We can search anything from git /learn anything

**Previously** :

How multiple members share code e.g. **Dockers / email /Dropbox**

**What is Git?**

Git is an **Open Source Distributed Version Control System**.

[Git](https://www.simplilearn.com/tutorials/git-tutorial/what-is-git) is a **version control system** for tracking changes in computer files and is used to help **coordinate work** among several people on a project while **tracking progress** over time. In other words, it’s a tool that facilitates **source code management** in software development.

Git favors both programmers and non-technical users by keeping **track of their project files**. It enables multiple users to work together and handles large projects efficiently.

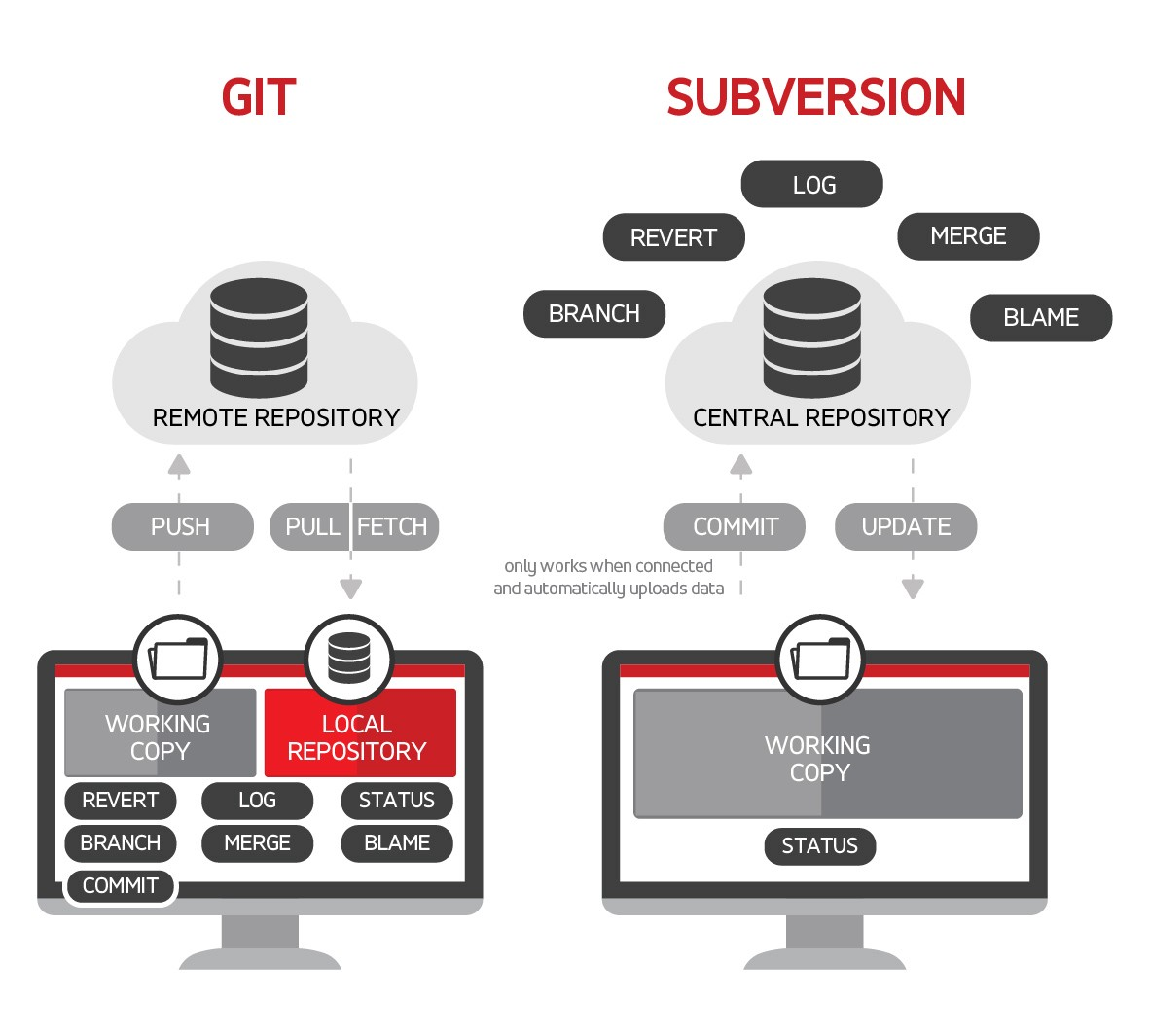
### Version Control System?

A version control system (VCS**) records/keep track of all the changes made to a file or set of data**, **so a specific version may be called later** if needed.

e.g. **Subversion** /Git (But what if server fails,therefore Git)

e.g. Created product 1.01🡪1.0.2🡪1.0.3

Also has backup of all features



**Distributed Version Control System**: Git has a **remote repository** which is stored in a **server** and a **local repository(distributed copies)** which is stored in the computer **of each developer**. This means that the **code is not just stored in a central server, but the full copy of the code is present in all the developers’ computers.** Git is a Distributed Version Control System since the code is present in every developer’s computer.

