

WEB STACK IMPLEMENTATION (LEMP STACK)

What is LEMP?

LEMP is a group of software that serves dynamic development and deployment of web applications and websites. It is an acronym that stands for Linux, Engine-X (Ngnix), MySQL or MariaDB and PHP ie Hypertext Preprocessor Perl or Pytho).

Linux operating system: It is the operating system just like Windows, iOS, and Mac OS. An operating system is software that manages all of the hardware resources associated with your desktop or laptop. It manages the communication between software and hardware. Without the operating system (OS), the software wouldn't function.

Nginx (Engine-X) Web Server: NginX, is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. It is a high performance web server that is used to display web pages to site visitors. It has a high level of responsiveness and resource efficiency i.e its lightweight utilization of resources and its flexibility to scale simply even with minimal equipment.

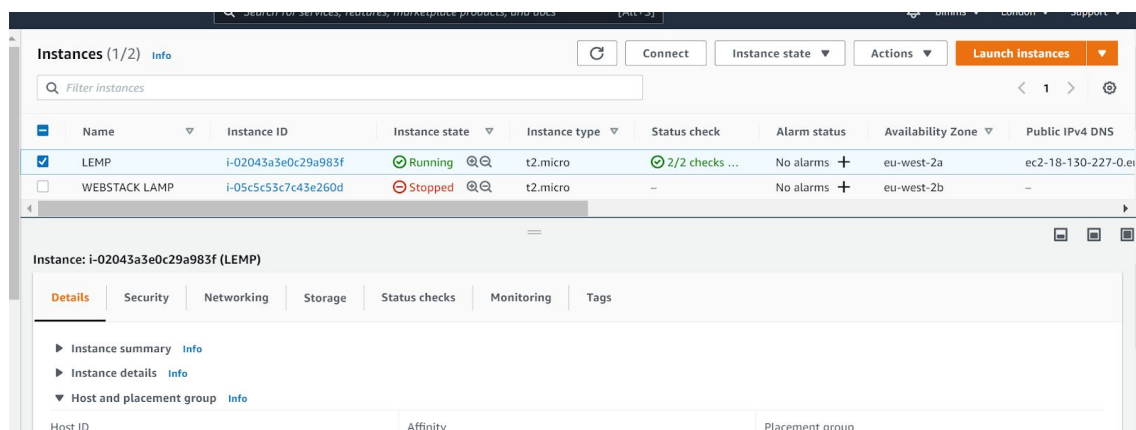
In this project, I will be using the apt package manager to obtain the software.

MySQL (or MariaDB i.e. a drop-in fork of MySQL) database server. MySQL is a popular database management system used within PHP environments. It helps to store and manage data for the website.

PHP - Hypertext Preprocessor: It is a widely-used Open Source general-purpose scripting language that is especially designed for web development and can be embedded into HTML. It processes code and generates dynamic content for the web server. The scripting role can also be filled with Python or Perl. Servers such as Gunicorn or UWSGI can be used in conjunction with Nginx to serve these applications. The hypertext preprocessor means dynamic data processing and programming. While Apache embeds the PHP interpreter in each request, Nginx requires an external program to handle PHP processing and act as a bridge between the PHP interpreter itself and the web server. This allows for a better overall performance in most PHP-based websites, but it requires additional configuration.

STEPS IN IMPLEMENTING LEMP STACK:

STEP 1: Create and configure a new EC2 Instance of t2.nano family with Ubuntu Server 20.04 LTS (HVM) image.



STEP 2: Installing the Nginx Web Server

In order to display web pages to our site visitors, we are going to employ Nginx, a high-performance web server. We'll use the apt package manager to install this package. Steps include:

```
$ sudo apt update
```

```
$ sudo apt install nginx
```

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo apt install nginx  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjpeg8  
  libjpeg-turbo8 libpng16-0 libnginx-mod-http-image-filter  
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libtiff5  
  libwebp6 libxpm4 nginx-common nginx-core  
Suggested packages:  
  libgd-tools fcgiwrap nginx-doc ssl-cert  
The following NEW packages will be installed:  
  fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjpeg8  
  libjpeg-turbo8 libpng16-0 libnginx-mod-http-image-filter  
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libtiff5  
  libwebp6 libxpm4 nginx-common nginx-core
```

When prompted, enter Y to confirm that you want to install Nginx. Once the installation is finished, the Nginx web server will be active and running on your Ubuntu 20.04 server.

To verify that nginx was successfully installed and is running as a service in Ubuntu, run:

```
$ sudo systemctl status nginx
```

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo systemctl status nginx  
● nginx.service - A high performance web server and a reverse proxy server  
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: ena  
   Active: active (running) since Sun 2021-02-28 20:59:43 UTC; 54s ago  
     Docs: man:nginx(8)  
  Main PID: 2081 (nginx)  
    Tasks: 2 (limit: 1160)  
   Memory: 4.5M  
    CGroup: /system.slice/nginx.service  
            └─2081 nginx: master process /usr/sbin/nginx -g daemon on; master  
              └─2082 nginx: worker process  
  
Feb 28 20:59:43 ip-172-31-27-96 systemd[1]: Starting A high performance web ser  
Feb 28 20:59:43 ip-172-31-27-96 systemd[1]: Started A high performance web serv  
lines 1-13/13 (END)
```

The above output confirms that the server has been launched in clouds.

Before we can receive any traffic by the Web Server, we need to open [TCP port 80](#) which is the default port that web browsers use to access web pages on the Internet.

The TCP port 22 is opened by default on EC2 machine and in order to access it via SSH, we need to add a rule to EC2 configuration to open inbound connection through port 80:

The screenshot shows the AWS Management Console interface. At the top, there's a header for 'Instances (1/2)' with filters and actions. Below this is a table of instances:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input checked="" type="checkbox"/>	LEMP	i-02043a3e0c29a983f	Stopped	t2.micro	2/2 checks ...	No alarms	eu-west-2a
<input type="checkbox"/>	WEBSOCKET LAMP	i-05c5c53c7c43e260d	Stopped	t2.micro	-	No alarms	eu-west-2b

Below the instances table, the 'Inbound rules' for the security group 'sg-056ec1ac3e0a0447f (Bims DevOps)' are shown:

Port range	Protocol	Source	Security groups
80	TCP	0.0.0.0/0	Bims DevOps
80	TCP	:::0	Bims DevOps

The server is running and we can access it locally and from the Internet (Source 0.0.0.0/0 means 'from any IP address').

Check the server is accessible via Ubuntu Shell:

`$ curl http://localhost:80` Or `$ curl http://127.0.0.1:80`

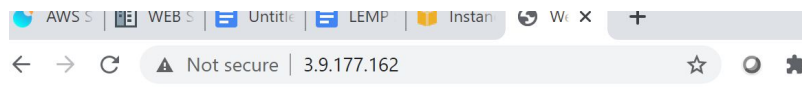
```
ubuntu@ip-172-31-27-96:~$ curl http://localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```

These 2 commands above actually do pretty much the same - they use 'curl' command to request our Apache HTTP Server on port 80 (actually you can even try to not specify any port - it will work anyway). The difference is that: in the first case we try to access our server via DNS name and in the second one - by IP address (in this case IP address 127.0.0.1 corresponds to DNS name 'localhost' and the process of converting a DNS name to IP address is called "resolution").

As an output you can see some strangely formatted test, do not worry, we just made sure that our Apache web service responds to 'curl' command with some payload.

Test Nginx: Test if Nginx server can respond to requests from the Internet. Open a web browser and access following url

<http://<Public-IP-Address>:80>



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

The output above shows that Nginx has been installed and it's accessible through the firewall.

Note: Public IP address can also be retrieved with the `curl` command:

`curl -s http://169.254.169.254/latest/meta-data/public-ipv4`

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ curl -s http://169.254.169.254/latest/meta-data/public-ipv4  
3.9.177.162
```

STEP 3: Installing MySQL

As defined above, The database. MySQL is a popular open source relational database management system used within the PHP environment for storing application data. The Database Management System (DBMS) is used to store and manage data for your site in a relational database.

Acquire and install the Mysql software:

`$ sudo apt install mysql-server`

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo apt install mysql-server  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.1-7  
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl  
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl  
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic  
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0  
  mysql-common mysql-server-8.0 mysql-server-core-8.0  
Suggested packages:
```

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

When the installation is finished, it's recommended to 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.`

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo mysql_secure_installation  
  
Securing the MySQL server deployment.  
  
Connecting to MySQL using a blank password.  
  
VALIDATE PASSWORD COMPONENT 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 component?  
  
Press y|Y for Yes, any other key for No: N  
Please set the password for root here.  
  
New password:  
  
Re-enter new password:  
By default, a MySQL installation has an anonymous user,
```

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.

If you answer “yes” to the validation question, 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.

