



# Jekyll Tutorial: How To Create a Static Website

# Ship and manage your web projects faster

Deploy your projects on Google Cloud Platform's top tier infrastructure. You'll get 25+ data centers to choose from, 24/7/365 expert support, and advanced security with DDoS protection.

Try for free

■ Data centers (25) ■ CDN locations (200)

In today's digital age, having a website is essential for establishing an online presence, promoting your brand, and reaching out to potential customers. However, creating a website can be a daunting task.

This is where [static site generators](#) (SSGs) come in — they make it easy to create beautiful, fast-loading websites without the need for complex backend systems or databases.

In this article, you will be introduced to one of the popular SSGs — Jekyll, learn how it works, and how you can create a [static website](#) with it.

Here's a [live demo of the blog site](#) you'll be building with Jekyll.

# Joe's Blog

This is the official blog of Joe. Here you can find all the latest news and updates.

## How to Read Books: Tips and Strategies for Maximum Learning

Reading books is one of the most powerful ways to learn new information, gain new perspectives, and expand your

— Blog website built with Jekyll

You can access the [project's GitHub repository](#) if you'd like to take a closer look.

## What Is Jekyll?

Jekyll is a free open-source SSG that is built and runs on the [Ruby](#) programming language. You don't need to understand how Ruby works to use Jekyll; you only need to have Ruby installed on your machine.

Jekyll can be used to build various types of [static sites](#) such as a personal blog, portfolio website, and documentation website without needing a database or using a content management system like [WordPress](#).

Here's what makes Jekyll stand out among SSGs:

1. **Easy to use:** Jekyll uses plain text files and markdown syntax to create and manage content, which means you don't need to have knowledge of [HTML](#) or [CSS](#) to get started.
2. **Fast and secure:** Jekyll does not deal with any server-side database or scripting, meaning there is less risk of vulnerabilities and attacks. It generates static HTML files making your website incredibly fast and secure.
3. **Customizable:** Jekyll is highly customizable, allowing you to use layouts and templates or even create plugins to extend functionality.
4. **Easy to deploy:** Jekyll generates static HTML files that can be deployed to a web server or hosting provider without needing a dynamic content management system.
5. **Backed by a large community:** Jekyll has a large community of users and developers, which means plenty of resources are available if you need help or want to extend the functionality of your site.

## How To Install Jekyll

You first need to install Ruby on your machine before you can proceed to install the Jekyll as stated in the [Jekyll documentation](#).

### How To Install Jekyll on macOS

By default, Ruby comes preinstalled with macOS, but it's not recommended you use such a version of Ruby to install Jekyll because it's old. For example, on Ventura, Apple's latest operating system, the version of Ruby that comes preinstalled is 2.6.10, of which the latest version is 3.1.3 (as of when this article is written).

To fix this, you'd need to install Ruby correctly using a version manager such as [chruby](#). You'd need to install [Homebrew](#) on your Mac first using the command below in your terminal:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/ins
```

Once the installation is successful, quit and restart Terminal, then check if Homebrew is ready to go by running this command:

```
brew doctor
```

If you get “**Your system is ready to brew**”, you can now move on to install `chruby` and **ruby-install** with the command below:

```
brew install chruby ruby-install
```

You can now install ruby’s latest version which is 3.1.3 using the `ruby-install` package you just installed:

```
ruby-install 3.1.3
```

This will take a few minutes. Once it’s successful, configure your shell to automatically use `chruby` with the command below:

```
echo "source $(brew --prefix)/opt/chruby/share/chruby/chruby.sh" >> ~/.zshrc
echo "source $(brew --prefix)/opt/chruby/share/chruby/auto.sh" >> ~/.zshrc
echo "chruby ruby-3.1.3" >> ~/.zshrc
```

You can now quit and relaunch your terminal, then check that everything works by running this command:

```
ruby -v
```

It should say **ruby 3.1.3**.

You now have Ruby's latest version installed on your machine. You can now proceed to install the latest Jekyll and bundler gem:

```
gem install jekyll bundler
```

## How To Install Jekyll on Windows

To install Ruby and Jekyll on a Windows machine, you'd use the [RubyInstaller](#). This can be done by downloading and installing a **Ruby+Devkit** version from [RubyInstaller Downloads](#) and using the default options for installation.

On the last stage of the installation wizard, run the `ridk install` step — which is used to install gems. Read more via the [RubyInstaller Documentation](#).

From the options, choose MSYS2 and MINGW development toolchain. Open a new command prompt window and install Jekyll and Bundler using the command below:

```
gem install jekyll bundler
```

## How To Verify That Jekyll Is Installed Correctly

To verify that Jekyll is installed correctly on your machine, open your terminal and run the following command:

```
jekyll -v
```

If you see the version number, it means that Jekyll is installed and working correctly on your system. You're now ready to use Jekyll!

## Jekyll Commands and Configuration

Before we start creating and customizing a static site with Jekyll, it's nice to get familiar with its various CLI commands and configuration file parameters.

### Jekyll CLI Commands

Here are some of the popular Jekyll CLI commands. You don't need to memorize them, but just know that they exist, and feel free to check back for what any of them does when you notice their usage later in this article.

- `jeekyll build`: Generates the static site in the `_site` directory.
- `jeekyll serve`: Builds the site and starts a web server on your local machine, allowing you to preview the site in your browser at `http://localhost:4000`.
- `jeekyll new [site name]`: Creates a new Jekyll site in a new directory with the specified site name.
- `jeekyll doctor`: Outputs any configuration or dependency issues that may be present.
- `jeekyll clean`: Deletes the `_site` directory, which is where the generated site files are stored.
- `jeekyll help`: Outputs the help documentation for Jekyll.
- `jeekyll serve --draft`: Generates and serves your Jekyll site, including any posts that are in the `_drafts` directory.

You can also append some options to these commands. See a full list of [Jekyll commands and options in the Jekyll documentation](#).

## Jekyll Configuration File

The Jekyll configuration file is a YAML file named `_config.yml` that contains settings and options for your Jekyll site. It is used to configure various aspects of your site, including the site title, description, URL, and other settings.

Here are some of the most commonly used configuration options:

- **title**: The title of your site.
- **description**: A short description of your site.
- **url**: The base [URL](#) of your site.
- **baseurl**: The subdirectory of your site, if it is hosted in a subdirectory of a domain.
- **permalink**: The URL structure for your posts and pages.
- **exclude**: A list of files or directories to exclude from the site generation process.
- **include**: A list of files or directories to include in the site generation process.
- **paginate**: The number of posts to show per page when using pagination.
- **plugins**: A list of Jekyll plugins to load.
- **theme**: By default, this is set to `minima`. You can use any [other theme](#) by setting its name and implementing other settings we will learn about later in this article.



You can also create custom variables in your configuration file and use them in your site's templates, layouts, and includes. For example, you could create a variable called `author_name` and then use it in your templates like this: `{{ site.author_name }}`.

To modify your Jekyll configuration file, open the `_config.yml` file in your Jekyll project directory in a text editor and make changes.

**Note:** Any changes you make to the configuration file will take effect the next time you generate your site with `jekyll build` or `jekyll serve`.

## How Do You Create a Static Website on Jekyll?

Now that you have Jekyll installed on your machine, it's now possible to create a Jekyll static website with one command within a few seconds. Open your terminal and run this command:

```
jekyll new joels-blog
```

Ensure you replace “joels-blog” with your preferred site name.

```
joelolanle@Joels-Air codes % jekyll new joels-blog
Running bundle install in /Users/joelolanle/Documents/codes/joels-blog...
Bundler: Fetching gem metadata from https://rubygems.org/.....
Bundler: Resolving dependencies...
Bundler: Using public_suffix 5.0.1
Bundler: Using bundler 2.3.26
Bundler: Using colorator 1.1.0
Bundler: Using concurrent-ruby 1.2.2
Bundler: Using eventmachine 1.2.7
Bundler: Using http_parser.rb 0.8.0
Bundler: Using ffi 1.15.5
Bundler: Using forwardable-extended 2.6.0
Bundler: Using google-protobuf 3.22.0 (arm64-darwin)
Bundler: Using rb-fsevent 0.11.2
Bundler: Using rexml 3.2.5
Bundler: Using safe_yaml 1.0.5
Bundler: Using unicode-display_width 2.4.2
Bundler: Using liquid 4.0.4
Bundler: Using mercenary 0.4.0
Bundler: Using rouge 4.1.0
Bundler: Using webrick 1.8.1
Bundler: Using pathutil 0.16.2
Bundler: Using addressable 2.8.1
Bundler: Using em-websocket 0.5.3
Bundler: Using sass-embedded 1.58.3 (arm64-darwin)
Bundler: Using kramdown 2.4.0
Bundler: Using terminal-table 3.0.2
Bundler: Using i18n 1.12.0
Bundler: Using rb-inotify 0.10.1
Bundler: Using jekyll-sass-converter 3.0.0
Bundler: Using kramdown-parser-gfm 1.1.0
Bundler: Using listen 3.8.0
Bundler: Using jekyll-watch 2.2.1
Bundler: Using jekyll 4.3.2
Bundler: Using jekyll-feed 0.17.0
Bundler: Using jekyll-seo-tag 2.8.0
Bundler: Using minima 2.5.1
Bundler: Bundle complete! 7 Gemfile dependencies, 33 gems now installed.
Bundler: Use 'bundle info [gemname]' to see where a bundled gem is installed.
New jekyll site installed in /Users/joelolanle/Documents/codes/joels-blog.
joelolanle@Joels-Air codes %
```

— Jekyll static website

Next, navigate to the website folder. You will notice a basic Jekyll site structure that includes directories and files like these:

```
??? _config.yml
??? _posts
??? Gemfile
??? Gemfile.lock
??? index.md
??? README.md
```

Here's what each of these directories and files is for:

- **\_config.yml:** The Jekyll configuration file containing your site's settings and options.
- **\_posts:** A directory that contains your site's content (can be blog posts). These can be Markdown or HTML files with a specific file naming convention: *YEAR-MONTH-DAY-title.MARKUP*.
- **Gemfile and Gemfile.lock:** Bundler uses these files to manage the Ruby gems on which your site relies.
- **index.md:** The default homepage for your site, generated from a Markdown or HTML file.

But there are more files/folders that can be used to customize your Jekyll website. These folders include:

- **\_includes:** A directory that contains reusable HTML snippets that can be included in your layouts and pages. Such as navbar, footer, e.t.c.
- **\_layouts:** A directory that contains HTML layout templates that define the structure of your pages.
- **assets:** A directory that contains any files that are used by your site, such as images and CSS files.
- **\_sass:** A directory that contains Sass files that can be used to generate CSS for your site.

This file structure provides a solid foundation for a Jekyll project, but you can modify it as needed to suit the specific needs of your project.

## Quick Start Option: Use Our GitHub Template

As an alternative to starting your project using the Jekyll CLI, you can create a Git repository using Kinsta's ["Hello World" Jekyll template](#) on GitHub. Select **Use this template > Create a new repository** to copy the starter code into a new repository within your own GitHub account.

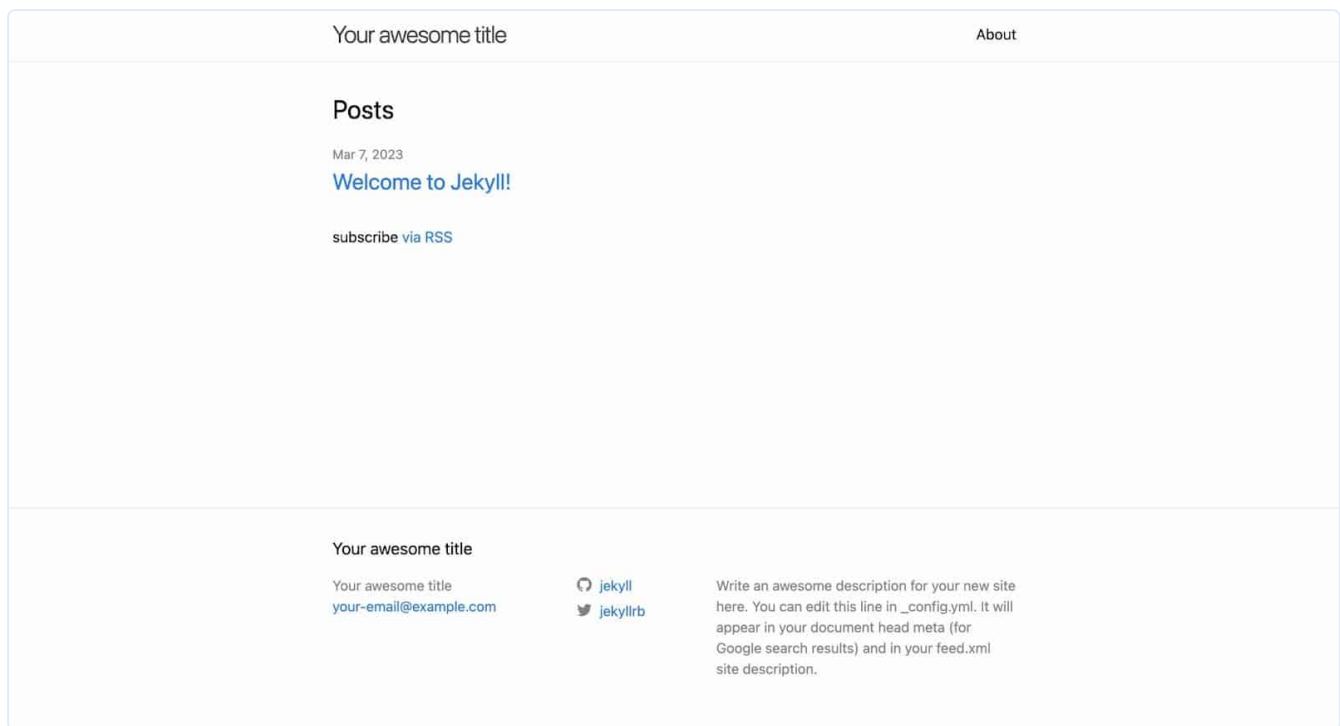
## How To Preview a Jekyll Site

You now have a Jekyll site, but how can you preview the website to see what it looks like before you maybe start customizing it to suit your needs? Open your terminal and move into your project's directory, then run the following command:

```
jeekyll serve
```

This will generate a **\_site** folder that includes all static files generated from your project. It will also start the Jekyll server, and you can preview your site via **http://localhost:4000**.

If you visit the URL in your web browser, you should see your Jekyll site with the minima theme:



— Default appearance

## How To Customize a Jekyll Site

When you create a site with Jekyll and use a theme, you can customize the site to suit your needs. For example, you may want to add new pages, change a page's layout, or adjust how some items are displayed. All these are possible with Jekyll.

### How Front Matter Works in Jekyll

When you open each page or blog post of your project folder, you will notice a block of information within three dashes (—) at the top of the page. This is referred to as **front matter**.

It is a metadata format used in Jekyll, to store information about a page or post — it can be written in either YAML or JSON.

```
---
title:  "Welcome to Jekyll!"
description: "Introduction to what Jekyll is about and how it works"
date:    2023-03-07 16:54:37 +0100
---
```

In the example above, the front matter includes variables, such as the post's title, description, and date. These variables can be used in the page or post's layout or content.

Jekyll parse's the front matter and uses it to generate the output HTML for your site. You can use front matter to specify various options, such as layout, permalink, published status, etc.

## How To Configure Default Front Matter

You can also configure default front matter, so you don't need to define the same front matter for similar files. For example, if each blog post uses the same author's name and layout. You can define a custom front matter in your **\_config.yml** file to serve all your blog posts.

The way it's structured it's a little bit complex, but here is what it will look like. Paste this below the last configuration in your **\_config.yml** file:

```
defaults:
-
  scope:
    path: "posts" # empty string means all files
  values:
    layout: "post"
    author: "John Doe"
```

When you now run the `jeekyll serve` command, you will notice that even when you don't define the layout and author on each post, you still have access to them within your files.

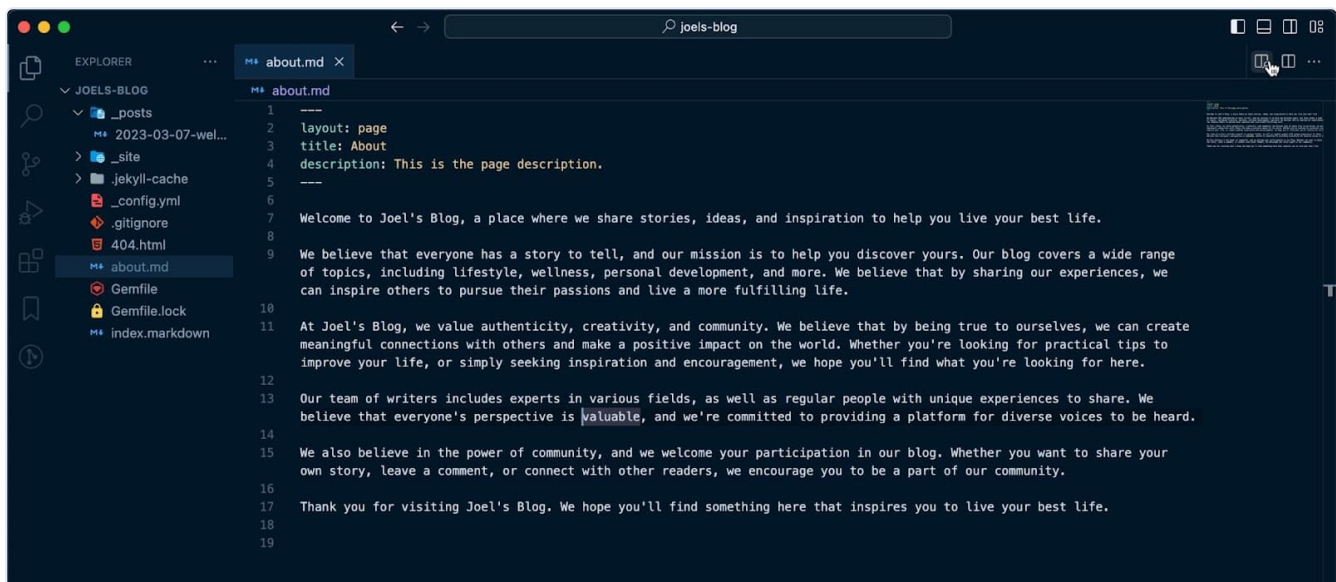
**Note:** When you don't define a path, all files will use the defined front matter values.

## Creating Pages in Jekyll

When you create a file in the root directory of your project, it is considered a page. The name you give to the file is most times used in the URL, so you'll name each page file a name that resonates well.

For example, if you want to create a page with the URL <https://example.com/about>, create a file named **about.html** or **about.md**. The file extension determines the markup language to use for the page content (HTML or Markdown).

Once you have created a file, add appropriate front matter and content. Save the file and refresh your Jekyll site in the browser. The new page should now be accessible at the URL corresponding to the filename.



— Front matter and content

## Creating Layouts in Jekyll

You can use Layouts to define the structure and design of your site's pages and posts. This is usually done majorly with HTML code. You can include a header, footer, meta information with the page's metadata.

The first step would be to create a **\_layouts** folder in your project's root directory. Then create a file for each layout — the file should have a descriptive name that reflects the purpose of the layout. For example, you might create a file named **page.html** to define the page layout for all pages on your site.

It's best to define the overall structure of your layouts with HTML or another markup language.

You can include placeholders for any dynamic content that will be inserted into the layout, such as the page title, content, and metadata. For example, you might create a basic layout that includes a header, footer, and content area like this:

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ page.title }}</title>
  </head>
  <body>
    <header>
      <!-- Navigation menu goes here -->
    </header>
    <main>{{ content }}</main>
    <footer>
      <!-- Footer content goes here -->
    </footer>
  </body>
</html>
```



In this example, the `{{ page.title }}` and `{{ content }}` placeholders will be replaced with the actual title and content of the page being rendered.

At this point, any page or post that uses `page` as its layout value in its front matter will have this layout. You can create multiple layouts within the **\_layouts** folder and style your layouts however you like. You can also override a theme's layout by defining a layout with a similar name.

## How the **\_includes** Folder Works in Jekyll

The **\_includes** folder contains reusable snippets of code that you can include in different parts of your website. For example, you can create a **navbar.html** file in your includes folder and add it to your **page.html** layout file using the `{% include %}` tag.

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ page.title }} </title>
  </head>
  <body>
    <header>{% include navbar.html %} </header>
    <main>{{ content }} </main>
    <footer>
      <!-- Footer content goes here -->
    </footer>
  </body>
</html>
```

At build time, Jekyll will replace the tag with the content of **navbar.html**.

The **\_includes** folder can contain any type of file, such as HTML, Markdown, or Liquid files. The principal aim is to keep your code DRY (Don't Repeat Yourself) by allowing you to reuse

code across your site.

## Looping Through Posts in Jekyll

You might want to create a dedicated blog page to hold all your blog posts, this means you'd want to fetch all the posts on your site and loop through them. With Jekyll it's easy to achieve using the `{% for %}` tag.

All posts on any Jekyll website are stored in the `site.posts` variable. You can loop through and use the `{{ post.title }}` Liquid variable to output the title of each post this way:

```
{% for post in site.posts %}
  <h2>{{ post.title }}</h2>
{% endfor %}
```

You can also use additional Liquid variables to output other information about each post, such as the post's date or author:

```
{% for post in site.posts %}
  <h2>{{ post.title }}</h2>
  <p>Published on {{ post.date | date: "%B %-d, %Y" }} by {{ post.author }}
{% endfor %}
```

Notice that in the example above, the date Liquid filter formats each post's date in a more human-readable format.

Now, you have an idea of some basic formatting that can be done to your Jekyll site. But you may not need to use all these to build a Jekyll site from scratch because you can always search for a theme that meets your needs and add it to your Jekyll site.

## How To Add a Theme To a Jekyll Site

There are various easy-to-add themes, but the good thing is that for each theme, there is always clear information about how to install them in the ReadMe file on GitHub. You can decide to clone the theme, or if it is a [gem-based theme](#), the process is easier.

For this article, you would install a [blog theme](#) and customize it to have a [blog site deployed to Kinsta](#). This is a gem-based theme and you can access its [source code](#) and [instructions on GitHub](#).

To add a gem-based theme, open your site's Gemfile and add the gem for the theme you want to use. The theme we will use is the **jekyll-theme-clean-blog**. You can replace the default **minima** theme in the Gemfile:

```
gem "jekyll-theme-clean-blog"
```

Run the `bundle install` command in your site's directory to install the theme's gem and its dependencies.

In your site's `_config.yml` file, add the following line to specify the theme you want to use:

```
theme: jekyll-theme-clean-blog
```

At this point, your theme is ready for use.

You will want to remove all layouts in the `_layouts` directory to avoid them overriding the theme's layout.

You can then find the layout names for your files in the [theme's documentation](#). For example, if you're using the **jekyll-theme-clean-blog** theme, the layout names for the **index.html** file is **home**, other pages is **page**, and all posts is **post**.

Finally, run `jekyll serve` to build and serve your site using the new theme.



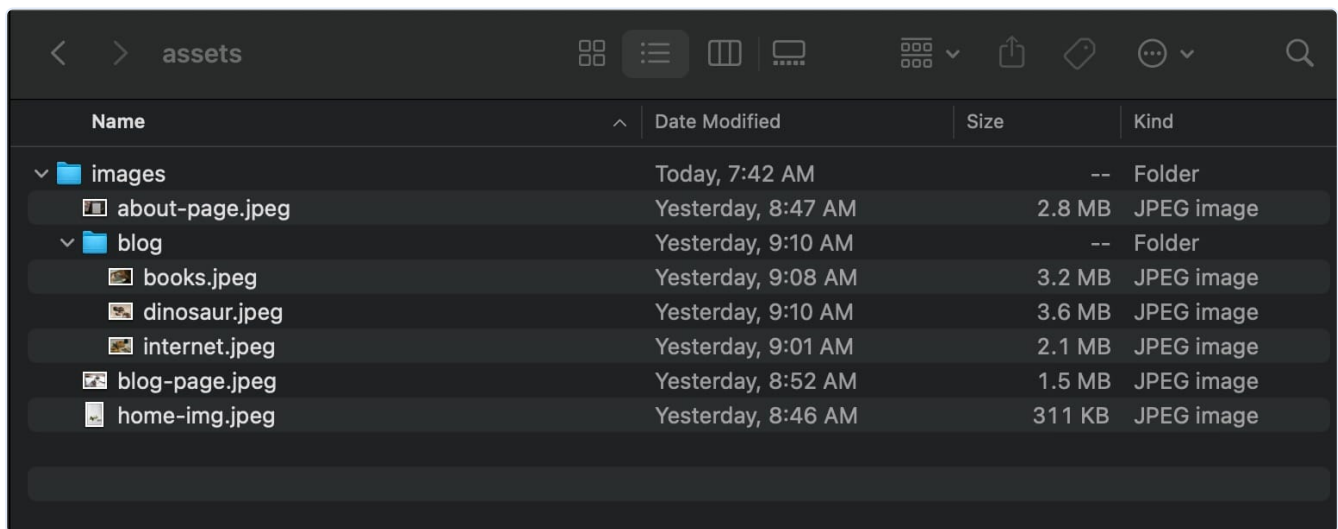
— Jekyll blog static site

That's it! Your Jekyll site should now be using the new gem-based theme you added. You can customize the theme further by modifying its layout in your site's `_layouts` directories.

## Customize a Jekyll Theme

You now have your theme added to your site, the next action would be to customize the site to meet your needs and work as it is intended to work. For example, the images for each page and post are not displayed rather it displays a grey background.

You can fix this by adding a front matter option to each page and post by specifying the path to an image you wish to use. In Jekyll, assets such as images are stored in the **assets** folder. In this folder, you can decide to create sub-folders to organize your images.



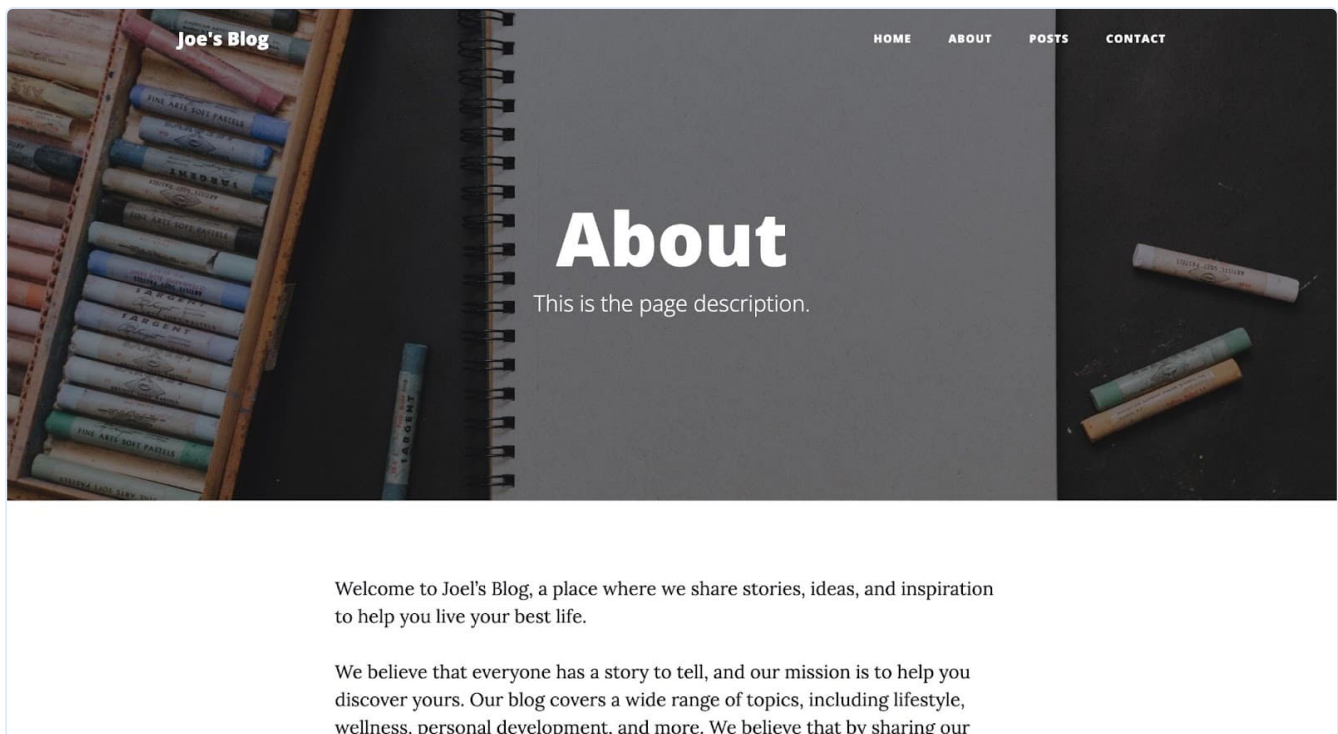
— Image folder

You can now add a background option to the front matter block and a path to its image. For example, on the about page, this is the front matter:

```
---  
layout: page  
title: About
```

```
description: This is the page description.  
permalink: /about/  
background: '/assets/images/about-page.jpeg'  
---
```

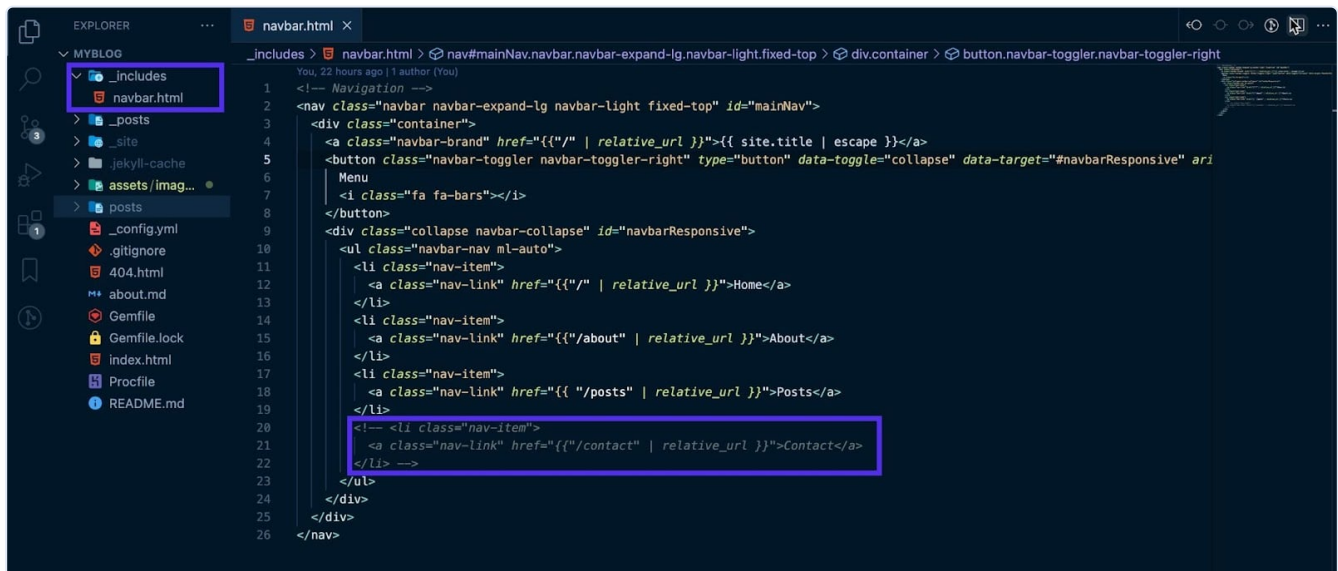
Do this for all pages and posts and your page will look like this:



— Background image is shown on the About page

Another customization you can do is adjust the navbar options. For example, you may not need a contact page, meaning you should remove its link from the navbar options. You can do this by studying the [theme's source code](#), noticing the file responsible for the nav links, and replicating the file exactly in your project, removing the option you don't need.

The nav links are in the [\\_includes folder's navbar.html file](#). You can create this file, paste the code from your source code, and remove the contact option or add any new option you wish.



```
1 <!-- Navigation -->
2 <nav class="navbar navbar-expand-lg navbar-light fixed-top" id="mainNav">
3   <div class="container">
4     <a class="navbar-brand" href="{%"/ | relative_url %}">{{ site.title | escape }}</a>
5     <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target="#navbarResponsive" ari
6       <i class="fa fa-bars"></i>
7     </button>
8     <div class="collapse navbar-collapse" id="navbarResponsive">
9       <ul class="navbar-nav ml-auto">
10        <li class="nav-item">
11          <a class="nav-link" href="{%"/ | relative_url %}">Home</a>
12        </li>
13        <li class="nav-item">
14          <a class="nav-link" href="{%"/about | relative_url %}">About</a>
15        </li>
16        <li class="nav-item">
17          <a class="nav-link" href="{%"/posts | relative_url %}">Posts</a>
18        </li>
19        <li class="nav-item">
20          <a class="nav-link" href="{%"/contact | relative_url %}">Contact</a>
21        </li>
22      </ul>
23    </div>
24  </div>
25 </nav>
```

— Customizing the navbar from the Jekyll theme

When you save your project, you will notice the nav options will be customized:



— Fully customized nav bar

Finally, one last customization would be creating a posts page that will hold all your blog posts — which means you will loop through all the posts on this page.

Create a file, **posts.html** and paste the following code:

```
---
layout: page
title: Blog
description: Expand your knowledge and stay informed with our engaging bl
background: '/assets/images/blog-page.jpeg'
---

