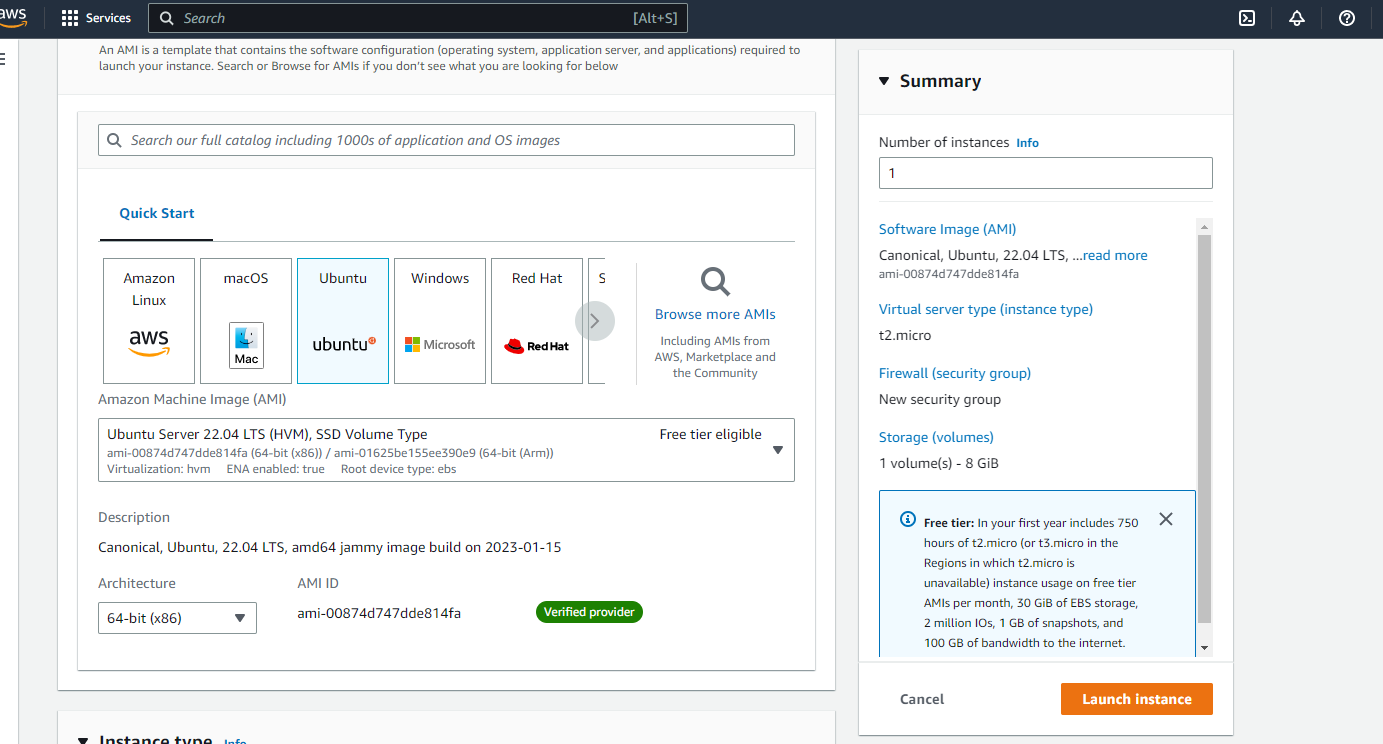
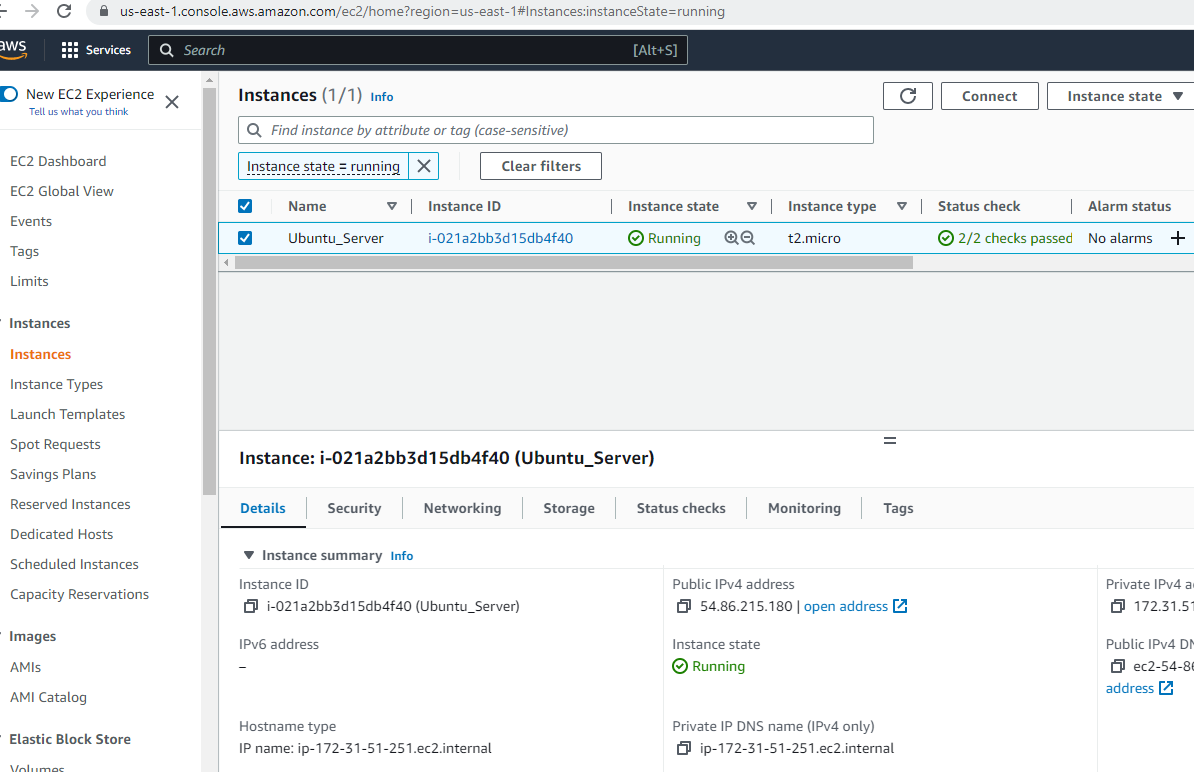
# **WEB STACK IMPLEMENTATION (LAMP STACK) IN AWS**

