CentOs 7 (Taken from https://www.howtoforge.com/tutorial/how-to-install-wikijs-on-centos-7/)

How to Install Wiki.js on CentOS 7

Wiki.js is a modern and powerful OpenSource wiki app based on Node.js, Git, and Markdown. Wiki.js runs on the flamingly fast Node.js engine and is optimized to conserve CPU resources. Some of the Wiki.js features worth mentioning are:

- Markdown editing, backed by Git
- Lightweight, yet extremely powerful
- Beautifully designed for the modern web
- Integrated Access Control
- Intuitive Assets Management
- Built-in search engine

In this tutorial, we will walk you through the Wiki.js version 1 installation process on a CentOS 7 operating system by using NGINX as a reverse proxy server, MongoDB as a database server, PM2 as a process manager and optionally you can secure transport layer by using acme.sh client and Let's Encrypt certificate authority to add SSL support.

Requirements

Requirements to run Wiki.js are the following:

- Node.js 6.11.1 to 10.x is required.
- MongoDB version 3.2 or later.
- Git version 2.7.4 or later.
- Web Server software such as NGINX, Apache, Caddy, H2O...
- An empty Git repository (optional).
- Minimum of 512MB RAM. 1GB of RAM recommended.
- About 300MB of disk space.
- Domain name with A/AAAA DNS records set up.

Prerequisites

- A CentOS 7 operating system.
- A non-root user with sudo privileges.

Initial Steps

Check your CentOS version:

```
cat /etc/centos-release
```

Set up the timezone:

```
timedatectl list-timezones
sudo timedatectl set-timezone 'Region/City'
```

Update your operating system packages (software). This is an important first step because it ensures you have the latest updates and security fixes for your operating system's default software packages:

```
sudo yum update -y
```

Install some essential packages that are necessary for basic administration of the CentOS operating system:

```
sudo yum install -y curl wget vim unzip socat epel-release
```

Wiki.js requires you to install Git version 2.7.4 or greater. Default Git version on CentOS is too old, so you will need to build it from source code. Here are the commands to quickly compile it:

```
# Remove existing git package if installed:
sudo yum remove -y git
sudo yum groupinstall -y "Development Tools"
sudo yum install -y gettext-devel openssl-devel perl-CPAN perl-devel zlib-devel curl-devel
wget https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.21.0.tar.gz && tar zxvf git-2.21.0.tar.gz
rm git-2.21.0.tar.gz
cd git-2.21.0
make configure
./configure make prefix=/usr/local all
sudo make prefix=/usr/local install
cd ~
which git
git --version
```

Step 1- Install Node.js and npm

Wiki.js is built on Node.js. We are going to install the latest recommended version for Wiki.js which is version 10 at the time of this writing. On Linux, you have a few Node.js installation options: Linux Binaries (x86/x64), Source Code or via Package Managers. We will use a Package Management option which makes installing and updating Node.js a breeze.

Download and install the latest recommended version of Node.js from the NodeSource repository:

```
curl --silent --location https://rpm.nodesource.com/setup_10.x | sudo bash -
sudo yum install -y nodejs
```

NOTE: npm is distributed with Node.js - which means that when you download Node.js, you automatically get npm installed on your system.

Check Node.js and npm version:

```
node -v && npm -v
# v10.15.1
# 6.4.1
```

Npm is a separate project from Node.js and tends to update more frequently. As a result, even if you've just downloaded Node.js (and therefore npm), you'll probably need to update your npm. Luckily, npm knows how to update itself! To update your npm, type this into your terminal:

```
sudo npm install -g npm@latest
```

This command will update npm to the latest stable version.

Re-check npm version with:

```
npm -v
# 6.8.0
```

And it should return the latest version number.

Step 2 - Install MongoDB database

Wiki.js needs a database to store its data, and the current stable version of Wiki.js supports only MongoDB database engine. According to that, we will need to install MongoDB database.

Create a /etc/yum.repos.d/mongodb-org-4.0.repo file so that you can install MongoDB directly using yum.

Run:

```
sudo vim /etc/yum.repos.d/mongodb-org-4.0.repo
```

And populate the file with the following config:

```
[mongodb-org-4.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.0.asc
```

To install the latest stable version of MongoDB, issue the following command:

```
sudo yum install -y mongodb-org
```

Check the MongoDB version:

```
mongo --version | head -n 1 && mongod --version | head -n 1
# MongoDB shell version v4.0.6
# db version v4.0.6
```

Start and enable (*set it to start on reboot*) MongoDB service if not already started and enabled:

```
sudo systemctl start mongod.service
sudo systemctl enable mongod.service
```

Step 3 - Install acme.sh client and obtain Let's Encrypt certificate (optional)

Securing your website with HTTPS is not necessary, but it is a good practice to secure your site traffic. In order to obtain an SSL certificate from Let's Encrypt, we will use acme.sh client. Acme.sh is a pure UNIX shell software for obtaining SSL certificates from Let's Encrypt with zero dependencies.

Download and install acme.sh:

```
sudo su - root
git clone https://github.com/Neilpang/acme.sh.git
cd acme.sh
./acme.sh --install --accountemail your_email@example.com
source ~/.bashrc
cd ~
```

Check acme.sh version:

```
acme.sh --version
# v2.8.1
```

Obtain **RSA** and **ECC/ECDSA** certificates for your domain/hostname:

```
# RSA 2048
acme.sh --issue --standalone -d example.com --keylength 2048
# ECDSA
acme.sh --issue --standalone -d example.com --keylength ec-256
```

If you want fake certificates for testing you can add --staging flag to the above commands.

To list your issued certs you can run:

```
acme.sh --list
```

Create folders to store your certs. We will use /etc/letsencrypt but it can be anything you prefer to store SSL certs.

```
mkdir -p /etc/letsencrypt/example.com
mkdir -p /etc/letsencrypt/example.com_ecc
```

Install/copy certificates to /etc/letsencrypt directory.

```
# RSA
acme.sh --install-cert -d example.com \
    --cert-file /etc/letsencrypt/example.com/cert.pem \
    --key-file /etc/letsencrypt/example.com/private.key \
    --fullchain-file /etc/letsencrypt/example.com/fullchain.pem \
    --reloadcmd "sudo systemctl reload nginx.service"

# ECC/ECDSA
acme.sh --install-cert -d example.com --ecc \
    --cert-file /etc/letsencrypt/example.com_ecc/cert.pem \
    --key-file /etc/letsencrypt/example.com_ecc/private.key \
    --fullchain-file /etc/letsencrypt/example.com_ecc/fullchain.pem \
    --reloadcmd "sudo systemctl reload nginx.service"
```

After running the above commands, your *certificates* and *keys* will be in:

- For **RSA**: /etc/letsencrypt/example.com directory.
- For **ECC/ECDSA**: /etc/letsencrypt/example.com_ecc directory.

All the certificates will be automatically renewed every 60 days.

After obtaining certs, exit from root user and return back to normal sudo user:

```
exit
```

Step 4 - Install and configure NGINX

Wiki.js (or any HTTP Node.js app) can run without any actual web server (such as NGINX or Apache). However, it is highly recommended to put a standard web server in front of Wiki.js. This ensures you can use features like SSL, multiple websites, caching, etc. We will use NGINX in this tutorial, but any other server will do, you just need to configure it properly.

Download and install NGINX:

```
sudo yum install -y nginx
```

Check the NGINX version:

```
nginx -v
# nginx version: nginx/1.12.2
```

Enable and start NGINX service:

```
sudo systemctl enable nginx.service
sudo systemctl start nginx.service
```

Run sudo vim /etc/nginx/conf.d/wiki.js.conf and configure NGINX as an HTTPS reverse proxy.

```
server {

  listen [::]:443 ssl http2;
  listen 443 ssl http2;
   listen [::]:80;
   listen 80;

   server_name example.com;

   charset utf-8;
   client_max_body_size 50M;

  ssl_certificate /etc/letsencrypt/example.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/example.com/private.key;
  ssl_certificate /etc/letsencrypt/example.com_ecc/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/example.com_ecc/private.key;
```

```
    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_next_upstream error timeout http_502 http_503 http_504;
    }

}
```

The only thing you need to change in the above config is server_name directive, and potentially proxy_pass directive if you decide to configure some other than 3000 port. Wiki.js uses port 3000 by default.

Check the NGINX configuration:

```
sudo nginx -t
```

Finally, for changes to take effect, we need to reload NGINX:

```
sudo systemctl reload nginx.service
```

Step 5 - Install and setup Wiki.js

Create a document root directory where Wiki.js should reside in:

```
sudo mkdir -p /var/www/wiki.js
```

Navigate to the document root directory:

```
cd /var/www/wiki.js
```

Change ownership of the /var/www/wiki.js directory to your_user:

```
sudo chown -R [your_user]:[your_user] /var/www/wiki.js
```

NOTE: *Replace your_user in the above command with your non-root user that you should have created as a prerequisite for this tutorial.*

From /var/www/wiki.js directory, run the following command to fetch and install the latest Wiki.js application:

```
curl -sSo- https://wiki.js.org/install.sh | bash
```

Once the installation is completed, you can run the following command in order to view the currently installed version of Wiki.js:

```
node wiki --version
# 1.0.117
```

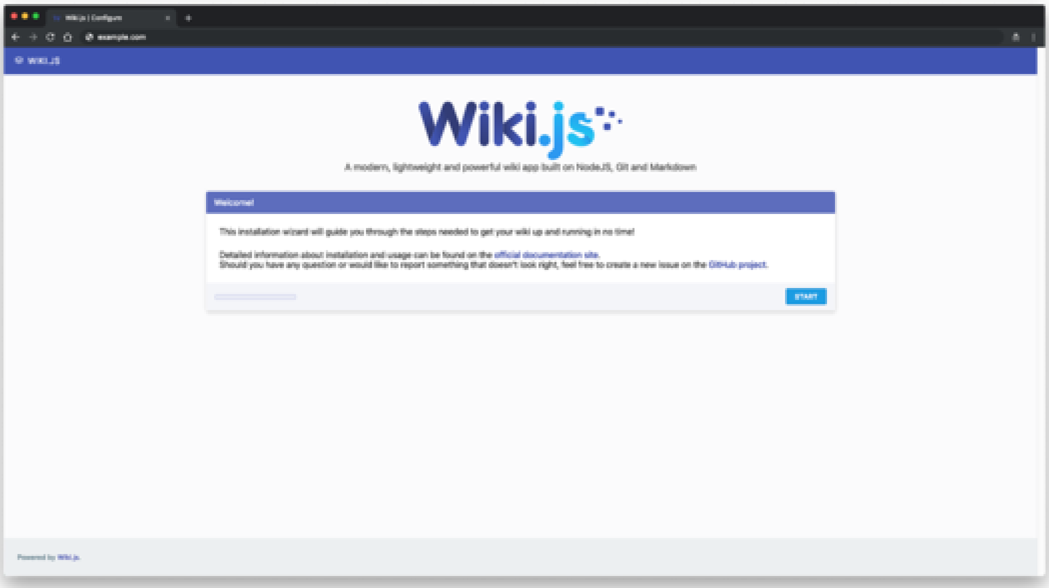
Once the installation is completed, you'll be prompted to launch the configuration wizard.

