To see GIT version ***git –version***

To set global variable of GIT

***git config --global user.name “Sibajee Ray”***

***git config –global user.email*** [***sibajee.ray@gmail.com***](mailto:sibajee.ray@gmail.com)

This will set ur name and email as git global variables. You can see all the config variables by

***git config --list***

To see all files and Folder in a Dir---Open CMD in that Directory, Then ***dir***

To see all folder and files there ***dir /a***

To see only the hidden files there ***dir /a:h***

To create .git folder(Git Local repository) in that directory ***git init*** (Hidden file). If .git folder is not inside the folder, tey u cannot run any git command in that directory. It will say this directory is not a git repository

Check the status ***git status***

This will list all the Tracked and Untracked files. Untracked means not synced with git repository. U can make it tracked by commit command. Here the .gitignore required.

Create .gitignore file. List the file name here which u don’t want to add in repository. Ex- Config files or any npm\_module.

Command to create file in windows.  ***echo > file\_name***

***echo > .gitignore***

Create this file in every folders of your project structure from where you don’t want any file to go to repository. Add Folder/File name as it is, or u can use wildcard blocking. Ex- \*.txt, \*.project

There are three stages in git operations

1) Working directory (Where ur project files are)

2) Staging Area(Choosing & preparing folder/files which we want to commit & which we Don’t)

3) .git Area(after Commit. Local Repository)

**Staging**

Add files to staging area every time before you run commit.

Add to staging Area

We have to stage the files before we commit those in repository. We can add files individually to stage area by  ***git add file\_name.***  For mass staging ***git add***  ***-A***

Files are still not added to git repository. We need to run commit command for that.

Remove from staging

To remove single file ***git reset file\_name***

To remove all files ***git reset***

After Staging if we give ***git status*** command, all the files will be shown as tracked files. And if will show “ *No commit yet. Changes to be commited*”.

Commit:

Commit command reflect the change from our working dir to repository. If we changed code in working dir and did not commit it, the changes will not be done in codes in repository.

***git commit***

This will reflect changes in only on the files added in staging area, not the files which are not in staging area.

Note: so always check ***git status*** before ***git commit.*** So that u can find out changes would be made on which files.

Commit with message:

Generally people commit changes with a commit message, just for reference why this change is for ?! ***git commit –m “Write All Your message here inside quote “***

Now after the commit if we will check ***git status*** , We will see “*Working directory clean. Nothing to commit*” until we make change in the file at our working dir. If any changes made there after, it will show “changes to be commited…modifile: File\_name”.

After commit, if we modified a file, we can see which are the modifications we have done in any file by

***git diff***  command. After each modification we hv to commit again to reflect the change in our local git repository.

**Note:** The change in codes are still not reflected to Remote Repository. Changes Took place only in local repository. For remote one we hv to run **push** command.

To see the commit log ***git log***

It will show author, commit date and commit message

**GIT Clone**

If one project is already present in remote repository and you want to get it in your working directory and work on that then u have to run clone command.

***git clone url\_of\_remote\_repository your\_working\_directory***

***ex-*** *git clone* [*https://github.com/sibajee/myrepo.git*](https://github.com/sibajee/myrepo.git) *D://code/project*

*git clone* [*https://github.com/sibajee/myrepo.git*](https://github.com/sibajee/myrepo.git) *. (to add on current directory, only dot)*

***git remote –v*** and ***git branch –a*** will show view and list of branch of the remote repository respectively

**GIT PULL AND PUSH**

These pull and push we use to interact our local and remote repository of git. To push the data from local repository to remote or made the changes reflected to remote repo