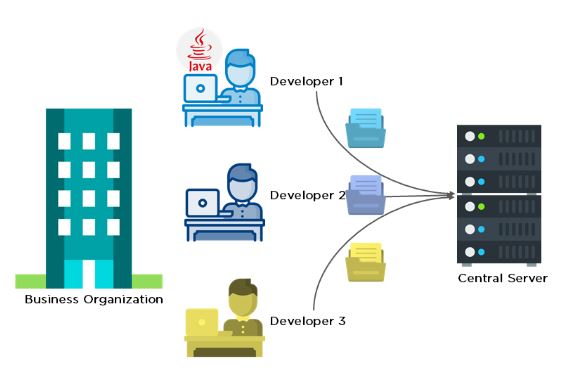
**Benefit:Even if server /remote repository fails 🡪You will have local copy**

Git is a open source distributed version control system used for tracking changes in computer files. It is generally used for source code management in software development.

* Git is used to tracking changes in the source code
* The distributed version control tool is used for **source code management**
* It allows multiple developers to work together Coding collaboration
* It keeps track of **contribution of developers** e.g. create code ,fix bugs,run test
* This helps **ensure** that all team members are working on the latest version of the file
* It supports non-linear development through its thousands of parallel branches
* Manage projects with **Repositories**
* **Clone** a project to work on a local copy
* Control and track changes with **Staging** and **Committing**
* **Branch** and **Merge** to allow for work on different parts and versions of a project
* **Pull** the latest version of the project to a local copy
* **Push** local updates to the main project



Created By :**Linus Torvald** in 2005



**Trunc Based Developemnt:**

Working on version 5.0

In local machine create branch has 5.1 🡪merge on master branch

**Git Alternatives :**

Azure Devops Server

Bazaar

AWS Code Summit

**GitHub:**

* Git is not the same as GitHub.
* GitHub makes tools that use Git.
* GitHub is **the largest host of source code** in the world, and has been owned by Microsoft since 2018

**Alternatives:**

GitBucket,Gitlab,NitBucket,Bitbucket

|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Git is a software | GitHub is a service |
| [Git can be installed](https://www.simplilearn.com/tutorials/git-tutorial/git-installation-on-windows) locally on the system | GitHub is hosted on the web  Cloud platform where we make and maintain repository |
| Provides a desktop interface called git GUI | Provides a desktop interface called GitHub Desktop. |
| It does not support user management features | Provides built-in user management |

**git version**

$ git --version

git version 2.36.1.windows.1

**Gitbash and git GUI**

Git command prompt

UI interface

**GIT CONFIG**

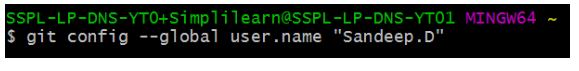
used to set Git configuration values on a global or local project level.

If you are working with other developers, you need to know who is checking the code in and out, and to make the changes.

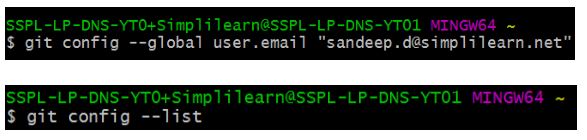
$ git config - -list #to check username/email,etc

$ git config - -global user.name #to check user name

$ git config - -global user.name “sandip”



$ git config - - global user.email [dd@rediffmail.com](mailto:dd@rediffmail.com)



$ mkdir myproject

**Git repository?**

Git repository refers to **a place where all the Git files are stored**. These files can either be stored on the **local repository or on the remote repository**.

Note:In remote .git is not present

.git is present



**Git Initialisation(Creating gir repository)**

If you want **to initialize an empty repository** to a directory in Git, you need to enter the git init command. After this command, **a hidden .git folder will appear**

The git init command adds a local Git repository to the project.

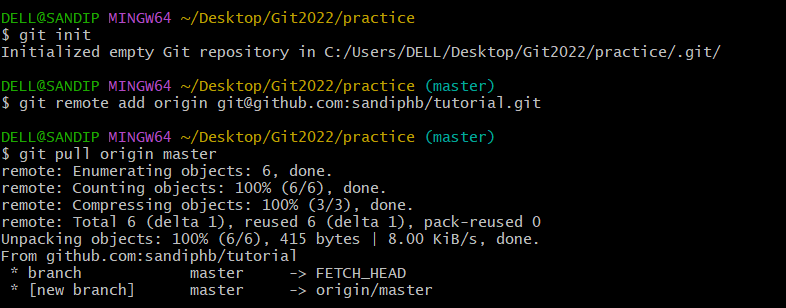
$ git init

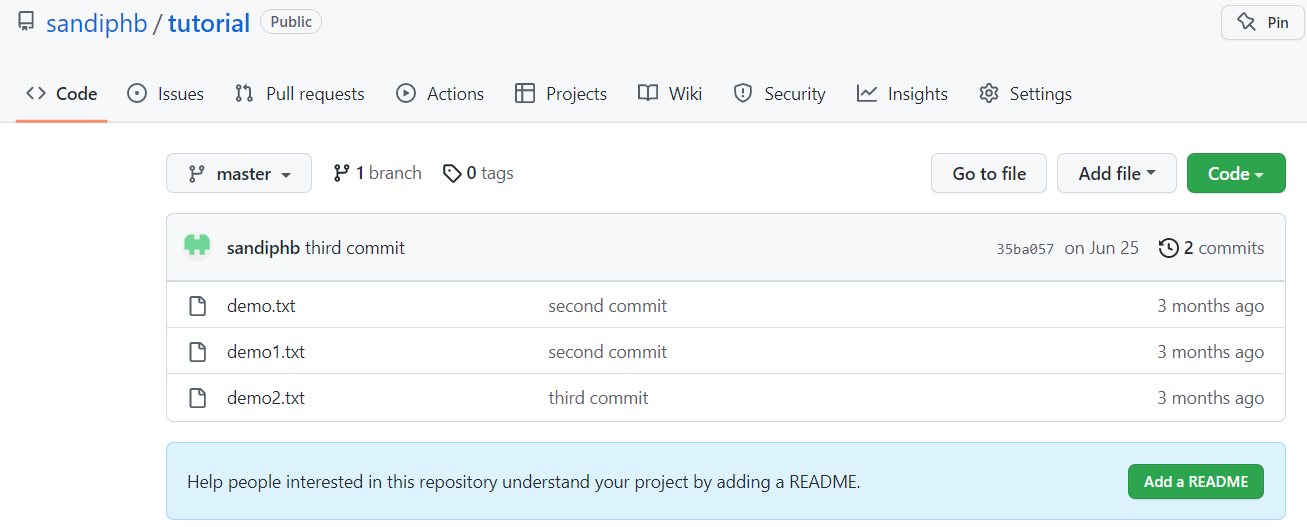
**$ git remote add origin url**

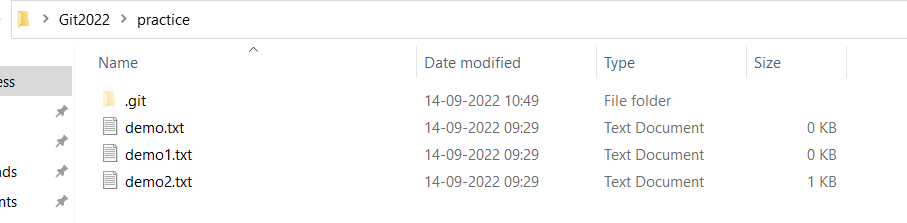
**#**make connection of local with remote

($ git remote remove origin -To remove origin)

