INSTALLATIONS

You can work on the real machine or on a virtual machine.

To create a virtual machine you can work with the Virtualbox software: https://www.virtualbox.org/wiki/Downloads

You can work with the Windows or Linux Operating System

WEB SERVERS IN LINUX

Linux desktops and servers are becoming more popular these days, especially for web development. You can download the Apache, MySQL, and PHP server source code packages and compile them on your Linux system, but unless you need the absolute latest version of things, that's not the recommended way to do it.

These days, most Linux distributions include packages for easily installing all the components you need for a complete web development environment. For Debian-based Linux distributions (such as Ubuntu and Linux Mint), you use the apt-get command-line tool to install software packages.

For Debian-based systems, such as Ubuntu, follow these steps to do that:

Step 1 — Installing Apache and Updating the Firewall

The Apache web server is among the most popular web servers in the world. It's well documented, has an active community of users, and has been in wide use for much of the history of the web, which makes it a great choice for hosting a website.
Start by updating the package manager cache. If this is the first time you're using `sudo` within this session, you'll be prompted to provide your user's password to confirm you have the right privileges to manage system packages with `apt`.

```
sudo apt update
```

Then, install Apache with:

```
sudo apt install apache2
```

You'll also be prompted to confirm Apache's installation by pressing `Y`, then `ENTER`.

Once the installation is finished, you'll need to adjust your firewall settings to allow HTTP traffic. UFW has different application profiles that you can leverage for accomplishing that.

To list all currently available UFW application profiles, you can run:

```
sudo ufw app list
```

You'll see output like this:

```
Output
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

Here's what each of these profiles mean:

•**Apache**: This profile opens only port `80` (normal, unencrypted web traffic).
•**Apache Full**: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic).
•**Apache Secure**: This profile opens only port `443` (TLS/SSL encrypted traffic).
For now, it's best to allow only connections on port `80`, since this is a fresh Apache installation and you still don't have a TLS/SSL certificate configured to allow for HTTPS traffic on your server.

To only allow traffic on port `80`, use the `Apache` profile:

```
sudo ufw allow in "Apache"
```

You can verify the change with:

```
sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
```

```
OpenSSH                    ALLOW       Anywhere
Apache                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Apache (v6)                ALLOW       Anywhere (v6)
```

Traffic on port `80` is now allowed through the firewall.

You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser (see the note under the next heading to find out what your public IP address is if you do not have this information already): `http://your_server_ip`

You'll see the default Apache web page, which is there for informational and testing purposes.

Step 2 — Installing MySQL

Now that you have a web server up and running, you need to install the database system to be able to store and manage data for your site. MySQL is a popular database management system used within PHP environments.

Again, use `apt` to acquire and install this software:

```
sudo apt install mysql-server
```

When prompted, confirm installation by typing `Y`, and then `ENTER`.

When the installation is finished, it's recommended that you run a security script that comes pre-installed with MySQL. This script will remove some insecure default settings and lock down access to your database system. Start the interactive script by running:

```
sudo mysql_secure_installation
```

This will ask if you want to configure the `VALIDATE PASSWORD PLUGIN`.

Note: Enabling this feature is something of a judgment call. If enabled, passwords which don't match the specified criteria will be rejected by MySQL with an error. It is safe to leave validation disabled, but you should always use strong, unique passwords for database credentials.

Answer `Y` for yes, or anything else to continue without enabling.

```
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:
```

If you answer "yes", you'll be asked to select a level of password validation. Keep in mind that if you enter `2` for the strongest level, you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words.

```
There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

Regardless of whether you chose to set up the `VALIDATE PASSWORD PLUGIN`, your server will next ask you to select and confirm a password for the MySQL **root** user. This is not to be confused with the **system root**. The **database root** user is an administrative user with full privileges over the database system. Even though the default authentication method for the MySQL root user dispenses the use of a password, **even when one is set**, you should define a strong password here as an additional safety measure. We'll talk about this in a moment.

