**🧑‍💻 Git Commands Cheat Sheet (Good + Advanced Version)**

**1. Configuration**

* git config --global user.name "Your Name" → Set your global Git username.
* git config --global user.email "you@example.com" → Set your global Git email.
* git config --list → List all Git configurations.
* git config --global core.editor nano → Set default editor (replace nano).

**2. Repository Operations**

* git init → Initialize a new Git repository.
* git clone <url> → Clone a remote repository.
* git clone --depth 1 <url> → Shallow clone (faster, only recent history).

**3. Staging and Committing**

* git status → Show the status of working directory and staging area.
* git add <file> → Stage a file for commit.
* git add . → Stage all changes.
* git commit -m "msg" → Commit staged changes.
* git commit -am "msg" → Stage + commit tracked files.
* git commit --amend → Modify last commit (message or staged changes).

**4. Viewing Differences**

* git diff → Show unstaged changes.
* git diff --staged → Show staged vs last commit.
* git diff <commit1> <commit2> → Compare commits.
* git diff <branch1> <branch2> → Compare branches.
* git difftool → Use external diff tool.

**5. Branching and Merging**

* git branch → List all branches.
* git branch <branch> → Create a new branch.
* git checkout <branch> → Switch branch.
* git checkout -b <branch> → Create and switch.
* git merge <branch> → Merge branch into current.
* git branch -d <branch> → Delete branch.
* git branch -D <branch> → Force delete branch.
* git merge --abort → Abort a merge conflict.

**6. Remote Repositories**

* git remote -v → List remotes.
* git remote add <name> <url> → Add remote.
* git fetch → Download changes (don’t merge).
* git pull → Fetch + merge changes.
* git pull --rebase → Rebase instead of merge (cleaner history).
* git push → Push changes.
* git push origin --delete <branch> → Delete remote branch.

**7. Viewing History**

* git log → Show commit history.
* git log --oneline → Condensed history.
* git log --graph --oneline --all → Visualize branches.
* git log -p <file> → Show history of a file.
* git show <commit> → Show details of a commit.
* git blame <file> → Show who changed each line.

**8. Reset Commands**

* git reset HEAD <file> → Unstage a file.

**Reset Modes:**

* git reset --soft <commit> → Move HEAD, keep staged.
* git reset --mixed <commit> *(default)* → Move HEAD, keep unstaged.
* git reset --hard <commit> → Move HEAD, discard changes ⚠️ Dangerous.
* git reset --merge <commit> → Reset but keep unmerged changes.
* git reset --keep <commit> → Reset but keep non-conflicting changes.

**Common usage:**

* git reset HEAD~1 → Undo last commit, keep unstaged.
* git reset --soft HEAD~1 → Undo last commit, keep staged.
* git reset --hard HEAD~1 → Undo last commit, discard all changes.

**9. Undoing Changes**

* git checkout -- <file> → Discard file changes.
* git revert <commit> → Create new commit to undo.
* git reflog → Show history of HEAD (recover lost commits).

**10. Stashing Work**

* git stash → Save changes temporarily.
* git stash list → List stashes.
* git stash apply → Apply most recent stash.
* git stash pop → Apply & remove most recent stash.
* git stash drop → Delete a stash entry.
* git stash branch <branch> → Create new branch from stash.

**11. Tagging**

* git tag → List tags.
* git tag <tag> → Create new tag.
* git tag -a <tag> -m "msg" → Annotated tag.
* git push origin <tag> → Push tag to remote.
* git push origin --tags → Push all tags.

**12. Other Useful Commands**

* git clean -f → Remove untracked files.
* git cherry-pick <commit> → Apply commit from another branch.
* git archive → Create a tar/zip of project files.
* git grep "text" → Search in repository.

**13. Advanced Workflows 🚀**

* git rebase <branch> → Reapply commits on top of another branch.
* git rebase -i HEAD~3 → Interactive rebase (squash, edit commits).
* git bisect start → Start binary search for a bug.
* git bisect bad / git bisect good <commit> → Mark commits in bisect process.
* git submodule add <url> → Add submodule.
* git submodule update --init --recursive → Initialize/update submodules.
* git worktree add ../newdir branch → Work on multiple branches simultaneously.
* git hook (scripts in .git/hooks/) → Automate workflows (pre-commit, pre-push).
* git filter-branch / git filter-repo → Rewrite history (cleanup, GDPR removals).
* git cherry -v <branch> → Show commits not merged upstream.

**14. Help and Version**

* git help <command> → Get help.
* git --version → Show Git version.

**✅ Tips**

* Use git status often.
* Commit small, meaningful changes with clear messages.
* Prefer branches for new features & fixes.
* Use rebase for clean history, merge for context-preserving history.
* Be cautious with history-rewriting commands (reset --hard, rebase, etc.).
* Never git reset --hard on shared branches (like main/master).