Procédure d'installation

WordPress

sur Debian 11 / 12

Composant	Version / Info
OS	Debian 11 (Bullseye) / 12 (Bookworm)
Serveur web	Apache 2.4
PHP	PHP 8.2
Base de données	MariaDB 10.6+
WordPress	6.x (dernière version stable)
HTTPS	Let's Encrypt (Certbot)

1. Introduction

WordPress est le CMS (système de gestion de contenu) le plus utilisé au monde, propulsant plus de 40 % des sites web. Il repose sur une stack LAMP : Linux, Apache, MySQL/MariaDB et PHP.

Ce guide détaille l'installation complète de WordPress sur un serveur Debian, depuis la configuration de la stack LAMP jusqu'à la sécurisation avec HTTPS via Let's Encrypt. À la fin de cette procédure, vous disposerez d'un site WordPress fonctionnel, sécurisé et prêt pour la production.

▶ Architecture déployée

- Debian 11/12 comme système d'exploitation serveur
- Apache 2.4 comme serveur web (avec mod_rewrite pour les permaliens)
- PHP 8.2 avec toutes les extensions requises par WordPress
- MariaDB comme serveur de base de données
- Certbot / Let's Encrypt pour le certificat SSL/TLS gratuit
- WordPress 6.x installé dans /var/www/html/wordpress

2. Prérequis

▶ 2.1 Mise à jour du système

Commencer par mettre à jour tous les paquets du système pour éviter tout conflit de dépendances lors des installations suivantes.

```
$ apt update && apt upgrade -y
```

▶ 2.2 Installation des utilitaires de base

Installer quelques outils essentiels qui seront utiles tout au long de la procédure.

```
$ apt install -y curl wget unzip git nano ufw
```

▶ 2.3 Vérification du nom d'hôte

S'assurer que le nom d'hôte du serveur est correctement défini. Il sera utilisé lors de la configuration du VirtualHost Apache et du certificat SSL.

```
$ hostnamectl set-hostname monsite.exemple.fr
$ hostname -f
```

> ⚠ **Note :** *Remplacer monsite.exemple.fr par votre nom de domaine réel. Le domaine doit pointer vers l'IP du serveur (enregistrement DNS de type A) avant de générer un certificat Let's Encrypt.*

3. Installation d'Apache

▶ 3.1 Installation du paquet Apache

Apache est le serveur web qui recevra les requêtes HTTP/HTTPS et les transmettra à PHP pour le traitement des pages WordPress.

```
$ apt install -y apache2
```

▶ 3.2 Activation des modules nécessaires

WordPress nécessite le module mod_rewrite pour la gestion des permaliens (URLs propres). Le module mod_headers est utile pour les en-têtes de sécurité.

```
$ a2enmod rewrite headers expires ssl
```

▶ 3.3 Démarrage et activation d'Apache

Activer Apache pour qu'il démarre automatiquement au boot du serveur, puis le lancer immédiatement.

```
$ systemctl enable --now apache2
$ systemctl status apache2
```

▶ 3.4 Vérification du pare-feu

Ouvrir les ports HTTP (80) et HTTPS (443) dans le pare-feu UFW pour permettre l'accès au serveur web.

```
$ ufw allow 'Apache Full'
$ ufw enable
$ ufw status
```

> ℹ **Info :** À ce stade, taper l'IP du serveur dans un navigateur doit afficher la page par défaut Apache 'It works!'.

4. Installation de PHP 8.2

▶ 4.1 Ajout du dépôt PHP (Ondřej Surý)

Debian n'inclut pas toujours la dernière version de PHP dans ses dépôts officiels. On ajoute le dépôt maintenu par Ondřej Surý, référence pour les paquets PHP sur Debian/Ubuntu.

```
$ apt install -y lsb-release apt-transport-https ca-certificates
$ wget -qO /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg
```

```
$ echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" >
/etc/apt/sources.list.d/php.list
$ apt update
```

▶ 4.2 Installation de PHP 8.2 et ses extensions

WordPress requiert un ensemble d'extensions PHP pour fonctionner correctement : gestion des images (gd), XML, base de données (mysql), archives (zip), cache (opcache), etc.

```
$ apt install -y php8.2 php8.2-cli php8.2-common php8.2-mysql php8.2-xml \
  php8.2-xmlrpc php8.2-curl php8.2-gd php8.2-imagick php8.2-cli \
  php8.2-imap php8.2-mbstring php8.2-opcache php8.2-soap \
  php8.2-zip php8.2-intl libapache2-mod-php8.2
```

