

Wikis, Documentation, and Blogs

- [WikiJS](#)
 - [WikiJS, Open Source, Powerful, Configurable, Intuitive](#)
- [WriteFreely](#)
 - [Install WriteFreely](#)
- [Bookstack](#)
 - [Install and Setup Bookstack, an Open Source Wiki](#)

Wikijs

WikiJS, Open Source, Powerful, Configurable, Intuitive

https://www.youtube.com/embed/Dd8_plibBYk

WikiJS is a tremendously powerful, extremely full-featured open source, self hostable Wiki system. It has a modern user interface, and tons of settings, configuration, and customization options.

Installing it with Docker-CE and Docker-Compose makes it a breeze to get WikiJS up and running in no time.

Today, we'll go through setting up WikiJS, and the basics of configuration and usage.

What you'll need

- Docker-CE
- Docker-Compose
- (optional) Portainer-CE (GUI for Docker)
- (optional) NGinX Proxy Manager (for full url access and SSL)
- About 15 minutes of your time.

Installing Docker-CE, Docker-Compsoe, Portainer-CE, and NGinX Proxy Manager

In the video, I show you how to create a user with sudo privileges, so you aren't setting up these things as root. It only takes a minute to do, so definitely take that step if you haven't already.

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://raw.githubusercontent.com/bmcgonag/docker_installs/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

```
./install_docker_nproxyman.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you'll be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of "sudo" in front of them.

Installing WikiJS

Now that we have Docker-CE and Docker-Compose installed, we can move forward with getting WikiJS installed and setup. First, let's create a folder structure to help keep our docker applications organized and easy to backup.

Let's create a "docker" folder to be the top level (parent) folder. If you already have an organizational structure for your docker installs, then don't worry about this part.

And we'll move into that parent level folder:

```
mkdir docker
```

```
cd docker
```

Now, let's make a directory for our WikiJS install and move into it.

```
mkdir wikijs
```

```
cd wikijs
```

Now that we are inside our "wikijs" folder, we'll create a file called "docker-compose.yml"

```
nano docker-compose.yml
```

We need to copy the following yaml code block, and paste it into the "docker-compose.yml" file we have open for editing. So, highlight the code-block below, and copy it.

```
version: "3"
services:
  wikijs:
    image: lscr.io/linuxserver/wikijs
    container_name: wikijs
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=America/Chicago
    volumes:
      - ./config:/config
      - ./data:/data
    ports:
      - 3000:3000
    restart: unless-stopped
```

Paste the code-block into the open nano editor with a right-click, paste, or with the hotkey combination CTRL + Shift + V.

In the file, you'll want to note a few places, where you may want to make changes.

First, let's look at the port mapping. 3000 is a very common port for lots of applications, so it's generally a good idea to change this to some other port on your host machine. The "host" machine is the physical system you are putting the dockerized application on. The "container" is the application space where the app runs (separated / sandboxed away from the physical OS). The port and volume mappings are setup as

```
<host>:<container>
```

So, feel free to change the host side of a volume mapping or port mapping in the docker-compose.yml file. Just do not change the container side unless you are a developer, and know what you are doing.

For my setup, we used port 9190 instead of port 3000, so my port mapping looked like:

```
ports:
  - 9190:3000
```

Additionally, if you are using the stacks feature in Portainer-CE to run this, instead of a docker-compose.yml file, change the volume mappings to use the absolute (full) path to the wikijs folder we created, instead of the relative `./config` and `./data` paths we have in the docker-compose.yml

file here.

In my case, for a Portainer-CE stack, I would have my volume mappings set like:

volumes:

- /home/brian/docker/wikijs/config:/config
- /home/brian/docker/wikijs/data:/data

Next, you'll want to make sure your PGID and PUID are set correctly. You can check this, by opening another terminal window (or SSHing into your host machine with a second terminal), and entering the command:

```
id
```

Then comparing the values in the output to the values in the docker-compose.yml file.

Finally, set your timezone correctly for the TZ variable.

Once everything is set in your docker-compose.yml file, save it with Ctrl + O, then press enter to confirm, and close the nano editor with CTRL + X.

Now, we are ready to run the docker-compose file and pull down the WikiJS image, and start the container.

Just enter:

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

In your terminal (from inside the ./docker/wikijs directory), and allow the image to download, and start.

When the container is done starting you should see `done` in the terminal. Make sure you don't see any errors in the terminal output. If everything went well, you should now be able to use a web browser to go to your host machine's IP and the port you entered in the docker-compose.yml file, and be presented with the first run wizard of WikiJS.

Check out the video linked at the top of this article to get details on the basic configuration and usage of WikiJS, as well as how to setup your Wiki with a FQDN (Fully Qualified Domain Name) and LetsEncrypt SSL Certificate.

Support my channel and content

Support my Channel and ongoing efforts through Patreon:

<https://www.patreon.com/bePatron?u=234177>

WriteFreely

A clean, simple blogging and journaling platform

Install WriteFreely

<https://www.youtube.com/embed/pR4QATSRyXA>

Sometimes you just need a writing tool that will allow you to get to work. Not something that will take you hours of setup and configuration, and theming, and layout, and on, and on, and on.... My friends, WriteFreely is an open source, self hosted platform that does just that. We'll set this up using a nice docker image a community member has created, and we'll make our time worthwhile once again.

Whether you are documenting the steps you use to change the oil on your car, or set the alarm in your home, or setup an entire domain with users and VLANs, and VPNs, and neighborhood hotspots, the most important part is just starting. Sometimes setting up software to help document it all can be the real hindrance to getting any project off the ground. With WriteFreely, we'll remove that hindrance.

What you'll need

- Docker and Docker Compose
- (Optional) NGinX Proxy Manager or some other reverse proxy application.
- (Optional) A Domain Name that you own and can setup DNS A-records for
- About 20 minutes of your time

Installation

Installation of Docker and Docker Compose via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://gitlab.com/bmcgonag/docker\_installs/-/raw/main/install\_docker\_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

```
./install_docker_nproxyman.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

Install and Configure Write Freely

Let's start by creating our folder structure for our installation. I like to have a parent level folder called "docker" that I then put all of my docker applications inside of. each in their own folders.

```
mkdir -p docker/write-freely
```

Now let's move into our new folder with

```
cd docker/write-freely
```

want to create our docker-compose.yml file that tells docker which image(s) to pull, and how to run our application.

```
nano docker-compose.yml
```

Now, use copy / paste to paste the following code block into that new file:

```
Version: '3.3'
services:
  write-freely:
    image: nephatrine/write-freely:latest
    container_name: write-freely
    environment:
      TZ: America/New_York
      PUID: 1000
```

```
PGID: 1000
ports:
- "70:70/tcp"
- "8080:8080/tcp"
volumes:
- ./write-freely:/mnt/config
```

In the above code block, you'll want to adjust a few items:

- Make sure you set the TZ: variable to your local timezone. In my case, I changed it to 'America/Chicago'
- on the ports, you'll likely want to change the left side port number from 8080 to some other, less common port. In my case I used 8250, so my mapping looked like this when done: '8250:8080/tcp'
- If you know your user id and group id are not 1000 on the system you are installing on, then you'll want to change those numbers to correspond to your user id and group id numbers as well, but most likely 1000 is correct.

Now save the file with CTRL + O, then press Enter to confirm, and use CTRL + X to exit the nano editor.

We are finally ready to run our application.

```
docker compose up -d
```

Once you run the above command, you'll see the image being pulled down, and the container being started up. Wait about 20 seconds or so after you are returned to the normal terminal prompt, then open your favorite, modern, web browser, and browse to the ip address of your host machine / server where you are running the application, and the port number you used on the left side of the second port mapping above.

In my setup I went to <http://192.168.10.60:8250>

You should see the main default page of WriteFreely. IF so, that's great, but we still have a bit of work to do.

