BookStack
Plateforme de documentation auto-hébergée
Présentation, fonctionnalités et provisioning

Thierry Gayet
Thierry.Gayet@gmail.com

Association LinuXMaine

2026

Plan I

Comment se déroule ce module ?

Structure de la présentation

1. **Rôle de BookStack** : le besoin auquel il répond
2. **Présentation** : histoire, philosophie, architecture
3. **Fonctionnalités techniques** : ce qu'il sait faire
4. **Installations** : Docker, Ansible, OpenTofu
5. **Cas d'usages** : exemples concrets
6. **Lexique & Références**

Ce que vous allez apprendre

- Comprendre la place de BookStack dans une PME ou une asso
- Choisir entre installation manuelle, Docker, Ansible ou IaC
- Déployer une instance auto-hébergée
- Configurer l'authentification (LDAP, OIDC, SAML)
- Sauvegarder, restaurer et superviser
- Intégrer BookStack dans un écosystème DevSecOps

Conseil

Cette présentation est un guide de référence : chaque section peut être lue indépendamment.

Dépôt Git & Ressources

Cloner le dépôt du projet officiel

```
git clone https://github.com/BookStackApp/BookStack.git
cd BookStack/
```

Ressources principales

- Site officiel : `https://www.bookstackapp.com`
- Documentation : `https://www.bookstackapp.com/docs/`
- Dépôt GitHub : `https://github.com/BookStackApp/BookStack`

Ressources LinuXMaine

- Atelier DevSecOps : Provisioning
- Dépôt Framagit (formations) : `https://framagit.org/linuxmaine/`
- Contact : `Thierry.Gayet@gmail.com`

Le rôle de BookStack

- BookStack est une **plateforme de documentation auto-hébergée** (self-hosted)
- Son rôle : **centraliser, organiser et partager** la connaissance d'une équipe ou d'une organisation
- Il remplace un **wiki interne**, une base de procédures, un manuel utilisateur, ou une documentation projet
- Pensé pour être **simple** pour les non-techniciens et **puissant** pour les administrateurs
- **Open source** (licence MIT) : pas de dépendance à un éditeur, pas de télémétrie

Le besoin auquel il répond

Comment garder la documentation **à jour, recherchable, versionnée, et accessible** ?
Comment éviter les *Word échangés par mail*, les *Confluence cloud coûteux*, ou les *wikis abandonnés* ?

Rôle — Problèmes résolus

Sans BookStack

- Documentation éparpillée (mails, Drive, Word)
- Versions multiples et incohérentes
- Recherche manuelle, fichier par fichier
- Pas de contrôle d'accès fin
- Départ d'un collaborateur = savoir perdu
- Pas d'historique des modifications

Avec BookStack

- Source unique de vérité (single source of truth)
- Hiérarchie claire : Étagère → Livre → Chapitre → Page
- Recherche plein texte instantanée
- Permissions par rôle et par contenu
- Historique complet (revisions, diff)
- Export PDF/HTML/Markdown à la demande

Rôle — Pour qui ?

Profils d'utilisateurs

- **Équipes IT / DevOps** : runbooks, procédures
- **Associations** (LinuXMaine, GULL) : tutoriels, comptes-rendus
- **PME** : base de connaissance interne
- **Établissements scolaires** : supports de cours
- **Projets open source** : documentation utilisateur
- **Particuliers** : carnet de notes structuré

Cas d'usages typiques

- Wiki interne (alternative à Confluence)
- Base de connaissance support client
- Documentation technique versionnée
- Manuels utilisateur multilingues
- Procedures qualité (ISO, RGPD)
- Onboarding des nouveaux arrivants

Position dans l'écosystème

BookStack se situe entre un wiki minimaliste (DokuWiki) et une plateforme entreprise (Confluence, Notion). Il vise la **simplicité d'usage** sans sacrifier les fonctions essentielles.

Qu'est-ce que BookStack ?

- Application web de **gestion de documentation** écrite en **PHP** (framework Laravel)
- Créée en **2015** par **Dan Brown** (développeur britannique indépendant)
- Distribuée sous licence **MIT** — libre et gratuite
- Auto-hébergeable : vos données restent chez vous (RGPD-friendly)
- Communauté active : +15 000 étoiles GitHub, traductions dans +40 langues (dont français)
- Inspirée de l'organisation des bibliothèques : *Bookshelves* → *Books* → *Chapters* → *Pages*

Philosophie

Simple, beau, fonctionnel. Faire le travail sans imposer de complexité à l'utilisateur final.

BookStack — En clair

Analogie

Imaginez une **bibliothèque numérique** pour votre organisation. **Sans BookStack** : des dossiers Word éparpillés, des PDF dans Drive, et personne ne sait où chercher. **Avec BookStack** : des *étagères* qui contiennent des *livres*, organisés en *chapitres* et *pages*, recherchables, modifiables collaborativement.

- **Auto-hébergé** : installé sur votre serveur (VPS, cloud privé, machine locale)
- **Web** : accessible depuis un navigateur, pas d'installation cliente
- **Multi-utilisateurs** : permissions, rôles, historique
- **WYSIWYG** ou **Markdown** : choix de l'éditeur par utilisateur

En résumé

BookStack = un wiki moderne, beau, simple, auto-hébergé, sous licence MIT.

