

A stylized, light-colored globe is centered in the background, showing the continents of North and South America. The globe is set against a solid orange background.

BUILD & DEPLOY



A WEBSITE USING PYTHON

A STEP-BY-STEP GUIDE

VIMALKUMAR VELAYUDHAN

Contents

About this course	1
Requirements	1
Section 1: Install required software	3
Install or verify Python installation	3
Setup a virtual environment	4
Install virtualenv and virtualenvwrapper	4
Create a virtual environment	6
Using the virtual environment	6
Install Pelican, the static site generator	8
Add support for Markdown	8
Git	8
Installation	9
ghp-import for Github deployment support	9
Section 2: Setup website	11
Create the site	11
Basic configuration	11
URL prefix, pagination and scripts	12
Hosting options	13
Build the site	14
Preview the site	16
Write content	17
Create new page	18
Create new blog post	19

Section 3: Host website	21
Hosting on GitHub	21
Setup a Git repository	21
Github configuration	22
Create repository on Github	22
Adding a remote	23
Publish site	24
Updating the site	26
Section 4: Links	29
Links	29

About this course

About this course

This is a step-by-step guide on creating and hosting a website and a blog using programs written in Python.

We will be using Pelican - a static site generator for building the website.

These are some advantages of using a static site generator

- There is no need to install and maintain software (ex. a content management system).
- There is no requirement for a database.
- Content is written in plain text using the Markdown or reStructuredText syntax. There is no need to write HTML directly.

Requirements

The basic requirement for this course is an installation of Ubuntu Linux or Windows.

These steps have been tested on *Ubuntu 14.04 LTS* and *Windows 7*.

Section 1: Install required software

Install or verify Python installation

Linux

Python is installed on Ubuntu and most other Linux distributions by default. However, **Python 2** is required to follow examples in this book.

You can check the version of Python installed using the following command in a terminal:

```
python -V
```

Windows

On Windows, you will need to download and install the latest version of **Python 2** (currently 2.7.8) from <http://www.python.org>.

Download the MSI installer - `python-2.7.8.msi` (as of this writing) and perform an installation using the default options.

Update system PATH

Add `C:\Python27` and `C:\Python27\Scripts` to Windows PATH. This is necessary so that we can call the `python` interpreter and scripts provided by the various Python packages in a `cmd.exe` session.

To do so,

1. Right click on "Computer", select "Properties"
2. Select "Advanced system settings" from the navigation pane on the left.
3. Click on the "Environment Variables" button

Setup a virtual environment

4. Under "User variables for user", add (or edit) a variable called `PATH` and append the following:

```
C:\Python27;C:\Python27\Scripts
```

5. Click the "OK" button to close the dialogs.
6. Open the command prompt (`cmd.exe`) and type:

```
python -V
```

Python should now be installed and available in the system `PATH`.

Setup a virtual environment

The second requirement is to create an isolated Python setup for our website. This is called as a *virtual environment*.

We can then install additional python packages in the virtual environment without affecting other programs installed on the system.

Mutiple such virtual environments can be created each independent of the other.

The **virtualenv** Python module can be used to setup virtual environments. **virtualenvwrapper** is an extension to **virtualenv** that makes it easier to create and manage virtual environments.

Install virtualenv and virtualenvwrapper

 Linux

Install the *virtualenv* and *virtualenvwrapper* packages using the following command:

```
sudo apt-get install python-virtualenv virtualenvwrapper
```

Setup a virtual environment

Important

Open a new terminal session before proceeding with the next steps.

Windows

We will first install [pip](#) - a tool for installing and managing Python packages. *pip* can then be used to install *virtualenv*.

1. Download the [get-pip.py](#) script from the pip installer website. "Right click" --> "Save link/target as"
2. Open the command prompt (`cmd.exe`), navigate to the directory where the script is downloaded. In my case, this was `C:\User\vl\Downloads`, so

```
C:\Users\vl> cd Downloads
```

and then type:

```
python get-pip.py
```

Install *virtualenv*

```
pip install virtualenv
```

Note

There port of *virtualenvwrapper* for Windows called as [virtualenvwrapper-win](#). However, I could not get it working due to a [bug](#).

Setup a virtual environment

I will update this section when I find a solution. Until then, we will use the *virtualenv* scripts directly.

