**Git**

**To add a folder from your computer to GitHub, follow these steps:**

**1. Navigate to Your Project Folder**

Open your terminal or Git Bash and navigate to the folder that you want to add to GitHub. You can use the cd command to change directories:

cd path/to/your/folder

For example, if your folder is called LearningProjects, you would run:

cd LearningProjects

**2. Initialize Git in Your Folder**

Once you're inside the folder, initialize a Git repository in that folder:

git init

This will create a .git folder inside your project directory, which means Git will now track changes in this folder.

**3. Add Your Files to Git**

Now, you need to add the files to Git's staging area. This tells Git to start tracking the files.

To add all the files in the folder:

git add .

The . tells Git to add all the files in the directory. You can also specify specific files by replacing . with the file names if needed.

**4. Commit Your Files**

Once your files are added to the staging area, you need to commit them with a message:

git commit -m "Initial commit"

This will save your changes locally with the message "Initial commit."

**5. Create a New Repository on GitHub**

Go to [GitHub](https://github.com/), log in, and create a new repository.

* Do **not** initialize the repository with a README or .gitignore file, since you're pushing an existing project.
* After the repository is created, GitHub will show you the repository URL, which will look like this:

https://github.com/your-username/repository-name.git

**6. Add the Remote Repository to Git**

Now, link your local Git repository to the GitHub repository by running this command:

git remote add origin https://github.com/your-username/repository-name.git

Make sure to replace https://github.com/your-username/repository-name.git with your actual repository URL.

**7. Push Your Code to GitHub**

Finally, push your local code to GitHub:

git push -u origin master

* -u sets the default upstream branch for future pushes.
* master is the name of the main branch. If your GitHub repository is set to use main, use git push -u origin main instead.

**8. Verify Your Project on GitHub**

After pushing, go to your GitHub repository URL. You should see your project files listed there.

**To access and review a file using Git from the command line, follow these steps:**

**1. Clone the Repository**

If you haven’t already cloned the repository:

1. Open your terminal or command prompt.
2. Use the git clone command to download the repository:
3. git clone <repository\_url>

Replace <repository\_url> with the URL of the repository (you can find it by clicking the green **Code** button on the repository page in GitHub).

**2. Navigate to the Repository**

Change into the repository’s directory:

cd <repository\_name>

Replace <repository\_name> with the name of the cloned repository.

**3. Check Out the Appropriate Branch**

If you want to review a specific branch:

git checkout <branch\_name>

Replace <branch\_name> with the name of the branch you want to review (e.g., main, develop, or feature-xyz).

**4. Find and Open the File**

1. Use ls (Linux/Mac) or dir (Windows) to list files and directories.
2. Navigate to the file location using cd.
3. Open the file in your preferred editor (e.g., vim, nano, or a GUI editor like VSCode):
4. vim <filename>

Or, if you’re using VSCode:

code <filename>

**5. Review the File**

* Read through the file to understand its content.
* If you find something that needs improvement or fixing, you can edit it in the same editor.

**6. Stage Changes (If Editing the File)**

If you made edits and want to save them:

1. Stage the changes:
2. git add <filename>
3. Commit the changes:
4. git commit -m "Your commit message"
5. Push the changes (if you have permission):
6. git push origin <branch\_name>

**7. View the Git History (Optional)**

To see previous changes made to the file:

git log -- <filename>

This will display the commit history for the specific file.

**8. Create a Pull Request (Optional)**

If your changes are ready for review:

1. Push them to a remote branch.
2. git push origin <new\_branch\_name>
3. Go to the GitHub repository and create a **pull request**.

**🚀 Complete Git Commands Cheat Sheet with Descriptions**

This list covers **all essential Git commands** along with their descriptions to help you work efficiently with Git & GitHub.

