**GIT/GIThub**

**Why is GIT around?**

It is a version control system. Helps in keeping track of files.

* Easily recover files
* Give info of who introduced an issue and when
* Rollback to previous working state

History of version control system:

To get over from zipping of files every time, they made-

* Local VCS(version control system: Database to keep track of files

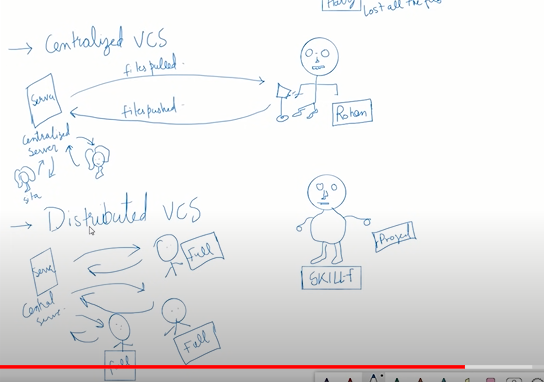
Pros: Can track files and roll back

Cons: If you loose your hard disk, everything is lost

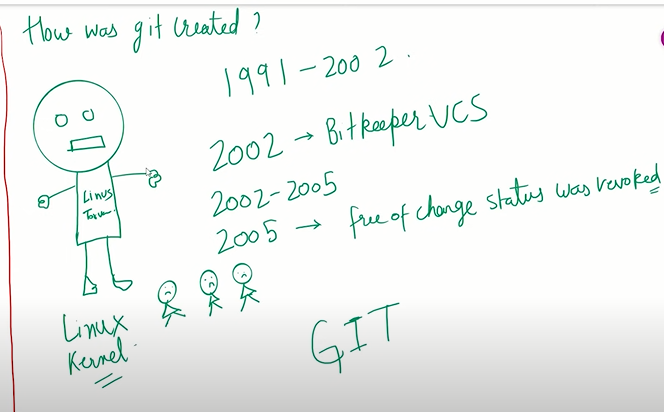
* Centralized VCS: collaborated person pulled files from server and then work and then files pushed.

What if server is damaged, then problem can happen, so now comes distributed VCS.

* Distributed VCS: every collaborated person has own complete backup of the project.



We do mirror of full repository in distributed VCS.



**Git and GitHub:**

GIT is a software. It is free.

GITHUB is a hosting service. It is a website which hosts a GIT repository. It takes money for maintenance. BITBUCKET, GITHUB, GITLAB etc. is a hosting service which hosts the GIT repository. Also we can host GIT repo in our own server.

Characteristics of GIT:

* Captures snapshot, not differences.

When you make GIT repository, a “.git” hidden folder will generate. Inside this everything, like full project history will be there.

Also you can fetch particular version like latest version from files like f1,f2,f3.

Also you can ask for files as eg. which we changed three years ago, what git will do is

, will remove files from working directory , and f1.f2,f3 which were present three years ago, that will be given to us.

Also you can get back old folder using commands.

* Almost every operation is local in GIT. That means all operations will be done is local computer.

No need to push in all server , work in local computer, After that you can push in remote GIT Repo or centralized repository/server like GITHUB, BITBUCKET etc.

So you can work in local computer, after that when you feel its completed or you can push, you can push to centralized server/centralized server.

* GIT has integrity:

Its not like someone will change our project and it will reflect in my project. What GIT do is

For all changes, GIT find sha-1 checksum.

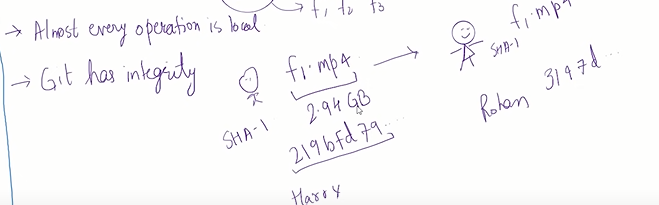
Checksum : suppose we have f1.mp4 file (294 gb as eg. ), and we are downloading it.

For file checksum will be like string “219bFd79”. For all file checksum will be unique.

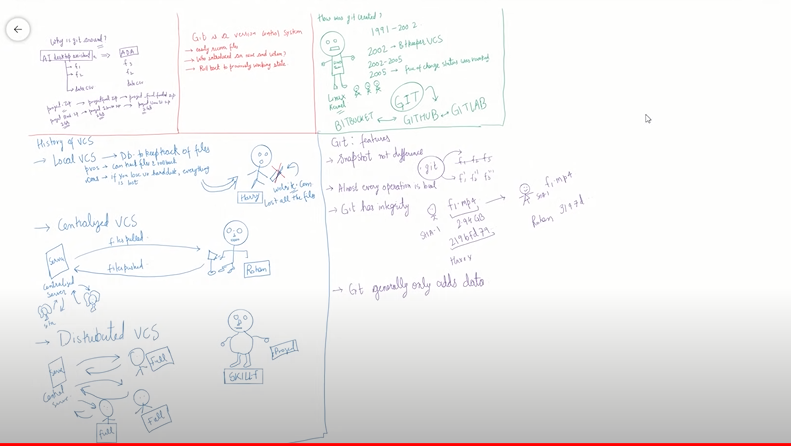
If we change small thing in file then checksum will be different.

We will find checksum, if checksum will be same means, no one touched that file and no edit happened.

What GIT does is, it internally check the checksum and takecare the .GIT folder no not get changed by anyone else.



* GIT generally only adds the data.



GIT Installation:

* GIT command line tool
* GIT bash (terminal program)

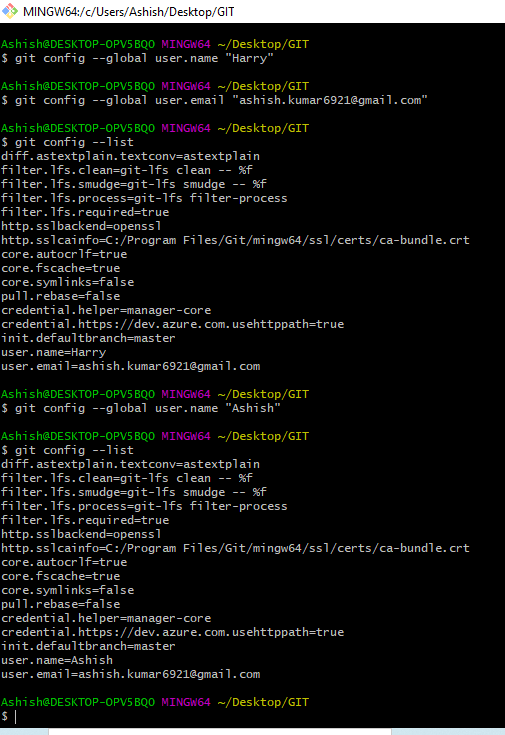
Command:

* Git
* Pwd
* Cd desktop
* Cd /c
* git status

configure username and email in GIT, whenever new GIT installed”

open git bash in directory where you are working and things stored

* $ git config --global user.name “Ashish”
* $ git config --global user.email "ashish.kumar6921@gmail.com"



Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$ git config user.name

Ashish

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

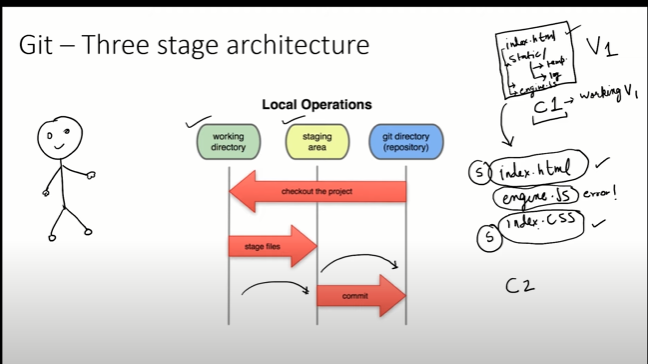
$ git config user.email

ashish.kumar6921@gmail.com

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$

Three stage architecture of GIT:



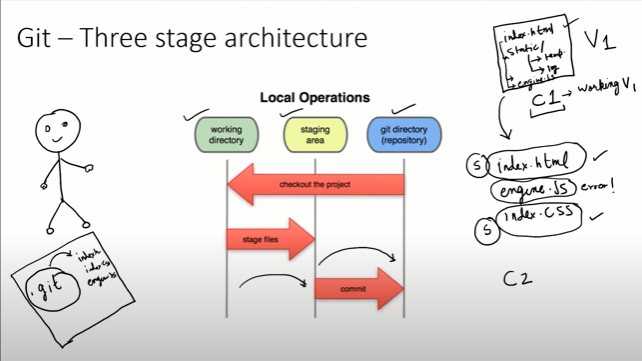
S: staging: staging area: that means index.html and index.css file is correct so we want to only these two to take it to next commit, but since engine.js is giving error second time, we want to keep same previous engine file which was working in next commit.

Staging area is nothing but, the file which we want to take to to next commit.

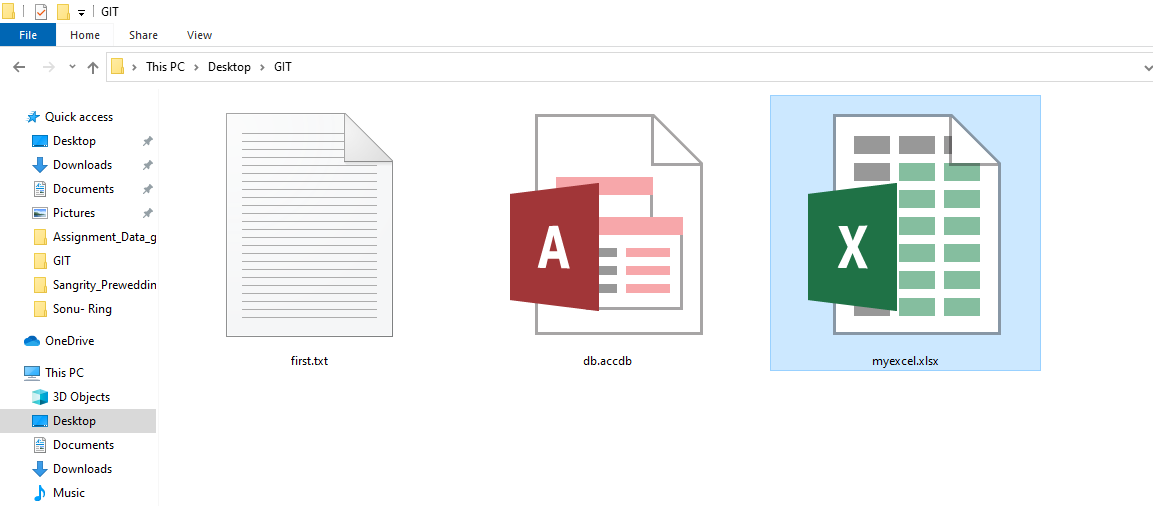
Working directory: the file types we use to see in windows explorer, that is working directory.

GIT repository(GIT directory): a directory/ hidden folder named as “.git” .

Inside .git folder all compressed folder c1, c2 etc will be there. All changed one will be there.



Made three files in our self made git folder:



Right click in folder-> open git bash

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$ git status

fatal: not a git repository (or any of the parent directories): .git

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$

From got status command we saw if it’s a git repo or not, here its nota git repo. If it would be git repo, then we don’t want any mistake, then we shall see status from “git status”

Now we want to make git repository to this folder, since we are already inside that folder, we will run:

$ git init

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$ git init

Initialized empty Git repository in C:/Users/Ashish/Desktop/GIT/.git/

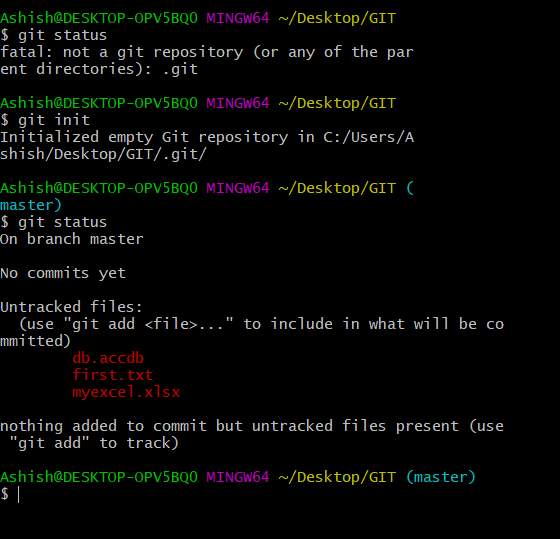
Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$

This output means, this folder is now git repo where all three files we made. Also one .git folder automatically made inside folder.

Now run git status:

Then it will give result

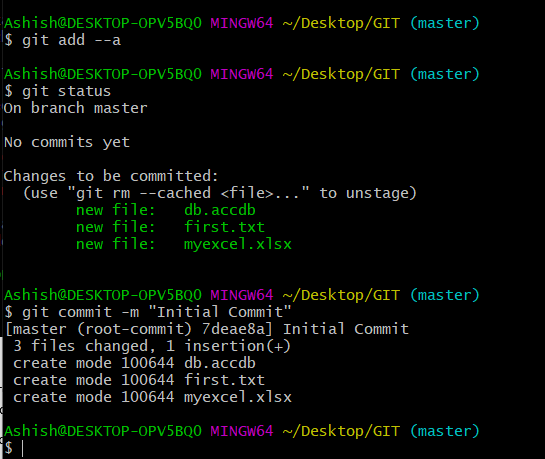


Now they are asking in output that no commits happens. Also its mentioned untracked files db,first and excel. So need to track

Now want to add all files at once:

$ git add --a ->adding all files in staging area

Since earlier files were not tracked, so added all files to track, now it comes to staging area, now needs to commit, so it will go to commit and snapshot happens .



Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

If you want to know which all you committed then:

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git log

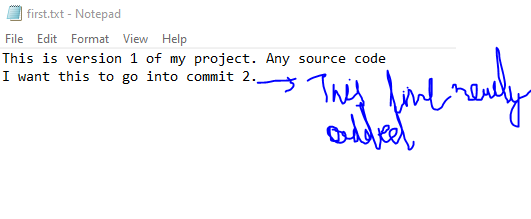
commit 7deae8a0591e332745de15fa6642dcfe6486d563 (HEAD -> master) --🡪hash code

Author: Ashish <ashish.kumar6921@gmail.com>

Date: Fri Jan 14 14:19:43 2022 +0530

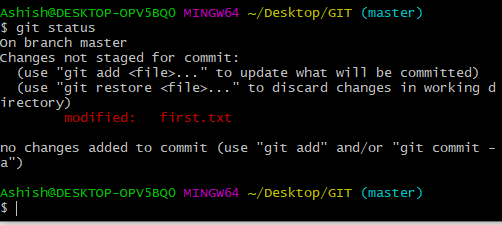
Initial Commit

Now suppose we changed something in first.txt as:

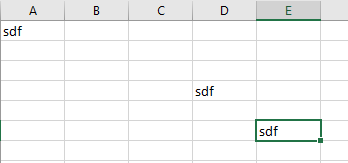


Now if we write:

$ git status ->then it will track, see below:



So okay, we modified frst.txt, we make good something by editing first.txt , with this giving some data in myexcel.xlsx file. Gave some random data and saved file and closed;



Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: first.txt

modified: myexcel.xlsx

Untracked files:

(use "git add <file>..." to include in what will be committed)

~$myexcel.xlsx

no changes added to commit (use "git add" and/or "git commit -a")

