CarpentryCon 2018 Workshop Building Skill-Up Session

Authors:
Elizabeth Wickes, Aleksandra Nenadic

Skill-up GitHub repository

https://github.com/anenadic/carpentrycon-workshop-website-building-skillup

Please add comments!

We know that the screen shots come from two different demos, but otherwise please add comments about anything confusing.

Overview

1. Agree with your host and co-instructor who is doing the workshop website
2. Decide on your workshop ID (slug)
3. Copy The Carpentries' workshop website template
4. Customise your workshop website
5. When things go wrong - errors and troubleshooting

Pre Reqs:

This lesson presumes that you have GitHub account and can log in, but will take you through everything else.

Detailed guide

Step 1: Agree who is making the workshop website

Only one website needs to be made for each workshop, and only one person can start the creation. This means that you need to pick one person in your instruction team to complete this process. This usually is done by the organizer or host, so you should communicate with your instruction team (and separate coordination team, if there is one) to see who should have this responsibility.

Some groups may have an organization account that this should be made under, which will be highlighted in the relevant step. You may also want to add your co-instructors and other relevant team members to be collaborators for the repository so they can help make edits. This option will also be highlighted at the relevant step.

Step 2: Decide on your workshop ID (slug)

Workshop ID (often called a slug in URLs) is used to uniquely identify you workshop in The Carpentries' record keeping system AMY (which is our instruction and human tracking database system, https://amy.software-carpentry.org/account/login/).

The same ID is used as the name of your workshop repository in GitHub. Using the same ID in different places is optional but keeps things nicely connected and easily identifiable.

Slug is normally in the format: `<yyyy-mm-dd>-<location>-<workshop_type>`, e.g.:

2018-05-15-manchester-swc
2017-11-03-chicago-dc
2016-05-29-online-ttt

No formal rules exist yet for the slug format; variants have been in operation and sometimes new structures are used as the types of our workshops are expanded. The only rule is that the slug needs to uniquely identify your workshop.

As long as you include a date, location, and type, your workshop will be uniquely identified (clashes possible but in practice never occurred so far). Types are usually included because it helps us understand what kind it was when reading a list of slugs on a page. Similarly, the location might be the name of the institution or group and not a geographic identifier.

The slug may be provided to you by The Carpentries' staff (particularly for the trainer workshops) when the workshop is centrally organized by staff.

Note that the date should be in the standard `yyyy-mm-dd` format. While workshops run two days, we usually report the first date as being the 'date' of the event. Check in with your site host or local organizer about any local naming conventions to follow for these labels. Your local conventions may disagree with this or add more to it, which is fine as this is an informal system.

Workshop type keys (informal and growing):

- swc: Software Carpentry
- dc: Data Carpentry
- ttt: Train The Trainer (a.k.a. Instructor Training)
- lc: Library Carpentry
- more workshop types likely coming soon

Step 2: Copy The Carpentries' workshop website template

The copying process uses a tool that may be unknown to some GitHub users. Whereas you may know about **forking**, this process will allow you only to create a 'copy' or a 'fork' of the original repository once and many backend hooks to the original repository remain to enable contributions to be pushed upstream to the original repository. As a workshop host, organizer or instructor, you will likely have to create workshop websites multiple times (one for each workshop), so you need to create a proper 'copy' (and not a 'fork') of the original repository.

You are required to use a feature called **GitHub importing**. Nothing will stop you from forking, but it won't work in the long run. This allows you to make a copy of someone else's repository into your account. All the commit and contribution metadata will be retained when you copy it, but that repository will be disconnected from the original. Most important, you can import a repository multiple times to have many copies of it (in this case, one for each workshop you organise).

Find The Carpentries' workshop website template repository you will be copying at:
https://github.com/swcarpentry/workshop-template
You will also find instructions there for future reference.

The README of this repository has the most up to date directions for this process, and should be regularly referenced whenever you need to do this. Important note: log into your GitHub account before you start this process. Otherwise the import tool link will look like a '404 Not Found' error.

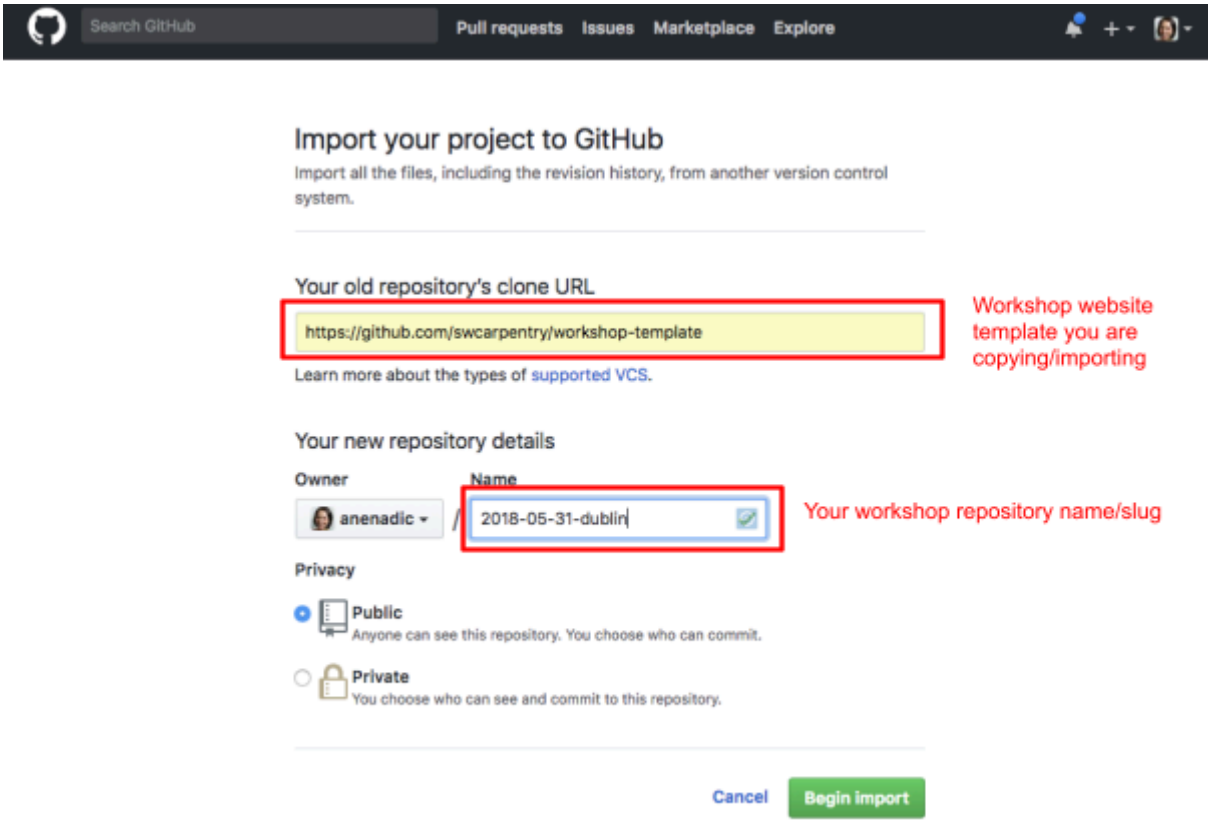
1. Log into your GitHub account.
2. Go to the workshop template repository at:
   https://github.com/swcarpentry/workshop-template
3. Scroll down to review the setup directions, which will have a link to the GitHub import tool to do the copy of the repository. Click the Import Tool link, preferably into a new window or tab. (this is in item 2, and will link to: https://github.com/new/import).
4. The importing process can seem like a lot of information, but there's not that much you need to do. Focus on only changing what you're being told to mess with.
5. Before you start this process you must have the workshop ID slug determined.
6. We'll be walking you through the importing steps below.

