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)

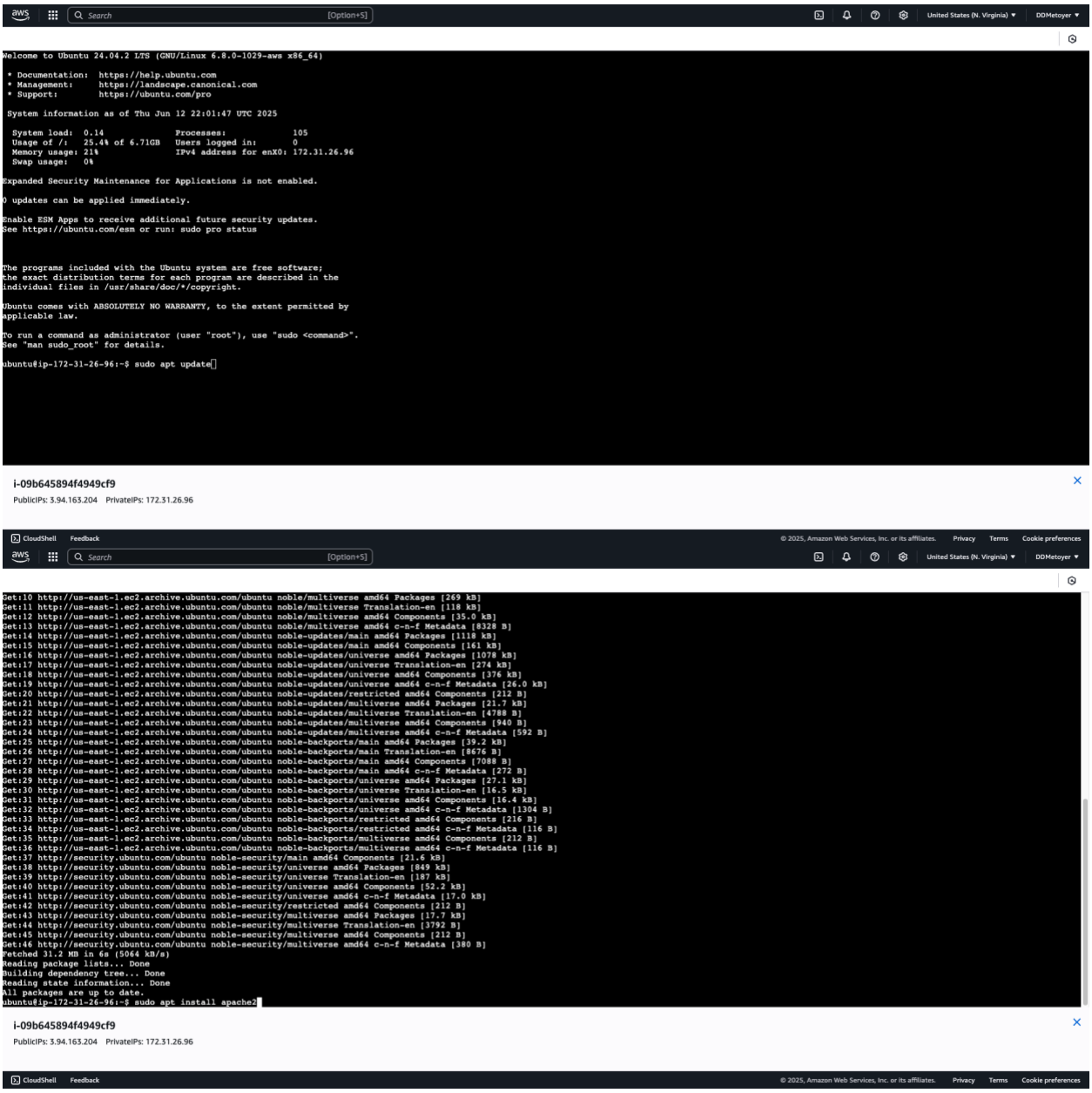
Tools Used:

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

Procedure

1. Initial Server Setup

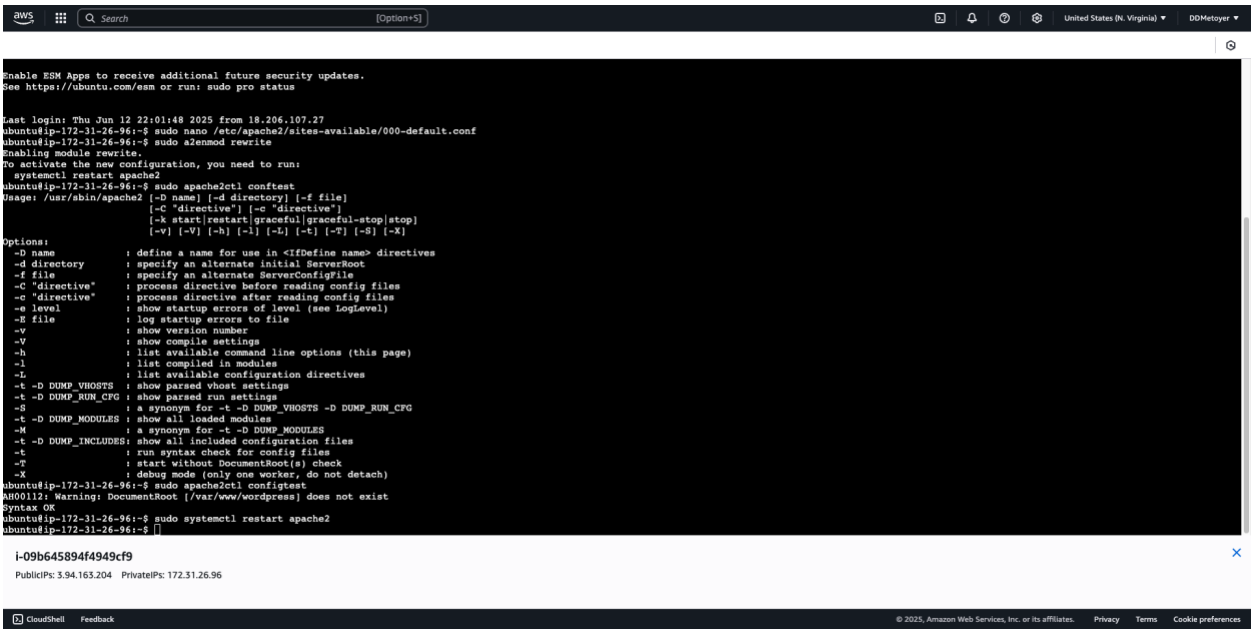
After launching the EC2 instance, the first step was to update the system and install necessary packages:



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:

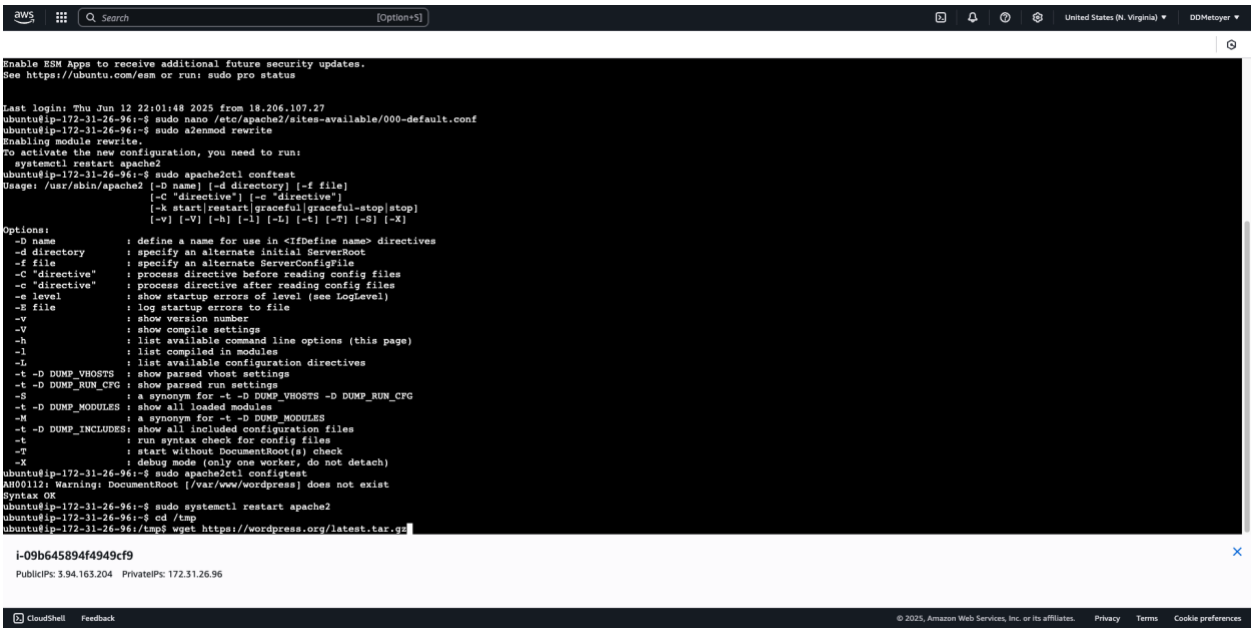


The document root was changed to /var/www/wordpress to serve the WordPress files. I also added the following directory configuration to allow WordPress to run correctly:

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.



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

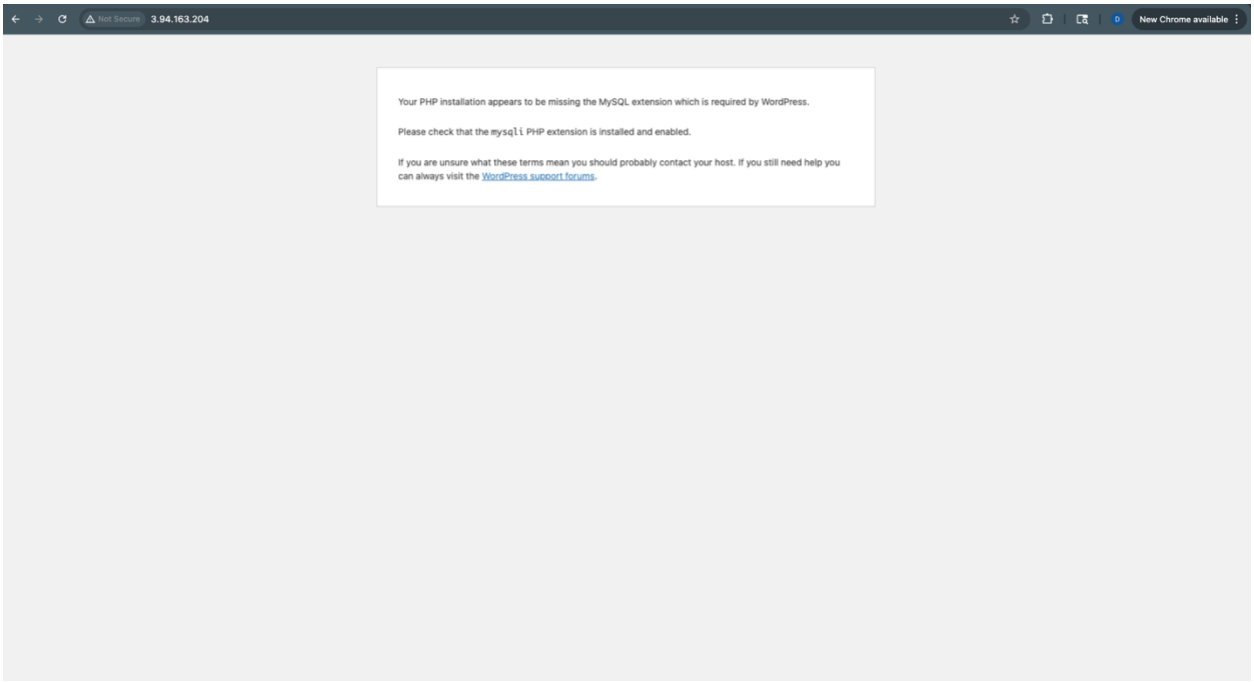
Upon attempting to access the site for the first time, I encountered a common issue: WordPress displayed an error stating that the required MySQL PHP extension was missing. This was resolved by explicitly installing php8.3-mysql, a package that provides the mysqli extension required by WordPress to communicate 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.

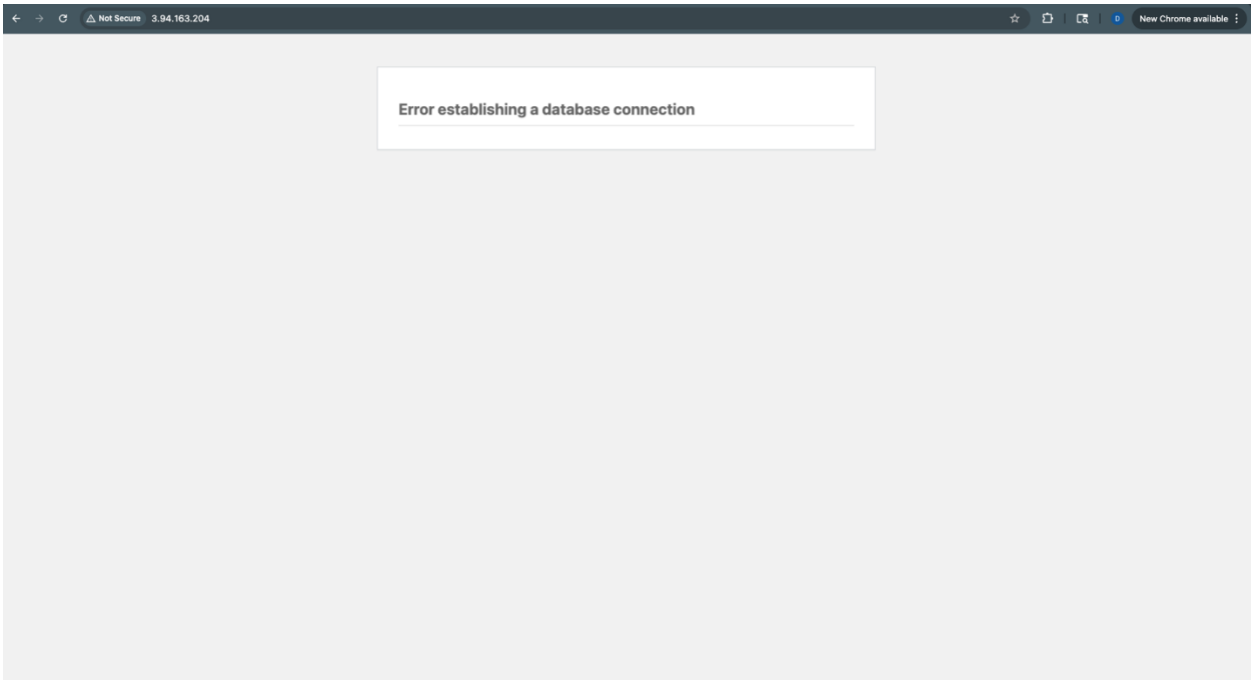
With all services correctly configured and running, I opened the server's public IP in a web browser. The WordPress installation wizard appeared, confirming that the application could connect to the database and load properly. I followed the prompts to name the site, set an administrative username and 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.

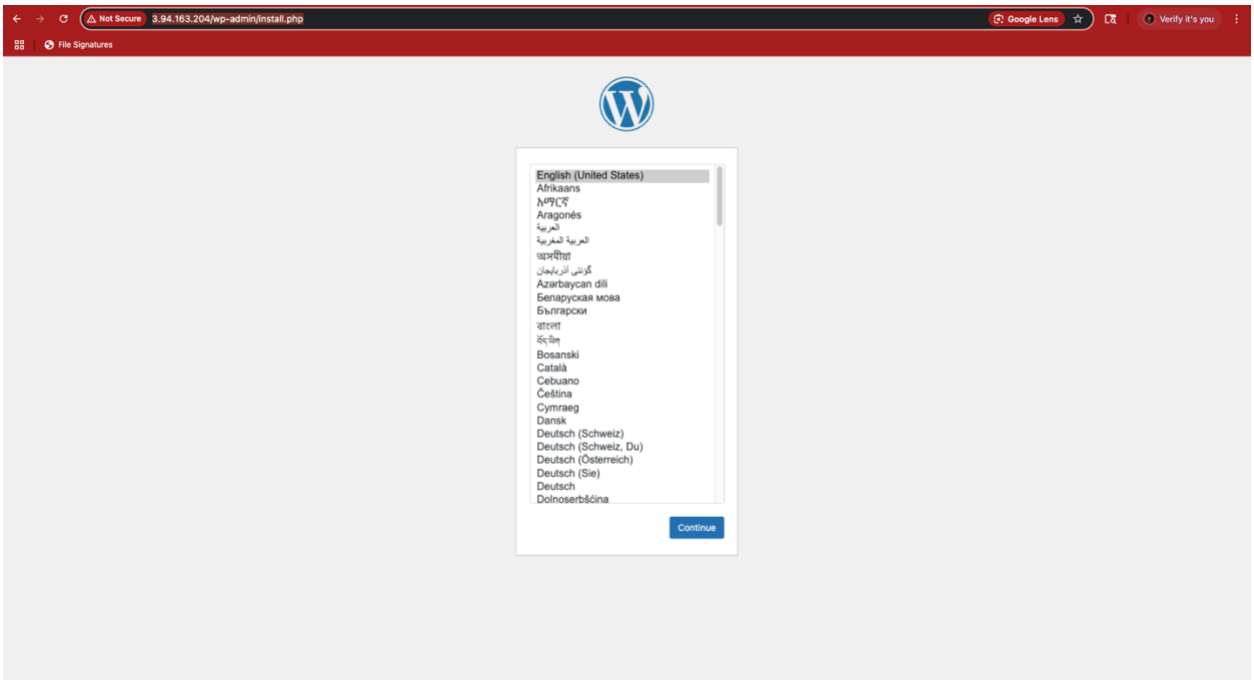
Several troubleshooting steps were necessary throughout the process. The initial error regarding the missing MySQL extension required research and package installation. I also had to confirm that MySQL was running, that the database and user were correctly set up, and that all file paths in the Apache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

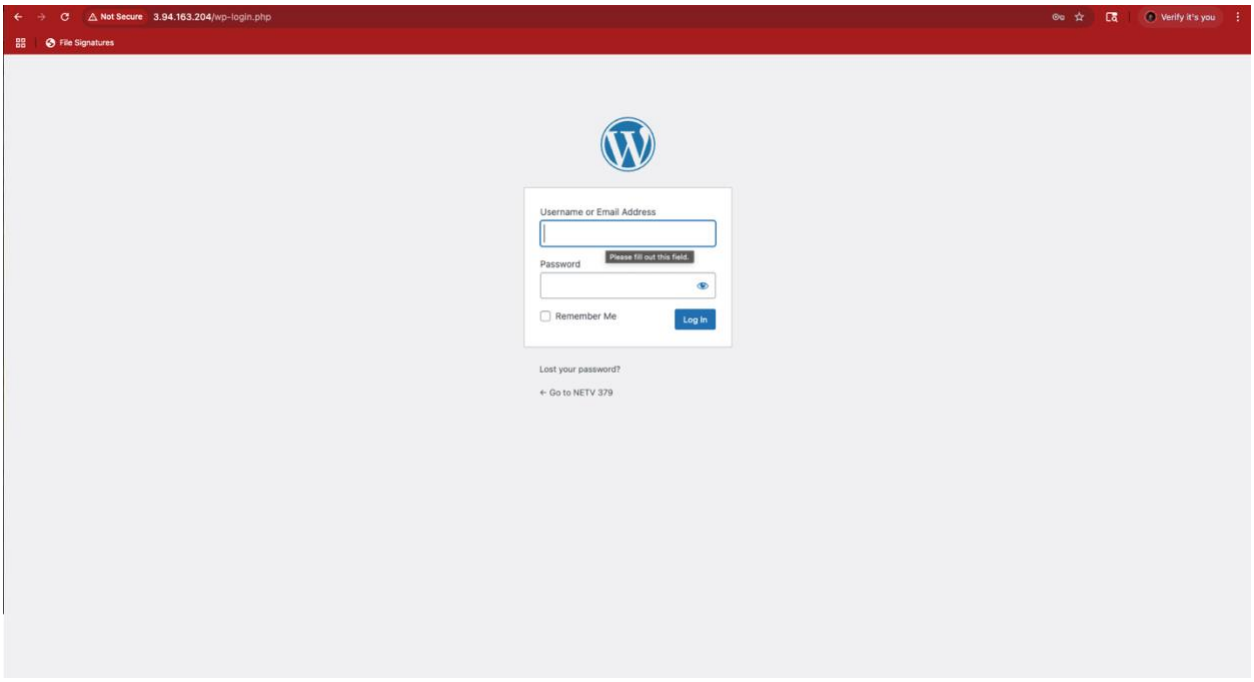
The result was a fully functional WordPress instance accessible from the browser, hosted entirely on a manually configured EC2 server. This lab deepened my knowledge of how dynamic web

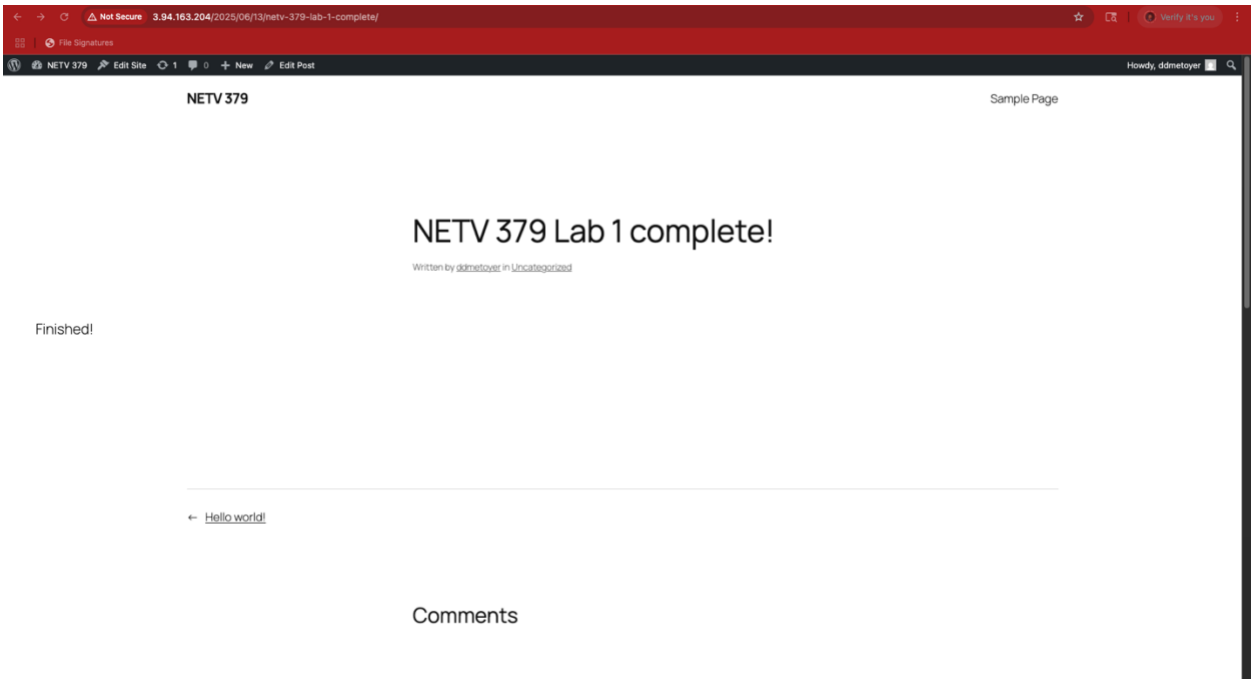
applications are structured and served. It demonstrated the importance of proper file permissions, the role of Apache directives, and the interdependency between application, web, and database layers. It also highlighted how critical PHP extensions and service restarts can be when things go wrong.






Not Secure
3.94.163.204/wp-admin/install.php
Google Lens
Verify it's you
File Signatures
English (United States)
Afrikaans
አማርኛ
Aragonés
العربية
العربية المغربية
অসমীয়া
گؤنئی آذربایجان
Azərbaycan dili
Беларуская мова
Български
বাংলা
বিষ্ণুপ্রিয়া
Bosanski
Català
Cebuano
Čeština
Cymraeg
Dansk
Deutsch (Schweiz)
Deutsch (Schweiz, Du)
Deutsch (Österreich)
Deutsch (Sie)
Deutsch
Dolnoserbšćina
Continue


Not Secure
3.94.163.204/wp-login.php
Verify it's you
File Signatures
Username or Email Address
Please fill out this field.
Password
Remember Me
Log In
Lost your password?
← Go to NETV 379


Not Secure 3.94.163.204/2025/06/13/netv-379-lab-1-complete/
File Signatures
NETV 379 Edit Site 1 0 New Edit Post
Howdy, ddmetoyer
NETV 379
Sample Page
NETV 379 Lab 1 complete!
Written by ddmetoyer in Uncategorized
Finished!
← Hello world!
Comments