BookStack - Wiki-Software

Aus dem Englischen übersetzt-BookStack ist eine kostenlose Open-Source-Wiki-Software, die auf eine einfache, selbst gehostete und benutzerfreundliche Plattform abzielt. Basierend auf Laravel, einem PHP-Framework, wird BookStack unter der MIT-Lizenz veröffentlicht.

- Installation
  - Installtion via Ansible
  - Installation via Docker image
- HTML Formatierungen, da BookStack nicht so viele Module wie Confluence hat
  - InfoBox
- Wartung und Bedienung
  - Terminal Cli Befehle
  - Backup and Restore

Installation

Installtion via Ansible

Vorrausetzungen.

Ein Debian Bullseye mit ssh Zugang per Schlüsseldatei.

Denn Bookstack installieren wir remote bequem mit Ansible.
Und auf unserem Remote Client / z.b Laptop oder PC installieren wir Ansible.

Wie Ihr Ansible installiert und verwendet seht ihr in unserem Buch: Ansible.
Aber auch ohne ansible Kenntnisse ist das hier installierbar.

Ansible und diese Ansible-Rollen installieren.
Vom Homeverzeichnis aus also:

```
cd ~
```

Nun diese Befehle absetzten

```
# install packages an ansible machine
sudo apt install python3-pip
pip install -U Jinja2
sudo pip install ansible
ansible-galaxy install supertarto.bookstack
ansible-galaxy install supertarto.apache
ansible-galaxy install supertarto.php
ansible-galaxy install supertarto.mariadb
```

Nun legen dazu einen neue Datei bookstack.yaml an.
Dies ist unser Playbook, sprich unser installationsscript.
Im Home Verzeichnis auf euren Computer / Laptop einfach ein neues Verzeichnis namens Ansible erstellen

```
mkdir  ~/ansible
```

Dann eine neue Datei anlegen

```
nano ~/ansible/boockstack.yaml
```

Folgender Inhalt einfügen:

```
# install packages an ansible machine
#ansible-galaxy install supertarto.bookstack
#ansible-galaxy install supertarto.apache
#ansible-galaxy install supertarto.php
#ansible-galaxy install supertarto.mariadb
#pip install -U Jinja2

#standard weblogin data :
#Username : admin@admin.com
#Pass : password



- hosts: all
  roles:
    - role: supertarto.apache
    - role: supertarto.php
    - role: supertarto.mariadb
    - role: supertarto.bookstack

  pre_tasks:
    - name: Update apt cache.
      apt:
        update_cache: true
        cache_valid_time: 600
      when: ansible_os_family == 'Debian'
      changed_when: false
  vars:
    php_packages:
      - php7.4
      - php7.4-mysql
      - php7.4-curl
      - php7.4-pdo
      - php7.4-xml
      - php7.4-mbstring
      - php7.4-gd
```

```
  - php-tokenizer
  - php7.4-tidy
#hier den tatsächlinen hostname aus der /etc/hostname angeben
bookstack_host: localhost
bookstack_db_name: bookstackdb
bookstack_db_user: bookstackuser
bookstack_db_password: <unser sicheres password für die db>
bookstack_lang: de
apache_create_vhosts: true
apache_mods_enabled:
  - rewrite

#wenn eine eigene vhosts angelegt werden soll
#apache_vhosts_filename: "bookstack.conf"
#wenn nur eintrag dann in die default vhosts
apache_vhosts_filename: "000-default.conf"
apache_vhost_config:
  - listen_ip: "*"
    listen_port: 80
    server_name: 192.168.178.209
    documentroot: "/var/www/Bookstack/public"
    serveradmin: admin@localhost
    custom_param: |
      ErrorLog ${APACHE_LOG_DIR}/error.log
      CustomLog ${APACHE_LOG_DIR}/access.log combined
      LogLevel warn
    directory:
      - path: "/var/www/Bookstack/public"
        config: |
          AllowOverride All
          Order deny,allow
          allow from all
          Options +FollowSymLinks
          RewriteEngine On
          RewriteCond %{REQUEST_FILENAME} !-d
          RewriteCond %{REQUEST_FILENAME} !-f
          RewriteRule ^ index.php [L]

mariadb_use_dump_script: false
mariadb_databases:
```

```
    - name: "{{ bookstack_db_name }}"

  mariadb_users:
    - name: "{{ bookstack_db_user }}"
      host: "{{ bookstack_host }}"
      password: "{{ bookstack_db_password }}"
      priv: "{{ bookstack_db_name }}.*:SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER,CREATE
TEMPORARY TABLES,LOCK TABLES"
```

Nun noch folgenden Abschnitt in der Datei anpassen.

```
...
bookstack_host: localhost
bookstack_db_name: bookstackdb
bookstack_db_user: bookstackuser
bookstack_db_password: <unser sicheres password für die db>
bookstack_lang: de
...



Das password anpassen und die Sprache auf de umstellen.
```

