How To Install and Secure phpMyAdmin on Ubuntu 18.04

Introduction

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

phpMyAdmin was created so that users can interact with MySQL 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 an Ubuntu 18.04 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 config with ufw, as described in the initial server setup guide for Ubuntu 18.04.

We're also going to assume that you've completed a LAMP (Linux, Apache, MySQL, and PHP) installati on your Ubuntu 18.04 server. If this is not completed yet, you can follow this guide on installing a LAM stack on Ubuntu 18.04.

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

- Communicates directly with your MySQL installation
- Handles authentication using MySQL 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 fo attack, you should never run phpMyAdmin on remote systems over a plain HTTP connection. If you do have an existing domain configured with an SSL/TLS certificate, you can follow this guide on securing Apache with Let's Encrypt on Ubuntu 18.04. This will require you to register a domain name, create DN 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 Ubuntu repositories.

This is done by updating your server's package index and then using the apt packaging system to pull 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 s Apache, the installer will *not* move the necessary files during installation. Hit SPACE, TAB, and then ENTER 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

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 you can do by typing:

```
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 interacting wit your MySQL databases, you will need to ensure that your MySQL 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 t user with the administrative password you set during installation, it's recommended that you log in as either your **root** MySQL user or as a user dedicated to managing databases through the phpMyAdmin interface.

Configuring Password Access for the MySQL Root Account

In Ubuntu systems running MySQL 5.7 (and later versions), the **root** MySQL user is set to authenticate the auth_socket plugin by default rather than with a password. This allows for some greater security ar usability in many cases, but it can also complicate things when you need to allow an external program

like phpMyAdmin — to access the user.

In order to log in to phpMyAdmin as your **root** MySQL user, you will need to switch its authentication

method from auth_socket to mysql_native_password if you haven't already done so. To do this, open up

MySQL prompt from your terminal:

```
sudo mysql
```

-

Next, check which authentication method each of your MySQL user accounts use with the following

command:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

-

Output

```
+------------------+------------------------------------------+-----------------------+-----------+
| user             | authentication_string                    | plugin                | host      |
+------------------+------------------------------------------+-----------------------+-----------+
| root             |                                          | auth_socket           | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password |
localhost |
| debian-sys-maint | *8486437DE5F65ADC4A4B001CA591363B64746D4C | mysql_native_password |
localhost |
| phpmyadmin       | *5FD2B7524254B7F81B32873B1EA6D681503A5CA9 | mysql_native_password |
localhost |
+------------------+------------------------------------------+-----------------------+-----------+
5 rows in set (0.00 sec)
```

In this example, you can see that the **root** user does in fact authenticate using the auth_socket plugin. T

configure the **root** account to authenticate with a password, run the following ALTER USER command.

sure to change password to a strong password of your choosing:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
```

-

Then, run FLUSH PRIVILEGES which tells the server to reload the grant tables and put your new change

effect:

```
FLUSH PRIVILEGES;
```

-

Check the authentication methods employed by each of your users again to confirm that **root** no longe

authenticates using the auth_socket plugin:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

-

Output

```
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | mysql_native_password |
localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password |
localhost |
| debian-sys-maint | *8486437DE5F65ADC4A4B001CA591363B64746D4C | mysql_native_password |
localhost |
| phpmyadmin       | *5FD2B7524254B7F81B32873B1EA6D681503A5CA9 | mysql_native_password |
localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
5 rows in set (0.00 sec)
```

You can see from this output that the **root** user will authenticate using a password. You can now log ir

the phpMyAdmin interface as your **root** user with the password you've set for it here.

Configuring Password Access for a Dedicated MySQL User

Alternatively, some may find that it better suits their workflow to connect to phpMyAdmin with a dedic

user. To do this, open up the MySQL shell once again:

```
sudo mysql
```

-

Note: If you have password authentication enabled, as described in the previous section, you will need to use a different command to access the MySQL shell. The following will run your MySQL client with regular user privileges, and you will only gain administrator privileges within the database by authenticating:

```
mysql -u root -p
```

-

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

```
CREATE USER 'sammy'@'localhost' IDENTIFIED BY 'password';
```

-

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

```
GRANT ALL PRIVILEGES ON *.* TO 'sammy'@'localhost' WITH GRANT OPTION;
```

