**How to start a new repo?**

Install Git (if not already installed)

Initialize a new Git repository

Create and add files

Commit your changes

Create a new repository on GitHub

Link your local repository to GitHub

git init

git add filename

git commit -m "Initial commit"

git remote add origin https://github.com/<username>/<repository-name>.git

git branch -M main (skip if don’t want to rename default branch)

git push -u origin main

**How to merge code into main/master branch with ‘pull request’?**

Push code to github first and then create a pull request to merge that code into main/master branch.

If main/master branch is protected (with rules), we can’t make changes to it. In order to do that we have to create a pull request for it to be merged in, if we wish to merge the code into main/master branch.

**How does forking work on GitHub?**

Click on fork button and when forked successfully, use this command to clone into your machine.

**git clone <url of forked repo>**

**Workflow of how to create a pull request.**

1. First clone the project from github on local machine.
2. Create feature branch (git checkout –b <feature-branch-name>
3. Add Git commits
4. After changes push to github (Git push origin feature-branch-name)
5. Create pull request and wait for approval

**How to setup remote upstream?**

Lets suppose I have forked some code and started working on it. Now I want to setup a remote upstream to get the latest updates along with my changes I am making on my forked code.

Steps to do that:

git remote add upstream <url of original repo>

**how to..**

if I have been working on my forked repo and keep changing it and I also want to get the updates from original repo.

I would have to apply these steps:

1. git clone [https://github.com/<your\_username>/<your\_repo>.git](https://github.com/%3cyour_username%3e/%3cyour_repo%3e.git)
2. git remote add upstream <https://github.com/original_username/original_repo.git>
3. git checkout -b upstream-updates (Create a new branch to track the updates from the original repository. This will help keep your changes separate from the upstream changes)
4. git pull upstream main
5. git add . # Stage the resolved changes
6. git commit # Commit the changes
7. git push origin main.

Make sure the upstream-updates branch has no conflicts and its ready to be merged in. then checkout into main branch and use merge (git merge upstream-updates) command to get the updates from original repo to your own branch.

**How to delete a particular commit:**

Colt course video name of how to delete commits

Folder name (**10 - Undoing Changes & Time Traveling**)

Video name(**008 Undoing Commits With Git Reset**)

**If you want to keep the work but remove the commit, follow these steps.**

So if you run these command.

**Git log --oneline** (it will show commits)

**Git reset 4661ab9** (it will reset to that particular commit and remove commit after that, but it will keep the work saved in working directory)

However….

Deleting particular commit

====

git log --oneline (it will show commits)

choose commit hash

git rebase -i HEAD~2

git drop <commit hash> <commit msg>

**if conflicts occur, this might fix the conflict:**

git add . (keep applying this until all conflicts are done)

git rebase -continue (keep applying this until all conflicts are done)

====

**Chatgpt answer down below for how to delete a commit:**

1. Make sure you are on the branch where the commit to be deleted is located

**git checkout crmA-part-2**

\*\***REMEMBER TO USE ^**

1. Reset the branch to the commit before the one you want to delete

**git reset --hard e4e2e2bb1397b6353b58868b46912612da52c054^**

1. Force-push the changes to the remote repository

**git push -f**

The “**–hard**” option discards changes in both your working directory and the commit history.

The “**^**” after the commit hash means "the commit before."

**If you want to keep the commit locally but remove it from the remote repository, you can also use:**

**git push origin +cmaA-part-2**

**Git track and untrack process:**

Once a git started tracking a file/folder and if we decide at some point we don’t want to include it in future pushes. We would run this command.

git rm -r --cached .\piano.txt

**and one more thing I learned was. I deleted a file which was tracked by git which was wordpad file and when I ran this command “git reset .” for discard all the changes made accidentally git also restored deleted wordpad file. Solution I found for this was I ran these commands for first to untrack that file and then to delete it manually and then git didn’t recovered when I ran this command “git reset .”:**