Add remote repository before u push ***git remote add origin ulr\_of\_repo***

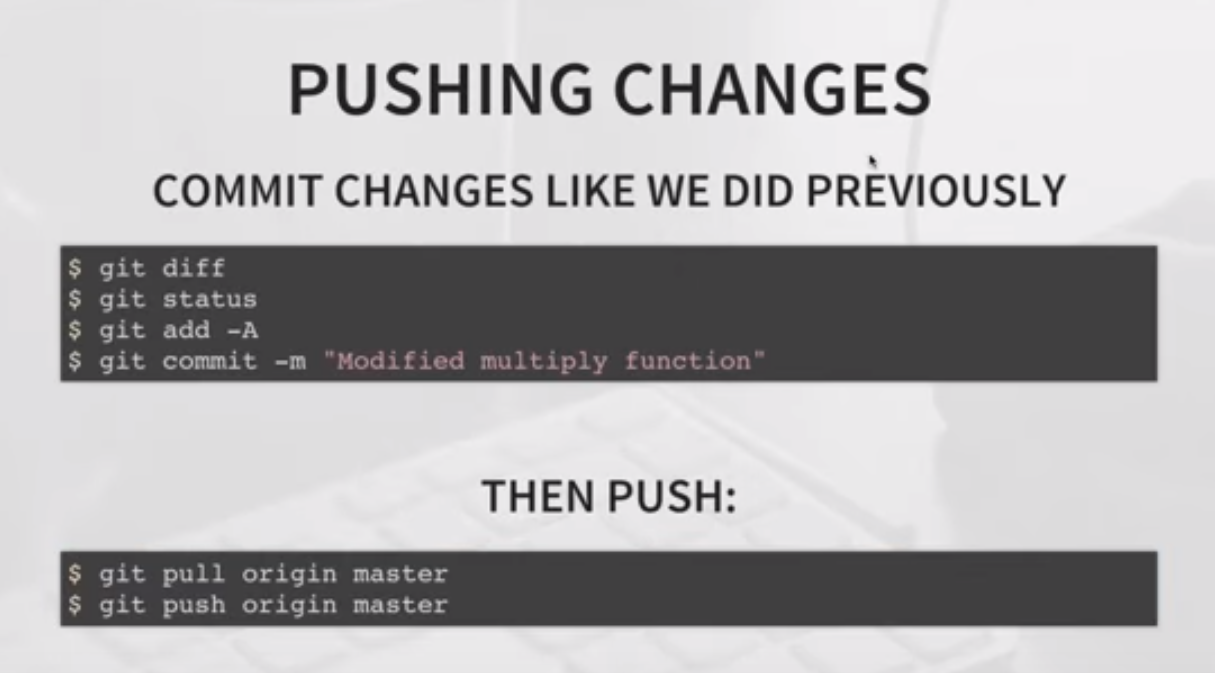
To change existing URL ***git remote set-url origin ulr\_of\_another\_repo***

***git push -u repository\_name branch\_name ex :* git push –u origin master**

If any changes doe in the code in any file by another developer, to get those changes in the local repository of your computer ***git pull repository\_name branch\_name***

***git pull --allow-unrelated-histories origin master***  if it shows **fatal: refusing to merge unrelated histories**

**NOTE:** We always should run **git pull** once before we run **git push.** Because if some other developer made changes in another file, which is under his maintainace, and then he pushed the code to remote repository, now if you push the code without pulling, then your code will make change in every file of remote repo and his changes will be overridden with your files.



**GIT BRANCH**

To see all the branches ***git brach***

Create New Brach ***git branch new\_branch***

Now run  *git branch.* You can see two branches. Master and New\_branch. But \* sign will be on master branch. Means even though repo has 2 branches, Master is the current working branch.

To change Current working branch ***git checkout new\_branch***

Now run *git branch,* \* sign will be on New\_branch. Any commit, push, pull operation will be done on New\_branch. Not on Master branch.

Reflect the new branch in remote repo

***git push –u repo\_name new\_branch,***  In 1st push after creating branch in local repo, we hav to run push with –u. That will create new\_branch in remote repo and use it as current branch.

Git Merge

To merge one ranch with another 1st switch to that branch. Lets we hv to merge new\_branch to master, then

* Switch to master ***git checkout master***
* Pull all the changes
* Check all the branches merged with this, if new\_branch is already merged or not.

***git branch --merged***

It will show all the branches already merged with master.

* Now merge new\_branch with master by  ***git merge new\_branch***
  + ***new-branch***  is merged with master now, but in local repo. Now if we’ll check **git branch –merge,**  it will list the merged branch.
* Now delete the new\_branch from local repo as it is already merged to master
  + ***git branch –d new\_branch***
  + To delete this from remote repository and push the final changes
    - **git push repo\_name –delete new\_branch**