Ian Valiante

10/28/2024

Git is a distributed version control system and is one of the most popular. It allows teams to track changes, collaborate, and manage project history. Being able to effectively use Git is essential as a developer. If you are unable to use Git, you will not last long at any job.

Git repositories, commonly referred to as “repos” are a data structure that stores files, and directories, typically the source code for a project. Repos are initialized by running Git in a project directory. This establishes a ‘.git’ folder which contains all the required files for tracking changes, branches, and commits. Each dev that has access to the repo can store their own local copy of the project, with its history and functionality. They can be stored locally, or on a remote platform like GitHub; note this is not the only platform that can be used for this purpose.

A Git workflow is the sequence a dev follows to contribute to a repo. The most common is known as the “Feature Branch Workflow”. In this workflow developers will create separate branches for each feature they work on. This workflow is the most common because it ensures a stable main branch and limits the scope of changes to feature branches. Git workflows typically include creating branches, making commits, pushing and pulling changes, merging branches, and handling merge conflicts. Example: Create a branch, make changes and commit, push changes, open pull request.

A Git commit is a snapshot of changes in the project, allowing developers to save their work. When making commits, we must include a brief descriptive message that anyone on the team would understand. Often this includes a “v X.Y.Z” to indicate the version number. Commits are essential to tracking the progress on a project. They should focus on a specific change to ensure clarity and assist in troubleshooting.

When you commit to a repo, you must then ‘push’ these commits to sync them with a remote repo. To do this the dev runs git push origin *branch\_name* where origin is the default name of the remote repo and branch\_name is the branch where the changes will occur.

When working with other developers we often must ‘pull’ from the remote repo. In essence we are taking the remote repository and merging it with our local repository. In practical applications we commonly use git pull origin branch\_name where branch\_name is the branch the dev wants to merge into their local branch.

Merging is when you integrate changes from one branch into another. This is primarily used when a dev has completed a feature in their branch and wants to bring it to the main. The command used to do so will be git merge branch\_name. Merging will be done automatically, but sometimes there are conflicts.

Merge conflicts happen when Git cannot automatically resolve differences between two branches. This typically occurs when two developers have edited the same lines of code differently or have made changes to things that are dependent upon another. If a merge conflict occurs, Git stops the merge process and allows the dev to review & resolve the conflicts. Some common commands used while working through merge conflicts are git status, git diff, git merge –abort, and git add filename.

Git and structure Git workflow have made the lives of developers much easier and gotten rid of many issues. Local network drives, email attachments and dedicated file servers are no longer the norm, and I am thankful I never experienced that. The version management, conflicts, and lost work that came from the methods of old must’ve been incredibly cumbersome and detrimental to efficiency. This is why it is essential for us to know how to use Git as a developer. Git is the backbone of version control and mastering the commands for commits, pushes, pulls, merges and resolving merge conflicts enables us to collaborate.