Historique de BookStack

- **2015** : premières lignes de code par Dan Brown (Royaume-Uni)
- **2016** : version 0.x publiée sur GitHub — adoption rapide par la communauté
- **2017** : version 0.20 — support LDAP, premières traductions
- **2018** : version 0.25 — API REST publique
- **2020** : version 0.30 — support Markdown natif, attachements, brouillons
- **2021** : version 21.x — nouveau schéma de versionnage (année.mois)
- **2022** : version 22.x — améliorations OIDC / SAML, custom HTML head
- **2024** : version 24.x — éditeur WYSIWYG modernisé, audit log amélioré
- **2025–2026** : versions 25.x / 26.x — consolidation, performances, recherche

De l'outil personnel à la solution adoptée par des PME, universités et administrations.

Architecture technique

Pile logicielle (stack)

- **PHP 8.2+** (langage serveur)
- **Laravel 11** (framework MVC)
- **MySQL 8** ou **MariaDB 10.6+**
- **Apache** ou **Nginx** (serveur web)
- **Composer** (paquets PHP)
- **Node.js** + **npm** (build des assets front)
- **TinyMCE** (éditeur WYSIWYG)
- **CodeMirror** (éditeur Markdown)

Composants optionnels

- **Redis** : cache et sessions
- **S3 / MinIO** : stockage des fichiers
- **LDAP / OIDC / SAML** : authentification
- **SMTP** : notifications email
- **Cron** : tâches planifiées (recherche)
- **Reverse proxy** (Traefik, Caddy, Nginx)
- **Let's Encrypt** : TLS gratuit

Empreinte

Une instance BookStack confortable tient sur **1 vCPU + 1 Go RAM + 5 Go disque**.
Pour 100 utilisateurs simultanés : 2 vCPU + 2 Go RAM suffisent largement.

Modèle hiérarchique du contenu

Organisation de la documentation

- **Shelves** (Étagères) : regroupent des livres par thème
  ex. : Étagère "Infrastructure"
- **Books** (Livres) : ouvrages complètement structurés
  ex. : Livre "Procédures réseau"
- **Chapters** (Chapitres) : sections d'un livre
  ex. : Chapitre "Configuration des switchs"
- **Pages** : feuilles de contenu (Markdown ou WYSIWYG)
  ex. : Page "Activation VLAN 42"

Particularités

- Une page peut être directement dans un livre (sans chapitre)
- Les chapitres ne sont pas obligatoires
- Les étagères sont une vue logique : un livre peut figurer sur plusieurs étagères
- Permissions héritées ou redéfinies à chaque niveau

Pourquoi cette structure ?

Elle reproduit l'organisation **naturelle** d'une bibliothèque : un utilisateur non-technique comprend immédiatement où chercher et où classer une nouvelle page.

À retenir — Présentation

Concepts clés

- BookStack = wiki moderne **auto-hébergé**, libre (MIT), écrit en PHP/Laravel
- Modèle hiérarchique : **Shelf** → **Book** → **Chapter** → **Page**
- Stack standard LAMP/LEMP : PHP, MySQL, Apache/Nginx
- Léger : 1 vCPU + 1 Go RAM suffisent pour démarrer
- Fonctionnalités entreprise : SSO, audit, API, permissions fines

Vocabulaire essentiel

- **Auto-hébergé (self-hosted)** : installé sur un serveur que vous contrôlez
- **Wiki** : site collaboratif éditable par les utilisateurs
- **WYSIWYG** : What You See Is What You Get — édition visuelle
- **Markdown** : langage de balisage léger (#, **, *)

Fonctionnalités éditoriales

Édition de contenu

- Éditeur **WYSIWYG** (TinyMCE)
- Éditeur **Markdown** (CodeMirror) avec aperçu temps réel
- Choix de l'éditeur par utilisateur ou par page
- Bloc de **code coloré** (200 langages)
- Insertion d'**images**, **tableaux**, **liens**, **diagrammes**
- Support des **PlantUML** et **Mermaid** (via plugins)
- Brouillons (drafts) sauvegardés automatiquement

Gestion documentaire

- **Historique** complet par page (revisions)
- **Diff** entre versions, restauration possible
- **Brouillons** et publications en deux temps
- **Pièces jointes** (attachements) liées à une page
- **Étiquettes** (tags) pour la classification croisée
- **Favoris** et **vu récemment** par utilisateur
- Tri et réorganisation par glisser-déposer

Fonctionnalités de recherche

- Recherche **plein texte** (full-text) sur le titre et le contenu de toutes les pages
- Recherche par **opérateurs** :
  - `{type:page}` `{type:book}` `{type:chapter}`
  - `[tag=valeur]` (filtre par étiquette)
  - `{updated_after:2026-01-01}`
  - `{created_by:me}` `{updated_by:me}`
- **Surlignage** des termes trouvés dans les résultats
- Indexation **automatique** à chaque modification
- Index **rebuilt** à la demande : `php artisan bookstack:regenerate-references`
- Recherche **contextuelle** dans un livre ou une étagère donnée

Performance

Index MySQL (FULLTEXT) : résultats en moins de 50 ms sur 10 000 pages.

Authentification & utilisateurs

Méthodes d'authentification

- **Local** (mot de passe en base, hash bcrypt)
- **LDAP / Active Directory**
- **SAML 2.0** (Keycloak, ADFS, Okta)
- **OIDC** (OpenID Connect : Google, Authentik, Keycloak)
- **Social login** : Google, GitHub, GitLab, Microsoft, Discord
- **2FA / TOTP** (depuis BookStack 23+)

Rôles & permissions

- Rôles prédéfinis : **Admin**, **Editor**, **Viewer**, **Public**
- Rôles **personnalisés** avec permissions granulaires
- Permissions par **shelf / book / chapter / page**
- Héritage des permissions parent → enfant
- **Override** possible à chaque niveau
- Mode **public** (lecture sans authentification)

Bonne pratique

Pour une organisation : synchroniser BookStack à votre IdP (Keycloak, Authentik) via OIDC ou SAML plutôt que de gérer les comptes manuellement.

