How To Set Up a Remote Database to Optimize Site Performance with MySQL on Ubuntu 18.04

5

Posted November 28, 2018 7.2k MYSQL DATABASES WORDPRESS UBUNTU 18.04 UBUNTU

By: Brian Boucheron By: Mark Drake

Not using **Ubuntu 18.04**? Choose a different version:

Introduction

As your application or website grows, there may come a point where you've outgrown your current server setup. If you are hosting your web server and database backend on the same machine, it may be a good idea to separate these two functions so that each can operate on its own hardware and share the load of responding to your visitors' requests.

In this guide, we'll go over how to configure a remote MySQL database server that your web application can connect to. We will use WordPress as an example in order to have something to work wi

SCROLL TO TOP

technique is widely applicable to any application backed by MySQL.

Prerequisites

Before beginning this tutorial, you will need:

- Two Ubuntu 18.04 servers. Each should have a non-root user with sudo privileges and a UFW firewall enabled, as described in our Initial Server Setup with Ubuntu 18.04 tutorial. One of these servers will host your MySQL backend, and throughout this guide we will refer to it as the **database server**. The other will connect to your database server remotely and act as your web server; likewise, we will refer to it as the **web server** over the course of this guide.
- Nginx and PHP installed **on your web server**. Our tutorial How To Install Linux, Nginx, MySQL, PHP (LEMP stack) in Ubuntu 18.04 will guide you through the process, but note that you should skip Step 2 of this tutorial, which focuses on installing MySQL, as you will install MySQL on your database server.
- MySQL installed **on your database server**. Follow How To Install MySQL on Ubuntu 18.04 to set this up.
- Optionally (but strongly recommended), TLS/SSL certificates from Let's Encrypt installed **on your web server**. You'll need to purchase a domain name and have DNS records set up for your server, but the certificates themselves are free. Our guide How To Secure Nginx with Let's Encrypt on Ubuntu 18.04 will show you how to obtain these certificates.

Step 1 — Configuring MySQL to Listen for Remote Connections

Having one's data stored on a separate server is a good way to expand gracefully after hitting the performance ceiling of a one-machine configuration. It also provides the basic structure necessary to load balance and expand your infrastructure even more at a later time. After installing MySQL by following the prerequisite tutorial, you'll need to change some configuration values to allow connections from other computers.

Most of the MySQL server's configuration changes can be made in the `mysqld.cnf` file, which is stored in the `/etc/mysql/mysql.conf.d/` directory by default. Open up this file on your **database server** with root privileges in your preferred editor. Here, we'll use `nano`:

```
$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

This file is divided into sections denoted by labels in square brackets (`[` and `]`). Find the section labeled `mysqld`:

/etc/mysql/mysql.conf.d/mysqld.cnf

```
. . .
[mysqld]
. . .
```

SCROLL TO TOP

Within this section, look for a parameter called `bind-address`. This tells the database software which network address to listen for connections on.

By default, this is set to `127.0.0.1`, meaning that MySQL is configured to only look for local connections. You need to change this to reference an *external* IP address where your server can be reached.

If both of your servers are in a datacenter with private networking capabilities, use your database server's private network IP. Otherwise, you can use its public IP address:

/etc/mysql/mysql.conf.d/mysqld.cnf

```
[mysqld]
. . .
bind-address = db_server_ip
```

Because you'll connect to your database over the internet, it's recommended that you require encrypted connections to keep your data secure. If you don’t encrypt your MySQL connection, anybody on the network could sniff sensitive information between your web and database servers. To encrypt MySQL connections, add the following line after the `bind-address` line you just updated:

/etc/mysql/mysql.conf.d/mysqld.cnf

```
[mysqld]
. . .
require_secure_transport = on
. . .
```

Save and close the file when you are finished. If you're using `nano`, do this by pressing `CTRL+X`, `Y`, and then `ENTER`.

For SSL connections to work, you will need to create some keys and certificates. MySQL comes with a command that will automatically set these up. Run the following command, which creates the necessary files. It also makes them readable by the MySQL server by specifying the UID of the **mysql** user:

```
$ sudo mysql_ssl_rsa_setup --uid=mysql
```

To force MySQL to update its configuration and read the new SSL information, restart the database:

```
$ sudo systemctl restart mysql
```

To confirm that the server is now listening on the external interface, run the following `netstat` command:

```
$ sudo netstat -plunt | grep mysqld
```

```
Output
```

```
tcp        0      0 db_server_ip:3306     0.0.0.0:*               LISTEN      27328/mysqld
```

`netstat` prints statistics about your server's networking system. This output shows us that a process called `mysqld` is attached to the `db_server_ip` at port `3306`, the standard MySQL port, confirming that the server is listening on the appropriate interface.

Next, open up that port on the firewall to allow traffic through:

```
$ sudo ufw allow mysql
```

Those are all the configuration changes you need to make to MySQL. Next, we will go over how to set up a database and some user profiles, one of which you will use to access the server remotely.

Step 2 — Setting Up a WordPress Database and Remote Credentials

Even though MySQL itself is now listening on an external IP address, there are currently no remote-enabled users or databases configured. Let's create a database for WordPress, and a pair of users that can access it.

Begin by connecting to MySQL as the **root** MySQL user:

```
$ sudo mysql
```

Note: If you have password authentication enabled, as described in Step 3 of the prerequisite MySQL tutorial, you will instead need to use the following command to access the MySQL shell:

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

After running this command, you will be asked for your MySQL **root** password and, after entering it, you'll be given a new `mysql>` prompt.

From the MySQL prompt, create a database that WordPress will use. It may be helpful to give this database a recognizable name so that you can easily identify it later on. Here, we will name it `wordpress`:

```
mysql> CREATE DATABASE wordpress;
```

Now that you've created your database, you next need to create a pair of users. We will create a local-only user as well as a remote user tied to the web server's IP address.

SCROLL TO TOP

First, create your local user, **wpuser**, and make this account only match local connection attempts by using **localhost** in the declaration:

```
mysql> CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'password';
```

Then grant this account full access to the `wordpress` database:

```
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'localhost';
```

This user can now do any operation on the database for WordPress, but this account cannot be used remotely, as it only matches connections from the local machine. With this in mind, create a companion account that will match connections exclusively from your web server. For this, you'll need your web server's IP address.

Please note that you must use an IP address that utilizes the same network that you configured in your `mysqld.cnf` file. This means that if you specified a private networking IP in the `mysqld.cnf` file, you'll need to include the private IP of your web server in the following two commands. If you configured MySQL to use the public internet, you should match that with the web server's public IP address.

```
mysql> CREATE USER 'remotewpuser'@'web_server_ip' IDENTIFIED BY 'password';
```

After creating your remote account, give it the same privileges as your local user:

```
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'remotewpuser'@'web_server_ip';
```

Lastly, flush the privileges so MySQL knows to begin using them:

```
mysql> FLUSH PRIVILEGES;
```

Then exit the MySQL prompt by typing:

```
mysql> exit
```

Now that you've set up a new database and a remote-enabled user, you can move on to testing whether you're able to connect to the database from your web server.

Step 3 — Testing Remote and Local Connections

Before continuing, it's best to verify that you can connect to your database from both the local machine — your database server — and from your web server.

First, test the local connection from your **database server** by attempting to log in with your new account:

```
$ mysql -u wpuser -p
```

When prompted, enter the password that you set up for this account.

If you are given a MySQL prompt, then the local connection was successful. You can exit out again by typing:

```
mysql> exit
```

Next, log into your **web server** to test remote connections:

```
$ ssh sammy@web_server_ip
```

You'll need to install some client tools for MySQL on your web server in order to access the remote database. First, update your local package cache if you haven't done so recently:

```
$ sudo apt update
```

Then install the MySQL client utilities:

```
$ sudo apt install mysql-client
```

Following this, connect to your database server using the following syntax:

```
$ mysql -u remotewpuser -h db_server_ip -p
```

Again, you must make sure that you are using the correct IP address for the database server. If you configured MySQL to listen on the private network, enter your database's private network IP. Otherwise, enter your database server's public IP address.

You will be asked for the password for your **remotewpuser** account. After entering it, and if everything is working as expected, you will see the MySQL prompt. Verify that the connection is using SSL with the following command:

```
mysql> status
```

If the connection is indeed using SSL, the `SSL:` line will indicate this, as shown here:

```
Output
```

```
--------------
mysql  Ver 14.14 Distrib 5.7.18, for Linux (x86_64) using  EditLine wrapper

Connection id:      52
Current database:
Current user:       remotewpuser@203.0.113.111
SSL:          Cipher in use is DHE-RSA-AES256-SHA
Current pager:      stdout
Using outfile:      ''
Using delimiter:    ;
Server version:     5.7.18-0ubuntu0.16.04.1 (Ubuntu)
Protocol version:   10
Connection:     203.0.113.111 via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    utf8
Conn.  characterset:    utf8
TCP port:       3306
Uptime:         3 hours 43 min 40 sec

Threads: 1  Questions: 1858  Slow queries: 0  Opens: 276  Flush tables: 1  Open tables: 184  Queries
--------------
```

After verifying that you can connect remotely, go ahead and exit the prompt:

```
mysql> exit
```

With that, you've verified local access and access from the web server, but you have not verified that other connections will be refused. For an additional check, try doing the same thing from a third server for which you did *not* configure a specific user account in order to make sure that this other server is *not* granted access.

Note that before running the following command to attempt the connection, you may have to install the MySQL client utilities as you did above:

```
$ mysql -u wordpressuser -h db_server_ip -p
```

This should not complete successfully, and should throw back an error that looks similar to this:

```
Output
ERROR 1130 (HY000): Host '203.0.113.12' is not allowed to connect to this MySQL server
```

This is expected, since you haven't created a MySQL user that's allowed to connect from this server, and also desired, since you want to be sure that your database server will deny unauthorizec SCROLL TO TOP your MySQL server.

After successfully testing your remote connection, you can proceed to installing WordPress on your web server.

Step 4 — Installing WordPress

To demonstrate the capabilities of your new remote-capable MySQL server, we will go through the process of installing and configuring WordPress — the popular content management system — on your web server. This will require you to download and extract the software, configure your connection information, and then run through WordPress's web-based installation.

On your **web server**, download the latest release of WordPress to your home directory:

```
$ cd ~
$ curl -O https://wordpress.org/latest.tar.gz
```

Extract the files, which will create a directory called `wordpress` in your home directory:

```
$ tar xzvf latest.tar.gz
```

WordPress includes a sample configuration file which we'll use as a starting point. Make a copy of this file, removing `-sample` from the filename so it will be loaded by WordPress:

```
$ cp ~/wordpress/wp-config-sample.php ~/wordpress/wp-config.php
```

When you open the file, your first order of business will be to adjust some secret keys to provide more security to your installation. WordPress provides a secure generator for these values so that you do not have to try to come up with good values on your own. These are only used internally, so it won't hurt usability to have complex, secure values here.

To grab secure values from the WordPress secret key generator, type:

```
$ curl -s https://api.wordpress.org/secret-key/1.1/salt/
```

This will print some keys to your output. You will add these to your `wp-config.php` file momentarily:

> **Warning!** It is important that you request your own unique values each time. **Do not** copy the values shown here!

```
Output
define('AUTH_KEY',         'L4|2Yh(giOtMLHg3#] DO NOT COPY THESE VALUES %G00o|te^5YG@)'):
define('SECURE_AUTH_KEY',  'DCs-k+MwB90/-E(=!/ DO NOT COPY THESE VALUES +WBzDq:7
define('LOGGED_IN_KEY',    '*0kP!|VS.K=;#fPMlO DO NOT COPY THESE VALUES +&[%8xF*,18c @');
```

SCROLL TO TOP

```
define('NONCE_KEY',        'fmFPF?UJi&(j-{8=$- DO NOT COPY THESE VALUES CCZ?Q+_~1ZU~;G');
define('AUTH_SALT',        '@qA7f}2utTEFNdnbEa DO NOT COPY THESE VALUES t}Vw+8=K%20s=a');
define('SECURE_AUTH_SALT', '%BW6s+d:7K?-`C%zw4 DO NOT COPY THESE VALUES 70U}PO1ejW+7|8');
define('LOGGED_IN_SALT',   '-l>F:-dbcWof%4kKmj DO NOT COPY THESE VALUES 8Ypslin3~d|wLD');
define('NONCE_SALT',       '4J(<`4&&F (WiK9K#] DO NOT COPY THESE VALUES ^ZikS`es#Fo:V6');
```

Copy the output you received to your clipboard, then open the configuration file in your text editor:

```
$ nano ~/wordpress/wp-config.php
```

Find the section that contains the dummy values for those settings. It will look something like this:

/wordpress/wp-config.php

```
. . .
define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');
. . .
```

Delete those lines and paste in the values you copied from the command line.

Next, enter the connection information for your remote database. These configuration lines are at the top of the file, just above where you pasted in your keys. Remember to use the same IP address you used in your remote database test earlier:

/wordpress/wp-config.php

```
. . .
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'remotewpuser');

/** MySQL database password */
define('DB_PASSWORD', 'password');

/** MySQL hostname */
define('DB_HOST', 'db_server_ip');
. . .
```

SCROLL TO TOP

And finally, anywhere in the file, add the following line which tells WordPress to use an SSL connection to our MySQL database:

/wordpress/wp-config.php

```
define('MYSQL_CLIENT_FLAGS', MYSQLI_CLIENT_SSL);
```

Save and close the file.

Next, copy the files and directories found in your `~/wordpress` directory to Nginx's document root. Note that this command includes the `-a` flag to make sure all the existing permissions are carried over:

```
$ sudo cp -a ~/wordpress/* /var/www/html
```

After this, the only thing left to do is modify the file ownership. Change the ownership of all the files in the document root over to **www-data**, Ubuntu's default web server user:

```
$ sudo chown -R www-data:www-data /var/www/html
```

With that, WordPress is installed and you're ready to run through its web-based setup routine.

Step 5 — Setting Up Wordpress Through the Web Interface

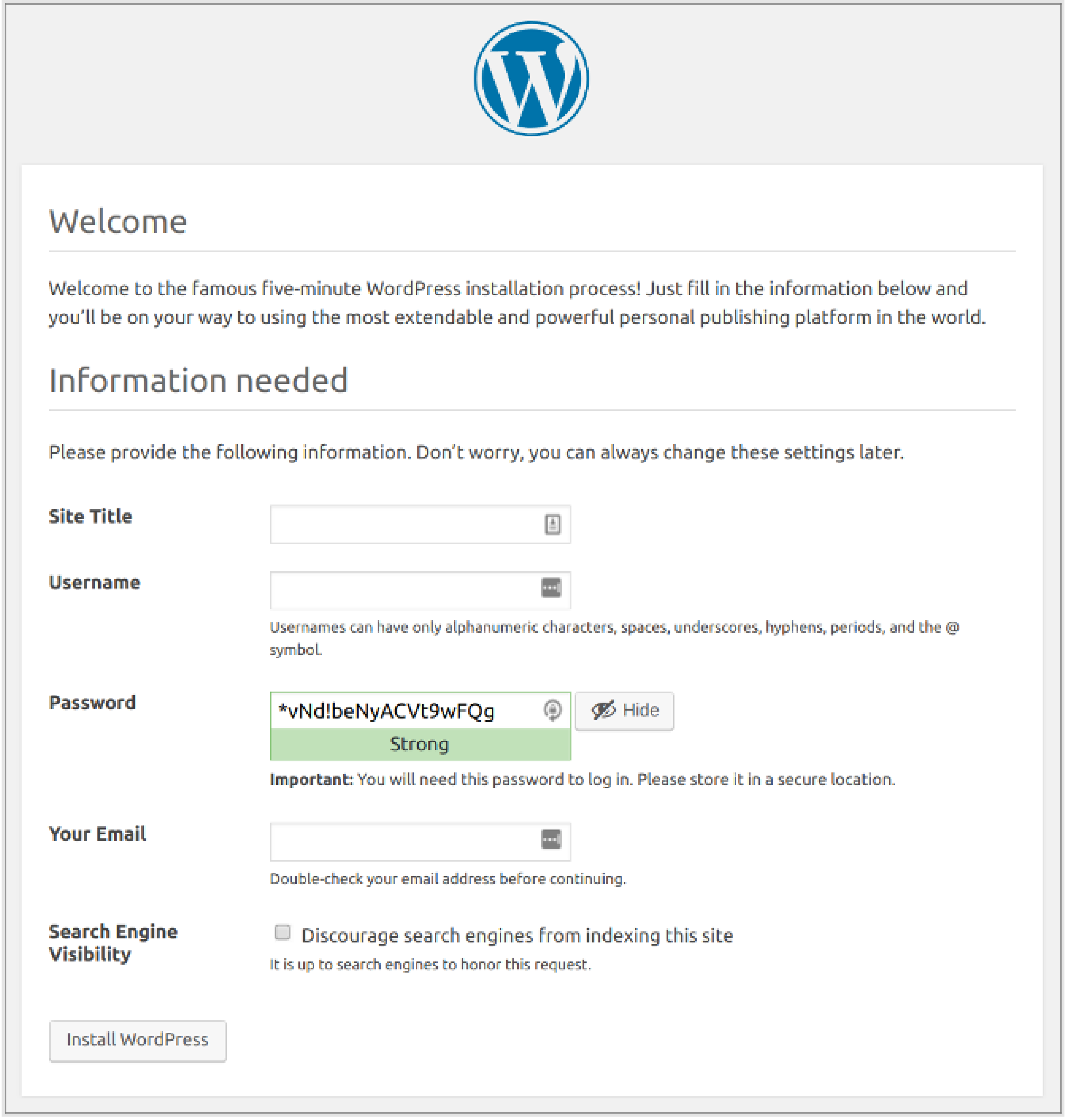
WordPress has a web-based setup process. As you go through it, it will ask a few questions and install all the tables it needs in your database. Here, we will go over the initial steps of setting up WordPress, which you can use as a starting point for building your own custom website that uses a remote database backend.

Navigate to the domain name (or public IP address) associated with your web server:

```
http://example.com
```

You will see a language selection screen for the WordPress installer. Select the appropriate language and click through to the main installation screen:

SCROLL TO TOP


Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password: *vNd!beNyACVt9wFQg — Strong

Important: You will need this password to log in. Please store it in a secure location.

Your Email

Double-check your email address before continuing.

Search Engine Visibility: ☐ Discourage search engines from indexing this site

It is up to search engines to honor this request.

Once you have submitted your information, you will need to log into the WordPress admin interface using the account you just created. You will then be taken to a dashboard where you can customize your new WordPress site.

Conclusion

By following this tutorial, you've set up a MySQL database to accept SSL-protected connections from a remote Wordpress installation. The commands and techniques used in this guide are applicable to any web application written in any programming language, but the specific implementation details will differ. Refer to your application or language's database documentation for more information.

By: Brian Boucheron   By: Mark Drake

Related Tutorials

How To Ensure Code Quality with SonarQube on Ubuntu 18.04

How To Sync and Share Your Files with Seafile on Ubuntu 18.04

How To Troubleshoot Issues in MySQL

How To Set Up Laravel, Nginx, and MySQL with Docker Compose

How To Install and Configure pgAdmin 4 in Server Mode

4 Comments

Leave a comment...

Log In to Comment

Nuwanda *December 6, 2018*

1 To clarify, as far as MYSQL is concerned wordpressuser@localhost and wordpressuser@w
different users?

SCROLL TO TOP

mdrake MOD *December 7, 2018*

0 Hello @Nuwanda!

Yes, the two are indeed separate users, although I can see how having two different users with the same name might be confusing. I've updated the tutorial to make it clearer that they're two different user profiles by renaming them `wpuser@localhost` and `remotewpuser@localhost`.

rickjm540 *December 7, 2018*

1 Nicely put together - thanks for this!

One nice addition - if you add your database server's IP address to the /etc/hosts file on your web server with a name, you can use that name instead of the IP address in your configuration files (esp nice if you have more than one wordpress or other database-driven installation). Then if you move your databases to a different server, you only have one place to change the IP.

propertyadvisor17 *December 22, 2018*

0 This is really help full as I was trying to optimize my site by making its database to a remote one.