# comment installer-glpi-inventory-management-sur-debian-12

GLPI is an open-source IT asset and Helpdesk management solution written in PHP. It's a complete solution IT management software for your organization. GLPI helps you manage incidents/requests, create forms, and define SLAs. It also helps you manage your hardware, software, and data center solution, which also allows you to link asset inventory and get control of your IT and business infrastructure.

Not only that but GLPI also can be used as financial management for tracking your expenses, contracts, and suppliers, creating new inventory objects, managing user databases, and generating reports. Furthermore, GLPI includes project management for task assignments, adding collaborators, setting up timelines and reminders, and also provides a Kanban board for easier task management.

In this guide, I'll show you how to install GLPI IT Management Software on a Debian 12 machine. You will install GLPI alongside the LAMP Stack (Apache2, MariaDB, and PHP). Furthermore, you will also secure GLPI via SSL/TLS certificates from Letsencrypt.

## Prerequisites

To get started, ensure that you have:

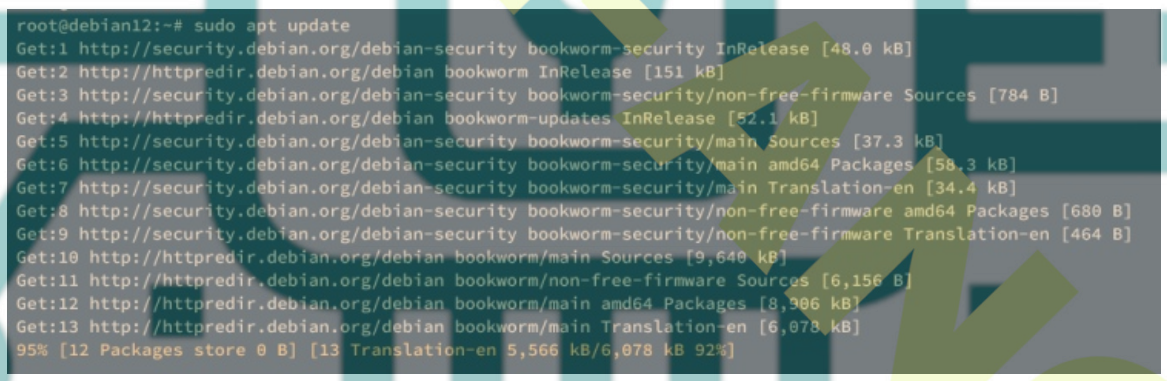
- A Debian 12 server.
- A non-root user with sudo administrator privileges.
- A domain name pointed to the server IP address.

## Installing Dependencies

GLPI is an open-source IT management software written in PHP with MySQL/MariaDB as the database. It can be run with Apache2 or Nginx web server. In this guide, you will install GLPI with the LAMP Stack (Apache2, MariaDB, and PHP), complete the following steps to install LAMP Stack, and some additional dependencies for your GLPI installation.

First, update and refresh your Debian package index by executing the apt update command below.

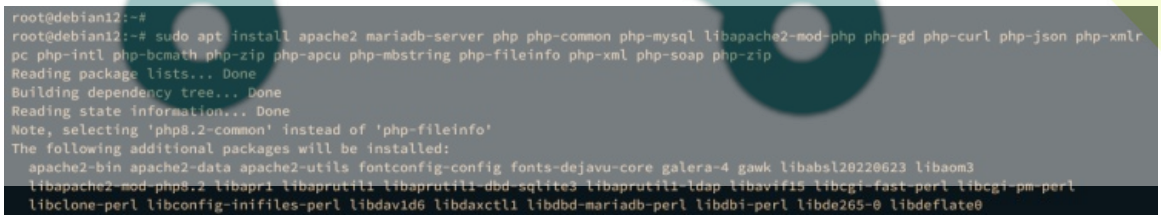
```
sudo apt update
```




Once the repository is updated, enter the following command to install package dependencies for GLPI installation, including the LAMP Stack (Apache2, MariaDB, and PHP) packages and some additional PHP extensions.

```
sudo apt install apache2 mariadb-server php php-common php-mysql libapache2-mod-php php-gd php-curl php-json php-xmlrpc php-intl php-bcmath php-zip php-apcu php-mbstring php-fileinfo php-xml php-soap php-zip
```

Type y to confirm and proceed with the installation.




After the dependencies are installed, verify each dependencies by executing the following command.

Verify the apache2 service to ensure that the service is running and enabled.

```
sudo systemctl is-enabled apache2
sudo systemctl status apache2
```

The output enabled confirms that the Apache2 service is enabled. The active (running) output confirms that Apache2 is running.

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled apache2
enabled
root@debian12:~# sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 15317 (apache2)
      Tasks: 6 (limit: 4642)
     Memory: 21.1M
        CPU: 197ms
```

Verify the mariadb service using the command below.