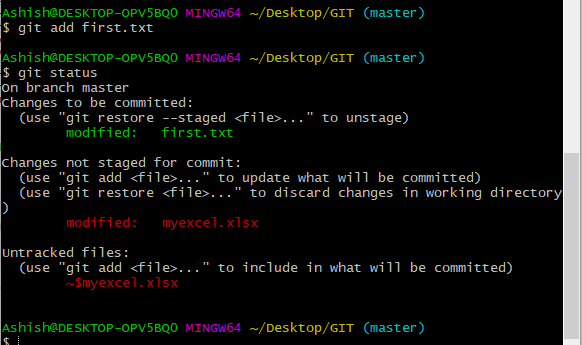
Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$

Now suppose we want to add first.txt in staging area, we are having hesitation for myexcel.xlsx file or I am getting error in this or some ore work needed in this, so only need to add first.txt in staging:

So run:

$ git add first.txt



Here we did not staged the change, that means our change is as prev, so needs to commit:

Commit ka matlab hamne kuchh change kiya hai:

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "Changed first.txt and added better design"

[master 3785566] Changed first.txt and added better design

1 file changed, 2 insertions(+), 1 deletion(-)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: myexcel.xlsx

Untracked files:

(use "git add <file>..." to include in what will be committed)

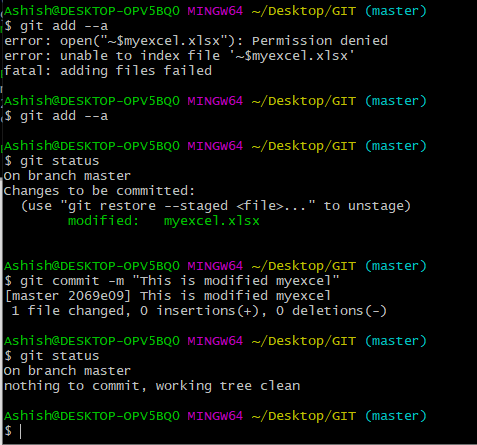
~$myexcel.xlsx

no changes added to commit (use "git add" and/or "git commit -a")

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$

If we want to add and commit all files:



Using git log, we can see all commit:

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git log

commit 2069e093e839572c5c355d3b72f2be1cb8b17b30 (HEAD -> master)

Author: Ashish <ashish.kumar6921@gmail.com>

Date: Fri Jan 14 14:59:16 2022 +0530

This is modified myexcel

commit 3785566fa8c978dba68186973d2b049951502324

Author: Ashish <ashish.kumar6921@gmail.com>

Date: Fri Jan 14 14:49:59 2022 +0530

Changed first.txt and added better design

commit 7deae8a0591e332745de15fa6642dcfe6486d563

Author: Ashish <ashish.kumar6921@gmail.com>

Date: Fri Jan 14 14:19:43 2022 +0530

Initial Commit

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$

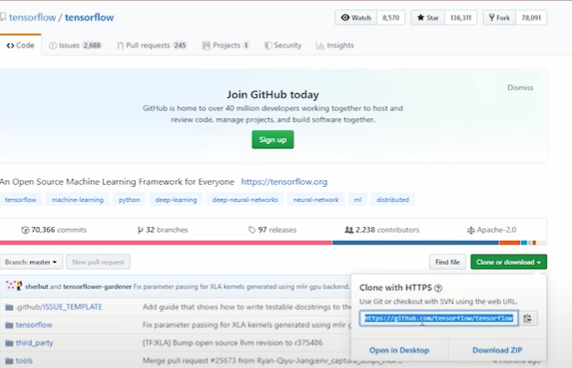
And hash code/commit code mentioned for all change in output, if we want to go back to previous one , we can go using this code, will see later how.

**Cloning a Remote Git Repository from GitHub**

Opening any Github api/website/repository, suppose we searching for Git hub website tensorflow in google:

We will clone git repo of tensorflow here.

Click on clone or download and copy link:

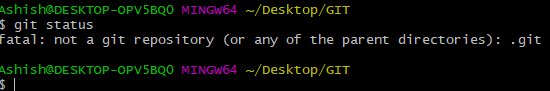


Note: if we want to stop track of any folder in git and you want that all inside that git folder would get deleted, run command inside that folder for which .git folder you want to delete,

rm –rf .git

Files will not get deleted, .git folder which is hidden will get deleted, so that you will not be able to track, like here we are inside GIT folder in desktop and three file are there and .git hidden folder inside this is there, so .git folder will get deleted (i.e git repo will get deleted and you will now not be able to track this in git.

Will run this, since we want to end this git repo.



Now all file in folder is not a git repo, i.e no tracking of the files.

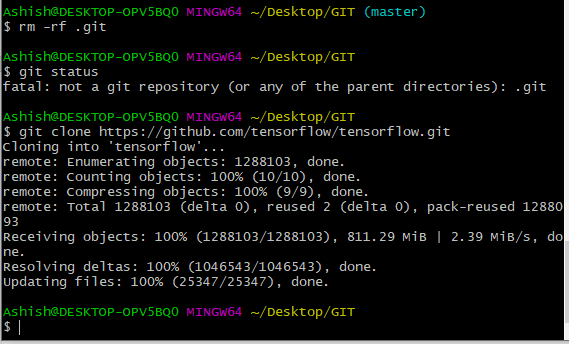
---

Now we will clone,

$ git clone (paste here the link copied from hithub website) , then enter

Tensorflow name ki directory ban jaegi, agar other name se rakhna hai directory ko to command k bad space + directory name as:

$ git clone (paste here the link copied from hithub website) Tensorflow\_updated



Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$ ls

db.accdb first.txt myexcel.xlsx tensorflow/

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

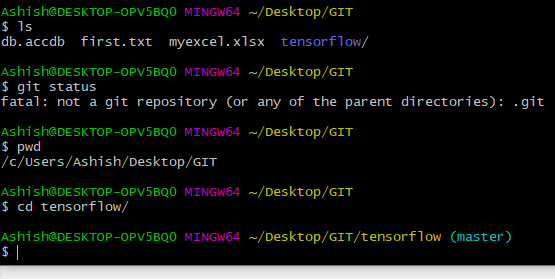
Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$ git status

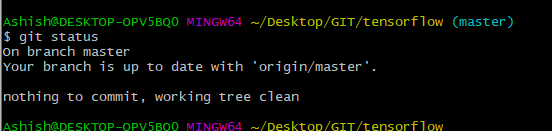
fatal: not a git repository (or any of the parent directories): .git

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

Since we are not in tensorflow directory so fatal error above

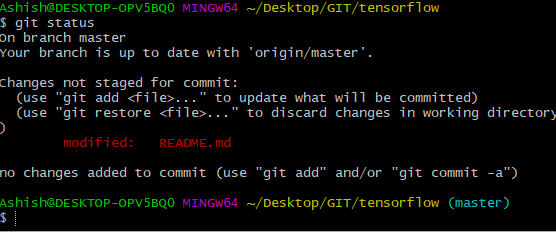


Now git status

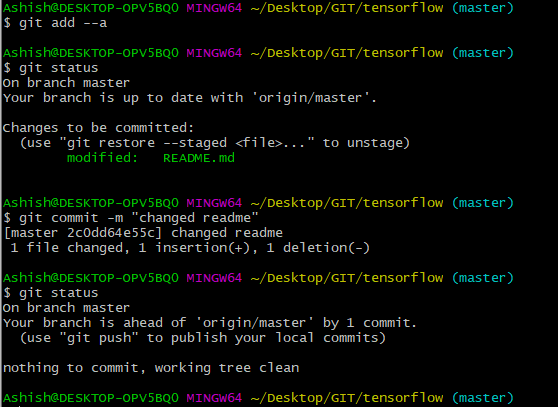


Suppose we are changing some file of tensorflow eg.. readme.md and inside this for eg.. removing l from tensorflow and save and close.

Now git status:



Adding all in staging, then commit.