**$ git pull origin master (master branch** **name**)i.e. all files are pulled from master branch







**Normal Flow when we Create Local Repository and Pull files**

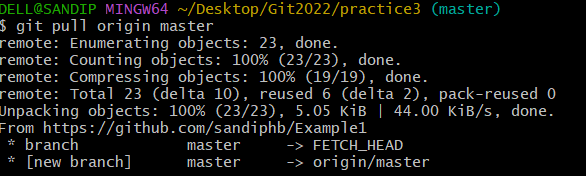
$ git init

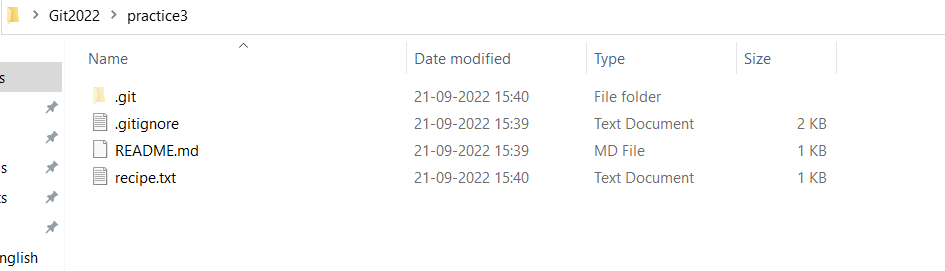
****

$ git remote add origin https://github.com/sandiphb/Example1.git

****

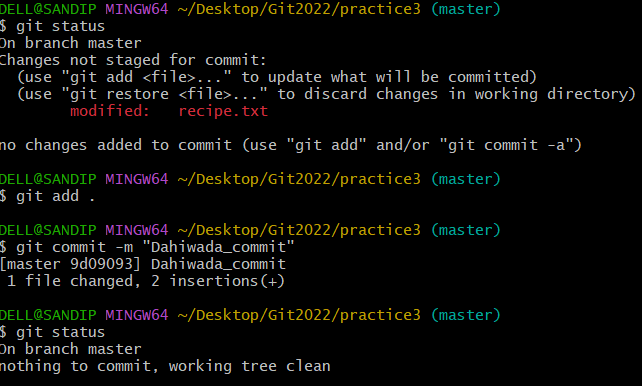
$ git pull origin master

****

****

$ start recipe.txt

**(Modified)**

****

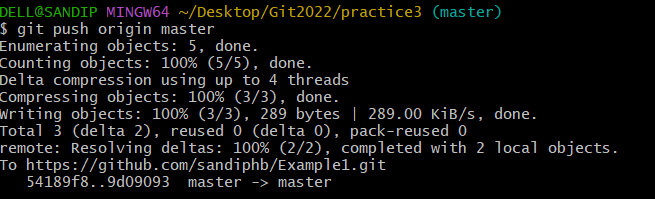
$ git add .

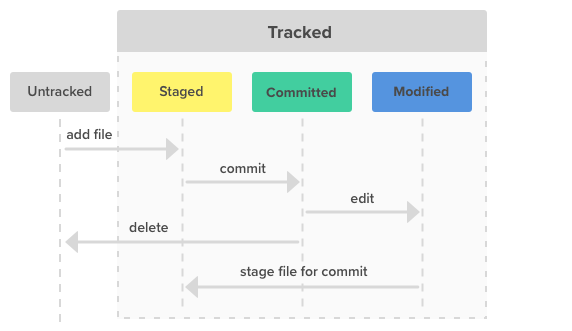
$ git commit -m "Dahiwada\_commit"

$ git push origin master

Or

git push --set-upstream origin master

****



Files in your Git repository folder can be in one of 2 states:

* **Tracked** - files that Git knows about and are added to the repository
* **Untracked** - files that are in your working directory, but not added to the repository

 When you first add files to an empty repository, they are all untracked. **To get Git to track** them, you need to stage them, or **add** them to the staging environment.

As you are working, you may be adding, editing and removing files. But whenever you hit a milestone or finish a part of the work, you should add the files to a Staging Environment.

**Staged** files are files that are ready to be **committed** to the repository you are working on

***git fetch --all***

**CLONE**

**copy of a specific repository or branch within a repository**

When you clone a repository, you don't get one file, like you may in other centralized version control systems. By cloning with Git**, you get the entire repository - all files, all branches, and all commits.**

cloning automatically creates a remote connection called "origin" pointing back to the original repository.

