

# LEMP STACK IMPLEMENTATION

Github Link: <https://github.com/Macbeth111/Dev-Ops-.git>

LinkedIn: <https://www.linkedin.com/feed/update/urn:li:activity:7254893625946333184/>

<b>PREREQUISITE:</b>	<b>2</b>
<b>STEP 1: INSTALL THE NGINX WEB SERVER</b>	<b>3</b>
##Install Nginx server	3
###Ensuring EC2 Was Receiving Traffic	4
###Checked Subnet Internet Access	4
###Allocated an Elastic IP	4
###Configured Security Group and Network ACL	5
###Tested Network Connectivity	5
##Install Nginx server	6
<b>STEP 2: INSTALL MYSQL SERVER</b>	<b>7</b>
## Install MySQL Server	8
## Log into the MySQL Console	8
## Set the Root User Password	8
## Exit the MySQL Console	9
## Run the MySQL Secure Installation Script	9
## Configure the VALIDATE PASSWORD PLUGIN	10
## Select the Level of Password Validation	10
## Set and Confirm Root User Password	11
<b>STEP 3: INSTALL PHP SERVER</b>	<b>11</b>
<b>STEP 4: CONFIGURING NGINX TO USE PHP PROCESSOR</b>	<b>12</b>
##Create root directory and change ownership	12
##Create new server block for your project	12
##Explanation of Directives and Location Blocks:	13
##Activate configuration	14
##Disable default Nginx host and reload new configuration	15
<b>STEP 5: TESTING PHP WITH NGINX</b>	<b>16</b>
##Configure Nginx to process PHP files	16
<b>STEP 6: RETRIEVING DATA FROM MYSQL DATABASE WITH PHP</b>	<b>17</b>
##Connect to MySQL with root privileges and create database	17
##Create user and assign privileges	18
<b>WHAT I LEARNED, CHALLENGES I FACED AND HOW I OVERCAME THEM:</b>	<b>22</b>

# PREREQUISITE:

Pre-requisite: Connecting to EC2 Instance via Gitbash

In this study, I initiated the process by connecting to my EC2 instance using Gitbash, a terminal emulator for Git that allows SSH connections. To connect, I followed these steps:

1. Accessing the EC2 instance: I selected my EC2 instance from the AWS Management Console, clicked on the "Connect" button, and chose the "SSH Client" option.
2. Running the SSH command: AWS provided a sample SSH command, which I executed in Gitbash as follows:

```
- ssh -i "your-key-pair-name.pem" ubuntu@ec2-172-31-17-208.us-west-2.compute.amazonaws.com
```

EC2 Dashboard

EC2 Global View

Events

**Instances**

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations [New](#)

**Images**

AMIs

AMI Catalog

**Elastic Block Store**

Volumes

Snapshots

Lifecycle Manager

**Instances (1/1) Info**

Last updated 1 minute ago [Refresh](#) [Connect](#) [Instance state](#)

[Actions](#) [Launch instances](#)

[All states](#)

[Instance state = running](#) [Clear filters](#)

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state
<input checked="" type="checkbox"/>	learning	i-0e3417d50f1774761	Running

**i-0e3417d50f1774761 (learning)**

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#)

**Instance summary**

Instance ID  
i-0e3417d50f1774761

Private IPv4 addresses  
172.31.17.208

Instance state  
Running

Public IPv4 address copied  
35.87.151.65 | [open address](#)

IPv6 address  
-

Public IPv4 DNS  
-

```

AARONS@LAPTOP-U0SC9GV7 MINGW64 ~
$ cd Downloads

AARONS@LAPTOP-U0SC9GV7 MINGW64 ~/Downloads
$ ssh -i "learnlemp.pem" ubuntu@ec2-35-87-151-65.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-35-87-151-65.us-west-2.compute.amazonaws.com (35.87.151.65)' can't be est
ablished.
ED25519 key fingerprint is SHA256:x5EnFTxpUBlcpruEnED1j/JwSNyJC05FZA1IoZatZss.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-35-87-151-65.us-west-2.compute.amazonaws.com' (ED25519) to the list of
known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Oct 23 10:09:48 UTC 2024

System load: 0.32          Processes:            106
Usage of /:  22.8% of 6.71GB Users logged in:       0
Memory usage: 21%          IPv4 address for enX0: 172.31.17.208
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

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-172-31-17-208:~$ |

