Git is a version control system

A version control system is a system which helps us in keeping track of files

GIT:

Easy file recovery/ easily recover files

Who introduced an issue and when?

Rollback to previously working state

History of version control systems:

Local VCS(version control system): Used database to keep track of files

Pros: Can track files and rollback

Cons: If you lose/damage your hard disk everything is lost

Centralized VCS: Storing of the code/files on a centralized server and than pulling and pushing files from it, more than one people can collaborate and push and pull the files and make their own changes but the master file or the final version of the file would always be stored in the main server

Pros: Data can be safely secure on a server and multiple people can work on it at same time

Cons: In case the server get’s damaged or destroyed the whole files or the old version of the files would be lost and rollback won’t be possible, also collaboration is not possible

Distributed Version Control System: This also uses a central server but in this case all the users collaborating on the project would get all the file history and change history , i.e the whole repository so in this case so when a user pulls the data he get’s the complete repository, so even if the central server get’s damaged each and every user would have complete repository on their system so the chances of losing the files would get minimized, so that means each user has a complete backup of all the files, and it’s so smartly made that it doesn’t take much of the disk space. So if a project is of 2GB of space the system should have a smart mechanism to ignore the data files and only pull the source file/code, and only save the change made in the file and not save the complete file again and again.

So Git is an example of distributed version control system

How was git created?

It was created by linus torvalds(wrote the linux kernel, major contribution)

From 1991-2002 linux OS was developed through patches and archive files( so basically all the contributers used to send each other the files and there was no version control system)

But in 2002 the contributers realized that they’re going to need a version control system

So in 2002 they used a VCS by the name of Bitkeeper

So from 2002-2005 they used bitkeeper VCS and everything was going great and linux was being developed easily and nicely

But in 2005 something happened in 2005 due to which bitkeeper told that now they can’t provide bitkeeper for free, you have to pay a heavy amount to use bitkeeper and put the contributers in great shock and then linus tornvald told everyone that he’s gonna do something about it and no one even thought about it and linus tornvald came up with a even better version control system and today it’s famously known by the name of GIT

So in 2005- Free of charge status was revoked

Why is git better and best of all? Characteristics of git. Features of GIT:

Fast and liable, captures snapshot and not differences means if you change a file it will capture the snapshot.

Almost every operation is local: Means that almost all the operations are saved in your local computer itself we don’t need to push the files in any server , we can work easily in our own computer and then push the files to the remote git repository like server(centralized server) like github. Push whenever you’re done with your operations. And we have wrote almost because some operations like pushing( sending to remote repository), pull are not local, these operations are remote meaning they’re performed using the internet.

GIT has integrity i.e the files are safe and protected. Each updated file has a different SHA1 checksum even if minutes of updations are made the SHA1 checksum would change, so a user can find the checksum of the file and match with with the SHA1 checksum to be same so if checksum of the file is same that means no one has updated, modified or manipulated the file while the person was downloading it.

Git generally only adds data: if you have a repository it’ll grow, so if we have git repository we’ll keep on adding the changes in the repository and barely delete anything from it

.git folder contains everything about your project, project history and we can pull a particular version of the file, if you request the latest version of the file it’ll fetch you the latest version of the file

What is github?

Github is a hosting service/ website which hosts the repositories of git

GIT command: GIT bash takes linux commands and not windows commands

pwd = Present Working Directory

cd = change directory

cd /c = to go to c directory (not cd C: like windows)

we can push/oull/add files using the bash

push: sending file to server/ remote repository

pull: pulling files from the server

add: adding files to the repository

Real programmer use command line as it provides better functionality and commands also this works behind the scenes so you know what exactly are you doing, and know how git actually works

You can easily open the git terminal/bash in any directory by simply pressing shift + right click and click on Git bash here to open the git terminal on that directory instead writing the whole cd command to change the directory to that folder/path

To configure your name in git you have to write the following command

git config –global user.name “Tanish”

To configure your email in git you have to write the following command

git config –global user.email “solanki.tanish57@gmail.com”

It tells git who you are and whenever you’re making some changes in the repository or adding new files it’ll keep track of your name and contact details to help people to contact you

To get the list of configurations you simply write the command:

git config –list

upon writing this command you’ll see several details pop up including user.name = Tanish

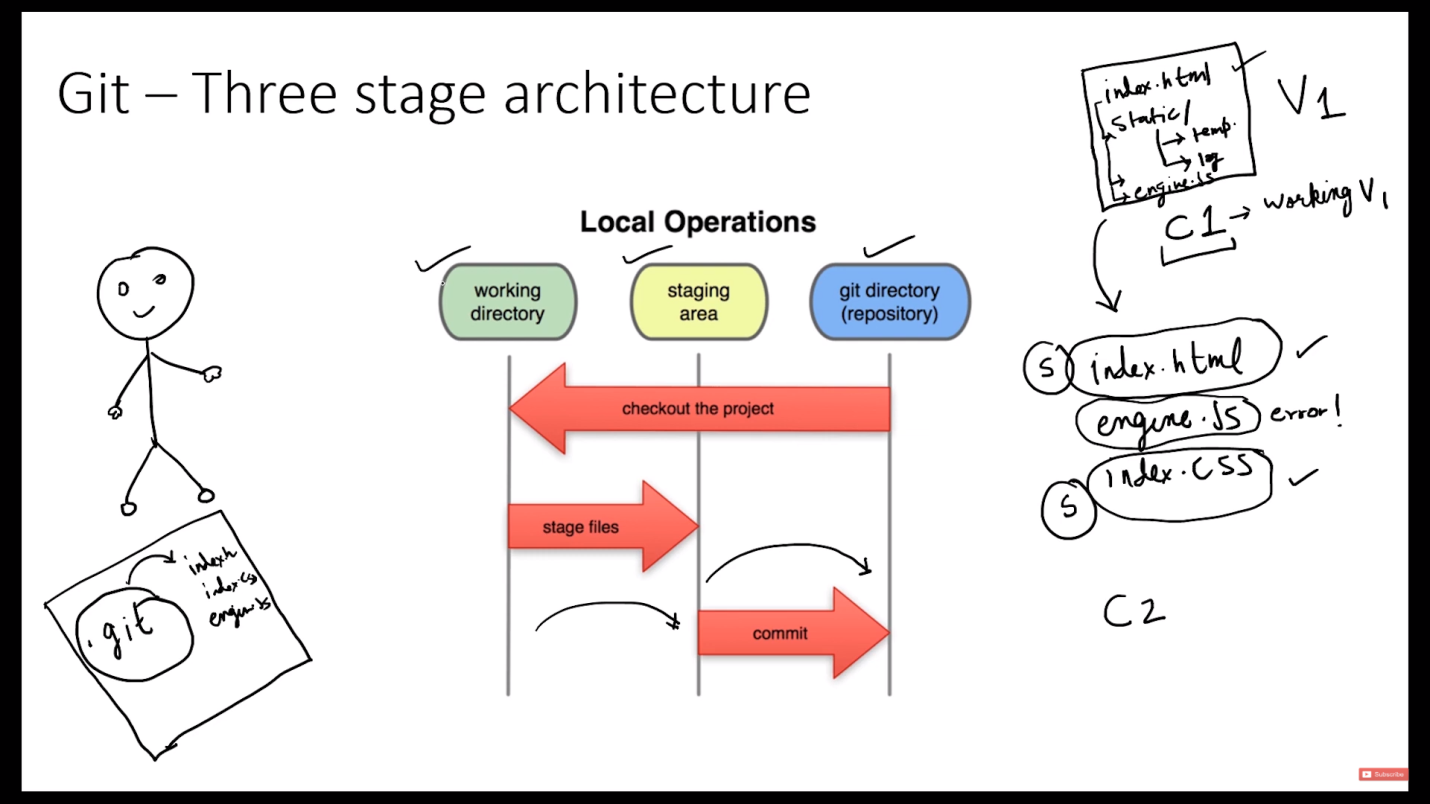
and user.email = [solanki.tanish57@gmail.com](mailto:solanki.tanish57@gmail.com)

to find the name of the user:

git config user.name

for email: git config user.email

GIT: Three stage architecture-



Tracking our first Git project:

Open git bash in the repository folder

And write the command: git status

It’s used to give the status of the git repository, but initially it’ll give an error that neither are we inside a parent repository nor it is a git repository

So no I have to make it a git repository so I’ll write the code:

git init : it is used to initialize a new git repository, it’ll give us a message that the that this folder is created as a git repository

now if we enter git status it’ll start tracking and give an output

to add all the files at once on the staging area enter the command:

git add --a , here --a means all

now if you check the status using git status it’ll show you all the staged files in green colour

now we’ll just commit the file with a message using:

git commit –m “Intitial Commit”

and now if we check the status using git status:

it’ll show the message : nothing to commit, working tree clean, meaning the snapshot have been taken and all the changes have been commited

upon writing git log : we can see all the commit we have done so far, here we get info : the username(tanish) have made a commit at a certain time in which he has written initial commit, and we even get a hash of the commit

now I have made some changes in my first.txt file

now if check status it’ll track that I have modified first.txt

Also I’ll make some changes in my excel file

And upon again checking the status it’ll show me that even the excel file is now modified

Now I only want to send first.txt to my staging area as I’m not yet sure about my excel file

So I’ll write git add first.txt

And now when I check the status it’ll show me that myexcel.xlsx is modified but not yet added to staging area, but my first.txt file is staged to be commited

Now i’ll write git commit -m “Changed first.txt and added better design”

Now for the record this type of commit is not good , we never write changed as commit itself means change, we only use commit when we have changed something

So upon writing the above commit it’ll show us that we have made changes to 1 file , made 2 insertions and 1 deletion

And now on checking status it’ll still show that myexcel.xlsx is modified and not yet staged, so git add is used to stage individual files and then upon writing commit they’ll be commited

Cloning a remote Git repository from GitHub

But first if we want to delete a git repository/ stop tracking a git repository:

We type the command: rm -rf .git

So it deletes the complete git folder, its a very fatal command

So after this command the tracking of the files present in this folder would be stopped