Datei speichern fertig.
Nun noch eine inventory Datei anlegen.
In die wird der Server eingetragen auf den dann das BookStack installiert werden soll, sprich das ansible Script / Playbook ausgeführt werden soll.

```
nano ~/ansible/inventory.ini
```

Folgender Inhalt, der parameter ansible_user= lehgt den Benutzer fest mit dem die Installation auf dem Server ausgeführt werden soll, hier root

```
192.168.178.210 ansible_user=root
```

Denn Ansible ist im Verzeichnis ~/.local/bin/ installiert

Nun kann das Playbook gestartet werden.

```
ansible-playbook bookstack.yml -i invetory.ini
```

Installation abgeschlossen.

Installation via Docker image

Beschreibung:

Hier installation via Docker mit composer file.

Installation.

Die Verzeichnisse erstellen und die Composter Datei.

Im root Verzeichnis ein neues Verzeichnis erstellen.

```
mkdir -p /root/bookstack
```

Ins Verzeichnis wechseln und eine neue datei anlegen mit .env anlegen.

```
cd /root/bookstack
nano /root/bookstack/.env
```

Inhalt

```
# Domain
DOMAIN=wiki.example.com

# BookStack Datenbank Variablen
DB_HOST=db
DB_DATABASE=bookstack
DB_USERNAME=bookstack_user
DB_PASSWORD=sicheres_passwort

# BookStack App Variablen
APP_URL=https://${DOMAIN}
#der app key kann in der console mit folgendem Befehl generiert werden
#echo "base64:$(openssl rand -base64 32)" Ausgabe :
base64:0HxGQ72frdcVZ+XyZQ1Q0Kr6FkFGwJS/UeYT/NLtZTo=
APP_KEY=base64:0HxGQ72frdcVZ+XyZQ1Q0Kr6FkFGwJS/UeYT/NLtZTo=
```

Nun eine angepasste php.ini für das Upload limit, hier 10 GB kann nach beliben geändert werden.
In der Datei alles was 10G ist mit eigenem Wert überschreiben

```
nano /root/bookstack/php.ini
```

Inhalt

```
upload_max_filesize = 10G
post_max_size = 10G
memory_limit = 512M
max_execution_time = 300
max_input_time = 300
```

Nun die composer Datei

```
nano /root/bookstack/docker-compose.yml
```

Inhalt, achtung bei phpmyadmin die interne ip anpassen, für den fall das das bookstack auch public erreichbar ist
Denn phpmyadmin solls es ja nicht sein

```
version: '3.8'

services:
  app:
    image: linuxserver/bookstack:latest
    container_name: bookstack_app
    environment:
      - DB_HOST=${DB_HOST}
      - DB_DATABASE=${DB_DATABASE}
      - DB_USERNAME=${DB_USERNAME}
      - DB_PASSWORD=${DB_PASSWORD}
      - APP_URL=${APP_URL}
      - APP_KEY=${APP_KEY}
      - APP_LANG=de
    volumes:
      - ./bookstack_data:/config
      - ./php.ini:/etc/php7/conf.d/99-custom.ini  # Mounte das php.ini-File
    depends_on:
      - db
    restart: always
```

```
  db:
    image: mariadb:10.6
    container_name: bookstack_db
    environment:
      - MYSQL_ROOT_PASSWORD=root_passwort
      - MYSQL_DATABASE=${DB_DATABASE}
      - MYSQL_USER=${DB_USERNAME}
      - MYSQL_PASSWORD=${DB_PASSWORD}
    volumes:
      - ./bookstack_db:/var/lib/mysql
    restart: always

  caddy:
    image: caddy:latest
    container_name: bookstack_caddy
    ports:
      - "80:80"
      - "443:443"
    restart: always
    environment:
      - CADDY_DOMAIN=${DOMAIN}
    volumes:
      - ./caddy_data:/data
      - ./caddy_config:/config
    command: caddy reverse-proxy --from ${DOMAIN} --to app:80

  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    container_name: phpmyadmin
    restart: always
    ports:
      - "172.0.2.2:8080:80"                    # phpMyAdmin wird unter http://172.0.2.2:8080 erreichbar sein
    environment:
      PMA_HOST: db                 # Der Hostname der Datenbank (muss mit dem Service-Namen übereinstimmen)
      PMA_USER: ${DB_USERNAME}             # Standard-Benutzername (optional, da phpMyAdmin Login-Maske
hat)
      PMA_PASSWORD: ${DB_PASSWORD}         # Standard-Passwort (optional)
      UPLOAD_LIMIT: 512M
    depends_on:
```

```
    - db
```

Die container starten

```
docker-compose up -d
```

Firewallregeln für Public nutzung hinzufügen.

ufw einrichten für docker

HTML Formatierungen, da BookStack nicht so viele Module wie Confluence hat

HTML Formatierungen, da BookStack nicht so viele Module wie Confluence hat

InfoBox

Beschreibung:

Eine Infobox die in der HTML Code Ansicht eingefügt werden kann:

  Hinweis: Dies ist ein wichtiger Hinweis.

