

Bonus Assignment: LAMP and WordPress with Docker

OPS 245: Open Systems Server

Section: NBB

Professor Hans Heim

Shib Prosad Roy

April 17, 2025

TABLE OF CONTENTS

Introduction.....	2
Why Was This Bonus Assignment Created?.....	2
Comparative Architectural Diagram	3
Prerequisites	3
Getting Started.....	4
1. Clone the Repository	4
2. Create or Edit the .env File	4
3. Start the Services	4
4. Access WordPress Setup in Browser.....	4
5. Screenshots.....	5
Verification: Exec into Containers.....	8
1. Accessing MariaDB and Running SQL Commands	8
2. Exploring the WordPress Container.....	9
Cleanup: Teardown	9
1. Stop the Containers.....	10
2. Stop and Remove Everything (including volumes and data)	10

Introduction

This bonus assignment demonstrates that a full-stack WordPress environment can be quickly and cleanly deployed using Docker without the need to manually install and configure Apache, PHP, or MariaDB on a traditional VM.

By reworking Assignment 2 in this way, I wanted to show how containerization enables fast prototyping and repeatable deployments, ideal for testing or local development scenarios. Rather than spinning up an entire VM with its own resources, Docker allows us to run lightweight, isolated services that simulate real-world infrastructure.

Using Docker Compose spec in one single file; we are able to:

- Stand up WordPress and MariaDB as independent but networked services.
- Manage credentials and persistent storage through environment variables and volumes.
- Mimic how databases are often run as standalone services — whether hosted externally or containerized alongside applications.

Why Was This Bonus Assignment Created?

The approach taken in this assignment aligns with modern DevOps workflows and reinforces the idea that system services (like databases) don't always need to live on the same host as the application. They can be containerized, orchestrated, and scaled independently; which is a powerful concept for both system administrators and developers.

Ultimately, this bonus assignment showcases the practicality and flexibility of using Docker to replicate and even improve upon traditional LAMP stack setups, while preserving the educational objectives of OPS245 Assignment 2.