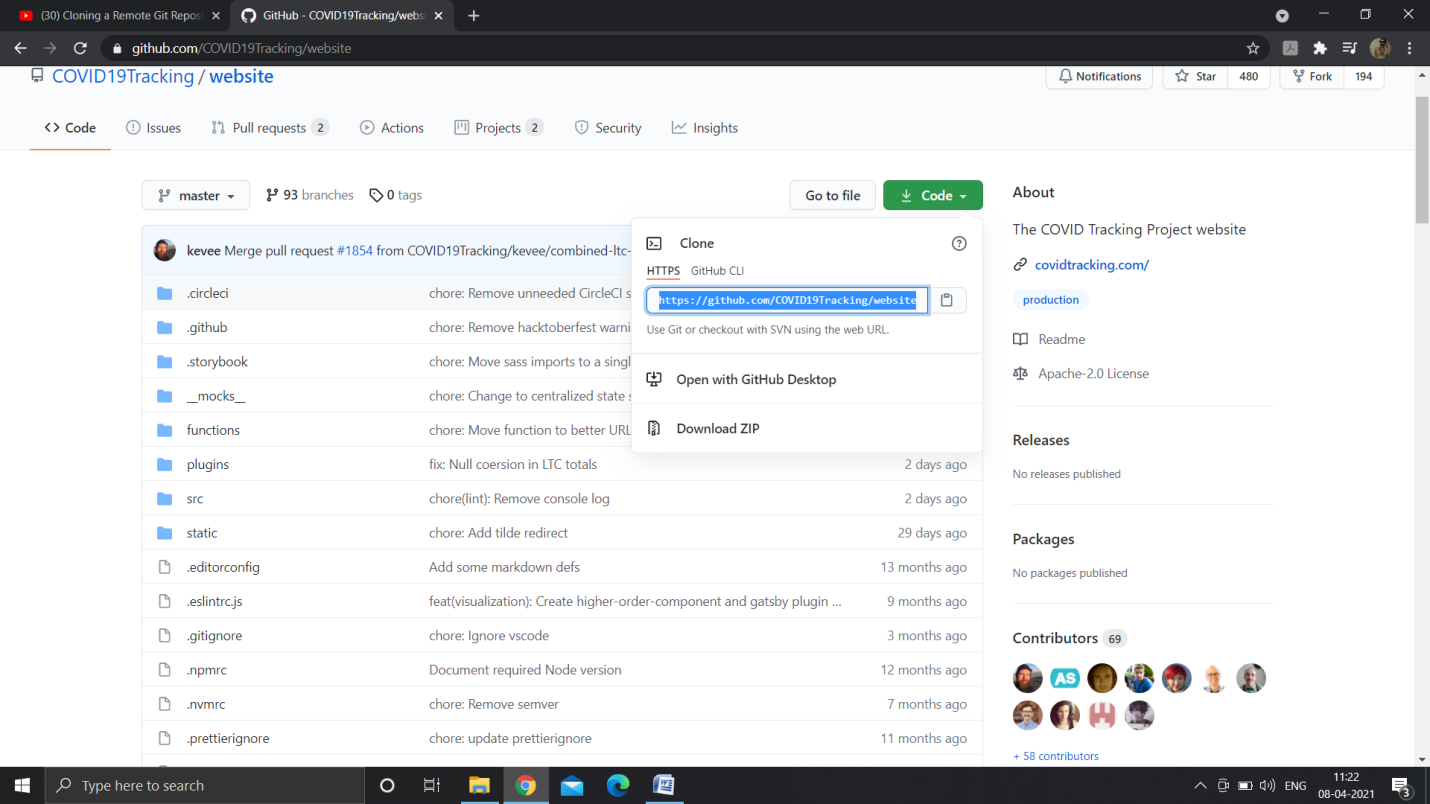
pwd: present working directory

ls: list of content in whichever folder you are

cd: change directory

now to clone a git repository from git hub , go to the github website

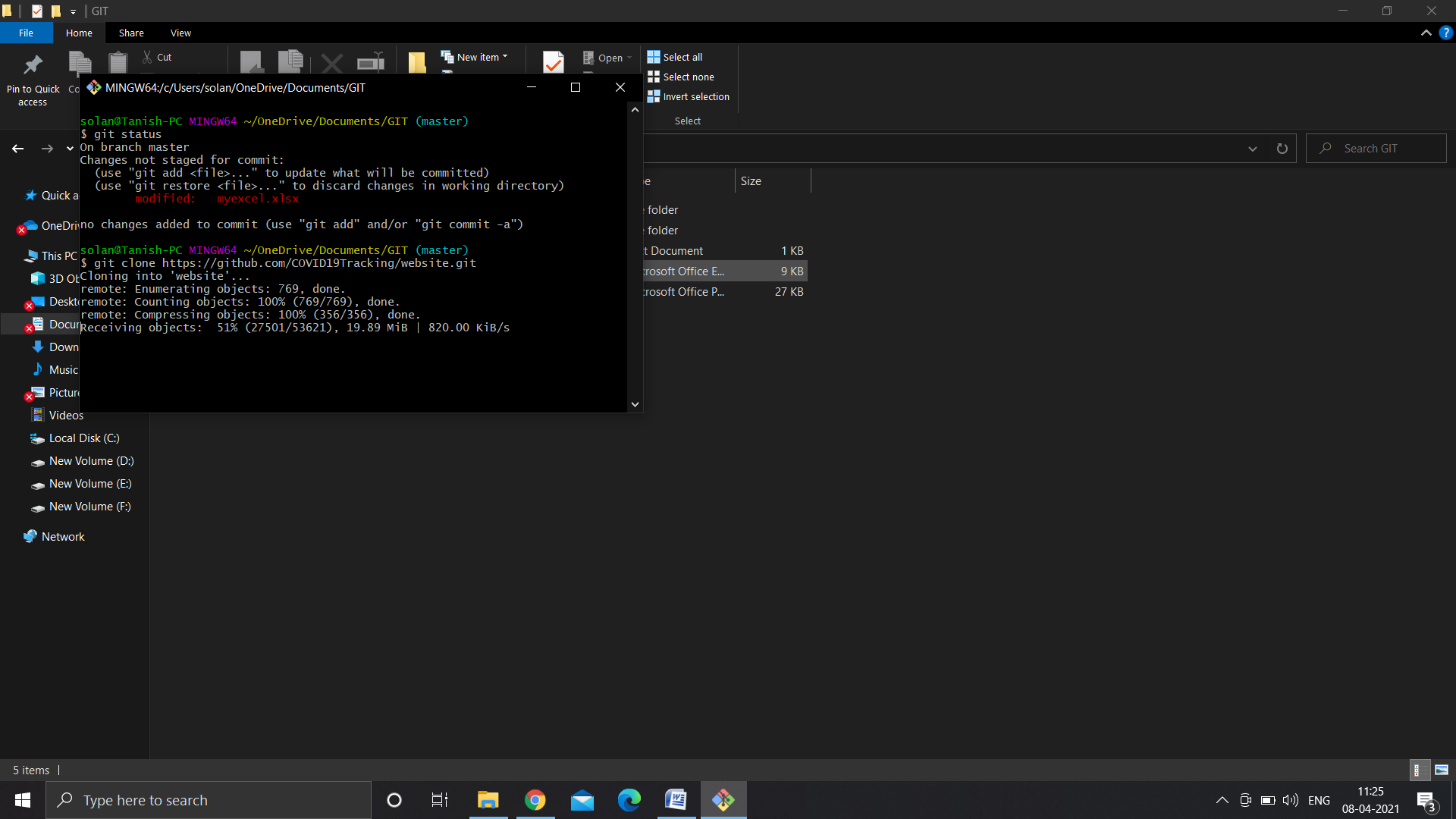
go to any project and click on code dropdown and then copy the HTTPS link/URL given there



Now come back to your git bash and run the following command

git clone <https://github.com/COVID19Tracking/website.git> (the link you copied)

doing this the git will start cloning the repository in your computer



And make a folder website, but if you want to make your own name for the repository you can simply write :

git clone <https://github.com/COVID19Tracking/website.git> clonedSite(the name)

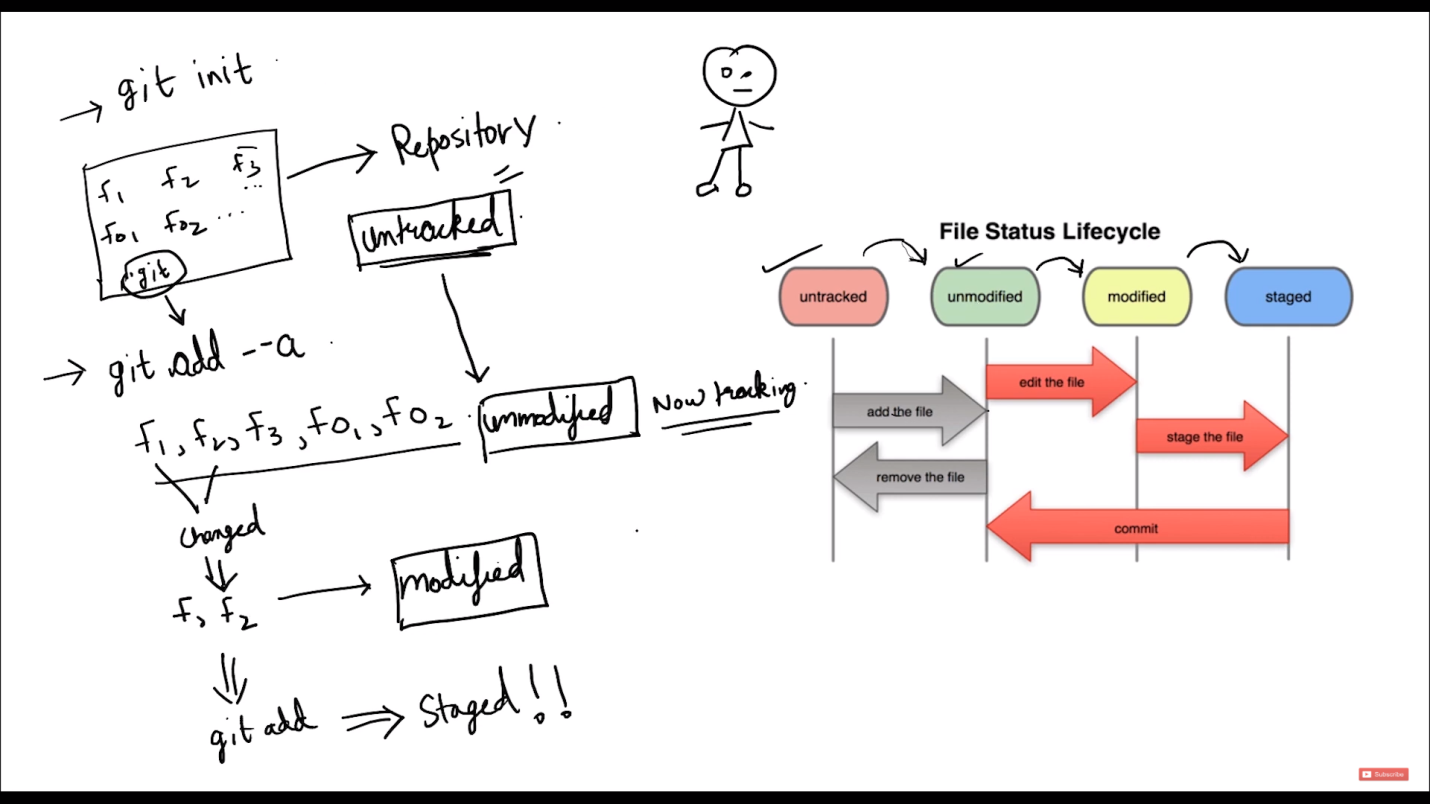
now we have to enter the folder that is cloned, i.e website

simply enter the command: cd website/

and it’ll change the directory to website

now you can edit some files, check logs, add the changed files to staging area and then commit the changes