Create a virtual environment

We will create a new virtualenv called **site** to hold all necessary programs required for the building and hosting of our site. To do this on

 Linux

Open a terminal and type:

```
mkvirtualenv site
```

Output:

```
vl@laptop:~$ mkvirtualenv site

New python executable in site/bin/python
Installing setuptools, pip...done.

(site)vl@laptop:~$
```

 Windows

Open a command prompt and type:

```
virtualenv site
```

This should create a folder called **site**.

Using the virtual environment

Setup a virtual environment

The setup process above should already place you in the virtualenv. Notice the change in the prompt to include the name of the virtualenv

If this is not the case, you can always switch to the virtualenv using the following command

```
workon site
```

 Windows

Run the `activate` command

```
C:\Users\vl\site\Scripts\activate
```

The prompt should change to

```
(site) C:\Users\vl>
```

Within the virtualenv, you can install additional Python modules or run scripts installed by these modules.

To exit, type:

```
deactivate
```

You will be returned back to the prompt.

Note

You will always need to be in the virtualenv when you are working with the site.

Install Pelican, the static site generator

Install Pelican, the static site generator

[Pelican](#) is a static site generator written in Python.

We can write content in a text editor using [reStructuredText](#) or [Markdown](#) formats.

The use of templates make it easy to change the entire look of the site. Pelican has support for [Jinja2](#) templates.

It includes programs for generating and deployment of the site.

Switch to the **site** virtualenv we created before

Install Pelican using [pip](#), the Python package management system

```
pip install pelican
```

Pelican should now be installed with all dependencies.

Add support for Markdown

Out of the box, Pelican supports writing content using the [reStructuredText](#) syntax. To add support for writing content using [Markdown](#), we will need to install the corresponding Python module.

Install markdown using pip.

 Linux and  Windows

```
pip install markdown
```

Git

We will use [Git](#) for the deployment of our website i.e., pushing content to a remote server ([Github](#), in this course) to make the website available online.

Install Pelican, the static site generator

Installation

 Linux

Install Git using apt

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

 Windows

Download the MSI installer from the [Git](#) website.

Git-1.9.4-preview20140815.exe (as of this writing).

Install Git using the default options except for the "Adjusting the PATH environment" setting.

Select the second option "Use Git from the Windows Command Prompt" here (See figure).



ghp-import for Github deployment support

Install Pelican, the static site generator

The `ghp-import` script simplifies the process of publishing our website on [Github](#). It can be installed using `pip`

 Linux and  Windows

```
pip install ghp-import
```

With all the dependencies installed, we can now proceed towards creating the website.

Section 2: Setup website

Create the site

Let's create a basic site using Pelican.

First, create a new directory:

```
cd
mkdir samplesite
cd samplesite
```

Activate the **site** virtualenv we had created earlier. This needs to be done every time we need to work with the site.

Setup a basic structure for the site using the `pelican-quickstart` command

```
pelican-quickstart
```

We will be asked a series of questions for the initial site configuration.

Hint

You can press `ENTER` to accept the default value for any option.

Basic configuration

The first series of questions are related to the basic configuration of the site

We will be creating our website in the **samplesite** directory. Since we are already in this directory (`cd samplesite`), we will just use the default value `.`. The `.` (dot) here refers to the current directory.

Next, we enter a title for our website. I have used *Sample site*.

Section 2: Setup website

The author name will be displayed in blog posts.

For language of the website, I have used the default value (*en - English*).

```
Welcome to pelican-quickstart v3.3.0.
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? Sample site
> Who will be the author of this web site? Vimalkumar Velayudhan
> What will be the default language of this web site? [en]
```

URL prefix, pagination and scripts

If you already have a domain registered for the website, enter it here (URL prefix) otherwise, type **n**.

Article pagination enables navigation of blog posts. This is the number of blog posts displayed per page. I've used the default(10).

A `Makefile` enables us to use commands for site generation and hosting.

You should also enable the creation of the auto-reload script. This script generates the website whenever there is a change in content.