Comparative Architectural Diagram

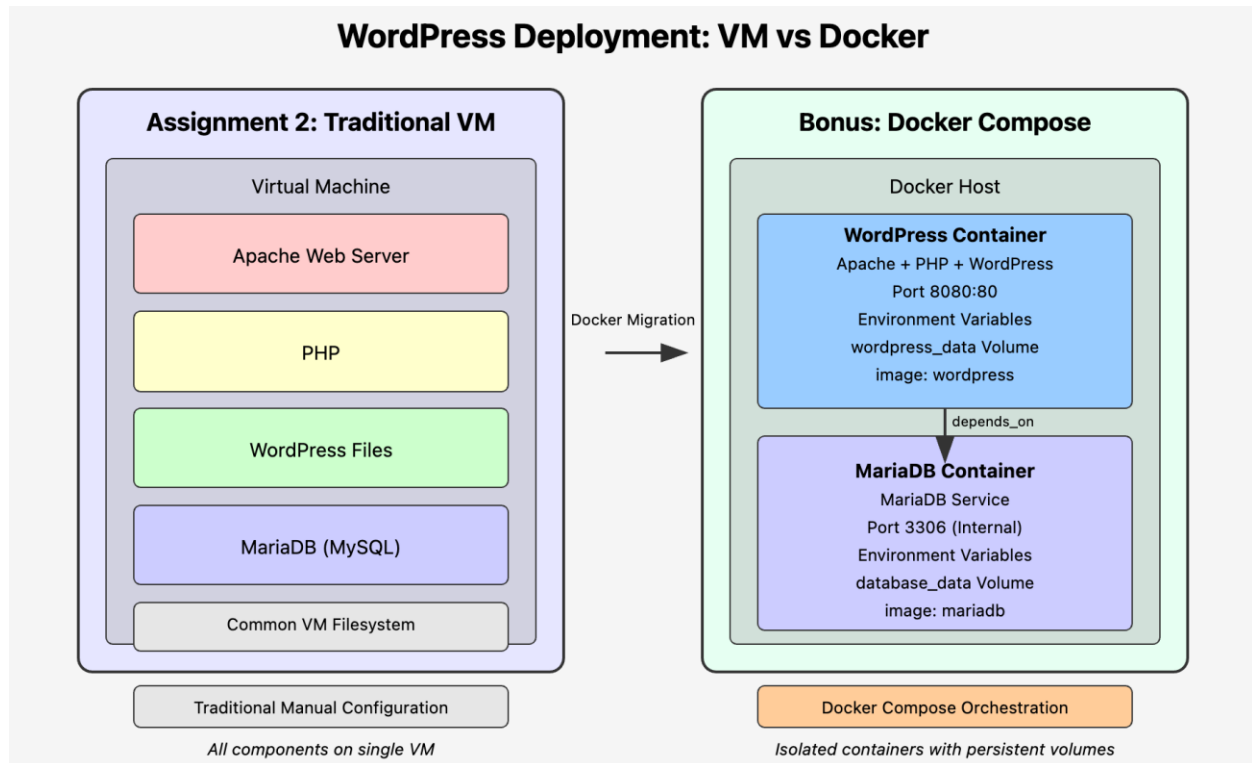


Fig 1: Comparison of Architectural Diagram of Both Implementation

Prerequisites

To complete or run this bonus assignment project, you should have the following:

Technical Prerequisites

- Docker Engine installed and running,
 - Install Docker: either Docker Desktop or engine via CLI, please read more about installation options here: <https://docs.docker.com/engine/install/ubuntu/#installation-methods>
- Docker Compose (usually included with Docker Desktop or Docker CLI v2+).
 - Basic understanding of docker compose and container technology.
- Basic knowledge of terminal usage.
 - Navigating directories, running commands, editing files (VS Code, nano, vim, etc.) and git operations.

Getting Started

Follow these steps to run the WordPress + MariaDB environment from this repository:

1. Clone the Repository

```
git clone https://github.com/shibroy98/ops245-bonus-docker.git  
cd ops245-bonus-docker
```

2. Create or Edit the .env File

Make sure the .env file exists in the root of the repo with the following content (already provided in the repo, but can be adjusted):

```
MYSQL_ROOT_PASSWORD=rootpass  
MYSQL_DATABASE=myblog  
MYSQL_USER=sproy  
MYSQL_PASSWORD=sproy
```

You can customize these variables as needed.

3. Start the Services

Use Docker Compose to launch WordPress and MariaDB:

```
docker compose up -d
```

This will:

- Pull WordPress and MariaDB images (Only for the first time).
- Set up a local Docker network.
- Persist database and WordPress files in volumes.
- Expose WordPress at <http://localhost:8080>.

4. Access WordPress Setup in Browser

Open your browser and go to:

<http://localhost:8080>

Then follow the WordPress installation wizard:

- Use the database name, user, and password from the .env file
- Set your site title, admin username/password, and email

5. Screenshots

After the container finishes spinning up, the following screens can be viewed:

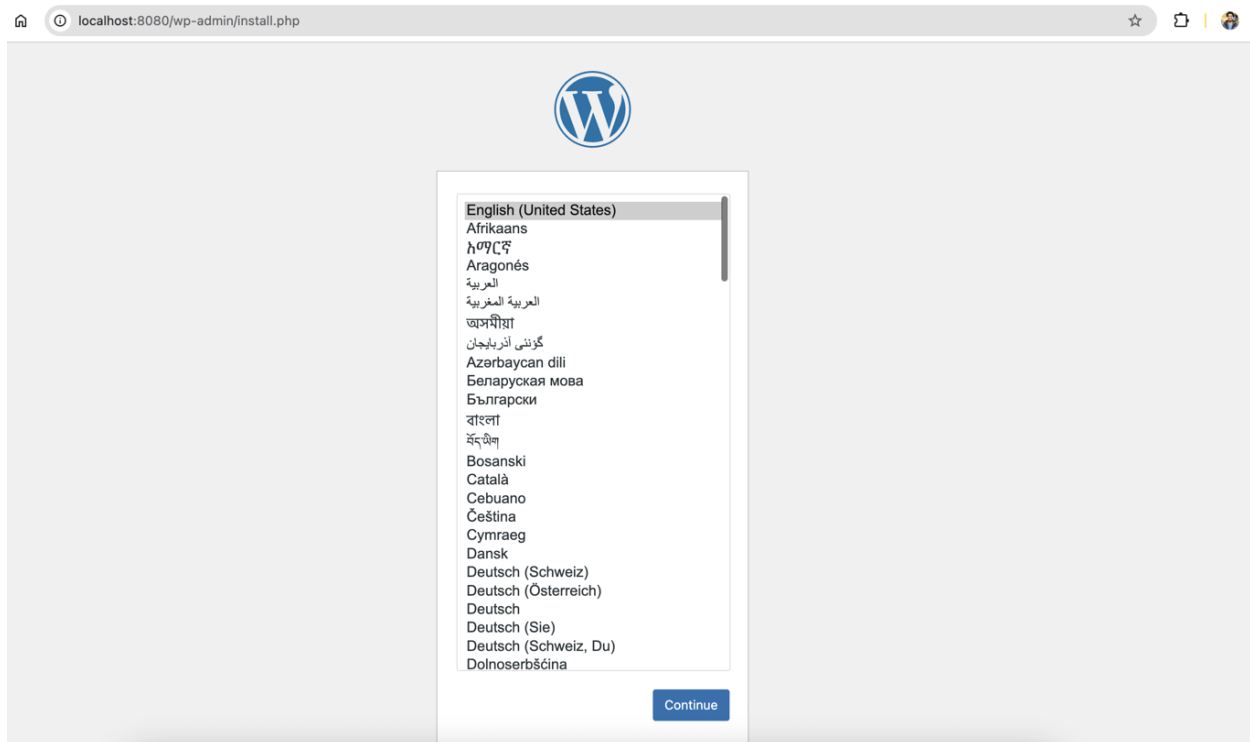
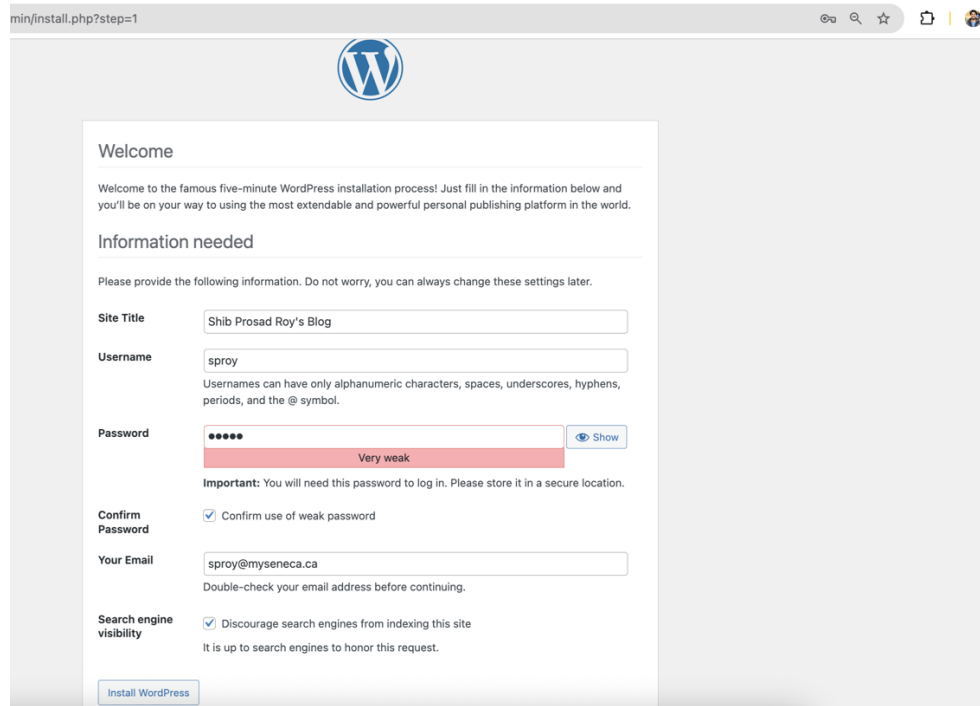


Fig 2: WordPress Installation Page Language Selection on First Access to localhost:8080



min/install.php?step=1

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title: Shib Prosad Roy's Blog

Username: sproy

Passwords can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password: ***** (Very weak)

Important: You will need this password to log in. Please store it in a secure location.

