Git - Rollback

1. Undo Uncommitted Changes (in working directory)

Undo changes in a file (reset it to last commit):
`git checkout -- filename`

Discard all changes in working directory:
`git reset --hard`

This **deletes all uncommitted changes** (working directory + staging).

2. Unstage files (keep changes, remove from staging area)

`git reset filename`

Keeps changes in the file, but unstages them (from git add).

3. Undo Last Commit (keep changes)

`git reset --soft HEAD~1`

Moves HEAD back one commit, **keeps all changes staged**.

Undo Last Commit (keep changes but unstaged):

`git reset --mixed HEAD~1`

Undo Last Commit (remove it completely and discard changes):

`git reset --hard HEAD~1`

This is destructive. Changes are **lost** unless backed up.

4. Undo a Commit that was Already Pushed (Safe way)

Use **revert** to create a new commit that undoes changes from a previous commit:

`git revert <commit-hash>`

Good for **public history** where you don't want to change existing commits.

5. Delete All Local Changes & Commits (Clean Reset)

Reset branch to match origin:

```
git fetch origin
git reset --hard origin/main
```

Use this if you want your local main to match the remote main.

Quick Summary Table

Task	Command
Discard file changes	git checkout -- filename
Unstage a file	git reset filename
Undo last commit (keep changes)	git reset --soft HEAD~1
Revert a commit	git revert <commit-hash>
Full reset to remote	git reset --hard origin/main

🧪 Scenario: Feature Development with Mistakes

👣 Step 1: Create a feature branch and switch to it

git switch sprint1

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
==========================================================================
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git branch
==========================================================================
* main
  sprint1
  sprint2
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ ls
==========================================================================
file1  newfile1.txt  newfile3.txt  testdir2  testdir4
file2  newfile2.txt  testdir1      testdir3
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git switch sprint1
==========================================================================
Switched to branch 'sprint1'
Your branch is behind 'origin/main' by 1 commit, and can be fast-
forwarded.
  (use "git pull" to update your local branch)
==========================================================================
```

👣 Step 2: Make and stage changes

echo "new feature" >> newfile1.txt
git add newfile1.txt

Realize: You added the file by mistake.

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ echo "new feature" >>
newfile1.txt

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ cat newfile1.txt
================================================================================
new feature
================================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git add newfile1.txt
```

Verify status

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
================================================================================
On branch sprint1
Your branch is behind 'origin/main' by 1 commit, and can be fast-
forwarded.
  (use "git pull" to update your local branch)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   newfile1.txt
================================================================================
```

Undo staging (keep your change)

git reset newfile1.txt

Now newfile1.txt is modified but **unstaged**.

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git reset newfile1.txt
================================================================================
Unstaged changes after reset:
M   newfile1.txt
================================================================================
```

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
==========================================================================
On branch sprint1
Your branch is behind 'origin/main' by 1 commit, and can be fast-
forwarded.
  (use "git pull" to update your local branch)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   newfile1.txt

no changes added to commit (use "git add" and/or "git commit -a”)
==========================================================================
```

👣 Step 3: You don't want the change at all anymore

git checkout -- newfile1.txt

This reverts newfile1.txt to last committed state (removes your unsaved changes).

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ cat newfile1.txt
==========================================================================
new feature
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git checkout --
newfile1.txt

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ cat newfile1.txt
==========================================================================

==========================================================================
```

👣 Step 4: Make multiple changes and commit

echo "temp code" >> file1
echo "debug line" >> file2
git add file1 file2
git commit -m "Temp debugging code”

Oops! You didn't mean to commit that!

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ echo "temp code" >>
file1

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ echo "debug line" >>
file2

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git add file1 file2

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git commit -m "Temp
debugging code”
========================================================================
[sprint1 75dce09] Temp debugging code
 2 files changed, 2 insertions(+)
 create mode 100644 file2
========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
========================================================================
On branch sprint1
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

nothing to commit, working tree clean
========================================================================
```

Undo last commit (keep changes staged)

git reset --soft HEAD~1

You can edit the files, fix the commit message, and recommit if needed.

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git reset --soft
HEAD~1

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
========================================================================
On branch sprint1
Your branch is behind 'origin/main' by 1 commit, and can be fast-
forwarded.
  (use "git pull" to update your local branch)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   file1
    new file:   file2
========================================================================
```

