SUMMARY

How To ... 1.1

Install Git Locally 1.2

Setup Dev environment with Visual Studio Code 1.3

Create new Github repo from existing local file folder 1.4

Create gh-pages branch for Github hosting 1.5

Use the Visual Studio Code Terminal 1.6

Kill a process based on port number 1.7

Validate HTML and CSS 1.8

Setup SSH 1.9

Setup Linux Shell on Windows 10 (Ubuntu) 1.10

How to Add an Admin User to Wordpress using mySQL 1.11

Wordpress: Additional Permissions 1.12

Codenvy: Get started 1.13

Lets Encrypt: HTTPS on Apache 1.14

Using Bootstrap with Vue.js 1.15

Git Command Line 1.16

Create a Droplet on Digital Ocean 1.17

Migrate Vue 2 Code from Vue CLI 2 to Vue CLI 3 1.18

How to ...

This collection of FAQ's serve to help a student accomplish tasks needed to develop web sites.

This is the answer to "How to install Git locally?"

Before you start this process, create an account on Github at https://github.com/.

Download and Install Git

https://git-scm.com/downloads

This will install a Folder with a number of file and programs.

Open a bash window (`terminal` on Mac or ' `git bash` ' on Windows) and type in

```
git --version
```

This should return a confirmation that git is installed with a version number.

Connect to your local workstation to Github with SSH

Without SSH connection setup you will have to enter your username and password every time you connect to Github, which you will usually have to do during a `git push` command. By sharing your workstation public key with Github, you will not have to provided username and password to connect.

Open a bash session (`terminal` on Mac or ' `git bash` ' on Windows). All commands below should be entered into the bash session.

Check to see if you already have a private key/public key setup by listing any files in the hidden direction .ssh.

```
ls -al ~/.ssh
```

If you don't see any of the following files you will need to create the key files

- id_rsa.pub
- id_ecdsa.pub
- id_ed25519.pub

Create public key/private key files if they don't already exist

You only need to do this step if you don't already have public/private key files.

You ll first create the key files using the email you provided to Github. This will label the keys with that email.

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

```
Generating public/private rsa key pair.
```

Next you'll be prompted to enter a file name for your keys and you can just press enter to accept the default.

```
Enter a file in which to save the key (/c/Users/you/.ssh/id_rsa):[Press enter]
```

Finally, you'll be prompted to enter a passphrase. This is a security feature and you will have to enter this passphrase when you connect to Github, so be sure to remember it. It can be as simple as a four digit pin number.

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
```

```
Enter same passphrase again: [Type passphrase again]
```

Start SSH Agent and Add Keys

The SSH Agent is a process that will allow you to manage SSH Keys. You'll start that process and add your keys with the following command.

```
eval $(ssh-agent -s)
```

You will see a response like this indicate that the process is running.

```
Agent pid 59566
```

The command below adds your key to the ssh agent. If the name of your key is different replace `id_rsa` with the name of your key.

```
ssh-add ~/.ssh/id_rsa
```

Add the SSH Key to you Github Acccount

Now you need to add the public key to your Github account. This will allow Github to decrypt the data you send it via SSH.

We start by using the `clip` command to copy the content of the public key SSH file to your local buffer.

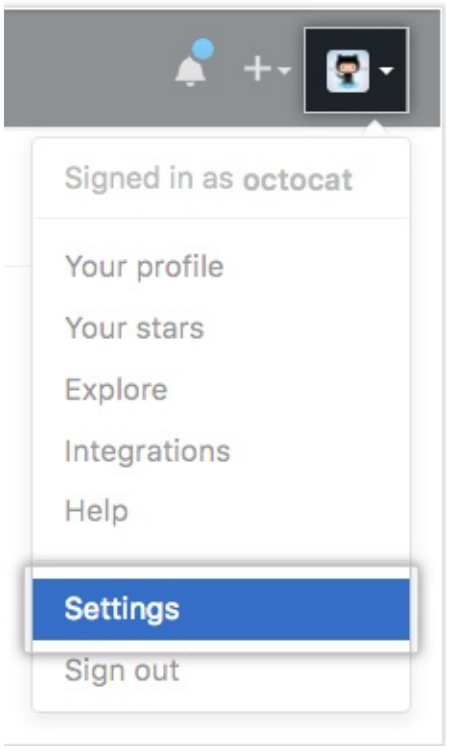
```
clip < ~/.ssh/id_rsa.pub
```

If the `clip` command is not installed you can list the file to the console and then copy the contents into your buffer with ctrl-c/cmd-c. The `cat` command will list the file to the console.

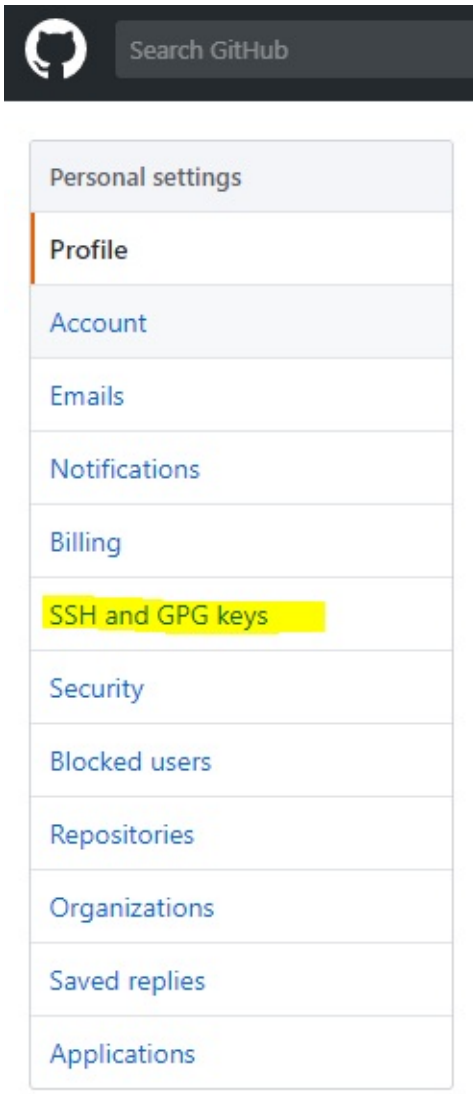
```
cat ~/.ssh/id_rsa.pub
```

You can also open the file with a text editor and copy the content into the buffer if you have trouble with the `clip` command.

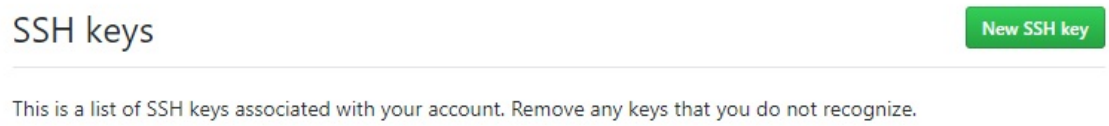
Next, go to your Github account online and click on your photo. This will reveal a dropdown selection where you should select **Settings**.



The **Setings** menu will appear on the left of the screen and from there you will click on SSH and GPG keys.



After you click on SSH and GPG keys you, you will you click on a button to add a new key.



Click on the Green button to add the key you have in you buffer. You should see a form. Enter a Title that help you to remember the workstation you got the key from and then paste the key into the key textarea. Finally click on the Add SSH key to add it to the list of keys known to Github.

Update User Name and Email in Local Git Config

The local Git program maintains configuration data about you that should correspond to your account on github.com. You'll want to provide the email and user name that you used for github.com in when you execute the following commands.

```
git config --global user.name "Mona Lisa"
```

```
git config --global user.email "email@example.com"
```

You can execute these commands to verify that you have set up the config variables properly.

```
git config --global user.name
```

```
git config --global user.email
```

Making VS Code the Default Editor for Git

You can make Visual Studio Code the default editor for git. If you ever forget to close the quotes when adding the message during commit (`git commit -m"update`) you find that git open up the default editor.

Before running the git config to set VS Code as the default editor, check that the application is in your computer's path so that it can be found. You can check this by going to the command line and running.

```
code --help
```

If a list of help topic is printed to the screen, the VS Code is in the path. If not investigate how to get VS Code into the Path. If in the path execute the following in the command line.