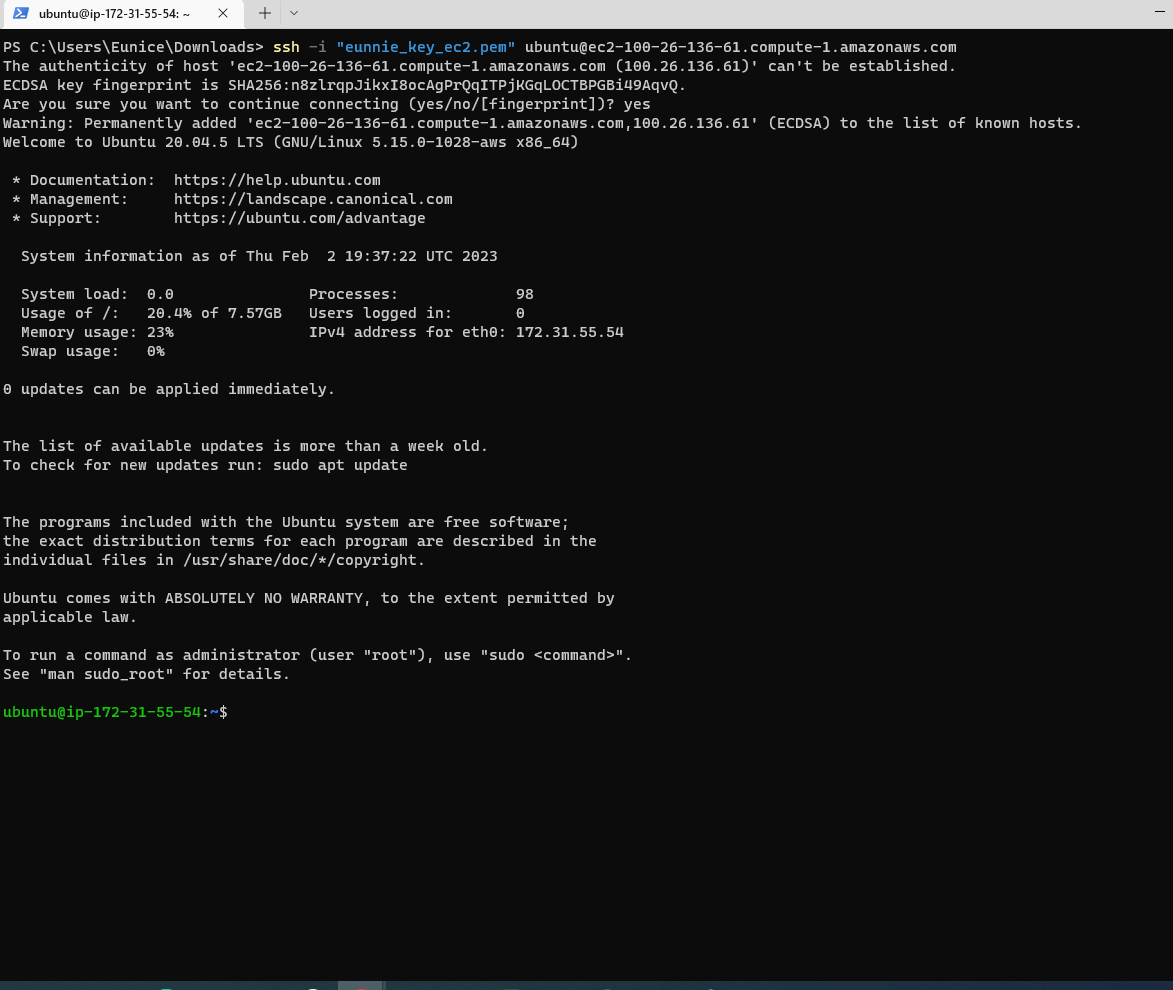
1. Create AWS Account
2. Login as Root User
3. Launched an EC2 instance