```
sudo systemctl is-enabled mariadb
sudo systemctl status mariadb
```

The output should be similar to the following:

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled mariadb
enabled
root@debian12:~# sudo systemctl status mariadb
● mariadb.service - MariaDB 10.11.3 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 13378 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 10 (limit: 4642)
```

Lastly, verify the PHP version and list of enabled extensions using the command below.

```
php -v
php -m
```

You should see that PHP 8.2 is installed with some extensions such as fileinfo, gd, intl, mysqli, and zlib enabled.

```
root@debian12:~#
root@debian12:~# php -v
PHP 8.2.7 (cli) (built: Jun  9 2023 19:37:27) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.2.7, Copyright (c) Zend Technologies
    with Zend OPcache v8.2.7, Copyright (c), by Zend Technologies
root@debian12:~#
root@debian12:~# php -m
[PHP Modules]
apcu
bcmath
calendar
Core
ctype
curl
date
dom
exif
FFI
fileinfo
filter
ftp
gd
```

## Configuring MariaDB Server

After installing LAMP Stack packages, next you will configure your MariaDB server by securing it via the *mariadb-secure-installation* utility. Then, you will also create a new MariaDB database and user that GLPI will use.

Execute the *mariadb-secure-installation* utility on your terminal to secure your MariaDB server installation.

```
sudo mariadb-secure-installation
```

Input Y to apply the configuration to your MariaDB server, or n for No, and reject the changes. Below are some related MariaDB configurations you will be asked for:

- Switch local authentication to unix_socket? Input n for no.
- Set up MariaDB root password? Input y, then type the new MariaDB root password and repeat.
- Remove the default anonymous user? Input y to confirm.
- Disable remote login for the root user? Input y to confirm.
- Remove the default database test? Input y to confirm.
- Reload table privileges and apply changes? Input y to confirm.

After securing the MariaDB server, you will create a new database and user that will be used by GLPI. You can create a new database and user via the mariadb client.

Log in to the MariaDB server using the following command, Be sure to input your MariaDB root password when asked.

```
sudo mariadb -u root -p
```

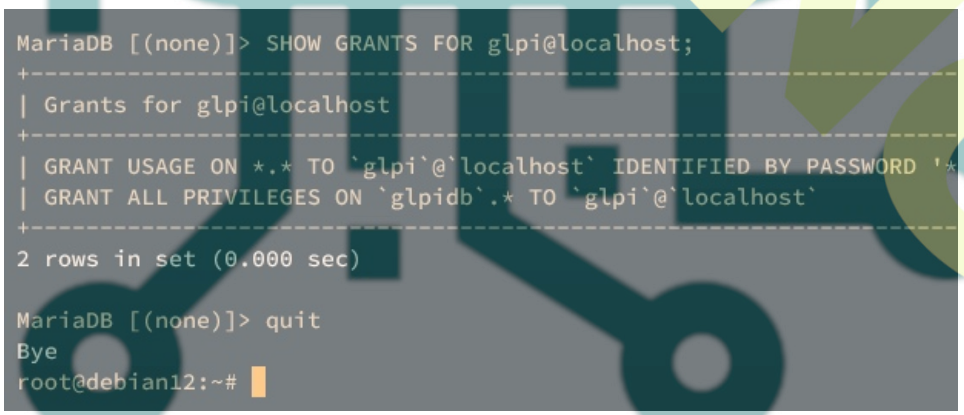
Next, execute the following queries to create a new MariaDB database glpidb with user glpi and the password is p4ssw0rd. Be sure to change the following with your info.

```
CREATE DATABASE glpidb;
CREATE USER glpi@localhost IDENTIFIED BY 'p4ssw0rd';
GRANT ALL PRIVILEGES ON glpidb.* TO glpi@localhost;
FLUSH PRIVILEGES;
```


Now run the following query to verify the privileges for user glpi@localhost.

```
SHOW GRANTS FOR glpi@localhost;
```

You should see the user *glpi@localhost* has privileges to access the database *glpidb*.




Type quit to exit from the MariaDB server.

## Configuring PHP

In the following section, you will set up your PHP installation by editing the php.ini file. Then, you will verify your Apache2 and PHP installation by creating a new PHPINFO file that will show you detailed information about your PHP installation.

Open the default *php.ini* configuration using the following nano editor command.

```
sudo nano /etc/php/8.2/apache2/php.ini
```

Change some of the following PHP configurations like this. Be sure to adjust the date.timezone with your proper timezone and the *memory_limit* with your proper memory configuration.

```
memory_limit = 512M
date.timezone = Europe/Stockholm
upload_max_filesize = 16M
session.cookie_httponly = on
```

Save and close the file when you're done.

Next, execute the following systemctl command to restart the Apache2 service and apply the changes that you've made.

```
sudo systemctl restart apache2
```

Then create a new PHPINFO file */var/www/html/info.php* by executing the following command.

```
echo "<?php phpinfo(); ?>" > /var/www/html/info.php
```

Launch your web browser and visit your server IP address followed by the /info.php path, such as http://192.168.10.15/info.php. If your configuration is successful, you should see the PHPINFO page like the following:


## Downloading GLPI Source Code

After configuring PHP, you will download the GLPI source code from GitHub and set up proper permission and ownership of the GLPI web root directory.