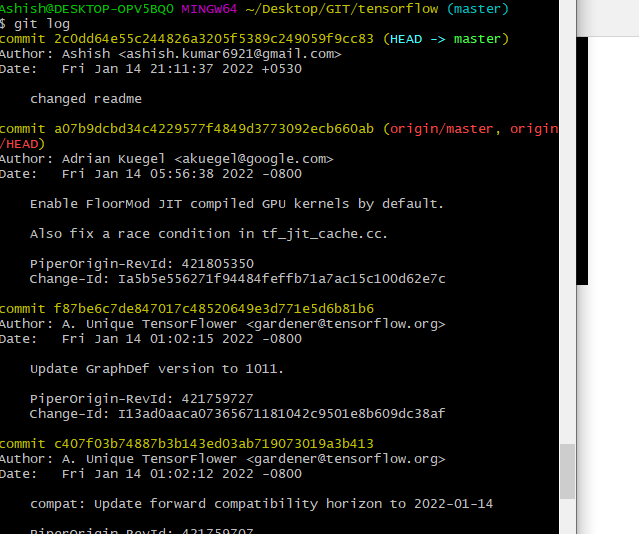
Our tensorflow is going 1commit ahead as compare guthub online tensorflow.

Obvisously right now we cant merge our tensirflow in online github, will lear later.

Whatever we copied that is our property , but not in online github.

We can see when and who changes the files using git log as:

$ git log

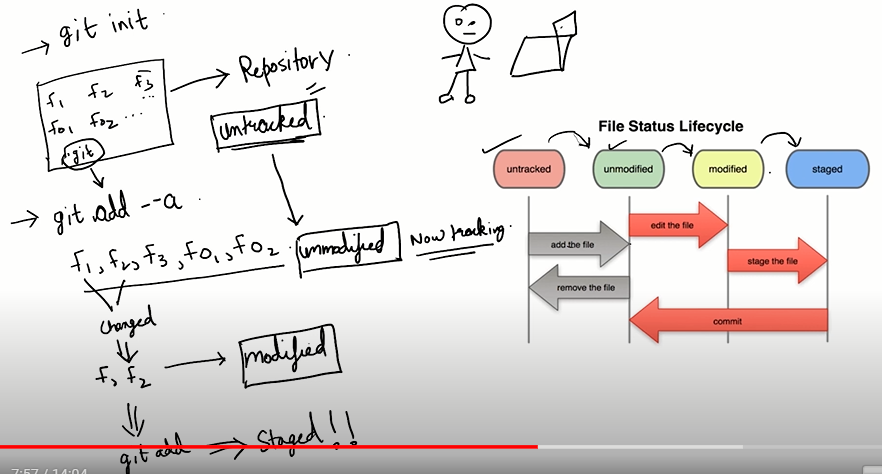


$ q-> for quit

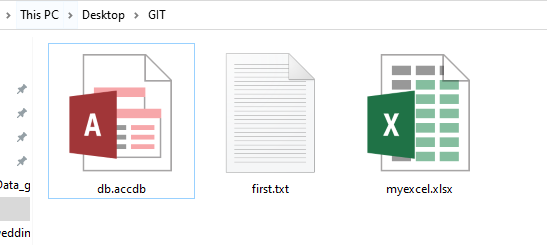
We can rollback to prev change , will see later.

Next video-6

**FILE STATUS LIFE CYCLE:**



Tensorflow folder deleted, made fresh folder .



Opening git bash inside folder,

$ git status

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$ git status

fatal: not a git repository (or any of the parent directories): .git

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

$ git init

Initialized empty Git repository in C:/Users/Ashish/Desktop/GIT/.git/

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

db.accdb

first.txt

myexcel.xlsx

nothing added to commit but untracked files present (use "git add" to track)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

We made git repo, but till now no tracking of files, so $ git add --a

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git add --a

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

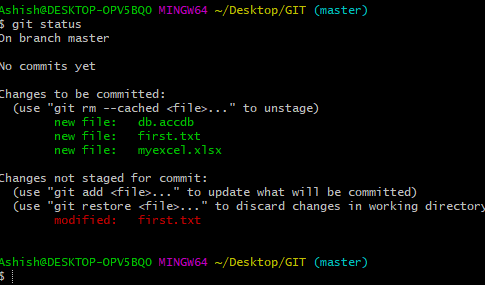
new file: db.accdb

new file: first.txt

new file: myexcel.xlsx

above tells changes to be committed, we have staged the changes. Here we started tracking first time, so we tell we took files to unmodified from untracked. Means it came from untracked to staging area.

Lets change the first.txt , eg wrote something there,save close then in bash,



Here we can see, our new file first.txt is in staging area and in modified as well. It came in both place because we kept first.txt earlier in staging area.

We put that in staging area then we did change. When we put anything in staging that means it ready to commit.

After staging we modified, then also what was about to go in commit that will go. Either we modified after staging, git will ignore.

If we run here :

$ git add first.txt

Now if we run git status, then new first.txt will get stage then if we commit, three will get commited.

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git add first.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: db.accdb

new file: first.txt

new file: myexcel.xlsx

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

.gitignore: Ignoring Files in Gi

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: db.accdb

new file: first.txt

new file: myexcel.xlsx

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "git tutorial 7"

[master (root-commit) 5e49c0c] git tutorial 7

3 files changed, 3 insertions(+)

create mode 100644 db.accdb

create mode 100644 first.txt

create mode 100644 myexcel.xlsx

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$

**Lets see how to ignore files:**

Lets assume inside files, our software is kept ready, and this generates logs files

Commands to generate blank files in linux:

$ touch error.log

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ touch error.log

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ ls

db.accdb error.log first.txt myexcel.xlsx

open error.log files write something and save.(consider software generting the content inside file), so suppose we want to ignore the error log file. Means whenever log file getting updated, we want no effects to git repo.

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

error.log

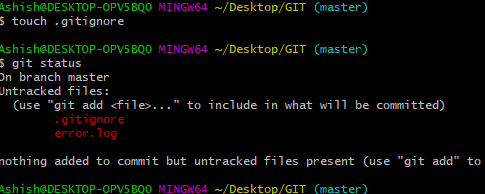
nothing added to commit but untracked files present (use "git add" to track)

So, we do not want to see the file here, showing error.log as status, we want working tree clean, so

Make one file:

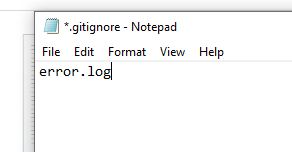
$ touch .gitignore

$git status

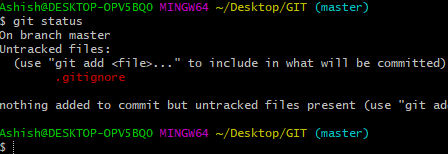


Open and write content but what to write?

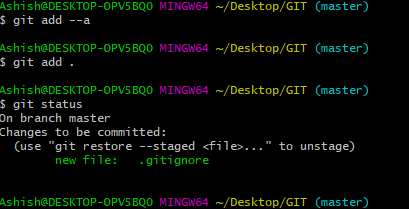
so since we want error.log file to ignore, hence write inside .gitignore file -> error.log, save and close



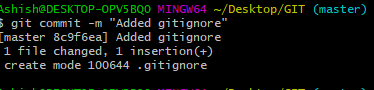
Now if we will run git status, and since we added error.log inside .gitignore then error .log file not showing see:



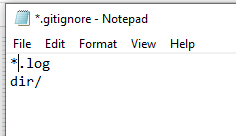
Now adding files,

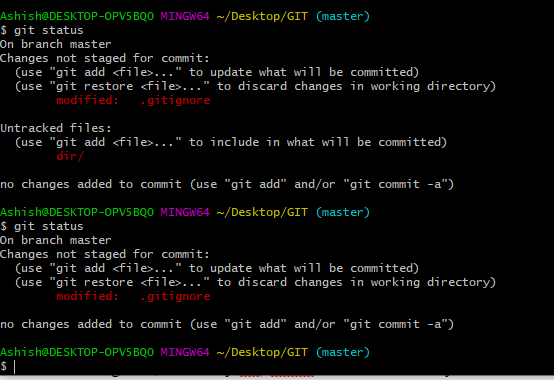
.gitignore came in staging area,

Now commit



Now if we will change the error .log file, then also it will not show for modified thing during git operation , since we kept error.log inside .gitignore.

