## LEMP Stack Project: My Web Application

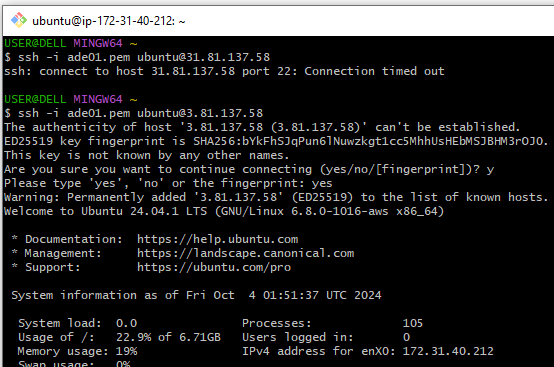
### Introduction

This project demonstrates the implementation of a LEMP stack to create a dynamic web application. The LEMP stack consists of Linux, Nginx, MySQL, and PHP.

### Project Setup

\* \*\*EC2 Instance:\*\* Created a t2.micro EC2 instance in the us-east-1 region and logged in to it using the gitbash. The gitbash gives additional flexibility.

\* \*\*OS Installation:\*\* Installed Ubuntu Server 22.04 LTS.



\* \*\*Nginx Installation:\*\*

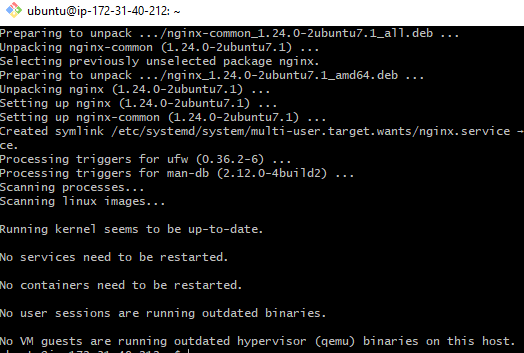
```bash

sudo apt update

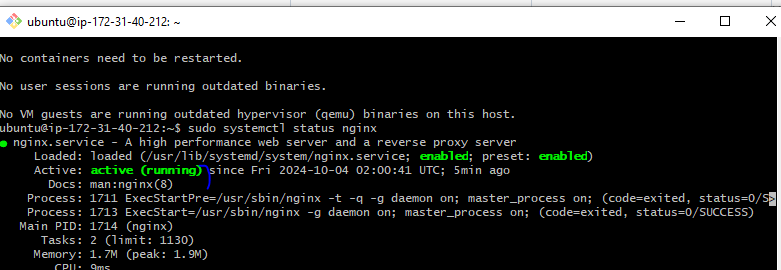
sudo apt install nginx

When prompted type yes.

You will see this after a successful installation.

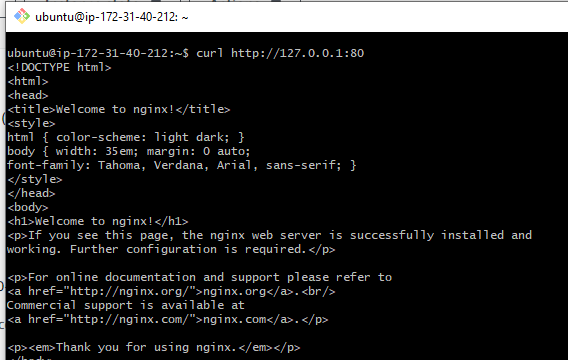


To verify the ngix server is running. Run the command “sudo systemctl status nginx”



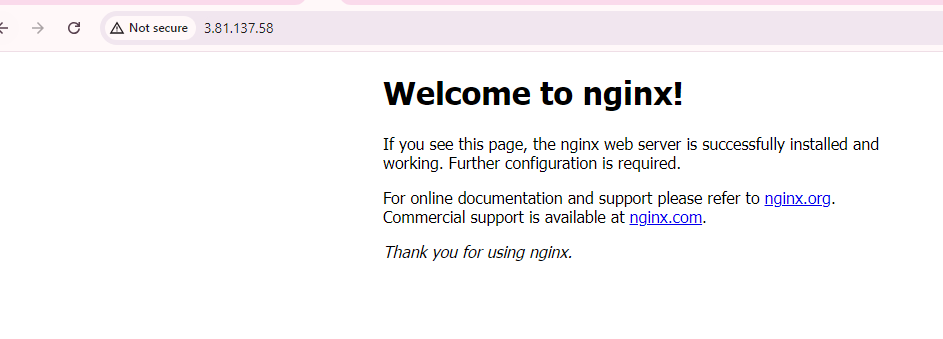
To access the webserver you must ensue port 80 of your ec2 instance is enabled.

You can test connectivity to the webserver locally from your git bash terminal or putty terminal by using <http://127.0.0.1:80>.



You can also test it through a browser by typing <http://3.81.137.58>. Replace it with the public IP address of your ec2 instance.

If you see the image below, this means you have successfully installed your server and completed the second implementation of the stack “LE”

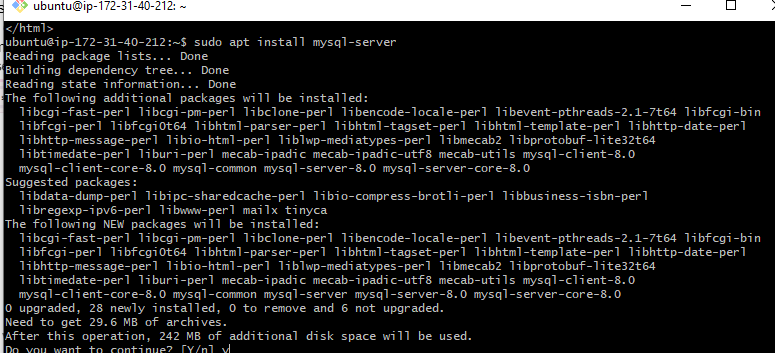


##Installing mysql which is the third component on the stack.

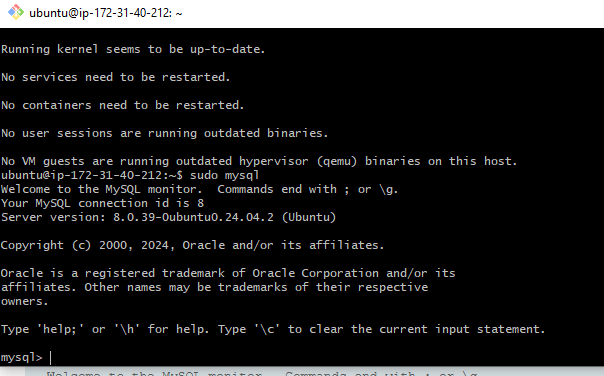
You need to install a [Database Management System (DBMS)](https://en.wikipedia.org/wiki/Database#Database_management_system) to be able to store and manage data for your site in a [relational database](https://en.wikipedia.org/wiki/Relational_database).

#$ sudo apt install MySQL-server

#Enter yes to continue the installation if prompted



After Installation, your screen should look like this.



Type the “sudo mysql” command to log in to mysql console and you will see the prompt changed from ubuntu to mysql. You are also connected as an administrative database root user.

You can also run the security script which comes pre-installed with mysql to give your database a baseline security.

