

**ASSURMER AP n°5**

INSTALLATION ET CONFIGURATION DE GLPI DANS UN ENVIRONEMENT DOCKER

Réalisation AP n°5 - 22/05/2024 v1.1 - AUTEURS :

RODRIGUES Antoine, BOUSSAHA Elijah, Aymeric PERRET DU CRAY

Validateur : Nassim MATOUK (maitre d’apprentissage Antoine)

**TABLES DES MATIERES**

[Prérequis pour l’installation de GLPI sur Docker 1](#_Toc319853025)

[Installation de Docker 2](#_Toc1743787695)

[Installation de Portainer 4](#_Toc1864731945)

# **Prérequis pour l’installation de GLPI sur Docker**

Pour installer GLPI, nous avons demandé au service infrastructure de nous fournir une machine physique comprenant certaine caractéristique.

Cette machine sera l’hôte de Docker. Étant donné que Docker ne nécessite pas de grandes performances, nous avons opté pour une machine avec des spécifications modestes.

Voici les caractéristiques de la machine, nommée ASSURDOCKER :

* 2 vCPU
* 8 Go de RAM
* SSD Nvme de 512 Go de stockage
* Système d’exploitation Debian 12 Bullseye

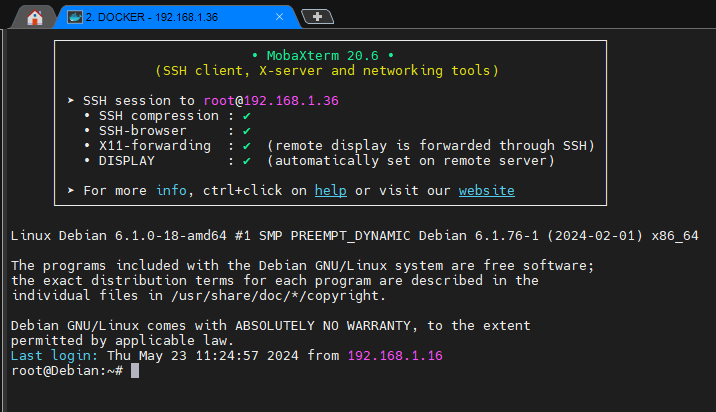
La connexion au réseau interne est également nécessaire pour effectuer la jonction à l’Active Directory, permettant ainsi aux collaborateurs d’accéder à GLPI avec leurs comptes personnels AD.

# **Installation de Docker**

On accède à la machine via un accès SSH dont l’IP est **192.168.1.36**.

1. La machine dispose d’un accès SSH incorporé et déjà configurer. On va donc se connecter au serveur en SSH grâce au logiciel **MobaXTerm**.

