Yosafat Sánchez

GIT

Table of Contents

[GIT COMMANDS 2](#_Toc66097204)

# GIT COMMANDS

|  |  |
| --- | --- |
| Command | Description |
| git status | Show the files that were modified in the working directory |
| git add |  |
| git add [file] | add a file to get it staged for your next commit |
| git add . | add all modified files to stagedfor the next commit |
| git reset |  |
| git reset [file] | unstage a file but retain the changes |
| git reset . | unstage all files retaining the changes |
| git diff |  |
| git diff [file] | Show the changes of a certain file |
| git diff | Show all the changes for all the files |
| git diff –staged | Show all the changes for the staged files |
| git commit –m “[message]” | Commit all your staged work as a new commit |
| git branch |  |
| git branch | list your branches (the active one is marked with \*) |
| git branch –v | Shows also the commit subjet and Sha1 |
| git branch –D [branch name] | delete specific branch |
| git log |  |
| git log | show all commits on the current branch history |
| git log branchB..branchA | Show th commits that are on branchA that are not on the branchB |
| git checkout |  |
| git checkout [branch name] | switch to the specified branch |
| git checkout –b [new branch name] | Create a new branch based on the current one |
| git checkout [file name] | Revert all new changes for a specified file and return it to the latest commit |
| git checkout . | Revert all new changes for all the files and return it to the latest commit |
| git stash |  |
| git stash | Save modified and staged changes |
| git stash list | List all stacked stashes |
| git stash pop | Write working from top of stash stack |
| git stash drop | discard changes on the top of stash stack |
| git pull |  |
| git pull | Fetch and merge all the commits from the tracking branch |
| git pull [remote] [branch] | Fetch and merge all the commits from the specified branch |
| git push |  |
| git push [remote] [branch] | Transmit local commits to the remote repository branch |
| git push [remote] [branch] -f | FORCE THE TRANSMITION **\*\*DONT USE IT IF YOU DIDNT DO A SQUASH\*\*** |
| git cherry-pick [SHA1 of the commit] | copy specified commit to the current branch |
| git rebase -i HEAD~[no of the commits] | work with the number of the commits specified to rebase them in different ways (explained later) |

## Examples for the different uses of the commands

### git rebase

For example when you do a ***git rebase HEAD~3*** You are indicating that you are going to work on the 3 last commits starting on the newer one, so you are going to see a file like this one:

Text

Description automatically generated

You can see here all the commits that you specified with their sha1 and the commit message assigned, here you can do a lot of things with your commits just changing the first word on the line (“pick”) with the option that you need, save the file and close it, for example:

* Reword your commits (with r or reword)
  + To do a reword just change the “pick” word with a “r” or reword instead, for example:  
      
      
    save and close the file and you would get a new file like this:  
    Text

    Description automatically generated  
      
    just change the message, save and close the file, and you have reworded youre commit!
* Use it for squash, that merges all commits into the selected one (with s or squash)
  + To do a squash you need to let the first one as a “pick” just to tell git on what commit we are going to squash the other ones, and then change the “pick” word for a “s” or “squash” on the other commits, for example:  
      
    Text

    Description automatically generated  
      
    then you are going to see a new file like this one:  
      
    Text

    Description automatically generated  
      
    Where you can see all the commits involved in the squash, now you just need to comment all the messages of the older commits (with #) and let one line without “#” to tell git whats te new message, for example:  
      
    Text

    Description automatically generated  
      
    and then save the file and close, and that’s it! Now you have your commits squashed! (You can check it with a “git log”)