Born2beroot 03: Installing WordPress on a Debian Server

By Mia Combeau

In 42 School Projects
March 13, 2022
10 Min read
Add comment

B

For the bonus part of the Born2beroot project, we will be installing WordPress on out virtual server, as well as another service of our choice. In order to create a functionnal WordPress website, we will need to install an HTTP server, a database manager and PHP. For the free-choice service, we will install and configure Fail2ban as a security measure.

Our virtual server this article series runs Debian 11.2 (bullseye) on VirtualBox 6.1.

Born2beroot : Installation | Configuration | Bonus

Table of Contents

What is WordPress?

WordPress is a content management system (CMS for short) that allows anyone anywhere to create and maintain a functional website, without any particular technical knowledge. Its user-friendliness, its free and open source nature, as well as its ability to adapt to its users' needs makes it a great success: 43% of all websites use it [source].

For our Born2beroot server to be able to host a WordPress website, we need three things: HTTP server software, a database manager, and PHP. We will also need to change some things in our firewall to authorize communication on certain ports.

Installing PHP for WordPress

PHP (PHP: Hypertext Preprocessor) is a very popular open-source programming language for the creation of dynamic web pages via web server. It is essential for the correct operation of WordPress.

One packet isn't enough to install PHP. At minimum, we will need four: `php-common`, `php-cgi`, `php-cli` et `php-mysql`. But we have a problem: APT and Aptitude only have the repositories for PHP version 7.4. But of course, we would like the latest PHP version (8.1 at the time of this writing). We will have to point our package manager to another repository.

We need Sury's repository. To retrieve it, we will need to install cURL, a simple command-line tool that allows us to access an URL withtout going through a browser:

```shell
$ sudo apt update
$ sudo apt install curl
$ sudo curl -sSL https://packages.sury.org/php/README.txt |
sudo bash -x
$ sudo apt update
```

Code language: Shell Session (shell)

Now that APT has the repository, we only need to install PHP version 8.1 and the other packets we need:

```
$ sudo apt install php8.1
$ sudo apt install php-common php-cgi php-cli php-mysql
```

Code language: Shell Session (shell)

To check PHP’s version on the Born2beroot system, let’s do this command:

```
$ php -v
```

Code language: Shell Session (shell)

Installing Lighttpd Web Server for WordPress

Now we need to install a web server program which answers requests via HTTP, the network protocol designed to distribute web content.

The open source web server that we have to choose here is lighttpd (or “lighty“). With a smaller memory footprint than other web servers (like Apache) and smart CPU load management, lighttpd is optimized for speed, all the while remaining secure, compliant and flexible.

However, it is very possible that Apache was installed on our server as a dependency for one of the PHP modules. To avoid conflicts between our web server lighttpd and Apache, the first thing we will do is check if Apache was installed and, if that is the case, uninstall it:

```
$ systemctl status apache2
$ sudo apt purge apache2
```

Code language: Shell Session (shell)

Once Apache uninstalled, we can install lighttpd :

```
$ sudo apt install lighttpd
```

Code language: Shell Session (shell)

Then we will start it, enable it at system startup, and check its version and status with the following commands:

```
$ sudo lighttpd -v
$ sudo systemctl start lighttpd
$ sudo systemctl enable lighttpd
$ sudo systemctl status lighttpd
```

Code language: Shell Session (shell)

