Getting started with Octopress

Peter Humburg

Published at *Genomic Campfire* on Jan 25 2015

Contents

Recently I decided that it was time for me to start my own blog. When thinking about the details I quickly realised that I wanted something that integrated well with my existing workflow. The process of writing, publishing and managing posts needed to be as unobtrusive as possible, otherwise I'd struggle to find the time. Since most of my code is hosted on GitHub it seemed attractive to host my blog there as well, using GitHub Pages. This allows me to manage the blog through a git repository, a process I'm familiar with. This means using static HTML pages but I don't expect to need anything else. There is no content management system, no database to fiddle with. Posts stored in a git repository, with all the version management that implies.

GitHub Pages have direct support for Jekyll, suggesting that it should be easy to set things up and just get started. This means using Markdown to author posts. Since I use Markdown for my work anyway, this fits well with my general workflow. If it requires using the command-line to get a post published that certainly isn't a problem and when using Jekyll GitHub will even take care of the Markdown to HTML conversion automatically.

Eventually, I decided to use Octopress rather than pure Jekyll. This means compiling posts locally before bushing the published version to GitHub but that is easy enough and it seemed like I would get more of the bells and whistles pre-configured. Below I describe my experience in getting this up and running. I'm hardly the first to do this and there are many sites offering instructions and advice. In the end I had to consult several of those sites to make everything work the way I want it to. While most of this isn't very original it might still be useful to have it all in one place. I'm assuming that you are familiar with the command-line and git but don't know much about Octopress.

Installing the basics

Since Octopress produces static pages that are then pushed to a GitHub repository[1] it is necessary to install Octopress locally to generate the content. I'm using Ubuntu (14.10) and all installation instructions are based on that. If you use a different flavour of Linux the process will be similar, although possibly using a different package manager[2]. I expect it to be largely the same on Macs as well (but I haven't tried). Don't ask me about installing Octopress on Windows[3].

Octopress is written in Ruby, so that is the first thing we'll need[4]. I already had the *ruby* package installed but you can easily do that with

```
sudo apt-get ruby
```

if necessary. As I found out the hard way the separate *ruby-dev* package is also required[5].

```
sudo apt-get ruby-dev
```

You may also want the *rubygems-integration* package.

```
sudo apt-get rubygems-integration
```

This makes ruby gems and corresponding Ubuntu packages interchangeable. So you don't have to install a gem if you already installed the package earlier.

You'll also need *git* if you don't have it already.

```
sudo apt-get install git
```

From here I followed the instructions from the Octopress documentation. The first step is to get Octopress by cloning the GitHub repository.

```
git clone git://github.com/imathis/octopress.git octopress
cd octopress
```

[1] or elsewhere, but I'll just be talking about GitHub deployment
[2] and if you prefer to compile from source you probably know what you are doing anyway
[3] but zerosharp shoud have you covered
[4] This may be a good time to confess that I haven't written a line of Ruby code in my life
[5] Separating binaries and header files like this is something Ubuntu insists on doing that frequently annoys me.

Of course you can choose a different name for your working directory. The cloned repo will still point to the octopress repository as a remote, but we'll take care of that in a minute. The only thing left to do is to install the default theme.

```
rake install
```

Deploying to GitHub

The deployment process is pretty well explained in the documentation so I'll only briefly summarise the steps required to deploy to a GitHub user page (as opposed to a project page). This means that the blog will be available at `http://username.github.io`.

The first thing that is required is a GitHub repository called *username.github.io*. Just head to the GitHub website and create a new repository with that name, replacing *username* with your GitHub user name, of course. It is important that the repository name is spelled correctly, so double check it. You can make the repository public or private but *don't* initialize it, we'll be importing the Octopress repository.

Now that the repository exists Octopress can do all the hard work for us by executing a rake task.

```
rake setup_github_pages
```

This will ask for the address of the repository we just created and then proceed to configure the local octopress installation such that this repo becomes the default *origin* remote (the octopress repository is still available as remote *octopress*). It also creates a *master* branch that is used for deployment and switches the active branch to *source*. The source branch is where all changes to the blog are made[6], i.e. it is your development branch for the blog. The master branch holds the static pages generated from the source and represents the published content. Although you can push to that branch directly, you shouldn't have to. There are rake tasks to take care of deployment. You **do** have to commit and push the source branch however.

```
git add .
git commit -m "Initial configuration"
git push origin source
```

Now everything is ready to deploy the blog but you may want to customise it a bit before going public. That doesn't mean you can't marvel at the fruits of your labours though. Running

```
rake preview
```

will serve your blog locally. It will also monitor your Octopress directory for changes and rebuild files as necessary. You'll still have to reload the page in the browser though. Direct your browser to *localhost:4000* to admire the blog you just created. That's nice, but somewhat bare and generic. Don't worry, we'll address that in the next steps.