File Status Lifecycle:



So when we add the files for the first time they go from untracked to unmodified and also they get staged to be committed but when we change a document after adding them to stage, the files which are staged would remain the same and stay ready to be committed but also the file changed would become modified and would show us that it’s modified and not staged yet, so when you enter the commit command git would ignore the modified file and commit the staged files (including the now changed file’s old version) but now if i add the changed file using git add first.txt, they new modified file would be staged, now on checking the status it’ll show the newly modified first.txt in the staging area and now when you commit these new 3 files would be commited

.gitignore: Ignoring files in git:

To create a .log file using bash we enter the command: touch error.log

This will generate an error.log file in our folder

So now this .log file doesn’t have any importance in our software so we will want be ignore it

Upon running git status it’ll show us that error.log is an untracked file

But we don’t want to do anything with error.log we want to ignore it and upon being checked by git status I want it return output as working tree clean

So now we’ll create a file using git bash and we’ll write the code:

touch .gitignore

this will create a new .gitignore file and we’ll add some contents to it

now upon checking the status we’ll see .gitignore and error.log are untracked files

now coming back to our .gitignore file, we’ll type error.log in it and save it.

And now on checking git status we’ll see after typing error.log in .gitignore its not showing the name of error.log in untracked files

Now we’ll add all the files to the staging area(i.e .gitignore)

git add --a

ok we can also use git add .

and now on checking status we’ll see .gitignore is in the staging area and ready to be commited

and now we commit

git commit -m “Added gitignore”

so now even if I change the error.log file it’ll not show anything in gitbash as I’ve added this file to .gitignore

so this means basically if you create a .gitignore file and add files to it those files added in this would be ignored by git

now consider I have many .log files made by my software and now i have to ignore all the .log files generated, so I now want to ignore any type of .log files

so now to do it we have to open our .gitignore file and write:

\*.log

This means ignore any and all kinds of .log files

So now on checking the git status it’ll ignore all log files and only show .gitignore to be modified/changed and unstaged

Now we can also ignore an directory

We’ll create a directory dir and add some files to it

And now on checking the status it’ll show dir/ is untracked i.e this directory is untracked

Now I don’t want to track this directory so I’ll open my .gitignore file and write :

dir/ in it

now on checking status we’ll see that this directory is no more showing up in untracked files cause the bash has ignored the directory

now if we create a new folder and add another folder to it by the name dir and add some files to it on checking the git status it won’t show anything except modified .gitignore file, that’s because our static folder had a folder named dir and in our .gitignore we have wrote dir/ so any folder named dir would be ignored by our bash

any folder named dir would be ignored

but why isn’t it showing static folder to be untracked?

Because it has dir folder in it and dir is ignored and static folder is considered to be empty and git ignores any empty directories

So now if we add another text file to the static folder and then check status it’ll show up to be untracked

Now we’re gonna add all the untracked files to the staging area using:

git add –a

and commit all the staged files

git commit –m “added static”

and now when we checked git status it will show working tree clean

but now what if I want to ignore only the outer git folder I’ll write /dir/ in my .gitignore folder

so now upon checking the git status we’ll see our static/dir/ to be untracked

we can also ignore a complete path by simply adding it to .gitignore:

static/dir

and this path would be ignored

but the file inside the static folder is already being tracked so now if I change that txt file in the folder and check status it’ll show the file to be modified

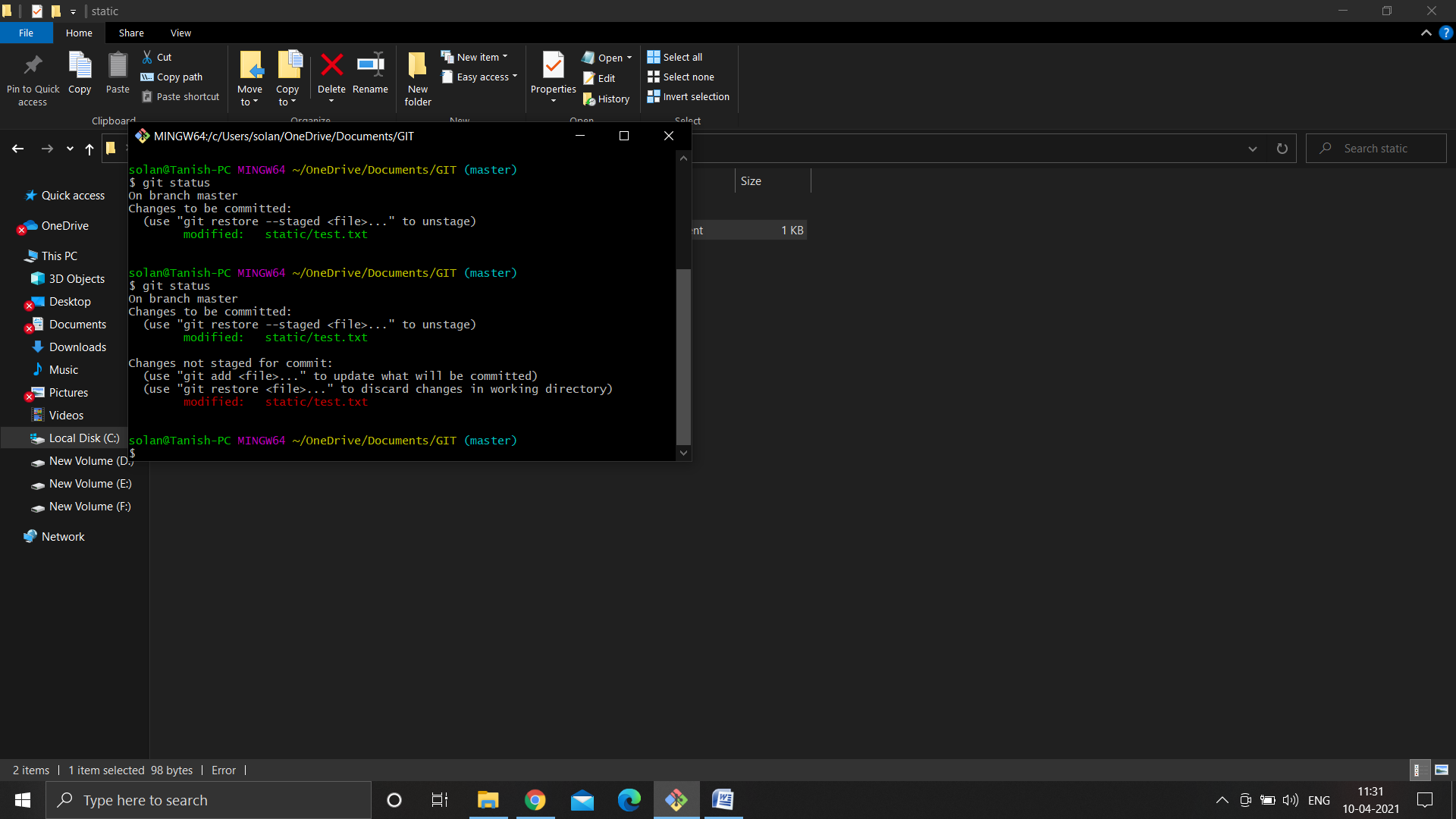
git diff: Showing changes between commits and staging area & working directory

so after the editing the txt file in static folder when we check the status It’ll show that file to be modified and now I have to check difference between working directory and staging area/ committed and staged files too