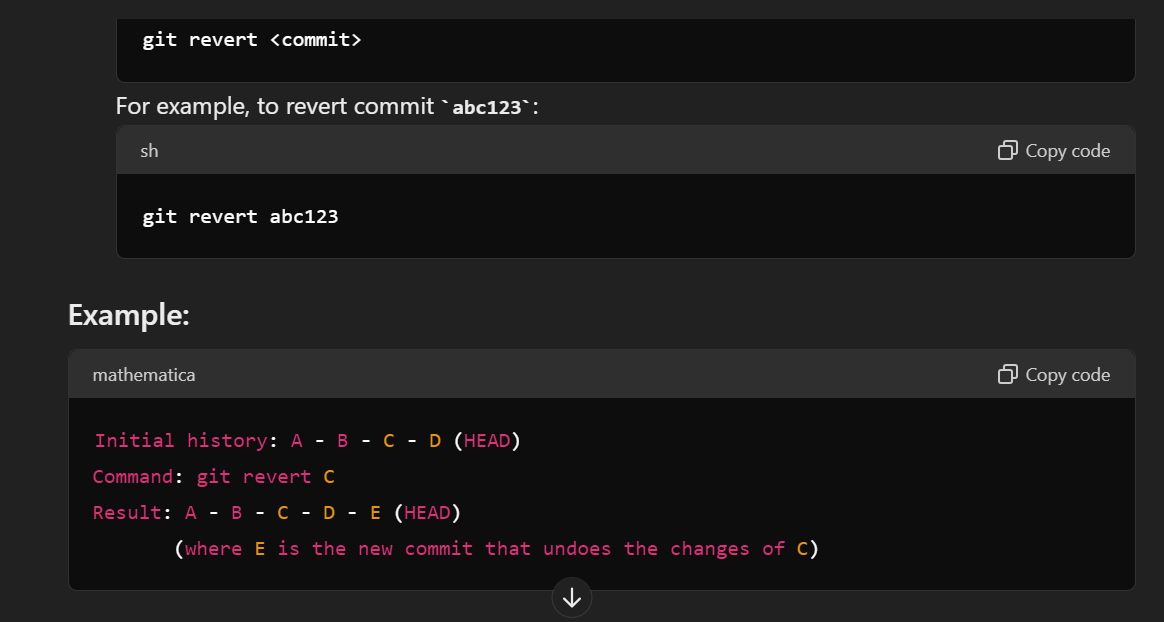
**example commands:**

git rm -r --cached .\piano.txt

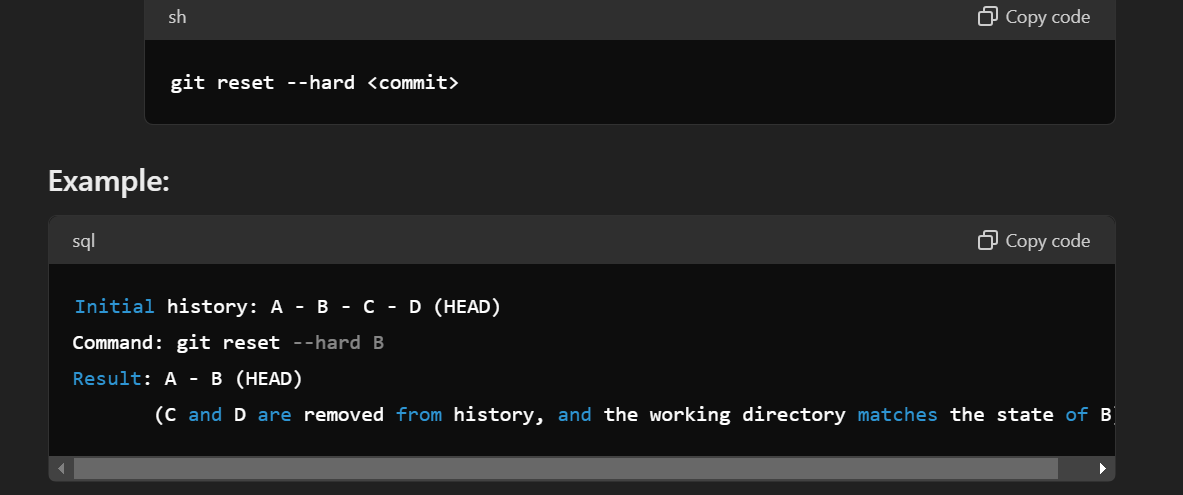
git reset .

**Summary:** If you want to discard all changes and it make conflicts like restoring some files/folder first untrack it and then delete those files manually.

**Difference b/w git revert and reset**



Revert will undo changes of a particular commit



Git reset will delete commits from current HEAD.