*"ZPR PWr - Integrated Development ProgramWrocław University of Technology"*

**Wrocław University of Science and Technology**

# Advanced Web Technologies

Lab

Subject: Preparation of LAMP environment, WordPress CMS installation

Prepared by: mgr Piotr Jozwiak

Date: July 2020

Number of hours: 2 hours

## Table of Contents

## Entry

WordPress is one of the most popular CMS (Content management system) on the Internet. This system allows you to easily and quickly create a blog or website. WordPress uses the popular PHP programming language and MySQL for data storage. The advantage of this system is the fact that once installed most of the steps administrative and update tasks can be performed using the i also the ease of use of the mentioned tool for people who are not familiar with programming web services. WordPress users also have great support from the WordPress community, which provides a range of mostly free extensions, templates, and features.

In this lab, we will focus on preparing the server instance on which we will run the WordPress website. It will be the basis for practicing basic administrative operations as well as in further laboratories as a development environment for implementing plugins.

Due to the high popularity of this software on the IT market, the abbreviation LAMP was developed to describe the full technological stack necessary to run WP. This abbreviation comes from the first letters of the components of this stack: Linux, Apache, MySQL and PHP. Therefore, our first steps will be directed to presenting the process of preparing the server environment.

## Prerequisites

In our lab, we will use the Ubuntu 18.04 LTS operating system, on which we will install the LAMP stack. We will not describe here how to install the operating system itself. I assume that as a last resort you can practice the following laboratory on a system loaded from USB, the so-called Live CD or use hosting. These instructions should work on any Debian-based system.

The choice was intentionally directed towards Linux systems, because it is a natural environment for running websites based on PHP+MySQL. Of course, nothing stands in the way of running the mentioned stack on MS Windows (then the stack takes the name WAMP), but due to the fact that such a configuration is not common in production environments, we will only discuss the LAMP stack. In fact, WAMP stacks are only used for development environments, which is not the best solution anyway, because LAMP and WAMP environments have some subtle differences in operation - which can translate into strange errors appearing only on one of these environments.

Since we have already selected the operating system, it should be mentioned at this point that in order to be able to install the necessary software, you will need a user with the appropriate permissions.

Ideally, the user will have sudo privileges. As a last resort, you can use the root user directly.

The issue of firewall configuration will not be discussed in this paper. So I'm assuming it's already well configured to share port 80, or it just doesn't have it installed at all.

## Installing the LAMP server

Let's move on to the description of the installation process. We log into the server. The following examples assume that the commands will be executed by the root user. If you do not have access to this user, remember to precede the commands with sudo.

### Step 1 - Apache web server installation

The Apache web server is one of the most popular web servers in the world. It is well documented and widely used for much of internet history, making it a great default choice for website hosting.

We start the installation by refreshing the apt manager package list:

```
λ aptupdate
```

Then we go to install the Apache server with the command:

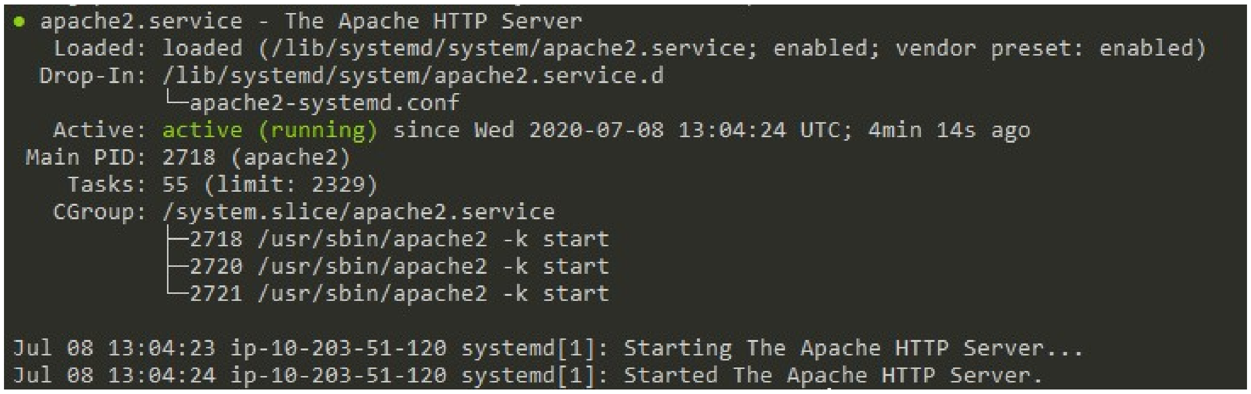
```
λ apt installapache2
```

When asked if we want to continue with the installation, answer Yes - by entering the letter Y
and pressing Enter.

