Tiki Wiki installation on Linux 2024 (Debian 12)

Bernard Sfez - 2024-04-17 09:45

Through this comprehensive guide learn how to setup your Debian 12 server and everything needed to install and maintain multiple Tiki Wiki websites or applications. This concise tutorial covers the entire process, from upgrading your Debian Linux operating system to publish your TikiWiki website. This step-by-step walkthrough will cover MariaDB, installing different PHP versions, installing Webmin and Virtuamin control panel, setup and use Git, set an SSL certificat and finally publish your Tiki Wiki website online.

This tutorial is tailored for intermediate IT administrators or full-stack developers seeking guidance on setting up and publishing websites.
We will learn how to setup and configure:

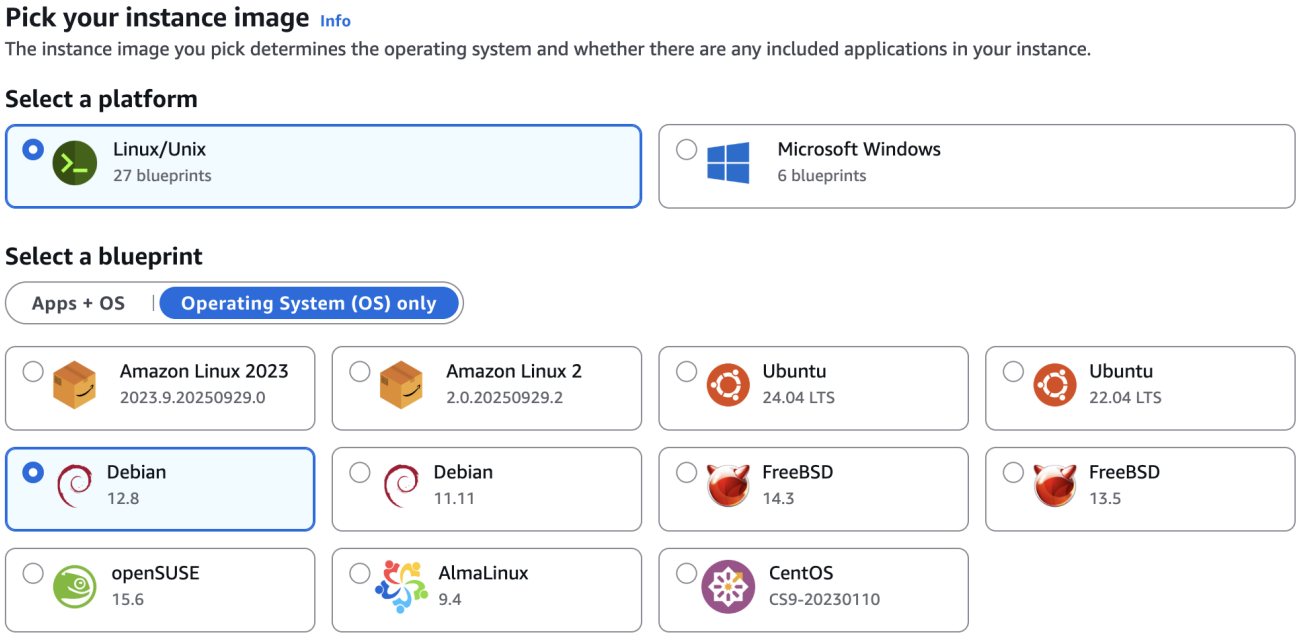
- Linux operating system, Debian12 and Apache2
- MariaDB (InnoDB)
- Virtualmin web hosting control panel
- PHP
- Git and the Tiki Gitlab repository
- All-in-one Web Application - Tiki Wiki

To embark on this Tiki Tutorial, familiarity with shell usage and a basic understanding of server operations is recommended. However, don't hesitate to dive in and learn along the way as long as you're not working with real data initially, in most cases you have the flexibility to terminate and reinstall instances until you successfully configure your Tiki setup.
If experimentation isn't your preference, or if your time is limited and you're in a rush, you can always opt to have someone else handle the setup for you. You can hire me to do it for you or explore the the Tiki Consultants list. Rest assured, with a Tiki Specialist handling the task, you'll receive a professional job well done.

