Irisolaris Greentariff — Documentation

DOC-IRIS-001 | Version A | 20/02/2026

Historique des modifications

Rédacteur / Validateur	Version	Date	Description
Daniel Blévin	A	20/02/2026	Création du document

Diffusion de la procédure

Service DSI

Périmètre

Irisolaris Greentariff

Objet

Documentation technique — Serveur WordPress et plugin Irisolaris Map

Contents

0.1 Vue d'ensemble de la stack

Cette documentation couvre trois couches de la stack de production. Chaque section dans la barre latérale correspond à l'une de ces couches — de l'infrastructure de base jusqu'aux fonctionnalités spécifiques du plugin.

Couche	Description
Plugin Irisolaris Map	La couche supérieure et le sujet principal de cette documentation : le plugin WordPress personnalisé qui fournit la carte interactive des centrales photovoltaïques, la vérification d'éligibilité, l'intégration CRM HubSpot et les outils d'import en masse.
Stack WordPress	La couche intermédiaire : configuration du cœur WordPress, thème Blocksy, 16 plugins complémentaires (sécurité, cache, SEO, sauvegardes) et la stack de cache multi-couches (Redis, WP Super Cache, en-têtes HTTP).
Plateforme et infrastructure	La couche de base : provisionnement du serveur Linux, stack LEMP (nginx, MariaDB, PHP, Redis), pare-feu et durcissement de la sécurité, certificats SSL et maintenance continue du serveur.

0.2 Plugin Irisolaris Map

Irisolaris Map est un plugin WordPress personnalisé qui propulse une carte interactive des centrales photovoltaïques pour la plateforme **irisolaris-greentariff.com**. Il permet aux utilisateurs de vérifier leur éligibilité à l'autoconsommation collective en fonction de leur proximité géographique aux centrales solaires.

- **Carte interactive** : centrales photovoltaïques affichées sur une carte MapLibre GL JS avec clustering, popups, filtrage et liste des centrales en barre latérale
- **Vérification d'éligibilité** : contrôle de proximité par adresse utilisant la classification de densité de population INSEE avec paliers de distance
- **Intégration CRM HubSpot** : soumissions du formulaire d'adhésion transmises à HubSpot avec métadonnées de la centrale enrichies côté serveur
- **Gestion des centrales** : CRUD complet via l'administration WordPress, plus import en masse CSV/XLSX avec logique d'upsert
- **Grille de densité** : import et consultation des données de densité communale INSEE pour les règles de distance d'éligibilité
- **Conformité RGPD** : expiration automatique des prospects et nettoyage quotidien par cron

Construit sur un backend PHP avec un frontend JavaScript modulaire ES6+ (Webpack). Dépendances clés : MapLibre GL JS (rendu cartographique), PhpSpreadsheet (import de fichiers), Base Adresse Nationale (géocodage), HubSpot Forms API (CRM) et OpenFreeMap (tuiles). Voir **Stack technique du plugin** pour plus de détails.

Présentation

Vue d'ensemble du plugin, contexte du projet et stack technique (MapLibre, HubSpot, BAN API, dépendances Composer).

Introduction | **Stack technique du plugin**

Architecture

Vue d'ensemble des modules, séquence de démarrage, modèle de données, API REST, intégrations WordPress et JavaScript frontend.

Vue d'ensemble | **Classes** | **Modèle de données** | **API REST**

Fonctionnalités

Carte interactive, vérification d'éligibilité, formulaire d'adhésion, import de centrales, grille de densité et gestion RGPD des prospects.

Carte | **Éligibilité** | **Formulaire d'adhésion** | **Import**

Opérations

Processus de build, configuration, cache et performance.

Build | **Configuration** | **Cache**

Évolution

Architecture cible de l'orchestrateur ACC avec Fastify, BullMQ et intégrations d'API étendues.

Architecture cible | **Orchestrateur Node.js** | **API externes**

0.3 Stack WordPress

L'installation WordPress qui héberge le plugin Irisolaris Map, ainsi que les plugins complémentaires et la stratégie de cache multi-couches.

- **WordPress** : locale française, permaliens optimisés et page d'accueil statique
- **Thème** : Blocksy avec un thème enfant actif pour les personnalisations
- **Plugins de sécurité** : NinjaFirewall (WAF), Disable REST API (liste blanche des endpoints requise), WP Anti-Clickjack
- **Plugins de performance** : Redis Object Cache, WP Super Cache (cache statique pleine page), WP-Optimize (nettoyage de la base de données)
- **Autres plugins** : Complianz (RGPD), Yoast SEO, Site Kit (analytics), UpdraftPlus (sauvegardes automatisées), GreenShift + Stackable (constructeurs de pages)

Stack WordPress

Infrastructure serveur, paramètres du site WordPress, thème actif et plugins installés.

Environnement | **Plugins actifs** | **Stack de cache**

0.4 Plateforme et infrastructure

La couche de base — provisionnement du serveur Linux, installation de la stack LEMP (nginx, MariaDB, PHP, Redis), durcissement de la sécurité et maintenance continue.

Configuration initiale du serveur

Première connexion, gestion des utilisateurs, authentification par clé SSH et désactivation de la connexion par mot de passe pour une sécurité renforcée.

Première connexion | **Vérifier le statut utilisateur** | **Désactiver la connexion par mot de passe**

Mesures de sécurité

Configuration du pare-feu avec UFW, mise en place de Fail2ban et pratiques complètes de durcissement du serveur.

Ajouter un pare-feu | **Ajouter Fail2ban** | **Durcissement du serveur**

Configuration du domaine

Instructions pour faire pointer un domaine vers votre serveur avec une configuration DNS appropriée utilisant les enregistrements A et CNAME.

Configuration du domaine

Installation de la stack LEMP

Installation complète de Linux, Nginx, MariaDB et PHP avec optimisation et durcissement de la sécurité.

Installer la stack d'hébergement | **Configurer le mail** | **Optimiser la stack d'hébergement**

Déploiement WordPress

Installation, durcissement de la sécurité et techniques d'optimisation des performances de WordPress.

Installation de WordPress | **Durcissement de WordPress** | **Optimiser WordPress**

Guide de maintenance serveur

Un guide complet pour la maintenance et l'optimisation de votre serveur WordPress, organisé par fréquence de maintenance et par composant.

Voir le guide

0.5 Stack serveur

Cette implémentation utilise la stack LEMP (Linux, Nginx, MariaDB, PHP) comme fondation technologique web, offrant une plateforme robuste et performante pour l'hébergement de sites WordPress.

LEMP

La stack LEMP est une solution de services web populaire qui combine la fiabilité de Linux, les performances de Nginx (prononcé « Engine-X »), la robustesse de MariaDB et la flexibilité de PHP. Cette combinaison fournit un environnement efficace et sécurisé pour l'hébergement de sites WordPress avec d'excellentes caractéristiques de performance et de sécurité.

- **Système d'exploitation** : Ubuntu LTS — choisi pour son support à long terme, ses excellentes fonctionnalités de sécurité et sa compatibilité optimale avec la stack LEMP
- **Nginx** : serveur web qui gère les requêtes HTTP, sert le contenu statique et transmet les requêtes dynamiques à PHP
- **MariaDB** : serveur de base de données qui stocke le contenu et les données de configuration de WordPress
- **PHP** : langage de script côté serveur utilisé pour traiter le contenu dynamique de WordPress
- **Redis** : stockage de structures de données en mémoire utilisé pour le cache et l'optimisation des performances

Pour les versions spécifiques, voir **Environnement WordPress**.

Partie I

Plugin Irisolaris Map

1 Présentation

1.1 Introduction

Qu'est-ce qu'Irisolaris Map ?

Irisolaris Map est un plugin WordPress qui alimente une carte interactive de centrales photovoltaïques pour la plateforme **Irisolaris Greentariff**. Il permet aux utilisateurs finaux de vérifier leur éligibilité à l'autoconsommation collective (ACC) en fonction de leur proximité géographique avec les centrales solaires.

Fonctionnalités principales

- **Carte interactive** — Affiche plus de 2 000 centrales photovoltaïques sur une carte MapLibre GL JS avec regroupement, popups et filtrage
- **Vérification d'éligibilité** — Détermine l'éligibilité de l'utilisateur en fonction de la proximité de son adresse et de la classification de densité de population INSEE
- **Intégration CRM HubSpot** — Transmet les demandes d'adhésion qualifiées à HubSpot avec les métadonnées de la centrale enrichies côté serveur (21 champs)
- **Gestion des centrales** — CRUD complet via l'administration WordPress, plus import CSV/XLSX en masse avec logique d'upsert
- **Gestion de la grille de densité** — Import et interrogation des données de densité communale INSEE pour les règles d'éligibilité
- **Conformité RGPD** — Expiration automatique des prospects et tâche cron de nettoyage quotidienne

Contexte du projet

Champ	Valeur
Nom du plugin	Irisolaris Map
Version (production)	1.1.20 (en date de février 2026)
Auteur	Daniel Blévin
Licence	GPLv2 ou ultérieure
URL de production	`https://irisolaris-greentariff.com`

Périmètre du plugin

Dans le périmètre :

- Affichage de la carte interactive et visualisation des centrales
- Gestion des données des centrales (CRUD + import en masse)
- Vérification d'éligibilité basée sur l'adresse avec niveaux de densité
- Capture de prospects et soumission vers le CRM HubSpot
- Gestion des données de la grille de densité (classification INSEE)

- Rétention des prospects conforme au RGPD et nettoyage automatique

Hors périmètre (géré par d'autres systèmes) :

- Configuration du CRM HubSpot et automatisation des workflows
- Intégration de l'API Symphonics/Enedis (prévue — voir **Architecture cible**)
- Génération de contrats
- Authentification/inscription des utilisateurs (utilise les rôles natifs de WordPress)

Structure de la documentation

Cette documentation est organisée en six sections :

1. **Présentation** — Vue d'ensemble du projet et pile technologique
2. **Architecture** — Architecture technique, modèle de données, API et intégrations WordPress
3. **Fonctionnalités** — Documentation détaillée des fonctionnalités avec flux d'exécution
4. **Déploiement** — Processus de build et configuration
5. **Performance & Cache** — Mise en cache et performance
6. **Évolution** — Architecture cible et développements prévus

1.2 Pile technologique du plugin

Pour l'environnement d'hébergement WordPress, le thème et l'écosystème de plugins, voir **Environnement WordPress**.

Le plugin repose sur un backend PHP et un frontend JavaScript, s'appuyant sur un petit ensemble de packages Composer et trois services externes. Les sections ci-dessous listent chaque dépendance et son rôle.

PHP

- **Namespace :** `Irisolaris\Map\{Core, Admin, API, Frontend}`
- **Autoloading :** PSR-4 via Composer
- **Prérequis minimum :** WordPress 6.0, PHP 7.4

JavaScript (Frontend)

Le frontend repose sur une seule bibliothèque externe pour le rendu cartographique ; tout le reste est du code personnalisé.

Bibliothèque	Version	Source	Rôle
MapLibre GL JS	2.4.0	CDN (unpkg.com)	Rendu cartographique interactif

Le frontend est construit à partir de fichiers source modulaires ES6+ (`assets/src/js/`) compilés via Webpack en un seul bundle (`frontend.js`) contenant les classes `IrisolarisMap` et `IrisolarisGeocoder`. Voir **Frontend JavaScript** pour plus de détails.

Dépendances PHP (Composer)

PhpSpreadsheet est la seule dépendance directe ; les autres packages sont transitifs.

Package	Version	Rôle
`phpoffice/phpspreadsheet`	1.29.11	Lecture de fichiers XLSX/XLS/CSV pour l'import de centrales
`ezyang/htmlpurifier`	4.18.0	Assainissement HTML (dépendance de phpspreadsheet)
`maennchen/zipstream-php`	3.1.2	Streaming ZIP (dépendance de phpspreadsheet)
`markbaker/complex`	3.0.2	Mathématiques de nombres complexes (dépendance de phpspreadsheet)
`markbaker/matrix`	3.0.1	Mathématiques matricielles (dépendance de phpspreadsheet)
`composer/pcre`	3.3.2	Wrappers PCRE typés

Services externes

Le plugin communique avec trois services externes à l'exécution, dont aucun ne nécessite de clé API.

Service	URL	Authentification	Rôle
Base Adresse Nationale (BAN)	`api-adresse.data.gouv.fr`	Aucune (public)	Géocodage gouvernemental français
HubSpot Forms API	`api.hsforms.com`	Portal + Form ID dans l'URL	Soumission de formulaire CRM
OpenFreeMap	`tiles.openfreemap.org`	Aucune (public)	Tuiles cartographiques (raster + vecteur)

2 Architecture

2.1 Vue d'ensemble de l'architecture

Vue d'ensemble des modules

Le plugin suit une architecture PHP modulaire organisée en quatre espaces de noms :

```
Irisolaris\Map\
├── Core\Plugin        → Orchestrateur principal (CPT/taxonomie, scripts, AJAX)
├── Admin\Admin        → Interface d'administration (menus, meta boxes, imports, paramètres)
├── API\API            → REST API (GeoJSON, géocodage, éligibilité, HubSpot)
├── API\DensityController → Endpoints REST pour les recherches de densité
└── Frontend\Frontend  → Rendu frontend (shortcodes, chargement des scripts, GeoJSON)
```

Point d'entrée : `irisolaris-map.php` — fichier d'amorçage procédural qui initialise tous les modules.

Responsabilités des composants

Module	Responsabilité
`Core\Plugin`	Enregistrement des CPT/taxonomies, chargement des scripts, gestionnaires AJAX éligibilité/géocodage, création de leads, notifications par e-mail
`Admin\Admin`	Menus d'administration, meta boxes, import de centrales (CSV/XLSX), import de densité, cron RGPD, paramètres, colonnes/filtres admin, opérations en masse
`API\API`	Endpoints REST pour GeoJSON, proxy de géocodage, vérification d'éligibilité, soumission de formulaire HubSpot, gestion du cache
`API\DensityContr`	Endpoints REST pour la recherche de densité (par codgeo, par code postal)
`Frontend\Fronten`	Chargement des scripts frontend, gestionnaire de shortcode, génération GeoJSON

Classes de support

Classe	Fichier	Rôle
`Irisolaris_Map_Activator`	`includes/class-irisolaris-map-activator.php`	Création des tables de base de données, options par défaut, termes de taxonomie par défaut, capacités de rôle
`Irisolaris_Map_Uninstaller`	`includes/class-irisolaris-map-uninstaller.php`	Nettoyage : suppression des tables, retrait des capacités, suppression optionnelle des données de densité

Séquence d'initialisation

```
1. plugins_loaded → irisolaris_map_init()
   ├── new Core\Plugin()     → Plugin::init()
   │   ├── add_action('init', 'register_post_types')
   │   ├── add_action('init', 'register_taxonomies')
   │   ├── add_action('wp_enqueue_scripts', 'register_scripts')
   │   ├── add_action('admin_enqueue_scripts', 'register_admin_scripts')
   │   └── AJAX handlers (eligibility, geocode)
   ├── new Admin\Admin()     → Admin::init()       [if is_admin()]
   │   ├── add_action('admin_menu', 'add_admin_menu')
   │   ├── add_action('add_meta_boxes', 'add_meta_boxes')
   │   ├── add_action('save_post_irisolaris_plant', 'save_plant_meta')
   │   ├── Admin columns/filters/sorting hooks
   │   ├── wp_schedule_event('daily', 'irisolaris_map_gdpr_cleanup')
```

```
   │   └── 16 AJAX handlers (import, density, cache, bulk ops)
   ├── new Frontend\Frontend() → Frontend::init()
   │   ├── add_action('wp_enqueue_scripts', 'enqueue_scripts')
   │   ├── add_shortcode('irisolaris_map', 'map_shortcode')
   │   └── add_filter('irisolaris_map_plants_data', 'filter_plants_data')
   └── new API\API()         → API::init()
       ├── add_action('rest_api_init', 'register_routes')
       └── Cache invalidation hooks (save_post, delete_post, set_object_terms)

2. init → irisolaris_map_register_blocks()
   └── register_block_type_from_metadata('irisolaris-map/map')
```

Graphe de dépendances

```
Browser
  │
  ├── MapLibre GL JS (CDN) ─── Map rendering
  ├── BAN API ──────────────── Address geocoding (direct)
  │
  └── WordPress
      ├── REST API
      │   ├── /plants ──────── GeoJSON (cached via transients)
      │   ├── /density/{code} — INSEE density lookup
      │   ├── /geocode ─────── BAN API proxy
      │   ├── /check-eligibility
      │   └── /submit-join-form ──► HubSpot Forms API
      │
      ├── AJAX (admin)
      │   ├── Import (plants) ─── PhpSpreadsheet
      │   ├── Import (density)
      │   └── Bulk operations
      │
      └── Cron
          └── GDPR cleanup (daily)
```

Décisions architecturales clés

1. **Cache basé sur les transients** — Les réponses GeoJSON sont mises en cache sous forme de transients WordPress avec invalidation par incrémentation de version, plutôt que d'utiliser une couche de cache dédiée
2. **Soumission HubSpot côté serveur** — Les métadonnées sensibles des centrales (identifiant chantier, SPV, PRM) ne sont jamais exposées via l'API publique ; elles sont récupérées côté serveur lors de la soumission du formulaire
3. **Éligibilité côté client** — La vérification initiale d'éligibilité (distance + niveaux de densité) s'exécute dans le navigateur pour un retour instantané ; le serveur effectue une vérification définitive lors de la soumission du formulaire
4. **Autoloading PSR-4** — Les classes PHP utilisent l'autoloading Composer avec l'espace de noms `Irisolaris\Map`
5. **JS modulaire compilé en bundle** — Le code source frontend est organisé en fichiers ES6+ modulaires (`assets/src/js/`), compilés via Webpack en un unique bundle `frontend.js`

2.2 Classes et modules

Cette page fournit un inventaire détaillé de chaque classe PHP et module JavaScript du plugin, ainsi que leurs responsabilités. Pour une vue de plus haut niveau sur l'articulation de ces composants, consultez la **Vue d'ensemble de l'architecture**.

Classes PHP

Structure des espaces de noms

```
Irisolaris\Map\
├── Core\
│   └── Plugin.php
├── Admin\
│   └── Admin.php
├── API\
│   ├── API.php
│   └── DensityController.php
└── Frontend\
    └── Frontend.php
```

Core\Plugin

Fichier : `src/Core/Plugin.php`

La classe orchestratrice principale. Gère l'initialisation du plugin, l'enregistrement des hooks WordPress et les gestionnaires AJAX principaux.

Responsabilités :

- Enregistrement des types de contenu personnalisés (`irisolaris_plant`, `irisolaris_lead`)
- Enregistrement des taxonomies (`irisolaris_status`, `irisolaris_region`)
- Chargement des scripts frontend et admin
- Gestionnaire AJAX de vérification d'éligibilité (avec règles par niveaux basées sur la densité)
- Gestionnaire AJAX de géocodage
- Création de leads (CPT WordPress)
- Notification par e-mail lors de nouveaux leads

Admin\Admin

Fichier : `src/Admin/Admin.php`

La classe la plus volumineuse du plugin. Gère toutes les fonctionnalités côté administration.

Responsabilités :

- Pages de menu d'administration (5 pages : importateur, paramètres, grille de densité, importateur de densité, liste de densité)
- Meta boxes pour l'édition des centrales et des leads (3 meta boxes)
- Import de centrales depuis CSV/XLSX (AJAX en 3 phases : initialisation, traitement, finalisation)
- Import de grille de densité (AJAX en 3 phases)
- Suppression en masse de centrales (AJAX en 3 phases)
- Planification du cron RGPD et nettoyage des leads
- Colonnes d'administration, filtres et colonnes triables pour les listes de centrales/leads
- Page de paramètres avec l'API WordPress Options
- Gestion du cache (purge manuelle)

Gestionnaires AJAX d'administration (16) :

Gestionnaire	Rôle
`irisolaris_import_init`	Import de centrales : validation du fichier, comptage des lignes
`irisolaris_import_process`	Import de centrales : traitement par lots (200 lignes)
`irisolaris_import_finalize`	Import de centrales : journalisation, nettoyage des fichiers temporaires
`irisolaris_clear_import_history`	Purge de la table du journal d'import
`irisolaris_clear_cache`	Purge du cache transient GeoJSON
`irisolaris_remove_plants_init`	Suppression en masse de centrales : initialisation
`irisolaris_remove_plants_process`	Suppression en masse de centrales : suppression par lots
`irisolaris_remove_plants_finalize`	Suppression en masse de centrales : finalisation
`irisolaris_import_density_init`	Import de densité : initialisation
`irisolaris_import_density_process`	Import de densité : traitement par lots
`irisolaris_import_density_finalize`	Import de densité : finalisation
`irisolaris_clear_density_import_history`	Purge des journaux d'import de densité
`irisolaris_get_density_details`	Obtenir un enregistrement de densité par codgeo
`irisolaris_delete_all_density`	Purge de la table de densité

API\API

Fichier : `src/API/API.php`

Gère tous les endpoints de l'API REST et la logique d'invalidation du cache.

Responsabilités :

- 7 endpoints REST (voir **API REST**)
- Génération GeoJSON avec cache par transients et invalidation par incrémentation de version
- Proxy de géocodage vers BAN API
- Endpoint de vérification d'éligibilité
- Soumission de formulaire HubSpot avec enrichissement côté serveur
- En-têtes HTTP de cache (`Cache-Control`, `ETag`)
- Hooks d'invalidation du cache sur `save_post`, `delete_post`, `set_object_terms`

API\DensityController

Fichier : `src/API/DensityController.php`

Contrôleur léger pour la recherche de données de densité.

Endpoints :

- `GET /density/{codgeo}` — Recherche par code commune INSEE
- `GET /density/postcode/{postcode}` — Recherche par code postal (rétrocompatibilité)

Frontend\Frontend

Fichier : `src/Frontend/Frontend.php`

Gère le rendu frontend et le chargement des scripts.

Responsabilités :

- Chargement de `frontend.js` et du CSS sur les pages contenant le shortcode de la carte
- Gestionnaire du shortcode `[irisolaris_map]` (rendu du conteneur de carte + passage de la configuration via `wp_localize_script`)
- Filtre `irisolaris_map_plants_data` pour la personnalisation des données GeoJSON

Modules JavaScript

Côté frontend, le plugin suit une architecture modulaire compilée en un unique bundle via Webpack.

Modules source (`assets/src/js/`)

Le frontend est construit à partir de fichiers source ES6+ modulaires, compilés via Webpack en un unique bundle (`assets/build/frontend.js`) :

Module	Responsabilité
`frontend.js`	Orchestrateur principal — importe et connecte tous les modules
`constants.js`	Niveaux de périmètre, niveaux de densité, configuration des clusters, correspondance des statuts, timings d'animation
`utils.js`	Distance Haversine, correspondance des statuts, formatage des nombres, debounce, détection du viewport
`LoadingManager.js`	Overlays de chargement et animations
`PopupManager.js`	Création et animation des popups pour centrales/clusters/adresses
`FilterManager.js`	Gestion de l'état des filtres et interface utilisateur
`EligibilityChecker.j`	Vérification d'éligibilité côté client avec niveaux de densité
`PlantListRenderer.js`	Rendu de la liste des centrales en barre latérale avec chargement paresseux
`geocoder.js`	Autocomplétion d'adresse via BAN API

Sortie de build (`assets/build/`)

Fichier	Rôle
`assets/build/frontend.js`	Bundle compilé : classes `IrisolarisMap` + `IrisolarisGeocoder`
`assets/build/frontend-style.css`	Styles du composant carte
`assets/build/public-style.css`	Styles supplémentaires publics

Autres scripts

Fichier	Rôle
`admin/js/import.js`	Interface d'import de centrales (basée sur jQuery, AJAX en 3 phases)
`src/blocks/map/edit.js`	Composant éditeur de bloc Gutenberg
`src/blocks/map/view.js`	Vue frontend du bloc Gutenberg

Activation et désinstallation

Le plugin enregistre des hooks pour l'activation et la désinstallation afin de gérer les tables de base de données, les options par défaut et les capacités de rôle.

Activateur (`Irisolaris_Map_Activator`)

S'exécute lors de l'activation du plugin : 1. Création des tables de base de données personnalisées (`irisolaris_import_logs`, `irisolaris_density`) 2. Définition des options WordPress par défaut (10 options) 3. Création des termes de taxonomie par défaut (statut, puissance, région) 4. Ajout de capacités personnalisées aux rôles Administrateur et Éditeur

Désinstallateur (`Irisolaris_Map_Uninstaller`)

S'exécute lors de la suppression du plugin : 1. Suppression des tables de base de données personnalisées 2. Retrait des capacités personnalisées de tous les rôles 3. Suppression optionnelle des données de densité (contrôlée par l'option `irisolaris_map_remove_data_on_uninstall`) 4. Nettoyage des transients et des options

2.3 Modèle de données

Le plugin Irisolaris Map stocke ses données en combinant des structures natives WordPress (types de contenu personnalisés, taxonomies, options, transients) et deux tables de base de données personnalisées. Cette page catalogue chaque structure de données utilisée par le plugin.

Types de contenu personnalisés

CPT	Slug	Public	REST	Archive	Icône de menu	Supports
`irisolaris_plant`	`centrale-pv`	Oui	Oui	Oui	`dashicons-admin-site`	title, editor, thumbnail, custom-fields
`irisolaris_lead`	`demande-eligibilite`	Non	Oui	Non	(sous le menu centrales)	title, custom-fields

Compteurs en production : 2k+ centrales, 0 leads, 23 pages, 5 articles.

Taxonomies

Les centrales sont classifiées au moyen de deux taxonomies hiérarchiques pour le statut et la région géographique.

Taxonomie	Slug	Types de contenu	Hiérarchique	Colonne admin
`irisolaris_status`	`statut-centrale`	`irisolaris_plant`	Oui	Oui
`irisolaris_region`	`region-centrale`	`irisolaris_plant`	Oui	Oui

Termes de taxonomie par défaut

Statut (`irisolaris_status`) :

- En construction
- En service
- En développement

Région (`irisolaris_region`) : 18 régions françaises (13 métropolitaines + 5 territoires d'outre-mer)

Clés meta des contenus

Chaque type de contenu personnalisé possède un ensemble de clés meta qui stockent ses données métier. Les meta des centrales sont réparties entre les champs exposés publiquement et les champs sensibles réservés à un usage côté serveur.

Meta des centrales (17 clés)

Clé meta	Type	Description
`_irisolaris_plant_name`	text	Nom de la centrale
`_irisolaris_plant_lat`	float	Latitude
`_irisolaris_plant_lng`	float	Longitude
`_irisolaris_plant_address`	text	Adresse postale
`_irisolaris_plant_city`	text	Ville
`_irisolaris_plant_postcode`	text	Code postal

Clé meta	Type	Description
`_irisolaris_plant_region`	text	Région
`_irisolaris_plant_power`	text	Puissance (kW)
`_irisolaris_plant_status`	text	Statut de la centrale
`_irisolaris_plant_active`	checkbox (1/0)	Indicateur de visibilité
`_irisolaris_plant_work_id`	text	Identifiant chantier unique (clé d'import)
`_irisolaris_plant_project_name`	text	Nom du projet
`_irisolaris_plant_spv`	text	SPV (Special Purpose Vehicle)
`_irisolaris_plant_department`	text	Département
`_irisolaris_plant_perimeter`	text	Périmètre (km)
`_irisolaris_plant_tariff`	text	Tarif
`_irisolaris_plant_prm`	text	PRM (référence compteur)

Champs sensibles

Les champs `work_id`, `spv`, `prm`, `address`, `project_name`, `tariff` et `perimeter` ne sont **jamais exposés** via l'API GeoJSON publique. Ils sont uniquement récupérés côté serveur lors de la soumission du formulaire HubSpot.

Meta des leads (11 clés)

Clé meta	Type	Description
`_irisolaris_lead_name`	text	Nom de la personne
`_irisolaris_lead_email`	text	E-mail
`_irisolaris_lead_phone`	text	Téléphone
`_irisolaris_lead_address`	text	Adresse
`_irisolaris_lead_lat`	float	Latitude
`_irisolaris_lead_lng`	float	Longitude
`_irisolaris_lead_city`	text	Ville
`_irisolaris_lead_postcode`	text	Code postal
`_irisolaris_lead_eligible`	checkbox (1/0)	Résultat d'éligibilité
`_irisolaris_lead_date`	datetime	Date de création
`_irisolaris_lead_expiration`	date	Date d'expiration RGPD

Options WordPress

Le plugin enregistre 10 options pour contrôler l'affichage de la carte, les seuils d'éligibilité et le comportement du système. Toutes sont configurables via la page de paramètres d'administration.

Option	Valeur par défaut	Description
`irisolaris_map_max_distance`	`2` (km)	Distance maximale d'éligibilité
`irisolaris_map_lead_expiration`	`180` (jours)	Durée de conservation des leads (RGPD)
`irisolaris_map_default_lat`	`46.603354`	Latitude du centre de la carte (centre de la France)
`irisolaris_map_default_lng`	`1.888334`	Longitude du centre de la carte

Option	Valeur par défaut	Description
`irisolaris_map_default_zoom`	5	Niveau de zoom par défaut
`irisolaris_map_tile_url`	URL OpenStreetMap	URL du service de tuiles
`irisolaris_map_attribution`	OSM contributors	Texte d'attribution de la carte
`irisolaris_map_notification_email`	`admin_email`	E-mail de notification d'éligibilité
`irisolaris_plants_cache_version`	1	Compteur de version du cache (auto-incrémenté)
`irisolaris_map_remove_data_on_uninstall`	`false`	Supprimer les données de densité à la désinstallation

Valeurs en production : La plupart des options utilisent les valeurs par défaut du code. La version du cache est auto-incrémentée (chaque sauvegarde/suppression de centrale l'incrémente). La durée de conservation des leads est fixée à **180 jours**.

Patterns de cache par transients

Au-delà des options ci-dessus, le plugin fait un usage intensif des transients WordPress pour mettre en cache les opérations coûteuses et suivre les processus d'import de longue durée.

Pattern	TTL	Rôle
`irisolaris_plants_geojson_{md5}`	7 jours (complet), 1 jour (filtré)	Cache des données GeoJSON des centrales
`irisolaris_import_{id}`	1 heure	État du lot d'import de centrales
`irisolaris_import_finalize_{id}`	1 heure	État de finalisation de l'import
`irisolaris_removal_{id}`	1 heure	État du lot de suppression en masse
`irisolaris_density_import_{id}`	1 heure	État du lot d'import de densité
`irisolaris_density_finalize_{id}`	1 heure	État de finalisation de la densité

Tables de base de données personnalisées

En complément des tables WordPress standard, le plugin crée deux tables personnalisées pour les données qui ne s'intègrent pas bien dans le modèle post/meta.

`{prefix}irisolaris_import_logs`

Stocke l'historique des opérations d'import.

Colonne	Type	Description
`id`	mediumint(9) AUTO_INCREMENT	Clé primaire
`import_date`	datetime	Date d'exécution de l'import
`import_type`	varchar(50)	Type d'import
`file_name`	varchar(255)	Nom du fichier original
`items_count`	int(11)	Nombre total d'éléments
`success_count`	int(11)	Imports réussis
`error_count`	int(11)	Imports échoués
`errors`	longtext	Détails des erreurs (sérialisés)

Colonne	Type	Description
`user_id`	bigint(20)	ID utilisateur WordPress

`{prefix}irisolaris_density`

Stocke les données de classification de densité des communes INSEE.

Colonne	Type	Description
`codgeo`	varchar(10) **PK**	Code commune INSEE
`libgeo`	varchar(255)	Nom de la commune
`dens`	decimal(10,2)	Densité de population
`libdens`	varchar(255)	Libellé de la catégorie de densité
`pmun22`	decimal(10,2)	Population municipale 2022
`p1`	decimal(10,2)	Indicateur de population p1
`p2`	decimal(10,2)	Indicateur de population p2
`p3`	decimal(10,2)	Indicateur de population p3
`dens_aav`	decimal(10,2)	Densité AAV
`libdens_aav`	varchar(255)	Libellé de densité AAV
`dens7`	decimal(10,2) **INDEXED**	Classification de densité à 7 niveaux
`libdens7`	varchar(255)	Libellé de densité à 7 niveaux

La colonne `dens7` est le champ principal utilisé pour le calcul des seuils de distance d'éligibilité.

Capacités de rôle

Enfin, le plugin étend le système de rôles natif de WordPress plutôt que de créer des rôles personnalisés. Les capacités sont ajoutées aux rôles existants lors de l'activation du plugin.

Administrateur — CRUD complet sur `irisolaris_plant` et `irisolaris_lead` (11 capacités chacun) :

- `edit_irisolaris_plants`, `edit_others_irisolaris_plants`, `publish_irisolaris_plants`, `read_private_irisolaris_plants`, `delete_irisolaris_plants`, `delete_others_irisolaris_plants`, `delete_published_irisolaris_plants`, `delete_private_irisolar` `edit_published_irisolaris_plants`, `edit_private_irisolaris_plants`, `read_irisolaris_plants`
- Même pattern pour `irisolaris_leads`

Éditeur — Lecture/édition de `irisolaris_plant` (6 capacités), lecture seule de `irisolaris_lead` (2 capacités).

2.4 API REST

Le plugin Irisolaris Map expose un ensemble d'endpoints REST pour les données cartographiques, le géocodage, la vérification d'éligibilité et l'intégration CRM. Cette page documente chaque endpoint, son format de réponse et la stratégie de mise en cache associée.

Vue d'ensemble des endpoints

Tous les endpoints appartiennent au namespace `irisolaris-map/v1`.

Méthode	Route	Permission	Rôle
GET	/data/map-theme	publique	JSON du thème de la carte (document de style MapLibre)
GET	/plants	publique	Toutes les centrales actives en GeoJSON FeatureCollection
GET	/plants/{id}	publique	Centrale unique en GeoJSON Feature
GET	/density/{codgeo}	publique	Données de densité pour un code commune
GET	/density/postcode/{postcode}	publique	Données de densité par code postal
GET	/geocode	publique	Géocodage d'adresse via proxy BAN API
POST	/check-eligibility	publique	Vérification complète d'éligibilité + création de lead
POST	/import	manage_options	Import de centrales depuis un fichier CSV
POST	/submit-join-form	publique	Soumission du formulaire d'adhésion à HubSpot

Restriction de l'API REST

Le plugin **Disable REST API** est actif sur le serveur de production. Il bloque l'accès anonyme à l'API REST, de sorte que tous les endpoints publics Irisolaris doivent etre ajoutés à la liste blanche dans les paramètres du plugin.

Endpoints publics

GET `/plants`

Retourne toutes les centrales actives sous forme de GeoJSON FeatureCollection.