```

# STEP 1: INSTALL THE NGINX WEB SERVER

## ##Install Nginx server

I first attempted to install the Nginx web server. The initial step in installing any new software on a Linux machine involves updating the system's package list. This ensures that the most up-to-date versions of the required packages are installed.

```
sudo apt update
```

```
sudo apt install nginx
```

Unfortunately, the Nginx installation failed. To troubleshoot, I conducted a connectivity test by running the following command:

```
ping google.com
```

The result indicated a 100% packet loss, suggesting that the instance was not receiving traffic. Recognizing this as a networking issue, I embarked on problem-solving measures to rectify the situation.

## ###Ensuring EC2 Was Receiving Traffic

### ###Checked Subnet Internet Access

I began by checking if my EC2 instance was in a public subnet with proper internet access.

- **Verified the Public Subnet:**
  - I navigated to the **VPC Dashboard** in AWS.
  - Under the **Subnets** section, I checked the subnet my instance was in, ensuring it had a **Route Table** with a route to the **Internet Gateway (IGW)**.
  - I verified that the route table included a route directing 0.0.0.0/0 traffic to the Internet Gateway (IGW).
- **Checked the Internet Gateway:**
  - In the **VPC Dashboard**, I went to **Internet Gateways**.
  - I confirmed that the Internet Gateway was attached to the VPC where my instance resided.

### ###Allocated an Elastic IP

I realized that the EC2 Connect key was using the private IP address, and my instance didn't have public internet access. This is where I needed to assign an **Elastic IP** to my instance to access it properly over the internet.

- **Checked for Elastic IP:**
  - I went to the **EC2 Dashboard** and selected my instance.
  - Under the **Description** tab, I confirmed that there was no **Public IPv4 address** assigned, which explained why I couldn't access the instance from outside.
- **Allocated an Elastic IP:**
  - I went to **Elastic IPs** under the EC2 Dashboard.
  - I clicked on **Allocate Elastic IP Address** and followed the steps to allocate one.
  - Once I had the Elastic IP, I **associated** it with my EC2 instance.
- **Reason for Elastic IP:**
  - Without a public IP, the EC2 Connect key was only using the **private IP**, which meant that traffic couldn't reach the instance from outside the VPC.

✕ Elastic IP address allocated successfully. Elastic IP address 50.112.244.217 Associate this Elastic IP address ✕

Elastic IP addresses (1/1) Refresh Actions Allocate Elastic IP address

Find resources by attribute or tag

Public IPv4 address: 50.112.244.217 ✕ Clear filters < 1 > Settings

<input checked="" type="checkbox"/>	Name	Allocated IPv4 address	Type	Allocation ID	Reverse DNS record
<input checked="" type="checkbox"/>	-	50.112.244.217	Public IP	eipalloc-071794bb4afd19237	-

New

Allocated IPv4 address  
50.112.244.217

Type  
Public IP

Allocation ID  
eipalloc-071794bb4afd19237

Reverse DNS record  
-

Association ID  
-

Scope  
VPC

Associated instance ID  
-

Private IP address  
-

Network interface ID  
-

Network interface owner account ID  
-

Public DNS  
-

NAT Gateway ID  
-

### ###Configured Security Group and Network ACL

Once the Elastic IP was allocated, I verified that both **inbound** and **outbound traffic** were allowed through the security groups and network ACLs.

#### Step-by-Step Process:

- **Checked Outbound Rules in Security Groups:**
  - In the **EC2 Dashboard**, I went to **Security Groups**.
  - I confirmed that the **outbound rules** allowed all traffic to 0.0.0.0/0 on all ports.
- **Checked Network Access Control List (NACL):**
  - I went to the **VPC Dashboard** and checked the **NACL** for the subnet where my instance was located.
  - I verified that both **inbound** and **outbound rules** allowed traffic from 0.0.0.0/0 for all ports.

