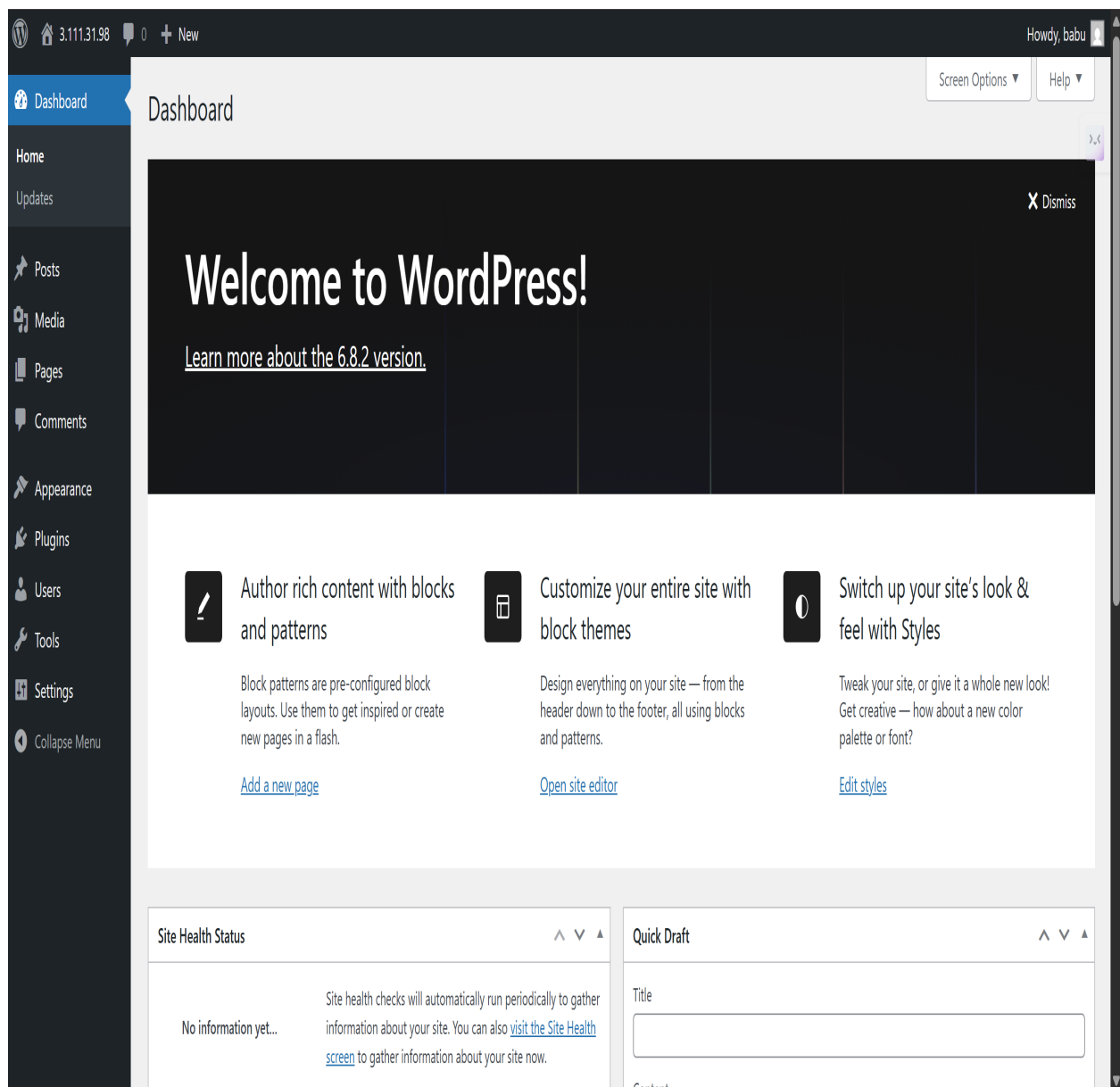


2-TIER ARCHITECTURE DEPLOYMENT OF LAMP STACK WITH WORDPRESS ON AWS EC2

Aniruddha Kharve



WordPress Login/Dashboard Screenshot

Contents

1	Introduction	2
2	Architecture Overview	2
3	Create EC2 Instances	2
4	Configure Security Groups	3
5	Allow SSH and MySQL Access from App Server to DB Server	4
6	DB Server Setup (MariaDB)	5
7	App Server Setup (Apache + PHP + WordPress)	7
8	Finish WordPress Installation	8
9	Conclusion	9

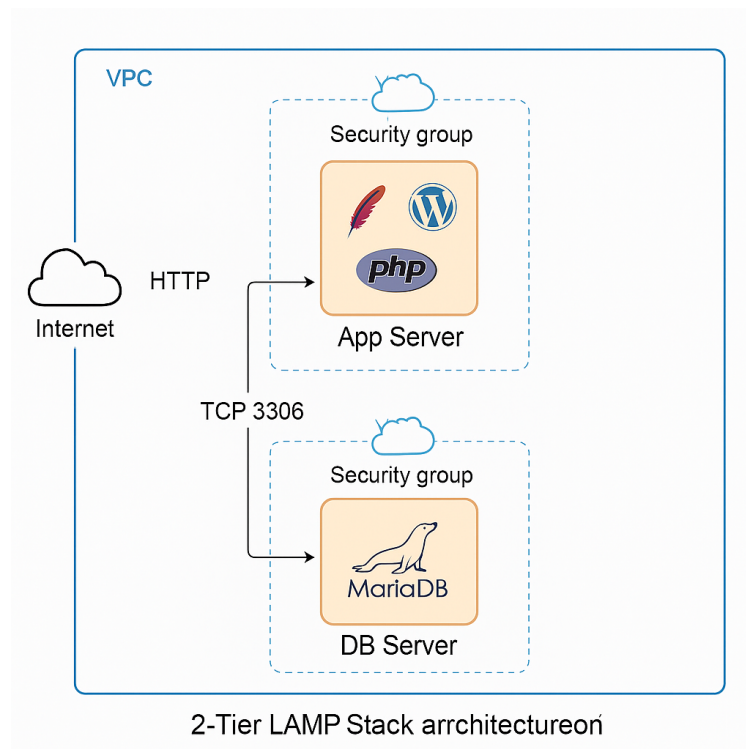
1 Introduction

This project aims to deploy a two-tier LAMP architecture on AWS EC2 instances, where the application and database layers are hosted on separate virtual machines. The project demonstrates real-world architecture design, server configuration, WordPress deployment, and cloud resource management suitable for Linux Admin or DevOps beginners.

2 Architecture Overview

The architecture includes:

- App Server (Amazon Linux AMI) hosting Apache, PHP, and WordPress
- DB Server (Amazon Linux AMI) running MariaDB 10.5
- Communication between them is done over private IP (port 3306)

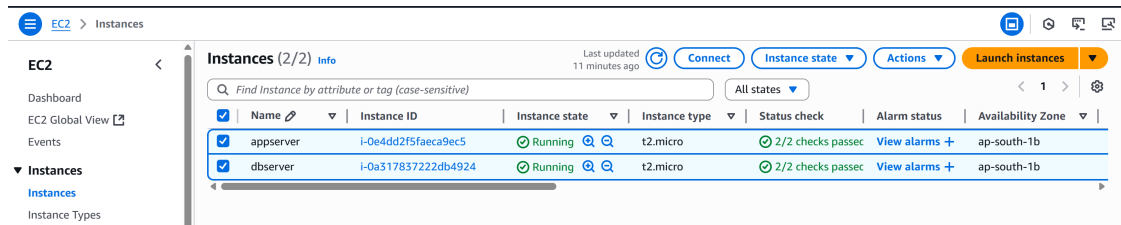


Architecture diagram showing App EC2 ↔ DB EC2

3 Create EC2 Instances

- Launch two EC2 instances using Amazon Linux AMI (App and DB)
- Use the same key pair for both for simplicity
- Choose "t2.micro" instance type under free tier

