Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

STEP1 Install Hugo on windows

Very clear instructions, and useful tutorials:
https://www.youtube.com/watch?v=G7umPCU-8xc&list=PLLAZ4kZ9dF-pOnyRlyS-liKL5ReHDcj4G3&index=2

STEP2 Create a new HUGO project

1. Open GitBash, go to your directory where you want to create the site folder.
2. Run: *hugo new site "name of your site folder"*
3. HUGO generate new SITE with the DEFAULT folders:
-archetypes (you can generate a MASTER.md that serves as a reference template when you generate new content)
-content
-data (is always empty the default, and normally is not uploaded into Git unless there's something inside the folder)
-layouts (is always empty the default, and normally is not uploaded into Git unless there's something inside the folder)
-static (this is where you will store all the images or other files)
-themes (where you can load a theme from HUGO)
config.toml (most important file)

STEP3 Download and install a HUGO theme

1. I decided I wanted to use this theme:
https://github.com/kishaningithub/hugo-creative-portfolio-theme
2. Instead of clonning the repository, you should add it as a SUBMODULE. See the steps below:
   - Go to your site directory and run the command (this will initialize this folder as a git repo): *git init*
   - Go to the themes git repo and copy the git link (will look something like this)
   https://github.com/kishaningithub/hugo-creative-portfolio-theme.git
   - On gitbash run the command:
   git submodule add https://github.com/kishaningithub/hugo-creative-portfolio-theme.git themes/"name of the theme"
3. This will add as a SUBMODULE the folder that contains the files and templates from the theme you choose
4. Go inside your *themes* folder and check it has been added.
Note I normally call the folder the same name as the original theme

STEP4 Change config file

In order to run the theme you need to modify your config file. It should be the same as the config file from the theme you choose. You could copy and paste it manually or instead:
1. Inside your directory folder run the command:
cp themes/"name of your themes folder"/exampleSite/ . -r*

What you are saying above is: copy the config file that is inside the example site, on this same folder. So it will override the default one and you should now see the config folder updated.

Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

STEP5 Personailze the config file

IDepending on the themes you've selected the config will be different. But basically here you can change and personalize the basic parameters of your site, such as: website title, etc.
Here is an example of how a config file looks like:

```
baseurl = "https://example.org/"
title = "20200217 PORTFOLIO TEST"
theme = "hugo-creative-portfolio-theme"
languageCode = "en-us"
# Enable comments by entering your Disqus shortname
disqusShortname = ""
# Enable Google Analytics by entering your tracking code
googleAnalytics = ""

[params]
  # Style options: default (pink), blue, green, pink, red, sea, violet
  # Use custom.css for your custom styling
style = "default"
description = "Describe your website"
copyright = "&copy;2016 Your name or company"
sidebarAbout = [
 "Intro about myself (TEST).",
 "More info about myself (TEST)."
]
```

STEP6 Preview your site on local server

After the above test if you run inside your root directory the following command, you should be able to see the site running. Command:

```
hugo server
```

STEP7 Hosting your site through GitHub Pages

If you don't have a GitHub account start by creating one.

Create a new repo. It is essential that you call it as follows:
default-github-username/<same as your githug user name>.github.io
for example mine looks like this:
pamelamartello/pamelamartello.github.io

Make it pubic, no need to initialize with a README at this point or license (unless you want).

Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

STEP8 Create a BRANCH

After you create the repo, the first thing we will do is to add a *BRANCH.* So we run the following commands:

```
git init (although we initialized previously)
git branch "develope"  (git branch <name of branch> is to create a new branch)
git remote add origin git@github.com:UserName/RepoName>.github.io
git remote pull origin master (push changes)
```

If when puulling oring master you encounter the following error:

```
Error: Permission denied (publickey)
```

You can find here the documentation about this error:

https://help.github.com/en/github/authenticating-to-github/error-permission-denied-publickey

What I did that worked in my case was changing the remote URLs from SSH to HTTPS:

https://help.github.com/en/github/using-git/changing-a-remotes-url

Documentation from GitHub

1.Open Git Bash.

