**GIT COMMANDS**

**1. Configuration & Setup**

| **Command** | **Description** |
| --- | --- |
| git config --global user.name "Your Name" | Set your name for commits |
| git config --global user.email "you@example.com" | Set your email for commits |
| git config --list | View current configuration settings |
| git config --global alias.co checkout | Create a shortcut (alias) for commands |
| git init | Initialize a new Git repository |
| git clone <repo-url> | Clone an existing repository |

**2. Working with Files**

| **Command** | **Description** |
| --- | --- |
| git status | Show the current status of the working directory |
| git add <file> | Stage changes for commit |
| git add . | Stage all changes in the working directory |
| git reset <file> | Unstage a file before commit |
| git reset --hard | Reset everything to last committed state |
| git commit -m "message" | Commit staged changes with a message |
| git commit --amend -m "new message" | Modify the last commit message |

**3. Branching & Merging**

| **Command** | **Description** |
| --- | --- |
| git branch | List all branches |
| git branch <branch-name> | Create a new branch |
| git checkout <branch-name> | Switch to a different branch |
| git checkout -b <branch-name> | Create and switch to a new branch |
| git merge <branch-name> | Merge another branch into the current branch |
| git branch -d <branch-name> | Delete a local branch |
| git push origin --delete <branch-name> | Delete a remote branch |
|  |  |

**4. Remote Repositories**

| **Command** | **Description** |
| --- | --- |
| git remote -v | List remote repositories |
| git remote add origin <repo-url> | Link local repo to a remote repo |
| git push origin <branch-name> | Push changes to the remote repository |
| git push --force | Force push changes (overwrites remote history) |
| git pull origin <branch-name> | Fetch and merge changes from remote repo |
| git fetch origin | Fetch remote changes without merging |

**5. Viewing History**

| **Command** | **Description** |
| --- | --- |
| git log | View commit history |
| git log --oneline --graph --all | View commit history as a graph |
| git log --author="name" | View commits by a specific author |
| git log --grep="keyword" | Search commit messages |
| git diff | View unstaged changes |
| git diff --staged | View staged changes |
| git show <commit-hash> | View details of a specific commit |

**6. Undoing Changes**

| **Command** | **Description** |
| --- | --- |
| git checkout -- <file> | Discard changes in a file |
| git reset HEAD <file> | Unstage a file |
| git revert <commit-hash> | Undo a commit by creating a new commit |
| git reset --hard HEAD~1 | Delete the last commit |
| git reflog | View recent changes to HEAD and recover lost commits |

**7. Stashing Changes**

| **Command** | **Description** |
| --- | --- |
| git stash | Save uncommitted changes temporarily |
| git stash list | View saved stashes |
| git stash apply | Apply the latest stash |
| git stash pop | Apply and delete the latest stash |
| git stash drop | Delete a specific stash |

**Bonus: Useful One-Liners**

| **Command** | **Description** |
| --- | --- |
| git shortlog -s -n | Show commit count by author |
| git grep "function\_name" | Search for text in files |
| git blame <file> | Show who modified each line in a file |
| git diff --name-only HEAD~1 | Show files changed in the last commit |

* If you **want flexibility**, use **MIT**.
* If you **want patent protection**, use **Apache 2.0**.
* If you **want all future versions to remain open-source**, use **GPL**.
* If you **want to restrict branding usage**, use **BSD 3-Clause**.
* If your repo is **not code-related**, use **Creative Commons**.