1. Selected Ubuntu Server free tier eligible
2. Completed all configurations for setting up instance (Enabled public IP, security group, and key pair) and it is launched

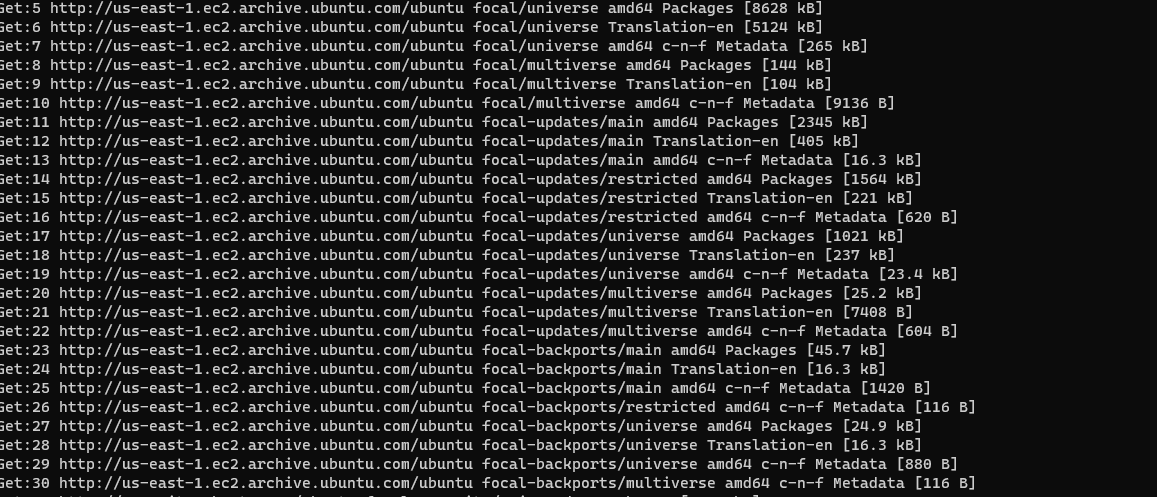


1. SSH into instance with Windows Terminal
2. Changed directory: cd Downloads to location of key-pair
3. I was able to connect to my instance

