**Git Module**

The commands below assume you've navigated to the folder for the Git repo.

### See What Branch You're On

* Run this command:
  + **git status**

### List All Branches

**NOTE:**The current local branch will be marked with an asterisk (\*).

* To see **local branches**, run this command:
  + **git branch**
* To see **remote branches**, run this command:
  + **git branch -r**
* To see **all local and remote branches**, run this command:
  + **git branch -a**

### Create a New Branch

* Run this command (replacing **my-branch-name**with whatever name you want):
  + **git checkout -b my-branch-name**
* You're now ready to commit to this branch.

### Switch to a Branch In Your Local Repo

* Run this command:
  + **git checkout my-branch-name**

### Switch to a Branch That Came From a Remote Repo

1. To get a list of all branches from the remote, run this command:
   * **git pull**
2. Run this command to switch to the branch:
   * **git checkout --track origin/my-branch-name**

### Push to a Branch

* If your local branch **does not exist**on the remote, run either of these commands:
  + **git push -u origin my-branch-name**
  + **git push -u origin HEAD**

**NOTE:**HEAD is a reference to the top of the current branch, so it's an easy way to push to a branch of the same name on the remote. This saves you from having to type out the exact name of the branch!

* If your local branch **already exists**on the remote, run this command:
  + **git push**

### Merge a Branch

1. You'll want to make sure your working tree is clean and see what branch you're on. Run this command:
   * **git status**
2. First, you must check out the branch that you want to merge another branch into (changes will be merged into this branch). If you're not already on the desired branch, run this command:
   * **git checkout master**
   * **NOTE:** Replace **master**with another branch name as needed.
3. Now you can merge another branch into the current branch. Run this command:
   * **git merge my-branch-name**
   * **NOTE:** When you merge, there may be a conflict. Refer to **Handling Merge Conflicts**(the next exercise) to learn what to do.

### Delete Branches

* To delete a **remote branch**, run this command:
  + **git push origin --delete my-branch-name**
* To delete a **local branch**, run either of these commands:
  + **git branch -d my-branch-name**
  + **git branch -D my-branch-name**
* **NOTE:** The -d option only deletes the branch if it has already been merged. The -D option is a shortcut for --delete --force, which deletes the branch irrespective of its merged status.

To Know the HEAD:

git show-ref –head

Soft Reset:

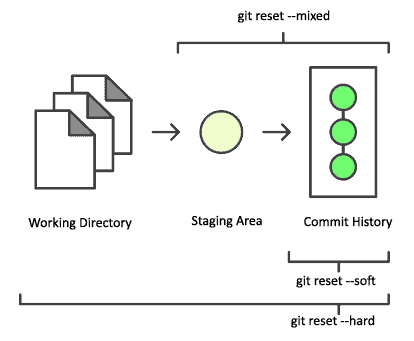
**git reset --soft HEAD~1**

**git reset --hard HEAD~1**

Git reset mixed option will move the HEAD.

It also updated the index (where HEAD points).

**Git reset mixed is the default option.**



* git diff View difference between Stage and Working Directory
* git diff --staged View difference between HEAD and Stage
* git diff HEAD View difference between HEAD and Working Directory

Diagram

Description automatically generated

Ex:

git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# new file: y

#

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: x

#

As you see, there is one file modified but not staged for commit, and a new file added that is ready to be committed.

git diff --staged will only show changes to files in the "staged" area.

git diff HEAD will show all changes to tracked files. If you have all changes staged for commit, then both commands will output the same.