So, start the configuration wizard by running:

```
node wiki configure
```

Using your web browser, navigate to http://example.com and follow the on-screen instructions. All settings entered during the configuration wizard are saved in the config.yml file. The configuration wizard will automatically start Wiki.js for you.

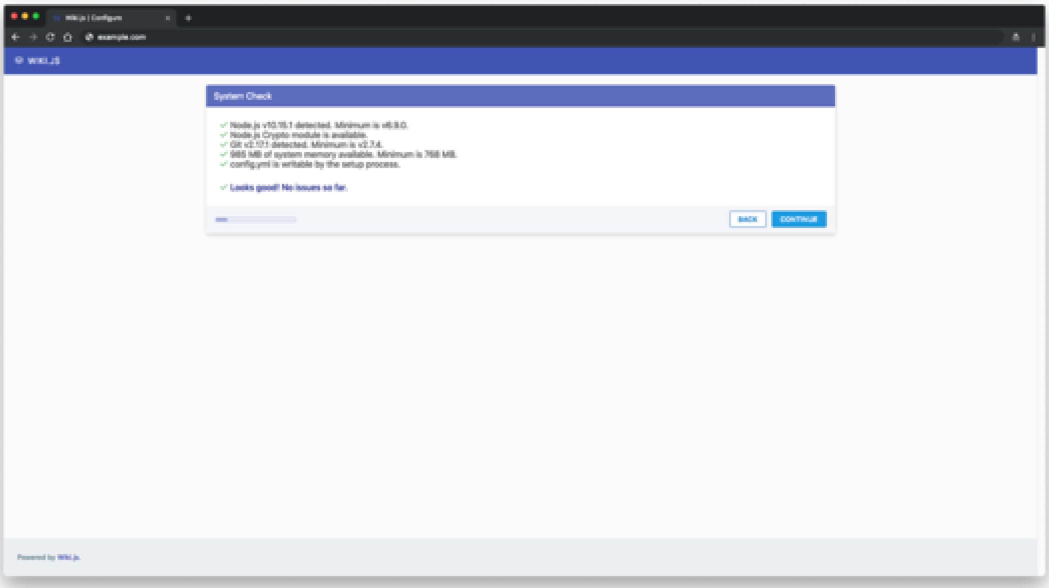
First, you will see a welcome message. Click on the "**Start**" button:


Wiki.js | Configure
example.com
WIKI.JS
Wiki.js
A modern, lightweight and powerful wiki app built on NodeJS, Git and Markdown
Welcome!
START
Powered by Wiki.js

This installation wizard will guide you through the steps needed to get your wiki up and running in no time!

Detailed information about installation and usage can be found on the official documentation site.
Should you have any question or would like to report something that doesn't look right, feel free to create a new issue on the GitHub project.

Next "System Check" page will appear. If all requirements are met, click on the "**Continue**" button.

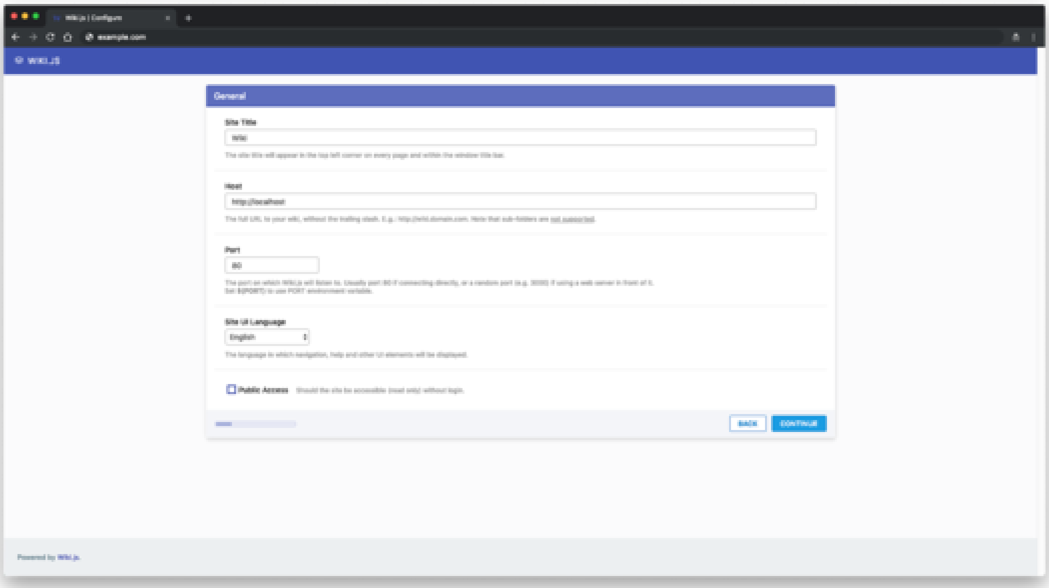


System Check

✓ Node.js v10.15.1 detected. Minimum is v6.9.0.
✓ Node.js Crypto module is available.
✓ Git v2.17.1 detected. Minimum is v2.7.4.
✓ 985 MB of system memory available. Minimum is 768 MB.
✓ config.yml is writable by the setup process.

✓ Looks good! No issues so far.

Enter general information about your wiki and click the "**Continue**" button:



General

Site Title
Wiki
The site title will appear in the top left corner on every page and within the window title bar.

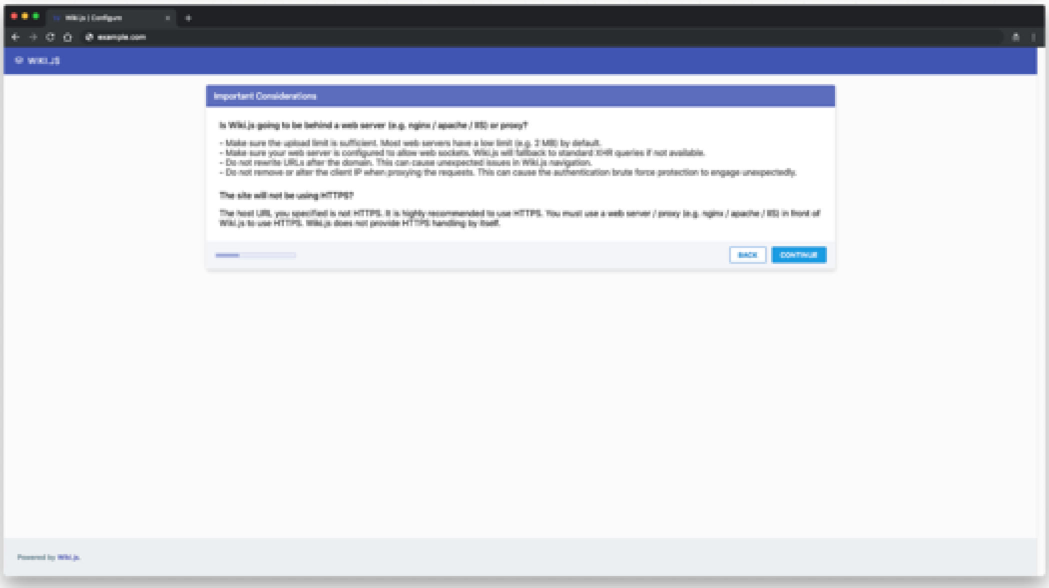
Host
http://localhost
The full URL to your wiki, without the trailing slash. E.g.: http://wiki.domain.com. Note that sub-folders are not supported.

Port
80
The port on which Wiki.js will listen to. Usually port 80 if connecting directly, or a random port (e.g. 3000) if using a web server in front of it.
Set $(PORT) to use PORT environment variable.

Site UI Language
English
The language in which navigation, help and other UI elements will be displayed.

□ Public Access Should the site be accessible (read only) without login.

Read the "Important Consideration" notice and click "**Continue**" for the next step:



Important Considerations

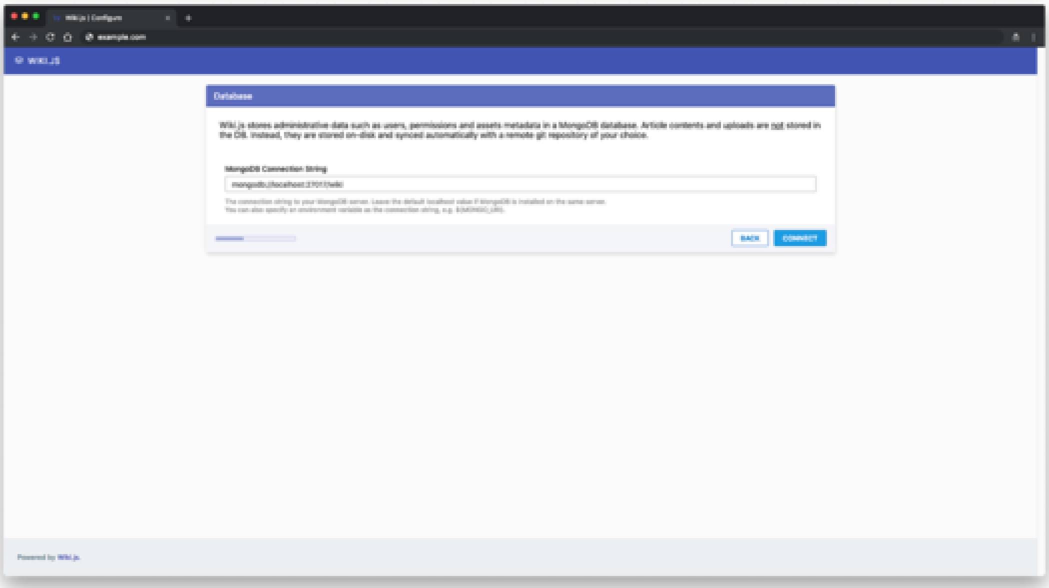
Is Wiki.js going to be behind a web server (e.g. nginx / apache / IIS) or proxy?

