

# LEMP Stack Implementation

## Introduction

This project demonstrates the implementation of a LEMP stack (Linux, Nginx, MySQL, PHP) to build a dynamic web application. The goal is to create a web-based application that allows users to manage tasks efficiently by leveraging the power of modern web technologies. The project involves setting up an Ubuntu server on AWS, configuring Nginx as the web server, using MySQL for database management, and employing PHP for server-side scripting. By the end of this project, a fully dynamic web application is developed, showcasing the integration of the LEMP stack for building robust and scalable web applications.

## Understanding the LEMP Stack

**Linux:** The operating system that serves as the foundation for the stack.

**Nginx:** The web server that handles incoming requests and serves web pages to users.

**MySQL:** The database management system that stores and organizes data.

**PHP:** The server-side scripting language that acts as the messenger, fetching content from the database for users and delivering user requests to where they're needed.

## Project Objectives

- i. To set up and configure the LEMP stack on an AWS Ubuntu server.
- ii. To develop a custom Nginx server block to serve a web application.
- iii. To implement a MySQL database for storing and managing task data.
- iv. To build a PHP script to retrieve and display task data dynamically.

## Project Steps

**Step 1:** Spin up Ubuntu Server and SSH into the Server

**Step 2:** Install Nginx

**Step 3:** Install MySQL

**Step 4:** Install PHP and its Dependencies

**Step 5:** Configuring Nginx to use PHP Processor

**Step 6:** Testing PHP with Nginx

**Step 7:** Retrieving data from MySQL database with PHP

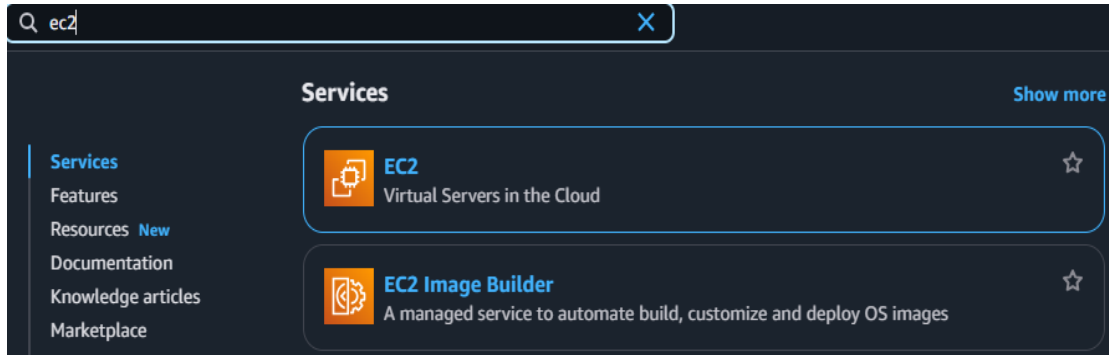
**Step 8:** Create a PHP script to connect to MySQL and query content

## Project Implementation

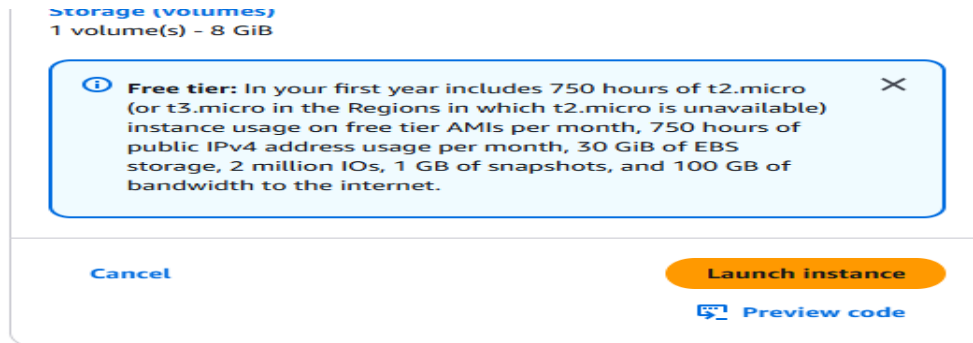
### Step 1: Spin up Ubuntu Server and SSH into the Server

Ubuntu Server is an open-source Linux operating system based on Debian, designed specifically for server environments. It is widely used for web servers, database servers, cloud services, and more.