Move to the */var/www* directory and download the GLPI source code via the wget command below. In this example, you will download GLPI v10.x, which is the latest and stable version of GLPI 10.x. Be sure to check the GLPI download page to get the latest version of it.

```
cd /var/www/
wget https://github.com/glpi-project/glpi/releases/download/10.0.9/glpi-10.0.9.tgz
```

Once the GLPI source code is downloaded, extract it using the tar command below, and you should see a new directory */var/www/glpi*.

```
tar -xf glpi-10.0.9.tgz
```

Now run the following command to change the ownership of the */var/www/glpi* directory to user *www-data* and allow the Apache2 web server to access it.

```
sudo chown -R www-data:www-data /var/www/glpi
```

Then, execute the following command to make the directory files and config writable. This will allow the Apache2 web server to write to those directories for storing GLPI data.

```
sudo chmod u+rw /var/www/glpi/{files,config}
```

## Configuring Apache2 Virtual Host

Now that you've downloaded the GLPI source code, the next step is to create a new Apache2 virtual host configuration that will be used to run GLPI. Before going further, ensure that you have a domain name pointed to your Debian server IP address.

Before creating the virtual host configuration, execute the following command to enable the *rewrite* module in Apache2.

```
sudo a2enmod rewrite
```

Create a new virtual host configuration */etc/apache2/sites-available/glpi.conf* using the following nano editor command.

```
sudo nano /etc/apache2/sites-available/glpi.conf
```

Insert the following configuration and be sure to change the domain name with your domain on the ServerName parameter.

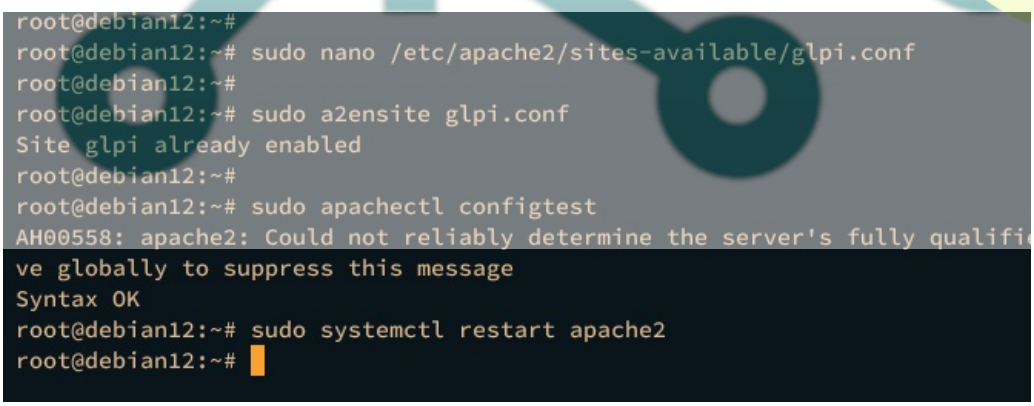
```
<VirtualHost *:80>
    ServerName glpi.hwdomain.io

    DocumentRoot /var/www/glpi/public

    # If you want to place GLPI in a subfolder of your site (e.g. your virtual host is serving multiple applications),
    # you can use an Alias directive:
    # Alias "/glpi" "/var/www/glpi/public"

    <Directory /var/www/glpi/public>
        Require all granted

        RewriteEngine On

        # Redirect all requests to the GLPI router, unless file exists.
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteRule ^(.*)$ index.php [QSA,L]
    </Directory>
</VirtualHost>
```

Save and close the file when you're done.

Next, execute the command below to activate the virtual host file glpi.conf. Then, verify your Apache2 syntax to ensure there is no syntax error.

```
sudo a2ensite glpi.conf
sudo apachectl configtest
```

If you've proper Apache2 syntax, you should get the output Syntax OK.

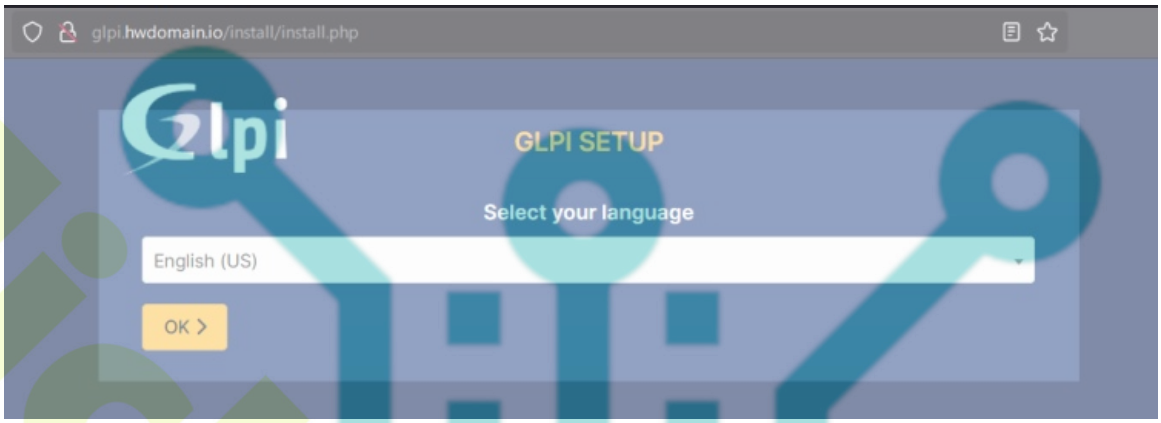



Now run the following systemctl command to restart the Apache2 service and apply the new changes on your virtual

host file.

```
sudo systemctl restart apache2
```

Lastly, open your web browser and visit the domain name of your GLPI installation, such as http://glpi.hwdomain.io. If your installation is successful, you should see the GLPI installation page like the following:

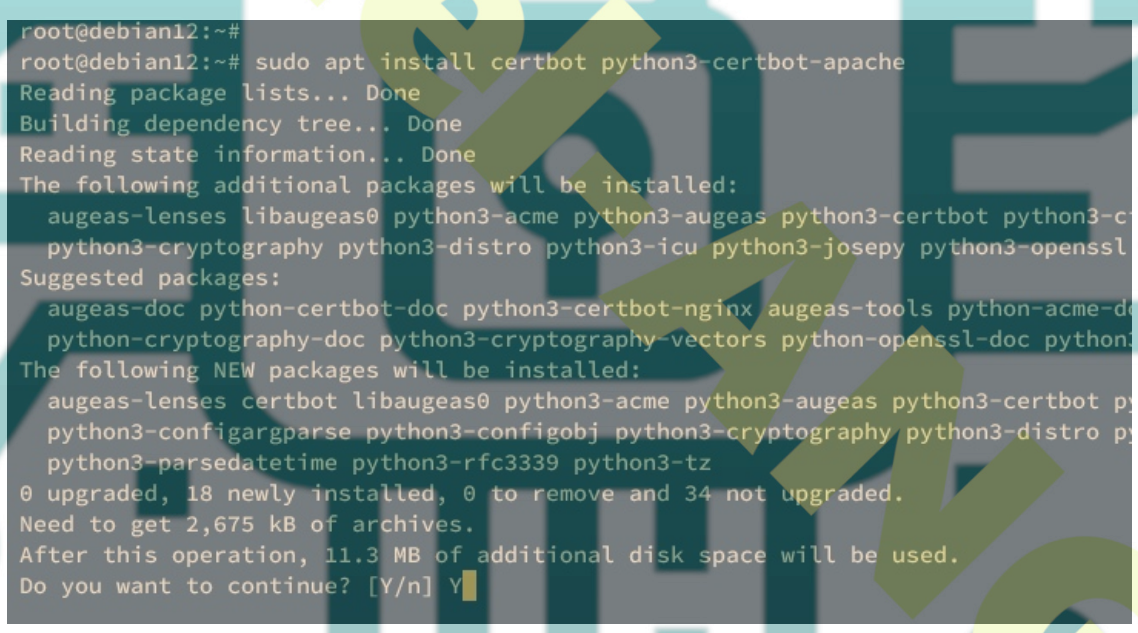



## Securing GLPI with SSL/TLS Certificates

At this point, you are almost ready to finish up GLPI installation. Now you will secure GLPI by generating SSL/TLS certificates from Letsencrypt via the Certbot tool and Certbot Apache plugin.

Run the following apt command to install the Certbot and Certbot Apache plugin. Type y to confirm and proceed with the installation.

```
sudo apt install certbot python3-certbot-apache
```




After Certbot is installed, run the certbot command below to generate new SSL/TLS certificates for your GLPI installation. Be sure to change the domain name and email address within the following command.

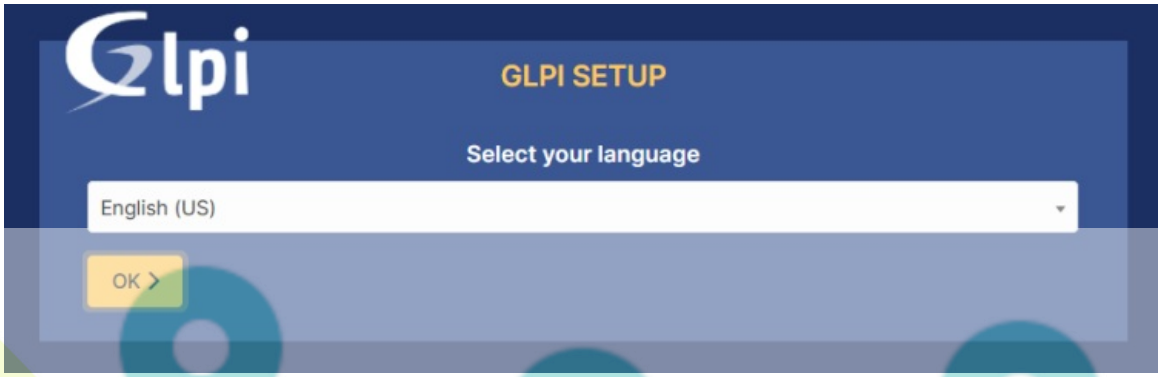
```
sudo certbot --apache --agree-tos --redirect --hsts --staple-ocsp --email alice@example.io -d glpi.example.io
```

Now your SSl/TLS certificates will be available at /etc/letsencrypt/live/domain.com directory. Also, your virtual host configuration glpi.conf is now configured with HTTPS, which is configured via the Certbot Apache2 plugin.

