Git Learn

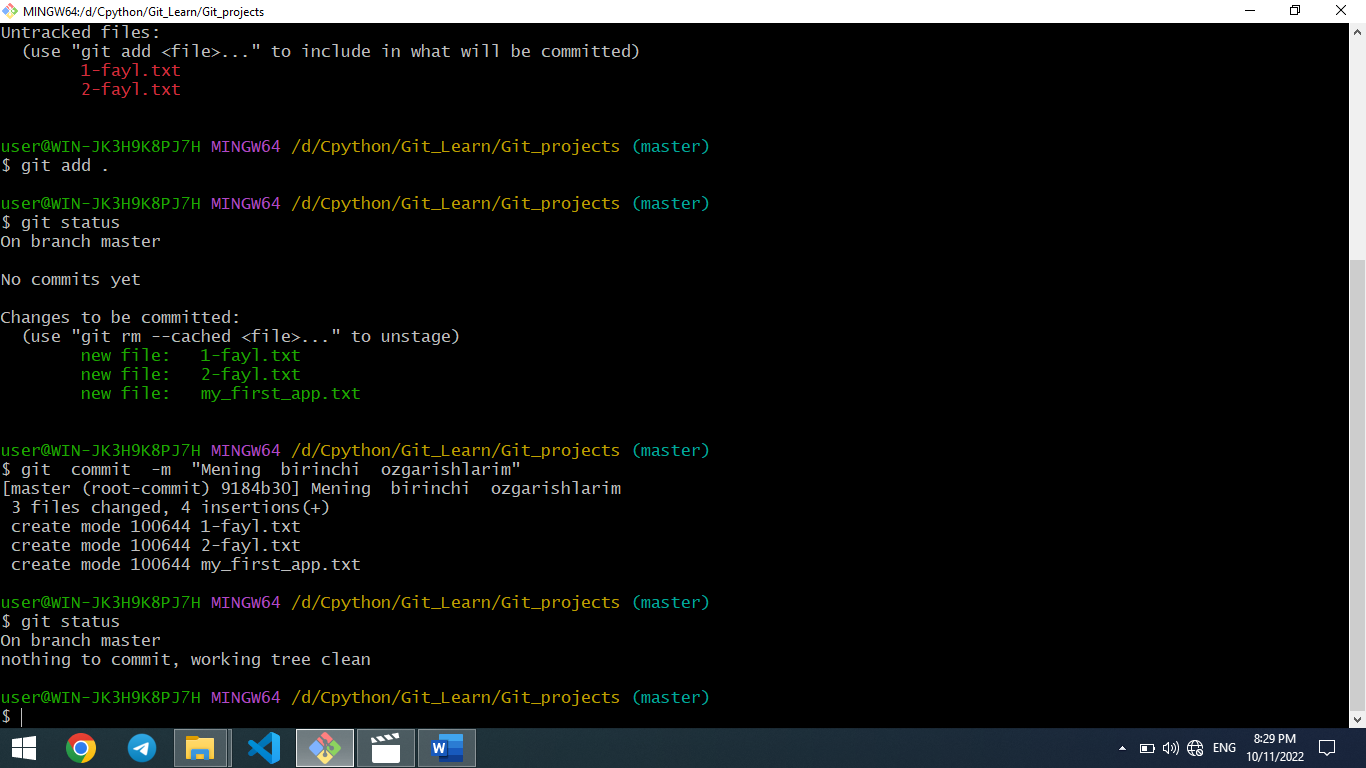
1. Git init --- > Biz yaratajak projectni gitda royxatdan otkazamiz. Buning uchun Wu project turgan yolni (yani \D>Cpython>Git\_learn …) kabi berishimiz kerak. Va Git init buyrugini bersak projectimiz gitdan ro’yxatdan o’tadi.

2. Git status -- > Bu buyruq projects dagi fayllar royxatdan otganmi , yoqmi, yangi fayllar bormi, yoqmi shularni aniqlab beradi

3. Git add -- > Fayllarni Gitda royxatdan o’tkazadi. Buni berish yoli git add fayl\_nomi (masalan nimadir.txt)

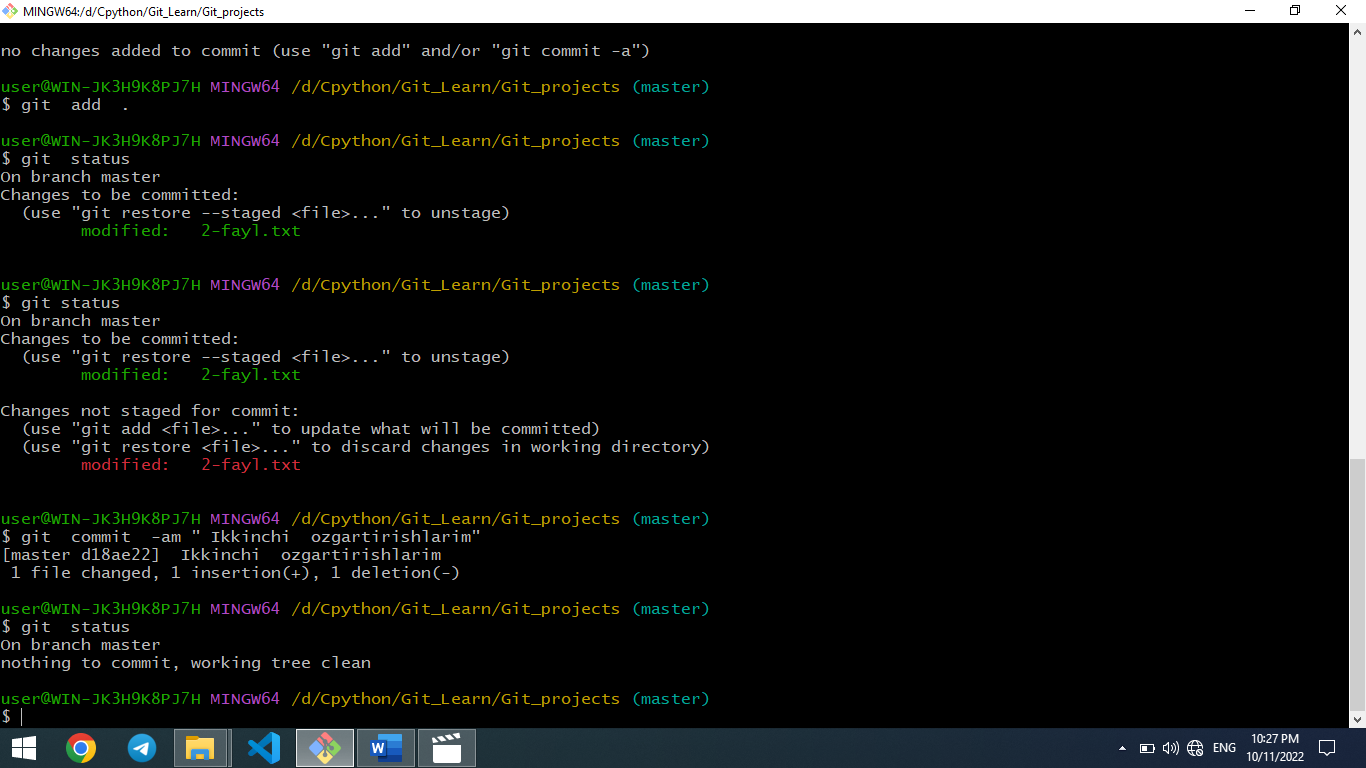
4. Git add . -- > Bu gitda barcha fayllarni ro’yxatdan otkazadi. Bularni git add buyrug’i orqali ham qilish mumkin. Lekin fayllarimiz bir nechta bo’lsa , git add . buyrug’i orqali hammasini ro’yxatdan o’tkazsa bo’ladi.

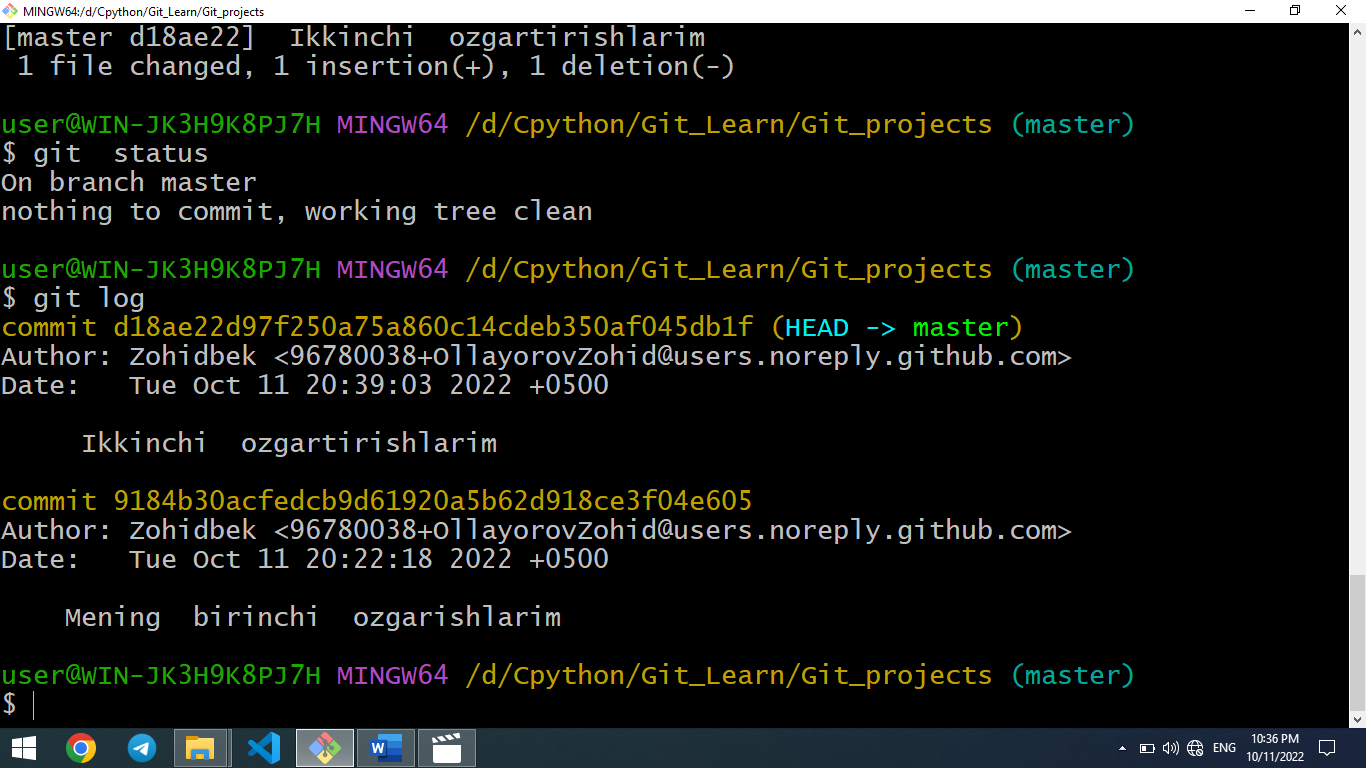
5. Git commit -m “Ozgartirishlarim” -- > Bu buyruq barcha o’zgarishlarni git talqini tarixiga qoshadi.