Once you open up the GitHub import tool, you'll need to provide two pieces of information. Paste the original workshop template's URL into the "Your old repository's clone URL". Below that there is a section for "Your new repository's details". Most of the time the default selections will be fine. The text box under "Name" is where you want to paste in the workshop ID/slug.

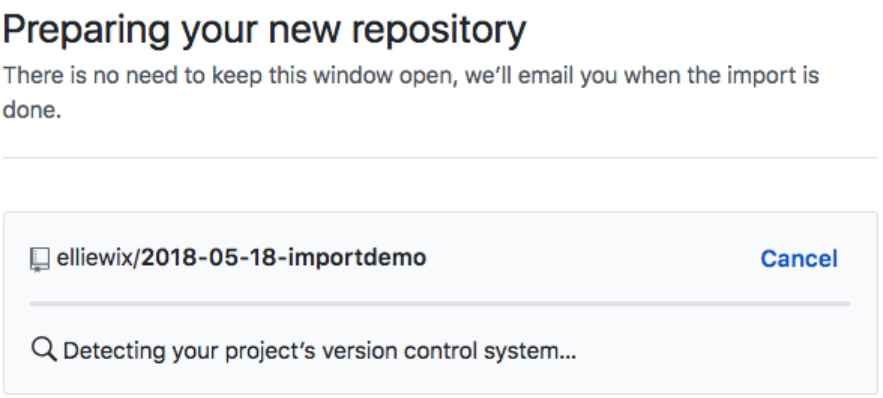
Optionally, you may want to create this under an organizational account in GitHub. You can change who the owner is with the drop-down under "Owner".

You will more than likely want to make this **public**, which is the default setting. However, you can choose to switch that to **private** if needed and if you are allowed to create private repositories in GitHub (for which you need a special approval or an organisational account). You may want to do this until you're happy with the content or you are practicing this process and don't want to advertise a workshop that doesn't exist.

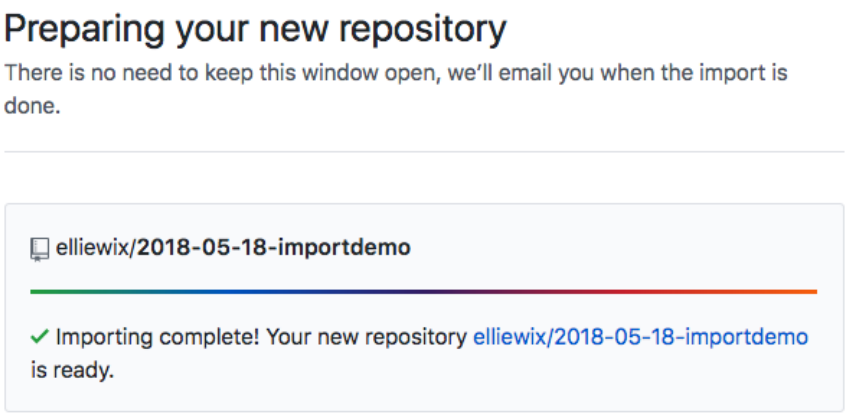
Once you're happy with all the selections, click the big green "Begin import" button.



The import process may take several minutes to complete and you will receive an email from GitHub once it is done.



This is what it will look like once the process has been completed.

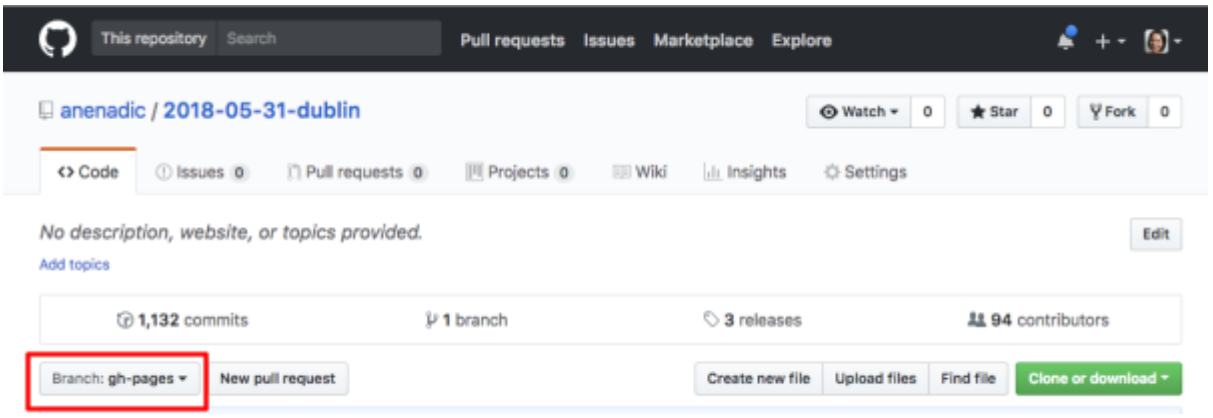


Step 3a: Customise your workshop website from GitHub

Once the import process has completed, the import page will have a link to the new repository that you can follow.