▶ 4.3 Vérification de PHP

Vérifier que PHP 8.2 est bien installé et actif.

```
$ php -v
$ php -m | grep -E 'mysql|gd|curl|zip|mbstring|xml'
```

▶ 4.4 Optimisation de PHP pour WordPress

Ajuster les paramètres PHP pour améliorer les performances et permettre l'upload de fichiers volumineux (thèmes, plugins, médias).

```
$ nano /etc/php/8.2/apache2/php.ini
```

Modifier ou vérifier les valeurs suivantes dans le fichier :

```
# Taille maximale des fichiers uploadés
upload_max_filesize = 64M
post_max_size = 64M
# Mémoire allouée à PHP
memory_limit = 256M
# Temps d'exécution maximum
max_execution_time = 300
# Activer l'OPcache pour de meilleures performances
opcache.enable = 1
opcache.memory_consumption = 128
$ systemctl restart apache2
```

5. Installation et configuration de MariaDB

▶ 5.1 Installation de MariaDB

MariaDB est un fork communautaire de MySQL, entièrement compatible et recommandé pour WordPress. C'est la base de données qui stockera tout le contenu du site (articles, pages, utilisateurs, paramètres).

```
$ apt install -y mariadb-server mariadb-client
$ systemctl enable --now mariadb
$ systemctl status mariadb
```

▶ 5.2 Sécurisation de MariaDB

Le script mysql_secure_installation permet de sécuriser l'installation MariaDB en définissant un mot de passe root, en supprimant les utilisateurs anonymes et en désactivant l'accès root à distance.

```
$ mysql_secure_installation
```

Répondre aux questions de l'assistant :

- Enter current password for root : appuyer sur Entrée (vide par défaut)
- Switch to unix_socket authentication : n

- Change the root password : y, puis définir un mot de passe fort
- Remove anonymous users : y
- Disallow root login remotely : y
- Remove test database : y
- Reload privilege tables : y

▶ 5.3 Création de la base de données WordPress

Créer une base de données dédiée à WordPress ainsi qu'un utilisateur MariaDB spécifique. Éviter d'utiliser l'utilisateur root pour les applications web.

```
$ mysql -u root -p
# Dans le shell MariaDB, exécuter les commandes suivantes :
CREATE DATABASE wordpress CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'MotDePasseForte123!';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

> ⚠ **Note :** *Remplacer wpuser et MotDePasseForte123! par vos propres valeurs. Conserver ces informations, elles seront nécessaires lors de la configuration de WordPress.*

6. Téléchargement et installation de WordPress

▶ 6.1 Téléchargement de la dernière version

Télécharger l'archive officielle WordPress depuis wordpress.org, puis l'extraire dans le répertoire web du serveur.

```
$ cd /tmp
$ wget https://wordpress.org/latest.tar.gz
$ tar -xzf latest.tar.gz
$ mv wordpress /var/www/html/wordpress
```

▶ 6.2 Configuration des permissions

Attribuer les permissions correctes sur le dossier WordPress. Apache s'exécute sous l'utilisateur www-data, qui doit être propriétaire des fichiers pour permettre les mises à jour et l'upload de médias.

```
$ chown -R www-data:www-data /var/www/html/wordpress
$ find /var/www/html/wordpress -type d -exec chmod 755 {} \;
$ find /var/www/html/wordpress -type f -exec chmod 644 {} \;
```

▶ 6.3 Création du fichier de configuration WordPress

Copier le fichier de configuration exemple et l'adapter à votre base de données.

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

Modifier les lignes suivantes avec vos informations de base de données :

```
# Informations de connexion à la base de données
define( 'DB_NAME', 'wordpress' );
define( 'DB_USER', 'wpuser' );
define( 'DB_PASSWORD', 'MotDePasseForte123!' );
define( 'DB_HOST', 'localhost' );
define( 'DB_CHARSET', 'utf8mb4' );
```

▶ 6.4 Génération des clés de sécurité WordPress

WordPress utilise des clés secrètes pour sécuriser les cookies et sessions. Les générer depuis l'API officielle WordPress et les coller dans wp-config.php en remplacement des lignes existantes.

```
$ curl -s https://api.wordpress.org/secret-key/1.1/salt/
```

> **Info :** Copier-coller le résultat dans wp-config.php en remplaçant le bloc put your unique phrase here existant.

7. Configuration du VirtualHost Apache

7.1 Création du VirtualHost

Créer un VirtualHost Apache dédié à WordPress. Cela permet d'héberger plusieurs sites sur le même serveur et de configurer précisément les options pour WordPress.

```
$ nano /etc/apache2/sites-available/wordpress.conf
```

Contenu du fichier (adapter le domaine et les chemins) :

```
<VirtualHost *:80>
    ServerName monsite.exemple.fr
    ServerAlias www.monsite.exemple.fr
    DocumentRoot /var/www/html/wordpress
    ErrorLog ${APACHE_LOG_DIR}/wordpress-error.log
    CustomLog ${APACHE_LOG_DIR}/wordpress-access.log combined

    <Directory /var/www/html/wordpress>
        Options FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

7.2 Activation du VirtualHost

Activer le VirtualHost WordPress et désactiver le site par défaut d'Apache.

```
$ a2ensite wordpress.conf
$ a2dissite 000-default.conf
$ apache2ctl configtest
$ systemctl reload apache2
```

> **Info :** La commande apache2ctl configtest doit retourner 'Syntax OK' avant de recharger Apache.

7.3 Configuration du fichier .htaccess

Le fichier .htaccess est utilisé par WordPress pour gérer les permaliens. Créer un fichier vide avec les bonnes permissions pour que WordPress puisse l'écrire.

```
$ touch /var/www/html/wordpress/.htaccess
$ chown www-data:www-data /var/www/html/wordpress/.htaccess
$ chmod 644 /var/www/html/wordpress/.htaccess
```

8. Configuration HTTPS avec Let's Encrypt

8.1 Installation de Certbot

Certbot est l'outil officiel de l'EFF pour obtenir et renouveler automatiquement des certificats SSL/TLS gratuits auprès de Let's Encrypt.

```
$ apt install -y certbot python3-certbot-apache
```

▶ 8.2 Obtention du certificat SSL

Certbot va automatiquement détecter la configuration Apache, obtenir un certificat pour votre domaine et reconfigurer Apache pour utiliser HTTPS. Le domaine doit être accessible depuis Internet sur le port 80 pour la validation Let's Encrypt.

```
$ certbot --apache -d monsite.exemple.fr -d www.monsite.exemple.fr
```

Répondre aux questions de Certbot :

- Entrer une adresse email pour les notifications d'expiration
- Accepter les conditions d'utilisation (A)
- Partager ou non l'email avec l'EFF (N recommandé)
- Certbot obtient le certificat et reconfigure Apache automatiquement

> ⚠ **Note :** *Le domaine doit pointer vers l'IP du serveur (enregistrement DNS A) et le port 80 doit être accessible depuis Internet pour que Let's Encrypt puisse valider le domaine.*

▶ 8.3 Vérification du renouvellement automatique

Let's Encrypt émet des certificats valables 90 jours. Certbot installe automatiquement une tâche cron ou un timer systemd pour renouveler les certificats avant leur expiration. Vérifier que le renouvellement fonctionne.

```
$ certbot renew --dry-run
$ systemctl status certbot.timer
```

> ℹ **Info :** Le renouvellement automatique se fait deux fois par jour. Aucune intervention manuelle n'est nécessaire.

9. Finalisation de l'installation WordPress

▶ 9.1 Assistant d'installation via le navigateur

Ouvrir un navigateur et se rendre sur le domaine configuré pour lancer l'assistant d'installation web de WordPress.

```
$ https://monsite.exemple.fr
```

L'assistant WordPress demande les informations suivantes :

- Titre du site : nom de votre site
- Identifiant : nom d'utilisateur administrateur (éviter 'admin')
- Mot de passe : générer un mot de passe fort
- Votre e-mail : adresse email de l'administrateur
- Visibilité pour les moteurs de recherche : décocher pendant le développement
- Cliquer sur 'Installer WordPress'

> ℹ **Info :** WordPress créera automatiquement toutes les tables nécessaires dans la base de données lors de cette étape.