- Make sure the upload limit is sufficient. Most web servers have a low limit (e.g. 2 MB) by default.
- Make sure your web server is configured to allow web sockets. Wiki.js will fallback to standard XHR queries if not available.
- Do not rewrite URLs after the domain. This can cause unexpected issues in Wiki.js navigation.
- Do not remove or alter the client IP when proxying the requests. This can cause the authentication brute force protection to engage unexpectedly.

The site will not be using HTTPS?

The host URL you specified is not HTTPS. It is highly recommended to use HTTPS. You must use a web server / proxy (e.g. nginx / apache / IIS) in front of Wiki.js to use HTTPS. Wiki.js does not provide HTTPS handling by itself.

Next, connect to the database and continue:



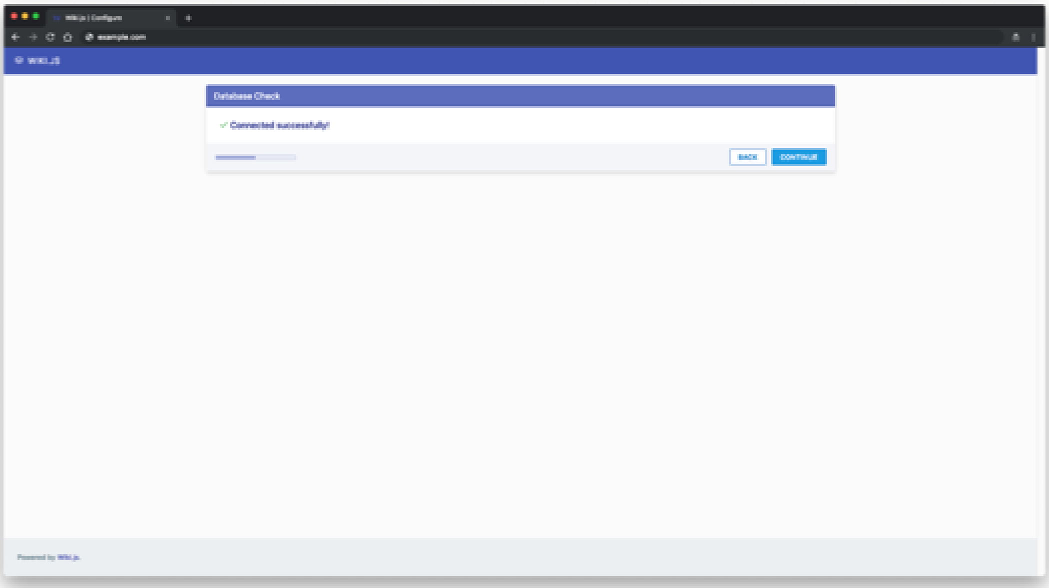
Database

Wiki.js stores administrative data such as users, permissions and assets metadata in a MongoDB database. Article contents and uploads are not stored in the DB. Instead, they are stored on-disk and synced automatically with a remote git repository of your choice.

MongoDB Connection String: mongodb://localhost:27017/wiki

The connection string to your MongoDB server. Leave the default localhost value if MongoDB is installed on the same server.
You can also specify an environment variable as the connection string, e.g. $(MONGO_URI).

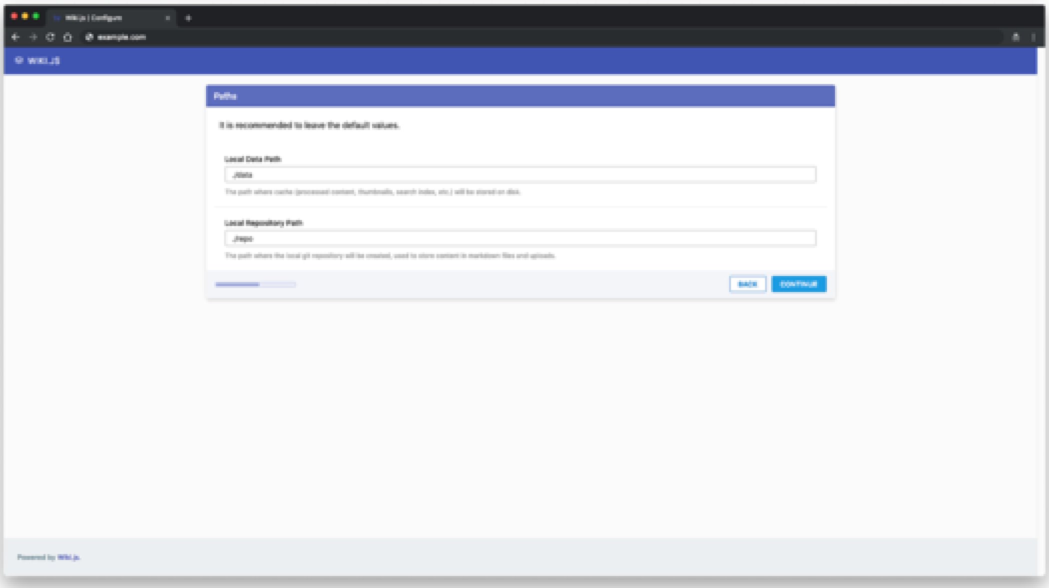
You should see a message that Wiki.js has been successfully connected to the database. Click the "**Continue**" button:



Database Check

✓ Connected successfully!

Set paths and continue:



Paths

It is recommended to leave the default values.

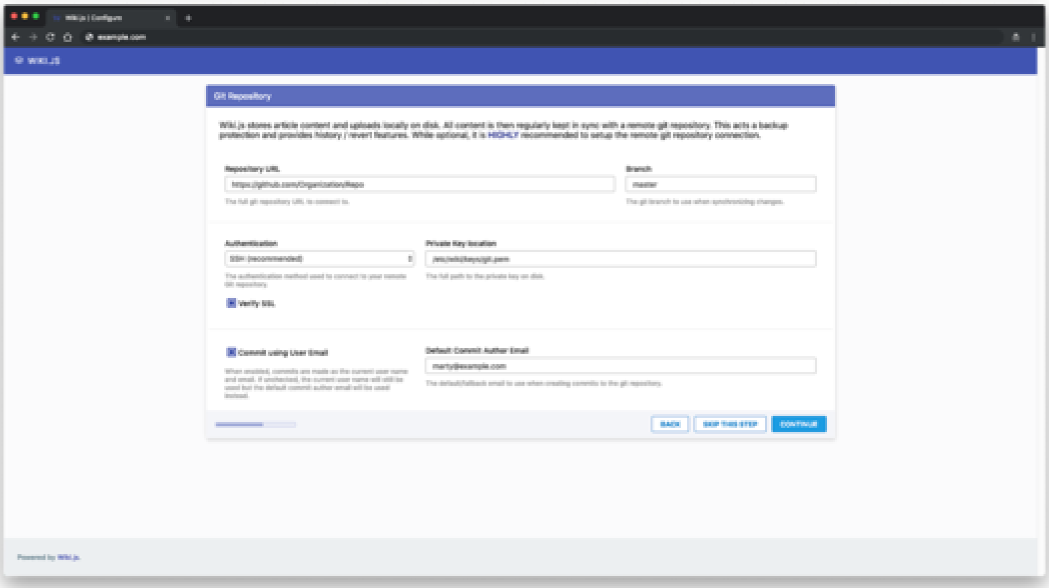
Local Data Path: ./data

The path where cache (processed content, thumbnails, search index, etc.) will be stored on disk.

Local Repository Path: ./repo

The path where the local git repository will be created, used to store content in markdown files and uploads.

Set up remote Git repo if you want or skip this step. This step is optional but highly recommended:



Git Repository

Wiki.js stores article content and uploads locally on disk. All content is then regularly kept in sync with a remote git repository. This acts a backup protection and provides history / revert features. While optional, it is HIGHLY recommended to setup the remote git repository connection.

Repository URL: https://github.com/Organization/Repo

The full git repository URL to connect to.

Branch: master

The git branch to use when synchronizing changes.

Authentication: SSH (recommended)

The authentication method used to connect to your remote Git repository.

Private Key location: /etc/wiki/keys/git.pem

The full path to the private key on disk.

☑ Verify SSL

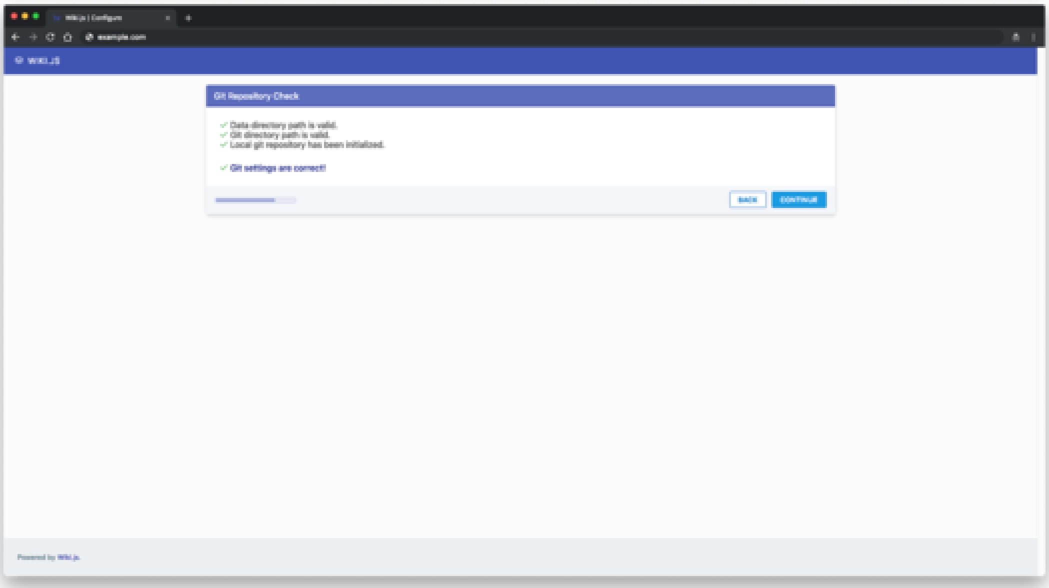
☑ Commit using User Email

When enabled, commits are made as the current user name and email. If unchecked, the current user name will still be used but the default commit author email will be used instead.

Default Commit Author Email: marty@example.com

The default/fallback email to use when creating commits to the git repository.

Next, click the "Continue" button:

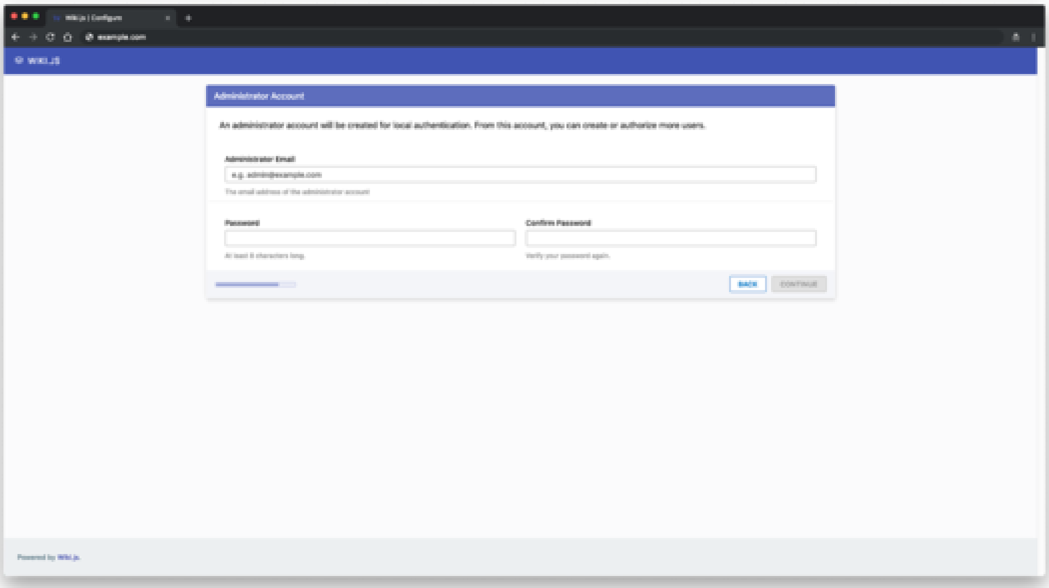


Git Repository Check

✓ Data directory path is valid.
✓ Git directory path is valid.
✓ Local git repository has been initialized.

✓ Git settings are correct!

Create an admin account and click the "**Continue**" button:



An administrator account will be created for local authentication. From this account, you can create or authorize more users.

Administrator Email

e.g. admin@example.com

The email address of the administrator account

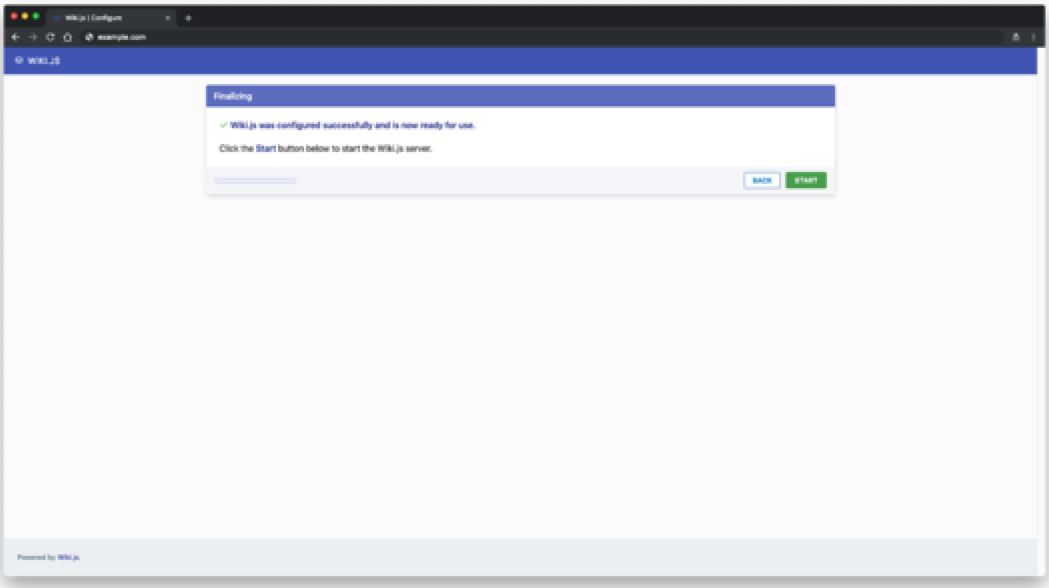
Password

At least 8 characters long.

Confirm Password

Verify your password again.

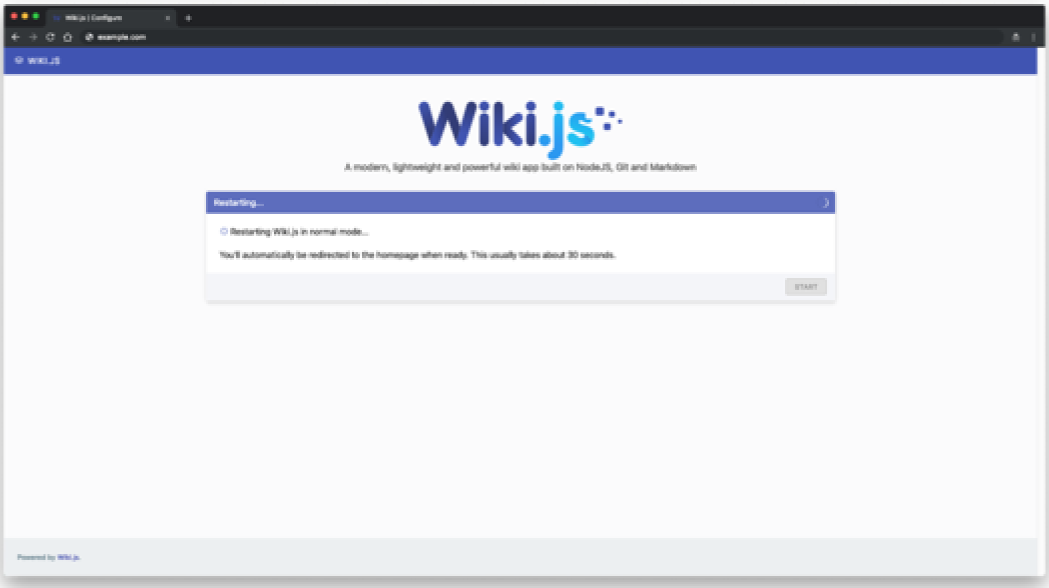
And finally, start the Wiki.js:



✓ Wiki.js was configured successfully and is now ready for use.

Click the Start button below to start the Wiki.js server.

Wait around 30 seconds and you should be redirected to Wiki.js homepage:



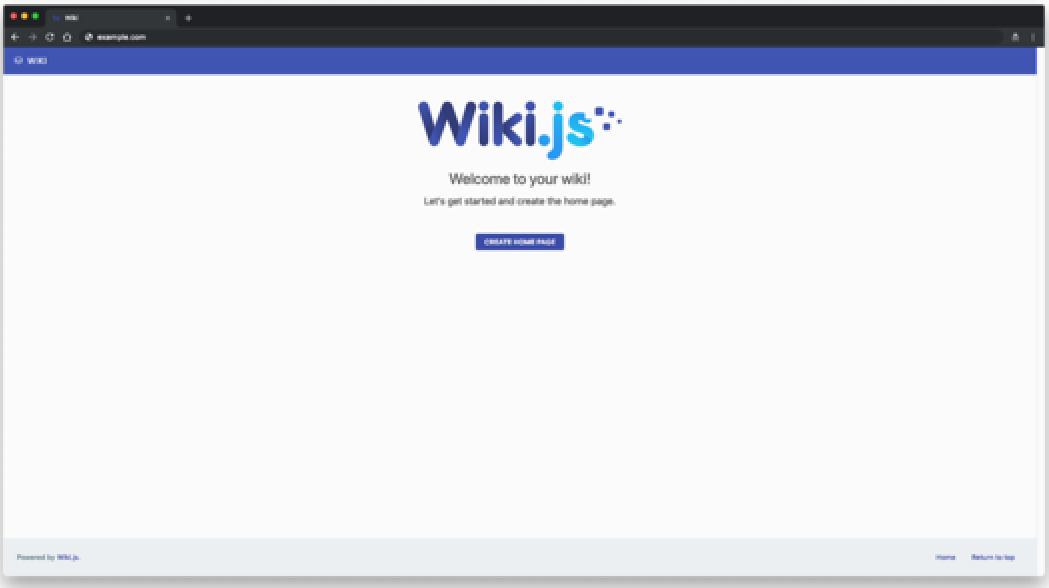
A modern, lightweight and powerful wiki app built on NodeJS, Git and Markdown

Restarting...

Restarting Wiki.js in normal mode...

You'll automatically be redirected to the homepage when ready. This usually takes about 30 seconds.

The installation is completed. You should wiki welcome page:



Welcome to your wiki!

Let's get started and create the home page.

Step 6 - Setup PM2 Process Manager

By default, Wiki.js will not start automatically after a system reboot. In order to make it start on boot, we need to setup PM2 process manager. PM2 comes bundled with Wiki.js as a local npm module, so we don't need to install PM2 globally.