Create an Admin User

Now, let's create an admin user. The author gives us a nice, and simple little command to do this, but first we need to setup a special configuration file for WriteFreely. In my video I did this through the mapped local volume, but here we'll do it through the docker container bash terminal.

To enter the bash environment of your running container enter the following command. Notice that the prompt will change when you do.

```
docker exec -it write-freely /bin/bash
```

Now, let's move into the folder where we need to copy the .ini file already generated.

```
cd /mnt/config/etc
```

Next, we'll create a new folder called 'writefreely':

```
mkdir writefreely
```

And we'll copy the existing file 'writefreely.ini' to this folder, renaming it to 'config.ini'

```
cp writefreely.ini ./writefreely/config.ini
```

Finally, let's edit our new config.ini file

```
nano ./writefreely/config.ini
```

Again, in the video there were a couple of things I said not to change, but we will actually want to change them if you are going to access your writefreely install from a fully qualified domain name (outside your own network).

```
[server]
hidden_host      =
port            = 8080
bind            = 0.0.0.0
tls_cert_path    =
tls_key_path     =
autocert         = false
templates_parent_dir = /var/www/writefreely-live
static_parent_dir  = /var/www/writefreely-live
pages_parent_dir   = /var/www/writefreely-live
keys_parent_dir    = /mnt/config/data
hash_seed        =
gopher_port       = 70
```

```
[database]
type    = sqlite3
filename = /mnt/config/data/writefreely.db
username =
password =
database =
host     = localhost
port     = 3306
tls      = false
```

[app]

```
site_name      = Blog Site # <-- Change this to your desired site title and remove this comment
site_description = A Blog Site # <-- Change this to your desired site subtitle and remove this comment
host           = http://localhost:8080/ # <-- Change localhost to your domain name and remove this
comment. e.g. http://myblog.example.com

theme          = write
editor         =
disable_js     = false
webfonts       = true
landing        =
simple_nav      = false
wf_modesty    = true
chorus         = false
forest         = false
disable_drafts = false
single_user    = false
open_registration = false
open_deletion  = true
min_username_len = 3
max_blogs      = 3
federation     = true
public_stats   = true
monetization   = false
notes_only     = false
private        = false
local_timeline = true
user_invites   = user
default_visibility = public
update_checks  = false
disable_password_auth = false
```

[oauth.slack]

```
client_id      =
client_secret  =
team_id        =
callback_proxy =
callback_proxy_api =
```

[oauth.writeas]

```
client_id      =
client_secret  =
auth_location  =
token_location =
inspect_location =
callback_proxy =
callback_proxy_api =
```

[oauth.gitlab]

```
client_id      =
client_secret  =
host           =
display_name   =
callback_proxy =
callback_proxy_api =
```

[oauth.gitea]

```
client_id      =
client_secret  =
host           =
display_name   =
callback_proxy =
callback_proxy_api =
```

[oauth.generic]

```
client_id      =
client_secret  =
host           =
display_name   =
callback_proxy =
callback_proxy_api =
token_endpoint =
inspect_endpoint =
auth_endpoint  =
scope          =
allow_disconnect = false
map_user_id    =
map_username   =
map_display_name =
map_email      =
```

I have identified the bits you'll want to change with a comment in the code block above. Make sure to make the necessary changes, then remove my comment.

You can edit in the VI editor by pressing `I` one time to go into insert mode. When done editing press `Esc` to get back out of editing mode.

Save the changes and exit the VI editor by pressing `:wq`.

Oddly, this container wants the `config.ini` and the `writefreely.ini` files to be the same, as it seems to use both. So, let's fix that real quick as well.

```
cp ./writefreely/config.ini ./writefreely.ini
```

Next, we'll run the command to create our new administrative user:

```
writefreely -c /mnt/config/etc/writefreely/config.ini --create-admin [username]:[password]
```

Replace the `[username]` with the username you want, and the `[password]` with a long, strong password. My command looked like this:

```
writefreely -c /mnt/config/etc/writefreely/config.ini --create-admin brian:aReallyLongStr0n6Pa55w0rd
```

Once run, you should see a success message. If so, we are done in the docker container.

