avec un **compte de service** (lecture seule), puis vérifie le mot de passe de chaque utilisateur. Cette étape se déroule sur les **contrôleurs de domaine** et sur **SRV-GLPI**.

REMARQUE IMPORTANTE

Windows Server 2025 impose la signature LDAP : les connexions LDAP simples (port 389, sans chiffrement) avec identifiant et mot de passe sont **refusées par défaut** sur un domaine neuf. GLPI doit donc utiliser **LDAPS** (LDAP chiffré, port **636**). Chaque contrôleur présente un certificat délivré par **SIO-CA** ; il suffit que **les conteneurs fassent confiance à cette autorité**.

A – Créer l'unité d'organisation, le compte de service et deux utilisateurs de test

Sur **SRV-DC1**, dans PowerShell (**administrateur**), créez les unités d'organisation (OU), le compte **svc-glpi** et deux utilisateurs pour les tests. Si votre domaine possède déjà une OU pour vos utilisateurs, utilisez-la à la place de « Utilisateurs » (ce sera la **BaseDN** de GLPI) :

POWERSHELL

```
$base = "DC=sio,DC=local"
New-ADOrganizationalUnit -Name "Comptes de service" -Path $base
New-ADOrganizationalUnit -Name "Utilisateurs" -Path $base
$mdp = Read-Host -AsSecureString "Mot de passe de svc-glpi"
New-ADUser -Name "svc-glpi" -SamAccountName "svc-glpi" `
    -UserPrincipalName "svc-glpi@sio.local" `
    -Path "OU=Comptes de service,$base" -AccountPassword $mdp `
    -Enabled $true -PasswordNeverExpires $true -CannotChangePassword $true
```

POWERSHELL

```
$mdp = Read-Host -AsSecureString "Mot de passe des utilisateurs de test"
New-ADUser -Name "Alice Martin" -GivenName Alice -Surname Martin `
    -SamAccountName amartin -UserPrincipalName amartin@sio.local `
    -EmailAddress amartin@sio.local -Path "OU=Utilisateurs,$base" `
    -AccountPassword $mdp -Enabled $true
New-ADUser -Name "Bruno Durand" -GivenName Bruno -Surname Durand `
    -SamAccountName bdurand -UserPrincipalName bdurand@sio.local `
    -EmailAddress bdurand@sio.local -Path "OU=Utilisateurs,$base" `
    -AccountPassword $mdp -Enabled $true
```

1 Contrôlez la création :

POWERSHELL

```
Get-ADUser svc-glpi | Select Name, Enabled, DistinguishedName
$ou = "OU=Utilisateurs,$base"
Get-ADUser -Filter * -SearchBase $ou | Select Name, SamAccountName
```

