

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
```

```
root@debian12:~# sudo apt update
Get:1 http://security.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:2 http://httpredir.debian.org/debian bookworm InRelease [151 kB]
Get:3 http://security.debian.org/debian-security bookworm-security/non-free-firmware Sources [784 B]
Get:4 http://httpredir.debian.org/debian bookworm-updates InRelease [52.1 kB]
Get:5 http://security.debian.org/debian-security bookworm-security/main Sources [37.3 kB]
Get:6 http://security.debian.org/debian-security bookworm-security/main amd64 Packages [58.3 kB]
Get:7 http://security.debian.org/debian-security bookworm-security/main Translation-en [34.4 kB]
Get:8 http://security.debian.org/debian-security bookworm-security/non-free-firmware amd64 Packages [680 B]
Get:9 http://security.debian.org/debian-security bookworm-security/non-free-firmware Translation-en [464 B]
Get:10 http://httpredir.debian.org/debian bookworm/main Sources [9,640 kB]
Get:11 http://httpredir.debian.org/debian bookworm/non-free-firmware Sources [6,156 B]
Get:12 http://httpredir.debian.org/debian bookworm/main amd64 Packages [8,906 kB]
Get:13 http://httpredir.debian.org/debian bookworm/main Translation-en [6,073 kB]
95% [12 Packages store 0 B] [13 Translation-en 5,566 kB/6,078 kB 92%]
```

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.

```
root@debian12:~#
root@debian12:~# sudo apt install apache2 mariadb-server php php-common php-mysql libapache2-mod-php php-gd php-curl php-json php-xmlr
pc php-intl php-bcmath php-zip php-apcu php-mbstring php-fileinfo php-xml php-soap php-zip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'php8.2-common' instead of 'php-fileinfo'
The following additional packages will be installed:
 apache2-bin apache2-data apache2-utils fontconfig-config fonts-dejavu-core galera-4 gawk libabsl20220623 libaom3
 libapache2-mod-php8.2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libavif15 libbgi fast perl libbgi-pm perl
 libclone-perl libconfig-inifiles-perl libdavid6 libdaxctl1 libdbd-mariadb-perl libdbi-perl libde265-0 libdeflate0
```

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:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
  Main PID: 13378 (mariabdb)
    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;
```

```
MariaDB [(none)]> CREATE DATABASE glpidb;  
Query OK, 1 row affected (0.001 sec)  
  
EGESMariaDB [(none)]> CREATE USER glpi@localhost IDENTIFIED BY 'p4ssw0rd';  
Query OK, 0 rows affected (0.004 sec)  
  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glpidb.* TO glpi@localhost;  
Query OK, 0 rows affected (0.001 sec)  
  
MariaDB [(none)]> LUSH PRIVFLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.001 sec)
```

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*.

```
MariaDB [(none)]> SHOW GRANTS FOR glpi@localhost;  
+-----+  
| Grants for glpi@localhost  
+-----+  
| GRANT USAGE ON *.* TO `glpi`@`localhost` IDENTIFIED BY PASSWORD '*  
| GRANT ALL PRIVILEGES ON `glpidb`.* TO `glpi`@`localhost`  
+-----+  
2 rows in set (0.000 sec)  
  
MariaDB [(none)]> quit  
Bye  
root@debian12:~#
```

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:



PHP Version 8.2.7	
System	Linux debian12 6.1.0-9-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.27-1 (2023-05-08) x86_64
Build Date	Jun 9 2023 19:37:27
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.2/apache2