```
git config --global core.editor "code --wait"
```

test with this: `git config --global -e`

If you set up, the command above will open the .gitconfig file which is where git stores its config variable in VS Code.

See the answer to a question about this on Stack Overflow for more information: Editor For Git]

How to Use Visual Studio Code as Default Editor For Git

Adding VS Code to your computer Path

Mac: Select

- Mac: **Shell Command: Install 'Code' command in path** from the Command Palette.
- Windows: Make sure you selected **Add to PATH** during the installation.
- Linux: Make sure you installed Code via our new .deb or .rpm packages.

Additional References

https://help.github.com/articles/connecting-to-github-with-ssh/

This is the answer to "How to set up Dev environment with Visual Studio Code?".

Install Visual Studio Code

Download and install VS Code using the link below.

https://code.visualstudio.com/

Plugins

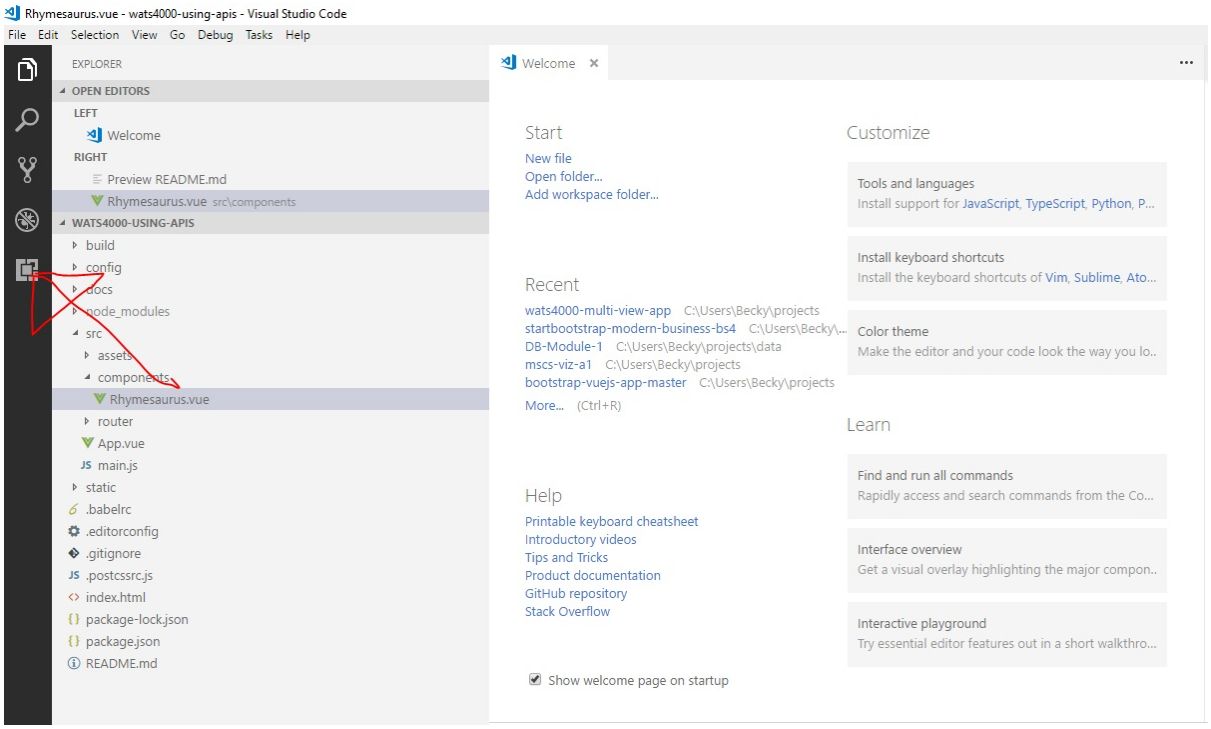
Plugins provide additional functionality.

Live-Server plugin

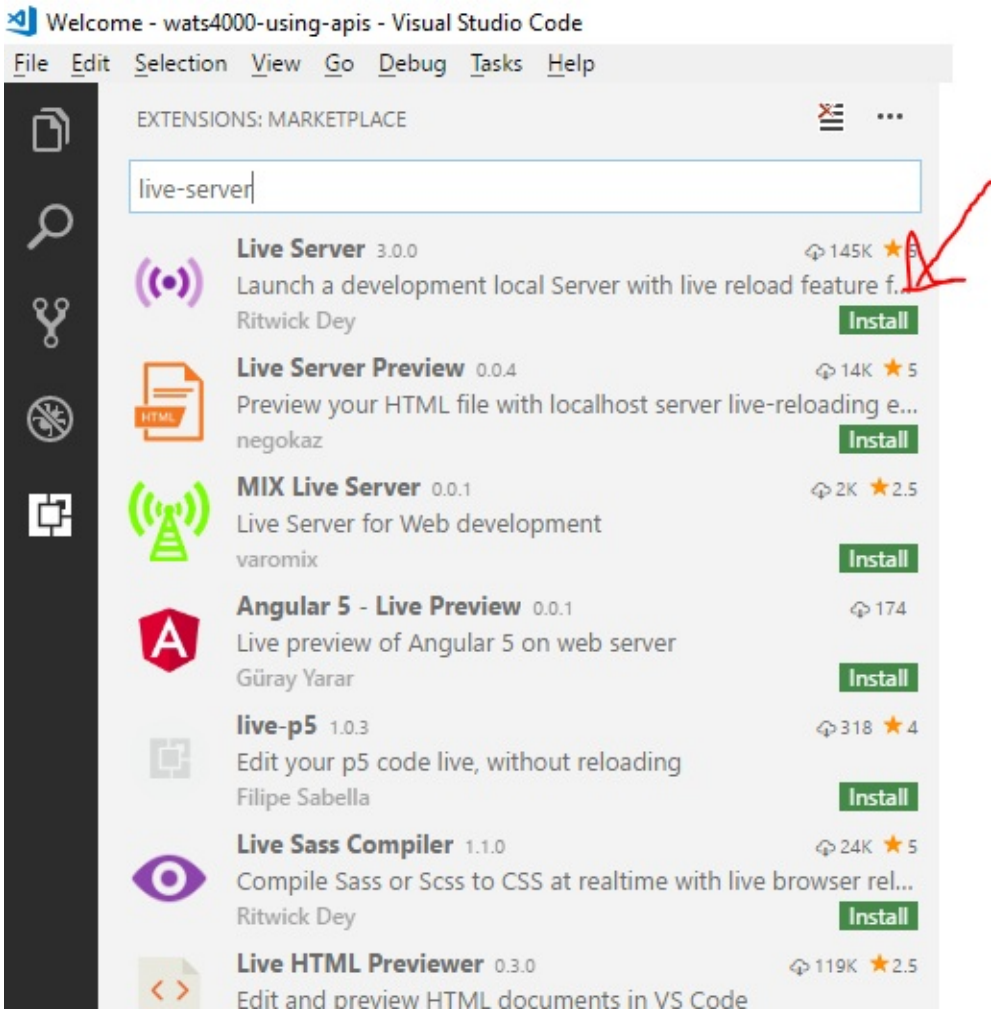
Live-Server will serve up web pages from your VS Code project to your default browser. It runs an HTTP server as a background process on the 5500 port, which is a development port.

To install:

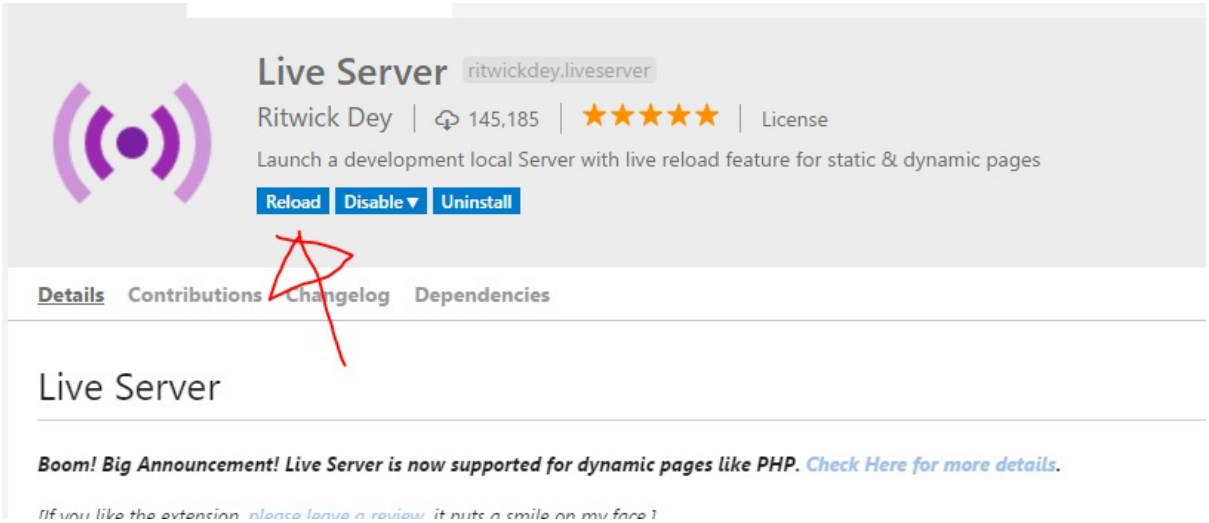
Click on the Extensions icon located at the bottom of the left nav bar in the VS Code Application.



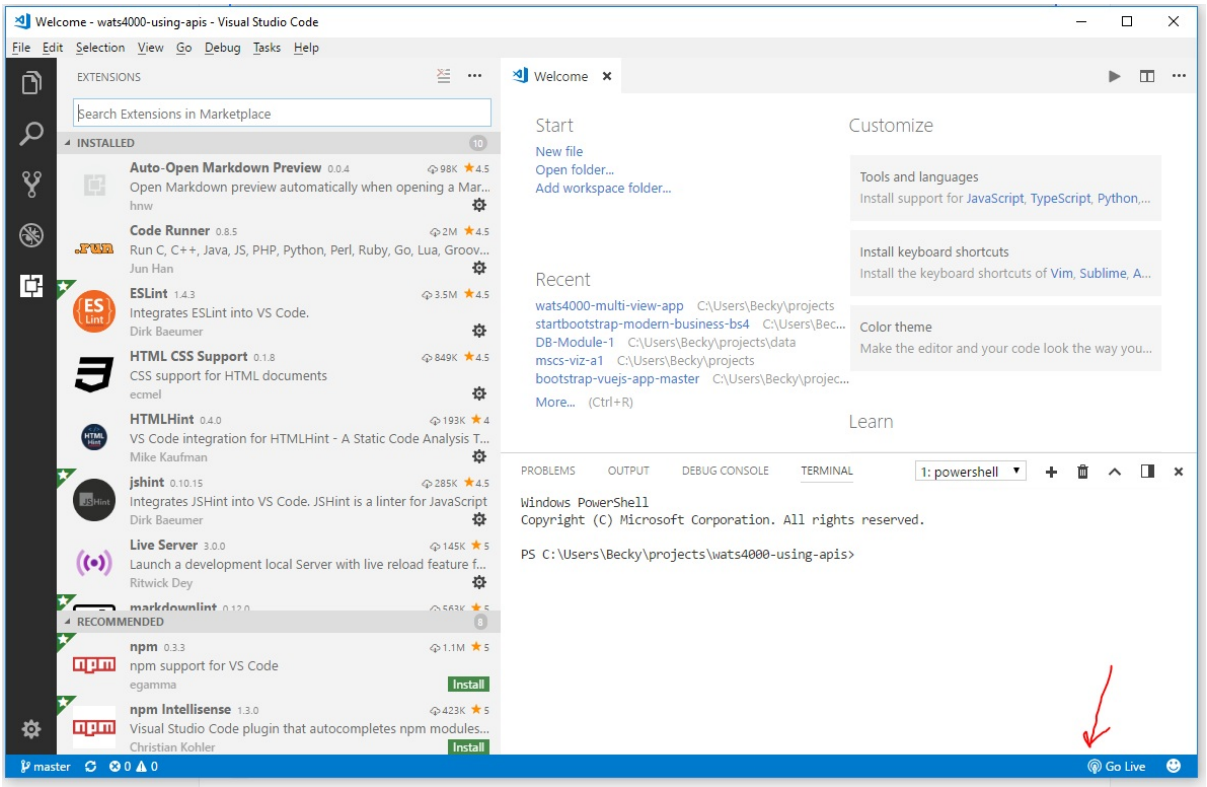
Search for 'Live-Server' and click on the Green Install button.



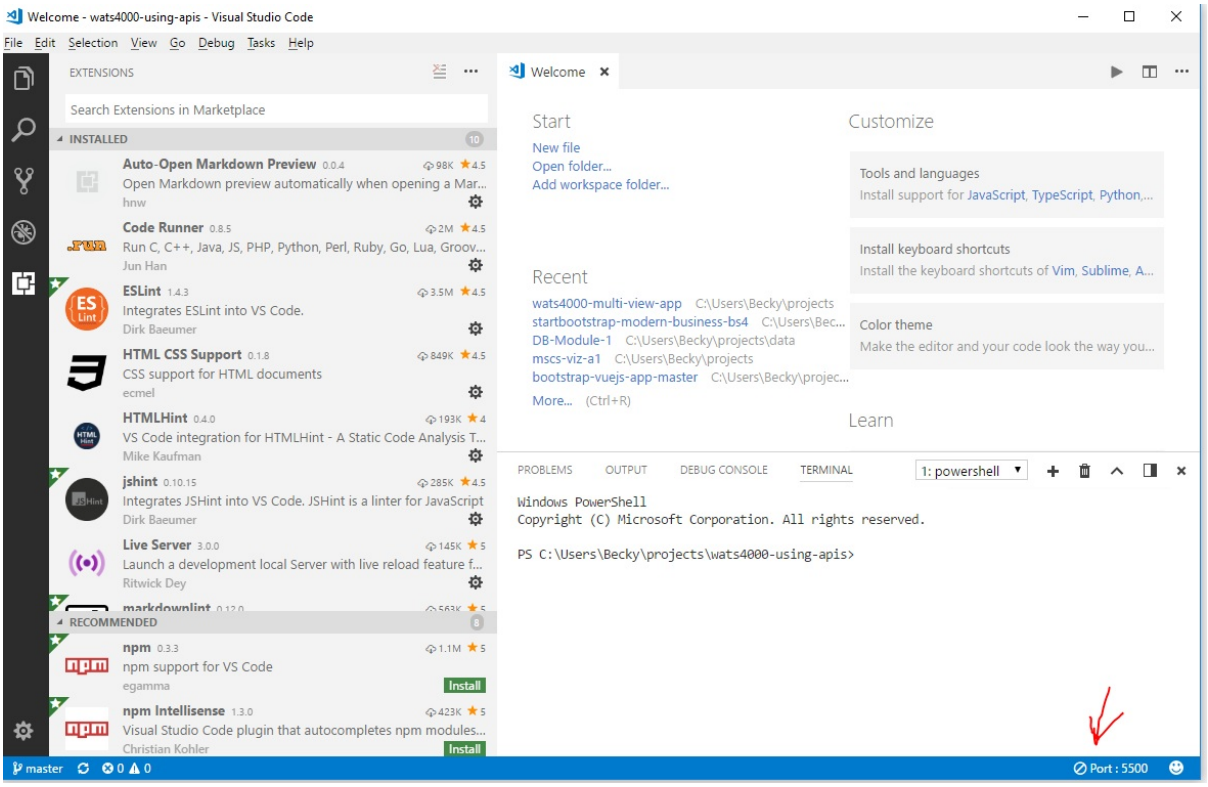
After the Extension is loaded, click on the Blue Reload button to complete the installation.



Now you should see a **Go Live** button in the bottom status bar. When you click on that, Live-Server will start and serve your **index.html** file by default.



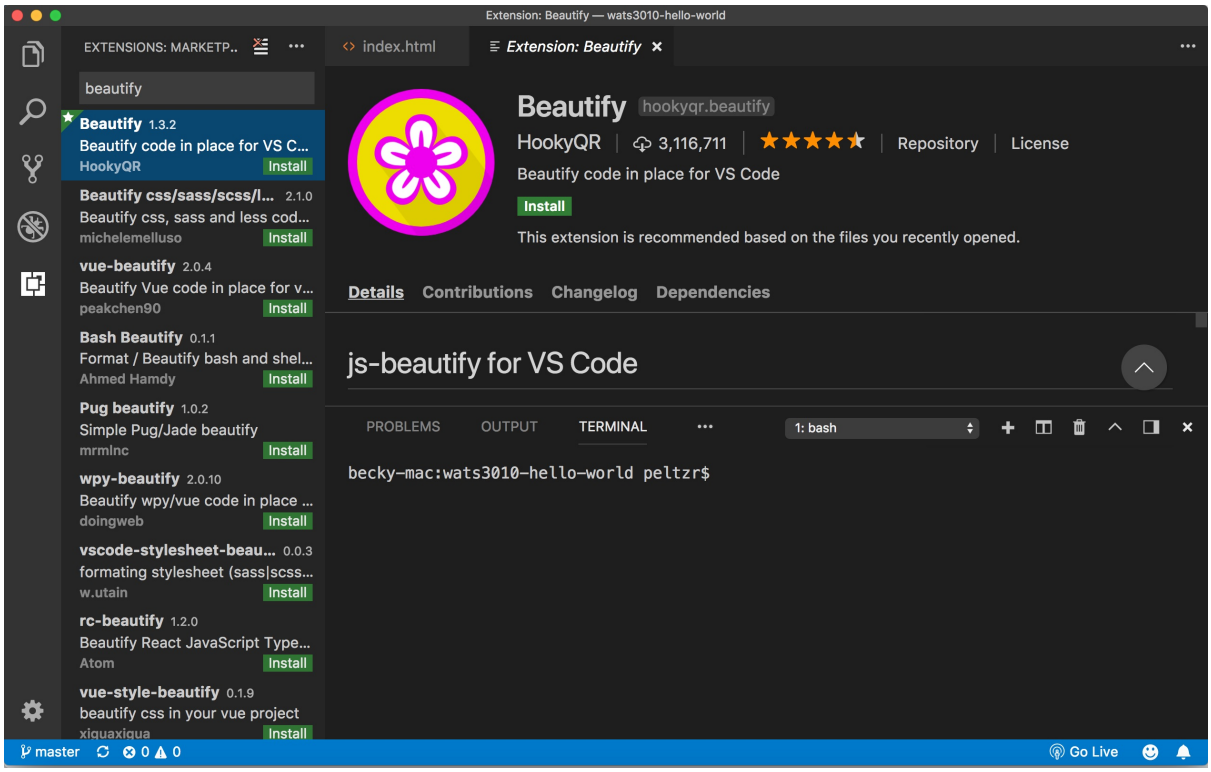
Once the server has been started, the **Go Live** button will change to **Port: 5500** which indicates that the server is running. The Go Live and Port:5500 buttons operate as a toggle. When you save changes in files that are being served, the server will automatically reserve the files. This will ensure that your browser is always running the latest saved changes.



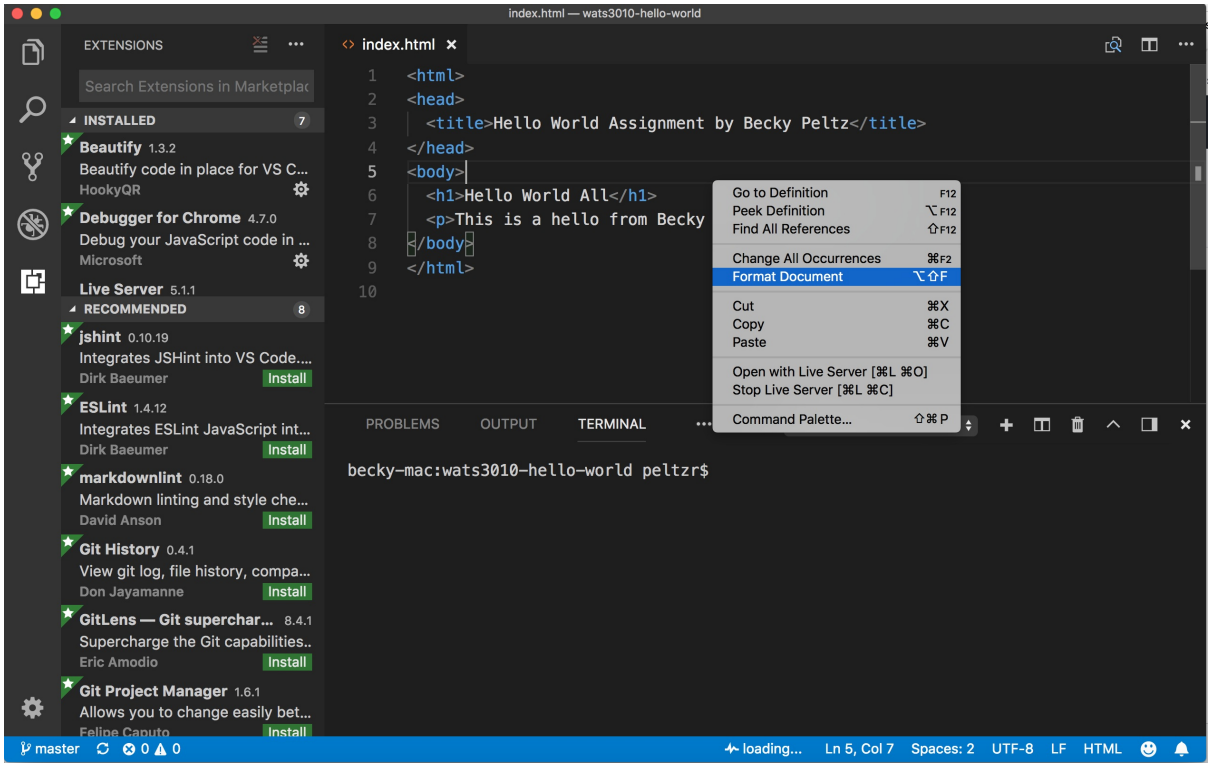
Beautify Plugin

The Beautify Plug in will help with code formatting. It will allow you to right-click on a document that you are editing and see a "Format Document" option. Clicking on this will format your page. As with all VS Code plugin installs, you go navigate to the plugin section, search for the plugin you want to install, press the **Install** button and then press the **Reload** button to make the plugin active.

The picture below shows the install button.

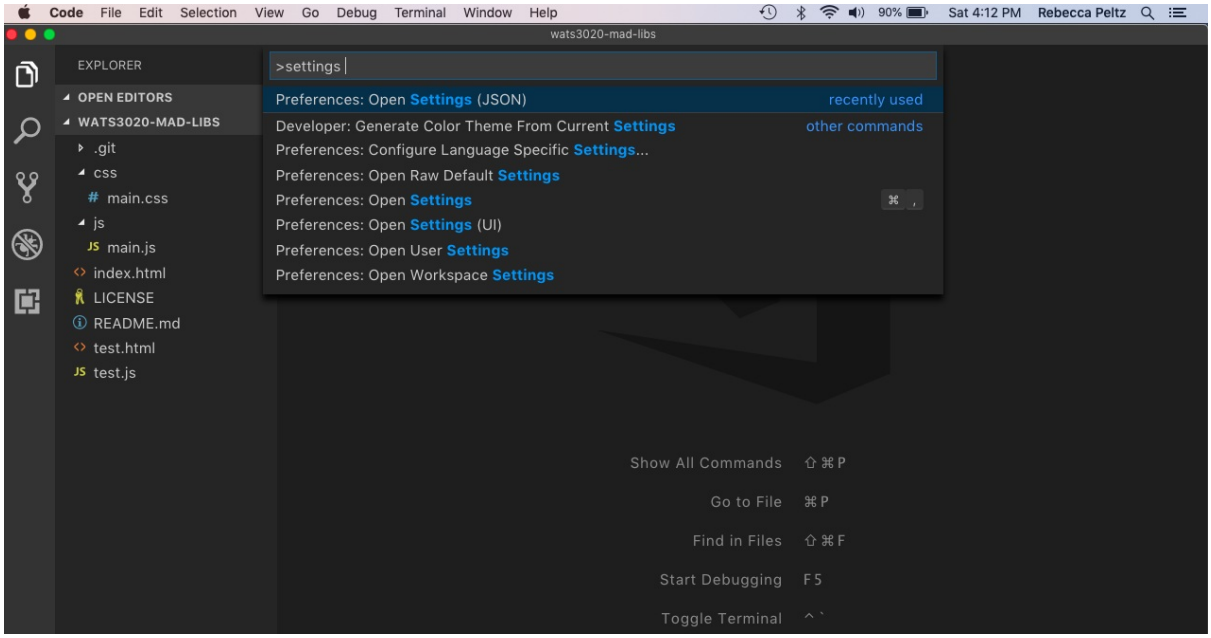


The picture below shows the Format Document option that you see when you right click in a document.

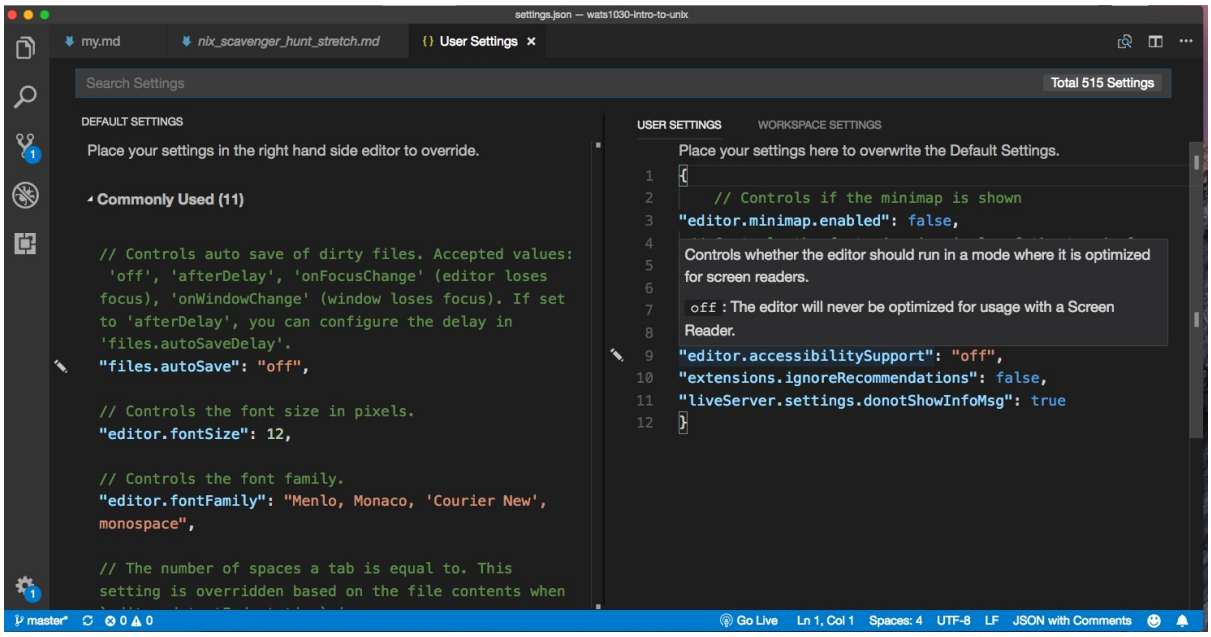


User Settings

VS Code allows you to customize settings for all projects (User Settings) or for a single project (Workspace settings). VS Code provides a GUI settings manager by default. To modify default settings by upgrading JSON configuration directly use the open command (CTRL-Shift-P on Windows or CMD-Shift-P on Mac) and type in "Open Settins (JSON)".



You will see 2 files side by side. On the left are the default settings and on the right are the User setting overrides.



To change a default setting find the setting on the left and then copy it to the right with your desired setting. For example, if you don't want to see the minimap on the right hand side of the application, you can make the following entry in the file on the right. Notice that options are key : value pairs and that they are commas separated.

```
{
"editor.minimap.enabled":false
}
```

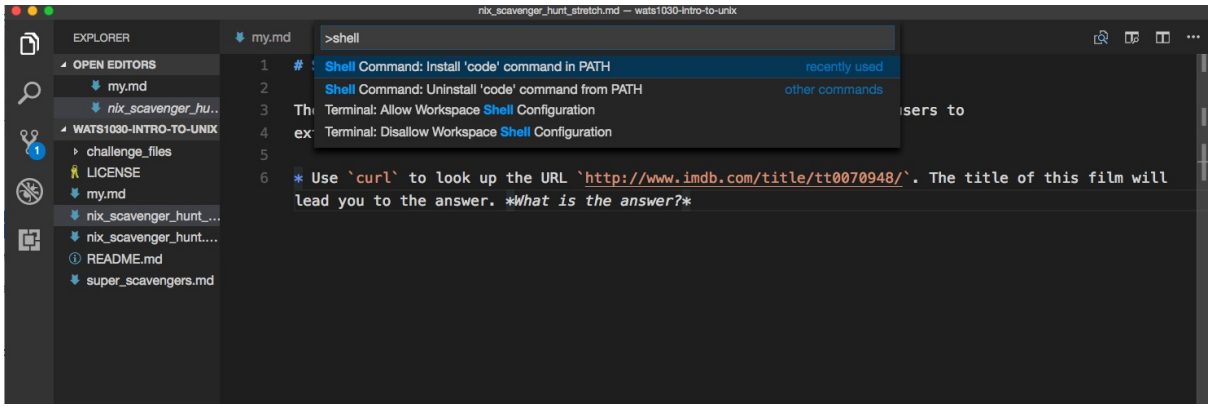
If **you are using Windows** and want to use "Git Bash" in for the VS Code terminal add the following to your User Settings:

```
"terminal.integrated.shell.windows": "C:\\Program Files\\Git\\bin\\bash.exe"
```

How to Easily Add VS Code to the Mac PATH

It is often handy to be able to open VS Code from the command line. In order to do this the "code" command which is the name of the VS Code program must be in the machine's list of programs which are stored in the environmental PATH variable. It will get installed in the windows PATH during Windows install. For the Mac, you need to do this:

Open Visual Studio Code and press `Command + Shift + P` then type **Shell** in command palette now you are able to find this option like **Shell Command : Install code in PATH** from suggested list in command palette. Select that options.



This is the answer to "How to install create new Github repo from existing file folder?"

New repository from existing local folder

This assumes you have installed Git locally.

Online:

Go to your github account https://github.com/<account name>

In the upper-right corner of any page, click the "+" icon and choose to 'New Repository'

Fill out the new repository name and give it a description in the New Repository form

Make note of the name of the repo

Local:

On Mac, open terminal and on Windows open Git Bash.

Follow instructions below substituting your account name and the repo you created above in the command.

`cd` to the root of the folder you want to add to github

`git init`

```
git remote add origin git@github.com:<account name>/<exisisting remote repo>.git
```

```
git push -u origin master
```

`git add .`

`git commit -m"first commit"`

If you added a license or Readme while setting up the new repository you will need to "pull" before "pushing"

`git pull origin master --allow-unrelated-histories`

`git push --set-upstream origin master`

This is the answer to "How to create gh-pages branch for Github hosting?"

Create gh-pages branch for Github hosting

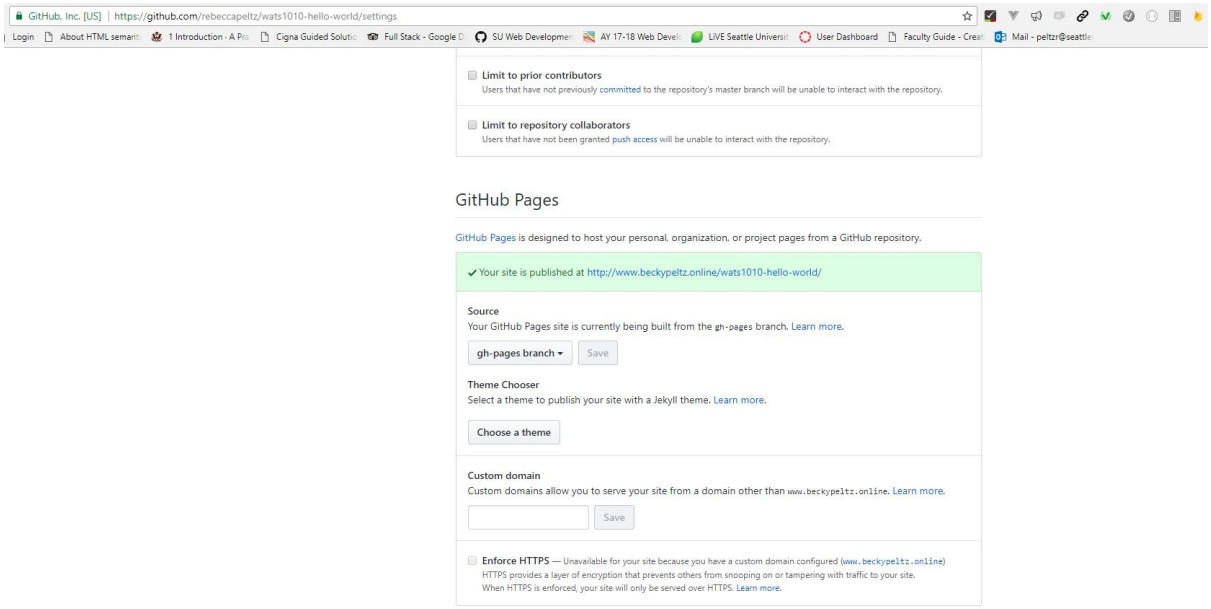
Github will host the web pages that you create in your Repos. One way to set up Github hosting is to create a gh-pages branch in your repo. Any code in that branch will be hosted at an address that follows this pattern

```
https://<account name>.github.io/<repo name>
```

For example if my account name is `janedev` and my repo name is `wats3010-hello-world` , and I have created a gh-pages branch on my repo, I will find the index.html located in the root of the repo served up at this URL:

```
https://janedev.github.io/wats3010-hello-world
```

If you have created a gh-pages branch, but are unsure where it is hosted you can click on the Settings tab on the main page of your repo and then scroll down to find the link to the hosted web pages.



Commands to create gh-pages

When you start working on a new repo you will be in the master branch.

From a bash terminal (git bash on Window or terminal on Mac)

Check which branch you're in

```
git status
```

If you are in master and you want to create a gh-pages branch on Github enter the following commands.

First push all your work to master. You can add a single file or all files. The dot (.) indicates all files in this folder and below.

`git add <filename>` or `git add .`

```
git commit -m"my comment message"
```

```
git push
```

Next run **checkout -b** to create a new branch to be created with the name **gh-pages**

```
git checkout -b gh-pages
```

Next run push -u origin to update Github which is a remote location. Specifiy gh-pages:gh-pages tells github that the branch is named gh-pages locally and remotely. The format is local:remote.

```
git push -u origin gh-pages:gh-pages
```

Commands to continue work on gh-pages branch

If you are returning to work on code in a repo that you have worked on before, you will not need to run checkout -b because you don't want to create a new branch, you just want to access the existing branch. Start with getting status.

```
git status
```

If the status indicates that you are already on gh-pages, you don't have to do anything. If you're on master, you can checkout gh-pages. Notice **we don't need the -b** when we are not creating a new gh-pages branch.

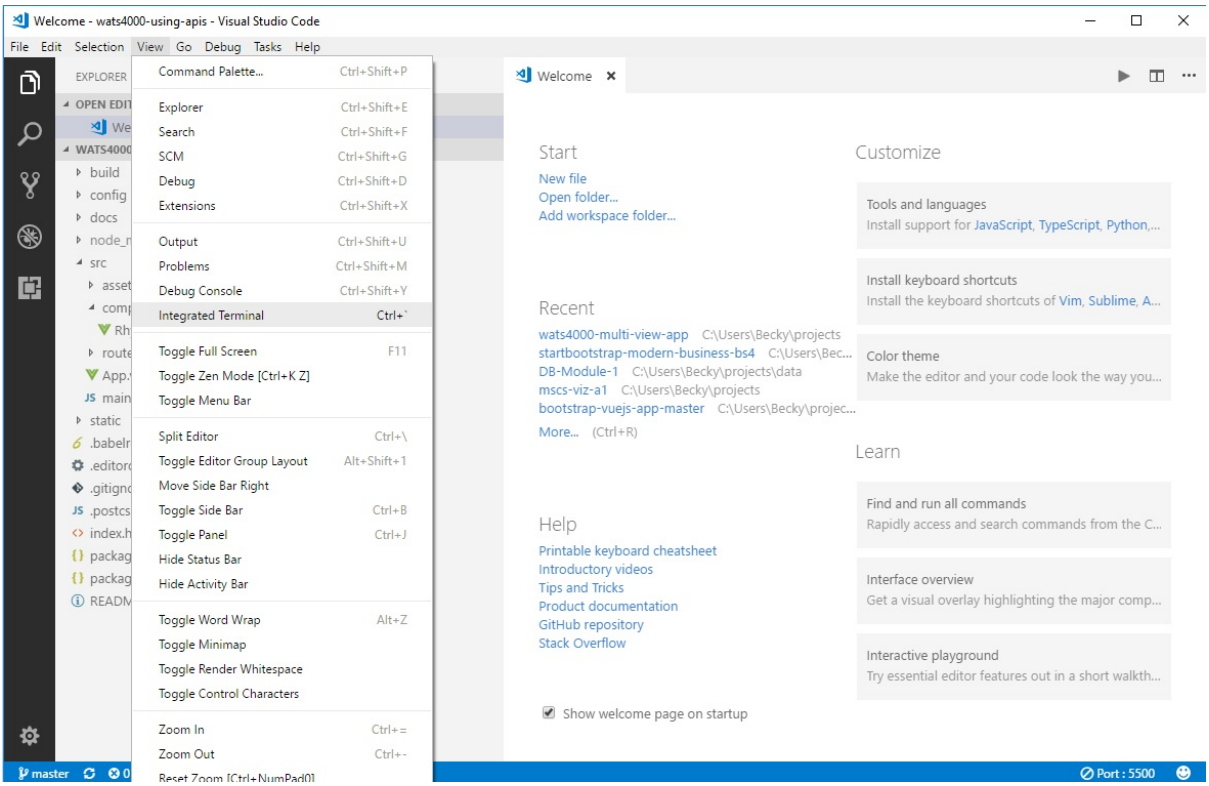
```
git checkout gh-pages
```

This is the answer to "How to use the Visual Studio Code Terminal?"

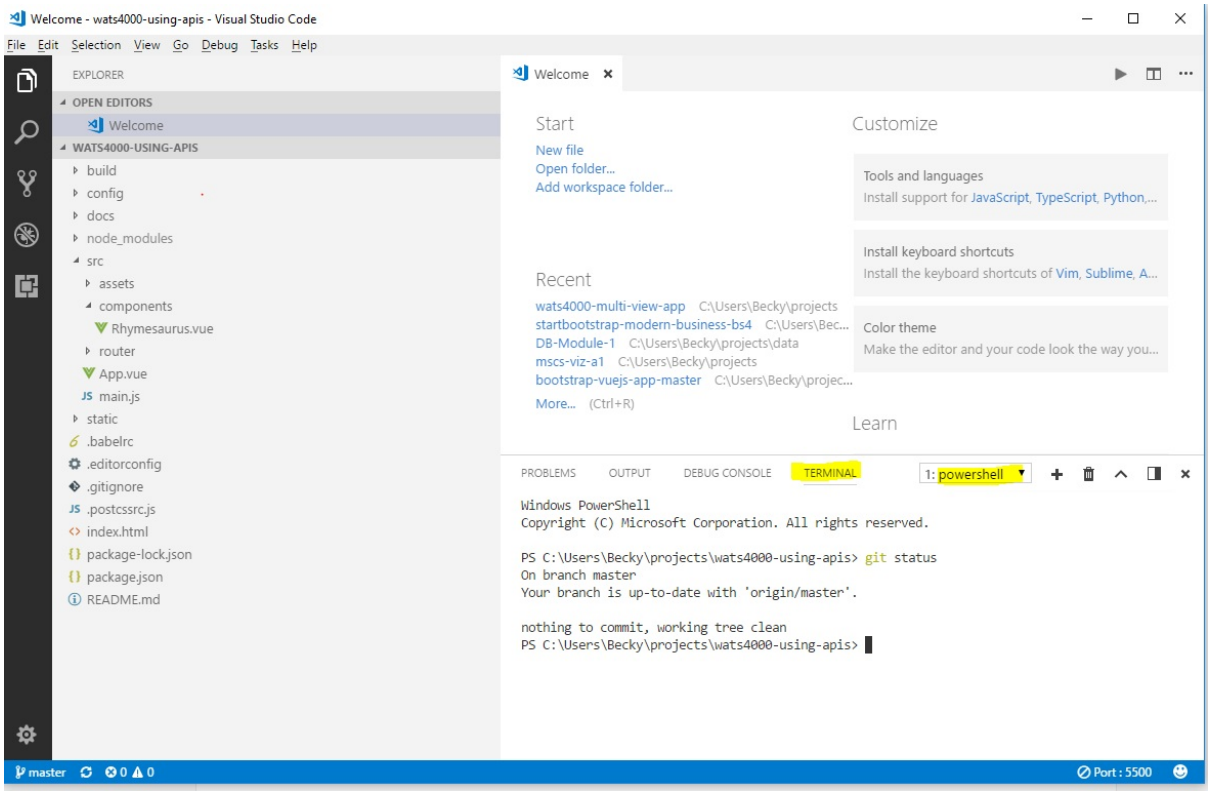
Using Integrated Terminal

Visual Studio Code allow you to access a terminal from within the application. You also have the ability to set up different terminal interfaces. As developers, we use the terminal to communicate with Github and issue commands to maintain the repo were working on.

To open the integrated terminal from the menu click on `View Integrated Terminal` from the menu. You can also use the shortcut Ctrl-Tick. The Tick is located on top left of the keyboard below the Escape key.



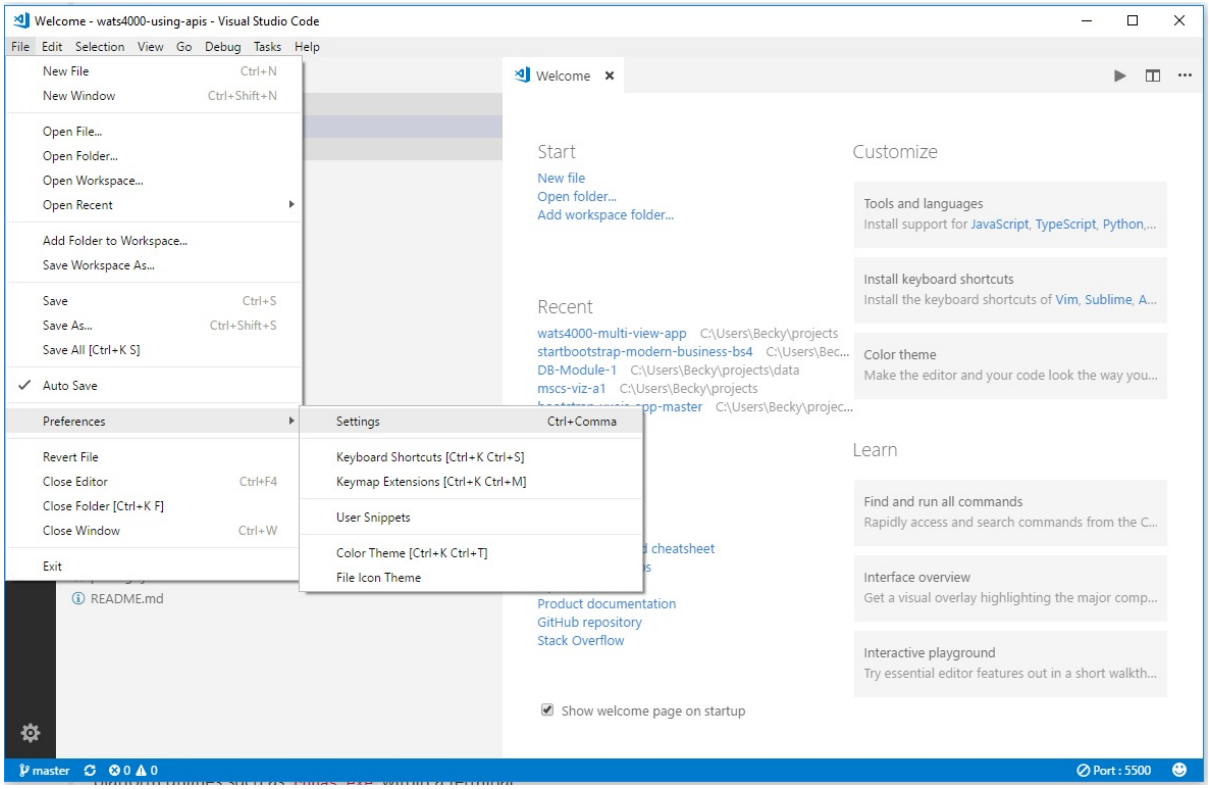
The picture below shows the terminal opened at the bottom and a command for git status has been issued. This terminal was opened on a Windows 10 workstation where **Powershell** it the default terminal. On a Mac the default terminal is **bash.**



Windows: Set 'Git Bash' as the default Integrate Terminal

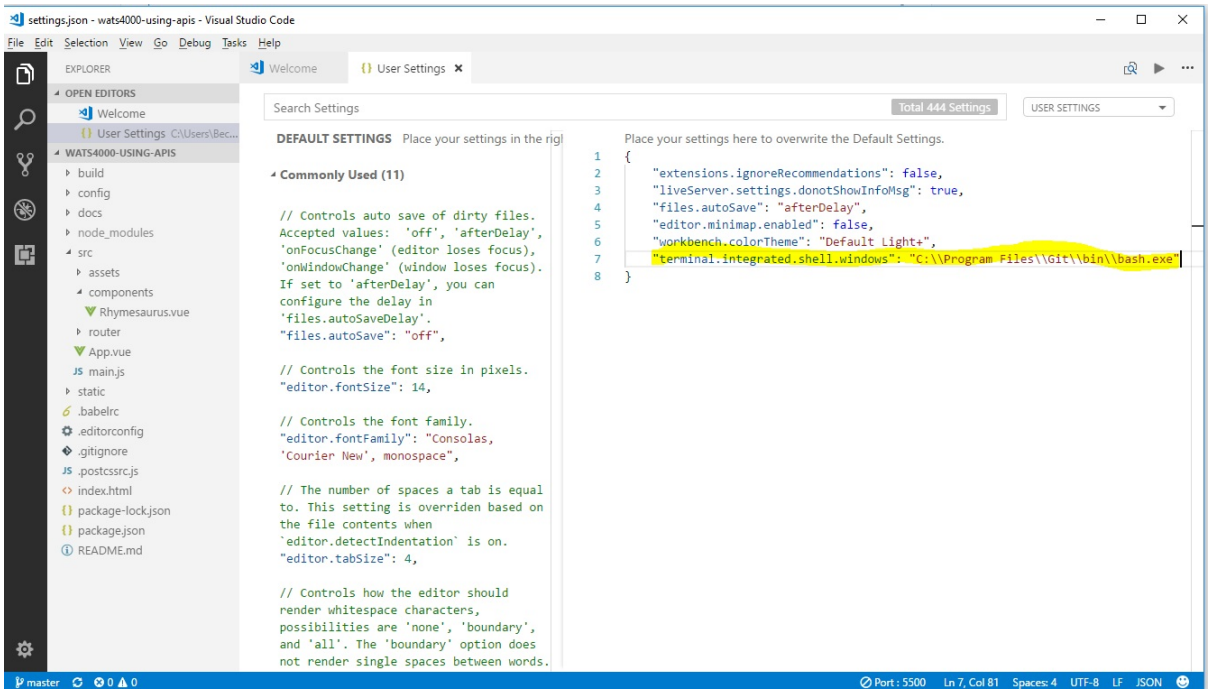
The default interface for Mac is the OS X bash, which is works well when communicating with Github. For Windows users, it is useful to configure your dev environment to use Git Bash as the default integrate terminal in VS Code.

You will first access User Settings on Windows from the **File Preferences Setttings** menu.

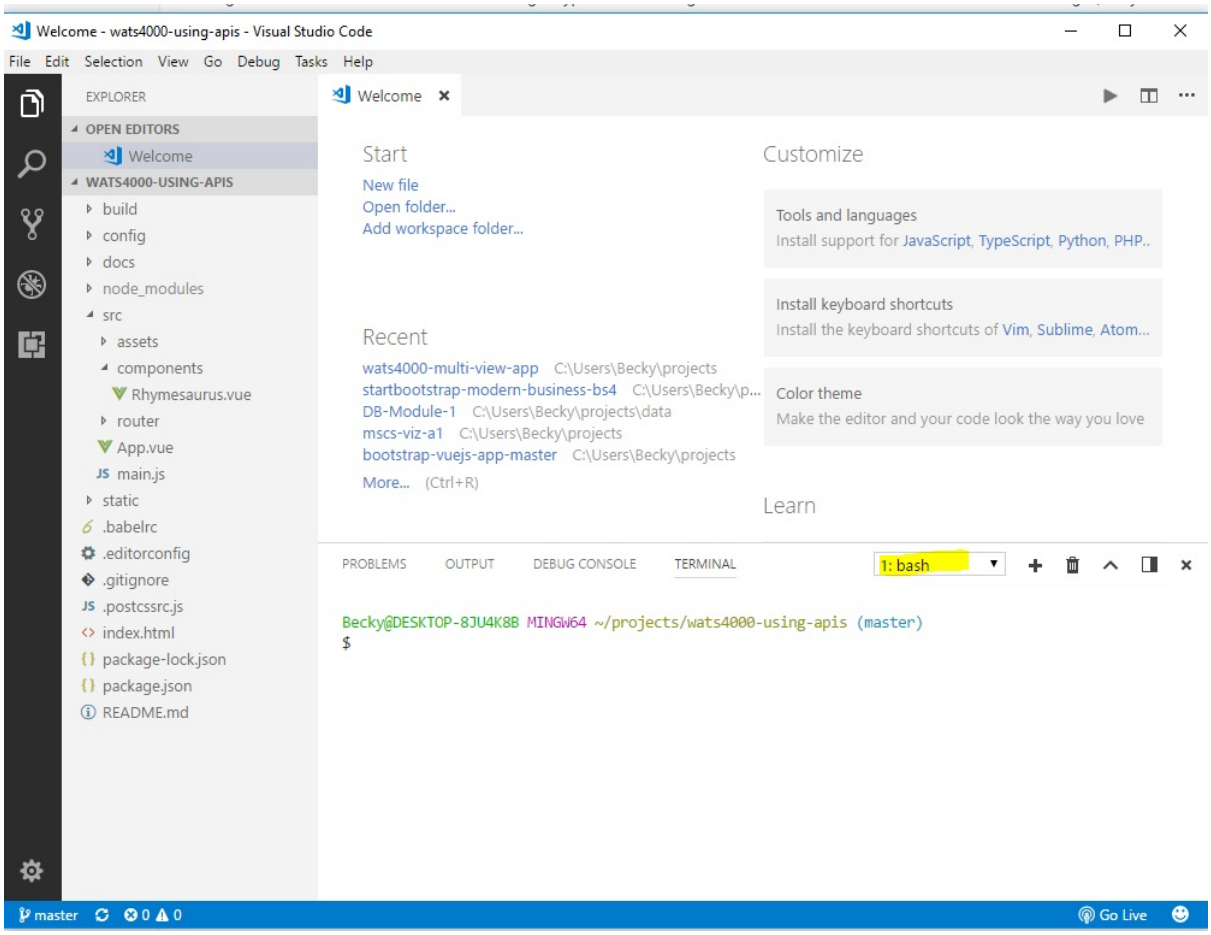


This will open up two documents side by side. On the left are the default settings and on the right are the User override settings. To configure Windows to use Git Bash, you must first have installed Git. Git installation will have placed the Git Bash executable in this location: C:\Program Files\Git\bin\bash.exe. You will create an entry in your User settings override to use this for the integrated terminal. In the screen on the right type the following. If there are other entries in the screen on the right, they must be comma separated. The extra back slashes (\) you see are escape characters. After making the entry you should save and close the file. To make the configuration complete close and reopen VS Code.

```
"terminal.integrated.shell.windows": "C:\\Program Files\\Git\\bin\\bash.exe"
```



After your new configuration is in your terminal should look like the picture below with bash as the terminal interface.


Welcome - wats4000-using-apis - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
Welcome
WATS4000-USING-APIS
build
config
docs
node_modules
src
assets
components
Rhymesaurus.vue
router
App.vue
main.js
static
.babelrc
.editorconfig
.gitignore
.postcssrc.js
index.html
package-lock.json
package.json
README.md
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
wats4000-multi-view-app C:\Users\Becky\projects
startbootstrap-modern-business-bs4 C:\Users\Becky\p...
DB-Module-1 C:\Users\Becky\projects\data
mscs-viz-a1 C:\Users\Becky\projects
bootstrap-vuejs-app-master C:\Users\Becky\projects
More... (Ctrl+R)
Customize
Tools and languages
Install support for JavaScript, TypeScript, Python, PHP...
Install keyboard shortcuts
Install the keyboard shortcuts of Vim, Sublime, Atom...
Color theme
Make the editor and your code look the way you love
Learn
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
Becky@DESKTOP-8JU4K8B MINGW64 ~/projects/wats4000-using-apis (master)
$
master
0 0
Go Live

This is the answer to "How to kill a process based on port number?"

Kill a process based on port number

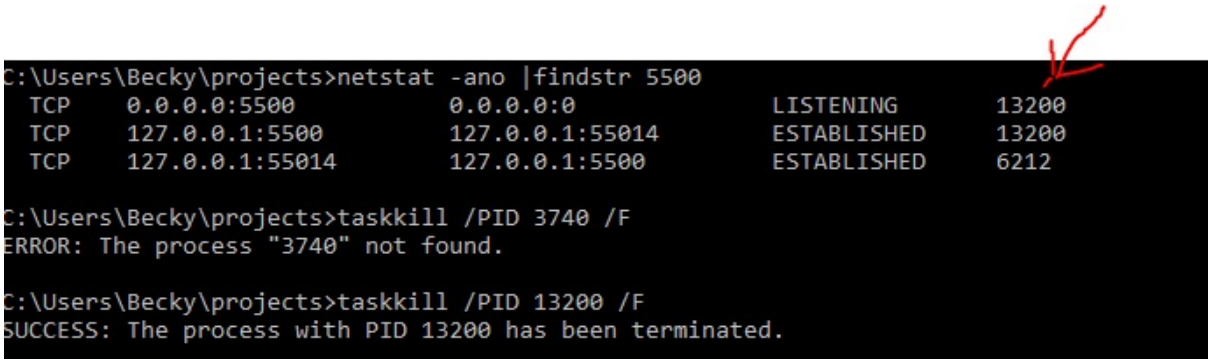
If you close out VS Code with an active live-server running, it may keep the server process alive and you won't be able to create a new server, just by opening a new VS Code project.

If you find yourself getting errors when trying to run live-server, you may need to kill an old process running in the background. The command to do this vary depending on operating system interface. Note that live-server run on port **5500** by default.

Windows DOS Commands

```
netstat -ano | findstr :5500
```

```
taskkill /PID <process ID> /F
```



Windows Powershell Commands

```
netstat -a -b -n -o
Stop-Process <pid>
```

You can also run **resmon.exe** to find the process ID.

Mac OS X Commands

```
lsof -i :5500
```

```
kill -9 <pid>
```

This is the answer to "How to validate HTML and CSS?"

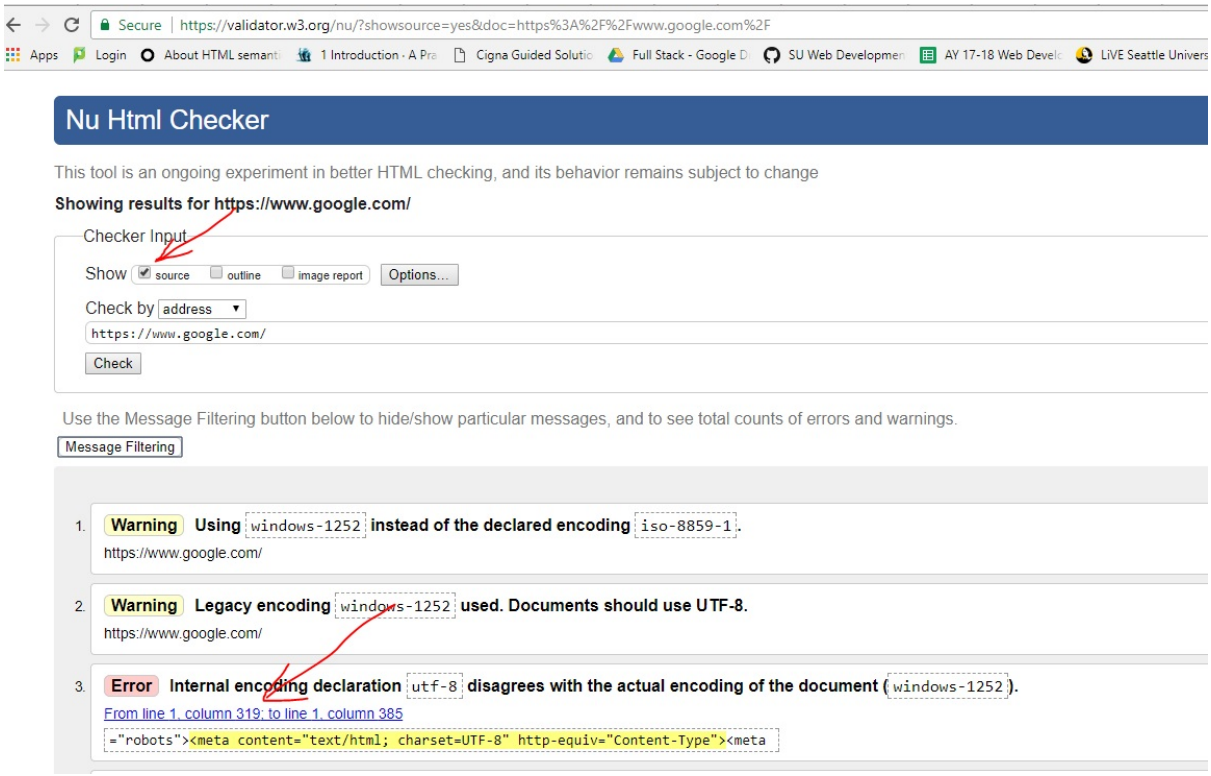
HTML Validation

You can use online validation or a chrome extension. Your web page must be running on the internet to use online validation.

Online HTML Validation

The URL for the online HTML Validation is: https://validator.w3.org.

1. Copy your URL in the buffer and paste it into the w3c validation form.
2. Check the Source box as this will allow you to look at the source code associated with any errors.
3. Submit the form to see the errors.

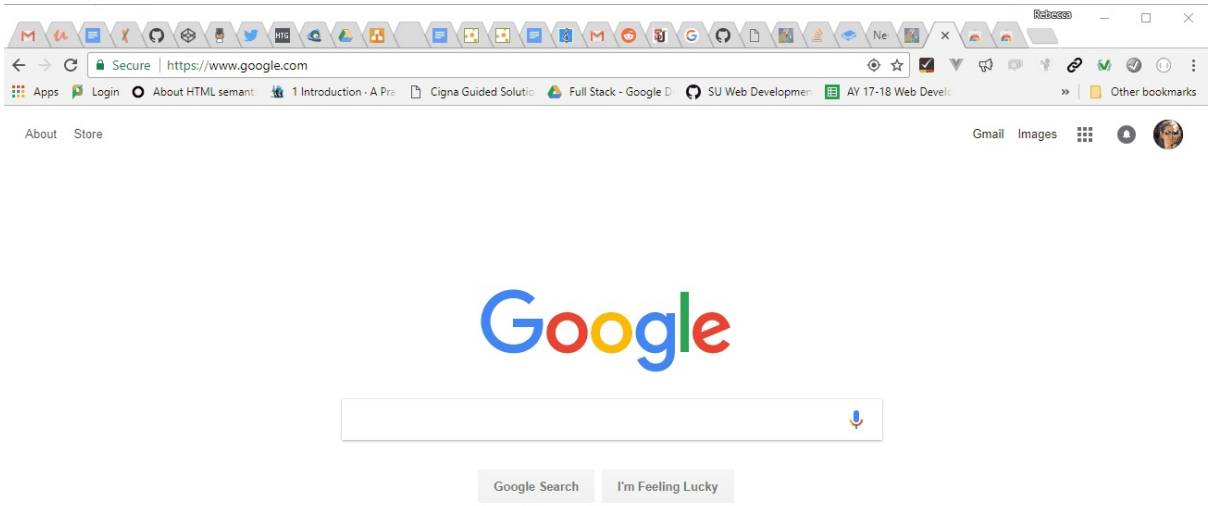


HTML Validation with Chrome Extension

Go to the chrome app store and Search for html validator (with Extensions selected) or go directly to the extension: Html Validator.

You will find the validation issues listed in Chrome Dev Tools which you can get to by right-clicking and selecting Inspect. There will be a new tab named HTML Validator that will show problems with HTML.

The gif below show an analysis of the google search page.



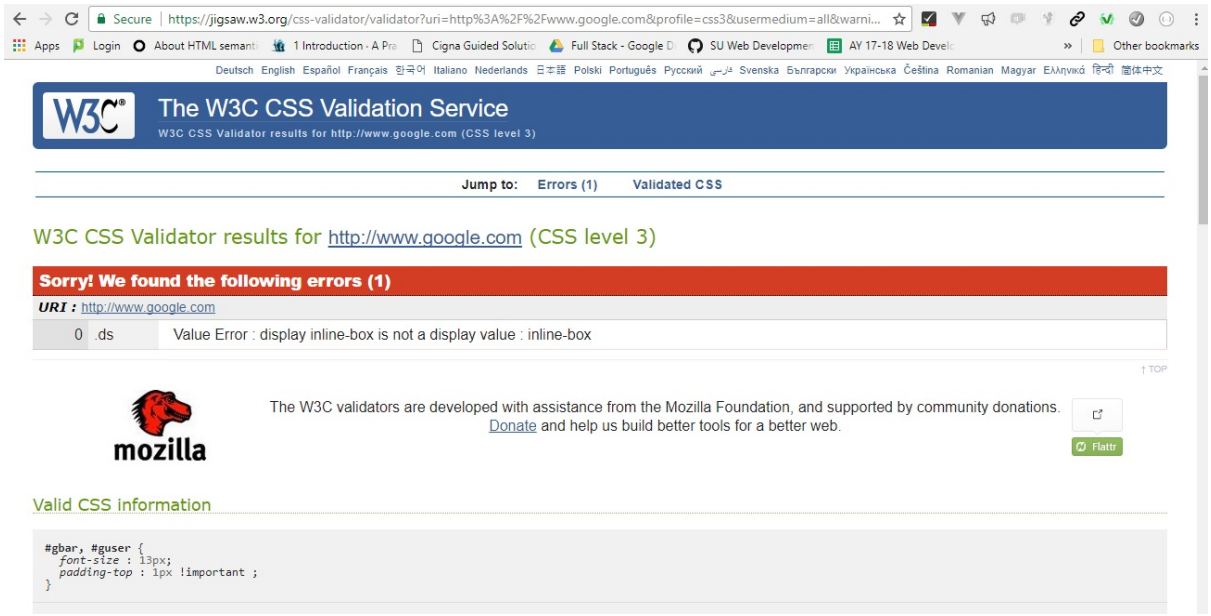
CSS Validation

There are online and chrome extension options for CSS Validation.

Online CSS Validation

Copy URL into buffer and test it at this location: https://jigsaw.w3.org/css-validator/

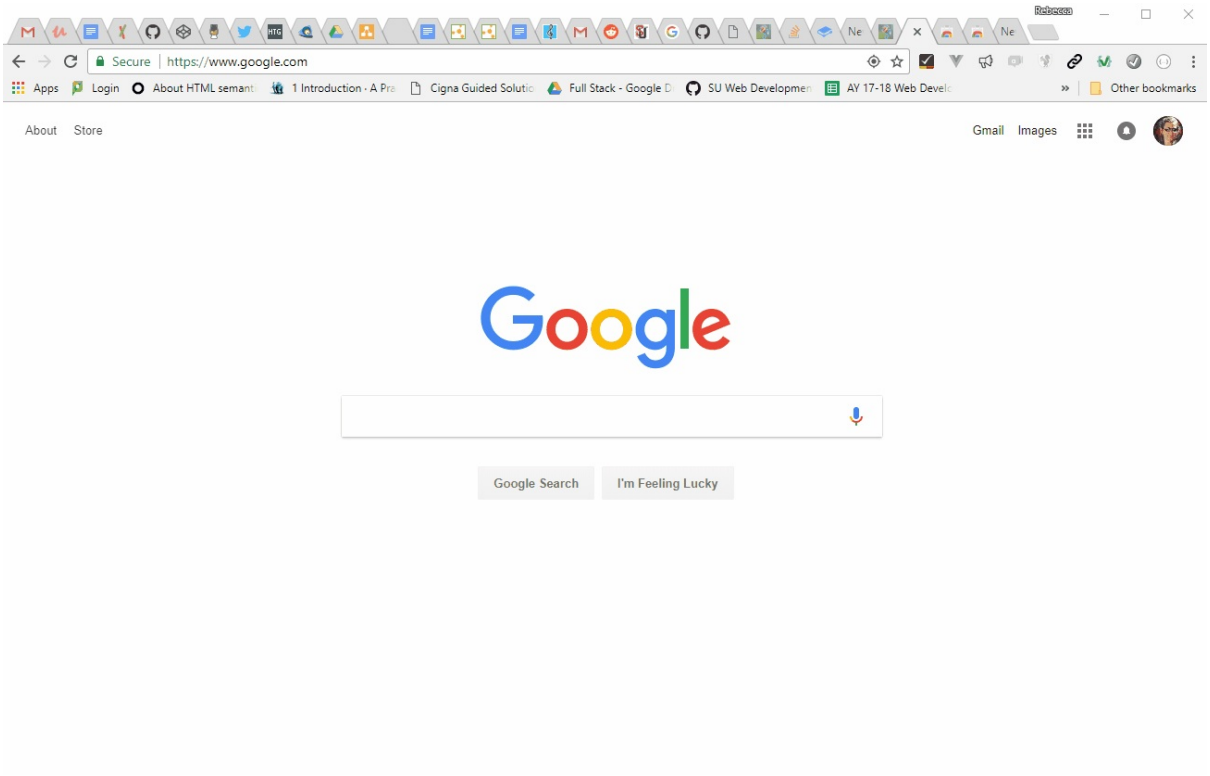
The picture below shows a CSS validation of the google search page.



CSS Validation with Chrome Extension

Go to the chrome app store and Search for html validator (with Extensions selected) or go directly to the extension: Style Validator.

Click on the extension icon in Chrome to see CSS errors.

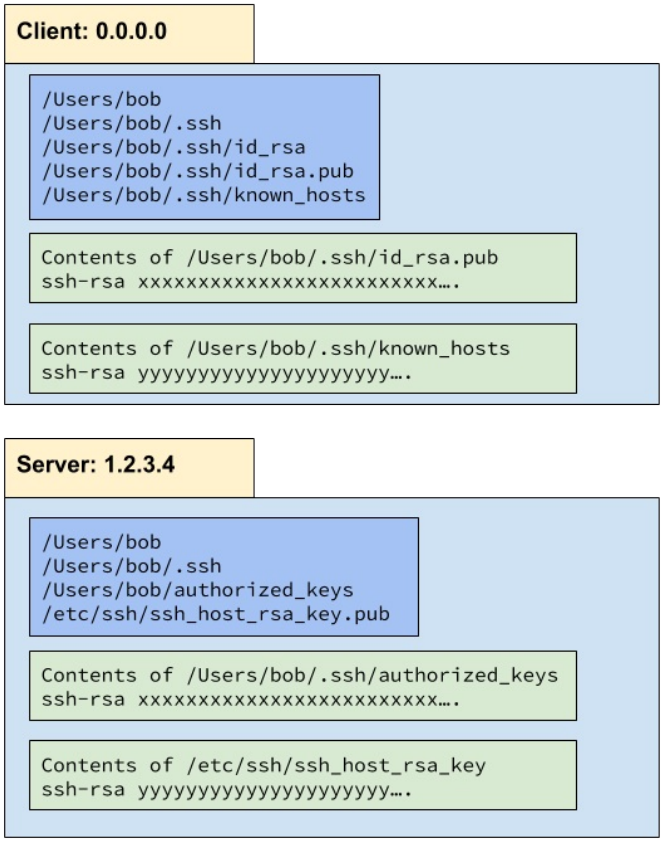


This answers the question "How to setup SSH on a client (local machine)?"

SSH is the acronym for "Secure Shell". An SSH connection allows you to connects one node(machine) in a network to another without have to to enter a password. The relationship between the two machines will follow a client server model. The machine on which you type "ssh <username>@<ip address" is the **client (aka "local machine")** an the machine that you are trying to connect to is the **server (aka "host machine")**. The server always maintains the information about the username and password. A machine may act as either client or server depending on whether the user is logged on to it or trying to connect ot it: if the user is already logged on to it, it is the client.

In order for a client to connect to a server using SSH, it must set up a public key/private key pair. The public key and private key provide the encryption needed for secure authentication and authorization. An algorithm can verify that a give private key matches a given public key. Both the public and private keys are stored on the client. Before the client can connect to the server, the server must record the public key of the client in an file called **authorized_keys** located in the **.ssh** directory under the home directory of the user on the server machine. When the user issues the SSH command from the client machine the public key (and proof that it has the matching private key) is sent to the server. If there is a match between the public key sent to the server and one of the keys in the **authorized_keys** file, then a test is made on the client to see if the server is trusted. During ssh initialization the host send the client a host key. The client checks the host key against entries in the **known_hosts** file. If the host key is not found a message pops up asking the user to verify the host. Once verified the host key is added to the client's **known_hosts** file.

For example if I want to issue the following command: ssh bob@1.2.3.4, the following setup must exist on the the client and server machines.



You can think of **authorized_keys** as a file that helps the server trust the client and **known_hosts** as a file that helps the client trust the server. The public key, **authorized_keys** and **known_hosts** files are all text files and the contents may be safely copy and pasted. It is also possible to add a config file to the client .ssh directory and the **config** file

can be used to map servers to different public/private key files if needed. In general, SSH will look in the .ssh directory and try all keys if needed when attempted to authenticate.

A machine may contain both and **authorized_keys** file and a **known_hosts** file because it can be operating in the either role at different times.

Setting up SSH on a Mac or Linux Client

Check to see if key files already exist because you don't want to replace them if they might already be in use.