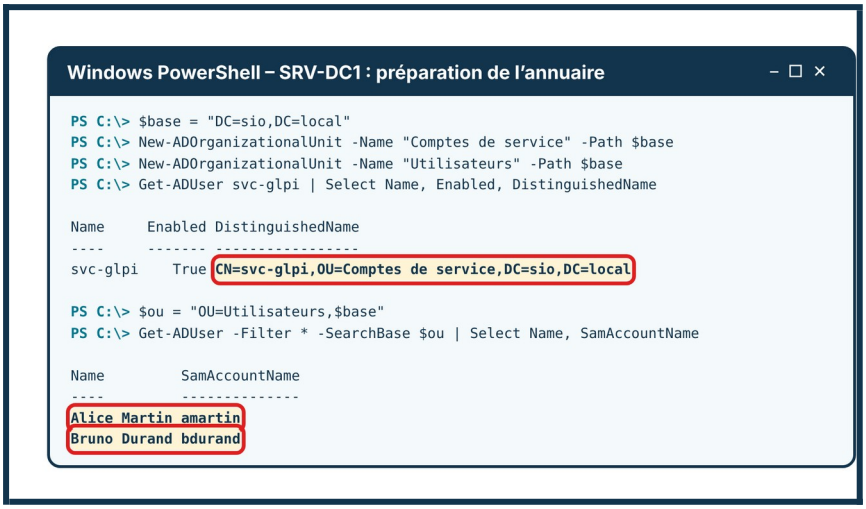


Figure 16 – PowerShell : OU, compte de service et utilisateurs

À SAVOIR

Le compte **svc-glpi** est un compte de domaine ordinaire : il peut lire l'annuaire, rien d'autre. Ne l'ajoutez à aucun groupe d'administration. Son mot de passe n'expire pas, sinon GLPI cesserait de fonctionner le jour de l'expiration.

B – Exporter le certificat de l'autorité SIO-CA

Les contrôleurs présentent un certificat signé par **SIO-CA** (tutoriel « Activer LDAPS sur AD DS »). Pour le vérifier, un client n'a besoin que du **certificat de l'autorité**, pas de ceux des contrôleurs : un seul fichier couvre SRV-DC1, SRV-DC2 et tout futur contrôleur. Sur **SRV-DC1**, exportez-le, copiez-le vers **SRV-GLPI** et contrôlez que le port **636** répond :

POWERSHELL (SRV-DC1)

```
New-Item -ItemType Directory -Path C:\Temp -Force | Out-Null
certutil -ca.cert C:\Temp\sio-ca.cer
scp C:\Temp\sio-ca.cer admin@192.168.20.62:/tmp/
Test-NetConnection srv-dc1.sio.local -Port 636
```

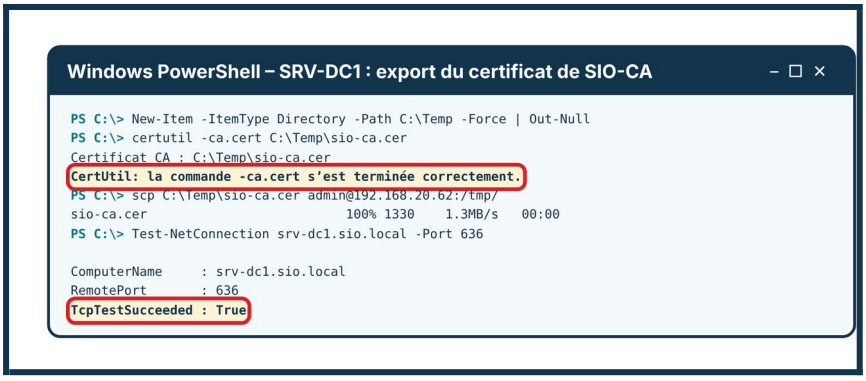


Figure 17 – PowerShell : export du certificat de SIO-CA et test du port 636

À SAVOIR

Ce test prouve seulement que le port est ouvert. La vérification du certificat lui-même se fait depuis Debian (point suivant). Si vous préférez WinSCP ou un partage, seul compte le résultat : le fichier `sio-ca.cer` sur SRV-GLPI.

C – Donner le certificat aux conteneurs et tester LDAPS

Sur **SRV-GLPI**, convertissez le certificat au format **PEM** attendu par OpenSSL et placez-le dans le dossier `certs` du projet : il est monté en lecture seule dans les conteneurs sous `/certs/sio-ca.crt` (c'est le fichier désigné par `LDAPTLS_CACERT`). Vérifiez ensuite la chaîne **depuis le conteneur**, puis la recherche d'un utilisateur depuis l'hôte (le mot de passe de **svc-glpi** est demandé) :

TERMINAL

```
sudo openssl x509 -inform der -in /tmp/sio-ca.cer -out /opt/glpi/certs/sio-ca.crt
cd /opt/glpi
sudo docker compose exec -T glpi openssl s_client -connect srv-dc1.sio.local:636 \
    -CAfile /certs/sio-ca.crt -verify_return_error </dev/null 2>&1 \
    | grep -E 'subject=|Verify'
sudo apt install -y ldap-utils
LDAPTLS_CACERT=/opt/glpi/certs/sio-ca.crt ldapsearch -H ldaps://srv-dc1.sio.local \
    -D "svc-glpi@sio.local" -W -b "OU=Utilisateurs,DC=sio,DC=local" \
    "(sAMAccountName=amartin)" dn mail
```