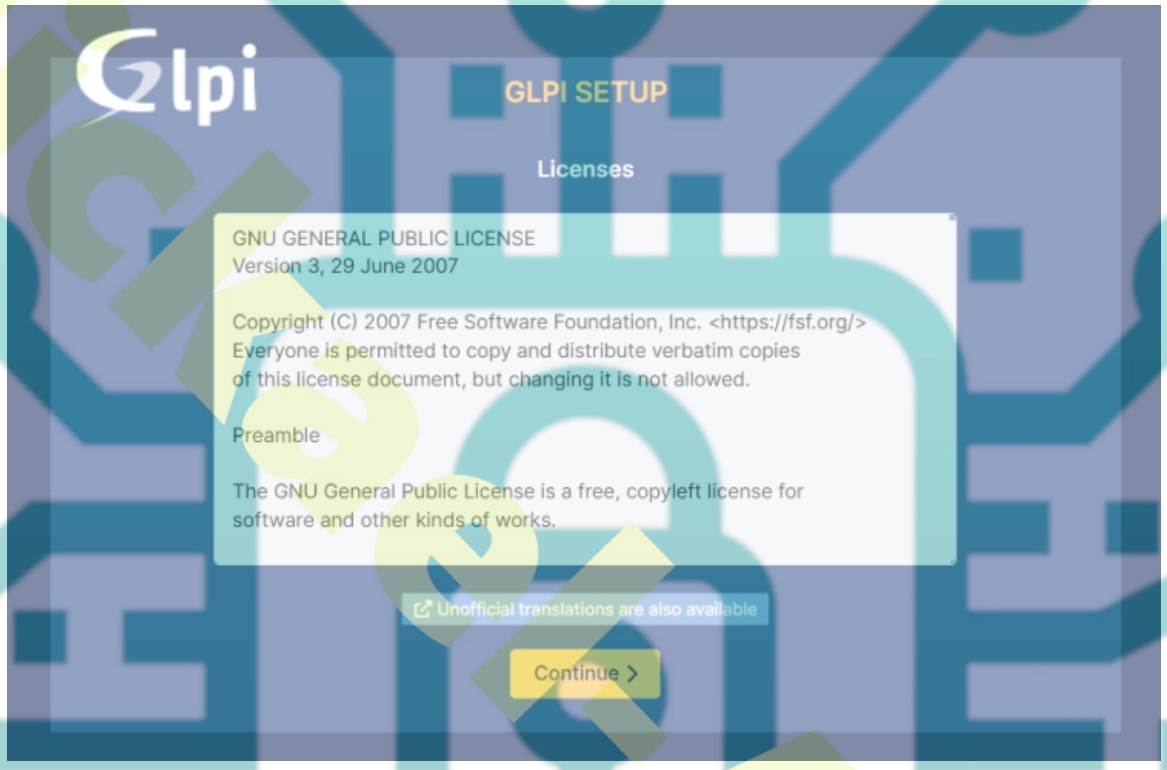
## GLPI Installation via Web Installer

Launch your web browser and visit the domain name of your GLPI installation, such as http://glpi.hwdomain.io/. Now you will be redirected to secure HTTPS connections and you will get the GLPI installation page.

First, select your preferred language and click **OK**.