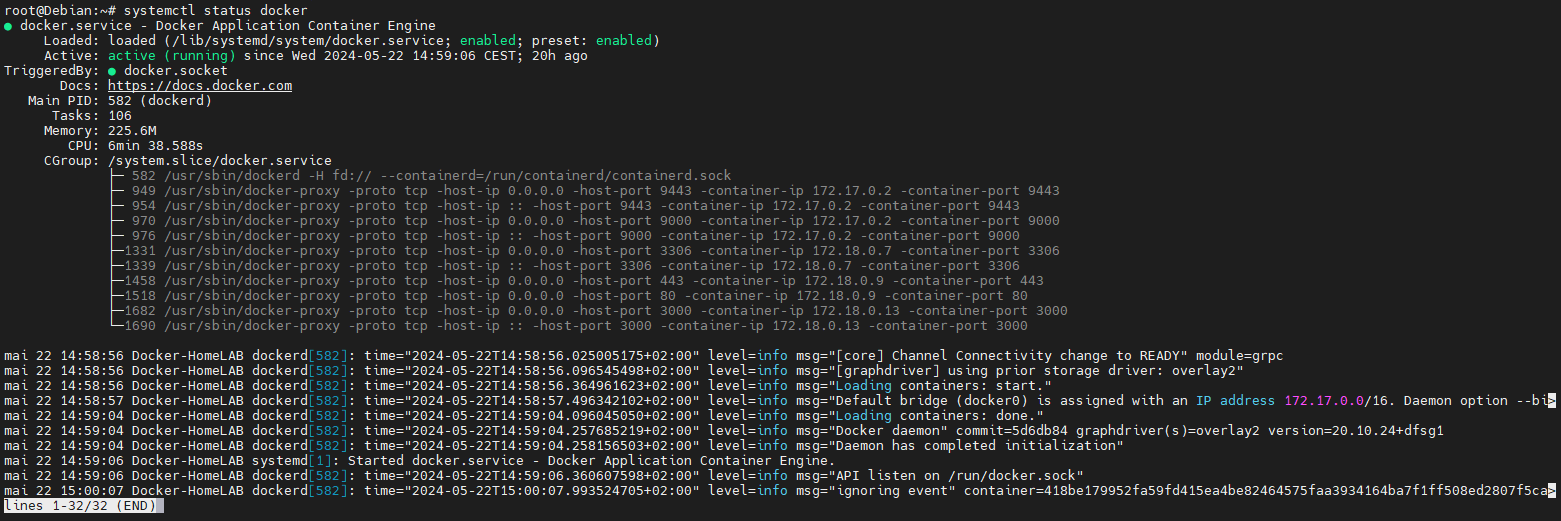




1. Ensuite il faut taper ces commandes qui sont les commandes pour l’installation de Docker fournis sur le site : [**https://docs.docker.com/engine/install/debian/**](https://docs.docker.com/engine/install/debian/)

sudo apt-get update  
sudo apt-get install ca-certificates curl  
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  
  
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" | \  
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
apt-get update

1. Nous vérifions si Docker est maintenant bien installé sur notre machine avec cette commande : systemctl status Docker



Docker est bien installé. On passe désormais à l’installation de Portainer.

# **Installation de Portainer**

Avant de commencer, Que-est-ce que Portainer ?

Portainer est lui-même un container, permettant de gérer graphiquement des containers que l’on a installé sur notre machine Docker. Cela permet une gestion plus simple, à l’aide d’une interface utilisateur simple d’utilisation et intuitive sur un navigateur web.

1. Toujours via SSH, on lance cette commande : docker run -d -p 9000:9000 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer\_data:/data portainer/portainer-ce:latest Cela va nous permettre d’installer le container “portainer” sur le port 9000.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

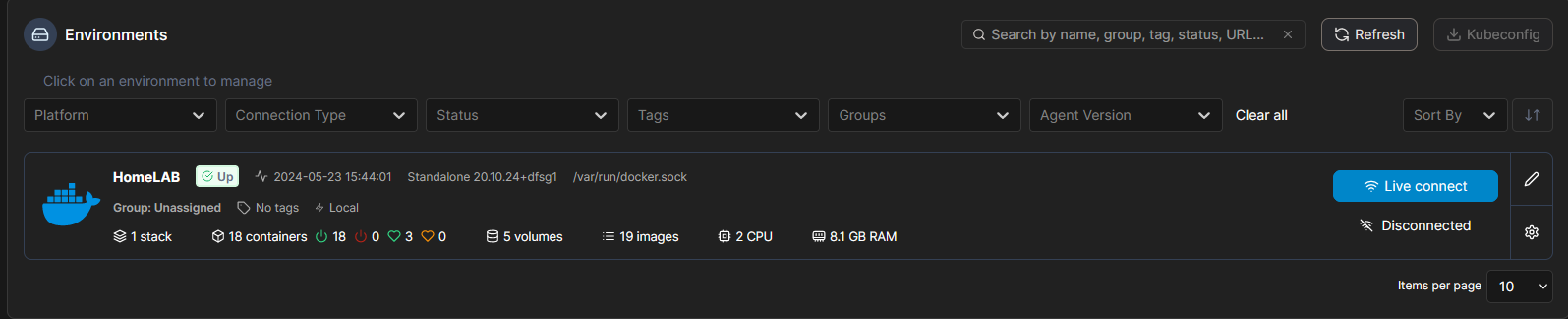
Description générée automatiquement

1. Maintenant il faut accéder à l’interface Web de Portainer, à l’adresse 192.168.1.36:9000 et ensuite créé les identifiants administrateurs.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

1. Portainer reconnais automatiquement l’environnement Docker que nous avons installé précédemment.



 **Installation et configuration de la base de données MySQL + PhpMyAdmin** 

Nous allons utiliser l’image Docker (version dérivée de l’initial) de GLPI créé par diouxx (<https://github.com/DiouxX/docker-glpi>). Cette image est très renommé et régulièrement mis à jour, elle est elle même open-source (pour être en accord avec la licence de GLPI), et il est donc possible de vérifier très rapidement son contenu et sa fiabilité.

Nous allons nous baser sur l’installation avec une base de données (MySQL) et données persistantes, vouée à l’utilisation en environnement de production. Ce type de base de données est aussi faite pour gérer une grande quantité de données le tout en restant très optimiser et peu gourmand en ressources.

Nous allons également rajouter un container PhPMyAdmin qui nous permettra de gérer toutes les bases de données dont celle de GLPI plus efficacement via une interface graphique. Cet ajout n’est pas nécessaire mais dans un but de facilité et d’un environnement de production c’est ce qu’il nous faut.

1. C’est le début ! Et comme au début rien n’est créé alors nous allons devoir créer pour la première et dernière fois une solution de reverse proxy avec nginx qui gère aussi les certificats dans un container docker afin de principalement rediriger les différents trafics (ex: [glpi.assurmer-asurance.fr](https://glpi.assurmer-assurance.fr)) vers les bons containers. Pour commencer nous allons créer un dossier nginx dans le /home et y insérer le contenu ce trouvant sur ce repo [github](https://github.com/evertramos/nginx-proxy-automation). Enfin, nous allons exécuter cette commande : ./fresh-start.sh --yes --skip-docker-image-check --use-nginx-conf-files -e contact@antoinerodrigues.com pour crée le container. Maintenant qu’il est crée il n’y à plus qu’a rajouter les variables d’environnement :

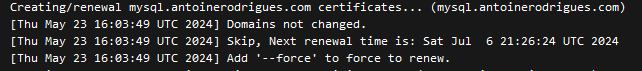
-e VIRTUAL\_HOST=glpi.antoinerodrigues.com

-e LETSENCRYPT\_HOST=glpi.antoinerodrigues.com

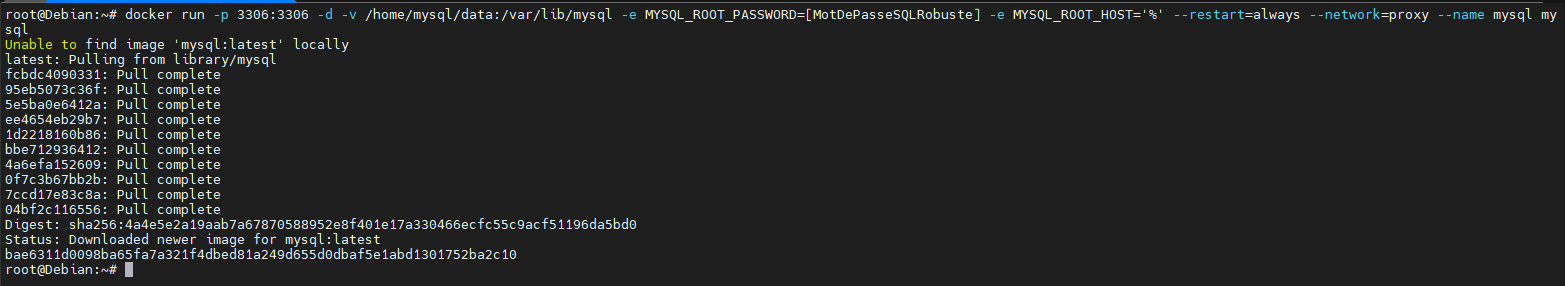
Pour que le container fasse le tout automatiquement de la gestion des certificats SSL aux redirections :



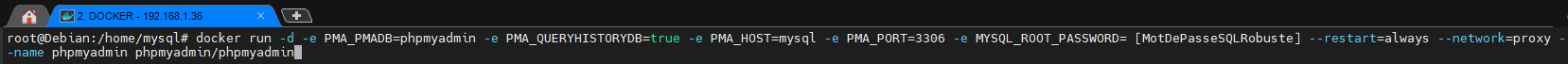
Exemple avec mysql :



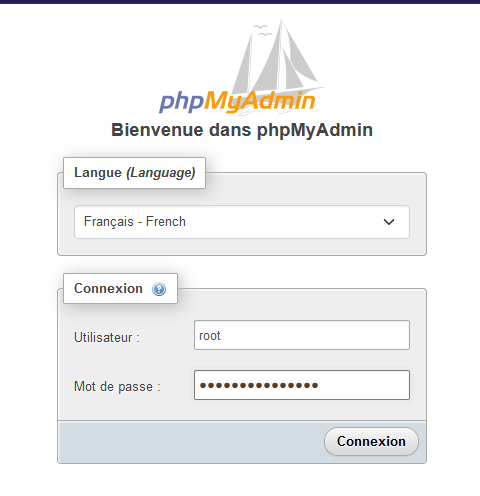
1. On créer le container qui contiendra la base de données MySQL avec la commande : docker run -p 3306:3306 -d -v /home/mysql/data:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=[MotDePasseSQLRobuste] -e MYSQL\_ROOT\_HOST='%' --restart=always --network=proxy --name mysql mysql



1. Et on fait la même chose pour le container PhpMyAdmin : docker run -d -e PMA\_PMADB=phpmyadmin -e PMA\_QUERYHISTORYDB=true -e PMA\_HOST=mysql -e PMA\_PORT=3306 -e MYSQL\_ROOT\_PASSWORD= [MotDePasseSQLRobuste] --restart=always --network=proxy --name phpmyadmin phpmyadmin/phpmyadmin



