Git is a very popular system that is used worldwide. It’s the most used and has public access for all people allowing for work to be shared and worked on through the cloud. Git workflow is what’s recommended on how to use GitHub to accomplish or complete tasks in the way that’s intended. It’s the standard way to go about using Git and has the best approach to how to most effectively use it. Using Git, teams big or small, can make changes and make branches through there code. This gives them a very high level of flexibility on being creative while also keeping their work. With how flexible Git is, there’s no one way to how to interact with it as people are left with their own creative self to use it best. But the same can’t be said for a team. A team should have one specific way to interact and use the Git cloud so that all members are on the same wavelength. This prevents any unnecessary conflicts both early and late into projects. Teams should have a standardized Git workflow to effectively communicate and interact with the GitHub cloud. One of the best Git workflows is to have a centralized workflow. This is where every team member creates their own clone of the project. All commits on this clone will not affect the main project. This lets a buffer be placed in between the main project and the changes made developers. This lets multiple people with different perspectives find any errors before pushing it to the main project. Conflicts occur when the commits made from a cloned repository to the main repository aren’t updated with the most up to date code made on the main repository. Git refuses to push the code and makes the cloned repository have the most up to date code from the main branch and edit it to match the same updated central commits. This lets projects move forward while always keeping the code up to date. Without letting people work on old code and making sure they are always up to date on the code, they are forced to always keep up with changes and make appropriate to their changes. Another popular Git workflow is called the centralized workflow. The idea behind this method is that all work to be done is made on branches. The main branch will get one big update later while developers are able to work on branch code more effectively. With less likely issues of merge conflicts, they can more effectively make quicker progress since they must only keep up with their branch rather with the main branch which is always updating itself. This method works better on teams where they need to work on multiple fronts at the same time. They can effectively work on their branch and make as many changes along with testing as they need. Once thoroughly tested and revised, they can push to the main more easily and continue working from another branch once again. Other popular Git workflows are featuring branching workflow, Gitlow workflow, forking workflow and may others. They all have their advantages and disadvantages. The main purpose of a team to effectively work and efficiently manage their work is by choosing the best model for them. The earlier they choose a model that both fits and works for them, the easier they will have later one. Project only gets bigger and more complex as more work and personnel are added to it. To always manage it effectively they need to find the best fit. A git workflow should manage a worker’s culture, schedule, workload, and management. Some other key things to look out for when working on a Git workflow is that branches become harder to merge the longer it is away from the main branch. Merge conflicts are bound to happen if a branch is kept away from the main for too long. That’s why it’s key to find the right balance and schedule to when someone should upload but also the necessary balance to let people have enough time to work on their branch to where they make effective progress. Balance is the most important thing for a git workflow.