Troubleshoot

- **My Apache webserver doesn't start unless I enter a password**

  This is expected behavior if you installed an encrypted, password-protected, private server key.

  You can remove the encryption and password requirement from the key. Assuming that you have a private encrypted RSA key called `custom.key` in the default directory, and that the password on it is **abcde12345**, run the following commands on your EC2 instance to generate an unencrypted version of the key.

  ```
  [ec2-user ~]$ cd /etc/pki/tls/private/
  [ec2-user private]$ sudo cp custom.key custom.key.bak
  [ec2-user private]$ sudo openssl rsa -in custom.key -passin pass:abcde12345 -out
   custom.key.nocrypt
  [ec2-user private]$ sudo mv custom.key.nocrypt custom.key
  [ec2-user private]$ sudo chown root:root custom.key
  [ec2-user private]$ sudo chmod 600 custom.key
  [ec2-user private]$ sudo systemctl restart httpd
  ```

  Apache should now start without prompting you for a password.

- **I get errors when I run sudo yum install -y mod_ssl.**

  When you are installing the required packages for SSL, you may see errors similar to the following.

  ```
  Error: httpd24-tools conflicts with httpd-tools-2.2.34-1.16.amzn1.x86_64
  Error: httpd24 conflicts with httpd-2.2.34-1.16.amzn1.x86_64
  ```

  This typically means that your EC2 instance is not running AL2. This tutorial only supports instances freshly created from an official AL2 AMI.

Tutorial: Host a WordPress blog on AL2

The following procedures will help you install, configure, and secure a WordPress blog on your AL2 instance. This tutorial is a good introduction to using Amazon EC2 in that you have full control over a web server that hosts your WordPress blog, which is not typical with a traditional hosting service.

You are responsible for updating the software packages and maintaining security patches for your server. For a more automated WordPress installation that does not require direct interaction with the web server configuration, the AWS CloudFormation service provides a WordPress template that can also get you started quickly. For more information, see Get started in the *AWS CloudFormation User Guide*. If you need a high-availability solution with a decoupled database, see Deploying a high-availability WordPress website in the *AWS Elastic Beanstalk Developer Guide*.

> **Important**
>
> These procedures are intended for use with AL2. For more information about other distributions, see their specific documentation. Many steps in this tutorial do not work on Ubuntu instances. For help installing WordPress on an Ubuntu instance, see WordPress in the Ubuntu documentation. You can also use CodeDeploy to accomplish this task on Amazon Linux, macOS, or Unix systems.

Topics

Prerequisites

This tutorial assumes that you have launched an AL2 instance with a functional web server with PHP and database (either MySQL or MariaDB) support by following all of the steps in Tutorial: Install a LAMP server on AL2. This tutorial also has steps for configuring a security group to allow HTTP and HTTPS traffic, as well as several steps to ensure that file permissions are set properly for your web server. For information about adding rules to your security group, see Add rules to a security group.

We strongly recommend that you associate an Elastic IP address (EIP) to the instance you are using to host a WordPress blog. This prevents the public DNS address for your instance from changing and breaking your installation. If you own a domain name and you want to use it for your blog, you can update the DNS record for the domain name to point to your EIP address (for help with this, contact your domain name registrar). You can have one EIP address associated with a running instance at no charge. For more information, see Elastic IP addresses in the *Amazon EC2 User Guide*.

If you don't already have a domain name for your blog, you can register a domain name with Route 53 and associate your instance's EIP address with your domain name. For more information, see Registering domain names using Amazon Route 53 in the *Amazon Route 53 Developer Guide*.

Install WordPress

Option: Complete this tutorial using automation

To complete this tutorial using AWS Systems Manager Automation instead of the following tasks, run the automation document.

Connect to your instance, and download the WordPress installation package.

To download and unzip the WordPress installation package

1. Download the latest WordPress installation package with the **wget** command. The following command should always download the latest release.

   ```
   [ec2-user ~]$ wget https://wordpress.org/latest.tar.gz
   ```

2. Unzip and unarchive the installation package. The installation folder is unzipped to a folder called `wordpress`.

   ```
   [ec2-user ~]$ tar -xzf latest.tar.gz
   ```

To create a database user and database for your WordPress installation

Your WordPress installation needs to store information, such as blog posts and user comments, in a database. This procedure helps you create your blog's database and a user that is authorized to read and save information to it.