Tell PM2 to configure itself as a startup service by running:

```
/var/www/wiki.js/node_modules/pm2/bin/pm2 startup
```

Finally, save the current PM2 configuration by running the command:

```
/var/www/wiki.js/node_modules/pm2/bin/pm2 save
```

Your Wiki.js now runs as a background process, using PM2 as its process manager.

Ubuntu 18.04

How to Install Wiki.js on Ubuntu 18.04 LTS

In this tutorial, we will walk you through the Wiki.js version 1 installation process on a Ubuntu 18.04 LTS operating system by using NGINX as a reverse proxy server, MongoDB as a database server, PM2 as a process manager and optionally you can secure transport layer by using acme.sh client and Let's Encrypt certificate authority to add SSL support.

Requirements

Requirements to run Wiki.js are the following:

- Node.js 6.11.1 to 10.x is required.
- MongoDB version 3.2 or later.
- Git version 2.7.4 or later.
- Web Server software such as NGINX, Apache, Caddy, H2O...
- An empty Git repository (optional).
- Minimum of 512MB RAM. 1GB of RAM recommended.
- About 300MB of disk space.
- Domain name with A/AAAA DNS records set up.

Prerequisites

- An Ubuntu 18.04 LTS operating system.
- A non-root user with sudo privileges.

Initial Steps

Check your Ubuntu version:

```
lsb_release -ds
# Ubuntu 18.04.1 LTS
```

Set up the timezone:

```
sudo dpkg-reconfigure tzdata
```

Update your operating system packages (software). This is an important first step because it ensures you have the latest updates and security fixes for your operating system's default software packages:

```
sudo apt update && sudo apt upgrade -y
```

Install some essential packages that are necessary for basic administration of Ubuntu operating system:

```
sudo apt install -y curl wget vim git unzip socat bash-completion apt-transport-https
build-essential
```

Step 1- Install Node.js and npm

Wiki.js is built on Node.js. We are going to install the latest recommended version for Wiki.js which is version 10 at the time of this writing. On Linux, you have a few Node.js installation options: Linux Binaries (x86/x64), Source Code or via Package Managers. We will use a Package Management option which makes installing and updating Node.js a breeze.

Download and install the latest recommended version of Node.js from the NodeSource repository:

```
curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
sudo apt install -y nodejs
```

NOTE: npm is distributed with Node.js - which means that when you download Node.js, you automatically get npm installed on your system.

Check Node.js and npm version:

```
node -v && npm -v
# v10.15.1
# 6.4.1
```

Npm is a separate project from Node.js and tends to update more frequently. As a result, even if you've just downloaded Node.js (and therefore npm), you'll probably need to update your npm. Luckily, npm knows how to update itself! To update your npm, type this into your terminal:

```
sudo npm install -g npm@latest
```

This command will update npm to the latest stable version.

Re-check npm version with:

```
npm -v
# 6.8.0
```

And it should return the latest version number.

Step 2 - Install MongoDB database

Wiki.js needs a database to store its data, and the current stable version of Wiki.js supports only MongoDB database engine. According to that, we will need to install MongoDB database.

Download and install MongoDB database:

```
sudo apt install -y mongodb
```

Check the MongoDB version:

```
mongo --version | head -n 1 && mongod --version | head -n 1
# MongoDB shell version v3.6.3
# db version v3.6.3
```

Start and enable (*set it to start on reboot*) MongoDB service if not already started and enabled:

```
sudo systemctl start mongodb.service
sudo systemctl enable mongodb.service
```

Step 3 - Install acme.sh client and obtain Let's Encrypt certificate (*optional*)

Securing your website with HTTPS is not necessary, but it is a good practice to secure your site traffic. In order to obtain an SSL certificate from Let's Encrypt, we will use acme.sh client. Acme.sh is a pure UNIX shell software for obtaining SSL certificates from Let's Encrypt with zero dependencies.

Download and install acme.sh:

```
sudo su - root
git clone https://github.com/Neilpang/acme.sh.git
cd acme.sh
./acme.sh --install --accountemail your_email@example.com
source ~/.bashrc
cd ~
```

Check acme.sh version:

```
acme.sh --version
# v2.8.1
```

Obtain **RSA** and **ECC/ECDSA** certificates for your domain/hostname:

```
# RSA 2048
acme.sh --issue --standalone -d example.com --keylength 2048
# ECDSA
acme.sh --issue --standalone -d example.com --keylength ec-256
```

If you want fake certificates for testing you can add --staging flag to the above commands.

To list your issued certs you can run:

```
acme.sh --list
```

Create folders to store your certs. We will use /etc/letsencrypt but it can be anything you prefer to store SSL certs.

```
mkdir -p /etc/letsencrypt/example.com
mkdir -p /etc/letsencrypt/example.com_ecc
```

Install/copy certificates to /etc/letsencrypt directory.

```
# RSA
acme.sh --install-cert -d example.com \
    --cert-file /etc/letsencrypt/example.com/cert.pem \
    --key-file /etc/letsencrypt/example.com/private.key \
    --fullchain-file /etc/letsencrypt/example.com/fullchain.pem \
    --reloadcmd "sudo systemctl reload nginx.service"

# ECC/ECDSA
acme.sh --install-cert -d example.com --ecc \
    --cert-file /etc/letsencrypt/example.com_ecc/cert.pem \
    --key-file /etc/letsencrypt/example.com_ecc/private.key \
    --fullchain-file /etc/letsencrypt/example.com_ecc/fullchain.pem \
    --reloadcmd "sudo systemctl reload nginx.service"
```

After running the above commands, your *certificates* and *keys* will be in:

- For **RSA**: /etc/letsencrypt/example.com directory.
- For **ECC/ECDSA**: /etc/letsencrypt/example.com_ecc directory.

All the certificates will be automatically renewed every 60 days.

After obtaining certs, exit from root user and return back to normal sudo user:

```
exit
```

Step 4 - Install and configure NGINX

Wiki.js (or any HTTP Node.js app) can run without any actual web server (such as NGINX or Apache). However, it is highly recommended to put a standard web server in front of Wiki.js. This ensures you can use features like SSL, multiple websites, caching, etc. We will use NGINX in this tutorial, but any other server will do, you just need to configure it properly.

Download and install NGINX from the official NGINX repo by issuing the following commands:

```
wget https://nginx.org/keys/nginx_signing.key
sudo apt-key add nginx_signing.key
rm nginx_signing.key
sudo -s
printf "deb https://nginx.org/packages/mainline/ubuntu/ $(lsb_release -sc) nginx\ndeb-src
https://nginx.org/packages/mainline/ubuntu/ $(lsb_release -sc) nginx\n" >>
/etc/apt/sources.list.d/nginx_mainline.list
exit
sudo apt update
sudo apt install -y nginx
```

Check the NGINX version:

```
sudo nginx -v
# nginx version: nginx/1.15.8
```

Enable and start NGINX service:

```
sudo systemctl enable nginx.service
sudo systemctl start nginx.service
```

Run sudo vim /etc/nginx/conf.d/wiki.js.conf

```
sudo vim /etc/nginx/conf.d/wiki.js.conf
```

and configure NGINX as an HTTPS reverse proxy.

```
server {

  listen [::]:443 ssl http2;
  listen 443 ssl http2;
   listen [::]:80;
   listen 80;

   server_name example.com;

   charset utf-8;
   client_max_body_size 50M;

  ssl_certificate /etc/letsencrypt/example.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/example.com/private.key;
  ssl_certificate /etc/letsencrypt/example.com_ecc/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/example.com_ecc/private.key;

   location / {
       proxy_set_header Host $http_host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_pass http://127.0.0.1:3000;
       proxy_http_version 1.1;
       proxy_set_header Upgrade $http_upgrade;
       proxy_set_header Connection "upgrade";
       proxy_next_upstream error timeout http_502 http_503 http_504;
   }

}
```

The only thing you need to change in the above config is server_name directive, the domain name in the ssl file paths, and potentially proxy_pass directive if you decide to configure some other than 3000 port. Wiki.js uses port 3000 by default.

Check the NGINX configuration:

```
sudo nginx -t
```

Finally, for changes to take effect, we need to reload NGINX:

```
sudo systemctl reload nginx.service
```

Step 5 - Install and setup Wiki.js

Create a document root directory where Wiki.js should reside in:

```
sudo mkdir -p /var/www/wiki.js
```

Navigate to the document root directory:

```
cd /var/www/wiki.js
```

Create a user for wikijs

```
sudo useradd -d /var/www/wiki.js wikijs
```

Change ownership of the /var/www/wiki.js directory to wikijs user:

```
sudo chown -R wikijs:wikijs /var/www/wiki.js
```

From /var/www/wiki.js directory, run the following command to fetch and install the latest Wiki.js application:

```
cd /var/www/wiki.js
sudo su wikijs
curl -sSo- https://wiki.js.org/install.sh | bash
```

Once the installation is completed, you can run the following command in order to view the currently installed version of Wiki.js:

```
node wiki --version
# 1.0.117
```

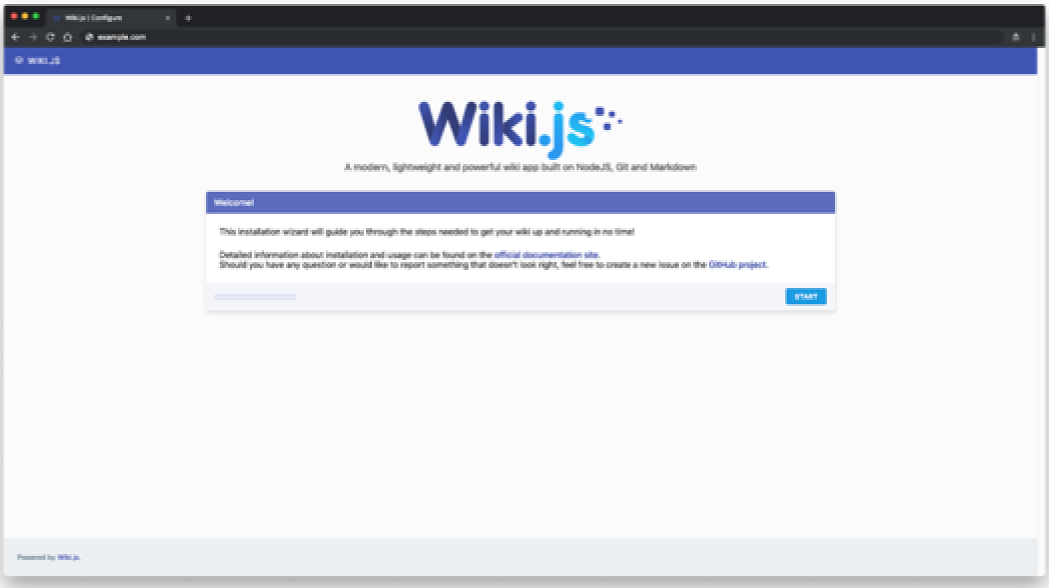
Once the installation is completed, you'll be prompted to launch the configuration wizard.