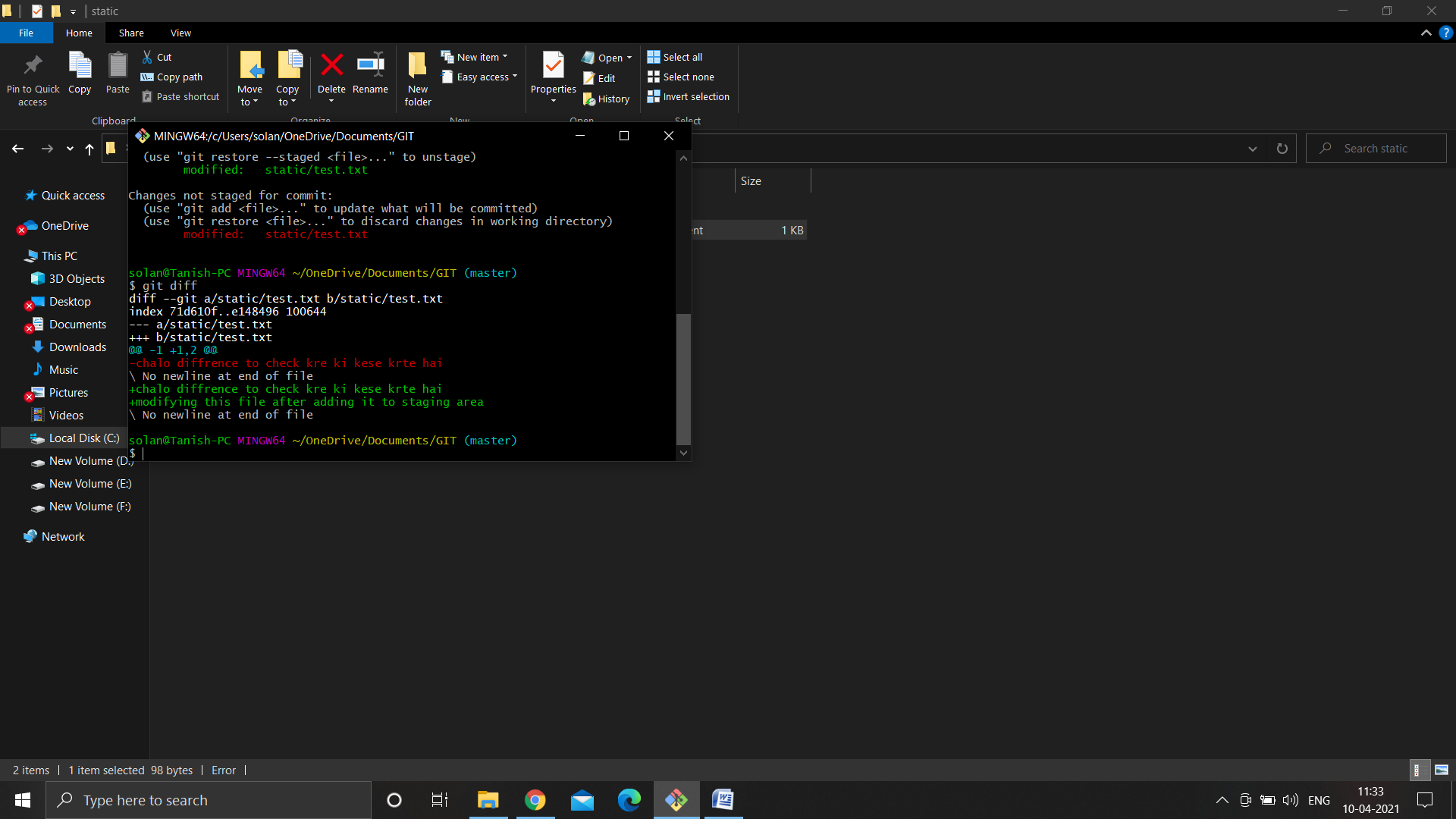
so now we’ll add it to staging area using git add .

and now on checking status It’ll show that the file is in staging area now

so we know if we modify this txt file again after adding it to staging area it’ll show both staged file and modified file on checking the status i.e both in green and red



And now on running git diff it’ll compare working directory with staging area



Here it’s showing that in static the test.txt file has two versions a and b

Here it’s showing that the staging area has the file in red and lines have been removed from it and lines have been added in the working directory(in green)

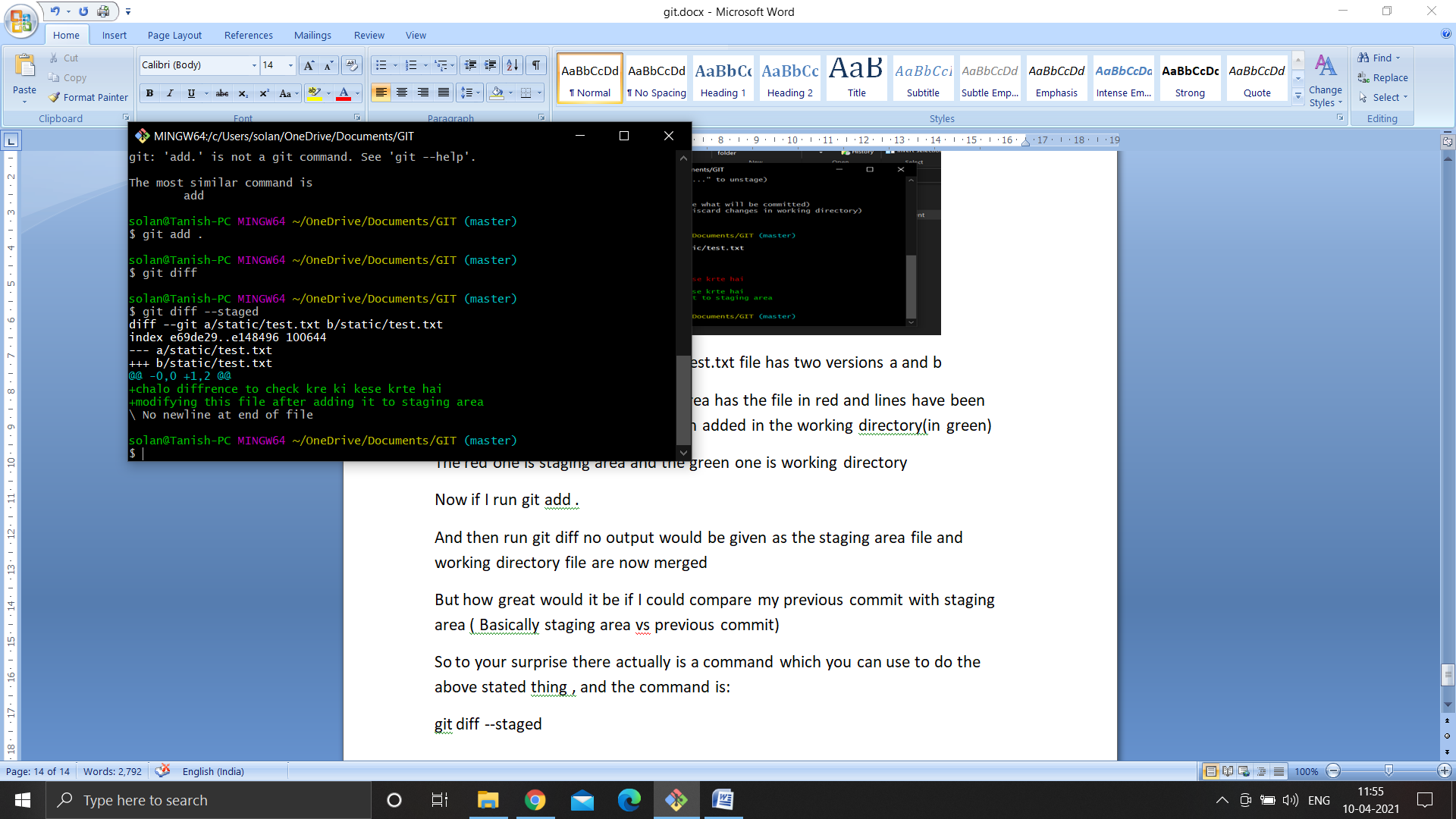
The red one is staging area and the green one is working directory

