**What is Git?**

Version Control System is a tool that helps to track changes in code. Git is a distributed version control system that tracks versions of files. It is often used to control source code by programmers collaboratively developing software.

* Popular
* Free & Open Source
* Fast & Scalable

Download: <https://git-scm.com/downloads>

**What is GitHub**

Website that allows developers to store and manage their code using Git. GitHub is a developer platform that allows developers to create, store, manage and share their code. It uses Git software, providing the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

**Git Branches**

Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug you spawn a new branch to encapsulate your changes.



**Git Commands**

1. **Basic Terminal Commands**
   1. **git --version:**
   2. **ls (List Files):**
   3. **ls -a:** Shows hidden files
   4. **ls -lart:** Shows hidden files
   5. **clear:** Clears the terminal panel.
   6. **cd:** Change in directory. cd “Location”
   7. **cd ..:** Change out of directory
   8. **git log:** Checks all commits
   9. **code .:**  Opens up Visual Studio Code
   10. **pwd:** Prints out the current directory. (Print Working Directory)
   11. **q:** Quits the panel.
   12. **^C:** It cancels a command.
2. **Configuration Commands**
   1. git config --global user.name “My Name” git config --global user.name
   2. git config --global user.email [abc@xyz.com](mailto:abc@xyz.com) git config --global user.email
   3. git config --local user.name “My Name”
   4. git config --local user.email [abc@xyz.com](mailto:abc@xyz.com)
   5. git config --global --edit
   6. git config --list
3. **Workflow**
   1. **clone:** Cloning a repository on our local machine. {git clone <-Link copied from GitHub->} Remote [GitHub] vs. Local [Laptop]
   2. **status:** Displays the state of the code. {git status} Tells whether a file is up to date or is it modified, or if there are some untracked files. You can see a summarized status of changes by using {git status -s}. M means File has been modified in the Staging Area. M means File has been modified in the Working Tree.
      1. **Untracked:** New file that Git does not track yet.
      2. **Modified:** Changed/updated file.
      3. **Staged:** File is ready to be committed.
      4. **Unmodified:** Unchanged file.
4. **Functional Commands**
   1. **add:** Adds new or changed files in the working directory to the Git staging area. {git add <- file name -> To add all the changes, use git add . command or git add -A.
   2. **commit:** It records the change. {git commit -m “some message”} You can commit all the added file by {git commit > Press “i” > Type “some message” > Press “Escape” > Type “wq”} You can skip the staging area and commit changes directly by using {git commit -a -m “some message”}.
   3. **touch:** It creates blank files. {touch “File Name.Extension”}
   4. **checkout:** It matches the file with the last commit. {git checkout <- File Name ->} You can match all the files (current working directory) with the last commit by using {git checkout -f}.
   5. **log:** It shows all the previous commits. {git log} You can check last “n” number of commits by using {git log -p –“n”}.
   6. **Push:** Uploads local repository content to remote repositories. {git push origin main} origin is the copy of root repository on GitHub while main is its branch.
   7. **Pull:** It is used to fetch and download contents from a remote repo and immediately update the local repo to match that content.
   8. **Init:** It is used to create a new Git repository.
      1. **git init:** It creates a new Git repository.
      2. **rm -rf .git:** It undoes git init.
      3. **git remote add origin <- link ->**
      4. **git remote -v:** It verifies the remote.
      5. **git branch:** It checks branch.
      6. **git branch -M main:** It renames branch.
      7. **git push origin main:**
      8. **git push -u origin main:** To set up stream.
   9. **mkdir:** It creates new directory or sub-directory.{mkdir newDirectory}
5. **Removing Files**
   1. **rm:** It removes files from the git repository altogether. {git rm <- File Name ->}
   2. **rm --cached:** It removes files from the staging area only. {git rm --cached <- File Name ->}
   3. **rm -**f: It force removes the files from the git repository. {git rm -f <- File Name ->}
   4. **touch .gitignore:** You can ignore any file in the Git repository by typing in it its name with extension {/mylogs.log}, only its extension {\*cpp} and the name of the sub directory (folder) {/<- Folder Name ->}.
6. **Branches**
   1. **git branch:** It checks branch.
   2. **git branch -M main:** It renames a branch.
   3. **git checkout <- Branch Name ->:** It navigates between branches.
   4. **git checkout -b <-New Branch Name ->:** It creates new branch.
   5. **git branch -d <- Branch Name->:** It deletes a branch.
7. **Merging Code**
   1. **git diff <- Branch Name ->:** It compares commits, branches, files and more. It compares the working tree with the staging area. If you add changes and then execute git diff command, Git Bash will show nothing because the working directory and the staging area are now same. But you can compare the staging area to the last commit by using {git diff --staged}.
   2. **git merge <- Branch Name ->:** It merges two branches.
   3. **Create a PR ()Pull Request:** It lets you tell others about changes you have pushed to a branch in a repository on GitHub.
8. **Merge Conflict:** An event that takes place when Git is unable to automatically resolve differences in codes between two commits.
9. **Undoing Changes**
   1. **Case 01:** Staged Changes
      1. git reset <- File Name ->
      2. git reset:
   2. **Case 02:** Committed Changes
      1. git reset HEAD~1
   3. **Case 03:** Committed Changes (for many commits)
      1. git reset <- commit hash ->
      2. git reset –hard <- commit hash ->: Undoes changes from the VS Code as well
10. **Fork**
    1. A fork is a new repository that shares code and visibility settings with the original “upstream” repository. Fork is a rough copy.

**WorkFlow:** Local Git > GitHub Repository > Clone to Local > Changes in Files > Add Changes > Commit Changes > Push Local Repository to Main Repository. git pull main origin.