- Go to your AWS Console and search for ec2 instance and click on it



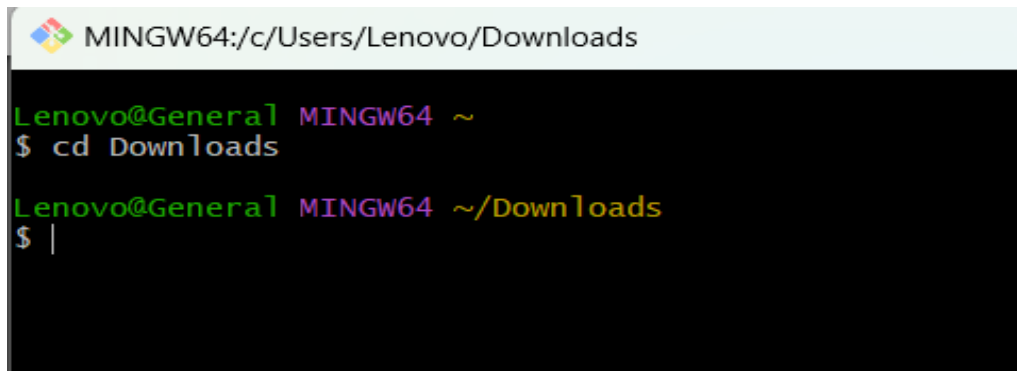
- Click Launch instance -> Give Instance a name -> Select ubuntu as your Application and OS image -> select t2.micro as your instance type -> create a new key pair -> click launch instance



Next, we would need to SSH (Secure Shell) into our newly created instance from our local host. To achieve this;

- Open your git bash and run this command to go to the directory where you save your key pair

**cd Downloads**



- Next we would need to change mode of our key pair. We always do this to enhance security. We would use 400 with means

- ✓ 4 (read permission for the owner)
- ✓ 0 (no permissions for the group)
- ✓ 0 (no permissions for others)

To change the mode of our keypair, run the command

**chmod 400 "Devops.pem"**

```
MINGW64:/c/Users/Lenovo/Downloads

Lenovo@General MINGW64 ~
$ cd Downloads

Lenovo@General MINGW64 ~/Downloads
$ chmod 400 "Devops.pem"

Lenovo@General MINGW64 ~/Downloads
$ |
```

- To SSH into our instance, run command in this format  
**ssh -i {your\_key.pem} ubuntu@{your\_instance\_public\_ip}**  
The command would look like this  
**ssh -i Devops.pem ubuntu@54.159.121.163**

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-84-39:~$ |
```

## Step 2: Install Nginx

Nginx is a widely used for serving web content, handling HTTP requests, and improving the scalability and reliability of web applications.

- Update your instance by running the command  
**sudo apt update**

```
ubuntu@ip-172-31-84-39: ~
ubuntu@ip-172-31-84-39:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2316 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2079 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [387 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2944 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [515 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1187 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [291 kB]
```

- ```
ubuntu@ip-172-31-84-39: ~  
ubuntu@ip-172-31-84-39:~$ sudo apt install nginx  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  fontconfig-config fonts-dejavu-core libdeflate0 libfontconfig1 libgd3 libjbig0 libjpeg-turbo8 libjpeg8  
  libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail  
  libnginx-mod-stream libnginx-mod-stream-geoip2 libtiff5 libwebp7 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 libdeflate0 libfontconfig1 libgd3 libjbig0 libjpeg-turbo8 libjpeg8  
  libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail  
  libnginx-mod-stream libnginx-mod-stream-geoip2 libtiff5 libwebp7 libxpm4 nginx-common nginx-core
```

- ```
ubuntu@ip-172-31-84-39:~$ sudo systemctl start nginx
ubuntu@ip-172-31-84-39:~$ 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: enabled)
   Active: active (running) since Sat 2025-02-15 07:39:09 UTC; 3min 49s ago
     Docs: man:nginx(8)
  Process: 2150 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 2151 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 2244 (nginx)
    Tasks: 2 (limit: 1130)
   Memory: 4.7M
      CPU: 20ms
   CGroup: /system.slice/nginx.service
           └─2244 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─2247 "nginx: worker process"