👣 Step 5: Recommit with better message

git commit -m "Refactored file1 and file2”

Then realize — you still made the wrong change…

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git commit -m
==========================================================================
"Refactored file1 and file2"
[sprint1 7689b5a] Refactored file1 and file2
 2 files changed, 2 insertions(+)
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
==========================================================================
On branch sprint1
Your branch and 'origin/main' have diverged,
and have 2 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

nothing to commit, working tree clean
==========================================================================
```

Undo last commit again (keep changes unstaged this time)

git reset --mixed HEAD~1

Now your edits to file1 and file2 are still there, but **unstaged**.

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git reset --mixed
HEAD~1
==========================================================================
Unstaged changes after reset:
M    file1
M    file2
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
==========================================================================
On branch sprint1
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file1
    modified:   file2

no changes added to commit (use "git add" and/or "git commit -a”)
==========================================================================
```

👣 Step 6: You decide all changes are bad

git reset —hard

All unstaged changes are now gone — file1 and file2 go back to committed state.

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git reset —hard
==========================================================================
HEAD is now at a3e486c rollback lab
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
==========================================================================
On branch sprint1
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

nothing to commit, working tree clean
==========================================================================
```

👣 Step 7: You pushed a bad commit earlier by accident

Let's say you want to **undo** a pushed commit safely.

git revert <commit-hash>

This creates a new commit that reverses the effects of the bad one.

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git log
==========================================================================
commit xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (HEAD -> sprint1, origin/sprint1)
Author: TiagoPaquete <user_email>
Date:   Wed May 21 14:31:32 2025 +0200

    rollback lab

commit xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Author: TiagoPaquete <user_email>
Date:   Wed May 21 12:24:08 2025 +0200

    changes staged and ready to commit

commit xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (origin/sprint2, sprint2)
Author: TiagoPaquete <user_email>
Date:   Wed May 21 12:03:03 2025 +0200

    test changes in files
```

```
commit xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Author: T-Paquete <user_email>
Date:   Tue May 20 17:44:28 2025 +0200

    Update file2
================================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git log --oneline
================================================================================
xxxxxxx (HEAD -> sprint1, origin/sprint1) rollback lab
xxxxxxx changes staged and ready to commit
xxxxxxx (origin/sprint2, sprint2) test changes in files
xxxxxxx Update file2
xxxxxxx updated new commit tests
xxxxxxx test commiting changes
xxxxxxx testfiles
================================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git revert a3e486c
================================================================================
  GNU nano 7.2    /…/.git/COMMIT_EDITMSG
Revert "rollback lab"

This reverts commit a3e486c3b3c506c358a1efb9893d9de31606a79d.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch sprint1
# Your branch and 'origin/main' have diverged,
# and have 1 and 1 different commits each, respectively.
#
# Changes to be committed:
#       modified:   file1
#       deleted:    file2
#
================================================================================

[sprint1 3a7cdc8] Revert the Revert "rollback lab"
 2 files changed, 2 deletions(-)
 delete mode 100644 file2

================================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
================================================================================
On branch sprint1
Your branch and 'origin/main' have diverged,
and have 2 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

nothing to commit, working tree clean
================================================================================
```

👣 Step 8: You totally messed up and want to reset sprint1 to remote

git fetch origin
git reset --hard origin/sprint1

Now your sprint1 branch is exactly like it is on GitHub (remote).

```
tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git fetch origin
==========================================================================
Username for 'https://github.com': T-Paquete
Password for 'https://T-Paquete@github.com':
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git reset --hard
==========================================================================
origin/sprint1
HEAD is now at a3e486c rollback lab
==========================================================================

tiago-paquete@Ubuntu1:~/GitHubProjects/T-Paquete$ git status
==========================================================================
On branch sprint1
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

nothing to commit, working tree clean
==========================================================================
```

Summary of Commands Used in the Scenario

Action	Command
Switch to branch	git switch sprint1
Unstage a file	git reset newfile1.txt
Discard file changes	git checkout -- newfile1.txt
Undo last commit (keep changes staged)	git reset --soft HEAD~1
Undo last commit (keep changes unstaged)	git reset --mixed HEAD~1
Discard all changes	git reset --hard
Revert a bad pushed commit	git revert <commit-hash>
Hard reset to match remote	git reset --hard origin/sprint1