Confirm Password: ☒ Confirm use of weak password

Your Email: sproy@myseneca.ca

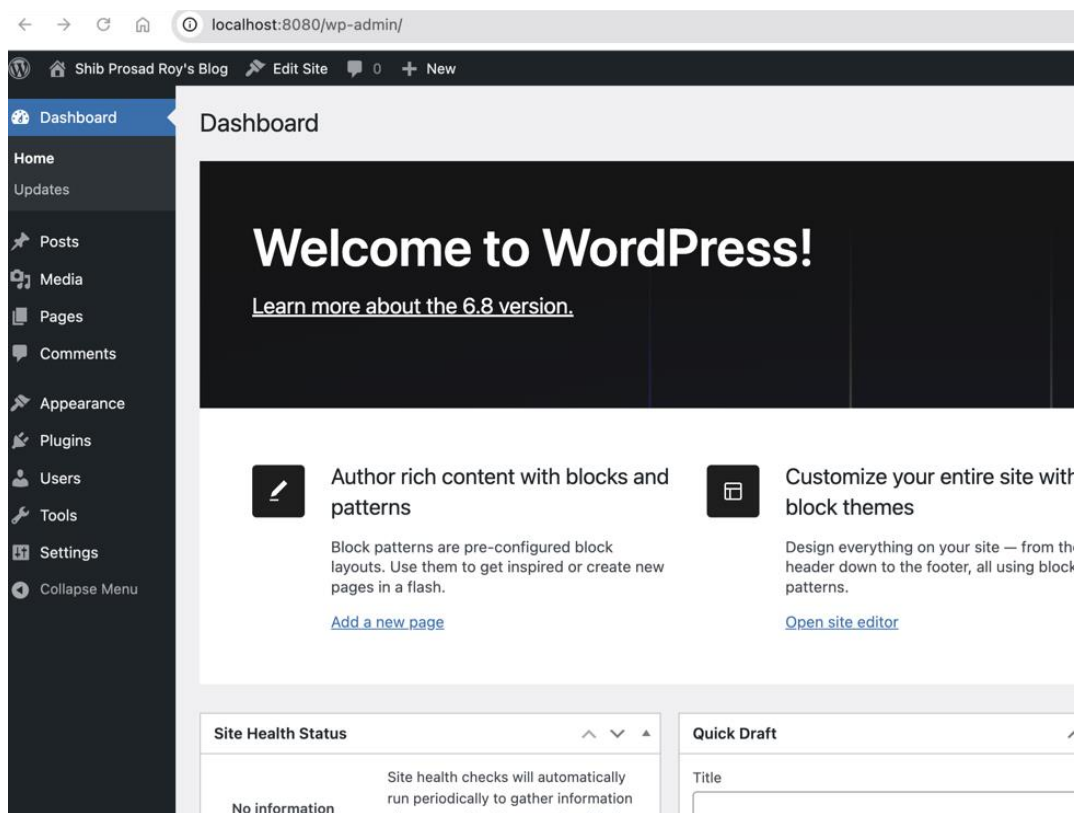
Double-check your email address before continuing.

Search engine visibility: ☒ Discourage search engines from indexing this site

It is up to search engines to honor this request.

[Install WordPress](#)

Fig 3: WordPress Installation Welcome Page



localhost:8080/wp-admin/

Shib Prosad Roy's Blog Edit Site 0 New

Dashboard

Welcome to WordPress!

[Learn more about the 6.8 version.](#)

Author rich content with blocks and patterns

Block patterns are pre-configured block layouts. Use them to get inspired or create new pages in a flash.

[Add a new page](#)

Customize your entire site with block themes

Design everything on your site — from the header down to the footer, all using block patterns.

