# Git [>>](marginnoteapp://note/E009A6DB-DD39-49A2-B4E8-FD7A05D66989)

One of the most popular Version Control Systems (VCS) is Git

## Git is a version control system for managing your source code history. GitHub is a hosting service for your Git repositories. [>>](marginnoteapp://note/E7D5FA8A-7822-4CBE-BB45-9D6711E82635)

## $ mkdir my\_git\_project $ cd my\_git\_project [>>](marginnoteapp://note/155281A8-95DF-41BF-9992-67D3B6AC148B)

If you want a repository in a new folder, then you will need to create a new directory (folder) and then switch to that directory. This is done at the command prompt (often indicated with $ or C:>) with the mkdir command for creating a directory and cd for navigating to a directory.
  
If you want to create the repository in an already existing folder, just navigate to that directory by typing cd directory\_name at the command line prompt.

## $ git init [>>](marginnoteapp://note/BEFA7865-EEC5-4536-91F7-4367572BC1BE)

Once inside the appropriate folder, use the git init command to turn the directory into an empty Git repository

### To see the files generated by git init, run the following command: $ ls -a [>>](marginnoteapp://note/37320FC3-5F3A-4DFF-9561-5118962AEE99)

## $ git commit -m "modified the button style" [>>](marginnoteapp://note/847B449A-3F4D-4C08-9B0B-F6A8A90EA1B0)

The git commit command saves the state of your project by adding snapshots of staged files to the repository.
The git commit can include the -m flag with a message describing what we've changed.

### $ git commit -a -m "commit all" [>>](marginnoteapp://note/6CA8E70D-9C93-426C-A75C-9781F1C3A158)

To commit modifications for every tracked file in the repository, use the git commit -a command

## $ git log [>>](marginnoteapp://note/714ACF78-5859-46D1-9EB6-E246D4E0552D)

If you want to see the history of your commits, use the git log command

## Cloning and Pushing [>>](marginnoteapp://note/66743EDA-0197-4BD2-BCD5-5FE4208E2620)

After committing the changes, the next step is pushing the local repository to the Git server on a remote location (such as GitHub or Bitbucket).  
After creating a remote repository we have the following options:
1) To download (clone) the repository and start making changes.
2) To initialize a local repository and then connect it with the remote one.

### $ git clone https://www.github.com/user/project\_name.git [>>](marginnoteapp://note/00F4E4EA-A8EF-4BF8-AE4D-460DC69CC706)

The clone command is used to download a remote repository

### $ git remote add origin https://www.github.com/user/project\_name.git [>>](marginnoteapp://note/5E3460E4-E636-482F-A27F-38A30EC23E69)

If you have already initialized a local repository, you can connect it to the remote one using the above command

### $ git push -u origin master [>>](marginnoteapp://note/1E3ED4C5-110A-4848-8991-744851FE8CA3)

After making our local changes and commits, its time to push the changes to the remote repository.
The push command tells Git where to put our commits.  
The name of our remote is origin and the default local branch name is master.  
The -u tells Git to remember the parameters, so that next time we can simply run git push and Git will know what to do.

## $ git pull origin master [>>](marginnoteapp://note/65A49E52-9EFF-4266-B3AF-2BDA5070865D)

We can check for changes on our GitHub repository and pull down any new changes by above command.
  
A local repository may have commits pushed by other users, who work on the repository. Get the latest updates on the project, especially when you are not the only one working on the project.

### $ git diff HEAD [>>](marginnoteapp://note/29E7BA6E-11D0-4328-A317-6CA29C3E52B2)

We can check what is different from our last commit by using the git diff command.
  
We want the diff of our most recent commit, which we can refer to using the HEAD pointer.

## Reset & Checkout [>>](marginnoteapp://note/6A243FB7-F36D-48AE-8C8C-5C273C5CBAC9)

It is good practice to regularly run git diff and reset files you accidentally changed.

### Running git diff --staged will show the changes you just staged. [>>](marginnoteapp://note/640A13BA-85E9-46E0-BBA4-6D427EDC170E)

### $ git reset 'file name' [>>](marginnoteapp://note/043D7B72-9863-4DE4-9C4E-6AB7681C4D08)

A stage can also be reset using the reset command
  
This removes the file from the staged status, meaning that all the changes will still remain in the file.