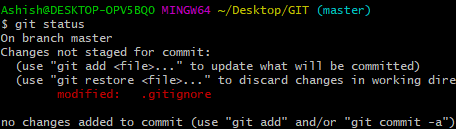
* If many .log file gets generated and we want to igmnore, then simply edit .ignore file with \*.log , save close. Then it will work.
* If any directory/folder is there, and we do not want to track/ we want to ignore then come to .ignore file and write dir/. If inside any folder also dir folder is there. That will ignore. Eg inside xyz folder there is dir named directory then also.
* 



If only outer dir folder need to ignore, but if any dir/ fodler is there inside any xyz folder then no ignore then, write inside .gitignore

/dir/, save close.

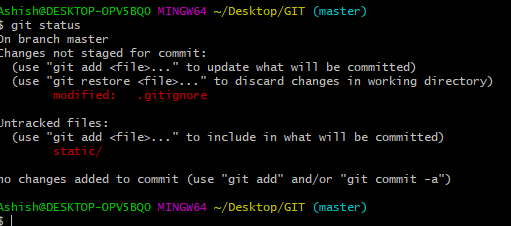
Eg… making one static folder, isode this ,making another folder named dir. And inside dir making one txt named “this is a doc”



Here inside static folder, the dir folder is ignored. Because git automatic ignores blank folder also we have mentioned dir/ in .gitignore to not track.

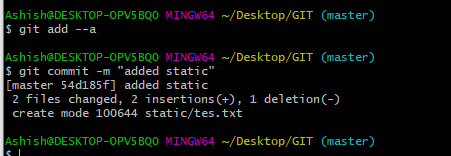
Suppose inside static, making another txt file where dir folder is already there,

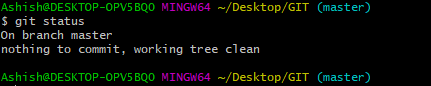
Now it will track,



Now if, we do

$ git add-- a and commit as:





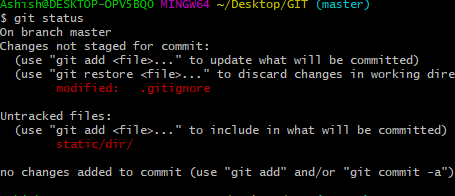
Suppose inside static->dir->txt file(hete we changed something), no affects on git status:

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

* dir/ in .gitignore means wherever dir folder found, git will ignore
* and if we want to ignore only outer dir then write /dir/ inside .gitignore .now it will start showing inside dir as untracked as:
* 

If we want to ignore full path, mention like static/dir in .gitignore

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

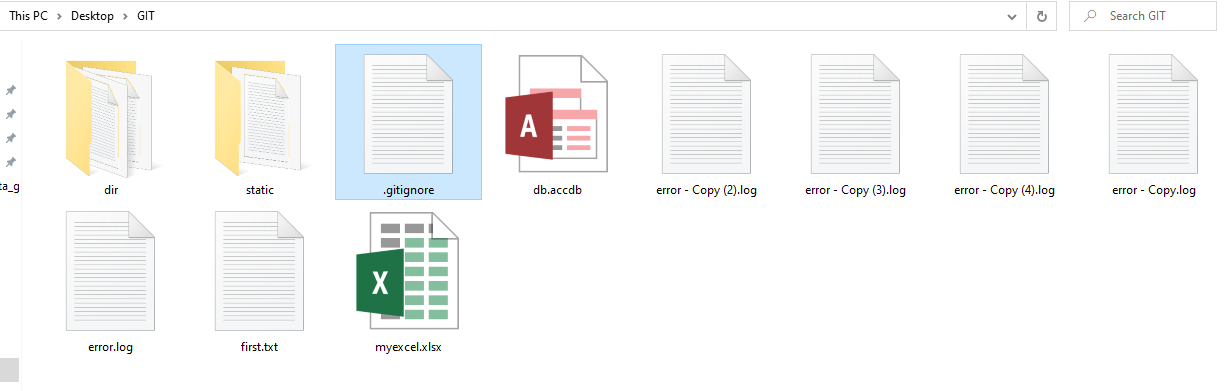
On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

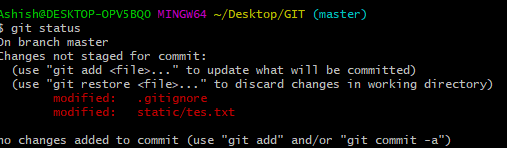
(use "git restore <file>..." to discard changes in working directory)

modified: .gitignore

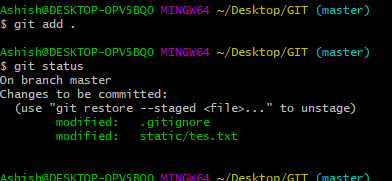


There are more patterns in gitignore concept.

# Next video-8: Showing Changes Between Commits/Staging Area & Working Directory

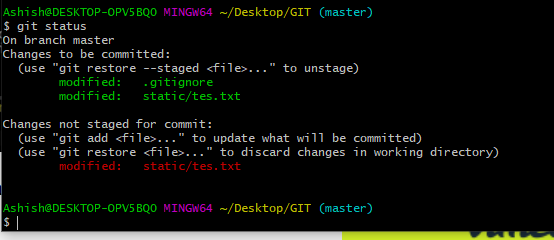


Showing as modified, now adding to staging area:



Above showing its in staging area.

Modifying tes.txt inside static, eg wrote something:o/p

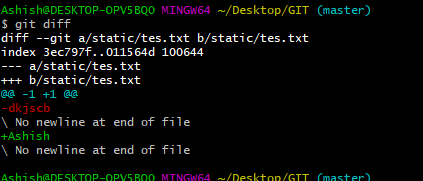


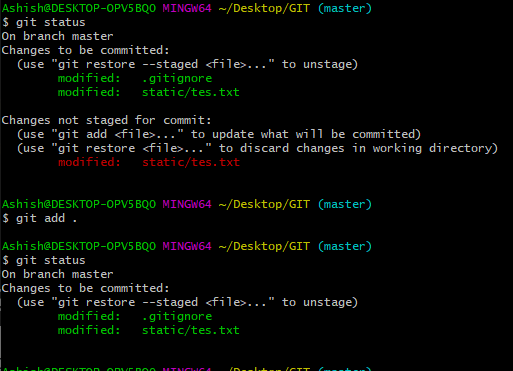
Here above picture showing it came in staging area and in modified as well.

If we will do **git add .** then modified will go to staging area, merged.

But if we do commit here, then what was in staging will get commit, not modified once.

$ git diff-> it will compare working dir with staging area. i.e compare between above red modified and green modified.





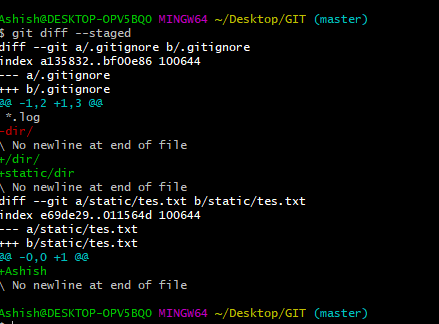
If all in staging area, git diff will not show anything.

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git diff

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**It would be better, if we can compare last commit to staging area as:**



# Skipping The Staging Area | Git Tutorials

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: .gitignore

modified: static/tes.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git add .