Format de réponse :

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [lng, lat]
      },
      "properties": {
        "id": 1234,
        "name": "Plant Name",
        "city": "City",
        "postcode": "75001",
        "region": "Île-de-France",
```

```
        "power": "250",
        "status": "En service",
        "active": 1
      }
    }
  ]
}
```

Mise en cache :

- Cache par transient : `irisolaris_plants_geojson_{md5}` (TTL de 7 jours pour les données completes, 1 jour pour les données filtrées)
- En-tetes HTTP : `Cache-Control: public, max-age=1800` + support ETag/304
- La clé de cache inclut un numéro de version incrémenté à chaque sauvegarde/suppression de centrale

Champs exposés : Uniquement les métadonnées publiques (`name`, `city`, `postcode`, `region`, `power`, `status`, `active`). Les champs sensibles des centrales sont exclus de l'API publique et récupérés côté serveur uniquement en cas de besoin (par exemple, lors de la soumission d'un formulaire).

GET `/plants/{id}`

Retourne une centrale unique sous forme de GeoJSON Feature. Meme structure de propriétés que l'endpoint de collection.

GET `/data/map-theme`

Retourne le document de style MapLibre (JSON) depuis `assets/json/`. Utilisé pour configurer l'apparence de la carte et les sources de tuiles.

GET `/geocode`

Fait office de proxy pour les requetes de géocodage vers la Base Adresse Nationale (BAN API).

Parametres de requete :

- `q` — Chaine de recherche d'adresse

Rôle : Proxy côté serveur pour éviter les problemes CORS et permettre au plugin de contrôler le comportement du géocodage.

GET `/density/{codgeo}`

Recherche les données de densité pour un code commune INSEE dans la table `irisolaris_density`.

La réponse inclut : `codgeo`, `libgeo`, `dens`, `dens7`, `libdens7` et d'autres indicateurs de densité.

GET `/density/postcode/{postcode}`

Recherche de densité rétrocompatible par code postal au lieu du code commune.

POST `/check-eligibility`

Effectue une vérification d'éligibilité côté serveur. Valide la proximité de l'adresse par rapport aux centrales en utilisant des seuils de distance basés sur la densité, et crée optionnellement un enregistrement de lead.

POST `/submit-join-form`

Soumet une demande d'adhésion au CRM HubSpot avec des données de centrale enrichies côté serveur.

Corps de la requete :

```
{
  "plant_id": 1234,
```

```
  "company": "Company Name",
  "phone": "+33...",
  "email": "user@example.com",
  "zip": "75001",
  "city": "Paris",
  "pdl": "12345678901234",
  "nonce": "wp_nonce_value"
}
```

Enrichissement côté serveur : Le plugin récupere 14 champs meta sensibles de la centrale (work_id, project_name, SPV, address, PRM, lat/lng, power, tariff, perimeter, department, status) et les inclut dans la soumission HubSpot. Ces champs ne sont jamais exposés au client.

Payload HubSpot : 21 champs au total (7 provenant du client + 14 enrichis côté serveur).

Endpoints d'administration

En complément des endpoints publics ci-dessus, un endpoint authentifié est disponible pour les administrateurs.

POST `/import`

Permission : `manage_options` (administrateurs uniquement)

Importe des centrales depuis un fichier CSV uploadé. Utilisé par l'interface d'import de l'administration.

Stratégie de mise en cache

L'API REST utilise une approche de cache à deux niveaux — les transients WordPress pour les données côté serveur et les en-tetes HTTP pour le cache côté client.

Cache par transients

La réponse GeoJSON est mise en cache via les transients WordPress :

1. **Clé de cache :** `irisolaris_plants_geojson_{md5}` où le hash MD5 inclut :
   - Les parametres de filtre (statut, région)
   - Le numéro de version du cache (option `irisolaris_plants_cache_version`)
2. **TTL :** 7 jours pour les requetes non filtrées, 1 jour pour les requetes filtrées
3. **Invalidation :** Stratégie d'incrémentation de version — lorsqu'une centrale est sauvegardée, supprimée ou que ses termes de taxonomie changent, le compteur de version est incrémenté. Les nouvelles requetes génèrent une nouvelle clé de cache (les anciens transients expirent naturellement).

Cache HTTP

Les réponses REST incluent :

- `Cache-Control: public, max-age=1800` (30 minutes)
- En-tete `ETag` basé sur le hash du contenu
- Réponses `304 Not Modified` lorsque l'ETag correspond

Flux d'invalidation du cache

```
Trigger (save/delete plant, term change)
  └── API::invalidate_plants_cache()
      └── bump_cache_version()
          └── irisolaris_plants_cache_version += 1
              └── Next GET /plants → new MD5 key → cache miss → fresh query
```

2.5 Frontend JavaScript

Architecture du bundle

Le frontend est construit a partir de fichiers source ES6+ modulaires (`assets/src/js/`) et compilé via Webpack 5 en un unique bundle situé dans `assets/build/frontend.js`. Ce bundle expose deux classes principales — `IrisolarisMap` et `IrisolarisGeocoder` — qui ensemble alimentent l'expérience cartographique interactive.

WordPress injecte la configuration d'exécution dans la page via `wp_localize_script`, fournissant au bundle les URLs des endpoints REST (`/plants`, `/data/map-theme`, `/geocode`), les paramètres par défaut de la carte (coordonnées du centre, niveau de zoom) et un nonce pour les requêtes authentifiées. Aucune URL d'API ni configuration n'est donc codée en dur dans le JavaScript — tout provient du serveur.

Initialisation de la carte

Lorsqu'une page contenant la carte se charge, le bundle suit la séquence suivante :

1. **Récupération du thème de la carte** — Une requête `GET /data/map-theme` récupère le document de style MapLibre (JSON) qui définit les sources de tuiles, les couches et le style visuel. Le style pointe vers OpenFreeMap (`tiles.openfreemap.org`) pour les tuiles cartographiques.
2. **Création de l'instance MapLibre** — `new maplibregl.Map()` est initialisé avec le thème et les coordonnées du centre par défaut (46.603354, 1.888334 — centre de la France) au niveau de zoom 5. La bibliothèque cartographique (MapLibre GL JS 2.4.0) est chargée depuis un CDN (unpkg.com).
3. **Chargement des données des centrales** — Une requête `GET /plants` retourne toutes les centrales actives sous forme de GeoJSON FeatureCollection. Cette réponse bénéficie à la fois du cache HTTP (`Cache-Control` de 30 minutes + ETag/304) et du cache par transients côté serveur (TTL de 7 jours).
4. **Rendu des marqueurs clusterisés** — Le GeoJSON est ajouté comme source MapLibre avec le clustering côté client activé. Trois couches gèrent le rendu :
   - **clusters** — Marqueurs circulaires pour les groupes de centrales, dimensionnés selon le nombre de points
   - **cluster-count** — Libellés numériques indiquant le nombre de centrales dans chaque cluster
   - **unclustered-point** — Marqueurs individuels des centrales aux niveaux de zoom élevés

   Le clustering côté client est essentiel ici : avec plus de 2 000 centrales, afficher des marqueurs individuels à faible zoom serait inutilisable.
5. **Alimentation de la barre latérale** — Un panneau de liste des centrales est affiché à côté de la carte avec chargement paresseux et synchronisation du défilement avec le viewport de la carte.

Interactions utilisateur

Popups et navigation

Un clic sur un marqueur de centrale individuel ouvre un popup affichant le nom de la centrale, la ville, le code postal, la région, la puissance (kW) et le statut avec un code couleur. Un clic sur un cluster effectue un zoom avant pour le déployer.

Filtrage

Les utilisateurs peuvent filtrer les centrales par **statut** (En construction, En service, En développement) et par **région** (18 régions françaises). Les filtres mettent à jour en temps réel les marqueurs de la carte et la liste de la barre latérale. Les requêtes filtrées passent par le même endpoint REST mais avec des paramètres de requête, et utilisent un cache par transient distinct avec un TTL plus court d'1 jour.

Recherche d'adresse et éligibilité

La classe `IrisolarisGeocoder` fournit un champ de saisie avec autocomplétion d'adresse. Pendant la saisie de l'utilisateur, l'entrée est soumise avec un debounce puis interroge la BAN (Base Adresse Nationale) API pour des suggestions d'adresses françaises. Lorsque l'utilisateur sélectionne une adresse, le flux de vérification d'éligibilité est déclenché.

Éligibilité côté client

La vérification d'éligibilité s'exécute entièrement dans le navigateur pour un retour instantané, en utilisant des seuils de distance basés sur la densité qui reflètent la réglementation française en matière d'autoconsommation collective :

Niveau de densité (`dens7`)	Classification	Distance maximale
1–2	Rural / très rural	2 km
3–4	Densité intermédiaire	10 km
5+	Urbain / urbain dense	20 km

Le flux fonctionne comme suit :

1. Le résultat du géocodage BAN API fournit les coordonnées et un `citycode` INSEE
2. Une requête `GET /density/{citycode}` récupère la valeur `dens7` de la commune
3. `findNearbyPlants()` utilise la formule de Haversine pour trouver toutes les centrales dans un rayon de 20 km
4. `isPlantEligible()` applique le seuil du niveau de densité à chaque candidate
5. Les centrales éligibles sont mises en évidence sur la carte, un marqueur est placé à l'adresse de l'utilisateur, et la liste de la barre latérale est filtrée pour n'afficher que les centrales éligibles

Le serveur effectue également une vérification définitive d'éligibilité lors de la soumission du formulaire (`POST /check-eligibility`), de sorte que la vérification côté client fournit un retour UX rapide tandis que le serveur reste l'autorité.

Organisation des modules source

Le code source dans `assets/src/js/` est organisé par domaine de responsabilité :

- **`constants.js`** — Configuration centrale : niveaux de périmètre, niveaux de densité, paramètres de style des clusters, correspondances statut-couleur des centrales et timings d'animation. Modifier les seuils de distance d'éligibilité ou ajouter un nouveau statut de centrale commence ici.
- **`utils.js`** — Utilitaires partagés : calcul de distance Haversine entre coordonnées, correspondance des chaînes de statut, formatage des nombres, debounce d'entrée et détection du viewport pour le comportement responsive.
- **`LoadingManager.js`** — Gère les overlays de chargement et les animations de spinner affichés pendant le chargement du thème de la carte et des données GeoJSON.
- **`PopupManager.js`** — Crée et anime les popups pour les centrales, les clusters et le marqueur d'adresse de l'utilisateur. Gère les transitions d'affichage/masquage.
- **`FilterManager.js`** — Gère l'état des filtres (quels filtres de statut/région sont actifs) et les gestionnaires de bascule de l'interface qui déclenchent les mises à jour de la carte et de la liste.
- **`EligibilityChecker.js`** — Implémente l'algorithme d'éligibilité côté client décrit ci-dessus : recherche de densité, filtrage par Haversine, application des niveaux et mise en évidence visuelle.
- **`PlantListRenderer.js`** — Affiche la liste des centrales dans la barre latérale avec chargement paresseux pour les performances et la maintient synchronisée avec le viewport de la carte.
- **`geocoder.js`** — L'autocomplétion d'adresse `IrisolarisGeocoder` : debounce d'entrée, requêtes BAN API, rendu du menu déroulant de suggestions et callbacks de sélection.
- **`frontend.js`** — L'orchestrateur principal qui importe tous les modules, les connecte entre eux et gère le cycle de vie global de l'application.

Interface d'import d'administration

L'interface d'import de centrales côté administration (`admin/js/import.js`) est un script jQuery distinct — il ne fait pas partie du bundle Webpack. Il pilote le processus d'import AJAX en 3 phases (initialisation, traitement par

lots, finalisation) avec une barre de progression, en gérant l'upload de fichier, le suivi de progression par lots et l'affichage du résultat final. Voir **Import de centrales** pour le flux d'import complet.

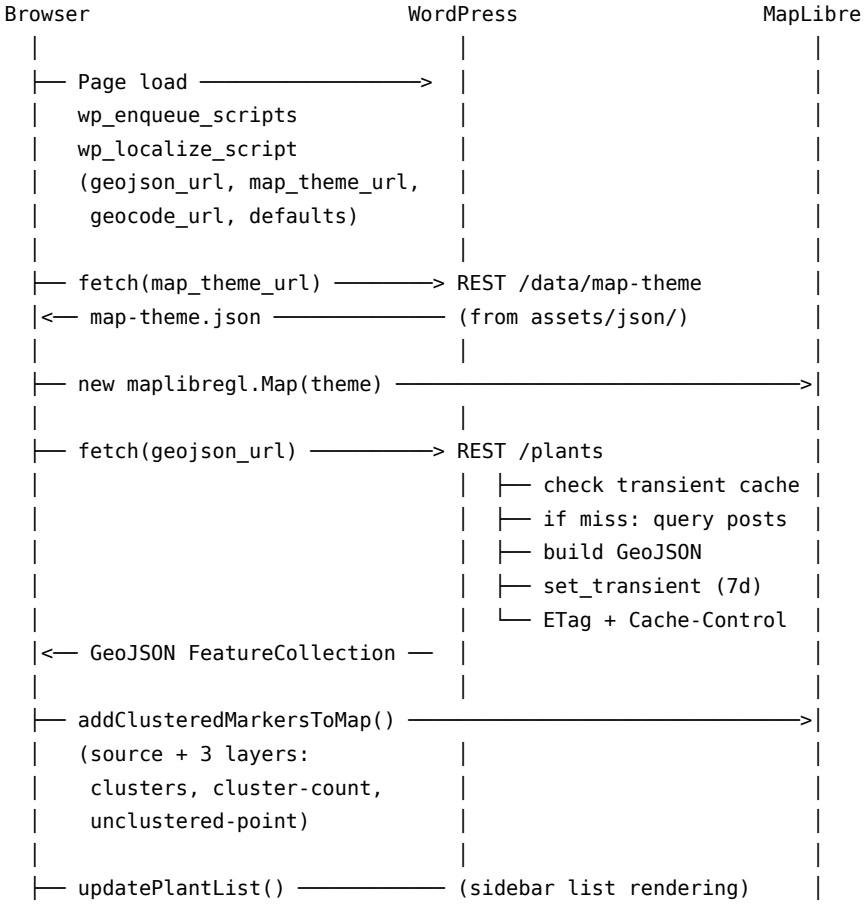
3 Fonctionnalités

3.1 Carte interactive

Vue d'ensemble

La carte interactive affiche toutes les centrales photovoltaïques actives sur une carte MapLibre GL JS. Les utilisateurs peuvent parcourir les centrales, filtrer par statut/région, rechercher des adresses et vérifier leur éligibilité à l'autoconsommation collective.

Flux d'affichage



Intégration de la carte

La carte peut être ajoutée à n'importe quelle page WordPress en utilisant un shortcode ou un bloc Gutenberg.

Shortcode

```
[irisolaris_map]
[irisolaris_map height="600px" width="100%"]
[irisolaris_map filter_status="En service"]
[irisolaris_map filter_region="Île-de-France"]
```

Attribut	Par défaut	Description
`height`	`500px`	Hauteur du conteneur de la carte
`width`	`100%`	Largeur du conteneur de la carte
`filter_status`	—	Pré-filtrer par statut de centrale

Attribut	Par défaut	Description
`filter_power`	—	Pré-filtrer par catégorie de puissance
`filter_region`	—	Pré-filtrer par région

Bloc Gutenberg

Le bloc `irisolaris-map/map` est également disponible. Le shortcode est la méthode d'intégration recommandée.

Couches de la carte

Une fois chargée, la carte affiche les centrales à l'aide de trois couches MapLibre sur une source GeoJSON unique avec le clustering activé :

Couche	Type	Fonction
`clusters`	circle	Cercles de regroupement (dimensionnés par nombre de points)
`cluster-count`	symbol	Étiquettes numériques indiquant le nombre de centrales par cluster
`unclustered-point`	circle	Marqueurs individuels des centrales

Le clustering est géré côté client par MapLibre, réduisant le nombre d'éléments DOM pour les 2 000+ centrales.

Popups des centrales

Un clic sur le marqueur d'une centrale individuelle ouvre un popup affichant :

- Nom de la centrale
- Ville et code postal
- Région
- Puissance de sortie (kW)
- Statut (avec code couleur)

Un clic sur un cluster effectue un zoom avant pour le déployer.

Liste latérale des centrales

À côté de la carte, un panneau latéral liste toutes les centrales visibles avec :

- Nom de la centrale, ville et statut
- Chargement progressif (lazy loading) pour la performance
- Synchronisation du défilement avec la fenêtre de la carte
- Mises à jour en temps réel lors des changements de filtres ou de la vérification d'éligibilité

Filtres

Les utilisateurs peuvent affiner les centrales affichées selon deux dimensions de filtrage.

La carte supporte le filtrage par :

- **Statut** — En construction, En service, En développement
- **Région** — 18 régions françaises

Les filtres mettent à jour simultanément les marqueurs de la carte et la liste latérale en temps réel. Les requêtes GeoJSON filtrées utilisent un cache transient séparé avec un TTL de 1 jour.

Configuration

Les paramètres par défaut de la carte sont contrôlés par les options WordPress :

Option	Par défaut	Description
`irisolaris_map_default_lat`	`46.603354`	Latitude du centre de la carte
`irisolaris_map_default_lng`	`1.888334`	Longitude du centre de la carte
`irisolaris_map_default_zoom`	`5`	Niveau de zoom par défaut
`irisolaris_map_tile_url`	OpenStreetMap URL	URL du service de tuiles
`irisolaris_map_attribution`	OSM contributors	Texte d'attribution de la carte

Ces valeurs sont transmises au frontend via `wp_localize_script`.

3.2 Vérification d'éligibilité

Vue d'ensemble

La vérification d'éligibilité détermine si un utilisateur à une adresse donnée peut participer à l'autoconsommation collective depuis une centrale photovoltaïque à proximité. Les règles sont basées sur :

1. **Proximité géographique** — distance entre l'adresse de l'utilisateur et la centrale
2. **Densité de population** — la classification de densité communale INSEE détermine la distance maximale autorisée

Niveaux de distance basés sur la densité

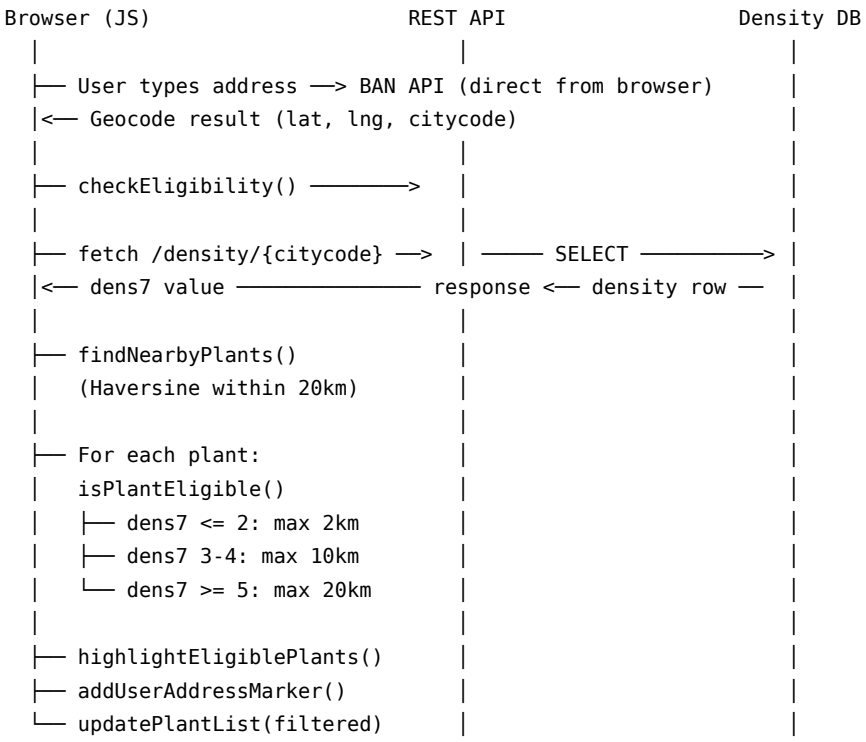
La distance d'éligibilité dépend du champ `dens7` de la grille de densité INSEE :

Valeur `dens7`	Classification	Distance maximale
1	Très peu dense (rural)	2 km
2	Peu dense	2 km
3	Densité intermédiaire basse	10 km
4	Densité intermédiaire haute	10 km
5	Dense	20 km
6	Très dense	20 km
7	Extrêmement dense (urban)	20 km

Note

Ce système par niveaux reflète la réglementation énergétique française pour l'autoconsommation collective : les zones plus denses autorisent des périmètres plus grands car l'infrastructure du réseau électrique supporte des connexions plus distantes.

Flux d'éligibilité



Étapes détaillées

1. **Géocodage de l'adresse** — L'utilisateur saisit une adresse. L'API BAN (Base Adresse Nationale) retourne les coordonnées (`lat`, `lng`) et un `citycode` INSEE.
2. **Recherche de densité** — Le frontend appelle `GET /density/{citycode}` pour récupérer la valeur `dens7` de la commune depuis la table de densité.
3. **Recherche de centrales à proximité** — `findNearbyPlants()` utilise la formule de Haversine pour trouver toutes les centrales dans un rayon de 20 km autour des coordonnées de l'utilisateur.
4. **Éligibilité par centrale** — `isPlantEligible()` applique le seuil du niveau de densité :
   - Si `dens7 <= 2` -> la centrale doit être à moins de 2 km
   - Si `dens7` vaut 3 ou 4 -> la centrale doit être à moins de 10 km
   - Si `dens7 >= 5` -> la centrale doit être à moins de 20 km
5. **Retour visuel** — Les centrales éligibles sont mises en surbrillance sur la carte, un marqueur est placé à l'adresse de l'utilisateur, et la liste latérale est filtrée pour n'afficher que les centrales éligibles.

Données de la grille de densité

Les données de densité sont stockées dans la table personnalisée `{prefix}irisolaris_density`, importées depuis les fichiers de classification communale INSEE (voir **Import de densité**).

Champs clés pour l'éligibilité :

- `codgeo` — Code commune INSEE (clé primaire, correspond au `citycode` de l'API BAN)
- `dens7` — Classification de densité à 7 niveaux (indexée pour des recherches rapides)
- `libdens7` — Libellé de densité lisible

Configuration

Option	Par défaut	Description
`irisolaris_map_max_distance`	2 (km)	Distance maximale d'éligibilité par défaut

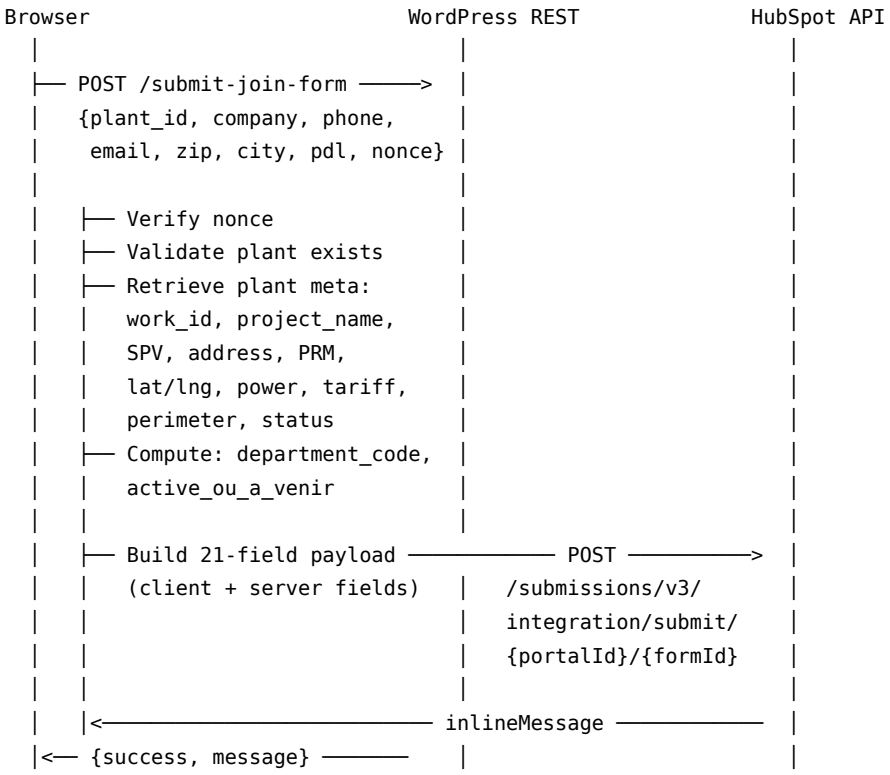
Les règles par niveaux de densité (2/10/20 km) sont définies à la fois dans le JS frontend et dans `Plugin.php`.

3.3 Formulaire d'adhésion (HubSpot)

Vue d'ensemble

Lorsqu'un utilisateur est éligible à l'autoconsommation collective à proximité d'une centrale, il peut soumettre un formulaire d'adhésion. Les données du formulaire sont envoyées au CRM HubSpot avec des métadonnées de la centrale enrichies côté serveur, garantissant que les champs sensibles ne sont jamais exposés au client.

Flux de soumission



Champs soumis par le client (7)

Les champs suivants sont collectés directement depuis la saisie du formulaire par l'utilisateur et envoyés dans la requête POST.

Champ	Source
`company`	Saisie utilisateur — nom de l'entreprise
`phone`	Saisie utilisateur — numéro de téléphone
`email`	Saisie utilisateur — adresse e-mail
`zip`	Saisie utilisateur — code postal
`city`	Saisie utilisateur — ville
`n__pdl___prm`	Saisie utilisateur — référence PDL (Point de Livraison)
`plant_id`	Automatique — identifiant de la centrale sélectionnée

Champs enrichis côté serveur (14)

Ces champs sont récupérés côté serveur depuis les métadonnées de l'article (post meta) de la centrale et ne sont **jamais exposés au client**. Ils couvrent la localisation de la centrale, les détails techniques et les identifiants de projet — 14 champs au total, incluant des valeurs calculées telles que le code département et le statut d'activité de la centrale.

Modèle de sécurité

La séparation entre les champs client et serveur est intentionnelle. La soumission du formulaire utilise un modèle d'**enrichissement côté serveur** pour la sécurité :

1. **API GeoJSON publique** — N'expose que les données non sensibles des centrales (nom, ville, code postal, région, puissance, statut). Les champs comme `work_id`, `SPV`, `PRM`, `address`, `tariff` et `perimeter` sont exclus.
2. **Récupération côté serveur** — Lors de la soumission du formulaire, le serveur récupère les métadonnées sensibles de la centrale directement depuis la base de données en utilisant le `plant_id`.
3. **Vérification du nonce** — La soumission du formulaire nécessite un nonce WordPress valide pour prévenir les attaques CSRF.
4. **Soumission HubSpot** — Le payload combiné de 21 champs est envoyé côté serveur à l'API Forms de HubSpot, de sorte que l'utilisateur ne voit jamais les données enrichies.

Intégration HubSpot

Paramètre	Valeur
Point d'accès API	`https://api.hsforms.com/submissions/v3/integration/submit/{portalId}/{formId}`
Authentification	Portal ID et Form ID intégrés dans l'URL
Réponse	Champ `inlineMessage` transmis au client

Gestion des erreurs

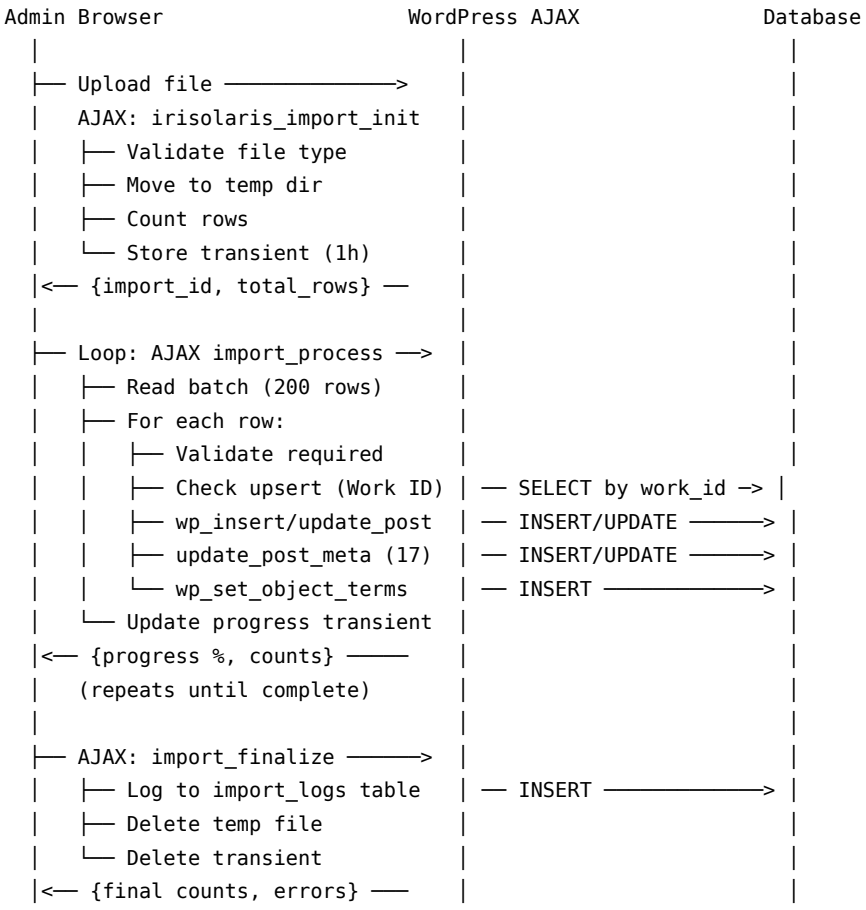
- **Échec du nonce** -> `wp_send_json_error()` avec erreur d'authentification
- **Centrale introuvable** -> `wp_send_json_error()` avec erreur de centrale invalide
- **Erreur HTTP HubSpot** -> Vérification du code de statut (200 = succès), message d'erreur transmis au client
- **Succès** -> `wp_send_json_success()` avec le message inline de HubSpot

3.4 Import des centrales (CSV/XLSX)

Vue d'ensemble

Les administrateurs peuvent importer en masse des centrales photovoltaïques à partir de fichiers CSV ou XLSX. L'import utilise un processus AJAX en 3 phases avec traitement par lots pour gérer de grands jeux de données sans problème de délai d'expiration.

Flux d'import



Processus en trois phases

Phase 1 : Initialisation (`irisolaris_import_init`)

1. Valide le type du fichier téléversé (CSV, XLSX, XLS)
2. Déplace le fichier dans un répertoire temporaire
3. Ouvre le fichier avec PhpSpreadsheet et compte les lignes
4. Stocke l'état de l'import dans un transient (TTL de 1 heure)
5. Retourne l'`import_id` et le `total_rows` au client

Phase 2 : Traitement par lots (`irisolaris_import_process`)

Traite **200 lignes par requête** :

1. Lit le lot suivant depuis le tableur
2. Pour chaque ligne :
   - Valide les champs obligatoires
   - Vérifie l'existence d'une centrale par `work_id` (logique d'upsert)
   - Crée ou met à jour l'article WordPress (`wp_insert_post` / `wp_update_post`)
   - Met à jour les 17 champs post meta

- Affecte les termes de taxonomie (statut, région)

3. Met à jour le transient de progression
4. Retourne le pourcentage de progression et les compteurs en cours

Optimisations de performance durant l'import :

- `wp_suspend_cache_invalidation()` — désactive l'invalidation du cache pendant le lot
- `wp_defer_term_counting()` — diffère le comptage des termes de taxonomie
- `update_meta_cache()` — pré-charge le cache meta en masse
- `update_object_term_cache()` — pré-charge le cache des termes en masse

Phase 3 : Finalisation (`irisolaris_import_finalize`)

1. Journalise les résultats de l'import dans la table `irisolaris_import_logs`
2. Supprime le fichier temporaire
3. Supprime le transient d'état de l'import
4. Retourne les compteurs finaux de succès/erreurs et le détail des erreurs

Logique d'upsert

Pour permettre la réimportation sûre de jeux de données mis à jour, l'import utilise le champ `work_id` comme clé unique :

- Si une centrale avec le même `_irisolaris_plant_work_id` existe -> **mise à jour** de l'article existant et de ses métadonnées
- Si aucune correspondance -> **insertion** d'un nouvel article

Cela permet des imports répétés du même jeu de données pour mettre à jour les enregistrements existants plutôt que de créer des doublons.

Correspondance des champs meta

Chaque ligne du tableur est mappée aux 17 champs meta de centrale listés ci-dessous.

Clé meta	Description
`_irisolaris_plant_name`	Nom de la centrale
`_irisolaris_plant_lat`	Latitude
`_irisolaris_plant_lng`	Longitude
`_irisolaris_plant_address`	Adresse postale
`_irisolaris_plant_city`	Ville
`_irisolaris_plant_postcode`	Code postal
`_irisolaris_plant_region`	Région
`_irisolaris_plant_power`	Puissance de sortie (kW)
`_irisolaris_plant_status`	Statut de la centrale
`_irisolaris_plant_active`	Indicateur de visibilité (1/0)
`_irisolaris_plant_work_id`	Identifiant de travail unique (clé d'import)
`_irisolaris_plant_project_name`	Nom du projet
`_irisolaris_plant_spv`	SPV (Special Purpose Vehicle)
`_irisolaris_plant_department`	Département
`_irisolaris_plant_perimeter`	Périmètre (km)
`_irisolaris_plant_tariff`	Tarif
`_irisolaris_plant_prm`	PRM (référence compteur)

Formats supportés

La lecture de tous les tableurs est assurée par **PhpSpreadsheet** (v1.29.11), qui supporte les formats suivants :

Format	Extension	Support
CSV	`.csv`	Complet
Excel (XLSX)	`.xlsx`	Complet
Excel (XLS)	`.xls`	Complet

Gestion des erreurs

- Les erreurs par ligne sont collectées et stockées (non bloquantes)
- Le détail des erreurs est sérialisé et journalisé dans la table `irisolaris_import_logs`
- L'interface d'import affiche une barre de progression avec les compteurs de succès/erreurs
- Le rapport final inclut le total d'éléments, les succès, les erreurs et le détail des erreurs

Interface d'administration

Accès via : **Centrales PV -> Importer** dans le menu d'administration WordPress.

Nécessite la capacité `manage_options` (administrateurs uniquement).

Actions supplémentaires d'administration :

- **Effacer l'historique d'import** — vide la table `irisolaris_import_logs`
- **Vider le cache** — supprime manuellement le cache transient GeoJSON

3.5 Import de la grille de densité

Vue d'ensemble

La grille de densité contient les données de classification communale INSEE utilisées pour le calcul des niveaux de distance d'éligibilité. Les données sont importées dans la table personnalisée `{prefix}irisolaris_density` et interrogées lors des vérifications d'éligibilité.

Source des données

Les données de densité proviennent des fichiers de classification communale de l'INSEE (Institut National de la Statistique et des Études Économiques). Ces fichiers contiennent les indicateurs de densité de population pour toutes les communes françaises.

Processus d'import

L'import de densité suit le même schéma AJAX en 3 phases que l'import des centrales :

Phase 1 : Initialisation (`irisolaris_import_density_init`)

1. Valide le fichier téléversé
2. Compte les lignes
3. Stocke l'état de l'import dans un transient (TTL de 1 heure)

Phase 2 : Traitement par lots (`irisolaris_import_density_process`)

Traite les lignes par lots, en insérant ou mettant à jour les enregistrements dans la table `irisolaris_density` en utilisant le `codgeo` (code commune INSEE) comme clé primaire.

Phase 3 : Finalisation (`irisolaris_import_density_finalize`)

1. Journalise les résultats de l'import
2. Nettoie les fichiers temporaires et les transients

Structure de la table

`{prefix}irisolaris_density`

Colonne	Type	Description
`codgeo`	varchar(10) **PK**	Code commune INSEE
`libgeo`	varchar(255)	Nom de la commune
`dens`	decimal(10,2)	Densité de population
`libdens`	varchar(255)	Libellé de la catégorie de densité
`pmun22`	decimal(10,2)	Population municipale 2022
`p1`	decimal(10,2)	Indicateur de population p1
`p2`	decimal(10,2)	Indicateur de population p2
`p3`	decimal(10,2)	Indicateur de population p3
`dens_aav`	decimal(10,2)	Densité AAV (Aire d'Attraction des Villes)
`libdens_aav`	varchar(255)	Libellé de la densité AAV
`dens7`	decimal(10,2) **INDEXED**	Classification de densité à 7 niveaux
`libdens7`	varchar(255)	Libellé de la densité à 7 niveaux

Champ clé : `dens7`

La colonne `dens7` est le champ principal utilisé pour les calculs d'éligibilité. Elle classe les communes sur une échelle à 7 niveaux :

Niveau	Libellé	Distance d'éligibilité
1	Très peu dense	2 km
2	Peu dense	2 km
3	Densité intermédiaire basse	10 km
4	Densité intermédiaire haute	10 km
5	Dense	20 km
6	Très dense	20 km
7	Extrêmement dense	20 km

La colonne est indexée pour des recherches rapides lors des vérifications d'éligibilité.

Accès via l'API REST

Une fois importées, les données de densité sont exposées au frontend via deux endpoints REST utilisés lors des vérifications d'éligibilité.

Méthode	Route	Fonction
`GET`	`/density/{codgeo}`	Recherche par code commune INSEE
`GET`	`/density/postcode/{postcode}`	Recherche par code postal (rétrocompatibilité)

Les deux sont des endpoints publics utilisés par le frontend lors des vérifications d'éligibilité.

Interface d'administration

Les administrateurs gèrent la grille de densité depuis un menu dédié de premier niveau dans WordPress.

Accès via : **Grilles de densité** (menu de premier niveau) dans l'administration WordPress.

Actions disponibles :

- **Importer la grille de densité** — Téléverser un fichier de classification INSEE
- **Voir tous les enregistrements de densité** — Parcourir les données importées
- **Supprimer toutes les données de densité** — Vider la table de densité
- **Effacer l'historique d'import** — Supprimer les entrées du journal d'import de densité

Toutes les actions nécessitent la capacité `manage_options` (administrateurs uniquement).

3.6 RGPD et gestion des leads

Vue d'ensemble

Statut

Le stockage local des leads est disponible mais n'est pas actuellement actif. Les soumissions du formulaire d'adhésion sont envoyées directement au CRM HubSpot via l'endpoint `/submit-join-form`. Le système de gestion des leads décrit ci-dessous peut être activé lorsqu'un suivi local des leads est nécessaire.

Le plugin inclut un système de gestion des leads conforme au RGPD avec conservation des données et nettoyage automatique à l'expiration. Lorsqu'il est activé, les leads sont créés en tant qu'articles personnalisés WordPress lorsque les utilisateurs soumettent des vérifications d'éligibilité ou des formulaires d'adhésion.

Cycle de vie d'un lead

1. **Création** — Un lead est créé en tant que type d'article personnalisé `irisolaris_lead` lorsqu'un utilisateur soumet une vérification d'éligibilité
2. **Stockage** — Les données du lead (nom, e-mail, téléphone, adresse, coordonnées, résultat d'éligibilité) sont stockées en tant que post meta
3. **Date d'expiration** — Calculée à la création : `date_courante + irisolaris_map_lead_expiration` (par défaut : 180 jours)
4. **Nettoyage RGPD** — Une tâche cron quotidienne supprime tous les leads ayant dépassé leur date d'expiration
5. **Suppression définitive** — Les leads sont supprimés de façon permanente (`wp_delete_post($id, true)`) — pas de corbeille

Données du lead (Post Meta)

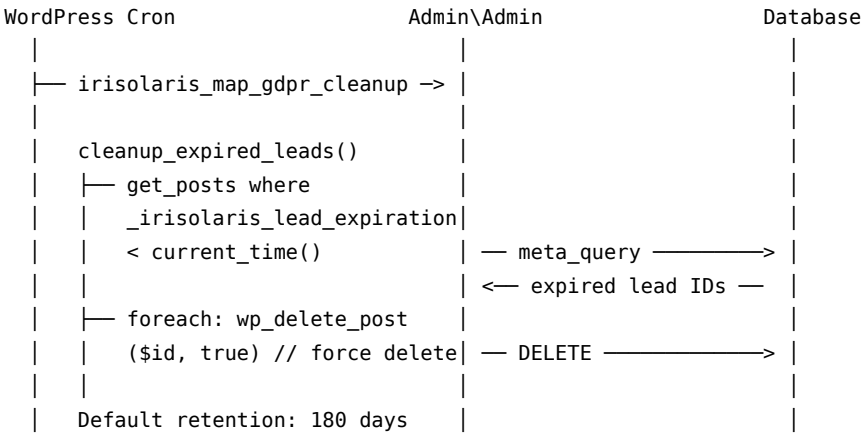
Clé meta	Type	Description
`_irisolaris_lead_name`	text	Nom de la personne
`_irisolaris_lead_email`	text	E-mail
`_irisolaris_lead_phone`	text	Téléphone
`_irisolaris_lead_address`	text	Adresse
`_irisolaris_lead_lat`	float	Latitude
`_irisolaris_lead_lng`	float	Longitude
`_irisolaris_lead_city`	text	Ville
`_irisolaris_lead_postcode`	text	Code postal
`_irisolaris_lead_eligible`	checkbox (1/0)	Résultat d'éligibilité
`_irisolaris_lead_date`	datetime	Date de création
`_irisolaris_lead_expiration`	date	Date d'expiration RGPD

Tâche cron de nettoyage RGPD

Planification

Événement	Fréquence	Gestionnaire
`irisolaris_map_gdpr_cleanup`	Quotidien	`Admin::cleanup_expired_leads()`

Flux de nettoyage



Détails d'implémentation

- La tâche cron est planifiée par `Admin\Admin` lors de l'initialisation via `wp_schedule_event()`
- Utilise `meta_query` pour trouver les leads où `_irisolaris_lead_expiration < current_time()`
- Chaque lead expiré est supprimé de façon permanente (contourne la corbeille)
- La tâche cron s'exécute même s'il n'y a aucun lead (vérification légère)

Configuration

Option	Par défaut	Valeur en production	Description
`irisolaris_map_lead_expiration`	`180`	`180`	Nombre de jours avant la suppression automatique des données du lead

Configurable via : **Centrales PV -> Paramètres** dans l'administration WordPress.

Statut en production

- **Création de leads :** Non actuellement actif — les soumissions sont envoyées directement à HubSpot
- **Tâche cron RGPD :** Enregistrée et exécutée quotidiennement
- **Période de conservation :** 180 jours (configurée)

Configuration du type d'article Lead

Paramètre	Valeur
Type d'article	`irisolaris_lead`
Slug	`demande-eligibilite`
Public	Non (non visible pour les visiteurs)
API REST	Oui (accessible via REST)
Archive	Non
Supporte	title, custom-fields

Les leads apparaissent dans l'administration WordPress sous le menu Centrales. Les colonnes d'administration affichent les détails du lead (nom, e-mail, éligibilité, dates). Les éditeurs ont un accès en lecture seule ; les administrateurs ont un accès CRUD complet.

Notifications par e-mail

Lorsqu'un nouveau lead est créé, une notification par e-mail est envoyée à l'adresse configurée dans `irisolaris_map_notification_email` (par défaut, l'adresse e-mail de l'administrateur WordPress).

4 Déploiement

4.1 Build et production

Cette page couvre la compilation du plugin depuis les sources, son empaquetage pour la distribution, et son déploiement sur le serveur de production.

Processus de build

Commandes

