Title: Git Commands You’ll Actually Use

1. One-time setup

- Set identity:
    - git config --global user.name "Your Name"
    - git config --global user.email "you@example.com"
- Recommended defaults:
    - git config --global init.defaultBranch main
    - git config --global pull.rebase false
    - git config --global color.ui auto
- Verify:
    - git config --list

2. Start or clone a repository

- New repo:
    - git init
    - git add -A
    - git commit -m "Initial commit"
- Clone existing:
    - git clone <repo-url>
    - cd <repo-folder>

3. See status and history

- Current state:
    - git status
- Diffs:
    - git diff (working tree vs last commit)
    - git diff --staged (staged vs last commit)
- History:

    - git log
    - git log --oneline --graph --decorate --all

4. Stage and commit

- Add files:
    - git add path/to/file1 path/to/file2
- Add all changes:
    - git add -A
- Commit:
    - git commit -m "Message"
- Amend last commit:
    - git commit --amend (don’t amend if already pushed)

5. Branching

- List:
    - git branch
- Create:
    - git branch feature/login
- Switch:
    - git checkout feature/login
- Create and switch:
    - git checkout -b feature/login
- Rename current:
    - git branch -m new-name
- Delete (merged):
    - git branch -d feature/login
- Force delete:
    - git branch -D feature/login

6. Remotes (GitHub)

- Add:
    - git remote add origin <url> (HTTPS or SSH)
- Verify:
    - git remote -v
- Change URL:
    - git remote set-url origin <new-url>

7. Push and pull

- First push and set upstream:
    - git push -u origin main
    - git push -u origin feature/login
- Subsequent:
    - git push
- Pull:
    - git pull
    - git pull --rebase (linear history)

8. Merge and rebase

- Merge into current:
    - git merge <branch>
- Typical flow:
    - git checkout main
    - git merge feature/login
- Rebase feature onto main:
    - git checkout feature/login
    - git rebase main
- During rebase:

    - git rebase --abort
    - git add <resolved-files> && git rebase --continue
    - git rebase --skip

Guidance:

- Merge preserves history and is safer for shared branches.
- Rebase keeps history linear but rewrites commits—avoid on shared branches.

9. Stash work-in-progress

- Stash:
    - git stash
    - git stash push -m "WIP: login flow"
- List:
    - git stash list
- Apply or pop:
    - git stash apply
    - git stash pop
- Drop specific:
    - git stash drop stash@{N}

10. Undo and fix mistakes

- Unstage file:
    - git reset <file>
- Discard local changes:
    - git restore <file> (new)
    - git checkout -- <file> (legacy)
- Discard all to last commit:
    - git restore --source=HEAD --worktree --staged .
- Move HEAD back:

    - git reset --soft HEAD~1 (keep staged)
    - git reset HEAD~1 (keep unstaged)
    - git reset --hard HEAD~1 (discard)
- Revert a commit:
    - git revert <commit-sha> (safe for shared history)

11. Compare and inspect

- Compare two refs:
    - git diff <ref1> <ref2>
- Show file from other branch:
    - git show other-branch:path/to/file

12. Tags

- Create:
    - git tag v1.0.0
    - git tag -a v1.0.0 -m "Release 1.0.0"
- Push:
    - git push --tags
- Delete:
    - git tag -d v1.0.0
    - git push origin :refs/tags/v1.0.0

13. Tracking and syncing

- Show tracking:
    - git branch -vv
- Set upstream:
    - git branch --set-upstream-to=origin/<branch>
- Fetch without merge:
    - git fetch

- Prune deleted remote branches:
    - git fetch --prune

14. .gitignore basics

- Add entries:
    - echo "node_modules/" >> .gitignore
    - echo ".env" >> .gitignore
- Show ignored:
    - git status --ignored

15. Submodules (if needed)

- Add:
    - git submodule add <url> path/submodule
- Init/update:
    - git submodule update --init --recursive
- Pull with submodules:
    - git pull --recurse-submodules

16. Common workflows

Initial project to GitHub (HTTPS)

- git init
- git add -A
- git commit -m "Initial commit"
- git branch -M main
- git remote add origin https://github.com/<user>/<repo>.git
- git push -u origin main

Daily cycle

- git pull --rebase
- edit files

- git add -A
- git commit -m "Implement X"
- git push

Feature branch flow

- git checkout -b feature/x
- work; git add -A; git commit -m "x"
- git push -u origin feature/x
- open PR; after merge:
    - git checkout main
    - git pull
    - git branch -d feature/x
    - git push origin --delete feature/x

Troubleshooting quick fixes

- No upstream:
    - git push -u origin <branch>
- “origin not a git repository”:
    - git remote -v
    - git remote add origin <url> or set-url
- Push rejected (non-fast-forward):
    - git pull --rebase
    - resolve conflicts
    - git push
- Undo last commit, keep changes:
    - git reset --soft HEAD~1
- Restore a deleted file:
    - git restore path/to/file (or git checkout -- path/to/file)