# Hugo and GitHub Pages Tutorial

## Table of Contents

## Goal

This tutorial will show you how to create, maintain, and host a website using Hugo and GitHub Pages. The goal is to make this process painless, easy, and quick. At the end of the tutorial, you'll have learned how to create and deploy a site, as well as developed some familiarity with some common tools.

**Disclaimer**: This tutorial assumes you are using MacOS. If you aren't, you will need different tools to install Hugo and some terminal commands may be different.

## Before you start

Make sure you meet the following pre-requisites before starting the tutorial steps:

- Create a GitHub account
- Install a text editor (I like to use Atom)

## Part 1 - Install your tools

### Goals

The goal of this section is to install the tools needed to create and deploy your Hugo. The tools are:

- Homebrew is a package manager that makes installing Hugo and Git very, very simple.
- Hugo is the static site generator that you will use to create your website.
- Git is a version control system you will use to deploy your website on GitHub Pages.

### Steps

You can install each of these tools with the terminal. To open the terminal, type "terminal" in Spotlight (the magnifying glass in the upper right corner of your Mac) and select the Terminal application.

#### Step 1 - Install Homebrew

Install Homebrew, by entering the following code into the terminal:

```
/bin/bash -c "$(curl -fsSL \
    https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

To verify that your Homebrew installation was successful, type `brew help` in your terminal. If you get an output, Homebrew was successfully installed. If you get `command not found`, Homebrew hasn't been installed.

#### Step 2 - Install Hugo

Now that you have Homebrew installed, use it to install Hugo. In the terminal, enter the following:

```
brew install hugo
```

To verify that your Hugo installation successful, enter `hugo version` into the terminal. If successful, a validation like this will appear:

```
Hugo Static Site Generator v0.13 BuildDate: 2015-03-09T21:34:47-05:00
```

### Step 3 - Install Git

You will perform a similar call to install Git. In the terminal, enter:

```
brew install git
```

To verify that your Git installation was successful, enter `git --version` into the terminal. If successful, a Git version will appear.

## Outcome

Congratulations! You've installed Homebrew, Hugo, and Git onto your computer. You now have the tools needed to create your website.

# Part 2 - Create your site

In this section you will create your website's "skeleton". A skeleton makes configuring and updating your site very easy.

## Goals

The goals of this section are:

- create two repositories
- build a bare-bones Hugo site and add its theme
- "link" your remote and local repositories

## Steps

### Step 1 - Create your GitHub repositories

You need two repositories to host your website with GitHub:

- The first will be your GitHub Pages repository that launches your website: `https://github.com/<USERNAME>/<USERNAME>.github.io`
- The second will be your general GitHub repository that stores your Hugo source files: `https://github.com/<USERNAME>/website`

The name of your GitHub Pages Repository *must* be the same as your GitHub account name. It should resemble `<USERNAME>.github.io`. This is why you get only one GitHub Pages Repository.

1. In the upper right corner of any GitHub page, select the **+** drop-down menu, then select **New Repository**.
2. Enter your username in the **Repository Name** field.
3. Check the **Add a README file** box.
4. Select **Create Repository**.

Follow similar steps to create your website repository, but this time name the repository `website`.

1. In the upper right corner of any GitHub page, select the **+** drop-down menu, then select **New Repository**.
2. Enter `website` in the **Repository Name** field.
3. Check the **Add a README file** box.
4. Select **Create Repository**.

## Step 2 - Build your website and add a theme

After you build your website, choose a Hugo theme. For this tutorial, we'll be using the Coder theme.

1. Build your website by entering the following code in the terminal:

```
hugo new site website
cd "website"
```

2. Add your website's theme by entering the following in the terminal:

```
git init
git submodule add https://github.com/luizdepra/hugo-coder.git themes/hugo-coder
```

> **IMPORTANT:**
> If you aren't using the Coder theme, be sure to replace your theme's `<URL>` and `themes/<THEME_NAME>` above.

### Step 3 - Link your repositories

Hugo builds websites locally and you must "link" them to your remote (GitHub) repositories.

1. Link your remote `website` repository to the local `website` directory. In the terminal, enter:

```
git remote add origin https://github.com/<USERNAME>/website.git
```

2. Merge your remote repository with your local directory. In the terminal, enter:

```
$ git add .
$ git commit -m "Initial commit"
$ git push origin master
```

3. Link the remote `<USERNAME>.github.io` repository to a local `public` directory. In the terminal, enter:

```
mkdir "public"

$ cd public
$ git remote add origin https://github.com/<USERNAME>/<USERNAME>.github.io
$ cd ..
```

## Outcome

Congratulations! You've successfully created your remote GitHub Repositories, built your local Hugo Site, and linked them together! In the next section, you will go about configuring and updating your website.

# Part 3 - Configure (and maintain) your site

Hugo boasts a very fast preview feature. This allows you to quickly check any changes you make to your website. Once satisfied with your changes/updates, use a simple script to deploy or update your site.

## Goals

The goals of this section are:

- Configure and preview your website
- Create a deployment script
- Update your website with it's first change

## Steps

### Step 1 - Configure and preview your website

Before you can preview your website, you need to configure the `toml` or `yaml` file. The simplest way to do this is to paste your theme's example site into our website's directory. This provides you with a functioning demo site that you can customize as you see fit.

1. Paste all the contents from `website > themes > hugo-coder > exampleSite` into your `website` directory.
2. Preview your website by entering `hugo serve` in the terminal while in the `website` directory.
3. Visit http://localhost:1313/ in your web browser to preview. Once you are finished, press Ctrl+C to end the preview.
4. Customize your website by updating your Markdown files. Now that your local site is functioning, you can customize as you see fit and preview as many times as necessary.

> **WARNING**
> In your config file, make sure to change the `baseURL` to `<USERNAME>.github.io` , otherwise your GitHub Pages repository won't read your config file correctly.

### Step 2 - Deploy Your Website

1. Add your generated site as a submodule. In the terminal, enter the following code:

```
git submodule add --force -b master \
    https://github.com/<USERNAME>/<USERNAME>.github.io.git \
    public
```

2. Deploy your website by entering the following into the terminal:

```
$ cd public
$ git init
$ git add .
$ git commit -m "Initial commit and deployment"
$ git push origin master
```

Congratulations! Visit `https://<USERNAME>.github.io` to view your live website!

## Step 3 - Create a deployment script

Creating a deployment script saves you time in the future when you need to update your site.

1. Create a deployment script to deploy your website. Create a text file named `deploy.sh` and paste the following:

```
#!/bin/sh

set -e

printf "\033[0;32mDeploying updates to GitHub...\033[0m\n"

hugo -t <YOUR_THEME>

cd public

git add .

msg="rebuilding site $(date)"
if [ -n "$*" ]; then
        msg="$*"
fi
git commit -m "$msg"

git push origin master
```

**WARNING**

> Replace your theme name in `hugo -t <YOURTHEME>`

3. Deploy updates to your site by with the following call, replacing the optional message:

```
./deploy.sh "OPTIONAL_MESSAGE"
```

## Outcome

Great work! You've created your script that automates some of the steps for publishing to your site.

# What you've learned

In this tutorial you've installed the tools needed to make your Hugo website. Then you created your website, made changes to it, and then published it. You even automated some of the steps so when you update your site again, the process will be quicker.