Server preparation

Install an Amazon Lightsail Instance

Log into your AWS Console (or create you account) and look for Lightsail in the search bar.
Select your instance location (the region you want the server to be set for... the closer to your targeted users the better) and pick your instance image by selecting a platform.
Following our case, pick your instance image by clicking on "Linux/Unix" blueprint choose "OS Only" and select Debian12.x.
Note: Installing Virtualmin requires a pristine OS and server tools or libraries so don't use anything. else than a core basic OS installation.



Below those selectors, you will be asked to use a new default SSH key or to change the SSH key linked to this account if one exist already.
Depending your case, select to use the default key created with your instance (you will need to download it into your computer, user/.ssh folder), upload an existing key or use an existing one for this account.
If you don't have an SSH key linked to your account you will be guide to create a new SSH key pair and help will be provided.

Choose your instance plan (I suggest you start with the cheapest as you can always upgrade your plan).
Identify your instance by giving it a name. While you can add tags, I suggest you be descriptive to avoid being lost later if you create other instances (for development or test).
Click on the orange buttons "Create instance". Your instance should be running in a minute and you should be already able to use the integrated SSH console by clicking on the terminal icon (>_)

Lightsail Networking and Security Group Setup

Configuring access to your site and the Virtualmin control panel is essential for server security. Begin by navigating to the Lightsail instance control panel and selecting Networking to manage and secure your access settings.

By default, ports 22 (SSH) and 80 (HTTP) should already be open.

1. Set the security group and click Add Rule to open additional ports:

- 443 for HTTPS (SSL-secured site access)
- 10000 for Virtualmin control panel access

2. Set Access Restrictions based on IP:
- Open ports 80 and 443 to all traffic (no IP restrictions).
- Restrict access to ports 22 and 10000 to your IP only for security. During setup, you may use the Lightsail browser-based SSH/RDP for SSH access.
- Keep port 22 open during setup but plan to restrict it to your IP afterward. For added security, consider changing SSH to a different port after setup.

3. Set Up a Static IP:
Your public IPv4 address changes when you stop and start your Lightsail instance. Create and attach a static IPv4 address to your instance to keep it from changing in the "IPv4 networking" section.
4.Apply and Test:
Reboot the instance and check IP is fixed (attached) and ports are open.
This configuration provides a secure foundation for site and server management in Lightsail, while protecting sensitive ports from

unauthorized access.

⚠ Enabling root login

You should have access to a user part of the sudoer like "admin" on AWS or able to switch to the super user (su). Enable root login only if you must.

Sudo or Set and use root login☒✓

Sudo or Set and use root login

Depending of your account settings, you may be able to use the root user or use the sudo command. If you can use the root user or sudo that's ok, you can skip this.
If you want to enable root login anyway... this is the way.

Logged as the default "admin" user. (I use nano but it is the same with VI, VIM or any editor)

Switch for the Super user (root)

```
sudo su
```

Edit ssh configuration file

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

Change `#PermitRootLogin prohibit-Password` to `PermitRootLogin yes`
and `#PasswordAuthentication no` to `PasswordAuthentication yes` and of course save your changes.
You will need to remove a small script that forbid root to use his SSH key

Edit the authorized_keys file

```
nano ~/.ssh/authorized_keys
```

The script begins with "no-port-fowarding..." you need to remove this script up to the ssh key itself that starts with "ssh-rsa...".
That way the ssh key will be accessible to the ssh login.

Restart SSHD to validate changes

```
systemctl restart sshd
```

Set the root password

```
passwd root
```

Reboot your instance

```
sudo reboot
```

You will be disconnected, reconnect and check everything is working fine when you login as root.

Login as root

```
ssh root@xx.xx.xx.xx (your instance IP)
```

Note: It is a good practice, and I highly recommend it, to disable root access on production servers once everything is in place. This can be done by setting the "PermitRootLogin" and "PasswordAuthentication" parameters to "no" in the "/etc/ssh/sshd_config" file.

First SSH Connection to Your Server/Instance

Once the instance is running, you can begin managing it. A simple way to access your server is through Lightsail's browser-based SSH/RDP, which provides direct, secure SSH access via an AWS terminal.
For basic management, this browser-based method is sufficient. However, if you encounter issues or wish to install additional applications or libraries, you may prefer connecting through an external terminal.