Installing a theme

I didn't really want to keep the default theme so I went out to find something else. Luckily there are a ton of Octopress themes outhere. The Octopress GitHub wiki has a list of themes that is a pretty good starting point. Even better, a gallery of these themes is also available. If none of them are to your liking, you favourite search engine should have no trouble finding more.

Many of the themes can be installed by cloning them into your Octopress installation. I chose the foxslide theme and did just that.

```
git submodule add https://github.com/sevenadrian/foxslide .themes/foxslide
git submodule update --init
rake install['foxslide']
```

When running the rake install task Octopress may warn you about a theme already being present. Replace the existing theme with the new one and rebuild the blog. Check the preview again to see the changes.

[6] including the authoring of new posts

Customising your blog

Now it is time to add your own touch. Some of what follows generally applies to Octopress but other parts are theme specific. I'll try to make this clear but if you encounter something that doesn't seem to make sense, it may be because you are using a different theme and things are set-up differently.

The first thing to look at is the configuration file `_config.yml`. This has several useful options to tweak your blog. Octopress should have populated the *url* field based on the name of your GitHub repo. You'll want to fill in your name and the title and description of your blog. The latter will be used to generate a *description* meta tag in the page header. If you would like RSS feeds for each category[7], add

```
category_feeds: true
```

in the RSS section. The following options are probably fine the way they are for now[8].

The next section of interest is the one containing third party settings towards the end of the file. Here you can tell Octopress about your accounts on various social networks as well as activate sharing services. For each (supported) social media platform that you want to link to you'll have to provide a username, or at least that's what Octopress calls it. What you really need to provide is the information necessary to complete the url that points to your profile page. This may not necessarily be what you would consider to be your username. For Google+ for example, this may be a long string of numbers. If in doubt you can figure this out by entering your best guess[9] and then compare the generated link (in the preview) to the actual url of your profile.

Here you can also activate *Disqus* integration. This will enable comments for your posts[10]. If you don't have a Disqus account you'll have to sign up first. You'll also need to add your blog to your Disqus account. In the process of doing this you'll choose a Disqus url. The first part of this will be the *shortname* you need to add to the config file. There is no need to worry about the integration instructions Disqus provides, Octopress will take care of that.

If you have chosen a third party theme there may be support for additional social media platforms. The *foxslide* theme comes with build-in support for a lot of platforms so I could easily add links to my *LinkedIn* and *Quora* profiles simply by adding `linkedin_user` and `quora_user` entries to `_config.yml`. I only realised these were already build-in when I looked at `source/_includes/custom/asides/found_on.html` to figure out how to add them.

That takes care of basic customisation. I also wanted a custom banner at the top. This is easily achieved by replacing the `source/images/heroBack.jpg` that comes with *foxslide* with an alternative image[11].

The first post

Okay, now everything should be in place to get started blogging. A simple `rake new_post["your fancy title"]` will create a file with the appropriate meta data. The file should look something like this:

```
---
layout: post
title: "your fancy title"
date: 2015-01-14 16:24:54 +0000
comments: true
categories:
---
```

Just fill in the empty fields and start writing[12]. Don't forget to check the preview. Once you are satisfied change the date if necessary. Then run

[7] you'll be creating categories by tagging your posts with category names
[8] and of course there is documentation for all of this, in case you want to play around with some of them
[9] or nonsense if you prefer
[10] comments can be disabled for individual post so this isn't all or nothing
[11] just make sure it's big enough, otherwise it'll look awkward
[12] If you want to provide more than one category for your post simply provide a space separated list

```
rake generate
rake deploy
```

Or, simply `rake gen_deploy` to do it all in one step.

It took me a bit longer than I had hoped to set this all up, but in the end it wasn't so bad and Octopress did take care of some of the potentially fiddly bits with. minimum fuss. At this point you basically should have a functioning blog. In a future post I'll talk about some of the other tweaks I made to the theme and Octopress to customise it for my needs.