First, exit the container with `exit`.

Now, let's restart our docker compose file.

```
docker compose restart
```

And allow the container to restart.

Awesome! Now, refresh your browser, and click the Log In button at the top of the screen. Enter your new administrative credentials to log in.

You can now start writing in your personal blog location, or you can click around a bit. Note, you do have a draft location that allows you to write and save, but won't publish your work publicly until you move it to your personal blog location.

Setup A Reverse Proxy

If you want to access your new site from outside of your home, you'll want to setup a reverse proxy to make sure requests for the site are routed properly to this docker container. I use NGinX Proxy Manager, but you are welcome to use any reverse proxy you are more comfortable with.

In NPM (NGinX Proxy Manager) click Add Proxy Host and in the new pop up window enter the domain name of your writefreely install.

You need to own the main domain, and have an A Record pointing to your public IP address. This is important if you want this to function properly.

I entered ytdiy.routemehome.org. I own the domain routemehome.org, and have an A-record already setup to point to the public IP address of my server. My firewall is configured to forward all incoming requests on port 80 and / or 443 to the server running NGinX Proxy Manager. NPM then handles the routing of the requests for my different services to the proper docker container / server.

Next, enter the local IP address of your server running WriteFreely into the IP field, and the port you mapped on the left side of the 8080 port mapping into the port field. I entered 192.168.10.60 and 8250 respectively.

Next, tick the options for Block Common Exploits, and Websocket Support.

Now select the SSL tab, and choose Request a New Certificate from the drop down, then tick the options for Force SSL, HTTP/2 Support, and optionally HSTS options as well. Make sure your email is filled in, and tick the option to accept the LetsEncrypt terms of service.

Click Save. The pop-up should just disappear if everything is setup correctly, and you should have a new entry in your proxy list for your new WriteFreely install.

You can click on the domain name in the list, and make sure it opens properly to your WriteFreely page. Login, and you're ready to start writing.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

<https://www.patreon.com/awesomeopensource>

Bookstack

Install and Setup Bookstack, an Open Source Wiki

Bookstack is an incredibly powerful, open source, self hosted wiki solution. It's set apart from other wiki solutions by the way it allows you to organize content. Think of it as your digital library of documentation.

You start with Shelves (a larger overall categorization) of various documentation. Then on those shelves you have your books. Books are the next level down of categorizing things. Within books you have chapters, again a level to more closely gather like information. And within those chapters you have pages. Pages are where the specific document lives.

Here's is an example, but certainly not the only way to use this organizational hierarchy.

Shelf 1: Networking -> Book 1: Networking Tools -> Chapter 1: IPv4 Tools - Page: Octets

We can repeat the entire line above except the page, and add a new page called "Network Mask".

Shelf 1: Networking -> Book 1: Networking Tools -> Chapter 1: IPv4 Tools - Page: Network Mask

And so on as we build out our wiki.

Here's the coolest part. You don't have to keep the Networking Tools book in just 1 shelf. You can have it on as many shelves as it matches. It's like having multiple copies of the book in different locations in your library.

Shelf 1: Networking

Shelf 2: VPNs

Shelf 3: Dynamic DNS

These could all be a home for a book called "Networking Tools". This guarantees a much higher percentage chance that you'll find the appropriate material when you go digging through topics that may be related, but not fit into quite the same top level category. OF course, you could take the topics of VPNs and Dynamic DNS and just as easily make them Books in the Networking shelf. There is no wrong way to organize things.

Installing and Running Bookstack

What You'll Need

- A server, either bare metal, VPS, or a VM or Container such as LXC, LXD, or Incus
- Docker and Docker Compose installed on that server
- (optional) A public IP address
- (optional) A Domain or Subdomain name for your installation
- (optional) a Reverse Proxy if running inside a LAN
- About 15 minutes of your time
 - * Optional items above are only if you want to expose your Wiki to the public internet.