Connect Using SSH from a Local Terminal

You can also connect to your server using your preferred terminal application (e.g., PuTTY, macOS Terminal, or other) to gain shell access. This approach requires the public IP of your instance and your locally stored SSH key. The default administrative user set by the AWS installer is admin. The command to connect (assuming your key is saved as LightsailDefaultKeyPair-us-east-1.pem) would look like this:

SSH using the stored default key

```
ssh -i ~/.ssh/LightsailDefaultKeyPair-us-east-1.pem admin@xx.xx.xx.xx
```

Replace xx.xx.xx.xx with your instance's public IP. This method provides a more versatile connection, particularly useful for advanced configuration and software installations.

If you didn't set the SSH key during the instance creation

SSH Key pair may be complicate story if you are not oriented IT Admin.
In very short, if you select to use the default key this is what I would do (replace "LightsailDefaultKeyPair-us-east-1.pem" with your key filename).
Download the key into my computer, move the file to my ~/.ssh directory and set the correct file permissions:

Set per file permissions

```
sudo chmod 600 ~/.ssh/LightsailDefaultKeyPair-us-east-1.pem
```

Then I should be able to use the following to connect :

SSH using the stored default key

```
ssh -i ~/.ssh/LightsailDefaultKeyPair-us-east-1.pem admin@xx.xx.xx.xx
```

You can find here more documentation about this :
https://lightsail.aws.amazon.com/ls/docs/en_us/articles/amazon-lightsail-ssh-using-terminal

Checking Hostname and Host informations

Hostname information

To ensure smooth operation of VirtualMin/Webmin, it's essential to configure the hostname and Fully Qualified Domain Name (FQDN) directly on the server.
It is then common your server will be able to auto-generate the hostname.

Once logged in your server through SSH use the following command.

```
hostnamectl
```

Inside the output, the "Static hostname" should shows by default the instance local IP.
We confirm this by entering the following command that should output only the hostname:

```
hostname
```

We need it to change it for your FQDN so we can proceed to the next step (Installing Virtualmin) and we need to edit configuration files to fix the issue.
We need to change this and set it for your host name, for example: sub.yourdomain.com

```
sudo nano /etc/hostname
```

Host information

The hosts file should show your Fully Qualified Domain Name (FQDN), "mydomain" (without http or https)

To check it do the following:

```
cat /etc/hosts
```

To set it, you need to edit the "hosts" file and add the line above the existing parameters. (change for your FQDN and domain name.

```
sudo nano /etc/hosts
```

It should look like this `127.0.1.1 subdomain.domain.com subdomain` but by default this is not set of course.

Make a reboot and check everything is in place by reviewing the hostname and host informations.
If you see what is expected you are good to go to the next step, installing VirtualMin else, check the information below.

Fix host file and make it permanent☒✓
You need also to check and add your host to the hosts file if your providing company didn't set it right.
But if you do it directly (editing /etc/hosts) the changes will be reverted on the next reboot.
To make it permanent, you need to modify the template used to recreate on each reboot the hosts file.

Some instance may use a Cloud configuration file other may use a Debian host file.
You will find useful information in the top comments of `/etc/hosts`
If the file start with: "Your system has configured 'manage_etc_hosts' as True.", it means your server is using cloud-init and you need to change the value of "preserve_hostname" to true at /etc/cloud/cloud.cfg

Edit hosts.debian.tmpl

```
nano /etc/cloud/cloud.cfg
preserve_hostname: true
```

Reboot the server and check changes are permanent.

Upgrade your Debian OS

Assuming you've been provided with root access and SSH information, enabling you to connect via password or SSH Key, let's begin by ensuring our instance is up-to-date.

We want to work on the last version of the dozens of software and library your server will depend on. Log onto your server and update/upgrade it with the following commands:

```
sudo apt update
sudo apt upgrade
```

Install Webmin and Virtualmin

Virtualmin will install everything you need and it is critical the installation of Virtualmin comes first.
Else you will be warned previous tools or libraries have been installed and sometimes it means you will have to clean them manually before proceding.

We will download Virtualmin automated install script and run it. It is basically a shell script that will handle rest of the installation once executed.

Download Virtualmin install script

```
wget https://software.virtualmin.com/gpl/scripts/install.sh
```

You can do a full install but also you can do a minimal install which will spare the instance resources. For exemple where I don't need a mail server I do minimal install.

