

Nicola Gigante

# Getting Started with Fleming on the Web



# Introduction

The purpose of this guide is to explain how the Fleming website works, and how to quickly deploy new content to the website. No experience in HTML, CSS or JavaScript is required to be able to add or remove content to the pages already set-up and functioning, but you may need to have some web development experience if you want to add new pages. By the end of this introduction, you'll be able to make changes to our website and deploy them to GitHub and Surge, either with the automatic Deploy script (only on macOS and Linux, sorry!) or by hand.

## Technologies Used by the Fleming Website

For those who would like to know how the website works in order to modify it, the following is a list of components used to build it.

- GatsbyJS
- React.JS
- Material Components for the Web
- Markdown

You will learn more about which technologies are used where as you go along with this guide. Again, if you are only trying to update the website, you only need to know some Markdown.

## Prerequisites

Whether you are trying to create new web pages, or update content on existing ones, you need to have Node.JS (npm), GatsbyJS and git installed in your machine. This can be accomplished in numerous ways, depending on your Operating System and the configuration you are running:

### macOS

You can download and install the Node.JS package via the official website [here](#). Alternatively, you may wish to install Node.JS using your preferred package manager, such as Homebrew or MacPorts. This can be achieved by running one of the appropriate commands listed [here](#). After installing Node.JS, open a Terminal and run the following command:

```
npm install -g gatsby-cli
```

You can now install git. This can be done via the official package available [here](#), or by using your preferred package manager (see [here](#) for more info).

If no error shows up, you're done! You can now start to modify the website and publish the changes.

### Windows

You can download and install the Node.JS package via the official website [here](#). Alternatively, you may wish to install Node.JS using your preferred package manager, such as Chocolatey or Scoop. This can be achieved by running one of the appropriate commands listed [here](#). After installing Node.JS, open a Command Prompt and run the following command:

```
npm install -g gatsby-cli
```

Git for Windows can be installed following the guide [here](#).

If no error shows up, you're done! You can now start to modify the website and publish the changes.

## Linux

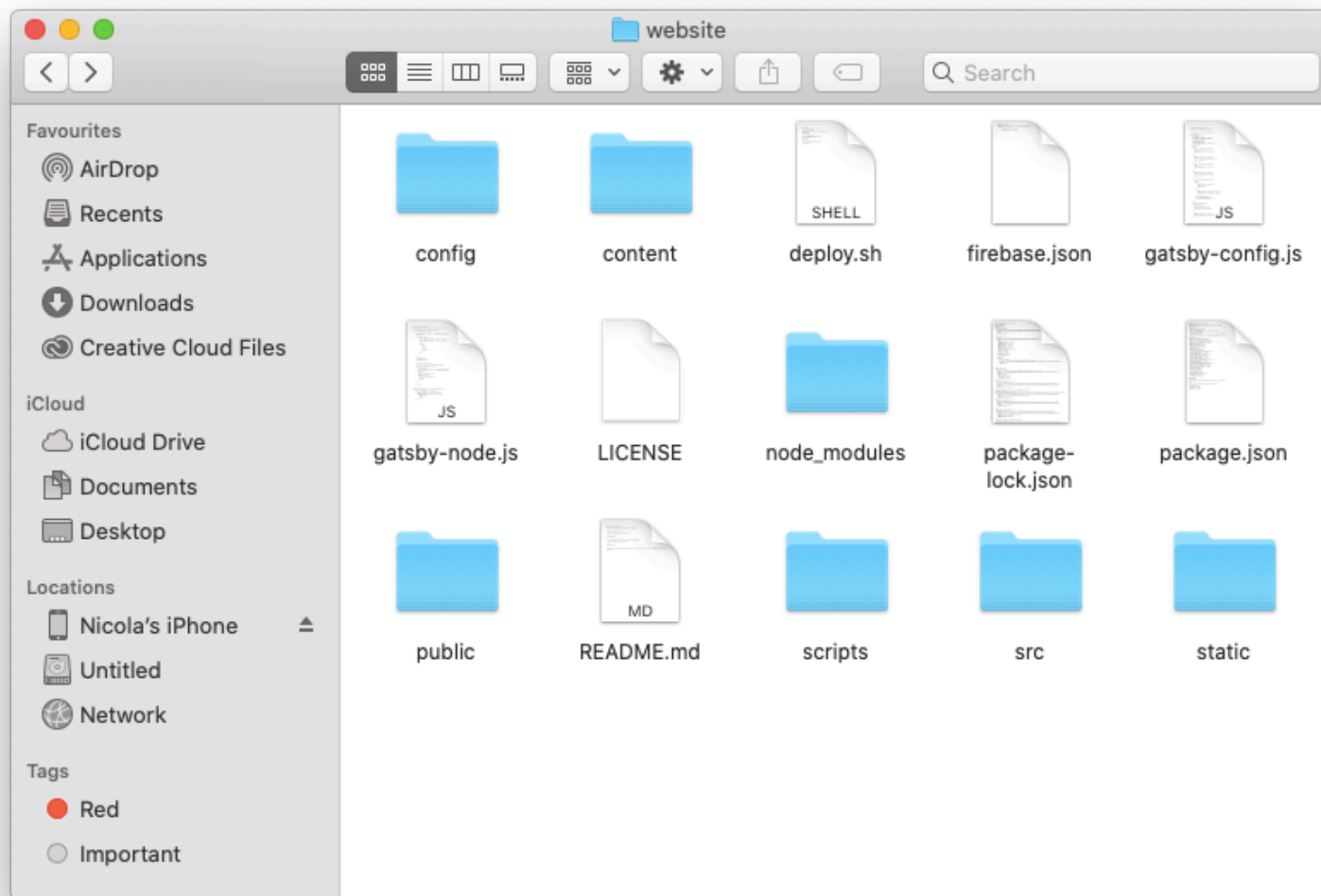
On Linux, you can either install Node.JS from source, or rely on the community maintained prebuilt packages. The latter option is a quicker way to get going, hence it is what we are going to do. The commands and repositories needed vary depending on which distribution of Linux you are running. Take a look at [this](#) page to know where to find the packages for your distro. Once Node.JS is installed, open a Terminal and run the following command:

```
npm install -g gatsby-cli
```

