

 Git can store the files either on the **local repository or on the remote repository**.

Git: Distributed Control System

**Platrforms**

1. GiHub:

2. BitBucket

3. Gitlab

Basic: https://www.youtube.com/watch?v=v4TMaD9v5kM

**Error (403):**

remote: Permission to MrsColombo/elsevier-twitter-rest-api.git denied to JuliaRietveld.

fatal: unable to access ‘https://github.com/MrsColombo/elsevier-twitter-rest-api.git/ 88’: The requested URL returned error: 403

**Solution:** I had this problem too but managed to solve it, the error is that ur computer has saved a git username and password so if you shift to another account the error 403 will appear. Below is the solution

For Windows you can find the keys here:

control panel > user accounts > credential manager > Windows credentials > Generic credentials

Next remove the Github keys.

**Remember:**

When you use Git, you create one repository for each program or project that you

work on.

I suggest that you create all your Git repositories under a single directory

called git.

**Note** that it is a good idea to avoid spaces in directory (and file) names.

**How can you fix a broken commit?**

In order to fix any broken commit, use the command “git commit --amend”. When you run this command, you can fix the broken commit message in the editor.

### ****What is the difference between git pull and git fetch?****

Git pull command pulls new changes or commits from a particular branch from your central repository and updates your target branch in your local repository.

Git fetch is also used for the same purpose but it works in a slightly different way. When you perform a git fetch, it pulls all new commits from the desired branch and stores it in a **new branch** in your local repository. If you want to reflect these changes in your target branch, git fetch must be followed with a git merge. Your target branch will only be updated after merging the target branch and fetched branch. Just to make it easy for you, **remember the equation below:**

**Git pull(updated same branch) = git fetch + git merge(new branch crate)**

### Git reflog

Git reflog is storying all steps you did in your **local repository.** You can reach it by git reflog. It will list every step you did. You can step back to any state by checking out the commit\_id mentioned at the start of the lines. It is an opportunity to do an **undo if you made a mistake.**

### Git stash

With git stash, you can do a **local save of changes**. It can be useful when you would like to change to another branch or commit, but you don’t want to commit your current changes. Type simply git stash and it will save your local changes. You can do multiple times git stash, it will store each change separately in a stack structure. You can list your stashed changes by calling git stash list and you can get them again by **git stash pop.**

### ****git stash apply’****

If you want to continue working where you had left your work then ‘git stash apply‘ command is used to bring back the saved changes onto your current working directory.

**git status**

This command lists all the files that **have to be** committed. means

‘git status’ shows you the difference between the **working directory and the index,** it is helpful in understanding a git more comprehensively. **‘git diff’ is similar to ‘git status’,**

### git rm

Usage: git rm [file]

This command deletes the file from your working directory and stages the deletion.

### git branch

Usage: git branch

This command lists all the local branches in the current repository.

Usage: git branch [branch name]

This command creates a new branch.

### git merge

Usage: git merge [branch name]

This command merges the specified branch’s history into the current branch.

### git push

Usage: git push [variable name] master

This command sends the **committed changes of master** branch to your remote repository.

Usage: git push [variable name] [branch]

This command sends the branch commits to your remote repository.

Usage: git push –all [variable name]

This command pushes all branches to your remote repository.

To know if a branch has been merged into master or not you can use the below commands:

**git branch --merged** – It lists the branches that have been merged into the current branch.  
**git branch --no-merged** – It lists the branches that have not been merged.

**Rebase:**

In Git, the rebase command is used to integrate changes from one branch into another. It is an alternative to the “merge” command.