# Building a data science portfolio: Making a data science blog

Vik Paruchuri
14 JUN 2016 in tutorials, python, matplotlib, blog, data, pandas, and portfolio

*This is the second in a series of posts on how to build a Data Science Portfolio. You can find links to the other posts in this series at the bottom of the post.*

Blogging can be a fantastic way to demonstrate your skills, learn topics in more depth, and build an audience. There are quite a few examples of data science and programming blogs that have helped their authors land jobs or make important connections. Blogging is one of the most important things that any aspiring programmer or data scientist should be doing on a regular basis.

Unfortunately, one very arbitrary barrier to blogging can be knowing how to setup a blog in the first place. In this post, we'll cover how to create a blog using Python, how to create posts using Jupyter notebook, and how to deploy the blog live using Github Pages. After reading this post, you'll be able to create your own data science blog, and author posts in a familiar and simple interface.

## Static Sites

Fundamentally, a static site is just a folder full of HTML files. We can run a server that allows others to connect to this folder and retrieve files. The nice thing about this is that it doesn't require a database or any other moving parts, and it's very easy to host on sites like Github. It's a great idea to have your blog be a static site, because it makes maintaining it very simple. One way to create a static site is to manually edit HTML, then upload the folder full of HTML to a server. In this scenario, you would at the minimum need an `index.html` file. If your website URL was `thebestblog.com`, and visitors visited `http://www.thebestblog.com`, they would be shown the contents of `index.html`. Here's how a folder of HTML might look for `thebestblog.com`:

```
thebestblog.com
│   index.html
│   first-post.html
│   how-to-use-python.html
│   how-to-do-machine-learning.html
│   styles.css
```

On the above site, visiting `http://www.thebestblog.com/first-post.html` would show you the content in `first-post.html`, and so on. `first-post.html` might look like this:

```html
<html>

<head>

  <title>The best blog!</title>

  <meta name="description" content="The best blog!"/>

  <link rel="stylesheet" href="styles.css" />

</head>

<body>

  <h1>First post!</h1>

  <p>This is the first post in what will soon become (if it already isn't)

  <p>Future posts will teach you about data science.</p>


<div class="footer">

  <p>Thanks for visiting!</p>

</div>

</body>

</html>
```

You might immediately notice a few problems with manually editing HTML:

- Manually editing HTML is incredibly painful.
- If you want to make multiple posts, you have to copy over styles, and other elements, like the title and footer.
- If you want to integrate comments or other plugins, you'll have to write javascript.

Generally, when you're blogging, you want to focus on the content, not spend time fighting with HTML. Thankfully, you can create a blog without hand editing HTML using tools known as static site generators.

## Static Site Generators

Static site generators allow you to write blog posts in simple formats, usually markdown, then define some settings. The generators then convert your posts into HTML automatically. Using a static site generator, we'd be able to dramatically simplify `first-post.html` into `first-post.md`:

```
# First post!


This is the first post in what will soon become (if it already isn't) the b


Future posts will teach you about data science.
```

This is much easier to manage than the HTML file! Common elements, like the title and the footer, can be placed into templates, so they can be easily changed.

There are a few different static site generators. The most popular is called Jekyll, and is written in Ruby. Since we'll be making a data science blog, we want a static site generator that can process Jupyter notebooks.

Pelican is a static site generator that is written in Python that can take in Jupyter notebook files and convert them to HTML blog posts. Pelican also makes it easy to deploy our blog to Github Pages, where other people can read our blog.

## Installing Pelican

Before we get started, here's a repo that's an example of what we'll eventually get to.

If you don't have Python installed, you'll need to do some preliminary setup before we get started. Here are setup instructions for Python. We recommend using `Python 3.5`. Once you have Python installed:

- Create a folder – we'll put our blog content and styles in this folder. We'll refer to it in this tutorial as `jupyter-blog`, but you can call it whatever you want.
- `cd` into `jupyter-blog`.
- Create a file called `.gitignore`, and add in the content from this file. We'll need to eventually commit our repo to git, and this will exclude some files when we do.
- Create and activate a virtual environment.
- Create a file called `requirements.txt` in `jupyter-blog`, with the following content:

```
Markdown==2.6.6

pelican==3.6.3

jupyter>=1.0

ipython>=4.0

nbconvert>=4.0

beautifulsoup4
```

```
ghp-import==0.4.1

matplotlib==1.5.1
```

- Run `pip install -r requirements.txt` in `jupyter-blog` to install all of the packages in `requirements.txt`.