If you enabled password validation, you'll be shown the password strength for the root password you just entered and your server will ask if you want to continue with that password. If you are happy with your current password, enter `Y` for "yes" at the prompt:

```
Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other
key for No) : y
```

For the rest of the questions, press `Y` and hit the `ENTER` key at each prompt. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes you have made.

When you're finished, test if you're able to log in to the MySQL console by typing:

```
sudo mysql
```

This will connect to the MySQL server as the administrative database user **root**, which is inferred by the use of `sudo` when running this command. You should see output like this:

```
mysql>
```

To exit the MySQL console, type:

```
exit
```

Notice that you didn't need to provide a password to connect as the **root** user, even though you have defined one when running the `mysql_secure_installation` script. That is because the default authentication method for the administrative MySQL user is `unix_socket` instead of `password`. Even though this might look like a security concern at first, it makes the database server more secure because the only users allowed to log in as the **root** MySQL user are the system users with sudo privileges connecting from the console or through an application running with the same privileges. In practical terms, that means you won't be able to use the administrative database **root** user to connect from your PHP application. Setting a password for the **root** MySQL account works as a safeguard, in case the default authentication method is changed from `unix_socket` to `password`.

For increased security, it's best to have dedicated user accounts with less expansive privileges set up for every database, especially if you plan on having multiple databases hosted on your server.

Note: At the time of this writing, the native MySQL PHP library `mysqlnd` doesn't support `caching_sha2_authentication`, the default authentication method for MySQL 8. For that reason, when creating database users for PHP applications on MySQL 8, you'll need to make sure they're configured to use `mysql_native_password` instead. We'll demonstrate how to do that in Step 6.

Your MySQL server is now installed and secured. Next, we'll install PHP, the final component in the LAMP stack.

Step 3 — Installing PHP

You have Apache installed to serve your content and MySQL installed to store and manage your data. PHP is the component of our setup that will process code to display dynamic content to the final user. In addition to the `php` package, you'll need `php-mysql`, a PHP module that allows PHP to communicate with MySQL-based databases. You'll also need `libapache2-mod-php` to enable Apache to handle PHP files. Core PHP packages will automatically be installed as dependencies.

To install these packages, run:

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

Once the installation is finished, you can run the following command to confirm your PHP version:

```
php -v
```

Step 4: Install phpMyAdmin

Now that Apache and PHP are installed the final step is to install phpMyAdmin and configure. To do that, run the commands below

```
sudo apt install phpmyadmin
```

When prompted to choose the webserver, select apache2 and continue.

When prompted again to allow dbconfig-common to install a database and configure select No.

Now, open your web browser and login to the server hostname or IP address followed by phpmyadmin

ex.: http://localhost/phpmyadmin

WEB SERVERS IN WINDOWS AND MAC

Installing and running the Apache, MySQL, and PHP servers in a Windows or Mac environment is very tricky, because there are lots of factors involved in how to install and configure them. For starters, both Windows and macOS come with a web server built in, so if you install the Apache web server you'll need to configure it to use an alternative TCP port.

Likewise, macOS includes an older version of PHP by default, so if you install an updated version of PHP, things get tricky trying to make sure which version is active. Because of these complexities, it's not recommended for beginners to install the Apache, MySQL, and PHP packages separately in the Windows and Mac environments. There's a much simpler way of getting that to work, which I'll describe in the next section.

Using premade servers

Trying to get a working Apache, MySQL, and PHP server in Windows (called WAMP) or in the Mac environment (called MAMP) can be a complicated process. There's a lot of work downloading each of the individual server packages, configuring them, and getting things to work together.