1. Start the database server.

   - ```
     [ec2-user ~]$ sudo systemctl start mariadb
     ```

2. Log in to the database server as the `root` user. Enter your database `root` password when prompted; this may be different than your `root` system password, or it might even be empty if you have not secured your database server.

   If you have not secured your database server yet, it is important that you do so. For more information, see To secure the MariaDB server (AL2).
     ```

```
[ec2-user ~]$ mysql -u root -p
```

3. Create a user and password for your MySQL database. Your WordPress installation uses these values to communicate with your MySQL database.

   Make sure that you create a strong password for your user. Do not use the single quote character (') in your password, because this will break the preceding command. Do not reuse an existing password, and make sure to store this password in a safe place.

   Enter the following command, substituting a unique user name and password.

```
CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';
```

4. Create your database. Give your database a descriptive, meaningful name, such as `wordpress-db`.

   > **Note**
   >
   > The punctuation marks surrounding the database name in the command below are called backticks. The backtick (`) key is usually located above the Tab key on a standard keyboard. Backticks are not always required, but they allow you to use otherwise illegal characters, such as hyphens, in database names.

```
CREATE DATABASE `wordpress-db`;
```

5. Grant full privileges for your database to the WordPress user that you created earlier.

```
GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
```

6. Flush the database privileges to pick up all of your changes.

```
FLUSH PRIVILEGES;
```

7. Exit the `mysql` client.

```
exit
```

To create and edit the wp-config.php file

The WordPress installation folder contains a sample configuration file called `wp-config-sample.php`. In this procedure, you copy this file and edit it to fit your specific configuration.

1. Copy the `wp-config-sample.php` file to a file called `wp-config.php`. This creates a new configuration file and keeps the original sample file intact as a backup.

   ```
   [ec2-user ~]$ cp wordpress/wp-config-sample.php wordpress/wp-config.php
   ```

2. Edit the `wp-config.php` file with your favorite text editor (such as **nano** or **vim**) and enter values for your installation. If you do not have a favorite text editor, `nano` is suitable for beginners.

   ```
   [ec2-user ~]$ nano wordpress/wp-config.php
   ```

   a. Find the line that defines `DB_NAME` and change `database_name_here` to the database name that you created in Step 4 of To create a database user and database for your WordPress installation.

   ```
   define('DB_NAME', 'wordpress-db');
   ```

   b. Find the line that defines `DB_USER` and change `username_here` to the database user that you created in Step 3 of To create a database user and database for your WordPress installation.

   ```
   define('DB_USER', 'wordpress-user');
   ```

   c. Find the line that defines `DB_PASSWORD` and change `password_here` to the strong password that you created in Step 3 of To create a database user and database for your WordPress installation.

   ```
   define('DB_PASSWORD', 'your_strong_password');
   ```

   d. Find the section called `Authentication Unique Keys and Salts`. These `KEY` and `SALT` values provide a layer of encryption to the browser cookies that WordPress users store on their local machines. Basically, adding long, random values here makes your site more secure. Visit https://api.wordpress.org/secret-key/1.1/salt/ to randomly generate a set of key values that you can copy and paste into your `wp-config.php` file. To paste text

into a PuTTY terminal, place the cursor where you want to paste the text and right-click your mouse inside the PuTTY terminal.

For more information about security keys, go to https://wordpress.org/support/article/editing-wp-config-php/#security-keys.

> **Note**
>
> The values below are for example purposes only; do not use these values for your installation.

```
define('AUTH_KEY',         ' #U$$+[RXN8:b^-L 0(WU_+ c+WFkI~c]o]-bHw+)/
Aj[wTwSiZ<Qb[mghEXcRh-');
define('SECURE_AUTH_KEY',  'Zsz._P=l/|y.Lq)XjlkwS1y5NJ76E6EJ.AV0pCKZZB,*~*r ?
6OP$eJT@;+(ndLg');
define('LOGGED_IN_KEY',    'ju}qwre3V*+8f_zOWf?{LlGsQ]Ye@2Jh^,8x>)Y |;(^[Iw]Pi
+LG#A4R?7N`YB3');
define('NONCE_KEY',        'P(g62HeZxEes|LnI^i=H,[XwK9I&[2s|:?0N}VJM%?;v2v]v+;
+^9eXUahg@::Cj');
define('AUTH_SALT',        'C$DpB4Hj[JK:?{ql`sRVa:{:7yShy(9A@5wg+`JJVb1fk%_-
Bx*M4(qc[Qg%JT!h');
define('SECURE_AUTH_SALT', 'd!uRu#}+q#{f$Z?Z9uFPG.${+S{n~1M&%@~gL>U>NV<zpD-@2-
Es7Q1O-bp28EKv');
define('LOGGED_IN_SALT',   ';j{00P*owZf)kVD+FVLn-~ >.|Y%Ug4#I^*LVd9QeZ^&XmK|
e(76miC+&W&+^0P/');
define('NONCE_SALT',       '-97r*V/cgxLmp?Zy4zUU4r99QQ_rGs2LTd%P;|
_e1tS)8_B/,.6[=UK<J_y9?JWG');
```

e. Save the file and exit your text editor.

To install your WordPress files under the Apache document root

- Now that you've unzipped the installation folder, created a MySQL database and user, and customized the WordPress configuration file, you are ready to copy your installation files to your web server document root so you can run the installation script that completes your installation. The location of these files depends on whether you want your WordPress blog to be available at the actual root of your web server (for example, *my.public.dns.amazonaws.com*) or in a subdirectory or folder under the root (for example, *my.public.dns.amazonaws.com/blog*).

- If you want WordPress to run at your document root, copy the contents of the wordpress installation directory (but not the directory itself) as follows:

```
[ec2-user ~]$ cp -r wordpress/* /var/www/html/
```

- If you want WordPress to run in an alternative directory under the document root, first create that directory, and then copy the files to it. In this example, WordPress will run from the directory `blog`:

```
[ec2-user ~]$ mkdir /var/www/html/blog
[ec2-user ~]$ cp -r wordpress/* /var/www/html/blog/
```

> **Important**
>
> For security purposes, if you are not moving on to the next procedure immediately, stop the Apache web server (`httpd`) now. After you move your installation under the Apache document root, the WordPress installation script is unprotected and an attacker could gain access to your blog if the Apache web server were running. To stop the Apache web server, enter the command **sudo systemctl stop httpd**. If you are moving on to the next procedure, you do not need to stop the Apache web server.

To allow WordPress to use permalinks

WordPress permalinks need to use Apache `.htaccess` files to work properly, but this is not enabled by default on Amazon Linux. Use this procedure to allow all overrides in the Apache document root.

1. Open the `httpd.conf` file with your favorite text editor (such as **nano** or **vim**). If you do not have a favorite text editor, `nano` is suitable for beginners.

```
[ec2-user ~]$ sudo vim /etc/httpd/conf/httpd.conf
```

2. Find the section that starts with `<Directory "/var/www/html">`.

```
<Directory "/var/www/html">
    #
    # Possible values for the Options directive are "None", "All",
```

```
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.4/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
    Require all granted
</Directory>
```

3. Change the `AllowOverride None` line in the above section to read `AllowOverride All`.

> **Note**
>
> There are multiple `AllowOverride` lines in this file; be sure you change the line in the `<Directory "/var/www/html">` section.

```
AllowOverride All
```

4. Save the file and exit your text editor.

To install the PHP graphics drawing library on AL2

The GD library for PHP enables you to modify images. Install this library if you need to crop the header image for your blog. The version of phpMyAdmin that you install might require a specific minimum version of this library (for example, version 7.2).

Use the following command to install the PHP graphics drawing library on AL2. For example, if you installed php7.2 from amazon-linux-extras as part of installing the LAMP stack, this command installs version 7.2 of the PHP graphics drawing library.

```
[ec2-user ~]$ sudo yum install php-gd
```

To verify the installed version, use the following command:

```
[ec2-user ~]$ sudo yum list installed php-gd
```

The following is example output:

```
php-gd.x86_64                 7.2.30-1.amzn2           @amzn2extra-php7.2
```

To fix file permissions for the Apache web server

Some of the available features in WordPress require write access to the Apache document root (such as uploading media though the Administration screens). If you have not already done so, apply the following group memberships and permissions (as described in greater detail in the Tutorial: Install a LAMP server on AL2).

1. Grant file ownership of `/var/www` and its contents to the `apache` user.

   ```
   [ec2-user ~]$ sudo chown -R apache /var/www
   ```

2. Grant group ownership of `/var/www` and its contents to the `apache` group.

   ```
   [ec2-user ~]$ sudo chgrp -R apache /var/www
   ```

3. Change the directory permissions of `/var/www` and its subdirectories to add group write permissions and to set the group ID on future subdirectories.

   ```
   [ec2-user ~]$ sudo chmod 2775 /var/www
   [ec2-user ~]$ find /var/www -type d -exec sudo chmod 2775 {} \;
   ```

4. Recursively change the file permissions of `/var/www` and its subdirectories.

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0644 {} \;
```

> **Note**
>
> If you intend to also use WordPress as an FTP server, you'll need more permissive Group settings here. Please review the recommended steps and security settings in WordPress to accomplish this.