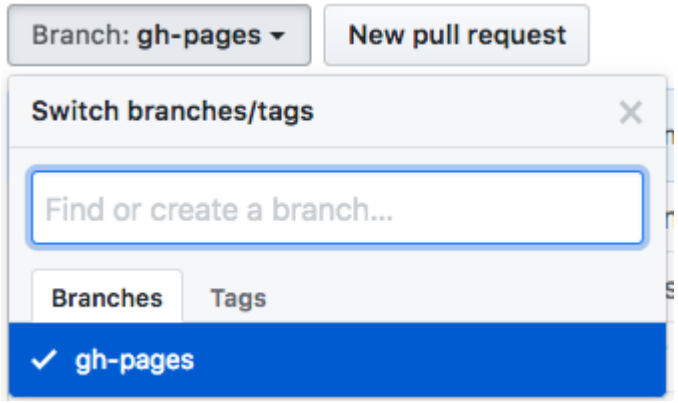
However, you may have navigated away from that page during the import process. You can still navigate to it from your main account page if that has happened. Going back to your main dashboard (what you see when you load GitHub as a logged in user). This new repository should be in the list of "Your repositories". Click on its name to go into the main repository's page (the root of the repository).

This will look like a normal repository, but with a lot of extra content and information that you may not be used to if you aren't a heavy GitHub user. Once you've gone through this import process, the previous contributor information will remain, but none of them will receive notifications about anything that you do to this repository. So commit away without fear!



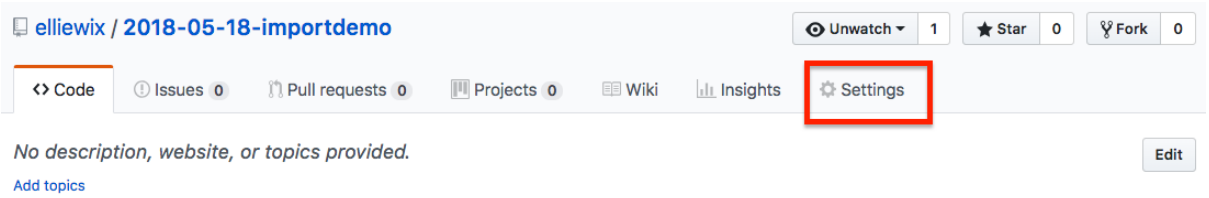
Going back to the repository's main page, we need to double check you are on `gh-pages` branch. This option tells GitHub to render your repository's content as a website. You can check this by looking at the drop down menu that begins with "Branch:" This drop down is just below the main metadata bar at the top of the page. It should say "Branch: gh-pages". Don't change a thing if that's what you've got.

If it says something else: you should be able to click it and select `gh-pages`.

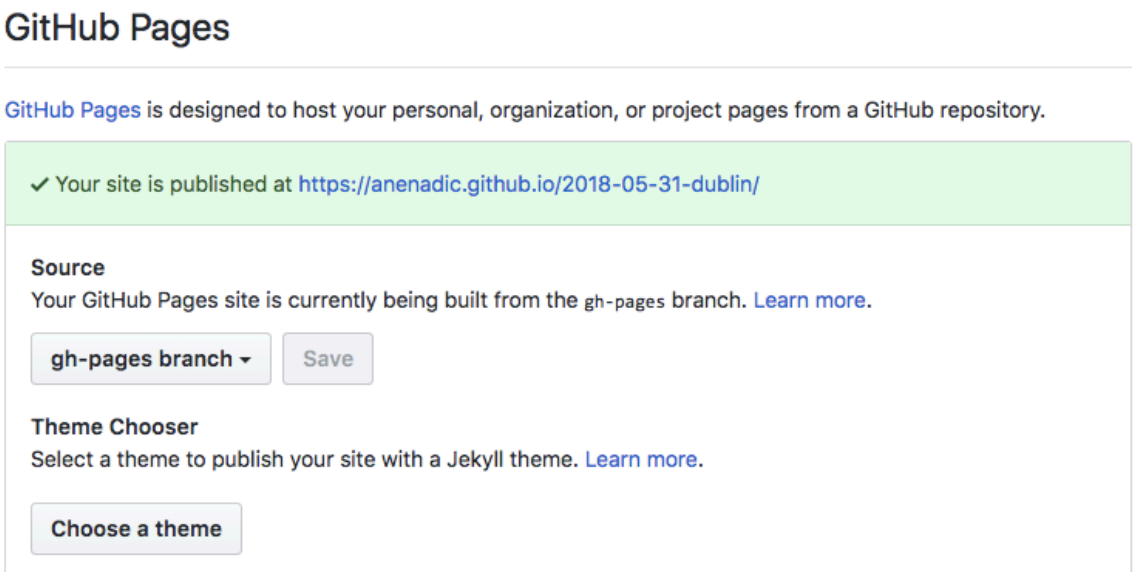


We already mentioned that by putting the content of your repository on the `gh-pages` branch, you are telling GitHub to render it as a website. This is done auto-magically by GitHub and you'll need to figure out the URL based on the information you know, as it won't be auto-populated anywhere on the page. Your workshop website will be accessible at: https://<your-github-name>.github.io/<your-workshop-slug>/.

You can find this within Settings as follows: at the top of the page directly under the repository's name you'll see a row of links. One of them should be Settings, with a gear icon. Click on that to access the settings control area.