Fortunately, some resourceful programmers have done that work for us! There are quite a few open-source packages that bundle the Apache web server, MySQL (or MariaDB) server, and PHP server together to install as a single package. This is by far the best way to go if you plan on using your Windows or Mac workstation or laptop as your web development environment

There are quite a few pre-loaded packages available, but these are the most common ones:
»»**XAMPP:** An all-in-one package that supports PHP and Perl server-side programming and also includes an email and FTP server, along with a self-signed certificate to use the Apache web server in HTTPS mode. It has installation packages available for Windows, Mac, and Linux.
»»**Wampserver:** A Windows-based all-in-one package that allows you to install multiple versions of the Apache, MySQL, and PHP servers at the same time. You can then mix-and-match which versions of which server you have active at any time, allowing you to duplicate almost any web-hosting environment.
»»**MAMP:** A Mac-based all-in-one package that is easy to install and use. It also has a commercial package called MAMP Pro that provides additional features for managing your web environment for professional developers.
Of these, the XAMPP package is by far the most popular. It was created by the Apache Friends organization to help promote the use of the Apache web server in web development environments. Follow these steps to install XAMPP in a Windows or macOS environment:

Introduction

XAMPP is a free and open-source tool used by web developers in the Windows family and other platforms to set up the development and testing environment. XAMPP server comes with the XAMPP control panel to manage all its components easily.

XAMPP stands for (X) Cross-platform, (A) Apache, (M) MySQL, (P) PHP, (P) Perl and with some additional modules including phpmyadmin (for the database), FileZilla, Mercury, and Tomcat.

Once you have installed and configured the XAMPP server in your system, you can easily work with any CMS like WordPress, Joomla, drupal and more. XAMPP server works like a local server in your system that is generally used by PHP developers to test the websites (web-projects).

This tutorial will show you the various steps on how to install and configure the XAMPP server on Windows 10.

Installation Process of the XAMPP Server

Step 1

To download the XAMPP server, visit the "Apache Friends" (https://www.apachefriends.org/index.html) website in your web browser.

Step 2

Click on "XAMPP for Windows". Then, navigate the downloading location and the file will be automatically downloaded.

Step 3

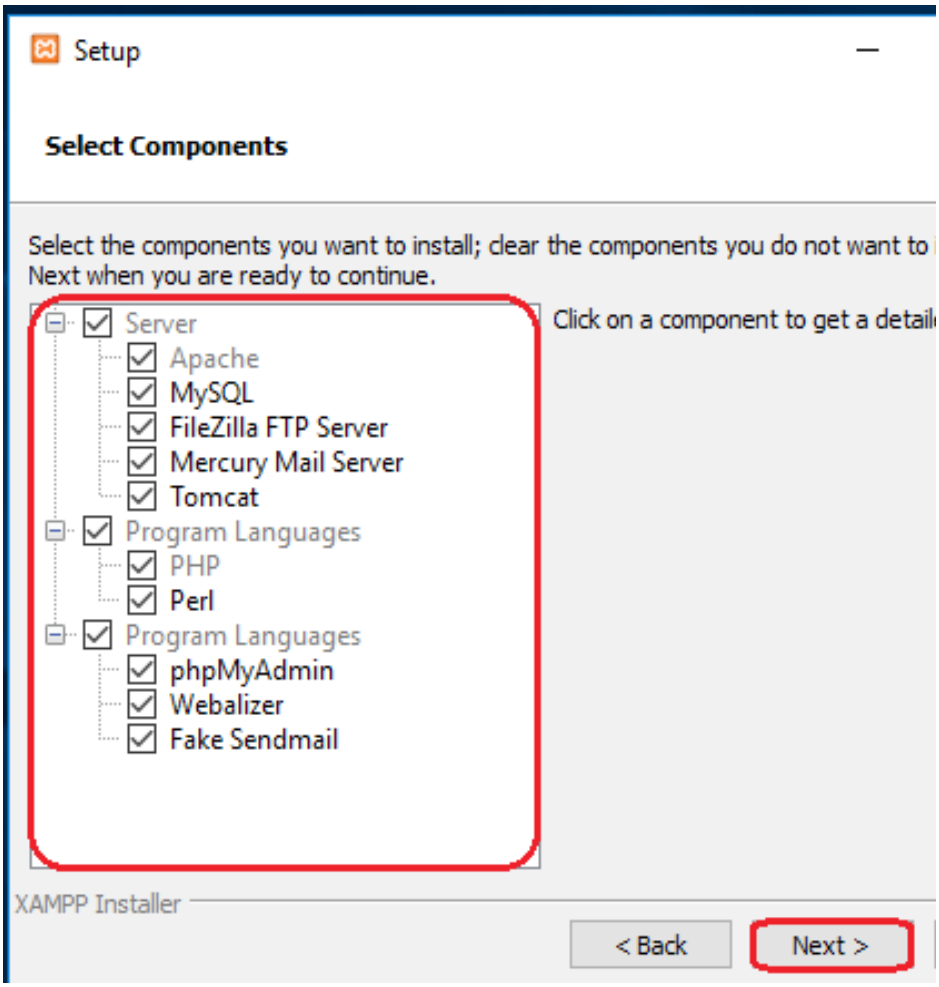
Double-click the downloaded file to launch the XAMPP installer.