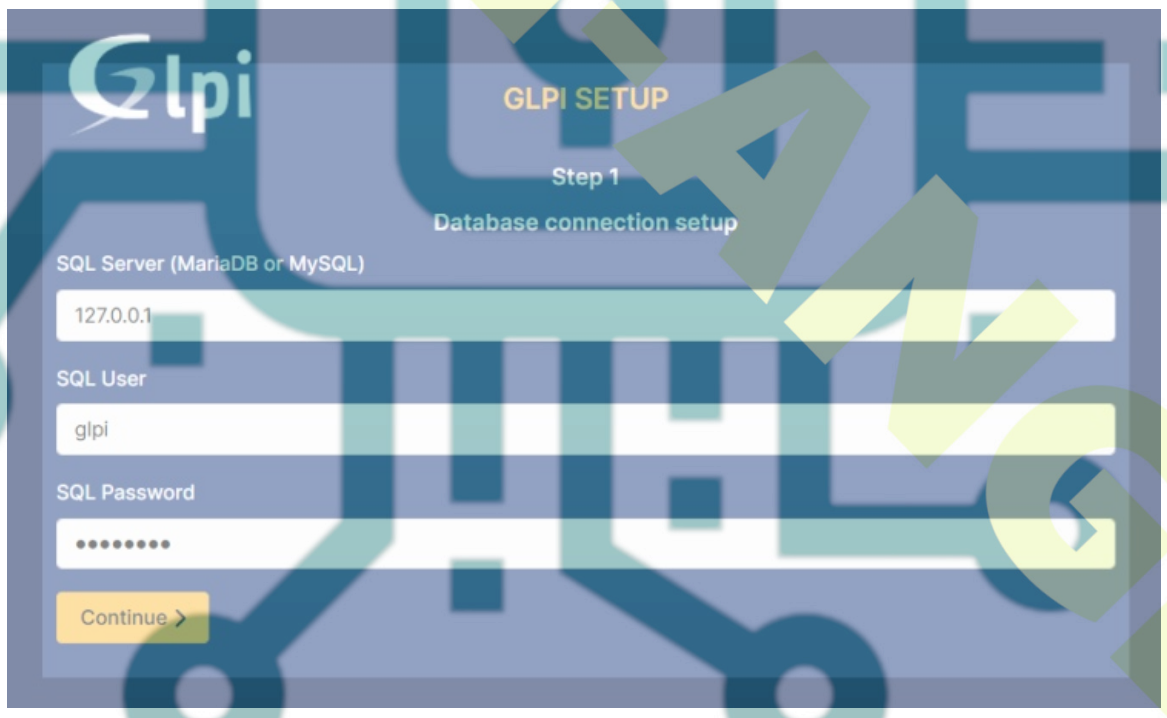

Accept the GLPI License and click **Continue** to proceed.



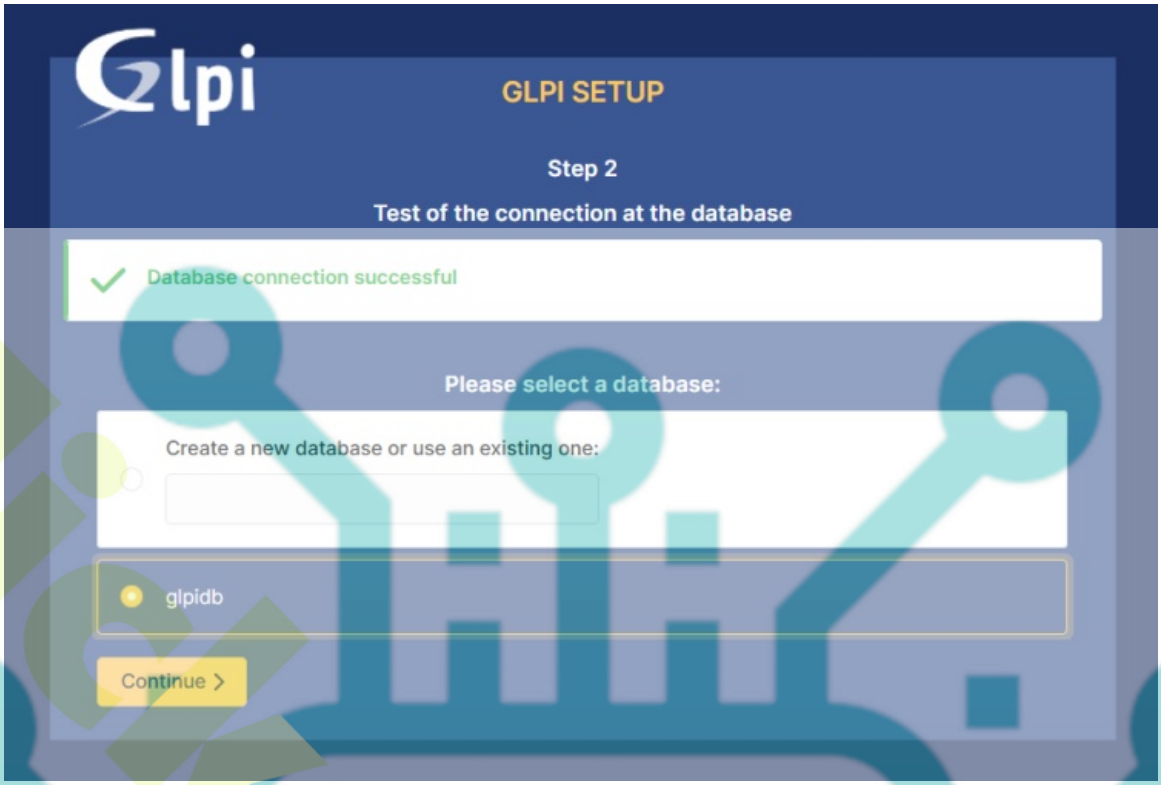

Click **Install** to start the GLPI installation.


Confirm that your server environment is met with GLPI requirement checks, then click **Continue**.


Input details MariaDB user that you've created for your GLPI installation and click **Continue**.



