Git Version control

CONCEPTS

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

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

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

`git config --global init.defaultBranch main`
Name new branches main

SNAPSHOTS

`git status`
Show changed and staged files

`git add <file>`
Stage a file

`git add .`
Stage everything here

`git commit -m "<msg>"`
Commit staged changes

`git diff`
Show unstaged changes

`git diff --staged`
Show staged changes

`git add -p`
Stage changes hunk by hunk

`git restore <file>`
Discard changes in a file

SNAPSHOTS (cont.)

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

`git commit --amend`
Rewrite the last commit

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

HISTORY

`git log --oneline --graph`
Compact branch history

`git show <commit>`
Show one commit

`git blame <file>`
Who changed each line

`git reflog`
Everything HEAD has pointed to

BRANCHES

`git branch`
List local branches

`git switch <branch>`
Switch branch

`git switch -c <branch>`
Create and switch

`git merge <branch>`
Merge branch into current

`git merge --abort`
Cancel a conflicted merge

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

`git rebase <branch>`
Replay commits on top of branch

`git rebase -i HEAD~<n>`
Edit the last n commits

`git cherry-pick <commit>`
Copy one commit here

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

`git push`
Upload commits

REMOTES (cont.)

`git fetch`
Download without merging

`git push -u origin <branch>`
Push and track a new branch

`git remote -v`
List remotes

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

UNDO

`git revert <commit>`
New commit that undoes one

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

`git reset --hard HEAD`
Throw away all local changes

What reset moves

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

STASH & TAGS

`git stash`
Shelve local changes

`git stash pop`
Restore the latest stash

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