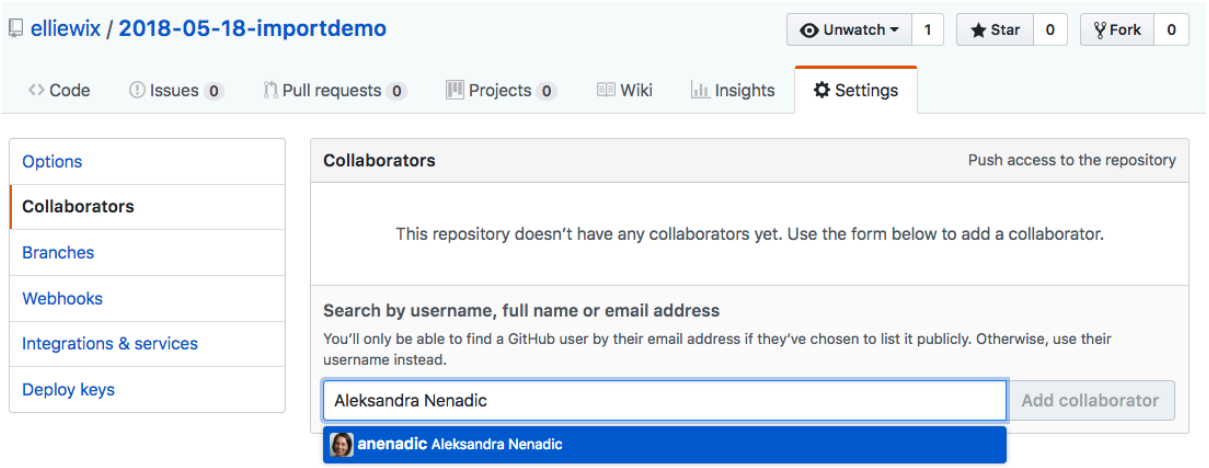
The default area you should arrive at inside of settings is a general page called Options. This is a very long page, so you'll need to scroll down of search to find the panel about "GitHub Pages". This area should look similar to the screenshot below. There will be a large green banner that says "Your site is published at…" followed by the URL to access your page. There's nothing that you'll need to change about this box.



Optional: add collaborators who can make edits to the page

While you are in Settings, you might want to add some of the hosts, organizers, instructors, or helpers as collaborators for the repository. This will give them permission to make changes to the repository (and your workshop website).

You can add collaborators by accessing the “Collaborators” section of Settings. The link should be on the left and just under “Options”. You can add usernames or emails associated with a person’s GitHub account. GitHub will send these people an invitation they will need to accept before they can gain access.



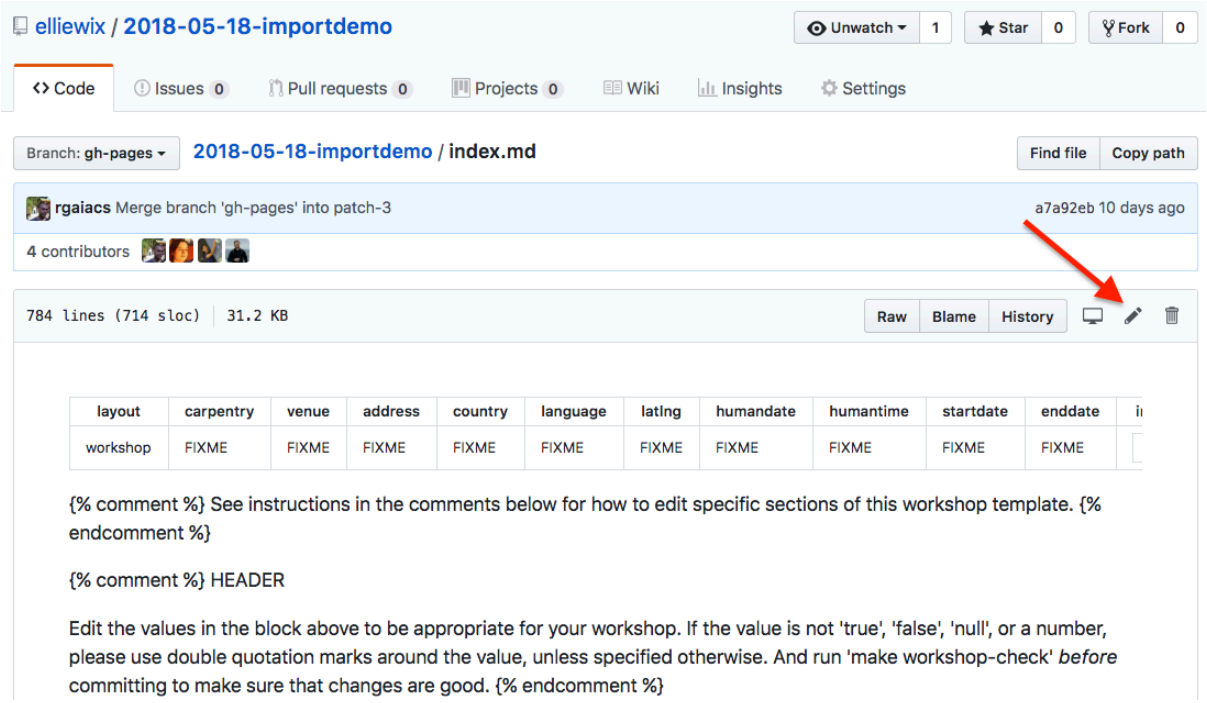
3a.1 Configure your workshop details in index.md

Now you can go in and start editing the pages to add your local workshop information. This all can look very cryptic, but there are code comments all over to guide you along. Best yet: you can do all the changes directly in GitHub (although we will show you below how to do this from command line too).

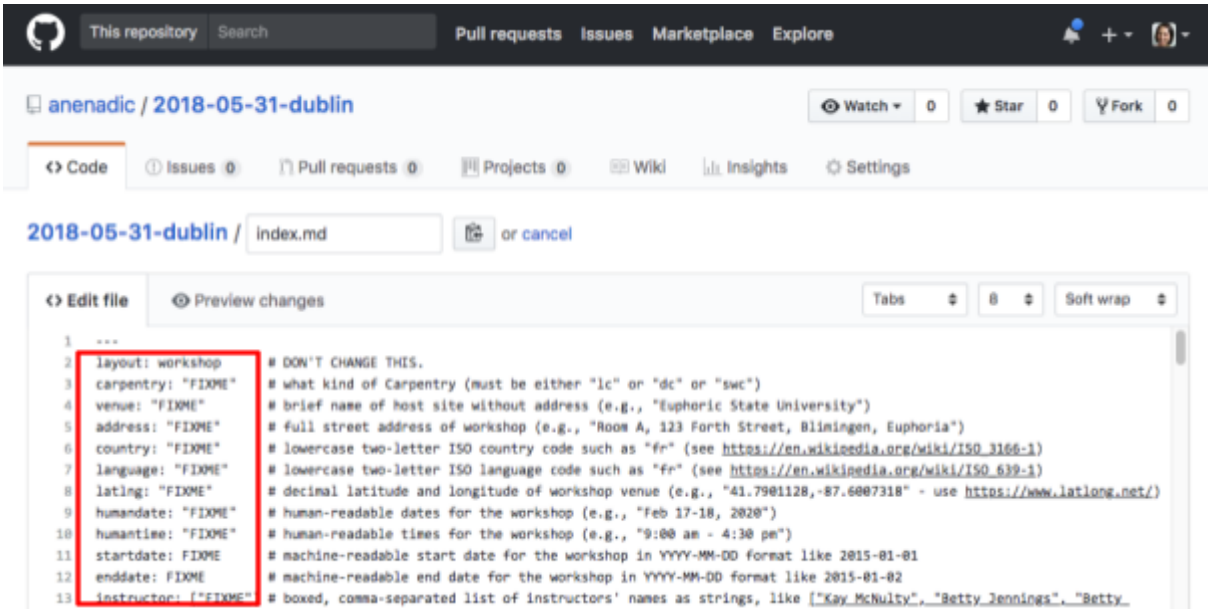
Your first stop should be at the “index.md” file. This will be in the file list in the main repository page (i.e. the root of the repository). Search for “index.md” if you’re having trouble finding it.