So, start the configuration wizard by running:

```
node wiki configure
```

Using your web browser, navigate to http://example.com and follow the on-screen instructions. All settings entered during the configuration wizard are saved in the config.yml file. The configuration wizard will automatically start Wiki.js for you.

First, you will see a welcome message. Click on the "**Start**" button:



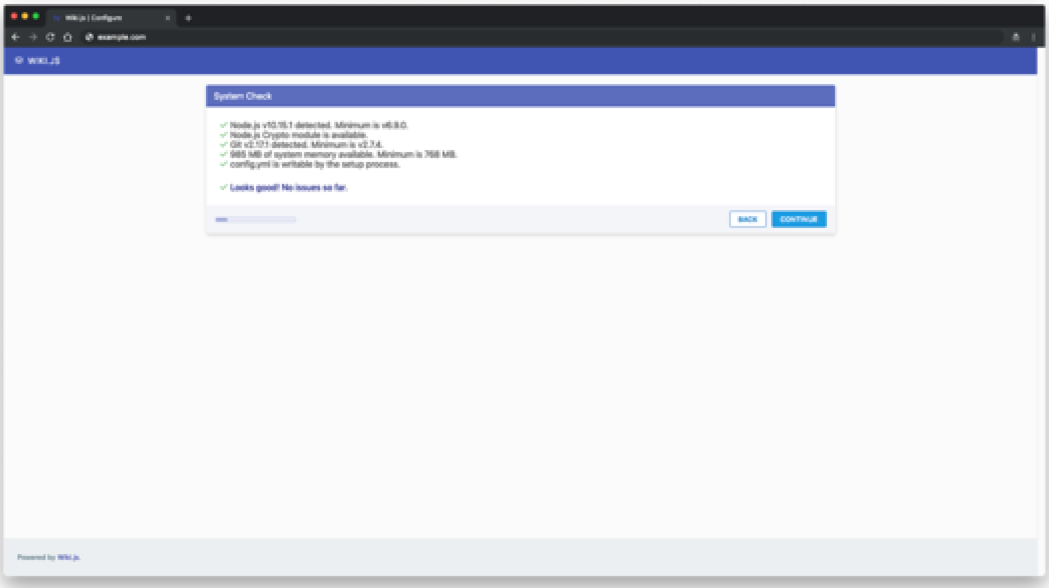
A modern, lightweight and powerful wiki app built on NodeJS, Git and Markdown

Welcome!

This installation wizard will guide you through the steps needed to get your wiki up and running in no time!

Detailed information about installation and usage can be found on the official documentation site.
Should you have any question or would like to report something that doesn't look right, feel free to create a new issue on the GitHub project.

Next "System Check" page will appear. If all requirements are met, click on the "**Continue**" button.

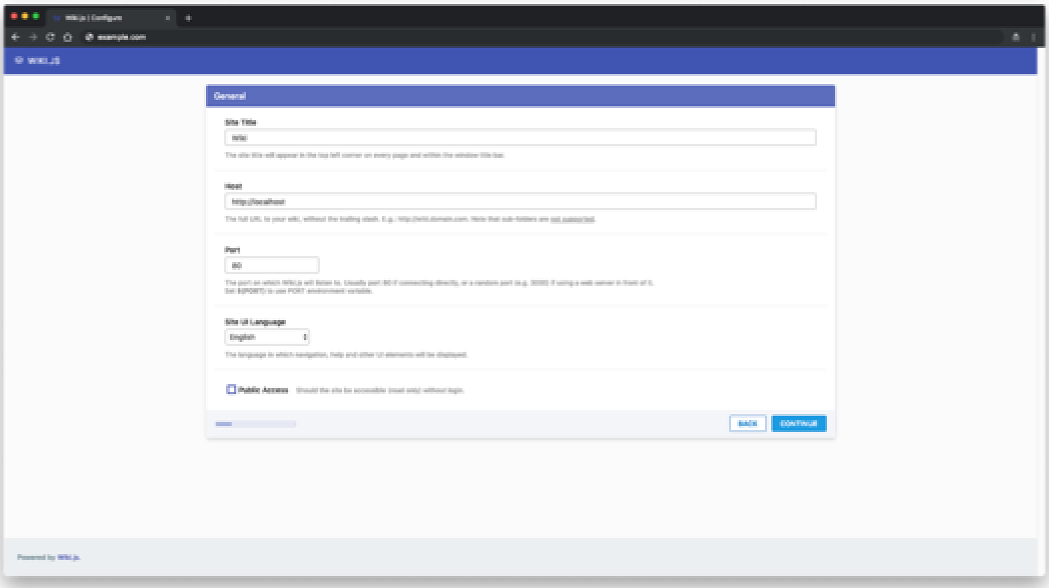


System Check

- ✓ Node.js v10.15.1 detected. Minimum is v6.9.0.
- ✓ Node.js Crypto module is available.
- ✓ Git v2.17.1 detected. Minimum is v2.7.4.
- ✓ 985 MB of system memory available. Minimum is 768 MB.
- ✓ config.yml is writable by the setup process.

✓ Looks good! No issues so far.

Enter general information about your wiki and click the "**Continue**" button:



Site Title

Wiki

The site title will appear in the top left corner on every page and within the window title bar.

Host

http://localhost

The full URL to your wiki, without the trailing slash. E.g.: http://wiki.domain.com. Note that sub-folders are not supported.

Port

80

The port on which Wiki.js will listen to. Usually port 80 if connecting directly, or a random port (e.g. 3000) if using a web server in front of it.
Set $(PORT) to use PORT environment variable.

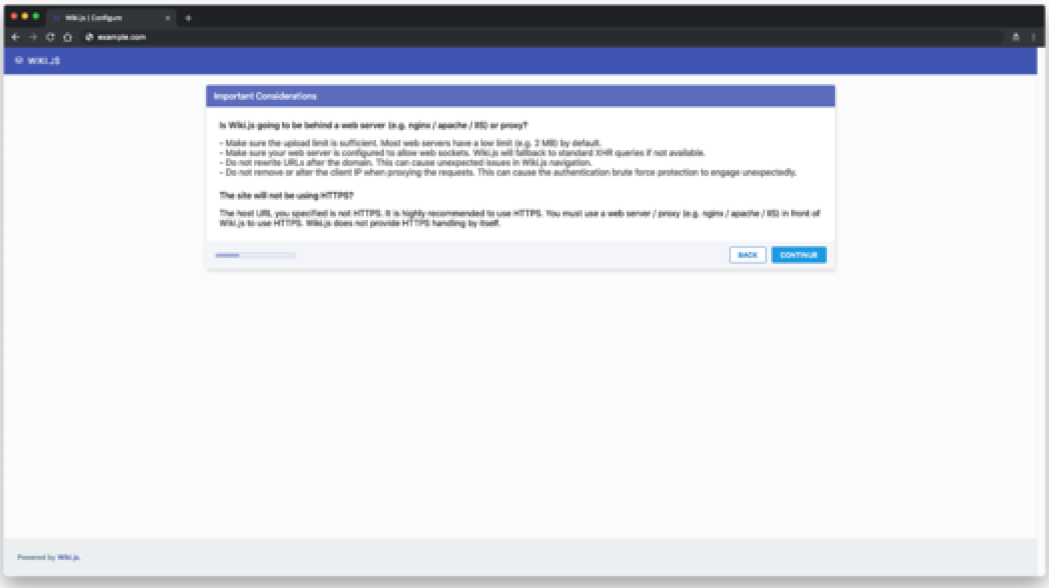
Site UI Language

English

The language in which navigation, help and other UI elements will be displayed.

☐ Public Access Should the site be accessible (read only) without login.

Read the "Important Consideration" notice and click "**Continue**" for the next step:



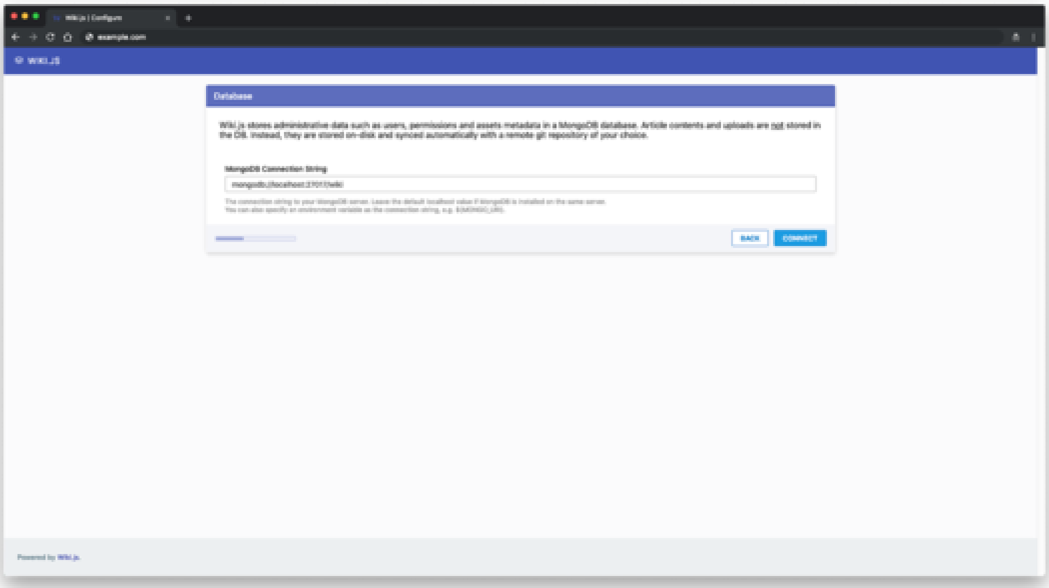
Is Wiki.js going to be behind a web server (e.g. nginx / apache / IIS) or proxy?