2.Change the current working directory to your local project.

List your existing remotes in order to get the name of the remote you want to change.

```
$ git remote -v
> origin  git@github.com:USERNAME/REPOSITORY.git (fetch)
> origin  git@github.com:USERNAME/REPOSITORY.git (push)
```

Change your remote's URL from SSH to HTTPS with the git remote set-url command.

```
$ git remote set-url origin https://github.com/USERNAME/REPOSITORY.git
```

Verify that the remote URL has changed.

```
$ git remote -v
# Verify new remote URL
> origin  https://github.com/USERNAME/REPOSITORY.git (fetch)
> origin  https://github.com/USERNAME/REPOSITORY.git (push)
```

After doing remote pull origin, you shuold get a similar message:

```
$ git pull origin master
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/pamelamartello/pamelamartello.github.io
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master
```

Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

The previous step will generate a branch containing your files from the root irectory you are working with. To check what is inside this new foler run:

```
git checkout “name of branch”
```

In my case it showed something like this:

```
git checkout develope
Switched to branch 'develope'
A	.gitmodules
A	themes/hugo-creative-portfolio-theme
```

It means you are inside the branch folder and contains those two files.

After this run

```
git status
```

to check what is missing to upload to the branch

STEP 9Compile files of branch and commit changes

After the steps above you need to commit your changes into the branch, so run the following:

```
git add .
git commit -m "first push to branch"
git push origin develope (you need to push the changes into your branch “develope” )
```

This is an example of the message I get:

```
$ git push origin develope
Enumerating objects: 50, done.
Counting objects: 100% (50/50), done.
Delta compression using up to 8 threads
Compressing objects: 100% (45/45), done.
Writing objects: 100% (49/49), 845.95 KiB | 13.87 MiB/s, done.
Total 49 (delta 10), reused 0 (delta 0)
remote: Resolving deltas: 100% (10/10), done.
remote:
remote: Create a pull request for 'develope' on GitHub by visiting:
remote:     https://github.com/pamelamartello/pamelamartello.github.io/pull/new/develope
remote:
To https://github.com/pamelamartello/pamelamartello.github.io.git
 * [new branch]      develope -> develope
```

STEP10 Compile and run HUGO to create the website

The you need to build the sites running HUGO command:

```
hugo
```

After running HUGO you should see a new PUBLICS folder, run ls to see them:

```
ls
```

I will look something like this:

```
ls
archetypes/  content/  layouts/  README.md  static/
config.toml  data/     public/   resources/  themes/
```

Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

STEP11Create and add the script

Open your tex editor (could be notes or other) copy and paste this script (make sure you change the name to your branch name inside the script on the red area).

```
#!/bin/bash

echo -e "\033[0;32mDeploying updates to GitHub...\033[0m"

# Build the project.
hugo

# Add changes to git.
git add -A

# Commit changes.
msg="rebuilding site `date`"
if [ $# -eq 1 ]
  then msg="$1"
fi
git commit -m "$msg"

# Push source and build repos.
git push origin  "develope"
git subtree split -P public -b published
git push origin published:master --force
git branch -D published
```

Now save it as this:

```
deploy.sh
```

The .sh is important as it means this is a script.

You should save it inside our root directory. (where content, static, themes....)

Run ls to see that the script is inside:

```
ls
archetypes/  content/  deploy.sh*  public/   resources/  themes/
config.toml  data/     layouts/    README.md  static/
```

Now run the script!

```
sh deploy.sh "add commit message"
```

You can add the commit message like when you run git commit, if you don't write a message it will show a default message.

Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

The script message will look something like this:

```
 sh deploy.sh
Deploying updates to GitHub...
Building sites ¡
               | EN
+------------------+----+
 Pages            | 24
 Paginator pages  | 0
 Non-page files   | 0
 Static files     | 51
 Processed images | 0
 Aliases          | 0
 Sitemaps         | 1
 Cleaned          | 0

Total in 106 ms
warning: LF will be replaced by CRLF in public/404.html.
The file will have its original line endings in your working directory (read below what this warning means***)
On branch develope
nothing to commit, working tree clean
Everything up-to-date
Created branch 'published'
d3b8104f76b3c4512008d4b8b397a3e1299d24c7
Everything up-to-date
Deleted branch published (was d3b8104).
sh: __git_ps1: command not found
```

****It's just how the end of line will be saved.*
In Unix systems the end of a line is represented with a line feed (LF). In windows a line is represented with a carriage return (CR) and a line feed (LF) thus (CRLF). when you get code from git that was uploaded from a unix system they will only have an LF.

Creating a website using HUGO, hugo themes an GITHUB Workflow

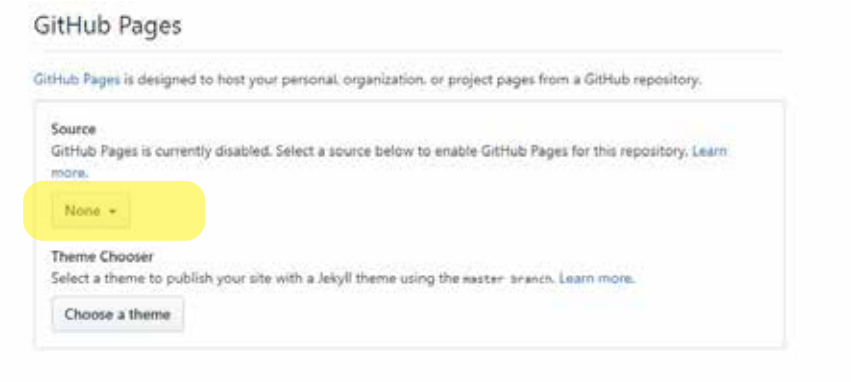
Basic steps

STEP12Site running

At this point your site should be running.
But it didn't work out for me for two reasons, after which it all went fine.

1Disable all other repos GitHubPages

1 If like me, you have several repositories that contain same folders, and all of them are GitHub Pages (website enabled), you will encounter issues when trying to direct to your just newly created HUGO site. What I had to do was to DISABLE all the GitHubPages from all my other repositories. Otherwise there could be some link error to the site, as the repositories name and files are similars.



2 Set your URL on your config file

Go to your config file and update the URL. This is an example of how mine looks like now:

```
baseurl = "https://pamelamartello.github.io/"
title = "20200217 PORTFOLIO TEST"
theme = "hugo-creative-portfolio-theme"
languageCode = "en-us"
# Enable comments by entering your Disqus shortname
disqusShortname = ""
# Enable Google Analytics by entering your tracking code
googleAnalytics = ""
```

STEP13Run script again

After doing the changes above, instead of doing *git add, git commit*, simply run the script:

```
sh deploy.sh "updates about page"
```

Basic steps

After running the script again it will look something like this (same as step 11)

```
sh deploy.sh "updates about page"
Deploying updates to GitHub...
Building sites ¡
               | EN
+-----------------+----+
 Pages           | 24
 Paginator pages  |  0
 Non-page files   |  0
 Static files     | 52
 Processed images |  0
 Aliases          |  0
 Sitemaps         |  1
 Cleaned          |  0

Total in 107 ms
warning: LF will be replaced by CRLF in public/404.html.
The file will have its original line endings in your working directory
[develope 7d43fc3] updates about page
 4 files changed, 6 insertions(+), 6 deletions(-)
 create mode 100644 public/img/test-pic-v2.jpg
 create mode 100644 static/img/test-pic-v2.jpg
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 24.70 KiB | 8.23 MiB/s, done.
Total 11 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/pamelamartello/pamelamartello.github.io.git
   e1ed804..7d43fc3  develope -> develope
Created branch 'published'
11d9e56b2747b32d4c8e881cd94cbf2914a30fda
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 24.33 KiB | 8.11 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/pamelamartello/pamelamartello.github.io.git
   f83a47b..11d9e56  published -> master
Deleted branch published (was 11d9e56).
sh: __git_ps1: command not found
```

Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

STEP13Final steps & continue deployment

At this point whenever I make any changes I just need to open GitBash and run the script.

