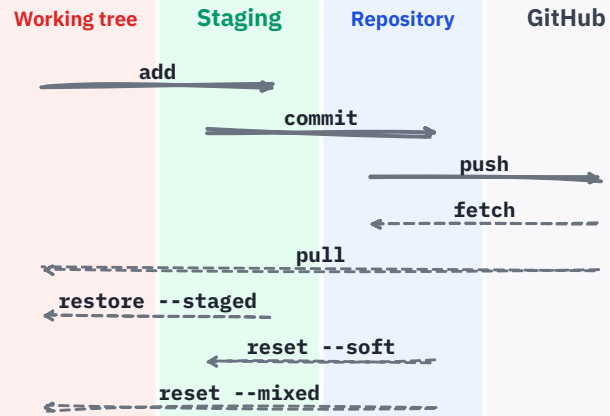




Record and undo

> Where each command moves your changes



> Get started

git init Start a repository here
git clone <url> Copy a repository from GitHub
git config user.name "<name>"
Your name on commits

> Files to ignore

One pattern per line in .gitignore

*.log log files
node_modules/ a whole directory
.env secrets and settings

> Record changes

git status See where things stand
git add <file> Stage a file
git add . Stage every change
git add -p Pick what to stage
git commit -m "<message>"
Commit what you staged

> Review changes and history

git diff Changes not staged yet
git diff --staged Changes you staged
git log --oneline Commit history, one line each
git show <commit> What a commit changed

> Discard or unstage

git restore <file> Throw away your edits ⚠
git restore --staged <file>
Unstage (keeps your edits)

> Fix or undo a commit

git commit --amend -m "<message>"
Fix the last commit
git reset --soft HEAD~1
Undo the last commit, keep the changes
git reset --hard HEAD~1
Undo it and discard the changes ⚠
git revert <commit>
Add a commit that reverses it (after a push)
git reflog Where HEAD has been (recover commits)

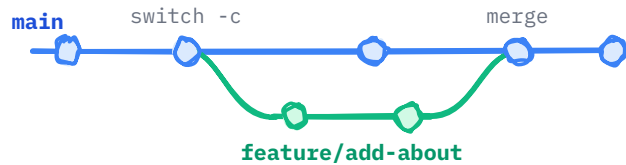
> Sync with GitHub

git remote add origin <url>
Connect a GitHub repository
git push -u origin main Push for the first time
git push Push
git pull Pull



Branch and work as a team

> Branches



git branch List branches (* is current)

git switch <branch> Switch branches

git switch -c <branch>

Create a branch and switch to it

git log --oneline --graph --all

See every branch as a graph

git branch -d <branch> Delete a merged branch

git branch -D <branch>

Delete it even if unmerged !

> Combine branches

Before: main moved on after you branched



After: git rebase main replays them on top



git merge <branch> Merge into the current branch

git merge --squash <branch>

Merge as one squashed commit

git rebase <branch>

Replay your commits on top of it

> Resolve a conflict

```

<<<<<< HEAD
color: blue;
=====
color: green;
>>>>>> feature/update-style
  
```

your branch (main)

incoming branch

Keep one side, delete the markers, git add, git commit

git rebase --continue

(rebase) Continue after fixing

git merge --abort Give up the merge

git rebase --abort Give up the rebase

> Work as a team



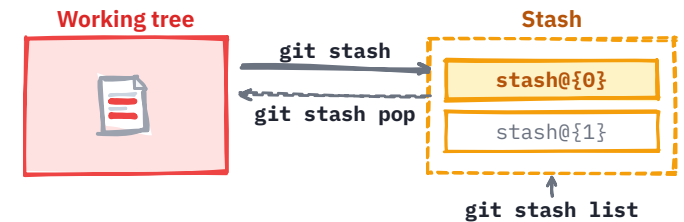
git push -u origin <branch>

Push a branch (before a pull request)

git fetch Fetch without merging

git tag <name> Tag the current commit

> Stash your work



The newest stash is stash@{0} (on top)

git stash Stash your changes

git stash pop Bring them back

git stash list List stashes

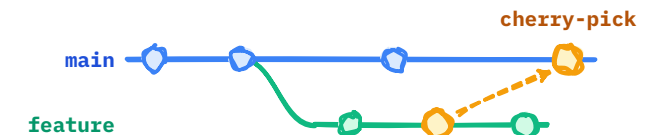
> Dig into and rewrite history

git log -- <file> Commits that touched a file

git log --oneline -S <text>

Commits that added or removed text

git blame <file> Who last changed each line



Copies just that commit onto your branch (the original stays)

git cherry-pick <commit> Copy one commit here

git rebase -i HEAD~<n>

Reorder or squash recent commits