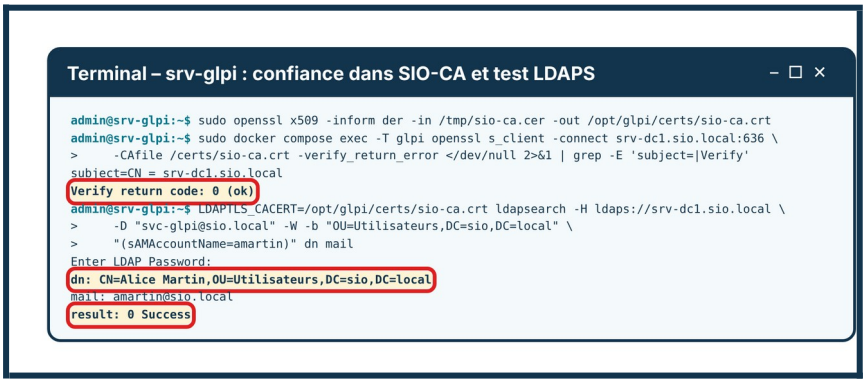


Figure 18 – Debian : confiance dans SIO-CA et recherche LDAPS

À SAVOIR

« Verify return code: 0 (ok) » prouve que le conteneur fait confiance au certificat. « result: 0 Success » et la ligne `dn:` prouvent que le compte de service, la BaseDN et LDAPS fonctionnent. Répétez avec `srv-dc2.sio.local` : le même fichier suffit. Si `-inform der` échoue, le fichier est déjà au format PEM : copiez-le simplement.

À SAVOIR

Si vos contrôleurs utilisent des certificats **auto-signés** (tutoriel « natif »), il n'y a pas d'autorité : exportez les certificats de SRV-DC1 et de SRV-DC2, convertissez-les en PEM puis réunissez-les dans le même fichier (`cat srv-dc1.crt srv-dc2.crt | sudo tee /opt/glpi/certs/sio-ca.crt`). Le reste du tutoriel est identique.

REMARQUE IMPORTANTE

Après avoir ajouté ou remplacé le fichier, **redémarrez le conteneur glpi** (`sudo docker compose restart glpi`) : PHP conserve en mémoire sa configuration TLS et ne relit pas toujours le fichier à chaud.

