**AWS EC2 Deployment Guide for PHP & MySQL/MariaDB Applications**

**Step 1: Launch an EC2 Instance**

1. Log in to AWS Management Console.
2. Navigate to EC2 Dashboard → Click Launch Instance.
3. Choose Ubuntu 22.04 LTS as your OS.
4. Select t2.micro (free-tier eligible) or a higher instance type if needed.
5. Click Next: Configure Instance Details → Keep default settings.
6. Click Next: Add Storage → Increase root volume (at least 20GB recommended).
7. Click Next: Add Tags → Add a tag (optional, e.g., Name = PHP-Server).
8. Click Next: Configure Security Group:
   * Add rules:
     + SSH (22) → Source: My IP
     + HTTP (80) → Source: Anywhere (0.0.0.0/0)
     + HTTPS (443) → Source: Anywhere (0.0.0.0/0)
     + MySQL (3306) → Source: My IP (for remote database access, optional)
9. Click Review and Launch.
10. Create and download a new key pair (.pem file) → Launch Instance.

**Step 2: Connect to Your EC2 Instance**

**For Windows Users (PuTTY):**

1. Convert .pem to .ppk using PuTTYgen:
   * Open PuTTYgen → Load .pem file → Save Private Key (.ppk).
2. Connect using PuTTY:
   * Host Name: ubuntu@your-ec2-public-ip
   * Port: 22
   * Auth: Load the .ppk file under "SSH → Auth → Credentials"
   * Click Open.

**For macOS/Linux Users:**

1. Open Terminal.
2. Set permissions for your key:
3. chmod 400 your-key.pem
4. Connect via SSH:
5. ssh -i your-key.pem ubuntu@your-ec2-public-ip

**Step 3: Update System and Install LAMP Stack**

Update system packages:

sudo apt update && sudo apt upgrade -y

**Install Apache Web Server**

sudo apt install apache2 -y

sudo systemctl enable apache2

sudo systemctl start apache2

**Install PHP and Common Extensions**

sudo apt install php libapache2-mod-php php-mysql php-curl php-gd php-json php-zip php-mbstring php-xml php-intl -y

Verify PHP installation:

php -v

**Install MySQL or MariaDB**

**Option 1: MySQL**

sudo apt install mysql-server -y

sudo systemctl enable mysql

sudo systemctl start mysql

# Secure MySQL installation

sudo mysql\_secure\_installation

**Option 2: MariaDB**

sudo apt install mariadb-server -y

sudo systemctl enable mariadb

sudo systemctl start mariadb

# Secure MariaDB installation

sudo mysql\_secure\_installation

During the secure installation, you'll be prompted to:

* Set a root password
* Remove anonymous users
* Disallow root login remotely
* Remove test database
* Reload privilege tables

**Create a Database and User**

sudo mysql -u root -p

Run these SQL commands:

CREATE DATABASE your\_database;

CREATE USER 'your\_user'@'localhost' IDENTIFIED BY 'your\_secure\_password';

GRANT ALL PRIVILEGES ON your\_database.\* TO 'your\_user'@'localhost';

FLUSH PRIVILEGES;

EXIT;

**Step 4: Configure Apache Virtual Host**

Create a new virtual host configuration:

sudo nano /etc/apache2/sites-available/your-domain.conf

Add this configuration:

<VirtualHost \*:80>

ServerName your-domain.com

ServerAlias www.your-domain.com

ServerAdmin webmaster@your-domain.com

DocumentRoot /var/www/html/your-project

<Directory /var/www/html/your-project>

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

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

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

</VirtualHost>

Save and exit.

Create the web directory:

sudo mkdir -p /var/www/html/your-project

Enable the virtual host and rewrite module:

sudo a2ensite your-domain.conf

sudo a2enmod rewrite

sudo systemctl restart apache2

**Step 5: Transfer Your PHP Application**

**Option 1: Using FileZilla (SFTP)**

1. Open FileZilla.
2. Go to Edit → Settings → SFTP → Add key file (.pem or .ppk).
3. In FileZilla Site Manager:
   * Host: your-ec2-public-ip
   * Protocol: SFTP
   * User: ubuntu
   * Password: Leave blank
   * Logon Type: Key File
   * Key File: Select your .ppk file
   * Click Connect.
4. Navigate to /var/www/html/your-project in the remote server.
5. Upload your PHP files.