After a while, the web server is installed and ready to use. We verify the operation of the server with the command:

```
λ systemctl statusapache2
```

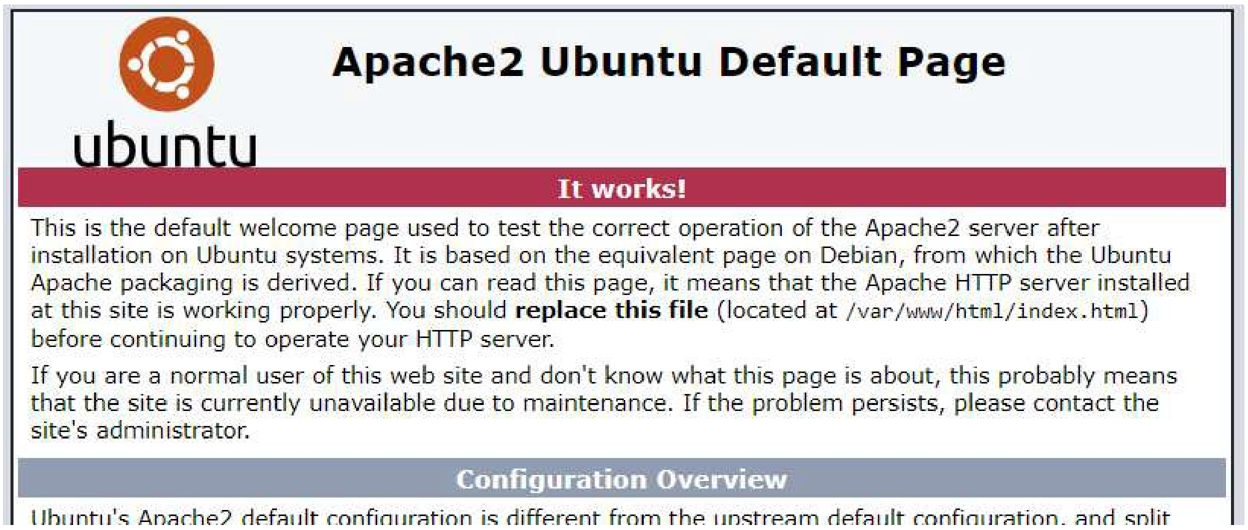
As a result of this command, we get information about the state of the Apache2 process. The correct state is: Active
as in the picture below:



```
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Wed 2020-07-08 13:04:24 UTC; 4min 14s ago
 Main PID: 2718 (apache2)
    Tasks: 55 (limit: 2329)
   CGroup: /system.slice/apache2.service
           ├─2718 /usr/sbin/apache2 -k start
           ├─2720 /usr/sbin/apache2 -k start
           └─2721 /usr/sbin/apache2 -k start

Jul 08 13:04:23 ip-10-203-51-120 systemd[1]: Starting The Apache HTTP Server...
Jul 08 13:04:24 ip-10-203-51-120 systemd[1]: Started The Apache HTTP Server.
```

So, we open a web browser and enter the IP address of our serverWWW.As a result of this operation, the welcome page of the Apache2 server should appear:



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split

If we see such a page, it means that we have installed the Apache server correctly and that it is available on the network. It's time for the next step.

## Step 2 - MySQL database installation

Now that we have the web server up and running, it's time to install MySQL. MySQL is a database management system. Basically, it will organize and provide access to databases where the site stores page content and settings.

Again, we'll use apt to get and install the software:

```
λ apt installmysql-server
```

This command, like the previous one, will first show a list of packages that are part of the MySQL server along with information about the amount of disk space required. Press Y to continue the installation.

Once the installation is complete, we run a simple security script that comes pre-installed with MySQL. It will remove some unsafe defaults and block access to the database system. Run the interactive script with the command:

```
λmysql_secure_installation
```

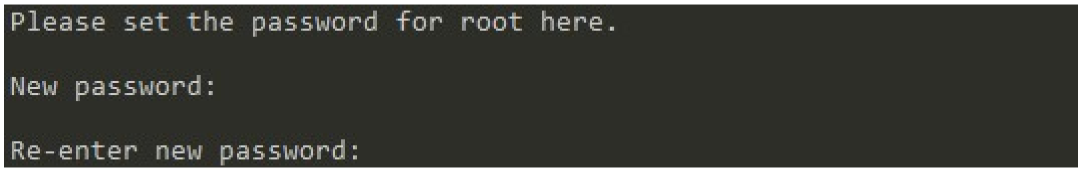
Firstthe question is about enabling the VALIDATE PASSWORD plugin. This plugin forces users to use strong passwords. For our lab purposes, it's not necessary to enable it, but for production environments it's worth doing. So in our case we answer N:

```
Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No: N
```

Nextthe question is about setting the password for the root user in the database:

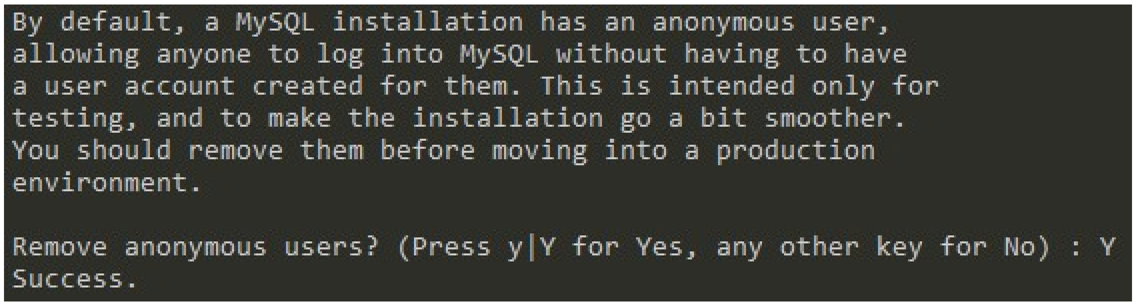


```
Please set the password for root here.

New password:

Re-enter new password:
```