Loaded Configuration File	/etc/php/8.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.2/apache2/conf.d
Additional .ini files parsed	/etc/php/8.2/apache2/conf.d/10-mysqld.ini, /etc/php/8.2/apache2/conf.d/10-openssl.ini, /etc/php/8.2/apache2/conf.d/10-pdo.ini, /etc/php/8.2/apache2/conf.d/15-xml.ini, /etc/php/8.2/apache2/conf.d/20-apcu.ini, /etc/php/8.2/apache2/conf.d/20-bcmath.ini, /etc/php/8.2/apache2/conf.d/20-calendar.ini, /etc/php/8.2/apache2/conf.d/20-ctype.ini, /etc/php/8.2/apache2/conf.d/20-curl.ini, /etc/php/8.2/apache2/conf.d/20-dom.ini, /etc/php/8.2/apache2/conf.d/20-exif.ini, /etc/php/8.2/apache2/conf.d/20-ffi.ini, /etc/php/8.2/apache2/conf.d/20-fileinfo.ini, /etc/php/8.2/apache2/conf.d/20-ftp.ini, /etc/php/8.2/apache2/conf.d/20-gd.ini, /etc/php/8.2/apache2/conf.d/20-gettext.ini, /etc/php/8.2/apache2/conf.d/20-iconv.ini, /etc/php/8.2/apache2/conf.d/20-intl.ini, /etc/php/8.2/apache2/conf.d/20-mbstring.ini, /etc/php/8.2/apache2/conf.d/20-mysqli.ini, /etc/php/8.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.2/apache2/conf.d/20-phar.ini, /etc/php/8.2/apache2/conf.d/20-posix.ini, /etc/php/8.2/apache2/conf.d/20-readline.ini, /etc/php/8.2/apache2/conf.d/20-shmop.ini, /etc/php/8.2/apache2/conf.d/20-simplexml.ini, /etc/php/8.2/apache2/conf.d/20-soap.ini, /etc/php/8.2/apache2/conf.d/20-sockets.ini, /etc/php/8.2/apache2/conf.d/20-sysmsg.ini, /etc/php/8.2/apache2/conf.d/20-syssem.ini, /etc/php/8.2/apache2/conf.d/20-sysvshm.ini, /etc/php/8.2/apache2/conf.d/20-tokenizer.ini, /etc/php/8.2/apache2/conf.d/20-xmlreader.ini, /etc/php/8.2/apache2/conf.d/20-xmlrpc.ini, /etc/php/8.2/apache2/conf.d/20-xmlwriter.ini, /etc/php/8.2/apache2/conf.d/20-xsl.ini, /etc/php/8.2/apache2/conf.d/20-zip.ini
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	AP420220829.NTS
PHP Extension Build	API20220829.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
Zend Max Execution Timers	disabled
IPv6 Support	enabled

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.

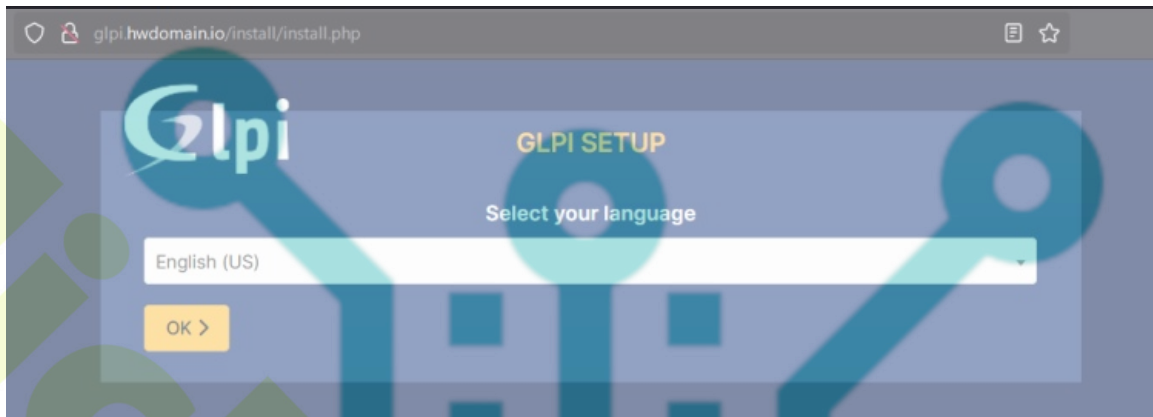
```
root@debian12:~#
root@debian12:~# sudo nano /etc/apache2/sites-available/glpi.conf
root@debian12:~#
root@debian12:~# sudo a2ensite glpi.conf
Site glpi already enabled
root@debian12:~#
root@debian12:~# sudo apachectl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, please
make sure your hostname is reachable and you've globally to suppress this message
Syntax OK
root@debian12:~# sudo systemctl restart apache2
root@debian12:~#
```

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
```

```
root@debian12:~#  
root@debian12:~# sudo apt install certbot python3-certbot-apache  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  augeas-lenses libaugeas0 python3-acme python3-augeas python3-certbot python3-c  
  python3-cryptography python3-distro python3-icu python3-josepy python3-openssl  
Suggested packages:  
  augeas-doc python-certbot-doc python3-certbot-nginx augeas-tools python-acme-d  
  python-cryptography-doc python3-cryptography-vectors python-openssl-doc python  
The following NEW packages will be installed:  
  augeas-lenses certbot libaugeas0 python3-acme python3-augeas python3-certbot py  
  python3-configargparse python3-configobj python3-cryptography python3-distro py  
  python3-parsedatetime python3-rfc3339 python3-tz  
0 upgraded, 18 newly installed, 0 to remove and 34 not upgraded.  