Quelltext dazu:

```
<div style="border: 1px solid #ffc107; background-color: #fff3cd; padding: 15px; margin-bottom: 20px;">
  <strong>Hinweis:</strong> Dies ist ein wichtiger Hinweis.
</div>
```

Wartung und Bedienung

Terminal Cli Befehle

Beschreibung:

Einige Dinge kann man in Bookstack nur über php programme einrichten im Terminal.
Das php programm nennt sich artisan und ist in der BookStack installation zu finden.

Der standard Pfad lautet:

```
/var/www/Bookstack
```

Also würde ein aufruf

```
cd /var/www/Boackstack
php artisan parameter
```

Die Befehle:

Create an Admin User

Create a new admin user via the command line. Can offer a good last resort if you ever get locked out the system. Will use the details provided as options otherwise will request them interactively.

```
# Interactive usage
php artisan bookstack:create-admin

# Non-interactive usage example
php artisan bookstack:create-admin --email="barry@example.com" --name="Bazza" --password="hunter2"

# Defining "External Authentication ID" instead of password for LDAP/SAML2/OIDC environments
php artisan bookstack:create-admin --email="barry.booker@example.com" --name="Bazza" --external-auth-id="bbooker"
```

Copy Shelf Permission

By default shelf permissions will not auto-cascade since a book can be in many shelves. This command will copy the permissions of a shelf to all child books. This can be done for a single shelf or for all shelves in the system:

```
# Run for all shelves
php artisan bookstack:copy-shelf-permissions --all

# Run for a single shelf
php artisan bookstack:copy-shelf-permissions --slug=my_shelf_slug
```

Update System URL

BookStack will store absolute URL paths for some content, such as images, in the database. If you change your base URL for BookStack this can be problematic. This command will essentially run a find & replace operation on all relevant tables in the database. Be sure to take a database backup for running this command.

```
# Searches for <oldUrl> and replaces it with <newUrl>
php artisan bookstack:update-url <oldUrl> <newUrl>

# Example:
php artisan bookstack:update-url http://docs.example.com https://demo.bookstackapp.com
```

Reset User MFA Methods

This will reset/clear any existing multi-factor-authentication methods for the given user. If MFA is enforced for one of their roles they’ll be prompted to reconfigure MFA upon next login.

```
# Via email address:
php artisan bookstack:reset-mfa --email=john@example.com

# Via system ID:
php artisan bookstack:reset-mfa --id=5
```

Refresh User Avatars

This will re-fetch avatar images for users in the system. By default avatars are fetched from Gravatar unless an alternative avatar system has been configured.

*Note: This will **not** load avatars from any connected authentication system.*

```
# Refresh avatar for just the user of the given email
php artisan bookstack:refresh-avatar --email=user@example.com
# Or the user of the given id
php artisan bookstack:refresh-avatar --id=101

# Refresh avatars for all users without avatars already assigned
php artisan bookstack:refresh-avatar --users-without-avatars

# Refresh avatars for all users in the system
php artisan bookstack:refresh-avatar --all
```

Delete All Activity History

This will clear all tracked activities in the system. Note: Some areas of the interface rely on this data, including the Audit Log and any “Recent Activity” lists.

```
php artisan bookstack:clear-activity
```

Delete Page Revisions

By default this deletes all page revisions apart from page update drafts which share the same system. A `-a` flag can be used to also delete these update drafts.

```
# Delete just the page revisions
php artisan bookstack:clear-revisions

# Delete all page revisions from the system including update drafts
php artisan bookstack:clear-revisions -a
```

Delete Page Views

Delete tracked content views from the system. These are primarily used to populate any “Recently Used” lists.

```
php artisan bookstack:clear-views
```

Cleanup Unused Images

Searches and removes images that are not used in page content. While this can be done in the “Maintenance” section of the interface, running this via the command-line is often safer to avoid timeouts.

```
php artisan bookstack:cleanup-images
```

Regenerate the Search Index

BookStack uses a custom database-based search index system for efficient querying within the system. The command below re-builds this search index. This does not usually need to be manually ran, but it can be useful if manually inserting pages into the system or to pick-up BookStack indexing changes.

```
php artisan bookstack:regenerate-search
```

Regenerate Access Permissions

BookStack pre-calculates and stores certain access permissions in the database so that access can be calculated in a performant manner. The below command will regenerate these permissions. This is primarily used in development but can be useful if manually adding content via the database.

```
php artisan bookstack:regenerate-permissions
```

