**UPLOADING TO GITHUB**

To get our content uploaded to GitHub, work through these steps:

1. We'll start by creating a new repository – for this, browse to <https://github.com>, and click on Sign Up in the top right.
2. Go ahead and follow the instructions provided on screen, including adding an email account (it's worth it!) – make sure you take note of the details you use for your account. Once done, sign in with your new account, and make sure it is validated.
3. Next, we need to create our repository – click on the + sign in the top right, then New repository.
4. On the next screen, the Owner field will you're your account name; enter a repository name in the format **username.github.io**, where username is your chosen username on GitHub.

Normally we can use any name when it comes to creating GitHub repositories – as we’re creating a GitHub Pages-based site, we need to use this format as one of the requirements.

1. Next, choose Public as the repository type, and click on the checkbox to initialize the repository with a readme file.
2. For the Add .gitignore option, choose Node, and set the Add a license to MIT.
3. Click on Create repository. Once done, click on Clone or download, then Open GitHubDesktop.exe to clone your new repository area to your local PC.
4. Click on View the files of your repository in Explorer to display the contents of your repository in Windows Explorer.

We now have our repository in place, ready for us to upload content. The next stage is to copy our site into this area – so that we can upload the content. You might want to read through these steps first, before completing each task first – there are a couple of points that may catch you out:

1. Fire up (or revert to, if already open) GitHub Desktop, then click on File | Options and sign in with your new account details. Once signed in, click on Show in Explorer to view the files stored locally – it's into this folder that we will store content.
2. Next – go ahead and copy all folders into this area, **except for the following**: /public and /node\_modules.

The folder structure should be roughly similar to [https://github.com/  
hexojs/hexo-starter](https://github.com/hexojs/hexo-starter), without the .gitmodules file.

1. Crack open the .gitignore file, and make sure it contains the following entries:

.DS\_Store

Thumbs.db

db.json

\*.log

node\_modules/

public/

.deploy\*/

1. Save the file and close it. Revert back to GitHub Desktop, then enter Initial version in the text field above the Description box (bottom left of the application window).
2. Hit Commit to master, then Push origin to upload your files to your site – you now have your content ready for action!

At this point, you can minimize both GitHub Desktop and your text editor – you may need them again, just in case the build and deployment process doesn't work the first time!

## Testing the code with Travis CI

**Testing code with Travis-CI**

To get Travis-CI added to your GitHub account and set up to run builds, follow these steps – they are a little tricky, so you may want to read through them first, before completing them:

1. Go to <https://travis-ci.org>, and sign in with your GitHub account that you created back in Part 1.
2. Click on Authorize to accept the Authorization of Travis CI to your account. Travis will redirect you to GitHub.
3. Let it redirect, then click on Activate all Repositories using GitHub Accounts | Approve & Install.
4. On a new tab, we need to generate a [new token](https://github.com/settings/tokens) with repo scopes – this is to allow Travis to build a production-ready version of your code. To do this, browse to <https://github.com/settings/tokens> in a new tab, then click on Generate new token.
5. It may prompt you to sign in – use your GitHub password if needed.
6. On the next screen, enter "Building demo hexo site" (no quotes!) into the field, then tick the repo option (which will select all four sub-options).
7. Note down the token value – keep this safe! At the bottom of the page, hit Generate token.
8. Switch back to the previous tab in your browser, and browse to this address: [https://travis-ci.org/github/<username](https://travis-ci.org/github/%3cusername) and site name>/settings, where username is your GitHub account name, and site name is the name you gave your site back in Part 1 of this exercise.
9. Scroll down to the Environment Variables entry, then add GH\_TOKEN as name and paste in the token value from step 7 in this exercise. Click Add to save it.
10. Switch to your text editor, then in a new file, add the following code, saving it as .travis.yml in the GitHub local version of your site (**not** the myblog folder!):

sudo: false

language: node\_js

node\_js:

- 10 # use nodejs v10 LTS

cache: npm

branches:

only:

- master # build master branch only

script:

- hexo generate # generate static files

deploy:

provider: pages

skip-cleanup: true

github-token: $GH\_TOKEN

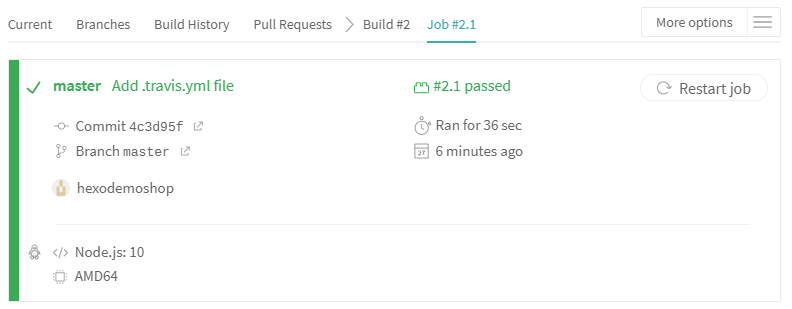
keep-history: true

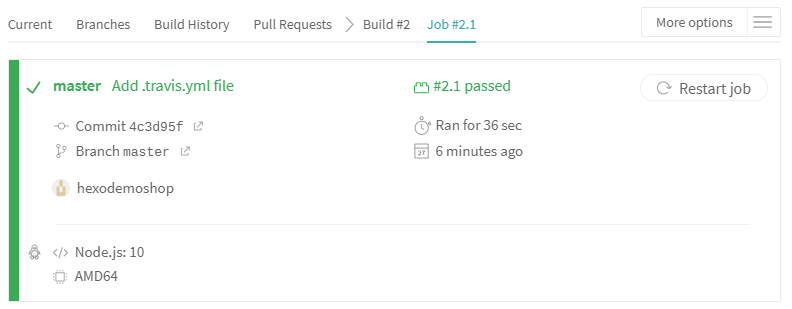
target\_branch: master

on:

branch: master

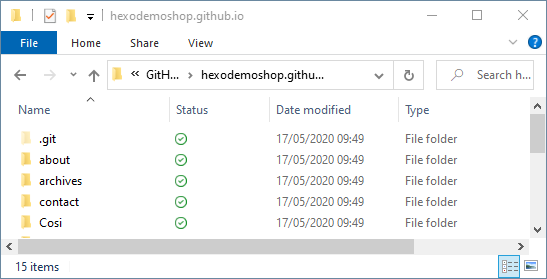
local-dir: public

1. Save the file, and push this up in the same way as we did before with our content files.
2. Switch back to the browser window with Travis-CI still running, and click on Dashboard – if all is well, we should see Travis building our site; we are aiming to have a green tick (
3. Figure 1):



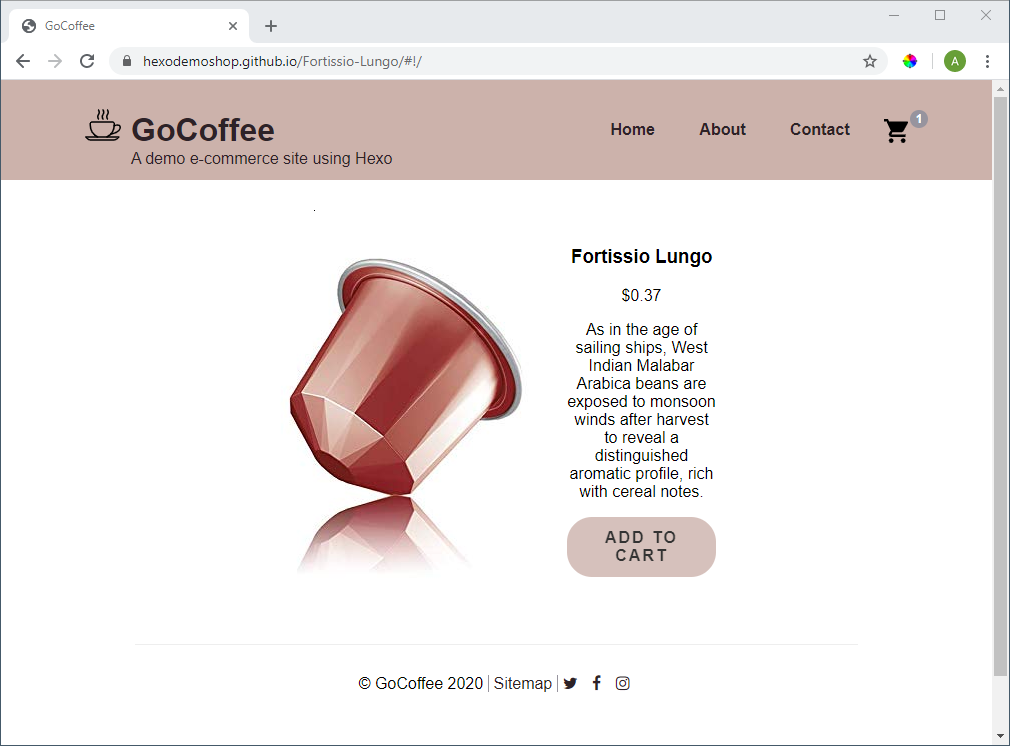
*Figure 1: Travis-CI tests running in our browser*

1. There is one more step to work through – switch back to GitHub Desktop and try pulling down a copy of your site. You will notice it looks different; Figure 2 shows part of what it will look like, now that Travis has built and tested our website for us:



*Figure 2: Our newly built/compiled site, cloned to desktop*

1. Once Travis CI finishes the deployment, GitHub stores the generated pages in the master branch of your repository – go ahead and browse to [https://github.com/<name](https://github.com/%3cname) of your repo to view the finished results.
2. Go ahead and navigate to your site at <https://username.github.io> – you can see a screenshot of the completed deployment, which will look similar to the screenshot shown in Figure 3:



*Figure 3: The completed store, hosted on GitHub Pages*