### ###Tested Network Connectivity

After configuring the subnet and assigning the Elastic IP, I tested the connectivity to ensure everything was working.

#### Step-by-Step Process:

- **Pinged a website:**
  - I ran the following command to ping a known external site (Google) from my instance:

*ping google.com*

```
ubuntu@ip-172-31-17-208:~$ ping google.com
PING google.com (142.250.69.206) 56(84) bytes of data.
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=1 ttl=58 time=7.74 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=2 ttl=58 time=7.76 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=3 ttl=58 time=7.89 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=4 ttl=58 time=7.98 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=5 ttl=58 time=8.45 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=6 ttl=58 time=8.05 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=7 ttl=58 time=7.77 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=8 ttl=58 time=8.04 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=9 ttl=58 time=8.01 ms
64 bytes from sea30s08-in-f14.1e100.net (142.250.69.206): icmp_seq=10 ttl=58 time=8.31 ms
^C
--- google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9006ms
rtt min/avg/max/mdev = 7.739/7.999/8.453/0.223 ms
ubuntu@ip-172-31-17-208:~$
```

### ##Install Nginx server

Now I run the following codes to update linux, install nginx and ensure that nginx was installed, active and running ;

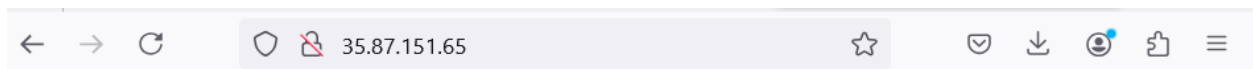
- *sudo apt update*

- `sudo apt install nginx`
- `sudo systemctl status nginx`

```
ubuntu@ip-172-31-17-208: ~  
ubuntu@ip-172-31-17-208:~$ sudo system status nginx  
sudo: system: command not found  
ubuntu@ip-172-31-17-208:~$ sudo systemctl status nginx  
● nginx.service - A high performance web server and a reverse proxy server  
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)  
   Active: active (running) since Wed 2024-10-23 10:49:25 UTC; 1min 45s ago  
     Docs: man:nginx(8)  
  Process: 1900 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)  
  Process: 1906 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)  
 Main PID: 1907 (nginx)  
    Tasks: 2 (limit: 1130)  
  Memory: 1.7M (peak: 1.9M)  
     CPU: 13ms  
   CGroup: /system.slice/nginx.service  
           └─1907 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"  
             └─1908 "nginx: worker process"  
  
Oct 23 10:49:24 ip-172-31-17-208 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server:  
Oct 23 10:49:25 ip-172-31-17-208 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server:  
lines 1-16/16 (END)
```

Now, I checked if my nginx server could respond to request from the internet using

<https://35.87.151.65:80>



## 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](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

## STEP 2: INSTALL MYSQL SERVER

Here is a step-by-step guide for installing MySQL on a Linux-based system.

To start, I installed the MySQL server using the following command:

### ## Install MySQL Server

- *\$ sudo apt install mysql-server*

Here, sudo gives me administrator privileges to install software, apt is the package manager used by Debian-based systems (like Ubuntu), and mysql-server is the actual package I'm installing. When prompted, I confirm the installation by typing y and pressing ENTER.

### ## Log into the MySQL Console

Once MySQL is installed, I log into the MySQL console to begin configuring the server:

- *\$ sudo mysql*

This command starts the MySQL client. Since I use sudo, I'm logging in as the root user (the administrative user for MySQL), giving me full access to the server's management interface.

### ## Set the Root User Password

Next, I secure my MySQL installation by setting a password for the root user. Inside the MySQL console, I run this command:

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

This command changes the password for the root user (root), who can connect only from the local machine (localhost). The password is set to 'PassWord.1'. It's important to set a strong password at this stage to prevent unauthorized access to the server.



```
ubuntu@ip-172-31-17-208: ~  
ubuntu@ip-172-31-17-208:~$ sudo mysql  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 8  
Server version: 8.0.39-0ubuntu0.24.04.2 (Ubuntu)  
  
Copyright (c) 2000, 2024, 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> ALTERUSER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Password.1';  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL  
server version for the right syntax to use near 'ALTERUSER 'root'@'localhost' IDENTIFIED WITH mysql_n  
ative_password BY 'Password.1'' at line 1  
mysql> exit  
Bye  
ubuntu@ip-172-31-17-208:~$
```

## ## Exit the MySQL Console

After setting the root password, I exit the MySQL console by typing:

- `mysql> exit`

Once the root password is set, the user exits the MySQL console by typing `exit`. This closes the connection to the MySQL server and returns the user to the regular terminal prompt.

