Using git submodule for Hugo themes

June 16, 2018 (Last Modified: June 17, 2018)

When I first started tinkering with **Hugo** for static website generation, I would find a **Hugo theme** that I like, `cd` into my themes directory and would simply `git clone` the repo into my website. This works, makes sense and is the most common install instructions from the theme authors.

However, you will likely run into an issue where you want to make an edit to the theme only for your website. Once you make the edit, you can't issue a Pull Request because this is specific to you. It then gets tricky over time to merge changes from the master repo into your theme without losing any of your edits.

The current way I approach this is as follows:

1. Fork the theme I want into my github account
2. Add the theme to my hugo site with `git submodule`
3. Tweak the theme and track the changes with git into my forked repo
4. Update my main website repo to the last theme commit
5. As needed, `git pull` changes from the original theme repo, merging changes if there are conflicts

SEARCH...

RECENT POSTS

How to generate an Android (React Native) SBOM in CycloneDX format

How to generate a Nodejs SBOM in CycloneDX format

Source Code vs Binary Analysis for SBOMs

Technical Introduction to Software Bill of Materials (SBOMs)

HOWTO setup a private git server on Ubuntu 18.04

CATEGORIES

howto

misc

programming

security

Below is a step by step with examples (so I can refer back to it!) with the assumption that you are using git for version control on your hugo website.

Step 1: Fork your targeted theme

For this blog, we'll target the **Mainroad theme**. Once you visit the theme's webpage, follow the link to it's **github repo** and fork the repo into your account. Note, this assumes you have a github account and are logged in. From that point forward, your interactions with the theme will be through your fork of the repo (e.g. **https://github.com/ahoog42/Mainroad**.

Step 2: add the forked repo to your website

We're now ready to add the theme to our Hugo website. Using your terminal app, `cd` into the themes directory of your website and run `git submodule add` with the URL of your forked repo, e.g.:

```
$ cd ~/git/andrewhoog.com/themes
$ git submodule add https://github.com/ahoog42/Mai
Cloning into '/Users/hiro/git/andrewhoog.com/theme
remote: Counting objects: 1823, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 1823 (delta 7), reused 16 (delta 7),
Receiving objects: 100% (1823/1823), 2.52 MiB | 4.
Resolving deltas: 100% (984/984), done.
```

Step 3: track changes to your forked theme

From here, you would make any edits to your theme. With git submodule, one important thing to understand is that while your parent git report is aware of the submodule, when you want to

tutorial

TAGS

ANDROID DEVOPS GIT GMAIL HUGO MACOS MOBILE NODEJS SBOM SPFEXPERT

SOCIAL

Twitter

Instagram

LinkedIn

GitHub

YouTube

check the status of changes to your theme or commit changes, you need to be in that submodule's directory (e.g. ~/git/andrewhoog.com/themes/Mainroad):

```
$ cd ~/git/andrewhoog.com/themes/Mainroad
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

  new file:   layouts/partials/categories.html
  modified:   layouts/partials/sidebar.html
  new file:   layouts/partials/tags.html
```

From here you do your normal git workflow like status, add, commit and push.

Step 4: update main repo with latest theme commit

In your main hugo website repo, the submodule tracks the latest submodule commit you explicitly checked in. In the case of the initial adding of the submodule, it points to the head of the submodule. But in Step 3 you made changes and committed them. So you need to update where the main repo's submodule commit HEAD points to.

The way this will show up is when you do a `git status` from your main website repo, in the "Changes not staged for commit" section you will see your theme submodule listed and one of two statuses:

- (modified content) - which means you've modified your submodules content but have not committed those changes

to the submobule (follow Step 3)

- (new commits) - changes in your submodule were committed

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be
  (use "git checkout -- <file>..." to discard chan
  (commit or discard the untracked or modified con

  modified:   themes/Mainroad (new commits)

no changes added to commit (use "git add" and/or "
```

Luckily, after following Step 3 it's a very simple process! You just treat it like any other change: add, commit and push. The **"Git Submodules basic explanation"** gist from gitaarik has more details.

Step 5: merge changes from original theme

I've not yet had the need to do this but on any active themes, there will obviously be changes you'll likely want to pull into your theme. One could hope that any changes upstream will not conflict with yours so it'll be simple to merge the changes in. However, you might find you have to manually merge if changes conflict. When I run into this, I'll push more details and update here but both of techniques will likely do the trick:

- **Keeping a fork up to date** from CristinaSolana
- Official GitHub documentation
  - **Configuring a remote for a fork**
  - and then **Syncing a fork**

HUGO GIT

About Andrew Hoog

I like to tinker in mobile forensics, security, privacy, tools development and nodejs. I'm an author, inventor, expert witness and co-founder of NowSecure.

« PREVIOUS
Fixing Async Calls Missing Callbacks

NEXT »
Configure git submodule for no password (ssh)