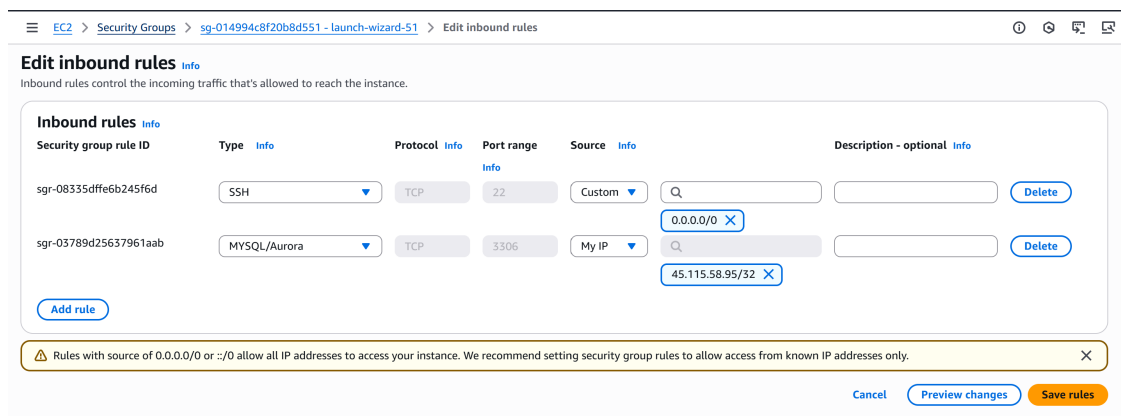
- Place both in the same VPC/subnet for private IP access



2 EC2 instance Created

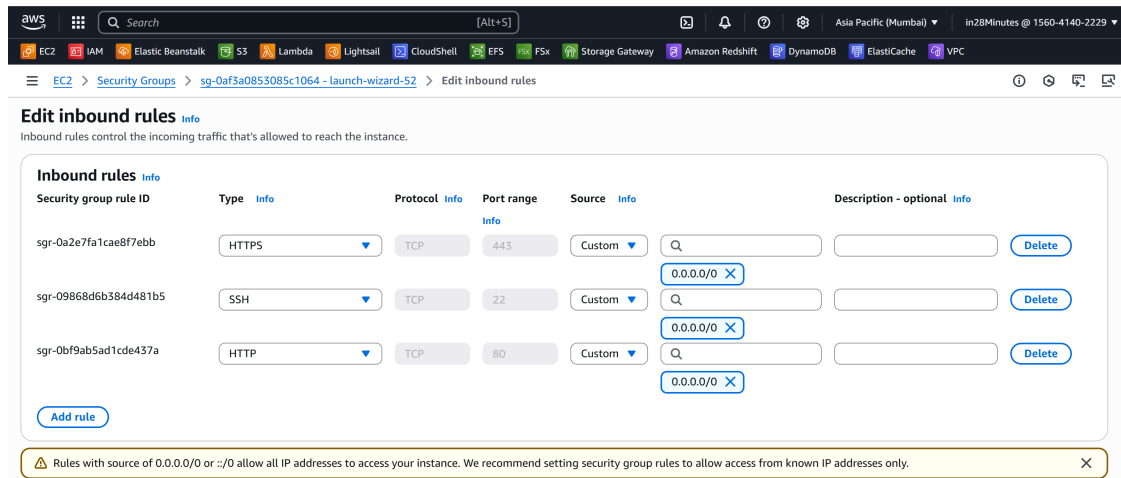
4 Configure Security Groups

- DB Server:
 - Allow inbound SSH (port 22) from App Server's private IP
 - Allow inbound MySQL (port 3306) from App Server's private IP



DB server Security Group

- App Server:
 - Allow inbound SSH (port 22) from your public IP
 - Allow inbound HTTP (port 80) from anywhere



APP server Security Group

5 Allow SSH and MySQL Access from App Server to DB Server

To enable your App Server to connect with the Database Server, follow these steps:

Step 1: Enable Password or SSH Key Access on DB Server

- On the DB server, edit the SSH configuration:

```
sudo nano /etc/ssh/sshd_config
```

- Make sure the following lines are set:

```
PermitRootLogin yes
PasswordAuthentication yes
```

- Restart the SSH service:

```
sudo systemctl restart sshd
```

Step 2: From App Server, SSH into DB Server

You can now access the DB Server using:

```
ssh ec2-user@<DB_SERVER_PRIVATE_IP>
# OR if root login is enabled:
ssh root@<DB_SERVER_PRIVATE_IP>
```

Step 3: Enable MySQL to Accept Remote Connections

Edit MariaDB config file on the DB Server:

```
sudo nano /etc/my.cnf
```

Add this line ****under**** '[client-server]':

```
bind-address=0.0.0.0
```

Step 4: Add Inbound Rule in DB Server's Security Group

- Go to AWS Console → EC2 → Security Groups
- Edit inbound rules of the ****DB Server****
- Add rule: ****MySQL/Aurora – TCP 3306 – Source: App Server's private IP****

This configuration will allow the WordPress site on the App Server to connect to MariaDB hosted on the DB Server securely using private IP.

6 DB Server Setup (MariaDB)

- SSH into the DB server:

```
ssh -i key.pem ec2-user@<DB_SERVER_PUBLIC_IP>
```

- Install MariaDB:

```
sudo dnf install -y mariadb105-server mariadb105
sudo systemctl enable --now mariadb
```

- Secure the database and create WordPress database:

```
sudo mysql -u root
CREATE DATABASE wordpress;
CREATE USER 'wordpress'@'%' IDENTIFIED BY 'yourpassword';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'%';
FLUSH PRIVILEGES;
```

- Allow external access by editing /etc/my.cnf:

```
sudo nano /etc/my.cnf
# Add this line under [client-server] or at the bottom
bind-address=0.0.0.0
```

- Restart MariaDB and open firewall (if needed):

```
sudo systemctl restart mariadb
sudo firewall-cmd --permanent --add-port=3306/tcp
sudo firewall-cmd --reload
```

Error: Unable to find a match: mariadb-server

```
[ec2-user@DB-server ~]$ sudo dnf install -y mariadb105-server mariadb105
```

Last metadata expiration check: 0:04:57 ago on Thu Jul 31 06:28:49 2025.
Dependencies resolved.

Package	Architecture	Version	Repository	Size
Installing:				
mariadb105-server	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	10 M
Installing dependencies:				
mariadb-connector-c	x86_64	3.3.10-1.amzn2023.0.1	amazonlinux	211 k
mariadb-connector-c-config	noarch	3.3.10-1.amzn2023.0.1	amazonlinux	9.9 k
mariadb105	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	1.5 M
mariadb105-common	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	28 k
mariadb105-errmsg	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	212 k
mysql-selinux	noarch	1.0.4-2.amzn2023.0.3	amazonlinux	36 k
perl-B	x86_64	1.80-477.amzn2023.0.7	amazonlinux	177 k
perl-DBD-MariaDB	x86_64	1.22-1.amzn2023.0.4	amazonlinux	153 k
perl-DBI	x86_64	1.643-7.amzn2023.0.3	amazonlinux	700 k
perl-Data-Dumper	x86_64	2.174-460.amzn2023.0.2	amazonlinux	55 k
perl-File-Copy	noarch	2.34-477.amzn2023.0.7	amazonlinux	20 k
perl-FileHandle	noarch	2.03-477.amzn2023.0.7	amazonlinux	15 k
perl-Math-BigInt	noarch	1:1.9998.39-2.amzn2023.0.2	amazonlinux	202 k
perl-Math-BigRat	noarch	0.2624-500.amzn2023.0.2	amazonlinux	42 k
perl-Math-Complex	noarch	1.59-477.amzn2023.0.7	amazonlinux	46 k
perl-Sys-Hostname	x86_64	1.23-477.amzn2023.0.7	amazonlinux	16 k
perl-base	noarch	2.27-477.amzn2023.0.7	amazonlinux	16 k
Installing weak dependencies:				
mariadb105-backup	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	6.0 M
mariadb105-cracklib-password-check	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	13 k
mariadb105-gssapi-server	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	15 k
mariadb105-server-utils	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	207 k

