**Submodules**

Submodules are essentially Git repositories within a Git repository.

If you've already executed **git add .** and you're still seeing untracked content in the status, it's possible that these files are located in submodules.

1. **Navigate into Each Submodule:** For each submodule showing untracked content, navigate into the submodule directory:

Code:

cd path/to/submodule

Replace "path/to/submodule" with the actual path to each submodule.

1. **Check Status in Submodule:** Once inside the submodule, check the status:

Code:

git status

It will show you the untracked files specific to that submodule.

1. **Add and Commit Changes in Submodule:** Add and commit changes within the submodule:

Code:

git add . git commit -m "Add untracked files and modifications in submodule"

1. **Go Back to the Main Repository:** After committing changes in the submodule, go back to the main repository:

Code:

cd ..

Repeat this process for each submodule showing untracked content.

1. **Check Status in Main Repository:** Once you have committed changes in all submodules, check the status in the main repository:

Code:

git status

Ensure that there are no remaining untracked or modified files.

1. **Commit Changes in Main Repository:** If everything looks good, commit the changes in the main repository:

Code:

git add . git commit -m "Add untracked files and modifications in submodules"

Now, your main repository should be clean, and all changes, including those in submodules, should be committed. If you have a remote repository, you can push the changes using **git push**.

THEN AFTER FOLLOW THE BELOW STEPS:-

If the submodules are showing "new commits" in the main repository, it means that the submodule reference in the main repository has been updated to point to a specific commit in each submodule. This reference is stored in the main repository, and when you make changes in the submodules, it's necessary to update this reference in the main repository.

Here are the steps to resolve this issue:

1. **Update Submodule References:** In the main repository, you need to update the submodule references to the latest commit in each submodule. Use the following command:

Code:

**git submodule update --remote**

This command fetches the latest changes from each submodule and updates the submodule reference in the main repository.

1. **Check the Status:** After updating the submodule references, check the status of the main repository:

Code:

git status

This should reflect that the submodule changes are no longer "new commits."

1. **Commit Changes in Main Repository:** If the status is clean, you can now commit the changes in the main repository:

Code:

git add . git commit -m "Update submodule references"

After completing these steps, your main repository should no longer show the "new commits" status for the submodules. The submodule references are now updated to point to the latest commits in each submodule. If you have a remote repository, you can push these changes using **git push**.

**How to add the submodules inside the main Repo?**

1. Inside the Main/Parent Repo, create .gitmodule file  
   touch .gitmodule
2. Manually update that file,  
   Ex:  
    [submodule "JavaScript/Projects in JS/OnePiano"]  
    path = JavaScript/Projects in JS/OnePiano

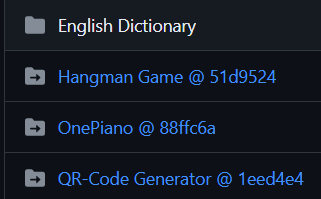
url = https://github.com/WebDevChandan/onePiano.git

1. Run the below command inside the main/parent repo to add the repo inside .gitmodule as submodule.  
   **git submodule update --init --recursive**
2. Run the following command  
   git submodule update --remote --recursive
3. Check the status of your main repo (It’ll be registered)  
   git status

**How to remove a particular submodule form the .gitmodule file?**

1. Open the **.gitmodules** file in the root of your main repository using a text editor.
2. Look for the required module that need not to be as submodule.  
   Delete it from .gitmodule file
3. Then you’ll need to staged the .gitmodule file, so run   
   git add . or git add .gitmodules
4. Run the below command:  
   git rm --cached pathToTheRequiredModule  
   Ex: git rm --cached 'NodeJs/ExpressJs Tutorial/Practical'
5. If you want it back, follow the steps for adding submodules

Note:

If the subfolder is showing in the parent repository like this:  


But the subfolders are not clickable or redirecting to any 404 page.

Then follow the below steps:  
1. Open git bash to each particular subfolder.  
  
2. Make sure the subfolder is updated by checking:  
 git status  
  
3. Check is there any remote available into that subfolder or not.  
 By executing the below command:  
 git remote -v  
  
4. If not, add remote git repo for that subfolder where you want it to redirect. By executing the below command:  
 git remote add origin “path/ofRepoForSubfolder.git”

6. Then open parent repository (/Learning) in VS Code and publish branch of particular subfolder