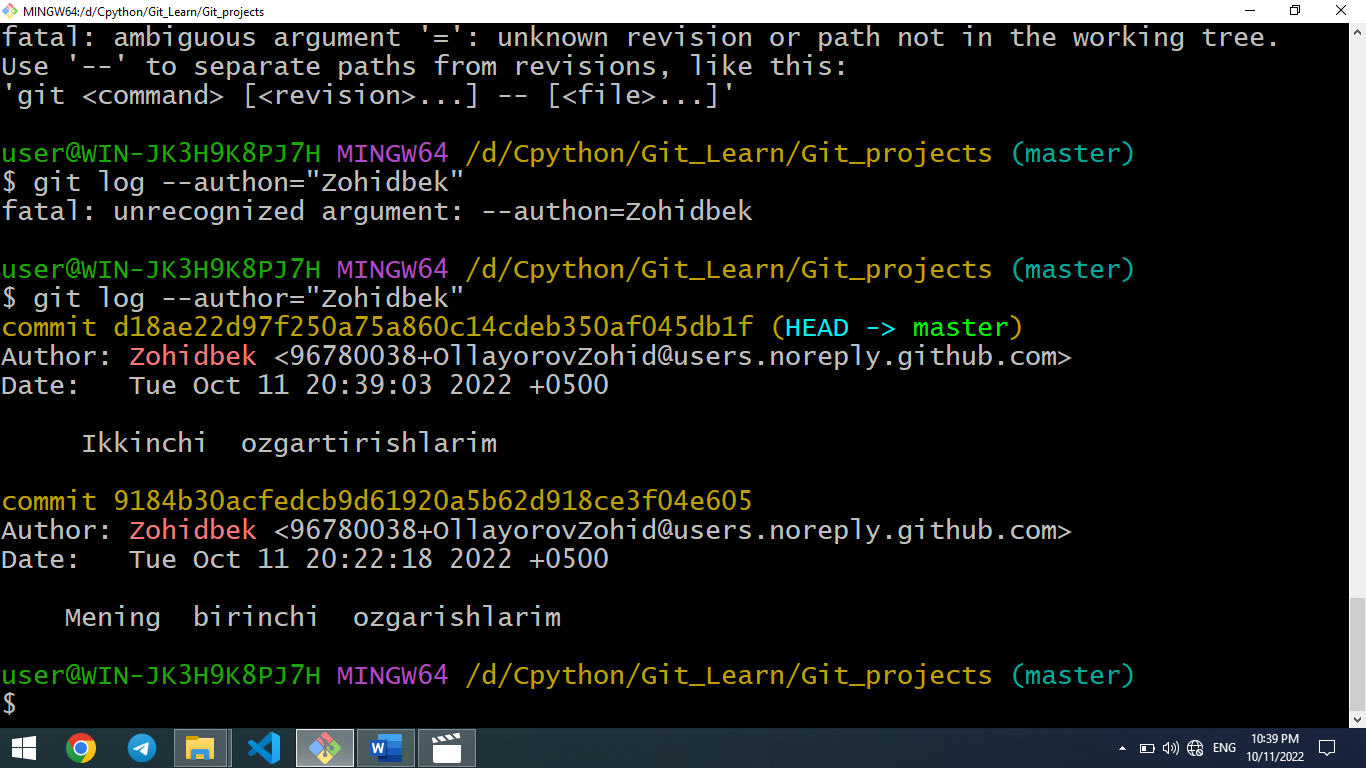


6. Git add VS Git commit - m “ozgarishlarim” -- > Git add buyrugi orqali biz yangi fayllarni gitda ro’yxatdan o’tkazib olamiz. Git commit -m “ozgarishlarim” buyrugu orqali talqinlar omboriga qoshib qo’yadi. Bunda o’zgartirish kiritilgan joyga qaytish mumkin bo’ladi. (Git commit qilingan yerga ).

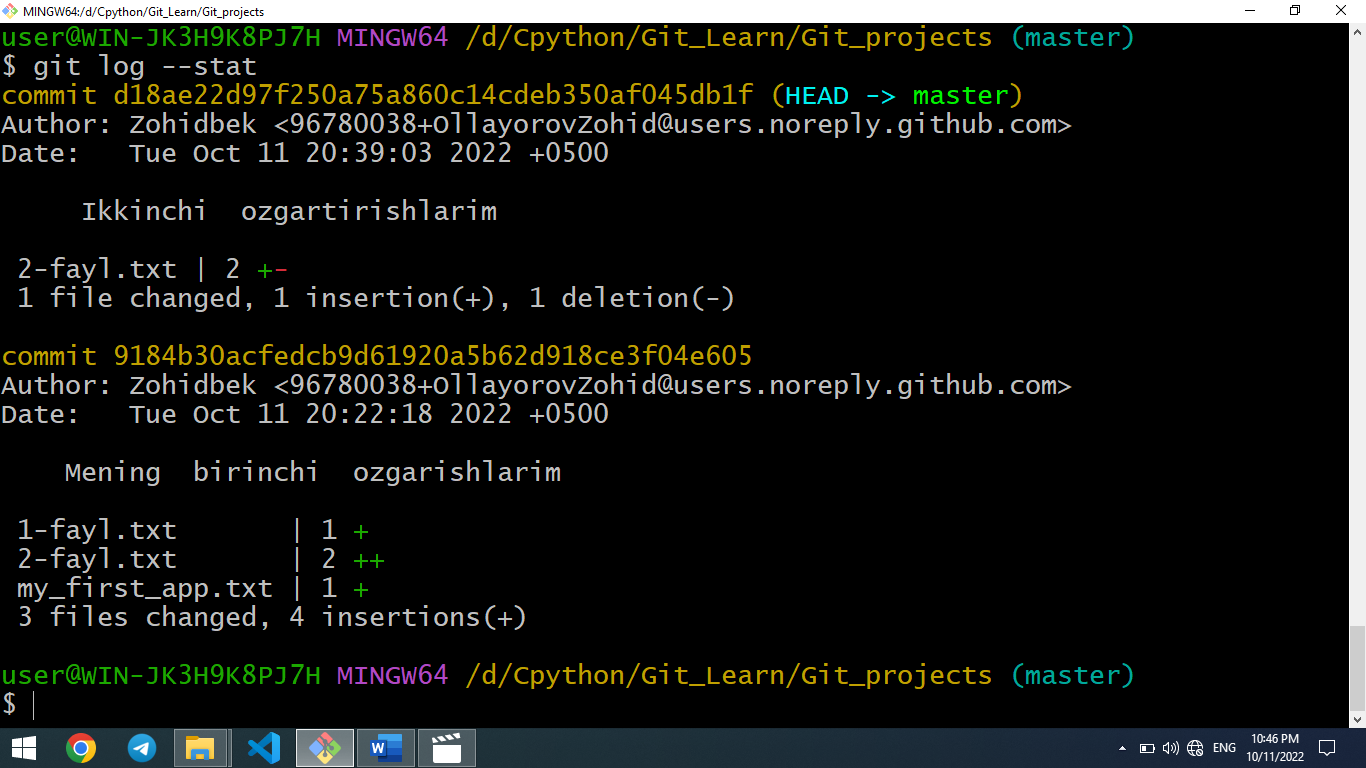
7. Git commit - am “Ikkinchi ozgartishlarim” -- > Bu buyruq git add . va git commit -m “ozgartirishlarim ” buyruqlarini bir vaqtda bajaradi. Yani fayllarni royxatdan otkazadi va talqinlar tarixiga qoshadi.Buni fiydasi shuki yuqoridagi ikkita buyruqni berish orniga bitta buyruqdan foydalanib , faylllarni gitdan royxatdan otkazish va talqinlar omboriga saqlash mumkin.



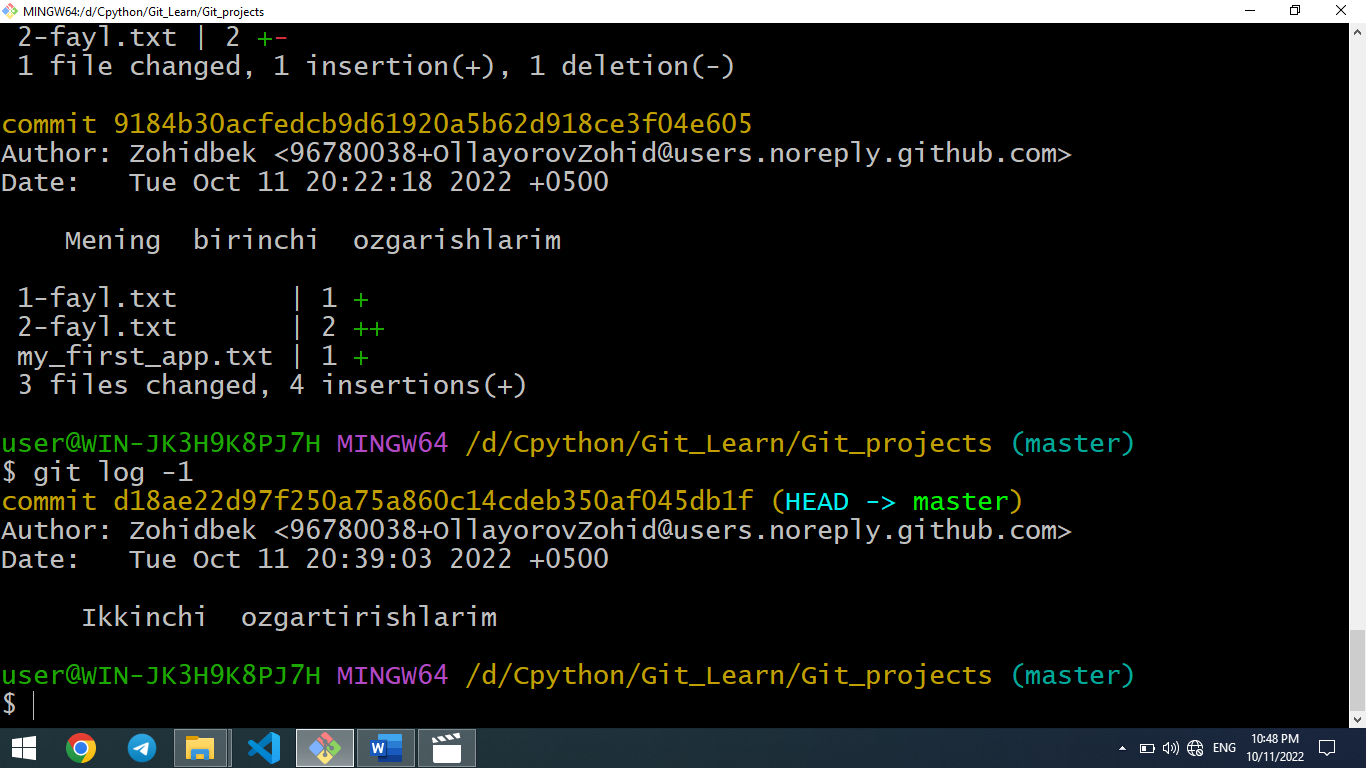
8. Git log -- > Bu buyruq gitni talqinlar omboriga qo’shilgan barcha ma’lumotlarni ko’rsatadi.