Transaction Summary

MariaDB installation

```
[ec2-user@DB-server ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 7
Server version: 10.5.29-MariaDB MariaDB Server

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)]> 
```

MariaDB Login

```
MariaDB [(none)]> CREATE DATABASE wordpress;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user 'wordpress'@'3.109.3.54' IDENTIFIED BY '@Aniruddha071';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'3.109.3.54' ;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> 
```

Database Creation

7 App Server Setup (Apache + PHP + WordPress)

- SSH into App Server and install packages:

```
sudo yum install -y httpd php php-mysqlnd php-fpm wget
sudo systemctl enable --now httpd
```

- Download and extract WordPress:

```
wget https://wordpress.org/latest.tar.gz
sudo tar -xzf latest.tar.gz -C /var/www/html/
```

- Set ownership and permissions:

```
sudo chown -R apache:apache /var/www/html/wordpress
sudo chmod -R 755 /var/www/html/wordpress
```

- Configure wp-config.php:

```
cd /var/www/html/wordpress
cp wp-config-sample.php wp-config.php
sudo nano wp-config.php
```

- Edit DB details:

```
define( 'DB_NAME', 'wordpress' );
define( 'DB_USER', 'wordpress' );
define( 'DB_PASSWORD', 'yourpassword' );
define( 'DB_HOST', '<DB_PRIVATE_IP>' );
```

[ec2-user@APP-Server ~]\$ sudo yum install httpd php php-mysqlnd php-fpm -y				
Amazon Linux 2023 Kernel Livepatch repository				
Dependencies resolved.				
Package	Architecture	Version	Repository	Size
Installing:				
httpd	x86_64	2.4.62-1.amzn2023	amazonlinux	48 k
php8.4	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	17 k
php8.4-fpm	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	2.0 M
php8.4-mysqlnd	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	156 k
Installing dependencies:				
apr	x86_64	1.7.5-1.amzn2023.0.4	amazonlinux	129 k
apr-util	x86_64	1.6.3-1.amzn2023.0.1	amazonlinux	98 k
generic-logos-httpd	noarch	18.0.0-12.amzn2023.0.3	amazonlinux	19 k
httpd-core	x86_64	2.4.62-1.amzn2023	amazonlinux	1.4 M
httpd-filesystem	noarch	2.4.62-1.amzn2023	amazonlinux	14 k
httpd-tools	x86_64	2.4.62-1.amzn2023	amazonlinux	81 k
libbrotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	315 k
libsodium	x86_64	1.0.19-4.amzn2023	amazonlinux	176 k
libxslt	x86_64	1.1.43-1.amzn2023.0.1	amazonlinux	183 k
mailcap	noarch	2.1.49-3.amzn2023.0.3	amazonlinux	33 k
nginx-filesystem	noarch	1:1.28.0-1.amzn2023.0.1	amazonlinux	9.5 k
php8.4-cli	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	3.8 M
php8.4-common	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	800 k
php8.4-pdo	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	100 k
php8.4-process	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	52 k
php8.4-xml	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	966 k
Installing weak dependencies:				
apr-util-openssl	x86_64	1.6.3-1.amzn2023.0.1	amazonlinux	17 k
mod_http2	x86_64	2.0.27-1.amzn2023.0.3	amazonlinux	166 k
mod_lua	x86_64	2.4.62-1.amzn2023	amazonlinux	61 k
php8.4-mbstring	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	541 k
php8.4-opcache	x86_64	8.4.10-1.amzn2023.0.1	amazonlinux	495 k