```
pnpm run build:assets      # Compile JS/CSS source → assets/build/ (Webpack, production)
pnpm run build:assets:dev  # Idem, mode développement
pnpm run build             # Build du bloc Gutenberg (wp-scripts)
pnpm run build:all         # Les deux : bloc + assets
```

Les fichiers source frontend dans `assets/src/js/` sont compilés via Webpack 5 en `assets/build/frontend.js`. Le bloc Gutenberg utilise la chaîne d'outils `@wordpress/create-block`.

Structure du répertoire dist

Le répertoire `dist/` contient le plugin prêt à être déployé :

```
dist/
├── irisolaris-map.php              # Point d'entrée du plugin
├── composer.json / composer.lock
├── vendor/                         # Dépendances Composer (PhpSpreadsheet, etc.)
├── src/
│   ├── Core/Plugin.php
│   ├── Admin/Admin.php
│   ├── API/API.php
│   ├── API/DensityController.php
│   ├── Frontend/Frontend.php
│   └── blocks/map/
│       ├── block.json
│       ├── edit.js
│       └── view.js
├── includes/
│   ├── class-irisolaris-map-activator.php
│   └── class-irisolaris-map-uninstaller.php
├── assets/
│   ├── build/
│   │   ├── frontend.js             # Application cartographique principale (compilée depuis les sources
modulaires)
│   │   ├── frontend-style.css
│   │   └── public-style.css
│   └── json/
│       └── map-theme.json          # Document de style MapLibre
├── admin/
│   ├── js/import.js                # UI d'import admin
│   └── partials/                   # Templates des pages admin
├── templates/
│   └── map-shortcode.php           # Template du shortcode
└── languages/                      # Fichiers de traduction
```

Empaquetage et versionnement

Une fois les assets compilés, le plugin doit être versionné et empaqueté pour le déploiement.

Gestion des versions

Le script `set-version.sh` met à jour le numéro de version dans trois fichiers :

- `package.json`
- `irisolaris-map.php` (en-tête du plugin)
- `includes/config.php` (constante `IRISOLARIS_MAP_VERSION`)

Package de distribution

Le script `package.sh` produit un zip déployable :

1. Exécute `pnpm run build:all` (compile le bloc et les assets frontend)
2. Empaquète le plugin (exclut les fichiers source, inclut la sortie compilée)
3. Crée un zip versionné : `irisolaris-map-v{version}-{timestamp}.zip`

Méthode de déploiement

1. Exécuter `./package.sh` pour produire un zip versionné
2. Uploader et extraire dans le répertoire des plugins WordPress sur le serveur (`wp-content/plugins/irisolaris-map/`)
3. Vérifier que le plugin est actif dans l'administration WordPress

Docker

Une configuration Docker Compose existe pour le développement local :

```
docker compose up -d
```

Paramètres de configuration

Le plugin utilise 10 options WordPress pour la configuration à l'exécution (pas de fichier `.env`). Voir **Configuration** pour plus de détails.

4.2 Configuration

Cette page couvre tout ce qu'il faut configurer après l'installation du plugin Irisolaris Map — options WordPress, connexions aux services externes, allowlisting de l'API REST et capacités des rôles.

Options WordPress

Toute la configuration du plugin est stockée sous forme d'options WordPress, accessibles via **Centrales PV -> Paramètres** dans le panneau d'administration.

Affichage de la carte

Option	Défaut	Description
`irisolaris_map_default_lat`	`46.603354`	Latitude du centre de la carte (centre de la France)
`irisolaris_map_default_lng`	`1.888334`	Longitude du centre de la carte
`irisolaris_map_default_zoom`	`5`	Niveau de zoom par défaut
`irisolaris_map_tile_url`	URL OpenStreetMap	URL du service de tuiles
`irisolaris_map_attribution`	OSM contributors	Texte d'attribution de la carte

Éligibilité et prospects

Option	Défaut	Description
`irisolaris_map_max_distance`	`2` (km)	Distance maximale de repli pour l'éligibilité
`irisolaris_map_lead_expiration`	`180` (jours)	Durée de conservation des prospects (RGPD)
`irisolaris_map_notification_email`	`admin_email`	Adresse e-mail pour les notifications d'éligibilité

Système

Option	Défaut	Description
`irisolaris_plants_cache_version`	`1`	Compteur de version du cache (auto-incrémenté, ne pas modifier manuellement)
`irisolaris_map_remove_data_on_uninstall`	`false`	Supprimer ou non les données de densité lors de la suppression du plugin

Valeurs en production

La plupart des options sur le serveur de production utilisent les valeurs par défaut du code. Exceptions notables :

Option	Valeur en production
`irisolaris_map_lead_expiration`	180 jours
`irisolaris_plants_cache_version`	Auto-incrémenté

Services externes

Le plugin communique avec trois services externes.

Base Adresse Nationale (BAN)

Paramètre	Valeur
URL	`https://api-adresse.data.gouv.fr/search/`
Authentification	Aucune (API publique)
Fonction	Service de géocodage du gouvernement français
Utilisé par	JS frontend (direct) + proxy REST `/geocode`

Aucune configuration nécessaire — l'API BAN est un service public gratuit.

HubSpot Forms API

Paramètre	Valeur
URL	`https://api.hsforms.com/submissions/v3/integration/submit/{portalId}/{formId}`
Authentification	Portal ID + Form ID (côté serveur)
Fonction	Soumission de formulaire CRM pour les demandes d'adhésion
Utilisé par	Endpoint REST `/submit-join-form` (côté serveur)

OpenFreeMap

Paramètre	Valeur
URL	`https://tiles.openfreemap.org`
Authentification	Aucune (public)
Fonction	Tuiles cartographiques (raster + vectoriel)
Configuration	JSON de thème cartographique dans `assets/json/map-theme.json`

Allowlisting de l'API REST

Le site utilisant le plugin **Disable REST API**, l'accès anonyme à tous les endpoints REST est bloqué par défaut. Les endpoints Irisolaris suivants doivent être explicitement autorisés pour que la carte fonctionne :

Endpoint	Raison
`/wp-json/irisolaris-map/v1/plants`	Données cartographiques publiques (GeoJSON)
`/wp-json/irisolaris-map/v1/plants/*`	Données d'une centrale individuelle
`/wp-json/irisolaris-map/v1/data/map-theme`	Configuration du thème cartographique
`/wp-json/irisolaris-map/v1/density/*`	Recherches de densité pour l'éligibilité
`/wp-json/irisolaris-map/v1/geocode`	Proxy de géocodage d'adresses
`/wp-json/irisolaris-map/v1/check-eligibility`	Vérification d'éligibilité
`/wp-json/irisolaris-map/v1/submit-join-form`	Soumission du formulaire d'adhésion

Configurer cela dans **Réglages -> Disable REST API** dans l'administration WordPress.

Capacités du plugin

Plutôt que de créer de nouveaux rôles, le plugin ajoute des capacités personnalisées aux rôles WordPress existants lors de l'activation.

Administrateur — CRUD complet sur `irisolaris_plant` et `irisolaris_lead` (11 capacités chacun)

Éditeur — Lecture/modification de `irisolaris_plant` (6 capacités), lecture seule de `irisolaris_lead` (2 capacités)

5 Performance & Cache

5.1 Performance et cache

Pour la pile de cache au niveau du site (Redis, WP Super Cache, cache HTTP, monitoring et sauvegardes), voir **Pile de cache**.

Cache applicatif (Transients)

Le plugin utilise les transients WordPress pour mettre en cache les réponses GeoJSON, évitant ainsi la regénération à chaque requête :

Motif du transient	TTL	Fonction
`irisolaris_plants_geojson_{md5}`	7 jours (complet), 1 jour (filtré)	FeatureCollection GeoJSON mise en cache

La clé du transient inclut un hash MD5 composé de :

- Les paramètres de filtre (statut, région)
- Le numéro de version du cache (`irisolaris_plants_cache_version`)

Invalidation du cache

Stratégie d'incrémentation de version

```
Déclencheur (save/delete plant, changement de terme)
  └── API::invalidate_plants_cache()
      └── bump_cache_version()
          └── irisolaris_plants_cache_version += 1
              └── Requête suivante → nouveau MD5 → cache miss → requête fraîche
                  Les anciens transients expirent naturellement (TTL de 7 jours)
```

Déclencheurs :

- `save_post` (toute sauvegarde d'un `irisolaris_plant`)
- `delete_post` (toute suppression d'un `irisolaris_plant`)
- `set_object_terms` (changements de termes de taxonomie)

Comportement :

- Le compteur de version s'incrémente automatiquement
- Les nouvelles requêtes utilisent la nouvelle version dans le hash MD5, provoquant un cache miss
- Les anciens transients ne sont pas activement supprimés — ils expirent naturellement après leur TTL
- Avec le cache objet Redis, les transients sont stockés dans Redis plutôt que dans la base de données

Purge manuelle du cache

Les administrateurs peuvent purger manuellement le cache GeoJSON via :

- **Centrales PV -> Importer** -> bouton "Clear cache"
- Cela déclenche le handler AJAX `irisolaris_clear_cache`

Optimisations de performance à l'import

Les imports en masse de milliers de centrales peuvent être coûteux. Pendant ces opérations, le plugin applique plusieurs optimisations au niveau WordPress pour éviter les timeouts et réduire la charge sur la base de données.

Optimisation	Fonction
`wp_suspend_cache_invalidation()`	Empêche les reconstructions de cache pendant les opérations par lots
`wp_defer_term_counting()`	Diffère le comptage des termes de taxonomie jusqu'à la fin du lot
`update_meta_cache()`	Pré-charge le cache des métadonnées en masse pour réduire les requêtes individuelles
`update_object_term_cache()`	Pré-charge le cache des termes en masse
Taille de lot : 200 lignes	Évite les timeouts PHP lors des imports volumineux
SQL en masse pour la suppression	Contourne l'API WordPress des posts pour une suppression plus rapide

Caractéristiques de performance

Les sections ci-dessous résument comment les différentes couches de cache interagissent à l'exécution pour offrir des chargements de page rapides et des interactions cartographiques réactives.

Chargement de la carte

1. **Thème cartographique** — Mis en cache par le navigateur (cache HTTP, 30 min)
2. **GeoJSON** — Mis en cache via transient (7 jours), cache HTTP (30 min)
3. **Tuiles cartographiques** — Mises en cache par le navigateur depuis le CDN OpenFreeMap
4. **Clustering** — Côté client (MapLibre), réduit les éléments DOM pour plus de 2 000 centrales

Base de données

- **Cache objet Redis** — Réduit les requêtes en base de données pour les opérations WordPress répétées
- **Colonne `dens7` indexée** — Recherches de densité rapides lors des vérifications d'éligibilité
- **Requêtes meta `work_id`** — Utilisées pour la logique d'upsert lors des imports

6 Évolution

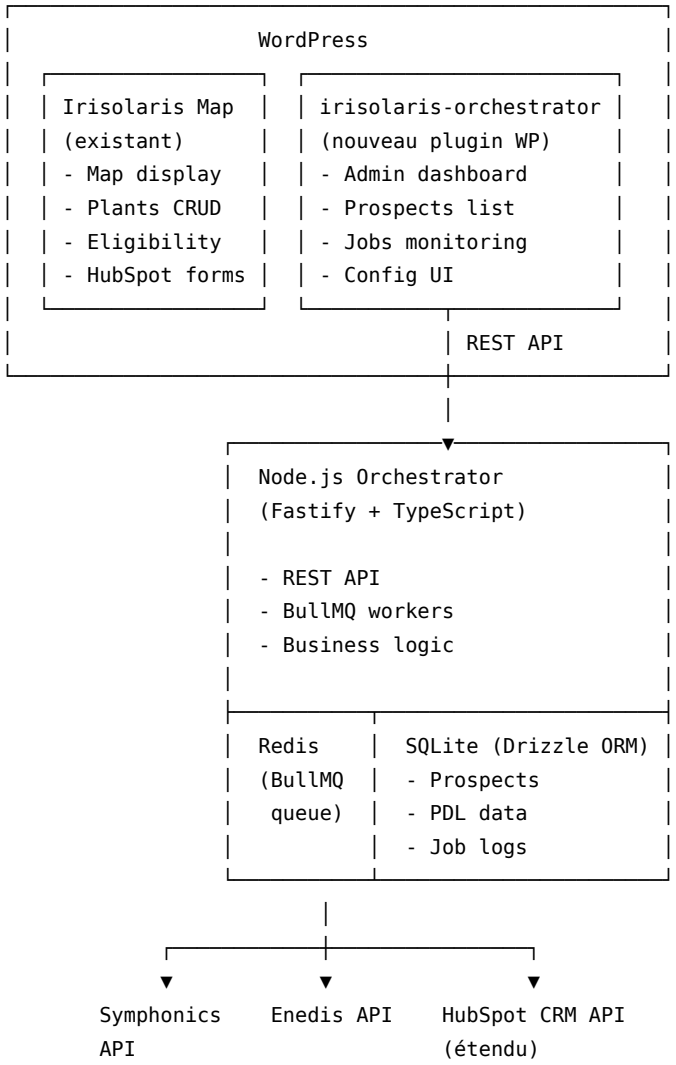
6.1 Architecture cible

Motivations

L'architecture cible répond au besoin d'**automatisation du processus ACC (Autoconsommation Collective)** :

- **État actuel :** Le processus ACC utilise un workflow actuel d'appels à l'API Symphonics pour la recherche de PDL, l'enrichissement de données Enedis et les mises à jour du CRM HubSpot
- **Volume :** Moins de 100 prospects/jour attendus, mais chacun nécessite plusieurs appels API asynchrones
- **Besoins métier :** Formulaire intelligent avec recherche de PDL en temps réel, pipeline contractuel automatisé, reporting et alertes

Composants cibles



Composant	Technologie	Fonction
`irisolaris-orchestrator`	Plugin WordPress (PHP)	Tableau de bord admin, liste des prospects, suivi des jobs, interface de configuration
Node.js Orchestrator	Fastify + TypeScript	API REST, workers BullMQ, logique métier
Redis (dédié)	Conteneur Docker	Backend de file d'attente de jobs (BullMQ)

Composant	Technologie	Fonction
SQLite	via Drizzle ORM	Prospects, données PDL, logs de jobs
Bull Board	Interface web	Tableau de bord de monitoring des files d'attente

Changements majeurs par rapport au système actuel

Domaine	Actuel (AS-IS)	Cible (TO-BE)
Traitement des formulaires	Soumission synchrone via HubSpot Forms	Pipeline asynchrone multi-étapes (formulaire -> recherche PDL -> enrichissement -> sync CRM)
Stockage des données	Post meta WordPress uniquement	WordPress + SQLite (données spécifiques à l'orchestrateur)
Traitement des jobs	Synchrone	Workers BullMQ avec logique de réessai
APIs externes	Géocodage BAN + HubSpot Forms	+ Symphonics + Enedis
Infrastructure	WordPress uniquement	WordPress + Docker (Node.js + Redis)
Monitoring	Prévu	Bull Board + tableau de bord admin

Nouvelles APIs externes

Service	Fonction
Symphonics API	Recherche de PDL par adresse, profils de contrats/consommation
Enedis API	Enrichissement de données énergétiques
HubSpot CRM API	Création/synchronisation de Contacts + Deals (extension de l'intégration actuelle limitée aux Forms)

Voir **APIs externes** pour plus de détails.

Impact sur Irisolaris Map

Le plugin Irisolaris Map existant continuera à fonctionner tel quel. L'orchestrateur est un **plugin séparé** qui étend la plateforme avec :

- Un nouveau plugin WordPress (`irisolaris-orchestrator`) pour l'interface d'administration
- Un service Node.js (Docker) pour le traitement asynchrone
- Aucune modification des fonctionnalités existantes de carte/éligibilité/import

L'intégration HubSpot Forms actuelle sera à terme remplacée par l'intégration CRM API de l'orchestrateur, permettant un suivi des prospects et une gestion du pipeline plus riches.

6.2 Orchestrateur Node.js

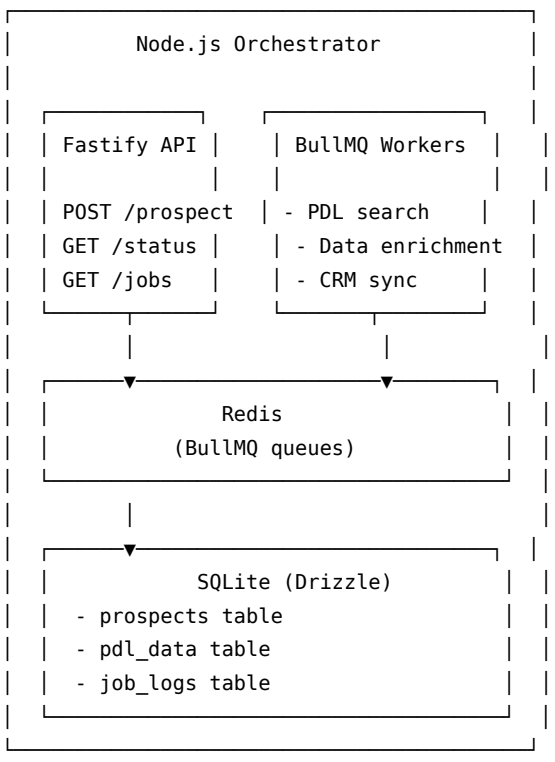
Vue d'ensemble

L'orchestrateur Node.js est un service prévu qui gérera l'automatisation asynchrone du processus ACC (autoconsommation collective). Il fonctionne aux côtés de WordPress en tant que conteneur Docker.

Pile technologique

Technologie	Version	Fonction
Fastify	—	Framework HTTP (API REST)
TypeScript	—	Développement typé
BullMQ	—	File d'attente de jobs avec backend Redis
Drizzle ORM	—	Accès à la base de données SQLite
SQLite	—	Stockage local des données (prospects, données PDL, logs de jobs)
Redis	Conteneur dédié	Backend de file d'attente BullMQ
Bull Board	—	Interface web de monitoring des files d'attente

Architecture



Pipeline de traitement des jobs

L'orchestrateur traite les prospects à travers un pipeline asynchrone multi-étapes :

```
1. Form Submission (from WordPress)
   └── Create prospect in SQLite
       └── Enqueue "pdl-search" job

2. PDL Search (BullMQ worker)
   ├── Call Symphonics API → search PDL by address
```

```
   ├── Store results in SQLite
   └── Enqueue "data-enrichment" job

3. Data Enrichment (BullMQ worker)
   ├── Call Enedis API → energy consumption data
   ├── Update prospect record
   └── Enqueue "crm-sync" job

4. CRM Sync (BullMQ worker)
   ├── Create/update HubSpot Contact
   ├── Create HubSpot Deal
   └── Update prospect status
```

Chaque étape est un job BullMQ indépendant avec :

- Logique de réessai automatique en cas d'échec
- Suivi de l'état du job (waiting -> active -> completed/failed)
- Monitoring via Bull Board

Stockage des données

Tables SQLite (via Drizzle ORM)

Prospects — Enregistrement central pour chaque demande ACC :

- Identité du prospect (nom, e-mail, téléphone, adresse)
- Référence à la centrale associée
- Statut de traitement
- Horodatages

PDL Data — Données de réponse de l'API Symphonics :

- Numéros de référence PDL
- Informations contractuelles
- Profils de consommation

Job Logs — Historique de traitement :

- Type de job, statut, horodatages
- Détails de l'erreur en cas d'échec
- Lien vers l'enregistrement du prospect

Intégration WordPress

Le plugin WordPress `irisolaris-orchestrator` fournit :

- **Tableau de bord admin** — Vue d'ensemble des prospects et du statut du pipeline de traitement
- **Liste des prospects** — Liste consultable/filtrable de toutes les demandes ACC
- **Suivi des jobs** — Visualisation du statut de traitement, réessai des jobs en échec
- **Interface de configuration** — Identifiants API, paramètres des files d'attente, politiques de réessai

La communication entre WordPress et le service Node.js se fait via des appels API REST.

Infrastructure

```
Docker Compose:
  ├── wordpress (existing)
  ├── orchestrator-api     (Fastify + TypeScript)
  ├── orchestrator-workers (BullMQ workers)
  └── redis-orchestrator   (dedicated Redis for BullMQ)
```

L'orchestrateur utilise une **instance Redis dédiée** séparée du Redis Object Cache de WordPress pour éviter les interférences.

État actuel

L'orchestrateur est en **phase de planification/spécification**. Aucun code n'a encore été déployé en production.

6.3 APIs externes (prévues)

Vue d'ensemble

L'architecture cible introduit trois nouvelles intégrations d'APIs externes (en plus du géocodage BAN et de HubSpot Forms existants) :

Service	Statut	Fonction
BAN (Base Adresse Nationale)	**Actif**	Géocodage d'adresses
HubSpot Forms API	**Actif**	Soumission de formulaires (actuel)
Symphonics API	**Prévu**	Recherche de PDL et données contractuelles
Enedis API	**Prévu**	Enrichissement de données énergétiques
HubSpot CRM API	**Prévu**	Gestion étendue des Contacts + Deals

Symphonics API

Fonction

L'API Symphonics donne accès aux données PDL (Point de Livraison) — les références des points de livraison d'électricité en France. Elle permet :

- **Recherche de PDL par adresse** — Trouver les points de raccordement électrique à proximité d'une adresse donnée
- **Profils contractuels** — Récupérer le type de contrat, la puissance souscrite et les détails du fournisseur
- **Données de consommation** — Accéder aux profils historiques de consommation électrique

Point d'intégration

L'orchestrateur Node.js appellera l'API Symphonics lors de l'étape "recherche PDL" du pipeline de traitement des prospects. Les résultats sont stockés dans SQLite pour les étapes de traitement suivantes.

Documentation de référence

La documentation complète de l'API est disponible dans le dépôt du plugin :

- `docs/process_update/symphonics_api/documentation.md`

Enedis API

Fonction

L'API Enedis fournit un enrichissement détaillé des données énergétiques :

- Profils de consommation électrique
- Détails de raccordement au réseau
- Données de comptage

Point d'intégration

Appelée lors de l'étape "enrichissement de données" du pipeline de traitement des prospects, après l'identification du PDL via Symphonics.

HubSpot CRM API (étendu)

Intégration actuelle

Le plugin Irisolaris Map existant utilise la **HubSpot Forms API** pour des soumissions de formulaires unidirectionnelles :

- `POST /submissions/v3/integration/submit/{portalId}/{formId}`
- 21 champs (7 client + 14 enrichis côté serveur)
- Le suivi du statut des soumissions et des enregistrements CRM est géré par l'extension prévue de l'orchestrateur

Extension prévue

L'orchestrateur utilisera la **HubSpot CRM API** complète pour des interactions plus riches :

Opération	Fonction
Create Contact	Nouvel enregistrement de prospect dans le CRM HubSpot
Update Contact	Enrichissement avec les données Symphonics/Enedis
Create Deal	Suivi du pipeline pour les contrats ACC
Update Deal	Progression à travers les étapes contractuelles
Associate Contact <-> Deal	Liaison du prospect à son deal ACC

Cela permet :

- La synchronisation bidirectionnelle (WordPress <-> HubSpot)
- Le suivi des étapes du pipeline
- La progression automatique des deals basée sur les résultats d'enrichissement des données
- Le reporting sur les taux de conversion et la santé du pipeline

Plan de migration

La transition des intégrations API actuelles vers les intégrations cibles :

1. **Phase 1 (actuelle) :** HubSpot Forms API uniquement — soumissions de formulaires unidirectionnelles
2. **Phase 2 (prévue) :** L'orchestrateur gère la soumission de formulaire -> recherche PDL -> enrichissement -> synchronisation CRM
3. **Phase 3 (prévue) :** Automatisation complète du pipeline avec monitoring et alertes

La soumission de formulaire Irisolaris Map existante continuera à fonctionner pendant la transition. L'intégration CRM de l'orchestrateur remplacera à terme la soumission directe via la Forms API.

Partie II

Stack WordPress

7 Environnement & Plugins

7.1 Environnement WordPress

Cette page decrit l'environnement d'hebergement de production du site Irisolaris Greentariff : la pile serveur, la configuration WordPress et le theme actif. Elle sert de reference pour toute personne qui configure, debogue ou reproduit l'environnement.

Infrastructure serveur

Le site fonctionne sur une pile LEMP standard avec Redis pour le cache objet persistant.

Derniere verification : fevrier 2026

Composant	Version	Role
Linux	Ubuntu LTS	Systeme d'exploitation
nginx	1.28.1	Serveur web / reverse proxy
PHP	8.3.30 (CLI, NTS)	Langage cote serveur
MariaDB	10.11.14	Base de donnees
Redis	7.0.15	Backend du cache objet

Parametres du site WordPress

Les parametres suivants definissent l'installation WordPress et sa structure d'URL.

Parametre	Valeur
URL du site	`https://irisolaris-greentariff.com`
WordPress	6.9.1
Locale	`fr_FR` (francais)
Fuseau horaire	`Europe/Paris`
Multisite	Non
Permaliens	`/%postname%/`
Page d'accueil	Statique — "Accueil"
Page du blog	"Latest News"

Theme actif

Le site utilise Blocksy avec un theme enfant pour les personnalisations.

Theme	Version	Role
Blocksy	2.1.27	Theme parent
blocksy-child	—	Theme enfant actif

Blocksy est un theme moderne et performant avec prise en charge complete de l'edition du site. Le theme enfant permet des personnalisations sans modifier le theme parent.

Pour la liste complete des plugins installes, consultez **Plugins actifs**. Pour les details sur les couches de cache, consultez **Pile de cache**.

7.2 Plugins actifs

Le serveur de production execute 16 plugins actifs en plus d'Irisolaris Map. Ils sont organises ci-dessous par categorie, avec des notes sur leur interaction avec le plugin.

Securite (3)

Plugin	Role
NinjaFirewall (WP Edition)	Pare-feu applicatif web (WAF)
Disable REST API	Bloque l'acces anonyme a l'API REST (les endpoints doivent etre autorises explicitement)
WP Anti-Clickjack	En-tete X-Frame-Options

Disable REST API bloque l'acces anonyme a tous les endpoints REST de WordPress. Tout endpoint que le site doit exposer publiquement — comme les routes d'Irisolaris Map — doit etre explicitement autorise. Consultez **Autorisation de l'API REST** pour les entrees requises.

Performance (3)

Plugin	Role
Redis Object Cache	Cache objet persistant adosse a Redis
WP Super Cache	Cache de pages statiques complet
WP-Optimize	Nettoyage de la base de donnees, compression d'images

Consultez **Pile de cache** pour les details sur les couches Redis et WP Super Cache.

Constructeurs de pages (2)

Plugin	Role
GreenShift	Blocs Gutenberg avec animations
Stackable (Premium)	Bibliotheque etendue de blocs Gutenberg

Autres

Les plugins restants gerent les fonctionnalites du theme, le SEO, l'analytique, les sauvegardes et la conformite.

Plugin	Role
Blocksy Companion (Premium)	Fonctionnalites complementaires du theme
Complianz	Consentement aux cookies RGPD/CCPA
Site Kit de Google	Tableau de bord analytique
UpdraftPlus	Sauvegarde automatique (toutes les 4 heures)
Yoast SEO	Gestion du SEO
Yoast Duplicate Post	Utilitaire de duplication d'articles

UpdraftPlus execute des sauvegardes automatiques toutes les 4 heures (base de donnees + fichiers). Consultez **Sauvegarde et restauration** pour les procedures de restauration.

7.3 Pile de cache

Le site utilise une strategie de cache multicouche. Le plugin Irisolaris Map ajoute son propre cache au niveau applicatif par-dessus — consultez **Performance et cache du plugin** pour le cache specifique au plugin.

Flux d'une requete

Le schema ci-dessous montre comment une requete type traverse les couches de cache avant d'atteindre la base de donnees.

```
Navigateur → Cache HTTP (30 min) → Cache de pages (WP Super Cache)
          → Cache objet (Redis) → Cache transitoire (GeoJSON)
          → Base de donnees (MariaDB)
```

Cache HTTP

Les reponses de l'API REST incluent des en-tetes de cache definis par le plugin Irisolaris Map :

En-tete	Valeur	Role
`Cache-Control`	`public, max-age=1800`	Cache navigateur/CDN pendant 30 minutes
`ETag`	Hash du contenu	Prise en charge des requetes conditionnelles
`304 Not Modified`	—	Renvoye lorsque l'ETag correspond

Cache de pages (WP Super Cache)

WP Super Cache genere des fichiers HTML statiques pour les pages non dynamiques. Nginx sert ces fichiers directement, sans passer par PHP. Cela beneficie aux pages marketing et aux articles de blog, mais ne s'applique pas aux endpoints de l'API REST.

Cache objet (Redis)

Le plugin Redis Object Cache remplace le cache objet en memoire par defaut de WordPress par un stockage persistant adosse a Redis. Les donnees mises en cache incluent :

- Les resultats des requetes a la base de donnees
- Les options WordPress
- Les recherches de metadonnees d'articles
- Les requetes sur les termes de taxonomie

Comme Redis persiste entre les requetes, les donnees en cache sont partagees entre les chargements de pages — contrairement au cache objet par defaut de WordPress qui est reconstruit a chaque requete.

Surveillance

Plusieurs outils sont disponibles pour surveiller l'etat et les performances du cache.

Diagnostics disponibles :

- **WP-Optimize** — Sante de la base de donnees, taille des tables, suggestions d'optimisation
- **Redis Object Cache** — Etat de la connexion, ratio succes/echec, utilisation memoire (via la page de reglages du plugin)
- **WP Super Cache** — Statistiques du cache et nombre de fichiers (via la page de reglages du plugin)
- **PHP error_log()** — Debogage au niveau applicatif

Sauvegarde et restauration

UpdraftPlus execute des sauvegardes automatiques toutes les 4 heures (base de donnees + fichiers). En cas de probleme :

1. Restaurer a partir de la sauvegarde UpdraftPlus la plus recente
2. Ou redeployer manuellement le plugin depuis le depot (voir **Build et production**)

Partie III

Infrastructure Serveur

8 Installation & Configuration

8.1 Installation et configuration du serveur

Introduction

Ce guide complet vous accompagne tout au long du processus de mise en place d'un environnement serveur sécurisé et optimisé pour WordPress. De l'accès initial au serveur jusqu'au réglage des performances, chaque section s'appuie sur la précédente pour créer une plateforme d'hébergement robuste. Que vous soyez débutant ou administrateur système expérimenté, ce guide fournit des instructions détaillées intégrant les meilleures pratiques en matière de sécurité et de performance à chaque étape.

Le guide est organisé selon une séquence logique, en commençant par l'accès de base au serveur et en progressant graduellement vers des sujets plus avancés comme le durcissement et l'optimisation. Chaque section comprend des commandes pratiques, des exemples de configuration et des explications sur les raisons pour lesquelles certaines approches sont recommandées.

Première connexion

Apprenez à vous connecter à votre serveur pour la première fois via SSH, à gérer le changement initial de mot de passe et à configurer votre éditeur de texte préféré pour modifier les fichiers de configuration.

Accéder au guide de première connexion

Vérifier votre statut utilisateur

Déterminez si vous êtes connecté en tant que `root` ou en tant qu'utilisateur non-root et comprenez les implications de sécurité de chaque cas. Apprenez à créer un nouvel utilisateur avec des privilèges administratifs pour une meilleure sécurité.

Accéder au guide de vérification du statut utilisateur

Désactiver la connexion par mot de passe

Renforcez la sécurité du serveur en mettant en place l'authentification par clé SSH et en désactivant la connexion par mot de passe. Cette section couvre la génération de paires de clés SSH, la copie des clés publiques vers le serveur, la simplification de la connexion avec la configuration SSH et la désactivation de l'authentification par mot de passe.

Accéder au guide de sécurité SSH

Ajouter un pare-feu

Protégez votre serveur en configurant UFW (Uncomplicated Firewall) pour contrôler le trafic entrant et sortant. Apprenez à autoriser les services essentiels tout en bloquant les connexions potentiellement malveillantes.

Accéder au guide de configuration du pare-feu

Ajouter Fail2ban

Mettez en place Fail2ban pour protéger votre serveur contre les attaques par force brute et les activités malveillantes. Cette section couvre l'installation, la configuration et la gestion de ce logiciel de prévention d'intrusion qui bloque automatiquement les adresses IP suspectes.

Accéder au guide Fail2ban

Durcissement du serveur

Appliquez des mesures de sécurité essentielles et des optimisations pour protéger et améliorer les performances de votre serveur. Les sujets abordés incluent le réglage du fuseau horaire correct, la configuration de l'espace swap et la mise en place de diverses améliorations de sécurité au niveau système.

Accéder au guide de durcissement du serveur

Associer un nom de domaine à votre serveur

Configurez les enregistrements DNS pour relier votre nom de domaine à l'adresse IP de votre serveur. Apprenez les enregistrements A, les enregistrements CNAME et comment vérifier et dépanner votre configuration DNS.

Accéder au guide de configuration du domaine

Installation de la pile LEMP

Mettez en place un environnement d'hébergement web performant et efficace en installant et configurant Linux, Nginx, MariaDB et PHP (pile LEMP). Cette base fournit les composants nécessaires à l'exécution de WordPress.

Accéder au guide d'installation de la pile LEMP

Configuration de la messagerie

Configurez les capacités d'envoi d'e-mails de votre serveur en utilisant MSMTP, permettant à WordPress d'envoyer des notifications essentielles, des réinitialisations de mot de passe et des soumissions de formulaires de contact sans avoir à gérer un serveur de messagerie complet.

Accéder au guide de configuration de la messagerie

Durcissement et optimisation de la pile LEMP

Renforcez la sécurité et les performances de votre installation Nginx, MariaDB et PHP grâce à des configurations avancées. Apprenez à réduire les surfaces d'attaque, améliorer les performances et optimiser l'utilisation des ressources.

Accéder au guide d'optimisation de la pile LEMP

Configuration des blocs serveur Nginx

Configurez des hôtes virtuels pour servir plusieurs sites web depuis un seul serveur. Comprenez la structure des blocs serveur et comment les configurer correctement pour les sites WordPress.

Accéder au guide des blocs serveur Nginx

Installation de WordPress

Suivez un guide étape par étape pour installer WordPress sur votre serveur LEMP, incluant la création de la base de données, la configuration des fichiers et le réglage correct des permissions.

Accéder au guide d'installation de WordPress

Durcissement de la sécurité WordPress

Mettez en place des mesures de sécurité complètes pour protéger votre installation WordPress contre les menaces courantes. Apprenez l'isolation des utilisateurs, le contrôle d'accès, le chiffrement, le filtrage des requêtes et le durcissement de l'application.

Accéder au guide de sécurité WordPress

Optimisation des performances WordPress

Améliorez la vitesse et l'efficacité de votre site WordPress grâce à la mise en cache, aux optimisations de configuration et à la gestion des ressources. Apprenez des techniques pour réduire la charge serveur, améliorer les temps de chargement des pages et enrichir l'expérience utilisateur.

Accéder au guide d'optimisation WordPress

8.2 1. Première connexion

```
ssh username@server_ip
```

Saisissez votre mot de passe lors de la première connexion, un changement de mot de passe sera demandé, puis vérifiez si vous êtes connecté en tant qu'utilisateur root.

Changer votre éditeur par défaut

Après vous être connecté, vous pouvez définir votre éditeur préféré pour modifier les fichiers de configuration. Utilisez la commande suivante :

```
sudo update-alternatives --config editor
```

Cela affichera une liste des éditeurs disponibles. Sélectionnez le numéro correspondant à votre choix (par exemple, `vim`, `nano`, etc.).

```
There are 4 choices for the alternative editor (providing /usr/bin/editor).

  Selection    Path                Priority   Status
------------------------------------------------------------

* 0            /bin/nano            40        auto mode
  1            /bin/ed             -100       manual mode
  2            /bin/nano            40        manual mode
  3            /usr/bin/vim.basic   30        manual mode
  4            /usr/bin/vim.tiny    15        manual mode

Press <enter> to keep the current choice[*], or type selection number:
```

8.3 2. Vérifier votre statut utilisateur

Si vous êtes connecté en tant que root, vous devrez créer un nouveau compte utilisateur non privilégié.

Vérifier si vous êtes root

Exécutez la commande suivante pour vérifier votre nom d'utilisateur actuel :

```
whoami
```

- Si le résultat est `root`, vous êtes connecté en tant qu'utilisateur root.
- Vous pouvez également vérifier l'UID (identifiant utilisateur) de votre compte en exécutant :

```
id
```

- Si l'UID est `0`, vous êtes connecté en tant qu'utilisateur root.

Vérifier les privilèges root pour un utilisateur non-root

Si vous n'êtes pas connecté en tant que `root`, vérifiez si votre utilisateur fait déjà partie du groupe `sudo` en exécutant :

```
groups
```

- Cela affichera la liste des groupes auxquels votre utilisateur appartient. Si `sudo` figure dans la liste, votre utilisateur dispose de droits administratifs (privilégiés).

Vous pouvez également vérifier en exécutant une commande qui nécessite les privilèges `sudo`. Par exemple :

```
sudo whoami
```

- Si un mot de passe vous est demandé et que vous voyez ensuite le résultat `root`, cela signifie que votre utilisateur a accès à `sudo`.

Que faire selon les résultats

- **Cas 1 : Votre utilisateur est root (résultat de `whoami` = `root` ou `id` affiche l'UID `0`)**
  - Créez un nouveau compte utilisateur non privilégié.
  - Déconnectez-vous du compte root et utilisez le compte nouvellement créé pour la suite des opérations.
- **Cas 2 : Votre utilisateur n'a pas d'accès privilégié (pas dans le groupe `sudo`)**
  - Vous pouvez demander à l'administrateur système de vous accorder les privilèges `sudo` ou vous connecter en tant que root pour ajouter votre utilisateur au groupe `sudo` (si vous disposez des identifiants root).
- **Cas 3 : Votre utilisateur fait déjà partie du groupe `sudo`**
  - Vous pouvez poursuivre votre travail car vous disposez déjà d'un accès administratif sans être directement l'utilisateur root.

Ce processus vous assure de commencer à travailler sur le serveur en toute sécurité, en respectant les bonnes pratiques de gestion des privilèges utilisateur.

Créer un nouveau compte utilisateur non privilégié

Dans les systèmes Linux, la gestion des privilèges root est une partie essentielle de l'administration des utilisateurs. Il existe deux méthodes courantes pour accorder les privilèges root : la commande `visudo` et la commande `usermod`.

Quand utiliser `visudo` ou `usermod` - Utilisez `visudo` pour des configurations détaillées et sécurisées. - Utilisez `usermod` pour une gestion simple et rapide des privilèges basée sur les groupes.

Utilisation de la commande `usermod`

La commande `usermod` est un moyen simple d'ajouter un utilisateur au groupe sudo (ou à d'autres groupes administratifs). Elle est idéale lorsque la configuration du système permet la gestion des privilèges par groupes.

Pour ajouter l'utilisateur au groupe `sudo`, utilisez la commande suivante :

```
sudo usermod -aG sudo username
```

- Remplacez `username` par le nom d'utilisateur réel de la personne que vous souhaitez ajouter au groupe sudo.
- L'option `-a` garantit que l'utilisateur est ajouté au groupe sans être retiré des autres groupes.
- L'option `-G` spécifie le groupe auquel l'utilisateur est ajouté (`sudo` dans ce cas).