9. Git log --author=”Zohidbek” -- > Bu buyruq jamoa bolib ishlaganda asqotadi. Ya ‘ni authori Zohidbek bolganlarni qaytaradi. Masalan:

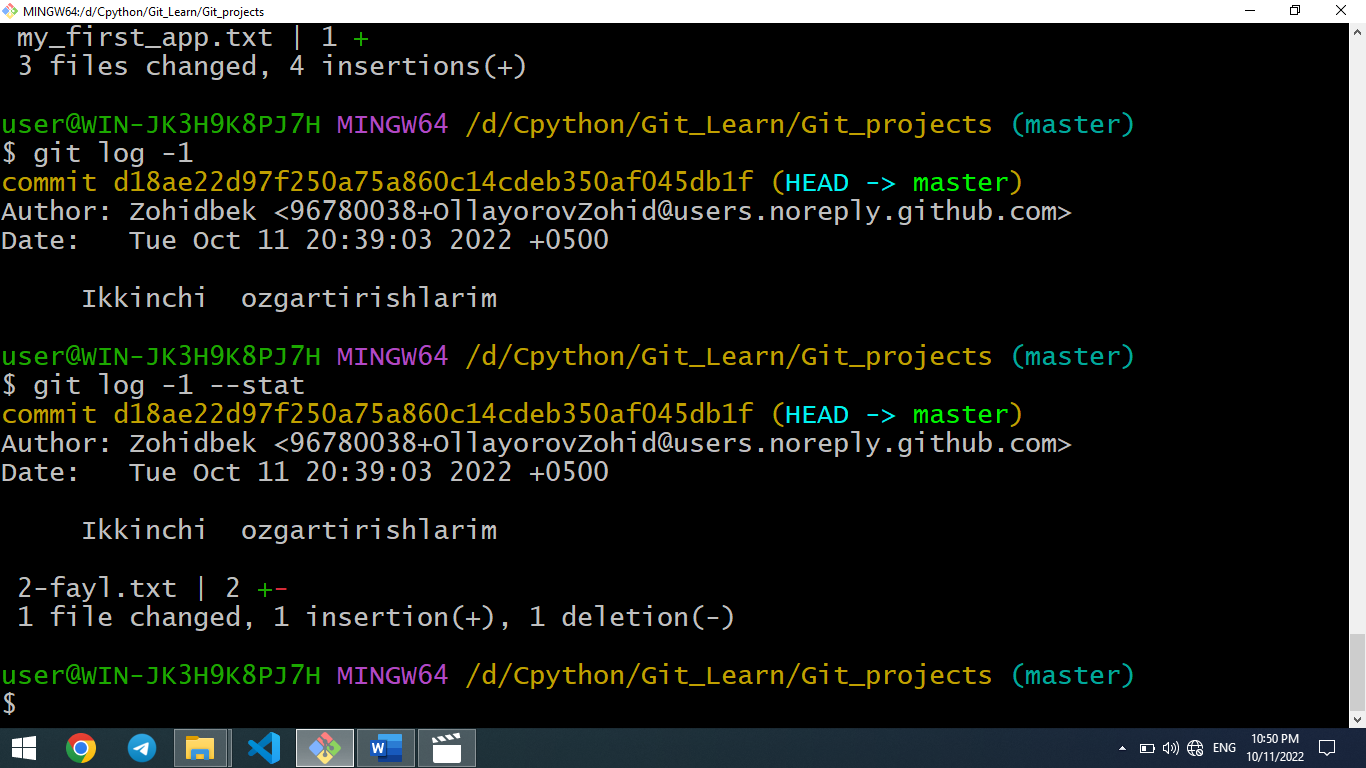
10. Git log --stat -- > Bu buyruq ham git log ni batafsilroq variant. Ya’ni bunda qaysi fayllarga qanchadan o’zgartirish kiritilganini ham ko’rsatadi. Mn:



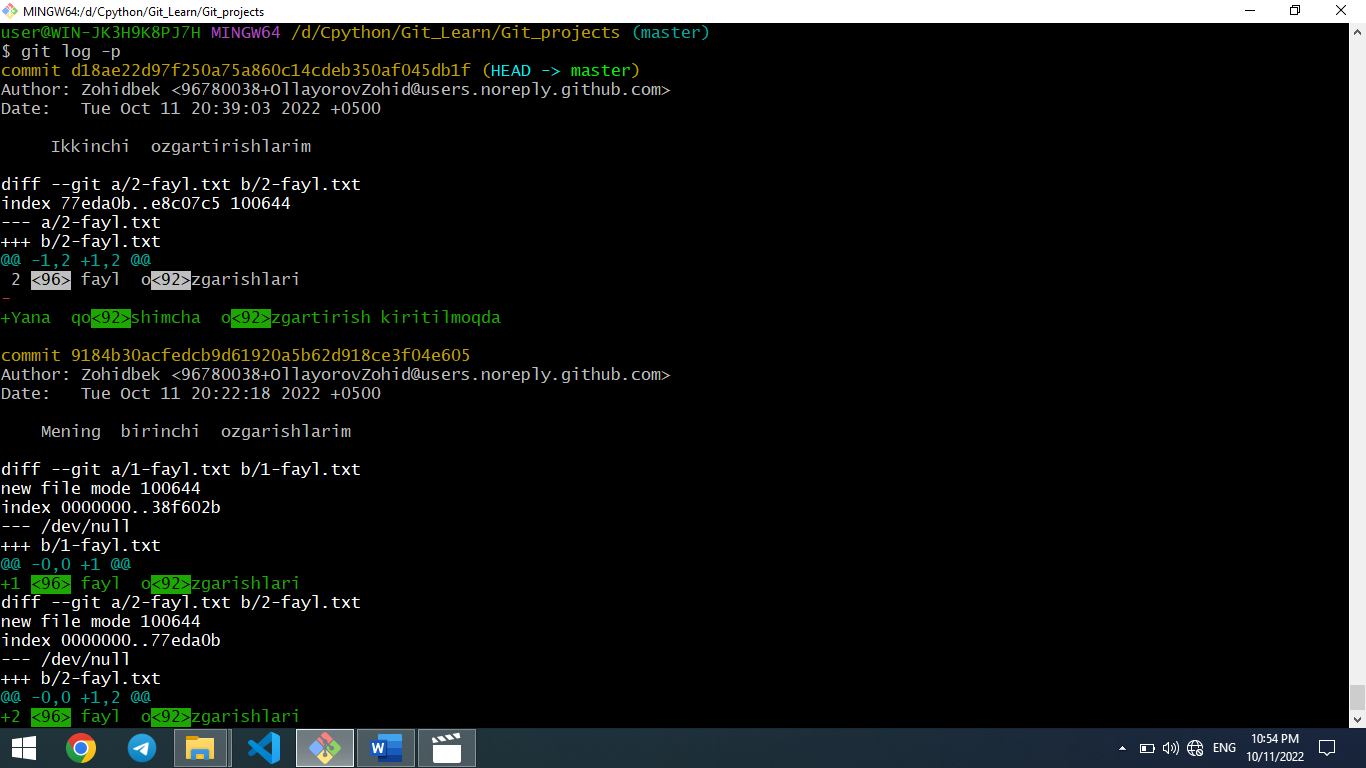
11. Git log -1 --> Bu buyruq Yuqoridan bitta o’zgarishni ko’rsatadi.



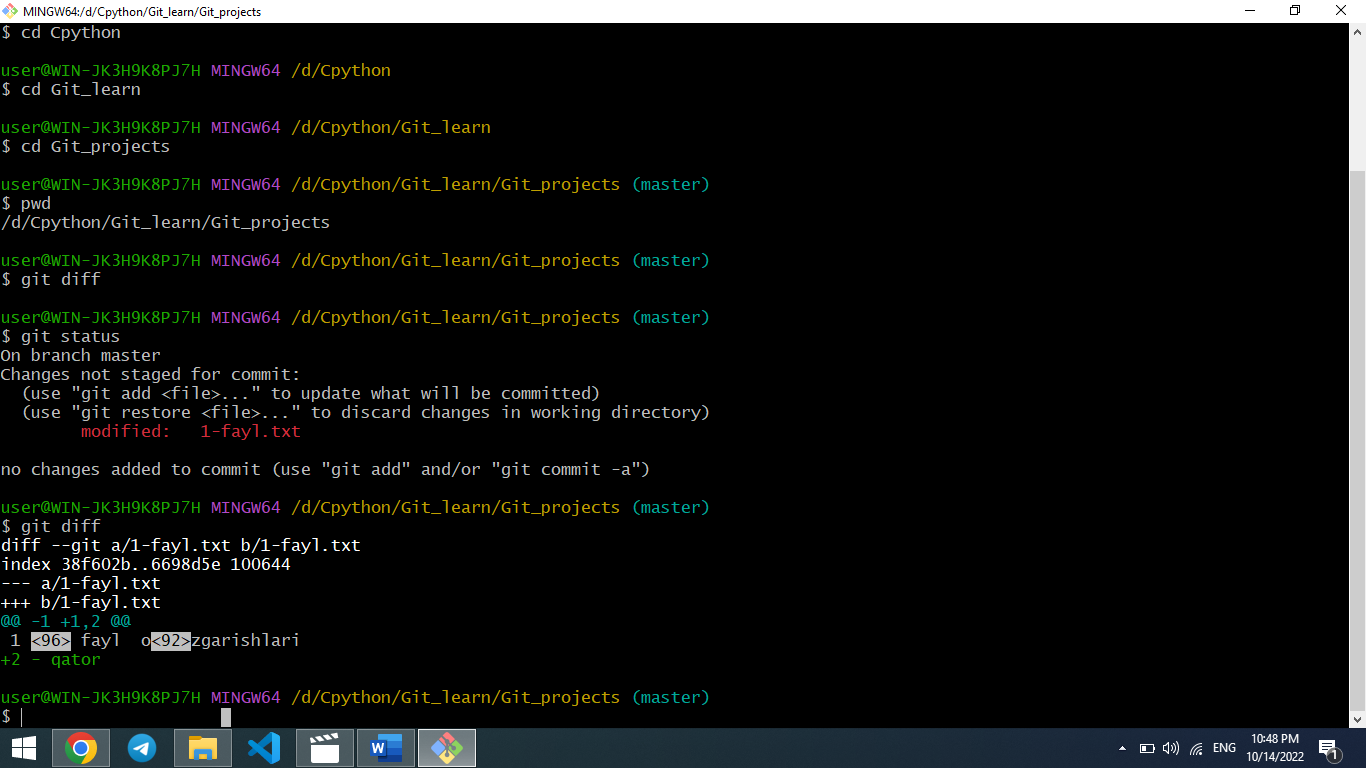
12. Git log -1 –stat 🡪 Ohirgi o’zgarishlarni batafsilroq ko’rish.