Installing Git on Linux couldn't be easier. Just use your distro's package management tool to install it. If you need help, see the relevant guide [here](#).

If no error shows up, you're done! You can now start to modify the website and publish the changes.

# The Folder Structure



Before making changes to the website, take a second to familiarise yourself on where things are located. Keep in mind that the folder structure is rigid, as the website will be compiled later before deployment, so make sure that every file you add or modify is placed in the right folder.



config

The config folder stores essential configuration files. These files dictate the website's name, as well as the URL and the share image.



content

The content folder stores the Markdown files used to generate content such as workshop tutorials and "Next Events" pages.



node\_modules

The node\_modules folder contains the packages used when compiling the website. It is an automatically generated folder and as such you shouldn't touch it.



public

The public folder stores the compiled website, ready for deployment. You shouldn't modify anything here by hand.



scripts

The scripts folder may or may not include failed scripts that I tried to make and I may or may not have forgotten to remove before making this. Nothing to see here, really.



src

The src folder contains the pages and static content of the website. These are written in JavaScript (using React.JS) and feature Material Design Components.



static

The static folder houses components that might not be appropriately placed in other folders, such as the robots.txt file and the share card image.

You should now have an idea of how the folder structure work and where to work if you want to change the layout of a webpage, add content to dynamic sections or creating a brand new section of the website.

