Camosun College

ICS 211 - Web Applications

Lab 4 - Generating a Website Using the Hugo SSG

Due Date: DEMO DUE BY SECOND DEMO CHECKPOINT, BY END OF YOUR LAB, WEEK OF NOV. 11

Background and Theory (READ THIS FIRST!)

YAML

YAML stands for YAML Ain’t Markup Language. YAML is mainly used for configuration data. Hugo uses YAML for its metadata (metadata is data about data). Docker Compose also uses YAML.

YAML is a lightweight language and there really isn’t much to the syntax:

- YAML files end in the `.yml` extension
- YAML is case sensitive
- YAML doesn’t allow the use of tabs; always use spaces (typically 2 spaces for each indent)
- YAML is like Python. **It uses indentation to indicate structure**.
- YAML is similar to JSON (in fact, you can convert between the two). Values in YAML (called Scalars) can be a number, string, boolean, or null
- Strings don't need quotes in YAML
- A key-value pair mapping is done by separating the key from the value with a colon followed by a space:

```
instructor:
  name:
    first: Jason
    last: Cumiskey
  email: cumiskey@camosun.ca
```

- Arrays (called Sequences in YAML) have their values listed by pre-pending them with a dash followed by a space. For example:

  ```
  fruits:
    - apples
    - oranges
    - bananas
  ```

Many new technologies are using YAML instead of XML or plain text for their configuration files. For the gory details on YAML, see http://yaml.org/spec/1.2/spec.html.

Markdown

Markdown is a lightweight markup language. It is a shorthand way to write HTML. It uses plain text for formatting. For example, to make this **bold**, I surrounded it with two asterisks. Markdown can be used for almost any type of documentation including README files, posts in forums, or even for making content for a website (what you will be doing in this lab!). *This lab itself was written in Markdown!* Markdown is typically converted into **`HTML`**. It can also be converted into **`epub`**, **`mobi`**, or **`pdf`**. The most popular implementation of Markdown is *GitHub Flavored Markdown (GFM)*. A good starting tutorial for GFM is here. In D2L, under Reference Sheets for this course, is a PDF Cheat Sheet for GFM.

Static Site Generators (SSGs)

SSGs take content that you create and generate a website from it. You do this ahead of time; unlike CMSs, pages are not generated dynamically. Like CMSs, you can pick a theme for your site. SSGs also have plugins and can use 3rd party services like a CMS. Jamstack contains a list of several common SSGs. The one that you will use in this lab is called Hugo. Hugo is written in a language called Go. Go is a compiled language with syntax similar to C. It means that Hugo can generate a static site very quickly! *You don't need to know Go to use Hugo*.

Themes

All CMSs and SSGs come with Themes (in fact, many of the *same* themes). Themes are the *visual blueprint* for your site. Themes dictate how your content will be laid out, site colors, site fonts, and how menus, images, and other content will be integrated into your site. The advantage of using a Theme is that all this work is done for you so you can generate an attractive site quickly. A downside to using Themes, however, is that you are getting a *cookie cutter* layout, dictated by the theme author. If you want to customize some features of the theme, it will require you to have knowledge of HTML, CSS, JavaScript, and potentially PHP if its a WordPress theme. Their exists companies whose sole purpose is customizing themes!!! Another downside of using Themes is your site might look nearly identical to someone else's who is using the exact same Theme!

Hugo has a huge number of pre-made Themes. You can find themes for Hugo at https://themes.gohugo.io/. Most themes are designed for blogging sites, the best use case for Hugo. But others have been designed for specific purposes. If you look at the Tags on the right-hand side, there are Company, Ecommerce, and Documentation Themes. For example, Restaurant Hugo is a Theme specifically designed for Restaurants. If you look at the demo, it is a professional, well-designed, responsive web app! All you have to do is change the content. All the HTML, CSS, and JavaScript have been written for you! And it didn't cost you a dime.

For the purposes of this lab, a Theme wil be chosen for you.

