AN INTRODUCTION TO GIT FOR TECHNICAL WRITERS

by Jordan Johnson

WHAT IS GIT?

Git is a free open source tool that allows you to track and manage changes to files and directories.

WHY TECHNICAL WRITERS SHOULD USE GIT

Even though Git is a tool traditionally used by software developers, technical writers gain many benefits from using a version control system such as Git. These benefits include:

- Having snapshots of your work all saved in one place.
- Being able to easily restore project backups and roll back unwanted changes.
- Being able to collaborate with other writers and review each other's changes.
- Being able to easily manage different versions of the same document.
- Having an automated and efficient documentation publishing process.

IMPORTANT GIT CONCEPTS

Below are key concepts new users must understand to start using Git:

Repositories

All the files for a project are stored in what's called a Git repository. A repository is a database of files that allows you to track the changes made to the files.

Repositories are hosted on a remote server. Multiple users can clone a remote repository and have a local version on their computer. Any project changes users make locally can then be pushed and merged into the remote version of the repository.

Storing project files in a repository allows users to review their project history and see:

- Who has made which changes.
- When changes were made.
- Why changes were made (via a Git commit message).

Snapshots

Each time you change a file Git takes a snapshot of your whole project and saves it. This allows you to easily backup files and ensures you never delete anything by accident.

If you make a mistake, you can easily revert your project back to an earlier state.

Pushing and Pulling Changes

Each team member will have a snapshot of the remote project repository on their machine. As users work, they will push their changes to the remote repository and pull changes other users have made down from the remote repository. The **git push** and **git pull** commands will allow you to push and pull your changes respectively.

Branching

A branch is a separate version of all the files from the main repository. Branching allows team members to work on different versions of a project without affecting the main branch. Software development teams often have separate branches to build various features for a product.

COMMON GIT COMMANDS

- **git add:** Stages the changes you made on your local repository to be included in the next snapshot that will be pushed to the remote repository.
- **git commit:** Saves the current snapshot of your repository to the project history.
- **git push:** Updates the remote repository with the commits made to your local repository.
- **git pull:** Adds changes from the remote repository into your local repository.
- **git status:** Displays a summary of the files that contain changes and are staged for the next commit.
- **git merge:** Merges together changes from separate branches.
- **git log:** Shows the commit history of the repository.
- **git stash:** Allows you to save the current state of your repository which contains your uncommitted changes and return to a clean version before you made those changes.
- **git revert:** Creates a new commit which reverses the changes you made in your previous commit. **Note:** Your current working environment must be clean with no uncommitted changes before running this command. You may have to run **git stash** before running **git revert**.
- **git checkout/git switch**: Switches your repository to a different branch and updates your local repository with the snapshot of files stored on that branch.
- **git branch:** Creates a separate version (branch) of a repository.
- **git tag:** Lets you tag specific points in a repository's history as being important.

RESOURCES

FOR MORE INFORMATION ON USING GIT SEE THE FOLLOWING RESOURCES:

Git Documentation
https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

Git Documentation on Git Branching
https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Interactive Tutorial on Git Branching
https://learngitbranching.js.org/

Interactive Git Tutorial
https://gitimmersion.com/

Git Gui Clients
https://git-scm.com/downloads/guis