## ## Run the MySQL Secure Installation Script

To further enhance security, I run the secure installation script

- `$ sudo mysql_secure_installation`

This script performs several tasks, such as removing anonymous users and disabling remote root login. It's a crucial step for locking down unnecessary features and improving the overall security of MySQL. I recommend accepting the default options for most questions during this process.

```

ubuntu@ip-172-31-17-208:~$ 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: Y

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

Skipping password set for root as authentication with auth_socket is used by default.
If you would like to use password authentication instead, this can be done with the "ALTER_USER" command.
See https://dev.mysql.com/doc/refman/8.0/en/alter-user.html#alter-user-password-management for more information.

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

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

```

## ## Configure the VALIDATE PASSWORD PLUGIN

During the secure installation script, I'm asked if I want to enable the VALIDATE PASSWORD PLUGIN. This plugin enforces stronger passwords by testing their strength. I choose whether to enable it by typing:

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

This plugin is optional but highly recommended for production environments. It ensures that any passwords I create in the future meet certain strength criteria.

## ## Select the Level of Password Validation

If I enable the plugin, I'm prompted to choose the level of password validation. The three levels are:

1. **LOW:** Requires passwords to be at least 8 characters long.
2. **MEDIUM:** Requires passwords to be at least 8 characters long, with a mix of numeric digits, uppercase and lowercase letters, and special characters.

3. **STRONG:** Requires all of the above and prevents passwords from being common dictionary words.

## ## Set and Confirm Root User Password

Regardless of whether I enabled the password validation plugin, the script asks me to set and confirm the password for the **MySQL root user**. This should not be confused with the system's root user.

Setting a strong password for the MySQL root user ensures that no one can access the MySQL server without authorization. Even though I set the password earlier, this step adds an extra layer of security. After setting the password, I used it to login to mySQL to ensure that it is working.

```
ubuntu@ip-172-31-17-208: ~  
ubuntu@ip-172-31-17-208:~$ sudo mysql -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 11  
Server version: 8.0.39-0ubuntu0.24.04.2 (Ubuntu)  
  
Copyright (c) 2000, 2024, 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>
```

**NOW MYSQL IS WORKING PROPERLY**

## STEP 3: INSTALL PHP SERVER

Here is a step-by-step guide for installing PHP on a Linux-based system.

To start, I install the PHP server using the following command:

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

## STEP 4: CONFIGURING NGINX TO USE PHP PROCESSOR

In this step, I'll be configuring the Nginx web server to use the PHP processor. By setting up server blocks (similar to virtual hosts in Apache), I can host multiple domains on the same server. For the purposes of this guide, I'll use projectLEMP as the domain name example.

Since I'm using Ubuntu 20.04, I know that Nginx has a single server block enabled by default, serving files from the `/var/www/html` directory. While this works well for one site, managing multiple sites would be difficult. Instead of modifying `/var/www/html`, I'll create a new directory for my domain within `/var/www/`, leaving `/var/www/html` as the default fallback directory for any unmatched requests.

### ##Create root directory and change ownership

First, I created the root web directory for my domain as follows:

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

Next, I changed the ownership of the directory to my current system user by using the `$USER` environment variable:

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

### ##Create new server block for your project

Then, I created a new server block configuration file for **projectLEMP** in Nginx's sites-available directory. I'll use nano as my text editor:

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

```
ubuntu@ip-172-31-17-208: ~
GNU nano 7.2 /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/php8.3-fpm.sock;
    }

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

## ##Explanation of Directives and Location Blocks:

1. **listen 80:**
  - This tells Nginx to listen for incoming HTTP connections on port 80 (the default HTTP port).
2. **server\_name projectLEMP www.projectLEMP:**
  - Specifies the domain names (projectLEMP and www.projectLEMP) that this server block should respond to.
3. **root /var/www/projectLEMP:**
  - Defines the root directory where the site's files are located. All requests will look for files in /var/www/projectLEMP.
4. **index index.html index.htm index.php:**
  - Defines the default index files to serve when accessing a directory. If index.html isn't found, it will try index.htm, and then index.php.
5. **location / { try\_files \$uri \$uri/ =404; }:**
  - This block handles all requests to the root ("/").
  - try\_files \$uri \$uri/ =404; tries to serve the requested URI as a file. If the file doesn't exist, it tries to find a directory. If neither is found, it returns a 404 error.

6. **location ~ .php\$ { include snippets/fastcgi-php.conf; fastcgi\_pass unix:/var/run/php/php8.3-fpm.sock; }:**

- This block matches PHP files (with .php extension).
- `include snippets/fastcgi-php.conf;` loads configuration for processing PHP files.
- `fastcgi_pass unix:/var/run/php/php8.3-fpm.sock;` passes PHP requests to the PHP FastCGI processor via a UNIX socket.

7. **location ~ /\.ht { deny all; }:**

- This block denies access to any files starting with .ht (like .htaccess). This is a security measure to prevent exposure of sensitive files.

By doing this, I set up Nginx to host my **projectLEMP** domain, ensuring that it doesn't interfere with any existing sites or configurations.

## ##Activate configuration

Afterwards, I went ahead to activate the configuration I had created for my Nginx server. To do this, I created a symbolic link from the configuration file in the sites-available directory to the sites-enabled directory. This step was necessary because Nginx reads configuration files from sites-enabled, and linking my custom configuration ensured it would be loaded the next time Nginx restarted. I used the code:

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

After linking the configuration, I wanted to ensure everything was set up correctly without any syntax errors. I ran the Nginx syntax test command:

```
- sudo nginx -t
```

The output confirmed that the syntax was correct and the test was successful, so I knew the configuration was ready for use:

```
ubuntu@ip-172-31-17-208:~$ sudo ln -s /etc/nginx/sites-available/projectLEMP /etc/nginx/sites-enabled/
ubuntu@ip-172-31-17-208:~$ 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-172-31-17-208:~$ |
```

## ##Disable default Nginx host and reload new configuration

Next, I needed to disable the default Nginx host, which was currently listening on port 80. This was important because it could conflict with my newly created configuration if both were active. To disable the default configuration, I ran:

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

With the default configuration removed, I reloaded Nginx to apply the changes and load the new configuration:

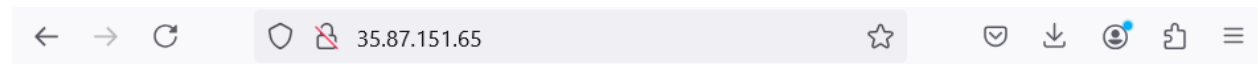
```
- sudo systemctl reload nginx
```

At this point, my new server block was active, but the web root directory (/var/www/projectLEMP) was still empty. To test that the server was functioning correctly, I created a simple index.html file. This file would serve as a placeholder to verify that the server was properly responding to requests.

