



Git Tutorials: Reset, Revert, Amend, Reflog, and Stash

Part 1: Understanding git reset – soft, mixed, and hard


1. git reset --soft

 Goal: Undo the last commit but keep changes staged (in the index).


 Use Case: “Oops! I committed too early, but I still want to keep the same changes ready to go.”


 Setup & Action:


- mkdir reset-soft-demo && cd reset-soft-demo
- git init
- echo "one" > file.txt
- git add file.txt
- git commit -m "Initial commit"
- echo "two" >> file.txt
- git add file.txt
- git commit -m "Oops, bad commit message"
- git reset --soft HEAD~1
- git commit -m "Added line two"

 Teaching Tip: Soft reset is great for redoing your last commit without losing work or changes.

2. git reset --mixed (default)


 Goal: Undo the last commit and unstage the changes, but keep them in the working directory.

 Use Case: “I committed, but now I want to make a few more edits before re-adding.”


 Setup & Action:


- mkdir reset-mixed-demo && cd reset-mixed-demo
- git init
- echo "start" > app.txt
- git add app.txt
- git commit -m "Start app"
- echo "feature A" >> app.txt


- `git add app.txt`
- `git commit -m "Add feature A"`
- `git reset --mixed HEAD~1`

 Teaching Tip: Mixed reset is the default and helps you uncommit without deleting your changes.


3. `git reset --hard`

 Goal: Undo the last commit and discard changes from working directory and staging area.

 Use Case: “I completely messed up and just want to wipe everything and start fresh.”


 Setup & Action:


- `mkdir reset-hard-demo && cd reset-hard-demo`
- `git init`
- `echo "keep this" > notes.txt`
- `git add .`
- `git commit -m "Save initial notes"`
- `echo "oops" >> notes.txt`
- `git add .`
- `git commit -m "Bad commit"`
- `git reset --hard HEAD~1`


 Teaching Tip: Hard reset is the nuclear option. Great for cleaning your mess—just make sure it's backed up first!

Part 2: Git Tutorials – revert, amend, reflog, and stash

1. Git Revert – Undo with a Safety Net


 Goal: Undo a commit by making a new commit that reverses its changes.

 Use Case: “I broke something in a commit I already pushed. I want to undo it without rewriting history.”

 Setup & Action:


- `mkdir git-revert-demo && cd git-revert-demo`
- `git init`
- `echo "hello" > greet.txt`
- `git add . && git commit -m "Initial commit"`
- `echo "oops" >> greet.txt`


- `git add . && git commit -m "Add mistake"`
- `git log --oneline`
- `git revert <commit-hash>`

 Teaching Tip: Use revert when you've already shared your commit with others (pushed it), and don't want to mess with shared history.


2. Git Amend – Fix the Last Commit

 Goal: Modify the most recent commit — either its message or content.

 Use Case: “I forgot to add a file or mistyped my commit message.”


 Setup & Action:


- `mkdir git-amend-demo && cd git-amend-demo`
- `git init`
- `echo "important" > readme.md`
- `git add .`
- `git commit -m "Initial file"`
- `echo "notes" > notes.txt`
- `git add notes.txt`
- `git commit --amend`

 Teaching Tip: Amend is great before you push. After pushing, amending rewrites history — so be careful.

3. Git Reflog – Your Time Machine

 Goal: View and recover previous HEAD positions.

 Use Case: “I deleted something by mistake. Can I go back to where I was 10 minutes ago?”

 Setup & Action:

- `mkdir git-reflog-demo && cd git-reflog-demo`
- `git init`
- `echo "a" > file.txt`
- `git add . && git commit -m "Commit A"`
- `echo "b" >> file.txt`
- `git add . && git commit -m "Commit B"`
- `git reset --hard HEAD~1`
- `git reflog`
- `git checkout <that-commit-hash>`

🧠 Teaching Tip: Reflog saves your HEAD movement history — it's your emergency exit after a mistake!

4. Git Stash – Temporary Workspace Save

🎯 Goal: Save unfinished changes without committing.

✅ Use Case: "I'm in the middle of something, but need to quickly check another branch."

🔧 Setup & Action:

- `mkdir git-stash-demo && cd git-stash-demo`
- `git init`
- `echo "start" > work.txt`
- `git add . && git commit -m "Initial"`
- `echo "draft work" >> work.txt`
- `git stash`
- `git stash pop`

🧠 Teaching Tip: Stash is your temporary shelf — great for clearing your desk without throwing anything away.

Git Stash Command Reference

- `git stash` — Stashes tracked changes (default)
- `git stash -u` — Includes untracked files
- `git stash list` — Shows all saved stashes
- `git stash apply` — Reapplies a stash (keeps it in the list)
- `git stash pop` — Reapplies and removes the stash
- `git stash drop stash@{0}` — Deletes a specific stash
- `git stash clear` — Deletes all stashes