Tasks

Task 1 - Initializing Your Site

1. Hugo is installed on your lab assigned VM. Open VSC and using the Remote-SSH connection, connect to your VM.
2. Open a Terminal in the bottom panel and navigate to your *ics211-labs/* directory:

   ```
   cd ics211-labs
   ```

3. You are going to initialize your Hugo project. Run the following command:

   ```
   hugo new site ics211-lab4 --format yaml
   ```

   The `--format yaml` switch tells Hugo to use YAML for its configuration files. By default, Hugo uses TOML for its configuration files. TOML is not as common as YAML. When the command finishes, you should see output like the following:

   ```
   Congratulations! Your new Hugo site is created in /home/student/ics211-labs/ics211-lab4.

   Just a few more steps...

   1. Change the current directory to /home/student/ics211-labs/ics211-lab4.
   2. Create or install a theme:
      - Create a new theme with the command "hugo new theme <THEMENAME>"
      - Install a theme from https://themes.gohugo.io/
   3. Edit hugo.yaml, setting the "theme" property to the theme name.
   4. Create new content with the command "hugo new content <SECTIONNAME>/<FILENAME>.<FORMAT>".
   5. Start the embedded web server with the command "hugo server --buildDrafts".

   See documentation at https://gohugo.io/.
   ```

   Follow the next steps in the lab instead of the ones given by hugo above.

4. Change into the directory it created (*ics211-lab4/*) and type `ll` (two lowercase Ls) for a directory listing. You should see the following files and directories:

```
. (parent project folder)
|-- archetypes/
|-- assets/
|-- content/
|-- data/
|-- hugo.yaml
|-- i18n/
|-- layouts/
|-- static/
|-- themes/
```

Each are explained below (you can also look here):

- **archetypes/** Hugo uses files in this directory as templates for new content (*content* files below). Currently, this folder will contain only one archetype, *default.md*. Currently, this file is using TOML, not YAML. You will fix this in Task 5.
- **assets/** this is a directory that contains global resources typically passed through an asset pipeline. Resources in this directory is typically images, CSS, and JavaScript code. You won't be using this directory in this lab.
- **content/** this directory is where you put your site's content.
- **data/** this directory is used to store configuration files that can be used by Hugo when generating your website. You won't be using this for this lab.
- **hugo.yaml** is the site configuration file. Many site settings are configured here. You will modify this file in this lab.
- **i18n/** which stands for Internationalization and localization, is a directory that contains translation tables for multilingual sites. You won't be using this directory in this lab.
- **layouts/** this directory stores files that can be used to customize specific content pages (i.e. it overrides the chosen theme). You won't be doing this, so you can ignore this directory for this lab.
- **static/** this directory stores all the static content: images, CSS, JavaScript, etc. When Hugo builds your site, all assets inside your static directory are copied over as-is. You will be adding an image to this directory in this lab.
- **themes/** contains the theme used for your site. You will install a theme in the next Task of this lab.

In addition, when you run the `hugo` command to build your site, a **public/** directory will be created that contains all the files necessary for publishing your website.

Task 2 - Adding a Theme

Before you can test and see your site in the browser, you need to add a Theme. Normally, you would either design your own Theme or choose a pre-made one (pre-made Themes for Hugo are at https://themes.gohugo.io/). **For the purposes of this lab, you're going to install a Theme called Mainroad**. This theme is based off of the MH Magazine Lite theme for Wordpress. It's fully responsive and has been designed to be interfaced with Disqus, a commenting service. **NOTE:** The reason you can't pick your own theme for this lab is that some themes can be tricky to install. Some have additional dependencies, some require custom installations, and some are outdated and won't work correctly.

1. **Navigate your shell to your Hugo project's *themes* sub-directory**. Issue the following command:

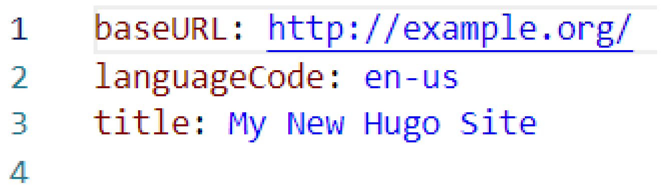
   ```
   git clone https://github.com/Vimux/Mainroad.git
   ```

2. Once it is done, take a directory listing of your *themes* directory:

   ```
   ls -la or ll
   ```

   You should see a new directory called *Mainroad*.