```
ls -la ~/.ssh/
```

Look for `id_rsa` _ _and `id_rsa.pub`

Generate the key files

```
ssh-keygen
```

```
Check that the key files have been properly generated
```

```
ls -la ~/.ssh/
```

```
Copy the public key to the buffer
```

```
cat ~/.ssh/id_rsa.pub
```

Select contents of file from screen and `CTRL-c` to put in buffer

Use a text editor and paste the key into the authorized_keys file on the server you wish to access.

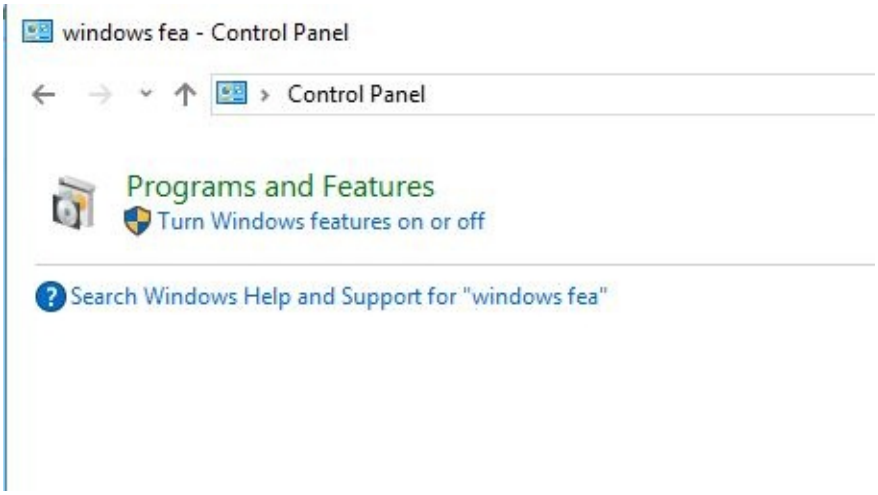
Setting up SSH on a Windows Client

If you are using Windows 10, you have 3 options for creating the public/private key needed for SSH and running SSH.

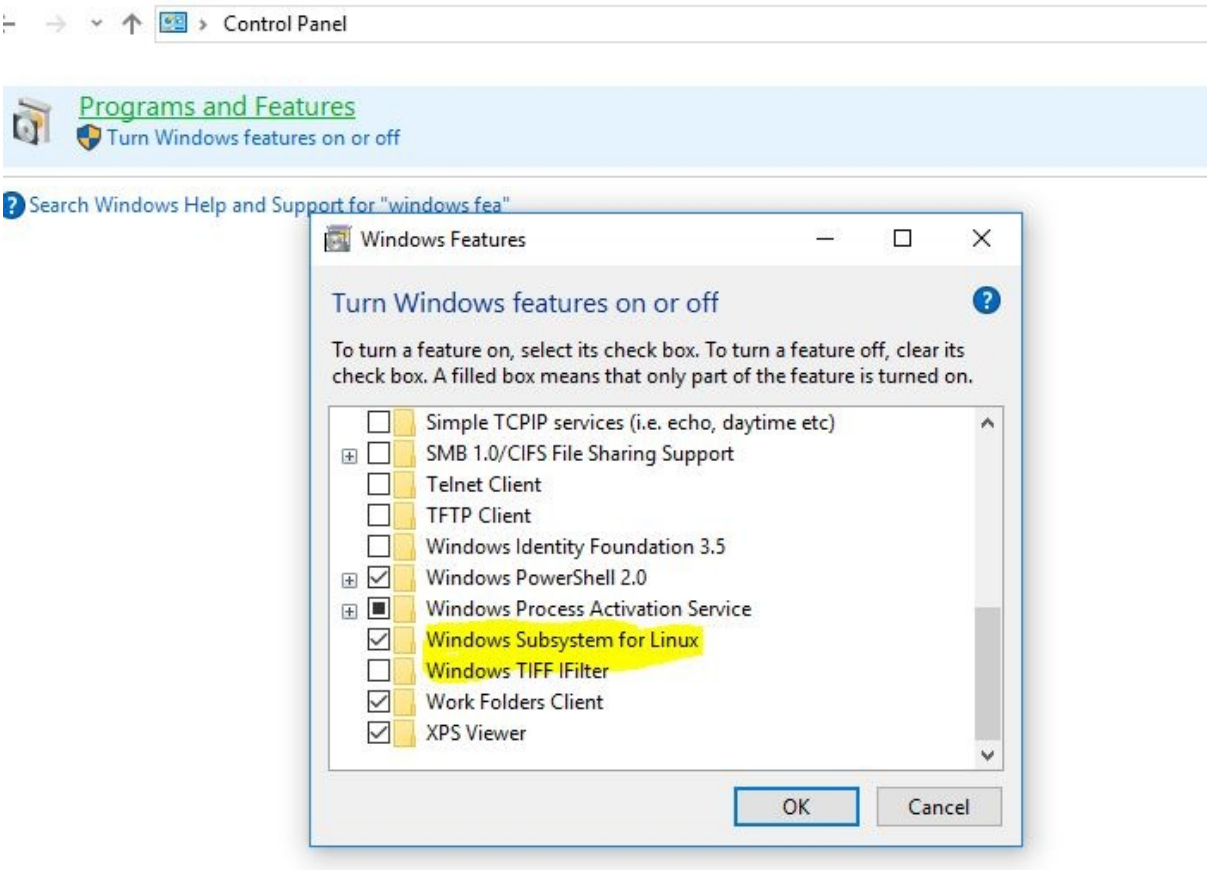
1. Use **git bash** to create keys and run SSH. When you download **git** to you windows machine, you will get the **git bash** program which will provide a bash interface. See this article on **github.com**: https://help.github.com/articles/testing-your-ssh-connection/.
2. Use **puTTYgen** to create the public and private keys. Use **puTTY** to create an SSH Session from a GUI. See this youtube video for an example of how to use puTTY to connect to a server.
3. Install a linux shell on your Windows 10 machine. This will create a new file system, so you need to keep this in mind if you're developing on the Windows file system. See instructions for setting up the Linux Shell for Windows in the WATS Lab FAQ.

This article answers the question "How to Setup a Linux Shell on Windows 10?"

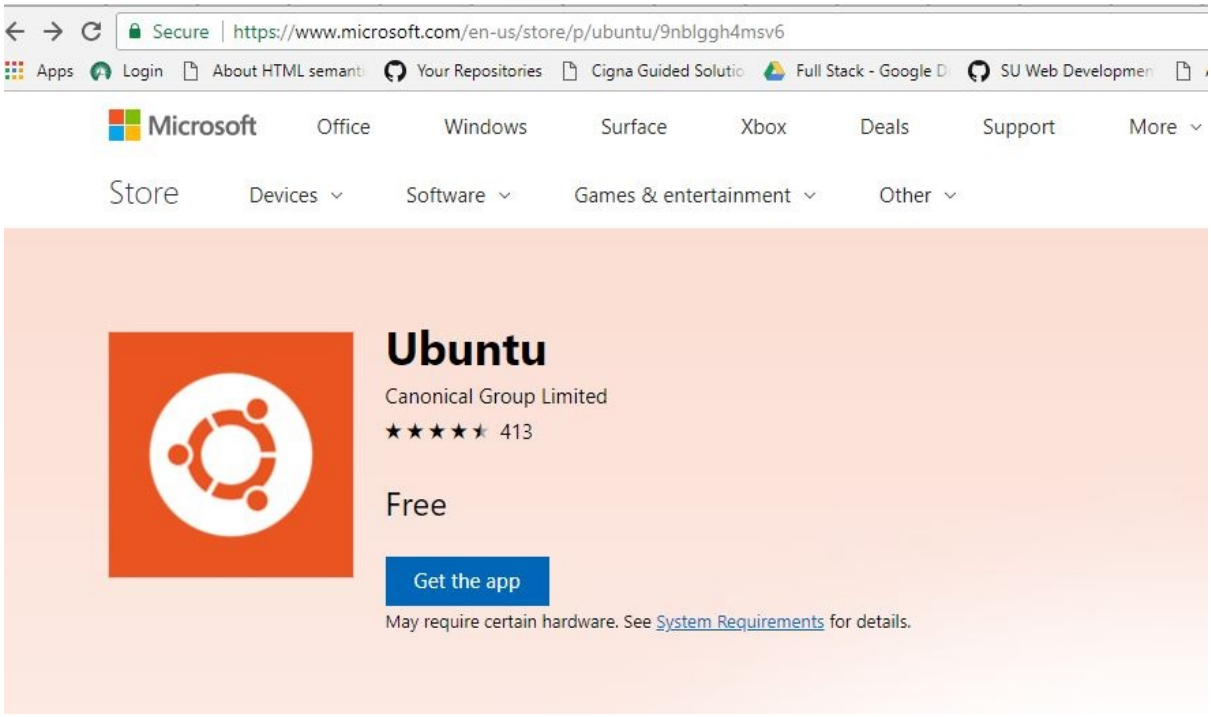
Modify the Windows Features to add Linux Subsystem. Click on Windows button, key in System, and choose System (Control Panel). Search for Windows Features. Click on "Turn Windows feature on or off".



Scroll down and put a check mark next to "Windows Subsystem for Linux" and click OK. This will ask you to restart your computer and you should respond yes.



When the computer is through rebooting, go to the Windows Store and search for Ubuntu. Click the "Get the app" button and let it download and install.



When it's through installing you can "Launch" the app. You will be prompted to create a user name and password. You can use the same username and password that you used to log into your windows computer but it will be in a different subsystem and will use a different home directory. To access it you'll have to launch the Ubuntu app. You'll be able to access files in the windows subsystem by referencing the /mnt/c/ folder to get to the root of the Windows subsystem. When you're in the Ubuntu shell you can use these commands to help identify who you are `whoami` and where you are `pwd` . If my user name is **bob** in both windows and in the Ubuntu shell, my Ubuntu home will be **/home/bob** and my Windows home will be **\Users\bob.** When you're the Ubuntu shell you can get to your linux home using `cd ~` and you get to your Windows home using `cd /mnt/c/Users/bob` .

How To Geek Provides a lot of documentation on setting up and working with a linux shell on Windows 10. See these links for more information:

How to tell if you have a 32 bit or 64 bit machine: https://www.howtogeek.com/howto/21726/how-do-i-know-if-im-running-32-bit-or-64-bit-windows-answers/

How to install linux shell on a windows 10 machine: https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/

How to install ubuntu bash on a windows 10 machine: https://www.howtogeek.com/261449/how-to-install-linux-software-in-windows-10s-ubuntu-bash-shell/

Using the linux terminal CLI (command line interface): https://www.howtogeek.com/140679/beginner-geek-how-to-start-using-the-linux-terminal/

This FAQ answer the question "How to add an admin user to Wordpress using mySQL?"

Add Admin User to Wordpress using mySQL

Sign on the mySQL. Digital Ocean provides the admin password to sign on into mySQL here: `/root/.digitalocean_password`

Copy the password into the buffer and paste after executing the following command:

```
mysql -u root -p
```

This will log you into mysql and you should see the 'mysql>' prompt.

Change to the wordpress database by executing the following command:

```
use wordpress;
```

Note you can find all databases with the following command

```
show databases;
```

and all tables with this command

```
show tables;
```

Verify that you are in the wordpress database by executing show tables; and seeing the "wp_" tables.

Then execute the following commands. Replace the "<>" with the data you want. Note that once the password is in the database it will be one-way encrypted with MD5 and you won't be able to see it in plain text. Note that the ID cannot already exist so you may want to issues the (`select * from `wordpress`.`wp_users`;`

to see what the current user ID's are and pick the next one)

Use the same ID value in all 3 queries. For example I'm using '4' in the queries below.

```
INSERT INTO `wordpress`.`wp_users` (`ID`, `user_login`, `user_pass`, `user_nicename`, `user_email`, `user_url`, `user_registered`, `user_activation_key`, `user_status`, `display_name`) VALUES ('4', 'demo', MD5('demo'), '<user nicename>', 'test@yourdomain.com','http://www.test.com/', '2018-04-17 00:00:00', '', '0', '<user display name>');
```

```
INSERT INTO `wordpress`.`wp_usermeta` (`umeta_id`, `user_id`, `meta_key`, `meta_value`) VALUES (NULL, '4', 'wp_capabilities', 'a:1:{s:13:"administrator";s:1:"1";}');
```

```
INSERT INTO `wordpress`.`wp_usermeta` (`umeta_id`, `user_id`, `meta_key`, `meta_value`) VALUES (NULL, '4', 'wp_user_level', '10');
```

A Web Reference: http://www.wpbeginner.com/wp-tutorials/how-to-add-an-admin-user-to-the-wordpress-database-via-mysql/

This FAQ answer the question "How to add an admin user to Wordpress using mySQL?"

Wordpress: Additional Permissions

If you navigate to <url to wordpress>/wp-admin and find that you have updates to process, you should be able to run them with the click of a button. Sometimes these will fail due to inadequate permissions granted to directories and files that the update process needs to do its job.

You can view Unix permissions for each file and directory by running ls -la. The output will resemble what is shown below:

```
drwxr-xr-x   17 peltzr  staff     578 Feb  1 14:38 .
drwxr-xr-x   46 peltzr  staff    1564 Apr 25 11:49 ..
-rw-r--r--    1 peltzr  staff     312 Feb  1 14:38 .babelrc
-rw-r--r--    1 peltzr  staff     147 Feb  1 14:38 .editorconfig
drwxr-xr-x   13 peltzr  staff     442 Feb  1 14:48 .git
-rw-r--r--    1 peltzr  staff     153 Feb  1 14:38 .gitignore
-rw-r--r--    1 peltzr  staff     197 Feb  1 14:38 .postcssrc.js
-rw-r--r--    1 peltzr  staff    6304 Feb  1 14:38 README.md
drwxr-xr-x   11 peltzr  staff     374 Feb  1 14:38 build
drwxr-xr-x    5 peltzr  staff     170 Feb  1 14:38 config
drwxr-xr-x    4 peltzr  staff     136 Feb  1 14:38 docs
-rw-r--r--    1 peltzr  staff     222 Feb  1 14:38 index.html
drwxr-xr-x  580 peltzr  staff   19720 Feb  1 14:39 node_modules
-rw-r--r--    1 peltzr  staff  289882 Feb  1 14:38 package-lock.json
-rw-r--r--    1 peltzr  staff    1698 Feb  1 14:38 package.json
drwxr-xr-x    7 peltzr  staff     238 Feb  1 14:38 src
```

Unix assigns Read (r), Write (w),and Execute (x) permissions to each object and directory objects have a (d) in front of the permission string. The rwx are grouped by owner, group, other. The **rwx** strings can be replaced by an Octal value that represents the sum of the permissions applied to an object. See this website, permissions-calculator, to see how changing the octal value changes the value of the read, write, execute properties.

The **chmod** command allows you to change the permissions on an object. If I want to enable all permission on a object I could issue the command `chmod 777 <filename` . In order to allow Wordpress to successfully update I used `chmod 775` on the following directories and files under the Wordpress installation. You should cd to /var/www/html before you execute these commands.

- chmod 775 wp-content
- chmod 775 wp-admin/includes
- chmod 775 wp-admin
- chmod 775 wp-includes
- chmod 775 wp-login.php
- chmod 775 wp-activate.php
- chmod 775 wp-signup.php
- chmod 775 wp-comments-post.php
- chmod 775 wp-comments-post.php

If you wanted to "loosen" permissions on everything - all files and directories - under a given directory the command is:

```
chmod -R 775 html
```

The -R means recursively apply the 775 to all files and directories under the html directory. For example, if you find you still can't install with the directed chmod command above, you can navigate to the directory above the server root and then set all files and directories under the server root to allow the user to write with the following commands:

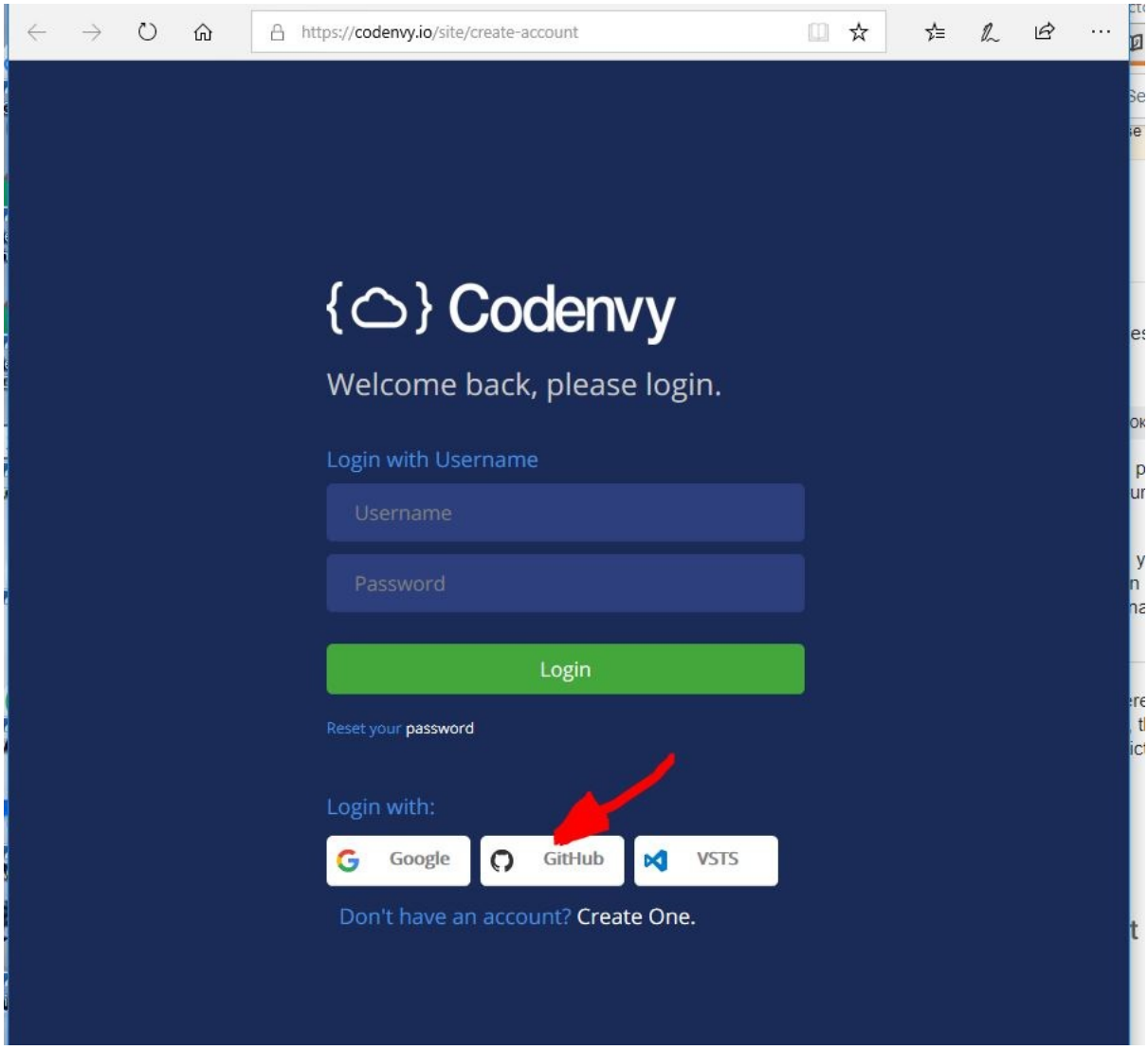
```
cd /var/www
chmod -R 775 html
```

When adding permissions to any application, such as Wordpress, you alway want to give just enough but not to much access. You can read about the Least Privilege Principle here in the Hardening Wordpress document.

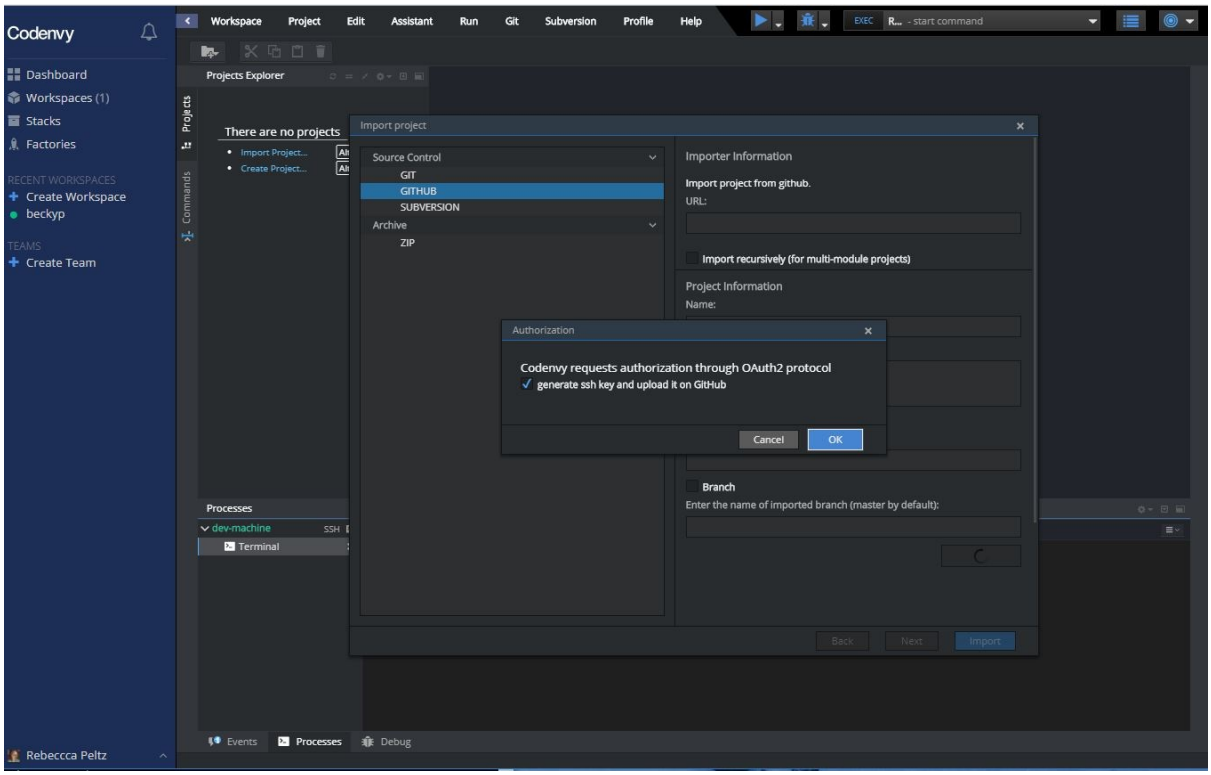
This FAQ answer the question "How to get started with Codenvy?"

Codenvy: Get Started

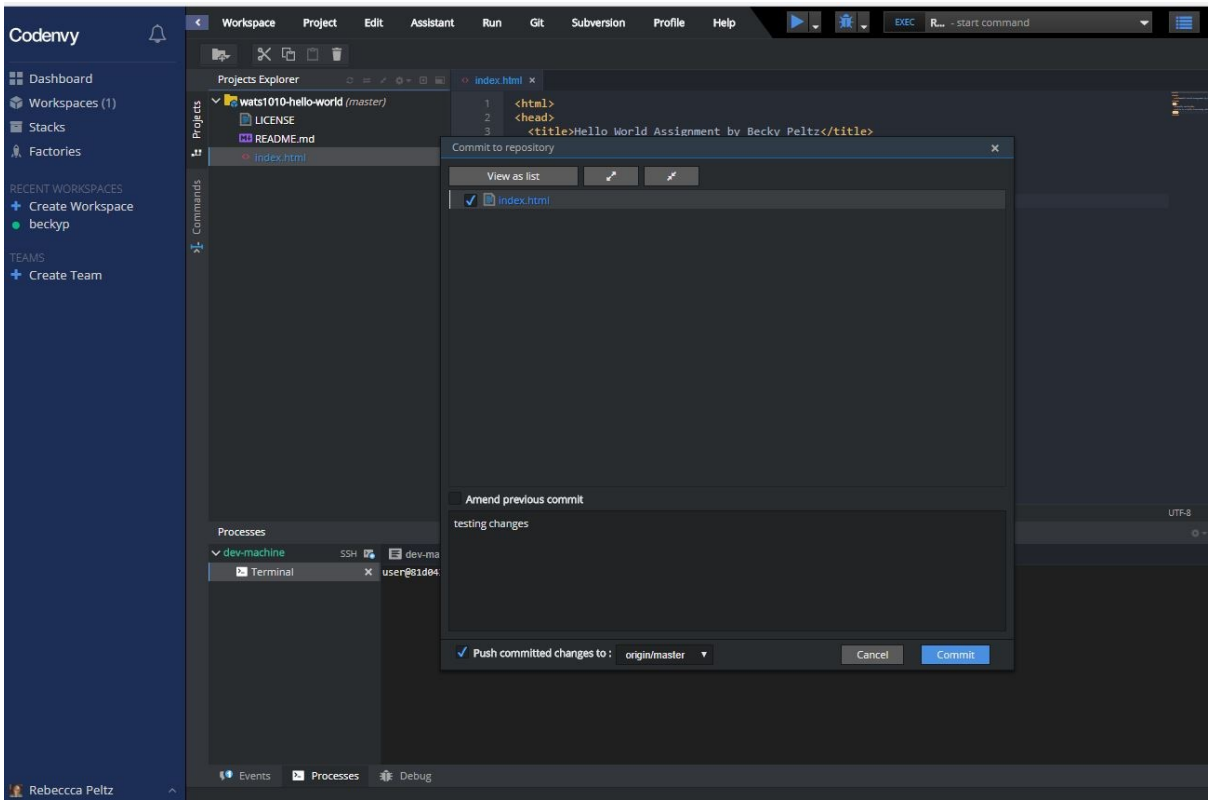
When you sign up for Codenvy, you should already have a github.com account and you want to choose your github account to Login with.



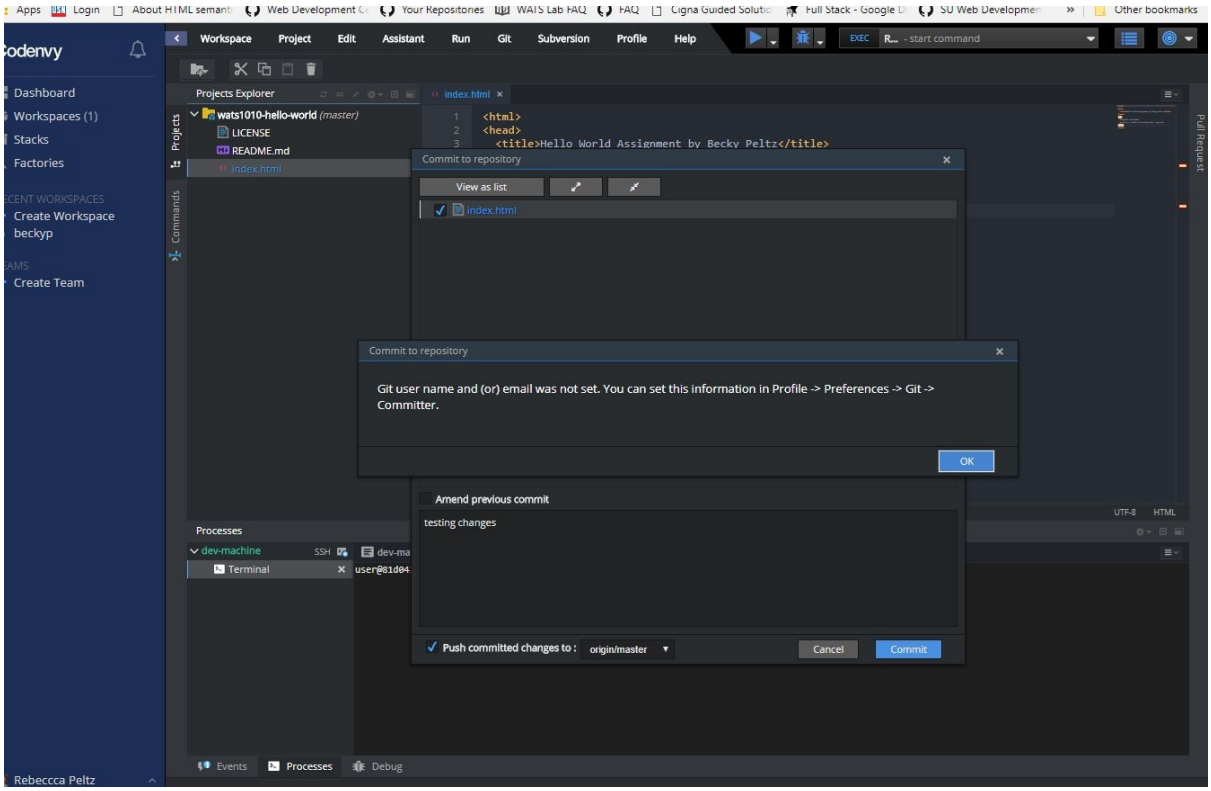
Authorizing OAuth2: This allows Codenvy to clone from and push to Github.com



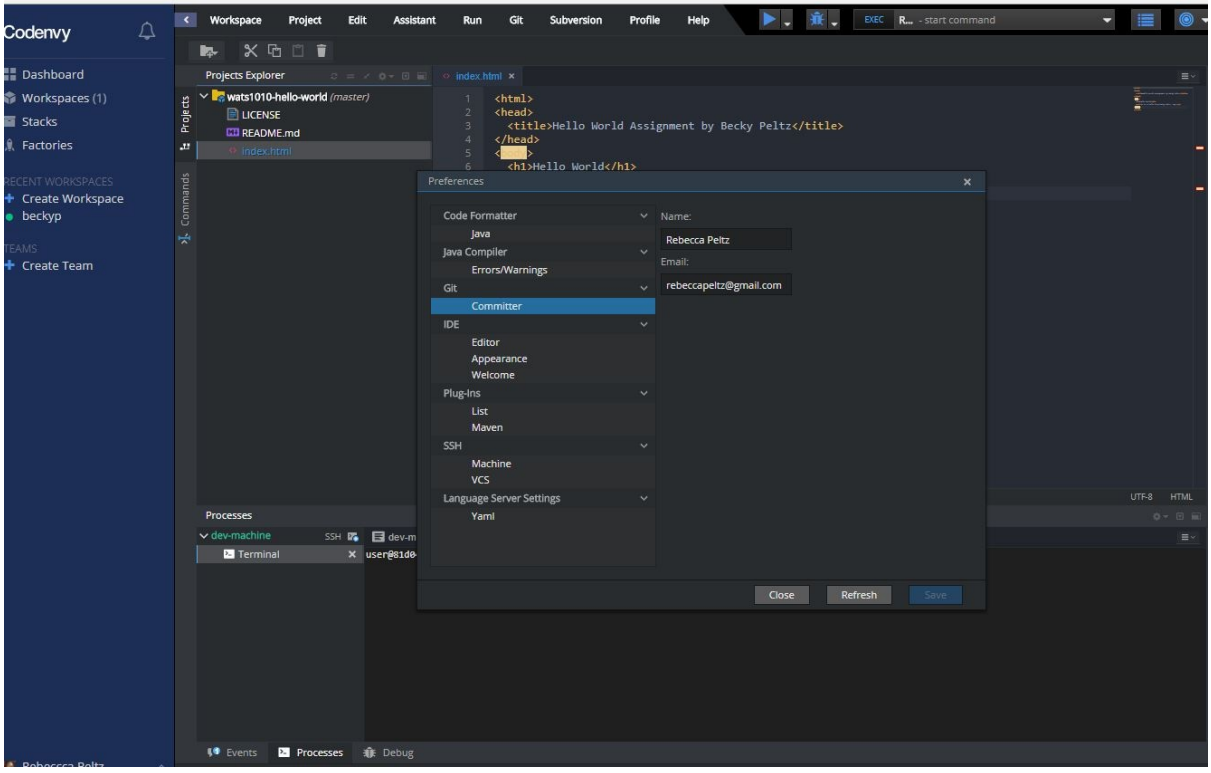
When you "push" to get be sure to check the "Push committed changes" checkbox



Directions to enter Username and Email



Adding Username and Email



This FAQ answer the question "How to add Lets Encrypt to Apache on Digital Ocean Ubuntu 16.04?"

Let's Encrypt: HTTPS on Apache

Create a non root sudoer user

"**sammy**" is just an example of a user I have created on my server.