[Open site editor](#)

Site Health Status

No information

Site health checks will automatically run periodically to gather information

Quick Draft

Title

Fig: WordPress Admin Dashboard

OPS245: Bonus Assignment w/ Docker

Hello world from a Docker Container!

Welcome to WordPress in a Docker Container!
A docker container is a modern, portable, and efficient way to deploy web applications like WordPress. As part of my bonus assignment for OPS245NBB, I decided to reimagine Assignment 2 by using containers instead of a traditional manual LAMP setup. In the original assignment, we installed and configured...

April 17, 2025

Fig: Homepage

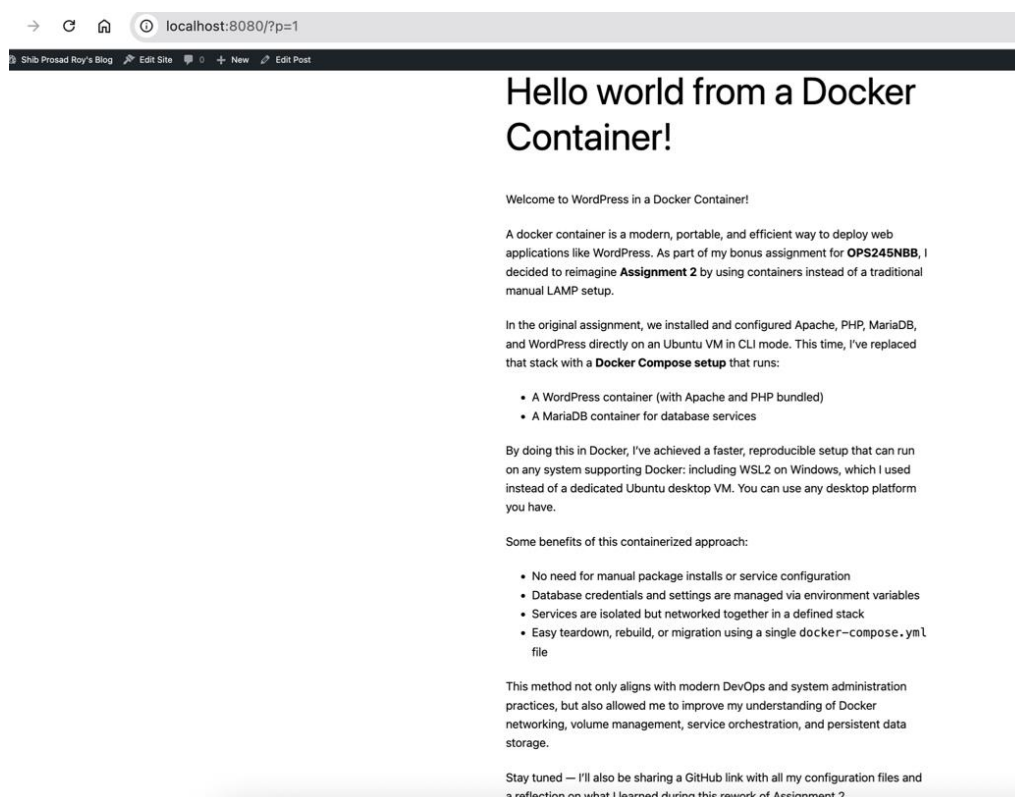


Fig: A blog post

Verification: Exec into Containers

Just like we manage services and applications inside a traditional VM, we can do the same within Docker containers using **docker compose exec**. This allows us to inspect the environment, run administrative commands, and confirm what's running under the hood.

