Enhance Project Workflow Using Git

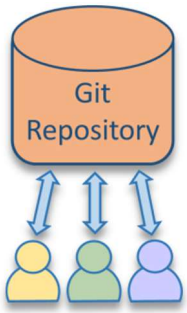
Project collaboration in a team environment can be difficult and chaotic. Typically, project work is divided among team members, and everyone is responsible for their own tasks. Eventually, all elements must be combined into one cohesive final product. But what happens when all components are combined, and the features appear incompatible or fragmentary? The key to successful team collaboration is finding a tool that makes it easy for team members to combine project elements and share information throughout the life cycle of a project in an effective way. A version control system (VCS), such as Git, is the ultimate collaboration tool, designed to transform project collaboration from a difficult and chaotic process to a simple and organized one.

What Is a VCS?

A VCS is a file repository that manages multiple files within a project, keeps a record of any file changes, and records which user made those changes. Users can access a file or project, contribute to it, and upload the changes so other users can view the latest revision.

What Is Git?

Git is a free open source and widely used VCS. It is a collaboration tool that enhances project workflow by making it easy for developers to access project history and contribute to shared projects of any size. Users can clone (i.e. download) repositories remotely to their computer to work on various project elements independently of their team, and easily combine their components with the main project at any stage of the project life cycle.



Team Collaboration Is Easy with Git

With Git, users can download the entire history of a project and access it offline, work at their own pace, and make progress on their local repository without fear of altering the final project. Users can view and make changes to the repository at any point in the history of the project. Git is efficient, reliable, and versatile. It is designed to provide developers with features that enhance project workflow and collaboration among team members.

A VCS can be centralized or distributed. The chart below shows why Git, which runs as a distributed version control system, is better suited for team collaboration than a centralized version control system such as Perforce or Subversion.

Attributes	Centralized Version Control System (Perforce, Subversion)		Distributed Version Control System (Git)	
Requires Central Server Communication	✓	All file and project information is stored on one central server and can be checked out by users.	✗	Once a repository is cloned, users do not need to constantly communicate with a central server.
Downloadable Local Repository	✗	Users must connect to the central server to access files.	✓	Repositories can be cloned to personal computers to easily access project history.
Project Protected from Data Loss	✗	Errors on the central server can lead to lost project files and history.	✓	Cloned repositories can reinstate project history if lost/corrupt due to server errors.
Accessible Offline	✗	A network connection is required to communicate with the server.	✓	Users can work on the local repository without a network connection.
Actions Performed Instantly	✗	Remote commits can be slow due to network conditions, and cannot be made at all if the server is down.	✓	Making or reverting changes is almost instant since there is no need to communicate with the server.
Protects Workflow from Bad Code	✗	Bad code could negatively impact the project for all users.	✓	Users can work on the same project remotely without risk of altering the project for everyone else, encouraging a non-linear workflow.
Control User Access	✓	Project leaders can easily control user access.	✓	Project leaders can easily control user access.

Collaborative Features

Explore how these Git features simplify and enhance project workflow.

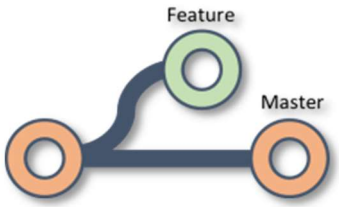
Commit

A change to the repository, known as a commit, is saved to the local repository before it can be pushed to the remote repository, where project collaboration occurs. Git uses snapshots to track changes between commits at one moment and creates a reference to the snapshot. If there are no changes to a file, Git links to an identical file rather than storing the file again. This is how Git commits can be pushed efficiently to the remote repository, where other users can access the latest project revision.

Git Commit

Branch

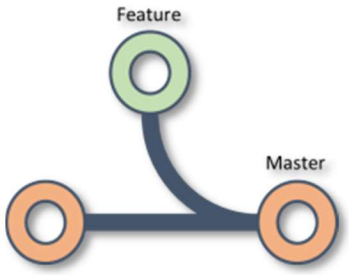
Users can diverge from the master branch (i.e. the root of the development tree) to work on a feature separately without impacting the final project. After downloading the project history to their computer, users can create branches and switch between them almost instantly, allowing them to work on multiple features at the same time.



Git Branch

Merge

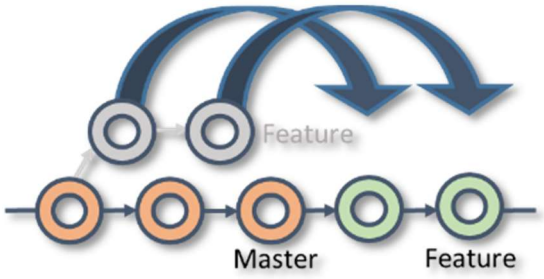
Users can merge feature branches back into the master branch quickly and efficiently as they are completed, rather than during the final stages of a project. This means any compatibility issues among features are discovered early in the project life cycle.



Git Merge

Rebase

A rebase pulls new information from the master branch before merging, reorganizing the history of the merge to create a more linear history. This makes it easier for users to read and follow the history of a project when collaborating with other users.



Git Rebase

Three Reasons to Use Git

Efficiency

Users can clone the entire history of a project directly to their personal computers, access the project without a network connection, and perform actions instantly. Users always have access to the project once the repository is cloned, so workflow is not impacted by network issues, distance between team members, or poor communication to a central server.

Reliability

Git keeps a record of all changes made, reducing the risk of users altering the project for everyone else. Commits are safely stored, and it is difficult for a user to delete a commit accidentally, making it a very reliable tool for team collaboration.

Versatility

Git is capable of supporting projects of any size, which makes it a desirable and popular collaboration tool. Since Git is an open source tool, there are many online resources and applications to help make Git more user-friendly.

Manage Git

Online tools like GitHub, GitLab, Bitbucket, and Gogs can be used to customize and enhance the usability of Git. These tools offer additional features that supplement Git's features and may further enhance your project workflow.

Summary

Project collaboration among team members doesn't need to be difficult and chaotic. With the proper tools, any team can complete a project successfully and with minimal conflicts. If you are looking for a collaboration tool that gives users the freedom to work on multiple tasks without a network connection, protects project history and data, and has many online tools available that offer customizability, then Git is the VCS for you. Consider making the switch to Git to enhance your project workflow.

Visit https://git-scm.com to learn more about Git.