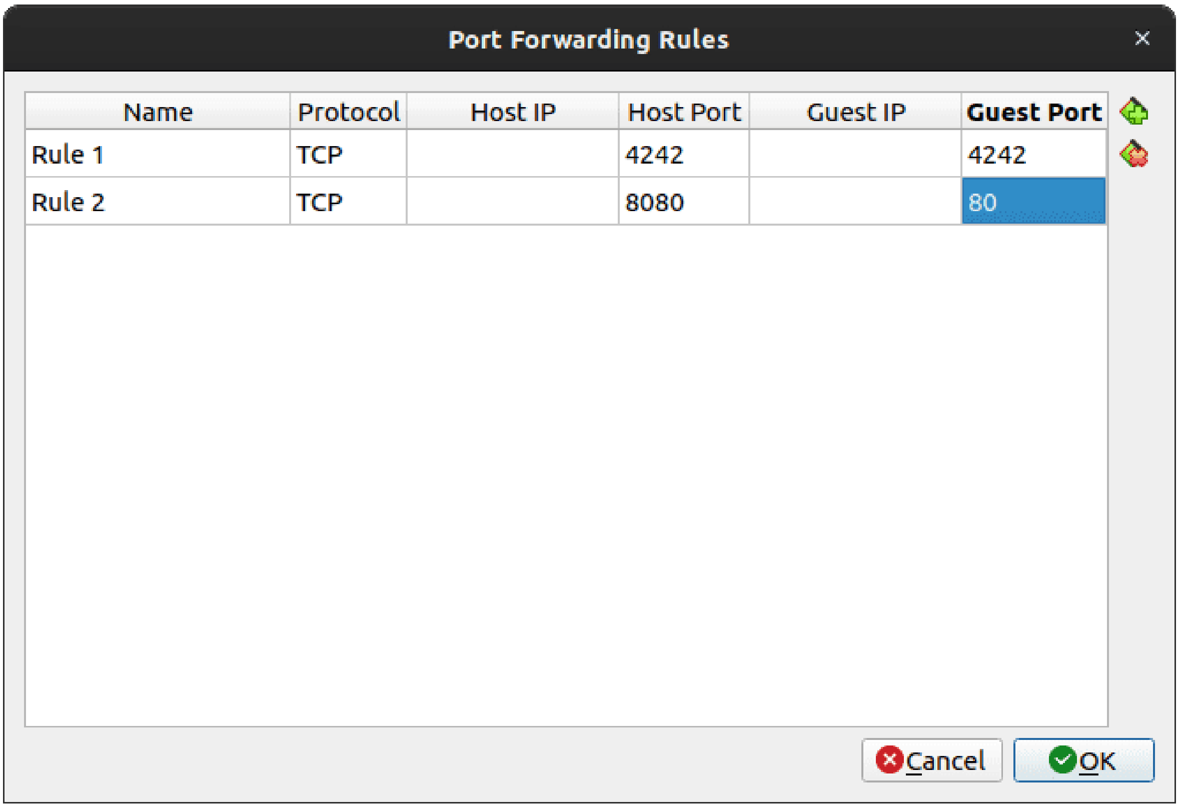
Its status should show active. All that is left to do is authorize HTTP traffic in our firewall settings:

```
$ sudo ufw allow http
$ sudo ufw status
```

Code language: Shell Session (shell)

When we check whether the rule has been added, we should see that port 80 is allowed. Port 80 is the default HTTP port. We also need to do some port forwarding in VirtualBox to be able to access to the virtual machine's port 80 from the outside, like we did before for port 4242:

- Settings >> Network >> Adapter 1 >> Advanced >> Port Forwarding
- Add a rule for Host Port: 8080, guest port: 80, as we don't want host port 80 to be affected.