Test MySQL:

Log in to the MySQL console by typing:

\$ sudo mysql

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo mysql  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 10  
Server version: 8.0.23-0ubuntu0.20.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2021, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql>  
mysql> exit  
Bye  
ubuntu@ip-172-31-27-96:~$
```

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:

To exit the MySQL console, type: **mysql> 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: 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.

MySQL server is now installed and secured. Next, is to install PHP, the final component in the LEMP stack.

STEP 4: Installing PHP

Nginx has been installed to serve the content and MySQL installed to store and manage the data. Next is to install PHP to process code and generate dynamic content for the web server.

While Apache embeds the PHP interpreter in each request, Nginx requires an external program to handle PHP processing and act as a bridge between the PHP interpreter itself and the web server. This allows for a better overall performance in most PHP-based websites, but it requires additional configuration. We need to install php-fpm, which stands for “PHP fastCGI process manager”, and tell Nginx to pass PHP requests to this software for processing. Additionally, you’ll need php-mysql, a PHP module that allows PHP to communicate with MySQL-based databases. Core PHP packages will automatically be installed as dependencies.

To install these 2 packages at once, run:

```
$ sudo apt install php-fpm php-mysql
```

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo apt install php-fpm php-mysql  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  php-common php7.4-cli php7.4-common php7.4-fpm php7.4-json php7.4-mysql  
  php7.4-opcache php7.4-readline  
Suggested packages:  
  php-pear  
The following NEW packages will be installed:  
  php-common php-fpm php-mysql php7.4-cli php7.4-common php7.4-fpm php7.4-json  
  php7.4-mysql php7.4-opcache php7.4-readline
```

When prompted, type **Y** and press **ENTER** to confirm installation.

PHP components have now been installed. Next, you will configure Nginx to use them.

STEP 4: Configuring Nginx to Use PHP Processor

When using the Nginx web server, we can create server blocks (similar to virtual hosts in Apache) to encapsulate configuration details and host more than one domain on a single server. ‘**projectLEMP**’ is used as an example domain name.

On Ubuntu 20.04, Nginx has one server block enabled by default and is configured to serve documents out of a directory at **/var/www/html**. While this works well for a single site, it can become difficult to manage if you are hosting multiple sites. Instead of modifying **/var/www/html**, we’ll create a directory structure within **/var/www** for the **your_domain** website, leaving **/var/www/html** in place as the default directory to be served if a client request does not match any other sites.

Create the root web directory for **your_domain (projectLEMP)** as follows:

```
$ sudo mkdir /var/www/projectLEMP
```

Next, assign ownership of the directory with the \$USER environment variable, which will reference your current system user:

```
$ sudo chown -R $USER:$USER /var/www/projectLEMP
```

Then, open a new configuration file in Nginx's **sites-available** directory using any command-line editor.

```
$ sudo nano /etc/nginx/sites-available/projectLEMP
```

This will create a new blank file. Paste in the following bare-bones configuration:

```
#/etc/nginx/sites-available/projectLEMP
```

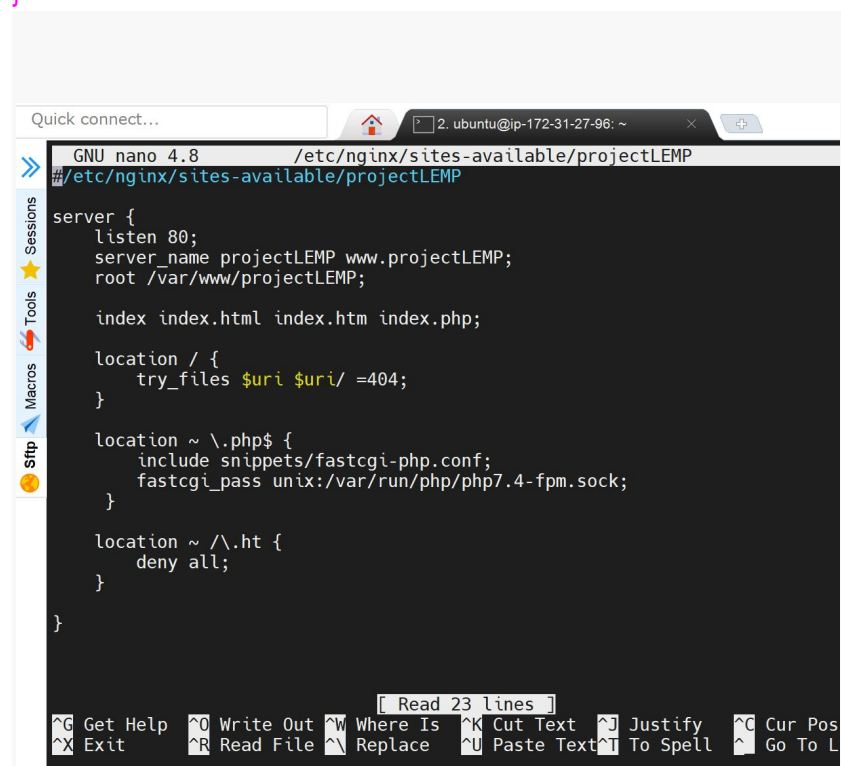
```
server {
    listen 80;
    server_name projectLEMP www.projectLEMP;
    root /var/www/projectLEMP;

    index index.html index.htm index.php;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```