Click the filename link that GitHub is offering in that file list. This will take you to the page to view the contents and information about that file. You’ll be greeted with the contents displayed and some metadata about it.

You need to edit the file in GitHub. In the panel that has all the contents of the file, you should see that there is a header area. On the right side you’ll see some buttons for Raw, Blame, and History (use one of these words with a search to get help finding this area). To the right of those you should see a pencil icon. Click the pencil to open up the edit page for this file.



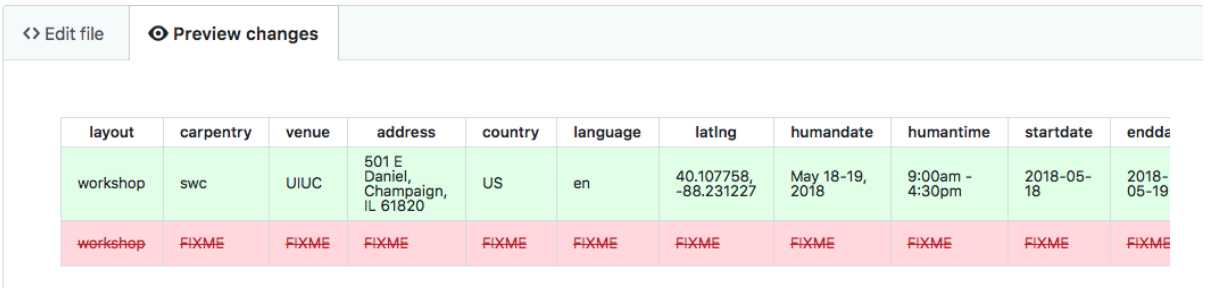
When the page loads with the edit options, take a moment to get resettled. It’ll look very different from the rendered Markdown that was originally shown. Look over to the top left of the raw file (as highlighted in red in the screenshot below). You should see a column of data labels and values, e.g. “carpentry”, “venue”, “address”, etc. These labels are followed with a colon (:) and then a “FIXME” or some other placeholder.



Add your relevant content to each area. The code comments to the right will prompt you about what to add. You can always edit this again if you make a mistake or don't know an answer at the moment.

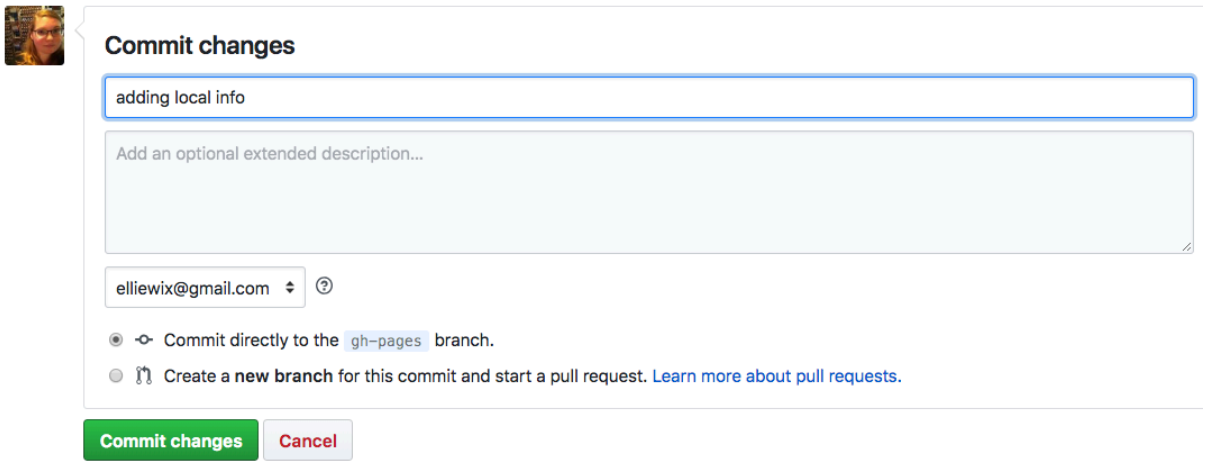
To make the edits, you should be able to double click the FIXME text and delete it leaving the quotes in place if there were quotes already around the FIXME text. There are a few of these that are meant to be machine readable and don't have quotes. Also, some of these data fields want lists of data. The code comments are there to help you navigate, so follow the suggested templates.

Once you've finished adding your answers, you can preview what you've done. At the top of the text area you should see a link that says "Preview" with an eye. Click on that and it'll highlight the rendered changes in markdown.



All the fields should look pretty readable.

Once you're happy with your changes, it's time to commit them! Scroll to the bottom of the page to the commit area. You need to provide a short commit message in the small text box, and then click "Commit changes". There are a lot of other options in here, and you should be able to leave them at default.



Once you click commit, it will reload the page to show your changes.