Need to get 2,675 kB of archives.  
After this operation, 11.3 MB of additional disk space will be used.  
Do you want to continue? [Y/n] Y
```

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.

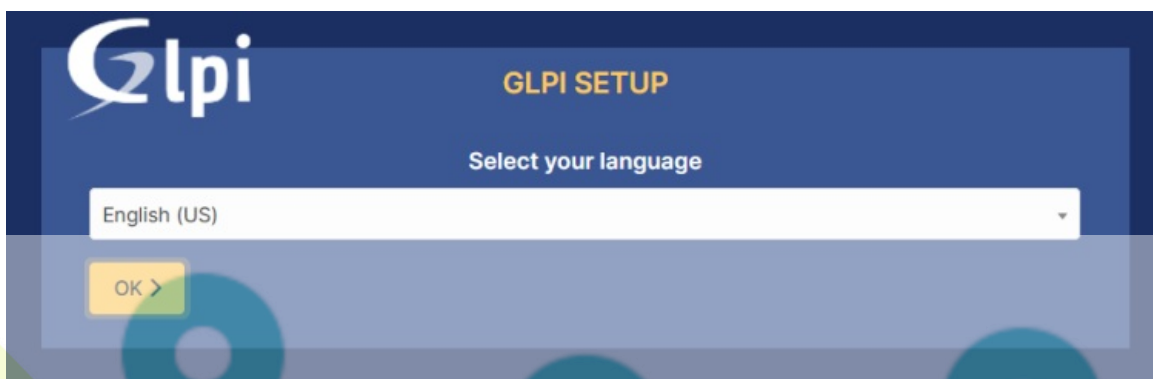
```
sudo certbot --apache --agree-tos --redirect --hsts --staple-ocsp --email alice@example.io -d glpi.example.io
```

Now your SSI/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**.

GLPI SETUP

Step 0

Checking of the compatibility of your environment with the execution of GLPI

TEST DONE	RESULTS
Required PHP Parser	✓
Required Sessions configuration	✓
Required Allocated Memory	✓
Required mysqli extension	✓
Required PHP core extensions	✓
Required curl extension <i>Required for remote access to resources (inventory agent requests, marketplace, RSS feeds, ...).</i>	✓
Required gd extension <i>Required for image handling.</i>	✓
Required intl extension <i>Required for internationalization.</i>	✓