## Creating your data science blog

Once you've done the preliminary setup, you're ready to create your blog! Run `pelican-quickstart` in `jupyter-blog` to start an interactive setup sequence for your blog. You'll get a sequence of questions that will help you setup your blog properly. For most of the questions, it's okay to just hit `Enter` and accept the default value. The only ones you should fill out are the title of the website, the author of the website, `n` for the URL prefix, and the timezone. Here's an example:

```
(jupyter-blog)➜  jupyter-blog ✗ pelican-quickstart

Welcome to pelican-quickstart v3.6.3.


This script will help you create a new Pelican-based website.


Please answer the following questions so this script can generate the files

needed by Pelican.



> Where do you want to create your new web site? [.]

> What will be the title of this web site? Vik's Blog

> Who will be the author of this web site? Vik Paruchuri
> What will be the default language of this web site? [en]

> Do you want to specify a URL prefix? e.g., http://example.com   (Y/n) n

> Do you want to enable article pagination? (Y/n)

> How many articles per page do you want? [10]

> What is your time zone? [Europe/Paris] America/Los_Angeles

> Do you want to generate a Fabfile/Makefile to automate generation and pub

> Do you want an auto-reload & simpleHTTP script to assist with theme and s

> Do you want to upload your website using FTP? (y/N)

> Do you want to upload your website using SSH? (y/N)

> Do you want to upload your website using Dropbox? (y/N)

> Do you want to upload your website using S3? (y/N)

> Do you want to upload your website using Rackspace Cloud Files? (y/N)

> Do you want to upload your website using GitHub Pages? (y/N)
```

After running `pelican-quickstart`, you should have two new folders in `jupyter-blog`, `content`, and `output`, along with several files, such as `pelicanconf.py` and `publishconf.py`. Here's an example of what should be in the folder:

```
jupyter-blog

│   output

│   content

│   .gitignore

│   develop_server.sh

│   fabfile.py

│   Makefile

│   requirements.txt

│   pelicanconf.py
```

```
|   publishconf.py
```

## Installing the Jupyter plugin

Pelican doesn't support writing blog posts using Jupyter by default – we'll need to install a plugin that enables this behavior. We'll install the plugin as a git submodule to make it easier to manage. If you don't have git installed, you can find instructions here. Once you have git installed:

- Run `git init` to initialize the current folder as a git repository.
- Create the folder `plugins`.
- Run `git submodule add git://github.com/danielfrg/pelican-ipynb.git plugins/ipynb` to add in the plugin.

You should now have a `.gitmodules` file and a `plugins` folder:

```
jupyter-blog
|   output
|   content
|   plugins
|   .gitignore
|   .gitmodules
|   develop_server.sh
|   fabfile.py
|   Makefile
|   requirements.txt
|   pelicanconf.py
|   publishconf.py
```

In order to activate the plugin, we'll need to modify `pelicanconf.py` and add these lines at the bottom:

```
MARKUP = ('md', 'ipynb')


PLUGIN_PATH = './plugins'

PLUGINS = ['ipynb.markup']
```

These lines tell Pelican to activate the plugin when generating HTML.

## Writing your first post

Once the plugin is installed, we can create a first post:

- Create a Jupyter notebook with some basic content. Here's an example you can download if you want.
- Copy the notebook file into the `content` folder.
- Create a file that has the same name as your notebook, but with the extension `.ipynb-meta`. Here's an example.

- Add the following content to the `ipynb-meta` file, but change the fields to match your own post:

```
Title: First Post

Slug: first-post

Date: 2016-06-08 20:00

Category: posts

Tags: python firsts

author: Vik Paruchuri

Summary: My first post, read it to find out.
```

Here’s an explanation of the fields:

- `Title` – the title of the post.
- `Slug` – the path at which the post will be accessed on the server. If the slug is `first-post`, and your server is `jupyter-blog.com`, you’d access the post at `http://www.jupyter-blog.com/first-post`.
- `Date` – the date the post will be published.
- `Category` – a category for the post – this can be anything.
- `Tags` – a space-separated list of tags to use for the post. These can be anything.
- `Author` – the name of the author of the post.
- `Summary` – a short summary of your post.

