Kaden Fuller-Aujla

Git Work Flow

Git is a “version control” system in which each Developer on a team can access their copy of code. Since it is a version control system Developers also have the entire version history at their disposal. Git uses a variety of commands to make this system work. Gits’ foundation is built through “commits”, which are known as “local milestones” throughout the development of a project. A commit is essentially capturing what you have done over a certain period and logging it as a milestone for the project locally. For example, if you make changes to a method and perform a commit, it will add a place along your project timeline with those changes. There are two advantages to this, one of them being that you can easily see the version history of a project (from each commit). The other is that you do not have to make a commit to the remote repository, making it so you only must touch the remote repository when you choose.

After a Developer has their commits in place, they can now perform what is known as a “push”. A push is used when you want to upload the local repository commits into the remote repository. An example of this is if you clone a remote repository, decide to make a few commits along the way, then want to push the code back to the remote repository. Since your code is “ahead” of the remote repository, once you push it back to the repository it will do a merge as well. This will result in the locally pushed repository merging with the remote repository, making it so the remote repository is up to date with all the changes you made. On the other hand, there is something known as a “pull”, and this is the opposite of a “push”. A pull is a combination of two actions, a “fetch” and a “merge”. What a pull does is first fetch any changes made to the remote repository and then merge them into your local repository. This ensures that later down the road you will not encounter any merge conflicts because your code was “out of date” with the remote repository. If you make changes to the local repository before pulling in the remote repository, when you go to push your changes, they will result in a merge conflict (more on that later). It is also worth noting that instead of doing a pull, it is safer to do a fetch. A fetch will not change the local repository “forcefully” like a pull.

Branching is one of the most important features of Git, and it is known to make safe changes to code bases before doing merges. A good example of this system is if there is a ‘main’ branch and someone must add a feature to it, they can create a branch called ‘feature\_1’ and work on the feature inside of this branch. So, when they are ready, they can safely merge the changes to the ‘main’ branch. There is one thing to be aware of when performing a merge, and it is known as a “merge conflict”. A merge conflict can be caused by multiple developers making code changes in the same area, even when they are in different branches. For example, if Bob were to clone a remote repository, make some changes in a separate branch, and go to merge to the main branch. There could be a merge conflict because during that time someone else may have pushed changes to the main branch. So now when Bob tries to merge his changes into the main branch, his branch is not up to date with all the other changes someone else made.

Citation

“Learn Git with bitbucket Cloud”. *Atlassian Bitbucket,* <https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud>. 30 Jan 2023.