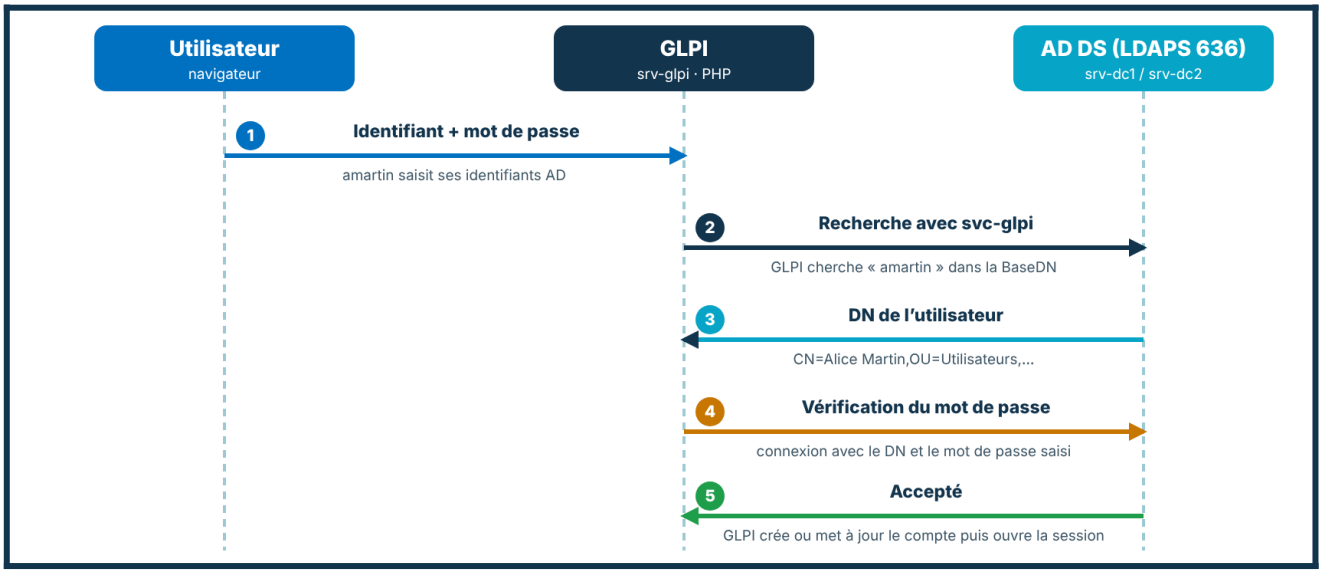
✔ **POINT DE CONTRÔLE**

- Les OU **Utilisateurs** et **Comptes de service**, le compte **svc-glpi** et les utilisateurs **amartin** et **bdurand** existent
- Le fichier `/opt/glpi/certs/sio-ca.crt` existe et « Verify return code: **0 (ok)** » s'affiche **depuis le conteneur** `glpi` pour `srv-dc1` et `srv-dc2`
- `ldapsearch` en **ldaps://** renvoie l'entrée de **Alice Martin** (« result: 0 Success »)

ÉTAPE 6 SUR 6

06 Lier GLPI à l'annuaire et importer les utilisateurs

Tout est prêt : on déclare l'annuaire dans GLPI, on teste, puis on importe les utilisateurs. Voici ce qui se passe quand un utilisateur de l'annuaire se connecte :



A – Déclarer l'annuaire AD DS

1 Connectez-vous à GLPI en **glpi**. Menu « **Configuration** » puis « **Authentification** » puis « **Annuaires LDAP** » ; cliquez « **+** » pour ajouter un annuaire et choisissez la préconfiguration « **Active Directory** » (elle renseigne le filtre et les champs d'AD)

2 Renseignez les champs suivants (les autres valeurs préconfigurées restent inchangées) puis cliquez « **Ajouter** » :

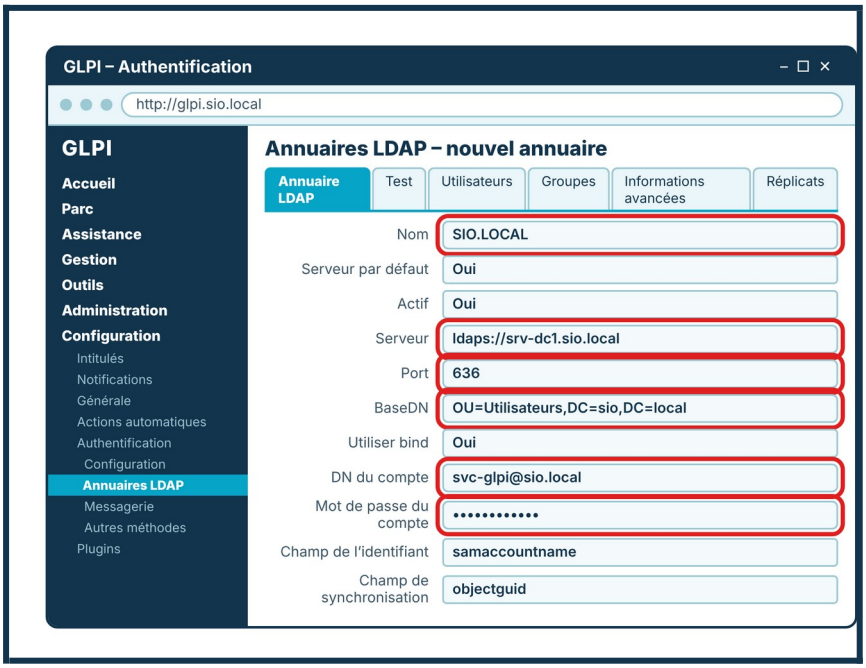


Figure 19 – Annuaires LDAP : formulaire de l'annuaire SIO.LOCAL

À SAVOIR

Le filtre de connexion préconfiguré pour Active Directory ne retient que les comptes utilisateurs **actifs**. Le champ de synchronisation `objectguid` identifie un compte de façon unique : renommer un utilisateur dans l'annuaire ne crée pas de doublon.

Champ	Valeur à saisir	Remarque
Nom	`SIO.LOCAL`	Nom affiché dans GLPI et sur la page de connexion
Serveur par défaut / Actif	Oui / Oui	Source d'authentification utilisée par défaut
Serveur	`ldaps://srv-dc1.sio.local`	Le préfixe **ldaps://** active le chiffrement ; le nom doit figurer dans le certificat
Port	`636`	Port LDAPS (389 = LDAP non chiffré)
BaseDN	`OU=Utilisateurs,DC=sio,DC=local`	Où chercher : seuls les comptes de cette OU sont visibles
Utiliser bind	Oui	Connexion authentifiée (le compte de service)
DN du compte	`svc-glpi@sio.local`	Format « identifiant@domaine » accepté par AD
Mot de passe du compte	mot de passe de **svc-glpi**	Stocké chiffré dans la base de GLPI
Champ de l'identifiant	`samaccountname`	Valeur préconfigurée : l'identifiant de connexion
Champ de synchronisation	`objectguid`	Valeur préconfigurée : identifiant unique du compte

B – Tester la connexion et ajouter le second contrôleur

1 Ouvrez l'annuaire créé, onglet « **Test** », puis cliquez « **Tester** » : le message de succès confirme la connexion et l'authentification du compte de service :

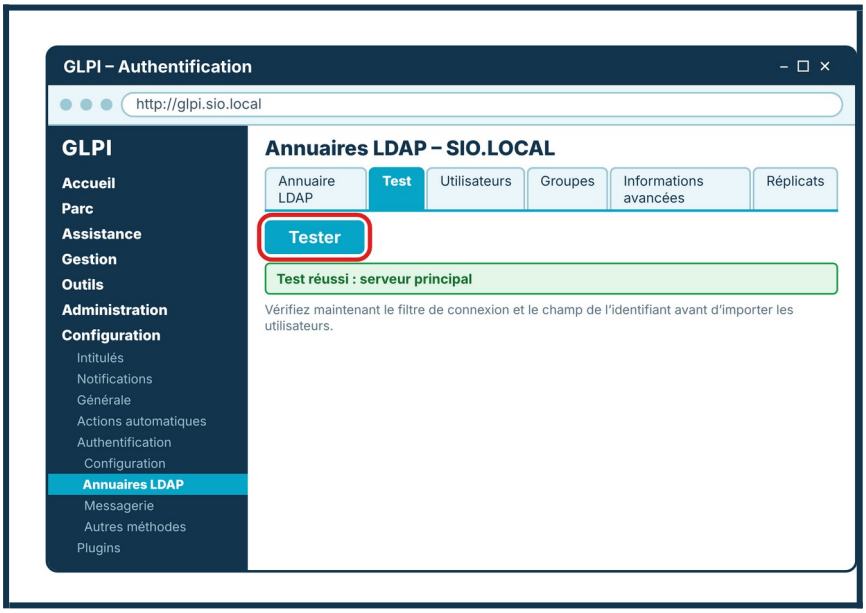


Figure 20 – Annuaire SIO.LOCAL : onglet Test

À SAVOIR

En cas d'échec, le message indique la cause. Reportez-vous au tableau de dépannage en fin de tutoriel : certificat non reconnu (fichier `sio-ca.crt` absent ou conteneur non redémarré), nom du serveur incorrect ou mot de passe erroné.

