GitHub Stage, Commit, and Push using VSC

Stage, commit, and push your changes to GitHub

TL;DR: Source

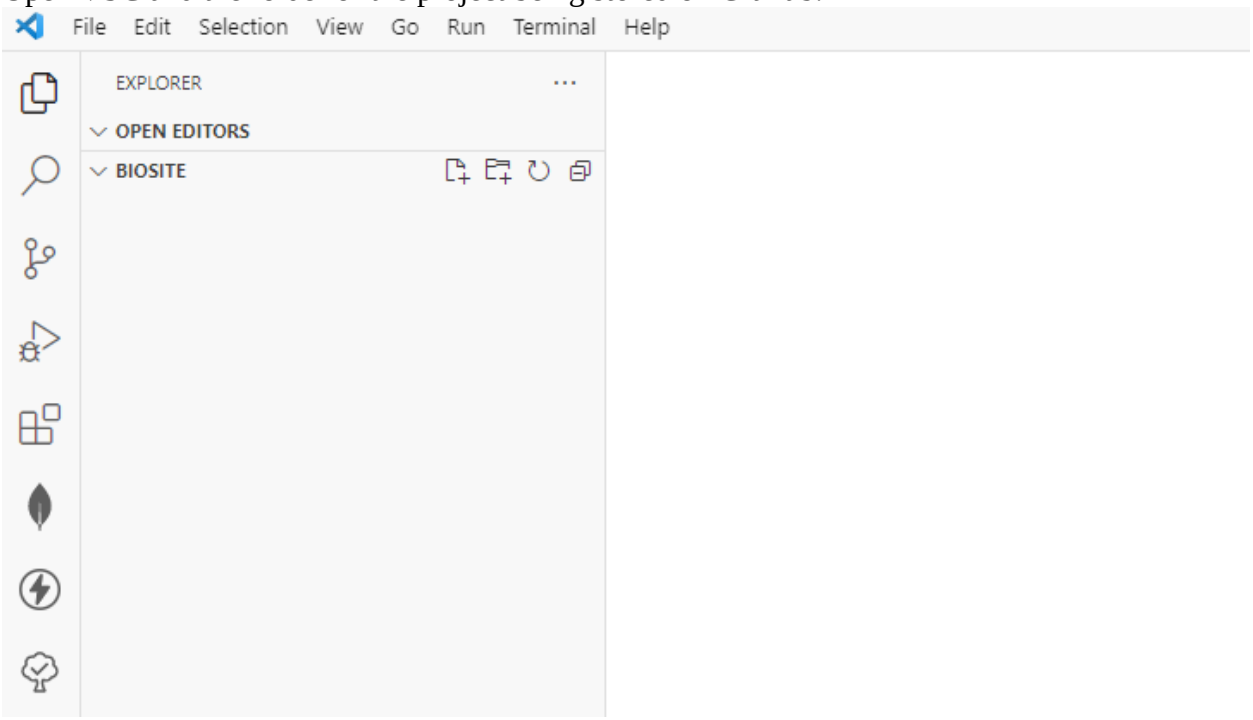
Anytime you make changes to a file in a project that is under version control (a.k.a., saved on GitHub), those changes are being flagged/tracked. Changes encompass, modifying an existing file, adding a new file to the project, or removing a file from the project. The first step in preparing your code to be shipped to GitHub is referred to as **stagging**. Stagging is the process of marking files that have been modified in your project folder.

The following steps assume you have already created a repository in GitHub and cloned it. If you skipped these steps, you will need to circle back before continuing with this guide.