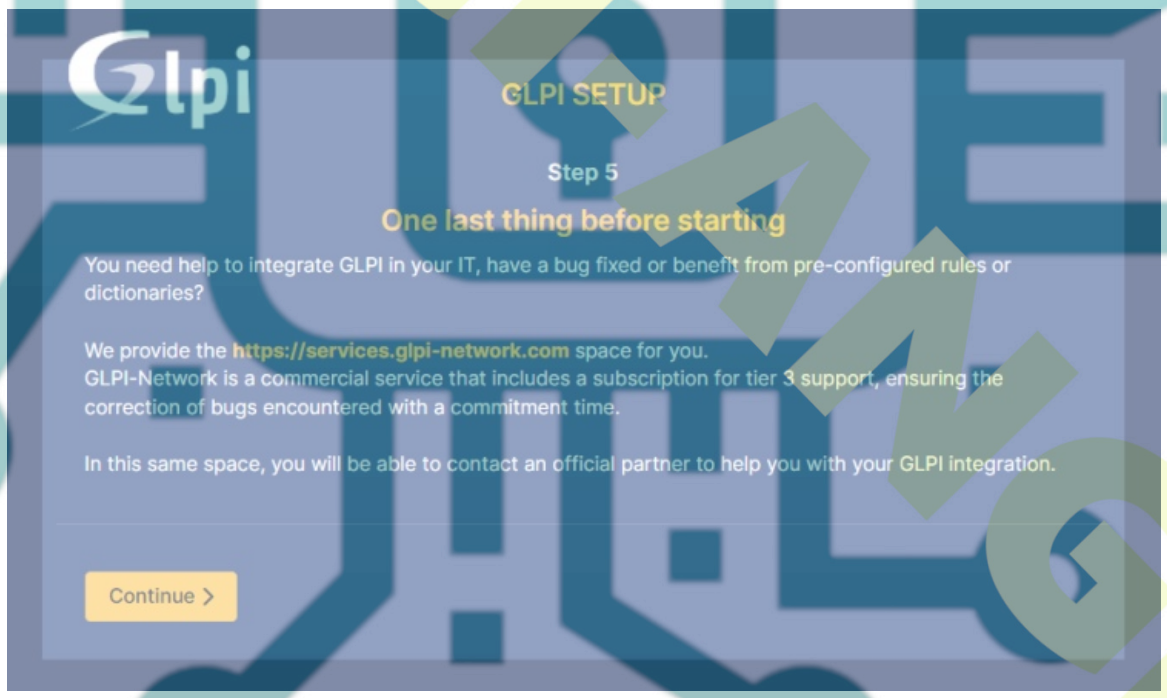

Select the database **glpidb** and **Continue** again.




The GLPI database will be initialized, click **Continue**.

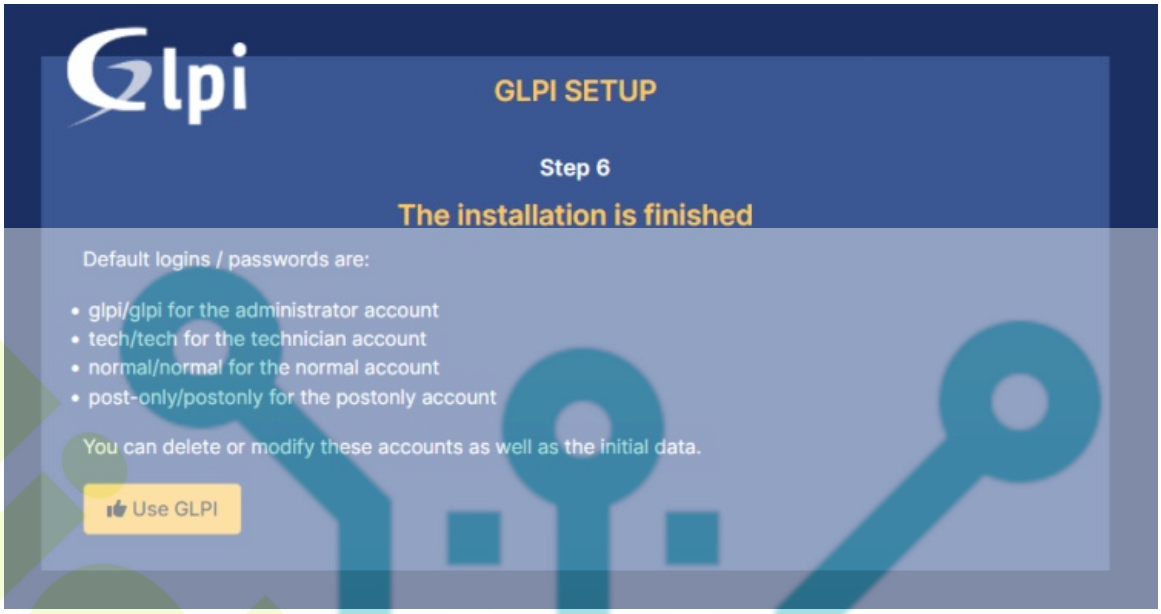

Configure the Collect data section with your preferred settings, then click **Continue**.


Select **Continue** to go to the next section.