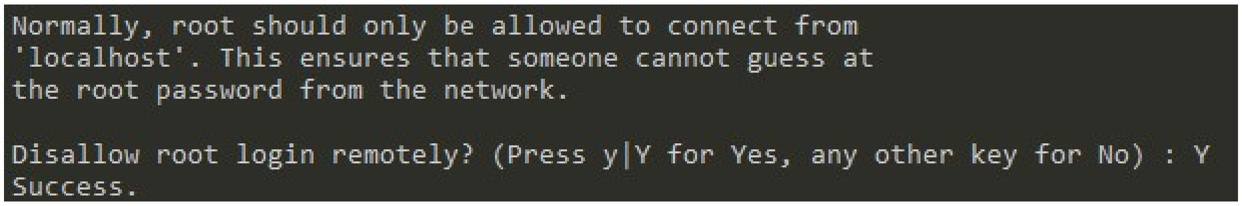
Then we are asked whether to remove anonymous userto the database. Of course, this account should be deleted giving the answer Y:



```
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : Y
Success.
```

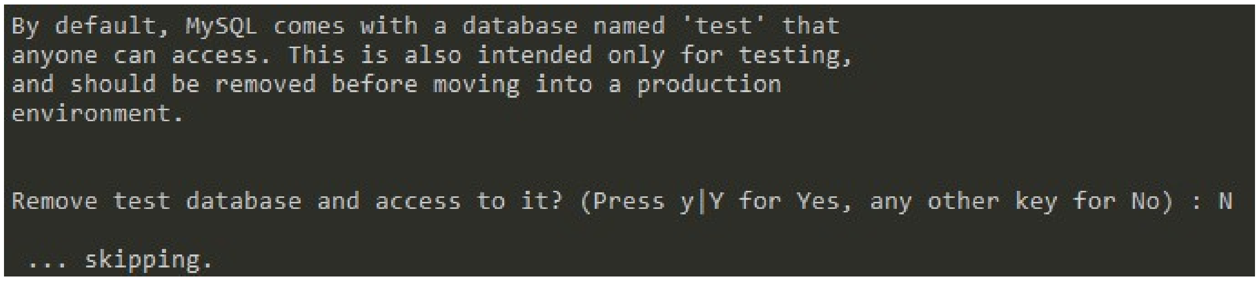
Going further, we are asked whether to disable access for the user**root**outside of the local machine. Here, too, it is a good practice to disable the possibility of logging in from the outside of this user. Therefore, we answer Y:



```
Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : Y
Success.
```

Another question concerns the removal of the 'test' sample database schema. Nothing prevents you from leaving this scheme:

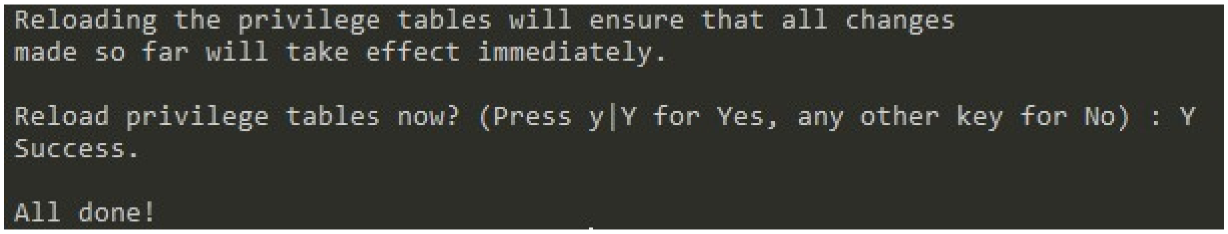


```
By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No) : N

 ... skipping.
```

The last question is about reloading the changes we made. Therefore, we answer Y:



```
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : Y
Success.

All done!
```

Thus, our database should already be pre-configured and working. One more question remains. MySQL from version 5.7 for the root user by default uses authentication via the auth_socket plugin instead of a password. This allows for greater security and usability in many cases, but can also complicate the situation when we need to allow an external program (e.g. PhpMyAdmin) to access this user. Therefore, if we want to change the way of logging into the database for this user, we must follow the instructions below.

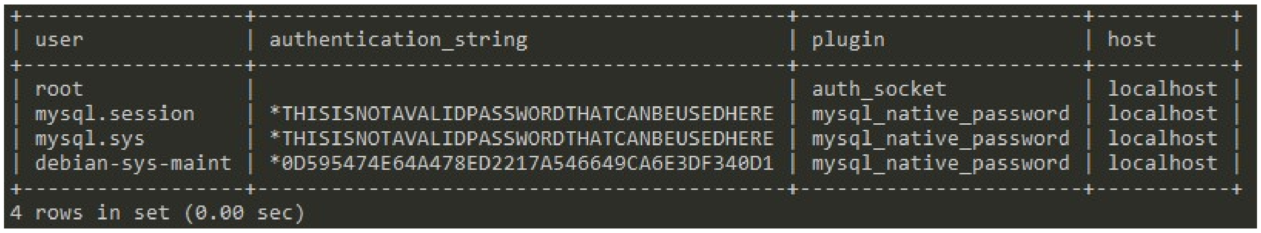
First, we log into the database:

```
λ mysql
```

Notice that the prompt has changed immediately. Then we check the authentication method for users by entering the following query:

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

In response, we receive information in the form of a table,where for the root user it is indicated auth_socket plagin:



```
+------------------+-------------------------------------------+-----------------------+-----------+
| 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 | *0D595474E64A478ED2217A546649CA6E3DF340D1 | mysql_native_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
4 rows in set (0.00 sec)
```

To configure the root account topassword authentication, run the following ALTER USER command. Remember to change your password to a strong password of your choice:

```
mysql> ALTER USER'root'@'localhost'IDENTIFIED WITH mysql_native_password BY'bye
ssword';
```

Then run FLUSH PRIVILEGES,which tells the server to reload the tables powers and introducing new changes into force:

```
mysql> FLUSH PRIVILEGES;
```

Let's verify the authentication method againfor the root user using the SQL command already discussed above. The plugin column should show mysql_native_password for the root user. We can log out of the database with the command:

```
mysql>exit
```

At this point, the database system is set up and we can proceed to install PHP, the last component of the LAMP stack.