3. Now that the Theme is installed, you need to activate it for your site. Your site's configurations is stored in a YAML file in the root folder of the project. **Open this file (called *hugo.yml*) in VSC**. Your *hugo.yml* file should have three key-value pairs as shown below:

   ```
   baseURL: http://example.org/
   languageCode: en-us
   title: My New Hugo Site
   ```

4. Add the following key-value pair (called a *mapping* in YAML) to this file:

   ```
   theme: Mainroad
   ```

   And save the file. That will activate the theme for your site.

Task 3 - Running the Hugo Development Server

Hugo comes with a built-in Development Server. This Server does three things:

- will compile (generate HTML) your code automatically.
- will start a Node.js Web Server to *serve your site at port 1313*.
- will watch your code for changes and automatically re-compile your site.

1. In the Terminal in VSC, in the root directory (*ics211-lab5/*) of your project , issue the following command:

`hugo server --bind your VM's IP Address`

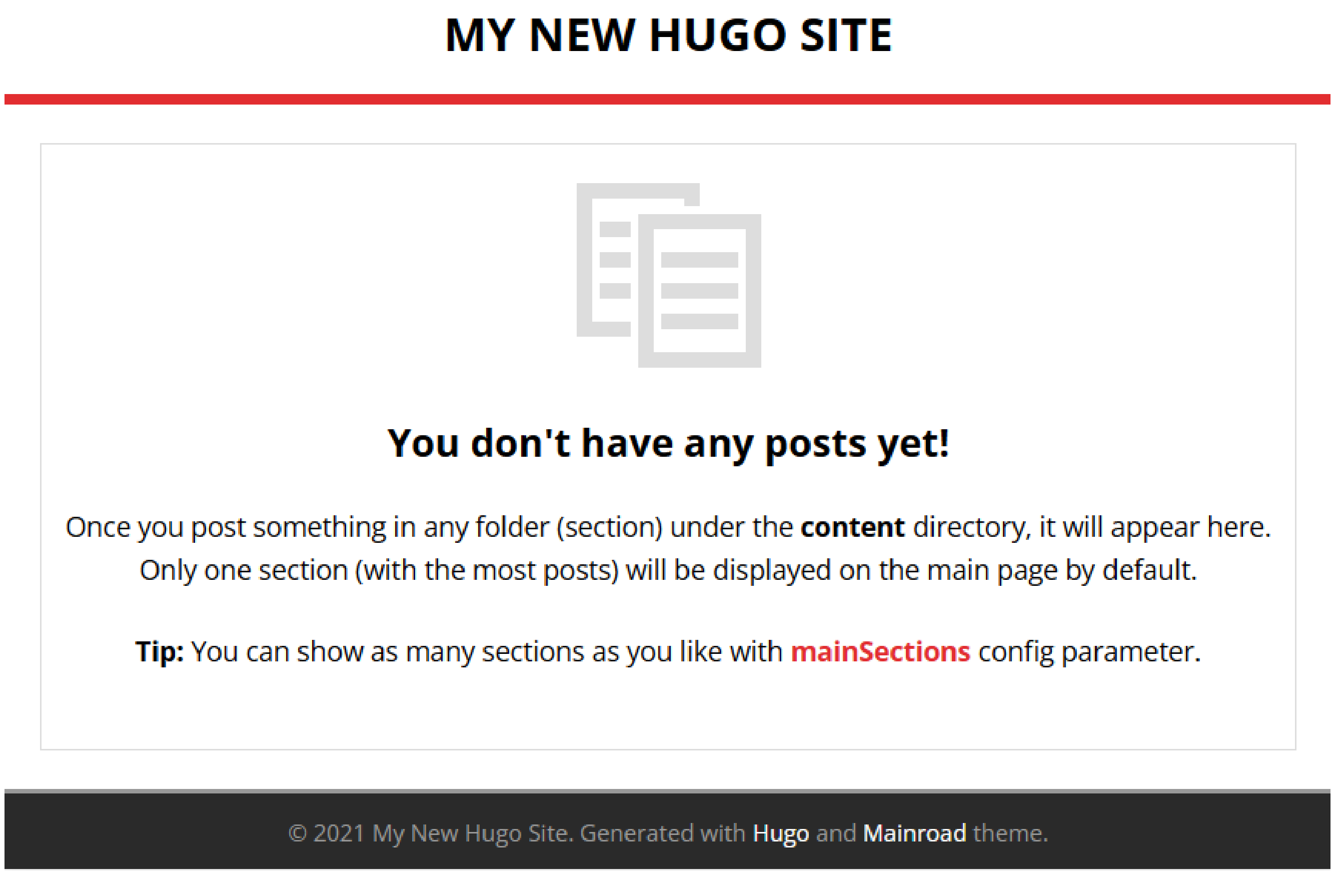
The **`--bind`** option ensures that it binds to the public IP interface and not localhost.