Now if I run git add .

And then run git diff no output would be given as the staging area file and working directory file are now merged

But how great would it be if I could compare my previous commit with staging area ( Basically staging area vs previous commit)

So to your surprise there actually is a command which you can use to do the above stated thing , and the command is:

git diff –staged

Skipping the staging area:

There would be times when we’d want to skip our staging area in git

So now we’ll change the file first.txt in our directory

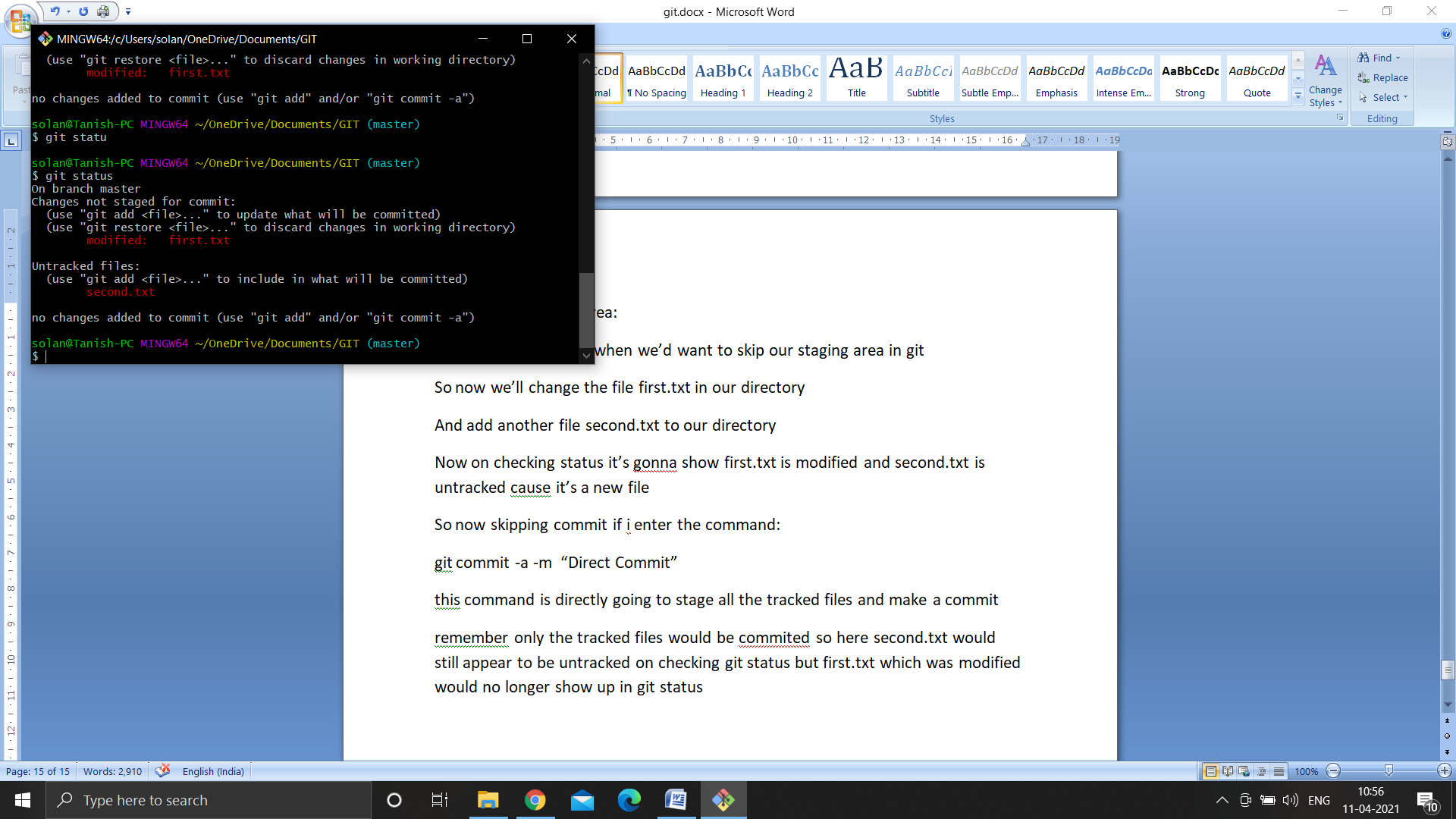
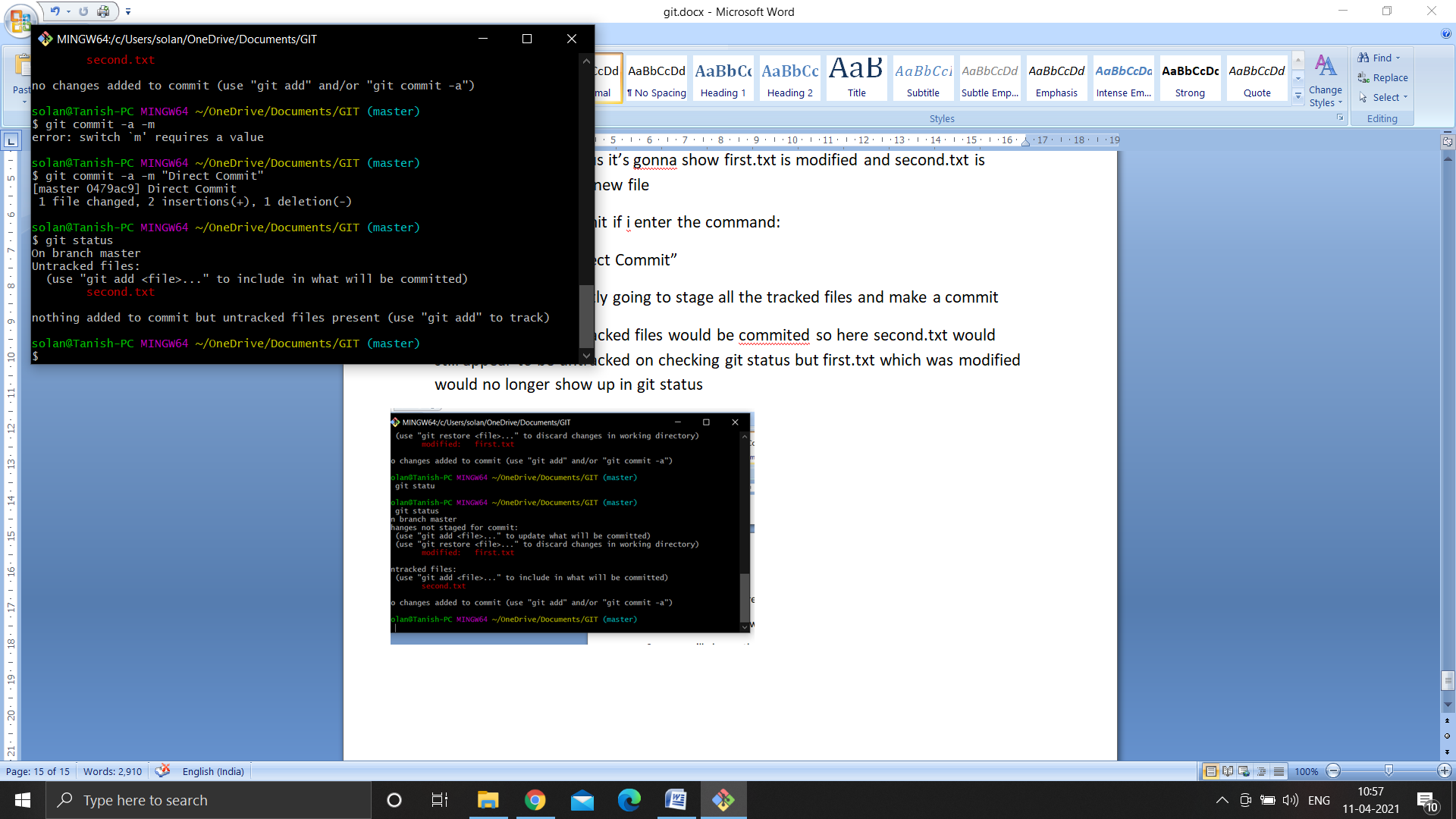
And add another file second.txt to our directory