## Step 3 - PHP installation

**PHP**is an application that processes code to display dynamic page contentWWW.It can run scripts, connect to MySQL databases for information, and pass the processed content to a web server for display.

We will use the apt installer again to install PHP. In addition, we will include some helper packages this time so that the PHP code can run under the Apache server and talk to the MySQL database:

```
apt install php libapache2-mod-phpphp-mysql
```

The above command should install the minimum set necessary to run the PHP interpreter in Apache2 and enable connection to the MySQL database.

In most cases, you'll want to modify the way Apache handles files when a directory is requested. Currently, if a user requests a directory from the server, Apache first looks for a file named index.html. We want to tell the server to prefer PHP files over others. Therefore, we will modify the server configuration so that it first looks for the index.php file.

INfor this we need to edit the file`/etc/apache2/mods-enabled/dir.conf` . By default

it looks like this:

```
<IfModulemod_dir.c>

    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm

</IfModule>
```

The DirectoryIndex option specifiesfile search order. You can see that index.php is far down the list. So let's move it to the top of this list. Let's save the changes to the file. For the changes to be accepted, we must restart the Apache server. The following command is used for this:

```
systemctl restartapache2
```

To verify that the system is properly configured to support PHP, we will create a very simple PHP script called info.php.

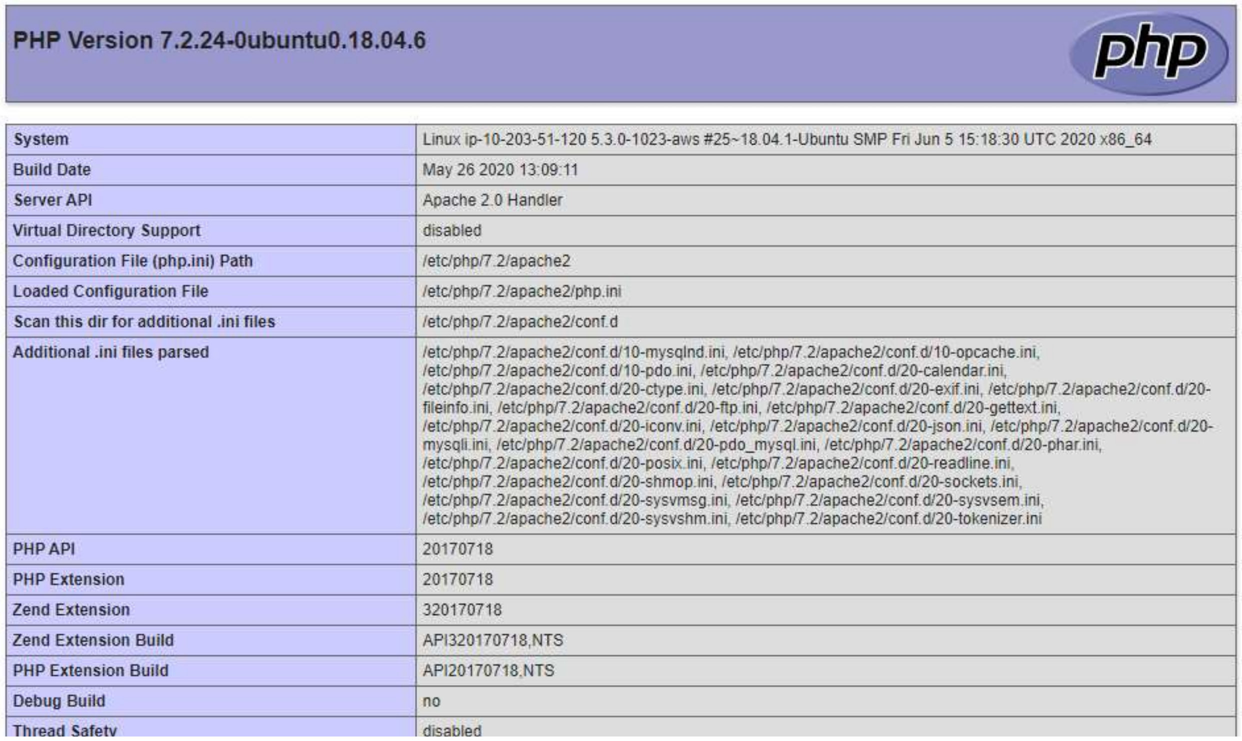
INFor this purpose, in the /var/www/html folder, let's create a file with the command:

```
touchinfo.php
```

Let's edit the file by entering the following content:

```
<?php
    phpinfo();
?>
```

Let's save the changes to the file. We will now test if the PHP interpreter is working by launching the browser and entering the address:http://ADDRESS_IP/info.php. A web page should appear describing the current configuration of the PHP interpreter, similar to the one in the image below:



PHP Version 7.2.24-0ubuntu0.18.04.6

| | |
|---|---|
| System | Linux ip-10-203-51-120 5.3.0-1023-aws #25~18.04.1-Ubuntu SMP Fri Jun 5 15:18:30 UTC 2020 x86_64 |
| Build Date | May 26 2020 13:09:11 |
| Server API | Apache 2.0 Handler |
| Virtual Directory Support | disabled |
| Configuration File (php.ini) Path | /etc/php/7.2/apache2 |
| Loaded Configuration File | /etc/php/7.2/apache2/php.ini |
| Scan this dir for additional .ini files | /etc/php/7.2/apache2/conf.d |
| Additional .ini files parsed | /etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini |
| PHP API | 20170718 |
| PHP Extension | 20170718 |
| Zend Extension | 320170718 |
| Zend Extension Build | API320170718,NTS |
| PHP Extension Build | API20170718,NTS |
| Debug Build | no |
| Thread Safety | disabled |

The page contains basic information about our server from the PHP perspective. This is useful for debugging and for checking PHP settings.

Finally, it's worth deleting this file after the test, as it may actually pass information about our server to unauthorized users.

The LAMP stack is now ready. We can move on to the next issue, which is installing WordPress.

## WordPress installation

The very installation of CMS Wordpress is very simple, because it has an easy-to-use installer. However, before we can run it, we need to prepare the environment.

### Step 1 - Prepare the database for WordPress

The first step we will take is to prepare the database. WordPress uses MySQL to manage and store site and user information. MySQL is already installed, but we need to create a database and user to use WordPress.

To perform this step, let's log into MySQL by issuing the following command:

```
λ mysql -u root-p
```

After this command, we will be asked to enter the password for the root user.

First, we need to create a separate database that will be controlled by WordPress. We can name it whatever we want. In our example, we'll be using the wordpressDB name to keep things simple. Let's create a database for WordPress by typing:

```
mysql> CREATE DATABASE wordpressDB DEFAULT CHARACTER SET utf8 COLLATE utf8_unicod
e_ci;
```

Next, we'll create a separate MySQL user account that we'll use exclusively for our new database. Creating single-function databases and accounts is a good idea from a management and security point of view. In this guide, we will use the name wpuser. Of course, we can choose a different name for this user.

In order, we will create a new account, set a password and grant access to the database we have created. We can do this by typing the following command. Remember to choose a strong password for the database user:

```
mysql> GRANT ALL ON wordpressDB.* TO'wpuser'@'localhost'IDENTIFIED BY'password
';
```

The database and user account is now ready. We still need to refresh the permissions to make it current the MySQL instance knew about the recent changes we made:

```
mysql> FLUSH PRIVILEGES;
```

We can now exit the database console with the command:

```
mysql> EXIT;
```

## Step 2 - Install additional PHP extensions

When setting up our LAMP stack, we only required a very minimal set of extensions for PHP to communicate with MySQL. WordPress and many of its plugins use additional PHP extensions.

We can download and install some of the most popular PHP extensions for use with WordPress by typing:

```
λ apt install php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-
intl php-zip
```

In order for the installed extensions to be used, restart the Apache server with the following command:

```
λ systemctl restart apache2
```

## Step 3 - Adjust Apache configuration

Next, we'll make a few minor changes to our Apache configuration. We'll use

**/var/www/wordpress**as the root of our WordPress installation.

### Create a folder for WordPress

To begin with, we will create the folder mentioned above where we will place the wordpress source files. To do this, follow the commands below:

```
λ mkdir /var/www/wordpress
λ chown -Rwww-date:www-data /var/www/wordpress
λ chmod -R 755 /var/www/wordpress
```

Notice that the folder has been set to the owner named www-data. This is the default user under which the Apache server runs. In order for it to be able to read this folder without any problems, it is necessary to take care of the access permissions to this directory.

Next, we change the Apache configuration file to point to our new site in the newly created folder. Let's edit the file: /etc/apache2/sites-available/000-default.conf. Let's find the line that starts with the word DocumentRoot and change it to the following value:

```
DocumentRoot/var/www/wordpress
```

Let's save the changes to the file. Let's not close the edition, in the next steps we will change other settings.

## Enabling support for .htaccess files

Currently, the use of .htaccess files is disabled. WordPress and many WordPress plugins make extensive use of these files to adjust directory settings to server behaviorWWW.

We will make further changes to the Apache configuration file opened for editing in the previous section. To allow .htaccess files, we need to set the AllowOverride directive in the directory block pointing to our root directory. Let's add the following block of text inside the VirtualHost block in the configuration file, making sure to use the correct root directory:

```
<directories/var/www/wordpress/
    >AllowOverride All
</directories>
```

Let's save the changes and close the file.

## Enable mod_rewrite module

Then we can enable the module**mod_rewrite**used by WordPress to generate permalinks. To do this, let's execute the command:

```
a2enmodrewrite
```

## Testing and running configuration changes

Before we run the changes, we will check if we have made any mistakes. For this we will use the command:

```
apache2ctlconfigtest
```

If everything is OK, we should get a message: Syntax OK. We can apply the

changes by restarting the Apache server:

```
systemctl restartapache2
```

## Step 4 - Download WordPress Sources

Now that the Apache server is set up, we can download and configure WordPress. In particular, for security reasons, it is always recommended to download the latest version of WordPress from the manufacturer's website.