Full install of Virtualmin

```
sh install.sh
```

Minimal install of Virtualmin

```
sudo sh install.sh --minimal
```

You should see now the different components being installed one by one.
This is taking some times... Just wait for the process to complete.

It should end with:
"SUCCESS to configure at https://xxx-xxx-xxx-xxx:10000 (or https://yourFQDN:10000)."
You should be ready to login to your Virtualmin panel using your root access if your firewall is set correctly.
It happens your hosting services open only certain ports by default like 80,443,22. For virtualmin you need to also open port 10000.

Setup your root access to Virtualmin

To be able to use Virtualmin as root without enabling root login I run a new webmin passwd command to change the password and set Webmin password authentication.

```
sudo webmin passwd --user root
```

Use it to login to your new Virtualmin.
https://xxx-xxx-xxx-xxx:10000 (or https://yourFQDN:10000

Virtualmin setup and post-installation optional features.

Once you login your Virtualmin HTMLM front end (to trust the SSL certificate may be needed).
Follow Virtualmin Post installation wizard.
It is pretty straightforward and everything is done to guide you however questions may be different based on your system and the Virtualmin version. And it is usually ok to use default proposed settings.

Preload Virtualmin libraries?
Default is no.

Run MariaDB database server?
I use MariaDB for my Tiki Wiki installs so, yes.
You will have to validate or change the MariaDB root password, enter it and save it somewhere if needed later.

Run PostgreSQL database server?
As stated above it will be a no.

DNS configuration
You can keep it as is (the primary domain you use as hostname) or adapt to your need (you need to know what you do of course).

You can validate and stop here or continu with the non mandatory post-install wizard will start

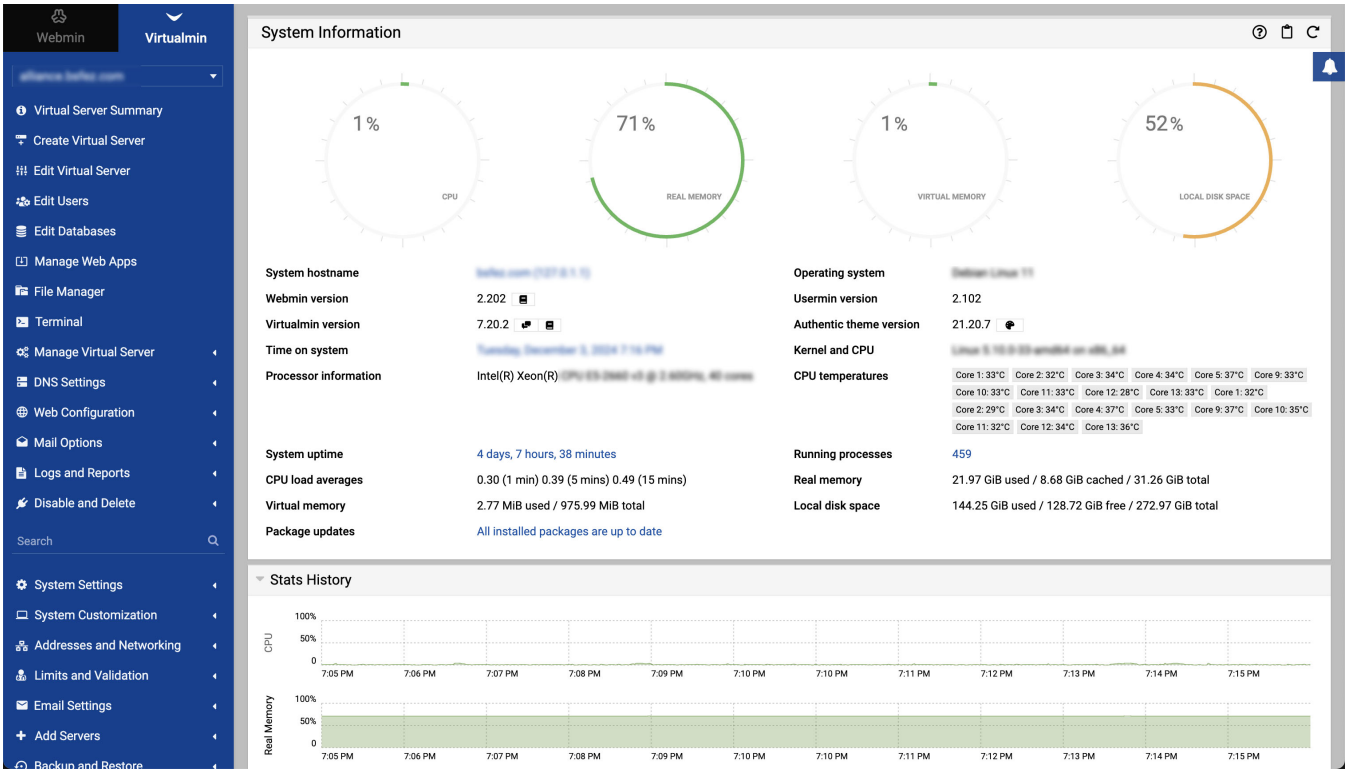
Password storage mode
I don't want to store in clear the passwords used on the server so I select "Only store hashed passwords".
One drawback is that you can't view again a password once set. So each time you create a password you will note it in your password storage application.