Step 4

Select the components that you want to install and click on the "Next" button.

Note

By default, all components are selected in your XAMPP installation.



Step 5

Choose a folder to install the XAMPP and click on the "Next" button.

Step 6

“Next button” and to finish click on the "Finish" button.

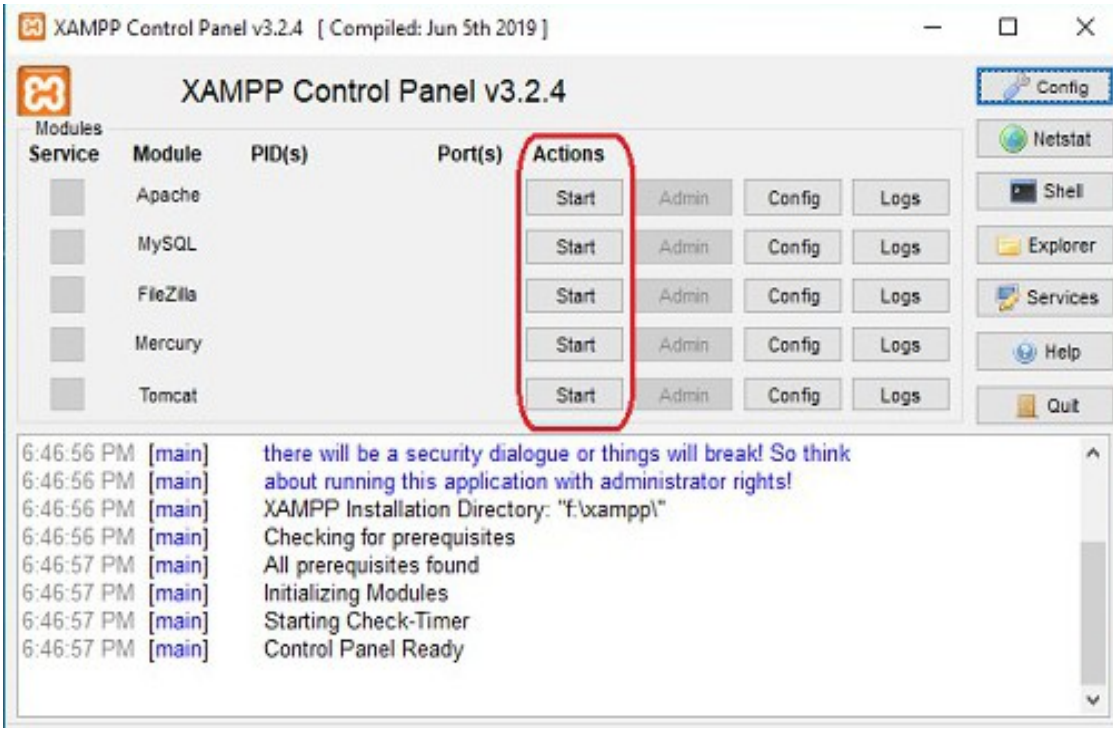
Start XAMPP Server

Step 1

Start the XAMPP control panel through the "Run as administrator" option.