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

> Do you want to generate a Fabfile/Makefile to automate
```

Section 2: Setup website

```
generation and publishing? (Y/n) y
```

```
> Do you want an auto-reload & simpleHTTP script to assist  
with theme and site development? (Y/n) y
```

Hosting options

The next series of questions relate to how we are going to host our website online.

We will need to choose from one of the many options available - FTP, SSH, Dropbox, Amazon S3, Rackspace or Github.

I will be using *Github* as an example in this course. The Github user name of the account is **samplesite** and when published, the site will be available online at <http://samplesite.github.io>

```
> Do you want to upload your website using FTP? (y/N) n  
> Do you want to upload your website using SSH? (y/N) n  
> Do you want to upload your website using Dropbox? (y/N) n  
> Do you want to upload your website using S3? (y/N) n  
> Do you want to upload your website using Rackspace  
Cloud Files? (y/N) n  
  
> Do you want to upload your website using GitHub Pages? (y/N) y  
> Is this your personal page (username.github.io)? (y/N) y  
  
Done. Your new project is available at /home/vl/samplesite
```

At the end of the process, `pelican-quickstart` saves all settings to a file called `pelicanconf.py`.

Build the site

Here is a listing of files produced on running `pelican-quickstart`

```
(site)vl@laptop:~/sample-site$ ls -l

drwxrwxr-x 2 vl vl 4096 Aug 30 10:26 content
-rwxr-xr-x 1 vl vl 2197 Aug 30 10:26 develop_server.sh
-rw-rw-r-- 1 vl vl 1809 Aug 30 10:26 fabfile.py
-rw-rw-r-- 1 vl vl 3856 Aug 30 10:26 Makefile
drwxrwxr-x 2 vl vl 4096 Aug 30 10:26 output
-rw-rw-r-- 1 vl vl  830 Aug 30 10:26 pelicanconf.py
-rw-rw-r-- 1 vl vl  508 Aug 30 10:26 publishconf.py
```

With the basic site structure in place, we can now proceed towards building the site.

Build the site



Pelican includes a script that will watch for changed content and regenerate the site.

Run this script using the following command

```
vl@laptop:~/samplesite$ make devserver
```

Output:

```
Starting up Pelican and pelican.server
Pelican and pelican.server processes now running in background.

(site)vl@laptop:~/samplesite$
```

Build the site

```
DEBUG: Adding current directory to system path
DEBUG: Temporarily adding PLUGIN_PATH to system path
DEBUG: Restoring system path
DEBUG: Missing dependencies for asc
---
AutoReload Mode: Monitoring ``content``, ``theme`` and
``settings`` for changes.
---
DEBUG: Temporarily adding PLUGIN_PATH to system path
DEBUG: Restoring system path

-> Modified: theme, settings. re-generating...
WARNING: No valid files found in content.
... lines truncated
-> writing /home/vl/samplesite/output/index.html
-> writing /home/vl/samplesite/output/tags.html
-> writing /home/vl/samplesite/output/categories.html
-> writing /home/vl/samplesite/output/authors.html
-> writing /home/vl/samplesite/output/archives.html
Done: Processed 0 articles and 0 pages in 0.21 seconds.
```

The "Processed 0 articles and 0 pages" indicates that we haven't written any content yet! We will do so in the next section.

Windows

Use the `pelican` command directly to watch for changes and generate output

```
pelican -r content
```

Build the site

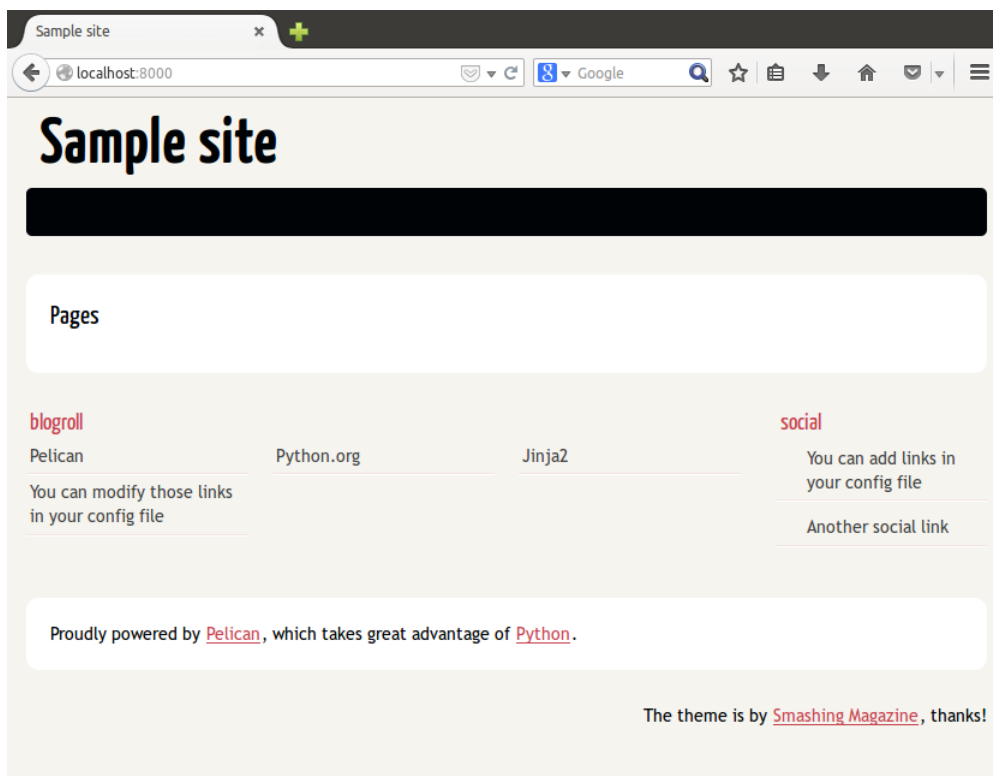
Preview the site

To preview the site on



Open <http://localhost:8000> in a browser.

The home page of the site should now be displayed.



To stop watching for changes, type:

```
vl@laptop:~/samplesite$ make stopserver
```

Note

You can also generate the site manually using

Write content

```
make html
```

Afterwards, you can serve the generated content using

```
make serve
```

To stop, press CTRL+C.

Windows

Open a new command prompt and run a HTTP server using Python itself.

```
cd C:\Users\vl\samplesite\output
python -m SimpleHTTPServer
```

Open <http://localhost:8000> in a browser.

The home page of the site should now be displayed.

We can now start adding new content to the website.

Write content

The site does not have any content yet. Let us see how we can add a new page and a new blog post.

Important

On Linux, the pelican development server should be running. Only then, Pelican will automatically generate the new page and post.

Write content

If this is not the case, you will have to generate the website manually using `make html`.

On Windows, you will have to start pelican using the command:

```
pelican -r content
```

Create new page

We will add an *About* page to our website. To do this, make a folder called **pages** inside the content directory

```
mkdir content/pages
```

Create a new file inside **pages** called `about.md` with the following content:

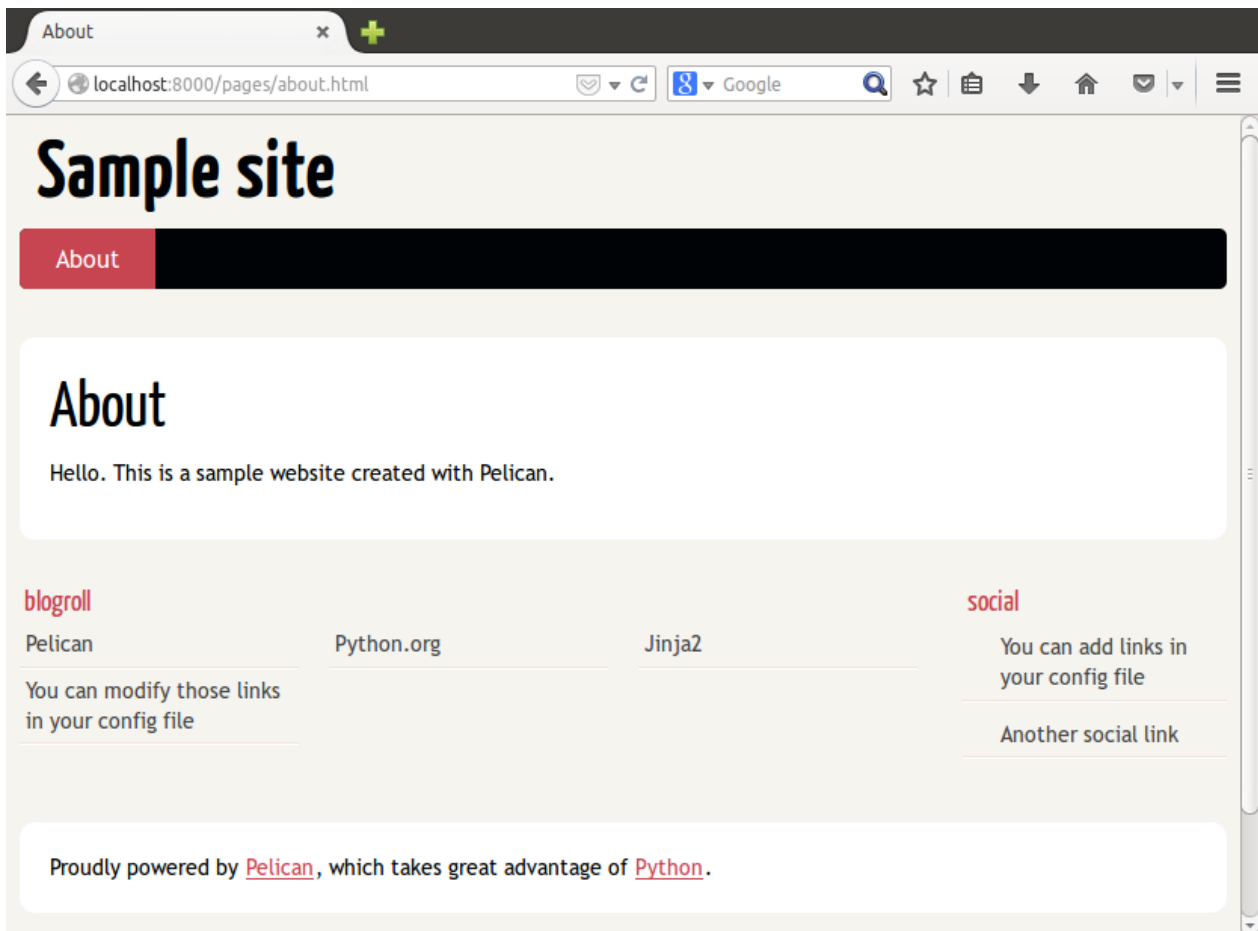
```
Title: About
```

