

How To Install and Secure phpMyAdmin on Debian 9



Posted September 5, 2018 30.2k APPLICATIONS PHP APACHE DATABASES DEBIAN 9 DEBIAN

By: Brennen Bearnes By: Mark Drake

Not using **Debian 9**? Choose a different version:

CentOS 7	>
Ubuntu 18.04	>
Ubuntu 16.04	>

Introduction

While many users need the functionality of a database management system like MariaDB, they may not feel comfortable interacting with the system solely from the MariaDB prompt.

phpMyAdmin was created so that users can interact with MariaDB through a web interface. In this guide, we'll discuss how to install and secure phpMyAdmin so that you can safely use it to manage your databases on a Debian 9 system.

Prerequisites

Before you get started with this guide, you need to have some basic steps completed.

First, we'll assume that your server has a non-**root** user with `sudo` privileges, as well as a firewall configured with `ufw`, as described in the initial server setup guide for Debian 9.

We're also going to assume that you've completed a LAMP (Linux, Apache, MariaDB, and PHP) installation on your Debian 9 server. If you've not yet done this, follow our guide on installing a LAMP stack on Debian 9 to set this up.

SCROLL TO TOP

Finally, there are important security considerations when using software like phpMyAdmin, since it:

- Communicates directly with your MariaDB installation
- Handles authentication using MariaDB credentials
- Executes and returns results for arbitrary SQL queries

For these reasons, and because it is a widely-deployed PHP application which is frequently targeted for attack, you should never run phpMyAdmin on remote systems over a plain HTTP connection. If you do not have an existing domain configured with an SSL/TLS certificate, you can follow this [guide on securing Apache with Let's Encrypt on Debian 9](#). This will require you to [register a domain name](#), [create DNS records for your server](#), and [set up an Apache Virtual Host](#).

Once you are finished with these steps, you're ready to get started with this guide.

Step 1 — Installing phpMyAdmin

To get started, we will install phpMyAdmin from the default Debian repositories.

This is done by updating your server's package index and then using the `apt` packaging system to pull down the files and install them on your system:

```
$ sudo apt update
$ sudo apt install phpmyadmin php-mbstring php-gettext
```

This will ask you a few questions in order to configure your installation correctly.

Warning: When the prompt appears, “apache2” is highlighted, but **not** selected. If you do not hit `SPACE` to select Apache, the installer will *not* move the necessary files during installation. Hit `SPACE`, `TAB`, and then `ENTER` to select Apache.

- For the server selection, choose `apache2`
- Select `Yes` when asked whether to use `dbconfig-common` to set up the database
- You will then be asked to choose and confirm a MySQL application password for phpMyAdmin

Note: MariaDB is a community-developed fork of MySQL, and although the two programs are closely related, they are not completely interchangeable. While phpMyAdmin was designed specifically for managing MySQL databases and makes reference to MySQL in various dialogue boxes, rest assured that your installation of MariaDB will work correctly with phpMyAdmin.

The installation process adds the phpMyAdmin Apache configuration file into the `/etc/apache2/conf-enabled/` directory, where it is read automatically. The only thing you need to do is explicitly enable the `mbstring` PHP extension which is used to manage non-ASCII strings and convert string encodings. Do this by typing: [SCROLL TO TOP](#)

```
$ sudo phpenmod mbstring
```

Afterwards, restart Apache for your changes to be recognized:

```
$ sudo systemctl restart apache2
```

phpMyAdmin is now installed and configured. However, before you can log in and begin managing your MariaDB databases, you will need to ensure that your MariaDB users have the privileges required for interacting with the program.

Step 2 — Adjusting User Authentication and Privileges

When you installed phpMyAdmin onto your server, it automatically created a database user called `phpmyadmin` which performs certain underlying processes for the program. Rather than logging in as this user with the administrative password you set during installation, it's recommended that you log in using a different account.

In new installs on Debian systems, the **root** MariaDB user is set to authenticate using the `unix_socket` plugin by default rather than with a password. This allows for some greater security and usability in many cases, but it can also complicate things when you need to allow an external program (e.g., phpMyAdmin) administrative rights through this user. Because the server uses the **root** account for tasks like log rotation and starting and stopping the server, it is best not to change the **root** account's authentication details. Since phpMyAdmin requires users to authenticate with a password, you will need to create a new MariaDB account in order to access the interface.