### $ git checkout -- 'file name' [>>](marginnoteapp://note/4E206EEB-FFA9-47DC-B391-4CDB0AA08605)

To reset the file to the latest committed version, the checkout command can be used

## Branches [>>](marginnoteapp://note/C97E78F8-8C94-46C5-8B8B-AF4F9E1A25F6)

Branching allows to make a "copy" of your working project and change it without affecting the main branch (master branch), giving an opportunity to work on the same project with different commits.
When you want to add a new feature or fix a bug - no matter how big or how small - you create a new branch to encapsulate your changes.
After the feature is done, you can merge it with the master branch.  
In order to see the list of your branches, run the git branch command in the project directory

### $ git checkout -b my\_new\_branch [>>](marginnoteapp://note/ECCC53A9-EA55-40D4-A635-9BE3055BE44D)

There is a shortcut to create and switch to a new branch

#### $ git branch my\_new\_branch [>>](marginnoteapp://note/FA5F55EC-6DDE-40CD-82EF-CF4864A1C7D5)

Creating a new branch is done using the branch command

#### $ git checkout my\_new\_branch [>>](marginnoteapp://note/BD7B1835-8A00-4733-BBD0-4A2898A2D026)

Then we need to switch to the branch using the checkout command

## git commit --amend -m "updated commit message" [>>](marginnoteapp://note/B246EAB6-CA0F-4622-BA07-C7A7C7F85B2E)

The git commit --amend command is a convenient way to modify the most recent commit.
It lets you combine staged changes with the previous commit instead of creating an entirely new commit.   
The -m option allows you to pass in a new message for the latest commit.

### Amending does not just alter the most recent commit, it replaces it entirely, meaning the amended commit will be a new entity. [>>](marginnoteapp://note/3AA88DD9-C0FE-48FD-A778-416237D11D8C)

## git stash [>>](marginnoteapp://note/36A095E8-529F-4BB4-B9F5-00CF77D18750)

The git stash command takes your uncommitted changes, both staged and unstaged, and saves them for later use.

### The git stash command temporarily caches any changes you've made to your working copy so you can switch to something else, and then come back and recover them later. [>>](marginnoteapp://note/9BD538CE-3662-4C42-8471-99D364C8888A)

After stashing you're free to make any changes, create new commits, switch branches, and perform any other Git operations. Then you can come back and re-apply your stash.

### There are two ways to re-load (re-apply) the stashed changes [>>](marginnoteapp://note/E6885459-03A2-44E2-901D-E01EB7995CD6)

#### git stash pop [>>](marginnoteapp://note/2B25375C-7ED2-4BD7-82DC-9DC7C73499A7)

pop will remove stashed changes from the stashed state.

#### git stash apply [>>](marginnoteapp://note/9F700751-D492-40A2-9950-BF3C5266FF21)

apply applies the same stashed changes to multiple branches.

## Merging Branches [>>](marginnoteapp://note/7938EEC0-7ABE-48A5-8C7C-C68B8CA71109)

### $ touch file{1..5}.txt $ git add . $ git commit -m "add files 1 to 5" [>>](marginnoteapp://note/4DB693DA-EC00-4AC5-A90F-4EF0CFE22472)

create and commit a few files on the master branch (main branch). The touch command below will create 5 text files, with names file1.txt, file2.txt, etc.

### git checkout -b test [>>](marginnoteapp://note/52DCCD5F-94C3-43D0-9F09-6DA037213BEC)

create and switch to our new branch

### touch file{6..10}.txt git rm file3.txt git rm file4.txt git add . git commit -m "add new files, remove 3&4" [>>](marginnoteapp://note/EF51B0A7-FA25-46B8-8B49-22F6E0B6BFA3)

We could just use the rm command, but it is better to use the git rm, which will not only remove the actual files from disk but will also stage the removal of the files for us.

### git checkout master git merge test [>>](marginnoteapp://note/CA49B971-981D-4245-BE86-36D7F3CBF4AF)

switch to the master branch and finally merge them together.  
git merge test tells the Git to merge the test branch into the current branch (the master branch in our example).

#### In order to remove a branch that you don't need anymore, use "git branch -d test" command. [>>](marginnoteapp://note/9DD0C9FE-0411-4954-B6F8-2DB74460A478)