**Intermediate Git Tutorial**

*(Skill level: Comfortable with basics, ready to work in teams & handle tricky situations)*

**1. Branch Management (More Than Just main)**

Branches let you work on features without disturbing the main code.

**Create a new branch**

bash

CopyEdit

git checkout -b feature/user-login

*(shortcut for git branch feature/user-login && git checkout feature/user-login)*

**Switch between branches**

bash

CopyEdit

git switch main

or

bash

CopyEdit

git checkout main

**Delete a branch**

bash

CopyEdit

git branch -d feature/user-login # delete locally if merged

git branch -D feature/user-login # force delete

**Push a new branch to remote**

bash

CopyEdit

git push -u origin feature/user-login

**2. Merging & Resolving Conflicts**

Merging combines work from one branch into another.

**Merge example:**

bash

CopyEdit

git switch main

git merge feature/user-login

If there’s a conflict:

1. Git will mark files with <<<<<<< HEAD and >>>>>>> branch-name.
2. Edit to keep the correct version.
3. Mark as resolved:

bash

CopyEdit

git add conflicted\_file.js

git commit

**3. Stashing Your Work**

When you need to switch branches but don’t want to commit yet:

bash

CopyEdit

git stash

Later:

bash

CopyEdit

git stash pop # apply & remove from stash

git stash apply # apply but keep in stash

git stash list # see stashed items

**4. Viewing History Like a Pro**

**Pretty logs**

bash

CopyEdit

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

**Show changes for a commit**

bash

CopyEdit

git show <commit-hash>

**5. Reset vs Revert**

* **Reset** = moves the branch pointer (dangerous if pushed!)
* **Revert** = makes a new commit that undoes changes (safe for shared branches)

**Reset (local history rewrite)**

bash

CopyEdit

git reset --hard HEAD~1 # remove last commit completely

**Revert (safe)**

bash

CopyEdit

git revert <commit-hash>

**6. Cherry-Picking Commits**

Take a single commit from one branch and apply it to another:

bash

CopyEdit

git cherry-pick <commit-hash>

**7. Rebasing (Clean History)**

Rebase puts your commits on top of another branch — cleaner than merge but riskier for shared branches.

**Example:**

bash

CopyEdit

git switch feature/user-login

git rebase main

If conflicts appear, fix them, then:

bash

CopyEdit

git rebase --continue

**8. Tags for Releases**

bash

CopyEdit

git tag v1.0.0

git push origin v1.0.0

**9. Undoing Local Changes**

* Undo unstaged changes:

bash

CopyEdit

git checkout -- file.txt

* Undo staged changes:

bash

CopyEdit

git reset file.txt

**10. .gitignore Updates**

Add ignored files without clearing your repo:

bash

CopyEdit

echo "node\_modules/" >> .gitignore

git rm -r --cached node\_modules/

git commit -m "Update gitignore"

💡 **Pro Tips:**

* Use git pull --rebase to avoid merge commits in your own feature branches.
* Avoid git push --force on shared branches — use --force-with-lease if necessary.
* Always check git status before committing or switching branches.