- Make sure the upload limit is sufficient. Most web servers have a low limit (e.g. 2 MB) by default.
- Make sure your web server is configured to allow web sockets. Wiki.js will fallback to standard XHR queries if not available.
- Do not rewrite URLs after the domain. This can cause unexpected issues in Wiki.js navigation.
- Do not remove or alter the client IP when proxying the requests. This can cause the authentication brute force protection to engage unexpectedly.

The site will not be using HTTPS?

The host URL you specified is not HTTPS. It is highly recommended to use HTTPS. You must use a web server / proxy (e.g. nginx / apache / IIS) in front of Wiki.js to use HTTPS. Wiki.js does not provide HTTPS handling by itself.

Next, connect to the database and continue:

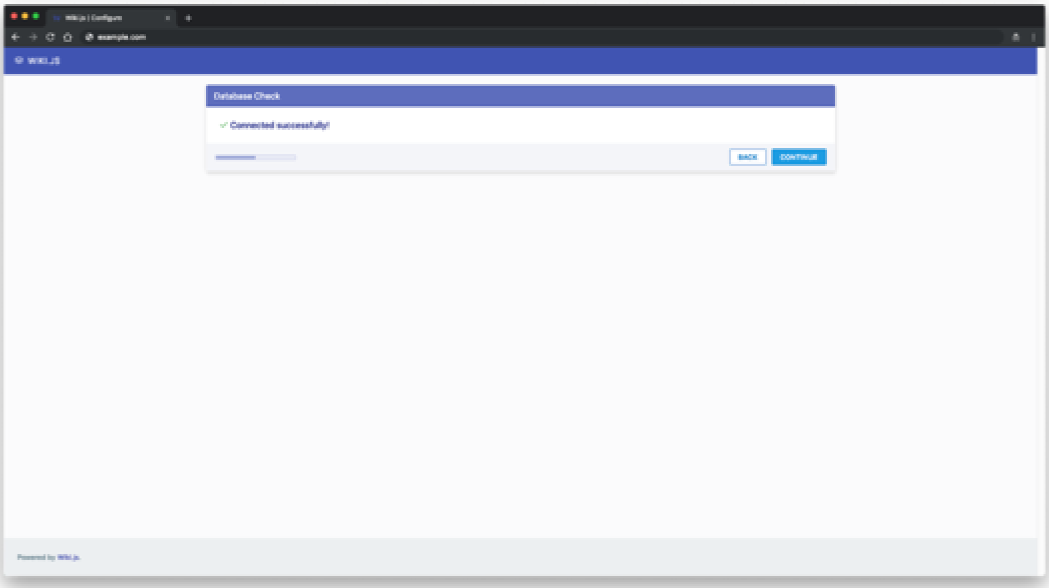


Wiki.js stores administrative data such as users, permissions and assets metadata in a MongoDB database. Article contents and uploads are not stored in the DB. Instead, they are stored on-disk and synced automatically with a remote git repository of your choice.

mongodb://localhost:27017/wiki

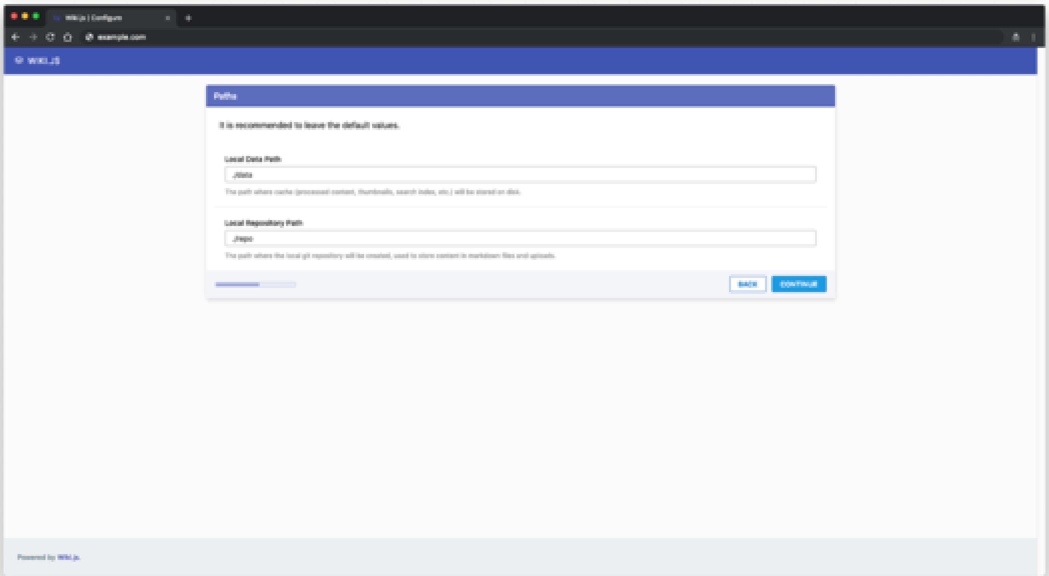
The connection string to your MongoDB server. Leave the default localhost value if MongoDB is installed on the same server.
You can also specify an environment variable as the connection string, e.g. $(MONGO_URI).

You should see a message that Wiki.js has been successfully connected to the database. Click the "**Continue**" button:



✓ Connected successfully!

Set paths and continue:



Paths

It is recommended to leave the default values.

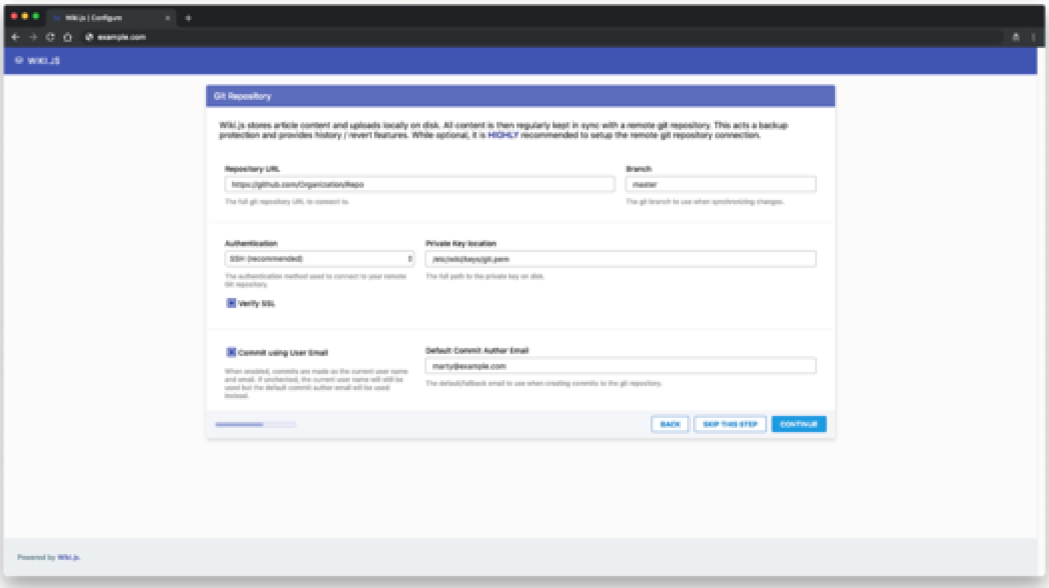
Local Data Path: ./data

The path where cache (processed content, thumbnails, search index, etc.) will be stored on disk.

Local Repository Path: ./repo

The path where the local git repository will be created, used to store content in markdown files and uploads.

Set up remote Git repo if you want or skip this step. This step is optional but highly recommended:



Git Repository

Wiki.js stores article content and uploads locally on disk. All content is then regularly kept in sync with a remote git repository. This acts a backup protection and provides history / revert features. While optional, it is HIGHLY recommended to setup the remote git repository connection.

Repository URL: https://github.com/Organization/Repo

The full git repository URL to connect to.

Branch: master

The git branch to use when synchronizing changes.

Authentication: SSH (recommended)

The authentication method used to connect to your remote Git repository.

Private Key location: /etc/wiki/keys/git.pem

The full path to the private key on disk.

☒ Verify SSL

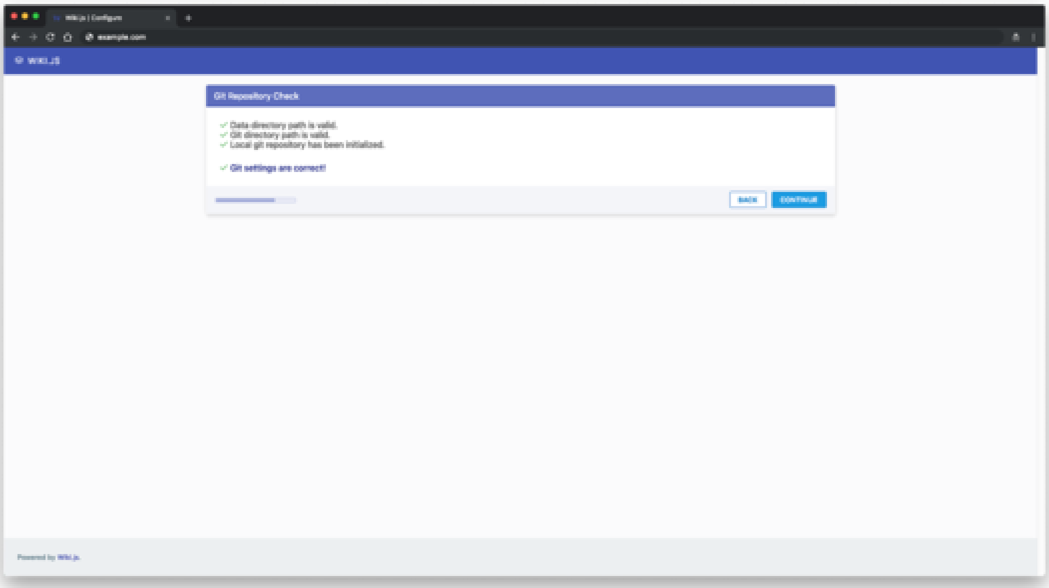
☒ Commit using User Email

When enabled, commits are made as the current user name and email. If unchecked, the current user name will still be used but the default commit author email will be used instead.