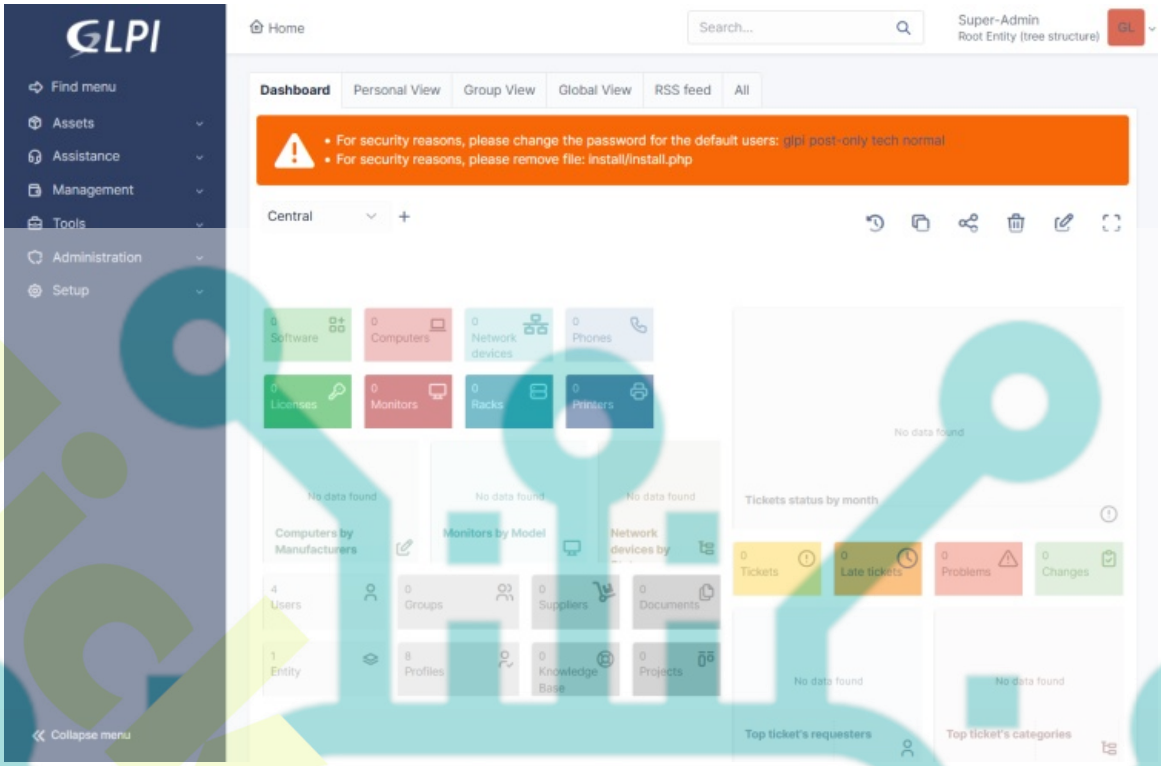

In the last, you should see some default GLPI users created, such as *glpi/glpi* as administrator, *tech/tech* for the technican account, *normal/normal* for the normal account, *post-only/postonly* for the postonly account.

Click **Use GLPI** to finish your installation.

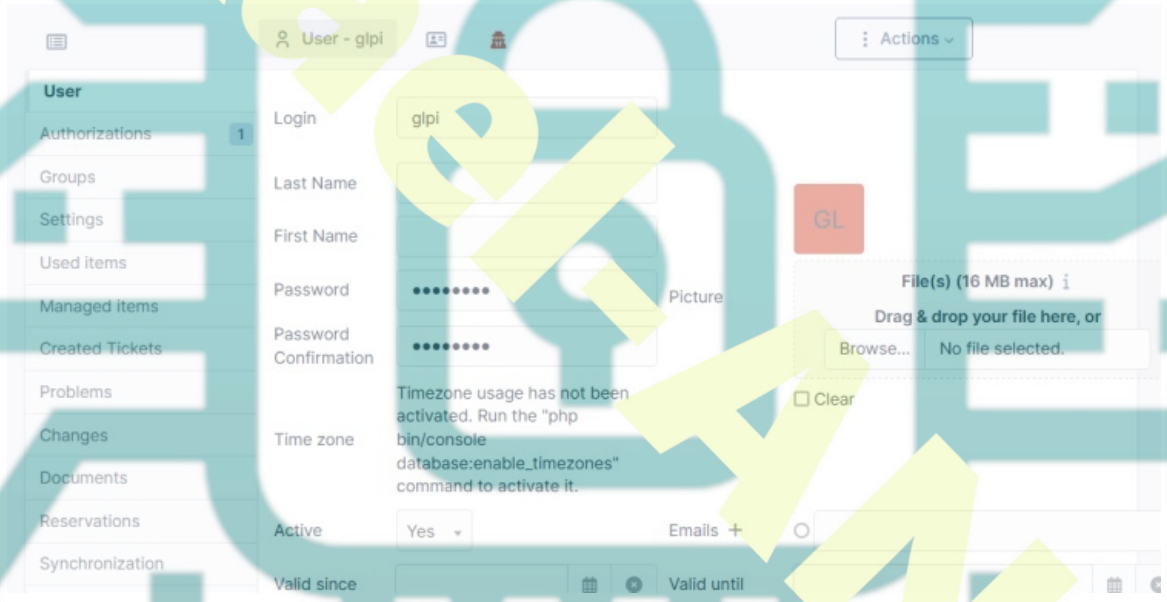



Now you should get the GLPI login page. Input the default user glpi with password glpi, then click **Sign In**.


Once logged in, you will get the GLPI administration dashboard like the following:




You will also be asked to change the default password for the glpi user. Input the new glpi administrator password and click **Save**.




Lastly, back to your terminal server and run the following command to remove the installer script.

```
sudo rm /var/www/glpi/public/install/install.php
```

## Conclusion

To wrap up, you've installed GLPI Inventory Management with the LAMP Stack (Apache2, MariaDB, and PHP) on the Debian 12 server. You've also secured GLPI with SSL/TLS certificates generated from Letsencrypt via Certbot and Certbot Apache plugin. Now you can explore the GLPI's vast capabilities as inventory management, data center infrastructure management (DCIM), and many more.