If you followed the prerequisite tutorial [on installing a LAMP stack](#) and created a MariaDB user account as described in Step 2, you can just log in to phpMyAdmin under that account using the password you created when you set it up by visiting this link:

https://your_domain_or_IP/phpmyadmin

If you haven't created a MariaDB user, or if you have but you'd like to create another user just for the purpose of managing databases through phpMyAdmin, continue on with this section to learn how to set one up.

Begin by opening up the MariaDB shell:

```
$ sudo mariadb
```

Note: If you have password authentication enabled, as you would if you've already created a new user account for your MariaDB server, you will need to use a different command to access the MariaDB shell. The following will run

SCROLL TO TOP

your MariaDB client with regular user privileges, and you will only gain administrator privileges within the database by authenticating:

```
$ mariadb -u user -p
```

From there, create a new user and give it a strong password:

```
MariaDB [(none)]> CREATE USER 'sammy'@'localhost' IDENTIFIED BY 'password';
```

Then, grant your new user appropriate privileges. For example, you could grant the user privileges to all tables within the database, as well as the power to add, change, and remove user privileges, with this command:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'sammy'@'localhost' WITH GRANT OPTION;
```

Following that, exit the MariaDB shell:

```
MariaDB [(none)]> exit
```

You can now access the web interface by visiting your server's domain name or public IP address, followed by `/phpmyadmin`:

```
https://your_domain_or_IP/phpmyadmin
```



Welcome to phpMyAdmin

Language

English ▼

Log in 

Username:

sammy

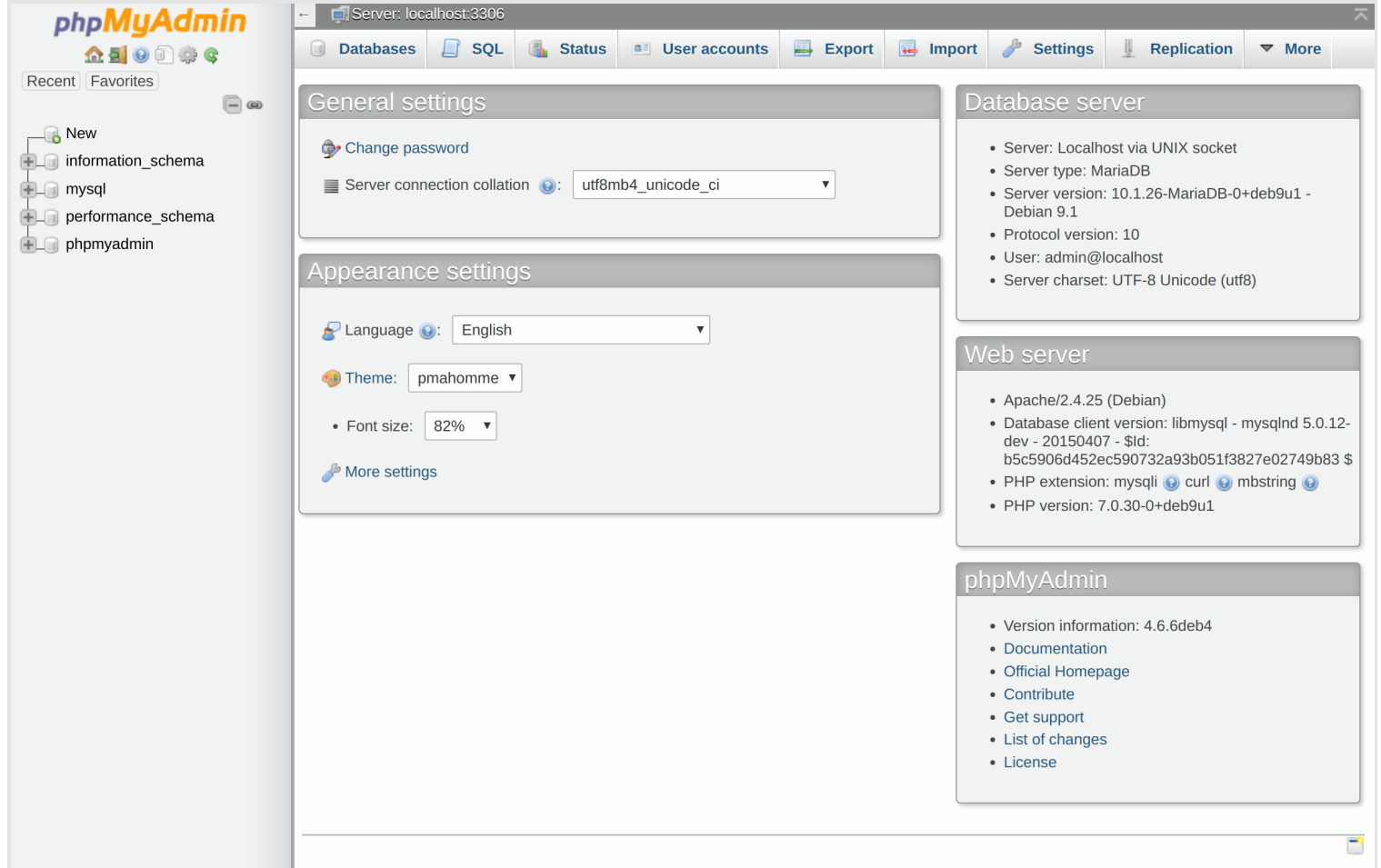
Password:

|

Go

Log in to the interface with the username and password you configured.

When you log in, you'll see the user interface, which will look something like this:



Now that you're able to connect and interact with phpMyAdmin, all that's left to do is harden your system's security to protect it from attackers.

Step 3 — Securing Your phpMyAdmin Instance

Because of its ubiquity, phpMyAdmin is a popular target for attackers, and you should take extra care to prevent unauthorized access. One of the easiest ways of doing this is to place a gateway in front of the entire application by using Apache's built-in `.htaccess` authentication and authorization functionalities.

To do this, you must first enable the use of `.htaccess` file overrides by editing your Apache configuration file.

Edit the linked file that has been placed in your Apache configuration directory:

```
$ sudo nano /etc/apache2/conf-available/phpmyadmin.conf
```

Add an `AllowOverride All` directive within the `<Directory /usr/share/phpmyadmin>` section of the configuration file, like this:

```
/etc/apache2/conf-available/phpmyadmin.conf
```

```
<Directory /usr/share/phpmyadmin>
    Options FollowSymLinks
    DirectoryIndex index.php
    AllowOverride All
    . . .
```

SCROLL TO TOP

When you have added this line, save and close the file.

To implement the changes you made, restart Apache:

```
$ sudo systemctl restart apache2
```

Now that you have enabled `.htaccess` use for your application, you need to create one to actually implement some security.

In order for this to be successful, the file must be created within the application directory. You can create the necessary file and open it in your text editor with root privileges by typing:

```
$ sudo nano /usr/share/phpmyadmin/.htaccess
```

Within this file, enter the following information:

```
/usr/share/phpmyadmin/.htaccess
```

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /etc/phpmyadmin/.htpasswd
Require valid-user
```

Here is what each of these lines mean:

- **AuthType Basic** : This line specifies the authentication type that you are implementing. This type will implement password authentication using a password file.
- **AuthName** : This sets the message for the authentication dialog box. You should keep this generic so that unauthorized users won't gain any information about what is being protected.
- **AuthUserFile** : This sets the location of the password file that will be used for authentication. This should be outside of the directories that are being served. We will create this file shortly.
- **Require valid-user** : This specifies that only authenticated users should be given access to this resource. This is what actually stops unauthorized users from entering.

When you are finished, save and close the file.