```
- TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
```

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

This command created an index.html file in the root directory, which displayed a message with the hostname and public IP address of the server. This allowed me to verify that the new server block was working as expected by visiting the website in a browser. By the end of this process, I had successfully configured PHP to work with Nginx, ensured the configuration was error-free, and set up a basic placeholder page to verify that the server was functioning correctly.



Hello LEMP from hostname ec2-35-87-151-65.us-west-2.compute.amazonaws.com with public IP 35.87.151.65

**AT THIS POINT LEMP WAS FULLY CONFIGURED**

# STEP 5: TESTING PHP WITH NGINX

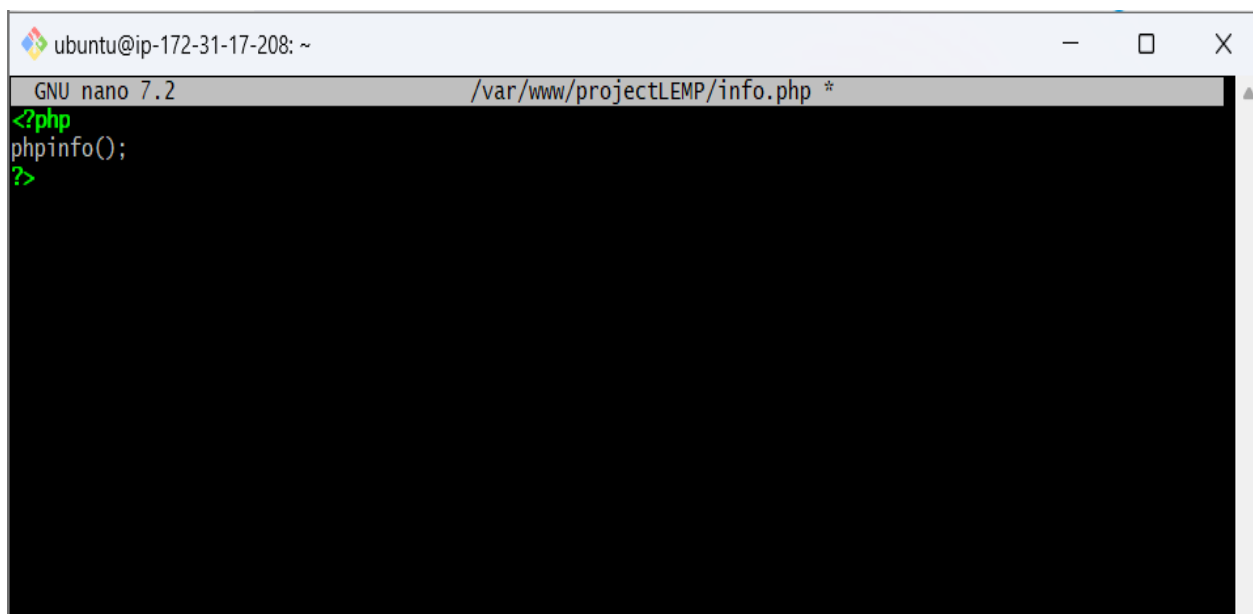
## ##Configure Nginx to process PHP files

Once I had completed setting up my LEMP stack, I needed to test if the installation was fully operational. Specifically, I wanted to confirm that Nginx could correctly process PHP files by handing them off to the PHP processor.

To do this, I decided to create a simple test PHP file. I opened my text editor and created a new file called `info.php` in the document root of my web directory:

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

In this file, I added a small block of PHP code that would output information about my server's PHP configuration when accessed. This would serve as a useful test to see if the PHP processor was working correctly:



```
ubuntu@ip-172-31-17-208: ~  
GNU nano 7.2 /var/www/projectLEMP/info.php *  
<?php  
phpinfo();  
?>
```

After saving the file, I accessed it via my web browser by navigating to the domain name or public IP address of my server, followed by `/info.php`. The URL looked something like this:

<http://35.87.151.65/info.php>

When I visited this page, it displayed a detailed PHP information page, which confirmed that Nginx was successfully processing PHP files through the FastCGI processor. This output included all the necessary details about the PHP version, loaded modules, and configuration settings. This step was crucial because it validated that the PHP processor was working as expected, ensuring that future PHP applications or scripts



would run without issues on my server. Having this confirmation gave me confidence that my LEMP stack was fully set up and ready for use.

Instances | EC2 | us-west-2

PHP 8.3.6 - phpinfo()

+


← → ↺

35.87.151.65/info.php

☆

📄 ⬇️ 👤 📄

PHP Version 8.3.6



System	Linux ip-172-31-17-208 6.8.0-1016-aws #17-Ubuntu SMP Mon Sep 2 13:48:07 UTC 2024 x86_64
Build Date	Sep 30 2024 15:17:17
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/fpm
Loaded Configuration File	/etc/php/8.3/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/fpm/conf.d
Additional .ini files parsed	/etc/php/8.3/fpm/conf.d/10-mysqlnd.ini, /etc/php/8.3/fpm/conf.d/10-opcache.ini, /etc/php/8.3/fpm/conf.d/10-pdo.ini, /etc/php/8.3/fpm/conf.d/20-calendar.ini, /etc/php/8.3/fpm/conf.d/20-ctype.ini, /etc/php/8.3/fpm/conf.d/20-exif.ini, /etc/php/8.3/fpm/conf.d/20-fileinfo.ini, /etc/php/8.3/fpm/conf.d/20-ftp.ini, /etc/php/8.3/fpm/conf.d/20-gettext.ini, /etc/php/8.3/fpm/conf.d/20-iconv.ini, /etc/php/8.3/fpm/conf.d/20-mysqli.ini, /etc/php/8.3/fpm/conf.d/20-pdo_mysql.ini, /etc/php/8.3/fpm/conf.d/20-phar.ini, /etc/php/8.3/fpm/conf.d/20-posix.ini, /etc/php/8.3/fpm/conf.d/20-readline.ini, /etc/php/8.3/fpm/conf.d/20-shmop.ini, /etc/php/8.3/fpm/conf.d/20-sockets.ini, /etc/php/8.3/fpm/conf.d/20-sysvmsg.ini, /etc/php/8.3/fpm/conf.d/20-sysvsem.ini, /etc/php/8.3/fpm/conf.d/20-sysvshm.ini, /etc/php/8.3/fpm/conf.d/20-tokenizer.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	AP420230831.NTS
PHP Extension Build	API20230831.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	disabled

# STEP 6: RETRIEVING DATA FROM MYSQL DATABASE WITH PHP

To retrieve data from MySQL using PHP, I followed a structured process. I first created a test database and configured it for access through an Nginx website. This would allow the site to query data from the database and display it effectively.

## ##Connect to MySQL with root privileges and create database

I started by connecting to the MySQL console with root privileges. This provided me the necessary administrative access to create and manage databases and users.

- `$ sudo mysql`

Once in the MySQL console, I created a new database named `example_database`. You can replace this name with a different one if needed, but this was used for demonstration purposes.

- `mysql> CREATE DATABASE `mac_database`;`

## ##Create user and assign privileges

With the database set up, the next task was to create a new user who would have full privileges over this newly created database. I opted for a user named `example_user`. Since MySQL requires authentication, I used the `mysql_native_password` method for this user. For demonstration purposes, I assigned a password of `'Password.1'`, though it's recommended to choose a stronger, more secure password for real-world applications.

- `mysql> CREATE USER 'macbeth'@'%' IDENTIFIED WITH mysql_native_password BY 'Password.1';`

The final step was to grant this new user full permissions to manage the `example_database`. This ensures the user can perform all actions such as querying, inserting, updating, and deleting data in the database.

- `mysql> GRANT ALL ON mac_database.* TO 'macbeth'@'%';`

```
mysql> CREATE DATABASE `mac_database`;
Query OK, 1 row affected (0.05 sec)

mysql> CREATE USER 'macbeth'@'%' IDENTIFIED WITH mysql_native_password BY 'Password.1';
'>
'> GRANT ALL ON mac_database.* TO 'macbeth'@'%';
'> exit
'> ^C
mysql> CREATE USER 'macbeth'@'%' IDENTIFIED WITH mysql_native_password BY 'Password.1';
Query OK, 0 rows affected (0.05 sec)

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

mysql> exit
Bye
```

After granting “macbeth” full privileges over the mac\_database, I exited the MySQL shell to test the new user credentials. This step ensures that the user has the correct permissions to access and manage the database.

To exit the MySQL shell, I ran the following command:

- *mysql> exit*

Next, I logged back into the MySQL console using the newly created example\_user credentials. The -p flag in the command prompted me to enter the password I had set for this user.

- *\$ mysql -u macbeth -p*

After entering the MySQL console, I verified that “macbeth” had access to the database by running the SHOW DATABASES; command. This command lists all the databases that the current user can interact with.