**🔹 1️Git Configuration**

| **Command** | **Description** |
| --- | --- |
| git config --global user.name "Your Name" | Sets your Git username globally. |
| git config --global user.email "your.email@example.com" | Sets your Git email globally. |
| git config --global --list | Shows all configured Git settings. |

**🔹 2️ Start a New Repository or Clone**

| **Command** | **Description** |
| --- | --- |
| git init | Initializes a new Git repository in the current folder. |
| git clone <repo\_url> | Clones an existing GitHub repository. |
| git clone <repo\_url> . | Clones a repository directly into the current folder. |

**🔹 3️ Working with Files**

| **Command** | **Description** |
| --- | --- |
| git status | Shows the status of modified, staged, or untracked files. |
| git add <file> | Stages a specific file for commit. |
| git add . | Stages all modified and new files. |
| git reset <file> | Unstages a file but keeps changes. |
| git reset --hard | Unstages all changes and resets to the last commit. |

**🔹 4️ Committing Changes**

| **Command** | **Description** |
| --- | --- |
| git commit -m "Commit message" | Commits staged changes with a message. |
| git commit --amend -m "New message" | Modifies the last commit message. |

**🔹 5️ Pushing to GitHub**

| **Command** | **Description** |
| --- | --- |
| git push origin master | Pushes changes to GitHub on the master branch. |
| git push -u origin master | Pushes and sets master as the upstream branch. |

**🔹 6️ Pulling and Fetching Changes**

| **Command** | **Description** |
| --- | --- |
| git pull origin master | Fetches and merges the latest changes from GitHub. |
| git fetch origin | Fetches changes but doesn’t merge them. |

**🔹 7️ Branching**

| **Command** | **Description** |
| --- | --- |
| git branch | Lists all branches. |
| git branch <new-branch> | Creates a new branch. |
| git checkout <branch> | Switches to a different branch. |
| git checkout -b <new-branch> | Creates and switches to a new branch. |
| git merge <branch> | Merges another branch into the current branch. |
| git branch -d <branch> | Deletes a local branch. |

**🔹 8️ Undoing Changes**

| **Command** | **Description** |
| --- | --- |
| git restore <file> | Discards local changes in a file. |
| git reset HEAD~1 | Removes the last commit but keeps the changes. |
| git revert <commit-hash> | Creates a new commit that undoes a previous commit. |

**🔹 9️ Working with Remote Repositories**

| **Command** | **Description** |
| --- | --- |
| git remote -v | Shows the remote repositories linked to the project. |
| git remote add origin <repo\_url> | Links a remote repository. |
| git remote remove origin | Removes the linked remote repository. |

**🔹 🔟 Handling Merge Conflicts**

| **Command** | **Description** |
| --- | --- |
| git merge <branch> | Merges another branch into the current branch. |
| git mergetool | Opens a merge tool to resolve conflicts. |
| git commit -m "Resolved merge conflicts" | Commits after resolving conflicts. |

**🔹 1️1️ Stashing Changes**

| **Command** | **Description** |
| --- | --- |
| git stash | Temporarily saves changes without committing. |
| git stash pop | Restores the last stashed changes. |
| git stash list | Lists all stashed changes. |
| git stash drop | Deletes the last stashed change. |

**🔹 1️2️ Viewing History**

| **Command** | **Description** |
| --- | --- |
| git log | Shows commit history. |
| git log --oneline | Shows commit history in one line. |
| git diff | Shows changes between commits or working directories. |

**🔹 1️3️ Deleting Files & Commits**

| **Command** | **Description** |
| --- | --- |
| git rm <file> | Removes a file and stages the deletion. |
| git reset --hard HEAD~1 | Deletes the last commit permanently. |

**📌 Bonus: Git Shortcuts**

| **Command** | **Description** |
| --- | --- |
| git checkout -- . | Discards all local changes. |
| git clean -f | Deletes untracked files. |
| git reflog | Shows a history of all Git actions. |

**✅ Summary**

* **Clone a repository** → git clone <repo\_url>
* **Track & commit changes** → git add . && git commit -m "message"
* **Push to GitHub** → git push origin master
* **Pull latest updates** → git pull origin master
* **Manage branches** → git checkout -b new-branch
* **Fix issues** → git stash, git reset, git revert

🚀 **Now you’re ready to master Git!** Let me know if you need more help. 😊