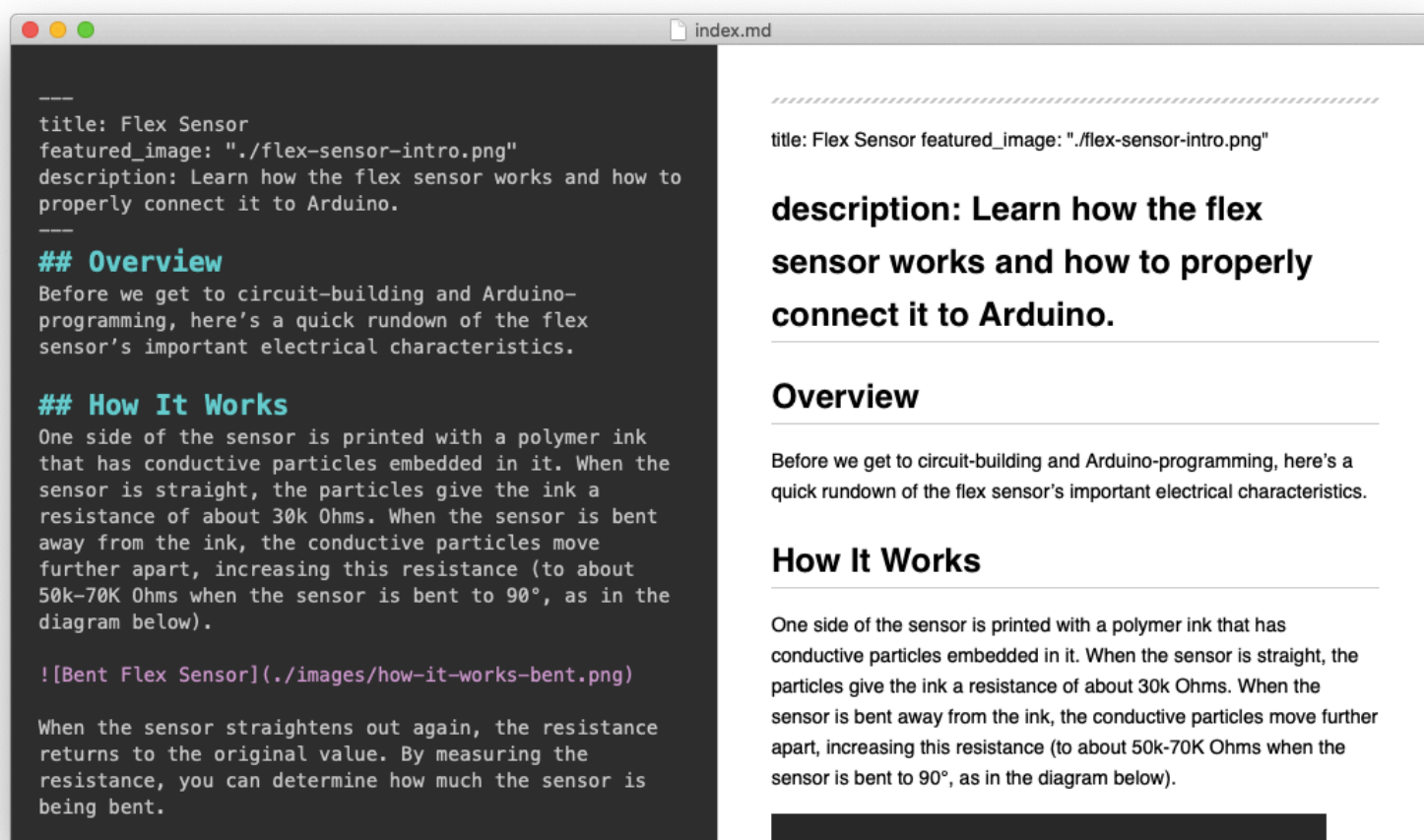
# Editing Markdown Content

Most content on the website is dynamically generated by Gatsby using Markdown files. This makes so that adding new content does not require any knowledge of HTML and CSS, even though you can add some basic HTML elements in your Markdown if you want to.

As stated previously, the Markdown content is located in the content folder. Inside that folder, you'll find a folder for each page that utilises Markdown content. **Please note that the homepage content is located in posts.**

Each Markdown page is contained in its own folder. Make sure that the folder has a clear name, as that will be the name of the page. Also note how every Markdown file is called "index.md". This is required.

Markdown can be edited with any text editor, but you may want to use a Markdown editor with a Render option in order to facilitate editing or creating Markdown files.



Note the header on top of existing Markdown files. This header dictates what will be shown on the preview cards on the website.

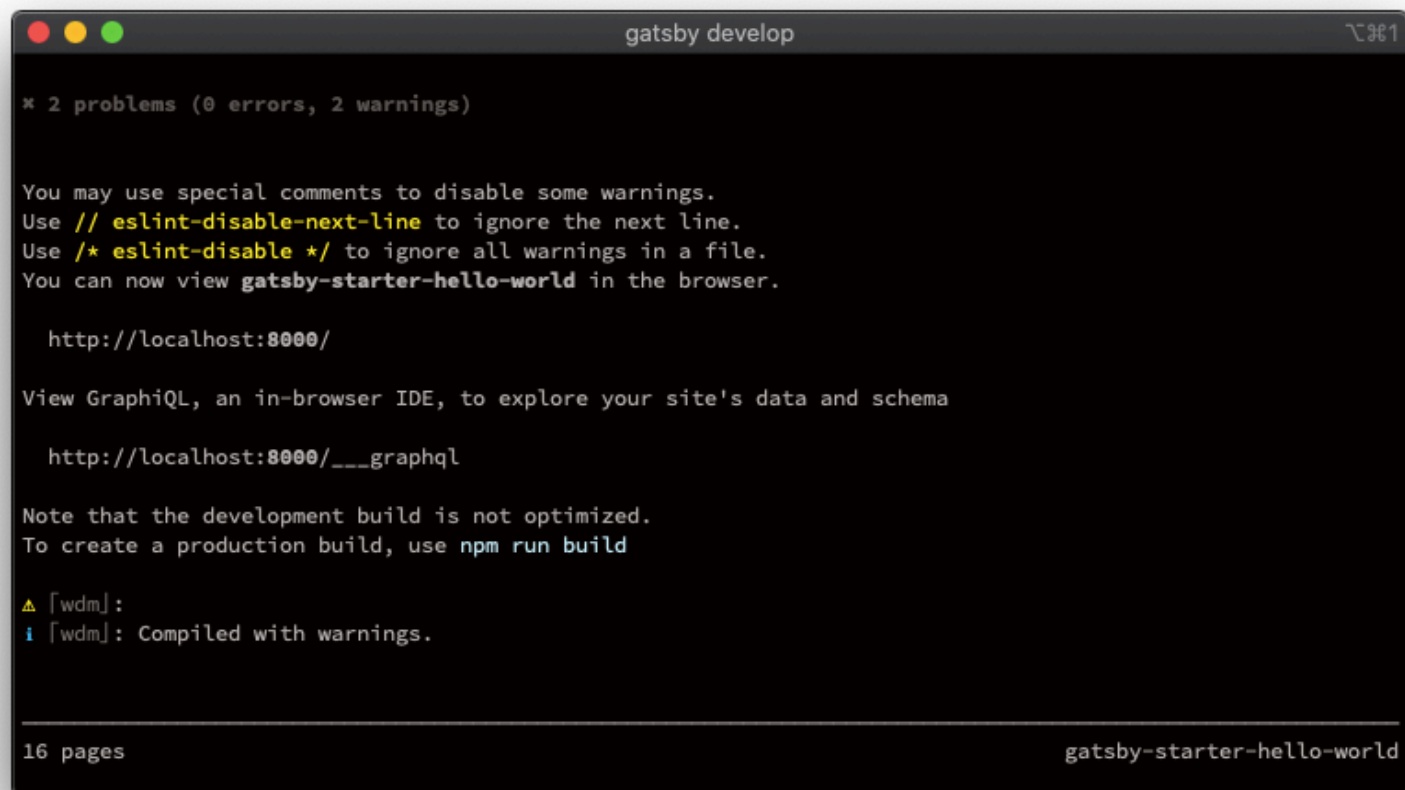
For more information on Markdown, visit [this website](#) or [click here](#) for a cheat sheet.