API REST

Caractéristiques

- API **REST** documentée (HATEOAS partiel)
- Authentification par **token API** (par utilisateur)
- Format **JSON**
- Pagination, filtrage, tri standardisés
- Versionnage de l'API (suivi de la version BookStack)

Endpoints principaux

- `GET/POST /api/books`
- `GET/POST /api/chapters`
- `GET/POST /api/pages`
- `GET/POST /api/shelves`
- `GET /api/search`
- `GET /api/users`
- `GET /api/attachments`

Cas d'usage de l'API

Migration depuis Confluence/Notion, génération automatique de pages depuis Ansible/Jenkins, intégration avec un système de tickets, export massif, sauvegarde personnalisée.

Exemple d'appel API

```
# Lister les livres (avec un token API)
$ curl -H "Authorization: Token <ID>:<SECRET>" \
       https://wiki.example.org/api/books

# Creer une nouvelle page dans un livre (corps JSON dans data.json)
$ curl -X POST \
       -H "Authorization: Token <ID>:<SECRET>" \
       -H "Content-Type: application/json" \
       --data @data.json \
       https://wiki.example.org/api/pages

# Recherche plein texte (operateurs avances supportes)
$ curl -H "Authorization: Token <ID>:<SECRET>" \
       "https://wiki.example.org/api/search?query=ansible+vault"
```

Génération de tokens

Profil utilisateur → *API Tokens* → *Create Token*.
Token = couple `ID:SECRET`, à stocker comme un secret (Vault, sealed-secret, .env).

Imports / Exports

Exports

- Page à page ou livre complet
- Formats : **PDF**, **HTML**, **Markdown**, **Plain Text**
- Export PDF via **wkhtmltopdf** ou **Chromium** (headless)
- Personnalisation via `custom-head.blade.php`
- Export par lots via l'API

Imports

- Pas d'import natif (en 2026)
- Import **custom** via l'API (scripts Python/Bash)
- Outils communautaires : `confluence-to-bookstack`, `notion-to-bookstack`
- Migration manuelle Markdown → pages BookStack

Astuce

Pour migrer une grande base : écrire un script qui parcourt vos sources Markdown et appelle `POST /api/pages` en boucle. Comptée en quelques minutes pour 1 000 pages.

Personnalisation & thèmes

- **Logo** et **nom** de l'application configurables (Paramètres)
- **Couleur primaire** (CSS) configurable
- **HTML personnalisé** dans le `<head>` (statistiques, fontes, CSS)
- Mode **sombre** / **clair** natif (depuis 23.x)
- Thèmes via `themes/<nom>/` et la variable `APP_THEME`
- Substitutions de vues Blade (`themes/<nom>/views/...`)
- Hooks de **webhook** (notifications externes Slack, Mattermost, Discord)
- Configuration **custom translations** (fichiers JSON par langue)

Recommandation

Pour LinuXMaine : thème custom avec logo asso, couleur primaire `lmblue`, lien retour vers le site.

Sécurité

Mesures intégrées

- Hash **bcrypt** des mots de passe
- **CSRF** tokens sur tous les formulaires
- **Rate limiting** sur le login
- **Sessions** chiffrées (clef `APP_KEY`)
- Protection **XSS** (Content Security Policy)
- Audit log (qui a fait quoi, quand)
- 2FA / TOTP (depuis 23.x)

Recommandations déploiement

- **HTTPS obligatoire** (Let's Encrypt, mTLS interne)
- Reverse proxy avec **HSTS**
- Pare-feu applicatif (**WAF**) en amont
- Restriction IP de l'admin (`geoip`, fail2ban)
- Sauvegardes **chiffrées** hors site
- Mise à jour mensuelle des dépendances PHP

Clients d'accès à BookStack

- BookStack est une **application 100 % web** : pas de client natif officiel
- Le frontend est **responsive** et fonctionne en **PWA** (Progressive Web App)
- L'éditeur Markdown reste utilisable sur écran mobile ; le WYSIWYG est plutôt bureau

Bureau (Linux / macOS / Windows)

- Tous navigateurs modernes (Firefox, Chromium, Safari, Edge)
- **Installation comme PWA** (*Installer cette application*) depuis Chromium/Edge
- Wrappers communautaires : **Nativefier**, **Tauri**, **ElectronWrap** pour créer une app de bureau à partir de l'URL

Mobile

- **Android** : app tierce *BookStack Mobile* (non-officielle, Bektroid) sur F-Droid / Play Store
- **iOS** : pas d'app dédiée — usage via Safari + *Ajouter à l'écran d'accueil*
- **CLI** : scripts utilisant l'API (`curl`, Python, Bash) ; outils communautaires sur GitHub

Position du projet

L'équipe BookStack n'a pas prévu de client natif : l'effort de maintenance multi-plateforme n'est pas justifié tant que le PWA et le mobile-web couvrent les usages.

Extensibilité & plugins

- BookStack **ne dispose pas** d'un système de plugins formel (à la WordPress)
- L'extensibilité passe par 4 mécanismes officiels :

1. Système de thèmes (`themes/`)

- Variable `APP_THEME=lmaine` dans `.env`
- Répertoire `themes/lmaine/`
- Surcharge de vues Blade (`views/...`)
- Traductions custom (`lang/<code>/...`)
- Fichier `functions.php` (hooks)

2. Logical Theme System

- `Theme::listen('routes-register-web', ...)`
- Événements : `commonmark-environment-configure`, `web-middleware`

3. HTML personnalisé & webhooks

- *Custom HTML head* pour CSS, JS, fontes, statistiques
- Intégration **Mermaid**, **PlantUML**, **KaTeX** via JS embarqué
- *Drawio* (diagrams.net) intégré nativement
- **Webhooks sortants** : Slack, Mattermost, Discord, n8n

4. API REST & artisan

- Tout ce que l'IHM ne fait pas, l'API peut le scripter
- Commandes Artisan custom dans `app/Console/Commands/`

SSO avec Keycloak (OIDC)

Pourquoi Keycloak ?

