

Introduction

The goal of this lab was to manually deploy a WordPress instance on a Linux-based AWS EC2 server. WordPress is one of the most popular open-source content management systems, widely used for both personal and commercial websites.

Rather than using automated scripts or pre-configured images, this exercise required building the entire WordPress stack by hand. The purpose was to demonstrate a deep understanding of how web and database servers work together to support a dynamic PHP-based application like WordPress.

This included:

- Setting up the web server (Apache)
- Installing PHP with required extensions
- Installing and configuring MySQL
- Deploying the WordPress application files
- Troubleshooting and securing the environment
- Verifying successful deployment

By completing this lab, I gained practical experience with Linux system administration, web server configuration, PHP dependency management, and database integration (all critical skills for managing modern web services.)

Lab Environment

EC2 Instance: Ubuntu 24.04 LTS

Instance Type: t2.micro

Public IP: 3.94.163.204

Software Versions Installed:

- Apache 2.4
- PHP 8.3
- MySQL 8.0
- WordPress 6.x (latest release at time of install)

Tools Used:

- AWS CloudShell Terminal
- MySQL command line client
- nano text editor
- Web browser for validation

Procedure

1. Initial Server Setup

After launching the EC2 instance, the first step was to update the system and install necessary packages:

```
aws [Option+S]
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

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

System information as of Thu Jun 12 22:01:47 UTC 2025

System load: 0.14          Processes:            105
Usage of /: 25.4% of 6.7GB Users logged in:      0
Memory usage: 21%         IPv4 address for enX0: 172.31.26.96
Swap usage: 0k

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 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-172-31-26-96:~$ sudo apt update[
```

i-09b645894f4949cf9
PublicIPs: 3.94.163.204 PrivateIPs: 172.31.26.96

```
aws [Option+S]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1118 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1078 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [274 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [376 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [26.0 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [21.7 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [4788 B]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [592 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [39.2 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main Translation-en [8676 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7088 B]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [272 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [27.1 kB]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [16.5 kB]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [16.4 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1304 B]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:37 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:38 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [849 kB]
Get:39 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [187 kB]
Get:40 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.2 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [17.0 kB]
Get:42 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:43 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [17.7 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [3792 B]
Get:45 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:46 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [380 B]
Fetched 31.2 MB in 6s (5064 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
ubuntu@ip-172-31-26-96:~$ sudo apt install apache2[
```

i-09b645894f4949cf9
PublicIPs: 3.94.163.204 PrivateIPs: 172.31.26.96

This provided the basic web server, PHP runtime, and database server needed to support WordPress.

2. Configuring Apache

I edited the default Apache virtual host configuration:

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

Last login: Thu Jun 12 22:01:48 2025 from 18.206.107.27
ubuntu@ip-172-31-26-96:~$ sudo nano /etc/apache2/sites-available/000-default.conf
ubuntu@ip-172-31-26-96:~$ sudo a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
ubuntu@ip-172-31-26-96:~$ sudo apache2ctl configtest
Usage: /usr/sbin/apache2 [-D name] [-d directory] [-f file]
                        [-C 'directive'] [-c 'directive']
                        [-k start|restart|graceful|graceful-stop|stop]
                        [-V] [-V] [-h] [-l] [-L] [-t] [-T] [-S] [-X]

Options:
  -D name                : define a name for use in <IfDefine name> directives
  -d directory            : specify an alternate initial ServerRoot
  -f file                : specify an alternate ServerConfigFile
  -C 'directive'         : process directive before reading config files
  -c 'directive'         : process directive after reading config files
  -e level               : show startup errors of level (see LogLevel)
  -E file               : log startup errors to file
  -v                    : show version number
  -V                    : show compile settings
  -h                    : list available command line options (this page)
  -l                    : list compiled in modules
  -L                    : list available configuration directives
  -t -D DUMP_VHOSTS      : show parsed vhost settings
  -t -D DUMP_RUN_CFG     : show parsed run settings
  -S                    : a synonym for -t -D DUMP_VHOSTS -D DUMP_RUN_CFG
  -t -D DUMP_MODULES     : show all loaded modules
  -M                    : a synonym for -t -D DUMP_MODULES
  -t -D DUMP_INCLUDES    : show all included configuration files
  -t                    : run syntax check for config files
  -T                    : start without DocumentRoot(s) check
  -X                    : debug mode (only one worker, do not detach)

ubuntu@ip-172-31-26-96:~$ sudo apache2ctl configtest
AH00112: Warning: DocumentRoot [/var/www/wordpress] does not exist
Syntax OK
ubuntu@ip-172-31-26-96:~$ sudo systemctl restart apache2
ubuntu@ip-172-31-26-96:~$
```

i-09b645894f4949cf9
Public IP: 3.94.163.204 Private IP: 172.31.26.96

The document root was changed to /var/www/wordpress to serve the WordPress files. I also added the following directory configuration to allow WordPress to run correctly:

```
GNU nano 7.2 /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/wordpress
    <Directory /var/www/wordpress/>
        AllowOverride All
    </Directory>

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #include conf-available/serve-cgi-bin.conf
</VirtualHost>

File Name to Write: /etc/apache2/sites-available/000-default.conf
^C Help      M-D DOS Format  M-A Append    M-S Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^B Browse

i-09b645894f4949cf9
Public IP: 3.94.163.204 Private IP: 172.31.26.96
```

After saving the configuration, I restarted Apache.

3. Installing wordpress

WordPress was then downloaded directly from its official site using wget. After extracting the tarball, I copied all files from the extracted wordpress directory into the new document root using `cp -a wordpress/. /var/www/wordpress`. File and directory permissions were then hardened by recursively setting the appropriate ownership (`www-data:www-data`) and limiting directory access to 750 and file access to 640. This ensured that only the web server could write to these files, while still allowing them to be served properly.

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

Last login: Thu Jun 12 22:01:48 2025 from 18.206.107.27
ubuntu@ip-172-31-26-96:~$ sudo nano /etc/apache2/sites-available/000-default.conf
ubuntu@ip-172-31-26-96:~$ sudo a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
ubuntu@ip-172-31-26-96:~$ sudo apache2ctl configtest
Usage: /usr/sbin/apache2 [-D name] [-d directory] [-f file]
       [-C "directive"] [-c "directive"]
       [-k start|restart|graceful|graceful-stop|stop]
       [-v] [-V] [-h] [-l] [-t] [-T] [-S] [-X]

Options:
  -D name           : define a name for use in <IfDefine name> directives
  -d directory      : specify an alternate initial ServerRoot
  -f file           : specify an alternate ServerConfigFile
  -C "directive"    : process directive before reading config files
  -c "directive"    : process directive after reading config files
  -e level          : show startup errors of level (see LogLevel)
  -E file           : log startup errors to file
  -v               : show version number
  -V               : show compile settings
  -h               : list available command line options (this page)
  -l               : list compiled in modules
  -t               : list available configuration directives
  -t -D DUMP_VHOSTS : show parsed vhost settings
  -t -D DUMP_RUN_CFG : show parsed run settings
  -S               : a synonym for -t -D DUMP_VHOSTS -D DUMP_RUN_CFG
  -t -D DUMP_MODULES : show all loaded modules
  -M               : a synonym for -t -D DUMP_MODULES
  -t -D DUMP_INCLUDES : show all included configuration files
  -t               : run syntax check for config files
  -T               : start without DocumentRoot(s) check
  -X               : debug mode (only one worker, do not detach)
ubuntu@ip-172-31-26-96:~$ sudo apache2ctl configtest
AH00112: Warning: DocumentRoot [/var/www/wordpress] does not exist
Syntax OK
ubuntu@ip-172-31-26-96:~$ sudo systemctl restart apache2
ubuntu@ip-172-31-26-96:~$ cd /tmp
ubuntu@ip-172-31-26-96:/tmp$ wget https://wordpress.org/latest.tar.gz

i-09b645894f4949cf9
PublicIP: 3.94.163.204 PrivateIP: 172.31.26.96
```

4. Configuring MySQL

At this point, I turned my attention to setting up the MySQL database that WordPress would use. After logging in as root, I created a new database named wordpress and a user named wordpressuser, granting the user full privileges on the database. These changes were finalized with FLUSH PRIVILEGES. I then created a copy of the default WordPress configuration file (wp-config.php) and edited it to reflect the new database name, username, and password. The connection host was left as localhost, since the database was hosted on the same server.

5. Installing PHP Extensions

Upon attempting to access the site for the first time, I encountered a common issue: WordPress displayed an error stating that the required MySQL PHP extension was missing. This was resolved by explicitly installing php8.3-mysql, a package that provides the mysqli extension required by WordPress to communicate with MySQL. I also ensured that Apache was using the correct version of PHP by enabling the relevant module with a2enmod php8.3 and restarting the Apache and MySQL services once again.

With all services correctly configured and running, I opened the server's public IP in a web browser. The WordPress installation wizard appeared, confirming that the application could connect to the database and load properly. I followed the prompts to name the site, set an administrative username and password, and complete the setup. Once logged in to the dashboard, I created a test blog post titled *"NETV 379 Lab 1 Complete!"* to verify that content creation and publishing were functioning.

Several troubleshooting steps were necessary throughout the process. The initial error regarding the missing MySQL extension required research and package installation. I also had to confirm that MySQL was running, that the database and user were correctly set up, and that all file paths in the Apache configuration were pointing to the correct locations. Each problem required a methodical approach, and solving them helped reinforce best practices in Linux system administration.

The result was a fully functional WordPress instance accessible from the browser, hosted entirely on a manually configured EC2 server. This lab deepened my knowledge of how dynamic web

applications are structured and served. It demonstrated the importance of proper file permissions, the role of Apache directives, and the interdependency between application, web, and database layers. It also highlighted how critical PHP extensions and service restarts can be when things go wrong.