Now on checking status it’s gonna show first.txt is modified and second.txt is untracked cause it’s a new file

So now skipping commit if i enter the command:

git commit -a -m “Direct Commit”

this command is directly going to stage all the tracked files and make a commit

remember only the tracked files would be commited so here second.txt would still appear to be untracked on checking git status but first.txt which was modified would no longer show up in git status

so all the untracked files cannot be directly commited we have to add them to staging area manually first using add

so now i’ll enter the command git add second.txt to add second.txt to the staging area and now on checking the git status it’ll show my second.txt in the staging area

and now we’re going to enter the command :

git commit -a -m “added second.txt”

and now on checking status it’ll show the working tree to be clean

Moving and Renaming Files:

Now if you directly rename a file and then check git status it’ll show that you have deleted the second.txt file and added a new file third.txt which is untracked

But now when I enter git add . and then check git status git will recognise that you have renamed the second.txt -> third.txt and it’ll show second.txt is renamed

And then i can git commit -m “Renamed second to third”

So this is one way of renaming a file directly and then commiting it

But now we’re gonna learn how to remove and rename files using git bash :

So to remove a file using git bash simply enter the command:

git rm file-name

for example I have to delete the file second.txt

git rm second.txt

this will delete second.txt

and now on checking git status it’ll even stage that second.txt is deleted

so this is a plus point of removing files from the git bash itself that you don’t have to manually add the changed file to staging area the rm command itself would stage the changes made for you and you can simply commit after using this command, if I removed a file manually then I’d have to run git add .

instead i can simply remove a file directly from the bash itself and then it’ll even get staged by itself and I can simply commit it

now we know how to remove files using git bash using git rm command

now we’ll learn how to rename files using git bash

so to rename a file using git bash simply enter the command:

git mv currentfilename renamedfilename

for example i have to rename the first.txt file to renamed\_first.txt

git mv first.txt first\_renamed.txt

this command will change my first.txt file name to first\_renamed.txt

and now on checking git status it’ll add the renamed file to the staging area by itself without having to use git add . command , status will show

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

and then we’ll commit the changes made using git commit -m “renamed”

to make our working tree clean

mv is called move command btw, this command is used in linux to move and rename files, this means move first.txt to first\_renamed.txt

but we’ll keep it simple mv is our rename command that’s it

NOW SOMETHING IMPORTANT:

Consider we add myexcel.xlsx to .gitignore and save it

On checking status it’ll show that we made some changes to our .gitignore

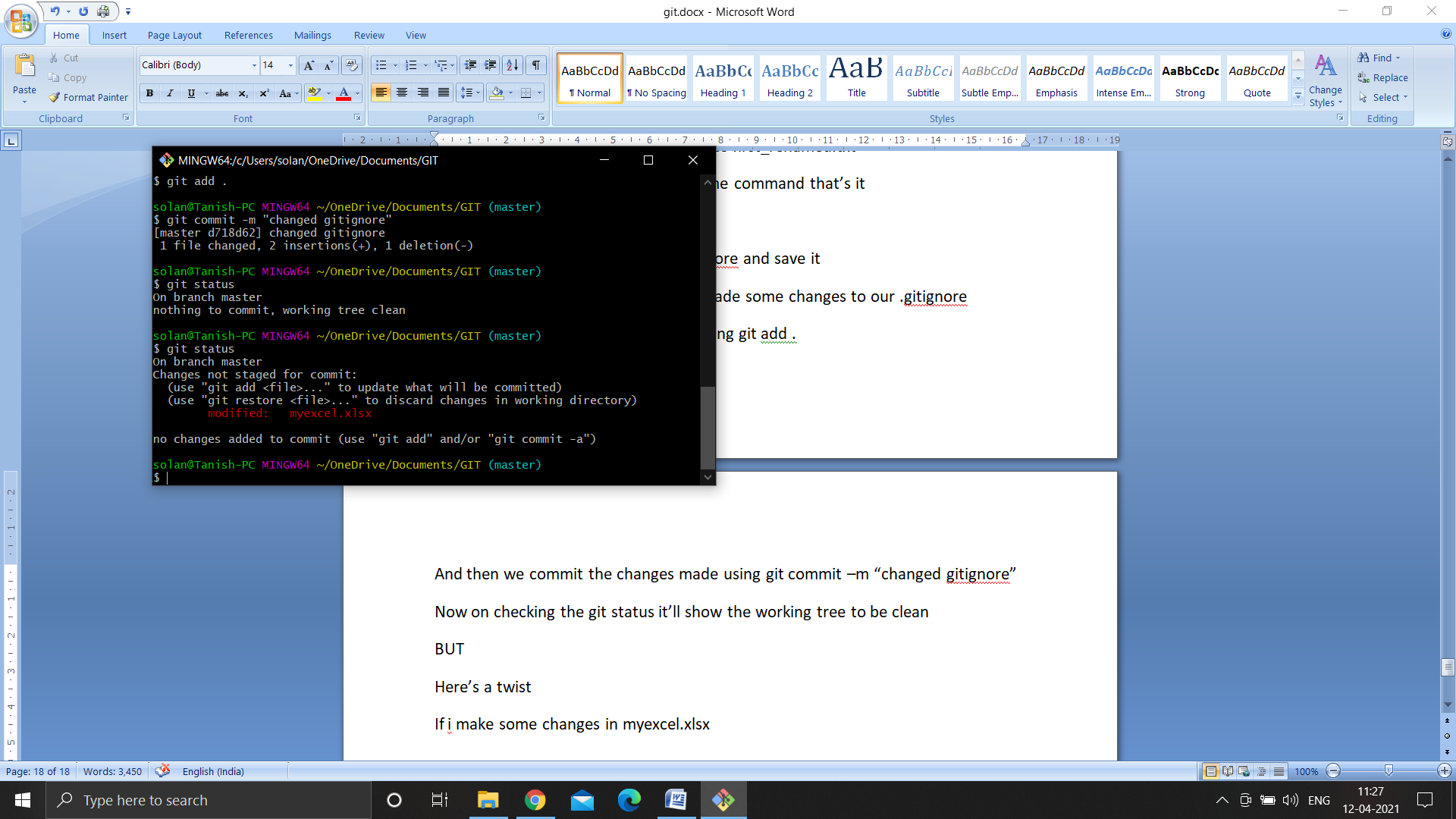
We add this changes to staging area using git add .