- IAM open source mature (Red Hat) supportant **OIDC** et **SAML 2.0**
- BookStack se branche en client **OIDC** (recommandé) ou SAML
- Avantages : SSO unique, MFA, fédération LDAP/AD, brute-force protection, audit

Côté Keycloak

- Créer un **Realm** (ou réutiliser celui de l'organisation)
- Créer un **Client OIDC** : `bookstack`, type *Confidential*
- Valid Redirect URI : `https://wiki.example.org/oidc/callback`
- Récupérer le **Client Secret** et l'**Issuer URL**
- Mapper les *groupes* Keycloak vers les *rôles* BookStack (mapper `groups`)

.env BookStack — bloc OIDC pour Keycloak

```
AUTH_METHOD=oidc
AUTH_AUTO_INITIATE=false
OIDC_NAME=Keycloak
OIDC_DISPLAY_NAME_CLAIMS=name
OIDC_CLIENT_ID=bookstack
OIDC_CLIENT_SECRET=AbCdEf123456...
OIDC_ISSUER=https://sso.example.org/realms/linuxmaine
OIDC_ISSUER_DISCOVER=true
OIDC_USER_TO_GROUPS=true
OIDC_GROUPS_CLAIM=groups
```

SSO Keycloak — bonnes pratiques

Sécurité

- Activer le **discovery OIDC** (`OIDC_ISSUER_DISCOVER=true`)
- TLS **obligatoire** côté Keycloak et BookStack
- Ne pas exposer le *Client Secret* en Git (utiliser ansible-vault, sealed-secret, ou 1Password CLI)
- Activer le **logout SLO** : `OIDC_END_SESSION_ENDPOINT=true`
- Activer la **MFA / TOTP** côté Keycloak (BookStack en hérite)

Mapping des rôles

- Créer dans Keycloak les groupes `bookstack-admin`, `bookstack-editor`, `bookstack-viewer`
- Créer dans BookStack les rôles *de même nom*
- BookStack synchronise automatiquement à chaque login
- `OIDC_REMOVE_FROM_GROUPS=true` : révoque les accès retirés dans Keycloak

Vérifications

- URL de test : `https://wiki.example.org/login` → bouton *Connexion via Keycloak*
- Logs : `storage/logs/laravel.log` côté BookStack, *Events* côté Keycloak

À retenir — Fonctionnalités

Points clés

- **Édition** : WYSIWYG + Markdown, brouillons, historique complet
- **Recherche** : full-text MySQL avec opérateurs avancés
- **Authentification** : local + LDAP + SAML + OIDC + 2FA, SSO Keycloak natif
- **Permissions** : par rôle et par contenu (héritage + override)
- **API REST** : automatisation, migration, intégration
- **Export** : PDF, HTML, Markdown, Plain Text
- **Clients** : web + PWA, app Android tierce, pas d'app iOS dédiée
- **Plugins** : pas de système formel, mais thèmes + hooks + webhooks + API

Pré-requis serveur

Système

- Serveur Linux (Debian 12, Ubuntu 24.04, Rocky/Alma 9)
- 1 vCPU minimum (2 recommandés)
- 1 Go RAM minimum (2 Go recommandés)
- 5 Go disque (croissance à prévoir)
- Accès Internet pour Composer / npm

Logiciels

- **PHP 8.2+** et extensions (mbstring, gd, mysql, xml, curl, zip, intl, fileinfo)
- **MySQL 8** ou **MariaDB 10.6+**
- **Apache 2.4** ou **Nginx 1.20+**
- **Composer 2**
- **Node.js 20+** et npm
- **Git**

Installation pas à pas (Debian 12)

```
# 1. Paquets systeme
sudo apt update && sudo apt install -y \
  apache2 mariadb-server git unzip curl \
  php php-{cli,fpm,mysql,mbstring,gd,xml,curl,zip,intl,fileinfo} \
  composer nodejs npm

# 2. Base de donnees
sudo mysql -e "CREATE DATABASE bookstack CHARACTER SET utf8mb4;"
sudo mysql -e "CREATE USER 'bookstack'@'localhost' IDENTIFIED BY 'changeme';"
sudo mysql -e "GRANT ALL ON bookstack.* TO 'bookstack'@'localhost';"

# 3. Sources de l'application
sudo git clone https://github.com/BookStackApp/BookStack.git \
     --branch release --single-branch /var/www/bookstack
cd /var/www/bookstack
sudo composer install --no-dev
sudo cp .env.example .env
sudo php artisan key:generate

# 4. Migrations + assets
sudo php artisan migrate --force
sudo npm ci && sudo npm run build
sudo chown -R www-data:www-data storage bootstrap/cache public/uploads
```

Fichier .env (extrait)

/var/www/bookstack/.env

```
APP_NAME="LinuXMaine Wiki"
APP_URL=https://wiki.linuxmaine.org
APP_KEY=base64:GENERE_PAR_php_artisan_key:generate
APP_DEBUG=false
APP_ENV=production

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_DATABASE=bookstack
DB_USERNAME=bookstack
DB_PASSWORD=changeme

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_CONNECTION=database

MAIL_DRIVER=smtp
MAIL_HOST=smtp.example.org
MAIL_PORT=587
MAIL_USERNAME=wiki@example.org
MAIL_PASSWORD=secret
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=wiki@example.org
MAIL_FROM_NAME="LinuXMaine Wiki"
```

