# Complete Git Command Guide (With Explanations)

## 1️⃣ Setup and Configuration

\*\*Purpose:\*\* Configure Git for the first time so it recognizes who you are when committing code.  
- `git --version` → Check if Git is installed.  
- `git config --global user.name "Your Name"` → Sets your Git username globally.  
- `git config --global user.email "youremail@example.com"` → Sets your Git email globally.  
- `git config --list` → Shows all your Git configurations.

## 2️⃣ Create or Initialize a Repository

\*\*Purpose:\*\* Start tracking your project with Git or clone an existing one.  
- `git init` → Initializes a new local Git repository in your folder.  
- `git clone <repo\_url>` → Copies (clones) a remote repository to your local system.

## 3️⃣ Connect Local Repo to GitHub

\*\*Purpose:\*\* Link your local project with a GitHub repository.  
- `git remote add origin <repo\_url>` → Connects your local repo to GitHub.  
- `git remote -v` → Shows which remote repositories are linked.  
- `git remote set-url origin <new\_url>` → Updates the URL of your remote repository.

## 4️⃣ Basic Workflow Commands

\*\*Purpose:\*\* Day-to-day Git operations for tracking and committing your work.  
- `git status` → Shows changes between commits and working directory.  
- `git add .` → Stages all files for the next commit.  
- `git add filename` → Adds only a specific file.  
- `git commit -m "message"` → Saves your changes locally with a message.

## 5️⃣ Push and Pull Changes

\*\*Purpose:\*\* Synchronize your local code with GitHub.  
- `git push -u origin main` → Uploads your commits to GitHub.  
- `git pull origin main` → Downloads and merges the latest changes from GitHub.  
- `git fetch` → Downloads changes but does not merge them automatically.

## 6️⃣ Branch Management

\*\*Purpose:\*\* Work on multiple features independently without affecting main code.  
- `git branch` → Lists all local branches.  
- `git checkout -b feature-name` → Creates and switches to a new branch.  
- `git merge branch-name` → Merges a branch into the current branch.  
- `git branch -d branch-name` → Deletes a branch locally.

## 7️⃣ View and Compare Changes

\*\*Purpose:\*\* See what changes were made in your code.  
- `git log` → Shows commit history.  
- `git log --oneline` → Compact view of commit history.  
- `git diff` → Displays line-by-line changes in your working directory.

## 8️⃣ Undo and Reset Commands

\*\*Purpose:\*\* Revert or fix mistakes safely.  
- `git reset filename` → Unstages a file (keeps changes).  
- `git reset --soft HEAD~1` → Undo last commit but keep files staged.  
- `git reset --hard HEAD~1` → Completely remove last commit and changes.  
- `git revert <commit-id>` → Safely undo a specific commit by creating a new one.

## 9️⃣ Delete Files or Repository

\*\*Purpose:\*\* Remove unwanted files or stop Git tracking completely.  
- `git rm filename` → Deletes file and stages the deletion.  
- `rm -rf .git` → Deletes the Git repository from your folder (dangerous!).

## 🔟 Stash (Temporary Save Work)

\*\*Purpose:\*\* Save uncommitted work temporarily without committing.  
- `git stash` → Stores your current work safely.  
- `git stash list` → Lists all stashed changes.  
- `git stash apply` → Applies the last stash.  
- `git stash drop` → Removes a stash from the list.

## 1️⃣1️⃣ Remote Repository Commands

\*\*Purpose:\*\* Manage your GitHub connections.  
- `git remote show origin` → Shows information about your GitHub link.  
- `git remote remove origin` → Removes the connection.  
- `git remote rename origin upstream` → Renames your remote name.

## 1️⃣2️⃣ Tagging Versions

\*\*Purpose:\*\* Mark specific commits as release points (e.g., v1.0).  
- `git tag v1.0` → Creates a tag.  
- `git push origin v1.0` → Pushes tag to GitHub.  
- `git tag` → Lists all tags.

## 1️⃣3️⃣ .gitignore File

\*\*Purpose:\*\* Exclude unnecessary files from Git (e.g., node\_modules, .env).  
Example:  
```  
node\_modules/  
.env  
.DS\_Store  
```

## 1️⃣4️⃣ Troubleshooting Common Errors

\*\*Purpose:\*\* Fix common Git issues quickly.  
- “fatal: remote origin already exists” → Remove and re-add origin.  
 ```bash  
 git remote remove origin  
 git remote add origin <repo\_url>  
 ```  
- “rejected because remote contains work” → Pull and rebase before pushing.  
 ```bash  
 git pull origin main --rebase  
 git push origin main  
 ```

## ✅ Example: Full Workflow (Existing Local Project)

```bash  
git init  
git add .  
git commit -m "Initial commit"  
git branch -M main  
git remote add origin https://github.com/username/repo.git  
git push -u origin main  
```  
This workflow initializes, commits, and uploads your local project to GitHub.