📌 Git Commands with Purpose

Here's a list of common Git commands along with their purposes:

Command	Purpose
`git init`	Initializes a new Git repository in a folder.
`git clone <repo_URL>`	Copies an existing GitHub repository to your local machine.
`git status`	Shows the current state of your repository (modified, staged, or untracked files).
`git add <file>`	Adds a specific file to the staging area.
`git add .`	Adds all modified and new files to the staging area.
`git commit -m "message"`	Saves the changes to the repository with a descriptive message.
`git push origin <branch>`	Uploads committed changes to a specific branch in GitHub.
`git pull origin <branch>`	Fetches and merges the latest changes from GitHub into your local repository.
`git fetch`	Downloads changes from the remote repository but doesn't merge them.
`git merge <branch>`	Merges a branch into the current branch.
`git branch`	Lists all branches in the repository.
`git branch <branch-name>`	Creates a new branch.
`git checkout <branch>`	Switches to a different branch.
`git checkout -b <branch>`	Creates and switches to a new branch.
`git reset <file>`	Removes a file from the staging area (undo `git add`).
`git revert <commit>`	Undoes a commit by creating a new commit.
`git log`	Displays a log of all commits.
`git diff`	Shows differences between commits, branches, or working directory changes.
`git rm <file>`	Removes a file from the repository and staging area.
`git stash`	Temporarily saves changes that aren't committed.
`git stash pop`	Restores the most recent stashed changes.
`git remote -v`	Lists all remote repositories connected to the local repository.
`git remote add origin <repo_URL>`	Links your local repository to a GitHub repository.
`git rebase <branch>`	Reapplies commits on top of another branch.

🛠️ Step-by-Step Guide to Upload Your Files (System) to GitHub

1️⃣ Create a GitHub Repository

1. Go to GitHub and log in.
2. Click on **"New Repository"**.
3. Enter a **repository name** and set it to **public or private**.
4. Click **"Create Repository"**.

2️⃣ Upload All Files from Your Local System to GitHub

(If Your Project Folder Doesn't Have Git Yet)

1. **Open VS Code** (or any terminal).
2. Navigate to your project folder:
   ```
   cd path/to/your-project-folder
   ```
3. Initialize Git in your project:
   ```
   git init
   ```
4. Add all files to the staging area:
   ```
    git add .
   ```
5. Commit the changes:
   ```
    git commit -m "Initial commit"
   ```
6. Connect the local repository to GitHub:
   ```
    git remote add origin <your_repo_URL>
   ```
7. Push your files to GitHub:
   ```
   git push -u origin main
   ```

🛠️ Uploading a Specific Edited File to GitHub

(When You Have Modified a Specific File in the Repository)

1. **Check the modified files**:
   `git status`

2. **Add the specific modified file**:
   `git add <filename>`

   (Example: `git add index.html` to add only `index.html`.)

3. **Commit the change with a message**:
   `git commit -m "Updated index.html with new features"`

4. **Push the changes to GitHub**:
   `git push origin main`

🛠️ How to Pull Files from GitHub, Edit in VS Code, and Push Back

1️⃣ Pull Files from GitHub to Your Local Machine

1. Open **VS Code** or **Git Bash**.

2. Navigate to your project folder:
```
cd path/to/your-project-folder
```

3. Pull the latest changes from GitHub:
```
git pull origin main
```

2️⃣ Edit Files in VS Code

- Open the project in VS Code.
- Make the necessary changes.

3️⃣ Stage and Commit Changes

1. Check what files were modified:
```
git status
```

2. Add the modified files:
```
git add .
```

3. Commit with a meaningful message:
```
git commit -m "Fixed bug in login system"
```

4️⃣ Push the Changes Back to GitHub

```
git push origin main
```

🚀 You're Now Ready to Work with Git and GitHub Efficiently!

Let me know if you need further clarification! 😊