-

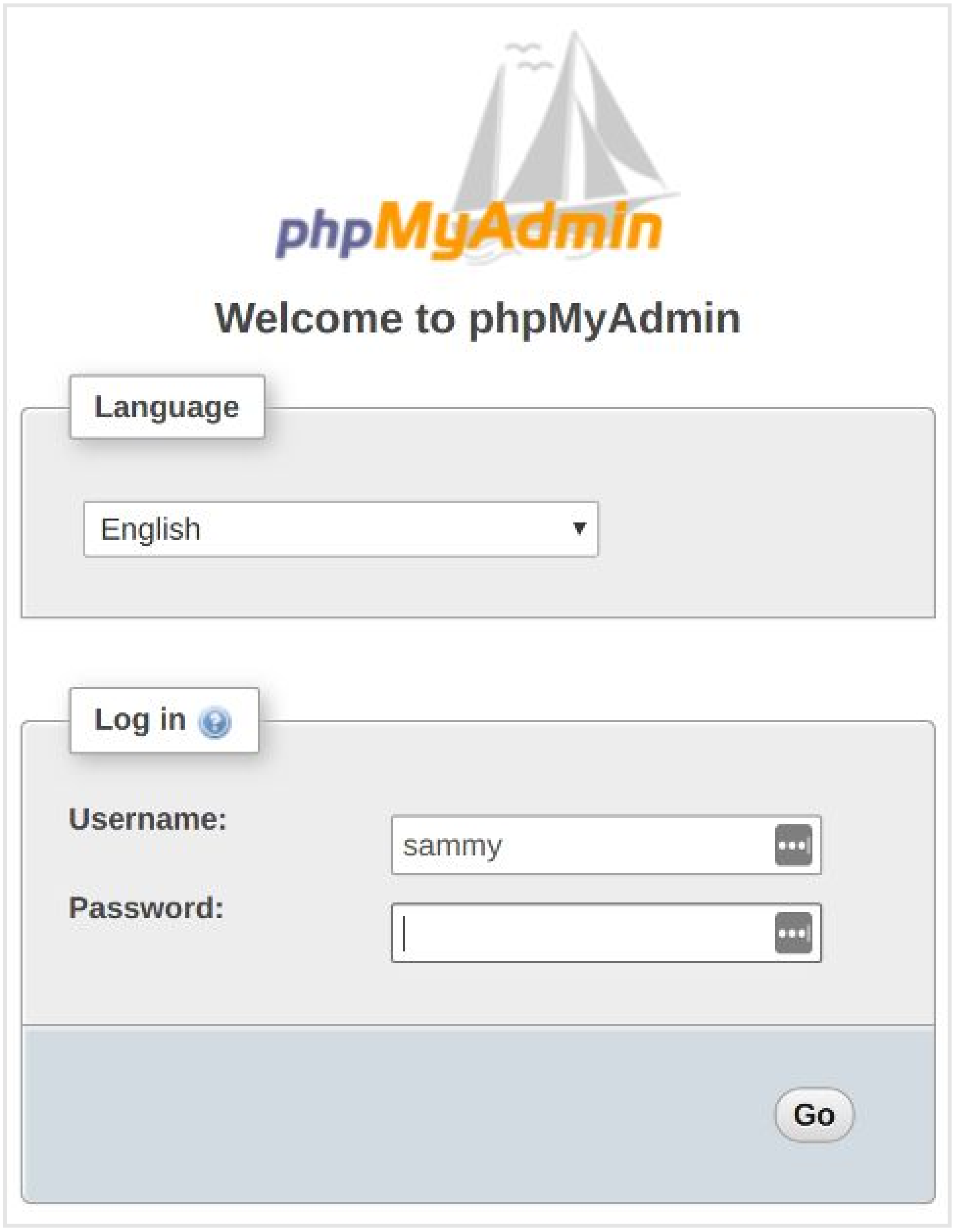
Following that, exit the MySQL shell:

```
exit
```