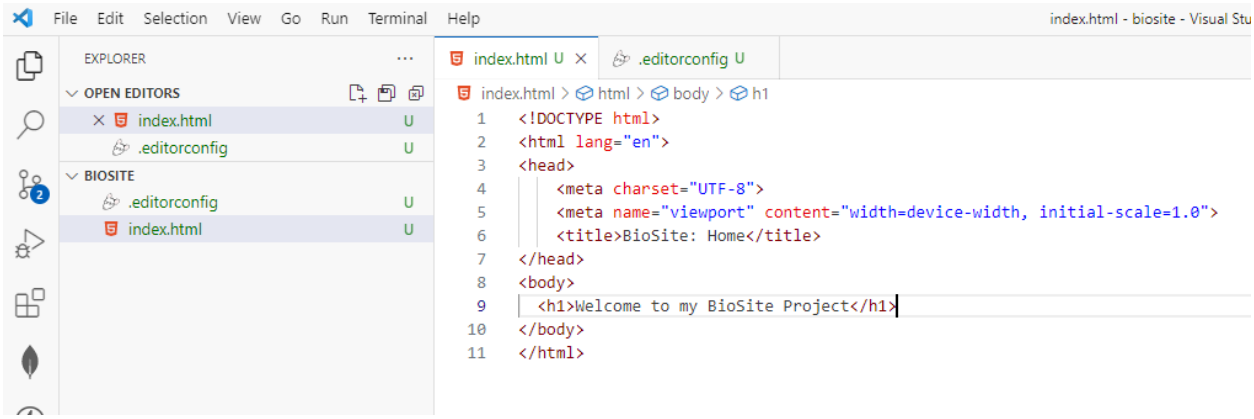
Something to consider: How do I know a project folder is under version control? There should be a file inside the project folder named .git. By default, most Operating Systems hide files that start with a period. So, to check if a project folder has a .git file, you will need to **show hidden files** on your computer. If a .git file is present, then the project is under version control.

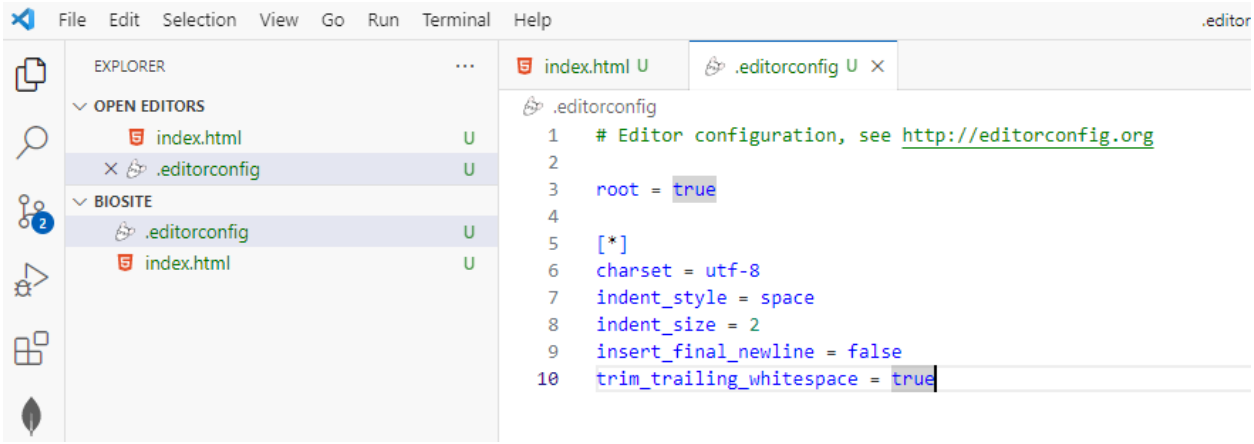
What is a .git file? A .git file is a configuration/settings file that git uses to manage your repository.

Open VSC and the folder of the project being stored on GitHub.

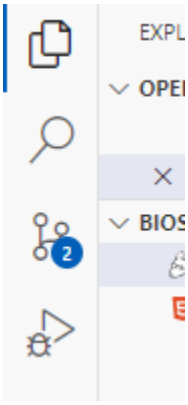


As you can see from the image, the project folder is empty. Let's start by adding an .editorconfig and HTML file to the project.

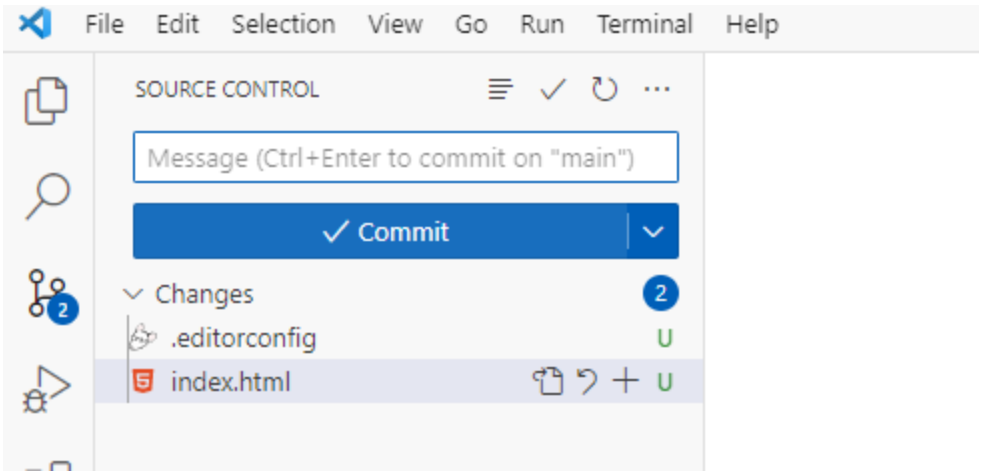




In the left-hand menu pane, you will notice there is an icon that looks like branches and in my case, it has a number two underneath it. This number represents how many changes I have pending. This means, there are two changes in this project folder that have not been added to GitHub.