Step 2

"XAMPP Control Panel" will appear on the screen and click on "Start" action to start the "Apache" and "MySQL" modules.



Other possibilities: work with docker

https://www.docker.com/

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Dockerfile allows us to generate our dockers automatically.

Our dockerfile content (this is not the last version xampp-linux-x64-7.4.*-installer.run):

```
FROM ubuntu:latest

RUN apt-get update
COPY ./xampp-linux-x64-7.4.*-installer.run /root/xampp-linux-x64-7.4.*-installer.run
RUN chmod a+x /root/xampp-linux-x64-7.4.*-installer.run && /root/xampp-linux-x64-7.4.*-installer.run && rm /root/xampp-linux-x64-7.4.*-installer.run
RUN apt-get -y install nano && apt purge --auto-remove && apt clean && rm -rf /var/lib/apt/lists/*
```

```
COPY httpd-xampp.conf /opt/lampp/etc/extra/httpd-xampp.conf
RUN mkdir /opt/lampp/htdocs/dwc/ &&  chmod -R 777 /opt/lampp/htdocs/dwc
VOLUME /opt/lampp/var/mysql

EXPOSE 80 443 3306
CMD /opt/lampp/lampp start && tail -F /opt/lampp/logs/error_log
```

Docker installation
1. Copy dockerdwc/ folder to Escriptori
2. Go to the folder
3. Build the docker
docker build -t dwc/xampp .
4. Grant permissions to dwc folder: chmod -R 777 /rutaabsolutafins/dockerdwc/dwc
chmod -R 777 /home/alumne/Escritorio/dockerdwc/dwc
5. Run the docker
docker run -it -d -v /rutaabsolutafins/dockerdwc/dwc:/opt/lampp/htdocs/dwc -p 8080:80 -p 443:443 -p 3306:3306 dwc/xampp
docker run -it -d -v /home/alumne/Escritorio/dockerdwc/dwc:/opt/lampp/htdocs/dwc -p 8080:80 -p 443:443 -p 3306:3306 dwc/xampp
6. Check it works
localhost:8080
7. Create your project in ./dwc folder and write **localhost:8080/dwc in browser**

VISUAL STUDIO CODE

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux.

Visit these links to install visual studio code:

https://www.digitalocean.com/community/tutorials/how-to-set-up-visual-studio-code-for-php-projects

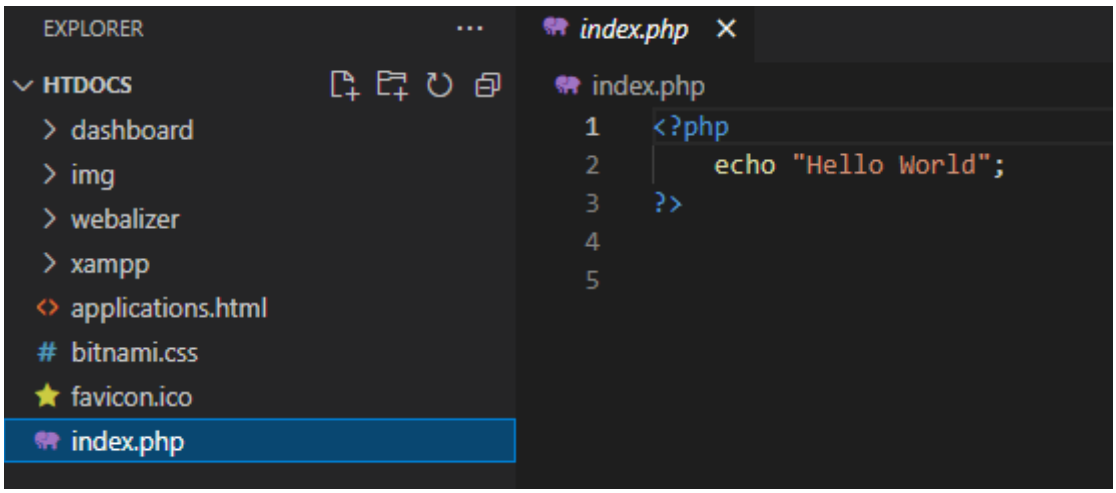
START WITH VISUAL STUDIO CODE (VSCode)