```
sudo adduser username
```

Vous devez fournir un mot de passe pour le nouveau compte. Ensuite, il vous sera demandé de saisir les détails du nouvel utilisateur, ce qui est facultatif.

Créez un nouveau compte utilisateur (remplacez username par le nom réel du compte utilisateur) :

```
sudo adduser username
```

Ajoutez le nouveau compte utilisateur au groupe sudo :

```
sudo usermod -aG sudo username
```

Vous pouvez vérifier l'appartenance au groupe avec la commande suivante :

```
groups username
```

Exemple :

```
groups ubuntu
```

```
ubuntu sudo users
```

Le nouveau compte utilisateur est maintenant un utilisateur sudo et peut exécuter toutes les commandes.

Cela est configuré par le paramètre par défaut du groupe sudo :

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Note

Les configurations correspondantes se trouvent dans `/etc/sudoers` et le répertoire `/etc/sudoers.d` respectivement.

Utilisation de la commande visudo

Voici comment ajouter les privilèges root pour un utilisateur avec `visudo` :

1. **Ouvrir le fichier `sudoers` :**

```
sudo visudo
```

Cette commande garantit que le fichier `sudoers` est vérifié pour les erreurs de syntaxe avant l'enregistrement, ce qui évite les erreurs de configuration potentielles. 2. **Ajouter l'utilisateur avec tous les privilèges :** Localisez la section du fichier où les privilèges utilisateur sont définis. Pour accorder à un utilisateur (`username`) les privilèges root complets, ajoutez la ligne suivante :

```
username ALL=(ALL:ALL) ALL
```

- Remplacez `username` par le nom d'utilisateur réel.
- Cette ligne permet à l'utilisateur d'exécuter n'importe quelle commande en tant que n'importe quel utilisateur ou groupe, avec les privilèges root.

3. **Enregistrer et quitter**
   - Si vous utilisez `vim` comme éditeur par défaut, appuyez sur Esc, tapez `:wq` et appuyez sur Entrée pour enregistrer et fermer.
   - Pour `nano`, appuyez sur Ctrl+O pour enregistrer, puis Ctrl+X pour quitter.

Comprendre la configuration `sudo` sans mot de passe

Note

Il existe des scénarios où aucun mot de passe ne vous sera demandé lors de l'utilisation de `sudo`. Cela se produit généralement en raison de configurations spécifiques dans le fichier `sudoers` ou de paramètres associés. Voici les principales raisons :

L'option `NOPASSWD` dans `sudoers`

Si votre utilisateur ou groupe (par exemple, `sudo`) est configuré avec l'option `NOPASSWD`, les commandes sudo peuvent être exécutées sans saisir de mot de passe. Ces configurations sont définies soit dans le fichier principal `sudoers`, soit dans les fichiers du répertoire `/etc/sudoers.d`.

Cette configuration peut être intentionnelle à des fins d'automatisation ou de commodité, mais elle doit être utilisée avec prudence pour éviter de compromettre la sécurité. Examinez et modifiez toujours les configurations `sudo` de manière responsable.

Réactiver les demandes de mot de passe

Pour réactiver les demandes de mot de passe pour les commandes `sudo`, suivez ces étapes :

1. **Modifier le fichier `sudoers` de manière sécurisée** avec `visudo` : `shell script  sudo visudo`
2. **Localiser et supprimer/modifier la règle `NOPASSWD`** :
   - Trouvez les lignes contenant `NOPASSWD`. Par exemple : `shell script  username ALL=(ALL) NOPASSWD:ALL`
- Remplacez `NOPASSWD:ALL` par `ALL`. Les lignes modifiées ressembleront à : `username ALL=(ALL) ALL`
3. **Enregistrer et quitter**

Vérifier et configurer le `PermitRootLogin`

Étapes pour vérifier et définir `PermitRootLogin` à `no`

Localiser le fichier de configuration SSH

Le fichier de configuration principal de SSH se trouve généralement à l'emplacement `/etc/ssh/sshd_config`.

Pour vérifier le comportement actuel de `PermitRootLogin`, exécutez : `bash  sudo grep PermitRootLogin /etc/ssh/sshd_c`

- Si le résultat est commenté (`# PermitRootLogin ...`), alors le comportement par défaut du système est appliqué (souvent `yes`).

Modifier la configuration SSH

Ouvrez le fichier dans un éditeur :

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

Recherchez la directive `PermitRootLogin`. Si elle est présente mais commentée (précédée de `#`), décommentez-la en supprimant le `#`. Modifiez la ligne (ou ajoutez-la si elle n'est pas présente) pour obtenir :

```
PermitRootLogin no
```

Cela désactive la connexion root via SSH.

- **Vérifier le dossier inclus pour les configurations supplémentaires**
  - De nombreuses distributions prennent en charge l'inclusion de fichiers de configuration supplémentaires depuis un dossier. Par exemple : `bash Include /etc/ssh/sshd_config.d/*.conf`
  - Pour vérifier si des fichiers de configuration supplémentaires sont inclus, recherchez une directive `Include` dans `sshd_config` : `bash grep -i include /etc/ssh/sshd_config`
  - SSH lit les configurations dans l'ordre suivant :
    1. Le **fichier principal** (`/etc/ssh/sshd_config`) est lu en premier.
    2. Les configurations des **fichiers du dossier inclus** (comme `/etc/ssh/sshd_config.d/*.conf`) sont lues ensuite.
    3. **Paramètres en conflit** : la dernière ligne lue a la priorité.
  - Cela signifie que si le fichier de configuration principal et les fichiers inclus spécifient tous deux `PermitRootLogin`, la valeur du dernier fichier (ou ligne) traité sera appliquée. Exemple : si `/etc/ssh/sshd_config` contient : `bash PermitRootLogin yes` et `/etc/ssh/sshd_config.d/custom.conf` contient : `bash PermitRootLogin no` Alors le système appliquera `PermitRootLogin no`, car cette valeur est lue en dernier.
  - Voir **Annexe A** pour plus de détails sur la priorité des fichiers de configuration SSH.

Si une telle ligne existe, tous les fichiers `.conf` dans `/etc/ssh/sshd_config.d/` (ou le dossier spécifié) seront également appliqués.

Appliquer la nouvelle configuration

1. **Vérifier la configuration.** Après avoir effectué les modifications, vérifiez la syntaxe du fichier de configuration : `bash sudo sshd -t` S'il n'y a pas d'erreurs, poursuivez.
2. **Redémarrer le service SSH.** Appliquez les modifications en redémarrant le service SSH : `bash sudo systemctl restar` :::caution Assurez-vous d'avoir un utilisateur non-root avec des privilèges `sudo` configuré au préalable pour éviter d'être bloqué. :::
3. **Tester la configuration.** Essayez de vous connecter en tant que `root` via SSH pour confirmer que l'accès est refusé : `bash ssh root@your_server_ip` —

Annexe A. Comprendre la priorité des fichiers de configuration SSH

1. **Ordre de lecture** :
   - SSHD lit `/etc/ssh/sshd_config` **EN PREMIER**
   - Le démon SSH **traite les directives `Include` lorsqu'elles sont rencontrées** dans le fichier `sshd_config`.
   - Si la directive `Include /etc/ssh/sshd_config.d/*.conf` apparaît au milieu de `sshd_config`, les fichiers dans `/etc/ssh/sshd_config.d/` sont traités **immédiatement à ce moment-là** en **ordre lexicographique**.
2. **La première valeur l'emporte** :
   - La **première occurrence** d'une directive de configuration (par exemple, `PermitRootLogin`) est appliquée.
   - Toute définition ultérieure est **ignorée**, quel que soit l'ordre des fichiers.

Annexe A.1 Exemple

Supposons ces fichiers inclus dans `/etc/ssh/sshd_config` :

`shell script Include /etc/ssh/sshd_config.d/*.conf`

Structure du répertoire :

```
/etc/ssh/sshd_config.d/
├── 50-cloud-init.conf
└── 60-cloudimg-settings.conf
```

Contenu de chaque fichier :

- **50-cloud-init.conf** : `shell script  PermitRootLogin yes`
- **60-cloudimg-settings.conf** : `shell script  PermitRootLogin no`

Résultat : Le serveur SSH applique `PermitRootLogin yes` car cette valeur est définie EN PREMIER dans `50-cloud-init.conf`, même si `60-cloudimg-settings.conf` la définit à `no`.

8.4 3. Désactiver la connexion par mot de passe

Générer une paire de clés SSH

Pour accéder de manière sécurisée à un serveur distant, une paire de clés SSH est utilisée comme mécanisme d'authentification. Cela implique deux clés : 1. Une **clé privée**, que vous conservez en sécurité sur votre machine locale. 2. Une **clé publique**, qui est partagée avec le serveur.

Cela élimine le besoin de mots de passe lors de la connexion et améliore considérablement la sécurité.

Vous pouvez suivre les instructions de la section **Générer une paire de clés SSH** ou **Générer une paire de clés SSH dans un sous-dossier personnalisé**, selon vos besoins :

- Utilisez **Générer une paire de clés SSH** pour générer et stocker la clé à l'emplacement par défaut.
- Utilisez **Générer une paire de clés SSH dans un sous-dossier personnalisé** pour générer et organiser la clé dans un sous-dossier personnalisé pour une meilleure gestion.

Générer une paire de clés SSH

> Cette section vous guide dans la génération d'une paire de clés SSH standard pouvant être utilisée pour s'authentifier auprès de serveurs distants.

Exécutez la commande suivante sur votre machine locale pour générer une paire de clés SSH :

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- `-t rsa` : Spécifie l'algorithme RSA pour la clé.
- `-b 4096` : Définit la taille de la clé à 4096 bits (utilisez `2048` si vous préférez une clé plus petite).
- `-C "your_email@example.com"` : Ajoute un commentaire à la clé, généralement votre adresse e-mail, pour une identification plus facile.

Lorsque vous y êtes invité :

- Entrez un nom de fichier (ou appuyez sur Entrée pour utiliser l'emplacement par défaut, par exemple, `~/.ssh/id_rsa`).
- (Facultatif) Fournissez une phrase de passe pour la clé privée pour une sécurité supplémentaire.

Cela crée :

- Une clé privée (par exemple, `~/.ssh/id_rsa`).
- Une clé publique (par exemple, `~/.ssh/id_rsa.pub`).

Générer une paire de clés SSH dans un sous-dossier personnalisé

> Cette approche est utile si vous devez gérer plusieurs clés SSH pour différents serveurs ou projets. Elle vous permet de stocker les clés SSH dans des sous-dossiers personnalisés, offrant une manière structurée d'organiser et de gérer vos clés à travers différents environnements.

Vous pouvez générer une clé SSH dans un sous-dossier au sein de `.ssh/` pour garder les choses organisées et gérer facilement plusieurs clés pour différents projets ou serveurs.

Pour générer une clé SSH dans un sous-dossier personnalisé à l'intérieur du répertoire `.ssh/`, suivez ces étapes :

`.ssh/` : 1. **Créer le sous-dossier** (s'il n'existe pas déjà) :

```
mkdir -p ~/.ssh/my_custom_folder
```

Remplacez `my_custom_folder` par le nom de dossier souhaité. 1. **Générer la clé dans le sous-dossier personnalisé** : Exécutez la commande `ssh-keygen` en spécifiant le chemin complet du fichier de clé :

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com" -f ~/.ssh/my_custom_folder/id_rsa
```

Vérifier les permissions correctes Assurez-vous que votre dossier `.ssh/`, le sous-dossier à l'intérieur et les fichiers de clés associés ont les permissions correctes :

```
chmod 700 ~/.ssh/
chmod 700 ~/.ssh/my_custom_folder/
chmod 600 ~/.ssh/my_custom_folder/id_rsa
chmod 644 ~/.ssh/my_custom_folder/id_rsa.pub
```

Copier la clé publique sur le serveur

Utilisez la commande suivante pour copier votre clé publique sur le serveur :

```
ssh-copy-id -i .ssh/id_rsa.pub username@server_ip
```

- Remplacez `username` par votre nom d'utilisateur sur le serveur.
- Remplacez `server_ip` par l'adresse IP ou le domaine du serveur.

Le mot de passe de votre serveur vous sera demandé. Une fois authentifié, la clé publique sera ajoutée sur le serveur dans `~/.ssh/authorized_keys` pour cet utilisateur.

ssh -i ~/my_custom_folder/id_rsa username@server_ip

Tester la connexion

Lors de la connexion au serveur avec la clé personnalisée, vous devrez peut-être spécifier explicitement le chemin de la clé privée si elle n'est pas détectée automatiquement.

```
ssh -i ~/.ssh/id_rsa username@server_ip
```

Simplifier la connexion SSH avec les paramètres de configuration

Pour faciliter votre processus de connexion SSH, vous pouvez configurer le client SSH pour qu'il mémorise et utilise automatiquement votre/vos clé(s) SSH personnalisée(s) pour des serveurs spécifiques en modifiant le fichier `~/.ssh/config`. Voici comment procéder :

Étapes pour ajouter la configuration SSH

1. **Modifier/Créer le fichier de configuration SSH** : Ouvrez ou créez le fichier `~/.ssh/config` en utilisant votre éditeur de texte favori (comme `vim` ou `vim`).

```
   vim ~/.ssh/config
```

1. **Ajouter une entrée d'hôte SSH** : Ajoutez la configuration suivante au fichier :

```
Host server-alias
   HostName server_ip
   User username
   IdentityFile ~/.ssh/id_rsa
   ServerAliveInterval 60
   ServerAliveCountMax 240
```

Remplacez :

- `server-alias` par un nom que vous préférez utiliser comme alias pour ce serveur (par exemple, `myserver`).
- `server_ip` par l'adresse IP ou le domaine du serveur.
- `username` par votre nom d'utilisateur sur le serveur.
- `~/.ssh/id_rsa` par le chemin vers votre clé privée (ajustez si vous avez généré une clé dans un sous-dossier personnalisé, par exemple, `~/.ssh/my_custom_folder/id_rsa`).

- `ServerAliveInterval 60` : Envoie un paquet keep-alive au serveur toutes les 60 secondes.
- `ServerAliveCountMax 240` : Permet à la connexion de rester active pendant 4 heures (240 x 60).

1. **Enregistrer et quitter** : Enregistrez le fichier et quittez votre éditeur.
2. **Définir les permissions correctes** : Assurez-vous que le fichier `~/.ssh/config` a les permissions correctes : `bash chmod 600 ~/.ssh/config`
3. **Tester la connexion SSH** : Vous pouvez maintenant vous connecter au serveur simplement en utilisant l'alias configuré :

```
ssh server-alias
```

Cela utilisera automatiquement la clé et le nom d'utilisateur spécifiés pour la connexion.

Exemple avec une clé et un dossier personnalisés

Si vous avez généré la clé dans un sous-dossier personnalisé (par exemple, `~/.ssh/my_custom_folder`), la configuration ressemblera à ceci :

```
Host server-alias
    HostName server_ip
    User username
    IdentityFile ~/.ssh/my_custom_folder/id_rsa
    ServerAliveInterval 60
    ServerAliveCountMax 240
```

Maintenant, exécutez `ssh server-alias` pour vous connecter rapidement sans spécifier explicitement le nom d'utilisateur ou le chemin de la clé.

Désactiver l'authentification par mot de passe sur le serveur

Pour définir "PasswordAuthentication" à "no" sur le serveur, vous devez modifier le fichier de configuration du serveur SSH (`sshd_config`). Cela garantit que seule l'authentification par clé est autorisée, renforçant davantage la sécurité du serveur. Voici comment procéder :

Désactiver l'authentification par mot de passe sur le serveur

1. **Accéder au serveur** : Connectez-vous à votre serveur via SSH : `shell script ssh username@server_ip`
2. **Modifier le fichier de configuration SSH** : Ouvrez le fichier `sshd_config` avec un éditeur de texte (par exemple, `vim`, `vim`) : `shell script sudo vim  sudo vim /etc/ssh/sshd_config.d/50-cloud-init.conf`
3. **Trouver et mettre à jour le paramètre PasswordAuthentication** : Recherchez la ligne `PasswordAuthentication` dans le fichier. Si elle n'existe pas, ajoutez-la. Définissez sa valeur à `no` :

```
PasswordAuthentication no
```

4. (*Facultatif*) Assurez-vous également que `PubkeyAuthentication` est défini à `yes` pour autoriser l'authentification par clé :

```
PubkeyAuthentication yes
```

   L'authentification par clé publique (authentification par clé SSH) est activée par défaut. Cela signifie que vous n'avez pas besoin d'ajouter ou de modifier explicitement la ligne `PubkeyAuthentication yes` sauf si votre configuration a été modifiée ou si vous souhaitez vous assurer explicitement qu'elle est définie.

5. **Enregistrer et quitter** :
6. **Redémarrer le service SSH** : Redémarrez le serveur SSH pour appliquer les modifications : `shell script sudo systemctl`
7. **Tester la configuration** : Assurez-vous que vous pouvez vous connecter au serveur avec votre clé avant de fermer votre session SSH actuelle : `shell script ssh -i ~/.ssh/id_rsa username@server_ip`

Vérifier que la connexion par mot de passe est désactivée

Pour confirmer les modifications, essayez de vous connecter au serveur sans fournir de clé. La connexion devrait être refusée : `shell script ssh username@server_ip`

```
shell script username@server_ip: Permission denied (publickey).
```

Si tout fonctionne comme prévu, votre serveur est maintenant sécurisé et l'authentification par mot de passe est désactivée.

8.5 4. Ajouter un pare-feu

UFW

Installation et configuration d'UFW (Uncomplicated Firewall) UFW est une interface conviviale pour gérer les règles de pare-feu sous Linux.

Vérifier l'état actuel :

```
sudo ufw status
```

Permet de s'assurer qu'UFW est installé et affiche s'il est actif ou non. Dans ce cas, l'état est "Inactive", ce qui signifie que le pare-feu n'est pas encore activé.

Définir les règles par défaut :

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

Ces commandes définissent les règles de base :

- *Refuser le trafic entrant* : Bloque toutes les connexions entrantes sauf celles explicitement autorisées.
- *Autoriser le trafic sortant* : Permet toutes les requêtes sortantes depuis le serveur.

Autoriser le trafic entrant sur des ports spécifiques : Ces commandes ouvrent les ports requis pour les services essentiels :

```
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https/tcp
sudo ufw allow https/udp
```

- Port 22 (SSH) : Nécessaire pour la gestion à distance.
- Port 80 (HTTP) : Permet aux visiteurs d'accéder aux pages web via HTTP.
- Ports 443 (HTTPS TCP/UDP) : Active la communication sécurisée via HTTPS. L'ouverture de ports spécifiques uniquement réduit la surface d'attaque du serveur.

Activer le pare-feu :

```
sudo ufw enable
```

Active UFW pour appliquer les règles de pare-feu configurées.

Revérifier l'état :

```
sudo ufw status
```

Affiche les règles actives appliquées par UFW. Par exemple :

```
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
443/udp                    ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
443/udp (v6)               ALLOW       Anywhere (v6)
```

Les ports listés et leurs protocoles (par exemple, TCP/UDP) confirment que les règles sont en place. Les règles IPv6 sont incluses si le serveur les prend en charge.

Pare-feu spécifique au fournisseur

De nombreux fournisseurs de serveurs cloud ou dédiés, comme OVHCloud, proposent des pare-feu intégrés. Ces pare-feu ajoutent une couche de protection supplémentaire en filtrant le trafic avant qu'il n'atteigne votre serveur.

Suivez les instructions sur : https://help.ovhcloud.com/csm/fr-dedicated-servers-firewall-network?id=kb_article_view&sysparm_article=KB0043455

- **Configuration des règles de pare-feu** : En suivant l'exemple OVHCloud :
- **Garder les ports des services essentiels ouverts** :
  - SSH (22) : Pour la gestion à distance du serveur.
  - HTTP (80) et HTTPS (443) : Pour le trafic web.
  - UDP (53) : Courant pour la communication DNS.
  - Autoriser ICMP : Utile pour les diagnostics (par exemple, ping).
- S'assurer que seuls les ports requis sont ouverts pour minimiser la vulnérabilité.
- Les pare-feu au niveau du fournisseur sont très efficaces car ils bloquent le trafic indésirable au niveau réseau (avant qu'il n'atteigne votre serveur).

Exemple de configuration

Pour s'assurer que seuls les ports SSH (22), HTTP (80), HTTPS (443) et UDP (53) restent ouverts tout en autorisant ICMP, suivez les règles ci-dessous :

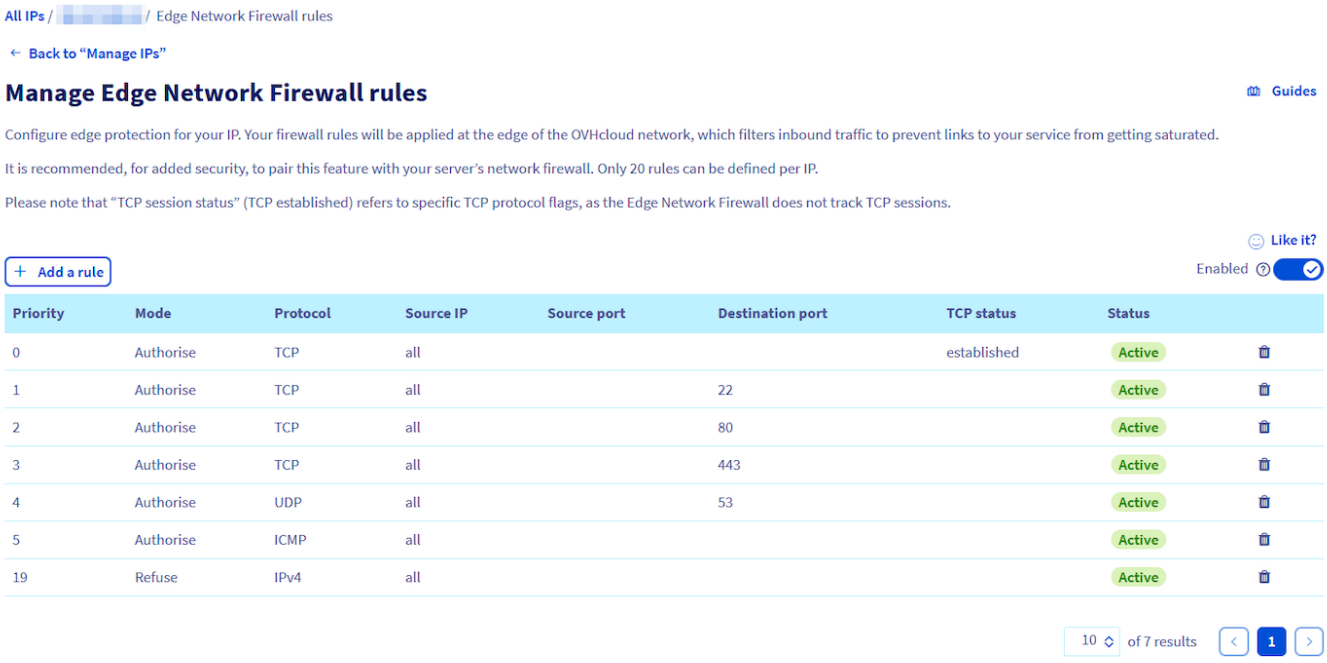


Figure 1: Exemple de configuration du pare-feu

8.6 5. Ajouter Fail2ban

Qu'est-ce que Fail2ban ?

Fail2ban est un logiciel de prévention d'intrusion qui protège votre serveur contre les attaques par force brute. Il fonctionne en surveillant les fichiers journaux à la recherche d'activités suspectes et en bannissant temporairement les adresses IP qui présentent des signes malveillants, comme de multiples tentatives de connexion échouées.

Avantages en matière de sécurité

La mise en place de Fail2ban réduit considérablement le risque d'accès non autorisé à votre serveur en bloquant automatiquement les adresses IP suspectes avant qu'elles ne puissent compromettre votre système.

Installation

Installez Fail2ban avec apt :

```
sudo apt install fail2ban
```

```
sudo systemctl status fail2ban
```

Accédez à `/etc/fail2ban/`

```
cd /etc/fail2ban/
```

```
sudo cp jail.conf jail.local
```

```
# /etc/fail2ban/jail.local
bantime  = 7d
findtime  = 3h
maxretry = 4
```

Recherchez `[sshd]`

```
# /etc/fail2ban/jail.local
[sshd]

mode   = aggressive
port    = ssh
logpath = %(sshd_log)s
backend = %(sshd_backend)s
enabled = true
```

```
sudo systemctl restart fail2ban
```

```
cat /var/log/fail2ban.log
```

Vérifiez que tout est correct, vous devriez voir des lignes ressemblant à ceci :

```
   Creating new jail 'sshd'
2025-03-02 01:52:12,536 fail2ban.jail           [2023]: INFO    Jail 'sshd' uses pyinotify {}
2025-03-02 01:52:12,538 fail2ban.jail           [2023]: INFO    Initiated 'pyinotify' backend
2025-03-02 01:52:12,539 fail2ban.filter         [2023]: INFO      maxLines: 1
2025-03-02 01:52:12,552 fail2ban.filter         [2023]: INFO      maxRetry: 4
2025-03-02 01:52:12,553 fail2ban.filter         [2023]: INFO      findtime: 10800
2025-03-02 01:52:12,553 fail2ban.actions        [2023]: INFO      banTime: 604800
```

```
sudo fail2ban-client status sshd
```

```
Status for the jail: sshd
|- Filter
|  |- Currently failed: 12
|  |- Total failed:     327
|  `- File list:        /var/log/auth.log
`- Actions
|- Currently banned: 10
|- Total banned:     10
`- Banned IP list:   193.168.198.61 92.118.39.72 193.32.162.130 218.92.0.154 2.57.122.188
92.255.85.188 37.44.238.88 94.204.155.90 92.255.85.189 92.255.57.132
```

Étapes pour débannir une adresse IP avec Fail2Ban

1. **Vérifier l'état actuel d'une prison (par exemple, `sshd`) :** Utilisez la commande suivante pour voir quelles adresses IP sont bannies :

```
sudo fail2ban-client status sshd
```

Recherchez la section `Banned IP list` pour trouver l'adresse IP que vous souhaitez débannir. 1. **Débannir l'adresse IP :** Pour débannir une adresse IP spécifique, utilisez la commande `unban` :

```
sudo fail2ban-client unban <IP_ADDRESS>
```

Remplacez `<IP_ADDRESS>` par l'adresse IP réelle que vous souhaitez débannir. Exemple :

```
sudo fail2ban-client unban 193.168.198.61
```

1. **Vérifier que l'adresse IP est débannie :** Vérifiez à nouveau l'état de la prison pour vous assurer que l'adresse IP n'est plus listée :

```
sudo fail2ban-client status sshd
```

2. **Redémarrer Fail2Ban si nécessaire :** Si l'adresse IP reste bannie après le débannissement, redémarrez Fail2Ban :

```
sudo systemctl restart fail2ban
```

Débannir toutes les adresses IP d'une prison spécifique

1. Vérifiez l'état de la prison pour voir les adresses IP bannies :

```
sudo fail2ban-client status <JAIL_NAME>
```

Remplacez `<JAIL_NAME>` par le nom réel de la prison (par exemple, `sshd`). 2. Utilisez la commande `unban` avec `--all` pour débannir toutes les adresses IP de cette prison spécifique :

```
sudo fail2ban-client unban --all
```

8.7 6. Durcissement du serveur

Réglage du fuseau horaire correct

Le réglage du fuseau horaire correct sur votre serveur est important pour la précision des journaux, les tâches planifiées et la cohérence des horodatages dans vos applications avec votre heure locale.

Pourquoi c'est important

Une configuration correcte du fuseau horaire garantit que les journaux et les horodatages sont synchronisés avec votre heure locale, rendant le dépannage et la surveillance plus efficaces.

Vérification et réglage du fuseau horaire

Commencez par vérifier votre configuration actuelle du fuseau horaire :

```
# Afficher le fuseau horaire actuel
sudo timedatectl
```

Pour trouver et définir le fuseau horaire approprié :

```
# Lister tous les fuseaux horaires disponibles
sudo timedatectl list-timezones

# Rechercher un fuseau horaire lié à une ville spécifique
sudo timedatectl list-timezones | grep city

# Exemple : Rechercher le fuseau horaire de Paris
sudo timedatectl list-timezones | grep Paris

# Définir le fuseau horaire à Europe/Paris
sudo timedatectl set-timezone Europe/Paris

# Vérifier le changement de fuseau horaire
sudo timedatectl
```

Exemple de sortie après le réglage du fuseau horaire :

```
               Local time: Sun 2025-03-02 14:29:40 CET
           Universal time: Sun 2025-03-02 13:29:40 UTC
                 RTC time: Sun 2025-03-02 13:29:40
                Time zone: Europe/Paris (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Configuration de l'espace swap

L'espace swap est une portion du disque dur que le système peut utiliser comme mémoire virtuelle lorsque la RAM physique est entièrement utilisée. Un espace swap correctement configuré peut prévenir les plantages système lors d'une utilisation intensive de la mémoire.

Considération de performance

Bien que l'espace swap soit essentiel pour la stabilité du système, un recours excessif au swap peut considérablement ralentir votre serveur. La quantité appropriée de swap dépend de la RAM de votre serveur et de sa charge de travail.

Taille de swap recommandée

Le tableau suivant fournit des recommandations pour la taille du swap en fonction de la RAM de votre serveur :

RAM	Sans hibernation	Avec hibernation	Maximum
256MB	256MB	512MB	512MB
512MB	512MB	1024MB	1024MB
1024MB	1024MB	2048MB	2048MB
2GB	1GB	3GB	4GB
4GB	2GB	6GB	8GB
8GB	3GB	11GB	16GB
16GB	4GB	20GB	32GB
32GB	6GB	38GB	64GB

*Source : **Ubuntu SwapFaq***

Vérification de l'état actuel du swap

Avant de créer un nouveau fichier swap, vérifiez si le swap est déjà activé :

```
# Vérifier l'état actuel du swap
sudo swapon -s

# Si aucune sortie n'apparaît, le swap n'est pas activé

# Vous pouvez également vérifier avec htop, qui affichera :
# Swp[ 0K/0K] lorsque le swap n'est pas activé
```

Suppression du swap existant (si nécessaire)

Si vous devez modifier un fichier swap existant :

```
# Désactiver le fichier swap
sudo swapoff /swapfile

# Sauvegarder fstab avant modification
sudo cp /etc/fstab /etc/fstab.bak

# Modifier fstab pour supprimer l'entrée swap
sudo nano /etc/fstab

# Supprimer le fichier swap
sudo rm /swapfile
```

Création d'un nouveau fichier swap

Choisissez la taille appropriée pour votre serveur en vous basant sur le tableau ci-dessus :

```
# Créer un fichier swap de 2 Gio
sudo dd if=/dev/zero of=/swapfile bs=1024 count=2097152

# Pour un fichier swap de 4 Gio
sudo dd if=/dev/zero of=/swapfile bs=1024 count=4194304

# Pour un fichier swap de 6 Gio
sudo dd if=/dev/zero of=/swapfile bs=1024 count=6291456

# Pour un fichier swap de 8 Gio
sudo dd if=/dev/zero of=/swapfile bs=1024 count=8388608
```

Exemple de sortie pour un fichier swap de 4 Gio :

```
4194304+0 records in
4194304+0 records out
4294967296 bytes (4.3 GB, 4.0 GiB) copied, 8.81315 s, 487 MB/s
```

Comprendre la commande dd :

- `sudo` : Exécute la commande avec les privilèges administratifs
- `dd` : Une commande bas niveau pour copier et convertir des données
- `if=/dev/zero` : Utilise `/dev/zero` comme entrée, qui fournit des octets nuls illimités
- `of=/swapfile` : Définit le fichier de sortie comme `/swapfile`
- `bs=1024` : Définit la taille de bloc à 1024 octets (1 Ko)
- `count=4194304` : Spécifie le nombre de blocs à écrire (4194304 blocs x 1024 octets = 4 Gio)

Configuration du fichier swap

Après avoir créé le fichier swap, configurez-le pour l'utilisation :

```
# Naviguer vers le répertoire racine
cd /

# Définir les permissions correctes pour le fichier swap
# (empêche les autres utilisateurs de le lire)
sudo chmod 600 /swapfile

# Configurer le fichier comme zone de swap Linux
sudo mkswap /swapfile

# Activer le nouveau fichier swap
sudo swapon /swapfile

# Vérifier l'état du swap
sudo swapon -s

# Rendre le swap permanent après redémarrage en modifiant fstab
sudo vim /etc/fstab

# Ajouter cette ligne au fichier fstab :
/swapfile swap swap defaults 0 0
```

Optimisation des paramètres swap

Pour améliorer les performances du système, vous pouvez ajuster l'utilisation du swap par le système :

```
# Vérifier les paramètres actuels de swappiness et cache pressure
sudo sysctl -a | grep -e vm.swappiness -e vm.vfs_cache_pressure

# Naviguer vers le répertoire sysctl.d
cd /etc/sysctl.d/

# Créer ou modifier custom_overrides.conf
sudo vim custom_overrides.conf
```

Ajoutez la configuration suivante :

```
# Personnalisation SWAP
vm.swappiness = 1
vm.vfs_cache_pressure = 50
```

Ces paramètres :

- `vm.swappiness = 1` : Minimise l'utilisation du swap en privilégiant la RAM, n'utilisant le swap qu'en dernier recours
- `vm.vfs_cache_pressure = 50` : Équilibre l'utilisation de la mémoire en conservant plus longtemps les métadonnées des fichiers en cache, améliorant la vitesse d'accès aux fichiers

Appliquez les modifications :

```
# Redémarrer pour appliquer les modifications
sudo reboot

# Après le redémarrage, vérifier les paramètres
sudo sysctl -a | grep -e vm.swappiness -e vm.vfs_cache_pressure
```

Sortie attendue :

```
vm.swappiness = 1
vm.vfs_cache_pressure = 50
```

Durcissement de la mémoire partagée

La mémoire partagée (`/dev/shm`) est un système de fichiers temporaire utilisé pour la communication inter-processus. Par défaut, elle peut présenter des vulnérabilités de sécurité exploitables par des attaquants.

Avantage de sécurité

Le durcissement de la mémoire partagée empêche les attaquants d'exécuter du code malveillant ou d'élever leurs privilèges via l'espace de mémoire partagée, qui est un vecteur d'attaque courant.

Vérification des paramètres actuels de la mémoire partagée

Commencez par vérifier la configuration actuelle de votre mémoire partagée :

```
# Confirmer les paramètres actuels
mount | grep shm

# Exemple de sortie :
# tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
```

Durcissement de la configuration de la mémoire partagée

Pour sécuriser la mémoire partagée, vous devez modifier le fichier `/etc/fstab` :

```
# Sauvegarder le fichier fstab avant modification
cd /etc
sudo cp fstab fstab.bak

# Modifier le fichier fstab
sudo vim /etc/fstab
```

Ajoutez la ligne suivante à la fin du fichier :

```
# DURCISSEMENT DE LA MEMOIRE PARTAGEE
none /dev/shm tmpfs defaults,noexec,nosuid,nodev 0 0
```

Cette configuration :

- `noexec` : Empêche l'exécution de binaires dans la zone de mémoire partagée
- `nosuid` : Bloque l'élévation de privilèges via les bits `setuid`/`setgid`
- `nodev` : Interdit la création de fichiers de périphérique dans la mémoire partagée

Vérification de la configuration

Après avoir enregistré les modifications, vous pouvez soit redémarrer, soit remonter la mémoire partagée pour appliquer les changements :

```
# Remonter /dev/shm avec les nouveaux paramètres
sudo mount -o remount /dev/shm

# Vérifier les nouveaux paramètres
mount | grep /dev/shm
```

Sortie attendue :

```
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,inode64)
```

La présence de `noexec`, `nosuid` et `nodev` dans la sortie confirme que le durcissement a été appliqué avec succès.

Désactivation d'IPv6

IPv6 est le protocole Internet de nouvelle génération, mais si vous ne l'utilisez pas, désactiver IPv6 peut réduire la surface d'attaque de votre serveur et simplifier votre configuration réseau.

Considération de sécurité

Si votre serveur ne nécessite pas de connectivité IPv6, sa désactivation élimine les vulnérabilités potentielles liées à IPv6 et réduit la complexité des règles de pare-feu.

Méthode 1 : Désactivation d'IPv6 via GRUB

Pour les systèmes utilisant le chargeur de démarrage GRUB, vous pouvez désactiver IPv6 au niveau du noyau :

```
# Ouvrir le fichier de configuration GRUB
sudo nano /etc/default/grub
```

Modifiez la ligne `GRUB_CMDLINE_LINUX` pour inclure le paramètre de désactivation d'IPv6 :

```
GRUB_CMDLINE_LINUX="ipv6.disable=1"
```

Si la ligne contient déjà d'autres paramètres, ajoutez le paramètre IPv6 avec un espace comme séparateur :

```
GRUB_CMDLINE_LINUX="existing_parameters ipv6.disable=1"
```

Mettez à jour GRUB et redémarrez pour appliquer les modifications :

```
# Mettre à jour la configuration GRUB
sudo update-grub

# Redémarrer le système
sudo reboot
```

Méthode 2 : Désactivation d'IPv6 avec Sysctl

Vous pouvez également désactiver IPv6 en utilisant la configuration sysctl :

```
# Modifier le fichier de configuration sysctl
sudo vim /etc/sysctl.d/99-sysctl.conf
```

Ajoutez les lignes suivantes pour désactiver IPv6 :

```
# Désactiver IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

Appliquez les modifications sans redémarrage :

```
# Recharger la configuration sysctl
sudo sysctl -p
```

Vérification de la désactivation d'IPv6

Pour confirmer qu'IPv6 a été désactivé avec succès :

```
# Vérifier la présence d'adresses IPv6
ip a | grep inet6
```

Si aucune sortie n'est affichée, IPv6 est désactivé avec succès. Si vous voyez encore des adresses IPv6, vous devrez peut-être redémarrer le système ou vérifier votre configuration.

Durcissement et optimisation de la couche réseau

L'optimisation de votre configuration réseau peut améliorer à la fois la sécurité et les performances. Les paramètres suivants aident à se protéger contre les attaques réseau courantes et à optimiser les performances réseau.

Paramètres de sécurité et de performance réseau

Naviguez vers le répertoire de configuration sysctl :

```
# Accéder au répertoire de configuration sysctl
cd /etc/sysctl.d/

# Créer ou modifier le fichier de surcharges personnalisées
sudo vim custom_overrides.conf
```

Ajoutez les directives de durcissement réseau suivantes :

```
# Protection contre l'usurpation d'IP
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
```

```
# Protection contre les inondations SYN
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 2048
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 5

# Routage des paquets source
net.ipv4.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0

# Augmenter le nombre de ports utilisables
net.ipv4.ip_local_port_range = 1024 65535

# Augmenter les descripteurs de fichiers et restreindre les core dumps
fs.file-max = 2097152
fs.suid_dumpable = 0

# Nombre de connexions entrantes / file d'attente
net.core.somaxconn = 65535
net.core.netdev_max_backlog = 262144

# Augmenter les tampons mémoire maximum
net.core.optmem_max = 25165824

# Tailles des tampons d'envoi/réception
net.core.rmem_default = 31457280
net.core.rmem_max = 67108864
net.core.wmem_default = 31457280
net.core.wmem_max = 67108864
```

Comprendre les paramètres de durcissement réseau

Ces paramètres améliorent la sécurité et les performances de votre serveur :

- **Protection contre l'usurpation d'IP** :
  - `rp_filter = 1` active la validation source des paquets, empêchant les attaques d'usurpation d'IP où les attaquants falsifient l'adresse source.
- **Protection contre les inondations SYN** :
  - `tcp_syncookies = 1` protège contre les attaques par inondation SYN en validant les demandes de connexion sans utiliser d'entrées dans la file de connexion.
  - `tcp_max_syn_backlog = 2048` augmente la file d'attente SYN pour gérer plus de demandes de connexion.
  - `tcp_synack_retries = 2` et `tcp_syn_retries = 5` optimisent l'établissement des connexions TCP.
- **Routage des paquets source** :
  - La désactivation de `accept_source_route` empêche les attaquants de spécifier la route que les paquets empruntent à travers le réseau.
- **Ports utilisables** :
  - `ip_local_port_range = 1024 65535` augmente la plage de ports locaux disponibles pour les connexions sortantes.
- **Descripteurs de fichiers et core dumps** :

- `fs.file-max = 2097152` augmente le nombre maximum de descripteurs de fichiers.
- `fs.suid_dumpable = 0` empêche les programmes setuid de créer des core dumps, qui pourraient exposer des informations sensibles.

- **Gestion des connexions** :
  - `somaxconn = 65535` augmente le nombre maximum de connexions en attente.
  - `netdev_max_backlog = 262144` augmente la longueur maximale de la file d'entrée du processeur.
- **Tampons mémoire et tampons de socket** :
  - Les paramètres `optmem_max`, `rmem_*` et `wmem_*` optimisent l'allocation mémoire pour les opérations réseau.

Application et vérification des paramètres

Appliquez les modifications en redémarrant le système :

```
# Redémarrer pour appliquer toutes les modifications
sudo reboot
```

Après le redémarrage, vérifiez que les paramètres ont été appliqués :

```
# Vérifier un paramètre spécifique (exemple)
sudo sysctl -a | grep net.core.optmem_max

# Sortie attendue :
# net.core.optmem_max = 25165824
```

Vous pouvez vérifier les autres paramètres de la même manière en remplaçant `net.core.optmem_max` par le paramètre spécifique que vous souhaitez vérifier.

Optimisation du contrôle de congestion TCP

Les algorithmes de contrôle de congestion TCP gèrent le trafic réseau pour éviter l'effondrement par congestion. L'algorithme BBR (Bottleneck Bandwidth and RTT) de Google peut améliorer significativement les performances réseau par rapport aux algorithmes traditionnels.

Avantage de performance

BBR peut améliorer le débit et réduire la latence, en particulier sur les réseaux avec perte de paquets. Cela peut se traduire par des temps de chargement de pages web plus rapides et de meilleures performances réseau globales.

Vérification des paramètres actuels de contrôle de congestion

Commencez par vérifier quels algorithmes de contrôle de congestion sont disponibles et lequel est actuellement actif :

```
# Vérifier les algorithmes de contrôle de congestion disponibles
sudo sysctl net.ipv4.tcp_available_congestion_control

# Vérifier l'algorithme actuellement actif
sudo sysctl net.ipv4.tcp_congestion_control
```

Activation de l'algorithme BBR

Pour activer l'algorithme de contrôle de congestion BBR :

```
# Charger le module BBR
sudo modprobe tcp_bbr

# S'assurer que BBR se charge au démarrage
sudo bash -c 'echo "tcp_bbr" > /etc/modules-load.d/bbr.conf'

# Vérifier que BBR est maintenant disponible
sudo sysctl net.ipv4.tcp_available_congestion_control
```

Définir BBR comme algorithme par défaut

Modifiez le fichier de configuration des surcharges personnalisées :

```
# Modifier le fichier de configuration sysctl
sudo vim /etc/sysctl.d/custom_overrides.conf
```

Ajoutez les lignes suivantes pour définir BBR comme algorithme de contrôle de congestion par défaut :

```
# Activer l'algorithme BBR
net.ipv4.tcp_congestion_control = bbr
net.core.default_qdisc = fq
```

L'ordonnanceur de paquets `fq` (Fair Queuing) fonctionne bien avec BBR pour réduire le bufferbloat et améliorer la latence.

Vérification de la configuration

Après avoir enregistré les modifications, vous pouvez les appliquer immédiatement et vérifier :

```
# Appliquer les modifications
sudo sysctl -p /etc/sysctl.d/custom_overrides.conf

# Confirmer que BBR est maintenant l'algorithme actif
sudo sysctl net.ipv4.tcp_congestion_control
```

Sortie attendue :

```
net.ipv4.tcp_congestion_control = bbr
```

Optimisation des temps d'accès aux fichiers

Par défaut, Linux met à jour le temps d'accès (atime) d'un fichier à chaque lecture, ce qui peut entraîner des opérations d'E/S disque inutiles. La désactivation de cette fonctionnalité peut améliorer les performances, en particulier pour les serveurs très sollicités.

Amélioration des performances

La désactivation des mises à jour du temps d'accès peut réduire les E/S disque de 10 à 15 % sur les serveurs très sollicités, ce qui améliore les performances globales du système et prolonge la durée de vie des SSD.

Vérification des options de montage actuelles

Commencez par vérifier les options de montage actuelles de votre système de fichiers :

```
# Afficher les systèmes de fichiers montés avec des tailles lisibles
df -h
```

```
# Afficher les options de montage détaillées
cat /proc/mounts
```

Exemple de sortie de `/proc/mounts` :

```
/dev/vda1 / ext4 rw,relatime,errors=remount-ro,data=ordered 0 0
```

ou

```
/dev/sda1 / ext4 rw,relatime,discard,errors=remount-ro,commit=30 0 0
```

Notez que la plupart des systèmes utilisent `relatime` par défaut, qui est un compromis ne mettant à jour les temps d'accès que s'ils sont antérieurs au temps de modification. Bien que meilleur que le `atime` par défaut, l'utilisation de `noatime` offre des performances encore meilleures.

Modification de la table du système de fichiers (fstab)

Pour désactiver définitivement les mises à jour du temps d'accès, modifiez le fichier `/etc/fstab` :

```
# Modifier la table du système de fichiers
sudo vim /etc/fstab
```

Trouvez la ligne de votre système de fichiers racine (/) et ajoutez l'option `noatime` :

Avant (exemple) :

```
UUID=8ce122bd-f39c-45ae-b129-9650cfc67567 / ext4 errors=remount-ro 0 1
```

Après (exemple) :

```
UUID=8ce122bd-f39c-45ae-b129-9650cfc67567 / ext4 errors=remount-ro,noatime 0 1
```

Si votre fstab utilise un format différent, assurez-vous d'ajouter `noatime` aux options de montage :

```
/dev/sda1 / ext4 rw,noatime,discard,errors=remount-ro,commit=30 0 0
```

Application et vérification des modifications

Vous pouvez soit redémarrer votre système, soit remonter le système de fichiers pour appliquer les modifications :

```
# Remonter le système de fichiers racine avec les nouvelles options
sudo mount -o remount /

# Vérifier les modifications
cat /proc/mounts
```

Sortie attendue (notez que l'option `noatime` est maintenant présente) :

```
/dev/vda1 / ext4 rw,noatime,errors=remount-ro,data=ordered 0 0
```

Cela confirme que le système de fichiers est maintenant monté avec l'option `noatime`, ce qui améliorera les performances d'E/S disque.

Augmentation des limites de fichiers ouverts

Les systèmes Linux ont des limites sur le nombre de fichiers qu'un processus peut ouvrir simultanément. Pour les serveurs très sollicités, en particulier ceux qui exécutent des bases de données ou des serveurs web, les limites par défaut peuvent être trop restrictives.

Optimisation du serveur

L'augmentation des limites de fichiers empêche les erreurs "Too many open files" qui peuvent provoquer des défaillances d'application, en particulier sous forte charge ou avec des applications qui maintiennent de nombreuses connexions simultanées.

Vérification des limites de fichiers actuelles

Commencez par vérifier vos limites actuelles de fichiers ouverts :

```
# Vérifier la limite stricte (maximum autorisé)
ulimit -Hn

# Vérifier la limite souple (paramètre actuel)
ulimit -Sn
```

Les valeurs par défaut sont généralement de 1024 pour les limites souples et 4096 pour les limites strictes, ce qui peut être insuffisant pour les serveurs très sollicités.

Augmentation des limites de fichiers à l'échelle du système

Pour augmenter les limites pour tous les utilisateurs, créez un fichier de configuration dans le répertoire limits.d :

```
# Naviguer vers le répertoire limits.d
cd /etc/security/limits.d/

# Créer ou modifier le fichier de surcharges personnalisées
sudo vim custom_overrides.conf
```

Ajoutez la configuration suivante pour augmenter les limites :

```
# <domain> <type>  <item>      <value>
*          soft    nofile      120000
*          hard    nofile      120000
root       soft    nofile      120000
root       hard    nofile      120000
```

Cette configuration :

- Définit les limites souples et strictes à 120 000 fichiers ouverts
- S'applique à tous les utilisateurs (`*`) et spécifiquement à l'utilisateur root
- La valeur de 120 000 est suffisante pour la plupart des charges de travail serveur, mais peut être augmentée si nécessaire

Ajustement des limites

Si vous rencontrez des erreurs "Too many open files" dans vos journaux, vous devrez peut-être augmenter davantage ces valeurs en fonction de votre charge de travail spécifique.

Activation des limites PAM

Pour que les limites prennent effet, vous devez vous assurer que le système PAM (Pluggable Authentication Modules) est configuré pour les appliquer. Modifiez les fichiers de configuration de session PAM :

```
# Ajouter pam_limits.so à common-session
sudo bash -c 'echo "session required pam_limits.so" >> /etc/pam.d/common-session'
```

```
# Ajouter pam_limits.so à common-session-noninteractive
sudo bash -c 'echo "session required pam_limits.so" >>
/etc/pam.d/common-session-noninteractive'
```

Vous pouvez également modifier ces fichiers directement :

```
# Modifier common-session
sudo vim /etc/pam.d/common-session

# Modifier common-session-noninteractive
sudo vim /etc/pam.d/common-session-noninteractive
```

Ajoutez la ligne suivante à chaque fichier si elle n'est pas déjà présente :

```
session required pam_limits.so
```

Vérification des modifications

Après avoir effectué ces modifications, vous devrez vous déconnecter et vous reconnecter pour qu'elles prennent effet. Vérifiez ensuite les nouvelles limites :

```
# Vérifier la nouvelle limite stricte
ulimit -Hn

# Vérifier la nouvelle limite souple
ulimit -Sn
```

La sortie devrait afficher les nouvelles limites (120000) que vous avez configurées.

8.8 7. Associer un nom de domaine à votre serveur

Comprendre le système de noms de domaine (DNS)

Le système de noms de domaine (DNS) traduit les noms de domaine lisibles par l'homme (comme example.com) en adresses IP (comme 192.0.2.1) que les ordinateurs utilisent pour s'identifier sur le réseau.

Pourquoi le DNS est important

Des enregistrements DNS correctement configurés garantissent que les visiteurs peuvent accéder à votre site web en utilisant votre nom de domaine au lieu de devoir mémoriser l'adresse IP de votre serveur.

Enregistrements DNS requis

Pour associer votre nom de domaine à votre serveur, vous devrez créer au moins deux enregistrements DNS :

Enregistrement A (Address Record)

L'enregistrement A associe directement votre nom de domaine à l'adresse IPv4 de votre serveur.

Type d'enregistrement	Hôte/Nom	Valeur/Contenu
A	@ ou domain.com	L'adresse IP de votre serveur (par exemple, 192.0.2.1)

Enregistrement CNAME (Canonical Name)

L'enregistrement CNAME crée un alias qui pointe le sous-domaine "www" vers votre domaine principal.

Type d'enregistrement	Hôte/Nom	Valeur/Contenu
CNAME	www	Votre domaine (par exemple, domain.com)

Configuration des enregistrements DNS chez votre registraire de domaine

Les étapes exactes pour configurer les enregistrements DNS varient selon votre registraire de domaine ou fournisseur DNS. Voici des instructions générales pour quelques fournisseurs populaires :

Étapes générales

1. Connectez-vous au compte de votre registraire de domaine
2. Naviguez vers la section de gestion DNS ou paramètres DNS
3. Recherchez une option pour ajouter ou modifier les enregistrements DNS
4. Ajoutez l'enregistrement A et l'enregistrement CNAME comme décrit ci-dessus
5. Enregistrez vos modifications

Délai de propagation DNS courant

Après la mise à jour de vos enregistrements DNS, les modifications peuvent prendre de quelques minutes à 48 heures pour se propager sur l'ensemble d'Internet. C'est normal et dépend de divers facteurs, notamment :

- Les serveurs DNS de votre registraire de domaine
- Les politiques de mise en cache de votre FAI
- Les paramètres de durée de vie (TTL) de vos enregistrements DNS

Vérification de votre configuration DNS

Vous pouvez vérifier que vos enregistrements DNS sont correctement configurés en utilisant divers outils :

Utilisation de la ligne de commande

```
# Vérifier l'enregistrement A
dig +short domain.com A

# Vérifier l'enregistrement CNAME
dig +short www.domain.com CNAME
```

Utilisation d'outils en ligne

Vous pouvez également utiliser des outils de recherche DNS en ligne tels que :

- **MxToolbox**
- **DNSChecker**
- **WhatsMyDNS**

Ces outils vous permettent de vérifier si vos enregistrements DNS se sont correctement propagés à travers les différents serveurs DNS dans le monde.

Dépannage des problèmes DNS

Si votre domaine ne pointe pas correctement vers votre serveur, essayez ces étapes de dépannage :

1. **Vérifier la syntaxe des enregistrements** : Assurez-vous qu'il n'y a pas de fautes de frappe dans votre nom de domaine ou votre adresse IP
2. **Vérifier la propagation** : Utilisez plusieurs outils de recherche DNS pour vérifier si les modifications se sont propagées
3. **Attendre plus longtemps** : Les modifications DNS peuvent parfois prendre jusqu'à 48 heures pour se propager complètement
4. **Vider votre cache DNS** : Sur votre machine locale, videz votre cache DNS pour voir les derniers enregistrements
5. **Contacter le support** : Si les problèmes persistent, contactez l'équipe de support de votre registraire de domaine

Exemple de configuration

Pour un domaine appelé "example.com" pointant vers un serveur avec l'adresse IP 192.0.2.1 :
Enregistrement A :

- Type : A
- Hôte/Nom : example.com (ou @)
- Valeur/Contenu : 192.0.2.1

Enregistrement CNAME :

- Type : CNAME
- Hôte/Nom : www
- Valeur/Contenu : example.com

Enregistrements DNS supplémentaires à considérer

Selon vos besoins, vous pourriez vouloir configurer des enregistrements DNS supplémentaires :

Enregistrements MX

Les enregistrements MX (Mail Exchange) dirigent les e-mails vers votre serveur de messagerie. Ils sont essentiels si vous prévoyez d'utiliser la messagerie avec votre domaine.

Enregistrements TXT

Les enregistrements TXT (Text) stockent des informations textuelles dans le DNS. Ils sont couramment utilisés pour la vérification de domaine et les protocoles de sécurité de messagerie comme SPF, DKIM et DMARC.

Enregistrements AAAA

Si votre serveur possède une adresse IPv6, vous devriez créer un enregistrement AAAA pour associer votre domaine à cette adresse, de manière similaire au fonctionnement d'un enregistrement A pour IPv4.

8.9 8. Installation de la pile LEMP (Linux, Nginx, MariaDB, PHP)

Introduction à la pile LEMP

La pile LEMP est un ensemble de logiciels qui fonctionnent ensemble pour héberger des sites web dynamiques et des applications web. LEMP signifie :

- **L**inux (Système d'exploitation)
- **E**ngine-x (Serveur web Nginx, prononcé "Engine-X")
- **M**ariaDB (Serveur de base de données)
- **P**HP (Langage de programmation)

Pourquoi LEMP ?

La pile LEMP est légère, rapide et sécurisée, ce qui en fait un excellent choix pour héberger des sites WordPress. Nginx utilise moins de mémoire qu'Apache et gère les connexions simultanées de manière plus efficace.

Installation et configuration de Nginx

Nginx est un serveur web haute performance et proxy inverse qui servira le contenu de votre site web aux visiteurs.

Ajout du dépôt Nginx

Commencez par ajouter le dépôt pour la dernière version de Nginx :

```
# Ajouter le dépôt Nginx
sudo add-apt-repository ppa:ondrej/nginx

# Mettre à jour les listes de paquets
sudo apt update
```

Installation de Nginx et des modules utiles

Installez Nginx avec quelques modules utiles :

```
# Installer Nginx avec des modules supplémentaires
sudo apt install nginx libnginx-mod-http-cache-purge libnginx-mod-http-headers-more-filter
libnginx-mod-http-brotli-filter libnginx-mod-http-brotli-static
```

Ces modules fournissent :

- **cache-purge** : Permet de purger le contenu du cache de Nginx
- **headers-more-filter** : Offre plus de contrôle sur les en-têtes HTTP
- **brotli-filter/static** : Active la compression Brotli pour de meilleures performances que gzip

Dépannage des problèmes IPv6

Si Nginx ne parvient pas à démarrer avec une erreur liée à IPv6 :

```
# Vérifier l'état de Nginx
sudo systemctl status nginx
```

Vous pourriez voir une erreur comme :

```
nginx: [emerg] socket() [::]:80 failed (97: Address family not supported by protocol)
```

Cela se produit lorsqu'IPv6 est désactivé mais que Nginx est configuré pour écouter sur des adresses IPv6. Pour corriger cela :

```
# Modifier la configuration du site par défaut
sudo vim /etc/nginx/sites-available/default
```

Trouvez et commentez la directive d'écoute IPv6 :

```
# Modifier cette ligne :
listen [::]:80 default_server;

# En ceci :
#listen [::]:80 default_server;
```

Démarrage et vérification de Nginx

Après avoir effectué les modifications, testez la configuration et démarrez Nginx :

```
# Tester la configuration de Nginx
sudo nginx -t

# Démarrer Nginx
sudo systemctl start nginx

# Vérifier l'état de Nginx
sudo systemctl status nginx

# S'assurer que Nginx démarre au boot
sudo systemctl is-enabled nginx
```

Test du site web par défaut

Vérifiez que Nginx sert correctement le contenu :

```
# Tester avec curl (remplacez par l'adresse IP de votre serveur)
curl -i http://your_server_ip

# Afficher le contenu de la page par défaut
cat /var/www/html/index.nginx-debian.html
```

Vérification de la version de Nginx

Vérifiez la version de Nginx installée :

```
# Vérifier la version de Nginx
nginx -v
```

Installation et configuration de MariaDB

MariaDB est un serveur de base de données robuste et open source qui stockera vos données WordPress.

Installation de MariaDB

```
# Installer le serveur MariaDB
sudo apt install mariadb-server
```

Vérification de l'installation de MariaDB

```
# Vérifier l'état de MariaDB
sudo systemctl status mariadb

# S'assurer que MariaDB démarre au boot
sudo systemctl is-enabled mariadb
```

Sécurisation de MariaDB

Exécutez le script de sécurité pour définir un mot de passe root et supprimer les paramètres par défaut non sécurisés :

```
# Exécuter le script d'installation sécurisée de MariaDB
sudo mysql_secure_installation
```

Suivez les invites pour : 1. Définir un mot de passe root 2. Supprimer les utilisateurs anonymes 3. Interdire la connexion root à distance 4. Supprimer la base de données de test 5. Recharger les tables de privilèges

Installation et configuration de PHP

PHP est le langage de programmation sur lequel WordPress est construit. Nous allons installer PHP-FPM (FastCGI Process Manager) pour de meilleures performances avec Nginx.

Ajout du dépôt PHP

```
# Ajouter le dépôt PHP
sudo add-apt-repository ppa:ondrej/php

# Mettre à jour les listes de paquets
sudo apt update
```

Installation de PHP et des extensions requises

Installez PHP 8.3 avec les extensions nécessaires pour WordPress :

```
# Installer PHP 8.3 avec les extensions
sudo apt install php8.3-fpm php8.3-gd php8.3-mbstring php8.3-mysql php8.3-xml php8.3-xmlrpc
php8.3-opcache php8.3-cli php8.3-zip php8.3-soap php8.3-intl php8.3-bcmath php8.3-curl
php8.3-imagick php8.3-ssh2
```

Ces extensions fournissent :

- **fpm** : FastCGI Process Manager pour l'intégration avec Nginx
- **gd** : Bibliothèque de traitement d'images
- **mbstring** : Gestion des chaînes multi-octets
- **mysql** : Connectivité avec la base de données MariaDB/MySQL
- **xml/xmlrpc** : Capacités de traitement XML
- **opcache** : Améliore les performances de PHP en mettant en cache le bytecode précompilé des scripts
- Et d'autres extensions essentielles pour le fonctionnement de WordPress

Vérification de l'installation de PHP

```
# Vérifier l'état de PHP-FPM
sudo systemctl status php8.3-fpm

# S'assurer que PHP-FPM démarre au boot
sudo systemctl is-enabled php8.3-fpm

# Vérifier la version de PHP
php -v
```

Configuration de Nginx pour fonctionner avec PHP

Pour faire fonctionner Nginx avec PHP, vous devez le configurer pour transmettre les requêtes PHP à PHP-FPM :

```
# Modifier la configuration du site par défaut de Nginx
sudo vim /etc/nginx/sites-available/default
```

Trouvez la section pour le traitement PHP (généralement commentée) et mettez-la à jour :

```
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php8.3-fpm.sock;
}
```

Testez et rechargez Nginx :

```
# Tester la configuration
sudo nginx -t

# Recharger Nginx
sudo systemctl reload nginx
```

Test du traitement PHP

Créez un fichier PHP de test pour vérifier que PHP fonctionne avec Nginx :

```
# Créer un fichier d'information PHP
echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php
```

Visitez `http://your_server_ip/info.php` dans votre navigateur. Vous devriez voir la page d'information PHP.

```
sudo rm /var/www/html/info.php
```

Note de sécurité

Après avoir confirmé que PHP fonctionne correctement, supprimez le fichier info.php car il révèle des informations sensibles sur votre serveur :

Votre pile LEMP est maintenant installée et configurée, fournissant une base solide pour l'installation de WordPress dans les sections suivantes.

8.10 9. Configuration de la messagerie

Introduction à la configuration de la messagerie serveur

La plupart des applications web, y compris WordPress, ont besoin d'envoyer des e-mails pour diverses raisons telles que les réinitialisations de mot de passe, les notifications et les soumissions de formulaires de contact. Dans cette section, nous allons configurer MSMTP, un client SMTP léger qui permet à votre serveur d'envoyer des e-mails via un serveur SMTP externe.

Pourquoi MSMTP ?

MSMTP est une solution simple et fiable pour les e-mails sortants qui ne nécessite pas l'exécution d'un serveur de messagerie complet sur votre système. Il agit comme un relais entre vos applications et un service SMTP externe comme Gmail, SendGrid ou le serveur de messagerie de votre hébergeur.

Installation de MSMTP

Commençons par installer les paquets nécessaires :

```
# Mettre à jour les listes de paquets
sudo apt update

# Installer MSMTP et le wrapper MTA (Mail Transfer Agent)
sudo apt install msmtp msmtp-mta
```

Le paquet `msmtp-mta` permet aux programmes système d'utiliser MSMTP comme expéditeur de courrier par défaut.

Configuration de MSMTP pour votre utilisateur

Création du fichier de configuration

Créez un fichier de configuration dans votre répertoire personnel :

```
# Naviguer vers votre répertoire personnel
cd

# Créer et modifier le fichier de configuration
nano .msmtprc
```

Modèle de configuration de base

Ajoutez la configuration suivante à votre fichier `.msmtprc`, en remplaçant les valeurs de substitution par les détails réels de votre service de messagerie :

```
# Paramètres globaux par défaut
defaults
# Paramètres TLS
tls on
tls_starttls on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
# Journalisation
logfile ~/.msmtp.log

# Configuration du compte
```

```
account mymail
# Paramètres du serveur SMTP
host smtp.example.com
port 587
auth on
user your_email@example.com
password your_app_password
from your_email@example.com

# Définir le compte par défaut
account default : mymail
```

```
host smtp.gmail.com
port 587
```

Exemple de configuration Gmail

Si vous utilisez Gmail, votre configuration ressemblera à :
Pour Gmail, vous devrez utiliser un “Mot de passe d’application” au lieu de votre mot de passe habituel. Vous pouvez en générer un dans les paramètres de votre compte Google sous Sécurité > Mots de passe d’applications (nécessite l’activation de la vérification en deux étapes).

Sécurisation du fichier de configuration

Comme le fichier de configuration contient votre mot de passe de messagerie, il est crucial de définir les permissions appropriées :

```
# Définir des permissions restrictives sur le fichier de configuration
sudo chmod 600 ~/.msmtprc
```

Création du fichier journal

Créez un fichier journal pour suivre l’activité d’envoi d’e-mails :

```
# Créer le fichier journal
touch ~/.msmtp.log

# Définir la propriété et les permissions appropriées
sudo chown $USER:$USER ~/.msmtp.log
sudo chmod 660 ~/.msmtp.log
```

Test d’envoi d’e-mail en ligne de commande

Vérifions que MSMTP fonctionne correctement en envoyant un e-mail de test :

```
# Envoyer un e-mail de test (remplacez par l'adresse e-mail du destinataire)
msmtp recipient@example.com
```

Après avoir exécuté cette commande : 1. Tapez une ligne d’objet et appuyez sur Entrée 2. Tapez le corps de votre message 3. Appuyez sur Ctrl+D pour envoyer l’e-mail

Si tout est correctement configuré, l’e-mail devrait être envoyé sans message d’erreur.

Configuration de MSMTP pour PHP

Pour permettre aux applications PHP (comme WordPress) d'envoyer des e-mails, nous devons configurer MSMTP à l'échelle du système.

Création d'une configuration à l'échelle du système

```
# Copier votre configuration utilisateur vers l'emplacement système
sudo cp ~/.msmtprc /etc/msmtprc

# Définir la propriété et les permissions appropriées
sudo chown www-data:www-data /etc/msmtprc
sudo chmod 660 /etc/msmtprc
```

Modification de la configuration système

Modifiez le fichier de configuration à l'échelle du système pour utiliser un emplacement de journal différent :

```
# Modifier la configuration système
sudo nano /etc/msmtprc
```

Changez le chemin du fichier journal par :

```
logfile /var/log/msmtp.log
```

Création du fichier journal système

```
# Naviguer vers le répertoire des journaux
cd /var/log/

# Créer le fichier journal
sudo touch msmtp.log

# Définir la propriété et les permissions appropriées
sudo chown www-data:adm msmtp.log
sudo chmod 640 msmtp.log
```

Test de la fonctionnalité e-mail PHP

Créez un script PHP pour tester l'envoi d'e-mails :

```
# Naviguer vers votre répertoire personnel
cd

# Créer un fichier PHP de test
nano php_mail_test.php
```

Ajoutez le contenu suivant au fichier, en remplaçant les adresses e-mail par vos adresses réelles :

```php
<?php
    // Activer le rapport d'erreurs pour le débogage
    ini_set('display_errors', 1);
    error_reporting(E_ALL);

    // Détails de l'e-mail
    $from = "your_email@example.com";  // Doit correspondre au 'from' dans la configuration
```

```
    msmtp
    $to = "recipient@example.com";
    $subject = "PHP Mail Test";
    $message = "This is a test email sent from PHP using MSMTP.";
    $headers = "From: " . $from;

    // Envoyer l'e-mail
    if(mail($to, $subject, $message, $headers)) {
        echo "Test email sent successfully!";
    } else {
        echo "Email sending failed.";
    }
?>
```

Exécution du test en tant qu'utilisateur du serveur web

Pour tester correctement la fonctionnalité e-mail telle qu'elle serait utilisée par vos applications web :

```
# Exécuter le script PHP en tant qu'utilisateur www-data
sudo -u www-data php php_mail_test.php
```

En cas de succès, vous devriez voir "Test email sent successfully!" et l'e-mail devrait arriver dans la boîte de réception du destinataire.

Dépannage

Si vous rencontrez des problèmes avec l'envoi d'e-mails :

1. **Vérifier les fichiers journaux** :

   ```
   cat ~/.msmtp.log    # Pour les problèmes au niveau utilisateur
   sudo cat /var/log/msmtp.log    # Pour les problèmes système/PHP
   ```

2. **Vérifier les paramètres SMTP** :
   - Confirmez l'adresse du serveur SMTP, le port et les identifiants
   - Pour Gmail, assurez-vous d'utiliser un mot de passe d'application si la 2FA est activée
3. **Tester la connectivité** :

   ```
   telnet smtp.example.com 587
   ```

   Si vous pouvez vous connecter, vous devriez voir une réponse du serveur
4. **Vérifier la configuration mail de PHP** :

   ```
   php -i | grep mail
   ```

   Assurez-vous que PHP est configuré pour utiliser le sendmail du système (qui est maintenant MSMTP)

Considérations de sécurité

- N'utilisez jamais votre mot de passe de messagerie principal ; utilisez des mots de passe spécifiques aux applications lorsque c'est possible
- Faites une rotation régulière de vos identifiants de messagerie
- Envisagez d'utiliser des variables d'environnement ou un coffre-fort sécurisé pour stocker les identifiants en environnement de production

- Supprimez ou sécurisez les fichiers de test après avoir confirmé le fonctionnement :

```
rm ~/php_mail_test.php
```

Avec MSMTP correctement configuré, votre serveur peut maintenant envoyer des e-mails via votre fournisseur SMTP spécifié, activant les fonctionnalités essentielles pour WordPress et d'autres applications.

8.11 11. Durcissement et optimisation de la pile LEMP

Introduction au durcissement et à l'optimisation

Après avoir installé la pile LEMP, il est essentiel de durcir et d'optimiser chaque composant pour garantir une sécurité et des performances maximales. Cette section couvre les configurations avancées pour Nginx, MariaDB et PHP-FPM qui permettront de :

1. Améliorer la sécurité en réduisant les surfaces d'attaque
2. Améliorer les performances pour gérer le trafic WordPress
3. Optimiser l'utilisation des ressources sur votre serveur
4. Préparer votre serveur pour les charges de travail de production

Avantages de performance

Des serveurs web correctement optimisés peuvent gérer significativement plus de trafic avec les mêmes ressources matérielles. Ces optimisations peuvent améliorer les temps de chargement des pages de 30 à 50 % et augmenter le nombre d'utilisateurs simultanés que votre serveur peut gérer.

Optimisation de la configuration Nginx

Nginx est hautement configurable et peut être affiné pour de meilleures performances et une meilleure sécurité. Nous allons organiser notre configuration en fichiers modulaires pour une gestion plus facile.

Préparation

Commençons par créer une sauvegarde de la configuration d'origine et mettre en place une structure de répertoires pour nos fichiers de configuration modulaires :

```
# Naviguer vers le répertoire de configuration Nginx
cd /etc/nginx/

# Créer un répertoire includes pour la configuration modulaire
sudo mkdir includes/

# Créer une sauvegarde de la configuration d'origine
sudo cp nginx.conf nginx.conf.bak

# Modifier le fichier de configuration principal
sudo vim nginx.conf
```

Optimisation du contexte principal

Ajoutez les directives suivantes au contexte principal (en dehors de tout bloc) dans votre fichier `nginx.conf` :

```
# Augmenter le nombre maximum de fichiers ouverts par worker
worker_rlimit_nofile 30000;

# Définir la priorité des processus worker (-20 à 19, plus bas = plus haute priorité)
worker_priority -10;

# Optimiser la résolution du minuteur interne
timer_resolution 100ms;
```

```
# Activer la compilation JIT PCRE pour de meilleures performances regex
pcre_jit on;
```

Ces paramètres :

- Augmentent la limite des descripteurs de fichiers pour les workers Nginx
- Donnent aux workers Nginx une priorité CPU plus élevée
- Optimisent la résolution du minuteur pour de meilleures performances
- Activent la compilation Just-In-Time pour les expressions régulières

Optimisation du contexte Events

Dans le bloc `events {}`, modifiez ou ajoutez ces directives :

```
events {
    # Augmenter le maximum de connexions par worker
    worker_connections 4096;

    # Activer l'accept mutex pour une meilleure distribution des connexions
    accept_mutex on;

    # Définir le délai de l'accept mutex
    accept_mutex_delay 200ms;

    # Utiliser le modèle d'événements epoll efficace sous Linux
    use epoll;
}
```

Ces paramètres optimisent la gestion des connexions par Nginx :

- Augmentent le nombre de connexions simultanées que chaque worker peut gérer
- Améliorent la distribution des connexions entre les processus worker
- Utilisent la méthode de traitement d'événements epoll efficace disponible sous Linux

Création de fichiers de configuration modulaires

Créons des fichiers de configuration séparés pour les différents aspects de Nginx :

```
# Naviguer vers le répertoire includes
cd /etc/nginx/includes/

# Créer les fichiers de configuration pour les différents aspects
sudo touch basic_settings.conf buffers.conf timeouts.conf file_handle_cache.conf gzip.conf
brotli.conf

# Lister les fichiers pour confirmer la création
ls -l
```

Configuration des paramètres de base

Créez le fichier de configuration des paramètres de base :

```
# Modifier la configuration des paramètres de base
sudo vim /etc/nginx/includes/basic_settings.conf
```

Ajoutez le contenu suivant :

```
##
# PARAMETRES DE BASE
##
# Définir l'encodage des caractères
charset utf-8;

# Activer la distribution efficace des fichiers
sendfile on;
sendfile_max_chunk 512k;

# Optimiser les paramètres TCP
tcp_nopush on;
tcp_nodelay on;

# Sécurité : Masquer les informations du serveur
server_tokens off;
more_clear_headers 'Server';
more_clear_headers 'X-Powered';

# Gestion des noms de serveur
server_name_in_redirect off;
server_names_hash_bucket_size 64;

# Paramètres des tables de hachage
variables_hash_max_size 2048;
types_hash_max_size 2048;

# Types MIME
include /etc/nginx/mime.types;
default_type application/octet-stream;
```

Ces paramètres :

- Définissent UTF-8 comme encodage de caractères par défaut
- Activent la distribution efficace des fichiers avec sendfile
- Optimisent TCP pour de meilleures performances
- Masquent les informations du serveur pour une meilleure sécurité
- Configurent les tables de hachage pour de meilleures performances

Configuration des tampons

Créez le fichier de configuration des tampons :

```
# Modifier la configuration des tampons
sudo vim /etc/nginx/includes/buffers.conf
```

Ajoutez le contenu suivant :

```
##
# TAMPONS
##
# Tampons pour les requêtes client
client_body_buffer_size 256k;
client_body_in_file_only off;
```

```
client_header_buffer_size 64k;
client_max_body_size 100m; # Envisagez de réduire après la configuration


# Tampons de connexion et d'E/S
connection_pool_size 512;
directio 4m;
ignore_invalid_headers on;
large_client_header_buffers 8 64k;
output_buffers 8 256k;
postpone_output 1460;
request_pool_size 32k;
```

Ces paramètres de tampons :

- Optimisent l'utilisation de la mémoire pour les requêtes client
- Définissent des tailles de tampons appropriées pour les en-têtes et le contenu du corps
- Configurent les opérations d'E/S pour de meilleures performances
- Définissent une taille maximale raisonnable du corps pour les téléchargements de fichiers

Configuration des délais d'attente

Créez le fichier de configuration des délais d'attente :

```
# Modifier la configuration des délais d'attente
sudo vim /etc/nginx/includes/timeouts.conf
```

Ajoutez le contenu suivant :

```
##
# DELAIS D'ATTENTE
##
# Délais de connexion
keepalive_timeout 5;
keepalive_requests 500;
lingering_time 20s;
lingering_timeout 5s;
keepalive_disable msie6;

# Réinitialiser les connexions expirées
reset_timedout_connection on;

# Délais de requête
send_timeout 15s;
client_header_timeout 8s;
client_body_timeout 10s;
```

Ces paramètres de délais d'attente :

- Optimisent la gestion des connexions et l'utilisation des ressources
- Empêchent les clients lents de consommer les ressources du serveur
- Ferment les connexions inactives après 5 secondes
- Autorisent jusqu'à 500 requêtes par connexion avant la fermeture

Paramètres de compression

Configuration Gzip

Créez le fichier de configuration de compression Gzip :

```
# Modifier la configuration gzip
sudo vim /etc/nginx/includes/gzip.conf
```

Ajoutez le contenu suivant :

```
##
# GZIP
##
# Activer la compression gzip
gzip on;
gzip_vary on;
gzip_disable "MSIE [1-6]\.";
gzip_static on;

# Paramètres gzip
gzip_min_length 1400;
gzip_buffers 32 8k;
gzip_http_version 1.0;
gzip_comp_level 5;
gzip_proxied any;

# Types MIME à compresser
gzip_types text/plain text/css text/xml application/javascript application/x-javascript
           application/xml application/xml+rss application/ecmascript application/json
           image/svg+xml;
```

Configuration Brotli

Créez le fichier de configuration de compression Brotli :

```
# Modifier la configuration brotli
sudo vim /etc/nginx/includes/brotli.conf
```

Ajoutez le contenu suivant :

```
##
# BROTLI
##
# Activer la compression Brotli
brotli on;
brotli_comp_level 6;
brotli_static on;

# Types MIME à compresser avec Brotli
brotli_types application/atom+xml application/javascript application/json application/rss+xml
             application/vnd.ms-fontobject application/x-font-opentype
             application/x-font-truetype
             application/x-font-ttf application/x-javascript application/xhtml+xml
             application/xml
```

```
            font/eot font/opentype font/otf font/truetype image/svg+xml
            image/vnd.microsoft.icon
            image/x-icon image/x-win-bitmap text/css text/javascript text/plain text/xml;
```

Avantages de la compression :

- Réduit l'utilisation de la bande passante de 70 à 90 % pour le contenu textuel
- Améliore les temps de chargement des pages pour les visiteurs
- Brotli obtient généralement une compression 15 à 25 % meilleure que Gzip

Configuration du cache des descripteurs de fichiers

Créez le fichier de configuration du cache des descripteurs de fichiers :

```
# Modifier la configuration du cache des descripteurs de fichiers
sudo vim /etc/nginx/includes/file_handle_cache.conf
```

Ajoutez le contenu suivant :

```
##
# CACHE DES DESCRIPTEURS DE FICHIERS
##
# Mettre en cache les descripteurs de fichiers ouverts et les informations
open_file_cache max=50000 inactive=60s;
open_file_cache_valid 120s;
open_file_cache_min_uses 2;
open_file_cache_errors off;
```

Ces paramètres :

- Mettent en cache les informations sur les fichiers ouverts pour réduire les E/S disque
- Stockent jusqu'à 50 000 entrées de fichiers dans le cache
- Suppriment les entrées non accédées pendant 60 secondes
- Valident les entrées du cache toutes les 120 secondes
- Nécessitent au moins 2 utilisations avant de mettre un fichier en cache

Intégration des fichiers de configuration dans Nginx

Maintenant que nous avons créé tous nos fichiers de configuration modulaires, nous devons les intégrer dans le fichier de configuration principal de Nginx.

