Git is a version control system that is vastly used by programmers to help organize and develop a variety of projects, from video games to important back-end code for businesses. Without Git or other version control systems, a developer may find themselves in a position where they want to reverse any changes they had made to their code, but are unable to as they likely only have one file that they already overwritten. Git allows one to traverse between all the versions a project has gone through, and has many other utilities that enable many other developers to work together on the same code in an organized manner. A Git workflow is how one would go about this. A good workflow is the key to using Git most effectively, and more importantly, the key to developing your project most effectively. The most important thing to consider while coming up is a workflow is to consider first and foremost how it will benefit your team, and then to consider how stable it will be.

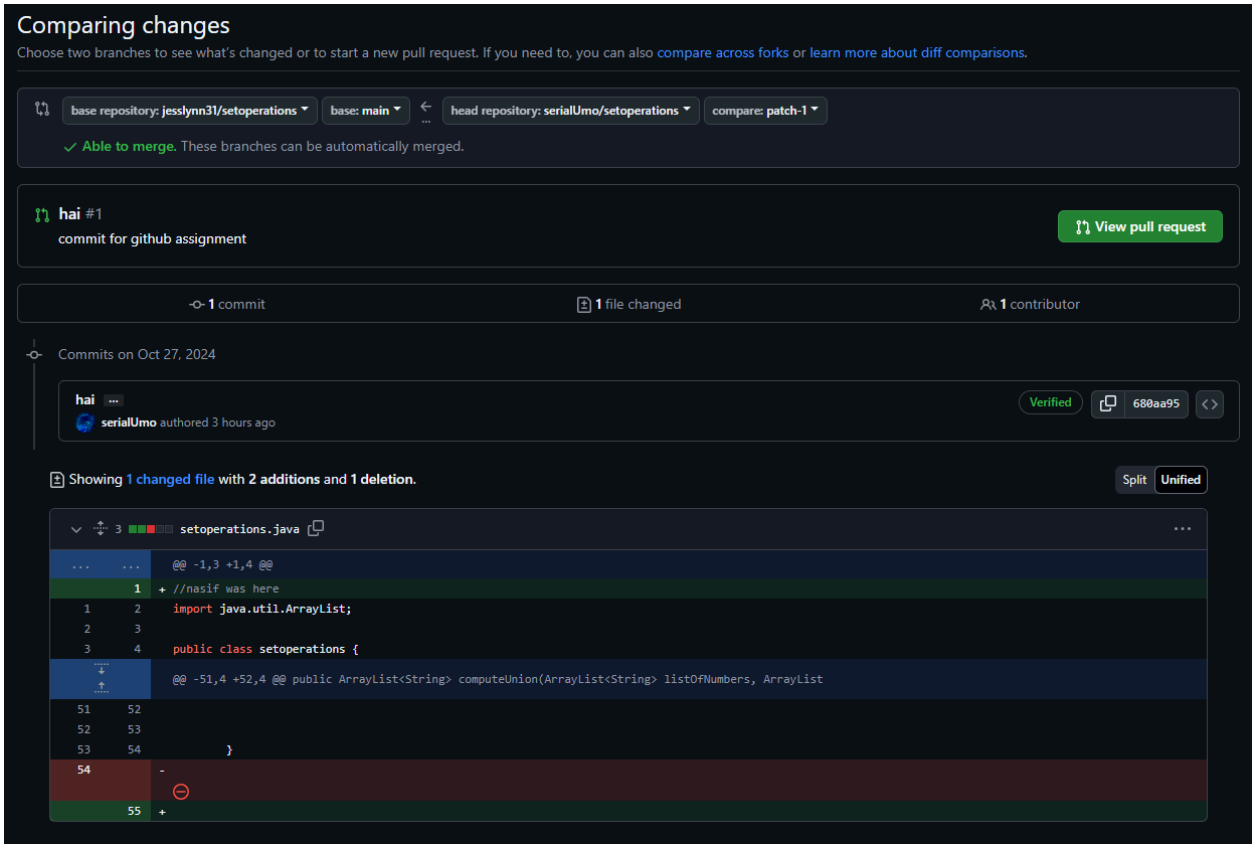
One of the types of workflows (and one of the most commonly applied ones) are the centralized workflow, where team members submit all their changes to a main, central repository. All of the different versions of the project are stored into the repository. Developers are then able to have their own local repositories of the code, and any changes that they make to it (or any "commits" they make to the local repository) are able to be "pushed" to the main branch, or published. The great thing about Git is that developers are able to navigate versions, so you are able to roll back to a previous version if something was pushed that was detrimental.

Sometimes outside developers may find interest in an open source project and want to change something. They are able to branch off of the main repository (or "fork" it) and commit any changes to that new forked repository. The forked repository could be its own project, but if the developers of the forked branch want to see their changes in the main branch, they can submit a "pull request". This asks the developers of the main branch to review the altered/added/removed code and see if they would like to incorporate the fork's changes into the main branch.

If any new changes are made to the main repository while a developer is working on their code, Git would not allow the developer to push their changes as it is based off of an older version. This is called a "merge conflict"--in simpler terms, different versions of the code are running into each other (ex: if there were two version 1.1s). In order to resolve this, the developer in question needs to update their code by "pulling" the new changes from the main branch, and to reincorporate their own changes on top of that (ex: rewriting your code to be based off the already made version 1.1, rather than version 1.0). Now, rather than having a branching tree where versions run into each other, there is one singular branch. After resolving the merge conflict, the developer can finally push their code (unless another developer pushed even more code before they could. At that point, consider working on your typing speed…)

Sources:

https://www.atlassian.com/git/tutorials/comparing-workflows



Commit to a fork of Jessica Smith's repo: