**Server Setup & Manual Build Instructions**

**Accessing the Server for File Transfer**

1. Download Filezilla here: <https://filezilla-project.org/download.php?show_all=1>

2. Log into UW-Stout VPN

3. Host is sftp://144.13.22.59

4. Username is agileg5

5. Password is aexpKiran

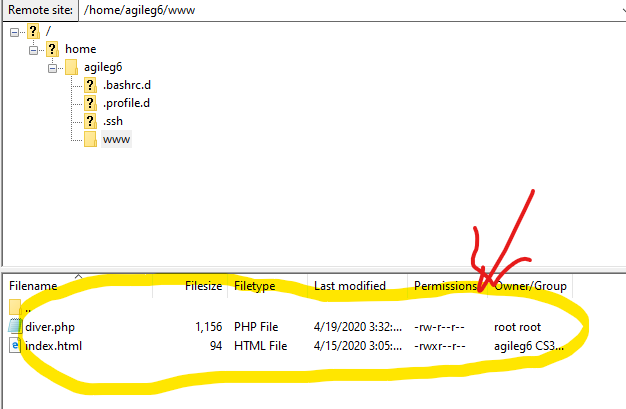
6. Port is 22

**Build Instructions**

1. Click agileg5 folder

2. Click www folder

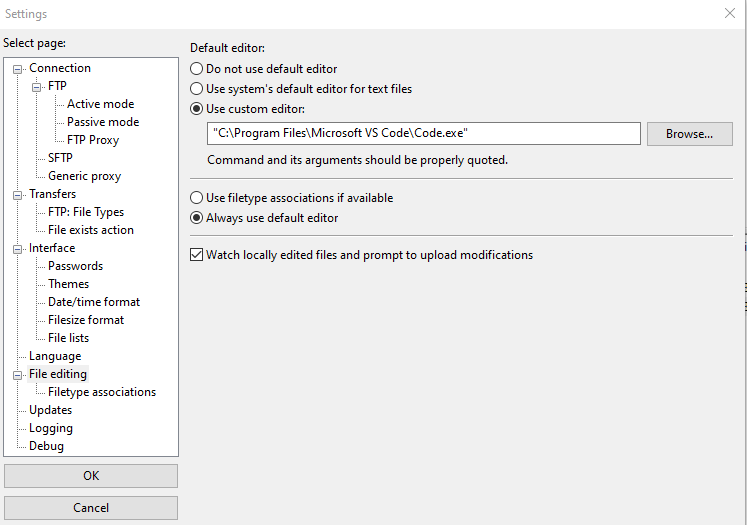
3. Drag and drop the files you want to add (php, html, etc.) from your GitHub repository or workspace into the box with the red arrow.



**Configuring Code Editor to Automatically Modify Files Directly on the Server (Larissa)**

If you would like to avoid dragging dropping your file from your project folder into FileZilla repeatedly, you can set up the editor function using FileZilla:

1. Go to Edit -> Settings
2. Click the File editing option and click the “Filetype Associations”
3. Click “Use Custom Editor:” and toggle “Always use default editor”
4. Find your VS code .exe file



1. Now, you can right click on the file you’d like to edit, and click the “View/Edit” option

-this will open VS code and you can edit it directly on the server.

-after editing, every time you save on VS code, this will open a prompt on FileZilla called “file has changed”

-click “yes”

-you can now refresh your page on he<http://144.13.22.59:85/> and it will show your edits

-you can keep editing, saving, and confirming from here.

**Build Instructions for Continuing Further or a Large Scale Project (Noah)**

Ideally, these processes would have been decided at the beginning of the project. However, given our time constraints and scope, it did not make much sense to implement either of these processes. This content sums up my research on build processes for the project.

--------------------------------------------------------------------------------------------------------------------------------------

Repeatedly moving files in and out of FileZilla with a standalone Github repository can get messy and out of sync quickly. Given that we are utilizing Github and the server is configured with Git, two approaches could be taken if the project were to continue:

1. **A true CI/CD Pipeline** using a tool like Travis CI or Github Actions (https://help.github.com/en/actions) to build and push to the server. A CI/CD pipeline would require the use of feature branches, which are merged back into the master branch. Upon merging, the CI tool would be activated, the code would be built, and then sent off to our server. No other scripts or management would be required, other than tracking if a build fails.
2. Or, more logically, **A script linking the Github repository to the server.** Rather than manually moving files, the script would pull from the GitHub repository master branch and place the files in their proper directories. A simple python script would be placed in the “www” or master directory of the server and would be run through the command line – eliminating manipulation through FileZilla.

A script would be a combination of these two processes:

