Mia Watts March 4, 2024

**An Analysis on Git Workflow, Commits, Pushes, Pulls, and More**

Distributed Version Control systems are known for their flexibility, allowing developers to publish projects either for company or personal purposes. Git is an example of such a version control system, being one of the most widely used systems for program sharing and project storing. In using Git, it is important to decide on a workflow that can benefit both individuals and companies overall. Workflows guide employees on how they can best use Git, attempting to increase efficiency and decrease difficulty. When discussing workflows within a business, selecting the correct workflow for the team in question is crucial in deciding how to best process and store work produced. After selecting the correct workflow based on which would work best, a main repository can be created, which is a storage system that assists in deleting, adding, or modifying new or preexisting files.

When using certain Git services, users can perform functions that can allow them to add to their repositories. These functions include commits, pulls, and pushes, each serving a different purpose and giving the individual or company the ability to take code from their preexisting repositories or create new files. To place code or other relevant files into a repository, the commit command should be used first. This command stores changes locally rather than switching them into the main repository, building a backlog of information that can eventually be added to the repository using the push command. The push command moves file changes that were stored locally into the main repository, updating the repository to reflect the information added or removed. Once information has been pushed to the main repository, others can access it if the repository is public. Otherwise, only the author can access the content unless they provide others with access to their files and code. Along with commit and push commands comes the pull command, which takes code from a repository and places it into an IDE or other software depending on what the files in the repository are. Any code that is stored within the repository can be moved to the machine on which the pull request was generated, allowing users to make changes without affecting the main code.

It is important to note that Git services come with merges and merge conflicts, the latter of which arises when information is changed that does not align with the information in the main repository. If two individuals are working on code from the same repository, if someone else commits and pushes their code first, the other individual must pull the changes before committing and pushing theirs or there will be a merge conflict. Merging overall ensures that code or other information from two separate branches is assimilated into the main repository without issues, combining the code from both pushes to maintain a singular repository and branch. There be multiple branches in a repository, but regardless of how many branches there are, merges are still able to add code to specific branches depending on which are being changed. To solve merge conflicts that may arise, small commits can be made after discovering which files are at fault, thus preventing a large conflict that may occur from trying to resolve the entire problem at one time.