```
Hello. This is a sample website created with Pelican.
```

Open <http://localhost:8000/pages/about.html> in a browser to view the new page.

The site's navigation menu will also be updated to include a link for the *About* page.

Write content



Create new blog post

We will now create a new blog post. Using a text editor, create a new file inside the **content** folder with the following content.

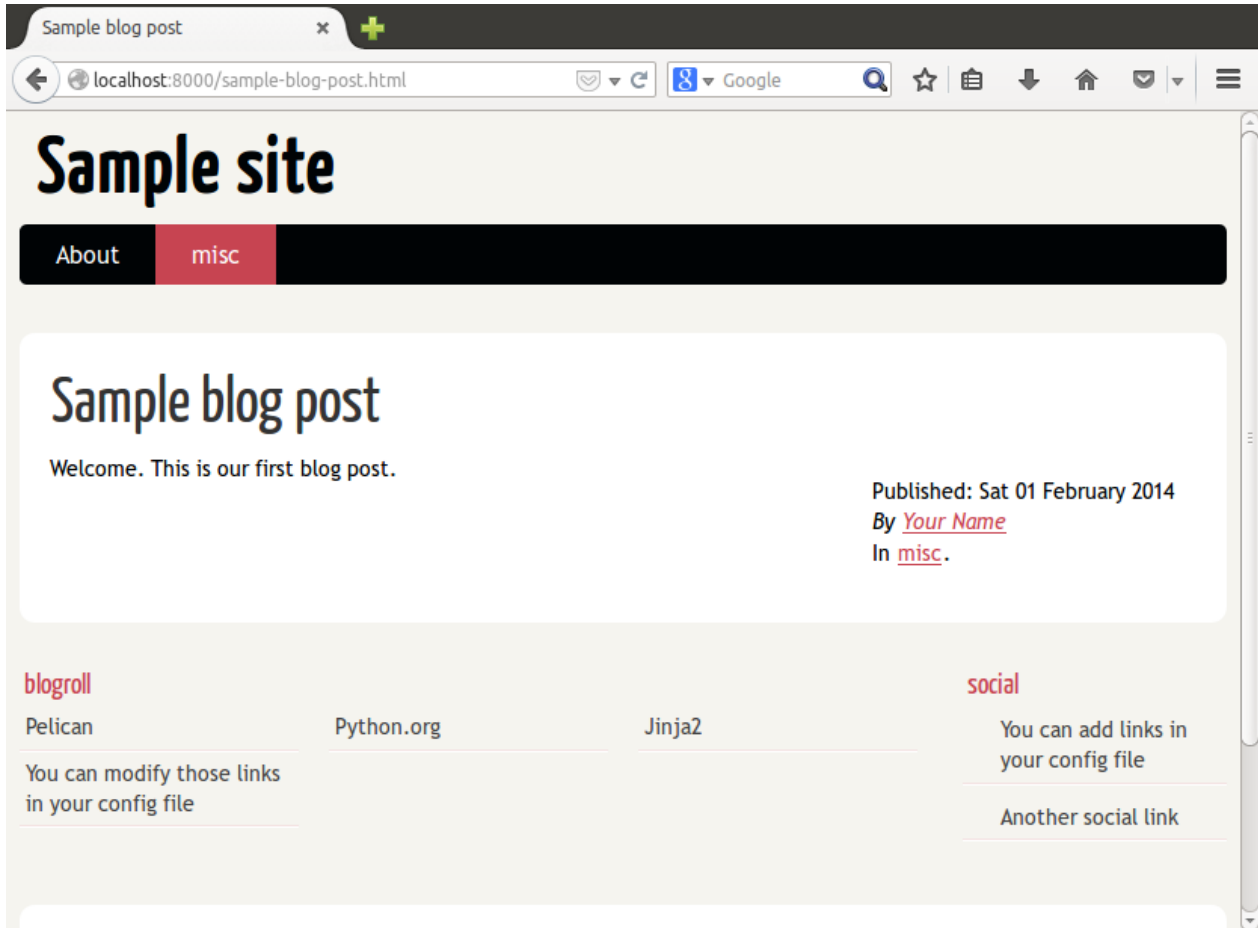
For example, **sample-post.md**

