

Dan Mine

Git is an open-source project made by the same person who created the Linux operating system, Linus Torvalds. Git can be used for a variety of different purposes. Whether it is open-source code or commercial products, Git is used as a repository that, unlike its predecessors, allows people to see the full change history of any code repository. Git itself is very flexible as it supports branching and tagging that match protocol for both small and large projects. It is also a distributed version control system that allows each developer to have a full copy of the project history locally and allows for work to continue even if the central server is down. This makes Git the go-to standard because it is well-supported, but it does have its downsides. A large criticism made toward Git is its learning curve, but its performance, security, and flexibility make it the de facto standard. The complex yet flexible nature of Git allows there to be no standard procedure on how to use it. One way to approach it is through a centralized workflow. This allows a main repository to be copied to each member of a team and allows for independent work on their own projects without worrying about the developments in the main repository. This is usually the standard method when creating a project. Other popular types of workflows include feature branching, gitflow, and forking. Feature branching is the idea that all development takes place outside of the main branch and is an extension of centralized workflow. Gitflow workflow is a very strict workflow that gives roles to different branches and how and when they will interact with each other. Lastly, forking workflow gives everyone access to the main repository. This allows for each person to have a public server-side repository and a private local repository. Once a workflow is settled on depending on the project scope, knowing how to use commits, pushes, and pulls. Commits are similar to snapshots of the project. However, committing a snapshot does not automatically put it into the main repository. Instead, it is saved in the local repository where it can then be pushed. This allows the local repository to build up commits to split, sort, or clean up before being published to the main repository. Git also has a staging area, where changes can be reviewed before committing. This helps manage what gets included in a commit. A push is what allows you to upload your local repository to another repository and overwrite changes. A pull is the opposite of a push and is used to download content from a remote repository and transfer it to the local repository. A developer can also use a pull request. A pull request is a way to ask a developer to merge one of your branches into their repository. Merging is also a useful feature in Git. Merges are mostly used to put two series of commits together. It identifies a common base commit between the branches and combines them into one. A problem you may encounter with merges is a merge conflict. A merge conflict is when multiple developers are trying to edit the same line of code. If a conflict is found by Git, it will mark the file as conflicted and stop the merging process until the issue is fixed. There are two types of merge conflicts which are failing to start the merge and failing during the merge. Failing to start the merge means that there are changes in the working directory or staging area of the project. This is not a conflict due to other developers, but rather because there are local changes that are unresolved. However, if Git fails during a merge, that means there is a conflict with another developer's code. The files will be merged as best they can, with the conflicted files left to the developers. Additional features, like rebase, allow for cleaner project history while tagging code marks important points in the history.