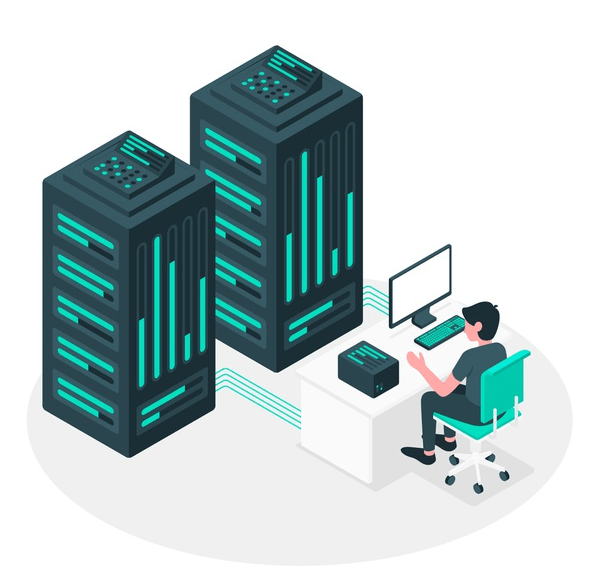
## Reset a local branch to match remote repository HEAD

You might need to do it often.

Why?

You made some changes in a branch. And in the meanwhile, your teammates push changes in the same branch.

Now you want to discard your changes and make sure you are in sync with the remote repository.



It is a two-step process

* Fetch all the changes from remote - ***$ git fetch origin***
* Remove  your changes - ***$ git reset --hard origin/main***

e104399@GH-B3QD1N2 MINGW64 /c/GK\_Work/codebase/Sample Project (master|REVERTING)

$ git clean -i

**Would remove the following item:**

Sample - Copy.txt

**\*\*\* Commands \*\*\***

1: clean 2: filter by pattern 3: select by numbers

4: ask each 5: quit 6: help

What now> Y

Huh (Y)?

**\*\*\* Commands \*\*\***

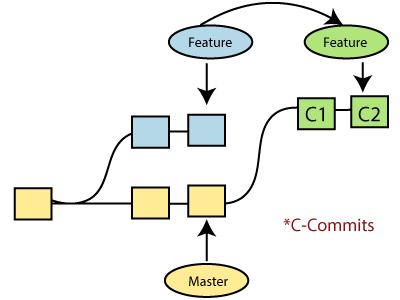
1: clean 2: filter by pattern 3: select by numbers

4: ask each 5: quit 6: help

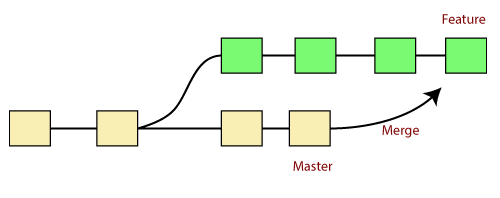
What now> 1

Removing Sample - Copy.txt

Rebase:



Merge:



|  |  |
| --- | --- |
| **Git Merge** | **Git Rebase** |
| Merging creates a final commit at merging. | Git rebase does not create any commit at rebasing. |
| It merges all commits as a single commit. | It creates a linear track of commits. |
| It creates a graphical history that might be a bit complex to understand. | It creates a linear history that can be easily understood. |
| It is safe to merge two branches. | Git "rebase" deals with the severe operation. |
| Merging can be performed on both public and private branches. | It is the wrong choice to use rebasing on public branches. |
| Merging integrates the content of the feature branch with the master branch. So, the master branch is changed, and feature branch history remains consistence. | Rebasing of the master branch may affect the feature branch. |
| Merging preserves history. | Rebasing rewrites history. |
| Git merge presents all conflicts at once. | Git rebase presents conflicts one by one. |

Squash:

git rebase -i HEAD~3

Revert :

Git revert sha value

<https://levelup.gitconnected.com/how-to-squash-git-commits-9a095c1bc1fc>

Git Cherry-pick

Git cherry-pick shavalue

Git Stash:

Git stash

Git stash save “Label”

Git stash list

Git stash apply stashid

Git stash pop index

Git stash show

Git stash show -p

Git stash drop

Git stash clear

$ git stash branch **<Branch** Name**>**

**To Know Current Branch:**

git branch --show-current