Complete Git and GitHub - Latest

Why git and GitHub?

- To maintain history of the project
- To know who and when made changes to the projects
- GitHub is used to share your code with the world.
- GitHub is one of the most popular git hosting platform among others.

Untracked files

The files whose history is not been saved yet. We need to stage these files first using command git add filename. Then save using command git commit -m “message”.

Linux commands

ls - List
mkdir project - Creates folder/directory named project.
ls -a - List all files even the ones which are hidden.
ls .git - Lists what’s inside the .git folder/directory.
touch filename - Creates a new file with filename.

Git commands (Known)

- **git init**
- **git status**
- **git add .**
- **git commit -m “message”**
- **git restore --staged filename**
- **git log**
- **git remote add origin repository_url**
- **git clone repository_url**
- **Git checkout branchname**
- **git remote -v**

Git Commands(New)

- **git reset 33bb9f7523e11d55b3773d0f900fda97cd98b533** - This command with the commit hash deletes all the commits above that particular commit.
- **git reset --hard 49f66576100aa1e9cfb436548c73adc3609d47bf** - This is another command to delete a commits above the specified commit.
- **git stash** - To send modified file to the back stage to work on later and get the working directory clean without committing those changes.
- **git stash pop** - To get back the stashed files to the staging area.
- **git stash clear** - To delete/clear everything inside stash.
- **git push origin master** - To push files in the master branch.
- **git remote add upstream upstream_url** - Adds the main repository url of the forked repository.
- **git branch name** - Creates a branch with a given name.
- **git push origin branchname -f** - Force push files to the branch.
- **git fetch --all --prune** - To fetch all the branches and also the deleted branches from the upstream.
- **git reset --hard upstream/main** - Resets the main branch of origin to the main branch of upstream.
- **git pull upstream main** - Resets/updates the main branch of origin to the main branch of upstream.
- **git rebase -i 49f66576100aa1e9cfb436548c73adc3609d47bf** - Merges the commits above together. More details below.

Deleting commit

Since the commits are stacked on to one another, you cannot just randomly delete any one commit from middle.

While deleting a commit all the commits made after that commit will also deleted.

To delete a commit you have to specify the hash value of the previous commit to the to be deleted commit in the git reset command.

The other way to delete a commit is using git reset --hard command. This will reset it using hard flag.

Stashing Changes

Stashing changes basically means sending the modified files to the back stage to get a clean working directory/branch without having to commit those modified changes.

All the changes that you dont want to use currently can be sent to the stash and you can get back the stashed files whenever you need to work on them.

To stash files first you need to send them in staging area and then to the stash.

You can also clear the stash and delete all the files inside it if you don’t wish to work on them anymore.

GitHub

Why Fork?

If you want to work on a remote project of which you dont have access to, you fork the project and make a copy of that project for your self to work on in your own account.
Since fork creates a copy a project into your own account, you can do whatever you want with it. It will not reflect into the main project until and unless the people who have approval for that project merge your code via a pull request.

You can clone the forked project on your local computer and start working on it.

Upstream

It is the url from where you have forked the project which is the url of the main project.

So the origin is the personal project url and the upstream is the main url of the forked project.

Pull Request

Whenever you create a copy of a main project and added some changes to the copy of that project and what those changes to be visible in the main project. You request for it to the owner of that main project using the pull request.

Never Commit on the main branch

Imagine that you are working on 10 projects. And you are working on 10 different features. And for every features you just create a one pull request.

How difficult would it be to review your code and to have the discussion?

So if you are working any feature or a bug fixes create a new pull request for each of them. This is why you should create separate branch for each feature or fixes since each branch can have only one pull request.

In simple language one branch is associated to one pull request.

So multiple branches means multiple pull requests.

Force Push

You can remove a commit on a remote repository via force push. It is used when you already have may be wrong commits pushed on your repository.

Making forked project even with the main project

Method 1:

- First go into the main branch of your local foked project
- Now fetch all the commits via the **git fetch --all --prune** command.
- Then reset the main branch of your origin to the main branch of upstream via **git reset --hard upstream/main** command.

Method 2:

You can use the **git pull upstream main** command to update your forked project with the main upstream project.

Git pull command internally does the same thing as method 1.

Method 3:

Go to the forked project on GitHub and click on fetch and merge button.

Here you don’t have to run any commands, that button will automatically make your forked project even with the main project.

Squashing Commits

Merging multiple commints together a one commit.

Method 1: Traditional

You can reset to the commit which is below all the commits which you want to merge together via the git reset commit_hash command.

This will bring all the commits in above the specified commit in the unstaged area.

Now you can those commits in staging area and make one single commit again to merge all the unstaged commits together.

Method 2: Using git rebase command

You merge multiple commits using **git rebase -i 49f66576100aa1e9cfb436548c73adc3609d47bf** command.
This command contains -i that is interective environment and the hash code of the commit of above which all the commits to be merged together.

This command will open up a vim editor. All the commit messages will have pick at the start.

Pick means the commit you want to use.

You can type s infront of all other commit messages you want to merge as one. The letter s basically means squash which means the commit message will be used but will be merged into the previous commit.

The editor will also give you a list of all other commands you can use with description. You can also try using them.

Merge Conflicts

When two people make changes in the same line of code in the same file in different branches. Now when those branches are to be merged together, the git be confused on which line of code is to be considered and throw a merge conflict.

This gives user the choice to choose which line of code to be consider while merging the branches together. So this has to be resolved by the user manually by removing one of the change.