VirtualHost Apache

/etc/apache2/sites-available/bookstack.conf

```
<VirtualHost *:80>
    ServerName wiki.linuxmaine.org
    DocumentRoot /var/www/bookstack/public
    <Directory /var/www/bookstack/public>
        AllowOverride All
        Require all granted
    </Directory>
    ErrorLog  ${APACHE_LOG_DIR}/bookstack-error.log
    CustomLog ${APACHE_LOG_DIR}/bookstack-access.log combined
</VirtualHost>
```

```
sudo a2enmod rewrite
sudo a2ensite bookstack
sudo systemctl reload apache2
# TLS via Let's Encrypt
sudo apt install -y certbot python3-certbot-apache
sudo certbot --apache -d wiki.linuxmaine.org
```

Pourquoi Docker ?

- Déploiement **reproductible** (image versionnée)
- **Isolation** de la pile PHP/MySQL du système hôte
- Mise à jour simple : `docker compose pull && docker compose up -d`
- Sauvegarde par **volumes** : `bookstack_app`, `bookstack_db`
- Combinable avec **Traefik** ou **Caddy** pour le TLS automatique
- Image officielle communautaire : `lscr.io/linuxserver/bookstack`

Recommandation

Pour une asso ou une PME : l'image LinuxServer.io + Docker Compose + Traefik est le combo le plus rapide à mettre en place.

docker-compose.yml minimal

docker-compose.yml

```
version: "3.9"
services:
  bookstack:
    image: lscr.io/linuxserver/bookstack:latest
    container_name: bookstack
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Paris
      - APP_URL=https://wiki.linuxmaine.org
      - APP_KEY=base64:GENERE_AVEC_php_artisan_key:generate
      - DB_HOST=bookstack_db
      - DB_PORT=3306
      - DB_DATABASE=bookstack
      - DB_USERNAME=bookstack
      - DB_PASSWORD=changeme
    volumes:
      - ./config:/config
    ports:
      - "6875:80"
    depends_on: [bookstack_db]
    restart: unless-stopped

  bookstack_db:
    image: lscr.io/linuxserver/mariadb:latest
    container_name: bookstack_db
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Paris
      - MYSQL_ROOT_PASSWORD=changeme_root
```

Démarrage et accès

```
# 1. Lancer la stack
$ docker compose up -d

# 2. Voir les logs (verifier la migration)
$ docker compose logs -f bookstack

# 3. Acceder a l'instance
#    http://localhost:6875
#    Identifiants par defaut : admin@admin.com / password
#    A CHANGER au premier login !

# 4. Mise a jour
$ docker compose pull
$ docker compose up -d

# 5. Sauvegarde des volumes
$ tar czf bookstack-backup-$(date +%F).tgz config/ db/

# 6. Generer une cle APP_KEY si besoin
$ docker exec -it bookstack \
    php /app/www/artisan key:generate --show
```

Production

Ajouter Traefik / Caddy en reverse proxy pour TLS, HSTS, et le routage par domaine.

Pourquoi Ansible pour BookStack ?

- Installation **reproductible** sur N serveurs identiques (PROD, PRE-PROD, DEV)
- Code **versionné** (Git) et révisable (Pull Request)
- **Idempotent** : relancer ne casse rien si rien n'a changé
- Intégrable dans une chaîne **CI/CD** (GitLab CI, GitHub Actions)
- Combinable avec **ansible-vault** pour chiffrer les mots de passe
- Rôles **communautaires** disponibles (`ansible-galaxy`) :
  ex. `geerlingguy.php`, `geerlingguy.mysql`

Approche préconisée

Un **rôle BookStack** maison qui dépend des rôles communautaires PHP / MySQL / Apache. Le playbook orchestre l'ensemble : `site.yml` → `roles/bookstack/`.

Inventaire et structure du rôle

inventory.ini

```
[wiki]
wiki01.linuxmaine.org ansible_user=deploy
wiki02.linuxmaine.org ansible_user=deploy

[wiki:vars]
bookstack_db_password={{ vault_db_password }}
bookstack_app_url=https://wiki.linuxmaine.org
```

Arborescence du role

```
roles/bookstack/
|-- defaults/main.yml          # variables par defaut
|-- vars/main.yml              # variables non-overridables
|-- tasks/
|   |-- main.yml               # entree principale
|   |-- packages.yml           # apt php / mariadb / apache
|   |-- database.yml           # creation DB + user
|   |-- install.yml            # git clone + composer + npm
|   |-- configure.yml          # template .env
|   `-- web.yml                # vhost apache + tls
|-- handlers/main.yml          # reload apache, restart php-fpm
|-- templates/
|   |-- env.j2                 # .env BookStack
|   `-- vhost.conf.j2          # vhost apache
`-- meta/main.yml              # dependencies (php, mysql)
```

Playbook site.yml

site.yml

```
---
- name: Deployer BookStack sur les serveurs wiki
  hosts: wiki
  become: true
  vars_files:
    - vars/secrets.yml          # ansible-vault edit
  roles:
    - role: geerlingguy.php
      vars:
        php_version: "8.2"
        php_packages:
          - php-cli
          - php-fpm
          - php-mysql
          - php-mbstring
          - php-gd
          - php-xml
          - php-curl
          - php-zip
          - php-intl
          - php-fileinfo
    - role: geerlingguy.mysql
      vars:
        mysql_databases:
          - { name: bookstack, encoding: utf8mb4 }
        mysql_users:
          - { name: bookstack, password: "{{ vault_db_password }}",
              priv: "bookstack.*:ALL" }
    - role: bookstack
```