Regenerate Reference Index

BookStack stores references between content within the system to track how content is interlinked. Such references are generally managed by BookStack upon certain actions, such as when saving a page, but in some cases the below command may help to re-index these references. This can be useful upon upgrade of old content, or when manually adding content via the database.

```
php artisan bookstack:regenerate-references
```

Delete Users

Delete all users from the system that are not original “admin” or system-level users.

```
php artisan bookstack:delete-users
```

Generate UTF8mb4 SQL Upgrade Commands

Generates out the SQL which can be used to upgrade the database to UTF8mb4. See UTF8mb4/Emoji Support for further details.

```
php artisan bookstack:db-utf8mb4
```

Backup and Restore

Beschreibung:

Bookstack besteht aus Dateien und Der Datenbank

Hier liegen die Dateien bzw Verzeichnisse

- `.env` - File, contains important configuration information.
- `public/uploads` - Folder, contains any uploaded images.
- `storage/uploads` - Folder, contains uploaded page attachments.
- `themes` - Folder, contains any configured visual/logical themes.

Backup Manuell:

Die Dateien sichern.

In das Verzeichnis gehen bei ner direkt installation:

```
cd /var/www/Bookstack
```

Bei ner docker installtion reicht das ganze Projektverzeichnis, das der Vorteil von der Docker.
nun ein archiv erstellen

```
tar -czvf bookstack-files-backup-$(date +"%Y-%m-%d_%H-%M-%S").tar.gz .env public/uploads storage/uploads themes
```

Nun die Datenbank sichern

```
mysqldump -u root bookstackdb > bookstack.backup-$(date +"%Y-%m-%d_%H-%M-%S").sql
```

Diese beiden Datein ins root Verzeichnsi vom neuen Server kopieren via scp zum Beispiel

Restore/Migration:

Manuell:

Im Docker container, weil direkt installation machen wir in Zukunft nicht mehr:
Falls der Conatiner schon unglücklicherweise gestartet wurde den container app stoppen

```
docker-compose stop app
```

Das Datenverzeichnis löschen

```
rm -r /root/bookstack/bookstack_data
```

Nun in phpmyadmin alle Tabellen löschen, falls der Conateiner schon gestartet wurde.

Nun die Datenbank restoren via phpmyadmin

Nun den app key aus der alten instance in die .env Datei eintragen.
in einer nicht Docker installation leigt diese dann unter

```
cat /var/www/Boockstack/.env
```

Ein bisschen hochscrollen da steht der APP_KEY diesen kopieren und in die neue .env einfügen / ersetzten

Nun die tar Datei hochladen die wir im Punkt backup erstellt haben

```
scp bookstack-files-backup-2025-02-02_14-23-51.tar.gz root@<ip>:/root/bookstack
```

Nun das archiv die Bilder entpacken nach bookstach_data

```
tar -xf bookstack-files-backup-2025-02-02_14-26-12.tar.gz  --strip-components=1 -C
/root/bookstack/bookstack_data public/images
```

Nun die Files/attachments entpacken

```
tar -xzf bookstack-files-backup-2025-02-02_14-23-51.tar.gz --strip-components=2 -C
/root/bookstack/bookstack_data/www/ storage/uploads
```

Nun die Pfade eventuell anpassen wenn die URL sich geändert haben sollte, sonst diesen Schritt überspringen

Dazu in die instance einloggen

```
docker-compose exec app bash
```

Nun ins Verzeichnis von bookstack gehen

```
cd /app/www
```

Nun mittels php artisan die Urls aktualisieren, als erstes die alte URL als zweiten Parameter die neue.
Falls die alte URL einen port mit dran ahtte zum docs.example.com:3000 dann diesen auch so angeben

```
# Searches for <oldUrl> and replaces it with <newUrl>
php artisan bookstack:update-url <oldUrl> <newUrl>

# Example:
php artisan bookstack:update-url http://docs.example.com https://demo.bookstackapp.com
```

Nun noch den cache löschen

```
php artisan cache:clear
```

Fertig

Nachbearbeitung bei Manuell wie auch mit Backup Tool wenn das Ziel KEIN Docker ist:

Sollte nicht auf die gleiche Version migriert werden nachdem der Container gestartet ist, die migration ausführen das die Datenbank angepasst wird.

Dzu einmal in den container einloggen

```
docker-compose exec app bash
```

Dann diese Befehler ausfürhen

```
cd /app/www
php artisan migrate
```

Ausgabe:

```
php artisan migrate
```

APPLICATION IN

PRODUCTION.

┌ Are you sure you want to run this command? ──────────┐
│ ● Yes / ○ No   │
└──┘

bestätigen