And then we commit the changes made using git commit –m “changed gitignore”

Now on checking the git status it’ll show the working tree to be clean

BUT

Here’s a twist

If i make some changes in myexcel.xlsx and then check status it’ll show up to be modified in my git status

This happens because we were already tracking it before we added it to our gitignore

We were tracking myexcel.xlsx file from before

And when we changed our .gitignore then too it’ll be tracked till we explicitly untrack it (tracking system se nahi hataenge git k)

So now we’re going to learn how to remove some files from git’s tracking system explicitly

Basically after adding something to gitignore we have to tell our git bash that I don’t want to track this file anymore so to do this we run the following command

git rm –cached myexcel.xlsx

don’t worry this will not remove the file myexcel.xlsx from our directory this will just untrack the file so this file won’t be tracked anymore if it was being tracked

the file would be present in the directory(you can check)

this command will only stop tracking the file

and now on checking git status it’ll show that myexcel.xlsx is deleted but actually it’s not deleted we just have started untracking it

that means we have removed myexcel.xlsx

so now we’ll just directly commit using

git commit –m “removed myexcel.xlsx”

and now on checking status it’ll show the working tree is clean

so now even if i make some changes to myexcel.xlsx file it’ll not disturb me when i check my git status

but even after removing it from our tracked files why isn’t it showing up to me untracked?

Cause we have added it to .gitignore, if we remove it from .gitignore and then check status it’ll show myexcel.xlsx is untracked in my bash

git log: Viewing & changing commits in git

if you enter the following command in the git bash:

rm -rf .git

this command will destroy the git repository meaning this folder is not a git repository anymore

on checking git status it’ll show not a git repository

now we’re going to clone a new repository pandas from github using git clone URL folder name

git clone <https://github.com/pandas-dev/pandas.git> mypanda

the above command will copy the pandas gitrepositor to mypanda folder and we’ll then open the git bash in mypanda folder

on checking git status it’ll show nothing to commit, working tree clean

on checking git log we’ll see all the commits made, when, by whom and what

now I want to see the diff, means I wanna see what changes has been made in those files in that commit

so i’ll run git log -p and this command will show me what has been removed along with commit

so in green we see all the lines added and on red we see all the lines removed

so if I write git log -p -3: it’ll show me only 3 commits not all along with the diff

another fun/good command is:

git log –stat

on running this command it’ll show us in very short that by which author at what time changes have been made and how many files have been changed, how many insertions and how many deletions have been done during that commit basically it’s a short summary of all the commits made

git log --pretty=oneline

this command will show me all the commits in one line

one line for one commit so basically if i have to print all the commits on a paper i can simply copy this output and paste it and print it

the output of this command is very pretty and structured and easily understandable

git log --pretty=short

on running this command this will show me the details about the commits made in short this command will show me the author of the commit its hash number and its detail it doesn’t show any diff or date

this command shows me the details of the commits in short this command is helpful if i have to find a commit really quickly I can find it really quickly using git log --pretty=short

another one is

git log --pretty=full

this shows a bit more information like author commit and message

short doesn’t show the who committed

but full shows author and who committed

author is the person who created the file the first time(jisne janm diya file ko)

commiter is the person who changed the file

if i write git log --since=2.days

it’ll show all the changes and work done in the last two days

i can even write git log --since=2.weeks to see all the work done in the recent two weeks

even write 2.months, years etc

git log --pretty=format:”%h -- %an”

so here using this command I have specified the format for the output of the log command (first show %h give two dash -- and then %an)

%h is abbreviated hash commit (hash ka commit) abbreviated mtlb short me hash

%an is author name

If i wrote %ae it would’ve given me author’s email

For more formats you can visit: git-scm.com/docs/git-log

git log -p -1

this command will show me only a single commit

if I have to change/make amend to a commit

git log -p -1

this will show us the last commit made

so the last commit is made by Simon Hawkins

now I’m gonna make some changes to any file in the repository and on running the git status it’s gonna show me I have modified the file(authors.md in this case)

so what if I want to merge this change which I have made with that commit which is made by Simon Hawkins, i want to reflect the change made by me in the commit made by simon Hawkins and also i want to change the commit message

so i’ll have to enter the command

git commit --amend

as soon as i run this command I’ll get an editor(vim editor)(in bash only) and in that i’ll see the message written by simon

now to edit the message in the vim editor first i would have to press ‘i’ after i run the above command in to be able to edit and write what i want

and after writing I will have to press esc(escape) key and then I’d get a writing pointer at the bottom there i will have to enter –

:wq

And then this will exit out of the vim editor and make the amends i made to the last commit

Now on running git log-p -1 I’d see my written text in the last commit by simon

On checking the status i’d see that authors.md is still shown to be modified that is because i forgot to add it to the staging area

So i’d run the command:

git add .

and i will again have to enter the command git commit --amend

and then press i in the vim editor make some changes press escape key and then :wq to exit the vim editor and save the amends

and now we’ll see commit has been made

now if we run git log -p -1 we’ll see authors.md file has been changed and what line has been added would also show up there

so this is how we can change an old commit

also now on checking status working tree will me clean

Unstaging and Unmodifying Files in git

So first get back to our git directory and open bash in there

Change the file first\_renamed.txt

And check the git status it’ll show first\_renamed.txt is modified and not staged

Now we’ll add first\_renamed.txt to the staging area using git add .

And now on checking git status we’ll see first\_renamed.txt is now staged and changes have to be committed

Now I wish to unstage this file from the staging area, so to unstage a file from the staging area you can simply enter the command:

git restore --staged first\_renamed.txt

and now when I check git status it’ll show first\_renamed.txt is modified and unstaged

now by mistake I modified my first\_renamed.txt file and emptied it’s important contents and now i want to restore my file back to how it used to be so what do I do?