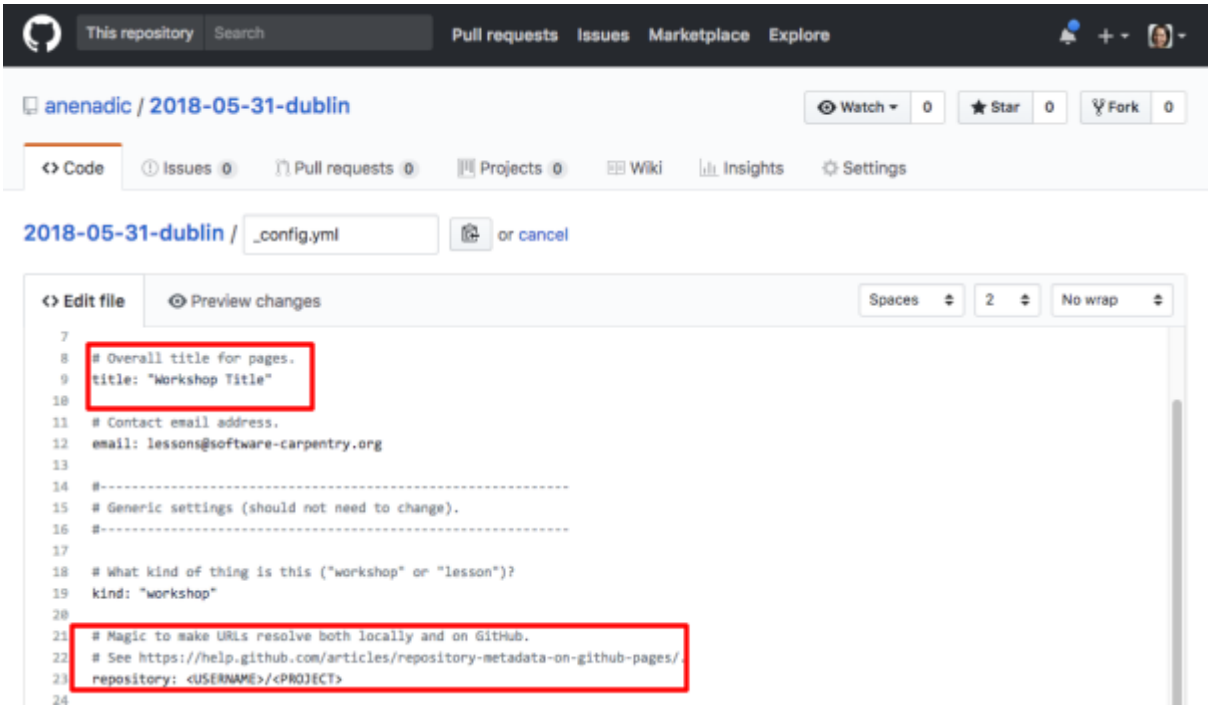
We're going to repeat this edit process across a few other files: find the file's page from the main repository file list, click the pencil icon to edit it, make the edits, and then commit them. The edit and commit process will be the same for every page, so we'll focus on guiding you through what changes to make for each page after this one.

3a.2 Configure your workshop details in _config.yml

Going back to the main repository page. You can find the "_config.yml" page. Click on it and open it up for editing as you just did for the index page.

Normally, you need not touch this file. In practice, not many people know that you still have to tweak it (this will probably change in the future), but it is not critical.

Modify parameters `title` and `repository` only.

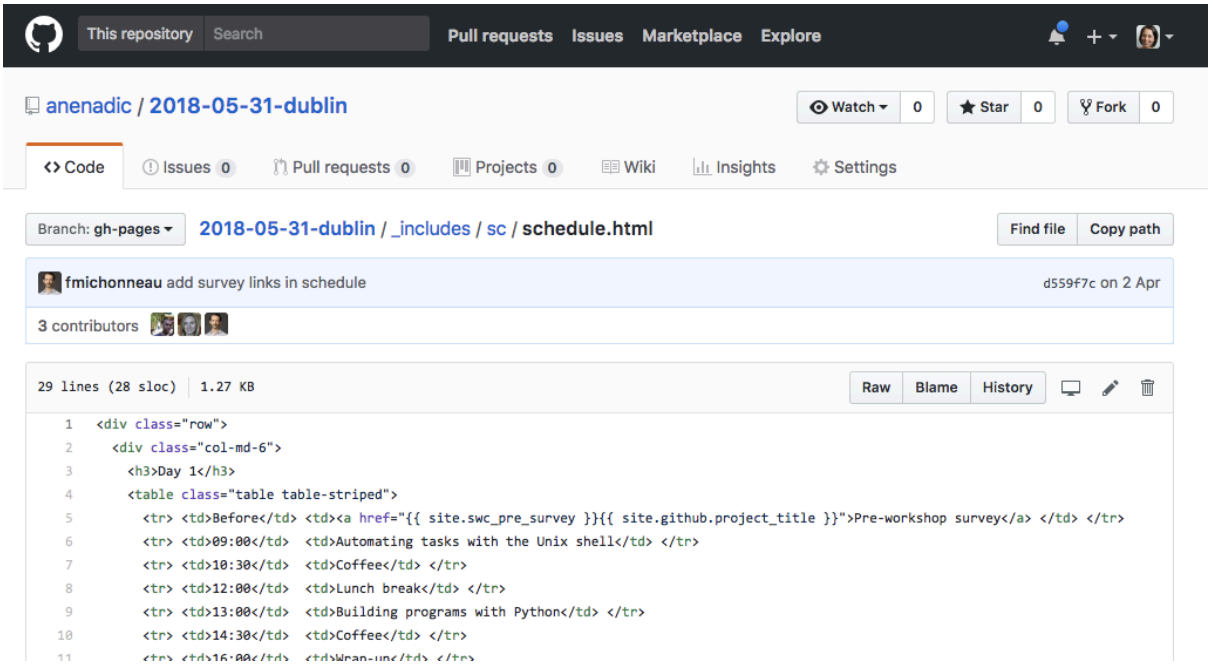


Title should be changed to your workshop title. Be sure to leave the quotes in.

Repository should be: `<your GitHub username>/<the name of your workshop repository>`. Note that no quotes are required here but keep the slash ('/') in.

3a.3 Tweak schedule in _includes/sc/schedule.html

You will probably want to tweak the schedule of the workshop to suit yours. Open up the `_includes/sc/schedule.html` file for editing (i.e. file `schedule.html` in folder `_includes/sc` off the root of the repository). Note that you'll have to navigate a few folders to find it.



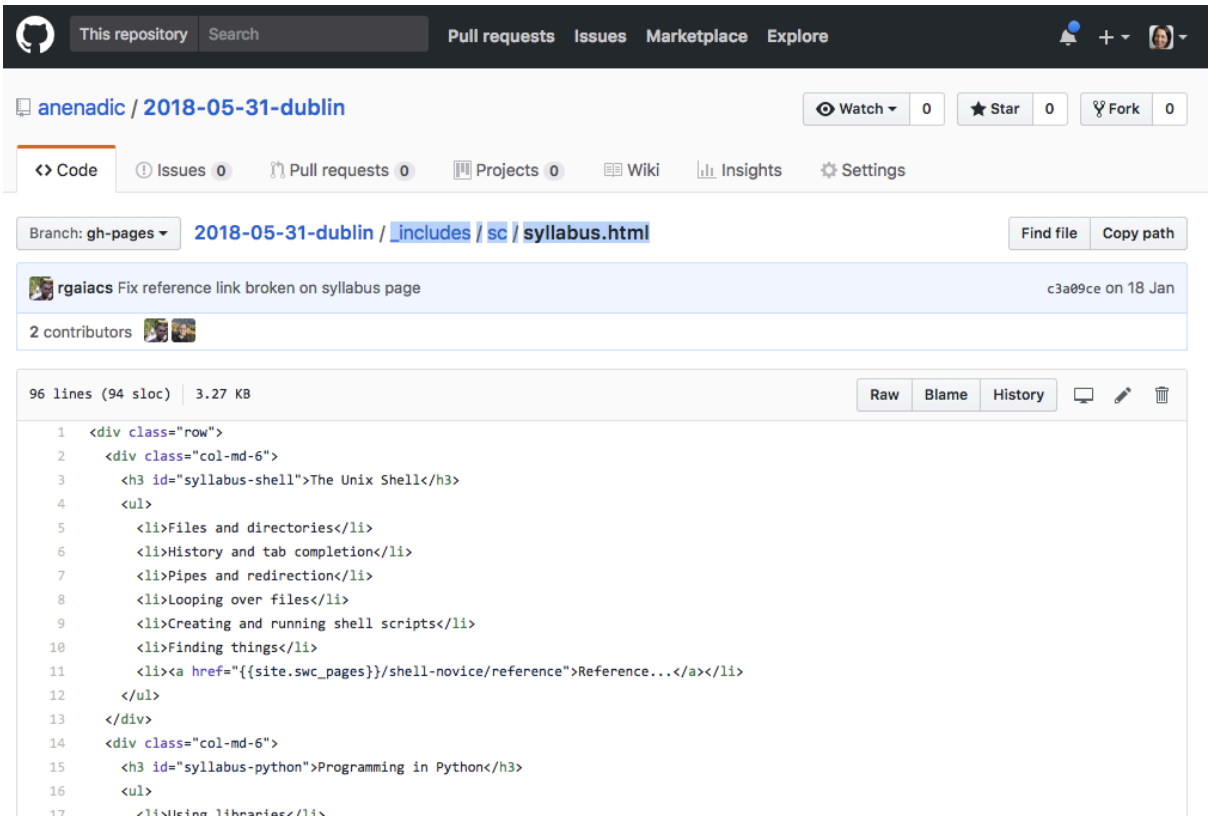
Note that if you are creating a SWC workshop website, your file will be in `includes/sc` folder. If you are creating a DC workshop website, your file will be in `includes/dc` folder. Instructor training and Library Carpentry will also have different folders.

The contents of this page will look different because this is an HTML page and not a Markdown or Yaml file. You should be able to change the times and labels for each thing as needed. As this is HTML, you won't need to worry about quotes or anything. Just leave all the HTML tags in place.

Follow your previous commit steps to save your changes.

3a.4 Tweak syllabus in _includes/sc/syllabus.html

Next open up `_includes/sc/syllabus.html`, which should be in the same folder you were just working in for the schedule. This is where you can change the list of tasks that you'll be doing for each module of the workshop. You may be getting in deep with the HTML here, so consider asking one of your co-instructors for help with this part if you are feeling overwhelmed.



Note that if you are creating a SWC workshop website, your file will be in `includes/sc` folder. If you are creating a DC workshop website, your file will be in `includes/dc` folder. Instructor training and Library Carpentry will also have different folders.

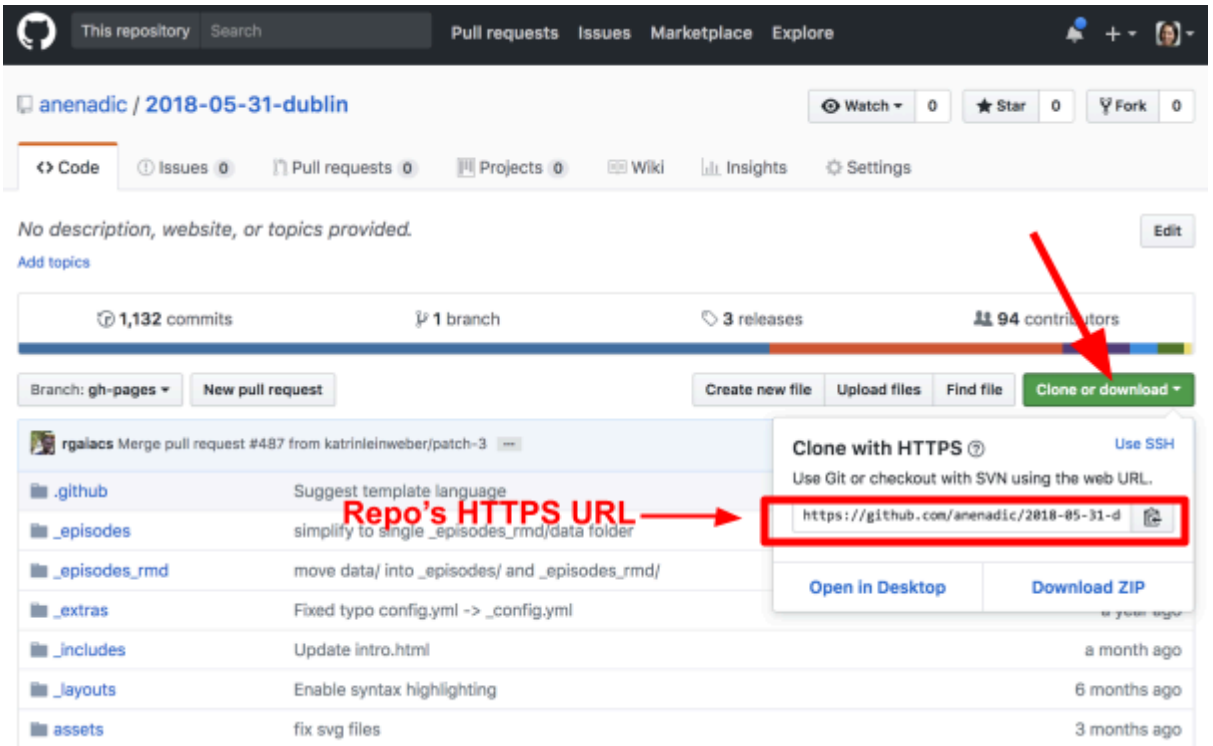
You should be able to copy/paste the div (class = “col-md-6”) chunks around if you need to change the order. However, the order shouldn’t matter much here so long as the content is correct.