**Syntax:**

**$ git clone url**

$ git clone git@github.com:madhuriburande/python\_gd\_notes.git

**Pwd**

$ pwd

/c/Users/DELL/Desktop/Git2022/practice2

**To show on which directory you are currently in**

**$ ls**

One.txt Two.txt

Which files are there

**$ cd**

**Cd tensorflow/**

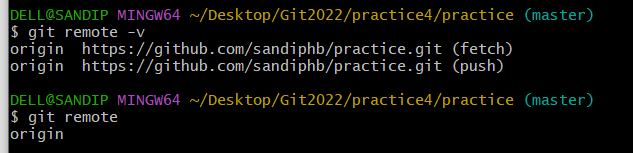
**To go on tensorflow folder**

**Shift+Insert to copy paste**

**$ git add .**

**$ git commit -a -m “message”**

**git** **remote**:



**$git push origin master**

**BRANCH in CLONE**

**$ git branch mybranch**

Modification

**$ git commit**

After that we have to push like this

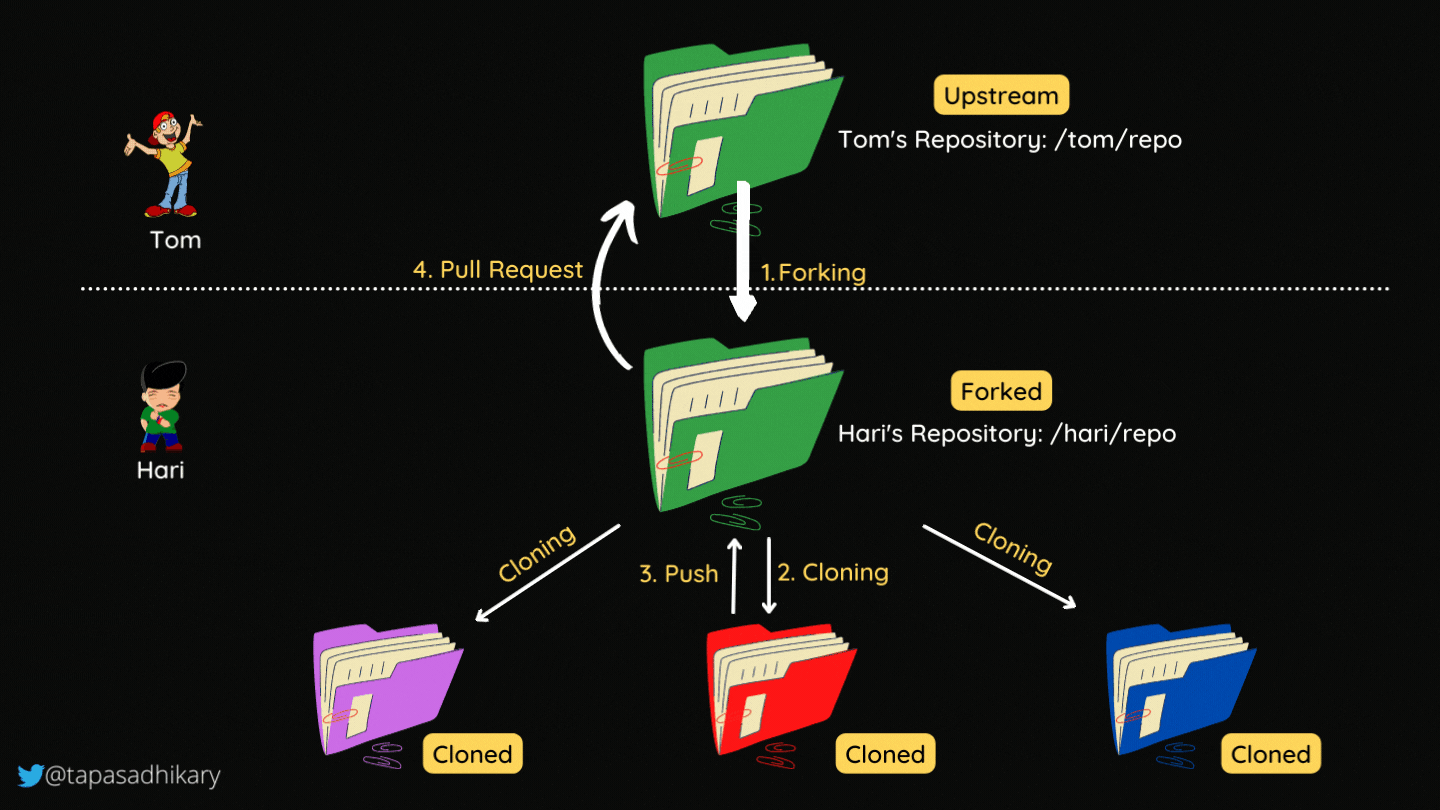
**$ git push --set-upstream origin mybranch**

Then merge in main repository

**FORK**

A fork is a copy of a repository. This is useful when you want to contribute to someone else's project or start your own project based on theirs.

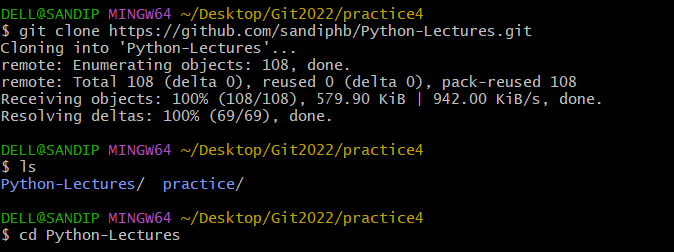
fork is not a command in Git, but something offered in GitHub and other repository hosts.



Now we have our own fork, but only on GitHub. We also want a clone on our local Git to keep working on it.

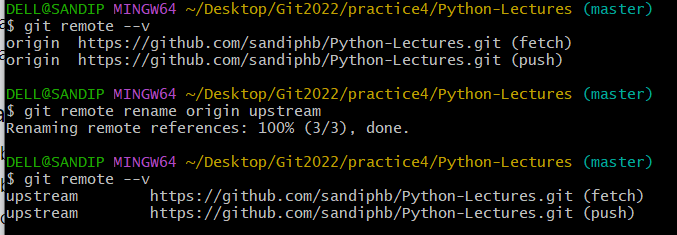
A clone is a full copy of a repository, including all logging and versions of files.

**$ git clone** [**https://github.com/sandiphb/Python-Lectures.git**](https://github.com/sandiphb/Python-Lectures.git)

****

$ ls

$ cd Python-Lectures

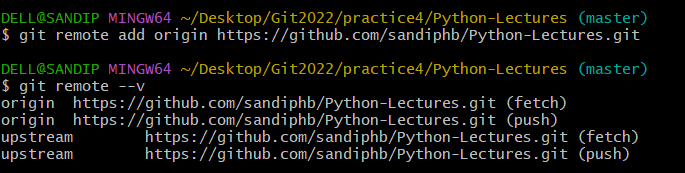
****

**$ git remote –v**

**$ git remote rename origin upstream**

**$ git remote add origin** [**https://github.com/sandiphb/Python-Lectures.git**](https://github.com/sandiphb/Python-Lectures.git)

Note:This is our forked (same as cloned)



* origin - our own fork, where we have read and write access
* upstream - the original, where we have read-only access