Default Commit Author Email: marty@example.com

The default/fallback email to use when creating commits to the git repository.

Next, click the "Continue" button:

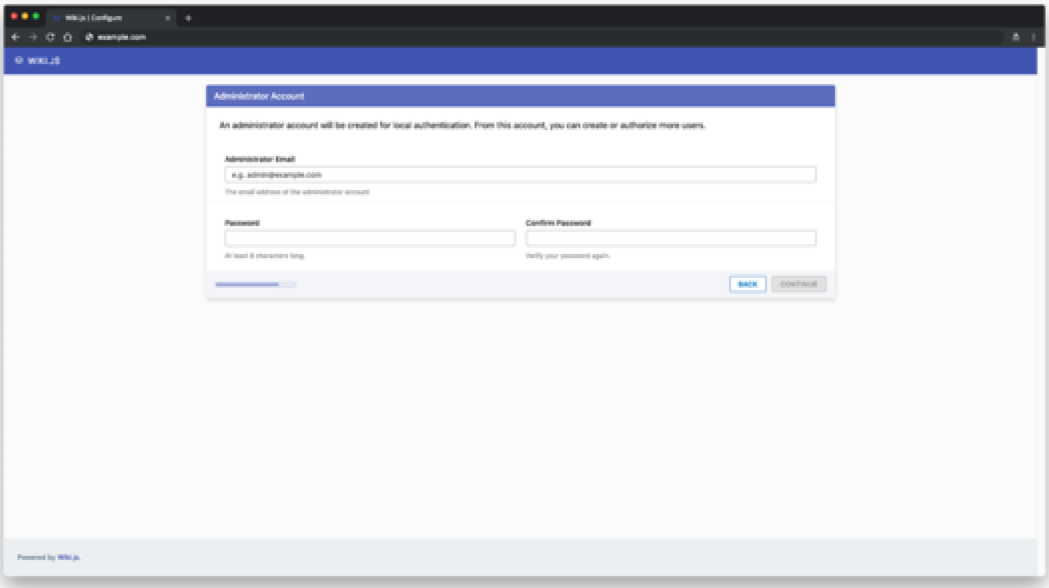


Git Repository Check

✓ Data directory path is valid.
✓ Git directory path is valid.
✓ Local git repository has been initialized.

✓ Git settings are correct!

Create an admin account and click the "**Continue**" button:



Administrator Account

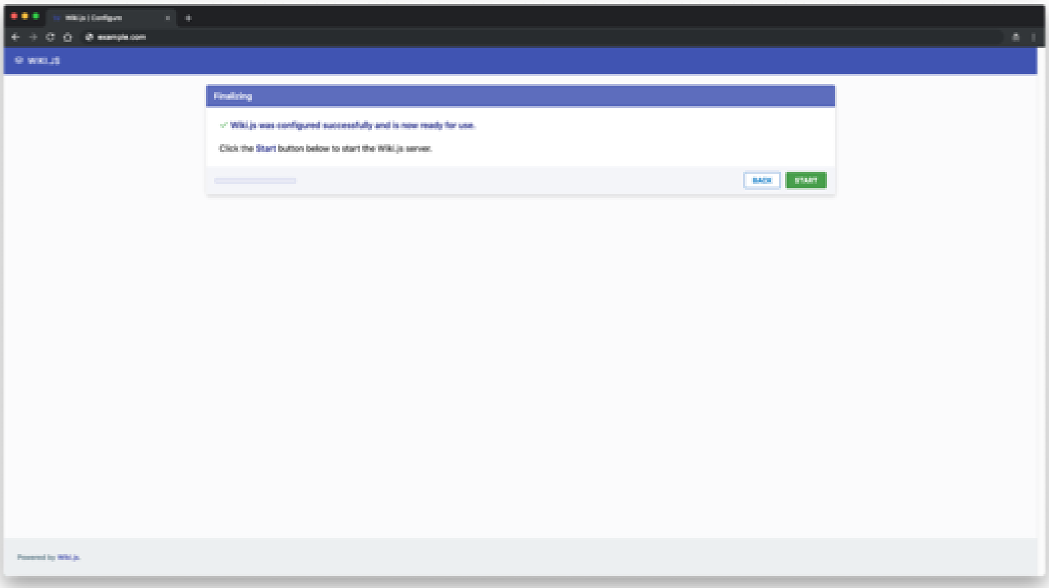
An administrator account will be created for local authentication. From this account, you can create or authorize more users.

Administrator Email: e.g. admin@example.com
The email address of the administrator account

Password:
At least 8 characters long.

Confirm Password:
Verify your password again.

And finally, start the Wiki.js:

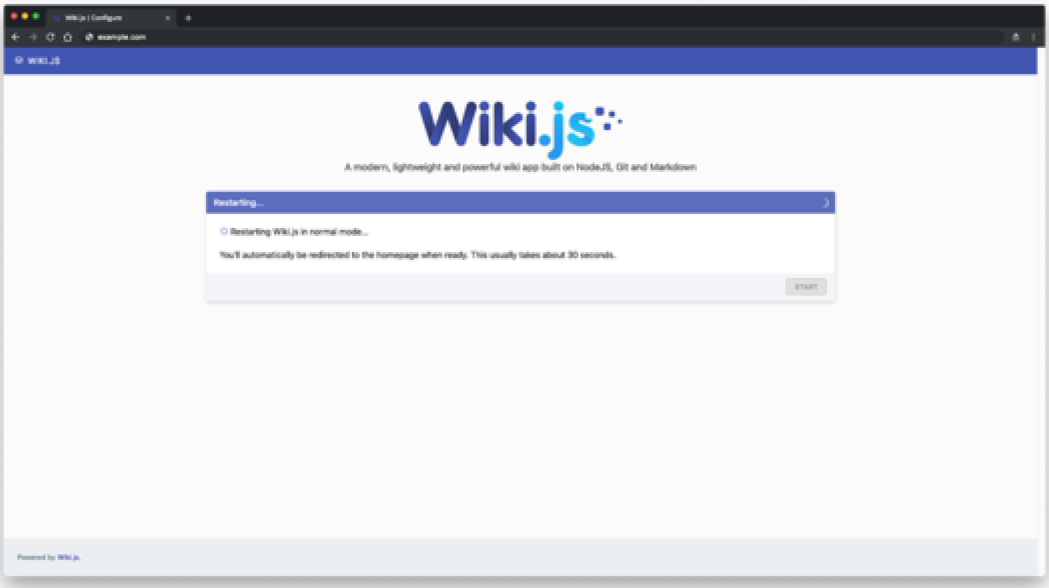


Finalizing

✓ Wiki.js was configured successfully and is now ready for use.

Click the Start button below to start the Wiki.js server.

Wait around 30 seconds and you should be redirected to Wiki.js homepage:



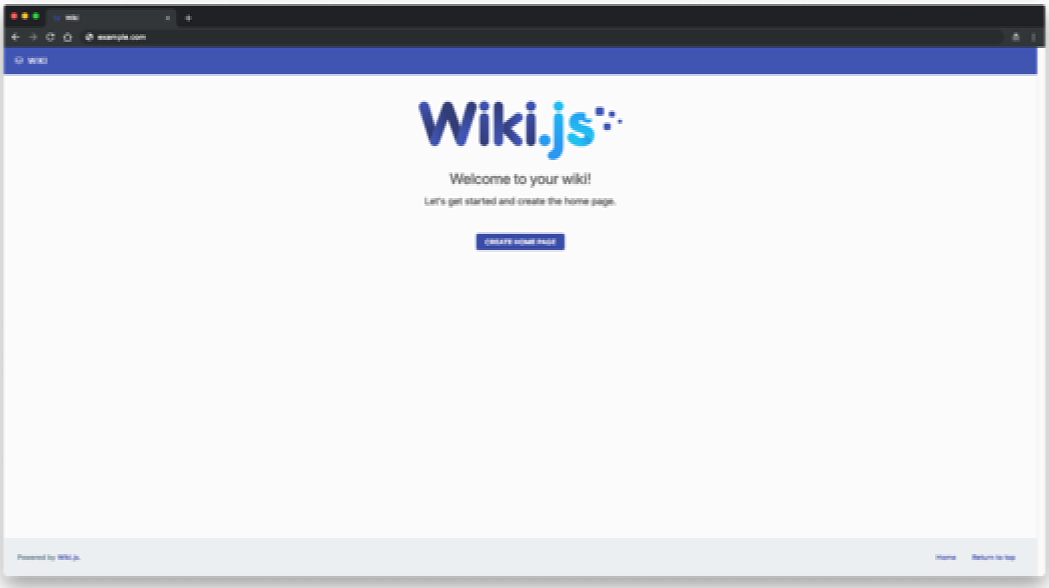
A modern, lightweight and powerful wiki app built on NodeJS, Git and Markdown

Restarting...

Restarting Wiki.js in normal mode...

You'll automatically be redirected to the homepage when ready. This usually takes about 30 seconds.

The installation is completed. You should wiki welcome page:



Welcome to your wiki!

Let's get started and create the home page.

Step 6 - Setup PM2 Process Manager

By default, Wiki.js will not start automatically after a system reboot. In order to make it start on boot, we need to setup PM2 process manager. PM2 comes bundled with Wiki.js as a local npm module, so we don't need to install PM2 globally.

Tell PM2 to configure itself as a startup service by running:

```
/var/www/wiki.js/node_modules/pm2/bin/pm2 startup
```

Finally, save the current PM2 configuration by running the command:

```
/var/www/wiki.js/node_modules/pm2/bin/pm2 save
```

Your Wiki.js now runs as a background process, using PM2 as its process manager.