I was curious why gitbash doesn't mark *(develope)* on my folder I am working on, so what we did was to run the following command to check on what branch are we currently working on:

```
tlacuachita@Pamelita MINGW64 /d/WEBSITE-DEVELOPMENT-MATERIAL/20200217
$ git branch
* develope
  master
sh: __git_ps1: command not found
```

This way you make sure under what branch you are working on.

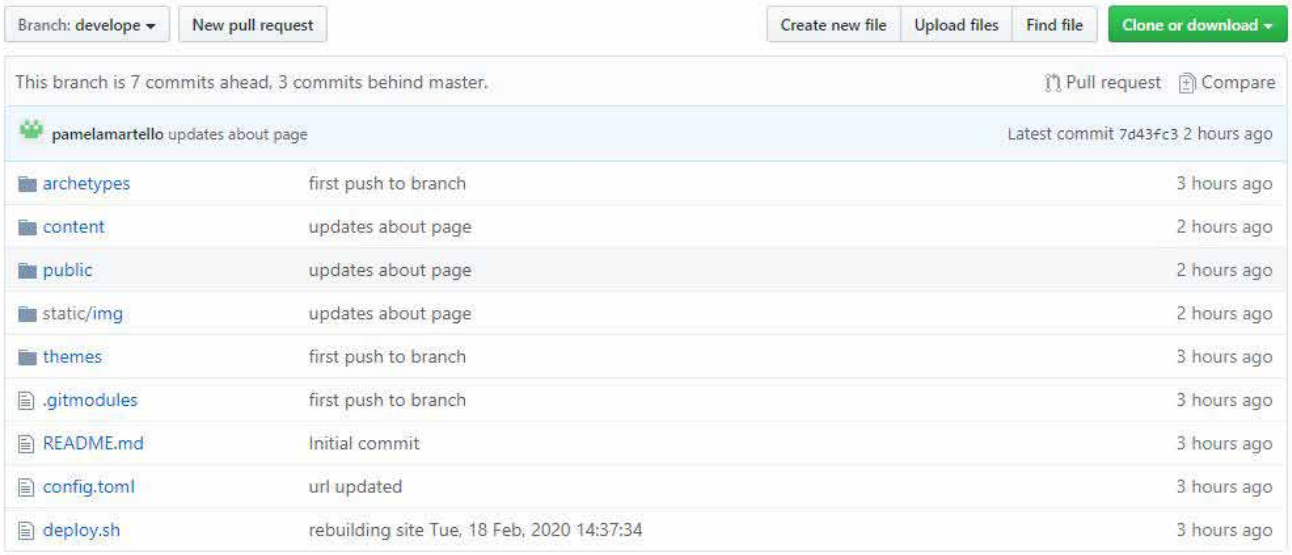
Creating a website using HUGO, hugo themes an GITHUB Workflow

Basic steps

General thoughts

What we did was:

1. Create two branches: master (if you configured github pages to be read from master) and the other one that we are calling *"develope"*
2. Inside develope folder we have the following files:



3. After this branch was created and files were uploaded, we ran the script.
4. This script will dump the result into a folder called public. Then the public folder is pushed to master, using git subtree.

Is sort of tricking GtHub pages to use the master which is being "controlled by the public branch.

And the important things to remember are:
* use a repo named: <username>.github.io
* then this repo MUST use master as the source for github pages
* then i used the script provided before to compile all