Jaiden Nunez

Git Workflow

Git is a very commonly used control system in modern software development. The reason for this is due to its incredible flexibility and ability to manage code on a much larger scale. At its core, A Git Workflow can be used as an outline for teams that use Git to complete their projects. For teams, when it comes to choosing the right workflow, the one they choose can drastically change both the success rate and productivity rate of their projects.

A Git repository is essentially the foundation of GIT. A Git repository is what stores a user's project files. Users can also edit their projects from this repository. There are two different types of repositories, there are local repositories and remote repositories. Local repositories are the repositories that are assigned to a developer's machine, and the remote repository is the repository stored on websites and applications like Github. What a developer would do is edit their local repository and upload it to the remote repository, where other developers can see and download their remote repository to their own local repository.

One of the keys to using Git effectively is by managing the *commits*, *pushes*, and *pulls*. A *commit* is a local snapshot of any changes that were made by a developer. *Commits* can be used to track a developer's progress incrementally, rather then all at once. Once a developer is finished tracking their progress using *commits*, what they would do is *push* those *commits* to their remote repository. This is a very useful tool for those working in larger teams, as it allows other developers to see the progress of individuals. If any developer wanted to have access to this new code that another developer *pushed*, they would have to *pull* that code from the repository. *Pulling* is very important because when working with many developers and many different pieces of code, if you are not up to date with the most recent code that was *pushed* by the other developers, then your code may cause conflicts due to you not being up to date with everybody else.

Merging is when two or more developers make updates or push multiple parts of code to a singular repository. However, when multiple developers make edits to the same portion of code, it will result in a Merge conflict. Usually, these conflicts need to be resolved manually by one of the users.