First change the root password, using the command

ALTER USER'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY'PassWord.1';

you can define your password after the word “BY”. mine is “PassWord.1”

Exit mysql console by typing exit

Now, start the interactive script that provides security by running this command.

You will be asked if you want to install the password validation plugin, you can choose Yes or No depending on you.

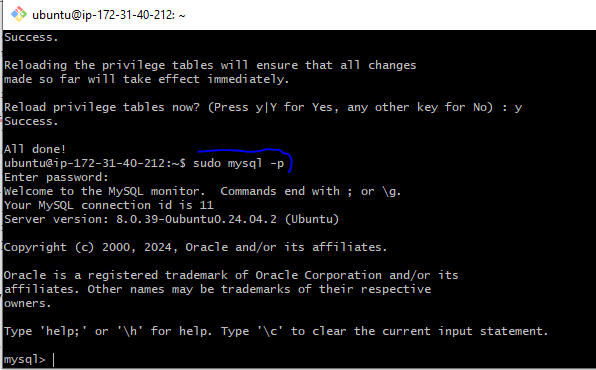
You will be asked if you want to change your Database root password, you can choose Yes or No depending on you.

For other prompts click yes.

Test if you’re able to log in to the MySQL console by typing:

$ sudo mysql -p

Now you have completed the installation of mysql, let's install PHP.



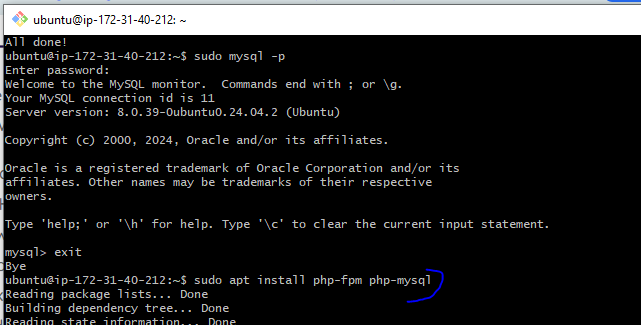
##Installation of PHP

install [PHP](https://www.php.net/) to process code and generate dynamic content for the web server.

One major thing to note is Apache has a PHP interpreter, and nginx needs an external PHP interpreter which will be installed during this installation. Also, we will install php-mysql to facilitate communication between php and the mysql database

## sudo apt install php-fpm php-mysq

l

.

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

Open a new configuration file in Nginx’s sites-available directory using your preferred command-line editor. Here.:$ 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/php8.3-fpm.sock;

}

location ~ /\.ht {

deny all;

}

}

Now, let us configure nginx to use the php processor installed.

We will configure the nginx server to host multiple domains, we will leave the default var/www/html but create a new directory var/www/projectLEMP for our project.

Sudo mkdir var/www/projectLEMP.

Activate your 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/

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

You shall see following message:

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok

nginx: configuration file /etc/nginx/nginx.conf test is successful

If any errors are reported, go back to your configuration file to review its contents before continuing.

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

When you are ready, reload Nginx to apply the changes:

$ sudo systemctl reload nginx

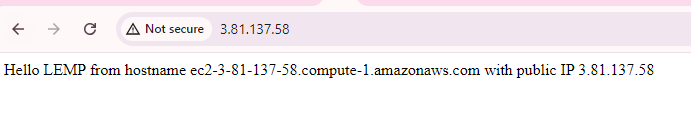
Your new website is now active, but the web root /var/www/projectLEMP is still empty. Create an index.html file in that location so that we can test that your new server block works as expected:

sudo echo 'Hello LEMP from hostname' $(TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` && curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-data/public-hostname) 'with public IP' $(TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` && curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-data/public-ipv4) > /var/www/projectLEMP/index.html

Now go to your browser and try to open your website URL using IP address:

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

You will see the page below if successful.



#### Testing PHP with Nginx

Your LEMP stack should now be completely set up.

At this point, your LAMP stack is completely installed and fully operational.

You can test it to validate that Nginx can correctly hand .php files off to your PHP processor.

You can do this by creating a test PHP file in your document root. Open a new file called info.php within your document root in your text editor:

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

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

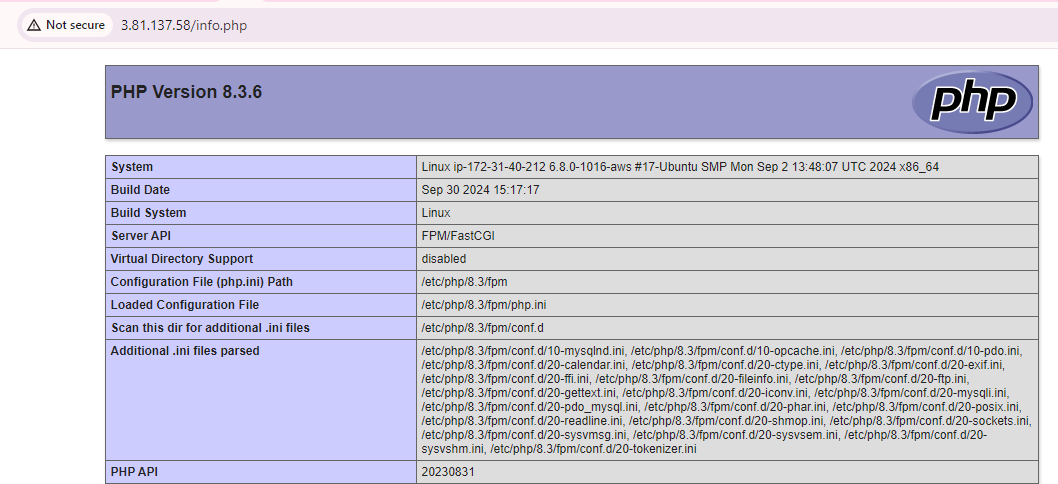
<?php

phpinfo();

You can now 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

You will see a web page containing detailed information about your server:



Retrieving data from mysql using php.

First, connect to the MySQL console using the **root** account:

$ sudo mysql

To create a new database, run the following command from your MySQL console:

mysql> CREATE DATABASE `example\_database`;

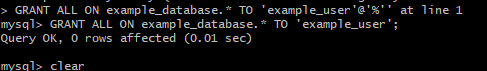
Now you can create a new user and grant him full privileges on the database you have just created.

The following command creates a new user named example\_user, using mysql\_native\_password as default authentication method. We’re defining this user’s password as PassWord.1, but you should replace this value with a secure password of your own choosing.

mysql> CREATE USER 'example\_user'@'%' IDENTIFIED WITH mysql\_native\_password BY 'PassWord.1';

