# Git Version control

**START HERE**

`git status`
What changed?

`git log --oneline --graph`
See history

`git add <file>`
Stage a file

`git restore --staged <file>`
Unstage a file

`git commit -m "<msg>"`
Save a snapshot

`git switch -c <branch>`
Start a branch

`git pull`
Get changes

`git push`
Share changes

**Working tree** your files → `git add` → **Staging area** next commit → `git commit` → **Local repo** history → `git push` → **Remote** origin

### What reset moves

| reset | HEAD | Index | Files |
|---|---|---|---|
| `--soft` | moves | kept | kept |
| `--mixed` | moves | reset | kept |
| `--hard` | moves | reset | replaced |

## CONCEPTS

Run git status often: it shows what changed and what to do next.

A branch is just a movable pointer to a commit, so branching is cheap.

Never rewrite history (reset --hard, rebase, force-push) on commits others have pulled.

## SETUP

`git init`
Create a repository here

`git clone <url>`
Copy a remote repository

`git config --global user.name "<name>"`
Set your name

`git config --global user.email "<email>"`
Set your email

## SNAPSHOTS

`git status`
Show changed and staged files
*Add -sb for a compact one-line-per-file view.*

## SNAPSHOTS (cont.)

`git add <file>`
Stage a file

`git add .`
Stage everything here
*Stages new, modified and deleted files below the current directory.*

`git commit -m "<msg>"`
Commit staged changes
*Without -m an editor opens for a longer message.*

`git diff`
Show unstaged changes
*Does not show untracked files.*

`git diff --staged`
Show staged changes

`git add -p`
Stage changes hunk by hunk
*y = stage, n = skip, s = split, q = quit.*

`git restore <file>`
Discard changes in a file
*Uncommitted work cannot be recovered afterwards.*

`git restore --staged <file>`
Unstage a file

`git commit --amend`
Rewrite the last commit
*Do not amend commits you have already pushed.*

## SNAPSHOTS (cont.)

`git rm <file>`
Delete a file and stage it

## HISTORY

`git log --oneline --graph`
Compact branch history
*Add --all to see every branch.*

`git show <commit>`
Show one commit

`git blame <file>`
Who changed each line
*Add -L 10,20 to limit it to those lines.*

`git reflog`
Everything HEAD has pointed to
*Lost a commit? Find its hash here, then git switch -c rescue <hash>.*

## BRANCHES

`git branch`
List local branches

`git switch <branch>`
Switch branch

`git switch -c <branch>`
Create and switch
*Starts at the current commit; add a start point to base it elsewhere.*

`git merge <branch>`
Merge branch into current

## BRANCHES (cont.)

`git merge --abort`
Cancel a conflicted merge
*Only works while the merge is still in progress.*

`git branch -d <branch>`
Delete a merged branch

`git rebase <branch>`
Replay commits on top of branch
*Fix conflicts, then git rebase --continue.*

`git rebase -i HEAD~<n>`
Edit the last n commits
*Change pick to squash, reword or drop in the editor.*

`git cherry-pick <commit>`
Copy one commit here
*Creates a new commit, so the hash differs.*

### merge, rebase or cherry-pick?

| Command | Result |
|---|---|
| `merge` | keeps both lines, adds a merge commit |
| `rebase` | replays your commits, linear history |
| `cherry-pick` | copies single commits |

## REMOTES

`git pull`
Fetch and integrate changes
*Equals git fetch plus merge (or rebase, if configured).*

## REMOTES (cont.)

`git push`
Upload commits

`git fetch`
Download without merging
*Updates origin/* only; your own branches are untouched.*

`git push -u origin <branch>`
Push and track a new branch
*-u lets a plain git push / pull work on this branch later.*

`git remote -v`
List remotes

`git remote add origin <url>`
Add a remote

## UNDO

`git revert <commit>`
New commit that undoes one
*Safe on shared branches: adds history instead of rewriting it.*

`git reset --soft HEAD~1`
Undo commit, keep changes staged

`git reset --hard HEAD`
Throw away all local changes
*Untracked files stay; use git clean for those.*

## STASH & TAGS

`git stash`
Shelve local changes

`git stash pop`
Restore the latest stash
*Pop removes the stash; apply keeps it.*

`git stash list`
List stashes

## RECIPES

### Start a feature

```
git switch -c feature/x
# work and commit
git push -u origin feature/x
```

### Tidy before a pull request

```
git fetch origin
git rebase -i origin/main
git push --force-with-lease
```

### Undo a pushed commit

```
git revert <commit>
git push
```

### Handy ~/.gitconfig

```
# ~/.gitconfig
[init]
  defaultBranch = main
[alias]
  st = status -sb
  lg = log --oneline --graph
```

## HELP

`git help <command>`
Manual page for a command