The location that you selected for your password file was `/etc/phpmyadmin/.htpasswd`. You can now create this file and pass it an initial user with the `htpasswd` utility:

```
$ sudo htpasswd -c /etc/phpmyadmin/.htpasswd username
```

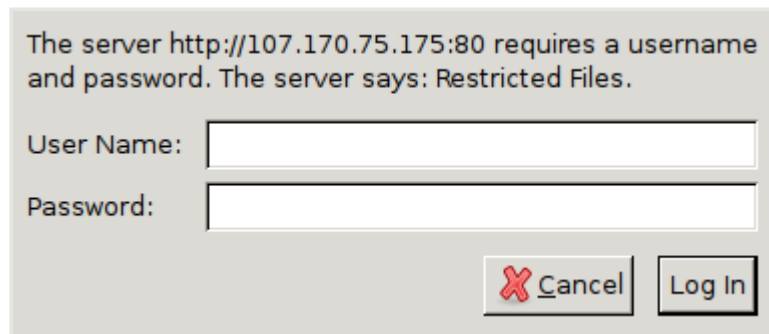
You will be prompted to select and confirm a password for the user you are creating. Afterwards, the file is created with the hashed password that you entered.

If you want to enter an additional user, you need to do so **without** the `-c` flag, like this:

```
$ sudo htpasswd /etc/phpmyadmin/.htpasswd additionaluser
```

Now, when you access your phpMyAdmin subdirectory, you will be prompted for the additional account name and password that you just configured:

`https://domain_name_or_IP/phpmyadmin`



After entering the Apache authentication, you'll be taken to the regular phpMyAdmin authentication page to enter your MariaDB credentials. This setup adds an additional layer of security, which is desirable since phpMyAdmin has suffered from vulnerabilities in the past.

Conclusion

You should now have phpMyAdmin configured and ready to use on your Debian 9 server. Using this interface, you can easily create databases, users, tables, etc., and perform the usual operations like deleting and modifying structures and data.

By: Brennen Bearnese By: Mark Drake

♥ Upvote (2)

📌 Subscribe

🔗 Share



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

[How To Sync and Share Your Files with Seafile on Debian 9](#)

[How To Deploy a PHP Application with Kubernetes on Ubuntu 16.04](#)

[How To Install YunoHost on Debian 9](#)

[How To Install and Configure an Apache ZooKeeper Cluster on Ubuntu 18.04](#)


[How To Sync and Share Your Files with Seafile on Ubuntu 18.04](#)

4 Comments

Leave a comment...

[Log In to Comment](#)

 [agupte](#) November 7, 2018

 Your previous tutorial which installed php installed 5.6 and I think this installs phpmyadmin for php7. So I am getting an error:

[SCROLL TO TOP](#)

^ [mdrake](#) MOD November 7, 2018

o Hello [@agupte!](#)

I'm having some trouble replicating your issue. What previous tutorial are you referring to? From what I can tell, the [How To Install Linux, Apache, MariaDB, PHP \(LAMP\) stack on Debian 9](#) prerequisite installs PHP 7.0, and **php-mbstring** is included in the **mods-available** directory after installing phpMyAdmin.



How To Install Linux, Apache, MariaDB, PHP (LAMP) stack on Debian 9

by Mark Drake

A "LAMP" stack is a group of open source software that is typically installed together to enable a server to host dynamic websites and web apps. This stack typically consists of the Linux operating system, the Apache web server, a MariaDB database, and PHP, a dynamic content processor. This

^ [agupte](#) November 8, 2018

1 I solved the problem by using:
apt-get install php5.6-mbstring

Thanx for asking.

^ [altern4tive](#) December 16, 2018

o I get this error while installing phpmyadmin.

An error occurred while installing the database:

mysql said: ERROR 1064 (42000) at line 1: You have an error in your SQL syntax; check th

SCROLL TO TOP

corresponds to your MySQL server version for the right syntax to use near 'IDENTIFIED BY '9rAxt8Hn7rSU'" at line 1 .

It actually fit the phpmyadmin user password. i've try with different one, and with letting it blank, but I'm always getting the same error. Tried to remove and purge, and same thing again.

I've looked it up on google, the only thing I found said to replace "timestamp(14)" with "timestamp" in /usr/share/dbconfig-common/data/phpmyadmin/install/mysql"

But I have no timestamp(14) in mine...

Any idea? :/



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)