Puis on vérifie que phpmyadmin fonctionne bien :



1. Et l’on va faire de même pour GLPI: docker run -d --user "$(id -u)" --volume "$PWD/glpi/var/www/html/glpi:/var/www/html/glpi" --env-file /home/glpi/mariadb.env --name=glpi --restart=always --network=proxy diouxx/glpi Comme le but étant de mettre GLPI en environnement de production nous avons fait en sorte de rediriger le nom de domaine [glpi.assurmer-assurance.fr](https://glpi.assurmer-assurance.fr/) vers le container GLPI que nous venons de crée sur docker grâce au système de reverse proxy que nous avons précédemment crée / configuré.



Et on vérifie la encore si GLPI est bien accessible depuis l’url [glpi.assurmer-assurance.fr](https://glpi.assurmer-assurance.fr/) :



Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

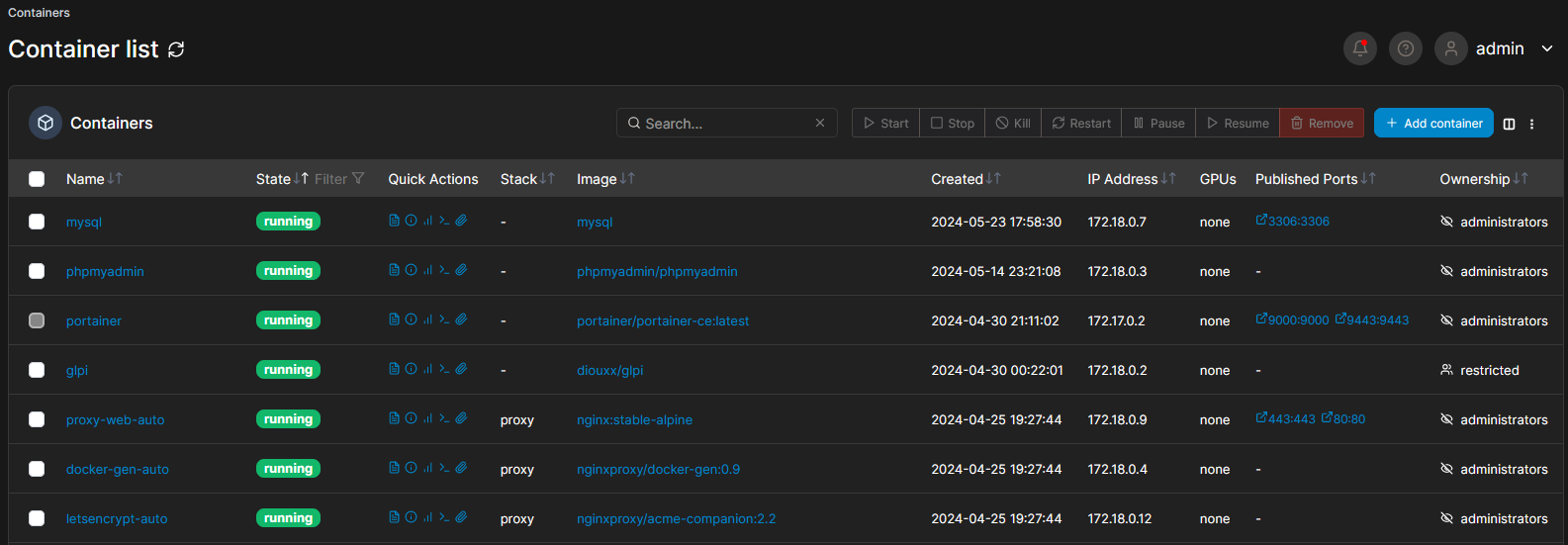
A la suite de toutes ces commandes effectuées en SSH nous devrions maintenant avoirs 7 containers créés et visible sur portainer :

- 3 pour le reverse proxy (proxy-web-auto, docker-gen-auto, letsencrypt-auto)

- 1 pour GLPI (glpi)

- 2 pour la gestion de base de données (mysql, phpmyadmin)

- 1 pour la gestion de containers (portainer) :



 **Configuration partie 1 (de base) de GLPI** 

1. Pour commencer la configuration de glpi il faut cliquer sur « OK »

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. Ensuite cliquer sur « Continuer »

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

1. Puis sur « Installer »

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. Veillez à ce que tout soit bien sélectionné et cliquer sur « Continuer ».

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

1. Entrer les informations de connexion à la base de données MySQL. Cliquer sur « Continuer ».



1. Une fois la base de données initialisé la page de connexion apparait.