▶ 9.2 Accès au tableau de bord

Une fois l'installation terminée, accéder au tableau de bord d'administration WordPress :

```
$ https://monsite.exemple.fr/wp-admin
```

10. Optimisation et sécurisation

▶ 10.1 Sécurisation du fichier wp-config.php

Empêcher l'accès direct au fichier de configuration WordPress depuis le web. Ajouter ces règles dans le .htaccess ou dans le VirtualHost Apache.

```
$ nano /var/www/html/wordpress/.htaccess
# Protéger wp-config.php
<Files wp-config.php>
    Order allow,deny
    Deny from all
</Files>
# Désactiver la navigation dans les répertoires
Options -Indexes
# Bloquer l'accès à xmlrpc.php si non utilisé
<Files xmlrpc.php>
    Order Deny,Allow
    Deny from all
</Files>
```

▶ 10.2 Déplacer wp-config.php hors de la racine web

Pour une sécurité maximale, déplacer wp-config.php un niveau au-dessus du DocumentRoot. WordPress le détecte automatiquement.

```
$ mv /var/www/html/wordpress/wp-config.php /var/www/html/wp-config.php
$ chown www-data:www-data /var/www/html/wp-config.php
```

▶ 10.3 Configuration des en-têtes de sécurité Apache

Ajouter des en-têtes HTTP de sécurité pour protéger contre les attaques XSS, clickjacking et autres vulnérabilités web courantes.

```
$ nano /etc/apache2/sites-available/wordpress.conf
```

Ajouter dans le bloc <VirtualHost *:443> :

```
Header always set X-Content-Type-Options nosniff
Header always set X-Frame-Options SAMEORIGIN
Header always set X-XSS-Protection "1; mode=block"
Header always set Referrer-Policy "strict-origin-when-cross-origin"
$ systemctl reload apache2
```

▶ 10.4 Permissions de sécurité renforcées

Renforcer les permissions sur les fichiers sensibles de WordPress.

```
$ chmod 600 /var/www/html/wp-config.php
$ chown -R www-data:www-data /var/www/html/wordpress/wp-content
```

▶ 10.5 Désactivation de l'éditeur de fichiers WordPress

Par sécurité, désactiver l'éditeur de fichiers intégré au tableau de bord WordPress pour empêcher toute modification de thèmes et plugins directement depuis l'interface.

Ajouter cette ligne dans wp-config.php :

```
$ nano /var/www/html/wp-config.php
define( 'DISALLOW_FILE_EDIT', true );
```

11. Optimisation des performances

▶ 11.1 Activation du cache OPcache PHP

OPcache améliore drastiquement les performances PHP en compilant et mettant en cache le bytecode PHP. Vérifier qu'il est bien activé.

```
$ php -r "echo opcache_get_status() ? 'OPcache actif' : 'OPcache inactif';"
```

▶ 11.2 Activation de la compression Gzip

Activer la compression Gzip dans Apache pour réduire la taille des ressources transférées et améliorer les temps de chargement.

```
$ a2enmod deflate
$ systemctl reload apache2
```

▶ 11.3 Plugins de cache recommandés

Depuis le tableau de bord WordPress (Extensions > Ajouter), installer un plugin de cache pour améliorer les performances :

- W3 Total Cache : solution complète (cache de page, objet, navigateur)
- WP Super Cache : simple et efficace, génère des fichiers HTML statiques
- LiteSpeed Cache : idéal si l'hébergeur utilise LiteSpeed
- WP Rocket : solution premium très performante (payant)

12. Sauvegarde et maintenance

▶ 12.1 Sauvegarde de la base de données

Créer un dump complet de la base de données WordPress. À automatiser via cron pour des sauvegardes régulières.

```
$ mysqldump -u wpuser -p wordpress > /backup/wordpress-db-$(date +%Y%m%d).sql
$ gzip /backup/wordpress-db-$(date +%Y%m%d).sql
```

▶ 12.2 Sauvegarde des fichiers

Sauvegarder le répertoire WordPress, notamment le dossier wp-content qui contient thèmes, plugins et médias uploadés.

```
$ tar -czf /backup/wordpress-files-$(date +%Y%m%d).tar.gz /var/www/html/wordpress/
```

