Make a site/blog using Jekyll and GitHub

ROBERTO FERRER 23 SEPTEMBER 2013

0. Prerequisites

I LIKE INSTALLING PACKAGES USING THE OFFICIAL REPOSITORIES USING EITHER SYNAPTIC OR `apt-get install foo`, EVEN THOUGH THIS USUALLY MEANS NOT HAVING THE LATEST VERSION AVAILABLE. FOR THIS SITE, HOWEVER, I ENDED UP NOT DOING IT FOR CERTAIN PACKAGES BECAUSE IT PROVED MORE convenient.

- Git (installed using Synaptic)
- Jekyll-Bootstrap framework (not a package, but a set of files *cloned* from GitHub. More on this below)
- Rake (installed using Synaptic to easily execute some tasks related to blogging, but not necessary)

1. Setup a GitHub account

OPEN AN ACCOUNT AN ACCOUNT WITH GUTHUB. MAKE SURE TO ADJUST YOUR SECURITY SETTINGS SO YOU ARE ABLE TO COMMUNICATE WITH GITHUB FROM YOUR BASH TERMINAL. THIS MAY INVOLVE SETTING A SSH PASSPHRASE. CHECK https://help.github.com/categories/56/articles AND MORE GENERALLY https://help.github.com/ for GitHub support.

2. Create a GitHub repository for your site

CREATE A REPOSITORY NAMED **USER.github.io**, WHERE *USER* IS THE USER-NAME OF YOUR GITHUB ACCOUNT. WHEN CREATING THE REPOSITORY, DO NOT INITIATE IT WITH A README, .GITIGNORE, OR LICENSE file.

3. Download the general framework for your site and publish

These files provide the basics in order to easily start your blog/site.

OPEN A TERMINAL WINDOW AND GO TO THE LOCAL DIRECTORY THAT WILL CONTAIN YOURSITE DIRECTORY (suppose it is the home folder). Run

```
$ git clone https://github.com/plusjade/jekyll-bootstrap.git USER.github.io
  $ cd USER.github.io
  $ git remote set-url origin git@github.com:USER/USER.github.io.git
  $ git push origin master
```

THE FIRST LINE CLONES THE *Jekyll-Bootstrap* PROJECT INTO THE `~/USER.github.io` DIRECTORY. THE SECOND LINE MOVES INTO THAT DIRECTORY. THE THIRD LINE SETS A *remote* SO YOU CAN LINK THE LOCAL DIRECTORY TO THE REPOSITORY THAT WAS CREATED EARLIER IN THE GITHUB SITE. AND FINALLY, THE FOURTH LINE SENDS THE CONTENT OF YOUR LOCAL DIRECTORY TO GITHUB, AND THEREFORE, TO THE WWW. NOTICE THAT *origin* REFERS to the GitHub repository and *master* to the name of the Git branch in your local machine.

If all went well, you should be able to navigate to your new site using your favorite web browser and pointing to http://USER.github.io. (It can take a few minutes for your site to be served by GitHub).

So, what's going on? Save yourself some trouble

Basically, when you downloaded *Jekyll-Bootstrap* you downloaded a suite of *markdown* and *html* files arranged in such a way that when pushed to your GitHub repository, GitHub parses them with a *Markdown* interpreter and Jekyll, and then does the publishing of the resulting HTML files. This is the reason there's a lag between the moment you `git push` and when the site is available on the WWW.

At times, GitHub can find an error in one of your Markdown/HTML files and won't be able to publish your latest modifications. If this is the case, you will receive an annoying email from GitHub indicating this was the case.

You can avoid this by doing a local build of your site directory. This just means that you parse the files yourself in your machine and then preview your site locally using a web browser, before sending the commit over to GitHub. The local parsing of source files will indicate if there is any error that must be wiped before pushing to GitHub. This preview is only available to you, and as such, also gives you some liberty over the feel and look of your site before displaying it to the rest of the world.

To be able to do this, you need to install the **Jekyll** package. One alternative is to use whatever version of Jekyll is available in oficial repositories. I tried this and it worked for a while; until GitHub started sending emails pointing to an error even though local builds of the site were running just fine. This situation did not allow me to debug the source files as I was unable to detect what GitHub's parsing was complaining about.

