WordPress

All docs related to WordPress

WordPress Installation

All docs related to wordpress installation

[How-To] Install Wordpress on Debian 12 LXC

Purpose

The purpose of this How-To article will be to install WordPress and the full LAMP stack on an LXC Container in proxmox. Most of this configuration could also be done on baremetal or a VM of any Debian flavor. We'll be using Debian 12.

Prerequisites

- A LXC with Debian 12 as its base
- User privileges: root or non-root user with sudo privileges

Installation

Step 1: Update the System

Before we start with LAMP installation, we need to update the system packages to the latest versions available.

```
sudo apt-get update -y && sudo apt-get upgrade -y
```

Step 2: Install Apache Web Server

We will start with the Apache web server from the LAMP stack first. To install the Apache Web server execute the following command:

```
sudo apt install apache2 -y
```

Once installed, start and enable the service.

```
sudo systemctl enable apache2 && sudo systemctl start apache2
```

Check if the service is up and running:

```
sudo systemctl status apache2
```

You should receive the following output:

```
root@host:~# sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Thu 2023-08-03 06:02:42 CDT; 22h ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 711 (apache2)
      Tasks: 10 (limit: 4644)
     Memory: 29.7M
        CPU: 4.878s
     CGroup: /system.slice/apache2.service
```

Step 3: Install PHP with Dependencies

Next, we will install PHP. PHP8.2 is by default enabled in the Debian 12 repository, so to install PHP8.2 with extensions, execute the following commands:

```
sudo apt-get install php8.2 php8.2-cli php8.2-common php8.2-imap php8.2-redis php8.2-snmp php8.2-xml
php8.2-mysqli php8.2-zip php8.2-mbstring php8.2-curl libapache2-mod-php -y
```

To check the installed PHP version, execute the following command:

```
php -v
```

You should get the following output:

```
root@host:~# php -v
Created directory: /var/lib/snmp/cert_indexes
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
```

Step 4: Install the MariaDB Database Server

The last of the LAMP stack is the MariaDB database server. To install it, execute the command below.

```
sudo apt install mariadb-server -y
```

Start and enable the mariadb.service with the following commands:

```
sudo systemctl start mariadb && sudo systemctl enable mariadb
```

Check the status of the mariadb.service

```
sudo systemctl status mariadb
```

You should receive the following output:

```
root@host:~# 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 Fri 2023-08-04 05:04:01 CDT; 26s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 8511 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 16 (limit: 4644)
     Memory: 174.3M
        CPU: 907ms
     CGroup: /system.slice/mariadb.service
             └─8511 /usr/sbin/mariadbd
```

Finally, setup the mariadb server with the following command:

```
mariadb-secure-installation
```

The script walks you through a series of prompts that will require you to make changes to the security options that involve the MariaDB database engine.

The first prompt asks you to provide the current root password, and since none has been set up yet, simply hit ENTER on your keyboard.

Next, you will be prompted for the database root password. This ensures that no one can log in as the root user without authentication. So, type ‘Y’ and provide the database root password and confirm it.

Then press ‘Y’ for the subsequent prompts in order to configure the database engine according to the best security practices. This does the following:

- Removes anonymous users from the database server
- Disables remote root login. This ensures that the root user can only log in to the database server from “localhost”
- Remove the test database which comes with MariaDB by default.
- Reloads privilege tables for the changes t take effect immediately.

At this point, you have successfully completed the initial security configuration for MariaDB.

Step 5: Create a WordPress Database and User

Next, we need to create a WordPress database, the WordPress user, and grant the permissions for that user to the database.

```
CREATE USER 'wordpress'@'localhost' IDENTIFIED BY 'YourStrongPasswordHere';
CREATE DATABASE wordpress;
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Be sure to store these credentials/information in a safe credentials manager as they should be treated the same way as a password.

Step 6: Download and Install WordPress

Before we install WordPress, we first need to download it in the default Apache document root:

```
cd /var/www/html

wget https://wordpress.org/latest.zip

unzip latest.zip

rm latest.zip
```

Set the right permissions to files and folders.

```
chown -R www-data:www-data wordpress/

cd wordpress/

find . -type d -exec chmod 755 {} \;

find . -type f -exec chmod 644 {} \;
```

Now, open the **wp-config.php** file with your favorite editor and enter the database credentials you created in the previous step.

```
mv wp-config-sample.php wp-config.php

nano wp-config.php
```

It should look similar to this:

```
// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** Database username */
define( 'DB_USER', 'wordpress' );

/** Database password */
define( 'DB_PASSWORD', 'YourStrongPasswordHere' );
```

Step 7: Create Apache Virtual Host File

Go into the Apache directory and create a configuration file for WordPress.

```
cd /etc/apache2/sites-available/

touch wordpress.conf
```

Open the file, paste the following lines of code, save the file and close it.

```
<VirtualHost *:80>
ServerName yourdomain.com
DocumentRoot /var/www/html/wordpress

<Directory /var/www/html/wordpress>
AllowOverride All
</Directory>

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Enable the Apache configuration for WordPress and rewrite the module.

```
sudo a2enmod rewrite

sudo a2ensite wordpress.conf
```

Check the syntax:

```
apachectl -t
```

You should receive the following output:

```
root@vps:~# apachectl -t
Syntax OK
```

If the syntax is OK, restartd the Apache service.

```
systemctl reload apache2
```

Once the Apache service is restarted, you can finish your WordPress installation at **http://yourdomain.com.**

That was all. You successfully installed and configured WordPress on Debian 12 LXC with the LAMP stack.

Admin Panel Access: HTTP://yourdomain.com/wp-admin

Documentation derived from:

https://www.rosehosting.com/blog/how-to-install-wordpress-on-debian-12/

https://www.cherryservers.com/blog/how-to-install-and-start-using-mariadb-on-ubuntu-20-04