2 Pour la **tolérance de panne**, ouvrez l'onglet « **Réplicats** », saisissez le nom **SRV-DC2**, le serveur `ldaps://srv-dc2.sio.local` et le port **636**, puis cliquez « **Ajouter** » : si SRV-DC1 est arrêté, GLPI interroge SRV-DC2 :

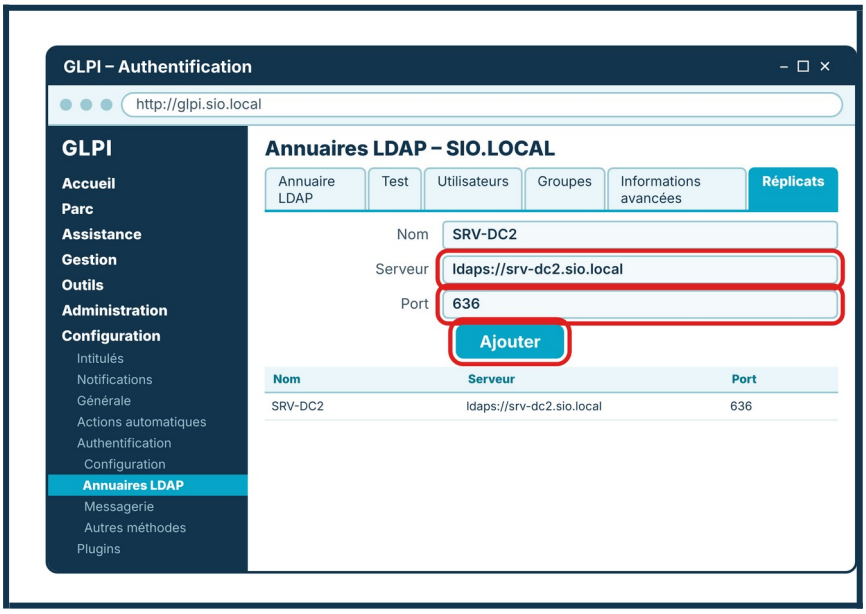


Figure 21 – Annuaire SIO.LOCAL : onglet Réplicats

À SAVOIR

Relancez ensuite le test : il vérifie le serveur principal et chaque réplicat.

C – Importer les utilisateurs

Un utilisateur de l'annuaire peut se connecter **sans import préalable** (GLPI le crée à sa première connexion), mais l'import permet de **préparer les comptes** : leur attribuer des droits ou les ajouter à un ticket avant leur première visite.

1 Menu « **Administration** » puis « **Utilisateurs** », bouton « **Lier un annuaire externe** » puis « **Importation de nouveaux utilisateurs** » ; cliquez « **Rechercher** », cochez les utilisateurs (ou la case d'en-tête pour tous) puis « **Actions** » et « **Importer** » :