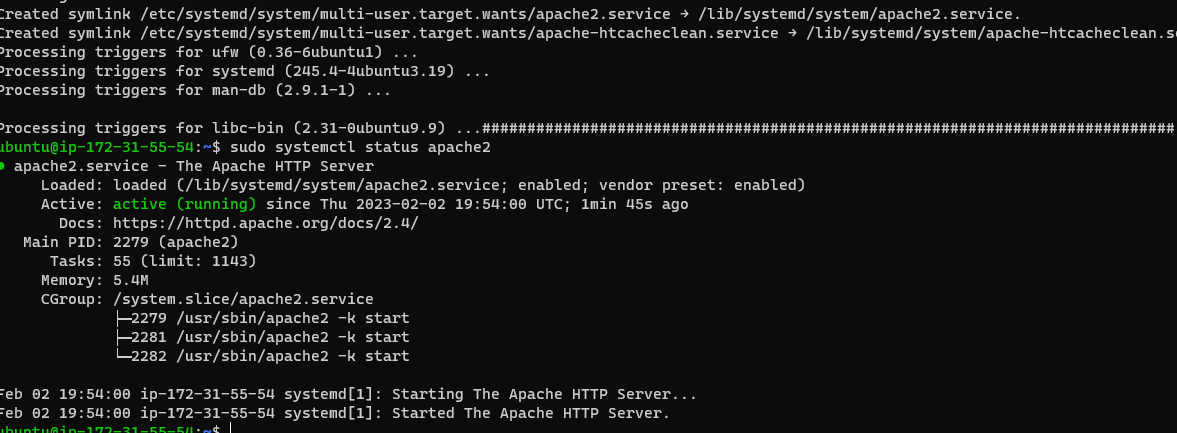


# **STEP 1 — INSTALLING APACHE AND UPDATING THE FIREWALL**

1. Install Apache using Ubuntu’s package manager [***‘apt’***](https://en.wikipedia.org/wiki/APT_(software)):
2. Run sudo apt update to update a list of packages in package manager



1. #run apache2 package installation using sudo apt install apache2
2. Verify that apache2 is running as a Service in our OS use command: sudo systemctl status apache2



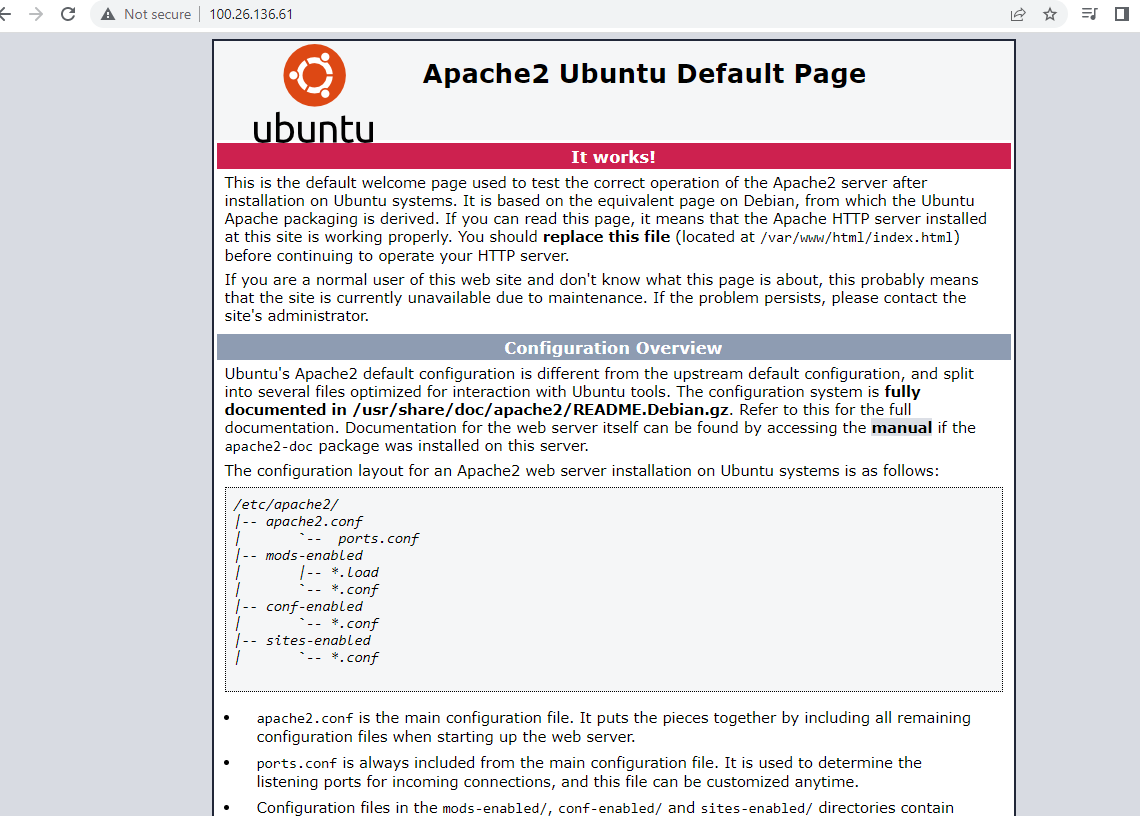
1. Green light will indicate Apache2 is running
2. Open TCP port 80 using command curl http://localhost:80

or

curl <http://127.0.0.1:80>

1. This command would output the Apache2 payload indicating that it is accessible locally in the Ubuntu shell.
2. Test that Apache HTTP server can respond to requests from the Internet. Open a browser and type the public IP of the Ubuntu instance: http://100.26.136.61:80

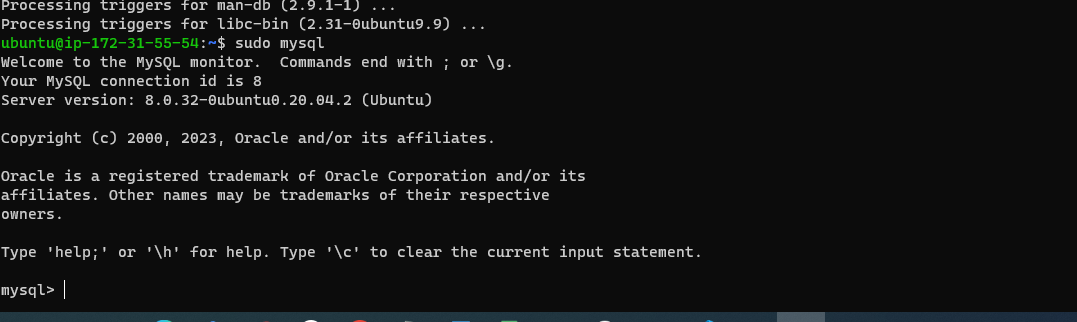
( You can also retrieve your public IP address by using this command: curl -s http://169.254.169.254/latest/meta-data/public-ipv4



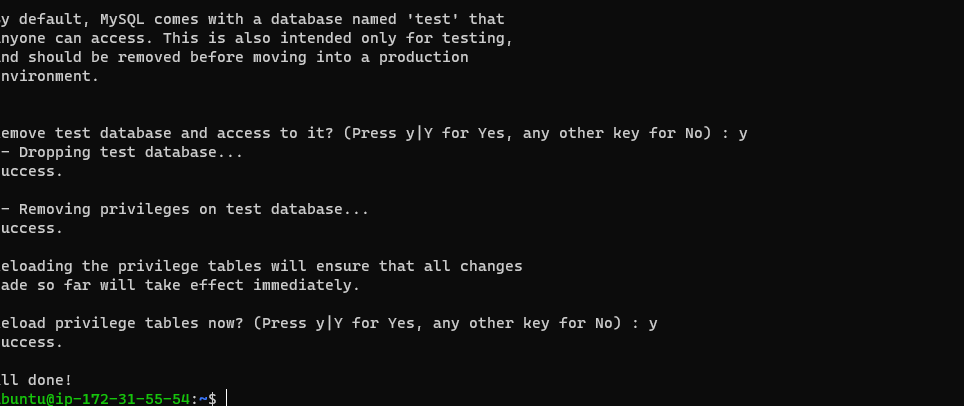
# **STEP 2 — INSTALLING MYSQL**

We 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).

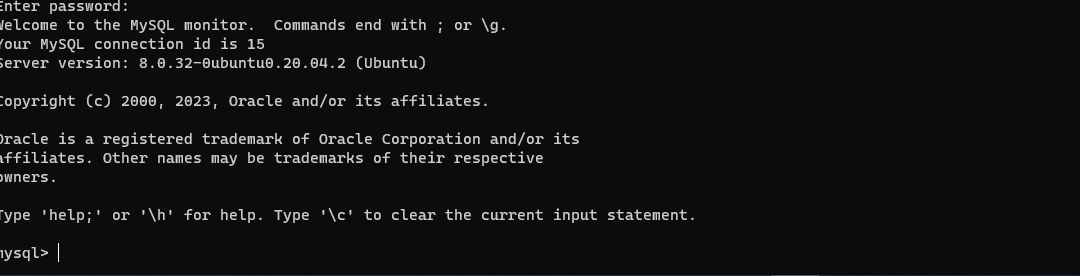
1. use ‘apt’ to acquire and install this software: $ sudo apt install mysql-server
2. Confirm installation with y
3. When installation is finished, log in to the MySql console with command: $ sudo mysql



1. Run security script that comes pre-installed with MySql: ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'PassWord.1';
2. Then exit MySQL shell with: mysql> exit
3. Start the interactive script with command $ sudo mysql\_secure\_installation and continue with instructions changed password to



1. After finishing, test if able to log in to the MySQL console with command : sudo mysql -p
2. MySQL server now installed



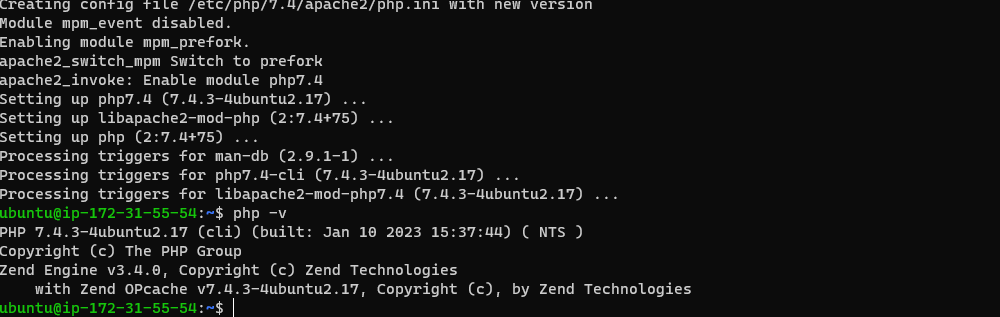
1. To exit use command: exit

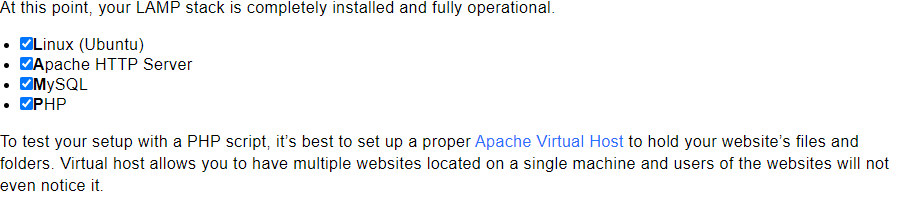
# **STEP 3 — INSTALLING PHP**

[PHP](https://www.php.net/) is the component of our setup that will process code to display dynamic content to the end user.In addition to the php package, you’ll need php-mysql

, a PHP module that allows PHP to communicate with MySQL-based databases. You’ll also need libapache2-mod-php to enable Apache to handle PHP files.