Rôle bookstack/tasks/install.yml

roles/bookstack/tasks/install.yml

```
---
- name: Cloner les sources BookStack (branche release)
  ansible.builtin.git:
    repo: https://github.com/BookStackApp/BookStack.git
    dest: /var/www/bookstack
    version: release
    force: yes

- name: Installer les dependances Composer
  ansible.builtin.command:
    cmd: composer install --no-dev --optimize-autoloader
    chdir: /var/www/bookstack
  environment:
    COMPOSER_ALLOW_SUPERUSER: "1"
  changed_when: true

- name: Generer la cle d'application
  ansible.builtin.command:
    cmd: php artisan key:generate --force
    chdir: /var/www/bookstack
    creates: /var/www/bookstack/.env

- name: Migrer la base de donnees
  ansible.builtin.command:
    cmd: php artisan migrate --force
    chdir: /var/www/bookstack
  changed_when: true

- name: Construire les assets front (npm)
  ansible.builtin.command:
    cmd: "{{ item }}"
```

Template env.j2

roles/bookstack/templates/env.j2

```
APP_NAME="{{ bookstack_app_name | default('BookStack') }}"
APP_URL={{ bookstack_app_url }}
APP_KEY={{ bookstack_app_key }}
APP_DEBUG=false
APP_ENV=production

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_DATABASE=bookstack
DB_USERNAME=bookstack
DB_PASSWORD={{ bookstack_db_password }}

MAIL_DRIVER=smtp
MAIL_HOST={{ smtp_host }}
MAIL_PORT={{ smtp_port }}
MAIL_USERNAME={{ smtp_user }}
MAIL_PASSWORD={{ smtp_password }}
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS={{ mail_from }}
MAIL_FROM_NAME="{{ mail_from_name }}"
```

Lancement

```
ansible-playbook -i inventory.ini site.yml -ask-vault-pass
```

Pourquoi OpenTofu pour BookStack ?

- OpenTofu = **fork open source** de Terraform (Linux Foundation, MPL 2.0)
- Approche **déclarative** : on décrit l'état désiré, l'outil calcule le diff
- Crée la **couche infrastructure** (VM, réseau, DNS, certificat) avant Ansible
- **State file** (`terraform.tfstate`) : source de vérité de l'infra
- Multi-cloud : AWS, Azure, GCP, Scaleway, OVH, Hetzner, Proxmox...
- Combinaison **IaC + Configuration Management** :
  OpenTofu crée la VM → Ansible installe BookStack

Exemple : VM Hetzner Cloud + DNS

main.tf

```
terraform {
  required_providers {
    hcloud = { source = "hetznercloud/hcloud", version = "~> 1.45" }
    cloudflare = { source = "cloudflare/cloudflare", version = "~> 4.0" }
  }
}

provider "hcloud"     { token = var.hcloud_token }
provider "cloudflare" { api_token = var.cf_token }

resource "hcloud_ssh_key" "deploy" {
  name       = "deploy-key"
  public_key = file("~/.ssh/id_ed25519.pub")
}

resource "hcloud_server" "bookstack" {
  name        = "wiki-linuxmaine"
  image       = "debian-12"
  server_type = "cx22"      # 2 vCPU, 4 Go RAM
  location    = "fsn1"
  ssh_keys    = [hcloud_ssh_key.deploy.id]
  user_data   = file("cloud-init.yml")
  labels = {
    role = "bookstack"
    env  = "prod"
  }
}

resource "cloudflare_record" "wiki" {
  zone_id = var.cf_zone_id
  name    = "wiki"
```

cloud-init et lien avec Ansible

cloud-init.yml

```
#cloud-config
package_update: true
packages:
  - python3
  - sudo
  - curl
users:
  - name: deploy
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash
    ssh_authorized_keys:
      - ssh-ed25519 AAAA...
```

Chaîne complète

```
# 1. Provisionner l'infra avec OpenTofu
$ tofu init
$ tofu plan -out=plan.bin
$ tofu apply plan.bin

# 2. Recuperer l'IP et generer l'inventaire Ansible
$ tofu output -raw bookstack_ip > inventory.ini

# 3. Deployer BookStack avec Ansible
$ ansible-playbook -i inventory.ini site.yml --ask-vault-pass

# 4. Verifier
$ curl -I https://wiki.linuxmaine.org
```

State file — à savoir

- Le `terraform.tfstate` contient des **secrets** (mots de passe, tokens)
- **Ne jamais** le commiter en clair dans Git
- Stockage recommandé : **remote backend** chiffré
  - S3 + DynamoDB (verrou)
  - GCS (Google Cloud Storage)
  - Azure Blob Storage
  - Backend HTTP/HTTPS (Atlantis, Terraform Cloud, Scalr)
  - Backend natif OpenTofu (lien avec un Vault chiffré)
- **Verrou** (state lock) obligatoire pour le travail en équipe
- `tofu state list / show / mv / rm` : manipuler l'état

Bonnes pratiques

Versionner le *code* (`*.tf`), **pas** l'*état* (`*.tfstate`). Le pipeline CI applique l'IaC, pas le poste local.