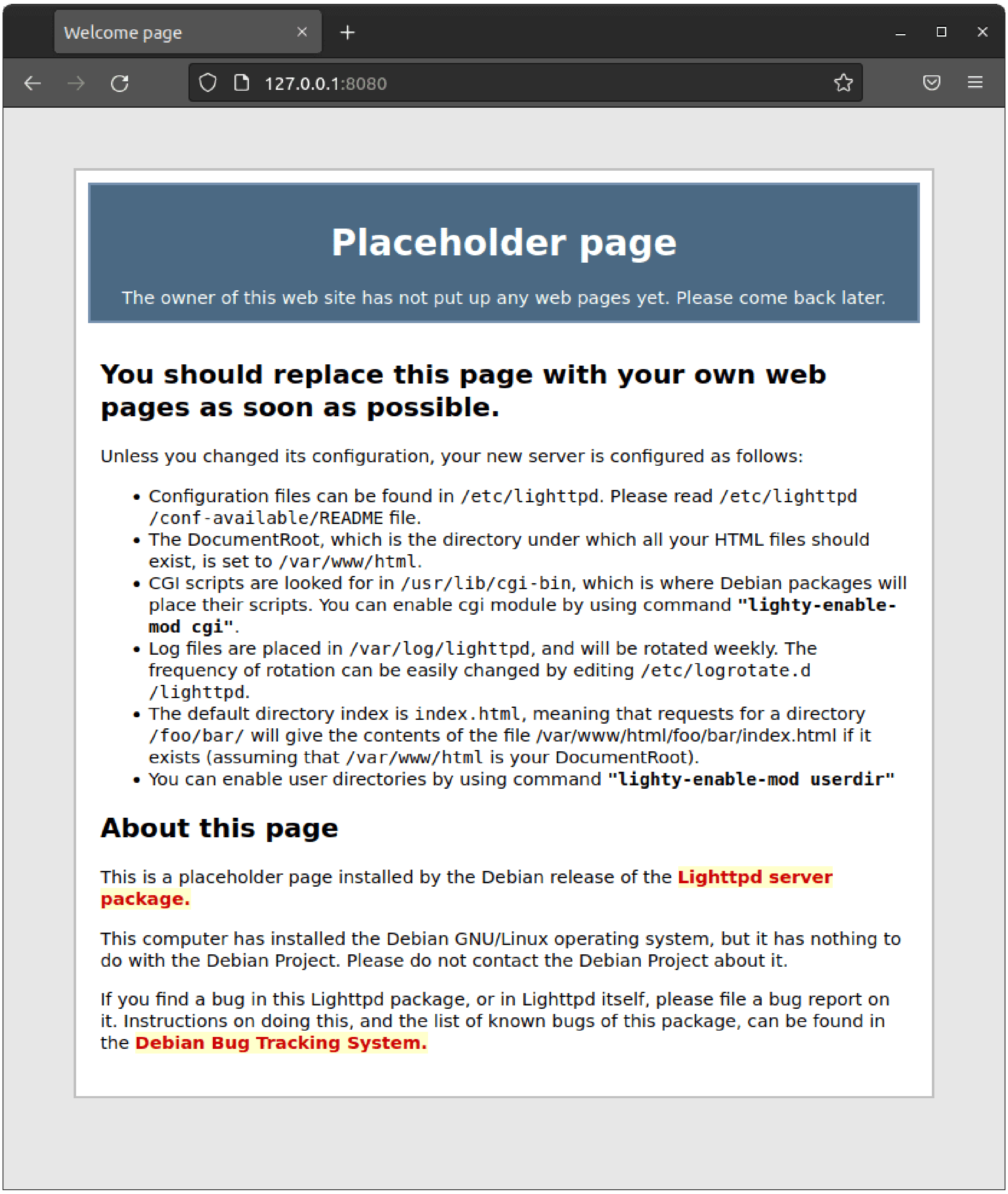


Port Forwarding Rules

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	TCP		4242		4242
Rule 2	TCP		8080		80

Testing Lighttpd Server

Finally, we can do a little test to check that lighttpd is working properly. In a browser on the host machine, we can connect to the following address and port: `http://127.0.0.1:8080` (or `http://localhost:8080`). We should see the lighttpd placeholder page, like this:



Placeholder page

The owner of this web site has not put up any web pages yet. Please come back later.

You should replace this page with your own web pages as soon as possible.