Install Apache server, PHP, Wordpress On App-server

```
[root@APP-Server ~]# cd /var/www/html
[root@APP-Server html]# sudo wget https://wordpress.org/latest.tar.gz
--2025-07-31 06:53:29-- https://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26925441 (26M) [application/octet-stream]
Saving to: 'latest.tar.gz'

latest.tar.gz          100%[=====>] 25.68M  6.69MB/s  in 5.5s

2025-07-31 06:53:35 (4.66 MB/s) - 'latest.tar.gz' saved [26925441/26925441]

[root@APP-Server html]# sudo tar -xzf latest.tar.gz
[root@APP-Server html]# ls
latest.tar.gz  wordpress
[root@APP-Server html]# sudo cp -r wordpress/* .
[root@APP-Server html]# ls
index.php  readme.html  wp-admin  wp-config-sample.php  wp-includes  wp-login.php  wp-signup.php
license.txt  wp-activate.php  wp-blog-header.php  wp-content  wp-links-opml.php  wp-mail.php  wp-trackback.php
[root@APP-Server html]# sudo chown -R apache:apache /var/www/html
[root@APP-Server html]#
```

WordPress Configuration

```
// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** Database username */
define( 'DB_USER', 'wordpress' );

/** Database password */
define( 'DB_PASSWORD', '@Aniruddha071' );

/** Database hostname */
define( 'DB_HOST', '172.31.1.171' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/**#@+
 * Authentication unique keys and salts.
 *
 * Change these to different unique phrases! You can generate these using
 * -- INSERT --
```

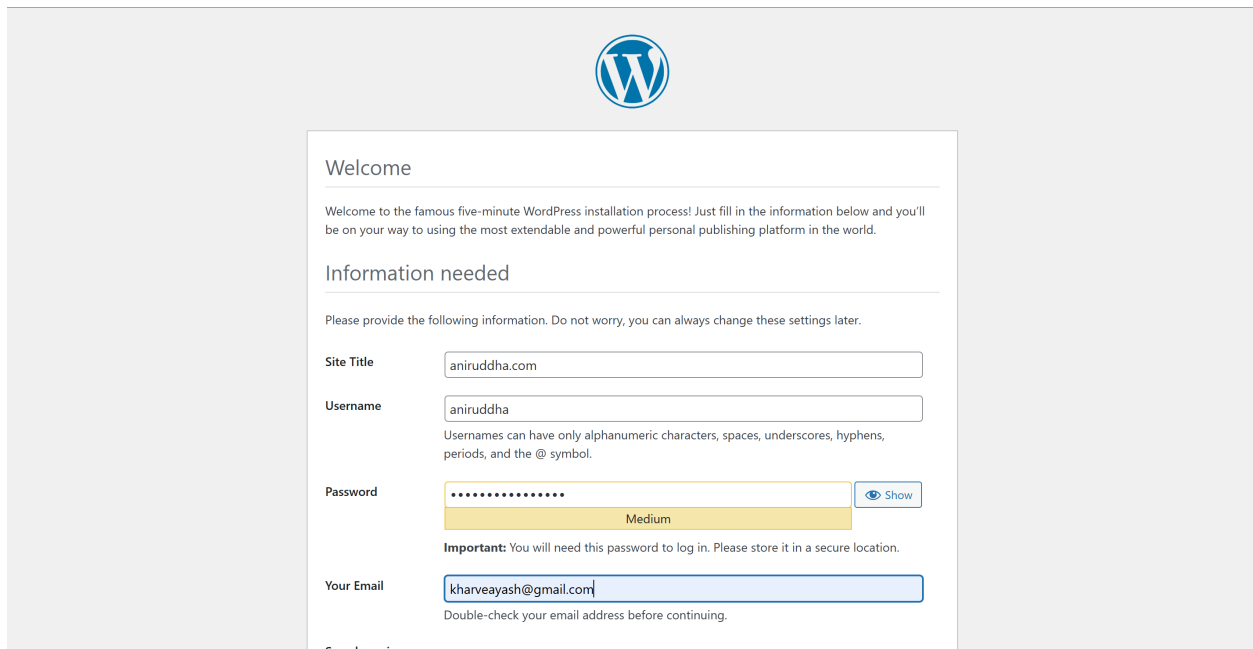
Database Congiguration in wp-config.php file