5. Restart the Apache web server to pick up the new group and permissions.
   - ```
     [ec2-user ~]$ sudo systemctl restart httpd
     ```

Run the WordPress installation script with AL2

You are ready to install WordPress. The commands that you use depend on the operating system. The commands in this procedure are for use with AL2.

1. Use the **systemctl** command to ensure that the `httpd` and database services start at every system boot.

   ```
   [ec2-user ~]$ sudo systemctl enable httpd && sudo systemctl enable mariadb
   ```

2. Verify that the database server is running.

   ```
   [ec2-user ~]$ sudo systemctl status mariadb
   ```

   If the database service is not running, start it.

   ```
   [ec2-user ~]$ sudo systemctl start mariadb
   ```

3. Verify that your Apache web server (`httpd`) is running.

   ```
   [ec2-user ~]$ sudo systemctl status httpd
   ```

   If the `httpd` service is not running, start it.

   ```
   [ec2-user ~]$ sudo systemctl start httpd
   ```
   ```

4. In a web browser, type the URL of your WordPress blog (either the public DNS address for your instance, or that address followed by the `blog` folder). You should see the WordPress installation script. Provide the information required by the WordPress installation. Choose **Install WordPress** to complete the installation. For more information, see Step 5: Run the Install Script on the WordPress website.

Next steps

After you have tested your WordPress blog, consider updating its configuration.

Use a custom domain name

If you have a domain name associated with your EC2 instance's EIP address, you can configure your blog to use that name instead of the EC2 public DNS address. For more information, see Changing The Site URL on the WordPress website.

Configure your blog

You can configure your blog to use different themes and plugins to offer a more personalized experience for your readers. However, sometimes the installation process can backfire, causing you to lose your entire blog. We strongly recommend that you create a backup Amazon Machine Image (AMI) of your instance before attempting to install any themes or plugins so you can restore your blog if anything goes wrong during installation. For more information, see Create your own AMI.

Increase capacity

If your WordPress blog becomes popular and you need more compute power or storage, consider the following steps:

- Expand the storage space on your instance. For more information, see Amazon EBS Elastic Volumes in the *Amazon EBS User Guide*.
- Move your MySQL database to Amazon RDS to take advantage of the service's ability to scale easily.

Improve network performance of your internet traffic

If you expect your blog to drive traffic from users located around the world, consider AWS Global Accelerator. Global Accelerator helps you achieve lower latency by improving internet traffic

performance between your users' client devices and your WordPress application running on AWS. Global Accelerator uses the AWS global network to direct traffic to a healthy application endpoint in the AWS Region that is closest to the client.

Learn more about WordPress

For information about WordPress, see the WordPress Codex help documentation at http://codex.wordpress.org/.

For more information about troubleshooting your installation, see Common installation problems.

For information about making your WordPress blog more secure, see Hardening WordPress.

For information about keeping your WordPress blog up-to-date, see Updating WordPress.

Help! My public DNS name changed and now my blog is broken

Your WordPress installation is automatically configured using the public DNS address for your EC2 instance. If you stop and restart the instance, the public DNS address changes (unless it is associated with an Elastic IP address) and your blog will not work anymore because it references resources at an address that no longer exists (or is assigned to another EC2 instance). A more detailed description of the problem and several possible solutions are outlined in Changing the Site URL.

If this has happened to your WordPress installation, you might be able to recover your blog with the procedure below, which uses the **wp-cli** command line interface for WordPress.

To change your WordPress site URL with the wp-cli

1. Connect to your EC2 instance with SSH.
2. Note the old site URL and the new site URL for your instance. The old site URL is likely the public DNS name for your EC2 instance when you installed WordPress. The new site URL is the current public DNS name for your EC2 instance. If you are not sure of your old site URL, you can use **curl** to find it with the following command.

   ```
   [ec2-user ~]$ curl localhost | grep wp-content
   ```

   You should see references to your old public DNS name in the output, which will look like this (old site URL in red):

```
<script type='text/javascript' src='http://ec2-52-8-139-223.us-
west-1.compute.amazonaws.com/wp-content/themes/twentyfifteen/js/functions.js?
ver=20150330'></script>
```

3. Download the **wp-cli** with the following command.

```
[ec2-user ~]$ curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/
phar/wp-cli.phar
```

4. Search and replace the old site URL in your WordPress installation with the following command. Substitute the old and new site URLs for your EC2 instance and the path to your WordPress installation (usually `/var/www/html` or `/var/www/html/blog`).

```
[ec2-user ~]$ php wp-cli.phar search-replace 'old_site_url' 'new_site_url' --path=/
path/to/wordpress/installation --skip-columns=guid
```

5. In a web browser, enter the new site URL of your WordPress blog to verify that the site is working properly again. If it is not, see Changing the Site URL and Common installation problems for more information.