Unless you changed its configuration, your new server is configured as follows:

- Configuration files can be found in `/etc/lighttpd`. Please read `/etc/lighttpd/conf-available/README` file.
- The DocumentRoot, which is the directory under which all your HTML files should exist, is set to `/var/www/html`.
- CGI scripts are looked for in `/usr/lib/cgi-bin`, which is where Debian packages will place their scripts. You can enable cgi module by using command **"lighty-enable-mod cgi"**.
- Log files are placed in `/var/log/lighttpd`, and will be rotated weekly. The frequency of rotation can be easily changed by editing `/etc/logrotate.d/lighttpd`.
- The default directory index is `index.html`, meaning that requests for a directory `/foo/bar/` will give the contents of the file /var/www/html/foo/bar/index.html if it exists (assuming that `/var/www/html` is your DocumentRoot).
- You can enable user directories by using command **"lighty-enable-mod userdir"**

About this page

This is a placeholder page installed by the Debian release of the **Lighttpd server package.**

This computer has installed the Debian GNU/Linux operating system, but it has nothing to do with the Debian Project. Please do not contact the Debian Project about it.

If you find a bug in this Lighttpd package, or in Lighttpd itself, please file a bug report on it. Instructions on doing this, and the list of known bugs of this package, can be found in the **Debian Bug Tracking System.**

We will replace this page with a WordPress website very soon !
Let’s do another quick test. In the virtual machine, let’s create a file named `info.php` in the `/var/www/html` directory like so:

```
$ sudo nano /var/www/html/info.php
```

Code language: Shell Session (shell)

Here we will write a small script to show information about PHP on this server:

```
<?php

phpinfo();

?>
```

Code language: PHP (php)

Now in our host browser, let's go see this file at the following address: `http://127.0.0.1/info.php`.

…And we get a "403 Forbidden" error… What is happening here?

Activating FastCGI

At the dawn of the World Wide Web, a web server would immediately send pre-written HTML pages for each HTTP request it received. Today, with CGI technology and dynamic web pages, the web server does not answer straightaway, but instead transfers the HTTP request data to an external application. The application treats the request and sends the generated HTML code back to the web server which can then answer the request.