You should see output like the following (the IP address at the bottom will be your VM):

```
Watching for changes in /home/student/ics211-labs/ics211-
lab4/{archetypes,assets,content,data,i18n,layouts,static,themes}
Watching for config changes in /home/student/ics211-labs/ics211-
lab4/hugo.yaml
Start building sites …
hugo v0.118.2-b665f1e8f16bf043b9d3c087a60866159d71b48d+extended
linux/amd64 BuildDate=2023-08-31T11:23:51Z VendorInfo=snap:0.118.2

                  | EN
-------------------+-----
  Pages            |  7
  Paginator pages  |  0
  Non-page files   |  0
  Static files     |  5
  Processed images |  0
  Aliases          |  3
  Sitemaps         |  1
  Cleaned          |  0

Built in 77 ms

Environment: "development"
Serving pages from memory
Running in Fast Render Mode. For full rebuilds on change: hugo server --
disableFastRender
Web Server is available at http://localhost:1313/ (bind address
10.21.75.120)
Press Ctrl+C to stop
```

This shows you any warnings, what files it processed, how long it took to build your site, and that it will monitor your site for any changes. It also shows you that the site is running at **`localhost:1313`** and hitting Ctrl+C stops the Server.

2. VSC might show a pop-up that if you click on it, will open the browser and load your site BUT it will try to load it from `localhost`. **`localhost` on your local machine is not the same as `localhost` on the VM!** Instead, enter **`http://Your VM's IP Address:1313/`** in the URL bar of your browser to load your site. You should see something like the following in your brower's window:


MY NEW HUGO SITE

You don't have any posts yet!

Once you post something in any folder (section) under the **content** directory, it will appear here. Only one section (with the most posts) will be displayed on the main page by default.

Tip: You can show as many sections as you like with mainSections config parameter.

© 2021 My New Hugo Site. Generated with Hugo and Mainroad theme.

3. This theme is quite basic but still looks good! And you didn't have to write any CSS, HTML, or JavaScript! Try changing the size of the browser window and notice how the layout of the site changes to accommodate, as best as possible, the new window size. This is what a responsive website is all about! It arranges the contents of the site so it is presented in the best way possible for different screen sizes. **Leave the Hugo Development Server running and you will see how easy it is to make changes to your site in the next Task**.

Task 4 - Customizing Your Hugo Website

1. Right now, your site is very generic! Start by changing the title. In VSC, edit your project's *hugo.yml* file. **Change the text for `title`**. If you were planning to make it a blogging site, you might want some sort of catchy name. Or for a portfolio site, perhaps just your full name.

2. Hugo's Development Server has *Live Reload capability*. What that means is that it will automatically re-build your site AND refresh the browser for when it detects changes. For example, you should see the change you made in the last step reflected in the browser (after you save the file).

3. Add the following to your *hugo.yml* file. **REMEMBER, YAML USES INDENTATION TO INDICATE STRUCTURE, LIKE PYTHON. MAKE SURE YOU MATCH THE INDENTATION EXACTLY**:

   ```
   Params:
     subtitle: ICS 211 Lab 4 Hugo Site
   ```

   `Params` is an object that is used to customize the theme. In this case, you're adding a **`subtitle`** (it's a property of **`Params`**).