```
Title: Sample blog post
Date: 2014-02-01 19:45
Slug: sample-blog-post
Author: Your Name
Summary: This is a sample blog post

Welcome. This is our first blog post.
```

Write content

Open <http://localhost:8000/sample-blog-post.html> to view the new blog post.



Hint

You can assign posts to a category using Category :

```
Category: Python
```

Adding tags to a post can be achieved using Tags :

```
Tags: Pelican, Howto
```

This must be added to the metadata before post content.

We can now proceed towards publishing this site.

Section 3: Host website

Pelican generates a complete static version of our website in the **output** folder.

To host the website online, we should copy all the contents in this folder to a server. This needs to be done whenever there is a change in the website - new page, new blog post, theme or layout changes etc.,

For this course, we will host the website on **Github**.

Hosting on GitHub

We will first need to setup version control using Git before we can host the website on Github.

Setup a Git repository

To start, initialize a git repository in the site directory using:

```
cd samplesite
git init
```

Important

On Windows, you will need to use the "Git Bash" program to be able to use the commands for SSH connections and key generation.

Git Bash also provides auto-completion and colored output among other features.

If you haven't already done so, add your name and email address to git configuration:

Section 3: Host website

```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

Github configuration

To proceed, you will need to

1. Create an account at <http://github.com>.
2. Add an SSH public key to your Github account. Please follow the [instructions on github](#) for generating SSH keys and adding it to your account.

Warning

Without this step, the following publishing steps will not work!

3. You can verify that you have setup this correctly using the command:

```
ssh -T git@github.com
```

```
Hi user! You've successfully authenticated, but GitHub  
does not provide shell access.
```

Note

You might be prompted for the passphrase which you had used during the key generation.

