Tyler Smith

Probability and Statistics Applied

GitHub is a very useful website and tool for creating a space where anyone can store their work and share it with others easily. It is especially useful for those in the computer science field, as one can push their code into a repository, and then their peer or boss is able to review, test, and modify the code as needed. All of this is done through what GitHub calls their "workflow". In this essay I will be going over the many different parts of GitHub workflow, what they do and how to set it up. In order to be able to implement the different parts of workflow, you will need to sign up and create a GitHub account and then create what is called a "repository", which contains all your project's files and their revision history.

Assuming that the prerequisites have been completed, one can begin to utilize the different features apart of GitHub workflow. In the repository created, the first step the user must complete is to create what is called a branch. It should be named something descriptive so that others know what the work is without having to open it. This allows the user to have a workspace that does not affect the default branch and gives others the chance to look at and modify the work in the branch. Two of the resources apart of GitHub that are very useful for branches are the commit and push options. A commit happens when the user has made an isolated and complete change, and GitHub applies this to the local repository. Making sure the change is complete allows reverting a commit simple. If the user decides later that they would like to change it again or try something different, they can revert the initial commit and complete the change in a different way. Commits allow the user to add a message along with, and that message should always be descriptive so that peers can fully understand what changes were made. A step further than the commit, is the GitHub push. While a commit adds changes to the user's local repository, the push transfers the commit to a remote server. Only when changes are pushed to the remote depository is when it will become accessible to view by the user's peers or anyone they are collaborating with.

Next, and one of the most valuable aspects of the GitHub workflow, is the pull request. A pull request is used to ask the user's peers or other collaborators for direct feedback on the changes made. To create a pull request, once in the branch that contain previously made commitments, there is a button that says, "pull request." The base branch drop down is the branch that the user wants the changes to merge into, and the compare branch drop down is the branch that the user made the initial changes in. Similarly to commits, GitHub requires the user to add a message along with the pull request, and within this the user should write a summary of the changes. If the user is using this as an opportunity for peers or collaborates to address and help with an issue, then that should be stated within this description. The user can also add comments on specific lines of the work if they would like to have a more direct request that is easier for others to identify. In response to feedback from others, the user can continue to make commits and pushes, and the pull request will accommodate them and update the pull request.

Finally, once the pull request is finished being reviewed and edited, the user should merge it. GitHub will then merge the user's branch to appear on the default branch. All of the comments and revisions made will be saved and can be viewed by future contributors. In some instances, a merge conflict can occur. This is when the user tries to merge two branches that have competing commits. This normally happens when multiple different collaborators make different changes on the same line of the same file. GitHub tells the user where the conflict occurs and then the user must manually edit the file and select which changes should stay during the merge.

Once all the above is completed, and any conflicts are taken care of, the pull request should merge. Finally, the user should delete the branch initially created. This means that the work on the branch is complete, no information will be lost, and the history of changes will be retained.

Sources:

https://docs.github.com/en/get-started/quickstart/github-flow

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/about-merge-conflicts

https://stackoverflow.com/questions/2745076/what-are-the-differences-between-git-commit-and-git-push

https://github.com/git-guides/git-push