MariaDB configuration size
This depend of your usage of the Tiki Wiki instance but following the I follow the suggested option is working fine.

Location for SSL certificates
Unless you know what you do keep the default. (Per-domain)

It should be the end. But sometimes pops a request to create a default virtual server.
If this question is displayed I choose to answer "no" so I can set manually all the parameters for my domain.

I recheck configuration after my changes, I reboot the server and I'm ready to create my first Virtual Server or simply said website.



Create your first Virtual Server (your website)

Set your domain name.

The credential... I experienced issues creating a password without SSH public key and I experienced adding an SSH public key without password.
Therefor I'm creating a user password that will be require in a few cases and I add my SSH public key for day to day operation.

In some cases (temporary domain name) I set my "Administration username" and set the "Default database name" with something like "thiswebsite_tiki27" as I like to keep track of the Tiki used for a database. (I don't keep the same db from Tiki to Tiki I clone them)

I keep the features as is but disable "Accept mail for domain" if I don't want to use mail for this domain.

By default your Virtual Server will be limited to the default plan.
While this is enough if you install a Tiki package, you may need more memory to run composer.
Adjust the quota at Edit Virtual Server => Quotas and limits.

Install Let's encrypt certificate

To complete properly, return to the Virtual Server List, select your Virtual Server and on the left menu under "Server Configuration", "SSL Certificate" and select the tab "Let's Encrypt".
On the form "Request certificate for" use "Domain names listed here " field and set the domains for whom you will need a certificate. Keep the "Automatically renew certificate?" option to yes and click on the button "Request Certificate".

You need first to use your domain name set it at the registrar properly.
Under Virtualmin => Server Configuration => SSL Certificate click on the "Let's Encrypt" tab and request certificate.
Note that by default Virtualmin want to setup several automatically domains the "Domains associated with this server" list. You may not have set all of them and the process will verify them and fail if they are not accessible. I usually set manually "Domain names listed here" list and Install the certificates only for the domains I'm using.

Checking servers running

On your dashboard you have a panel where you can check servers status. We can see the PHP version running and we need to look (again) at the mail and mailbox application.
Keep Postfix running that is useful for sending email from apps or system notifications.
You can turn off Dovecot that do not need to be running on a system that is not receiving or processing mail.
But this will not be permanent. To make it permanent, on the left menu click on "Webmin", "System" and "Bootup and Shutdown".
Select anything "Dovecot" and at the bottom of the page click on "Disable On Boot".
(it is a good idea to enable fail2ban and a firewall)

Install several versions of PHP

The following information was valid at the time I wrote this tutorial

Debian 12 comes with PHP8.2 a version that may or may not fit your needs.
As I want to use PHP8.1 I have to set it on the server.

Adding PHP8.1 from the sury repository

Don't forget that anyway you have to install a few additional libraries and they should be accessible to complete the setup.
That's why these have been added above but you can also add them for your other PHP versions

```
sudo apt-get install php8.2-gd php8.2-intl php8.2-curl php8.2-zip php8.2-bcmath
```

Secure the server (basic)

Enhancing server security requires skills, knowledge, and expertise beyond the scope of this article. However, it is crucial to implement basic measures to protect your server and data. Here, we will review the minimum steps you should take. For a comprehensive security review of your Tiki and server, please contact me.

Secure your MySQL database (mariaDB)

The last version of MySQL using the MariaDN engine will be installed by Virtualmin.
As seen Then we will secure the installation to tight our defences against the "bad guys".

Secure mariaDB

```
sudo mysql_secure_installation
```

You should be able to answer most of the questions without too much thinking those are my answers (explanations can be found on the web)

- Enter current password for root (enter for none) => enter the root password set during Virtualmin install, else:

- Set root password? [Y/n] y - *As it is a first install*
- Switch to unix_socket authentication [Y/n] Y
- Change the root password? [Y/n] n
- Remove anonymous users? [Y/n] y
- Disallow root login remotely? [Y/n] y
- Remove test database and access to it? [Y/n] y
- Reload privilege tables now? [Y/n] y

Done, let's check MariaDB is running:

Test mariaDB status

```
sudo systemctl status mariadb
```

Enable and configure fail2ban

First, in virtualmin check if the "Fail2Ban Intrusion Detector" is installed and running.
In the Webmin, Un-used module, by default you should see the "Fail2Ban Intrusion Detector" and you need to start it, install the necessary packages and enable on (server) boot.

Now on the Webmin, Networking panel you see "Fail2Ban Intrusion Detector" module but it is still stopped.
You can double check using your shell with the following

Check fail2ban status

```
sudo systemctl status fail2ban
```

On Debian12 there is a problem at the time this article was written. A default jail local file is missing and that doesn't allow fail2ban to start and you need to manually create it.

Create and edit the jail.local file

```
sudo nano /etc/fail2ban/jail.local
```

If there, add already another missing parameters for a filter you should use right away, SSHD.

Add sshd backed parameter

```
[sshd]
backend=systemd
```

Start and check fail2ban

```
sudo systemctl start fail2ban
sudo systemctl status fail2ban
```

We will enable as a minimum the following filters action jail from within the fail2ban panel:
sshd
apache-badbots
webmin-auth (add "/var/webmin/miniserv.log" to the logpath of this filter)

I developed an "apache-4xx" filter designed to ban rapid incoming connections from the same IP, typically from crawlers or brute force attempts resulting in 40x errors.
If you're interested in this filter, feel free to contact me.
I'd be delighted to share it with you!

Installing Tiki from Tiki repo (anonymous)

Navigate into your html directory. If you don’t know where it is located on your new server on Virtualmin check the Virtual Server Summary.
There is at https://tiki.org a complete installation guide
But in short, I use the git clone command to download Tiki. In my case I needed Tiki branch (version) 27.x without previous history (depth=1) but you can change and adapt based on your needs.

Download Tiki from the git repo

```
git clone --depth=1 --branch=27.x https://gitlab.com/tikiwiki/tiki.git .
```

Installing Tiki using ssh (your repo)

To use the SSH key used on your Gitlab account you need to create a config file at your_home/.ssh
For exemple `bsfez/.ssh` copy the following inside "config" file.

Content of the ssh config file

```
# GitLab.com
Host gitlab.com
#  User git
  PreferredAuthentications publickey
  IdentityFile ~/.ssh/id_rsa
```

Replace "id_rsa" with your SSH key file and set ownership to your virtual server user and permission to 400 (chmod).

Navigate into your html directory (public_html). If you don’t know where it is located on your new server on Virtualmin check the Virtual Server Summary.
There is at https://tiki.org a complete installation guide

Download Tiki from your git repo

```
git clone --branch=your_branch --depth=1 git@gitlab.com:bsfez/your_repo .
```

It happen that for the first download you have to point your key file. □□♂
You can do it using the following command. (again replace "id_rsa" with you key filename)

Download Tiki from your git repo

```
git clone --branch=your_branch --depth=1 git@gitlab.com:bsfez/your_repo --config core.sshCommand="ssh -i
../.ssh/id_rsa" .
```

Then it is required to run tiki setup to install Composer files and fix the files and directories permissions.

Tiki 27 uses a different method to complete the setup

From Tiki27 Tiki uses the Tiki 27 plus Build System.
This includes the integration of tools like Composer for PHP dependencies and Node.js for JavaScript and CSS dependencies.
Please refer to an article I wrote, How to upgrade to Tiki Wiki 27

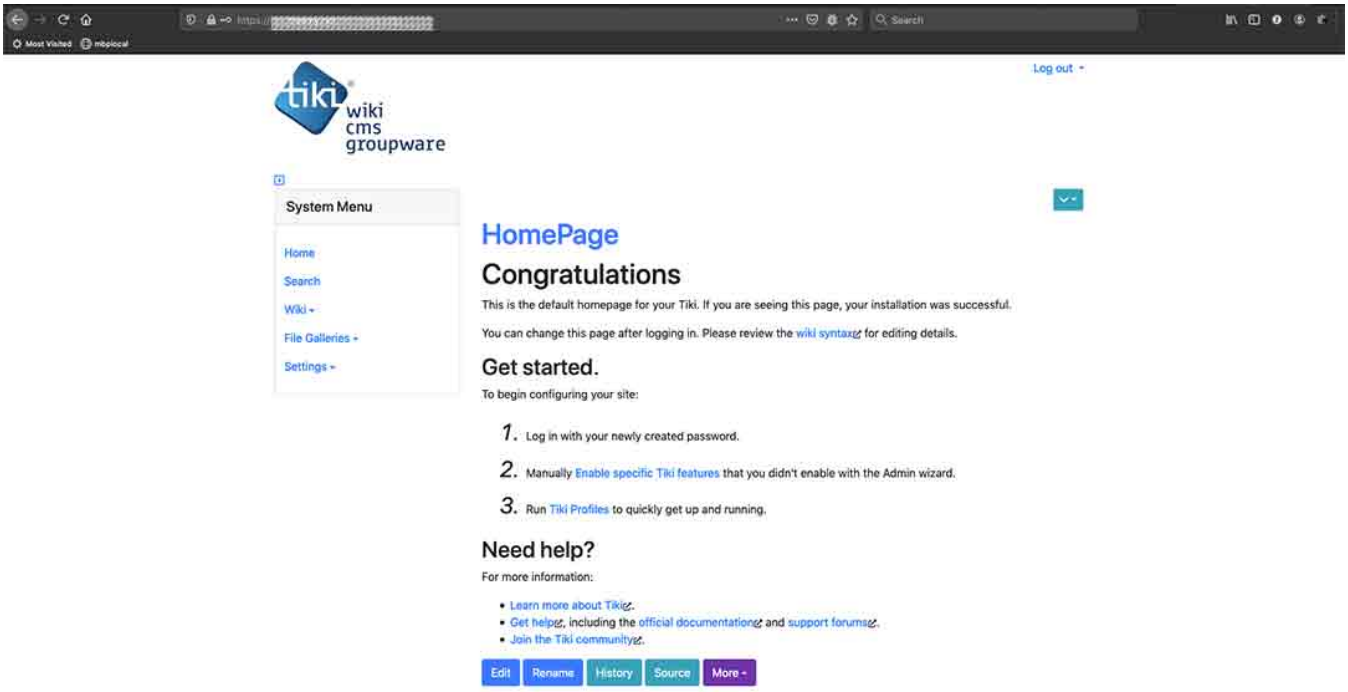
If you have a single version of PHP installed, it will be the one used. If you have several version of PHP installed, you will need to point the setup script to the CLI version of the PHP you want/need to use.
I want to run PHP8.3 (I have several PHP version installed) so I add the path to the right PHP Version.

Tiki setup to run php8.3

```
sh setup.sh -p /usr/bin/php8.3
```

From here follow the regular Tiki install process (setup.sh (see additional notes below), database creation) and you have a Tiki

ready to be installed !



Additional settings

Tiki optimized htaccess

Apache Caches: Enabling Essential Modules

In some Virtualmin setups, key modules that enhance cache management and improve page ranking and loading speed may not be activated. To ensure optimal performance, verify the following Apache modules are enabled:

- headers
- expires
- mime
- deflate

To check if these modules are enabled, enter the following command:

Check apache 2 enabled modules

```
sudo apache2ctl -M | grep -E 'headers|expires|mime|deflate'
```

You should see output similar to:

Apache 2 enabled modules output

```
deflate_module (shared)
 expires_module (shared)
 headers_module (shared)
 mime_module (shared)
```

If expires_module (shared) and/or headers_module (shared) are missing from the output, enable them by running:

Enabled apache2 modules

```
sudo a2enmod headers
sudo a2enmod expires
sudo systemctl reload apache2
```

After running these commands, recheck the enabled modules to confirm they are now active.
This process ensures that your Apache server is optimized for better performance and cache management.