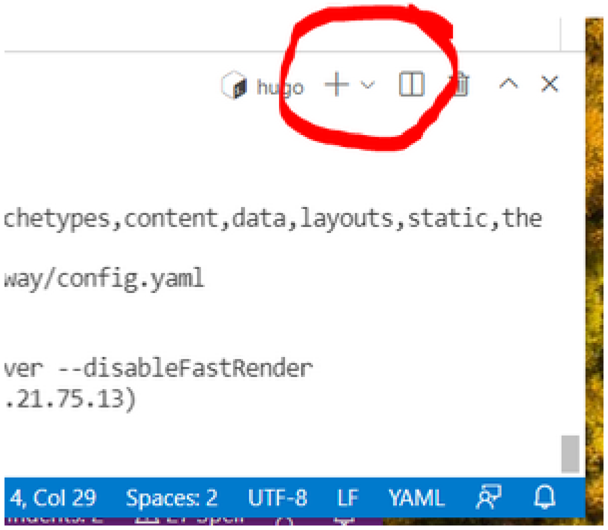
4. Change the copyright text. Add a new key called **`copyright`** under **`Params`** (remember to indent to indicate it is a property of **`Params`**). Put your name as the value. You should see the updated copyright text on the left side of the black bar at the bottom of the webpage.

Task 5 - Adding Content: Adding a Blog Post

1. Before you add a blog post, you need to modify the contents of *default.md* so that it uses YAML instead of TOML. This file is used as a template for new content. **Open this file in VSC now**.

2. There's only a couple of changes you need to make to this file to convert it into YAML. First, change the three pluses (**+++**) into three dashes (**`---`**). Next, change the equals (**=**) to colons (**`:`**). That's it! Save and close this file. You won't need to modify it again. **NOTE**: Anything inside the opening and closing double curly braces (**`{{ }}`**), called moustaches, denotes the GO Templating Language. It allows you to access variables provided by Hugo.

3. Adding new content to your site is pretty easy. Open a *new* Bash Shell (so as to keep the Development Server running) in the Terminal in VSC by hitting the '+' or the split window icon in the Bottom Panel:

   ```
   chetypes,content,data,layouts,static,the
   way/config.yaml
   ver --disableFastRender
   .21.75.13)
   ```

4. Now type the following command in the root directory (*ics211-lab5/*) of your project:

   `hugo new content posts/my-first-post.md`

5. After creating the post, a new directory *posts* under *content* will be created and a new file `my-first-post.md` will be in this new *posts* directory. You'll notice that the file has a `.md` extension indicating it's a markdown file! Open this file in VSC.

6. At the beginning of this file is what is known as *front matter* (the front matter is generated based on the `default.md` file you modified earlier that is in the *archetypes* directory). It's the text between the opening and closing three dashes `---`. The front matter contains metadata (data about data) about the page. The title and date are already filled in for you. The `draft: true` mapping indicates that this file is currently *in draft*. Hugo will not build and publish *draft* content. This allows you to work on the content without worrying that it will get published accidentally. You can add some tags to your post. Add the following to the front matter (before the closing `---`):

   ```
     tags:
       - learning
       - hugo
       - ics 211
       - camosun
   ```

   This is called a sequence in YAML. Sequences are akin to arrays in programming languages.

7. **After** the closing three dashes (`---`), you write your content, in markdown. **Write some now!** Refer to the *Markdown section* in the *Background and Theory section* of this lab for some useful links on Markdown. Note that if you can't think of anything to write and just want to fill in some text to see what it looks like, there's a useful tool for that. It's called the **Lorem Ipsum Generator** (Lorem Ipsum means placeholder text). You can find it at https://loremipsum.io/. You select the number of paragraphs you want then copy the text. Keep a note of this tool, it can come in handy for testing websites with content! **Make sure you have added a heading before your text**.