Required zlib extension <i>Required for the handling of compressed communication with inventory agents, installation of gzip packages from the marketplace, and PDF generation.</i>	✓
Required Sodium ChaCha20-Poly1305 size constant <i>Enable usage of ChaCha20-Poly1305 encryption required by GLPI. This is provided by libsodium 1.0.12 and newer.</i>	✓
Required Permissions for log files	✓
Required Permissions for GLPI data directories	✓
Suggested PHP supported version <i>An officially supported PHP version should be used to get the benefits of security and bug fixes.</i>	✓

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

GLPI SETUP

Step 1

Database connection setup

SQL Server (MariaDB or MySQL)

127.0.0.1

SQL User

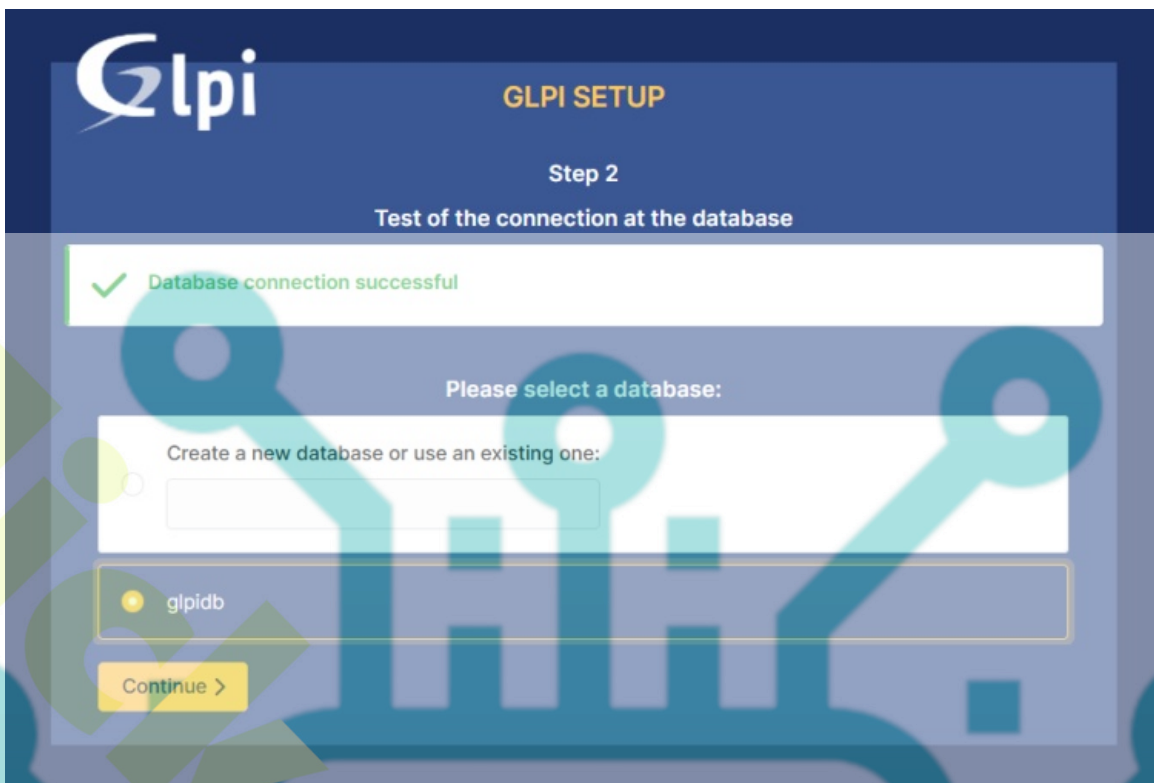
glpi

SQL Password

.....

Continue >

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



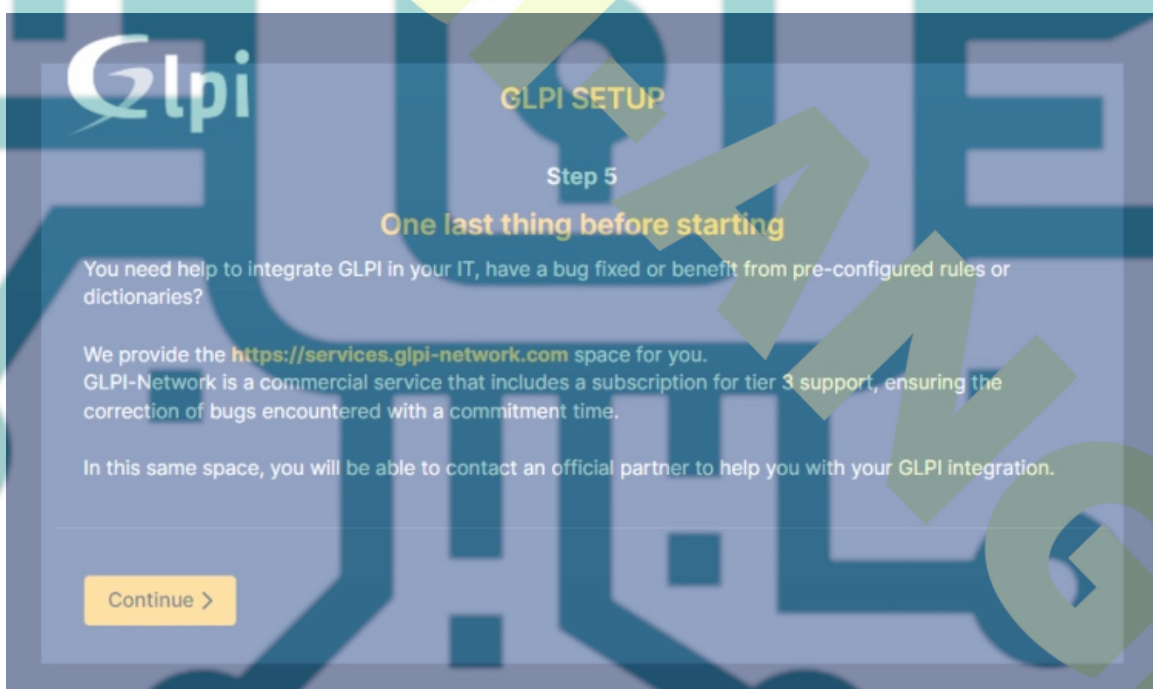
The GLPI database will be initialized, click **Continue**.