1. Accessing MariaDB and Running SQL Commands

We can connect to the running MariaDB container and access the database CLI as the root user using the following command. It prompts for a password, so we should use same password we defined in `.env` file. Here first string **mariadb** is the name of docker container defined in `docker-compose.yml` file and second string **mariadb** is the actual database name.

docker compose exec -it mariadb mariadb -u root -p

```
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 235
Server version: 11.7.2-MariaDB-ubu2404 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| myblog    |
| mysql     |
| performance_schema |
| sys       |
+-----+
5 rows in set (0.002 sec)

MariaDB [(none)]> use myblog
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [myblog]> SHOW tables;
+-----+
| Tables_in_myblog |
+-----+
| wp_commentmeta   |
| wp_comments      |
| wp_links         |
| wp_options       |
| wp_postmeta      |
| wp_posts         |
| wp_term_relationships |
| wp_term_taxonomy |
| wp_termmeta      |
| wp_terms         |
| wp_usermeta      |
+-----+
```

Fig: DB Operations in a Docker Container

This allowed us to run SQL commands, such as `SHOW DATABASES;`, to verify that the `myblog` database had been created successfully.

2. Exploring the WordPress Container

We can also access the WordPress container to explore its file system and installed packages:

docker compose exec -it wordpress bash

Inside the container, we can run traditional commands like `apt list` and `ls`

This confirms the base image is Debian/Ubuntu-based and includes necessary system-level packages and packages required to run WordPress such as `apache2`.

```
root@84ec1e5135ca:/var/www/html# apt list
Listing... Done
adduser/now 3.134 all [installed,local]
apache2-bin/now 2.4.62-1~deb12u2 arm64 [installed,local]
apache2-data/now 2.4.62-1~deb12u2 all [installed,local]
apache2-utils/now 2.4.62-1~deb12u2 arm64 [installed,local]
apache2/now 2.4.62-1~deb12u2 arm64 [installed,local]
apt/now 2.6.1 arm64 [installed,local]
autoconf/now 2.71-3 all [installed,local]
base-files/now 12.4+deb12u10 arm64 [installed,local]
base-passwd/now 3.6.1 arm64 [installed,local]
bash/now 5.2.15-2+b7 arm64 [installed,local]
binutils-aarch64-linux-gnu/now 2.40-2 arm64 [installed,local]
binutils-common/now 2.40-2 arm64 [installed,local]
binutils/now 2.40-2 arm64 [installed,local]
bsdutils/now 1:2.38.1-5+deb12u3 arm64 [installed,local]
bzip2/now 1.0.8-5+b1 arm64 [installed,local]
ca-certificates/now 20230311 all [installed,local]
coreutils/now 9.1-1 arm64 [installed,local]
cpp-12/now 12.2.0-14 arm64 [installed,local]
cpp/now 4:12.2.0-3 arm64 [installed,local]
curl/now 7.88.1-10+deb12u12 arm64 [installed,local]
dash/now 0.5.12-2 arm64 [installed,local]
debconf/now 1.5.82 all [installed,local]
debian-archive-keyring/now 2023.3+deb12u1 all [installed,local]
debianutils/now 5.7-0.5~deb12u1 arm64 [installed,local]
diffutils/now 1:3.8-4 arm64 [installed,local]
dpkg-dev/now 1.21.22 all [installed,local]
dpkg/now 1.21.22 arm64 [installed,local]
```

Fig: List packages inside WordPress container showing `apache2`

```
root@84ec1e5135ca:/var/www/html# ls
index.php  wp-activate.php  wp-comments-post.php  wp-config.php  wp-includes  wp-login.php  wp-signup.php
license.txt  wp-admin  wp-config-docker.php  wp-content  wp-links-opml.php  wp-mail.php  wp-trackback.php
readme.html  wp-blog-header.php  wp-config-sample.php  wp-cron.php  wp-load.php  wp-settings.php  xmlrpc.php
root@84ec1e5135ca:/var/www/html# exit
```

Fig: WordPress files inside wordpress docker container

Cleanup: Teardown

When we're done with the project, we can stop and remove the running containers and associated resources.

1. Stop the Containers

docker compose down

This stops and removes the containers but **keeps our volumes and database data**, so we restart later and have our WordPress blog same as it's with just **docker compose up** command.

2. Stop and Remove Everything (including volumes and data)

If we want, we can remove everything including the database and WordPress files:

docker compose down --volumes

This removes:

- Containers
- Volumes (WordPress content and MariaDB data)
- The Docker network created for this Docker Compose project

Warning: This will reset your WordPress site and database.