{% for post in site.posts %}

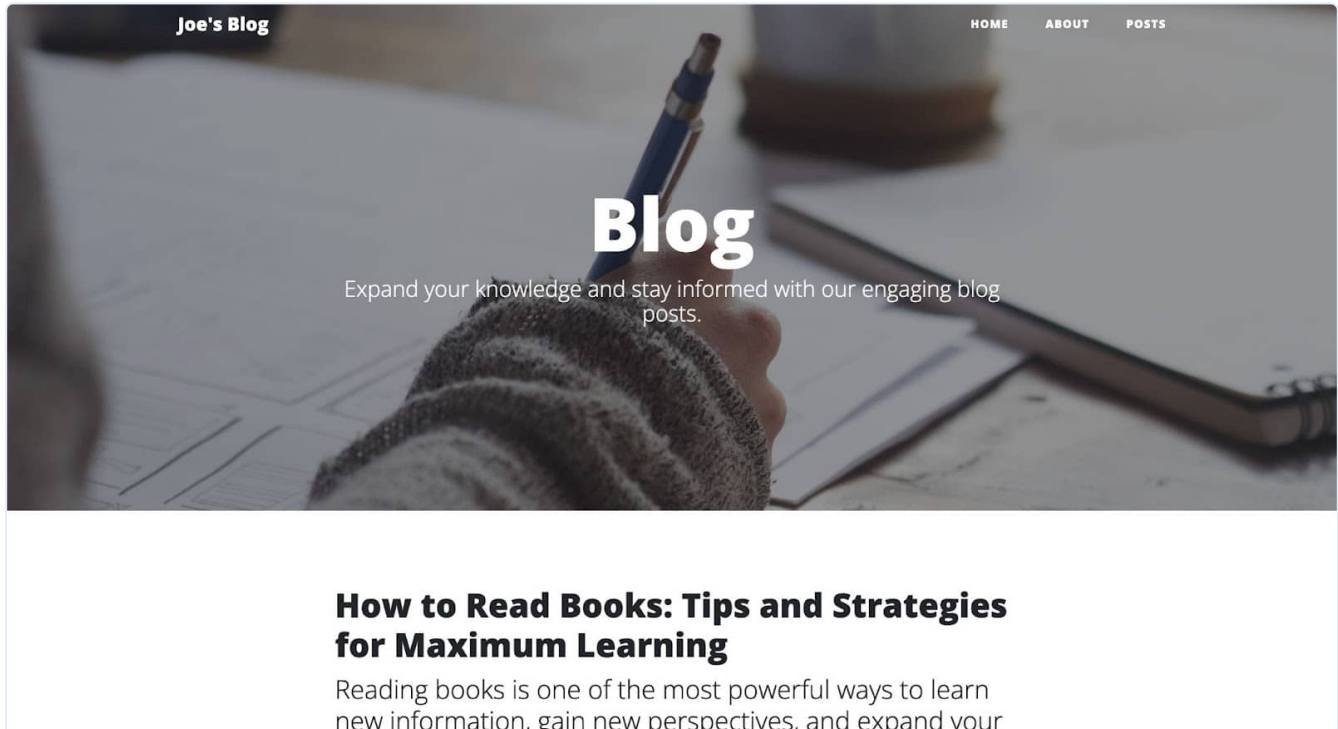
<article class="post-preview">
  <a href="{{ post.url | prepend: site.baseurl | replace: '//', '/' }}"
    <h2 class="post-title">{{ post.title }}</h2>
    {% if post.subtitle %}
    <h3 class="post-subtitle">{{ post.subtitle }}</h3>
    {% else %}
    <h3 class="post-subtitle">
      {{ post.excerpt | strip_html | truncatewords: 15 }}
    </h3>
    {% endif %}
  </a>
  <p class="post-meta">
    Posted by {% if post.author %} {{ post.author }} {% else %} {{ si
    }} {% endif %} on {{ post.date | date: '%B %d, %Y' }} · {% includ
    read_time.html content=post.content %}
  </p>
</article>

<hr />

{% endfor %}
```



Feel free to adjust the background image to reflect your saved image. In the code above, you are looping through all the posts to display all posts on this page. This is what the posts page will look like when saved.



— Posts page

## How To Add Content To a Jekyll Site

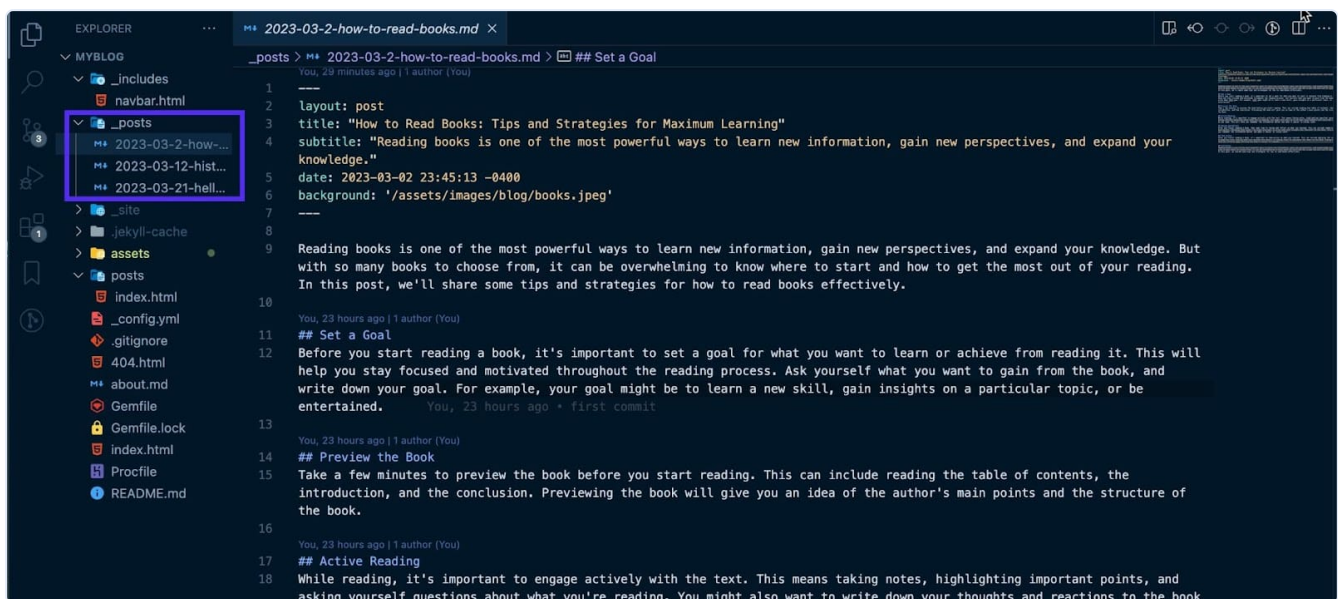
You have now created a Jekyll site and configured the site to meet your needs. The last step would be to add content or learn how content can be added to a Jekyll site.

All content is stored in the `_posts` folder. Each content is stored in a file with a similar naming convention of **YYYY-MM-DD-title.md** (or **.html** for HTML files). For example, if you want to create a post called "My First Post", create a **2023-03-08-my-first-post.md** in the `_posts` directory.

Next, for each post/content file, ensure you add front matter about the post at the top. This will include the layout, title, description, date and other information.