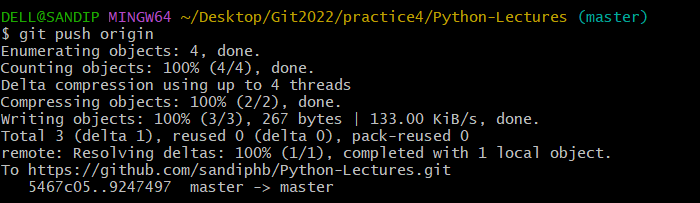
Now we can modify things here,

$ touch sandy.txt

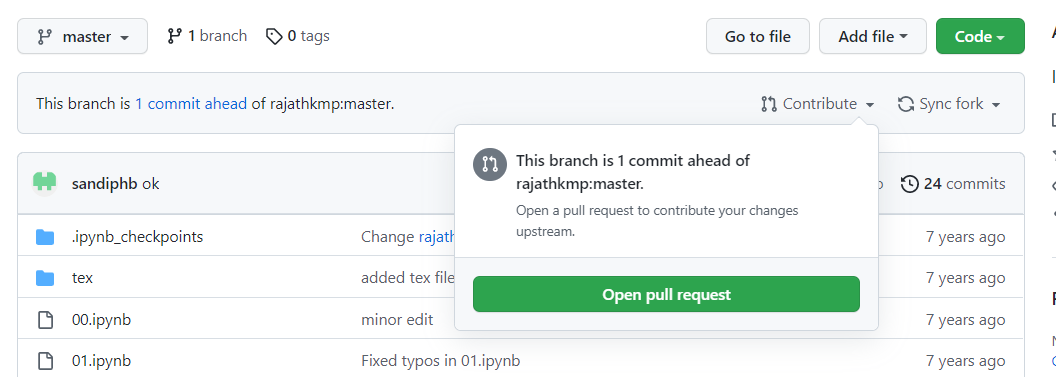
$ git add .

$ git coomit -m "ok"

$ git push origin

’

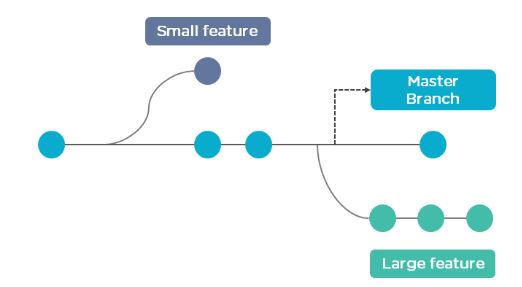
**Pull request**

****

**Branches in Git**

Branch in Git is used to keep your changes until they are ready. You can **do your work on a branch** while the **main branch (master) remains stable**. **After you are done with your work, you can merge it** with the main office.

With a new branch , edit the code directly without impacting the main branch

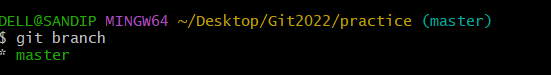


The above diagram shows there is a master branch. There are two separate branches called “small feature” and “large feature.” Once you are finished working with the two separate branches, you can merge them and create a master branch.

**$ git branch**

It shows how many branches are there

Note:master in green color



**$ git branch -a**

### List all branches (both local and remote)

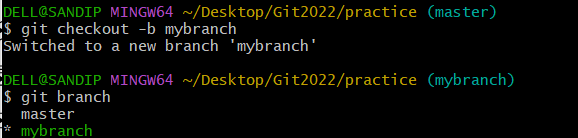
### 

### $ git branch -vv

### List local branches and their corresponding upstream remote branches

### Creating new branch First way :

### $ git checkout -b mybranch



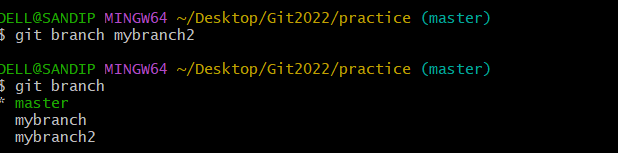
Note :Here no need to use checkout/switch to go on branch

**Second Way:**

git branch branch name

e.g.

**$ git branch mybranch2**



Here ,we need to manually checkout to switch on it

**$ git checkout mybranch2**



It is recommended **to create local branch and do operations** .

**Don’t mess with master branch** .

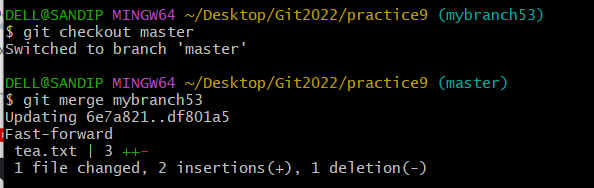
Master branch is used for **deployment** and everyone including **client sees it** .