```
GNU nano 4.8 /etc/nginx/sites-available/projectLEMP
#/etc/nginx/sites-available/projectLEMP

server {
    listen 80;
    server_name projectLEMP www.projectLEMP;
    root /var/www/projectLEMP;

    index index.html index.htm index.php;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

Read 23 lines

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^N Replace ^U Paste Text ^T To Spell ^_ Go To L

Here's what each of these directives and location blocks do:

- **listen** — Defines what port Nginx will listen on. In this case, it will listen on port **80**, the default port for HTTP.
- **root** — Defines the document root where the files served by this website are stored.
- **index** — Defines in which order Nginx will prioritize index files for this website. It is a common practice to list **index.html** files with a higher precedence than **index.php** files to allow for quickly setting up a maintenance landing page in PHP applications. You can adjust these settings to better suit your application needs.
- **server_name** — Defines which domain names and/or IP addresses this server block should respond for. **Point this directive to your server's domain name or public IP address.**
- **location /** — The first location block includes a **try_files** directive, which checks for the existence of files or directories matching a URI request. If Nginx cannot find the appropriate resource, it will return a 404 error.
- **location ~ \.php\$** — This location block handles the actual PHP processing by pointing Nginx to the **fastcgi-php.conf** configuration file and the **php7.4-fpm.sock** file, which declares what socket is associated with **php-fpm**.
- **location ~ /\.ht** — The last location block deals with **.htaccess** files, which Nginx does not process. By adding the **deny all** directive, if any **.htaccess** files happen to find their way into the document root, they will not be served to visitors.

Save and close the file.

Activate the above configuration by linking to the config file from Nginx's **sites-enabled** directory:

```
$ sudo ln -s /etc/nginx/sites-available/projectLEMP /etc/nginx/sites-enabled/
```

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo mkdir /var/www/projectLEMP  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ ls -la /var/www/  
total 16  
drwxr-xr-x  4 root root 4096 Feb 28 21:15 .  
drwxr-xr-x 14 root root 4096 Feb 28 20:59 ..  
drwxr-xr-x  2 root root 4096 Feb 28 20:59 html  
drwxr-xr-x  2 root root 4096 Feb 28 21:15 projectLEMP  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ ls -la /var/www/html  
total 12  
drwxr-xr-x  2 root root 4096 Feb 28 20:59 .  
drwxr-xr-x  4 root root 4096 Feb 28 21:15 ..  
-rw-r--r--  1 root root 612 Feb 28 20:59 index.nginx-debian.html  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo chown -R $USER:$USER /var/www/projectLEMP  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo nano /etc/nginx/sites-available/projectLEMP  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo nano /etc/nginx/sites-available/projectLEMP  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo ln -s /etc/nginx/sites-available/projectLEMP /etc/  
nginx/sites-enabled/  
ubuntu@ip-172-31-27-96:~$
```

This will tell Nginx to use the configuration next time it is reloaded. You can test your configuration for syntax errors by typing:

```
$ sudo nginx -t
```

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo nginx -t  
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful  
ubuntu@ip-172-31-27-96:~$
```

We also need to disable default Nginx host that is currently configured to listen on port 80, for this run:

```
sudo unlink /etc/nginx/sites-enabled/default
```

Then reload Nginx to apply the changes:

```
$ sudo systemctl reload nginx
```

Your new website is now active, but the web root /var/www/projectLEMP is still empty. An index.html file will be created in that location in order to test if the new server block works as expected:

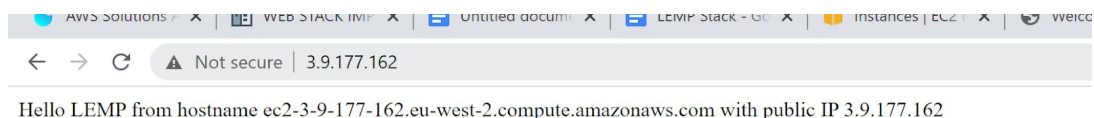
```
sudo echo 'Hello LEMP from hostname' $(curl -s  
http://169.254.169.254/latest/meta-data/public-hostname) 'with public IP' $(curl -s  
http://169.254.169.254/latest/meta-data/public-ipv4) > /var/www/projectLEMP/index.html
```

```
ubuntu@ip-172-31-27-96:~$ sudo systemctl reload nginx  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ sudo echo 'Hello LEMP from hostname' $(curl -s http://  
169.254.169.254/latest/meta-data/public-hostname) 'with public IP' $(curl -s htt  
p://169.254.169.254/latest/meta-data/public-ipv4) > /var/www/projectLEMP/index.h  
tml  
ubuntu@ip-172-31-27-96:~$
```

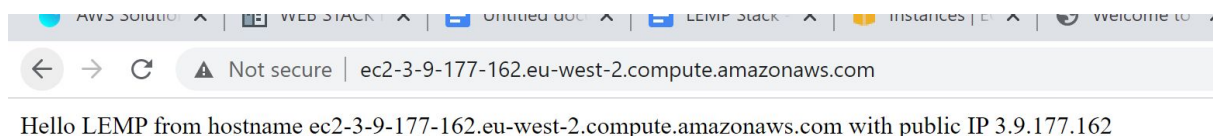
Open the website URL using IP address:

<http://<Public-IP-Address>:80>

<http://http://3.9.177.162:80>



<http://<Public-DNS-Name>:80>



The website can be accessed through either the IP or the public DNS name.

We can leave this file in place as a temporary landing page for your application until you set up an [index.php](#) file to replace it. Once done, the [index.html](#) file needs to be removed or renamed from the document root, as it would take precedence over an [index.php](#) file by default.

LEMP stack is now fully configured.

In the next step, we'll create a PHP script to test that Nginx is in fact able to handle `.php` files within your newly configured website.

STEP 5: Testing PHP with Nginx

Having set the LEMP, the validity of will now be tested so as to ensure that Nginx can correctly hand `.php` files off to the PHP processor.

The steps include the below:

- Create a test PHP file in the document root.
 - Open a new file called `info.php` within the document root within the text editor:

```
$ nano /var/www/projectLEMP/info.php
```

```
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$  
ubuntu@ip-172-31-27-96:~$ nano /var/www/projectLEMP/info.php  
ubuntu@ip-172-31-27-96:~$
```

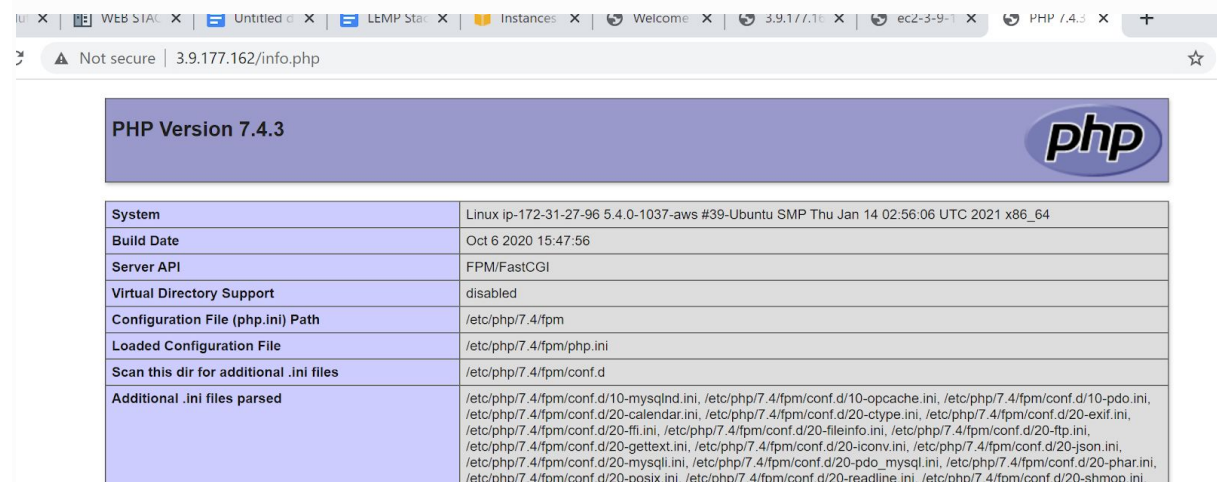
- Type or paste the following lines into the new file. This is the valid PHP code that will return information about your server:

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

- Access this page in your web browser by visiting the domain name or public IP address you've set up in your Nginx configuration file, followed by `/info.php`:

```
http://server_domain_or_IP/info.php
```

```
http://3.9.177.162/info.php
```



System	Linux ip-172-31-27-96 5.4.0-1037-aws #39-Ubuntu SMP Thu Jan 14 02:56:06 UTC 2021 x86_64
Build Date	Oct 6 2020 15:47:56
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/fpm
Loaded Configuration File	/etc/php/7.4/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/fpm/conf.d
Additional .ini files parsed	/etc/php/7.4/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.4/fpm/conf.d/10-opcache.ini, /etc/php/7.4/fpm/conf.d/10-pdo.ini, /etc/php/7.4/fpm/conf.d/20-calendar.ini, /etc/php/7.4/fpm/conf.d/20-ctype.ini, /etc/php/7.4/fpm/conf.d/20-exif.ini, /etc/php/7.4/fpm/conf.d/20-ffi.ini, /etc/php/7.4/fpm/conf.d/20-fileinfo.ini, /etc/php/7.4/fpm/conf.d/20-ftp.ini, /etc/php/7.4/fpm/conf.d/20-gettext.ini, /etc/php/7.4/fpm/conf.d/20-iconv.ini, /etc/php/7.4/fpm/conf.d/20-json.ini, /etc/php/7.4/fpm/conf.d/20-mysqli.ini, /etc/php/7.4/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7.4/fpm/conf.d/20-phar.ini, /etc/php/7.4/fpm/conf.d/20-posix.ini, /etc/php/7.4/fpm/conf.d/20-readline.ini, /etc/php/7.4/fpm/conf.d/20-shmop.ini,

After checking the relevant information about your PHP server through that page, it's best to remove the file you created as it contains sensitive information about your PHP environment and your Ubuntu server. You can use `rm` to remove that file:

```
$ sudo rm /var/www/your_domain/info.php
```

```
sudo rm /var/www/projectLEMP/info.php
```

STEP 6 — Retrieving data from MySQL database with PHP

In this step we will create a test database (DB) - 'Bims' with simple "To do list" and configure access to it, so the Nginx website would be able to query data from the DB and display it.

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. We'll need to create a new user with the `mysql_native_password` authentication method in order to be able to connect to the MySQL database from PHP.

We will create a database named '**Bims**' and a user named '**John**',

- a. Connect to the MySQL console using the **root** account:

```
$ sudo mysql
```

- b. Create a new database

```
mysql> CREATE DATABASE `example_database`;
```

```
mysql> CREATE DATABASE 'Bims';
```

- c. Confirm this creation

```
mysql> Show Databases;
```

```
mysql> Show Databases;
+-----+
| Database |
+-----+
| Bims     |
| information_schema |
+-----+
2 rows in set (0.00 sec)

mysql>
```

- d. Create a new user and grant him full privileges on the database 'Bims'

```
mysql> CREATE USER 'baxy'@ '%' IDENTIFIED WITH mysql_native_password BY 'password';
```

```
mysql> CREATE USER 'John'@ '%' IDENTIFIED WITH mysql_native_password BY 'john';
```

- e. Confirm the users on the Database

```
MySQL> Select user, host from mysql.user;
```

```
mysql> select user, host from mysql.user;
+-----+-----+
| user          | host          |
+-----+-----+
| John          | %             |
| debian-sys-maint | localhost    |
| mysql.infoschema | localhost    |
| mysql.session  | localhost    |
| mysql.sys      | localhost    |
| root          | localhost    |
+-----+-----+
6 rows in set (0.00 sec)

mysql>
mysql> exit
exit
^C
mysql>
mysql> exit
```

f Give this user permission over 'Bims' database:

```
mysql> GRANT ALL ON example_database.* TO 'example_user'@'%';
```

```
mysql> GRANT ALL ON Bims.* TO 'John'@'%';
```

This will give John the full privileges over the Bims database, while preventing this user from creating or modifying other databases on your server.

g. Exit the MySQL shell:

```
mysql> exit
```

h. Test if the new user has the proper permissions by logging in to the MySQL console again, this time using the custom user credentials:

```
$ mysql -u example_user -p
```

```
mysql -u John -p
```

Notice the **-p** flag in this command - This will prompt you for the password used when creating the user (John). After logging in to the MySQL console, confirm that you have access to the 'Bims' DB.:

Output:

```
mysql>
mysql> CREATE DATABASE `Bims`;
Query OK, 1 row affected (0.01 sec)

mysql> CREATE USER 'John'@'%' IDENTIFIED WITH mysql_native_password BY 'john';
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT ALL ON Bims.* TO 'John'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
ubuntu@ip-172-31-27-96:~$
ubuntu@ip-172-31-27-96:~$
ubuntu@ip-172-31-27-96:~$
ubuntu@ip-172-31-27-96:~$ mysql -u John -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.23-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
ubuntu@ip-172-31-27-96:~$ mysql -u John -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.23-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql>
mysql>
mysql> Show Databases;
+-----+
| Database |
+-----+
| Bims     |
| information_schema |
+-----+
2 rows in set (0.00 sec)

mysql>
```

- i. Create a test table named **todo_list**. From the MySQL console by running the following statement:

```
CREATE TABLE example_database.todo_list (
```

```
MySQL> CREATE TABLE Bims.todo_list (  
mysql> item_id INT AUTO_INCREMENT,  
mysql> content VARCHAR(255),  
mysql> PRIMARY KEY(item_id)  
mysql> );
```

- j. Insert a few rows of content in the test table. You might want to repeat the next command a few times, using different VALUES:

```
mysql> INSERT INTO example_database.todo_list (content) VALUES ("My first important item");
```

```
MySQL> INSERT INTO Bims.todo_list (content) VALUES ("My first important item");  
MySQL> INSERT INTO Bims.todo_list (content) VALUES ("My second important item");  
MySQL> INSERT INTO Bims.todo_list (content) VALUES ("My third important item");
```

Output

```
mysql>  
mysql> INSERT INTO Bims.todo_list (content) VALUES ("My first important item");  
Query OK, 1 row affected (0.01 sec)  
  
mysql> INSERT INTO Bims.todo_list (content) VALUES ("My second important item");  
Query OK, 1 row affected (0.01 sec)  
  
mysql> INSERT INTO Bims.todo_list (content) VALUES ("My third important item");  
Query OK, 1 row affected (0.01 sec)  
  
mysql> INSERT INTO Bims.todo_list (content) VALUES ("My fourth important item");  
Query OK, 1 row affected (0.00 sec)  
  
mysql> SELECT * FROM Bims.todo_list;  
+-----+-----+  
| item_id | content  
+-----+-----+  
| 1 | My first important item |  
| 2 | My second important item |  
| 3 | My third important item |  
| 4 | My fourth important item |  
+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql>  
mysql> exit
```

- k. Confirm that the data was successfully saved to your table, run:

```
mysql> SELECT * FROM example_database.todo_list;  
mysql> SELECT * FROM Bims.todo_list;
```

Output

```
mysql> SELECT * FROM Bims.todo_list;  
+-----+-----+  
| item_id | content  
+-----+-----+  
| 1 | My first important item |  
| 2 | My second important item |  
| 3 | My third important item |  
| 4 | My fourth important item |  
+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql>  
mysql> exit  
Bye  
ubuntu@ip-172-31-27-96:~$
```


Exit MySQL after the confirmation

```
mysql> exit
```

1. Create a PHP script that will connect to MySQL and query for your content. Create a new PHP file in your custom web root directory using your preferred editor. We'll use nano for that:

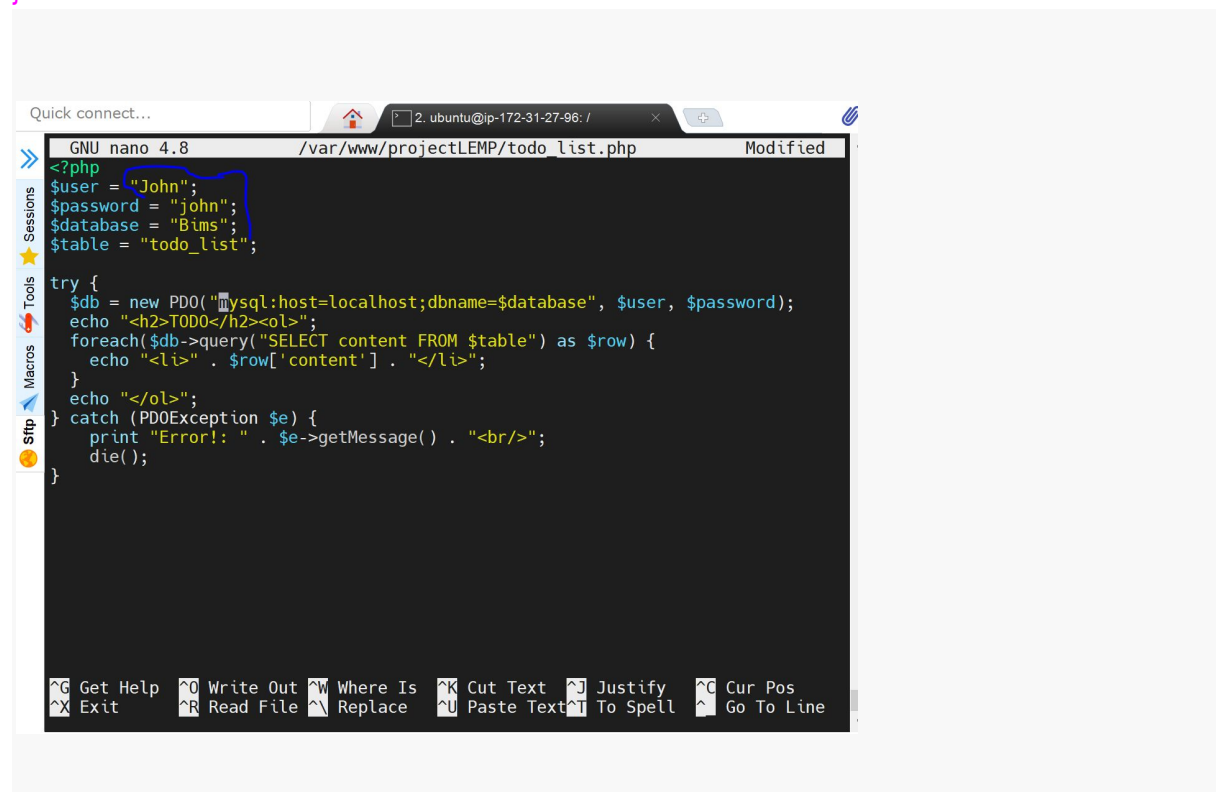
```
$ nano /var/www/projectLEMP/todo_list.php
```

The following PHP script connects to the MySQL database and queries for the content of the **todo_list** table, displays the results in a list. If there is a problem with the database connection, it will throw an exception.

Copy this content into the **todo_list.php** script:

```
<?php
$user = "example_user";
$password = "password";
$database = "example_database";
$table = "todo_list";

try {
    $db = new PDO("mysql:host=localhost;dbname=$database", $user, $password);
    echo "<h2>TODO</h2><ol>";
    foreach($db->query("SELECT content FROM $table") as $row) {
        echo "<li>" . $row['content'] . "</li>";
    }
    echo "</ol>";
} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
```



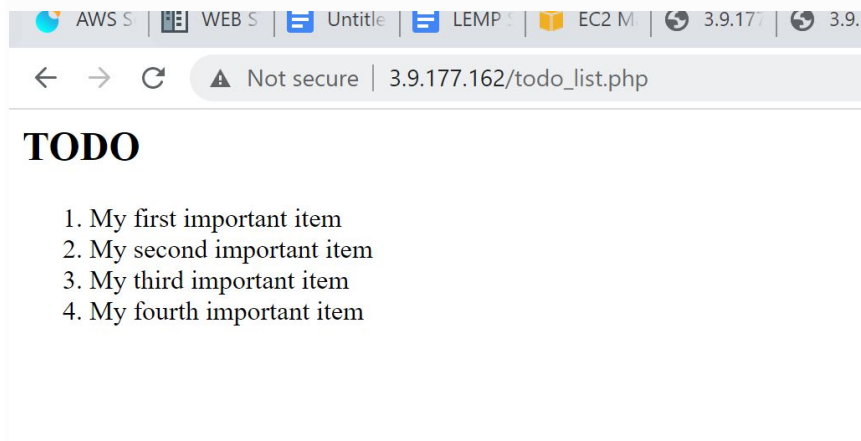
Save and close the file: **CTRL +X; Y; ENTER**

Access this page in your web browser by visiting the domain name or public IP address configured for your website, followed by **/todo_list.php**:

http://<Public_domain_or_IP>/todo_list.php

http://3.9.177.162/todo_list.php

You should see a page like this, showing the content you've inserted in your test table:



The above output confirms that the PHP environment is ready to connect and interact with the MySQL server.

Credits
darey.io

MySQL Syntax: https://www.w3schools.com/sql/sql_syntax.asp

DBMS: https://en.wikipedia.org/wiki/Database#Database_management_system

PHP: <https://www.php.net/manual/en/preface.php>

LEMP: <https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04>

<https://www.scaleway.com/en/docs/installing-lemp-stack-on-ubuntu-linux-nginx-mysql-php/>

MySQL/PHP: <https://linuxize.com/post/how-to-manage-mysql-databases-and-users-from-the-command-line/>