Now we need to give this user permission over the example\_database database:

mysql> GRANT ALL ON example\_database.\* TO 'example\_user'@'%';



This will give the **example\_user** user full privileges over the **example\_database** database, while preventing this user from creating or modifying other databases on your server.

Now exit the MySQL shell with:

mysql> exit

You can 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

Notice the -p flag in this command, which will prompt you for the password used when creating the example\_user user. After logging in to the MySQL console, confirm that you have access to the example\_database database:

mysql> SHOW DATABASES;

This will give you the following output:

Output

+--------------------+

| Database |

+--------------------+

| example\_database |

| information\_schema |

+--------------------+

2 rows in set (0.000 sec)

Next, we’ll create a test table named **todo\_list**. From the MySQL console, run the following statement:

CREATE TABLE example\_database.todo\_list (

mysql> item\_id INT AUTO\_INCREMENT,

mysql> content VARCHAR(255),

mysql> PRIMARY KEY(item\_id)

mysql> );

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");

To confirm that the data was successfully saved to your table, run:

mysql> SELECT \* FROM example\_database.todo\_list;

You’ll see the following output:

Output

+---------+--------------------------+

| item\_id | content |

+---------+--------------------------+

| 1 | My first important item |

| 2 | My second important item |

| 3 | My third important item |

| 4 | and this one more thing |

+---------+--------------------------+

4 rows in set (0.000 sec)

After confirming that you have valid data in your test table, you can exit the MySQL console:

mysql> exit

Now you can 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 vi 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 your todo\_list.php script:

<?php

$user = "example\_user";

$password = "PassWord.1";

$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 when you are done editing.

You can now 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

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