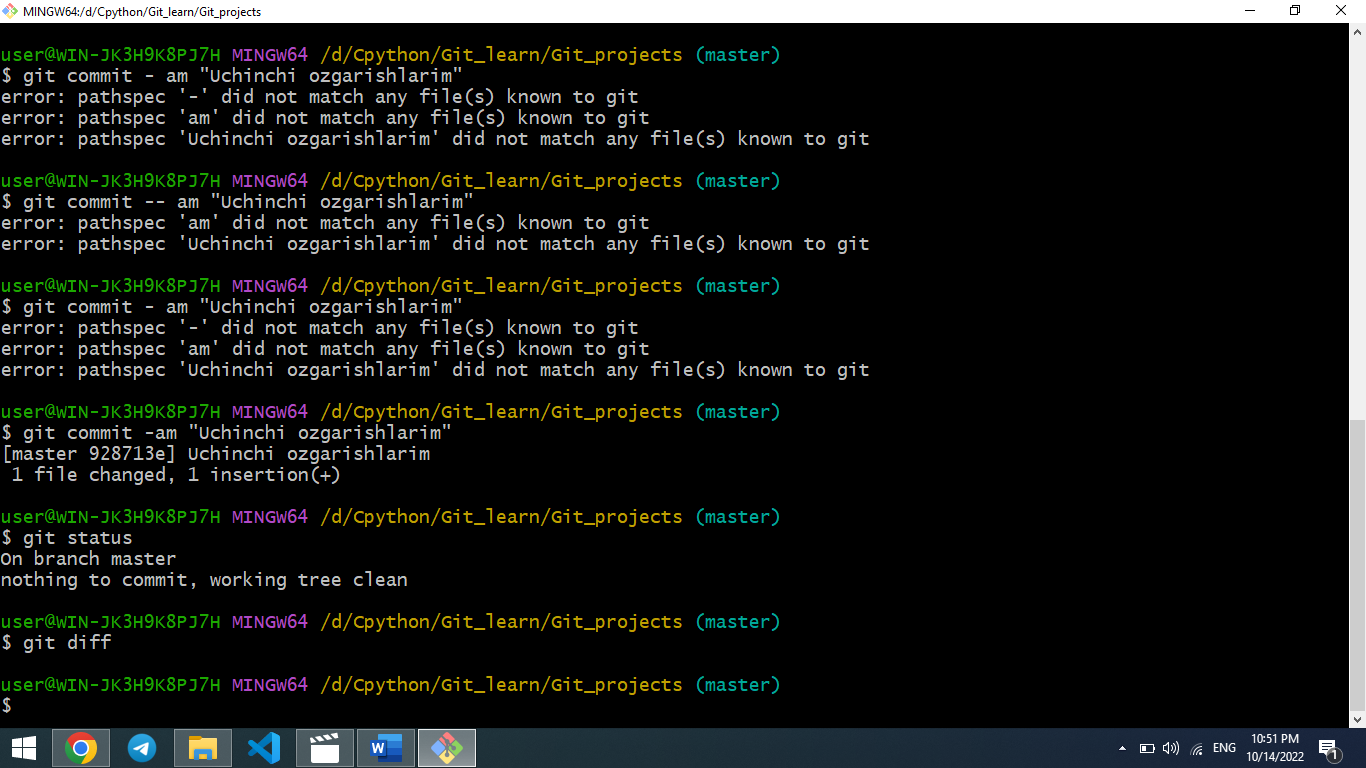


13. Git log -p -- > Bu git log –stat ga ham qaraganda batafsilroq ko’rsatadi. Bunda qaysi faylga qanday o’zgartirish kiritilganini va qaysi qatorga o’zgartirish kiritilganini ham ko’rsatadi.Mn:

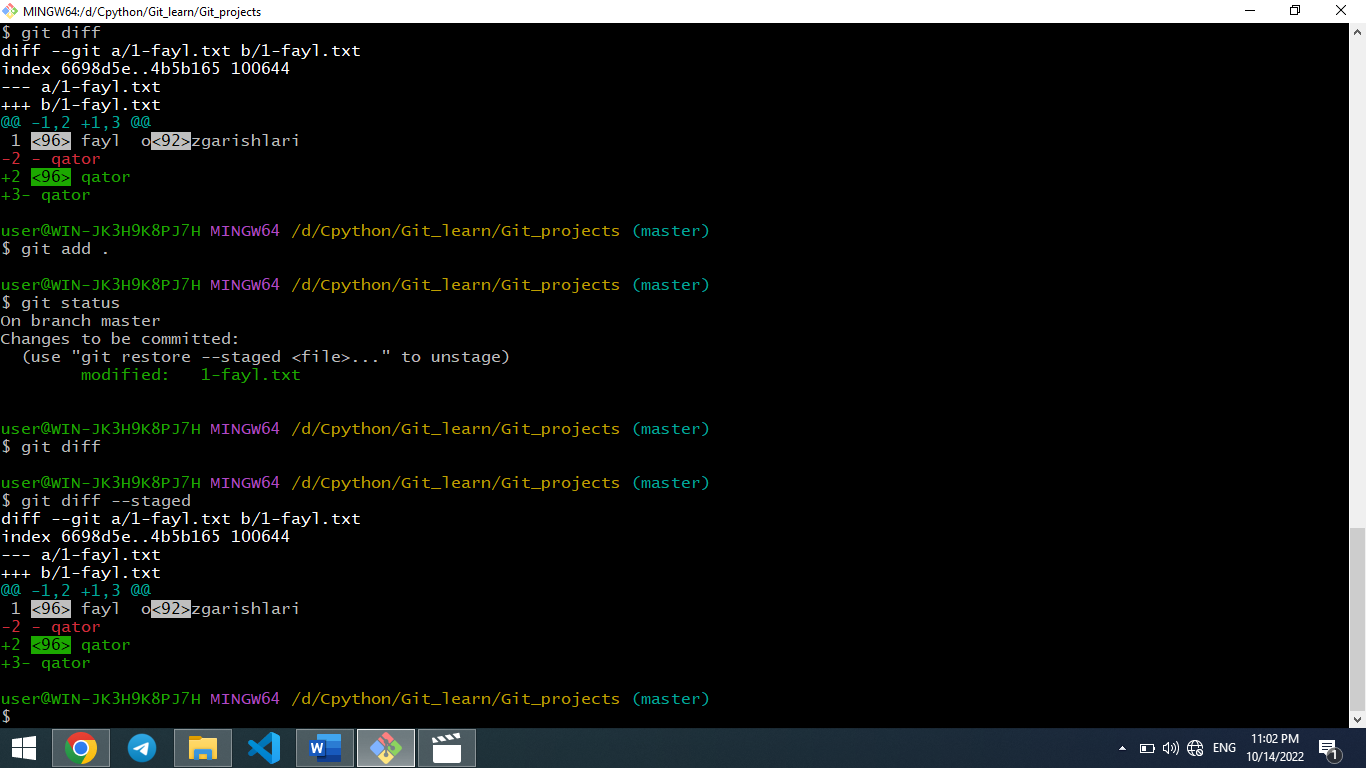


14.Git diff -- > Bu buyruq o’zgarishlarni taqqoslaydi . Misol uchun biz 1 faylni gitni talqinlar omboriga joyladik. Keyin shu faylga o’zgaritish kiritdik . Shunda git diff buyruguni bersak oldingi va keyingi fayllarni farqini solishtirib beradi.Agar o’zgartirish kiritgandan keyin ham talqinlar omboriga saqlasak git diff buyrugi hech nima qaytarmaydi. Mn:



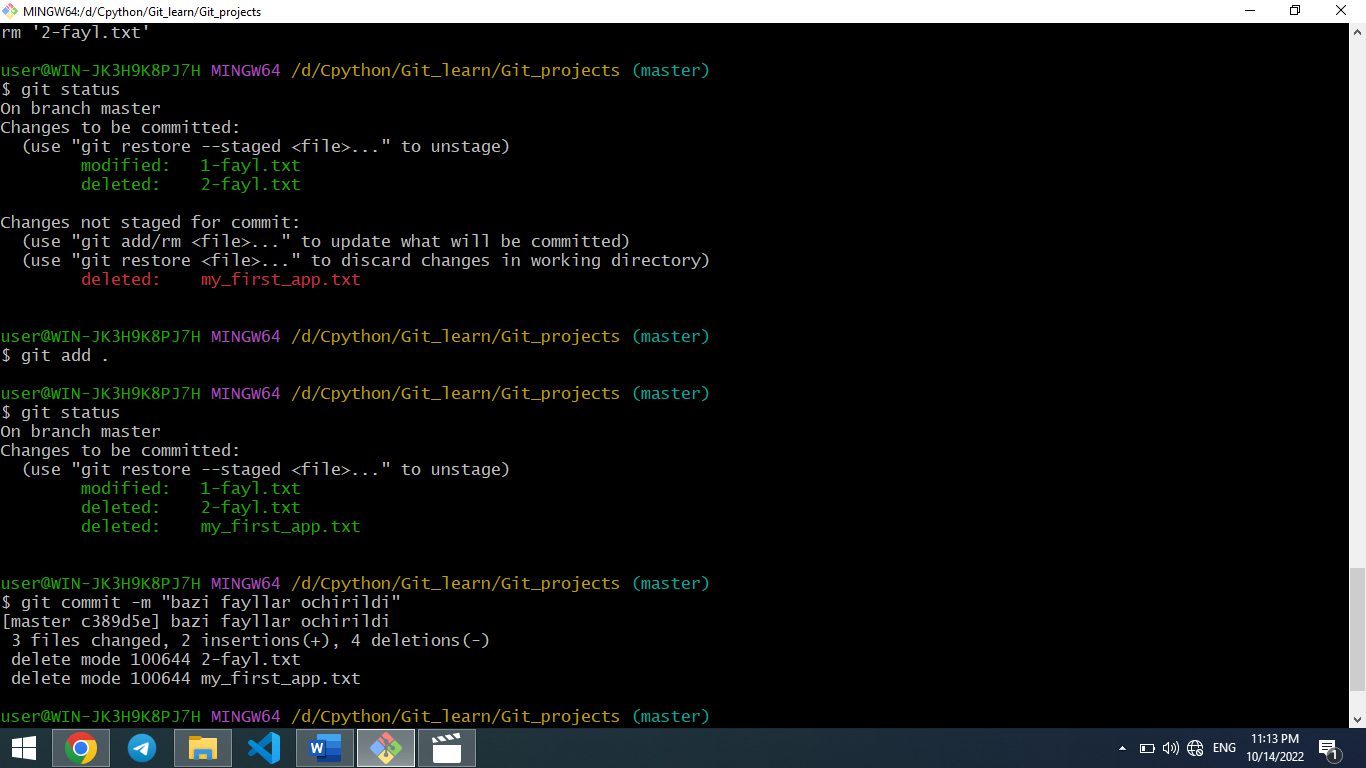


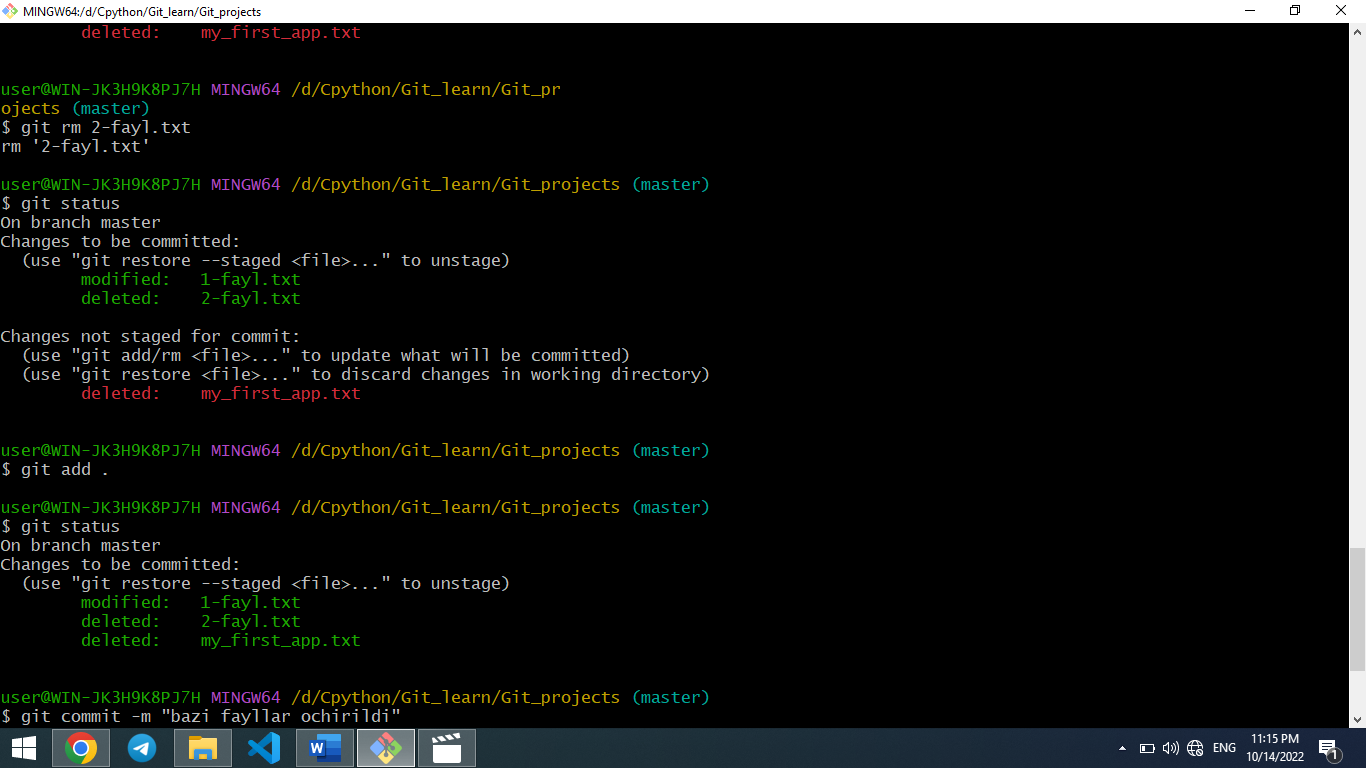
15. Git diff --staged -- > Bu buyruqni git diff dan farqi faylga o’zgartirish kiritilgandan so’ng gitdan royxatdan o’tkazdan keyin (git add qilgandan keyin) git diff -- staged buyruqi royxatdan o’tgan va talqinlar omboriga qoshilgan fayllarni farqini qaytaradi. Qisqa qilib aytganda faylga o’zgartirish kiritgandan so’ng git add buyrugini berib git diff desak hech nima qaytarmaydi. Buning o’rniga git diff -- staged dan foydalanish kerak. Mn:



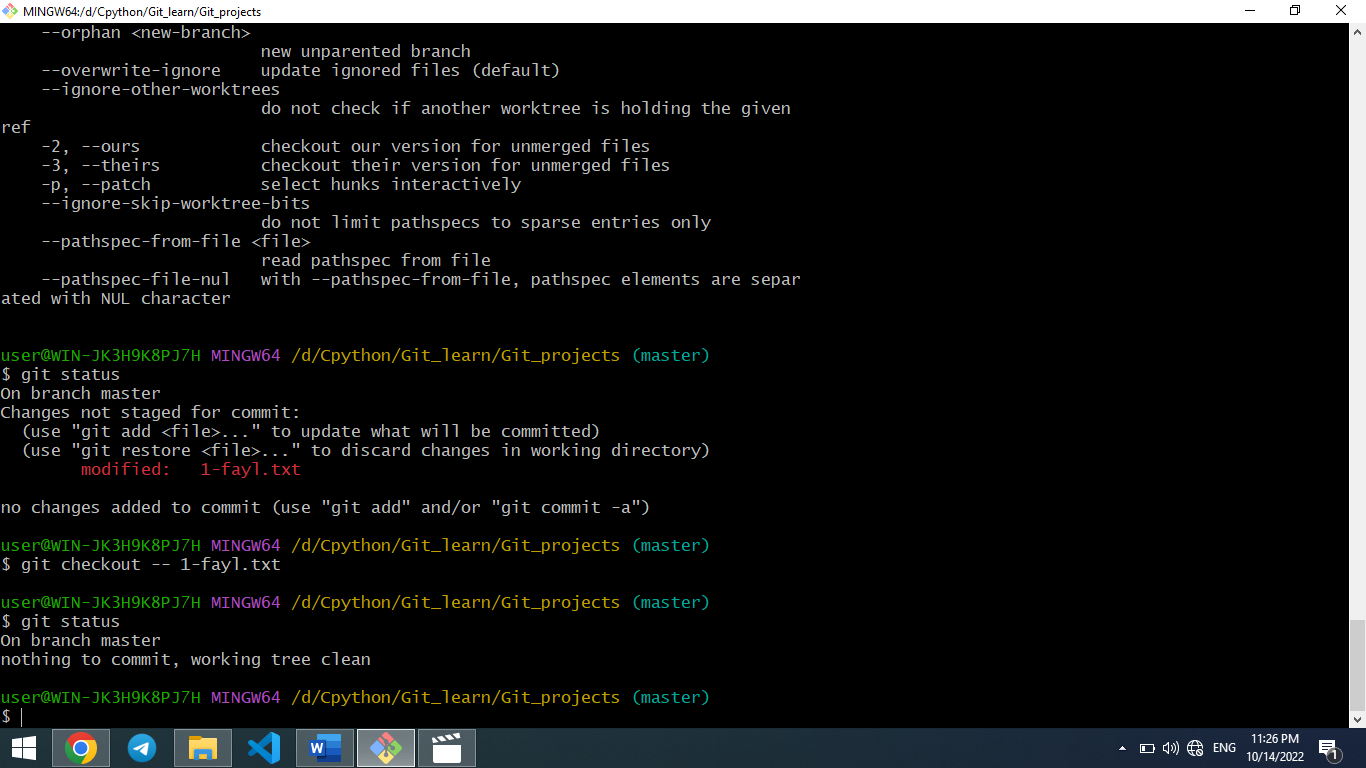
16. Git rm fayl\_nomi ( git rm 2-fayl.txt) -- > Biz papkadan birorta faylni o’chirsak , u fayl papkadan o’chadi lekin gitni ro’yxatidan o’chmaydi.

kabi bo’ladi. Uni gitdan ham ochirish uchun git add . buyrugudan foydalanishimiz kerak. Qisqa qilib aytganda papkadan faylni o’chirishimiz bilan gitdan o’chmaydi. Gitda bizdan soraydi shu fayl ochirilsinmi deb, git add . buyrugi orqali uni gitdan ham ochirishimiz mumkin.

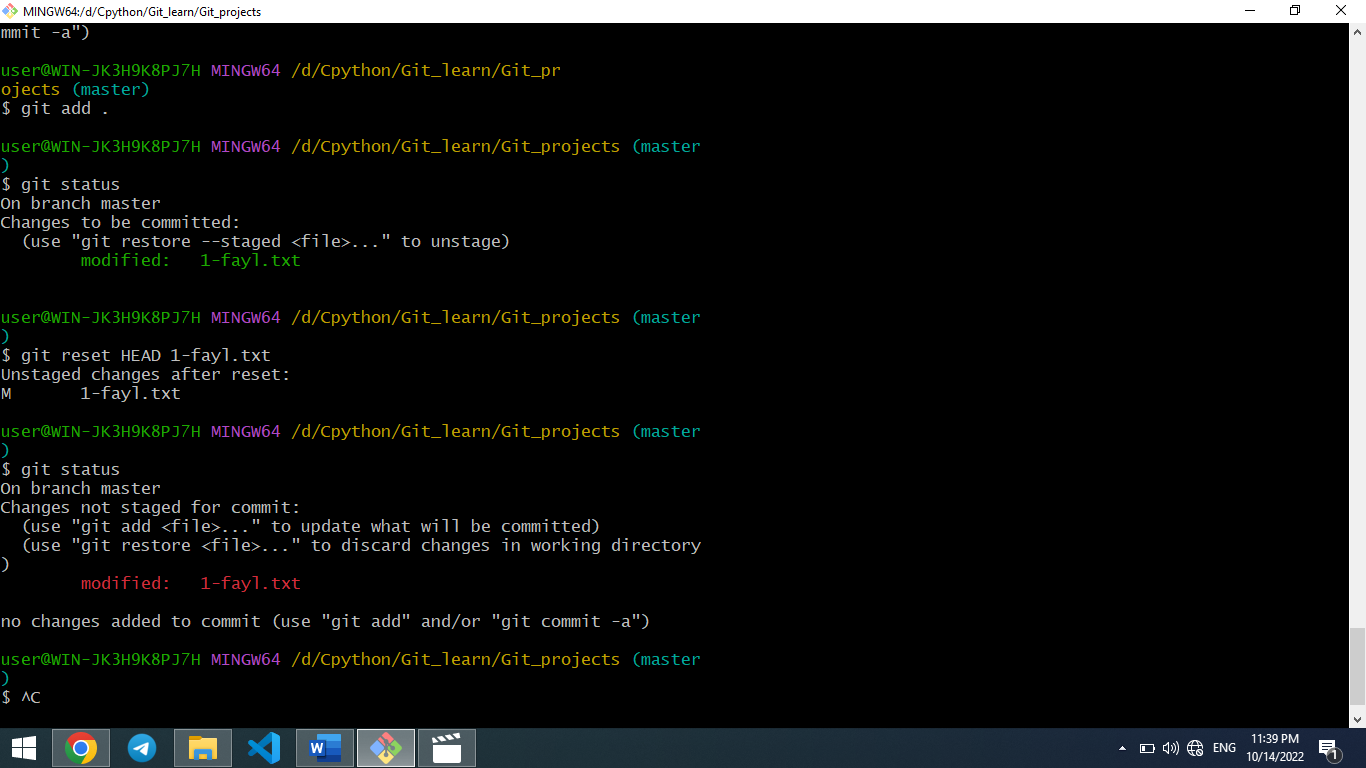
  
Bunda git rm fayl\_nomi orqali biz faylni ham papkadan , ham gitdan o’chirishimiz mumkin . Buni qulayliki 2 ta ishni bittada bajaramiz.



17. Git checkout -- fayl\_nomi ( git checkout -- 1-fayl.txt) -- > 1-faylga o’zgartirish kiritdik va git status buyrug’ini bersak gitda qizil bilan ro’yxatdan otmagan degan yozuvni chiqaradi. Bizga oldingi holiga qaytish kerak bolib qoldi(yani faylga ozgartirish kiritganimizdan oldingi holatga) . Bunda biz . Git checkout -- fayl\_nomi ( git checkout -- 1-fayl.txt) buyrugidan foydalanamiz. Keyin ozgartirish kiritilgan faylimiz ham asliga qaytadi, git status buyruguni bersak ham hech qanday o’zgarish bo’lmaydi.