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

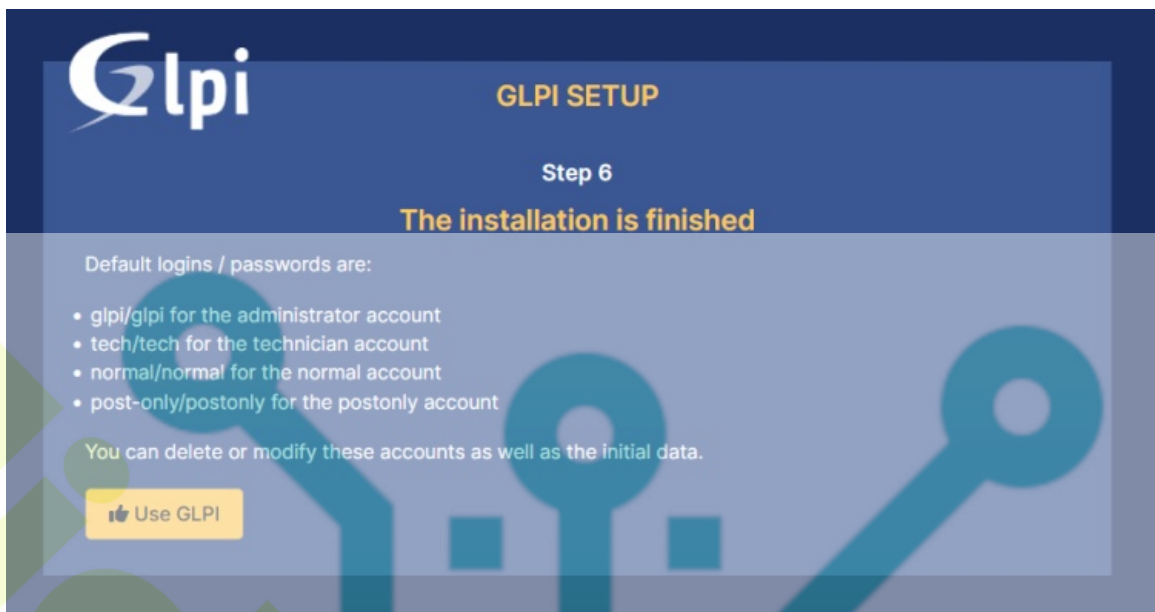


Select **Continue** to go to the next section.