8. Once you have finished the post, **change the `draft` from `true` to `false` in the front matter**. Once you save the file (or if you have enabled auto-save in VSC), Hugo will no longer ignore it and you should see the changes you made reflected in the browser. The only thing that doesn't appear in the browser is the tag list.

9. To enable the tag list to appear, open the *hugo.yml* file. Add the following under `Params` (remember to indent `sidebar` to indicate it is a property of `Params`):

   ```
   sidebar:
     home: right
     list: left
     widgets:
       - taglist
   ```

   Match the indentation! Remember, YAML is like Python and uses indentation to indicate structure.

You can think of **`Params`** as an object. It has a **`sidebar`** property that itself is an object. **`sidebar`** has **`home`**, **`list`**, **`single`**, and **`widgets`** as properties. The first three have strings as values while the last property, **`widgets`** is a YAML sequence. The configuration here puts the tag list on the right side and then moves it to the left if you click on one of the tags (a list of tags page). You can experiment with this in the browser. *To move the tag list back to the right side, click on the title of your site in the browser*.

10. You can enable some more features of this particular theme. For example, a recent posts list. Add a **`- recent`** value to your **`widgets`** sequence:

    ```
    sidebar:
      home: right
      list: left
      widgets:
        - taglist
        - recent
    ```

    And you should see a "Recent Posts" section show up on your site in the browser!

Task 6 - Adding Content: Adding Static Assets

Task 6A - Adding an Image

1. Suppose you want to post an inline image to your site. On your VM, there's an image in the *ics211-lab-files* directory. If you run the following command in your *home* directory of your VM, it will copy it over to the *static* sub-directory of your Hugo Project:

   `cp ics211-lab-files/johnny-castaway.png ics211-labs/ics211-lab4/static/`

   Alternatively, you can download your own PNG or JPG image on your local machine and upload it to your VM in the same *static* sub-directory using the **`scp`** command:

   `scp name-of-your-image.png student@10.21.74.X:~/ics211-labs/ics211-lab4/static/`

2. Once your image is in the static directory, you can include it in a post very easily! Open your project's *my-first-post.md* file (under the *content/* sub-directory) in VSC. To add an inline image in Markdown you enter an exclamation mark (!), followed by alternate text for the image in square brackets, and then a forward slash followed by the filename in round brackets. For example:

   `![Johnny Castaway](/johnny-castaway.png)`

   The exclamation mark indicates it's a local file that you want added inline with the text.

   NOTE: REMEMBER THAT THE HOME PAGE OF THE SITE SHOWS THE POST'S TITLE, NOT CONTENT. YOU NEED TO CLICK ON THE POST'S TITLE TO SEE ITS CONTENT AND THUS THE IMAGE YOU ADDED.

Task 6B - Adding a link to a PDF file

1. Suppose you want to post a PDF of your Resume. Create a new post and call the page *my-resume* (refer to Task 5 if you don't remember how to create a New Post). **Add at least three tags of your choice inside the front matter of this new post**.
2. Below the Front Matter, place some text indicating that below it, is a link to your Resume. **Make it bold, italic, or a heading (using Markdown)**.
3. Now you will need to upload a PDF on your local machine to the *static/* sub-folder of your Hugo Project (use the **`scp`** command). Either use a real PDF of your Resume, or any other PDF (for example, the PDF of this lab).
4. To post a link, do the same thing as Task 6A, step 2 **EXCEPT** don't put an exclamation mark in the front. This way, it will appear as a link instead. Clicking on the link should open your PDF in the browser!
5. Once you're done, **don't forget to change the `draft` in the front matter from `true` to `false` to see the results in the browser**.

Task 7 - Adding a Third Party Service: Disqus Commenting System

SSGs are very good for blog, documentation, and portfolio sites (sites that display static text, images, and other assets and don't require any input from Users). **But** what if you want to add some interactivity to your SSG built site? Does it mean you have to re-design your whole site using a different technology stack? Fortunately, no. There are many third-party services that you can integrate with your SSG built site. In fact, you can build an entire interactive, e-commerce SSG Web App by integrating these services!