Probably the best route to follow is installing the Ruby gem 'github-pages' (if installing this gem, there's no need to install Jekyll separately). This is a good idea because the gem provides the packages used by GitHub to parse the files you send in. A successfull local build is likely to produce a successfull GitHub build. To do this, you should install previously:

- Ruby (installed using Synaptic, both ruby1.9.1 and ruby1.9.1-dev)
- Bundle (installed using Synaptic)

Detailed instructions for the rest of the installation are at https://help.github.com/articles/using-jekyll-with-pages. Afterwards, to build the site locally, within the root of the site directory run `bundle exec jekyll serve`.

Now you can browse to http://localhost:4000 and see the site. Modify it as you find appropriate and check the changes locally. When you're happy with them, run `git push -u origin master` to deploy to the WWW.

Remember to keep Jekyll (i.e. the 'github-pages' gem) up-to-date running `bundle update` in your local site directory, every now and then. This ensures that the local build is consistent with the GitHub remote build.

Make your first changes to the site

As you have probably noticed already, the *Jekyll-Bootstrap* project comes along with some sample pages and posts for your new site. The information contained therein has basic instructions on how do modify the site to suit your needs.

The first thing you should do is modify the `~/USER.github.io/_config.yml` file to reflect your information. At this point, run `bundle exec jekyll serve` and check your site locally to visually acknowledge the changes you just made.

Next, you can create new posts and pages using `rake post` and `rake page`, respectively. Navigate to the root directory of your site:

```
$ cd ~/USER.github.io
```

To create a new post titled *foo*, run

```
$ rake post title="foo"
```

To create a new page titled *foo*, run

```
$ rake page name="foo.md"
```

Both, posts and pages are created as markdown files which can be modified with your favorite text editor. Posts are created in the `~/USER.github.io/_posts` directory. Pages can be created almost anywhere. I have defined other rake tasks using code from https://github.com/gummesson/jekyll-rake-boilerplate and adding it to the original rakefile provided by the Jekyll-Bootstrap project. You can check my rakefile at https://github.com/refp16/

Troubleshooting

After pushing your last changes to GitHub the website may fail to load. GitHub now sends an email specifying the conflicting file. The name of the file can also be located in the *Settings* page of the website repository in GitHub. For example, settings page is https://github.com/refp16/refp16.github.io/settings.

An incomplete description of my site directory (refp16.github.io)

_site: is a directory created by the local Jekyll installation and it contains the local build of the site which can be previewed at localhost:4000. GitHub makes no use of this directory and it is set up to be ignored by Git (through the *.gitignore* file).

_config.yml: this file configures Jekyll settings. The GitHub site will work without it, but for example, the blog setup will not be executed.

Gemfile: this file is used by **Bundle** to install the 'github-pages' gem. This ensures that Jekyll local build is consistent with GitHub's remote build. After creating this file, run in the site directory `bundle install` (after installing **Bundler** - see here) to start downloading the 'github-pages' gem. Bundler uses the Gemfile to figure out what it should install.

Rakefile: this is a convenience file that creates tasks based on other commmands/instructions. It allows, for example, to quickly create posts and deploy to the WWW.

{404.html, archive.html, atom.html, categories.html, pages.html, rss.xml, sitemap.txt, tags.html,} are all files used to create corresponding pages within the site.

Add the possibility of rendering mathematics

If you're interested in using mathematics in your site, you can do so using MathJax. I haven't found the ideal solution to this problem but what I have will do for the moment.

Add the following JavaScript snippet to the `<head>` block of the HTML template file `~/USER.github.io/_includes/themes/twitter/default.html`:

```
<script type="text/javascript" src="http://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS-MML_HTMLorMML"></script>
```

With this minor change I have managed the following:

- One way of doing inline math: `\\( 1/x^2 \\)`, rendering $1/x^{2}$.
- Three ways of doing block math.