18. Git reset HEAD fayl\_nomi (git reset HEAD 1-fayl.txt) -- > Biz biror faylga o’zgartirish kiritdik va uni gitdan ro’yxatdan o’tkazdik(git add . orqali). Endi shu faylni ishchi maydonga qaytarmoqchimiz (ya’ni gitdan royxatdan otgan faylni orqasiga qaytarmoqchimiz) buni git reset HEAD 1-fayl.txt . Bu gitdan royxatdan o’tgan faylni ishchi maydonga qaytarib otkazadi (yani royxatdan otkazmaydi qisqasi orqasiga qaytaradi) . yana git checkout – fayl\_nomi orqali biz faylni avvalgi holatiga keltirishimiz mumkin ekan.Git checkout fayl\_nomi va git reset HEAD fayl\_nomi ikkisini farqi git reset HEAD fayl\_nomi git add . bolgan faylni ishchi maydonga (yani orqasiga ) qaytaradi. Git checkout – fayl\_nomi esa ishhi maydondan faylni asl holatiga qaytaradi.

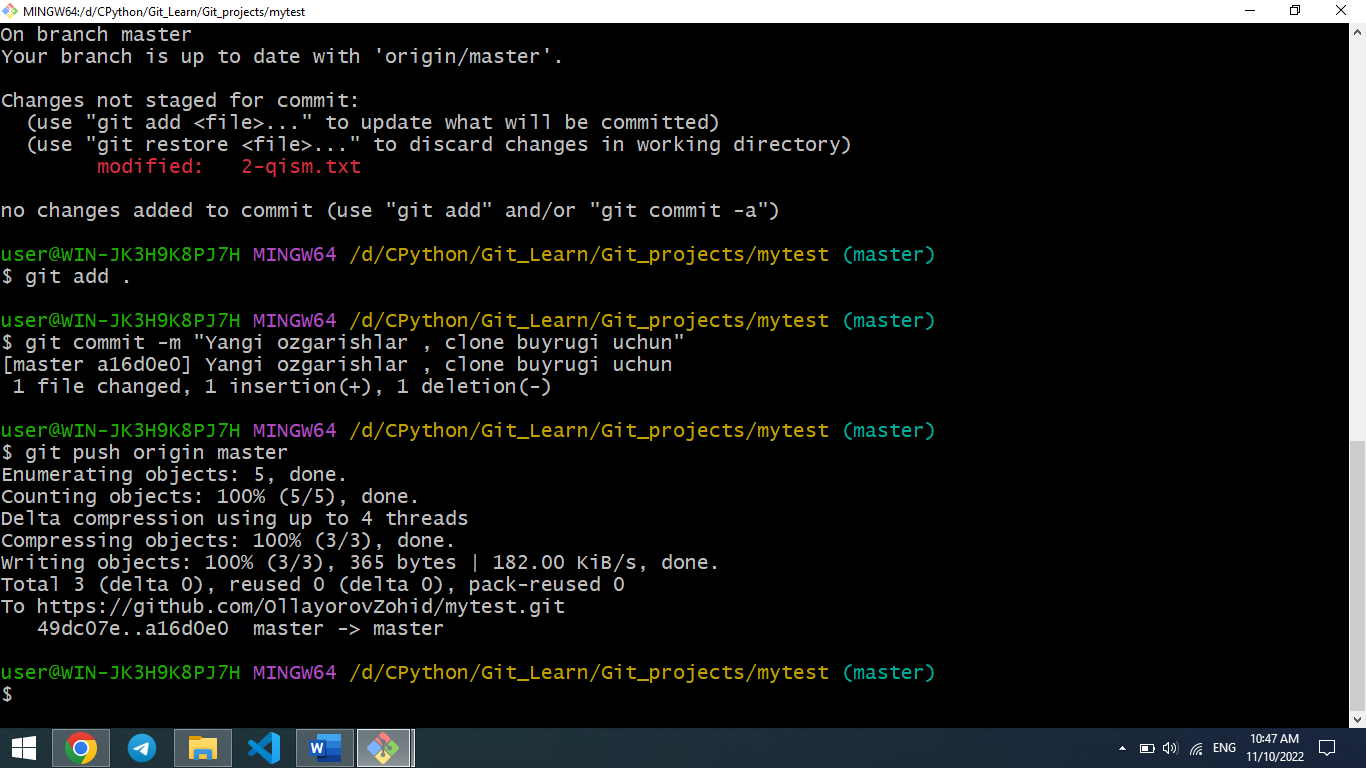


19. Git remote add origin https://github.com/OllayorovZohid/mytest.git --> BU buyruq gitdan commit qilingan fayllar bilan githubdagi mytest repositorysini boglaydi.

20. Git push -u origin master -- > Bu buyruq orqali esa gitdan commit qilingan fayllarni githubga yuklaydi.

21.Git clone <https://github.com/OllayorovZohid/mytest.git> -- > Bu orqali biz githubdagi mytest nomli repositoryni yuklab ( yani clone qilib) olamiz. Git clone buyrugini qayerda bersak (yani qaysi papka Ichida ) fayl ham shu yerga clone boladi. Clone qisqacha qilib aytganda githubdan faylni yukklab olish.

22. Git clone VS git Download -- >1. Faylni Git download qilib olsak , bu fayl kompyuterimizga tushadi va unga hech qanday o’zgartirish kirita olmaymiz. 2. Git clone buyrugi orqali yuklab olsak esa faylga o’zgartirish kiritib, git add . orqali gitdan royxatdan otkazamiz va git commit -m “ozgarishlar” orqali talqinlar omboriga joylaganimizdan song git push origin master orqali Githubdagi faylga ozgartirish kiritib ham bilamiz . Ikkalasini farqi shunda .

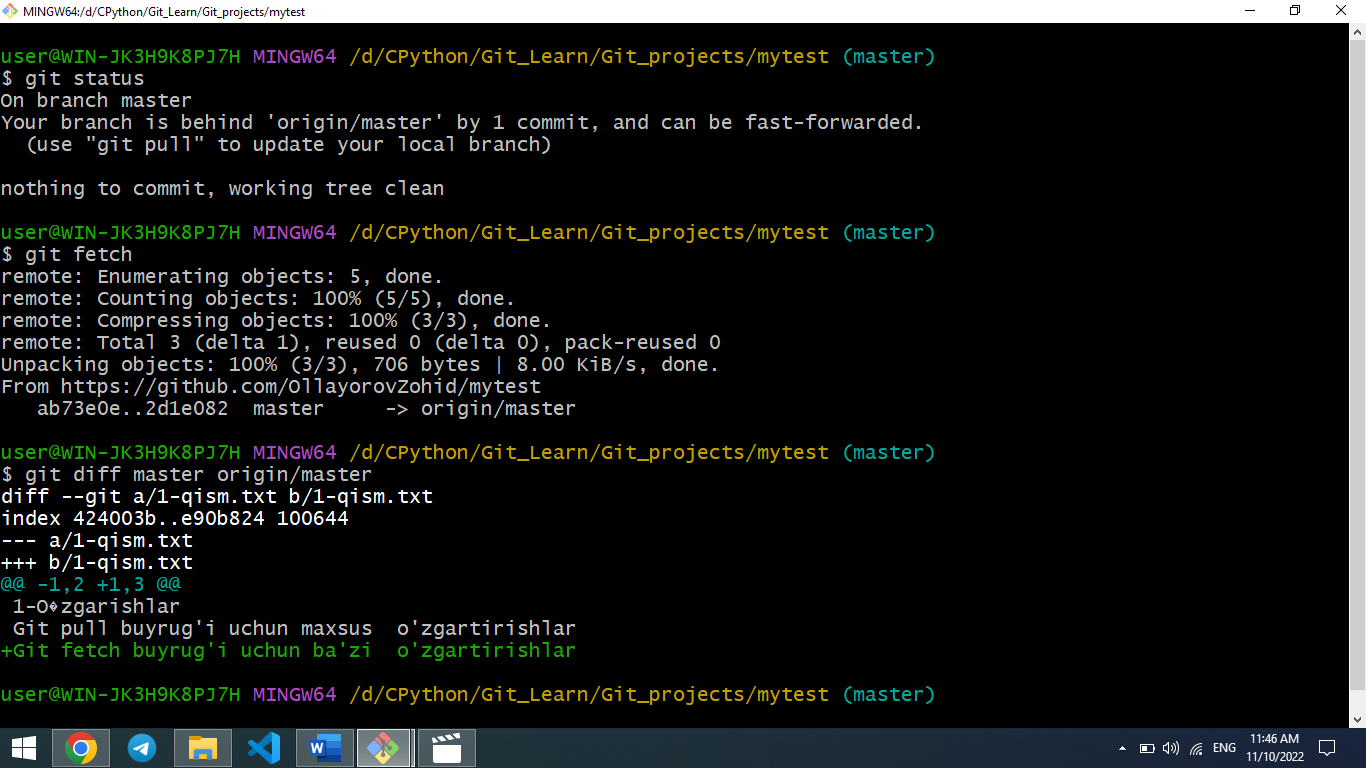


Ozgartirish kiritildi va githubga yuklandi.

23.Git pull origin master -- > **Git Pull bu Git [buyrug'i](https://www.freecodecamp.org/news/git-pull-explained/" \t "_blank) bo'lib, mahalliy ombordan tarkibni olish va yuklab olish va shu tarkibga mos kelish uchun masofadan boshqarish pultidan omborning mahalliy versiyasini yangilash uchun ishlatiladi.Qisqacha qilib aytganda Markaziy ombordagi o’zgarishlarni mahalliy omborga ko’chirish. Masalan jamoa bo’lib ishlayotganda biror dasturchi faylga o’zgartirish kiritdi va biz uni yuklab olishimiz kerak.Git pull origin master --> buyrugi orqali biz shu faylni o’zgartirishlar kiritilgan holda yuklab olamiz.**