## Testing Your Changes

You may want to test your changes before publishing them. You can do so by executing the command

```
gatsby develop
```

in the main folder. This will compile the website in **debug** mode and create a local web server for you to test that everything is fine before deployment.

A terminal window titled "gatsby develop" with standard macOS window controls (red, yellow, green buttons) in the top-left corner. The terminal output is as follows:

```
* 2 problems (0 errors, 2 warnings)

You may use special comments to disable some warnings.
Use // eslint-disable-next-line to ignore the next line.
Use /* eslint-disable */ to ignore all warnings in a file.
You can now view gatsby-starter-hello-world in the browser.

  http://localhost:8000/

View GraphQL, an in-browser IDE, to explore your site's data and schema

  http://localhost:8000/___graphql

Note that the development build is not optimized.
To create a production build, use npm run build

▲ [wdm]:
i [wdm]: Compiled with warnings.
```

---

```
16 pages
```

gatsby-starter-hello-world



## Deploying Your Changes

After testing your changes, you may want to deploy them! This chapter assumes that you have gained access to our GitHub and our Surge, or you have cloned our GitHub and created your own Surge account in order to publish this website.

You can deploy your changes in a number of ways. If you're using macOS or Linux, you can use the provided shell script to automatically clean the working directory, compile the website, and push the changes to GitHub and Surge. Easy. If you're using Windows, you'll need to do these steps manually.

### Automatic Deploy

To deploy using the script, make sure that the script is runnable by executing

```
chmod a+x ./deploy.sh
```

in the main directory (you will only need to do this once). Then, execute the script

```
./deploy.sh
```

Note that you may be asked to login to GitHub and Surge if you haven't used them in the past on your PC. You will also need to input a Git commit message using Vim (or, in some rare cases, nano) and you'll be asked to enter the domain of the website.

You can find a cheat sheet for dealing with Vim [here](#). (I know, it is a pain to deal with. It gets better.)

## Manual Deploy

Before deploying your website, make sure that you clean your working directory and compile the latest version of your website by executing the commands

```
gatsby clean
```

```
gatsby build
```

Then, deploy your changes to GitHub

```
git add .
```

```
git commit -m "Your Message"
```

```
git push
```

Remember to replace Your Message with a brief description of the changes you made! As an added bonus, you won't have to deal with Vim. Lastly, deploy the website to Surge using the command

```
surge public/
```

As with the automatic deploy, keep in mind that you will be asked to login to GitHub and Surge if you have never done so on your machine.

## Conclusion

You should now be ready to work with the website. Some topics were not covered here (like editing JavaScript files) because they are out of the scope of this guide, but I'd be glad to help if you face any issues with that (remember that you need to know a bit of JS, HTML and CSS before venturing into that!). If you have any questions or suggestions, feel free to reach out, either via UCL's email (I might see it sooner there) or my personal email (visible on GitHub).

Thanks for reading!