1. To install these 3 packages at once run: **sudo apt install php libapache2-mod-php php-mysql**
2. Once the installation is finished, you can run the following command to confirm your PHP version: **php -v**





# **STEP 4 — CREATING A VIRTUAL HOST FOR YOUR WEBSITE USING APACHE**

1. Create the directory for projectlamp using **‘mkdir’** command as follows: sudo mkdir /var/www/projectlamp
2. Next, assign ownership of the directory with your current system user: sudo chown -R $USER:$USER /var/www/projectlamp
3. Next, create and open a new configuration file in Apache’s sites-available directory using your preferred command-line editor. Here, we’ll be using vi or vim (They are the same by the way):sudo vi /etc/apache2/sites-available/projectlamp.conf
4. This creates a new blank file:

<VirtualHost \*:80>

ServerName projectlamp

ServerAlias www.projectlamp

ServerAdmin webmaster@localhost

DocumentRoot /var/www/projectlamp

ErrorLog ${APACHE\_LOG\_DIR}/error.log

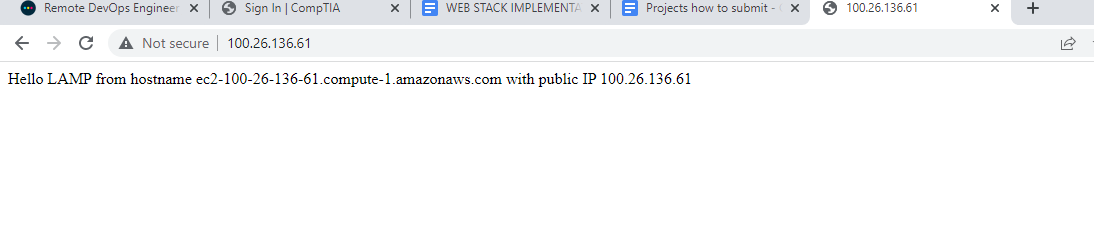
CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

1. Save and close the file
2. You can use the ls command to show the new file in the sites-available directory: sudo ls /etc/apache2/sites-available
3. You can now use ***a2ensite*** command to enable the new virtual host:sudo a2ensite projectlamp
4. You might want to disable the default website that comes installed with Apache. This is required if you’re not using a custom domain name, because in this case Apache’s default configuration would overwrite your virtual host. To disable Apache’s default website use ***a2dissite*** command , type:sudo a2dissite 000-default
5. To make sure your configuration file doesn’t contain syntax errors, run:sudo apache2ctl configtest
6. Finally, reload Apache so these changes take effect:sudo systemctl reload apache2
7. Your new website is now active, but the web root **/var/www/projectlamp** is still empty. Create an index.html file in that location so that we can test that the virtual host works as expected:

sudo echo 'Hello LAMP 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/projectlamp/index.html

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



1. If you see the text from ***‘echo’*** command you wrote to index.html file, then it means your Apache virtual host is working as expected.

In the output you will see your server’s public hostname (DNS name) and public IP address. You can also access your website in your browser by public DNS name, not only by IP – try it out, the result must be the same (port is optional)

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

# **STEP 5 — ENABLE PHP ON THE WEBSITE**

1. With the default DirectoryIndex settings on Apache, a file named index.html will always take precedence over an index.php file. This is useful for setting up maintenance pages in PHP applications, by creating a temporary index.html file containing an informative message to visitors. Because this page will take precedence over the index.php page, it will then become the landing page for the application. Once maintenance is over, the index.html is renamed or removed from the document root, bringing back the regular application page.
2. To change this behavior, you need to edit the **/etc/apache2/mods-enabled/dir.conf** file and change the order in which the **index.php** file is listed within the **DirectoryIndex** directive: sudo vim /etc/apache2/mods-enabled/dir.conf

<IfModule mod\_dir.c>

#Change this:

#DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm

#To this:

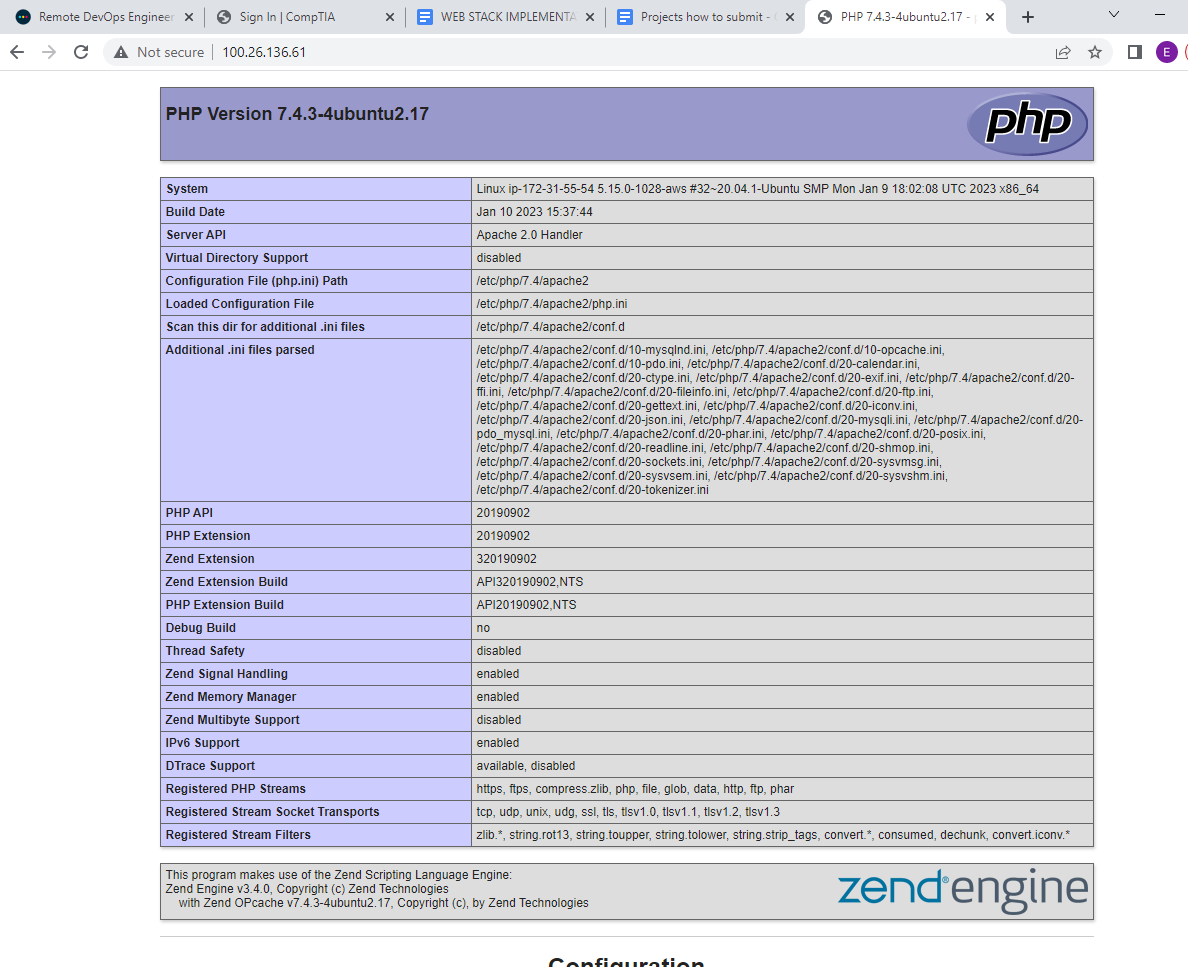
DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm

</IfModule>

1. After saving and closing the file, you will need to reload Apache so the changes take effect: **sudo systemctl reload apache2**
2. This will open a blank file. Add the following text, which is valid PHP code, inside the file:



1. When I finished, I saved and closed the file, refreshed and saw this final image:



This page provides information about your server from the perspective of PHP. It is useful for debugging and to ensure that your settings are being applied correctly.

If you can see this page in your browser, then your PHP installation is working as expected.

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 do so: sudo rm /var/www/projectlamp/index.php

You can always recreate this page if you need to access the information again later.