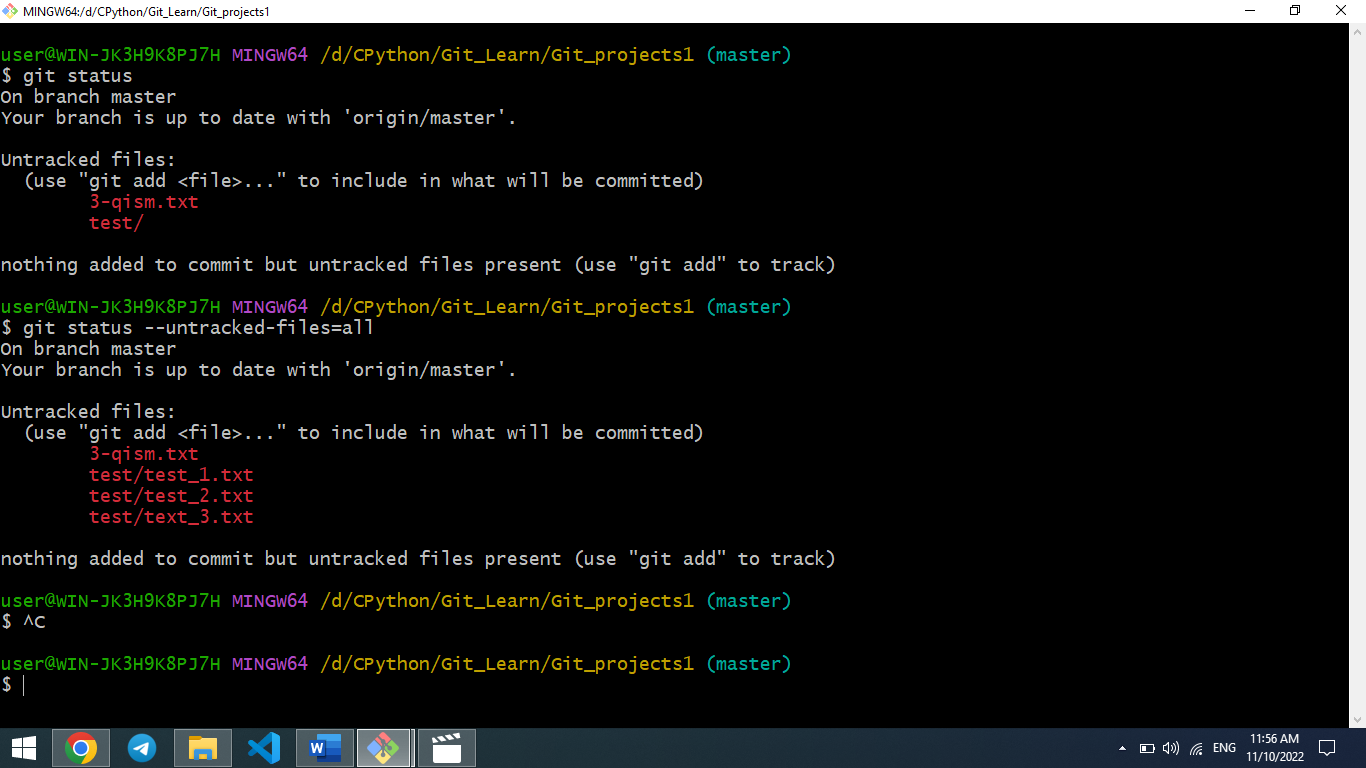
**24.Git fetch -- > Markaziy ombordagi o’zgarishni mahalliy omborga ko’chiradi faqat soliwtiriwimiz manosida , git pull buyrugi togridan – togri faylga o’zgartirish kiritilsa , git fetchda bu o’zgarishlarni qilmaydi. Lekin git fetchni farqi markaziy ombordan o’zgarishlarni olib keladi faaqat u bilan soliwiriw imkonini beradi. Ozgarishlarni mahalliy omborga olib kelish uchun esa git pull buyrugidan foydalanishimiz kerak.Git fetch ozgarishlarni ko’rish uchun ko’chirib olinadi lekin faylga oz’garishlarni olib kelmaydi . U bilan faqat solishtirsa boladi.**

**100 marta aytgandan ko’ra 1 marta ko’rgan yaxwi deganlaridek, git fetch buyrugini ishlatib ko’ramiz.**



Agar shu yerda git fetch buyrugini bermasdan oldin git diff buyrugini bersak hech nima qaytarmaydi.

25. git status --untracked-files=all -- > Bu buyruq git statusdan farqli o’laroq Barcha fayllarni korsatadi. Papka ichidagi papkalarni ham barchasini korsatadi. Git statusda esa papka ichidagi fayllarni korsatmaydi.



26. .Gitignore -- > Inobatga olmaslik. Ya’ni git status qilganda hamma fayllarni ham chiqaradi. Agar .gitignore ni Ichida yozilgan (masalan: text\_3.txt) fayllarni inobatga olmaydi.Qisqasi .gitignore ni Ichida yozilgan fayllarni inobatga olmaydi. Masalan:

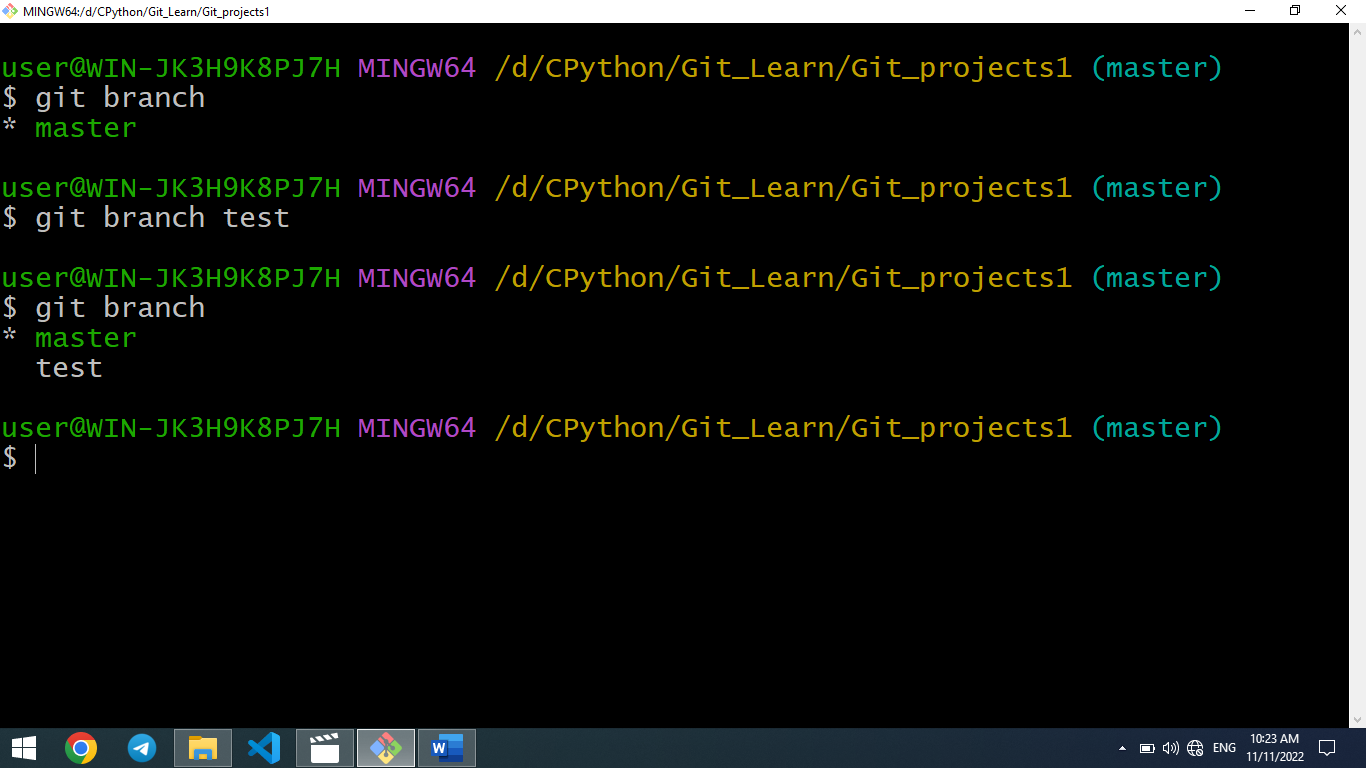
test/\*.txt 🡪 Bu test papkani ichidagi barcha .txt fayllarni inobatga olinmasin degani.

!test/\*.txt 🡪 Bu test papkani ichidagi barcha .txt fayllarni inobatga olsin degani.

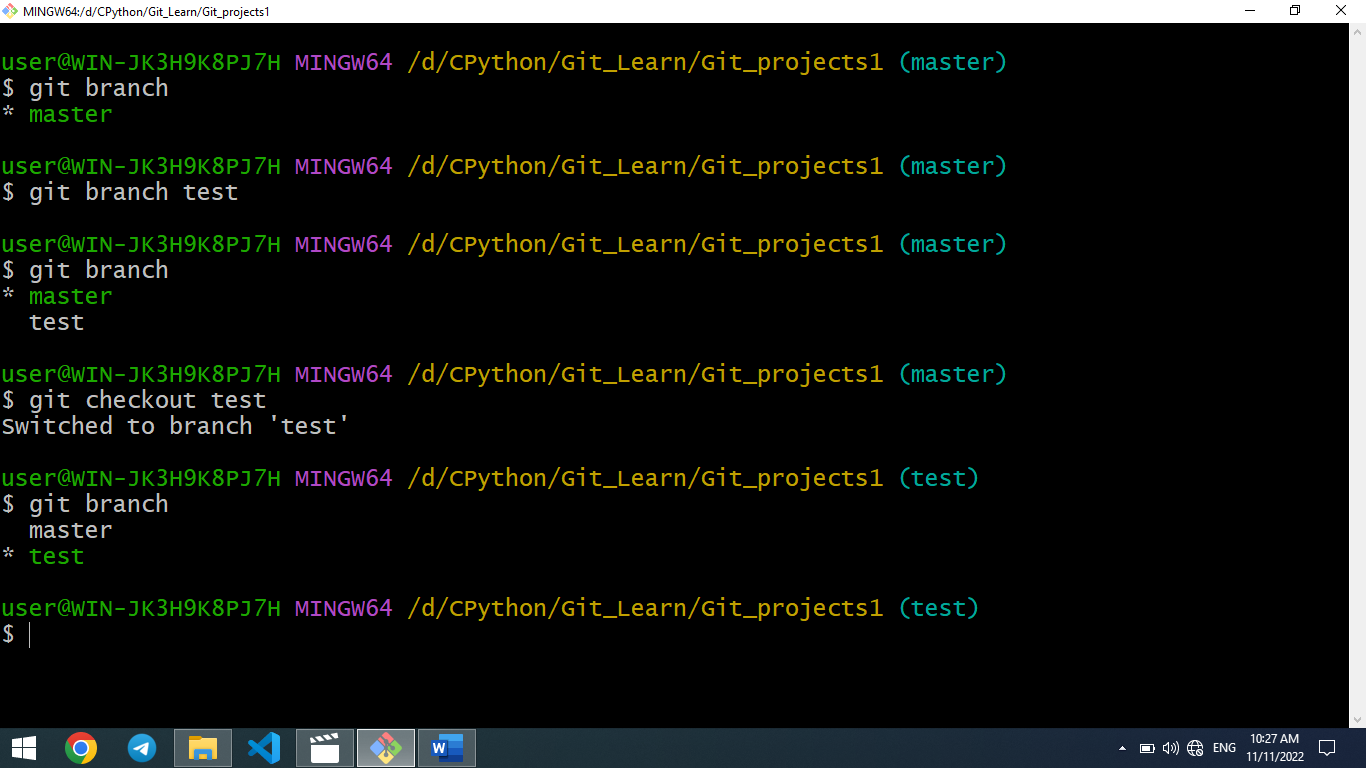
\*.txt 🡪 faylimizdagi barcha .txt fayllarni inobatga olmaydi