Let's change to the temporary directory and then download the compressed version by typing:

```
continued/tmp
curl -O https://wordpress.org/latest.tar.gz
```

Let's extract the compressed file to create the WordPress directory structure:

```
tar xzvflatest.tar.gz
```

We'll move these files to our server root in a momentWWW.Before we do that, we'll add a dummy .htaccess file for WordPress to use.

Create a file by typing:

```
touch/tmp/wordpress/.htaccess
```

We'll also copy the sample config file into a place that WordPress actually reads:

```
cp /tmp/wordpress/wp-config-sample.php/tmp/wordpress/wp-config.php
```

We can also create an update directory so that WordPress doesn't run into permission issues when trying to do this ourselves after a software update:

```
mkdir/tmp/wordpress/wp-content/upgrade
```

Now we can copy the entire contents of the directory to our server rootWWW. We use a dot at the end of our source directory to indicate that everything in the directory should be copied, including hidden files (such as the .htaccess file we created):

```
cp -a /tmp/wordpress/./var/www/wordpress
```

## Step 5 - WordPress Folder Permissions

Before doing WordPress configuration from the browser, we need to customize some items in our WordPress directory.

### Repair WordPress file and folder permissions

One of the more important things we need to do is set reasonable file permissions.

We'll start by setting the files to the www-data user and group. This is the user the Apache server is running under and Apache will need to read and write WordPress files to serve the site and perform automatic updates.

Let's update the property with chown:

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

Next, we will run two find commands to set the correct permissions on WordPress directories and files:

```
find /var/www/wordpress/ -type d -exec chmod 750 {}\;
find /var/www/wordpress/ -type f -exec chmod 640 {}\;
```

At the moment, such permissions should be sufficient to start work.

## Configuration of WordPress keys and connections to MySQL

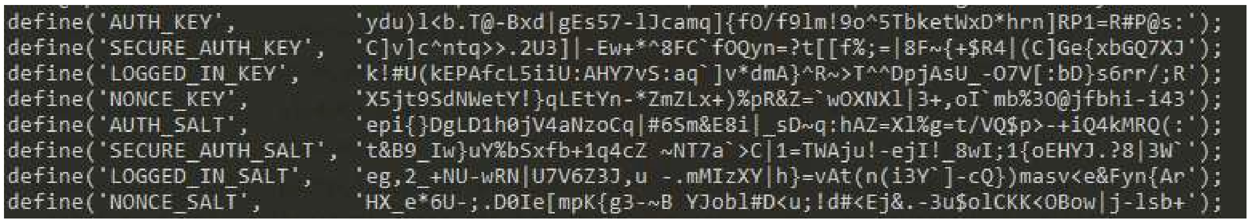
Now we need to make changes to the main WordPress configuration file.

After opening the file, our first step will be to customize some secret keys to ensure the security of our installation. WordPress provides a secure generator of these values so we don't have to try to come up with good values ourselves.

To retrieve the secure values from the WordPress key generator, let's type:

```
curl -shttps://api.wordpress.org/secret-key/1.1/salt/
```

As a result of this command, we will get values that will resemble those in the image below:



```
define('AUTH_KEY',         'ydu)l<b.T@-Bxd|gEs57-lJcamq]{fO/f9lm!9o^5TbketWxD*hrn]RP1=R#P@s:');
define('SECURE_AUTH_KEY',  'C]v]c^ntq>>.2U3]|-Ew+*^8FC`fOQyn=?t[[f%;=|8F~{+$R4|(C]Ge{xbGQ7XJ');
define('LOGGED_IN_KEY',    'k!#U(kEPAfcL5iiU:AHY7vS:aq`]v*dmA}^R~>T^^DpjAsU_-O7V[:bD}s6rr/;R');
define('NONCE_KEY',        'X5jt9SdNWetY!}qLEtYn-*ZmZLx+)%pR&Z=`wOXNXl|3+,oI`mb%3O@jfbhi-i43');
define('AUTH_SALT',        'epi{}DgLD1h0jV4aNzoCq|#6Sm&E8i|_sD~q:hAZ=Xl%g=t/VQ$p>-+iQ4kMRQ(:');
define('SECURE_AUTH_SALT', 't&B9_Iw}uY%bSxfb+1q4cZ ~NT7a`>C|1=TWAju!-ejI!_8wI;1{oEHYJ.?8|3W`');
define('LOGGED_IN_SALT',   'eg,2_+NU-wRN|U7V6Z3J,u -.mMIzXY|h}=vAt(n(i3Y`]-cQ})masv<e&Fyn{Ar');
define('NONCE_SALT',       'HX_e*6U-;.D0Ie[mpK{g3-~B YJobl#D<u;!d#<Ej&.-3u$olCKK<OBow|j-lsb+');
```

These are configuration lines that we can paste directly into our configuration file to set secure keys. Let's copy the resulting output to the clipboard. Next, let's open the WordPress configuration file /var/www/wordpress/wp-config.php for editing.

Let's find a place in the file with key definitions and replace them with those from the clipboard.

Next, we need to modify the database connection settings. These settings are at the beginning of the file. We need to enter the database name, database user, and associated password that we set up in the MySQL section.

Another change we need to make is to set the method WordPress should use to write to the file system. Since we allowed the web server to write there, where necessary, we can explicitly set the filesystem method to direct. Failure to do so with our current settings will cause WordPress to attempt to make changes via FTP. The line configuring this WordPress property needs to be added anywhere in the file. This parameter is not in the default configuration file.

```
define('DB_NAME','wordpressDB');

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

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

. . .
```

Let's save and close the file.

