Alyssia Marshall

Git Workflow

Git workflow is a collaboration miracle. It allows software developers to utilize the open source platform for small to large projects alike using designated guidelines. This helps to prevent overlap and miscommunication which can easily occur in group projects, however, it is still important that everyone on the team is privy to what workflow recipe is being used. In choosing the workflow, it is recommended that the character of the team is taken into consideration, as different people have different strengths. To bring out those individual strengths and optimize the development process, choosing the right workflow is essential.

The most traditional workflow from which other workflows stem is referred to as the Centralized Workflow. It is the most straightforward approach that I’ve used myself (and currently use), as it allows the team to work from one central repository, usually referred to as "main", from which other team members clone the code and work on their own individual machines. It allows the use of only one branch to which team members upload their changes using push, which updates the project for everyone. One of the advantages of Git is that each team member can have a local copy of the project on their machine, giving them the ability to work independently and at their own convenience. Each team can determine at which breakpoint they should put their branch of code and this makes synchronizing the code easier. Any project is prone to incur conflict, but it’s easier to resolve by pulling the whole project, making the necessary changes, and then pushing it back again. Centralized workflow, in particular, minimizes branching and puts emphasis on a single server-side repository without specific patterns that the team members have to follow (besides pushing and pulling). For smaller groups, which is often what we have while in school, this minimizes confusion and makes the process easier for collaboration.

Though the Centralized Workflow is foundational, it becomes insufficient for conflict resolution the larger the team gets. Another specialized workflow is called Feature branching. Feature branching uses encapsulation in that features are to be broken up and developed separately from the main branch. This allows the main base of the code to remain unperturbed while isolating the feature code throughout the development process. This is better for larger projects. Another workflow is known as the Gitflow Workflow. The Gitflow workflow is very similar to Feature branching, except that each branch has a strict designation that defines its role in each project release. This is also good for larger projects, especially ones with multiple releases and different interacting branches. The Forking Workflow is a bit of a standout workflow in that each member of the team is operating from their own private server-side repository as well as the public one.

Any of the above workflows could technically be used to enhance any given project, however, which workflow to select depends heavily on the nature of the team. There are also more guidelines to consider. Branches should be “short-lived”, meaning the time spent separate from the main branch should be kept to a minimum in order to avoid potential conflict in merging and product deployment. Once a project has been merged, it is also important that the workflow you choose minimizes the need for reversion, but eases the process along if a change is necessary. Furthermore, product release timing is important for any business, meaning the main branch needs to be stable. Which workflow to use is an important preliminary decision because it directly influences the productivity of the team, which also affects the businesses profit; however, there is no right answer. The decision of Git workflow is completely unique to the team and the vision you all share for the production process.