Commençons par modifier le fichier de configuration principal :

```
# Modifier le fichier de configuration principal de Nginx
sudo vim /etc/nginx/nginx.conf
```

Suppression des blocs de configuration par défaut

Dans le bloc `http {}` de votre fichier nginx.conf, vous devrez supprimer ou commenter les sections par défaut suivantes :

```
##
# Basic Settings
##

##
# SSL Settings
##
```

```
##
# Logging Settings
##

##
# Gzip Settings
##
```

Ajout des directives d'inclusion

Remplacez les sections supprimées par des directives d'inclusion vers nos fichiers de configuration modulaires. Votre bloc `http {}` devrait ressembler à ceci :

```
http {
    # Paramètres de base
    include /etc/nginx/includes/basic_settings.conf;

    # Paramètres de tampons
    include /etc/nginx/includes/buffers.conf;

    # Paramètres de délais d'attente
    include /etc/nginx/includes/timeouts.conf;

    # Paramètres du cache des descripteurs de fichiers
    include /etc/nginx/includes/file_handle_cache.conf;

    # Paramètres SSL
    # (Conserver les paramètres SSL existants ici)

    # Paramètres de journalisation
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    # Paramètres de compression
    include /etc/nginx/includes/gzip.conf;
    include /etc/nginx/includes/brotli.conf;

    # Configurations des hôtes virtuels
    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

Cette approche modulaire rend votre configuration :

- Plus facile à maintenir et à mettre à jour
- Plus organisée et lisible
- Plus simple à dépanner en cas de problème

Test et application de la configuration

Après avoir effectué toutes les modifications, il est essentiel de tester la configuration avant de l'appliquer :

```
# Tester la configuration Nginx pour les erreurs de syntaxe
sudo nginx -t
```

Si le test est réussi, rechargez Nginx pour appliquer les modifications :

```
# Recharger Nginx avec la nouvelle configuration
sudo systemctl reload nginx
```

Vérification des limites de descripteurs de fichiers

Pour s'assurer que nos paramètres de limite de descripteurs de fichiers sont correctement appliqués :

```
# Trouver les processus worker Nginx
ps aux | grep nginx

# Vérifier les limites de descripteurs de fichiers pour un processus worker
# Remplacez XXXX par le PID réel de la commande précédente
sudo cat /proc/XXXX/limits | grep "open files"
```

Si vous devez augmenter davantage la limite des descripteurs de fichiers, vous pouvez modifier la directive `worker_rlimit_nofile` dans le contexte principal de nginx.conf :

```
# Modifier nginx.conf à nouveau si nécessaire
sudo vim /etc/nginx/nginx.conf
```

Augmentez la valeur si nécessaire :

```
# Augmenter la limite des descripteurs de fichiers si nécessaire
worker_rlimit_nofile 45000;
```

Après toute modification, testez et rechargez Nginx à nouveau :

```
# Tester la configuration
sudo nginx -t

# Recharger Nginx
sudo systemctl reload nginx
```

Création d'alias Bash utiles

Les alias Bash peuvent vous faire gagner du temps en créant des raccourcis pour les commandes fréquemment utilisées. C'est particulièrement utile lors de la gestion de votre serveur.

Mise en place des alias de gestion serveur

Créez ou modifiez votre fichier `.bash_aliases` :

```
# Modifier votre fichier d'alias bash
vim ~/.bash_aliases
```

Ajoutez les alias utiles suivants pour la gestion du serveur :

```
# Mise à jour et maintenance système
alias server_update='sudo apt update && sudo apt upgrade && sudo apt autoremove'

# Gestion Nginx
alias ngt='sudo nginx -t'                                # Tester la configuration Nginx
alias ngr='sudo systemctl reload nginx'                  # Recharger Nginx
```

```
alias ngsa='cd /etc/nginx/sites-available/ && ls'        # Naviguer vers sites-available
alias ngin='cd /etc/nginx/includes/ && ls'               # Naviguer vers le répertoire includes

# Gestion PHP-FPM
alias fpmr='sudo systemctl restart php8.3-fpm'           # Redémarrer PHP-FPM
```

Activation de vos alias

Pour rendre vos alias disponibles dans votre session actuelle :

```
# Sourcer votre fichier d'alias bash
source ~/.bash_aliases
```

Pour tester que vos alias fonctionnent :

```
# Tester la configuration nginx avec l'alias
ngt
```

Gain de temps

Ces alias peuvent réduire significativement la saisie et vous aider à gérer votre serveur plus efficacement. Vous pouvez ajouter d'autres alias pour d'autres tâches courantes au fur et à mesure que vous vous familiarisez avec votre flux de travail.

Optimisation et sécurisation de MariaDB

MariaDB est le serveur de base de données qui stocke toutes vos données WordPress. Sa configuration correcte est cruciale tant pour la sécurité que pour les performances.

Sécurisation de l'installation de MariaDB

Commencez par exécuter le script d'installation sécurisée de MariaDB pour supprimer les paramètres par défaut non sécurisés et verrouiller l'accès :

```
# Exécuter le script de sécurité
sudo mysql_secure_installation
```

Comprendre les options du script de sécurité

Lors de l'exécution du script `mysql_secure_installation`, plusieurs options de sécurité vous seront présentées. Voici ce que chaque option signifie et les paramètres recommandés :

1. **Mot de passe root** : Si c'est la première fois que vous exécutez le script, il vous sera demandé de définir un mot de passe root. Choisissez un mot de passe fort et conservez-le en lieu sûr.
2. **Authentification par socket Unix** : Cette option vous permet de vous authentifier en utilisant le fichier socket Unix au lieu d'un mot de passe. Pour la plupart des configurations WordPress, vous pouvez répondre `n`.
3. **Changer le mot de passe root** : Si vous avez déjà défini un mot de passe root, vous pouvez choisir de le changer ou de le conserver en répondant `n`.
4. **Supprimer les utilisateurs anonymes** : Les utilisateurs anonymes sont un risque de sécurité et doivent être supprimés en environnement de production. Répondez `Y`.
5. **Interdire la connexion root à distance** : Empêcher la connexion root à distance est une bonne pratique de sécurité. Répondez `Y`.
6. **Supprimer la base de données de test** : La base de données de test n'est pas nécessaire en production et doit être supprimée. Répondez `Y`.

7. **Recharger les tables de privilèges** : Cela applique immédiatement toutes les modifications. Répondez `Y`.

Bonnes pratiques de sécurité

Complétez toujours l'intégralité du processus `mysql_secure_installation` sur les serveurs de production. Ces mesures de sécurité aident à protéger votre base de données contre les vecteurs d'attaque courants.

Optimisation des performances de MariaDB

Après avoir sécurisé MariaDB, optimisons ses performances pour WordPress. Commencez par créer une sauvegarde du fichier de configuration :

```
# Naviguer vers le répertoire de configuration de MariaDB
cd /etc/mysql/mariadb.conf.d/

# Créer une sauvegarde de la configuration du serveur
sudo cp 50-server.cnf 50-server.cnf.bak

# Modifier le fichier de configuration
sudo vim 50-server.cnf
```

Activation du Performance Schema

Le Performance Schema fournit une instrumentation pour aider à surveiller l'exécution du serveur MariaDB à bas niveau. Ajoutez ces lignes à la section `[mysqld]` :

```
# Performance Schema
performance_schema=ON
performance-schema-instrument='stage/%=ON'
performance-schema-consumer-events-stages-current=ON
performance-schema-consumer-events-stages-history=ON
performance-schema-consumer-events-stages-history-long=ON
```

Désactivation des recherches DNS

Ajoutez cette ligne pour améliorer la vitesse de connexion en empêchant les recherches DNS :

```
# Désactiver les recherches DNS pour les connexions client
skip-name-resolve
```

Cela empêche le serveur de tenter de résoudre les noms d'hôte des clients, ce qui peut ralentir les connexions, en particulier lorsque le DNS est lent ou indisponible.

Configuration du cache de requêtes

MariaDB inclut une fonctionnalité de cache de requêtes qui peut améliorer les performances pour les charges de travail à forte lecture en mettant en cache les résultats des requêtes SELECT. Vous pouvez vérifier les paramètres actuels du cache de requêtes :

```
# Se connecter à MariaDB
sudo mysql -u root -p

# Vérifier si le cache de requêtes est disponible
MariaDB [(none)]> show variables like 'have_query_cache';
```

```
# Vérifier les paramètres du cache de requêtes
MariaDB [(none)]> show variables like 'query_cache%';
```

Pour les sites WordPress avec un trafic modéré, vous pouvez activer et configurer le cache de requêtes en ajoutant ces lignes à la section `[mysqld]` :

```
# Configuration du cache de requêtes
query_cache_type = 1
query_cache_size = 16M
query_cache_limit = 1M
```

Considérations sur le cache de requêtes

Bien que le cache de requêtes puisse améliorer les performances pour les sites avec plus de lectures que d'écritures, il peut en réalité diminuer les performances sur les sites à fort trafic avec des changements de données fréquents. Surveillez les performances de votre site et ajustez en conséquence.

Optimisation des journaux binaires

Les journaux binaires enregistrent toutes les modifications de votre base de données et sont utilisés pour la réplication et la récupération ponctuelle. Par défaut, ces journaux sont conservés pendant 10 jours, ce qui peut consommer un espace disque important. Vous pouvez vérifier et modifier la période de rétention :

```
# Vérifier le paramètre actuel d'expiration des journaux binaires
MariaDB [(none)]> show variables like 'expire_logs_days';

# Définir l'expiration des journaux binaires après 3 jours
MariaDB [(none)]> set global expire_logs_days = 3;

# Vérifier la modification
MariaDB [(none)]> show variables like 'expire_logs_days';

# Purger les journaux binaires pour appliquer les modifications
MariaDB [(none)]> flush binary logs;
```

Pour rendre cette modification permanente, ajoutez cette ligne à la section `[mysqld]` de votre configuration MariaDB :

```
# Paramètres des journaux binaires
expire_logs_days = 3
```

Optimisation du pool de tampons InnoDB

InnoDB est le moteur de stockage par défaut de MariaDB et est optimisé pour les performances et la fiabilité. Le pool de tampons est l'un des paramètres les plus importants à régler pour les performances :

```
# Vérifier les paramètres actuels du pool de tampons InnoDB
MariaDB [(none)]> show variables like '%innodb_buffer%';

# Vérifier les paramètres des fichiers journaux InnoDB
MariaDB [(none)]> show variables like '%innodb_log%';
```

Pour des performances optimales, ajoutez ces paramètres à la section `[mysqld]` de votre configuration MariaDB :

```
# Paramètres InnoDB
innodb_buffer_pool_size = 800M
innodb_log_file_size = 200M
```

Dimensionnement du pool de tampons

Le `innodb_buffer_pool_size` devrait être défini à environ 70-80 % de la RAM disponible pour les serveurs de base de données dédiés, ou 25-30 % pour les serveurs qui exécutent également le serveur web et d'autres services. Pour un serveur avec 4 Go de RAM exécutant la pile LEMP complète, 800 Mo à 1 Go est un bon point de départ.

Après avoir effectué ces modifications, redémarrez MariaDB pour les appliquer :

```
# Redémarrer MariaDB
sudo systemctl restart mariadb
```

Optimisation avancée InnoDB

Pour les serveurs plus importants avec plus de RAM, vous pouvez optimiser davantage InnoDB :

```
# Paramètres InnoDB avancés pour les serveurs avec 8 Go+ de RAM
innodb_buffer_pool_size = 2048M
innodb_log_file_size = 512M
```

```
# Arrêter MariaDB
sudo systemctl stop mariadb

# Effectuer les modifications de configuration
sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf

# Démarrer MariaDB
sudo systemctl start mariadb
```

Modification de la taille du fichier journal

Lors de la modification de `innodb_log_file_size`, vous devez arrêter complètement MariaDB avant d'effectuer la modification :
Cela est nécessaire car les fichiers journaux doivent être recréés avec la nouvelle taille au démarrage du serveur.

Utilisation de MySQLTuner pour une optimisation avancée

MySQLTuner est un script Perl qui analyse les performances de votre MariaDB et fournit des recommandations pour les améliorer :

```
# Télécharger MySQLTuner
wget http://mysqltuner.pl/ -O mysqltuner.pl

# Le rendre exécutable
chmod +x mysqltuner.pl
```

```
# Exécuter MySQLTuner (meilleur après au moins 24 heures de fonctionnement du serveur)
sudo ./mysqltuner.pl
```

MySQLTuner analysera votre configuration et vos schémas d'utilisation actuels, puis fournira des recommandations spécifiques pour optimiser votre installation MariaDB. C'est particulièrement précieux après que votre site WordPress a fonctionné pendant un certain temps, car il peut identifier les goulots d'étranglement basés sur l'utilisation réelle.

Augmentation des limites de descripteurs de fichiers de MariaDB

MariaDB, comme d'autres services, est limitée par le nombre de fichiers ouverts qu'elle peut avoir. Pour les bases de données très sollicitées, les limites par défaut peuvent être trop basses :

```
# Vérifier les limites actuelles de descripteurs de fichiers pour MariaDB
ps aux | grep mysql
cat /proc/$(ps aux | grep mysql | grep -v grep | awk '{print $2}')/limits | grep "Max open
files"
```

Pour augmenter les limites de descripteurs de fichiers pour MariaDB :

```
# Naviguer vers le répertoire systemd
cd /etc/systemd/system/

# Créer un répertoire pour les surcharges du service MariaDB s'il n'existe pas
sudo mkdir -p mariadb.service.d/

# Naviguer vers le nouveau répertoire
cd mariadb.service.d/

# Créer un fichier de configuration des limites
sudo nano limits.conf
```

Ajoutez le contenu suivant au fichier limits.conf :

```
[Service]
LimitNOFILE=40000
```

Appliquez les modifications :

```
# Recharger systemd pour reconnaître les modifications
sudo systemctl daemon-reload

# Redémarrer MariaDB
sudo systemctl restart mariadb

# Vérifier les nouvelles limites
cat /proc/$(ps aux | grep mysql | grep -v grep | awk '{print $2}')/limits | grep "Max open
files"
```

Vous devriez voir que la valeur "Max open files" a été augmentée à 40000, ce qui permettra à MariaDB de gérer plus de connexions et d'opérations simultanées.

Durcissement et optimisation de PHP-FPM

PHP-FPM (FastCGI Process Manager) est l'implémentation PHP qui fonctionne avec Nginx. La configuration correcte de PHP-FPM est essentielle tant pour la sécurité que pour les performances.

Paramètres de sécurité et de performance PHP

Modifiez le fichier de configuration PHP :

```
# Trouver le fichier de configuration PHP principal
sudo find /etc/php/8.3/ -name php.ini

# Modifier le fichier de configuration PHP (ajustez le chemin si nécessaire)
sudo vim /etc/php/8.3/fpm/php.ini
```

Ajoutez ou modifiez les paramètres suivants :

```
; Paramètres de sécurité
allow_url_fopen = Off        ; Empêche PHP d'ouvrir des fichiers distants
cgi.fix_pathinfo = 0         ; Empêche les vulnérabilités de traversée de chemin
expose_php = Off             ; Masque la version de PHP dans les en-têtes HTTP

; Paramètres de performance
upload_max_filesize = 100M   ; Taille maximale autorisée pour les fichiers téléchargés
post_max_size = 125M         ; Taille maximale des données POST
(devrait être supérieure à upload_max_filesize)
max_input_vars = 3000        ; Variables d'entrée maximum
(important pour les formulaires WordPress complexes)
memory_limit = 256M          ; Mémoire maximale qu'un script peut consommer

; Limites de ressources
rlimit_files = 32768         ; Nombre maximum de fichiers ouverts
rlimit_core = unlimited      ; Autoriser les core dumps pour le débogage
```

Configuration du pool PHP-FPM

Pour de meilleures performances, vous pouvez également optimiser la configuration du pool PHP-FPM :

```
# Modifier le fichier www.conf
sudo vim /etc/php/8.3/fpm/pool.d/www.conf
```

Trouvez et modifiez ces paramètres :

```
; Paramètres du gestionnaire de processus
pm = dynamic                  ; Gestionnaire de processus dynamique
pm.max_children = 50          ; Nombre maximum de processus enfants
pm.start_servers = 5          ; Nombre de processus enfants créés au démarrage
pm.min_spare_servers = 5      ; Nombre minimum de processus serveur inactifs
pm.max_spare_servers = 35     ; Nombre maximum de processus serveur inactifs
pm.max_requests = 500         ; Nombre de requêtes que chaque processus enfant doit exécuter
avant de se relancer
```

Ajustement des paramètres PHP-FPM

Les valeurs optimales pour ces paramètres dépendent de la mémoire disponible et du CPU de votre serveur. Pour un serveur avec 4 Go de RAM, les valeurs ci-dessus sont un bon point de départ. Pour les serveurs avec plus de ressources, vous pouvez augmenter ces valeurs proportionnellement.

Application des modifications

Après avoir effectué ces modifications, redémarrez PHP-FPM et rechargez Nginx :

```
# Redémarrer PHP-FPM
sudo systemctl restart php8.3-fpm

# Recharger Nginx
sudo systemctl reload nginx
```

Conclusion

Vous avez maintenant durci et optimisé avec succès votre pile LEMP (Linux, Nginx, MariaDB et PHP) pour l'hébergement WordPress. Ces configurations fournissent un bon équilibre entre sécurité et performances pour la plupart des sites WordPress.

N'oubliez pas de surveiller les performances de votre serveur et d'ajuster ces paramètres selon vos besoins en fonction de votre charge de travail spécifique et de vos schémas de trafic. Des outils comme MySQLTuner, les pages d'état Nginx et les pages d'état PHP-FPM peuvent vous aider à identifier les goulots d'étranglement et à affiner votre configuration.

8.12 13. Configuration des blocs serveur Nginx

Comprendre les blocs serveur Nginx

Les blocs serveur Nginx (similaires aux hôtes virtuels d'Apache) vous permettent d'héberger plusieurs sites web sur un seul serveur. Chaque site web peut avoir sa propre configuration, son propre répertoire racine et son propre nom de domaine.

Pourquoi les blocs serveur sont importants

Les blocs serveur sont essentiels pour héberger plusieurs sites WordPress sur un seul serveur. Ils vous permettent de garder les fichiers de chaque site séparés et d'appliquer des configurations différentes à chaque site selon les besoins.

Anatomie d'un bloc serveur

Un bloc serveur est défini dans la directive `server { }` des fichiers de configuration Nginx. Décortiquons les composants d'un bloc serveur en le construisant étape par étape :

Structure de base d'un bloc serveur

Le bloc serveur le plus basique est simplement un bloc vide :

```
server {
    # Les directives de configuration vont ici
}
```

Ajout d'un port d'écoute

Ensuite, nous spécifions sur quel port Nginx doit écouter (généralement le port 80 pour HTTP) :

```
server {
    listen 80;
    # Plus de directives de configuration
}
```

Définition du nom de serveur

La directive `server_name` indique à Nginx quels noms de domaine doivent être gérés par ce bloc serveur :

```
server {
    listen 80;
    server_name example.com www.example.com;
    # Plus de directives de configuration
}
```

Spécification du répertoire racine

La directive `root` définit l'emplacement des fichiers du site web :

```
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example.com/public_html;
    # Plus de directives de configuration
}
```

Définition du fichier d'index par défaut

La directive `index` spécifie quel fichier doit être servi lorsqu'un répertoire est demandé :

```
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example.com/public_html;
    index index.php index.html index.htm;
    # Plus de directives de configuration
}
```

Configuration du traitement des URL

Le bloc `location /` avec la directive `try_files` est crucial pour que WordPress gère les permaliens personnalisés :

```
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example.com/public_html;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }
    # Plus de directives de configuration
}
```

Traitement des fichiers PHP

Enfin, nous ajoutons un bloc location pour gérer les fichiers PHP et les transmettre à PHP-FPM :

```
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example.com/public_html;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
    }
}
```

Amélioration des blocs serveur avec des optimisations de performance

Un bloc serveur basique fonctionne bien, mais nous pouvons l'améliorer avec des optimisations de performance comme la journalisation, la mise en cache du navigateur et le réglage FastCGI.

Ajout de la configuration de journalisation

Pour une meilleure surveillance et un meilleur dépannage, ajoutez des journaux d'accès et d'erreur personnalisés pour chaque site :

```
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example.com/public_html;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
    }

    # Configuration de journalisation personnalisée
    access_log /var/log/nginx/access_example.com.log combined buffer=256k flush=60m;
    error_log /var/log/nginx/error_example.com.log;
}
```

Les paramètres `buffer` et `flush` optimisent les performances d'écriture des journaux en mettant les entrées en tampon en mémoire et en les écrivant périodiquement sur le disque.

Mise en place de la mise en cache du navigateur

La mise en cache du navigateur améliore significativement les performances du site web en indiquant aux navigateurs de stocker les ressources statiques localement. Créez un fichier de configuration réutilisable pour la mise en cache du navigateur :

```
# Créer un répertoire pour les includes s'il n'existe pas
sudo mkdir -p /etc/nginx/includes/

# Créer le fichier de configuration de mise en cache du navigateur
sudo vim /etc/nginx/includes/browser_caching.conf
```

Ajoutez le contenu suivant au fichier :

```
# Mettre en cache les images et fichiers multimédias pendant 1 an
location ~* \.(webp|gif|jpg|jpeg|png|ico|wmv|avi|asf|asx|mpg|mpeg|mp4|pls|mp3|mid|wav|swf|flv|
exe|zip|tar|rar|gz|tgz|bz2|uha|7z|doc|docx|xls|xlsx|pdf|iso)$ {
    expires 365d;
    add_header Cache-Control "public, no-transform";
    access_log off;
}

# Mettre en cache les fichiers JavaScript pendant 30 jours
location ~* \.(js)$ {
    expires 30d;
    add_header Cache-Control "public, no-transform";
```

```
    access_log off;
}

# Mettre en cache les fichiers CSS pendant 30 jours
location ~* \.(css)$ {
    expires 30d;
    add_header Cache-Control "public, no-transform";
    access_log off;
}

# Mettre en cache les polices web pendant 30 jours
location ~* \.(eot|svg|ttf|woff|woff2)$ {
    expires 30d;
    add_header Cache-Control "public, no-transform";
    access_log off;
}
```

Optimisation des paramètres FastCGI

FastCGI est le protocole utilisé pour communiquer entre Nginx et PHP-FPM. L'optimisation de ces paramètres peut améliorer les performances :

```
# Créer le fichier de configuration d'optimisation FastCGI
sudo vim /etc/nginx/includes/fastcgi_optimize.conf
```

Ajoutez le contenu suivant au fichier :

```
# Paramètres de délai de connexion
fastcgi_connect_timeout 60;
fastcgi_send_timeout 180;
fastcgi_read_timeout 180;

# Paramètres de tampon pour le traitement des réponses
fastcgi_buffer_size 512k;
fastcgi_buffers 512 16k;
fastcgi_busy_buffers_size 1m;
fastcgi_temp_file_write_size 4m;
fastcgi_max_temp_file_size 4m;

# Gestion des erreurs
fastcgi_intercept_errors on;
```

Comprendre les paramètres FastCGI

- **Paramètres de délai** : Contrôlent combien de temps Nginx attend la réponse de PHP-FPM
- **Paramètres de tampon** : Déterminent la quantité de mémoire allouée pour le traitement des réponses PHP
- **Gestion des erreurs** : Lorsqu'activée, Nginx peut afficher ses propres pages d'erreur au lieu des erreurs PHP

Création d'une configuration complète de bloc serveur

Maintenant, rassemblons tout pour créer une configuration complète de bloc serveur pour un site WordPress :

```
# Créer un nouveau fichier de configuration de bloc serveur
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez la configuration complète suivante :

```
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example.com/public_html;
    index index.php;

    # Permaliens personnalisés WordPress
    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }

    # Traitement PHP avec paramètres FastCGI optimisés
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        include /etc/nginx/includes/fastcgi_optimize.conf;
    }

    # Inclure les règles de mise en cache du navigateur
    include /etc/nginx/includes/browser_caching.conf;

    # Journalisation personnalisée
    access_log /var/log/nginx/access_example.com.log combined buffer=256k flush=60m;
    error_log /var/log/nginx/error_example.com.log;
}
```

Activation du bloc serveur

Après avoir créé la configuration du bloc serveur, vous devez l'activer en créant un lien symbolique dans le répertoire `sites-enabled` :

```
# Vérifier les sites existants
cd /etc/nginx/
ls sites-available/ sites-enabled/

# Créer un lien symbolique pour activer le site
sudo ln -s /etc/nginx/sites-available/example.com.conf
/etc/nginx/sites-enabled/example.com.conf

# Vérifier que le lien a été créé
ls -l sites-enabled/
```

Test et application de la configuration

Avant de recharger Nginx, il est important de tester la configuration pour s'assurer qu'il n'y a pas d'erreurs de syntaxe :

```
# Tester la configuration Nginx
sudo nginx -t
```

Si le test est réussi, rechargez Nginx pour appliquer les modifications :

```
# Recharger Nginx
sudo systemctl reload nginx
```

Préparation du répertoire racine web

Enfin, assurez-vous que le répertoire racine web existe et a les permissions correctes :

```
# Créer le répertoire racine web s'il n'existe pas
sudo mkdir -p /var/www/example.com/public_html

# Définir la propriété correcte (ajustez le nom d'utilisateur si nécessaire)
sudo chown -R www-data:www-data /var/www/example.com/public_html

# Définir les permissions correctes
sudo chmod -R 755 /var/www/example.com/public_html

# Vérifier le répertoire
ls -la /var/www/example.com/public_html/
```

Sites multiples

Répétez ce processus pour chaque site web que vous souhaitez héberger sur votre serveur, en changeant le nom de domaine, le nom du fichier de bloc serveur et le répertoire racine web selon les besoins.

Avec ces étapes terminées, votre serveur Nginx est maintenant configuré pour servir votre site WordPress. Dans la section suivante, nous verrons comment installer WordPress dans cet environnement préparé.

8.13 14. Installation de WordPress

Introduction à l'installation de WordPress

Après avoir configuré votre pile LEMP et les blocs serveur Nginx, vous êtes prêt à installer WordPress. Ce processus implique la création d'une base de données, le téléchargement de WordPress, sa configuration et la définition des permissions appropriées.

Installation propre

Ce guide couvre une installation WordPress vierge. Si vous migrez un site existant, des étapes supplémentaires seront nécessaires pour transférer votre contenu et votre base de données.

Création d'une base de données pour WordPress

WordPress nécessite une base de données MySQL/MariaDB pour stocker son contenu. Créons-en une :

```
# Se connecter à MariaDB en tant que root
sudo mysql
```

Une fois connecté, créez une base de données et un utilisateur avec les commandes suivantes :

```
-- Créer une nouvelle base de données
CREATE DATABASE wordpress_db;

-- Créer un nouvel utilisateur avec un mot de passe fort
CREATE USER 'wordpress_user'@'localhost' IDENTIFIED BY 'your_strong_password';

-- Accorder à l'utilisateur tous les privilèges sur la base de données
GRANT ALL PRIVILEGES ON wordpress_db.* TO 'wordpress_user'@'localhost';

-- Appliquer les modifications
FLUSH PRIVILEGES;

-- Vérifier les droits accordés
SHOW GRANTS FOR 'wordpress_user'@'localhost';

-- Quitter MariaDB
EXIT;
```

```
cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 30 | head -n 1
```

Note de sécurité

Remplacez `'your_strong_password'` par un vrai mot de passe fort. Vous pouvez générer un mot de passe aléatoire avec :

Commandes de gestion de base de données (référence)

Voici quelques commandes MariaDB utiles pour référence :

```
-- Lister toutes les bases de données
SHOW DATABASES;
```

```
-- Lister tous les utilisateurs
SELECT host, user FROM mysql.user;

-- Supprimer une base de données
DROP DATABASE database_name;

-- Supprimer un utilisateur
DROP USER 'username'@'localhost';

-- Sélectionner une base de données à utiliser
USE database_name;

-- Afficher les tables de la base de données actuelle
SHOW TABLES;

-- Afficher la structure d'une table
DESCRIBE table_name;
```

Téléchargement et extraction de WordPress

Téléchargeons et extrayons maintenant la dernière version de WordPress :

```
# Naviguer vers votre répertoire personnel
cd ~

# Télécharger le dernier paquet WordPress
wget https://wordpress.org/latest.tar.gz

# Extraire l'archive
tar xf latest.tar.gz

# Vérifier l'extraction
ls -la wordpress/
```

Configuration de WordPress

Création et modification du fichier de configuration

```
# Naviguer vers le répertoire WordPress
cd ~/wordpress/

# Créer un fichier de configuration à partir du modèle
cp wp-config-sample.php wp-config.php

# Modifier le fichier de configuration
nano wp-config.php
```

Mettez à jour les paramètres de la base de données dans le fichier de configuration :

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

```
/** Database username */
define( 'DB_USER', 'wordpress_user' );

/** Database password */
define( 'DB_PASSWORD', 'your_strong_password' );

/** Database hostname */
define( 'DB_HOST', 'localhost' );
```

Génération et ajout des clés de sécurité

WordPress utilise des clés de sécurité pour renforcer la sécurité de votre installation. Générez ces clés :

```
# Générer les clés de sécurité
curl -s https://api.wordpress.org/secret-key/1.1/salt/
```

Copiez le résultat et remplacez la section de substitution dans votre fichier `wp-config.php` :

```
/**#@+

 * Authentication unique keys and salts.
 */
define( 'AUTH_KEY',         'put your unique phrase here' );
define( 'SECURE_AUTH_KEY',  'put your unique phrase here' );
define( 'LOGGED_IN_KEY',    'put your unique phrase here' );
define( 'NONCE_KEY',        'put your unique phrase here' );
define( 'AUTH_SALT',        'put your unique phrase here' );
define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
define( 'LOGGED_IN_SALT',   'put your unique phrase here' );
define( 'NONCE_SALT',       'put your unique phrase here' );
/**#@-*/
```

Ajout de paramètres de sécurité supplémentaires

Ajoutez ces lignes à la fin de votre fichier `wp-config.php`, avant la ligne qui dit “That’s all, stop editing!” :

```
/** Allow direct updates without FTP */
define('FS_METHOD', 'direct');

/** Disable editing of themes and plugins using the built-in editor */
define('DISALLOW_FILE_EDIT', true);

/** Disable automatic WordPress updates */
define('WP_AUTO_UPDATE_CORE', false);
define('AUTOMATIC_UPDATER_DISABLED', true);
```

Installation des fichiers WordPress

Maintenant que WordPress est configuré, copions les fichiers vers votre répertoire racine web :

```
# Retourner à votre répertoire personnel
cd ~
```

```
# Copier les fichiers WordPress vers votre répertoire racine web
sudo rsync -artv wordpress/ /var/www/example.com/public_html/

# Définir la propriété correcte
cd /var/www/example.com/
sudo chown -R www-data:www-data public_html/

# Vérifier les permissions
ls -la public_html/
```

Finalisation de l'installation

Ouvrez votre navigateur web et naviguez vers votre domaine (par exemple, `http://example.com`). Vous devriez voir l'assistant d'installation WordPress. Suivez ces étapes :

1. Sélectionnez votre langue et cliquez sur "Continuer"
2. Saisissez les informations du site :
   - Titre du site : Le nom de votre site web
   - Identifiant : Créez un identifiant administrateur (n'utilisez pas "admin")
   - Mot de passe : Utilisez un mot de passe fort
   - Votre e-mail : Saisissez votre adresse e-mail
   - Visibilité pour les moteurs de recherche : Cochez si vous souhaitez décourager les moteurs de recherche d'indexer votre site
3. Cliquez sur "Installer WordPress"

Une fois l'installation terminée, vous pouvez vous connecter avec votre identifiant et mot de passe à l'adresse `http://example.com/wp-login.php`.

Post-installation

Après avoir installé WordPress, il est important de durcir et d'optimiser votre installation pour la sécurité et les performances. Nous aborderons ces étapes dans les sections suivantes.

8.14 15. Durcissement de la sécurité WordPress

Introduction à la sécurité WordPress

WordPress alimente plus de 40 % de tous les sites web sur Internet, ce qui en fait une cible privilégiée pour les attaquants. Cette section couvre les mesures de sécurité essentielles pour protéger votre installation WordPress contre les menaces courantes.

La sécurité est un processus continu

La sécurité n'est pas une configuration ponctuelle mais un processus continu. Mettez régulièrement à jour le noyau WordPress, les thèmes et les extensions, et surveillez vos journaux pour détecter toute activité suspecte.

Les mesures de sécurité de ce guide suivent une approche de défense en profondeur, implémentant plusieurs couches de protection :

1. **Isolation** : Séparer les sites WordPress en utilisant des utilisateurs dédiés et des pools PHP-FPM
2. **Contrôle d'accès** : Restreindre l'accès au système de fichiers et définir les permissions appropriées
3. **Chiffrement** : Implémenter SSL/TLS avec des paramètres de sécurité optimaux
4. **Filtrage des requêtes** : Bloquer les requêtes malveillantes avant qu'elles n'atteignent WordPress
5. **Durcissement de l'application** : Limiter les fonctionnalités WordPress qui pourraient être exploitées

En implémentant ces mesures, vous réduirez significativement la surface d'attaque de votre installation WordPress et rendrez beaucoup plus difficile la compromission de votre site par les attaquants.

Surveillance de la sécurité avec les journaux

Avant d'implémenter des mesures de sécurité, il est important de comprendre comment surveiller votre serveur pour détecter les activités suspectes.

Vérification des journaux Nginx

Les journaux Nginx fournissent des informations précieuses sur les requêtes vers votre site web, y compris les tentatives d'attaque potentielles :

```
# Naviguer vers le répertoire des journaux Nginx
cd /var/log/nginx

# Lister les fichiers journaux disponibles
ls

# Afficher le contenu d'un fichier journal spécifique
sudo cat access_example.com.log

# Afficher les journaux avec pagination pour les gros fichiers
sudo less error_example.com.log
```

Recherchez des schémas comme des tentatives de connexion échouées répétées, des méthodes HTTP inhabituelles ou des requêtes vers des fichiers PHP inexistants, qui pourraient indiquer des tentatives d'attaque.

Séparation des sites avec des pools PHP - Partie 1

Création d'un utilisateur dédié pour WordPress

Un principe de sécurité clé est d'exécuter WordPress sous un compte utilisateur dédié plutôt que l'utilisateur www-data par défaut. Cela offre une meilleure isolation et un meilleur contrôle.

Commencez par créer un utilisateur dédié pour votre site WordPress :

```
# Créer un nouvel utilisateur pour votre site WordPress
sudo useradd wordpress_user
```

Configuration des permissions de groupe utilisateur

Configurez les relations de groupe appropriées pour permettre au serveur web et à votre utilisateur spécifique au site d'accéder aux fichiers nécessaires :

```
# Ajouter www-data au groupe de l'utilisateur WordPress
sudo usermod -a -G wordpress_user www-data

# Ajouter l'utilisateur WordPress au groupe www-data
sudo usermod -a -G www-data wordpress_user

# Si nécessaire, ajouter votre utilisateur admin au groupe de l'utilisateur WordPress
sudo usermod -a -G wordpress_user admin_username
```

Cette configuration crée une isolation appropriée tout en maintenant les permissions d'accès nécessaires.

Séparation des sites avec des pools PHP - Partie 2

Par défaut, tout le traitement PHP passe par un seul pool PHP-FPM. La création d'un pool dédié pour chaque site WordPress offre une meilleure isolation de sécurité et un meilleur contrôle des ressources.