Stratégie de sauvegarde

Ce qu'il faut sauvegarder

- **Base de données** (`mysqldump` ou snapshot)
- **Dossier** `public/uploads/` (images uploadées)
- **Dossier** `storage/uploads/` (pièces jointes)
- Fichier `.env` (configuration et `APP_KEY`)
- Thèmes / vues custom (`themes/`, `resources/views/`)

Règle 3-2-1

- **3** copies des données
- sur **2** supports différents
- dont **1** hors site
- **Chiffrement** via age, gpg ou `restic`
- **Test** de restauration périodique

Script de sauvegarde

/usr/local/sbin/bookstack-backup.sh

```
#!/usr/bin/env bash
set -euo pipefail

BACKUP_DIR=/var/backups/bookstack
APP_DIR=/var/www/bookstack
DATE=$(date +%F-%H%M)
mkdir -p "$BACKUP_DIR"

# 1. Dump MySQL
mysqldump --single-transaction --quick --routines bookstack \
  | gzip > "$BACKUP_DIR/db-$DATE.sql.gz"

# 2. Tar des uploads et du .env
tar czf "$BACKUP_DIR/files-$DATE.tgz" \
  -C "$APP_DIR" public/uploads storage/uploads .env

# 3. Chiffrement (age)
age -r "age1xxxxxxxxxxxx" \
  -o "$BACKUP_DIR/db-$DATE.sql.gz.age" \
  "$BACKUP_DIR/db-$DATE.sql.gz"
rm -f "$BACKUP_DIR/db-$DATE.sql.gz"

# 4. Synchronisation S3 / Backblaze
restic -r "s3:s3.amazonaws.com/bucket-bookstack" backup "$BACKUP_DIR"

# 5. Retention 14 jours
find "$BACKUP_DIR" -mtime +14 -delete
```

Mise à jour de BookStack

```
# Mode "binaire" (git) - depuis /var/www/bookstack
sudo -u www-data git pull origin release
sudo -u www-data composer install --no-dev --optimize-autoloader
sudo -u www-data php artisan migrate --force
sudo -u www-data php artisan cache:clear
sudo -u www-data php artisan view:clear
sudo -u www-data npm ci && sudo -u www-data npm run build

# Mode Docker
docker compose pull
docker compose up -d

# Verifier la version
php artisan --version
# ou en API : GET /api/system/info
```

Bonne pratique

Toujours **sauvegarder avant** une mise à jour majeure (24.x → 25.x). Lire les *release notes* sur GitHub pour les ruptures de configuration.

Cas d'usage 1 — Wiki d'association

Contexte

- Association loi 1901 (LinuXMaine) : 50 membres, 5 administrateurs
- Besoin : comptes-rendus de réunions, tutoriels Linux, statuts, procés-verbaux
- Budget : VPS à 5 EUR/mois (Hetzner CX11)

Architecture

- VPS Debian 12 (1 vCPU, 2 Go RAM, 20 Go SSD)
- BookStack en Docker derrière Caddy (TLS auto)
- Authentification Keycloak (OIDC) partagée avec d'autres outils de l'asso
- Sauvegardes quotidiennes vers Backblaze B2 (chiffrement age)

Résultat

Wiki opérationnel en 2 heures, coût total : 60 EUR/an + 1 EUR/mois de stockage backup.

Cas d'usage 2 — Documentation projet open source

Contexte

- Projet logiciel libre avec 5 développeurs
- Besoin : documentation utilisateur publique + documentation interne privée
- Workflow GitOps souhaité

Architecture

- BookStack auto-hébergé sur un VPS, deux instances (publique + privée)
- Synchronisation des pages publiques depuis un dépôt Git (Markdown) via l'API REST
- Pipeline GitLab CI : `git push` → `POST /api/pages`
- Rôle Public en lecture seule, role Editor pour les contributeurs

Résultat

La documentation suit le même cycle de vie que le code, mais reste lisible par les utilisateurs finaux.

Cas d'usage 3 — Knowledge base PME

Contexte

- PME de 30 personnes (commercial, technique, RH)
- Besoin : procédures internes, base de connaissance support client, onboarding
- Conformité RGPD : aucune donnée exfiltrée chez un éditeur tiers

Architecture

- OpenTofu provisionne 2 VM Proxmox (BookStack + MySQL répliqué)
- Ansible déploie BookStack, Apache, MariaDB, certbot
- Authentification SAML via l'AD Microsoft de l'entreprise
- Reverse proxy Nginx interne (**pas** d'exposition Internet)
- Sauvegardes quotidiennes sur NAS Synology + réplique hors site

Résultat

Confluence remplacé en 2 semaines, économie annuelle estimée à 8 000 EUR.

Cas d'usage 4 — Documentation DevSecOps

Contexte

- Équipe DevOps de 10 personnes, 50 microservices, 200 runbooks
- Besoin : corréler runbooks, schémas, post-mortems, on-call

Intégration

- Une **étagère** BookStack par produit, un **livre** par microservice
- Webhooks BookStack → Mattermost (notifications de modifications)
- Diagrammes **Mermaid** dans les pages (architecture, séquences)
- Liens croisés BookStack ↔ Grafana ↔ Jira
- **Ansible Lightspeed** génère les playbooks, BookStack documente les runbooks

Résultat

Réduction du MTTR (Mean Time To Repair) de 25% grâce à la centralisation des procédures.