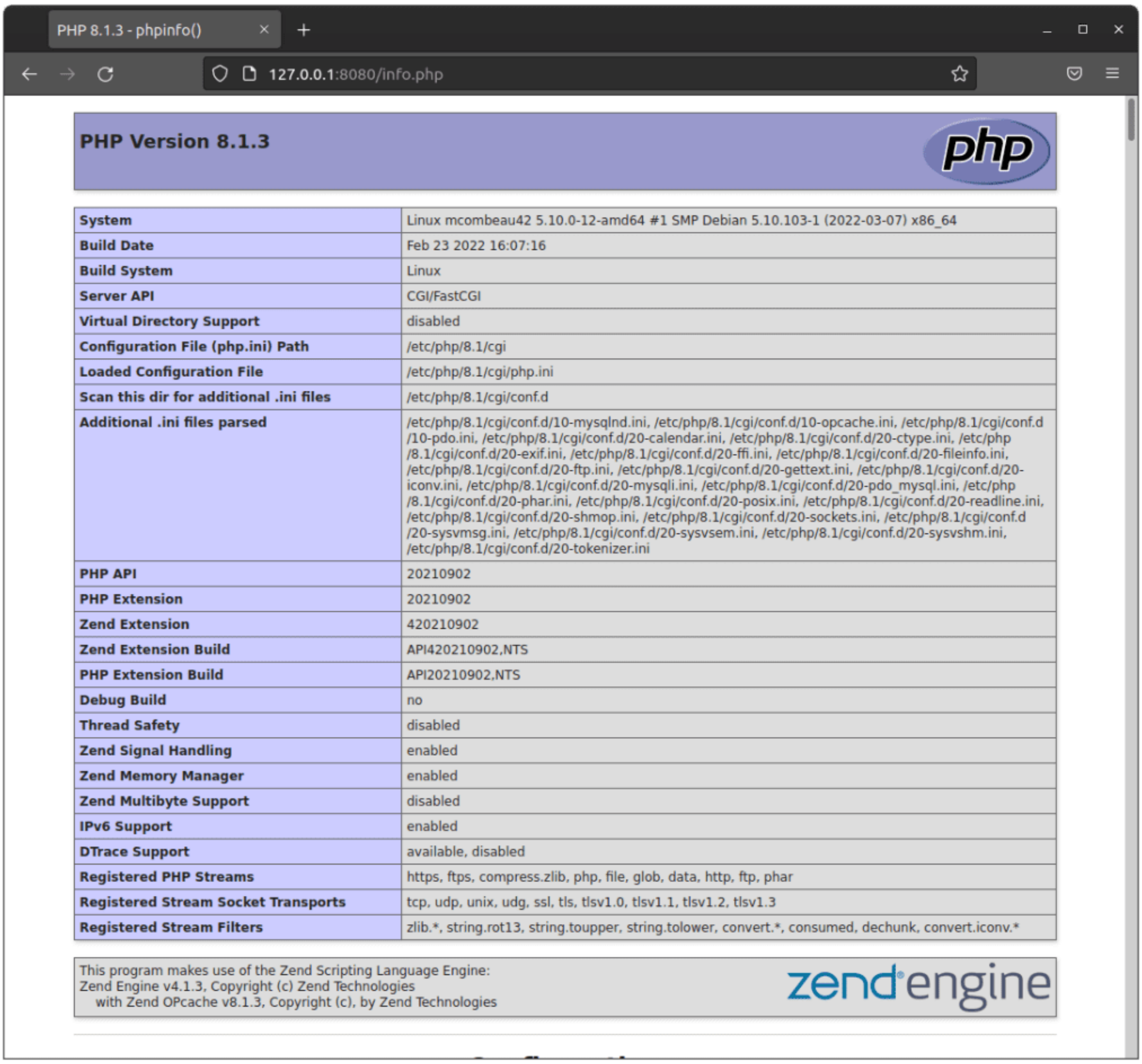
FastCGI (*Fast Common Gateway Interface*) is a binary protocol that allows a web server to interact with external application, in our case, PHP. We need to set up this protocol between lighttpd and PHP in order to be able to access our `info.php` page from a web browser.

So let's activate lighttpd's FastCGI modules with the following commands:

```
$ sudo lighty-enable-mod fastcgi
$ sudo lighty-enable-mod fastcgi-php
$ sudo service lighttpd force-reload
```

Code language: Shell Session (shell)

Now, we should see a page like this when we go to `http://127.0.0.1:8080/info.php`:



PHP Version 8.1.3

System	Linux mcombeau42 5.10.0-12-amd64 #1 SMP Debian 5.10.103-1 (2022-03-07) x86_64
Build Date	Feb 23 2022 16:07:16
Build System	Linux
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/cgi
Loaded Configuration File	/etc/php/8.1/cgi/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/cgi/conf.d
Additional .ini files parsed	/etc/php/8.1/cgi/conf.d/10-mysqlnd.ini, /etc/php/8.1/cgi/conf.d/10-opcache.ini, /etc/php/8.1/cgi/conf.d/10-pdo.ini, /etc/php/8.1/cgi/conf.d/20-calendar.ini, /etc/php/8.1/cgi/conf.d/20-ctype.ini, /etc/php/8.1/cgi/conf.d/20-exif.ini, /etc/php/8.1/cgi/conf.d/20-ffi.ini, /etc/php/8.1/cgi/conf.d/20-fileinfo.ini, /etc/php/8.1/cgi/conf.d/20-ftp.ini, /etc/php/8.1/cgi/conf.d/20-gettext.ini, /etc/php/8.1/cgi/conf.d/20-iconv.ini, /etc/php/8.1/cgi/conf.d/20-mysqli.ini, /etc/php/8.1/cgi/conf.d/20-pdo_mysql.ini, /etc/php/8.1/cgi/conf.d/20-phar.ini, /etc/php/8.1/cgi/conf.d/20-posix.ini, /etc/php/8.1/cgi/conf.d/20-readline.ini, /etc/php/8.1/cgi/conf.d/20-shmop.ini, /etc/php/8.1/cgi/conf.d/20-sockets.ini, /etc/php/8.1/cgi/conf.d/20-sysvmsg.ini, /etc/php/8.1/cgi/conf.d/20-sysvsem.ini, /etc/php/8.1/cgi/conf.d/20-sysvshm.ini, /etc/php/8.1/cgi/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v4.1.3, Copyright (c) Zend Technologies
with Zend OPcache v8.1.3, Copyright (c), by Zend Technologies

