|  |  |
| --- | --- |
| **Description** | **Command** |
| **Cloning a repository** | |
| This command **clones the repository** to local machine/desktop | **git clone git@<URL\_to\_repository>.git Eg. git clone git@github.com:colaberry/Filtered.AI.git** |
| **Pulling/Downloading latest repository and code from remote (Github.com)** | |
| **Pull** the **latest version** **of repository** **from** the master source on **github.com**. Note that this command pulls data from whichever branch, terminal control is currently on. Say in local terminal if you are on master, then latest version of master repository is pulled into your local machine. If you are on another branch, latest version of that branch alone is pulled into your machine. | **git pull** |
| **Branch Operations** | |
| **Creating a new branch** can be done using the command to the right. Note that when issuing this command on the terminal, make sure you are on 'master' branch, because this command creates a branch from the base you are on. If you are on a branch and not on master, and then issue this command, the new branch will be created with the current branch (on which you are currently on) as base, and not the master. | **git checkout -b <new\_branch\_name> Eg. git checkout -b python-intro** |
| **Switching/Traversing** between **branches.** This can be done using the git checkout command without any tags. This can be done to switch from current branch to another existing branch. | **git checkout <branch\_name> Eg. git checkout -b python-advanced** |
| To **find** **which branch** you are on, use the git branch command. | **git branch** |
| To **find** **difference between** one **branch** and another, use git diff. This should generally be done while being on master (or the branch) which you consider as base and want to merge the code into. | **git diff <branch\_name>** |
| To **merge content** of another **branch** into base branch. Do this while on master (or any branch which you want to consider as base for this particular merge), to merge to master. The squash tag allows you to squash multiple commits into a single commit and merge it, instead of cluttering the base branch with multiple commits. | **git merge -squash <branch\_name>** |
| To **delete** a **branch** **from local** machine, use the command on the right. Do this while you are on master branch. You cannot delete a branch on which control is currently on, i.e. base branch. | **git branch -d <branch\_name>** |
| To **delete** a **branch** **from remote**, use the command on the right. Do this if you have already pushed/committed a local branch onto remote location. If branch is new and has never been pushed to remote, then just delete it from local machine using above command. | **git push origin --delete <branch\_name>** |
| **Adding and Committing files to local repository and pushing to remote** | |
| **Adding new/edited files** to the repository folder. After doing the edits on files and physically pasting/saving the editted files in the local machine's required git repository directory/folder, change path to respective folder in bash/terminal. Then use the 'git add' command. | **git add <file\_name> Eg. git add python-basics.ipynb (or) git add picture1.jpg** |
| **Adding all files** in current directory to git repository | **git add .** |
| **Deleting files in repository** folder. Use the command to delete from git repository index, only after physically deleting the file from the said directory. | **git rm <path/file> Eg. git rm python-basics.ipynb (or) git rm DA/course/lessons/python-basics.ipynb** |
| **Performing a commit** of all code/files added to the current repository is the equivalent of saving all changes made to the repository. | **git commit -m "commit comments" Eg. git commit -m "Added notebook"** |
| **Pushing the code** from local to master. If you do this from branch then use set-upstream origin. If done on master directly, then simply type in 'git push'. | **git push --set-upstream origin <branch\_name> Eg. git push --set-upstream origin test\_branch** |
| **Merging multiple commits** in vi editor. While merging content from a branch (which had multiple commits over time), to master, it is advisable to combine all commits and push to master as a single commit. This is called rebasing commits and can be performed in the vi editor. The vi editor can be opened using the command on the right. Do this while you are on the branch and not on the base or master, to which you want to merge. | **git rebase -i <master>** |
| **Within the vi editor**You may have many comments based on the actions. If you have multiple comments it may be messy so to change the comments we do some editing. Comments are displayed at the top of the vi editor. Pick is one of the keywords. If you want to squash the second comment, click before pick (not anywhere else) and delete pick. Then press “I” key to insert new commands. Type in “s” or “squash”. Check spaces as well. Then once press Esc to get into normal mode and then type in “:wq” to quit |  |
| **Handling untracked files** | |
| When you have untracked files in the repository, they are not added to the commit. It may become tricky as the git control doesn’t allow you to re-do the file addition step (git add) for the same file again. So how to deal with untracked files | |
| **Viewing untracked files** | **git clean -n** |
| **Deleting all untracked files** | **git clean -f** |
| **Adding all untracked files** | **git add -all** |