That’s it - you should not need to do any more tweaks to the workshop website, unless you want to! Make sure it renders correctly (it may take GitHub a few seconds to compile your files so the changes may not appear immediately).

Step 3b: Optional - customise your website from command line

To create local copy of your workshop repository on your PC (called “cloning”) using the command line, you’ll first need to find out its URL. We will use the HTTPS method to clone, rather than SSH for the sake of simplicity (SSH method requires you to have a private/public key pair). When you use git to work with a remote repository using HTTPS URLs on the command line, you'll be asked for your GitHub username and password and no keys are necessary.

To find your repository’s HTTPS URL, click the green “Clone or download” button on the right-hand side just above the listing of the files of your repository. In the window that pops-up, you will find both HTTP and SSH URLs for your repository. Select and copy the HTTPS URL (which is the default).



From the command line on your PC, do:

```
$git clone <your_repo_url>
```

The clone command will create a copy of the remote repository in a directory with the same name as the repository. Change your location into that directory.

```
$cd <your_repo_name>
```

First, make sure you are on `gh-pages` branch:

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

nothing to commit, working tree clean
$
```

Use your favorite text editor to do changes as described in steps 3a.1 - 3a.4.

If you want to preview your changes before committing them to make sure all is in order, you will need a tool called Jekyll to compile your website in the same way that GitHub does (GitHub's gh-pages use jekyll underneath). Installation guide for Jekyll can be found online, but is out of the scope of this workshop.

The following command issued from the root of your local repository will compile your website and make it accessible from your browser at http://localhost:4000

```
$jekyll serve
```

If you are happy with your changes, stage and commit them and then push to GitHub. You can do that for all the modified files at once as:

```
$git add index.md _config.yml _includes/sc/syllabus.html
_includes/sc/schedule.html

$git commit -m "Configured details, schedule and syllabus of the
workshop"

$git push origin gh-pages
```

As usual, after a couple of seconds needed by GitHub to compile your changes, you can check your workshop website at: https://<your-github-name>.github.io/<your-workshop-slug>/

Step 4: Errors and troubleshooting

You may get into difficulties while setting up your workshop website. Errors when working with git are out of the scope of this workshop - you will have to search for help online. There are also numerous guides online to help you get started with git, for example Git Basics.

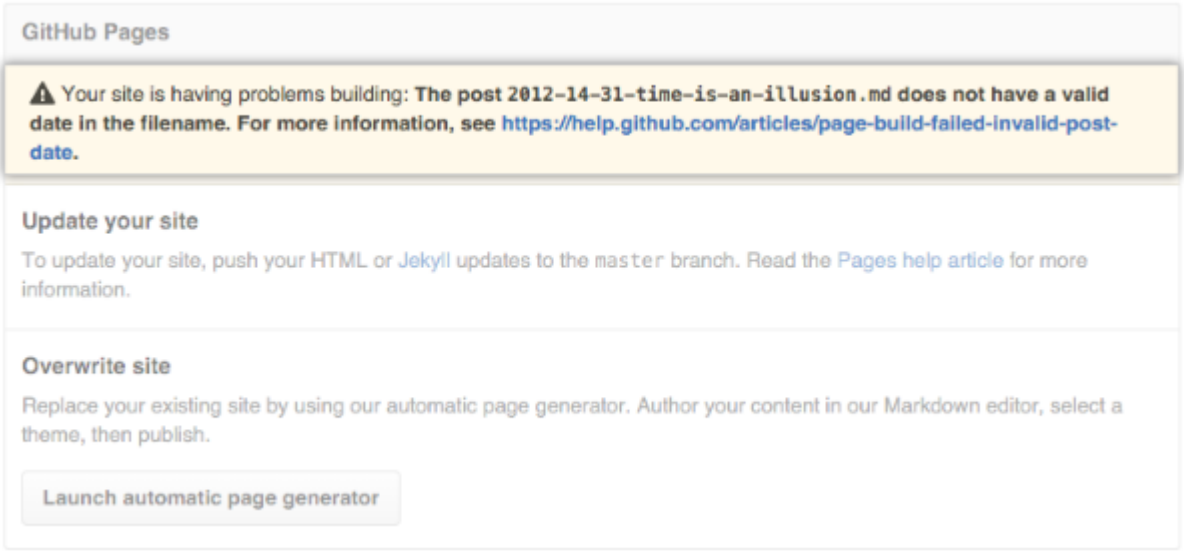
Other typical difficulties you may experience is when your website does not build correctly. This will mainly be due to:

- Jekyll build errors
- Config errors
- Markdown errors
- Liquid errors (Liquid is a scripting language used by Jekyll and its tags are delimited with “{%” and “%}”

This all may sound very complicated but in practice most errors involve a tag not being properly terminated or closed or a syntax error in a config file.

If your website fails to build for any reason, you can troubleshoot your build error by reviewing common problems or specific error messages at: https://help.github.com/articles/using-jekyll-as-a-static-site-generator-with-github-pages/.

If you are making modifications directly in GitHub, you will receive Jekyll build error messages by email or you can view them in your repository under Settings under “GitHub Pages” section.



You will normally know that you have a build error if you make some changes and after a few minutes you cannot see them on your workshop website (i.e. the site remains the same as before the changes). For more serious build errors, it may not even be possible to see the previous version of the site. That is when you know you need to go check your email for error messages from GitHub or go into Settings to see what went wrong.

If you are using command line, Jekyll server you use to serve your website (using command `jekyll serve`) will spit out any build errors in the command line for you and you will be able to spot them immediately.