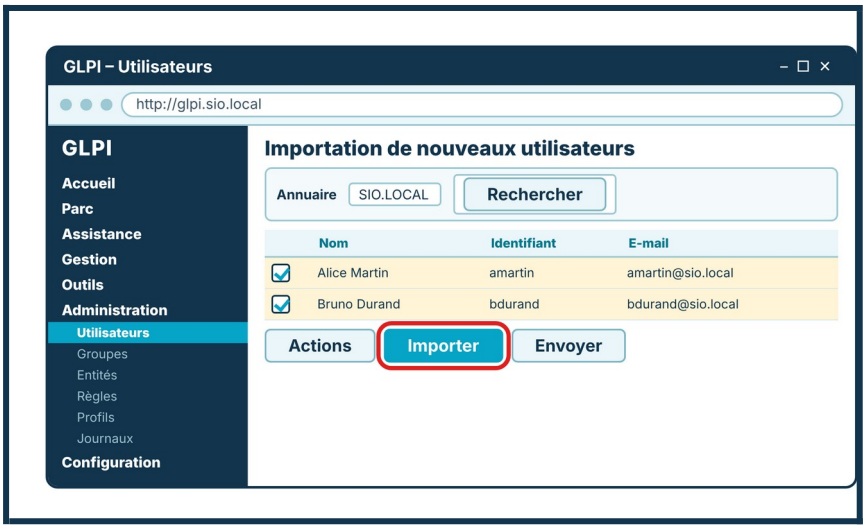


Figure 22 – Importation de nouveaux utilisateurs

À SAVOIR

Seuls les comptes de la BaseDN qui répondent au filtre apparaissent. Un compte désactivé dans l'annuaire n'est pas proposé.

2 La liste des utilisateurs contient désormais **amartin** et **bdurand**, dont la source d'authentification est l'annuaire :

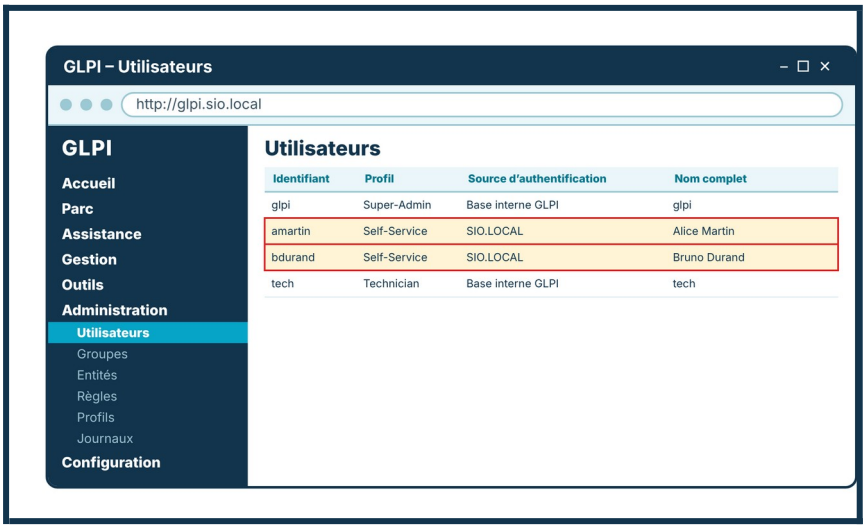


Figure 23 – Administration > Utilisateurs : comptes importés

À SAVOIR

Les utilisateurs importés reçoivent le profil par défaut du paramétrage de GLPI (en général « Self-Service » : créer des tickets et suivre les siens). Pour confier un autre rôle, ouvrez le compte, onglet « Habilitations », et ajoutez un profil (par exemple « Technician »).

D – Se connecter avec un compte Windows

1 Déconnectez-vous de GLPI, puis ouvrez la page de connexion dans une **fenêtre de navigation privée** : saisissez **amartin**, son mot de passe de domaine, et sélectionnez **SIO.LOCAL** dans « Connexion à » :



Figure 24 – Page de connexion de GLPI avec l'annuaire

À SAVOIR

Le mot de passe n'est jamais copié dans GLPI : à chaque connexion, GLPI le fait vérifier par l'annuaire. Changer ou réinitialiser le mot de passe dans Active Directory suffit donc.

2 **Pour aller plus loin** : la commande `sudo docker compose exec --user www-data glpi php bin/console glpi:ldap:synchronize_users --help` décrit la synchronisation en masse des utilisateurs (créations, mises à jour, désactivations), qui peut être planifiée

✔ POINT DE CONTRÔLE

- Le **test** de l'annuaire SIO.LOCAL réussit, serveur principal et réplicat
- **amartin** et **bdurand** sont importés avec l'annuaire comme source d'authentification
- **amartin** ouvre une session GLPI avec son **mot de passe Windows**
- Si SRV-DC1 est arrêté, la connexion fonctionne encore grâce à **SRV-DC2**

Notre GLPI est **opérationnel** : il tourne dans trois conteneurs sur Debian 13, avec ses données dans des volumes, sécurisé, sauvegardable, et relié à AD DS en LDAPS avec un second contrôleur en secours. Il reste à le remplir (inventaire, entités, profils, plugin d'inventaire) et à le publier en HTTPS.