You’ll need to copy in a notebook file, and create an `ipynb-meta` file whenever you want to add a new post to your blog.

Once you've created the notebook and the meta file, you're ready to generate your blog HTML files. Here's an example of what the `jupyter-blog` folder should look like now:

```
jupyter-blog
│   output
│   content
    │   first-post.ipynb
    │   first-post.ipynb-meta
│   plugins
│   .gitignore
│   .gitmodules
│   develop_server.sh
│   fabfile.py
│   Makefile
│   requirements.txt
│   pelicanconf.py
│   publishconf.py
```

## Generating HTML

In order to generate HTML from our post, we'll need to run Pelican to convert the notebooks to HTML, then run a local server to be able to view them:

- Switch to the `jupyter-blog` folder.
- Run `pelican content` to generate the HTML.
- Switch to the `output` directory.

- Run `python -m pelican.server`.
- Visit `localhost:8000` in your browser to preview the blog.

You should be able to browse a listing of all the posts in your blog, along with the specific post you created.

## Creating a Github Page

Github Pages is a feature of Github that allows you to quickly deploy a static site and let anyone access it using a unique URL. In order to set it up, you'll need to:

- Sign up for Github if you haven't already.
- Create a repository called `username.github.io`, where `username` is your Github username. Here's a more detailed guide on how to do this.
- Switch to the `jupyter-blog` folder.
- Add the repository as a remote for your local git repository by running `git remote add origin git@github.com:username/username.github.io.git` – replace both references to `username` with your Github username.

A Github page will display whatever HTML files are pushed up to the `master` branch of the repository `username.github.io` at the URL `username.github.io` (the repository name and the URL are the same).

First, we'll need to modify Pelican so that URLs point to the right spot:

- Edit `SITEURL` in `publishconf.py`, so that it is set to `http://username.github.io`, where `username` is your Github username.

- Run `pelican content -s publishconf.py`. When you want to preview your blog locally, run `pelican content`. Before you deploy, run `pelican content -s publishconf.py`. This uses the correct settings file for deployment.

## Committing your files

If you want to store your actual notebooks and other files in the same Git repo as a Github Page, you can use git branches.

- Run `git checkout -b dev` to create and switch to a branch called `dev`. We can't use `master` to store our notebooks, since that's the branch that's used by Github Pages.
- Create a commit and push to Github like normal (using `git add`, `git commit`, and `git push`).

## Deploy to Github Pages

We'll need to add the content of the blog to the `master` branch for Github Pages to work properly. Currently, the HTML content is inside the folder `output`, but we need it to be at the root of the repository, not in a subfolder. We can use the ghp-import tool for this:

- Run `ghp-import output -b master` to import everything in the `output` folder to the `master` branch.
- Use `git push origin master` to push your content to Github.
- Try visiting `username.github.io` – you should see your page!

Whenever you make a change to your blog, just re-run the `pelican content -s publishconf.py`, `ghp-import` and `git push` commands above, and your Github Page will be updated.

## Next steps

We've come a long way! You now should be able to author blog posts and push them to Github Pages. Anyone should be able to access your blog at `username.github.io` (replacing `username` with your Github username). This gives you a great way to show off your data science portfolio.

As you write more posts and gain an audience, you may want to dive more into a few areas:

- Theming
  - Pelican supports themes. You can see dozens of themes here, and use the one you like most.
- Your own custom URL.
  - Using `username.github.io` is nice, but sometimes you want a more custom domain. Here's a guide on using a custom domain with Github Pages.
- Plugins
  - Check out the list of plugins here. Plugins can help you setup analytics, commenting, and more.
- Promotion
  - Trying promoting your blog posts on sites like DataTau, Twitter, Quora, and others to build an audience.

At Dataquest, our interactive guided projects are designed to help you start building a data science portfolio to demonstrate your skills to employers and get a job in data. If you're interested, you can signup and do our first module for free.

---

*If you liked this, you might like to read the other posts in our 'Build a Data Science Portfolio' series:*

- *Storytelling with data.*
- *Building a machine learning project.*
- *The key to building a data science portfolio that will get you a job.*
- *17 places to find datasets for data science projects*
- *How to present your data science portfolio on Github*

**Vik Paruchuri**

Developer and Data Scientist in San Francisco; Founder of Dataquest.io (Learn Data Science in your Browser).

**Share this post**

Get in touch @vikparuchuri.