Feb 15 07:39:09 ip-172-31-84-39 systemd[1]: Starting A high performance web server and a reverse proxy server...
```

- # Welcome to nginx!

*Thank you for using nginx.*

Note: Make sure your port 80 is opened in your instance security group to listen from 0.0.0.0 (anywhere)

### Step 3: Install MySQL

MySQL is a popular open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) for managing and manipulating databases. It is widely used for web applications, data storage, and as a backend database for many software systems. MySQL is known for its reliability, ease of use, and performance.

- To install MySQL, run this command

***sudo apt install mysql-server***

```
ubuntu@ip-172-31-84-39: ~  
ubuntu@ip-172-31-84-39:~$ sudo apt install mysql-server  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  libcgi-fast-perl libcgi-pm-perl libclone-perl libencode-locale-perl libevent-pthreads-2.1-7 libfcgi-bin  
  libfcgi-perl libfcgi0ldbl libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl  
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmecab2 libprotobuf-lite23 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:  
  libdata-dump-perl libipc-sharedcache-perl libbusiness-isbn-perl libwww-perl mailx tinycal  
The following NEW packages will be installed:  
  libcgi-fast-perl libcgi-pm-perl libclone-perl libencode-locale-perl libevent-pthreads-2.1-7 libfcgi-bin  
  libfcgi-perl libfcgi0ldbl libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl  
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmecab2 libprotobuf-lite23 libtimedate-perl  
  liburi-perl mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common  
  mysql-server mysql-server-8.0 mysql-server-core-8.0  
0 upgraded, 28 newly installed, 0 to remove and 35 not upgraded.  
Need to get 29.6 MB of archives.  
After this operation, 243 MB of additional disk space will be used.
```

- Run the security script with these command and follow the prompts;

***sudo mysql\_secure\_installation***

- Test for mysql with this command

***sudo mysql***

```
ubuntu@ip-172-31-84-39: ~  
ubuntu@ip-172-31-84-39:~$ sudo mysql  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 10  
Server version: 8.0.41-0ubuntu0.22.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2025, 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> |
```

### Step 4: Install PHP and its Dependencies

PHP (Hypertext Preprocessor) is a widely-used open-source scripting language that is especially suited for web development and can be embedded into HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

- To install php and its dependencies, run

***sudo apt install php-fpm php-mysql***

```
ubuntu@ip-172-31-84-39: ~  
ubuntu@ip-172-31-84-39:~$ sudo apt install php-fpm php-mysql  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  bzip2 mailcap mime-support php-common php8.1-cli php8.1-common php8.1-fpm php8.1-mysql php8.1-opcache  
  php8.1-readline  
Suggested packages:  
  bzip2-doc php-pear  
The following NEW packages will be installed:  
  bzip2 mailcap mime-support php-fpm php-mysql php8.1-cli php8.1-common php8.1-fpm php8.1-mysql  
  php8.1-opcache php8.1-readline  
0 upgraded, 12 newly installed, 0 to remove and 35 not upgraded.  
Need to get 5389 kB of archives.  
After this operation, 22.2 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://us-east-1-ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bzip2 amd64 1.0.8-5build1 [24.8 kB]
```

- To verify if PHP is installed, run the command

***php -v***