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)
Construction : « Could not resolve host: github.com »	Les DNS du domaine ne relaient pas vers Internet	Configurer un redirecteur DNS sur SRV-DC1, ou ajouter temporairement un DNS public dans `/etc/resolv.conf`
« 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 (`"8080:80"`)
Le conteneur `glpi` est en « Restarting »	Erreur dans `glpi.conf` ou `php.ini`	`sudo docker compose logs --tail 30 glpi` ; corriger puis `docker compose up -d --build`
Assistant : « Connection refused » ou « php_network_getaddresses »	Serveur SQL saisi `localhost` ou base pas encore prête	Saisir **db** ; attendre l'état « healthy » dans `docker compose ps`
Assistant : « Access denied for user glpi »	Mot de passe de `.env` modifié après la création du volume	Saisir le mot de passe d'origine, ou `docker compose down -v` (efface les données) puis redémarrer
Assistant : ligne « Requis » en rouge	Extension PHP absente de l'image	Ajouter l'extension au Dockerfile, `docker compose build`, `up -d`
Erreur 500 ou page blanche	Erreur PHP ou de configuration	`sudo docker compose logs --tail 30 glpi` et `docker compose exec glpi ls /var/log/glpi`
Test LDAP : « Can't contact LDAP server »	`sio-ca.crt` absent ou invalide, conteneur non redémarré, nom différent du certificat, port 636 bloqué	Refaire l'étape 5-C ; `docker compose restart glpi` ; utiliser le **nom complet** du contrôleur, pas son adresse IP
`ldapsearch` : « Strong(er) authentication required »	Connexion LDAP non chiffrée refusée par Windows Server 2025	Utiliser `ldaps://` et le port **636**, jamais `ldap://` avec mot de passe
« Invalid credentials » (code 49)	DN du compte ou mot de passe incorrect, compte désactivé	Saisir `svc-glpi@sio.local` ; tester `ldapsearch -W` ; vérifier `Get-ADUser svc-glpi`
Test réussi mais aucun utilisateur à importer	BaseDN incorrecte ou OU vide	Comparer avec `Get-ADUser -SearchBase` ; vérifier la casse et les virgules de la BaseDN
Les actions automatiques ne s'exécutent pas	Conteneur `glpi-cron` arrêté	`sudo docker compose ps` ; `sudo docker compose logs glpi-cron` ; `sudo docker compose up -d`
L'alerte « install.php » est revenue	Le conteneur a été recréé (le fichier est dans l'image)	Refaire la suppression (étape 4-E)

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 --build`	Reconstruit l'image si le Dockerfile a changé, puis recrée les conteneurs concernés
`sudo docker compose restart glpi`	Redémarre GLPI sans le recréer (après un changement de certificat, par exemple)
`sudo docker compose exec glpi sh`	Ouvre un terminal dans le conteneur GLPI
`sudo docker compose down`	Supprime conteneurs et réseau (les volumes sont conservés ; `-v` les efface !)
`/opt/glpi/compose.yaml` **et** `.env`	Configuration du projet : à sauvegarder, `.env` reste confidentiel
`/opt/glpi/certs/sio-ca.crt`	Certificat de l'autorité SIO-CA, monté dans les conteneurs
`sudo docker volume ls`	Volumes existants (`glpi_glpi-config` contient la clé de chiffrement : à sauvegarder)
Configuration > Authentification > Annuaires LDAP	Déclarer, tester et dupliquer les annuaires (réplicats)
`ldapsearch -H ldaps://… -D … -W`	Tester l'annuaire sans passer par GLPI