**Merge branches local with master**

First checkout to master branch

git checkout master

git merge mybranch53



Now we can push

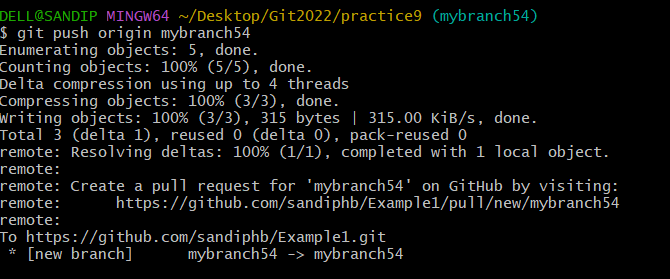
**$ git push origin master**

Now can delete mybranch

**git branch -d mybranch53**

**SECOND WAY**

Directly pushing the new branch to remote after commit

****

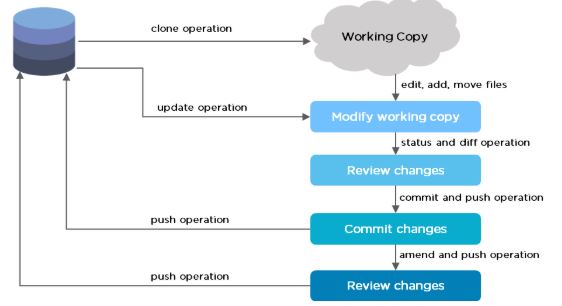
**Github Tips Khushboo mam:**

Repository-clone

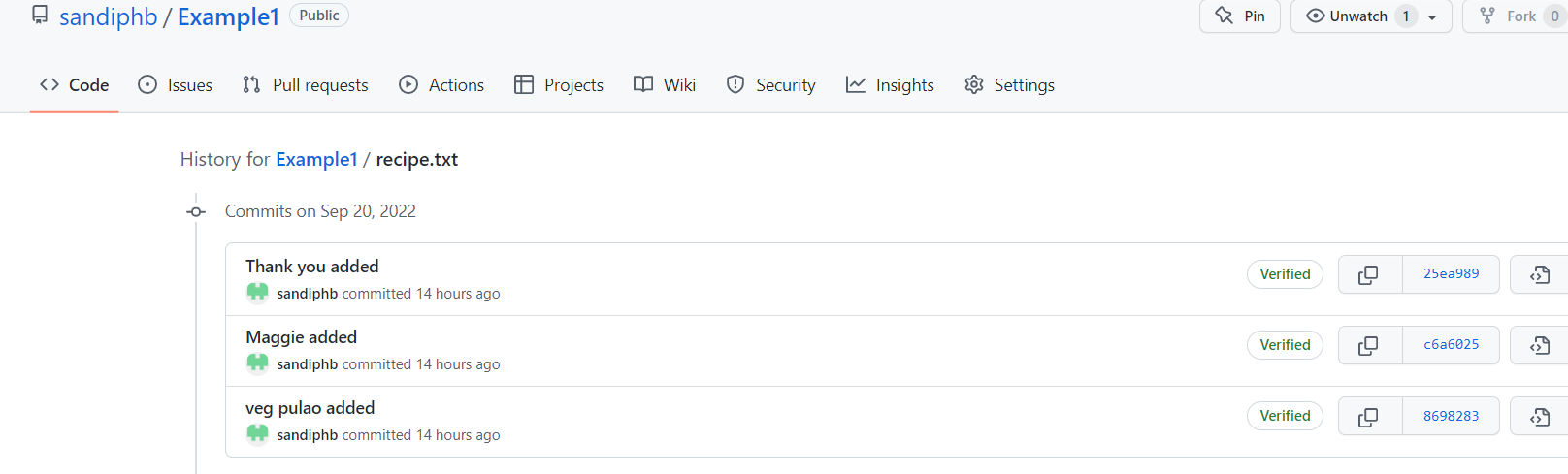
Make local copy in local machine(not related with git)—

Do modification here in local copy 🡪copy in repo

After push in final repo



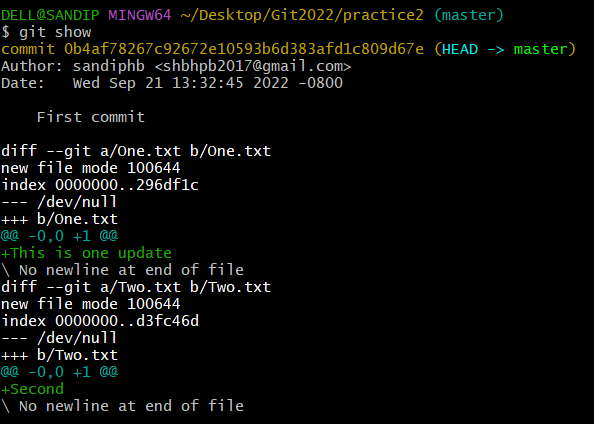
**COMMIT ADDED in GitHub:**

****

**We can go to Previous versions in GitHub**

**$ git show**

Shows log message and diff about the commit you are on.