Création d'un pool PHP-FPM spécifique au site

```
# Naviguer vers le répertoire de configuration des pools PHP-FPM
cd /etc/php/8.3/fpm/pool.d/

# Créer une nouvelle configuration de pool en copiant celle par défaut
sudo cp www.conf example.com.conf

# Modifier la nouvelle configuration de pool
sudo vim example.com.conf
```

Configuration du pool PHP-FPM

Modifiez le fichier de configuration du pool avec ces paramètres orientés sécurité :

```
; Changer le nom du pool de 'www' au nom de votre site
[example]

; Définir l'utilisateur et le groupe à votre utilisateur spécifique au site
user = wordpress_user
group = wordpress_user

; Utiliser un fichier socket spécifique au site
listen = /run/php/php8.3-fpm-example.com.sock

; Définir les limites de ressources
rlimit_files = 15000
rlimit_core = 100
```

```
; Désactiver l'affichage des erreurs mais activer la journalisation
php_flag[display_errors] = off
php_admin_flag[log_errors] = on
php_admin_value[error_log] = /var/log/fpm-php.example.com.log
```

Durcissement des permissions du socket

Ajoutez ces directives pour sécuriser davantage le socket PHP-FPM :

```
; Définir la propriété et les permissions du socket
listen.owner = www-data
listen.group = www-data
listen.mode = 0600
```

Ces paramètres :

- Restreignent l'accès au socket uniquement à l'utilisateur du serveur web
- Empêchent les autres utilisateurs du système de se connecter à votre pool PHP-FPM

Création du fichier journal PHP-FPM

```
# Créer le fichier journal
sudo touch /var/log/fpm-php.example.com.log

# Définir la propriété appropriée
sudo chown wordpress_user:www-data /var/log/fpm-php.example.com.log

# Définir les permissions appropriées
sudo chmod 660 /var/log/fpm-php.example.com.log
```

Application de la configuration PHP-FPM

```
# Recharger PHP-FPM pour appliquer les modifications
sudo systemctl reload php8.3-fpm

# Vérifier le chemin du socket dans votre configuration
sudo grep "listen = /" example.com.conf
```

Mise à jour de Nginx pour utiliser le nouveau pool PHP-FPM

Modifiez votre bloc serveur Nginx pour utiliser le nouveau socket PHP-FPM :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Trouvez la section de traitement PHP et mettez à jour la directive `fastcgi_pass` :

```
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/run/php/php8.3-fpm-example.com.sock;
    include /etc/nginx/includes/fastcgi_optimize.conf;
}
```

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx si le test est réussi
sudo systemctl reload nginx
```

Propriété et permissions WordPress

La propriété et les permissions des fichiers sont critiques pour la sécurité de WordPress. L'objectif est de permettre à WordPress de fonctionner normalement tout en restreignant l'accès aux fichiers sensibles.

Configuration standard des permissions

Pour un équilibre entre sécurité et fonctionnalité, utilisez ces permissions :

```
# Définir la propriété à l'utilisateur WordPress
sudo chown -R wordpress_user:wordpress_user /var/www/example.com/public_html/

# Définir les permissions des répertoires à 770
# (propriétaire et groupe : lecture, écriture, exécution)
sudo find /var/www/example.com/public_html/ -type d -exec chmod 770 {} \;

# Définir les permissions des fichiers à 660 (propriétaire et groupe : lecture et écriture)
sudo find /var/www/example.com/public_html/ -type f -exec chmod 660 {} \;

# Définir wp-config.php en lecture seule pour le propriétaire
sudo chmod 400 /var/www/example.com/public_html/wp-config.php
```

Configuration renforcée des permissions

Pour une sécurité maximale (peut nécessiter des ajustements pour certaines extensions) :

```
# Définir la propriété à l'utilisateur WordPress
sudo chown -R wordpress_user:wordpress_user /var/www/example.com/public_html/

# Définir les permissions des répertoires à 550 (propriétaire et groupe : lecture et exécution)
sudo find /var/www/example.com/public_html/ -type d -exec chmod 550 {} \;

# Définir les permissions des fichiers à 440 (propriétaire et groupe : lecture)
sudo find /var/www/example.com/public_html/ -type f -exec chmod 440 {} \;

# Autoriser l'écriture dans le répertoire wp-content pour les mises à jour et les
# téléchargements
sudo find /var/www/example.com/public_html/wp-content/ -type d -exec chmod 770 {} \;
sudo find /var/www/example.com/public_html/wp-content/ -type f -exec chmod 660 {} \;
```

Explication des permissions

- **4** : Permission de lecture
- **2** : Permission d'écriture
- **1** : Permission d'exécution (nécessaire pour que les répertoires soient accessibles)
- Le premier chiffre est pour le propriétaire, le deuxième pour le groupe et le troisième pour les autres

Limiter l'accès au système de fichiers avec PHP open_basedir

La directive `open_basedir` restreint les fichiers auxquels PHP peut accéder, empêchant les attaquants d'accéder aux fichiers en dehors de votre installation WordPress.

Configuration des répertoires temporaires PHP

Commencez par créer un répertoire temporaire dédié pour votre site WordPress :

```
# Créer un répertoire temporaire
cd /var/www/example.com/
sudo mkdir tmp/

# Définir la propriété et les permissions appropriées
sudo chown wordpress_user:wordpress_user tmp/
sudo chmod 770 tmp/
```

Configuration de la restriction open_basedir

Modifiez la configuration de votre pool PHP-FPM pour ajouter la directive open_basedir :

```
# Modifier la configuration du pool PHP-FPM
sudo vim /etc/php/8.3/fpm/pool.d/example.com.conf
```

Ajoutez ces lignes à la configuration :

```
; Définir des répertoires temporaires personnalisés
php_admin_value[upload_tmp_dir] = /var/www/example.com/tmp/
php_admin_value[sys_temp_dir] = /var/www/example.com/tmp/

; Restreindre l'accès aux fichiers PHP à des répertoires spécifiques
php_admin_value[open_basedir] = /var/www/example.com/public_html/:/var/www/example.com/tmp/
```

Appliquez les modifications :

```
# Recharger PHP-FPM
sudo systemctl reload php8.3-fpm
```

Désactivation des fonctions PHP dangereuses

Restreignez les fonctions PHP qui pourraient être utilisées à des fins malveillantes :

```
; Désactiver les fonctions PHP potentiellement dangereuses
php_admin_value[disable_functions] = shell_exec, opcache_get_configuration, opcache_get_status,
disk_total_space, diskfreespace, dl, exec, passthru, pclose, pcntl_alarm, pcntl_exec,
pcntl_fork, pcntl_get_last_error, pcntl_getpriority, pcntl_setpriority, pcntl_signal,
pcntl_signal_dispatch, pcntl_sigprocmask, pcntl_sigtimedwait, pcntl_sigwaitinfo,
pcntl_strerror, pcntl_waitpid, pcntl_wait, pcntl_wexitstatus, pcntl_wifcontinued,
pcntl_wifexited, pcntl_wifsignaled, pcntl_wifstopped, pcntl_wstopsig, pcntl_wtermsig, popen,
posix_getpwuid, posix_kill, posix_mkfifo, posix_setpgid, posix_setsid, posix_setuid,
posix_uname, proc_close, proc_get_status, proc_nice, proc_open, proc_terminate, show_source,
system
```

Compatibilité des fonctions

Certaines extensions WordPress peuvent nécessiter certaines fonctions PHP. Si vous rencontrez des problèmes après avoir implémenté cette restriction, vous devrez peut-être réactiver sélectivement des fonctions spécifiques.

Installer et configurer des certificats SSL GRATUITS - Partie 1

Le chiffrement HTTPS est essentiel pour la sécurité de WordPress. Let's Encrypt fournit des certificats SSL gratuits qui sont reconnus par tous les navigateurs majeurs.

Installation de Certbot

Certbot est un outil qui automatise le processus d'obtention et de renouvellement des certificats Let's Encrypt :

```
# Mettre à jour les listes de paquets
sudo apt update

# Installer Certbot et le plugin Nginx
sudo apt install certbot python3-certbot-nginx
```

Obtention d'un certificat

Utilisez le plugin webroot pour obtenir un certificat sans interrompre votre site web :

```
# Obtenir un certificat pour votre domaine
sudo certbot certonly --webroot -w /var/www/example.com/public_html/ -d example.com -d
www.example.com
```

Suivez les invites pour terminer le processus d'émission du certificat.

Installer et configurer des certificats SSL GRATUITS - Partie 2

Création de paramètres DH forts

Pour renforcer la sécurité de votre implémentation SSL/TLS, créez un fichier de paramètres Diffie-Hellman fort :

```
# Créer le répertoire SSL
cd /etc/nginx/
sudo mkdir ssl/
cd ssl/

# Générer les paramètres DH (cela peut prendre quelques minutes)
sudo openssl dhparam -out dhparam.pem 2048
```

Création de la configuration SSL spécifique au site

Créez un fichier de configuration pour les certificats SSL de votre site :

```
# Créer le fichier de configuration des certificats SSL
sudo vim /etc/nginx/ssl/ssl_certs_example.com.conf
```

Ajoutez le contenu suivant :

```
# Chemins des certificats
ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
```

```
ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;

# Certificat de stapling SSL
ssl_trusted_certificate /etc/letsencrypt/live/example.com/chain.pem;
```

Installer et configurer des certificats SSL GRATUITS - Partie 3

Création de la configuration SSL globale

Créez un fichier de configuration SSL global avec des paramètres de sécurité optimaux :

```
# Créer le fichier de configuration SSL global
sudo vim /etc/nginx/ssl/ssl_all_sites.conf
```

Ajoutez le contenu suivant :

```
# CONFIGURATION RESULTAT EN NOTE A+ SUR SSLLABS.COM
# MISE A JOUR DES DIRECTIVES POUR MAINTENIR LA NOTE A+ - VERIFIER LA DATE
# DATE: OCTOBER 2024

# CACHE SSL ET PROTOCOLES
ssl_session_cache shared:SSL:20m;
ssl_session_timeout 180m;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_prefer_server_ciphers on;
# ssl_ciphers doit etre sur une seule ligne, ne pas repartir sur plusieurs lignes
ssl_ciphers
ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-G
ssl_dhparam /etc/nginx/ssl/dhparam.pem;
ssl_stapling on;
ssl_stapling_verify on;
# resolver defini sur Cloudflare
# timeout peut etre defini jusqu'a 30s
resolver 1.1.1.1 1.0.0.1;
resolver_timeout 15s;
ssl_session_tickets off;
# EN-TETES HSTS
add_header Strict-Transport-Security "max-age=31536000;" always;
# Apres avoir configure TOUS vos sous-domaines - commentez la ligne ci-dessus et decommentez la
# directive ci-dessous
# add_header Strict-Transport-Security "max-age=31536000; includeSubDomains;" always;
# Activer QUIC et HTTP/3
ssl_early_data on;
add_header Alt-Svc 'h3=":$server_port"; ma=86400';
add_header x-quic 'H3';
quic_retry on;
```

Configuration de Nginx pour HTTPS

Mettez à jour votre bloc serveur Nginx pour utiliser HTTPS :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez un bloc serveur pour rediriger HTTP vers HTTPS :

```
# Redirection HTTP vers HTTPS
server {
    listen 80;
    server_name example.com www.example.com;

    # Redirection permanente 301
    return 301 https://example.com$request_uri;
}
```

Configurez le bloc serveur HTTPS :

```
# Bloc serveur HTTPS
server {
    # Activer SSL, HTTP/2 et HTTP/3
    listen 443 ssl;
    http2 on;

    # Support HTTP/3
    listen 443 quic reuseport;
    http3 on;

    server_name example.com www.example.com;

    # Inclure les fichiers de configuration SSL
    include /etc/nginx/ssl/ssl_certs_example.com.conf;
    include /etc/nginx/ssl/ssl_all_sites.conf;

    # Reste de votre configuration serveur...
}
```

```
listen 443 quic;
http3 on;
```

Support HTTP/3

Pour les serveurs avec plusieurs sites, utilisez `reuseport` uniquement sur le premier bloc serveur. Pour les blocs serveur supplémentaires, utilisez :

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx si le test est réussi
sudo systemctl reload nginx
```

Test de la configuration SSL

Vérifiez que votre configuration SSL fonctionne correctement :

```
# Tester la redirection HTTP vers HTTPS
curl -I http://example.com

# Tester la connexion HTTPS
curl -I https://example.com
```

Vous pouvez également tester votre configuration SSL avec des outils en ligne :

- **SSL Labs** - Pour un test SSL/TLS complet
- **HTTP3 Check** - Pour vérifier le support HTTP/3

Configuration du renouvellement automatique des certificats

Les certificats Let's Encrypt expirent après 90 jours, le renouvellement automatique est donc essentiel :

```
# Modifier le crontab root
sudo crontab -e
```

Ajoutez les lignes suivantes :

```
# Renouveler les certificats deux fois par mois et recharger Nginx
00 1 14,28 * * certbot renew --force-renewal --deploy-hook "systemctl reload nginx" >>
/var/log/certbot-renew.log 2>&1
```

Définissez les permissions appropriées pour le fichier journal :

```
# Créer le fichier journal s'il n'existe pas
sudo touch /var/log/certbot-renew.log

# Définir les permissions appropriées
sudo chmod 644 /var/log/certbot-renew.log
```

Durcissement des en-têtes de réponse HTTP

Les en-têtes de sécurité HTTP aident à protéger votre site contre diverses attaques comme le XSS, le clickjacking et le reniflage de type de contenu.

Création d'une configuration d'en-têtes de sécurité

Créez un fichier de configuration pour les en-têtes de sécurité :

```
# Créer le fichier de configuration des en-têtes de sécurité
cd /etc/nginx/includes/
sudo vim http_headers.conf
```

Ajoutez le contenu suivant :

```
# Politique de référent
# Contrôle la quantité d'informations de référent incluse dans les requêtes
add_header Referrer-Policy "strict-origin-when-cross-origin" always;

# Options de type de contenu
# Empêche les navigateurs de deviner le type MIME d'une réponse
add_header X-Content-Type-Options "nosniff" always;

# Options de cadre
# Protège contre le clickjacking en empêchant votre page d'être affichée dans un cadre
```

```
add_header X-Frame-Options "sameorigin" always;

# Protection XSS
# Active le filtre de script intersite (XSS) dans les navigateurs
add_header X-XSS-Protection "1; mode=block" always;

# Politique de permissions
# Contrôle quelles fonctionnalités et API du navigateur peuvent être utilisées
add_header Permissions-Policy
"geolocation=(),midi=(),sync-xhr=(),microphone=(),camera=(),magnetometer=(),gyroscope=(),fullscreen=self,p
always;
```

Création d'une configuration d'en-têtes de mise en cache et de sécurité

Pour le contenu statique, créez une configuration combinant mise en cache et en-têtes de sécurité :

```
# Créer le fichier de configuration des en-têtes de mise en cache et de sécurité
sudo vim /etc/nginx/includes/browser_caching_security_headers.conf
```

Ajoutez le contenu suivant :

```
# Activer la mise en cache du navigateur
expires 30d;
etag on;
if_modified_since exact;
add_header Pragma "public" always;
add_header Cache-Control "public, no-transform" always;

# Inclure les en-têtes de sécurité
include /etc/nginx/includes/http_headers.conf;

# Désactiver la journalisation d'accès pour le contenu statique
access_log off;
```

Implémentation des en-têtes de sécurité

Ajoutez les en-têtes de sécurité à votre bloc serveur Nginx :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez la directive d'inclusion au-dessus du bloc location de traitement PHP :

```
# Inclure les en-têtes de sécurité
include /etc/nginx/includes/http_headers.conf;
```

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx si le test est réussi
sudo systemctl reload nginx
```

Utilisation des directives Nginx pour protéger WordPress

Nginx peut être configuré pour bloquer l'accès aux fichiers WordPress sensibles et prévenir les vecteurs d'attaque courants.

Création des directives de sécurité WordPress

Créez un fichier de configuration pour les directives de sécurité WordPress :

```
# Créer le fichier de directives de sécurité WordPress
cd /etc/nginx/includes/
sudo vim nginx_security_directives.conf
```

Ajoutez le contenu suivant :

```
# Refuser l'accès aux fichiers WordPress sensibles
location = /wp-config.php { deny all; }
location = /wp-admin/install.php { deny all; }
location ~* /readme\.html$ { deny all; }
location ~* /readme\.txt$ { deny all; }
location ~* /licence\.txt$ { deny all; }
location ~* /license\.txt$ { deny all; }
location ~ ^/wp-admin/includes/ { deny all; }
location ~ ^/wp-includes/[^/]+\.php$ { deny all; }
location ~ ^/wp-includes/js/tinymce/langs/.+\.php$ { deny all; }
location ~ ^/wp-includes/theme-compat/ { deny all; }

# Désactiver l'exécution PHP dans les répertoires potentiellement dangereux
location ~* ^/wp\-content/uploads/.*\.(?:php[1-7]?|pht|phtml?|phps)$ { deny all; }
location ~* ^/wp\-content/plugins/.*\.(?:php[1-7]?|pht|phtml?|phps)$ { deny all; }
location ~* ^/wp\-content/themes/.*\.(?:php[1-7]?|pht|phtml?|phps)$ { deny all; }

# Désactiver l'accès au fichier de surcharge PHP du site
location = /.user.ini { deny all; }

# Filtrer les méthodes de requête suspectes
if ( $request_method ~* ^(TRACE|DELETE|TRACK)$ ) { return 403; }

# Filtrer les chaînes de requête suspectes dans l'URL
set $susquery 0;
if ( $args ~* "\.\./" ) { set $susquery 1; }
if ( $args ~* "\.(bash|git|hg|log|svn|swp|cvs)" ) { set $susquery 1; }
if ( $args ~* "etc/passwd" ) { set $susquery 1; }
if ( $args ~* "boot\.ini" ) { set $susquery 1; }
if ( $args ~* "ftp:" ) { set $susquery 1; }
if ( $args ~* "(<|%3C)script(>|%3E)" ) { set $susquery 1; }
if ( $args ~* "mosConfig_[a-zA-Z_]{1,21}(=|%3D)" ) { set $susquery 1; }
if ( $args ~* "base64_decode\(" ) { set $susquery 1; }
if ( $args ~* "%24&x" ) { set $susquery 1; }
if ( $args ~* "127\.0" ) { set $susquery 1; }
if ( $args ~* "(globals|encode|request|localhost|loopback|request|insert|concat|union|
declare)" ) { set $susquery 1; }
if ( $args ~* "%[01][0-9A-F]" ) { set $susquery 1; }
# Exceptions pour les fonctionnalités WordPress légitimes
```

```
if ( $args ~ "^loggedout=true" ) { set $susquery 0; }
if ( $args ~ "^action=jetpack-sso" ) { set $susquery 0; }
if ( $args ~ "^action=rp" ) { set $susquery 0; }
if ( $http_cookie ~ "wordpress_logged_in_" ) { set $susquery 0; }
if ( $http_referer ~* "^https?://maps\.googleapis\.com/" ) { set $susquery 0; }
if ( $susquery = 1 ) { return 403; }

# Bloquer les injections SQL courantes
set $block_sql_injections 0;
if ($query_string ~ "union.*select.*\(") { set $block_sql_injections 1; }
if ($query_string ~ "union.*all.*select.*") { set $block_sql_injections 1; }
if ($query_string ~ "concat.*\(") { set $block_sql_injections 1; }
if ($block_sql_injections = 1) { return 403; }

# Bloquer les injections de fichiers
set $block_file_injections 0;
if ($query_string ~ "[a-zA-Z0-9_]=http://") { set $block_file_injections 1; }
if ($query_string ~ "[a-zA-Z0-9_]=(\.\.//?)+") { set $block_file_injections 1; }
if ($query_string ~ "[a-zA-Z0-9_]=/([a-z0-9_.]//?)+") { set $block_file_injections 1; }
if ($block_file_injections = 1) { return 403; }

# Bloquer les exploits courants
set $block_common_exploits 0;
if ($query_string ~ "(<|%3C).*script.*(>|%3E)") { set $block_common_exploits 1; }
if ($query_string ~ "GLOBALS(=|\[|\%[0-9A-Z]{0,2})") { set $block_common_exploits 1; }
if ($query_string ~ "_REQUEST(=|\[|\%[0-9A-Z]{0,2})") { set $block_common_exploits 1; }
if ($query_string ~ "proc/self/environ") { set $block_common_exploits 1; }
if ($query_string ~ "mosConfig_[a-zA-Z_]{1,21}(=|\%3D)") { set $block_common_exploits 1; }
if ($query_string ~ "base64_(en|de)code\(.*\)") { set $block_common_exploits 1; }
if ($block_common_exploits = 1) { return 403; }

# Bloquer le spam
set $block_spam 0;
if ($query_string ~ "\b(ultram|unicauca|valium|viagra|vicodin|xanax|ypxaieo)\b") { set
$block_spam 1; }
if ($query_string ~ "\b(erections|hoodia|huronriveracres|impotence|levitra|libido)\b") { set
$block_spam 1; }
if ($query_string ~ "\b(ambien|blue\spill|cialis|cocaine|ejaculation|erectile)\b") { set
$block_spam 1; }
if ($query_string ~ "\b(lipitor|phentermin|pro[sz]ac|sandyauer|tramadol|troyhamby)\b") { set
$block_spam 1; }
if ($block_spam = 1) { return 403; }

# Bloquer les agents utilisateurs malveillants
set $block_user_agents 0;
if ($http_user_agent ~ "Indy Library") { set $block_user_agents 1; }
if ($http_user_agent ~ "libwww-perl") { set $block_user_agents 1; }
if ($http_user_agent ~ "GetRight") { set $block_user_agents 1; }
if ($http_user_agent ~ "GetWeb!") { set $block_user_agents 1; }
if ($http_user_agent ~ "Go!Zilla") { set $block_user_agents 1; }
if ($http_user_agent ~ "Download Demon") { set $block_user_agents 1; }
```

```
if ($http_user_agent ~ "Go-Ahead-Got-It") { set $block_user_agents 1; }
if ($http_user_agent ~ "TurnitinBot") { set $block_user_agents 1; }
if ($http_user_agent ~ "GrabNet") { set $block_user_agents 1; }
# Bloquer les outils de scan de sécurité courants
if ($http_user_agent ~ "dirbuster") { set $block_user_agents 1; }
if ($http_user_agent ~ "nikto") { set $block_user_agents 1; }
if ($http_user_agent ~ "sqlmap") { set $block_user_agents 1; }
if ($http_user_agent ~ "fimap") { set $block_user_agents 1; }
if ($http_user_agent ~ "nessus") { set $block_user_agents 1; }
if ($http_user_agent ~ "whatweb") { set $block_user_agents 1; }
if ($http_user_agent ~ "Openvas") { set $block_user_agents 1; }
if ($http_user_agent ~ "jbrofuzz") { set $block_user_agents 1; }
if ($http_user_agent ~ "libwhisker") { set $block_user_agents 1; }
if ($http_user_agent ~ "webshag") { set $block_user_agents 1; }
if ($http_user_agent ~ "Acunetix") { set $block_user_agents 1; }
if ($block_user_agents = 1) { return 403; }
```

Implémentation des directives de sécurité WordPress

Ajoutez les directives de sécurité à votre bloc serveur Nginx :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez la directive d'inclusion au-dessus du bloc location de traitement PHP :

```
# Inclure les directives de sécurité WordPress
include /etc/nginx/includes/nginx_security_directives.conf;
```

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx si le test est réussi
sudo systemctl reload nginx
```

Autoriser l'exécution PHP pour des fichiers d'extensions spécifiques

Certaines extensions nécessitent l'exécution PHP dans le répertoire des extensions. Puisque nous avons bloqué l'exécution PHP dans ce répertoire pour des raisons de sécurité, nous devons l'autoriser sélectivement pour les fichiers d'extensions légitimes.

Test du blocage de l'exécution PHP

Vérifiez d'abord que l'exécution PHP est bloquée dans le répertoire des extensions :

```
# Créer un fichier PHP de test dans le répertoire des extensions
cd /var/www/example.com/
sudo vim public_html/wp-content/plugins/test.php
```

Ajoutez ce contenu au fichier de test :

```
<?php
phpinfo();
?>
```

Lorsque vous essayez d'accéder à ce fichier dans votre navigateur (`https://example.com/wp-content/plugins/test.php`), vous devriez recevoir une erreur 403 Forbidden, confirmant que l'exécution PHP est bloquée.

Autoriser des fichiers d'extensions spécifiques

Pour autoriser l'exécution PHP pour des fichiers d'extensions spécifiques, ajoutez un bloc location pour chaque fichier :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez un bloc location pour chaque fichier d'extension nécessitant l'exécution PHP :

```
# Autoriser l'exécution PHP pour des fichiers d'extensions spécifiques
location = /wp-content/plugins/specific-plugin/file.php {
    allow all;
    include snippets/fastcgi-php.conf;
    fastcgi_param HTTP_HOST $host;
    fastcgi_pass unix:/run/php/php8.3-fpm-example.com.sock;
    include /etc/nginx/includes/fastcgi_optimize.conf;
}
```

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx et redémarrer PHP-FPM
sudo systemctl reload nginx && sudo systemctl restart php8.3-fpm
```

N'oubliez pas de supprimer le fichier de test :

```
# Supprimer le fichier PHP de test
sudo rm /var/www/example.com/public_html/wp-content/plugins/test.php
```

Protection DDoS Nginx et WordPress

Protection contre le hotlinking

Le hotlinking se produit lorsque d'autres sites web créent des liens directs vers vos images ou fichiers, consommant votre bande passante. Empêchez cela avec Nginx :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez ce qui suit à votre bloc serveur :

```
# Empêcher le hotlinking des images
location ~* \.(jpg|jpeg|png|gif|webp|svg)$ {
    valid_referers none blocked server_names *.example.com example.com;
    if ($invalid_referer) {
        return 403;
```

```
    }

    # Inclure la mise en cache du navigateur pour les images
    include /etc/nginx/includes/browser_caching_security_headers.conf;
}
```

Stopper les attaques par force brute - Limitation de débit Nginx

La limitation de débit aide à prévenir les attaques par force brute en limitant le nombre de requêtes qu'un client peut effectuer dans une période donnée.

Configuration de la limitation de débit

Commencez par ajouter la zone de limitation de débit à votre configuration Nginx principale :

```
# Modifier la configuration Nginx principale
cd /etc/nginx/
sudo vim nginx.conf
```

Ajoutez ce qui suit au contexte http :

```
# Zone de limitation de débit pour WordPress
limit_req_zone $binary_remote_addr zone=wp:10m rate=30r/m;
```

Cela crée une zone de 10 Mo nommée "wp" qui limite les clients à 30 requêtes par minute.

Création d'une configuration de limitation de débit

Créez un fichier de configuration pour la limitation de débit du login WordPress et XML-RPC :

```
# Créer le fichier de configuration de limitation de débit
cd /etc/nginx/includes/
sudo vim rate_limiting.conf
```

Ajoutez le contenu suivant :

```
# Limiter le débit de la page de connexion WordPress
location = /wp-login.php {
    limit_req zone=wp burst=20 nodelay;
    limit_req_status 444;
    include snippets/fastcgi-php.conf;
    fastcgi_param HTTP_HOST $host;
    fastcgi_pass unix:/run/php/php8.3-fpm-example.com.sock;
    include /etc/nginx/includes/fastcgi_optimize.conf;
}

# Limiter le débit du XML-RPC WordPress (souvent utilisé dans les attaques par force brute)
location = /xmlrpc.php {
    limit_req zone=wp burst=20 nodelay;
    limit_req_status 444;
    include snippets/fastcgi-php.conf;
    fastcgi_param HTTP_HOST $host;
    fastcgi_pass unix:/run/php/php8.3-fpm-example.com.sock;
    include /etc/nginx/includes/fastcgi_optimize.conf;
}
```

Explication de la limitation de débit

- `burst=20` autorise une rafale de 20 requêtes
- `nodelay` signifie que ces requêtes ne sont pas retardées mais traitées immédiatement
- `limit_req_status 444` renvoie un statut "Connection Closed Without Response", qui est plus efficace que le 503 par défaut

Implémentation de la limitation de débit

Ajoutez la configuration de limitation de débit à votre bloc serveur Nginx :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez la directive d'inclusion :

```
# Inclure la configuration de limitation de débit
include /etc/nginx/includes/rate_limiting.conf;
```

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx si le test est réussi
sudo systemctl reload nginx
```

Interdire les modifications de fichiers

WordPress permet aux administrateurs de modifier les fichiers de thèmes et d'extensions directement depuis le tableau de bord d'administration. Cela peut être un risque de sécurité si un attaquant obtient l'accès à votre compte administrateur.

Désactivation des modifications de fichiers dans wp-config.php

Modifiez votre fichier wp-config.php pour désactiver les modifications de fichiers :

```
# Modifier wp-config.php
sudo vim /var/www/example.com/public_html/wp-config.php
```

Ajoutez la ligne suivante avant le commentaire "That's all, stop editing!" :

```
/* Désactiver les modifications de fichiers */
define('DISALLOW_FILE_MODS', true);
```

Appliquez les modifications :

```
# Recharger PHP-FPM
sudo systemctl reload php8.3-fpm
```

Restriction des privilèges utilisateur de la base de données

L'utilisateur de la base de données WordPress ne devrait avoir que les privilèges nécessaires à son fonctionnement, et non un accès complet à la base de données.

Limitation des privilèges utilisateur de la base de données

Connectez-vous à votre serveur MariaDB et restreignez les privilèges de l'utilisateur de la base de données WordPress :

```
# Se connecter à MariaDB
sudo mysql -u root -p
```

Exécutez ces commandes SQL :

```
-- Révoquer tous les privilèges
REVOKE ALL PRIVILEGES ON wordpress_db.* FROM 'wordpress_user'@'localhost';

-- Accorder uniquement les privilèges nécessaires
GRANT SELECT, INSERT, UPDATE, DELETE ON wordpress_db.* TO 'wordpress_user'@'localhost';

-- Appliquer les modifications
FLUSH PRIVILEGES;
```

```
-- Exemple : Accorder des privilèges supplémentaires pour des extensions spécifiques
GRANT CREATE, ALTER, INDEX ON wordpress_db.* TO 'wordpress_user'@'localhost';
FLUSH PRIVILEGES;
```

Privilèges supplémentaires

Certaines extensions peuvent nécessiter des privilèges supplémentaires. Si nécessaire, vous pouvez les accorder sélectivement :

Quittez la console MariaDB :

```
EXIT;
```

Durcissement de l'API REST WP

L'API REST WordPress peut être une cible pour les attaquants. Par défaut, elle est accessible à tous, mais nous pouvons restreindre l'accès aux utilisateurs authentifiés uniquement.

Restriction de l'accès à l'API REST

Ajoutez le code suivant au fichier functions.php de votre thème ou à une extension personnalisée :

```
/**

 * Restreindre l'API REST aux utilisateurs authentifiés
 */
function restrict_rest_api_to_authenticated_users($access) {
    // Si non authentifié, bloquer l'accès
    if (!is_user_logged_in()) {
        return new
        WP_Error('rest_api_restricted', 'REST API access is restricted to authenticated users.', array('st
    }

    // Sinon, autoriser l'accès
    return $access;
```

```
}
add_filter('rest_authentication_errors', 'restrict_rest_api_to_authenticated_users');
```

Restriction de l'API REST avec Nginx

Pour une couche de sécurité supplémentaire, vous pouvez également restreindre l'accès à l'API REST au niveau Nginx :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez le bloc location suivant :

```
# Restreindre l'API REST WP
location /wp-json/ {
    # Autoriser l'accès uniquement si connecté ou depuis des IP spécifiques
    if ($http_cookie !~* "wordpress_logged_in_") {
        # Autoriser des IP spécifiques (remplacez par votre IP)
        allow 192.168.1.1;
        # Bloquer tous les autres
        deny all;
    }

    # Continuer le traitement de la requête
    try_files $uri $uri/ /index.php?$args;
}
```

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx si le test est réussi
sudo systemctl reload nginx
```

Utilisation d'un pare-feu applicatif web

Un pare-feu applicatif web (WAF) fournit une couche de sécurité supplémentaire en filtrant et surveillant le trafic HTTP entre une application web et Internet.

Installation de ModSecurity avec Nginx

ModSecurity est un WAF open source qui peut être intégré à Nginx :

```
# Installer ModSecurity et les dépendances
sudo apt update
sudo apt install -y libmodsecurity3 libmodsecurity-dev nginx-module-security
```

Configuration de ModSecurity

Créez une configuration de base ModSecurity :

```
# Créer le répertoire de configuration ModSecurity
sudo mkdir -p /etc/nginx/modsec
```

```
# Télécharger le jeu de règles OWASP Core Rule Set
sudo git clone https://github.com/coreruleset/coreruleset /etc/nginx/modsec/coreruleset

# Créer le fichier de configuration ModSecurity
sudo vim /etc/nginx/modsec/modsec.conf
```

Ajoutez le contenu suivant :

```
# Configuration de base ModSecurity
SecRuleEngine On
SecRequestBodyAccess On
SecResponseBodyAccess On
SecResponseBodyMimeType text/plain text/html text/xml application/json
SecResponseBodyLimit 1024

# Inclure le jeu de règles OWASP Core Rule Set
Include /etc/nginx/modsec/coreruleset/crs-setup.conf
Include /etc/nginx/modsec/coreruleset/rules/*.conf
```

Activation de ModSecurity dans Nginx

Modifiez votre configuration Nginx pour activer ModSecurity :

```
# Modifier la configuration Nginx principale
sudo vim /etc/nginx/nginx.conf
```

Ajoutez ce qui suit au contexte http :

```
# Charger le module ModSecurity
load_module modules/ngx_http_modsecurity_module.so;
```

Modifiez le bloc serveur de votre site :

```
# Modifier la configuration Nginx de votre site
sudo vim /etc/nginx/sites-available/example.com.conf
```

Ajoutez les directives ModSecurity à votre bloc serveur :

```
# Activer ModSecurity
modsecurity on;
modsecurity_rules_file /etc/nginx/modsec/modsec.conf;
```

Appliquez les modifications :

```
# Tester la configuration Nginx
sudo nginx -t

# Recharger Nginx si le test est réussi
sudo systemctl reload nginx
```

Conclusion

En implémentant ces mesures de sécurité, vous avez considérablement durci votre installation WordPress contre les menaces courantes. N'oubliez pas que la sécurité est un processus continu et que vous devez :

1. Mettre régulièrement à jour le noyau WordPress, les thèmes et les extensions
2. Surveiller vos journaux pour détecter les activités suspectes
3. Effectuer des audits de sécurité réguliers
4. Conserver des sauvegardes de votre site et de votre base de données
5. Rester informé des nouvelles menaces de sécurité et des bonnes pratiques

Sécurité en profondeur

Les installations WordPress les plus sécurisées utilisent plusieurs couches de protection. Aucune mesure de sécurité n'est infaillible seule, mais ensemble, elles créent une défense robuste contre les attaquants.

8.15 16. Optimisation des performances WordPress

Introduction

Après avoir sécurisé votre installation WordPress, l'étape suivante est l'optimisation de ses performances. Cette section couvre les techniques d'optimisation essentielles qui permettront de :

1. Réduire l'utilisation des ressources serveur
2. Améliorer les temps de chargement des pages
3. Gérer plus de visiteurs simultanés
4. Améliorer l'expérience utilisateur globale

Impact sur les performances

Un site WordPress bien optimisé peut se charger 2 à 5 fois plus vite qu'une installation par défaut, améliorant significativement l'expérience utilisateur et le référencement SEO.

Politique de révisions des articles

WordPress enregistre automatiquement des révisions de vos articles et pages au fur et à mesure que vous les modifiez. Bien qu'utile pour la récupération de contenu, des révisions excessives peuvent alourdir votre base de données.

Limitation des révisions des articles

Vous pouvez contrôler les révisions des articles en ajoutant une directive à votre fichier `wp-config.php` :

```
# Naviguer vers votre installation WordPress
cd /var/www/example.com/
sudo vim public_html/wp-config.php
```

Ajoutez l'une de ces lignes à votre configuration :

```
// Désactiver complètement les révisions des articles
define('WP_POST_REVISIONS', false);

// Ou limiter à un nombre spécifique (recommandé : 3-5)
define('WP_POST_REVISIONS', 3);
```

Après avoir effectué les modifications, redémarrez PHP-FPM pour vider le opcache :

```
sudo systemctl restart php8.3-fpm
```

Définition de la limite de mémoire maximale

WordPress peut être gourmand en mémoire, surtout avec des thèmes et des extensions complexes. Augmenter la limite de mémoire peut prévenir les erreurs "out of memory".

Configuration du pool PHP-FPM

Commencez par augmenter la limite de mémoire dans la configuration de votre pool PHP-FPM :

```
# Modifier la configuration du pool PHP-FPM de votre site
sudo vim /etc/php/8.3/fpm/pool.d/example.com.conf
```

Trouvez et modifiez la ligne de limite de mémoire :

```
; Augmenter de 32M par défaut à 256M
;php_admin_value[memory_limit] = 32M
php_admin_value[memory_limit] = 256M
```

Configuration WordPress

Définissez également la limite de mémoire dans WordPress lui-même :

```
# Modifier votre configuration WordPress
sudo vim /var/www/example.com/public_html/wp-config.php
```

Ajoutez cette ligne à votre configuration :

```
/** Memory Limit for WordPress */
define('WP_MEMORY_LIMIT', '256M');
```

Remplacement du WP CRON par un VRAI CRON

WordPress utilise un système cron virtuel (WP-Cron) qui se déclenche lors du chargement des pages, ce qui peut être inefficace et peu fiable. Le remplacer par une vraie tâche cron système améliore les performances et la fiabilité.

Désactivation de WP-Cron

Commencez par désactiver le WP-Cron intégré :

```
# Modifier votre configuration WordPress
sudo vim /var/www/example.com/public_html/wp-config.php
```

Ajoutez cette ligne à votre configuration :

```
/** Désactiver le système cron intégré */
define('DISABLE_WP_CRON', true);
```

Redémarrez PHP-FPM pour appliquer les modifications :

```
sudo systemctl restart php8.3-fpm
```

Mise en place d'une vraie tâche cron

Maintenant, configurez une tâche cron système pour déclencher les événements cron de WordPress :

```
# Modifier votre crontab
crontab -e
```

Ajoutez cette ligne pour exécuter le cron WordPress toutes les 15 minutes :

```
# Exécuter le cron WordPress toutes les 15 minutes
*/15 * * * * wget -q -O - https://example.com/wp-cron.php?doing_wp_cron >/dev/null 2>&1
```

Nom de domaine

Remplacez `example.com` par votre nom de domaine réel.

Introduction à la mise en cache

La mise en cache est l'un des moyens les plus efficaces pour améliorer les performances de WordPress. Elle réduit les requêtes à la base de données et l'exécution PHP en stockant du contenu pré-généré.

Pour les besoins de mise en cache, les sites WordPress peuvent être catégorisés comme :

- **Sites statiques** : Le contenu change rarement et est identique pour tous les utilisateurs (blogs, sites vitrine)
- **Sites dynamiques** : Le contenu change fréquemment ou est personnalisé pour chaque utilisateur (e-commerce, sites avec abonnement)

Types de mise en cache

Il existe plusieurs niveaux de mise en cache qui peuvent être implémentés :

1. **Mise en cache de pages** : Stocke des pages HTML complètes
2. **Mise en cache d'objets** : Stocke les résultats des requêtes à la base de données
3. **Mise en cache du navigateur** : Indique aux navigateurs de stocker les ressources statiques
4. **Mise en cache CDN** : Distribue le contenu mis en cache à travers des serveurs mondiaux

Implémentation de PHP OPcache

PHP OPcache améliore les performances en stockant le bytecode précompilé des scripts en mémoire, éliminant le besoin pour PHP de charger et analyser les scripts à chaque requête.

Configuration d'OPcache

Modifiez la configuration du pool PHP-FPM de votre site :

```
# Naviguer vers le répertoire des pools PHP-FPM
cd /etc/php/8.3/fpm/pool.d/
ls
sudo vim example.com.conf
```

Ajoutez ces paramètres OPcache à votre configuration :

```
; CONFIGURATION OPCACHE - SERVEUR DE DEVELOPPEMENT
php_admin_flag[opcache.enabled] = 1
php_admin_value[opcache.memory_consumption] = 256
php_admin_value[opcache.interned_strings_buffer] = 32
php_admin_value[opcache.max_accelerated_files] = 20000
php_admin_flag[opcache.validate_timestamps] = 1
php_admin_value[opcache.revalidate_freq] = 2
php_admin_flag[opcache.validate_permission] = 1
```

Pour les serveurs de production, utilisez plutôt ces paramètres :

```
; CONFIGURATION OPCACHE - SERVEUR DE PRODUCTION
php_admin_flag[opcache.enabled] = 1
php_admin_value[opcache.memory_consumption] = 256
php_admin_value[opcache.interned_strings_buffer] = 32
php_admin_value[opcache.max_accelerated_files] = 20000
php_admin_flag[opcache.validate_timestamps] = 0
php_admin_flag[opcache.validate_permission] = 1
```

Production vs Développement

La différence clé est `validate_timestamps` : En développement, défini à 1 pour voir les modifications immédiatement. En production, défini à 0 pour de meilleures performances.

Appliquez les modifications :

```
sudo systemctl reload php8.3-fpm
```

Détermination de la valeur optimale de max_accelerated_files

Pour déterminer la valeur optimale de `max_accelerated_files`, comptez les fichiers PHP dans votre installation :

```
cd /var/www/
sudo find . -type f -print | grep php | wc -l
```

Définissez `max_accelerated_files` à au moins 20 % de plus que ce nombre.

Mise en cache d'un site WordPress "statique"

Pour les sites dont le contenu ne change pas fréquemment et est identique pour tous les utilisateurs, vous pouvez implémenter des stratégies de mise en cache agressives. Choisissez l'une des méthodes suivantes en fonction de vos préférences et exigences.

Utilisation du cache FastCGI

Le cache FastCGI est une solution de mise en cache au niveau serveur intégrée à Nginx. Elle est très efficace et ne nécessite pas d'extensions WordPress.

Configuration de Nginx pour le cache FastCGI

Commencez par modifier la configuration principale de Nginx :

```
cd /etc/nginx
sudo vim nginx.conf
```

Ajoutez les directives suivantes au contexte http (juste au-dessus du commentaire "Virtual Host Configs") :

```
### CACHE FASTCGI
# Directive fastcgi_cache_path - PATH et NAME doivent être uniques pour chaque site
# Ajoutez un nouveau fastcgi_cache_path pour chaque site avec un nouveau nom keys_zone
fastcgi_cache_path /var/run/SITE levels=1:2 keys_zone=NAME:100m inactive=60m;
# Appliqué à tous les sites
fastcgi_cache_key "$scheme$request_method$host$request_uri";
fastcgi_cache_use_stale error timeout invalid_header http_500;
fastcgi_ignore_headers Cache-Control Expires Set-Cookie;
```

Variables de configuration

Remplacez `SITE` par votre nom de domaine (par exemple, `example_com`) et `NAME` par un identifiant unique pour votre zone de cache.

Création des règles d'exclusion du cache

Créez un fichier pour les règles d'exclusion du cache :

```
cd /etc/nginx/includes/
sudo vim fastcgi_cache_excludes.conf
```

Ajoutez ces règles pour exclure le contenu dynamique de la mise en cache :

```
# FICHIER D'INCLUSION DES EXCLUSIONS DU CACHE NGINX
set $skip_cache 0;
# Les requêtes POST et les URL avec une chaîne de requête doivent toujours être transmises à
# PHP
if ($request_method = POST) {
    set $skip_cache 1;
}
if ($query_string != "") {
    set $skip_cache 1;
}
# Ne pas mettre en cache les URI contenant les segments suivants
if ($request_uri ~* "/wp-admin/|/xmlrpc.php|wp-.*.php|/feed/|index.php|sitemap(_index)?.xml") {
    set $skip_cache 1;
}
# Ne pas utiliser le cache pour les utilisateurs connectés ou les commentateurs récents
if ($http_cookie ~* "comment_author|wordpress_[a-f0-9]+|wp-postpass|wordpress_no_cache|
wordpress_logged_in") {
    set $skip_cache 1;
}
```

Mise à jour de la configuration Nginx de votre site

Modifiez la configuration Nginx de votre site :

```
cd /etc/nginx/sites-available
sudo vim example.com.conf
```

Ajoutez ces lignes à votre bloc serveur :

```
# Inclure les règles d'exclusion du cache
include /etc/nginx/includes/fastcgi_cache_excludes.conf;

# Ajouter l'en-tête de statut du cache
add_header X-FastCGI-Cache $upstream_cache_status;

# Ajouter ces directives au bloc location PHP
location ~ \.php$ {
    # Directives de traitement PHP existantes...

    # Directives du cache FastCGI
    fastcgi_cache_bypass $skip_cache;
    fastcgi_no_cache $skip_cache;
    fastcgi_cache NAME;
    fastcgi_cache_valid 60m;
}
```

Ajout de la capacité de purge du cache

Ajoutez un bloc location pour purger le cache :

```
# Ajouter ceci à votre bloc serveur
location ~ /purge(/.*) {
    fastcgi_cache_purge NAME "$scheme$request_method$host$1";
}
```

Appliquez les modifications :

```
sudo nginx -t
sudo systemctl reload nginx
```

Test du cache

Testez si la mise en cache fonctionne :

```
# La première requête devrait afficher MISS
curl -I https://example.com

# La deuxième requête devrait afficher HIT
curl -I https://example.com
```

Recherchez l'en-tête `X-FastCGI-Cache` dans la réponse.

Suppression du cache FastCGI (si nécessaire)

Si vous devez supprimer le cache FastCGI :

```
cd /etc/nginx/sites-available
sudo vim example.com.conf
```

Commentez ou supprimez ces lignes :

```
# Supprimer ou commenter ces lignes
# include /etc/nginx/includes/fastcgi_cache_excludes.conf;
# add_header X-FastCGI-Cache $upstream_cache_status;

# Dans le bloc location PHP, supprimer ou commenter :
# fastcgi_cache_bypass $skip_cache;
# fastcgi_no_cache $skip_cache;
# fastcgi_cache NAME;
# fastcgi_cache_valid 60m;

# Supprimer le bloc location de purge
# location ~ /purge(/.*) {
#     fastcgi_cache_purge NAME "$scheme$request_method$host$1";
# }
```

Appliquez les modifications :

```
sudo nginx -t
sudo systemctl reload nginx
sudo systemctl restart php8.3-fpm
```

Utilisation de WP Super Cache

WP Super Cache est une extension populaire de mise en cache WordPress qui génère des fichiers HTML statiques à partir de votre contenu WordPress dynamique.

Installation de WP Super Cache

1. Connectez-vous à votre tableau de bord d'administration WordPress
2. Allez dans Extensions > Ajouter
3. Recherchez "WP Super Cache"
4. Cliquez sur "Installer maintenant" puis "Activer"

Configuration de Nginx pour WP Super Cache

Créez un fichier de configuration pour WP Super Cache :

```
cd /etc/nginx/includes/
sudo vim wp_super_cache_excludes.conf
```

Ajoutez ces règles :

```
# Règles d'exclusion du cache NGINX pour WP Super Cache
set $cache_uri $request_uri;

# Les requêtes POST et les URL avec une chaîne de requête doivent toujours être transmises à
# PHP
if ($request_method = POST) {
    set $cache_uri 'null cache';
}

if ($query_string != "") {
    set $cache_uri 'null cache';
}

# Ne pas mettre en cache les URI contenant les segments suivants
if ($request_uri ~* "(/wp-admin/|/xmlrpc.php|/wp-(app|cron|login|register|mail).php|wp-.*.php|
/feed/|index.php|wp-comments-popup.php|wp-links-opml.php|wp-locations.php|sitemap(_index)?.xml|
[a-z0-9_-]+-sitemap([0-9]+)?.xml)") {
    set $cache_uri 'null cache';
}

# Ne pas utiliser le cache pour les utilisateurs connectés ou les commentateurs récents
if ($http_cookie ~* "comment_author|wordpress_[a-f0-9]+|wp-postpass|wordpress_logged_in") {
    set $cache_uri 'null cache';
}

# Utiliser le fichier en cache ou le fichier réel s'il existe, sinon transmettre la requête à
# WordPress
location / {
    try_files /wp-content/cache/supercache/$http_host/$cache_uri/index-https.html $uri $uri/
    /index.php?$args;
}
```

Mettez à jour la configuration Nginx de votre site :

```
cd /etc/nginx/sites-available/
sudo vim example.com.conf
```

Commentez le bloc location par défaut et incluez la configuration WP Super Cache :

```
# Commenter le bloc location par défaut
# location / {
#     try_files $uri $uri/ /index.php$is_args$args;
# }

# Inclure la configuration WP Super Cache
include /etc/nginx/includes/wp_super_cache_excludes.conf;
```

Appliquez les modifications :

```
sudo nginx -t
sudo systemctl reload nginx
```

Configuration de l'extension WP Super Cache

1. Allez dans Réglages > WP Super Cache dans votre administration WordPress
2. Activez la mise en cache en sélectionnant "Caching On"
3. Sous l'onglet Avancé :
   - Sélectionnez "Use mod_rewrite to serve cache files"
   - Cochez "Compress pages"
   - Cochez "Don't cache pages for known users"
   - Cochez "Cache rebuild"
4. Cliquez sur "Update Status"

Suppression de WP Super Cache (si nécessaire)

Si vous devez supprimer WP Super Cache :

1. Désactivez et désinstallez l'extension depuis l'administration WordPress
2. Restaurez votre configuration Nginx :

```
cd /etc/nginx/sites-available/
sudo vim example.com.conf
```

Supprimez l'inclusion de WP Super Cache et restaurez le bloc location par défaut :

```
# Supprimer cette ligne
# include /etc/nginx/includes/wp_super_cache_excludes.conf;

# Décommenter le bloc location par défaut
location / {
    try_files $uri $uri/ /index.php$is_args$args;
}
```

Appliquez les modifications :

```
sudo nginx -t
sudo systemctl reload nginx
sudo systemctl reload php8.3-fpm
```

Utilisation de W3 Total Cache (W3TC)

W3 Total Cache est une extension de mise en cache complète avec des options de configuration étendues pour diverses méthodes de mise en cache.

Installation de W3 Total Cache

1. Connectez-vous à votre tableau de bord d'administration WordPress
2. Allez dans Extensions > Ajouter
3. Recherchez "W3 Total Cache"
4. Cliquez sur "Installer maintenant" puis "Activer"

Préparation pour l'intégration Nginx de W3TC

Créez un fichier pour que W3TC puisse écrire ses règles Nginx :

```
cd /var/www/example.com/public_html/
sudo touch nginx.conf
sudo chown username:username nginx.conf
sudo chmod 660 nginx.conf
```

Nom d'utilisateur

Remplacez `username` par votre nom d'utilisateur système réel.

Ajoutez une directive de sécurité pour empêcher l'accès direct au fichier nginx.conf :

```
cd /etc/nginx/includes/
sudo vim nginx_security_directives.conf
```

Ajoutez cette ligne :

```
location = /nginx.conf { deny all; }
```

Création des règles d'exclusion du cache W3TC

Créez un fichier de configuration pour les exclusions de cache W3TC :

```
cd /etc/nginx/includes/
sudo vim w3tc_cache_excludes.conf
```

Ajoutez ces règles :

```
# ---------------------
# FICHIER D'EXCLUSIONS W3 TOTAL CACHE
# ---------------------
set $cache_uri $request_uri;

# Les requêtes POST et les URL avec une chaîne de requête doivent toujours être transmises à
# PHP
if ($request_method = POST) {
    set $cache_uri 'null cache';
}
if ($query_string != "") {
    set $cache_uri 'null cache';
}
# Ne pas mettre en cache les URI contenant les segments suivants
if ($request_uri ~* "(/wp-admin/|/xmlrpc.php|/wp-(app|cron|login|register|mail).php|wp-.*.php|
/feed/|index.php|wp-comments-popup.php|wp-links-opml.php|wp-locations.php|sitemap(_index)?.xml|
[a-z0-9_-]+-sitemap([0-9]+)?.xml)") {
```

```
    set $cache_uri 'null cache';
}
# Ne pas utiliser le cache pour les utilisateurs connectés ou les commentateurs récents
if ($http_cookie ~* "comment_author|wordpress_[a-f0-9]+|wp-postpass|wordpress_logged_in") {
    set $cache_uri 'null cache';
}
# Utiliser le fichier en cache ou le fichier réel s'il existe, sinon transmettre la requête à
# WordPress
location / {
    try_files /wp-content/w3tc/pgcache/$cache_uri/_index.html $uri $uri/ /index.php?$args;
}
```

Mise à jour de la configuration Nginx de votre site

Modifiez la configuration Nginx de votre site :

```
cd /etc/nginx/sites-available/
sudo vim example.com.conf
```

Commentez le bloc location par défaut et incluez les configurations W3TC :

```
# Commenter le bloc location par défaut
#location / {
#    try_files $uri $uri/ /index.php$is_args$args;
#}

# Inclure les configurations W3TC
include /etc/nginx/includes/w3tc_cache_excludes.conf;
include /var/www/example.com/public_html/nginx.conf;
```

Installez l'extension PHP Tidy requise par W3TC :

```
sudo apt update
sudo apt install php8.3-tidy
```

Appliquez les modifications :

```
sudo nginx -t
sudo systemctl reload nginx
sudo systemctl reload php8.3-fpm
```

Configuration de l'extension W3 Total Cache

1. Allez dans Performance > General Settings dans votre administration WordPress
2. Activez Page Cache et définissez la méthode à "Disk: Enhanced"
3. Activez Browser Cache
4. Activez Minify (facultatif)
5. Sous Performance > Page Cache :
   - Cochez "Cache SSL"
   - Cochez "Don't cache pages for logged in users"
6. Enregistrez tous les paramètres

Suppression de W3 Total Cache (si nécessaire)

Si vous devez supprimer W3 Total Cache :

1. Désactivez et désinstallez l'extension depuis l'administration WordPress
2. Restaurez votre configuration Nginx :

```
cd /etc/nginx/sites-available/
sudo vim example.com.conf
```

Supprimez les inclusions W3TC et restaurez le bloc location par défaut :

```
# Restaurer le bloc location par défaut
location / {
    try_files $uri $uri/ /index.php$is_args$args;
}

# Supprimer ces lignes
# include /etc/nginx/includes/w3tc_cache_excludes.conf;
# include /var/www/example.com/public_html/nginx.conf;
```

Supprimez les fichiers W3TC :

```
cd /var/www/example.com/public_html/wp-content/
sudo rm -rf cache/ w3tc-config/

cd /var/www/example.com/public_html/
sudo rm nginx.conf
```

Appliquez les modifications :

```
sudo systemctl reload php8.3-fpm
sudo systemctl reload nginx
```

Création d'un cache d'objets persistant avec Redis

La mise en cache d'objets stocke les résultats des requêtes à la base de données en mémoire, réduisant significativement la charge de la base de données. Redis est un excellent choix pour la mise en cache d'objets grâce à sa rapidité et ses capacités de persistance.

Installation de Redis

Installez le serveur Redis et l'extension PHP Redis :

```
sudo apt update
sudo apt install redis-server php8.3-redis
```

Vérifiez que Redis fonctionne :

```
sudo systemctl status redis-server
```

Vérifiez le journal Redis pour les avertissements ou erreurs :

```
sudo cat /var/log/redis/redis-server.log
```

Optimisation de la configuration Redis

Correction de l'avertissement de surengagement mémoire

Si vous voyez cet avertissement dans les journaux Redis :

```
WARNING overcommit_memory is set to 0! Background save may fail under low memory condition.
```

Corrigez-le en créant un fichier de configuration sysctl :

```
cd /etc/sysctl.d/
sudo vim 11-redis.conf
```

Ajoutez cette ligne :

```
vm.overcommit_memory = 1
```

Définition des limites de mémoire

Configurez les limites de mémoire Redis :

```
sudo vim /etc/redis/redis.conf
```

Trouvez et modifiez ces paramètres :

```
# Définir la mémoire maximale à utiliser (ajustez selon les ressources de votre serveur)
maxmemory 256mb

# Définir la politique d'éviction (LRU = Least Recently Used)
maxmemory-policy allkeys-lru
```

Appliquez les modifications :

```
sudo systemctl restart redis-server
```

Allocation de mémoire

Allouez environ 10 à 20 % de la RAM de votre serveur à Redis, mais assurez-vous de laisser suffisamment pour les autres processus.

Configuration de WordPress pour la mise en cache d'objets Redis

Installation d'une extension de cache d'objets Redis

1. Connectez-vous à votre tableau de bord d'administration WordPress
2. Allez dans Extensions > Ajouter
3. Recherchez "Redis Object Cache"
4. Installez et activez l'extension de Till Krüss

Ajout de la configuration Redis à WordPress

Modifiez votre configuration WordPress :

```
cd /var/www/example.com/public_html/
sudo vim wp-config.php
```

Ajoutez ces lignes :

```
/** Redis Object Cache Settings */
define('WP_CACHE_KEY_SALT', 'example.com');
```

Nom de domaine

Remplacez `example.com` par votre nom de domaine réel pour garantir des clés de cache uniques.

Considérations WooCommerce

Si vous utilisez WooCommerce, vous avez besoin d'une configuration spéciale pour empêcher la mise en cache des données de session :

```
/** EMPECHER LA MISE EN CACHE REDIS DES DONNEES DE SESSION WOO */
define('WP_REDIS_IGNORED_GROUPS', 'wc_session');
```

Excluez également ces pages de la mise en cache de pages :

- `/cart/`
- `/my-account/`
- `/checkout/`

Et excluez ces cookies de la génération de clés de cache :

- `woocommerce_cart_hash`
- `woocommerce_items_in_cart`
- `wp_woocommerce_session_`
- `woocommerce_recently_viewed`
- `store_notice[notice id]`
- `_wc_session_`

Mise en cache d'un site WordPress "dynamique"

Pour les sites avec du contenu personnalisé ou des changements fréquents (e-commerce, sites avec abonnement), vous avez besoin d'une approche de mise en cache différente qui combine la mise en cache de pages avec la mise en cache d'objets.

Introduction à la mise en cache de sites dynamiques

Les sites dynamiques nécessitent : 1. La mise en cache d'objets (Redis) pour réduire la charge de la base de données 2. Une mise en cache de pages sélective excluant le contenu personnalisé 3. La mise en cache du navigateur pour les ressources statiques

Combinaison WP Super Cache et Redis

Cette approche utilise WP Super Cache pour la mise en cache de pages avec des exclusions soigneuses et Redis pour la mise en cache d'objets.

1. Suivez les étapes d'installation de Redis de la section **Création d'un cache d'objets persistant avec Redis**
2. Suivez les étapes d'installation de WP Super Cache de la section **Utilisation de WP Super Cache**
3. Dans les paramètres WP Super Cache :
   - Activez "Don't cache pages for known users"
   - Ajoutez des exclusions personnalisées pour les pages dynamiques

Combinaison W3TC et Redis

Cette approche utilise W3 Total Cache avec Redis comme backend de cache d'objets.

1. Suivez les étapes d'installation de Redis de la section **Création d'un cache d'objets persistant avec Redis**
2. Suivez les étapes d'installation de W3TC de la section **Utilisation de W3 Total Cache (W3TC)**
3. Dans W3TC Performance > General Settings :
   - Définissez la méthode de cache d'objets à "Redis"
   - Configurez l'hôte Redis (généralement "localhost") et le port (6379)
4. Dans les paramètres de cache de pages :
   - Activez "Don't cache pages for logged in users"
   - Ajoutez des exclusions personnalisées pour les pages dynamiques

Réglage de PHP-FPM

Une configuration correcte de PHP-FPM est cruciale pour des performances WordPress optimales.

Analyse de l'utilisation des ressources PHP-FPM

Commencez par identifier les utilisateurs de vos pools PHP-FPM :

```
# Afficher tous les noms d'utilisateurs des pools PHP-FPM
grep -E '^\s*user\s*=' /etc/php/8.3/fpm/pool.d/*.conf | awk -F= '{print $2}' | xargs | tr ' '
'\n' | sort -u
```

Calculez l'utilisation mémoire moyenne par processus PHP-FPM :

```
# Remplacez POOL_USER par votre nom d'utilisateur de pool réel
ps -C php-fpm --user POOL_USER -o rss= | awk '{ sum += $1; count++ } END { if (count > 0)
printf ("%d%s\n", sum/NR/1024,"M") }'
```

Exemple :

```
ps -C php-fpm --user wordpress_user -o rss= | awk '{ sum += $1; count++ } END { if (count > 0)
printf ("%d%s\n", sum/NR/1024,"M") }'
```

Configuration du gestionnaire de processus OnDemand

Pour la plupart des sites WordPress, le gestionnaire de processus "ondemand" est optimal :

```
cd /etc/php/8.3/fpm/pool.d/
sudo vim example.com.conf
```

Mettez à jour les paramètres du gestionnaire de processus :

```
; Utiliser le gestionnaire de processus ondemand
pm = ondemand

; Définir max_children selon : (RAM totale - RAM réservée) / Taille moyenne du processus
; Exemple : (2048Mo - 512Mo) / 50Mo = 30
pm.max_children = 30

; Terminer les processus inactifs après 10 secondes
pm.process_idle_timeout = 10s

; Redémarrer les processus après avoir traité 500 requêtes
pm.max_requests = 500
```

Appliquez les modifications :

```
sudo systemctl reload php8.3-fpm
```

Surveillance des limites PHP-FPM

Vérifiez si vos processus PHP-FPM atteignent les limites :

```
sudo grep max_children /var/log/php8.3-fpm.log
```

Si vous voyez cet avertissement, augmentez votre paramètre `pm.max_children` :

```
WARNING: [pool www] server reached max_children setting (25), consider raising it
```

Intégration Cloudflare

Cloudflare est un réseau de diffusion de contenu (CDN) et un service de sécurité qui peut améliorer significativement les performances et la sécurité de WordPress.

Avantages de Cloudflare

1. **CDN mondial** : Sert le contenu mis en cache depuis des centres de données dans le monde entier
2. **Protection DDoS** : Atténue les attaques par déni de service distribué
3. **Pare-feu applicatif web** : Bloque les vecteurs d'attaque courants
4. **SSL/TLS** : Fournit des certificats SSL gratuits et des connexions TLS optimisées
5. **Optimisation d'images** : Optimise automatiquement les images pour un chargement plus rapide

Configuration de Nginx pour Cloudflare

Pour garantir une journalisation correcte des adresses IP des visiteurs lors de l'utilisation de Cloudflare, vous devez configurer Nginx pour reconnaître les adresses IP de Cloudflare :

```
cd /etc/nginx/includes
sudo vim cloudflare_ip_list.conf
```

Ajoutez les plages IP de Cloudflare :

```
# Plages IP Cloudflare
# Dernière mise à jour OCT 2022 - Vérifiez https://www.cloudflare.com/ips pour les mises à jour
set_real_ip_from 173.245.48.0/20;
set_real_ip_from 103.21.244.0/22;
set_real_ip_from 103.22.200.0/22;
set_real_ip_from 103.31.4.0/22;
set_real_ip_from 141.101.64.0/18;
set_real_ip_from 108.162.192.0/18;
set_real_ip_from 190.93.240.0/20;
set_real_ip_from 188.114.96.0/20;
set_real_ip_from 197.234.240.0/22;
set_real_ip_from 198.41.128.0/17;
set_real_ip_from 162.158.0.0/15;
set_real_ip_from 104.16.0.0/13;
set_real_ip_from 104.24.0.0/14;
set_real_ip_from 172.64.0.0/13;
set_real_ip_from 131.0.72.0/22;
set_real_ip_from 2400:cb00::/32;
set_real_ip_from 2606:4700::/32;
set_real_ip_from 2803:f800::/32;
set_real_ip_from 2405:b500::/32;
set_real_ip_from 2405:8100::/32;
set_real_ip_from 2a06:98c0::/29;
set_real_ip_from 2c0f:f248::/32;
real_ip_header CF-Connecting-IP;
```

Incluez ce fichier dans la configuration Nginx de votre site :

```
cd /etc/nginx/sites-available/
sudo vim example.com.conf
```

Ajoutez cette ligne à votre bloc serveur :

```
include /etc/nginx/includes/cloudflare_ip_list.conf;
```

Appliquez les modifications :

```
sudo nginx -t
sudo systemctl reload nginx
```

Paramètres Cloudflare recommandés pour WordPress

Après vous être inscrit sur Cloudflare et avoir pointé votre domaine vers leurs serveurs de noms :

1. **SSL/TLS** : Définissez à “Full” ou “Full (Strict)” si vous avez un certificat SSL valide
2. **Mise en cache** :
   - Définissez le niveau de mise en cache à “Standard”
   - Activez “Always Online”
   - Définissez le TTL du cache navigateur à au moins 4 heures
3. **Vitesse** :
   - Activez Auto Minify pour HTML, CSS et JavaScript
   - Activez la compression Brotli
   - Activez Early Hints
   - Activez Rocket Loader pour JavaScript
4. **Règles de page** :
   - Créez une règle pour contourner le cache pour `/wp-admin/*`
   - Créez une règle pour contourner le cache pour les pages de connexion et de paiement

Conclusion

L’optimisation des performances WordPress est une approche multicouche qui implique :

1. **Optimisations au niveau serveur** : Réglage de PHP-FPM, configuration d’OPcache
2. **Configuration WordPress** : Révisions des articles, limites de mémoire, WP-Cron
3. **Stratégies de mise en cache** : Mise en cache de pages, mise en cache d’objets, mise en cache du navigateur
4. **Diffusion de contenu** : Intégration CDN avec Cloudflare

N’oubliez pas que l’optimisation des performances est un processus continu. Surveillez régulièrement les performances de votre site avec des outils tels que :

- Google PageSpeed Insights
- GTmetrix
- WebPageTest

Ajustez votre stratégie d’optimisation en fonction des besoins spécifiques de votre site WordPress et de vos schémas de trafic.

Choisissez votre stratégie de mise en cache

Pour des performances optimales, choisissez une approche de mise en cache statique ou dynamique selon les besoins de votre site :

Pour les sites statiques (blogs, sites vitrine) :

- Utilisez soit le cache FastCGI, WP Super Cache ou W3TC pour la mise en cache de pages

Pour les sites dynamiques (e-commerce, abonnement) :

- Combinez la mise en cache d’objets Redis avec soit WP Super Cache, soit W3TC
- Configurez soigneusement les exclusions de cache pour le contenu dynamique

9 Maintenance

9.1 Guide de maintenance serveur

Introduction

Ce guide de maintenance fournit les tâches et procédures essentielles pour garder votre serveur WordPress sécurisé, optimisé et fonctionnel. Il s'appuie sur la documentation d'installation et de configuration du serveur et se concentre sur les activités de maintenance régulières à effectuer pour garantir la santé et les performances de votre serveur.

Le guide est organisé par fréquence de maintenance (quotidienne, hebdomadaire, mensuelle) et par composant (serveur, WordPress, base de données, etc.), ce qui facilite la création d'un calendrier de maintenance adapté à vos besoins.

Tâches de maintenance quotidiennes

Surveillance système

1. **Vérifier les ressources système**
   Surveillez l'utilisation du processeur, de la mémoire et du disque pour identifier les problèmes potentiels avant qu'ils ne deviennent critiques.

```
# Vérifier l'utilisation actuelle des ressources système
htop

# Vérifier l'utilisation de l'espace disque
df -h

# Vérifier l'utilisation de la mémoire
free -m
```

2. **Examiner les journaux système**
   Vérifiez régulièrement les journaux système pour détecter les erreurs, avertissements ou activités inhabituelles.

```
# Vérifier les journaux système
sudo journalctl -p err..emerg --since "24 hours ago"

# Vérifier les journaux d'authentification pour les tentatives de connexion échouées
sudo grep "Failed password" /var/log/auth.log
```

3. **Surveiller le statut de Fail2ban**
   Vérifiez si Fail2ban bloque activement les adresses IP malveillantes.

```
# Vérifier le statut de Fail2ban
sudo fail2ban-client status

# Vérifier le statut d'une prison spécifique (par ex. sshd)
sudo fail2ban-client status sshd
```

Maintenance WordPress

1. **Vérifier les mises à jour WordPress**
   Vérifiez régulièrement les mises à jour du noyau WordPress, des thèmes et des extensions.

```
# Accéder à votre installation WordPress
cd /var/www/example.com/public_html

# Vérifier les mises à jour WordPress avec WP-CLI
sudo -u www-data wp core check-update
sudo -u www-data wp plugin list --update=available
sudo -u www-data wp theme list --update=available
```

2. **Examiner les journaux d'erreurs WordPress**
   Vérifiez les journaux d'erreurs WordPress pour détecter les erreurs ou avertissements PHP.

```
# Vérifier le journal d'erreurs WordPress
sudo tail -n 100 /var/www/example.com/logs/php_errors.log
```

Tâches de maintenance hebdomadaires

Maintenance de sécurité

1. **Mettre à jour les paquets système**
   Maintenez votre système à jour avec les derniers correctifs de sécurité.

```
# Mettre à jour la liste des paquets
sudo apt update

# Installer les mises à jour disponibles
sudo apt upgrade -y

# Supprimer les paquets inutilisés
sudo apt autoremove -y
```

2. **Vérifier les accès non autorisés**
   Examinez les comptes utilisateurs et l'historique des connexions pour détecter tout accès non autorisé.

```
# Lister tous les comptes utilisateurs
cat /etc/passwd | grep /home

# Vérifier l'historique des connexions
last

# Vérifier les tentatives de connexion échouées
sudo grep "Failed password" /var/log/auth.log | tail -n 20
```

3. **Vérifier les règles du pare-feu**
   Assurez-vous que les règles de votre pare-feu sont correctement configurées et actives.

```
# Vérifier le statut UFW
sudo ufw status verbose
```

Mises à jour WordPress

1. **Appliquer les mises à jour WordPress**
   Appliquez les mises à jour en attente du noyau WordPress, des thèmes et des extensions.

```
# Accéder à votre installation WordPress
cd /var/www/example.com/public_html

# Mettre à jour le noyau WordPress
sudo -u www-data wp core update

# Mettre à jour les extensions
sudo -u www-data wp plugin update --all

# Mettre à jour les thèmes
sudo -u www-data wp theme update --all
```

2. **Tester le fonctionnement du site**
   Après avoir appliqué les mises à jour, testez votre site web pour vous assurer que tout fonctionne correctement.

```
# Vérifier l'état de santé du site WordPress
sudo -u www-data wp site health status
```

Maintenance de la base de données

1. **Optimiser la base de données WordPress**
   Optimisez régulièrement votre base de données WordPress pour améliorer les performances.

```
# Accéder à votre installation WordPress
cd /var/www/example.com/public_html

# Optimiser la base de données WordPress
sudo -u www-data wp db optimize
```

2. **Sauvegarder la base de données WordPress**
   Créez des sauvegardes régulières de votre base de données WordPress.

```
# Accéder à votre installation WordPress
cd /var/www/example.com/public_html

# Sauvegarder la base de données WordPress
sudo -u www-data wp db export /var/www/example.com/backups/db-backup-$(date +%Y%m%d).sql
```

Tâches de maintenance mensuelles

Optimisation des performances

1. **Vérifier et optimiser l'espace d'échange (swap)**
   Vérifiez que l'espace d'échange est correctement configuré et fonctionnel.

```
# Vérifier le statut du swap
sudo swapon -s

# Si le swap n'est pas activé ou doit être recréé, suivez les instructions du guide de
# durcissement serveur :
# Server Hardening Guide - Configuring Swap Space
```

↪ **Server Hardening Guide - Configuring Swap Space**

2. **Vérifier les paramètres d'optimisation réseau**
   Vérifiez que les paramètres d'optimisation réseau sont toujours en place.

```
# Vérifier les paramètres d'optimisation réseau
sudo sysctl -a | grep -e "net.ipv4.tcp_syncookies" -e "net.core.somaxconn" -e
"net.ipv4.tcp_max_syn_backlog"

# Si les paramètres ne sont pas optimaux, suivez les instructions du guide de durcissement
# serveur :
# Server Hardening Guide - Network Layer Optimization
```

↪ **Server Hardening Guide - Network Layer Optimization**

3. **Vérifier les paramètres PHP OPcache**
   Vérifiez que PHP OPcache est correctement configuré pour des performances optimales.

```
# Vérifier les paramètres PHP OPcache
sudo grep -r "opcache" /etc/php/8.3/fpm/pool.d/

# Si les paramètres doivent être ajustés, suivez les instructions du guide d'optimisation
# WordPress :
# WordPress Optimization Guide - PHP OPcache
```

↪ **WordPress Optimization Guide - PHP OPcache**

4. **Vérifier la configuration du cache**
   Vérifiez que votre solution de mise en cache (FastCGI Cache ou WP Super Cache) fonctionne correctement.

```
# Pour FastCGI Cache, vérifier si les fichiers de cache existent
ls -la /var/run/SITE/

# Tester le cache avec curl
curl -I https://example.com

# Pour WP Super Cache, vérifier le statut de l'extension dans WordPress
cd /var/www/example.com/public_html
sudo -u www-data wp plugin status wp-super-cache
```

Audit de sécurité

1. **Exécuter des analyses de sécurité**
   Effectuez des analyses de sécurité régulières pour identifier les vulnérabilités potentielles.

```
# Installer et exécuter le scanner de sécurité Lynis
sudo apt install lynis
sudo lynis audit system
```

2. **Vérifier les modifications de fichiers**
   Vérifiez que les fichiers critiques du système et de WordPress n'ont pas été modifiés de manière inattendue.

```
# Pour les fichiers du noyau WordPress
cd /var/www/example.com/public_html
sudo -u www-data wp core verify-checksums
```

```
# Pour les fichiers système, si AIDE est installé
sudo aide.wrapper --check
```

3. **Examiner les permissions utilisateur**
   Assurez-vous que les permissions des fichiers et répertoires sont correctement définies.

```
# Vérifier les permissions des fichiers WordPress
cd /var/www/example.com/public_html
find . -type f -exec stat -c "%a %n" {} \; | grep -v "644"
find . -type d -exec stat -c "%a %n" {} \; | grep -v "755"

# Si les permissions doivent être corrigées, suivez les instructions du guide de
# durcissement WordPress :
# WordPress Security Guide - File Permissions
```

↪ **WordPress Security Guide - File Permissions**

Sauvegardes complètes

1. **Sauvegarde complète du système**
   Créez une sauvegarde complète de votre serveur, incluant tous les fichiers et bases de données.

```
# Créer un répertoire de sauvegarde s'il n'existe pas
sudo mkdir -p /var/backups/wordpress

# Sauvegarder les fichiers WordPress
sudo tar -czf /var/backups/wordpress/files-backup-$(date +%Y%m%d).tar.gz -C
/var/www/example.com .

# Sauvegarder la base de données WordPress
cd /var/www/example.com/public_html
sudo -u www-data wp db export /var/backups/wordpress/db-backup-$(date +%Y%m%d).sql

# Compresser la sauvegarde de la base de données
sudo gzip /var/backups/wordpress/db-backup-$(date +%Y%m%d).sql
```

2. **Tester la restauration des sauvegardes**
   Testez périodiquement votre processus de restauration des sauvegardes pour vous assurer que celles-ci sont valides.

```
# Créer un répertoire de test
sudo mkdir -p /var/www/backup-test

# Extraire les fichiers dans le répertoire de test
sudo tar -xzf /var/backups/wordpress/files-backup-$(date +%Y%m%d).tar.gz -C
/var/www/backup-test

# Vérifier que les fichiers critiques existent
ls -la /var/www/backup-test/public_html/wp-config.php

# Nettoyer le répertoire de test
sudo rm -rf /var/www/backup-test
```

Tâches de maintenance trimestrielles

Optimisation système

1. **Examiner et optimiser la configuration serveur**
   Examinez périodiquement la configuration de votre serveur pour identifier les optimisations potentielles.

```
# Vérifier la configuration Nginx
sudo nginx -t

# Vérifier la configuration PHP-FPM
sudo php-fpm8.3 -t

# Vérifier la configuration MariaDB
sudo mysqlcheck --all-databases --check --optimize
```

2. **Mettre à jour les certificats SSL**
   Assurez-vous que les certificats SSL sont à jour et correctement configurés.

```
# Vérifier l'expiration des certificats SSL
sudo certbot certificates

# Renouveler les certificats si nécessaire
sudo certbot renew --dry-run
```

Nettoyage WordPress

1. **Nettoyer les révisions d'articles**
   Si vous n'avez pas limité les révisions d'articles dans wp-config.php, nettoyez-les manuellement.

```
# Accéder à votre installation WordPress
cd /var/www/example.com/public_html

# Nettoyer les révisions d'articles avec WP-CLI
sudo -u www-data wp post delete $(wp post list --post_type=revision --format=ids) --force
```

2. **Supprimer les thèmes et extensions inutilisés**
   Supprimez les thèmes ou extensions qui ne sont pas utilisés.

```
# Lister les extensions inactives
sudo -u www-data wp plugin list --status=inactive

# Supprimer les extensions inactives
sudo -u www-data wp plugin delete $(wp plugin list --status=inactive --field=name)

# Lister les thèmes inactifs
sudo -u www-data wp theme list --status=inactive

# Supprimer les thèmes inactifs
# (sauf les thèmes Twenty* qui sont les thèmes par défaut de WordPress)
sudo -u www-data wp theme delete $(wp theme list --status=inactive --field=name |
 grep -v "^Twenty")
```

Références de maintenance

Pour des informations plus détaillées sur des tâches de maintenance spécifiques, consultez les guides suivants :

1. **Server Hardening Guide**
2. **Firewall Configuration Guide**
3. **Fail2ban Guide**
4. **LEMP Stack Optimization Guide**
5. **WordPress Security Guide**
6. **WordPress Optimization Guide**

Conclusion

La maintenance régulière est essentielle pour garder votre serveur WordPress sécurisé, optimisé et fiable. En suivant ce guide de maintenance et en établissant un calendrier de maintenance régulier, vous pouvez prévenir de nombreux problèmes courants et garantir que votre serveur continue de fonctionner de manière optimale.

N'oubliez pas d'adapter ce guide à vos besoins spécifiques et à la configuration de votre serveur. Certaines tâches peuvent nécessiter une exécution plus ou moins fréquente en fonction de votre volume de trafic, de vos exigences en matière de sécurité et d'autres facteurs.