1. You're going to add a commenting system to your site. Instead of building your own, you'll use a third-party service called Disqus. Disqus is a blog commenting hosting service. Go to https://disqus.com/ and **click on the *Get Started* button located at the upper right of its landing page**.
2. You need to sign up to Disqus. It's possible you may already have an account if you have ever visited a site that uses Disqus. If you don't, or you don't want to use a personal account, make an account here. You can use your ICS assigned email address. You will have to agree to their Terms of Service.
3. Once you sign in, you will be presented with a page with two large white boxes. **Click on the one that says *I want to install Disqus on my site***.
4. Next, will be a page that shows your name as the site owner and it asks you your Website Name and Category. You can use any name you want but make sure you **use only lowercase letters and no spaces since it becomes part of a URL**. It also has to be unique so don't use anything too generic. If it's being used by someone else, Disqus will append some random number and letters to it (you will see this in the URL shown below the textbox). Whatever you name it, **REMEMBER IT**. This is called your **Disqus shortname**.

5. Pick a Category. What will you be blogging about? Then click on the *Create Site* button.

6. The next page is the *Select a Plan* page. **Scroll to the bottom, below the three premium plans, there will be one called *Basic* (the free one)**. Hit the *Subscribe Now* button.

7. The next page is the *Select Platform* page. It's asking what platform you're using. You will recognize a few - Wordpess, Squarespace, and Shopify are some that are listed. **Hugo is not shown here so scroll to the bottom and select the rounded button that says *I don't see my platform listed, install manually with Universal Code***.

8. The next page has a video, some code, and other information for manually adding Disqus to your site. Fortunately, the theme you're using has Disqus integration built-in. All you have to do is add a key-value pair to your *hugo.yml* file:

   ```
   disqusShortname: the shortname you selected
   ```

9. You can close the Disqus page in the browser. The Disqus integration with your Hugo site won't show up with the Development Server. You need to build your site for production (next Task).

Task 8 - Building and Viewing Your SSG Site

SSGs are neither CSR or SSR. **An SSG site is pre-rendered during a build phase**. You run a build command and the HTML will be generated for you. You can then send these files, along with static assets, to a hosting service. When a User visits your SSG built site, no rendering work needs to be done. The User's browser will download the HTML, CSS, and other assets from the Server and load it in the browser. This makes SSG built sites very, very fast!!! Note that once you start adding third-party services, some of the site *will* end up being CSR or SSR. The Disqus functionality that you added in the last task is CSR. Notice how it takes longer to load in the browser than the rest of the page!

1. To initiate the build process for your Hugo site, in a Bash Shell in the *root* folder of your project, run the following command:

   ```
   hugo
   ```

   that's it! Hugo will pre-render your site and put all the files into the *public/* folder of your project.

2. Normally, you would publish the *public/* folder on a hosting service. To host your site on your VM, change into the *public/* sub-folder using the same Bash Shell you ran the `hugo` command:

   ```
   cd public
   ```

3. Now run the a simple Node.js Web Server (this doesn't have hot reloading) called HTTP Server:

   ```
   http-server
   ```

4. Now load it in your browser: `http://Your VM's IP Address:8080`. You should see your published site, including the Disqus Commenting Integration. **Note:** You can even run `hugo` with the `-w` switch. This

will cause it to re-build the site anytime any of the files in the project are changed.

Lab DEMO (see top of lab for due date)

Demo to the lab instructor:
- that you built your site and it is running on port 8080 using **`http-server`** (Task 8)
- that the Mainroad theme is activated (Task 2)
- that you have a custom Title on your site and your name is beside the copyright in the black bottom bar (Task 4)
- that you have a link to a "My First Post" posting on the site's home page and a tag list appears on the right (Task 5)
- clicking on the "My First Post" post shows some text and below it, an image (Task 6A)
- clicking on the "My Resume" post shows a link to a PDF file. Ensure the link works (Task 6B)
- that at the bottom of the "my Resume" post is a Disqus Comment section (Task 7)
- **YOU MUST USE YOUR VM'S IP ADDRESS, NOT `127.0.0.1` TO SEE THIS WORK**