**Option 2: Using Git**

1. Install Git:
2. sudo apt install git -y
3. Navigate to your project directory:
4. cd /var/www/html
5. Clone your repository:
6. sudo git clone https://github.com/username/your-project.git

**Option 3: Using SCP (Secure Copy)**

From your local machine:

scp -i your-key.pem -r /path/to/local/project ubuntu@your-ec2-public-ip:/var/www/html/your-project

**Step 6: Set Proper Permissions**

Set ownership:

sudo chown -R www-data:www-data /var/www/html/your-project

Set proper permissions:

sudo find /var/www/html/your-project -type d -exec chmod 755 {} \;

sudo find /var/www/html/your-project -type f -exec chmod 644 {} \;

**Step 7: Configure PHP Application**

Create a configuration file if needed:

sudo nano /var/www/html/your-project/config.php

Example configuration file:

<?php

// Database connection

define('DB\_HOST', 'localhost');

define('DB\_USER', 'your\_user');

define('DB\_PASS', 'your\_secure\_password');

define('DB\_NAME', 'your\_database');

// Connect to database

$conn = new mysqli(DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**Step 8: Set Up SSL/TLS with Certbot**

Install Certbot:

sudo apt install certbot python3-certbot-apache -y

Obtain and install SSL certificate:

sudo certbot --apache -d your-domain.com -d www.your-domain.com

Follow the prompts to complete the setup.

Test certificate auto-renewal:

sudo certbot renew --dry-run

**Step 9: Configure PHP for Production**

Edit php.ini:

sudo nano /etc/php/\*/apache2/php.ini

Recommended production settings:

memory\_limit = 256M

upload\_max\_filesize = 64M

post\_max\_size = 64M

max\_execution\_time = 300

display\_errors = Off

error\_reporting = E\_ALL & ~E\_DEPRECATED & ~E\_STRICT

log\_errors = On

error\_log = /var/log/php\_errors.log

date.timezone = UTC

Restart Apache:

sudo systemctl restart apache2

**Step 10: Install phpMyAdmin (Optional)**

Install phpMyAdmin:

sudo apt install phpmyadmin -y

During installation:

* Select Apache2 when prompted
* Choose to configure database with dbconfig-common
* Set phpMyAdmin password

Secure phpMyAdmin:

sudo nano /etc/apache2/conf-available/phpmyadmin.conf

Add these lines inside the <Directory /usr/share/phpmyadmin> section:

AllowOverride All

<IfModule mod\_authz\_core.c>

<RequireAny>

Require ip YOUR\_IP\_ADDRESS

Require local

</RequireAny>

</IfModule>

Create a .htaccess file for additional security:

sudo nano /usr/share/phpmyadmin/.htaccess

Add:

AuthType Basic

AuthName "Restricted Access"

AuthUserFile /etc/phpmyadmin/.htpasswd

Require valid-user

Create a username/password for phpMyAdmin access:

sudo htpasswd -c /etc/phpmyadmin/.htpasswd your\_username

Enable the configuration and restart Apache:

sudo a2enconf phpmyadmin

sudo systemctl restart apache2

**Step 11: Set Up Database Backups**

Create a backup script:

sudo nano /usr/local/bin/backup-db.sh

Add this content:

#!/bin/bash

TIMESTAMP=$(date +"%Y%m%d-%H%M%S")

BACKUP\_DIR="/home/ubuntu/database-backups"

MYSQL\_USER="your\_user"

MYSQL\_PASSWORD="your\_secure\_password"

DATABASE\_NAME="your\_database"

# Create backup directory if it doesn't exist

mkdir -p $BACKUP\_DIR

# Create SQL dump

mysqldump -u $MYSQL\_USER -p$MYSQL\_PASSWORD $DATABASE\_NAME > $BACKUP\_DIR/$DATABASE\_NAME-$TIMESTAMP.sql

# Compress backup

gzip $BACKUP\_DIR/$DATABASE\_NAME-$TIMESTAMP.sql

# Delete backups older than 30 days

find $BACKUP\_DIR -name "\*.sql.gz" -type f -mtime +30 -delete

Make it executable:

sudo chmod +x /usr/local/bin/backup-db.sh

Schedule automatic backups with cron:

sudo crontab -e

Add this line to run the backup daily at 2 AM:

0 2 \* \* \* /usr/local/bin/backup-db.sh

**Troubleshooting Common Issues**

**Apache Issues**

* Check Apache status: sudo systemctl status apache2
* Check Apache error logs: sudo tail -f /var/log/apache2/error.log
* Test Apache configuration: sudo apache2ctl configtest

**PHP Issues**

* Check PHP error log: sudo tail -f /var/log/php\_errors.log
* Verify PHP modules: php -m
* Test PHP installation: php -i

**MySQL/MariaDB Issues**

* Check database status: sudo systemctl status mysql or sudo systemctl status mariadb
* Check database logs: sudo tail -f /var/log/mysql/error.log
* Test database connection: mysql -u your\_user -p -h localhost

**Permission Issues**

* If you get "Permission denied" errors, check ownership: ls -la /var/www/html/your-project
* Fix permissions with: sudo chown -R www-data:www-data /var/www/html/your-project

**SSL Issues**

* Verify SSL certificate: sudo certbot certificates
* Renew certificate: sudo certbot renew
* Check SSL configuration: sudo apache2ctl -t -D DUMP\_VHOSTS