**The ITC GIT Workflow**

Working properly in Git is an important part of being a programmer, especially when working with other programmers. Even though you might be the only contributor on a project now, things can start to get messy once others are involved. This workflow is intended to give you a real taste of how developers work when using Git.

**Steps**:

1. Clone the github classroom repo
2. **Create a branch for the feature** (aka Milestone you are working on: If you are working on Milestone 1, the branch name should be milestone 1.
3. When commiting changes to your feature branch, the message should be in the following format:

<stage> (status): <short summary (50 chars or less)>.

Ex: “**M2 (WIP): Change the header to red text**”.

\***Stage**: In our case whatever milestone you are working on, so M1, M2, M3, ,etc..

\***Status**:

* **WIP**(work in progress) -- Whatever it is that you are working on for that feature.
* **HF**(hotfix) -- What you fixed (bugs, problems). Patches to the code.
* **F**(finished) -- feature or milestone complete without problems.

1. Make sure you pull often from the default branch: other developers might have been working and pushing changes in that branch.
2. When you have completed your feature branch (when you’ve finished a milestone) switch over to the default branch (check how it is called: main, master, or anything else) and **merge in** the feature branch.
3. Now push those changes on the default branch to the remote’s default branch.

**Ninja Step**:

* When you are merging your feature branch to master use the **squash** option so all your commits are not included on master.

**Notes**:

* Your commits must be clean. Committing several changes on a single commit is not allowed. (If you find yourself using the word AND, you’re making too many changes on your commit)

**Terms:**

\***Default branch**: the branch that the remote repository considers as the ‘principal’ one. By default is called main. It used to be called ‘master’, so you can see it with different names across different projects.

**Resources:**

* <https://www.udacity.com/course/version-control-with-git--ud123>
* <https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>
* <https://git-scm.com/doc>
* [https://learngitbranching.js.org](https://learngitbranching.js.org/)
* https://www.conventionalcommits.org