```
ubuntu@ip-172-31-84-39: ~  
ubuntu@ip-172-31-84-39:~$ php -v  
PHP 8.1.2-1ubuntu2.20 (cli) (built: Dec 3 2024 20:14:35) (NTS)  
Copyright (c) The PHP Group  
Zend Engine v4.1.2, Copyright (c) Zend Technologies  
with Zend OPcache v8.1.2-1ubuntu2.20, Copyright (c), by Zend Technologies  
ubuntu@ip-172-31-84-39:~$
```

### Step 5: Configuring Nginx to use PHP Processor

Configuring Nginx to work with a PHP processor is like setting up a team to handle different tasks on a website. Nginx is great at delivering static content (like images or plain HTML files) quickly, but it doesn't know how to run PHP code, which is used to create dynamic, interactive web pages (like login forms or shopping carts). To solve this, Nginx hands off PHP requests to PHP-FPM, a helper that specializes in running PHP scripts. Once PHP-FPM processes the code, it sends the result back to Nginx, which then delivers it to the user's browser. This teamwork makes your website faster and more efficient, ensuring that both static and dynamic parts of your site work smoothly together.

- Create a root web directory for your website. Run  
***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***
- With the help of a text editor, open a new configuration file in Nginx's sites-available directory using your preferred command-line editor  
***sudo vi /etc/nginx/sites-available/projectLEMP***
- Paste and save the below script

```
#/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;  
    }  
}
```

```
}  
}
```

```
ubuntu@ip-172-31-84-39: ~  
# /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;  
    }  
}
```

- Activate your configuration by linking to the config file from Nginx's sites-enabled directory: This will tell Nginx to use the configuration next time it is reloaded. Run  
**`sudo ln -s /etc/nginx/sites-available/projectLEMP /etc/nginx/sites-enabled/`**

- We would check the configuration for syntax errors. Run  
**`sudo nginx -t`**

```
ubuntu@ip-172-31-84-39: ~  
ubuntu@ip-172-31-84-39:~$ 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-172-31-84-39:~$
```

- 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`**
- Reload Nginx to apply the changes. Run;  
**`sudo systemctl reload nginx`**

The new website is now active, but the web root /var/www/projectLEMP is still empty. We will create an index.html file in that location so that we can test that our new server block works as expected

- Navigate to your Nginx web directory. Run;  
**`cd /var/www/projectLEMP`**
- Create a file called index.html  
**`touch index.html`**

- Use a text editor to edit and paste your html file in the index.html file

***sudo vi index.html***

```
ubuntu@ip-172-31-84-39: /var/www/projectLEMP
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My ProjectLEMP Page</title>
</head>
<body>
  <h1>Welcome to My ProjectLEMP!</h1>
  <p>This is a simple HTML page served by Nginx.</p>
</body>
</html>
```

- Next, we would need to change owner and mode so that Nginx can serve our html file  
***sudo chown -R www-data:www-data /var/www/projectLEMP***  
***sudo chmod -R 755 /var/www/projectLEMP***
- Restart Nginx  
***sudo systemctl restart nginx***

Open your web browser and navigate to your server's IP address to see your html file.



## Step 6: Testing PHP with Nginx

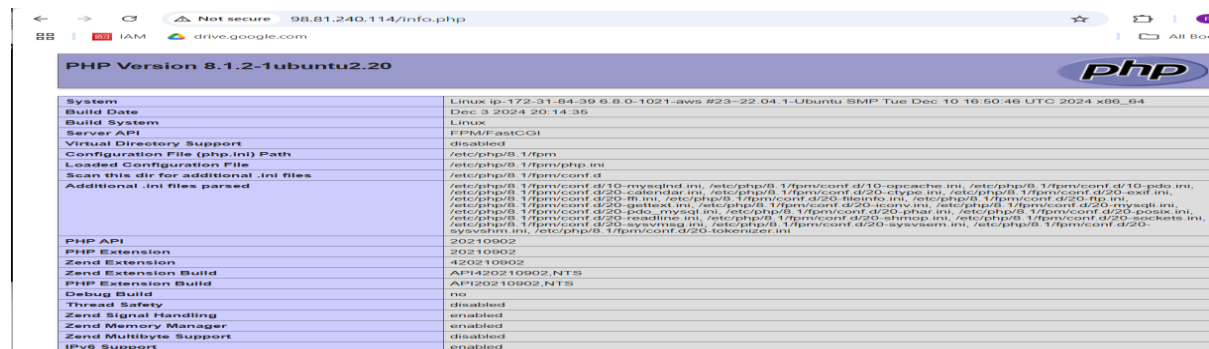
Nginx doesn't natively process PHP—it relies on PHP-FPM to handle PHP requests. Proper configuration is required to ensure Nginx communicates correctly with PHP-FPM, and issues like misconfigurations, PHP-FPM not running, or file permission errors can prevent PHP scripts from working. Testing confirms everything is set up correctly.

- Edit your web directory with a text editor. Run  
***sudo vi /var/www/projectLEMP/info.php***
- Paste the below and save  
***<?php***  
***phpinfo();***