```
mysql> SHOW DATABASES;
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mac_database |
| performance_schema |
+-----+
3 rows in set (0.02 sec)

mysql>
```

The expected output confirmed that “macbeth” had access to mac\_database:

With access confirmed, I proceeded to create a table inside the example\_database. The table was named todo\_list, and it would store to-do items, each with a unique item\_id and a content field for the description of the to-do item. The following SQL command was used to create the table:

```
CREATE TABLE mac_database.todo_list (

    item_id INT AUTO_INCREMENT,

    content VARCHAR(255),

    PRIMARY KEY (item_id)

);
```

This setup established a basic structure for storing data, ensuring that the “macbeth” could manage the table and retrieve data using PHP later on. After creating the todo\_list table inside the mac\_database, I proceeded to insert some data into the table to test that everything was working correctly. Each row inserted represents a different to-do item, with the content field storing the item's description.

I used the following SQL command to insert the first row:

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

To add more test data, I repeated this command multiple times, changing the content each time. This allowed me to have multiple entries to test the functionality further.

For example, I ran similar commands to add more items:

```
- mysql> INSERT INTO mac_database.todo_list (content) VALUES ("My second important item");
- mysql> INSERT INTO mac_database.todo_list (content) VALUES ("My third important item");
- mysql> INSERT INTO mac_database.todo_list (content) VALUES ("and this one more thing");
```

After inserting the data, I wanted to confirm that everything was saved correctly. I used the SELECT command to retrieve and display all the entries from the todo\_list table:

- `mysql> SELECT * FROM mac_database.todo_list;`

```
mysql> INSERT INTO mac_database.todo_list (content) VALUES ("My first important item");
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO mac_database.todo_list (content) VALUES ("My second important item");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO mac_database.todo_list (content) VALUES ("My third important item");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO mac_database.todo_list (content) VALUES ("and this one more thing");
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM mac_database.todo_list;
+-----+-----+
| 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 (0.00 sec)

mysql> |
```

Seeing this output confirmed that the data had been successfully saved to the database.

After verifying the data, I exited the MySQL console with the following command:

- `mysql> exit`

At this point, I had successfully created a table, inserted data into it, and retrieved the data, confirming that everything was functioning correctly.

After exiting mysql, I created a PHP file using the following code:

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

where I inserted the PHP script to connect to MySQL and query content from the todo\_list table:

```
ubuntu@ip-172-31-17-208: ~
GNU nano 7.2 /var/www/projectLEMP/todo_list.php
<?php
$user = "macbeth";
$password = "Password.1";
$database = "mac_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();
}
?>
```

Afterwards, I saved and exited the nano page.

Then I went back to my web-browser and accessed the PHP page to see if the changes had taken place. The results indicate that I had successfully configured PHP to retrieve information from MySQL.



## TODO

1. My first important item
2. My second important item
3. My third important item
4. and this one more thing

## WHAT I LEARNED, CHALLENGES I FACED AND HOW I OVERCAME THEM:

Throughout the process of implementing the LEMP stack, I gained significant insight into the critical components of Linux, Nginx, MySQL, and PHP, which form the foundation for creating a robust web server environment. I learned how to install each of these components, ensuring they work seamlessly together to deliver web applications efficiently. By starting with setting up an Ubuntu server, I developed my Linux command-line skills, followed by configuring Nginx as a web server to handle requests. Understanding how to secure the MySQL database, manage databases, and connect them to PHP was a valuable experience that reinforced my knowledge of database management and server-side scripting. The hands-on approach enabled me to troubleshoot configuration issues and deepen my understanding of server environments, which has been instrumental in developing a comprehensive view of web hosting architecture.

However, the process was not without its challenges. One major difficulty I faced was troubleshooting server errors, particularly when services failed to run after installation. Configuring the firewall to allow HTTP/HTTPS traffic and resolving issues with PHP and MySQL connectivity required detailed research and trial and error. Despite these challenges, I employed self-study and problem-solving strategies, such as consulting online documentation and forums, to overcome them. This experience not only enhanced my technical skills but also improved my ability to work independently and resolve technical issues, thereby reinforcing my confidence in implementing complex server environments.