```
sudo usermod -aG sudo sammy
```

Look Up Digital Ocean Docs

https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-14-04

Step 1: Execute steps to download and update the Let's Encrypt Client

Step 2: Execute steps to set up the SSL Certificate on just the Wordpress subdomain

I have installed Wordpress on my server and provided the domain name "wp", so I will just install the certificate on that subdomain.

```
  sudo certbot --apache -d wp.beckypeltz.online
```

I chose the redirect options. This means if the user types http://wp.beckypeltz.online, into the browser, I'll redirect to https://wp.beckypeltz.online. See the image below of this choice.

```
sammy@lamp-s-1vcpu-1gb-sfo2-01:~$ sudo certbot --apache -d wp.beckypeltz.online
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Starting new HTTPS connection (1): acme-v01.api.letsencrypt.org
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for wp.beckypeltz.online
Waiting for verification...
Cleaning up challenges
Created an SSL vhost at /etc/apache2/sites-available/wp-le-ssl.conf
Enabled Apache socache_shmcb module
Enabled Apache ssl module
Deploying Certificate to VirtualHost /etc/apache2/sites-available/wp-le-ssl.conf
Enabling available site: /etc/apache2/sites-available/wp-le-ssl.conf

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
-------------------------------------------------------------------------------
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
-------------------------------------------------------------------------------
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
```

This FAQ answers the question "How to add Bootstrap 4 to Vue.js project?"

Bootstrap 4 and Vue.js

This document describes two ways to add Bootstrap 4 to a Vue.js project. The first way is linking to CDN's in the index.html. The second way is to use Vue-Bootstrap which will load into the apps main.js like a component. If you use the CDN method, you can use standard bootstrap class names provided by the Bootstrap documentation.

Method 1: Linking to CDN's

Linking to CDN's in a Vue.js project is similar to linking to them in any HTML5 document. You'll follow the directions on the Bootstrap home page to add them to your **index.html** in the root of the application code. There is one css link and three JavaScript links. Bootstrap 4 requires jquery and popper.js.

CSS

Copy-paste the stylesheet <link> into your <head> before all other stylesheets to load our CSS.

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.cs
```

Copy

JS

Many of our components require the use of JavaScript to function. Specifically, they require jQuery, Popper.js, and our own JavaScript plugins. Place the following <script>s near the end of your pages, right before the closing </body> tag, to enable them. jQuery must come first, then Popper.js, and then our JavaScript plugins.

We use jQuery's slim build, but the full version is also supported.

```
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3U
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha3
```

Copy

The code in your index.html will look something like this after you retrieve the links from the BS 4 homepage.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1.0">
  <title>Hikes</title>
  <!-- Latest compiled and minified CSS -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity
="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
    crossorigin="anonymous">
  <link rel="icon" type="image/png" href="static/images/backpacker.png">
  <link href="https://fonts.googleapis.com/css?family=Ubuntu" rel="stylesheet">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.2.0/css/all.css" integrity="sha384-hWVjf
lwFxL6sNzntih27bfxkr27PmbbK/iSvJ+a4+0owXq79v+lsFkW54bOGbiDQ" crossorigin="anonymous">
</head>
<body>
  <div id="app"></div>
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/
rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
```

```
    crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApN
bgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
    crossorigin="anonymous"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh
4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
    crossorigin="anonymous"></script>
</body>

</html>
```

There is a sample project deployed on github that uses the CDN approach.

Method 2: Using Vue-Bootstrap

To use vue-bootstrap, you start by installing it from npm. The Vue-Bootstrap documentation provides these install instructions. You'll enter this npm command into your terminal. Use the save option to record the package in your package.json file, so that future users of your code will pick it up when they `npm install`.

```
npm i bootstrap-vue --save
```

You'll notice that the bootstrap-vue install added both bootstrap and bootstrap-vue to your node-modules directory.

In your main.js file, add the following code to register the functionality provided by bootstrap as a Vue Component and make the CSS available.

```
import Vue from 'vue'
import BootstrapVue from 'bootstrap-vue'
import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'

Vue.use(BootstrapVue);
```

Here's the code in a sample application using Bootstrap Vue.Notice that there is no reference to Bootstrap or jquery in the index.html.

index.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1.0">
  <title>Hikes</title>
  <link href="https://fonts.googleapis.com/css?family=Ubuntu" rel="stylesheet">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.2.0/css/all.css" integrity="sha384-hWVjf
lwFxL6sNzntih27bfxkr27PmbbK/iSvJ+a4+0owXq79v+lsFkW54bOGbiDQ" crossorigin="anonymous">
</head>
<body>
  <div id="app"></div>
</body>

</html>
```

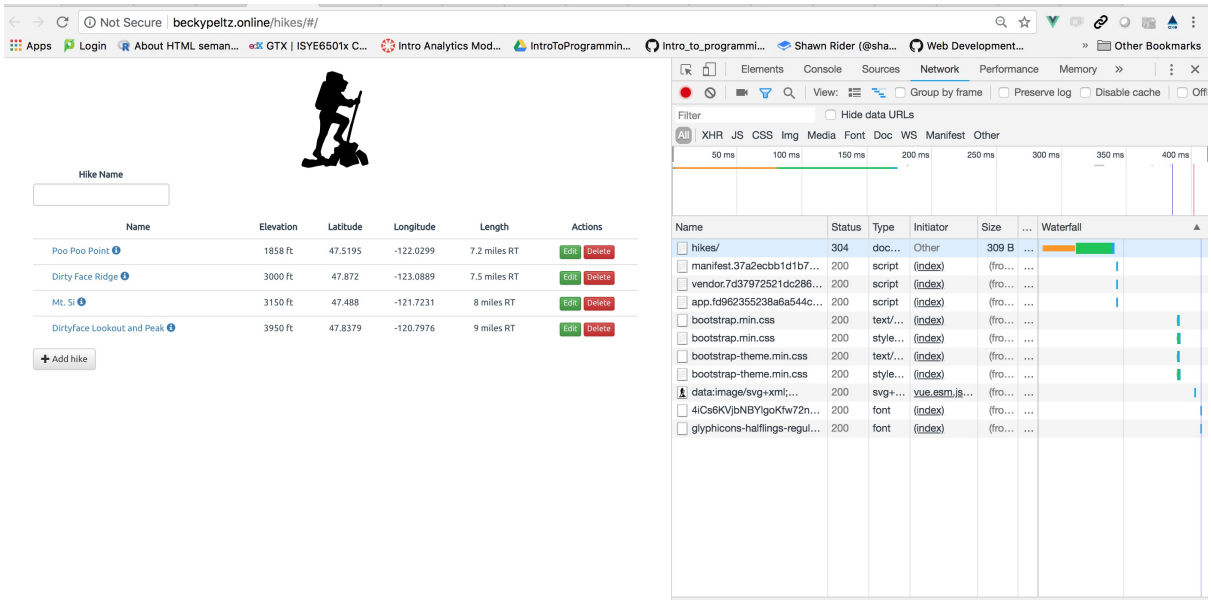
main.js

```
// The Vue build version to load with the `import` command
// (runtime-only or standalone) has been set in webpack.base.conf with an alias.
```

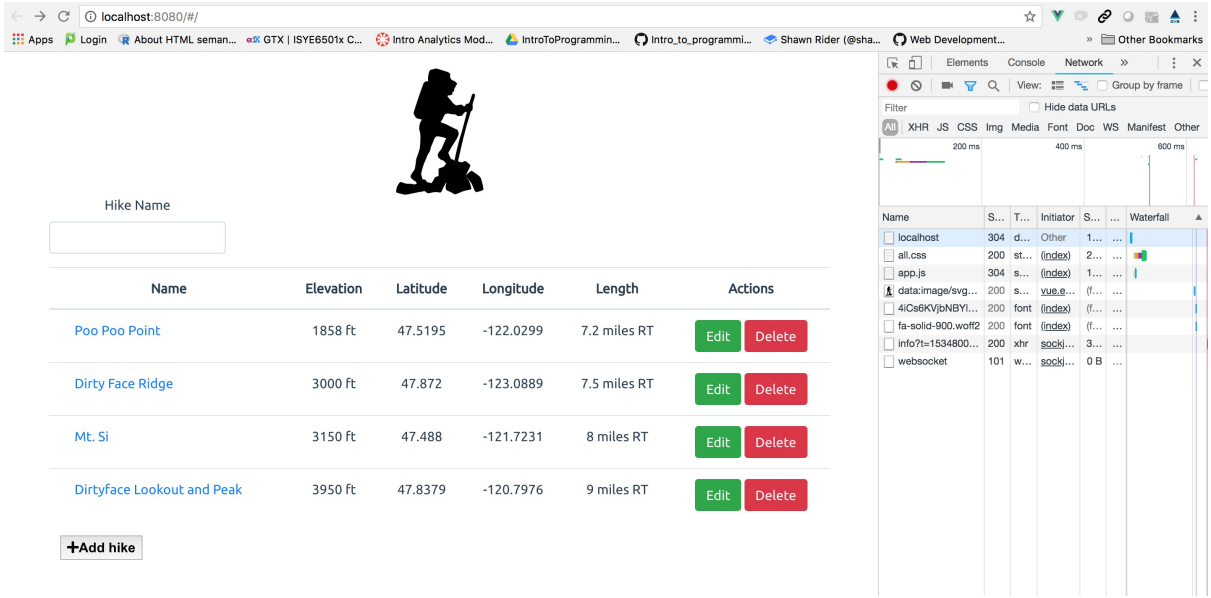
```
import Vue from 'vue'
import App from './App'
import router from './router'
import BootstrapVue from 'bootstrap-vue'

import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'

Vue.use(BootstrapVue);

Vue.config.productionTip = false

/* eslint-disable no-new */
new Vue({
  el: '#app',
  router,
  components: { App },
  template: '<App/>'
})
```

Network Downloads: What to Expect from each Method

The picture below shows the network downloads for the 1st method using CDNs. Note that the bootstrap files can be seen as downloads.



In the picture below where Vue-Bootstrap is used on a branch from the code above, you won't see the bootstrap files in the network display.


localhost:8080/#/
Apps
Login
About HTML seman...
GTX | ISYE6501x C...
Intro Analytics Mod...
IntroToProgrammin...
Intro_to_programmi...
Shawn Rider (@sha...
Web Development...
Other Bookmarks
Hike Name
Name
Elevation
Latitude
Longitude
Length
Actions
Poo Poo Point
1858 ft
47.5195
-122.0299
7.2 miles RT
Edit
Delete
Dirty Face Ridge
3000 ft
47.872
-123.0889
7.5 miles RT
Edit
Delete
Mt. Si
3150 ft
47.488
-121.7231
8 miles RT
Edit
Delete
Dirtyface Lookout and Peak
3950 ft
47.8379
-120.7976
9 miles RT
Edit
Delete
+Add hike
Elements
Console
Network
View:
Group by frame
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
200 ms
400 ms
600 ms
Name
Initiator
Waterfall
localhost
all.css
app.js
data:image/svg...
4iCs6KVjbNBYl...
fa-solid-900.woff2
info?t=1534800...
websocket

This FAQ answers the question "How to use the Git Command Line?"

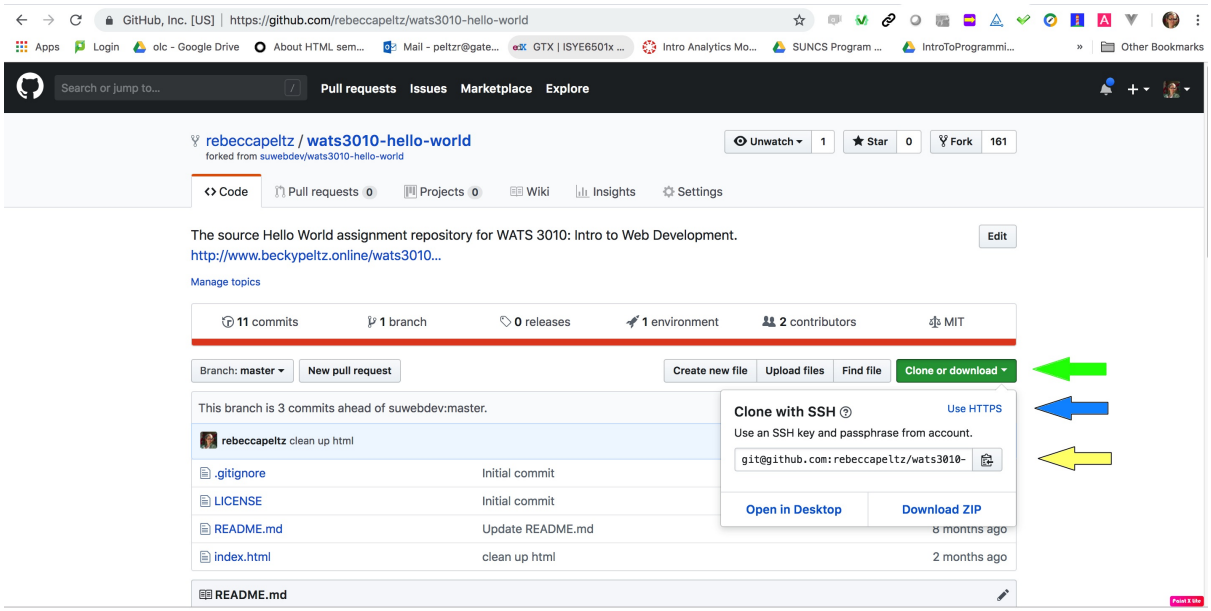
Git Command Line

This is not an all inclusive document on how to use the Git Command Line. This a quick reference on the command to clone a repo from Github.com and then push changes back to Github.com.

I'm am using a repo, **wats3010-hello-world** from my account, **rebeccapeltz**, on Github.com to demonstrate these commands.

Copy the ssh git address of your repo into your clipboard

The picture below shows that I have clicked on the green button (see green arrow in picture) labeled **Clone or Download**. I have also clicked on the link labeled "**Use SSH**" (see blue arrow in the picture). I see an address **git@github.com:rebeccapeltz/wats3010-** and I have clicked on the **copy icon** to copy this address into my clipboard (see yellow arrow in the picture).



Clone command

You can open a command line window in visual studio code or using terminal (Mac) or git bash (Windows with Git installed). From the command line you can navigate to your projects directory (a directory you have created under your home directory) and then issue the clone command by typing clone and then pasting the git address onto the command line.

```
cd ~/projects
git clone git@github.com:rebeccapeltz/wats3010-hello-world.git
```

You will see output showing that the code from github.com is being copied down to your local machine.

Push Changes back to Github

After you have made changes to your code and tested to see that they are working as you expect, you will issue the following commands in the root of your project to push the changes back to github. Lines with a # at the start are comments. You only execute the command that start with git below.

```
# optional check to see the file you have changed they will appear in red
git status
# add the file to the local git repo - you can name each file or use . to specify all files below current
# directory
git add .
# if you issue git status now the files added will be green
# commit the files to the directory and provide a message describing the changes
git commit -m"my changes for this commit"
# if you issue git status now you'll see a notice that you have files to push
# push the files to github.com repo
git push origin master
```

You might want to commit to memory

```
git add .
git commit -m"my changes"
git push origin master
```

Default Editor for Git

If you forget to enter a message for your commit, you'll find that an editor will open. This default editor is VIM which may be unfamiliar to you. For that reason, it a good idea to configure an editor you're familiar with, which would be Visual Studio Code. You only need to do this once.