▶ 12.3 Script de sauvegarde automatique

Créer un script de sauvegarde automatique et le planifier via cron.

```
$ nano /usr/local/bin/backup-wordpress.sh
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR=/backup/wordpress
mkdir -p $BACKUP_DIR
# Sauvegarde BDD
mysqldump -u wpuser -pMotDePasseForte123! wordpress | gzip >
$BACKUP_DIR/db_$DATE.sql.gz
# Sauvegarde fichiers
tar -czf $BACKUP_DIR/files_$DATE.tar.gz /var/www/html/wordpress/wp-content/
# Suppression des sauvegardes de plus de 30 jours
find $BACKUP_DIR -mtime +30 -delete
echo "Sauvegarde terminée : $DATE"
$ chmod +x /usr/local/bin/backup-wordpress.sh
```

Ajouter une tâche cron pour exécuter le script tous les jours à 2h du matin :

```
$ crontab -e
0 2 * * * /usr/local/bin/backup-wordpress.sh >> /var/log/backup-wordpress.log 2>&1
```

▶ 12.4 Mises à jour WordPress

Les mises à jour WordPress (core, thèmes, plugins) se gèrent depuis le tableau de bord. Il est également possible de les effectuer en ligne de commande avec WP-CLI.

```
$ curl -O
https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
$ chmod +x wp-cli.phar && mv wp-cli.phar /usr/local/bin/wp
# Mettre à jour WordPress core
$ wp --allow-root --path=/var/www/html/wordpress core update
# Mettre à jour tous les plugins
$ wp --allow-root --path=/var/www/html/wordpress plugin update --all
# Mettre à jour tous les thèmes
$ wp --allow-root --path=/var/www/html/wordpress theme update --all
```

13. Vérifications finales

13.1 Vérification des services

```
$ systemctl status apache2
$ systemctl status mariadb
$ systemctl status certbot.timer
```

13.2 Vérification des logs

```
$ tail -f /var/log/apache2/wordpress-error.log
$ tail -f /var/log/apache2/wordpress-access.log
```

13.3 Test de la configuration Apache

```
$ apache2ctl configtest
$ apache2ctl -S
```

13.4 Test du certificat SSL

Vérifier la validité et la date d'expiration du certificat SSL.

```
$ certbot certificates
$ openssl s_client -connect monsite.exemple.fr:443 -brief
```

13.5 Vérification des permissions

```
$ ls -la /var/www/html/wordpress/
$ ls -la /var/www/html/wp-config.php
```

14. Dépannage courant

Erreur 403 Forbidden

Vérifier les permissions des fichiers et dossiers, et s'assurer que AllowOverride All est bien présent dans le VirtualHost.

```
$ chown -R www-data:www-data /var/www/html/wordpress
$ apache2ctl configtest && systemctl reload apache2
```

Erreur 500 Internal Server Error

Consulter les logs d'erreur Apache et vérifier la syntaxe du .htaccess.

```
$ tail -50 /var/log/apache2/wordpress-error.log
$ php -l /var/www/html/wordpress/wp-config.php
```

▶ Connexion à la base de données impossible

Vérifier les identifiants dans wp-config.php et tester la connexion MariaDB directement.

```
$ mysql -u wpuser -p wordpress -e 'SHOW TABLES;'
```

▶ Boucle de redirection infinie

Souvent causé par une mauvaise configuration HTTPS. Ajouter dans wp-config.php :

```
define('FORCE_SSL_ADMIN', true);
if (strpos($_SERVER['HTTP_X_FORWARDED_PROTO'], 'https') !== false)
    $_SERVER['HTTPS'] = 'on';
```

15. Conclusion

WordPress est maintenant installé, configuré et sécurisé sur votre serveur Debian. Voici un récapitulatif de ce qui a été mis en place :

- Stack LAMP complète : Apache 2.4, PHP 8.2, MariaDB
- WordPress 6.x avec permissions sécurisées
- HTTPS avec certificat Let's Encrypt (renouvellement automatique)
- En-têtes de sécurité HTTP
- OPcache et compression Gzip pour les performances
- Script de sauvegarde automatique quotidien
- WP-CLI pour la gestion en ligne de commande

> ℹ **Info :** Documentation officielle WordPress : https://developer.wordpress.org | Documentation Let's Encrypt : https://letsencrypt.org/docs/

Procédure rédigée pour WordPress 6.x sur Debian 11/12 – 2025