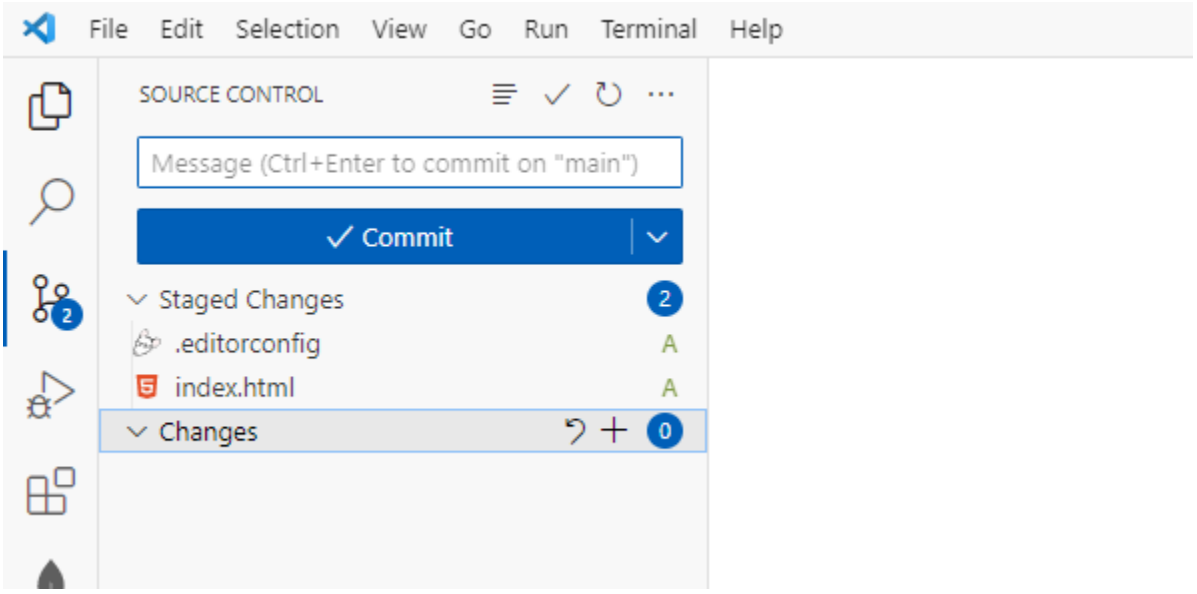
Clicking on the icon reveals the two files under the **Changes** menu.



The green letter U to the right of each file indicates the file is **Untracked**. And, if you remember, from my previous explanation, this means the file has not been staged yet. So, the first step is to stage the files so they are no longer flagged as untracked.

