Short and Sweet Step-by-Step:
Git and GitHub Workflow

This guide details a step-by-step workflow for committing changes to software projects with Git and GitHub. Please feel free to download and share it.

Note: This document assumes you have already: 1.) set up a GitHub account; and 2.) installed Git on your computer. If you don't have Git installed yet, please visit the course *Short and Sweet: Get Started with Git and GitHub Right Now* and watch Lecture 2: "Setting up Your GitHub Account" and Lecture 3: "Installing Git."

Git and GitHub Workflow: New Repository and Initial Commit

Note: Use these steps if you **have not yet initialized** Git and GitHub repositories for your specific software project. Each software project needs its own local Git repository and matching remote (online) GitHub repository.

1. Log in to your GitHub account at https://github.com. In the upper right corner of the page, click the plus sign and choose 'Create New Repository.' Enter a new repository name and click Create Repository. This step creates your remote (online) GitHub repository for the project.
2. Using a command-line terminal on your computer, navigate to the software project directory (folder) where you want to use Git.
3. Initialize a new local Git repository by typing 'git init'.
4. Create a README file by typing 'echo "# repo-name" >> README.md' where repo-name is the name of your remote (online) GitHub repository. You can add more information to this file later using a text editor of your choice.
5. Create a .gitignore file by typing 'touch .gitignore'.
6. Open the .gitignore file with a text editor of your choice and edit its contents to make sure you list all files in the project directory that you do NOT want to upload to your remote (online) GitHub repository. List one file on each line, then save and close the .gitignore file.
   Tip: You can use wildcards such as asterisks to match multiple files, so if you want to ignore everything in a particular sub-directory, type 'FolderToIgnore/*' and Git will ignore all files in the folder called FolderToIgnore.
7. Add all project files to your local Git repository by typing 'git add --all'.
   Tip: This command will ignore all files listed in the .gitignore file.
8. Make your initial commit in the local Git repository by typing 'git commit -m "Initial commit"'.
9. Link your online GitHub repository to your local Git repository by typing 'git remote add origin https://github.com/username/repo-name' where username is your GitHub username and repo-name is your remote (online) GitHub repository name.

10. You may be prompted to enter your GitHub username and password. If so, enter this information.
11. Push files from your local Git repository to your remote GitHub repository by typing ‘git push -u origin master’. If you are prompted to enter your GitHub username and password, enter this information.

Git and GitHub Workflow: Commits to an Existing Repository

Note: These steps will work **only if you already have initialized** Git and GitHub repositories for your specific software project. Each software project needs its own local Git repository and matching remote (online) GitHub repository.

If you don’t have repositories initialized yet for your software project, follow the steps on Page 1 instead.

1. Using a command-line terminal, navigate to your software project directory.
2. Type ‘git status’. You will see any modified files listed in the terminal.
3. If only one file has been modified (changed), type ‘git add filename’ to add it to your local Git repository. OR:
   If multiple files have been changed, type ‘git add --all' to add them to your local Git repository.
4. Type ‘git commit -m “Commit message”’ to record the change in your project history. The commit message should be short and descriptive. Examples include “Fix bug in createList method” or “Add a deleteList method”.
5. Push your changes from your local Git repository to the corresponding remote (online) GitHub repository by typing ‘git push -u origin master’.
6. Check the status by typing ‘git status’. You should see a message that says, “On branch master. Your branch is up-to-date with ‘origin/master’. Nothing to commit, working directory clean”.

There’s much more to learn about Git and GitHub, but I hope this step-by-step guide helps make these powerful tools less intimidating. Future Short and Sweet courses will dive more deeply into Git and GitHub, among other programming and technical topics.

If you have any questions, please feel free to email me at stephanie@shortandsweetcourses.com, send me a message via Udemy, or post in the discussion forums, and I’ll do my best to help you out.

Many thanks, and happy programming!