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

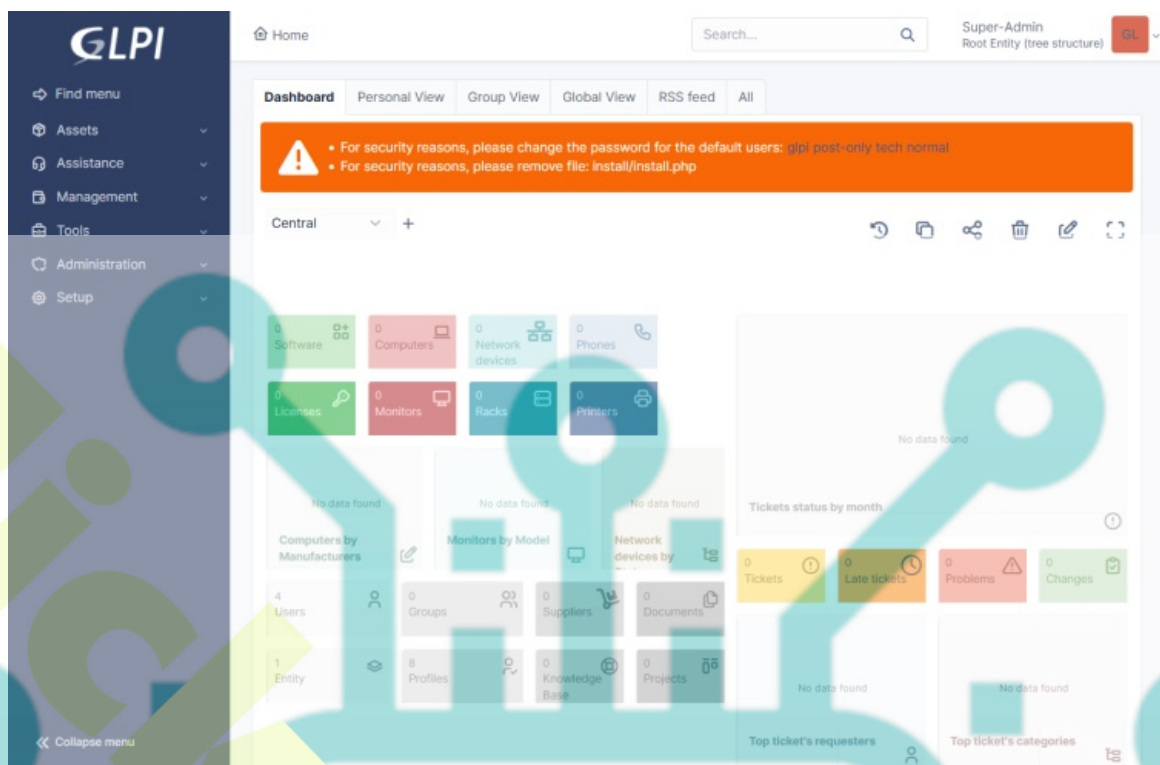
Click **Use GLPI** to finish your installation.



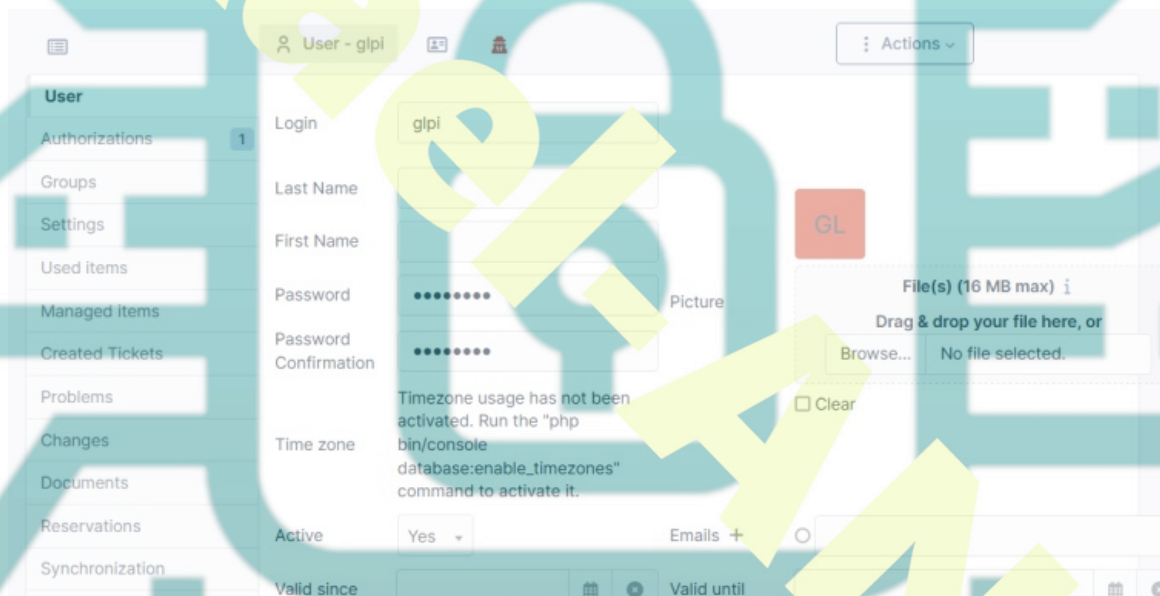
Now you should get the GLPI login page. Input the default user glpi with password glpi, then click **Sign In**.



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.