**Git**

Git is a powerful version control system widely used for tracking changes in source code during software development. Created by Linus Torvalds in 2005, Git has become an essential tool for developers worldwide. Understanding Git can significantly enhance your coding efficiency and collaboration.

This Git Tutorial will walk you through the basics of Git, making it easy to learn and implement in your projects. You will learn GitHub fundamentals and advanced concepts like branches, pushing merge conflicts, and many useful Git Commands.

Git is a distributed version control system (DVCS) that allows developers to track changes in their codebase, collaborate with others, and manage different versions of their projects efficiently.

**What does Git do?**

* Manage projects with Repositories
* Clone a project to work on a local copy
* Control and track changes with Staging and Committing
* Branch and Merge to allow for work on different parts and versions of a project
* Pull the latest version of the project to a local copy
* Push local updates to the main project.

**Working with Git?**

* **Initializing a Repository:** When you initialize a folder with Git, it becomes a repository. Git logs all changes made to a hidden folder within that repository.
* **Staging Changes:** Git marks modified files as “staged.” Staging prepares changes for a snapshot you want to keep.
* **Committing Changes:** Once staged changes are satisfactory, commit them. Git maintains a complete record of each commit.

**What is GitHub?**

GitHub, a hosting service for Git repositories, allows you to access and download projects from any computer. Here’s what you can do with GitHub:

* **Store Repositories:** GitHub hosts your repositories.
* **Collaborate:** Work with other developers from any location.
* **Version Control:** Manage collaborative workflows using Git and GitHub.

**Git Commands**

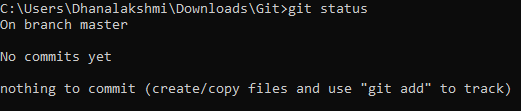
* **git init**

git init is a command used to initialize a new Git repository in a directory. It creates a .git folder, which tracks the version history and other Git-related data for the project. After running git init, the directory is ready to start tracking changes, adding files, and making commits.

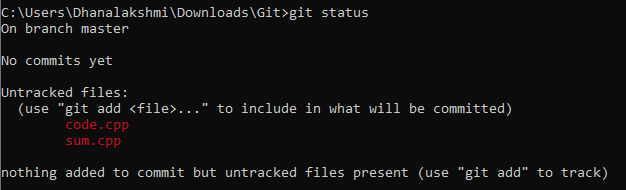


* **git status**

git status shows the current state of the working directory and staging area. It displays which files are modified, staged for commit, or untracked. It helps you see what changes are ready to be committed and what still needs to be added.

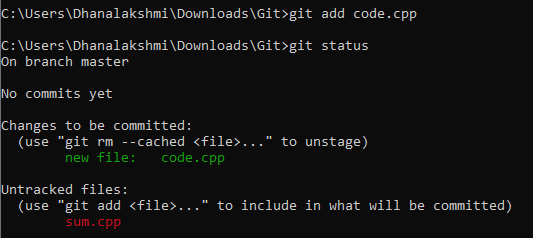


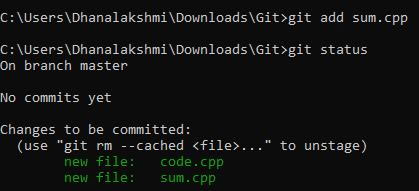
* Creating a files code.cpp and sum.cpp



* **git add**

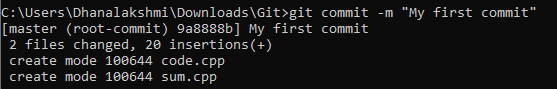
git add is a command used to stage changes in the working directory, preparing them to be included in the next commit. It adds modified or new files to the staging area, making them ready for commit.





* **git commit -m "My first commit"**

git commit is a command used to save staged changes to the local repository. It records the changes made to the files in the staging area with a message describing the update.

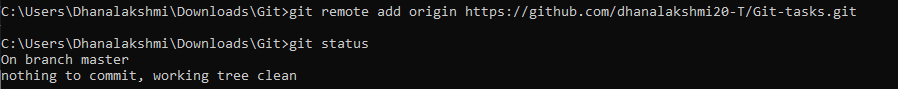


* **git status**



* **git remote**

git remote is a command used to manage remote repositories in Git. It allows you to view, add, and modify remote repositories that are connected to your local repository, enabling you to push and pull changes between them.



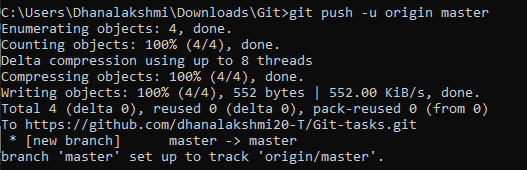
* **git branch**

git branch is a command used tolist, create, or delete branches in a Git repository. A branch allows you to work on separate features or tasks independently of the main codebase.



* **git push**

git push is a command used to upload local repository changes (commits) to a remote repository, such as GitHub. It updates the remote branch with your local commits.



* **git pull**

git pull is a command used to **fetch and merge changes** from a remote repository into your local branch. It updates your local repository with the latest changes from the remote.

**Some other commands in git**

* **git clone:** Creates a copy of a remote repository on your local machine.
* **git log:** Shows the commit history for the repository, including commit IDs, authors, dates, and messages.
* **git checkout:** Switches to a different branch or restores working directory files to a specific state.
* **git merge:** Combines changes from one branch into another.
* **git reset:** Undoes changes by moving the current branch to a previous commit, optionally altering the staging area or working directory.
* **git rm:** Removes files from the working directory and stages the deletion for commit.
* **git diff:** Shows the differences between changes in the working directory and the last commit or staged changes.
* **git fetch:** Retrieves changes from a remote repository without merging them into the local branch.
* **git stash:** Temporarily saves changes in the working directory and reverts to the last committed state.