Refcard — Commandes administrateur

Maintenance

- `php artisan migrate -force` — migration BD
- `php artisan cache:clear` — vider le cache
- `php artisan view:clear` — vider les vues compilées
- `php artisan key:generate` — nouvelle `APP_KEY`
- `php artisan bookstack:regenerate-references`
- `php artisan bookstack:cleanup-images`
- `php artisan list bookstack:*`

Diagnostic

- `php artisan -version`
- `php artisan route:list`
- Logs : `storage/logs/laravel.log`
- Test mail : `php artisan bookstack:test-email`
- Test SSO : page `/saml2/metadata`
- `tail -f storage/logs/laravel.log`

Refcard — Variables `.env` essentielles

Général

- `APP_NAME` — nom affiché
- `APP_URL` — URL canonique (HTTPS)
- `APP_KEY` — clé de chiffrement
- `APP_DEBUG` — jamais `true` en prod
- `APP_LANG` — langue par défaut (`fr`)
- `APP_THEME` — thème custom

Authentification

- `AUTH_METHOD` — `standard|ldap|saml2|oidc`
- `LDAP_SERVER` — serveur LDAP/AD
- `OIDC_ISSUER` — URL IdP OIDC
- `SAML2_IDP_ENTITYID` — IdP SAML
- `ALLOWED_IFRAME_HOSTS` — iframes autorisés
- `REVISION_LIMIT` — nombre de revisions

Lexique (1/2)

Termes généraux

Auto-hébergé Application installée sur un serveur que vous contrôlez

Wiki Site web collaboratif éditable par les utilisateurs autorisés

CMS Content Management System — système de gestion de contenu

WYSIWYG What You See Is What You Get — édition visuelle

Markdown Langage de balisage léger lisible (#, **, *)

Reverse proxy Serveur frontal qui relaie vers une application

TLS / HTTPS Chiffrement de la communication navigateur ↔ serveur

Termes BookStack

Shelf Étagère — regroupement de livres par thème

Book Livre — ouvrage de documentation

Chapter Chapitre — section d'un livre

Page Page — unité de contenu (Markdown ou HTML)

Revision Version historisée d'une page

Draft Brouillon non publié

Tag Étiquette de classification croisée

Attachment Pièce jointe d'une page

Audit log Journal des actions des utilisateurs

Theme Personnalisation graphique

Lexique (2/2)

Termes infrastructure

LAMP Linux + Apache + MySQL + PHP

LEMP Linux + Nginx + MySQL + PHP

VPS Virtual Private Server — serveur privé virtuel

Conteneur Processus isolé (Docker, Podman)

Volume Espace persistent associé à un conteneur

IaC Infrastructure as Code — infra décrite en fichiers

Idempotence N exécutions = même résultat qu'une seule

Provisioning Installation automatisée d'un système

State file Fichier d'état (Terraform / OpenTofu)

Cron Planificateur de tâches Unix

Termes authentification

LDAP Protocole d'annuaire (Active Directory)

SAML 2.0 Protocole d'authentification fédérée XML

OIDC OpenID Connect — SSO basé sur OAuth 2.0

SSO / SLO Authentification / déconnexion unique

IdP Identity Provider — fournisseur d'identité (Keycloak, Authentik...)

Keycloak IAM open source Red Hat (OIDC + SAML)

MFA / TOTP Double facteur (codes temporaires)

RBAC Contrôle d'accès basé sur les rôles

Références — Liens utiles

Documentation officielle BookStack

- https://www.bookstackapp.com/ — site officiel
- https://www.bookstackapp.com/docs/ — documentation
- https://github.com/BookStackApp/BookStack — code source
- https://demo.bookstackapp.com/ — démo en ligne
- https://www.bookstackapp.com/blog/ — blog officiel
- https://github.com/BookStackApp/BookStack/releases — versions

Provisioning, IaC & SSO

- https://docs.ansible.com/ — Ansible
- https://galaxy.ansible.com/ — rôles Ansible
- https://opentofu.org/ — OpenTofu
- https://docs.docker.com/compose/ — Docker Compose
- https://docs.linuxserver.io/images/docker-bookstack/ — image Docker
- https://www.keycloak.org/ — Keycloak (IdP OIDC/SAML)
- https://www.bookstackapp.com/docs/admin/oidc-auth/ — guide OIDC officiel

Références — Communauté & écosystème

Communauté

- https://discord.gg/ztkBqR2 — Discord BookStack
- https://github.com/BookStackApp/BookStack/discussions — forum
- https://reddit.com/r/BookStack/ — subreddit
- https://crowdin.com/project/bookstack — traductions

Ressources francophones

- https://www.linuxtricks.fr/wiki/ — tutoriels Linux
- https://www.it-connect.fr/ — articles BookStack
- https://framalibre.org/ — annuaire libre
- https://www.linuxmaine.org/ — LinuXMaine

Sécurité

https://owasp.org/www-project-top-ten/ — OWASP Top 10
https://www.cnil.fr/ — recommandations RGPD (CNIL)

Bibliographie (en français)

Ouvrages et articles francophones

- **B. Boutherin, B. Delaunay** — *Sécuriser un serveur Linux*, ENI Éditions, 5e édition, 2023.
- **Sébastien Rohaut** — *Linux — Préparation à la certification LPIC-1*, ENI, 2024.
- **Jean-François Pillou** — *Tout sur les serveurs web*, Dunod, 2022.
- **Stephane Bortzmeyer** — *Cyberstructure — L'Internet, un espace politique*, C&F Éditions, 2018.
- **Olivier Hoarau** — *Administrer un serveur Debian/Ubuntu*, livre numérique 2024 (`https://www.funix.org`).
- **Linuxfr.org** — Dépêches BookStack et auto-hébergement, 2018–2026.
- **Framasoft** — *Dégôoglisons Internet*, livret pédagogique, 2020.
- **April** — *Catalogue libre*, fiche « BookStack », mise à jour 2025.
- **ANSSI** — *Recommandations de sécurité relatives à un système GNU/Linux*, guide technique, 2023.
- **LinuXMaine** — *Atelier Provisionning DevSecOps*, supports de cours, 2026.

Merci !

Questions ?

Thierry.Gayet@gmail.com
Association LinuXMaine — 2026