Installing MariaDB Database Manager for WordPress

WordPress stores the contents of a website in a database. MariaDB is a free, open source database manager, based on MySQL. To install it, we only need to do:

```
$ sudo apt install mariadb-server
```

Code language: Shell Session (shell)

Then, we will start, enable and check the status of MariaDB:

```
$ sudo systemctl start mariadb
$ sudo systemctl enable mariadb
$ systemctl status mariadb
```

Code language: Shell Session (shell)

We should see that MariaDB is active. But we still need to secure its installation with the command:

```
$ sudo mysql_secure_installation
```

Code language: Shell Session (shell)

To set up MariaDB's security parameters, we have to answer several questions (and here, root doesn't refer to our virtual machine's root user, it refers to MariaDB's root user!):

```
Enter current password for root (enter for none): <Enter>
Switch to unix_socket authentication [Y/n]: Y
Set root password? [Y/n]: Y
New password: 101Asterix!
Re-enter new password: 101Asterix!
Remove anonymous users? [Y/n]: Y
Disallow root login remotely? [Y/n]: Y
Remove test database and access to it? [Y/n]:  Y
Reload privilege tables now? [Y/n]:  Y
```

Code language: Shell Session (shell)

We must then restart the MariaDB service:

```
$ sudo systemctl restart mariadb
```

Code language: Shell Session (shell)

Now that MariaDB is properly installed, we need to set up a new database for our WordPress website.

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

Code language: Shell Session (shell)

We will need to supply the root password for MariaDB (not the VM's root password!). Finally, we can create our WordPress database with the following SQL commands:

```
MariaDB [(none)]> CREATE DATABASE wordpress_db;
MariaDB [(none)]> CREATE USER 'admin'@'localhost' IDENTIFIED
BY 'WPpassw0rd';
MariaDB [(none)]> GRANT ALL ON wordpress_db.* TO
'admin'@'localhost' IDENTIFIED BY 'WPpassw0rd' WITH GRANT
OPTION;
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> EXIT;
```

Code language: SQL (Structured Query Language) (sql)

Now if we go back to MariaDB with the earlier command `mysql -u root -p` and we do:

```
MariaDB [(none)]> show databases;
```

Code language: SQL (Structured Query Language) (sql)

We should see something like this:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| wordpress_db       |
+--------------------+
```

Code language: plaintext (plaintext)

Our `wordpress_db` database is there.

Installing WordPress

Before we can start installing WordPress on our Born2beroot virtual server, we need the following packets: `wget` to download from a web server, and `tar` to decompress a file.

```
$ sudo apt install wget
$ sudo apt install tar
```

Code language: Shell Session (shell)

Then, we will download the archive of the latest version of WordPress from the official website, extract it and place its contents in the `/var/www/html` directory. Then we will clean up the archive and the extraction directory:

```
$ wget http://wordpress.org/latest.tar.gz
$ tar -xzvf latest.tar.gz
$ sudo mv wordpress/* /var/www/html/
$ rm -rf latest.tar.gz wordpress/
```

Code language: Shell Session (shell)

We need a configuration file for WordPress. A sample is included in our files, so let's rename and edit it:

```
$ sudo mv /var/www/html/wp-config-sample.php
/var/www/html/wp-config.php
$ sudo nano /var/www/html/wordpress/wp-config.php
```

Code language: Shell Session (shell)

Here, we want to modify the database parameters to direct WordPress toward the one we created with MariaDB.

```
<?php
/* ... */
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress_db' );

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

/** Database password */
define( 'DB_PASSWORD', 'WPpassw0rd' );

/** Database host */
define( 'DB_HOST', 'localhost' );
```

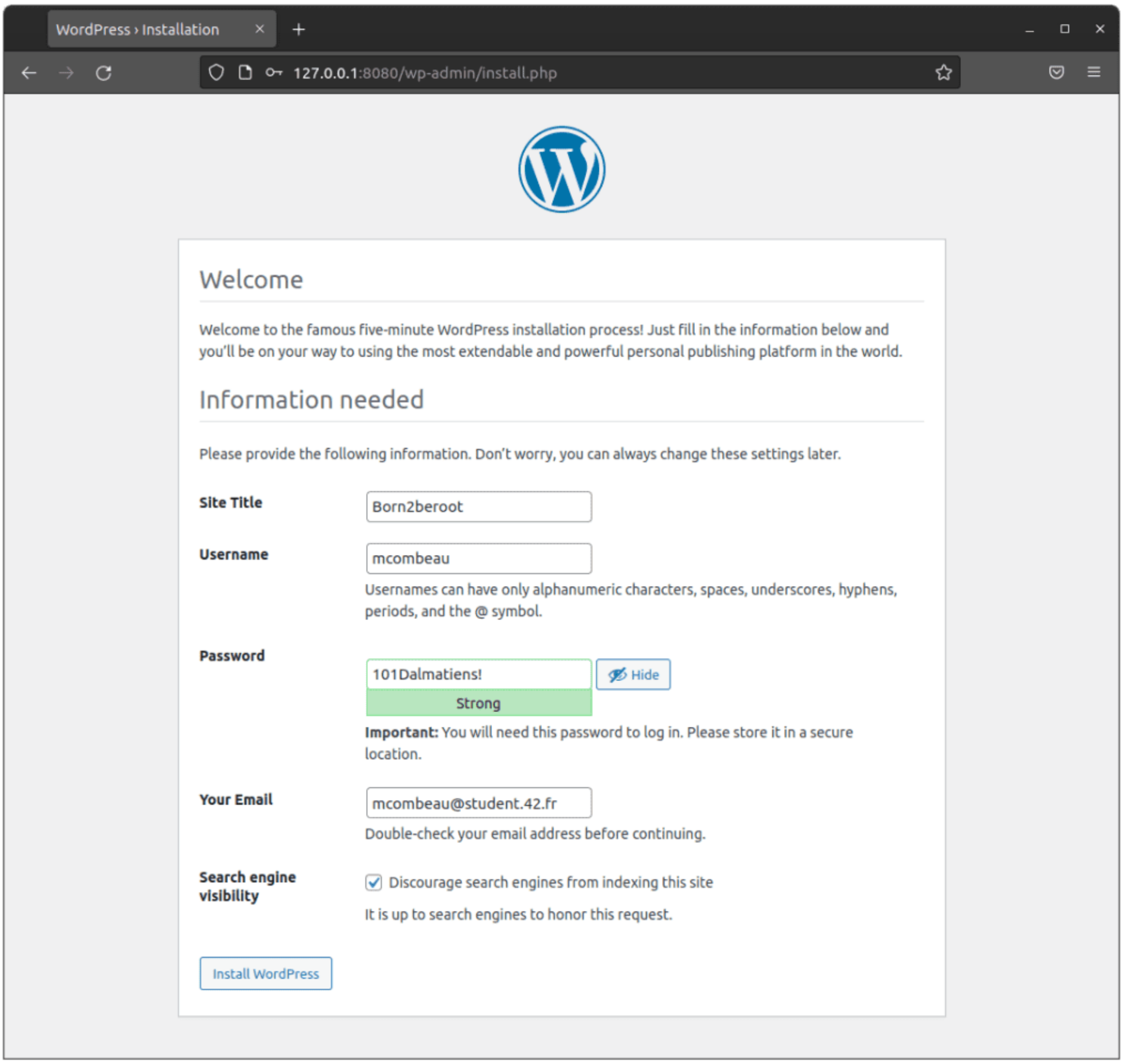
Code language: PHP (php)

Lastly, we need to change the permissions for the WordPress directories for the `www-data` user (our web server) and restart lighttpd:

```
$ sudo chown -R www-data:www-data /var/www/html/
$ sudo chmod -R 755 /var/www/html/
$ sudo systemctl restart lighttpd
```

Code language: Shell Session (shell)

Finally, we can connect to `http://127.0.0.1:8080` in our host browser to reach the WordPress installation menu for our new website.



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 Born2beroot

Username mcombeau

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

Password 101Dalmatiens! Hide

Strong

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

Your Email mcombeau@student.42.fr

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.

Install WordPress

There! Once the installation is complete, we can connect and customize our website however we want. Anything is possible!

Installing Fail2ban

Fail2ban is a program that analyses server logs to identify and ban suspicious IP addresses. If it finds multiple failed login attempts or automated attacks from an IP address, it can block it with the firewall, either temporarily or permanently.

This is the service we will install for the second Born2beroot bonus. We will then start and enable Fail2ban, as well as check its status.

```shell
$ sudo apt install fail2ban
$ sudo systemctl start fail2ban
$ sudo systemctl enable fail2ban
$ sudo systemctl status fail2ban
```

Code language: Shell Session (shell)

We will then need to create and modify the `/etc/fail2ban/jail.local` file to configure its parameters.

```shell
$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
$ sudo nano /etc/fail2ban/jail.local
```

Code language: Shell Session (shell)

To apply Fail2ban to SSH connections, we have to add a few lines to the file under the “SSH servers” section that starts at line 279:

```
#

# SSH servers

#



[sshd]



# To use more aggressive sshd modes set filter parameter "mode" in jail.local:

# normal (default), ddos, extra or aggressive (combines all).

# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.

# mode   = normal

enabled  = true

maxretry = 3

findtime = 10m
```

```
bantime   = 1d

port      = 4242

logpath   = %(sshd_log)s

backend   = %(sshd_backend)s
```

Code language: Bash (bash)

In order to see the failed connection attempts and banned IP addresses, all we need to do is use the following commands:

```
$ sudo fail2ban-client status
$ sudo fail2ban-client status sshd
$ sudo tail -f /var/log/fail2ban.log
```

Code language: Shell Session (shell)

To test that Fail2ban is actually banning IP addresses, we can change the SSH ban time to a lower value, like 30m, in the `/etc/fail2ban/jail.local` configuration file. Then try connecting multiple times from the host machine via SSH with the wrong password. After a few attempts, it should refuse the connection and the `fail2ban-client status sshd` command should show the banned IP address.

And that's it for the Born2beroot bonuses!

Please leave a comment if you have some tips about the WordPress installation or to let us know which bonus service you chose to install!

Sources and Further Reading

- WordPress, *Before You Install* [WordPress.org]
- Lighttpd [official website]
- MariaDB [official website]
  - *mysql_secure_installation* [MariaDB]
  - *MariaDB Server Documentation* [MariaDB]
- Richard, *Creating New MySQL User and Database for WordPress* [Website for Students]

- PHP [official website]
- Ondřej Surý, *deb.sury.org: Frequently Asked Questions* [GitHub]
- WordPress, *How to Install WordPress* [WordPress.org]
- SuperHosting, *What is CGI, FastCGI?* [SuperHosting.bg]
- Abhishek Prakash, *Secure Your Linux Server With Fail2Ban [Beginner’s Guide]* [Linux Handbook]