****

**Git status Error because no repository**

Hence,create repository using git init

****

**$ git add –a**

**$ git add -all**

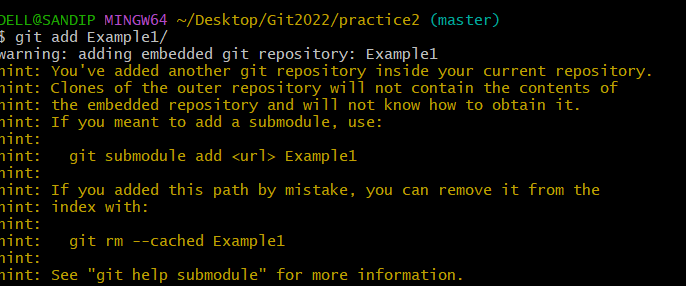
**$git add .**

**Means all files go in staging area**

**$ git add filename(particular file to add in staging area)**

**$ git add\*.py (To add all files with .py extension)**

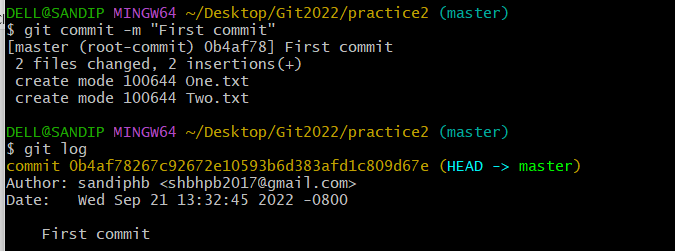
**To add cloned /new repository to current repository**

****

$ git log

**To show details of all commit**

**Note:press q to end loop in case**

****

**Start/open Filename**

**e.g.**

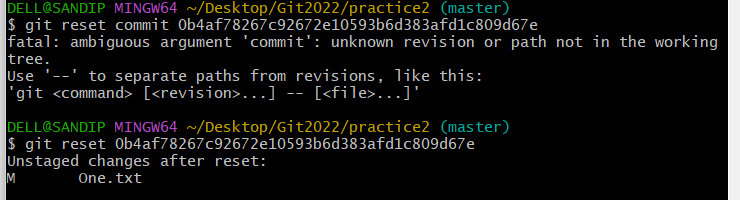
$ start One.txt

**To open file directly**

**Reset**

**$ git reset commit\_id**

**Roll back to specif commit**

****

**Delete repository**

**$ rm -rf .git**

**Note:.git repository will be deleted**

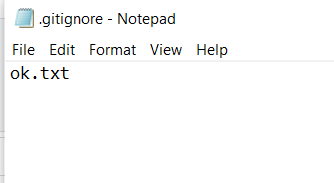
**$ touch ok.txt**

**It will create file named ok.txt**

**Git Ignore**

$ touch .gitignore

**Here ,making .gitignore text file and in it writing name of file which I want to be ignored**

****

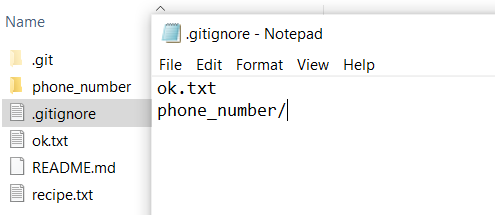
**So ok.txt will be ignored from status**

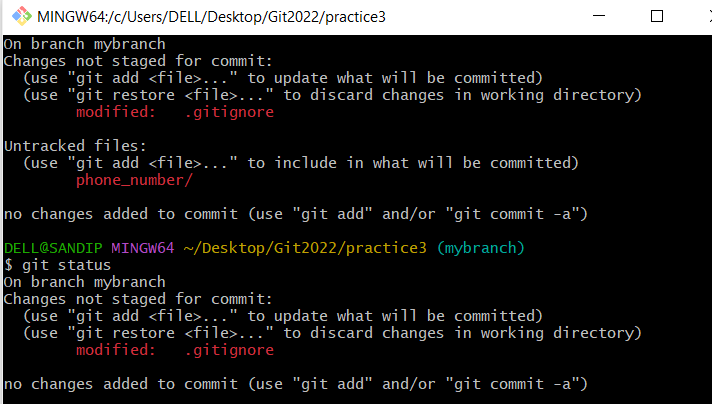
Note :If want to ignore all specific extension file then just write \*.log or \*.txt

Note:Even can ignore whole folder

Just write its name/

e.g. phone\_number/

****

****

**Note:At first it is showing folder untracked but after putting in ignore ,it is not showing**

## **Git Commit without Stage**

git commit -a -m "Updated index.html with a new line"

**$ git restore --staged tea.txt**

To restore file from staged to untrack area

**CONFLICT**

**REBASE**

**PICK AND SQUASH or RESET**

**STASH AND POP STASH**

**HTTP AND SSH**

**UPSTREAM**