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "This is a good commit"

[master 58a405a] This is a good commit

2 files changed, 3 insertions(+), 1 deletion(-)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Now changing in first.txt.**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: first.txt

no changes added to commit (use "git add" and/or "git commit -a")

**one file created in GIT folder, second.txt**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: first.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

second.txt

no changes added to commit (use "git add" and/or "git commit -a")

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**So doing commit without staging area:**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -a -m "Direct commit"

[master 9a99200] Direct commit

1 file changed, 1 insertion(+)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Untracked files:

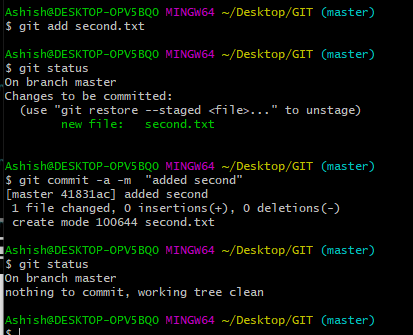
(use "git add <file>..." to include in what will be committed)

second.txt

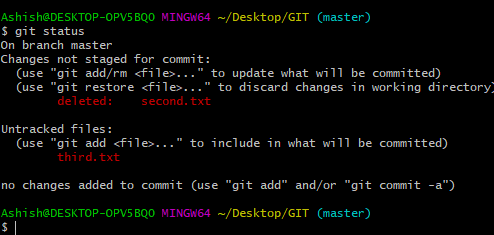
nothing added to commit but untracked files present (use "git add" to track)

**Here you can see , second.txt is untracked still and dint went to staging, that means, all tracked file will go to commit by skipping the staging, but untracked file will not go:**

**Add specifically untracked file to staging area.**

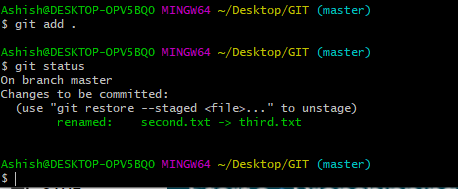


**Suppose if we renaming second.txt to third.txt manually and save, then execute command :**



**Git thinking we deled second and added one new third.txt which is untrccked.**

**Now if we execute add command , then git will know we renamed as:**



**Doing commit.**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "rename second to third"

[master 178cd07] rename second to third

1 file changed, 0 insertions(+), 0 deletions(-)

rename second.txt => third.txt (100%)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Now how to delete file using git command,**

**Suppose from folder manually deleted third.txt>**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add/rm <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

deleted: third.txt

no changes added to commit (use "git add" and/or "git commit -a")

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Now if we do commit then it will go to commit.**

**But not deleting like this, doing ctrl+z to undo third.txt**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git rm third.txt

rm 'third.txt'

**third.txt has been removed now.**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

deleted: third.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**It has done staged as well. But manual deleting domnt do staged by self.**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "remove bad file third.txt"

[master 163adf8] remove bad file third.txt

1 file changed, 0 insertions(+), 0 deletions(-)

delete mode 100644 third.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Renaming file using git command.**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git mv first.txt first\_renamed.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

renamed: first.txt -> first\_renamed.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Above Staged as well.**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "renamed"

[master 58fc4ee] renamed

1 file changed, 0 insertions(+), 0 deletions(-)

rename first.txt => first\_renamed.txt (100%)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

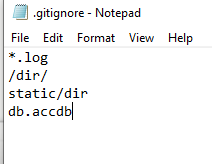
$ git status

On branch master

nothing to commit, working tree clean

**suppose now we want to ignore db.accdb file**

**so editing in .gitignore file**



Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "changed gitignore"

[master 017ab21] changed gitignore

1 file changed, 2 insertions(+), 1 deletion(-)

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Now if we want to change something in db.accdb**

**Wrote something in that:**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: db.accdb

no changes added to commit (use "git add" and/or "git commit -a")

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**We kept in .gitignore , then also its showing as modified, why?**

**Because when we changed db.accdb, that time it was already in tracked zone, so when we kept it in .gitignore then also it didn’t get ignored, so have to untrack explicitly:**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git rm --cached db.accdb

rm 'db.accdb'

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

deleted: db.accdb

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Now its showing deleted. Not deleted frm folder, but got untracked.**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "remove db.accdb"

[master de3d0e0] remove db.accdb

1 file changed, 0 insertions(+), 0 deletions(-)

delete mode 100644 db.accdb

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

**Again if we change in db.accdb then:**

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

**we removed from tracked, still its not showing as untracked since we have kept it in .gitignore**

**removing db.accdb from gitignore, now see:**

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: .gitignore

Untracked files:

(use "git add <file>..." to include in what will be committed)

db.accdb

no changes added to commit (use "git add" and/or "git commit -a")

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

# Git Log: Viewing & Changing Commits In Git

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git add .

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git commit -m "for tutorial"

[master dbe8a09] for tutorial

2 files changed, 1 deletion(-)

create mode 100644 db.accdb

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ rm -rf .git

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT

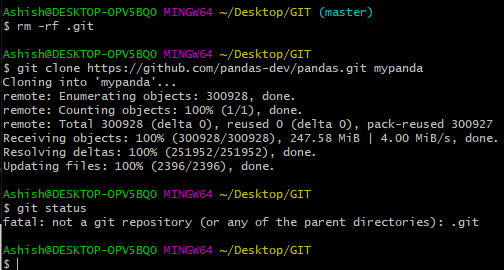
$

# rm –rf .git -> removed rom git repo.

**Now, got to github pandas, taking random repo . copy url that is clone, ctrl c on url,**

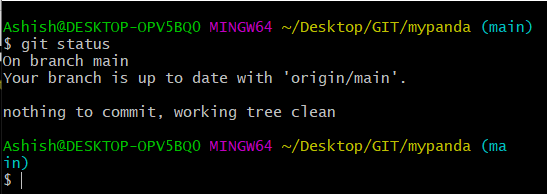
**$ git clone** [**https://github.com/pandas-dev/pandas.git**](https://github.com/pandas-dev/pandas.git) **mypanda**

**all git repo of panda github will come under mypanda folder, which is already a git repo.**



**It si telling its not a git repo, since its all inside mypanda and that is git repo,**

**So $ cd mypanda**



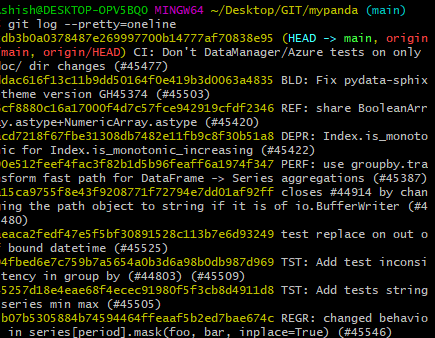
**Logs who changed : to view commits with help of git log**

**$ git log –p**

**$ git log –p -3 ->will show 3 commit along with diff(change)**

**$ git log - -stat -> chnge /commit in short**

**$git log –pretty=oneline -> will show all commits in oneline**



**$ git log - -pretty= short**

**$ git log - - pretty=full**

Use filter in git log on basis of time:

**$ git log --since=2.days**

**$ git log --since=2.weeks**

**$ git log --since=2.months and so on.**

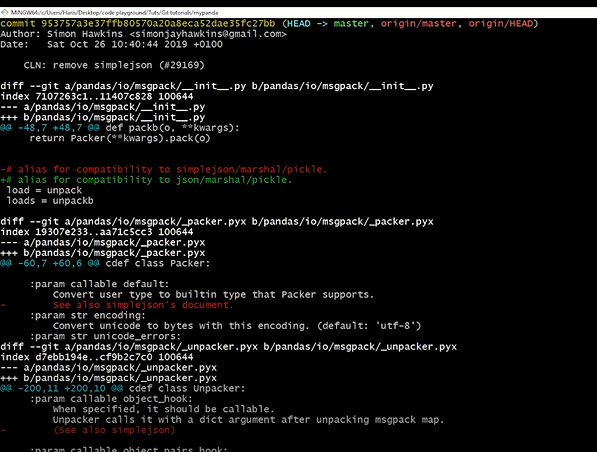
**$ git log –pretty=format:”%h --%an” -> will show the logs in a format “abbreviated commit hash – author name”**



**$ git log –pretty=format:”%h --%ae” -> will show the logs in a format “abbreviated commit hash – author email”**

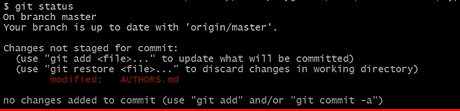
**🡺goto Git scm documentation in google for logs format and all.**