```
---
layout: post
title: "How to Read Books: Tips and Strategies for Maximum Learning"
subtitle: "Reading books is one of the most powerful ways to learn new in
date: 2023-03-02 23:45:13 -0400
background: '/assets/images/blog/books.jpeg'
---
```

Then you can add your content below the front matter block with HTML tags or markdown syntax.



— Adding new posts to the \_posts folder

Save the file, and Jekyll will automatically build and include it in your site.

## How To Deploy Your Jekyll Site on Kinsta

Kinsta is a cloud platform that allows you to [host static websites](#), including Jekyll. This can be done by setting up some configurations, pushing your codes to GitHub, and finally deploying to Kinsta.

### Prerequisites: Configuring Your Jekyll Site

Check your **Gemfile.lock** file and be sure it includes a platform for all devices. To be sure all the platforms are properly configured, run the following command:

```
bundle lock --add-platform arm64-darwin-22 x64-mingw-ucrt x86_64-linux
```

Then you can proceed to create a **Procfile** — this file specifies the commands that are executed when your site deploys. This file automatically updates the commands to be executed in the Process tab of MyKinsta. This is the command to be added to your Procfile:

```
web: bundle exec jekyll build && ruby -run -e httpd _site
```

### Push Your Jekyll Site to GitHub

For this article, let's use [Git commands](#) to [push your codes](#) to GitHub. First, create a repository on GitHub; this will give you access to the repository's URL.

You can now initialize a local Git repository by opening your terminal, navigating to the directory that contains your project, and running the following command:

```
git init
```

Now add your code to the local Git repository using the following command:

```
git add
```

You can now commit your changes using the following command:

```
git commit -m "my first commit"
```

**Note:** You can replace “my first commit” with a brief message describing your changes.

Finally, push your code to GitHub using the following commands:

```
git remote add origin [repository URL]  
git push -u origin master
```

**Note:** Ensure you replace “[repository URL]” with your own GitHub repository URL.

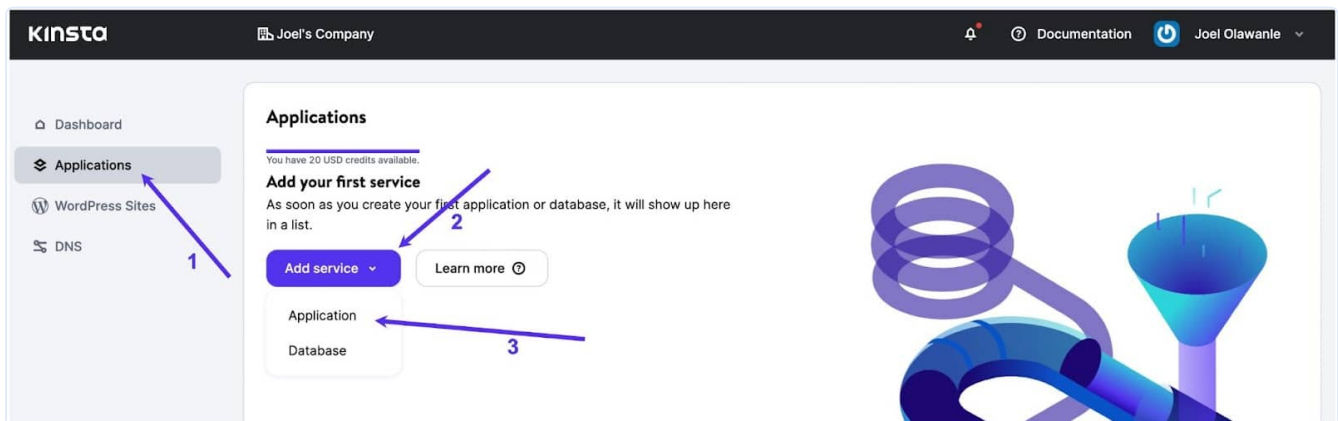
Once you have completed these steps, your code will be pushed to GitHub and accessible through your repository’s URL. You can now deploy to Kinsta!

## Deploying Your Jekyll Site To Kinsta

Deployment to Kinsta happens in just minutes. Start at the [My Kinsta](#) dashboard to log in or create your account. Next, [authorize Kinsta on GitHub](#).

You can then follow these steps to deploy your Jekyll site with [Kinsta Application Hosting](#):

1. Click **Applications** on the left sidebar
2. Click **Add service**
3. Click **Application** from the dropdown



— Deploying to Kinsta's application hosting

A modal will appear through which you can select the repository you wish to deploy. Select a branch you wish to deploy if you have multiple branches in your repository.

You can then assign a name to this application. Select a [data center location](#) among the 25 available, and then the Procfile will automatically supply the web process command.

The screenshot displays the Kinsta deployment dashboard. On the left is a sidebar with navigation links: Back, Deployments, Logs, Processes, Analytics, Settings, and Domains. The main content area is titled 'Deployment details' and includes a 'Change settings' button and a 'Redeploy' button. Below this, a table lists deployment information:

Deployed branch	Commit	By	Deploy started	Deploy time	Data center location
olawanlejoel/myblog  main	aec5d8	olawanlejoel	Mar 8, 2023, 11:18 AM	2m 5s	europa-west3

Below the table, the 'Commit message' is shown as 'add assets'. The 'Deployment progress' section shows three steps, all completed with green checkmarks:

- Build process completed (Mar 8, 2023, 11:18 AM)
- Rollout process completed (Mar 8, 2023, 11:20 AM)
- Deployment available at myblog-84b54.kinsta.app (Mar 8, 2023, 11:20 AM)

A 'View runtime logs' link is provided for the final step. A blue notification bubble is visible in the bottom right corner.

— Successful deployment of Jekyll static site

Your application will start deploying. Within a few minutes, a link will be provided to access the deployed version of your website.

Alternatively, you can [deploy your Jekyll site for free with Kinsta Static Site Hosting](#).

## Summary

So far, you have learned how Jekyll works and the various customizations that you can do with Jekyll. It's now safe to agree that Jekyll is an excellent tool for creating static websites due to its simplicity, flexibility, and powerful features.

Jekyll's intuitive templating system, liquid templates, and built-in support for markdown and other markup languages make it easy to create and manage content-rich sites quickly.

Feel free to host all your static websites with [Kinsta's Application Hosting for free](#), and if you like it, opt for our [Hobby Tier](#) plan, **starting at \$7/month**.

*What is your thought on Jekyll? Have you utilized Jekyll to build anything? Please feel free to share your projects and experiences with us in the comments section below.*