8 Finish WordPress Installation

- Go to your App server's public IP in the browser:

`http://<APP_SERVER_PUBLIC_IP>/wordpress`

- Follow on-screen instructions to complete WordPress installation
- Set site name, admin user, password, and email
- Login to the WordPress dashboard

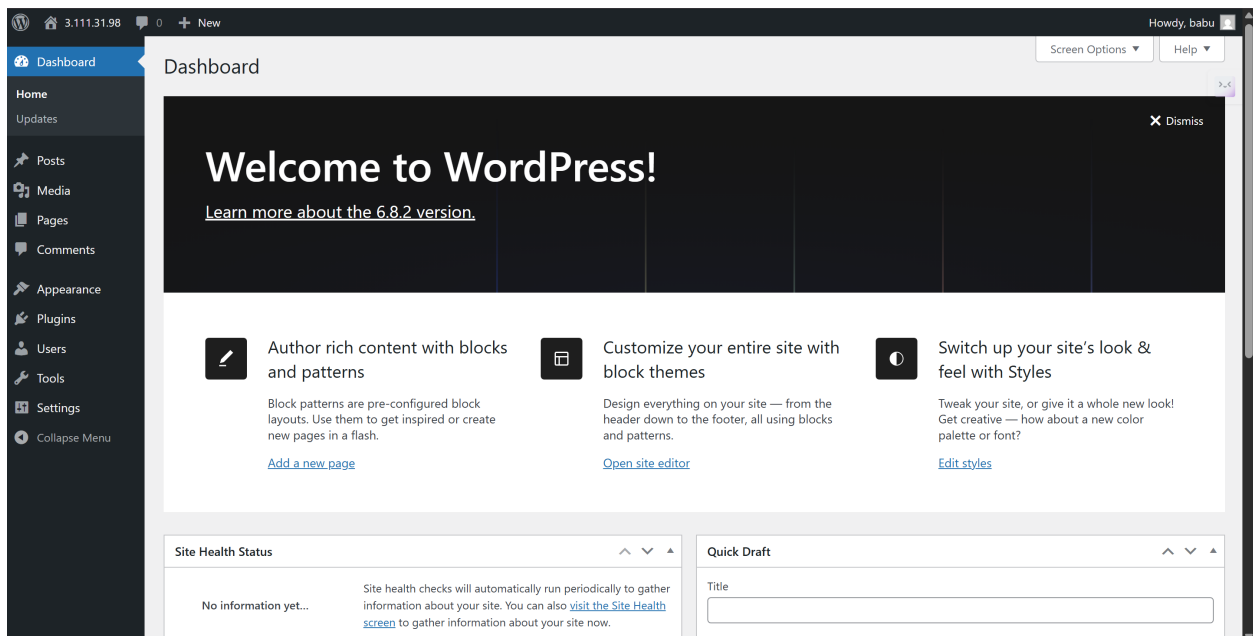


The image shows the WordPress installation 'Welcome' screen. At the top is the WordPress logo. Below it, a 'Welcome' heading is followed by a paragraph: 'Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.' The 'Information needed' section asks for the following details:

- Site Title:**
- Username:**
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
- Password:**
A strength indicator shows 'Medium'. A 'Show' button is next to it.
- Your Email:**
Double-check your email address before continuing.

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

Create account in WordPress First



Your Wordpress Account Page

9 Conclusion

This project successfully deployed a two-tier LAMP stack using AWS EC2 instances and separated the database layer from the application server. You learned how to configure Apache, PHP, and MariaDB, as well as use cloud infrastructure securely and efficiently.