**$ git log –p -1**



**Will go to one file and edit,**



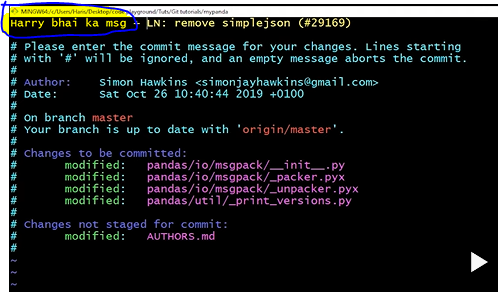


**$ git add .**

Now we want to merge all commits/changes which we did modification in authors.md to the commits which simon did(see output ss above) and message as well

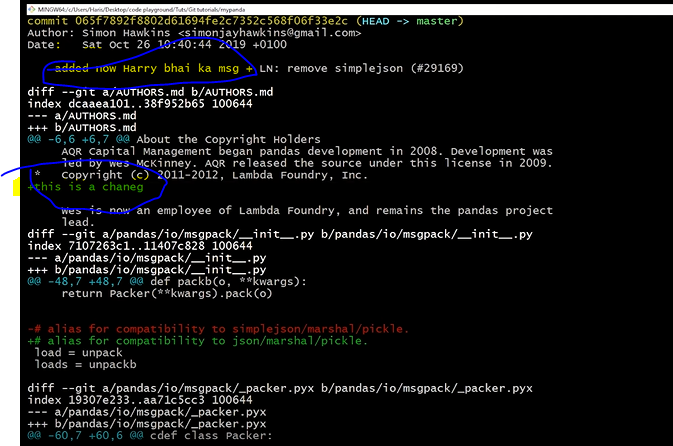
$ git commit –amend ->amend means change

Then editor will get open, will add something as:



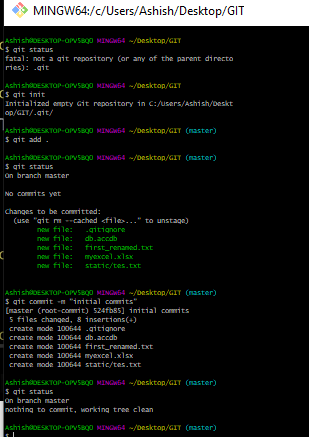
Run $ git log –p -1

o/p



My panda deleted, learn another,

# Next video:12 , Unstaging & Unmodifying Files In Git | Git Tutorials #12



Suppose chnguing something in file first\_renamed file and save:

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: first\_renamed.txt

no changes added to commit (use "git add" and/or "git commit -a")

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

Now, git add first\_renamed.txt

Came in staging area,

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git add first\_renamed.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: first\_renamed.txt

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

Now we want to unstage the above staged file;

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: first\_renamed.txt

no changes added to commit (use "git add" and/or "git commit -a")

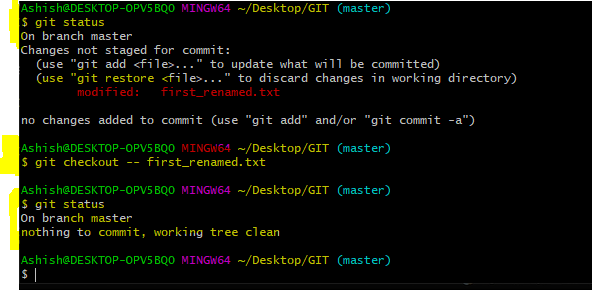
--

it came to unstage now by doing above command.

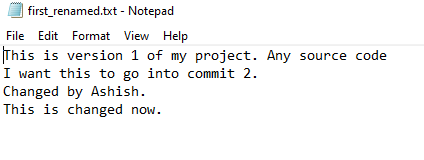
Now suppose we modified first\_renamed.txt,and realised we have done mistake, changed to wrong and saved, then realised I have removed important info ,

In that case whatever we modified , that we can restore/unmodified

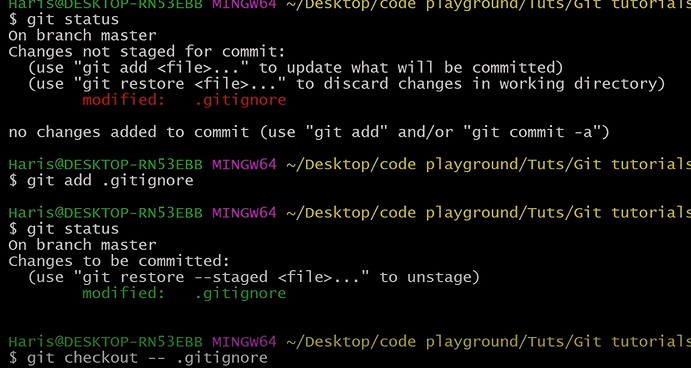
Whatever we changed , it will get match to prev commit

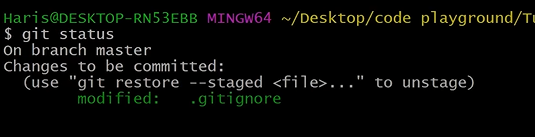


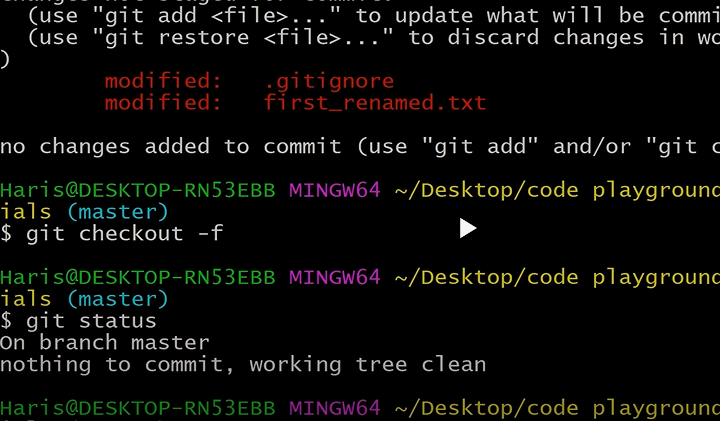
And whatever we changed , it gets rolled back.



If after modifying, we executed add command, then it cant get rolled back.







Means, whatever changes we did that got loosed, and match with prev commit.

# Next video: 13 GitHub: Working with Remote Repositories | Git Tutorials

Go to Github.com, signup.

Create new repo as e.g. learning git.

Go to Git folder where we were working,

Open git bash

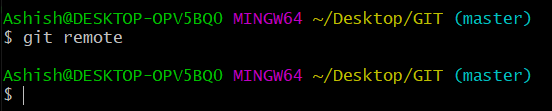
In the folder named GIT in desktop, where we are working, for which we make repo,

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

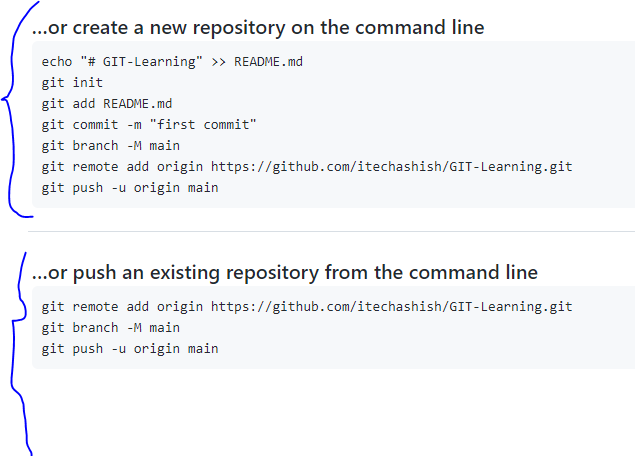


Here after execution of git remote, nothing came because…

Since git sis distributed control system where we push code/files in server . we can pull/push, if we are pulling , then all history, when who changed and all will come in my computer.

Push means , to send our code in remote repo.

So, we have to make remote now , that is repository or website which is in internet eg, gitlab, bitbucket, github wtc.