27. Git branch -- > Bizda mavjud bo’lgan barcha tarmoqlarni chiqaradi.

28. Git branch test -- > test nomli yangi tarmoq hosil qiladi.

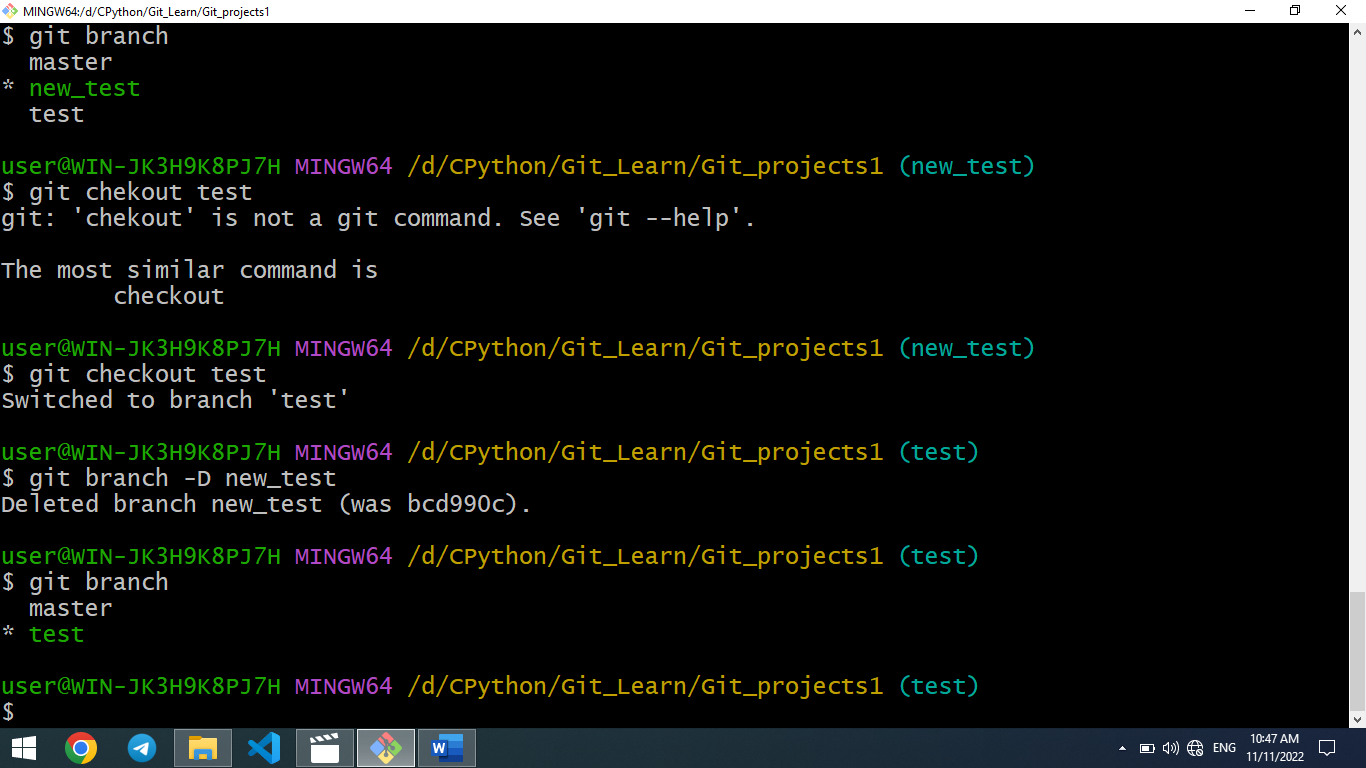


29. Git checkout test -- > test nomli tarmoqqa o’tadi.

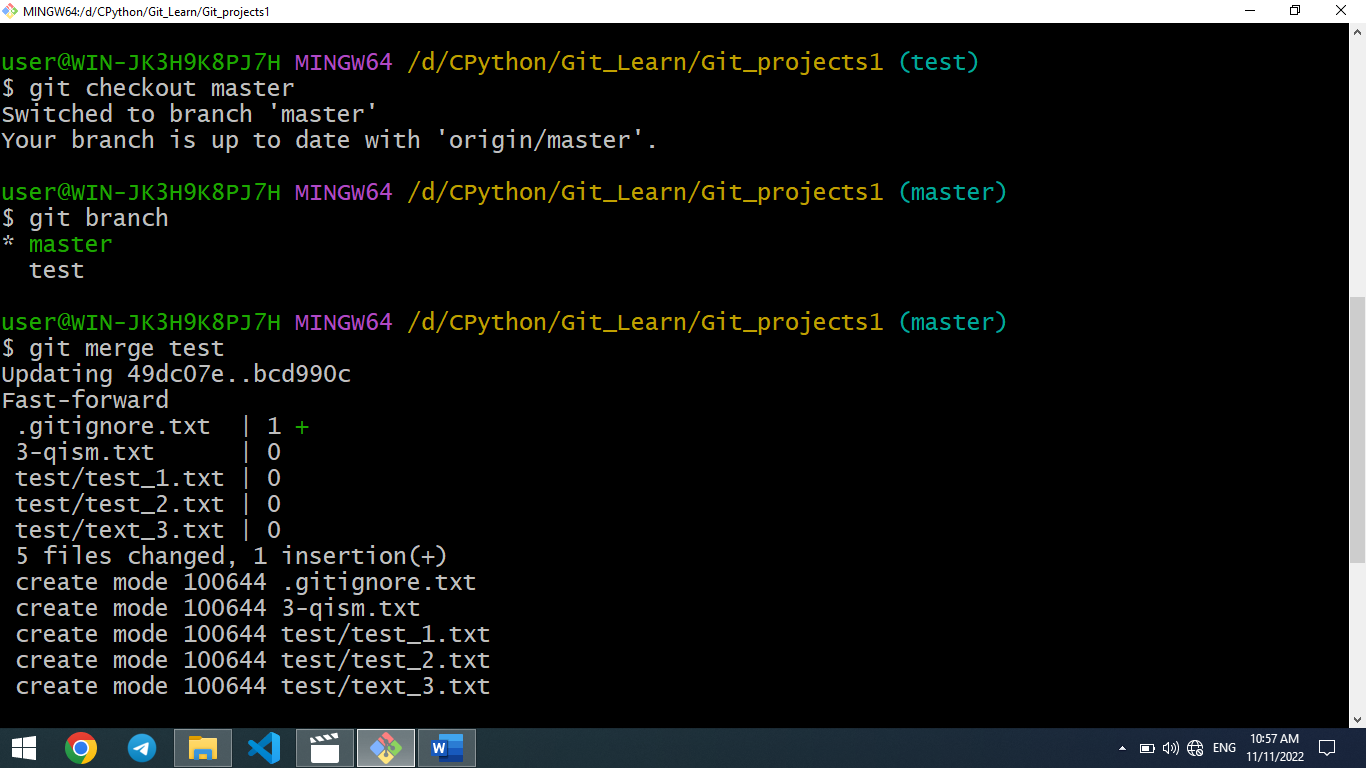


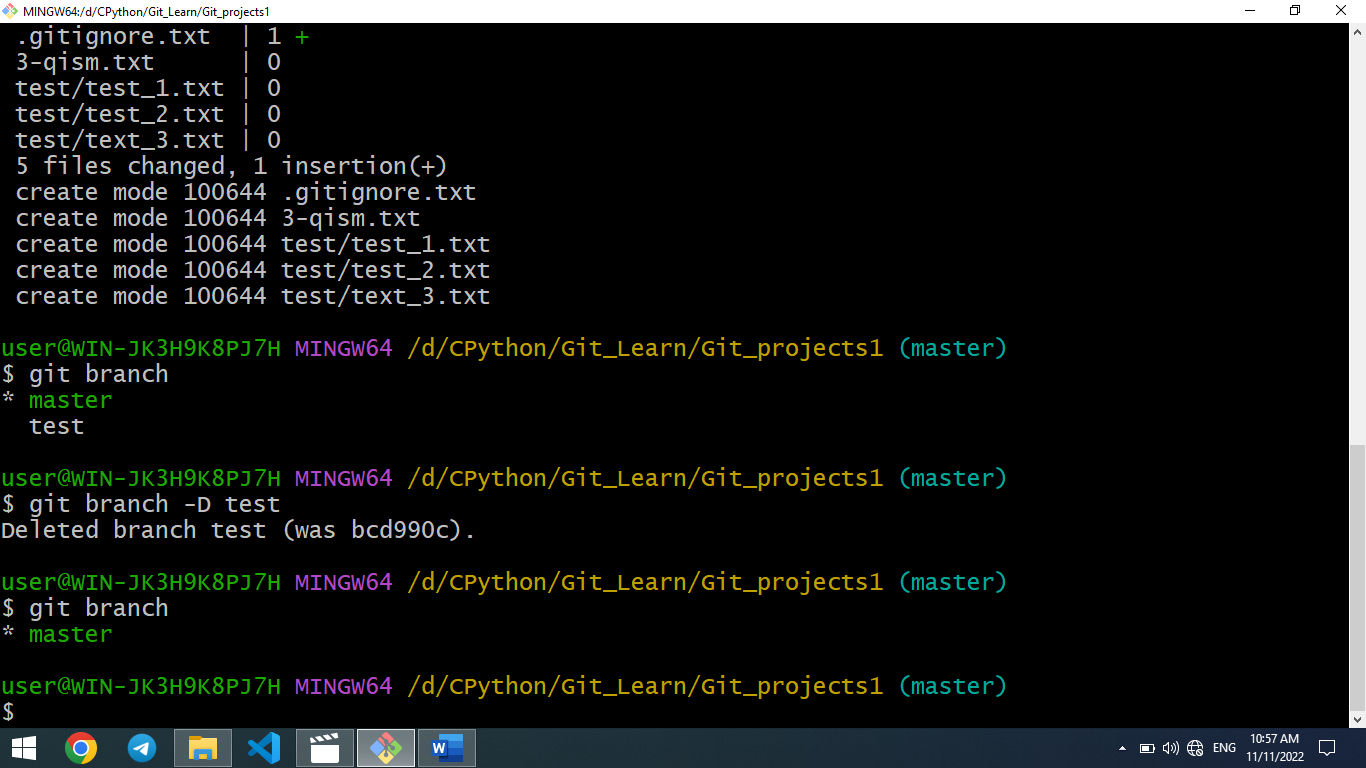
30. Git checkout -b new\_test -- > Yangi new\_test nomli branch hosil qiladi vas hu tarmoqqa o’tadi. Ya’ni yuqoridagi 2 ta ishni 1 ta buyruq orqali qilamiz. New\_test nomli tarmoqda avval qaysi tarmoqda turgan bo’lsa shu tarmoqni fallarini olib o’tadi.

31. Git branch -D new\_test -- > new\_test nomli tarmoqni o’chiradi.Agar siz new\_test nomli tarmoqda turgan bo’lsangiz error qaytaradi. Yaniki siz turgan tarmoqni o’chirib bilmaysiz. Boshqa tarmoqqa o’tib keyin o’chirishiningiz mumkin.



32. Git merge test -- > Hozirgi turgan tarmoqimizga (ya’ ni masterga) test nomli tarmoqli birlashtiradi. Yani bu ikkita tarmoqni birlahtiradi. Bizni kodda test va master tarmoqlari master tarmoqiga birlashtirildi. Endi test nomli tarmoqni o’chirib tashlasak ham bo’ladi(Chunki uni bizga keragi yo’q.)





33.Coflict -- > Gitda conflict tushunchasi. Bu ikkita tarmoqdagi aynan bitta faylni bir xil qatorini o’zgartirganda hosil bo’ladi. Yani git qaysi o’zgarishini qilishini bilmaydi va sizdan so’raydi(yani dasturchidan) siz shunda qaysi ozgarish kerak bolsa shuni berasiz. Bunday holatlar jamos bilan ishlaganda kop uchrab turadi. Hohlasangiz ikkita ozgarishni ham saqlashingiz mumkin.