```
ubuntu@ip-172-31-84-39: /var/www/projectLEMP
<?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



PHP Version 8.1.2-1ubuntu2.20	
System	Linux ip-172-31-84-39 6.8.0-1021-aws #23-22.04.1-Ubuntu SMP Tue Dec 10 16:50:46 UTC 2024 x86_64
Build Date	Dec 3 2024 20:14:36
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/fpm
Loaded Configuration File	/etc/php/8.1/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/fpm/conf.d
Additional .ini files parsed	/etc/php/8.1/fpm/conf.d/10-mysqlnd.ini, /etc/php/8.1/fpm/conf.d/10-opcache.ini, /etc/php/8.1/fpm/conf.d/10-pdo.ini, /etc/php/8.1/fpm/conf.d/20-calendar.ini, /etc/php/8.1/fpm/conf.d/20-ctype.ini, /etc/php/8.1/fpm/conf.d/20-exif.ini, /etc/php/8.1/fpm/conf.d/20-gdlib.ini, /etc/php/8.1/fpm/conf.d/20-gettext.ini, /etc/php/8.1/fpm/conf.d/20-iconv.ini, /etc/php/8.1/fpm/conf.d/20-mysql.ini, /etc/php/8.1/fpm/conf.d/20-pdo_mysql.ini, /etc/php/8.1/fpm/conf.d/20-phar.ini, /etc/php/8.1/fpm/conf.d/20-posix.ini, /etc/php/8.1/fpm/conf.d/20-readline.ini, /etc/php/8.1/fpm/conf.d/20-shmop.ini, /etc/php/8.1/fpm/conf.d/20-sockets.ini, /etc/php/8.1/fpm/conf.d/20-sysvshm.ini, /etc/php/8.1/fpm/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902.NTS
PHP Extension Build	API20210902.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled

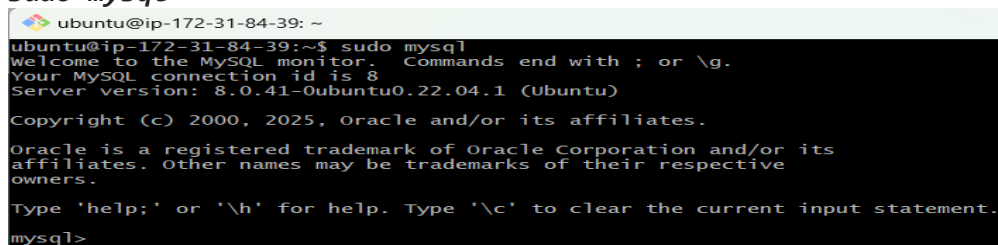
It is advisable you remove this page after checking it because it contains sensitive information about your server. To remove run this command;

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

## Step 7: Retrieving Data from MySQL Database with PHP

Retrieving data from a MySQL database using PHP involves connecting to the database, executing a query to fetch the desired data, and then processing or displaying the results.

- connect to the MySQL console using the root account: with this command  
**sudo mysql**



```
ubuntu@ip-172-31-84-39: ~  
mysql>  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 8  
Server version: 8.0.41-0ubuntu0.22.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2025, 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>
```

- create a new database using this command  
**CREATE DATABASE osita\_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 osita\_user, using mysql\_native\_password as default authentication method. We're defining this user's password as Osita@1989. Run this command

```
CREATE USER 'osita_user'@'%' IDENTIFIED WITH mysql_native_password BY 'Osita@1989';  
CREATE USER 'webaccess'@'%' IDENTIFIED BY 'Osita@1989';
```

- To check for users, run;  
**SELECT User, Host FROM mysql.user;**

```
mysql> SELECT User, Host FROM mysql.user;
+-----+-----+
| User           | Host           |
+-----+-----+
| osita_user     | %              |
| webaccess      | %              |
| debian-sys-maint | localhost      |
| mysql.infoschema | localhost      |
| mysql.session  | localhost      |
| mysql.sys      | localhost      |
| root           | localhost      |
+-----+-----+
7 rows in set (0.00 sec)
```

- Now we need to give this user permission over the osita\_database database  
**GRANT ALL ON osita\_database.\* TO 'osita\_user'@'%';**

You can test if the new user has the proper permissions by logging out and into the MySQL console again, this time using the custom user credentials;

**mysql -u osita\_user -p**

```
ubuntu@ip-172-31-84-39: ~
ubuntu@ip-172-31-84-39:~$ mysql -u osita_user -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.41-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2025, 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>
```

- Next, we'll create a test table named **todo\_list**. From the MySQL console, run the following statement:
- **CREATE TABLE osita\_database.todo\_list (item\_id INT AUTO\_INCREMENT, content VARCHAR(255), PRIMARY KEY(item\_id));**
- Insert a few rows of content in the test table. You might want to repeat the next command a few times, using different VALUES:  
**INSERT INTO osita\_database.todo\_list (content) VALUES ("My first important item");**  
**INSERT INTO osita\_database.todo\_list (content) VALUES ("My second important item");**  
**INSERT INTO osita\_database.todo\_list (content) VALUES ("My third important item");**  
**INSERT INTO osita\_database.todo\_list (content) VALUES ("My fourth important item");**
- To confirm that the data was successfully saved to your table, run:  
**SELECT \* FROM osita\_database.todo\_list;**

```
mysql> SELECT * FROM osita_database.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)
```

Exit from the mysql

## Step 8: Create a PHP Script to connect to MySQL and Query Content

Creating a PHP script to connect to MySQL and query content is essential for dynamic web applications that rely on database interactions. PHP, being a server-side scripting language, allows you to connect to a MySQL database using the mysqli or PDO extension, execute SQL queries, and retrieve or manipulate data.

- Create a new PHP file in your custom web root directory. Run;  
***sudo vi /var/www/projectLEMP/todo\_list.php***
- Paste the below configuration and save the file

```
<?php

$user = "osita_user";

$password = "Osita@1989";

$database = "osita_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) {

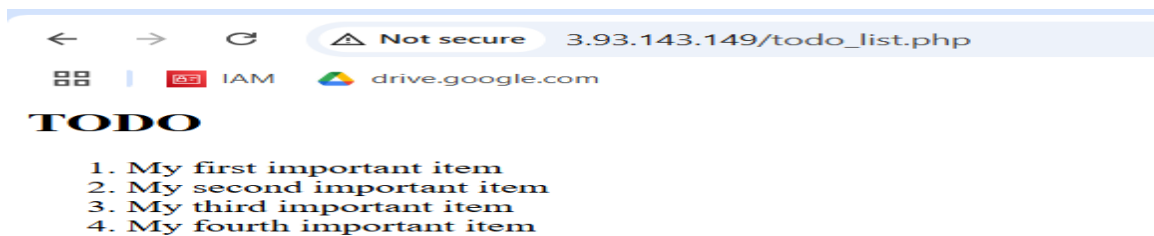
    print "Error!: " . $e->getMessage() . "<br/>";

    die();

}
```

```
ubuntu@ip-172-31-84-39: ~  
<?php  
$user = "osita_user";  
$password = "Osita@1989";  
$database = "osita_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();  
}  
~
```

- 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***



## Conclusion

In this project, we successfully implemented a LEMP stack on an AWS Ubuntu server to build a dynamic task management web application. By configuring Nginx as the web server, integrating MySQL for database management, and utilizing PHP for server-side scripting, we created a robust and scalable solution for handling user tasks efficiently. The project demonstrated the power of modern web technologies in delivering dynamic content while ensuring optimal performance and security.

Through this implementation, we gained practical experience in setting up and managing a LEMP environment, customizing server configurations, and developing database-driven web applications. This project serves as a foundation for future enhancements, such as adding authentication, user roles, or API integrations, to further improve functionality and usability.