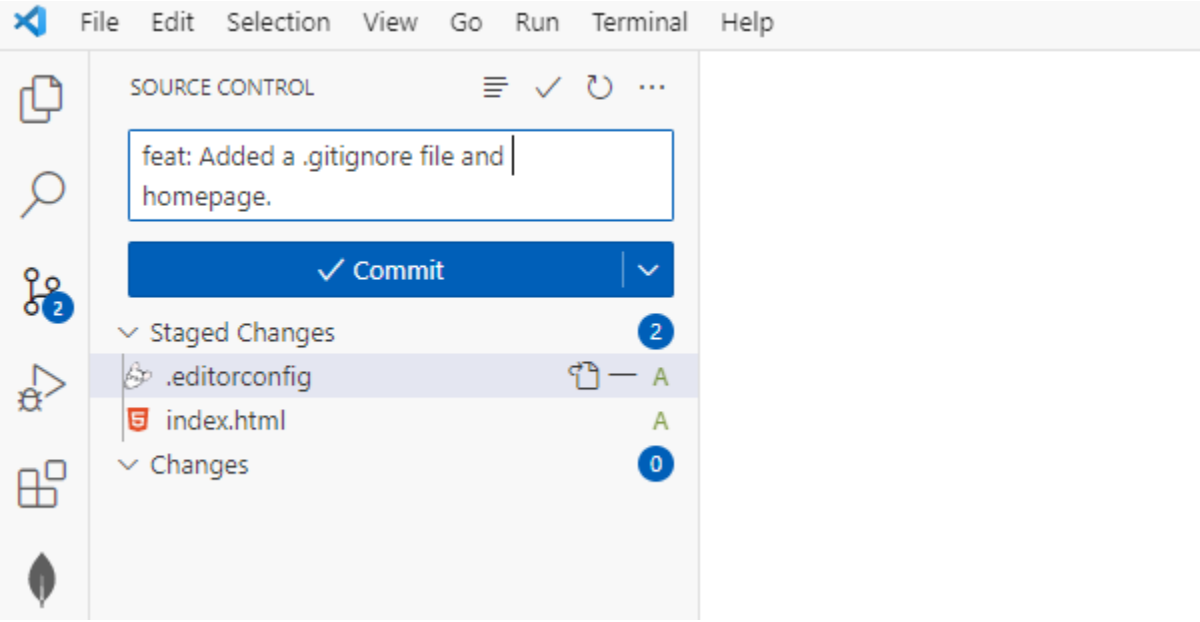
Something to consider: Did you know, if you click on the file in this tab, it will open a new window where it shows version history. The window to the left is the previous version and the window to the right is the new version. This is very helpful when you want to see what changed in a file before sending it to GitHub.

Next to each file you will see a series of icons: clipboard, curved arrow, and a plus sign. The plus sign is for stagging the file. Let's go ahead and click on the plus file for each file.



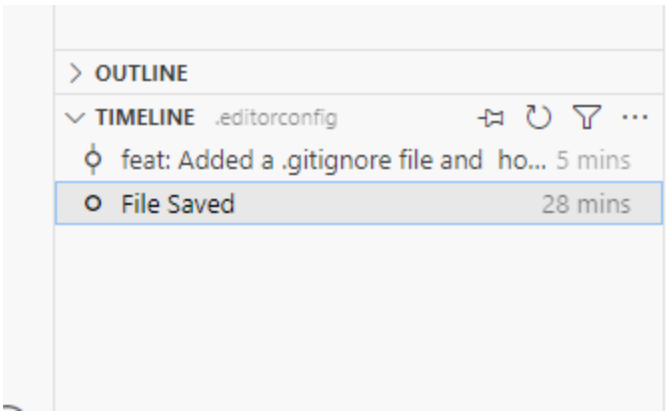
Notice now the files moved from the **Changes** section and are now placed under a section named **Staged Changes**. Also, the **U** icon has turned to an **A** icon. These files are now flagged to be added to GitHub. Meaning, the next time you commit these files will be added to GitHub. You can unstage a file by selecting the minus icon.

The next step is to commit the staged changes. To comment your stagged changes, type a commit message in the upper text box and select the **Commit** button.



Something to consider: Why are you using **feat: {description}** for the commit message? I am using what's called **Convention Commits** it is a specification for adding human and machine-readable meanings to commit messages. It is very consice and easy to read. Addition, it provides a standardized approach for creating commit messages. Is this a requirement in the Web Development program (i.e., am I required to follow this approach)? The answer is no, but it is a good practice and used heavily in open-source projects. So, unless you already have a specific way you like adding commit messages, which is highly unlikely, since this is probably your first-time using git and GitHub. It would not hurt to follow it.

Clicking on the **Commit** button saves your changes to the local Git repository, allowing you to revert to a previous version if needed (this does not send the code to GitHub). You can review all of the commits you have locally by expanding the timeline section from the **Project Folder** view. To do this, simply click on the Project Folder icon and open one of the files you just stagged and committed. Next, expand the timeline section at the bottom of the Explorer.



As you can see from the image, our commit is shown with the message we left. Once you have made commits to your local Git repository and you are ready to push them to GitHub, open the **Source Control** icon and select the ellipse. Next, select **Push**.

Something to consider: VSC might open a dialog window at the bottom of the editor asking if **Would you like Visual Studio Code to periodically run "git fetch?"** What its asking is you is if you would like VSC to always check for changes to your GitHub repository that have been pushed outside of your solution. This is really only needed when you are working on a team with other developers. For now, select **No**.

The final step is to check if the code was actually updated in GitHub. To do this, sign into your GitHub account and open the repository you created. If the stage, commit, and push were successful, the files should appear with the message we used during the commit.