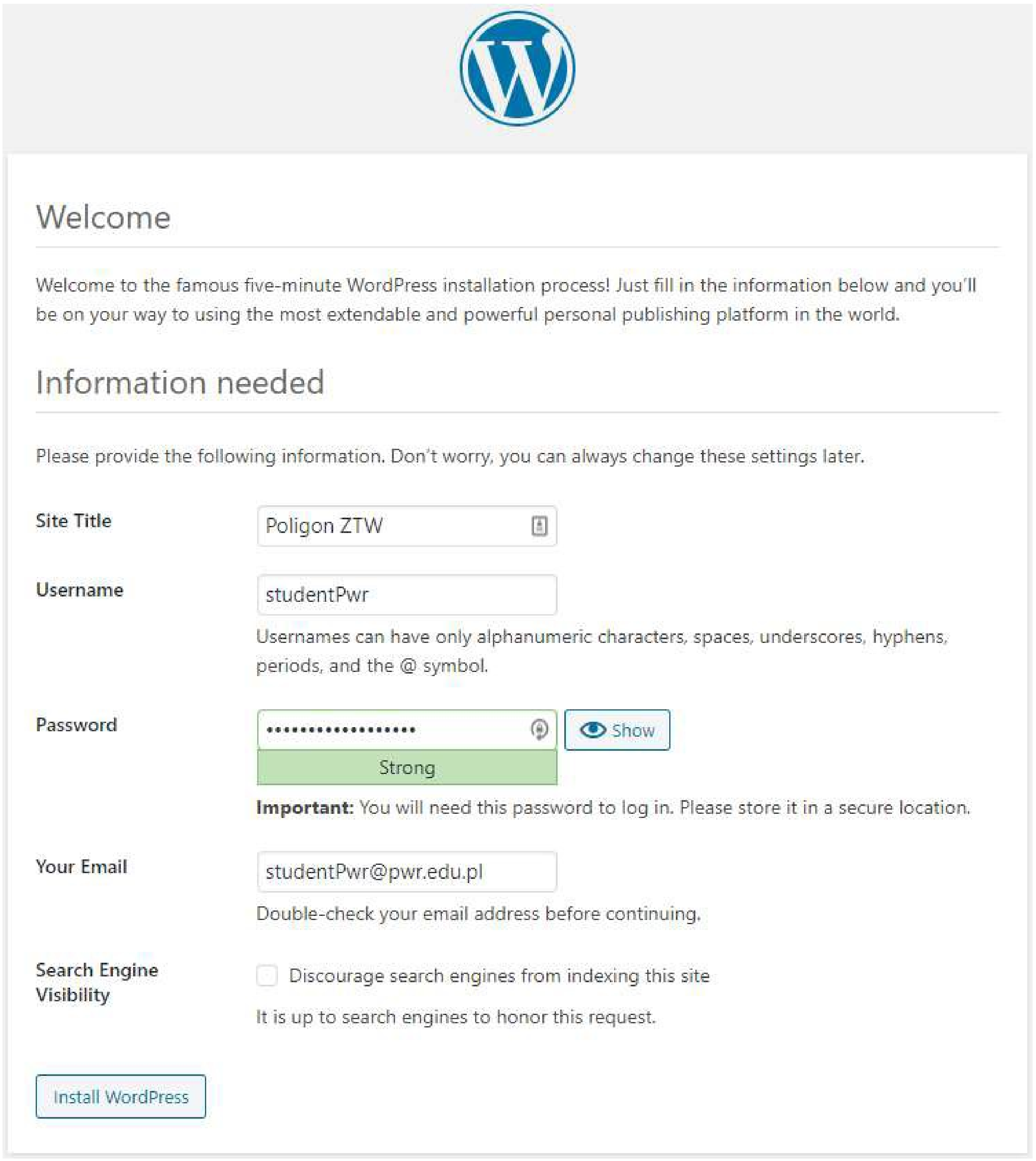
## Step 6 - Completing the installation via a web browser

Now that the server configuration is complete, we can complete the installation through the web interface.

In a web browser, let's navigate to the domain name or IP address of the

server:http://ADDRESS_IP/

Let's choose the installation language. Then we will be redirected to the main WordPress configurator page. Let's enter the basic data into the form:



Welcome

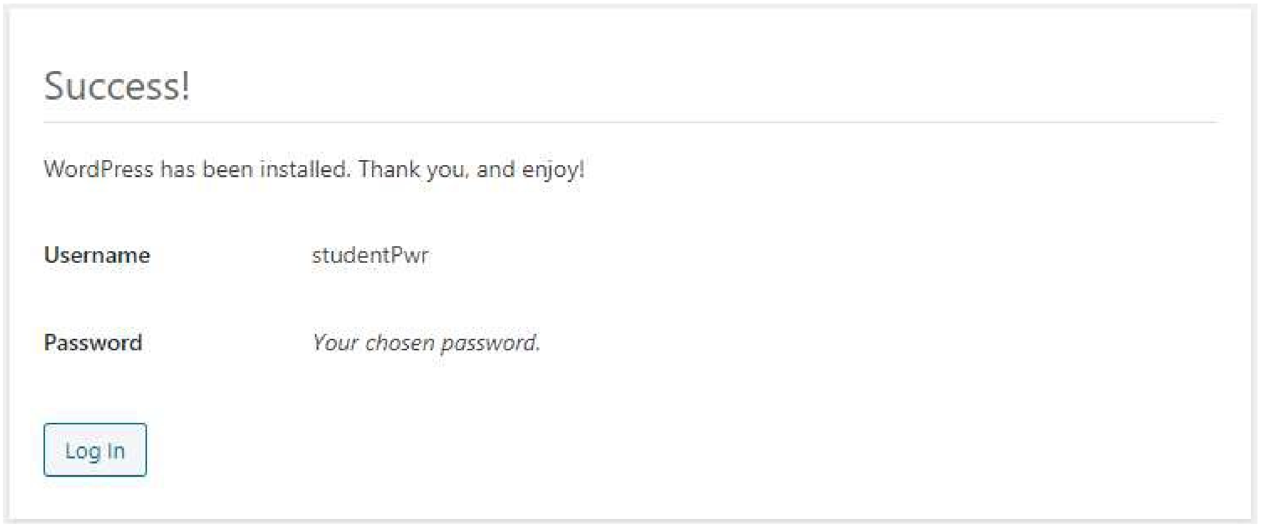
Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

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

| | |
|---|---|
| Site Title | Poligon ZTW |
| Username | studentPwr<br>Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol. |
| Password | •••••••••••••••••• Show<br>Strong<br>**Important:** You will need this password to log in. Please store it in a secure location. |
| Your Email | studentPwr@pwr.edu.pl<br>Double-check your email address before continuing. |
| Search Engine Visibility | ☐ Discourage search engines from indexing this site<br>It is up to search engines to honor this request. |

Then confirm with the Install WordPress button. At this point, WordPress installs the database schemas and does the rest of the file configuration. If everything was successful, we should see a screen confirming this fact in the image below:

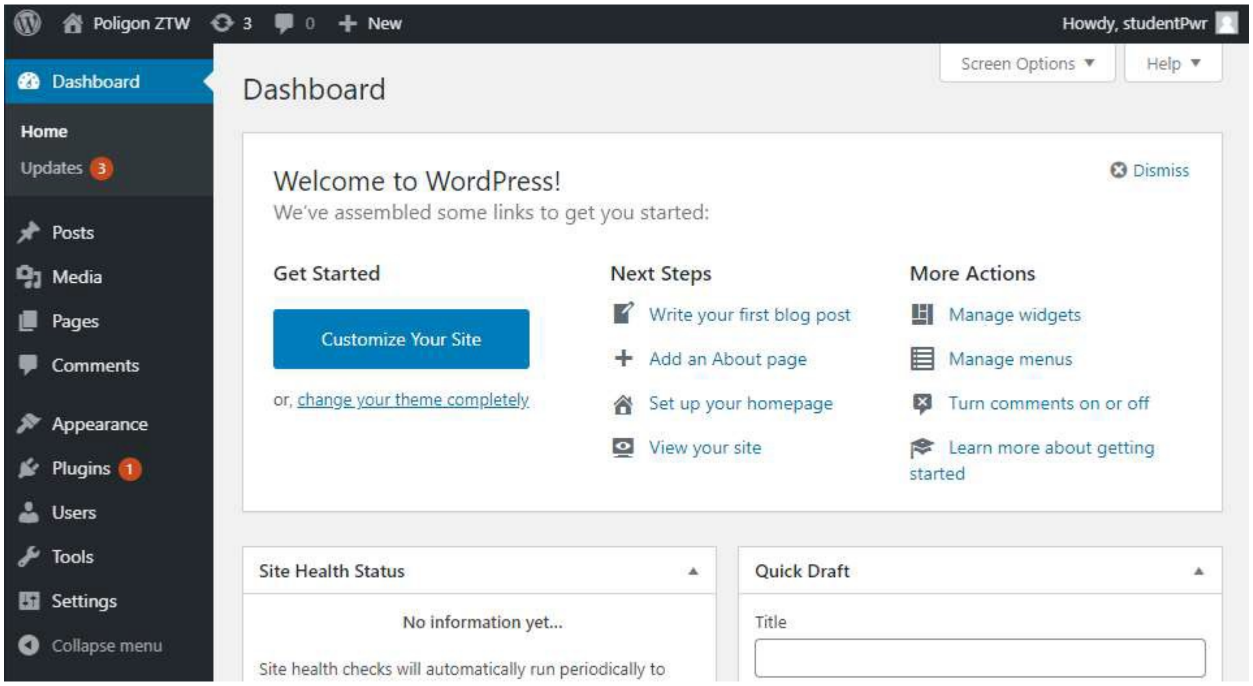


Success!

WordPress has been installed. Thank you, and enjoy!

| | |
|---|---|
| Username | studentPwr |
| Password | *Your chosen password.* |

After pressing the Log In buttonwe will be transferred to the admin panel login form. Let's log in to the panel. This panel resembles the one in the image below:



Dashboard

Welcome to WordPress!

We've assembled some links to get you started:

Get Started

or, change your theme completely

Next Steps

- Write your first blog post
- Add an About page
- Set up your homepage
- View your site

More Actions

- Manage widgets
- Manage menus
- Turn comments on or off
- Learn more about getting started

No information yet...

Site health checks will automatically run periodically to

Thus we have come to the end of the commands from this laboratory. We have a fully functioning website built on the WordPress engine.