Installation via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget -O install-docker.sh https://gitlab.com/bmcgonag/docker\_installs/-/raw/main/install\_docker\_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker.sh
```

and then run the script with the command:

```
./install_docker.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

Install and Setup Bookstack in Docker using Docker Compose

First, let's create a folder structure which will support a solid backup strategy for our containers. We'll make a directory structure with a top-level directory called `docker`. Inside that directory we'll then create a directory for each service or application we want to run. In this case that second level directory will be called `"bookstack"`. We can make both directories with a single command, and this command gives the added benefit, that if either directory already exists, it will just use that directory.

`mkdir -p docker/bookstack`

From the man pages of the `mkdir` command:

`-p, --parents`
no error if existing, make parent directories as needed

Next, we'll move into that directory with the command:

```
cd docker/bookstack
```

Inside the `'bookstack'` directory we need to create one more directory called `'data'`. This is where the container will house and persist the data and database for our docker container. This is a very important folder, and should be backed up regularly in order to preserve your bookstack data should anything catastrophic happen to the installation.

```
mkdir data
```

Now, let's create our file to tell Docker Compose how to bring up and configure our Bookstack Installation. I'm actually going to add two versions of the file below.

The first version will be for those wanting to use the built in Bookstack authentication system (login with username and password). The second will be for those who already have an Identity Provider setup like Authentik, Authelia, Keycloak, etc. and want to use OpenIDConnect (OIDC) to login to their Bookstack with a single sign on (SSO) method.

Create a new file in your `'docker/bookstack'` folder, and call it `'compose.yaml'` using this command:

```
nano compose.yaml
```

Now copy the yaml code from the version you need below, and paste it into your text editor. For the terminal you can paste by right-clicking with the mouse and selecting `'Paste'`, or you can use the `CTRL + Shift + V` hotkey combination on your keyboard (Linux / Windows), or `CMD + V` for MacOS.

`compose.yaml` to use the built in Login with Bookstack

services:

bookstack:

image: lscr.io/linuxserver/bookstack

container_name: bookstack

environment:

- TZ=America/Chicago
- PUID=1000 # check this value with the command 'id'
- PGID=1000 # check this value with the command 'id'
- APP_URL="https://books.yourgreatdomain.com"
- DB_HOST=bookstack_db
- DB_USER=bookstack
- DB_PASS=a-different-long-strong-password-with-a-lot-of-numbers-and-letters
- DB_DATABASE=bookstackapp
- MAIL_DRIVER=smtp
- MAIL_HOST=smtp.youremaildomain.org
- MAIL_PORT=587
- MAIL_USERNAME=you@youremaildomain.org
- MAIL_PASSWORD=your-smtp-email-password
- MAIL_ENCRYPTION=tls
- MAIL_FROM=you@youremaildomain.org

volumes:

- ./data:/config

ports:

- 80:80

restart: unless-stopped

depends_on:

- bookstack_db

bookstack_db:

image: lscr.io/linuxserver/mariadb

container_name: bookstack_db

environment:

- PUID=1000
- PGID=1000
- MYSQL_ROOT_PASSWORD=a-long-strong-password-with-numb3rs-and-le77er5
- TZ=America/Chicago
- MYSQL_DATABASE=bookstackapp
- MYSQL_USER=bookstack
- MYSQL_PASSWORD=a-different-long-strong-password-with-a-lot-of-numbers-and-letters # must match

the one in the above section

volumes:

- ./data:/config

restart: unless-stopped

Next we have the version to use if you have your own IdP to authenticate through. If you don't know what this means, you probably should use the first compose file above.

compose.yaml for those wanting OIDC for SSO

services:

bookstack:

image: lscr.io/linuxserver/bookstack

container_name: bookstack

environment:

- TZ=America/Chicago
- PUID=1000 # check this value with the 'id' command
- PGID=1000 # check this value with the 'id' command
- APP_URL="https://docs.yourgreatdomain.org"
- DB_HOST=bookstack_db
- DB_USER=bookstack
- DB_PASS=a-long-strong-password-with-lots-of-numbers-and-letters
- DB_DATABASE=bookstackapp
- MAIL_DRIVER=smtp
- MAIL_HOST=smtp.yourgreatdomain.org
- MAIL_PORT=587
- MAIL_USERNAME=you@yourgreatdomain.com
- MAIL_PASSWORD=your-long-strong-email-password-for-the-smtp-server
- MAIL_ENCRYPTION=tls
- MAIL_FROM=you@yourgreatdomain.org
- # Set OIDC to be the authentication method
- AUTH_METHOD=oidc
- # Control if BookStack automatically initiates login via your OIDC system
- # if it's the only authentication method. Prevents the need for the
- # user to click the "Login with x" button on the login page.
- # Setting this to true enables auto-initiation.
- AUTH_AUTO_INITIATE=false # feel free to change it to 'true'

```
# Set the display name to be shown on the login button.
# (Login with <name>)
- OIDC_NAME=Authentik # whatever you want the login button to say
# Name of the claims(s) to use for the user's display name.
# Can have multiple attributes listed, separated with a '|' in which
# case those values will be joined with a space.
# Example: OIDC_DISPLAY_NAME_CLAIMS=given_name|family_name
- OIDC_DISPLAY_NAME_CLAIMS=name
# OAuth Client ID to access the identity provider
- OIDC_CLIENT_ID=the-client-id-from-your-auth-server
# OAuth Client Secret to access the identity provider
- OIDC_CLIENT_SECRET=the-auth-secret-from-your-auth-server-for-the-oidc-provider-you-created-ahead-
```

of-time

```
# Issuer URL
# Must start with 'https://'
- OIDC_ISSUER=https://auth.youroidcprovider.org/application/o/bookstack-oidc/
# Enable auto-discovery of endpoints and token keys.
# As per the standard, expects the service to serve a
# `<issuer>/.well-known/openid-configuration` endpoint.
- OIDC_ISSUER_DISCOVER=true
```

volumes:

```
- ./data:/config
```

ports:

```
- 80:80
```

restart: unless-stopped

depends_on:

```
- bookstack_db
```

bookstack_db:

image: lscr.io/linuxserver/mariadb

container_name: bookstack_db

environment:

```
- PUID=1000
- PGID=1000
- MYSQL_ROOT_PASSWORD=a-different-long-strong-password-with-lots-of-numbers-and-letters
- TZ=America/Chicago
- MYSQL_DATABASE=bookstackapp
- MYSQL_USER=bookstack
- MYSQL_PASSWORD=a-long-strong-password-with-lots-of-numbers-and-letters # must match the one in
```

the section above

volumes:

- ./data:/config

restart: unless-stopped

Regardless of the yaml you choose to setup, you need to make sure to update several of the environment variables, and potentially the ports that you'll access the service on.

You can modify the port on the left side of the port mapping in the yaml. The left side of the mapping is the port where the service is accessed on the host machine. 80 is the standard web server port, so I suggest if you are running this inside a LAN, you change the port to something non-standard, and then use a reverse proxy to access the service as needed from outside the LAN. A port like 8089 would be good for instance. As long as no other service is using port 8089 on the host machine, this should not cause an issue. If there is already a service using 8089, then you can use any valid port number you like which is not already in use.

Additionally, you'll need to change the passwords for the database connections. Use a very long, strong, password, and make sure the password entries for the variables `MYSQL_PASSWORD` and `DB_PASS` match exactly.

You'll need to update the email / smtp settings if you want your install to be able to send emails to you as well. If not, feel free to leave the placeholder info there.

Once you've made the necessary changes, save the file with `CTRL + O`, then press `Enter` to confirm. This is a good time to briefly go back through the file, check your spacing (as yaml is space dependent), and also make sure you have entered everything correctly.

Exit the nano text editor with `CTRL + X`.

Bring Up our Bookstack Install