-

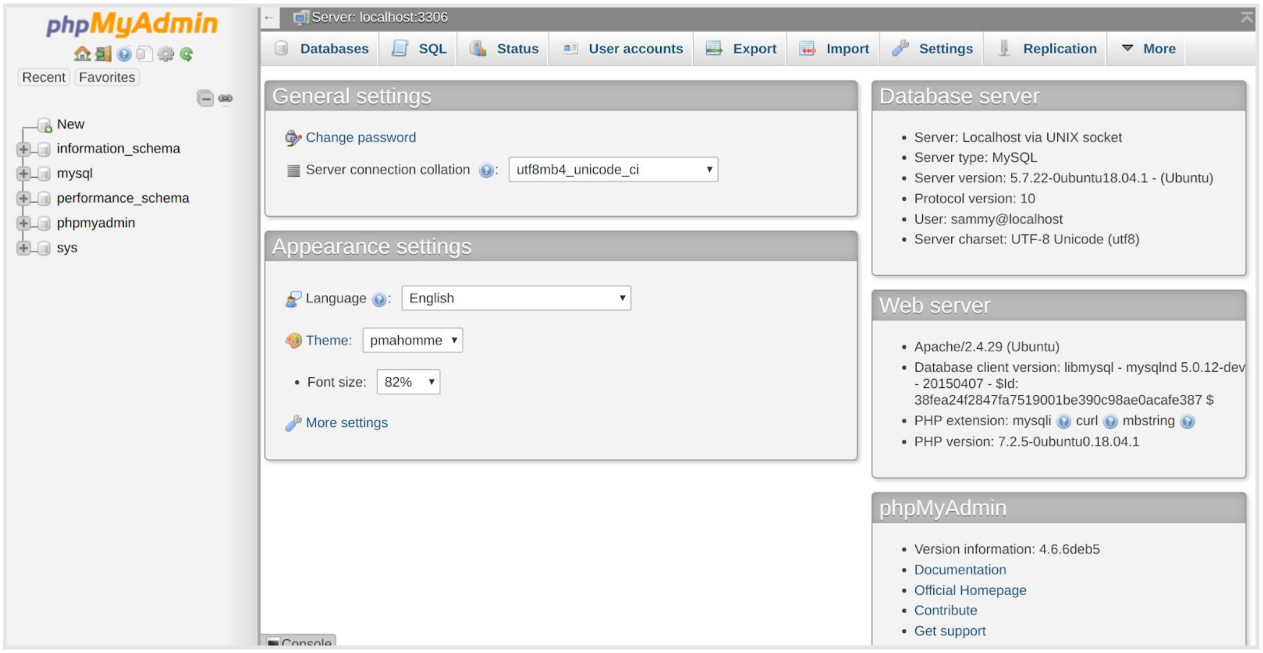
You can now access the web interface by visiting your server’s domain name or public IP address follo by /phpmyadmin:

```
http://your_domain_or_IP/phpmyadmin
```



Log in to the interface, either as **root** or with the new username and password you just configured.

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



Server: localhost:3306

General settings

Change password

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Language: English

Theme: pmahomme

- Font size: 82%

More settings

Database server

- Server: Localhost via UNIX socket
- Server type: MySQL
- Server version: 5.7.22-0ubuntu18.04.1 - (Ubuntu)
- Protocol version: 10
- User: sammy@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.29 (Ubuntu)
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: 38fea24f2847fa7519001be390c98ae0acafe387 $
- PHP extension: mysqli curl mbstring
- PHP version: 7.2.5-0ubuntu0.18.04.1

phpMyAdmin

- Version information: 4.6.6deb5
- Documentation
- Official Homepage
- Contribute
- Get support

Now that you’re able to connect and interact with phpMyAdmin, all that’s left to do is harden your syste

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 t

prevent unauthorized access. One of the easiest ways of doing this is to place a gateway in front of th

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 configuratio

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 configu

file, like this:

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

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

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 imple

some security.

In order for this to be successful, the file must be created within the application directory. You can crea

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 c

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 f

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:

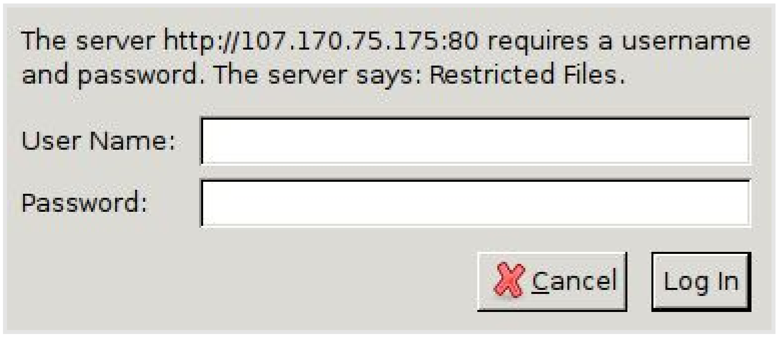
```
sudo htpasswd /etc/phpmyadmin/.htpasswd additionaluser
```

-

Now, when you access your phpMyAdmin subdirectory, you will be prompted for the additional accou

name and password that you just configured:

```
https://domain_name_or_IP/phpmyadmin
```


The server http://107.170.75.175:80 requires a username and password. The server says: Restricted Files.

User Name:

Password:

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

Conclusion

You should now have phpMyAdmin configured and ready to use on your Ubuntu 18.04 server. Using th
interface, you can easily create databases, users, tables, etc., and perform the usual operations like de
and modifying structures and data.