**GITHUB**

GitHub is an Open-source Community where people around the globe work together on Open Source projects and make contributions. Also, GitHub lets you and your mates work together on projects. GitHub is also a code hosting platform for collaboration and version control.

Oh...Okay! What is **Version Control?**

* *Version control* is a system that records changes to a file or set of files over time to time so that you can recall specific versions later. This is what Git means.

So, Git is a version control system that lets you manage and keep track of your source code history. GitHub is a cloud-based hosting service that lets you manage Git repositories. If you have open-source projects that use Git, then GitHub is designed to help you better manage them.

Some Major tech companies also use some source control software, it's very useful for the good architecture of the software, helps you keep track of the changes of different files and syncing between different computers, it'd also really useful for those managers that like to use a number of lines for measuring work of a certain employee.

Some essential things you should be aware of:

* Repositories
* Branches
* Commits
* Pull Requests

**Repository**

A GitHub repository can be used to store a development project. It can contain folders or any type of files (HTML, CSS, JavaScript, Documents, Data, Images). A GitHub repository should also include a license file and a README file about the project.

**Branches**

A GitHub branch is used to work with different versions of a repository at the same time. By default, a repository has a master branch. Any other branch is a copy of the master branch. New Branches are for bug fixes and feature work separate from the master branch. When changes are ready, they can be merged into the master branch. If you make changes to the master branch while working on a new branch, these updates can be pulled in.

**Commits**

At GitHub, changes are called commits. Each commit has a description of why a change was made.

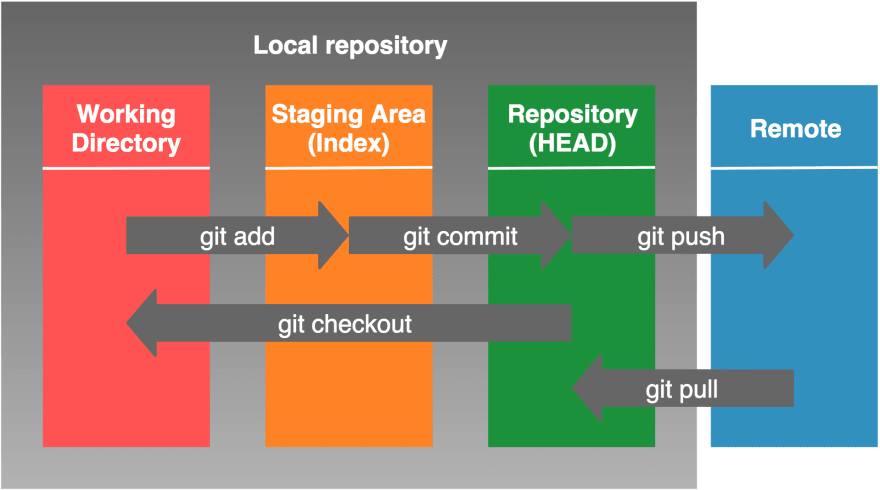
**Pull Requests**

Pull Requests are the heart of GitHub collaboration. With a pull request, you are proposing that your changes should be merged with the master. Pull requests show content differences, changes, additions, and subtractions in colors (green and red).

As soon as you have a commit, you can open a pull request and start a discussion, even before the code is finished.

A great way to learn GitHub, before working on larger projects, is to open pull requests in your own repository and merge them yourself. You merge any changes into the master by clicking a "Merge pull request" button. After merging you can delete the branch by clicking a "Delete branch button".

Check out this self-explanatory reference image before continuing further.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--1Mashcev--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/i/yslouwacz4y0b0h246ly.png)

I hope you got what these things mean. Now let me give you git commands you need.

But At first, you should be aware of basic Windows commands. If you are not, then check out my post [Basic Windows Commands every one MUST know](https://dev.to/mahithchigurupati/basic-windows-commands-every-one-must-know-53ho)

Let's start from the beginning.

Firstly, Open the folder which you want to create a repository in GitHub for either using Git bash option or by entering the path manually using common windows commands.

**1)git init** - This is the first thing you should do to initialize a .gitinit file for the folder.

**2)git status** - At any point of time, you can type git status to know the status of your git repo folder.

**3)git add *file-name.extension*** - The second step is to enter this command got add followed by filename with extension. If there are multiple files you want to upload to GitHub simply enter git add followed by filename with its extension. This will add all the contents of the folder.

This will adds the new or newly modified file-name to Git's staging area (index).

**git add .**

* The period parameter for the git add command will recursively add all new and newly modified files.

**4) git commit -m "*commit-message*"** - This is the command you should enter next. Type git commit followed by -m followed by a message you want to add for this commit. The message must be entered within double-quotes.

This command is to Commit all files currently in Git's staging area. The -m parameter allows for a commit message directly from the command line.

**git commit -am "*commit-message*"** Use the -a parameter with the git commit command to directly commit newly modified tracked files. Warning: Only do this for small changes. Tracked files are files that have been previously added to Git (committed or staged).

Now you have successfully created a commit. The steps you just performed are for working locally. Now, let's move to work remote, I mean to upload files into GitHub.

**Creating a remote repository**

5)Login to your account @[GitHub](https://github.com/) Then create a new repository by clicking on + icon, give a name and description to your repository, and save it.

6) Come back to your local system and perform the commands as shown below.

**Creating a remote repository reference**

**7) git remote add *remote-name remote-repository-location*** Using git remote add command allows us to associate a remote repository. Normally, you want to paste in the full URL for the remote repository given to you by your Git host (GitHub). By convention, the first or primary remote repository is named origin.

**List Git's Remotes**

**8) git remote -v** The git remote command lists the names of all the remote repositories and the -v parameter (verbose) will display the full URL of the remote repository for each remote name listed

**Send Changes to Remote**

**9) git push -u *remote-name branch-name*** The git push sends all your local changes (commits) on branch branch-name to the remote named remote-name. The -u parameter is needed the first time you push a branch to the remote.

**Receive Changes from Remote**

**10) git pull *remote-name branch-name*** The git pull receives all your remote changes (commits) from the remote named remote-name and on branch branch-name.

**Unstage File**

11)**git reset HEAD *file-name*** - Following the above command will "unstage" the specified file from Git's staging area (aka index).

**Backout Working Directory Changes**

**12)git checkout -- *file-name*** - Following the above command will back out any changes made to the specified file and replace it with the version last committed in Git

**Seeing Repository History**

**13. git log** - Git's log command displays the repository's history in reverse chronological order. The no-params version displays the standard view.

**Removing a file using Git**

**14. git rm *file-name*** - Removing a file using Terminal

**15. rm *file-name*** This removes the file outside Git's knowledge

**Updating Git's Index (staging area)**

**16. git add -u** - The -u parameter will recursively update Git's staging area regarding deleted/moved files outside of Git.

**Making a directory (folder)**

**17. mkdir *folder-name*** - The mkdir command is a nearly universal command for creating a directory/folder.

**Moving a directory (folder)**

**18. git mv *source destination*** - The git mv command will move the source (file or folder) to the destination with Git.

These are some basic and must know commands to perform your operations to use GitHub. However, there are some additional things you should be aware of as given below as bonus.

**BONUS:**  
**19).gitignore file** - If you don't want to push some files into remote repository then you can create a .gitignore file using notepad .gitignore command and type the names of files you don't want to push to GitHub.

**20)readme.md file** - Having a Readme file for every repository is a best practice as it will let people know what actually a particular repository about. So always maintain a readme file.