Since already we have made repo of GIT folder second para we will do which came after we made repo in github:



It’s like adding website(git remote add origin https://github.com/itechashish/GIT-Learning.git

) to remote by keeping alias name of website as ‘origin’.

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git remote -v

origin https://github.com/itechashish/GIT-Learning.git (fetch) -> telling origin

name ka url hai and from here we will pull

origin https://github.com/itechashish/GIT-Learning.git (push) -> telling origin

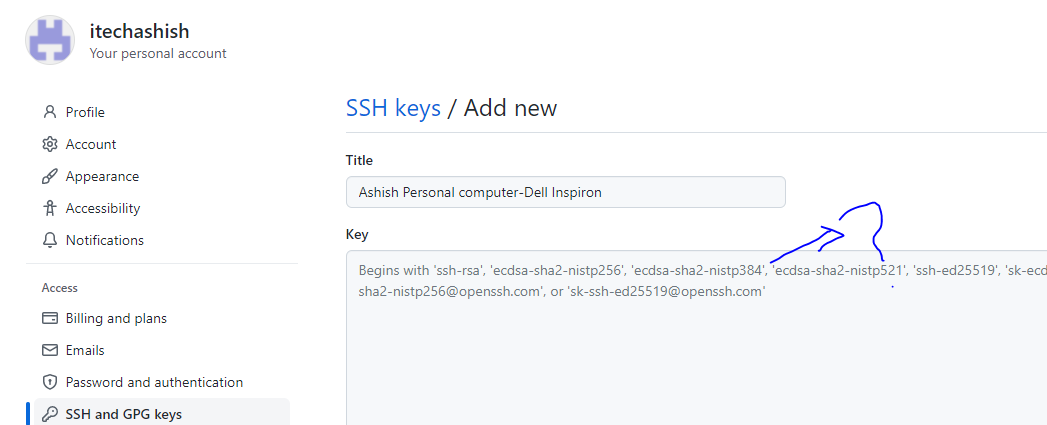
name ka url hai and from here we will push

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

Now $ git push –u origin master

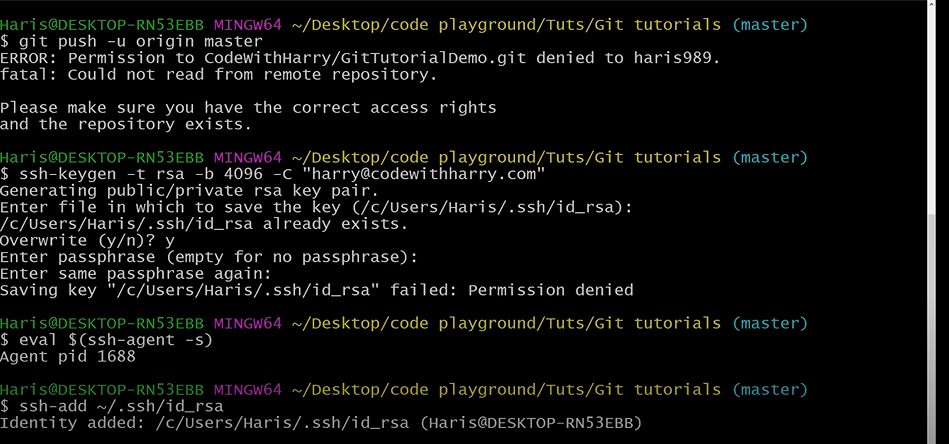
If it says no permission,

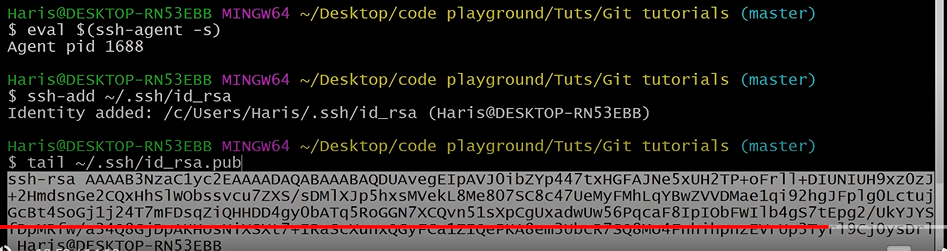
Then , go to setting in github-> ssh and gpg keys->



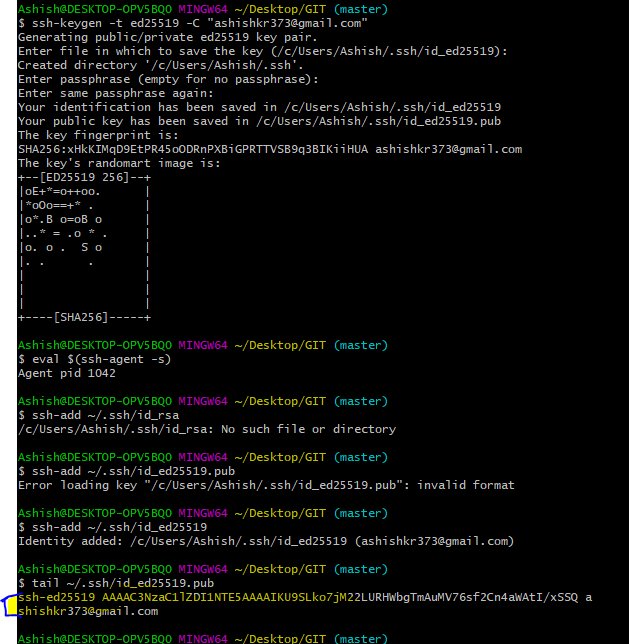
If we don’t know how to write here key,

Goto google search: ssh key github and generate using command





Mine sample:



Highlighted is the ssh key , copy and paste there, add key.

Now push using push command.

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git status

On branch master

nothing to commit, working tree clean

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git remote

origin

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)

$ git push -u origin master

Enumerating objects: 11, done.

Counting objects: 100% (11/11), done.

Delta compression using up to 4 threads

Compressing objects: 100% (7/7), done.

Writing objects: 100% (11/11), 27.64 KiB | 1.26 MiB/s, done.

Total 11 (delta 1), reused 0 (delta 0), pack-reused 0

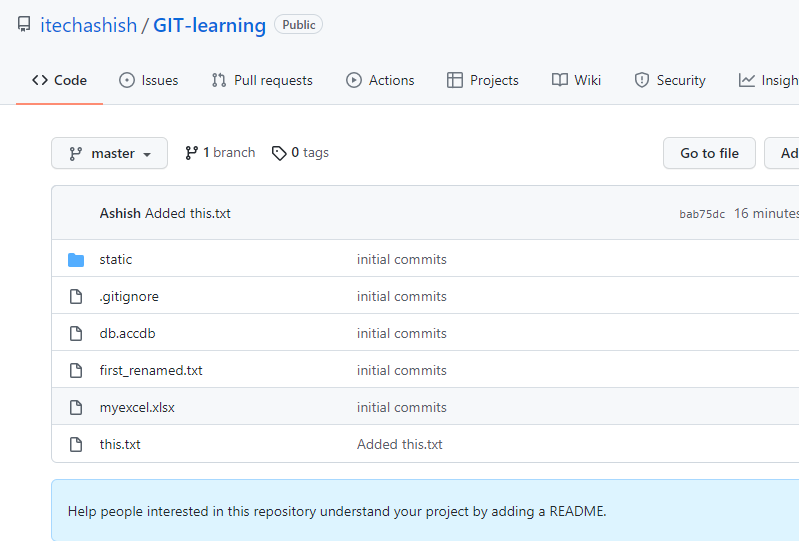
remote: Resolving deltas: 100% (1/1), done.

To https://github.com/itechashish/GIT-learning.git

\* [new branch] master -> master

Branch 'master' set up to track remote branch 'master' from 'origin'.

Ashish@DESKTOP-OPV5BQ0 MINGW64 ~/Desktop/GIT (master)



Suppose we created one file as test.txt in GIT folder manually,