Create repository on Github

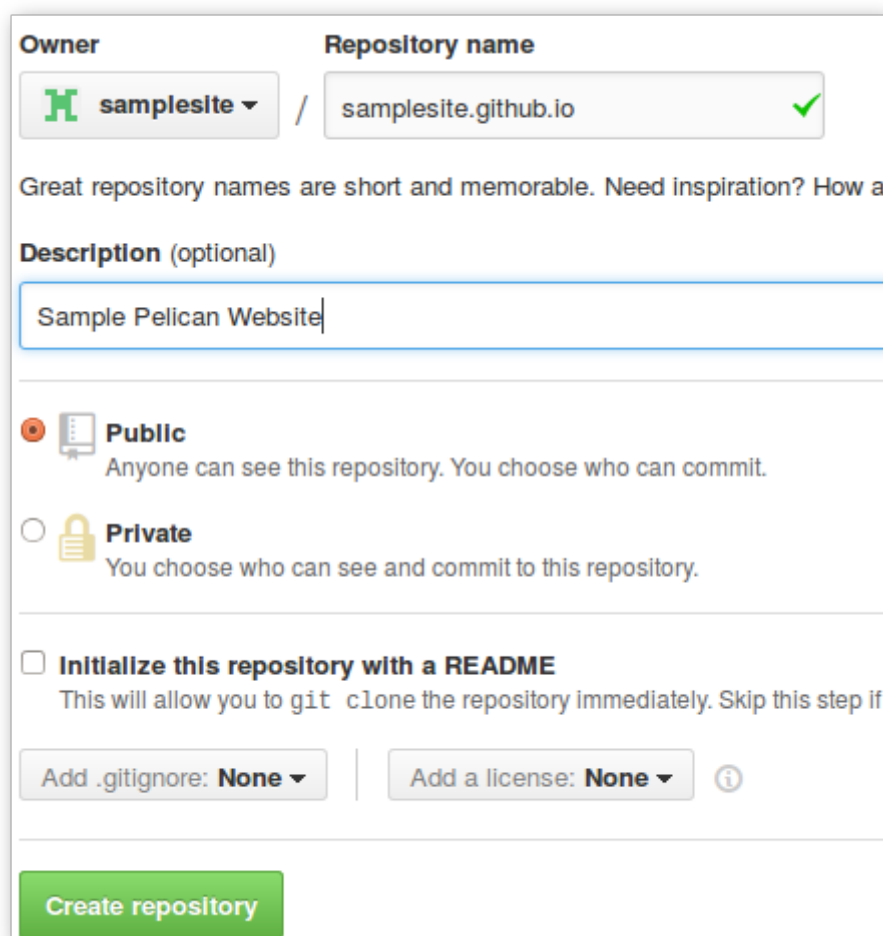
Section 3: Host website

On Github, create a new repository for the site using the following steps

1. Create a new repository using the **New repository** button.
2. For the **Repository name**, enter `user.github.io` where **user** is your user name on Github.

For a user **samplesite**, this would be `samplesite.github.io`

3. Click the **Create repository** button to create new repository.



The screenshot shows the GitHub 'Create new repository' form. At the top, the 'Owner' is 'samplesite' and the 'Repository name' is 'samplesite.github.io'. Below this, there is a 'Description' field containing 'Sample Pelican Website'. The repository is set to 'Public'. There are options to 'Initialize this repository with a README', 'Add .gitignore', and 'Add a license'. A green 'Create repository' button is at the bottom.

Adding a remote

Before we can publish our **samplesite** on Github, we will need to add a pointer to the repository we created on Github. This is called a **remote**. To do this:

Section 3: Host website

```
cd ~/samplesite
git remote add origin \
git@github.com:samplesite/samplesite.github.io.git
```

Here we add the remote Github repository with the name **origin**.

Note

Please replace **samplesite** with your repository name.

Publish site

To push content in the "output" folder and publish the website on Github, we can use

 Linux

```
make github
```

Output:

```
(site)vl@lenovo:~/samplesite$ make github

pelican /home/vl/samplesite/content
-o /home/vl/samplesite/output
-s /home/vl/samplesite/publishconf.py

WARNING: Feeds generated without SITEURL set properly may not
be valid
Done: Processed 1 article(s), 0 draft(s) and 1 page(s)
```

Section 3: Host website

```
in 0.19 seconds.

ghp-import -b master /home/vl/samplesite/output
git push origin master

Counting objects: 60, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (39/39), done.
Writing objects: 100% (60/60), 34.62 KiB | 0 bytes/s, done.
Total 60 (delta 16), reused 48 (delta 15)
To git@github.com:samplesite/samplesite.github.io.git
 * [new branch]      master -> master
```

