Git Command Reference

Setup

`git config --global user.name "Your Name"`
Set the name written into every commit you make.

`git config --global user.email you@example.com`
Set the e-mail written into your commits.

`git config --global init.defaultBranch main`
New repositories start on a branch called main.

`git config --global core.editor "code --wait"`
Use VS Code (or any editor) for commit messages.

`git config --list --show-origin`
Show every setting and the file it comes from.

Start

`git init`
Turn the current folder into a new repository.

`git clone <url>`
Copy a remote repository, with full history, into a new folder.

`git clone --depth 1 <url>`
Shallow clone: only the latest commit, much faster for big repos.

Snapshot

`git status`
What changed, what is staged, which branch you are on.

`git status -sb`
Short status with branch and ahead/behind counts.

`git add <file>`
Stage a file for the next commit.

`git add -p`
Stage changes hunk by hunk, answering y/n for each.

`git add -A`
Stage every change, including deletions and new files.

`git commit -m "message"`
Record the staged changes as a commit.

`git commit --amend`
Replace the last commit (message or content). Avoid after pushing.

`git commit --amend --no-edit`
Add staged changes to the last commit, keep its message.

`git rm <file>`
Delete a file and stage the deletion.

`git rm --cached <file>`
Stop tracking a file but keep it on disk (then add it to .gitignore).

`git mv <old> <new>`
Rename or move a file and stage it.

Inspect

`git diff`
Unstaged changes, line by line.

`git diff --staged`
What will go into the next commit.

`git diff main...feature`
What the feature branch changed since it left main.

`git log --oneline --graph --all`
Compact history of every branch as a graph.

`git log -p <file>`
History of one file with each change shown.

`git log --author="Name" --since="2 weeks ago"`
Filter history by author and date.

`git log -S "text"`
Find commits that added or removed a piece of text.

`git show <commit>`
One commit: message, author and the full diff.

`git blame <file>`
Who last changed each line, and in which commit.

`git shortlog -sn`
Commit count per author.

Branch

`git branch`
List local branches; * marks the current one.

`git branch -a`
List local and remote-tracking branches.

`git switch <branch>`
Move to an existing branch.

`git switch -c <branch>`
Create a branch and move to it.

`git branch -d <branch>`
Delete a merged branch (-D forces an unmerged one).

`git branch -m <new-name>`
Rename the current branch.

`git branch --merged`
Branches already merged into the current one: safe to delete.

Merge

`git merge <branch>`
Join another branch's history into the current branch.

`git merge --no-ff <branch>`
Always create a merge commit, even if a fast-forward is possible.

`git merge --abort`
Give up a merge with conflicts and go back.

`git rebase main`
Replay your branch's commits on top of main for a straight history.

`git rebase -i HEAD~5`
Edit the last 5 commits: reorder, squash, reword or drop.

`git rebase --continue`
Carry on after fixing a conflict during a rebase.

`git rebase --abort`
Cancel the rebase and return to where you started.

`git cherry-pick <commit>`
Copy one commit from elsewhere onto the current branch.

Remote

`git remote -v`
Show remote names and their URLs.

`git remote add origin <url>`
Connect the repository to a remote called origin.

`git fetch`
Download new commits from the remote without changing your files.

`git fetch --prune`
Fetch and remove references to deleted remote branches.

`git pull`
Fetch and merge the remote branch into yours.

`git pull --rebase`
Fetch and replay your local commits on top: no merge commit.

`git push`
Upload your commits to the tracked remote branch.

`git push -u origin <branch>`
Push a new branch and set it to track the remote.

`git push --force-with-lease`
Overwrite the remote branch only if nobody else pushed meanwhile.

`git push origin --delete <branch>`
Delete a branch on the remote.