The first one uses

```
<div>$$ f = ma. $$</div>
```

The second one uses the LaTeX `equation` environemnt on its own:

```
\begin{equation}
   \frac{1}{x^3}
\end{equation}
```

and the third one uses surrounding `<div>` tags with the `equation` environment:

```
<div>
\begin{equation}
\frac{1}{x^4}.
\end{equation}
</div>
```

These three pieces of code render, respectively:

$$f = ma,$$

$$\begin{equation} \frac{1}{x^3} \end{equation}$$

and

$$\begin{equation} \frac{1}{x^4}. \end{equation}$$

The `<div>` tags are meant to protect the latex syntax from being messed up by the markdown parser. Interestingly, when using the `equation` environment, there seems to be no need for tags. I have no explanation for this (and many other things related to the rendering of mathematics).

More details can be found at:

http://jekyllbootstrap.com/

http://jekyllbootstrap.com/usage/jekyll-quick-start.html

http://jekyllrb.com

http://daringfireball.net/projects/markdown/

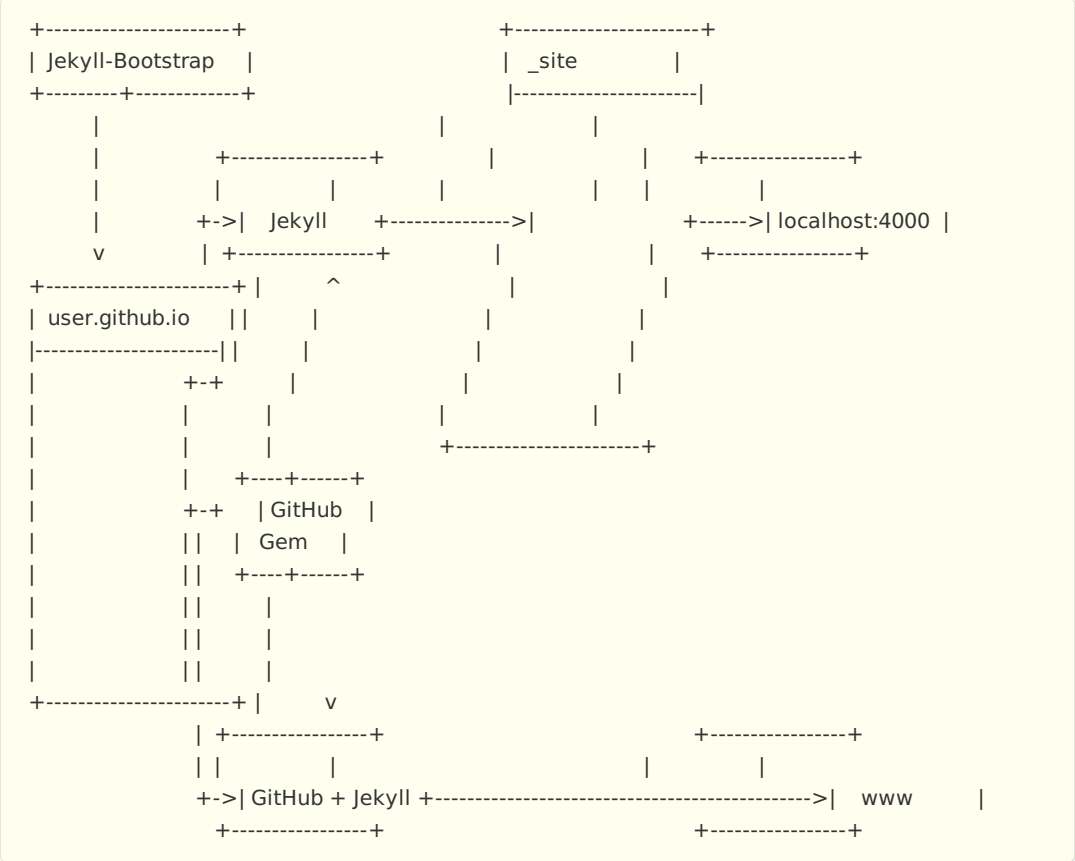
http://git-scm.com/

Credits

http://truongtx.me/2013/05/08/blogging-using-your-favorite-text-editor-with-git-and-jekyll/

http://stackoverflow.com/questions/10987992/using-mathjax-with-jekyll

Carl Boettiger (personal correspondance)



Updated: 15Mar2014

jekyll [2] jekyll-bootstrap [1] git [1] github [2] mathjax [1] blog [2] site [2]

← Earlier Post Next Post →