Windows

```
(site) C:\Users\vl\samplesite> python
C:\Users\vimal\site\Scripts\ghp-import -b master output

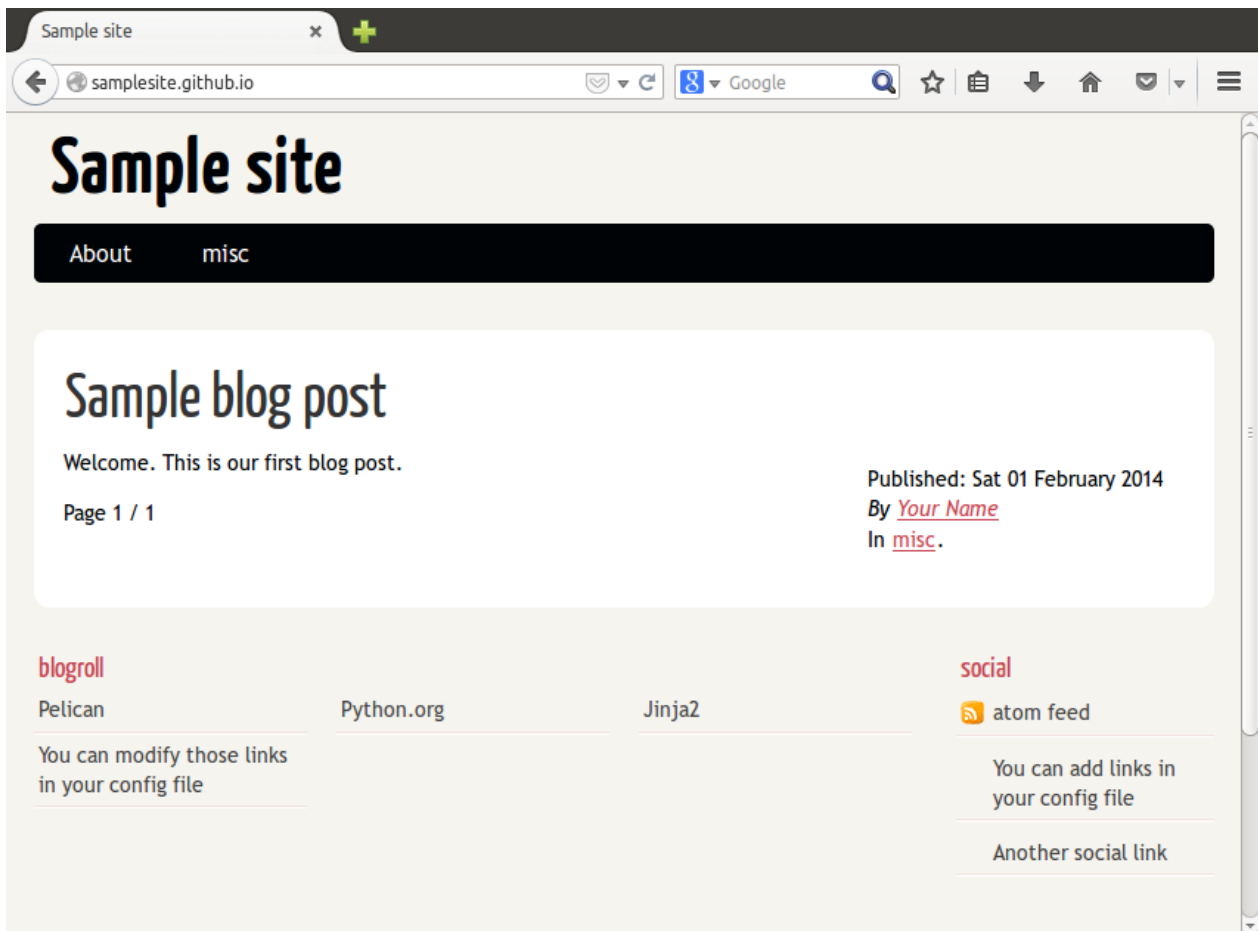
(site) C:\Users\vl\samplesite> git push origin master
```

After few minutes, the website will be available online at <http://user.github.io>. In this example, it is <http://samplesite.github.io>.

Note

According to Github, it might take upto ten minutes before the page is visible online after the first push. The next updates will be a lot faster.

Updating the site



Updating the site

Let's update the site by creating a new blog post.

Create a new file **second-post.md** in the **content** folder with the following text:

```
Title: This is our second blog post
Date: 2014-02-02 20:00
Slug: our-second-blog-post
Author: Your Name
Summary: This is our second blog post

This is our second blog post
```

Updating the site

Save **second-post.md** and load <http://localhost:8000> in a browser to view the new blog post.

Push changes

After few minutes, load <http://user.github.io> in a browser.

Section 4: Links

Links

Git

Website - <http://git-scm.com>

Documentation - <http://git-scm.com/documentation>

Github

Website - <http://github.com>

Help - <http://help.github.com>

Github Pages - <http://pages.github.com>

Generating SSH Keys - <http://help.github.com/articles/generating-ssh-keys>

ghp-import

Website - <http://github.com/davisp/ghp-import>

Python

Website - <http://python.org>

Pelican

Website - <http://blog.getpelican.com/>

Documentation - <http://docs.getpelican.com/>

Ubuntu

Website - <http://ubuntu.com>

Documentation (version 14.04) - <http://help.ubuntu.com/14.04/index.html>

virtualenv

Documentation - <http://www.virtualenv.org/en/latest/>

Section 4: Links

virtualenvwrapper

Documentation - <http://virtualenvwrapper.readthedocs.org/en/latest>