If you are on Windows:

The default folder for Xampp is:

C:\xampp\htdocs

Open this Folder from VSCode:

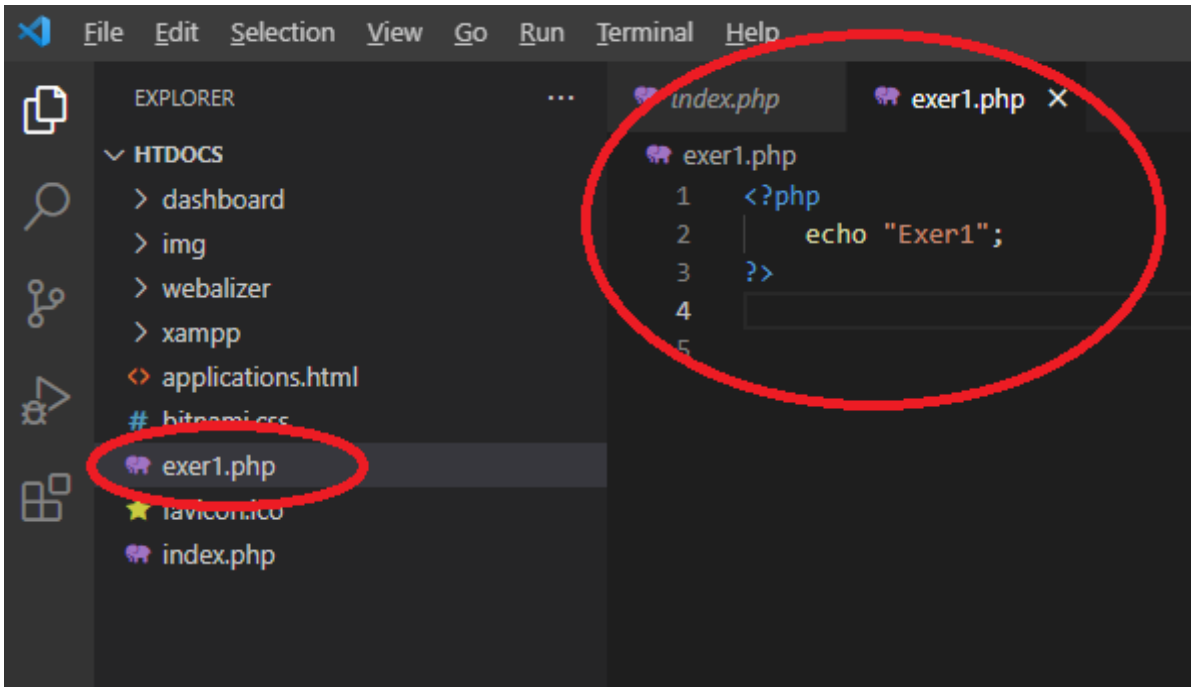


Type in index.php “Hello World”, Save the file and Open in Browser:

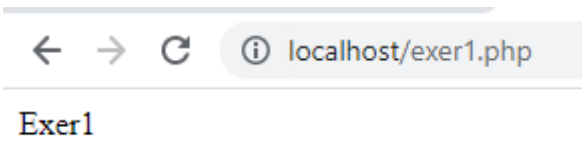
http://localhost

You will see the execution of index.

You can create more Php Files in this folder: 'File → New File'



And running with: http://localhost/exer1.php



If you are on Ubuntu:

The default folder for Apache is: `/var/www/html`

Everything else the same as in Windows