To configure Visual studio code, make sure that you have this program in your path. You can test this.

```
code --help
```

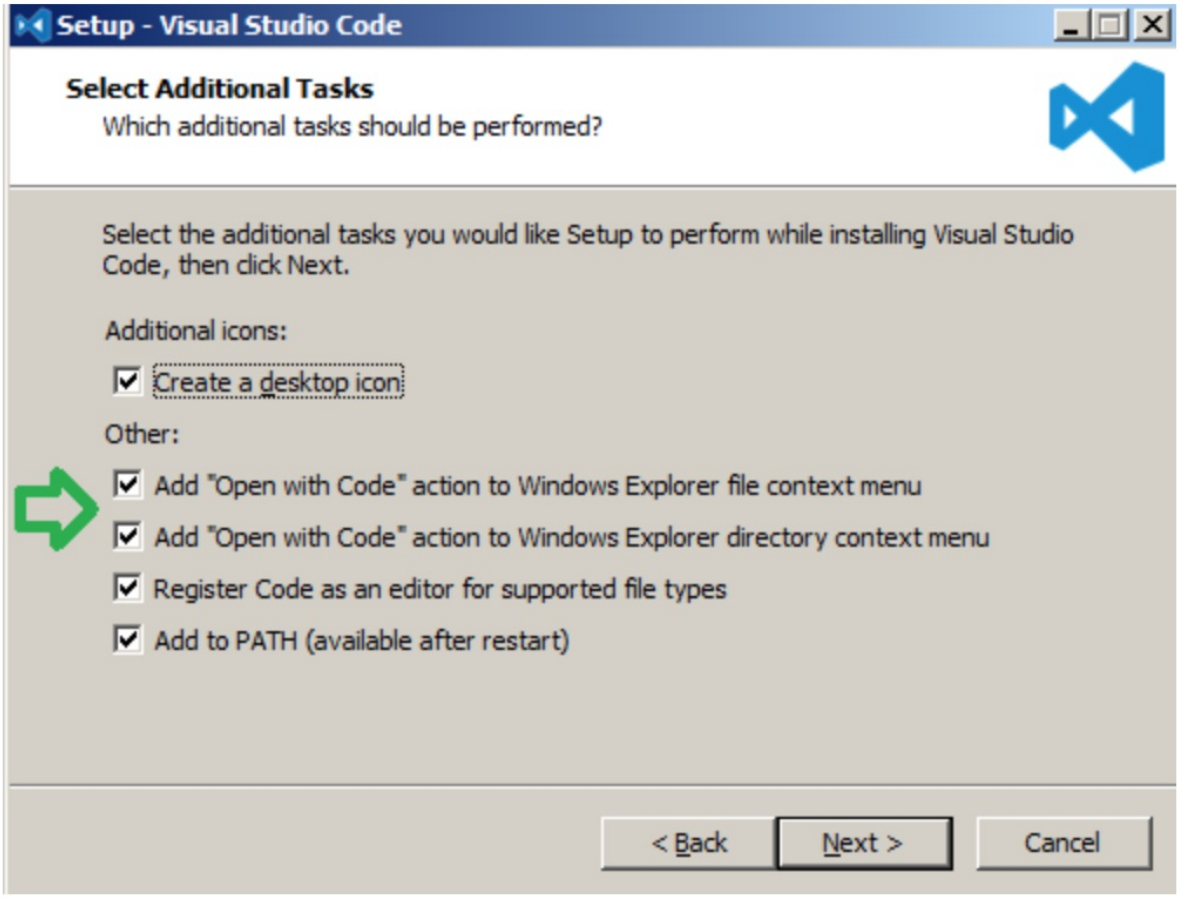
which should show you help output. If it's not in the path you can add it to the path

Mac Install Code in Path

In VS Code enter **CTRL-SHIFT-P** to open the Command Pallete and type in **Shell Command: Install 'Code' Command in command path**

Windows Install Code in Path

You can check a box during installation to add Code to Windows path. If you didn't do this you can rerun the installation.



Set VS Code as the default editor for Git

Once you've verified that VS code is in the path, you can execute the following command to make it the default editor for Git.

```
git config --global core.editor "code --wait"
```

You can test that VS Code is your default editor for git by typing the following command. This will open your machine's **~/.gitconfig** file in VS Code. This is a hidden file under your user root that contains all the git configuration settings.

```
git config --global -e
```

This FAQ answers the question "How to create a Digital Ocean droplet and provide SSH connection to Github?"

Overall Process

Providing an SSH Connection to Github can be done either before or after creating the Digital Ocean (DO) droplet. It turns out that it's easier to set up an SSH key in DO before creating the droplet than to manually add the SSH key to the DO droplet after creating it. For that reason, I'll describe setting up the SSH Connection before describing setting up the droplet.

Provide SSH Connection to Github

Setup SSH Keys for the Account

Because the application that runs to setup a droplet prompts the user for an existing SSH key (or keys), it makes sense to set these keys up before attempting to create the droplet.

1. Start by creating SSH Keys on you local machine (See the FAQ on Install Git Locally)
2. Copy the public SSH Key into your buffer `cat ~/.ssh/id_rsa.pub` and select and copy.
3. Open Digital Ocean in your browser and navigate to Account | Security. Then use the Add SSH Key form to add the public key to Digital Ocean.

The net effect of having an SSH Key uploaded to you DO account is that you will see it offered up when you are creating your droplet.

Upload SSH Keys to Digital Ocean Account

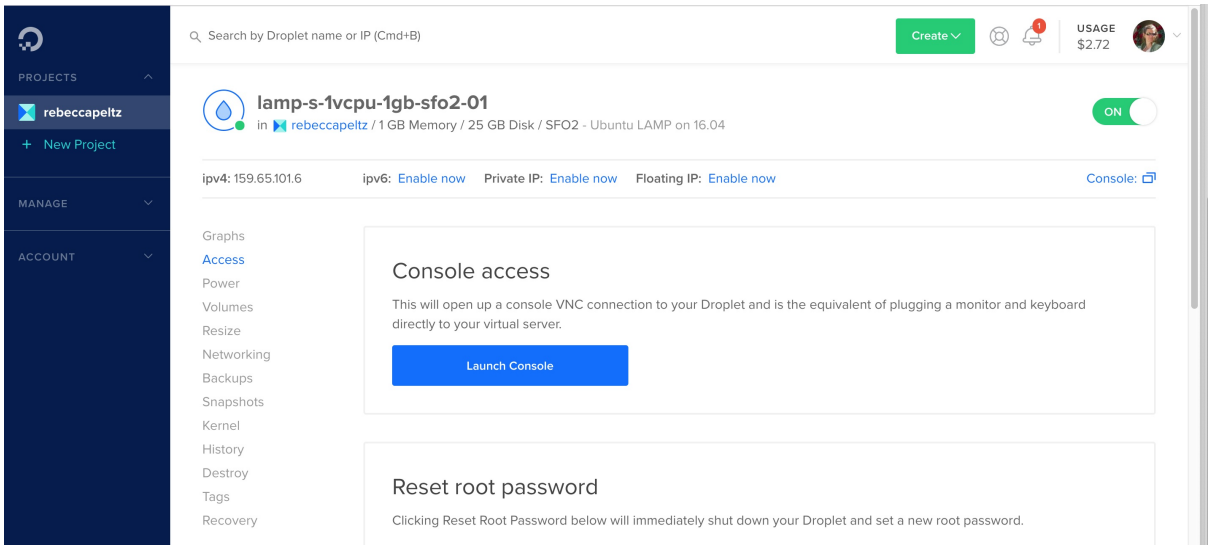
How to Add SSH Keys

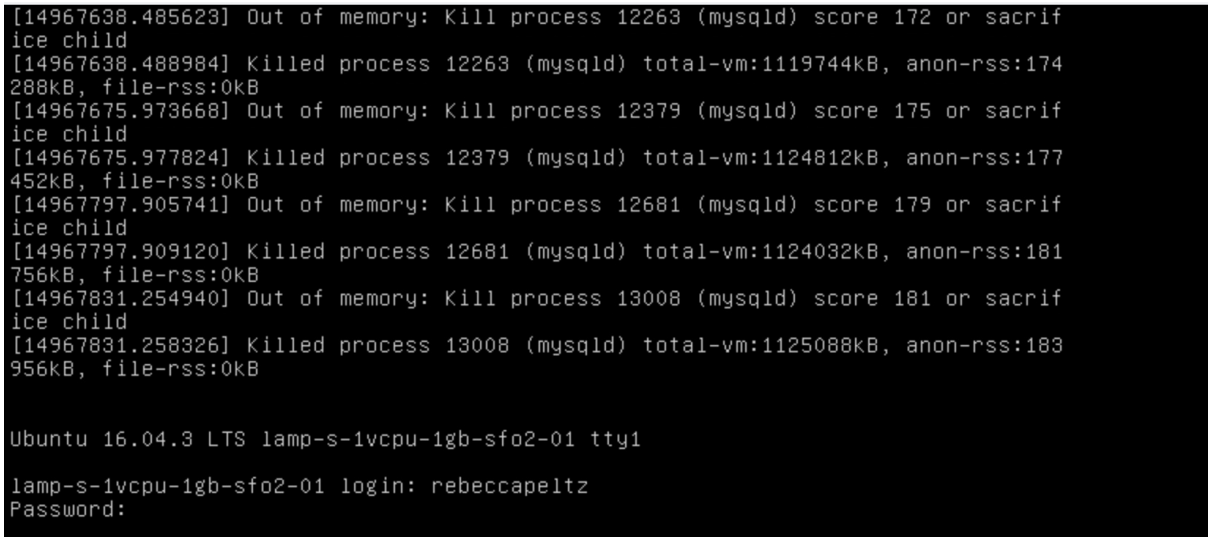
How to Add an SSH Key Manually to an Existing Droplet

If you already have a droplet and you want to add an SSH key, you will need to do it manually.

1. On you local machine copy the key into the buffer `cat ~/.ssh/id_rsa.pub` and select and copy.
2. Launch the console from Digital Ocean dashboard and login using your DO login credentials.
3. Use `ls -la ~/.ssh/authorized_keys` to determine of the authorized *keys files exists. If it doesn't create it with* `touch ~/.ssh/authorized_keys`
4. Use nano to edit the authorized_keys file (nano has copy/paste) and paste the SSH key from your local machine into the authorized keys file and save.
5. Set permissions on the `authorized_keys` file to make it accessible

```
chmod -R go= ~/.ssh
chown -R $USER:$USER ~/.ssh
```





[Upload keys on an existing droplet](https://www.digitalocean.com/docs/droplets/how-to/add-ssh-keys/to-existing-droplet

Create a Digital Ocean Droplet

Follow the step for creating a droplet in the link below. Choose an Ubuntu images/1 GB memory/25 GB disk/San Francisco for your data center as it's the closest to Seattle. You want the simplest images for your server as possible and it should only cost $5/month. You don't need a back up or block storage. If you uploaded an SSH key you should see it available when under "Add SSH key" and you should select it. Then just click on create. If you make a mistake just delete the image from the console and try it again. Don't get attached to your server, especially when there's nothing on it. Make note of the IP Address.

Once the image it created, go to your local machine and login using ssh. Here's an example if your IP address is `203.0.113.0`

```
ssh root@203.0.113.0
```

If your using a Mac you can do this from the terminal and if you're using Windows, I recommend doing it from Git Bash.

Create droplet

Connect with SSH

This FAQ answers the question "How to migrate Vue 2 code from Vue CLI 2 to Vue CLI 3?"

Migrate Vue 2 Code From Vue CLI 2 to Vue CLI 3

Vue.js has rearchitected the way that you build the Vue.js 2 code. It's important to discern the difference between the way that you code Vue.js and the way that you build Vue.js. Vue.js released an new version of the CLI (the command line interface for building Vue.js code in to Vanilla JavaScript and CSS) in 2017. The versioning moved from 2 to 3.

It is not necessary to upgrade the Vue.js 2 code, but if you want to continue to develop in Vue.js you should upgrade the CLI to version 3. There are a number of important architecture changes to the way that a project is configured and architected in CLI 3. It is not necessary to modify your code to make this move. The changes required involved moving new configuration files and changes to the file structure (where your code is stored).

This document outlines a process to migrate your code repo and add new config files so that it can take advantage of the CLI 3.

Upgrade Node

The first step should be to make sure you're using the latest version of node and npm. This can be done by executing the global install command. This command will update node and npm.

`npm install -g node`

If you're on a Mac, depending on how you installed node, you made need to use the `sudo` command for permission to do this upgrade.

`sudo npm install -g node`

Its always a good idea to check your upgrade by checking versions.

```
npm --version
node --version
```

Upgrade Vue CLI

To upgrade Vue CLI you can follow instructions on this page: https://cli.vuejs.org/guide/installation.html

If you have Vue 2 CLI installed you need to uninstall it.

```
npm uninstall vue-cli -g
```

The command to install CLI 3 is

```
npm install -g @vue/cli
```

Then verify the version

```
vue --version
```

As of this writing, the current version is **3.3.0**, but there is a lot of development taking place with Vue.js and this may not be the version that you get. Your major version (the first number should at least be a **3**).

Build and Dev

CLI 3 provides a production and dev build just as CLI 2 did but the command to run the dev build and server has changed. To run the dev build use the following command:

```
npm run serve
```

To run the production build, which should create runnable html/css/js in the docs directory:

```
npm run build
```

Migrate Code

Branch Existing Code

Do the migrate on a branch and then when it's working properly, merge to the Master branch. For example (below), create a branch named cli3 and check it out to your local machine to work on.

```
git checkout -b cli3
```

The new file structure that CLI 3 is looking for is shown in the picture below. Your goal will be to migrate your file structure to this new structure. Notice that the config and build directories are gone. The node_modules is listed in **.gitignore** so it the name appears faded out in VS Code. There are some new **.js** files used for configuration.

The changes to look for in this picture are:

- the router code is in a router.js file in the root instead of router/index.js
- there is a new `views` directory - in CLI 3 the best practice is to put components referenced in the router into the views directory and use the `components` directory for non-view components
- there is a `.gitkeep` file in the `components` directory and the `views` directory
- there some new config files - this document will provide the contents for these files, in particular `.babel.config.js` , `postcssrc.js`
- there a couple of config files that I created specifically to make migration easier to put production build code in the `docs` directory that you'll add `aliases.config.js` and `vue.config.js`
- the `package.json` library dependencies has changed significantly and you'll want to replace the entire content of `package.json` with code provided in this document
- the `static` folder has been renamed to `public` and the index.html has moved into the `public` folder

NOTE: It's possible that the config code provided in this document may change. You can always generate the lates config code by running the new project `create` command to create a new project that will have all of the latest config code. The project create command will not create the vue.config.js or the aliases.config.js as those are user created

and I created them to allow for the use of the `@` symbol to specify `src` and to make the build create distributable files in the `docs` folder so that we can host on `github.io`.

To create a new project in Vue CLI 3 you can run the command below which will create a project called `hello-world`. Notice that the keyword has changed froom `init` to `create`. You should also pick the default babel/eslint.

```
vue create hello-world
```

See this page for instructions on creating a new project and note special instructions for Windows users that are using the `git bash` terminal: https://cli.vuejs.org/guide/creating-a-project.html#vue-create.

The purposed of these migration instructions is that you shouldn't have to create a new project - you should be able to migrate the code by just adding config files and modifying the file structure.

Migration Steps

delete `config` and `build` directories

delete `babel.rc`

rename `static` to `public`

move `index.html` (and any other static contents) into `public`

create an `aliases.config.js` file and load it with the contents specified in this document

create an `babel.config.js` file and load it with the contents specified in this document

create an `vue.config.js` file and load it with the contents specified in this document

delete `docs` directory as it will be recreated when you run the build

delete `package-lock.json` file - this file will get automatically recreated when you run `npm install`

replace the contents of `package.json` with the code contents specified in this document

create a `router.js` file in the `src` of the project and move the contents of router/index.js into this file

create a `views` directory and move any files in the `components` directory that are reference in the router.js into the `views` directory

add an empty .gitkeep file to the `views` and `components` directory (this is to keep them around even if empty)

update links in `router.js` to point to files in the `views` directory

delete `node_module` and `npm install` to get new ones

test code build by running `npm run serve`

build production code into docs by running `npm run build`

push branch to github `git push -set -upstream origin <branch name>`

you should see your branch and master when you run `git branch`

merge to master by checking out master locally and running merge

```
git checkout master
git merge <branch name>
```

add/commit/push migrated code to github and test on github.io

Once you're done merging you can delete the branch. It's good practice for cleanup. You'll delete it locally and remotely.

Local: `git branch -d <branch name>`

Remote: `git push origin --delete <branch-name>`

Contents of Config Files

babel.config.js

```
module.exports = {
  presets: [
    '@vue/app'
  ]
}
```

postcssrc.js

```
module.exports = {
  "plugins": {
    "postcss-import": {},
    "postcss-url": {},
    // to edit target browsers: use "browserslist" field in package.json
    "autoprefixer": {}
  }
}
```

aliases.config.js

```
const path = require('path')
function resolveSrc(_path) {
  return path.join(__dirname, _path)
}
const aliases = {
  '@': 'src',
  '@src': 'src'
}
module.exports = {
  webpack: {},
  jest: {}
}
for (const alias in aliases) {
  module.exports.webpack[alias] = resolveSrc(aliases[alias])
  module.exports.jest['^' + alias + '/(.*)$'] =
    '<rootDir>/' + aliases[alias] + '/$1'
}
```

vue.config.js

```
const path = require('path');
module.exports = {
    configureWebpack: {
        resolve: {
            //allow for @ or @src alias for src
            alias: require('./aliases.config').webpack
```

```
        }
    },
    chainWebpack: config => {
        //turn off elint for webpack transpile
        config.module.rules.delete('eslint');
    },
    runtimeCompiler: true,
    css: {
        sourceMap: true
    },
    publicPath: '',
    //build for docs folder to enable gh-pages hosting
    outputDir: './docs/',
    assetsDir: 'assets'
}
```

package.json

```
{
  "name": "hello-world",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "axios": "^0.18.0",
    "vue": "^2.5.21",
    "vue-router": "^3.0.2",
    "vue2-animate": "^2.1.0"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "^3.3.0",
    "@vue/cli-plugin-eslint": "^3.3.0",
    "@vue/cli-service": "^3.3.0",
    "babel-eslint": "^10.0.1",
    "eslint": "^5.8.0",
    "eslint-plugin-vue": "^5.0.0",
    "vue-template-compiler": "^2.5.21"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/essential",
      "eslint:recommended"
    ],
    "rules": {},
    "parserOptions": {
      "parser": "babel-eslint"
    }
  },
  "postcss": {
    "plugins": {
      "autoprefixer": {}
    }
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not ie <= 8"
  ]
}
```